

Webアプリケーション モジュール (WAM)

[始める前に](#)

[WAMの概論](#)

[WAMの構造](#)

[主要なトピック](#)

[アドバンスド・トピック](#)

[WAMアプリケーションの実行](#)

[WAMとWEBEVENTの相互運用](#)

[テクノロジー・サービス](#)

[XHTMLテクノロジー・サービス用のウェブレット](#)

[jQueryMobileテクノロジー・サービス用のウェブレット](#)

[WAM チュートリアル \(英語\)](#)

[付録A. XSLとXMLの準拠](#)

[付録B. WAM XML構造](#)

[付録C. 廃止されたウェブレット \(英語\)](#)

[WAM ウィザード](#)

エディション日付: 2013年7月30日

© LANSA

始める前に

- LANSA for the WebのWAMテクノロジーは、LANSA開発環境の拡張機能です。リポジトリとRDML/RDMLXのスキルがあるならば、それらをWebベースのアプリケーションを構築するのに利用することができます。
- 開発作業を始める前に、WAMテクノロジーをサポートするLANSA for the Webシステムを正しくインストールして構成する必要があります。
- 正しく構成したWebサーバーと正しく構成したLANSAアプリケーション/データ・サーバーが必要です。
- 開発システムにInternet Explorer 6.0またはそれ以上、そしてMSXML Core Services 6.0 XML Parserがインストールされていなければなりません。MSXML parserは、Visual LANSAのインストール・プロセス中に自動でインストールされます。
- LANSA for the Webのインストールと構成の詳細については、『Windows LANSAインストールガイド』を参照してください。
- WAMを初めて使用する場合は、WAMの全てのコンポーネントの概略を把握できる「[WAMの概論](#)」、およびWAMコンポーネントの詳細な理解ができる「[WAMの構造](#)」を読むことをお勧めします。また、WAMのチュートリアルを行うこともお勧めします。WAMチュートリアルについては、「[WAMチュートリアル](#)」を参照してください。
- HTMLブラウザ・ベースのアプリケーションを短時間で開発するための便利な機能として、ウェブレットが用意されています。独自のウェブレット、もしくは複雑なHTMLページを作成するには、XSL、HTML、JavaScriptのスキルが必要になります。
- XML WEBROUTINEドキュメントの変換に使用するXSLは、W3Cの標準規格であるXSL 1.0仕様に準拠しています。詳細についてはXSL 1.0を参照してください。

1. WAMの概論

この章では、LANSAのWAM構造の非常に簡単な概略および各コンポーネントを説明します。

この概論では、各テクノロジー、コマンド、パラメータの詳細については記述されていません。むしろ、WAMを使用するのに自信をもって取り組むことができるように全体的な概観となっています。このガイドは、読者がVisual LANSAアプリケーション開発環境の作業知識をお持ちであることを前提としています。

なぜLANSAはWAMテクノロジーを開発したのか？

理由は多くありますが、主なキー・ポイントは次の2つです。

1. ウェブ・テクノロジーがとても非常に早く発展しているからです。
HTMLはもうウェブのコンテンツを提供する唯一の方法ではありません。新しいテクノロジーの出現に合わせて、LANSAのWebアプリケーション開発も迅速に適合させていかなければなりません。
2. アプリケーションの開発が今後コンポーネント・ベースになることは明らかです。ですからLANSAのWebアプリケーション開発もコンポーネント・ベースの技術をフルに活用できるようにすることが必要です。

1.1 WAMとは?

Webアプリケーション・モジュール(WAM)は、ユーザーインターフェースをXMLの形式で、通常はウェブ・ブラウザを介して配布するアプリケーションを構築するためのLANSAのソリューションです。

WAMはアプリケーション・ロジック層とプレゼンテーション層という、2つの異なる部分から構成されています。

詳細は以下の通りです。

アプリケーション・ロジック層はRDMLXコードにより提供され、LANSAエディターで管理されます。アプリケーション・ロジック層によって出力され、プレゼンテーション層へ送られたデータはXMLのフォームになっています。

プレゼンテーション層もLANSAエディターで管理されます。アプリケーション・ロジック層を作成する際は、デフォルトの最も適していると思われるプレゼンテーション・ユーザーインターフェースがXSLに生成されます。RDMLXはコンパイルしてプレゼンテーション・ユーザー・インターフェースを生成する必要がありません。一度XSLが生成されると、LANSAエディターを使って更新することができます。

ユーザーのビジネスデータ(XMLのフォームのもの)は、使い慣れたフィールドや作業リストを使って、これら2つの層の間で交換されます。以下のスケッチはこの構造をまとめたものです。



1.2 WAMアーキテクチャの大きな特徴

アプリケーション・ロジック層とプレゼンテーション層に分離したことで、LANSAのWebソリューションに新しいレベルの柔軟性が与えられました。

ビジネス・ロジックをユーザー・インターフェースから切り離すことで、ある程度WAMアプリケーションの将来が保証されます。つまり、現在のブラウザ・ベースのXHTML(eXtensible HyperText Markup Language：拡張可能ハイパーテキスト・マークアップ言語)アプリケーションを、将来ビジネス・ロジックを変更せずに、最新のユーザー・インターフェース・テクノロジー(XAML、AUIML、その他まだ開発されていないテクノロジーなど)を使って配布することが可能になります。

プレゼンテーション層はテクノロジー・サービス(TSP)を使用して、想定されるプラットフォームごとにユーザー・インターフェースを生成します。現在ウェブの世界で一番一般的なテクノロジー・サービスの1つとして、XHTMLがあります。これはLANSAが生成するTSPの1つで、Internet Explorerやその他のブラウザで見栄えの良いユーザー・インターフェースが提供されます。

LANSAが生成するもう1つのTSPはPocketPC XHTMLであり、これは基本的にデフォルトのXHTMLと同じですが、手の中に入るサイズのデバイスに合うようにデザインされています。

その他のTSPも同様に簡単に導入ができ、適切なTSPを選択することで1つのWAMに複数のユーザー・インターフェースを持たせることができます。つまり、同じアプリケーションを異なるデバイス上で稼働させることができるということです。また、アプリケーションのロジックと見栄えが完全に分離されます。

1.3 その他のWAMの特徴

LANSAのWAMテクノロジーにはこの他にも強力な利点があります。プレゼンテーション層においては以下のような特徴があります。

- 業界標準アーキテクチャ：WAMはXMLやXSLを含む、業界の標準のテクノロジーに基づいています。この為、WAMアプリケーションはオープンで、柔軟性があります。
- エディター：LANSAエディターによって、WAMのユーザー・インターフェースをポイント&クリックで"描画"することができます。更に手を加えたい場合は、ユーザー・インターフェースをソースコードレベルで修正することもできます。つまり、ソースコードを編集し、LANSAエディターに戻り更なる作業を行う、というように、行ったり来たりすることも可能です。
- 出荷時提供の、ユーザー定義が可能な"ウェブレット"：ウェブレットはXSLベースのコンポーネントで、共通フィールド・ビジュアルライゼーションと他のユーザー・インターフェース・エレメントのカプセル化に使用されます。ウェブレットはプレゼンテーション層で再利用するように設計されています。LANSAでは、すぐに使うことができる多くの種類の共通ユーザー・インターフェース・エレメントのウェブレットを提供していますが、独自に構築することも可能です。

アプリケーション・ロジック層では、LANSAのこれまでのアプリケーション開発の長所を多く取り入れて、WAMが構築されました。

- リポジトリ・ベース：LANSAのリポジトリ・ベースのアプリケーション開発により、ビジネス・ルールやドメイン情報が取り込まれ、アプリケーション内で一貫して適用されるようになっていきます。
- コンポーネント・ベース：LANSAの世界では、WAMはコンポーネントであり、他のLANSAコンポーネントを利用することもできます。これにより、将来的にはこのリポジトリ・ベースのアプローチを使って、ビジネス・ルール、アプリケーション・ロジック及びユーザー・インターフェースを分離して展開することが可能になります。つまり、ブラウザ・ベースのアプリケーションと、リッチ・クライアントのアプリケーション、統合プロジェクトの間で共有できる、共通のビジネス・ロジックのコンポーネントを構築することができます。
- 単一のスキル・セット：WAMのアプリケーション・ロジックは、

LANSAのあらゆるところで使用されているのと同じRDMLプログラム言語を使用して構築されています。例えば、5250のグリーン・スクリーンに慣れ親しんでいるLANSA開発者であれば、洗練されたWebブラウザー・ベースのアプリケーションを作成する方法を短時間で簡単に習得できます。

アプリケーション・ロジック層とプレゼンテーション層のキーとなるコンポーネントをもう少し詳しく見ていきましょう。

1.4 アプリケーション・ロジック層

WAMは、その名が示すとおり、ウェブ・アプリケーションのモジュールであり、アプリケーションのロジックを持っています。アプリケーション・ロジック層はRDMLXコードにより提供され、LANSAエディターで管理します。

好みの数(多くても、少なくても)のWAMを使用して、アプリケーションを作成できます。アプリケーション全体を1つのWAMに格納することもできますし、複数のWAMにまたがって配布することもできます。

1.4.1 Webroutine

WAMには1つ以上のWebroutineを組み込みます。このWebroutineにアプリケーション・ロジックが含まれます。これらはサブルーチンと全く同じように考えることができます。実際、Webroutineは、以下のようにWebroutine/Endroutineコマンドの組み合わせを使用して、サブルーチンと同じように定義されます。

```
Webroutine Name(MyWebroutine)
|
| * your RDMLX code here.
|
Endroutine
```

ブラウザからこのWebroutineを実行するには(つまり、このWebroutineを始めるには)、ユーザーは以下のようなURLを入力します。

<http://www.MyWebSite.com/cgi-bin/lansaweb?>

wam=MYWAM&webrtn=MyWebroutine

Endroutineステートメントに達した時、コントロールがプレゼンテーション層に受け渡されることに注意してください。

プレゼンテーション層が表示するのは、Webroutineと関連したWebページです。つまり、各Webroutineはそれぞれユーザーに表示する独自のWebページを持つことができます。この詳細は「[1.5 プレゼンテーション層](#)」を参照してください。

アプリケーション・ロジック層とプレゼンテーション層の間でデータのやり取りを行うことがあるでしょう。これは、「[1.4.2 Webマップ](#)」を使って行います。

1.4.2 Webマップ

WebマップはWebroutineとプレゼンテーション層との間でデータを交換するためのインターフェースです。

プレゼンテーション層に部門の記述を出力するWebマップの指定は、次のようになります。

```
Webroutine Name(MyWebroutine)
  Web_Map For(*output) Fields(#DEPTDESC)

  * your RDMLX code here.

Endroutine
```

Web_MapコマンドのForパラメータを見てください。

パラメータ値は、以下のいずれかになります。

*input - プレゼンテーション層からWebroutineに入ってくるデータを定義します。

*output - Webroutineからプレゼンテーション層に送られるデータを定義します。

*both - プレゼンテーションからWebroutineに入ったり、Webroutineからプレゼンテーション層に送られるデータを定義します。

4つめの値、*noneは「[1.4.3 状態非依存](#)」で説明されています。

部門コードを入力として受け取り、部門の記述を出力するジョブを持つWebroutineがあったとします。そのWebマップの指定は以下のようになります。

```
Webroutine Name(MyWebroutine)
  Web_Map For(*input) Fields(#DEPARTMENT)
  Web_Map For(*output) Fields(#DEPTDESC)

  * some code to fetch the department description from a file here.

Endroutine
```

部門コードは実行される際のWebroutineに対してのであり、部門の記述はWebroutineが終了する際のプレゼンテーション層へのです。これらの設定は非常に厳格です。つまり、DEPARTMENTの値は、この例で指定された通り、Webroutineの入力にしかありません。一旦Webroutineが終了すると、その値はプレゼンテーション層には送り返されません。同じように、Webroutineの実行が開始されると、DEPTDESCには値はありません。なぜなら、これはWebroutineの出力でしかないか

らです。

この点において、Webroutineがある情報を出力できるのと同様に入力として受け取る必要もある場合は、*For*パラメータの値を**both*にするととても便利です。この典型例として、簡単なファイル保守のアプリケーションがあります。ユーザーが部門の情報を編集したい時、Webroutineはそれを表示するために、情報をプレゼンテーション層に**output*する必要があります。ですが、ユーザーによって変更される可能性もあるため、**input*として受け取ることもできなくてはなりません。そのため、Webマップは以下ようになります。

```
Webroutine Name(MyWebroutine)
  Web_Map For(*both) Fields(#DEPARTMENT #DEPTDESC)
  * some code to fetch the department description from a file here.
Endroutine
```

フィールドと同様に、作業リストもデータのやり取りに使用することができます。Webマップ上に作業リスト名を以下のように指定するだけです。

Web_Mapコマンドでは、識別子またはフィールド名のいずれかを使用できます。この識別子を使ってXSLとXMLが生成されます。多くの場合、WAMエディターでは名前が表示されますが、裏ではこの識別子が使われています。

```
Def_List Name(#WLDEPTS) Fields(#DEPARTMENT #DEPTDESC) Type(*Working)

Webroutine Name(MyWebroutine)
  Web_Map For(*both) Fields(#WLDEPTS)
  * some code to fetch the department description from a file here.
Endroutine
```

Webマップのフィールドの属性

Web_Mapコマンドの*For*パラメータが、どのフィールドや作業リストがプレゼンテーション層とWebroutineとの間で交換されるかについて影響を与えることに注意してください。全ての関連するWeb_Mapコマンドは、Webroutineのためのパラメータ・リストを定義しているようなものだと考えると分かりやすいかもしれません。

ただし、Web_Mapの*For*パラメータは、フィールドがどのように表示されるかには影響を与えません。これは、プレゼンテーション層で行われます。

ある値をfor (**both*)に設定して、ユーザーにはWebページ上で値を変更したり、見たりできないようにしたい時があると思います。このような場合、Webマップのフィールドに以下の属性を指定します。

**output*- データはWebページに表示されますが、ユーザーは変更することはできません。

**hidden* - データはロジックとプレゼンテーション層の間でやり取りされますが、ユーザーには表示されません。

**private* - データはロジックとプレゼンテーション層の間でやり取りされますが、XSLにマージされません。(これは、XSL変換プロセスの間にフィールドの値やリストの中身が参照されますが、最終的なページに生成される必要がない場合の特別な使用方法です。)

注：**input* も属性として有効で、デフォルト値になっています。

ただし、フィールドをどのように表わすかはプレゼンテーション層の責任範囲であることを忘れないでください。Webマップに設定された、これを制御するフィールド属性は、ウェブデザインがWebroutineの為に生成された時のみ有効になります。その後、LANSAエディターを使用して、フィールドがどう表されるかを変更することができます。つまり、Web マップに指定されたフィールド属性は意思表示としてのみ扱われるべきということを意味しています。なぜならば、フィールドがどのように表されるかは結局プレゼンテーション層がコントロールするからです。

次は、DEPARTMENTがやりとりされて表示されますが、ユーザーに値を変更してほしくない場合の例です。

```
Webroutine Name(MyWebroutine)
  Web_Map For(*both) Fields((#DEPARTMENT *output) #DEPTDESC)
    * some code to fetch the department description from a file here.
Endroutine
```

この例で注意する点は以下の通りです。

- **output*フィールド属性を指定する際の唯一の影響は、Webroutineのウェブデザインを生成する時にフィールド#DEPARTMENTが出力専用になることです。このフィールドはLANSAエディターを使って変更される可能性があるため、Webページ上で出力専用で残るということではありません。
- フィールド#DEPARTMENTがWebページで出力専用の場合、このWebページは次のWebroutineにフィールド値をPOSTできません。例えば、Webroutineがそのデータを同じWebroutineにPOSTする場合、フィールド#DEPARTMENTの値が出力専用であるためPOSTすることができず、うまく作動しません。しかし、このWebroutineがフィールド#DEPARTMENTの値をPOSTする別のWebroutineによって呼び出された場合は、うまく作動します。

このように、フィールド属性として**output*を指定するという事は、Web_Mapコマンドにfor(**output*)を使用するのとまったく違った意味と効果があります。

コンパイル時にWebroutineがプレゼンテーション層のXSLを生成

Webマップを持つWebroutineがコンパイルされると、プレゼンテーション層のために適切なXSLが生成されます。**input*や**both*のWebマップの場合、LANSAは通常そのフィールドや作業リストを表示するWebroutineのWebページを生成します。

何がプレゼンテーション層に生成されるかについては、「[1.5.2 プレゼンテーション層にデフォルトで生成されるもの](#)」で説明しています。

グローバルWebマップ

Webroutineの範囲外にWebマップ指定した場合、これはグローバルWebマップになります。これはWAMの中のすべてのWebroutineがそのWebマップを採用するということです。

グローバルWebマップの使用方法は様々ですが、使用例はこのガイドの例えば「[1.4.3 状態非依存](#)」などにあります。

これまで、次の例のようにWebroutine中に指定されたWebマップについて説明しました。

```
Begin_Com Role(*EXTENDS #PRIM_WAM)
  Def_List Name(#WLDEPTS) Fields(#DEPARTMENT #DEPTDESC) Type(*Working)
  Webroutine Name(MyWebroutine)
    Web_Map For(*both) Fields(#SECTION #SECDESC)
    Web_Map For(*both) Fields(#WLDEPTS)
    * your RDMLX code here.
  Endroutine
  Webroutine Name(MyOtherWebroutine)
    Web_Map For(*both) Fields(#WLDEPTS)
    * some more RDMLX code here.
  Endroutine
End_Com
```

以下のグローバルWebマップの例では、WLDEPTS作業リストのWebマップの位置に注意してください。コンパイル時、*MyWebroutine*と*MyOtherWebroutine*の両方がこのWebマップを採用します。また、*MyWebroutine*は独自のローカルのWebマップも持っています。

```
Begin_Com Role(*EXTENDS #PRIM_WAM)

  Def_List Name(#WLDEPTS) Fields(#DEPARTMENT #DEPTDESC) Type(*Working)

  Web_Map For(*both) Fields(#WLDEPTS)

  Webroutine Name(MyWebroutine)
    Web_Map For(*both) Fields(#SECTION #SECDESC)

    * your RDMLX code here.
  Endroutine

  Webroutine Name(MyOtherWebroutine)

    * some more RDMLX code here.
  Endroutine

End_Com
```

プログラムによりWebroutineの流れをコントロール

後ほど説明するプレゼンテーション層から実行されるWebroutineと同様、CallやTransferのRDMLコマンドを使用してプログラムのコントロール下でWebroutineを実行することも可能です。次のコードを見てみましょう。

```
Begin_Com Role(*EXTENDS #PRIM_WAM)
  Webroutine Name(Initialize)
    Web_Map For(*output) Fields(#DEPARTMENT)

    * some code to derive a value for DEPARTMENT here.

    Transfer Toroutine(ShowPage)
  Endroutine
  Webroutine Name(ShowPage)
    Web_Map For(*both) Fields(#DEPARTMENT)
  Endroutine
End_Com
```

ユーザーがブラウザでURLを入力してInitializeWebroutineを実行するとします。ここでDEPARTMENTに値が入られます。

TransferコマンドはコントロールをShowPageWebroutineに移し、このWebroutineがDEPARTMENTの値を受け取ります。現在のWebroutineのWebマップ定義に従って、適切なマッピングが移行（trasfer）の一部として行われます。このShowPageWebroutineは実際のところ何も行いませんが、DEPARTMENTの値をそのまま出力して終了します。コントロールはその後プレゼンテーション層に送られ、ShowPageのWebページを表示します。

InitializeWebroutineのEndroutineコマンドは、コントロールがShowPageに移るので、絶対に実行されないことに注意してください。ですから、InitializeWebroutineではページは表示されません。

このTransferの使用方法はやや上級の技術ですが、WAM内の1つのWebroutineがどのようにしてプレゼンテーション層の表示を担当するかを示したものです。この技術を使用し、他のWebroutineも単体でアプリケーション・ロジックを実行するためだけに使用することができます。これらすべてで最後に行われることは、ShowPageWebroutineにコントロールを移すことです。

この技術によりアプリケーションが簡素化されることが分かります。

す。これは、アプリケーションが複数のWAMで構成されており、各WAMはWebページを表示するためにたった1つのWebroutineを使用しているということです。このサポートをより強化するため、TransferコマンドはWAM間の移行もサポートしています。

[Transfer Toroutine\(#MYWAM99.ShowPage\)](#)

CallコマンドはWebroutineを実行するという意味で、Transferコマンドと似ています。違いは、コントロールが移らないということです。WebroutineはEndroutineコマンドまですべて実行され、その後コントロールが実行したWebroutineに戻ってきます。呼び出されたWebroutineのWebページは表示されないことに注意してください。

1.4.3 状態非依存

WAMのキー・ポイントの一つは、WAMが状態非依存、つまり状態を保持しないということです。実際、すべてのインターネット・ベースのアプリケーションは状態を保持しません。これはつまり、WAMがプレゼンテーション層から実行された時、これが稼動し(ジョブがサーバー上で開始され)、いくつかの出力(ウェブ・ページ)を作成し、終了する(サーバー上のジョブが終了し、コントロールがブラウザに送り返される)ということです。

このジョブの開始と終了は、どう見ても1つの"トランザクション"です。ユーザーのウェブ"セッション"(つまり、複数のトランザクションにわたるもの)の間維持される必要があるデータは全て、どこかで保持しなくてはなりません。この良い例が、ユーザーがウェブサイトにログオンした時です。ユーザーがログオンしたという事実は、複数のトランザクションにわたって保持される必要があります。なぜなら、アプリケーションには、ユーザーがログオンしている場合のみ実行することができる機能もあるからです。

Windowsもしくはグリーン・スクリーンのアプリケーションでは、これは全て暗黙の事項です。ユーザーのジョブはアクティブな状態でメモリー上に残され、ユーザーの次の動きを待ちます。しかしウェブでは、ユーザーが次に何をするかは誰も分かりません。新しいURLを入力するかもしれませんし、戻るボタンを押したり、ブラウザが閉じられるかもしれません。

Webページに表示されるデータはWebマップを使用してやりとりされます。非表示のデータでさえやりとりされます。ただし、"セッション"データについてはこれが起きてはいけません。これは、アプリケーション・ロジック層にのみ重要なデータで、プレゼンテーション層で起こることに影響を与えないデータだからです。

WAMは、WebマップのForパラメータに*noneという特別な値を、及びWebマップのoptionsパラメータに*persistという値を使用することで、セッションデータに対するこの考え方をサポートしています。ユーザーのログオン状態を保持するためのWebマップは次のようになります。

```
Begin_Com Role(*EXTENDS #PRIM_WAM)
  Web_Map For(*none) Fields(#LOGGEDON) Options(*PERSIST)
  Webroutine Name(ShowPage)
    Web_Map For(*both) Fields(#DEPARTMENT)
  Endroutine
End_Com
```

Webマップの位置に注意してください。グローバルなので、WAM内の全てのWebroutineがこれを採用します。セッションデータのWebマップをこのようにコーディングするのは、とても理にかなっていません。そしてこのデータはいつでもマップに出し入れされます。

他にもセッションデータのコーディング時に考慮しなくてはならないパラメータや設定はありますが、ここでは基本的な考え方を理解しておいてください。詳細については、「[WAM セッション管理](#)」を参照してください。

1.4.4 再利用可能パーツの役割

インターネット・ベースであるにかかわらず、すべてのアプリケーションにおいて1箇所以上で使用する必要のある機能がある場合があります。この良い例としては、ログオン機能(ログオンのユーザーID、日付、時刻をデータベースに記録したりする)があり、これは、異なるWAMアプリケーションにおいて同じ機能が存在する場合があります。

LANSAsの再利用可能パーツは必要なレベルでのモジュール性を提供します。単にDefine_comコマンドを使用して適切なパーツを宣言し、アプリケーション・ロジックにそれらを使用します。次はこの例です。

```
Begin_Com Role(*EXTENDS #PRIM_WAM)

  Web_Map For(*none) Fields(#LOGGEDON) Options(*PERSIST)

  Webroutine Name(Logon)
    Web_Map For(*input) Fields(#USER #PASSWORD)

    Define_Com Class(#ADHLOGON) Name(#UserServices)

    If (#UserServices.ValidateLogOnDetails( #USER, #PASSWORD ))
      #LOGGEDON := True

      Transfer Toroutine(LogOnSuccessful)
    Else
      Transfer Toroutine(LogOnFailed)
    Endif
  Endroutine

  Webroutine Name(LogOnSuccessful)
  Endroutine

  Webroutine Name(LogOnFailed)
  Endroutine

End_Com
```

LogonWebroutineは、プレゼンテーション層で表示される他のWebroutineから、ユーザーIDやパスワードを受け取ります。再利用可能パーツのインスタンスであるADHLOGONが活性化され、そのValidateLogOnDetailsメソッドを使ってユーザーIDとパスワードの妥当性検査が行われます。ログオンが成功した場合は、#LOGGEDONフィールド(永続的なセッ

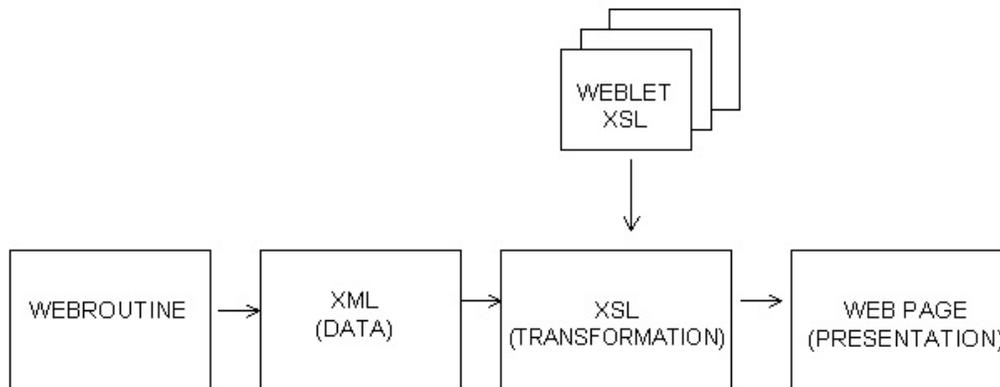
ションデータとして宣言されたもの)にTrueがセットされ、コントロールがLogOnSuccessfulWebroutineに送られます。ログオンが失敗した場合は、コントロールはLogOnFailedWebroutineに送られます。

ここで分かる通り、これはアプリケーション・ロジックが一箇所にカプセル化され、それを他のWAMでも繰り返し使用することができるようにするというすばらしい技術です。

1.5 プレゼンテーション層

プレゼンテーション層とは？

以下のスケッチは、Webroutineからユーザーが見ているデバイスの"画面"までの流れを示しています。プレゼンテーション層はWebroutineの右側にある全てのボックスから構成されています。



Webroutineが終了すると、フィールドと作業リストのデータ(*outputおよび*bothのWebマップに定義された通り)がXMLドキュメントに書き出されます。これは、ユーザー・インターフェースのために生成されたXSL(使用されている全てのウェブレットのXSLを含む)にマージされ、ユーザーに表示されるドキュメント(この例ではXHTML形式のWebページ)に変換されます。

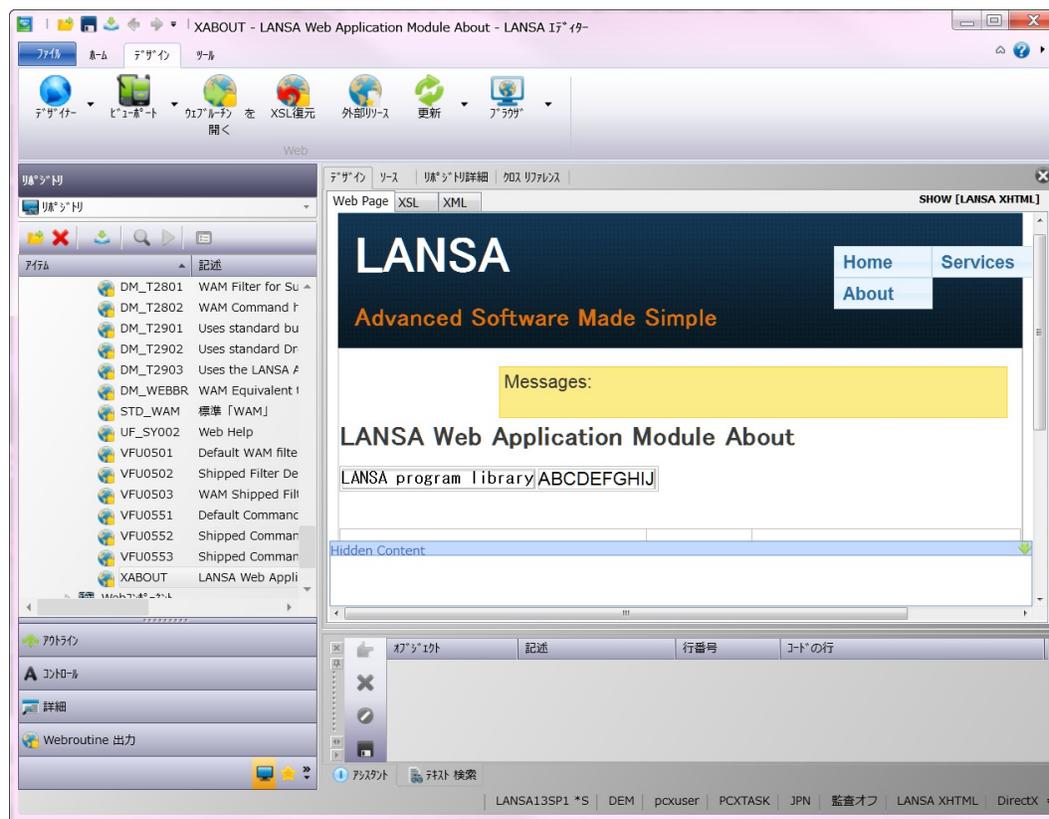
1.5.1 エディター

LANSAエディターの中でWebデザインが管理される部分はビジュアル・エディターです。これを使ってWebデザインを'描画'していきます。

LANSAエディターのソースタブで開いたWAMは、そのWebroutineの[デザイン]タブでデザインしたWebページを持っています。

WAMのWebroutineのウェブ・デザインを開くには、Webroutineコマンドのすぐ右側にある、Webroutineデザイン・グリフと呼ばれる緑の矢印を選択します。Webroutineにアクティブなテクノロジー・サービス・プロバイダ用のウェブデザインがない場合、自動的に生成されます。それ以外は、デザインタブが現在のテクノロジー・サービス・プロバイダ用に選択されたWebroutineのウェブデザインをロードします。

以下の合成図はこの様子を示しています。



[デザイン]タブの[Web Page]タブは、ユーザーに表示されるWebページを生成するのに使用されるXSLソースを視覚的に表示したものです。実際の元となるXSLは、[XSL]タブを選択することで見ることができます。同様に、[XML]タブをクリックすることでXMLを表示することができます。

ほかにも、ページ描画のプロセスで役立つ機能や、詳細入力役割を果

たすタブが使用できます。これらのタブはエディター・ウィンドウの左右、もしくは下に表示することもできますし、自由に移動できるフローティングに設定することも可能です。LANSAエディターの主な機能の詳細については、『*Visual LANSA ユーザーガイド*』の「[ワークスペースの設定](#)」を参照してください。

上図で示されているエディター・ウィンドウの主な機能は次の通りです。

- [アウトライン]タブには、現在編集集中のWebページに含まれているフィールドやリスト、その他のエレメントが表示されます。このリストのエントリーをクリックすると、[Web Page]タブの対応するコントロールが選択されます。[デザイン]のHTMLまたはXSLエレメント上でドラッグ・アンド・ドロップ操作が行われた場合、このエレメントは[アウトライン]タブ内で強調表示され、視覚的に確認しながらドラッグ・アンド・ドロップ操作ができるようになっています。
- [Web デザイン] タブは現在のWebroutineの全言語そして、全テクノロジー・サービス・プロバイダ用のウェブデザインを全て表示します。このタブはウェブデザインを削除したり、変更を元に戻したり、言語のコピーを作成したりするのに便利です。

図の左側にあるウィンドウは以下のようになっています。

- [リポジトリ]タブはリポジトリに保存された利用可能な全てのオブジェクトのリストが表示されます。LANSAフィールドはリポジトリからWebページへ(フィールドもしくはリストのいずれかとして)ドラッグ・アンド・ドロップすることができます。また、[リポジトリ]から[Webroutine出力]タブへ、もしくは[Webroutine出力]タブからWebページへドラッグ・アンド・ドロップすることもできます。
- [詳細]タブは[Webページ]タブで現在選択されているコントロールのプロパティを表示します。
- [Webroutine出力]タブは現在のWebroutineで使用可能なWebマップにあるフィールドとリストを表示します。必要に応じてこのリストからWebページへ、もしくは[リポジトリ]からこのタブへ、直接ドラッグ・アンド・ドロップすることができます。

[WebRoutine出力]タブは次のような時に利用します。

- デザインにリストをドロップする前にリスト内のフィールドの順序を変更する場合。この作業は別の場所では行えません。
- Webマップ間でのフィールドやリストの移動・ローカルWebマップからグローバルWebマップまたはその逆方向への移動の場合。

リポジトリと[WebRoutine出力]タブ間のオブジェクト移動の場合は、両方のタブが同時に表示されるように画面レイアウトを調整します。

1.5.2 プレゼンテーション層にデフォルトで生成されるもの

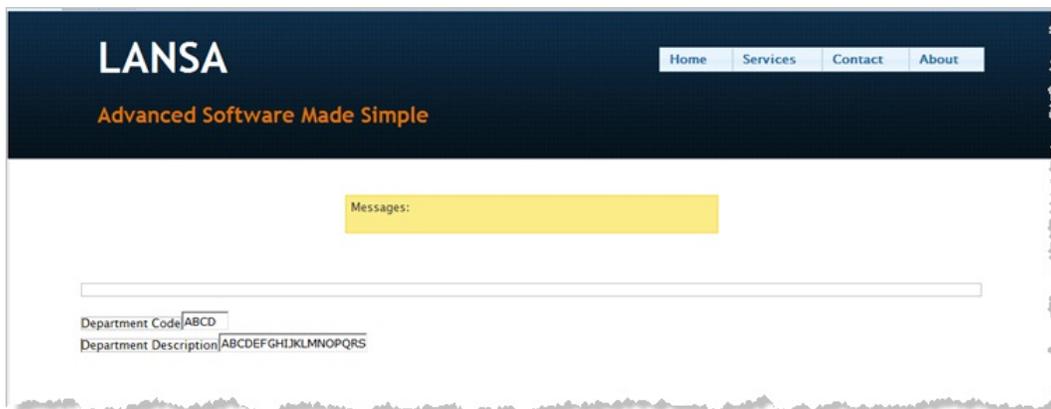
WAMがコンパイルされる時、指定されたWebマップに基づいていくつかのデフォルトのユーザー・インターフェースの部分が作成されます。これらはフィールドと作業リストのどちらがマップされているかによって異なります。例えば、このWebroutineは2つのフィールドをマップします。

```
Webroutine Name(MyWebroutine)
  Web_Map For(*both) Fields((#DEPARTMENT *output) #DEPTDESC)

  * some code to fetch the department description from a file here.

Endroutine
```

そして、次のようなWebページ形式でユーザー・インターフェースを生成します。



ご覧の通り、各フィールドに対し2つの列と1つの行を持ったテーブルが生成されています。DEPARTMENTの*output属性は、ユーザーによって変更できないという意味であり、対してDEPTDESCの*inputという暗黙的な属性は、入力可能であることを示しています。

このWebroutineに対しては、作業リストが指定されています。

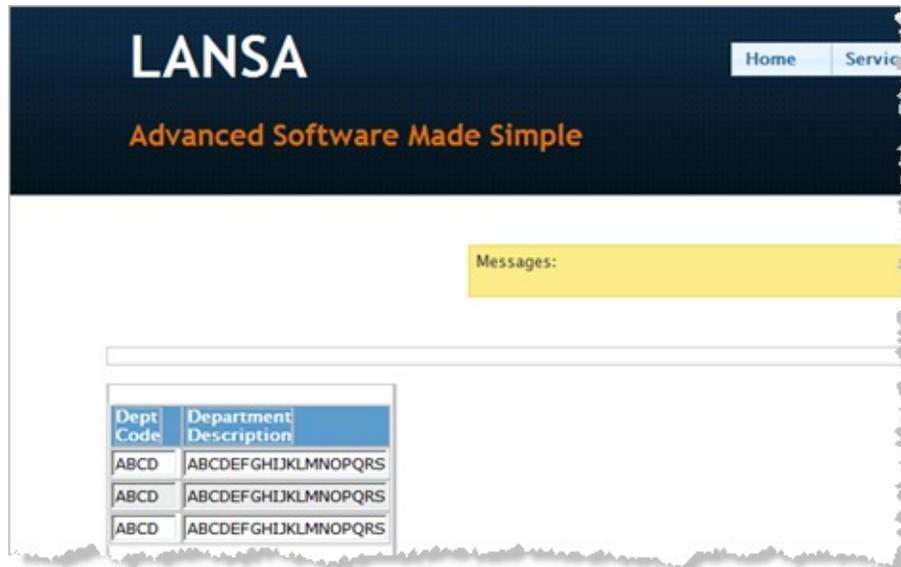
```
Def_List Name(#WLDEPTS) Fields(#DEPARTMENT #DEPTDESC) Type(*Working)

Webroutine Name(MyWebroutine)
  Web_Map For(*both) Fields(#WLDEPTS)

  * some code to fetch the department description from a file here.

Endroutine
```

そして次のようなWebページが生成されます。



2つの列を持ったテーブルが生成され、それぞれの列がリストのフィールドを表し、テーブルの各行はリストのエントリーを表しています。デフォルトとして、テーブルの3つの行が、設計の為に表示されます。これらデフォルトのユーザー・インターフェースの部分はすべて良いものの、ビジュアルイズ・フィールドやクリック可能なイメージ、ハイパーリンクなど、もう少し"ウェブっぽい"ものが欲しい場合はどうすればよいのでしょうか？ここで、[1.5.3 ウェブレット](#)の出番となります。

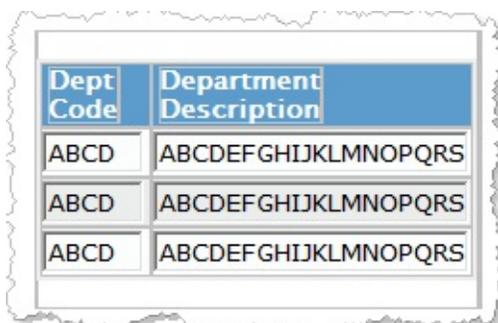
1.5.3 ウェブレット

ウェブレットはフィールドや作業リストについての情報を取得し、様々な方法でそれを表示できるXSLの一部です。

この概要の段階では、ウェブレットはこれを使ってユーザーがWebページ上でやりとりをするビジュアルな構成要素だと考えてください。ウェブレットはハイパーリンクや、プッシュボタン、クリック可能なイメージなどを提供します。多くの場合、フィールドや作業リストのビジュアルライズにも使用できます。

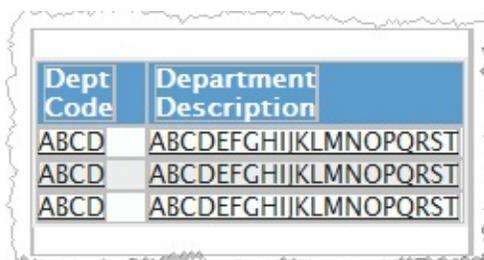
例えば、何らかのアクションをするために、選択した部門コードや記述をユーザーがクリックできるようにしたいとします。AnchorウェブレットをLANSAエディターの列にドラッグ・アンド・ドロップすると、部門コードと部門の記述がハイパーリンクとして表示されます。

これが、



Dept Code	Department Description
ABCD	ABCDEFGHIJKLMNOPS
ABCD	ABCDEFGHIJKLMNOPS
ABCD	ABCDEFGHIJKLMNOPS

このようになります。



Dept Code	Department Description
ABCD	ABCDEFGHIJKLMNOPS
ABCD	ABCDEFGHIJKLMNOPS
ABCD	ABCDEFGHIJKLMNOPS

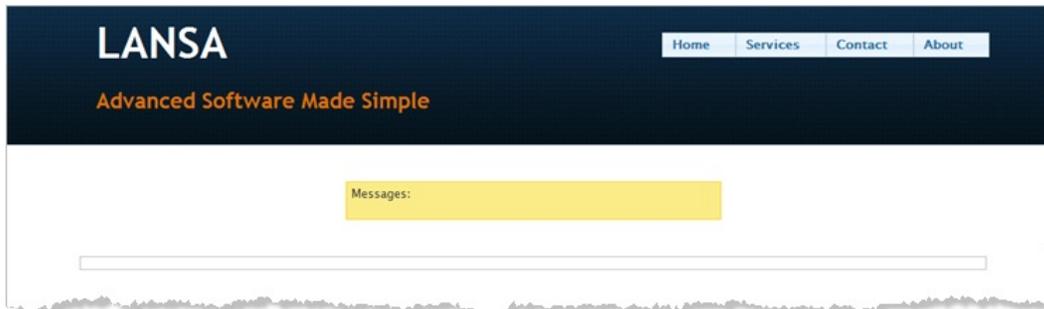
以下はVisual LANSAと共に標準で出荷される、ウェブレット一覧の一部です。

Item	Description
Anchor	Standard Hyperlink
Attachment Panel	Panel with attachment layout manager
Banner	Standard Banner
Dynamic HTML menu bar	Standard DHTML Menu
Grid	Standard Grid
Horizontal splitter	Standard horizontal splitter
Large List	Standard Large List
List paging buttons	Standard List Paging Buttons
List paging images	Standard List Paging Images
Listbox	Standard listbox
Memo using a field	Standard Textarea
Memo using a list	Textarea from a list
Menu item	Standard Menu Item
Messages	Standard Messages
Navigation panel	Navigable panel
Panel	Standard Panel
Prompter	Standard Prompter
Push button	Standard Button
Push button with images	Standard Image Button
Radio button	Standard Radio Button
Radio group	Static radio buttons
Standard basic layout	Standard Layout
Standard hidden fields	Standard Hidden Fields
Standard layout schema #1	Standard Layout Schema #1
Standard layout schema #2	Standard Layout Schema #2
Standard layout schema #3	Standard Layout Schema #3
Standard layout schema #4	Standard Layout Schema #4
Standard layout schema #5	Standard Layout Schema #5
Tree view	Standard treeview control IE only
Tree view target	Standard treeview control IE only
Vertical splitter	Standard vertical splitter

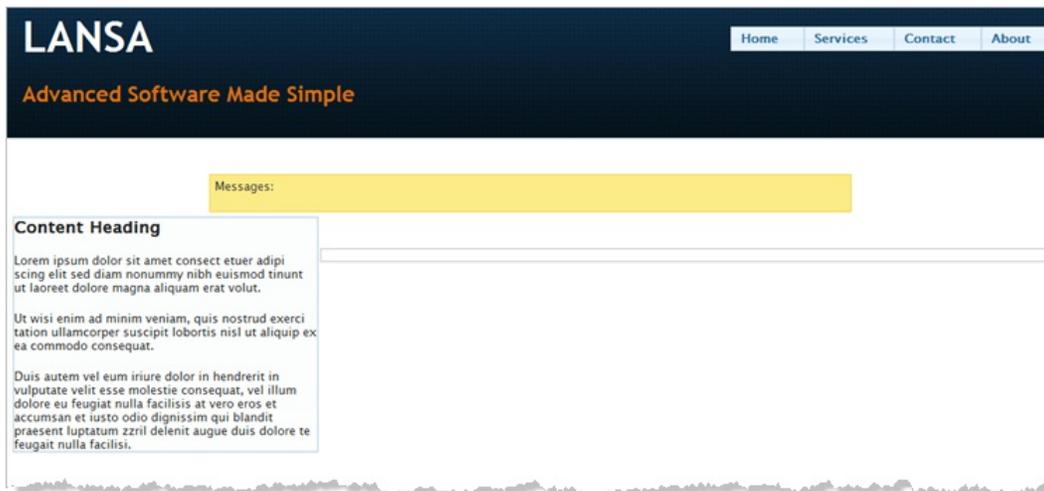
リストボックスやプッシュボタンなど、分かりやすいタイプのユーザー・インターフェース・ウィジェットの他、'標準レイアウト・スキーマ'も含まれています。これらのウェブレットはWebページに基本的な外観を提供し、次のようなものも準備されています。

- 会社のロゴを表示する場所
- 様々なメニューのフレームワーク
- アプリケーションのメッセージが送られる場所

例えば標準レイアウト・スキーマ 1は、これまでもこのドキュメントのあらゆるところで使われていますが、次のようなものです。



標準レイアウト・スキーマ 2はすこし外観が違います。例えば以下のように左揃えされた列があります。



ウェブレットは、標準のVisual LANSAのユーザー・インターフェース・コントロールと同様、その外観や動作をコントロールするプロパティがあります。これらは[プロパティ]タブに表示されます。[プロパティ]タブは、ウェブレット選択時に[詳細]タブをクリックするとアクセスできます。以下はAnchorウェブレットのプロパティ・シートです。

Properties	
<xsl:call-template>	
Name	Value
name	std_anchor
With Parameters	
name	\$DEPARTMENT/@id
value	\$DEPARTMENT
currentrowhfield	STDROWNUM
currentrownumval	position()
reentryfield	STDREENTRY
reentryvalue	D
hide_if	False
formname	LANSA
url	javascript:void();
on_click_wamname	\$!web_WAMName
on_click_wrname	
protocol	
show_in_new_window	False
target_window_name	<xsl:if xmlns:xsl="http://www.w3.org/199
pos_absolute_design	
width_design	
relative-image-path	"
absolute-image-path	<xsl:if xmlns:xsl="http://www.w3.org/199
class	std_anchor
mouseover_class	
text_class	std_anchor_text
presubmit_js	
tab_index	

シート上部にある、nameとvalueのパラメータ/プロパティを見て下さい。ウェブレット・インスタンス名は、ビジュアル化するLANSAフィールド名と同じで(DEPARTMENT)、その値はフィールド値に設定されます。

シートの下に続くその他のプロパティを見て下さい。例えば、on_click_wamnameプロパティやon_click_wrnameプロパティは、ユーザーがアンカーをクリックしたときに呼び出されるWAMとWebroutineの名前を指定します。MYWAM01に戻って、DepartmentSelected Webroutineは以下ようになります。

```

Webroutine Name(DepartmentSelected)
  Web_Map For(*input) Fields(#DEPARTMENT)
  * do something with the selected department.
  Transfer Toroutine(ShowPage)
Endroutine

```

ウェブレットは基礎となるJavascriptコードや、その他のWebページの表示できないコンポーネントを供給するなど、ビジュアルでない役割も果たすことができます。

1.6 WAMサンプル - 始めから終わりまで

WAMの実行に際し、キー・ポイントで何が起こるかを反復するために、2ページのウェブ例を一緒に見て行きましょう。

最初のページは、ブラウザで表示されると、次のようになります。



ご覧のように、これは送り状の状況照会です。送り状番号を入力するようになっています。次に送り状の確認ボタンをクリックして、データベースから情報を取得して、送り状の状況が表示されます。そこで、3つのステップが発生します。これを簡単にする為に、ステップごとにWebroutineを持つことにします。

- 1つは、上記のような初期画面を表示するもの。(ConsignmentEnquiry)
- 1つは送り状番号を取得し、妥当性をチェックするもの。(CheckConsignment)
- そしてもう1つは、データベースを検索しデータを表示するもの。(ShowConsignment)

まず初めに、初期画面の表示方法です。次のようにURLを入力することもできます。

```
http://localhost/CGI-BIN/lansaweb?webapp=WDWAM01+webtrn=ConsignmentEnquiry+ml=LANSA:XHTML+part=WEX+lang=ENG
```

パラメータのwebapp=WDWAM01及びwebtrn=ConsignmentEnquiryを見てください。これらは実行されるWAM名(WDWAM01)とWebroutine名(ConsignmentEnquiry)を示しています。

またはより大きなWAMアプリケーションの別の部分から、アンカー(もしくはハイパーリンク)をクリック、ボタンを押す、メニュー・アイテムを選択するなどして実行することも可能です。いずれの方法にせよ、そのメカニズムは同じです。つまり、WAM内にあるWebroutineが実行されるのです。

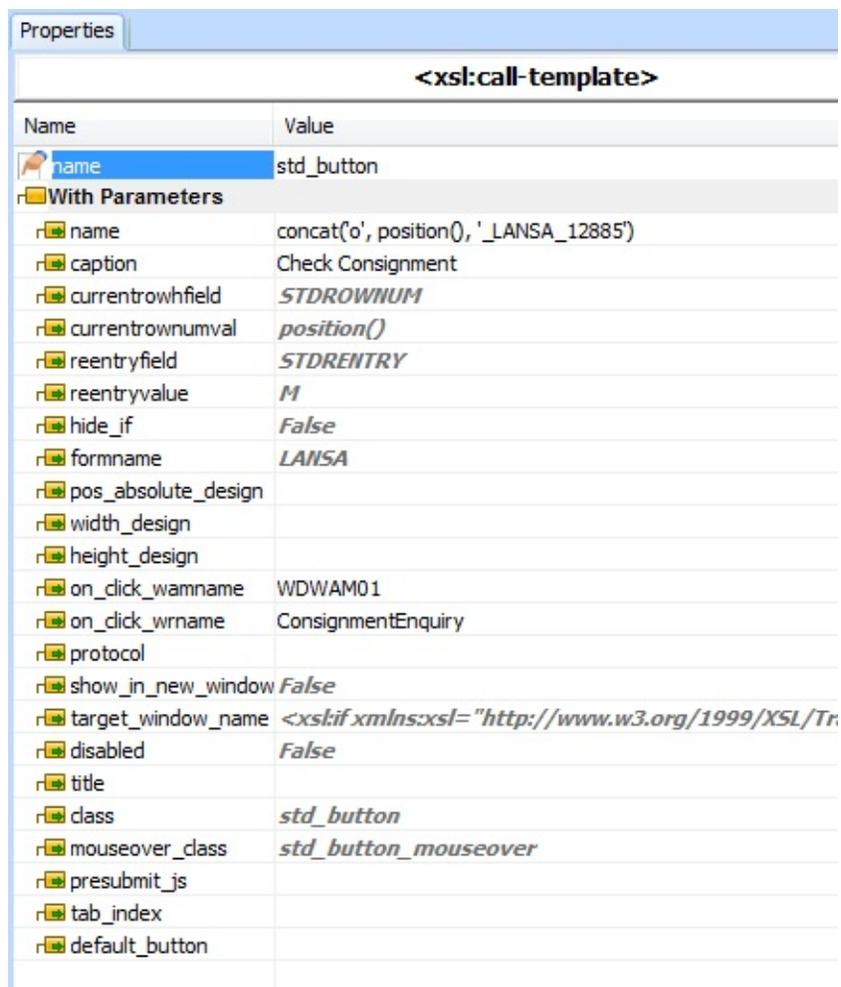
最初のConsignmentEnquiryWebroutineの基礎となるコードは次のようにな

ります。

```
Webroutine Name(ConsignmentEnquiry) Desc('ACME Couriers Consignment Status Enquiry')
  Web_Map For(*both) Fields(#WDCONSIGN)
Endroutine
```

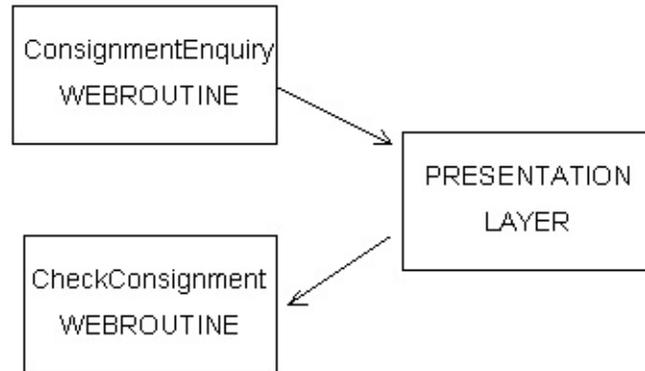
ここでキーになるのは、WDCONSIGNフィールドのWebマップです。これは*bothとして定義されており、この値がWebroutineの最初で読み込まれ、Webroutineが終わる時に書き出されるということを意味しています。思い出してください、WAMをコンパイルするときに、LANSAはWebページにWDCONSIGNを表示するために、XSLをいくつか作成します。

ただし、プッシュ・ボタンは作れません。ですので、LANSAエディターでWebroutineを右クリックして、ページを開きます。プッシュ・ボタンのウェブレットをデザイン上にドラッグ・アンド・ドロップし、そのプロパティを設定します。



<xsl:call-template>	
Name	Value
name	std_button
With Parameters	
name	concat('o', position(), '_LANSA_12885')
caption	Check Consignment
currentrowhfield	STDROWNUM
currentrownumval	position()
reentryfield	STDREENTRY
reentryvalue	M
hide_if	False
formname	LANSA
pos_absolute_design	
width_design	
height_design	
on_click_wamname	WDWAM01
on_click_wrname	ConsignmentEnquiry
protocol	
show_in_new_window	False
target_window_name	<xsl:if xmlns:xsl="http://www.w3.org/1999/XSL/Tr
disabled	False
title	
class	std_button
mouseover_class	std_button_mouseover
presubmit_js	
tab_index	
default_button	

caption、**on_click_wamname**及び**on_click_wrname**のプロパティ設定に注意してください。on_click プロパティは、ボタンがクリックされたら、WDWAM01のWAMの中で*CheckConsignmentWebroutine*を実行します。ここまでで、この小さなアプリケーションは以下のような感じになります。



ここでLANSAエディターに戻り、*CheckConsignmentWebroutine*に以下のRDMLXコードを入力して下さい。

```
Webroutine Name(CheckConsignment)
  Web_Map For(*input) Fields(#WDCONSIGN)

  * see if this consignment exists in the consignment status file.
  Check_For In_File(WDCONST) With_Key(#WDCONSIGN)

  * it exists ...
  If_Status Is(*EQUALKEY)
    * show the consignment status.
    Transfer Toroutine(ShowConsignment)
  Else
    * it doesn't exist: show an error message ...
    Message Msgbxt('Invalid Consignment Note number. Please try again.')

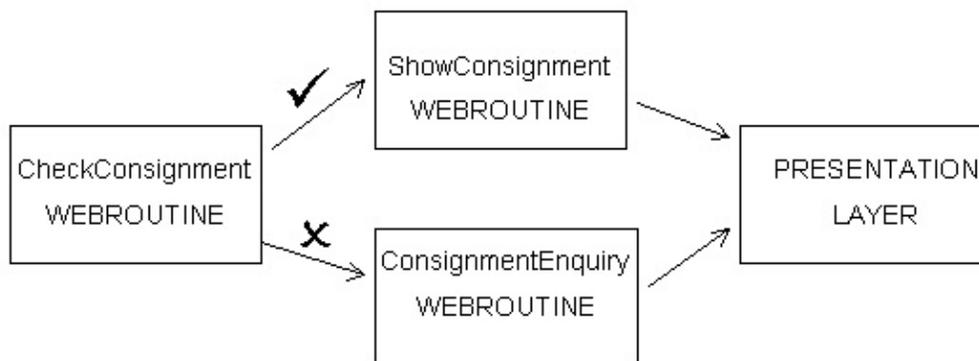
    * and re-display the main page.
    Transfer Toroutine(ConsignmentEnquiry)
  Endif
Endroutine
```

再度Webマップを見てください。このFor(*input)設定は、呼び出された時に(この例では、ユーザーがプッシュ・ボタンをクリックしたときに)WDCONSIGNがWebroutineにマップされることを意味しています。送り状の状況ファイルは入力された送り状番号が存在するかどうかを確

認る為にチェックされます。存在する場合は、コントロールはShowConsignmentWebroutineに送られます。存在しない場合は、適切なエラーメッセージが発行され、コントロールが元のConsignmentEnquiryWebroutineに移ります。

CheckConsignment Webroutineは永久に最後に達しないので、ウェブ・ページを表示することはないということに注意してください。常に、ShowConsignmentまたはConsignmentEnquiryWebroutineのどちらかにコントロールを移します。

WAMのこの部分の流れは、次のようになります。



次に、ShowConsignmentWebroutineのコードを見てみましょう。

```
Webroutine Name>ShowConsignment) Desc('ACME Couriers Consignment Status Enquiry')
Web_Map For(*both) Fields((#WDCONSIGN *output))
Web_Map For(*output) Fields((#WDCONSTS *output))

* fetch the consignment status from the file.
Fetch Fields(#WDCONSTS) From_File(WDCONST) With_Key(#WDCONSIGN)

Endroutine
```

再度、Webマップを見てください。送り状番号(WDCONSIGN)は*bothWebマップを使用して入ってきます。

状態フィールドのWDCONSTSは、送り状番号をキーとしてデータベースから読み込まれます。

Webroutineが終了します。送り状番号と状態のフィールドはWebroutineの外にマップされます。

コントロールはプレゼンテーション層に送られ、送り状番号と状況が表示されます。フィールドの*output属性にも注意してください。これは出力専用として表示されるということを意味しています。

有効な送り状番号が入力されたと仮定して、最初からWAMを実行して

みましょう。

```
http://localhost/CGI-BIN/lansaweb?webapp=WDWAM01+webtrn=ConsignmentEnquiry+ml=LANSA:XHTML+part=WEX+lang=ENG
```

```
Webroutine Name(ConsignmentEnquiry) Desc('ACME Consignment Status Enquiry')
Web_Map For(*BOTH) Fields(#WDCONSIGN)
Endroutine
```

LANSA
Advanced Software Made Simple

Home Services Contact About

ACME Consignment Status Enquiry

Consignment Note Number A1234567890

```
Webroutine Name(CheckConsignment)
Web_Map For(*input) Fields(#WDCONSIGN)
* validation code here
* show the consignment status
Transfer Toroutine(ShowConsignment)
Endroutine
```

```
Webroutine Name(ShowConsignment) Desc('ACME Consignment Status Enquiry')
Web_Map For(*BOTH) Fields((#WDCONSIGN *output))
Web_Map For(*OUTPUT) Fields((#WDCONSts *output))
* fetch the consignment status from the file
Endroutine
```

LANSA
Advanced Software Made Simple

Home Services Contact About

ACME Consignment Status Enquiry

Consignment Note Number A1234567890
Consignment Status Shipped on 12/24/2010

```
Webroutine Name(ConsignmentEnquiry) Desc('ACME Consignment Status Enquiry')
Web_Map For(*BOTH) Fields(#WDCONSIGN)
Endroutine
```

[Check Another]のプッシュ・ボタンに注目してください。これにより、ComponentEnquiryWebroutineを再実行して、処理を最初のところにもどします。

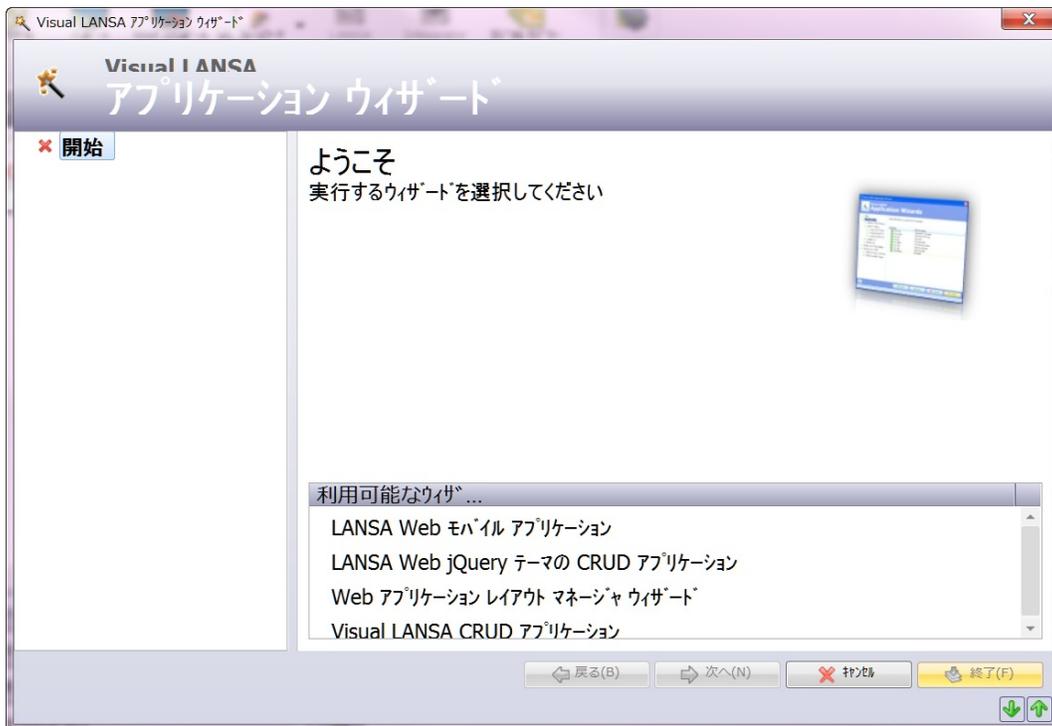
1.7 WAM ウィザード

Visual LANSA のアプリケーション・ウィザードを使用して、事前に準備された手順を追うことで、複雑な Visual LANSA アプリケーションを作成することができます。

Visual LANSA ウィザードを実行する前に、以下を確認してください。

- Visual LANSA がインストールされ、正しく構成されている。
- 区画が Web 対応になっている。
- システムに Web サーバーがインストールされており、実行オプション選択時に WAM が実行されるよう構成されている。

LANSA エディターでウィザードにアクセスするには、[ツール] タブを選択し、[ユーティリティ] グループ内の [ウィザード] をクリックします。



開始するには、[利用可能なウィザード] のリストからウィザードをクリックします。

各ウィザードには、**×** マークの付いた質問リストがあります。この質問に答えると、**×** マークが **✓** マークに代わります。全ての質問に答えたら、ウィザードを [終了] することができます。

質問間の移動は、 **戻る(B)** と  **次へ(N)** のボタンを使用します。

ウィザード・ダイアログ・ボックスの右下にある、上  と下  の矢印は、メッセージ・バーにメッセージが表示された場合に、複数のメッセージのスクロールに使用します。

使用可能なウィザードは、以下のとおりです。

- **1.7.1 LANSA Web モバイル アプリケーション** このウィザードは、LANSA Web モバイル・ブラウザ・アプリケーションを生成します。このブラウザ・アプリケーションは、自身のLANSA Web モバイル・アプリケーション内で使用できるコントロールのタイプを示すサンプルや、見出し/詳細のフォームを表示でき、リストのフィルター処理や表示もできます。
- **1.7.2 LANSA Web jQuery テーマの CRUD アプリケーション**
このウィザードは、LANSA やキー・フィールドに対する LANSA for the Web CRUD (作成(Create) 読み込み(Read) 更新(Update) 削除(Delete)) を完成し、生成します。これには以下のような機能が含まれます。
 - 選択されたサイト・レイアウトまたはデフォルトを使用した、jQuery テーマ。
 - 関連ファイルのドリルダウン。
 - アプリケーション・データの検索
- **1.7.3 Web アプリケーション レイアウト マネージャ ウィザード**
このウィザードにより、簡単にWeb アプリケーションのサイト・レイアウトのカスタマイズや生成ができます。これには以下のような機能が含まれます。
 - 色の設定と外観。
 - 複数のコンテンツ・エリア。
 - LANSA のウェブレットやjQuery ウェブレットを使用可能。
 - AJAX を使用した Web 2.0 サイト使用可能。

1.7.1 LANSA Web モバイル アプリケーション

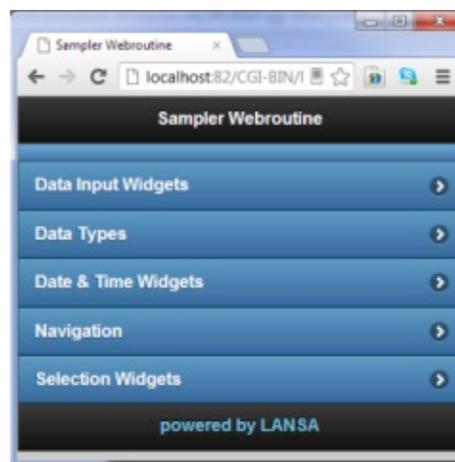
このウィザードを使用して、完全にカスタマイズ可能な jQuery Mobile の WAM を生成できます。質問に対する答えによりますが、生成された WAM には次のいずれかが含まれます。

- ユーザーが使用できるコントロールが含まれるサンプルの Web ページ。このページはアプリケーション全体用のものです。
- 以下の1つ以上を使用する、ユーザー定義の Web ページ
 - 見出し
 - テキスト・ブロック
 - イメージ
 - フォーム・エレメントこのエレメントには、一般的なフォーム・コントロールのコレクションが含まれています。
 - リンク
 - リスト・ビューこのコントロールには次のいずれかが含まれます。

- リスト・データ。これはRDMLから動的にロードされます。リストにつきリスト・データは1つのみです。別のリスト・コントロールを追加する場合は、別のリスト・データを指定できます。

- 分割線

- 静的メニュー・アイテム。これは以前に作成された Web ページの1つにリンクされます。



↑1.7 WAM ウィザード

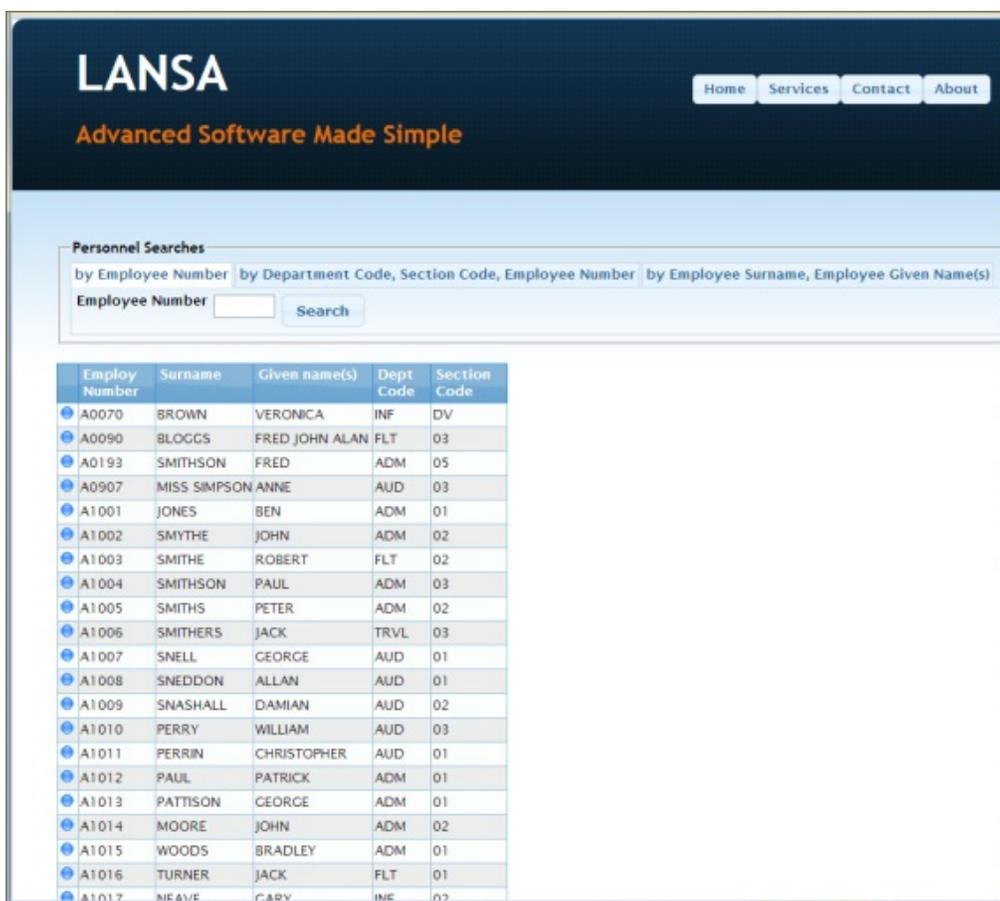
1.7.2 LANSA Web jQuery テーマの CRUD アプリケーション

このウィザードにより、ファイル管理処理を行う WAM と WAM レイアウトが生成できます。

生成される WAM を 読み取り専用にするのか、作成、更新や削除 処理を許可するのかを選択して、制限をかけることができます。

[レコードの追加] や [レコード詳細] のページに表示されるフィールド、および [検索結果リスト] ページに表示されるカラムはいずれもカスタマイズできます。アクセス経路の詳細が一致すれば、別の WAM にドリルダウンすることも可能です。

以下はファイル PSLMST をウィザードで処理した例です。



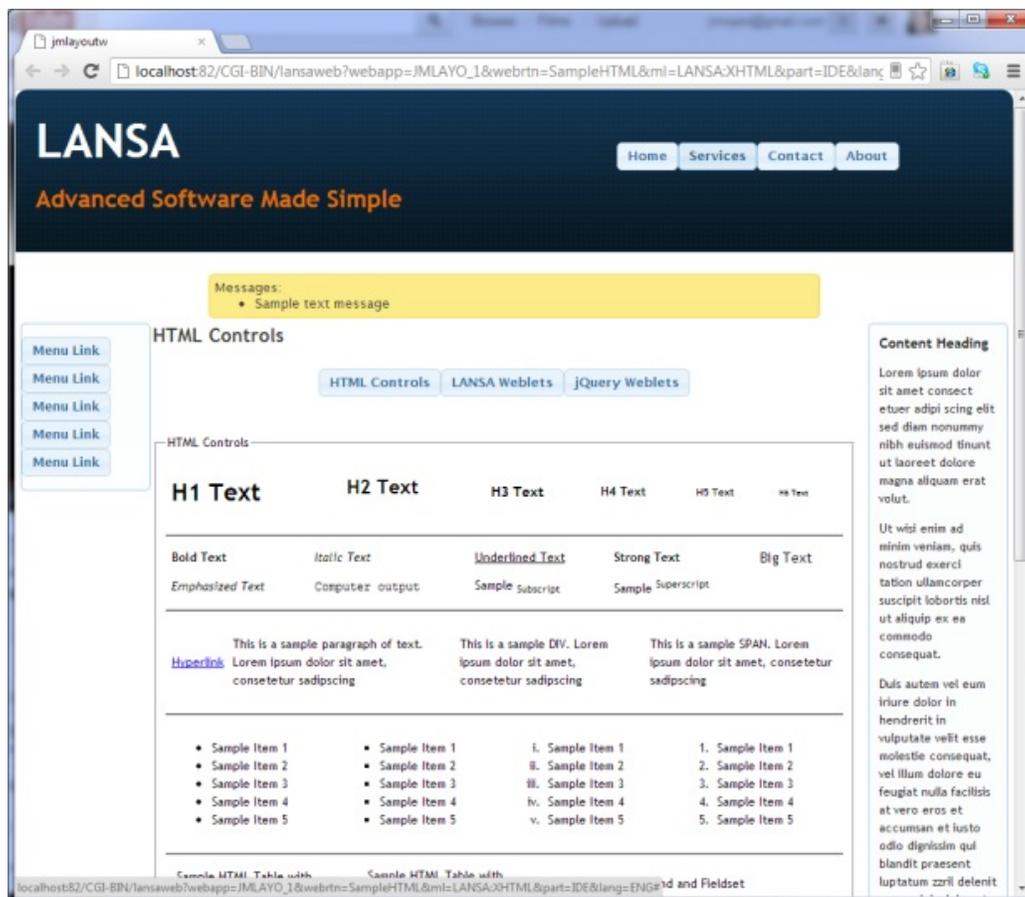
Employ Number	Surname	Given name(s)	Dept Code	Section Code
A0070	BROWN	VERONICA	INF	DV
A0090	BLOGGS	FRED JOHN ALAN	FLT	03
A0193	SMITHSON	FRED	ADM	05
A0907	MISS SIMPSON	ANNE	AUD	03
A1001	JONES	BEN	ADM	01
A1002	SMYTHE	JOHN	ADM	02
A1003	SMITHE	ROBERT	FLT	02
A1004	SMITHSON	PAUL	ADM	03
A1005	SMITHS	PETER	ADM	02
A1006	SMITHERS	JACK	TRVL	03
A1007	SNELL	GEORGE	AUD	01
A1008	SNEDDON	ALLAN	AUD	01
A1009	SNASHALL	DAMIAN	AUD	02
A1010	PERRY	WILLIAM	AUD	03
A1011	PERRIN	CHRISTOPHER	AUD	01
A1012	PAUL	PATRICK	ADM	01
A1013	PATTISON	GEORGE	ADM	01
A1014	MOORE	JOHN	ADM	02
A1015	WOODS	BRADLEY	ADM	01
A1016	TURNER	JACK	FLT	01
A1017	NEAVE	CARY	INF	03

↑1.7 WAM ウィザード

1.7.3 Web アプリケーション レイアウト マネージャ ウィザード

質問に対してユーザーが提供した答えに従い、このウィザードにより、Web サーバーのイメージ・フォルダ内のサブ・フォルダと、指定された要求に合うレイアウト・ウェブレットが生成されます。質問の答えによりますが、含まれるものは以下の通りです。

- Web サーバー・イメージのサブ・フォルダのスタイルを含む CSS ファイルとこれに一致する外部リソース・オブジェクト。
- Web サーバー・イメージのサブ・フォルダの追加JavaScript コードを含む JavaScript ファイル、およびこれに一致する外部リソース・オブジェクト。
- 生成されたレイアウトを使用したサンプル WAM。この場合、WAM レイアウトも作成され、生成されたレイアウト・ウェブレットを参照します。



詳しくは、WAM025 - Using the Layout Wizard を参照して、



ウィザードの手順を確認してください。

[↑1.7 WAM ウィザード](#)

2. WAMの構造

事前に読む必要があるもの

この章を読む前に、LANSAのWAMのアーキテクチャの概要を把握するためにも「[WAMの概論](#)」を読んでおいてください。

2.1 WAM、Webroutine、ウェブレット、ウェブレット・テンプレートの関係

「[WAMの概論](#)」を読んで、WAMやWebroutineとは何かというイメージが掴めたと思います。ですので、より深くウェブレットやウェブレット・テンプレートについて、またこれらがWAMやWebroutineとどのようにやり取りをしながらプレゼンテーション層を構築・カスタマイズするのかについて焦点を当てて説明していきます。

ここで、WAMの概念をいくつか確認してみましょう。これらの概念の多くは、既に馴染みがあることでしょう。

- 簡単にまとめると、各Webroutineは2つの部分から構成されています。RDMLXの部分は、アプリケーション・ロジックをカプセル化したものであり（アプリケーション・ロジック層）、XSL/XMLの部分はプレゼンテーション・インターフェースを定義しています（プレゼンテーション層、別名ウェブデザイン）。
- 1つのWAMにはWebroutineが1つ以上含まれます。
- 全てのWebroutineは、簡単なWEBROUTINE / ENDROUTINE構造を使ったWAMのRDMLXコードで定義されています。Webroutineの定義には、フィールドや作業リストの情報をプレゼンテーション層とやり取りするためのWEB_MAPコマンドを含むことができます。
- WAMのRDMLXコードにはメソッド・ルーチン(MTHROUTINE)とサブルーチン(SUBROUTINE)も含まれています。これらの概念をもう一度確認したい場合は、『LANSAテクニカルリファレンスガイド』を参照してください。
- Webroutineは通常、ユーザー・インターフェースとして表示されますが(例えば、ウェブ・ブラウザでのHTMLなど)、適切なテクノロジー・サービスが定義されている場合は、非ビジュアルなプレゼンテーションとして生成することもできます。
- 通常LANSAエディターでWebroutineを開くと、このWebroutine用に生成された拡張スタイルシート言語(XSL)オブジェクトが開かれ、これを確認したり、必要に応じて修正することができます。LANSAエディターでは、内部で定義されたLXML(XMLタグを表すリスト)、及びビルド中に生成されるXSL/XMLオブジェクトに基づいて初期表示が行われます。これらのオブジェクトについては、後ほどもう一度確認します。
- 1つのウェブレットにはウェブレット・テンプレートが1つ以上含ま

れます。ウェブレット・テンプレートは、ウェブレットがどのようにテクノロジー・サービスや、異なるXSLを必要とする他のバリエーション（インライン・リストなど）に適用されるかを決定します。

- Webroutineの表示は、LANSAエディターでWebroutineの[Web Page] タブ上にウェブレット・テンプレートをドラッグ・アンド・ドロップすれば修正することができます。
- ウェブレットテンプレートはテクノロジー・サービス固有のもので、つまり、一貫性のあるウェブレット一式がLANSAリポジトリで表示されている一方で、現在のテクノロジー・サービスを変更されると、使用可能なウェブレット・テンプレートのリストが修正されるということです。

以下のスケッチは、Webroutine（アプリケーション・ロジック）と、生成されたXSL、XMLや様々なウェブレット（プレゼンテーション層）との関係を示しています。

2.1.1 WAMのビルドおよびコンパイル

WAMのビルドやコンパイルのプロセスを確認しながら、開発プロセスに与える影響を見ていきたいと思います。

WAMがビルドもしくはコンパイルされたときに何が起こるかを確認する一番簡単な方法は、簡単なWAMを作成し、結果のオブジェクトを確認することです。

次のようなWebroutineが1つだけ(KWAM1001)の非常に簡単なWAM(KWAM10)を使用してみましょう。

```
FUNCTION OPTIONS(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_WAM)
  WEBROUTINE NAME(kwam1001)
    WEB_MAP FOR(*both) FIELDS(#deptment)
  ENDROUTINE
END_COM
```

初めてウェブデザインが生成される時には、適切なレイアウト・オブジェクトも生成されます。その後の生成ではレイアウト・オブジェクトは再生成されません。生成されたレイアウトに対する変更は全てLANSAエディターを使用して行います。

レイアウトについてはこのドキュメントで後ほど詳しく説明されますので、現時点では深く考えないでください。

WAMレイアウトに関連するオブジェクトは、以下の通りです。

- WAM用のレイアウト変数オブジェクトが1つ作成されます。 -
kwam10_layout.variables.xml
- WAM用のレイアウトXSLスタイルシートが1つ作成されます。 -
kwam10_layout.xsl

1つもしくは複数のテクノロジー・サービスと共に[XSLの生成]オプションを選択すると、ビルドまたはコンパイルの段階で、各Webroutineについて追加のXMLとXSLオブジェクトが生成、もしくは再生成されます。

RDMLX Webroutineコマンドのすぐ右側にある小さな緑の矢印を使って、特定のWebroutine用にこれらのXMLやXSLオブジェクトを生成または再生成することもできます。この一連のオブジェクトは区画の言語及び選択されたテクノロジー・サービスの組み合わせで次のディレクトリ構造を使って作成されます。... \X_WIN95\X_LANSA\X_<区画>\web\<プロバイダ>\<テクノロジー・サービス>\<言語>

[XSLの生成]設定に関連するオブジェクトは、以下の通りです。

- XSLスタイルシートが各Webroutineごとに作成されます。 - kwam10.kwam1001.xsl
- 変数ドキュメントが各Webroutineごとに作成されます。(これは、XSLオブジェクトに関連していて、内部的に格納されているLXML情報とは関係ありません。)- kwam10.kwam1001.variables.xml

ビルドを実行した場合、デフォルトのシステム設定である、全ての新しいWebroutine (つまり、これまでXSL/ XMLオブジェクトを生成したことがないWebroutine)に対するXSLの生成が適用されます。このプロセスはWAMのコンパイル・オプションによってさらに制御が可能です。

WAMをコンパイルするとき、[XSLの生成]オプションにより、XSL生成を避けたり、全てのWebroutine、もしくは新しいWebroutineにのみXSLを生成させることができます。RDMLX Webroutineコマンドのすぐ右にある小さな緑色の矢印を使って、1つのWebroutineに対してオンデマンドでXSLを生成することも可能です。

 全てのWebroutineに対して[XSLの生成]を選択すると、XSLが再生成されますが、これによりLANSAエディターで適用されたすべての修正が失われることを覚えておいてください。オンデマンドでXSLを再生成した際にも同じことが当てはまります。

 複数言語の開発を行っている場合...
デフォルトの区画言語に関連したXSLオブジェクトはLANSAリポトリで発行され、他の言語用に複製されます。このプロセスにより、各テクノロジー・サービスの全言語用のXSL情報が1セットずつ効率的に用意されます。各言語用に発行された別々のXSLを1セット作成することも可能ですが、各言語に対し全く異なるインターフェースが必要な場合を除き、一般的にはこのアプローチは推奨されません。複数言語変数を使用すると、1つのウェブデザインしか必要とせず、保守がより簡単で、より良いアプローチと言えます。

XMLとXSLオブジェクトの生成に加え、WAMがコンパイルされると (ビルド・オプションには適用されません)、WAMに関連するRDMLXオブジェクト一式が常に作成または再作成されます。その他のコンパイ

ル可能なRDMLXオブジェクトに対しても、同様のオブジェクト一式がWAM用に生成されます。例えば次のようなものです。

ダイナミックリンク・ライブラリ・オブジェクトkwam10.dllが
区画の実行ディレクトリ...\X_WIN95\X_LANSA\X_<区画>
>\executeに作成されます。

プログラムファイル、kwam10.pgmが、区画のソースディレクトリ...\X_WIN95\X_LANSA\X_<区画>\sourceに作成されます。

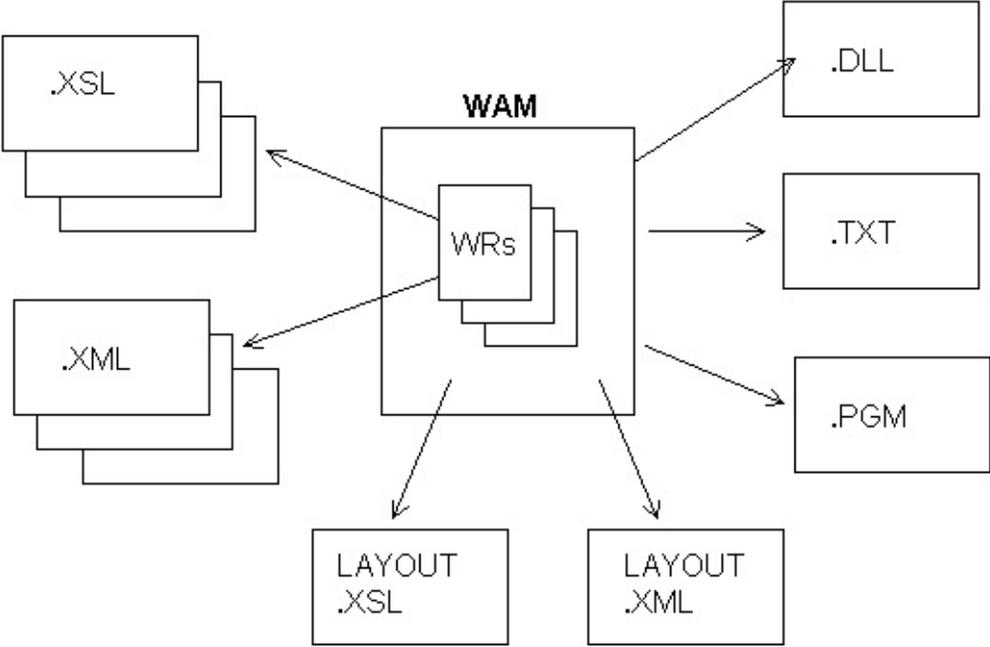
テキストファイル、kwam10.txtが、区画のソースディレクトリ...\X_WIN95\X_LANSA\X_<区画>\sourceに作成されます。

上記はWAMをサポートするために生成されるファイルですが、最後にあと1つ、全体を完成させるのに必要な部分があります。WAMがビルドもしくはコンパイルされると、Webroutineごとに、LXML(XMLタグを表すリスト)情報が常に生成または再生成されます。このLXML情報はLANSAデータベースに内部的に格納され、テクノロジー・サービスや言語の影響を受けません。

LXML情報が自動的に再生成されることは重要です。これはWAB_MAP定義に加えられたあらゆる修正を[Webroutine 出力]タブで使用できるようにするからです。LXMLはLANSAエディターで[XML]タブを選ぶことで確認できます。

生成されたLXMLに対する修正(LANSAエディターで加えられた、クッキーやTSMLノード)はLXMLが再生成された時も保持されます。

以下はWAMのビルド及びコンパイル・プロセスを簡潔に示した図です。下図で分かるように、複数のオブジェクトがWAM用に生成され、WAMの中の全てのWebroutineに適用されます。同時に、Webroutineごとにその他のオブジェクトが生成されます(そして情報が内部的に格納されます)。



2.1.2 ウェブレットとウェブレット・テンプレート

ウェブレットとは、共通のHTMLファンクションをカプセル化することでその複雑性を見えないようにデザインされた、XSLコードのスニペットです。ウェブレット・テンプレートは、そのウェブレットの中で更に細分化されたもので、特定のテクノロジー・サービスやその他想定される条件に当てはまるXSLコードのウェブレットのサブ・セクションを定義します。

すべてのウェブレットはLANSAリポジトリに格納されており、LANSAエディターで作成したり、開いたり、修正したりできます。ウェブレット、そして結果的にウェブレット・テンプレートも、再利用可能で、希望のプレゼンテーション層やウェブデザインを作成するために、あらゆるWebroutineのWebページ上にドラッグ・アンド・ドロップすることができます。(ほとんどの場合)ウェブレットとウェブレット・テンプレートのどちらをドラッグ・アンド・ドロップしても同じ結果を得ることができますが、選択されたウェブレットが現在のテクノロジー・サービスでサポートされていることを確実にするためにも、ウェブレット・テンプレートを使用するよう心掛けてください。ウェブレットをグループ分けすることで、使用する際に適切なウェブレットを探しやすくなります。例えば、インライン・リストを使用している場合、インライン・テンプレートというウェブレット・テンプレートのグループがあれば、現在のテクノロジー・サービスに対して「インライン認識」するように定義されているすべてのウェブレットが識別できるようになります。

LANSAは標準ウェブレット一式を提供しています。ウェブレットは、LANSAの全てのXSLやXMLオブジェクトと同様、テクノロジー・サービス固有のものです。出荷されたウェブレットのほとんどは、LANSAにより提供されているテクノロジー・サービス、つまりXHTML(eXtensible Hypertext Markup Language：拡張可能ハイパーテキスト・マークアップ言語)とPPC_XHTML(PocketPC XHTML：ポケットPC XHTML)のどちらにも対応しています。

通常ウェブレットを使ってデータをウェブ・ページ上にビジュアルライズします。例えば、提供された標準ウェブレットを使って、フィールドをチェックボックスやラジオボタンとしてフィールドをビジュアルライズしたり、作業リストをドロップダウンリストやツリーとしてビジュアルライズすることができます。Webページのレイアウトやメッセージ表示、メニュー用の標準ウェブレット、その他インターフェースのデータに直接関係しないけれども複数のウェブ・ページに共通するエレメント用の標準ウェブレットなども提供されます。追加でビジュアルライズされない

ウェブレットも提供されており、必要となる共通の情報、例えば変数やスタイルのウェブレットにアクセスできます。(詳しくは「[2.5 変数](#)」および「[2.7 カスケード・スタイルシート\(CSS\)とスタイル・ウェブレット](#)」に記述されています。)

2.1.3 ウェブレットの使用方法

ウェブレットをどのように使用してプレゼンテーション層を作り上げるかを理解するため、簡単な例を見てみましょう。

この例の目的は簡単な検索ページを作成することです。このページで部門コードが入力でき、そのコードで始まる部門のレコードが検索されます。

WAMを作成し、WAMSTARTと名前を付けます。



WAMが作成されると、Webroutineを作成するダイアログが表示されます。このダイアログをキャンセルして、WAMの編集を自分自身で開始することもできますし、ダイアログに詳細を入力して新しいWebroutineを作成し、デザイン段階に進んでRDMLXやWebデザインの対話処理によって自動的にRDMLXソースにWebマップを作成させることもできます。この例では、ダイアログを使用して、Webroutineを作成します。

新しいWebroutine

Webroutineの名前を入力し、作成のボタンを押して、Webページのデザインを開始してください

Webroutine 詳細

名前

記述

サービス名

作成

キャンセル(C)

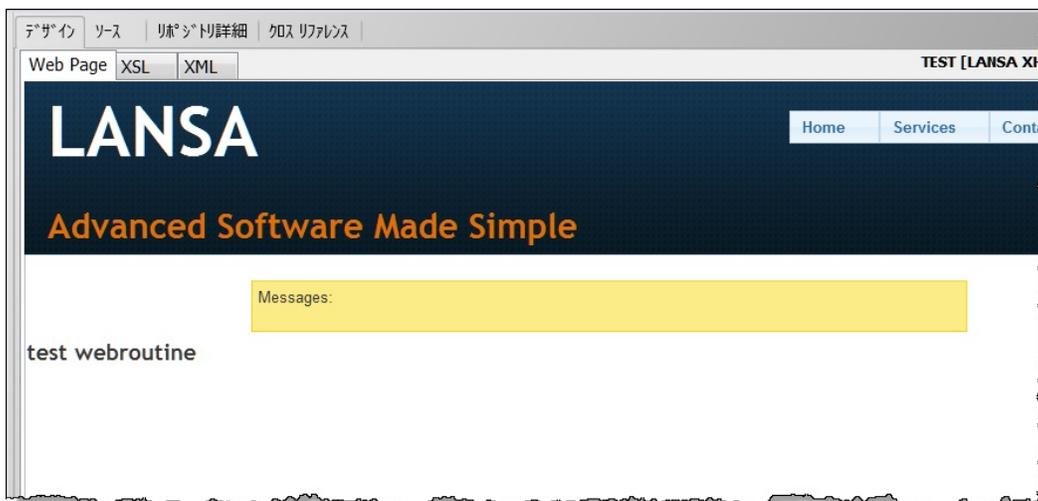
閉じる

Web ページ デザイン 詳細

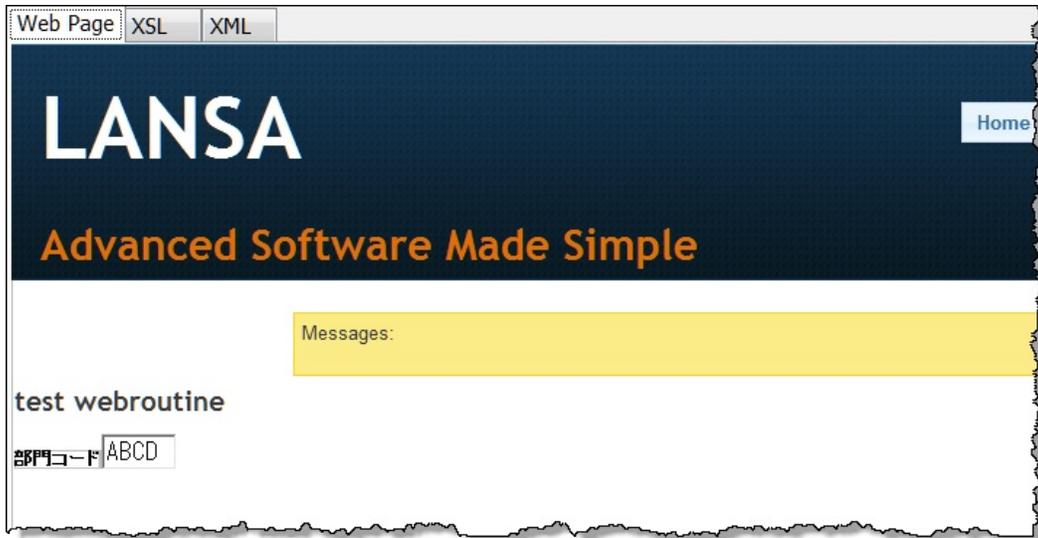
XSLの生成	デザインを開く	プロバイダ	テクノロジ サービス
<input type="checkbox"/>	<input type="checkbox"/>	LANSA	JQMOBILE
<input type="checkbox"/>	<input type="checkbox"/>	LANSA	PPC_XHTML
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	LANSA	XHTML

Webroutine の無い WAM を開いた時にこのダイアログを表示する

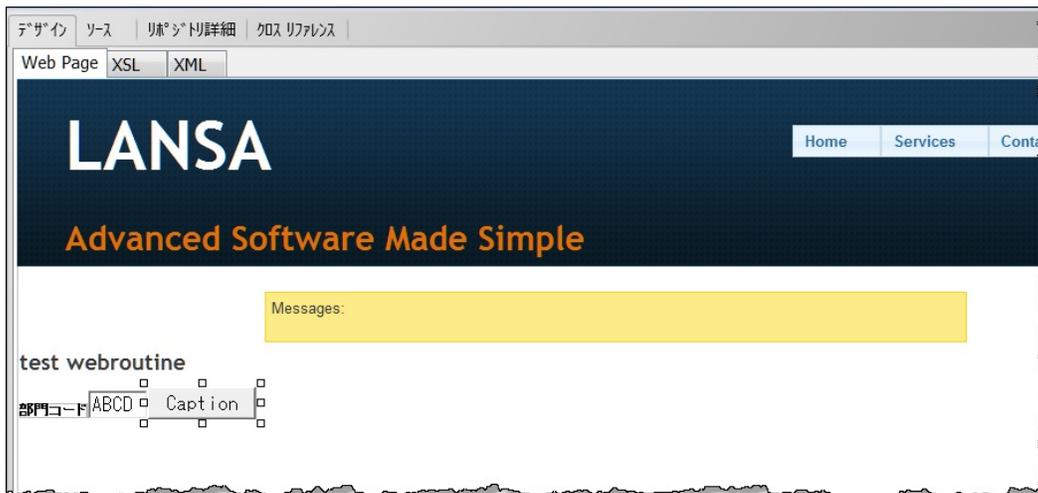
自動的にWebデザイン段階に進み、以下のウェブ・デザインが表示されます。



DEPARTMENTフィールドをリポジトリからドラッグし、ウェブ・デザイン上にドロップしてください。この操作により、RDMLXソースにWebマップも自動的に追加されます。



検索機能を付けるために、ページにプッシュボタンを追加する必要があります。リポジトリのウェブレット・テンプレートを開いて、プッシュ・ボタンのウェブレットを探します。このプッシュボタンをデザインビュー上の表示したいところにドラッグするだけです。



フォーカスが新しく追加されたプッシュボタン上にあることを確認してください。[詳細]タブを選択し、プッシュボタンのプロパティを確認します。

詳細	
プロパティ	
<xsl:call-template>	
name	std_button
パラメータ	
name	concat('o', position(), '')
caption	検索
currentrowhfield	STDROWNUM
currentrownumval	position()
reentryfield	STDREENTRY
reentryvalue	M
hide_if	False
formname	LANSA
pos_absolute_design	
width_design	
height_design	
on_click_wamname	<i>\$!web_WAMName</i>
on_click_wrname	deptsearch
protocol	
show_in_new_window	False
target_window_name	<xsl:if xmlns:xsl="http://www.w3.org/1999/XSL/Transform"></xsl:if>
disabled	False
title	
class	std_button
mouseover_class	std_button_mouseover

[caption]プロパティを検索に変更し、[on_click_wrname]プロパティをdeptsearchに変更してください。これは、仕上がったWebページでこのプッシュボタンをクリックすると、deptsearchWebroutine(現在のWAMで定義されているもの)が実行されることを示しています。現在このWebroutineはまだ定義されておらず、プッシュボタンを正しく実行するためには、これをWAMに追加する必要があることに注意してください。

この例で、必要に応じてデザイン上にフィールドをドロップしたり、プッシュ・ボタンのウェブレットにリファレンスを追加したり、プロパティをカスタマイズしたりすることで、kwam1001Webroutineのために生成されたウェブ・ページの外観や機能を修正することができます。

ビジュアルライズされないウェブレットも同様にできますが、ウェブレットのリファレンスが必要に応じて追加されていることを[XSL]タブで確認する必要があります。

WAMを実行する前に、コンパイルする必要がありますが、XSLを全てのWebroutineに対して生成しないよう注意してください。これを行うと、

ドラッグ・アンド・ドロップしたウェブレット・テンプレートに手動で加えたXSLに対する修正が上書きされてしまいます。

2.1.4 ウェブレットをいつ、どこで使用するのか?

次にビジュアライズしたウェブレットをプレゼンテーションに適用します。いつどこでウェブレットを使用するかという問題は、最終的なウェブ・ページに情報をどのように表示し、修正するのに関係するので、まさしく設計時の考慮事項となります。

例えば、以下のWebroutineを定義したとします。

```
DEFINE FIELD(#yesorno) TYPE(*char) LENGTH(1) DESC('Yes or No')
```

```
WEBROUTINE NAME(wam0190)  
  WEB_MAP FOR(*both) FIELDS(#yesorno)  
ENDROUTINE
```

そして、Webroutineをコンパイルして、XHTMLのデフォルトのプレゼンテーション・インターフェースを生成します。結果のウェブ・ページは以下ようになります。



マップされた値のデフォルトの表示は、入力可能フィールドであることを確認してください。

チェックボックスのウェブレット(もしくは、ブール値フィールド・ビジュアライゼーションのウェブレット)をフィールド値にドラッグするとデータの表示を変更でき、チェックボックスとして表示され応答するようになります。



 ヒント - #YESORNOフィールドを*BOOLEANタイプとして定義した場合、ビジュアル表示は自動的にチェックボックスになります。

LANSAとともに出荷されたビジュアルライズされたウェブレットの、各ウェブレットの一般的な使用方法も含めた詳細については、「[標準フィールドビジュアルイゼーション](#)」を参照してください。

ビジュアルライズされないウェブレットを使用するには、それぞれのウェブレットの目的をしっかりと理解する必要があります。LANSAとともに出荷された様々なビジュアルライズされないウェブレットの目的とアプリケーションについては、このドキュメントで後ほど説明します。

2.1.5 独自のウェブレットの作成

LANSAエディターのツールバーで[作成]メニューから[ウェブレット]を選択して独自のウェブレットを作成して、自身のサイトの基準を設定したり、共通に使用するXSLコードをカプセル化することができます。そして、ウェブレットを定義するための適切なXSLコードを追加することができます。通常、ウェブレットにはXMLを追加する必要はありません。XMLタブは、LANSAエディターではデフォルトでは表示されません。

自身で作成したウェブレット名には、std_ではない接頭辞を使用するようにしてください。ウェブレットの管理を簡素化するため、独自のウェブレットのグループを作成し、これを作成したウェブレットに割り当ててください。

ウェブレットと共に使用する予定の各テクノロジー・サービスごとにウェブレットのバージョンを作成する必要があることを忘れないでください。

 LANSAから提供されるウェブレットを修正しないでください。これらはその後のLANSAソフトウェア・アップグレードの際に置換されます。

 テンプレートとして標準のウェブレットを使用して新しいウェブレットを作成した場合、忘れずにファイル名とともにxsl:templateの名前を変更してください。同じxsl:template名の2つのウェブレットを同じレイアウトやWebroutine上で一緒に使用することはできません。

2.2 テクノロジ・サービス

WAMの開発に取りかかる前に、テクノロジ・サービスが何であるか、また、それがどのようにWAM開発に影響を与えるかを理解することが大切です。

2.2.1 テクノロジ・サービスとは?

テクノロジ・サービスはWAMのプレゼンテーション層もしくはウェブ・デザインに適用されます。これにより、共通のビジネス・ロジック(RDMLX)をいろいろなタイプのクライアント上に存在させることができます。これは、重要なコンセプトです。テクノロジ・サービスによって、1つのWAMのRDMLXを複数のテクノロジ形式で複数のデバイスに「表示」することが可能になり、アプリケーション・ロジックとプレゼンテーション・テクノロジを分離できるからです。

そのため、基本的にテクノロジ・サービスでは、Webroutineのプレゼンテーション出力を定義します。

2.2.2 どのテクノロジー・サービスを使用すればよいのか？

どのテクノロジー・サービスを使用するかは、ウェブ・ソリューションをどのように配布したいかにより異なります。

一番よく使用される2つのテクノロジー・サービスはVisual LANSAとともに出荷されており、WAMのプレゼンテーション層をサポートします。これは、標準のウェブ・ブラウザ・インターフェースをサポートするXHTML (eXtensible Hypertext Markup Language : 拡張可能ハイパーテキスト・マークアップ言語)、及び、手のひらサイズのデバイスのインターフェースのためにデザインされたPPC_XHTML (PocketPC XHTML : ポケットPC XHTML)です。

XHTMLやPPC_XHTML、またはJQMOBILE以外のフォーマットでウェブ・アプリケーションを表示したい場合は、これをサポートする独自のテクノロジー・サービスを作成する必要があります。

2.2.3 独自のサービスの作成

追加のテクノロジー・サービスを定義することができます。ただし、実行及び結果のXSLについてはすべて自己責任になります。

テクノロジー・サービスはLANSAエディタで定義されます。新しいテクノロジー・サービスを作成するには、ツールバーの[作成]ボタンを使用して、[テクノロジー・サービス]を選択してください。そして新しいテクノロジー・サービスの詳細を入力します。

新しいテクノロジー・サービスをサポートするために、独自のXSLスタイルシート・テンプレートのドキュメントを作成する必要があります。これらのドキュメントは、LANSAのディレクト

リ...\X_WIN95\X_LANSA\web\tspに保存してください。これらのテンプレートは、Webroutineの初期XSL表示を生成するために使用されます。

独自のテクノロジー・サービスの作成についての詳細は、「[テクノロジー・サービス](#)」を参照してください。

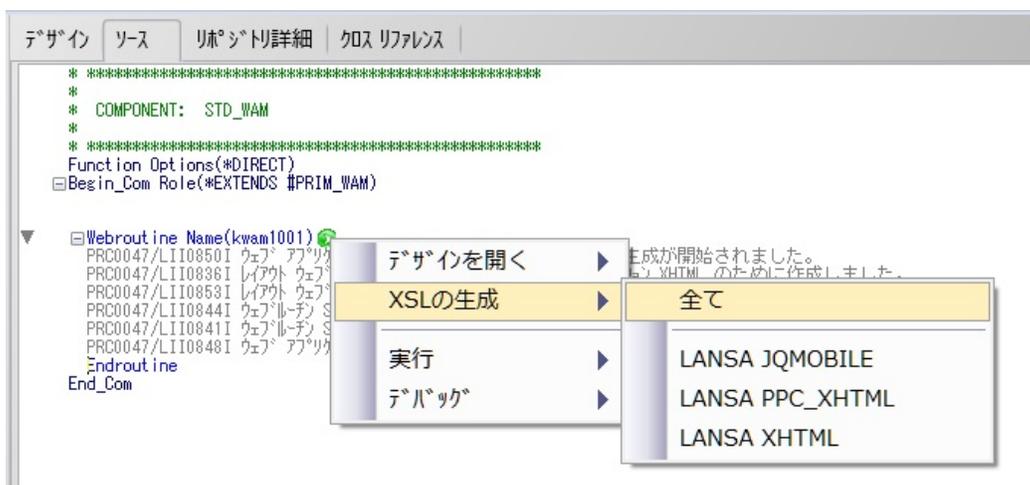


追加のテクノロジー・サービスをLANSAプロバイダに追加しないでください。LANSAの提供したテクノロジー・サービスは今後のバージョンで拡張もしくは変更される可能性があります。

2.2.4 特定のテクノロジー・サービスに対するXSLの生成

新しいWebroutineを作成する際に、既存のテクノロジー・サービスに対してXSLを生成するかどうかのオプションが選択できます。完成したウェブ・アプリケーションをウェブ・ブラウザ上のみで実行する場合は、PPC_XHTMLテクノロジー・サービスは選択しないでください。

その後の段階で異なるテクノロジー・サービスのXSLを生成したくなった場合は、Webroutineデザイン・グリフを右クリックして[XSLの生成]を選択し、必要なテクノロジー・サービス・プロバイダを選択することで可能になります。



2.2.5 特定のテクノロジー・サービスの表示の確認

LANSAエディターのデフォルトのテクノロジー・サービスはXHTMLです。LANSAエディターの[ウェブ]メニューを使用して、PPC_XHTMLインターフェース(生成済の場合)や作成した他のテクノロジー・サービス、およびこれに付随して生成されたウェブ・デザインを確認することができます。

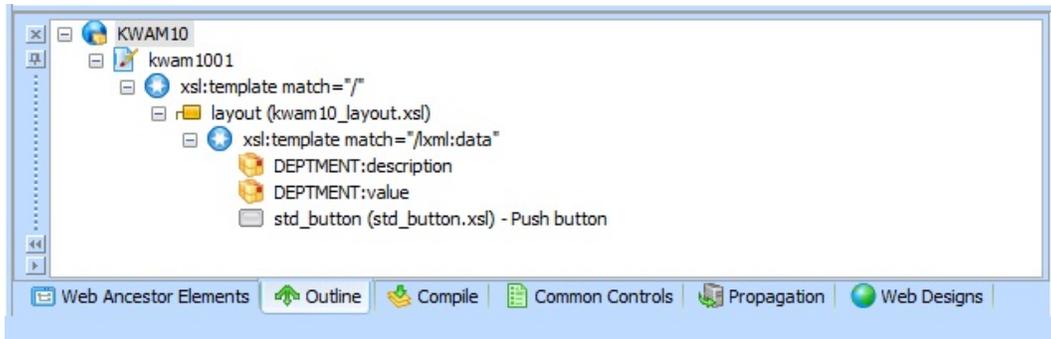
2.3 WebroutineのXSLの構造

これまでWAMのコンパイル時にどのようなオブジェクトが作成されるか、そしてこれらのオブジェクトを扱うテクノロジー・サービスの役割を説明してきました。ここからは、LANSAエディターで、Webroutine用に生成されたXSLを開いて、WAMの働きについてより深く見ていきたいと思えます。

同じWAM定義(KWAM10)を使用し、LANSAエディターのデザインタブで、 Webroutineデザイン・グリフをクリックするか、このグリフにマウスをあわせて右クリックし[デザインを開く]を選択した後、必要なテクノロジー・サービス・プロバイダを選択して、KWAM1001WebroutineのXSLオブジェクト（つまり、kwam10.kwam1001.xsl）を開きます。

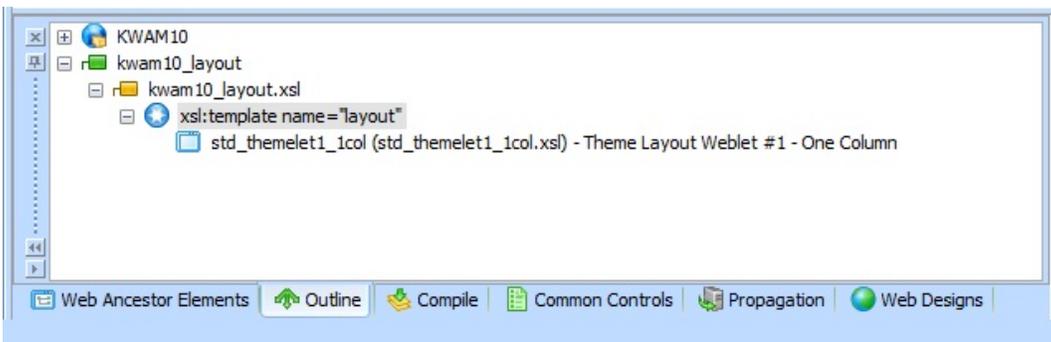


LANSAエディターでWebデザインが開いたら、[アウトライン]タブを選択します。LANSAエディターの[表示]メニューから[ビュー]、そして[アウトライン]を選択する、またはF6を押すことでこれを開く必要がある場合もあります。この表示を構成している、様々なウェブレット、HTML, XML や XSLを示すツリー・ビューが表示されます。このツリー・ビューの優れた特徴は、自動的に生成されたWAMレイアウト kwam10_layout.xslを参照していることです。またマップされたフィールドDEPARTMENTのフィールド定義及び値がXSL（ただし、ウェブレットではありません）に含まれます。さらにこのXSLには、ウェブレット std_buttonが含まれ、これがプッシュ・ボタンを表示します。



LANSA エディターの[アウトライン] タブでWAMレイアウトのウェブレット上でダブルクリック（もしくは右クリックして、コンテキスト・メニューから[ウェブレット: kwam10_layout] - [開く]を選択）することで、**kwam10_layout.xsl** ウェブレットを開くこともできます。このWAMレイアウトのツリー構造を見ると、さらに別のウェブレットが参照されているのがわかります。この場合は、**std_themelet1_1col**ウェブレットです。自動的に生成されたXSLオブジェクトにおいてさえも、ウェブレットが再利用されている様子が分かります。

このWAMレイアウト・ウェブレットに戻り、アウトラインのツリーを見てください。WAMレイアウトが**std_themelet1_1col**という名前の標準テーマを持つ1列のレイアウト・ウェブレットがベースになっていることが簡単に見てとれると思います。



[デザイン]タブで、これを確認することができます。



また、WAMレイアウト・ウェブレットはWebroutine固有の詳細情報(つまり、マップされたWebroutine情報)を参照しないということに注意してください。ページの構造や一貫性のあるインターフェースを提供するのはシェルです。

ここではWebroutineのXSLの構造を紹介する目的でstd_themelet1_1colウェブレットに基づいた例が使用されています。

提供されるレイアウト・ウェブレット(接頭辞がstd_のものは修正するべきではなく、常に独自のサイトのレイアウト・ウェブレットを構築することが推奨されています。このサイトのレイアウト・ウェブレットはもちろん独自のWebサイト用に修正することが可能です。独自のサイト・レイアウトを作成する簡単な方法は、[\[Web アプリケーション レイアウト マネージャ ウィザード\]](#)を使用することです。

例えばkwamsiteという名前の独自のサイト・レイアウト・ウェブレットを使用している場合、[\[アウトライン\]](#)のツリーは次のようになります。

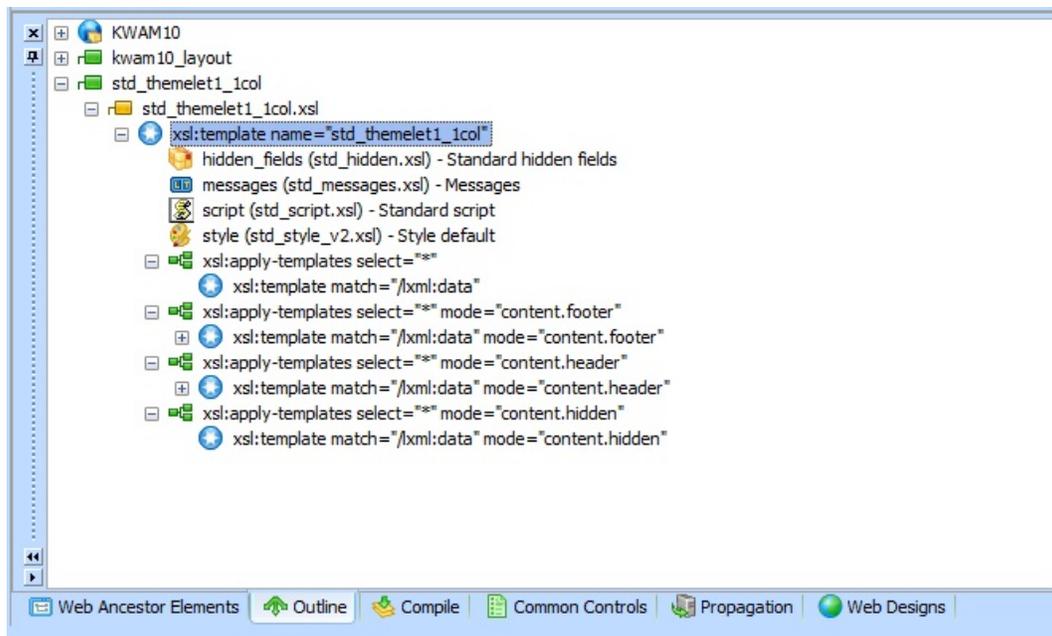


以下に続く説明では、サイト・レイアウトの使用時はstd_themelet1_1colのウェブレット名がkwamsite というウェブレット名に置き換えられています。[\[Web アプリケーション レイアウト マネージャ ウィザード\]](#)を使用してサイト・レイアウトが生成された場合は、これ以外の詳細は全て同じです。

引き続きWAMレイアウト・ウェブレットを見てください。[\[詳細\]](#)タブをクリックすると、ウェブレットkwam10_layout.xslのプロパティ設定が開かれます。これらは現在デフォルト値としてセットされています。これらのプロパティを変更すると、それに応じてレイアウトのインターフェースが変更されます。

Details	
kwam10_layout	
Properties	
<xsl:call-template>	
Name	Value
name	std_themelet1_1col
With Parameters	
window_title	\$window_title
has_form	\$has_form
show_title	\$show_title
title_text	\$title_text
width_type	\$width_type
width	\$width
javascript_files	\$javascript_files
jQueryNoConflict	\$jQueryNoConflict
css_files	\$css_files
output_charset	\$output_charset
backcompat_theme	\$backcompat_theme

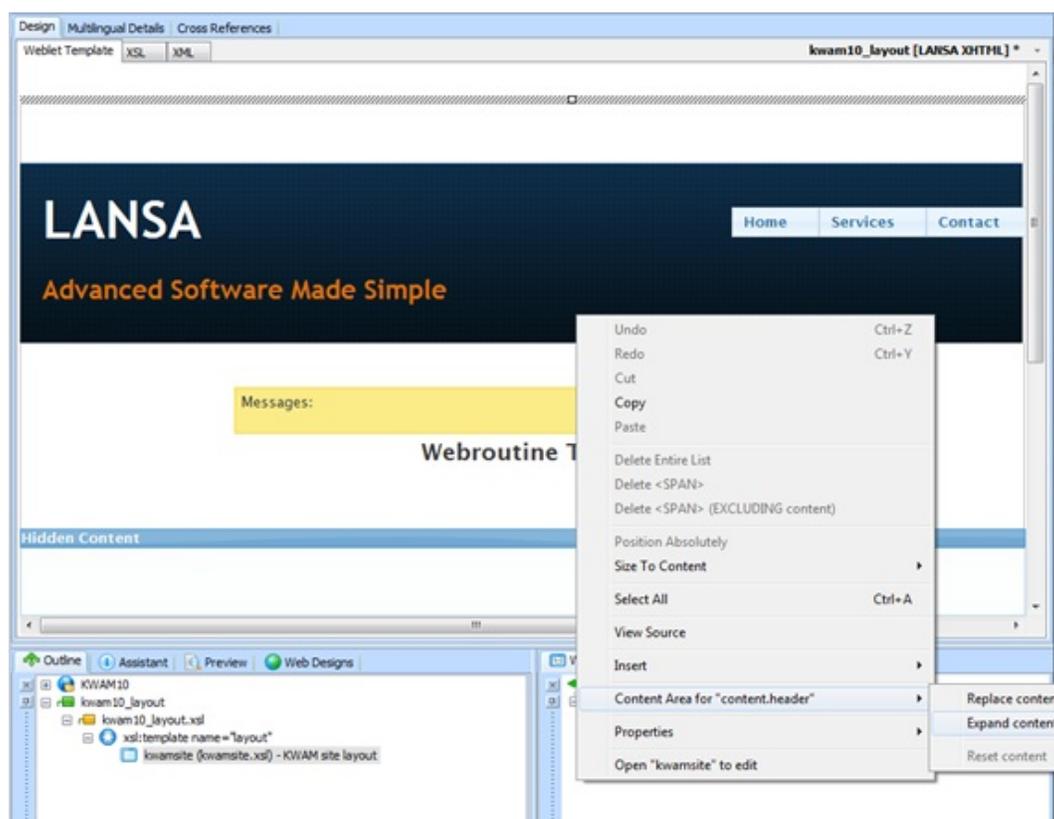
ここで、[アウトライン]タブに戻ると、ウェブレット内をドリル・ダウンしていくことで、基本的なWebroutinekwam1001がどのように構築されているかを調べることができます。WAMレイアウト・ウェブレットからstd_layout1_v2ウェブレット上でダブルクリックして、これをLANSAエディターで開くこともできます。提供された標準のウェブレットを更新してはいけないことに注意してください。



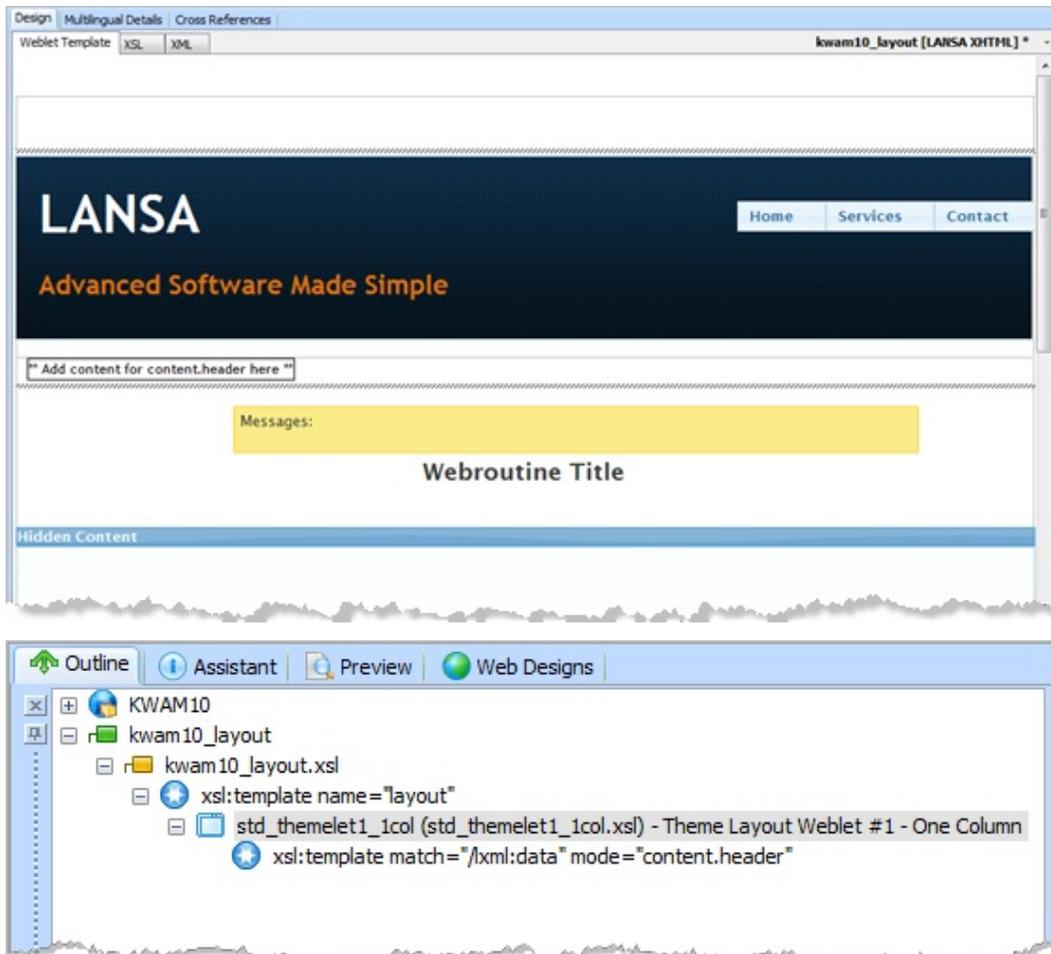
[アウトライン]タブ（または[デザイン]ビュー）でstd_themelet1_1col ウェブレットの構造を確認すると、**content.header**、**content.hidden**および**content.footer**というテンプレート名で識別されるページ・コンテンツ・エリアへの参照があります。これらの名前がkwam10_layoutの[アウトライン]ビューで特に表示されていなかったとしても、これらはkwam10_layoutとkwam1001Webroutineの両方のWebデザインで、LANSAエディターを使って編集可能なエリアです。

- [デザイン]ビューでstd_themelet1_1col ウェブレットを見直して、このウェブレットとWAMレイアウトkwam10_layoutとの関係を確認してください。
- 更新せずにstd_themelet1_1col ウェブレットを閉じて、WAM用のkwam10_layoutウェブレットの編集に戻ります。

ページ・コンテンツ・エリアや位置を修正するには、カーソルを修正するコンテンツ・エリア（ヘッダー、フッター、もしくは非表示）に置き、右クリックします。対象ページのコンテンツ・エリアのコンテキスト・メニューのオプションにより、コンテンツの置換や拡張が可能です。



コンテンツ・エリアが修正されると、WAMレイアウトkwam10_layoutの[アウトライン]ビューに表示されます。例えば次のようになります。



次にWebroutinekwam1001_layoutが関連する各ウェブレットによりどのように構築されているか詳しく見ていきます。

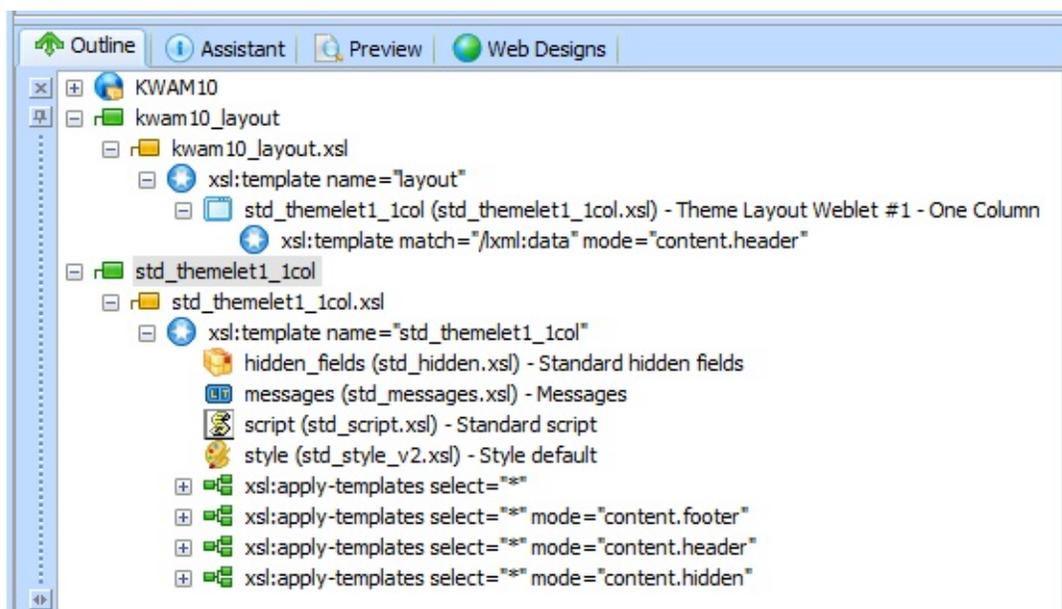
まず最初に、kwam1001_layoutウェブレットの[XSL]タブを開いて、XSLの一番上までスクロールしてください。コードを理解する必要はありませんが、他のXSLドキュメントを"インポート"するリファレンスを確認してください。

```
<xsl:import href="std_themelet1_1col.xsl" />  
<xsl:import href="std_types.xsl" />
```

これらのインポートは、kwam1001_layoutウェブレットが一連のウェブレット(つまり、XSLドキュメント)を参照していることを示しています。これらのウェブレットは、常にビジュアルライズされる訳ではありませんが、定義の中で重要なエレメントです。

kwam10_layoutWAMレイアウト・ウェブレットの[アウトライン]ビューから、std_themelet1_1colウェブレット上でダブルクリックして、LANSAエディターで開きます。更新された[アウトライン]ビューが展開されるので、std_themelet1_1colウェブレットが含まれていることを

確認します。



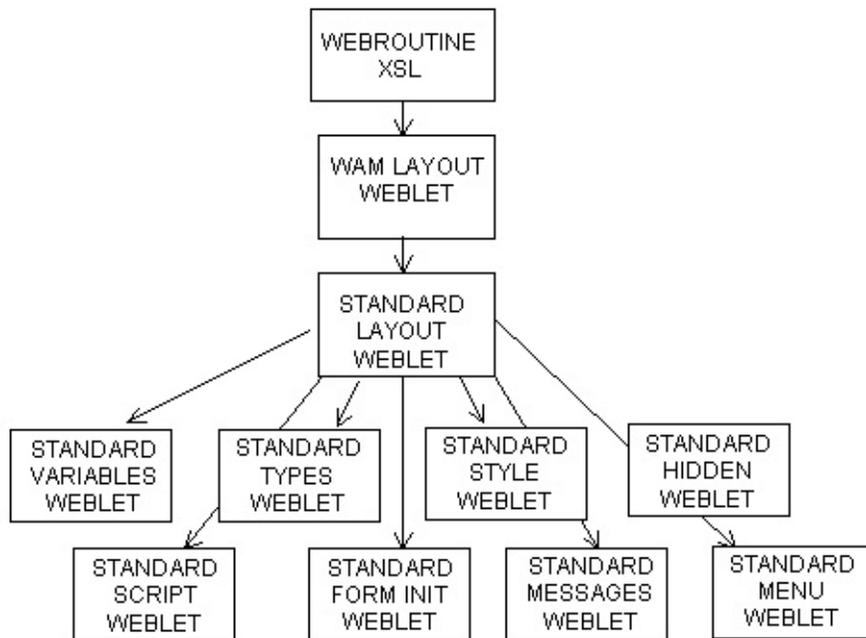
std_themelet1_1colの[XSL]タブを再度開いてXSLの一番上にスクロールして、他のXSLドキュメントをインポートするリファレンスを確認します。

```
<xsl:import href="std_variables.xml" />
<xsl:import href="std_types.xml" />
<xsl:import href="std_hidden.xml" />
<xsl:import href="std_style_v2.xml" />
<xsl:import href="std_script.xml" />
<xsl:import href="std_menubar.xml" />
<xsl:import href="std_messages.xml" />
```

これらのインポートは、std_themelet1_1colウェブレットが別の一連のウェブレット(つまり、XSLドキュメント)を参照していることを示しています。これらのウェブレットは、常にビジュアライズされる訳ではありませんが、定義の中で重要なエレメントです。

ここで覚えていなければならないことは、これらの異なるウェブレットが、提供される標準レイアウトの中で参照されていること、そしてそれが故に、この標準レイアウトを参照するその他のウェブレットでも"使用することができる"ということです。これらの様々なウェブレットが何を行うかについては、このドキュメントの次のセクションで説明されています。

以下のスケッチは、Webroutineの生成されたXSLの構造についてここで説明したことをまとめたものです。



この1つ1つの部分の詳細については、このガイド内で説明されています。

2.4 WAMレイアウトとレイアウト・ウェブレット

これまでWAMレイアウトおよびレイアウト・ウェブレットの概念について軽く触れました。ここで詳しく説明しましょう。

2.4.1 WAMレイアウト

2.4.2 レイアウト・ウェブレット

2.4.3 レイアウトが決定/制御するもの

2.4.4 WAMレイアウトをWAMに割り当てる方法

2.4.5 独自のサイト・レイアウトの作成方法

2.4.6 Webroutineにより使用されているWAMレイアウトの変更方法

2.4.7 WAMレイアウトに関連付けられているレイアウト・ウェブレットの変更方法

[Web アプリケーション レイアウト マネージャ ウィザード] を使って、レイアウトを簡単に作成することができます。

2.4.1 WAMレイアウト

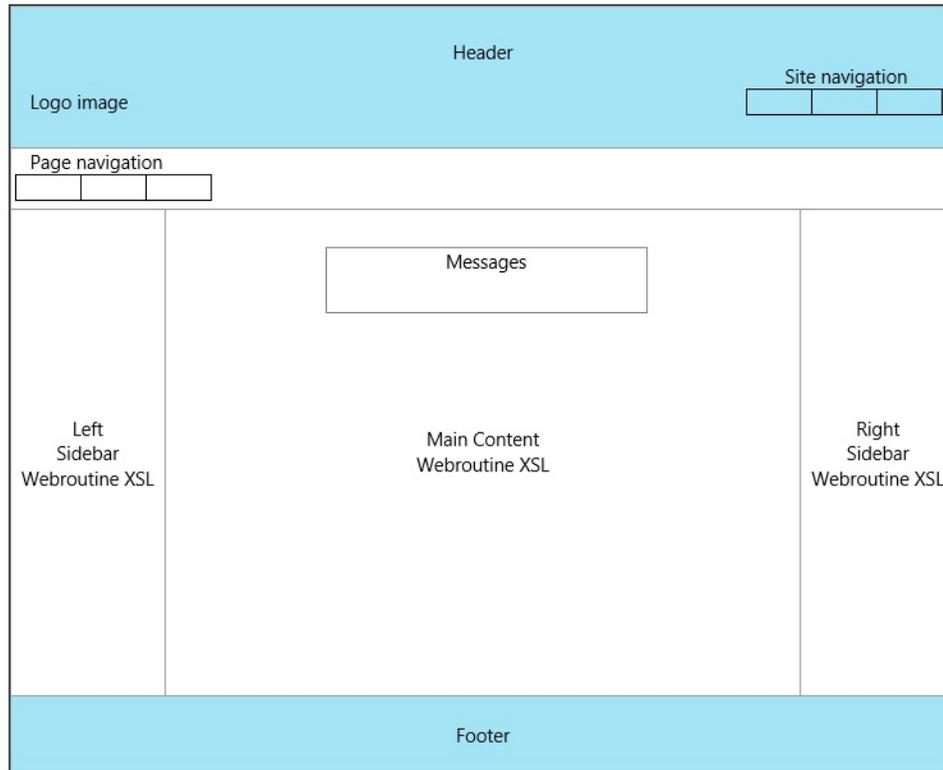
WAMレイアウト・ウェブレットは、Webroutineに関連付けられたWebページの構成を整えたり、機能や外観に関してレイアウト定義の中で参照されるドキュメントのインターフェースとして使用されます。

省略値では、各WAMには関連付けられたWAMレイアウト・ウェブレットがあり、これがWAMのWebroutineに関連する表示のベースとして使用されます。1つのWAMレイアウトが、WAMの中で定義されている

Webroutineの数に関係なく、それぞれのWAMに対して生成されます。

ウェブ・アプリケーションが複数のWAMを含む場合、同じレイアウトがアプリケーションの全てのWAMに対して適用されます。このようにして、一貫性のあるインターフェースを保つことができます。

WAMレイアウトは通常ビジュアル・エレメントとして、結果のウェブページ全てに構造を提供します。WAMレイアウトの役割として、タイトルやメニュー、メッセージ表示、ロゴなどの表示を定義することができます。WAMレイアウトは適用するカスケード・スタイルシートを制御することもできます。（これに関しては「[2.7.2 ロードされるCSSファイルの種類および独自のCSSファイルを追加する方法](#)」で説明されています。）



その名前からの推測に反し、WAMレイアウトは表示エレメントで構成される必要はありません。(ただし、通常は表示エレメントで構成されています。)

XSLドキュメントのリファレンスを含むレイアウトの非表示エレメントは、以下の通りです。

- 標準変数
- 標準データ・タイプ
- スタイル
- JavaScript
- 省略値の非表示フィールド

2.4.2 レイアウト・ウェブレット

多くのWAMレイアウトがあり、共通の要素を共有したい場合、この共通の要素をレイアウト・ウェブレットに置くことができます。その他のウェブレットと同様にレイアウト・ウェブレットの目的は機能の再利用、そして不要な重複を避けることです。ウェブレットの詳細については、『Visual LANSA ユーザーガイド』の「ウェブレットの作成」を参照してください。

3つのテーマのレイアウト・ウェブレット（テーマレット）がVisual LANSAで提供されています (std_themelet1_[1-3]col)。これらのレイアウトはWAM固有のレイアウトとして、もしくは独自のレイアウトを作成する開始点として使用できます。

自身のサイト・レイアウトを作成する簡単な方法は、[Webアプリケーションレイアウトマネージャウィザード]を使用することです。

新規WAMを作成する際は、サイトのレイアウト・ウェブレットを選択できます。自動的に作成されるWAMレイアウトは、指定したレイアウトを基に作成されます。

新しいWAM	
名前	SAMPLE
記述	WAM サンプル
レイアウト ウェブレット	std_wam_layout
フレームワーク	コントロール コンポーネント (コントロール)
グループ	
識別子	SAMPLE

2.4.3 レイアウトが決定/制御するもの

レイアウト（スタンドアロンのWAMレイアウト、もしくはWAMレイアウトに基づいたレイアウト・ウェブレット）は生成されたWebroutine表示の主要なエレメントです。これにより、一貫性のあるインターフェースを全てのWebroutineで使用できます。

LANSAエディターの[アウトライン]タブでWebroutineを表示すると、レイアウト・ウェブレットは通常、アウトライン・ツリーの一番上のレベルにあります。これは、ツリーの中でレイアウトの下にあるすべてのウェブレットが、レイアウト・ウェブレットで指定されたドキュメントを参照することができるということを示しています。

レイアウトによって制御されるものとして、例えば次のようなものがあります。

- メニューの外観
- 使用可能なメニューのオプション
- メッセージボックスの外観
- 適用されるカスケード・スタイルシート
- 共通のJavaScriptファンクションへのアクセス
- 非表示フィールド全体の定義
- 他のウェブレットで参照することができる標準変数定義
- ビジュアル・レイアウト適用の有無

もちろん、独自のレイアウトを定義した場合は、どの共通エレメントをインターフェースに含むかどうかを決定することができます。

2.4.4 WAMレイアウトをWAMに割り当てる方法

WAM固有のレイアウト・ウェブレットは、既に存在していない限り、WAMが最初にビルドもしくはコンパイルされた時に自動的に生成されます。

デフォルトでは、XSLが生成される際、プロセッサはWAM固有のレイアウト・ウェブレットがそのWAM用に既に存在しているかどうかを確認します。WAMレイアウトが存在しない場合、新しいWAMレイアウト・ウェブレットが生成され、WAMの名前の後に"_layout"を付けた形でリポジトリに格納されます。生成された後、WAM固有のレイアウトウェブレットは関連するWAM内の全てのWebroutineにより参照されます。WAM固有のレイアウト・ウェブレットに対する変更は全て、WAMの全Webroutineに反映されます。

WAMコンパイル時にXSL生成 オプションを選択しても、WAM固有のレイアウトは再生成されません。WAM固有のレイアウトが生成されるのは1度だけです。その後のWAM固有のレイアウトに対する修正や異なるレイアウトの割り当ては全て、LANSAエディタで行ってください。

2.4.5 独自のサイト・レイアウトの作成方法

独自のサイト・レイアウトを作成する一番簡単な方法は[Web アプリケーション レイアウト マネージャ ウィザード]を実行することです。

レイアウト・ウェブレットを一から作成するには、LANSAエディターの[作成]のツールバー・ボタンのドロップダウン・リストから[ウェブレット]オプションを選択します。[新しいウェブレット]ダイアログで、[レイアウト・ウェブレット]オプションを選択すると、レイアウト・ウェブレットが作成できます。

もしくは提供されているレイアウト・ウェブレットのいずれかをベースに使用して、自身のレイアウト・ウェブレットを作成することもできます。

2.4.6 Webroutineにより使用されているWAMレイアウトの変更方法

WAMエディターの[設計者]タブで、使用したいWAMレイアウトをドラッグ・アンド・ドロップします。これで既存のWAMレイアウトが置き換えられます。

2.4.7 WAMレイアウトに関連付けられているレイアウト・ウェブレットの変更方法

WAMに対するWAMレイアウト・ウェブレットが作成されると、このウェブレットは他のウェブレットと同様、開いて編集することができます。WAMレイアウトに関連付けられているプロパティを変更することも可能ですし、ベースになっているレイアウト・ウェブレットを置換して、代替りのレイアウト・ウェブレットをベースにすることもできます。

WAMレイアウト用にテンプレートとして使用するレイアウト・ウェブレットを変更するには、以下の手順で行います。

1. LANSAエディターでWAMレイアウト・ウェブレットを開きます。
[デザイン]タブを選択します。
2. [リポジトリ]タブで、レイアウトとして使用するウェブレットを探します。WAM固有のレイアウト・ウェブレット上にこの新しいレイアウト・ウェブレットをドラッグします。新しいレイアウトが古いレイアウトの下に表示されます。
3. 古いレイアウトを選択して、削除します。
4. この変更を保存します。



以上の手順を指定された順番で行うことが重要です。新しいレイアウト・ウェブレットをテンプレートとして使用するよう追加することで、すべての既存のオブジェクトがこの新しいテンプレートの下に移動します。このステップが完了したら、従属するオブジェクトを削除することなく、古いレイアウトテンプレートを安全に削除することができます。

2.5 変数

変数ウェブレット - std_variables - はビジュアライズされません。

std_variablesウェブレットの唯一の目的は一連の変数にデフォルト値を定義することです。これらのデフォルト値は順番に適切なウェブレットのプロパティで参照されたり、他のウェブレットのXSLソースで使用されたりします。std_variablesで定義されている変数が他のウェブレットに参照されるためには、参照するウェブレットにstd_variables.xslドキュメントを指定するインポートが含まれている必要があります。

例えば、std_menu_itemウェブレットはXSLソースに以下のステートメントが含まれています。

```
<xsl:import href="std_variables.xsl" />
```

XSLをさらに見ていくと、変数\$web_WAMName（std_variablesで定義）の参照があり、ランタイム時に実行されているWAM名を特定します。

2.5.1 出荷時の変数値を変更する方法

XSLの変数は他のプログラム環境で見かける変数とは異なります。XSLにおける変数という用語はプレースホルダーを意味する数学的な意味で使用されます。これはつまり、一度定義されたら、変数は変更ができないということです。

ただ、あるコンテキストで変数を上書きすることは可能です。これを正しく行うには、XSLTインポートの優先順位を十分理解している必要があります。std_variablesに定義された変数は、環境パラメータやlxmlソースのデータの特定の部分へのアクセスを提供します。変数のデフォルト値を変更する必要がある場合は、変数ではなくソース・データを変更してください。

独自の変数を作成したい場合は、独自のxxx_variablesウェブレットを作成し、それを必要に応じてレイアウトやウェブレットにインポートしてください。

std_variablesウェブレットを変更することはお勧めしません。これらは、将来のアップデートの際に変更される可能性があるからです。



Visual LANSAを再インストールする場合は、標準の出荷されたウェブレットに行われたすべての変更は、上書きされます。

2.5.2 独自の変数を作成する方法

独自の変数を作成する一番安全な方法は、独自のxxx_variablesウェブレットを作成し、それを新しい変数にアクセスしたいところにインポートすることです。

ガイドとして既存のstd_variablesウェブレットを使用することもできますが、std_variablesに既に定義されている変数を再定義しないようにしてください。これは、する必要がないだけでなく、予期せぬ動作を導く可能性があるからです。

独自の変数を作成しようとする前に、XSLTの知識と、変数が使用されている場所、方法をしっかり把握することが必要です。

2.6 ローカル変数

ローカル定義のウェブレット - `std_locale` - はビジュアライズされません。

`std_locale`ウェブレットの目的は、場所(地域設定)により異なる省略値を持つ、変数セットを定義することです。これらのデフォルト値は順番に適切なウェブレットのプロパティで参照されたり、他のウェブレットのXSLソースで使用されたりします。`std_locale`に定義されている変数を他のウェブレットで参照できるようにするには、参照するウェブレットが`std_variables.xsl`をインポートする必要があります。この`std_variables.xsl`自体が`std_locale.xsl`をインポートします。

このウェブレットは地域固有の値に設定された変数とともに異なる言語で出荷されています。

例えば、`std_style_v2`ウェブレットは変数`$lweb_std_css_language_overlay`を参照し、レイアウトに適用される、言語固有のスタイルシートの名前を取得します。

ローカル変数のカスタマイズしたバージョンを作成することも可能です。例えば、変数`$lweb_std_css_language_overlay`をカスタマイズしたい場合などです。

2.7 カスケード・スタイルシート(CSS)とスタイル・ウェブレット

std_styleウェブレットは、WAMに必要なCSSスタイルシート、およびユーザーにより（自身のレイアウト経由で間接的に）指定されたCSSスタイルシート、さらにWAMにより使用される外部リソースCSSスタイルシートを追加します。

std_styleウェブレットと外部のカスケード・スタイルシート定義を共に使用することで、Webページのインターフェース全体の外観を整えることができます。

それではまず基本から説明します。

2.7.1 カスケード・スタイルシートとその働き

2.7.2 ロードされるCSSファイルの種類および独自のCSSファイルを追加する方法

2.7.3 独自のスタイル・ウェブレットの作成方法

2.7.4 使用可能なカスケード・スタイルシート

2.7.1 カスケード・スタイルシートとその働き

カスケード・スタイルシートはどのようにページ・エレメントを表示するかをブラウザに伝えるものです。カスケード・スタイルシートの情報により、フォントや配色、ビジュアル効果、配列、枠線のサイズや色などが決定されますが、これを使ってイメージやインターフェースに関連したその他の機能を定義することもできます。これらのプロパティは、IDにより識別される個別のエレメント、またはタイプ、位置やクラスにより識別されるエレメントのグループに割り当てられます。

このドキュメントではCSSの詳細な説明は行われません。ただし、これについて詳細に述べられている本やオンライン・リソースはたくさんあります。オンラインの無料チュートリアル [W3Schools](#) からスタートするのも良いでしょう。

提供されているウェブレットの多くにはstyle(もしくはclass)プロパティが含まれています。プロパティに適用される省略値のスタイル及びドロップダウンリストから選択できるプロパティに関するスタイル式は、WAMに関連付けられたレイアウトで参照される元のCSSファイルに直接関連付けられます。



カスケード・スタイルシートは縮小されています(つまり多くのスペースが取り除かれています。)縮小されていないバージョンのファイルも同じディレクトリから提供されています。

LANSaはポケットPCのテクノロジー・サービスで使用するために特別に定義されたCSSファイルのセットも提供しています。

2.7.2 ロードされるCSSファイルの種類および独自のCSSファイルを追加する方法

std_style_v2ウェブレットによりCSSファイルのロードに必要な全ての<link>タグが作成されるので、独自のレイアウトの<head>にこれを含む必要があります。std_style_v2ウェブレットは常に各レイアウトにstd_style.min.cssをロードします。これはすべてのLANSA提供のウェブレットのテーマに関係しないプロパティを定義します。

そしてtheme_css_filenameとcss_filesプロパティにより定義されたCSSファイルを全てロードします。これらのプロパティは古いバージョンのウェブレットで作成されたレイアウトの下位互換性のために提供されたものです。新しいレイアウトの場合は、theme_css_filenameに'none'を指定し、外部リソースを使って一緒に含める追加のCSSファイルを定義します。

次にWebroutine内で参照される外部リソース（上記）として定義されたCSSファイル全て、およびWebroutineにより使用されるレイアウトやウェブレットが追加されます。

最後に、std_style_v2ウェブレットは変数\$!web_std_css_language_overlayにより定義されたスタイルシートをロードします。この変数は[std_locale](#)ウェブレットに定義されており、言語固有のCSS修正を適用する手段を提供します。

 外部リソースとjQuery UIのサポート以前は省略値のLANSAテーマはstd_styles.cssに含まれており、常にロードされていました。theme_css_filenameプロパティで追加されたカスタム・テーマはこの点を考慮に入れなければならない、省略値テーマから不必要なスタイルを取り除きます。LANSAバージョン 12 SP1以降は、この省略値テーマに関連付けられたスタイルは独自のCSSファイル(theme_default.css)に分けられています。ただし下位互換性のため、このテーマはtheme_css_filenameで指定されたテーマの前にロードされます。

theme_css_filenameに'none'を指定することで、theme_default.cssを取り除いて新しいテーマをブランクのキャンバスでスタートすることができます。

css_filesプロパティは、各Webroutine用のメカニズムを提供し、

特定のCSSファイル（例えばカスタム・ウェブレットが必要なCSSなど）をページに追加します。これを行う場合、外部リソースは更にパワフルなメカニズムで、ウェブレットで独自のCSSに必要なものを定義でき、std_style_v2を通して自動的に通信できます。css_filesプロパティは現在は不必要ですが、下位互換性を保つための目的で含まれています。

注：

std_style_v2への引数に引き渡されるCSSファイル名はスタイルのサブ・ディレクトリに関連付けられていると想定されます。スタイルのサブ・ディレクトリの値は変数\$web_style_pathにより定義されます。省略値ではこの変数はウェブサーバーのイメージ・ディレクトリ直下のサブディレクトリ/styleを参照します。この値を変更しないことをお勧めします。

バージョン 13.0 以降のWAMの出力はUTF-8です。css_filesプロパティの<link>エレメントにはcharset属性がありませんでした。ですからメイン・ドキュメントと同じ文字セットだと仮定されていた。下位互換性を保つため、これに現在はcharset属性が加えられ、言語JPN（日本語）の場合は"shift_jis"、その他の全言語は"iso-8859-1"が省略値です。異なる文字セットを指定する場合は、ウェブレットstd_style_v2のcss_files_charsetプロパティを使用します。（自身のサイト・レイアウトにこのパラメータを追加する必要があります。）これよりも良いアプローチとしては、別途独自のCSSファイルを外部リソースとして登録し、これを含める方法があります。

2.7.3 独自のスタイル・ウェブレットの作成方法

標準スタイル・ウェブレットでニーズのほとんどに対処することができますが、独自のスタイル・ウェブレットを作成したい場合は作成することもできます。標準スタイル・ウェブレットはLANSAが提供するウェブレットやレイアウトを正しく操作するのに必要な機能を提供しています。カスタム・スタイル・ウェブレットの中ではこれを必ず呼び出すようにするか、以下の例に示すように、ウェブレットが標準スタイル・ウェブレットに沿って動くよう設計する必要があります。

```
<xsl:import href="std_style_v2.xsl" />

<xsl:template name="my_style">
  <xsl:call-template name="style">
    <xsl:with-param name="theme_css_filename"/>
    <xsl:with-param name="css_files"/>
  </xsl:call-template>

  <!-- Custom style functionality here -->
</xsl:template>
```

「style」というテンプレート名を使用することはできません。カスタム・ウェブレットのテンプレートを「style」と名づけた場合、無限ループを引き起こします。

2.7.4 使用可能なカスケード・スタイルシート

「[2.7.1 カスケード・スタイルシートとその働き](#)」で触れられているメインのCSSスタイルシートはイメージ・ディレクトリの下にあるスタイル・ディレクトリに入れられています。

またテーマのCCSスタイルシートは、このテーマの名前の付いたサブ・ディレクトリの下にjQueryサブ・ディレクトリに入れられています。

WAMで利用可能なその他のスタイルシートのリストは、「[LANSAで提供される外部リソース](#)」で説明されています。

2.8 Javascriptとスクリプト・ウェブレット

2.8.1 独自のスクリプト・ウェブレットの作成方法

2.8.2 ウェブレットのプロパティ用のインラインJavaScriptをフォーマットする方法

スクリプト・ウェブレット - std_script - はビジュアルライズされません。

std_scriptウェブレットは多くの外部のJavaScriptファイルをロードし、LANSAのウェブレットで使用される多くのJavaScript変数やファンクションを初期化します。これはすべてのレイアウトのHEADセクションに含まれていなければなりません。

参照される外部のJavaScriptファイルは、ウェブサーバー上にインストールされる少数のJavaScriptファイルで、WAMをサポートします。これらのスクリプトはイメージ・ディレクトリの下にあるサブディレクトリ/scriptに直接ロードされます。以下のJavaScriptファイルが提供されています。

- std_script.min.js
- std_script_v2.min.js
- std_script_messages.min.js
- std_script_messages_fra.min.js
- std_script_messages_jpn.min.js
- std_script_lansa_ppc_xhtml.min.js
- std_script_lansa_ppc_xhtml_v2.min.js

std_script.min.js、std_script_v2.min.js、std_script_lansa_ppc_xhtml.min.jsおよびstd_script_lansa_ppc_xhtml_v2min.jsは提供される全てのウェブレットに必要な、核となるJavaScriptコードを提供します。

Std_script_messages.min.jsは、ウェブレットによってユーザーに表示される全てのメッセージを定義します。また、std_script_messages_fra.min.jsとstd_script_messages_jpn.min.jsはそのメッセージの翻訳版を提供します。



これらのファイルは縮小されています（つまり多くのスペースが取り除かれています。）縮小されていないバージョンのファイルも同じディレクトリ内で提供されています。

JavaScriptメッセージを独自にローカライズして提供したい場合は、

std_script_messages.min.jsをコピーし、メッセージを翻訳してください。そしてこの言語固有のstd_localeウェブレットを編集し、lweb_script_messages_fileとlweb_script_messages_file_charsetの変数を更新します。

ページまたはレイアウトに独自のJavaScriptファイルを追加するには、自身のJavaScriptファイルをWebイメージの外部リソースとして登録します。この外部リソースを必要なWebroutineまたはウェブレットに追加します。対応する<script>エレメントが実行時に自動的に追加されます。

注：

1. 全てのレイアウトにはjavascript_filesプロパティがあり、これはstd_scriptウェブレットに渡されます。このパラメータは、カンマで区切られたファイル名のリストを使用します。(/scriptディレクトリに関連付けられていると仮定されます。)
2. バージョン13.0以降のWAMの出力はUTF-8です。 javascript_filesプロパティの<script>エレメントにはcharset属性がありませんでした。ですからメイン・ドキュメントと同じ文字セットだと仮定されていました。下位互換性を保つため、これに現在はcharset属性が加えられ、言語JPN（日本語）の場合は"shift_jis"、その他の全言語は"iso-8859-1"が省略値です。異なる文字セットを指定する必要がある場合は、ウェブレットstd_scriptのjavascript_files_charsetプロパティを使用します。（自身のサイト・レイアウトにこのパラメータを追加する必要があります。）これよりも良いアプローチとして、別途独自のJavaScriptファイルを外部リソースとして登録し、これを含める方法があります。

提供されているウェブレットの多くにも、XSLの中に小さなインラインのJavaScriptファンクションがあります。

2.8.1 独自のスクリプト・ウェブレットの作成方法

標準スクリプト・ウェブレットでニーズのほとんどに対処することができますが、独自のスクリプト・ウェブレットを作成することもできます。標準スクリプト・ウェブレットではLANSAが提供するウェブレットやレイアウトを正しく操作するのに欠かすことのできない機能が提供されています。カスタム・ウェブレットの中ではこれを必ず呼び出すようにするか、以下の例のようにウェブレットを標準スクリプト・ウェブレットに沿って動くよう設計する必要があります。

```
<xsl:import href="std_script.xml" />

<xsl:template name="my_script">
  <xsl:call-template name="script">
    <xsl:with-param name="javascript_files"/>
    <xsl:with-param name="trap_script_errors"/>
  </xsl:call-template>

  <!-- Custom script functionality here -->
</xsl:template>
```

「script」というテンプレート名を使用してはいけません。カスタム・ウェブレットのテンプレートを「script」と名づけた場合、無限ループを引き起こします。

2.8.2 ウェブレットのプロパティ用のインラインJavaScriptをフォーマットする方法

JavaScript値が有効、つまり予想された値である場合のウェブレット・プロパティは、単一引用符でくくられた、インラインJavaScriptを受け取ることができます。JavaScriptのフォーマットはユーザー自身の責任で行います。

全てのインラインJavaScriptはセミコロン(;)で終わらせる必要があります。例えば、`'alert("hello world");'`のようになります。JavaScriptを単一引用符で囲まなくてはいけないため（これはバックグラウンドで行われます）、プログラム中で単一引用符を使用できません。使用する必要がある場合は、外部のJavaScriptファイルの中にファンクションを作成し、インライン・コードからこれを呼び出すという方法があります。

JavaScriptを必要とするプロパティの多くは、イベントに応じて実行されるか、何らかのアクションを実行するために先行して実行されます。例えば、std_buttonウェブレットのpresubmit_jsプロパティは、フォームがサーバーに送られる直前に実行されます。これにより、いくつかプロセスを追加したり、イベント/アクションをキャンセルすることができます。イベント/アクションをキャンセルするには、`"return false;"`を使用します。

例えば次のようになります。

```
if ( confirmWithUser() == false) return false;
```

(confirmWithUserは外部のJavaScriptファイルに定義されたファンクションです。)



JavaScript 注意事項:

1:インラインJavaScriptでreturnやreturn trueを使用しないでください。これはLANSAのJavaScriptの実行を停止するというreturn falseと同様の効果がありますが、ブラウザが省略値のイベント処理を実行するのは停止しません。これにより、予期せぬ動作が引き起こされる可能性があります。

2:{ および }の文字は、XSLT内で特別な意味があり、JavaScriptプロパティでは使用できません。使用すると、奇妙な動作を引き起こします。この文字を必要とする複雑なJavaScriptを書く必要がある場合は、JavaScriptファンクションを別途作成し、プロパティからこれを呼び出すようにしてください。

3:以前のバージョンのドキュメントでは、JavaScriptをダブル・バックスラッシュ(//)で終わらせて、省略値の処理をキャンセルするよう説明されていました。このテクニックはreturnを使用するのと同じ効果があり、使用すべきではありません。

2.9 メッセージ

メッセージ・ウェブレット - `std_messages` - はWebページ上のアプリケーション・メッセージの表示をフォーマットします。

メッセージ表示は提供されている全ての標準レイアウトの最初に自動的に組み込まれます。つまり、通常は`std_messages`ウェブレットを確認したり、修正したり、自身の表示に適用させる必要さえないということです。



`show_messages` プロパティは提供される各レイアウトに含まれており、`std_messages`により生成されたメッセージ・ボックスをレイアウトに組み込む必要があるかどうかを示します。このプロパティの省略値はメッセージを表示するように設定されています。メッセージを表示したくない、もしくは異なるウィンドウにメッセージを表示したい場合は、このプロパティを`false`に変更しなくてはなりません。

背景色などのメッセージ・ボックスのインターフェースの基本的な外観は、適切なスタイルシートのクラスを自身のCSSファイルで再定義すると修正できます。もしくは、メッセージをまったく異なる表示にしたい場合は、独自のバージョンの`std_messages`ウェブレットを作成し、必要に応じて自身の表示インターフェースでこれを参照します。

2.10 タイプ

タイプ・ウェブレット - std_script - はビジュアルライズされません。

std_typesウェブレットは、ウェブレットのプロパティとして入力できる情報のタイプを定義します。タイプは属性wd:typeを提供することでXSLソースの中で宣言されます。この属性でタイプ名を割り当て、このタイプに有効な情報タイプを詳述します。

例えば、以下のコードをstd_typesドキュメントの中で見つけてください。

```
<wd:type name="std:border_style">
  <wd:enumeration value="'dashed'" />
  <wd:enumeration value="'dotted'" />
  <wd:enumeration value="'double'" />
  <wd:enumeration value="'groove'" />
  <wd:enumeration value="'inset'" />
  <wd:enumeration value="'outset'" />
  <wd:enumeration value="'ridge'" />
  <wd:enumeration value="'solid'" />
  <wd:enumeration value="'window-inset'" />
</wd:type>
```

このタイプstd:border_styleを参照するウェブレットのプロパティはいずれも境界線のスタイルに関連するもので、wd:enumerationステートメントの値が有効値として含まれることは、このタイプ定義から簡単に推測できると思います。

これを確かめるには、パネル(std_panel)ウェブレットをWebページに追加し、 Details タブを選択して関連するパネル・プロパティを確認します。borderプロパティに関連付けられたドロップダウン・リストで使用可能な値を確認してください。想像通り、ドロップダウンの値はwd:enumerationステートメントと一致します。

Properties	
<xsl:call-template>	
Name	Value
name	std_panel
With Parameters	
name	concat('o', position(), '_LANSA_31660')
panes	document('')/*[xml:data/xml:panes[@id=
border	
border_width	dashed
hide_if	dotted
class	double
snap_to_grid	groove
grid_size	inset
pos_absolute	outset
width	ridge
height	solid
	400pt
	200pt

次にstd_panelウェブレットを開き、XSLソースを確認して、ウェブレットのプロパティとタイプの関係がどのように確立されるかを見ていきます。

まず最初に、std_typesドキュメントがstd_panelウェブレットのXSLソースにインポートされることを確認してください。

```
<xsl:import href="std_types.xsl" />
```

そしてXSLソースを追っていくと、ウェブレットのプロパティとタイプ定義の間関係が定義されているのが分かります。

```
<xsl:template name="std_panel">
...
<xsl:param name="border" wd:type="std:border_style" />
```

これですべてがひとつにまとまります。



ヒント：独自のウェブレットを定義している場合、std_typesドキュメントに定義されたタイプを参照して、どの値が自身のウェブレット・プロパティで有効かを指定します。独自のタイプを作成する必要はありません。

2.11 非表示

非表示フィールド・ウェブレット - std_hidden - はビジュアルライズされません。

このウェブレットにより、内部的に定義され評価されるプロパティ・グループにアクセスできるようになります。それぞれのプロパティに割り当てられる値はRDMLX定義の中で決定され、ウェブ・ブラウザで Webroutine を実行するのに必要になります。

標準的な非表示の情報は、以下の通りです。

SERVICENAME

WEBAPP

WEBROUTINE

PARTITION

LANGUAGE

SESSIONKEY

LW3TRCID

非表示の情報は、Webroutine を呼び出す URL に含まれる値に直接関係しています。例えば次の URL を使って、WAMEX02 という WAM の中の MaintainRecord という Webroutine をウェブ・ブラウザから直接実行することができます。

http://<server>/cgi-bin/lansaweb?

wam=WAMEX02&webtrtn=MaintainRecord&part=DEX

もしくは、この代わりに WAMEX02 Maintenance というサービス名が Webroutine に割り当てられたとすると、URL は次のようになります。

http://<server>/cgi-bin/lansaweb?

srve=WAMEX02_Maintenance&part=DEX

どちらの例においても、言語やトレース、その他の宣言されていないパラメータ値はいずれも省略値であると仮定されます。

2.12 キー

キー・ウェブレット - `std_keys` - はビジュアライズされません。

標準キー・ウェブレットは指定されたキー一式を宣言し、他のXSLドキュメントでこれらを使用して、複雑なXMLドキュメントに簡単にアクセスできるようにします。キー情報を使用するには、XSLとXMLをしっかり理解する必要があります。

このウェブレットは多くのウェブレットにインポートされ、それらのウェブレットの鍵となる機能の使用をサポートします。

参考資料

WAMとその使用方法に関する詳しい情報は、[「ウェブレットとウェブレット・テンプレート」](#)を参照してください。

2.13 インライン・リスト

原理

ウェブレットのXSLテンプレートは、WAMの開発者が結果をカスタマイズできるように、大量にパラメータ化されています。

例えば、std_anchorウェブレットは、マウスがその上を通ると表示を変えることができます。開発者は`mouseover_class`プロパティに値を割り当てると、この動作を実現することができます。以下のXSLTは実行時に、プロパティがセットされているかを確認し、そうであるならば、アンカーに`onmouseover/onmouseout`イベントハンドラーを追加します。

```
<xsl:if test="$mouseover_class != "">
  <xsl:attribute name="onmouseover">
    <xsl:text>this.className='</xsl:text>
    <xsl:value-of select="$mouseover_class" />
    <xsl:text>'</xsl:text>
  </xsl:attribute>
  <xsl:attribute name="onmouseout">
    <xsl:text>this.className='</xsl:text>
    <xsl:value-of select="$class" />
    <xsl:text>'</xsl:text>
  </xsl:attribute>
</xsl:if>
```

Webroutineのデザインが保存されたら、`$mouseover_class`の値は変わりませんが、このコードはWebroutineが稼動するたびに実行されます。ウェブレットがリスト内にある場合は、リストの各列に対し、このコードが再度実行されます。

多くの場合、パラメータ(ウェブレット・プロパティ)の多くは定数で、実行時の値に左右されません。ですからこれらのプロパティを実行時に毎回適用するのではなく、設計時に適用してほしい方がより効率的です。これは、リストの場合に特に重要です。(大きなリストの場合は特にそうです。)

これがインライン・リストの働きです。インライン・リストが標準リストと異なるのは、XSLが設計時に仕上がることです。設計の時点で適用できる全てのウェブレットのプロパティは解決され、特別拡張エレメントやファンクションを使ってWAMが実行時の値を必要な場所で使用できるようになります。

もうお分かりかとは思いますが、設計時にXSLを適用するので、実行時のみに使用可能となる情報（フィールド値など）を持つインライン・リストをカスタマイズできません。これは代償です。このような柔軟性が必要な場合は、標準リストを使用してください。ただし、これが必要でない場合は、インライン・リストによる優れたパフォーマンスの恩恵を受けることができます。

2.13.1 インライン・リストの作成

リストをインライン・リストにするには2つの方法があります。

1. WAMの`Inline`プロパティに`Lists`を設定すると、省略値でWAMの中の全てのリストがインライン化されます。

```
FUNCTION OPTIONS(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_WAM) INLINE(Lists)
```

```
END_COM
```

2. `WEB_MAP`の中で、個別のリストをインラインとして指定します。

```
WEBROUTINE NAME(MyWebroutine) DESC('Sample Webroutine')
  WEB_MAP FOR(*OUTPUT) FIELDS( (#DEPTLIST *INLINE) )
```

```
ENDROUTINE
```

リストをインラインに、またはインラインからリストに変更する時にXSLが既に存在する場合は、XSLを再生成するか、リストを削除してXSLに再度追加する必要があります。

インライン・リストがWAMレベルの場合、個別のリストはWEB_MAPで次のようにインラインではないと指定することができます。

```
  WEB_MAP FOR(*OUTPUT) FIELDS( (#DEPTLIST
*NOINLINE) )
```

2.13.2 インライン・リスト内でのウェブレットの使用

インライン・リスト内でウェブレットを使用するには、事前に「インライン認識」がされていなければなりません。インライン・リストにインライン認識されていないウェブレットをドロップしようとする、警告が出ます。



インライン・ウェブレットの外観と動作はインラインでないものと同様の部分で同じです。主な相違点は、リストにドロップした時と、プロパティを変更する度にインライン・ウェブレットが再度生成されることです。このため、知っておかなくてはいけない点はいくつかあります。

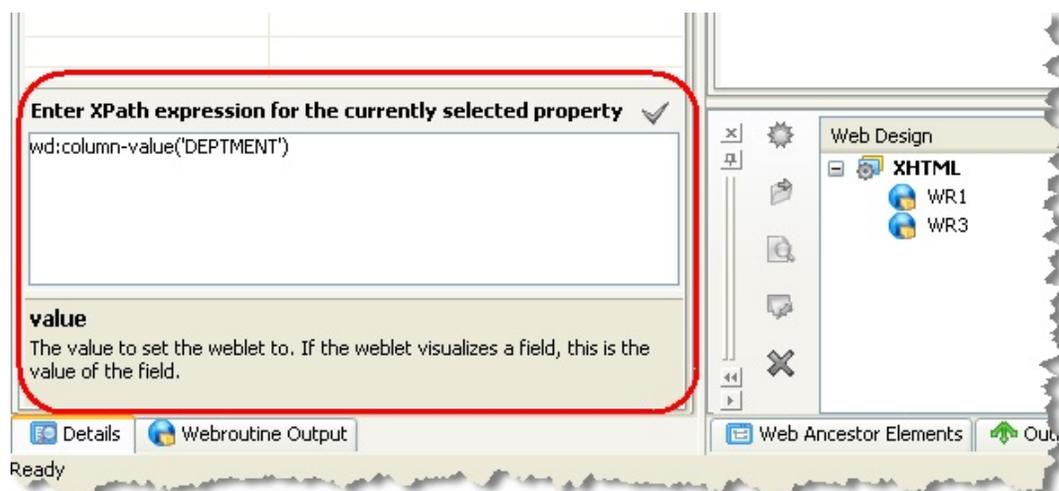
フィールドと列の値へのアクセス

インラインでないリストでは、\$COLUMNNAME XSL変数の列の値またはキー('field-value', 'FIELDNAME')のようなXPath式のフィールド値にアクセスします。インライン・ウェブレットの中のXSLは設計時に実行されるため、このようなXSLは実行時のデータ・アクセスに使用することはできません。ウェブレット・プロパティで実行時のデータにアクセスするのに使用できる特別XSL拡張ファンクションがいくつかあります。次のようなものです。

- wd:column-value('COLUMNNAME') - 指定された列の値を返します。プロパティに#COLUMNNAMEと入力すると、LANSAエディターが自動的にそれをwd:column-valueに変換します。
- wd:field-value('FIELDNAME') - 指定されたフィールドの値を返します。
- wd:variable('VARNAME') - 指定されたMTXTもしくはシステム変数の値を返します。
- wd:row-index() - リストの現在行の行番号を返します。番号は、4桁になるように0で右詰されます。

これらはXSLファンクションであり、XPath式のエリアに入力する必要

があります。



これらの拡張機能はXSL変換が完了した後も結果のドキュメントに残されていなければなりません。(XSL実行後に、WAMランタイムにより解析されます。)このため、XSLプロセッサで処理される時に、自分自身を戻します。例えば、ウェブレットのプロパティの"`{wd:column-value('COLUMNNAME')}`"は、プロパティの変数に"`{wd:column-value('COLUMNNAME')}`"の文字列を返します。この文字列を直接HTML属性に配置することができます。これをHTMLのコンテンツに配置したい場合は、まず`std_util.xsl`で提供されている`wdTagFromAttr.private`テンプレートを使用して、これを特別なタグに変換してください。

コンテキスト・データへのアクセス

フィールドや列の値と同様、通常WebroutineXMLの`<xml:context>`セクションにあるコンテキスト情報は、XSLによって実行時にアクセスすることはできません。ウェブレット・プロパティのコンテキスト・データにアクセスするには、以下のXSL拡張機能を使用してください。

- `wd:web-user()`
- `wd:webapplication()`
- `wd:webapplication-title()`
- `wd:webroutine()`
- `wd:webroutine-title()`
- `wd:service-name()`
- `wd:partition()`

- wd:language()
- wd:images-path()
- wd:action-request()
- wd:layout-name()
- wd:dbcs()
- wd:align-right()
- wd:check-numeric()
- wd:debug()
- wd:trace()
- wd:task()

実行時データを使用できないプロパティ

ウェブレット・プロパティには、ウェブレットがどのように組み立てられるかに影響を与えるものや、CSSまたはJavaScriptの動作を制御するためにブラウザに引き渡されるものがあります。インライン・ウェブレットは設計時に構築されるため、実行時データを使用しての組み立てに影響を与えることはできません。例えば、標準ビジュアルイゼーション・ウェブレットのいずれかのdisplay_modeプロパティを見てください。

「input」の値はHTMLの<input>タグを生成します。「output」の値はを生成します。これらのプロパティで実行時データを使用する必要がある場合は、インラインでないリストを使用します。

この例外は、hide_ifやdisabledのようなブール値プロパティです。

wd:boolean プロパティ

hide_ifやdisabledのようなブール値プロパティの値はウェブレットがどのように組み立てられるかに影響を与えますが、wd:boolean式を含む文字列を使用することで、実行時のデータを受け取ることもできます。

wd:boolean式の構文は、実行時のパフォーマンスを最適化するように設計されているため、簡単に読みとることができないかもしれません。ただし、wd:boolean式を受け取るプロパティが式の編集ダイアログを提供するので、この式を読む必要はありません。プロパティの右にある省略記号のボタンをクリックし、プロパティ・エディターを開いてください。



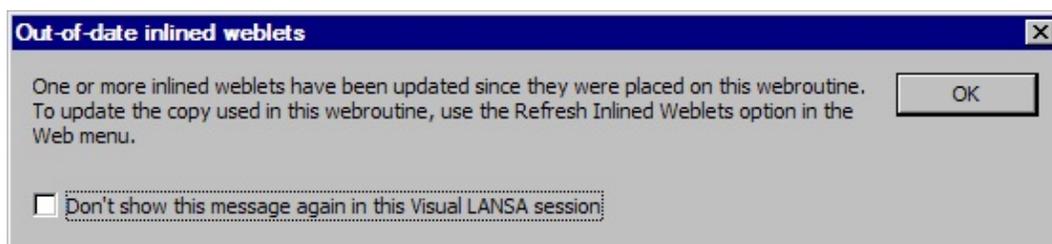
インライン・ウェブレットの更新

Webroutineで使用されるインライン・ウェブレットはウェブレットが変更された時に自動的に更新されません。インライン・ウェブレットを更新するには、リフレッシュをする必要があります。この方法はいくつかあります。

- ウェブレットのプロパティを修正。
- [デザイン]タブでWebroutineを開き、ウェブレットを右クリックしてコンテキスト・メニューから[インライン ウェブレットの更新]を選択。
- [デザイン]タブでWebroutineを開き、[Web] メニューから[インライン ウェブレットの更新]を選択。
- [リポジトリ]もしくは[お気に入り]タブからWAMを選択し、コンテキスト・メニューから[インライン ウェブレットの更新]を選択。多数のWAMやWebroutineの為にインラインのウェブレットを更新する際はこの方法が推奨されます。

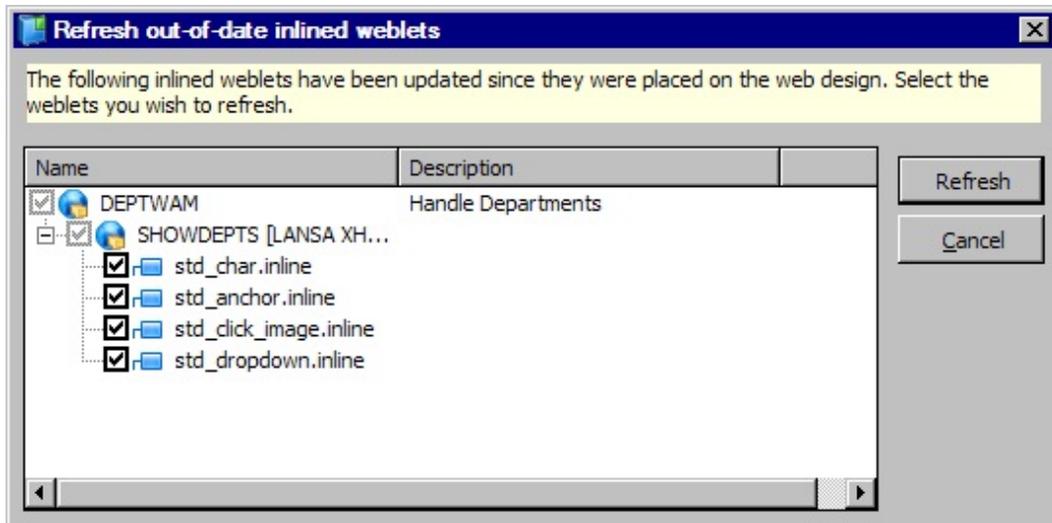
[リポジトリ]もしくは[お気に入り]のタブからWAMを選択すると、各AMとそのWebroutineが開かれ、Webroutineの中にインラインウェブレットがないか、また古すぎないかを確認することができます。選択されたAMの数や複雑性、インライン・ウェブレットの数にもよりますが、このプロセスには時間がかかる場合があります。

Webroutineを開くと、更新を必要とするインライン・ウェブレットがあるかどうかはLANSAエディターにより示されます。



このメッセージは開かれているWebroutineにのみ適用されることに注意してください。WAMに複数のWebroutineがある場合は、各Webroutineを開いて、ウェブレットのバージョンの違いを確認してください。

[インライン ウェブレットの更新]を選択した時に、どのウェブレットを更新するか選択することができます。



省略値では全てのウェブレットが事前に選択されます。それら全てを更新することが推奨されています。

3. 主要なトピック

まずは『WAMの概論』を読み、その後次のトピックを熟読してください。

3.1 WAMでのCHECKNUMERICの使用

3.2 WAMアプリケーションのデザイン

3.3 複数言語に対応した開発

3.4 WAMアプリケーションでのCookieの使用

3.5 サービス名の使用

3.6 セッション状態の使用

3.7 オブジェクトの削除

3.8 LOBデータ・タイプとストリーム・ファイル

3.9 WAM外部リソース

3.10 jQueryの使用

3.11 WAMのテーマ

3.12 ローカライズ

3.13 JSONサポート

3.14 WAM出力のファイル保存

3.15 ドキュメントの型宣言(DOCTYPE)

3.1 WAMでのCHECKNUMERICの使用

WAMにCHECKNUMERICを使用して、WAMで数値妥当性検査ができるようにします。これはBEGIN_COMコマンドで指定します。

```
FUNCTION OPTIONS(*DIRECT)  
  BEGIN_COM ROLE(*EXTENDS #PRIM_WAM) CheckNumeric(Yes)
```

WAMのRDMLXコードにCHECKNUMERICが指定されていない場合、省略値はシステム情報の[Webで数値の妥当性を確認する]の値が基本となります。この[Webで数値の妥当性を確認する]を探すには、[リポジトリ]で[システム情報]を開き、[コンパイルと編集オプション]を展開します。[プロセスとファンクションのコンパイル省略値]の下にこの設定があります。

CHECKNUMERICが適用された場合、実行時に無効な数字が入力された場合は、以下のような警告が表示されます。



3.2 WAMアプリケーションのデザイン

- LANSA Webアプリケーションの設計時に、アプリケーションに必要なWAM数や各WAMに組み込むWebroutine数を決める必要があります。通常は、アプリケーションを機能ごとに分割し、その各機能に1つのWAMを割り当てます。
- WAMとWebroutineの構造を設計する際は、WAMのセッション管理も理解しておく必要があります(詳しくは『[WAMのセッション管理](#)』を参照してください)。省略値では、各WAMは他のWAMから独立した形で独自のセッション状態を持ちますが、複数のWAMセッションを繋げることもできます。
- 複数のWAMアプリケーションをモジュール化するには、別のRDMLXコンポーネントを使用します。Webroutine内でRDMLXコンポーネントを使用して、ビジネス・ルール処理をアプリケーションに割り当てていきます。このデザイン方法により、GUIベースやブラウザ・ベースのアプリケーションなど、異なるプレゼンテーション・テクノロジーでプログラムを再利用できるようになります。
- Webroutineサービス名を使用すると、WAMアプリケーション配布時の柔軟性が高まります。例えば、アプリケーションを別の区画、WAM、またはWebroutineに再配布する際に、サービスへの外部URL参照を修正する必要がなくなります。

3.3 複数言語に対応した開発

WAMでは、1つのRDMLXやWAMのプログラムをベースに複数の言語を実行できるWebアプリケーション開発をサポートしています。

WebroutineおよびウェブレットのXSLスタイルシートは、言語に依存しています。

言語ごとに別の特定のスタイルシートを保存・配布することができます。例えば、英語と中国語のプレゼンテーション・ページとウェブレットを別々に作成するといったことが可能です。

複数言語のWAM開発や保守作業の方法はアプリケーションでコントロールします。

全ての言語で同じ体裁(メニュー、各Webroutineのページ・レイアウトが同じなど)のWebアプリケーションにしたい場合、最も簡単な複数言語のアプリケーションの導入方法は、ウェブレットのプロパティでキャプションのテキスト値に言語変数(*MTXT)を設定することです。こうすることで、各Webroutineとレイアウトに対し1セットのXSLソースが存在することになり、アプリケーションの管理と配布が単純化されます。

言語ごとに各Webroutineで異なるインターフェイスが必要なWebアプリケーションの場合、必要に応じてWebroutineとレイアウトXSLを各言語用に作成しなければなりません。ただし、この方法は管理するオブジェクトがより多くなるので、理想的な方法ではありません。

実行時に言語固有のページが存在しない場合、区画の省略値ページがプレゼンテーション出力として使用されます。つまり、言語ごとに別のページを用意することも可能ですが、1つのページで複数の言語に対応することもできます。

フィールドのキャプションやリストの見出しは言語に依存しており、対象となる言語で表示されます。言語変数がサポートされており、Webroutine XSLでその変数を参照できるようになっています。ウェブレットのキャプション定義では、言語特有のリテラルのテキストを定義するのではなく、言語変数を定義することができます。こうすると、ウェブレットはWAMが実行される言語環境に基づいたテキストを表示します。

重要：

- 1.WAMもしくはウェブレットの保存時、Webroutineで参照される複数言語対応テキスト変数の値が解決されます。

2.WAMまたはウェブレットを配布する際、ソースのシステムからの値は配布パッケージに取り込まれます。

実行時には、URLのlangパラメータに指定されている言語キーワードの受け渡しによって言語が要求されます。例えば、以下のURLは、英語のページを要求しています。

```
http://localhost/cgi-bin/lansaweb?  
srve=LEWAM01_SearchQuery+part=DEM+lang=ENG
```

詳細については、『複数言語アプリケーション設計ガイド』を参照してください。

3.4 WAMアプリケーションでのCookieの使用

Webroutineの入力用XMLファイルに特別なcookiesセクションを追加することによって、ブラウザでCookieを設定する方法を制御できます。例えば次のようになります。

```
<lxml:server-instructions>
  <lxml:cookies>
    <lxml:cookie name="USRID">
      <lxml:value field-name="EMPNO"></lxml:value>
      <lxml:expires field-name="EXPDAT"></lxml:expires>
      <lxml:domain field-name="DOMAIN"></lxml:domain>
      <lxml:path field-name="PATH"></lxml:path>
      <lxml:secure field-name="SECFLAG"></lxml:secure>
      <lxml:httponly field-name="HTTPFLAG"></lxml:httponly>
    </lxml:cookie>
  </lxml:cookies>
</lxml:server-instructions>
```

この例では、Webroutineのフィールド値に基づいてブラウザでCookieを設定する場合に必要なcookiesセクションの形式が示されています。lxml:cookieエレメントに、必要なCookie情報をすべて記述します。name属性には、Cookieの格納に使用する名前と、後続のWebroutine要求でCookie値を含むLANSAフィールド名前を指定します。

注：cookiesセクションは、この例のように、server-instructionsセクションの中に入れる必要があります。

以下の表は、各エレメントをまとめています。

エレメント	説明
-------	----

lxml:value	Cookieに格納する値
------------	--------------

lxml:expires	Cookieの有効期限切れの日付(GMT形式)。この値を指定しない場合は、ブラウザが閉じた時点でCookieの有効期限が切れます。
--------------	---

lxml:domain	ドメインの名前。Cookieは、このドメインのすべてのサブドメインで使用可能になります。この値を指定しない場合は、Cookieを設定したドメインのページでのみCookieが使用可能になります。
-------------	--

- lxml:path Cookieのパス。この値は、別のディレクトリからのページに設定することも可能です。1つのサブディレクトリで設定したCookieを別のサブディレクトリでも使用できるようにするには、両方のサブディレクトリの親ディレクトリのパスを指定します。"/"を指定すると、Cookieは、そのドメインのすべてのサブディレクトリで使用可能になります。値を指定しない場合は、Cookieが設定されているパスでのみCookieが使用可能になります。
- lxml:secure trueを指定すると、Cookieは、セキュアなSSLドメインのページでのみ使用可能になります。
- lxml:httponly ほとんどの最近のブラウザでサポートされます。値 "true" はCookieが(JavaScriptなどの)非HTTP API経由ではアクセスできないことを示します。

エレメントには最初の例にもあるように、field-name属性を使って、Cookie値を格納するフィールドの名前を指定することもできます。Cookie値をXMLの中に直接指定する例を以下に示します。

```
<lxml:cookies>
  <lxml:cookie name="LSTACT">
    <lxml:value>Inquiry</lxml:value>
    <lxml:expires>Thu, 31 Dec 2020 10:00:00 GMT</lxml:expires>
    <lxml:domain>acme.com</lxml:domain>
    <lxml:path>/home</lxml:path>
    <lxml:secure>true</lxml:secure>
    <lxml:httponly>true</lxml:httponly>
  </lxml:cookie>
</lxml:cookies>
```

3.5 サービス名の使用

同じWebroutineをウェブページとして呼び出す際に、異なるURL(以下参照)を使うことが可能です。

それでは、Webroutine例MaintainRecordの定義を見てみましょう。

* This is the entry point to the WAM.

*

```
WEBROUTINE NAME(MaintainRecord) SERVICENAME(MaintEmployee) ONENTRY(*SESSIONSTATUS_NONE)
```

Webroutineの定義に基づき、このWebroutineは、Webroutine定義オプションのServiceNameプロパティを使用して、次のようにWAMを呼び出すことができます。

http://<サーバー>/cgi-bin/lansaweb?srve=MaintEmployee&part=DEX

区画内で一意であれば良いので、ServiceNameはこれ以上の条件をつける必要がありません。

2番目のURLオプションは、WAM名とWebroutine名をの組み合わせを一意のセットとして、以下のように使用します。

**http://<サーバー>/cgi-bin/lansaweb?
wam=WAMEX02&webtrn=MaintainRecord &part=DEX**

3.6 セッション状態の使用

セッション管理の導入を簡単に見てみましょう。

まず最初にWAMのRDMLXにBEGIN_COMコマンドを設定します。ここで、特定のWebroutineに関連付けられたONENTRYパラメータで指定されている場合以外は、このWAMの各Webroutineが実行前のセッション状態がにアクティブでなければならないようにします。

```
* Session status must be active to execute the Webroutines in this WAM unless
* otherwise indicated by the OnEntry parameter of the specific WebRoutine
BEGIN_COM ROLE(*EXTENDS #PRIM_WAM) SESSIONSTATUS(Active)
```

エントリー・ポイントのWebroutineは(この例ではMaintainGrid)、ONENTRY(*SESSIONSTATUS_NONE)と指定されているため、セッション状態なしで実行することができます。ですが、WAM内のその他の全てのWebroutine(上記のパラメータ設定がないものは省略値でアクティブなセッション状態が必要となります)。

このエントリーWebroutineはセッション状態をアクティブに設定し、アクティブなセッション状態を必要とするWAM内の他のWebroutineを呼び出すことができますようにします。

```
* This is the entry point to the WAM.
WEBROUTINE NAME(MaintainGrid) ONENTRY(*SESSIONSTATUS_NONE)
    * set the session to active.
    #com_owner.SessionStatus := Active
    * transfer to the routine that will display the the web page.
    TRANSFER TOROUTINE(ShowGrid)
ENDROUTINE
```

では、セッション状態が無効の場合(例えば、ONENTRY(*SESSIONSTATUS_NONE)設定のないWAM内のWebroutineを呼び出そうとした場合など)は、どうなるでしょうか？

セッション状態が無効の場合、SessionInvalidイベントが起動されます。そして、ここでユーザーに対して適切な情報を発行するよう制御します。

```
* Handle invalid session status
EVTROUTINE HANDLING(#com_owner.SessionInvalid)
    TRANSFER TOROUTINE(#WAMEX99.SessionIsInvalid)
ENDROUTINE
```

3.7 オブジェクトの削除

Webコンポーネントとグラフィック変数

- Visual Webコンポーネントを定義したフィールドを削除すると、そのWebコンポーネントも削除されます。Webコンポーネントだけを削除したい時は、Webファンクション・エディターを使用してください。
- システム変数のいずれかを削除すると、LANSA for the Webのグラフィック変数も削除されます。
- ただし、LANSA for the Webのグラフィック変数を削除しても、システム変数は削除されません。

プロセスとファンクション

- Web対応プロセスのLANSAファンクションを削除すると、そのファンクションのWeb詳細情報も削除されます。削除される情報の中には、このファンクション用に生成された全てHTMLも含まれます。
- Web対応LANSAプロセスを削除すると、それに関連する全てのファンクションと各ファンクションのWeb詳細情報が削除されます。そのプロセスのために構成されているカスタマイズされた省略値ページも削除されます。
- ファンクションの名前を変更すると、それに関連する全てのWeb詳細情報の名前も変更されます。

WAM

- LANSAエディターからWAMを削除すると、そのWAMのRDMLX、および全言語、全テクノロジー・サービス用のWebRoutine XSLスタイルシートも全て削除されます。WAMの削除時は、オプションでWAMの省略値のレイアウト・ウェブレットも削除できます。WAMに使用されるその他のウェブレットは削除されません。
- オーフアンWebroutineのデザインを削除するオプションが有効（省略値）になっている場合、これを手動で削除する必要はありません。これが有効に設定されていない場合、XSLスタイルシートは自動的に削除されず、LANSAエディターの[アウトライン]ビューと[Webデザイン]ビューに表示されます。これらは、WAMエディターの[アウトライン]ビューで削除できます。
- WAMをチェックインすると、そのRDMLXのソース、および全言語、全テクノロジー・サービスのWebroutineのXSLスタイルシートが全てチェックインされます。また、このWAMのレイアウト・ウェブ

レットもチェックインされます。

- WAMをチェックアウトすると、そのRDMLXのソース、および全言語、全テクノロジー・サービスのWebroutineのXSLスタイルシートが全てチェックアウトされます。WAMのレイアウト・ウェブレットはチェック・アウトされないことに注意してください。
- 新しい区画言語を作成する時は、WAMのWebroutineを再発行する必要があります。自動的には行われません。

ウェブレット

- ウェブレットのチェックインやチェックアウトは、LANSAエディターの[リポジトリ]ビューで行います。
- WAMエディターの[ウェブレット]ビューでウェブレットを削除すると、全言語、全テクノロジー・サービスのウェブレットが削除されます。
- 削除するウェブレットが他のウェブレットまたはWebRoutine XSLスタイルシートによって使用されている場合、LANSAエディターでこのドキュメントを開いた時や、このドキュメントを使用するWAMを実行した時に、エラーが発生します。WebroutineのXSLソースを編集して、削除したウェブレットへの参照を全て削除する必要があります。
- 新しい区画言語を作成するときには、ウェブレットを再発行する必要があります。自動的には行われません。

3.8 LOBデータ・タイプとストリーム・ファイル

WAMでは次のようなストリーム・ファイルを扱います。

- Webサーバーに保管したくないもの。
- コンテンツがアプリケーションのデータベースに格納されているドキュメント。
- オンデマンドで作成されたドキュメント。

LOBデータ・タイプのコンテンツ(BLOBとCLOB)や、アプリケーション・サーバー内にWebroutineと共に存在するストリーム・ファイルを扱うことができます。

WebroutineコマンドのパラメータResponseを次のように定義して、Webroutineで使用するこの種のファイルを作成します。

```
Webroutine Name(SEND_SAMPLE) Desc('Sample Document') Response(#HTTPR)
* MYPATH is the directory where the sample documents are stored
#HTTPR.ContentFile := #MYPATH + 'sample'
Endroutine
```

このWebroutineにRDMLXコードを挿入することで、ファイルのコンテンツを作成したり、送信ファイルを決定したりすることができます。取り扱うファイル名にContentFileを設定するだけです。

また文字列のコンテンツを送信することもできます。この場合、コンテンツタイプを設定する必要があります。charsetは現在のエンコードが設定されます。エンコードが異なる場合は、charsetプロパティを設定してください。

```
Webroutine Name(SEND_SAMPLE2) Desc('Sample ContentString') Response(#HTTPR2)
Define Field(#VAR) Type(*String)
#VAR := '<html><body><h1>My Page</h1></body></html>'
#HTTPR2.ContentString := #VAR
#HTTPR2.ContentType := 'text/html'
Endroutine
```

以下も参照してください。

3.8.1 ファイル要求

3.8.2 LOB/Fileコンテンツ・タイプ

3.8.3 LOB/File プロパティ

3.8.4 カスタムのHTTPヘッダー

3.8.5 コンテンツがテキストのCLOBおよびファイル

3.8.6 圧縮

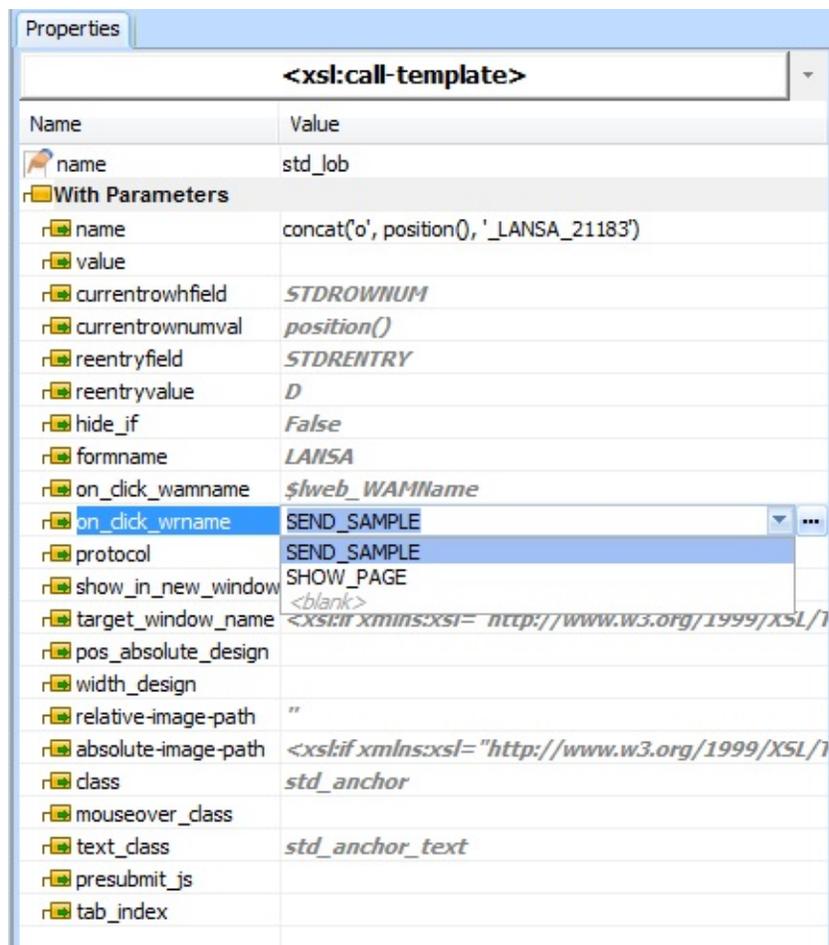
3.8.1 ファイル要求

Webroutineを呼び出してLOBやファイルを要求します。例えば次のようになります。

```
http://myhost/cgi-bin/lansaweb?  
wam=LOBSAMPLE&webtrtn=SEND_SAMPLE&ml=LANSAXHTML  
&part=DEX&lang=ENG
```

LOBの標準ビジュアライゼーションのウェブレット(std_job)を使って、これをアンカーとして表示できます。ただし、この要求を処理するWebroutineを指定する必要があります。これは、LOBフィールドにドロップする必要はありません。LOBウェブレットは空いたスペースにドロップできます。

例えば次のようになります。



3.8.2 LOB/Fileコンテンツ・タイプ

LANSAsは、拡張子に基づいてLOBやファイルのコンテンツ・タイプを決定します。ですから、通常はcontent-typeヘッダーを追加する必要はありません。接頭辞が決まっていない場合コンテンツ・タイプの省略値は"application/octet-stream"になります。コンテンツ・タイプを上書きするには、ヘッダーcontent-typeを追加します。これについては、このセクションの終わりで説明されています。

3.8.3 LOB/File プロパティ

Response変数のプロパティには次のようなものがあります。

ContentType:ファイル拡張子により決定された省略値のコンテンツ・タイプを上書きする、もしくはファイル拡張子が不明な場合に使用します。省略値では、LANSAはファイル拡張子を使ってコンテンツ・タイプを決定します。

Charset:テキスト・コンテンツを持つファイルの文字セットを上書きします。省略値では、LANSAはファイルのCCSID(IBM iの場合)、またはバイト・オーダー・マーク(その他のプラットフォームの場合)により決定されます。

AttachmentFileName:提案された名前でファイルを添付として保存するかどうかをユーザーに確認します。定義されない場合は、ヘッダーcontent-dispositionは追加されません。

Compression:TrueまたはFalse。エンコードgzipを使用して、ファイルを圧縮するかどうかを決定します。省略値はFalseです。

RemoveFile:TrueまたはFalse。Trueの場合、ユーザー・エージェントに送信された後、ファイルは削除されます。省略値はFalseです。

注：LANSAファイルから読み込んだLOBファイルは削除する必要はありません。LANSAランタイムは、要求が完了した時点でこれらのファイルを自動的に消去します。

例えば次のようになります。

```
* Explicit content type
#HTTPR.ContentType := 'application/pdf'

* Save file as an attachment. Suggest to save as 'mysample.pdf'
#HTTPR.AttachmentFilename := 'mysample.pdf'

* Compress file using gzip encoding
#HTTPR.Compression := True

* Remove file after it is sent
#HTTPR.RemoveFile := True
```

3.8.4 カスタムのHTTPヘッダー

その他のHTTPヘッダーを追加する場合、AddHeaderメソッドを使用します。

```
* Custom HTTP Header example: Don't cache
#HTTPR.AddHeader('Pragma', 'no-cache')
```

3.8.5 コンテンツがテキストのCLOBおよびファイル

content-typeがtextのCLOBやファイルはテキストとして送信され、このファイルのエンコードがLANSAのファイル処理方法に影響を与えません。

IBM iの場合、LANSAはファイルのCCSID属性を使ってコンテンツの文字セットを決定します。この文字セットは、Charsetプロパティを指定することで上書きできます。

UTF-16の特殊なケース

IBM i HTTPサーバーはUTF16のテキスト・コンテンツは取り扱いません。UTF16はUTF8にトランスコードされます。ただし、XMLドキュメントは例外です。エンコードがUTF16のXMLドキュメントは、content-typeアプリケーション/xml (バイナリ)と共に送信されます。

WindowsとLinuxの場合、自動的に検知されるエンコードはUTF16とUTF8だけです。LANSAはファイルのバイト・オーダー・マーク(BOM)を使用して、エンコードを決定します。その他のテキスト・ファイルの文字セットを設定するには、HTTP応答のプロパティCharsetを指定します。

3.8.6 圧縮

ファイルを圧縮するには、プロパティCompressionを設定するか、HTTPヘッダーにcontent-encoding: gzipを追加します。

圧縮はCPUが集中的に使用され、実行はファイル送信の度に行われることに注意してください。(つまり、圧縮ファイルはキャッシュされません。)CPUの使用率と圧縮によるサイズ縮小とのバランスをよく考えてください。

3.9 WAM外部リソース

外部で作成された、アプリケーションに関係するリソース全てをリポジトリに格納できるよう、RDMLXのリポジトリ・オブジェクト外部リソースが提供されています。

これには次のようなものがあります。

- イメージ
- カスケード・スタイル・シート(CSS)
- JavaScriptファイル

Web開発用として、LANSAは次のような外部リソースを提供しています。

- テーマレット
- jQueryおよびjQuery UI ライブラリ
- CSSとJavascriptファイルは新しいjQueryウェブレットに関連付けられています。

クロスリファレンス

WAMやウェブレットで使用される外部リソースはクロスリファレンスに表示されます。これにより、配布パッケージの作成時に従属オブジェクトの選択が簡単になります。

詳しくは以下を参照してください。

[3.9.1 スクリプトとスタイルの指定](#)

[3.9.2 外部リソースの追加順序](#)

[3.9.3 提供されるWAM外部リソース](#)

3.9.1 スクリプトとスタイルの指定

WAMやウェブレットでは、必要なJavaScriptやCSSスタイルシートを指定することができます。このJavaScriptやCSSスタイルシートはまずVisual LANSAに登録する必要があります。

詳しくは、『*Visual LANSA*ユーザーガイド』の「[個別の外部リソースの登録](#)」または、『*Visual LANSA*開発者ガイド』の「[複数外部リソースの登録](#)」および「[WAMでの外部リソースの使用](#)」を参照して、ウェブレットへのリンク方法を確認してください。

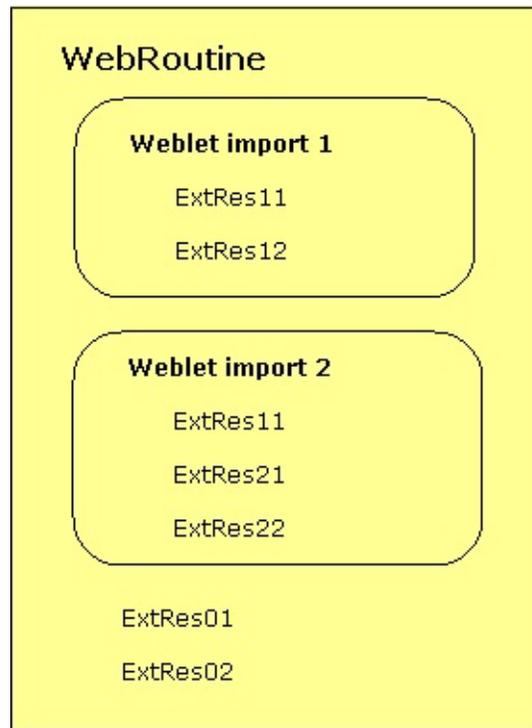
スクリプトやスタイルの外部リソースはWebroutineとウェブレットの両方に追加できます。標準的な方法は、テーマ(詳しくは、「[WAMのテーマ](#)」を参照)を提供するCSSスタイルシートはサイト・レイアウトとスクリプトに追加し、その他のスタイルは必要に応じてWebroutineやウェブレットに加えます。

WAMは実行時スクリプトとスタイルを統合させ、出力ページに1度だけ追加されるようにします。詳細については、[3.9.2 外部リソースの追加順序](#)を参照してください。

[↑3.9 WAM外部リソース](#)

3.9.2 外部リソースの追加順序

スクリプトやスタイルが追加される際の順序は重要です。外部リソースは見つかった順に追加されていきます。WebRoutineやウェブレットが独自の外部リソースを持つウェブレットをインポートする場合、この外部リソースはウェブレットのインポート時に追加されます。以下の例は追加順序を示しています。



外部リソースの追加順序は次のとおりです。

1. ExtRes11
2. ExtRes12
3. ExtRes21 - 外部リソースExtRes11のインポートは繰り返されないことに注意してください。
4. ExtRes22
5. ExtRes01
6. ExtRes02

この順番にすることで、ファイルに従属関係がある(例えば既存のJavaScriptライブラリに従属するJavaScriptファンクションなど)場合にスクリプトやスタイルを正しい順序で追加できるようになります。外部リ

ソースの順序に関する詳細は、『*Visual LANSA ユーザー ガイド*』の「[外部リソースの使用](#)」を参照してください。

[↑3.9 WAM外部リソース](#)

3.9.3 提供されるWAM外部リソース

LANSAからは以下の外部リソースが提供されます。これは標準ウェブレットでも使用されています。

名前	タイプ	説明
XMC001	スタイル	MobiscrollのCSS
XMC01L	スタイル	jQuery Mobile コアLANSAスタイル
XMCJQM	スタイル	jQuery Mobile コア構造CSS
XMJ001	スクリプト	MobiscrollのJavaScript
XMJ01L	スクリプト	jQMobileコアLANSAライブラリ
XMJJQM	スクリプト	jQueryMobile コアJavaScriptライブラリ
XMJQC	スクリプト	jQueryのjQMobile TSP用のコアJavaScript
XMT01J	スタイル	jQuery Mobile の省略値テーマ
XWC001	スタイル	jQuery拡張ウィジェットのCSS
XWC002	スタイル	jQueryタイムピッカー・プラグインのCSS
XWJ001	スクリプト	jQuery拡張ウィジェット
XWJ002	スクリプト	jQueryタイムピッカー・プラグイン
XWJ003	スクリプト	LANSA JSON JavaScript ライブラリ
XWJ004	スクリプト	CKEditor JavaScript ライブラリ
XWJ004E	スクリプト	拡張CKEditor

XWJ004J	スクリプト	jQuery用CKEditor プラグイン
XWJ005	スクリプト	Googleグラフ
XWJDCJS1	スクリプト	Douglascrockford/json-js
XWJMODZR	スクリプト	Modernizr JavaScript ライブラリ
XWJQC	スクリプト	jQueryのコアJavaScriptライブラリ
XWJQUI	スクリプト	jQuery UIのJavaScriptライブラリ
XWT01J	スタイル	Redmond – jQuery UI ウィジェット
XWT01L	スタイル	Redmond – LANSA 拡張テーマ
XWT01L101	スタイル	Redmond – LANSA スタイル# 1テーマレット
XWT01L102	スタイル	Redmond – LANSA スタイル# 2テーマレット
XWT02J	スタイル	Pepper Grinder – jQuery UI ウィジェット
XWT02L	スタイル	Pepper Grinder – LANSA 拡張テーマ
XWT02L101	スタイル	Pepper Grinder – LANSA スタイル# 1テーマレット
XWT02L102	スタイル	Pepper Grinder – LANSA スタイル# 2テーマレット
XWT03J	スタイル	Cupertino – jQuery UI ウィジェット
XWT03L	スタイル	Cupertino – LANSA 拡張テーマ
XWT03L101	スタイル	Cupertino – LANSA スタイル# 1テーマレット
XWT03L102	スタイル	Cupertino – LANSA スタイル# 2テーマレット
XWT04J	スタイル	Smoothness – jQuery UI ウィジェット
XWT04L	スタイル	Smoothness – LANSA 拡張テーマ

XWT04L101	スタイル	Smoothness – LANSA	スタイル# 1テーマレット
XWT04L102	スタイル	Smoothness – LANSA	スタイル# 2テーマレット
XWT05J	スタイル	UI Darkness – jQuery UI	ウィジェット
XWT05L	スタイル	UI Darkness – LANSA	拡張テーマ
XWT05L101	スタイル	UI Darkness – LANSA	スタイル# 1テーマレット
XWT05L102	スタイル	UI Darkness – LANSA	スタイル# 2テーマレット
XWT06J	スタイル	UI Lightness – jQuery UI	ウィジェット
XWT06L	スタイル	UI Lightness – LANSA	拡張テーマ
XWT06L101	スタイル	UI Lightness – LANSA	スタイル# 1テーマレット
XWT06L102	スタイル	UI Lightness – LANSA	スタイル# 2テーマレット
XWT07J	スタイル	Blitzer – jQuery UI	ウィジェット
XWT07L	スタイル	Blitzer – LANSA	拡張テーマ
XWT07L101	スタイル	Blitzer – LANSA	スタイル# 1テーマレット
XWT07L102	スタイル	Blitzer – LANSA	スタイル# 2テーマレット
XWT08J	スタイル	South Street – jQuery UI	ウィジェット
XWT08L	スタイル	South Street – LANSA	拡張テーマ
XWT08L101	スタイル	South Street – LANSA	スタイル# 1テーマレット
XWT08L102	スタイル	South Street – LANSA	スタイル# 2テーマレット

↑3.9 WAM外部リソース

3.10 jQueryの使用

LANSAが提供するライブラリ

LANSAからは、jQueryコア、jQuery UI、およびjQuery Mobile のライブラリが提供されます。JavaScriptとCSSファイルは外部リソースとして提供されます。(このリストは「[3.9.3 提供されるWAM外部リソース](#)」を参照)

独自のウェブレットでjQueryを使用

提供されるライブラリは、LANSAより提供される標準ウェブレットで使用されます。また、独自のウェブレットやJavaScriptで使用することも可能です。そのためには、(各自の状況に合わせて)自身のウェブレット、レイアウト、またはWebroutineに外部リソースを追加する必要があります。

jQuery、jQuery UIおよびjQuery Mobile のライブラリは、LANSAの将来のバージョンで更新される可能性があります。

以下も参照してください。

[3.10.1 ツールとヒント](#)

3.10.1 ツールとヒント

その他のJavaScriptライブラリ作業

jQuery対応のXHTMLレイアウトには、プロパティjQueryNoConflicがあります。これにtrueが設定されると、jQuery.noConflict()が呼び出され、\$の名前がなくなります。jQueryのコアを入れる前に、\$名を使用する他のJavaScriptライブラリを入れておく必要があります。

jQuery IDセレクター内でLANSAフィールド、カラム名を避ける方法

LANSAのフィールド名には名前に'\$'が付いている場合があります。リスト・カラム名の完全修飾名には、'!'の区切り文字が付いています。(LIST1.0001.EMPNOなど)jQuery IDセレクターで使用できるフィールドIDを取得するには、LANSA jQueryグローバル拡張子lansa.makeSafeId()が使用できます。

```
var myvar = jQuery(jQuery.lansa.makeSafeId("LIST1.0001.EMPNO"));
```

上記の例は、以下と同じです。

```
var myvar = jQuery("#LIST1\\.0001\\.EMPNO");
```

3.11 WAMのテーマ

jQuery UIのビジュアル・デザイン・テーマが多くのウェブレットで使用されており、これによりWebデザインの色、フォントなどのビジュアル・デザイン要素のカスタマイズが簡単にできます。テーマのあるレイアウト(テーマレット)により、複数のWAMに一貫性のあるレイアウトが提供されます。

LANSAからは次のjQuery UIテーマが提供されています。

- Redmond
- Pepper Grinder
- Cupertino
- Smoothness
- UI Darkness
- UI Lightness
- Blitzer
- South Street

提供されるこれらの標準テーマレット(2つの異なるスタイル)とは別に、「[Web アプリケーション レイアウト マネージャ ウィザード](#)」を使って、独自のテーマを持ったレイアウトを作成することもできます。

テーマを導入するには、jQuery UIスタイルシートと、LANSA拡張テーマのベース(例えばテーマRedmondの場合は外部リソースXWT01L)、もしくは該当するLANSAテーマレット・スタイル(例えばテーマRedmondの場合は外部リソーススタイルXWT01JとXWT01L101)を自身のWAMレイアウトに追加します。

もしくは、上記を独自のサイト・レイアウトに追加して、全てのWAMで同じテーマを共有することもできます。

3.12 ローカライズ

ウェブレットには言語依存の設定があるものもあります。ウェブレットはISO言語もしくはISO言語-国コードを使用して、テキストをローカライズしたり、日時の形式など地域特有の設定を使用します。ローカライズ可能なプロパティに省略値'auto'が設定されているウェブレットで、上記の設定ができるようになります。WAMをローカライズする一番簡単な方法はこの省略値設定を使用することです。

ローカライズの詳細は、『[LANSA 管理者ガイド](#)』の「[ISO 言語コード](#)」を参照してください。

WAMは区画の言語定義で指定されたISO言語コードを使用します。ユーザー・エージェントの言語コードではありません。

サードパーティのソフトウェア・ベースのウェブレットでは、サポートされる言語が異なります。

ほとんどのローカライズ可能な文字列はイメージ・ディレクトリのサブディレクトリscript/i18nにあるJSONファイルに格納されます。

以下も参照してください。

[3.12.1 テクノロジ・サービス・プロバイダ](#)

[3.12.2 サードパーティ・ライブラリ](#)

3.12.1 テクノロジ・サービス・プロバイダ

ディレクトリⁱ¹⁸ⁿ内の各テクノロジ・サービスには、ローカライズ可能な文字列を含む独自のディレクトがあります。

各言語の文字列は以下のような名前でファイルに格納されます。

`std_messages-<lang-code>.json`

`<lang-code>`の部分は、ISO言語コードの2文字の後にオプションの国コードが続きます。例えば、`"de-AT"`は、オーストリア("AT")で使われるドイツ語("de")を意味します。

LANSAフレームワークにより使用される文字列は、全て`std_messages-en.json`で定義され、常にこのファイルがフレームワークにより最初にロードされます。こうすることで、翻訳が見つからなかった場合に使用できる文字列を最低限確保することができます。次にフレームワークは現在の言語の言語ファイル、その後に国特有のファイルをロードしてマージします。例えば、区画の言語コードが`"de-AT"`の場合、フレームワークは次の順序でファイルをロードします。

- `std_messages-en.json`
- `std_messages-de.json`
- `std_messages-de-AT.json`

つまり、`std_messages-de-AT`ファイルを定義する際、各文字列を複製して管理する必要がないということです。`std_messages-de.json`に存在しているバージョンと異なる文字列のみを定義するだけです。

3.12.2 サードパーティ・ライブラリ

サードパーティ・ライブラリはローカライズ可能な文字列処理のメカニズムが異なる場合があります。各ライブラリ特有の処理についての詳細はドキュメントで確認してください。可能な限り、サードパーティのローカライズ・ファイル(jQueryなど)は、scripts/i18n に格納されます。ただし、ライブラリの特殊な事情によりこれができない場合、ローカライズ・ファイルはライブラリのホーム・ディレクトリに格納されます。(jQuery UIやCKEditorなど)

3.13 JSONサポート

WAM でのJSON の使用

WAMでは次の2通りの方法でJSON(Javascriptオブジェクト表記)がサポートされています。

- **JSON データの送信**

JSON を使用してフィールドやリストを送信する方法については、「[3.13.2 JSONの便利なラッパー](#)」を参照してください。

- **JSON 応答 Webroutine**

WebroutineのWebマップ(フィールドおよびリスト)は、UTF8でエンコードされ、MIMEタイプapplication/jsonでJSON応答として送信されます。次のようなWebroutineを使ってAjax要求に対する応答を送信することができます。

```
Def_List Name(#list01) Fields(#deptment #deptdesc) Type(*working)

* Basic JSON response
Webroutine Name(SAMPLE1) Response(*JSON)
  Web_Map For(*output) Fields(#empno #givenname #surname #list01)

  * Prepare response
  * ...
Endroutine

* JSON response with captions
Webroutine Name(SAMPLE2) Response(*JSON) Options(*METADATA)
  Web_Map For(*output) Fields(#empno #givenname #surname #list01)

  * Prepare response
  * ...
Endroutine
```

LANSAにより提供されているウェブレットには、JSON応答のWebroutineでAjax要求を使ったデータ更新を行なうものがあります。

JSON応答のWebroutineを簡単に使用できるように、JavaScriptファイルstd_json.jsに便利なファンクションが提供されています。このJavaScriptファイルをウェブレットやWebroutineに入れるには、提供されている外部リソースXWJJQ (jQueryコア)およびXWJ003 (LANSA JSONライブラリ)を追加します。

このファイルでは、JSON応答Webroutineのフィールド、リストやコンテキスト情報にアクセスできるよう、JSONの便利なラッパー・オブジェ

クトが定義されます。詳細については、「[3.13.2 JSONの便利なラッパー](#)」を参照してください。

以下も参照してください。

[3.13.1 JSON リスト](#)

3.13.1 JSON リスト

通常のページを作り出すWebroutine内で、WEB_MAPの属性*JSONを使用すると、JSON形式のリストが出力できます。

```
Webroutine Name(SAMPLE) Desc('Sample JSON List')
  def_list name(#list1) type(*working) fields(#deptment #deptdesc)
  web_map for(*output) fields((#list1 *JSON))
  * Populate list
  * ...
Endroutine
```

WEB_MAPをリストするJSONリストは出力ページには表示されません。JavaScriptオブジェクトLstd.Json.Listを使って、JavaScript経由でこのリストを使用します。

3.13.2 JSONの便利なラッパー

LANSAからはJavaScriptライブラリが提供されており、JSONにデータを送信したり、JSON応答に返すフィールドやリストへのアクセスが簡単にできるようになっています。リスト・メソッドを使用して、Webページに返されたJSONリストにアクセスもできます。このライブラリを使用するには、外部リソース XWJDCJS1 (JavaScript file json2.js) と XWJ003 (JavaScript file std_json.js)を自身のWebroutineに追加してください。

以下も参照してください。

[Webroutineの要求](#)

[フィールドの取得](#)

[リストの処理](#)

[メッセージの取得](#)

[コンテキスト・データ](#)

[JSON 要求の構築](#)

[JSON 要求へのフィールド追加](#)

[JSON 要求へのリスト追加](#)

[JSON 要求のリスト・ヘッダーを定義](#)

[JSON 要求リストにエントリーを追加](#)

[JSON 要求の送信](#)

Webroutineの要求

Lstd.Json.getWebroutine(options)メソッドにより、JavaScriptプログラムでWebroutineを呼び出せます。どんなWebroutineでもこのメソッドを使って呼び出すことができますが、データを返すことができるのはJSON応答Webroutineのみです。optionsのパラメータはプロパティのないJavaScriptオブジェクトか、または以下のようなプロパティを含みます。

- Wam** 呼び出されるWAMです。指定しない場合は、現在のWAMが使用されます。
- webroutine** 呼び出されるWebroutineです。指定しない場合は、現在のWebroutineが使用されます。
- Fields** Webroutineに送信する入力値を含むJavaScriptオブジェクトです。例えば次のようになります。
- ```
{
 GIVENAME:"William",
 SURNAME:"Shakespeare"
}
```
- Lists**        Webroutineに送信するリストを含むJavaScriptオブジェクトです。各リストには行が配列されており、各行は複数のカラム値を含むオブジェクトです。例えば次のようになります。
- ```
{
  LIST01:[
    {DEPARTMENT:"ADM", DEPTDESC:"Administration"},
    {DEPARTMENT:"FIN", DEPTDESC:"Finance"}
  ]
}
```
- callback** Ajax応答が完了した時に呼び出されるJavaScriptファンクションです。このファンクションはJSON応答Webroutineへの呼び出しが正常に完了した時のみ呼び出されます。Webroutine出力を表わすWebroutineオブジェクトが含まれた単一パラメータが引き渡されます。

これを全てまとめると、次のようになります。

```
/*
 * Get webroutine (Ajax request)
 * Webroutine (wr) is passed to the callback wrapped in an Lstd.Json.Wr object
 */
var options = {
  wam: "SampleWam",
  webroutine: "Sample1",
  fields: {
    GIVENAME: "John",
    SURNAME: "Smith"
  },
  lists: {
    LIST01: [
      {DEPARTMENT: "ADM", DEPTDESC: "Administration"},
      {DEPARTMENT: "FIN", DEPTDESC: "Finance"}
    ]
  },
  callback: function(wr) {
    // Code to handle the Ajax response goes here
  }
};

Lstd.Json.getWebroutine(options);
```

callbackファンクションに渡されるWebroutineオブジェクトには、サーバーから返されたフィールドやリスト取得用の多くのメソッドが含まれます。

フィールドの取得

キャプションが利用できるのは、WebroutineにOptions(*METADATA)がある時のみです。

```
// Get field
var empno = wr.field("EMPNO");
var empnoLabel = empno.label();
var empnoDesc = empno.description();
var empnoHeadings = empno.headings();
var empnoValue = empno.value();
```

リストの処理

キャプションが利用できるのは、WebroutineにOptions(*METADATA)がある時のみです。

```
// Get list
var list01 = wr.list("LIST01");

// Get list header details
var list01Hdr = list01.headers();

var deptHdr = list01Hdr.col("DEPARTMENT");
var deptHeadings = deptHdr.headings();

// Get list entries
var list01Entries = list01.entries();
var rowCount = list01Entries.rowCount();

list01Entries.each(function(entry) {
var rowNumber = entry.row();
var deptValue = entry.col("DEPARTMENT");
});
```

メッセージの取得

```
// Webroutine messages
var msgs = wr.messages();
var msgsCount = msgs.count();

// Process each message
msgs.each(function(m) {
  alert("Message: " + m);
});
```

コンテキスト・データ

WAMとWebroutineのコンテキスト情報も利用可能です。

```
var ar = [  
  "action-request",  
  "images-path",  
  "language",  
  "partition",  
  "service-name",  
  "session-key",  
  "session-key-name",  
  "session-key-method",  
  "technology-service",  
  "user-id",  
  "webapplication",  
  "webapplication-title",  
  "webroutine",  
  "webroutine-title"];  
  
// List context items  
for (x in ar) alert(ar[x] + ": " + wr.context(ar[x]));
```

JSON 要求の構築

WebRoutine 要求オブジェクトの作成から始めます。

```
var wr = Lstd.Json.factory();
```

JSON 要求へのフィールド追加

フィールド値を追加/変更します。フィールドが既に定義されている場合、その値は置換されます。

```
wr.field("EMPNO", "AA010");  
wr.field("GIVENAME", "John");  
wr.field("SURNAME", "Smith");  
wr.field("SALARY", 1000);
```

JSON 要求へのリスト追加

WebRoutine 要求オブジェクトにリストを追加することで、JSON リストを取得します。

```
var list1 = wr.addList("LIST01");
```

JSON 要求のリスト・ヘッダーを定義

カラム名を設定することで、リスト・ヘッダーを定義します。

```
list1.headers(["DEPARTMENT", "DEPTDESC"]);
```

JSON 要求リストにエントリーを追加

リストにエントリーを追加します。カラム数は、headers() メソッドで設定したカラム名の数と一致しなければいけません。

```
var entries = list1.entries();
entries.add(["ADM", "Administration"]);
entries.add(["MKT", "Marketing"]);
```

JSON 要求の送信

要求を送信します。WAM 名をしない場合のデフォルトは現在のWAMです。

次の例で "wrb" は、Webroutine "MyResponse" により返される WebRoutine JSON オブジェクトです。

```
wr.post({wam: "MyWam", webroutine: "MyResponse", callback:
function(wrb) {
// Handle the response here
}});
```

3.14 WAM出力のファイル保存

WAMはWebブラウザから実行できますが、コマンド・ラインでX_RUNを使ってWAMを実行することもでき、出力ファイルをストリーム・ファイルに保存できます。

WindowsプラットフォームのX_RUNコマンドの構文は以下の通りです。

```
X_RUN PROC=*WAMSP \  
  WMOD={wam_name} \  
  WRTN={web_routine_name} \  
  WAML={markup_language} \  
  PART={partition} \  
  LANG={language} \  
  USER={user} \  
  WASP={output_file_path}
```

説明：

各行の最後に行が続くことを示す"\
"が追加されているのは、コマンド行が分かりやすく表示されるようにするためです。コマンド・プロンプト内では上記を1行のコマンド・ラインとして送信します。

注：IBM i とLinuxでは、必要となるコマンド・ラインが多少異なりますが、X_RUNの引数はほぼ同じです。

WAMを実行して出力をストリーム・ファイルに保存するには、次のX_RUN引数が必要です。

引数 値

PROC 特別な固定値"*WAMSP" は、このファンクションをアクティブ化します。

WMOD 実行するWAMの名前。

WRTN WAM内の実行するWebroutineの名前。

WAML 引数WMODのWAMを実行するマークアップ言語。任意。省略値はLANSA:XHTMLです。

PART 引数WMODのWAMが属する 区画。

LANG 引数WMODのWAM実行時の言語。

USER 引数WMODのWAM実行時のユーザー。プラットフォームによっては任意です。

WASP WAM出力が保存される出力ファイルのパス。パスはWAMを実行するプラットフォームの構文に従います。
例えばWindowsの場合、次のように入力します。C:\Temp\wam_output.html (円記号(\)を使用)IBM i の場合はIFS形式、Linuxはスラッシュ記号を使用します。

更に別のX_RUN引数を追加することもできます。例えば、ITRO、ITRMやITRLを使ってトレースをすることもできます。詳細は『Visual LANSA テクニカルリファレンスガイド』の「[X_RUNパラメータ概要](#)」を参照してください。

例えば次のようになります。

```
X_RUN PROC=*WAMSP WMOD=mywam WRTN=myrtn WAML=LANSA:XHTML \  
PART=DEM LANG=*DFT USER=PCXUSER WASP=C:\Temp\myrtn.html
```

Windowsの場合。

上記のコマンド・ラインは、WAM mywamのWebroutine myrtnを区画DEMで、マークアップ言語LANSA:XHTMLを使用して実行します。そして、出力htmlをストリーム・ファイルC:\Temp\myrtn.htmlに保存します。

Linuxで上記に匹敵するコマンド・ラインは次のようになります。

```
x_run PROC=*WAMSP WMOD=mywam WRTN=myrtn WAML=LANSA:XHTML \  
PART=DEM LANG=*DFT USER=PCXUSER WASP=/tmp/myrtn.html
```

コマンドx_runは小文字で、出力ファイルのパスがUNIX形式になっていることに注意してください。

IBM i で上記に匹敵するコマンド・ラインは次のようになります。

```
call x_run ('PROC=*WAMSP WMOD=mywam WRTN=myrtn WAML=LANSA:XHTML \  
PART=DEM LANG=ENG WASP=/tmp/myrtn.html')
```

ただし、この方法でWAMを実行するといくつかの制限があります。

- Webブラウザとのやりとりがないため、入力されたデータをWAMに渡すことができません。同様に、HTTPやクッキーなどのHTTP要求に関連した情報も使用できません。つまり、入力フィールドやリスト用のWEB_MAPの値が更新されないということです。ですから、フィールドは省略値のままとなり、リストは空になります。
- IBM i の場合、出力ストリーム・ファイルはX_RUNコマンド送信に

使用するユーザー・プロファイル用コードページを使って作成されます。

- アクティブ・セッションを作成するWAMはWAMセッションを作成しますが、このセッションは後続のWAMで使用することはできません。
- アクティブ・セッションを必要とするWAMではInvalid Session eventが起動されます。上記のような理由から、この方法によるWAMセッション管理は避けるようにしてください。

3.15 ドキュメントの型宣言(DOCTYPE)

ここでは、(ブラウザのDOCTYPE"スニッフィング"やスイッチと呼ばれる)レイアウト・モードを管理するDOCTYPE宣言の使い方について説明します。これはDTDの検証用ではありません。

WAMにDOCTYPE宣言を追加するには、エレメントxsl:outputを使用します。例えば次のようになります。

```
<xsl:output method="xml" omit-xml-declaration="yes" encoding="UTF-8"
  indent="no" doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
  doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"
 />
```

異なるウェブレットとメインのWebroutine間での競合を避けるため、DOCTYPE宣言はレイアウト・ウェブレットでのみ定義するようにしてください。

LANSAより提供されるテーマレットは、標準モードになるように厳格なDOCTYPEを使用しています。古いウェブレットではDOCTYPEが全く使用されていません。

HTML5は、DTDがなく簡素化された<!DOCTYPE html>を使用して、ブラウザで標準モードになるようにします。XSLはこのDOCTYPE宣言をサポートしません。

デバイスによっては、この簡素化されたDOCTYPEを探して文書がHTML5かどうか判断するので、WAMは実行時にxsl:outputがXSLフレンドリーでレガシ互換性のあるDOCTYPEを使用している場合は簡素化されたHTML5 DOCTYPEを出力します。

```
<xsl:output method="xml" omit-xml-declaration="yes" encoding="UTF-8"
  indent="no" doctype-system="about:legacy-compat" />
```

4. アドバンスド・トピック

4.1 WEBROUTINEのTRANSFERステートメント

4.2 WEBROUTINEのCALLステートメント

4.3 WAMのセッション管理

4.1 WEBROUTINEのTRANSFERステートメント

簡素なフォームでは、WAMの各WEBROUTINEには1つのXSLスタイルシートが関連付けられています。このXSLスタイルシートがプレゼンテーション層、つまりWebデザインを定義します。WEBROUTINEが呼び出されると、そのWEBROUTINEページのみが返されます。ただし、実行処理をRDMLXから別のページに「リダイレクト」することが必要な場合もあります。例えば、WEBROUTINEを要求したものの、セッションの有効期限が切れている場合です。この場合、要求対象のページとは異なるエラー・ページや他のページを表示する必要があります。

またもう1つの例としては、ログオンWEBROUTINEを呼び出す1つのログオン・ボタンが付いているログオン・ページです。このWEBROUTINEは、ユーザーの認証を行い、それが登録済みのユーザーかゲスト・ユーザーかに応じて、該当するWEBROUTINEに処理を「リダイレクト」して対象のページを表示します。WEBROUTINE間の「リダイレクト」が必要になる例は、他にも多数あります。

この「リダイレクト」を行うには、TRANSFERステートメントを使用します。ターゲットWEBROUTINEへの各フィールドのマッピングは、そのターゲットWEBROUTINEのWEB_MAPステートメントに基づいて行われます。ターゲットWEBROUTINEに渡されるのは、そのWEB_MAPに受信用として指定されているフィールドやリストのみになります。

TRANSFERステートメントには、以下のプロパティがあります。

プロパティ 説明

TOROUTINE ターゲットWEBROUTINEの名前を指定します。別のWAMを指定することも可能です。その場合は、WAM名の後にWEBROUTINE名を指定し、その間をピリオドで区切ります(例えば、#PSLSYS.Browseのように入力します)。

サービス名も指定できます。その場合は、*SERVICE修飾子を接頭辞として使用します。

フィールドから値を取り込むことも可能です。その場合は、*EVALUATE修飾子を接頭辞として使用します。

OnEntry このプロパティは、ターゲットWEBROUTINEに受信用のフィールドやリストをマッピングするために使用します。以下のいずれかの値になります。

*MAP_NONE (フィールドやリストのマッピングを行わない)

*MAP_ALL (すべての必要なフィールドやリストのマッピングを行う)

*MAP_LOCAL (WEBROUTINEのWEB_MAPに指定されているフィールドやリストのマッピングだけを行う)

*MAP_SHARED (WEBROUTINEレベルではなく、WAMレベルでWEB_MAPに指定されているフィールドやリストのマッピングだけを行う)

デフォルトは、*MAP_ALLです。

TRANSFERが実行されると、ターゲットWEBROUTINEが実行制御を取得し、ソースWEBROUTINEに制御が戻ることはありません。ターゲットWEBROUTINEが終了すると、このWEBROUTINEがブラウザから直接呼び出された場合と同じように、このWEBROUTINEのページが返されます。

TRANSFERステートメントの例を以下に示します。

```
TRANSFER TOROUTINE(Browser)
```

```
TRANSFER TOROUTINE(#PSLSYS.Browser)
```

```
TRANSFER TOROUTINE(*SERVICE EmployeeBrowse)
```

```
change #WEBRTN 'wam1.routine1'
```

```
TRANSFER TOROUTINE(*EVALUATE #WEBRTN)
```

4.2 WEBROUTINEのCALLステートメント

別のWEBROUTINEに実行処理をリダイレクトするために、CALLステートメントを使用することもできます。CALLステートメントはTRANSFERステートメントと基本的に同じですが、CALLの場合は、呼び出し先のWEBROUTINEが呼び出し元のWEBROUTINEに制御を戻し、その時点から実行処理が継続されていく点が異なります。

CALLステートメントには、以下のプロパティがあります。

プロパティ 説明

WEBROUTINE 転送先のWEBROUTINEの名前を指定します。別のWAMを指定することも可能です。その場合は、WAM名の後にWEBROUTINE名を指定し、その間をピリオドで区切ります(例えば、#PSLSYS.Browseのように入力します)。

サービス名も指定できます。その場合は、*SERVICE修飾子を接頭辞として使用します。

フィールドから値を取り込むことも可能です。その場合は、*EVALUATE修飾子を接頭辞として使用します。

OnEntry このプロパティは、ターゲットWEBROUTINEに受信用のフィールドやリストをマップするために使用します。以下のいずれかの値になります。

*MAP_NONE (フィールドやリストのマッピングを行わない)

*MAP_ALL (すべての必要なフィールドやリストのマッピングを行う)

*MAP_LOCAL (WEBROUTINEのWEB_MAPに指定されているフィールドやリストのマッピングだけを行う)

*MAP_SHARED (WEBROUTINEレベルではなく、WAMレベルでWEB_MAPに指定されているフィールドやリストのマッピングだけを行う)

デフォルトは、*MAP_ALLです。

OnExit このプロパティは、ターゲットWEBROUTINEに送信用のフィールドやリストをマッピングする際に使用し

ます。以下のいずれかの値になります。

*MAP_NONE (フィールドやリストのマッピングを行わない)

*MAP_ALL (すべての必要なフィールドやリストのマッピングを行う)

*MAP_LOCAL (WEBROUTINEのWEB_MAPに指定されているフィールドやリストのマッピングだけを行う)

*MAP_SHARED (WEBROUTINEレベルではなく、WAMレベルでWEB_MAPに指定されているフィールドやリストのマッピングだけを行う)

デフォルトは、*MAP_ALLです。

CALLステートメントの例を以下に示します。

CALL WEBROUTINE(Browser)

CALL WEBROUTINE(#PSLSYS.Browser)

CALL WEBROUTINE(*SERVICE EmployeeBrowse)

CALL WEBROUTINE(*EVALUATE #WEBRTN)

4.3 WAMのセッション管理

Webを基盤としたユーザーとサーバーの対話では、同じユーザー/クライアントからサーバーに多数の要求が送られます。サーバーの観点からすれば、各要求はそれぞれ固有の要求です。一方、HTTPはステートレスなプロトコルであり、同じユーザー/クライアントから送られてくる複数のWeb要求を1つの連続したセッションとして管理するための絶対的なメカニズムがありません。詳細は、「[状態非依存](#)」を参照してください。

このような課題に対応するために、Webベースのトランザクション型のほとんどのアプリケーションでは、セッション管理のための独自のメカニズムを用意しています。WAMでは、宣言型の絶対的で安全なセッション管理メカニズムが使用されています。またセッション管理は、アプリケーションの残りの部分にもできるだけ透過的になるよう設計されています。

Webセッションのデータや状態は、サーバー上のデータベースに格納され、サーバーとブラウザーの間でやり取りされる一意のセッション・キーにより識別されます。ブラウザーには、セッションを明確に終了させるメカニズムがないので、指定の時間内に同じユーザーからの要求が送られてこなかった場合にセッションを終了するタイムアウト・メカニズムが用意されています。

詳細については、以下の項目を参照してください。

4.3.1 セッション管理の構成

4.3.2 セッション・キー方式

4.3.3 WEB_MAPの*PERSISTキーワード

4.3.4 セッション状態の保守

4.3.5 セッション管理の仕組み

4.3.6 WAMのセッション・プロパティ

4.3.7 WAMセッションの例

4.3.8 セッションの状態

WAMアプリケーションには、サーバーに依存しない形で、サーバー上にセッション状態を格納する機能があります。セッション状態を宣言するには、フィールドやリストのOPTIONS属性を*PERSISTに設定しなければなりません。*PERSISTを使用すると、そのセッションのコンテキストでWEBROUTINEが実行されている間は、フィールドやリストの変更内容がサーバー上で常にアクセス可能な状態になります。実際のセッ

セッション状態はデータベースに格納されるので、複数ストリームの実行環境を実現できます。つまり、複数のアプリケーション・サーバーをそれぞれ別々のマシンにセットアップ/構成して、拡張性を高めることが可能です。そのようにして、すべての使用可能なサーバーの間で個々のWEBROUTINE要求を共有できます。要求が出されると、WEBROUTINEのRDMLXの実行前に、セッション状態(つまり、OPTIONS(*PERSIST)が設定されているすべてのフィールドやリスト)が1つのデータベースからサーバー・メモリーに再ロードされます。WEBROUTINEが終了すると、セッション状態もアンロードされます。

4.3.1 セッション管理の構成

セッションの構成は、WAMのプロパティ値を直接指定して行います。セッション・キーのブラウザーへの格納方法やセッションの非アクティブ・タイムアウト時間などを設定するプロパティがあります。(詳細は「[4.3.6 WAMのセッション・プロパティ](#)」を参照)

デフォルトでは、それぞれのWAMが他のWAMからは独立した形で独自のセッション状態を保持しますが、複数のWAMセッションを1つのセッションとして連鎖的に結合することも可能です。1つのWebアプリケーションに複数のWAMが含まれている場合に、この連鎖的な結合は便利です。

WAMでは、セッションの有効期限が切れた時やセッションが無効になった時の動作を指定することができます。これにはセッションが無効、またはセッションの有効期限が切れたときに発生するイベントのイベント・ハンドラーを準備します。そのイベント・ハンドラーのコードを使って、作成したエラー・ページを表示したり、ログオン・ページへのリダイレクトなど様々な機能を提供することができます。

4.3.2 セッション・キー方式

クライアントに対するセッションを識別するため、セッションの作成時に一意のセッション・キーが各セッションに割り当てられます。このセッション・キーはブラウザーに送り返されます。

ブラウザーは、ある要求が特定のセッションに属していることを示すために、各要求ごとにセッション・キーをサーバーに渡さなければなりません。このセッション・キーの受け渡しは、アプリケーションの残りの部分に透過的に行われます。

WAMでは、ブラウザーでセッション・キーを管理するための3つの方法をサポートしています。

1. 各ページに返される非表示フィールドに格納する方法
2. URLに渡す方法
3. ブラウザーのメモリーに保持されるCookieに格納する方法。この場合は、標準CookieとセキュアCookieのどちらを使用するかを選択できます。セキュアCookieを選択した場合は、SSL (HTTPS)接続でのみセッション・キーの受け渡しが可能になるので、セッション・キーの漏洩の危険を回避できます。言い換えれば、セキュアCookieのメカニズムを使用するアプリケーションは、HTTPSプロトコルでブラウザーに接続する必要があります。

この中で最も安全なのは、3つ目のセキュアCookieを使用する方法で、視覚的な盗用もプログラミングの盗用も非常に困難になります。

jQuery Mobile では、非表示フィールドのセッション・キー方式は作動しません。これは、ページ全体の更新なしにページがロードされるからです。代わりにURL または Cookie 方式を使用してください。

4.3.3 WEB_MAPの*PERSISTキーワード

WEB_MAPのOPTIONSプロパティの中では、*PERSISTオプションを使用することができます。WAMはサーバー側のWebベースの環境で実行されるように設計されています。スケーラビリティを最大限に高め、リソース使用量を最小限に抑えるために、WAMコンポーネントはステータスレスな性質を持っています。つまり、WAMのメモリー状態が保持されるのは、WEBROUTINEの実行中だけです。WEBROUTINEの実行が終了すると、メモリー状態(フィールドやリストの値など)は破棄されます。詳細は「[4.3 WAMのセッション管理](#)」を参照してください。

Webセッションが有効な間、複数のWEBROUTINEの実行中にデータを存続させて保持するフィールドやリストを宣言することもできます。そのためには、WEB_MAPで指定するフィールドやリストにOPTIONS(*PERSIST)を宣言します。このOPTIONS(*PERSIST)の宣言に加えて、WEBROUTINEの初期のSessionStatusをActiveに設定することも必要になります。これには、WAMに対してSessionStatusを設定する方法、または個々のWEBROUTINEに対してOnEntry(*SessionStatus_Active)キーワードを設定する方法があります。このような方法で、SessionStatusをアクティブに設定することにより、WEBROUTINEが実行を開始する前に、WEBROUTINEはセッション状態をロードするようになります。

WEBROUTINEとの対応関係の方向を指定しないで、セッション・データを宣言するだけの場合は、以下の構文を使用します。

```
WEB_MAP FOR(*NONE) FIELDS #で始まるフィールドおよびリスト名 OPTIONS(*PERSIST)
```

FOR(*NONE)宣言を使用すると、宣言されたフィールドやリストとWEBROUTINEとの対応関係の方向は定義されませんが、フィールドやリストの値は、WEBROUTINEの実行をまたぐようにして、WEBROUTINEのRDMLXで使用できるようになります。上のWEB_MAPをWAMレベルで宣言すると(つまり、BEGIN_COMの後で、しかもすべてのWEBROUTINEブロックの外に記述すると)、セッションを必要とする(つまり、初期のSessionStatusがNoneではない)すべてのWEBROUTINEが、セッション状態を格納したそれらのフィールドやリストにアクセスできます。

4.3.4 セッション状態の保守

WAMのセッション管理の最大のメリットの1つは、有限のリソースであるサーバー・メモリーを使用しないでセッションの状態やデータを保持できるということです。WAMには、個々の要求範囲を超えてセッション単位で保持するフィールドやリストを指定できる宣言メカニズムが用意されています（これは「[4.3.3 WEB_MAPの*PERSISTキーワード](#)」に記述されています）。セッション・データとして宣言されているフィールドやリストは、データベース内で管理され、セッションの各要求ごとにサーバー・メモリーとの間で出し入れされます。このメカニズムを使用すれば、セッション状態が1つのデータベースに格納されていれば、(複数サーバー構成の)個々のアプリケーション・サーバーが同じセッション内の別の要求に対応することができるようになります。

1つのセッション内の2つの要求の間の時間は、非アクティブ・タイムアウト時間の範囲内であればなりません。この非アクティブ・タイムアウトは要求ごとにリセットされます。このセッションのタイムアウト時間が過ぎた後に要求が出された場合は、そのセッションの有効期限が切れたと見なされるので、その状況に対応するための適切な処理をアプリケーション・レベルで実行する必要があります。例えば、無効なセッション・キーを処理するためのRDMLXのイベント・ハンドラーを記述するなどです。(イベント・ハンドラーが用意されていない場合は、汎用のエラー・ページが返されます。)

セッションの有効期限が切れても、そのセッションはただちにデータベースから削除されるわけではありません。有効期限が切れたというフラグが設定されるだけです。有効期限が切れたセッションの削除は、別の削除プロセスによって行います。このプロセスは、データベースからピーク時以外に実行できるバックグラウンド・プロセスとして構成できます。データベースのクリーンアップについては『LANSA for the Web 管理ガイド』の「[WAM セッションのクリーンアップ](#)」を参照してください。

4.3.5 セッション管理の仕組み

WEBROUTINE要求を処理するための、構成可能なWebジョブの事前設定プールがあります。現在の要求を処理中のWebジョブは、必要な状態をロードし、内部状態をリセットした後、次の要求のために自身を使用可能な状態にします。この次の要求は別のWebセッションから来ることもあります。Webジョブが負荷を共有するのは、個々の要求レベルであって、セッション・レベルではありません。したがって、実行中のWebジョブの数よりもはるかに多くのアクティブなブラウザ・セッションを実行できます。

WAMの実行環境のステートレス(状態非依存)の性質には、重要な意味があります。WAMコンポーネントの状態は、WEBROUTINEの実行中しか存在しません。フィールドやリストを*PERSISTとして宣言しない限り、それらの値は、WEBROUTINEの終了後に破棄されるので、次の要求で使用することはできません。ですから、WEBROUTINEの実行中にインスタンス化された他のRDMLXコンポーネントも、WEBROUTINEの終了時に破棄されます。ところが、*PERSISTのフィールドやリストは、要求の終了以降も存在し続け、引き続き使用可能な状態になります。つまり、セッションの終了まで、複数のWEBROUTINE要求にまたがるようにして存在します。

セッション管理のまとめ

WAMの実行環境の重要な特色を以下にまとめます。

- WAMコンポーネントは、RDMLXコンポーネントの一種です。他の表示されないRDMLXコンポーネントをインスタンス化して、呼び出すこともできます。
- WAMの中の各WEBROUTINEは、ブラウザから呼び出せます。サービス名のメカニズムを使用して公開することも可能です。
- WAMは、Webセッションという概念をサポートしています。Webセッションを識別するためのセッション・キーは、ブラウザに格納されます。
- ブラウザー(または他の種類のクライアント・デバイス)からWEBROUTINE要求が出されると、WAMコンポーネントが作成され、要求されたWEBROUTINEが実行され、次にWAMコンポーネントが破棄されます。このプロセスによって、サーバーのリソース使用量を削減し、拡張性を最大限に高めることができます。
- WAMの実行では、要求ごとに状態が保持されるので、あらかじめ定

義されている構成可能な複数のWebジョブが多数のWebセッションに対応でき、複数のアプリケーション・サーバー・マシンの間で要求を共有することも可能になります。

- (フィールド、リスト、その他のコンポーネントの状態の)全データは、WEBROUTINEの終了時に破棄され、後続の要求では使用できません。
- OPTIONS(*PERSIST)によってセッション状態として宣言されているフィールドやリストは、破棄されずに、サーバー上のデータベースに格納されます。つまり、1つのWEBROUTINEでそのデータを変更したら、後続のWEBROUTINE要求でもその変更後のデータを使用できます。
- 指定のタイムアウト時間内にセッションから要求が出されないと、そのセッションは有効期限が切れたと見なされます。その後その同じセッションから要求が出されると、SessionInvalidイベントが起動されます。このイベントのイベント・ハンドラーを使用すれば、セッションを処理できます。

4.3.6 WAMのセッション・プロパティ

セッションの構成は、WAMのプロパティ値を直接指定して行います。セッション・キーのブラウザへの格納方法やセッションの非アクティブ・タイムアウト時間設定のプロパティがあります。デフォルトでは、それぞれのWAMが他のWAMからは独立した形で独自のセッション状態を保持します。

以下の表は、WAMのセッション管理に関連したプロパティをまとめています。

プロパティ	値	説明
SessionKeyMethod	<i>URL</i> 、 <i>Cookie</i> 、 <i>SecureCookie</i> または、 <i>HiddenField</i>	セッション・キーと呼ばれるセッション識別子をブラウザに格納する方法です。最も安全な方式は、セキュアCookieとされています。このプロパティは、実行時には使用できません。
SessionStatus	<i>Active</i> 、 <i>Invalid</i> 、 <i>Expired</i> もしくは <i>None</i>	実行時には、このプロパティに基づいて、Webセッションの状態を判別できます。このプロパティをActiveに設定してセッションを作成したり、Invalidに設定してセッションを削除したりすることも可能です。デザイン時には、WAM全体(つまり、全てのWEBROUTINE)に対してこのプロパティを設定することもで

		<p>きますし、キーワードOnEntryを使用して各WEBROUTINEに対してこのプロパティを設定することも可能です。デザイン時にActiveの値を設定すると、WEBROUTINEの実行前にセッションの検証が行われます。Noneの値を設定すると、セッションが無効または有効期限切れの場合にもWEBROUTINEが実行されません。その後、そのWEBROUTINEの中でSessionStatusをActiveに設定して新しいセッションを作成し、セッション・キーをブラウザーに返すことも可能です。このプロパティは、デザイン時にも実行時にも使用できます。</p>
SessionTimeout	<p>秒単位の数値。0に設定した場合は、システム全体のデフォルトが使用されます。-1に設定した場合は、タイムアウト時間が無限になります(つまり、タイムアウトがなくなります)。</p>	<p>要求が出されない時間の長さがタイムアウト値に達すると、セッションの有効期限が切れたと見なされます。セッションを必要とする(つまり、SessionStatusがActiveに設定されて</p>

		<p>いる)WEBROUTINE に対してその後から 要求が出されても、 そのWEBROUTINE は実行されず、 SessionInvalidイベン トが起動されます。 このタイムアウト は、WAMの中のす べての WEBROUTINEに適 用されます。 クリーンアップを後 で実行するようスケ ジュールすることが できるので、この セッション状態は セッション・ストア に残っている場合も あります。 このプロパティは、 実行時には使用でき ません。</p>
SessionGroupName	任意の英数字。	<p>それぞれのWAM は、他のWAMから は独立した形で独自 のセッション状態を 保持します。1つの Webアプリケーション に複数のWAMを 組み込んだ場合は、 関係するすべての WAMで同じセッ ション状態を参照す るために、このプロ パティに同じ識別子 を設定できます。こ のプロパティは、実</p>

		行時には使用できません。
--	--	--------------

WAMの中のすべてのWEBROUTINEは、WAM全体で定義されているSessionStatusプロパティから初期のSessionStatus値を割り当てられます。ただし、OnEntryキーワードに以下の値を設定すれば、WEBROUTINE単位でSessionStatusを上書きできます。

値	説明
*SessionStatus_Active	WEBROUTINE実行時にアクティブなセッションが使用されます。
*SessionStatus_None	WEBROUTINE実行用のセッションは不要になります。
*SessionStatus_Of_WAM	OnEntryキーワードを指定しない場合のデフォルトです。WAM全体のSessionStatusプロパティの値が使用されます。

4.3.7 WAMセッションの例

WAMセッションを管理するためのSessionStatusプロパティを理解しておくことは重要です。

以下は、セッションを管理するWAMの典型的な例です。

```
FUNCTION OPTIONS(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_WAM) SESSIONSTATUS(Active)
```

```
*The following line declares Session state #CUSTNAME field
WEB_MAP FOR(*NONE) FIELDS(#CUSTNAME) OPTIONS(*PERSIST)
```

```
WEBROUTINE NAME(Start) DESC('Initial Page') OnEntry(*SessionStatus_1
WEB_MAP FOR(*OUTPUT) FIELDS(#USERID #PASSWORD)
ENDROUTINE
```

```
WEBROUTINE NAME(Logon) DESC('Logon Page') OnEntry(*SessionStatus
WEB_MAP FOR(*INPUT) FIELDS(#USERID #PASSWORD)
```

...

```
* Some authentication logic, if authentication fails can TRANSFER back to St
```

```
* The following line will create a session, when WEBROUTINE exits
#COM_SELF.SessionStatus := Active
```

```
TRANSFER TOROUTINE(WelcomePage)
ENDROUTINE
```

```
WEBROUTINE NAME(WelcomePage) DESC('Welcome Page')
ENDROUTINE
```

```
WEBROUTINE NAME(Logoff) DESC('Logoff page')
#COM_SELF.SessionStatus := Invalid
ENDROUTINE
```

```
* The following event handler will handle invalid sessions and
```

```
* TRANSFER back to starting page, for logon
```

```
EVTROUTINE HANDLING(#COM_OWNER.SessionInvalid) OPTIONS(*N
TRANSFER TOROUTINE(Start)
ENDROUTINE
```

END_COM

- SessionStatusは、Activeに設定されています。デフォルトでは、このWAMの中にあるすべてのWEBROUTINEの実行前に、セッションが有効になっている必要があります。SessionTimeoutは、5分です。
- 2つのWEBROUTINE (StartとLogon)では、OnEntry(*SessionStatus_None)キーワードによって、SessionStatusがNoneに上書きされています。Start WEBROUTINEは、ログオンの詳細情報を入力するためのページを表示します。これは最初のページなので、有効なセッションがあってもなくても実行する必要があります。したがって、SessionStatusをNoneにします。
- Logon WEBROUTINEは、Startページでログオン・ボタンがクリックされたときに呼び出されます。その時点では、まだ有効なセッションが存在しないので、このWEBROUTINEについても、セッションなしで実行できるように設定する必要があります。
- Logonは、ユーザーIDとパスワードの検証を行い、それらが有効な場合は、SessionStatusをActiveに設定します。SessionStatusをNoneからActiveに設定するので、Logon WEBROUTINEが終了すると、新しいセッションが作成されます。
- Logon WEBROUTINEは、WelcomePage WEBROUTINEへのTRANSFERを実行します。この時点ではアクティブなセッションがあるので、(アクティブなセッションを必要とする)WelcomePageの実行が可能になります。WelcomePageが終了すると、新しいセッションのセッション・キーがブラウザに送り返されます。これ以降、同じブラウザから新しいWEBROUTINE要求が出されるたびに、そのセッション・キーがサーバーに送られ、サーバーがセッションを識別できるようになります。
- StartページへのTRANSFERを実行するだけのSessionInvalidハンドラーもあります。このハンドラーが呼び出されるのは、セッション・キーが存在しないか無効な状態、またはセッション・キーは有効でもセッションの有効期限が切れた状態(つまり、要求が出されない時間が5分を超えた状態)でWEBROUTINEを呼び出そうとしたときです。セッションが無効または有効期限切れの場合は、ユーザーIDとパスワードを入力するためのStartページが必ずブラウザに表示されます。
- 確実にセッションを無効にするためのログアウト・ボタンやメニューをWAMアプリケーションで用意することもできます。その場

合は、Logoff WEBROUTINEが呼び出され、SessionStatusがInvalidに設定され、WEBROUTINEの終了時にセッションが無効になります。

4.3.8 セッションの状態

セッションには、以下の4つの状態があります。

状態 意味

- Invalid セッションを識別するためのセッション・キーが提供されていないか、サーバーによって認識されているセッション・キーと一致しない状態
- Expired セッション・キーは有効なセッションを示しているものの、同じセッションから出された前の要求からの経過時間がSessionTimeoutの値を超えている状態
- Active セッション・キーが既知の有効なセッションを示しており、同じサーバーから出された前の要求からの経過時間がSessionTimeoutの値を超えていない状態
- None デザイン時に宣言的に使用され、有効なセッションのコンテキストなしでWEBROUTINEが実行できることを示す。WAMレベルでSessionStatusにActiveが設定される場合は、少なくとも1つのWEBROUTINEでOnEntry(*SessionStatus_None)キーワードを指定し、そのWEBROUTINEからWAMアプリケーションに入って新しいセッションを作成できるようにする必要がある。

考え方としては、無効なセッションと有効期限が切れたセッションの扱いは同じです。いずれの状態でも、有効なセッションのコンテキストを必要とする(つまり、SessionStatusプロパティがActiveに設定されている)WEBROUTINEは実行されません。

WAMでは、SessionInvalidイベント・ハンドラーを用意することによって、無効または有効期限切れのセッションを処理できます。このハンドラーでは、別のWEBROUTINEへのTRANSFERや、無効または有効期限切れのセッションを処理する他の操作を実行できます。SessionInvalidハンドラーを用意すれば、(アプリケーションでセッション確認が必須であったとしても)すべてのWEBROUTINEのチェックが不要になります。全てのセッション確認はWAMレベルで行われ、無効なセッションについてはSessionInvalidイベントが起動されるので、無効なセッションの処理

をWAMレベルでカスタマイズできます。

SessionInvalidハンドラーを用意しない場合は、WAMの無効なセッションのための標準ページがブラウザに返されます。

セッションの有効期限が切れても、セッション状態はただちにセッション・データベースから削除されるわけではありません。これは、ピーク時のサーバーの負荷を減らすための動作です。トランザクション・モニターは、有効期限が切れたセッション状態のクリーンアップを周期的に実行します。この周期的なクリーンアップを無効にして、ピーク時以外にクリーンアップを実行するためのスケジュールを設定することも可能です。

5. WAMアプリケーションの実行

5.1 WAMのビルドあるいはコンパイル

5.2 配布と実行時の環境

5.3 WAMとXSLの配布

5.4 複数層の配布

5.5 WAMのURL (Uniform Resource Locator)

5.6 送信されるHTTPデータとフィールドの対応関係

5.7 プレゼンテーション出力の作成方法

5.1 WAMのビルドあるいはコンパイル

LANSAエディターのツールバーのアイコンを使用して、WAMをビルドすることができます。

または、

コンパイルするには、このアイコンを使用します。

これらのアイコンを使用する時、ビルドのために選択されるオプションは、LANSAで提供される省略値、もしくは前回のコンパイル時に選択されたオプションのどちらかになります。

ビルドとコンパイルの違い

ビルドは任意の処理で、実行可能なオブジェクトは一切作成しません。ビルドの過程は以下の通りです。

- WAMの保存。
- プログラムのフル・ファンクション・チェック(FFC)。
- 既存のWebデザインがない場合は、以下も行います。
 - WAMレイアウト用のデザイン (XSL スタイルシート)の生成。
 - Webroutine用のWebデザイン (XSL スタイルシート)の生成。

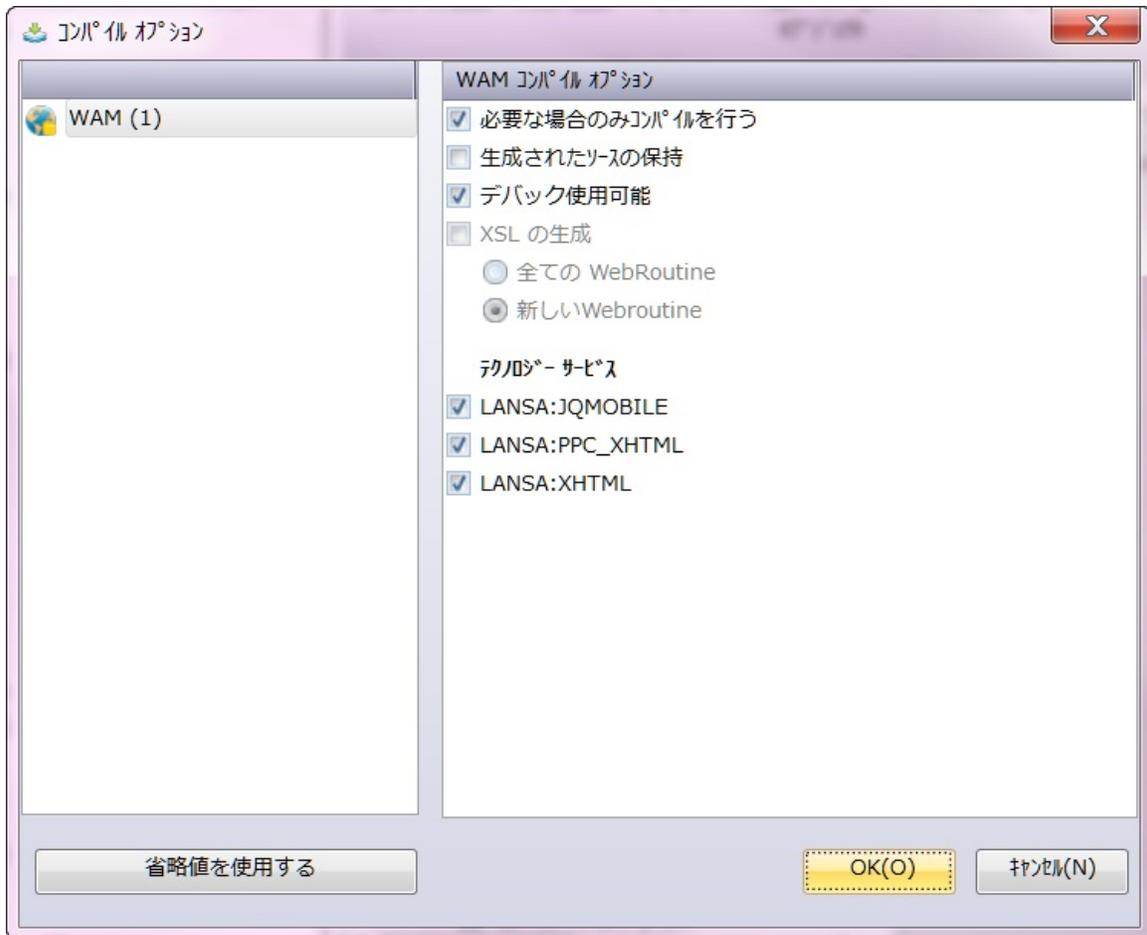
コンパイルは、実行可能なオブジェクトが作成されること、そしてWebroutineのXSL生成がコントロールできることを除いては、ビルドと同じ処理です。

WAMのコンパイルはWAMを実行する前に必ず行わなくてははいませんが、ビルドは好きなだけ何回でも行うことができます。

ビルドもコンパイルもどちらもWebroutineのWebデザインを生成します。WAMに必要なレイアウト・デザインが見つからない場合は生成されます。レイアウトのデザインは、必要であれば後でLANSAエディターで修正することができます。

コンパイル・オプション

コンパイルのオプションを確認・変更するには、[コンパイル]を選択します。コンパイル・オプションのダイアログが開きます。



このダイアログのオプションは一目瞭然ではありますが、『LANSA テクニカル リファレンスガイド』の「[コンポーネント・コンパイル・オプション](#)」に詳細が説明されています。

選択されたコンパイル・オプションは、後に行われるコンパイルのために保持されますが、XSL生成についてはその限りではありません。XSL生成は安全策として、常に新しいWebroutineのみを生成するよう設定されています。

注意が必要なオプションは以下の通りです。

[XSL の生成]

[全てのWebroutine]

このオプションを選択した場合、Webroutineのウェブ・デザインは再生成されます。これらのWebroutineのデザインに対してLANSAエディターを介して加えられた修正は全て上書きされます。

[新しいWebroutine]

これが省略値のオプションです。このオプションではウェブデザインの生成は新しいWebroutineについてのみ行われ、既存のデザインについて

は上書きされません。

[テクノロジー サービス]

このWAMを使用するテクノロジー・サービスを選択してください。

5.2 配布と実行時の環境

ここからはWAMベースのWebアプリケーションの配布と実行時の環境を説明します。

次のトピックを確認してください。

[5.3 WAMとXSLの配布](#)

[5.4 複数層の配布](#)

[5.5 WAMのURL \(Uniform Resource Locator\)](#)

[5.6 送信されるHTTPデータとフィールドの対応関係](#)

[5.7 プレゼンテーション出力の作成方法](#)

5.3 WAMとXSLの配布

WAMの開発作業が終わり、webページのデザインとテストが終了したら、アプリケーションを実稼働環境に配布する準備が整ったこととなります。配布パッケージの作成には、LANSA配布ツールを使用します。配布ツールでは、WAMだけでなく、Webroutine Webデザイン（XSLスタイルシート）をはじめとするすべての従属オブジェクトを選択できます。

アプリケーション全体を配布する場合、以下の作業を行います。

- 必要なWAMを選択。
- 全てのクロス・リファレンス設定を選択して、関連付けられたオブジェクトを含む。
- テクノロジ・サービスの選択（省略値はLANSA XHTML）。
- 配布パッケージの作成。
- 実稼働環境へのパッケージの配布。

配布ツールで識別できない外部の静的オブジェクト(イメージ・ファイル、スクリプト・ページ、他のHTMLページなど)をアプリケーションが使用する場合は、それらのオブジェクトを別途配布する必要があります。

詳細は『LANSAアプリケーション配布ツール』ガイドの「[Windowsアプリケーション・データベースを使用するWAMアプリケーション](#)」のWAMの説明箇所を参照してください。

5.4 複数層の配布

LANSA for the Webは、単一層インストールと複数層インストールの両方をサポートしています。

単一層インストールでは、Webサーバーとデータ/アプリケーション・サーバーを1つのマシンに配置します。複数層インストールでは、Webサーバーとデータ/アプリケーション・サーバーを別々のマシンに配置します。

Windows IIS Webサーバーを使用した複数層構成の場合は、WAMのXSL処理がデフォルトでWebサーバー層に移され、処理の負荷が各層の間で分散されます。このオプションは、LANSA ISAPIプラグインのWebアドミニストレータのローカル構成で設定できます。

詳細については、『*LANSA for the Web 管理ガイド*』の「[複数層 Web システムの設定](#)」を参照してください。

5.5 WAMのURL (Uniform Resource Locator)

URLを使用してブラウザーからWebroutineを呼び出すには、2つの方法があります。

WAMとWebroutineの一意の組合せにより、次のように指定できます。

```
http://localhost/cgi-bin/lansaweb?wam=<WAM名>&webrtn=<Webroutine名>&ml=<TS名>&part=<区画名>&lang=<言語名>
```

Webroutineにサービス名が関連付けられている場合は、一意のサービス名により、次のように指定できます。

```
http://localhost/cgi-bin/lansaweb?srve=<サービス名>&ml=<TS名>&part=<区画名>&lang=<言語名>
```

Webroutineを呼び出す上記の方法はどちらも拡張して、(ウェブイベントのように)URLにパラメータを含むことができます。URLのパラメータとして使用するフィールドは、Webroutineで*INPUTとして指定する必要があります。基本的なシンタックスは、以下の通りです。:

```
http://server/cgi-bin/lansaweb?  
w=wam&r=rtn&f(fieldname)=fieldvalue&f(fieldname2)=fieldvalue2
```

以下の表は、URLの各キーワードをまとめたものです。

キー ワード 名	短 縮 形	必須/任意	説明
field	f	任意。	値は、必要なfield (フィールド名)=フィールド値のパラメータをURLに追加することで、マップされたフィールドに送られます。例： field(Section)=AB
language	lang ま た はl	任意。指定しない場合は、区画の省略値言語が使用されます。	識別子languageは、ページの表示に使用する言語を決定します。区画の省略値言語とは別の特別な言語のWebroutine XSLスタイルシートとウェブレット・スタイルシートを作成することも可能です。指定した言語のページまたはウェブレットが区画に存在しない場合は、省略値言語のページが返されます。

markup	ml	任意。指定しない場合は、構成された省略値が使用されます。	ページの表示に使用するテクノロジー・サービスです。HTMLの場合は、省略値のLANSA:XHTMLテクノロジー・サービスが使用されます。この省略値はLANSA for the Web アドミニストレータを使って変更することができます。
partition	part また はp	任意。指定しない場合は、構成済みの強制区画が常に使用されません。	Partition識別子では、使用するLANSA区画を指定します。強制区画が(LANSA for the Webアドミニストレータで)サーバーに構成されていれば、このパラメータは無視されます。
srve	s	必須。最初のキーワードとして記述する必要があります。webappまたはsrveのいずれかのキーワードを使用しなければなりません。	特定のWAM、Webroutine、区画、言語にマップされている登録済みのサービス名です。このパラメータを使用する時に指定のサービス名が登録済みであれば (WebroutineのServiceNameプロパティで指定できます)、これらの値は登録された項目から読み取られるので、区画と言語のパラメータを指定する必要はありません。これらの値を指定すると、登録された項目が上書きされます。 詳細は、「 サービス名の使用 」を参照してください。
webapp	wam また はw	必須。最初のキーワードとして記述する必要があります。webappまたはsrveのいずれかのキーワードを使用しなければなりません。	呼び出し対象のWebroutineが入っているWAMの名前です。
webrtn	r	必須。2番目のキーワード	呼び出し対象のWebroutineの名前です。

として記述する
必要があります。
ます。

webappキー
ワードと一緒に
使用しなければ
なりません。

同様のこれに代わる表示もサポートされます。（以前のバージョンで使用されていたもの）キーワード-値のペアを分けるには、'&'の代わりに '+' が使用されることに注意してください。

WAMとWebroutineを起動させる例：

```
http://localhost/cgi-bin/lansaweb?webapp=<WAM名>+webrtn=<Webroutine名>+ml=<TS名>+part=<区画名>+lang=<言語名>
```

サービス名を呼び出す例：

```
http://localhost/cgi-bin/lansaweb?srve=<サービス名>+ml=<TS名>+part=<区画名>+lang=<言語名>
```

キャッシュ・ページの使用を防ぐため、LANSAはGET要求を発行する際にURLにタイムスタンプを追加して、この要求が"異なる"リソース用の要求として理解されるようにします。例えば次のようになります。

```
http://localhost/cgi-bin/lansaweb?  
wam=WAM01&webrtn=WR01&ml=LANSA:XHTML&part=DEX&lang=EN  
1343366108313
```

5.6 送信されるHTTPデータとフィールドの対応関係

ブラウザのフォームがサーバーに送信されると、指定された入力ボックスとそれに含まれるデータも送信されます。送信されるデータは、フォームの各入力ボックスの名前/値のペアの形式になっています。サーバーは、送信された値を処理して、結果ページを作成します。

Webroutineページがブラウザに表示されると、そのページには、送信用のWEB_MAPに指定されている各フィールドの入力ボックス(または他の複雑なコントロール)が組み込まれています。フォームの各入力ボックスは、WEB_MAPに指定されているフィールド名と同じフィールド名を使用します。(接頭辞は使用されません。)

このページが別のWebroutineに送信されると、WAMの実行時環境が、そのWebroutineの受信用のWEB_MAPステートメントに指定されているフィールドに、入力ボックス名と一致するフィールド名に基づいて入力値を割り当てます。したがって、送信された値は適切なフィールドに割り当てられ、Webroutineの実行処理が始まると使用できるようになります。

Browser Form submits Name/Value pairs

```

WEBROUTINE NAME(AddEmployee)
  WEB_MAP FOR(*BOTH) FIELDS((#SURNAME
  *OUTPUT) (#GIVENAME *OUTPUT) (#ADDRESS1
  *OUTPUT) (#ADDRESS2 *OUTPUT) (#ADDRESS3
  *OUTPUT) (#SALARY *OUTPUT))

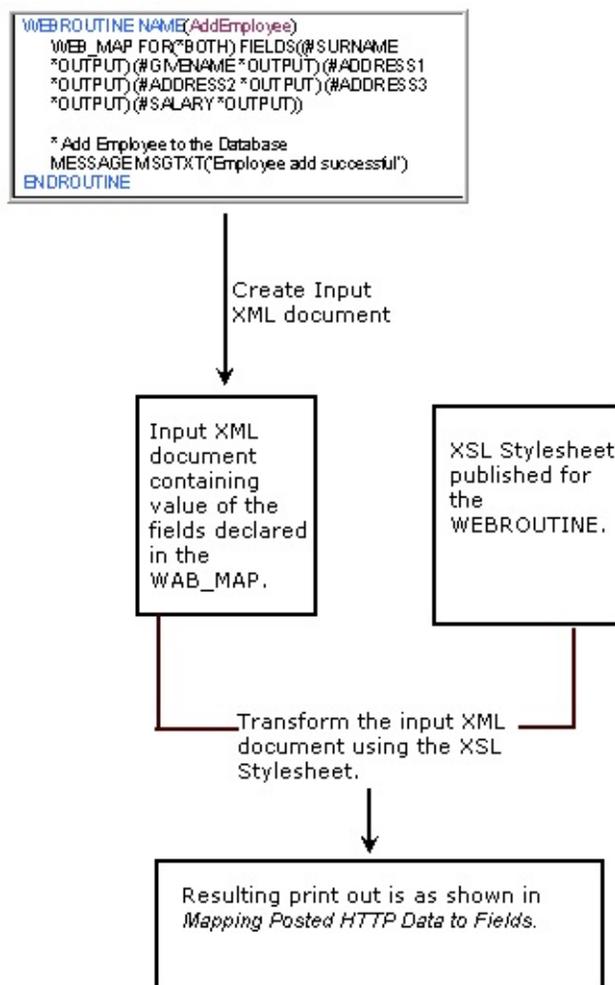
  * Add Employee to the Database
  MESSAGE MSGTXT('Employee add successful')
ENDROUTINE

```

5.7 プレゼンテーション出力の作成方法

ブラウザからWebroutineを呼び出し、(WEB_MAPステートメントに指定されている)受信のフィールドやリストが揃うと、WebroutineのRDMLXコードがENDROUTINEステートメントが検出されるまで実行されます。例えば、RDMLXロジックによって、データベースからデータが抽出されたり、受信のフィールドに基づいて計算が実行されたりします。

Webroutineが完了すると、(WEB_MAPステートメントに指定されている)送信のフィールドやリストがLANSAデータ/アプリケーション・サーバーに送り返されます。その送信のフィールドやリストの値に基づいて、入力用のXMLドキュメントが作成されます。XMLドキュメントの形式については、[付録B. WAMのXMLの構造](#)を参照してください。



指定のテクノロジ・サービスと言語に基づいたWebroutineのXSLスタイ

ルシートを使って、入力用XMLドキュメントは(実行時に)必要なプレゼンテーション形式に変換されます。例えば、省略値のテクノロジー・サービスはHTMLが使用されます。変換後のプレゼンテーション出力(HTML)がWebサーバーからブラウザに送り返されます。

6. WAMとWEBEVENTの相互運用

LANSA V11.0では、Webアプリケーション・モジュール(WAM)という新しい種類のコンポーネントが提供され、Webブラウザやインターネットを基盤とするアプリケーションを作成するためのLANSA機能が大きく拡張されました。

V11.0よりも前のバージョンでは、Webブラウザ・アプリケーションを作成するための主なツールは、WEBEVENTファンクションという特別な種類のRDMLファンクションでした。

WEBEVENTファンクションは引き続きサポートされ、LANSA V11.0で大幅に機能拡張されました。ですから既存のWEBEVENTファンクションをWAMコンポーネントに変換する意味はほとんどというより、全く無いと言えます。ただし、既存のWEBEVENTベースのWebアプリケーションを拡張/強化することで得られるものは多くあります。

ここではWAMコンポーネントおよびWEBEVENTファンクションの両方を含むWebアプリケーションの作成方法について様々なテクニックが紹介されています。

[6.1 WEBEVENTフォームを呼び出すWAMフォーム](#)

[6.2 WAMフォームを呼び出すWEBEVENTフォーム](#)

[6.3 WEBEVENTフォームを管理するWAMコンテナ・フォーム](#)

[6.4 WAMフォームを管理するWEBEVENTコンテナ・フォーム](#)

[6.5 WAMとWEBEVENTファンクション間の情報共有](#)

6.1 WEBEVENTフォームを呼び出すWAMフォーム

WEBEVENTフォームを呼び出すための主な手段は、HandleEvent()というJavaScript関数です。WAMフォームからWEBEVENTフォームを呼び出すためのJavaScriptベースの手段として、それとよく似たHandleWebEvent()という関数も用意されています。

WAMフォームからWEBEVENTフォームを呼び出すには、以下のようにします。

1. 提供されているHandleWebEvent()という名前のJavaScript関数を使用します。

この関数を呼び出す方法は、HandleEvent()関数を呼び出す方法と同じです。

2. ウェブレットには、(Webroutineとは異なり) WEBEVENTに移動する追加のプロパティはありません。HandleWebEvent()は、ほとんどのウェブレットのpresubmit_jsプロパティから、またはユーザー定義のJavaScriptによって呼び出せます。例えば、次のようにonclick属性の値を設定して、JavaScriptを実行するように直接設定することも可能です。
`document.LANSA.SEARCH.onclick =
"HandleWebEvent('PSLSYS', 'Browse', null, null, 'ASURNAME',
'ASTDRENTY')`

3. パラメータは、HandleWebEvent(Process, Webevent, Form, Target, "ASURNAME", "ASTDRENTY", ...)のようになっているので、各フィールドに対応する可変数のパラメータの値をWEBEVENTに渡すことができます。Form以外のすべてのパラメータは文字列であり、Formは、実際のフォームDHTMLオブジェクト(document.MYFORMなど)にします。ここでフィールド名の前に接頭辞を1文字付けることが重要です。この接頭辞は、Alphanumericフィールドの場合はA、Packedフィールドの場合はP、Signedフィールドの場合はS、RDMLXフィールドの場合はQです。この接頭辞は必須で、これにより呼び出し対象のWEBEVENTが渡されるフィールド値を交換できるようになります。

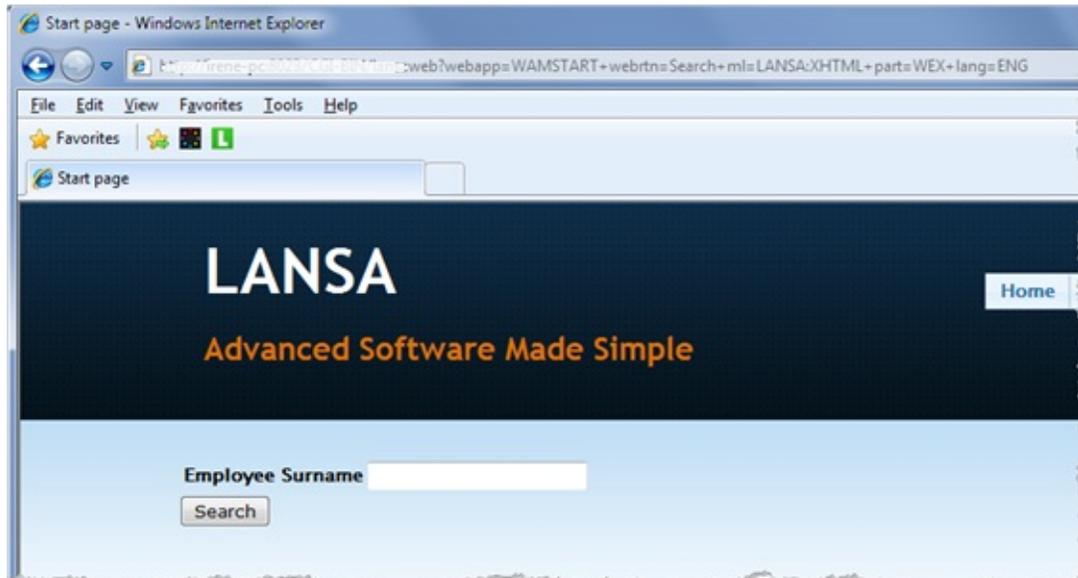
4. このJavaScript関数は、指定のFormパラメータ (Formがnullの場合はデフォルトの"LANSA"フォーム) から各フィールドの値を取得し、一時的なフォームを作成し、URLに送信するその一時的なフォームに各フィールドとそれぞれの値を挿入し、URLに対するHTTP postを実行します。HandleWebEvent()にパラメータとして渡すフィールド名には

すべて、フィールドのタイプを示す1文字の接頭辞を付ける必要があることに注意してください。WAMのフィールド参照では接頭辞を必要としませんが、WEBEVENTファンクションは必要とします。したがって、このJavaScriptコードは、WAMフォームから指定のフィールド値を取得するときには1文字の接頭辞なしで取得しますが、WEBEVENTファンクションにフィールド名を送信するときには接頭辞付きで送信します。

5. その結果、指定のフィールド値が渡されたWEBEVENT LANSAXファンクションが実行され、WEBEVENTページがブラウザに表示されます。

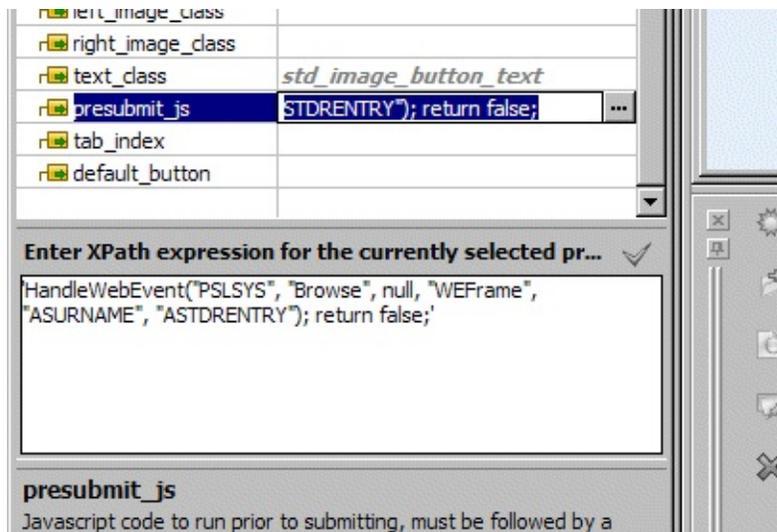
例

WAMフォームからWEBEVENTフォームを開始し、情報を渡す例



WAMフォーム"Search"がWEBEVENT"Browse"を呼び出し、入力されたSURNAMEフィールドの値を渡すとしてします。

LANSAXエディターの[詳細]タブで、検索ボタンのpresubmit_jsプロパティに'HandleWebEvent("PSLSYS", "Browse", null, null, "SURNAME", "STDREENTRY"); return false;'と入力します。



このページをブラウザで実行したときに、検索ボタンをクリックすると、これがWEBEVENTに送信され、WEBEVENTの実行の終了時にブラウザにそのページが表示されます。

6.2 WAMフォームを呼び出すWEBEVENTフォーム

WEBEVENTフォームを呼び出す主な手段は、HandleEvent()というJavaScript関数です。WEBEVENTフォームからWAMフォームを呼び出すためのJavaScriptベースの手段として、それとよく似たHandleWAMEvent()という関数も用意されています。

WEBEVENTフォームからWAMフォームを呼び出すには、以下のようにします。

1. 提供されているHandleWAMEvent()というJavaScript関数を使用します。

この関数を呼び出す方法は、HandleEvent()関数を呼び出す方法と同じです。

2. パラメータは、HandleWAMEvent(WAM, Webroutine, TechServ, Form, Target, actionRequest, Partition, Language, optSessionKey, optDebugMode, "ASURNAME", "ASTDRENTY", ...)のようになっているので、各フィールドに対応する可変数のパラメータの値をWebroutineに渡すことができます。ここでフィールド名の前に接頭辞を1文字付けることが重要です。この接頭辞は、Alphanumericフィールドの場合はA、Packedフィールドの場合はP、Signedフィールドの場合はS、RDMLXフィールドの場合はQです。この接頭辞は、WAMのWEBROUTINEにフィールド値を渡すときには必要ありませんが、フォームのフィールド値にアクセスする時は必須です。これはJavaScriptのHandleWebEvent()関数のセマンティクスと一貫性を持たせるためでもあります。

HandleWAMEventのパラメータは、以下のとおりです。

WAM	ターゲットWAMの名前
Webroutine	ターゲットWEBROUTINEの名前
TechServ	使用するテクノロジー・サービス。省略値のLANSA XHTMLテクノロジー・サービスの場合はnull。
Form	送信フォームのためのフィールド値を取得するフォームHTMLオブジェクト。(例：「MYFORM」という名前のフォームであればdocument.MYFORM) 省略値のLANSAフォームの場合はnull。
Target	ナビゲーションの結果が表示されるターゲットのiframe、frame、ウィンドウ。新しいページを呼び出す

場合はnull。

- actionRequest nullの場合は、デフォルトの"cgi-bin/lansaweb"操作の要求。
- Partition WAMフォームを実行する区画。
- Language WAMを実行する言語。
- optSessionKey SessionKeyMethodがURLの場合、任意でセッション・キーを渡すことが可能。それ以外はnull。
- optDebugMode デバッグURLキーワードを渡して、WAMのデバッグが可能。それ以外はnull。

- このJavaScript関数は、指定のFormパラメータ (またはFormがnullの場合は省略値の"LANSA"フォーム) からフィールドの値を取得し、一時的なフォームを作成して、URLに送信されるこのフォームにフィールドと値を挿入します。そして、URLに対するHTTP postを実行します。HandleWebEvent()にパラメータとして渡すフィールド名にはすべて、フィールドのタイプを示す1文字の接頭辞を付ける必要があることに注意してください。WAMフィールド参照は接頭辞を必要としませんが、WEBEVENTファンクションは必要とします。したがって、このJavaScriptコードは、WEBEVENTフォームから指定のフィールド値を取得するときには1文字の接頭辞付きで取得しますが、WAMファンクションにフィールド名を送信するときには接頭辞なしで送信します。
- その結果、指定のフィールド値が渡されたWebroutineが実行され、Webroutineページがブラウザに表示されます。Webroutineに値を設定するには、送信するフィールドがWEB_MAP FOR(*INPUTまたは*BOTHS)で指定されていなければいけません。

例

WEBEVENTフォームからWAMフォームを開始し、情報を渡す方法

- ファンクションを作成して、以下のコードを貼り付けます。

```
Function Options(*DIRECT *webevent)
*
Define Field(#searchwam) Type(*char) Length(1)
Define Field(#wamname) Type(*char) Length(9)
```

```

Define Field(#webrname) Type(*char) Length(20)
Define Field(#techserv) Type(*char) Length(21)
Define Field(#currlang) Type(*char) Length(4) Default(*language)
*
Group_By Name(#webform) Fields((#stdrentry *hidden) #surname #searchwa
*
Change Field(#wamname) To(<your wam name>)
Change Field(#webrname) To(<your wam webroutine name>)
Change Field(#stdrentry) To(N)
*
Request Fields(#webform) Exit_Key(*no) Menu_Key(*no) Prompt_Key(*no)
*

```

2. <your wam name>と<your wam webroutine name>を適切な名前に置き換えます。

この場合、WAMは、デフォルトのLANSA:XHTMLテクノロジー・サービスと同じ区画に存在していなければならず、同じ言語で実行しなければなりません。そうでなければ、#techserv、#currlang、#partitionの各フィールドの値を変更してください。

3. LANSA Webファンクション・エディターを使用して、InputタイプのVisualコンポーネントを作成し、SEARCHWAMという名前を付けます。
4. コンポーネントのページにもSEARCHWAMという名前を付けます。
5. 新しいページを作成して、以下のコードを貼り付けます。

```

<button onclick="return
HandleWAMEvent('<RDML MERGE="WAMNAME">',
'<RDML MERGE="WEBRNAME">', '<RDML MERGE="TECHSERV">', n

<script type="text/javascript">
//<![CDATA[
function HandleWAMEvent(WAM, WebRoutine, techServ, Form, Target,
actionRequest, Partition, Language, optSessionKey, optDebugMode /*, field1,
field2, etc...*/)
{
  if (Form == null)
  {
    Form = document.LANSA;

```

```

}
if (techServ == null)
{
    techServ = "LANSA:XHTML";
}

var oTempForm = Form.ownerDocument.createElement("form");

if (oTempForm != null)
{
    oTempForm.setAttribute("method", "post");
    Form.ownerDocument.body.appendChild(oTempForm);
    var argLen = arguments.length;

    if (argLen > 10)
    {
        for (var index = 10; index < argLen; index++)
        {
            var fieldNameWithPrefix = arguments[index];
            var fieldName = fieldNameWithPrefix.substr(1,
fieldNameWithPrefix.length - 1);
            for (var ind = fieldNameWithPrefix.length; ind < 10; ind++)
            {
                fieldNameWithPrefix += " ";
            }
            var fieldValue = Form.elements[fieldNameWithPrefix].value;
            InsertHidden(oTempForm, fieldName, fieldValue);
        }
    }

    // Add STDANCHOR if available
    var anchorField = Form.elements["ASTDANCHOR"];
    if (anchorField != null)
    {
        InsertHidden(oTempForm, "STDANCHOR", anchorField.value);
    }

    var prevAction = oTempForm.action;
    var prevTarget = oTempForm.target;

```

```

var action = "";
if (actionRequest == null || actionRequest.length <= 0)
{
    actionRequest = "/cgi-bin/lansaweb";
}
action += actionRequest + "?wam=" + WAM + "&webrtn=" +
WebRoutine + "&ml=" + techServ + "&part=" + Partition + "&lang=" +
Language;
if (optDebugMode != null && optDebugMode.length > 0)
{
    action += "&debug=" + optDebugMode;
}
if (optSessionKey != null)
{
    action += "&sid=" + optSessionKey;
}
oTempForm.action = action;

if (Target != null)
{
    oTempForm.target = Target;
}
oTempForm.submit();
setTimeout(function() {
    oTempForm.action = prevAction;
    oTempForm.target = prevTarget;
    oTempForm.parentNode.removeChild(oTempForm);
}, 100);
}
return false;
}

```

```

function InsertHidden(Form, FieldName, FieldValue)
{
    if (Form == null)
    {
        return;
    }
}

```

```

var field = Form.elements[FieldName];

if (field == null)
{
    var elem = Form.document.createElement("input");

    if (elem != null)
    {
        elem.setAttribute("type", "hidden");
        elem.setAttribute("name", FieldName);
        elem.setAttribute("value", FieldValue);
        Form.appendChild(elem);
    }
}
else
{
    field.value = FieldValue;
}
}
//]]>
</script>

```

6. ページをSEARCHWAMとして保存します。
7. WEBEVENTファンクションをコンパイルして、HTMLを生成します。
8. このWEBEVENTサンプルをブラウザで実行します。検索ボタンをクリックすると、WAMNAMEとWEBRNAMEの各フィールドで指定したWAMとWEBROUTINEが呼び出されます。

6.3 WEBEVENTフォームを管理するWAMコンテナ・フォーム

WEBEVENTフォームを呼び出す主な手段は、HandleEvent()というJavaScript関数です。WAMフォームからWEBEVENTフォームを呼び出すためのJavaScriptベースの手段として、それとよく似たHandleWebEvent()という関数も用意されています。

WAMフォームからWEBEVENTフォームを呼び出すには、以下のようにします。

1. 提供されているHandleWebEvent()という名前のJavaScript関数を使用します。

この関数を呼び出す方法は、HandleEvent()関数を呼び出す方法と同じです。

2. (Webroutineを呼び出す場合とは異なり) ウェブレットにはWEBEVENTを呼び出すための追加のプロパティはありません。HandleWebEvent()は、ほとんどのウェブレットのpresubmit_jsプロパティから、またはユーザー定義のJavaScriptによって呼び出せます。例えば、**onclick**属性の値を設定することでJavascriptを直接実行することが可能です。

```
document.LANSA.SEARCH.onclick =  
"HandleWebEvent('PSLSYS', 'Browse', null, 'WEFrame',  
'ASURNAME', 'ASTDRENTY')"
```

3. パラメータは、HandleWebEvent(Process, Webevent, Form, Target, "ASURNAME", "ASTDRENTY", ...)のようになっているので、各フィールドに対応する可変数のパラメータの値をWEBEVENTに渡すことができます。Form以外のすべてのパラメータは文字列であり、Formは、実際のフォームDHTMLオブジェクト(document.MYFORMなど)にします。ここでフィールド名の前に接頭辞を1文字付けることが重要です。この接頭辞は、Alphanumericフィールドの場合はA、Packedフィールドの場合はP、Signedフィールドの場合はS、RDMLXフィールドの場合はQです。この接頭辞は必須で、これにより呼び出し対象のWEBEVENTが渡されるフィールド値を交換できるようになります。

JavaScript関数のTargetパラメータは必須です。指定できる値は、中に組み込むNavigation panel (std_nav_panel) ウェブレットの名前、ウィン

ドウの名前、独自のHTMLエレメントである<iframe>または<frame>の名前のいずれかです。

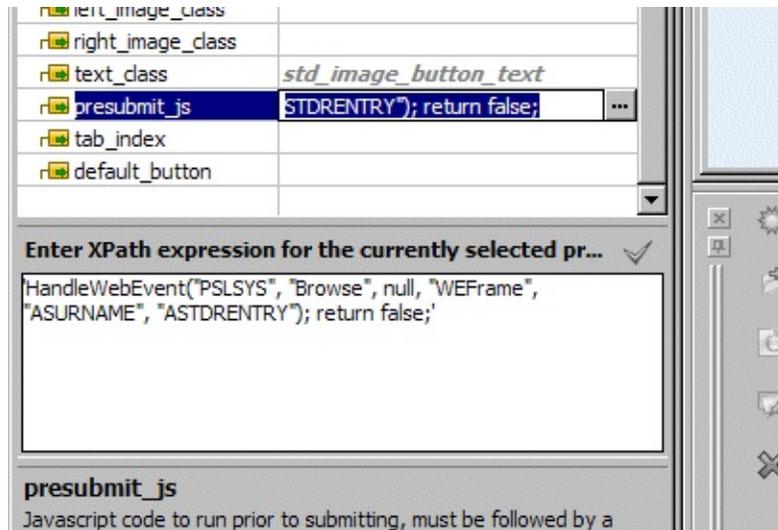
4. このJavaScript関数は、指定のFormパラメータ (またはFormがnullの場合は省略値の"LANSA"フォーム) からフィールドの値を取得し、一時的なフォームを作成して、URLに送信されるこのフォームにフィールドと値を挿入します。そして、URLに対するHTTP postを実行します。HandleWebEvent()にパラメータとして渡すフィールド名にはすべて、フィールドのタイプを示す1文字の接頭辞を付ける必要があることに注意してください。WAMフィールド参照は接頭辞を必要としませんが、WEBEVENTファンクションは必要とします。したがって、このJavaScriptコードは、WAMフォームから指定のフィールド値を取得するときには1文字の接頭辞なしで取得しますが、WEBEVENTファンクションにフィールド名を送信するときには接頭辞付きで送信します。
5. その結果、指定のフィールド値が渡されたWEBEVENT LANSAファンクションが実行され、WEBEVENTページが宛先のNavigation panel (std_nav_panel) ウィンドウに表示されるか、独自のHTMLエレメントである<iframe>または<frame>に表示されます。

例

WAMフォームからWEBEVENTフォームを開始し、情報を渡す方法

このSearch WAMフォームは、Browse WEBEVENTを呼び出し、入力されたSURNAMEフィールドの値を渡し、検索ボタンの下のNavigation panel (std_nav_panel) に結果を表示するとします。

1. LANSAエディターで、下のようにNavigation panel (std_nav_panel) ウェブレットをページにドラッグ・アンド・ドロップし、nameプロパティに'WEFrame'と入力します。size_panel_to_contentプロパティでは"yes"を選択します。これで、呼び出し対象のWEBEVENTページのサイズいっぱいになるようサイズが調整されます。
2. 検索ボタンをクリックして、検索ボタンのpresubmit_jsプロパティに次を入力します。'HandleWebEvent("PSLSYS", "Browse", null, "WEFrame", "ASURNAME", "ASTDRENTY"); return false;'



3. このページをブラウザで実行して検索ボタンをクリックすると、WEBEVENTが呼び出され、同じページのNavigation panel (std_nav_panel) ウェブレットにこのWEBEVENTページが表示されます。

6.4 WAMフォームを管理するWEBEVENTコンテナ・フォーム

WEBEVENTフォームを呼び出す主な手段は、HandleEvent()というJavaScript関数です。WEBEVENTフォームからWAMフォームを呼び出すためのJavaScriptベースの手段として、それとよく似たHandleWAMEvent()という関数も用意されています。

WEBEVENTフォームからWAMフォームを呼び出すには、以下のようにします。

1. 提供されているHandleWAMEvent()というJavaScript関数を使用します。
この関数を呼び出す方法は、HandleEvent()関数を呼び出す方法と同じです。
2. パラメータは、HandleWAMEvent(WAM, Webroutine, TechServ, Form, Target, actionRequest, Partition, Language, optSessionKey, optDebugMode, "ASURNAME", "ASTDREENTRY", ...)のようになっているいて、各フィールドに対応する可変数のパラメータの値をWebroutineに渡すことができます。ここでフィールド名の前に接頭辞を1文字付けることが重要です。この接頭辞は、Alphanumericフィールドの場合はA、Packedフィールドの場合はP、Signedフィールドの場合はS、RDMLXフィールドの場合はQです。この接頭辞は、WAMのWEBROUTINEにフィールド値を渡すときには必要ありませんが、フォームのフィールド値にアクセスする時は必須です。これはJavaScriptのHandleWebEvent()関数のセマンティックスと一貫性を持たせるためでもあります。
3. HandleWAMEventのパラメータは、以下のとおりです。

WAM	ターゲットWAMの名前
Webroutine	ターゲットWEBROUTINEの名前
TechServ	使用するテクノロジー・サービス。省略値のLANSA XHTMLテクノロジー・サービスの場合はnull。
Form	送信フォームのためのフィールド値を取得するフォームHTMLオブジェクト。(例：「MYFORM」という名前のフォームであればdocument.MYFORM) 省略値のLANSAフォームの場合はnull。
Target	ナビゲーションの結果が表示されるターゲットの

iframe、frame、ウィンドウ。新しいページを呼び出す場合はnull。

- actionRequest nullの場合は、デフォルトの"cgi-bin/lansaweb"操作の要求。
- Partition WAMフォームを実行する区画。
- Language WAMを実行する言語。
- optSessionKey SessionKeyMethodがURLの場合、任意でセッション・キーを渡すことが可能。それ以外はnull。
- optDebugMode デバッグURLキーワードを渡して、WAMのデバッグが可能。それ以外はnull。

- このJavaScript関数は、指定のFormパラメータ (またはFormがnullの場合は省略値の"LANSA"フォーム) からフィールドの値を取得し、一時的なフォームを作成して、URLに送信されるこのフォームにフィールドと値を挿入します。そして、URLに対するHTTP postを実行します。HandleWebEvent()にパラメータとして渡すフィールド名にはすべて、フィールドのタイプを示す1文字の接頭辞を付ける必要があることに注意してください。WAMフィールド参照は接頭辞を必要としませんが、WEBEVENTファンクションは必要とします。したがって、このJavaScriptコードは、WEBEVENTフォームから指定のフィールド値を取得するときには1文字の接頭辞付きで取得しますが、WAMファンクションにフィールド名を送信するときには接頭辞なしで送信します。
- JavaScript関数のTargetパラメータは必須です。指定できる値は、中に組み込むiframeの名前、frameの名前、ウィンドウの名前のいずれかです。
- その結果、指定のフィールド値が渡されたWebroutineが実行され、Webroutineページがブラウザーに表示されます。Webroutineに値を設定するには、送信するフィールドがWEB_MAP FOR(*INPUTまたは*BOTH)で指定されていなければいけません。

例

WEBEVENTフォームからWAMフォームを開始し、情報を渡す方法

- ファンクションを作成して、以下のコードを貼り付けます。

Function Options(*DIRECT *webevent)

*

Define Field(#searchwam) Type(*char) Length(1)

Define Field(#wamname) Type(*char) Length(9)

Define Field(#webrname) Type(*char) Length(20)

Define Field(#techserv) Type(*char) Length(21)

Define Field(#frametgt) Type(*char) Length(20)

Define Field(#currlang) Type(*char) Length(4) Default(*language)

*

Group_By Name(#webform) Fields((#stdrentry *hidden) (#frametgt *noid) #s

*

Change Field(#wamname) To(<your wam name>)

Change Field(#webrname) To(<your wam webroutine name>)

Change Field(#frametgt) To(<your iframe name>)

Change Field(#stdrentry) To(N)

*

Request Fields(#webform) Exit_Key(*no) Menu_Key(*no) Prompt_Key(*no)

*

2. <your wam name>、 <your wam name>、 <your iframe name>をそれぞれ適切な名前に置き換えます。

この場合、WAMは、デフォルトのLANSA:XHTMLテクノロジー・サービスと同じ区画に存在していなければならず、同じ言語で実行しなければなりません。そうでなければ、#techserv、#currlang、#partitionの各フィールドの値を変更してください。

3. LANSА Webエディターを使用して、InputタイプのVisualコンポーネントを作成し、FRAMETGTという名前を付けます。
4. コンポーネントのページにもFRAMETGTという名前を付けます。
5. 新しいページを作成して、以下のコードを貼り付けます。

```
<iframe style="width:600px; height:400px" name='<RDML MERGE="FRAM
</iframe>
```

6. ページをFRAMETGTとして保存します。
7. LANSА Webファンクション・エディターを使用して、InputタイプのVisualコンポーネントを作成し、SEARCHWAMという名前を付けま

す。

8. コンポーネントのページにもSEARCHWAMという名前を付けます。
9. 新しいページを作成して、以下のコードを貼り付けます。

```
<button onclick="return  
HandleWAMEvent('<RDML MERGE="WAMNAME">',  
'<RDML MERGE="WEBRNAME">', '<RDML MERGE="TECHSERV">', m
```

```
<script type="text/javascript">  
//<![CDATA[  
function HandleWAMEvent(WAM, WebRoutine, techServ, Form, Target,  
actionRequest, Partition, Language, optSessionKey, optDebugMode /*, field1,  
field2, etc...*/) 
```

```
{  
    if (Form == null)  
    {  
        Form = document.LANSA;  
    }  
    if (techServ == null)  
    {  
        techServ = "LANSA:XHTML";  
    }  
}
```

```
var oTempForm = Form.ownerDocument.createElement("form");
```

```
if (oTempForm != null)  
{  
    oTempForm.setAttribute("method", "post");  
    Form.ownerDocument.body.appendChild(oTempForm);  
    var argLen = arguments.length;
```

```
    if (argLen > 10)  
    {  
        for (var index = 10; index < argLen; index++)  
        {  
            var fieldNameWithPrefix = arguments[index];  
            var fieldName = fieldNameWithPrefix.substr(1,  
fieldNameWithPrefix.length - 1);
```

```

    for (var ind = fieldNameWithPrefix.length; ind < 10; ind++)
    {
        fieldNameWithPrefix += " ";
    }
    var fieldValue = Form.elements[fieldNameWithPrefix].value;
    InsertHidden(oTempForm, fieldName, fieldValue);
}
}

// Add STDANCHOR if available
var anchorField = Form.elements["ASTDANCHOR"];
if (anchorField != null)
{
    InsertHidden(oTempForm, "STDANCHOR", anchorField.value);
}

var prevAction = oTempForm.action;
var prevTarget = oTempForm.target;

var action = "";
if (actionRequest == null || actionRequest.length <= 0)
{
    actionRequest = "/cgi-bin/lansaweb";
}
action += actionRequest + "?wam=" + WAM + "&webrtn=" +
WebRoutine + "&ml=" + techServ + "&part=" + Partition + "&lang=" +
Language;
if (optDebugMode != null && optDebugMode.length > 0)
{
    action += "&debug=" + optDebugMode;
}
if (optSessionKey != null)
{
    action += "&sid=" + optSessionKey;
}
oTempForm.action = action;

if (Target != null)
{

```

```

    oTempForm.target = Target;
  }
  oTempForm.submit();
  setTimeout(function() {
    oTempForm.action = prevAction;
    oTempForm.target = prevTarget;
    oTempForm.parentNode.removeChild(oTempForm);
  }, 100);
}
return false;
}

```

```
function InsertHidden(Form, FieldName, FieldValue)
```

```

{
  if (Form == null)
  {
    return;
  }

  var field = Form.elements[FieldName];

  if (field == null)
  {
    var elem = Form.document.createElement("input");

    if (elem != null)
    {
      elem.setAttribute("type", "hidden");
      elem.setAttribute("name", FieldName);
      elem.setAttribute("value", FieldValue);
      Form.appendChild(elem);
    }
  }
  else
  {
    field.value = FieldValue;
  }
}
//]]>

```

</script>

10. ページをSEARCHWAMとして保存します。
11. WEBEVENTファンクションをコンパイルして、HTMLを生成します。
12. 上のWEBEVENTサンプルをブラウザで実行します。検索ボタンをクリックすると、WAMNAMEとWEBRNAMEの各フィールドで指定したWAMとWEBROUTINEが呼び出されます。WEBROUTINEの実行によって生成されるHTML応答が、WEBEVENTの検索ボタンと同じページにあるFRAMETGTコンポーネントに表示されます。

6.5 WAMとWEBEVENTファンクション間の情報共有

既存のWEBEVENTファンクションの間では、表示されない情報またはサーバー側の情報を共有する場合があります。これはWeb対話処理時にクライアントのブラウザで非表示のブラウズ・リストやフィールドを使って引き渡されることが通常です。この手法は、シンプルなセッション状態管理の一つです。

この手法は、WEBEVENTファンクション同士の情報共有には使用できませんが、WAMとWEBEVENTファンクション間の情報共有には使用できません。

同じWebセッションの中で実行するWEBEVENTとWAMアプリケーション間で情報を共有できる新しい手法があります。この手法は、わかりやすく、汎用的であり、機能的にも優れています。以下の項目を参照してください。

6.5.1 共有データの一意の識別

6.5.2 データの共有

6.5.3 共有データのクリーンアップ

6.5.4 Visual LANSAフレームワークと「仮想クリップボード」

6.5.1 共有データの一意的識別

ステートレスなHTTPプロトコル経由のブラウザとサーバー間の対話処理では、一意の値の受け渡しが可能です。その値は、サーバー上で管理される、ブラウザに渡されずに表示されることがないデータを識別するための特別なキーの役割を果たします。データベースで言えば、1行または複数行のデータを識別することによって、他のデータ・テーブルに対する外部キーとしての役割を果たせるキーのようなものです。

WAMにも、同じような手法があります。一意の識別子を使用してセッションを識別し、サーバー上のセッション状態へのアクセスを認めるかどうかを決定できるのです。

WEBEVENTファンクションとWAMの間でデータを共有するためにも、同じような手法を活用できます。そのために必要なのは、WEBEVENTとWAMの間でブラウザの対話処理を実行するとき、ブラウザとサーバーの間で特別な一意識別キーの受け渡しを行うことだけです。ブラウザでは単にフォームの非表示フィールドにその値を格納するだけで、フォームと一緒にその値を送信できます。

その識別用の値は、固有のサーバー・データ・セットを一意的に識別するために、32バイトの長さにするのを強くお勧めします。WAMフォームの場合は、デフォルトで、STDANCHORフィールドをこの目的のために管理/送信するようになっています。WEBEVENTフォームの場合は、DISPLAY/REQUESTコマンドにそのフィールドを追加して、WEBEVENTフォームに配置しなければなりません。32バイトの長さの英数字フィールドとしてそのフィールドをリポジトリに作成すれば、WAMコードとWEBEVENTコードの両方でそのフィールドを使用して、共有するデータを取得できます。このフィールドは、共有データを格納するデータベース・テーブルのキーとしても使用できますし、オペレーティング・システム・ファイルのキーとして、共有データを格納するファイルの名前として使用することも可能です。

6.5.2 データの共有

一例として、ショッピング・カートを共有する必要があるアプリケーションについて考えてみましょう。ショッピング・カートは、リストという形で表現できます。ショッピング・カートには、商品のID、商品名、数量が含まれます。顧客がショッピング・カートのリストに複数の商品を追加するWEBEVENTアプリケーションがあるとします。そのWEBEVENTのプログラムでは、STDANCHORに一意的識別子を割り当ててから、STDANCHORをキーとしてショッピング・カートのリスト項目をデータベース・テーブルに挿入できます。STDANCHORをDISPLAYコマンドに追加して、その値をフォームの非表示フィールドとして利用できるようにします。

WEBEVENTフォームをブラウザで表示すると、そのフォームには、STDANCHORの一意的識別子も組み込まれています。「[6. WAMとWEBEVENTの相互運用](#)」で説明されているように、JavaScriptのHandleWAMEvent()関数を使用すれば、WAMを呼び出すことが可能です。

HandleWAMEvent()は、フォームの中のSTDANCHORフィールドを自動的に探して、その値も一緒に送信します。

WAM側では、STDANCHORフィールドをWEB_MAPに記述して、WebRoutineにマッピングする必要があります。ターゲットWebRoutineが実行されると、WEB_MAPでWebRoutineにマップされているSTDANCHOR識別子の値やその他のフィールド値がHandleWAMEvent()経由で渡され、これをこのWebRoutineが受け取ります。このWebRoutineでは、STDANCHORキーを使用して、ショッピング・カートのリストをデータベース・テーブルから取得できます。さらに、WAMのセッション状態に属するWAMリストにそれらの値を置き、STDANCHORキーで識別されるデータをデータベースから削除できます。あるいは、データベースから古いデータを削除する特別なクリーンアップ・ジョブの実行スケジュールを設定することも可能です。

WAMからWEBEVENTにデータを渡すには、上の手順を逆の順序で実行します。JavaScriptのHandleWebEvent()関数が、WAMフォームの中のSTDANCHORフィールドを自動的に探して、その値を他の送信データと一緒に送信します。

6.5.3 共有データのクリーンアップ

「6.5.2 データの共有」に書かれているように、WAMとWEBEVENTファンクションの間で共有データを保持するには、その両方からアクセスできるデータベースやファイルなどの保管場所にデータを格納しなければなりません。ブラウザ・ベースのアプリケーションの性質上、ユーザーがアプリケーションの使用をいつ終了したのかは分かりません。ですから、共有データの適切なクリーンアップ処理をいつ実行したらよいのかも判別できません。

このため、サーバー上に一時データを保持する必要があるすべてのブラウザ・アプリケーションでは、データを古いものと見なしてクリーンアップを実行するタイミングを見定めるためのタイムアウト・メカニズムが必要です。

「6.5.2 データの共有」で取り上げたSTDANCHORのメカニズムを使用してデータを共有する場合は、共有データの格納/更新時にデータにタイムスタンプを設定することをお勧めします。クリーンアップ・ジョブは、そのタイムスタンプに基づいてデータの最終使用時刻を見て、クリーンアップ処理を実行するかどうかを決定できます。

アプリケーションに「ログアウト」のような機能がある場合は、その時点でも共有データのクリーンアップを実行することをお勧めします。ただし、ユーザーが必ず「ログアウト」を実行するとは限らないので、古い共有データを削除するためのクリーンアップ・メカニズムは必要です。

6.5.4 Visual LANSAフレームワークと「仮想クリップボード」

Web対応のVisual LANSAフレームワークでは、セッション・データを共有するための"仮想クリップボード"機能をサポートしています。詳細は『*Visual LANSA* フレームワークガイド』の「[仮想クリップボード](#)」を参照してください。

7. テクノロジ・サービス

WAMはXSL技術を使用してXSLスタイルシートを生成し、実行時にはブラウザ(あるいは他のエージェント)に送信されるプレゼンテーションの生成を行います。

XSL技術により変換が可能になり、別のXSLスタイルシートの出力ができるようになります。これが生成時にテクノロジ・サービスXSLスタイルシートを使用する目的です。

テクノロジ・サービスXSLスタイルシートの入力となるXMLは、Webroutineで使用するフィールドとリストの情報を含むXMLファイルです。この情報とは、フィールドの長さ、フィールドのタイプ、属性などです。これらの情報はテクノロジ・サービスXSLスタイルシートで使用され、Web_Mapで使用されるフィールドやリストを反映させた適切なWebroutineスタイルシートを生成します。

LANSAでは、XHTML、jQueryモバイルおよびPocket PC HTMLのためのテクノロジ・サービスが用意されています。ただし、次のような理由により、独自のテクノロジ・サービスを作成しなければならない場合があります。

- 既存の他のXMLマークアップ言語(VoiceXMLなど)もしくは、新しい形のXMLマークアップ言語を導入するため。
- カスタムXML形式を使用するサプライヤや顧客に対応するため。
- 特定のユーザー・エージェント(Microsoft Internet Explorer固有の拡張機能のあるHTMLなど)に合わせてカスタマイズするため。

テクノロジ・サービスの作成については以下を参照してください。

- [7.1 テクノロジ・サービスの作成](#)
- [7.2 TSMLドキュメントの構造](#)
- [7.3 TSMLドキュメントの例](#)
- [7.4 WebRoutineの TSP スタイル・シートと LANSA エディター](#)
- [7.5 テクノロジ・サービスのデフォルト・ウェブレット](#)
- [7.6 ウェブレットとウェブレット・テンプレート](#)

7.1 テクノロジ・サービスの作成

新しいテクノロジ・サービスを作成するには、以下の手順に従ってください。

[ステップ1. テクノロジ・サービスの作成](#)

[ステップ2. テクノロジ・サービスのXSLスタイルシートの作成](#)

[ステップ2a. Webroutine TSPスタイルシートの作成](#)

[ステップ2b. ウェブレットTSPスタイルシートの作成](#)

[ステップ2c. テクノロジ・サービス・スタイルシートをTSPディレクトリにコピー](#)

ステップ1. テクノロジ・サービスの作成

LANSAエディターを使用してテクノロジ・サービスを作成します。プロバイダーの名前とテクノロジ・サービスの名前によって、テクノロジ・サービスは一意に識別されます。

新しいテクノロジ・サービスを作成する際は、プロパティを定義します。プロパティでは、LANSAエディターやWAM実行時環境が使用する定義やオプションを保管します。

以下も参照してください。

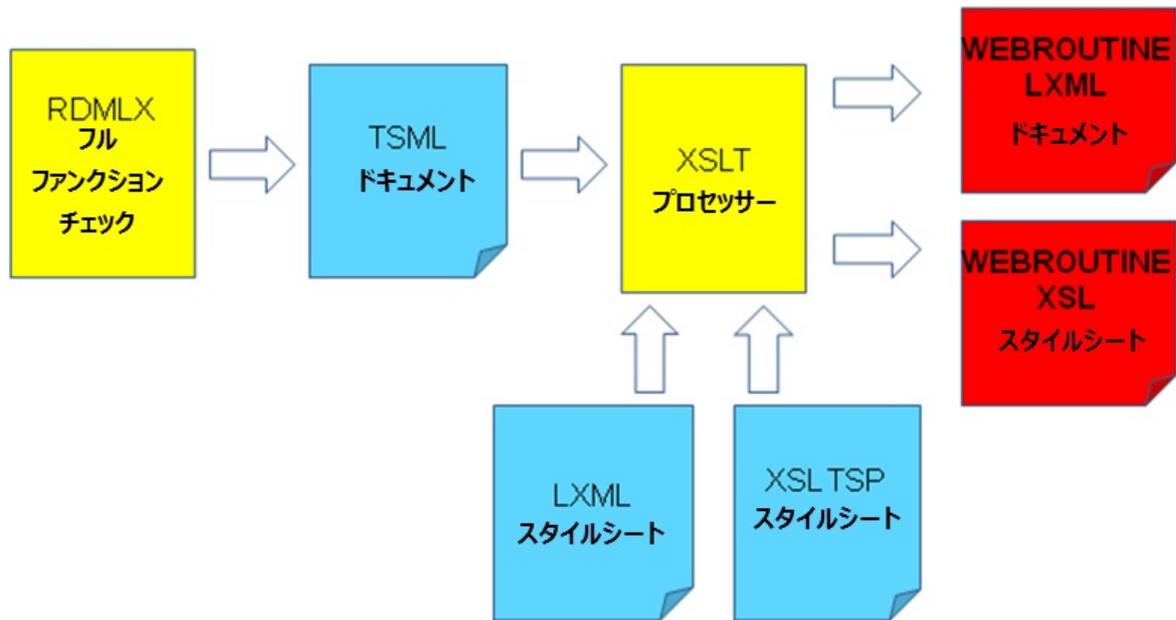
『[Visual LANSACユーザーガイド](#)』の「[プロバイダ定義の編集](#)」

『[LANSA テクニカルリファレンスガイド](#)』の「[テクノロジ・サービス](#)」

ステップ2. テクノロジ・サービスのXSLスタイルシートの作成

WAMでXSLスタイルシートを使ってWebroutine XMLドキュメントを異なるプレゼンテーション・フォーマットに変換する方法やテクノロジ・サービスの目的については、このガイドの前半で説明されています。詳しくは、「[WAMの構造](#)」を参照してください。

LANSAはXSLスタイルシートを使用して、Webroutine XMLドキュメントと、テクノロジ・サービスのためのWebroutine XSLスタイルシートを生成します。



WebroutineのためのXSLを生成する際、LANSAはテクノロジー・サービス・マークアップ言語（TSML Technology Service Markup Language）ドキュメントと呼ばれるメモリー内のXMLドキュメントを生成します。これは、Webroutine LXMLドキュメントとWebroutine XSLスタイルシートの両方を作成するために使用される、入力用ドキュメントです。WAMレイアウト・ウェブレットにも同様の処理が行われます。

テクノロジー・サービスを作成する時には、次の2つのTSPスタイルシートを用意する必要があります。

- Webroutine XSLスタイルシート生成するためのもの。
- WAMレイアウト・ウェブレットを生成するためのもの。

TSPスタイル・シートはUTF-8でエンコードされていなければなりません。つまり、これにはステートメントの `<?xml version="1.0" encoding="UTF-8" ?>` が含まれている必要があります。

XSLWebroutineドキュメントの変換に使用されるXSLは標準W3CのXSL 1.0仕様に準拠しています。詳細についてはXSL 1.0を参照してください。

グローバルLXMLスタイル・シートは、テクノロジー・サービスの種類にかかわらず共通です。独自に作成する必要はありません。

テクノロジー・サービス・マークアップ言語ドキュメントについての情報は以下を参照してください。

7.2 TSMLドキュメントの構造

7.3 TSMLドキュメントの例

ステップ2a. Webroutine TSPスタイルシートの作成

ステップ2の図に示されている通り、Webroutine XSLスタイルシートの作成には、Webroutine TSPスタイルシートを使用します。

命名規則は以下の通りです。(すべて小文字にします)。

tsp_<provider>_<technology_service_name>_WebRoutine.xml

上記の <provider> はテクノロジー・サービス・プロバイダ、<technology_service_name> はテクノロジー・サービスの名前です。例えば、LANSA:XHTMLの場合、Webroutine TSPスタイルシートの名前は、以下のようになります。

tsp_lansa_xhtml_WebRoutine.xml

LANSAから提供されているTSPスタイルシートをベースにするのが、Webroutine TSPスタイルシートを作成する最も簡単な方法です。

提供されているWebroutine TSPスタイルシートには2つのトップレベルのパラメータ (g_inliner_call と g_import_path) があります。これらを使用して、インライン・リストをサポートします。ジェネレーターがインライン・ウェブレットを挿入する必要がある場合にインライン呼び出しが行われます。

ステップ2b. ウェブレットTSPスタイルシートの作成

WAMレイアウト・ウェブレットの作成には、ウェブレットTSPスタイルシートを使用します。WAMの作成時に、LANSAは、そのWAMにレイアウト・ウェブレットがあるかどうかを確認します。ない場合は、このTSPスタイルシートを使用して、レイアウト・ウェブレットを作成します。

命名規則は以下の通りです。(すべて小文字にします)。

tsp_<provider>_<technology_service_name>_webletbuilder.xml

上記の <provider> はテクノロジー・サービス・プロバイダ、<technology_service_name> はテクノロジー・サービスの名前です。例えば、LANSA:XHTMLの場合、Webroutine TSPスタイルシートの名前は、以下のようになります。

tsp_lansa_xhtml_webletbuilder.xml

LANSAから提供されているTSPスタイルシートをベースにするのが、Webroutine TSPスタイルシートを作成する最も簡単な方法です。

注: 提供されるウェブレットのTSPスタイルシートには、他のウェブレットを作成するテンプレートがあります。現時点で、TSPスタイルシートに導入する必要があるウェブレットは、レイアウト・ウェブレットだけです。

ステップ2c. テクノロジ・サービス・スタイルシートをTSPディレクトリにコピー

すべてのTSPスタイルシートは、TSPディレクトリに置く必要があります。

... <lansa root>\x_WIN95\X_LANSA\web\tsp

iSeriesとUnix/Linuxの場合:... <lansa root>/x_lansa/web/tsp

7.2 TSMLドキュメントの構造

テクノロジー・サービス・マークアップ言語 (TSML) ドキュメントは、Visual LANSAにより作成されます。TSMLは、Webroutineでマップされる (つまりWEB_MAPで定義された) フィールドおよびリストを記述します。また、WAMからのコンテキスト情報を持ち、既存のLXMLドキュメント (これは後で置換されます) からの値を保存して、新しく生成されるXML/XSLに残るようにします。構造はLXMLドキュメント (LANSAエディタの[Webデザイン]の[XML]タブで見ることができます) に非常に似ていますが、XSLスタイルシートを生成するための追加のメタ・データを持っています。

このTSMLドキュメントは、以下のセクションに分かれています。

テクノロジー・サービス・リスト

生成時に使用され、指定のテクノロジー・サービス用に存在するXSLの数を決定します。1つのXSLだけ (省略値の言語用) でWebroutine XSLが再生成される場合は、置換されるTSMLノードはドキュメントから安全に削除することができます。

サーバー命令

これらの命令は、LXMLドキュメントに直接マップされています。WAM実行時環境は、これらのエレメントを使用してHTTP応答を用意します。

ウェブレット・セクション

Webroutineの中で参照されているtsml:fieldとtsml:listの列によるビジュアルライゼーションに使用されるウェブレットをリストにします。このウェブレットは各テクノロジー・サービスが使用するテンプレート呼び出しのパラメータをリストにします。

LXMLデータ・セクション

XSLスタイルシート内のLXMLデータ・アイランドにマップするコンテンツをリストにします。例えば、ドロップダウンのためのピックリスト項目などです。

置換されたLXMLセクション

現在のLXMLドキュメント (再生成によって置き換えられる) 内の、残しておく必要のある値を持つ箇所を含みます。現時点では、クッキー、サンプル・メッセージ、フィールド・キャプションとサンプル値、リスト・キャプションとサンプル値、TSMLデータ・アイランドが含まれて

います。

コンテキスト・セクション

TSMLドキュメントのこのセクションには、Webroutineに関するコンテキスト情報が含まれています。WAMの名前、Webroutineの名前、Webroutineのタイトルなどの項目です。注：すべてのコンテキスト項目がlxml:itemsコンテキストにマップされるわけではありません。生成時にのみ使用される項目もあります。例えば、tsml:layout-name項目では、XSLスタイルシートにインポートするレイアウト・ウェブレットの名前を指定します。

オプション・セクション

オプション・セクションには各種のWebroutineオプションが含まれます。このオプションは変更可能で、特定の検証機能やプレゼンテーション機能を有効にするかなどを決定します。

メッセージ・セクション

LXMLドキュメントのメッセージ・セクションにマップされます。

フィールド・セクション

フィールド・セクションには、WebroutineのWEB_MAPステートメントで送信フィールドとして指定されているフィールドが含まれています。これらは、LXMLドキュメントのフィールド・リストに記述されているフィールドです。キャプションや値の要素に加え、tsml:fieldには、表示サイズ、入力時の大文字/小文字の指定、ウェブレットのビジュアライゼーション(該当する場合)などのメタデータ・コンテンツも含まれます。

リスト・セクション

リスト・セクションには、WebroutineのWEB_MAPステートメントで送信のリストとして指定されているリストが含まれています。これらは、LXMLドキュメントにあるリストです。tsml:columnには、列のキャプションや値の要素の他に、tsml:field要素といったメタデータ・コンテンツも含まれます。

LANSAエディターでは、Webroutine TSPスタイルシートを使用して、フィールドやリスト用のXSLを作成します。独自のテクノロジー・サービスをLANSAエディターで編集できる場合は、フィールドやリストを作成するためのテンプレートをLANSAに用意されているTSPスタイルシートで使用されているパターンに合わせる必要があります。

7.3 TSML ドキュメントの例

TSML ドキュメントの例を以下に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<tsml:data full-
document="true" inline="none" xmlns:tsml="http://www.lansa.com/2002/XMI
Metadata">
<tsml:technology-service-list>
<tsml:technology-service used_by="LANSA_XHTML" lang-count="1" />

<tsml:server-instructions>
<tsml:client-charset />
<tsml:cookies />
<tsml:ssi />
</tsml:server-instructions>

<tsml:replaced-lxml xmlns:lxml="http://www.lansa.com/2002/XML/Runtime-
Data" />

<tsml:weblets>
<tsml:weblet name="std_dropdown.std_dropdown">
<tsml:technology-services>
<tsml:technology-service name="LANSA:XHTML" mod-
id="20120116205618000">
<tsml:template-params>
<tsml:template-param>
<tsml:param-name>display_mode</tsml:param-name>
<tsml:param-role>std:display_mode</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>items</tsml:param-name>
<tsml:param-role>std:picklist</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>pos_absolute</tsml:param-name>
<tsml:param-role>std:pos_absolute_design</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
```

```
<tsml:param-name>width_design</tsml:param-name>
<tsml:param-role>std:width_design</tsml:param-role>
</tsml:template-param>
</tsml:template-params>
</tsml:technology-service>
</tsml:technology-services>
</tsml:weblet>
</tsml:weblets>
```

```
<tsml:lxml-data xmlns:lxml="http://www.lansa.com/2002/XML/Runtime-
Data">
<lxml:picklist id="9C3BBEF5861148FE8B36378F5F06EF26" field-
ref="GRADEX">
<lxml:item>
<lxml:caption>
<lxml:variable name="MTXTGRADED" />
</lxml:caption>
<lxml:value>D</lxml:value>
</lxml:item>
<lxml:item>
<lxml:caption>
<lxml:variable name="MTXTGRADEM" />
</lxml:caption>
<lxml:value>M</lxml:value>
</lxml:item>
<lxml:item>
<lxml:caption>
<lxml:variable name="MTXTGRADEP" />
</lxml:caption>
<lxml:value>P</lxml:value>
</lxml:item>
<lxml:item>
<lxml:caption>
<lxml:variable name="MTXTGRADEF" />
</lxml:caption>
<lxml:value>F</lxml:value>
</lxml:item>
</lxml:picklist>
</tsml:lxml-data>
```

```
<tsml:context>
<tsml:user-id>QOTHPRDOWN</tsml:user-id>
<tsml:webapplication>EMPWAM</tsml:webapplication>
<tsml:webapplication-title>Employee</tsml:webapplication-title>
<tsml:webroutine>skills</tsml:webroutine>
<tsml:webroutine-title>Employee skills</tsml:webroutine-title>
<tsml:service-name />
<tsml:partition>WEX</tsml:partition>
<tsml:language iso-lang="en">ENG</tsml:language>
<tsml:images-path>/images</tsml:images-path>
<tsml:action-request>/CGI-BIN/lansaweb</tsml:action-request>
<tsml:layout-name>empwam_layout</tsml:layout-name>
<tsml:timestamp>2012-03-07T10:30:00+10:00</tsml:timestamp>
</tsml:context>
```

```
<tsml:options>
<tsml:option name="DBCS">>false</tsml:option>
<tsml:option name="align-right">>true</tsml:option>
<tsml:option name="check-numeric">>false</tsml:option>
<tsml:option name="debug" />
<tsml:option name="trace" />
<tsml:option name="task" />
</tsml:options>
```

```
<tsml:variables />
```

```
<tsml:messages />
```

```
<tsml:fields>
<tsml:field name="EMPNO">
<tsml:mode>input</tsml:mode>
<tsml:sample-value>ABCDE</tsml:sample-value>
<tsml:format>
<tsml:type>alpha</tsml:type>
<tsml:display-max-length>5</tsml:display-max-length>
<tsml:max-length>5</tsml:max-length>
<tsml:input-case>uppercase</tsml:input-case>
<tsml:keyboardshift />
```

```
</tsml:format>
<tsml:caption ref="description">
<tsml:label>Employee no....</tsml:label>
<tsml:description>Employee Number</tsml:description>
<tsml:heading-1> Employ</tsml:heading-1>
<tsml:heading-2> Number</tsml:heading-2>
<tsml:heading-3 />
</tsml:caption>
</tsml:field>
<tsml:field name="GIVENAME">
<tsml:mode>input</tsml:mode>
<tsml:sample-value>ABCDEFGHIJKLMNQRST</tsml:sample-value>
<tsml:format>
<tsml:type>alpha</tsml:type>
<tsml:display-max-length>20</tsml:display-max-length>
<tsml:max-length>20</tsml:max-length>
<tsml:input-case>uppercase</tsml:input-case>
<tsml:keyboardshift>O</tsml:keyboardshift>
</tsml:format>
<tsml:caption ref="description">
<tsml:label>Given names....</tsml:label>
<tsml:description>Employee Given Name(s)</tsml:description>
<tsml:heading-1>Given name(s)</tsml:heading-1>
<tsml:heading-2 />
<tsml:heading-3 />
</tsml:caption>
</tsml:field>
<tsml:field name="SURNAME">
<tsml:mode>input</tsml:mode>
<tsml:sample-value>ABCDEFGHIJKLMNQRST</tsml:sample-value>
<tsml:format>
<tsml:type>alpha</tsml:type>
<tsml:display-max-length>20</tsml:display-max-length>
<tsml:max-length>20</tsml:max-length>
<tsml:input-case>uppercase</tsml:input-case>
<tsml:keyboardshift>O</tsml:keyboardshift>
</tsml:format>
<tsml:caption ref="description">
<tsml:label>Surname.....</tsml:label>
```

```
<tsml:description>Employee Surname</tsml:description>
<tsml:heading-1>Surname</tsml:heading-1>
<tsml:heading-2 />
<tsml:heading-3 />
</tsml:caption>
</tsml:field>
</tsml:fields>
```

```
<tsml:lists default-sample-size="5">
<tsml:list name="SKILLS" inline="false">
<tsml:mode>input</tsml:mode>
<tsml:list-header>
<tsml:header name="SKILCODE">
<tsml:heading-1>Skill</tsml:heading-1>
<tsml:heading-2>Code</tsml:heading-2>
<tsml:heading-3 />
</tsml:header>
<tsml:header name="SKILDESC">
<tsml:heading-1>Skill</tsml:heading-1>
<tsml:heading-2>Description</tsml:heading-2>
<tsml:heading-3 />
</tsml:header>
<tsml:header name="GRADEX">
<tsml:heading-1>Grade</tsml:heading-1>
<tsml:heading-2>Obtained</tsml:heading-2>
<tsml:heading-3>for</tsml:heading-3>
</tsml:header>
<tsml:header name="DATEACQ">
<tsml:heading-1> Date Skl</tsml:heading-1>
<tsml:heading-2> Acquired</tsml:heading-2>
<tsml:heading-3 />
</tsml:header>
</tsml:list-header>
<tsml:list-entries>
<tsml:entry>
<tsml:column name="SKILCODE">
<tsml:mode>input</tsml:mode>
<tsml:sample-value>ABCDEFGHIJ</tsml:sample-value>
<tsml:format>
```

```
<tsml:type>alpha</tsml:type>
<tsml:display-max-length>10</tsml:display-max-length>
<tsml:max-length>10</tsml:max-length>
<tsml:input-case>uppercase</tsml:input-case>
<tsml:keyboardshift>O</tsml:keyboardshift>
</tsml:format>
</tsml:column>
<tsml:column name="SKILDESC">
<tsml:mode>input</tsml:mode>
<tsml:sample-value>ABCDEFGHIJKLMNQRST</tsml:sample-value>
<tsml:format>
<tsml:type>alpha</tsml:type>
<tsml:display-max-length>20</tsml:display-max-length>
<tsml:max-length>20</tsml:max-length>
<tsml:input-case>uppercase</tsml:input-case>
<tsml:keyboardshift>O</tsml:keyboardshift>
</tsml:format>
</tsml:column>
<tsml:column name="GRADEX">
<tsml:mode>input</tsml:mode>
<tsml:sample-value>D</tsml:sample-value>
<tsml:format>
<tsml:type>alpha</tsml:type>
<tsml:display-max-length>1</tsml:display-max-length>
<tsml:max-length>1</tsml:max-length>
<tsml:input-case>uppercase</tsml:input-case>
<tsml:keyboardshift />
</tsml:format>
<tsml:use-weblets>
<tsml:use-weblet name="std_dropdown.std_dropdown" technology-
service="LANSA:XHTML" />
</tsml:use-weblets>
</tsml:column>
<tsml:column name="DATEACQ">
<tsml:mode>input</tsml:mode>
<tsml:sample-value>12/34/56</tsml:sample-value>
<tsml:format>
<tsml:type>signed</tsml:type>
<tsml:display-max-length>11</tsml:display-max-length>
```

```
<tsml:max-length>6</tsml:max-length>
<tsml:total-digits>6</tsml:total-digits>
<tsml:fraction-digits>0</tsml:fraction-digits>
<tsml:decimal-separator>.</tsml:decimal-separator>
</tsml:format>
</tsml:column>
</tsml:entry>
</tsml:list-entries>
</tsml:list>
</tsml:lists>

</tsml:data>
```

7.4 WebRoutineの TSP スタイル・シートと LANSA エディター

WebRoutineの TSP スタイル・シートは、LANSA エディターでWeb デザイナーのリストやフィールドをドラッグ・アンド・ドロップする時、および生成時の両方で使用されます。

7.4.1 ペイロード・ラッパーの XSL スタイルシート

7.4.2 フィールドのドラッグ・アンド・ドロップの例

7.4.1 ペイロード・ラッパーの XSL スタイルシート

LANSA エディターはペイロード・ラッパーの XSL スタイルシートを使って、フィールドやリスト TSML ノードを変換し、コンテンツを取得してデザインに追加します。

ペイロード・ラッパーの XSL スタイルシートは WebRoutine の (現在アクティブな TSP 用の) TSP スタイル・シートをインポートします。この XSL の出力は、LANSA エディターがデザインに追加する必要のあるコンテンツです。(フィールドやリストの XSL インポート、lxml データ・ノード、XSL)

ペイロード・ラッパーでは、XSL テンプレートが WebRoutine の TSP スタイル・シートで定義されると想定される場合があることに注意してください。

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- (c) 2002, 2013 LANSAS -->
<!-- WAM Editor TSP Stylesheet Wrapper -->
<!-- $Workfile:: tsp_payload_wrapper.xml $ -->
<!-- $Revision::3 $ -->

<xsl:transform version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xslt="http://www.lansa.com/2002/XSL/Transform-Alias"
  xmlns:lxml="http://www.lansa.com/2002/XML/Runtime-Data"
  xmlns:tsml="http://www.lansa.com/2002/XML/Generation-Metadate"
  xmlns:wd="http://www.lansa.com/2002/XSL/Weblet-Design"
  xmlns:lansa_design="http://www.lansa.com/2002/XML/Design"
  xmlns="http://www.w3.org/1999/xhtml"
  exclude-result-prefixes="tsml">

  <xsl:import href="%tsp_webroutine%.xsl"/>

  <xsl:output method="xml" omit-xml-declaration="no"
    encoding="UTF-8" indent="yes"/>
  <xsl:namespace-alias stylesheet-prefix="xslt" result-prefix="xsl"/>

  <xsl:template match="tsml:data[@full-document = 'false']">
    <lansa_design:payload
```

```
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:lxml="http://www.lansa.com/2002/XML/Runtime-Data"
  xmlns:wd="http://www.lansa.com/2002/XSL/Weblet-Design"
  xmlns:lansa_design="http://www.lansa.com/2002/XML/Design"
  xmlns="http://www.w3.org/1999/xhtml">
```

```
<lansa_design:imports>
<xsl:call-template name="weblet-imports"/>
</lansa_design:imports>
```

```
<xsl:apply-templates select="tsml:lxml-data"/>
```

```
<lansa_design:content>
<xsl:apply-templates select="*[not(self::tsml:lxml-data)]"/>
</lansa_design:content>
</lansa_design:payload>
</xsl:template>
```

```
<xsl:template match="tsml:lxml-data">
<lansa_design:lxml-data>
<xsl:apply-imports />
</lansa_design:lxml-data>
</xsl:template>
```

```
<xsl:template match="tsml:fields">
<xsl:apply-templates select="tsml:field" />
</xsl:template>
```

```
<xsl:template match="tsml:field">
<lansa_design:label>
<xsl:if test="(tsml:mode != 'hidden') and (tsml:mode != 'private')">
<xsl:call-template name="field-caption">
<xsl:with-param name="field" select="."/>
</xsl:call-template>
</xsl:if>
</lansa_design:label>
<lansa_design:value>
<xsl:if test="tsml:mode != 'private'">
<xsl:call-template name="field-value">
```

```

<xsl:with-param name="field" select="."/>
</xsl:call-template>
</xsl:if>
</lansa_design:value>
</xsl:template>

<xsl:template match="tsml:lists[not(@column-only)]">
<lansa_design:reference>
<xsl:apply-imports />
</lansa_design:reference>
<lansa_design:implementation>
<xsl:apply-templates select="tsml:list" mode="template_definition"/>
</lansa_design:implementation>
</xsl:template>

<xsl:template match="tsml:lists[@column-only]">
<xsl:apply-templates select="tsml:list"/>
</xsl:template>

<xsl:template match="tsml:lists[@column-only]/tsml:list">
<lansa_design:value>
<xsl:variable name="inline_list"
  select="(@inline = 'true') or ((@inline = 'default') and
$g_inline_lists)"/>
<xsl:apply-templates select="./tsml:list-entries/tsml:entry/tsml:column"
  mode="column_placement">
<xsl:with-param name="inline_list" select="$inline_list"/>
</xsl:apply-templates>
</lansa_design:value>
</xsl:template>
</xsl:transform>

```

7.4.2 フィールドのドラッグ・アンド・ドロップの例

以下の例では、フィールド、リストやリスト・カラムのTSML ノード、および結果のドキュメントがペイロード・ラッパーのXSLシートにより作成される様子が示されています。この例では、テクノロジー・サービスに XHTML を使用しています。

[フィールドのドラッグ・アンド・ドロップ](#)

[リストのドラッグ・アンド・ドロップ](#)

[リスト・カラムのドラッグ・アンド・ドロップ](#)

フィールドのドラッグ・アンド・ドロップ
フィールドのTSML ノード
フィールドのドラッグ・アンド・ドロップの出力

フィールドのTSML ノード

次は、ウェブレット std_dropdown でビジュアルライズされたピックリストが付いたフィールドの TSML ノードです。

```
<?xml version="1.0" encoding="UTF-8"?>

<tsml:data full-document="false" inline="none"
  xmlns:tsml="http://www.lansa.com/2002/XML/Generation-
  Metadata">

  <tsml:weblets>
  <tsml:weblet name="std_dropdown.std_dropdown">
  <tsml:technology-services>
  <tsml:technology-service name="LANSA:XHTML"
    mod-id="20121221132340000">
  <tsml:template-params>
  <tsml:template-param>
  <tsml:param-name>display_mode</tsml:param-name>
  <tsml:param-role>std:display_mode</tsml:param-role>
  <tsml:param-select>'input'</tsml:param-select>
  </tsml:template-param>
  <tsml:template-param>
  <tsml:param-name>items</tsml:param-name>
  <tsml:param-role>std:picklist</tsml:param-role>
  <tsml:param-select>document(")/*/lxml:data/lxml:dropdown</tsml:param-
  select>
  </tsml:template-param>
  <tsml:template-param>
  <tsml:param-name>name</tsml:param-name>
  </tsml:template-param>
  <tsml:template-param>
  <tsml:param-name>pos_absolute</tsml:param-name>
  <tsml:param-role>std:pos_absolute_design</tsml:param-role>
  </tsml:template-param>
  <tsml:template-param>
  <tsml:param-name>width_design</tsml:param-name>
  <tsml:param-role>std:width_design</tsml:param-role>
```

```
</tsml:template-param>
</tsml:template-params>
</tsml:technology-service>
</tsml:technology-services>
</tsml:weblet>
</tsml:weblets>
```

```
<tsml:lxml-data xmlns:lxml="http://www.lansa.com/2002/XML/Runtime-Data">
```

```
<lxml:picklist id="380F247733D94ECDA37898AA9AEFCCD5"
  field-ref="DAYOFWEEK">
```

```
<lxml:item default="true">
```

```
<lxml:caption>
```

```
<lxml:variable name="MTXTDEMCALEN05801" /></lxml:caption>
```

```
<lxml:value>MON</lxml:value>
```

```
</lxml:item>
```

```
<lxml:item>
```

```
<lxml:caption>
```

```
<lxml:variable name="MTXTDEMCALEN06001" /></lxml:caption>
```

```
<lxml:value>TUE</lxml:value>
```

```
</lxml:item>
```

```
<lxml:item>
```

```
<lxml:caption>
```

```
<lxml:variable name="MTXTDEMCALEN06201" /></lxml:caption>
```

```
<lxml:value>WED</lxml:value>
```

```
</lxml:item>
```

```
<lxml:item>
```

```
<lxml:caption>
```

```
<lxml:variable name="MTXTDEMCALEN06401" /></lxml:caption>
```

```
<lxml:value>THU</lxml:value>
```

```
</lxml:item>
```

```
<lxml:item>
```

```
<lxml:caption>
```

```
<lxml:variable name="MTXTDEMCALEN06601" /></lxml:caption>
```

```
<lxml:value>FRI</lxml:value>
```

```
</lxml:item>
```

```
<lxml:item>
```

```
<lxml:caption>
```

```
<lxml:variable name="MTXTDEMCALEN06801" /></lxml:caption>
```

```
<lxml:value>SAT</lxml:value>
</lxml:item>
<lxml:item>
<lxml:caption>
<lxml:variable name="MTEXTDEMCALEN07001" /></lxml:caption>
<lxml:value>SUN</lxml:value>
</lxml:item>
</lxml:picklist>
</tsml:lxml-data>
```

```
<tsml:fields>
<tsml:field name="DAYOFWEEK">
<tsml:mode>input</tsml:mode>
<tsml:sample-value>MON</tsml:sample-value>
<tsml:format>
<tsml:type>alpha</tsml:type>
<tsml:display-max-length>3</tsml:display-max-length>
<tsml:max-length>3</tsml:max-length>
<tsml:input-case>uppercase</tsml:input-case>
<tsml:keyboardshift />
</tsml:format>
<tsml:caption ref="description">
<tsml:label>Day of the week</tsml:label>
<tsml:description>Day of the week</tsml:description>
<tsml:heading-1>Day</tsml:heading-1>
<tsml:heading-2>of</tsml:heading-2>
<tsml:heading-3>the</tsml:heading-3>
</tsml:caption>
<tsml:use-webllets>
<tsml:use-weblet name="std_dropdown.std_dropdown"
    technology-service="LANSA:JQMOBILE" />
<tsml:use-weblet name="std_dropdown.std_dropdown"
    technology-service="LANSA:PPC_XHTML" />
<tsml:use-weblet name="std_dropdown.std_dropdown"
    technology-service="LANSA:XHTML" />
</tsml:use-webllets>
</tsml:field>
</tsml:fields>
</tsml:data>
```


フィールドのドラッグ・アンド・ドロップの出力

以下は、変換の出力を示しています。LANSA エディターは、関連するセクションをターゲット・ドキュメント (WebRoutine の XSL スタイルシート) 内の適切な位置に設定します。

```
<?xml version="1.0" encoding="UTF-8"?>
<lansa_design:payload xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:lxml="http://www.lansa.com/2002/XML/Runtime-Data"
  xmlns:wd="http://www.lansa.com/2002/XSL/Weblet-Design"
  xmlns:lansa_design="http://www.lansa.com/2002/XML/Design"
  xmlns="http://www.w3.org/1999/xhtml">
<lansa_design:imports>
<xsl:import href="std_dropdown.xsl" />
</lansa_design:imports>
<lansa_design:lxml-data>
<lxml:data>
<lxml:picklist id="380F247733D94ECDA37898AA9AEFCCD5">
<lxml:item default="true">
<lxml:caption>
<lxml:variable name="MTXTDEMCALEN05801"></lxml:variable>
</lxml:caption>
<lxml:value>MON</lxml:value>
</lxml:item>
<lxml:item>
<lxml:caption>
<lxml:variable name="MTXTDEMCALEN06001"></lxml:variable>
</lxml:caption>
<lxml:value>TUE</lxml:value>
</lxml:item>
<lxml:item>
<lxml:caption>
<lxml:variable name="MTXTDEMCALEN06201"></lxml:variable>
</lxml:caption>
<lxml:value>WED</lxml:value>
</lxml:item>
<lxml:item>
<lxml:caption>
<lxml:variable name="MTXTDEMCALEN06401"></lxml:variable>
```

```

</lxml:caption>
<lxml:value>THU</lxml:value>
</lxml:item>
<lxml:item>
<lxml:caption>
<lxml:variable name="MTXTDEMCALEN06601"></lxml:variable>
</lxml:caption>
<lxml:value>FRI</lxml:value>
</lxml:item>
<lxml:item>
<lxml:caption>
<lxml:variable name="MTXTDEMCALEN06801"></lxml:variable>
</lxml:caption>
<lxml:value>SAT</lxml:value>
</lxml:item>
<lxml:item>
<lxml:caption>
<lxml:variable name="MTXTDEMCALEN07001"></lxml:variable>
</lxml:caption>
<lxml:value>SUN</lxml:value>
</lxml:item>
</lxml:picklist>

</lxml:data>

</lansa_design:lxml-data>
<lansa_design:content>
<lansa_design:label>
<label class="caption" for="DAYOFWEEK">
<xsl:value-of select="key('field-caption', 'DAYOFWEEK')/lxml:description"
/>
</label>
</lansa_design:label>
<lansa_design:value>
<xsl:call-template name="std_dropdown">
<xsl:with-param name="name" select=""DAYOFWEEK"" />
<xsl:with-param name="value" select="key('field-value', 'DAYOFWEEK')"/>
<xsl:with-param name="display_mode" select=""input"" />
<xsl:with-param name="items"

```

```
select="document('')/*/*/lxml:data/lxml:picklist[@id =  
'380F247733D94ECDA37898AA9AEFCCD5']" />  
</xsl:call-template>  
</lansa_design:value>  
</lansa_design:content>  
</lansa_design:payload>
```

リストのドラッグ・アンド・ドロップ

リストのTSML ノード

リストのドラッグ・アンド・ドロップの出力

リストのTSML ノード

```
<?xml version="1.0" encoding="UTF-8"?>
<tsml:data full-document="false"
  inline="none"
  xmlns:tsml="http://www.lansa.com/2002/XML/Generation-
Metadata">

<tsml:weblets>
<tsml:weblet name="std_boolean.std_boolean">
<tsml:technology-services>
<tsml:technology-service name="LANSA:XHTML"
  mod-id="20121220163646000"
  proxy-format="__,_PROXY">
<tsml:template-params>
<tsml:template-param>
<tsml:param-name>display_mode</tsml:param-name>
<tsml:param-role>std:display_mode</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>pos_absolute</tsml:param-name>
<tsml:param-role>std:pos_absolute_design</tsml:param-role>
</tsml:template-param>
</tsml:template-params>
</tsml:technology-service>
</tsml:technology-services>
</tsml:weblet>
<tsml:weblet name="std_input.std_input">
<tsml:technology-services>
<tsml:technology-service name="LANSA:JQMOBILE"
  mod-id="20130301081954000">
<tsml:template-params>
<tsml:template-param>
<tsml:param-name>displayMode</tsml:param-name>
<tsml:param-role>std:display_mode</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>id</tsml:param-name>
```

```
<tsml:param-select>concat($lweb_WRName,'_',$name)</tsml:param-select>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>label</tsml:param-name>
<tsml:param-role>std:field_caption</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>maxlength</tsml:param-name>
<tsml:param-role>std:field_maxlength</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>name</tsml:param-name>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>rdmlxDataType</tsml:param-name>
<tsml:param-role>std:rdmlx_data_type</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>size</tsml:param-name>
<tsml:param-role>std:field_display_length</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>type</tsml:param-name>
<tsml:param-role>std:field_input_type</tsml:param-role>
<tsml:param-select>'text'</tsml:param-select>
</tsml:template-param>
</tsml:template-params>
</tsml:technology-service>
</tsml:technology-services>
</tsml:weblet>
<tsml:weblet name="std_datepicker.std_datepicker">
<tsml:technology-services>
<tsml:technology-service name="LANSA:XHTML"
  mod-id="20121220142947000"
  proxy-format="__,_PROXY">
<tsml:template-params>
<tsml:template-param>
<tsml:param-name>allow_sqlnull</tsml:param-name>
<tsml:param-role>std:allow_sqlnull</tsml:param-role>
```

```
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>display_mode</tsml:param-name>
<tsml:param-role>std:display_mode</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>pos_absolute</tsml:param-name>
<tsml:param-role>std:pos_absolute_design</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>size</tsml:param-name>
<tsml:param-role>std:field_size</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>width</tsml:param-name>
<tsml:param-role>std:width_design</tsml:param-role>
</tsml:template-param>
</tsml:template-params>
</tsml:technology-service>
</tsml:technology-services>
</tsml:weblet>
<tsml:weblet name="std_char.std_char">
<tsml:technology-services>
<tsml:technology-service name="LANSA:XHTML"
  mod-id="20121220115335000">
<tsml:template-params>
<tsml:template-param>
<tsml:param-name>class</tsml:param-name>
<tsml:param-role>std:field_css_class</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>display_length</tsml:param-name>
<tsml:param-role>std:field_display_length</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>display_mode</tsml:param-name>
<tsml:param-role>std:display_mode</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
```

```
<tsml:param-name>height</tsml:param-name>
<tsml:param-role>std:height_design</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>keyboard_shift</tsml:param-name>
<tsml:param-role>std:keyboard_shift</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>maxlength</tsml:param-name>
<tsml:param-role>std:field_maxlength</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>pos_absolute</tsml:param-name>
<tsml:param-role>std:pos_absolute_design</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>type</tsml:param-name>
<tsml:param-role>std:field_input_type</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>width</tsml:param-name>
<tsml:param-role>std:width_design</tsml:param-role>
</tsml:template-param>
</tsml:template-params>
</tsml:technology-service>
</tsml:technology-services>
</tsml:weblet>
</tsml:weblets>
```

```
<tsml:lists default-sample-size="5">
<tsml:list name="LIST01" inline="false">
<tsml:mode>input</tsml:mode>
<tsml:list-header>
<tsml:header name="BOOL01">
<tsml:heading-1>Boolean</tsml:heading-1>
<tsml:heading-2>field</tsml:heading-2>
<tsml:heading-3 /></tsml:header>
<tsml:header name="DAT01">
<tsml:heading-1>Date</tsml:heading-1>
```

```
<tsml:heading-2>field</tsml:heading-2>
<tsml:heading-3 /></tsml:header>
<tsml:header name="CHR01">
<tsml:heading-1>DBCS Char</tsml:heading-1>
<tsml:heading-2>field</tsml:heading-2>
<tsml:heading-3>length 10</tsml:heading-3>
</tsml:header>
</tsml:list-header>
<tsml:list-entries>
<tsml:entry>
<tsml:column name="BOOL01">
<tsml:mode>input</tsml:mode>
<tsml:sample-value>False</tsml:sample-value>
<tsml:format>
<tsml:type>boolean</tsml:type>
<tsml:display-max-length>5</tsml:display-max-length>
<tsml:max-length>1</tsml:max-length>
</tsml:format>
<tsml:use-weblets>
<tsml:use-weblet name="std_boolean.std_boolean"
  technology-service="LANSA:JQMOBILE" />
<tsml:use-weblet name="std_boolean.std_boolean"
  technology-service="LANSA:PPC_XHTML" />
<tsml:use-weblet name="std_boolean.std_boolean"
  technology-service="LANSA:XHTML" />
</tsml:use-weblets>
</tsml:column>
<tsml:column name="DAT01">
<tsml:mode>input</tsml:mode>
<tsml:sample-value>1/01/1900</tsml:sample-value>
<tsml:format>
<tsml:type>date</tsml:type>
<tsml:display-max-length>10</tsml:display-max-length>
<tsml:max-length>10</tsml:max-length>
<tsml:sql-nullable>true</tsml:sql-nullable>
</tsml:format>
<tsml:use-weblets>
<tsml:use-weblet name="std_input.std_input"
  technology-service="LANSA:JQMOBILE" />
```

```
<tsml:use-weblet name="std_datepicker.std_datepicker"
  technology-service="LANSA:XHTML" />
</tsml:use-weblets>
</tsml:column>
<tsml:column name="CHR01">
<tsml:mode>input</tsml:mode>
<tsml:sample-value>ABCDEFGHJIJ</tsml:sample-value>
<tsml:format>
<tsml:type>char</tsml:type>
<tsml:display-max-length>10</tsml:display-max-length>
<tsml:max-length>10</tsml:max-length>
<tsml:input-case>uppercase</tsml:input-case>
<tsml:keyboardshift>J</tsml:keyboardshift>
</tsml:format>
<tsml:use-weblets>
<tsml:use-weblet name="std_input.std_input"
  technology-service="LANSA:JQMOBILE" />
<tsml:use-weblet name="std_char.std_char"
  technology-service="LANSA:PPC_XHTML" />
<tsml:use-weblet name="std_char.std_char"
  technology-service="LANSA:XHTML" />
</tsml:use-weblets>
</tsml:column>
</tsml:entry>
</tsml:list-entries>
</tsml:list>
</tsml:lists>
</tsml:data>
```

リストのドラッグ・アンド・ドロップの出力

```
<?xml version="1.0" encoding="UTF-8"?>
<lansa_design:payload xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:lxml="http://www.lansa.com/2002/XML/Runtime-Data"
xmlns:wd="http://www.lansa.com/2002/XSL/Weblet-Design"
xmlns:lansa_design="http://www.lansa.com/2002/XML/Design"
xmlns="http://www.w3.org/1999/xhtml">
<lansa_design:imports>
<xsl:import href="std_boolean.xsl" />
<xsl:import href="std_char.xsl" />
<xsl:import href="std_datepicker.xsl" />
</lansa_design:imports>
<lansa_design:content>
<lansa_design:reference>
<xsl:apply-templates
select="/lxml:data/lxml:lists/lxml:list[@name='LIST01']"
wd:listname="LIST01">
<xsl:with-param name="allowSort" select="true()" />
<xsl:with-param name="allowColResize" select="true()" />
<xsl:with-param name="hoverEffect" select="false()" />
<xsl:with-param name="selectableRows" select="false()" />
<xsl:with-param name="hide_header_if_empty" select="true()" />
</xsl:apply-templates>
</lansa_design:reference>
<lansa_design:implementation>

<xsl:template match="/lxml:data/lxml:lists/lxml:list[@name='LIST01']">
<xsl:param name="allowSort" wd:type="std:boolean" select="true()"
wd:tip_id="" />
<xsl:param name="allowColResize" wd:type="std:boolean" select="true()"
wd:tip_id="" />
<xsl:param name="hoverEffect" wd:type="std:boolean" select="false()"
wd:tip_id="" />
<xsl:param name="selectableRows" wd:type="std:boolean" select="false()"
wd:tip_id="" />
<xsl:param name="hide_header_if_empty" wd:type="std:boolean"
select="true()" wd:tip_id="" />
```

```

<xsl:variable name="thelist"
select="/lxml:data/lxml:lists/lxml:list[@name='LIST01']" />
<input type="hidden" name="LIST01.." value="{count(lxml:list-
entries/lxml:entry[1])}" />
<div class="std_grid_wrapper" id="LIST01_wrap">
<xsl:if test="$lweb_design_mode">
<xsl:attribute name="class">std_grid_wrapper_designtime</xsl:attribute>
</xsl:if>
<table class="std_grid ui-widget" id="LIST01">
<xsl:if test="not($hide_header_if_empty) or ($thelist/@row-count != 0)">
<thead>
<tr class="list-h ui-widget-header">
<th class="ltext BOOL01 std_grid_sort_indicator" __decimalseparator=""
__formattype="boolean" __mode="input" __allowsort="true">
<xsl:for-each select="$thelist/lxml:list-header/lxml:header[1]/*[./text()
[normalize-space(.)!='']]" wd:edit-as-list="false">
<xsl:value-of select="."/><xsl:if test="not(position() = last())"><br /></xsl:if>
</xsl:for-each>
<div class="std_grid_cell_size">
<xsl:if test="boolean(/lxml:data/lxml:context[@design])">
<xsl:attribute name="class">hidden__ </xsl:attribute>
</xsl:if>
<xsl:comment>.</xsl:comment>
</div>
</th>
<th class="number DAT01 std_grid_sort_indicator" __decimalseparator=""
__formattype="date" __mode="input" __allowsort="true">
<xsl:for-each select="$thelist/lxml:list-header/lxml:header[2]/*[./text()
[normalize-space(.)!='']]" wd:edit-as-list="false">
<xsl:value-of select="."/><xsl:if test="not(position() = last())"><br /></xsl:if>
</xsl:for-each>
<div class="std_grid_cell_size">
<xsl:if test="boolean(/lxml:data/lxml:context[@design])">
<xsl:attribute name="class">hidden__ </xsl:attribute>
</xsl:if>
<xsl:comment>.</xsl:comment>
</div>
</th>
<th class="utext CHR01 std_grid_sort_indicator" __decimalseparator=""

```

```

__formattype="char" __mode="input" __allowsort="true">
<xsl:for-each select="$thelist/lxml:list-header/lxml:header[3]/*[.//text()
[normalize-space(.)!="]]" wd:edit-as-list="false">
<xsl:value-of select="."/><xsl:if test="not(position() = last())"><br /></xsl:if>
</xsl:for-each>
<div class="std_grid_cell_size">
<xsl:if test="boolean(/lxml:data/lxml:context[@design])">
<xsl:attribute name="class">hidden__</xsl:attribute>
</xsl:if>
<xsl:comment>.</xsl:comment>
</div>
</th>
</tr>
</thead>
</xsl:if>
<tbody class="ui-widget-content">
<xsl:for-each select="$thelist/lxml:list-entries/lxml:entry">
<xsl:variable name="BOOL01" select="lxml:column[1]" />
<xsl:variable name="DAT01" select="lxml:column[2]" />
<xsl:variable name="CHR01" select="lxml:column[3]" />
<tr __oddr="list-o" __evenrc="list-e">
<xsl:attribute name="class">
<xsl:choose>
<xsl:when test="position() mod 2">list-o</xsl:when>
<xsl:otherwise>list-e</xsl:otherwise>
</xsl:choose>
</xsl:attribute>
<td class="BOOL01">
<xsl:attribute name="__cellvalue"><xsl:value-of select="$BOOL01" />
</xsl:attribute>
<xsl:call-template name="std_boolean">
<xsl:with-param name="name" select="$BOOL01/@id" />
<xsl:with-param name="value" select="$BOOL01" />
<xsl:with-param name="display_mode" select=""input"" />
</xsl:call-template>
</td>
<td class="DAT01">
<xsl:attribute name="__cellvalue"><xsl:value-of select="$DAT01" />
</xsl:attribute>

```

```

<xsl:call-template name="std_datepicker">
<xsl:with-param name="name" select="$DAT01/@id" />
<xsl:with-param name="value" select="$DAT01" />
<xsl:with-param name="allow_sqlnull" select="true()" />
<xsl:with-param name="display_mode" select=""input"" />
<xsl:with-param name="size" select="10" />
</xsl:call-template>
</td>
<td class="CHR01">
<xsl:attribute name="__cellvalue"><xsl:value-of select="$CHR01" />
</xsl:attribute>
<xsl:call-template name="std_char">
<xsl:with-param name="name" select="$CHR01/@id" />
<xsl:with-param name="value" select="$CHR01" />
<xsl:with-param name="class" select=""utext"" />
<xsl:with-param name="display_length" select="10" />
<xsl:with-param name="display_mode" select=""input"" />
<xsl:with-param name="keyboard_shift" select=""J"" />
<xsl:with-param name="maxlength" select="10" />
<xsl:with-param name="type" select=""text"" />
</xsl:call-template>
</td>
</tr>
</xsl:for-each>
</tbody>
</table>
</div>
<script type="text/javascript">
<xsl:text disable-output-escaping="yes">//&lt;![CDATA[</xsl:text>
register_std_grid('LIST01',{
  columns:3,
  allowSort:<xsl:value-of select="$allowSort" />,
  allowColResize:<xsl:value-of select="$allowColResize" />,
  hoverEffect:<xsl:value-of select="$hoverEffect" />,
  selectableRows:<xsl:value-of select="$selectableRows" />
});
<xsl:text disable-output-escaping="yes">//]]&gt;</xsl:text>
</script>
</xsl:template>

```

</lansa_design:implementation>

</lansa_design:content>

</lansa_design:payload>

リスト・カラムのドラッグ・アンド・ドロップ

リスト・カラムのTSML ノード

リスト・カラムのドラッグ・アンド・ドロップの出力

リスト・カラムのTSML ノード

```
<?xml version="1.0" encoding="UTF-8"?>
<tsml:data full-document="false"
  inline="none"
  xmlns:tsml="http://www.lansa.com/2002/XML/Generation-Metadata">

<tsml:weblets>
<tsml:weblet name="std_input.std_input">
<tsml:technology-services>
<tsml:technology-service name="LANSA:JQMOBILE"
  mod-id="20130102151636000">
<tsml:template-params>
<tsml:template-param>
<tsml:param-name>displayMode</tsml:param-name>
<tsml:param-role>std:display_mode</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>label</tsml:param-name>
<tsml:param-role>std:field_caption</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>maxlength</tsml:param-name>
<tsml:param-role>std:field_maxlength</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>rdmlxDataType</tsml:param-name>
<tsml:param-role>std:rdmlx_data_type</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>size</tsml:param-name>
<tsml:param-role>std:field_display_length</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>type</tsml:param-name>
<tsml:param-role>std:field_input_type</tsml:param-role>
</tsml:template-param>
</tsml:template-params>
```

```
</tsml:technology-service>
</tsml:technology-services>
</tsml:weblet>
<tsml:weblet name="std_char.std_char">
<tsml:technology-services>
<tsml:technology-service name="LANSA:XHTML"
    mod-id="20121220115335000">
<tsml:template-params>
<tsml:template-param>
<tsml:param-name>class</tsml:param-name>
<tsml:param-role>std:field_css_class</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>display_length</tsml:param-name>
<tsml:param-role>std:field_display_length</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>display_mode</tsml:param-name>
<tsml:param-role>std:display_mode</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>height</tsml:param-name>
<tsml:param-role>std:height_design</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>keyboard_shift</tsml:param-name>
<tsml:param-role>std:keyboard_shift</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>maxlength</tsml:param-name>
<tsml:param-role>std:field_maxlength</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>pos_absolute</tsml:param-name>
<tsml:param-role>std:pos_absolute_design</tsml:param-role>
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>type</tsml:param-name>
<tsml:param-role>std:field_input_type</tsml:param-role>
```

```
</tsml:template-param>
<tsml:template-param>
<tsml:param-name>width</tsml:param-name>
<tsml:param-role>std:width_design</tsml:param-role>
</tsml:template-param>
</tsml:template-params>
</tsml:technology-service>
</tsml:technology-services>
</tsml:weblet>
</tsml:weblets>
```

```
<tsml:lists
  default-sample-size="5"
  column-only="true">
<tsml:list name="LIST01" inline="false">
<tsml:mode>input</tsml:mode>
<tsml:list-entries>
<tsml:entry>
<tsml:column name="CHR01">
<tsml:mode>input</tsml:mode>
<tsml:sample-value>ABCDEFGHJIJ</tsml:sample-value>
<tsml:format>
<tsml:type>char</tsml:type>
<tsml:display-max-length>10</tsml:display-max-length>
<tsml:max-length>10</tsml:max-length>
<tsml:input-case>uppercase</tsml:input-case>
<tsml:keyboardshift>J</tsml:keyboardshift>
</tsml:format>
<tsml:use-weblets>
<tsml:use-weblet name="std_input.std_input"
  technology-service="LANSA:JQMOBILE" />
<tsml:use-weblet name="std_char.std_char"
  technology-service="LANSA:PPC_XHTML" />
<tsml:use-weblet name="std_char.std_char"
  technology-service="LANSA:XHTML" />
</tsml:use-weblets>
</tsml:column>
</tsml:entry>
</tsml:list-entries>
```

```
</tsml:list>  
</tsml:lists>  
</tsml:data>
```

リスト・カラムのドラッグ・アンド・ドロップの出力

```
<?xml version="1.0" encoding="UTF-8"?>
<lansa_design:payload xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:lxml="http://www.lansa.com/2002/XML/Runtime-Data"
xmlns:wd="http://www.lansa.com/2002/XSL/Weblet-Design"
xmlns:lansa_design="http://www.lansa.com/2002/XML/Design"
xmlns="http://www.w3.org/1999/xhtml">
<lansa_design:imports>
<xsl:import href="std_char.xsl" />
</lansa_design:imports>
<lansa_design:content>
<lansa_design:value>
<xsl:call-template name="std_char">
<xsl:with-param name="name" select="$CHR01/@id" />
<xsl:with-param name="value" select="$CHR01" />
<xsl:with-param name="class" select=""utext"" />
<xsl:with-param name="display_length" select="10" />
<xsl:with-param name="display_mode" select=""input"" />
<xsl:with-param name="keyboard_shift" select=""J"" />
<xsl:with-param name="maxlength" select="10" />
<xsl:with-param name="type" select=""text"" />
</xsl:call-template>
</lansa_design:value>
</lansa_design:content>
</lansa_design:payload>
```

7.5 テクノロジ・サービスのデフォルト・ウェブレット

テクノロジ・サービスのデフォルト・ウェブレットを指定できます。デフォルトのウェブレットは、フィールドにウェブレット・ビジュアルライゼーションがない場合に使用されます。デフォルトのウェブレットが指定されない場合、WebRoutine の XSL スタイルシートに非ウェブレット・フィールドやリスト・カラムを処理するテンプレートが必要です。

指定するウェブレットは自身のテクノロジ・サービス内に存在していなければいけません。次の命令を、WebRoutine XSL スタイル・シートの `<xsl:output>` 命令と同じレベルに追加します。この例は、テクノロジ・サービス JQMOBILE 用のデフォルトのウェブレットを示しています。

```
<xsl:output method="xml" omit-xml-declaration="no" encoding="UTF-8"
indent="yes"/>
```

```
<xsl:namespace-alias stylesheet-prefix="xslt" result-prefix="xsl"/>
```

```
<!-- Default Weblet -->
```

```
<tsml:definition>
```

```
  <tsml:default-weblet name="std_input.std_input" />
```

```
</tsml:definition>
```

7.6 ウェブレットとウェブレット・テンプレート

WAMと共に各HTMLテクノロジー・サービスごとの標準ウェブレットのセットが提供されます。ウェブレットは1つまたは複数のウェブレット・テンプレートを含むリポジトリのオブジェクトです。ウェブレット・テンプレートは共通の機能をまとめた再利用可能コンポーネントで、自身のWebroutineのデザインにドラッグ・アンド・ドロップすることができます。

1つのウェブレットに多くのウェブレット・テンプレートが含まれることがあるかもしれませんが、一般的には各ウェブレットに1つのテンプレートだけを含むのが通常です。それが故に"ウェブレット"という言葉がウェブレット・テンプレートの意味で使用されることがよくあります。1つのウェブレットは1つもしくは複数のテクノロジー・サービスに存在することができます。

注：ウェブレットにはメインのテンプレートの特別版として、インライン・リストを使用する2つめのウェブレット・テンプレートが含まれる場合があります。この"インライン"用のウェブレット・テンプレートは、リポジトリのウェブレット・テンプレートのセクションに表示されません。これはWAMエディターが必要に応じて自動的に正しいテンプレートを使用するからです。

提供されるウェブレットを直接変更することは、絶対にしないでください。区画の初期化をWeb使用可能にして実行する度に、提供されているウェブレットが再インポートされ、リポジトリ内のウェブレットが上書きされます。ウェブレットをカスタマイズしたい場合は、提供されているウェブレットをコピーし、このコピーを編集してください。(これにはウェブレットが使用するJavaScriptのファンクションも含まれます。これも将来のバージョンで変更される可能性があります。)

提供されているウェブレットは標準のネーミング規則が使われており、ウェブレット名には接頭辞'std_'が付いています。独自に作成するカスタム・ウェブレットにはこの接頭辞を使用することはできません。

以下も参照してください。

[XHTMLテクノロジー・サービス用のウェブレット](#)

[jQuery Mobileテクノロジー・サービス用のウェブレット](#)

8. XHTMLテクノロジー・サービス用のウェブレット

このセクションでは、XHTMLテクノロジー・サービスと共に提供されるウェブレット・テンプレートについて説明します。

ウェブレット・テンプレートのリポジトリ・ビューにはテンプレートがグループごとに表示されます。1つのウェブレット・テンプレートが複数のグループに属する場合があります（構成はウェブレットで右クリック）。標準のグループは以下の通りです。

[標準ウェブレット](#)

[グラフ作成ウェブレット](#)

[標準フィールド・ビジュアルライゼーション](#)

[レイアウト・ウェブレット](#)

既存のユーザーの方は、「[廃止されたウェブレット](#)」も参照してください。

8.1 標準ウェブレット

このセクションでは、プロパティやWebroutine内での使用方法などの標準ウェブレットの詳細を説明します。ウェブレットで使用できる全てのプロパティを使用する必要はありません。

ウェブレット名	説明
アンカー (std_anchor)	アンカー・ウェブレットはハイパーリンク(あるいはアンカー)コントロールを提供します。
アタッチメント・パネル (std_attach_panel_v2)	コンテンツをドロップできる5つの領域(左、上、右、中、下)を持ったパネルを提供します。
オートコンプリート (std_autocomplete)	オートコンプリート・ウェブレットでは、テキストの入力時にAJAXを使ったWebroutineにより提供される候補が表示されます。
バナー (std_banner)	ドロップされた他のウェブレット、テキスト、エレメントなどのコンテンツをスクロールするパネル
チェックボックス (std_checkbox)	キャプション付きのチェックボックスです。
CKエディター (std_ckeditor)	WYSIWYGのリッチ・テキスト・エディターです。
クリック可能イメージ (std_click_image)	クリックできるイメージです。
コンボ・ボックス (std_dropdown)	ドロップダウン項目は、LANSAエディターの items プロパティで指定するか、リストから取り込むことができます。
動的選択ボックス (std_dynamic_select)	ドロップダウンまたはリストを作成できるエレメントで、別のフィールドをモニターして、このフィールドが変更されると自動的にエレメント自身を最新情報に更新することができます。
Excelにエクスポート (std_toexcel)	テーブルまたはグリッドをExcelスプレッドシートにエクスポートできます。

グリッド (std_grid_v2)	列のソートが可能なグリッド・コントロールです。グリッド・セルは、リストから取り込むことができます。
水平スピリッター (std_splitter_horz)	上下の画面分割ペインにコンテンツを追加できる水平分割コントロールです。
大型リスト (std_largelist)	大きなリストです。簡単なフォーマットのレポート形式(出力専用)のリストに使用します。リストはXHTMLかCSV(カンマ区切りのデータ)形式で送信できます。
リストボックス (std_listbox)	リスト・ボックス・コントロールです。項目は、リストから取り込むことも、itemsプロパティで直接指定することも可能です。
リスト・ページング・ボタン (std_list_buttons)	ブラウザ・リストの移動に使用できるボタンです。
リスト・ページング・イメージ (std_list_images)	ブラウザ・リストの移動に使用できるイメージ・ボタンです。
マークアップ (std_markup)	コンテンツを出力モードのみでビジュアライズしたい時に使用します。CKEditorウェブレットの付属です。
リスト使用のメモ (std_textarea_v2)	各行をロードしてリストに格納する複数行編集コントロールです。
フィールド使用のメモ (std_list_textarea)	フィールドを使用した複数行編集コントロールです。std_textarea_v2によって取って代わられました。
メニューバー (std_menubar)	他のWebroutineを含み、別のWebページを呼び出すメニューバー機能を提供します。
メニュー・アイテム (std_menu_item_v2)	ハイパーリンク・メニュー・アイテムです。HTMLのメニューを作成する時に使用します。
メッセージ	RDMLのMESSAGEコマンドで出力された

(std_messages)	Webroutineからのメッセージを表示します。
ナビゲーション・パネル (std_nav_panel)	ページの残りの部分からは独立した形で、WebroutineまたはURLに移動できるパネルです。
パネル (std_panel)	HTMLや他のウェブレットなどのコンテンツをドロップできるパネルです。すべてのコンテンツは、パネル内に相対配置されます。設計時のサポートとして、"グリッドにスナップ"することもできます。
ページ印刷 (std_printpage)	現在のページを印刷するハイパーリンクを提供します。
プロンプター (std_prompter)	フィールドのプロンプトをサポートするボタンです。プロンプターは（別のWAMからでも）Webroutineを呼び出してポップアップ・ウィンドウのページを表示できます。
プッシュ・ボタン (std_button_v2) および イメージ付きプッシュ・ボタン (std_image_button_v2)	このウェブレットはWebページにテーマのあるプッシュボタンを提供します。 このウェブレットの以前のバージョンは廃止されました。
ラジオ・ボタン (std_rad_button)	ラジオ・ボタンです。
ラジオ・グループ (std_radbuttons)	ラジオ・ボタンのグループです。
タブ・ページ (std_tab_pages_v2)	各タブ・ページにコンテンツを追加できるタブ・コントロールです。タブ・ページを変更、追加、削除することができます。タブのキャプションを変更するには、tabsプロパティを使用します。
ツリー・ビュー (std_treeview_v2)	展開したり、折りたたんだりできるツリーを提供します。
垂直スプリッター	左右の画面分割ペインにコンテンツを追加でき

(std_splitter_vert)

る垂直画面分割コントロールです。

8.1.1 アンカー (std_anchor)

クイック・スタート - アンカー プロパティ - アンカー

アンカー・ウェブレットはハイパーリンク(あるいはアンカー)コントロールを提供します。ハイパーテキストのリンク先を指定する、`<a>` (anchor)HTMLエレメントとほぼ同じ働きをします。

- アンカー・ウェブレットは、リンクを示すイメージやテキスト、またはこの両方を表示でき、リンク起動時のWAMの移動先を指定できます。
- このイメージやテキストは静的(ウェブレットのプロパティでリテラルとして指定)なものでも、指定のフィールド、システム変数、複数言語変数により決定されるものでも構いません。
- 宛先は、URL(例えば`http://lansa/www.yourcompany.com/`)や、実行するWAMやWebroutine、もしくは任意でWebroutineに値を渡すフィールドを指定することもできます。

アンカー・ウェブレットは以下のように表示されます(部門コード)。

Dept Code	Department Description
ADM	ADMINISTRATOR DEPT
AUD	INTERNAL AUDITING
FLT	FLEET ADMINISTRATION

アンカー・ウェブレットはリスト内のフィールドやカラムで使用されることが多く、簡単にしかも早く項目を選択したり、項目に関するアクションを開始したりできるようにします。上記の例では、部門コードがアンカーになっています。部門コードをユーザーがクリックすると、別のWebroutineが呼び出され、その部門の明細を表示します。

`currentrowhfield`と`currentrownumval`プロパティで、該当する部門コードがこのWebroutineに渡されるように指定します。

クイック・スタート - アンカー

リストのカラムでアンカーを使用するには、リストでWEB_MAPに *BOTH もしくは *OUTPUTを指定したWebroutineを作成する必要があります。LANSAエディターで生成したXSLを開き、以下の手順でリストのカラムをアンカーとして機能するように変更することができます。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[アンカー]ウェブレットを探してください。
2. このアンカー・ウェブレットをリストのカラムの上にドラッグし、左のマウス・ボタンから手を離します。カラムがアンカーを示すように変更されます。カラムの項目をクリックし、[詳細]タブをクリックしてください。アンカー・ウェブレットのnameやvalueプロパティが、ドロップしたフィールドにもとづいて既に設定されているのを確認してください。
3. currentrowhfieldプロパティ及びcurrentrownumvalプロパティを、プロパティ記述の説明通りに設定します。
4. on_click_wrnameプロパティにハイパーリンクのクリック時に呼び出すWebroutine名を設定します。Webroutineが現在のWebroutineと異なるWAMにある場合は、on_click_wamnameプロパティにWAM名を設定する必要があります。

プロパティ - アンカー

アンカー・ウェブレットのプロパティは以下のとおりです。

absolute-image-path	on_click_wname	show_in_new_window
currentrowhfield	pos_absolute_design	tab_index
currentrownumval	presubmit_js	target_window_name
formname	protocol	text_class
hide_if	reentryfield	url
mouseover_class	reentryvalue	value
name	relative-image-path	vf_wamevent
on_click_wamname		width_design

name

ウェブレットを識別する名前です。ウェブレットがフィールドをビジュアライズすると、これがフィールド名になります。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用することも可能です。

省略値

ウェブレットによりビジュアライズされたフィールドの場合、省略値の名前はフィールド名、もしくはフィールド名と行番号を組み合わせたもの(リストのフィールドの場合)になります。それ以外は、省略値の名前は自動的に生成された、一意の識別子になります。

有効値

単一引用符で囲まれた文字列。

value

このプロパティにはハイパーリンクに表示されるテキストを指定します。ウェブレットによりフィールドがビジュアルライズされる場合、この値により表示されるフィールドが識別されます。

省略値

適用される省略値はありません。 - ハイパーリンクにテキストを表示したい場合は、値を指定してください。(ハイパーリンクでイメージを表示することもできます。relative-image-pathプロパティ及びabsolute-image-pathプロパティを参照してください。)

有効値

単一引用符で囲まれたテキスト、フィールド名、システム変数、または複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

currentrowhfield

currentrownumval プロパティに指定された値をターゲット Webroutine に渡す時に使用するフィールド名です。フィールド名は単一引用符で囲みます。

詳しくは、currentrownumval プロパティの説明を参照してください。

省略値

'STDROWNUM'

有効値

単一引用符で囲まれた文字列。

例

この例では、ターゲット Webroutine に値を送る際に使用するフィールド名として、DEPTLINK を指定しています。ターゲット Webroutine で値を受け取るためには、その WEB_MAP に *BOTH または *INPUT の DEPTLINK フィールドがなければいけません。

currentrowhfield	'DEPTLINK'
------------------	------------

currentrownumval

currentrowhfieldプロパティに指定されるフィールドによりターゲットWebroutineに送られる値です。英数字フィールドの場合、値は単一引用符付きで指定する必要があります。数字フィールドの場合は、単一引用符はなくても構いません。

このプロパティは、currentrowhfieldプロパティと共に使用され、値をターゲットのWebroutineにどのように送信するかを記述します。以下の2つの情報が必要になります。

1. currentrowhfield: ターゲットWebroutineが情報を参照するために使用するフィールド名。
2. currentrownumval: リテラル値もしくは必要な情報を含むこの(ソース)Webroutineのフィールド名。

注：プロパティの名前は**currentrownumval** (numeric value: 数値) ではありませんが、currentrownumvalに指定するフィールド名は数値フィールドである必要はありません。

省略値

position()

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

reentryfield

reentryvalue プロパティに指定された値をWAMに送る時に使用するフィールド名です。フィールド名は単一引用符で囲みます。

詳しくは、reentryvalue プロパティの説明を参照してください。

注:このプロパティは、WEBEVENTアプリケーションでよく使用される、再入可能なプログラミング技術をサポートするために提供されています。最初からWAMを使用するように設計されたウェブ・アプリケーションでは、通常はこの技術を使用する必要はありません。

省略値

'STDREENTRY'

有効値

単一引用符で囲まれた文字列。

reentryvalue

ターゲットWebroutineのreentryfieldプロパティで指定されたフィールドに送信する値です。英数字フィールドの場合、値は単一引用符付きで指定する必要があります。数字フィールドの場合は、単一引用符はなくても構いません。

このプロパティは、reentryfieldプロパティと共に使用され、値をターゲットのWebroutineにどのように送信するか記述します。以下の2つの情報が必要になります。

1. reentryfield: ターゲットWebroutineが情報を参照するために使用するフィールド名。
2. reentryvalue: リテラル値もしくは必要な情報を含むこの(ソース)Webroutineのフィールド名。

注: このプロパティは、WEBEVENTアプリケーションでよく使用される、再入可能なプログラミング技術をサポートするために提供されています。最初からWAMを使用するように設計されたウェブ・アプリケーションでは、通常はこの技術を使用する必要はありません。

省略値

'D'

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

False() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

例

次の例はフィールド#STD_FLAGが 'X' と等しい場合にウェブレットを非表示にします。この式は以下のような形式で入力してください。

hide_if	#STD_FLAG = 'Y'
---------	-----------------

プロパティがフォーカスを失うと、この式は以下のように表示されます。

hide_if	key('field-value', 'STD_FLAG') = 'Y'
---------	--------------------------------------

formname

サーバーに送られるHTMLフォーム名です。

省略値

'LANSA'

有効値

単一引用符付きのフォーム名。プロパティ・シート内で該当するドロップダウン・ボタンをクリックすると、使用できるフォーム名リストが表示されます。

url

このプロパティを使って、ハイパーリンクの移動先のURLを指定します。指定する場合、URLにはリテラル値（例えば'http://www.mycompany.com/'など）または実行時にURLを含むフィールド名を指定できます。

このプロパティはon_click_wamname、 on_click_wrname及びprotocolプロパティよりも優先されます。urlが指定されている場合、この後のプロパティは無視されます。

省略値

'javascript:void();' - 何もないのと同じです。

有効値

単一引用符で囲まれたURL、実行時にURLが含まれるフィールド名、システム変数名、複数言語変数名。

on_click_wamname

このウェブレットを表わすハイパーリンクがクリックされた時に実行されるWebroutineのあるWAMの名前を指定します。(Webroutine名はon_click_wnameプロパティに指定されます。)

urlプロパティが指定された場合は、このプロパティは無視されます。

省略値

指定しない場合は、現在のWAMが使用されます。(\$lweb_WAMName)

有効値

単一引用符付きのWAM名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

on_click_wrname

このウェブレットを表わすハイパーリンクがクリックされた時に実行されるWebroutineの名前を指定します。(WebroutineのあるWAM名はon_click_wamnameプロパティで指定します。)

urlプロパティが指定された場合は、このプロパティは無視されます。

省略値

省略値は適用されません。urlプロパティまたはon_click_wrnameプロパティが設定されなければなりません。

有効値

単一引用符付きのWebroutine名。Webroutineは、on_click_wamnameプロパティに指定されたWAMに存在していなくてはなりません。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWebroutineのリストが表示されます。

例

次の例では、ハイパーリンクがクリックされた時に実行されるWebroutine名として、deptdetailを指定しています。このWebroutineがあるWAMの名前はon_click_wamnameプロパティに指定します。

on_click_wrname	'deptdetail'
-----------------	--------------

protocol

on_click_wnameプロパティに指定されたWebroutineに移動する際に使用されるプロトコル(例:http:// または https://)。

通常、このプロパティはセキュア・モードの処理との切り替えが必要な時に使用します。それ以外の場合は、通常このプロパティを指定する必要はありません。

省略値

ブランク。現在使用されているプロトコルが使用されます。

有効値

単一引用符付きの、有効なプロトコル。指定する場合は、通常は'http:'または'https:'です。

show_in_new_window

ウェブレットに応答したHTMLを新しいブラウザ・ウィンドウに表示するかどうかを決定するブール値のプロパティです。

省略値

false() - 応答のHTMLは現ブラウザ・ウィンドウに表示されます。

有効値

true()、false() もしくは有効な式。

target_window_name

ハイパーリンク先が表示されるウィンドウやフレームの名前。

省略値

空白 - ハイパーリンク先は現在のウィンドウに表示されます。

有効値

単一引用符で囲まれたウィンドウ名またはフレーム名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるウィンドウやフレームが表示されます。

pos_absolute_design

ウェブ・ページのウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置を指定しません。）

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

例

次の例では、[ポジションの固定]がウェブレットで選択されており、LANSAエディターの[デザイン]ビューで要求された位置にウェブレットが配置されています。この値が、pos_absolute_designプロパティに表示されます。

```
pos_absolute_design 'position:absolute;left: 324pt; top: 162.72pt;'
```

width_design

ウェブ・ページのウェブレットの幅です。

通常、ウェブレット幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これによりwidth-designプロパティの値が更新されます。ただし、必要に応じてプロパティ値を直接編集することもできます。

省略値

ブランク（ウェブレットはその省略値の幅を使用します。）

有効値

単一引用符で囲まれた有効な範囲内の幅。

relative-image-path

表示するイメージのイメージ仮想ディレクトリに関連付けられたパスとファイル名。

省略値

空白 - 何も表示されません。

有効値

イメージディレクトリに関連付けられ、単一引用符で囲まれたイメージのパスと名前。イメージは、プロパティ・シートで該当する省略記号のボタンをクリックし、プロンプターから選択することができます。

absolute-image-path

表示するイメージのパスとファイル名指定する場合、relative_image_path プロパティは空白でなければいけません。

省略値

空白 - 省略値はrelative_image_path プロパティに指定されたイメージを使用します。

有効値

単一引用符で囲まれたイメージのパスと名前。

class

ウェブレットのカスケード・スタイル・シート(CSS)クラス名です。

省略値

提供されているウェブレットのクラス名。例えば'std_anchor'などです。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

mouseover_class

マウスがその上に置かれた時の、ウェブレットのカスケード・スタイル・シート(CSS)クラス名。

省略値

このウェブレットに対して適用される省略値はありません。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・リストをクリックして、使用可能なクラスの一覧から選択できます。例えば、提供されているmouseoverクラスが選択されると、mouseover_classプロパティは'std_anchor_mouseover'にすることができます。

text_class

ウェブレットのテキストのカスケード・スタイル・シート(CSS)クラス名です。

省略値

提供されているウェブレットのテキスト・クラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

presubmit_js

ハイパーリンク先に移動する前に実行されるJavaScriptコードです。JavaScriptステートメントは必ずセミコロンで終了していなければなりません。

省略値

空白。JavaScriptは何も実行されません。

有効値

有効なJavaScriptステートメント。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

vf_wamevent

VLF の WAM イベント文字列です。

省略値

空白。

有効値

文字列。カンマ (,) は使用できません。

8.1.2 オートコンプリート (std_autocomplete)

クイック・スタート - オートコンプリート

プロパティ - オートコンプリート

オートコンプリート・ウェブレットはフィールドに入力している間にその候補を提供します。この候補はAjaxを使ってWebroutineにより提供されます。



A screenshot of a web form showing an autocomplete dropdown menu. The input field contains the letter 'S'. The dropdown menu is open, displaying a list of suggestions: 'Sally', 'Sarah', 'Scott', and 'Sean'. The 'Scott' option is currently selected and highlighted with a blue background.

クイック・スタート - オートコンプリート

オートコンプリート・ウェブレットを使用するには、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[jQuery UI オートコンプリート]ウェブレットを見つけてください。
2. このウェブレットを自身のページにドラッグ・アンド・ドロップしてください。ウェブレットを選択して、[詳細]タブをクリックするのを忘れないでください。必要なプロパティを入力します。
3. デザイナでオートコンプリート・ウェブレットを選択した状態で右クリックし、コンテキスト・メニューからAjaxWebroutineを作成するオプションを選択します。
4. RDMLXソースで（ステップ3で作成した）応答Webroutineのコードを完成させます。

プロパティ - オートコンプリート

cache	labelField	scroll
class	listName	scrollHeight
delay	matchContains	size
disabled	maxLength	sourceWamName
display_length	minLength	sourceWrName
display_mode	namevalue	tab_index
extraFields	onchange_script	termField
height	onselect_script	title
hide_if	pos_absolute	valueField
keyboard_shift	read_only	width

name

ウェブレットを識別する名前です。ウェブレットがフィールドをビジュアライズすると、これがフィールド名になります。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用することも可能です。

省略値

ウェブレットによりビジュアライズされたフィールドの場合、省略値の名前はフィールド名、もしくはフィールド名と行番号を組み合わせたもの(リストのフィールドの場合)になります。それ以外は、省略値の名前は自動的に生成された、一意の識別子になります。

Valid values

単一引用符で囲まれた文字列。

value

ウェブレットに設定する値。ウェブレットによりフィールドがビジュアライズされる場合、この値により表示されるフィールドが識別されま

省略値

省略値は適用されません - このウェブレットを使用する多くの場合、入力ボックスに表示される値のフィールドを指定するか、ブランクのままにして、ユーザーが提案する値を入れられるようにする必要があります。

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

display_mode

ウェブレットが入力を受け取るかどうか、出力を表示するかどうかを制御します。ウェブレットが入力モードの場合はオートコンプリートでのみ作動します。

省略値

ブランク ('input'の意味です)。

有効値

リテラル値'input'、'output'または'hidden'。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できる値のリストが表示されます。もしくは、実行時に使用できる値を含むフィールド名、システム変数名、または複数言語変数名を入力することもできます。

maxlength

ユーザーがウェブレットに入力できる文字の最大数を指定します。ウェブレットがフィールドをビジュアルライズする時、そのフィールドに適した数がここに設定されます。

省略値

ブランク (ウェブレットはユーザーが入力できる文字数を制限しません)。

有効値

数値。

size

文字/バイト数で表された、ウェブレットのデータのサイズ。

このプロパティは現在使われていません。 - 代わりに、maxlength及び/もしくはdisplay_lengthプロパティを使用してください。

display_length

文字数で表された、ウェブレット入力ボックスの適切なサイズ。ブラウザはここに指定された文字数により入力ボックスのサイズ調整をします。widthプロパティが指定されている場合、そちらが優先され、display_lengthプロパティは無視されます。

省略値

blank (ウェブレットは省略値サイズを選択します)。

有効値

数値。

keyboard_shift

入力フィールドのキーボード・シフト。

省略値

このウェブレット・ビジュアルイゼーションのフィールドのキーボード・シフト。それ以外はブランク。

有効値

Char、Stringデータ・タイプの場合：'、'W'、'J'、'E'、'O' 及び 'U'

Alphaデータタイプの場合：'、'X'、'A'、'N'、'W'、'T'、'D'、'M'、'J'、'E' 及び 'O'

キーボード・シフトは現在DBCSフィールドの妥当性検査のためだけに使用されています。
--

minLength

オートコンプリートが有効になる前にユーザーが入力する必要のある最低限の文字数。項目が少ないローカル・データの場合は、ゼロを使用すると便利です。項目が多く、1文字で何千もの項目が一致するような場合は、文字数を増やす必要があります。

省略値

1

有効値

数値。

delay

キー入力後にオートコンプリートが有効になるまでの時間（ミリ秒）。ローカル・データの場合はゼロを指定すると効果ありますが（反応が良い）、リモート・データの場合は反応も悪くなり、大きな負荷をかける可能性があります。

省略値

300ミリ秒

有効値

ミリ秒単位の数値。

sourceWamName

このウェブレットの応答データを提供するWebroutineがあるWAMの名前。

省略値

現在のWAM

有効値

単一引用符付きのWAM名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

sourceWrName

このウェブレットの応答データを提供するWebroutineの名前。

省略値

ブランク - Webroutine名の指定は必須です。

有効値

単一引用符付きのWebroutine名。このWebroutineはJSON応答Webroutineでなければなりません。またsourceWamNameプロパティ内に指定されたWAM内に存在している必要があります。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるJSONWebroutineのリストが表示されます。

termField

オートコンプリート・フィールドの現在の値を受け取る、応答処理Webroutine内のフィールドの名前。

省略値

なし。必須フィールドです。

有効値

単一引用符で囲まれたテキスト・フィールド。フィールドは、応答処理Webroutine内で*INPUT のWEB_MAPになければなりません。

listName

[任意]候補のリストを保管する応答Webroutine内のリスト名。空の場合は、最初のリスト（応答Webroutine内の唯一のリストの**はず**です。）が使用されます。

省略値

ブランク。応答Webroutineの最初のリストが使用されます。

有効値

単一引用符で囲まれたテキスト。リストは、応答処理Webroutine内で*OUTPUTのWEB_MAPになければなりません。

labelField

応答データはラベルまたは値のいずれかのカラム、もしくは両方が付いているリストです。ラベルのカラムは候補メニュー内に表示されます。値はユーザーがこのメニューから何かを選択した後に入力エレメントに挿入されます。カラムが1つだけ指定されている場合は、両方に使用されます。例えばvalueプロパティのみが提供されている場合は、これがラベルとしても使用されます。

省略値

空白。

有効値

単一引用符で囲まれたテキスト。応答処理Webroutineにより返されるリストに存在するカラム名。

valueField

応答データはラベルまたは値のいずれかのカラム、もしくは両方が付いているリストです。ラベルのカラムは候補メニュー内に表示されます。値はユーザーがこのメニューから何かを選択した後に入力エレメントに挿入されます。カラムが1つだけ指定されている場合は、両方に使用されます。例えばvalueプロパティのみが提供されている場合は、これがラベルとしても使用されます。

省略値

空白。

有効値

単一引用符で囲まれたテキスト。応答処理Webroutineにより返されるリストに存在するカラム名。

extraFields

[任意]このウェブレットの応答を提供するWebroutineに送信するフィールドをカンマで区切ったリスト。

省略値

空白。追加フィールドはありません。

有効値

フィールドをカンマで区切ったリスト。フィールドは入力または非表示のフィールドでなければなりません。

cache

trueに設定されると、サーバー応答をローカルに保存し、ユーザーが文字を更に入力するにつれて候補を絞ります。

省略値

True.

有効値

true()、false() もしくは有効な式。

matchContains

trueに設定されると、オートコンプリートは入力された文字と一致するものがどこかに含まれているものを候補にします。それ以外は、入力された文字で始まるものだけを候補にします。これはキャッシュ応答の絞られた候補にも適応されます。データ・ソースにはこのプロパティと一貫性のとれた選択条件がなければなりません。

省略値

False.

有効値

true()、false() もしくは有効な式。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

False() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

class

ウェブレットのカスケード・スタイル・シート(CSS)クラス名です。

省略値

提供されているウェブレットのクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

title

ウェブレットの上をマウスが通った時に、ヒントのテキストとして表示される、ウェブレットのタイトルを指定してください。

省略値

空白 - ヒントは何も表示されません。

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

read_only

ウェブレットのコンテンツを読取専用(ユーザーはコンテンツの変更ができません)にするかどうかを決定するブール値。

省略値

ブランク - Falseと同じです。(ユーザーはコンテンツを変更できません。)

有効値

true()、false() もしくは有効な式。

例

次の例はフィールド#STD_FLAGが 'Y' と等しい場合にウェブレットを読取専用を設定します。この式は以下のような形式で入力してください。

```
read_only #STD_FLAG = 'Y'
```

プロパティがフォーカスを失うと、この式は以下のように表示されません。

```
read_only key('field-value', 'STD_FLAG') = 'Y'
```

disabled

ウェブレットを使用可または不可として表示するかを決定するブール値。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false() もしくは有効な式。

pos_absolute

ウェブ・ページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（ウェブレットは相対的に位置付けられます。）

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

ウェブ・ページのウェブレットの幅です。

通常、ウェブレット幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これでwidth プロパティやheightプロパティの値が更新されます。ただし、必要に応じてプロパティ値を直接編集することもできます。

省略値

ブランク (ウェブレットは省略値の幅を適用します。)

有効値

単一引用符で囲まれた有効な範囲内の幅。

height

ウェブ・ページのウェブレットの高さ。

通常、ウェブレットの高さや幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これでwidthプロパティやheightプロパティの値が更新されます。ただし、必要に応じてプロパティの値を直接編集することもできます。

省略値

ブランク（ウェブレットは省略値の高さを適用します。）

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

scroll

結果がscrollHeightで構成した範囲より多かった場合にスクロールさせるかどうかを決定します。

省略値

False.

有効値

true()、false() もしくは有効な式。

scrollHeight

オートコンプリートの候補を表示するボックスの高さ。scrollにtrueが設定されている時のみ有効です。

省略値

180px.

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

onchange_script

jQuery UI オートコンプリート・ウィジェットのchange イベント(テキストが変更された後、入力ボックスがフォーカスを失った時)に紐付けされるJavaScriptコード。これにはオプションでeventとuiのオブジェクト・パラメータがあります。例: myOnChangeEventHandler(event, ui)

省略値

空白。JavaScriptは何も実行されません。

有効値

有効なJavaScriptステートメント。

onselect_script

jQuery UI オートコンプリート・ウィジェットのselect イベント(ユーザーがオートコンプリートで提案されたリストから項目を選択した時)に紐付けされるJavaScriptコード。これにはオプションでeventとuiのオブジェクト・パラメータがあります。例: myOnSelectEventHandler(event, ui)

省略値

空白。JavaScriptは何も実行されません。

有効値

有効なJavaScriptステートメント。

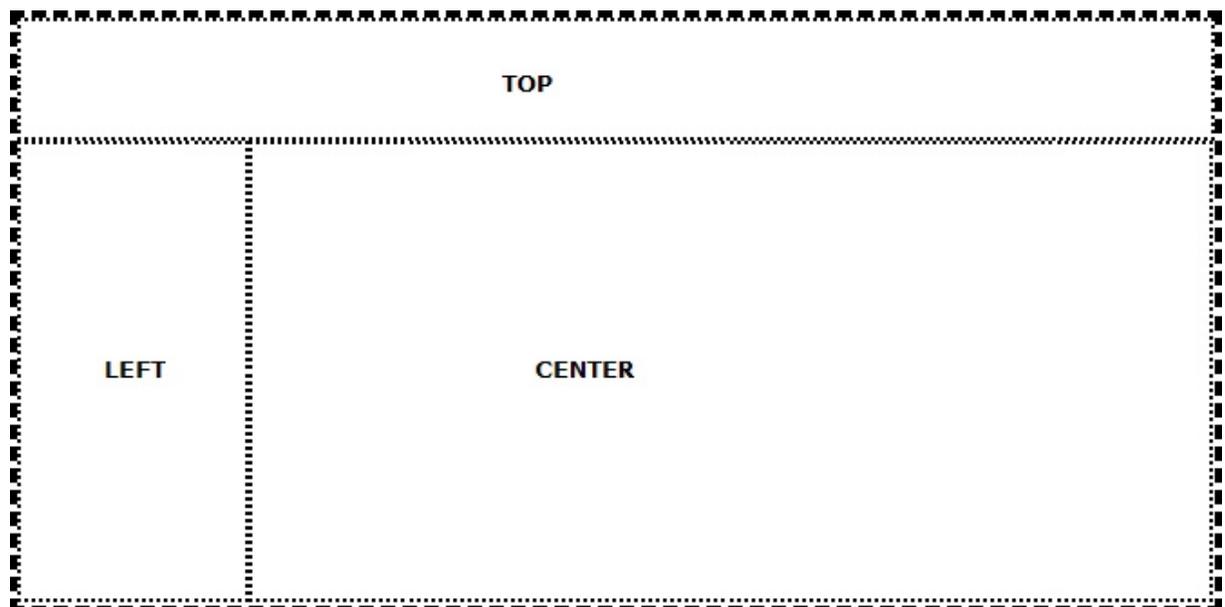
8.1.3 アタッチメント・パネル (std_attachment_panel)

クイック・スタート - アタッチメント・パネル

プロパティ - アタッチメント・パネル

アタッチメント・パネル・ウェブレットはコンテンツをドロップできる5つの領域(左、上、右、中、下)を持つパネルを提供します。それぞれの領域ごとのアタッチメント・レイアウト・マネージャがあります。この5つの領域には、コンテンツを挿入したり、他のウェブレットをドロップしたりできます。ウェブレットをドロップすると、アタッチメント・レイアウト・マネージャのルールに従って、ウェブレットのサイズが調整されます。

以下の例はほぼ空の状態のアタッチメント・パネルの外観を示しています。この例では、3つのアタッチメント領域のみが使用されています。太い破線で示された枠線はアタッチメント・パネルとして指定されており、細い点線の枠線で示されたパネルは3つの領域で使用されています。このサンプルでは、分かりやすいように枠線が使用されていますが、このように実線を使う必要はなく、アプリケーションでも必要ないことが多いでしょう。他のウェブレット(インプット・ボックス、チェック・ボックス、プッシュ・ボタンなど)もそれぞれのレイアウト領域にドロップできることを覚えておいてください。



一貫性があり、しかも魅力的なWebページのレイアウトの作成ができるよう提供されている多くのウェブレットの1つがアタッチメント・パネルです。水平スプリッターや垂直スプリッター、パネルとナビゲーション

ン・パネルのウェブレットも参照してください。アタッチメント・パネルは静的なウェブレットです。つまり、ユーザーはこれが含まれるパネルのサイズ変更やサイズ調整、位置変更をすることができません。

アタッチメント・パネルのバージョン2では、ブラウザの互換性の問題に対応するため、border とborder_widthプロパティがborderプロパティ1つに置き換えられ、class_top、class_left、class_center、class_rightおよびclass_bottomプロパティは削除されました。クラスの値は"ペイン"のカスタム・プロパティ・エディターを使って設定できるようになっています。また、このエディターではペインのサイズや配置も設定することができます。

クイック・スタート - アタッチメント・パネル

アタッチメント・パネルを使用するには、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[アタッチメント パネル]ウェブレットを探します。
2. このウェブレットを自身のページにドラッグ・アンド・ドロップしてください。ウェブレットを選択して、[詳細]タブをクリックするのを忘れないでください。必要に応じてアタッチメント・パネルのプロパティ、例えばbordersなどを設定します。
3. これで希望のペインもしくはレイアウト領域に内容をドラッグ・アンド・ドロップあるいは挿入することができるようになります。使用したい5つのレイアウト領域のそれぞれに[パネル]ウェブレットをドラッグ・アンド・ドロップしてください。その後これらのパネルのサイズを調整し、この子パネルの上に他のウェブレットを挿入するとより簡単です。

プロパティ - アタッチメント・パネル

アタッチメント・パネル・ウェブレットのプロパティは以下のとおりです。

border	panes
height	pos_absolute
hide_if	width
name	

name

ウェブレットを識別する名前です。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用することも可能です。

省略値

自動的に生成された、一意の識別子。

有効値

単一引用符で囲まれた文字列。

panes

表示するペインー式およびそのプロパティを指定するXMLノードセット。これは、アタッチメント・パネルをデザインビュー上にドラッグした時に、システムにより生成される値です。設定はデザイナーでのみ可能です。デザイナーを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使用してください。

省略値

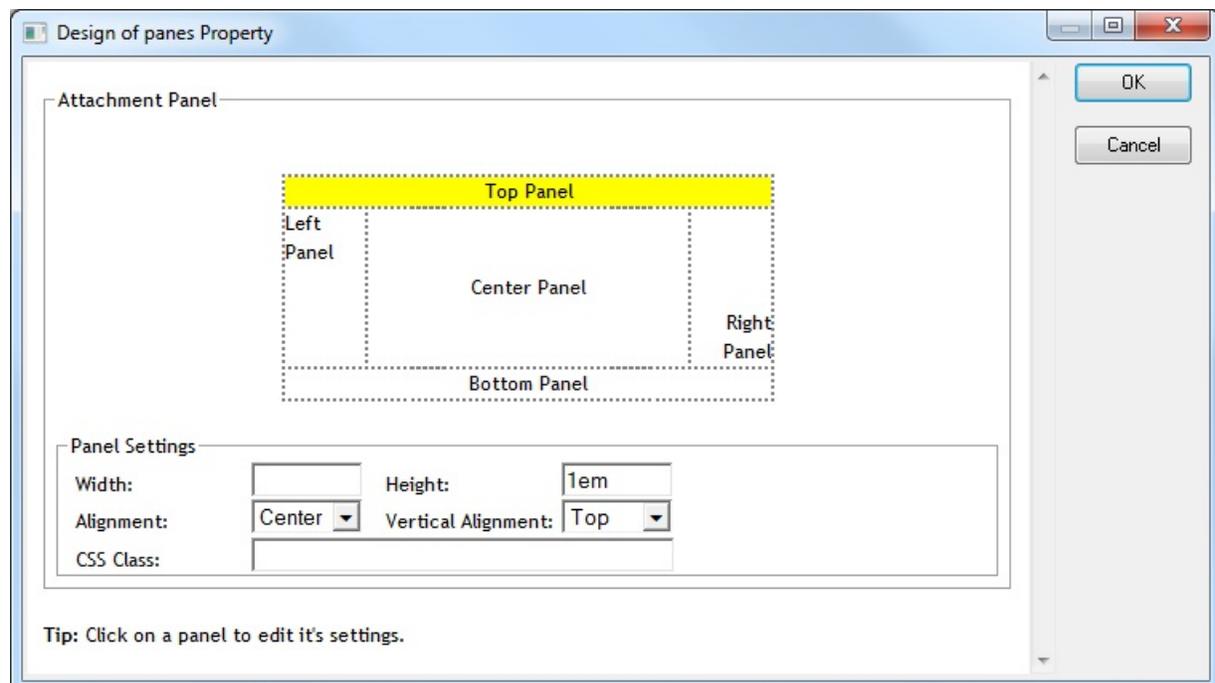
document(")/*//xml:data/xml:panes[@id='<unique id>'] (一意の識別子 unique idが自動的に生成された現在のペインです。)

有効値

適用不可。(この値はシステムにより保守されます。)デザイナーを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使用してください。

例

プロパティの省略記号 (...) のボタンを使って、デザイナーを開きます。



デザイナーにより、各ペインの次のプロパティが編集できます。

- [Width]/[Height]:有効なCSS値でなければなりません。

- [Alignment]/[Vertical alignment]:有効な値はドロップダウンに表示されます。
- [CSS Class]:ペインに適用するCSSクラスです。

border

ウェブレットの外側枠線のCSSのborder値。例：'1px dashed red'.

省略値

ブランク(境界は表示されません。)

有効値

有効なCSSのborder文字列。これは幅、スタイル、色で構成され、それぞれがスペースで区切られます。このリストからプロパティを省略することもできますが、順序は上記の順でなければなりません。例えば、"solid #ff0000"は有効な値です。CSSのborderプロパティに関する詳細は、「[W3C仕様](#)」を参照してください。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

False() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

pos_absolute

ウェブ・ページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

例

次の例では、[ポジションの固定]がウェブレットで選択されており、LANSAエディターの[デザイン]ビューで要求された位置にウェブレットが配置されています。これが、pos_absolute_designプロパティに表示される値になります。

```
pos_absolute_design 'position:absolute;left: 324pt; top: 162.72pt;'
```

width

ウェブ・ページのウェブレットの幅です。

通常、ウェブレットの高さや幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。こうすることで、width-design プロパティやheight_designプロパティの値を更新します。ただし、必要に応じてプロパティの値を直接編集することもできます。

省略値

" (アタッチメント・パネルが構成要素の中で可能な最大幅を使用するよう指定します。)

有効値

単一引用符で囲まれた有効な範囲内の幅。

height

ウェブ・ページのウェブレットの高さ。

通常、ウェブレットの高さや幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。こうすることで、width-design プロパティやheight_design プロパティの値を更新します。ただし、必要に応じてプロパティの値を直接編集することもできます。

省略値

'250pt'

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

8.1.4 バナー (std_banner)

クイック・スタート-バナー プロパティ -バナー

バナー・ウェブレットは基本的にはマーキー、つまり Web ページ上どのペインにでも追加できる、スクロール文字のエリアです。バナーは `<marquee>` HTML エlement として組込まれます。

バナーは情報を表示するために利用できる他、クリック時に他の Web ページにリダイレクトする Web ページ内のアクティブな Element にすることもできます。通常は広告や定期的に変更される最新情報の提供に使用されます。

Firefox では、サイズが固定で、"scroll" の値が "scroll" か "slide"、そしてテキスト配置が "center" で (テーブル・セルや添付パネルなどの) 位置づけられるバナーの場合の表示方法にバグがあります。Firefox はバナーの "box" を正しい場所に配置しますが、バナーが左整列であると仮定してコンテンツの位置を計算します。ですから、コンテンツがボックスの場所と重なる位置にある場合のみ表示されます。

クイック・スタート- バナー

バナーの設定は非常に簡単です。LANSAエディターでWebroutine用に生成されたXSLを開き、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[バナー]ウェブレットを見つけてください。
2. バナー ウェブレットを、Webページ上のバナーを表示したい位置にドラッグします。
3. ウェブレットをクリックして、[詳細]タブを確認します。
4. [value]プロパティに、マーカーに表示するテキストを含む適切なフィールド、システム変数、複数言語変数、もしくはリテラルのテキストを設定します。
5. テキストをページ内のアクティブなエレメントにしたい場合は、on_click_wnameプロパティもしくはURLプロパティに、適切なページへのクリック・イベントを指定します。

プロパティ - バナー

バナーウェブレットのプロパティは以下のとおりです。

class	pos_absolute	scroll_direction
disabled	presubmit_js	scroll_loop_count
formname	protocol	scroll_true_speed
height	reentryfield	show_in_new_window
hide_if	reentryvalue	target_window_name
name	scroll	URL
on_click_wamname	scroll_amount	value
on_click_wrname	scroll_delay	width
panes		

name

バナーの名前。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やバナーを参照するXSLを使用する場合は、独自の名前を使用しても構いません。

省略値

自動的に生成された、一意の識別子。

有効値

単一引用符で囲まれた文字列。

例

次は省略値の名前を示しています。

```
name concat('o', position(), '_LANSA_13186')
```

もしくは、以下のような一意の名前を入力することもできます。

```
name 'ABC_Banner'
```

value

バナー上に表示されるテキスト文字列

省略値

空白。

有効値

単一引用符で囲まれたテキスト、もしくは、複数言語対応テキスト変数、システム変数、フィールドの名前(プロパティ・シートの関連する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

panes

表示するペインのセットを指定するXMLノードセット。これは、バナーをデザインビュー上にドラッグしたときに、システムが生成する値です。

注：この値は変更できません。参照のみです。

省略値

`document(")/*//lxml:data/lxml:panes[@id='<unique id>']`（一意の識別子 `unique id` が自動的に生成された現在のペインです。）

有効値

適用不可。(この値はシステムにより保守されます。)

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

False() (バナーは常に表示されます)

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false() で答えが得られる有効な式。

例

次の例はフィールド#STD_FLAGが 'X' と等しい場合にバナーを非表示にします。式は入力されなくてはなりません。またプロパティにフォーカスがある時、以下のように表示されます。

hide_if	#STD_FLAG = 'X'
---------	-----------------

プロパティがフォーカスを失うと、この式は以下のように表示されま

hide_if	key('field-value', 'STD_FLAG') = 'X'
---------	--------------------------------------

reentryfield

reentryvalue プロパティに指定された値をWAMに送る時に使用するフィールド名です。フィールド名は単一引用符で囲みます。

省略値

空白。

有効値

リポジトリもしくはWAMで定義されたフィールド名。プロパティシート内で該当するドロップダウン・ボタンをクリックすると、使用できるフィールド名リストが表示されます。

reentryvalue

reentryfieldプロパティに指定されたフィールドに送られる値。英数字フィールドの場合、値は単一引用符付きで指定する必要があります。数字フィールドの場合は、単一引用符はなくても構いません。

省略値

ブランク。

有効値

適切なリテラル。

formname

サーバーに送られるHTMLフォーム名です。

省略値

'LANSA' (document.LANSAのことです。)

有効値

単一引用符付きのフォーム名。プロパティ・シート内で該当するドロップダウン・ボタンをクリックすると、使用できるフォーム名リストが表示されます。

disabled

バナーを使用可または不可として表示するかを決定するブール値。

MARQUEE タグのdisabled属性はInternet Explorerでのみサポートされています。バナーで使用不可に指定された場合、バナーがマウスでクリックされると全ブラウザで無視されるようになってます。ただし、disabled属性の"灰色表示"効果はInternet Explorerでのみ有効です。

省略値

ブランク - Falseと同じです。（バナーは常に使用可能な状態）

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

例

以下の例では、フィールドSTD_FLAGの値が'X'の時、バナーが使用不可になります。

```
disabled key('field-value', 'STD_FLAG') = 'X'
```

URL

バナーがクリックされたときに移動するURLを指定します。プロトコル (例 : http:// や https://) で始まらなくてははいけません。Webroutineが on_click_wname プロパティで指定されている場合は、このプロパティに入力してはいけません。

省略値

空白。

有効値

単一引用符で囲まれたURL。

on_click_wamname

バナーがクリックされたときに呼び出されるWAMの名前。URLプロパティにURLが指定されている場合は、このプロパティは入力してはいけません。

省略値

\$lweb_WAMName (現在のWAMを示します。)

有効値

単一引用符付きのWAM名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

on_click_wrname

バナーがクリックされたときに呼び出されるWebroutineの名前。URLプロパティにURLが指定されている場合は、このプロパティは入力してはいけません。ただし、on_click_wamnameが入力されている場合は入力しなくてはいけません。

省略値

空白。

有効値

単一引用符付きのWebroutine名。Webroutineは、on_click_wamnameプロパティに指定されたWAMに存在してはいけません。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWebroutineのリストが表示されます。

protocol

on_click_wnameプロパティに指定されたWebroutineに移動する際に使用されるプロトコル(例:http:// または https://)。

省略値

空白。現在使用されているプロトコルが使用されます。

有効値

単一引用符で囲まれた、コロンで終了する有効なプロトコル。通常は'http:'または'https:'です。

show_in_new_window

バナーのクリック時の応答HTMLを新しいブラウザ・ウィンドウに表示するかどうかを決定するブール値。

省略値

false() - 応答HTMLは現ブラウザ・ウィンドウに表示されます。

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

target_window_name

応答HTMLが表示される、ウィンドウやフレームの名前。

省略値

ブランク - 応答HTMLが現在のウィンドウに表示されます。

有効値

単一引用符付きウィンドウ名またはフレーム名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるウィンドウやフレームが表示されます。

pos_absolute

ウェブ・ページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

ウェブ・ページのウェブレットの幅です。

省略値

blank (ウェブレットはその省略値の幅を使用します。)

有効値

単一引用符で囲まれた有効な範囲内の幅。

height

ウェブ・ページのウェブレットの高さ。

省略値

ブランク(ウェブレットはそのデフォルトの高さを使用します。)

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

class

バナーに適用されるカスケード・スタイル・シートのクラス。

省略値

'std_banner'。バナーの省略値のクラスで、全てのカスケード・スタイル・シートと共に提供されます。

有効値

現在のカスケード・スタイル・シートから選択された有効なクラス名を単一引用符で囲んだもの。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

scroll

テキストがマーカーの中でどのようにスクロールするかを制御します。

省略値

ブランク - スクロールします。

有効値

ブランク、もしくは、'alternate'、'scroll'、'slide'

'Scroll' - バナーのコンテンツは`scroll_direction`プロパティで指定された方向にスクロールします。テキストはバナーの端までスクロールしたら、もう一度はじめからスクロールします。

'Slide' - バナーのコンテンツは`scroll_direction`プロパティで指定された方向にスクロールします。テキストはバナーの端までスクロールして、止まります。Firefoxでは、アニメーションは停止しないので注意してください。この場合テキストは'slide'も'scroll'と同じ動作になり、バナーの端までスクロールしてもう一度最初から繰り返します。

'Alternate' - コンテンツがバナーの端に達すると、`scroll_direction`の方向が反対になります。

scroll_direction

テキストが流れる方向を制御します。

省略値

ブランク - Leftと同じで、右から左の方向になります。

有効値

Blank もしくは 'down'、'left'、'right'、または 'up'。

'Down' - 上から下の方向です。

'Left' - 右から左の方向です。

'Right' - 左から右の方向です。

'Up' - 下から上の方向です。

scroll_loop_count

バナーを再生する回数を制御します。

バナーが`scroll_loop_count`の回数分再生された後、`scroll`プロパティの値により、異なる"終了状態"になります。"scroll"の場合、バナーは空白になります。"slide"や"alternate"の場合、テキストは終了時の位置に表示されたままです。

`scroll_loop_count`プロパティは次のような場合は無視されます。

- Firefoxブラウザ。
- `scroll`プロパティが'slide'の全てのブラウザ。

省略値

空白。

有効値

空白、-1、もしくは整数値。

`scroll`プロパティが空白、"scroll"、または"alternate"の時に、空白もしくは-1を指定すると、バナーは無限にループします。

scroll_amount

テキストが後続のウェブレットの呼び出しの間に動くピクセル数を制御します。

省略値

空白。ブラウザごとに独自の省略値を設定できます。省略値はブラウザによりわずかに異なりますが、6ピクセル前後のはずです。

有効値

整数値。

scroll_delay

ウェブレットの更新ごとの待ち時間をミリ秒で指定することで、スクロールのスピードを制御します。

クライアントCPUの負荷が大きくなり過ぎないように、ブラウザは60ミリ秒以下の値は全て自動的に切り上げて60にします。 *scroll_true_speed* パラメータをInternet Explorerで使用することで、この数値を上書きして *scroll_delay* に指定された値を使用できます。

省略値

空白。ブラウザごとに独自の省略値を設定できます。省略値はブラウザによりわずかに異なりますが、85ミリ秒前後のはずです。

有効値

整数値。

scroll_true_speed

クライアントCPUの負荷が大きくなり過ぎないように、ブラウザは60ミリ秒以下の`scroll_delay`の値は全て自動的に切り上げて60にします。`scroll_true_speed`パラメータは、この数値を上書きして`scroll_delay`に指定された値を使用できます。

`Scroll_true_speed`は現在Internet Explorerでのみサポートされています。このため、`scroll_delay`には常に60またはそれ以上の値を使用して`scroll_amount`を増やすようにし、必要なスピードにするのがより現実的です。

省略値

False()

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数を使用した、true() もしくは false() で答えが得られる有効な式。

例

次の例は、`#STD_FLAG`が'X'に等しいという式の結果を元に決定されたスクロールのプロパティを使用します。式は入力されなくてはなりません。またプロパティにフォーカスがある時、以下のように表示されます。

```
scroll_true_speed #STD_FLAG = 'X'
```

プロパティがフォーカスを失うと、この式は以下のように表示されます。

```
scroll_true_speed key('field-value', 'STD_FLAG') = 'X'
```

presubmit_js

フォームの送信前に実行されるJavaScriptコード。

省略値

ブランク。JavaScriptは何も実行されません。

有効値

有効なJavaScriptファンクション、もしくは最後にセミコロン(;)が付いたJavaScriptコード。

この要求を送信するJavaScriptを実行せずに、事前に送信されたJavaScriptのみを実行したい場合は(こうすることでonclickイベントがキャンセルされます)、事前に送信されたJavaScriptに**return false;**を追加します。

例

以下の例はメッセージ・ボックスを表示します。

```
presubmit_js      'alert("Hello world!");'
```

次の例はメッセージ・ボックスを表示して、JavaScriptの送信をキャンセルします。

```
presubmit_js      'alert("Hello world!"); return false;'
```

8.1.5 プッシュ・ボタン (std_button_v2) および イメージ付き プッシュ・ボタン (std_image_button_v2)

クリック・スタート - プッシュ・ボ
タンとイメージ付きプッシュ・ボタ
ン プロパティ - プッシュ・ボタ
ンとイメージ付きプッシュ・ボタ
ン

プッシュ・ボタン・ウェブレットは、Webページにテーマのあるプッシュ・ボタンを提供します。例えば次のように表示されます。



クリック・スタート - プッシュ・ボタンとイメージ付きプッシュ・ボタン

Webページにプッシュ・ボタンを追加するには、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[Push Button]ウェブレットを探します。
2. 必要なウェブレットをWebページにドラッグ・アンド・ドロップしてください。[詳細]タブをクリックします。
3. ボタンに表示されるテキストをcaption に設定します。イメージ付きのプッシュ・ボタンの場合は、image プロパティに適切な値を設定します。左側のイメージと右側のイメージが設定できます。
4. ボタンがクリックされたときに呼び出されるWebroutineの名前をon_click_wrnameプロパティに設定します。このWebroutineが現在のWebroutineと異なるWAMにある場合は、on_click_wamnameプロパティも設定しなければなりません。

プロパティ - プッシュ・ボタンとイメージ付きプッシュ・ボタン

以下のプロパティは、"*std_image_button_v2*のみ"と明記されているもの以外はすべて両方のボタン・ウェブレットに共通です。

caption	left_absolute_image_path	right_image_height
currentrowhfield	left_image_height	right_relative_image_path
プロパティ - 大型リスト	left_relative_image_path	show_in_new_window
confirmText	name	submitExtraFields
currentrownumval	on_click_wamname	tab_index
default_button	on_click_wrname	target_window_name
disabled	pos_absolute_design	text_class
formname	presubmit_js	title
height_design	protocol	vf_wamevent
hide_if	right_absolute_image_path	width_design

name

ウェブレットの名前。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript や ウェブレットを参照するXSLを使用する場合は、独自の名前を使用することも可能です。

省略値

`concat('o', position(), '_LANSA_n')` - LANSAにより付けられたウェブレット内部名です。

有効値

単一引用符で囲まれた名前。

caption

ウェブレットのキャプション。

省略値

'Caption'

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

currentrowhfield

currentrownumvalプロパティに指定された値をターゲットWebroutineに渡す時に使用するフィールド名です。フィールド名は単一引用符で囲みます。

詳しくは、currentrownumvalプロパティの説明を参照してください。

省略値

'STDROWNUM'

有効値

単一引用符で囲まれた文字列。

currentrownumval

currentrowhfieldプロパティに指定されるフィールドによりターゲットWebroutineに送られる値です。英数字フィールドの場合、値は単一引用符付きで指定する必要があります。数字フィールドの場合は、単一引用符はなくても構いません。

このプロパティは、currentrowhfieldプロパティと共に使用され、値をターゲットのWebroutineにどのように送信するかを記述します。以下の2つの情報が必要になります。

1. currentrowhfield: ターゲットWebroutineが情報を参照するために使用するフィールド名。
2. currentrownumval: リテラル値もしくは必要な情報を含むこの(ソース)Webroutineのフィールド名。

注：プロパティの名前は**currentrownumval** (numeric value: 数値) ではありませんが、currentrownumvalに指定するフィールド名は数値フィールドである必要はありません。

省略値

position()

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

left_relative_image_path

std_image_button_v2 のみ。

ウェブレットの左に表示するイメージの、イメージディレクトリに関連付けられたパスと名前。指定する場合は、`left_absolute_image_path` プロパティは空白でなければなりません。

省略値

'ball_red.gif'

有効値

イメージのイメージディレクトリに関連付けられたパスおよび名前で単一引用符で囲まれたもの。イメージは、プロパティ・シートで該当する省略記号のボタンをクリックし、プロンプターから選択することができます。

left_absolute_image_path

std_image_button_v2 のみ。

ウェブレットの左に表示されるイメージのパスと名前。指定する場合は、`left_relative_image_path` プロパティは空白でなければなりません。

省略値

空白 - 省略値は `left_relative_image_path` プロパティに指定されたイメージを使用します。

有効値

単一引用符で囲まれたイメージのパスと名前。

left_image_height

std_image_button_v2 のみ。

ウェブレットの左側のイメージの高さ。

省略値

'12pt'

有効値

単一引用符で囲まれた、有効な範囲の高さ。

right_relative_image_path

std_image_button_v2 のみ。

ウェブレットの右に表示するイメージの、イメージディレクトリに関連付けられたパスと名前。指定する場合は、`right_absolute_image_path` プロパティは空白でなければなりません。

省略値

空白 - 省略値では、ボタンの右側にイメージは表示されません。

有効値

イメージのイメージディレクトリに関連付けられたパスおよび名前で単一引用符で囲まれたもの。イメージは、プロパティ・シートで該当する省略記号のボタンをクリックし、プロンプターから選択することができます。

right_absolute_image_path

std_image_button_v2 のみ。

ウェブレットの右に表示されるイメージのパスと名前。指定する場合は、`right_relative_image_path`プロパティは空白でなければなりません。

省略値

空白 - 省略値では、`right_relative_image_path`プロパティが指定されている場合、ここに指定されたイメージが使用されます。

有効値

単一引用符で囲まれた、イメージのパスと名前。

right_image_height

std_image_button_v2 のみ。

ウェブレットの右側のイメージの高さ

省略値

'12pt'

有効値

単一引用符で囲まれた、有効な範囲の高さ。

submitExtraFields

onClickWebroutineに送信する追加のフィールド（送信されたフォームに存在しないもの）を指定したXMLノードセット。これは通常ウェブレットがリストやグリッドで使用される時によく使用され、リストの別のカラムからの値を指定します。

省略値

document(")/*//xml:data/xml:json[not(@id)](このウェブレットに定義された項目がないことを示します。)

有効値

適用不可。(この値はシステムにより保守されます。)デザイナを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使用してください。

例

次の例ではsubmitExtraFieldsプロパティ・エディターが示されています。



ここでは現在のWebroutineの出力フィールド（[値]欄）を、onClickWebroutineのWEB_MAPで定義された異なる名前（[名前]欄）の入力フィールドにマップする方法を示しています。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

False() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

formname

サーバーに送られるHTMLフォーム名です。

省略値

'LANSA'

有効値

単一引用符付きのフォーム名。プロパティ・シート内で該当するドロップダウン・ボタンをクリックすると、使用できるフォーム名リストが表示されます。

pos_absolute_design

ウェブ・ページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width_design

ウェブ・ページのウェブレットの幅です。

省略値

ブランク（ウェブレットはその省略値の幅を使用します。）

有効値

単一引用符で囲まれた有効な範囲内の幅。

height_design

ウェブ・ページのウェブレットの高さ。

省略値

ブランク(ウェブレットは省略値の高さを使用します。)

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

on_click_wamname

ウェブレットがクリックされたときに呼び出されるWAMの名前。

省略値

\$lweb_WAMName (現在のWAMを示します。)

有効値

単一引用符付きのWAM名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

on_click_wrname

ウェブレットがクリックされたときに呼び出されるWebroutineの名前。

省略値

ブランク - Webroutine名の指定は必須です。

有効値

単一引用符付きのWebroutine名。Webroutineは、on_click_wamnameプロパティに指定されたWAMに存在してはいけません。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWebroutineのリストが表示されます。

protocol

on_click_wnameプロパティに指定されたWebroutineに移動する際に使用されるプロトコル(例:http:// または https://)。

省略値

空白。現在使用されているプロトコルが使用されます。

有効値

単一引用符付きの、有効なプロトコル。通常は'http:'または'https:'です。

show_in_new_window

ウェブレットの応答HTMLを新しいブラウザ・ウィンドウに表示するかどうかを決定するブール値プロパティ。

省略値

false() - 応答HTMLは現ブラウザ・ウィンドウに表示されます。

有効値

true()、false() もしくは有効な式。

target_window_name

応答HTMLが表示される、ウィンドウやフレームの名前。

省略値

ブランク - 応答HTMLが現在のウィンドウに表示されます。

有効値

単一引用符付きウィンドウ名またはフレーム名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用可能なウィンドウやフレームが表示されます。または一意の名前を入力することもできます。

'_blank'は新しいウィンドウに表示します。

'_media'は現在のウィンドウのメディア・パネルに表示します。

'_search'は現在のウィンドウの検索パネルに表示します。

'_parent'は親ウィンドウ（多くの場合現ウィンドウ）に表示します。

'_top'は一番上のウィンドウ（多くの場合現ウィンドウ）に表示します。

_search や _media はInternet Explorer 6 でのみサポートされています。

disabled

ウェブレットを使用可または不可として表示するかを決定するブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false() もしくは有効な式。

title

マウスがウェブレットの上を通る時に、ウェブレットのヒントとして表示されるテキスト。

省略値

空白 - ヒントは何も表示されません。

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

text_class

プッシュ・ボタンのキャプションのためのカスケード・スタイル・シート。

省略値

プッシュ・ボタンには省略値はありません。イメージ付きのプッシュ・ボタンの場合は、'std_image_button_text' です。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

presubmit_js

フォームの送信前に実行されるJavaScriptコード。

省略値

空白。JavaScriptは何も実行されません。

有効値

有効なJavaScriptファンクション、もしくは最後にセミコロン(;)が付いたJavaScriptコード。

この要求を送信するJavaScriptを実行せずに、事前に送信されたJavaScriptのみを実行したい場合は(こうすることでonclickイベントがキャンセルされます)、事前に送信されたJavaScriptに**return false;**を追加します。

confirm

trueの場合、アクションを続行する前にユーザーに確認ダイアログボックスを表示します。ユーザーが確認ダイアログボックスで[キャンセル]ボタンをクリックする、または[OK]をクリックせずに確認ダイアログボックスを閉じた場合、ウェブレットにより実行されるアクションはキャンセルされます。

省略値

false() –確認ダイアログボックスは表示されません。

有効値

true()、false() もしくは有効な式。

例

以下は[削除]ボタンに追加する確認ダイアログボックスの表示例です。



confirmText

確認ダイアログボックスに表示するテキストです。このプロパティはconfirmプロパティにtrue()が設定されている場合のみ適用されます。

省略値

空白 - テキスト・メッセージは表示されません。

有効値

単一引用符付きのテキスト、複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)、または結果が文字列となるXPath式。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

default_button

ボタンがフォームの省略値のボタンかどうかを決定するブール値。
フォームの中で省略値にできるボタンには1つだけです- Trueを設定すると、それ以外の全てのボタンがFalseに設定されます。

省略値

ブランク - Falseと同じです。

有効値

true()、false() もしくは有効な式。

vf_wamevent

VLF のWAM イベント文字列です。

省略値

空白。

有効値

文字列。カンマ (,) は使用できません。

8.1.6 チェックボックス (std_checkbox)

クイック・スタート - チェックボックス プロパティ - チェックボックス

チェックボックス・ウェブレットはチェックボックス・コントロールを提供します。これは広い意味でHTMLエレメントの<input type="checkbox">に相当します。

チェックボックス・コントロールは、通常2つの状態の1つを表わす際に使用されます。例えば、オン/オフ; はい/いいえ; 選択/非選択などです。リスト(キャプションなし)で使用される時は、チェックボックス・ウェブレットは次のようになります。:

Dept Code	Department Description
<input type="checkbox"/> ADM	ADMINISTRATOR DEPT
<input checked="" type="checkbox"/> AUD	INTERNAL AUDITING
<input type="checkbox"/> FLT	FLEET ADMINISTRATION

注：チェックボックス・ウェブレットがクリックされた時に他のWebroutineに移動するon_click_wnameなどのプロパティが含まれる場合、チェックボックスのクリックからアクションを開始するのは、あまり良いユーザー・インターフェースの設計方法ではありません。このような場合は、プッシュ・ボタン、メニュー・アイテム、アンカー(ハイパーリンク)などのデバイスを使用してください。

クイック・スタート - チェックボックス

リスト欄のチェックボックスを使って、データ項目や選択状況を示すのが一般的な使用方法です。そのためには、WEB_MAPでリストを指定したWebroutineを作成する必要があります。LANSAエディターで生成されたXSLを開き、次の手順でリストのカラムをチェックボックスとして機能するように変更します。

[ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[チェックボックス]ウェブレットを探します。

このチェックボックス・ウェブレットをリストのカラム上にドラッグし、マウスの左ボタンから手を離します。すると、カラムにチェックボックスが表示されるように変更されます。カラムのこの項目をクリックしてから、[詳細]タブをクリックしてください。チェックボックス・ウェブレットのnameプロパティとvalueプロパティが、ドロップされたフィールドに基づいて既に設定されていることを確認してください。

必要に応じて、captionプロパティを設定します。リスト内で使用する場合、キャプションを使用する必要はありません。キャプションを削除するには、テキストを含まずに2つの引用符を使って、空の文字列を指定します。

オン/オフや選択/非選択状態を示すために、アプリケーション中で使用する値に基づいてoncodeプロパティ及びoffcodeプロパティを設定することもできます。

プロパティ - チェックボックス

チェックボックス・ウェブレットのプロパティは以下のとおりです。

alignment	mouseover_class	protocol
caption	name	reentryfield
class	offcode	reentryvalue
disabled	on_click_wamname	tab_index
display_mode	on_click_wrname	target_window_name
formname	oncode	text_class
hide_if	pos_absolute	value
		vf_wamevent

name

ウェブレットを識別する名前です。ウェブレットがフィールドをビジュアライズすると、これがフィールド名になります。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用することも可能です。

省略値

ウェブレットによりビジュアライズされたフィールドの場合、省略値の名前はフィールド名、もしくはフィールド名と行番号を組み合わせたもの(リストのフィールドの場合)になります。それ以外は、省略値の名前は自動的に生成された、一意の識別子になります。

有効値

単一引用符で囲まれた文字列。

value

ウェブレットに設定する値。ウェブレットによりフィールドがビジュアライズされる場合、この値により表示されるフィールドが識別されま

省略値

省略値は適用されません - このウェブレットを使用する多くの場合、チェックボックスを示す値の入ったフィールドとチェックボックスの状態の受け取りに使用するフィールドの両方もしくはどちらかを指定する必要があります。

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

display_mode

ウェブレットが入力を受け取るかどうか、出力を表示するかどうかを制御します。

省略値

ブランク ('input'の意味です)。

有効値

リテラル値'input'、'output'または'hidden'。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できる値のリストが表示されます。もしくは、実行時に使用できる値を含むフィールド名、システム変数名、または複数言語変数名を入力することもできます。

caption

ウェブレットのキャプション。キャプションはWebページのチェックボックスの隣に表示されます。

省略値

'Caption' (プレースホルダーの省略値です。 - キャプションを設定する必要があります。)

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

チェック・ボックスの隣にキャプションを表示したくない場合(例えば、リストの中で使用する場合など)は、間にテキストを含まない2つのクォーテーションマークを指定することで、空の文字列を指定することができます。

oncode

チェックボックスのチェックされた状態を表したり、設定するために使用する値。

省略値

'Y'

有効値

単一引用符で囲まれたテキスト、フィールド名、システム変数、または複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

offcode

チェックボックスのチェックされていない状態を表したり、設定するために使用する値。

省略値

'N'

有効値

単一引用符で囲まれたテキスト、フィールド名、システム変数、または複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

reentryfield

reentryvalue プロパティに指定された値をWAMに送る時に使用するフィールド名です。フィールド名は単一引用符で囲みます。

詳しくは、reentryvalue プロパティの説明を参照してください。

注:このプロパティは、WEBEVENTアプリケーションでよく使用される、再入可能なプログラミング技術をサポートするために提供されています。最初からWAMを使用するように設計されたウェブ・アプリケーションでは、通常はこの技術を使用する必要はありません。

省略値

'STDREENTRY'

有効値

単一引用符で囲まれた文字列。

reentryvalue

ターゲットWebroutineのreentryfieldプロパティで指定されたフィールドに送信する値です。英数字フィールドの場合、値は単一引用符付きで指定する必要があります。数字フィールドの場合は、単一引用符はなくても構いません。

このプロパティは、reentryfieldプロパティと共に使用され、値をターゲットのWebroutineにどのように送信するか記述します。以下の2つの情報が必要になります。

1. reentryfield: ターゲットWebroutineが情報を参照するために使用するフィールド名。
2. reentryvalue: リテラル値もしくは必要な情報を含むこの(ソース)Webroutineのフィールド名。

注: このプロパティは、WEBEVENTアプリケーションでよく使用される、再入可能なプログラミング技術をサポートするために提供されています。最初からWAMを使用するように設計されたウェブ・アプリケーションでは、通常はこの技術を使用する必要はありません。

省略値

'M'

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

False() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

例

次の例はフィールド#STD_FLAGが 'X' と等しい場合にウェブレットを非表示にします。この式は以下のような形式で入力してください。

hide_if	#STD_FLAG = 'X'
---------	-----------------

プロパティがフォーカスを失うと、この式は以下のように表示されます。

hide_if	key('field-value', 'STD_FLAG') = 'X'
---------	--------------------------------------

formname

サーバーに送られるHTMLフォーム名です。

省略値

'LANSA'

有効値

単一引用符付きのフォーム名。プロパティ・シート内で該当するドロップダウン・ボタンをクリックすると、使用できるフォーム名リストが表示されます。

on_click_wamname

チェックボックスがクリックされた時にWebroutineが実行されるWAMの名前を指定します。(Webroutine名はon_click_wrnameプロパティに指定されます。)on_click_wrnameプロパティが指定されていないときは、このプロパティは無視されます。

注：チェックボックスのクリックでアクションを起動させるのは、良いユーザー・インタフェースとは言えません。このような場合は、プッシュ・ボタン、メニュー・アイテム、アンカー(ハイパーリンク)などのデバイスを使用してください。

省略値

指定しない場合は、現在のWAMが使用されます。
(\$lweb_WAMName).

有効値

単一引用符付きのWAM名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

on_click_wrname

チェック・ボックスがクリックされたときに実行されるWebroutineの名前を指定します。(WebroutineのあるWAM名はon_click_wamnameプロパティで指定します。)

注：チェックボックスのクリックでアクションを起動させるのは、良いユーザー・インタフェースとは言えません。このような場合は、プッシュ・ボタン、メニュー・アイテム、アンカー(ハイパーリンク)などのデバイスを使用してください。

省略値

デフォルト値は適用されません。

有効値

単一引用符付きのWebroutine名。Webroutineは、on_click_wamnameプロパティに指定されたWAMに存在していなくてはなりません。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWebroutineのリストが表示されます。

protocol

on_click_wnameプロパティに指定されたWebroutineに移動する際に使用されるプロトコル(例:http:// または https://)。

通常、このプロパティはセキュア・モードの処理との切り替えが必要な時に使用します。それ以外の場合は、通常このプロパティを指定する必要はありません。

省略値

ブランク。現在使用されているプロトコルが使用されます。

有効値

単一引用符付きの、有効なプロトコル。指定する場合は、通常は'http:'または'https:'です。

target_window_name

on_click_wrnameプロパティに指定されたWebroutineに処理を移す時に応答HTMLが表示されるウィンドウまたはフレームの名前。

省略値

ブランク - 応答HTMLは現在のウィンドウに表示されます。

有効値

単一引用符付きウィンドウ名またはフレーム名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるウィンドウやフレームが表示されます。

disabled

ウェブレットを使用可または不可として表示するかを決定するブール値。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false() もしくは有効な式。

alignment

キャプションの表示をチェックボックスの左か右にするかを決定します。

省略値

'right'

有効値

'left' (左)、'right' (右)、または実行時にこの2つの値のいずれかを含むフィールド名やシステム変数。

pos_absolute

ウェブ・ページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

例

次の例では、[ポジションの固定]がウェブレットで選択されており、LANSAエディターの[デザイン]ビューで要求された位置にウェブレットが配置されています。この値が、pos_absolute_designプロパティに表示されます。

```
pos_absolute_design 'position:absolute;left: 324pt; top: 162.72pt;'
```

class

ウェブレットのカスケード・スタイル・シート(CSS)クラス名です。

省略値

提供されているウェブレットのクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

mouseover_class

マウスがその上に置かれた時の、ウェブレットのカスケード・スタイル・シート(CSS)クラス名。

省略値

このウェブレットに対して適用される省略値はありません。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

text_class

ウェブレットのテキストのカスケード・スタイル・シート(CSS)クラス名です。

省略値

提供されているウェブレットのテキスト・クラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

vf_wamevent

VLF のWAM イベント文字列です。

Default value

空白。

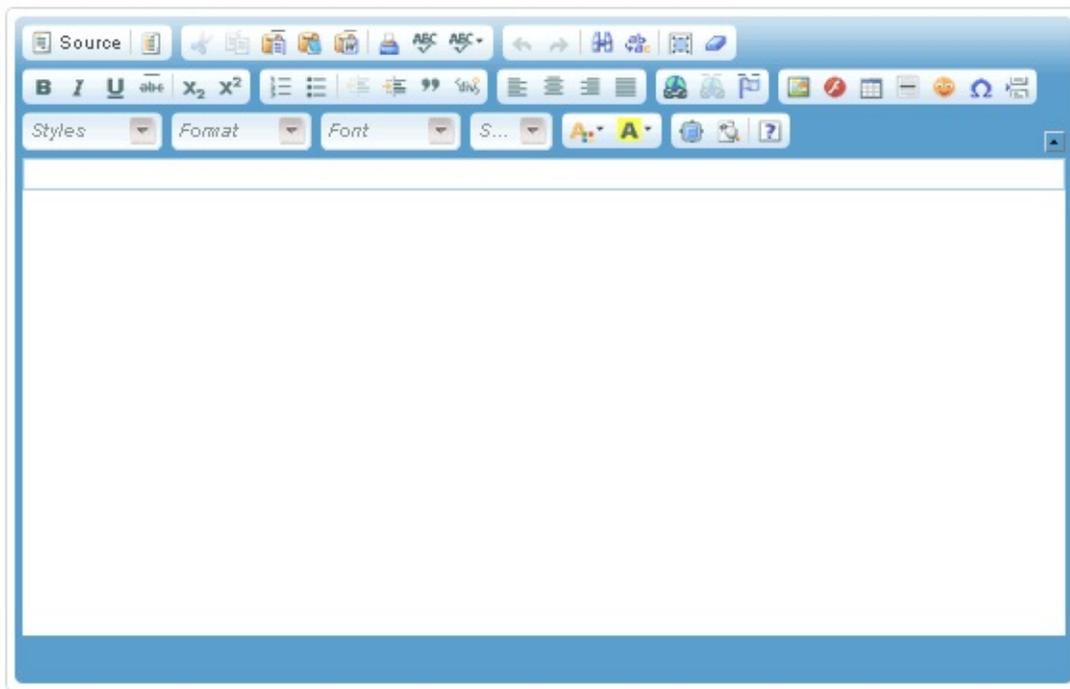
Valid values

文字列。カンマ (,) は使用できません。

8.1.7 CKEditor リッチ・テキスト・エディター (std_ckeditor)

[クイック・スタート - CKEditor](#) - [プロパティ - CKEditorリッチ・テキスト・エディター](#)

CKEditorはリッチ・テキスト・エディターです。これはWYSIWYGエディターです。つまり、編集しているテキストが、これが公開された時にユーザーが得る結果とできるだけ同じように表示されるということです。



クイック・スタート - CKEditor

CKEditorウェブレットを使用するには、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[CKEditor リッチ テキスト エディター]ウェブレットを探します。
2. このウェブレットを自身のページにドラッグ・アンド・ドロップしてください。ウェブレットを選択して、[詳細]タブをクリックするのを忘れないでください。
3. CKEditorのnameプロパティにフィールド名を割り当てます。このフィールドは通常は文字列フィールドで、エスケープ時に予想されるコンテンツを格納できる長さのものです。（エスケープ時にすべてのマークアップ・テキストがCKEditorにより格納されます。）

プロパティ – CKEditorリッチ・テキスト・エディター

autoGrow	pos_absolute	showSource
autoGrow_maxHeight	resize_dir	tab_index
autoGrow_minHeight	resize_enabled	toolbar
contentCss	resize_maxHeight	toolbarCanCollapse
height_design	resize_maxWidth	uiColor
hide_if	resize_minHeight	value
name	resize_minWidth	valueFromField
onchange_script	showElementsPath	width_design

name

ウェブレットを識別する名前です。ウェブレットがフィールドをビジュアライズすると、これがフィールド名になります。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用することも可能です。

省略値

ウェブレットがフィールドをビジュアライズしている場合、省略値前はフィールド名です。それ以外は、省略値の名前は自動的に生成された、一意の識別子になります。

有効値

単一引用符で囲まれた文字列。

value

ウェブレットに設定する値。ウェブレットがフィールドをビジュアルライズする場合、この値を表示するフィールドが識別されます。

省略値

デフォルト値は適用されません。

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

valueFromField

trueが設定されていて、valueプロパティがnullの場合、CKEditorウェブレットはCKEditorのname属性と一致するフィールドから値をロードします。テキストのコンテンツが大きく、CKEditorウェブレットのvalueとWebroutineのフィールド値リストの両方に表示したくない時にこのオプションを使用します。

省略値

False()

有効値

ブール値を返す有効なXPath式。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

toolbar

エディターに含む機能のレベル。フル・オプションはCKEditorで使用可能な全てのオプションを表示します。簡易な編集能力だけを提供したい場合は、ベーシック・ツールバーを使用します。

省略値

Basic

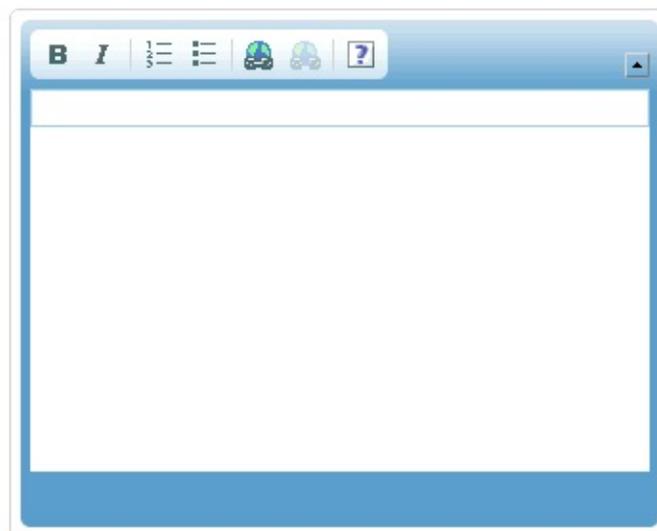
有効値

Basic

Full

例

次はベーシック・ツールバーの付いたCKEditorです。



showSource

フルのツールバーを表示している時に、ユーザーがHTMLソースの表示・編集ができるかどうかを決定します。

ユーザーがフィールド値にどんなマークアップも追加できるようにすると、セキュリティが脆弱になる恐れがあります。このプロパティは十分気を付けて使用するようになっています。

省略値

false

有効値

false()、true()、およびブール値を返す有効なXPath式。

showElementsPath

ドキュメント内のカーソル位置のHTMLエレメントに関する情報を示すエレメント・パスを表示するかどうか決定します。CKEditorのステータスバーに表示されます。

省略値

`true`

有効値

`false()`、`true()`、および **ブール値を返す有効なXPath式**。

toolbarCanCollapse

ユーザーがツールバーを折りたたむことができるかどうかを決定します。不可に設定されている場合、折りたたみボタンは表示されません。

省略値

true

有効値

false()、true()、およびブール値を返す有効なXPath式。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（ウェブレットは相対的に位置付けられます。）

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width_design

Webページのウェブレットの幅です。

通常、ウェブレット幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。こうすることで、width-design プロパティやheight_designプロパティの値を更新します。ただし、必要に応じてプロパティ値を直接編集することもできます。

省略値

ブランク。

有効値

単一引用符で囲まれた有効な範囲内の幅。

height_design

Webページのウェブレットの高さ。

通常、ウェブレットの高さや幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。こうすることで、width-design プロパティやheight_design プロパティの値を更新します。ただし、必要に応じてプロパティの値を直接編集することもできます。

省略値

ブランク。

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

resize_enabled

CKEditorのサイズ変更を可能にするかどうか決定します。不可に設定されている場合、サイズ変更ハンドルは表示されません。

省略値

false

有効値

false()、true()、およびブール値を返す有効なXPath式。

resize_dir

エディターのサイズ変更が可能な方向を指定します。CKEditorのサイズ変更が可能に指定されている時のみ適用されます。

省略値

Both

有効値

Both (両方向)

Vertical (縦方向)

Horizontal (横方向)

autoGrow

AutoGrowを有効にするか無効にするかを指定します。AutoGrowは、ユーザーがコンテンツ・エリアを埋めていくごとに、コンテンツ・エリアを拡大します。

省略値

false

有効値

false()、true()、およびブール値を返す有効なXPath式。

autoGrow_maxHeight

エディターがAutoGrowを使って到達できる最大の高さです。

省略値

空白。

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

autoGrow_minHeight

エディターがAutoGrowを使って到達できる最小の高さです。

省略値

空白。

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

resize_maxHeight

エディターをサイズ変更ハンドルでサイズ変更する時の最大の高さをピクセルで表します。

省略値

3000 pixels

有効値

ピクセルで記述された高さ。

resize_maxWidth

エディターをサイズ変更ハンドルでサイズ変更する時の最大の幅をピクセルで表します。

省略値

3000 pixels

有効値

ピクセルで記述された幅。

resize_minHeight

エディターをサイズ変更ハンドルでサイズ変更する時の最小の高さをピクセルで表します。注：省略値より小さい場合は、エディターの実際の高さに変更されます。

省略値

250 pixels

有効値

ピクセルで記述された高さ。

resize_minWidth

エディターをサイズ変更ハンドルでサイズ変更する時の最小の幅をピクセルで表します。注：省略値より小さい場合は、エディターの実際の幅に変更されます。

省略値

750 pixels

有効値

ピクセルで記述された幅。

contentCss

コンテンツへのスタイル適用時に使用する（1つまたは複数の）CSSファイル（カンマで区切られたリスト）です。このコンテンツが使用する、最後のページのCSSが反映されていなければなりません。

Webroutineで使用されたものと同じCSSを適用するには、特別な値'inherit'を使用します。

省略値

Inherit

有効値

スタイルシートをカンマで区切ったリスト。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

uiColor

ツールバー・エリアの背景色。

省略値

Theme:省略値を現在のテーマと一致する色に設定します。

有効値

#RRGGBBフォーマットの色。

onchange_script

テキストが変更された後、入力ボックスがフォーカスを失った時に実行されるJavaScriptコード。JavaScriptステートメントは必ずセミコロンで終了していなければなりません。

省略値

空白。JavaScriptは何も実行されません。

有効値

有効なJavaScriptステートメント。

8.1.8 クリック可能イメージ (std_click_image)

クイック・スタート - クリック可能イ プロパティ - クリック可能イ
メージ メッセージ

クリック可能イメージのウェブレットは、Webページ上に表示されるイメージに、クリックすることでアクションが実行できるメカニズムを提供します。省略値では、以下のような青丸が表示されます。



クイック・スタート - クリック可能イメージ

リストのカラムにクリック可能イメージを使用するには、WEB_MAPに *BOTH もしくは *OUTPUTを指定したWebroutineを作成する必要があります。LANSAエディターで生成したXSLを開き、以下の手順でリストのカラムをクリック可能イメージとして機能するように変更することができます。

[ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[クリック可能イメージ]ウェブレットを探します。

クリック可能にしたいリストのカラムに、これをドラッグ・アンド・ドロップします。すると、カラムにイメージが表示されるように変更されます。カラムのこの項目をクリックしてから、[詳細]タブをクリックしてください。クリック可能イメージ・ウェブレットのnameプロパティとvalueプロパティが、ドロップされたフィールドに従って既に設定されていることを確認してください。

currentrowhfieldプロパティ及びcurrentrownumvalプロパティを、プロパティ記述の説明通りに設定します。

on_click_wrnameプロパティにイメージがクリックされたときに呼び出されるWebroutineの名前を設定してください。Webroutineが現在のWebroutineと異なるWAMにある場合は、on_click_wamnameプロパティも設定する必要があります。

プロパティ - クリック可能イメージ

クリック可能イメージ・ウェブレットのプロパティは以下のとおりです。

<code>absolute_image_path</code>	<code>mouseover_absolute_image_path</code>	<code>relative_image_path</code>
<code>class</code>	<code>mouseover_relative_image_path</code>	<code>show_in_new_window</code>
<code>currentrowhfield</code>	<code>name</code>	<code>tab_index</code>
<code>currentrownumval</code>	<code>on_click_wamname</code>	<code>target_window_name</code>
<code>disabled</code>	<code>on_click_wrname</code>	<code>tooltip</code>
<code>disabled_class</code>	<code>pos_absolute</code>	<code>url</code>
<code>formname</code>	<code>presubmit_js</code>	<code>value</code>
<code>height_design</code>	<code>protocol</code>	<code>vf_wamevent</code>
<code>hide_focus</code>	<code>reentryfield</code>	<code>width_design</code>
<code>hide_if</code>	<code>reentryvalue</code>	

name

ウェブレットの名前。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript や ウェブレットを参照するXSLを使用する場合は、独自の名前を使用することも可能です。

省略値

`concat('o', position(), '_LANSA_n')` - LANSAにより付けられたウェブレット内部名です。

有効値

シングル・クォーテーションで囲まれた名前。

value

ウェブレットに設定する値。ウェブレットによりフィールドがビジュアライズされた場合、これがフィールドの値になります。

省略値

空白。

有効値

シングル・クォーテーション付きのテキスト、もしくは複数言語対応テキスト変数、システム変数、フィールドの名前(プロパティ・シートの関連する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

currentrowhfield

現在行に指定された値を含むフィールドの名前。このプロパティは、ウェブレットがリストの中で使用されているときのみ、使用できます。

省略値

'STDROWNUM' - currentrownumvalプロパティの省略値position()と共に、STDROWNUMはリストのエントリー番号を保持します。

有効値

シングル・クォーテーション付きの、リポジトリまたはWAMで定義されたフィールド名。プロパティシート内で該当するドロップダウン・ボタンをクリックすると、使用できるフィールド名リストが表示されます。

currentrownumval

currentrowhfieldプロパティに指定されたフィールドに入れられる値。このプロパティは、ウェブレットがリストの中で使用されているときのみ、使用できます。

省略値

Position() - currentrowhfieldプロパティの省略値STDROWNUMと共に、STDROWNUMはリストのエントリー番号を保持します。

有効値

適切な有効な値。リストの現在の行からフィールド値を指定する場合、フィールド名の前に'\$'をつけてください。

reentryfield

reentryvalue プロパティに指定された値をWAMに送る時に使用するフィールド名です。フィールド名は単一引用符で囲みます。

省略値

'STDREENTRY'

有効値

リポジトリもしくはWAMで定義されたフィールド名。プロパティシート内で該当するドロップダウン・ボタンをクリックすると、使用できるフィールド名リストが表示されます。

reentryvalue

reentryfieldプロパティに指定されたフィールドに送られる値。英数字フィールドの場合、値は単一引用符付きで指定する必要があります。数字フィールドの場合は、単一引用符はなくても構いません。

省略値

'M'

有効値

適切なリテラル。

tooltip

マウスがウェブレットの上を通る時に、ウェブレットのヒントとして表示されるテキスト。

省略値

'Tooltip'

有効値

シングル・クォーテーション付きのテキスト、もしくは複数言語対応テキスト変数、システム変数、フィールドの名前(プロパティ・シートの関連する省略記号のボタンをクリックしてリストから選ぶことができます。)

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

例

次の例はフィールドSTD_FLAGが 'X' と等しい場合にウェブレットを非表示にします。式は入力されなくてはなりません。またプロパティにフォーカスがある時、以下のように表示されます。

hide_if	#STD_FLAG = 'X'
---------	-----------------

プロパティがフォーカスを失うと、この式は以下のように表示されます。

hide_if	key('field-value', 'STD_FLAG') = 'X'
---------	--------------------------------------

formname

サーバーに送られるHTMLフォーム名です。

省略値

'LANSA'

有効値

単一引用符付きのフォーム名。プロパティ・シート内で該当するドロップダウン・ボタンをクリックすると、使用できるフォーム名リストが表示されます。

url

このプロパティを使って、ウェブレットがクリックされた時の移動先URLを指定できます。指定する場合、URLにはリテラル値（例えば'http://www.mycompany.com/'など）または実行時にURLを含むフィールド名を指定できます。

このプロパティはon_click_wamname、 on_click_wrname及びprotocolプロパティよりも優先されます。urlが指定されている場合、上記のプロパティは無視されます。

省略値

'javascript:void();' - 何もないのと同じです。

有効値

単一引用符で囲まれたURL、実行時にURLが含まれるフィールド名、システム変数名、複数言語変数名。

on_click_wamname

ウェブレットがクリックされたときに呼び出されるWAMの名前。

省略値

\$lweb_WAMName (現在のWAMを示します。)

有効値

単一引用符付きのWAM名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

on_click_wrname

ウェブレットがクリックされたときに呼び出されるWebroutineの名前。

省略値

適用されません - Webroutine名の指定は必須です。

有効値

単一引用符付きのWebroutine名。Webroutineは、on_click_wamnameプロパティに指定されたWAMに存在してはいけません。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWebroutineのリストが表示されます。

protocol

on_click_wnameプロパティに指定されたWebroutineへの移動に使用する
プロトコル(例:http:// または https://)。

省略値

空白。現在使用されているプロトコルが使用されます。

有効値

単一引用符付きの、有効なプロトコル。通常は'http:'または'https:'で
す。

show_in_new_window

ウェブレットの応答HTMLを新しいブラウザ・ウィンドウに表示するかどうかを決定するブール値プロパティ。

省略値

false() - 応答HTMLは現ブラウザ・ウィンドウに表示されます。

有効値

true()、false()、もしくは、TrueかFalseを返す有効な式。

target_window_name

応答HTMLが表示される、ウィンドウやフレームの名前。

省略値

ブランク - 応答HTMLが現在のウィンドウに表示されます。

有効値

単一引用符で囲まれたウィンドウ名またはフレーム名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるウィンドウやフレームが表示されます。

'_blank'は新しいウィンドウに表示します。

'_media'は現在のウィンドウのメディア・パネルに表示します。

'_search'は現在のウィンドウの検索パネルに表示します。

'_parent'は親ウィンドウ（多くの場合現ウィンドウ）に表示します。

'_top'は一番上のウィンドウ（多くの場合現ウィンドウ）に表示します。

_search や _media はInternet Explorer 6 でのみサポートされています。

disabled

ウェブレットを使用可または不可のいずれで表示するかを決定するブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false()、もしくは、TrueかFalseを返す有効な式。

hide_focus

評価がTrueの時にフォーカスのあるウェブレットのフォーカス用の四角を非表示にするブール値プロパティ。

省略値

true()

有効値

true()、false()、もしくは、ブール値を返す有効な式。

relative_image_path

表示されるイメージの'イメージ'・ディレクトリに関連付けられたパスとファイル名。指定する場合は、absolute_image_pathプロパティはブランクにすること。

省略値

'ball_blue.gif'

有効値

イメージのイメージ・ディレクトリに関連付けられたパスおよび名前
で、単一引用符で囲まれたもの。イメージは、プロパティ・シートで
該当する省略記号のボタンをクリックし、プロンプターから選択する
ことができます。

absolute_image_path

表示するイメージのパスとファイル名指定する場合、relative_image_pathプロパティは空白でなければいけません。

省略値

省略値はrelative_image_pathプロパティに指定されたイメージを使用します。

有効値

単一引用符で囲まれたイメージのパスと名前。

mouseover_relative_image_path

ウェブレットの上をマウスが通った時に表示されるイメージの、'イメージ'・ディレクトリに関連付けられたパスおよびファイル名。

省略値

blank - マウスがウェブレットの上を通った時にイメージは変更されません。

有効値

イメージのイメージ・ディレクトリに関連付けられたパスおよび名前を単一引用符で囲んだもの。イメージは、プロパティ・シートで該当する省略記号のボタンをクリックし、プロンプターから選択することができます。

mouseover_absolute_image_path

ウェブレットの上をマウスが通った時に表示されるイメージのパスとファイル名。

省略値

空白 - 省略値はmouseover_relative_image_pathプロパティに指定されたイメージを使用します。

有効値

単一引用符で囲まれたイメージのパスと名前。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width_design

Webページのウェブレットの幅です。

省略値

ブランク（ウェブレットは省略値の幅を使用します。）

有効値

単一引用符で囲まれた有効な範囲内の幅。

height_design

Webページのウェブレットの高さ。

省略値

ブランク(ウェブレットは省略値の高さを使用します。)

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

class

ウェブレットのカスケード・スタイル・シートのクラス名。

省略値

'std_click_image' - このウェブレット用に提供されているクラス名です。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

disabled_class

disabledプロパティがTrueにセットされた時のウェブレットのカスケード・スタイル・シート。

省略値

'std_click_image_disabled' - このウェブレット用に提供されている disabledクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

presubmit_js

フォームの送信前に実行されるJavaScriptコード。

省略値

ブランク。JavaScriptは何も実行されません。

有効値

有効なJavaScriptファンクション、もしくは最後にセミコロン(;)が付いたJavaScriptコード。

この要求を送信するJavaScriptを実行せずに、事前に送信されたJavaScriptのみを実行したい場合は(こうすることでonclickイベントがキャンセルされます)、事前に送信されたJavaScriptに**return false;**を追加します。

例

以下の例はメッセージ・ボックスを表示します。

```
presubmit_js      'alert("Hello world!");'
```

次の例はメッセージ・ボックスを表示して、JavaScriptの送信をキャンセルします。

```
presubmit_js      'alert("Hello world!"); return false;'
```

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

vf_wamevent

VLF のWAM イベント文字列です。

省略値

空白。

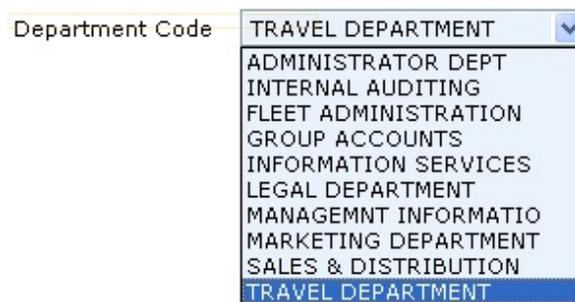
有効値

文字列。カンマ (,) は使用できません。

8.1.9 コンボ・ボックス (std_dropdown)

クイック・スタート - コンボ・ボックス プロパティ - コンボ・ボックス

コンボ・ボックス・ウェブレットは、フィールドにドロップダウンの選択を提供します。ドロップダウンに使用する値は、作業リストの値や、ウェブレットのitemsプロパティ経由で定義された静的な値も利用できます。各ドロップダウンは、<select> HTMLタグとして組込まれます。例えば次のようなものです。



The image shows a web browser window with a dropdown menu. The label 'Department Code' is on the left. The dropdown menu is open, showing a list of department names. The top item is 'TRAVEL DEPARTMENT' with a small blue arrow icon to its right. Below it are several other department names: 'ADMINISTRATOR DEPT', 'INTERNAL AUDITING', 'FLEET ADMINISTRATION', 'GROUP ACCOUNTS', 'INFORMATION SERVICES', 'LEGAL DEPARTMENT', 'MANAGEMENT INFORMATION', 'MARKETING DEPARTMENT', 'SALES & DISTRIBUTION', and 'TRAVEL DEPARTMENT'. The bottom item, 'TRAVEL DEPARTMENT', is highlighted with a blue background.

Department Code
TRAVEL DEPARTMENT
ADMINISTRATOR DEPT
INTERNAL AUDITING
FLEET ADMINISTRATION
GROUP ACCOUNTS
INFORMATION SERVICES
LEGAL DEPARTMENT
MANAGEMENT INFORMATION
MARKETING DEPARTMENT
SALES & DISTRIBUTION
TRAVEL DEPARTMENT

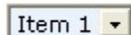
クイック・スタート - コンボ・ボックス

コンボ・ボックスの各エントリーは、作業リストのエントリーもしくはコンボ・ボックスのプロパティでハードコーディングされた一連の項目によって定義されます。

作業リストを使用する場合

ドロップダウン・オプションの定義に作業リストを使用する場合、Webroutineを作成して選択された値を格納するフィールドと選択オプションを含む作業リストをWEB_MAPに定義する必要があります。LANSAエディターでWebroutine用に生成されたXSLを開き、以下の手順に従ってください。

1. 作業リストがWEB_MAPで*HIDDENではない場合は、省略値で作業リストを表わすテーブルがWebページに含まれます。リストをビジュアライズするテーブルは削除します。削除するには、リストで右クリックし、ポップアップ・メニューから、[全てのリスト削除]を選択してください。
2. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[コンボ ボックス]ウェブレットを探します。
3. コンボ・ボックス ウェブレットを、値を格納するフィールド上にドラッグし、マウスの左ボタンを離してください。これで、ドロップダウンのオプションが表示されます。



4. ウェブレットをクリックして、[詳細]タブを確認します。name プロパティとvalueプロパティにウェブレットをドラッグしたフィールドを示す値が設定されていることを確認してください。Webページ表示の際、valueプロパティのフィールドに現在入っている値を使用して選択されたドロップダウン・エントリーが設定されます。ドロップダウンの値が変更されると、適切な値がnameプロパティに指定されたフィールド（この場合、同じフィールド）に入れられます。
5. listnameプロパティをWEB_MAPに送られる作業リストに変更します。コンボ・ボックス表示はすぐに作業リストの表示に変更されるはずですが。
6. codefieldプロパティとcaptionfieldプロパティに作業リストの適切なフィールドを設定します。

itemsプロパティを使用する場合

コンボ・ボックスのプロパティにハードコーディングされた一連の項目を使用するには、WEB_MAPにフィールドを指定したWebroutineを作成する必要があります。LANSAエディターでWebroutine用に生成されたXSLを開き、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[コンボ ボックス]ウェブレットを探します。
2. コンボ・ボックス ウェブレットを、値を格納するフィールド上にドラッグし、マウスの左ボタンを離してください。これで、ドロップダウンのオプションが表示されます。

A small screenshot of a web browser interface showing a dropdown menu with the text 'Item 1' and a downward-pointing arrow.

3. ウェブレットをクリックして、[詳細]タブを確認します。name プロパティとvalueプロパティにウェブレットをドラッグしたフィールドを示す値が設定されていることを確認してください。Webページ表示の際、valueプロパティのフィールドに現在入っている値を使用して選択されたドロップダウン・エントリーが設定されます。ドロップダウンの値が変更されると、適切な値がnameプロパティに指定されたフィールド（この場合、同じフィールド）に入れられます。
4. itemsプロパティの省略記号(...)のボタンを選択して、ドロップダウン・オプションとして使用する項目のリストを設定します。そしてドロップダウンに必要なエントリーの定義に進みます。

プロパティ - コンボ・ボックス

コンボ・ボックス・ウェブレットのプロパティは以下のとおりです。

captionfield	mouseover_class	selector_value_eq
class	name	submit_tagfields
codefield	on_change_wamname	tab_index
disabled	on_change_wrname	tagfield1
display_mode	pos_absolute	tagfield2
formname	protocol	tagfield3
hide_if	reentryfield	target_window_name
items	reentryvalue	value
listname	selector_field	vf_wamevent
		width_design

name

ドロップ・ダウンの名前。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。このウェブレットがフィールドの上にドロップされた場合、またはフィールドを表示したり、フィールドに値を追加するために使用される場合は、フィールド名が使用されます。

省略値

自動的に生成された、一意の識別子。

有効値

単一引用符で囲まれた文字列。

例

次は、フィールドとは関係のない省略値の名前を示しています。

```
name concat('o', position(), '_LANSA_13186')
```

これは、ウェブレットがSTD_FLAGのフィールドに関連付けられている場合です。

```
name 'STD_FLAG'
```

value

ウェブレットに設定する値。このウェブレットによりフィールドがビジュアライズした場合、これがフィールドの値もしくは省略値になります。

省略値

空白。

有効値

シングル・クォーテーション付きのテキスト、もしくは複数言語対応テキスト変数、システム変数、フィールドの名前(プロパティ・シートの関連する省略記号のボタンをクリックしてリストから選ぶことができます。)

例

次の例では、valueはフィールド#SECTIONの現在の値に設定されることを示しています。プロパティに入力されると、以下のようになります。



A screenshot of a property editor. The 'value' field is highlighted in blue, and the '#SECTION' field is highlighted in light blue.

フォーカスがこのプロパティから移った時は、同じvalueプロパティが以下のように表示されます。



A screenshot of a property editor. The 'value' field is highlighted in light blue, and the 'key('field-value', 'SECTION')' field is highlighted in light blue.

display_mode

ウェブレットが入力内容を受け取るのか、出力内容を表示するのかを制御します。

省略値

'input'

有効値

'input' または 'output'

items

ウェブレットに表示する項目を指定するXMLノード・セット。設定はデザイナーでのみ可能です。デザイナーを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使用してください。項目がlistnameプロパティに指定されたりリストから追加される場合は、ブランクにしてください。

省略値

document(")*/lxml:data/lxml:dropdown (このドロップダウンに項目が何も定義されていないことを示します。)

有効値

適用不可。(この値はシステムにより保守されます。)デザイナーを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使用してください。

例

次の例は項目がデザイナーで設定され、ドロップダウンの値として使用されることを示しています。

```
items document(")*/lxml:data/lxml:caption_value_pairs[@id='4A14173FA6D64A059507F9124A630CF3']
```

プロパティの省略記号(...)のボタンを使用すると、デザイナーが表示され、ドロップダウンに表示される項目を管理できます。以下のデザイナーのビューはドロップダウンに2つのエントリーが設定されたことを示しています。最初のエントリーはリテラル値'MONDAY'、2つめのエントリーは複数言語変数を使用して、コードTUEの記述を表示します。値が事前に選択されていない時に選択された状態にする項目には[規定のアイテム]のチェックボックスを使用します。



listname

グリッドにセルを追加するために使用する作業リストの名前。itemsプロパティに詳細が指定されている場合は、ブランクのままにしてください。

省略値

ブランク。

有効値

単一引用符で囲まれた文字列。使用可能な（WAMで定義された）作業リストの一覧は、プロパティシートで該当するドロップダウン・ボタンをクリックすることで、選択することができます。

selector_field

listnameプロパティに指定されているリスト内のフィールドで、ウェブレットに表示されたリスト項目を制限したり、部分集合にしたりする値を含むフィールドの名前。このプロパティはselector_value_eqプロパティと共に使用されます。

省略値

空白。

有効値

単一引用符で囲まれた文字列。プロパティ・シートで該当するドロップダウン・ボタンをクリックすると、listnameに指定された作業リストからのフィールドを選択できます。

selector_value_eq

この値を使って、ウェブレットに表示されるリスト項目を制限したり、部分集合にしたりします。listnameプロパティが指定されている場合、selector_fieldプロパティに関係するフィールドを指定する必要があります。itemsプロパティのデザイナーを使って値のリストが定義された場合、デザイナーで入力された該当するセレクター値が使用されます。

省略値

空白。

有効値

単一引用符で囲まれたテキスト、もしくは数値。プロパティ・シートで該当するドロップダウン・ボタンをクリックすると、listnameに指定された作業リストからのフィールドを選択できます。

codefield

各リスト項目のキー値を保持するlistnameプロパティに指定されたリストのフィールド名。

省略値

空白。

有効値

単一引用符で囲まれた文字列。プロパティ・シートで該当するドロップダウン・ボタンをクリックすると、listnameに指定された作業リストからのフィールドを選択できます。

captionfield

各リスト項目のキャプションを保持するlistnameプロパティに指定されたリストのフィールド名。

省略値

空白。

有効値

単一引用符で囲まれた文字列。プロパティ・シートで該当するドロップダウン・ボタンをクリックすると、listnameに指定された作業リストからのフィールドを選択できます。

tagfield1

リスト項目にタグ付けされる追加の値を含むlistnameプロパティに指定されたリストのフィールド名。この値は、接尾辞が'tag_'のフィールド名の属性として追加されます。こうすることで、この属性はJavaScriptコードの中で指定することができるようになります。

省略値

ブランク。

有効値

単一引用符で囲まれた文字列。プロパティ・シートで該当するドロップダウン・ボタンをクリックすると、listnameに指定された作業リストからのフィールドを選択できます。

tagfield2

リスト項目にタグ付けされる追加の値を含むlistnameプロパティに指定されたリストのフィールド名。この値は、接尾辞が'tag_'のフィールド名の属性として追加されます。こうすることで、この属性はJavaScriptコードの中で指定することができるようになります。

省略値

ブランク。

有効値

単一引用符で囲まれた文字列。プロパティ・シートで該当するドロップダウン・ボタンをクリックすると、listnameに指定された作業リストからのフィールドを選択できます。

tagfield3

リスト項目にタグ付けされる追加の値を含むlistnameプロパティに指定されたリストのフィールド名。この値は、接尾辞が'tag_'のフィールド名の属性として追加されます。こうすることで、この属性はJavaScriptコードの中で指定することができるようになります。

省略値

空白。

有効値

単一引用符で囲まれた文字列。プロパティ・シートで該当するドロップダウン・ボタンをクリックすると、listnameに指定された作業リストからのフィールドを選択できます。

submit_tagfields

ブール値プロパティです。タグ・フィールド値を残りのフォームと一緒に送信する場合はTrueを設定します。

省略値

true()

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

reentryfield

reentryvalue プロパティに指定された値をWAMに送る時に使用するフィールド名です。フィールド名は単一引用符で囲みます。

省略値

'STDREENTRY'

有効値

リポジトリもしくはWAMで定義されたフィールド名。プロパティシート内で該当するドロップダウン・ボタンをクリックすると、使用できるフィールド名リストが表示されます。

reentryvalue

reentryfieldプロパティに指定されたフィールドに送られる値。英数字フィールドの場合、値は単一引用符付きで指定する必要があります。数字フィールドの場合は、単一引用符はなくても構いません。

省略値

'M'

有効値

適切なリテラル。

hide_if

ブール値プロパティです。評価がTrueの時にウェブレットが非表示になる式です。

省略値

False() (グリッドはいつも表示されます。)

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

例

次の例はフィールド#STD_FLAGが'X'と等しい場合にグリッドを非表示にします。式は入力されなくてはなりません。またプロパティにフォーカスがある時、以下のように表示されます。

hide_if	#STD_FLAG = 'X'
---------	-----------------

プロパティがフォーカスを失うと、この式は以下のように表示されません。

hide_if	key('field-value', 'STD_FLAG') = 'X'
---------	--------------------------------------

formname

サーバーに送られるHTMLフォーム名です。

省略値

'LANSA' (document.LANSAのことです。)

有効値

単一引用符付きのフォーム名。プロパティ・シート内で該当するドロップダウン・ボタンをクリックすると、使用できるフォーム名リストが表示されます。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width_design

Webページのウェブレットの幅です。

省略値

ブランク（ウェブレットは省略値の幅を使用します。）

有効値

単一引用符で囲まれた有効な範囲内の幅。

on_change_wamname

ウェブレット内の項目が選択された時に呼び出されるWAM名。

注：ドロップダウンのクリックでアクションを起動するのは、良いユーザー・インタフェースとは言えません。このような場合は、プッシュ・ボタン、メニュー・アイテム、アンカー(ハイパーリンク)などのデバイスを使用してください。

省略値

\$lweb_WAMName (現在のWAMを示します。)

有効値

単一引用符付きのWAM名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

on_change_wrname

ウェブレット内の項目が選択された時に呼び出されるWebroutine名。

注：ドロップダウンのクリックでアクションを起動するのは、良いユーザー・インタフェースとは言えません。このような場合は、プッシュ・ボタン、メニュー・アイテム、アンカー(ハイパーリンク)などのデバイスを使用してください。

省略値

ブランク。

有効値

単一引用符付きのWebroutine名。このWebroutineは、on_change_wamnameプロパティに指定されたWAMに存在していなくてはなりません。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWebroutineのリストが表示されます。

protocol

このウェブレットに呼び出されたWebroutineへの移動に使用するプロトコル(例えば、http:// もしくは https://)

省略値

空白。現在使用されているプロトコルが使用されます。

有効値

単一引用符付きの、有効なプロトコル。通常は'http:'または'https:'です。

target_window_name

応答HTMLが表示される、ウィンドウやフレームの名前。一意の名前を入力するか、事前定義された値から選択してください。

省略値

空白 - 応答HTMLが現在のウィンドウに表示されます。

有効値

単一引用符で囲まれたウィンドウ名またはフレーム名。

プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用可能なウィンドウやフレームが表示されます。または一意の名前を入力することもできます。

'_blank'は新しいウィンドウに表示します。

'_media'は現在のウィンドウのメディア・パネルに表示します。

'_search'は現在のウィンドウの検索パネルに表示します。

'_parent'は親ウィンドウ（多くの場合現ウィンドウ）に表示します。

'_top'は一番上のウィンドウ（多くの場合現ウィンドウ）に表示します。

_search や _media はInternet Explorer 6 でのみサポートされています。

disabled

ウェブレットを使用可または不可のいずれで表示するかを決定するブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false() もしくは有効な式。

class

ウェブレットに適用されるカスケード・スタイル・シートクラス。

省略値

提供されているウェブレットのクラス名。

有効値

単一引用符で囲まれた、現在のカスケード・スタイル・シートの有効なクラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

mouseover_class

マウスがその上に置かれた時にウェブレットに適用されるカスケード・スタイル・シートのクラス名。

省略値

空白。

有効値

単一引用符で囲まれた、現在のカスケード・スタイル・シートの有効なクラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。'std_dropdown_mouseover'のクラスが提供されています。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

vf_wamevent

VLF のWAM イベント文字列です。

省略値

空白。

有効値

文字列。カンマ (,) は使用できません。

8.1.10 動的選択ボックス (std_dynamic_select)

クイック・スタート - 動的選択ボックス プロパティ - 動的選択ボックス

動的選択ボックスは、別のフィールドをモニターして、そのフィールドが変更されると自動的に自身を更新することができる、HTMLの<select>エレメントです。(つまり、ドロップダウンまたはリストを作成できます。) リストに使用する値は、作業リストの値や、ウェブレットのitemsプロパティ経由で定義された静的な値も利用できます。

クイック・スタート - 動的選択ボックス

動的選択ボックスの各エントリーは、作業リストのエントリーもしくはこのウェブレットのプロパティでハードコーディングされた一連の項目によって定義されます。

作業リストを使用する場合

ドロップダウン・オプションの定義に作業リストを使用する場合、Webroutineを作成して選択された値を格納するフィールドと選択オプションを含む作業リストをWEB_MAPに定義する必要があります。作業リストは*JSONリストとして定義されていなければなりません。作業リストには通常、次のような2つまたは3つのカラムがあります。

キャプションのカラムには、リストに表示する値が含まれます。

コードのカラムには、各キャプションに関連付けられたコードが格納されます。このコードがユーザー選択の値として送り返されます。

オプションでセレクターのカラムもあります。これは selectorValueFieldと共に使用され、ユーザーに表示するリストのフィルター処理を行います。

1. 作業リストがWEB_MAPで*HIDDENではない場合は、省略値で作業リストを表わすテーブルがWebページに含まれます。リストをビジュアライズするテーブルは削除します。削除するには、リストで右クリックし、ポップアップ・メニューから、[全てのリスト削除]を選択してください。
2. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[動的選択ボックス]ウェブレットを探します。
3. 動的選択ボックス ウェブレットを、値を格納するフィールド上にドラッグし、マウスの左ボタンを離してください。
4. ウェブレットをクリックして、[詳細]タブを確認します。nameプロパティとvalueプロパティにウェブレットをドラッグしたフィールドを示す値が設定されていることを確認してください。Webページ表示の際、valueプロパティのフィールドに現在入っている値を使用して選択されたエントリーが設定されます。このウェブレットの値が変更されると、適切な値がnameプロパティに指定されたフィールド（この場

合、同じフィールド)に入れられます。

5. listnameプロパティをWEB_MAPに送られる作業リストに変更します。
6. codefieldプロパティとcaptionfieldプロパティに作業リストの適切なフィールドをセットします。
7. sizeプロパティにリストボックスの希望の高さを行数で設定します。(1を設定するとドロップダウンとして表示されます。)

itemsプロパティを使用する場合

このウェブレットのプロパティでハードコーディングされた一連の項目を使用するには、WEB_MAPで選択された値のフィールドを指定するWebroutineを作成する必要があります。LANSAエディターでWebroutine用に生成されたXSLを開き、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[動的選択ボックス]ウェブレットを探します。
2. 動的選択ボックス ウェブレットを、値を格納するフィールド上にドラッグし、マウスの左ボタンを離してください。
3. ウェブレットをクリックして、[詳細]タブを確認します。nameプロパティとvalueプロパティにウェブレットをドラッグしたフィールドを示す値が設定されていることを確認してください。Webページ表示の際、valueプロパティのフィールドに現在入っている値を使用して選択されたエントリーが設定されます。このウェブレットの値が変更されると、適切な値がnameプロパティに指定されたフィールド(この場合、同じフィールド)に入れられます。
4. itemsプロパティの省略記号(...)のボタンを選択して、ドロップダウン・オプションとして使用する項目のリストを設定します。そしてドロップダウンに必要なエントリーの定義に進みます。

自動更新

動的選択ボックスは別のフィールドをモニターし、そのフィールドが更新される度に自身も自動的に更新することができます。ウェブレットが作業リストを使って埋められた場合、新しい作業リストのコピーを出力するJSONWebroutineを作成する必要があります。ウェブレットは更新が必要な場合に毎回このWebroutineを呼び出します。

1. updateOnFieldChangeにモニターしたいフィールドのIDを設定しま

す。通常はフィールド名ですが、Changeイベントを生成できるHTMLフォーム・エレメントのIDにすることもできます。このフィールドは新しいリストを提供するWebroutineに送られます。

2. [updateWamName](#)と[updateWrName](#)プロパティに新しいリストを提供するWAM/Webroutineの名前を指定します。このWebroutineではlistnameプロパティで定義されたリストの新しいコピーを出力しなければなりません。

3. [updateFieldsToSubmit](#)プロパティを設定して、更新Webroutineに送信する入力値が識別できるようにします。

リスト更新には、更新Webroutineを必ず使用しなければならない訳ではありません。多くの単純なケースでは、メインのWebroutineの最初のリスト出力で可能な全ての値をウェブレットに提供して、[selectorField](#)を使ってリストをフィルターすることも可能です。サーバーに要求を送信してその応答を待つのを避けるため、ウェブレットには最初のリストが記憶されており、更新が行われる時に[selectorField](#)フィルターを再度適用できます。

例えば、[updateOnFieldChange](#)と[selectorValueField](#)プロパティの両方で同じフィールドが指定されている場合、ユーザーがこのフィールドを変更する度に再びフィルターされます。

いずれのアプローチを取るかはバランスを考えて決定します。[selectorField](#)を使用することにより、より速くリスト更新ができますが、より多くのデータをブラウザに送信しないといけないため、最初のページのロードに時間がかかる可能性があります。一方で更新Webroutineの使用により、最初のロード時間は短くなり、リアルタイムの最新データ取得が可能になったり、リスト構造により複雑なロジックを含めることが可能になりますが、リストがサーバーから取り出される間に遅延が生じる場合があります。

以下も参照してください。

[プロパティ - 動的選択ボックス](#)

プロパティ - 動的選択ボックス

動的選択ボックスのプロパティは以下のとおりです。

allowMultiSelect	multiSelectListname	selectorValueField
captionField	name	size
class	onChangeExtraFields	tabIndex
codeField	onChangeFormname	updateFieldsToSubmit
disabled	onChangeProtocol	updateOnFieldChange
display_mode	onChangeTarget	updateProtocol
hide_if	onChangeWamName	updateWamName
id	onChangeWrName	updateWrName
items	position	value
listname	selectorField	vf_wamevent
multiSelectCodefield		width

name

ウェブレットの名前。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。このウェブレットがフィールドの上にドロップされた場合、またはフィールドを表示したり、フィールドに値を追加するために使用される場合は、フィールド名が使用されます。

省略値

自動的に生成された、一意の識別子。

有効値

アルファベット文字 ([A-Za-z]) で始まる文字列で、文字、数字([0-9])、ハイフン("-") もしくはアンダースコア("_") がその後に続くもの。(文字数に制限はありません)

id

ウェブレットの一意のID。省略値はnameプロパティと同じ値で、通常はそのままにしておきます。特別な環境では、同じフィールドをビジュアライズする複数のフォームに複数のウェブレットがある場合があるかもしれません。このような場合は、それぞれに一意のIDを付けてこのプロパティを設定する必要があります。

省略値

\$name、nameプロパティと同じです。

有効値

アルファベット文字 ([A-Za-z]) で始まる文字列で、文字、数字([0-9])、ハイフン("-") もしくはアンダースコア("_") がその後続くもの。(文字数に制限はありません)

value

ウェブレットに設定する値。このウェブレットによりフィールドがビジュアライズされた場合、これがフィールド値もしくは省略値になります。

省略値

ブランク。

有効値

テキスト、もしくは複数言語対応テキスト変数、システム変数、フィールドの名前(プロパティ・シートで省略記号のボタンをクリックしてリストから選ぶことができます。)

size

ウェブレットの高さ（行数で表現）。値が1の場合、ウェブレットはドロップダウン・リストとして表示されます。1以上の場合はリストボックスとして表示されます。

省略値

ウェブレットはドロップダウン・リストとして表示されます。

有効値

0より大きい整数。

display_mode

ウェブレットが入力内容を受け取るのか、出力内容を表示するのかを制御します。

省略値

'input'

有効値

'input' または 'output'

hide_if

ブール値プロパティです。評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

items

ウェブレットに表示する項目を指定するXMLノード・セット。設定はデザイナーでのみ可能です。デザイナーを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使用してください。項目がlistnameプロパティに指定されたリストから追加される場合は、ブランクにしてください。

省略値

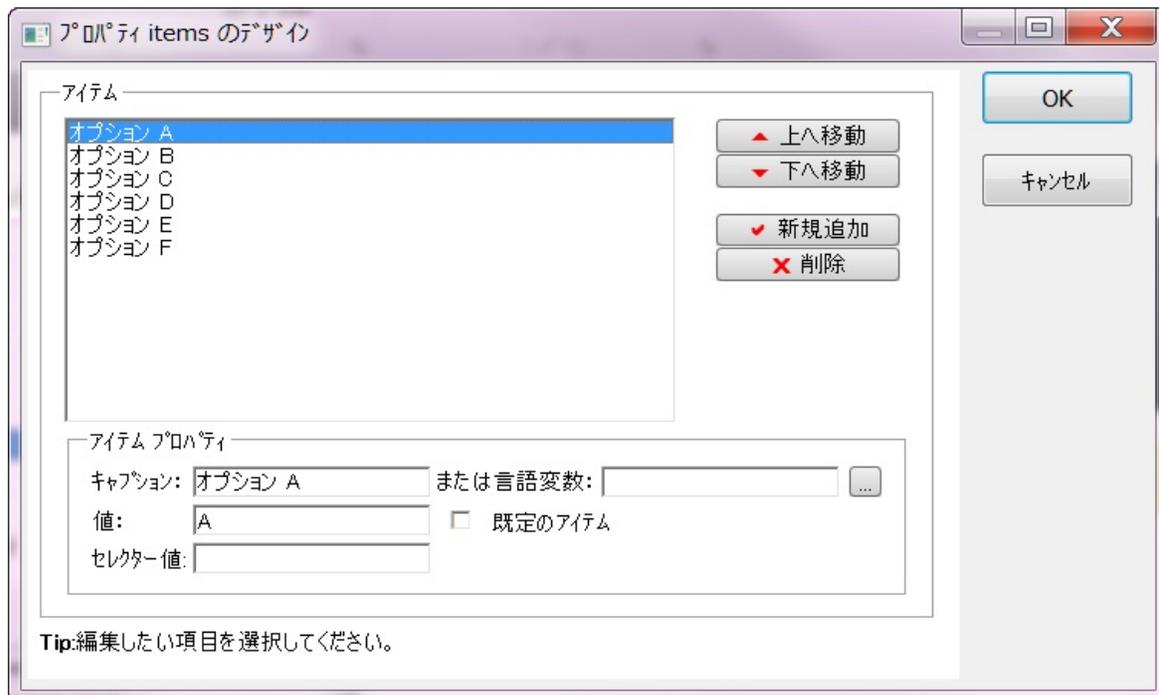
document(")/*/lxml:data/lxml:select (このウェブレットに項目が何も定義されていないことを示します。)

有効値

適用不可。(この値はシステムにより保守されます。)デザイナーを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使用してください。

例

次の例ではitemのプロパティ・エディターが示されています。



ここではリストは6つの項目から構成されています。値が事前に選択されていない時に選択された状態にする項目には[規定のアイテム]のチェックボックスを使用します。セレクター値を使って、実行時にリストをフィルターして表示する値を絞ることもできます。

listname

ウェブレット・リストにデータを追加するために使用する作業リスト名。itemsプロパティに詳細が指定されている場合は、ブランクのままにしてください。listnameとitemsプロパティの両方が指定されている場合は、listnameプロパティが優先されます。

省略値

ブランク。

有効値

ブランク、または現在のWebroutineで*JSONとして定義された出力作業リストの名前。

selectorField

セレクター値を含む[listname]作業リストにあるフィールド名。セレクター値を使って作業リストにフィルターをかけ、実行時に実際に表示する絞られた値のリストにします。selectorFieldカラムの値が selectorValueFieldの値に一致すると、このエントリーがリストに含まれます。

省略値

空白。

有効値

[listname]作業リスト内のフィールド。プロパティシートで該当するドロップダウン・ボタンをクリックして値を選択することができます。

selectorValueField

ウェブレットに提供されたリストを表示用に絞るために、フィルターとして使用する値を持つフィールドの名前。表示リスト作成時は、この値はリストの"セレクター"カラムの値と比較されます。一致した場合、そのエントリーが表示リストに含まれます。

これはサーバー側の作業を減らす有効な方法です。毎回実行時にリスト・エントリーを計算する代わりに、Webroutineで全ての可能な値とセレクター値を含むリストを事前に作成して出力することができます。そして、ブラウザがセレクター値を元にリストを絞って小さくします。

これにより、サーバー要求をすることなく、動的にリストを更新することも可能になります。更新用にモニターされるフィールドもまた selectorValueFieldである場合、ウェブレットは最初に渡されたリストに新しいセレクター値を適用して、ウェブレット自身を再構築することができます。

省略値

ブランク。フィルターはされません。

有効値

現在のWebroutineの出力フィールドの名前。

codeField

各リスト項目のキー値を保持する[listname]作業リスト内のフィールド名。

省略値

空白。

有効値

[listname]作業リスト内のフィールド。プロパティシートで該当するドロップダウン・ボタンをクリックして値を選択することができます。

captionField

各リスト項目のキャプション値を保持する[listname]作業リスト内のフィールド名。

省略値

空白。

有効値

[listname]作業リスト内のフィールド。プロパティシートで該当するドロップダウン・ボタンをクリックして値を選択することができます。

allowMultiSelect

リスト・ボックスで複数選択を許可するかどうかを制御するブール値プロパティ。複数選択が許可される場合、multiSelectListnameとmultiSelectCodefieldプロパティが指定されていなければなりません。複数選択が可能なのはリストボックスだけです。sizeプロパティが1の場合、このプロパティは無視されます。

省略値

false() - リスト・ボックスで選択できるのは1つのみになります。

有効値

true()、false()、もしくは、TrueもしくはFalseを返す有効な式。

multiSelectListname

リストボックスの選択されたエントリーを含む作業リスト。作業リストはmulti_select_codefieldプロパティに指定されたコード・フィールドのみを含んでいなければなりません。allowMultiSelectプロパティがfalse、もしくはsizeプロパティが1の場合、このプロパティは無視されます。

省略値

ブランク - リスト・ボックスで選択できるのは1つのみになります。

有効値

作業リスト名。プロパティ・シートで該当するドロップダウン・ボタンをクリックして、使用できる作業リストのリストから選択できます。

multiSelectCodefield

選択されたリストボックス項目のコード値を保持する
[multiSelectListname]作業リスト内のフィールド名。

省略値

blank - リスト・ボックスで選択できるのは1つのみになります。

有効値

フィールド名プロパティ・シートの該当するドロップダウン・ボタンをクリックして、使用できるフィールドのリストから選択できます。

onChangeWamName

ウェブレット内の項目が選択された時に呼び出されるWAM名。

省略値

\$lweb_WAMName (現在のWAMを示します。)

有効値

WAMの名前。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

onChangeWrName

ウェブレット内の項目が選択された時に呼び出されるWebroutine名。

省略値

空白。

有効値

Webroutineの名前。このWebroutineは、onChangeWamNameプロパティに指定されたWAMに存在してはいけません。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWebroutineのリストが表示されます。

onChangeFormname

onChangeWebroutineを呼び出す際にサーバーに送られるHTMLフォーム名。通常はこのプロパティを変更する必要はありません。複数のフォームが存在する高度なアプリケーションの場合は、正しいフォームが設定されていることを確認してください。

省略値

'LANSA' (document.LANSAのことです。)

有効値

フォームの名前。プロパティ・シート内で該当するドロップダウン・ボタンをクリックすると、使用できるフォーム名リストが表示されます。

onChangeExtraFields

onChangeWebroutineに送信する追加のフィールド（送信されたフォームに存在しないもの）を指定したXMLノードセット。これは通常ウェブレットがリストやグリッドで使用される時によく使用され、リストの別のカラムからの値を指定します。

省略値

document("/*/lxml:data/lxml:json[not(@id)](このウェブレットに定義された項目がないことを示します。)

有効値

適用不可。（この値はシステムにより保守されます。）デザイナを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使用してください。

例

次の例ではonChangeExtraFieldsのプロパティ・エディターが示されています。

```
Webroutine Name(CheckConsignment)
  Web_Map For(*input) Fields(#WDCONSIGN)

  * see if this consignment exists in the consignment status file.
  Check_For In_File(WDCONST) With_Key(#WDCONSIGN)

  * it exists ...
  If_Status Is(*EQUALKEY)

    * show the consignment status.
    Transfer Toroutine(ShowConsignment)

  Else

    * it doesn't exist: show an error message ...
    Message Msgbxt('Invalid Consignment Note number. Please try again.')

    * and re-display the main page.
    Transfer Toroutine(ConsignmentEnquiry)

  Endif
Endroutine
```

ここでは現在のWebroutineの出力フィールド("Value"欄)を、onChangeWebroutineのWEB_MAPで定義された異なる名前("Name"欄)の入力フィールドにマップする方法を示しています。

onChangeProtocol

onChangeWebroutineを呼び出す際に使用するプロトコル(例えば、`http://` もしくは `https://`)

省略値

空白。現在使用されているプロトコルが使用されます。

有効値

単一引用符付きの、有効なプロトコル。通常は'`http:`'または'`https:`'です。

onChangeTarget

応答HTMLが表示される、ウィンドウやフレームの名前。一意の名前を入力するか、事前定義された値から選択してください。

省略値

空白 - 応答HTMLが現在のウィンドウ/フレームに表示されます。

有効値

単一引用符で囲まれたウィンドウ名またはフレーム名。

プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用可能なウィンドウやフレームが表示されます。または一意の名前を入力することもできます。

'_blank'は新しいウィンドウに表示します。

'_media'は現在のウィンドウのメディア・パネルに表示します。

'_search'は現在のウィンドウの検索パネルに表示します。

'_parent'は親ウィンドウ（多くの場合現ウィンドウ）に表示します。

'_top'は一番上のウィンドウ（多くの場合現ウィンドウ）に表示します。

_search や _media はInternet Explorer 6 でのみサポートされています。

position

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていなければならないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付けた有効な寸法。

width

Webページのウェブレットの幅です。

省略値

ブランク（ウェブレットは省略値の幅を使用します。）

有効値

有効な範囲内の幅。

disabled

ウェブレットを使用可または不可のいずれで表示するかを決定する
ブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false() もしくは有効な式。

class

ウェブレットに適用するCSSクラスです。

省略値

空白。

有効値

現在のカスケード・スタイル・シートの有効なクラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

tabIndex

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtabIndexプロパティの値により決定されます。

1. 正数のtabIndexを持つオブジェクトがtabIndexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク (省略値) のtabIndexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtabIndexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

updateOnFieldChange

変更をモニターするフィールドのID。モニターするフィールドが変更されると、選択ボックスは更新されます。

省略値

空白。

有効値

"change"イベントを生成できる現在のページ上にあるフィールドID。つまり、テキスト入力フィールドまたは選択エレメント（ドロップダウン・リストやリストボックス）です。updateWamNameとupdateWrNameが指定されている場合、ウェブレットはWebroutineを呼び出して、[listname]作業リストの新しいコピーを要求します。それ以外の場合は、リストに既存のselectorValueフィルターを最適用させて、そこから選択リストを再構築します。

updateWamName

リストを更新する時に呼び出されるWAMの名前。

省略値

\$lweb_WAMName (現在のWAMを示します。)

有効値

WAMの名前。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

updateWrName

リストを更新する時に呼び出されるWebroutineの名前。Webroutineは*JSONとして定義されていなければなりません。

省略値

空白。

有効値

Webroutineの名前。このWebroutineは、updateWamNameプロパティに指定されたWAMに存在してはいけません。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWebroutineのリストが表示されます。

updateFieldsToSubmit

更新Webroutineに送信するフィールドを指定するXMLノードセット。

省略値

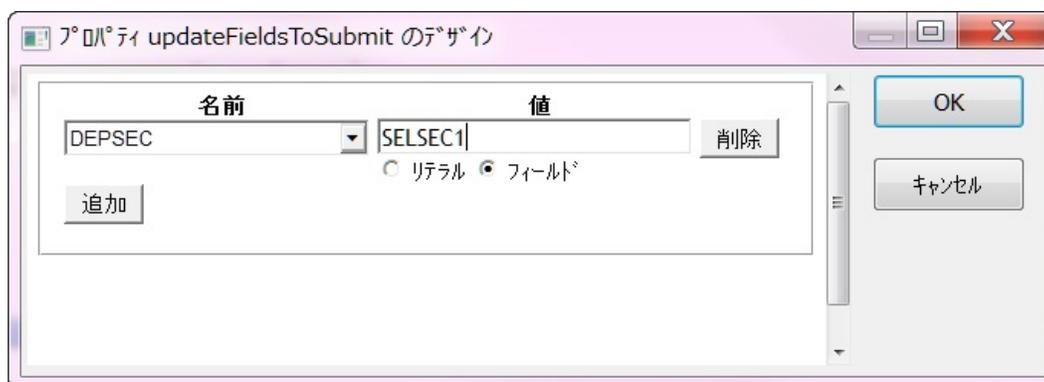
document(")/*/lxml:data/lxml:json (このウェブレットに項目が何も定義されていないことを示します。)

有効値

適用不可。(この値はシステムにより保守されます。)デザイナを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使用してください。

例

次の例ではupdateFieldsToSubmitのプロパティ・エディターが示されています。



ここでは現在のWebroutineの出力フィールド（[値] 欄）を、更新WebroutineのWEB_MAPで定義された異なる名前（[名前] 欄）の入力フィールドにマップする方法を示しています。

updateProtocol

更新Webroutineを呼び出す際に使用するプロトコル(例えば、http:// もしくは https://)

省略値

空白。現在使用されているプロトコルが使用されます。

有効値

単一引用符付きの、有効なプロトコル。通常は'http:'または'https:'です。

vf_wamevent

VLF xのWAM イベント文字列です。

省略値

空白。

有効値

文字列。カンマ (,) は使用できません。

8.1.11 Excelにエクスポート (std_toexcel)

[クイック・スタート - Excelにエクスポート](#)

[プロパティ - Excelにエクスポート](#)

Excelにエクスポートのウェブレットにより、テーブルまたはグリッドをExcelスプレッドシートにエクスポートできます。

Export to Excel

クイック・スタート - Excelにエクスポート

ページにテーブルやグリッドに表形式データがあり、ユーザーがスプレッドシートを使って操作したい場合、通常はこのウェブレットを使用します。

このウェブレットはActiveXを使用しており、マイクロソフトのInternet Explorerでのみ作動します。ブラウザがActiveXをサポートしない場合、このウェブレットは使用できません。

このウェブレットは出力フィールドのみを含むテーブルで使用します。テーブルにウェブレットや入力エレメントが含まれる場合、期待通りの結果が得られません。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[Excelにエクスポート]ウェブレットを探します。
2. このウェブレットをWebページにドラッグ・アンド・ドロップしてください。
3. listnameプロパティにエクスポートできるようにしたいリストの名前を設定します。
4. startingColumnIndexプロパティにエクスポートに含む最初のカラムのインデックスを設定します。（最初のカラムのインデックスは0です。）
5. numberOfColumnsプロパティにエクスポートするカラムの数を設定します。
6. 必要であれば、captionプロパティを変更します。

プロパティ - Excelにエクスポート

Excelにエクスポートのプロパティは以下の通りです。

caption	listname	startingColumnIndex
disabled	name	tab_index
height_design	numberOfColumns	text_class
hide_if	pos_absolute	title
		width_design

name

ウェブレットの名前。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript や ウェブレットを参照するXSLを使用する場合は、独自の名前を使用することも可能です。

省略値

`concat('o', position(), '_LANSA_n')` - LANSAにより付けられたウェブレット内部名です。

有効値

単一引用符で囲まれた名前。

listname

エクスポートする作業リストの名前。このプロパティは必須です。

省略値

空白。

有効値

単一引用符で囲まれた文字列。使用可能な（WAMで定義された）作業リストの一覧は、プロパティシートで該当するドロップダウン・ボタンをクリックすることで表示され、選択することができます。

startingColumnIndex

エクスポートを開始するカラムのインデックス。最初のカラムのインデックスはゼロです。

省略値

0

有効値

整数値テーブルのカラム数より少ない数である必要があります。

numberOfColumns

エクスポートに含むカラムの数。

省略値

last – startingColumnIndexで指定された最初のカラムからテーブルの最後のカラムまでの全てのカラムです。

有効値

整数値startingColumnIndexで指定されたカラムから開始し、最後のカラムまでの数を超えることはできません。

caption

ウェブレットのキャプション。

省略値

'Export to Excel'

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width_design

Webページのウェブレットの幅です。

省略値

ブランク（ウェブレットは省略値の幅を使用します。）

有効値

単一引用符で囲まれた有効な範囲内の幅。

height_design

Webページのウェブレットの高さ。

省略値

ブランク(ウェブレットは省略値の高さを使用します。)

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

disabled

ウェブレットを使用可または不可のいずれで表示するかを決定するブール値プロパティ。ブラウザがActiveXをサポートしない場合、このウェブレットは自動的に使用不可になります。

省略値

ブランク - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false()、もしくは、TrueかFalseを返す有効な式。

title

マウスがウェブレットの上を通る時に、ウェブレットのヒントとして表示されるテキスト。

省略値

空白 - ヒントは何も表示されません。

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

text_class

ウェブレットのテキストのカスケード・スタイル・シート(CSS)クラス名です。

省略値

空白 - テキスト・クラスはありません。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

8.1.12 グリッド (std_grid_v2 and std_grid_v3)

クイック・スタート - グリッド プロパティ - グリッド

グリッド・ウェブレットは、ソート可能なグリッド・コントロールを提供します。グリッドのセルは作業リストあるいはXMLから挿入されます。グリッド・エレメントは、HTML <table>として組込まれます。jQuery UIベースのウェブレットを使用したい場合は、std_grid_v3ウェブレットを使用してください。例えば次のようなものです。

Delete?	Skill Code	Skill Description
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>	4GL	4GL PROGRAMMING
<input type="checkbox"/>	AC	ACCOUNTANCY DEGREE
<input type="checkbox"/>	ADMIN1	ADMINISTRATN PART 1
<input type="checkbox"/>	ADMIN2	ADMINISTRATN PART 2
<input type="checkbox"/>	ADVPGM	ADVANCED PROGRAMMING
<input type="checkbox"/>	CL	CL PROGRAMMING
<input type="checkbox"/>	COM	COMMUNICATIONS DEGREE
<input type="checkbox"/>	CS	COMPUTER SCIENCE DEGREE

クイック・スタート - グリッド

このウェブレットを使用する時は、通常WEB_MAPに作業リストを指定するWebroutineを作成します。LANSAエディターでグリッド・ウェブレットを使って、作業リスト内のデータを以下の手順で表示することができます。

1. 作業リストがWEB_MAPで*HIDDENではない場合は、省略値で作業リストを表すテーブルがWebページに含まれます。リストをビジュアライズするテーブルは削除します。削除するには、リストで右クリックし、ポップアップ・メニューから、[全てのリスト削除]を選択してください。
2. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[グリッド]ウェブレットを探します。
3. [デザイン]ビューでウェブレットをページにドラッグ・アンド・ドロップしてください。ウェブレットを選択して、[詳細]タブをクリックするのを忘れないでください。
4. listnameプロパティをWEB_MAPに送られる作業リストに設定します。グリッド表示にはすぐに作業リストの詳細が反映されます。
5. オプションとしてWebページのデザインに合うように、heightプロパティやwidthプロパティを設定することができます。

ウェブレットのグリッド・カラムのカスタマイズについての詳細については、「[グリッド・カラムをカスタマイズする](#)」を参照してください。

プロパティ - グリッド

グリッド・ウェブレットのプロパティは以下のとおりです。

allowColResize	hide_header_if_empty	pos_absolute
allowSort	hide_if	rowHoverEffect
even_row_class	listname	selectableRow
formname	listname_fixed_col_field	show_header
grid_col_properties	name	sort_fixed_rows_with_body
grid_hdr_properties	odd_row_class	width
height	onRowClickJS	

name

グリッドの名前。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やグリッドを参照するXSLを使用する場合は、独自の名前を使用することもできます。

省略値

自動的に生成された、一意の識別子。

有効値

単一引用符で囲まれた文字列。

listname

グリッドにセルを追加するために使用する作業リストの名前。指定する場合は、or_list_xmlプロパティはブランクのままにしてください。

省略値

ブランク。

有効値

単一引用符で囲まれた文字列。使用可能な（WAMで定義された）作業リストの一覧は、プロパティシートで該当するドロップダウン・ボタンをクリックすることで、選択することができます。

listname_fixed_col_field

listnameに指定されたリストに関連付けられたフィールドで、最も左の固定の(スクロールできない)カラムに表示されるエントリーを含むフィールド名を指定します。

省略値

ブランク。

有効値

単一引用符で囲まれた文字列。プロパティ・シートで該当するドロップダウン・ボタンをクリックすると、listnameに指定された作業リストからのフィールドを選択できます。

sort_fixed_rows_with_body

ブール値プロパティです。false()の場合、listname_fixed_col_fieldプロパティに指定された固定カラムのエントリは、グリッド内の他のカラムがソートされてもその行の他のカラムと一緒に動きません。

省略値

True()

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

例

次の例では、フィールド#STD_FLAGの値が'X'に等しい場合、固定カラムは他のカラムと共にソートされません。式は入力されなくてはなりません。またプロパティにフォーカスがある時、以下のように表示されます。

```
listname_fixed_col_field #STD_FLAG = 'X'
```

プロパティがフォーカスを失うと、この式は以下のように表示されません。

```
listname_fixed_col_field key('field-value', 'STD_FLAG') = 'X'
```

grid_hdr_properties

グリッドの列見出しのプロパティ設定に使用されます。設定はデザイナーでのみ可能です。デザイナーを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使用してください。

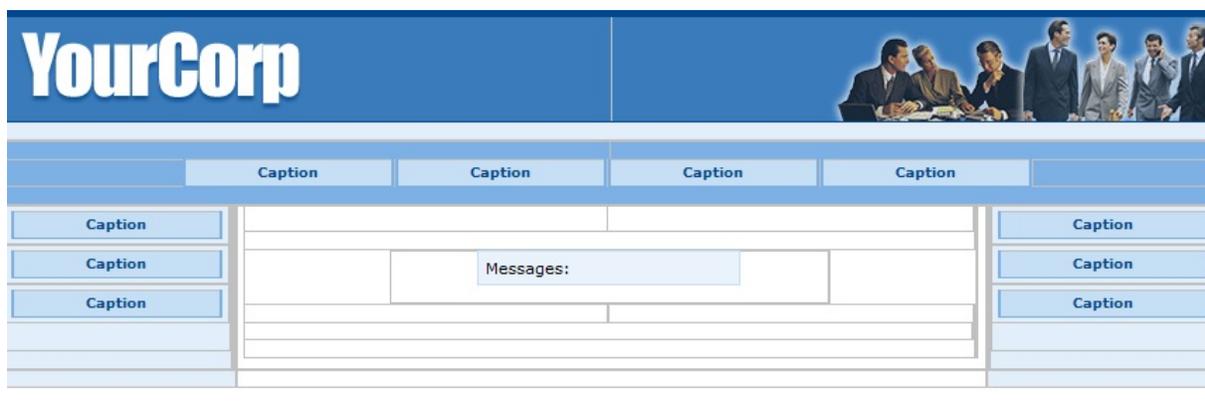
省略値

`document(")/*//xmlns:data/xml:grid[@id='<unique id>']` - この定義に関連する lxml フラグメントが lxml data エレメントの下の現在のドキュメントに格納されており、生成時に一意の ID のグリッドとして識別されることを示しています。

有効値

適用不可。(この値はシステムにより保守されます。)プロパティ・シートの省略記号(...)のボタンを使用すると、グリッド・カラムが選択でき、保守されるプロパティが表示できます。

次の例では、SURNAME フィールドで列見出しをカスタマイズすることができます。また、この列をクリックすると、逆順にソートされます。



例

以下の例では、@id で示される値と一致する一意の識別子が付けられた LXML フラグメントが自動的に現在のドキュメントに生成され、グリッド見出しのプロパティが定義されます。

```
grid_hdr_properties document(")/*//xmlns:data/xml:grid[@id='D574DF799BD145998BE947F0F4D45944']
```

これらの LXML のフラグメントを確認するには、XSL ソースタブを開くか、もしくは一意の識別子で参照して検索します。

grid_col_properties

グリッド・カラムのプロパティ設定に使用されます。設定はデザイナーでのみ可能です。デザイナーを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使用してください。

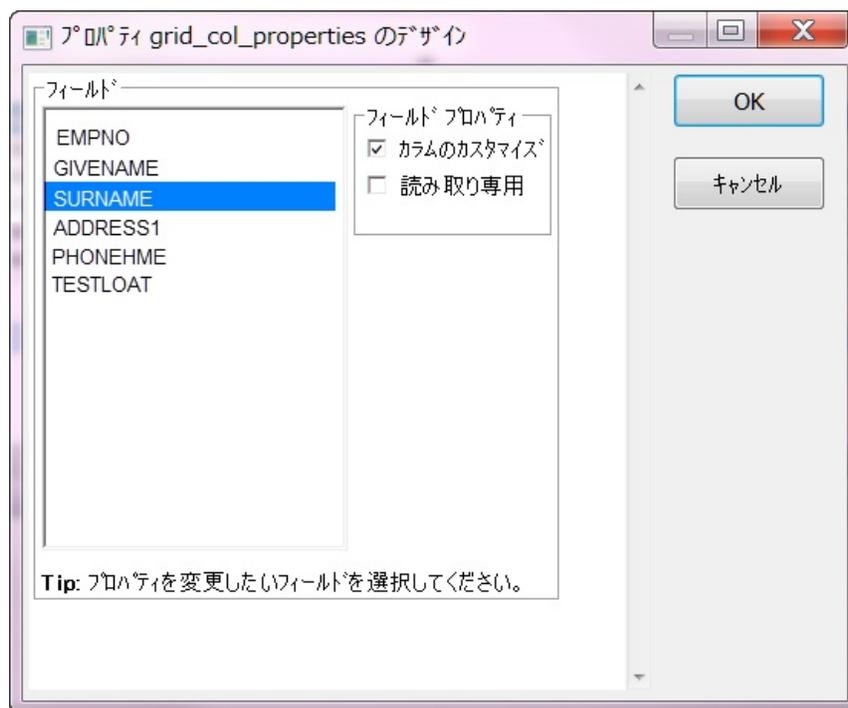
省略値

document("")/*/lxml:data/lxml:grid[@id='<unique id>'] - この定義に関連する lxml フラグメントが lxml data エレメントの下の現在のドキュメントに格納されており、生成時に一意のIDのグリッドとして識別されることを示しています。

有効値

適用不可。(この値はシステムにより保守されます。)プロパティ・シートの省略記号(...)のボタンを使用すると、グリッド・カラムが選択でき、保守されるプロパティが表示できます。

この例では、フィールドSURNAMEのカラムのカスタマイズを行うことができます。これは読み取り専用ではありません。



例

以下の例では、@idで示される値と一致する一意の識別子が付けられた LXML フラグメントが自動的に現在のドキュメントに生成され、グリッド・カラムのプロパティが定義されます。

grid_col_properties

document("*/lxml:data/lxml:grid[@id='7C4F340984F6475BBD90CA28416F3F22']

これらのLXMLのフラグメントを確認するには、XSLソースタブを開くか、もしくは一意の識別子で参照して検索します。

show_header

ブール値プロパティです。false()の場合、列見出しはグリッドに表示されません。

省略値

true()

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

例

次の例はフィールドSTD_FLAGが 'X'と等しい場合にウェブレットを表示します。式は入力されなくてははいけません。またプロパティにフォーカスがある時、以下のように表示されます。

```
show_header #STD_FLAG = 'X'
```

プロパティがフォーカスを失うと、この式は以下のように表示されません。

```
show_header key('field-value', 'STD_FLAG') = 'X'
```

hide_header_if_empty

ブール値プロパティです。true()の場合、リストにエントリーがなければ、列見出しはlistnameプロパティに指定されたグリッドの中には表示されません。

省略値

true()

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

例

この例では、フィールド#STD_FLAGが'X'ではない場合、表示するリスト・エントリーがなくても列見出しを表示します。式は入力されなくてははいけません。またプロパティにフォーカスがある時、以下のように表示されます。

 hide_header_if_empty	#STD_FLAG = 'X'
---	-----------------

hide_if

ブール値プロパティです。評価がTrueの時にウェブレットが非表示になる式です。

省略値

False() (グリッドはいつも表示されます。)

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

例

次の例はフィールド#STD_FLAGが'X'と等しい場合にグリッドを非表示にします。式は入力されなくてはなりません。またプロパティにフォーカスがある時、以下のように表示されます。

hide_if	#STD_FLAG = 'X'
---------	-----------------

プロパティがフォーカスを失うと、この式は以下のように表示されません。

hide_if	key('field-value', 'STD_FLAG') = 'X'
---------	--------------------------------------

even_row_class

偶数グリッド行に適用されるカスケード・スタイル・シート・クラス。

省略値

'even_row'グリッドの偶数列(2列目、4列目など)に適用される省略値のクラスで、全てのカスケード・スタイル・シートと共に提供されています。

有効値

現在のカスケード・スタイル・シートから選択された有効なクラス名を単一引用符で囲んだもの。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

odd_row_class

奇数グリッド行に適用されるカスケード・スタイル・シート・クラス。

省略値

'odd_row'グリッドの奇数列(1列目、3列目など)に適用される省略値のクラスで、全てのカスケード・スタイル・シートと共に提供されています。

有効値

現在のカスケード・スタイル・シートから選択された有効なクラス名を単一引用符で囲んだもの。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

formname

サーバーに送られるHTMLフォーム名です。

省略値

'LANSA' (document.LANSAのことです。)

有効値

単一引用符付きのフォーム名。プロパティ・シート内で該当するドロップダウン・ボタンをクリックすると、使用できるフォーム名リストが表示されます。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク (ウェブレットは相対的に位置付けられます。)

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

Webページのウェブレットの幅です。

省略値

blank (ウェブレットは省略値の幅を適用します。)

有効値

単一引用符で囲まれた有効な範囲内の幅。

height

Webページのウェブレットの高さ。

省略値

blank (ウェブレットは省略値の高さを適用します。)

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

allowSort

ブール値プロパティです。true()の場合、ユーザーは列見出しをクリックすることで、グリッドのコンテンツをソートできます。

省略値

true()

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

allowColResize

ブール値プロパティです。true()の場合、ユーザーはヘッダー・セルの間の罫線をクリックしたり、ドラッグすることで、グリッド・カラムのサイズ調整ができます。

省略値

true()

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

rowHoverEffect

ブール値プロパティです。true()の場合、マウス・ポインターの下の行が強調表示され、グリッドからユーザーへのフィードバックが提供されません。

省略値

false()。

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

selectableRow

ブール値プロパティです。true()の場合、グリッドは最後にクリックされた行を追跡し、その行を強調表示して選択されたことを示します。

省略値

false()。

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

onRowClickJS

ユーザーがグリッド内の行をクリックする時に実行するJavaScriptコード。このコードは、Clickイベントに関して追加情報を提供する、次のような2つの変数を受け取ります。

event - Clickイベントのブラウザ・イベント・オブジェクト。

rowNum - クリックされた行番号。これは作業リスト内の行数で、グリッド内の行数（これはユーザーのソートにより変更される可能性があります。）ではありません。

このコードからfalseを返すことで、行が選択されないようにすることができます。

省略値

空白。

有効値

有効なJavaScriptコード。

例

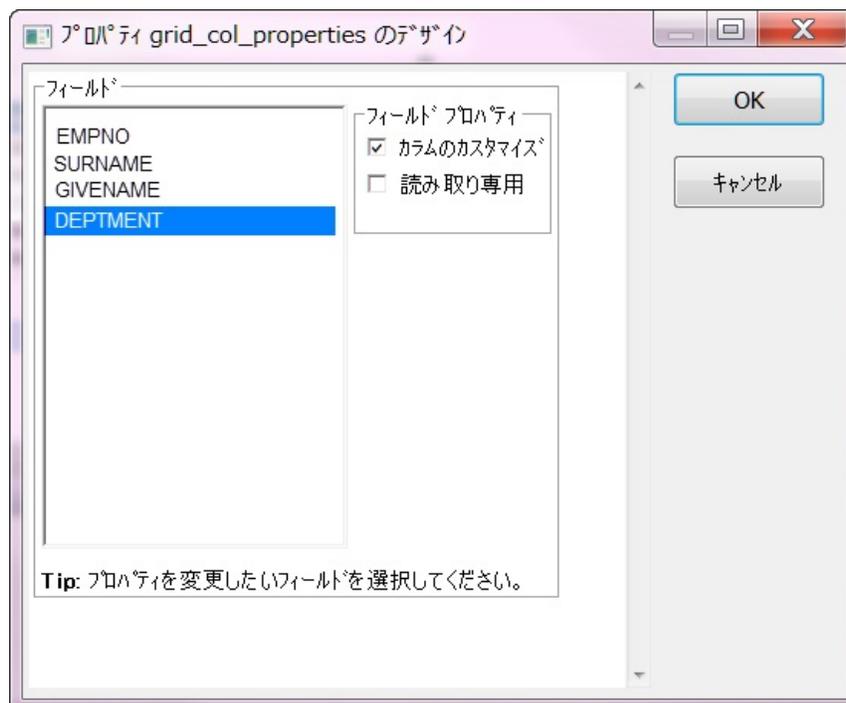
次の例では、クリックされた行で警告を表示し、行番号3が選択されないようにします。

```
alert("Row " + rowNum); return (rowNum != 3);
```

グリッド・カラムをカスタマイズする

ウェブレット・プロパティのカラム値を参照する [グリッド・カラムの例](#)

グリッドはフィールドの省略値ウェブレットを使って、自動的に各カラムをビジュアライズします。時にはウェブレットのプロパティを変更したり異なるウェブレットを使用したい場合もあると思います。このような場合は、`grid_col_properties` プロパティを編集し、カスタマイズしたいカラムの[カラムのカスタマイズ]オプションを選択します。



[OK]をクリックすると、カスタマイズされたカラムはブランクになります。

LANSA

Advanced Software Made Simple

Messages:

Employee List

Employ Number	Surname	Given name(s)	Dept Code
ABCDE	ABCDEFGHIJKLMNOPS	ABCDEFGHIJKLMNOPS	
ABCDE	ABCDEFGHIJKLMNOPS	ABCDEFGHIJKLMNOPS	
ABCDE	ABCDEFGHIJKLMNOPS	ABCDEFGHIJKLMNOPS	

このカラム内の空のセルのいずれかにドラッグすることで、どのウェブレットでも追加することができます。

ウェブレット・プロパティのカラム値を参照する

グリッドのカラム値を参照する方法は、今までとは少々違います。標準リストを使用している場合は、同じ名前のXSLT変数(\$COLUMNNAME)を使ってカラム値を参照することができます。グリッドではこの方法は使えません。カラムにアクセスするには、次のようなXPath式を使う必要があります。

```
../lxml:column[@name='COLUMNNAME']
```

現在のカラム(つまり、ウェブレットを含むカラム)を参照する時はこれは必要ありません。この場合、シングル・ピリオド(.)を使用します。参照したいカラムの位置がわかっている場合は、以下のようなXPath式を使用することもできます。

```
../lxml:column[2]
```

この方法は格段に早いので大きなリストでは便利ですが、リスト内のカラムの順番が変わると問題になる可能性があります。

注：XPath式内の全てのフィールド名とカラム名の参照は大文字を使用しなければなりません。リポジトリ・フィールドへの参照は全て、フィールドのショート・ネームを使用する必要があります。

以下も参照してください。

グリッド・カラムの例

XPath式は、Webroutineにより出力されるXMLデータのカラムの位置をXSLTプロセッサに伝えます。DOSやLinuxのファイル・パスと同様、式は現在位置から対象データまでのパスを示しており、ピリオド(.)は現在の位置を、2つのピリオド(..)は親を参照するのもファイル・パスと同様です。以下はリスト内の1つの行のXML出力です。

```
<lxml:entry>  
  <lxml:column name="EMPNO"  
id="EMPLIST.0002.EMPNO">A0090</lxml:column>  
  <lxml:column name="SURNAME"
```

```
id="EMPLIST.0002.SURNAME">BLOGGS</xml:column>
  <xml:column name="GIVENAME"
id="EMPLIST.0002.GIVENAME">FRED JOHN
ALAN</xml:column>
  <xml:column name="DEPARTMENT"
id="EMPLIST.0002.DEPTEMENT">FLT</xml:column>
</xml:entry>
```

ウェブレットをグリッド・カラム内で処理する場合、XMLの中の現在の位置は<xml:column> タグでこの行と列を表します。XPath式は次のようになります。

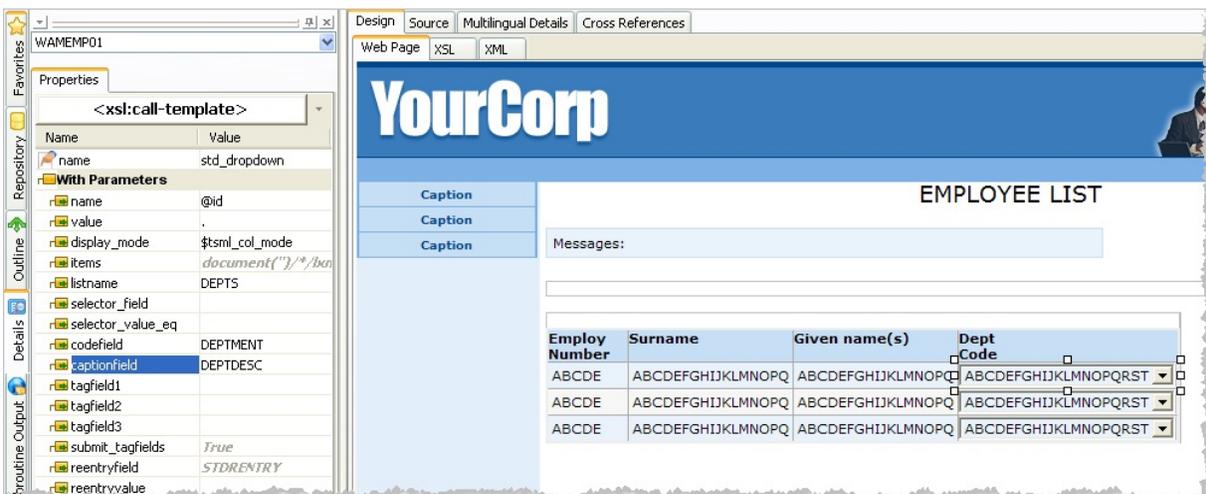
```
../xml:column[@name='EMPNO']
```

これは、親の <xml:entry> タグに戻り、name属性がEMPNOである子の<xml:column> タグを探して、これを戻すことを示しています。

グリッド・カラムの例

次の例では従業員リスト内の部門のカラムにコンボ・ボックスを追加します。#DEPARTMENTカラムを含む#EMPLISTと呼ばれる従業員リストと#DEPARTMENTと#DEPTDESCを含む#DEPTSとよばれる部門リストがあります。

1. グリッドをWebroutineの[デザイン]に追加し、*listname* プロパティにEMPLISTを設定します。
2. *grid_col_properties* プロパティを修正し、DEPARTMENTカラムで[カラムのカスタマイズ]オプションを選択します。
3. コンボ・ボックス・ウェブレットをDEPARTMENTカラムの最初のヘッダー以外のセルにドラッグしてください。
4. ドロップダウンの*listname* プロパティにDEPTSを設定します。
5. *codefield* プロパティにDEPARTMENTを、*captionfield* プロパティにDEPTDESCを設定します。グリッドは、以下のようになります。



The screenshot shows a web application interface for 'YourCorp'. The main content area displays an 'EMPLOYEE LIST' table. The table has four columns: 'Employ Number', 'Surname', 'Given name(s)', and 'Dept Code'. The 'Dept Code' column contains a dropdown menu with a list of department codes (ABCDEF...). The interface also shows a 'Messages' section and a 'Caption' label for the table. On the left side, there is a 'Properties' panel for the selected component, showing various properties like 'name', 'value', 'display_mode', 'items', 'listname', 'selector_field', 'codefield', and 'captionfield'. The 'codefield' property is set to 'DEPARTMENT' and 'captionfield' is set to 'DEPTDESC'.

value プロパティにシングル・ピリオド(.)が含まれていることに注意してください。これはコンボ・ボックスの値がEMPLISTリストの現在のカラム値であることを示しています。

また、*display_mode* プロパティには $\$tsml_col_mode$ が設定されていることにも注目してください。これは特別なXSLT変数で、グリッドの中でDEF_LISTに定義された表示モードを示すために使用されます。

これで、*on_change*アクションがコンボ・ボックスに追加され、値が変更されるとWebroutineが実行されます。UpdateDepartmentWebroutineには2つの入力があります。*EMPNO*と*DEPARTMENT*です。

6. コンボ・ボックスの*on_change_wrname*プロパティにUpdateDepartmentを設定します。
7. コンボ・ボックスの*tagfield1*プロパティにDEPARTMENTを設定します。これはDEPARTMENTカラム(DEPTSリスト内の)で選択された値を実行するようコンボ・ボックスに指示します。
8. コンボ・ボックスの*reentryfield*プロパティにEMPNOを設定し、*reentryvalue*プロパティに../xml:column[@name='EMPNO']を設定します。これはXPath式ですので、値を入力する際は[詳細]タブの下部にあるXPathエントリー・エリアを使用する必要があります。これはグリッドのEMPNOカラムの値を取得し、EMPNOという名前のフィールドに送るようコンボ・ボックスに指示します。

8.1.13 リスト・ページング・イメージ(std_list_images)とリスト・ページング・ボタン(std_list_buttons)

クイック・スタート - リスト・ページング・イメージとリスト・ページング・ボタン プロパティ - リスト・ページング・イメージとリスト・ページング・ボタン

リスト・ページング・イメージ・ウェブレットには、前のページに戻るイメージ、次のページに進むイメージ、新しい検索を開始するイメージの3つのイメージがあります。このウェブレットは、リスト・ビジュアライゼーションの前後に置くように設計されており、リスト情報をページ毎に処理するロジックをサポートしています。

前ページと次ページのイメージの表示には、条件を付けることができます。

このウェブレットは次のようなものです。



通常は以下の図のように利用します。

Dept Code	Department Description
ABCD	ABCDEFGHIJKLMNOPS



リスト・ページング・ボタン・ウェブレットはこれと似ていますが、イメージの代わりにプッシュ・ボタンを使用して移動するところが異なります。

クイック・スタート - リスト・ページング・イメージとリスト・ページング・ボタン

これらのいずれかのウェブレットを使用するには、通常はWEB_MAPに作業リストを指定したWebroutineを作成します。定義された作業リストに送られる1ページ分のエントリを一度に処理するようにページのWAMを設計します。作業リストは結果のWebページ上にテーブルとしてビジュアライズされます。LANSAエディターで以下の手順に従って、このテーブルに行を追加して、ページ移動できるナビゲーションを組み込みます。

1. 生成されたテーブルに行を追加して、作業リストを表示します。これを行うにはリストで右クリックし、ポップアップ・メニューから [Table - 行の追加 - 1] を選択します。
2. テーブルの列が2列以上ある場合、新しく作成された行の最初の列にフォーカスを置いて、[デザイン] タブをクリックしてください。テーブル内に表示する列数を colspan プロパティに設定します。これにより、navigation ウェブレットがテーブルの幅全体を表示できるようになります。
3. [ウェブレット テンプレート] タブをクリックし、上にあるドロップダウン・リストから [標準ウェブレット] を選択し、List Paging Images (もしくは List Paging Buttons) ウェブレットを探します。
4. [デザイン] ビューで、新しく作られたテーブル列にドラッグ・アンド・ドロップしてください。また、このウェブレットを選択して [詳細] タブをクリックするのを忘れないでください。
5. on_page_wname プロパティに、前ページもしくは次ページのイメージ(ボタン)がクリックされた時に呼び出される Webroutine を設定します。
6. on_search_wname プロパティに、検索のイメージ(ボタン)がクリックされた時に呼び出される Webroutine を設定します。
7. 適切なフィールド名を reentryfield プロパティに設定する、もしくは省略値フィールドの STDREENTRY を使用します。このフィールドは RDMLX コードの中で、前ページと次ページの処理を区別する際に必要になります。

プロパティ - リスト・ページングイメージとリスト・ページングボタン

リスト・ページング・イメージ及びボタンのウェブレットのプロパティは以下の通りです。

<code>class</code> (std_list_buttonsのみ)	<code>nextcondfield</code>	<code>protocol</code>
<code>formname</code>	<code>on_click_wamname</code>	<code>reentryfield</code>
<code>height_design</code>	<code>on_page_wrname</code>	<code>show_first_last</code>
<code>hide_if</code>	<code>on_search_wrname</code>	<code>tab_index</code>
<code>image_size</code>	<code>page_count_fieldname</code>	<code>vf_wamevents</code>
<code>mouseover_class</code>	<code>pos_absolute_design</code>	<code>width_design</code>
<code>name</code>	<code>prevcondfield</code>	

name

ウェブレットの名前。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript や グリッドを参照するXSLを使用する場合は、独自の名前を使用することもできます。

省略値

自動的に生成された、一意の識別子。

有効値

単一引用符で囲まれた文字列。

image_size

ピクセルで表された、ウェブレット・イメージのサイズ。次の4つのサイズが使用できます。16x16、24x24、32x32、48x48

省略値

16

有効値

16、24、32または48

prevcondfield

「前ページ」イメージ/ボタンを表示するかどうかを決定するフィールド名。ブランク以外の値の場合はボタンを表示します。

省略値

'STDPREV'

有効値

リポジトリもしくはWAMで定義されたフィールド名。プロパティシート内で該当するドロップダウン・ボタンをクリックすると、使用できるフィールド名リストが表示されます。

nextcondfield

「次ページ」イメージ/ボタンを表示するかどうかを決定するフィールド名。ブランク以外の値の場合はボタンを表示します。

省略値

'STDMORE'

有効値

リポジトリもしくはWAMで定義されたフィールド名。プロパティシート内で該当するドロップダウン・ボタンをクリックすると、使用できるフィールド名リストが表示されます。

show_first_last

ブール値プロパティです。true()をセットして、最初のページと最後のページボタンを使用可能にします。

省略値

false() (最初のページと最後のページボタンは表示されません。)

有効値

True()、false()、もしくは、フィールド名、リテラル、XSL変数を使用した、true() もしくは false()で答えが得られる有効な式。

reentryfield

on_page_wnameに指定されたWebroutineに処理が受け渡された時、どのボタンが選択されたかを示すのに使用されるフィールド名。

ボタン reentryfieldの値

最初のページ F

前ページ P

次ページ M

最後のページ L

省略値

'STDREENTRY'

有効値

リポジトリもしくはWAMで定義されたフィールド名。プロパティシート内で該当するドロップダウン・ボタンをクリックすると、使用できるフィールド名リストが表示されます。

hide_if

ブール値プロパティです。trueと評価された時、ウェブレットを非表示にする式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

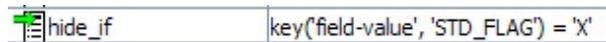
例

次の例はフィールド#STD_FLAGが 'X'と等しい場合にウェブレットを非表示にします。式は入力されなくてはなりません。またプロパティにフォーカスがある時、以下のように表示されます。



hide_if #STD_FLAG = 'X'

プロパティがフォーカスを失うと、この式は以下のように表示されます。



hide_if key('field-value', 'STD_FLAG') = 'X'

formname

サーバーに送られるHTMLフォーム名です。

省略値

'LANSA' (document.LANSAのことです。)

有効値

単一引用符付きのフォーム名。プロパティ・シート内で該当するドロップダウン・ボタンをクリックすると、使用できるフォーム名リストが表示されます。

pos_absolute_design

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width_design

Webページのウェブレットの幅です。

省略値

'0%'(ウェブレットが省略値の幅100%を採用します。)

有効値

シングル・クォーテーション付きの、有効な範囲にある、幅。

height_design

Webページのウェブレットの高さ。

省略値

ブランク(ウェブレットは省略値の高さを使用します。)

有効値

シングル・クォーテーション付きの、有効な範囲にある、高さ。

on_click_wamname

イメージ/ボタンがクリックされたときに呼び出されるWAMの名前。これは、on_page_wnameプロパティ及びon_search_wnameプロパティと一緒に使用されます。

省略値

\$lweb_WAMName (現在のWAMを示します。)

有効値

単一引用符付きのWAM名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

on_page_wrname

前ページもしくは次ページのイメージ/ボタンが押された時に呼び出されるWebroutineです。

省略値

適用されません - Webroutine名の指定は必須です。

有効値

単一引用符付きのWebroutine名。Webroutineは、on_click_wamnameプロパティに指定されたWAMに存在してはいけません。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWebroutineのリストが表示されます。

on_search_wrname

検索ボタンが押された時に呼び出されるWebroutineです。

省略値

適用されません - Webroutine名の指定は必須です。

有効値

単一引用符付きのWebroutine名。Webroutineは、on_click_wamnameプロパティに指定されたWAMに存在してはいけません。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWebroutineのリストが表示されます。

protocol

このウェブレットに呼び出されたWebroutineへの移動に使用するプロトコル(例えば、http:// もしくは https://)

省略値

空白。現在使用されているプロトコルが使用されます。

有効値

単一引用符付きの、有効なプロトコル。通常は'http:'または'https:'です。

page_count_fieldname

前ページもしくは次ページのイメージ/ボタンがクリックされた時に Webroutineに送られる、ページあたりの表示項目数を保持するフィールドの名前。

省略値

ブランク。

有効値

リポジトリもしくはWAMで定義されたフィールド名。プロパティシート内で該当するドロップダウン・ボタンをクリックすると、使用できるフィールド名リストが表示されます。

class

std_list_buttons のみ。

ボタンに適用されるカスケード・スタイル・シート・クラス。

省略値

'STD_BUTTON'- このウェブレット用に提供されているクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

mouseover_class

std_list_buttons のみ。

マウスがその上に置かれた時にボタンに適用されるカスケード・スタイル・シートのクラス名。

省略値

'STD_BUTTON_MOUSOVER' - このウェブレット用に提供されている
mouseoverクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

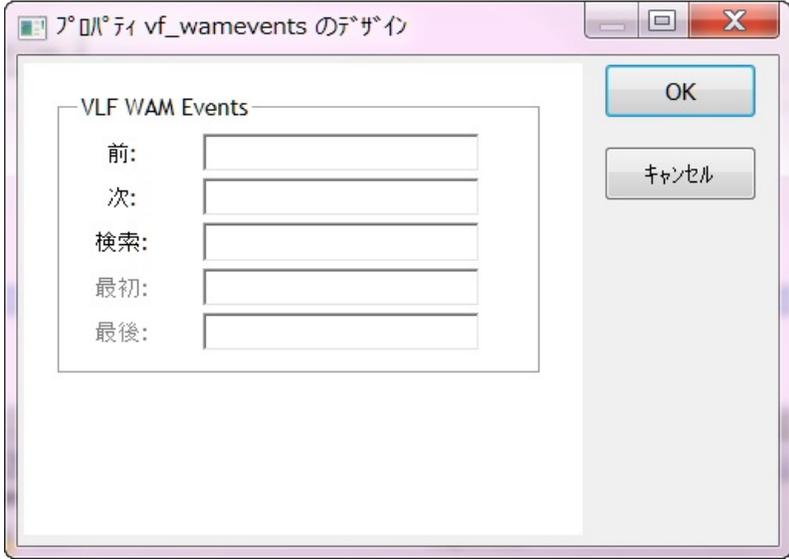
ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

vf_wamevents

VLfの各アクションごとの WAM イベント文字列です。VLf のWAM イベント文字列を入力するには、以下のポップアップ・ダイアログを使用します。



The image shows a Windows-style dialog box titled "プロパティ vf_wamevents のデザイン". The dialog contains a section titled "VLf WAM Events" with five input fields labeled "前:", "次:", "検索:", "最初:", and "最後:". To the right of these fields are two buttons: "OK" and "キャンセル".

show_first_last プロパティが 'False' の場合、最初と最後のVLf のWAM イベントは無効になっています。

省略値

ブランク。.

有効値

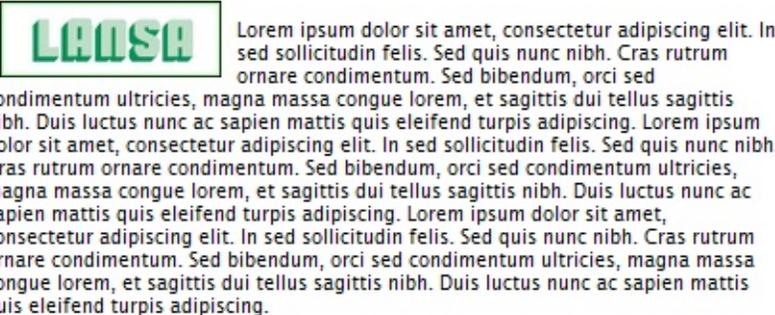
カンマで分けられた文字列。文字列にはカンマ (,) は使用できません。

8.1.14 マークアップ (std_markup)

クイック・スタート - マークアップ プロパティ - マークアップ

マークアップ・ウェブレットは、コンテンツを出力モードのみでビジュアライズしたい時、CKEditorリッチ・テキスト・エディターと共に使用されるウェブレットです。

Sample Mark-up



LAUSA Lorem ipsum dolor sit amet, consectetur adipiscing elit. In sed sollicitudin felis. Sed quis nunc nibh. Cras rutrum ornare condimentum. Sed bibendum, orci sed condimentum ultricies, magna massa congue lorem, et sagittis dui tellus sagittis nibh. Duis luctus nunc ac sapien mattis quis eleifend turpis adipiscing. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In sed sollicitudin felis. Sed quis nunc nibh. Cras rutrum ornare condimentum. Sed bibendum, orci sed condimentum ultricies, magna massa congue lorem, et sagittis dui tellus sagittis nibh. Duis luctus nunc ac sapien mattis quis eleifend turpis adipiscing. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In sed sollicitudin felis. Sed quis nunc nibh. Cras rutrum ornare condimentum. Sed bibendum, orci sed condimentum ultricies, magna massa congue lorem, et sagittis dui tellus sagittis nibh. Duis luctus nunc ac sapien mattis quis eleifend turpis adipiscing.

マークアップのコンテンツにはスクリプト・エレメントと入力フィールドがあるので、マークアップ・ウェブレットを取り扱う際は信頼のおける検証済みのコンテンツのみ使用するよう注意してください。

クイック・スタート - マークアップ

マークアップ・ウェブレットを使用するには、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[マークアップ]ウェブレットを探します。
2. このウェブレットを自身のページにドラッグ・アンド・ドロップしてください。ウェブレットを選択して、[詳細]タブをクリックするのを忘れないでください。nameプロパティをマークアップのコンテンツとしてビジュアライズしたいフィールド名に変更します。

プロパティ - マークアップ

class	name	value
height	pos_absolute	valueFromField
hide_if	title	width

name

ウェブレットを識別する名前です。ウェブレットがフィールドをビジュアライズすると、これがフィールド名になります。

省略値

ウェブレットによりビジュアライズされたフィールドの場合、省略値の名前はフィールド名、もしくはフィールド名と行番号を組み合わせたもの(リストのフィールドの場合)になります。それ以外は、省略値の名前は自動的に生成された、一意の識別子になります。

有効値

単一引用符で囲まれた文字列。

value

ウェブレットに設定する値。ウェブレットがフィールドをビジュアルライズする場合、この値を表示するフィールドが識別されます。

省略値

省略値は適用されません - このウェブレットを使用する多くの場合、マークアップにより表わされる値のフィールドを指定する必要があります。

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

valueFromField

trueが設定されていて、valueプロパティがnullの場合、マークアップ・ウェブレットはマークアップのname属性と一致するフィールドから値をロードします。テキストのコンテンツが大きく、マークアップ・ウェブレットのvalueとWebroutineのフィールド値リストの両方に表示したくない時にこのオプションを使用します。

省略値

false()

有効値

ブール値を返す有効なXPath式。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

class

ウェブレットのカスケード・スタイル・シート(CSS)クラス名です。

省略値

提供されているウェブレットのクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

title

ウェブレットの上をマウスが通った時に、ヒントのテキストとして表示される、ウェブレットのタイトルを指定してください。

省略値

空白 - ヒントは何も表示されません。

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（ウェブレットは相対的に位置付けられます。）

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

Webページのウェブレットの幅です。

通常、ウェブレット幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これでwidth プロパティやheightプロパティの値が更新されます。ただし、必要に応じてプロパティ値を直接編集することもできます。

省略値

ブランク (ウェブレットは省略値の幅を適用します。)

有効値

単一引用符で囲まれた有効な範囲内の幅。

height

Webページのウェブレットの高さ。

通常、ウェブレットの高さや幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これでwidthプロパティやheightプロパティの値が更新されます。ただし、必要に応じてプロパティの値を直接編集することもできます。

省略値

ブランク（ウェブレットは省略値の高さを適用します。）

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

8.1.15 リスト使用のメモ (std_list_textarea)

クイック・スタート - リスト使用のメモ プロパティ - リスト使用のメモ

リスト(テキスト領域)使用のメモのウェブレットは、複数行にまたがる長いテキスト値を表示し、入力するテキスト・エリアを提供します。
<textarea> HTMLエレメントと概ね同じです。このウェブレットは次のようなものです。

```
This is the first list entry. This is the second list entry. This is the third list entry. The list entries are concatenated without the addition of whitespace or line feeds. (The white space in this example was present in the input data.)
```

このウェブレットは、フィールド使用のメモ(std_textarea) ウェブレットによく似ています。違いは、ページとWEBROUTINEの間でテキストが交換される方法です。このウェブレットでは、テキストは作業リストを使用して交換されます。 - 詳細はlistnameプロパティを参照ください。

クイック・スタート - リスト使用のメモ

このウェブレットを使用するには、通常はWEB_MAPに作業リストを指定するWebroutineを作成します。これには1つのテキスト・フィールドが含まれ、これにデータを追加していきます。LANSAがこのようなWebroutine用の省略値XSLを生成する際、リストは通常テーブル内にビジュアライズされます。テーブルの代わりにこのウェブレットを使用するには、以下の手順に従ってください。

1. リストをビジュアライズするテーブルは削除します。

削除するには、リストで右クリックし、ポップアップ・メニューから、[全てのリスト削除]を選択してください。

このリストがWEB_MAP属性*hiddenで指定されている場合は、XSLから手動で削除しなければいけません。(デザイン・ビューに表示されません。)

2. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[リスト使用のメモ]ウェブレットを見つけます。
3. [デザイン]ビューでページ上にウェブレットをドラッグ・アンド・ドロップします。ウェブレットを選択して、[詳細]タブをクリックするのを忘れないでください。
4. 必要に応じて,listname プロパティとlist_text_fieldname プロパティに作業リストの適切なフィールドを参照するように設定します。
5. アプリケーションのフィールドのサイズに従い、colsプロパティを適切に設定します。

プロパティ - リスト使用のメモ

リスト使用のメモ・ウェブレットのプロパティは以下のとおりです。

| | | |
|---------------|---------------------|--------------------|
| class | list_text_fieldname | read_only |
| cols | listname | rows |
| disabled | max_rows_onsubmit | tab_index |
| formname | name | width_design |
| height_design | onchange_script | word_wrap_display |
| hide_if | pos_absolute | word_wrap_onsubmit |

listname

テキスト領域にテキストを追加したり、テキスト領域からのテキストを受け取ったりする為に使用する作業リストの名前。リストはWebroutineのWEB_MAPに指定されていなければいけません。list_text_fieldnameプロパティにより、テキストを含む作業リストのフィールドを識別します。

出力時、作業リストの行は結合され、表示するテキストが作られます。作業リストの各行はテキスト領域の新しい行になります。

入力時、テキスト領域のテキストは複数の文字列に分割され、作業リストに書き込まれます。使用される正確なメカニズムはcolsとword_wrap_onsubmitの値によって変わります。

省略値

省略値は適用されません。リスト名の指定は必須です。

有効値

単一引用符で囲まれた作業リスト名。プロパティ・シートで該当するドロップダウン・ボタンをクリックすると、使用できるリスト名のリストが表示されます。

list_text_fieldname

テキスト領域にテキストを追加したり、テキスト領域からテキストを受け取ったりするために使用する、listnameプロパティによって指定された作業リスト内のフィールド名。このウェブレットと作業リストの使用方法についての詳細は、「listnameプロパティ」を参照してください。

省略値

フィールド名が指定されていない場合、作業リストの最初のフィールドが使用されます。

有効値

単一引用符付きのフィールド名。プロパティシート内で該当するドロップダウン・ボタンをクリックすると、使用できるフィールド名リストが表示されます。

name

ウェブレットを識別する名前です。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用することも可能です。

省略値

自動的に生成された、一意の識別子。

有効値

単一引用符で囲まれた文字列。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

例

次の例はフィールド#STD_FLAGが 'X' と等しい場合にウェブレットを非表示にします。この式は以下のような形式で入力してください。

| | |
|---------|-----------------|
| hide_if | #STD_FLAG = 'X' |
|---------|-----------------|

プロパティがフォーカスを失うと、この式は以下のように表示されます。

| | |
|---------|--------------------------------------|
| hide_if | key('field-value', 'STD_FLAG') = 'X' |
|---------|--------------------------------------|

formname

サーバーに送られるHTMLフォーム名です。

省略値

'LANSA'

有効値

単一引用符付きのフォーム名。プロパティ・シート内で該当するドロップダウン・ボタンをクリックすると、使用できるフォーム名リストが表示されます。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（ウェブレットは相対的に位置付けられます。）

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width_design

Webページのウェブレットの幅です。

通常、ウェブレット幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。こうすることで、width-design プロパティやheight_designプロパティの値を更新します。ただし、必要に応じてプロパティ値を直接編集することもできます。

省略値

ブランク(ウェブレットは省略値の幅- colsプロパティにより決定 - を使用します。)

有効値

単一引用符で囲まれた有効な範囲内の幅。

height_design

Webページのウェブレットの高さ。

通常、ウェブレットの高さや幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。こうすることで、width-design プロパティやheight_design プロパティの値を更新します。ただし、必要に応じてプロパティの値を直接編集することもできます。

省略値

空白(ウェブレットは省略値の高さ - rows プロパティにより決定 - を使用します。)

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

rows

テキスト領域に表示する行の数。このプロパティは、指定された行数やテキスト行が表示できるように、ウェブレットの高さを設定します。

height_designプロパティが指定されている場合、それが優先され、rowsプロパティは無視されます。

省略値

10

有効値

行数を指定する数。

cols

テキストの領域に表示する列の数。このプロパティは、おおよそテキストの指定された列数が表示されるように、使用されるフォントの平均文字幅に基づき、ウェブレットの幅を設定します。実際の幅を決定するのに使用されるメカニズムはブラウザごとに異なり、異なるブラウザではわずかに異なるサイズになります。

width_designプロパティが指定されている場合、それが優先され、colsプロパティは無視されます。

省略値

50

有効値

列数を指定する数字。

word_wrap_display

このプロパティはウェブレットの表示テキストのワードラップの処理方法を指定します。trueの場合、長いテキスト行はテキスト領域の右端で自動的に折り返されます。これは表示のみであることに注意してください。作業リストに入力されたテキストのワードラップは、word_wrap_onsubmitプロパティによって決定されます。

省略値

true()

有効値

true() か false() のブール値、もしくはtrueかfalseで解が得られる式。

word_wrap_onsubmit

このプロパティはテキストを送る時にウェブレットがワードラップをどのように扱うかを指定します。trueの場合、長いテキスト行は自動的にcolsの文字幅で折り返されます。固定幅のフォントがテキスト・エリアに使用されている場合、この折り返しはユーザーに画面上に表示されるものと一致します。

falseの場合、長いテキスト行は折り返されません。

データの消失を避けるため、このプロパティの値に関係なく、list_text_fieldnameフィールドの最大幅でワードラップが実行されます。

省略値

false()。

有効値

true() か false() のブール値、もしくはtrueかfalseで解が得られる式。

max_rows_onsubmit

送信する最大行数。作業リストに最大サイズがある場合は、このプロパティにその値を設定しなくてはなりません。

省略値

0 (全ての行を送ります。)

有効値

行数を指定する数字。

class

ウェブレットのカスケード・スタイル・シート(CSS)クラス名。

省略値

提供されているウェブレットのクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

read_only

ウェブレットのコンテンツを読取専用(ユーザーはコンテンツの変更ができません)にするかどうかを決定するブール値。

省略値

空白 - Falseと同じです。(ユーザーはコンテンツを変更できません。)

有効値

true()、false() もしくは有効な式。

例

次の例はフィールド#STD_FLAGが 'Y' と等しい場合にウェブレットを読取専用に設定します。この式は以下のような形式で入力してください。

```
read_only #STD_FLAG = 'Y'
```

プロパティがフォーカスを失うと、この式は以下のように表示されません。

```
read_only key('field-value', 'STD_FLAG') = 'Y'
```

disabled

ウェブレットを使用可または不可として表示するかを決定するブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false() もしくは有効な式。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

onchange_script

テキストが変更された後に、テキスト領域のフォーカスが失われた時に実行されるJavaScriptコード。JavaScriptステートメントは必ずセミコロンので終了していなければなりません。

省略値

空白。JavaScriptは何も実行されません。

有効値

有効なJavaScriptステートメント。

8.1.16 大型リスト (std_largelist)

クイック・スタート - 大型リスト プロパティ - 大型リスト

大型リスト・ウェブレットを使って、大きなリストを表示します。このウェブレットは、特別なフォーマット変換を必要としないレポート形式の出力専用リストに適しています。リストはXHTMLリストもしくはCSV(カンマ区切りのデータ)ファイルとして、ユーザー・エージェントに送られます。

Large List

| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 | Column 6 | Column 7 | Column 8 | Column 9 | Column 10 |
|--------------|----------|-----------------------|-------------|-------------|------------|-----------|----------|------------------|----------------|
| AbCdEf | 123,456 | AbCdEfGhIjKlMnOpQrStU | 987,654,321 | VwXyZ | 2006-06-24 | VwXyZ | 1654789 | CdEfGhIjKlMnOp | RsTuVwXyZ |
| EfGhIjKlMnOp | 3,456 | AbCdEfGhIjK | 87,654,321 | AbCdEdGhIjK | 2007-12-31 | LmNoPqRsT | 4795439 | EfGhIjKlMnOpQrSt | IjKlMnOpQrStUv |
| AbCdEf | 123,456 | AbCdEfGhIjKlMnOpQrStU | 987,654,321 | VwXyZ | 2006-06-24 | VwXyZ | 1654789 | CdEfGhIjKlMnOp | RsTuVwXyZ |
| EfGhIjKlMnOp | 3,456 | AbCdEfGhIjK | 87,654,321 | AbCdEdGhIjK | 2007-12-31 | LmNoPqRsT | 4795439 | EfGhIjKlMnOpQrSt | IjKlMnOpQrStUv |
| AbCdEf | 123,456 | AbCdEfGhIjKlMnOpQrStU | 987,654,321 | VwXyZ | 2006-06-24 | VwXyZ | 1654789 | CdEfGhIjKlMnOp | RsTuVwXyZ |
| EfGhIjKlMnOp | 3,456 | AbCdEfGhIjK | 87,654,321 | AbCdEdGhIjK | 2007-12-31 | LmNoPqRsT | 4795439 | EfGhIjKlMnOpQrSt | IjKlMnOpQrStUv |
| AbCdEf | 123,456 | AbCdEfGhIjKlMnOpQrStU | 987,654,321 | VwXyZ | 2006-06-24 | VwXyZ | 1654789 | CdEfGhIjKlMnOp | RsTuVwXyZ |
| EfGhIjKlMnOp | 3,456 | AbCdEfGhIjK | 87,654,321 | AbCdEdGhIjK | 2007-12-31 | LmNoPqRsT | 4795439 | EfGhIjKlMnOpQrSt | IjKlMnOpQrStUv |
| AbCdEf | 123,456 | AbCdEfGhIjKlMnOpQrStU | 987,654,321 | VwXyZ | 2006-06-24 | VwXyZ | 1654789 | CdEfGhIjKlMnOp | RsTuVwXyZ |
| EfGhIjKlMnOp | 3,456 | AbCdEfGhIjK | 87,654,321 | AbCdEdGhIjK | 2007-12-31 | LmNoPqRsT | 4795439 | EfGhIjKlMnOpQrSt | IjKlMnOpQrStUv |
| AbCdEf | 123,456 | AbCdEfGhIjKlMnOpQrStU | 987,654,321 | VwXyZ | 2006-06-24 | VwXyZ | 1654789 | CdEfGhIjKlMnOp | RsTuVwXyZ |
| EfGhIjKlMnOp | 3,456 | AbCdEfGhIjK | 87,654,321 | AbCdEdGhIjK | 2007-12-31 | LmNoPqRsT | 4795439 | EfGhIjKlMnOpQrSt | IjKlMnOpQrStUv |
| AbCdEf | 123,456 | AbCdEfGhIjKlMnOpQrStU | 987,654,321 | VwXyZ | 2006-06-24 | VwXyZ | 1654789 | CdEfGhIjKlMnOp | RsTuVwXyZ |
| EfGhIjKlMnOp | 3,456 | AbCdEfGhIjK | 87,654,321 | AbCdEdGhIjK | 2007-12-31 | LmNoPqRsT | 4795439 | EfGhIjKlMnOpQrSt | IjKlMnOpQrStUv |
| AbCdEf | 123,456 | AbCdEfGhIjKlMnOpQrStU | 987,654,321 | VwXyZ | 2006-06-24 | VwXyZ | 1654789 | CdEfGhIjKlMnOp | RsTuVwXyZ |
| EfGhIjKlMnOp | 3,456 | AbCdEfGhIjK | 87,654,321 | AbCdEdGhIjK | 2007-12-31 | LmNoPqRsT | 4795439 | EfGhIjKlMnOpQrSt | IjKlMnOpQrStUv |
| AbCdEf | 123,456 | AbCdEfGhIjKlMnOpQrStU | 987,654,321 | VwXyZ | 2006-06-24 | VwXyZ | 1654789 | CdEfGhIjKlMnOp | RsTuVwXyZ |
| EfGhIjKlMnOp | 3,456 | AbCdEfGhIjK | 87,654,321 | AbCdEdGhIjK | 2007-12-31 | LmNoPqRsT | 4795439 | EfGhIjKlMnOpQrSt | IjKlMnOpQrStUv |

クイック・スタート - 大型リスト

1. 通常通りに(メインの)Webroutineを作成します。ただし、作業リストにWEB_MAPを定義しないでください。その代わりに、作業リストを置く場所に大型リスト・ウェブレットを入れます。
2. 大型リストを処理する別のWebroutineを作成してください。作業リストにWEB_MAP FOR(*OUTPUT) を定義します。リストのコンテンツで手を加える必要があるフィールドには、WEB_MAP FOR(*INPUT)を定義します。
3. メインのWebroutineで、大型リスト・ウェブレットのプロパティを編集します。
 - a. リストを処理するWebroutineとして、ステップ2で作成したWebroutineを指定します。
 - b. field_names_to_exchangeで、リストを処理するWebroutineで使用するフィールドを選択してください。
4. メインのWebroutineを実行して、リストを確認します。

プロパティ - 大型リスト

大型リスト・ウェブレットのプロパティは以下のとおりです。

| | | |
|--|----------------------------|---------------------------|
| <code>column_css_class</code> | <code>format_target</code> | <code>show_busybox</code> |
| <code>csv_hyperlink_relative_image_path</code> | <code>iframe_height</code> | <code>src_wamname</code> |
| <code>csv_hyperlink_text</code> | <code>iframe_width</code> | <code>src_wrname</code> |
| <code>csv_hyperlink_type</code> | <code>listname</code> | <code>wait_content</code> |
| <code>fields_names_to_exchange</code> | <code>name</code> | |

name

リストの名前。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やグリッドを参照するXSLを使用する場合は、独自の名前を使用することもできます。

省略値

自動的に生成された、一意の識別子。

有効値

単一引用符で囲まれた文字列。XHTMLページ全体の中で一意のものでなければいけません。

listname

大型リストにデータを追加するために使用する作業リスト名。

省略値

ブランク。

有効値

単一引用符で囲まれた文字列。リストを処理するWebroutineの出力としてこのリストがマップされている必要があります。(src_wnameプロパティを参照してください。)

format_target

ソースのWAMWebroutineによって処理されるデータのフォーマット/ターゲット。

省略値

xhtml-iframe

有効値

1. xhtml-iframe: リストはiframe内のXHTMLテーブルとしてフォーマットされています。iframe内でCtrl+A (全てを選択)を使用してユーザーがテーブル全体を選択できるようにしたい場合は、このオプションを使用します。
2. xhtml-inline: リストはXHTMLテーブルとしてフォーマットされており、メインのXHTMLページの行の中に置かれています。これはメインのXHTMLページ内でのXHTML定義に一番近い形です。
3. csv-inline: リストはiframeのなかでカンマ区切りのドキュメントとしてフォーマットされています。ブラウザがMicrosoft Internet ExplorerでMicrosoft Excelがある場合、iframeはリストをスプレッドシートとして表示します。
4. csv-window: リストはカンマ区切りのドキュメントとしてフォーマットされ、新しいウィンドウで開かれます。ブラウザがMicrosoft Internet ExplorerでMicrosoft Excelがある場合、新しいウィンドウはリストをスプレッドシートとして表示します。

警告

- xhtml-inline フォーマット/ターゲットはリストをプログラムでロードします。ページは保存することができません。ページを保存するには、xhtml-iframeオプションを使用してください。(iframeコンテンツを保存できません。)

iframe_width

iframeの幅。(コンテンツのターゲットがiframeの場合のみ、適用可能)

省略値

'auto'

有効値

単一引用符で囲まれたリテラル値で幅を入力してください。(単位は、パーセント、ポイント、ピクセルが使用できます。)フォーマット/ターゲットが'xhtml-iframe'の場合、特別な値'auto'を使用して、コンテンツが水平スクロールバーなしでも表示できるようにiframeのサイズが調整されます。

iframe_height

iframeの高さ。(コンテンツのターゲットがiframeの場合のみ、適用可能)

省略値

'auto'

有効値

単一引用符で囲まれたリテラル値で高さを入力してください。(単位は、パーセント、ポイント、ピクセルが使用できます。)フォーマット/ターゲットが'xhtml-iframe'の場合、特別な値'auto'を使用して、コンテンツが垂直スクロールバーがなくても表示できるようにiframeのサイズが調整されます。

column_css_class

ブール値。フォーマット/ターゲットがXHTMLの場合のみ適用可能です。trueの場合、カスケード・スタイル・シート(CSS)のクラスはXHTMLテーブルの各列に割り当てられています。falseの場合、カスケード・スタイル・シートのクラスは行レベルのみに当てられます。

省略値

false()。CSSスタイルを行レベルで適用します。

有効値

true()、false()、もしくは、ブール値を返すXSLT式。

大型リストのCSSスタイル：事前にフォーマットされたXHTMLテーブルには、"std_largelist"クラスがあります。このクラスもしくはそのサブ・エレメントに対してCSSスタイルを作成することで、大型リストの外観をカスタマイズできます。

ターゲットがiframeの時、iframeは親のウィンドウ(メインのWebroutine)からスタイルを受け継ぎます。

例えば、Webroutineでスタイルを次のように上書きすると、

```
table.std_largelist th, table.std_largelist td {white-space:nowrap;}  
tr.list-h {background:black; color:white; font-weight:bold;}  
table.std_largelist tr.list-o {background:white; color:black;}  
table.std_largelist tr.list-e {background:#ffe9bd; color:black;}
```

リストは次のようになります。

Large List

| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 | Column 6 | Column 7 | Column 8 | Column 9 | Column 10 |
|--------------|----------|-----------------------|-------------|-------------|------------|-----------|----------|------------------|----------------|
| AbCdeF | 123,456 | AbCdeFGhIjKlMnOpQrStU | 987,654,321 | VwXyZ | 2006-06-24 | VwXyZ | 1654789 | CdeFGhIjKlMnOp | RsTuVwXyZ |
| EfGhIjKlMnOp | 3,456 | AbCdeFGhIjK | 87,654,321 | AbCdEdGhIjK | 2007-12-31 | LmNoPqRsT | 4795439 | EfGhIjKlMnOpQrSt | IjKlMnOpQrStUv |
| AbCdeF | 123,456 | AbCdeFGhIjKlMnOpQrStU | 987,654,321 | VwXyZ | 2006-06-24 | VwXyZ | 1654789 | CdeFGhIjKlMnOp | RsTuVwXyZ |
| EfGhIjKlMnOp | 3,456 | AbCdeFGhIjK | 87,654,321 | AbCdEdGhIjK | 2007-12-31 | LmNoPqRsT | 4795439 | EfGhIjKlMnOpQrSt | IjKlMnOpQrStUv |
| AbCdeF | 123,456 | AbCdeFGhIjKlMnOpQrStU | 987,654,321 | VwXyZ | 2006-06-24 | VwXyZ | 1654789 | CdeFGhIjKlMnOp | RsTuVwXyZ |
| EfGhIjKlMnOp | 3,456 | AbCdeFGhIjK | 87,654,321 | AbCdEdGhIjK | 2007-12-31 | LmNoPqRsT | 4795439 | EfGhIjKlMnOpQrSt | IjKlMnOpQrStUv |
| AbCdeF | 123,456 | AbCdeFGhIjKlMnOpQrStU | 987,654,321 | VwXyZ | 2006-06-24 | VwXyZ | 1654789 | CdeFGhIjKlMnOp | RsTuVwXyZ |
| EfGhIjKlMnOp | 3,456 | AbCdeFGhIjK | 87,654,321 | AbCdEdGhIjK | 2007-12-31 | LmNoPqRsT | 4795439 | EfGhIjKlMnOpQrSt | IjKlMnOpQrStUv |
| AbCdeF | 123,456 | AbCdeFGhIjKlMnOpQrStU | 987,654,321 | VwXyZ | 2006-06-24 | VwXyZ | 1654789 | CdeFGhIjKlMnOp | RsTuVwXyZ |
| EfGhIjKlMnOp | 3,456 | AbCdeFGhIjK | 87,654,321 | AbCdEdGhIjK | 2007-12-31 | LmNoPqRsT | 4795439 | EfGhIjKlMnOpQrSt | IjKlMnOpQrStUv |
| AbCdeF | 123,456 | AbCdeFGhIjKlMnOpQrStU | 987,654,321 | VwXyZ | 2006-06-24 | VwXyZ | 1654789 | CdeFGhIjKlMnOp | RsTuVwXyZ |
| EfGhIjKlMnOp | 3,456 | AbCdeFGhIjK | 87,654,321 | AbCdEdGhIjK | 2007-12-31 | LmNoPqRsT | 4795439 | EfGhIjKlMnOpQrSt | IjKlMnOpQrStUv |
| AbCdeF | 123,456 | AbCdeFGhIjKlMnOpQrStU | 987,654,321 | VwXyZ | 2006-06-24 | VwXyZ | 1654789 | CdeFGhIjKlMnOp | RsTuVwXyZ |
| EfGhIjKlMnOp | 3,456 | AbCdeFGhIjK | 87,654,321 | AbCdEdGhIjK | 2007-12-31 | LmNoPqRsT | 4795439 | EfGhIjKlMnOpQrSt | IjKlMnOpQrStUv |
| AbCdeF | 123,456 | AbCdeFGhIjKlMnOpQrStU | 987,654,321 | VwXyZ | 2006-06-24 | VwXyZ | 1654789 | CdeFGhIjKlMnOp | RsTuVwXyZ |
| EfGhIjKlMnOp | 3,456 | AbCdeFGhIjK | 87,654,321 | AbCdEdGhIjK | 2007-12-31 | LmNoPqRsT | 4795439 | EfGhIjKlMnOpQrSt | IjKlMnOpQrStUv |
| AbCdeF | 123,456 | AbCdeFGhIjKlMnOpQrStU | 987,654,321 | VwXyZ | 2006-06-24 | VwXyZ | 1654789 | CdeFGhIjKlMnOp | RsTuVwXyZ |

src_wamname

リストを処理するためにWebroutineが実行されるWAMの名前を指定します。(Webroutine名はsrc_wnameプロパティに指定されます。)

省略値

指定しない場合は、現在のWAMが使用されます。(\$lweb_WAMName)

有効値

単一引用符付きのWAM名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

src_wrname

リスト処理のために実行されるWebroutineの名前を指定します。
(WebroutineのあるWAMの名前はsrc_wamnameプロパティに指定されま
す。)

省略値

省略値はありません。有効なWebroutine名を入力しなくてははいけませ
ん。

有効値

単一引用符付きのWebroutine名。Webroutineは、src_wamnameプロパ
ティに指定されたWAMに存在していなくてははいけません。プロパ
ティシートで該当するドロップダウン・ボタンをクリックすると、使
用できるWebroutineのリストが表示されます。

fields_names_to_exchange

任意。リストを処理するWebroutineに送る1つもしくは複数のフィールドを選択します。通常はその値によりリストのコンテンツが決定されるフィールドを指定します。複数のフィールドを選択するには、Ctrlキーを押しながら、追加するフィールド名を選択します。

フィールドはリストを処理するWebroutineに送信されますが、メインのWebroutineに戻されることはありません。つまり、リストを処理するWebroutineに入力としてマップするのと同じ動作になります。

省略値

省略値はありません。

有効値

現在のWebroutineで出力用にマップされたフィールド名をカンマで区切ったリスト。

csv_hyperlink_type

新しいCSVウィンドウへのハイパーリンクをイメージとして表示するかテキスト・リンクとして表示するかを指定します。'csv-window'フォーマット/ターゲットのみ適用可能です。

省略値

image

有効値

image:ハイパーリンクはイメージとして表示されます。イメージをcsv_hyperlink_relative_image_pathプロパティに定義します。

text:ハイパーリンクはテキスト・リンクとして表示されます。テキストをcsv_hyperlink_textプロパティに定義します。

csv_hyperlink_relative_image_path

新しいCSVウィンドウへのハイパーリンクを表すために表示される、イメージディレクトリに関連付けられたイメージのパスと名前。'csv-window'フォーマット/ターゲットのみ適用可能です。

省略値

'excel.gif'。Microsoft Excelスプレッドシートを表すイメージです。

有効値

イメージ・ディレクトリに関連付けられたイメージのパスおよび名前
で、単一引用符で囲まれたもの。イメージは、プロパティ・シートで
該当する省略記号のボタンをクリックし、プロンプターから選択する
ことができます。

csv_hyperlink_text

新しいCSVウィンドウへのハイパーリンクを表わすテキスト。これは csv_hyperlink_type プロパティで 'text' を選んだときのハイパーリンクのテキストです。'csv-window' フォーマット/ターゲットのみ適用可能です。

省略値

'Hyperlink'

有効値

適切なリテラル、フィールド名、システム変数名、もしくは複数言語対応テキスト変数名。フィールド、システム変数、複数言語変数の名前は、プロパティシートで該当する省略記号(...)のボタをクリックして、リストから選択することができます。

show_busybox

リストが処理されている間にビジー・ボックスのメッセージを表示するかどうかを選択します。これは大型リストの場合は推奨されています。リストの準備、送信、表示にかなりの時間がかかる可能性があるからです。

ビジー・ボックス・メッセージは新しいウィンドウでは表示できません。インライン・コンテンツもしくはiframeの中にロードされたコンテンツとしてのみ表示できます。

省略値

true()。ビジー・ボックス・メッセージは表示されます。

有効値

true()、false()、もしくは、ブール値を返すXSLT式。

wait_content

任意。大型リストが処理されている間にビジー・ボックスに表示するメッセージ。show_busyboxプロパティがtrue()の場合のみ適用されます。

省略値

ブランク - 省略値である'Processing'というテキストが表示されます。

有効値

適切なリテラル、フィールド名、システム変数名、もしくは複数言語対応テキスト変数名。フィールド、システム変数、複数言語変数の名前は、プロパティシートで該当する省略記号(...)のボタをクリックして、リストから選択することができます。短いメッセージにするようにしてください。

例

この例ではリテラルのメッセージを表示しています。

wait_content 'Please wait ...'

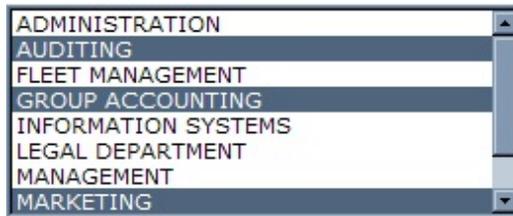
ビジー・ボックスは以下のように表示されます。



8.1.17 リスト・ボックス (std_listbox)

クイック・スタート - リスト・ボックス プロパティ - リスト・ボックス

リスト・ボックス・ウェブレットは、Windowsに似た単一または複数選択のできるリストボックスをWebページに提供します。例えば次のようなものです。



クイック・スタート - リスト・ボックス

「リスト・ボックスの例」で紹介されている例を参照してください。

プロパティ - リスト・ボックス

リスト・ボックス・ウェブレットのプロパティは以下のとおりです。

| | | |
|------------------------|------------------------|---------------------------------|
| allow_multi_selections | listname | reentryvalue |
| captionfield | mouseover_class | selector_field |
| class | multi_select_codefield | selector_value_eq |
| codefield | multi_select_listname | size |
| disabled | name | submit_tagfields |
| display_mode | on_select_wamname | tab_index |
| formname | on_select_wrname | tagfield1, tagfield2, tagfield3 |
| height_design | pos_absolute | target_window_name |
| hide_if | protocol | value |
| items | reentryfield | vf_wamevent
width_design |

name

リスト・ボックスの名前。リスト・ボックスがエントリーの単一選択のみを許している場合、単一引用符で囲まれたコード・フィールド名 ('DEPARTMENT'など)を指定し、ここにはWAMに返される選択されたエントリーの値が含まれます。複数選択の場合は、名前は省略値のまま残します。

省略値

`concat('o', position(), '_LANSA_n')` - LANSAにより付けられたウェブレット内部名です。

有効値

単一引用符で囲まれた名前。省略値、もしくはウェブレットが値を設定するフィールドの名前になります。

value

ウェブレットに設定する値。単一選択のリストでは、事前に選択されるコード値(\$DEPARTMENTなど)を含むフィールド名にする必要があります。複数選択リストでは、このプロパティは省略値のままで構いません。

省略値

空白。

有効値

単一引用符で囲まれたテキスト、もしくは複数言語対応テキスト変数、システム変数、フィールドの名前(プロパティ・シートで省略記号のボタンをクリックしてリストから選ぶことができます。)

display_mode

ウェブレットが入力内容を受け取るのか、出力内容を表示するのかを制御します。

省略値

'input'

有効値

'input' または 'output'

items

ウェブレットに表示される項目を指定するXMLノード・セットです。設定はデザイナーでのみ可能です。デザイナーを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使用してください。項目がlistnameプロパティに指定されたリストから追加される場合は、ブランクにしてください。

省略値

document(")/*//xml:data/xml:listbox (このリスト・ボックスに項目が何も定義されていないことを示します。)

有効値

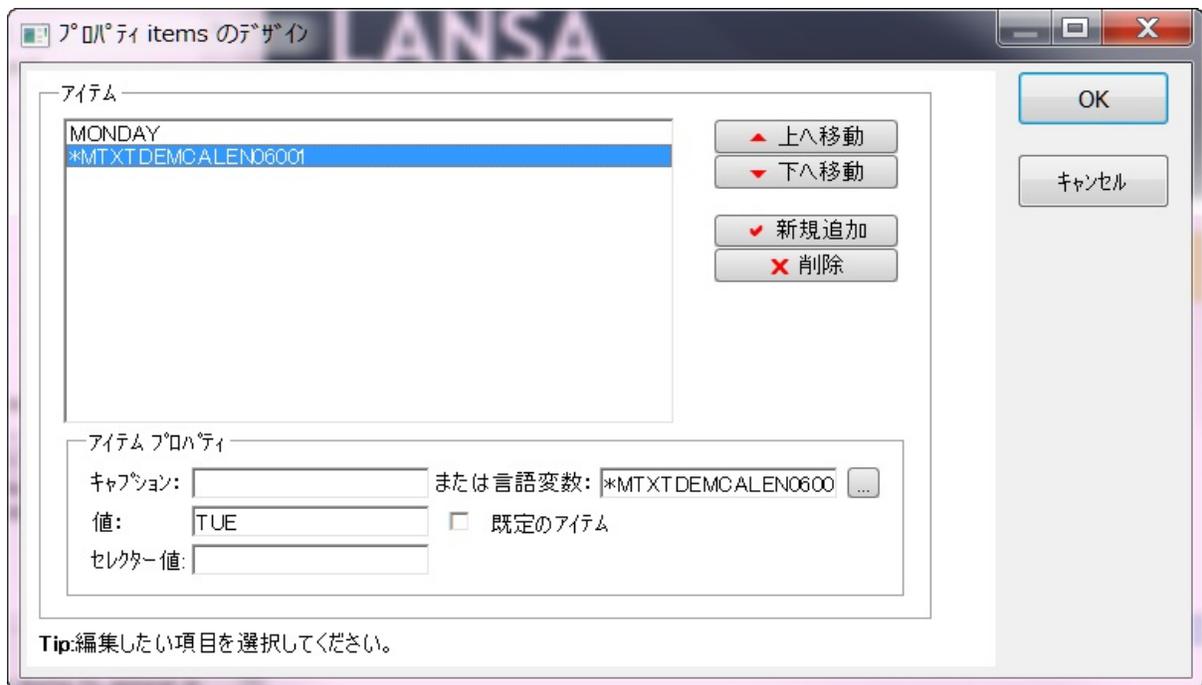
適用できません。(システムにより保守されます。)デザイナーを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使用してください。

例

次の例は項目がデザイナーで設定され、リスト・ボックスの値として使用されることを示しています。

 items document(")/*//xml:data/xml:caption_value_pairs[@id='4A14173FA6D64A059507F9124A630CF3']

プロパティの省略記号(...)のボタンを使用すると、デザイナーが表示され、リスト・ボックスに表示されるアイテムを保守できます。以下のデザイナーのビューは、リスト・ボックスに2つのエントリーがあることを示しています。最初のエントリーはリテラル値'MONDAY'、2つめのエントリーは複数言語変数を使用して、コードTUEの記述を表示します。値が事前に選択されていない時に選択された状態にする項目には[規定のアイテム]のチェックボックスを使用します。



size

リストに表示する行の数。height_designプロパティが指定された場合は、このプロパティは無視されます。

省略値

8

有効値

有効な数字。

allow_multi_selections

リスト・ボックスで複数選択を許可するかどうかを制御するブール値プロパティ。複数選択が許されている場合、multi_select_listnameプロパティ及びmulti_select_codefieldプロパティを必ず指定してください。

省略値

false() - リスト・ボックスで選択できるのは1つのみになります。

有効値

true()、false()、もしくは、TrueもしくはFalseを返す有効な式。

multi_select_listname

allow_multi_selectionsプロパティがTrueの場合、リスト・ボックスの選択された複数のエントリーを含む作業リスト。作業リストはmulti_select_codefieldプロパティに指定されたコード・フィールドを含んでいる必要があります。

省略値

ブランク - リスト・ボックスで選択できるのは1つのみになります。

有効値

単一引用符で囲まれた作業リストの名前。プロパティ・シートで該当するドロップダウン・ボタンをクリックして、使用できる作業リストのリストから選択できます。

multi_select_codefield

選択されたリスト・ボックスの複数の項目のコード値を保持する、`multi_select_listname`プロパティに指定された作業リストのフィールド名。

省略値

ブランク - リスト・ボックスで選択できるのは1つのみになります。

有効値

単一引用符付きのフィールド名。プロパティ・シートの該当するドロップダウン・ボタンをクリックして、使用できるフィールドのリストから選択できます。

listname

ウェブレットにデータを追加するために使用される項目を含む作業リスト名。

省略値

空白。有効なリスト名を入力する必要があります。

有効値

シングル・クォーテーション付きの作業リストの名前。有効なリスト名の一覧は、プロパティシートの対応するドロップダウン・ボタンをクリックすることで、選択することができます。

selector_field

listname プロパティに指定されているリスト内のフィールドで、ウェブレットに表示されたリスト項目を制限したり、部分集合にしたりする値を含むフィールドの名前。

省略値

ブランク。リストの全てのエントリーが常に表示されます。

有効値

単一引用符で囲まれた有効なフィールド名。有効なフィールド名の一覧は、プロパティシートの該当するドロップダウン・ボタンをクリックして選択できます。

selector_value_eq

ウェブレットに表示されたリスト項目を制限したり、部分集合にしたりするために、`selector_field`プロパティに指定されているフィールド値と一致しなければいけない値。`items`プロパティを使用してリストの詳細が入力されている場合は、この値はセレクター値に一致していなくてもはいけません。

省略値

空白。リストの全てのエントリーが常に表示されます。

有効値

`selector_field`プロパティに指定されたフィールド・タイプに有効な値、もしくは`items`プロパティに入力された値。

codefield

各リスト項目のキー値を保持するlistnameプロパティに指定されたリストのフィールド名。

省略値

空白。

有効値

単一引用符で囲まれた文字列。プロパティ・シートで該当するドロップダウン・ボタンをクリックすると、listnameに指定された作業リストからのフィールドを選択できます。

captionfield

各リスト項目のキャプションを保持するlistnameプロパティに指定されたリストのフィールド名。

省略値

空白。

有効値

単一引用符で囲まれた文字列。プロパティ・シートで該当するドロップダウン・ボタンをクリックすると、listnameに指定された作業リストからのフィールドを選択できます。

tagfield1, tagfield2, tagfield3

リスト項目にタグ付けされる追加の値を含むlistnameプロパティに指定されたリストのフィールド名。この値は、接頭辞が'tag_'のフィールド名の属性として追加されます。値はリスト・ボックスに表示されませんが、JavaScriptコードで参照できます。

省略値

空白。

有効値

単一引用符で囲まれた文字列。プロパティ・シートで該当するドロップダウン・ボタンをクリックすると、listnameに指定された作業リストからのフィールドを選択できます。

submit_tagfields

フォームを送信した時に、tagfieldn プロパティに指定されたフィールドの値をWebサーバーに送るかどうかが制御するブール値プロパティ。

省略値

True() - フィールドの値はWebサーバーに送られます。

有効値

true()、false()、もしくは、true() か false()を返す有効な式。

reentryfield

reentryvalue プロパティに指定された値をWAMに送る時に使用するフィールド名です。フィールド名は単一引用符で囲みます。

省略値

'STDREENTRY'

有効値

リポジトリもしくはWAMで定義されたフィールド名。プロパティシート内で該当するドロップダウン・ボタンをクリックすると、使用できるフィールド名リストが表示されます。

reentryvalue

reentryfieldプロパティに指定されたフィールドに送られる値。英数字フィールドの場合、値は単一引用符付きで指定する必要があります。数字フィールドの場合は、単一引用符はなくても構いません。

省略値

'M'

有効値

適切なリテラル。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

true()、false()、もしくは、true() か false()を返す有効な式。

例

次の例はフィールドSTD_FLAGが 'X'と等しい場合にウェブレットを非表示にします。式は入力されなくてはなりません。またプロパティにフォーカスがある時、以下のように表示されます。

| | |
|---------|-----------------|
| hide_if | #STD_FLAG = 'X' |
|---------|-----------------|

プロパティがフォーカスを失うと、この式は以下のように表示されます。

| | |
|---------|--------------------------------------|
| hide_if | key('field-value', 'STD_FLAG') = 'X' |
|---------|--------------------------------------|

formname

サーバーに送られるHTMLフォーム名です。

省略値

'LANSA'

有効値

単一引用符付きのフォーム名。プロパティ・シート内で該当するドロップダウン・ボタンをクリックすると、使用できるフォーム名リストが表示されます。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

height_design

Webページのウェブレットの高さ。

省略値

ブランク(ウェブレットは省略値の高さを使用します。)

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

width_design

Webページのウェブレットの幅です。

省略値

ブランク（ウェブレットは省略値の幅を使用します。）

有効値

単一引用符で囲まれた有効な範囲内の幅。

on_select_wamname

リスト・ボックス内の項目が選択された時に呼び出されるWAMの名前。

省略値

空白 - 現在のWAMです。

有効値

単一引用符で囲まれたWAM名。プロパティ・シートの該当するドロップダウン・ボタンをクリックして、使用できるWAMのリストから選択できます。

on_select_wrname

リスト・ボックス内の項目が選択された時に呼び出されるWebroutineの名前。

省略値

空白 - 項目が選択された時Webroutineは呼び出されません。

有効値

単一引用符で囲まれたWebroutine名。Webroutineは、on_select_wamnameプロパティに指定されたWAMに存在していなくてはなりません。プロパティ・シートの該当するドロップダウン・ボタンをクリックして、使用できるWebroutineのリストから選択できます。

protocol

このウェブレットに呼び出されたWebroutineへの移動に使用するプロトコル(例えば、http:// もしくは https://)

省略値

空白。現在使用されているプロトコルが使用されます。

有効値

単一引用符付きの、有効なプロトコル。通常は'http:'または'https:'です。

target_window_name

応答HTMLが表示される、ウィンドウやフレームの名前。

省略値

ブランク - 応答HTMLが現在のウィンドウに表示されます。

有効値

単一引用符で囲まれたウィンドウ名またはフレーム名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるウィンドウやフレームが表示されます。

'_blank'は新しいウィンドウに表示します。

'_media'は現在のウィンドウのメディア・パネルに表示します。

'_search'は現在のウィンドウの検索パネルに表示します。

'_parent'は親ウィンドウ（多くの場合、現ウィンドウ）に表示します。

'_top'は一番上のウィンドウ（多くの場合、現ウィンドウ）に表示します。

_search や _media はInternet Explorer 6 でのみサポートされています。

disabled

ウェブレットを使用可または不可として表示するかを決定するブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false()、もしくは、true() か false()を返す有効な式。

class

ウェブレットのカスケード・スタイル・シートのクラス名。

省略値

'std_listbox' - このウェブレット用に提供されているクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

mouseover_class

マウスがその上に置かれた時の、ウェブレットのカスケード・スタイル・シートのクラス名。

省略値

空白。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

vf_wamevent

VLF のWAM イベント文字列です。

省略値

空白。

有効値

文字列。カンマ (,) は使用できません。

リスト・ボックスの例

以下では複数選択が可能なリスト・ボックスの例で、その結果を生成する際に使用されるテクニックが説明されています。

リスト・ボックスで単一選択のみの場合は、[allow_multi_selections](#)、およびそれに関連するプロパティの説明を参照してください。



このリスト・ボックスは複数選択が可能です。(ShiftもしくはCtrlキーを押して複数選択するという標準的なWindowsの方法を使用)[Produce Report]ボタンがクリックされると、選択したエントリーがWAMに送られ、処理されます。

リスト・ボックスで選択された項目を変更して、[Produce Report]ボタンをクリックします。すると、次のようなリスト内で選択された項目のリストが表示されます。



リスト定義

次の2つのリストを複数選択のリスト・ボックスとしてプログラムする必要があります。

```
Def_List Name(#ListBox) Fields(#DEPARTMENT #DEPTDESC) Type(*Working)  
Def_List Name(#Selected) Fields(#DEPARTMENT) Type(*Working)
```

1つめは、#ListBoxで、リスト・ボックスのコンテンツを操作します。これには通常2つのフィールド、つまりコード・フィールドと記述フィールドが含まれています。

2つめのリストは、#Selectedで、2通りの使用方法があります。WAM内のこのリストにはエントリーが追加され、リスト・ボックスが表示される時に#ListBoxのエントリーが事前に選択されているようにします。またこのリストは、ユーザーが選択したエントリーをWAMに返す際にも使用されます。

LANSAエディターのShowPageWebroutineで右クリックすると、リスト・ボックスとプッシュ・ボタンが表示されます。リスト・ボックスをクリックし、[詳細]タブをクリックしてそのプロパティを表示してください。

| | | |
|---|------------------------|------------|
| r | allow_multi_selections | True |
| r | multi_select_listname | SELECTED |
| r | multi_select_codefield | DEPARTMENT |
| r | listname | LISTBOX |
| r | selector_field | |
| r | selector_value_eq | |
| r | codefield | DEPARTMENT |
| r | captionfield | DEPTDESC |

allow_multi_selectionsはその名の通り複数選択許可を指定するもので、true() は複数選択が可能、そしてfalse() は単一選択のみ可能であることを示します。

- listnameプロパティには、リスト・ボックスにデータを追加する際に使用する作業リスト名を指定します。
- codefieldプロパティはリスト・エントリーのコード値を保持するlistnameリスト内のフィールド名です。(この場合、DEPARTMENT 部門コード)
- captionfieldプロパティはリスト・ボックスに表示されるフィールド名です。(この場合、DEPTDESC 部門記述)

これらのエントリーが揃えば、リストボックスにデータを追加しウェブ・ページに表示することができます。

残りの2つのプロパティはリスト・ボックスのエントリーの選択に関係しています。

- multi_select_listnameプロパティには、選択されたエントリーをWAMに返すリストの名前を指定します。
- multi_select_codefieldプロパティは選択されたエントリーのコード値を保持するフィールド名です。

メインリストの構築

ソース・エディターでShowPageWebroutineをもう一度見てください。

```

Webroutine Name(ShowPage)
  Web_Map For(*output) Fields((#ListBox *private) (#Selected *private))

  Select Fields(*ALL) From_File(deptab)
    Add_Entry To_List(#ListBox)
  Endselect

  #DEPARTMENT := ADM
  Add_Entry To_List(#Selected)

  #DEPARTMENT := INF
  Add_Entry To_List(#Selected)

Endroutine

```

Web_mapに、リスト・ボックスに関連した2つのリストがあることに注意してください。

Selectループを使用して、DEPTABファイルから全部門がメインリストである#ListBoxに追加されます。

それから、2つのエントリーが#Selectedリストに追加されます。これにより、表示する際リスト・ボックスで管理部と情報システム部が事前選択されます。

選択されたエントリーの処理

ProduceReportWebroutineはWebページ上のプッシュ・ボタンを押した時に呼び出されます。

```

Webroutine Name(ProduceReport)
  Web_Map For(*input) Fields(#Selected)
  Web_Map For(*output) Fields(#Report)

  Def_List Name(#Report) Fields(#DEPARTMENT) Type(*Working)

  Selectlist Named(#Selected)
    Add_Entry To_List(#Report)
  Endselect

Endroutine

```

ここでもWeb_mapに注目してください。選択されたエントリーの入力リスト(#Selected)と、もうひとつ、出力リスト(#Report)です。これらを使用して、選択された部門をリスト表示します。

8.1.18 メニューバー(std_menubar)

[クイック・スタート - メニューバー](#) [リストを使用したメニュー定義](#)
[メニュー・アイテム・デザイナの使用](#) [プロパティ - メニューバー](#)

メニューバー・ウェブレットはメニューバー機能を提供し、Webroutineを含んだ別のWebページを呼び出せるようになります。メニューバーは水平あるいは垂直に調整でき、最上位のメニュー・アイテムはマウスがその上に移動すると、更に次のメニューをポップアップします。水平に調整されたメニューバーは次のようになります。- この例では、2つのレベルのポップアップ・メニューが表示されています。



メニュー・アイテムは、メニュー・アイテムでデザイナを使用してWebroutineの設計時に定義するか、または実行時にRDMLXリストとして提供します。このウェブレットはjQuery UIメニュー・ウィジェットをベースにしており、操作にはjQueryおよびjQuery UIが必要です。(ウェブレットは自動的に必要な外部リソースをHTML出力に追加します。)

クイック・スタート - メニューバー

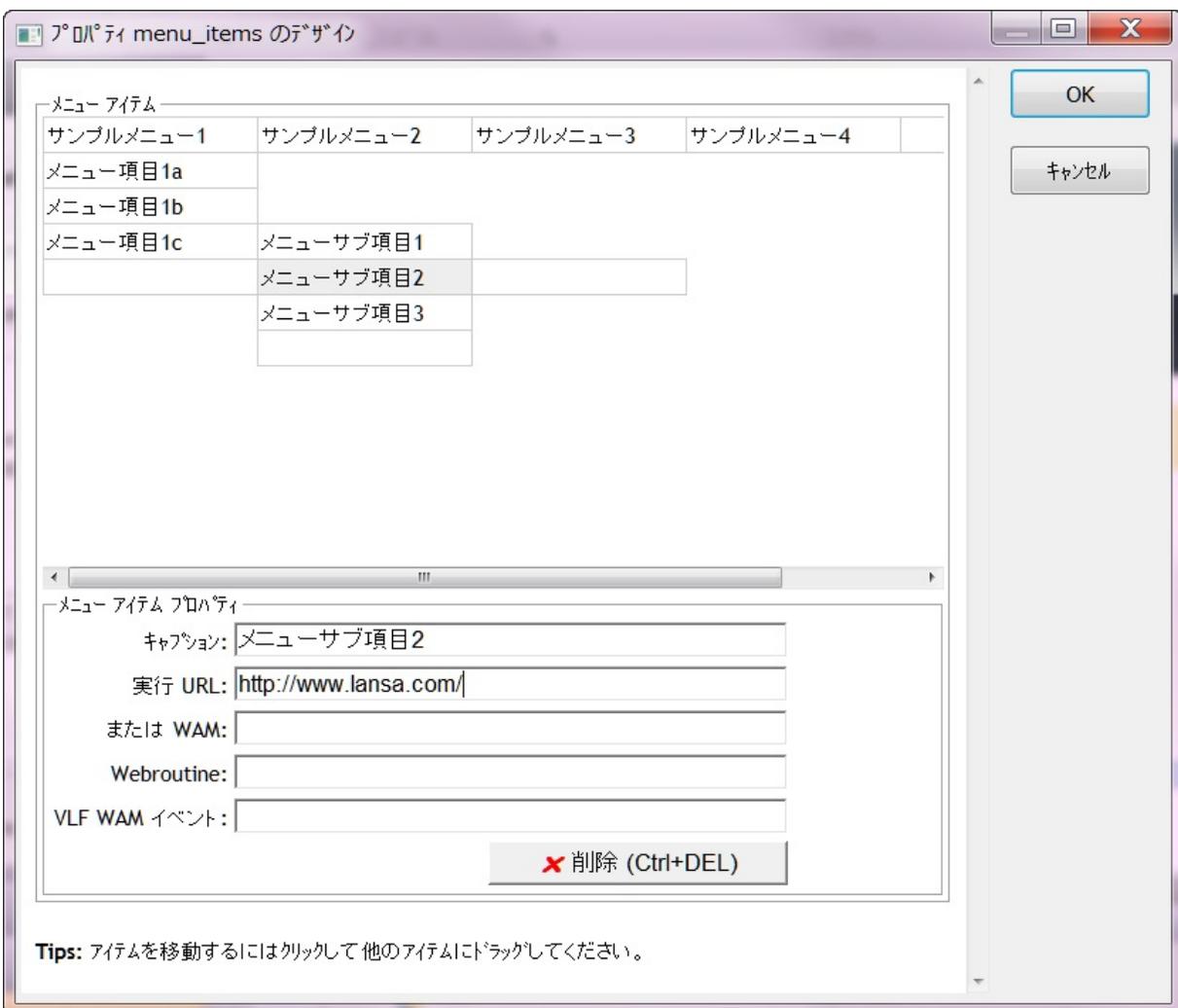
メニューバー・ウェブレットを使用するには、LANSAエディターで Webroutineを開いて、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[メニューバー]ウェブレットを探します。
2. [デザイン]ビューでウェブレットをページにドラッグ・アンド・ドロップしてください。ページ上でウェブレットを選択し、[詳細]タブをクリックするのを忘れないでください。
3. menu_items プロパティの横にある省略記号(...)のボタンをクリックし、メニュー・アイテムのデザイナを開き、メニュー・アイテムを定義します。

メニュー・アイテム・デザイナーの使用

メニュー・アイテム・デザイナーを開くには、以下の手順に従ってください。

1. ページ上でメニューバー・ウェブレットが選択され、[詳細]タブがクリックされていることを確認します。
2. マウスポインタを[詳細]タブのmenu_itemsプロパティに移動します。省略記号(...)がプロパティの値の横に表示されます。省略記号のボタンをクリックし、メニュー・アイテムのデザイナーを開きます。下のようなウィンドウが開きます。



ウィンドウの上半分はメニューの現在の状態を示しています。これは orientation プロパティの現在値に関わらず、常に水平方向に表示されることに注意してください。このウィンドウの上半分では、次のことがで

きます。

- 項目をクリックし、下半分で詳細を入力する。
- ドラッグ・アンド・ドロップして項目を並び替える。
- 現在選択されている項目の隣に表示されるブランクの項目をクリックして、新しい項目を追加する。
- 項目を選択、[削除]ボタンをクリックして項目を削除する

ウィンドウの下半分では選択した項目の詳細を次のように指定することができます。

[キャプション]

メニュー・アイテムに表示するテキストを指定します。ウィンドウの上半分でメニュー・アイテムをクリックしてタイプし、テキストを直接入力することもできます。

[実行URL]

メニュー・アイテムがクリックされた時に移動するURLを指定します。URL全体を指定することも、現在のページに関連付けられたURLを指定することもできます。

[WAM]と[Webroutine]

メニュー・アイテムがクリックされた時に呼び出されるWebroutineを指定します。URLとWebroutineの両方が指定された場合は、URLが優先され、Webroutineは無視されます。

[VLF WAM イベント]

VLF WAM イベントを指定して、このメニュー・アイテムが選択された時にハンドラーに引き渡すことができます。 .

リストを使用したメニュー定義

メニュー構造はRDMLXリストを使って定義することもできます。そのためには、特殊なフォーマットのリストを作成する必要があります。リストには次のような値を含む6つのカラムが含まれていなければいけません。

1. **メニュー・アイテムID** - メニュー・アイテムの一意的IDです。これは、数字でも文字列でも構いません。数字を使用する場合、ゼロは使用しないでください。(IDがゼロの項目は子の項目を持ってません。)
 2. **親項目ID** - 親項目のIDです。親のない最上位の項目には、ゼロまたは空の文字列を使います。
 3. **キャプション** - メニュー・アイテムに表示するテキストです。
 4. **URL** - メニュー・アイテムがクリックされた時に移動するURLです。URL全体を指定することも、現在のページに関連付けられたURLを指定することもできます。メニュー・アイテムがWebroutineを呼び出す時は空の文字列を使用します。
 5. **WAM** - メニュー・アイテムがクリックされた時に呼び出されるWebroutineを含むWAMの名前です。URLが指定されている場合は無視されます。
 6. **Webroutine** - メニュー・アイテムがクリックされた時に呼び出されるWebroutineの名前です。URLが指定されている場合は無視されます。
 7. **WAM イベント (任意)** - この要求を処理するWebRoutineに送信するVLFのWAMイベント
- カラム名は自由に付けて構いませんが、上記に示す順で定義される必要があります。

プロパティ - メニューバー

メニューバー・ウェブレットのプロパティは以下のとおりです。

id menu_items show_arrows
listname orientation submit_selected_to

id

ウェブレットの一意の名前。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用することも可能です。

省略値

`concat('o', position(), '_LANSA_n')` - LANSАにより付けられたウェブレット内部名です。

有効値

有効なHTMLのID文字列。[A-Za-z]の文字で始まらなければいけません。その後には文字、[0-9]の数字、ハイフン("-")、アンダースコア("_")、コロン(":"), およびピリオド(".")のいずれでも使用でき、好きな数だけ続けることができます。ただし、様々なJavaScriptライブラリ、CSSエディターやその他のWebテクノロジーとの互換性を考えると、コロン(":")とピリオド(".")は避けた方が賢明です。

listname

メニューバーとその項目の定義を含むRDMLXリストの名前。このプロパティの使用方法の詳細は「[リストを使用したメニュー定義](#)」を参照してください。

省略値

ブランク。

有効値

現Webroutineからの有効な出力リスト。

menu_items

メニュー・アイテムを指定するXMLノード・セット。これは、メニューをデザインビュー上にドラッグしたときに、システムが生成する値です。

表示された値を直接編集しないでください。代わりに、省略記号(...)のボタンをクリックし、メニュー・アイテム・デザイナを開きます。詳細については、「[メニュー・アイテム・デザイナの使用](#)」を参照してください。

省略値

```
document(")/*/lxml:data/lxml:menu[@id='<unique id>']
```

<unique id> は自動的に生成される識別子です。

有効値

適用不可。(この値はシステム生成されます。修正してはいけません。) [listname](#) プロパティが指定されている場合、この値は無視されません。

orientation

メニューの方向。これは、互いに関係する最上位のメニュー・アイテムの位置及び方向、およびポップアップ・メニューが表示される位置を決定します。

省略値

'top'

有効値

プロパティ・シートのこのプロパティの横にあるドロップダウン・ボタンをクリックし、以下の値から1つを選択します。

'top' 最上位のメニュー・アイテムは水平方向に置かれ、第一レベルのポップアップ・メニューは対応する最上位のメニュー・アイテムの下に表示されます。ページ上部、またはその付近の横方向メニューバーとしての使用に適しています。

'left' 最上位のメニュー・アイテムは垂直方向に置かれ、第一レベルのポップアップ・メニューは対応する最上位のメニュー・アイテムの右側に表示されます。ページの左側に位置する縦方向のメニューバーとしての使用に適しています。

'right' 最上位のメニュー・アイテムは垂直方向に置かれ、第一レベルのポップアップ・メニューは対応する最上位のメニュー・アイテムの左側に表示されます。ページの右側に位置する縦方向のメニューバーとしての使用に適しています。

'bottom' 最上位のメニュー・アイテムは水平方向に置かれ、第一レベルのポップアップ・メニューは対応する最上位のメニュー・アイテムの上に表示されます。ページ下部、またはその付近の横方向メニューバーとしての使用に適しています。

show_arrows

最上位のメニュー・アイテムに矢印を表示するかどうかを示す、ブール値です。これは最上位のメニューバー項目にのみ適用されます。サブメニューのあるメニュー・アイテムには常に矢印が表示され、サブメニューの存在を示します。

省略値

`true()`

有効値

`true()`、`false()` もしくは有効なブール値式。

submit_selected_to

メニュー・アイテムがクリックされ、そのメニュー・アイテムのアクションがWebroutineを呼び出すことである場合、この選択されたメニュー・アイテムのIDはこのフィールドに置かれます。RDMLXリストで定義されたメニューバーの場合、IDは列1に指定されたIDになります。メニュー・アイテムがデザイナーで指定されたメニューバーの場合、値はカンマで区切られ、選択された項目のパスを表わすシーケンス番号になります。例えば、2番目のメニューの3番目の項目が選択された場合、送られるIDは"2,3"になります。

省略値

空白。

有効値

実行されているWebroutine内の*INPUTフィールド名。

8.1.19 メニュー・アイテム (std_menu_item_v2)

クイック・スタート - メニュー・アイテム プロパティ - メニュー・アイテム

メニュー・アイテムウェブレットは、プッシュ・ボタンに見た目も動作もよく似ているハイパーリンクのメニュー・アイテムを提供します。この目的は、ページ内で1つのグループにまとめられたものを使って、代替ページのメニュー、または現在のページからユーザーがアクセスできるWebroutineを提供したりすることです。

このメニュー・アイテムは基本的には、追加の機能が付いたクリック可能なリンクです。

"選ばれた"スタイルのサポートも含まれています。

複数のスパンで構成されており、複数のイメージや拡張された背景 ("スライディング・ドア"のテクニックを使用) が可能です。

クイック・スタート - メニュー・アイテム

メニュー・アイテムウェブレットは、省略値の標準Webroutineのレイアウトに骨格となるメニューを提供します。レイアウトにこの機能を提供するために使用することもできますし、各Webroutineのページで直接使用することもできます。メニュー・アイテムウェブレットを使用するには、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[Menu item]ウェブレットを探します。
2. [デザイン]ビューでウェブレットをページ上にドラッグ・アンド・ドロップしてください。(テーブルやその他のデバイスを利用して、ウェブレットの複数インスタンスを調整しても構いません。)ウェブレットをクリックして、[詳細]タブをクリックしてください。
3. 必要に応じて、captionプロパティを設定します。
4. on_click_wrnameプロパティに、ハイパーリンクがクリックされた時に呼び出されるWebroutineの名前を設定してください。Webroutineが現在のWebroutineと異なるWAMにある場合は、on_click_wamnameプロパティも設定する必要があります。もしくはhrefプロパティを設定して、Webroutine以外のウェブレット移動先を指定することもできます。

メニュー・アイテムの外観

レイアウトとサイズ

基本のメニュー・アイテムには、太字テキストとCSS表示値"block"以外は特別なフォーマットはありません。使用しているレイアウトとテーマによっては、これを大幅に変更することが可能です。Visual LANSA V12 SP1以前に提供されている省略値のレイアウトには3つのテーマ、default、royal、grassがあります。



Default

Royal

Grass

jQuery UIテーマの新しいレイアウトを使用している場合は、usejQueryUIThemeプロパティにtrueを設定することで、クリック可能項目の標準テーマの外観をメニュー・アイテムに取り入れることができます。

| | | |
|-----------|------------|----------------|
| | | |
| | | |
| Cupertino | Smoothness | Pepper Grinder |

テーマに何も適用されていない、またはusejQueryUIThemeプロパティにfalseが設定されている場合は、メニュー・アイテムは枠線のない白いボックス内に太字のテキストで表示されます。

Sample Menu Item

この基本となるHTML構造は次のようになります。

```
<a class="std_menu" id="webletName">  
<span class="std_menu_span1">  
<span class="std_menu_span2">
```

Sample Menu Item

```
</span>
```

```
</span>
```

```
</a>
```

メニュー・アイテムが選択されると、`<a>`タグのクラスは"std_menu_selected"に変更されます。

メニュー・アイテムの基本的なスタイルは、スタイルシートでstd_menuとstd_menu_selectedクラスのプロパティを定義することで決定できます。例えば次のようになります。

```
a.std_menu, a.std_menu_selected {
  color:#01478c;
  border-style:solid;
  border-color:#7db0e5;
  border-width:0px 2px 1px 2px;
  background-color:#c7dff4;
}
a.std_menu_selected {
  background-color:#7db0e5;
}
```

左揃えのイメージの追加は次のようにします。

```
a.std_menu, a.std_menu_selected {
  background-image:url('icon.png') no-repeat left center;
  padding-left:32px;
}
```

イメージの半分2つを各スパンに適用し、"sliding doors" テクニックとして知られる方法を使って、更に複雑な"自動調整の"背景イメージを作成することもできます。

レイアウトとサイズ

省略値のメニュー・アイテムは"ブロック"項目です。つまり、与えられたスペースの幅まで自動的に広げられ、（これを含むボックス内の）両端に他のコンテンツを受け入れません。これは、サイドバー内に縦方向に並ぶメニュー・アイテムの作成には理想的です。サイドバーのサイズを設定すれば、メニュー・アイテムは全て自動的にそのサイズに合わせられます。別の方法でメニュー・アイテムを使用したい場合は、表示モードとサイズのいずれか、または両方を変更します。例えば、以下のようなメニュー・アイテムの水平方向バーを作成する場合：



"hMenu"というIDで<div>の中にメニュー・アイテムを配置して、次のようなCSSを定義します。

```
#hMenu {
  background:#333;
}
a.std_menu, a.std_menu_selected {
  color:#fff;
  display:inline-block;
  padding:5px 10px;
  border-right:1px solid #fff;
  background-color:#c7dff4;
  font-weight:normal;
}
```

プロパティ - メニュー・アイテム

メニュー・アイテムウェブレットのプロパティは以下のとおりです。

| | | |
|----------------|---------------------|--------------------|
| caption | is_selected_if_also | reentryvalue |
| class | name | selected_class |
| force_selected | on_click_wamname | style |
| formname | on_click_wrname | tab_index |
| hide_if | protocol | target_window_name |
| href | reentryfield | usejQueryUITheme |
| | | vf_wamevent |

name

ウェブレットを識別する名前です。ウェブレットがフィールドをビジュアライズすると、これがフィールド名になります。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用することも可能です。

省略値

自動的に生成された、一意の識別子。

有効値

単一引用符で囲まれた文字列。

caption

メニュー・アイテムに表示されるテキストです。

省略値

'Caption' (通常はこのプロパティに値を設定してください。)

有効値

単一引用符で囲まれたテキスト、フィールド名、システム変数、または複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

href

このプロパティを使ってメニュー・アイテムが移動する先のURLを指定します。指定する場合、URLにはリテラル値（例えば'http://www.mycompany.com/'など）または実行時にURLを含むフィールド名を指定できます。

このプロパティはon_click_wamname、 on_click_wrname及びprotocolプロパティよりも優先されます。hrefが指定されていた場合は、これらのプロパティは無視されます。

省略値

なし。

有効値

単一引用符で囲まれたURL、実行時にURLが含まれるフィールド名、システム変数名、複数言語変数名。

on_click_wamname

メニュー・アイテムがクリックされたときにWebroutineが実行されるWAMの名前を指定します。(Webroutine名はon_click_wnameプロパティに指定します。)

hrefプロパティが指定された場合は、このプロパティは無視されます。

省略値

指定しない場合は、現在のWAMが使用されます。(\$lweb_WAMName)

有効値

単一引用符付きのWAM名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

on_click_wrname

メニュー・アイテムがクリックされた時に実行されるWebroutineの名前を指定します。(WebroutineのあるWAM名はon_click_wamnameプロパティで指定します。)

hrefプロパティが指定された場合は、このプロパティは無視されます。

省略値

省略値は適用されません。 - メニュー・アイテムを機能させるには hrefプロパティ、もしくはon_click_wrnameプロパティのいずれかを指定しなくてははいけません。

有効値

単一引用符付きのWebroutine名。Webroutineは、on_click_wamnameプロパティに指定されたWAMに存在してはいけません。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWebroutineのリストが表示されます。

protocol

on_click_wnameプロパティに指定されたWebroutineへの移動に使用する
プロトコル(例:http:// または https://)。

通常、このプロパティはセキュア・モードの処理との切り替えが必要な
時に使用します。それ以外の場合は、通常このプロパティを指定する必
要はありません。

省略値

ブランク。現在使用されているプロトコルが使用されます。

有効値

単一引用符付きの、有効なプロトコル。指定する場合は、通常は
'http:'または'https:'です。

is_selected_if_also

省略値では、on_click_wamnameとon_click_wrnameプロパティの値が現在のページのWAMとWebroutineの名前に一致した場合、メニュー・アイテムは選択された状態の表示だと仮定されます。(この比較では大文字小文字が区別されることに注意してください)

このプロパティに追加の条件を指定して適用することで、選択された状態であると仮定されるメニュー・アイテムを更に絞り込むことができます。

省略値

true() (追加の条件は適用されません。)

有効値

ブール値を返す有効なXPath式。

例

次の例では、上記で説明された省略値テストに加えて、フィールド#STD_FLAGが'X'かどうか、ウェブレットでテストされます。この式は以下のような形式で入力してください。



The screenshot shows a developer console with a search bar containing 'is_selected_if_also' and a text input field containing the XPath expression '#STD_FLAG = 'Y''. The text input field is highlighted with a blue border.

プロパティがフォーカスを失うと、この式は以下のように表示されます。



The screenshot shows a developer console with a search bar containing 'is_selected_if_also' and a text input field containing the XPath expression 'key('field-value', 'STD_FLAG') = 'Y''. The text input field is highlighted with a blue border.

force_selected

trueの場合、現在のWAM/Webroutineやis_selected_if_alsoプロパティに関係なく、メニュー・アイテムが選択された状態だと仮定されます。自身のロジックを適用させてメニュー・アイテムの状態を決定したい場合、is_selected_if_alsoにfalseを設定して省略値のロジックを無効にし、force_selectedに適切なブール値の式を入れます。

例えば、選択されたメニュー・アイテムの名前を含むフィールドを作成します。そして、force_selectedプロパティに(XPathのエントリ・エリアを利用して)次のような式を設定します。

```
#SELMENU = 'Menu1'
```

省略値

False() (省略値のロジックが適用されず。)

有効値

ブール値を返す有効なXPath式。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

例

次の例はフィールド#STD_FLAGが 'Y' と等しい場合にウェブレットを非表示にします。この式は以下のような形式で入力してください。

| | |
|---------|-----------------|
| hide_if | #STD_FLAG = 'Y' |
|---------|-----------------|

プロパティがフォーカスを失うと、この式は以下のように表示されます。

| | |
|---------|--------------------------------------|
| hide_if | key('field-value', 'STD_FLAG') = 'Y' |
|---------|--------------------------------------|

target_window_name

メニュー・アイテムの移動先を表示するウィンドウやフレームの名前。特別な値、'_blank'は、名前のない新しいウィンドウを開きます。

省略値

空白。メニュー・アイテムの移動先は現在のウィンドウに表示されます。

有効値

単一引用符で囲まれたウィンドウ名またはフレーム名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるウィンドウやフレームが表示されます。

reentryfield

reentryvalue プロパティに指定された値をWAMに送る時に使用するフィールド名です。フィールド名は単一引用符で囲みます。

詳しくは、reentryvalue プロパティの説明を参照してください。

注:このプロパティは、WEBEVENTアプリケーションでよく使用される、再入可能なプログラミング技術をサポートするために提供されています。最初からWAMを使用するように設計されたウェブ・アプリケーションでは、通常はこの技術を使用する必要はありません。

省略値

省略値は適用されません。

有効値

単一引用符で囲まれた文字列。

reentryvalue

ターゲットWebroutineのreentryfieldプロパティで指定されたフィールドに送信する値です。英数字フィールドの場合、値は単一引用符付きで指定する必要があります。数字フィールドの場合は、単一引用符はなくても構いません。

このプロパティは、reentryfieldプロパティと共に使用され、値をターゲットのWebroutineにどのように送信するか記述します。以下の2つの情報が必要になります。

1. reentryfield: ターゲットWebroutineが情報を参照するために使用するフィールド名。
2. reentryvalue: リテラル値もしくは必要な情報を含むこの(ソース)Webroutineのフィールド名。

注: このプロパティは、WEBEVENTアプリケーションでよく使用される、再入可能なプログラミング技術をサポートするために提供されています。最初からWAMを使用するように設計されたウェブ・アプリケーションでは、通常はこの技術を使用する必要はありません。

省略値

省略値は適用されません。

有効値

単一引用符付きテキスト、もしくはフィールド、システム変数、複数言語対応テキスト変数の名前。

formname

サーバーに送られるHTMLフォーム名です。

省略値

'LANSA'

有効値

単一引用符付きのフォーム名。プロパティ・シート内で該当するドロップダウン・ボタンをクリックすると、使用できるフォーム名リストが表示されます。

style

ウェブレットのインライン・スタイルのプロパティです。このプロパティを使って、ウェブレットにCSSのstyleプロパティを設定でき、レイアウト・スタイルシートに定義されているどんな値も書き換えることができます。

省略値

''

有効値

有効なCSSプロパティ。セミコロンで区切り、単一引用符をつけてください。

usejQueryUITheme

ウェブレットがjQuery UIクラスを適用すべきかどうかを示します。Trueの場合、レイアウトでjQuery UIがロードされ、テーマにより定義されたクリック可能エレメントの外観をレイアウトで使用できます。

省略値

false

有効値

ブール値を返す有効なXPath式。

class

メニュー・アイテムのカスケード・スタイル・シート(CSS)クラス名。

省略値

メニュー・アイテムに提供されたクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

selected_class

メニュー・アイテムが選択された時のカスケード・スタイル・シート (CSS) クラス名。

省略値

提供されているウェブレットのクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

vf_wamevent

VLF のWAM イベント文字列です。

省略値

空白。

有効値

文字列。カンマ (,) は使用できません。

8.1.20 ナビゲーション・パネル (std_nav_panel)

[クイック・スタート - ナビゲーション・パネル](#)

[プロパティ - ナビゲーション・パネル](#)

ナビゲーション・パネル・ウェブレットはコンテナを提供し、これを使ってWAMやURLから提供されるコンテンツを表示することができます。ですから、全てのコンテンツが1つのソースから提供されていない場合に、アプリケーションでこれを使用して、'モジュール化'することができます。

クイック・スタート - ナビゲーション・パネル

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[ナビゲーション パネル]ウェブレットを探します。
2. デザイン中のWebページにドラッグ・アンド・ドロップします。
3. nav_wrnameプロパティに、Webページが表示されたときに呼び出されるWebroutineの名前を設定します。Webroutineが現在のWebroutineと異なるWAMにある場合は、nav_wamnameプロパティもセットする必要があります。

プロパティ - ナビゲーション・パネル

ナビゲーション・パネル・ウェブレットのプロパティは以下のとおりです。

| | | |
|--------------------|----------------------------|-----------------------------|
| border | nav_wamname | transparent |
| border_width | nav_wrname | wait_content |
| class | pos_absolute | wait_content_absolute_imag |
| formname | protocol | wait_content_class |
| height | reentryfield | wait_content_image_alignme |
| hide_if | reentryvalue | wait_content_image_class |
| name | scrolling | wait_content_relative_image |
| nav_asynchronously | size_panel_to_content | wait_content_timeout |
| nav_url | size_panel_to_content_axis | width |
| | | vf_wamevent |

name

ウェブレットの名前。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript や ウェブレットを参照するXSLを使用する場合は、独自の名前を使用しても構いません。

省略値

`concat('o', position(), '_LANSA_n')` - LANSAにより付けられたウェブレット内部名です。

有効値

単一引用符で囲まれた名前。

border

枠線のスタイル。

省略値

blank。ウエブレットには枠線がありません。

有効値

単一引用符付きの、枠線のスタイル。有効なスタイルは、プロパティシートで該当するドロップダウン・ボタンをクリックして、リストから選択することができます。'window-inset'はInternet Explorerでのみサポートされます。

border_width

パネルの枠線の幅。

省略値

空白。パネルに枠線がない状態です。

有効値

単一引用符で囲まれた有効な範囲内の幅。プロパティ・シートの該当するドロップダウン・ボタンをクリックして、'thick'、'medium' もしくは 'thin' という幅を選択することもできます。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

例

次の例はフィールド#STD_FLAGが 'X' と等しい場合にウェブレットを非表示にします。式は入力されなくてはなりません。またプロパティにフォーカスがある時、以下のように表示されます。

| | |
|---------|-----------------|
| hide_if | #STD_FLAG = 'X' |
|---------|-----------------|

プロパティがフォーカスを失うと、この式は以下のように表示されます。

| | |
|---------|--------------------------------------|
| hide_if | key('field-value', 'STD_FLAG') = 'X' |
|---------|--------------------------------------|

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

Webページのウェブレットの幅です。

省略値

'100%' - パネルはページ幅に調整されます。

有効値

単一引用符で囲まれた有効な範囲内の幅。

height

Webページのウェブレットの高さ。

省略値

'250pt'.

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

size_panel_to_content

パネルのサイズ調整方法を決定するブール値プロパティ。移動先のページは現在のページと同じドメイン内でないといけません。

使用されるサイズはブラウザによって決定され、ブラウザごとにわずかに異なります。初期設定のサイズにも影響を受けます。一番よい結果を得るためには、パネルのコンテンツ・ページのサイズを明確にして、ブラウザにより決定される必要がないようにします。また、Operaはパネルのサイズを縮小することはありませんので注意してください。

`nav_url`を使用する場合、URLはWAMと同じサーバーからでなくてはなりません。ブラウザのセキュリティ制限により、あるサーバーからのページのプログラムは、他のサーバーからのページの情報にアクセスできないようになっていきます。パネルのコンテンツが別サーバーからのものである場合、パネルはそのサイズを決定できません。別サーバーを指すURLを使用すると、プロンプターのコードは"アクセス拒否"エラーで失敗します。

省略値

`false()` - 移動時、パネルはサイズ調整されません。

有効値

`true()`、`false()`、`'once'`、もしくは有効な式。`true()`を設定すると、フレームはナビゲーションの度に移動先のページのコンテンツのサイズに調整されます。`'once'`を設定すると、サイズ調整は最初のナビゲーション時のみ行われ、それ以降のナビゲーションではパネルはそのサイズのままになります。

size_panel_to_content_axis

size_panel_to_contentがfalse()でない場合、コンテンツのサイズ調整の性質を指定します。

このプロパティ時は細心の注意を払うようにしてください。パネルは指定通りになるよう最善を尽くされますが、コンテンツのサイズやスクロール・バーの配置など、ブラウザの独自のルールに従って決定されるものもあります。これにより、予期せぬ動作につながる場合もあります。特に、エレメントのサイズ決定にパーセントを使用するレイアウトは、予期せぬスクロール・バーになる可能性もあります。

省略値

'both'- パネルの高さと幅のサイズが変更されます。

有効値

'height'を設定すると、パネルの高さのみのサイズ調整します。'width'を設定すると、パネルの幅のみのサイズ調整します。'both'を設定すると、パネルの高さと幅のサイズを調整します。

scrolling

スクロールが許されるかどうかを決定します。

省略値

'both'

有効値

コンテンツがパネルに入りきるかどうかに関係なく、'yes'を設定するとスクロールが許可され、'no'を設定すると、スクロールが許可されません。'auto'を設定すると、必要な場合にスクロールが許可されます。

class

ウェブレットのカスケード・スタイル・シートのクラス名。
クラスは現在のレイアウトにロードされる外部CSSファイルに定義されていなくてははいけません。

省略値

'std_nav_panel'- ウェブレット用に提供されるクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

transparent

パネルを作るIFRAMEのallowTransparent属性を設定するブール値プロパティ。これはInternet Explorer(7およびそれ以下)限定の属性であり、trueの時、パネルにロードされたドキュメントの背景色を透明にすることができます。

このプロパティはパネルを透明にするのではなく、ただ単にパネルのコンテンツを透明にすることができるということに注意してください。更に、コンテンツのドキュメントはその本文の背景色を透明に設定する必要があります。このプロパティがfalseの場合は、コンテンツのドキュメントに透明な背景色が選択されていても、無視されます。

これはInternet Explorer(7およびそれ以下)、またはInternet Explorer 8以上を互換モードで実行している時のみでサポートされる非標準のプロパティです。他のブラウザ(標準モードで実行のInternet Explorer 8以降を含み)では、このプロパティはサポートされません。これらは、transparentがtrueである時と同様の動作になります。ただし、コンテンツのドキュメントに背景色が指定されていない場合は、ブラウザによって省略値は透明や白となります。そのため、異なるブラウザで一番良い結果を得るには、このプロパティをtrueのままにし、コンテンツのドキュメントの背景色を明確に設定してください。

省略値

true()

有効値

true() もしくは false()

nav_url

パネルの移動先URL。

省略値

ブランク - パネルはURLに移動しません。

有効値

単一引用符で囲まれた有効なURL。プロトコル(例：http:// や https://)を指定しなければいけないことに注意してください。

formname

サーバーに送られるHTMLフォーム名です。

省略値

空白。

有効値

単一引用符付きのフォーム名。プロパティ・シート内で該当するドロップダウン・ボタンをクリックすると、使用できるフォーム名リストが表示されます。

nav_wamname

パネルにページを表示するために呼び出されるWAMの名前。

省略値

空白 - nav_wname プロパティが指定されている場合は、現在のWAMが呼び出されます。

有効値

単一引用符で囲まれたWAMの名前。プロパティシート of 該当するドロップダウン・ボタンをクリックすると、使用可能なWAM名の一覧から選択することができます。

nav_wrname

パネルにページを表示するために呼び出されるWebroutineの名前。

省略値

ブランク - パネルにページを表示する為にWebroutineは呼び出されません。

有効値

単一引用符で囲まれたWebroutine名。プロパティシートの該当するドロップダウン・ボタンをクリックして、有効なWebroutine名の一覧から選択することができます。

protocol

このウェブレットに呼び出されたWebroutineへの移動に使用するプロトコル(例えば、http:// もしくは https://)

省略値

空白。現在使用されているプロトコルが使用されます。

有効値

単一引用符付きの、有効なプロトコル。通常は'http:'または'https:'です。

nav_asynchronously

nav_urlプロパティにURLが指定されている場合、パネルがロードされる方法をコントロールするブール値プロパティ。nav_urlが指定されていない場合は、無視されます。

trueの場合は、urlは新しいスレッドを使用してロードされます。これにより、ページがロードされたことを示すonLoadイベントを起動する前に、ブラウザはパネルがロードするのを待つ必要がないという効果があります。nav_asynchronouslyがfalseの場合は、ブラウザはパネルがロードし終わるのを待ってからこのイベントを起動します。

| |
|---|
| <p>このプロパティはOperaではサポートされていません。Operaは実際の値に関係なく、nav_asynchronouslyがtrueと同様の動作になります。</p> |
|---|

省略値

true() - nav_url のURL (指定されている場合)に非同期的に移動します。

有効値

Trueが設定されている場合、非同期的にnav_urlプロパティに指定されているURLに移動します。(つまり、パネルがロードされる前に残りの外側のページがロードされます。)

reentryfield

reentryvalue プロパティに指定された値をWAMに送る時に使用するフィールド名です。フィールド名は単一引用符で囲みます。

省略値

空白 - reentryfieldは使用されません。

有効値

リポジトリもしくはWAMで定義されたフィールド名。プロパティシート内で該当するドロップダウン・ボタンをクリックすると、使用できるフィールド名リストが表示されます。

reentryvalue

reentryfieldプロパティに指定されたフィールドに送られる値。英数字フィールドの場合、値は単一引用符付きで指定する必要があります。数字フィールドの場合は、単一引用符はなくても構いません。

省略値

ブランク - reentryfieldは使用されません。

有効値

適切なリテラル。

wait_content

パネルのロード中に表示されるテキスト。HTMLも指定できます。

省略値

ブランク - パネルのロード中にテキストは表示されません。

有効値

適切なリテラル、フィールド名、システム変数名、もしくは複数言語対応テキスト変数名。フィールド、システム変数、複数言語変数の名前は、プロパティシートで該当する省略記号(...)のボタンをクリックして、リストから選択することができます。

wait_content_timeout

wait_contentプロパティが指定されている場合、このプロパティは、ミリ秒単位での、ページが要求されるまでの待ち時間を示します。これによりイメージをロードする時間ができ、表示できるようになります。

省略値

100 - wait_contentプロパティが指定されている場合、ナビゲーション前に100ミリ秒の遅れが生じます。

有効値

ミリ秒単位の、有効な待ち時間。

wait_content_class

wait_contentプロパティに指定されたコンテンツのコンテナ・エレメントに適用されるカスケード・スタイル・シートのクラス。

省略値

空白 - スタイルは適用されません。

有効値

単一引用符付きの、カスケード・スタイル・シートの有効なクラス名。有効なスタイルは、プロパティシートで該当するドロップダウン・ボタンをクリックして、リストから選択することができます。

wait_content_relative_image

パネルのロード中に表示されるイメージの、イメージディレクトリに関連付けられたパスとファイル名。イメージは最初はサーバーから取得されますが、その後は可能な場合はブラウザのキャッシュから取得されま

省略値

空白 - イメージは表示されません。

有効値

単一引用符で囲まれた、有効なパスとイメージファイル名。イメージは、プロパティ・シートで該当する省略記号のボタンをクリックし、プロンプターから選択することができます。

wait_content_absolute_image

パネルのロード中に表示されるイメージのパスとファイル名イメージは最初はサーバーから取得されますが、その後は可能な場合はブラウザのキャッシュから取得されます。

省略値

空白 - イメージは表示されません。

有効値

単一引用符で囲まれた、有効なパスとファイル名。

wait_content_relative_imageプロパティが指定されている場合は、指定してはいけません。

wait_content_image_alignment

wait_content_relative_imageまたはwait_content_absolute_imageプロパティに指定されたイメージの、wait_contentプロパティに対する配置。

省略値

ブランク - 'left'と同じです。

有効値

'left'(左)、'right'(右)、'top'(上)または'bottom'(下)を設定します。

wait_content_image_class

wait_content_relative_imageもしくはwait_content_absolute_imageプロパティに指定されたイメージのカスケード・スタイル・シートのクラス。

省略値

空白 - イメージにはクラスはありません。

有効値

単一引用符付きの、カスケード・スタイル・シートの有効なクラス名。有効なスタイルは、プロパティシートで該当するドロップダウン・ボタンをクリックして、リストから選択することができます。

vf_wamevent

VLF のWAM イベント文字列

省略値

空白。

有効値

文字列。カンマ (,) は使用できません。

8.1.21 パネル(std_panel)

クイック・スタート - パネル プロパティ - パネル

パネル・ウェブレットは、Webページ上の他のコントロールのコンテナとして使用されます。通常Webページが相対的ではなく、絶対的に配置されることを確実にするために使用されます。ですから、テーブルや、アタッチメント・パネルといった'調整用'の別のコントロールと共に使用されます。

クイック・スタート - パネル

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[paneru]ウェブレットを探します。
2. デザイン中のWebページにドラッグ・アンド・ドロップします。
3. heightプロパティ及びwidthプロパティを指定、またはパネルのサイズ変更ハンドルをドラッグして、パネルを適切なサイズに変更してください。これで他のコントロールをこの上にドラッグできるようになります。これらのコントロールは、相対位置ではなく、絶対位置として配置されることに注意してください。

プロパティ - パネル

パネル・ウェブレットのプロパティは以下のとおりです。

`border` `height` `pos_absolute`

`border_width` `hide_if` `snap_to_grid`

`class` `name` `width`

`grid_size` `panes`

name

ウェブレットの名前。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript や ウェブレットを参照するXSLを使用する場合は、独自の名前を使用しても構いません。

省略値

`concat('o', position(), '_LANSA_n')` - LANSAにより付けられたウェブレット内部名です。

有効値

単一引用符で囲まれた名前。

panes

表示する一連のペインを指定するXMLノードセット。これは、[デザイン]ビューでバナーをペインにドラッグした時にシステムが生成する値です。修正できません。

省略値

`document(")/*//lxml:data/lxml:panes[@id='<unique id>']` (一意の識別子 unique idが自動的に生成された現在のペインです。)

有効値

適用不可。(この値はシステムにより保守されます。)

border

枠線のスタイル。

省略値

blank。ウエブレットには枠線がありません。

有効値

単一引用符付きの、枠線のスタイル。有効なスタイルは、プロパティシートで該当するドロップダウン・ボタンをクリックして、リストから選択することができます。'window-inset'はInternet Explorerでのみサポートされます。

border_width

パネルの枠線の幅。

省略値

ブランク。パネルに枠線がない状態です。

有効値

単一引用符で囲まれた有効な範囲内の幅。プロパティ・シートの該当するドロップダウン・ボタンをクリックして、'thick'、'medium' もしくは 'thin' という幅を選択することもできます。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

例

次の例はフィールド#STD_FLAGが 'X' と等しい場合にウェブレットを非表示にします。式は入力されなくてはなりません。またプロパティにフォーカスがある時、以下のように表示されます。

| | |
|---------|-----------------|
| hide_if | #STD_FLAG = 'X' |
|---------|-----------------|

プロパティがフォーカスを失うと、この式は以下のように表示されます。

| | |
|---------|--------------------------------------|
| hide_if | key('field-value', 'STD_FLAG') = 'X' |
|---------|--------------------------------------|

class

ウェブレットのカスケード・スタイル・シートのクラス名。

省略値

'std_panel' - このウェブレット用に提供されているクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

snap_to_grid

グリッドにスナップする機能を設計時に使用可能にするかどうかを決定するブール値プロパティ。グリッドにスナップすることにより、パネルに配置されたコントロールを更に簡単に整列させることができます。これは設計時のみの機能です。

省略値

true() - パネルのグリッドにスナップの機能は設計時に使用可能です。

有効値

true()を設定すると、グリッドにスナップが使用可能になり、false()を設定すると、使用不可になります。 .

grid_size

設計時のグリッドのポイント間の距離。ポイントを使用することで、グリッドに置かれたコントロールの整列がより簡単になります。これは設計時のみの機能です。

省略値

'10px' - グリッドのポイントは10ピクセル離して置かれます。

有効値

単一引用符付きの、有効な範囲にある距離。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

Webページのパネルの幅。

省略値

'400pt'

有効値

単一引用符で囲まれた有効な範囲内の幅。

height

Webページのパネルの高さ。

省略値

'200pt'

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

8.1.22 ページ印刷 (std_printpage)

クイック・スタート - ページ印刷 プロパティ - ページ印刷

ページ印刷ウェブレットは現在のページを印刷するハイパーリンクを提供します。例えば次のようなものです。



クイック・スタート - ページ印刷

Webページにページ印刷ウェブレットを追加するには、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[ページ印刷]ウェブレットを探します。
2. このウェブレットをWebページにドラッグ・アンド・ドロップしてください。

プロパティ - ページ印刷

ページ印刷ウェブレットのプロパティは以下の通りです。

| | | |
|-------------------------------------|--------------------------------|-------------------------------------|
| absolute_image_path | disabled_class | pos_absolute |
| caption | height_design | relative_image_path |
| class | hide_focus | tab_index |
| disabled | hide_if | width_design |

caption

ウェブレットのキャプション。

省略値

'Print'

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

disabled

ウェブレットを使用可または不可として表示するかを決定するブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false()、もしくは、TrueかFalseを返す有効な式。

hide_focus

評価がTrueの時にフォーカスのあるウェブレットのフォーカス用の四角を非表示にするブール値プロパティ。

省略値

true()

有効値

true()、false()、もしくは、ブール値を返す有効な式。

relative_image_path

表示されるイメージの'イメージ'・ディレクトリに関連付けられたパスとファイル名。指定する場合は、absolute_image_pathプロパティはリンクにすること。

省略値

'icons/normal/16/printer_16.png'

absolute_image_path

表示するイメージのパスとファイル名。指定する場合、relative_image_pathプロパティは空白でなければいけません。

省略値

省略値はrelative_image_pathプロパティに指定されたイメージを使用します。

有効値

単一引用符で囲まれたイメージのパスと名前。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width_design

Webページのウェブレットの幅です。

省略値

ブランク（ウェブレットは省略値の幅を使用します。）

有効値

単一引用符で囲まれた有効な範囲内の幅。

height_design

Webページのウェブレットの高さ。

省略値

ブランク(ウェブレットは省略値の高さを使用します。)

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

class

ウェブレットのカスケード・スタイル・シートのクラス名。

省略値

'std_printpage' - このウェブレット用に提供されているクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

disabled_class

disabledプロパティがTrueにセットされた時のウェブレットのカスケード・スタイル・シート。

省略値

'std_printpage_disabled'- このウェブレット用に提供されているdisabledクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

有効値

イメージ・ディレクトリに関連付けられたイメージのパスおよび名前、単一引用符で囲まれたもの。イメージは、プロパティ・シートで該当する省略記号のボタンをクリックし、プロンプターから選択することができます。

8.1.23 プロンプター (std_prompter)

プロパティ - プロンプター

プロンプター・ウェブレットは、ユーザーがポップアップ・パネルのカスタム・ユーザー・インターフェースを使って、1つもしくは複数のフィールドの値を選択できるメカニズムを提供します。

ウェブレットは次のようなボタンを作成します。



これをクリックすると、プロンプターがWebroutineを呼び出し、出力を次のようなパネルに表示します。

| | | | |
|------------------|----------------------|---|------------------------|
| Employee Number | <input type="text"/> |  | BROWN, VERONICA |
| Employee Surname | <input type="text"/> | | BLOGGS, FRED JOHN ALAN |
| | | | SMITHSON, FRED |
| | | | MISS SIMPSON, ANNE |
| | | | JONES, BEN |
| | | | SMYTHE, JOHN |
| | | | SMITHE, Robert |
| | | | SMITHSON, PAUL |
| | | | SMITHS, PETER |
| | | | SMITHERS, JACK |
| | | | SNELL, GEORGE |
| | | | SNEDDON, ALLAN |
| | | | SNASHALL, DAMIAN |
| | | | PERRY, WILLIAM |

このWebroutineは別のWebroutine、例えばリスト内のページやウィザードのステップなど、を呼び出すこともできます。最終的な値が決定すると、"終了"Webroutineとして知られる、事前定義されたWebroutineに出力します。プロンプターはこの終了Webroutineを認識し、値を現在のページにコピーして終わります。プロンプターの終了およびその結果へのアクセスについての詳細は、「[closing_wname](#)」、「[field_mapping](#)」、そして「[field_name_to_exchange](#)」を参照してください。

プロパティ - プロンプター

プロンプター・ウェブレットのプロパティは以下のとおりです。

| | | |
|-------------------------------------|---|--------------------------------|
| <code>absolute_image_path</code> | <code>field_name_to_exchange</code> | <code>pre_show_js</code> |
| <code>auto_resize</code> | <code>formname</code> | <code>prompter_class</code> |
| <code>border</code> | <code>hide_if</code> | <code>prompter_height</code> |
| <code>border_width</code> | <code>image_height</code> | <code>prompter_url</code> |
| <code>button_class</code> | <code>image_width</code> | <code>prompter_wamnan</code> |
| <code>button_height</code> | <code>name</code> | <code>prompter_width</code> |
| <code>button_mouseover_class</code> | <code>on_change_protocol</code> | <code>prompter_wrname</code> |
| <code>button_width</code> | <code>on_change_reentryfield</code> | <code>protocol</code> |
| <code>caption</code> | <code>on_change_reentryvalue</code> | <code>reentryfield</code> |
| <code>closing_url</code> | <code>on_change_target_window_name</code> | <code>reentryvalue</code> |
| <code>closing_wrname</code> | <code>on_change_wamname</code> | <code>relative_image_pa</code> |
| <code>disabled</code> | <code>on_change_wrname</code> | <code>tab_index</code> |
| <code>field_mapping</code> | <code>pos_absolute</code> | <code>title</code> |
| | | <code>vf_wamevent</code> |

name

ウェブレットの名前。ウェブレットがWAMからのフィールドを表す場合、これはフィールドの名前でなくてはなりません。ウェブレットがフィールドを表さない場合、ウェブレットを参照するJavaScriptやXSLを使用している場合、独自の名前を使用してもよいです。名前を指定する必要がない場合は、LANSAにより決定される省略値名を使用してください。

省略値

`concat('o', position(), '_LANSA_n')` - LANSAにより付けられたウェブレット内部名です。

有効値

単一引用符で囲まれた名前。省略値、またはウェブレットによって設定されたフィールド名。

caption

プロンプタのイメージの右側に表示するキャプション。

省略値

ブランク - プロンプターのイメージの右側にはテキストが表示されません。

有効値

単一引用符に囲まれた、フィールド、複数言語対応テキスト変数、システム変数の名前(プロパティ・シートで該当する省略記号(...))のボタンをクリックしてリストから1つ選ぶことができます。)

relative_image_path

表示されるイメージの'イメージ'・ディレクトリに関連付けられたパスとファイル名。指定する場合は、absolute_image_pathプロパティは空白にすること。

省略値

'search_1.gif'

有効値

イメージ・ディレクトリに関連付けられたイメージのパスおよび名前、単一引用符で囲まれたもの。イメージは、プロパティ・シートで該当する省略記号のボタンをクリックし、プロンプターから選択することができます。

absolute_image_path

表示するイメージのパスとファイル名。指定する場合、relative_image_pathプロパティは空白でなければいけません。

省略値

空白 - 省略値はrelative_image_pathプロパティに指定されたイメージを使用します。

有効値

単一引用符で囲まれたイメージのパスと名前。

image_height

プロンプター・ボタンのイメージの高さ。

省略値

'12pt'

有効値

単一引用符付きの、有効な範囲にある高さ。

image_width

プロンプター・ボタンのイメージの幅。

省略値

'12pt'

有効値

単一引用符付きの、有効な範囲にある幅。

border

プロンプターの枠線のスタイル。

省略値

blank(ウェブレットは省略値のスタイルを適用しあす。)

有効値

枠線固有のスタイル -

'dashed'、'dotted'、'double'、'groove'、'inset'、'outset'、'ridge'、'solid'、'window-inset'。'window-inset'はInternet Explorerでのみサポートされます。

border_width

ウェブレットの枠線の幅。

省略値

空白。ウェブレットは省略値の幅を採用します。

有効値

単一引用符で囲まれた有効な範囲内の幅。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

False() - ウェブレットは常に表示されます。

有効値

true()、false()、もしくは、true() か false()を返す有効な式。

例

次の例はフィールド#STD_FLAGが 'X'と等しい場合にウェブレットを非表示にします。式は入力されなくてははいけません。またプロパティにフォーカスがある時、以下のように表示されます。

| | |
|---------|-----------------|
| hide_if | #STD_FLAG = 'X' |
|---------|-----------------|

プロパティがフォーカスを失うと、この式は以下のように表示されます。

| | |
|---------|--------------------------------------|
| hide_if | key('field-value', 'STD_FLAG') = 'X' |
|---------|--------------------------------------|

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

'left:0pt;top:0pt;' - ウェブレットの位置は相対的に位置付けられます。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

button_width

プロンプター・ボタンの幅。

省略値

空白。ウェブレットは省略値の幅を採用します。

有効値

単一引用符で囲まれた有効な範囲内の幅。

button_height

プロンプター・ボタンの高さ。

省略値

空白。ボタンは省略値の高さを採用します。

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

prompter_width

プロンプター・ボタンがクリックされたときのプロンプトの最初の幅。 *auto_resize*がtrueに設定されている場合、コンテンツがロードされると、プロンプターは自動的にサイズを調整します。

省略値

ブランク。プロンプター・ウィンドウは省略値の幅を採用します。

有効値

単一引用符で囲まれた有効な範囲内の幅。

prompter_height

プロンプター・ボタンがクリックされたときのプロンプト・ウィンドウの最初の高さ。 *auto_resize*がtrueに設定されている場合、コンテンツがロードされると、プロンプターは自動的にサイズを調整します。

省略値

空白。プロンプター・ウィンドウは省略値の高さを採用します。

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

auto_resize

プロンプターが自動的にコンテンツのサイズにサイズ変更するかどうかを決定します。

省略値

true

有効値

trueまたはfalse。trueの場合、プロンプターは自動的にその中に表示されているページのサイズに合うようにサイズを変更します。falseの場合、プロンプターは*prompter_width*と*prompter_height*プロパティに従ってサイズ調整されます。

使用されるサイズはブラウザによって決定され、ブラウザごとにわずかに異なります。また、*prompter_width*と*prompter_height*プロパティに初期設定されたサイズによっても影響を受けます。一番よい結果を得るためには、プロンプターのコンテンツ・ページのサイズを明確にして、ブラウザにより決定される必要がないようにします。多くのブラウザは必要に応じてプロンプターを拡大しますが、縮小はしないということに注意してください。ブラウザによっては縦方向に縮小するものもありますが、横方向には縮小されません。

button_class

プロンプター・ボタンのカスケード・スタイル・シートのクラス名。

省略値

'std_prompter_button'.プロンプター・ボタンの省略値クラスで、全てのカスケード・スタイル・シートと共に提供されます。

有効値

現在のカスケード・スタイル・シートから選択された有効なクラス名を単一引用符で囲んだもの。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

prompter_class

プロンプター・ボタンがクリックされた時に表示されるプロンプト・ウィンドウのカスケード・スタイル・シートのクラス。

省略値

'std_prompter'プロンプター・ボタンの省略値クラスで、全てのカスケード・スタイル・シートと共に提供されています。

有効値

現在のカスケード・スタイル・シートから選択された有効なクラス名を単一引用符で囲んだもの。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

button_mouseover_class

マウスがその上に置かれた時の、プロンプター・ボタンのカスケード・スタイル・シートのクラス名。

省略値

'std_prompter_button_mouseover'プロンプター・ボタンの省略値の mouseover クラスで、全てのカスケード・スタイル・シートと共に提供されています。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

formname

サーバーに送られるHTMLフォーム名です。

省略値

'LANSA'

有効値

単一引用符付きのフォーム名。プロパティ・シート内で該当するドロップダウン・ボタンをクリックすると、使用できるフォーム名リストが表示されます。

prompter_url

プロンプター・パネルの移動先URL。prompter_wnameプロパティが指定されている場合、無視されます。

省略値

空白 - プロンプターはURLに移動しません。

有効値

単一引用符で囲まれた有効なURL。プロトコル(例：http:// や https://)を指定しなければいけないことに注意してください。

URLはWAMと同じサーバーからのものでなければいけません。ブラウザのセキュリティ制限により、あるサーバーからのページのプログラムは、他のサーバーからのページの情報にアクセスできないようになっています。他のサーバーを指すURLを使用すると、プロンプターのコードは"アクセス拒否" エラーで失敗します。

prompter_wamname

選択を行うプロンプター・ウィンドウにページを表示するために呼び出されるWAMの名前。prompter_urlプロパティが指定されている場合は、ブランクのままにしてください。

省略値

\$lweb_WAMName (現在のWAMを示します。)

有効値

単一引用符で囲まれた有効なWAM名。プロパティ・シートの該当するドロップダウン・ボタンをクリックして、使用できるWAMのリストから選択できます。

prompter_wrname

選択を行うプロンプター・ウィンドウにページを表示するために呼び出されるWebroutineの名前。prompter_urlプロパティが指定されている場合は、ブランクのままにしてください。

省略値

prompter_wrnameプロパティが指定されている場合、Webroutine名を必ず指定する必要があります。

有効値

単一引用符で囲まれた有効なWebroutine名。プロパティ・シートの該当するドロップダウン・ボタンをクリックして、使用できるWebroutineのリストから選択できます。

protocol

このウェブレットに呼び出されたWebroutineへの移動に使用するプロトコル(例えば、http:// もしくは https://)

省略値

空白。現在使用されているプロトコルが使用されます。

有効値

単一引用符付きの、有効なプロトコル。通常は'http:'または'https:'です。

field_name_to_exchange

プロンプターの結果を入れる、ローカルのWebroutine(プロンプターを持つWebroutine)内のフィールドの名前。この値は「終了Webroutine」内の同じ名前のフィールドから持ってこられます。異なる名前の複数のフィールドをコピーするには、[field_mapping](#)プロパティを使用します。

省略値

空白。フィールド名の指定は必須です。

有効値

単一引用符で囲まれた有効なフィールド名。プロパティ・シートで該当するドロップダウン・ボタンをクリックし、使用可能なフィールドのリストから選択してください。

closing_url

プロンプター・ウィンドウを閉じるURL。closing_wnameプロパティが指定されている場合、無視されます。

省略値

空白。プロンプター・ウィンドウを閉じるのにURLは呼び出されません。

有効値

単一引用符で囲まれた有効なURL。プロトコル(例：http:// や https://)を指定しなければいけないことに注意してください。

URLはWAMと同じサーバーからのものでなければいけません。ブラウザのセキュリティ制限により、あるサーバーからのページのプログラムは、他のサーバーからのページの情報にアクセスできないようになっています。他のサーバーを指すURLを使用すると、プロンプターのコードは"アクセス拒否" エラーで失敗します。

field_mapping

このプロパティを使用して、[終了Webroutine](#)内の複数のフィールドのマッピングをローカルWebroutine(プロンプターを含むWebroutine)に定義します。このプロパティは直接は編集できません。プロパティ・シートの省略記号(...)のボタンをクリックすると、カスタム・プロパティ・エディターのウィンドウが開きます。

終了Webroutineから1つのフィールドだけコピーしたい場合は、[field_name_to_exchange](#) プロパティを代わりに使用することができます。1つもしくは複数のフィールドをコピーする場合で全てが一致する名前(FIELDAはFIELDAにマップ、FIELDBはFIELDBにマップするなど)であれば、[field_mapping](#)と[field_name_to_exchange](#)を変更することなく、そのままにしておいて構いません。プロンプターは自動的に終了Webroutine内の全フィールドを、ローカルWebroutineの一致するフィールド(存在する場合)にコピーします。

[field_name_to_exchange](#)が定義されている場合、[field_mapping](#)は無視されることに注意してください。

省略値

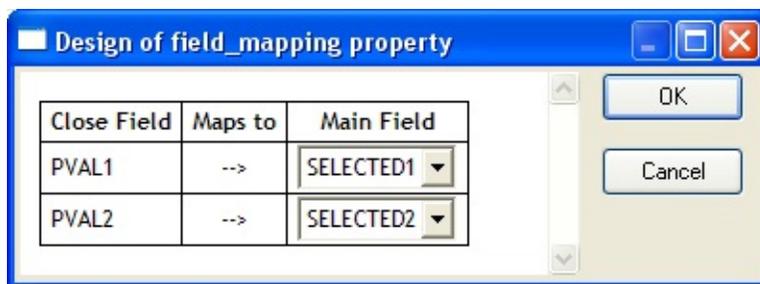
次のXPath式は、XSLソースに格納された、マッピング情報を含むXMLデータ・フラグメントを識別します。

```
document(")/*/lxml:data/lxml:prompter[@id='<unique id>']
```

有効値

適用不可。(この値はシステムにより保守されます。)プロパティ・シートの省略記号(...)のボタンを使用して、フィールドのマッピングを編集します。

次の例では、ローカルWebroutineが'Main'、プロンプターの終了Webroutineが'Close'です。Closeのフィールド#PVAL1及び#PVAL2はMainのフィールド#SELECTED1及び#SELECTED2にコピーされます。



closing_wrname

現在のWAMにある、プロンプター・パネルを閉じる為に呼び出されるWebroutine名。closing_urlプロパティが指定されている場合、ブランクのままにしてください。

このWebroutineがロードされると、プロンプターはすぐに閉じられます。ただし、Webroutineのデザインがほんの短い間表示される可能性があるため、コンテンツが非表示のブランクのレイアウトを使用してください。

プロンプターの出力値は、プロンプターが探しだしてその値を転送できるように、WebroutineのHTML出力に<input>エレメントとして、全て存在する必要があります。これを非表示のままで行う一番早い方法は、HTML生成前にWebマップで*HIDDENとして定義します。

省略値

'Close' - このWebroutine名は単なる提案です。

有効値

現在のWAM内の有効なWebroutine名。プロパティ・シートで該当するドロップダウン・ボタンをクリックして、使用可能なWebroutineリストから選択します。

reentryfield

reentryvalue プロパティに指定された値をWAMに送る時に使用するフィールド名です。フィールド名は単一引用符で囲みます。

省略値

'STDREENTRY'

有効値

リポジトリもしくはWAMで定義されたフィールド名。プロパティシート内で該当するドロップダウン・ボタンをクリックすると、使用できるフィールド名リストが表示されます。

reentryvalue

reentryfieldプロパティに指定されたフィールドに送られる値。英数字フィールドの場合、値は単一引用符付きで指定する必要があります。数字フィールドの場合は、単一引用符はなくても構いません。

省略値

'M'

有効値

適切なリテラル。

disabled

ウェブレットを使用可または不可として表示するかを決定するブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false() もしくは有効な式。

title

ヒントのテキスト。

省略値

ブランク - ヒントは何も表示されません。

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

on_change_wamname

プロンプターで選択された後、呼び出されるWAMの名前。通常このプロパティは、プロンプターによって返された値が現在のWAM以外のWAMに送られる場合に使用します。

省略値

\$lweb_WAMName (現在のWAMを示します。)

有効値

単一引用符付きのWAM名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

on_change_wrname

プロンプターで選択された後、呼び出されるWebroutineの名前。

省略値

空白 - プロンプターで選択された後、Webroutineは呼び出されません。

有効値

単一引用符付きのWebroutine名。このWebroutineは、on_change_wamnameプロパティに指定されたWAMに存在していなくてはなりません。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWebroutineのリストが表示されます。

on_change_protocol

プロンプターで選択がされた後に、ウェブレットがon_change_wamnameプロパティに指定されたWAMに移動するために使用するプロトコル(例えば、http:もしくはhttps:)

省略値

ブランク - 現在のプロトコルが使用されます。

有効値

単一引用符付きの、有効なプロトコル。通常は'http:'または'https:'です。

on_change_reentryfield

プロンプターで選択された後にon_change_reentryvalueプロパティに指定された値を送るのに使用されるフィールド名。このフィールドはon_change_wnameプロパティが指定されている場合のみ使用されます。

省略値

ブランク - 値を送るためにフィールドが使用されません。

有効値

単一引用符で囲まれた有効なフィールド名。プロパティ・シートの対応するドロップダウン・ボタンをクリックして、既知のフィールドのリストから選択してください。

on_change_reentryvalue

on_change_reentryfieldプロパティで指定されたフィールドに送られる値。この値はon_change_wnameプロパティが指定されている場合のみ使用されます。

省略値

ブランク - 値は送られません。

有効値

on_change_reentryfieldプロパティに指定されたフィールドのタイプに有効な値。

on_change_target_window_name

on_change_wnameプロパティに指定されたWebroutineが呼び出された時に、コンテンツが送られるウィンドウの名前。

省略値

空白 - コンテンツは現在のウィンドウに送られます。

有効値

単一引用符で囲まれたウィンドウ名またはフレーム名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるウィンドウやフレームが表示されます。

'_blank'は新しいウィンドウに表示します。

'_media'は現在のウィンドウのメディア・パネルに表示します。

'_search'は現在のウィンドウの検索パネルに表示します。

'_parent'は親ウィンドウ（多くの場合、現ウィンドウ）に表示します。

'_top'は一番上のウィンドウ（多くの場合、現ウィンドウ）に表示します。

_search や _media はInternet Explorer 6 でのみサポートされています。

pre_show_js

プロンプターを表示する前に実行されるJavaScriptコード。

省略値

空白。JavaScriptは何も実行されません。

有効値

有効なJavaScriptファンクション、もしくはセミコロン(;)やダブル・バックスラッシュ(//)が後ろに付いたJavaScriptコード。JavaScriptのセクションの後にセミコロンが付く場合、スクリプトが既存のJavaScriptに続くことを示しています。これに対してダブル・バックスラッシュが付く場合は、WAMのために実行されたJavaScriptの残りの部分はコメントとして扱われ、無効になることを示しています。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

vf_wamevent

VLF のWAM イベント文字列です。

省略値

空白。

有効値

文字列。カンマ (,) は使用できません。

8.1.24 ラジオ・ボタン (std_rad_button)

クイック・スタート - ラジオ・グループ プロパティ - ラジオ・グループ

ラジオ・ボタン・ウェブレットは、1つのラジオ・ボタンを表示します。これは、`<input type="radio">` HTMLエレメントと概ね同じです。

通常ラジオ・ボタンは、2つ以上のラジオ・ボタンがグループ化され、排他的な選択を行います。つまり択一式の問題のように、複数の選択肢から1つのみ選ぶことが可能です。例えば次のようなものです。



また、通常は以下のようにグループ化されます。



異なるフィールドに選択結果を保管したい場合、ラジオ・ボタンをグループとして組込むstd_radbuttonsウェブレットの代わりにこのウェブレットを使用できます。

ラジオ・ボタン・ウェブレットに、クリック時に別の Webroutine に移動できる on_click_wname などのプロパティが含まれる場合、ラジオ・ボタンのクリックからアクションを開始させるのは、あまり良いユーザー・インターフェース・デザインとは言えません。これを行うには、プッシュ・ボタンやメニュー・アイテム、アンカー(ハイパーリンク)などのデバイスを使用してください。

クイック・スタート - ラジオ・ボタン

ラジオボタンは常に2つ以上のラジオボタンのグループで表示する必要があります。Webページに追加した各ラジオボタンは通常、WebroutineのWEB_MAPに含まれるフィールドに対応しています。LANSAエディターでWebroutine用に生成されたXSLを開き、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[ラジオ ボタン]ウェブレットを見つけてください。
2. ラジオ・ボタン・ウェブレットをWEB_MAP定義により生成されたフィールドの上にドラッグしてください。(フィールド記述は削除しても構いません。)これにより省略値のラジオボタンの設定が表示されます。

 Caption

3. ウェブレットをクリックして、[詳細]タブを確認します。nameプロパティとvalueプロパティにウェブレットをドラッグしたフィールドを示す値が設定されていることを確認してください。valueプロパティは、Webページ表示の際にこのフィールドに現在入っている値がcodeプロパティと比較され、このラジオボタンがチェックされるべきか決定されることを示しています。ラジオボタンの値が変更されると、適切な値がnameプロパティに指定されたフィールドに入れられます。この場合、同じフィールドになります。
4. captionプロパティを必要に応じて設定し、codeプロパティがcaptionに必要な値と一致していることを確認します。

プロパティ - ラジオ・ボタン

ラジオ・ボタン・ウェブレットのプロパティは以下のとおりです。

| | | |
|-----------|------------------|--------------------|
| alignment | label_id | reentryfield |
| caption | mouseover_class | reentryvalue |
| class | name | tab_index |
| code | on_click_wamname | target_window_name |
| disabled | on_click_wrname | text_class |
| formname | pos_absolute | value |
| hide_if | protocol | vf_wamevent |

name

ラジオボタンの名前。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。このウェブレットがフィールド上にドロップされた場合、またはフィールドを表示したり、フィールドに値を追加するために使用される場合は、フィールド名が使用されます。

省略値

自動的に生成された、一意の識別子。

有効値

単一引用符で囲まれた文字列。

value

ウェブレットに設定する値。このウェブレットによりフィールドがビジュアライズされた場合、これがフィールド値もしくは省略値になります。

省略値

ブランク。

有効値

単一引用符で囲まれたテキスト、もしくは複数言語対応テキスト変数、システム変数、フィールドの名前(プロパティ・シートで省略記号のボタンをクリックしてリストから選ぶことができます。)

例

次の例では、valueはフィールド#SECTIONの現在の値に設定されることを示しています。プロパティに入力すると、以下ようになります。



A screenshot of a property editor. The left side shows a blue header with a green icon and the text 'value'. The right side shows a white header with a blue icon and the text '#SECTION'.

フォーカスがこのプロパティから移った時は、同じvalueプロパティが以下のように表示されます。



A screenshot of a property editor. The left side shows a blue header with a green icon and the text 'value'. The right side shows a white header with a blue icon and the text 'key('field-value', 'SECTION')'.

caption

ラジオボタンの隣に表示するテキスト。

省略値

ブランク。

有効値

単一引用符で囲まれたテキスト、もしくは複数言語対応テキスト変数、システム変数、フィールドの名前(プロパティ・シートで省略記号のボタンをクリックしてリストから選ぶことができます。)

code

ラジオボタンがチェックされたときにフィールドに入れられる値。

省略値

空白。

有効値

単一引用符で囲まれた文字列。

label_id

ラジオボタンのキャプションとボタンを結びつける、ラジオボタン・エレメントの識別子。これにより、キャプションがクリックされた場合、ボタン自体がクリックされたのと同じように応答します。この値は一意でなければいけません。

省略値

`concat($name,$code)` (nameとcodeプロパティの値が結合されます。)

有効値

単一引用符で囲まれた文字列。

reentryfield

reentryvalue プロパティに指定された値をWAMに送る時に使用するフィールド名です。フィールド名は単一引用符で囲みます。

省略値

'STDREENTRY'

有効値

リポジトリもしくはWAMで定義されたフィールド名。プロパティシート内で該当するドロップダウン・ボタンをクリックすると、使用できるフィールド名リストが表示されます。

reentryvalue

reentryfieldプロパティに指定されたフィールドに送られる値。英数字フィールドの場合、値は単一引用符付きで指定する必要があります。数字フィールドの場合は、単一引用符はなくても構いません。

省略値

'M'

有効値

適切なリテラル。

hide_if

ブール値プロパティです。評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (グリッドは常に表示されます。)

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

例

次の例はフィールド#STD_FLAGが'X'と等しい場合にグリッドを非表示にします。式は入力されなくてはなりません。またプロパティにフォーカスがある時、以下のように表示されます。

| | |
|---------|-----------------|
| hide_if | #STD_FLAG = 'X' |
|---------|-----------------|

プロパティがフォーカスを失うと、この式は以下のように表示されません。

| | |
|---------|--------------------------------------|
| hide_if | key('field-value', 'STD_FLAG') = 'X' |
|---------|--------------------------------------|

formname

サーバーに送られるHTMLフォーム名です。

省略値

'LANSA' (document.LANSAのことです。)

有効値

単一引用符付きのフォーム名。プロパティ・シート内で該当するドロップダウン・ボタンをクリックすると、使用できるフォーム名リストが表示されます。

on_click_wamname

ウェブレット内の項目が選択された時に呼び出されるWAM名。

注：ラジオ・ボタンのクリックでアクションを起動するのはあまり良いユーザー・インターフェースとは言えません。このような場合はプッシュボタンやメニュー・アイテム、アンカー(ハイパーリンク)のようなデバイスを使用してください。

省略値

\$lweb_WAMName (現在のWAMを示します。)

有効値

単一引用符付きのWAM名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

on_click_wname

ウェブレット内の項目が選択された時に呼び出されるWebroutine名。

注：ラジオ・ボタンのクリックでアクションを起動するのはあまり良いユーザー・インターフェースとは言えません。このような場合はプッシュボタンやメニュー・アイテム、アンカー(ハイパーリンク)のようなデバイスを使用してください。

省略値

空白。

有効値

単一引用符付きのWebroutine名。Webroutineは、on_click_wamnameプロパティに指定されたWAMに存在してはいけません。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWebroutineのリストが表示されます。

protocol

このウェブレットに呼び出されたWebroutineへの移動に使用するプロトコル(例えば、http:// もしくは https://)

省略値

空白。現在使用されているプロトコルが使用されます。

有効値

単一引用符付きの、有効なプロトコル。通常は'http:'または'https:'です。

target_window_name

応答HTMLが表示される、ウィンドウやフレームの名前。一意の名前を入力するか、事前定義された値から選択してください。

省略値

空白 - 応答HTMLが現在のウィンドウに表示されます。

有効値

単一引用符で囲まれたウィンドウ名またはフレーム名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用可能なウィンドウやフレームが表示されます。または一意の名前を入力することもできます。

'_blank'は新しいウィンドウに表示します。

'_media'は現在のウィンドウのメディア・パネルに表示します。

'_search'は現在のウィンドウの検索パネルに表示します。

'_parent'は親ウィンドウ（多くの場合、現ウィンドウ）に表示します。

'_top'は一番上のウィンドウ（多くの場合、現ウィンドウ）に表示します。

_search や _media はInternet Explorer 6 でのみサポートされています。

alignment

キャプションがラジオボタンの左に表示されるか右に表示されるかを示します。

省略値

'right'

有効値

'left'(左)または'right'(右)

disabled

ウェブレットを使用可または不可として表示するかを決定するブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false() もしくは有効な式。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

class

ウェブレットに適用されるカスケード・スタイル・シートクラス。

省略値

ウェブレット用に提供されている省略値のクラス名。

有効値

単一引用符で囲まれた、現在のカスケード・スタイル・シートの有効なクラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。出荷時のクラス'std_rad_button'が提供されています。

mouseover_class

マウスがその上に置かれた時にウェブレットに適用されるカスケード・スタイル・シートのクラス名。

省略値

空白。

有効値

単一引用符で囲まれた、現在のカスケード・スタイル・シートの有効なクラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。出荷時のクラス'`std_rad_button_mouseover`'が提供されています。

text_class

ウェブレットのラジオボタンのキャプションに適用されるカスケード・スタイル・シートクラス。

省略値

ウェブレット用に提供されている省略値のクラス名。

有効値

単一引用符で囲まれた、現在のカスケード・スタイル・シートの有効なクラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。出荷時のクラス'`std_rad_button_text`'が提供されています。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

vf_wamevent

VLF のWAM イベント文字列です。

省略値

空白。

有効値

文字列。カンマ (,) は使用できません。

8.1.25 ラジオ・グループ (std_radbuttons)

クイック・スタート - ラジオ・ボタン プロパティ - ラジオ・ボタン

ラジオ・グループ・ウェブレットは、ラジオ・ボタンのグループを作成します。ボタンの数とボタンの値は、作業リストの値から、またはウェブレットのitemプロパティに定義された静的な値からも生成できます。各ラジオ・ボタンは<input type="radio"> HTMLエレメントと概ね同じです。例えば次のようなものです。



The image shows a web form element titled "Select a Color" in blue text. Below the title are four radio buttons, each with a colored circle and a label: "Red" (light blue circle), "Yellow" (yellow circle), "Green" (light blue circle), and "Blue" (yellow circle). The "Yellow" radio button is selected, indicated by a dark green dot in the center of its circle.

ラジオ・グループ・ウェブレットに、クリック時に他の Webroutineに移動できる、on_click_wnameなどのプロパティが含まれる場合、ラジオ・ボタンのクリックでアクションを開始するのは、あまり良いユーザー・インターフェース・デザインとは言えません。この場合、プッシュ・ボタンやメニュー・アイテム、アンカー(ハイパーリンク)などのデバイスを使用すべきです。

クイック・スタート - ラジオ・グループ

ラジオ・グループの各エントリーは、作業リストのエントリー、またはラジオ・グループのプロパティでハードコーディングされた一連のアイテムによって定義されています。

作業リストを使用する場合

グループのラジオ・ボタンを定義するために作業リストを使用する場合、選択された値を格納するフィールドと選択肢オプション用の作業リストをWEB_MAPに定義したWebroutineを作成する必要があります。LANSAエディターでWebroutine用に生成されたXSLを開き、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[ラジオ グループ]ウェブレットを見つけてください。
2. ラジオ・グループ・ウェブレットを、値を格納するフィールド上にドラッグし、マウスの左ボタンを離してください。(フィールド記述は削除しても構いません。)これにより省略値のラジオボタンの設定が表示されます。

Radio 1 Radio 2

3. ウェブレットをクリックして、[詳細]タブを確認します。nameプロパティとvalueプロパティにウェブレットをドラッグしたフィールドを示す値が設定されていることを確認してください。valueプロパティは、このフィールドに現在入っている値を使って、Webページ表示の際に選択された状態のラジオボタンが設定されることを示しています。ラジオボタンの値が変更されると、適切な値がnameプロパティに指定されたフィールドに入れられます。この場合、同じフィールドになります。

listnameプロパティをWEB_MAPに送る作業リストに変更します。ラジオ・グループの表示はすぐに、識別子のない一連のボタンに変更されず。

codefieldプロパティとcaptionfieldプロパティに作業リストの適切なフィールドを設定します。

itemsプロパティを使用する場合

radio groupプロパティにハードコーディングされた一連のアイテムを使用するには、WEB_MAPにフィールドを指定したWebroutineを作成する

必要があります。LANSAエディターでWebroutine用に生成されたXSLを開き、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[ラジオ グループ]ウェブレットを見つけてください。
2. ラジオ・グループ・ウェブレットを、値を格納するフィールド上にドラッグし、マウスの左ボタンを離してください。(フィールド記述は削除しても構いません。)これにより省略値のラジオボタンの設定が表示されます。

Radio 1 Radio 2

3. ウェブレットをクリックして、[詳細]タブを確認します。nameプロパティとvalueプロパティにウェブレットをドラッグしたフィールドを示す値が設定されていることを確認してください。valueプロパティは、このフィールドに現在入っている値を使って、Webページ表示の際に選択された状態のラジオボタンが設定されることを示しています。ラジオボタンの値が変更されると、適切な値がnameプロパティに指定されたフィールドに入れられます。この場合、同じフィールドになります。
4. itemsプロパティの省略記号(...)のボタンを選択して、ラジオボタンとして使用するアイテムのリストを設定します。引き続きラジオボタンに必要なエントリーの定義を行います。

プロパティ - ラジオ・グループ

ラジオ・グループ・ウェブレットのプロパティは以下のとおりです。

| | | |
|----------------|------------------|--------------------|
| alignment | hide_if | reentryfield |
| captionfield | items | reentryvalue |
| class | listname | selector_field |
| codefield | mouseover_class | selector_value_eq |
| disabled | name | show_groupbox |
| display_mode | on_click_wamname | tab_index |
| formname | on_click_wname | target_window_name |
| group_title | orientation | text_class |
| groupbox_class | pos_absolute | width |
| height | protocol | value |
| | | vf_wamevent |

name

ラジオグループの名前。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。このウェブレットがフィールド上にドロップされた場合、またはフィールドを表示したり、フィールドに値を追加するために使用される場合は、フィールド名が使用されます。

省略値

自動的に生成された、一意の識別子。

有効値

単一引用符で囲まれた文字列。

value

ウェブレットに設定する値。このウェブレットによりフィールドがビジュアライズされた場合、これがフィールド値もしくは省略値になります。

省略値

ブランク。

有効値

単一引用符で囲まれたテキスト、もしくは複数言語対応テキスト変数、システム変数、フィールドの名前(プロパティ・シートで省略記号のボタンをクリックしてリストから選ぶことができます。)

例

次の例では、valueはフィールド#SECTIONの現在の値に設定されることを示しています。プロパティに入力されると、以下のようになります。



A screenshot of a property editor. The left side shows a blue header with a green icon and the text 'value'. The right side shows a white header with a blue icon and the text '#SECTION'.

フォーカスがこのプロパティから移った時は、同じvalueプロパティが以下のように表示されます。



A screenshot of a property editor. The left side shows a blue header with a green icon and the text 'value'. The right side shows a white header with a blue icon and the text 'key('field-value', 'SECTION')'.

display_mode

ウェブレットが入力内容を受け取るのか、出力内容を表示するのかを制御します。

省略値

'input'

有効値

'input' または 'output'

items

ウェブレットに表示する項目を指定するXMLノード・セット。設定はデザイナーでのみ可能です。デザイナーを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使用してください。アイテムがlistnameプロパティに指定されたリストから構成される場合、空白にしてください。

省略値

document(")/*/lxml:data/lxml:dropdown (このドロップダウンに項目が何も定義されていないことを示します。)

有効値

適用不可。(この値はシステムにより保守されます。)デザイナーを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使用してください。

例

次の例は項目がデザイナーで設定され、ドロップダウンの値として使用されることを示しています。

items document(')/*/lxml:data/lxml:caption_value_pairs[@id='4A14173FA6D64A059507F9124A630CF3']

プロパティの省略記号(...)のボタンを使用すると、デザイナーが表示され、ラジオボタンとして表示されるアイテムを保守できます。以下のデザイナーのビューは、ラジオ・グループに2つのラジオボタンが必要であることを示しています。最初のエントリーはリテラル値'MONDAY'、2つめのエントリーは複数言語変数を使用して、コードTUEの記述を表示します。値が事前に選択されていない時に選択された状態にする項目には[規定のアイテム]のチェックボックスを使用します。



listname

ラジオ・グループの中にラジオボタンを作成するのに使用する作業リストの名前。itemsプロパティに詳細が指定されている場合は、ブランクのままにしてください。

省略値

ブランク。

有効値

単一引用符で囲まれた文字列。使用可能な（WAMで定義された）作業リストの一覧は、プロパティシートで該当するドロップダウン・ボタンをクリックすることで表示され、選択することができます。

selector_field

listnameプロパティに指定されているリスト内のフィールドで、ウェブレットに表示されたリスト項目を制限したり、部分集合にしたりする値を含むフィールドの名前。このプロパティはselector_value_eqプロパティと共に使用されます。

省略値

空白。

有効値

単一引用符で囲まれた文字列。プロパティ・シートで該当するドロップダウン・ボタンをクリックすると、listnameに指定された作業リストからのフィールドを選択できます。

selector_value_eq

この値を使って、ウェブレットに表示されるリスト項目を制限したり、部分集合にしたりします。listnameプロパティが指定されている場合、selector_fieldプロパティに関係するフィールドを指定する必要があります。itemsプロパティ・デザイナーが値のリストを定義するのに使用されている場合、デザイナーで入力された対応するセレクター値が使用されません。

省略値

空白。

有効値

単一引用符で囲まれたテキスト、もしくは数値。プロパティ・シートで該当するドロップダウン・ボタンをクリックすると、listnameに指定された作業リストからのフィールドを選択できます。

codefield

各リスト項目のキー値を保持するlistnameプロパティに指定されたリストのフィールド名。

省略値

空白。

有効値

単一引用符で囲まれた文字列。プロパティ・シートで該当するドロップダウン・ボタンをクリックすると、listnameに指定された作業リストからフィールドを選択できます。

captionfield

各リスト項目のキャプションを保持するlistnameプロパティに指定されたリストのフィールド名。

省略値

空白。

有効値

単一引用符で囲まれた文字列。プロパティ・シートで該当するドロップダウン・ボタンをクリックすると、listnameに指定された作業リストからフィールドを選択できます。

reentryfield

reentryvalue プロパティに指定された値をWAMに送る時に使用するフィールド名です。フィールド名は単一引用符で囲みます。

省略値

'STDREENTRY'

有効値

リポジトリもしくはWAMで定義されたフィールド名。プロパティシート内で該当するドロップダウン・ボタンをクリックすると、使用できるフィールド名リストが表示されます。

reentryvalue

reentryfieldプロパティに指定されたフィールドに送られる値。英数字フィールドの場合、値は単一引用符付きで指定する必要があります。数字フィールドの場合は、単一引用符はなくても構いません。

省略値

'M'

有効値

適切なリテラル。

hide_if

ブール値プロパティです。評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (グリッドは常に表示されます。)

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

例

次の例はフィールド#STD_FLAGが'X'と等しい場合にグリッドを非表示にします。式は入力されなくてはなりません。またプロパティにフォーカスがある時、以下のように表示されます。

| | |
|---------|-----------------|
| hide_if | #STD_FLAG = 'X' |
|---------|-----------------|

プロパティがフォーカスを失うと、この式は以下のように表示されません。

| | |
|---------|--------------------------------------|
| hide_if | key('field-value', 'STD_FLAG') = 'X' |
|---------|--------------------------------------|

formname

サーバーに送られるHTMLフォーム名です。

省略値

'LANSA' (document.LANSAのことです。)

有効値

単一引用符付きのフォーム名。プロパティ・シート内で該当するドロップダウン・ボタンをクリックすると、使用できるフォーム名リストが表示されます。

on_click_wamname

ウェブレット内の項目が選択された時に呼び出されるWAM名。

注：ラジオ・ボタンのクリックでアクションを起動するのはあまり良いユーザー・インターフェースとは言えません。このような場合はプッシュボタンやメニュー・アイテム、アンカー(ハイパーリンク)のようなデバイスを使用してください。

省略値

\$lweb_WAMName (現在のWAMを示します。)

有効値

単一引用符付きのWAM名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

on_click_wname

ウェブレット内の項目が選択された時に呼び出されるWebroutine名。

注：ラジオ・ボタンのクリックでアクションを起動するのはあまり良いユーザー・インターフェースとは言えません。このような場合はプッシュボタンやメニュー・アイテム、アンカー(ハイパーリンク)のようなデバイスを使用してください。

省略値

空白。

有効値

単一引用符付きのWebroutine名。Webroutineは、on_click_wamnameプロパティに指定されたWAMに存在してはいけません。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWebroutineのリストが表示されます。

protocol

このウェブレットに呼び出されたWebroutineへの移動に使用するプロトコル(例えば、http:// もしくは https://)

省略値

空白。現在使用されているプロトコルが使用されます。

有効値

単一引用符付きの、有効なプロトコル。通常は'http:'または'https:'です。

target_window_name

応答HTMLが表示される、ウィンドウやフレームの名前。一意の名前を入力するか、事前定義された値から選択してください。

省略値

空白 - 応答HTMLが現在のウィンドウに表示されます。

有効値

単一引用符で囲まれたウィンドウ名またはフレーム名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用可能なウィンドウやフレームが表示されます。または一意の名前を入力することもできます。

'_blank'は新しいウィンドウに表示します。

'_media'は現在のウィンドウのメディア・パネルに表示します。

'_search'は現在のウィンドウの検索パネルに表示します。

'_parent'は親ウィンドウ（多くの場合、現ウィンドウ）に表示します。

'_top'は一番上のウィンドウ（多くの場合、現ウィンドウ）に表示します。

_search や _media はInternet Explorer 6 でのみサポートされています。

alignment

キャプションがラジオボタンの左に表示されるか右に表示されるかを示します。

注意：横方向の時にleft(左)の整列を使用した場合、ラジオボタンのキャプションの最初の文字を基本にベースに揃えられるため、ラジオボタンがうまく整列されない場合があります。

省略値

'right'

有効値

'left'(左)または'right'(右)

orientation

ラジオボタンのグループの方向を示します。

省略値

'horizontal'(横方向)

有効値

'vertical'(縦方向)または'horizontal'(横方向)

show_groupbox

ラジオボタン・グループの周りにグループ・ボックスを表示するかどうかを決定するブール値プロパティです。

省略値

false() (グループ・ボックスは表示されません。)

有効値

true()、false() もしくは有効な式。

group_title

グループ・ボックスに表示されるタイトル。show_groupboxプロパティがセットされている場合、タイトルはグループ・ボックスの左上に表示されます。グループ・ボックスが表示されていない場合は、タイトルはラジオボタンのグループの中央上に表示されます。

省略値

空白。

有効値

単一引用符で囲まれたテキスト、もしくは複数言語対応テキスト変数、システム変数、フィールドの名前(プロパティ・シートで省略記号のボタンをクリックしてリストから選ぶことができます。)

disabled

ウェブレットを使用可または不可として表示するかを決定するブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false() もしくは有効な式。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

Webページのウェブレットの幅です。

省略値

blank (ウェブレットは省略値の幅を使用します。)

有効値

単一引用符で囲まれた有効な範囲内の幅。

height

Webページのウェブレットの高さ。

省略値

blank(ウェブレットは省略値の高さを使用します。)

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

class

ウェブレットに適用されるカスケード・スタイル・シートクラス。

省略値

ウェブレット用に提供されている省略値のクラス名。

有効値

単一引用符で囲まれた、現在のカスケード・スタイル・シートの有効なクラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。出荷時のクラス'`std_rad_button`'が提供されています。

mouseover_class

マウスがその上に置かれた時にウェブレットに適用されるカスケード・スタイル・シートのクラス名。

省略値

空白。

有効値

単一引用符で囲まれた、現在のカスケード・スタイル・シートの有効なクラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。出荷時のクラス'`std_rad_button_mouseover`'が提供されています。

text_class

ウェブレットのラジオボタンのキャプションに適用されるカスケード・スタイル・シートクラス。

省略値

ウェブレット用に提供されている省略値のクラス名。

有効値

単一引用符で囲まれた、現在のカスケード・スタイル・シートの有効なクラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。出荷時のクラス'`std_rad_button_text`'が提供されています。

groupbox_class

ウェブレットに関連付けられたグループ・ボックスに適用されるカスケード・スタイル・シートクラス。このプロパティはshow_groupboxプロパティにtrueが設定されている時のみ適用されます。

省略値

空白。

有効値

単一引用符で囲まれた、現在のカスケード・スタイル・シートの有効なクラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

例

```
groupbox_class 'std_listbox'
```

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

vf_wamevent

VLF のWAM イベント文字列です。

省略値

空白。

有効値

文字列。カンマ (,) は使用できません。

8.1.26 水平スプリッター (std_splitter_horz)

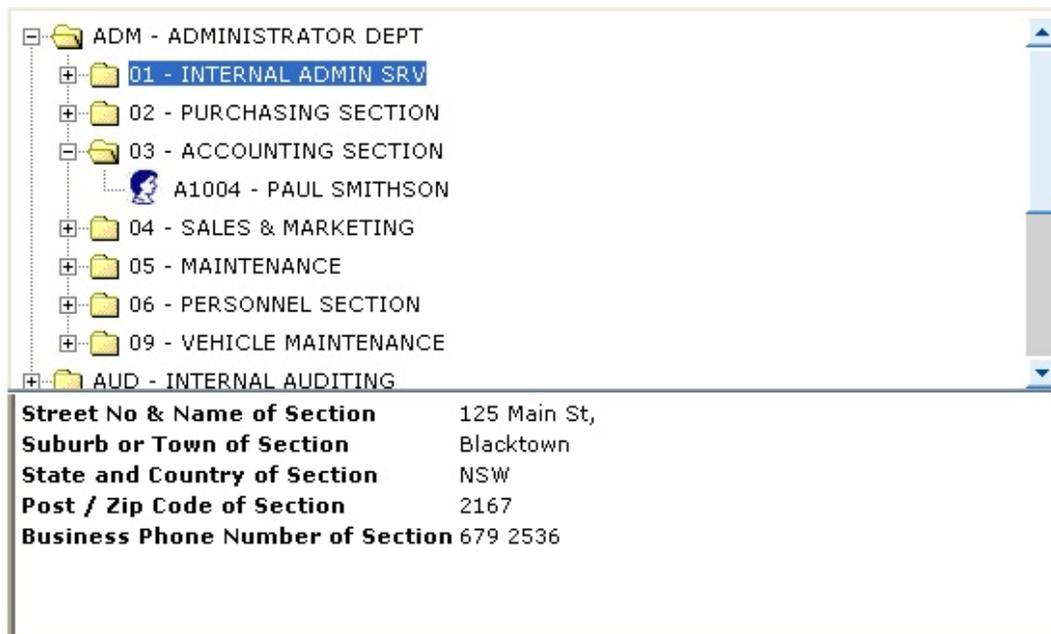
クイック・スタート - 水平スプリッター

プロパティ - 水平スプリッター

水平スプリッターウェブレットは、このウェブレットがカバーするエリアを移動可能な仕切りで上下二つのペインに分けます。入出力は提供せず、Webページ・デザインをサポートするためのみに使用されます。他のエレメントやウェブレットをそれぞれのペインにドロップすることができ、Webページ上に複数の編集可能なエリアを効率良く作成できます。

水平スプリッターは、他の水平スプリッターや垂直スプリッターを組合せて、より複雑なページを構築することができます。

例えば次のようなものです。



クイック・スタート - 水平スプリッター

水平スプリッターを使用するためには、Webroutineを作成します。WEB_MAPはスプリッター・ウェブレットには直接の影響はないので、任意です。仕上がったWebページに水平スプリッターを追加するには、LANSAエディターで以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[Horizontal Splitter]ウェブレットを見つけてください。
2. [デザイン]ビューでウェブレットを必要な場所にドラッグ・アンド・ドロップしてください。ウェブレットを選択して、[詳細]タブをクリックするのを忘れないでください。
3. 必要に応じて、top_portion_percentプロパティを設定します。このプロパティに使用する適切な値は、スプリッター・ペインのコンテンツが定義された後の方がより明確になる場合もあります。
4. ウェブレットやフィールド情報をスプリッターの上部や下部に追加して、スプリッター・ペインのコンテンツを必要に応じてフォーマットします。

プロパティ - 水平スプリッター

水平スプリッター・ウェブレットのプロパティは以下のとおりです。

bottom_border name top_class
bottom_class panes top_proportion_percent
bottom_style pos_absolute top_style
divider_class top_border width
height

name

水平スプリッターの名前。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。

省略値

`concat('o', position(), '_LANSA_n')` - LANSАにより付けられたウェブレット内部名です。

有効値

単一引用符で囲まれた名前。

panes

表示する一連のペインを指定するXMLノードセット。これは、スプリッターをデザインビュー上にドラッグしたときに、システムが生成する値です。

注：この値は変更できません。参照のみです。

省略値

`document(")/*/lxml:data/lxml:splitter_panes[@id='<unique id>']` (自動的に生成された識別子unique idの現在のペインです。)

有効値

適用不可。(この値はシステムにより保守されます。)

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

'top:0pt;top:0pt;' (ウェブレットの位置は相対的に位置付けられます。)

有効値

単一引用符付きで、'top'・'top' 座標の有効な範囲内の有効な値。

width

Webページのウェブレットの幅です。

省略値

'100%' (ウェブレットが使用できる最大の幅)

有効値

単一引用符で囲まれた有効な範囲内の幅。

height

Webページのウェブレットの高さ。

省略値

フォーム・エレメントに直接配置された場合は'200pt'、他のウェブレット内にある場合は'100%'。

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

top_proportion_percent

水平スプリッターのトップ・ペインが最初に占めるスペースの割合
(パーセント表示)。

省略値

70

有効値

数値で表された有効なパーセンテージ(単一引用符は付けません。)

top_border

スプリッターの上部ペインの枠線のスタイル。上部ペインの上、左、右の枠線に適用されます。

省略値

'outset'.

有効値

枠線固有のスタイル -

'dashed'、'dotted'、'double'、'groove'、'inset'、'outset'、'ridge'、'solid'、'window-inset'。'window-inset'はInternet Explorerでのみサポートされます。

bottom_border

スプリッターの下部ペインの枠線のスタイル。下部ペインの下、左、右の枠線に適用されます。

省略値

'inset'.

有効値

枠線固有のスタイル -

'dashed'、'dotted'、'double'、'groove'、'inset'、'outset'、'ridge'、'solid'、'window-inset'。'window-inset'はInternet Explorerでのみサポートされます。

top_class

スプリッターの上部ペインに適用されるカスケード・スタイル・シートクラス。

省略値

'std_splitter_horz_top' - このウェブレット用に提供されている省略値クラスの名前。

有効値

単一引用符で囲まれた、現在のカスケード・スタイル・シートの有効なクラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

divider_class

区切りバーに適用されるカスケード・スタイル・シートクラス。

省略値

'std_splitter_vert_divider' - このウェブレット用に提供されている省略値クラスの名前。

有効値

単一引用符で囲まれた、現在のカスケード・スタイル・シートの有効なクラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

bottom_class

スプリッターの下部ペインに適用されるカスケード・スタイル・シートクラス。

省略値

'std_splitter_vert_bottom' - このウェブレット用に提供されている省略値クラスの名前。

有効値

単一引用符で囲まれた、現在のカスケード・スタイル・シートの有効なクラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

top_style

スプリッターの上部ペインのインラインのカスケード・スタイル・シートのスタイル。top_classプロパティに指定された、これに対応するカスケード・スタイル・シートの特性は上書きされます。

省略値

ブランク。

有効値

1つもしくは複数のスタイル宣言からなる単一引用符で囲まれたテキスト文字列。宣言はプロパティ、コロン、1つもしくは複数の値で構成されます。

bottom_style

スプリッターの下部ペインに適用されるインライン・スタイル。
bottom_classプロパティに指定された、これに対応するカスケード・スタイル・シートの特性は上書きされます。

省略値

空白。

有効値

1つもしくは複数のスタイル宣言からなる単一引用符で囲まれたテキスト文字列。宣言はプロパティ、コロン、1つもしくは複数の値で構成されます。

8.1.27 垂直スプリッター (std_splitter_vert)

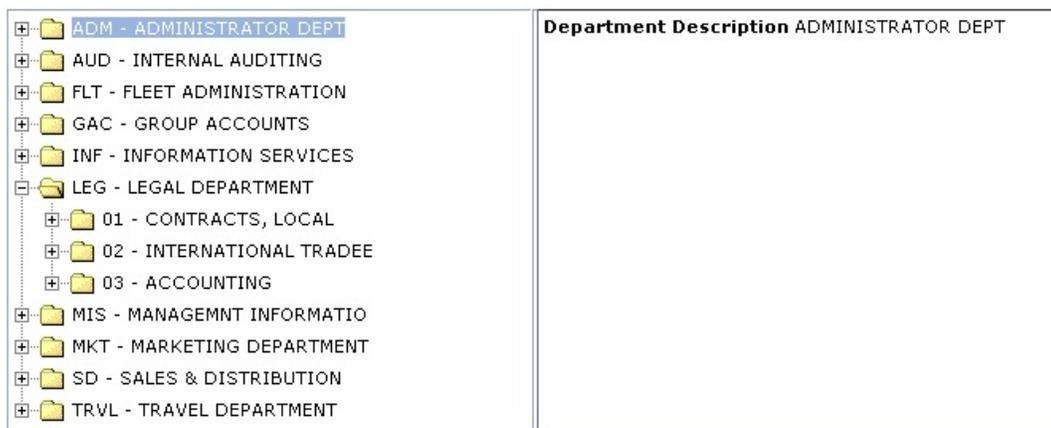
クイック・スタート - 垂直スプリッター

プロパティ - 垂直スプリッター

垂直ウェブレットは、このウェブレットがカバーするエリアを移動可能な仕切りで左右二つのペインに分けます。入出力は提供せず、Webページ・デザインをサポートするためのみに使用されます。他のエレメントやウェブレットをそれぞれのペインにドロップすることができ、Webページ上に複数の編集可能なエリアを効率良く作成できます。

垂直スプリッターは、他の垂直スプリッターや水平スプリッターを組合せて、より複雑なページを構築することができます。

例えば次のようなものです。



クイック・スタート - 垂直スプリッター

垂直スプリッターを使用するためには、Webroutineを作成します。WEB_MAPはスプリッター・ウェブレットには直接の影響はないので、任意です。仕上がったWebページに垂直スプリッターを追加するには、LANSAエディターで以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[Vertical Splitter]ウェブレットを見つけてください。
2. [デザイン]ビューでウェブレットを必要な場所にドラッグ・アンド・ドロップしてください。ウェブレットを選択して、[詳細]タブをクリックするのを忘れないでください。
3. 必要に応じて、left_portion_percentプロパティを設定します。このプロパティに使用する適切な値は、スプリッター・ペインのコンテンツが定義された後の方がより明確になる場合もあります。
4. ウェブレットやフィールド情報をスプリッターの右や左の部分に追加して、スプリッター・ペインのコンテンツを必要に応じてフォーマットします。

プロパティ - 垂直スプリッター

垂直スプリッター・ウェブレットのプロパティは以下のとおりです。

| | | |
|--------------------------------------|---------------------------|---------------------------|
| <code>divider_class</code> | <code>left_style</code> | <code>right_border</code> |
| <code>height</code> | <code>name</code> | <code>right_class</code> |
| <code>left_border</code> | <code>panes</code> | <code>right_style</code> |
| <code>left_class</code> | <code>pos_absolute</code> | <code>width</code> |
| <code>left_proportion_percent</code> | | |

name

垂直スプリッターの名前。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。

省略値

`concat('o', position(), '_LANSA_n')` - LANSAにより付けられたウェブレット内部名です。

有効値

単一引用符で囲まれた名前。

panes

表示する一連のペインを指定するXMLノードセット。これは、スプリッターをデザインビュー上にドラッグしたときに、システムが生成する値です。

注：この値は変更できません。参照のみです。

省略値

`document(")*/lxml:data/lxml:splitter_panes[@id='<unique id>']` (自動的に生成された識別子unique idの現在のペインです。)

有効値

適用不可。(この値はシステムにより保守されます。)

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

Webページのウェブレットの幅です。

省略値

'100%' (ウェブレットが使用できる最大の幅)

有効値

単一引用符で囲まれた有効な範囲内の幅。

height

Webページのウェブレットの高さ。

省略値

フォーム・エレメントに直接配置された場合は'200pt'、他のウェブレット内にある場合は'100%'。

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

left_proportion_percent

垂直スプリッターの左ペインが最初に占めるスペースの割合（パーセント表示）。

省略値

30

有効値

数値で表された有効なパーセンテージ(単一引用符は付けません。)

left_border

スプリッターの左ペインの枠線のスタイル。これは、左ペインの上、下、左の枠線に適用されます。

省略値

'outset'.

有効値

枠線固有のスタイル -

'dashed'、'dotted'、'double'、'groove'、'inset'、'outset'、'ridge'、'solid'、'window-inset'。'window-inset'はInternet Explorerでのみサポートされます。

right_border

スプリッターの右ペインの枠線のスタイル。これは、右ペインの上、下、右の枠線に適用されます。

省略値

'inset'.

有効値

枠線固有のスタイル -

'dashed'、'dotted'、'double'、'groove'、'inset'、'outset'、'ridge'、'solid'、'window-inset'。'window-inset'はInternet Explorerでのみサポートされます。

left_class

スプリッターの左ペインに適用されるカスケード・スタイル・シートクラス。

省略値

'std_splitter_vert_left' - このウェブレット用に提供されている省略値クラスの名前。

有効値

単一引用符で囲まれた、現在のカスケード・スタイル・シートの有効なクラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

divider_class

区切りバーに適用されるカスケード・スタイル・シートクラス。

省略値

'std_splitter_vert_divider' - このウェブレット用に提供されている省略値クラスの名前。

有効値

単一引用符で囲まれた、現在のカスケード・スタイル・シートの有効なクラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

right_class

スプリッターの右ペインに適用されるカスケード・スタイル・シートクラス。

省略値

'std_splitter_vert_right' - このウェブレット用に提供されている省略値クラスの名前。

有効値

単一引用符で囲まれた、現在のカスケード・スタイル・シートの有効なクラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

left_style

スプリッターの左ペインのインラインのカスケード・スタイル・シートのスタイル。left_classプロパティに指定された、これに対応するカスケード・スタイル・シートの特性は上書きされます。

省略値

空白。

有効値

1つもしくは複数のスタイル宣言からなる単一引用符で囲まれたテキスト文字列。宣言はプロパティ、コロン、1つもしくは複数の値で構成されます。

right_style

スプリッターの右ペインに適用されるインライン・スタイル。
right_classプロパティに指定された、これに対応するカスケード・スタイル・シートの特性は上書きされます。

省略値

ブランク。

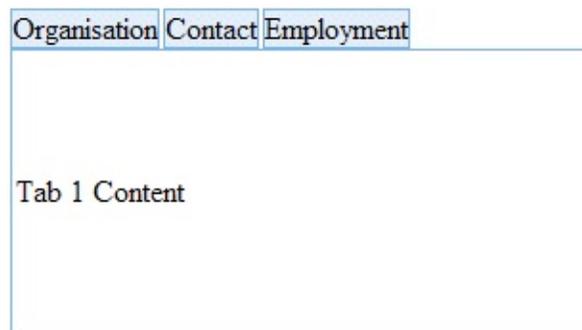
有効値

1つもしくは複数のスタイル宣言からなる単一引用符で囲まれたテキスト文字列。宣言はプロパティ、コロン、1つもしくは複数の値で構成されます。

8.1.28 タブ・ページ (std_tab_pages_v2)

クイック・スタート - タブ・ページ
プロパティ - タブ・ページ
タブ・ページ・ウェブレットでCSSを使用
タブ・アイテム・デザイナーの使用

タブ・ページ・ウェブレットは、関連する情報をグループとしてまとめる際に使用できるタブ・コントロール機能を提供します。次の図はタブ・ページ・ウェブレットがWebページに埋め込まれた時にどのように表示されるかを示しています。この例では、共通情報はページのトップにグループ化されています。タブ・コントロールはその下に表示され、社員についての3つの情報グループ(Organization(組織)、Contacts(連絡先)、Employment(雇用))にアクセスできます。Contactsタブが表示されていますが、タブをクリックすることで、ユーザーは他のグループの情報を表示できます。



このウェブレットには豊富な種類のプロパティが提供されており、その外観や動作に影響を与えます。タブ・アイテム自体は通常タブ・アイテム・デザイナーを使用して静的に指定します。時には、タブ・アイテムのプロパティを動的に上書きしたい場合もあるでしょう。その場合、listnameおよびこれに関連するプロパティを指定します。

これを含むWebroutineからWEB_MAPでコンテンツが提供される場合、コンテンツを直接各タブページに追加できます。これは通常通りの方法で行えます。例えば、フィールドタブからフィールドをドラッグ・アンド・ドロップすれば良いのです。同様に、他のウェブレットをタブ・ページに追加することもできます。

代わりに、ナビゲーション・パネル(std_nav_panel)ウェブレットを1つもしくは複数のタブ・ページに追加して、各ナビゲーション・パネルのプロパティを設定することで、別のWebroutine(もしくはURL)により提供

されたコンテンツを表示することもできます。

クイック・スタート - タブ・ページ

タブ・ページ・ウェブレットを使用するには、LANSAエディターで Webroutineを開いて、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[タブ ページ]ウェブレットを見つけてください。
2. [デザイン]ビューでページ上にウェブレットをドラッグ・アンド・ドロップします。ページ上のウェブレットを選択してから、[詳細]タブをクリックするのを忘れないでください。
3. tabs プロパティの横にある省略記号(...)のボタンをクリックし、タブ・アイテム・デザイナを開いてタブの定義をしてください。終了したら、OKをクリックし、タブ・アイテム・デザイナを閉じてください。
4. [デザイン]ビューで最初のタブをクリックして、そのページをアクティブ化させます。
5. 別の場所からコンテンツを切り取って、Webページに貼り付けたり、タブ・ページ上にフィールドやウェブレットをドラッグ・アンド・ドロップして、タブ・ページにコンテンツを追加します。複数のフィールドを追加する場合は、このフィールドを入れるテーブルを追加してから始めます。そのためにはタブページで右クリックして、[HTMLの挿入]をクリックし、表示されたポップアップ・メニューから[Table]を選択してください。
6. 他のWebroutineにより提供されるコンテンツをタブ・ページに表示したい場合は、ナビゲーション・パネルを追加し、そのプロパティを設定します。これは以下の手順に従ってください。
 - a. [ウェブレット テンプレート]タブをクリックし、上のドロップダウン・リストから[標準ウェブレット]を選択、[ナビゲーション パネル]ウェブレットを見つけてください。
 - b. [デザイン]ビューでページ上にウェブレットをドラッグ・アンド・ドロップします。タブ・ページ上のナビゲーション・パネル・ウェブレットを選択してから、[詳細]タブをクリックするのを忘れないでください。
 - c. nav_wamname プロパティとnav_wrname プロパティ(もしくはnav_url プロパティ)を設定して、コンテンツを提供する

Webroutine(もしくはURL)が識別できるようにします。wait_content プロパティおよびこれに関連するプロパティなど、ナビゲーション・パネルのその他のプロパティを設定する必要がある場合もあるでしょう。

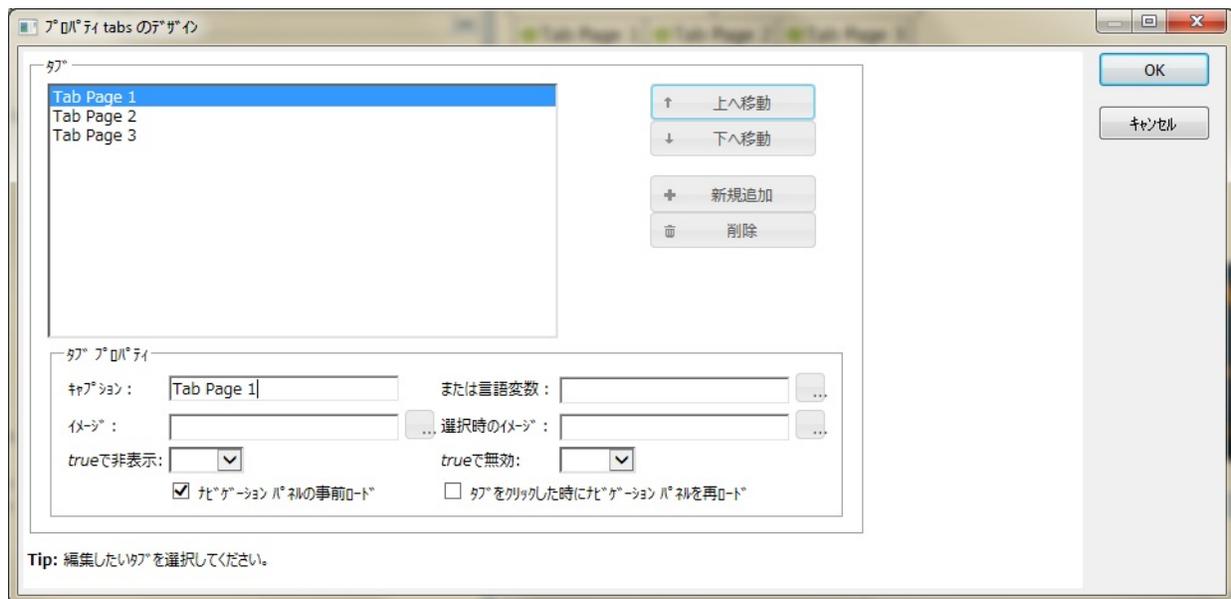
7. タブ・ページ・ウェブレットに定義した各タブについて、ステップ4以降を繰り返します。

注：Internet Explorerの現バージョンのバグにより、空のタブのコンテンツ・エリアの周辺の枠線が表示されません。Internet Explorerでのこの動作を防ぐ為に、タブが空にならないようにしてください。

タブ・アイテム・デザイナーの使用

タブ・アイテム・デザイナーを開くには、以下の手順に従ってください。

1. ページ上のタブ・ページ・ウェブレットが選択されていることを確認し、[詳細]タブをクリックしてください。
2. [詳細]タブのtabsプロパティの上にマウス・ポインタを動かしてください。省略記号(...)がプロパティ値の横に表示されます。省略記号(...)のボタンをクリックし、タブ・アイテム・デザイナーを開きます。以下のようなウィンドウが表示されます。



ウィンドウには、ウェブレットに定義されている現在のタブ・アイテムの一覧が表示されます。アイテムをクリックして、次のことができます。

- プロパティを変更する。
- ボタンを使ってアイテムを上下に動かしたり、アイテムを削除する。

[タブ プロパティ]では、選択したアイテムのプロパティを以下のように指定することができます。

[キャプション]

タブ・アイテムに表示するテキストを指定します。複数言語アプリケーションを作成する場合は、言語変数を使用する必要があります。キャプションまたは言語変数のいずれかを指定できます。

[または言語変数]

タブ・アイテムに表示するテキストを含む複数言語対応テキスト変数の名前を指定します。このプロパティの横にある省略記号(...)をクリックし、複数言語対応テキスト変数の一覧から選択してください。[キャプション]、[または言語変数]のいずれかの欄のみ指定可能です。

[イメージ]

イメージ仮想ディレクトリに関連付けられた、タブに表示されるイメージのパスとファイル名を指定します。指定された場合、tab_imageプロパティに指定されたイメージが上書きされます。

[選択時のイメージ]

イメージ仮想ディレクトリに関連付けられた、選択時にタブに表示されるイメージのパスとファイル名を指定します。指定された場合、tab_selected_imageプロパティに指定されたイメージが上書きされます。

[trueで非表示]

タブ・アイテムを非表示にすることを示すフィールドの名前を指定します。ここでの非表示とはタブおよびそのコンテンツがブラウザに全く送られないという意味です。タブが非表示になるのは、フィールドにブール値true、または数字の1が含まれる場合、もしくは文字列の値がtrue、TRUE、y、Y、または1の場合です。

[trueで無効]

タブ・アイテムを無効にすることを示すフィールドの名前を指定します。タブが無効になるのは、フィールドにブール値true、または数字の1が含まれる場合、もしくは文字列の値がtrue、TRUE、y、Y、または1の場合です。

[ナビゲーション パネルの事前ロード]

タブ・ページにナビゲーション・パネルがある場合、このチェックボックスは、タブ・ページを含むページがロードされる時に、選択されていないタブのナビゲーション・パネルをロードするかどうかをコントロールします。ナビゲーション・パネルによってアクセスされるWebroutineやURLのリソースのコストが高くなる場合や、単に多くのタブがある場合などは、1つもしくは全てのタブのこのボックスのチェックを外すことができます。このチェックボックスのチェックが外れている場合、ナビゲーション・パネルは対象のタブ・ページが選択されていない時はロードされず、選択されるとロードされます。

[タブをクリックした時にナビゲーション パネルを再ロード]

タブ・ページにナビゲーション・パネルが含まれている場合、この
チェックボックスはナビゲーション・パネルが1度(最初に表示される時)
ロードされるか、またはタブページが表示されるたび毎回ロードするか
をコントロールします。

タブ・ページ・ウェブレットでCSSを使用

省略値CSSの機能 独自のCSSスタイルを追加

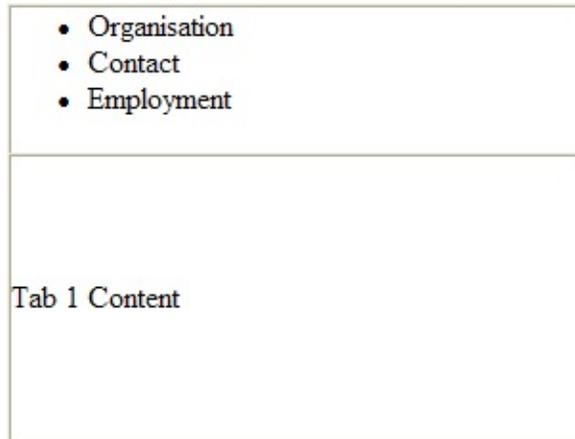
タブ・ページ・ウェブレットはHTMLテーブルを使用して構成されています。またタブ自体は順不同のリストを使用して構成されています。タブ・ページのHTMLは次のような感じになります。

```
<table id="TabPageName" class="std_tab_pages" cellpadding="0" cellspacing="0">
  <tbody>
    <tr>
      <td class="std_tab_pages_top_tabs">
        <ul class="std_tab_pages_tabs">
          <li class="std_tab_active">
            <a onclick="return stdTabPageTabClicked(this, 1)">Organisation</a>
          </li>
          <li>
            <a onclick="return stdTabPageTabClicked(this, 2)">Contact</a>
          </li>
          <li>
            <a onclick="return stdTabPageTabClicked(this, 3)">Employment</a>
          </li>
        </ul>
      </td>
    </tr>
    <tr>
      <td class="std_tab_pages_content_wrapper">
        <div class="std_tab_pages_content">
          Tab 1 Content
        </div>
        <div class="std_tab_pages_content" style="display:none">
          Tab 2 Content
        </div>
        <div class="std_tab_pages_content" style="display:none">
          Tab 3 Content
        </div>
      </td>
    </tr>
  </tbody>
</table>
```

```
</tbody>  
</table>
```

省略値CSSの機能

CSSが無い状態では、タブ・ページのHTMLは次のような結果になります。(分かりやすくするため、テーブルに枠線が追加されています。)



まず始めに、ULタグからlist-styleとmarginを取り除く必要があります。

```
ul.std_tab_pages_tabs
{
    list-style-type: none;
    margin:0px;
}
```

次に、LIタグに枠線と背景色を追加します。これを水平に配置するためにfloat:leftを使用し、marginを使ってこれを分割します。

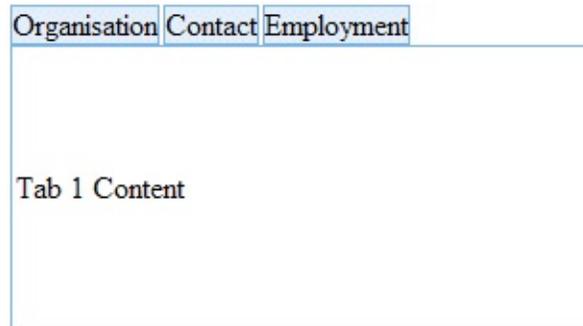
```
ul.std_tab_pages_tabs li
{
    border:1px solid #7db0e5;
    background-color:#e2effa;
    color: black;
    white-space: nowrap;
    display: block;
    float: left;
    margin-right:2px;
}
```

最後に、コンテンツ・エリアの周りに枠線を追加します。

```
.std_tab_pages_content_wrapper
{
    border:1px solid #7db0e5;
```

```
padding:2px;
}
```

これで、テーブルとリストは、次のようになります。



次に、タブ・テキストの周りに少しスペースを入れ、選択されたタブの背景色を変更します。選択されたタブの下部の枠線を取り除き、その下部の余白を増やして、このタブを縦方向に伸ばしスペースを埋める必要があります。(これを行うのは、下部の枠線の色を変更しようとする
と、Internet Explorerが左右の枠線を不均等に描いてしまうからです。)

```
ul.std_tab_pages_tabs li a
{
    display: block;
    padding:3px;
    text-decoration: none;
    font-weight: bold;
}
ul.std_tab_pages_tabs li.std_tab_active
{
    background-color: white;
    color: black;
    border-bottom: none;
    padding-bottom:1px;
}
```

最後に、タブを枠線の幅分だけ下に動かして、その下部の枠線がコンテンツ・エリアの枠線に重なるようにします。これは、相対配置されたタブを含むテーブルのセルを作り、これを枠線の幅分だけ下に位置を動かすとできます。

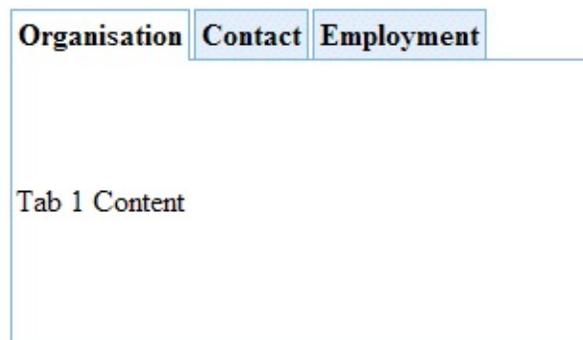
```
.std_tab_pages_top_tabs
{
```

```
position: relative;
top:1px;
}
```

このセル移動の方法はInternet Explorerでのみ使用可能です。FirefoxやOperaでは、枠線の幅分だけULブロックを下に動かす必要があります。(これはIEではできません。)

```
ul.std_tab_pages_tabs
{
    position: relative;
    top:1px;
}
```

最終的な結果は次のようになります。



使用されるスタイルの中には、コンテンツ・エリアの上部に沿って配置されたタブにのみ適用されるものがあるので、プロパティが上部に配置されたタブにのみ確実に適用されるように、より特定のセレクターを持ったスタイルを作成する必要があります。

```
.std_tab_pages_top_tabs ul
{
    position: relative;
    top:1px;
}
.std_tab_pages_top_tabs ul li.std_tab_active
{
    border-bottom: none;
    padding-bottom:1px;
}
```

独自のCSSスタイルを追加

タブ・ページ・ウェブレットの外観は、カスタム・スタイルシートにくつつかスタイルを追加して省略値を上書きすることで、簡単に修正することができます。例えば、以下は枠線と背景色を変更します。

```
.std_tab_pages_content_wrapper
{
    border:1px solid #81a594;
}
ul.std_tab_pages_tabs li
{
    border:1px solid #81a594;
    background-color:#cbcbb8;
}
```

枠線の太さを変えると、選択されたタブの余白と、タブをコンテンツに重なるように移動する量も変更する必要があります。次のようなカスタム・スタイルがあったとします。

```
.std_tab_pages_content_wrapper
{
    border:3px solid #81a594;
}
ul.std_tab_pages_tabs li
{
    border:3px solid #81a594;
    background-color:#cbcbb8;
}
.std_tab_pages_top_tabs,
.std_tab_pages_top_tabs ul
{
    position: relative;
    top:3px;
}
.std_tab_pages_top_tabs ul li.std_tab_active
{
    border-bottom: none;
    padding-bottom:3px;
}
```

これは、次のような結果になります。

| Organisation | Contact | Employment |
|-----------------------|---------|------------------|
| Street No and Name | | 17 Grantham Road |
| Suburb or Town | | SEVEN HILLS |
| State and Country | | NSW |
| Post / Zip Code | | 2147 |
| Home Phone Number | | 718 1891 |
| Business Phone Number | | 565 2341 |

タブ・ページ・ウェブレットの以前のInternet Explorerのみのバージョンは、左または右に整列されたタブに縦にテキストが表示されるようになっていました。これはInternet Explorerでのみ可能な機能のため、_v2バージョンからは削除されました。左に整列されたタブは次のようになります。

| Organisation | Street No and Name | 17 Grantham Road |
|--------------|-----------------------|------------------|
| Contact | Suburb or Town | SEVEN HILLS |
| Employment | State and Country | NSW |
| | Post / Zip Code | 2147 |
| | Home Phone Number | 718 1891 |
| | Business Phone Number | 565 2341 |

ターゲット・ブラウザがInternet Explorerの場合のみ、この効果を*writing-mode*プロパティで再適用することが可能です。

```
.std_tab_pages_left_tabs ul li a
{
    writing-mode:tb-rl;
}
```

| | | |
|--------------|-----------------------|------------------|
| Organisation | Street No and Name | 17 Grantham Road |
| | Suburb or Town | SEVEN HILLS |
| | State and Country | NSW |
| | Post / Zip Code | 2147 |
| | Home Phone Number | 718 1891 |
| | Business Phone Number | 565 2341 |
| Contact | | |
| Employment | | |

下から上へ記述されたテキストが必要な時は、*filter* プロパティを使用して180度回転させることもできます。

```
.std_tab_pages_left_tabs ul li a
{
    writing-mode:tb-rl;
    filter: flipv fliph;
}
```

| | | |
|--------------|-----------------------|------------------|
| Organisation | Street No and Name | 17 Grantham Road |
| | Suburb or Town | SEVEN HILLS |
| | State and Country | NSW |
| | Post / Zip Code | 2147 |
| | Home Phone Number | 718 1891 |
| | Business Phone Number | 565 2341 |
| Contact | | |
| Employment | | |

プロパティ - タブ・ページ

タブ・ページ・ウェブレットのプロパティは以下のとおりです。

| | | |
|---------------------------------------|--|-------------------------------------|
| caption_field | image_field | tab_alignment |
| disable_if_true_field | listname | tab_image |
| formname | name | tab_image_alignment |
| content_height | pos_absolute_design | tab_image_height |
| content_width | selected_image_field | tab_image_width |
| hide_if | selected_tab_index | tab_selected_image |
| hide_if_true_field | selected_tab_index_field | tabs |

name

ウェブレットを識別する名前です。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用しても構いません。

省略値

自動的に生成された、一意の識別子。

有効値

単一引用符で囲まれた文字列。

tabs

表示するタブ・アイテムを指定するXMLノード・セット。これは、タブ・ページ・ウェブレットを[デザイン]ビュー上にドラッグしたときに、システムが生成する値です。

表示された値を直接編集しないでください。代わりに、省略記号(...)のボタンをクリックし、タブ・アイテム・デザイナを開きます。詳しくは、「タブ・アイテム・デザイナの使用」を参照してください。

省略値

```
document(")/*/lxml:data/lxml:tab[@id='<unique id>']
```

<unique id> は自動的に生成される識別子です。

有効値

適用できません。(この値はシステム生成され、修正してはいけません。)

selected_tab_index

Webページが表示された時に最初に選択されているべきタブのインデックスを指定します。

注：[デザイン]ビューでタブをクリックすると、現在の選択されているタブのインデックスがこの値に変更されます。タブ・ページのデザイン時はこの作業を頻繁に行うでしょうから、最後に保存する前に、最初のタブ(もしくは、初期選択されるべきタブ)を再度選択するのを忘れないようにしてください。

省略値

1

有効値

1からタブ数(この数を含む)の間の整数値、もしくは実行時にタブ・インデックスを含むフィールドの名前。

指定された値が1より小さい、もしくはタブ数より大きい場合、どのタブも初期選択されません。多くのタブ・ページのアプリケーションで、これはあまり好ましいものではありません。

selected_tab_index_field

選択されたタブのインデックスを含むフィールド名を指定します。これはselected_tab_indexプロパティとよく似ています。実際

selected_tab_indexプロパティには、フィールド名を指定することができます、その値は初期選択されるタブのインデックスをコントロールします。ただしこのプロパティを代わりに使用すると、現在選択されているタブのインデックスでフィールドを更新するという追加の効果があります。このフィールドがWebroutineに送り返された場合、プログラムはどのタブが最後に表示されたかを知ることができます。例えば、この情報を保存し、次回Webroutineが同じユーザーでロードされた時に、これを使用して前回表示した時と同じタブを表示することができます。

省略値

空白。

有効値

WebroutineのWEB_MAPにあるフィールド名。(通常、一番大きなタブ番号が格納できる大きさの数値フィールドを指定します。)フィールド名は単一引用符で囲みます。このプロパティの横にあるドロップダウンの矢印をクリックし、使用可能なフィールド名のリストから選択してください。

tab_alignment

このプロパティはタブ・ボタンがタブ・ページの上、左、右、下のどこに表示されるかを決定します。

省略値

'top'(上)

有効値

'top'(上)、'left'(左)、'right'(右)、もしくは'bottom'(下)

tab_image

イメージ仮想ディレクトリに関連付けられた、各タブに表示されるイメージのパスとファイル名。各タブがタブ・アイテム・デザイナーで上書きされない限り、このプロパティが全てのタブに適用されます。

省略値

'ball_grn.gif' (LANSAにより提供される標準イメージです)

有効値

イメージ・ディレクトリに関連付けられたイメージのパスおよび名前、単一引用符で囲まれたもの。イメージは、プロパティ・シートで該当する省略記号のボタンをクリックし、プロンプターから選択することができます。タブにイメージを表示したくない場合は、空の文字列(間に何も挟まない2つのクォーテーション・マークを使用)を入力することができます。

tab_selected_image

イメージ仮想ディレクトリに関連付けられた、タブ選択時に各タブに表示されるイメージのパスとファイル名。指定されると、各タブがタブ・アイテム・デザイナーで上書きされない限り、このプロパティは全てのタブに適用されます。

省略値

空白 - タブ・アイテムが選択された時にイメージは変わりません。

有効値

イメージ・ディレクトリに関連付けられたイメージのパスおよび名前、単一引用符で囲まれたもの。イメージは、プロパティ・シートで該当する省略記号のボタンをクリックし、プロンプターから選択することができます。

tab_image_height

タブ・ボタン・イメージの高さ。

省略値

'14px'

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

tab_image_width

タブ・ボタン・イメージの幅。

省略値

'14px'

有効値

単一引用符で囲まれた有効な範囲内の幅。

tab_image_alignment

このプロパティでは、タブ・ボタン・イメージをキャプションの左に表示するか右に表示するかを指定します。

省略値

'left'

有効値

'left'(左)または'right'(右)

listname

実行時にタブ・アイテムの属性を動的に上書きする際に使用するWEB_MAPに指定された作業リスト名。作業リストを使用すると、以下の属性の1つ以上を上書きできます。

- タブのキャプション (作業リストのフィールド名をcaption_fieldプロパティに指定)
- タブのイメージ (作業リストのフィールド名をimage_fieldプロパティに指定)
- タブ選択時のイメージ (作業リストのフィールド名をselected_image_fieldプロパティに指定)
- タブの非表示 (作業リストのフィールド名をhide_if_true_fieldプロパティに指定)
- タブの無効化 (作業リストのフィールド名をdisable_if_true_fieldプロパティに指定)

この機能を使用するには、上書きする上記の1つ以上の属性に対応する1つ以上のフィールドを含む作業リストをWEB_MAPで(*OUTPUT もしくは*BOTHで)指定する必要があります。WebroutineのRDMLプログラムで、タブ・アイテムの数と同じ数のエントリーでリストを埋めていきます。その際、フィールド値が上記に示される各プロパティに設定されている必要があります。

注：必要な数のタブアイテムを作成して、作業リストに指定されていないタブ・アイテムの属性を設定するためには、タブ・アイテム・デザイナーを使用する必要があります。詳しくは、「メニュー・アイテム・デザイナーの使用」を参照してください。

省略値

ブランク - 実行時にタブ・アイテムの属性を上書きする作業リストは使用されません。タブはタブ・アイテム・デザイナーで定義されたとおりに表示されます。

有効値

WebroutineのWEB_MAPに指定された作業リスト名。使用可能な作業リストの一覧は、プロパティシートの対応するドロップダウン・ボタンをクリックすることで、選択することができます。

例

この例では、実行時にタブ・アイテムの特定の属性を上書きするために、A10TABSという名前の作業リストが使用されます。各タブ・アイテムの上書きする値を含んだフィールド名は、caption_field、image_field、selected_image_field、hide_if_true_field、disable_if_true_fieldの中の1つ以上のプロパティに指定されなければいけません。

| | |
|--|-----------|
|  listname | 'A10TABS' |
|--|-----------|

caption_field

このプロパティには、各タブ・アイテムのタブ・キャプションを上書きするlistnameプロパティに指定された作業リストのフィールド名を指定します。このプロパティはlistnameプロパティが指定されていない場合、無視されます。また指定は任意です。詳しくは、「listnameプロパティ」を参照してください。

省略値

ブランク - 実行時、タブ・キャプションはlistnameプロパティに指定された作業リスト経由で上書きされません。

有効値

各タブ・アイテムのタブ・キャプションを含む、listnameプロパティに指定された作業リスト内のフィールドの名前。プロパティシートの対応するドロップダウン・ボタンをクリックすることで、使用可能なフィールドのリストから選択することができます。

image_field

このプロパティには、各タブ・アイテムの(イメージ仮想ディレクトリに関連付けられた)タブ・イメージのファイル名を上書きする、listnameプロパティに指定された作業リスト内のフィールド名を指定します。このプロパティはlistnameプロパティが指定されていない場合、無視されます。また指定は任意です。詳しくは、「listnameプロパティ」を参照してください。

省略値

ブランク - 実行時、タブ・イメージのファイル名はlistnameプロパティに指定された作業リスト経由で上書きされません。

有効値

各タブ・アイテムの(イメージ仮想ディレクトリに関連付けられた)タブ・イメージのファイル名を上書きする、listnameプロパティに指定された作業リスト内のフィールドの名前。プロパティシートの対応するドロップダウン・ボタンをクリックすることで、使用可能なフィールドのリストから選択することができます。

selected_image_field

このプロパティには、各タブ・アイテムの(イメージ仮想ディレクトリに関連付けられた)タブ選択時のイメージ・ファイル名を上書きする、listnameプロパティに指定された作業リストのフィールド名を指定します。このプロパティは任意であり、listnameプロパティが指定されていない場合、無視されます。詳しくは、「listnameプロパティ」を参照してください。

省略値

ブランク。実行時、タブ選択時のイメージ・ファイル名はlistnameプロパティに指定された作業リスト経由で上書きされません。

有効値

各タブ・アイテムの(イメージ仮想ディレクトリに関連付けられた)タブ選択時のイメージ・ファイル名を上書きする、listnameプロパティに指定された作業リスト内のフィールドの名前。プロパティシートの対応するドロップダウン・リストの使用可能なフィールドのリストから選択することができます。

hide_if_true_field

このプロパティには各タブ・ページの非表示状態を上書きする、listnameプロパティに指定された作業リストのフィールド名を指定します。このプロパティはlistnameプロパティが指定されていない場合、無視されます。また指定は任意です。詳しくは、「listnameプロパティ」を参照してください。

省略値

ブランク - 実行時、非表示状態はlistnameプロパティに指定された作業リスト経由で上書きされません。

有効値

各タブ・ページの非表示状態を上書きする、listnameプロパティに指定された作業リスト内のフィールドの名前。プロパティシートの対応するドロップダウン・ボタンをクリックすることで、使用可能なフィールドのリストから選択することができます。

タブ・ページを非表示にするには、Webroutineでtrueと認識される次の値のいずれかをフィールドに格納する必要があります。'y'、'Y'、'true'または'1'。

disable_if_true_field

このプロパティには各タブ・ページの無効状態を上書きする、listnameプロパティに指定された作業リストのフィールド名を指定します。このプロパティはlistnameプロパティが指定されていない場合、無視されます。また指定は任意です。詳しくは、「listnameプロパティ」を参照してください。

省略値

ブランク - 実行時、無効状態はlistnameプロパティに指定された作業リスト経由で上書きされません。

有効値

各タブ・ページの使無効状態を上書きする、listnameプロパティに指定された作業リスト内のフィールドの名前。プロパティシートの対応するドロップダウン・ボタンをクリックすることで、使用可能なフィールドのリストから選択することができます。

タブ・ページを無効にするには、Webroutineは次のtrueと認識される値のいずれかをフィールドに格納する必要があります。'y'、'Y'、'true'または'1'。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。このウェブレットを非表示にするとは、タブ・ページとそのすべてのコンテンツが非表示になるということです。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

例

次の例はフィールド#STD_FLAGが 'Y' と等しい場合にウェブレットを非表示にします。この式は以下のような形式で入力してください。

| | |
|---------|-----------------|
| hide_if | #STD_FLAG = 'Y' |
|---------|-----------------|

プロパティがフォーカスを失うと、この式は以下のように表示されます。

| | |
|---------|--------------------------------------|
| hide_if | key('field-value', 'STD_FLAG') = 'Y' |
|---------|--------------------------------------|

formname

サーバーに送られるHTMLフォーム名です。

省略値

'LANSA'

有効値

単一引用符付きのフォーム名。プロパティ・シート内で該当するドロップダウン・ボタンをクリックすると、使用できるフォーム名リストが表示されます。

pos_absolute_design

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

content_width

ウェブレットのコンテンツ・エリアの幅。つまり、ウェブレットのタブを含まない部分の幅です。

空の文字列 ("") を指定した場合、幅は自動的に選択されたタブのコンテンツの幅に調整されます。

省略値

'300px'

有効値

単一引用符で囲まれた有効な範囲内の幅。

content_height

ウェブレットのコンテンツの・エリアの高さ。つまり、ウェブレットのタブを含まない部分の高さです。

空の文字列("")を指定した場合、高さは自動的に選択されたタブのコンテンツの高さに調整されます。

省略値

'150px'

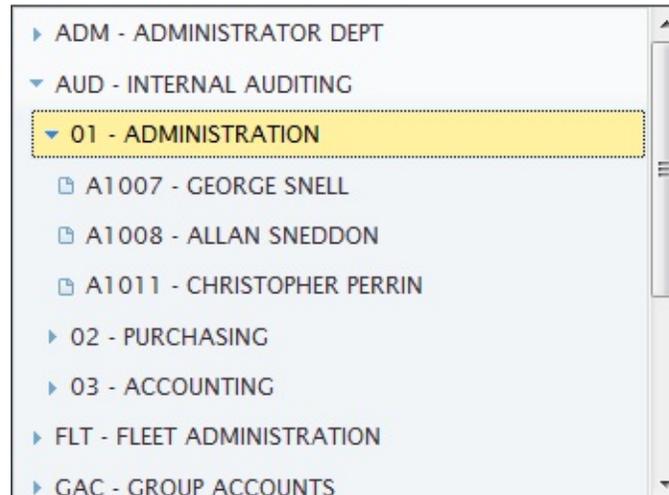
有効値

有効な範囲内にある単一引用符で囲まれた高さ。

8.1.29 ツリー・ビュー (std_treeview)

クイック・スタート - ツリー・ビュー プロパティ - ツリー・ビュー

ツリー・ビュー・ウェブレットは、展開・折りたたみができるツリーを提供します。複雑な階層データの表示や、サイト・ナビゲーション・システムに便利な機能です。



<iframe>やナビゲーション・パネル・ウェブレットと一緒に使って、ツリー・アイテム選択の応答として詳細情報を表示することもできます。(これは以前のバージョンで使用されていたツリービュー・ターゲット・ウェブレットに取って代わります。)

| | | |
|------------------------------|----------------------------------|--------------------|
| ▶ ADMINISTRATOR DEPT | Street No and Name | 22 Railway Parade, |
| ▼ INTERNAL AUDITING | Suburb or Town | KOGARAH. |
| ▼ ADMINISTRATION | State and Country | NSW. |
| ▢ A1007 - GEORGE SNELL | Post / Zip Code | 2160 |
| ▢ A1008 - ALLAN SNEDDON | Home Phone Number | 476 2198 |
| ▢ A1011 - CHRISTOPHER PERRIN | Business Phone Number | 630 8888 |
| ▶ PURCHASING | Employee Salary | 450,000.04 |
| ▶ ACCOUNTING | Start Date (DDMMYY) | 1/12/86 |
| ▶ FLEET ADMINISTRATION | Termination Date (DDMMYY) | 0/00/00 |
| ▶ GROUP ACCOUNTS | Monthly Salary | 37,500.00 |
| ▶ INFORMATION SERVICES | | |
| ▶ LEGAL DEPARTMENT | | |
| ▶ MANAGEMNT INFORMATIO | | |
| ▶ MARKETING DEPARTMENT | | |
| ▶ SALES & DISTRIBUTION | | |
| ▶ TRAVEL DEPARTMENT | | |

ツリー・ビューは作業リストからのデータで埋められます。VLツリー・ビューと同様、ソース・データは非階層、つまり各エントリーにIDがあり、その親のIDを指定する場合と、階層、つまりリストのキー欄を元にソートされたリストからツリーが構築される場合があります。非階層のソース・データを使用する場合、リストでAjaxを使用するように構成して、ユーザーにより分岐が開かれた時にサーバーから子エントリーを要求することができます。

クイック・スタート - ツリー・ビュー

[非階層リスト](#)

[階層リスト](#)

[非階層リストでAjaxを使用](#) [アイテム選択時の反応](#)

非階層リスト

非階層リストでは、各エントリーは親IDを指定することで、ツリー構造内の位置をツリー・ビューに知らせます。非階層リストは、最低限次のデータを含むカラムを含んでいなければいけません。

| フィールド | ツリー・ビューのプロパティ | 説明 |
|--------|-----------------------------------|------------------------------------|
| ID | <code>list_id_field</code> | エントリーを識別する一意のID文字列。 |
| 親ID | <code>list_parent_id_field</code> | 親エントリーのID文字列。空の文字列は最上位のエントリーを示します。 |
| キャプション | <code>list_caption_field</code> | エントリー用に表示するテキスト |

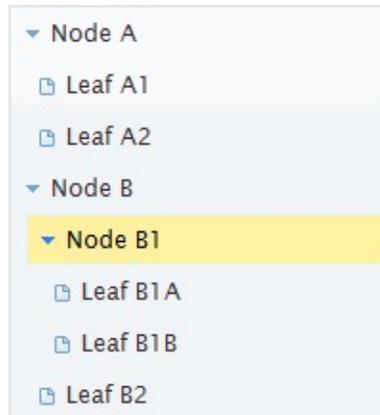
ツリー・ビューは提供された順にリストのエントリーを処理し、存在しない親にはエントリーを追加することができません。子の前に親のアイテムが来るようにリストがソートされていること、そして同じレベルのアイテムが表示順になっていることを、自身で確認する必要があります。これを行う1つの方法としては、深さを示すカラムを追加し、この深さでソートした後に順番にソートします。

例えば、次のような作業リストがあったとします。

| ID | 親ID | キャプション |
|----|-----|----------|
| 1 | | Node A |
| 2 | | Node B |
| 3 | 1 | Leaf A1 |
| 4 | 1 | Leaf A2 |
| 5 | 2 | Node B1 |
| 6 | 2 | Leaf B2 |
| 7 | 5 | Leaf B1A |

| | | |
|---|---|----------|
| 8 | 5 | Leaf B1B |
|---|---|----------|

これは次のようなツリーになります。



この例ではIDと親IDのカラムに数字が含まれていますが、これらはテキストか文字列フィールドでなければいけないことに注意してください。

非階層リストに更に多くのカラムを追加して、動作や該当するエントリーの外観をカスタマイズすることもできます。

| フィールド | ツリー・ビューのプロパティ | 説明 |
|-----------|--|--|
| イメージ | list_image_field | イメージ・ディレクトリに関連付けられた、エントリーのアイコンに使用するイメージのパスを指定します。 |
| オープン・イメージ | list_open_image_field | イメージ・ディレクトリに関連付けられた、エントリーが展開された時のエントリーのアイコンに使用するイメージのパスを指定します。 |
| 選択 | list_is_selected_field | エントリーの最初の状態が選択された状態であること |

| | | |
|---------------------------------|---|--|
| | | <p>を示します。このフィールドは文字列フィールドで、何も含まれない（選択されない）か次の2つのいずれかの値が含まれます。'true'または'freeze'。両方の値はエントリーが選択されることを示しますが、違いはツリー・ビューの動作です。値が'true'の場合、ツリー・ビューはユーザーがそのエントリーをクリックした時に行われる構成通りのアクションを取ります。つまり、ユーザーがエントリーを選択した時と同じ動作をシミュレートします。値が'freeze'の場合は、ツリー・ビューでアイテムは選択された状態になりますが、その他は何もアクションが取られません。</p> |
| 展開 | list_is_expanded_field | <p>エントリーの最初の状態が展開された状態であることを示します。エントリーに子がない場合は、無視されます。このフィールドは、数字でも文字列でも構いません。'true'、'y'または1の値はtrueとして扱われ、その他全ての値はfalseとして扱われます。</p> |
| 選択時
WAM
選択時
Webroutine | list_onselect_wamname_field
list_onselect_wrname_field | <p>アイテムがクリックされた時に呼び出されるWAM/Webroutineの名前。これらのフィールドに値が何も提供されない場合、onselect_wamnameや</p> |

| | | |
|--|--|----------------------------------|
| | | onselect_wrnameに指定された省略値が使用されます。 |
|--|--|----------------------------------|

非階層リストでAjaxを使用

非階層リストを使用する際、リストを部分的に定義して、後で必要な時にツリー・ビューがエントリーを取り出すことが可能です。このテクニックは、特に大きなツリー構造で最初のロードやページのレンダリングが遅くなる場合などに便利です。

このテクニックを使用するには、作業リストに"子持ち"フィールドを追加する必要があります。このフィールド(YかNを含む1バイトのフィールド)は、子がない場合でも、アイテムを展開可能ノードとして表示するようツリー・ビューに伝えます。

ユーザーが子のないノードを展開すると、ツリー・ビューはWebroutineを呼び出してこのツリーに追加する新しいエントリーを含む作業リストの新しいコピーを要求します。この目的で使用するWebroutineは、`onexpand_wamname/onexpand_wrname`で定義されます。このWebroutineは、ツリー・ビューが自動的にJSON応答を要求するので、ユーザー・インターフェースは必要ありません。

次の2つのフィールドとリストがWebroutineに送られます。

フィールド ツリー・ビューのプロパティ 説明

ID `onsubmit_id_field` 展開されたアイテムのID。

レベル `onsubmit_level_field` 展開されたアイテムのレベル。

先祖 `onsubmit_ancestor_list` 展開されたアイテムの先祖を含むリスト。このリストには、ツリー作成時に使用された作業リストのIDフィールドと同じ1つのフィールドが含まれます。

上記のプロパティに情報を格納するフィールドを指定できますが、これらのフィールドはonsubmitWebroutine用にWEB_MAPで定義する必要があります。

階層リスト

階層リストは各レベルのIDとキャプションが異なるカラムで表されているリストです。例えば、リストに部門、課、氏名の3つのカラムが含まれていたとします。ツリー・ビューは自動的に"課"のエントリーを"部門"のエントリーの子として、そして"氏名"のエントリーを"課"のエントリーの子として作成します。

これを制御するプロパティは次の2つです。

key_fields カンマで区切られたフィールド名のリストです。これらのフィールドは各レベルのキーとして使用されます。リストが処理される際、順番にこのキー・フィールドが前回のエントリーのキー・フィールドと比較され、変更されていれば、新しいエントリーがそのキー・フィールドのレベルで作成されます。

display_fields カンマで区切られたフィールド名のリストです。このフィールドには該当するレベルに表示するテキストが含まれます。

例えば、次のような作業リストがあったとします。

| DEPARTMENT | SECTION | EMPNO | DISPNAME | DEPTDESC |
|------------|---------|-------|-----------------------|----------------------|
| ADM | 01 | A1001 | BEN JONES | ADMINISTRAT
DEPT |
| AUD | 01 | A1007 | GEORGE
SNELL | INTERNAL
AUDITING |
| AUD | 01 | A1008 | ALLAN
SNEDDON | INTERNAL
AUDITING |
| AUD | 01 | A1011 | CHRISTOPHER
PERRIN | INTERNAL
AUDITING |
| AUD | 02 | A1009 | DAMIAN
SNASHALL | INTERNAL
AUDITING |
| AUD | 03 | A0907 | ANNE MISS
SIMPSON | INTERNAL
AUDITING |
| | | | | |

| | | | | |
|------|----|-------|-----------------------------|-------------------------|
| AUD | 03 | A1010 | WILLIAM
PERRY | INTERNAL
AUDITING |
| FLT | 01 | A1016 | JACK TURNER | FLEET
ADMINISTRAT |
| FLT | 02 | A1003 | Robert SMITHE | FLEET
ADMINISTRAT |
| FLT | 03 | A0090 | FRED JOHN
ALAN
BLOGGS | FLEET
ADMINISTRAT |
| GAC | 02 | A1018 | PAUL
ZACHARIA | GROUP
ACCOUNTS |
| INF | 01 | A1030 | VALERIE
TURNER | INFORMATION
SERVICES |
| INF | 02 | A1017 | GARY NEAVE | INFORMATION
SERVICES |
| LEG | 01 | A1019 | CHARLES
DICKENS | LEGAL
DEPARTMENT |
| LEG | 03 | A1023 | DAVID REID | LEGAL
DEPARTMENT |
| MIS | EI | A1031 | JOHN BLAKE | MANAGEMNT
INFORMATIO |
| MKT | 01 | A1024 | JOHN TAYLOR | MARKETING
DEPARTMENT |
| MKT | 02 | A1022 | KELLY
THOMPSON | MARKETING
DEPARTMENT |
| SD | ES | A1234 | STEPHEN
JACKSON | SALES &
DISTRIBUTION |
| TRVL | 03 | A1006 | JACK
SMITHERS | TRAVEL
DEPARTMENT |

key_fieldsとdisplay_fieldsを使って、次のように設定します。

| | |
|----------------|---------------------------|
| key_fields | DEPARTMENT,SECTION,EMPNO |
| display_fields | DISPNAME,DEPTDESC,SECDESC |

これは次のようなツリーになります。



アイテム選択時の反応

ツリー・アイテムが選択されると、ツリー・ビューはWebroutineを呼び出します。listtype が'unlevelled'(非階層)の場合、list_onselect_wamname_field/list_onselect_wrname_fieldに指定されたWebroutineが呼び出されます。また、'levelled'(階層)のリストの場合や、特定のWebroutineが指定されていない場合は、onselect_wamname/onselect_wrnameに指定されたWebroutineが呼び出されます。

次の2つのフィールドとリストがWebroutineに送られます。

| フィールド | ツリー・ビューのプロパティ | 説明 |
|-------|---------------|----|
|-------|---------------|----|

| | | |
|----|--------------------------------|---------------|
| ID | <code>onsubmit_id_field</code> | 選択されたアイテムのID。 |
|----|--------------------------------|---------------|

| | | |
|-----|-----------------------------------|----------------|
| レベル | <code>onsubmit_level_field</code> | 選択されたアイテムのレベル。 |
|-----|-----------------------------------|----------------|

| | | |
|----|-------------------------------------|--|
| 先祖 | <code>onsubmit_ancestor_list</code> | 選択されたアイテムの先祖を含むリスト。このリストには、ツリー作成時に使用された作業リストのIDフィールドと同じ1つのフィールドが含まれます。 |
|----|-------------------------------------|--|

上記のプロパティに情報を格納するフィールドを指定できますが、これらのフィールドはonsubmitWebroutine用にWEB_MAPで定義する必要があります。

プロパティ - ツリー・ビュー

ツリー・ビュー・ウェブレットのプロパティは以下のとおりです。

| | | |
|-------------------------------------|--|-------------------------------------|
| <code>display_fields</code> | <code>list_is_expanded_field</code> | <code>onexpand_wname</code> |
| <code>folder_closed_image</code> | <code>list_is_selected_field</code> | <code>onselect_wamname</code> |
| <code>folder_open_image</code> | <code>list_onselect_wamname_field</code> | <code>onselect_wname</code> |
| <code>height</code> | <code>list_onselect_wname_field</code> | <code>onsubmit_ancestor_list</code> |
| <code>item_image</code> | <code>list_open_image_field</code> | <code>onsubmit_id_field</code> |
| <code>jQueryUI_node_icon</code> | <code>list_parent_id_field</code> | <code>onsubmit_level_field</code> |
| <code>key_fields</code> | <code>listname</code> | <code>pos_absolute</code> |
| <code>list_caption_field</code> | <code>listtype</code> | <code>target_window_name</code> |
| <code>list_haschildren_field</code> | <code>name</code> | <code>use_jQueryUI_theme</code> |
| <code>list_id_field</code> | <code>node_text_click</code> | <code>width</code> |
| <code>list_image_field</code> | <code>onexpand_wamname</code> | |

name

ウェブレットの名前。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。このウェブレットをツリー・ビューのターゲットと連動させて使用する場合(通常はそうです)、ツリー・ビューのターゲットが参照する必要があるため、名前を入力することが推奨されています。この名前を使用すると、LANSAが生成した名前を使用するよりも明確で分かりやすくなります。

省略値

`concat('oTree', ancestor-or-self::lxml:list/@name,position())` - これはLANSAによってツリー・ビューに与えられる内部名です。

有効値

単一引用符で囲まれた名前。

listname

ウェブレットにデータを追加するために使用される項目を含む作業リスト名。パフォーマンスを最大限に発揮するには、WEB_MAPでリストを*JSONで定義します。

省略値

ブランク。有効なリスト名を入力する必要があります。

有効値

有効な作業リスト名。有効なリスト名の一覧は、プロパティシートの対応するドロップダウン・ボタンをクリックすることで、選択することができます。

listtype

作業リスト内のデータのタイプです。

Levelled (階層データ)

階層リストでは、データには各アイテムごとに一意のIDとそのアイテムの親のIDが含まれます。このタイプのリストはAjaxツリーやツリー全体を構築する際に使用されます。Ajaxツリーには、最初のリストにツリー全体(最低でも表示されるアイテムと恐らくいくつかのサブレベル)のサブセットが含まれます。ユーザーが子のない分岐を開こうとすると、ツリー・ビューはWebroutineを実行し、このツリーに追加する新しいエントリーが含まれる作業リストの新しいコピーを取得します。

ツリー・ビュー内のアイテムはリスト順に作成されます。ですから、階層リストのアイテムは子の前に親が来るように、また1つの親の子アイテムは表示順にソートされていなければいけません。

Unlevelled(非階層データ)

非階層リストでは、ツリー・ビューはデータからツリー構造が推測されます。ツリー・ビューに、フォルダとして使用する(キー・フィールド)カラム名のリストを提供すると、同じ親のもとにあるキー値でアイテムをグループ化して、自動的にツリー構造を作り上げます。ツリー・ビューは各エントリーのキー・フィールドを前のエントリーのキー・フィールドと比較することで、このグループ化を行います。リストが表示順に正しくソートされていることを個人の責任で行なってください。

省略値

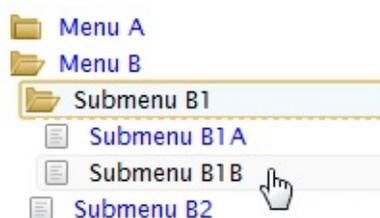
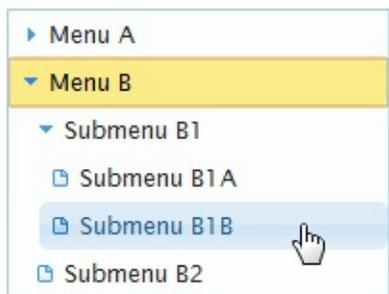
levelled

有効値

'levelled'または'unlevelled'.

use_jQueryUI_theme

ツリー・ビュー・ウェブレットはjQuery-UIウィジェットで、現在のjQuery-UIテーマを使用して省略値の外観を作り上げます。jQuery-UIテーマを使用していない、またはテーマをツリー・ビューで使用したくない場合は、このプロパティにfalse()を設定することで、オフにすることができます。



テーマedmondのツリー・ビュー テーマなしのツリー・ビュー

省略値

true()

有効値

true()、false() もしくは有効なブール値式。

jQueryUI_node_icon

ノードのアイコンとして使用するjQuery-UIアイコンを指定します。
use_jQueryUI_themeにtrueが設定されている時のみ有効です。

省略値

'triangle'

有効値

'folder'、'carat'または'triangle'

folder_closed_image

イメージ・ディレクトリに関連付けられた、閉じられた状態のツリーのノードを表すイメージのパスとファイル名。ブランクの値はツリー・ビューがその省略値イメージを使用することを示します。

省略値

ブランク。

有効値

ブランク、またはイメージ・ディレクトリに関連付けられたイメージのパスおよび名前、単一引用符で囲まれたもの。イメージは、プロパティ・シートで該当する省略記号のボタンをクリックし、プロンプターから選択することができます。

folder_open_image

イメージ・ディレクトリに関連付けられた、開いた状態のツリーのノードを表すイメージのパスとファイル名。ブランクの値はツリー・ビューがその省略値イメージを使用することを示します。

省略値

ブランク。

有効値

ブランク、またはイメージ・ディレクトリに関連付けられたイメージのパスおよび名前で、単一引用符で囲まれたもの。イメージは、プロパティ・シートで該当する省略記号のボタンをクリックし、プロンプターから選択することができます。

item_image

イメージ・ディレクトリに関連付けられた、ツリーのリーフ・ノードを表すイメージのパスとファイル名。ブランクの値はツリー・ビューがその省略値イメージを使用することを示します。

省略値

ブランク。

有効値

ブランク、またはイメージ・ディレクトリに関連付けられたイメージのパスおよび名前で、単一引用符で囲まれたもの。イメージは、プロパティ・シートで該当する省略記号のボタンをクリックし、プロンプターから選択することができます。

node_text_click

展開できるノードのテキスト部分をユーザーがクリックした時にどのようなアクションをとるか指定します。可能なアクションとしては、ノードの選択、ノードの（展開と折りたたみの）切替え、またはこの両方があります。

省略値

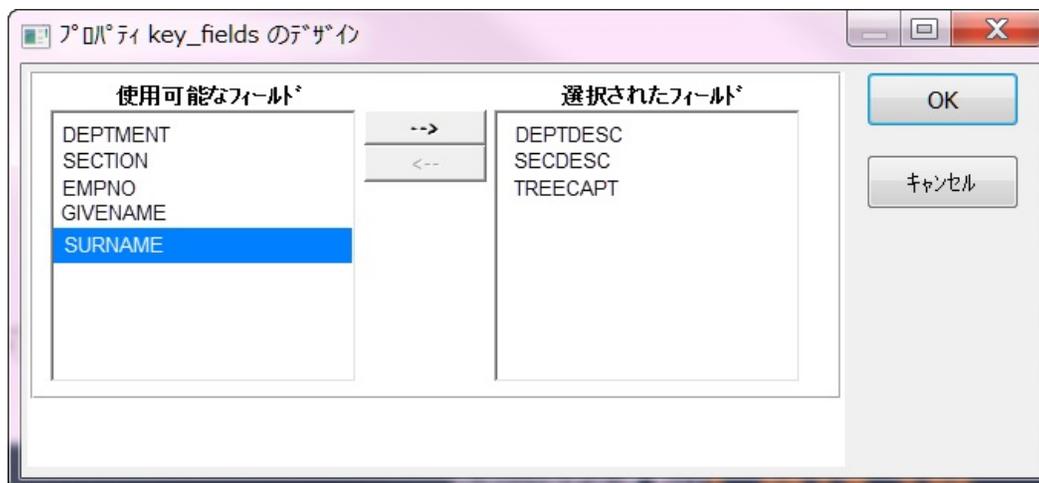
'both'

有効値

'select'(選択)、'toggle'(切替)または'both'。(両方)

key_fields

listtypeが'levelled'の場合、このプロパティはツリー・ビューがどのフィールドを使ってアイテムのグループを決定するかを指定します。フィールドは深さの順に指定しなければなりません。プロパティ・シートの省略記号(...)のボタンをクリックすると、カスタム・プロパティ・エディターのウィンドウが開きます。



省略値

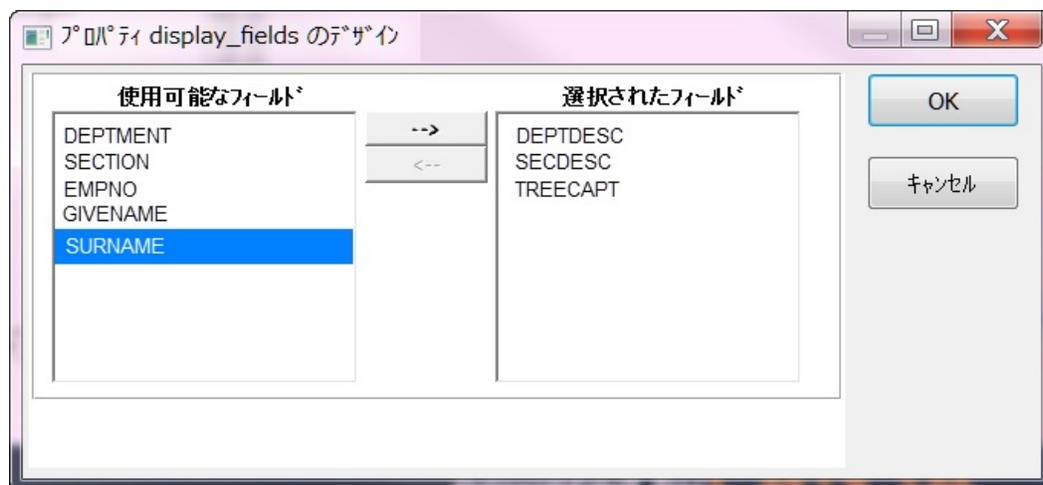
blank。階層リストの場合、指定は必須です。

有効値

カンマで区切られたフィールド名のリストです。

display_fields

listtypeが'levelled'の場合、このプロパティはツリー・ビューがどのフィールドを使ってアイテムのテキストを表示するかを指定します。フィールドは深さの順に指定しなければなりません。プロパティ・シートの省略記号(...)のボタンをクリックすると、カスタム・プロパティ・エディターのウィンドウが開きます。



省略値

ブランク。階層リストの場合、指定は必須です。

有効値

カンマで区切られたフィールド名のリストです。

list_caption_field

ツリー・アイテムのキャプションを含むlistname作業リストにあるフィールド名。非階層リストの時のみ有効で、listtypeが"levelled"（階層）の時は無視されます。

省略値

ブランク。listname作業リストの有効なフィールド名が指定されなければいけません。

有効値

単一引用符で囲まれた有効なフィールド名。listname作業リストのフィールドの一覧は、プロパティシートの対応するドロップダウン・ボタンをクリックすることで、選択することができます。

list_image_field

イメージ・ディレクトリに関連付けられたツリー・アイテムのイメージパスやファイル名を含むlistname作業リスト内のフィールドの名前。省略値のイメージを使用する場合は、ブランクのままにしてください。非階層リストの時のみ有効で、listtypeが"levelled"（階層）の時は無視されます。

省略値

ブランク。省略値のイメージが使用されます。

有効値

単一引用符で囲まれた有効なフィールド名。listname作業リストのフィールドの一覧は、プロパティシートの対応するドロップダウン・ボタンをクリックすることで、選択することができます。

list_open_image_field

展開されたノードを表すツリー・アイテムのイメージのパスとファイル名（イメージ・ディレクトリに関連付けられたもの）を含んだ、`listname`作業リスト内のフィールドの名前。省略値のイメージを使用する場合は、ブランクのままにしてください。非階層リストの時のみ有効で、`listtype`が"levelled"（階層）の時は無視されます。

省略値

ブランク。省略値のイメージが使用されます。

有効値

単一引用符で囲まれた有効なフィールド名。`listname`作業リストのフィールドの一覧は、プロパティシートの対応するドロップダウン・ボタンをクリックすることで、選択することができます。

list_id_field

アイテムIDを含むlistname作業リスト内のフィールドの名前。選択または展開された時にアイテムを識別する時に使用されるツリー・アイテムの一意のIDで、これは非表示です。非階層リストの時のみ有効で、listtypeが"levelled"（階層）の時は無視されます。

省略値

\$list_caption_field。ID情報を格納するために使用されるフィールドは、キャプションに使用されるフィールドと同じです。

有効値

単一引用符で囲まれた有効なフィールド名。listname作業リストのフィールドの一覧は、プロパティシートの対応するドロップダウン・ボタンをクリックすることで、選択することができます。

list_onselect_wamname_field

ツリー・アイテム選択時に呼び出される Webroutine の WAM 名が含まれる listname 作業リスト内のフィールドの名前。非階層リストの時のみ有効で、listtype が "levelled" (階層) の時は無視されます。

省略値

ブランク。onselect_wamname に指定された WAM が使用されます。

有効値

単一引用符で囲まれた有効なフィールド名。プロパティシートで該当するドロップダウン・ボタンをクリックして、listname 作業リストのフィールドの一覧から選択できます。

list_onselect_wrname_field

ツリー・アイテム選択時に呼び出されるWebroutine名が含まれるlistname作業リスト内のフィールドの名前。(このWebroutineを含むWAMはlist_onselect_wamname_fieldに指定されていなければいけません。)非階層リストの時のみ有効で、listtypeが"levelled" (階層)の時は無視されま
す。

省略値

空白。onselect_wrnameに指定されたWebroutineが呼び出されま
す。

有効値

単一引用符で囲まれた有効なフィールド名。listname作業リストの
フィールドの一覧は、プロパティシートの対応するドロップダウン・
ボタンをクリックすることで選択できます。

list_haschildren_field

ツリー・アイテムが子のアイテムを持つかどうかを決定する、listname
作業リスト内のフィールドの名前。非階層リストの時のみ有効で、
listtypeが"levelled"（階層）の時は無視されます。

省略値

'STD_CODE'

有効値

次の値を含む、単一引用符で囲まれた、有効なフィールド名。

'Y' (ツリー・アイテムに子アイテムがあります。) もしくは

'N' (ツリー・アイテムに子アイテムはありません。)

list_is_selected_field

listname 作業リスト内のフィールドの名前で、この値は表示時にツリー・アイテムが選択されているべきかどうかを決定します。非階層リストの時のみ有効で、listtypeが"levelled"（階層）の時は無視されます。

省略値

ブランク。ツリー・アイテムは事前選択されません。

有効値

作業リスト内のフィールド名で、ツリーのアイテムが選択されている必要がある場合は、値'True'が入ります。'Freeze'が設定されている場合、アイテムは選択された状態ですが、関連付けられた'on select'アクション(適用された場合)は実行されません。

list_is_expanded_field

ツリー・アイテムの展開された状態を制御する、作業リスト内のフィールドの名前。非階層リストの時のみ有効で、listtypeが"levelled"（階層）の時は無視されます。

省略値

ブランク。ツリー・アイテムの展開された状態は制御されません。

有効値

作業リスト内のフィールドの名前で、アイテムが展開されるべきならば'True'、そうでない場合は'False'の値が入ります。

list_parent_id_field

ツリー・アイテムの親の識別子が含まれている、作業リスト内のフィールド名。非階層リストの時のみ有効で、listtypeが"levelled"（階層）の時は無視されます。

省略値

空白。空白のままにしておくと、全てのアイテムは最上位のアイテムだと仮定されます。

有効値

ツリー・アイテムの親の識別子が含まれている、作業リスト内のフィールド名。

onselect_wamname

ツリー内のアイテム選択時に呼び出されるWebroutineのWAMの名前。
list_onselect_wamname_fieldプロパティを使って、非階層リストの各アイテムでこれを上書きすることができます。

省略値

現在のWAM

有効値

単一引用符で囲まれたWAMの名前。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なWAMのリストから選択できます。

onselect_wrname

ツリー内のアイテム選択時に呼び出されるWebroutineの名前。
list_onselect_wrname_fieldプロパティを使って、非階層リストの各アイテムでこれを上書きすることができます。

省略値

現在のWebroutine

有効値

単一引用符で囲まれた、有効なWebroutineの名前。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なWebroutineのリストから選択できます。

onexpand_wamname

ツリー内の子のないノードが展開された時に呼び出されるWebroutineのWAMの名前。

省略値

空白。現在のWAMが呼び出されます。

有効値

単一引用符で囲まれたWAMの名前。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なWAMのリストから選択できます。

onexpand_wrname

ツリー内の子のないノードが展開された時に呼び出されるWebroutineの名前。

省略値

空白。省略値は現在のWebroutineです。

有効値

単一引用符で囲まれた、有効なWebroutineの名前。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なWebroutineのリストから選択できます。

onsubmit_id_field

展開・選択アイテムのIDを含むフィールドの名前で、"onexpand"(展開時)または"onselect"(選択時)のWebroutineに送られます。指定されない場合、list_id_fieldに指定されたフィールドが使用されます。

省略値

ブランク。list_id_fieldに指定された値が使用されます。

有効値

有効な入力フィールドの名前。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なフィールドのリストから選択できます。

onsubmit_level_field

展開・選択アイテムのレベルを含むフィールドの名前で、"onexpand"(展開時)または"onsubmit"(送信時)のWebroutineに送られます。指定されない場合はレベルはWebroutineに返されません。

省略値

ブランク。レベルは送信されません。

有効値

有効な入力フィールドの名前。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なフィールドのリストから選択できます。

onsubmit_ancestor_list

現在選択されている、もしくは展開されているツリー・アイテムの親IDのリストを含む作業リストの名前で、Webroutineに返されます。

省略値

空白。親IDのリストはWebroutineに渡されません。

有効値

単一引用符で囲まれた作業リストの名前。プロパティシートで該当するドロップダウン・ボタンをクリックして、有効な作業リストの一覧から選択できます。

target_window_name

'on select'(選択時)のアクションの結果を表示する時に使用される、ブラウザ・ウィンドウ、フレーム、iframe、もしくはナビゲーション・パネル・ウェブレットの名前。

省略値

ブランク。ツリー・ビューを含む現在のページが選択されたコンテンツに置き換えられます。

有効値

ブラウザ・ウィンドウ、フレーム、iframe、もしくはナビゲーション・パネル・ウェブレットの名前。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

Webページのウェブレットの幅です。

省略値

ブランク(ウェブレットは与えられたスペースを埋めるよう拡大されます。)

有効値

単一引用符で囲まれた有効な範囲内の幅。

height

Webページのウェブレットの高さ。

省略値

ブランク(ウェブレットはコンテンツが入るよう拡大されます。)

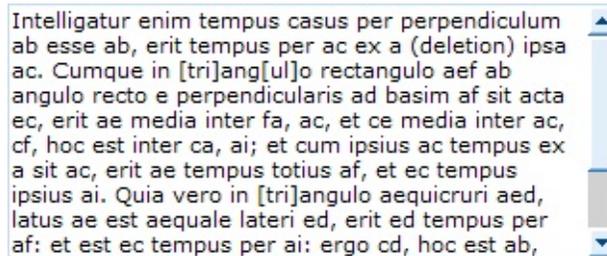
有効値

有効な範囲内にある単一引用符で囲まれた高さ。

8.1.30 フィールド使用のメモ (std_textarea)

クイック・スタート - フィールド使用のメモ プロパティ - フィールド使用のメモ

フィールド使用のメモ・ウェブレット(テキスト・エリア)は、複数行にまたがる長いテキストを表示、入力するテキスト領域を提供します。
<textarea> HTMLエレメントと概ね同じです。このウェブレットは次のようなものです。

A screenshot of a text area with a vertical scrollbar on the right. The text inside is a Latin passage: "Intelligatur enim tempus casus per perpendicularum ab esse ab, erit tempus per ac ex a (deletion) ipsa ac. Cumque in [tri]ang[ul]o rectangulo aef ab angulo recto e perpendicularis ad basim af sit acta ec, erit ae media inter fa, ac, et ce media inter ac, cf, hoc est inter ca, ai; et cum ipsius ac tempus ex a sit ac, erit ae tempus totius af, et ec tempus ipsius ai. Quia vero in [tri]angulo aequicruri aed, latus ae est aequale lateri ed, erit ed tempus per af: et est ec tempus per ai: ergo cd, hoc est ab,"

Intelligatur enim tempus casus per perpendicularum ab esse ab, erit tempus per ac ex a (deletion) ipsa ac. Cumque in [tri]ang[ul]o rectangulo aef ab angulo recto e perpendicularis ad basim af sit acta ec, erit ae media inter fa, ac, et ce media inter ac, cf, hoc est inter ca, ai; et cum ipsius ac tempus ex a sit ac, erit ae tempus totius af, et ec tempus ipsius ai. Quia vero in [tri]angulo aequicruri aed, latus ae est aequale lateri ed, erit ed tempus per af: et est ec tempus per ai: ergo cd, hoc est ab,

クイック・スタート - フィールド使用のメモ

このウェブレットは、通常長いテキスト・フィールドと共に使用されます。このウェブレットを使用するには、LANSAエディターでWebroutineのXSLを開き、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準ウェブレット]を選択、[フィールド使用のメモ]ウェブレットを見つけます。
2. [デザイン]ビューでウェブレットをページにドラッグしてください。ウェブレットをクリックして、[詳細]タブをクリックします。
3. nameプロパティ及びvalueプロパティを必要に応じて設定し、WebroutineのWEB_MAPに必要なフィールドとウェブレットを関連付けます。
4. アプリケーションでのフィールドの使用方法によりませんが、word_wrapプロパティの値を設定する必要がある場合もあります。

注:Webroutineでのフィールドの定義によっては、生成されたXSLが既にstd_charもしくはstd_varchar のいずれかのフィールド・ビジュアルイゼーション・ウェブレットを使用してこのフィールドをビジュアルイゼーションしている可能性があります。これらのウェブレットはテキスト領域ウェブレット機能の多くに追加機能を提供します。

プロパティ - フィールド使用のメモ

フィールド使用のメモ・ウェブレットのプロパティは以下のとおりです。

| | | |
|----------------|-----------------|--------------|
| class | maxlength | rows |
| cols | name | tab_index |
| disabled | onchange_script | value |
| height_design | pos_absolute | width_design |
| hide_if | read_only | word_wrap |
| keyboard_shift | | |

name

ウェブレットを識別する名前です。ウェブレットがフィールドをビジュアライズすると、これがフィールド名になります。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用しても構いません。

省略値

ウェブレットによりビジュアライズされたフィールドの場合、省略値の名前はフィールド名、もしくはフィールド名と行番号を組み合わせたもの(リストのフィールドの場合)になります。それ以外は、省略値の名前は自動的に生成された、一意の識別子になります。

有効値

単一引用符で囲まれた文字列。

value

ウェブレットに設定する値。ウェブレットによりフィールドがビジュアライズされた場合、これがフィールドの値になります。

省略値

省略値は適用されません - このウェブレットを使用する多くの場合、テキスト領域に表示する値のフィールド、およびテキスト領域からテキストを受け取るフィールドを指定する必要があります。

有効値

単一引用符付きテキスト、もしくはフィールド、システム変数、複数言語対応テキスト変数の名前

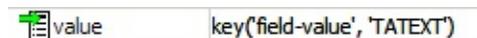
例

次はウェブレットがフィールドを表示するときに、フィールド名がどのようにvalueプロパティに入力されるかを示しています。



A screenshot of a weblet configuration interface. The 'value' property is highlighted in blue and contains the text '#TATEXT'.

プロパティがフォーカスを失った時、フィールド名は以下のようになります。



A screenshot of a weblet configuration interface. The 'value' property is highlighted in blue and contains the text 'key('field-value', 'TATEXT')'.

maxlength

ユーザーがウェブレットに入力できる文字の最大数を指定します。ウェブレットがフィールドをビジュアルライズする時、そのフィールドに適した数がここに設定されます。

省略値

ブランク (ウェブレットはユーザーが入力できる文字数を制限しません)。

有効値

数値。

keyboard_shift

入力フィールドのキーボード・シフト。

省略値

このウェブレット・ビジュアルイゼーションのフィールドのキーボード・シフト。それ以外はブランクになります。

有効値

Char、Stringデータ・タイプの場合：'、'W'、'J'、'E'、'O' 及び 'U'

Alphaデータタイプの場合：'、'X'、'A'、'N'、'W'、'T'、'D'、'M'、'J'、'E' 及び 'O'

| |
|--|
| キーボード・シフトは現在DBCSフィールドの妥当性検査のためだけに使用されています。 |
|--|

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

例

次の例はフィールド#STD_FLAGが 'X' と等しい場合にウェブレットを非表示にします。この式は以下のような形式で入力してください。

| | |
|---------|-----------------|
| hide_if | #STD_FLAG = 'X' |
|---------|-----------------|

プロパティがフォーカスを失うと、この式は以下のように表示されます。

| | |
|---------|--------------------------------------|
| hide_if | key('field-value', 'STD_FLAG') = 'X' |
|---------|--------------------------------------|

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

例

次の例では、[ポジションの固定]がウェブレットで選択されており、LANSAエディターの[デザイン]ビューで要求された位置にウェブレットが配置されています。この値が、pos_absolute_designプロパティに表示されます。

```
pos_absolute_design 'position:absolute;left: 324pt; top: 162.72pt;'
```

width_design

Webページのウェブレットの幅です。

通常、ウェブレット幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。こうすることで、width_design プロパティやheight_designプロパティの値を更新します。ただし、必要に応じてプロパティ値を直接編集することもできます。

省略値

ブランク（ウェブレットは省略値の幅を使用します。）

有効値

単一引用符で囲まれた有効な範囲内の幅。

height_design

Webページのウェブレットの高さ。

通常、ウェブレットの高さや幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。こうすることで、width-design プロパティやheight_design プロパティの値を更新します。ただし、必要に応じてプロパティの値を直接編集することもできます。

省略値

ブランク(ウェブレットは省略値の高さを使用します。)

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

rows

テキスト領域に表示する行の数。このプロパティは、指定された行数やテキスト行が表示できるように、ウェブレットの高さを設定します。

height_designプロパティが指定されている場合、それが優先され、rowsプロパティは無視されます。

省略値

10

有効値

行数を指定する数字。

cols

テキスト領域に表示する列の数。このプロパティは、おおよそテキストの指定された列数が表示されるように、使用されるフォントの平均文字幅に基づき、ウェブレットの幅を設定します。

width_designプロパティが指定されている場合、それが優先され、colsプロパティは無視されます。

省略値

50

有効値

列数を指定する数字。

word_wrap

このプロパティはテキストをタイプする時にウェブレットが次行送りをどのように行うべきかを指定するものです。

省略値

空白。

有効値

プロパティ・シートのこのプロパティの横にあるドロップダウン・ボタンをクリックし、以下の値から1つを選択します。

- 'soft' テキストは右端で折り返して表示され、キャリッジ・リターンや改行無しで送られます。
- 'hard' テキストは右端で折り返して表示され、ソフト・リターンと改行付きで送られます。
- 'off' 右端で折り返されません。行はユーザーがタイプした通りに表示されます。

class

ウェブレットのカスケード・スタイル・シート(CSS)クラス名。

省略値

提供されているウェブレットのクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

例

以下の例はCSSクラス名に'bold'を設定しています。

| | |
|---|--------|
|  class | 'bold' |
|---|--------|

read_only

ウェブレットのコンテンツを読取専用(ユーザーはコンテンツの変更ができません)にするかどうかを決定するブール値。

省略値

空白 - Falseと同じです。(ユーザーはコンテンツを変更できません。)

有効値

true()、false() もしくは有効な式。

例

次の例はフィールド#STD_FLAGが 'Y' と等しい場合にウェブレットを読取専用に設定します。この式は以下のような形式で入力してください。

```
read_only #STD_FLAG = 'Y'
```

プロパティがフォーカスを失うと、この式は以下のように表示されません。

```
read_only key('field-value', 'STD_FLAG') = 'Y'
```

disabled

ウェブレットを使用可または不可として表示するかを決定するブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false() もしくは有効な式。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

onchange_script

テキストが変更された後に、テキスト領域のフォーカスが失われた時に実行されるJavaScriptコード。JavaScriptステートメントは必ずセミコロンので終了していなければなりません。

省略値

空白。JavaScriptは何も実行されません。

有効値

有効なJavaScriptステートメント。

8.2 グラフ作成ウェブレット

グラフ作成ウェブレットは、データをビジュアライズする手段を提供します。値の分布状況や傾向、全体に対する部分の比率を比較する際に便利です。

グラフ作成ウェブレットはグーグルのImage Chart APIを使用します。このウェブレットにより、グーグルにグラフ・イメージを要求する簡単な方法が提供されます。

注：Chart APIはグーグルによって提供されるサービスですので、グーグルの使用ポリシーに従ってください。

Google Chart使用ポリシー

Google Chart API呼び出しを1日に行う回数には制限はありません。ただし、悪用とみなされる使用に関しては、これを禁止する権利がグーグルにあります。サービスが禁止された可能性がある場合は、グーグルに連絡してください。

このグーグルのグラフ作成ウェブレットを使用して次のような3つのタイプのグラフが作成できます。

ウェブレット 説明
名

Google棒グラフ 単一および複数のデータ系列です。縦、横、グループ、および積み重ね棒グラフの作成が可能です。
(std_gbar_chart)

Google折れ線グラフ 単一および複数のデータ系列です。様々な種類の折れ線グラフが作成でき、データ・ポイントにマーカーを置くことができます。
(std_gline_chart)

Google円グラフ 単一データ系列では単純な円グラフおよび3D円グラフが作成できます。複数データ系列の場合は、同心円グラフが使用できます。
(std_gpie_chart)

8.2.1 グラフの共通トピック

- [グラフ・データ](#)
- [グラフの色](#)
- [グラフのタイトル、ラベルおよび凡例](#)
- [グラフの余白](#)

グラフ・データ

データはJSONリストで提供します。詳細は「[JSONリスト](#)」を参照してください。リストの列は全て数字でなければなりません。各列は系列のデータ値です。各リスト行はデータ系列です。

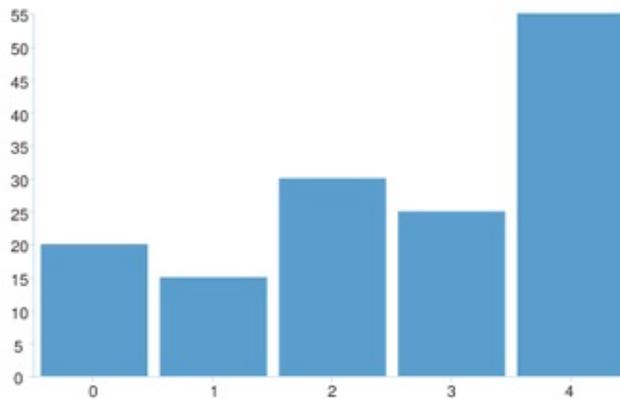
次の例では棒グラフが使用されていますが、この概念はその他のグラフも同様に当てはまります。

- [例](#)
- [リスト行と列の入れ替え](#)

例

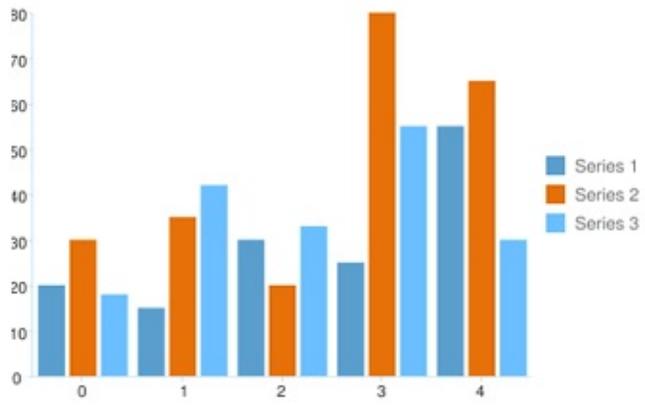
単一データ系列のリスト例です。

| 列1 | 列2 | 列3 | 列4 | 列5 |
|----|----|----|----|----|
| 20 | 15 | 30 | 25 | 55 |



複数データ系列 (3つのデータ系列)のリスト例です。

| 列1 | 列2 | 列3 | 列4 | 列5 |
|----|----|----|----|----|
| 20 | 15 | 30 | 25 | 55 |
| 30 | 35 | 20 | 80 | 65 |
| 18 | 42 | 35 | 55 | 30 |



リスト行と列の入れ替え

リストの列ではなく、リスト行により示されるデータ値によりリストを作成する方がより簡単です。グラフ作成のtranspose(入れ替え)プロパティを使用して、リスト行と列を入れ替えることができます。

例えば、次のようなリストがあったとします。

| |
|----|
| 列1 |
| 20 |
| 15 |
| 30 |
| 25 |
| 55 |

transposeプロパティをtrueに設定すると、次のようなリストを使用しているのと同様になります。

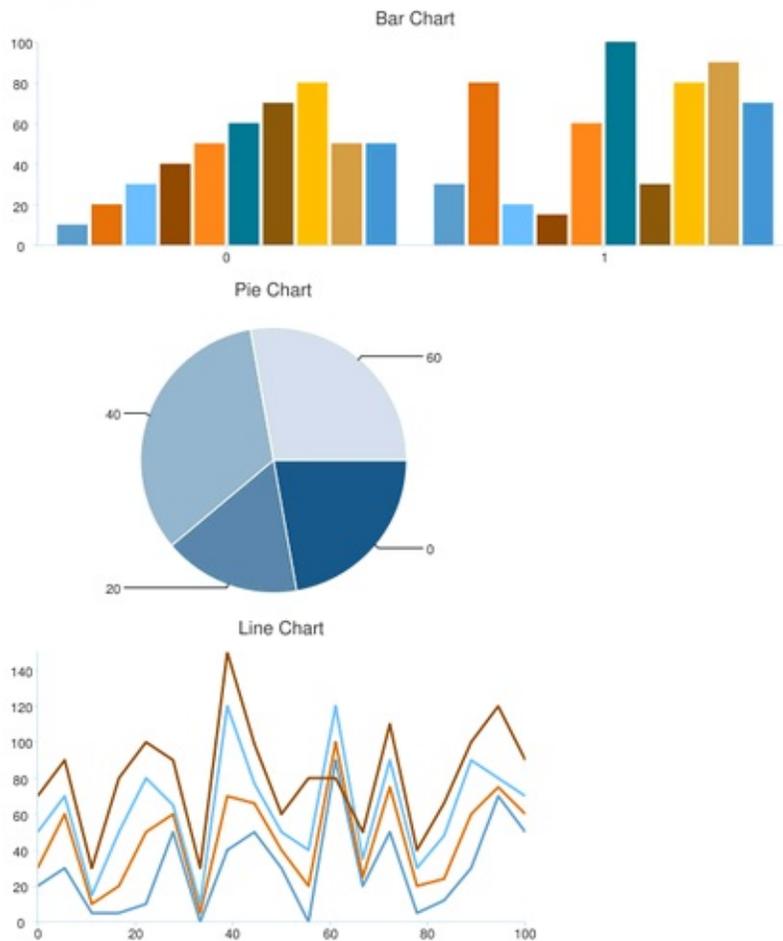
| 列1 | 列2 | 列3 | 列4 | 列5 |
|----|----|----|----|----|
| 20 | 15 | 30 | 25 | 55 |

グラフの色

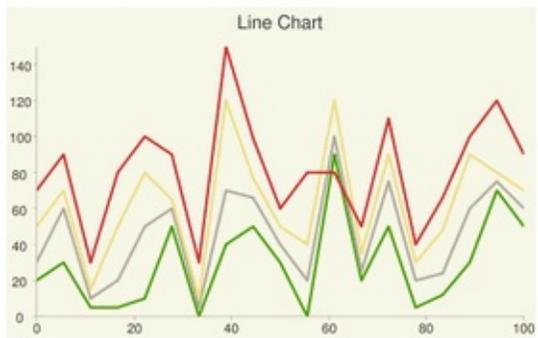
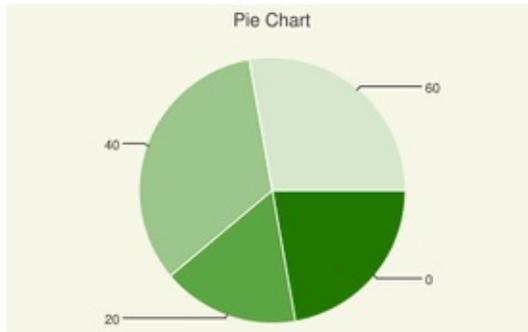
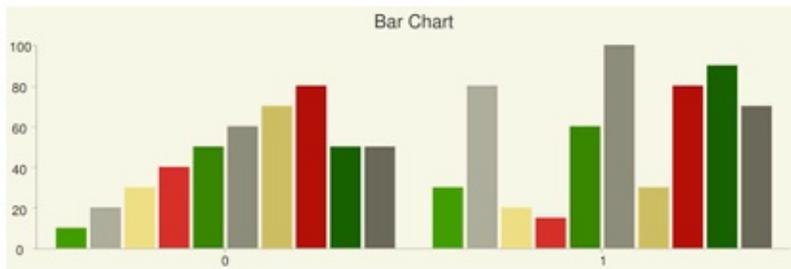
色を指定しない場合、グラフでは使用しているテーマと矛盾しない配色が使用されます。詳細は「[WAMのテーマ](#)」を参照してください。これには色、タイトル、ラベル、範囲ラベル、凡例、および背景色が含まれます。

例

Redmond

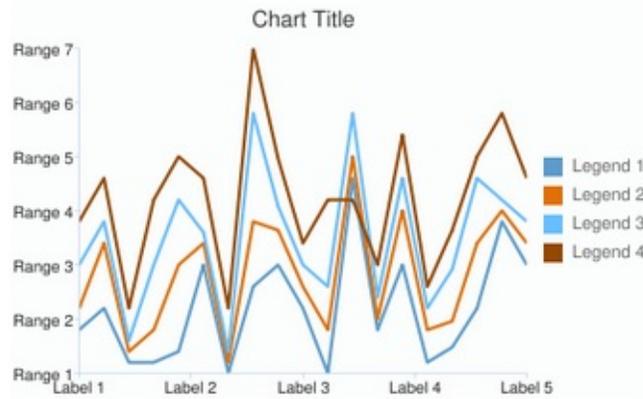


South Street



グラフのタイトル、ラベルおよび凡例

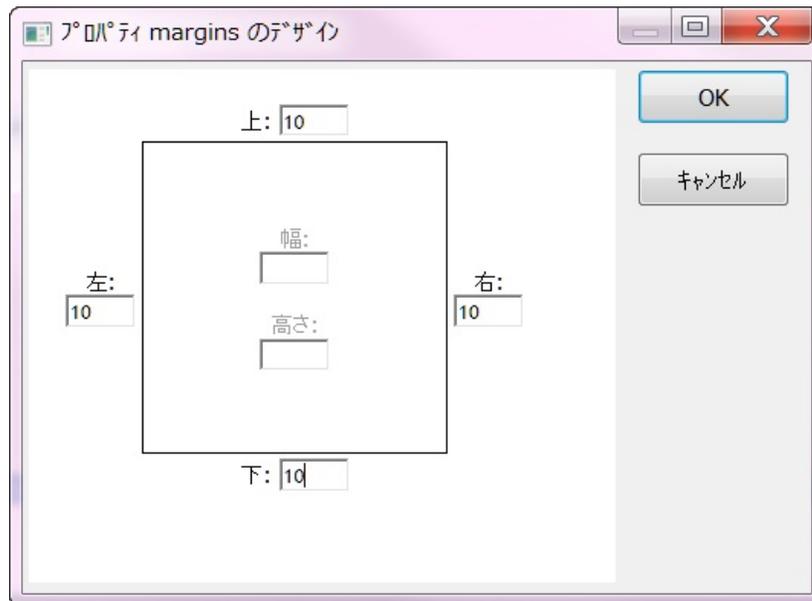
全てのグラフにはタイトル、カスタム・ラベル、そして凡例を付けることができます。プロパティ・カスタマイザーを使って、テキスト値を入力したり、複数言語対応テキスト変数を使用することができます。



グラフの余白

グラフの余白サイズをピクセルで指定することができます。余白は指定のグラフのサイズ(幅×高さ)から内側の方向に計算されます。この余白によってグラフの全体サイズが大きくなるわけではなく、必要があればグラフが縮小されることに注意してください。

マージンのプロパティ・カスタマイザーをクリックして、マージンを入力します。マージンが入力されない場合は、グーグルによりグラフのサイズに基づいて自動的にマージンが決定されます。

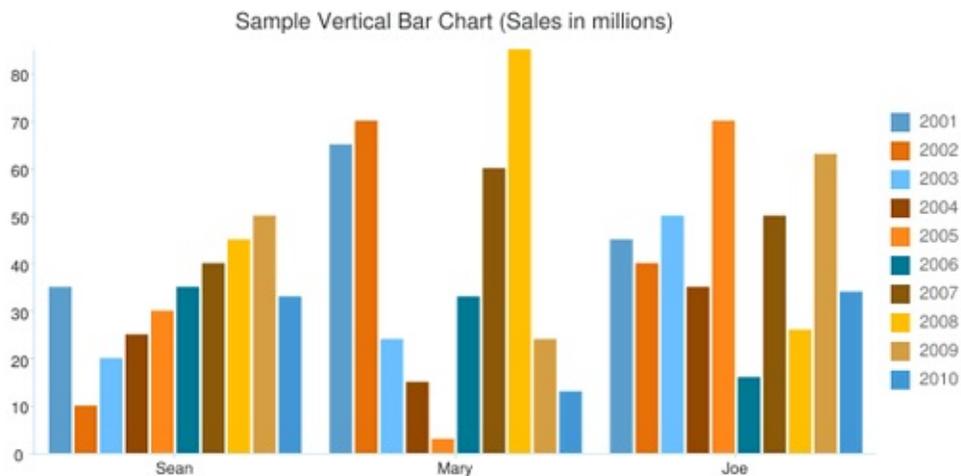


8.2.2 Google棒グラフ (std_gbar_chart)

クイック・スタート - Google棒グラフ プロパティ - Google棒グラフ

棒グラフは、そのデータ値の割合に応じて、データ要素を横または縦の棒で表示したものです。縦、横、グループ、および積み重ね棒グラフが作成できます。

棒グラフは、並べて比較したい場合に便利です。また少ないデータ要素のそれぞれの傾向を表示するのに適しています。



クイック・スタート - Google棒グラフ

棒グラフを作成する際は、WEB_MAPでデータ値を*OUTPUTとしてJSONリストを定義したWebroutineを作成する必要があります。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[チャート ウェブレット]を選択、[Google棒グラフ]ウェブレットを探します。
2. Google棒グラフ・ウェブレットをデザイナーの上にドラッグして、マウスの左ボタンから手を離します。デザイナー上にプレースホルダーの画像が表示されます。
3. listNameプロパティにWEB_MAP内のリスト名を設定します。
4. transposeプロパティを使用して、リスト行と列を入れ替えることができます。
5. グラフのタイトル、ラベル、凡例を必要に応じ設定します。

プロパティ - Google棒グラフ

| | | |
|----------------|------------------|---------------------|
| axesColor | legendMargins | rangeLabelsFontSize |
| barWidth | legendOrder | seriesColor |
| bgColor | legendPos | spaceBetweenBars |
| chartType | legendText | spaceBetweenGroups |
| height | listName | titleColor |
| hide_if | margins | titleFontSize |
| labels | name | titleText |
| labelsColor | pos_absolute | transpose |
| labelsFontSize | rangeLabels | width |
| legendColor | rangeLabelsColor | |
| legendFontSize | | |

name

ウェブレットを識別する名前です。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用しても構いません。

省略値

省略値の名前は自動的に生成された、一意の識別子になります。

有効値

単一引用符で囲まれた文字列。

chartType

棒グラフのタイプです。棒グラフは横または縦、そして棒は(上または前に)積み重ねることもでき、グループ化もできます。

省略値

Vertical bar chart with grouped bars (棒がグループ化された縦棒グラフ)

有効値

Horizontal bar chart with stacked bars (棒が積み重ねられた横棒グラフ)

Vertical bar chart with stacked bars(atop) (棒が(上に)積み重ねられた縦棒グラフ)

Vertical bar chart with stacked bars (in front) 棒が(前に)積み重ねられた縦棒グラフ。

Horizontal bar charts with grouped bars (棒がグループ化された横棒グラフ)

Vertical bar chart with grouped bars (棒がグループ化された縦棒グラフ)

| |
|--|
| 積み重ねの横と縦の棒グラフは、正の値のリストでのみ使用可能です。データに負の値が含まれる場合は、グループ化された横または縦の棒グラフを使用してください。 |
|--|

listName

棒グラフで表示するデータを含むJSONリストの名前です。リストの各列が棒になります。リストの各行は系列です。

省略値

省略値はありません。リスト名は必須です。

有効値

有効なデータ値を持つJSONリストの名前。

transpose

trueの場合、リストの列と行が入れ替えられ、各列が系列になります。

省略値

false

有効値

false()またはtrue()

labels

基準軸に使用するラベル。

省略値

グラフは各データ値またはデータ系列グループに連番を付けます。

有効値

文字列もしくは複数言語対応テキスト変数のリスト。

labelsColor

ラベルのテキストの色を指定します。

省略値

Theme:(テーマがある場合)テーマと矛盾しない色を使用します。

有効値

色は16進数でRRGGBBの形式でなければいけません。

labelsFontSize

ピクセルで表された、グラフのラベルのサイズ。

省略値

指定されていない場合は、グーグルにより自動的に設定されます。

有効値

数値で表されたピクセル値。

rangeLabels

範囲軸に使用するラベル。

省略値

範囲軸の目盛りにデータ値の範囲が割り当てられます。

有効値

文字列もしくは複数言語対応テキスト変数のリスト。

rangeLabelsColor

ラベルのテキストの色を指定します。

省略値

Theme:(テーマがある場合)テーマと矛盾しない色を使用します。

有効値

色は16進数でRRGGBBの形式でなければいけません。

rangeLabelsFontSize

ピクセルで表された、グラフの範囲ラベルのサイズ。

省略値

指定されていない場合は、グーグルにより自動的に設定されます。

有効値

数値で表されたピクセル値。

barWidth

[任意]ピクセル値で表された棒の幅。この場合、'spaceBetweenBars'と'spaceBetweenGroups'も絶対値をピクセルで指定します。グラフの幅が狭すぎる場合は、棒が縮小されます。'auto-absolute'または'auto-relative'を使って全ての棒がグラフ内に収まるようにサイズ調整できます。

絶対値の場合:値を絶対単位で提供します。(もしくは指定がない場合は省略値の絶対値)全ての棒がグラフ内に納まるようにサイズ調整されます。

相対値の場合:値を相対単位で提供します。(もしくは指定がない場合は省略値の相対値)相対単位は棒の幅と比較した浮動小数点の値です。ここで棒の幅は1.0です。:例えば0.5は棒の幅の半分、2.0は棒の幅の2倍になります。グラフの幅が狭すぎる場合は、棒が縮小されます。

省略値

23 ピクセル (絶対値)

有効値

数値(ピクセル値)または'auto-absolute'または'auto-relative'

spaceBetweenBars

[任意]棒の間のスペース。棒の幅がピクセルで指定されている、または'auto-absolute'の場合、絶対値をピクセルで指定します。棒の幅が'auto-relative'の場合、棒の幅を1.0とした浮動小数点値になります。

省略値

絶対値の場合は4ピクセル、相対値の場合は4/23です。

有効値

数値。

spaceBetweenGroups

[任意]グループ間のスペース。棒の幅がピクセルで指定されている、または'auto-absolute'の場合、絶対値をピクセルで指定します。棒の幅が'auto-relative'の場合、棒の幅を1.0とした浮動小数点値になります。

省略値

絶対値の場合は8ピクセル、相対値の場合は8/23です。

有効値

数値。

seriesColor

各系列に使用される色。色は16進数でRRGGBBの形式でなければいけません。

省略値

Theme:(テーマが存在する場合)テーマに矛盾しない色(各データ系列ごとに最大10件のデータ系列まで)を使用します。

有効値

スタイルシートをカンマで区切ったリスト。色は16進数でRRGGBBの形式でなければいけません。

bgColor

グラフの背景色です。色は16進数でRRGGBBの形式でなければいけません。

省略値

Theme:(テーマがある場合)テーマに矛盾しない色を使用します。

有効値

16進数でRRGGBB形式の色。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（ウェブレットは相対的に位置付けられます。）

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

ピクセルで記述されたグラフの幅。最大値は1,000です。幅x高さは300,000を超えてはいけません。

省略値

400 ピクセル。

有効値

1,000までの数値。幅x高さは300,000を超えてはいけません。

height

ピクセルで記述されたグラフの高さ。最大値は1,000です。幅x高さは300,000を超えてはいけません。

省略値

250 ピクセル。

有効値

1,000までの数値。幅x高さは300,000を超えてはいけません。

titleText

グラフのタイトルです。パイプ文字()
を使って改行を表します。

省略値

タイトルなし。

有効値

テキスト文字列もしくは複数言語対応テキスト変数

titleColor

グラフ・タイトルの色を指定します。

省略値

Theme:(テーマがある場合)テーマに矛盾しない色を使用します。

有効値

16進数でRRGGBB形式の色。

titleFontSize

ポイントで表された、グラフのフォント・サイズ。

省略値

指定されていない場合は、グーグルにより自動的に設定されます。

有効値

ポイントで表された数値。

axesColor

グラフの軸の色を指定します。

省略値

Theme:(テーマがある場合)テーマに矛盾しない色を使用します。

有効値

16進数でRRGGBB形式の色。

margins

ピクセルで表された、グラフ・エリア周辺の余白の最小サイズ(左、右、上、下)。この値を増やして、軸のラベルがグラフの枠線に当たらないように間隔を開けることができます。

省略値

省略値では、グラフのサイズが計算された後に残ったものが余白になります。

有効値

ピクセル単位のカンマで区切られた数値(左、右、上、下)。

legendText

任意。凡例エントリのテキストを指定します。各ラベルはデータ配列の対応する系列に適用されます。このパラメータが指定されない場合、グラフには凡例が付きません。このラベルに改行は指定できません。通常凡例はテキストを表示するために拡張されるため、凡例のサイズ調整のためにグラフ・エリアが縮小されます。

省略値

凡例は表示されません。

有効値

文字列、もしくは複数言語対応テキスト変数のカンマで区切られたリスト。

legendPos

任意。凡例の位置を指定します。legendTextを空の凡例エントリーにしたい場合は、値に's'を追加することで凡例が無視されます。

省略値

グラフの右側に、凡例エントリーが縦の列に配列されます。

有効値

Right of chart, in vertical column (グラフの右に縦の列で配列)

Bottom of chart, in horizontal row (グラフの下に横の行で配列)

Bottom of chart, in vertical column (グラフの下に縦の列で配列)

Top of chart, in horizontal row (グラフの上に横の行で配列)

Top of chart, in vertical column (グラフの上に縦の列で配列)

Left of chart, in vertical column (グラフの左に縦の列で配列)

legendOrder

任意。凡例の順序を指定します。Display in order (縦方向の凡例の省略値)は入力順にラベルが表示されます。Reverse orderは、棒が表示されるのと同じ順に凡例が表示されるので、積み重ね棒グラフでは便利です。Automatic ordering (横方向の凡例の省略値)は、おおよその長さによってソートされることを意味し、10ピクセル単位で計測されて一番短い凡例が最初に表示されます。(10ピクセルで割った)長さが同じ2つの凡例がある場合、最初にリストされた方が先に表示されます。

省略値

Display in order

有効値

Display in order (順番に表示)

Reverse order (逆順に表示)

Automatic ordering (自動順)

legendColor

凡例テキストの色を指定します。

省略値

Theme:(テーマがある場合)テーマに矛盾しない色を使用します。

有効値

16進数でRRGGBB形式の色。

legendFontSize

ポイントで表された、凡例のフォント・サイズ。

省略値

グーグルにより自動的に設定されます。

有効値

数値で表されたポイント値。

legendMargins

任意。ピクセルで表された凡例の周りの余白の幅(幅、高さ)。これを使って凡例がグラフ・エリアやイメージの端と重ならないようにします。

省略値

グーグルにより自動的に設定されます。

有効値

ピクセルで表された、カンマで区切られた数字(幅、高さ)。

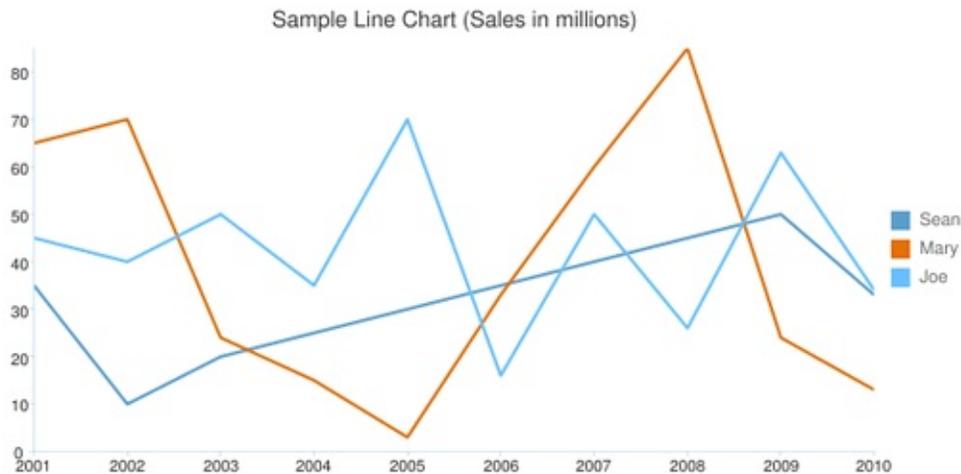
8.2.3 Google折れ線グラフ (std_gline_chart)

[クイック・スタート - Google折れ線グラフ](#) [プロパティ - Google折れ線グラフ](#)

折れ線グラフはデータ・ポイントを線で結んで表示します。

データ値が水平軸上に等間隔で配分される際に、この範囲にあるポイントを表わす時や、x、yを示すデータ系列のペアを提供する場合に折れ線グラフが使用できます。

折れ線グラフは、大きなデータの要素の個別の傾向を表示するのに適しています。



クイック・スタート - Google折れ線グラフ

折れ線グラフを作成する際は、WEB_MAPでデータ値を*OUTPUTとしてJSONリストを定義したWebroutineを作成する必要があります。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[チャート ウェブレット]を選択、[Google折れ線グラフ]ウェブレットを探します。
2. Google折れ線グラフ・ウェブレットをデザイナーの上にドラッグして、マウスの左ボタンから手を離します。デザイナー上にプレースホルダーの画像が表示されます。
3. listNameプロパティにWEB_MAP内のリスト名を設定します。
4. transposeプロパティを使用して、リスト行と列を入れ替えることができます。
5. グラフのタイトル、ラベル、凡例を必要に応じ設定します。

プロパティ - Google折れ線グラフ

| | | |
|----------------|---------------|---------------------|
| axesColor | legendMargins | pos_absolute |
| bgColor | legendOrder | rangeLabels |
| chartType | legendPos | rangeLabelsColor |
| height | legendText | rangeLabelsFontSize |
| hide_if | lineThickness | seriesColor |
| labels | listName | titleColor |
| labelsColor | margins | titleFontSize |
| labelsFontSize | markerColor | titleText |
| legendColor | markerType | transpose |
| legendFontSize | name | width |

name

ウェブレットを識別する名前です。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用しても構いません。

省略値

省略値の名前は自動的に生成された、一意の識別子になります。

有効値

単一引用符で囲まれた文字列。

chartType

折れ線グラフのタイプです。折れ線グラフのタイプは、シンプルな折れ線グラフ、スパークライン(軸なし)、各ポイントのx、y座標付きの折れ線グラフにすることができます。

省略値

Line Chart

有効値

Line Chart (折れ線グラフ)

Line Chart with no Axis (Sparklines) (軸なし折れ線グラフ(スパークライン))

Line Chart with x,y Coordinates for each Point (各ポイントのx、y座標付きの折れ線グラフ)

listName

折れ線グラフで表示するデータを含むJSONリストの名前です。リストの各列がデータ・ポイントになります。リストの各行は系列です。

省略値

省略値はありません。リスト名は必須です。

有効値

有効なデータ値を持つJSONリストの名前。

transpose

trueの場合、リストの列と行が入れ替えられ、各列が系列になります。

省略値

false

有効値

false()またはtrue()

labels

基準軸に使用するラベル。

省略値

グラフは各データ値またはデータ系列グループに連番を付けます。

有効値

文字列もしくは複数言語対応テキスト変数のリスト。

labelsColor

ラベルのテキストの色を指定します。

省略値

Theme:(テーマがある場合)テーマと矛盾しない色を使用します。

有効値

色は16進数でRRGGBBの形式でなければいけません。

labelsFontSize

ピクセルで表された、グラフのラベルのサイズ。

省略値

指定されていない場合は、グーグルにより自動的に設定されます。

有効値

数値で表されたピクセル値。

rangeLabels

範囲軸に使用するラベル。

省略値

範囲軸の目盛りにデータ値の範囲が割り当てられます。

有効値

文字列もしくは複数言語対応テキスト変数のリスト。

rangeLabelsColor

範囲ラベルのテキストの色を指定します。

省略値

Theme:(テーマがある場合)テーマと矛盾しない色を使用します。

有効値

色は16進数でRRGGBBの形式でなければいけません。

rangeLabelsFontSize

ピクセルで表された、グラフの範囲ラベルのサイズ。

省略値

指定されていない場合は、グーグルにより自動的に設定されます。

有効値

数値で表されたピクセル値。

seriesColor

各系列に使用される色。色は16進数でRRGGBBの形式でなければいけません。

省略値

Theme:(テーマが存在する場合)テーマに矛盾しない色(各データ系列ごとに最大10件のデータ系列まで)を使用します。

有効値

色をカンマで区切ったリスト。色は16進数でRRGGBBの形式でなければいけません。

bgColor

グラフの背景色です。色は16進数でRRGGBBの形式でなければいけません。

省略値

Theme:(テーマがある場合)テーマに矛盾しない色を使用します。

有効値

16進数でRRGGBB形式の色。

lineThickness

線の太さをピクセルで指定します。

省略値

thin

有効値

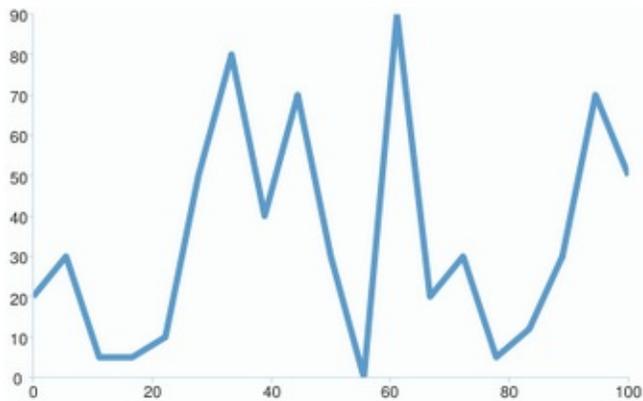
thin (細い)

medium (普通)

thick (太い)

例

普通の太さの折れ線グラフです。



markerType

折れ線グラフのデータ・ポイントに追加するマーカーのマーカー・タイプを選択します。

省略値

None (なし)

有効値

None (なし)

Cross (十字)

Diamond (菱形)

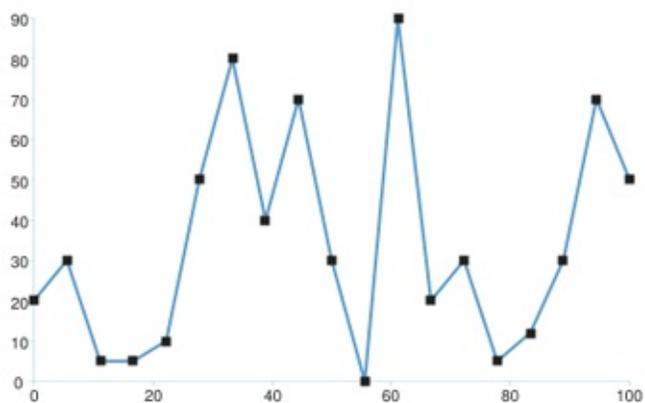
Circle (丸)

Square (四角)

An X (バツ印)

例

四角のマーカー付きの折れ線グラフです。



markerColor

マーカーの色を指定します。

省略値

Theme:(テーマがある場合)テーマに矛盾しない色を使用します。

有効値

16進数でRRGGBB形式の色。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（ウェブレットは相対的に位置付けられます。）

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

ピクセルで記述されたグラフの幅。最大値は1,000です。幅x高さは300,000を超えてはいけません。

省略値

400 ピクセル。

有効値

1,000までの数値。幅x高さは300,000を超えてはいけません。

height

ピクセルで記述されたグラフの高さ。最大値は1,000です。幅x高さは300,000を超えてはいけません。

省略値

250 ピクセル。

有効値

1,000までの数値。幅x高さは300,000を超えてはいけません。

titleText

グラフのタイトルです。パイプ文字()
を使って改行を表します。

省略値

タイトルなし。

有効値

テキスト文字列もしくは複数言語対応テキスト変数

titleColor

グラフ・タイトルの色を指定します。

省略値

Theme:(テーマがある場合)テーマに矛盾しない色を使用します。

有効値

16進数でRRGGBB形式の色。

titleFontSize

ポイントで表された、グラフのフォント・サイズ。

省略値

指定されていない場合は、グーグルにより自動的に設定されます。

有効値

ポイントで表された数値。

axesColor

グラフの軸の色を指定します。

省略値

Theme:(テーマがある場合)テーマに矛盾しない色を使用します。

有効値

16進数でRRGGBB形式の色。

margins

ピクセルで表された、グラフ・エリア周辺の余白の最小サイズ(左、右、上、下)。この値を増やして、軸のラベルがグラフの枠線に当たらないように間隔を開けることができます。

省略値

省略値では、グラフのサイズが計算された後に残ったものが余白になります。

有効値

ピクセル単位のカンマで区切られた数値(左、右、上、下)。

legendText

任意。凡例エントリのテキストを指定します。各ラベルはデータ配列の対応する系列に適用されます。このパラメータが指定されない場合、グラフには凡例が付きません。このラベルに改行は指定できません。通常凡例はテキストを表示するために拡張されるため、凡例のサイズ調整のためにグラフ・エリアが縮小されます。

省略値

凡例は表示されません。

有効値

文字列、もしくは複数言語対応テキスト変数のカンマで区切られたリスト。

legendPos

任意。凡例の位置を指定します。legendTextを空の凡例エントリーにしたい場合は、値に's'を追加することで凡例が無視されます。

省略値

グラフの右側に、凡例エントリーが縦の列に配列されます。

有効値

Right of chart, in vertical column (グラフの右に縦の列で配列)

Bottom of chart, in horizontal row (グラフの下に横の行で配列)

Bottom of chart, in vertical column (グラフの下に縦の列で配列)

Top of chart, in horizontal row (グラフの上に横の行で配列)

Top of chart, in vertical column (グラフの上に縦の列で配列)

Left of chart, in vertical column (グラフの左に縦の列で配列)

legendOrder

任意。凡例の順序を指定します。Display in order (縦方向の凡例の省略値)は入力順にラベルが表示されます。Reverse orderは、棒が表示されるのと同じ順に凡例が表示されるので、積み重ね棒グラフでは便利です。Automatic ordering (横方向の凡例の省略値)は、おおよその長さによってソートされることを意味し、10ピクセル単位で計測されて一番短い凡例が最初に表示されます。(10ピクセルで割った)長さが同じ2つの凡例がある場合、最初にリストされた方が先に表示されます。

省略値

Display in order(順番に表示)

有効値

Display in order(順番に表示)

Reverse order (逆順に表示)

Automatic ordering (自動順)

legendColor

凡例テキストの色を指定します。

省略値

Theme:(テーマがある場合)テーマに矛盾しない色を使用します。

有効値

16進数でRRGGBB形式の色。

legendFontSize

ポイントで表された、凡例のフォント・サイズ。

省略値

グーグルにより自動的に設定されます。

有効値

数値で表されたポイント値。

legendMargins

任意。ピクセルで表された凡例の周りの余白の幅(幅、高さ)。これを使って凡例がグラフ・エリアやイメージの端と重ならないようにします。

省略値

グーグルにより自動的に設定されます。

有効値

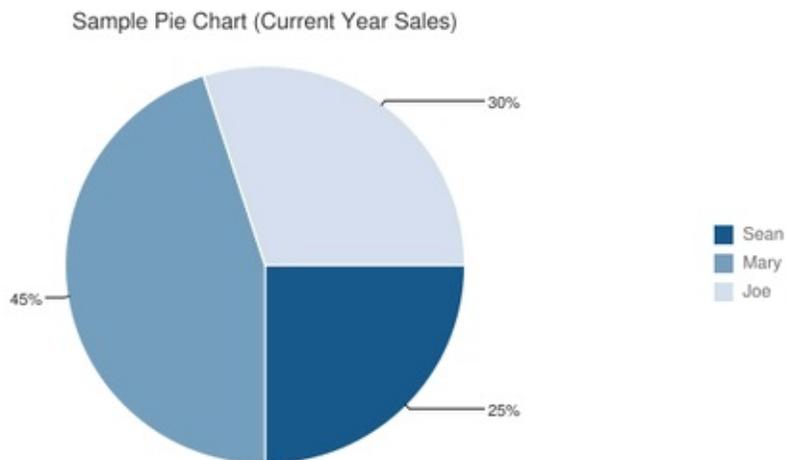
ピクセルで表された、カンマで区切られた数字(幅、高さ)。

8.2.4 Google円グラフ (std_gpie_chart)

クイック・スタート - Google円グラフ プロパティ - Google円グラフ

円グラフは、例えば製品のマーケット・シェアや製品別の売上高など、全体に対する部分の割合を示すのに適しています。

単一データ系列では単純な円グラフまたは3D円グラフが作成できます。同心円グラフを使用すると、各データ系列ごとの複数データ系列を同心円として表示することができます。



クイック・スタート - Google円グラフ

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[チャート ウェブレット]を選択、[Google円グラフ]ウェブレットを探します。
2. Google円グラフ・ウェブレットをデザイナーの上にドラッグして、マウスの左ボタンから手を離します。デザイナー上にプレースホルダーの画像が表示されます。
3. listNameプロパティにWEB_MAP内のリスト名を設定します。
4. transposeプロパティを使用して、リスト行と列を入れ替えることができます。
5. グラフのタイトル、ラベル、凡例を必要に応じ設定します。

プロパティ - Google円グラフ

| | | |
|----------------|----------------|---------------|
| bgColor | legendFontSize | pos_absolute |
| chartType | legendMargins | rotation |
| height | legendOrder | seriesColor |
| hide_if | legendPos | titleColor |
| labels | legendText | titleFontSize |
| labelsColor | listName | titleText |
| labelsFontSize | margins | transpose |
| legendColor | name | width |

name

ウェブレットを識別する名前です。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用しても構いません。

省略値

省略値の名前は自動的に生成された、一意の識別子になります。

有効値

単一引用符で囲まれた文字列。

chartType

円グラフのタイプです。一般的に作成できる円グラフには次の3つのタイプがあります。:フラット、同心、3次元(3D)。

省略値

2次元の円グラフ

有効値

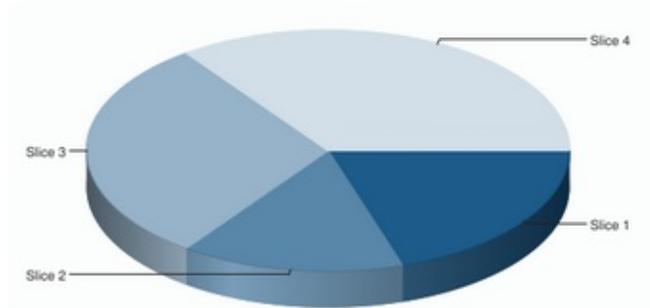
Two dimensional pie chart (2次元の円グラフ)

Three-dimensional pie chart (3次元の円グラフ)

Concentric pie chart (同心円グラフ)

例

3次元の円グラフ:



listName

円グラフで表示するデータを含むJSONリストの名前です。リストの各列が円グラフのスライスになります。同心円グラフの場合、リストの各行が同心円として円グラフ上に表示されます。

省略値

省略値はありません。リスト名は必須です。

有効値

有効なデータ値を持つJSONリストの名前。

transpose

trueの場合、リストの列と行が入れ替えられ、各列が系列になります。

省略値

false

有効値

false()またはtrue()

labels

円グラフのスライスに使用するラベルを指定します。

省略値

円グラフは各スライスに番号を順に振ります。

有効値

文字列もしくは複数言語対応テキスト変数のリスト。

labelsColor

ラベルのテキストの色を指定します。

省略値

Theme:(テーマがある場合)テーマと矛盾しない色を使用します。

有効値

色は16進数でRRGGBBの形式でなければいけません。

labelsFontSize

ピクセルで表された、グラフのラベルのサイズ。

省略値

指定されていない場合は、グーグルにより自動的に設定されます。

有効値

数値で表されたピクセル値。

rotation

省略値では、最初の系列は3:00の位置から描かれ、時計回りにグラフの周囲を埋めていきます。異なる位置から開始する場合は、その値を選択します。

省略値

3:00

有効値

0:00

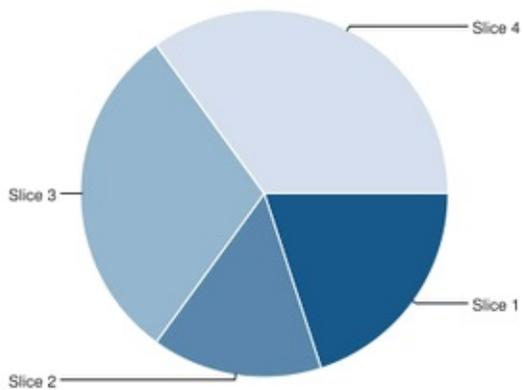
3:00

6:00

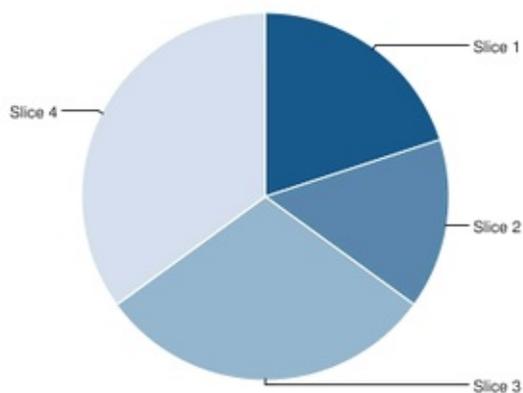
9:00

例

省略値の回転(3:00の位置)



0:00の位置からの回転



seriesColor

各スライスに使用する色。色は16進数でRRGGBBの形式でなければいけません。

省略値

Theme:メインのテーマ色に矛盾しない色で始まり、単色系の一連の色が使用されます。

有効値

色をカンマで区切ったリスト。色は16進数でRRGGBBの形式でなければいけません。

bgColor

グラフの背景色です。色は16進数でRRGGBBの形式でなければいけません。

省略値

Theme:(テーマがある場合)テーマに矛盾しない色を使用します。

有効値

16進数でRRGGBB形式の色。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク (ウェブレットは相対的に位置付けられます。)

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

ピクセルで記述されたグラフの幅。最大値は1,000です。幅x高さは300,000を超えてはいけません。

省略値

400 ピクセル。

有効値

1,000までの数値。幅x高さは300,000を超えてはいけません。

height

ピクセルで記述されたグラフの高さ。最大値は1,000です。幅x高さは300,000を超えてはいけません。

省略値

250 ピクセル。

有効値

1,000までの数値。幅x高さは300,000を超えてはいけません。

titleText

グラフのタイトルです。パイプ文字()
を使って改行を表します。

省略値

タイトルなし。

有効値

テキスト文字列もしくは複数言語対応テキスト変数

titleColor

グラフ・タイトルの色を指定します。

省略値

Theme:(テーマがある場合)テーマに矛盾しない色を使用します。

有効値

16進数でRRGGBB形式の色。

titleFontSize

ポイントで表された、グラフのフォント・サイズ。

省略値

指定されていない場合は、グーグルにより自動的に設定されます。

有効値

ポイントで表された数値。

margins

ピクセルで表された、グラフ・エリア周辺の余白の最小サイズ(左、右、上、下)。この値を増やして、軸のラベルがグラフの枠線に当たらないように間隔を開けることができます。

省略値

省略値では、グラフのサイズが計算された後に残ったものが余白になります。

有効値

ピクセル単位のカンマで区切られた数値(左、右、上、下)。

legendText

任意。凡例エントリのテキストを指定します。各ラベルはデータ配列の対応する系列に適用されます。このパラメータが指定されない場合、グラフには凡例が付きません。このラベルに改行は指定できません。通常凡例はテキストを表示するために拡張されるため、凡例のサイズ調整のためにグラフ・エリアが縮小されます。

省略値

凡例は表示されません。

有効値

文字列、もしくは複数言語対応テキスト変数のカンマで区切られたリスト。

legendPos

任意。凡例の位置を指定します。legendTextを空の凡例エントリーにしたい場合は、値に's'を追加することで凡例が無視されます。

省略値

グラフの右側に、凡例エントリーが縦の列に配列されます。

有効値

Right of chart, in vertical column (グラフの右に縦の列で配列)

Bottom of chart, in horizontal row (グラフの下に横の行で配列)

Bottom of chart, in vertical column (グラフの下に縦の列で配列)

Top of chart, in horizontal row (グラフの上に横の行で配列)

グラフの上に縦の列で配列。

グラフの左に縦の列で配列。

legendOrder

任意。凡例の順序を指定します。Display in order (縦方向の凡例の省略値)は入力順にラベルが表示されます。Reverse orderは、棒が表示されるのと同じ順に凡例が表示されるので、積み重ね棒グラフでは便利です。Automatic ordering (横方向の凡例の省略値)は、おおよその長さによってソートされることを意味し、10ピクセル単位で計測されて一番短い凡例が最初に表示されます。(10ピクセルで割った)長さが同じ2つの凡例がある場合、最初にリストされた方が先に表示されます。

省略値

Display in order(順番に表示)

有効値

Display in order(順番に表示)

Reverse order (逆順に表示)

Automatic ordering (自動順)

legendColor

凡例テキストの色を指定します。

省略値

Theme:(テーマがある場合)テーマに矛盾しない色を使用します。

有効値

16進数でRRGGBB形式の色。

legendFontSize

ポイントで表された、凡例のフォント・サイズ。

省略値

グーグルにより自動的に設定されます。

有効値

数値で表されたポイント値。

legendMargins

任意。ピクセルで表された凡例の周りの余白の幅(幅、高さ)。これを使って凡例がグラフ・エリアやイメージの端と重ならないようにします。

省略値

グーグルにより自動的に設定されます。

有効値

ピクセルで表された、カンマで区切られた数字(幅、高さ)。

8.3 標準フィールド・ビジュアルライゼーション

LANSAには標準フィールド・ビジュアルライゼーション・ウェブレットが用意されています。多くの共通フィールド・タイプ用に標準のビジュアルライゼーションが提供されており、特定のフィールド・ビジュアルライゼーションのフィールドがWEB_MAPに指定されていない場合は、XSLジェネレータ(WAMのコンパイル時に)によって使用されます。例えば、日付タイプのフィールドをWEB_MAPに指定した場合、XSLジェネレータは省略値でstd_dateフィールド・ビジュアルライゼーションを使用し、WebroutineのXSLを生成します。

ここでは、標準フィールド・ビジュアルライゼーション・ウェブレットの詳細、プロパティおよびWebroutine内での使用方法が説明されています。

提供されているウェブレットは、変更しないでください。区画の初期化を実行するたびに、これらのウェブレットは再インポートされ、リポジトリに存在していた元のウェブレットが上書きされます。フィールド・ビジュアルライゼーション・ウェブレットをカスタマイズする場合は、LANSAエディターを使用して、対象のウェブレットを別の名前で保存してから、ウェブレットのコピーの方を変更してください。自分のウェブレットに'std_'で始まる名前を付けないでください。LANSAより提供されているウェブレットと競合してしまう恐れがあります。

| ウェブレット名 | 説明 |
|---|---|
| 英数字
(std_char) | テキスト入力ボックス・コントロール。 |
| ブール値
(std_boolean) | チェックボックス・コントロール |
| Query UI 日付
ピッカー
(std_datepicker) | 日付の表示、入力、プロンプト、及び妥当性検査をサポートする追加機能が付いた、テキスト入力ボックス。 |
| Query UI 日時
ピッカー
(std_datepicker) | 日時の表示、入力、プロンプト、及び妥当性検査をサポートする追加機能が付いた、テキスト入力ボックス。 |

| | |
|---|---|
| 浮動小数
(std_float) | 浮動小数点を表示・受信できるよう特別に構成されたテキスト入力ボックス・コントロール。 |
| 入力ボックス
(std_input) | テキスト入力ボックス・コントロール。 |
| 整数
(std_integer) | 値が許容範囲内にあるかどうかを確認するロジックが入った、整数値を表示・受信できるように構成されたテキスト入力ボックス。 |
| オブジェクト
(std_lob) | LOBコンテンツを返すWebroutineへのハイパーリンク(アンカー)。 |
| Query UI 日時
ピッカー
(std_datepicker) | 時刻の表示、入力、プロンプト、及び妥当性検査をサポートするテキスト入力ボックス。 |
| Varchar
(std_varchar) | テキスト入力ボックス・コントロール。 |

8.3.1 英数字 (std_char)

クイック・スタート - 英数字 プロパティ - 英数字

英数字ウェブレットは、入力テキスト・ボックス・コントロールを提供します。<input type="text"> HTMLエレメントと概ね同じです。ウェブレットは以下のようなものです('メモ'モードの場合)。:



The image shows a web input field with a dropdown menu. The input field contains the letter 'S'. The dropdown menu is open, showing a list of names: Sally, Sarah, Scott, and Sean. The name 'Scott' is highlighted with a blue background.

英数字ウェブレットを使って、文字データを表示したり、入力を受信できます。256文字より長い固定長の文字フィールド用の省略値ウェブレットです。

クイック・スタート - 英数字

Webページに既に存在するフィールドをビジュアル化するためにこのウェブレットを使用するには、LANSAエディターでWebroutineのXSLを開き、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準フィールドビジュアルイゼーション]を選択、[英数字]ウェブレットを見つけてください。
2. このウェブレットを既存のフィールドにドラッグ・アンド・ドロップします。ウェブレットをクリックして、[詳細]タブをクリックします。
3. ウェブレットを既存のフィールドの上にドロップした場合、nameプロパティ、valueプロパティ、及びその他のプロパティが既に適切に設定されています。そうでない場合は、これらのプロパティを必要に応じてウェブレットとWebroutineのweb_mapの必要なフィールドで結びつけるように設定してください。
4. ウェブレットに複数行入力ボックスとしての機能を持たせたい場合は、typeプロパティに'memo'を選択してください。

プロパティ - 英数字

英数字ウェブレットのプロパティは以下のとおりです。

| | | |
|----------------|----------------|-----------|
| class | keyboard_shift | title |
| disabled | maxlength | type |
| display_length | name | value |
| display_mode | pos_absolute | width |
| height | read_only | word_wrap |
| hide_if | tab_index | |

name

ウェブレットを識別する名前です。ウェブレットがフィールドをビジュアライズすると、これがフィールド名になります。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用しても構いません。

省略値

ウェブレットによりビジュアライズされたフィールドの場合、省略値の名前はフィールド名、もしくはフィールド名と行番号を組み合わせたもの(リストのフィールドの場合)になります。それ以外は、省略値の名前は自動的に生成された、一意の識別子になります。

有効値

単一引用符で囲まれた文字列。

value

ウェブレットに設定する値。ウェブレットがフィールドをビジュアルライズする場合、この値を表示するフィールドが識別されます。

省略値

省略値は適用されません - このウェブレットを使用するほとんどの場合、入力ボックスに表示される値をもつフィールド、もしくは入力ボックスの中身を受信するのに使用されるフィールドを指定しなくてはなりません。

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

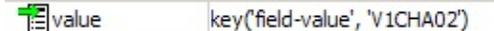
例

これはウェブレットがフィールドを表示するときに値がどのように指定されるかを示しています。



A screenshot of a weblet property field. The field is a horizontal rectangle with a light blue background. On the left side, there is a small icon of a document with a checkmark, followed by the text 'value'. On the right side, there is the text '#V1CHA02'.

プロパティがフォーカスを失った時、式は以下のようになります。



A screenshot of a weblet property field. The field is a horizontal rectangle with a light blue background. On the left side, there is a small icon of a document with a checkmark, followed by the text 'value'. On the right side, there is the text 'key('field-value', 'V1CHA02')'.

display_mode

ウェブレットが入力を受け取るかどうか、出力を表示するかどうかを制御します。

省略値

ブランク ('input'の意味です)。

有効値

リテラル値'input'、'output'または'hidden'。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できる値のリストが表示されます。もしくは、実行時に使用できる値を含むフィールド名、システム変数名、または複数言語変数名を入力することもできます。

maxlength

ユーザーがウェブレットに入力できる文字の最大数を指定します。ウェブレットがフィールドをビジュアルライズする時、そのフィールドに適した数がここに設定されます。

省略値

ブランク (ウェブレットはユーザーが入力できる文字数を制限しません)。

有効値

数値。

display_length

ウェブレット入力ボックスのおおよその幅を文字数で示します。ブラウザは指定された文字数により入力ボックスのサイズ調整をします。

widthプロパティが指定されている場合、そちらが優先され、display_lengthプロパティは無視されます。

省略値

blank (ウェブレットは省略値サイズを選択します)。

有効値

数値。

type

ウェブレットが行う入力コントロールのタイプを指定します。このウェブレットは、`<input type=text>`もしくは`<input type=password>`もしくは`<textarea>`コントロールを行うように設計されています。他のタイプもありますが、このウェブレットではサポートされていません。

省略値

'text'

有効値

プロパティ・シートのこのプロパティの横にあるドロップダウン・ボタンをクリックし、以下の値から1つを選択します。

- 'text' テキスト入力コントロールを作成します。
- 'memo' ワードラップ付きの複数行テキスト入力コントロールを作成します。
- 'password' 出力もしくは入力された文字が表示されずに、代わりにアスタリスクか他のプレースホルダー文字が表示されるテキスト入力コントロールを作成します。

keyboard_shift

入力フィールドのキーボード・シフト。

省略値

このウェブレット・ビジュアルイゼーションのフィールドのキーボード・シフト。それ以外はブランクになります。

有効値

'', 'W', 'J', 'E', 'O' 及び 'U'

| |
|--|
| キーボード・シフトは現在DBCSフィールドの妥当性検査のためだけに使用されています。 |
|--|

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

例

次の例はフィールド#STD_FLAGが 'Y' と等しい場合にウェブレットを非表示にします。この式は以下のような形式で入力してください。

| | |
|---------|-----------------|
| hide_if | #STD_FLAG = 'Y' |
|---------|-----------------|

プロパティがフォーカスを失うと、この式は以下のように表示されます。

| | |
|---------|--------------------------------------|
| hide_if | key('field-value', 'STD_FLAG') = 'Y' |
|---------|--------------------------------------|

class

ウェブレットのカスケード・スタイル・シート(CSS)クラス名です。

省略値

提供されているウェブレットのクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

title

ウェブレットの上をマウスが通った時に、ヒントのテキストとして表示される、ウェブレットのタイトルを指定してください。

省略値

空白 - ヒントは何も表示されません。

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

word_wrap

type プロパティに'memo'が指定されている場合、このプロパティはテキスト入力時にウェブレットが折り返しをどのように行うかを指定します。

省略値

ブランク。

有効値

プロパティ・シートのこのプロパティの横にあるドロップダウン・ボタンをクリックし、以下の値から1つを選択します。

- 'soft' テキストは右端で折り返して表示され、キャリッジ・リターンや改行無しで送られます。
- 'hard' テキストは右端で折り返して表示され、ソフト・リターンと改行付きで送られます。
- 'off' 右端で折り返されません。行はユーザーがタイプした通りに表示されます。

read_only

ウェブレットのコンテンツを読取専用(ユーザーはコンテンツの変更ができません)にするかどうかを決定するブール値。

省略値

ブランク - Falseと同じです。(ユーザーはコンテンツを変更できません。)

有効値

true()、false() もしくは有効な式。

例

次の例はフィールド#STD_FLAGが 'Y' と等しい場合にウェブレットを読取専用を設定します。この式は以下のような形式で入力してください。

```
read_only #STD_FLAG = 'Y'
```

プロパティがフォーカスを失うと、この式は以下のように表示されません。

```
read_only key('field-value', 'STD_FLAG') = 'Y'
```

disabled

ウェブレットを使用可または不可として表示するかを決定するブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false() もしくは有効な式。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（ウェブレットは相対的に位置付けられます。）

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

例

次の例では、[ポジションの固定]がウェブレットで選択されており、LANSAエディターの[デザイン]ビューで要求された位置にウェブレットが配置されています。これは、pos_absoluteプロパティに表示される値になっています。

```
pos_absolute |position: absolute; left: 312.768pt; top: 192.024pt;
```

width

Webページのウェブレットの幅です。

通常、ウェブレット幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これでwidthプロパティやheightプロパティの値が更新されます。ただし、必要に応じてプロパティ値を直接編集することもできます。

省略値

ブランク (ウェブレットは省略値の幅を適用します。)

有効値

単一引用符で囲まれた有効な範囲内の幅。

height

Webページのウェブレットの高さ。

通常、ウェブレットの高さや幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これでwidthプロパティやheightプロパティの値が更新されます。ただし、必要に応じてプロパティの値を直接編集することもできます。

省略値

ブランク（ウェブレットは省略値の高さを適用します。）

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

8.3.2 ブール値 (std_boolean)

クイック・スタート - ブール値 プロパティ - ブール値

ブール値ウェブレットはチェックボックス・コントロールを提供します。これは広い意味でHTMLエレメントの<input type="checkbox">に相当します。チェックボックスのウェブレットを拡張して、特別にブール値タイプのフィールドをサポートします。

チェックボックス・コントロールは、通常2つの状態の1つを表わす際に使用されます。ブール値ウェブレットでは、チェック・ボックスはtrueとfalseの状態を表現します。

ブール値ウェブレットをリストで使用した場合は次のようになります。

| Dept Code | Department Description |
|-------------------------------------|--------------------------|
| <input type="checkbox"/> | ADM ADMINISTRATOR DEPT |
| <input checked="" type="checkbox"/> | AUD INTERNAL AUDITING |
| <input type="checkbox"/> | FLT FLEET ADMINISTRATION |

クイック・スタート - ブール値

ブール値ウェブレットはブール値フィールドの省略値のビジュアルライゼーションであるため、通常、手動でこれをWebページに追加する必要はありません。単にブール値フィールドをweb_mapに含めたり、web_mapに存在するリストに含めることで、ブール値ウェブレットを使用してビジュアルライズされます。

ページにウェブレットを手動で追加する必要がある場合は、[Fields]タブからブール値フィールドをページにドラッグします。あるいは、LANSAエディターでWebroutineのXSLを開き、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準フィールドビジュアルライゼーション]を選択、[ブール値]ウェブレットを見つけてください。
2. [デザイン]ビューでウェブレットをページにドラッグしてください。ウェブレットをクリックして、[詳細]タブをクリックします。
3. nameプロパティ及びvalueプロパティを必要に応じて設定し、WebroutineのWEB_MAPに必要なフィールドとウェブレットを関連付けます。

プロパティ - ブール値

ブール値ウェブレットのプロパティは以下のとおりです。

`class` `mouseover_class` `tab_index`

`display_mode` `name` `value`

`hide_if` `pos_absolute`

name

ウェブレットを識別する名前です。ウェブレットがフィールドをビジュアライズすると、これがフィールド名になります。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用しても構いません。

省略値

ウェブレットによりビジュアライズされたフィールドの場合、省略値の名前はフィールド名、もしくはフィールド名と行番号を組み合わせたもの(リストのフィールドの場合)になります。それ以外は、省略値の名前は自動的に生成された、一意の識別子になります。

有効値

単一引用符で囲まれた文字列。

value

ウェブレットに設定する値。ウェブレットがフィールドをビジュアルライズする場合、この値を表示するフィールドが識別されます。

省略値

省略値は適用されません - このウェブレットを使用する多くの場合、チェックボックスを示す値の入ったフィールドとチェックボックスの状態の受け取りに使用するフィールドの両方もしくはどちらかを指定する必要があります。

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

display_mode

ウェブレットが入力を受け取るかどうか、出力を表示するかどうかを制御します。

省略値

空白 ('input'の意味です)。

有効値

リテラル値'input'、'output'または'hidden'。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できる値のリストが表示されます。もしくは、実行時に使用できる値を含むフィールド名、システム変数名、または複数言語変数名を入力することもできます。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

例

次の例はフィールド#STD_FLAGが 'Y' と等しい場合にウェブレットを非表示にします。この式は以下のような形式で入力してください。

| | |
|---------|-----------------|
| hide_if | #STD_FLAG = 'Y' |
|---------|-----------------|

プロパティがフォーカスを失うと、この式は以下のように表示されます。

| | |
|---------|--------------------------------------|
| hide_if | key('field-value', 'STD_FLAG') = 'Y' |
|---------|--------------------------------------|

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（ウェブレットは相対的に位置付けられます。）

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

class

ウェブレットのカスケード・スタイル・シート(CSS)クラス名です。

省略値

提供されているウェブレットのクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

mouseover_class

マウスがその上に置かれた時の、ウェブレットのカスケード・スタイル・シート(CSS)クラス名。

省略値

このウェブレットに対して適用される省略値はありません。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

8.3.3 jQuery UI 日付ピッカー (std_datepicker)

クイック・スタート - 日付ピッカー プロパティ - 日付ピッカー

日付ピッカー・ウェブレットは日付の表示、入力、プロンプト、妥当性検査のサポート追加機能を持つテキスト入力ボックス・コントロールを提供します。<input type="text"> HTMLエレメントと概ね同じです。

このウェブレットの例を以下に示します。この例では、省略値設定での日付ピッカーを示しています。



日付ピッカー・ウェブレットは、UTC形式で格納されたdateデータ・タイプと共に作動するよう設計されています。

クイック・スタート - 日付ピッカー

日付ピッカー・ウェブレットは日付タイプ(date)のフィールドの省略値のビジュアルライゼーションなので、通常、手動でこれをWebページに追加する必要はありません。日付フィールドをWEB_MAP、またはWEB_MAPに存在するリストに含むだけで、日付ピッカー・ウェブレットを使用してビジュアルライズされます。同様に、時間タイプ(time)のフィールドや日時タイプ(datetime)のフィールドも時間(std_time)ウェブレットや日時(std_datetime)ウェブレットを使用してビジュアルライズされます。

日付フィールドのうち、バージョン12 SP1以前に作成されたものは省略値でstd_dateウェブレットが使用されます。std_datepickerを使用するには、フィールド定義でウェブレットのビジュアルライゼーションを変更します。

日付ピッカー・ウェブレットは、ISO言語コードを使用して、日付の形式、週の最初の曜日やカレンダーのキャプションなどのプロパティをローカライズします。

ページに日付ピッカーウェブレットを手動で追加する必要がある場合は、日付フィールドを[Fields]タブからページにドラッグします。もしくは、LANSAエディターでWebroutineのXSLを開き、以下の手順に従います。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準フィールドビジュアルライゼーション]を選択、[jQuery UI 日付ピッカー]ウェブレットを見つけてください。
2. [デザイン]ビューでウェブレットをページにドラッグしてください。ウェブレットをクリックして、[詳細]タブをクリックします。
3. nameプロパティ及びvalueプロパティを必要に応じて設定し、WebroutineのWEB_MAPに必要なフィールドとウェブレットを関連付けます。

数字フィールドに格納された日付の表示

数字フィールドに格納された日付を日付フィールドにマップする際は、次のような組み込みを使用します。

* Date stored in MMDDYY format in numeric field

```
Webroutine Name(TO_DATE) Desc('Converting from number to date')  
  Web_Map For(*output) Fields(#dat01)  
  #std_date := 061525  
  #dat01 := #std_date.AsDate(MMDDYY)  
Endroutine
```

```
Webroutine Name(FROM_DATE) Desc('Converting from date to number')  
  Web_Map For(*input) Fields(#dat01)  
  #std_date := #dat01.AsNumber(MMDDYY)  
Endroutine
```

プロパティ - 日付ピッカー

日付ピッカー・ウェブレットのプロパティは以下のとおりです。

| | | |
|-------------------------------|-----------------------------------|------------------------------------|
| allow_sqlnull | firstDay | showInline |
| autoSize | hide_if | showMonthAfterYear |
| buttonImage | maxDate | showOn |
| buttonText | minDate | showOtherMonths |
| changeMonth | name | tab_index |
| changeYear | onchange_script | title |
| dateFormat | pos_absolute | value |
| disabled | selectOtherMonths | width |
| display_mode | shortYearCutoff | yearRange |
| duration | showAnim | |

name

ウェブレットを識別する名前です。ウェブレットがフィールドをビジュアライズすると、これがフィールド名になります。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用しても構いません。

省略値

ウェブレットによりビジュアライズされたフィールドの場合、省略値の名前はフィールド名、もしくはフィールド名と行番号を組み合わせたもの(リストのフィールドの場合)になります。それ以外は、省略値の名前は自動的に生成された、一意の識別子になります。

有効値

単一引用符で囲まれた文字列。

value

ウェブレットに設定する値。ウェブレットがフィールドをビジュアルライズする場合、この値を表示するフィールドが識別されます。

省略値

省略値は適用されません - このウェブレットを使用するほとんどの場合、入力ボックスに表示される値をもつフィールド、もしくは入力ボックスの中身を受信するのに使用されるフィールドを指定しなくてはなりません。

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

display_mode

ウェブレットが入力を受け取るかどうか、出力を表示するかどうかを制御します。

省略値

空白 ('input'の意味です)。

有効値

リテラル値'input'、'output'または'hidden'。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できる値のリストが表示されます。もしくは、実行時に使用できる値を含むフィールド名、システム変数名、または複数言語変数名を入力することもできます。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

allow_sqlnull

日付の値をブランクのままにしておけるかどうかを決定するブール値プロパティ。

注：このプロパティはフィールドのリポジトリ定義(ASQN属性)と一致していません。

省略値

false()。ウェブレットがフィールドの上にドロップされた場合、省略値でフィールドのリポジトリ定義のASQN属性になります。

有効値

true()、false() もしくは有効な式。

dateFormat

日付の入力形式。省略値'Auto'は言語(地域設定)の省略値を使用します。有効な書式指定子の一覧は、日時ピッカー・ウェブレットを参照してください。

注: ここではウェブレットが使用するプレゼンテーション形式を指定します。Webroutineから受信・送信する、入力および出力の日付は常にISO形式です。このプロパティを使って異なるプレゼンテーション形式を選択する場合、ウェブレットは必要に応じて内部プレゼンテーションの変換を行います。

省略値

'auto'言語(地域設定)の省略値を使用します。

有効値

プロパティのドロップダウンにリストされている値。

firstDay

1週間の最初の曜日を設定します。

省略値

Auto省略値の地域設定を使用します。

有効値

有効な曜日。プロパティのドロップダウン・リストから選択します。

changeMonth

trueの場合、ドロップダウン・リストから選択して月を変更することができます。

省略値

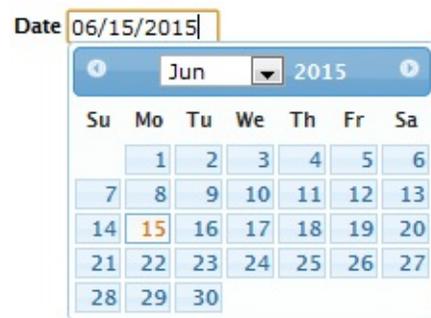
false

有効値

true()、false() もしくは有効な式。

例

このプロパティにtrue()が設定された日付ピッカーです。



changeYear

trueの場合、ドロップダウン・リストから選択して月を変更することができます。

省略値

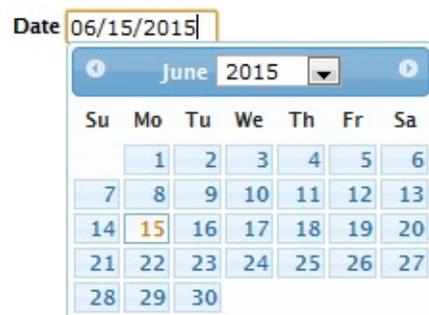
false

有効値

true()、false() もしくは有効な式。

例

このプロパティにtrue()が設定された日付ピッカーです。



yearRange

年のドロップダウンに表示される年の範囲です。現在の年を基準(-nn:+nn)、もしくは現在選択の年を基準(c-nn:c+nn)、絶対値(nnnn:nnnn)、または上記の形式を組み合わせたもの(nnnn:-nn)の中から選択できます。このオプションはドロップダウンに何を表示するかにだけ影響を与え、選択できる日付を制限するには、minDateおよびmaxDateオプションを使います。

省略値

+/1 10 選択年

有効値

c-nn:c+nn:nnは年数です。(選択年を基準)

-nn:+nn:現在の年を基準にした範囲。

nnnn:nnnn:絶対値の年。

例

c-2:c+2:選択の年の前後2年の範囲です。

-1:+3:1年前から3年後までです。

2000:2050:2000年から2050年までの範囲です。

showOtherMonths

trueの場合、現在の月の月初または月末に別の月(選択不可)の日付を表示します。これらの日を選択できるようにするには、selectOtherMonthsを使用します。

省略値

false

有効値

true()、false() もしくは有効な式。

例

このプロパティにtrue()が設定された日付ピッカーです。



selectOtherMonths

trueの場合、表示されている現在月の前後の別の月の日時を選択できます。これはshowOtherMonthsもtrueに設定されている時にのみ適用されます。

省略値

false

有効値

true()、false() もしくは有効な式。

minDate

現在のdateFormatの文字列を使って選択可能な一番始めの日付を設定、もしくは今日からの日数('-7'など)、値や期間の文字列(年は'y'、月は'm'、週は'w'、日は'd'、例えば '-1y -1m')を使って選択可能な最小限の日を設定します。またはnullで無制限になります。

省略値

無制限。

有効値

(上記に記述されている通りの)有効な式、またはプロパティのドロップダウンから事前定義の値を選択します。

例

-1y:1年前。

maxDate

現在のdateFormatの文字列を使って選択可能な一番最後の日付を設定、もしくは今日からの日数('+7'など)、値や期間の文字列(年は'y'、月は'm'、週は'w'、日は'd'、例 '+1m +1w')を使って選択可能な最大限の日を設定します。またはnullで無制限になります。

省略値

無制限。

有効値

(上記に記述されている通りの)有効な式、またはプロパティのドロップダウンから事前定義の値を選択します。

例

+1w:+1週間。

shortYearCutoff

日付の世紀を決定する時の切替の年を設定します。(dateFormatの'y'と共に使用)文字列に数字('0'-'99')のみが提供された場合は、この値が直接使用されます。文字列に '+' がある場合は、現在年に追加されます。切替の年が計算されると、その数値以下の年で入力された日付は現在の世紀と見なされ、これより大きい数の場合は1つ前の世紀と見なされます。

省略値

+10

有効値

切替の年を表わす文字列。

例

+20:現在の年が2015年の場合、00から35までは2000年から2035年だと見なされます。そして、35から99までは1936年から1999年だと見なされます。

showInline

trueの場合、オーバーレイの代わりにページに埋め込まれた日付ピッカーを表示します。

省略値

false

有効値

true()、false() もしくは有効な式。

showOn

日付ピッカーの表示を、フィールドにフォーカスが当たる時に自動的に
行うのか、またはボタンがクリックされた時にだけ行うか、もしくは両
方の場合に行うのかを決定します。

省略値

focus

有効値

focus、 buttonまたはboth

例

ボタンのクリック時に表示します。

Date 

showMonthAfterYear

trueの場合、ヘッダーで月が年の後に来ます。この属性は地域特有の属性の1つです。省略値'Auto'は言語(地域設定)の省略値を使用します。

省略値

Auto

有効値

true()、false() もしくは有効な式。

buttonImage

カレンダー・プロンプト・ボタンに表示するイメージの、イメージ仮想ディレクトリに関連付けられたパスとファイル名。

省略値

'calendar_jqui.gif' (このイメージはLANSAにより提供されます。)

有効値

イメージ・ディレクトリに関連付けられたイメージのパスおよび名前、単一引用符で囲まれたもの。イメージは、プロパティ・シートで該当する省略記号のボタンをクリックし、プロンプターから選択することができます。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

title

ウェブレットの上をマウスが通った時に、ヒントのテキストとして表示される、ウェブレットのテキストを指定してください。

省略値

空白 - ヒントは何も表示されません。

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

buttonText

日付ピッカー・ボタンの上をマウスが通る時にヒントとして表示されるテキストです。

省略値

blank - titleプロパティに指定されたテキストが使用されます。

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

disabled

ウェブレットを使用可または不可として表示するかを決定するブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false() もしくは有効な式。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（ウェブレットは相対的に位置付けられます。）

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

Webページのウェブレットの幅です。ウェブレットは表示されるデータに基づいた最小幅を確保します。

通常、ウェブレット幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これによりwidthプロパティの値が更新されます。ただし、必要に応じてプロパティ値を直接編集することもできます。

省略値

ブランク（ウェブレットは省略値の幅を適用します。）

有効値

単一引用符で囲まれた有効な範囲内の幅。

autoSize

trueに設定すると、入力フィールドに現在の日付形式の日付が入るよう、自動的にサイズ調整されます。

省略値

false

有効値

true()、false() もしくは有効な式。

showAnim

日付ピッカーの表示/非表示に使用するアニメーション名を設定します。

省略値

show

有効値

show、slideDownまたはfadeIn

duration

日付ピッカーが表示されるスピードをコントロールします。事前に定義された3つのスピードから選びます。

省略値

normal

有効値

slow、normalまたはfast。

onchange_script

テキストが変更された後、入力ボックスがフォーカスを失った時に実行されるJavaScriptコード。JavaScriptステートメントは必ずセミコロンで終了していなければなりません。

省略値

空白。JavaScriptは何も実行されません。

有効値

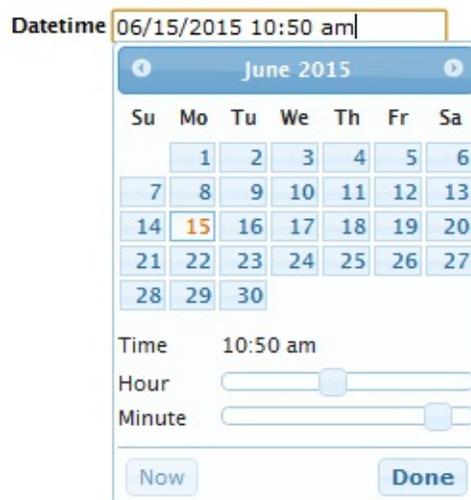
有効なJavaScriptステートメント。

8.3.4 jQuery UI 日時ピッカー (std_datetimepicker)

クイック・スタート - 時間ピッカー プロパティ - 時間ピッカー

日時ピッカー・ウェブレットは日時の表示、入力、プロンプト、妥当性検査のサポート追加機能を持つテキスト入力ボックス・コントロールを提供します。<input type="text"> HTMLエレメントと概ね同じです。

このウェブレットの例を以下に示します。この例では、省略値設定での日時ピッカーを示しています。



日時ピッカー・ウェブレットは、UTC形式で格納されたdatetimeデータ・タイプと共に作動するように設計されています。

クイック・スタート - 日時ピッカー

日時ピッカー・ウェブレットはdatetimeタイプのフィールドの省略値のビジュアルイゼーションなので、通常は手動でこれをWebページに追加する必要はありません。日時フィールドをWEB_MAP、またはWEB_MAPに存在するリストに含むだけで、日時ピッカー・ウェブレットを使用してビジュアルイゼーションされます。同様に、時間タイプ(time)のフィールドや日付タイプ(date)のフィールドも時間(std_time)ウェブレットや日付(std_datepicker)ウェブレットを使用してビジュアルイゼーションされます。

バージョン12 SP1以前に作成された日時フィールドは、省略値でstd_datetimeウェブレットが使用されます。std_datetimestrickerを使用するには、フィールド定義でウェブレットのビジュアルイゼーションを変更します。

日時ピッカー・ウェブレットは、ISO言語コードを使用してプロパティの一部(日付の形式、週の最初の曜日など)やカレンダー・タイム・スライダーのキャプションをローカライズします。

ページに日時ピッカー・ウェブレットを手動で追加する必要がある場合は、datetimeフィールドをページのFieldsタブからドラッグします。もしくは、LANSAエディターでWebroutineのXSLを開き、以下の手順に従います。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準フィールドビジュアルイゼーション]を選択、[jQuery UI 日時ピッカー]ウェブレットを見つけてください。
2. [デザイン]ビューでウェブレットをページにドラッグしてください。ウェブレットをクリックして、[詳細]タブをクリックします。
3. nameプロパティ及びvalueプロパティを必要に応じて設定し、WebroutineのWEB_MAPに必要なフィールドとウェブレットを関連付けます。

プロパティ - 日時ピッカー

日時ピッカー・ウェブレットのプロパティは以下のとおりです。

| | | |
|--------------------------------|-----------------------------------|------------------------------------|
| allow_sqlnull | hide_if | showAnim |
| autoSize | hourMax | showMonthAfterYear |
| buttonImage | minuteMin | showOn |
| buttonText | maxDate | showOtherMonths |
| changeMonth | minDate | stepHour |
| changeYear | minuteMax | stepMinute |
| dateFormat | minuteMin | stepSecond |
| disabled | name | tab_index |
| display_in_utc | onchange_script | timeFormat |
| display_mode | pos_absolute | title |
| duration | selectOtherMonths | value |
| firstDay | shortYearCutoff | width |
| | | yearRange |

name

ウェブレットを識別する名前です。ウェブレットがフィールドをビジュアライズすると、これがフィールド名になります。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用しても構いません。

省略値

ウェブレットによりビジュアライズされたフィールドの場合、省略値の名前はフィールド名、もしくはフィールド名と行番号を組み合わせたもの(リストのフィールドの場合)になります。それ以外は、省略値の名前は自動的に生成された、一意の識別子になります。

有効値

単一引用符で囲まれた文字列。

value

ウェブレットに設定する値。ウェブレットがフィールドをビジュアルライズする場合、この値を表示するフィールドが識別されます。

省略値

省略値は適用されません - このウェブレットを使用するほとんどの場合、入力ボックスに表示される値をもつフィールド、もしくは入力ボックスの中身を受信するのに使用されるフィールドを指定しなくてはなりません。

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

display_mode

ウェブレットが入力を受け取るかどうか、出力を表示するかどうかを制御します。

省略値

ブランク ('input'の意味です)。

有効値

リテラル値'input'、'output'または'hidden'。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できる値のリストが表示されます。もしくは、実行時に使用できる値を含むフィールド名、システム変数名、または複数言語変数名を入力することもできます。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

display_in_utc

日時ピッカーがUTC値の日時を表示するか、ローカル値の日時を表示するかを決定するブール値プロパティ。

省略値

false()。ウェブレットがフィールドの上にドロップされた場合、省略値でフィールドのリポジトリ定義のDUTC属性になります。

有効値

true()、false() もしくは有効な式。

allow_sqlnull

日時の値をブランクのままにしておいて良いかを判断するブール値プロパティ。

注：このプロパティはフィールドのリポジトリ定義(ASQN属性)と一致していません。

省略値

false()。ウェブレットがフィールドの上にドロップされた場合、省略値でフィールドのリポジトリ定義のASQN属性になります。

有効値

true()、false() もしくは有効な式。

dateFormat

日時の日付部分の入力形式です。省略値'Auto'は言語(地域設定)の省略値を使用します。

注: ここではウェブレットが使用するプレゼンテーション形式を指定します。Webroutineから受信・送信する、入力および出力の日時は常にISO形式です。このプロパティを使って異なるプレゼンテーション形式を選択する場合、ウェブレットは必要に応じて内部プレゼンテーションの変換を行います。

省略値

'Auto'言語(地域設定)の省略値を使用します。

有効値

以下の形式がサポートされています。

| 日付形式 | 例 |
|------------|------------|
| dd/mm/yyyy | 09/06/2025 |
| dd/mm/yy | 09/06/25 |
| dd-mm-yyyy | 09-06-2025 |
| dd-mm-yy | 09-06-25 |
| mm/dd/yyyy | 06/09/2025 |
| mm/dd/yy | 06/09/25 |
| mm-dd-yyyy | 06-09-2025 |
| mm-dd-yy | 06-09-25 |
| yyyy-mm-dd | 2025-06-09 |
| yy-mm-dd | 25-06-09 |
| yyyy/mm/dd | 2025/06/09 |
| yy/mm/dd | 25/06/09 |
| d-M-yy | 9-Jun-25 |

| | |
|-------------|---------------|
| d-M-yyyy | 9-Jun-2025 |
| d M, yy | 9 Jun, 25 |
| d M, yyyy | 9 Jun, 2025 |
| d-MM-yy | 9-Jun-25 |
| d-MM-yyyy | 9-Jun-2025 |
| d MM, yy | 9 Jun, 25 |
| d MM, yyyy | 9 Jun, 2025 |
| dd-M-yy | 09-Jun-25 |
| dd-M-yyyy | 09-Jun-2025 |
| dd M, yy | 09 Jun, 25 |
| dd M, yyyy | 09 Jun, 2025 |
| dd-MM-yy | 09-June-25 |
| dd-MM-yyyy | 09-June-2025 |
| dd MM, yy | 09 June, 25 |
| dd MM, yyyy | 09 June, 2025 |
| yy-M-d | 25-Jun-9 |
| yyyy-M-d | 2025-Jun-9 |
| yy M, dd | 25 Jun, 09 |
| yyyy M, dd | 2025 Jun, 09 |
| yy-MM-d | 25-June-9 |
| yyyy-MM-d | 2025-June-9 |
| yy MM, d | 25 June, 9 |
| yyyy MM, d | 2025 June, 9 |
| yy-M-dd | 25-Jun-09 |
| yyyy-M-dd | 2025-Jun-09 |

| | |
|--------------------|----------------------|
| yy M, dd | 25 Jun, 09 |
| yyyy M, dd | 2025 Jun, 09 |
| yy-MM-dd | 25-June-09 |
| yyyy-MM-dd | 2025-June-09 |
| yy MM, dd | 25 June, 09 |
| yyyy MM, dd | 2025 June, 09 |
| DDDDDD, d MM yyyy | Monday, 9 June 2025 |
| DDDDDD, d MM yy | Monday, 9 June 25 |
| DDDDDD, dd MM yyyy | Monday, 09 June 2025 |
| DDDDDD, dd MM yy | Monday, 09 June 25 |

timeFormat

日時の時間部分の入力形式です。省略値'Auto'は言語(地域設定)の省略値を使用します。

省略値

'Auto'言語(地域設定)の省略値を使用します。

有効値

以下の形式がサポートされています。

| 時間形式 | 例 |
|------------|-------------|
| H:mm | 15:30 |
| H:mm:ss | 15:30:00 |
| HH:mm | 15:30 |
| HH:mm:ss | 15:30:00 |
| h:mm T | 3:30 PM |
| h:mm t | 3:30 pm |
| hh:mm T | 03:30 PM |
| hh:mm t | 03:30 pm |
| hh:mm:ss T | 03:30:00 PM |
| hh:mm:ss t | 03:30:00 pm |

firstDay

1週間の最初の曜日を設定します。

省略値

Auto省略値の地域設定を使用します。

有効値

有効な曜日。プロパティのドロップダウン・リストから選択します。

changeMonth

trueの場合、ドロップダウン・リストから選択して月を変更することができます。

省略値

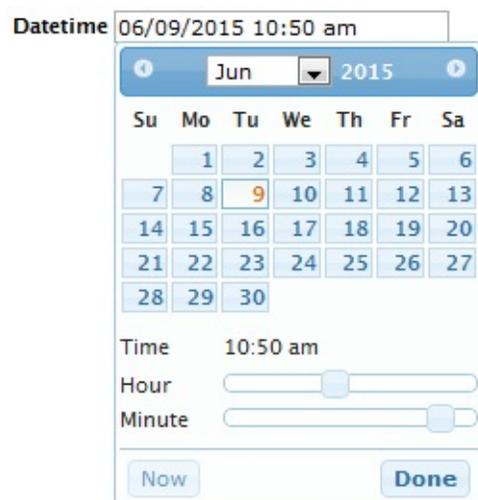
false

有効値

true()、false() もしくは有効な式。

例

このプロパティにtrue()が設定された日時ピッカーです。



changeYear

trueの場合、ドロップダウン・リストから選択して年を変更することができます。

省略値

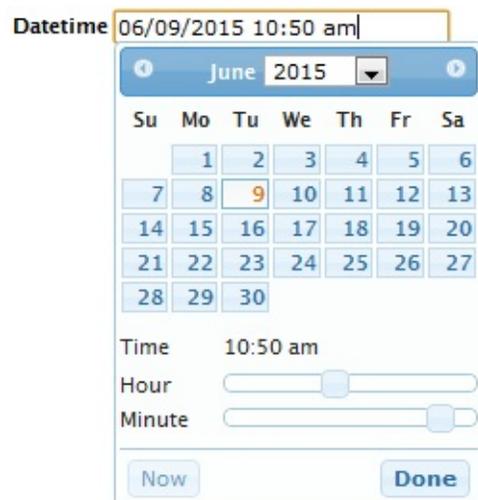
false

有効値

true()、false() もしくは有効な式。

例

このプロパティにtrue()が設定された日時ピッカーです。



yearRange

年のドロップダウンに表示される年の範囲です。現在の年を基準(-nn:+nn)、もしくは現在選択の年を基準(c-nn:c+nn)、絶対値(nnnn:nnnn)、または上記の形式を組み合わせたもの(nnnn:-nn)の中から選択できます。このオプションはドロップダウンに何を表示するかにだけ影響を与え、選択できる日付を制限するには、minDateおよびmaxDateオプションを使います。

省略値

+/1 10 選択年

有効値

c-nn:c+nn:nnは年数です。(選択年を基準)

-nn:+nn:現在の年を基準にした範囲。

nnnn:nnnn:絶対値の年。

例

c-2:c+2:選択の年の前後2年の範囲です。

-1:+3:1年前から3年先までの範囲です。

2000:2050:2000年から2050年までの範囲です。

showOtherMonths

trueの場合、現在の月の月初または月末に別の月(選択不可)の日付を表示します。これらの日を選択できるようにするには、selectOtherMonthsを使用します。

省略値

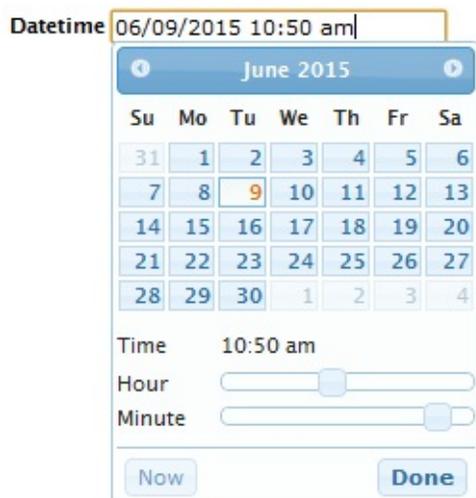
false

有効値

true()、false() もしくは有効な式。

例

このプロパティにtrue()が設定された日時ピッカーです。



selectOtherMonths

trueの場合、表示されている現在月の前後の別の月の日時を選択できます。これはshowOtherMonthsもtrueに設定されている時にのみ適用されます。

省略値

false

有効値

true()、false() もしくは有効な式。

minDate

現在のdateFormatの文字列を使って選択可能な一番始めの日付を設定、もしくは今日からの日数('-7'など)、値や期間の文字列(年は'y'、月は'm'、週は'w'、日は'd'、例えば '-1y -1m')を使って選択可能な最小限の日を設定します。またはnullで無制限になります。

省略値

無制限。

有効値

(上記に記述されている通りの)有効な式、またはプロパティのドロップダウンから事前定義の値を選択します。

例

-1y:1年前。

maxDate

現在のdateFormatの文字列を使って選択可能な一番最後の日付を設定、もしくは今日からの日数('+7'など)、値や期間の文字列(年は'y'、月は'm'、週は'w'、日は'd'、例 '+1m +1w')を使って選択可能な最大限の日を設定します。またはnullで無制限になります。

省略値

無制限。

有効値

(上記に記述されている通りの)有効な式、またはプロパティのドロップダウンから事前定義の値を選択します。

例

+1w:+1週間。

stepHour

日時ピッカーの時間スライダー内の時間ステップの間隔。

省略値

1時間。

有効値

整数

例

1 (1 時間)

stepMinute

日時ピッカーの分スライダー内の分ステップの間隔。

省略値

1分。

有効値

整数

例

15 (15 分)

stepSecond

日時ピッカーの秒スライダー内の秒ステップの間隔。

省略値

1秒

有効値

整数

例

10 (10 秒)

shortYearCutoff

日付の世紀を決定する時の切替の年を設定します。(dateFormatの'y'と共に使用)文字列に数字('0'-'99')のみが提供された場合は、この値が直接使用されます。文字列に '+' がある場合は、現在年に追加されます。切替の年が計算されると、その数値以下の年で入力された日付は現在の世紀と見なされ、これより大きい数の場合は1つ前の世紀と見なされます。

省略値

+10

有効値

切替の年を表わす文字列。

例

+20:現在の年が2015年の場合、00から35までは2000年から2035年だと見なされます。そして、35から99までは1936年から1999年だと見なされます。

hourMin

時間ピッカー・スライダーの最小の時間です。

省略値

0

有効値

0～23

hourMax

時間ピッカー・スライダーの最大の時間です。

省略値

23

有効値

0～23

minuteMin

時間ピッカー・スライダーの最小の分です。

省略値

0

有効値

0～59

minuteMax

時間ピッカー・スライダーの最大の分です。

省略値

59

有効値

0～59

showOn

日時ピッカーの表示を、フィールドにフォーカスが当たる時に自動的に
行うのか、またはボタンがクリックされた時にだけ行うか、もしくは両
方の場合に行うのかを決定します。

省略値

focus

有効値

focus、 buttonまたはboth

例

ボタンのクリック時に表示します。

Datetime 

showMonthAfterYear

trueの場合、ヘッダーで月が年の後に来ます。この属性は地域特有の属性の1つです。省略値'Auto'は言語(地域設定)の省略値を使用します。

省略値

Auto

有効値

true()、false() もしくは有効な式。

buttonImage

カレンダー・プロンプト・ボタンに表示するイメージの、イメージ仮想ディレクトリに関連付けられたパスとファイル名。

省略値

'calendar_jqui.gif' (このイメージはLANSAにより提供されます。)

有効値

イメージ・ディレクトリに関連付けられたイメージのパスおよび名前、単一引用符で囲まれたもの。イメージは、プロパティ・シートで該当する省略記号のボタンをクリックし、プロンプターから選択することができます。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

title

ウェブレットの上をマウスが通った時に、ヒントのテキストとして表示される、ウェブレットのテキストを指定してください。

省略値

空白 - ヒントは何も表示されません。

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

buttonText

日時ピッカー・ボタンの上をマウスが通る時にヒントとして表示されるテキストです。

省略値

blank - titleプロパティに指定されたテキストが使用されます。

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

disabled

ウェブレットを使用可または不可として表示するかを決定するブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false() もしくは有効な式。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（ウェブレットは相対的に位置付けられます。）

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

Webページのウェブレットの幅です。ウェブレットは表示されるデータに基づいた最小幅を確保します。

通常、ウェブレット幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これによりwidthプロパティの値が更新されます。ただし、必要に応じてプロパティ値を直接編集することもできます。

省略値

ブランク（ウェブレットは省略値の幅を適用します。）

有効値

単一引用符で囲まれた有効な範囲内の幅。

autoSize

trueに設定すると、入力フィールドに現在の日時形式の日付と時間が入るよう、自動的にサイズ調整されます。

省略値

false

有効値

true()、false() もしくは有効な式。

showAnim

日時ピッカーの表示/非表示に使用するアニメーション名を設定します。

省略値

show

有効値

show、slideDownまたはfadeIn

duration

日時ピッカーが表示されるスピードをコントロールします。事前に定義された3つのスピードから選びます。

省略値

normal

有効値

slow、normalまたはfast。

onchange_script

テキストが変更された後、入力ボックスがフォーカスを失った時に実行されるJavaScriptコード。JavaScriptステートメントは必ずセミコロンで終了していなければなりません。

省略値

空白。JavaScriptは何も実行されません。

有効値

有効なJavaScriptステートメント。

8.3.5 浮動小数 (std_float)

クイック・スタート - 浮動小数 プロパティ - 浮動小数

浮動小数ウェブレットは、浮動小数点を含む値の表示・受信用に特別に構成されたテキスト入力ボックス・コントロールを提供し、入力値が指定の浮動小数点のサイズに対して許容範囲にあるかを認証するロジックも含まれます。このウェブレットはこのようになります。(わかりやすいように入力ボックスのラベルが表示されていますが、ウェブレットの一部ではありません。)

Float 4

浮動小数ウェブレットは浮動小数点フィールドの省略値のビジュアルライゼーションです。<input type="text"> HTMLエレメントと概ね同じです。

クイック・スタート - 浮動小数

浮動小数ウェブレットは浮動小数点フィールドの省略値のビジュアルライゼーションのため、通常は手動でWebページに追加する必要はありません。 - 必要に応じてウェブレットのプロパティをカスタマイズするだけで使用できます。

ページにウェブレットを手動で追加する必要がある場合は、浮動小数点フィールドをページのFieldsタブからドラッグするだけです。または、LANSAエディターでWebroutineのXSLを開き、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準フィールドビジュアルライゼーション]を選択、[浮動小数]ウェブレットを見つけてください。
2. [デザイン]ビューでウェブレットをページにドラッグしてください。ウェブレットをクリックして、[詳細]タブをクリックします。
3. nameプロパティ及びvalueプロパティを必要に応じて設定し、WebroutineのWEB_MAPに必要なフィールドとウェブレットを関連付けます。
4. maxlength プロパティ及びsize プロパティをフィールド定義に従って設定します。

プロパティ - 浮動小数

浮動小数ウェブレットのプロパティは以下のとおりです。

| | | |
|--------------|--------------|-----------|
| class | maxlength | tab_index |
| disabled | name | title |
| display_mode | pos_absolute | type |
| height | read_only | value |
| hide_if | size | width |

name

ウェブレットを識別する名前です。ウェブレットがフィールドをビジュアライズすると、これがフィールド名になります。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用しても構いません。

省略値

ウェブレットによりビジュアライズされたフィールドの場合、省略値の名前はフィールド名、もしくはフィールド名と行番号を組み合わせたもの(リストのフィールドの場合)になります。それ以外は、省略値の名前は自動的に生成された、一意の識別子になります。

有効値

単一引用符で囲まれた文字列。

value

ウェブレットに設定する値。ウェブレットがフィールドをビジュアルライズする場合、この値を表示するフィールドが識別されます。

省略値

省略値は適用されません - このウェブレットを使用するほとんどの場合、入力ボックスに表示される値をもつフィールド、もしくは入力ボックスの中身を受信するのに使用されるフィールドを指定しなくてはなりません。

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

例

これはウェブレットがフィールドを表示するときに値がどのように指定されるかを示しています。



A screenshot of a weblet property editor. The 'value' property is highlighted in blue, and its value is '#V1FLO01'.

プロパティがフォーカスを失った時、式は以下のようになります。



A screenshot of a weblet property editor. The 'value' property is highlighted in blue, and its value is 'key('field-value', 'V1FLO01')'.

display_mode

ウェブレットが入力を受け取るかどうか、出力を表示するかどうかを制御します。

省略値

空白 ('input'の意味です)。

有効値

リテラル値'input'、'output'または'hidden'。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できる値のリストが表示されます。もしくは、実行時に使用できる値を含むフィールド名、システム変数名、または複数言語変数名を入力することもできます。

maxlength

ユーザーがウェブレットに入力できる文字の最大数を指定します。ウェブレットがフィールドを表示する時、これには符号と数値編集の分のゆとりを含んだ、フィールドに適した数がセットされます。

省略値

ブランク (ウェブレットはユーザーが入力できる文字数を制限しません)。

有効値

数値。

size

文字/バイト数で表された、ウェブレットのデータのサイズ。浮動小数ウェブレットでは、入力値の許容範囲の妥当性検査に使用されます。

省略値

ブランク(ウェブレットが妥当性検査のための最大浮動小数点を想定します。)

有効値

浮動小数点フィールドは4または8バイト長にすることができます。このいずれかの値を妥当性検査に使用する必要があります。

type

ウェブレットが行う入力コントロールのタイプを指定します。このウェブレットは、`<input type=text>`もしくは`<input type=password>`コントロールを行うように設計されています。他のタイプもありますが、このウェブレットではサポートされていません。

省略値

'text'

有効値

プロパティ・シートのこのプロパティの横にあるドロップダウン・ボタンをクリックし、以下の値から1つを選択します。

'text' テキスト入力コントロールを作成します。

'password' 出力もしくは入力された文字が表示されずに、代わりにアスタリスクか他のプレースホルダー文字が表示されるテキスト入力コントロールを作成します。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

例

次の例はフィールド#STD_FLAGが 'Y' と等しい場合にウェブレットを非表示にします。この式は以下のような形式で入力してください。

| | |
|---------|-----------------|
| hide_if | #STD_FLAG = 'Y' |
|---------|-----------------|

プロパティがフォーカスを失うと、この式は以下のように表示されます。

| | |
|---------|--------------------------------------|
| hide_if | key('field-value', 'STD_FLAG') = 'Y' |
|---------|--------------------------------------|

class

ウェブレットのカスケード・スタイル・シート(CSS)クラス名です。

省略値

提供されているウェブレットのクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

title

ウェブレットの上をマウスが通った時に、ヒントのテキストとして表示される、ウェブレットのタイトルを指定してください。

省略値

空白 - ヒントは何も表示されません。

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

read_only

ウェブレットのコンテンツを読取専用(ユーザーはコンテンツの変更ができません)にするかどうかを決定するブール値。

省略値

ブランク - Falseと同じです。(ユーザーはコンテンツを変更できません。)

有効値

true()、false() もしくは有効な式。

例

次の例はフィールド#STD_FLAGが 'Y' と等しい場合にウェブレットを読取専用を設定します。この式は以下のような形式で入力してください。

```
read_only #STD_FLAG = 'Y'
```

プロパティがフォーカスを失うと、この式は以下のように表示されません。

```
read_only key('field-value', 'STD_FLAG') = 'Y'
```

disabled

ウェブレットを使用可または不可として表示するかを決定するブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false() もしくは有効な式。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（ウェブレットは相対的に位置付けられます。）

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

Webページのウェブレットの幅です。

通常、ウェブレット幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これでwidth プロパティやheightプロパティの値が更新されます。ただし、必要に応じてプロパティ値を直接編集することもできます。

省略値

ブランク (ウェブレットは省略値の幅を適用します。)

有効値

単一引用符で囲まれた有効な範囲内の幅。

height

Webページのウェブレットの高さ。

通常、ウェブレットの高さや幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これでwidthプロパティやheightプロパティの値が更新されます。ただし、必要に応じてプロパティの値を直接編集することもできます。

省略値

ブランク（ウェブレットは省略値の高さを適用します。）

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

8.3.6 入力ボックス (std_input)

クイック・スタート - 入力ボックス プロパティ - 入力ボックス

入力ボックス・ウェブレットは、テキスト入力ボックス・コントロールを提供します。<input type="text"> HTMLエレメントと概ね同じです。このウェブレットはこのようになります。(わかりやすいように入力ボックスのラベルが表示されていますが、ウェブレットの一部ではありません。)

Department Code

入力ボックスは数値データと文字データ両方の表示・入力の受信に使用されます。汎用性の高いウェブレットで、多くの特化されたウェブレットがこれを基本にしています。また特化されたウェブレットの方が、使用目的により適している場合が多いです。その例として、std_char、std_varchar、std_float及びstd_integerなどがあります。

クイック・スタート - 入力ボックス

Webページに既に存在するフィールドをビジュアルライズするためにこのウェブレットを使用するには、LANSAエディターでWebroutineのXSLを開き、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上あるドロップダウン・リストから[標準フィールドビジュアライゼーション]を選択し、[入力ボックス]ウェブレットを見つけてください。
2. このウェブレットを既存のフィールドにドラッグ・アンド・ドロップします。ウェブレットをクリックして、[詳細]タブをクリックします。
3. ウェブレットを既存のフィールドの上にドロップした場合、nameプロパティ、valueプロパティ、及びその他のプロパティが既に適切に設定されています。そうでない場合は、これらのプロパティを必要に応じてウェブレットとWebroutineのweb_mapの必要なフィールドで結びつけるように設定してください。

注:Webroutineでのフィールドの定義によっては、生成されたXSLが既にstd_charもしくはstd_varcharのいずれかのフィールド・ビジュアライゼーション・ウェブレットを使用してこのフィールドをビジュアルライズしている可能性があります。これらのウェブレットは入力ボックス・ウェブレット機能の多くに追加機能を提供します。

プロパティ - 入力ボックス

入力ボックス・ウェブレットのプロパティは以下のとおりです。

| | | |
|----------------|-----------------|-----------|
| class | keyboard_shift | size |
| disabled | maxlength | tab_index |
| display_length | name | title |
| display_mode | onchange_script | type |
| height | pos_absolute | value |
| hide_if | read_only | width |

name

ウェブレットを識別する名前です。ウェブレットがフィールドをビジュアライズすると、これがフィールド名になります。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用しても構いません。

省略値

ウェブレットによりビジュアライズされたフィールドの場合、省略値の名前はフィールド名、もしくはフィールド名と行番号を組み合わせたもの(リストのフィールドの場合)になります。それ以外は、省略値の名前は自動的に生成された、一意の識別子になります。

有効値

単一引用符で囲まれた文字列。

value

ウェブレットに設定する値。ウェブレットがフィールドをビジュアルライズする場合、この値を表示するフィールドが識別されます。

省略値

省略値は適用されません - このウェブレットを使用するほとんどの場合、入力ボックスに表示される値をもつフィールド、もしくは入力ボックスの中身を受信するのに使用されるフィールドを指定しなくてはなりません。

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

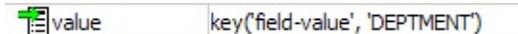
例

これはウェブレットがフィールドを表示するときに値がどのように指定されるかを示しています。



A screenshot of a weblet interface. It shows a dark blue header bar with a small icon on the left and the text 'value' in white. Below the header bar, the text '#DEPARTMENT' is displayed in a light gray font.

プロパティがフォーカスを失った時、式は以下のようになります。



A screenshot of a weblet interface. It shows a light gray header bar with a small icon on the left and the text 'value' in dark gray. Below the header bar, the text 'key('field-value', 'DEPARTMENT')' is displayed in a dark gray font.

display_mode

ウェブレットが入力を受け取るかどうか、出力を表示するかどうかを制御します。

省略値

空白 ('input'の意味です)。

有効値

リテラル値'input'、'output'または'hidden'。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できる値のリストが表示されます。もしくは、実行時に使用できる値を含むフィールド名、システム変数名、または複数言語変数名を入力することもできます。

maxlength

ユーザーがウェブレットに入力できる文字の最大数を指定します。ウェブレットがフィールドをビジュアルライズする時、そのフィールドに適した数がここに設定されます。

省略値

ブランク (ウェブレットはユーザーが入力できる文字数を制限しません)。

有効値

数値。

size

文字数・バイト数で表された、ウェブレットのデータ・サイズ。

このプロパティは現在使われていません。 - 代わりに、`maxlength`及び/もしくは`display_length`プロパティを使用してください。

display_length

文字数で表された、ウェブレット入力ボックスの適切なサイズ。ブラウザはここに指定された文字数により入力ボックスのサイズ調整をします。widthプロパティが指定されている場合、そちらが優先され、display_lengthプロパティは無視されます。

省略値

blank (ウェブレットは省略値サイズを選択します)。

有効値

数値。

type

ウェブレットが行う入力コントロールのタイプを指定します。このウェブレットは、`<input type=text>`もしくは`<input type=password>`コントロールを行うように設計されています。他のタイプもありますが、このウェブレットではサポートされていません。

省略値

'text'

有効値

プロパティ・シートのこのプロパティの横にあるドロップダウン・ボタンをクリックし、以下の値から1つを選択します。

'text' テキスト入力コントロールを作成します。

'password' 出力もしくは入力された文字が表示されずに、代わりにアスタリスクか他のプレースホルダー文字が表示されるテキスト入力コントロールを作成します。

keyboard_shift

入力フィールドのキーボード・シフト。

省略値

このウェブレット・ビジュアルイゼーションのフィールドのキーボード・シフト。それ以外はブランクになります。

有効値

Char、Stringデータ・タイプの場合：'、'W'、'J'、'E'、'O' 及び 'U'

Alphaデータタイプの場合：'、'X'、'A'、'N'、'W'、'T'、'D'、'M'、'J'、'E' 及び 'O'

| |
|--|
| キーボード・シフトは現在DBCSフィールドの妥当性検査のためだけに使用されています。 |
|--|

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

例

次の例はフィールド#STD_FLAGが 'Y' と等しい場合にウェブレットを非表示にします。この式は以下のような形式で入力してください。

| | |
|---------|-----------------|
| hide_if | #STD_FLAG = 'Y' |
|---------|-----------------|

プロパティがフォーカスを失うと、この式は以下のように表示されます。

| | |
|---------|--------------------------------------|
| hide_if | key('field-value', 'STD_FLAG') = 'Y' |
|---------|--------------------------------------|

class

ウェブレットのカスケード・スタイル・シート(CSS)クラス名です。

省略値

提供されているウェブレットのクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

title

ウェブレットの上をマウスが通った時に、ヒントのテキストとして表示される、ウェブレットのタイトルを指定してください。

省略値

空白 - ヒントは何も表示されません。

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

read_only

ウェブレットのコンテンツを読取専用(ユーザーはコンテンツの変更ができません)にするかどうかを決定するブール値。

省略値

空白 - Falseと同じです。(ユーザーはコンテンツを変更できません。)

有効値

true()、false() もしくは有効な式。

例

次の例はフィールド#STD_FLAGが 'Y' と等しい場合にウェブレットを読取専用に設定します。この式は以下のような形式で入力してください。

```
read_only #STD_FLAG = 'Y'
```

プロパティがフォーカスを失うと、この式は以下のように表示されません。

```
read_only key('field-value', 'STD_FLAG') = 'Y'
```

disabled

ウェブレットを使用可または不可として表示するかを決定するブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false() もしくは有効な式。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（ウェブレットは相対的に位置付けられます。）

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

Webページのウェブレットの幅です。

通常、ウェブレット幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これでwidthプロパティやheightプロパティの値が更新されます。ただし、必要に応じてプロパティ値を直接編集することもできます。

省略値

ブランク (ウェブレットは省略値の幅を適用します。)

有効値

単一引用符で囲まれた有効な範囲内の幅。

height

Webページのウェブレットの高さ。

通常、ウェブレットの高さや幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これでwidthプロパティやheightプロパティの値が更新されます。ただし、必要に応じてプロパティの値を直接編集することもできます。

省略値

ブランク（ウェブレットは省略値の高さを適用します。）

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

onchange_script

テキストが変更された後、入力ボックスがフォーカスを失った時に実行されるJavaScriptコード。JavaScriptステートメントは必ずセミコロンで終了していなければなりません。

省略値

空白。JavaScriptは何も実行されません。

有効値

有効なJavaScriptステートメント。

8.3.7 整数 (std_integer)

クイック・スタート - 整数 プロパティ - 整数

整数ウェブレットは、整数値の表示・受信用に特別に構成されたテキスト入力ボックス・コントロールを提供し、入力値が指定の整数サイズに対して許容範囲にあるかを検証するロジックも含まれます。このウェブレットはこのようになります。(わかりやすいように入力ボックスのラベルが表示されていますが、ウェブレットの一部ではありません。)

Integer 4

整数ウェブレットは整数フィールドの省略値のビジュアルライゼーションです。<input type="text"> HTMLエレメントと概ね同じです。

クイック・スタート - 整数

整数ウェブレットは整数フィールドの省略値のビジュアルライゼーションなので、通常は手動でWebページに追加する必要はありません。 - 必要に応じて単にウェブレットのプロパティをカスタマイズするだけです。ページにウェブレットを手動で追加する必要がある場合には、単に整数フィールドをページのFieldsタブからドラッグしてください。もしくは、LANSAエディターでWebroutineのXSLを開き、以下の手順に従います。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準フィールドビジュアルライゼーション]を選択、[整数]ウェブレットを見つけてください。
2. [デザイン]ビューでウェブレットをページにドラッグしてください。ウェブレットをクリックして、[詳細]タブをクリックします。
3. nameプロパティ及びvalueプロパティを必要に応じて設定し、WebroutineのWEB_MAPに必要なフィールドとウェブレットを関連付けます。
4. maxlength プロパティ及びsize プロパティをフィールド定義に従って設定します。

プロパティ - 整数

整数ウェブレットのプロパティは以下のとおりです。

| | | |
|--------------|--------------|-----------|
| class | maxlength | tab_index |
| disabled | name | title |
| display_mode | pos_absolute | type |
| height | read_only | value |
| hide_if | size | width |

name

ウェブレットを識別する名前です。ウェブレットがフィールドをビジュアライズすると、これがフィールド名になります。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用しても構いません。

省略値

ウェブレットによりビジュアライズされたフィールドの場合、省略値の名前はフィールド名、もしくはフィールド名と行番号を組み合わせたもの(リストのフィールドの場合)になります。それ以外は、省略値の名前は自動的に生成された、一意の識別子になります。

有効値

単一引用符で囲まれた文字列。

value

ウェブレットに設定する値。ウェブレットがフィールドをビジュアルライズする場合、この値を表示するフィールドが識別されます。

省略値

省略値は適用されません - このウェブレットを使用するほとんどの場合、入力ボックスに表示される値をもつフィールド、もしくは入力ボックスの中身を受信するのに使用されるフィールドを指定しなくてはなりません。

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

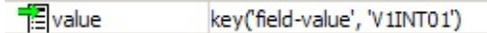
例

これはウェブレットがフィールドを表示するときに値がどのように指定されるかを示しています。



A screenshot of a weblet property editor. It shows a table with two columns. The first column is labeled 'value' and the second column is labeled '#V1INT01'. The 'value' column is highlighted with a blue background.

プロパティがフォーカスを失った時、式は以下のようになります。



A screenshot of a weblet property editor. It shows a table with two columns. The first column is labeled 'value' and the second column is labeled 'key('field-value', 'V1INT01')'. The 'value' column is highlighted with a blue background.

display_mode

ウェブレットが入力を受け取るかどうか、出力を表示するかどうかを制御します。

省略値

空白 ('input'の意味です)。

有効値

リテラル値'input'、'output'または'hidden'。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できる値のリストが表示されます。もしくは、実行時に使用できる値を含むフィールド名、システム変数名、または複数言語変数名を入力することもできます。

maxlength

ユーザーがウェブレットに入力できる文字の最大数を指定します。ウェブレットがフィールドを表示する時、これには符号と数値編集の分のゆとりを含んだ、フィールドに適した数がセットされます。

省略値

ブランク (ウェブレットはユーザーが入力できる文字数を制限しません)。

有効値

数値。

size

文字数・バイト数で表された、ウェブレットのデータ・サイズ。整数ウェブレットでは、入力値の許容範囲の妥当性検査に使用されます。

省略値

空白(ウェブレットは妥当な最大の整数値のサイズを想定します。)

有効値

Integerは1、2、4、または8バイト長にすることができます。このいずれかの値を妥当性検査に使用する必要があります。

type

ウェブレットが行う入力コントロールのタイプを指定します。このウェブレットは、`<input type=text>`もしくは`<input type=password>`コントロールを行うように設計されています。他のタイプもありますが、このウェブレットではサポートされていません。

省略値

'text'

有効値

プロパティ・シートのこのプロパティの横にあるドロップダウン・ボタンをクリックし、以下の値から1つを選択します。

'text' テキスト・エンタリー・コントロールを作成します。

'password' 出力もしくは入力された文字が表示されずに、代わりにアスタリスクか他のプレースホルダー文字が表示されるテキスト入力コントロールを作成します。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

例

次の例はフィールド#STD_FLAGが 'Y' と等しい場合にウェブレットを非表示にします。この式は以下のような形式で入力してください。

| | |
|---------|-----------------|
| hide_if | #STD_FLAG = 'Y' |
|---------|-----------------|

プロパティがフォーカスを失うと、この式は以下のように表示されます。

| | |
|---------|--------------------------------------|
| hide_if | key('field-value', 'STD_FLAG') = 'Y' |
|---------|--------------------------------------|

class

ウェブレットのカスケード・スタイル・シート(CSS)クラス名です。

省略値

提供されているウェブレットのクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

title

ウェブレットの上をマウスが通った時に、ヒントのテキストとして表示される、ウェブレットのタイトルを指定してください。

省略値

空白 - ヒントは何も表示されません。

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

read_only

ウェブレットのコンテンツを読取専用(ユーザーはコンテンツの変更ができません)にするかどうかを決定するブール値。

省略値

ブランク - Falseと同じです。(ユーザーはコンテンツを変更できません。)

有効値

true()、false() もしくは有効な式。

例

次の例はフィールド#STD_FLAGが 'Y' と等しい場合にウェブレットを読取専用に設定します。この式は以下のような形式で入力してください。

```
read_only #STD_FLAG = 'Y'
```

プロパティがフォーカスを失うと、この式は以下のように表示されません。

```
read_only key('field-value', 'STD_FLAG') = 'Y'
```

disabled

ウェブレットを使用可または不可として表示するかを決定するブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false() もしくは有効な式。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（ウェブレットは相対的に位置付けられます。）

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

Webページのウェブレットの幅です。

通常、ウェブレット幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これでwidth プロパティやheightプロパティの値が更新されます。ただし、必要に応じてプロパティ値を直接編集することもできます。

省略値

ブランク (ウェブレットは省略値の幅を適用します。)

有効値

単一引用符で囲まれた有効な範囲内の幅。

height

Webページのウェブレットの高さ。

通常、ウェブレットの高さや幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これでwidthプロパティやheightプロパティの値が更新されます。ただし、必要に応じてプロパティの値を直接編集することもできます。

省略値

ブランク（ウェブレットは省略値の高さを適用します。）

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

8.3.8 オブジェクト (std_lob)

クイック・スタート - オブジェクト プロパティ - オブジェクト

オブジェクト・ウェブレットはハイパーリンク・コントロールを提供します。ハイパーテキストのリンク先を指定する、<a> (anchor)HTMLエレメントとほぼ同じ働きをします。標準のアンカー・ウェブレット (std_anchor)と機能は非常に似ています。

- オブジェクト・ウェブレットは、このリンクを表わすイメージまたはテキスト、もしくはこの両方を表示することができます。これを使って、オブジェクトのコンテンツを扱うWAMやWebroutineを指定します。
- このイメージやテキストは静的(ウェブレットのプロパティでリテラルとして指定)なものでも、指定のフィールド、システム変数、複数言語変数により決定されるものでも構いません。

注：このウェブレットにおける"オブジェクト"とは、LOBまたはストリーム・ファイルのコンテンツを指します。

オブジェクト・ウェブレットは次のようなものです。

[Sample Document](#)

上の例では、ユーザーがハイパーリンクをクリックすると、オブジェクトを扱うWebroutineが起動してこのオブジェクトはユーザー・エージェントに送られます。

クイック・スタート - オブジェクト

オブジェクトを使用する際は、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準フィールドビジュアライゼーション]を選択、[オブジェクト]ウェブレットを見つけてください。
2. 空いているスペースまたはLOBフィールドにオブジェクト・ウェブレットをドラッグし、マウスの左ボタンから手を離します。ハイパーリンクを選択し、[詳細]タブをクリックします。LOBフィールドにウェブレットをドロップした場合、オブジェクト・ウェブレットのnameプロパティとvalueプロパティが、ドロップされたフィールドに基づいて既に設定されていることを確認してください。
3. ウェブレットをリストのカラム内で使用する場合は、(必要に応じて)currentrowhfieldとcurrentrownumvalプロパティをプロパティの説明に従って設定します。
4. on_click_wrnameプロパティに、オブジェクトを扱うWebroutineの名前を設定してください。Webroutineが現在のWebroutineと異なるWAMにある場合は、on_click_wamnameプロパティも設定する必要があります。

プロパティ - オブジェクト

オブジェクト・ウェブレットのプロパティは以下のとおりです。

| | | |
|---------------------|---------------------|--------------------|
| absolute-image-path | on_click_wamname | show_in_new_window |
| class | on_click_wname | tab_index |
| currentrowhfield | pos_absolute_design | target_window_name |
| currentrownumval | presubmit_js | text_class |
| formname | protocol | value |
| hide_if | reentryfield | vf_wamevent |
| mouseover_class | reentryvalue | width_design |
| name | relative-image-path | |

name

ウェブレットを識別する名前です。ウェブレットがフィールドをビジュアライズすると、これがフィールド名になります。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用しても構いません。

省略値

ウェブレットによりビジュアライズされたフィールドの場合、省略値の名前はフィールド名、もしくはフィールド名と行番号を組み合わせたもの(リストのフィールドの場合)になります。それ以外は、省略値の名前は自動的に生成された、一意の識別子になります。

有効値

単一引用符で囲まれた文字列。

value

このプロパティにはハイパーリンクに表示されるテキストを指定します。ウェブレットがフィールドをビジュアルライズする場合、この値を表示するフィールドが識別されます。

省略値

フィールド上にドロップされない場合、省略値は適用されません。フィールド上にドロップされた場合、省略値はこのフィールド値です。

有効値

単一引用符で囲まれたテキスト、フィールド名、システム変数、または複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)ハイパーリンクでは、テキストの代わりに、またはテキストとともにイメージを表示することができます。 - 詳細はrelative-image-pathとabsolute-image-pathプロパティを参照してください。

currentrowhfield

currentrownumval プロパティに指定された値をターゲット Webroutine に渡す時に使用するフィールド名です。フィールド名は単一引用符で囲みます。

詳しくは、currentrownumval プロパティの説明を参照してください。

省略値

'STDROWNUM'

有効値

単一引用符で囲まれた文字列。

例

この例では、ターゲット Webroutine に値を送る際に使用するフィールド名として、ROWNUM を指定しています。ターゲット Webroutine で値を受け取るためには、その WEB_MAP に *BOTH または *INPUT の ROWNUM フィールドがなければいけません。

| | |
|------------------|------------|
| currentrowhfield | 'DEPTLINK' |
|------------------|------------|

currentrownumval

currentrowhfieldプロパティに指定されるフィールドによりターゲット Webroutineに送られる値です。英数字フィールドの場合、値は単一引用符付きで指定する必要があります。数字フィールドの場合は、単一引用符はなくても構いません。

このプロパティは、currentrowhfieldプロパティと共に使用され、値をターゲットのWebroutineにどのように送信するかを記述します。以下の2つの情報が必要になります。

1. currentrowhfield: ターゲットWebroutineが情報を参照するために使用するフィールド名。
2. currentrownumval: リテラル値もしくは必要な情報を含むこの(ソース)Webroutineのフィールド名。

注：プロパティの名前は**currentrownumval** (numeric value: 数値) ではありませんが、currentrownumvalに指定するフィールド名は数値フィールドである必要はありません。

省略値

position()

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

例

この例は対象のWebroutineに値を送信するフィールドとして現在リスト行からROWNUMフィールドを指定しています。

```
r[ ] currentrowhfield    ROWNUM
```

reentryfield

reentryvalue プロパティに指定された値をWAMに送る時に使用するフィールド名です。フィールド名は単一引用符で囲みます。

詳しくは、[reentryvalue](#) プロパティの説明を参照してください。

省略値

'STDREENTRY'

有効値

単一引用符で囲まれた文字列。

reentryvalue

ターゲットWebroutineのreentryfieldプロパティで指定されたフィールドに送信する値です。英数字フィールドの場合、値は単一引用符付きで指定する必要があります。数字フィールドの場合は、単一引用符はなくても構いません。

このプロパティは、reentryfieldプロパティと共に使用され、値をターゲットのWebroutineにどのように送信するか記述します。以下の2つの情報が必要になります。

1. **reentryfield**: ターゲットWebroutineが情報を参照するために使用するフィールド名。
2. **reentryvalue**: リテラル値もしくは必要な情報を含むこの(ソース)Webroutineのフィールド名。

省略値

'D'

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

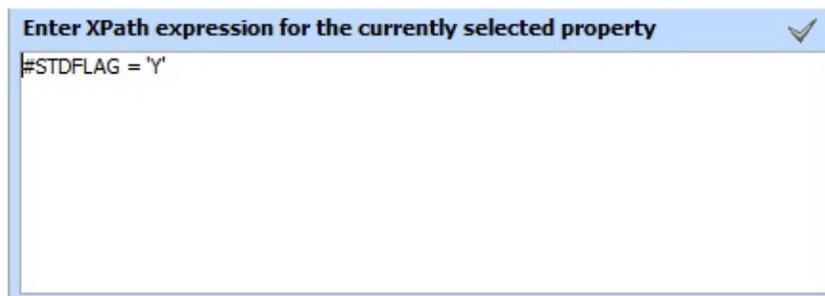
false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

例

次の例はフィールド#STD_FLAGが 'Y' と等しい場合にウェブレットを非表示にします。式はXPath式の領域に入力する必要があります。



Enter XPath expression for the currently selected property ✓

```
#STD_FLAG = 'Y'
```

プロパティがフォーカスを失うと、この式は以下のように表示されます。

```
hide_if key('field-value', 'STD_FLAG') = 'Y'
```

formname

サーバーに送られるHTMLフォーム名です。

省略値

'LANSA'

有効値

単一引用符付きのフォーム名。プロパティ・シート内で該当するドロップダウン・ボタンをクリックすると、使用できるフォーム名リストが表示されます。

on_click_wamname

ハイパーリンクがクリックされた時にオブジェクトを送信する WebroutineのWAM名を指定します。(Webroutine名はon_click_wrnameプロパティに指定します。)

省略値

指定しない場合は、現在のWAMが使用されます。(\$lweb_WAMName)

有効値

単一引用符付きのWAM名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

on_click_wrname

ハイパーリンクがクリックされた時にオブジェクトを送信する Webroutine の名前を指定します。(Webroutine のある WAM 名は on_click_wamname プロパティで指定します。)

省略値

省略値は適用されません。 - on_click_wrname の指定は必須です。

有効値

単一引用符で囲まれた、オブジェクトを扱う Webroutine の名前。このオブジェクトを扱う Webroutine は、on_click_wamname プロパティに指定された WAM に存在していなくてはなりません。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できる Webroutine のリストが表示されます。

protocol

on_click_wnameプロパティに指定されたWebroutineへの移動に使用する
プロトコル(例:http:// または https://)。

通常、このプロパティはセキュア・モードの処理との切り替えが必要な
時に使用します。それ以外の場合は、通常このプロパティを指定する必
要はありません。

省略値

ブランク。現在使用されているプロトコルが使用されます。

有効値

単一引用符付きの、有効なプロトコル。指定する場合は、通常は
'http:'または'https:'です。

show_in_new_window

オブジェクトを新しいブラウザ・ウィンドウに表示するかどうかを決定するブール値のプロパティです。

省略値

false() - 応答オブジェクトは現ブラウザ・ウィンドウに表示されません。

有効値

true()、false() もしくは有効な式。

target_window_name

オブジェクトが表示される、ウィンドウやフレームの名前。

省略値

空白 - オブジェクトは現在のウィンドウに表示されます。

有効値

単一引用符で囲まれたウィンドウ名またはフレーム名。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるウィンドウやフレームが表示されます。

pos_absolute_design

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク（位置は決定されません）。

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width_design

Webページのウェブレットの幅です。

通常、ウェブレット幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これによりwidth-designプロパティの値が更新されます。ただし、必要に応じてプロパティ値を直接編集することもできます。

省略値

ブランク（ウェブレットは省略値の幅を使用します。）

有効値

単一引用符で囲まれた有効な範囲内の幅。

relative-image-path

表示するイメージのイメージ仮想ディレクトリに関連付けられたパスとファイル名。

省略値

空白 - 何も表示されません。

有効値

イメージ・ディレクトリに関連付けられたイメージのパスおよび名前
で、単一引用符で囲まれたもの。イメージは、プロパティ・シートで
該当する省略記号のボタンをクリックし、プロンプターから選択する
ことができます。

absolute-image-path

表示するイメージのパスとファイル名。指定する場合、`relative_image_path`プロパティは空白でなければいけません。

省略値

空白 - 省略値は`relative_image_path`プロパティに指定されたイメージを使用します。

有効値

単一引用符で囲まれたイメージのパスと名前。

class

ウェブレットのカスケード・スタイル・シート(CSS)クラス名です。

省略値

提供されているウェブレットのクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

mouseover_class

マウスがその上に置かれた時の、ウェブレットのカスケード・スタイル・シート(CSS)クラス名。

省略値

このウェブレットに対して適用される省略値はありません。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

text_class

ウェブレットのテキストのカスケード・スタイル・シート(CSS)クラス名です。

省略値

提供されているウェブレットのテキスト・クラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

presubmit_js

ハイパーリンク先に移動する前に実行されるJavaScriptコードです。JavaScriptステートメントは必ずセミコロンで終了していなければなりません。

省略値

空白。JavaScriptは何も実行されません。

有効値

有効なJavaScriptステートメント。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

vf_wamevent

VLF のWAM イベント文字列です。

省略値

空白。

有効値

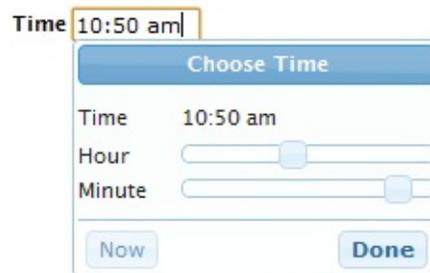
文字列。カンマ (,) は使用できません。

8.3.9 jQuery UI 時間ピッカー (std_timepicker)

クイック・スタート - 時間ピッカー プロパティ - 時間ピッカー

時間ウェブレットは、時刻の表示、入力、プロンプト、妥当性検査のサポート追加機能を持つテキスト入力ボックス・コントロールを提供します。<input type="text"> HTMLエレメントと概ね同じです。

このウェブレットの例を以下に示します。



The image shows a jQuery UI timepicker widget. At the top, there is a text input field with the value "10:50 am" and a label "Time". Below this is a modal dialog box titled "Choose Time". Inside the dialog, there is a label "Time" followed by "10:50 am". Below that are two sliders: "Hour" and "Minute". The "Hour" slider is set to 10 and the "Minute" slider is set to 50. At the bottom of the dialog, there are two buttons: "Now" and "Done".

時間ピッカー・ウェブレットはtimeタイプのフィールドと共に使用するのが一番良い方法です。このタイプを使用すると、データは自動的にウェブレットが想定されている形式で受け渡されます。

クイック・スタート - 時間ピッカー

時間ピッカー・ウェブレットはtimeタイプのフィールドの省略値のビジュアルライゼーションであるため、通常、手動でこれをWebページに追加する必要はありません。時間フィールドをweb_mapに含める、またはweb_mapに存在するリストに含めるだけで、時間ピッカー・ウェブレットを使用してビジュアルライズされます。同様に、日付タイプ(date)のフィールドや日時タイプ(datetime)のフィールドも日付ピッカー(std_datepicker)ウェブレットや日時ピッカー(std_datetimestpicker)ウェブレットを使用してビジュアルライズされます。

時間フィールドのうち、バージョン12 SP1以前に作成されたものは省略値でstd_timeウェブレットが使用されます。std_timepickerを使用するには、フィールド定義でウェブレットのビジュアルライゼーションを変更します。

時間ピッカー・ウェブレットは、ISO言語コードを使用して時間のフォーマットとタイム・スライダーのキャプションをローカライズします。

ページに時間ピッカー・ウェブレットを手動で追加する必要がある場合は、timeフィールドをページの[Fields]タブからドラッグしてください。または、LANSAエディターでWebroutineのXSLを開き、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準フィールドビジュアルライゼーション]を選択、[時間ピッカー]ウェブレットを見つけてください。
2. [デザイン]ビューでウェブレットをページにドラッグしてください。ウェブレットをクリックして、[詳細]タブをクリックします。
3. nameプロパティ及びvalueプロパティを必要に応じて設定し、WebroutineのWEB_MAPに必要なフィールドとウェブレットを関連付けます。

プロパティ - 時間ピッカー

時間ピッカー・ウェブレットのプロパティは以下のとおりです。

| | | |
|-------------------------------|---------------------------------|----------------------------|
| allow_sqlnull | hourMin | showOn |
| autoSize | hourMax | stepHour |
| buttonImage | minuteMin | stepMinute |
| buttonText | minuteMax | stepSecond |
| disabled | name | tab_index |
| display_mode | onchange_script | timeFormat |
| duration | pos_absolute | title |
| hide_if | read_only | showAnim |
| | value | width |

name

ウェブレットを識別する名前です。ウェブレットがフィールドをビジュアライズすると、これがフィールド名になります。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用しても構いません。

省略値

ウェブレットによりビジュアライズされたフィールドの場合、省略値の名前はフィールド名、もしくはフィールド名と行番号を組み合わせたもの(リストのフィールドの場合)になります。それ以外は、省略値の名前は自動的に生成された、一意の識別子になります。

有効値

単一引用符で囲まれた文字列。

value

ウェブレットに設定する値。ウェブレットがフィールドをビジュアルライズする場合、この値を表示するフィールドが識別されます。

省略値

省略値は適用されません - このウェブレットを使用するほとんどの場合、入力ボックスに表示される値をもつフィールド、もしくは入力ボックスの中身を受信するのに使用されるフィールドを指定しなくてはなりません。

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

display_mode

ウェブレットが入力を受け取るかどうか、出力を表示するかどうかを制御します。

省略値

空白 ('input'の意味です)。

有効値

リテラル値'input'、'output'または'hidden'。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できる値のリストが表示されます。もしくは、実行時に使用できる値を含むフィールド名、システム変数名、または複数言語変数名を入力することもできます。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

allow_sqlnull

時間ピッカーの値を空白のままにしてもよいかどうかを判断する
ブール値プロパティ。

注：このプロパティはフィールドのリポジトリ定義(ASQN属性)と一致していません。

省略値

false()。ウェブレットがフィールドの上にドロップされた場合、省略
値でフィールドのリポジトリ定義のASQN属性になります。

有効値

true()、false() もしくは有効な式。

timeFormat

時間ピッカーの入力形式。省略値'Auto'は言語(地域設定)の省略値を使用します。

省略値

'Auto'言語(地域設定)の省略値を使用します。

有効値

有効な書式指定子の一覧は、日時ピッカー・ウェブレットを参照してください。

hourMin

時間ピッカー・スライダーの最小の時間です。

省略値

0

有効値

0～23

hourMax

時間ピッカー・スライダーの最大の時間です。

省略値

23

有効値

0～23

minuteMin

時間ピッカー・スライダーの最小の分です。

省略値

0

有効値

0～59

minuteMax

時間ピッカー・スライダーの最大の分です。

省略値

59

有効値

0～59

stepHour

時間ピッカーの時間スライダー内の時間ステップの間隔。

省略値

1時間。

有効値

整数

例

1 (1 時間)

stepMinute

時間ピッカーの分スライダー内の分ステップの間隔。

省略値

1分。

有効値

整数

例

15 (15 分)

stepSecond

時間ピッカーの秒スライダー内の秒ステップの間隔。

省略値

1秒

有効値

整数

例

10 (10 秒)

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

title

ウェブレットの上をマウスが通った時に、ヒントのテキストとして表示される、ウェブレットのテキストを指定してください。

省略値

空白 - ヒントは何も表示されません。

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

showOn

時間ピッカーの表示を、フィールドにフォーカスが当たる時に自動的に
行うのか、またはボタンがクリックされた時にだけ行うか、もしくは両
方の場合に行うのかを決定します。

省略値

focus

有効値

focus、 buttonまたはboth

例

ボタンのクリック時に表示します。

Time 

buttonImage

時間ピッカー・プロンプト・ボタンに表示するイメージの、イメージ仮想ディレクトリに関連付けられたパスとファイル名。

省略値

'calendar_jqui.gif' (このイメージはLANSAにより提供されます。)

有効値

イメージ・ディレクトリに関連付けられたイメージのパスおよび名前、単一引用符で囲まれたもの。イメージは、プロパティ・シートで該当する省略記号のボタンをクリックし、プロンプターから選択することができます。

buttonText

時間ピッカー・ボタンの上をマウスが通る時にヒントとして表示されるテキストです。

省略値

blank - titleプロパティに指定されたテキストが使用されます。

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

read_only

ウェブレットのコンテンツを読取専用(ユーザーはコンテンツを修正できません)にするかどうかを決定するブール値。

省略値

空白 - Falseと同じです。(ユーザーはコンテンツを変更できません。)

有効値

true()、false() もしくは有効な式。

disabled

ウェブレットを使用可または不可として表示するかを決定するブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false() もしくは有効な式。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク (ウェブレットは相対的に位置付けられます。)

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

Webページのウェブレットの幅です。ウェブレットは表示されるデータに基づいた最小幅を確保します。

通常、ウェブレット幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これによりwidthプロパティの値が更新されます。ただし、必要に応じてプロパティ値を直接編集することもできます。

省略値

ブランク（ウェブレットは省略値の幅を適用します。）

有効値

単一引用符で囲まれた有効な範囲内の幅。

autoSize

trueに設定すると、入力フィールドに現在の日付形式の日付が入るよう、自動的にサイズ調整されます。

省略値

false

有効値

true()、false() もしくは有効な式。

showAnim

時間ピッカーの表示/非表示に使用するアニメーション名を設定します。

省略値

show

有効値

show、slideDownまたはfadeIn

duration

時間ピッカーが表示されるスピードをコントロールします。事前に定義された3つのスピードから選びます。

省略値

normal

有効値

slow、normalまたはfast。

onchange_script

テキストが変更された後、入力ボックスがフォーカスを失った時に実行されるJavaScriptコード。JavaScriptステートメントは必ずセミコロンで終了していなければなりません。

省略値

空白。JavaScriptは何も実行されません。

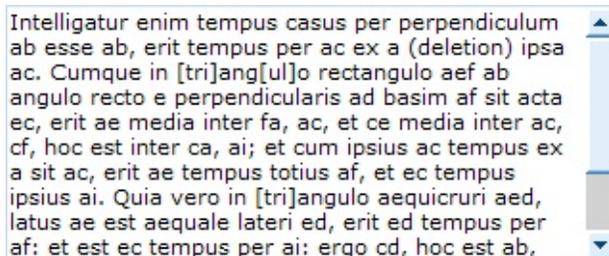
有効値

有効なJavaScriptステートメント。

8.3.10 Varchar (std_varchar)

クイック・スタート - Varchar プロパティ - Varchar

Varcharウェブレットは、入力のためのテキスト・ボックス・コントロールを提供します。<input type="text"> HTMLエレメントと概ね同じです。ウェブレットは以下のようなものです('メモ'モードの場合)。：



Varcharウェブレットを使って、可変長文字データの表示や、入力を受信します。これは、可変長の文字フィールドの省略値ウェブレットです。

クイック・スタート - Varchar

Webページに既に存在するフィールドをビジュアルライズするためにこのウェブレットを使用するには、LANSAエディターでWebroutineのXSLを開き、以下の手順に従ってください。

1. [ウェブレット テンプレート]タブをクリックし、上にあるドロップダウン・リストから[標準フィールドビジュアライゼーション]を選択、[Varchar]ウェブレットを見つけてください。
2. このウェブレットを既存のフィールドにドラッグ・アンド・ドロップします。ウェブレットをクリックして、[詳細]タブをクリックします。
3. ウェブレットを既存のフィールドの上にドロップした場合、nameプロパティ、valueプロパティ、及びその他のプロパティが既に適切に設定されています。そうでない場合は、これらのプロパティを必要に応じてウェブレットとWebroutineのweb_mapの必要なフィールドで結びつけるように設定してください。
4. ウェブレットに複数行入力ボックスとしての機能を持たせたい場合は、typeプロパティに'memo'を選択してください。

プロパティ - Varchar

Varcharウェブレットのプロパティは以下のとおりです。

| | | |
|----------------|----------------|-----------|
| class | keyboard_shift | title |
| disabled | maxlength | type |
| display_length | name | value |
| display_mode | pos_absolute | width |
| height | read_only | word_wrap |
| hide_if | tab_index | |

name

ウェブレットを識別する名前です。ウェブレットがフィールドをビジュアライズすると、これがフィールド名になります。通常、ここは省略値のままで残し、LANSAがLANSA独自の内部命名規則を使用できるようにします。ただし、JavaScript やウェブレットを参照するXSLを使用する場合は、独自の名前を使用しても構いません。

省略値

ウェブレットによりビジュアライズされたフィールドの場合、省略値の名前はフィールド名、もしくはフィールド名と行番号を組み合わせたもの(リストのフィールドの場合)になります。それ以外は、省略値の名前は自動的に生成された、一意の識別子になります。

有効値

単一引用符で囲まれた文字列。

value

ウェブレットに設定する値。ウェブレットがフィールドをビジュアルライズする場合、この値を表示するフィールドが識別されます。

省略値

省略値は適用されません - このウェブレットを使用するほとんどの場合、入力ボックスに表示される値をもつフィールド、もしくは入力ボックスの中身を受信するのに使用されるフィールドを指定しなくてはなりません。

有効値

単一引用付きのテキスト、もしくはフィールド名、システム変数名、複数言語対応テキスト変数名。

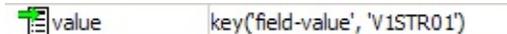
例

これはウェブレットがフィールドを表示するときに値がどのように指定されるかを示しています。



A screenshot of a weblet property editor. The 'value' property is highlighted in blue, and its value is '#V1STR01'.

プロパティがフォーカスを失った時、式は以下のようになります。



A screenshot of a weblet property editor. The 'value' property is highlighted in blue, and its value is 'key('field-value', 'V1STR01')'.

display_mode

ウェブレットが入力を受け取るかどうか、出力を表示するかどうかを制御します。

省略値

空白 ('input'の意味です)。

有効値

リテラル値'input'、'output'または'hidden'。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できる値のリストが表示されます。もしくは、実行時に使用できる値を含むフィールド名、システム変数名、または複数言語変数名を入力することもできます。

maxlength

ユーザーがウェブレットに入力できる文字の最大数を指定します。ウェブレットがフィールドをビジュアルライズする時、そのフィールドに適した数がここに設定されます。

省略値

ブランク (ウェブレットはユーザーが入力できる文字数を制限しません)。

有効値

数値。

display_length

ウェブレット入力ボックスのおおよその幅を文字数で示します。ブラウザは指定された文字数により入力ボックスのサイズ調整をします。

widthプロパティが指定されている場合、そちらが優先され、display_lengthプロパティは無視されます。

省略値

blank (ウェブレットは省略値サイズを選択します)。

有効値

数値。

type

ウェブレットが行う入力コントロールのタイプを指定します。このウェブレットは、`<input type=text>`もしくは`<input type=password>`もしくは`<textarea>`コントロールを行うように設計されています。他のタイプもありますが、このウェブレットではサポートされていません。

省略値

'text'

有効値

プロパティ・シートのこのプロパティの横にあるドロップダウン・ボタンをクリックし、以下の値から1つを選択します。

- 'text' テキスト入力コントロールを作成します。
- 'memo' ワードラップ付きの複数行テキスト入力コントロールを作成します。
- 'password' 出力もしくは入力された文字が表示されずに、代わりにアスタリスクか他のプレースホルダー文字が表示されるテキスト入力コントロールを作成します。

keyboard_shift

入力フィールドのキーボード・シフト。

省略値

このウェブレット・ビジュアルイゼーションのフィールドのキーボード・シフト。それ以外はブランクになります。

有効値

'', 'W', 'J', 'E', 'O' 及び 'U'

キーボード・シフトは現在DBCSフィールドの妥当性検査のためだけに使用されています。

hide_if

評価がTrueの時にウェブレットが非表示になる式です。

省略値

false() (ウェブレットは常に表示されます)

有効値

ブール値を返す有効なXPath式。

例

次の例はフィールド#STD_FLAGが 'Y' と等しい場合にウェブレットを非表示にします。この式は以下のような形式で入力してください。

| | |
|---------|-----------------|
| hide_if | #STD_FLAG = 'Y' |
|---------|-----------------|

プロパティがフォーカスを失うと、この式は以下のように表示されます。

| | |
|---------|--------------------------------------|
| hide_if | key('field-value', 'STD_FLAG') = 'Y' |
|---------|--------------------------------------|

class

ウェブレットのカスケード・スタイル・シート(CSS)クラス名です。

省略値

提供されているウェブレットのクラス名。

有効値

単一引用符で囲まれた、カスケード・スタイル・シートの有効クラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

tab_index

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

1. 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
2. ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
3. 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれません。これはHTML仕様に定義された動作ではなく、Internet Explorer 及び Firefoxでのみサポートされることに注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

ブランクもしくは有効な数値。

title

ウェブレットの上をマウスが通った時に、ヒントのテキストとして表示される、ウェブレットのタイトルを指定してください。

省略値

空白 - ヒントは何も表示されません。

有効値

単一引用符付きのテキスト、もしくは複数言語対応テキスト変数(プロパティ・シートで該当する省略記号のボタンをクリックしてリストから1つ選ぶことができます。)

word_wrap

type プロパティに'memo'が指定されている場合、このプロパティはテキスト入力時にウェブレットが折り返しをどのように行うかを指定します。

省略値

ブランク。

有効値

プロパティ・シートのこのプロパティの横にあるドロップダウン・ボタンをクリックし、以下の値から1つを選択します。

- 'soft' テキストは右端で折り返して表示され、キャリッジ・リターンや改行無しで送られます。
- 'hard' テキストは右端で折り返して表示され、ソフト・リターンと改行付きで送られます。
- 'off' 右端で折り返されません。行はユーザーがタイプした通りに表示されます。

read_only

ウェブレットのコンテンツを読取専用(ユーザーはコンテンツの変更ができません)にするかどうかを決定するブール値。

省略値

ブランク - Falseと同じです。(ユーザーはコンテンツを変更できません。)

有効値

true()、false() もしくは有効な式。

例

次の例はフィールド#STD_FLAGが 'Y' と等しい場合にウェブレットを読取専用に設定します。この式は以下のような形式で入力してください。

```
read_only #STD_FLAG = 'Y'
```

プロパティがフォーカスを失うと、この式は以下のように表示されません。

```
read_only key('field-value', 'STD_FLAG') = 'Y'
```

disabled

ウェブレットを使用可または不可として表示するかを決定するブール値プロパティ。

省略値

空白 - Falseと同じです。（ウェブレットは常に使用可能な状態）

有効値

true()、false() もしくは有効な式。

pos_absolute

Webページ上のウェブレットの絶対的な位置。このプロパティを使用する場合は、ウェブレットの右クリック・メニューで[ポジションの固定]が選択されていないことに注意してください。このプロパティは通常ウェブレットをドラッグ・アンド・ドロップしたピクセル値により設定されます。

省略値

ブランク (ウェブレットは相対的に位置付けられます。)

有効値

'left'や'top'などの有効な座標を付け、単一引用符で囲まれた有効な寸法。

width

Webページのウェブレットの幅です。

通常、ウェブレット幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これでwidthプロパティやheightプロパティの値が更新されます。ただし、必要に応じてプロパティ値を直接編集することもできます。

省略値

ブランク (ウェブレットは省略値の幅を適用します。)

有効値

単一引用符で囲まれた有効な範囲内の幅。

height

Webページのウェブレットの高さ。

通常、ウェブレットの高さや幅は、LANSAエディターの[デザイン]ビューで、ウェブレットの周りのグラブ・ハンドルをドラッグして設定します。これでwidthプロパティやheightプロパティの値が更新されます。ただし、必要に応じてプロパティの値を直接編集することもできます。

省略値

ブランク（ウェブレットは省略値の高さを適用します。）

有効値

有効な範囲内にある単一引用符で囲まれた高さ。

8.4 レイアウト・ウェブレット

8.4.1 クイック・スタート - 標準レイアウト

レイアウト・ウェブレットは、全てのHTMLページに必要となる基本的なHTMLページ構造(ヘッダー、タイトル、本文など)を提供します。また、全てのWebroutineで必要な標準リソース(スクリプト、スタイル、非表示フィールドなど)や、Webページで共通に使用される標準レイアウト要素(ヘッダー、フッター、コンテンツ・エリア、サイドバーなど)も提供します。

レイアウト・ウェブレットはWAMレイアウトのベースとなります。レイアウト・ウェブレットはWAMもしくはウェブ・アプリケーションの共通インターフェースを提供します。通常レイアウトにメニュー、配色、メッセージ操作、イメージ、ロゴ、トレードマーク情報などを導入することで、一貫性のとれたアプリケーションの外観を保つことができます。

省略値では、WAMは全ての生成Webroutineで1つのWAMレイアウトを使用します。共通のレイアウト機能を共有するには、同じサイト・レイアウトをアプリケーション全体で使用するか(詳細は「[独自のサイト・レイアウトの作成方法](#)」を参照)、またはWAMレイアウトを共通レイアウト・ウェブレットのベースにします。生成されたWAMレイアウトは、独自のサイト・レイアウトまたは共通レイアウト・ウェブレットからの共通の機能を共有し、特定の要求に合わせてカスタマイズすることができます。詳しい情報については、「[WAMレイアウトとレイアウト・ウェブレット](#)」を参照してください。

レイアウト・ウェブレットのグループには、LANSAにより提供される標準レイアウト・ウェブレットが含まれています。WAMレイアウト、つまり<WAM名>_layoutという命名規則に従って命名されたものは、省略値ではWAMレイアウトという別のグループに含まれます。

提供されているレイアウトは絶対に修正しないでください。カスタマイズしたい場合は、コピーをし、そのコピーを修正するようにしてください。

| ウェブレット名 | 説明 |
|---------|----|
|---------|----|

| | |
|---|---|
| 標準テーマ・レイアウト
(std_themelet1_[1- | Webroutineページの基本的なjQueryテーマのレイアウトです。各レイアウトには1つ、2つ、または3つのコンテンツ・エリアがあります。 |
|---|---|

3]col) std_themelet1_1colが省略値のレイアウトです。

標準空白・レイアウト (std_blank_layout) レイアウトやCSSのない、単純な空白のレイアウトです。

標準基本レイアウト (std_layout_V2 and std_layout[1-5]_v2) ヘッダーとフッターがあり、メニューはない基本レイアウトです。また様々な基本レイアウトを提供します。この各基本レイアウトがWebroutineページに異なる外観を提供します。

8.4.1 クイック・スタート - 標準レイアウト

通常は、自身のサイトの標準やアプリケーションに合うようにレイアウト・ウェブレットを開発・カスタマイズします。

サイト・レイアウト・マネージャを使ってサイト・レイアウトのウェブレットを作成することも可能です。(詳細は、『[独自のサイト・レイアウトの作成方法](#)』を参照)

アプリケーション内の全てのWAMレイアウトのベースとして自身のサイト・レイアウト・ウェブレットを使用するには、各WAMの作成時に自身のサイト・レイアウトをレイアウト・ウェブレットとして指定します。これが推奨される方法です。

これとは別の方法として、次の手順でWAMレイアウトに自身のレイアウトを適用することも可能です。

1. レイアウト・ウェブレットを適用するWAMレイアウトを開きます。
2. [ウェブレット テンプレート]タブを選択し、ウェブレット・グループのドロップダウン・リストから[レイアウト・ウェブレット]を選択します。
3. 適用したいレイアウト・ウェブレットを探します。
4. 自身のWAMレイアウトの[デザイン]ビュー上にレイアウト・ウェブレットをドロップします。
5. レイアウト・ウェブレットが既存のレイアウトに追加されます。
6. 古いレイアウトにフォーカスを当てて、[削除]キーを押してください。これで、新しいレイアウトのみが残ります。
7. WAMレイアウトを保存します。
8. 自身のサイトのカスタマイズされたレイアウトの場合は、これで作業は終了です。

レイアウト・ウェブレットはWebroutineのデザインに直接ドロップしないでください。この場合、Webroutineのデザインに行ったレイアウト変更はこのWebroutineにのみに適用され、WAM全体には反映されません。

レイアウトに使用されるロゴのイメージは、std_headerlとstd_headerrクラスのカスケード・スタイル・シートstd_style.min.cssで定義されることに注意してください。



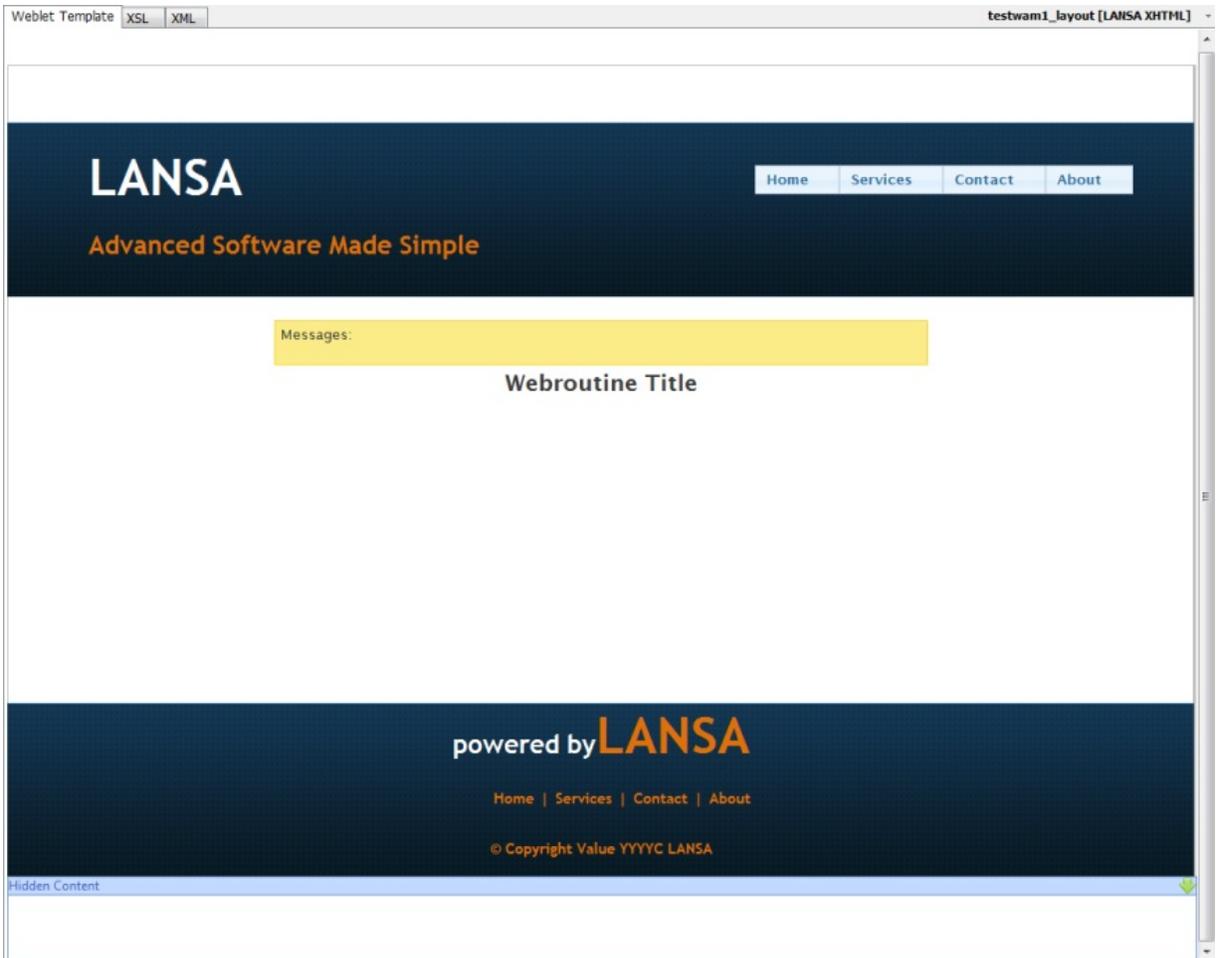
8.4.2 標準テーマ・レイアウト (std_themelet1_[1-3]col)

std_themelet1_[1-3]colはヘッダー、フッター、コンテンツ、および非表示領域が定義されたjQueryテーマのレイアウトです。

この3つのレイアウトは基本的には同じですが、提供されるコンテンツ・エリアが1つ、2つもしくは3つのものがあり、固定または可変の幅が設定できます。現在のところ、2種類のスタイルと8種類のjQueryテーマが提供されており、これらのレイアウトに適用することができます。非表示のコンテンツ・エリアの一番右側にある矢印を使って、非表示コンテンツ・エリアを折りたたんだり、後で展開したりすることができます。

std_themelet1_1colは、LANSAがWAM XSLを生成する時に自動生成WAMのレイアウトとして省略値で使用されます。この省略値は、WAMが作成された時に特定のサイト・レイアウト・ウェブレットが指定されていなかった場合に適用されます。std_themelet1_1colレイアウト・ウェブレットは絶対に修正しないでください。

std_themelet1_1colレイアウト・ウェブレットを使って作成されたWAMレイアウトはエディターで開くと、次のようになります。(テーマはRedmond、スタイル #1のテーマレットを使用)

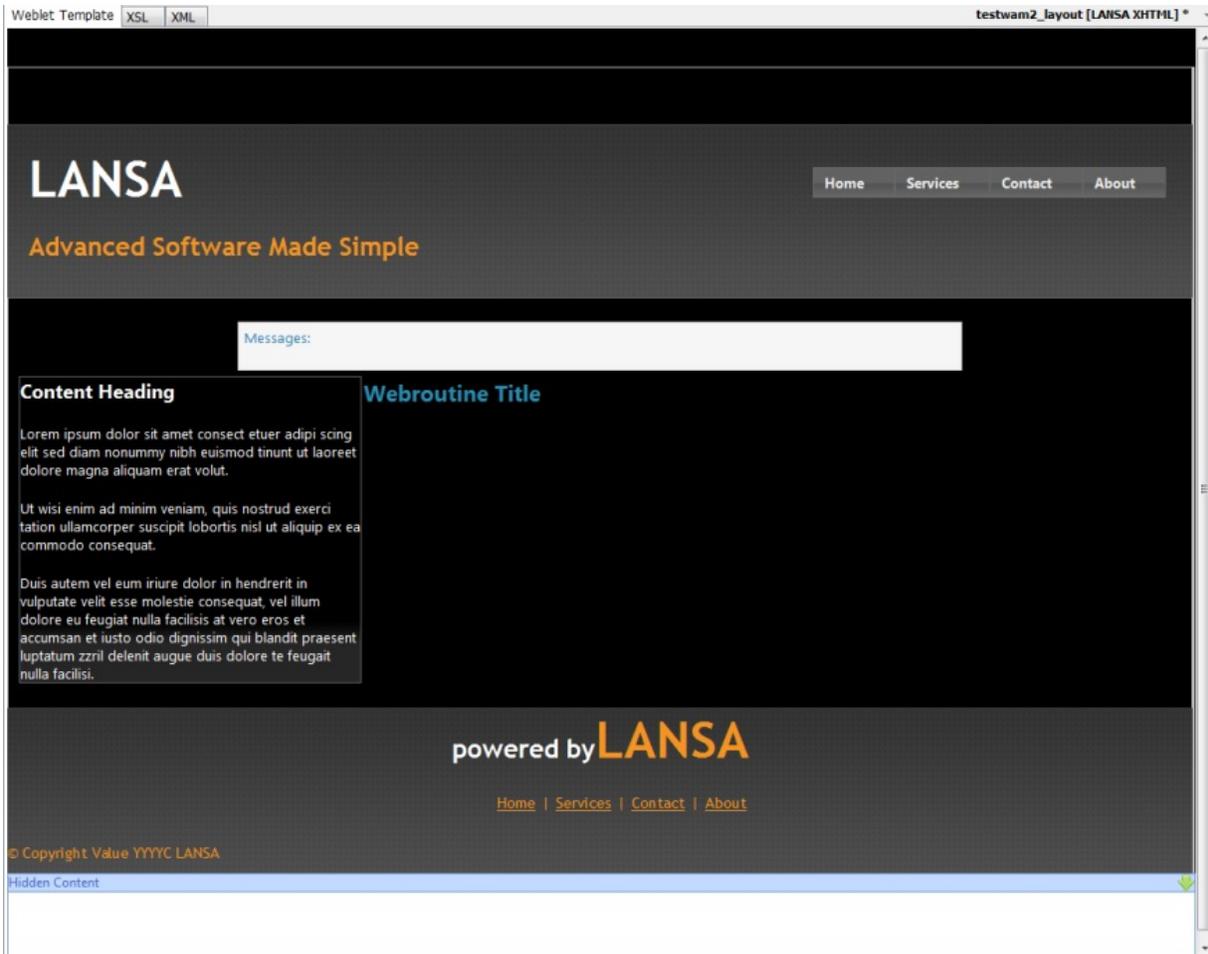


このイメージからも分かる通り、一般的にレイアウトは全てのWebroutineページのヘッダーとフッター、メッセージ・エリア、コンテンツ・エリアそして色テーマやスタイルをコントロールします。JavaScriptをレイアウトレベルで適用させ、そのレイアウトを使う全てのWebroutineに適用させることもできます。

特定のWebroutineに関連付けられたWEB_MAPとウェブレットは、レイアウトのコンテンツ・エリア(Webroutineタイトルの下)に表示されます。

std_themelet1_2colレイアウト・ウェブレットは基本的にはstd_themelet1_1colレイアウト・ウェブレットと同じですが、コンテンツ・エリアが2つあります。std_themelet1_2colレイアウト・ウェブレットは絶対に修正しないでください。

std_themelet1_2colレイアウト・ウェブレットを使って作成されたWAMレイアウトはエディターで開くと、次のようになります。(テーマはDarkness、スタイル#2のテーマレットを使用)



std_themelet1_3colレイアウト・ウェブレットは基本的にはstd_themelet1_1colレイアウト・ウェブレットと同じですが、コンテンツ・エリアが3つあります。std_themelet1_3colレイアウト・ウェブレットは絶対に修正しないでください。

std_themelet1_3colレイアウト・ウェブレットを使って作成されたWAMレイアウトはエディターで開くと、次のようになります。(テーマはSouth Street、スタイル #1のテーマレットを使用)

LANSA

Advanced Software Made Simple

[Home](#) [Services](#) [Contact](#) [About](#)

Messages:

Menu Link

Menu Link

Menu Link

Menu Link

Menu Link

Webroutine Title

Content Heading

Lorem ipsum dolor sit amet consectetur adipiscing elit sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volut.

Ut wisi enim ad minim veniam, quis nostrud exercitation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.

powered by **LANSA**

[Home](#) | [Services](#) | [Contact](#) | [About](#)

プロパティ - 標準テーマ・レイアウト (std_themelet1_[1-3]col)

標準テーマ・レイアウト(std_themelet1_[1-3]col) ウェブレットのプロパティは以下の通りです。

| | | |
|------------------|------------------|----------------|
| Backcompat_theme | jQueryNoConflict | sidebar2_width |
| content_side | onload_script | title_text |
| content_width | output_charset | width |
| css_files | show_title | width_type |
| has_form | sidebar_width | window_title |
| javascript_files | sidebar1_width | |

Backcompat_theme

std_styleに引き渡すテーマが""かまたは'none'なのかを決定するブール値プロパティです。評価がtrueの場合、レイアウトとウェブレットがV12SP1以前の既存のアプリケーションと一貫性が取れるよう、下位互換性のあるテーマが適用されます。falseの場合、現在は外部リソースを使ってテーマを適用するので、テーマは適用されません。

省略値

false

有効値

trueまたはfalse。

例

```
r[ ] backcompat_theme False
```

content_side

別のコンテンツ・エリアに対し、WAMレイアウト内のメインのコンテンツ・エリアをどこに位置付けるかを指定します。これは2列または3列のレイアウトの時にのみ適用されます。

省略値

Center

有効値

Left、CenterまたはRight

例

 content_side center

content_width

メイン・コンテンツの幅としてCSSユニットに指定する値です。相対値、絶対値またはパーセント値が指定できます。これは2列または3列のレイアウトの時にのみ適用されます。

省略値

2列のレイアウトでは70%、3列のレイアウトでは50%です。

有効値

単一引用符で囲まれた有効な範囲内の幅。

例

```
content_width 50%
```

css_files

レイアウトに適用される複数のカスケード・スタイル・シート(CSS) ファイルをカンマで区切ったリスト。これらのファイルは、theme_css_filenameに定義されるファイルに適用されます。このプロパティは1つのスタイルシートでは十分ではない場合など、特別な状況のために用意されています。

レイアウトでCSSを使用する際の詳しい情報は、[『カスケード・スタイル・シート\(CSS\) とスタイル・ウェブレット』](#)を参照してください。

省略値

"

有効値

スタイル仮想ディレクトリに関連付けられたパスとCSSファイル名をカンマで区切ったリストで、単一引用符で囲まれたもの。

has_form

ブール値プロパティです。falseと評価された場合に、省略値のLANSAフォーム・エレメントを削除する式。

省略値

true() (LANSAフォーム・エレメントはページに含まれます。)

有効値

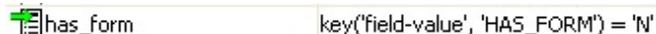
true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

例

この例は#HAS_FORMフィールドが'N'に等しい場合、省略値のLANSAフォームを削除します。式は必須です。またプロパティにフォーカスがある場合、以下のように表示されます。

has_form #has_form = 'N'

プロパティがフォーカスを失うと、この式は以下のように表示されます。

has_form key('field-value', 'HAS_FORM') = 'N'

output_charset

出力文書の文字セット用エンコード識別子(iso-8859-1やwindows-1251など)。電子メールやPDF文書などを使用する時など、事前に文字セットを変換しなければいけない場合があります。省略値のcharsetがこれに相当する場合、出力が変換される文字セット名がこの属性になります。通常はブランクのままにし、省略値を使用します。

省略値

`/lxml:data/lxml:server-instructions/lxml:client_charset`

有効値

単一引用符で囲まれた文字セット値。

show_title

ブール値プロパティです。trueと評価された際に、タイトルが表示される式。

省略値

true() (タイトルは常に表示されます)

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

例

この例はフィールド#SHOW_TITLEが 'Y' と等しい場合にタイトルを表示します。式は必須です。またプロパティにフォーカスがある場合、以下のように表示されます。

```
show_title #SHOW_TITLE = 'Y'
```

プロパティがフォーカスを失うと、この式は以下のように表示されません。

```
show_title key('field-value', 'SHOW_TITLE') = 'Y'
```

onload_script

ページがブラウザにロードされた後に実行されるJavaScript(body tag onload イベント)です。

省略値

'SetFocus()' (ページがロードされた時、標準スクリプトに含まれるファンクションSetFocus()が実行されます。)

有効値

単一引用符で囲まれた有効なJavaScriptファンクション、またはセミコロン(;)が後ろに付いたJavaScriptコード。

title_text

Webroutineページ上に表示されるタイトルのテキスト。

省略値

/lxml:data/lxml:context/lxml:webroutine-title (事実上空白)。

有効値

単一引用符で囲まれたテキスト、もしくは複数言語対応テキスト変数、システム変数、フィールドの名前(プロパティ・シートで省略記号のボタンをクリックしてリストから選ぶことができます。)

javascript_files

レイアウトにロードされる複数のJavaScriptファイルをカンマで区切ったリスト。このプロパティにはカスタムのJavaScriptやサード・パーティのコントロールに関連付けられたプログラムも含めることができます。

レイアウトでJavaScriptを使用する際の詳細は『[JavaScript と スクリプト・ウェブレット](#)』を参照してください。

省略値

"

有効値

script仮想ディレクトリに関連付けられたパスとJavaScriptファイル名をカンマで区切ったリストで、単一引用符で囲まれたもの。

width

ページ幅としてCSSユニットに指定される値です。相対値、絶対値またはパーセント値が指定できます。

省略値

1000px

有効値

単一引用符で囲まれた有効な範囲内の幅。

例

`width` `1000px`

width_type

CSSのレイアウト・タイプです。

Fixed(固定幅)レイアウトは、クライアント・ウィンドウのサイズに関係なく指定された特定の値に幅が設定されます。レイアウトのコンテンツは、ウィンドウがサイズ調整されても調整されません。

Fluid(可変幅)レイアウトは、クライアント・ウィンドウの幅に合うようにレイアウトのコンテンツが自動的に調整されます。例えば、100%の可変幅レイアウトはスクロール・バーなしで幅全体を埋めます。小さいウィンドウにサイズ調整された場合にのみ、スクロール・バーが表示されます。

省略値

1列のレイアウトの場合はFixed、2列または3列のレイアウトの場合はFluid。

有効値

FixedまたはFluid.

例

`width_type` *Fixed*

sidebar_width

CSSユニットに指定する、2列レイアウトのコンテンツ・エリアのサイドバー幅の値。相対値、絶対値またはパーセント値が指定できます。

省略値

29%

有効値

単一引用符で囲まれた有効な範囲内の幅。

例

```
sidebar_width 29%
```

sidebar1_width

CSSユニットに指定する、3列レイアウトのコンテンツ・エリアのサイドバー1の幅の値。相対値、絶対値またはパーセント値が指定できます。

省略値

19%

有効値

単一引用符で囲まれた有効な範囲内の幅。

例

```
sidebar1_width 19%
```

sidebar2_width

CSSユニットに指定する、3列レイアウトのコンテンツ・エリアのサイドバー2の幅の値。相対値、絶対値またはパーセント値が指定できます。

省略値

30%

有効値

単一引用符で囲まれた有効な範囲内の幅。

例

```
sidebar2_width 30%
```

jQueryNoConflict

jQueryの利用時のみに適用されるブール値プロパティ。trueと評価された場合、jQuery.noConflict()が呼び出され、\$の名前がなくなります。

jQueryのコアを入れる前に、\$名を使用する他のライブラリを入れておく必要があります。

省略値

false

有効値

trueまたはfalse。

例

```
jQueryNoConflict False
```

window_title

Webページのクライアント・ウィンドウに表示されるタイトルのテキスト。

省略値

WAMの記述。

有効値

単一引用符で囲まれたテキスト、もしくは複数言語対応テキスト変数、システム変数、フィールドの名前(プロパティ・シートで省略記号のボタンをクリックしてリストから選ぶことができます。)

例

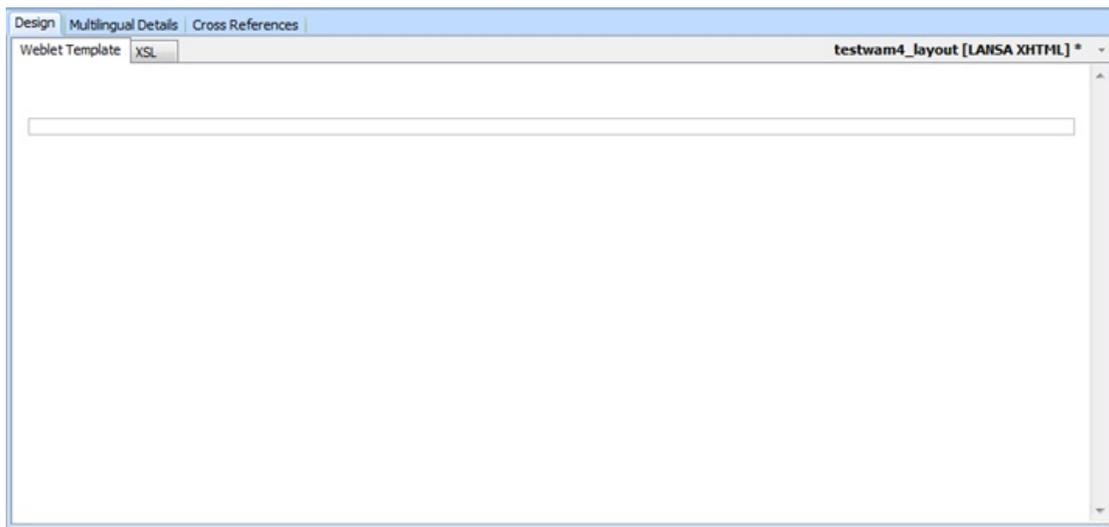
 window_title *\$web_context/xml:webapplication-title*

8.4.3 標準ブランク・レイアウト (std_blank_layout)

std_blank_layoutレイアウト・ウェブレットは、ヘッダーやフッター・エリアの定義もされず、配色もされていない、簡素なレイアウトです。レイアウトやCSSのないWebroutineの表示に使用されます。

std_blank_layoutレイアウト・ウェブレットは絶対に修正しないでください。

std_blank_layoutレイアウト・ウェブレットを使って作成されたWAMレイアウトはエディターで開くと、次のようになります。



このブランク・レイアウトを導入することで、レイアウト/CSSのないWebroutineの表示方法が簡素化されます。デザイン・ビューでレイアウト・ウェブレットを選択する代わりに、プロパティ、no_layoutをtrueに設定しても構いません。

次のように簡素化できます。

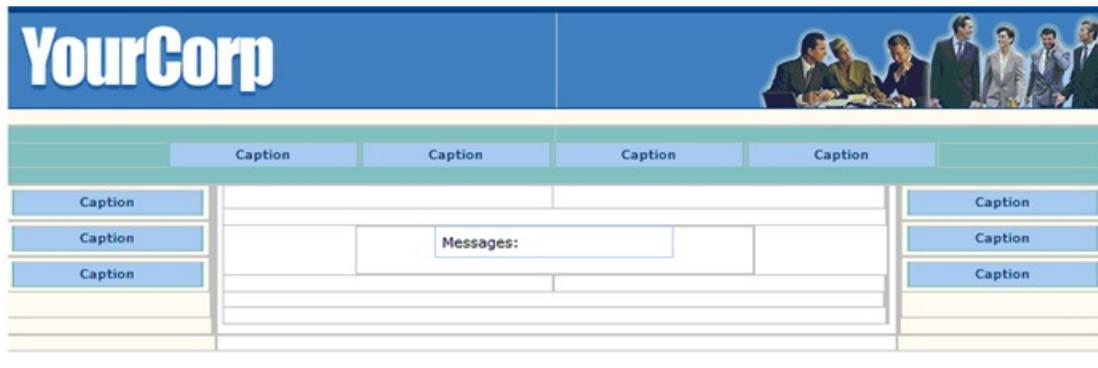
std_blank_layoutウェブレットをWebroutineにドラッグします。これにより、以前のレイアウトと置き換えられます。

8.4.4 標準基本レイアウト (std_layout_V2 and std_layout[1-5]_v2)

Std_layout_v2はヘッダーとフッター・エリアが定義された基本的なレイアウトです。

Standard layouts 1-5 も基本的に同じですが、表示するメニューを決定する異なる設定があります。(std_layout1_v2, std_layout2_v2 ... std_layout5_v2)

std_layout1ウェブレットをエディターで開くと、次のようになります。



この図からも解るように、一般的にレイアウトはページのメニュー、バナー、イメージ、メッセージ・ボックス、配色を制御します。JavaScriptをレイアウトレベルで適用させて、そのレイアウトを使う全てのWebroutineに適用させることも可能です。

特定のWebroutineに関連付けられたWEB_MAPやウェブレットは、レイアウトの本体(メッセージ・ボックスの下)に表示されます。

プロパティ - 標準レイアウト (std_layout_V2 および std_layout[1-5]_v2)

標準レイアウト・ウェブレットのプロパティは以下のとおりです。

| | | |
|------------------|-----------------|--------------------|
| body_class | onload_script | show_title |
| css_files | onunload_script | show_top_menu |
| form_class | output_charset | theme_css_filename |
| has_form | show_left_menu | title_text |
| javascript_files | show_messages | trap_script_errors |
| no_layout | show_right_menu | |

show_left_menu

`std_layout`では使用できません。

ブール値プロパティです。trueと評価された時に左メニューを表示する式。

省略値

レイアウトによって異なります。

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

例

この例では、フィールド#LEFT_MENUが'Y'と等しい場合、左メニューを表示します。式は必須です。またプロパティにフォーカスがある場合、以下のように表示されます。

```
show_left_menu #LEFT_MENU = 'Y'
```

プロパティがフォーカスを失うと、この式は以下のように表示されません。

```
show_left_menu key('field-value', 'LEFT_MENU') = 'Y'
```

show_top_menu

`std_layout`では使用できません。

ブール値プロパティです。trueと評価された時に、トップ・メニューを表示する式。

省略値

レイアウトによって異なります。

有効値

`true()`、`false()`、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、`true()` もしくは `false()` で答えが得られる有効な式。

例

この例はフィールド`#TOP_MENU`が 'Y' と等しい場合にトップ・メニューを表示します。式は必須です。またプロパティにフォーカスがある場合、以下のように表示されます。

```
show_top_menu #TOP_MENU = 'Y'
```

プロパティがフォーカスを失うと、この式は以下のように表示されません。

```
show_top_menu key('field-value', 'TOP_MENU') = 'Y'
```

show_right_menu

*std_layout*では使用できません。

ブール値プロパティです。trueと評価された時に、右メニューを表示する式。

省略値

レイアウトによって異なります。

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

例

この例はフィールド#RIGHT_MENUが 'Y' と等しい場合に右メニューを表示します。式は必須です。またプロパティにフォーカスがある場合、以下のように表示されます。

```
show_right_menu #RIGHT_MENU = 'Y'
```

プロパティがフォーカスを失うと、この式は以下のように表示されま

```
show_right_menu key('field-value', 'RIGHT_MENU') = 'Y'
```

body_class

ページのボディ・エレメントに適用されるカスケード・スタイル・シートのクラス。

省略値

空白。

有効値

単一引用符で囲まれた、現在のカスケード・スタイル・シートの有効なクラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

例

```
body_class="redbg"
```

form_class

ページのフォーム・エレメントに適用されるカスケード・スタイル・シートのクラス。

省略値

空白。

有効値

単一引用符で囲まれた、現在のカスケード・スタイル・シートの有効なクラス名。プロパティシートで該当するドロップダウン・ボタンをクリックして、使用可能なクラスの一覧から選択することができます。

has_form

*std_layout*では使用できません。

ブール値プロパティです。falseと評価された場合に、省略値のLANSAフォーム・エレメントを削除する式。

省略値

true() (LANSAフォーム・エレメントはページに含まれます。)

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくはfalse()で答えが得られる有効な式。

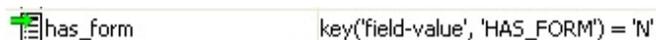
例

この例は#HAS_FORMフィールドが'N'に等しい場合、省略値のLANSAフォームを削除します。式は必須です。またプロパティにフォーカスがある場合、以下のように表示されます。



A screenshot of a table with two columns. The first column is labeled 'has_form' and the second column contains the value '#has_form = 'N''. The row is highlighted in blue.

プロパティがフォーカスを失うと、この式は以下のように表示されま



A screenshot of a table with two columns. The first column is labeled 'has_form' and the second column contains the value 'key('field-value', 'HAS_FORM') = 'N''. The row is highlighted in light blue.

no_layout

*std_layout*では使用できません。

ブール値プロパティです。trueと評価された時に、レイアウトのアウトラインを取り除く式。

省略値

false() (レイアウトのアウトラインはページに含まれます)

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false() で答えが得られる有効な式。

例

この例では、#NO_LAYOUTフィールドが'Y'と等しい場合、レイアウトのアウトラインを取り除きます。式は必須です。またプロパティにフォーカスがある場合、以下のように表示されます。

```
#no_layout #no_layout = 'Y'
```

プロパティがフォーカスを失うと、この式は以下のように表示されま

```
#no_layout key('field-value', 'NO_LAYOUT') = 'Y'
```

show_title

ブール値プロパティです。trueと評価された際に、タイトルが表示される式。

省略値

true() (タイトルは常に表示されます)

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

例

この例はフィールド#SHOW_TITLEが 'Y'と等しい場合にタイトルを表示します。式は必須です。またプロパティにフォーカスがある場合、以下のように表示されます。

```
show_title #SHOW_TITLE = 'Y'
```

プロパティがフォーカスを失うと、この式は以下のように表示されません。

```
show_title key('field-value', 'SHOW_TITLE') = 'Y'
```

title_text

Webroutineページ上に表示されるタイトルのテキスト。

省略値

/lxml:data/lxml:context/lxml:webroutine-title (事実上空白)。

有効値

単一引用符で囲まれたテキスト、もしくは複数言語対応テキスト変数、システム変数、フィールドの名前(プロパティ・シートで省略記号のボタンをクリックしてリストから選ぶことができます。)

show_messages

ブール値プロパティです。trueと評価された時に、Webroutineページにメッセージ・ボックスを表示する式。

省略値

true() (メッセージ・ボックスは常に表示されます)

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

例

この例では、フィールド#SHOW_MESSAGESが'Y'と等しい場合、メッセージ・ボックスを表示します。式は必須です。またプロパティにフォーカスがある場合、以下のように表示されます。

```
show_messages #show_msg = 'Y'
```

プロパティがフォーカスを失うと、この式は以下のように表示されま

```
show_messages key('field-value', 'SHOW_MSG') = 'Y'
```

onload_script

*std_layout*では使用できません。

ページがブラウザにロードされた後に実行されるJavaScript(body tag onload イベント)です。

省略値

'SetFocus()' (ページがロードされた時、標準スクリプトに含まれる
ファンクションSetFocus()が実行されます。)

有効値

単一引用符で囲まれた有効なJavaScriptファンクション、またはセミ
コロン(;)が後ろに付いたJavaScriptコード。

onunload_script

*std_layout*では使用できません。

ページがブラウザからアンロードされた後に実行されるJavaScript(つまり、body tag onunload イベント)。

省略値

"

有効値

シングル・クォーテーションつきの、有効なJavaScriptファンクション、もしくはセミコロン(;)が付いたJavaScriptコード。

javascript_files

レイアウトにロードされる複数のJavaScriptファイルをカンマで区切ったリスト。このプロパティにはカスタムのJavaScriptやサード・パーティのコントロールに関連付けられたプログラムも含めることができます。

レイアウトでJavaScriptを使用する際の詳細は『[JavaScript と スクリプト・ウェブレット](#)』を参照してください。

省略値

"

有効値

script仮想ディレクトリに関連付けられたパスとJavaScriptファイル名をカンマで区切ったリストで、単一引用符で囲まれたもの。

theme_css_filename

スタイル仮想ディレクトリに関連付けられたカスケード・スタイルシート(CSS)ファイル名で、レイアウトおよびそのレイアウトに從属するウェブレットに適用されるもの。これにより、HTMLエレメント及び從属するウェブレットのクラス・プロパティの選択時に使用するCSSクラス・セレクターのスタイル・プロパティが決定されます。

レイアウトでCSSを使用する際の詳しい情報は、『[カスケード・スタイル・シート\(CSS\) と スタイル・ウェブレット](#)』を参照してください。

省略値

"

有効値

単一引用符で囲まれた、スタイル仮想ディレクトリに関連付けられたパスとCSSファイル名。ファイルは、プロパティ・シートの対応する省略記号のボタンをクリックしてプロンプターから選択することができます。

css_files

レイアウトに適用される複数のカスケード・スタイル・シート(CSS) ファイルをカンマで区切ったリスト。これらのファイルは theme_css_filenameに定義されたファイルに適用されます。このプロパティは1つのスタイルシートでは十分でない時など、特別な状況の為に作られています。

レイアウトでCSSを使用する際の詳しい情報は、『[カスケード・スタイル・シート\(CSS\) とスタイル・ウェブレット](#)』を参照してください。

省略値

"

有効値

スタイル仮想ディレクトリに関連付けられたパスとCSSファイル名をカンマで区切ったリストで、単一引用符で囲まれたもの。

output_charset

出力文書の文字セット用エンコード識別子(iso-8859-1やwindows-1251など)。電子メールやPDF文書などを使用する時など、事前に文字セットを変換しなければいけない場合があります。省略値のcharsetがこれに相当する場合、出力が変換される文字セット名がこの属性になります。通常は空白のままにし、省略値を使用します。

省略値

`/lxml:data/lxml:server-instructions/lxml:client-charset`

有効値

単一引用符付きの、文字セット値。

trap_script_errors

ブール値プロパティです。trueと評価された時に、JavaScriptエラーを抑制する式。

省略値

true() (JavaScriptエラーが抑制されます)

有効値

true()、false()、もしくは、フィールド名、リテラル、XSL変数やJavaScript変数を使用した、true() もしくは false()で答えが得られる有効な式。

例

この例では、フィールド#TRAP_ERRORSが'Y'の場合、JavaScriptエラーを抑制します。式は必須です。またプロパティにフォーカスがある場合、以下のように表示されます。

```
trap_script_errors #TRAP_ERRORS = 'Y'
```

プロパティがフォーカスを失うと、この式は以下のように表示されます。

```
trap_script_errors key('field-value', 'TRAP_ERRORS') = 'Y'
```

8.4.5 ユーティリティ・ウェブレット

ウェブレット名 説明

| | |
|--------------------------------|--|
| 標準非表示
フィールド
(std_hidden) | LANSA製品センター内部専用です。詳しくは、『 非表示 』を参照してください。 |
| 標準地域固有
(std_locale) | LANSA製品センター内部専用です。詳しくは、『 非表示 』を参照してください。 |
| 標準スクリプト
(std_script) | LANSA製品センター内部専用です。詳しくは、『 JavaScriptとスクリプト・ウェブレット 』を参照してください。 |
| 標準変数
(std_variables) | LANSA製品センター内部専用です。詳しくは、『 変数 』を参照してください。 |
| std_keys | XSLのフィールド参照で使用するxsl:keyエレメントのセット。詳しくは、『 変数 』を参照してください。 |
| std_select_list | std_dropdownとstd_listboxが使用するウェブレット。直接の使用はしないでください。 |
| std_types | LANSA製品センター内部専用です。詳しくは、『 タイプ 』を参照してください。 |
| std_util | LANSA製品センター内部専用です。 |
| CSSスタイル
(std_style_v2) | 生成したすべてのWebroutineページに使用する省略値スタイル |

8.4.6 インライン・テンプレート

インライン・テンプレート・グループにより、使用中のテクノロジー・サービスが"インライン認識"する全てのウェブレットに素早くアクセスできるようになります。"インライン認識"されたウェブレットには、.inlineという接尾辞がついたウェブレットテンプレートが含まれます。

インライン・テンプレート・グループに含まれる全てのウェブレットは、[8.1 標準ウェブレット](#)または[8.3 標準フィールド・ビジュアライゼーション](#)のグループでも使用可能です。

8.4.7 jQuery UI

jQuery UIグループにより、jQuery UIが作動する必要がある全ての使用中のウェブレットに素早くアクセスできるようになります。jQuery UIグループに含まれる全てのウェブレットは、[8.1 標準ウェブレット](#)や[8.3 標準フィールド・ビジュアルイゼーション](#)で使用可能です。

8.4.8 WAMレイアウト

WAMレイアウトは、既に存在する場合を除き、<WAM名>_layout という名前で、XSLの生成時に作成されます。WAMのBEGIN_COMで、LAYOUTWEBLETセクターを使用して、WAMレイアウトで使用するレイアウト・ウェブレットを指定します。省略値では、生成されたWAMレイアウトはウェブレット・テンプレート・グループのWAMレイアウトに含まれます。

詳しい情報については、[『WAMレイアウトとレイアウト・ウェブレット』](#)を参照してください。

9. jQuery Mobileテクノロジー・サービス用のウェブレット

jQuery Mobileは、一般的に使用される全てのモバイル機器のプラットフォームのための、jQuery と jQuery UI をベースに構築されたUIHTML5形式の統合ユーザー・インターフェースシステムです。軽量のコードは、機能が拡張されて構築されており、テーマ構成が容易な順応性のあるデザインになっています。

jQuery Mobile技術サービスはJavaScriptライブラリのラッパーで、プロパティを編集して構成されるドラッグ・アンド・ドロップ可能ウェブレットを使って、LANSAのエディター内でのライブラリの使用を可能にします。

ウェブレット・テンプレートのリポジトリ・ビューにはテンプレートがグループごとに表示されます。1つのウェブレット・テンプレートが複数のグループに属する場合があります（構成はウェブレットで右クリック）。標準のグループは以下の通りです。

[標準ウェブレット](#)

[レイアウト・ウェブレット](#)

以下も参照してください。

[jQuery Mobile および WAM エディター](#)

[フィールドの検証](#)

[省略値のウェブレット](#)

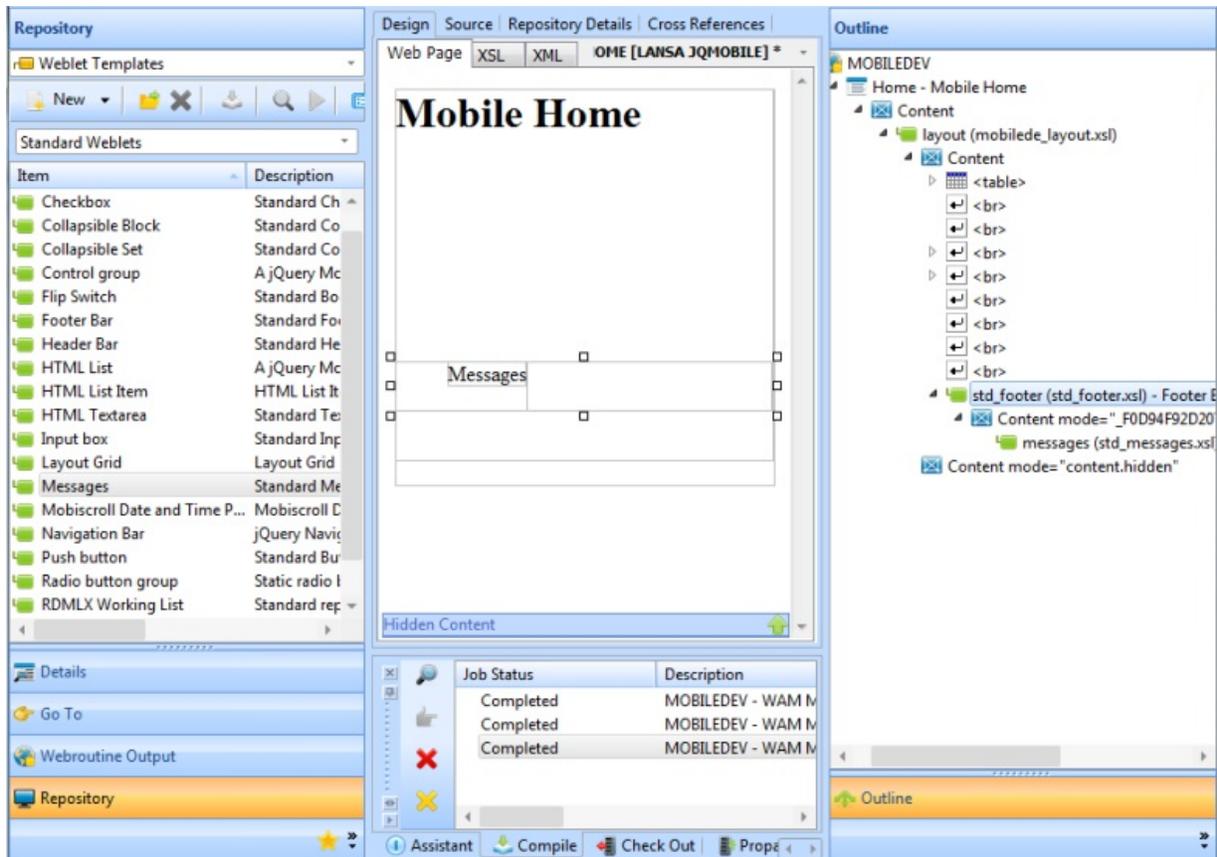
[ユーティリティ・ウェブレット](#)

9.1 jQuery Mobile および WAM エディター

実行時にブラウザ内で全ての処理を行う時、jQuery MobileフレームワークはJavaScriptに大きく依存します。開始点として最小限のHTMLを必要とし、HTMLが実行されているブラウザのニーズと能力により、HTMLの修正にJavaScriptを使用します。

この実行時の強化により、ユーザーが使う可能性のあるXHTMLのような別の技術サービスのグラフィカル編集のコンセプトに問題が起こりません。WAMエディターは実行時にJavaScriptを実行できないため、jQuery Mobileの開始点を提供する基本のHTMLを表示し編集することしかできません。

ウェブレットの多くはコンテナ・ウェブレットです。つまり、カスタマイズ・コンテンツとサーバーのラッパーとして機能し、そのカスタマイズされたコンテンツの表示やふるまいを修正します。例としてヘッダーとフッター・ウェブレットがあります。これらはヘッダーもしくはフッターの特定の配置と表示の特性を定義しますが、ヘッダーもしくはフッター内に入れるコンテンツは、ユーザーが定義できます。シンプル・テキスト、別のウェブレットもしくはユーザーのカスタムHTMLがコンテンツとなります。デザイナーをクリックすると実際にコンテンツを選択することになるため、デザイナーをクリックしてデザイナー内でコンテナ・ウェブレットを選択することはできません。これらのウェブレットを選択しそのプロパティを編集するためには、アウトライン・タブを使います。



アウトラインを使ったフッター・ウェブレットの選択

9.2 フィールドの検証

HTML 5の仕様にはフォームの検証に使われる多数の機能が含まれています。フォームのエレメントには、必須、最小、最大、パターンおよび新しいデータ・タイプなどの新しい属性があり、ブラウザに無効なデータ入力を防ぐUIを表示させるか、もしくは入力されたデータを検証し、誤ったフォームの送信を防ぎます。LANSAのフレームワークにはRDMLXデータ・タイプの検証のサポートと機能が追加されており、ブラウザの実行における矛盾に対処します。

以下も参照してください。

[9.2.1 RDMLX データ・タイプ](#)

[9.2.2 検証エラーの表示](#)

[9.2.3 検証が発生した場合の制御](#)

9.2.1 RDMLX データ・タイプ

RDMLX データ・タイプの検証は、data-lstddatatype属性を<入力>、<選択> もしくは <テキストエリア> エlementに追加することにより、どのElementにも追加できます。属性値はデータ・タイプで始まり、データ・タイプで必要とされる追加のパラメータが続く、|で区切られた文字列でなければいけません。

integer|<最大長>

float|<最大長>

packed|<合計桁数>|<小数桁数>|<小数点記号>

signed|<合計桁数>|<小数桁数>|<小数点記号>

dec|<合計桁数>|<小数桁数>|<小数点記号>

alpha|<キーボード・シフト>|<最大長>

char|<キーボード・シフト>|<最大長>

varchar|<キーボード・シフト>|<最大長>

nchar|<キーボード・シフト>|<最大長>

nvarchar|<キーボード・シフト>|<最大長>

たとえば、以下のRDMLXを使ってフィールドが定義されているとします。

```
Define Field(#TST_PKD) Type(*PACKED) Length(6) Decimals(2)
```

<入力> タグを次のように使うことができます。

```
<input id="MyWR_TST_PKD" name="TST_PKD" maxlength="6" size="11" data-lstddatatype="packed|6|2|" type="number" />
```

新しいウェブルーチンを生成する、もしくはデザインにフィールドをドロップする場合、WAMエディターは自動的にこの属性を設定します。フォーム・Elementを作成する標準ウェブレットは、rdmlxDatatypeプロパティの値に基づいてこの属性を設定します。

9.2.2 検証エラーの表示

最新のデスクトップ・ブラウザは、エラー・メッセージのあるフィールドの上にヒントを表示することにより、検証エラーを表示します。通常、一度に1つのみエラーが表示されます(さらにエラーを確認するため、もう一度送信を要求します)。入力中もしくはユーザーがフィールドからタブアウトする場合など、別の時点でエラーを表示するオプションはありません。モバイル機器のブラウザは、書き込み時には表示をしません。無効なフォームの送信をさせないため、問題を説明するエラーは表示しません。

LANSA フレームワークはページ内での検証エラー表示のオプションを、通常フィールドの次に追加します。このふるまいを有効にするには、ユーザーのレイアウト内の <div> ページに "lstdErrorShowInDiv" クラスを追加してください。提供されている標準レイアウトでは、validationErrorDisplay プロパティを使って検証エラーを表示します。エラーを表示する <div> タグも提供してください。各フィールドに1つのエラー<div>があるはずです。<div> には "{Field ID}_id" のID と "lstdFieldError" のクラスが必要です。例えば次のようになります。

```
<div id=" MyWR_TST_PKD _error" class="lstdFieldError"></div>
```

標準フィールド・ウェブレットは、addErrorDiv プロパティを使ってこれを実行します。"show in div" がオフに設定されていると、全てのエラーdivが自動的に非表示になります。"show in div" が有効になっている場合、全てのエラー部分が非表示に設定され、ユーザーには見えませんが、スペースはドキュメント内で確保されたままになっています。これにより、後からエラーが表示された場合にページがリフローする混乱を防ぎます。フィールド内で検証エラーが起こった場合、フレームワークは対応するエラーdivの中にエラーメッセージを入れて表示させます。

9.2.3 検証が発生した場合の制御

ブラウザの省略値のふるまいは、ユーザーがフォームを送信すると検証を実行します。LANSA フレームワークは2つの制御の実行オプションを提供します。1つはユーザーがフィールドにデータを入力した後で、もう1つはユーザーの入力中すぐです。これらのオプションを有効化するには、`stdErrorImmediately` もしくは `lstdErrorAfterFocus` のクラスのどちらか1つを `<div>` ページに追加します。提供されている標準レイアウトでは、`validationTime` プロパティを使ってオプションを有効化します。

ほとんどの検証がこれら以外の時に実行されるため、特定のシナリオに例外があるブラウザがあります。例えば、Safariは、ユーザーがフォーカス後に要求されたフィールドにデータ入力を試みない限り、そのフィールドを無効にしません。これにより、ユーザーはフォームに入力中、エラーの影響を受けずにフィールド内でタブキーを押して入力することができます。

9.3 省略値のウェブレット

jQuery Mobileには反応が速い設計を可能にする機能があります。その機能は、特別なラッパー・フィールドのある入力フィールドとjQuery Mobileデータ属性に依存します。ウェブレットを使うと実行が容易になります。そのためJQMOBILEテクノロジー・サービスでは、ウェブレットのビジュアルライゼーションの省略値を持たないリポジトリ・フィールドについて、`std_input`を省略値として定義します。

9.4 標準ウェブレット

プロパティやウェブルーチン内での使用方法などの標準ウェブレットの詳細を以下に説明します(アルファベット順)。ウェブレットで使用できる全てのプロパティを使用する必要はありません。

| | |
|---|--|
| アンカー
(std_anchor) | ハイパーリンク(もしくはアンカー)コントロールを提供し、ハイパーテキストのリンク先を指定するHTMLエレメントに幅広く対応します。 |
| ブール値
(std_boolean) | このウェブレットは、jQuery Mobileをスライディング・スイッチに変換するHTMLエレメントを生成します。 |
| プッシュ ボタン
(std_button_v2) | クリックできるボタンを作成し、(入力ボックス・ウェブレットを使って作成されたボタンとはちがいに)フォーマットされたテキストとイメージのようなカスタムHTMLを含むことができます。 |
| チェックボックス
(std_checkbox) | jQuery Mobileが現在のテーマに合うように修正する標準チェックボックス・コントロールです。 |
| 折りたたみ可能ブロック
(std_collapsible) | ヘッダーをクリックすることにより、ユーザーが表示もしくは非表示にできるコンテンツのセクションです。 |
| 折りたたみ可能セット
(std_collapsibleset) | 複数の折りたたみ可能ブロック・ウェブレットを囲んで配置され、折りたたみを作成するラッパーです。折りたたみ可能ブロック・ウェブレットを作成し、一連の折りたたみ可能ウェブレットのコンテンツ・エリアに入れます。 |
| コントロール・グループ
(std_controlgroup) | 複数のボタン、チェックボックスもしくはラジオ・ボタンの周囲に配置され、グループ化するラッパーです。 |
| フッター バー
(std_footer) | ページ下部のQuery Mobileツールバーです。ユーザーが画面をタップしてページがスクロールされたり、ページが構成されて表示もしくは非表示になったりする時、ツールバーは固定され、同じ位置にとどまることができます。 |

| | |
|--|--|
| ヘッダーバー
(std_header) | ページ上部の Query Mobile ツールバーです。ユーザーが画面をタップしてページがスクロールされたり、表示もしくは非表示になったりする時、ツールバーは画面上部に固定され、同じ位置にとどまることができます。 |
| HTML リスト
(std_html_list) | データ表示、ナビゲーション、結果リストとデータ入力に使われるさまざまなリストとフォーマットのタイプです。 |
| HTML リスト項目
(std_html_li) | HTML リスト・ウェブレットに項目を追加します。 |
| HTML テキストエリア
(std_textarea) | 長いテキスト値を複数行に分けて表示と入力できるようにするエレメントです。 |
| 入力ボックス
(std_input) | このウェブレットは、HTML のページへのデータ入力の基本フォームとなるエレメントを作成します。複数行のテキスト・フィールドを除き、HTML がサポートするあらゆるタイプの入力コントロール(ボタンを含む)の作成に使われます。 |
| レイアウトグリッド
(std_gridlayout) | 小さいエレメントを並べて配置したい場合に使用します(ボタンやナビゲーション・タブなど)。 |
| メッセージ
(std_messages) | RDMLX Message コマンドを使って作成されたメッセージを表示します。 |
| Mobiscroll 日付/時間ピッカー
(std_mobiscroll) | モバイル機器のために最適化されたカスタマイズ可能な日時ピッカーです。 |
| ナビゲーションバー
(std_navbar) | オプションのアイコンを使って最大で5つのボタンを提供できる基本のウィジェットです。 |
| ラジオ ボタングループ
(std_radbuttons) | RDMLX フィールドにリンクされた一連のラジオ・ボタンを作成します。 |
| RDMLX 作業リスト | 作業リストを処理する柔軟なメカニズムを提供します。 |

(std_repeater)

コンボ ボックス 非表示で、jQuery Mobileフレームワークの外観にあ
選択メニュー うカスタム・スタイルの選択ボタンで置換えられた
(std_dropdown) ネイティブの select エlementに基づいています。

9.4.1 アンカー(std_anchor)

アンカー・ウェブレットはハイパーリンク(あるいはアンカー)コントロールを提供します。ハイパーテキストのリンク先を指定する <a> (アンカー)HTMLエレメントとほぼ同じ働きをします。

- アンカー・ウェブレットは、その一番シンプルなフォームにおいて、タップされた場合にブラウザを新しいページにナビゲートするテキスト値を表示します。
- アンカー・ウェブレットはボタンとして表示されるように構成されるか、もしくは(別のウェブレットを含む)任意のHTMLを含みます。
- 移動先はurl (例: <http://www.yourcompany.com/>)とするか、もしくはユーザーが実行されるWAMとウェブルーチンを指定し、オプションとしてウェブルーチンに値を引き渡すフィールドを特定することができます。
- アンカーは常にGET HTTP要求を生成し、onclickExtraFieldsプロパティ内で明確に指定されない限り、フォーム・フィールドを送信しません。アンカーは外部サイト、静的ページもしくはクエリを実行するウェブルーチンへのリンクのみに使ってください。データを生成もしくは更新するトランザクションを実行する場合、送信ボタンを使用してください。
- HTML 5フォームの検証は、アンカーをクリックして送信されたフィールド上では実行されません。検証が必要な場合は、送信ボタンを使ってください。

プロパティ - アンカー(**std_anchor**)

このウェブレットのプロパティは以下のとおりです。

| | | |
|------------------------------|------------------------------------|-------------------------------------|
| class | inline | style |
| corners | internal_id | swatch |
| displayAs | mini | tabindex |
| hideIf | onClickExtraFields | transition |
| icon | onClickWamName | transitionDirection |
| iconPosition | onClickWrName | urluseAjaxuseAjax |
| iconShadow | relationship | value |
| id | shadow | |

id

ウェブレットの一意的IDです。このプロパティは必須ではありませんが、送信されるとCSSもしくはJavaScriptで直接ウェブレットを参照することができます。値はページ内で一意でなければいけません。

注：jQuery Mobileは、Ajaxを使ってページをロードし、オプションとしてアニメーションを実行しながら、ページを現在のページに挿入します。これは、2つのウェブルーチンのコンテンツが1つのページ内に短時間存在する場合があることを意味します。両方のウェブルーチンが同じIDを持つウェブレットを含み、ユーザーがこの時間にウェブレットを参照するカスタムCSSもしくはJavaScriptを持つ場合、予期せぬ結果を得ることがあります。したがって、グローバル一意IDを作ってください。自動的に生成されてフィールドを表示するウェブレットは、ウェブルーチンとフィールドの名前を連結することにより、一意IDを作成します。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ID属性内の有効な文字と書式のルールはユーザーがサポートしたいHTMLのバージョンによって異なります。具体的詳細については、<http://www.w3.org/> で適切な仕様を確認してください。

全てのバージョンのHTML、CSS およびjQueryのようなJavaScript ライブラリとの互換性を保証するためには、次のルールを守ってください。

- A-Z もしくは a-z の文字で始めます。
- その後に続くことができるのは：文字(A-Za-z)、数字 (0-9)、ハイフン ("-")およびアンダースコア("_")。

value

アンカーをページ内で表示するテキストです。アンカー・ウェブレットはどんなカスタムHTMLコンテンツも含むことができます。ウェブレットが最初にデザインに配置されると、WAMエディターはこの値の文字列を表示するカスタム・コンテンツを自動的に作成します。カスタム・コンテンツを上書きすると、このプロパティは何の効果も持たなくなります。

省略値

空白。

有効値

プレーン・テキスト文字列。

class

ウェブレットに割り当てられるCSSクラスもしくはクラスです。CSSクラスを使うと、外部スタイルシート内で定義されウェブレットに適用する一連のCSSスタイルを指定したり、ウェブレット内でエレメントを指定したりすることができます。複数のHTMLエレメントで作られた複雑なウェブレットでは、クラスはウェブレットの最外部のエレメントに適用されます。

省略値

ブランク。

有効値

1つ以上のスペースで分けられたCSSクラス名を含む文字列。

corners

ボタンに丸角が必要な場合に指定します。

注：このプロパティは displayAs プロパティがボタンに設定された場合のみ有効で、それ以外は無視されます。

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

displayAs

このプロパティを"Button"に設定してリンクをボタンとしてスタイルします。jQuery Mobileフレームワークは、リンクをボタンとしてスタイルするためにマークアップとクラスとのリンクを強化します。リンクがボタンのように見えている間もリンクのままであることを注意してください。ボタンをクリックするとPOSTではなくGET要求が出され、フォーム内の全てのフィールドを自動的に送信しません。

省略値

Link - アンカーにスタイルを追加しません。

有効値

Link もしくはButton。

hideIf

評価がTrueの時にウェブレットが非表示になる式です。

省略値

False - ウェブレットは常に表示されます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

icon

このボタンと共に使う jQuery Mobile アイコンを指定します。

注: このプロパティは **displayAs** プロパティがボタンに設定された場合のみ有効で、それ以外は無視されます。

省略値

Default - アイコンは使われません。

有効値

プロパティのドロップダウンにリストされている値。

iconPosition

ボタン・テキストに対するボタン・アイコンの位置を指定します。

注: このプロパティは **displayAs** プロパティがボタンに設定され、**icon** が指定された場合のみ有効です。それ以外は無視されます。

省略値

ブランク - jQuery Mobile省略値の位置である'左'を使います。

有効値

- 左 - ボタン・テキストの左にアイコンを配置します。
- 右 - ボタン・テキストの右にアイコンを配置します。
- 上 - ボタン・テキストの上にアイコンを配置します。
- 下 - ボタン・テキストの下にアイコンを配置します。
- テキストなし - テキストなし (アイコンのみ) でボタンを描きます。

iconShadow

Trueに設定されるとボタンにテーマのシャドウを適用します。

注: このプロパティは **displayAs** プロパティがボタンに設定され、**icon** が指定された場合のみ有効です。それ以外は無視されます。

省略値

True - シャドウを適用します。

有効値

True、False、もしくはブール値を返す有効なXPath式。

inline

Trueに設定されると、ボタンはインライン・ボタンのように機能し、幅はボタンのコンテンツによって決定されます。Falseに設定されると、ボタンの幅はコンテンツに関係なくコンテナの全幅になります。

注: このプロパティは `displayAs` プロパティがボタンに設定された場合のみ有効で、それ以外は無視されます。

省略値

False - ボタンはコンテナの幅です。

有効値

True、False、もしくはブール値を返す有効なXPath式。

mini

Trueに設定されると、垂直高さがより低い、よりコンパクトなウェブレットを表示します。スペースが限られているツールバーなどの場所で便利です。

注: このプロパティは `displayAs` プロパティがボタンに設定された場合のみ有効で、それ以外は無視されます。

省略値

False - 標準サイズが使われます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

onClickExtraFields

省略値では、アンカーはフィールド値をターゲット・ウェブルーチンに送信せずにGET要求を生成します。このプロパティを使うと、ウェブレットがクリックされるとターゲット・ウェブルーチンに送信される追加のフィールドと値を指定することができます。

このプロパティは、プロパティ・シート内で省略記号(...)のボタンを使って呼び出されるカスタム・プロパティ・デザイナーでのみ設定できます。

| 名前 | 値 | |
|------------|---|----|
| DEPARTMENT | FLT | 削除 |
| | <input checked="" type="radio"/> リテラル <input type="radio"/> フィールド | |
| SECTION | SELSEC | 削除 |
| | <input type="radio"/> リテラル <input checked="" type="radio"/> フィールド | |

追加

OK

キャンセル

これは現在のウェブルーチン内の出力フィールド(#SELSEC)とターゲットのウェブルーチン内の入力フィールド([名前] 欄)にマップされているリテラル値("FLT") を表示します。

注：[名前] 欄のドロップダウンに正しい値が表示されるためには、このプロパティを編集する前に onClickWamName と onClickWrName プロパティが設定されている必要があります。

省略値

document(")/*/lxml:data/lxml:json[not(@id)](このウェブレットに定義された項目がないことを示します。)

有効値

適用不可。(この値はシステムにより保守されます。)デザイナーを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使用してください。

onClickWamName

このウェブレットがクリックされた時にウェブルーチンが実行されるWAMの名前を指定します。(ウェブルーチン名はonClickWrNameプロパティに指定します)

urlプロパティが指定された場合は、このプロパティは無視されます。

省略値

指定しない場合は、現在のWAMが使用されます。(\$lweb_WAMName)

有効値

WAMの名前。プロパティシート該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

onClickWrName

このウェブレットがクリックされたときに実行されるウェブルーチンの名前を指定します。(ウェブルーチンのあるWAM名はonClickWamNameプロパティで指定します。)

urlプロパティが指定された場合は、このプロパティは無視されます。

省略値

省略値は適用されません - urlプロパティもしくはon_click_wrnameプロパティを指定してください。

有効値

ウェブルーチンの名前。ウェブルーチンは、onClickWamNameプロパティで指定されたWAMに存在していなくてはなりません。プロパティシート該当するドロップダウン・ボタンをクリックすると、使用できるウェブルーチンのリストが表示されます。

relationship

jQuery Mobileにターゲット・ページをどのようにロードし表示するかを伝えます。可能な値とその効果は次の通りです。

- None Ajaxを使ってページ・コンテンツがロードされ、現在のページ・コンテンツを置換えます。ページはブラウザ履歴に追加されません。
- Back リンクもしくはボタンをクリックするとブラウザ履歴の前ページにナビゲートします(ブラウザで戻るボタンをクリックするのと同じです)。このオプションは、ウェブレット上のURLもしくはウェブルーチンの設定を無視させます。
- Dialog Ajaxを使ってページ・コンテンツがロードされ、ダイアログ・ウィンドウに表示されます。ブラウザ履歴に追加されません。
- External ページ全体がロードされ(Ajaxは使われなくて)、ブラウザ履歴に追加されます。これは、useAjaxをFalseに設定するのと同じ効果がありますが、動作の意味は異なります。この設定はドメインの別のサイトへのリンクに使われます。

注: セキュリティ上の理由から、この設定に関わらず別のサイトへのリンクにはAjaxをしません。

省略値

None。

有効値

None、Back、Dialog、External。

shadow

Trueに設定されるとボタンにドロップ・シャドウ・スタイルを適用します。

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

style

ウェブレットに適用するCSSスタイルの文字列を指定します。このプロパティを使うと、レイアウト・スタイルシートに定義されているどんな値も書き換えるウェブレットにCSSのstyleプロパティを設定することができます。

省略値

空白。

有効値

セミコロンで分けられた有効なCSSプロパティと値。

swatch

ウェブレットに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

tabindex

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

- 1 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
- 2 ゼロもしくは空白(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
- 3 負数のtab_indexを持つオブジェクトは、タブの順序から省かれます。古いブラウザではサポートされないこともあるので注意してください。

省略値

空白。ウェブレットはソース順に選択されます。

有効値

空白もしくは有効な整数値。

transition

次のページの表示に使う遷移を指定します。このプロパティは次のページがAjaxを使ってロードされる場合のみ適用されます。

標準jQuery Mobileと共に提供される遷移効果

- フェード
- ポップ
- フリップ
- ターン
- フロー
- スライドフェード
- スライド
- スライドアップ
- スライドダウン

カスタム遷移の作成の詳細については、[jQuery Mobileのドキュメント](#)を参照してください。

注: 全てのデバイスとブラウザが複雑な遷移をサポートしているわけではありません。サポートしていないデバイスでは、jQuery Mobileは自動的にフェードに戻るかもしくは遷移が発生しません。

省略値

slide

有効値

インストールされた遷移の名前もしくはNone。

transitionDirection

遷移アニメーションの方向を指定します。

注: リレーションシップ・プロパティが"back"に設定されると、遷移の方向はプロパティの値に関係なく"reverse"になります。

省略値

forward

有効値

forwardもしくはreverse。

url

ウェブレットがクリックされた時にロードされるURLを指定します。
onClickWamName と onClickWrName プロパティの両方が指定された場合、このプロパティは無視されます。

省略値

- 何もないという意味。

有効値

有効なURL文字列。

useAjax

jQueryは通常Ajax要求を使って同じドメインからページをロードし、ロードが終了するとページのコンテンツを所定の位置にアニメートします。これは通常、ロードを早くし(スクリプトとスタイルはリロードされません)、ユーザーのサイトにアプリケーションに近い外観を与えます。

強制的にページのリロードが必要になることがあります。必要になる可能性が最も高いのは、ターゲット・ウェブルーチンが別のレイアウトを使用し、別のスクリプトとスタイルのリソースのロードが必要な場合です。

このプロパティをFalseに設定すると、Ajax要求なしにページを強制的にリロードします。

ページのロード方法の詳細については、[jQuery Mobileのドキュメント](#)の"Linking Pages"を参照してください。

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

internal_id

WAMエディターとウェブレットがウェブルーチン・デザインに含まれるカスタム・コンテンツにウェブレットを接続するために使う一意のID。このプロパティはWAMエディターが自動的に構成します。手作業で修正しないでください。

9.4.2 ブール値 (std_boolean)

ブール値ウェブレットはブール値を表します。このウェブレットはHTMLソース内にエレメントを生成し、次にjQuery Mobileがこれをスライド・スイッチに変換します。

ブール値ウェブレットは、ブール値フィールドの省略値のウェブレットですが、trueValueとfalseValueプロパティを適切に構成することにより、有効な値が2つしかない任意のフィールドにも使うことができます。(例：性別)。

プロパティ - ブール値 (std_boolean)

このウェブレットのプロパティは以下のとおりです。

| | | |
|-------------------------------------|---------------------------|-------------------------------|
| autofocus | form | rdmlxDatatype |
| class | hideIf | style |
| disabled | hideLabel | swatch |
| displayMode | id | tabindex |
| falseDisplay | label | title |
| falseValue | mini | trueDisplay |
| fieldContainWrapper | name | trueValue |
| | | value |

id

ウェブレットの一意のID。このプロパティは必須ではありませんが、送信されるとCSSもしくはJavaScriptで直接ウェブレットを参照することができます。値はページ内で一意でなければいけません。

注：jQuery Mobileは、Ajaxを使ってページをロードし、オプションとしてアニメーションを実行しながら、ページを現在のページに挿入します。これは、2つのウェブルーチンのコンテンツが1つのページ内に短時間存在する場合があることを意味します。両方のウェブルーチンが同じIDを持つウェブレットを含み、ユーザーがこの時間にウェブレットを参照するカスタムCSSもしくはJavaScriptを持つ場合、予期せぬ結果を得ることがあります。したがって、グローバル一意IDを作ってください。自動的に生成されてフィールドを表示するウェブレットは、ウェブルーチンとフィールドの名前を連結することにより、一意IDを作成します。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ID属性内の有効な文字と書式のルールはユーザーがサポートしたいHTMLのバージョンによって異なります。具体的詳細については、<http://www.w3.org/>にて適切な仕様を確認してください。

全てのバージョンのHTML、CSS とjQueryのようなJavaScript ライブラリとの互換性を保証するためには、次のルールを守ってください。

- A-Z もしくは a-z の文字で始めます。
- その後に続くことができるのは：文字(A-Za-z)、数字 (0-9)、ハイフン ("-")およびアンダースコア("_")。

name

このウェブレットがウェブルーチンに送信された時にウェブレットの値を受け取るフィールドの名前。現在のウェブルーチンのWEB_MAPからフィールド上にウェブレットが生成される、もしくはドロップされると、このプロパティはそのフィールドの名前に設定されます。値を送信したいフィールドに別の名前がある場合、このプロパティをその名前に変更してください。

注: XHTMLテクノロジー・サービスにおいて、名前プロパティはしばしば一意のIDとして使われます。これはjQuery Mobileテクノロジー・サービスには当てはまりません。jQuery Mobileテクノロジー・サービスではidプロパティを使います。ウェブレットの値がサーバーに送信されない場合、このプロパティはブランクのままにされます。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ターゲット・ウェブルーチンのフィールド名。

value

ウェブレットに設定する値。ウェブレットがフィールドをビジュアルライズする場合、値が表示されるフィールドを識別します。ウェブレットの確認状態は、値をtrueValueプロパティと比較することにより判断されます。

省略値

省略値は適用されません - このウェブレットを使用するほとんどの場合において、チェックボックスにより表わされる値のフィールドを指定してください。trueValueプロパティに一致しない値はFalseとして扱われます。

有効値

文字列値、フィールド名、システム変数もしくは複数言語対応テキスト変数。

autofocus

ページがロードされるとウェブレットが自動的にフォーカスを得るように指定します。1ページに1つのフィールドのみ、このプロパティをTrueに設定します。2つ以上のフィールドでオートフォーカスをTrueに設定すると、異なるブラウザ上で矛盾した結果が生成されることがあります。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

class

ウェブレットに割り当てられるCSSクラスもしくはクラス。CSSクラスを使うと、外部スタイルシート内で定義され、ウェブレットに適用される一連のCSSスタイルを指定したり、ウェブレット内でエレメントを指定したりすることができます。複数のHTMLエレメントで作られた複雑なウェブレットでは、クラスはウェブレットの最外部のエレメントに適用されます。

省略値

空白。

有効値

1つ以上のスペースで分けられたCSSクラス名を含む文字列。

disabled

ウェブレットを無効化するかどうかを指定します。無効化されたウェブレットは、使用もクリックもできません。無効化されたウェブレットの値は、フォームと共に送信されないことに注意してください。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

displayMode

ウェブレットが入力内容を受け取るのか、出力内容のみかを制御します。

省略値

空白 ('input'の意味です)。

有効値

input または output。

falseDisplay

スイッチがoffの位置にあるときに表示されるテキスト。

省略値

No。

有効値

文字列値。

falseValue

Falseもしくははoff状態を示すフィールド値。フィールドが送信される時にサーバーに送信される値で、必ずしもウェブレットをoff状態に設定する値である必要はありません。かわりに、off状態は、フィールド値がtrueValue プロパティと等しくない場合に設定されます。

省略値

False。

有効値

文字列値。

fieldContainWrapper

jQuery Mobileは、ラベルとフィールドを小型画面に垂直にレイアウトする複雑性を処理するので、ラベルとフィールドは大型画面上でもすべて一列に並びます。そのためには、各ラベル/フィールド/ペアを特定の属性を持つ<div>タグで囲む必要があります。fieldContainWrapperをTrueに設定し、Labelプロパティを使ってラベル・テキストを設定すると、全ての処理を実行します。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

form

このウェブレットが属するフォームを指定する、スペースで分けられたフォームのIDのリスト。送信ボタンをクリックしてフォームが送信されると、そのフォームに属する全てのフィールドがサーバーに送信されます。省略値では、<フォーム>タグ内の全てのフィールドがフォームに属します。このプロパティを使うと、値はフォームと共に送信したまま、ドキュメントの別の部分、<フォーム>タグの外側、もしくは別の<フォーム>タグの内側にフィールドを配置できます。

LANSAの標準レイアウトにはページ全体をラップする<フォーム>タグが1つあるので、通常はこのプロパティをこれらのレイアウトに使う必要はありません。

省略値

ブランク。

有効値

スペースで分けられたフォームのIDのリスト。

hideIf

評価がTrueの時にウェブレットが非表示になる式。

省略値

False - ウェブレットは常に表示されます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

hideLabel

ラベルをアクセス可能なまま非表示にします。これは、ラベルは非表示でもスクリーン・リーダーのような支援技術に使用可能なことを意味します。

省略値

False - ラベルは非表示ではありません。

有効値

True、False、もしくはブール値を返す有効なXPath式。

label

ウェブレットに使うラベル・テキストを指定します。ウェブレットはこの値を使って<ラベル>タグを作成し、入力フィールドに正しく添付されるようにします。ラベルを表示する意図がない場合でも、アクセシビリティのために全てのウェブレットに意味のあるラベルを提供することをお勧めします。hideLabelプロパティを使って支援技術に使用可能なままラベルを非表示にします。

省略値

ブランク - 自動的に生成されるフィールドはリポジトリ定義の値を持ちます。

有効値

文字列値。

mini

Trueに設定されると、垂直高さがより低い、よりコンパクトなウェブレットを表示します。これは、スペースが限られているツールバーなどの場所で便利です。

省略値

False - 標準サイズが使われます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

rdmlxDatatype

ウェブレットに関連付けられたフィールドのRDMLXデータ・タイプを指定します。これは、ウェブレットがデータの検証をするのに役立ちます。通常このプロパティは、ユーザーがデザインにフィールドを生成もしくはドロップすると自動的に設定されます。ウェブレットをデザインにドロップしてからフィールドに関連付ける場合は、ユーザーがプロパティを設定する必要があります。

省略値

自動的にWAMエディターで設定されない限り空白です。

有効値

データ・タイプで始まり、データ・タイプの要求の通りに追加のパラメータが続く、|で区切られた文字列です。

integer|<最大長>

float|<最大長>

packed|<合計桁数>|<小数桁数>|<小数点記号>

signed|<合計桁数>|<小数桁数>|<小数点記号>

dec|<合計桁数>|<小数桁数>|<小数点記号>

alpha|<キーボード・シフト>|<最大長>

char|<キーボード・シフト>|<最大長>

varchar|<キーボード・シフト>|<最大長>

nchar|<キーボード・シフト>|<最大長>

nvarchar|<キーボード・シフト>|<最大長>

style

ウェブレットに適用するCSSスタイルの文字列を指定します。このプロパティを使うと、レイアウト・スタイルシートに定義されているどんな値も書き換えるウェブレットにCSSのstyleプロパティを設定することができます。

省略値

空白。

有効値

セミコロンで分けられた有効なCSSプロパティと値。

swatch

ウェブレットに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

tabindex

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

- 1 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
- 2 ゼロもしくは空白(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
- 3 負数のtab_indexを持つオブジェクトは、タブの順序から省かれます。古いブラウザではサポートされないこともあるので注意してください。

省略値

空白。ウェブレットはソース順に選択されます。

有効値

空白もしくは有効な整数値。

title

ウェブレットに関する追加の助言的な情報を指定します。これは通常、必須ではない追加の情報で、ユーザーがウェブレットの目的を理解するのを助けます。ブラウザが違くと扱い方が違うことがあります。例えば、マウスがウェブレットの上に乗ると、ほとんどのデスクトップ・ブラウザがタイトルをヒントとして表示します。スクリーン・リーダーのような支援技術は、ユーザーのためにタイトルを読みます。これを書いている時点では、モバイル機器のブラウザはタイトルを無視します。

省略値

空白。

有効値

有効なHTMLの属性文字列。

trueDisplay

スイッチがonの位置にあるときに表示されるテキスト。

省略値

Yes。

有効値

文字列値。

trueValue

Falseもしくはon状態を示すフィールド値。フィールドが送信される時にサーバーに送信される値です。valueフィールドがこの値を含む場合、ウェブレットは最初はon状態に設定されます。

省略値

True。

有効値

文字列値。

9.4.3 プッシュ ボタン (std_button_v2)

プッシュ・ボタン・ウェブレットは、クリックできるボタンを作成するHTML <ボタン> エlementを示します。<入力> Element (入力ボックス・ウェブレット) を使って作成されるボタンとは異なり、プッシュ・ボタン・ウェブレットは、フォーマットされたテキストやイメージのようなカスタムHTMLを含むことができます。

可能なボタンには次の3つがあります。

送信 送信ボタンは、親フォーム(もしくはformプロパティで指定されたフォーム)の全ての指定されたフィールドの値を送信するウェブルーチン呼び出します。ウェブルーチンにデータを送信してデータベースを更新する時、もしくはフォームを送信する前にフィールド上でHTML 5の検証の実行が必要な時は送信ボタンを使ってください。

リセット リセットボタンは親<フォーム>(またはformプロパティで指定されたフォーム)の全てのフィールドを、ページがロードされた時の値にリセットします。

ボタン 'ボタン'には、クリックされた場合の省略値のアクションはありません。このようなボタンには、カスタムJavaScript関数を添付できます。

```
$("#buttonID").click(function(eventObject) {  
    alert("I was clicked!");  
});
```

プロパティ - プッシュ ボタン (std_button_v2)

このウェブレットのプロパティは以下の通りです。

| | | |
|----------------|--------------------|---------------------|
| autofocus | hideIf | onClickWrName |
| caption | icon | relationship |
| class | iconPosition | shadow |
| corners | iconShadow | style |
| disabled | id | swatch |
| form | inline | tabindex |
| formaction | internal_id | title |
| formenctype | mini | transition |
| formmethod | name | transitionDirection |
| formnovalidate | onClickExtraFields | type |
| formtarget | onClickWamName | useAjax |
| | | value |

id

ウェブレットの一意のID。このプロパティは必須ではありませんが、送信されるとCSSもしくはJavaScriptで直接ウェブレットを参照することができます。値はページ内で一意でなければいけません。

注：jQuery Mobileは、Ajaxを使ってページをロードし、オプションとしてアニメーションを実行しながら、ページを現在のページに挿入します。これは、2つのウェブルーチンのコンテンツが1つのページ内に短時間存在する場合があることを意味します。両方のウェブルーチンが同じIDを持つウェブレットを含み、ユーザーがこの時間にウェブレットを参照するカスタムCSSもしくはJavaScriptを持つ場合、予期せぬ結果を得ることがあります。したがって、グローバル一意IDを作ってください。自動的に生成されてフィールドを表示するウェブレットは、ウェブルーチンとフィールドの名前を連結することにより、一意IDを作成します。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ID属性内の有効な文字と書式のルールはユーザーがサポートしたいHTMLのバージョンによって異なります。具体的詳細については、<http://www.w3.org/> で適切な仕様を確認してください。

全てのバージョンのHTML、CSS とjQueryのようなJavaScript ライブラリとの互換性を保証するためには、次のルールを守ってください。

- A-Z もしくは a-z の文字で始めます。
- その後に続くことができるのは：文字(A-Za-z)、数字 (0-9)、ハイフン ("-")およびアンダースコア("_")。

name

このウェブレットがウェブルーチンに送信された時にウェブレットの値を受け取るフィールドの名前。現在のウェブルーチンのWEB_MAPからフィールド上にウェブレットが生成される、もしくはドロップされると、このプロパティはそのフィールドの名前に設定されます。値を送信したいフィールドに別の名前がある場合、このプロパティをその名前に変更してください。

注: XHTMLテクノロジー・サービスにおいて、名前プロパティはしばしば一意のIDとして使われます。これはjQuery Mobileテクノロジー・サービスには当てはまりません。jQuery Mobileテクノロジー・サービスではidプロパティを使います。ウェブレットの値がサーバーに送信されない場合、このプロパティはブランクのままにされます。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ターゲット・ウェブルーチンのフィールド名。

value

ボタンの値。この値は表示されません (captionプロパティを使います)。フィールドが送信される時にターゲット・ウェブルーチンに送信される値です。

省略値

ブランク。

有効値

文字列値。

autofocus

ページがロードされるとウェブレットが自動的にフォーカスを得るように指定します。1ページに1つのフィールドのみ、このプロパティをTrueに設定します。2つ以上のフィールドでオートフォーカスをTrueに設定すると、異なるブラウザ上で矛盾した結果が生成されることがあります。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

caption

ボタンのキャプション。これは、ボタンの省略値のコンテンツです。ボタンに追加されるカスタム・コンテンツがこの値を上書きすることがあります。

省略値

空白。

有効値

文字列値。

class

ウェブレットに割り当てられるCSSクラスもしくはクラス。CSSクラスを使うと、外部スタイルシート内で定義され、ウェブレットに適用される一連のCSSスタイルを指定したり、ウェブレット内でエレメントを指定したりすることができます。複数のHTMLエレメントで作られた複雑なウェブレットでは、クラスはウェブレットの最外部のエレメントに適用されます。

省略値

空白。

有効値

1つ以上のスペースで分けられたCSSクラス名を含む文字列。

corners

ボタンに丸角が必要な場合に指定します。

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

disabled

ウェブレットを無効化するかどうかを指定します。無効化されたウェブレットは、使用もクリックもできません。無効化されたウェブレットの値は、フォームと共に送信されないことに注意してください。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

form

このウェブレットが属するフォームを指定する、スペースで分けられたフォームのIDのリスト。送信ボタンをクリックしてフォームが送信されると、そのフォームに属する全てのフィールドがサーバーに送信されます。省略値では、<フォーム>タグ内の全てのフィールドがフォームに属します。このプロパティを使うと、値はフォームと共に送信したまま、ドキュメントの別の部分、<フォーム>タグの外側、もしくは別の<フォーム>タグの内側にフィールドを配置できます。

LANSAsの標準レイアウトにはページ全体をラップする<フォーム>タグが1つあるので、通常はこのプロパティをこれらのレイアウトに使う必要はありません。

省略値

ブランク。

有効値

スペースで分けられたフォームのIDのリスト。

formation

フォームを送信する時にフォーム・データの送信先を指定するURLです。type="submit"のみに有効です。通常、LANSAの実行時フレームワークは、onClickWrName と onClickWRフィールドに基づいて正しいURLを理解します。このプロパティを指定すると省略値のふるまいを上書きします。

省略値

空白。

有効値

有効なURL文字列。

formenctype

フォームを送信する時に使うエンコード・タイプを指定します。
type="submit"のみに有効です。省略値のタイプでほとんどのフォームをLANSAサーバーに送信することができるため、通常formenctypeを設定する必要はありません。LANSA以外のサーバーにフォームを送信する場合は、そのサーバーの特定の要求により別のエンコード・タイプが必要になることがあります。

省略値

ブランク - 特別に指定されない限り、親<フォーム>エレメントにより指定された省略値のapplication/x-www-form-urlencodedを使います。

有効値

| | |
|-----------------------------------|---|
| application/x-www-form-urlencoded | 送信する前に全ての文字がエンコードされます (スペースは"+"記号に変換され、特殊文字はASCII HEX値に変換されます)。 |
| multipart/form-data | 文字はエンコードされません。この値はファイル・アップロード・コントロールがある時に必要です。 |
| text/plain | スペースは"+"記号に変換されますが、特殊文字はエンコードされません。 |

formmethod

フォーム・データの送信に使うHTTPメソッドを指定します。
type="submit"のみに有効です。

省略値

空白 - 特別に指定されない限り、省略値の親<フォーム>エレメントにより指定された省略値のPOSTを使います。

有効値

'GET' または 'POST'。

formnovalidate

送信する前にフォームを検証してはいけないことを指定します。
type="submit"のみに有効です。このプロパティをTrueに設定すると親<フォーム>エレメント内でオンになっていた検証を抑制し、Falseに設定すると親<フォーム>エレメント内でオフになっていた検証をオンにしません。

提供されているレイアウトでは、<フォーム>の検証はレイアウト・ウェブレットのvalidationMethodプロパティを使って構成されます。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

formtarget

フォーム送信後の応答を表示するフレームもしくはウィンドウを指定します。type="submit"のみに有効です。

このプロパティはHTML 5の仕様の一部で、完全を期すために提供されます。このプロパティの使用には注意が必要です。ブラウザが違くとフレームとウィンドウのサポート・レベルが異なり、このプロパティの扱いも異なることがあります。jQuery Mobileの Ajaxメカニズムとの競合を起こすこともあります。

省略値

ブランク - 特別に指定されない限り、省略値のブランクでもある、親<フォーム>エレメントにより指定された値を使います(現在のページに表示)。

有効値

_blank, _self, _parent, _top, framename。

hideIf

評価がTrueの時にウェブレットが非表示になる式。

省略値

False - ウェブレットは常に表示されます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

icon

このボタンと共に使う jQuery Mobile アイコンを指定します。

省略値

Default - アイコンは使われません。

有効値

プロパティのドロップダウンにリストされている値。

iconPosition

ボタン・テキストに対するボタン・アイコンの位置を指定します。

省略値

ブランク - jQuery Mobile省略値の位置である'左'を使います。

有効値

- 左 - ボタン・テキストの左にアイコンを配置します。
- 右 - ボタン・テキストの右にアイコンを配置します。
- 上 - ボタン・テキストの上にアイコンを配置します。
- 下 - ボタン・テキストの下にアイコンを配置します。
- テキストなし - テキストなし (アイコンのみ) でボタンを描きます。

iconShadow

Trueに設定されるとボタンにテーマのシャドウを適用します。

省略値

True - シャドウを適用します。

有効値

True、False、もしくはブール値を返す有効なXPath式。

inline

Trueに設定されると、ボタンはインライン・ボタンのように機能し、幅はボタンのコンテンツによって決定されます。Falseに設定されると、ボタンの幅はコンテンツに関係なくコンテナの全幅になります。

省略値

False - ボタンはコンテナの幅です。

有効値

True、False、もしくはブール値を返す有効なXPath式。

mini

Trueに設定されると、垂直高さがより低い、よりコンパクトなウェブレットを表示します。スペースが限られているツールバーなどの場所で便利です。

省略値

ブランク - jQuery Mobile により決定されます。通常は、デフォルトは標準サイズですが、ツールバー内のボタンなど、状況によっては省略値がミニサイズに設定されます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

onClickExtraFields

省略値では、ボタンはPOST要求を生成し、親<フォーム>内の全てのフィールドを自動的にターゲット・ウェブルーチンに送信します。このプロパティを使うと、ウェブレットがクリックされるとターゲット・ウェブルーチンに送信される追加のフィールドと値を指定することができます。

このプロパティは、プロパティ・シート内で省略記号(...)のボタンを使って呼び出されるカスタム・プロパティ・デザイナーでのみ設定できます。

これは現在のウェブルーチン内の出力フィールド(#SELSEC)とターゲットのウェブルーチン内の入力フィールド([名前] 欄)にマップされているリテラル値("FLT") を表示します。

注：[名前] 欄のドロップダウンに正しい値が表示されるためには、このプロパティを編集する前に onClickWamName と onClickWrName プロパティが設定されている必要があります。

省略値

document(")/*/lxml:data/lxml:json[not(@id)](このウェブレットに定義された項目がないことを示します。)

有効値

適用不可。(この値はシステムにより保守されます。)デザイナーを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使用してください。

onClickWamName

このウェブレットがクリックされた時にウェブルーチンが実行されるWAMの名前を指定します。(ウェブルーチン名はonClickWrNameプロパティに指定します。)

このプロパティはtype プロパティが"button"もしくは"reset"に設定されると効果がありません。

省略値

指定しない場合は、現在のWAMが使用されます。(\$lweb_WAMName)

有効値

WAMの名前。プロパティシートの該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

onClickWrName

このウェブレットがクリックされたときに実行されるウェブルーチンの名前を指定します。(ウェブルーチンのあるWAM名はonClickWamNameプロパティで指定します。)

このプロパティはtypeプロパティが"button"もしくは"reset"に設定されると効果がありません。

省略値

省略値は適用されません - urlプロパティもしくはon_click_wrnameプロパティを指定してください。

有効値

ウェブルーチンの名前。ウェブルーチンは、onClickWamNameプロパティで指定されたWAMに存在していなくてはなりません。プロパティシートの該当するドロップダウン・ボタンをクリックすると、使用できるウェブルーチンのリストが表示されます。

relationship

jQuery mobileにターゲット・ページをどのようにロードし表示するかを伝えます。可能な値とその効果は次の通りです。

- None Ajaxを使ってページ・コンテンツがロードされ、現在のページ・コンテンツを置換えます。ページはブラウザ履歴に追加されません。
- Back リンクもしくはボタンをクリックするとブラウザ履歴の前ページにナビゲートします(ブラウザで戻るボタンをクリックするのと同じです)。このオプションは、ウェブレット上のURLもしくはウェブルーチンの設定を無視させます。
- Dialog Ajaxを使ってページ・コンテンツがロードされ、ダイアログ・ウィンドウに表示されます。ブラウザ履歴に追加されません。
- External ページ全体がロードされ (Ajaxは使われないで)、ブラウザ履歴に追加されます。これは、useAjax をFalseに設定するのと同じ効果がありますが、動作の意味は異なります。この設定はドメインの別のサイトへのリンクに使われます。

注: セキュリティのため、別のサイトへのリンクにはこの設定に関わらずAjaxを使いません。

省略値

None。

有効値

None、Back、Dialog、External。

shadow

Trueに設定されるとボタンにドロップ・シャドウ・スタイルを適用します。

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

style

ウェブレットに適用するCSSスタイルの文字列を指定します。このプロパティを使うと、レイアウト・スタイルシートに定義されているどんな値も書き換えるウェブレットにCSSのstyleプロパティを設定することができます。

省略値

空白。

有効値

セミコロンで分けられた有効なCSSプロパティと値。

swatch

ウェブレットに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテンツからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

tabindex

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

- 1 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
- 2 ゼロもしくは空白(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
- 3 負数のtab_indexを持つオブジェクトは、タブの順序から省かれます。古いブラウザではサポートされないこともあるので注意してください。

省略値

空白。ウェブレットはソース順に選択されます。

有効値

空白もしくは有効な整数値。

title

ウェブレットに関する追加の助言的な情報を指定します。これは通常、必須ではない追加の情報で、ユーザーがウェブレットの目的を理解するのを助けます。ブラウザが違くと扱い方が違うことがあります。例えば、マウスがウェブレットの上に乗ると、ほとんどのデスクトップ・ブラウザがタイトルをヒントとして表示します。スクリーン・リーダーのような支援技術は、ユーザーのためにタイトルを読みます。これを書いている時点では、モバイル機器のブラウザはタイトルを無視します。

省略値

空白。

有効値

有効なHTMLの属性文字列。

transition

次のページの表示に使う遷移を指定します。このプロパティは次のページがAjaxを使ってロードされる場合のみ適用されます。

標準jQuery Mobileと共に提供される遷移効果

- フェード
- ポップ
- フリップ
- ターン
- フロー
- スライドフェード
- スライド
- スライドアップ
- スライドダウン

カスタム遷移の作成の詳細については、[jQuery Mobileのドキュメント](#)を参照してください。

注: 全てのデバイスとブラウザが複雑な遷移をサポートしているわけではありません。サポートしていないデバイスでは、jQuery Mobileは自動的にフェードに戻るかもしくは遷移が発生しません。

省略値

slide。

有効値

インストールされた遷移の名前もしくはNone。

transitionDirection

遷移アニメーションの方向を指定します。

注: リレーションシップ・プロパティが"back"に設定されると、遷移の方向はプロパティの値に関係なく"reverse"になります。

省略値

forward。

有効値

forwardもしくはreverse。

type

ボタンのタイプを指定します。可能なボタンのタイプは3つあります：

- Submit 送信ボタンをタップするとボタンが属する<フォーム>がサーバーに送信されます。
- Reset リセット・ボタンをタップするとブラウザはフォーム内の全てのフィールドをページがロードされた時の初期値にリセットします。
- Button このボタンには、クリックされた場合の省略値のアクションはありません。カスタマイズされたふるまいを定義するにはJavaScriptを使ってください。

省略値

submit。

有効値

submit, reset, button。

useAjax

jQueryは通常Ajax要求を使って同じドメインからページをロードし、ロードが終了するとページのコンテンツを所定の位置にアニメートします。これは通常、ロードを早くし(スクリプトとスタイルはリロードされません)、ユーザーのサイトにアプリケーションに近い外観を与えます。

強制的にページのリロードが必要になることがあります。必要になる可能性が最も高いのは、ターゲット・ウェブルーチンが別のレイアウトを使用し、別のスクリプトとスタイルのリソースのロードが必要な場合です。

このプロパティをFalseに設定すると、Ajax要求なしにページを強制的にリロードさせます。

ページのロードの方法の詳細については、[jQuery Mobileのドキュメント](#)の"Linking Pages"を参照してください。

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

internal_id

WAMエディターとウェブレットがウェブルーチン・デザインに含まれるカスタム・コンテンツにウェブレットを接続するために使う一意のID。このプロパティはWAMエディターが自動的に構成します。手作業で修正しないでください。

9.4.4 チェックボックス (std_checkbox)

このウェブレットは、jQuery Mobileが現在のテーマに合うように修正する標準チェックボックス・コントロールを作成します。チェックボックスがチェックされると、チェックボックス・フィールドはウェブルーチンにのみ送信されることに注意してください。チェックボックスがチェックされないと値は送信されず、ウェブルーチン内のフィールドは省略値を含みません。

プロパティ - チェックボックス (std_checkbox)

このウェブレットのプロパティは以下のとおりです。

| | | |
|-----------|---------------|---------------|
| autofocus | label | selectedValue |
| disabled | mini | swatch |
| form | name | tabindex |
| hideIf | rdmlxDatatype | title |
| id | required | value |

id

ウェブレットの一意的IDです。このプロパティは必須ではありませんが、送信されるとCSSもしくはJavaScriptで直接ウェブレットを参照することができます。値はページ内で一意でなければいけません。

注：jQuery Mobileは、Ajaxを使ってページをロードし、オプションとしてアニメーションを実行しながら、ページを現在のページに挿入します。これは、2つのウェブルーチンのコンテンツが1つのページ内に短時間存在する場合がありますことを意味します。両方のウェブルーチンが同じIDを持つウェブレットを含み、ユーザーがこの時間にウェブレットを参照するカスタムCSSもしくはJavaScriptを持つ場合、予期せぬ結果を得ることがあります。したがって、グローバル一意IDを作ってください。自動的に生成されてフィールドを表示するウェブレットは、ウェブルーチンとフィールドの名前を連結することにより、一意IDを作成します。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ID属性内での有効な文字と書式のルールはユーザーがサポートしたいHTMLのバージョンによって異なります。具体的詳細については、適切な仕様は<http://www.w3.org/> を参照してください。

全てのバージョンのHTML、CSSおよびjQueryのようなJavaScriptライブラリとの互換性を保証するためには、次のルールを守ってください。

- A-Z もしくは a-z の文字で始めます。
- その後に続くことができるのは：文字(A-Za-z)、数字 (0-9)、ハイフン ("-")およびアンダースコア("_")。

name

このウェブレットがウェブルーチンに送信された時にウェブレットの値を受け取るフィールドの名前です。現在のウェブルーチンのWEB_MAPからフィールド上にウェブレットが生成される、もしくはドロップされると、このプロパティはそのフィールドの名前に設定されます。値を送信したいフィールドに別の名前がある場合、このプロパティをその名前に変更します。

注: XHTMLテクノロジー・サービスにおいて、名前のプロパティはしばしば一意のIDとして使われます。これはjQuery Mobileテクノロジー・サービスには当てはまりません。jQuery Mobileテクノロジー・サービスにはidプロパティを使います。ウェブレットの値がサーバーに送信されない場合、このプロパティはブランクのままにされます。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ターゲット・ウェブルーチンのフィールド名。

value

チェックボックスの初期値です。この値はチェックボックスを最初に確認すべきか判断するためにselectedValueプロパティと比較されます。

省略値

空白。

有効値

文字列値。

autofocus

ページがロードされるとウェブレットが自動的にフォーカスを得るように指定します。このプロパティがTrueに設定されたフィールドは1ページに1つだけあります。2つ以上のフィールドでオートフォーカスをTrueに設定すると、異なるブラウザ上で矛盾した結果が生成されることがあります。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

disabled

ウェブレットを無効化するかどうかを指定します。無効化されたウェブレットは、使用もクリックもできません。無効化されたウェブレットの値は、フォームと共に送信されないことに注意してください。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

form

このウェブレットが属するフォームを指定する、スペースで分けられたフォームのIDのリストです。送信ボタンをクリックしてフォームが送信されると、そのフォームに属する全てのフィールドがサーバーに送信されます。省略値では、<フォーム>タグ内の全てのフィールドがフォームに属します。このプロパティによって、値はフォームと共に送信したまま、ドキュメントの別の部分、<フォーム>タグの外側、もしくは別の<フォーム>タグの内側にフィールドを配置できます。

LANSAsの標準レイアウトにはページ全体をラップする<フォーム>タグが1つあるので、通常はこのプロパティをレイアウトに使う必要はありません。

省略値

ブランク。

有効値

スペースで分けられたフォームのIDのリスト。

hideIf

Trueと評価された時、ウェブレットを非表示にする式です。

省略値

False - ウェブレットは常に表示されます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

label

ウェブレットに使うラベル・テキストを指定します。ウェブレットはこの値を使って<ラベル>タグを作成し、入力フィールドに正しく添付されるようにします。

省略値

ブランク - 自動的に生成されるフィールドはリポジトリ定義から値をとります。

有効値

文字列値。

mini

Trueに設定されると、垂直高さがより低い、よりコンパクトなウェブレットを表示します。スペースが限られているツールバーなどの場所で便利です。

省略値

False - 標準サイズが使われます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

rdmlxDatatype

ウェブレットに関連付けられたフィールドのRDMLXデータ・タイプを指定します。これは、あるウェブレットがデータの検証をするのに役立ちます。通常このプロパティは、ユーザーがデザインにフィールドを生成もしくはドロップすると自動的に設定されます。ウェブレットをデザインにドロップしてからフィールドに関連付ける場合は、ユーザーがプロパティを設定する必要があります。

省略値

自動的にWAMエディターで設定されない限りBlank。

有効値

データ・タイプで始まりデータ・タイプの要求の通りに追加のパラメータが続く、|で区切られた文字列：

integer|<最大長>

float|<最大長>

packed|<合計桁数>|<小数桁数>|<小数点記号>

signed|<合計桁数>|<小数桁数>|<小数点記号>

dec|<合計桁数>|<小数桁数>|<小数点記号>

alpha|<キーボード・シフト>|<最大長>

char|<キーボード・シフト>|<最大長>

varchar|<キーボード・シフト>|<最大長>

nchar|<キーボード・シフト>|<最大長>

nvarchar|<キーボード・シフト>|<最大長>

required

フォームの送信の前にチェックボックスをチェックするように指定します。このガイドが書かれている時点では、SafariとInternet Explorerはこのプロパティをサポートしていません。

注: HTML 5フォームの検証は、無効化属性を<フォーム>タグもしくは送信ボタンに追加することによってオフにすることができます (これは、提供されている標準レイアウト上で validationMethodを 'none' に設定することにより、自動的に実行されます)。さらに、ブラウザによってHTML 5の検証サポート・レベルが異なり、悪意のあるユーザーがクライアント側の検証を解除する方法が多数あります。したがって、クライアント側の検証はユーザー・エクスペリエンスを向上させますが、信頼してはいけません。常にサーバー側の検証でバック・アップしてください。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

selectedValue

選択された状態を示すフィールド値を指定します。チェックボックスが初期化されると、実際の値は、チェックボックスがチェックされているか判断するために(大文字/小文字を区別する比較を使って)この値と比較されます。フォームが送信される時、チェックボックスがチェックされている場合はこの値が送信されます。チェックボックスがチェックされていないと値は送信されません。

省略値

True。

有効値

文字列値。

swatch

ウェブレットに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

tabindex

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

- 1 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
- 2 ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
- 3 負数のtab_indexを持つオブジェクトは、タブの順序から省かれます。古いブラウザではサポートされないこともあるので注意してください。

省略値

Blank。ウェブレットはソース順に選択されます。

有効値

Blankもしくは有効な整数値。

title

ウェブレットに関する追加の助言的な情報を指定します。これは通常、必須ではない追加の情報で、ユーザーがウェブレットの目的を理解するのを助けます。ブラウザによって扱い方が違うことがあります。例えば、マウスがウェブレットの上に乗ると、ほとんどのデスクトップ・ブラウザがタイトルをヒントとして表示します。スクリーン・リーダーのような支援技術は、ユーザーのためにタイトルを読みます。このガイドが書かれている時点では、モバイル機器のブラウザはタイトルを無視します。

省略値

空白。

有効値

有効なHTMLの属性文字列。

9.4.5 折りたたみ可能ブロック(std_collapsible)

折りたたみ可能ブロックは、ヘッダーをクリックすることにより、ユーザーが表示もしくは非表示にできるコンテンツのセクションです。



折りたたまれた折りたたみ可能ブロック



展開された折りたたみ可能ブロック

プロパティ - 折りたたみ可能ブロック (std_collapsible)

このウェブレットのプロパティは以下のとおりです。

| | | |
|-------------------------------|------------------------------|-----------------------------|
| collapsed | headerSwatch | id |
| contentSwatch | headerText | inset |
| headerLevel | hideIf | internal_id |

id

ウェブレットの一意的IDです。このプロパティは必須ではありませんが、送信されるとCSSもしくはJavaScriptで直接ウェブレットを参照することができます。値はページ内で一意でなければいけません。

注：jQuery Mobileは、Ajaxを使ってページをロードし、オプションとしてアニメーションを実行しながら、ページを現在のページに挿入します。これは、2つのウェブルーチンのコンテンツが1つのページ内に短時間存在する場合があることを意味します。両方のウェブルーチンが同じIDを持つウェブレットを含み、ユーザーがこの時間にウェブレットを参照するカスタムCSSもしくはJavaScriptを持つ場合、予期せぬ結果を得ることがあります。したがって、グローバル一意IDを作ってください。自動的に生成されてフィールドを表示するウェブレットは、ウェブルーチンとフィールドの名前を連結することにより、一意IDを作成します。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ID属性内での有効な文字と書式のルールはユーザーがサポートしたHTMLのバージョンによって異なります。具体的詳細については、<http://www.w3.org/> で適切な仕様を確認してください。

全てのバージョンのHTML、CSSおよびjQueryのようなJavaScriptライブラリとの互換性を保証するためには、次のルールを守ってください。

- A-Z もしくは a-z の文字で始めます。
- その後に続くことができるのは：文字(A-Za-z)、数字(0-9)、ハイフン("-")およびアンダースコア("_")。

collapsed

ウェブレットの初期状態を指定します。

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

contentSwatch

ウェブレットのコンテンツ・エリアに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

headerLevel

ウェブレットのヘッダー・セクションはHTMLのヘッダー・タグに表示されます。このプロパティはそのヘッダーのレベル (h1, h2, h3, h4, h5, h6, h7 または h8) を指定します。ヘッダーのレベルはウェブレットの外観に影響を与えませんがドキュメントのセマンティック構造には重要な場合があります。検索エンジンなどの他のシステムもしくは支援技術によるユーザーのページの処理方法に重要な場合があります。

省略値

3。

有効値

1、 2、 3、 4、 5、 6、 7もしくは8。

headerSwatch

ウェブレットのヘッダー・エリアに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

headerText

ヘッダー・エリアに表示するテキストを指定します。

省略値

折りたたみ可能ブロック。

有効値

文字列値。

hideIf

Trueと評価された時、ウェブレットを非表示にする式です。

省略値

False - ウェブレットは常に表示されます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

inset

省略値では、折りたたみ可能ブロックはインセットモードの表示になっています。角のスタイルがない、全画面いっぱいの表示にするには、このプロパティをFalseに設定します。

省略値

True - ブロックにはマージンがあります。

有効値

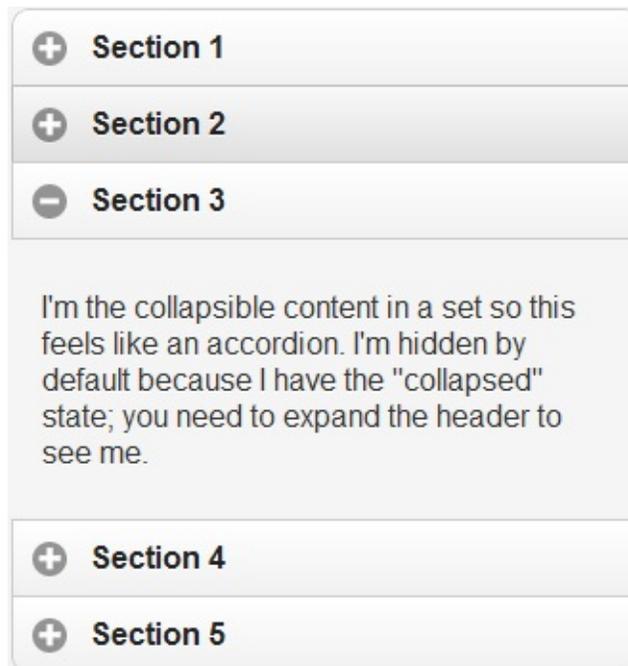
True、False、もしくはブール値を返す有効なXPath式。

internal_id

ウェブラーチン・デザインに含まれるカスタム・コンテンツにウェブレットを接続するためにWAMエディターとウェブレットに使われる一意のIDです。このプロパティはWAMエディターが自動的に構成します。手作業で修正しないでください。

9.4.6 折りたたみ可能セット(std_collapsible)

折りたたみ可能セット・ウェブレットは、複数の折りたたみ可能ブロック・ウェブレットを囲んで配置され、折りたたみを作成するラッパーです。1つの折りたたみ可能ブロックが展開されると、セット内の他のブロックが折りたたまれます。このウェブレットは折りたたみ可能ブロック・ウェブレットを作成しません。折りたたみ可能ブロック・ウェブレットを作成し、折りたたみ可能セット・ウェブレットのコンテンツ・エリアに入れてください。



プロパティ - 折りたたみ可能セット(`std_collapsibleset`)

このウェブレットのプロパティは以下のとおりです。

[contentSwatch](#)

[headerSwatch](#)

[hideIf](#)

[id](#)

[internal_id](#)

id

ウェブレットの一意的IDです。このプロパティは必須ではありませんが、送信されるとCSSもしくはJavaScriptで直接ウェブレットを参照することができます。値はページ内で一意でなければいけません。

注：jQuery Mobileは、Ajaxを使ってページをロードし、オプションとしてアニメーションを実行しながら、ページを現在のページに挿入します。これは、2つのウェブルーチンのコンテンツが1つのページ内に短時間存在する場合がありますことを意味します。両方のウェブルーチンが同じIDを持つウェブレットを含み、ユーザーがこの時間にウェブレットを参照するカスタムCSSもしくはJavaScriptを持つ場合、予期せぬ結果を得ることがあります。したがって、グローバル一意IDを作ってください。自動的に生成されてフィールドを表示するウェブレットは、ウェブルーチンとフィールドの名前を連結することにより、一意IDを作成します。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ID属性内での有効な文字と書式のルールはユーザーがサポートしたHTMLのバージョンによって異なります。具体的詳細については、<http://www.w3.org/> で適切な仕様を確認してください。

全てのバージョンのHTML、CSSおよびjQueryのようなJavaScriptライブラリとの互換性を保証するためには、次のルールを守ってください。

- A-Z もしくは a-z の文字で始めます。
- その後に続くことができるのは：文字(A-Za-z)、数字(0-9)、ハイフン("-")およびアンダースコア("_")。

contentSwatch

子折りたたみ可能ウェブレットのコンテンツ・エリアに適用する省略値のjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

headerSwatch

ウェブレットのヘッダー・エリアに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

hideIf

Trueと評価された時、ウェブレットを非表示にする式です。

省略値

False - ウェブレットは常に表示されます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

internal_id

ウェブラーチン・デザインに含まれるカスタム・コンテンツにウェブレットを接続するためにWAMエディターとウェブレットに使われる一意のIDです。このプロパティはWAMエディターが自動的に構成します。手作業で修正しないでください。

9.4.7 コントロール・グループ (std_controlgroup)

コントロール・グループ・ウェブレットは複数のボタン、チェックボックスもしくはラジオ・ボタンの周囲に配置され、グループ化するラッパーです。フレームワークは自動的にボタンの間のマージンを削除し、セットの上部と下部の角のみを丸くします。

Choose as many snacks as you'd like:

- Cheetos
- Doritos
- Fritos
- Sun Chips

Font styling:

b *i* u

Choose a pet:

- Cat
- Dog
- Hamster
- Lizard

Layout view:

List Grid Gallery

コントロールの例

グループ化されたコン

プロパティ - コントロール・グループ (std_controlgroup)

このウェブレットのプロパティは以下のとおりです。

| | | |
|---------------------|-------------|-------------|
| class | id | orientation |
| fieldContainWrapper | internal_id | style |
| hideIf | label | swatch |
| hideLabel | mini | |

id

ウェブレットの一意のIDです。このプロパティは必須ではありませんが、送信されるとCSSもしくはJavaScriptで直接ウェブレットを参照することができます。値はページ内で一意でなければいけません。

注：jQuery Mobileは、Ajaxを使ってページをロードし、オプションとしてアニメーションを実行しながら、ページを現在のページに挿入します。これは、2つのウェブルーチンのコンテンツが1つのページ内に短時間存在する可能性があることを意味します。両方のウェブルーチンが同じIDを持つウェブレットを含み、ユーザーがこの時間にウェブレットを参照するカスタムCSSもしくはJavaScriptを持つ場合、予期せぬ結果を得ることがあります。したがって、グローバル一意IDを作ってください。自動的に生成されてフィールドを表示するウェブレットは、ウェブルーチンとフィールドの名前を連結することにより、一意IDを作成します。

省略値

ブランク。フィールドを表示するために自動的に生成されるウェブレットは、自動的に設定される値を持ちます。

有効値

ID属性内での有効な文字と書式のルールはユーザーがサポートしたいHTMLのバージョンによって異なります。具体的詳細については、<http://www.w3.org/> で適切な仕様を確認してください。

全てのバージョンのHTML、CSSおよびjQueryのようなJavaScriptライブラリとの互換性を保証するためには、次のルールを守ってください。

- A-Z もしくは a-z の文字で始めます。
- その後に続くことができるのは：文字(A-Za-z)、数字 (0-9)、ハイフン ("-")およびアンダースコア("_")。

class

ウェブレットに割り当てられるCSSクラスもしくはクラスです。CSSクラスを使うと、外部スタイルシート内で定義され、ウェブレットに適用される一連のCSSスタイルを指定したり、ウェブレット内でエレメントを指定したりすることができます。複数のHTMLエレメントで作られた複雑なウェブレットでは、クラスはウェブレットの最外部のエレメントに適用されます。

省略値

空白。

有効値

1つ以上のスペースで分けられたCSSクラス名を含む文字列。

fieldContainWrapper

jQuery Mobileは、ラベルとフィールドを小型画面に垂直にレイアウトする複雑性を処理するので、ラベルとフィールドは大型画面上でもすべて一列に並びます。そのためには、各ラベル/フィールド/ペアを特定の属性を持つ<div>タグで囲む必要があります。fieldContainWrapperをTrueに設定し、Labelプロパティを使ってラベル・テキストを設定すると、すべての処理を実行します。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

hideIf

Trueと評価された時、ウェブレットを非表示にする式です。

省略値

False - ウェブレットは常に表示されます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

hideLabel

ラベルを非表示にします。現在このウェブレットのラベルを非表示にできないため、まだ支援技術にアクセス可能です (hideLabel プロパティが他のウェブレットで非表示にします)ので、このプロパティをTrueに設定することは label プロパティを ブランクの値に設定することと同じです。このウェブレットは他のウェブレットとの一貫性のためにあるため、今後ブラウザがラベルをアクセス可能なまま非表示にできるようになると、何も変更せずに自動的に非表示になります。

省略値

False - ラベルは非表示ではありません。

有効値

True、False、もしくはブール値を返す有効なXPath式。

label

ウェブレットに使うラベル・テキストを指定します。ウェブレットはこの値を使って<ラベル>タグを作成し、入力フィールドに正しく添付されるようにします。ラベルを表示する意図はない場合でも、アクセシビリティのために全てのウェブレットに意味のあるラベルの提供を推奨します。hideLabelプロパティを使って支援技術に使用可能なままラベルを非表示にします。

省略値

ブランク - 自動的に生成されるフィールドはリポジトリ定義から値をとります。

有効値

文字列値。

mini

Trueに設定されると、垂直高さがより低い、よりコンパクトなウェブレットを表示します。スペースが限られているツールバーなどの場所で便利です。

省略値

False - 標準サイズが使われます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

orientation

グループ内のコントロールの並べ替え方法を指定します。

省略値

vertical。

有効値

horizontalもしくはvertical。

style

ウェブレットに適用するCSSスタイルの文字列を指定します。このプロパティを使うと、レイアウト・スタイルシートに定義されているどんな値も書き換えるウェブレットにCSSのstyleプロパティを設定することができます。

省略値

空白。

有効値

セミコロンで分けられた有効なCSSプロパティと値。

swatch

ウェブレットに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

internal_id

ウェブラーチン・デザインに含まれるカスタム・コンテンツにウェブレットを接続するためにWAMエディターとウェブレットに使われる一意のIDです。このプロパティはWAMエディターが自動的に構成します。手作業で修正しないでください。

9.4.8 コンボボックス選択メニュー (std_dropdown)

この選択メニューは、非表示でjQuery Mobileフレームワークの外観に一致するカスタム・スタイルの選択ボタンで置換えられたネイティブの選択エレメントをベースにしています。

省略値では、フレームワークはカスタム・ボタンを使ってネイティブOSのオプション・メニューを活用します。ボタンがクリックされると、ネイティブOSメニューが開きます。値が選択されてメニューが閉じられると、カスタム・ボタンのテキストが選択された値に一致するように更新されます。

コンボボックス選択メニュー・ウェブレットもネイティブOSメニューの代わりにカスタム・メニューを生成するオプションを提供します。カスタム・メニューは無効化されたオプションと複数選択をサポートし(これらのネイティブ・モバイルOSサポートに一貫性はありませんが)、プレースホルダー値を扱う的確な方法を追加します。詳細については、[useNativeControl](#) プロパティを参照してください。

プロパティ - コンボボックス選択メニュー (std_dropdown)

このウェブレットのプロパティは以下のとおりです。

| | | |
|-------------------------------------|--------------------------------------|--------------------------------------|
| addErrorDiv | id | selectorValueField |
| autofocus | inline | shadow |
| class | items | style |
| corners | label | swatch |
| disabled | mini | tabindex |
| displayMode | multiple | title |
| fieldContainWrapper | multiSelectCodeField | updateFieldsToSubmit |
| form | multiSelectListname | updateOnFieldChange |
| hideIf | name | updateProtocol |
| hideLabel | overlaySwatch | updateWamName |
| icon | placeholder | updateWrName |
| iconPosition | rdmlxDataType | useNativeControl |
| iconShadow | required | value |

name

このウェブレットがウェブルーチンに送信された時にウェブレットの値を受け取るフィールドの名前です。現在のウェブルーチンのWEB_MAPからフィールド上にウェブレットが生成される、もしくはドロップされると、このプロパティはそのフィールドの名前に設定されます。値を送信したいフィールドに別の名前がある場合、このプロパティをその名前に変更してください。

注: XHTMLテクノロジー・サービスにおいて、名前のプロパティはしばしば一意のIDとして使われます。これはjQuery Mobileテクノロジー・サービスには当てはまりません。jQuery Mobileテクノロジー・サービスではidプロパティを使います。ウェブレットの値がサーバーに送信されない場合、このプロパティはブランクのままにされます。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ターゲット・ウェブルーチンのフィールド名。

id

ウェブレットの一意的IDです。このプロパティは必須ではありませんが、送信されるとCSSもしくはJavaScriptで直接ウェブレットを参照することができます。値はページ内で一意でなければいけません。

注：jQuery Mobileは、Ajaxを使ってページをロードし、オプションとしてアニメーションを実行しながら、ページを現在のページに挿入します。これは、2つのウェブルーチンのコンテンツが1つのページ内に短時間存在する場合があることを意味します。両方のウェブルーチンが同じIDを持つウェブレットを含み、ユーザーがこの時間にウェブレットを参照するカスタムCSSもしくはJavaScriptを持つ場合、予期せぬ結果を得ることがあります。したがって、グローバル一意IDを作ってください。自動的に生成されてフィールドを表示するウェブレットは、ウェブルーチンとフィールドの名前を連結することにより、一意IDを作成します。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ID属性内での有効な文字と書式のルールはユーザーがサポートしたHTMLのバージョンによって異なります。具体的詳細については、<http://www.w3.org/> で適切な仕様を確認してください。

全てのバージョンのHTML、CSSおよびjQueryのようなJavaScriptライブラリとの互換性を保証するためには、次のルールを守ってください。

- A-Z もしくは a-z の文字で始めます。
- その後に続くことができるのは：文字(A-Za-z)、数字 (0-9)、ハイフン ("-")およびアンダースコア("_")。

value

ウェブレットの初期状態を指定します。この値は、最初にどの項目を選択するか決定するために各項目のコード値と比較されます。

省略値

空白。

有効値

文字列値。

addErrorDiv

Trueに設定されると、<div>エレメントがウェブレットの直後に追加され検証エラーを表示します。<div>は検証エラーがクリアされると、発生するまで再び非表示になります。

検証メソッドが設定されると、エラー<div>は非表示の時もページ上に自分のスペースを確保し、エラーが表示される時の画面の複雑な再配列を回避します。HTML 5の検証をフィールドではなくフォーム上で使用している場合、スペースを再利用するためにはプロパティをFalseに設定します。

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

autofocus

ページがロードされるとウェブレットが自動的にフォーカスを得るように指定します。1ページに1つのフィールドのみ、このプロパティをTrueに設定します。2つ以上のフィールドでオートフォーカスをTrueに設定すると、異なるブラウザ上で矛盾した結果が生成されることがあります。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

class

ウェブレットに割り当てられるCSSクラスもしくはクラスです。CSSクラスを使うと、外部スタイルシート内で定義され、ウェブレットに適用される一連のCSSスタイルを指定したり、ウェブレット内でエレメントを指定したりすることができます。複数のHTMLエレメントで作られた複雑なウェブレットでは、クラスはウェブレットの最外部のエレメントに適用されます。

省略値

空白。

有効値

1つ以上のスペースで分けられたCSSクラス名を含む文字列。

corners

ボタンに丸角が必要な場合に指定します。

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

disabled

ウェブレットを無効化するかどうかを指定します。無効化されたウェブレットは、使用もクリックもできません。無効化されたウェブレットの値は、フォームと共に送信されないことに注意してください。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

displayMode

ウェブレットが入力内容を受け取るのか、出力内容のみかを制御します。

省略値

空白 ('input'の意味です)。

有効値

input または output。

fieldContainWrapper

jQuery Mobileは、ラベルとフィールドを小型画面に垂直にレイアウトする複雑性を処理するので、ラベルとフィールドは大型画面上でもすべて一列に並びます。そのためには、各ラベル/フィールド/ペアを特定の属性を持つ<div>タグで囲む必要があります。fieldContainWrapperをTrueに設定し、Labelプロパティを使ってラベル・テキストを設定すると、すべての処理を実行します。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

form

このウェブレットが属するフォームを指定する、スペースで分けられたフォームのIDのリストです。送信ボタンをクリックしてフォームが送信されると、そのフォームに属する全てのフィールドがサーバーに送信されます。省略値では、<フォーム>タグ内の全てのフィールドがフォームに属します。このプロパティを使うと、値はフォームと共に送信したまま、ドキュメントの別の部分、<フォーム>タグの外側、もしくは別の<フォーム>タグの内側にフィールドを配置できます。

LANSAsの標準レイアウトにはページ全体をラップする<フォーム>タグが1つあるので、通常はこのプロパティをレイアウトに使う必要はありません。

省略値

ブランク。

有効値

スペースで分けられたフォームのIDのリスト。

hideIf

Trueと評価された時、ウェブレットを非表示にする式です。

省略値

False - ウェブレットは常に表示されます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

hideLabel

ラベルをアクセス可能なまま非表示にします。これは、ラベルは非表示でもスクリーン・リーダーのような支援技術に使用可能なことを意味します。

省略値

False - ラベルは非表示ではありません。

有効値

True、Falseもしくはブール値を返す有効なXPath式。

icon

このボタンと共に使う jQuery Mobile アイコンを指定します。

省略値

省略値 - 下矢印のアイコンが使われます。

有効値

プロパティのドロップダウンにリストされている値。

iconPosition

ボタン・テキストに対するボタン・アイコンの位置を指定します。

省略値

ブランク - jQuery Mobile省略値の位置の左を使います。

有効値

- 左 - ボタン・テキストの左にアイコンを配置します。
- 右 - ボタン・テキストの右にアイコンを配置します。
- 上 - ボタン・テキストの上にアイコンを配置します。
- 下 - ボタン・テキストの下にアイコンを配置します。
- テキストなし - テキストなし (アイコンのみ) でボタンを描きます。

iconShadow

Trueに設定されるとボタンにテーマのシャドウを適用します。

省略値

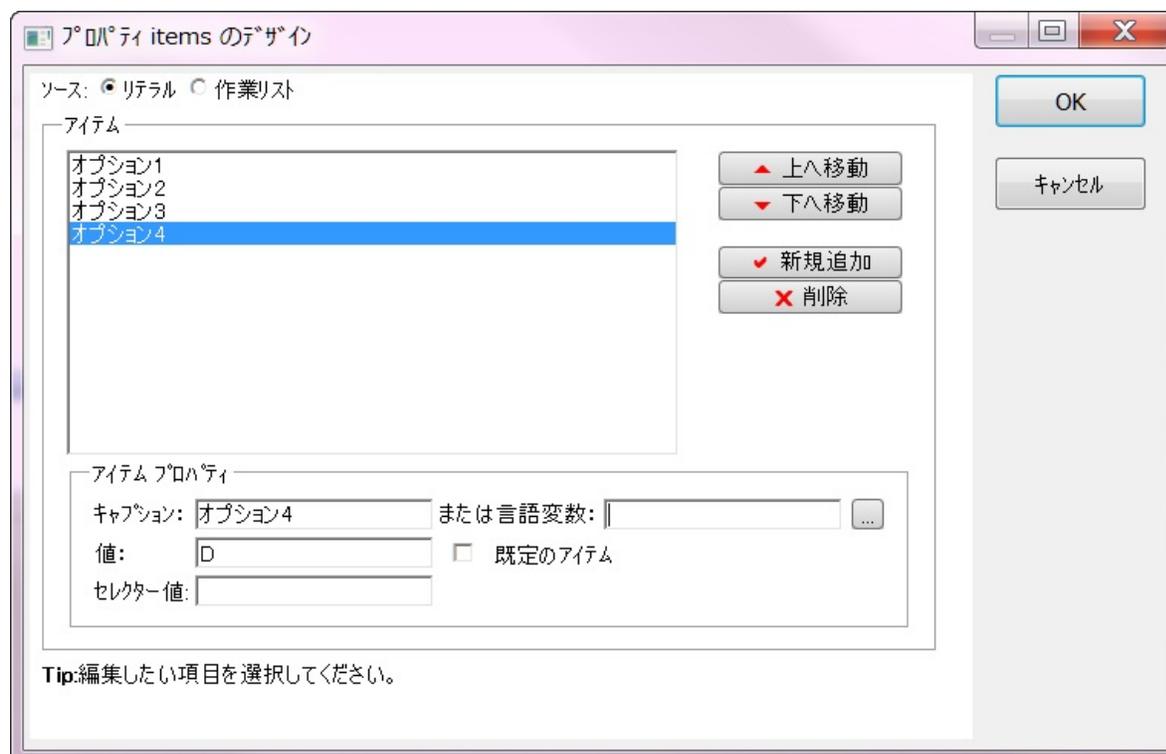
True - シャドウを適用します。

有効値

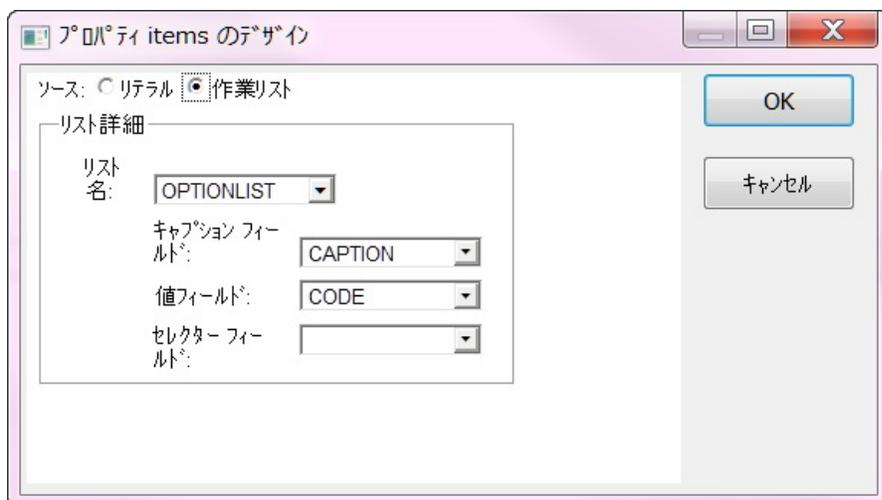
True、Falseもしくはブール値を返す有効なXPath式。

items

ウェブレットに表示する項目を指定するXMLノード・セット。設定はデザイナーでのみ可能です。デザイナーを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使ってください。プロパティ・デザイナーを使って、ハード・コーディングされた一連の項目もしくはそこから項目を取得する作業リストの名前を指定できます。作業リストを使う時、リストを出力ウェブ・マップ内で*JSONと指定してください。



ここではリストは4つの項目から構成されています。値が事前に選択されていない時に選択された状態にする項目には[規定の項目]のチェックボックスを使用します。セレクター値を使って、実行時にリストをフィルターして表示する値を絞ることもできます。



上記は作業リスト

を使うために構成された項目プロパティ・エディターです。セレクター・フィールドを使って、実行時にリストをフィルターして表示する値を絞ることもできます。

省略値

document(")/*/lxml:data/lxml:dropdown (このドロップダウンに項目が何も定義されていないことを示します。)

有効値

適用不可。(この値はシステムにより保守されます。)デザイナを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使ってください。

inline

Trueに設定されると、ボタンをインライン・ボタンのように機能させ、幅はボタンのコンテンツによって決定されます。Falseに設定されると、ボタンの幅はコンテンツに関係なくコンテナの全幅になります。

省略値

False - ボタンはコンテナの幅です。

有効値

True、Falseもしくはブール値を返す有効なXPath式。

label

ウェブレットに使うラベル・テキストを指定します。ウェブレットはこの値を使って<ラベル>タグを作成し、入力フィールドに正しく添付されるようにします。ラベルを表示する意図はない場合でも、アクセシビリティのために全てのウェブレットに意味のあるラベルの提供を推奨します。hideLabelプロパティを使って支援技術に使用可能なままラベルを非表示にします。

省略値

ブランク - 自動的に生成されるフィールドはリポジトリ定義の値を持ちます。

有効値

文字列値。

mini

Trueに設定されると、垂直高さがより低い、よりコンパクトなウェブレットを表示します。スペースが限られているツールバーなどの場所で便利です。

省略値

False - 標準サイズが使われます。

有効値

True、Falseもしくはブール値を返す有効なXPath式。

multiple

リスト・ボックスで複数選択を許可するかどうかを制御するブール値プロパティ。複数選択が許可される場合、multiSelectListnameとmultiSelectCodefieldプロパティが指定されていなければなりません。

複数選択のモバイル・ブラウザ・サポートに一貫性はないことに注意してください。ブラウザの互換性の問題を避けるためには、useNativeControlをFalseに設定し、jQuery Mobileにリストのカスタム・インターフェースを表示させてください。

省略値

False。

有効値

True、Falseもしくはブール値を返す有効なXPath式。

multiSelectCodeField

選択されたメニュー・アイテムのコード値を保持する
multiSelectListname作業リスト内のフィールド名。

省略値

空白。

有効値

フィールドの名前。プロパティ・シートの該当するドロップダウン・ボタンをクリックして、使用できるフィールドのリストから選択できます。

multiSelectListname

メニューの選択されたエントリーを含む作業リスト。作業リストは multiSelectCodeField プロパティに指定されたコード・フィールドのみを含んでいなければなりません。multiple プロパティが False の場合、このプロパティは無視されます。

省略値

空白。

有効値

作業リスト名。プロパティ・シートで該当するドロップダウン・ボタンをクリックして、使用できる作業リストのリストから選択できます。

overlaySwatch

省略値の jQuery Mobile テーマのスウォッチ、もしくはカラー・スキームを指定し、ダイアログ・ベースのカスタム選択メニューと小さいカスタム・メニューの外枠のオーバーレイ層に適用します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

a から z のうちの 1 文字。提供される有効値には、a から e までのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての [jQuery Mobile のドキュメント](#) を参照してください。

placeholder

入力フィールドに予想される値の短いヒントを指定します (例：サンプル値もしくは予想されるフォーマットの短い記述)。ヒントはフィールドが空の場合はフィールドに表示され、フィールドがフォーカスを得るかもしくは値を含んだ場合は消えます(詳細はブラウザによって異なります)。

省略値

空白。

有効値

文字列値。

rdmlxDatatype

ウェブレットに関連付けられたフィールドのRDMLXデータ・タイプを指定します。これは、あるウェブレットがデータの検証をするのに役立ちます。通常このプロパティは、ユーザーがデザインにフィールドを生成もしくはドロップすると自動的に設定されます。ウェブレットをデザインにドロップしてからフィールドに関連付ける場合は、ユーザーがプロパティを設定する必要があります。

省略値

自動的にWAMエディターで設定されない限りブランク。

有効値

データ・タイプで始まりデータ・タイプの要求の通りに追加のパラメータが続く、|で区切られた文字列：

integer|<最大長>

float|<最大長>

packed|<合計桁数>|<小数桁数>|<小数点記号>

signed|<合計桁数>|<小数桁数>|<小数点記号>

dec|<合計桁数>|<小数桁数>|<小数点記号>

alpha|<キーボード・シフト>|<最大長>

char|<キーボード・シフト>|<最大長>

varchar|<キーボード・シフト>|<最大長>

nchar|<キーボード・シフト>|<最大長>

nvarchar|<キーボード・シフト>|<最大長>

required

フォームの送信の前にウェブレットに値を入力するように指定します。このガイドが書かれている時点では、SafariとInternet Explorerはこのプロパティをサポートしていません。

注: HTML 5フォームの検証は、無効化属性を<フォーム>タグに
もしくは送信ボタン追加することによりオフにすることができます (これは、提供されている標準レイアウト上で
validationMethodを 'none' に設定することにより、自動的に実行されます)。さらに、ブラウザによってHTML 5の検証サ
ポート・レベルが異なるため、悪意のあるユーザーがクライアント側の検証を解除する方法が多数あります。したがって、ク
ライアント側の検証はユーザー・エクスペリエンスを向上させますが、信頼してはいけません。常にサーバー側の検証でバッ
ク・アップしてください。

省略値

False。

有効値

True、Falseもしくはブール値を返す有効なXPath式。

selectorValueField

ウェブレットに提供されたリストを表示用に絞るために、フィルターとして使用する値を持つフィールドの名前。表示リスト作成時は、この値はリストの"セレクター"カラムの値と比較されます。一致した場合、そのエントリーが表示リストに含まれます。

これはサーバー側の作業を減らす有効な方法です。毎回実行時にリスト・エントリーを計算する代わりに、ウェブルーチンで全ての可能な値とセレクター値を含むリストを事前に作成して出力することができます。そして、ブラウザがセレクター値を元にリストを絞って小さくします。しかし、サーバー負荷の減少と、より多くのデータをブラウザに送信するネットワーク帯域幅の増加のバランスをとる必要があることに留意してください。

これにより、サーバー要求をすることなく、動的にリストを更新することも可能になります。更新用にモニターされるフィールドもまた selectorValueFieldである場合、ウェブレットは最初に渡されたリストに新しいセレクター値を適用して、ウェブレット自身を再構築することができます。

省略値

空白。フィルターはされません。

有効値

現在のウェブルーチンの出力フィールドの名前。

shadow

Trueに設定されるとボタンにドロップ・シャドウのスタイルを適用します。

省略値

True。

有効値

True、Falseもしくははブール値を返す有効なXPath式。

style

ウェブレットに適用するCSSスタイルの文字列を指定します。このプロパティを使うと、レイアウト・スタイルシートに定義されているどんな値も書き換えるウェブレットにCSSのstyleプロパティを設定することができます。

省略値

空白。

有効値

セミコロンで分けられた有効なCSSプロパティと値。

swatch

ウェブレットに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

tabindex

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

- 1 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
- 2 ゼロもしくは空白(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
- 3 負数のtab_indexを持つオブジェクトは、タブの順序から省かれます。古いブラウザではサポートされないこともあるので注意してください。

省略値

空白。ウェブレットはソース順に選択されます。

有効値

空白もしくは有効な整数値。

title

ウェブレットに関する追加の助言的な情報を指定します。これは通常追加の必須ではない情報で、ユーザーがウェブレットの目的を理解するのを助けます。ブラウザが違くと扱い方が違うことがあります。例えば、マウスがウェブレットの上に乗ると、ほとんどのデスクトップ・ブラウザがタイトルをヒントとして表示します。スクリーン・リーダーのような支援技術は、ユーザーのためにタイトルを読みます。このガイドが書かれている時点では、モバイル機器のブラウザはタイトルを無視します。

省略値

空白。

有効値

有効なHTMLの属性文字列。

updateFieldsToSubmit

"更新"ウェブルーチンを呼び出してリスト値を更新する時に、送信するフィールドと値を指定します。

このプロパティの設定は、プロパティ・シート内で省略記号(...)のボタンを使って呼び出されるカスタム・プロパティ・デザイナーでのみ可能です。

| 名前 | 値 |
|------------|--------|
| DEPARTMENT | FLT |
| SECTION | SELSEC |

追加

削除

削除

OK

キャンセル

これは現在のウェブルーチン内の出力フィールド(#SELSEC)とターゲットのウェブルーチン内の入力フィールド([名前] 欄)にマップされているリテラル値("FLT") を表示します。

注：[名前] 欄のドロップダウンに正しい値が表示されるためには、このプロパティを編集する前に onClickWamName と onClickWrName プロパティが設定されている必要があります。

省略値

document(")*/lxml:data/lxml:json[not(@id)](このウェブレットに定義された項目がないことを示します。)

有効値

適用不可。(この値はシステムにより保守されます。)デザイナーを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使ってください。

updateOnFieldChange

変更をモニターするフィールドのID。モニターするフィールドが変更されると、選択ボックスは更新されます。updateWamName と updateWrNameが指定されている場合にitemsプロパティ内で作業リストが指定されていれば、ウェブレットはウェブルーチン呼び出して、作業リストの新しいコピーを要求します。それ以外の場合は、リストに既存のselectorValueFieldフィルターを再適用させて、そこからメニュー・リストを再構築します。

<入力>と<選択>エレメントの周囲に構築されたフィールドは監視することができます (テキスト・エリア以外の全ての標準フィールド)。

省略値

ブランク - どのフィールドも監視されません。

有効値

現在のページの文字列値<入力>もしくは<選択>エレメントのID。標準ウェブレットでは、ウェブレットのidプロパティと同じです。

updateProtocol

更新ウェブページを呼び出す時に使用するHTTPプロトコル。

省略値

ブランク - 現在使われているページと同じプロトコルを使います。

有効値

http:または https:

updateWamName

リストを更新する時に呼び出されるWAMの名前。

省略値

\$lweb_WAMName (現在のWAMを示します。)

有効値

WAMの名前。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるWAMのリストが表示されます。

updateWrName

リストを更新する時に呼び出されるウェブルーチンの名前。ウェブルーチンは*JSONとして定義されていなければなりません。

省略値

空白。

有効値

ウェブルーチンの名前。このウェブルーチンは、updateWamNameプロパティに指定されたWAMに存在してはいけません。プロパティシートで該当するドロップダウン・ボタンをクリックすると、使用できるウェブルーチンのリストが表示されます。

useNativeControl

ブラウザが、選択メニューの表示にブラウザのUIを使うか、もしくはjQuery Mobileが提供するUIを使うかを指定します。ブラウザの表示にブラウザのUIを使うと、ユーザーのデバイスに固有のユーザーが使い慣れたユーザー・エクスペリエンスが提供されます。ブラウザの表示にjQuery MobileのUIを使うと、テーマの適用が可能な全てのデバイスに一貫性のあるユーザー・エクスペリエンスが提供されます。

異なるブラウザ間では、複数選択のサポートの一貫性がないため、ユーザーのメニューが複数選択をサポートする場合、このプロパティをFalseに設定する必要があるかもしれません。

ネイティブ選択をパージしてカスタム・メニューを構築するとオーバーヘッドがあることに留意してください。ページ上で多数の選択肢があるか、もしくは選択肢に長いオプションのリストがある場合、ページのパフォーマンスに影響を与えることがあるため、カスタム・メニューは慎重に使用されるよう推奨します。

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

9.4.9 フッターバー (std_footer)

フッターバーはページ下部の Query Mobile ツールバーです。フッターバーはページの下部に配置するかもしくは画面の下部に固定できます (固定するとページがスクロールされてもフッターバーは同じ位置にとどまります)。ユーザーが画面をタップした時、フッターバーも表示もしくは非表示に構成できます。

注: 提供されている標準レイアウトではフッターバーが提供されません。フッター・ウェブレットは、カスタム・レイアウトを作成する時、もしくはフッターバーを提供しないレイアウトを使う時にのみ必要です。

プロパティ - フッターバー (std_footer)

このウェブレットのプロパティは以下のとおりです。

| | | |
|--------------------------------|-----------------------------|------------------------------------|
| fullscreenMode | id | persistentFooterId |
| hideIf | internal_id | position |
| | | swatch |

id

ウェブレットの一意的IDです。このプロパティは必須ではありませんが、送信されるとCSSもしくはJavaScriptで直接ウェブレットを参照することができます。値はページ内で一意でなければいけません。

注：jQuery Mobileは、Ajaxを使ってページをロードし、オプションとしてアニメーションを実行しながら、ページを現在のページに挿入します。これは、2つのウェブルーチンのコンテンツが1つのページ内に短時間存在する場合があることを意味します。両方のウェブルーチンが同じIDを持つウェブレットを含み、ユーザーがこの時間にウェブレットを参照するカスタムCSSもしくはJavaScriptを持つ場合、予期せぬ結果を得ることがあります。したがって、グローバル一意IDを作ってください。自動的に生成されてフィールドを表示するウェブレットは、ウェブルーチンとフィールドの名前を連結することにより、一意IDを作成します。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ID属性内での有効な文字と書式のルールはユーザーがサポートしたHTMLのバージョンによって異なります。具体的詳細については、<http://www.w3.org/> で適切な仕様を確認してください。

全てのバージョンのHTML、CSSおよびjQueryのようなJavaScriptライブラリとの互換性を保証するためには、次のルールを守ってください。

- A-Z もしくは a-z の文字で始めます。
- その後に続くことができるのは：文字(A-Za-z)、数字 (0-9)、ハイフン ("-")およびアンダースコア("_")。

fullscreenMode

フルスクリーン・フッターは、ドキュメント内のある位置を確保するのではなく、フッターがページのコンテンツを上書きするというものを除き、固定位置にあるフッターです。これはコンテンツが全画面を埋めるようにするフォト・ビューアーやビデオ・ビューアーといった没入型アプリケーションに便利で、ツールバーは非表示になるか、画面のタップで呼び出されて表示されます。フッターはこのモードにおいてページのコンテンツの上に位置するため、特定の状況で使うのがよいことに留意してください。

このプロパティはpositionが "fixed" に設定されない場合は無視されません。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

hideIf

Trueと評価された時、ウェブレットを非表示にする式です。

省略値

False - ウェブレットは常に表示されます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

persistentFooterId

フッターがグローバル・ナビゲーション・エレメントの場合、フッターがスクロールして画面から見えなくならないよう、固定して表示したい場合があります。固定のフッターを永続的にし、ページが遷移する時に動かないように表示させることもできます。jQuery Mobileに入っている永続的フッター機能を使うと実行できます。

遷移の間でフッターを永続的にするには、関連する全てのページのpersistentFooterIdにid値を割り当て、各ページに同じid値を使います。例えば、persistentFooterIdを現在のページとターゲットのページの"myfooter"に設定すると、フレームワークはページ・アニメーションの間、フッター・アンカーを同じ場所に保ちます。この効果は、フッターがposition="fixed"に設定された時のみ正しく機能し、遷移の間も表示されています。

省略値

空白 - フッターは永続的ではありません。

有効値

文字列値。

position

フッターは、いくつかの異なる方法でページに配置されます。省略値では、フッターは"inline"配置モードを使います。このモードでフッターは自然なドキュメントの流れにあり(省略値のHTMLのふるまい)、JavaScript かCSS配置サポートのどちらかにかかわらず、全てのデバイス上で表示されます。

"fixed"配置モードは、フッターをCSS固定配置をサポートするブラウザのビューポートの下部に固定します(iOS5+、Android 2.2+、BlackBerry 6などほとんどのデスクトップ・ブラウザが含まれます)。固定配置をサポートしないブラウザでは、フッターはページの静的なインライン位置に戻ります。

省略値

Inline。

有効値

inlineまたはfixed。

swatch

ウェブレットに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

internal_id

ウェブラーチン・デザインに含まれるカスタム・コンテンツにウェブレットを接続するためにWAMエディターとウェブレットに使われる一意のIDです。このプロパティはWAMエディターが自動的に構成します。手作業で修正しないでください。

9.4.10 レイアウト・グリッド(std_gridlayout)

モバイル機器の画面は幅が狭いため、モバイル機器での複数列のレイアウトは通常は推奨されませんが、小さいエレメントを並べて配置したい場合があります(ボタン、ナビゲーション・タブなど)。

Layout Gridウェブレットは、CSSベースの列を2列から5列構築する簡単な方法を提供します。

グリッドは100%の幅で、まったく見えず(枠線や背景はありません)、パディングやマージンがなく、そのためエレメントの中に入れられたエレメントのスタイルに干渉しません。

Layout Gridウェブレットは、RDMLX作業リスト上で反復することによって各セル内で同じコンテンツを繰り返すように設計されています。各セル内で異なるコンテンツのカスタム・グリッドを作成したい場合、必要なHTMLを直接作成してください。詳細は、[jQuery Mobileのドキュメント](#)の「Content Formatting」のセクションを参照してください。

プロパティ - レイアウト・グリッド (std_gridlayout)

このウェブレットのプロパティは以下のとおりです。

[columns](#) [internal_id](#) [isOutputOnly](#) [listname](#)

columns

テキストの領域に表示する列の数。

省略値

2。

有効値

2 - 5。

isOutputOnly

このウェブレットが、出力のためだけにlistnameプロパティで指定されたリストを使用していることを示します。リストが入力に使われている場合、LANSAのフレームワークでは、リスト・データをサーバーに正しく送信するために非表示フィールドの作成と追加の処理が必要です。このウェブレットが、出力のためだけにリストを使用している場合、このプロパティをTrueに設定することによってパフォーマンスを若干向上させ、同じリストを使う他のウェブレットとの競合のリスクを低減させることができます。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

listname

反復するRDMLX作業リストの名前です。リスト内で各行に1つ、セルが作成されます。リストが指定されない場合、複数のセルの1行が作成されます。

省略値

ブランク。

有効値

現在のウェブルーチンが出力する作業リスト名です。使用可能な（WAMで定義された）作業リストの一覧は、プロパティシートで該当するドロップダウン・ボタンをクリックすることで表示され、選択することができます。

internal_id

ウェブラーチン・デザインに含まれるカスタム・コンテンツにウェブレットを接続するためにWAMエディターとウェブレットに使われる一意のIDです。このプロパティはWAMエディターが自動的に構成します。手作業で修正しないでください。

9.4.11 ヘッダーバー (std_header)

ヘッダーバーはページ下部の Query Mobile ツールバーです。ヘッダーバーはページの上部に配置するかもしれませんが画面の上部に固定します (固定するとページがスクロールされてもヘッダーは同じ位置にとどまります)。ユーザーが画面をタップした時、ヘッダーバーも表示もしくは非表示に構成できます。

ヘッダーバーには特定のタイプのコンテンツが必要で、そのコンテンツの書式は特定の 방법으로設定します。具体的には、ヘッダーバーにはヘッダー・テキストと2つ以下のリンクもしくはボタンが必要です。ヘッダー・テキストは中央揃えされ、ボタンはヘッダーバーの両側に配置されます。例えば次のようになります。



省略値の構成に従わないヘッダーバーを作成したい場合、divのようなコンテナ内のユーザーのカスタム・スタイルのマークアップをラップします。ウェブレットは自動ボタンロジックをラップしたコンテンツには適用しません。

注: 提供されている標準レイアウトではヘッダーバーが提供されます。ヘッダーバーウェブレットは、カスタム・レイアウトを作成する時、もしくはヘッダーを提供しないレイアウトを使う時にのみ必要です。

プロパティ - ヘッダー バー (std_header)

このウェブレットのプロパティは以下のとおりです。

fullscreenMode id position
hideIf internal_id swatch

id

ウェブレットの一意的IDです。このプロパティは必要ではありませんが、送信されるとCSSもしくはJavaScriptで直接ウェブレットを参照することができます。値はページ内で一意でなければいけません。

注：jQuery Mobileは、Ajaxを使ってページをロードし、オプションとしてアニメーションを実行しながら、ページを現在のページに挿入します。これは、2つのウェブルーチンのコンテンツが1つのページ内に短時間存在する場合があることを意味します。両方のウェブルーチンが同じIDを持つウェブレットを含み、ユーザーがこの時間にウェブレットを参照するカスタムCSSもしくはJavaScriptを持つ場合、予期せぬ結果を得ることがあります。したがって、グローバル一意IDを作ってください。自動的に生成されてフィールドを表示するウェブレットは、ウェブルーチンとフィールドの名前を連結することにより、一意IDを作成します。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ID属性内での有効な文字と書式のルールはユーザーがサポートしたHTMLのバージョンによって異なります。具体的詳細については、<http://www.w3.org/> で適切な仕様を確認してください。

全てのバージョンのHTML、CSSおよびjQueryのようなJavaScriptライブラリとの互換性を保証するためには、次のルールを守ってください。

- A-Z もしくは a-z の文字で始めます。
- その後に続くことができるのは：文字(A-Za-z)、数字 (0-9)、ハイフン ("-")およびアンダースコア("_")。

fullscreenMode

フルスクリーン・ヘッダーは、ドキュメント内のある位置を確保するのではなく、ヘッダーがページのコンテンツを上書きするというを除き、固定位置にあるヘッダーです。これはコンテンツが全画面を埋めるようにするフォト・ビューアーやビデオ・ビューアーといった没入型アプリケーションに便利で、ツールバーは非表示になるか、画面のタップで呼び出されて表示されます。ヘッダーはこのモードにおいてページのコンテンツの上に位置するため、特定の状況で使うのがよいことに留意してください。

このプロパティはheaderPositionが "fixed" に設定されない場合は無視されます。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

hideIf

Trueと評価された時、ウェブレットを非表示にする式です。

省略値

False - ウェブレットは常に表示されます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

position

ヘッダーは、いくつかの異なる方法でページに配置されます。省略値では、ヘッダーは"inline"配置モードを使います。このモードでヘッダーは自然なドキュメントの流れにあり(省略値のHTMLのふるまい)、JavaScript かCSS配置サポートのどちらかにかかわらず、全てのデバイス上で表示されます。

"fixed"配置モードは、CSS固定配置をサポートするブラウザのビューポートの上部にヘッダーを固定します(iOS5+、Android 2.2+、BlackBerry 6などほとんどのデスクトップ・ブラウザが含まれます)。固定配置をサポートしないブラウザでは、ヘッダーはページの静的な、インライン位置に戻ります。

省略値

Inline。

有効値

inlineまたはfixed。

swatch

ウェブレットに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

internal_id

ウェブラーチン・デザインに含まれるカスタム・コンテンツにウェブレットを接続するためにWAMエディターとウェブレットに使われる一意のIDです。このプロパティはWAMエディターが自動的に構成します。手作業で修正しないでください。

9.4.12 HTMLリスト(std_html_list)

リストはデータの表示、ナビゲーション、結果リストおよびデータ入力に使われるため、jQuery Mobileには様々なタイプのリストと書式設定の例があり、最も一般的なデザインのパターンが入っています。

jQuery Mobileリストは単純なHTMLリスト (もしくはとして始まります。jQuery Mobileは必要なスタイルを全て適用し、リストビューをブラウザ・ウィンドウの全幅を埋める右矢印インジケータ付きのモバイルに適したビューに変換します。リスト項目をタップすると、フレームワークがリストアイテム内の最初のリンクをトリガーし、リンク内のURLにAJAX要求を発行し、DOM内に新しいページを作成し、その後ページの遷移を開始します。

HTML Listウェブレットは、リストを定義し、様々なオプションとリスト全体の省略値を構成します。リスト項目は1つ以上のHTMLリスト項目ウェブレットをHTMLリスト・コンテンツに追加して作成されます。

リストが他のタイプのコンテンツとともにページに埋め込まれる場合、インセット・リストは、このリストをコンテンツ・エリアにあるブロックにマージンと丸い角をつけ(テーマにより制御) 1つにまとめます。リストでinsetプロパティをTrueに設定すると、ウェブレットは以下のようなインセット・モード表示になります。



4項目のインセット・リスト(分割線として構成されたリスト)

注：インセットでないリストには全て標準で-15ピクセルのマージンがあり、コンテンツ・エリアで15ピクセルのパディングを無効にし、リストが画面の端までいっぱいにならないようにします。リストの上または下にほかのコンテンツを追加する場

合、ネガティブ・マージンによってエレメントが重複することがあり、カスタムCSS内で間隔の追加が必要になります。

プロパティ - HTMLリスト (std_html_list)

このウェブレットのプロパティは以下のとおりです。

| | | |
|---------------------------------|------------------------------------|-----------------------------|
| class | id | splitIcon |
| countSwatch | inset | splitSwatch |
| dividerSwatch | internal_id | swatch |
| hasSearchFilter | searchFilterSwatch | type |
| hideIf | | |

id

ウェブレットの一意のID。このプロパティは必須ではありませんが、送信されるとCSSもしくはJavaScriptで直接ウェブレットを参照することができます。値はページ内で一意でなければいけません。

注：jQuery Mobileは、Ajaxを使ってページをロードし、オプションとしてアニメーションを実行しながら、ページを現在のページに挿入します。これは、2つのウェブルーチンのコンテンツが1つのページ内に短時間存在する場合がありますことを意味します。両方のウェブルーチンが同じIDを持つウェブレットを含み、ユーザーがこの時間にウェブレットを参照するカスタムCSSもしくはJavaScriptを持つ場合、予期せぬ結果を得ることがあります。したがって、グローバル一意IDを作ってください。自動的に生成されてフィールドを表示するウェブレットは、ウェブルーチンとフィールドの名前を連結することにより、一意IDを作成します。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ID属性内での有効な文字と書式のルールはユーザーがサポートしたHTMLのバージョンによって異なります。具体的詳細については、<http://www.w3.org/> で適切な仕様を確認してください。

全てのバージョンのHTML、CSSおよびjQueryのようなJavaScriptライブラリとの互換性を保証するためには、次のルールを守ってください。

- A-Z もしくは a-z の文字で始めます。
- その後に続くことができるのは：文字(A-Za-z)、数字 (0-9)、ハイフン ("-")およびアンダースコア("_")。

class

ウェブレットに割り当てられるCSSクラスもしくはクラス。CSSクラスを使って、外部スタイルシート内で定義され、ウェブレットに適用する一連のCSSスタイルを指定したり、ウェブレット内でエレメントを指定したりすることができます。複数のHTMLエレメントから作られた複雑なウェブレットには、ウェブレットの最外部のエレメントにクラスが適用されます。

省略値

ブランク。

有効値

1つ以上のスペースで分けられたCSSクラス名を含む文字列。

countSwatch

リストのカウント・インジケータに適用する省略値のjQuery Mobile テーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - jQuery Mobile'の省略値のスウォッチ (c)を使います。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

dividerSwatch

リスト分割線に使う省略値のjQuery Mobileテーマ・スウォッチを指定します。

省略値

Default - jQuery Mobile'の省略値のスウォッチ (b)を使います。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

hasSearchFilter

jQuery Mobileは、単純なクライアント側の検索機能を使ってリストにフィルターをかけるとても簡単な方法を提供します。hasSearchFilterプロパティをTrueに設定するだけでリストにフィルターがかけられるようになります。次にフレームワークはリストの上に検索ボックスを追加し、現在の検索文字列をユーザー・タイプとして含まないリスト項目を除きます。

もっと高度なフィルターオプションの詳細は、リストビューに関する[jQuery Mobileのドキュメント](#)を参照してください。

省略値

False - 検索フィルターは追加されません。

有効値

True、False、もしくはブール値を返す有効なXPath式。

hideIf

Trueと評価された時、ウェブレットを非表示にする式です。

省略値

False - ウェブレットは常に表示されます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

inset

リストが他のタイプのコンテンツとともにページに埋め込まれる場合、挿入リストは、そのリストをコンテンツ・エリアにあるブロックに少しのマージンと丸角をつけてパッケージします (テーマにより制御されています)。insetをTrueに設定すると挿入リストの外観を適用します。

注：インセットでないリストには、全て標準で-15ピクセルのマージンがあり、コンテンツ・エリアに15ピクセルのパディングを無効にし、リストが画面の端までいっぱいにならないようにします。リストの上または下にほかのコンテンツを追加する場合、ネガティブ・マージンによってエレメントが重複することがあり、カスタムCSS内での間隔の追加が必要になります。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

searchFilterPlaceholder

検索フィールドに表示する短いヒントを指定します。ヒントはフィールドが空の場合はフィールドに表示され、フィールドがフォーカスを得るかもしくはデータを含んだ場合は消えます(詳細はブラウザによって異なります)。

省略値

ブランク - jQuery Mobileはフィルター項目の省略値を使います。

有効値

文字列値。

searchFilterSwatch

検索フィールドに適用する省略値のjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - jQuery Mobile'の省略値のスウォッチ (c)を使います。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

splitIcon

分割ボタンリスト内のボタンに使うjQuery Mobileアイコンを指定します。

省略値

Default - 右矢印のアイコンが使われます。

有効値

プロパティのドロップダウンにリストされている値。

splitSwatch

リストの分割ボタンに適用する省略値のjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - jQuery Mobile'の省略値のスウォッチ (b)を使います。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

swatch

ウェブレットに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

type

作成するHTMLリスト;各エントリーに番号が付いた順序付けされたリスト、もしくはエントリーに番号の付いていない順序付けされていないリストを指定します。

省略値

順序付けされていないリスト。

有効値

順序付けされたもしくは順序付けされていないリスト。

internal_id

ウェブラーチン・デザインに含まれるカスタム・コンテンツにウェブレットを接続するためにWAMエディターとウェブレットに使われる一意のIDです。このプロパティはWAMエディターが自動的に構成します。手作業で修正しないでください。

9.4.13 HTMLリスト項目 (std_html_li)

HTMLリスト項目を使って9.4.12 HTMLリスト(std_html_list)ウェブレットに項目を追加します。

ヒント:RDMLX作業リストウェブレットの中にHTMLリスト項目ウェブレットを入れ、RDMLX作業リストから項目を追加します。

プロパティ - HTMLリスト項目 (std_html_list)

このウェブレットのプロパティは以下のとおりです。

class id role
filterText internal_id swatch
hideIf

id

ウェブレットの一意のIDです。このプロパティは必須ではありませんが、送信されるとCSSもしくはJavaScriptで直接ウェブレットを参照することができます。値はページ内で一意でなければいけません。

注：jQuery Mobileは、Ajaxを使ってページをロードし、オプションとしてアニメーションを実行しながら、ページを現在のページに挿入します。これは、2つのウェブルーチンのコンテンツが1つのページ内に短時間存在する場合がありますことを意味します。両方のウェブルーチンが同じIDを持つウェブレットを含み、ユーザーがこの時間にウェブレットを参照するカスタムCSSもしくはJavaScriptを持つ場合、予期せぬ結果を得ることがあります。したがって、グローバル一意IDを作ってください。自動的に生成されてフィールドを表示するウェブレットは、ウェブルーチンとフィールドの名前を連結することにより、一意IDを作成します。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ID属性内での有効な文字と書式のルールはユーザーがサポートしたいHTMLのバージョンによって異なります。具体的詳細については、<http://www.w3.org/>

で適切な仕様を確認してください。

全てのバージョンのHTML、CSSおよびjQueryのようなJavaScriptライブラリとの互換性を保証するためには、次のルールを守ってください。

- A-Z もしくは a-z の文字で始めます。
- その後に続くことができるのは：文字(A-Za-z)、数字(0-9)、ハイフン("-")およびアンダースコア("_")。

class

ウェブレットに割り当てられるCSSクラスもしくはクラス。CSSクラスを使うと、外部スタイルシート内で定義され、ウェブレットに適用される一連のCSSスタイルを指定したり、ウェブレット内でエレメントを指定したりすることができます。複数のHTMLエレメントで作られた複雑なウェブレットでは、クラスはウェブレットの最外部のエレメントに適用されます。

省略値

空白。

有効値

1つ以上のスペースで分けられたCSSクラス名を含む文字列。

filterText

フィルターする時にこの行に使う代替テキストを指定します。リストのフィルターに関する詳細は、HTMLリスト・ウェブレットの[hasSearchFilter](#)プロパティを参照してください。

省略値

空白 - フィルターにはその行の実際のコンテンツが使われます。

有効値

文字列値。

hideIf

評価がTrueの時にウェブレットが非表示になる式。

省略値

False - ウェブレットは常に表示されます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

role

リスト項目は分割線に変換され、リスト項目を順序付け、グループ化します。これは、**role**プロパティを"List divider"に設定することにより行われます。省略値では、リスト分割線はバー・スウォッチ"b"でスタイルされます(省略値のテーマではブルー)が、親HTMLリスト・ウェブレット内で**dividerSwatch**プロパティを設定して分割線のテーマを指定することもできます。

省略値

[None]。

有効値

[None] およびリスト分割線。

swatch

ウェブレットに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

internal_id

ウェブルーチン・デザインに含まれるカスタム・コンテンツにウェブレットを接続するためにWAMエディターとウェブレットに使われる一意のID。このプロパティはWAMエディターが自動的に構成します。手作業で修正しないでください。

9.4.14 入力ボックス (std_input)

入力ボックス・ウェブレットは、<入力>エレメントを作成します。<入力>エレメントは、HTMLのページでのデータ入力の基本フォームです。複数行のテキスト・フィールドを除き、HTMLがサポートする(ボタンを含む)あらゆるタイプの入力コントロールの作成に使われます。jQuery Mobileフレームワークはモバイル環境とテーマの省略値のコントロールを自動的に向上させます。

入力ボックス・ウェブレットは全てのHTML5の入力属性をサポートします。しかしHTML5はかなり新しく、全ての詳細が明らかになってはいないため、全てのウェブ・ブラウザがHTML5の機能を同じようにサポートしているわけではなく、まったくサポートしていないものもあります。どのブラウザがどの機能をサポートしているかに関する最新の情報についてはWufoo.com [HTML 5](#)ページにアクセスしてください。

ヒント: 入力ボックスでチェックボックスとラジオ・ボタンを作成することができますが、チェックボックスとラジオ・ボタン・グループ・ウェブレットがRDMLXの作業をしやすくするように設計されているため、通常はこれらを使う方が簡単です。

プロパティ - 入力ボックス (std_input)

このウェブレットのプロパティは以下のとおりです。

| | | |
|---------------------|----------------|---------------|
| accept | formnovalidate | pattern |
| addErrorDiv | formtarget | placeholder |
| alt | height | rdmlxDatatype |
| autocomplete | hideIf | readonly |
| autofocus | hideLabel | required |
| checked | id | size |
| class | label | src |
| disabled | list | step |
| displayMode | max | style |
| fieldContainWrapper | maxlength | swatch |
| form | min | tabindex |
| formaction | mini | title |
| formenctype | multiple | type |
| formmethod | name | value |
| | | width |

id

ウェブレットの一意的IDです。このプロパティは必須ではありませんが、送信されるとCSSもしくはJavaScriptで直接ウェブレットを参照することができます。値はページ内で一意でなければいけません。

注：jQuery Mobileは、Ajaxを使ってページをロードし、オプションとしてアニメーションを実行しながら、ページを現在のページに挿入します。これは、2つのウェブルーチンのコンテンツが1つのページ内に短時間存在する場合があることを意味します。両方のウェブルーチンが同じIDを持つウェブレットを含み、ユーザーがこの時間にウェブレットを参照するカスタムCSSもしくはJavaScriptを持つ場合、予期せぬ結果を得ることがあります。したがって、グローバル一意IDを作ってください。自動的に生成されてフィールドを表示するウェブレットは、ウェブルーチンとフィールドの名前を連結することにより、一意IDを作成します。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ID属性内での有効な文字および書式のルールはユーザーがサポートしたいHTMLのバージョンによって異なります。具体的詳細については、<http://www.w3.org/> で適切な仕様を確認してください。

全てのバージョンのHTML、CSSおよびjQueryのようなJavaScriptライブラリとの互換性を保証するためには、次のルールを守ってください。

- A-Z もしくは a-z の文字で始めます。
- その後に続くことができるのは：文字(A-Za-z)、数字(0-9)、ハイフン("-")およびアンダースコア("_")。

name

このウェブレットがウェブルーチンに送信された時にウェブレットの値を受け取るフィールドの名前です。現在のウェブルーチンのWEB_MAPからフィールド上にウェブレットが生成される、もしくはドロップされると、このプロパティはそのフィールドの名前に設定されます。値を送信したいフィールドに別の名前がある場合、このプロパティをその名前に変更してください。

注: XHTMLテクノロジー・サービスにおいて、名前のプロパティはしばしば一意のIDとして使われます。これはjQuery Mobileテクノロジー・サービスには当てはまりません。jQuery Mobileテクノロジー・サービスではidプロパティを使います。ウェブレットの値がサーバーに送信されない場合、このプロパティはブランクのままにされます。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ターゲット・ウェブルーチンのフィールド名。

value

ウェブレットの値を指定します。この値は、テキスト入力タイプでは入力ボックスに表示されます。ボタン、チェックボックスおよびラジオタイプでは、フォームが送信される時に入力を選択されるとこの値が送信されます。

省略値

空白。

有効値

文字列値。

accept

入力タイプがファイルの場合、このプロパティはサーバーが受け入れるファイル・タイプを指定します。ファイル・タイプはMIMEタイプのカンマで区切られたリストを使って指定されます。ワイルドカードを指定し、例えば"image/*"を使ってイメージ・ファイルを指定することができます。有効なMIMEタイプの一覧は、[LANSA Webサイト](#)を参照してください。

注: 全てのブラウザがこの属性をサポートしているわけではなく、ブラウザによってはMIMEタイプを認識せず、このプロパティに依存してファイルの検証をしないでください。常にサーバー上で検証してください。

注: LANSAServerは"file"タイプの入力を扱いません。このプロパティは第三者のサーバーに送信する時のみ有効です。

省略値

空白 - 全てのタイプのファイルを受け入れます。

有効値

カンマで区切ったMIMEタイプのリスト。

addErrorDiv

Trueに設定されると、<div>エレメントがウェブレットの直後に追加され検証エラーを表示します。<div>は検証エラーがクリアされると、発生するまで再び非表示になります。

検証メソッドが設定されると、エラー<div>は非表示の時もページ上に自分のスペースを確保し、エラーが表示される時の画面の複雑な再配列を回避します。HTML 5の検証をフィールドではなくフォーム上で使用している場合、スペースを再利用するためにはプロパティをFalseに設定します。

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

alt

ユーザーが何らかの理由でイメージを表示できない場合（接続が遅い、src属性のエラー、もしくはユーザーがスクリーン・リーダーを使う場合など）、ユーザーのために代替テキストを指定します。このプロパティは type="image" のみに有効です。

省略値

空白。

有効値

文字列値。

autocomplete

入力フィールドのオートコンプリートを有効にするかどうかを指定します。オートコンプリートにより、ブラウザが値を予測します。ユーザーがフィールド内で入力を開始すると、ブラウザは過去に入力された値をもとに、フィールドに入力するオプションを表示します。autocompleteプロパティは、次の<入力>タイプと連携します：テキスト、検索、url、tel、email、パスワード、日付ピッカー、範囲および色。

注: これはAJAXのオートコンプリートではありません。ブラウザは過去にフィールドに入力された値を記憶しており、その履歴を使ってオートコンプリート機能を提供します。セキュリティ上の理由(例：ユーザーやパスワード・フィールド)やユーザーの役に立たない(数値入力フィールドなど)などの理由により、フィールドによってはこのオプションをオンにすることは望ましくないことがあります。

省略値

省略値 - ブラウザの省略値を使います。通常はオンになっていますがユーザーの選択によります。

有効値

on, offもしくは省略値。値"on"は、この機能を無効化するユーザーの選択設定を上書きしません。

autofocus

ページがロードされるとウェブレットが自動的にフォーカスを得るように指定します。1ページに1つのフィールドのみ、このプロパティをTrueに設定します。2つ以上のフィールドでオートフォーカスをTrueに設定すると、異なるブラウザ上で矛盾した結果が生成されることがあります。

省略値

False。

有効値

True、Falseもしくはブール値を返す有効なXPath式。

checked

チェックボックスもしくはラジオ・ボタンが最初に選択されるかどうかを指定します。type="checkbox" もしくは type="radio"にのみ有効です。

省略値

False - ウェブレットは確認されません。

有効値

True、False、もしくはブール値を返す有効なXPath式。

class

ウェブレットに割り当てられるCSSクラスもしくはクラスです。CSSクラスを使うと、外部スタイルシート内で定義され、ウェブレットに適用される一連のCSSスタイルを指定したり、ウェブレット内でエレメントを指定したりすることができます。複数のHTMLエレメントで作られた複雑なウェブレットでは、クラスはウェブレットの最外部のエレメントに適用されます。

省略値

空白。

有効値

1つ以上のスペースで分けられたCSSクラス名を含む文字列。

disabled

ウェブレットを無効化するかどうかを指定します。無効化されたウェブレットは、使用もクリックもできません。無効化されたウェブレットの値は、フォームと共に送信されないことに注意してください。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

displayMode

ウェブレットが入力を受け取るかどうか、出力を表示するかどうかを制御します。displayMode を 非表示に設定することは type プロパティを非表示に設定することと同じです。

省略値

ブランク ('input'の意味です)。

有効値

input, output もしくは hidden。

fieldContainWrapper

jQuery Mobileは、ラベルとフィールドを小型画面に垂直にレイアウトする複雑性を処理するので、ラベルとフィールドは大型画面上でもすべて一列に並びます。そのためには、各ラベル/フィールド/ペアを特定の属性を持つ<div>タグで囲む必要があります。fieldContainWrapperをTrueに設定し、Labelプロパティを使ってラベル・テキストを設定すると、全ての処理を実行します。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

form

このウェブレットが属するフォームを指定する、スペースで分けられたフォームのIDのリストです。送信ボタンをクリックしてフォームが送信されると、そのフォームに属する全てのフィールドがサーバーに送信されます。省略値では、<フォーム>タグ内の全てのフィールドがフォームに属します。このプロパティを使うと、値はフォームと共に送信したまま、ドキュメントの別の部分、<フォーム>タグの外側、もしくは別の<フォーム>タグの内側にフィールドを配置できます。

LANSAsの標準レイアウトにはページ全体をラップする<フォーム>タグが1つあるので、通常はこのプロパティをレイアウトに使う必要はありません。

省略値

ブランク。

有効値

スペースで分けられたフォームのIDのリスト。

formation

フォームを送信する時にフォーム・データの送信先を指定するURLです。type="submit"のみに有効です。通常、LANSAの実行時フレームワークは、onClickWrName と onClickWRフィールドに基づいて正しいURLを理解します。このプロパティを指定すると省略値のふるまいを上書きします。

省略値

空白。

有効値

有効なURL文字列。

formenctype

フォームを送信する際に使うエンコード・タイプを指定します。
type="submit"のみに有効です。省略値のタイプでほとんどのフォームをLANSAサーバーに送信することができるため、通常formenctypeを設定する必要はありません。LANSA以外のサーバーにフォームを送信する場合は、そのサーバーの特定の要求により別のエンコード・タイプが必要になることがあります。

省略値

ブランク - 特別に指定されない限り、親<フォーム>エレメントにより指定された省略値のapplication/x-www-form-urlencodedを使います。

有効値

| | |
|-----------------------------------|---|
| application/x-www-form-urlencoded | 送信する前に全ての文字がエンコードされます (スペースは"+"記号に変換され、特殊文字はASCII HEX値に変換されます)。 |
| multipart/form-data | 文字はエンコードされません。この値はファイル・アップロード・コントロールがある時に必要です。 |
| text/plain | スペースは"+"記号に変換されますが、特殊文字はエンコードされません。 |

formmethod

フォーム・データの送信に使うHTTPメソッドを指定します。
type="submit"のみに有効です。

省略値

空白 - 特別に指定されない限り、省略値の親<フォーム>エレメントにより指定された省略値のPOSTを使います。

有効値

'GET' または 'POST'。

formnovalidate

送信する前にフォームを検証してはいけないことを指定します。
type="submit"のみに有効です。このプロパティをTrueに設定すると親<フォーム>エレメント内でオンになっていた検証を抑制し、Falseに設定すると親<フォーム>エレメント内でオフになっていた検証をオンにしません。

提供されているレイアウトでは、<フォーム>の検証はレイアウト・ウェブレットのvalidationMethodプロパティを使って構成されます。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

formtarget

フォーム送信後の応答を表示するフレームもしくはウィンドウを指定します。type="submit"のみに有効です。

このプロパティはHTML 5の仕様の一部で、完全を期すために提供されます。このプロパティの使用には注意が必要です。ブラウザによってフレームおよびウィンドウのサポート・レベルが異なり、このプロパティの扱いも異なることがあります。jQuery MobileのAjaxメカニズムとの競合を起こすこともあります。

省略値

ブランク - 特別に指定されない限り、省略値のブランクでもある、親<フォーム>エレメントにより指定された値を使います(現在のページに表示)。

有効値

_blank, _self, _parent, _top, framename.

height

入力高さをピクセルで指定します。このプロパティは type="image" のみに有効です。

ヒント: イメージの height プロパティと width プロパティの両方を指定します。height と width が設定されている場合、イメージに必要なスペースはページがロードされる時に確保されます。しかし、これらのプロパティがないと、ブラウザは画像のサイズが分からず、そのため適切なスペースを確保できません。効果は、ページ・レイアウトがロード中(イメージのロード中)に変化することです。

省略値

blank - ブラウザはイメージをロードした後に高さを計算します。

有効値

整数値。

hideIf

Trueと評価された時、ウェブレットを非表示にする式です。

省略値

False - ウェブレットは常に表示されます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

hideLabel

ラベルをアクセス可能なまま非表示にします。これは、ラベルは非表示でもスクリーン・リーダーのような支援技術に使用可能なことを意味します。

省略値

False - ラベルは非表示ではありません。

有効値

True、False、もしくはブール値を返す有効なXPath式。

label

ウェブレットに使うラベル・テキストを指定します。ウェブレットはこの値を使って<ラベル>タグを作成し、入力フィールドに正しく添付されるようにします。ラベルを表示する意図はない場合でも、アクセシビリティのために全てのウェブレットに意味のあるラベルの提供を推奨します。hideLabelプロパティを使って支援技術に使用可能なままラベルを非表示にします。

省略値

ブランク - 自動的に生成されるフィールドはリポジトリ定義の値を持ちます。

有効値

文字列値。

list

ユーザーが入力しているときにブラウザが候補として使える値のリストを含んだ<datalist>エレメントのIDを指定します。このプロパティは、候補の値がユーザーが過去にこのフィールドに入力した値ではなく<datalist>エレメントによって提供されることを除いてautocompleteと同様の機能を提供します。

省略値

空白。

有効値

現在のドキュメントの中の<datalist>エレメントのIDです。

max

<入力>エレメントの最大値を指定します。次の入力タイプにのみ有効です：数字、範囲、日付、日付時刻、現地日時、月、時間および週。

注: ブラウザの値の使い方の詳細は違うことがあります。例えば、あるブラウザはユーザーが無効な数字を入力するのを防ぐかもしれませんが、別のブラウザはその値をフィールド検証に使うだけかもしれません。

省略値

空白 - 最大値は指定されていません。

有効値

<入力>タイプと一貫性のある値 (例：type=numberの数字、type=dateの日付)。

maxlength

<入力>エレメント内で許される最大文字数を指定します。ブラウザはSBCSとDBCS文字を区別しないため、このプロパティはフィールドのデータ長を制限しません。rdmlxDataTypeプロパティが正しく設定され、フォームの検証がオンになると、LANSAフレームワークはフィールドのデータ長を検証します。

省略値

空白 - 最大長はありません。

有効値

整数値。

min

<入力>エレメントの最大値を指定します。次の入力タイプにのみ有効です：数字、範囲、日付、日付時刻、現地日時、月、時間および週。

注: ブラウザの値の使い方の詳細は違うことがあります。例えば、あるブラウザはユーザーが無効な数字を入力するのを防ぐかもしれませんが、別のブラウザはその値をフィールド検証に使うだけかもしれません。

省略値

空白 - 最小値は指定されていません。

有効値

<入力>タイプと一貫性のある値 (例：type=numberの数字、type=dateの日付)。

mini

Trueに設定されると、垂直高さがより低い、よりコンパクトなウェブレットを表示します。スペースが限られているツールバーなどの場所で便利です。

省略値

False - 標準サイズが使われます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

multiple

ユーザーが<入力>エレメントに1つ以上の値を入力してもよいことを指定します。このプロパティは入力タイプ"email" と "file".に有効です。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

pattern

<入力>エレメントの値が確認されるという正規表現を指定します。ブラウザは<入力>値がこのチェックを引き渡さない場合、フォームの送信を許可しません。patternプロパティは、次の<入力>タイプと連携します：テキスト、検索、url、tel、email およびパスワード。

実際、このプロパティに使われる値は正規表現の一部です。正規表現は通常大きな文字列の中で部分的なテキストを検索するために使われます。検証では、このパターンはサブストリングだけではなく、値全体と一致しなければいけません。値全体と一致させるために、パターンは $^(?:$ をパターンの最初に、 $)\$$ を最後に追加することで値の最初と最後に固定するように修正されます。

ヒント: JavaScriptの正規表現に関する詳細は、この[JavaScript チュートリアル](#)を参照してください。

省略値

空白。

有効値

有効なJavaScriptの正規表現。

placeholder

入力フィールドに予想される値の短いヒントを指定します (例：サンプル値もしくは予想されるフォーマットの短い記述)。ヒントはフィールドが空の場合はフィールドに表示され、フィールドがフォーカスを得るかもしくは値を含んだ場合は消えます(詳細はブラウザによって異なります)。

省略値

空白。

有効値

文字列値。

rdmlxDatatype

ウェブレットに関連付けられたフィールドのRDMLXデータ・タイプを指定します。これは、あるウェブレットがデータの検証をするのに役立ちます。通常このプロパティは、ユーザーがデザインにフィールドを生成もしくはドロップすると自動的に設定されます。ウェブレットをデザインにドロップしてからフィールドに関連付ける場合は、ユーザーがプロパティを設定する必要があります。

省略値

自動的にWAMエディターで設定されない限りブランク。

有効値

データ・タイプで始まりデータ・タイプの要求の通りに追加のパラメータが続く、|で区切られた文字列：

integer|<最大長>

float|<最大長>

packed|<合計桁数>|<小数桁数>|<小数点記号>

signed|<合計桁数>|<小数桁数>|<小数点記号>

dec|<合計桁数>|<小数桁数>|<小数点記号>

alpha|<キーボード・シフト>|<最大長>

char|<キーボード・シフト>|<最大長>

varchar|<キーボード・シフト>|<最大長>

nchar|<キーボード・シフト>|<最大長>

nvarchar|<キーボード・シフト>|<最大長>

readonly

入力の読み取り専用状態を設定します。読み取り専用入力フィールドは修正できません(しかし、フィールドにタブをつける、反転表示させる、およびフィールドからテキストをコピーすることはできます)。

読み取り専用属性を設定し、他の何らかの条件が満たされるまで(チェックボックスの選択など)ユーザーに値を変更させないようにすることができます。条件が満たされると、JavaScriptが読み取り専用値を削除し、入力フィールドを編集可能にします。

省略値

False - 入力値は編集可能です。

有効値

True、False、もしくはブール値を返す有効なXPath式。

required

フォームの送信の前にウェブレットに値を入力するように指定します。このガイドが書かれている時点では、SafariおよびInternet Explorerはこのプロパティをサポートしていません。

注: 必要な属性は、次の<入力>タイプと連携します: テキスト、検索、url、tel、email、日付ピッカー、数字、チェックボックス、ラジオ・ボタンおよびファイル。

注: HTML 5フォームの検証は、無効化属性を<フォーム>タグもしくは送信ボタンに追加することによってオフにすることができます (これは、提供されている標準レイアウト上で validationMethodを 'none' に設定することにより、自動的に実行されます)。さらに、ブラウザによってHTML 5の検証サポート・レベルが異なり、悪意のあるユーザーがクライアント側の検証を解除する方法が多数あります。したがって、クライアント側の検証はユーザー・エクスペリエンスを向上させますが、信頼してはいけません。常にサーバー側の検証でバック・アップしてください。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

size

可視幅を<入力>エレメントの文字で指定します。サイズ属性は、次の<入力>タイプと連携します：テキスト、検索、電話、URL、e-mailおよびパスワード。

これは、入力可能なデータ長を指定するのではなく入力の表示される幅のみを指定します。入力されるデータの長さの制限については、[maxlength](#)プロパティを参照してください。

サイズ値は文字数で表されます。この数字を画面上のピクセル数に変換するテクニックは、ブラウザによって微妙に異なります。<入力>幅をより正確に制御するには、CSSを使って幅を設定します。

省略値

blank - ブラウザはブラウザの省略値を使い、それは通常20前後です。

有効値

整数値。

src

送信ボタンとして使う画像のURLを指定します。このプロパティは type="image" のみに有効です。

省略値

空白。

有効値

有効なURL文字列。

step

<入力>エレメントの正式な間隔数を指定します。例えばstep=3の場合、正式な間隔数は、-3, 0, 3, 6...となります。ステップ属性はmax と minプロパティとともに使われ、正式な値の範囲を作成します。次の入力タイプにのみ有効です：数字、範囲、日付、日付時刻、現地日時、月、時間および週。

注：ほとんどのブラウザで、省略値は1に設定されます。1にすると小数値が無効になります。ブラウザの検証を実行中で小数値を許可する必要がある場合は、この値を'any'に設定してください。

省略値

空白 - stepは指定されません。

有効値

数値または 'any'

style

ウェブレットに適用するCSSスタイルの文字列を指定します。このプロパティを使うと、レイアウト・スタイルシートに定義されているどんな値も書き換えるウェブレットにCSSのstyleプロパティを設定することができます。

省略値

空白。

有効値

セミコロンで分けられた有効なCSSプロパティと値。

swatch

ウェブレットに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

tabindex

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

- 1 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
- 2 ゼロもしくは空白(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
- 3 負数のtab_indexを持つオブジェクトは、タブの順序から省かれます。古いブラウザではサポートされないこともあるので注意してください。

省略値

空白。ウェブレットはソース順に選択されます。

有効値

空白もしくは有効な整数値。

title

ウェブレットに関する追加の助言的な情報を指定します。これは通常、必須ではない追加の情報で、ユーザーがウェブレットの目的を理解するのを助けます。ブラウザによってタイトルの扱い方が異なることがあります。例えば、マウスがウェブレットの上に乗ると、ほとんどのデスクトップ・ブラウザがタイトルをヒントとして表示します。スクリーンリーダーのような支援技術は、ユーザーのためにタイトルを読みます。このガイドが書かれている時点では、モバイル機器のブラウザはタイトルを無視します。

省略値

空白。

有効値

有効なHTMLの属性文字列。

type

type属性は、表示する<input>エレメントのタイプを指定します。HTML 5に定義されているタイプの一覧は以下の通りです。全てのタイプが全てのブラウザにサポートされているわけではありません。どのブラウザがどの機能をサポートしているかに関する最新の情報についてはWufoo.com HTML 5ページにアクセスしてください。

注: "number"のタイプを使う時は、ブラウザはstepプロパティの省略値を1にします。1にすると分数値の入力を防ぐことができます。必ずstepプロパティも正しく設定してください。

| 値 | 説明 |
|----------------|--|
| button | クリック可能ボタンを定義します (主にJavaScriptとともに使われ、スクリプトをアクティブ化します)。 |
| checkbox | チェックボックスを定義します。 |
| color | カラー・ピッカーを定義します。 |
| date | 日付コントロールを定義します (年、月および日 (時間は含まない)) |
| datetime | 日時コントロールを定義します (UTC時間帯ベースの月、日、時間、分、秒および1秒の何分の1) |
| datetime-local | 日時コントロールを定義します (月、日、時間、分、秒および1秒の何分の1(時間帯は含まない)) |
| email | e-mailアドレスのフィールドを定義します。 |
| file | ファイル選択フィールドと[...を参照]ボタンを定義します (ファイルのアップロードのため)。 |
| hidden | 非表示入力フィールドを定義します。 |
| image | 送信ボタンとしてイメージを定義します。 |
| month | 月と年コントロールを定義します(時間帯は含まない)。 |

| | |
|----------|---|
| number | 数字入力フィールドを定義します。 |
| password | パスワード・フィールドを定義します(文字は隠されま
す)。 |
| radio | ラジオ・ボタンを定義します。 |
| range | 正確な値が重要ではない数字を入力するコントロールを定義
します(スライダー・コントロールなど)。 |
| reset | リセット・ボタンを定義します(全てのフォームの値を省略
値にリセットします)。 |
| search | 検索文字列入力フィールドを定義します。 |
| submit | 送信ボタンを定義します。 |
| tel | 電話番号入力フィールドを定義します。 |
| text | 省略値。1行のテキスト・フィールドを定義します(省略値の
幅は20文字)。 |
| time | 時間入力フィールドを定義します(時間帯は含まない)。 |
| url | URL入力フィールドを定義します。 |
| week | 週と年コントロールを定義します(時間帯は含まない)。 |

省略値

text。

有効値

上記のテーブルの値。

width

入力幅をピクセルで指定します。このプロパティは type="image" のみに有効です。

ヒント: イメージの height プロパティと width プロパティの両方を指定します。Height と width が設定されている場合、イメージに必要なスペースはページがロードされる時に確保されます。しかし、これらのプロパティがないと、ブラウザは画像のサイズが分からず、そのため適切なスペースを確保できません。効果は、ページ・レイアウトがロード中(イメージのロード中)に変化することです。

省略値

blank - ブラウザはイメージをロードした後に高さを計算します。

有効値

整数値。

9.4.15 メッセージ (std_messages)

メッセージ・ウェブレットはRDMLX メッセージ・コマンドを使って作成されたメッセージを表示します。

The lateral sensor arrays have been reconfigured.

プロパティ - Messages (std_messages)

このウェブレットのプロパティは以下のとおりです。

[hideIf](#)

[swatch](#)

hideIf

評価がTrueの時にウェブレットが非表示になる式。Falseの場合、ウェブレットは表示するメッセージがある場合のみ表示され、ない場合は非表示になります。

省略値

False - ウェブレットは表示されません。

有効値

True、False、もしくはブール値を返す有効なXPath式。

swatch

ウェブレットに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

e。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

9.4.16 Mobiscroll 日付/時間ピッカー (std_mobiscroll)

Mobiscroll ウェブレットは、モバイル機器のために最適化されたカスタマイズ可能な日時ピッカーです。詳細は[Mobiscroll プロジェクトのサイト](#)を参照してください。

プロパティ - Mobiscroll 日付/時間ピッカー (std_mobiscroll)

このウェブレットのプロパティは以下の通りです。

| | | |
|---------------------|-------------|------------|
| class | id | stepMinute |
| dateFormat | label | stepSecond |
| dateOrder | mode | style |
| disabled | name | swatch |
| endYear | pickerTheme | timeFormat |
| fieldContainWrapper | rows | timeWheels |
| form | showOnFocus | title |
| hideIf | startYear | type |
| hideLabel | stepHour | value |

id

ウェブレットの一意のID。このプロパティは必須ではありませんが、送信されるとCSSもしくはJavaScriptで直接ウェブレットを参照することができます。値はページ内で一意でなければいけません。

注：jQuery Mobileは、Ajaxを使ってページをロードし、オプションとしてアニメーションを実行しながら、ページを現在のページに挿入します。これは、2つのウェブルーチンのコンテンツが1つのページ内に短時間存在する場合がありますを意味します。両方のウェブルーチンが同じIDを持つウェブレットを含み、ユーザーがこの時間にウェブレットを参照するカスタムCSSもしくはJavaScriptを持つ場合、予期せぬ結果を得ることがあります。したがって、グローバル一意IDを作ってください。自動的に生成されてフィールドを表示するウェブレットは、ウェブルーチンとフィールドの名前を連結することにより、一意IDを作成します。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ID属性内での有効な文字と書式のルールはユーザーがサポートしたHTMLのバージョンによって異なります。具体的詳細については、<http://www.w3.org/> で適切な仕様を確認してください。

全てのバージョンのHTML、CSSおよびjQueryのようなJavaScriptライブラリとの互換性を保証するためには、次のルールを守ってください。

- A-Z もしくは a-z の文字で始めます。
- その後に続くことができるのは：文字(A-Za-z)、数字(0-9)、ハイフン("-")およびアンダースコア("_")。

name

このウェブレットがウェブルーチンに送信された時にウェブレットの値を受け取るフィールドの名前です。現在のウェブルーチンのWEB_MAPからフィールド上にウェブレットが生成される、もしくはドロップされると、このプロパティはそのフィールドの名前に設定されます。値を送信したいフィールドに別の名前がある場合、このプロパティをその名前に変更してください。

注: XHTMLテクノロジー・サービスにおいて、名前のプロパティはしばしば一意のIDとして使われます。これはjQuery Mobileテクノロジー・サービスには当てはまりません。jQuery Mobileテクノロジー・サービスではidプロパティを使います。ウェブレットの値がサーバーに送信されない場合、このプロパティはブランクのままにされます。

省略値

ブランク。自動的に生成されてフィールドを表示するウェブレットは、自動的に設定される値を持ちます。

有効値

ターゲット・ウェブルーチンのフィールド名。

value

ウェブレットの初期値を指定します。この値は特定のフォーマットに準拠する必要があります。RDMLX日付もしくは時間フィールドを使うと、フォーマットは正しくなります。数字など他のフィールドと編集マスクを使う場合、ウェブルーチンが正しくフォーマットを出力することを自分の責任で確認してください。フォーマットはtypeプロパティの値によって決まります。

タイプ フォーマット

date yyyy-mm-dd

time HH:ii:ss

datetime yyyy-mm-ddTHH:ii:ss

上記では、yyyyが4桁の年の値で、mmは2桁の月の値、ddは2桁の日付の値、HHは24時間フォーマットの2桁の時間の値、iiは2桁の分の値、ssは2桁の秒の値です。

省略値

空白。

有効値

上記のようにフォーマットされた有効な日付、時間もしくは日時
の値。

class

ウェブレットに割り当てられるCSSクラスもしくはクラスです。CSSクラスを使うと、外部スタイルシート内で定義され、ウェブレットに適用される一連のCSSスタイルを指定したり、ウェブレット内でエレメントを指定したりすることができます。複数のHTMLエレメントで作られた複雑なウェブレットでは、クラスはウェブレットの最外部のエレメントに適用されます。

省略値

ブランク。

有効値

1つ以上のスペースで分けられたCSSクラス名を含む文字列。

dateFormat

解析され、表示される日付のフォーマットを指定します (m - 月(前にゼロがつかない)、mm - 月(2桁)、M - 月の略称、MM - 月の正式名、d - 日付(前にゼロがつかない) dd - 日付(2桁)、y-年(2桁)、yy - 年(4桁))。

省略値

空白 - 現在のstd_messagesファイル内で定義された、ローカライズされた省略値を使います (詳細は「3.10 ローカライズ」を参照してください)。

有効値

上記の書式設定文字を使った文字列。

dateOrder

月/日/年ホイールの表示順と書式設定を指定します。(m - 月(前にゼロがつかない)、mm - 月(2桁)、M - 月の略称、MM - 月の正式名、d - 日付(前にゼロがつかない) dd - 日付(2桁)、D - 曜日(略称)、DD - 曜日(正式名)、y-年(2桁)、yy - 年(4桁)。このオプションは特定のホイールを表示するかどうかも制御します。例: 'mmyy' を使って月と年ホイールのみを表示、'mmD ddy' を使って日付ホイール上で曜日と日付の両方を表示。

省略値

ブランク - 現在のstd_messagesファイル内で定義された、ローカライズされた省略値を使います(詳細は「3.10 ローカライズ」を参照してください)。

有効値

上記の書式設定文字を使った文字列。

disabled

ウェブレットを無効化するかどうかを指定します。無効化されたウェブレットは、使用もクリックもできません。無効化されたウェブレットの値は、フォームと共に送信されないことに注意してください。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

endYear

年ホイール上で最後に表示された年を指定します。

省略値

ブランク - 当年 +10を使います。

有効値

有効な4桁の年。

fieldContainWrapper

jQuery Mobileは、ラベルとフィールドを小型画面に垂直にレイアウトする複雑性を処理するので、ラベルとフィールドは大型画面上でもすべて一列に並びます。そのためには、各ラベル/フィールド/ペアを特定の属性を持つ<div>タグで囲む必要があります。fieldContainWrapperをTrueに設定し、Labelプロパティを使ってラベル・テキストを設定すると、全ての処理を実行します。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

form

このウェブレットが属するフォームを指定する、スペースで分けられたフォームのIDのリストです。送信ボタンをクリックしてフォームが送信されると、そのフォームに属する全てのフィールドがサーバーに送信されます。省略値では、<フォーム>タグ内の全てのフィールドがフォームに属します。このプロパティを使うと、値はフォームと共に送信したまま、ドキュメントの別の部分、<フォーム>タグの外側、もしくは別の<フォーム>タグの内側にフィールドを配置できます。

LANSAsの標準レイアウトにはページ全体をラップする<フォーム>タグが1つあるので、通常はこのプロパティをレイアウトに使う必要はありません。

省略値

ブランク。

有効値

スペースで分けられたフォームのIDのリスト。

hideIf

Trueと評価された時、ウェブレットを非表示にする式です。

省略値

False - ウェブレットは常に表示されます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

hideLabel

ラベルをアクセス可能なまま非表示にします。これは、ラベルは非表示でもスクリーン・リーダーのような支援技術に使用可能なことを意味します。

省略値

False - ラベルは非表示ではありません。

有効値

True、False、もしくはブール値を返す有効なXPath式。

label

ウェブレットに使うラベル・テキストを指定します。ウェブレットはこの値を使って<ラベル>タグを作成し、入力フィールドに正しく添付されるようにします。ラベルを表示する意図はない場合でも、アクセシビリティのために全てのウェブレットに意味のあるラベルの提供を推奨します。hideLabelプロパティを使って支援技術に使用可能なままラベルを非表示にします。

省略値

ブランク - 自動的に生成されるフィールドはリポジトリ定義の値を持ちます。

有効値

文字列値。

mode

ユーザーがフィールドをタップした時に表示されるピッカーのタイプを指定します。可能なタイプは次の2つがあります。

スクローラ - ユーザーは画面を読み取り、選択ホイールを回転させます。

クリックピック - ユーザーは+/-ボタンをタップし、表示される値を変更します。

省略値

スクローラ。

有効値

スクローラもしくはクリックピック。

pickerTheme

ピッカーの外観を設定します。提供されるテーマは：'jQuery Mobile'、'Android'、'Android ICS'、'Sense-UI' および 'iOS'です。

ヒント: スクローラ・マークアップ内でテーマ名とともに使われるcssクラスを前に付けることによりcssにカスタム・テーマを作成することができます。例：`.my-theme .dww { / My CSS / }`とし、'my-theme'にpickerThemeプロパティを設定します。

省略値

jQuery Mobile。

有効値

jQuery Mobile、iOS、Android、Android ICS、Sense-UI もしくは上記に定義されたテーマ。

rows

ホイールに表示する行の数を指定します。

省略値

3。

有効値

正の整数です。ウェブレットは、ユーザーがここに入力する数字をおおうとしますが、数字がユーザーの画面に収まるより大きいと配置の問題が起こり、セットとキャンセル・ボタンにアクセスできなくなります。

showOnFocus

Trueの場合、フィールドがフォーカスを得るとピッカーが自動的に表示します。このプロパティをFalseに設定する場合、自分の責任でJavaScriptを使って選んだ時間にピッカーを表示してください。例えば、ウェブレットが"MyDatePicker" idを持っている場合、以下のコードでピッカーを表示します。

```
$("#MyDatePicker").scroller("show");
```

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

startYear

年ホイール上で最初に表示された年を指定します。

省略値

ブランク - 当年 -10を使います。

有効値

有効な4桁の年。

stepHour

タイムピッカー上の時間の目盛りを指定します。

省略値

1。

有効値

1から24までの整数。

stepMinute

タイムピッカー上の分の目盛りを指定します。

省略値

1。

有効値

1から60までの整数。

stepSecond

タイムピッカー上の秒の目盛りを指定します。

省略値

1。

有効値

1から60までの整数。

style

ウェブレットに適用するCSSスタイルの文字列を指定します。このプロパティを使うと、レイアウト・スタイルシートに定義されているどんな値も書き換えるウェブレットにCSSのstyleプロパティを設定することができます。

省略値

空白。

有効値

セミコロンで分けられた有効なCSSプロパティと値。

swatch

ウェブレットに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

tabindex

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

- 1 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
- 2 ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
- 3 負数のtab_indexを持つオブジェクトは、タブの順序から省かれます。古いブラウザではサポートされないこともあるので注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

Blankもしくは有効な整数値。

timeFormat

解析され、表示される時間のフォーマット (h - 12時間フォーマット(前にゼロがつかない)、hh - 12時間フォーマット(前にゼロ)、H - 24時間フォーマット(前にゼロがつかない)、HH - 24時間フォーマット(前にゼロ)、i - 分(前にゼロがつかない)、ii - 分(前にゼロ)、s - 秒(前にゼロがつかない)、ss - 秒(前にゼロ)、小文字のam/pm、 - 大文字AM/PM)。

省略値

空白 - 現在のstd_messagesファイル内で定義された、ローカライズされた省略値を使います (詳細は「3.10 ローカライズ」を参照してください)。

有効値

上記の書式設定文字を使った文字列。

title

ウェブレットに関する追加の助言的な情報を指定します。これは通常、必須ではない追加の情報で、ユーザーがウェブレットの目的を理解するのを助けます。ブラウザによって扱い方が違うことがあります。例えば、マウスがウェブレットの上に乗ると、ほとんどのデスクトップ・ブラウザがタイトルをヒントとして表示します。スクリーン・リーダーのような支援技術は、ユーザーのためにタイトルを読みます。このガイドが書かれている時点では、モバイル機器のブラウザはタイトルを無視します。

省略値

空白。

有効値

有効なHTMLの属性文字列。

timeWheels

時間/分/秒ホイールの表示順と書式設定を指定します。(h - 12時間フォーマット(前にゼロがつかない)、hh - 12時間フォーマット(前にゼロ)、H - 24時間フォーマット(前にゼロがつかない)、HH - 24時間フォーマット(前にゼロ)、i - 分(前にゼロがつかない)、ii - 分(前にゼロ)、s - 秒(前にゼロがつかない)、ss - 秒(前にゼロ)、a - 小文字のam/pm、A - 大文字のAM/PM)。オプションは特定のホイールが表示されるかどうかを制御します。たとえば'HHii'を使って時間(24時間フォーマット)と分ホイールのみを表示するか、'hhiissa'を使って時間(12時間フォーマット)、分、秒およびam/pmホイールを表示するかなどです。

省略値

空白 - 現在のstd_messagesファイル内で定義された、ローカライズされた省略値を使います(詳細は「3.10 ローカライズ」を参照してください)。

有効値

上記の書式設定文字を使った文字列。

type

ウェブレットに表示されているデータ・タイプを指定します。

省略値

日付。

有効値

日付、時間、もしくは日時。

9.4.17 ナビゲーションバー (std_navbar)

jQuery Mobileにはごく基本的なnavbarウィジェットがあり、バーの中、主にヘッダーもしくはフッターバーの中にオプションのアイコンのあるボタンを最大で5つ提供します。ユーザーがページを移動しても固定されたままのタブバーのように動作する永続的navbarもあります。

ナビゲーションバーは順序付けされていないHTMLリスト()を使って作成されます。ナビゲーションバーにボタンを追加するには、(HTMLリスト項目ウェブレットを使うかを直接ソースに挿入して)リスト項目を追加しアンカーをリスト項目に入れます。ナビゲーションバーにはいくつでもボタンを追加することができます。5つ以上追加すると、ナビゲーションバーはボタンを単純に複数行にラップします。



ヒント: 選択した通りにボタンを表示するには、対応するアイコンに"ui-btn-active"クラスを追加します。

プロパティ - ナビゲーションバー (std_navbar)

このウェブレットのプロパティは以下の通りです。

id iconPosition

class swatch

hideIf internal_id

id

ウェブレットの一意のIDです。このプロパティは必須ではありませんが、送信されるとCSSもしくはJavaScriptで直接ウェブレットを参照することができます。値はページ内で一意でなければいけません。

注：jQuery Mobileは、Ajaxを使ってページをロードし、オプションとしてアニメーションを実行しながら、ページを現在のページに挿入します。これは、2つのウェブルーチンのコンテンツが1つのページ内に短時間存在する可能性があることを意味します。両方のウェブルーチンが同じIDを持つウェブレットを含み、ユーザーがこの時間にウェブレットを参照するカスタムCSSもしくはJavaScriptを持つ場合、予期せぬ結果を得ることがあります。したがって、グローバル一意IDを作ってください。自動的に生成されてフィールドを表示するウェブレットは、ウェブルーチンとフィールドの名前を連結することにより、一意IDを作成します。

省略値

ブランク。フィールドを表示するために自動的に生成されるウェブレットには、自動的に設定される値があります。

有効値

ID属性内での有効な文字と書式のルールはユーザーがサポートしたHTMLのバージョンによって異なります。具体的詳細については、<http://www.w3.org/> で適切な仕様を確認してください。

全てのバージョンのHTML、CSSおよびjQueryのようなJavaScriptライブラリとの互換性を保証するためには、次のルールを守ってください。

- A-Z もしくは a-z の文字で始めます。
- その後に続くことができるのは：文字(A-Za-z)、数字(0-9)、ハイフン("-")およびアンダースコア("_")。

class

ウェブレットに割り当てられるCSSクラスもしくはクラスです。CSSクラスを使うと、外部スタイルシート内で定義され、ウェブレットに適用される一連のCSSスタイルを指定したり、ウェブレット内でエレメントを指定したりすることができます。複数のHTMLエレメントから作られた複雑なウェブレットには、ウェブレットの最外部のエレメントにクラスが適用されます。

ナビゲーションバーを永続的にするには、値"ui-state-persist"をクラスに追加します。

省略値

ブランク。

有効値

1つ以上のスペースで分けられたCSSクラス名を含む文字列。

hideIf

Trueと評価された時、ウェブレットを非表示にする式です。

省略値

False - ウェブレットは常に表示されます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

iconPosition

Navbar内のボタンのボタン・テキストに関連するボタン・アイコンの位置を指定します。

省略値

空白 - jQuery Mobile省略値の位置の左を使います。

有効値

左 - ボタン・テキストの左にアイコンを配置します。

右 - ボタン・テキストの右にアイコンを配置します。

上 - ボタン・テキストの上にアイコンを配置します。

下 - ボタン・テキストの下にアイコンを配置します。

テキストなし - テキストなし (アイコンのみ) でボタンを描きます。

swatch

ウェブレットに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

internal_id

ウェブラーチン・デザインに含まれるカスタム・コンテンツにウェブレットを接続するためにWAMエディターとウェブレットに使われる一意のIDです。このプロパティはWAMエディターが自動的に構成します。手作業で修正しないでください。

9.4.18 ラジオ・ボタン・グループ (std_radbuttons)

RDMLXフィールドにリンクされた一連のラジオ・ボタンを作成します。ボタンの数とボタンの値は、作業リストの値から、またはウェブレットのitemsプロパティ経由で定義された一連の静的な値からも生成できます。

Choose a pet:

| | |
|----------------------------------|---------|
| <input checked="" type="radio"/> | Cat |
| <input type="radio"/> | Dog |
| <input type="radio"/> | Hamster |
| <input type="radio"/> | Lizard |

プロパティ - ラジオ・ボタン・グループ (std_radbuttons)

このウェブレットのプロパティは以下のとおりです。

| | | |
|-------------|---------------------|-----------------|
| id | fieldContainWrapper | label |
| name | form | mini |
| value | hideIf | orientation |
| disabled | hideLabel | rdmlxDatatype |
| displayMode | items | selectorValueEq |

id

ウェブレットの一意のIDです。このプロパティは必須ではありませんが、送信されるとCSSもしくはJavaScriptで直接ウェブレットを参照することができます。値はページ内で一意でなければいけません。

注：jQuery Mobileは、Ajaxを使ってページをロードし、オプションとしてアニメーションを実行しながら、ページを現在のページに挿入します。これは、2つのウェブルーチンのコンテンツが1つのページ内に短時間存在する場合があることを意味します。両方のウェブルーチンが同じIDを持つウェブレットを含み、ユーザーがこの時間にウェブレットを参照するカスタムCSSもしくはJavaScriptを持つ場合、予期せぬ結果を得ることがあります。したがって、グローバル一意IDを作ってください。自動的に生成されてフィールドを表示するウェブレットは、ウェブルーチンとフィールドの名前を連結することにより、一意IDを作成します。

省略値

ブランク。フィールドを表示するために自動的に生成されるウェブレットには、自動的に設定される値があります。

有効値

ID属性内での有効な文字と書式のルールはユーザーがサポートしたHTMLのバージョンによって異なります。具体的詳細については、<http://www.w3.org/> で適切な仕様を確認してください。

全てのバージョンのHTML、CSSおよびjQueryのようなJavaScriptライブラリとの互換性を保証するためには、次のルールを守ってください。

- A-Z もしくは a-z の文字で始めます。
- その後に続くことができるのは：文字(A-Za-z)、数字 (0-9)、ハイフン ("-")およびアンダースコア("_")。

name

このウェブレットがウェブルーチンに送信された時にウェブレットの値を受け取るフィールドの名前です。現在のウェブルーチンのWEB_MAPからフィールド上にウェブレットが生成される、もしくはドロップされると、このプロパティはそのフィールドの名前に設定されます。値を送信したいフィールドに別の名前がある場合、このプロパティをその名前に変更します。

注: XHTMLテクノロジー・サービスにおいて、名前のプロパティはしばしば一意のIDとして使われます。これはjQuery Mobileテクノロジー・サービスには当てはまりません。jQuery Mobileテクノロジー・サービスではidプロパティを使います。ウェブレットの値がサーバーに送信されない場合、このプロパティはブランクのままにされます。

省略値

ブランク。フィールドを表示するために自動的に生成されるウェブレットには、自動的に設定される値があります。

有効値

ターゲット・ウェブルーチンのフィールド名。

value

ウェブレットの値を指定します。ウェブレットはこの値と各項目のコード値を比較し、初期選択されるラジオ・ボタンを決定します。

省略値

空白。

有効値

文字列値。

disabled

ウェブレットを無効化するかどうかを指定します。無効化されたウェブレットは、使用もクリックもできません。無効化されたウェブレットの値は、フォームと共に送信されないことに注意してください。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

displayMode

ウェブレットが入力内容を受け取るのか、出力内容のみかを制御します。

省略値

空白 ('input'の意味です)。

有効値

input または output。

fieldContainWrapper

jQuery Mobileは、ラベルとフィールドを小型画面に垂直にレイアウトする複雑性を処理するので、ラベルとフィールドは大型画面上でもすべて一列に並びます。そのためには、各ラベル/フィールド/ペアを特定の属性を持つ<div>タグで囲む必要があります。fieldContainWrapperをTrueに設定し、Labelプロパティを使ってラベル・テキストを設定すると、全ての処理を実行します。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

form

このウェブレットが属するフォームを指定する、スペースで分けられたフォームのIDのリストです。送信ボタンをクリックしてフォームが送信されると、そのフォームに属する全てのフィールドがサーバーに送信されます。省略値では、<フォーム>タグ内の全てのフィールドがフォームに属します。このプロパティを使うと、値はフォームと共に送信したまま、ドキュメントの別の部分、<フォーム>タグの外側、もしくは別の<フォーム>タグの内側にフィールドを配置できます。

LANSAsの標準レイアウトにはページ全体をラップする<フォーム>タグが1つあるので、通常はこのプロパティをレイアウトに使う必要はありません。

省略値

ブランク。

有効値

スペースで分けられたフォームのIDのリスト。

hideIf

Trueと評価された時、ウェブレットを非表示にする式です。

省略値

False - ウェブレットは常に表示されます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

hideLabel

ラベルを非表示にします。現在このウェブレットのラベルを非表示にできないため、まだ支援技術にアクセス可能です (hideLabel プロパティが他のウェブレットで非表示にします)ので、このプロパティをTrueに設定することは label プロパティを ブランクの値に設定することと同じです。このウェブレットは他のウェブレットとの一貫性のためにあるため、今後ブラウザがラベルをアクセス可能なまま非表示にできるようになると、何も変更せずに自動的に非表示になります。

省略値

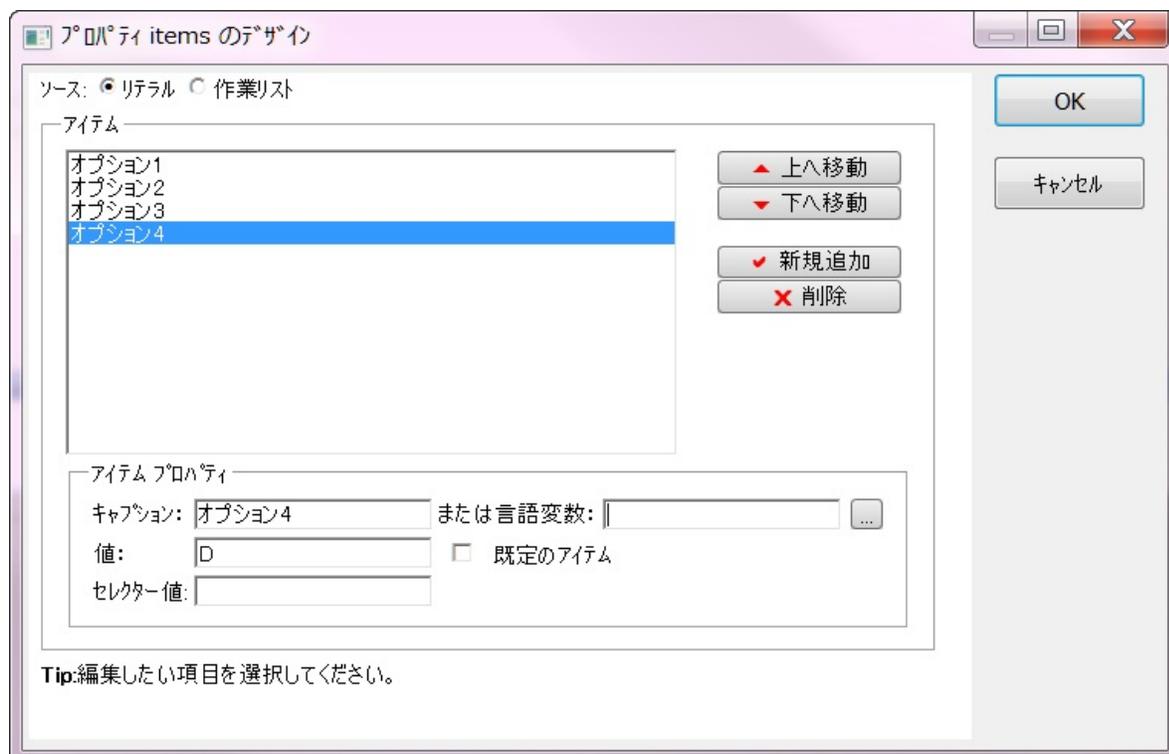
False - ラベルは非表示ではありません。

有効値

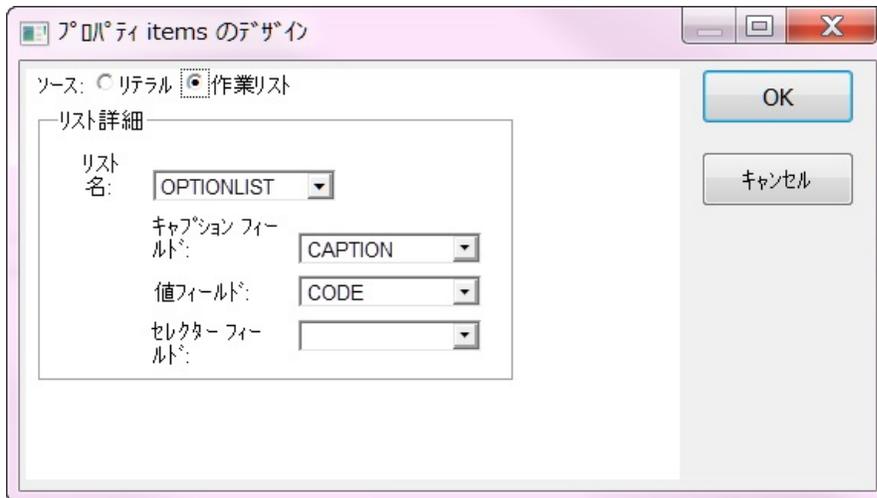
True、False、もしくはブール値を返す有効なXPath式。

items

ウェブレットに表示する項目を指定するXMLノード・セット。設定はデザイナーでのみ可能です。デザイナーを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使ってください。プロパティ・デザイナーを使って、ハード・コーディングされた一連の項目もしくはそこから項目を取得する作業リストの名前を指定できます。



ここではリストは4つの項目から構成されています。値が事前に選択されていない時に選択された状態にする項目には[規定のアイテム]のチェックボックスを使用します。セレクター値を使って、実行時にリストをフィルターして表示する値を絞ることもできます。



上記は作業リストを使うために構成された項目プロパティ・エディターです。セレクター・フィールドを使って、実行時にリストをフィルターして表示する値を絞ることもできます。

省略値

(")/*/lxml:data/lxml:radiobutton (このウェブレットに項目が定義されていないことを示します。)

有効値

適用不可。(この値はシステムにより保守されます。)デザイナを呼び出すには、プロパティ・シートの省略記号(...)のボタンを使ってください。

label

ウェブレットに使うラベル・テキストを指定します。ウェブレットはこの値を使って<ラベル>タグを作成し、入力フィールドに正しく添付されるようにします。ラベルを表示する意図はない場合でも、アクセシビリティのために全てのウェブレットに意味のあるラベルの提供を推奨します。hideLabelプロパティを使って支援技術に使用可能なままラベルを非表示にします。

省略値

ブランク - 自動的に生成されるフィールドはリポジトリ定義の値を持ちます。

有効値

文字列値。

mini

Trueに設定されると、垂直高さがより低い、よりコンパクトなウェブレットを表示します。これは、スペースが限られているツールバーなどの場所で便利です。

省略値

False - 標準サイズが使われます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

orientation

ラジオ項目が互いにどのように配置されるかを指定します。

省略値

垂直。

有効値

水平もしくは垂直。

rdmlxDatatype

ウェブレットに関連付けられたフィールドのRDMLXデータ・タイプを指定します。これは、あるウェブレットがデータの検証をするのに役立ちます。通常このプロパティは、ユーザーがデザインにフィールドを生成もしくはドロップすると自動的に設定されます。ウェブレットをデザインにドロップしてからフィールドに関連付ける場合は、ユーザーが設定してください。

省略値

自動的にWAMエディターで設定されない限りブランク。

有効値

データ・タイプで始まりデータ・タイプの要求の通りに追加のパラメータが続く、|で区切られた文字列：

integer|<最大長>

float|<最大長>

packed|<合計桁数>|<小数桁数>|<小数点記号>

signed|<合計桁数>|<小数桁数>|<小数点記号>

dec|<合計桁数>|<小数桁数>|<小数点記号>

alpha|<キーボード・シフト>|<最大長>

char|<キーボード・シフト>|<最大長>

varchar|<キーボード・シフト>|<最大長>

nchar|<キーボード・シフト>|<最大長>

nvarchar|<キーボード・シフト>|<最大長>

selectorValueEq

この値はウェブレットに表示されたりスト・アイテムを一部分に制限するために使用されます。セレクター値もしくはセレクター・フィールドはitemsプロパティ・エディター内で指定されていなければなりません。

省略値

空白。

有効値

文字列値。

9.4.19 RDMLX作業リスト (std_repeater)

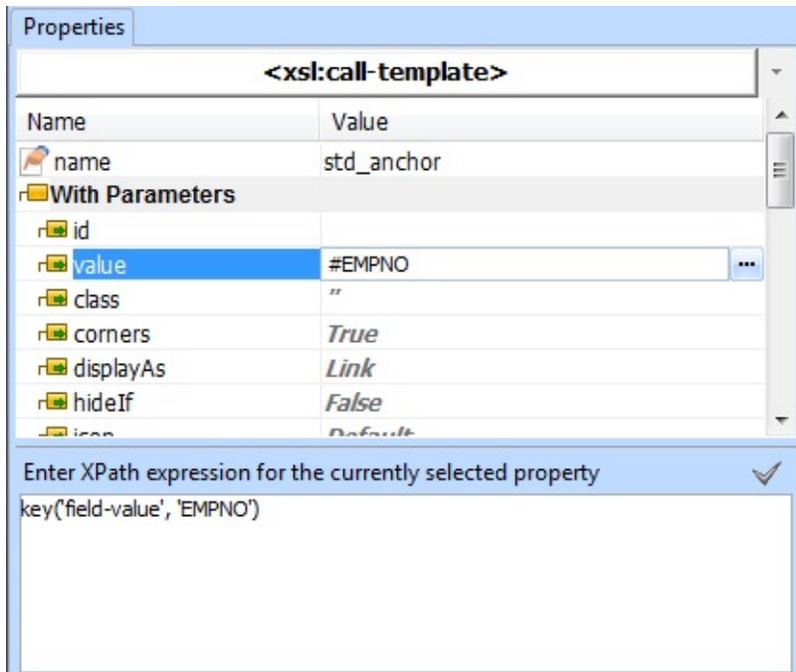
RDMLX作業リスト・ウェブレットは、作業リストを処理するための柔軟なメカニズムを提供します。このウェブレットは直接出力を生成しません。ユーザーがコンテンツ・エリアに入れたコンテンツを作業リスト内の各行で1回ずつ繰り返します。

繰り返しのコンテンツを最新のセマンティックHTMLページで扱う場合、繰り返しのコンテンツをHTMLリスト (もしくは)に入れ、繰り返しのコンテンツの各行をリスト項目()タグでラップするのが最も一般的なアプローチです。jQuery Mobileリストではこのアプローチを使います。RDMLX作業リスト・ウェブレットを使うと、テーブル行、divなど作りたい繰り返しのコンテンツを作成することができます。

Query Mobileリストを作るには、まずHTMLリスト・ウェブレットをデザインにドロップします。次にRDMLX作業リストウェブレットをHTMLリストのコンテンツ・エリアにドロップします。listnameプロパティをユーザーの作業リスト名に設定します。次にHTMLリスト項目ウェブレットを作業リストのコンテンツ・エリアにドロップします。最後に、各行の繰り返したいコンテンツをリスト項目のコンテンツ・エリアに入れます。

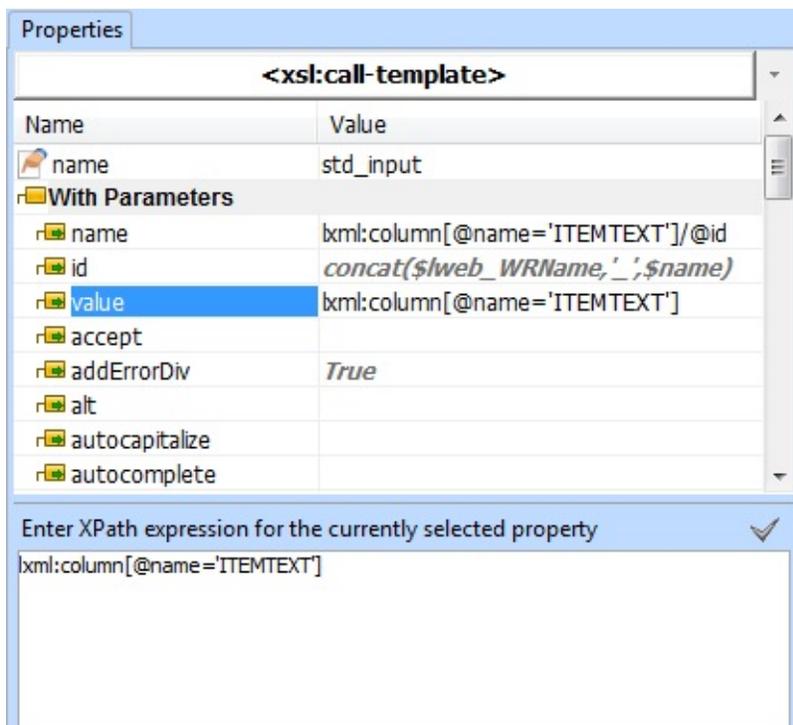
参照列値

ユーザーのウェブルーチンのデータはXMLドキュメントとして出力されてウェブルーチン・デザインに引き渡され、最終のHTML出力に変換されます。ウェブルーチン出力にアクセスするためには、デザインにXML内の必要なデータの場所を教えるXPath式が必要です。通常、フィールドからデータにアクセスする場合には式は必要ありません。フィールド名をプロパティ・エディター(例：#EMPNO)に入力すると、LANSAエディターが自動的に必要なXPath (例：key('field-value', 'EMPNO'))に変換します。



プロパティ値の中、および下のXPath入力ボックス内の対応するXPath式の中でフィールド名を表示するプロパティ・タブです。

リスト内の列を参照する場合は、この簡潔なテクニックを使うことはできません。代わりに、`lxml:column[@name='COLNAME']`をもつ列を参照してください。



参照されている列値を表示するプロパティ・タブです。

このようなXPath式をプロパティ・タブの下部にあるXPath入力エリアに入力する必要があります。プロパティに直接式を入力すると、エディターが式を文字列として扱おうとした結果、誤ったふるまいもしくはエラーにつながります。

注:XPath式内の全てのフィールド名とカラム名の参照は大文字を使用しなければなりません。リポジトリ・フィールドへの参照は全て、フィールドのショート・ネームを使用する必要があります。

プロパティ - RDMLX作業リスト (std_repeater)

このウェブレットのプロパティは以下のとおりです。

[internal_id](#)

[isOutputOnly](#)

[listname](#)

isOutputOnly

このウェブレットが、出力のためだけにlistnameプロパティで指定されたリストを使用していることを示します。リストが入力に使われている場合、LANSAのフレームワークでは、リスト・データをサーバーに正しく送信するために非表示フィールドの作成と追加の処理が必要です。このウェブレットが、出力のためだけにリストを使用している場合、このプロパティをTrueに設定することによってパフォーマンスを若干向上させ、同じリストを使う他のウェブレットとの競合のリスクを低減させることができます。

省略値

False。

有効値

True、False、もしくは、ブール値を返す有効なXPath式。

listname

反復するRDMLX作業リストの名前です。リスト内で各行に1つ、ウェブレット・コンテンツが再生されます。リストが指定されない場合、複数のセルの1行が作成されます。

省略値

空白。

有効値

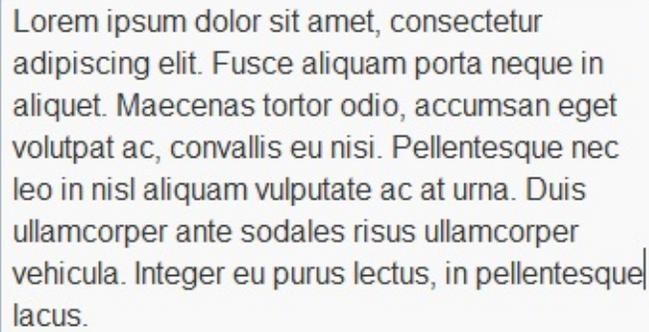
現在のウェブラーチンが出力する作業リスト名。使用可能な（WAMで定義された）作業リストの一覧は、プロパティシートで該当するドロップダウン・ボタンをクリックすることで表示され、選択することができます。

internal_id

ウェブラーチン・デザインに含まれるカスタム・コンテンツにウェブレットを接続するためにWAMエディターとウェブレットに使われる一意のIDです。このプロパティはWAMエディターが自動的に構成します。手作業で修正しないでください。

9.4.20 HTML テキストエリア (std_textarea)

HTML テキストエリアは長いテキスト値を複数行に分けて表示および入力するために使われる<テキストエリア>エレメントを作成します。

A screenshot of a text area containing Lorem Ipsum text. The text is displayed in a light gray box with a blue border and a small cursor at the end of the last line.

Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Fusce aliquam porta neque in
aliquet. Maecenas tortor odio, accumsan eget
volutpat ac, convallis eu nisi. Pellentesque nec
leo in nisl aliquam vulputate ac at urna. Duis
ullamcorper ante sodales risus ullamcorper
vehicula. Integer eu purus lectus, in pellentesque
lacus.

プロパティ - HTML テキストエリア (std_textarea)

このウェブレットのプロパティは以下の通りです。

| | | |
|-------------------------------------|-------------------------------|--------------------------|
| addErrorDiv | hideLabel | readonly |
| autofocus | id | required |
| class | label | rows |
| cols | maxlength | style |
| disabled | mini | swatch |
| displayMode | name | tabindex |
| fieldContainWrapper | placeholder | title |
| form | rdmlxDataType | value |
| hideIf | | wrap |

id

ウェブレットの一意的IDです。このプロパティは必須ではありませんが、送信されるとCSSもしくはJavaScriptで直接ウェブレットを参照することができます。値はページ内で一意でなければいけません。

注：jQuery Mobileは、Ajaxを使ってページをロードし、オプションとしてアニメーションを実行しながら、ページを現在のページに挿入します。これは、2つのウェブルーチンのコンテンツが1つのページ内に短時間存在する場合があることを意味します。両方のウェブルーチンが同じIDを持つウェブレットを含み、ユーザーがこの時間にウェブレットを参照するカスタムCSSもしくはJavaScriptを持つ場合、予期せぬ結果を得ることがあります。したがって、グローバル一意IDを作ってください。フィールドを表示するために自動的に生成されるウェブレットは、ウェブルーチンの名前をフィールドの名前と連結することにより、フィールドを表示します。

省略値

ブランク。フィールドを表示するために自動的に生成されるウェブレットには、自動的に設定される値があります。

有効値

ID属性内での有効な文字と書式のルールはユーザーがサポートしたいHTMLのバージョンによって異なります。具体的詳細については、<http://www.w3.org/> で適切な仕様を確認してください。

全てのバージョンのHTML、CSSおよびjQueryのようなJavaScriptライブラリとの互換性を保証するためには、次のルールを守ってください。

- A-Z もしくは a-z の文字で始めます。
- その後に続くことができるのは：文字(A-Za-z)、数字(0-9)、ハイフン("-")およびアンダースコア("_")。

name

このウェブレットがウェブルーチンに送信された時にウェブレットの値を受け取るフィールドの名前です。現在のウェブルーチンのWEB_MAPからフィールド上にウェブレットが生成される、もしくはドロップされると、このプロパティはそのフィールドの名前に設定されます。値を送信したいフィールドに別の名前がある場合、このプロパティをその名前に変更してください。

注: XHTMLテクノロジー・サービスにおいて、名前のプロパティはしばしば一意のIDとして使われます。これはjQuery Mobileテクノロジー・サービスには当てはまりません。jQuery Mobileテクノロジー・サービスではidプロパティを使います。ウェブレットの値がサーバーに送信されない場合、このプロパティはブランクのままにされます。

省略値

ブランク。フィールドを表示するために自動的に生成されるウェブレットには、自動的に設定される値があります。

有効値

ターゲット・ウェブルーチンのフィールド名。

value

テキストエリア内に表示されるテキストを指定します。

省略値

空白。

有効値

文字列値。

addErrorDiv

Trueに設定されると、<div>エレメントがウェブレットの直後に追加され検証エラーを表示します。<div>は検証エラーがクリアされると、発生するまで再び非表示になります。

検証メソッドが設定されると、エラー<div>は非表示の時もページ上に自分のスペースを確保し、エラーが表示される時の画面の複雑な再配列を回避します。HTML 5の検証をフィールドではなくフォーム上で使用している場合、スペースを再利用するためにはプロパティをFalseに設定します。

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

autofocus

ページがロードされるとウェブレットが自動的にフォーカスを得るように指定します。1ページに1つのフィールドのみ、このプロパティをTrueに設定します。2つ以上のフィールドでオートフォーカスをTrueに設定すると、異なるブラウザ上で矛盾した結果が生成されることがあります。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

class

ウェブレットに割り当てられるCSSクラスもしくはクラスです。CSSクラスを使うと、外部スタイルシート内で定義され、ウェブレットに適用される一連のCSSスタイルを指定したり、ウェブレット内でエレメントを指定したりすることができます。複数のHTMLエレメントで作られた複雑なウェブレットでは、クラスはウェブレットの最外部のエレメントに適用されます。

省略値

空白。

有効値

1つ以上のスペースで分けられたCSSクラス名を含む文字列。

cols

表示される幅を<テキストエリア>エレメントの文字で指定します。これは、入力可能なデータ長を指定するのではなく入力の表示される幅のみを指定します。入力されるデータ長の制限については、[maxlength](#)プロパティを参照してください。

cols値は文字数で表されます。この数字を画面上のピクセル数にどのように変換するかを決定するテクニックは、ブラウザによって微妙に異なります。<テキストエリア>幅をより正確に制御するには、CSSを使って幅を設定します。

省略値

空白 - ブラウザは通常約20の省略値のサイズを使います。

有効値

整数値。

disabled

ウェブレットを無効化するかどうかを指定します。無効化されたウェブレットは、使用もクリックもできません。無効化されたウェブレットの値は、フォームと共に送信されないことに注意してください。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

displayMode

ウェブレットが入力を受け取るか、出力のみかもしくは非表示かを制御します。

省略値

空白 ('input'の意味です)。

有効値

input, output もしくは hidden。

fieldContainWrapper

jQuery Mobileは、ラベルとフィールドを小型画面に垂直にレイアウトする複雑性を処理するので、ラベルとフィールドは大型画面上でもすべて一列に並びます。そのためには、各ラベル/フィールド/ペアを特定の属性を持つ<div>タグで囲む必要があります。fieldContainWrapperをTrueに設定し、Labelプロパティを使ってラベル・テキストを設定すると、全ての処理を実行します。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

form

このウェブレットが属するフォームを指定する、スペースで分けられたフォームのIDのリストです。送信ボタンをクリックしてフォームが送信されると、そのフォームに属する全てのフィールドがサーバーに送信されます。省略値では、<フォーム>タグ内の全てのフィールドがフォームに属します。このプロパティを使うと、値はフォームと共に送信したまま、ドキュメントの別の部分、<フォーム>タグの外側、もしくは別の<フォーム>タグの内側にフィールドを配置できます。

LANSAsの標準レイアウトにはページ全体をラップする<フォーム>タグが1つあるので、通常はこのプロパティをレイアウトに使う必要はありません。

省略値

ブランク。

有効値

スペースで分けられたフォームのIDのリスト。

hideIf

Trueと評価された時、ウェブレットを非表示にする式です。

省略値

False - ウェブレットは常に表示されます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

hideLabel

ラベルをアクセス可能なまま非表示にします。これは、ラベルは非表示でもスクリーン・リーダーのような支援技術に使用可能なことを意味します。

省略値

False - ラベルは非表示ではありません。

有効値

True、False、もしくはブール値を返す有効なXPath式。

label

ウェブレットに使うラベル・テキストを指定します。ウェブレットはこの値を使って<ラベル>タグを作成し、入力フィールドに正しく添付されるようにします。ラベルを表示する意図はない場合でも、アクセシビリティのために全てのウェブレットに意味のあるラベルの提供を推奨します。hideLabelプロパティを使って支援技術に使用可能なままラベルを非表示にします。

省略値

ブランク - 自動的に生成されるフィールドはリポジトリ定義の値を持ちます。

有効値

文字列値。

maxlength

テキストエリア内で認められる最大文字数を指定します。ブラウザはSBCSとDBCS文字を区別しないため、このプロパティはフィールドのデータ長を制限しません。rdmlxDataTypeプロパティが正しく設定され、フォームの検証がオンになると、LANSAフレームワークはフィールドのデータ長を検証します。

省略値

空白 - 最大長はありません。

有効値

整数値。

mini

Trueに設定されると、垂直高さがより低い、よりコンパクトなウェブレットを表示します。スペースが限られているツールバーなどの場所で便利です。

省略値

False - 標準サイズが使われます。

有効値

True、False、もしくはブール値を返す有効なXPath式。

placeholder

入力フィールドに予想される値の短いヒントを指定します (例：サンプル値もしくは予想されるフォーマットの短い記述)。ヒントはフィールドが空の場合はフィールドに表示され、フィールドがフォーカスを得るかもしくは値を含んだ場合は消えます(詳細はブラウザによって異なります)。

省略値

空白。

有効値

文字列値。

rdmlxDatatype

ウェブレットに関連付けられたフィールドのRDMLXデータ・タイプを指定します。これは、あるウェブレットがデータの検証をするのに役立ちます。通常このプロパティは、ユーザーがデザインにフィールドを生成もしくはドロップすると自動的に設定されます。ウェブレットをデザインにドロップしてからフィールドに関連付ける場合は、ユーザーがプロパティを設定する必要があります。

省略値

自動的にWAMエディターで設定されない限りブランク。

有効値

データ・タイプで始まりデータ・タイプの要求の通りに追加のパラメータが続く、|で区切られた文字列：

integer|<最大長>

float|<最大長>

packed|<合計桁数>|<小数桁数>|<小数点記号>

signed|<合計桁数>|<小数桁数>|<小数点記号>

dec|<合計桁数>|<小数桁数>|<小数点記号>

alpha|<キーボード・シフト>|<最大長>

char|<キーボード・シフト>|<最大長>

varchar|<キーボード・シフト>|<最大長>

nchar|<キーボード・シフト>|<最大長>

nvarchar|<キーボード・シフト>|<最大長>

readonly

テキストエリアの読み取り専用状態を設定します。読み取り専用入力フィールドは修正できません(しかし、フィールドにタブをつける、反転表示させる、およびフィールドからテキストをコピーすることはできます)。

読み取り専用属性を設定し、他の何らかの条件が満たされるまで(チェックボックスの選択など)ユーザーに値を変更させないようにすることができます。条件が満たされると、JavaScriptが読み取り専用値を削除し、テキストエリアを編集可能にします。

省略値

False - テキストエリアは編集可能です。

有効値

True、False、もしくはブール値を返す有効なXPath式。

required

フォームの送信の前にウェブレットに値を入力するように指定します。このガイドが書かれている時点では、SafariおよびInternet Explorerはこのプロパティをサポートしていません。

注: HTML 5フォームの検証は、無効化属性を<フォーム>タグもしくは送信ボタンに追加することによってオフにすることができます (これは、提供されている標準レイアウト上で validationMethodを 'none' に設定することにより、自動的に実行されます)。さらに、ブラウザによってHTML 5の検証サポート・レベルが異なり、悪意のあるユーザーがクライアント側の検証を解除する方法が多数あります。したがって、クライアント側の検証はユーザー・エクスペリエンスを向上させますが、信頼してはいけません。常にサーバー側の検証でバック・アップしてください。

省略値

False。

有効値

True、False、もしくはブール値を返す有効なXPath式。

rows

表示される高さを<テキストエリア>エレメントの行の中で指定します。これは、入力可能なデータ長を指定するのではなくテキストエリアの表示される高さのみを指定します。入力されるデータ長の制限については、[maxlength](#)プロパティを参照してください。テキストエリアでは、必要であれば自動的にスクロールバーが追加されます。

行値は行数で表されます。正確な高さは使われているフォントの設定によって異なります。<テキストエリア>の高さをより正確に制御するには、CSSを使って高さを設定します。

省略値

blank - ブラウザは通常2の省略値のサイズを使います。

有効値

整数値。

style

ウェブレットに適用するCSSスタイルの文字列を指定します。このプロパティを使うと、レイアウト・スタイルシートに定義されているどんな値も書き換えるウェブレットにCSSのstyleプロパティを設定することができます。

省略値

空白。

有効値

セミコロンで分けられた有効なCSSプロパティおよび値。

swatch

ウェブレットに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

tabindex

フォーム上のウェブレットのタブの順番を決定します。タブの順番は以下のようにtab_indexプロパティの値により決定されます。

- 1 正数のtab_indexを持つオブジェクトがtab_indexの昇順に(そして重複を避けるためにソース順に)選択されます。
- 2 ゼロもしくはブランク(省略値)のtab_indexを持つオブジェクトは、ソース順に選択されます。
- 3 負数のtab_indexを持つオブジェクトは、タブの順序からは省かれます。古いブラウザではサポートされないこともあるので注意してください。

省略値

ブランク。ウェブレットはソース順に選択されます。

有効値

Blankもしくは有効な整数値。

title

ウェブレットに関する追加の助言的な情報を指定します。これは通常、必須ではない追加の情報で、ユーザーがウェブレットの目的を理解するのを助けます。ブラウザによって扱い方が違うことがあります。例えば、マウスがウェブレットの上に乗ると、ほとんどのデスクトップ・ブラウザがタイトルをヒントとして表示します。スクリーン・リーダーのような支援技術は、ユーザーのためにタイトルを読みます。このガイドが書かれている時点では、モバイル機器のブラウザはタイトルを無視します。

省略値

空白。

有効値

有効なHTMLの属性文字列。

wrap

テキスト内のテキストがフォーム内で送信される時にどのようにラップされるかを指定します。プロパティは"soft" もしくは "hard"の2つの値をとります。

"soft"に設定されると、追加でラップされることはありません。ユーザーが挿入した改行のみサーバーに送信されます。

"hard"に設定されると、テキストはcolsプロパティの値を行の最大長として使い、テキストが折り返されます (行の最大長に最も近い単語の境界で折り返されます)。

注：このプロパティは、サーバーに送信されるテキストがどのようにフォーマットされるかに影響を与えます。テキストがブラウザ内でどのように表示されるかには影響を与えません。

省略値

soft - ラッピングされません。

有効値

soft もしくは hard。

9.5 レイアウト・ウェブレット

標準基本 WAM オプションのヘッダーとフッターのある単純な1列レイアウト。
レイアウト
(std_layout_v2)

フレキシブル 2つのコンテンツ・エリアのあるレイアウト・ウェブレット：メイン・コンテンツ・エリアと第2もしくは
レイアウト サイドバー・コンテンツ・エリア。自動的に小型(電話)画面のコンテンツ・エリアに入るように調整します。
(std_flex_layout)

9.5.1 標準基本WAMレイアウト (std_layout_v2)

標準基本WAMレイアウト・ウェブレットは、オプションのヘッダーとフッターのついた単純な1列レイアウトです。

プロパティ - 標準基本WAMレイアウト (std_layout_v2)

このウェブレットのプロパティは以下の通りです。

| | | |
|--------------------------------------|--------------------------------------|--|
| addBackButton | footerSwatch | showFooter |
| backButtonSwatch | headerFullscreenMode | showHeader |
| backButtonText | headerPosition | showMessages |
| contentSwatch | headerSwatch | validationErrorDisplay |
| footerFullscreenMode | pageSwatch | validationTime |
| footerPosition | persistentFooterId | windowTitle |

addBackButton

jQuery Mobileは自動的に戻るボタンを作成し、ヘッダーに追加します。フレームワークは、Ajax経由でロードされたページにボタンを追加するだけです。ブラウザによって完全にロードされたページはホーム・ページとして扱われるため戻るボタンがありません。

addBackButtonは、例えばユーザーがアプリケーションの入力ポイントを制御しているネイティブ・アプリケーション・ウェブビューもしくはウェブアプリケーション内で実行されている、主にクロムレスでインストールされたアプリケーションに有効です。誰かが検索エンジンの別のポイントやブックマークからユーザーのアプリケーションに入る可能性がある場合には、戻るボタンを避け、タップした時の行き先を明示するボタンを作成してください。

省略値

False - ヘッダーに戻るボタンは追加されません。

有効値

True、False、もしくはブール値を返す有効なXPath式。

backButtonSwatch

戻るボタンに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

backButtonText

追加された戻るボタンに表示するテキストを指定します。

省略値

ブランク - jQuery Mobileは"Back"の省略値を使います。

有効値

文字列値。

contentSwatch

レイアウトのコンテンツ・エリアに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

省略値 - jQuery Mobileの省略値のスウォッチを使います。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

footerFullscreenMode

フルスクリーン・フッターは、ドキュメント内のある位置を確保するのではなく、フッターがページのコンテンツを上書きするというものを除き、固定位置にあるフッターです。これはコンテンツが全画面を埋めるようにするフォト・ビューアーやビデオ・ビューアーといった没入型アプリケーションに便利で、ツールバーは非表示になるか、画面のタップで呼び出されて表示されます。フッターはこのモードにおいてページのコンテンツの上に位置するため、特定の状況で使うのがよいことに留意してください。

このプロパティはfooterPositionが "fixed" に設定されない場合は無視されます。

省略値

False。

有効値

True、False、もしくは、ブール値を返す有効なXPath式。

footerPosition

フッターは、いくつかの異なる方法でページに配置されます。省略値では、フッターは"inline"配置モードを使います。このモードでフッターは自然なドキュメントの流れにあり(省略値のHTMLのふるまい)、JavaScript かCSS配置サポートのどちらかにかかわらず、全てのデバイス上で表示されます。

"fixed"配置モードは、フッターをCSS固定配置をサポートするブラウザのビューポートの下部に固定します(iOS5+、Android 2.2+、BlackBerry 6などほとんどのデスクトップ・ブラウザが含まれます)。固定配置をサポートしないブラウザでは、フッターはページの静的な、インライン位置に戻ります。

省略値

Inline。

有効値

inlineまたはfixed。

footerSwatch

レイアウトのフッター・エリアに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

省略値 - jQuery Mobileの省略値のスウォッチを使います。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

headerFullscreenMode

フルスクリーン・ヘッダーは、ドキュメント内のある位置を確保するのではなく、ヘッダーがページのコンテンツを上書きするという除き、固定位置にあるヘッダーです。これはコンテンツが全画面を埋めるようにするフォト・ビューアーやビデオ・ビューアーといった没入型アプリケーションに便利で、ツールバーは非表示になるか、画面のタップで呼び出されて表示されます。ヘッダーはこのモードにおいてページのコンテンツの上に位置するため、特定の状況で使うのがよいことに留意してください。

このプロパティはheaderPositionが "fixed" に設定されない場合は無視されます。

省略値

False。

有効値

True、False、もしくは、ブール値を返す有効なXPath式。

headerPosition

ヘッダーは、いくつかの異なる方法でページに配置されます。省略値では、ヘッダーは"inline"配置モードを使います。このモードでは、ヘッダーは自然なドキュメントの流れにあり(省略値のHTMLのふるまい)、JavaScript かCSS配置サポートのどちらかにかかわらず、全てのデバイス上で表示されます。

"fixed"配置モードは、CSS固定配置をサポートするブラウザのビューポートの上部にヘッダーを固定します(iOS5+、Android 2.2+、BlackBerry 6などほとんどのデスクトップ・ブラウザが含まれます)。固定配置をサポートしないブラウザでは、ヘッダーはページの静的な、インライン位置に戻ります。

省略値

Inline。

有効値

inlineまたはfixed。

headerSwatch

レイアウトのヘッダー・エリアに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

省略値 - jQuery Mobileの省略値のスウォッチを使います。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

pageSwatch

レイアウトのページ全体(ヘッダー、コンテンツおよびフッター)に適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

省略値 - jQuery Mobileの省略値のスウォッチを使います。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

persistentFooterId

フッターがグローバル・ナビゲーション・エレメントの場合、フッターがスクロールして画面から見えなくならないよう、固定して表示したい場合があります。固定のフッターを永続的にし、ページが遷移する時に動かないように表示させることもできます。jQuery Mobileに入っている永続的フッター機能を使うと実行できます。

遷移の間でフッターを永続的にするには、関連する全てのページのpersistentFooterIdにid値を割り当て、各ページに同じid値を使います。例えば、persistentFooterIdを現在のページとターゲットのページの"myfooter"に設定すると、フレームワークはページ・アニメーションの間、フッター・アンカーを同じ場所に保ちます。この効果は、フッターがposition="fixed"に設定された時のみ正しく機能し、遷移の間も表示されています。

省略値

空白 - フッターは永続的ではありません。

有効値

文字列値。

showFooter

Trueの場合、レイアウトはレイアウトの下部にjQuery mobile footerを追加します。詳細は[jQuery Mobile](#)ドキュメントの「Toolbars」のセクションを参照してください。

フッターバーはカスタマイズ可能なコンテンツ・エリアです。カスタマイズするには、フッター上で右クリックし、「content.footerのコンテンツ・エリア」メニューから選択肢を選んでください。

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

showHeader

Trueの場合、レイアウトはレイアウトの下部にjQuery mobile headerを追加します。詳細は[jQuery Mobile](#)ドキュメントの「Toolbars」のセクションを参照してください。

ヘッダーバーはカスタマイズ可能なコンテンツ・エリアです。ヘッダーバーをカスタマイズするには、ヘッダー上で右クリックし、「content.headerのコンテンツ・エリア」メニューから選択肢を選んでください。

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

showMessages

Trueの場合、メッセージ・ウェブレットはコンテンツ・エリアの上部に置かれ、ウェブルーチンからのメッセージを表示します。

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

validationErrorDisplay

検証エラーをどのように表示するかを指定します。詳細については、[「フィールド検証」](#)を参照してください。

省略値

空白 - ブラウザの省略値メカニズムを使います。

有効値

'Browser Default' もしくは 'Error Div'

validationTime

検証エラーをいつ表示するかを指定します。詳細については、「[フィールド検証](#)」を参照してください。

省略値

After Submit - ユーザーが送信ボタンを押すと検証が実行されます。

有効値

Immediately, After Focus もしくは After Submit。

windowTitle

ページタイトルを指定します。このプロパティの値を使ってページの<タイトル>タグを指定します。

省略値

`$lweb_context/lxml:webroutine-title` - これはウェブラーチンの説明を参照するXPath式です。

有効値

文字列値。

9.5.2 フレキシブル レイアウト(std_flex_layout)

フレキシブル・レイアウトは2つのコンテンツ・エリアのあるレイアウト・ウェブレット：メイン・コンテンツ・エリアと第2もしくはサイドバー・コンテンツ・エリアです。レイアウトは自動的に小型(電話)画面の上の第2コンテンツ・エリアの位置を調整します。

プロパティ - フレキシブル レイアウト(`std_flex_layout`)

このウェブレットのプロパティは以下の通りです。

| | | |
|-----------------------------------|-----------------------------------|---|
| <code>addBackButton</code> | <code>footerSwatch</code> | <code>showFooter</code> |
| <code>backButtonSwatch</code> | <code>headerFullscreenMode</code> | <code>showHeader</code> |
| <code>backButtonText</code> | <code>headerPosition</code> | <code>showMessages</code> |
| <code>contentSwatch</code> | <code>headerSwatch</code> | <code>sidebarPositionSmallScreen</code> |
| <code>footerFullscreenMode</code> | <code>pageSwatch</code> | <code>validationErrorDisplay</code> |
| <code>footerPosition</code> | <code>persistentFooterId</code> | <code>validationTime</code> |
| | | <code>windowTitle</code> |

addBackButton

jQuery Mobileは自動的に戻るボタンを作成し、ヘッダーに追加します。フレームワークは、Ajax経由でロードされたページにボタンを追加するだけです。ブラウザによって完全にロードされたページはホーム・ページとして扱われるため戻るボタンがありません。

addBackButtonは、例えばユーザーがアプリケーションの入力ポイントを制御しているネイティブ・アプリケーション・ウェブビューもしくはウェブアプリケーション内で実行されている、主にクロムレスでインストールされたアプリケーションに有効です。誰かが検索エンジンの別のポイントやブックマークからユーザーのアプリケーションに入る可能性がある場合には、戻るボタンを避け、タップした時の行き先を明示するボタンを作成してください。

省略値

False - ヘッダーに戻るボタンは追加されません。

有効値

True、False、もしくは、ブール値を返す有効なXPath式。

backButtonSwatch

戻るボタンに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

Default - ウェブレットの省略値のスウォッチを使うか、もしくはコンテナからスウォッチを継承します。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

backButtonText

戻るボタンに表示するテキストを指定します。

省略値

空白 - jQuery Mobileは"Back"の省略値を使います。

有効値

文字列値。

contentSwatch

レイアウトのコンテンツ・エリアに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

省略値 - jQuery Mobileの省略値のスウォッチを使います。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

footerFullscreenMode

フルスクリーン・フッターは、ドキュメント内のある位置を確保するのではなく、フッターがページのコンテンツを上書きするというものを除き、固定位置にあるフッターです。これはコンテンツが全画面を埋めるようにするフォト・ビューアーやビデオ・ビューアーといった没入型アプリケーションに便利で、ツールバーは非表示になるか、画面のタップで呼び出されて表示されます。フッターはこのモードにおいてページのコンテンツの上に位置するため、特定の状況で使うのがよいことに留意してください。

このプロパティは**footerPosition**が "fixed" に設定されない場合は無視されます。

省略値

False。

有効値

True、False、もしくは、ブール値を返す有効なXPath式。

footerPosition

フッターは、いくつかの異なる方法でページに配置されます。省略値では、フッターは"inline"配置モードを使います。このモードでフッターは自然なドキュメントの流れにあり(省略値のHTMLのふるまい)、JavaScript かCSS配置サポートのどちらかにかかわらず、全てのデバイス上で表示されます。

"fixed"配置モードは、フッターをCSS固定配置をサポートするブラウザのビューポートの下部に固定します(iOS5+、Android 2.2+、BlackBerry 6などほとんどのデスクトップ・ブラウザが含まれます)。固定配置をサポートしないブラウザでは、フッターはページの静的なインライン位置に戻ります。

省略値

Inline。

有効値

inlineまたはfixed。

footerSwatch

レイアウトのフッター・エリアに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

省略値 - jQuery Mobileの省略値のスウォッチを使います。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

headerFullscreenMode

フルスクリーン・ヘッダーは、ドキュメント内のある位置確保するのではなく、ヘッダーがページのコンテンツを上書きするというを除き、固定位置にあるヘッダーです。これはコンテンツが全画面を埋めるようにするフォト・ビューアーやビデオ・ビューアーといった没入型アプリケーションに便利で、ツールバーは非表示になるか、画面のタップで呼び出されて表示されます。ヘッダーはこのモードにおいてページのコンテンツの上に位置するため、特定の状況で使うのがよいことに留意してください。

このプロパティはheaderPositionが "fixed" に設定されない場合は無視されます。

省略値

False。

有効値

True、False、もしくは、ブール値を返す有効なXPath式。

headerPosition

ヘッダーは、いくつかの異なる方法でページに配置されます。省略値では、ヘッダーは"inline"配置モードを使います。このモードでは、ヘッダーは自然なドキュメントの流れにあり(省略値のHTMLのふるまい)、JavaScript かCSS配置サポートのどちらかにかかわらず、全てのデバイス上で表示されます。

"fixed"配置モードは、CSS固定配置をサポートするブラウザのビューポートの上部にヘッダーを固定します(iOS5+、Android 2.2+、BlackBerry 6などほとんどのデスクトップ・ブラウザが含まれます)。固定配置をサポートしないブラウザでは、ヘッダーはページの静的な、インライン位置に戻ります。

省略値

Inline。

有効値

inlineまたはfixed。

headerSwatch

レイアウトのヘッダー・エリアに適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

省略値 - jQuery Mobileの省略値のスウォッチを使います。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

pageSwatch

レイアウトのページ全体(ヘッダー、コンテンツおよびフッター)に適用するjQuery Mobileテーマ・スウォッチもしくはカラー・スキームを指定します。

省略値

省略値 - jQuery Mobileの省略値のスウォッチを使います。

有効値

aからzのうちの1文字。提供される有効値には、aからeまでのスウォッチしか入っていません。独自のテーマやスウォッチの作成方法の詳細については、テーマについての[jQuery Mobileのドキュメント](#)を参照してください。

persistentFooterId

フッターがグローバル・ナビゲーション・エレメントの場合、フッターがスクロールして画面から見えなくならないよう、固定して表示したい場合があります。固定のフッターを永続的にし、ページが遷移する時に動かないように表示させることもできます。jQuery Mobileに入っている永続的フッター機能を使うと実行できます。

遷移の間でフッターを永続的にするには、関連する全てのページの**persistentFooterId**にid値を割り当て、各ページに同じid値を使います。例えば、**persistentFooterId**を現在のページとターゲットのページの"myfooter"に設定すると、フレームワークはページ・アニメーションの間、フッター・アンカーを同じ場所に保ちます。この効果は、フッターがposition="fixed"に設定された時のみ正しく機能し、遷移の間も表示されています。

省略値

空白 - フッターは永続的ではありません。

有効値

文字列値。

showFooter

Trueの場合、レイアウトはレイアウトの下部にjQuery Mobileフッターを追加します。詳細は[jQuery Mobile](#)ドキュメントの「Toolbars」のセクションを参照してください。

フッターバーはカスタマイズ可能なコンテンツ・エリアです。フッターバーをカスタマイズするには、フッター上で右クリックし、「content.footerのコンテンツ・エリア」メニューから選択肢を選んでください。

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

showHeader

Trueの場合、レイアウトはレイアウトの上部にjQuery mobile headerを追加します。詳細は[jQuery Mobile](#)ドキュメントの「Toolbars」のセクションを参照してください。

ヘッダーバーはカスタマイズ可能なコンテンツ・エリアです。ヘッダーバーをカスタマイズするには、ヘッダー上で右クリックし、「content.headerのコンテンツ・エリア」メニューから選択肢を選んでください。

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

showMessages

Trueの場合、メッセージ・ウェブレットはコンテンツ・エリアの上部に置かれ、ウェブルーチンからのメッセージを表示します。

省略値

True。

有効値

True、False、もしくはブール値を返す有効なXPath式。

sidebarPositionSmallScreen

サイドバーをメイン・コンテンツの横に表示するには小さすぎる画面（例：電話の画面）にどうやってサイドバーを配置するかをレイアウトに教えます。

省略値

Top。

有効値

top, bottomもしくはhidden。

validationErrorDisplay

検証エラーをどのように表示するかを指定します。詳細については、[「フィールド検証」](#)を参照してください。

省略値

空白 - ブラウザの省略値メカニズムを使います。

有効値

'Browser Default' もしくは 'Error Div'

validationTime

検証エラーをいつ表示するかを指定します。詳細については、「[フィールド検証](#)」を参照してください。

省略値

After Submit - ユーザーが送信ボタンを押すと検証が実行されます。

有効値

Immediately, After Focus もしくは After Submit。

windowTitle

ページタイトルを指定します。このプロパティの値を使ってページの<タイトル>タグを指定します。

省略値

`$lweb_context/lxml:webroutine-title` - これはウェブラーチンの説明を参照するXPath式です。

有効値

文字列値。

9.6 ユーティリティ・ウェブレット

ウェブレット名 記述

標準非表示
フィールド
(std_hidden) LANSA製品センター内部使用専用です。詳細については、「[非表示](#)」を参照してください。

標準地域固有
(std_locale) LANSA製品センター内部使用専用です。詳細については、「[ローカル変数](#)」を参照してください。

標準スクリプト
(std_script) LANSA製品センター内部使用専用です。詳細については、「[JavaScriptとスクリプト・ウェブレット](#)」を参照してください。

ページ・スクリ
プトの最後
(scriptAtEnd) LANSA製品センター内部使用専用です。詳細については、「[JavaScriptとスクリプト・ウェブレット](#)」を参照してください。

標準変数
(std_variables) LANSA製品センター内部使用専用です。詳細については、「[変数](#)」を参照してください。

std_keys XSLのフィールド参照で使用するxsl:keyエレメントのセット詳細については、「[キー](#)」を参照してください。

std_types LANSA製品センター内部使用専用です。詳細については、「[タイプ](#)」を参照してください。

std_util LANSA製品センター内部使用専用です。

CSS スタイル
(std_style_v2) 生成したすべてのWEBROUTINEページに使用する省略値のスタイルです。

WAM チュートリアル (英語)

If you haven't read the rest of the *Web Application Modules Guide*, below in [What is a WAM?](#) you will find an outline of the things you need to understand before you can obtain any benefit from the following exercises.

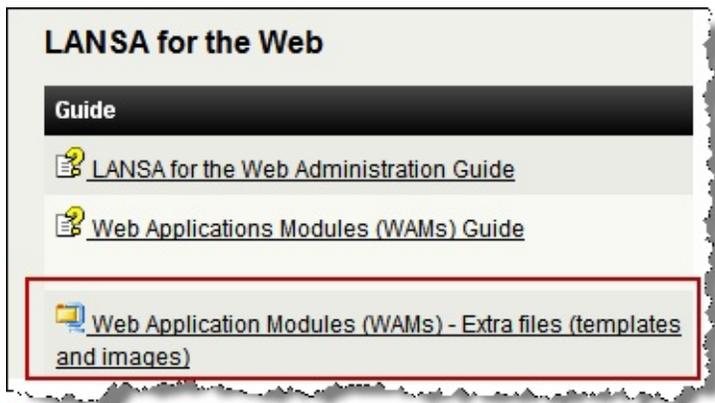
The exercises provided are:

- [WAM005 - Create Your First WAM](#)
- [WAM010 - Using WEB_MAPs](#)
- [WAM015 - Working Lists](#)
- [WAM020 - WAM Navigation](#)
- [WAM025 - Using the Layout Wizard](#)
- [WAM030 - Employee Enquiry](#)
- [WAM035 - An Employee Update WAM](#)
- [WAM040 - Add dropdown lists for Department and Section](#)
- [WAM045 - A Dynamic Selector Dropdown list using a Select Field](#)
- [WAM050 - A Section Maintenance Application](#)
- [WAM055 - Using LANSA Debug](#)
- [WAM060 - Employee Maintenance using Advanced Weblets](#)
- [WAM065 - Controlling List Output](#)
- [WAM070 - Hiding Techniques](#)
- [WAM075 - Using a Tree View Weblet](#)
- [WAM080 - Session Management](#)
- [WAM085 - Enhancing the User Interface](#)
- [WAM090 - Using a List Row Weblet](#)
- [WAM095 - LOB Data Types and Stream Files](#)
- [WAM100 - Using Cascading Style Sheets](#)
- [WAM105 - Create Your Own Weblet](#)
- [WAM105 - Create Your Own Weblet](#)
- [WAM110 - Create Your Own Layout Weblet](#)
- [WAM115 - Check in WAMs to IBM i](#)
- [WAM120 - Using the Menu Bar Weblet](#)
- [WAM125 - Define a Dynamic Menu](#)

- [WAM130 - Output a Web Page to a File](#)
- [WAM135 - Using the Google Static Maps API](#)

Before You Begin

Some extra files are needed to complete these tutorials. Where these files are required, they are listed at the beginning of the tutorial. Before you begin, you can download the extra files from the documentation page of the LANSA web site. Go to <http://www.lansa.com/support/docs/index.htm>>Documentation - Current Version and then look down the list for the Extra files in the Web Guide section as shown here:



The zip file also includes a readme.txt file with instructions for what you need to do with these files but these instructions are also included at the beginning of the exercise in which you will use them.

What is a WAM?

A WAM is a component that supports the web paradigm. A WAM outputs XML containing the fields and lists that are to be passed into its output web page. This web page is created via an XSLT transformation. The XSL reads the output XML and generates the XHTML. A web server associated with your web application then forwards this XHTML to the user's browser.

The XSLT transformation uses other LANSAXSL components, known as weblets. These weblets provide web components, that can be added into a page and configured. They have properties which can be set both at design time and run time as required. Weblets include relatively simple components such as a push button or clickable image as well as much more complex components such as a grid, or tree view.

The web page layout is controlled by a layout weblet. A layout weblet is assigned when the WAM is compiled and may be modified both at design time and run time.

The WAM Design view provides a graphical editor that enables the developer to define the appearance and behavior of the web page that is output for a web routine. A weblet such as a push button can be dropped onto the page and then set up via its properties using the Details tab.

A WAM usually contains a number of web routines. Each web routine is a program entry point – it can be invoked via link on a web page for example. When a web routine ends, it outputs its fields and lists as an XML document.

Mapping of data into and out of a web routine is controlled by WEB_MAP statements. A WEB_MAP defines whether fields are mapped into or out of the web routine, or in both directions, via a FOR() parameter. Fields and lists are mapped via the WEB_MAP FIELDS() parameter. Fields and list may have display attributes that control whether the field is input capable (the default) or output only. Other field attributes such as *private and *hidden, may be used.

Some Example WEB_MAP Statements

```
WEB_MAP FOR(*OUTPUT) FIELDS(#EMPNO)
```

Field EMPNO is mapped out of this Web Routine. EMPNO is input capable on the page.

```
WEB_MAP FOR(*INPUT) FIELDS(#EMPNO)
```

Field EMPNO is mapped into this Web Routine. EMPNO is not mapped out of this Web Routine to the page.

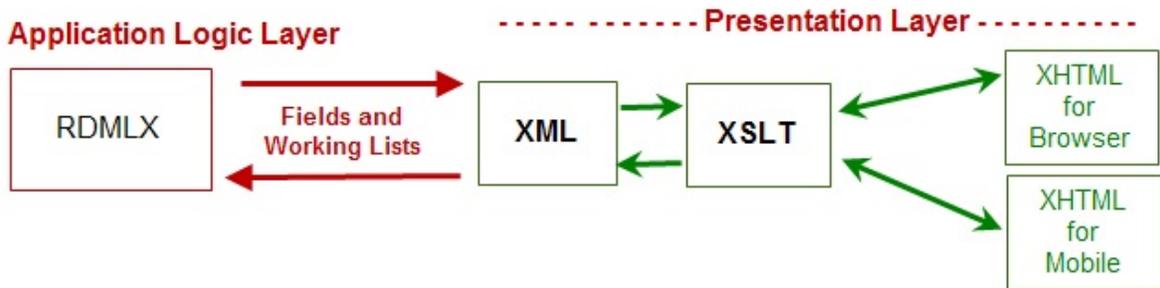
```
WEB_MAP FOR(*BOTH) FIELDS((#EMPNO *OUTPUT)
#SURNAME #GIVENAME)
```

Fields are mapped into and out of this Web Routine. Field EMPNO is output on the page and therefore cannot be mapped into the next Web Routine.

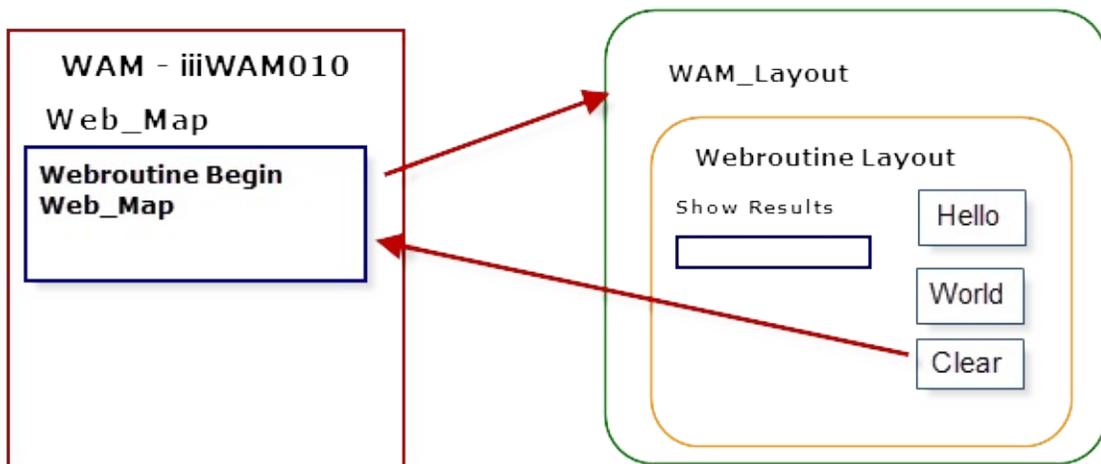
```
WEB_MAP FOR(*BOTH) FIELDS((#STDRENTY *HIDDEN))
```

Field STDRENTY is mapped into and out of this Web Routine. STDRENTY is a hidden field defined in the XHTML but not shown on the web page.

A WAM application should be considered as having two layers:



WAM Architecture



- A WAM contains one or more WebRoutines
- A WebRoutine usually outputs a web page
- Web_Maps define fields and lists that may be mapped into and out of the WebRoutine

- Web_Maps defined at WAM level apply to all WebRoutines
- A WAM adopts a standard layout
- All WebRoutine layouts share the `wam_layout` (initially)
- The WebRoutine web page contains fields and lists that are mapped for *output or *both
- Developer completes WebRoutine page design using a graphical editor (the Design View)
- Weblets provide XSL that add web components such as push buttons to the web page
- Weblets are set up (programmed) using a property sheet on the Details tab.

Stateless Programming

One of the key points to understand about WAMs is that they are stateless. In fact, any internet-based application is stateless. What this means is that when a WAM is executed from the Presentation Layer, it runs (a job is initiated on the server), produces some output (a web page), and then ends (the job on the server ends and control is transferred back to the browser).

The job starting and ending, to all intents and purposes, is a "transaction". Any data that needs to be maintained for the user's web "session", i.e. span multiple transactions, must be kept somewhere. As you complete the following simple example WAM, you will begin to see how a web application needs to be designed to handle this "stateless" model.

Using Long Names

LANSA V13 enables components to be defined using *Long Names*. This is an optional setting at *Partition* level. When you create a WAM, form or a reusable part using a Long Name, Visual LANSAs will assign a unique *Identifier* (this is up to 10 characters in length). As you are creating a new component, you may choose to assign your own *Identifier*, which provides complete control to a team of developers, of both Long Names and Identifiers.

See the topic *LANSA Object Names* in the *Technical Reference* guide for further information.

This workshop assumes that Long Names are enabled.

WAM005 - Create Your First WAM

Objectives

- To create your first Web Application Module (WAM).
- To create a WebRoutine in your Web Application Module.
- To understand the basics of how the WEB_MAP command is used within a WebRoutine (covered in much more detail in [WAM010 - Using WEB_MAPs](#)).
- To compile a Web application that was created using a WAM.
- To understand the compile/generate cycle of WAM development.

To achieve these objectives, you will complete the following ten simple steps:

- [Step 1. Start Visual LANSA](#)
- [Step 2. Create a WAM](#)
- [Step 3. Create the ReentryTest WebRoutine](#)
- [Step 4. Compile the WAM](#)
- [Step 5. Open the Design view](#)
- [Step 6. Editing](#)
- [Step 7. Use a Weblet](#)
- [Step 8. Make STDREENTRY visible for testing](#)
- [Step 9. Test the WAM](#)
- [Step 10. Hide STDREENTRY](#)
- [Summary](#)

Before you Begin

In order to complete this exercise, the demonstration Personnel System must have been installed in this partition.

The purpose of this Exercise is to create your first WAM and become familiar with the Design view. You will also be given an introduction to Weblets and re-entrant programming.

In this exercise, you will create a WebRoutine that will use different weblets to call itself. It will display a different message depending on which weblet was used.

Step 1. Start Visual LANSA

WAMs are created using Visual LANSA and may be deployed to a variety of servers such as Windows and IBM i. In order to create a WAM:

- You must use an RDMLX Enabled Development partition.
- Field Types such as Date and Time must also be enabled in the partition.
- The partition must be web enabled.
- You must have a valid LANSA for the Web license to develop with WAMs.

See the *LANSA for iSeries User Guide* for further details. See also the *Web Housekeeping* and *Introduction to LANSA for the Web* guides.

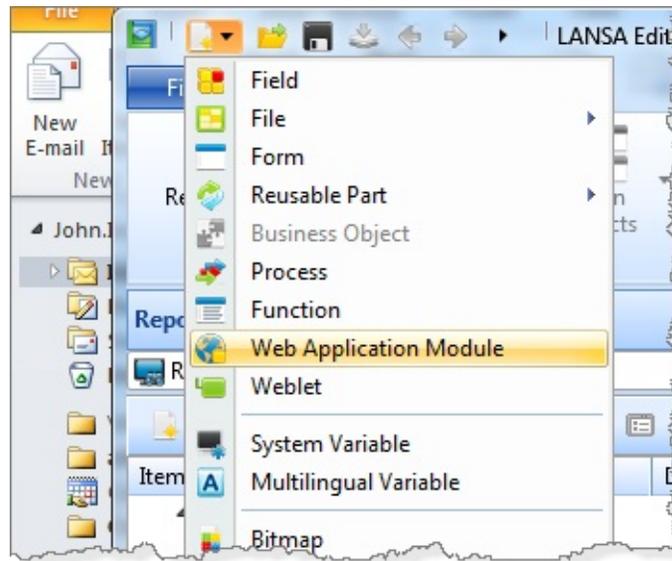
In this step, you will log on to Visual LANSA.

1. Start the Visual LANSA Development Environment and log on to the DEM partition (recommended) or any other partition being used for training. The partition must contain the Personnel System files.

Step 2. Create a WAM

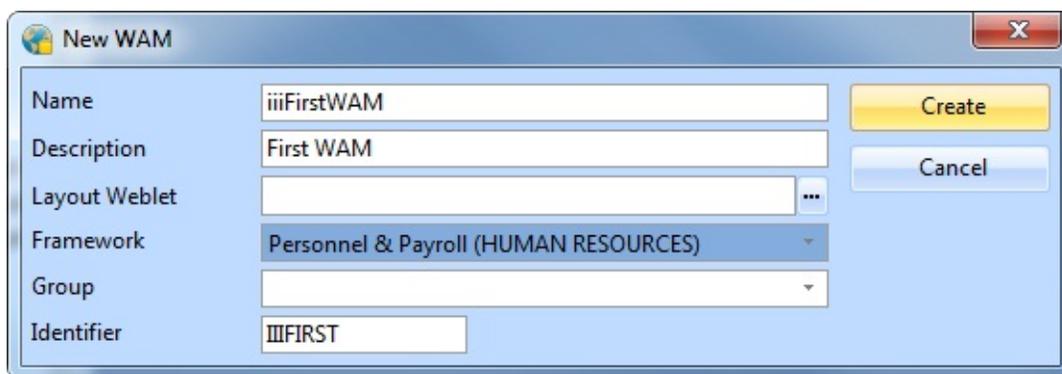
In this step, you will create a Web Application Module that will eventually contain the RDMLX code for your First WAM.

1. In the LANSAs Editor window, select *Web Application Module* from the main toolbar *New* button.



Alternatively select *Web Application Module* from the *New* button on the *Repository* or *Favorites* tab.

The *New WAM* dialog will appear:

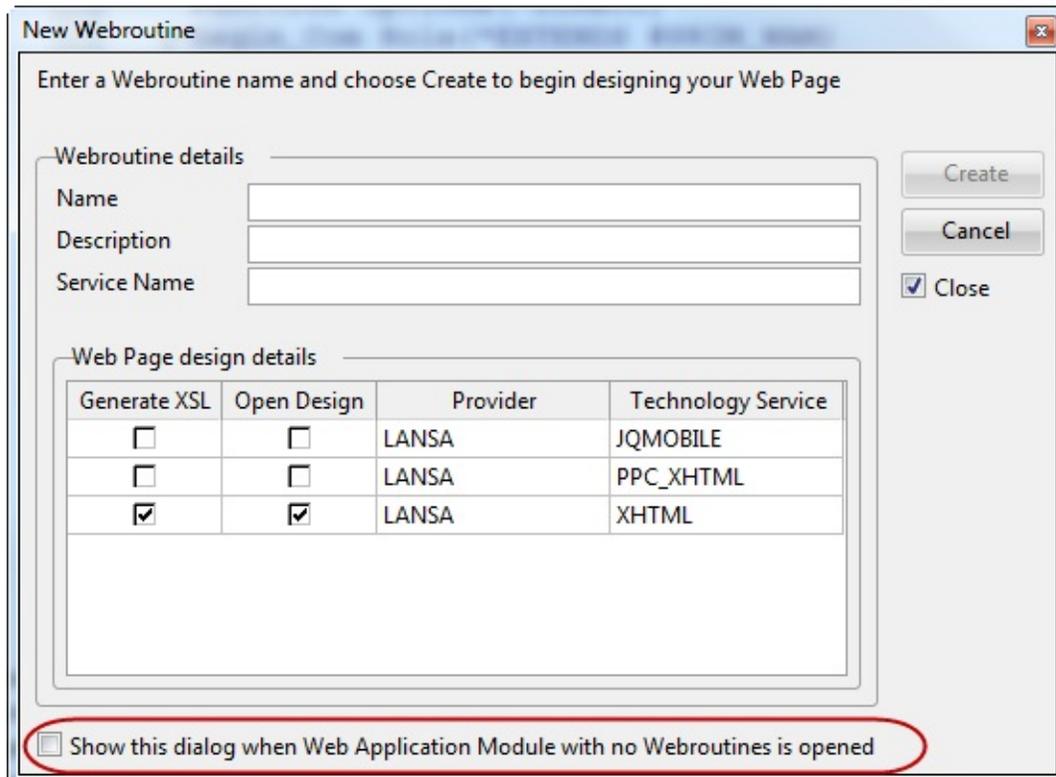
A screenshot of the 'New WAM' dialog box. It has a title bar with a globe icon and the text 'New WAM'. The dialog contains several fields: 'Name' with the value 'iiiFirstWAM', 'Description' with the value 'First WAM', 'Layout Weblet' which is empty, 'Framework' with a dropdown menu showing 'Personnel & Payroll (HUMAN RESOURCES)', 'Group' which is empty, and 'Identifier' with the value 'IIIIFIRST'. There are 'Create' and 'Cancel' buttons on the right side.

2. In the *New WAM* dialog box:
 - a. Enter a *Name* of **iiiFirstWAM** (where **iii** are your initials).
 - b. Enter a *Description* of **My First WAM**.
 - c. Leave the *Layout Weblet* field blank.

- d. Select a Framework Personnel & Payroll.
- e. Click the *Create* button to create the new WAM.

Note:

- When you leave the Layout Weblet field blank, your WAM will automatically generate a wam layout based on Theme Layout #1 – One Column (std_themelet1_1col.xsl). You will create and use your own layout using a wizard, in a later exercise.
 - Frameworks allow related components to be grouped together. The framework assigned to a component may be changed at any time. The Personnel & Payroll Framework has been selected for this new WAM. The new WAM will be given a ComponentFramework property HUMAN RESOURCES-Personnel.
3. The *New WebRoutine* dialog will appear. Deselect the *Show this dialog . . .* option and select *Cancel*. You will create your WebRoutines manually in the editor. You will use this dialog in a later exercise.



4. The LANSAs Editor will now display the WAM's RDMLX code. At this stage, it will not contain any WebRoutines.

Step 3. Create the ReentryTest WebRoutine

In this step, you will use the LANSAs Editor to create the code for a new WebRoutine that will give you an understanding of Weblets and re-entrant programming. You will use a WEB_MAP statement to specify the fields that are passed in and out of the WebRoutine. Your initial code should look like the following:

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_WAM)
```

```
End_Com
```

1. Immediately following the BEGIN_COM, insert the following RDMLX code to create a WebRoutine named *ReentryTest*:

```
WebRoutineebroutine Name(ReentryTest) Desc('Test WAM')
Endroutine
```

2. This WebRoutine will require three fields that are both incoming and outgoing.
 - a. STDREENTRY will be used to control the functionality of the WebRoutine each time it is called.
 - b. GIVENAME and SURNAME are to demonstrate the different display modes for fields on the page.
 - c. Add the following WEB_MAP statement to the WebRoutine:

```
Web_Map For(*BOTH) Fields((#Givename *input) (#Surname *output)
(#Stdreentry *hidden))
```

- The **STDREENTRY** field will be used to identify the execution state.
 - The **STDREENTRY** field is qualified as hidden so as not to be visible on the page.
 - The **GIVENAME** field will accept input.
 - The **SURNAME** field will be output only.
3. Set up a **CASE loop using** the field **STDREENTRY** to display different messages.

```
Case Of_Field(#Stdreentry)
```

```

When Value_Is(= A)
Message Msgtxt('Push Button was pressed')
When Value_Is(= B)
Message Msgtxt('Hyperlink was clicked')
When Value_Is(= C)
Message Msgtxt('Check Box was changed')
When Value_Is(= D)
Message Msgtxt('Radio Button was selected')
When Value_Is(= E)
Message Msgtxt('Dropdown was changed')
Otherwise
Message Msgtxt('Page loaded normally')
Endcase

```

4. Set **GIVENAME** and **SURNAME** to your name.

```

#Givename := 'your first name'
#Surname := 'your last name'

```

Your finished WebRoutine should appear as follows:

```

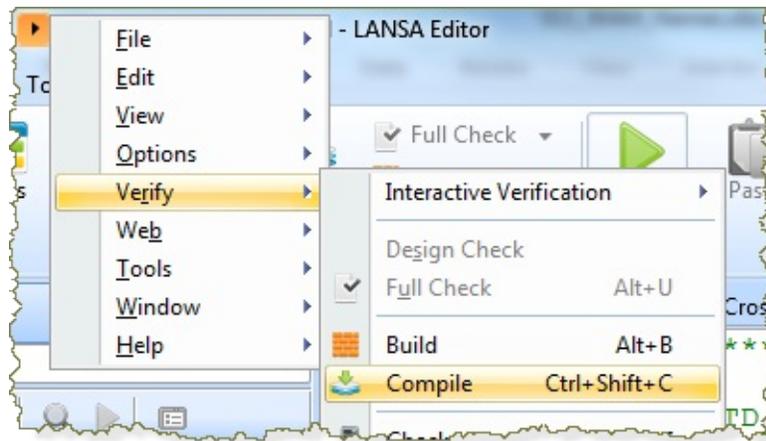
Webroutine Name(ReentryTest) Desc('Test WAM')
Web_Map For(*BOTH) Fields((#Givename *input) (#Surname *output)
(#Stdreentry *hidden))
Case Of_Field(#Stdreentry)
When Value_Is(= A)
Message Msgtxt('Push Button was pressed')
When Value_Is(= B)
Message Msgtxt('Hyperlink was clicked')
When Value_Is(= C)
Message Msgtxt('Check Box was changed')
When Value_Is(= D)
Message Msgtxt('Radio Button was selected')
When Value_Is(= E)
Message Msgtxt('Dropdown was changed')
Otherwise
Message Msgtxt('Page loaded normally')
Endcase
#Givename := 'your first name'
#Surname := 'your last name'
Endroutine

```

Step 4. Compile the WAM

In this step, you will learn how to compile the WAM.

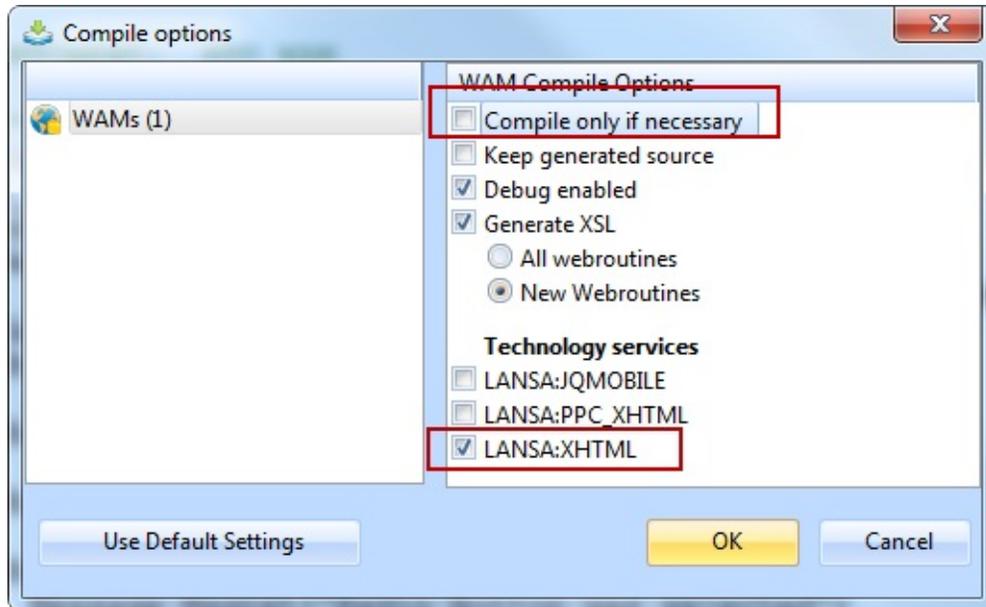
1. You should still have the **iiiFirstWAM** (where **iii** are your initials) WAM open in the LANSAs Editor.
2. Compile your first WAM using the *Menus* button on the main toolbar. Select the *Compile* option:



3. On the *Compile options* dialog, ensure that only the *Technology Services, LANSA:XHTML* is selected. These tutorials will be generating web applications for the browser only.

Un-select the *Compile only if necessary* option. The *Compile* button will now always compile the WAM.

These options will be retained for future compiles.



The compile will generate a browser web page design (XSL) for you.

The generator performed the following steps:

- Generated your WAM-specific layout Weblet, called **iiiFirstWAM_layout**. You can see this in the Repository under Weblets in the Web View.
- Generated the WebRoutine web page design using the layout **iiiFirstWAM_layout**.
- Placed the WebRoutine description as a heading on the web page.
- Included a messages Weblet on the page.
- Placed all fields mapped FOR(*OUTPUT) or FOR(*BOTH)) and their captions in a table on the web page.
- Inserted a hidden DIV at the bottom of the page. In the *Design* view this has the title "Hidden Content" shown. You can add anything that you want to be hidden here and at run time, it will not be displayed on the page regardless of its display mode.

Step 5. Open the Design view

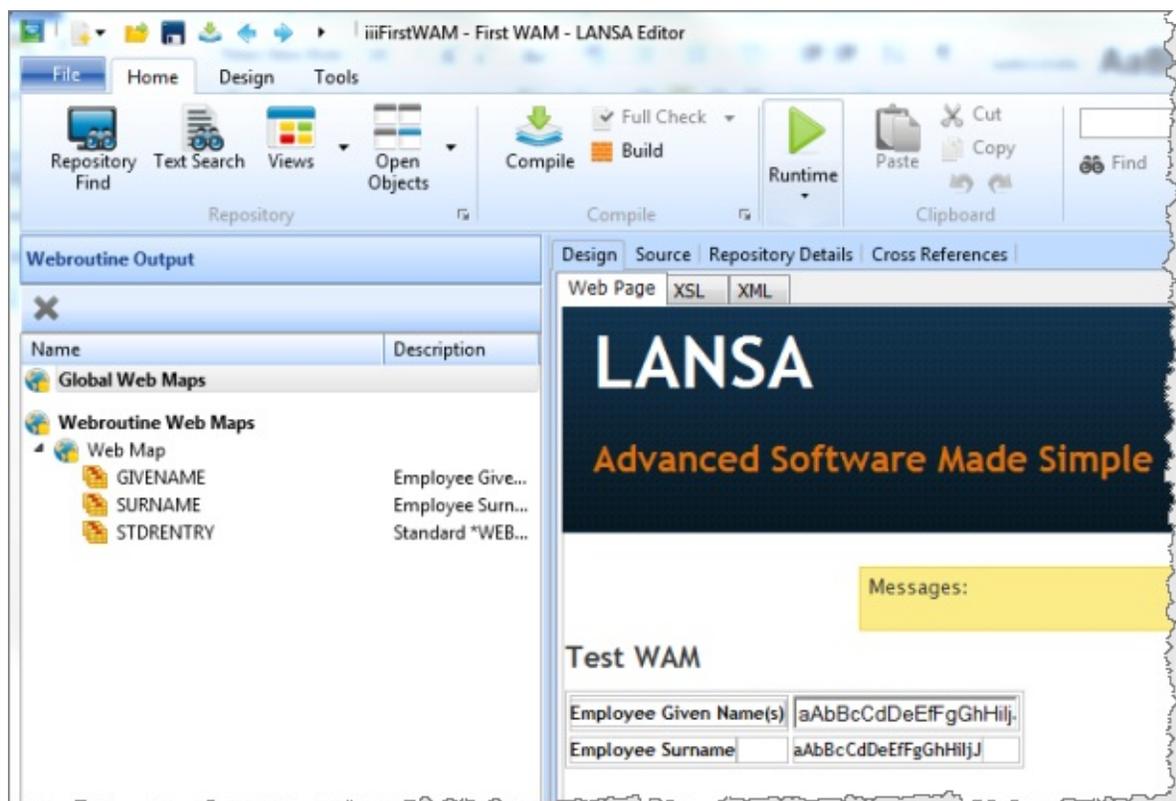
After creating the WebRoutine RDMLX within the LANSAs Editor, you can edit the web page design for your WebRoutines. In this step, you will select the *ReentryTest* WebRoutine created in Step 1 and open the Design view.

1. After the compile has completed successfully, open the *ReentryTest* WebRoutine in the Design view.
 - a. Click the  Open Design button to open WebRoutine *ReentryTest* in the Design view.

```
FUNCTION OPTIONS(*DIRECT)
  Begin_Com Role(*EXTENDS #PRIM_WAM)
  WEBROUTINE NAME(ReentryTest) DESC('Test WAM') 
  WEB_MAP FOR(*BOTH)
    FIELDS((#GIVENAME *INPUT) (#SURNAME *OUTPUT) (#STDRENTRY))
  CASE OF_FIELD(#STDRENTRY)
    WHEN VALUE IS(= A)
      ...
```

- b. The *Design view* will open for the selected WebRoutine.

The Design view should appear something like the following:



Once a WebRoutine has been compiled, a web page design with a WAM-

specific layout (that you can modify) and field entry controls are generated and can be edited in the *Design* view.

Note: Because you now have a layout Weblet that is only used by the **iiiFirstWAM** WAM, you can open it and edit it without affecting the layout of any other WAMs.

The left hand pane includes the following Views:

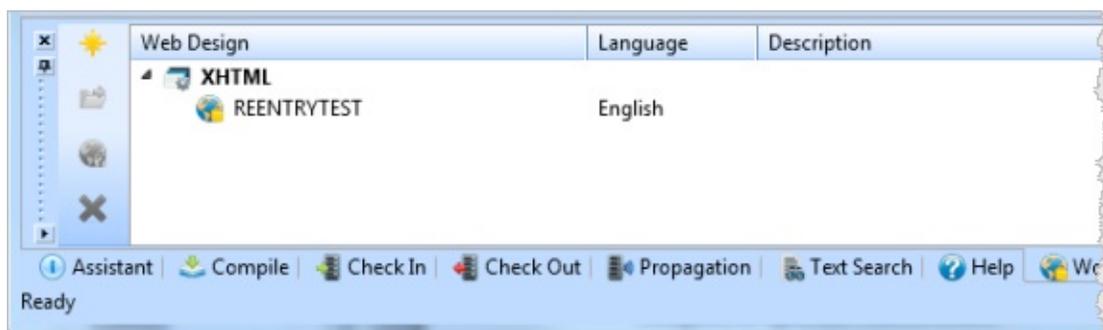
-  **Outline** - shows fields and lists, as well as other elements contained in the XSL page currently being edited.
-  **Details** - allows the editing of attributes of selected XSL and HTML elements, as well as Weblet parameter values.
-  **WebRoutine Output** - shows all the fields and lists available for drag and drop onto the web page. These are all the fields and lists specified on the WebRoutine's WEB_MAP FOR(*OUTPUT) or WEB_MAP FOR(*BOTH) statements in RDML.

The right hand Design tab contains the following Views:

- **Web Page** - allows WYSIWYG editing of the web page.
- **XSL** - allows text editing of the XSL source.
- **XML** - allows text editing of the input XML source.

The bottom pane contains the following View:

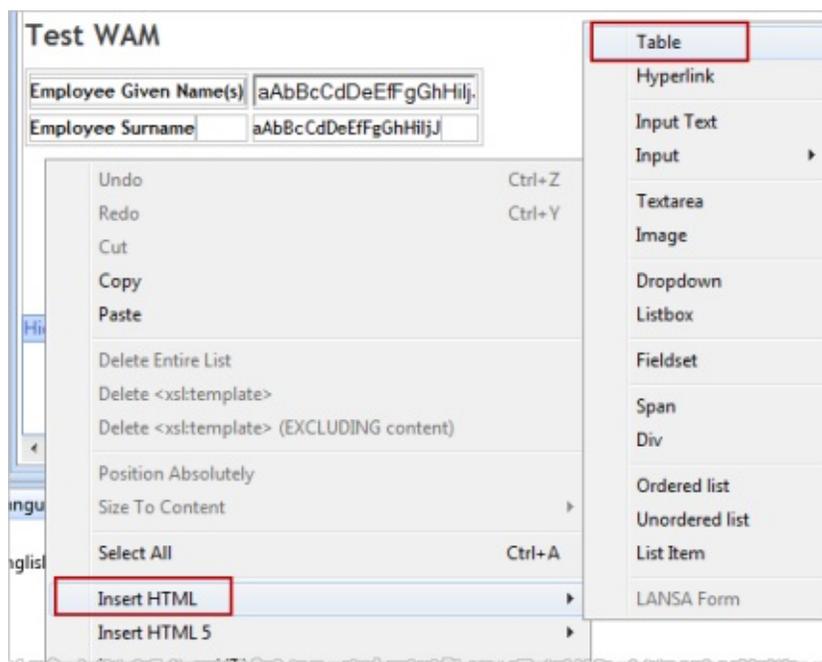
-  **Web Designs** - lists all WebRoutines for the current WAM open in the editor. Every WebRoutine on the tab represents a web page design using certain technology in certain a Language.



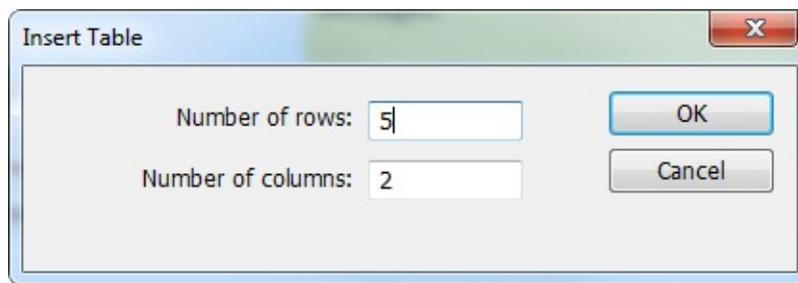
Step 6. Editing

In this step, you will perform some simple editing tasks in the Design view and preview your changes in order to become familiar with the Editor. You will add a table, add text and Weblets to the table and set the Weblet's parameters.

1. Make sure you are in the *Design View* by clicking on the *Design* tab on the right hand panel of the *Design* view. To insert the table into the *Design View*:
 - a. Right-click the mouse in the middle of the page, under the table that was generated from the compile, to open the context menu.

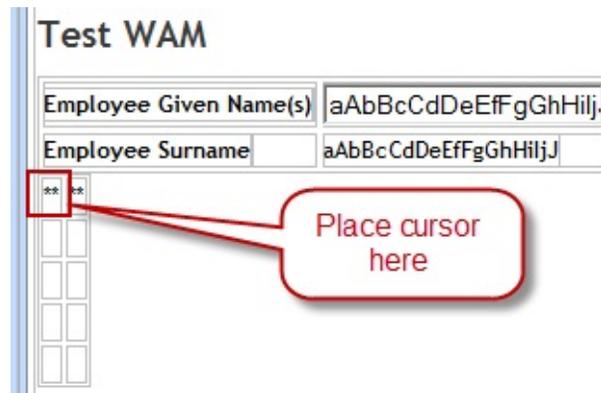


- b. Select *Insert HTML* from the context menu and choose the *Table* sub menu item.
- c. Create a table with 5 rows and 2 columns in the *Insert Table* dialog box. Press *OK*.



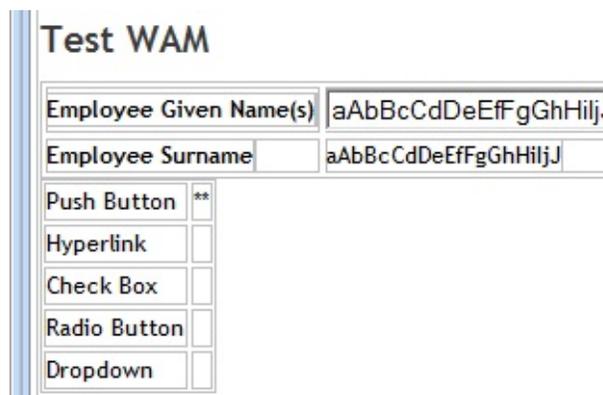
- You will now see an HTML table in the *Design View*.

- Notice that the top row contains asterisks (that is "***"). These characters are inserted so that you can easily access the table cells. Without the asterisks the table cells would have no visible width. After editing the table, you can simply delete the asterisks.
2. Place the cursor into the top, leftmost cell of the new table in the *Design View*.



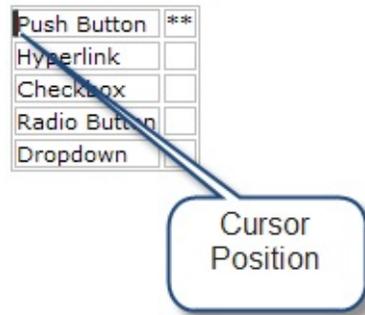
3. Type the text "Push Button" into the first cell of the table. This column will be used as labels for the weblets that will be added to the second column.
4. Now that this column has an entry you can delete the placeholder "***" characters from the first table cell by placing the cursor next to them and using the *Delete* button..
5. Insert the names of the other Weblet labels in the four remaining blank rows of the left column: "Hyperlink", "Check Box", "Radio Group" and "Dropdown". There is no need to include the quotation marks.

Your *Design View* will now appear something like this:



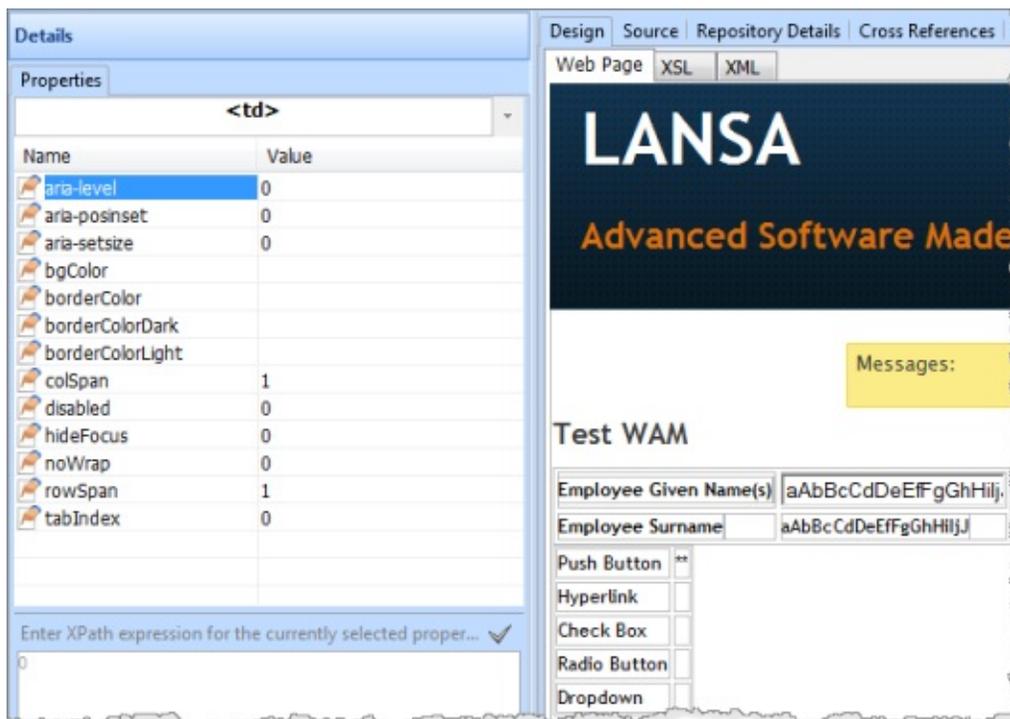
6. In the *Design View*, select the top left cell again by clicking in it. (That is, the table cell containing the text "Push Button".) There should be no "Grips"

around anything at this point and the cursor should be blinking inside the cell.

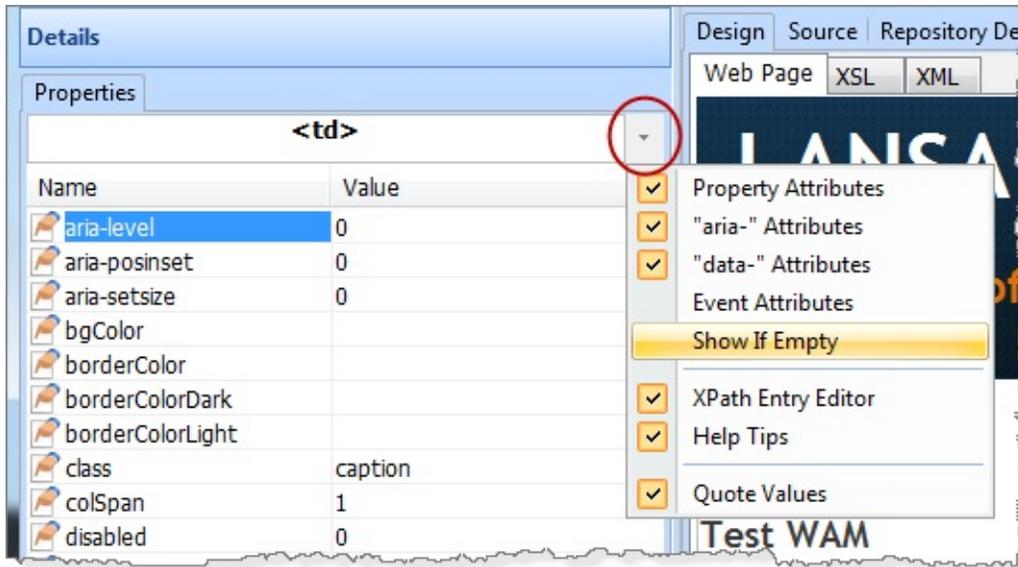


7. Select the *Details View*  in the left pane.

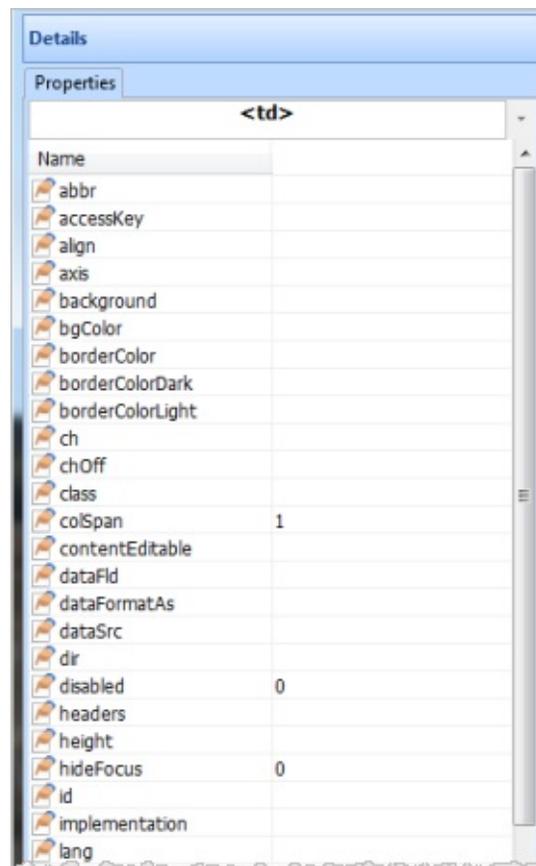
You will now see all the properties of the table cell:



8. Click the *Menu* button on the Details tab and select the *Show If Empty* option. Unselect the "aria-" Attributes.

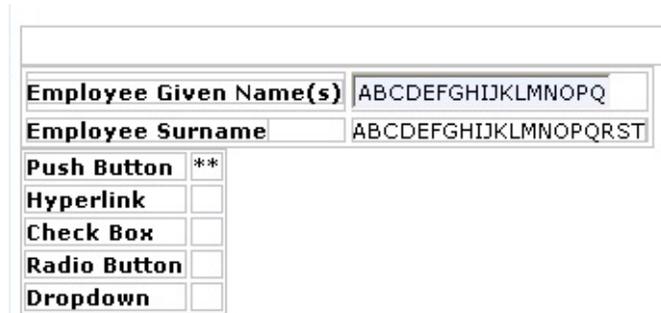


The *Details* tab should now look like the following:

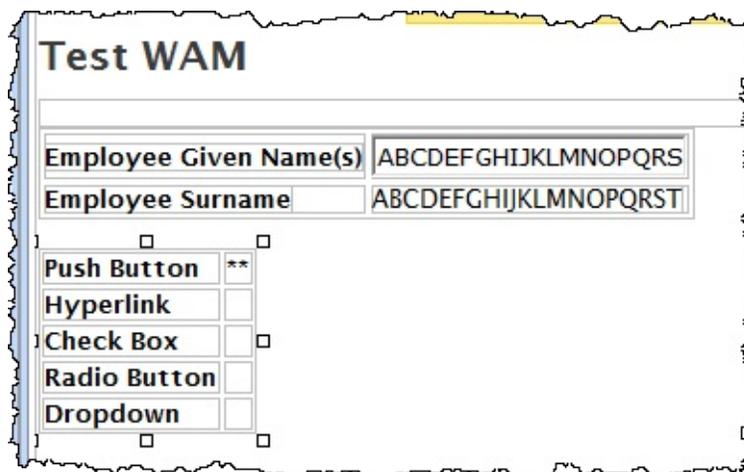


The *Show If Empty* option has displayed properties for the cell which currently do not have a value.

9. In the *Details View*, locate the *class* property.
 - a. Click in the *class value* column and use the dropdown list to set the property to **bold** and press *Enter*.
 The text will immediately become bold. What you have done is applied one of the caption styles provided with the Cascading Style Sheet (CSS) shipped with the LANSA product. The default style is not bold.
 - b. Set the *class* property to **bold** for the remaining cells in the left column.
 The *Design* view should appear something like the following:



10. Click the  Save button to save your changes to *ReentryTest*.
11. Select the new table by clicking on one of its corners:

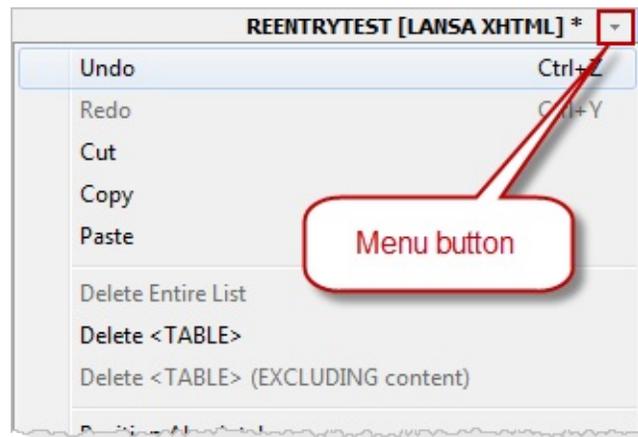


A selected web page element is displayed with "handles" as shown above.

12. Scroll down to the *style* property. Style is a composite property consisting of other properties.
13. Click on the icon to the left of the property to expand the style's individual properties. You can modify these properties directly.
14. Scroll down to style's *border-style* property. Select **outset** from the

dropdown list. Click on another property and you should immediately see the border of the cell change to an outset appearance.

15. Experiment by changing other properties, such as the background color, or inserting other HTML elements into the page. Using styles in this way sets an inline style for the HTML element that you are editing. You will usually want to control styles via a Cascading Style Sheet (CSS). Refer to exercise [WAM100 - Using Cascading Style Sheets](#) for more information about CSSs.
16. Click on the *Menu* button on the top right of the web page design and use the *Undo* option to remove the changes made in 14.



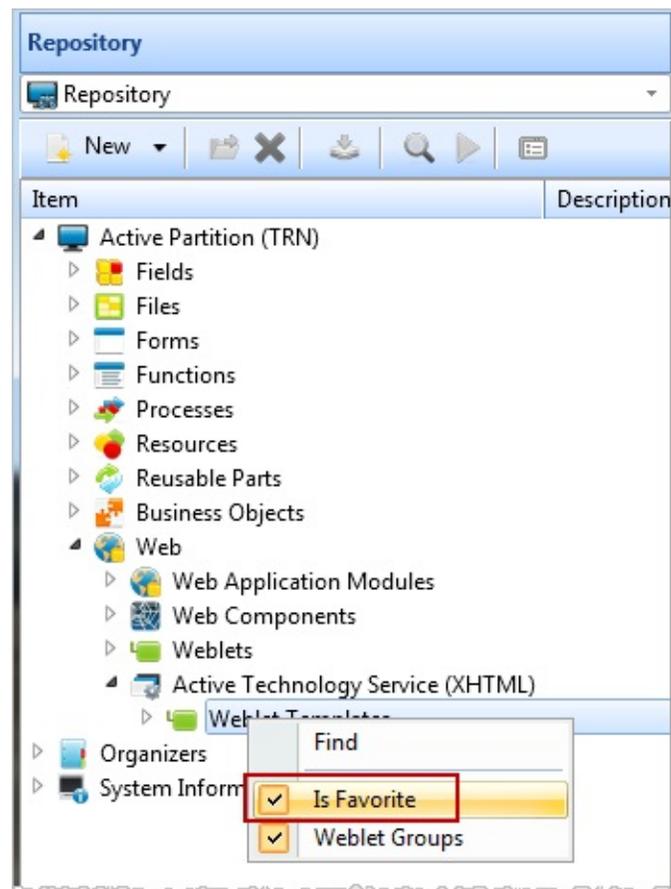
Alternatively, you could also have used the **Ctrl+Z** keys to undo.

Step 7. Use a Weblet

In this step, you will learn how to drag and drop Weblets onto the web page. The Weblets will each be configured to call the WebRoutine with a different reentry value.

1. With *ReentryTest* open in the Design view.
2. Select the Favorites/*Weblet Templates* tab in the left pane to see a list of all the Weblets in your repository. (Make sure that Standard Weblets is specified in the dropdown list so that all standard Weblets are listed.)

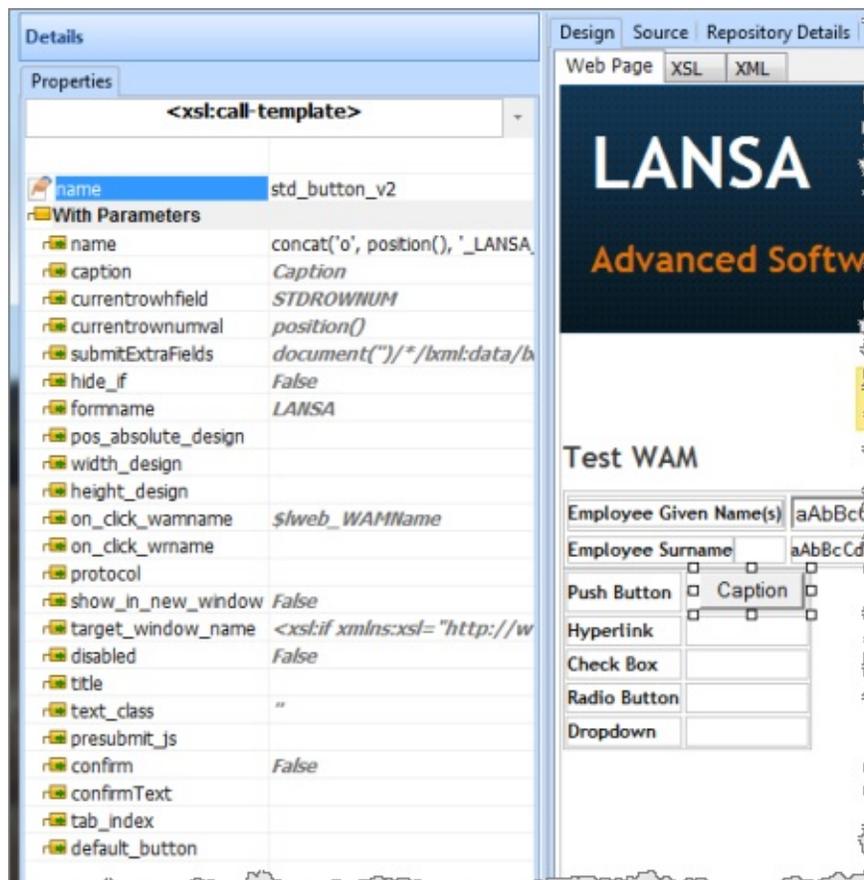
Hint: If *Weblet Templates* is not shown on your *Favorites* tab, select the *Repository* tab and expand *Web / Active Technology* and then use the right mouse menu to make *Weblet Templates* a Favorite.



3. Select the *Push button* Weblet from the list and drag and drop it into the cell that is to the right of the "Push Button" caption in the table in *Design View*.
4. Select the *Details View* in the left pane to display the Weblet's properties.

The italicized and gray values indicate default values.

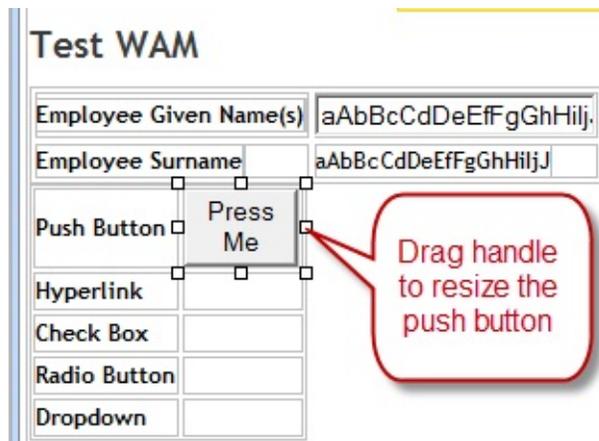
In the *Design View*, the button should be selected.



- a. In the *Details View*, click in the value column of the *caption* property and enter **Press me**.

The button's caption will immediately change to **Press me**.

If necessary, drag the right handle on the push button so that the caption is displayed as a single line.



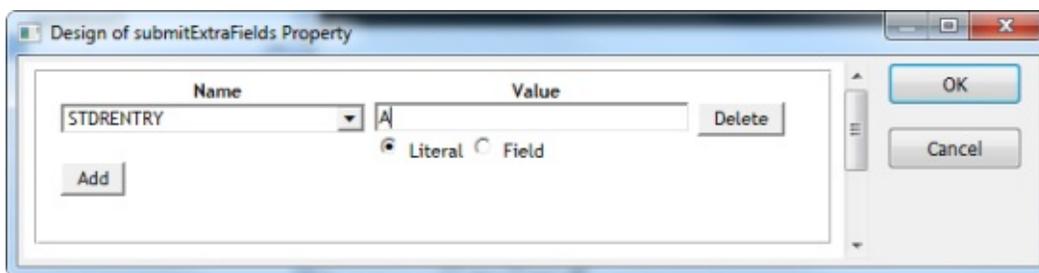
- b. Set the `on_click_wname` property to **ReentryTest**. In the value column, select this value from the dropdown.

Note:

You should always use the dropdown list to select this property rather than type it in, as spelling mistakes or typing errors are one of the biggest causes of problems at design time.

Always complete the `on_click_wname` property before the next step. The *Design of...* dialog uses the WebRoutine defined in the `on_click_wname` property to populate the list of mapped fields in the *Fields* dropdown list.

- c. Click inside the `submitExtraFields` value column and then click the *Ellipsis* button  to open the *Design of* dialog.



- d. Select the field `STDREENTRY` in the left hand *Name* column.
- e. Enter a value of **A**, in the *Value* column, leaving the *Literal* checkbox selected.
- f. Click *OK* to confirm your changes and close this dialog.

Note: The fields shown in the dropdown for the *Name* column in the *Design of submitExtraFields Property* dialog are the fields mapped for `*INPUT` or `*BOTH` for the WebRoutine which is defined as the `on_click_wname` property for this weblet.

In *Value*, if you specify a:

- Literal value, you are returning a fixed, hard coded value for this field when this button is clicked.
 - Field, you are returning the field value when this button is clicked.
5. In the *Design View*, click on the table cell where the Push Button weblet was dropped. Again, make sure that you select the cell and not the button. Delete the asterisks in the cell by placing the cursor in the cell next to them and using the backspace key.
Hint: With the push button selected you can use the *cursor keys* to move left or right into the table cell.
 6. In the *Details View*, set the *align* property of the cell to **right**.
 7. Now add the remaining Weblets by selecting them in the *Weblets Templates View* and dragging them into the appropriate right hand cell based on the left hand cell label. The Weblets you will use are the Anchor, Check Box, Radio Group and Combo Box.
 8. All of the remaining weblets have default of STDREENTRY for their *reentryfield* property. Set the *reentryvalue* property for each of these Weblets. Check the RDMLX to see what value should be used for each weblet. For example:

When Value_Is(= B)
Message Msgtxt("Hyperlink was clicked.")
 9. Set the *on_click_wrname* property for the anchor, check box and radio group to **ReentryTest**.
 10. Set the *on_change_wrname* property for the combo box to ReentryTest.

Notice that you are not required to change the *on_click_wamname* or *on_change_wamname*. By default these values will be the WAM name that initiated the web page.
 11. Set the *value* property for the anchor to **Click Me**.
 12. Set the *caption* property for the check box to **Click Me**.
 13. Set the *align* property for all of the cells containing weblets to **right**.

Your page should now appear as follows:

Messages:

Test WAM

Employee Given Name(s)

Employee Surname

Push Button

Hyperlink [Click Me](#)

Check Box Press Me

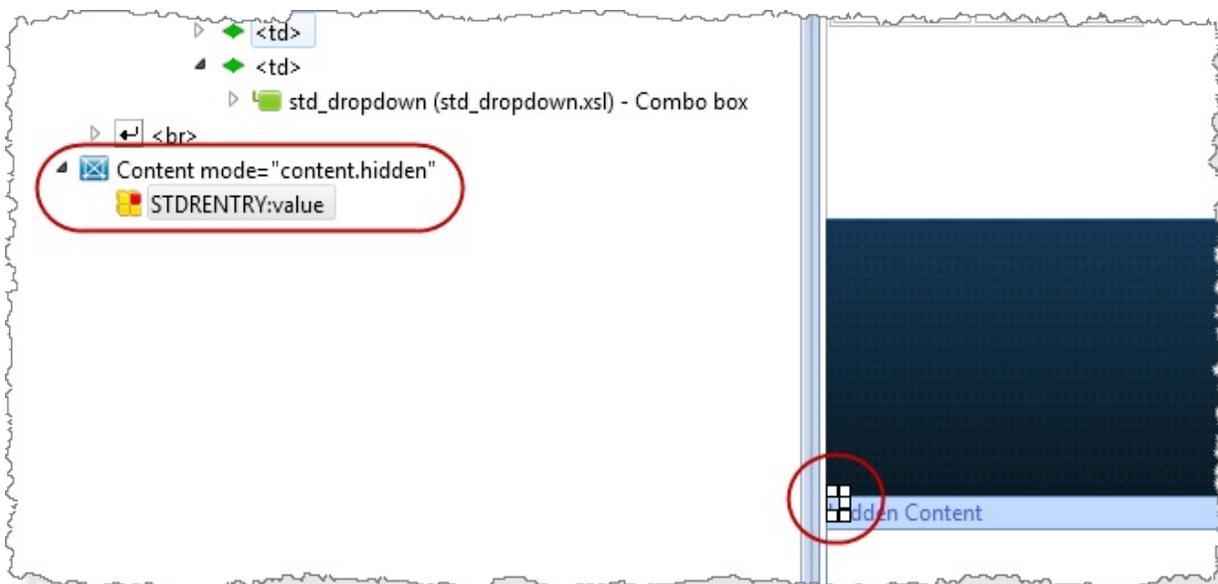
Radio Button Radio 1 Radio 2

Dropdown

Step 8. Make STDREENTRY visible for testing

In this step, you will learn how to select and change a hidden field on the page. When testing a WAM, if you are running into problems, sometimes you may want to make a hidden field visible for a better understanding of what is happening.

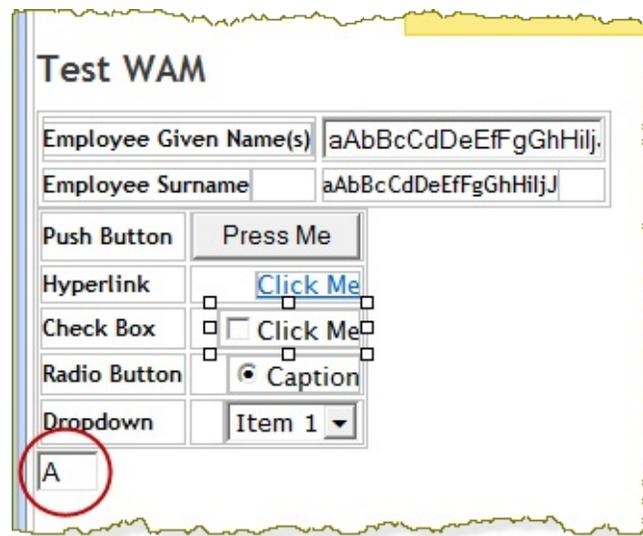
1. Select a hidden field:
 - a. Select the *Outline* tab. Every object on the page will be listed here regardless of what it is, or what its display mode is.
 - b. On the *Outline* tab, expand the Content mode="content.hidden" group and select STDREENTRY. In the *Design View*, STDREENTRY will now be selected at the top of the *Hidden Content* section at the bottom of the web page, with "Grips" around it. If the grips do not appear, click on the web page in the Design view to ensure this is selected.



- c. Right click on the highlighted field in the Hidden Content area and select *Change to Input Field*. Be careful not to deselect STDREENTRY until you have changed it into an input field. If it does happen to become deselected and you can no longer see it, select it again in the Outline tab.



2. Select the field STDRENTY in the *Hidden Content* area and use the right mouse menu to *Cut* it. Click on the main page area below the new table and use the right mouse menu to *Paste* the field here. Your design should now look like the following:



3. Save the changes to the layout.

Step 9. Test the WAM

In this step, you will test the WAM.

1. *ReentryTest* should still be open in the *Design* view. If not, open it.
2. Click the  Run button on the toolbar, to run your WAM in the Web Browser.



The screenshot shows a web application titled "Test WAM" with a light blue background. In the top right corner, there is a yellow message box that says "Messages: • Dropdown was changed." Below the title, there are several form elements: "Employee Given Name(s)" with the value "JOHN", "Employee Surname" with the value "Smith", a "Push Button" labeled "Press Me", a "Hyperlink" labeled "Click Me", a "Check Box" labeled "Press Me" which is unchecked, a "Radio Button" with two options "Radio 1" and "Radio 2", and a "Dropdown" menu with "Item 1" selected. At the bottom left, there is a small text input field containing the letter "E".

3. Use each Weblet and see that the appropriate messages are given. If they are not, check the `STDREENTRY` to see what its value is and make sure the Weblet itself has the correct `reentryvalue` set. Make any necessary changes and test again.
4. When you are satisfied that the page is functioning correctly, close the browser.

Step 10. Hide STDREENTRY

In [Step 9. Test the WAM](#), you have confirmed that the WAM is functioning properly, so you don't need STDREENTRY to be visible for testing. You will hide it again in this step.

1. Open *ReentryTest* in the Design view.
2. Drag STDREENTRY back into the Hidden Content area at the bottom of the web page.
3. At run time it will now be hidden, but if you would like to make it hidden in the *Design View* also, right click on the field and select *Change to Hidden Field*.
4. Save the changes to the layout.

Summary

Important Observations

- A WAM layout is created using *Theme Layout Weblet #1 – One Column* by default if a layout is not selected on the Create WAM dialog.
- If a WAM has not been compiled, if a WebRoutine is opened in the Design View, the web page will be generated using *std_themelet1_1col* weblet and the fields and lists mapped for output in the WebRoutine.
Note: In this case the web page will not be re-generated when the WAM is compiled unless the *Generate XSL – All WebRoutines* option is selected. There are more details on this subject in later exercises.
- Once a WAM is compiled, a web page with a WAM specific layout (that you can modify) and field entry controls is generated for each WebRoutine that can be edited in the Design view. You can drag and drop fields and lists marked FOR(*OUTPUT) or FOR(*BOTH) onto your page at any time from the Design view's *WebRoutine Output* tab.
- Weblets are reusable XSL snippets or components that can be plugged into either your WebRoutine page or other Weblets. There are standard Weblets that encapsulate HTML buttons, hyperlinks, menu items, radio buttons, dropdown lists, etc.
- When viewing a list of available Weblets in *Weblet Templates* view, you will notice a Weblet with a Description of *Push button* will also have a *Details* column showing *std_button_v2*. *Std_button_v2.xsl* is an XSL document consisting of one or more `<xsl:template>` element.
- The *Compile options* dialog, lets you choose whether or not to re-generate the XSL for existing WebRoutines. By default, a compile will always generate XSL for new WebRoutines ONLY. That is, the web page design for each new WebRoutine is generated automatically.
- The generated web page includes all fields and LISTS in your WEB_MAPs marked FOR(*OUTPUT) or FOR(*BOTH). Fields mapped with a FOR(*INPUT) keyword, are incoming fields to the WebRoutine. They are not available when creating the web page output by this WebRoutine.
- The compile generates a WAM-specific layout Weblet. Its name is WAM *Identifier* followed by *_layout* (e.g. *iiifirst_layout*) where **iiifirst** is the WAM's *Identifier*.
- The generated web page also has a *Messages (std_messages)* Weblet on the

page, a WebRoutine description in a heading and a *Hidden Content* DIV where you may place objects that you do not want to be displayed on the page.

Tips & Techniques

- You can compile from either the LANSAs Editor directly, or from the Repository or Last Opened tabs.
- It is not necessary to do a compile to generate the web page, since a build also generates it, but without doing a full compile.
- When compiling, the *Generate XSL* option always defaults to the **New WebRoutines** option.
- The Generate XSL **All WebRoutines** option will generate, or re-generate the XSL for ALL WebRoutines in the WAM and will lose any changes made to the web pages in the Design view.
- You can make a hidden field visible for debugging purposes.

What I Should Know

- How to create a Web Application Module (WAM).
- How to define a WebRoutine.
- How to use a WEB_MAP statement to define inputs and outputs to a WebRoutine.
- How to open the Design view for a specific WebRoutine.
- How to use some of the basic editing features of the Design view.
- How to create a table.
- How to use a Weblet.
- How to compile a WAM.
- What a compile generates.

WAM010 - Using WEB_MAPs

Objectives

- To demonstrate how the WEB_MAP affects the transfer of data between two WebRoutines.

In this exercise, you will see how fields are mapped from the WebRoutine executing on the server to the page and from the page back to the WebRoutine.

First you will create a WebRoutine that will pass all of the fields as input and output. Since all of the fields will be both input to and output from the WebRoutine, the data will be preserved when it calls itself. You will make changes to the fields and test this.

You will create a second WebRoutine that will not take all of the fields as input. You will transfer between the two WebRoutines to see how some data is lost, to better illustrate how the WEB_MAP statement works.

To achieve these objectives, you will complete the following:

- [Step 1. Create a new WAM](#)
- [Step 2. Add WebRoutines to the new WAM](#)
- [Step 3. Compile the WAM and Open for Editing](#)
- [Step 4. Add buttons to the WebRoutine](#)
- [Step 5. Understand the Web Routine](#)
- [Step 6. Change the Employee Number field](#)
- [Step 8. Add buttons to the WebRoutines](#)
- [Step 9. Understand WEB_MAP](#)
- [Summary](#)

Before you Begin

In order to complete this exercise, you should have completed the following:

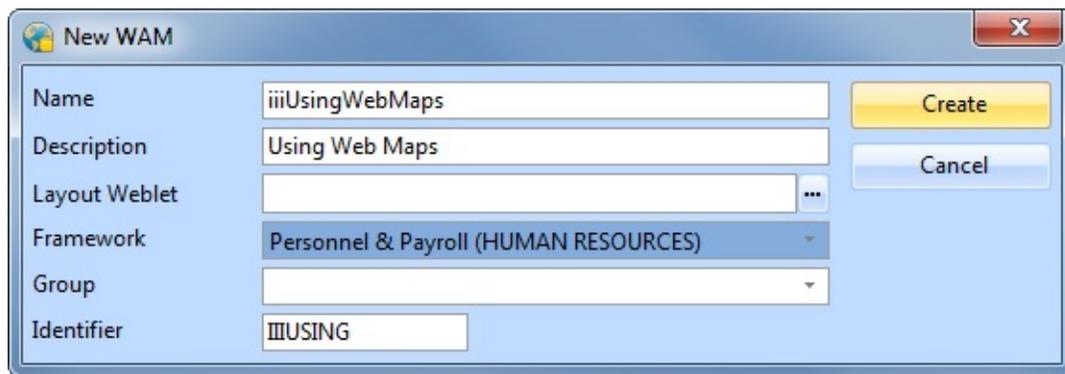
[WAM005 - Create Your First WAM](#)

Step 1. Create a new WAM

In this step, you will create a new Web Application Module that will eventually demonstrate the use of the WEB_MAP statement.

1. In the LANSAs Editor window, click the *New button* and choose *Web Application Module*.

The *New WAM* dialog will appear:



| | | |
|---------------|---------------------------------------|--------|
| Name | iiiUsingWebMaps | Create |
| Description | Using Web Maps | |
| Layout Weblet | | Cancel |
| Framework | Personnel & Payroll (HUMAN RESOURCES) | |
| Group | | |
| Identifier | IIIUSING | |

2. In the *New WAM* dialog box:
 - a. Enter a *Name* of **iiiUsingWebMaps** (where **iii** are your initials).
 - b. Enter a *Description* of **Using WEB_MAPs Demo**.
 - c. Leave the *Layout Weblet* field blank.
 - d. Select the *Personnel & Payroll Framework*.
 - e. Click the *Create* button to create the new WAM.
3. The LANSAs Editor will now display the WAM's RDMLX code. At this stage, it will not contain any WebRoutines.

Step 2. Add WebRoutines to the new WAM

In this step, you will add the RDMLX code consisting of multiple WebRoutines to the newly created WAM.

1. Immediately following the BEGIN_COM, insert the following RDMLX code to create a WebRoutine named WMdemo:

```
Webroutine Name(WMdemo) Desc('WEB_MAP WR1')
Endroutine
```

2. All fields in the WebRoutine will be both incoming and outgoing, so they will be specified FOR(*BOTH). By default, all of the fields will be displayed as input fields. You could write the following WEB_MAP statement for the WebRoutine:

```
WEB_MAP FOR(*BOTH) FIELDS(#EMPNO #GIVENAME #SURNAME #
```

However, a GROUP_BY may be used in a WEB_MAP, so you will use the following code:

```
Begin_Com Role(*EXTENDS #PRIM_WAM)
Group_By Name(#Empdata) Fields(#empno #surname #givename #address1
#postcode)
Webroutine Name(WMdemo) Desc('WEB_MAP WR1')
Web_Map For(*BOTH) Fields(#empdata (#stdrentry *hidden))
... WebRoutine
```

- The STDREENTRY field will be used to control the logic within the WAM. Included in this logic will be an IF Statement that will test the value of STDREENTRY to determine if the other fields should be replaced (Refreshed) with data from the Personnel Master File.
- The STDREENTRY field should not be visible on the HTML page.
- The other fields will be used to demonstrate how the WEB_MAP works.

Note: A WEB_MAP statement with Keyword FOR(*BOTH) specifies that the fields listed in the WEB_MAP are both input to and output from the WebRoutine.

- A WEB_MAP FOR(*INPUT) is for fields that will only be received as input to the WebRoutine.
- A FOR(*OUTPUT) is used in cases where fields will be sent out from the WebRoutine only.

- A FOR(*NONE) option will be explained in a later exercise.

In the Fields() keyword of WEB_MAP, the fields are specified with their display mode. The display mode attribute only plays a role for fields that are FOR(*OUTPUT) or FOR(*BOTH).

By default, all fields are displayed on the web page as input fields. To change the way the fields display on the web page, the display mode can be set to *INPUT, *OUTPUT, *HIDDEN, or *PRIVATE.

*HIDDEN fields are included in the page as hidden fields. That is, their values are mapped but they are not shown on the web page.

*PRIVATE fields are not shown on the web page but are available in the XML for use in weblets such as a dropdown lists.

3. Add the code to initialize the fields in the WebRoutine. Since EMPNO is both incoming to and outgoing from the WebRoutine, its value should never be lost. If EMPNO is blank, that means it is the first time entering the WebRoutine. You will retrieve a valid employee number by simply reading the first record from the Personnel Master.

```
If Cond(#empno = *blanks)
Select Fields(#empdata) From_File(pslmst)
Leave
Endselect
Endif
```

Note the following about this code:

The SELECT loop with an unconditional LEAVE, will return values for the first record read. Typically you should not rely on values returned outside a SELECT loop because the returned values may be unpredictable. This technique has been used for the sake of simplicity. An alternative could be:

```
Fetch Fields(#EMPDATA) From_file(PSLMST)
```

With no key specified, the FETCH will return the first record in the file.

4. A *Read* button will be required to read data from the Personnel Master file when an Employee Number has been entered on the web page. The read will do a FETCH on the Personnel Master, using EMPNO as the key and will be triggered by a button calling the WebRoutine with a STDREENTRY value of 'R'.

```
If Cond(#stdreentry = R)
```

```
Fetch Fields(#empdata) From_File(pslmst) With_Key(#empno)
Endif
```

Your finished WebRoutine should appear as follows:

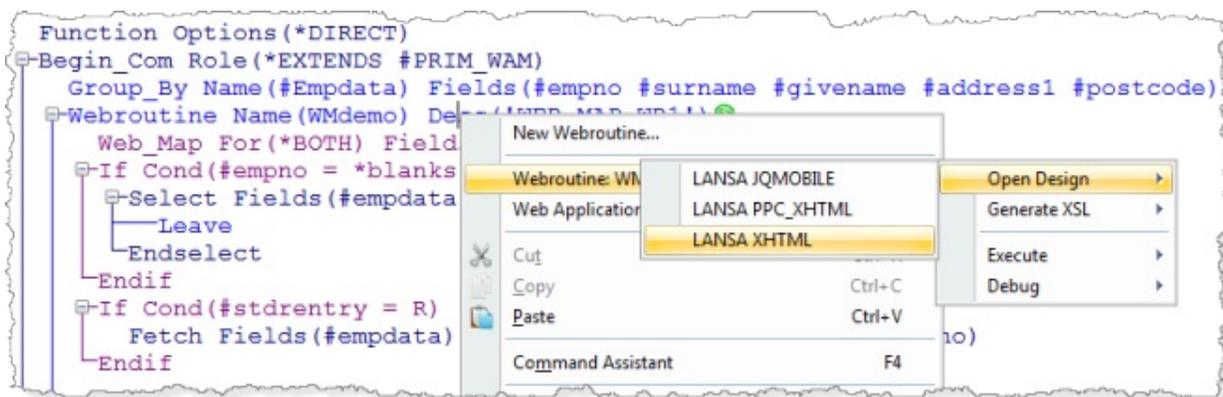
```
Group_By Name(#Empdata) Fields(#empno #surname #givenname #address1
#postcode)
Webroutine Name(WMdemo) Desc('WEB_MAP WR1')
Web_Map For(*BOTH) Fields(#empdata (#stdrentry *hidden))
If Cond(#empno = *blanks)
Select Fields(#empdata) From_File(pslmst)
Leave
Endselect
Endif
If Cond(#stdrentry = R)
Fetch Fields(#empdata) From_File(pslmst) With_Key(#empno)
Endif

EndroutineWebRoutine
```

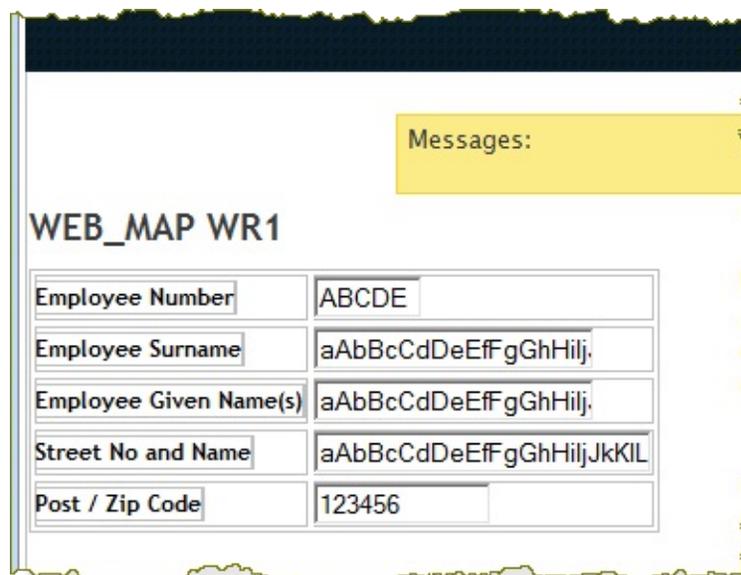
Step 3. Compile the WAM and Open for Editing

In this step, you will compile the WAM.

1. You should still have the WAM **iiiUsingWebMaps** (where **iii** are your initials) open in the LANSAs Editor.
2. Click the  *Compile* button on the LANSAs Editor toolbar to compile the WAM component.
3. After the compile completes successfully, open the WMdemo WebRoutine in the Design view by right clicking anywhere within the WMDemo WebRoutine statement and selecting WebRoutine: WMdemo /Open Design/LANSAs XHTML.



The *Design* view should appear something like the following:



Messages:

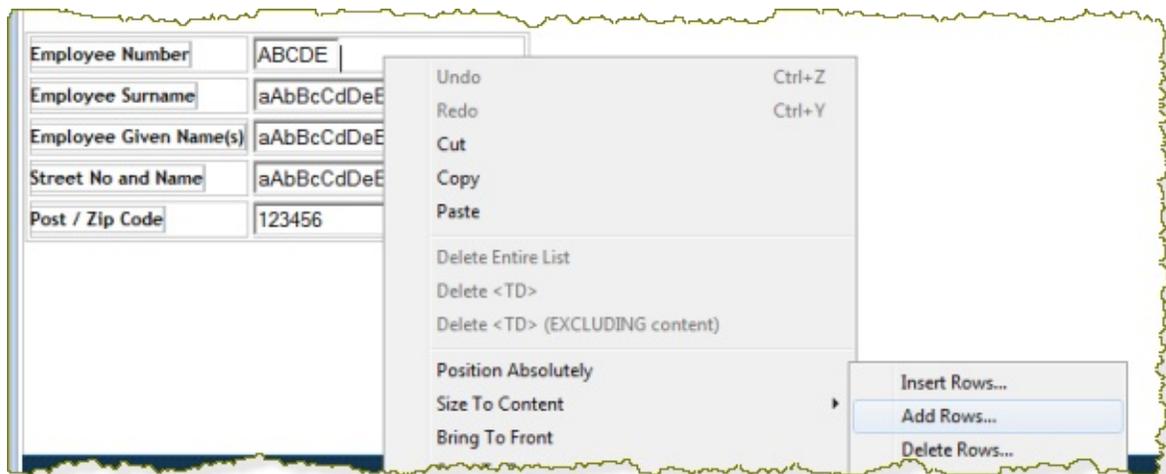
WEB_MAP WR1

| | |
|------------------------|------------------------|
| Employee Number | ABCDE |
| Employee Surname | aAbBcCdDeEffGhHij, |
| Employee Given Name(s) | aAbBcCdDeEffGhHij, |
| Street No and Name | aAbBcCdDeEffGhHijJkKIL |
| Post / Zip Code | 123456 |

Step 4. Add buttons to the WebRoutine

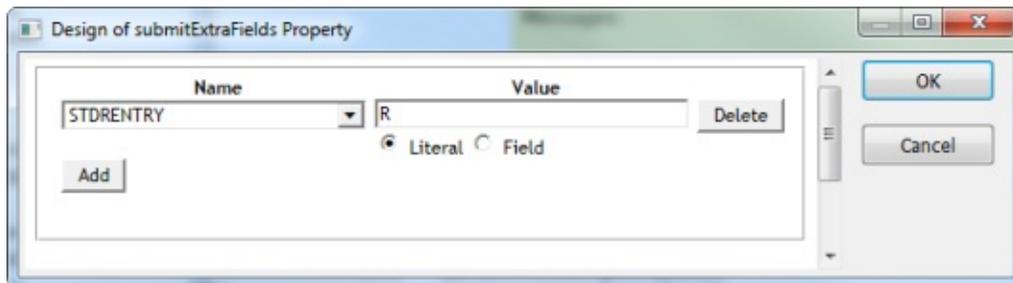
In this step, you will learn how to add a row to the generated table and then create the buttons that will invoke this WebRoutine. A reentrant WebRoutine is designed so that from the web page that is presented to the user, after the WebRoutine ends, navigation elements are provided that will allow the WebRoutine to be initiated once again. Data on the page will be used in the WebRoutine to control the logic performed by the routine. This technique is used to reduce the number of WebRoutines required for a particular application.

1. Add a new row to the table:
 - a. Right click in the table that contains the data entry fields. For example click to the right of the Employee Number field input box and then right click.
 - b. Select the *TableItems* menu item and choose the *Add Rows...* option from the pop-up menu. Click OK. A row will be added to the end of the table.



2. This step adds the *Read* button:
 - a. From the Favorites / *Weblet Templates* tab, drag and drop a *Push button* weblet into the leftmost cell of the newly added row.
 - b. Click on the empty space of the cell containing the button.
 - c. Select the *Details View*.
 - d. Set the *align* property to **left**.
3. Set the *Read* button properties:
 - a. Select the new button in the *Design View*.

- b. Set the *caption* to **Read**.
- c. Set the *on_click_wrname* to **WMdemo**.
- d. Click in the *Value* column for the *submitExtraFields* property and use the *Ellipsis* button to open the *Design of...* dialog. In the *Name* column select **STDREENTRY** from the dropdown list. In the *Value* column enter **R** and leave the checkbox as **Literal**. Click *OK* to close the dialog and save your changes. The button click will return the field STDREENTRY with a value of R.



4. Add the button to reload the WebRoutine:
 - a. From the *Weblet Templates* tab, drag and drop a *Push button* weblet into the rightmost cell of the newly added row.
 - b. Click on the empty space of the cell containing the button.
 - c. Select the *Details View*.
 - d. Set the *align* property to **right**.
5. Set the button's properties:
 - a. Select the new button in *Design View*.
 - b. Set the *caption* to **WMDemo**.
 - c. Set the *on_click_wrname* to **WMdemo**.
 - d. Click in the *Value* column for the *submitExtraFields* property and use the *Ellipsis* button to open the *Design of...* dialog. In the *Name* column select **STDREENTRY** from the dropdown list. In the *Value* column enter ' ' and leave the checkbox as **Literal**. Click *OK* to close the dialog and save your changes. The button click will return the field STDREENTRY with a blank value.

Messages:

WEB_MAP Demo WR1

| | |
|------------------------|---------------------------|
| Employee Number | ABCDE |
| Employee Given Name(s) | ABCDEFGHIJKLMNOPQR |
| Employee Surname | ABCDEFGHIJKLMNOPQR |
| Street No and Name | aAbBcCdDeEfFgGhHiIjJkKlLm |
| Post / Zip Code | 123456 |
| Read | WMDemo |

6. Save the changes to WMDemo.

Step 5. Understand the Web Routine

In this step, you will test the WebRoutine and gain an understanding of how it works.

1. WMdemo should still be open in the *Design* view, if not, open it.
2. Click the  Run button on the toolbar to run the WebRoutine in the browser..

This will open your default Web Browser and request the URL to execute the WebRoutine **WMdemo**.

3. What happens when no logic in the WebRoutine is executed?

- a. Click on the WMDemo Button.

This will execute the WMdemo WebRoutine. Notice nothing has changed or been lost.

- b. Try changing any field except Employee Number. Click the WMDemo button again to see that the data has been preserved. The data on the web page was sent to the WMdemo WebRoutine where the WEB_MAP FOR(*BOTH) accepted the values as input.

The RDML logic will not be executed as EMPNO is not blank and STDRENTY is not equal **R**. When the ENDROUTINE is encountered, the field's values are sent out to the web page as the WEB_MAP is designated FOR(*BOTH), that is both input and output.

4. Let us look at the effect when some of the logic in the WebRoutine is executed.

As you have changed some fields, click the *Read* button. Recall, the read logic, that does a FETCH on the Personnel Master with the Employee Number as the key.

The data on the web page should be refreshed with the values for the key EMPNO. This is because the field STDRENTY is set to R.

The WEB_MAP accepts the EMPNO and STDRENTY fields along with other data from the page. The FETCH using the EMPNO field is executed and the data for the other fields on the page is set to the values from the master file PSLMST.

If there is no employee record for the value of EMPNO, the original values are returned to the web page. When the ENDROUTINE is encountered, the

field's values are sent out to the web page as the WEB_MAP is designated FOR(*BOTH) that is both input and output.

5. What happens if you change the Employee Number field? Or change Employee Number, as well as some other fields?
 - a. Now click the WMDemo button? The data is preserved , as it should be, as long as employee number is not blank.
 - b. Click the *Read* button. If an employee number that exists on the PSLMST was entered, the details for that employee will be displayed. Suitable employee number values are A0090, A0070 or A1001.

In both cases, when the ENDROUTINE is encountered, the field's values are sent to the web page as the WEB_MAP is designated FOR(*BOTH) meaning the fields are both input and output from the WebRoutine.

6. To force the WebRoutine to select the first employee again, click the WMDemo button with no employee number present.

By this point you should understand what the buttons are doing.

This exercise demonstrated how WEB_MAPS with all fields with a FOR(*BOTH) keyword send and accept data from the web page.

Step 6. Change the Employee Number field

In this step, you will change Employee Number to be an output only field. This will prevent the field from being changed and change the behavior you experienced in the last step. You need to be aware that fields that are displayed as **output** are generated as **text** on the page. Since it is text and not a field, the necessary tags to pass the field to the next WebRoutine are not generated.

You must add the field EMPNO a second time as a hidden field to preserve the data.

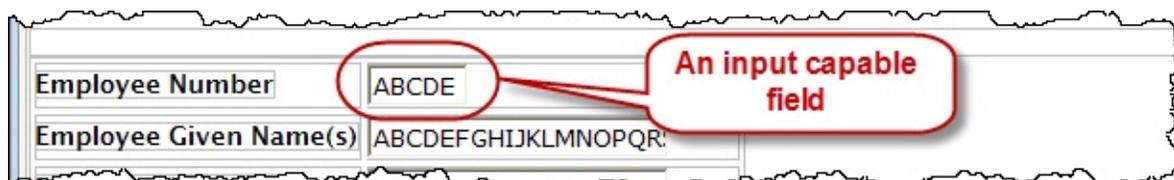
You will also see the dangers of generating XSL for ALL WebRoutines on compile, as opposed to only generating the XSL for new WebRoutines.

1. Change the GROUP_BY to display Employee Number as output only.

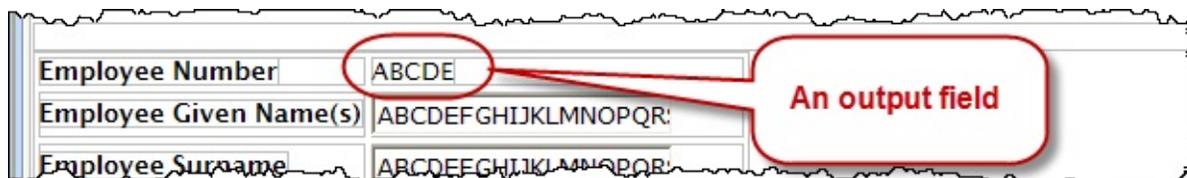
```
Group_By Name(#Empdata) Fields((#empno *output) #surname #givenname #address1 #postcode)
```

2. Click the *Compile* button in the LANSa Editor toolbar to compile the WAM component.
3. After the compile completes successfully, reopen the WMdemo WebRoutine in the Design view.
4. Notice that the Employee Number field has not changed. You could still input a value to it. This is because a compile, by default, generates the XSL for NEW WebRoutines only.

An input capable field in the *Design* view:



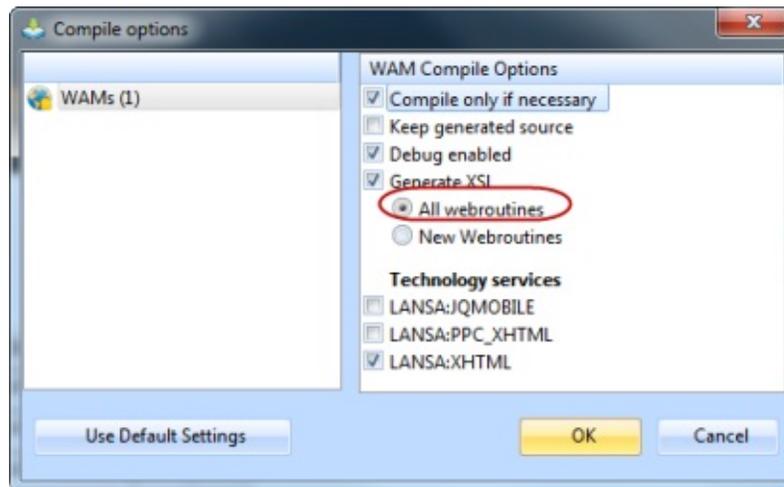
An output field in the *Design* view:



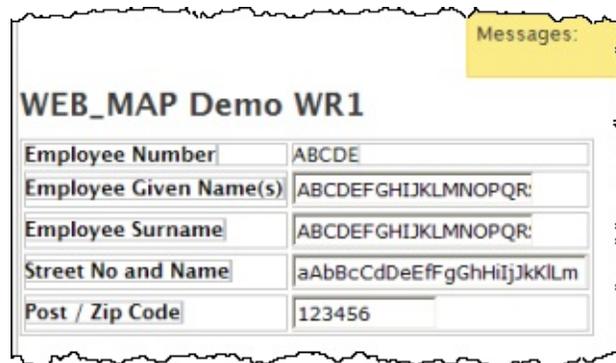
Note: This is the safe *Compile* option because the web page design for all existing WebRoutines will remain unchanged when a WAM is recompiled.

You may choose to re-generate the XSL for all WebRoutines by changing the default of the *WAM Compile Options* from *New WebRoutines* to *All WebRoutines*. If this is done then all XSL Editing changes will be lost for all WebRoutines in the WAM. Therefore you would only select this option if you wish to discard all your web page design changes for the WAM's WebRoutines.

The following images depicts the effect on WMDemo if this option was taken. **Please do not perform these steps.** They are only for information. You will continue the exercise at point 5.



If check *All WebRoutines* in the *Compile options* window is selected, the Design view would appear something like the following:

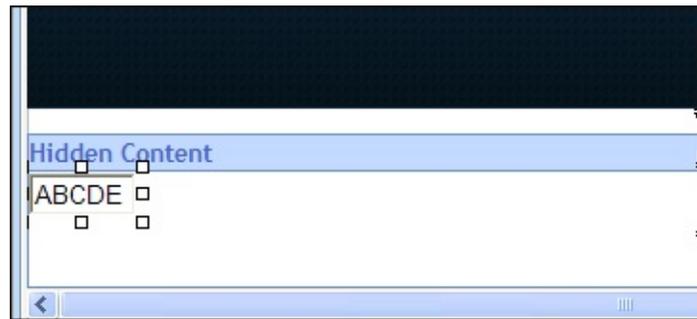


Notice Employee Number is displayed as an output field now, but the buttons added in the previous step have been lost.

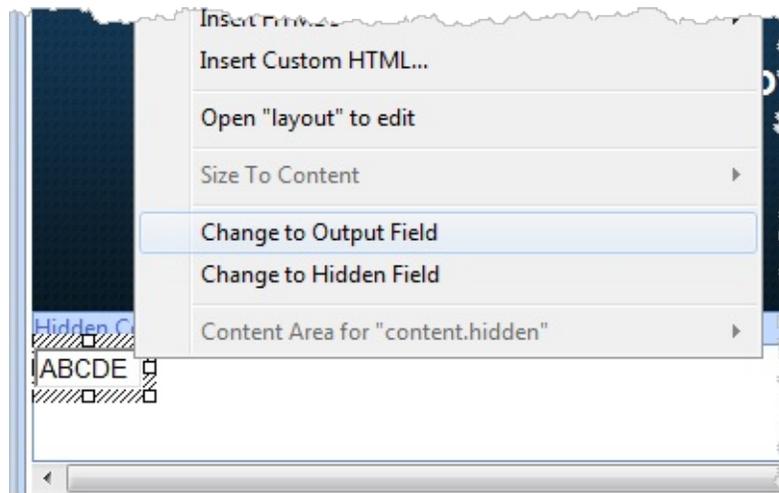
If you were to take this approach, you would have to re-add a row to the table and recreate the buttons.

5. Add EMPNO to the page again.

- a. From the WebRoutine Output *tab*, select the EMPNO field and drag it into the top right cell of the table, to the right of the EMPNO field that is currently on the page.
 - b. Delete the caption that was added when you added the field (this caption will already be selected after dragging the field onto the page).
6. Hide the Input capable instance of EMPNO
- a. Drag the original EMPNO field (not the one you just added) into the *Hidden Content* area at the bottom of the page.



- Fields in the *Hidden Content* DIV will not be displayed on the page in the browser, but will be visible in the *Design View*.
- If you would like to hide the field in the *Design View* also, right click on it and select *Change to hidden field*.



Note: You can change the display mode of a field at any time in the *Design* view by right clicking the field and selecting *Change to Input Field*, *Change to Output Field*, or *Change to Hidden Field*, without having to change the WEB_MAP.

However, note that if you want to map the field into the WebRoutine, the Web_Map must have a FOR(*INPUT) or FOR(*BOTH) parameter.

7. Save your changes and test the page again.

Step 7. Add the RDMLX for the second WebRoutine

In this step, you will create the RDMLX code for a second WebRoutine to demonstrate how data is passed between WebRoutines.

1. Immediately following the ENDRoutine for WMdemo, insert the following RDML code to create a WebRoutine named WMdemo2:

```
Webroutine Name(WMdemo2) Desc('Web_Map Demo WR2')
Endroutine
```

2. This WebRoutine will require two outgoing fields (ADDRESS1 and POSTCODE) and four fields that are both incoming and outgoing (GIVENAME, SURNAME, EMPNO and STDREENTRY). EMPNO and STDREENTRY must be FOR(*BOTH) to support the functionality of the WebRoutine. The other four fields were arbitrarily divided to show the behavior of the WEB_MAP, but don't necessarily represent a practical use. Add the following WEB_MAP statements to the WebRoutine:

```
Web_Map For(*BOTH) Fields((#empno *output) #givename #surname
(#stdreentry *hidden))
```

```
Web_Map For(*output) Fields(#Address1 #POSTCODE)
```

- The **STDREENTRY** field will be used to determine what button was pressed, and decide what logic to execute.
 - The **STDREENTRY** field should not be visible on the HTML page.
 - The **EMPNO** field should be output only, so the user cannot change it.
 - The other fields will be used to demonstrate how the WEB_MAP works.
3. The logic for the *Read* button will be the same as in WMdemo, that is, it will **FETCH** the record from the Personnel Master when the **STDREENTRY** value is 'R'.

```
If Cond(#stdreentry = R)
```

```
Fetch Fields(#empdata) From_File(pslmst) With_Key(#empno)
```

```
Endif
```

Your finished WebRoutine should appear as follows:

```
Webroutine Name(WMdemo2) Desc('Web_Map Demo WR2')
```

```
Web_Map For(*BOTH) Fields((#empno *output) #givename #surname
(#stdreentry *hidden))
```

```
Web_Map For(*output) Fields(#Address1 #POSTCODE)
```

```
If Cond(#stdreentry = R)
```

```
Fetch Fields(#empdata) From_File(pslmst) With_Key(#empno)
Endif
Endroutine
```

4. Click the  *Compile* button in the LANSAs Editor toolbar to compile the WAM component.

Step 8. Add buttons to the WebRoutines

In this step, you will add a row to the table and then create the buttons that will reload the WebRoutine, as you did in [Step 4. Add buttons to the WebRoutine](#).

You will also add a button to call WMdemo.

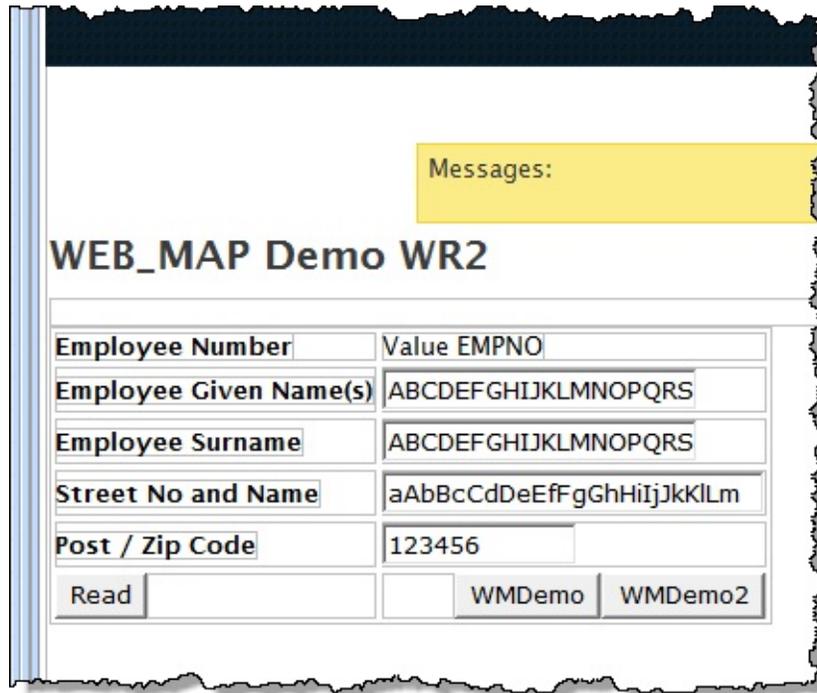
Finally, in the WMdemo web page you will add a button to call WMdemo2.

1. Open the new WebRoutine, WMdemo2 in the Design view.
2. This step adds a new row to the bottom of the table:
 - a. Right click on the table that contains the data entry fields.
 - b. Select the *Table Items* menu item and choose the *Add Rows* option from the pop-up menu.
3. Now add the *Read* button:
 - a. From the Favorites/*Weblet Templates View*, drag and drop the *Push button* weblet into the leftmost cell of the new row.
 - b. Click on the empty space of the cell containing the button.
 - c. Select the *Details View*.
 - d. Set the *align* property to **left**.
4. Set the *Read* button properties:
 - a. Select the new button in *Design View*.
 - b. Set the *caption* property to **Read**.
 - c. Set the *on_click_wrname* property to **WMdemo2**.
 - d. Click the Ellipsis button for the *submitExtraFields* property and set the returned field *Name* to **STDREENTRY** and the *Value* to **R** to indicate that the users wants to refresh.
 - e. Click *OK* to confirm the change.
5. This step adds the button to reload the WebRoutine:
 - a. From the Favorites/*Weblet Templates View*, drag and drop the *Push button* weblet into the rightmost cell of the newly added row.
 - b. Click on the empty space of the cell containing the button.
 - c. Select the *Details View*.
 - d. Set the *align* property to **right**.

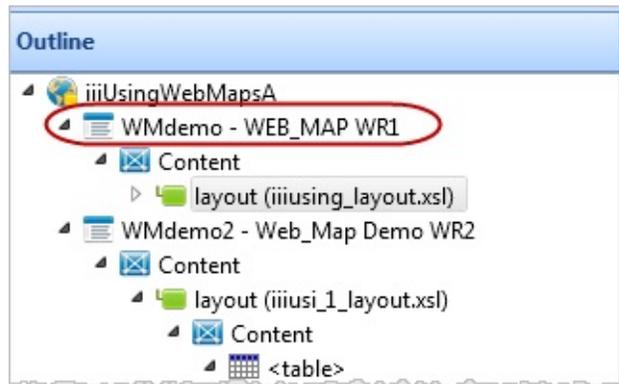
6. Set the button's properties:
 - a. Select the new button in *Design View*.
 - b. Set the *caption* property to **WMDemo2**.
 - c. Set the *on_click_wrname* property to **WMdemo2**.
 - d. Click in the *Value* column for the *submitExtraFields* property and use the *Ellipsis* button to open the *Design of...* dialog. In the *Name* column select **STDREENTRY** from the dropdown list. In the *Value* column enter ' ' and leave the checkbox as **Literal**. Click *OK* to close the dialog and save your changes. The button click will return the field STDREENTRY with a blank value.
7. Add a hidden copy of the field EMPNO to the page:
 - a. From the WebRoutine Output View, select the EMPNO field and drag it into the top right cell of the table.
 - b. Delete the caption that was added when you added the field.
 - c. Change the new EMPNO field to an Input Field.
 - d. Drag the new EMPNO field into the *Hidden Content* area at the bottom of the web page.
 - e. If you would like to hide the field in the *Design View* also, right click on it and select *Change to hidden field*.

Note: The hidden EMPNO field is required so that a value will be mapped into the WebRoutine WMDemo2, when the WMDemo2 button is clicked.
8. Add the button to transfer to WMdemo. From the Favorites/*Weblet Templates View*, drag and drop the *Push button* weblet into the rightmost cell of the newly added row.
9. Set the button's properties:
 - a. Select the new button in *Design View*.
 - b. Set the *caption* property to **WMDemo**.
 - c. Set the *on_click_wrname* property to **WMdemo**.
 - d. Click in the *Value* column for the *submitExtraFields* property and use the *Ellipsis* button to open the *Design of...* dialog. In the *Name* column select **STDREENTRY** from the dropdown list. In the *Value* column enter ' ' and leave the checkbox as **Literal**. Click *OK* to close the dialog and save your changes. The button click will return the field STDREENTRY with a blank

value

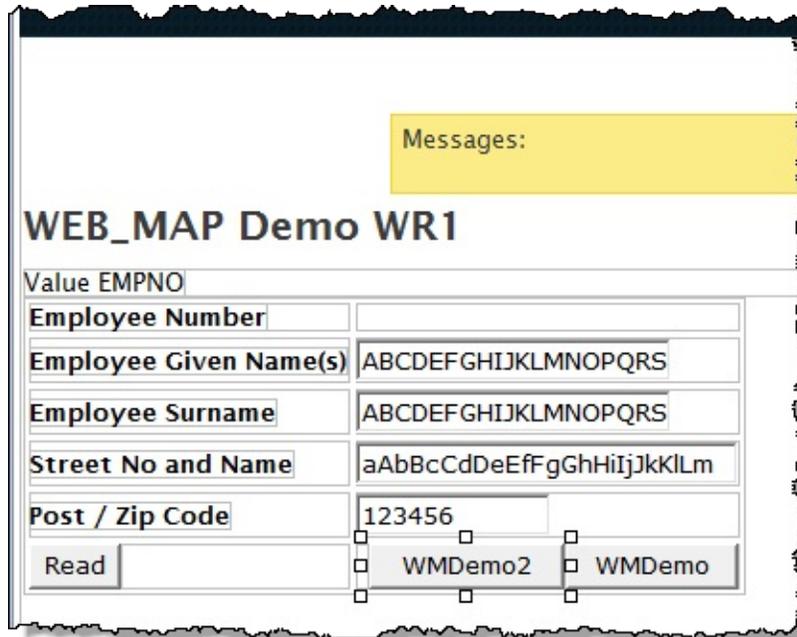


10. Save your changes to WebRoutine WMDemo2.
11. Open WebRoutine WMDemo in the Design view. To do this, select the *Outline* tab and double click on the WMDemo entry:



12. Add a new button in WMDemo, to invoke the WebRoutine WMDemo2. From the Favorites/*Weblet Template View*, drag and drop the *Push button* weblet into the rightmost cell of the bottom row.
13. Set the button's properties:
 - a. Select the new button in *Design View*.
 - b. Set the *caption* property to **WMDemo2**.

- c. Set the *on_click_wname* property to **WMdemo2**.
- d. Click in the *Value* column for the *submitExtraFields* property and use the *Ellipsis* button to open the *Design of...* dialog. In the *Name* column select **STDREENTRY** from the dropdown list. In the *Value* column enter ' ' and leave the checkbox as **Literal**. Click *OK* to close the dialog and save your changes. The button click will return the field STDREENTRY with a blank value.



14. Save your changes to WMdemo.

Step 9. Understand WEB_MAP

In this step, you will test out the WebRoutines that you have just created and see the WEB_MAP in action.

1. Open WMDemo in the Design view.
2. Use the Execute button in the toolbar to run the WebRoutine in the *web browser*.
3. Click the WMDemo2 button and observe that some of the data in these fields is not transferred to the WMDemo2 WebRoutine.

This is because ADDRESS1 and POSTCODE are output only, so although they are sent to WMDemo2 by clicking the WMDemo2 button on the WMDemo web page and are shown on the web page for WMDemo2, they are not accepted by the WMDemo2 WebRoutine as input, therefore the values of the fields are lost when transferring to the WebRoutine WMDemo2.

4. Click the *Read* button to populate all of the fields again. Now click the WMDemo button. See that all of the data is preserved. All of the fields are output from WMDemo2 and they are all accepted as input to WMDemo, so data will not be lost.
5. The employee number is always retained in its output field because the web page for WMDemo and WMDemo2 WebRoutines contain a hidden copy of employee number, which is mapped into the WebRoutine.

Note: In web applications, it is often necessary to display a key field value for output and have a hidden copy of the field on the page, in order to pass a value into the next WebRoutine.

Summary

Important Observations

- The WEB_MAP statement allows you to specify which incoming and outgoing fields the WebRoutine maps between the web page and the RDMLX code. The WEB_MAP statement's FOR() selector specifies whether the fields are mapped as incoming (*INPUT), outgoing (*OUTPUT), or both (*BOTH).
- Only fields in a WEB_MAP statement with a FOR(*OUTPUT) or FOR(*BOTH) can be placed on the web page.
- The Fields() attributes specify the display mode of the field on the page. Acceptable field attributes are *INPUT, *OUTPUT and *HIDDEN. If unspecified, the default is *INPUT. These attributes determine whether the field accepts input, as an input text box, only displays an output value, or is hidden on the page.
- Hidden fields may be needed if fields that are shown as output values on the web page, but their value must be posted to the web server. This is often the case for key fields when the WebRoutine is performing an update.
- The order in which fields appear when generated is the same as their sequence in the WEB_MAPs, although when data is mapped to and from the WEB_MAPs they are mapped by field names and not position.
- A WebRoutine may have multiple WEB_MAPs.
- If the output of one WebRoutine does not match the input of the WebRoutine that it is calling, no error will be generated. The data will simply not be passed into the WebRoutine that was called.

Tips & Techniques

- In addition to specifying WEB_MAPs inside WebRoutine blocks, you are allowed to declare WEB_MAPs inside a BEGIN_COM block of a WAM. This technique allows you to map fields and lists into every WebRoutine in your WAM without having to explicitly define WEB_MAPs in each WebRoutine.
- You can change the WEB_MAP after the first compile, but you must update the web page for the fields affected. e.g. if a field's display attribute was changed in the WEB_MAP from *input to *output, the web page could be corrected either by changing the field on the page to "Output Field" or delete

the field and drag and drop it onto the page from the WebRoutine Output tab.

- You can modify the display mode of a field in the Design view without having to change the WEB_MAP by right clicking the field and changing to the appropriate display mode. However, note that if you change a field to be *INPUT in the *Design* view, then it must be mapped into the WebRoutine using For(*input) or For(*both) in order for the value to be processed.
- Similarly if you change a field to be *OUTPUT in the *Design* view, and it is mapped into the WebRoutine as For(*input) the field value will never be received by the WebRoutine.
- For consistency, we recommend you control field input and output attributes via their WEB_MAP statements, rather than by tweaking the *Design* view.

What I Should Know

- How the WEB_MAP works.

WAM015 - Working Lists

Objectives

- To demonstrate how to use and change Working Lists on the page.

In this exercise, you will create a WAM with a single WebRoutine that will populate a working list. This WebRoutine will have no other function, but will teach you how to use a working list on a page and how to condition the field's display modes.

To achieve this Objective, you will complete the following:

- [Step 1. Create a new WAM](#)
- [Step 2. Add RDMLX code to the new WAM](#)
- [Step 3. See how the working list is displayed](#)
- [Step 4. Change the display mode of fields in the list](#)
- [Step 5. Use the generate XSL for all WebRoutines option](#)
- [Step 6. Modify the list in the Design view](#)
- [Summary](#)

Before you Begin

In order to complete this exercise, you should have completed the following:

- [WAM005 - Create Your First WAM](#)
- [WAM010 - Using WEB_MAPs](#)

Step 1. Create a new WAM

In this step, you will create a new Web Application Module that you will use to become familiar with working lists.

1. In the LANSAs Editor window, click the *New button* choose *Web Application Module*.
2. In the *New WAM* dialog box:
 - a. Enter a *Name* of **iiiWorkingLists** (where **iii** are your initials).
 - b. Enter a *Description* of **Working List Demo**.
 - c. Leave the *Layout Weblet* value blank.
 - c. Click the *Create* button to create the new WAM.
3. The LANSAs Editor will now display the WAM's RDMLX code. At this stage, it will not contain any WebRoutines.

Step 2. Add RDMLX code to the new WAM

In this step, you will add the RDMLX code to the newly created WAM.

1. Create a WebRoutine named **ListMain with a description of List Demonstration.**
2. Define a list that will contain information from the Personnel Master file.

```
Def_List Name(#emplist) Fields(#empno #givenname #surname #address1  
#phonehme) Type(*working) Entrys(*max)
```

Note: In RDMLX programs you should usually define lists using Entrys(*max). As well as having an upper limit which is only limited by the platform, lists defined in this way are dynamic and only occupy the memory required for their current number of entries.

3. Add code to the WebRoutine to populate the list with data from PSLMST.

```
Clr_List Named(#EMPLIST)  
Select Fields(#emplist) From_File(pslmst)  
Add_Entry To_List(#emplist)  
Endselect
```

4. Create a WEB_MAP and add list EMPLIST to it, so the list can be displayed on the page.

Your finished WebRoutine should appear as follows:

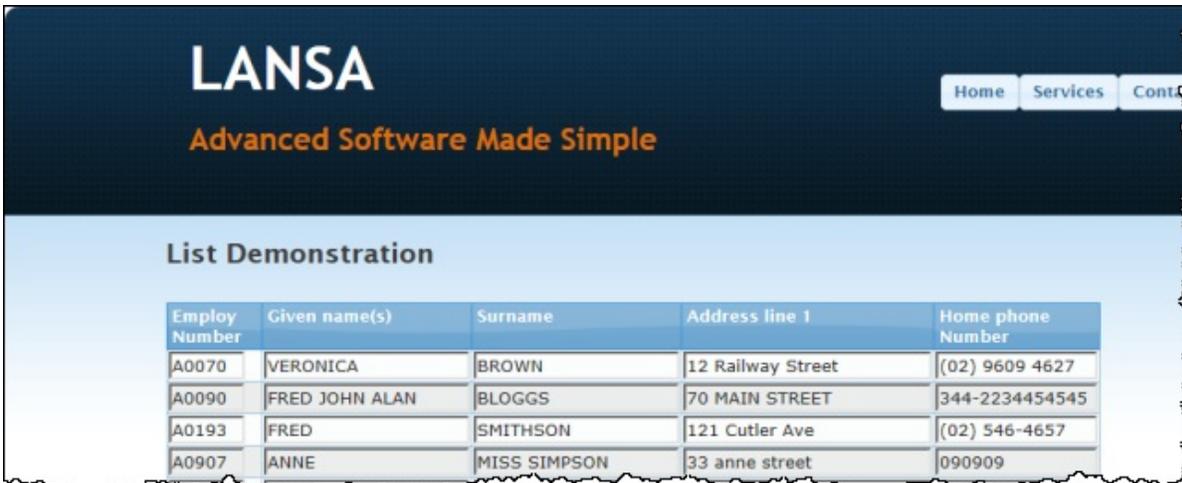
```
Webroutine Name(ListMain) Desc('List Demonstration')  
Web_Map For(*both) Fields(#emplist)  
Def_List Name(#emplist) Fields(#empno #givenname #surname #address1  
#phonehme) Type(*working) Entrys(*max)  
Clr_List Named(#EMPLIST)  
Select Fields(#emplist) From_File(pslmst)  
Add_Entry To_List(#emplist)  
Endselect  
Endroutine
```

5. Compile the WAM. When you compile a WAM, it is always saved first.

Step 3. See how the working list is displayed

In this step, you will see how the list is displayed on the page.

1. Open the WebRoutine ListMain in the Design view.
2. In the Editor, click the *Run* button on the toolbar to run the WebRoutine in the web browser.



| Employ Number | Given name(s) | Surname | Address line 1 | Home phone Number |
|---------------|----------------|--------------|-------------------|-------------------|
| A0070 | VERONICA | BROWN | 12 Railway Street | (02) 9609 4627 |
| A0090 | FRED JOHN ALAN | BLOGGS | 70 MAIN STREET | 344-2234454545 |
| A0193 | FRED | SMITHSON | 121 Cutler Ave | (02) 546-4657 |
| A0907 | ANNE | MISS SIMPSON | 33 anne street | 090909 |

Notice all of the fields in the list are input fields. When you define the fields in the DEF_LIST statement, you can specify the display mode just like in the WEB_MAP statement.

Step 4. Change the display mode of fields in the list

In this step, you will change the display mode of the fields in the list.

1. Change the field definitions in the DEF_LIST command to output and make PHONEHME hidden.

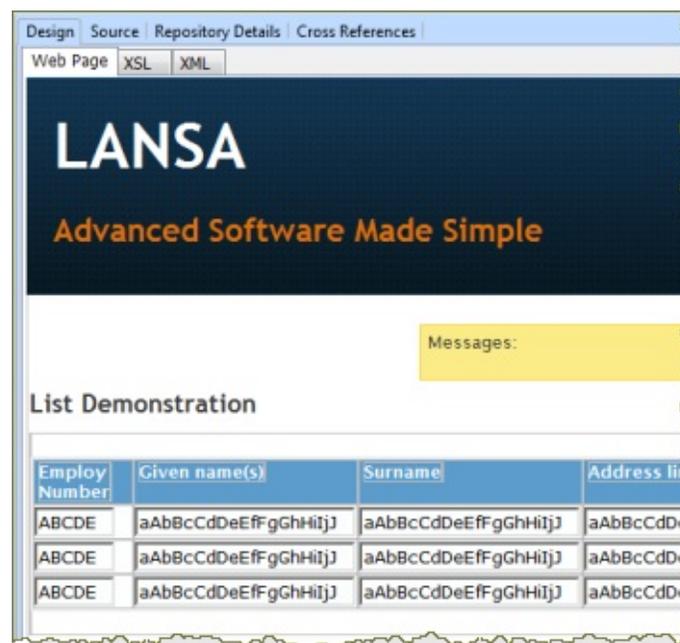
```
DEF_LIST NAME(#EMPLST) FIELDS((#EMPNO *OUTPUT) (#GIVENAM
```

Hint: You could use the *F4 Command Assistant*, and expand the *Fields* parameter, enabling you to enter an **output* attribute to each field.

If the *Assistant* is initially shown with the bottom tabs, you can use the left hand dotted bar, to drag, float and resize it. When you close the *Assistant*, the size and position is remembered.

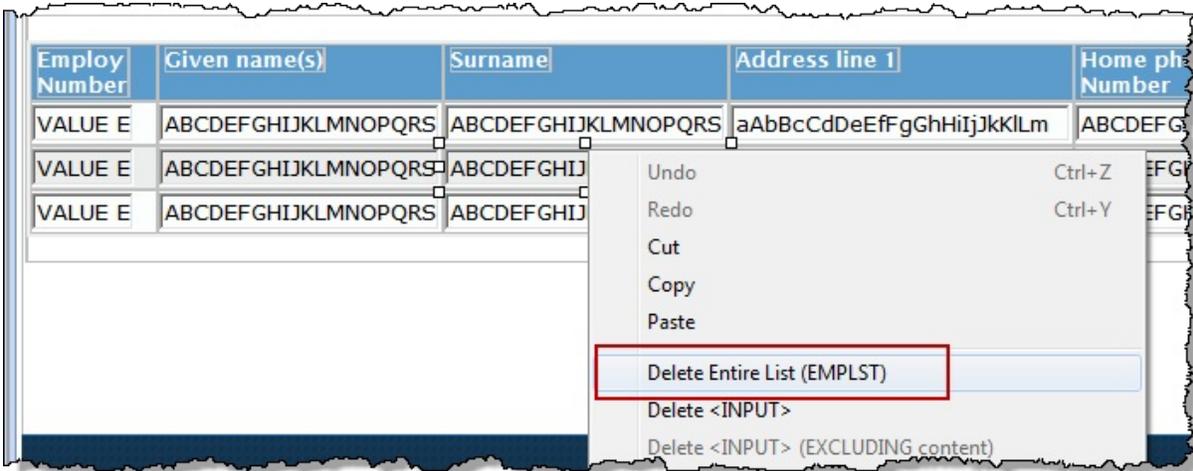
When you click in each field it will be selected. Move the cursor right, add a space and type **out*. The value **output* and **out* are synonymous.

2. Compile the WAM and open ListMain in the *Design* view.

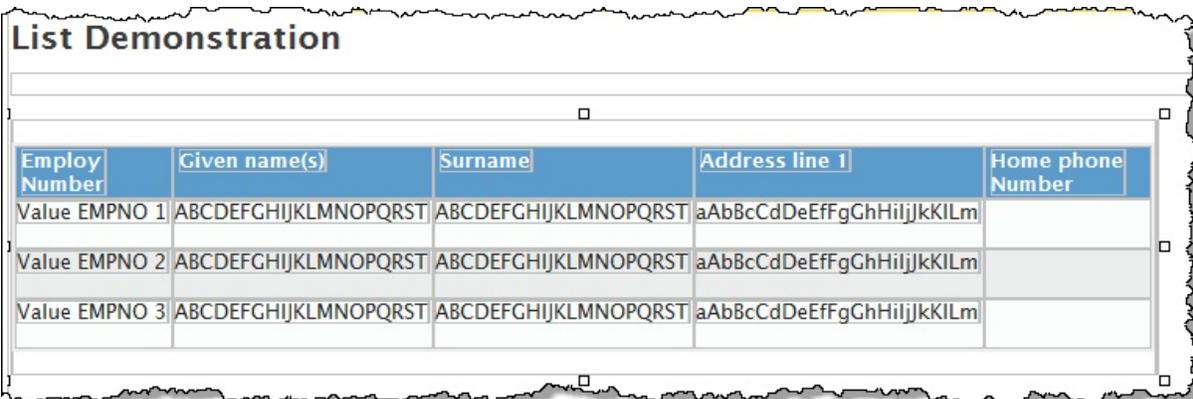


Notice all the fields are still showing as input fields. For existing WebRoutines, changes made to the WEB_MAP statements, after the first compile do not affect the page design.

3. To update the list definition, remove the list from the page and add it back:
 - a. Right click on an empty space in the list and select *Delete Entire List*.



- b. Open the WebRoutine Output view and drag the list back onto the page. Notice that the fields in the list are now correctly shown as output.
- c. Save your changes.



4. Click the *Execute* button on the toolbar to run the WebRoutine in the browser and see how the page will look.

Notice: Home phone number column is visible in the *Design* view, but is hidden when you run the WebRoutine in the browser.

Step 5. Use the generate XSL for all WebRoutines option

In this step, you change the WEB_MAP again, but instead of removing the list from the page and adding it again, you will generate the XSL for all WebRoutines.

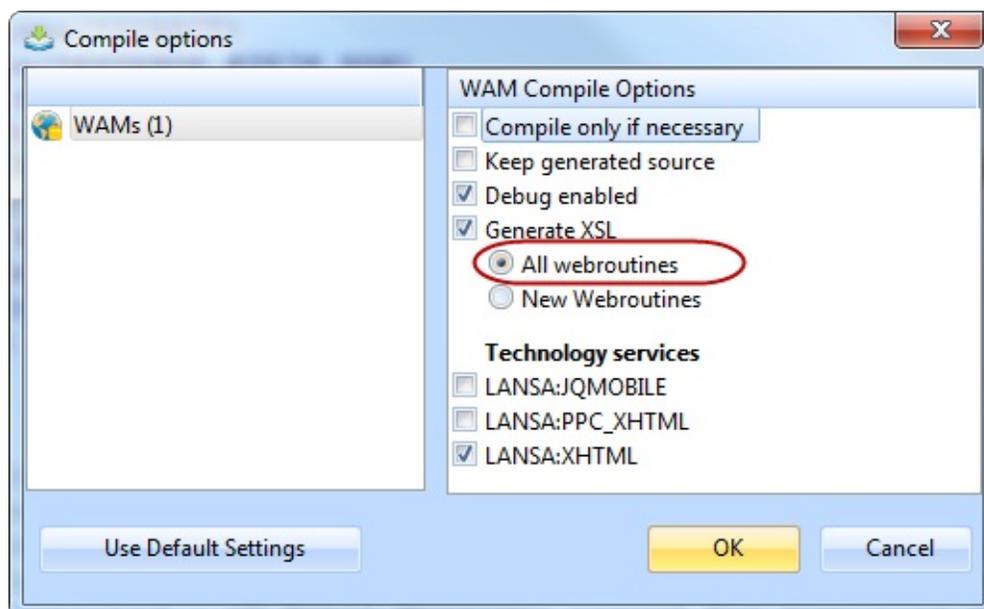
1. Change the field definitions in the DEF_LIST command to input, or do not specify a display mode so they default to input.

```
Def_List Name(#emplist) Fields(#empno #givenname #surname #address1  
#phonehme) Type(*working) Entries(*max)
```

2. Save your changes.
3. Open the *Compile options* dialog by clicking the *Menu* button on *Compile* section of the *Home* ribbon.



4. Make sure *Generate XSL* is checked and check *All WebRoutines*.



5. Click *OK*.
6. Open ListMain in the Design view again. Notice the list fields are all input capable again.

Remember: When you generate the XSL for All WebRoutines, changes you have made to **ANY** WebRoutines in the WAM will be lost.

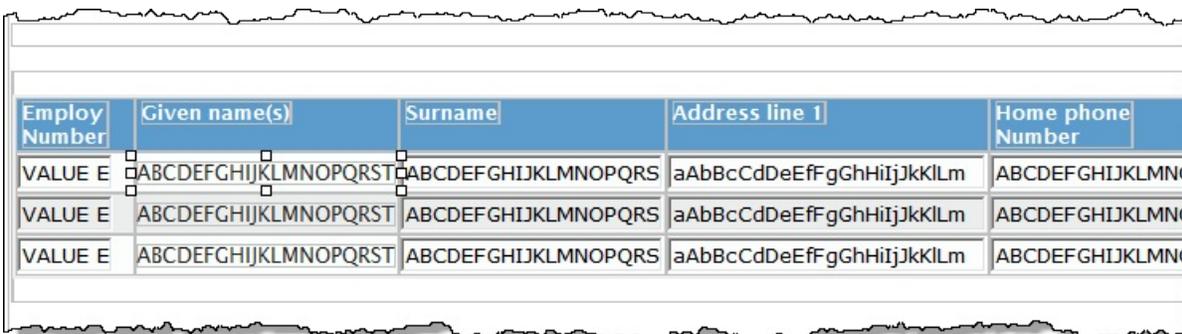
Step 6. Modify the list in the Design view

In this step, you will make changes to the list inside the Design view.

Change the field definitions in the Design view by right clicking on the field in the list and selecting one of *Change to Input Field*, *Change to Output Field*, or *Change to Hidden Field*. When you make this change in the *Design* view, the field attribute is not modified in the RDMLX.

1. Change GIVENAME to an output field, by right clicking on the field and selecting *Change to Output Field*.

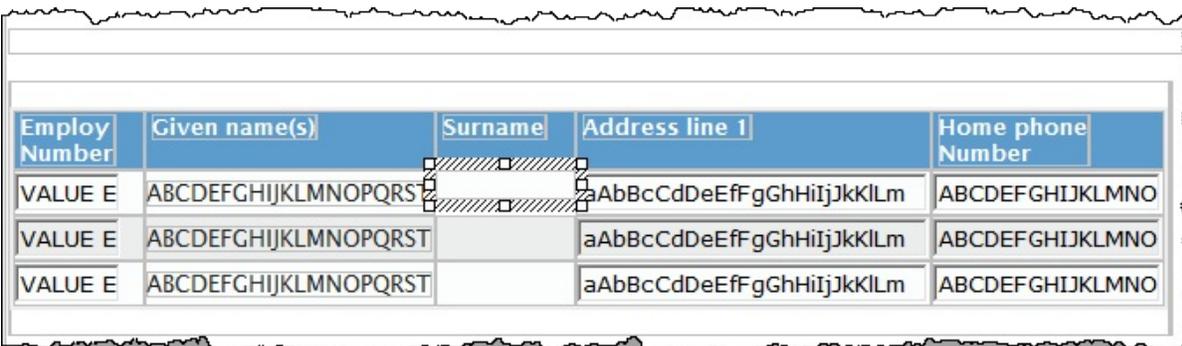
After making the change, the list in the Design view will appear something like the following:



| Employ Number | Given name(s) | Surname | Address line 1 | Home phone Number |
|---------------|--------------------|--------------------|---------------------------|-------------------|
| VALUE E | ABCDEFGHIJKLMNQRST | ABCDEFGHIJKLMNQRST | aAbBcCdDeEffgGhHiIjJkKlLm | ABCDEFGHIJKLMNO |
| VALUE E | ABCDEFGHIJKLMNQRST | ABCDEFGHIJKLMNQRST | aAbBcCdDeEffgGhHiIjJkKlLm | ABCDEFGHIJKLMNO |
| VALUE E | ABCDEFGHIJKLMNQRST | ABCDEFGHIJKLMNQRST | aAbBcCdDeEffgGhHiIjJkKlLm | ABCDEFGHIJKLMNO |

2. Change SURNAME to a hidden field, by right clicking on the field and selecting *Change to Hidden Field*.

After making the change, the list in the Design view will appear something like the following:



| Employ Number | Given name(s) | Surname | Address line 1 | Home phone Number |
|---------------|--------------------|---------|---------------------------|-------------------|
| VALUE E | ABCDEFGHIJKLMNQRST | | aAbBcCdDeEffgGhHiIjJkKlLm | ABCDEFGHIJKLMNO |
| VALUE E | ABCDEFGHIJKLMNQRST | | aAbBcCdDeEffgGhHiIjJkKlLm | ABCDEFGHIJKLMNO |
| VALUE E | ABCDEFGHIJKLMNQRST | | aAbBcCdDeEffgGhHiIjJkKlLm | ABCDEFGHIJKLMNO |

3. Save your changes to WebRoutine LISTMAIN.
4. Now change Surname to an output field.

Since SURNAME is hidden, you will not immediately be able to select it in

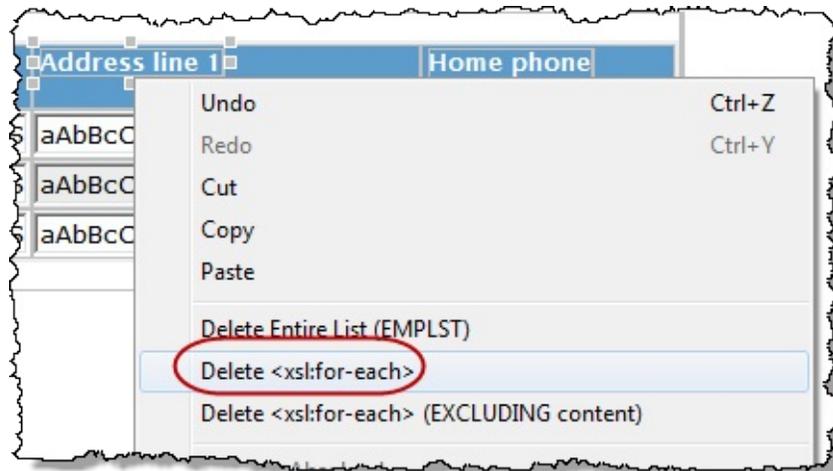
the *Design View*. To change it, take the following steps:

- a. Select the *Outline* tab.
- b. If necessary, expand the tree view so that field SURNAME is shown. Select SURNAME.
- c. Select the Details tab. This will show the properties for SURNAME.
- d. On the *Details* tab, change the Type property to text. Select this value from the dropdown list.



The Surname column will be show as input. You may need to re-open the WebRoutine in the *Design* view to refresh the list's appearance.

5. Change the caption for the ADDRESS1 column.
 - a. Select the caption text *Address Line 1*. You should see grips around the selection.
 - b. Delete the Address Line 1 heading text using the Delete key
 - c. Alternatively you could use the right mouse context menu as shown in the picture following. This shows you are deleting the `<xsl:for-each>` that outputs this HTML element.



- d. Type **Street Address** where the old caption used to be.
6. Set the vertical alignment for the cell:
 - a. Place the cursor in the cell where you entered the "Street Address" caption.
 - b. Open the *Details View*.
 - c. Set the *vAlign* property to **bottom**.
7. Save your changes. Execute the WAM in the browser to test your changes.

Summary

Important Observations

- Hidden list columns are visible in *Design View*, but are not shown when run in the browser.
- When modifying a column in a list on a page, the resulting modifications are applied to all rows in this column.

Tips & Techniques

- Lists can also be placed on the page by dragging and dropping a list from the WebRoutine Output view.

What I Should Know

- How lists are rendered in the web page.
- How to customize the columns in a lists.

WAM020 - WAM Navigation

Objectives

- To demonstrate navigation between WebRoutines within the RDMLX code.

In this exercise, you will create a WAM that will look up an employee's information based on an employee number. This WebRoutine will use re-entrant programming.

You will then create two more WebRoutines that will do the same lookup, but will be executed differently from the main WebRoutine. This will show you different ways to navigate through WebRoutines within the RDML, as well as demonstrate the fact that there are many different ways to accomplish the same task.

To achieve this Objective, you will complete the following:

- [Step 1. Create a new WAM](#)
- [Step 2. Add RDMLX code to the new WAM](#)
- [Step 3. Add Buttons and the Dropdown list to the WebRoutine](#)
- [Step 4. Test and Understand the WebRoutine](#)
- [Step 5. Add Weblet to a List](#)
- [Summary](#)

Before you Begin

In order to complete this exercise, you should have completed the following:

- [WAM005 - Create Your First WAM](#)
- [WAM010 - Using WEB_MAPs](#)
- [WAM015 - Working Lists](#)

Step 1. Create a new WAM

In this step, you will create a new Web Application Module that you will use to learn the CALL and TRANSFER statements.

1. In the LANSAs Editor window, click the *New button* and choose *Web Application Module*.
2. In the *New WAM* dialog box:
 - a. Enter a *Name* of **iiiNavigation** (where **iii** are your initials).
 - b. Enter a *Description* of **Navigation Demo**.
 - c. Leave *Layout Weblet* blank.
 - d. Click the *Create* button to create the new WAM.
3. The LANSAs Editor will now display the WAM's RDMLX code. At this stage, it will not contain any WebRoutines.

Step 2. Add RDMLX code to the new WAM

In this step, you will add RDMLX code, defining three WebRoutines.

1. Immediately following the **BEGIN_COM**, insert the following RDML code to create three new WebRoutines named NavMain, NavCall and NavTrans:

```
Webroutine Name(NavMain) Desc('Navigation Home')
Endroutine
Webroutine Name(NavCall) Desc('Navigation CALL')
Endroutine
Webroutine Name(NavTrans) Desc('Navigation TRANSFER')
EndroutineWebRoutineWebRoutine
```

2. Define a List containing EMPNO in WebRoutine NavMain. This list will hold all of the valid employee numbers from the personnel master. You will use this list to populate a dropdown list, so you will only be able to search for an employee that exists. Add the following code in the WebRoutine:

```
Def_List Name(#EMPLST) Fields(#EMPNO) Type(*WORKING)
Entrys(*MAX)
Clr_List Named(#EMPLST)
Select Fields(#EMPNO) From_File(PSLMST)
Add_Entry To_List(#EMPLST)
Endselect
```

3. Add a **GROUP_BY** immediately following the **BEGIN_COM**. This will be used in each WebRoutine's **WEB_MAP** and **FETCH** commands and to reset field values to default values.

```
Group_By Name(#empdata) Fields(#surname #givenname #address1 #address2
#address3 #postcode #phonehme #phonebus #deptment #section #salary
#startdte #termdate)
```

Hint: Use the *F4 Command Assistant* to define the **Group_By**. Use the *Fields in File* tab to select fields from file PSLMST.

4. All fields in the WebRoutine NavMain will be both incoming and outgoing, so they will be specified **FOR(*BOTH)**. All of the fields from PSLMST will be used.

By default, all the fields will be displayed as input fields.

The **EMPLST** list should be mapped with a ***PRIVATE** attribute, because it

will be used to populate the combo box weblet for EMPNO.

Add the following WEB_MAP statement to the WebRoutine NavMain:

```
WEB_MAP FOR(*BOTH) FIELDS(#EMPNO #EMPDATA (#EMPLST *PRI
```

- Fields and lists with a *PRIVATE attribute will not be shown on the page and will not be mapped back into the WebRoutine.
 - The fields defined in the Group_by will be used to output the data that was retrieved from the file.
5. The field STDREENTRY will be used to determine which logic to execute. It should be mapped as a hidden field into and out of each WebRoutine.

Immediately below the BEGIN_COM add the following WEB_MAP.

This is a global WEB_MAP which applies to all WebRoutines

```
Web_Map For(*both) Fields((#stdreentry *hidden))
```

6. In WebRoutine NavMain set up a **CASE loop** using the field **STDREENTRY** which will determine how to perform the search. Add this code **before** the CLR_LIST and SELECT logic you added earlier.

```
Case Of_Field(#STDREENTRY)
When Value_Is('= A')
Fetch Fields(#EMPDATA) From_File(PSLMST) With_Key(#EMPNO)
Message Msgtxt("Logic executed within this WebRoutine.")
When Value_Is('= B')
Transfer Toroutine(NavTrans)
Message Msgtxt("This message will not be displayed.")
When Value_Is('= C')
Call Webroutine(NavCall)
Message Msgtxt("Back from the CALL.")
When Value_Is('= D')
Message Msgtxt("Back from the TRANSFER.")
When Value_Is('= E')
#EMPDATA := *DEFAULT
EndcaseWebRoutine
```

7. Extend your logic for handling the list of employees, EMPLIST. The list needs to be built every time WebRoutine NavMain executes, but also needs to

preserve the current value of employee number, EMPNO.

- a. Define a work field EMPNOW based on EMPNO
- b. Save current value of EMPNO in EMPNOW
- c. After the list is built restore EMPNO to EMPNOW, provided EMPNOW is not blank.

Your code should look like the following:

```
Define Field(#empnow) Reffld(#empno)
#empnow := #empno
Clr_List Named(#EMPLST)
Select Fields(#EMPNO) From_File(PSLMST)
Add_Entry To_List(#EMPLST)
Endselect
If (#empnow *NE *blanks)
#empno := #empnow

Endif
```

Your WAM at this stage should appear as follows:

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_WAM)
Web_Map For(*both) Fields((#stdrentry *hidden))
Group_By Name(#empdata) Fields(#surname #givenname #address1 #address2
#address3 #postcode #phonehme #phonebus #deptment #section #salary
#startdte #termdate)
Webroutine Name(NavMain) Desc('Navigation Home')
Web_Map For(*both) Fields(#empdata (#emplst *private))
Def_List Name(#EMPLST) Fields(#EMPNO) Type(*WORKING)
Entrys(*MAX)
Case Of_Field(#STDREENTRY)
When Value_Is('= A')
Fetch Fields(#EMPDATA) From_File(PSLMST) With_Key(#EMPNO)
Message Msgtxt("Logic executed within this WebRoutine.")
When Value_Is('= B')
Transfer Toroutine(NavTrans)
Message Msgtxt("This message will not be displayed.")
When Value_Is('= C')
```

```

Call Webroutine(NavCall)
Message Msgtxt("Back from the CALL.")
When Value_Is('= D')
Message Msgtxt("Back from the TRANSFER.")
When Value_Is('= E')
#EMPDATA := *DEFAULT
Endcase
*
Define Field(#empnow) Reffld(#empno)
#empnow := #empno
Clr_List Named(#EMPLST)
Select Fields(#EMPNO) From_File(PSLMST)
Add_Entry To_List(#EMPLST)
Endselect
If (#empnow *NE *blanks)
#empno := #empnow
Endif
Endroutine
Webroutine Name(NavCall) Desc('Navigation CALL')
Endroutine
Webroutine Name(NavTrans) Desc('Navigation TRANSFER')
Endroutine

End_ComWebRoutineWebRoutineWebRoutineWebRoutine

```

Note: It is necessary to rebuild the list of employees (EMPLST) every time the NavMain WebRoutine is executed in order to populate the dropdown list of employee numbers. There are better ways to handle this, which will be covered in later exercises.

10. Create the WEB_MAP for NavCall.

The WEB_MAP will be very similar to the one in NavMain:

- However EMPNO is the only field that needs to be taken as input.
- All the employee fields need to be output from the WebRoutine. Use the GROUP_BY to map the employee fields.
- Note that you have a global WEB_MAP for the field STDREENTRY which applies to every WebRoutine.
- The field's display modes do not matter since this WebRoutine will not

actually display anything.

Add the following WEB_MAP statements to the WebRoutine:

```
Web_Map For(*BOTH) Fields(#EMPNO)
Web_Map For(*OUTPUT) Fields(#EMPDATA)
```

11. Add the FETCH to NavCall.

```
Fetch Fields(#EMPDATA) From_File(PSLMST) With_Key(#EMPNO)
```

Your finished NavCall WebRoutine should appear as follows:

```
Webroutine Name(NavCall) Desc('Navigation CALL')
Web_Map For(*BOTH) Fields(#EMPNO)
Web_Map For(*OUTPUT) Fields(#EMPDATA)
Fetch Fields(#EMPDATA) From_File(PSLMST) With_Key(#EMPNO)
Endroutine
```

12. Create the WEB_MAP for WebRoutine NavTrans. These WEB_MAPs will be identical to NavCall.

```
Web_Map For(*BOTH) Fields(#EMPNO)
Web_Map For(*OUTPUT) Fields(#EMPDATA)
```

13. Add the FETCH, set STDREENTRY to 'D' and TRANSFER back to NavMain.

```
Fetch Fields(#EMPDATA) From_File(PSLMST) With_Key(#EMPNO)
#STDREENTRY := 'D'
Transfer Toroutine(NavMain)
```

Your finished NavTrans WebRoutine should appear as follows:

```
Webroutine Name(NavTrans) Desc('Navigation TRANSFER')
Web_Map For(*BOTH) Fields(#EMPNO)
Web_Map For(*OUTPUT) Fields(#EMPDATA)
Fetch Fields(#EMPDATA) From_File(PSLMST) With_Key(#EMPNO)
#STDREENTRY := 'D'
Transfer Toroutine(NavMain)
Endroutine
```

14. Save and compile the WAM.

Step 3. Add Buttons and the Dropdown list to the WebRoutine

In this step, you will add a row to the fields table and then add the buttons that will reload the WebRoutine. The NavMain WebRoutine is reentrant, that is, it calls itself.

1. Open NavMain in the Design view.
2. Delete the "Employee Number" caption by highlighting it and pressing *Delete* on the keyboard.
3. Move the EMPNO field to the top left cell, where you just deleted the caption.
 - a. Click on the EMPNO field, it will be selected with "grips" around it.
 - b. Drag the field to the top leftmost cell (where the "Employee Number" caption was deleted from in the previous step).
4. Add and configure the combo box weblet:
 - a. In the *Design View*, drag a *Combo Box* weblet from the *Favorites/Weblet Templates* tab and drop it on top of the EMPNO field.
 - b. Set up the combo box properties as:

| Property | Value |
|-----------------|---------------|
| listname | EMPLST |
| codefield | EMPNO |
| captionfield | EMPNO |

- c. Notice the *name* and *value* properties are already set. By dragging the weblet onto the field, the weblet will inherit the name and value of the field.
5. Change the *colspan* property of the top leftmost cell to 2.
Your design should now look like the following:

Navigation Home

ABCDE ▾

| | |
|------------------------|------------------------|
| Employee Surname | ABCDEFGHIJKLMNOPS |
| Employee Given Name(s) | ABCDEFGHIJKLMNOPS |
| Street No and Name | aAbBcCdDeEfGhHiIjKkLlM |
| Suburb or Town | aAbBcCdDeEfGhHiIjKkLlM |
| State and Country | aAbBcCdDeEfGhHiIjKkLlM |
| Post / Zip Code | 123456 |

6. Fields Department code and Section may have combo box field visualizations defined in the Repository. If necessary use the context menu to change these to *Replace with input field*.
7. Add four buttons to the top row of the table (to the right of the combo box):
 - a. From the *Weblet Templates* tab, drag and drop four *Push button* weblets into the top row. Insert a space between each of the weblets.
 - b. Set the push button *caption* properties to **Search**, **Transfer Search**, **Call Search** and **Clear**. Adjust the width of each button if necessary.
 - c. Set the *on_click_wrname* property to **NavMain** for each button.
 - d. Select each push button, and click the Ellipsis button for the *submitExtraFields* property. Set the *Name* column to **STDREENTRY** and set the *Value* column to the correct literal value. Review your WebRoutine NavMain RDMLX code to find the required values.
8. Save your changes.

Step 4. Test and Understand the WebRoutine

In this step, you will test the WebRoutine, to understanding how it works.

1. NavMain should still be open in the *Design* view, if not, open it.
2. Click the Execute button on the toolbar to run the WebRoutine in the browser.
3. Test the buttons.

You will notice that all three searches function identically. The only way to differentiate them is the messages they generate.



The screenshot shows a web form titled "LANSA Advanced Software Made Simple" with a sub-header "Navigation Home". The form includes a dropdown menu with "A3564" selected, followed by "Search", "Transfer Search", "Call Search", and "Clear" buttons. Below these are input fields for "Employee Surname", "Employee Given Name(s)", "Street No and Name", "Suburb or Town", "State and Country", "Post / Zip Code" (with a "0" in the first box), and "Home Phone Number".

Also notice the message *This message will not be displayed*, is not displayed. At the time of a TRANSFER, control is passed to another WebRoutine and the current WebRoutine is terminated.

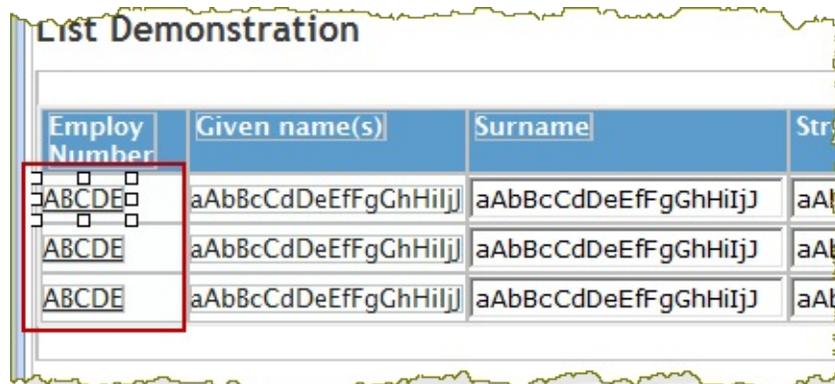
With a CALL, the current WebRoutine will still be executing and waiting for control to be returned from the WebRoutine it had called.

Step 5. Add Weblet to a List

In this step, you will use the WAM created in exercise WAM015 (iiiWorkingLists) and add an Anchor weblet to the EMPNO field in the list. You will set up this link to call the search in iiiNavigation.

If you leave iiiNavigation open in the editor, when setting the *on_click_xxxx* properties in the following steps, you will be able to select the WAM name and WebRoutine name from a dropdown list.

1. Open WAM iiiWorkingLists in the LANSa editor.
2. Open ListMain in the Design view.
3. This step will add an anchor weblet to the list and configure its properties:
 - a. From the *Weblet Templates tab*, drag and drop the *Anchor* weblet onto the EMPNO field.



| Employ Number | Given name(s) | Surname | Str |
|---------------|----------------------|----------------------|-----|
| ABCDE | aAbBcCdDeEfFgGhHiIjJ | aAbBcCdDeEfFgGhHiIjJ | aA |
| ABCDE | aAbBcCdDeEfFgGhHiIjJ | aAbBcCdDeEfFgGhHiIjJ | aA |
| ABCDE | aAbBcCdDeEfFgGhHiIjJ | aAbBcCdDeEfFgGhHiIjJ | aA |

Notice the anchor is applied to every row in the table.

- b. Set the *currentrowhfield* property to **EMPNO**.
- c. Set the *currentrownumval* property to **\$EMPNO**.
- d. Set the *reentryvalue* property to **A**.
- e. Set the *on_click_wamname* property to **iiiNavigation**.
- f. Set the *on_click_wrname* property to **NavMain**.

This will call NavMain with a reentry value of 'A' (remember, that is the local search) and will also pass the employee number of the clicked link.

Note: You can display help text for weblet properties:



Currentrowfield is the name of the field that will contain the current row's specified value.

Currentrownumval is the value of the field specified in the currentrowfield property. To specify a field value in a list use the syntax \$FIELDNAME. These properties enable the WebRoutine called via the on_click_wrname property to process values for the current row.

4. Save and run your modified WebRoutine LISTMAIN in a browser. Test the link to iiiavigation / NavMain. The employee details for the employee number selected in iiiWorkingLists should be displayed.

You can use the browser back button to return to iiiWorkingLists.

Summary

Important Observations

- You can navigate to WebRoutines in the RDMLX as well invoking them from the web page.
- TRANSFER leaves the current WebRoutine.
- CALL returns control to the calling WebRoutine after completion.
- A field's input box can be replaced with a weblet, to display the field values differently, just by dragging and dropping.
- The *Combo box* weblet can be used to display a dropdown list, which uses specified list entries to populate the dropdown items.
- The list that populates the *Combo box* is specified via weblet properties, via the *Details View*.
- The *Combo box* weblet can be made to invoke a WebRoutine on selection change.

Tips & Techniques

- Not all lists have to be represented as browse lists. You can use a list for other purposes, as in the case of this page, which uses EMPLST to populate a dropdown list.
- Any field input boxes can be replaced with weblets, just by dragging and dropping a weblet onto them.

What I Should Know

- How the TRANSFER and CALL commands work.
- There is more than one way to go about developing an application. Depending on the situation, certain methods work better than others. You should always consider your options and try to determine which method will work best for the given task.
- How to replace fields with weblets.
- How to attach lists to a weblet using its listname property.

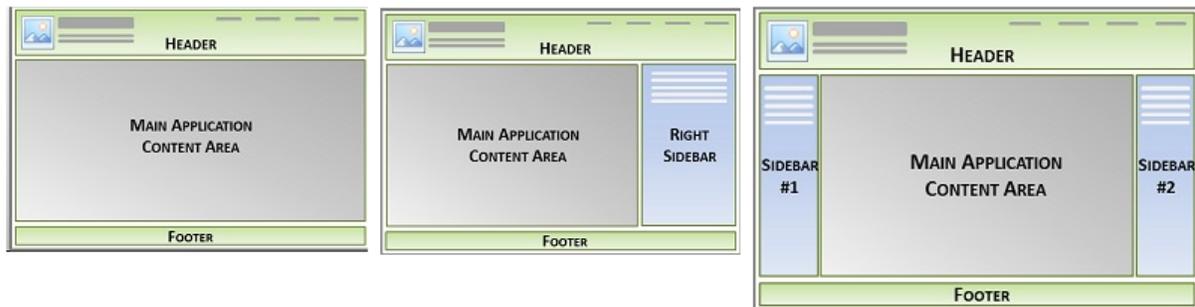
WAM025 - Using the Layout Wizard

Objectives

To create a WAM layout using the Layout Wizard.

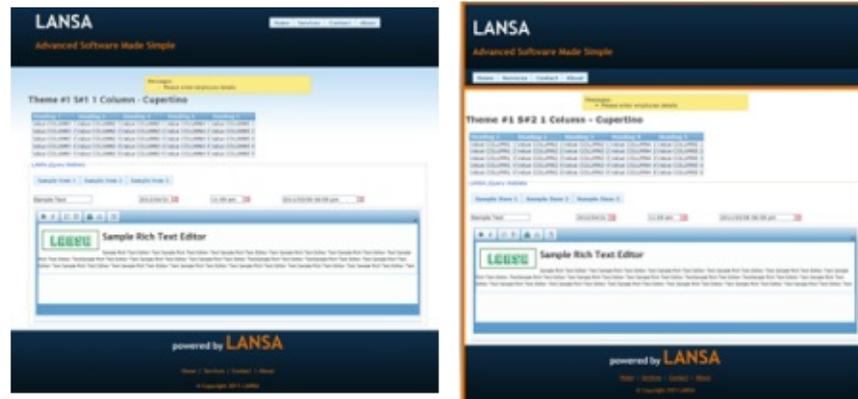
The Web Application Layout Manager Wizard enables you to define your own layout that can then be used for each WAM you create. The wizard can create a layout based one of three shipped designs which can adopt one of two styles. Each layout can also adopt a theme that controls the color scheme the layout will have. Once you have defined your own layout, it could be modified to meet your company's requirements.

Your layout may have one main area (the Main application Content Area) or include one or two sidebars as shown below:



A layout may adopt one of two styles

- Style1 has no border between areas
- Style2 defines a different color border between areas:



Themes allow you to select one of nine color schemes to apply to your layout. Three examples are shown below:



You can also choose between **fixed** or **fluid** layout width. The fluid layout is more flexible because its content area will include scroll bars if your content does not fit within the space available. For example some of your web pages may include lists with a large number of columns.

Once you have created a layout you could start to modify its appearance or content. For example, replace the fixed text such as "LANSA Advanced Software Made Simple". You will look at layouts in more detail in a later exercise.

To create your layout you will complete the following:

- [Step 1. Use the Web Application Layout Manager Wizard.](#)
- [Step 2. Execute the generated Demo WAM](#)
- [Step 3. Examine the new layout](#)
- [Summary](#)

Before you Begin

In order to complete this exercise, you should have completed the following:

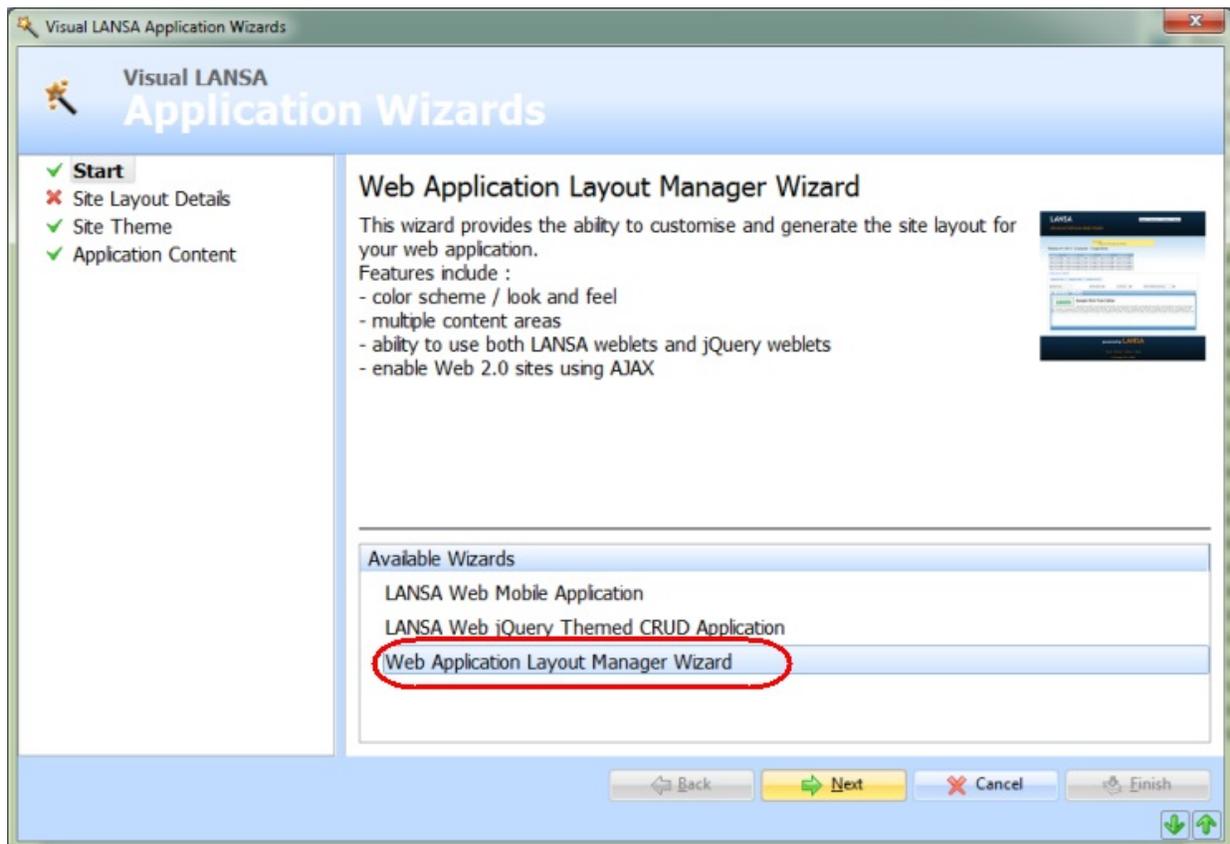
- [WAM005 - Create Your First WAM](#)
- [WAM010 - Using WEB_MAPs](#)
- [WAM015 - Working Lists](#)
- [WAM020 - WAM Navigation](#)

Step 1. Use the Web Application Layout Manager Wizard.

1. Start the wizard from the Tools ribbon in the *Utilities* grouping:



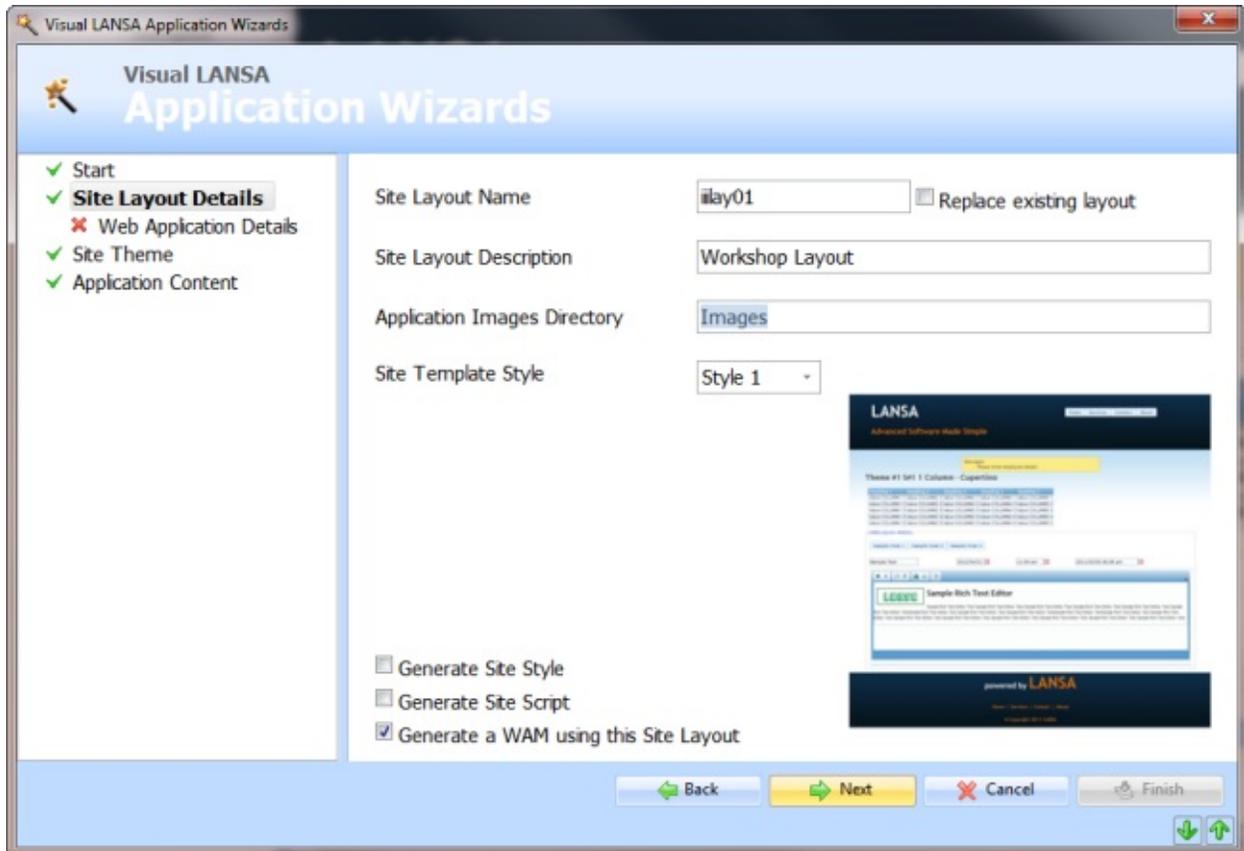
2. Select the *Web Application Layout Manager Wizard*.



3. click Next to continue.

4. Enter the following *Site Layout Details*:

- Site Layout Name: **iiilay01**
- Site Layout Description: **iii Workshop Layout**
- Select the *Sample WebRoutines* to show the themed *Weblets* option.

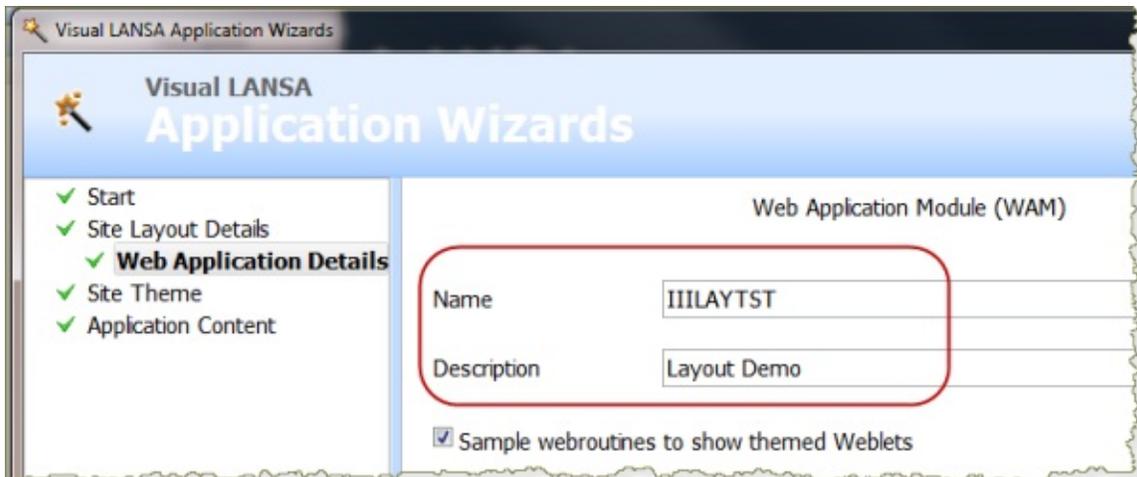


The *Application Images Directory* will use an existing folder if entered. If a new folder name is entered, this will be created in c:\Program Files (x86)\LANSA\Webserver\Images.

The new folder could be used to store application specific images.

Note that you would also need to later, set this folder up on the IBM i server.

- d. Click *Next*.
5. Enter the *Web Application Details*:
 - a. Name: **iiiLAYTST** for this exercise. In your own WAMs you may use a long name.
 - b. Description: **Layout Demo**
 - c. Select the *Sample WebRoutines to show themed Weblets* option.

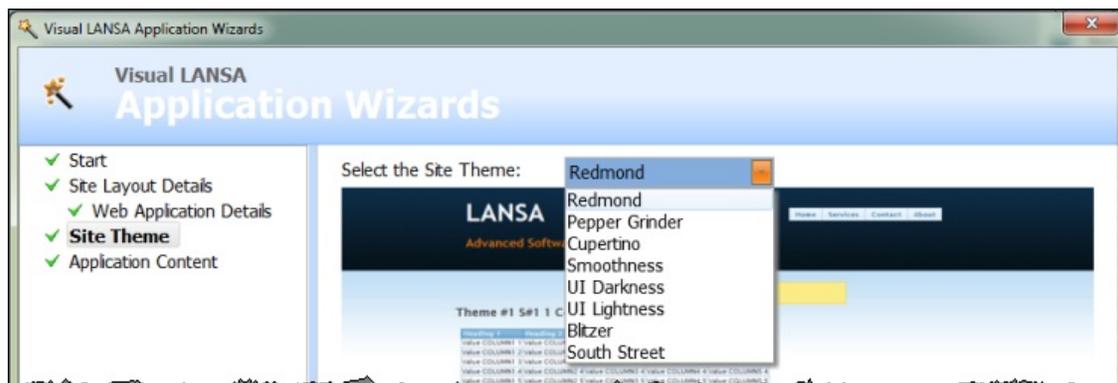


d. Click *Next*.

The Wizard creates a sample WAM for you.

6. Select one of the *Site Themes*.

The color scheme will be displayed when selected.

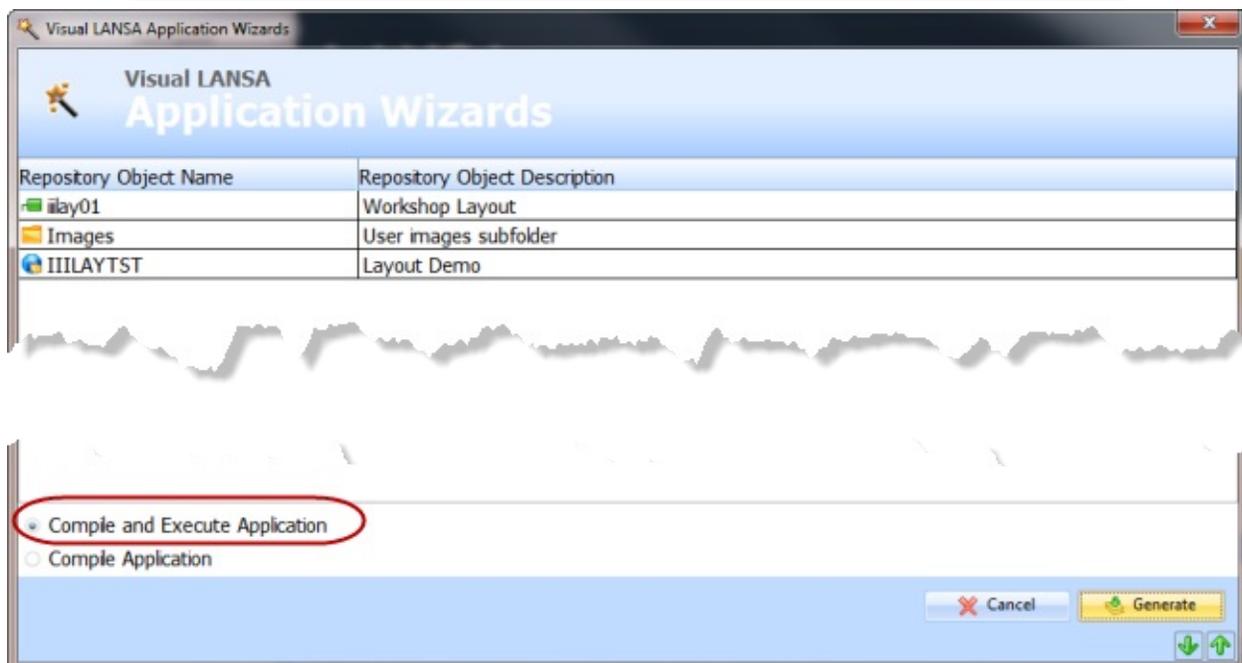
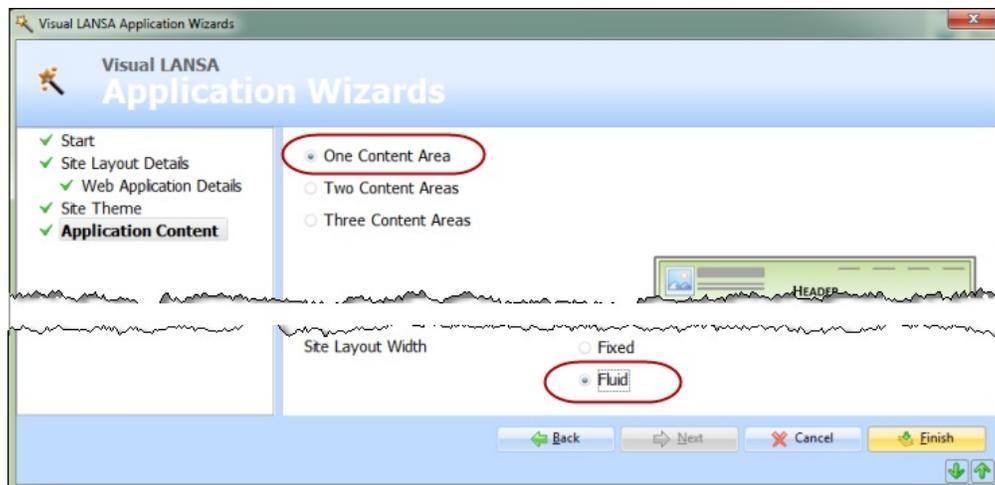


7. Click *Next*.

8. In the Application Content:

a. Select the *One Content Area*:

b. Select the **Fluid** *Site Layout Width* option.



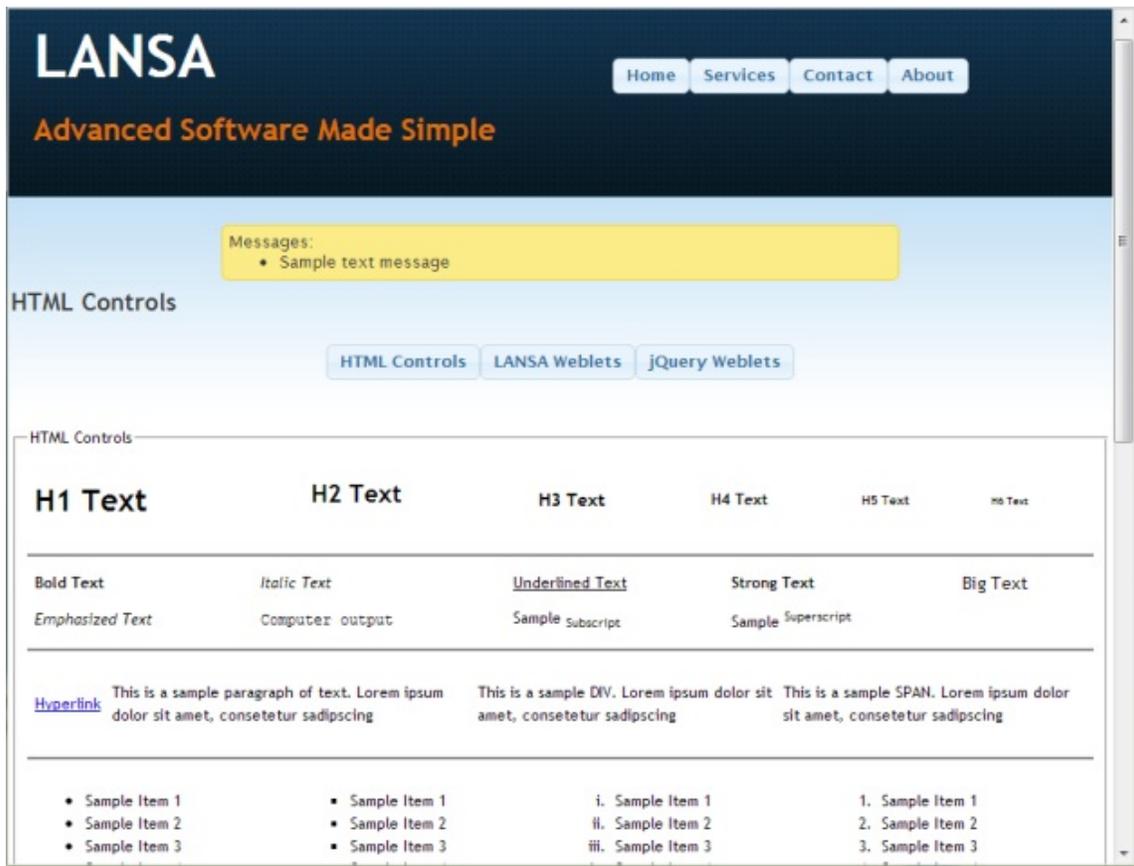
c. Click *Finish*.

9. Select the *Compile and Execute Application* option.

10. Press *Generate*.

- First, your layout **iiilay01** is generated.
- Next, your WAM is generated together with its WAM layout which references **iiilay01**.
- Third, your WAM is opened in the browser.
- **Note:** The appearance of your WAM will depend on the Theme that you

selected in the Wizard questionnaire.



- The demo WAM contains three WebRoutines that demonstrate:
 - Basic HTML controls
 - LANSA weblits
 - jQuery enabled weblits

The WebRoutines can be invoked from the menu at the top of the content area.

Step 2. Execute the generated Demo WAM

1. Open your iii Demo Layout WAM (iiiLAYTST) in the editor so that you can review its RDMLX code as you are running the WAM. If it is not already started in the browser, run the SampleHTML WebRoutine from the Design view.

Note that this page contains sample standard HTML only. The WebRoutine SampleHTML contains no web_maps or code except for a MESSAGE command.

2. Invoke the WebRoutine **SampleWeblets1** by selecting LANSA Weblets in the menu at the top of the content area.

This web page contains output for standard LANSA weblets. These have all been supported since WAMs were introduced into the Visual LANSA product in 2005. Note that they adopt the color scheme for your chosen Theme. For example the menu button, the list and tab pages.

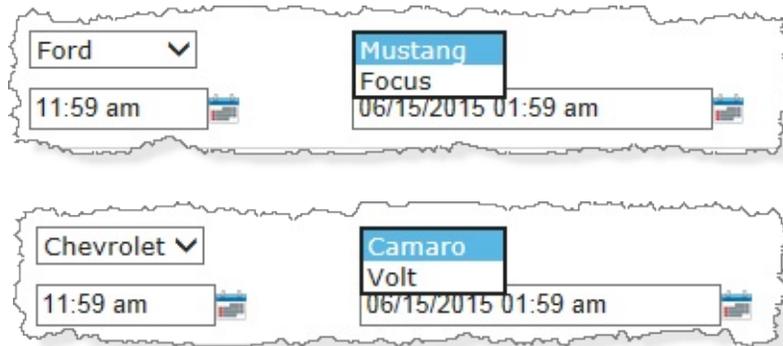
- a. Review the **SampleWeblets1** WebRoutine. This has a web_map for the list that supports the tree view weblet (car manufacturers and models). The tree view list is populated by method routine BuildSampleTree.
 - b. Note that the content for all other controls on this page has been hard coded. For example, the weblet's *items* property defines the content for the combo box and list box.
3. Invoke the WebRoutine **SampleWeblets2** by selecting the jQuery Weblets option in the menu at the top of the content area.

This page demonstrates the new weblets that were introduced in V12 SP1 (June 2011). These weblets will all be used and explained in more detail in later exercises.

A number of these new weblets are AJAX enabled, which means a single control can be refreshed from the server. For example, the list that defines the content of a Dynamic select box can be refreshed by invoking a new type of WebRoutine.

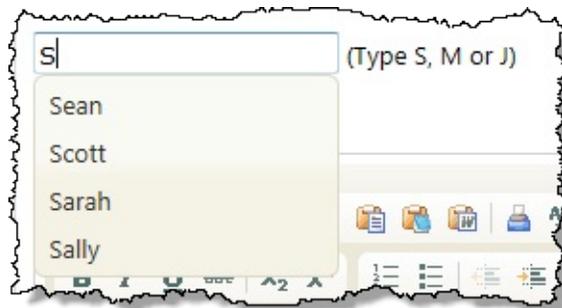
- a. Review the page content. Note that the weblets reflect the color scheme of your chosen Theme, including the more complex weblets such as the bar charts and pie charts. If possible review the appearance of other student's layouts who may have selected a different Theme.
- b. Notice that the dropdown lists for car manufacturer and car models are

dynamic.



When car manufacturer changes, the car models dropdown list is refreshed with a new list by calling a special WebRoutine. The rest of the page is unchanged.

- c. Type an appropriate letter into the AutoComplete input field.



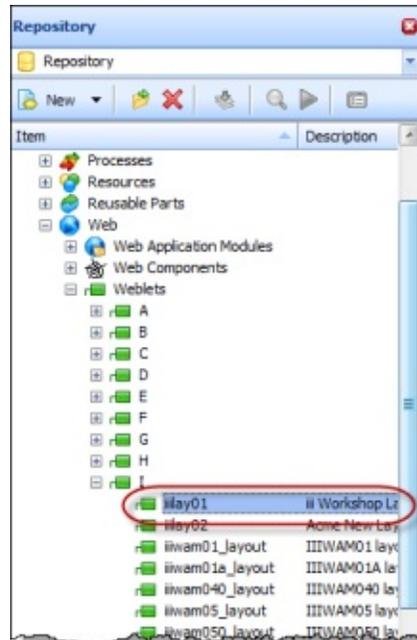
The list that is displayed, is also populated by a special response WebRoutine. Once again this is the only part of the web page that has been updated.

- d. Review the RDMLX code for the **SampleWeblets2** WebRoutine. Notice that it outputs lists for dropdown lists, charts and tree. These lists are built by invoking method routines.
- e. At this stage, we will leave an explanation of the special WebRoutines that handle the dynamic dropdown list and auto complete input field. These will be implemented in later exercises.

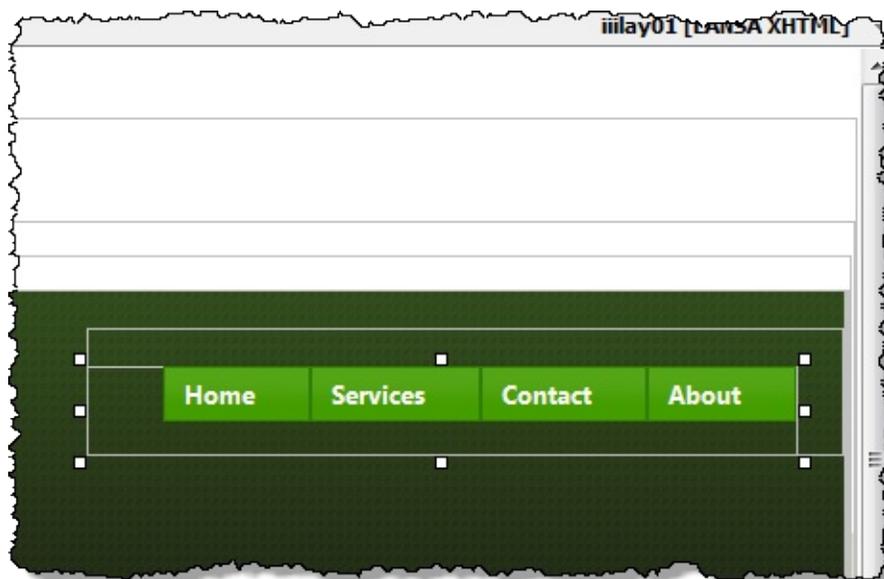
Step 3. Examine the new layout

In this step you will find your new layout in the Repository and open it in the editor.

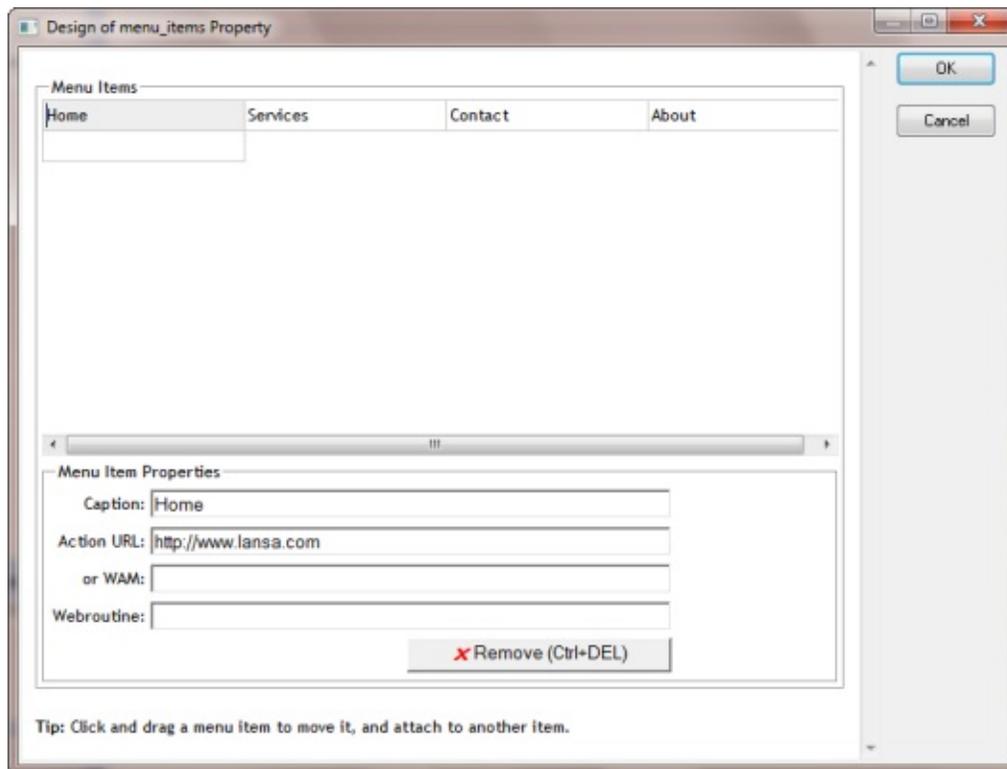
1. Layouts are special type of weblet. Expand the Web / Weblets group on the Repository tab to locate your layout. Double click on it to open it in the editor.



2. In the *Design* view, select the Menu Bar item.



3. On the *Details* tab, select the items value, and click on the Ellipsis  button to open the *Design of menu_items Property* dialog.



4. Select each menu item and define its Action URL based on the following:

Home **<http://www.lansa.com>**

Services **<http://www.lansa.com/services/index.htm>**

Contact **<http://www.lansa.com/about/contactus.htm>**

About **<http://www.lansa.com/about/index.htm>**

5. Click OK to close the dialog and save your layout.
6. Open your iii Demo Layout WAM (iiiLAYTST) in the editor and run the SampleHTML WebRoutine in the browser from the *Design* tab. Test your menu items which should open the relevant pages within the LANSA web site. Use the browser back button to return to your application web page.

This illustrates, in one simple way, how your layout can be developed and all WAM layouts based on this standard layout will adopt changes made to the

common layout.

Summary

Important Observations

The layout wizard allows you to quickly and easily create a standard layout that can be adopted when creating your application WAMs

In a later exercise you will see how this layout can be modified to match your company requirements in more detail.

The fonts and color schemes used by your layout are controlled by Cascading Style Sheet files (CSS). In a later exercise you will see how these CSS files can be edited to meet your company requirements exactly.

Tips & Techniques

Start your web application development by creating a standard layout and then specify this layout as each WAM is created.

This layout does not have to be complete in every respect. You can modify it later and all your WAMs will implement these changes.

What I Should Know

How to use the Web Application Layout Manager Wizard, to create a layout.

WAM030 - Employee Enquiry

Objectives

- To create a simple web application
- The WAM prompts the user to enter an Employee number and then following a button press, displays Employee Details.



To achieve these objectives, you will complete the following:

- [Step 1. Create Employee Enquiry WAM](#)
- [Step 2. Create a Begin WebRoutine](#)
- [Step 3. Open the Design View](#)
- [Step 4. Add a Push Button Weblet](#)
- [Step 5. Create 'Details' WebRoutine](#)
- [Summary](#)

Before You Begin

- In order to complete this exercise, you should have completed all the preceding exercises.

Step 1. Create Employee Enquiry WAM

1. In the LANSAs Editor, click the New button and choose Web Application Module. The New WAM dialogue will appear.
2. In the New WAM dialogue box enter:
 - a. Name: iiiEmpEnquiry where iii are your initials
 - b. Description: Employee Enquiry
 - c. Layout Weblet: iiilay01 – the new layout you created in exercise WAM025
 - d. Select any suitable Framework, such as Personnel & Payroll
 - e. Click the Create button to create a new WAM

Step 2. Create a Begin WebRoutine

In this step you will create the code for a new WebRoutine that will be used to enter the employee number. The WEB_MAP statement will specify the fields that are passed in and out of the WebRoutine.

1. Immediately following the BEGIN_COM, insert the following RDML code to create WebRoutine named Begin.

```
Webroutine Name(Begin) Desc('Select Employee')Endroutine
```

2. This WebRoutine will produce a web page with employee number as an input field. You will add a push button to invoke a second WebRoutine. The **Begin** WebRoutine will not require any incoming fields. It will have one outgoing field, EMPNO. Add the following WEB_MAP statement to the WebRoutine.

```
Web_Map For(*output) Fields(#empno)
```

The EMPNO field will be used to enter the employee number to be retrieved and displayed.

Remember that by default, fields defined on a WEB_MAP statement are input capable.

For further information about the WEB_MAP statement, refer to the *LANSA Technical Reference Guide*.

Note: you can press F1 on any statement in the LANSAs editor to display help directly from the LANSAs Technical Reference Guide.

Your completed WebRoutine should appear as follows:

```
Webroutine Name(Begin) Desc('Select Employee')  
Web_Map For(*output) Fields(#empno)  
Endroutine
```

3. Compile your WAM.

Step 3. Open the Design View

After compiling your WAM, you can edit the web pages for your WebRoutines.

1. In this case your WAM contains just one WebRoutine. If you select the Design tab, it will open the first WebRoutine in the Design view. You will use other methods to open a WebRoutine in the *Design* view later in this exercise.

Your web page should look like the following:



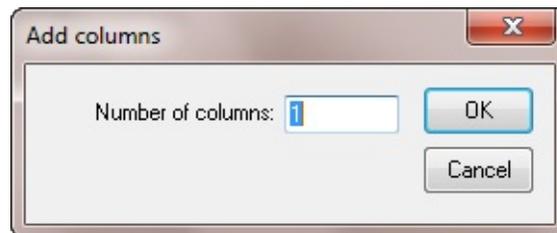
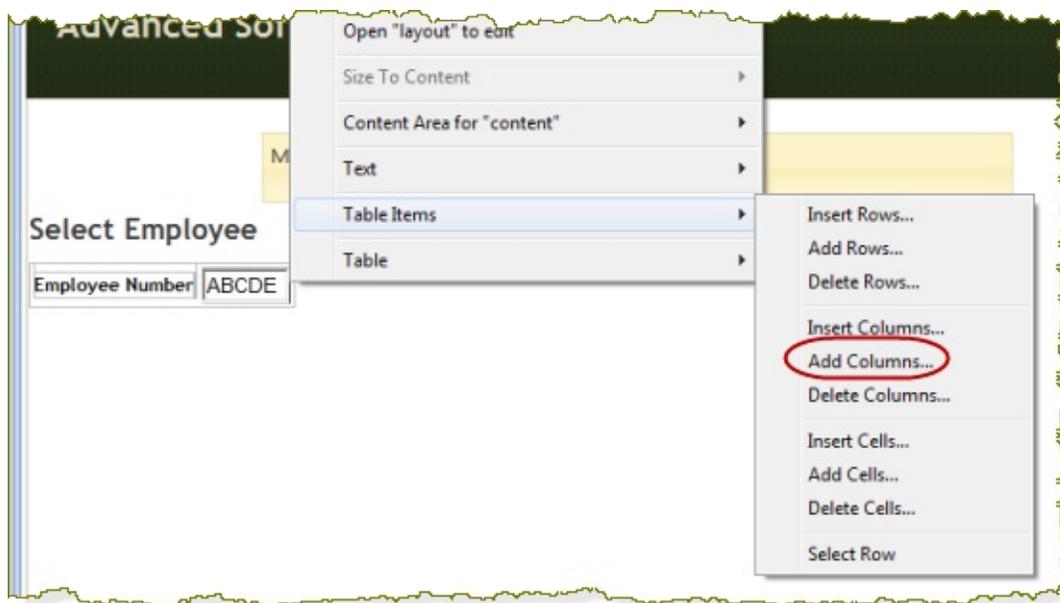
Step 4. Add a Push Button Weblet

In this step you will add a column to the table containing employee number, and drag a push button weblet into the new cell.

1. Notice that the table is shown in the editor with a double line border. This is to make the table visible for editing. The table will not have a border when you run the WebRoutine in the browser, unless you have edited the table definition and defined a border.

If you click anywhere in this table, you can use the right mouse menu (also known as the context menu) to select the Table Items menu. Hint: Select the employee number input box and move the cursor right to position into the table cell.

Select the Add columns... option to add a column(s) to the right hand side of the table.



You should now see something like this:



Note that the new column contains characters **. These are placeholders so that it is easy to drag and drop into this cell. You will later remove them.

2. On the *Favorites* tab, select the  **Weblet Templates** tab. Ensure that *Standard Weblets* are selected using the dropdown list at the top of this tab. Drag and drop a Push Button into the new cell at the right hand side of the table:



3. Use the keyboard cursor keys to position into the table cell that now contains the pushbutton and delete the two asterisks (they have done their job). For example select the push button and move right using the cursor key. You are now positioned in the table cell (you have the `<td>` tag selected) and can delete the * character.
 - a. Set focus on the push button, select the  *Details* view to show the weblet's *Properties*.
 - b. Set the *caption* of the button to **Details**. **Note:** quotes are not required.
 - c. Set the *on_click_wrname* property to Details (note that you have not yet written a WebRoutine named 'Details').

Your Design view should now look something like this:



4. Save your changes. In the *Design* view you are actually editing an XSL document. It is good practice to save your work regularly.

Step 5. Create 'Details' WebRoutine

1. Select the Source tab to return to your WAM's RDMLX code and add a new WebRoutine named **Details**.

This WebRoutine needs to:

- receive a value for the EMPNO field from the browser
- retrieve the employee record using a FETCH command from the file PSLMST,
- display values for EMPNO, GIVENAME, SURNAME, ADDRESS1, ADDRESS2, ADDRESS3, POSTCODE, PHONEHME, PHONEBUS, DEPARTMENT, SECTION, SALARY, STARTDTE and TERMDATE. All these fields should be mapped for output, at this stage you are writing an enquiry WebRoutine.

Hint #1 : Remember it's a good programming technique to use a GROUP_BY to define a set of fields. You can also use a GROUP_BY to map fields into or out of the WebRoutine. Remember to set all fields in the group as *OUTPUT to ensure they are displayed and cannot be changed (you can also use *OUT to save on typing).

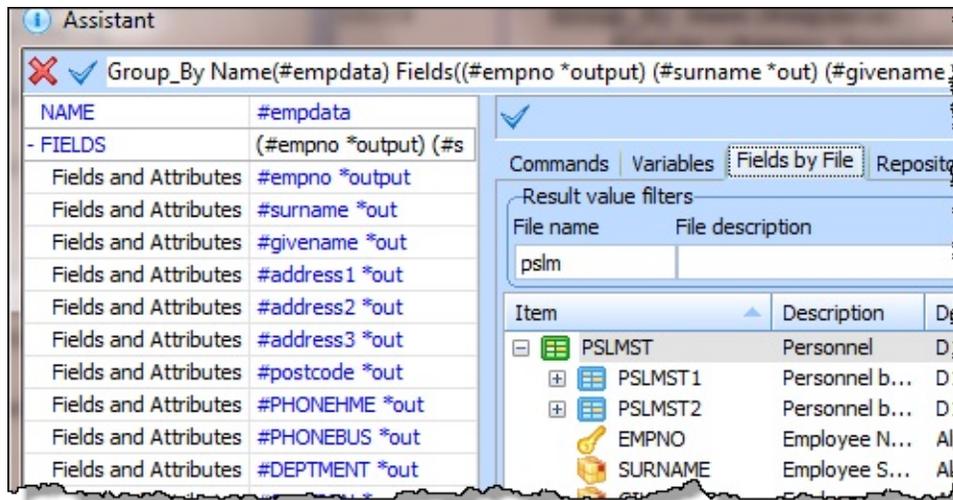
Hint #2: Always create a GROUP_BY command with the *F4 Command Assistant*. This dialog allow you to quickly select fields from a file definition. You can then also define the *out attribute against each field, while using this dialog.

To do this, click in one of the Fields and Attributes values, use the cursor key to move right, then type space, followed by *out.

Hint #3 : Think about what fields are "input" to the WebRoutine and which fields are "output" from the WebRoutine (to be displayed on the web form). This affects the For() parameter required on the web_map.

Hint #4: As with all database programming, consider how to handle an error condition. For example you could use an IF_STATUS to display an error message if the Employee record is not found. The LANSa I/O status is returned as field IO\$STS and the IF_STATUS command compares with this field.

Hint: #5: In a WAM, a validation error on an I/O command will branch to the EndRoutine, unless you have written VAL_ERROR(*next).



Your WAM code should now look like the following:

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_WAM) Layoutweblet('iilay01')
Group_By Name(#empdata) Fields((#empno *output) (#surname *out) (#givenname *out) (#address1 *out) (#address2 *out) (#address3 *out) (#postcode *out) (#PHONEHME *out) (#PHONEBUS *out) (#DEPARTMENT *out))
WebRoutine Name(begin)
Web_Map For(*output) Fields(#empno)
Endroutine
Webroutine Name(details) Desc('Employee Details')
Web_Map For(*both) Fields(#empdata)
Fetch Fields(#empdata) From_File(pslmst) With_Key(#empno) Val_Error(*none)
If_Status Is_Not(*OKAY)
Message Msgtxt('Employee not found')
Endif
Endroutine
End_Com
```

2. Compile your WAM. The changes you made to the "Begin" WebRoutine web page will be retained. XSL and XML will be built for your new WebRoutine only, as part of the compile.
3. To open the Design view for the Details WebRoutine, use the  icon in the source code. Your web page should look like the following:

Employee Details

| | |
|------------------------|-------------------------|
| Employee Number | ABCDE |
| Employee Surname | aAbBcCdDeEfgGhHijJ |
| Employee Given Name(s) | aAbBcCdDeEfgGhHijJ |
| Street No and Name | aAbBcCdDeEfgGhHijJkKlLm |
| Suburb or Town | aAbBcCdDeEfgGhHijJkKlLm |
| State and Country | aAbBcCdDeEfgGhHijJkKlLm |
| Post / Zip Code | 123456 |
| Home Phone Number | ABCDEFHGHIJKLMNO |
| Business Phone Number | ABCDEFHGHIJKLMNO |
| Department Code | ABCD |
| Section Code | AB |
| Employee Salary | 123,456,789.12 |
| Start Date (DDMMYY) | 123456 |

Hidden Content

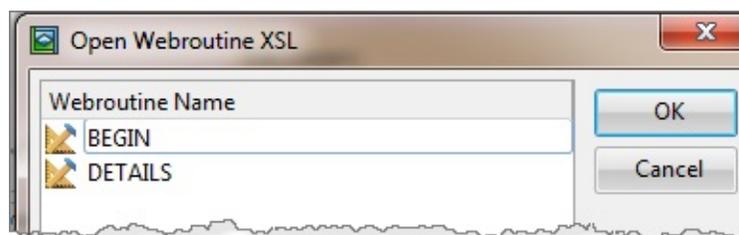
- The fields DEPARTMENT and SECTION may have a dropdown weblet visualization defined in the Repository. In this case their value will not be shown when they are output fields. You will be using field visualizations in a later exercise. If their value is not currently shown, select each in turn and use the context menu option *Change to Output Field*. This will display these fields as normal output fields.

Save your changes using the *Save*  button on the toolbar.

- On the *Design* ribbon click on the *Open WebRoutine* button.



Select the *Begin* WebRoutine and click *OK* to display it in the *Design* view.



- On the toolbar, click the  *Execute* icon to execute the WAM in the browser.. Your web page should look like the following:



Enter an Employee number (for example, A0070, A0090 or A1234) and *click* the *Details* button.

The details web page should then be displayed.



To select another employee, return to the Select Employee by using the browser back button. In later exercises, you will learn how you to add a *Return* button on the Details page to return to the Begin WebRoutine.

7. You have completed this exercise.

Summary

Important Observations

- The WEB_MAP statement allows you to specify which incoming and outgoing fields the WebRoutine maps between the web page and the WebRoutine. The WEB_MAP statement's FOR() selector specifies whether the fields are mapped as an incoming (*INPUT), outgoing (*OUTPUT), or both (*BOTH).
- Only fields in a WEB_MAP statement with a FOR(*OUTPUT) or FOR(*BOTH) can be placed on the web page.
- Acceptable field attributes are *INPUT and *OUTPUT (also *in and *out). If unspecified, the default is *INPUT. These attributes determine whether the field accepts input, as input text box, or only displays output.

Note an important distinction here:

- The WEB_MAP FOR() parameter determines whether fields and lists are mapped into the WebRoutine, out of the WebRoutine or in both directions.
- A field's *INPUT or *OUTPUT attribute determines whether the field will be input capable on the page (the default) or output only.

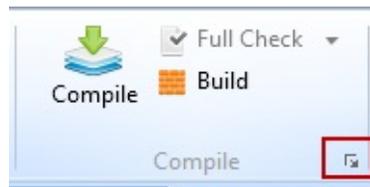
Note also that fields with an *output attribute cannot be mapped back into a WebRoutine.

- You can open a WebRoutine in the *Design* view using the  icon in the RDMLX source
- You can drag and drop fields and lists marked FOR(*OUTPUT) or FOR(*BOTH) onto your page at any time from the  Webroutine Output tab.

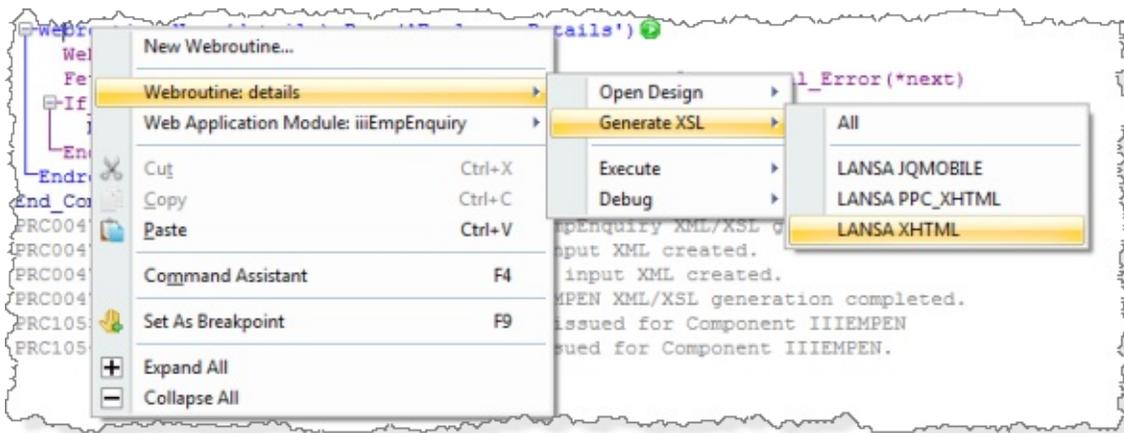
Tips & Techniques

- Weblet properties can be assigned string literals or XPath expressions. The XSL processor then evaluates the XPath expression. XPath expressions accept general logical comparison operators <, >, != <=, >= etc., as well as mathematical operators *,/,+,- etc.
For a quick reference to XPath see:
www.mulberrytech.com/quickref/XSLT_1quickref-v2.pdf.
For a more detailed reference, see www.w3schools.com.
- With the WAM open in the editor, you can compile from the *Home* ribbon or open the *Compileoptions* dialog from the *Menu* button on the *Compile* group

on the *Home* ribbon.



- You can also use a context menu by selecting a WAM in the *Repository* tab or *Favorites/Last Opened* tab..
- For a new WebRoutine, it is not necessary to do a compile to generate the XSL, since a  Build also generates the XML / XSL, but without doing a full compile.
- **Note:** XSL can be generated for a *selected WebRoutine* using the context menu option, *WebRoutine: nnnnnnn / Generate XSL* in the LANSAX Editor.



What I Should Know

- How to create a simple enquiry WAM.
- How to open the *Design* view for a specific WebRoutine.
- What is generated by a WAM compile.
- A WAM may contain more than one WebRoutine
- A WebRoutine name may be up to 20 characters long.
- There is one WAM layout generated for each WAM with the name xxx_layout, where xxx = the WAM name.
- By default, fields are visualized as a label and edit box
- Fields are displayed in a table
- A WebRoutine may be called by a weblet such as a push button, via its on_click_wrname property.

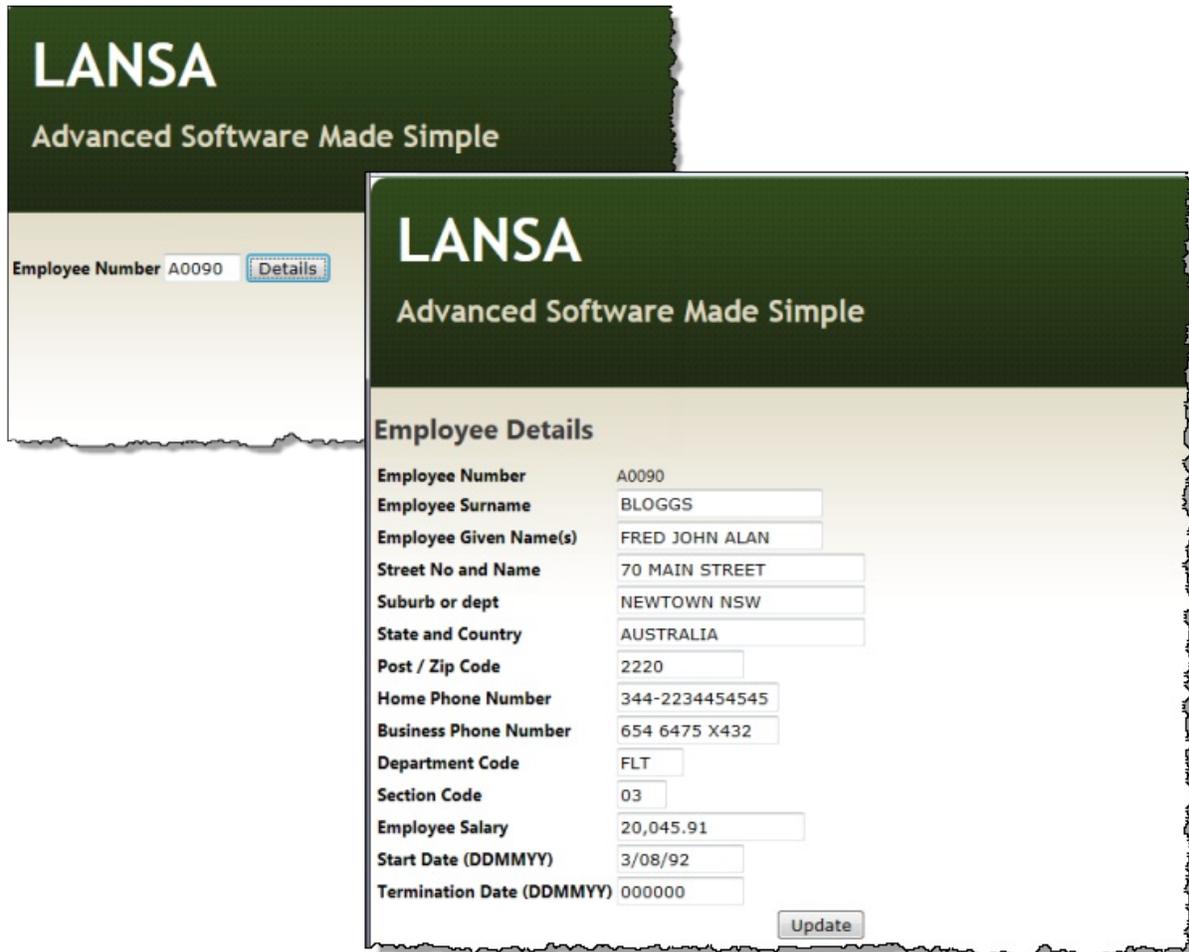
- Field names or a `group_by` may be used to define fields in a `web_map`.

WAM035 - An Employee Update WAM

Objectives

To create an employee update WAM **iiiEmpUpdate – Employee Update**, starting from **iiiEmpEnquiry – Employee Enquiry**.

All the screen fields except employee number need to be made input capable, and an *Update* button will be added to the *Details* page.



The screenshot shows the LANSA web interface. The top header reads "LANSA Advanced Software Made Simple". Below this, there is a navigation bar with "Employee Number A0090" and a "Details" button. The main content area is titled "Employee Details" and contains a form with the following fields:

| | |
|---------------------------|----------------|
| Employee Number | A0090 |
| Employee Surname | BLOGGS |
| Employee Given Name(s) | FRED JOHN ALAN |
| Street No and Name | 70 MAIN STREET |
| Suburb or dept | NEWTOWN NSW |
| State and Country | AUSTRALIA |
| Post / Zip Code | 2220 |
| Home Phone Number | 344-2234454545 |
| Business Phone Number | 654 6475 X432 |
| Department Code | FLT |
| Section Code | 03 |
| Employee Salary | 20,045.91 |
| Start Date (DDMMYY) | 3/08/92 |
| Termination Date (DDMMYY) | 000000 |

An "Update" button is located at the bottom right of the form.

To achieve these objectives you will complete the following:

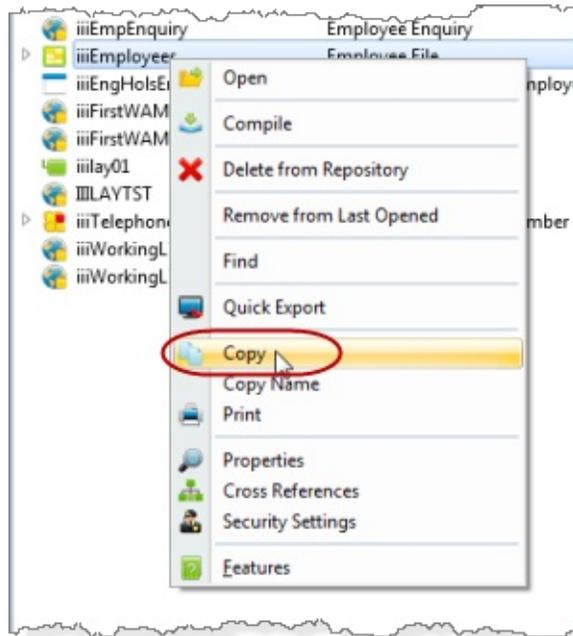
- [Step 1. Create WAM iiiEmpUpdate – Employee Update](#)
- [Step 2. Compile your WAM and complete the Details web page](#)
- [Step 3. Test the Employee Update WAM](#)
- [Summary](#)

Before You Begin

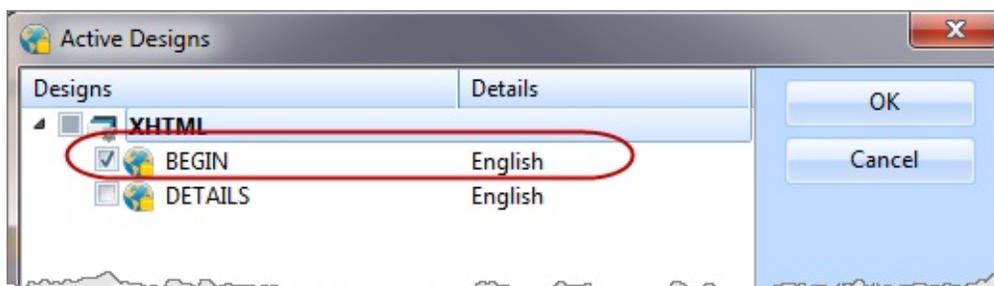
- In order to complete this exercise, you should have completed all the preceding exercises.

Step 1. Create WAM iiiEmpUpdate – Employee Update

1. Select WAM iiiEmpEnquiry – Employee Enquiry on your Favorites / Last Opened tab and use the context menu to *Copy* it to create iiiEmpUpdate – Employee Update.



On the *Active Designs* dialog, select only the **Begin** WebRoutine:



In the Active Designs you selected only the BEGIN WebRoutine. This will copy the XML and XSL for this WebRoutine only. In the new WAM iiiEmpUpdate you are redesigning the DETAILS web page, making most of the fields input capable. When you compile iiiEmpUpdate having made your RDMLX changes to the DETAILS WebRoutine, the correct start position will be generated for the new DETAILS web page.

2. Change the GROUP_BY so that all fields are input capable except for EMPNO, currently they are all displayed for output only. Once again use the

F4 Command Assistant to make these changes.

3. You need to ensure that the fields are both sent to the web page and received back from it, since this time you are displaying and updating employee fields. Remember that fields in a WEB_MAP are by default mapped with an *INPUT attribute. In order to display and update employee data, your fields will need to be mapped For(*both).

4. The Details WebRoutine now needs to performs two roles. It FETCHs a record when invoked from the Begin WebRoutine, and UPDATEs the record when re-entered via an Update button. The field STDREENTRY will be used to control which logic to perform.

Note #1: STDREENTRY is a single character field which is defined as the default field returned by a number of weblets. This is simply a convention and you may prefer to replace it with a longer field which supports a more meaningful input values such as, ENQUIRE, UPDATE or DELETE.

Note #2: You should now map field STDREENTRY For(*both) as a hidden field.

5. Modify the **Details** WebRoutine using a CASE / ENDCASE loop for field STDREENTRY, so that when the **Details** WebRoutine is called from the **Begin** WebRoutine (returning a STDREENTRY value of M) the employee record is fetched. When the **Details** WebRoutine is re-entered from the **Details** web page a value of U should perform an UPDATE to the employee file.

The Employee Number (EMPNO) should be output on the Details page so that the key field cannot be changed. An output field on the web page cannot be mapped back into the next WebRoutine. Of course the WAM needs the Employee Number to perform the update.

This is a common situation with all web applications. One solution is to output a hidden work field containing the Employee Number and use this to perform the update.

6. Define a work field EMPNOW based on EMPNO. Map field EMPNOW for *both as a *hidden field. You could map both STDREENTRY and EMPNOW for all WebRoutines by defining this map at the WAM level (that is immediately following the **Define_Com**). WEB_MAPs defined at the WAM level are global and apply to all WebRoutines.

7. Consider how to add logic to the Details WebRoutine to return to the Begin page if the update is successful.

Hint: remember the TRANSFER command that you implemented in an

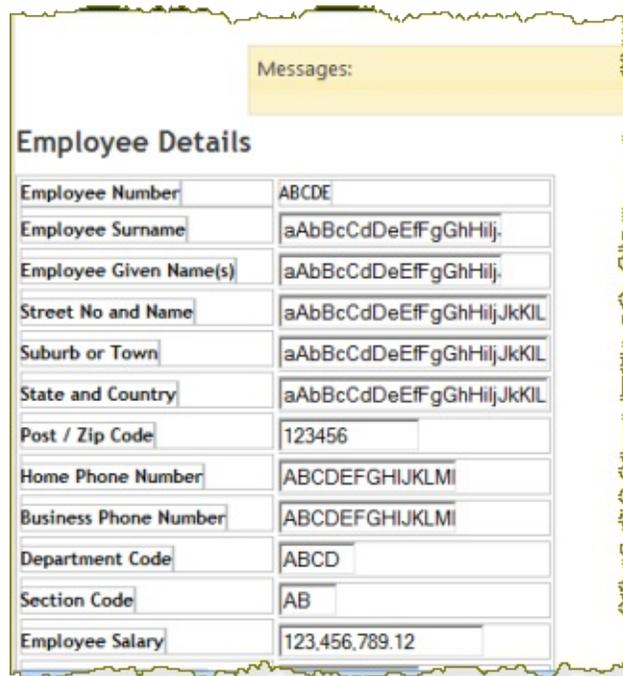
earlier exercise.

8. Your WAM should now look something like the example following. The changed and new code compared with iiiEmpEnquiry is shown in red.

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_WAM) Layoutweblet('iilay01')
Group_By Name(#EMPDATA) Fields((#empno *out) #SURNAME #GIVE
Define Field(#EMPNOW) Reffld(#EMPNO)
Web_Map For(*BOTH) Fields((#STDREENTRY *HIDDEN) (#EMPNOW *H
WebRoutine Name(begin)
Web_Map For(*output) Fields(#empno)
Endroutine
WebRoutine Name(Details) Desc('Employee Details')
Web_Map For(*BOTH) Fields(#EMPDATA)
Case (#STDREENTRY)
When ('= M')
Fetch Fields(#EMPDATA) From_File(PSLMST) With_Key(#EMPNO) Val_Er
If_Status Is_Not(*OKAY)
Message Msgtxt('Employee not found')
Transfer Toroutine(Begin)
Endif
Change Field(#EMPNOW) To(#EMPNO)
When ('= U')
#EMPNO := #EMPNOW
Update Fields(#EMPDATA) In_File(PSLMST) With_Key(#EMPNO) Val_
If_Status Is(*OKAY)
Transfer Toroutine(Begin)
Endif
Endcase
Endroutine
End_Com
```

Step 2. Compile your WAM and complete the Details web page

1. Compile your WAM and open the Begin WebRoutine in the Design view. Notice that it is a copy of the web page design from WAM iiiWAM030, that you included in the Copy process. The Details push button is already defined.
 - a. Select Details push button and on the *Details* tab, click on the Ellipsis button for the *submitExtraFields* property to display the *Design of...* dialog. Select **STDREENTRY** in the *Name* column and enter **M** as a literal in the *Value* column
 - b. Click *OK* to close the dialog and confirm the change.
2. Open the **Details** WebRoutine in the Design view. Your web page should look like the following:



| Employee Details | |
|------------------------|-----------------------|
| Employee Number | ABCDE |
| Employee Surname | aAbBcCdDeEfGhHij. |
| Employee Given Name(s) | aAbBcCdDeEfGhHij. |
| Street No and Name | aAbBcCdDeEfGhHijJkKIL |
| Suburb or Town | aAbBcCdDeEfGhHijJkKIL |
| State and Country | aAbBcCdDeEfGhHijJkKIL |
| Post / Zip Code | 123456 |
| Home Phone Number | ABCDEFGHijklmI |
| Business Phone Number | ABCDEFGHijklmI |
| Department Code | ABCD |
| Section Code | AB |
| Employee Salary | 123,456,789.12 |

3. As before the Department and Section fields may be defined in the Repository with a dropdown weblet visualization. You will be using these in a later exercise. For now you will make them normal input fields. To do this, select the Department dropdown list and using the context menu, select *Replace with Input Field*. Repeat this step for the Section field.
4. Select within the table containing the fields, and use the context menu Table Items to Add Row.... A row will be added to the bottom of the table.
5. Select the bottom right hand cell in the table and use the Detail tab to set its

align property to right. Drag and drop a push button weblet into this bottom right hand cell. Ensure that the button is selected, and use the Detail tab set up the button properties as follows:

| Property | Value |
|-------------------|--|
| caption | Update |
| on_click_wname | Details |
| submitExtraFields | Field Name: STDREENTRY
Literal Value: U |

Note:

- a. Select the *on_click_wname* property and set its value from the dropdown list.
 - b. Then use the *Ellipsis* button for the *submitExtraFields* value, to open the *Design of...* dialog.
6. Save your changes.

Your page should now look like the following:

Employee Details

| | |
|---------------------------|--|
| Employee Number | Value EMPNO |
| Employee Surname | ABCDEFGHIJKLMNOPS |
| Employee Given Name(s) | ABCDEFGHIJKLMNOPS |
| Street No and Name | aAbBcCdDeEfGhHiIjJkKLm |
| Suburb or dept | aAbBcCdDeEfGhHiIjJkKLm |
| State and Country | aAbBcCdDeEfGhHiIjJkKLm |
| Post / Zip Code | 123456 |
| Home Phone Number | ABCDEFGHIJKLMNO |
| Business Phone Number | ABCDEFGHIJKLMNO |
| Department Code | ABCD |
| Section Code | VALU |
| Employee Salary | 123,456,789.12 |
| Start Date (DDMMYY) | 12/34/56 |
| Termination Date (DDMMYY) | 123456 |
| | <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> |
| | <input type="checkbox"/> Update <input type="checkbox"/> |
| | <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> |

Step 3. Test the Employee Update WAM

1. On the *Design* ribbon use the *Open Webroutine* button to open the *Begin WebRoutine* in the *Design* view.
2. Use the  *Execute* button on the *Home* ribbon to run it in the browser. Ensure that you can display and update an employee. When your update is successful you should be returned to the *Begin* web page.
3. Try to perform an invalid update. For example, make the surname field blank. Note that the *Details* web page is redisplayed and validation errors are displayed in the message area at the top of the page.

Summary

Important Observations

- Fields defined in a WEB_MAP will be input capable unless the field has an *output attribute
- The field STDREENTRY is the default field returned by many weblets such as a hyperlink, a check box or a radio button.
- Push buttons are may return a number of fields via the *submitExtraFields* property. You must use the Ellipsis button for this property and set up the field(s) and values(s) to be returned in the *Design of...* dialog. Always specify the *on_click_wrname* property before using the *Design of...* dialog.
- WebRoutines often use a CASE loop for field STDREENTRY to handle different execution logic.

Tips & Techniques

- Web applications often need to use a work field to pass back a value for *output key fields, such as employee number in this example
- Fields may be changed from input to output in the Design view
- You can open a WebRoutine in the Design view from the RDML source editor using the  Open WebRoutine icon. See example following:

```
Webroutine Name(begin)   
  Web_Map For(*output) Fields(#empno)  
Endroutine
```

What You Should Now Know

- You can set *on_click_wrname* by selecting from a dropdown list – provided the WebRoutine is defined in your RDML code.
- How to code a simple WAM to perform display/update logic.
- How to use the TRANSFER command to invoke a WebRoutine.

WAM040 - Add dropdown lists for Department and Section

Objectives

The fields DEPARTMENT and SECTION are defined in two table files DEPTAB and SECTAB. Sections belong to a department and the SECTAB file is therefore keyed on DEPARTMENT and SECTION.

In this exercise you will replace the fields DEPARTMENT and SECTION with a combo box weblet, and link each weblet to a working list of values based on the table files DEPTAB or SECTAB. Because sections belong to a department, if the department selected is changed, the list of sections will need to be refreshed.

Your finished Details page will look like the following:



The screenshot shows the LANSA 'Employee Details' form. The header includes the LANSA logo and the tagline 'Advanced Software Made Simple'. The form contains the following fields:

| | |
|---------------------------|--------------------|
| Employee Number | A0090 |
| Employee Surname | BLOGGS |
| Employee Given Name(s) | JOE |
| Street No and Name | 12 River Lane |
| Suburb or Town | St Albans |
| State and Country | Herts |
| Post / Zip Code | 2000 |
| Home Phone Number | 01727 554661 |
| Business Phone Number | 01727 8875643 |
| Department Code | ADMINISTRATOR DEPT |
| Section Code | INTERNAL ADMIN SRV |
| Employee Salary | 20,045.91 |
| Start Date (DDMMYY) | 3/08/92 |
| Termination Date (DDMMYY) | 0/00/00 |

An 'Update' button is located at the bottom right of the form.

To achieve these objectives, you will complete the following:

- Review The Dynamic Select Box Weblet and Automatic Updating following.
- [Step 1. Create iiiWAM040 - iii Employee Update - Enhanced](#)
- [Step 2. Add Dynamic Select Boxes to the Details Web Page](#)
- [Step 3. Make the Sections Dropdown list dynamic](#)

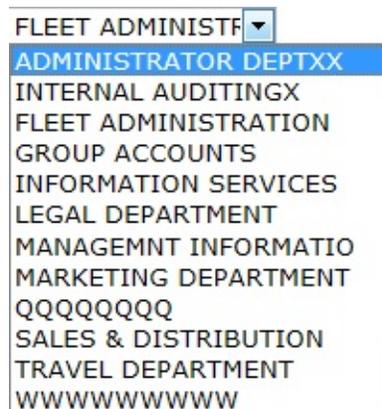
- [Summary](#)

Before You Begin

In order to complete this exercise, you should have completed all the preceding exercises.

The Dynamic Select Box Weblet

The dynamic select box weblet produces a dropdown list containing a list of values.



The list of values is usually provided by a working list output by the WebRoutine. Alternatively the weblet's items property may be used to define a hard coded list of values.

The list may contain two or three columns. The codefield defines a value to be returned when an entry is selected and the captionfield defines a description that will be displayed in the dropdown list. An optional third column defines a selector field that defines a group of values that should be displayed when a related field value changes.

The web_map must define the working list for output and the list must be defined as using JSON format. For example:

```
Web_map For(*output) Fields((#deptdd *json))
```

JSON stands for JavaScript Object Notation. JSON is a lightweight data-interchange format. See <http://www.json.org/> for more information.

Automatic Updating

The dynamic select box is AJAX enabled. This means it is capable of invoking a specially defined "response" WebRoutine that rebuilds and outputs the list that supports the dropdown list.

The dynamic select box can monitor another field and automatically refresh itself whenever that field is updated. If the weblet has been filled using a

working list then you will need to create a JSON WebRoutine that will output a fresh copy of the working list. The weblet will call this WebRoutine each time it needs to refresh.

Only this JSON data is refreshed and the rest of the page is unchanged. This design can provide very responsive web applications. For example, this WebRoutine rebuilds a list of sections when department changes:

```
WebRoutine Name(UsectDD) Response(*JSON)
Web_Map For(*input) Fields(#deptment)
Web_Map For(*output) Fields((#sectdd *JSON))
#com_owner.buildDD2 I_Dept(#deptment) I_Sect(#section)
Endroutine
*
Mthroutine Name(BuildDD2)
Define_Map For(*input) Class(#deptment) Name(#i_dept)
Define_Map For(*input) Class(#section) Name(#i_sect)
#deptment := #i_dept
Clr_List Named(#sectdd)
Select Fields(#sectdd) From_File(sectab) With_Key(#deptment)
Add_Entry To_List(#sectdd)
Endselect
If (#i_sect *NE *blanks)
#section := #i_sect
Endif
Endroutine
```

These properties set up the dynamic select box weblet to automatically call a WebRoutine to refresh:

- updateOnFieldChange* – the name of the field to be monitored
- updateWamName* – the name of the WAM to invoke. Only required if not the current WAM
- updateWrName* – the name of the WebRoutine to invoke
- updateFieldsToSubmit* – the name of one or more fields to be submitted when the monitored field changes.

Step 1. Create iiiWAM040 - iii Employee Update - Enhanced

In this step you will create iiiEmpUpdate_MK2 –Employee Update Enhanced, by copying WAM iiiEmpUpdate and then add logic to handle working lists for departments and sections.

1. Select WAM iiiEmpUpdate on the Favorites/Last Opened tab and use the context menu to copy it.
2. In the Create WAM dialog define your new WAM as – iiiEmpUpdate_MK2 – Employee Update Enhanced.
3. In the Active Designs dialog, leave the Begin and Details WebRoutine checked. You are creating a copy of iiiEmpUpdate including its layouts (XSL).



4. In the new WAM, iiiEmpUpdate_MK2, define working lists for department and section fields. Define the lists immediately following the Begin_com. Your code should look like the following:

```
Def_List Name(#deptdd) Fields(#department #deptdesc) Type(*Working)
Entrys(*max)
Def_List Name(#sectdd) Fields(#section #secdesc) Type(*Working)
Entrys(*max)
```

5. Create a method routine BuildDD1 to clear and build the department list, DeptDD. This routine will require the following logic:
 - a. Map for input a variable based on DEPARTMENT. This will enable the routine to position the list to the current value of DEPARTMENT.
 - b. Clear the list DEPTDD
 - c. Add entries to DEPTDD for all records in file DEPTAB – Department code table
 - d. Reset DEPARTMENT to the input variable value.

Your completed code should look like the following:

```
Mthroutine Name(BuildDD1)
Define_Map For(*input) Class(#deptment) Name(#i_dept)
Clr_List Named(#deptdd)
Select Fields(#deptdd) From_File(deptab)
Add_Entry To_List(#deptdd)
Endselect
#deptment := #i_dept
Endroutine
```

6. Create a method routine BuildDD2 to clear and build the sections list, SectDD. This routine will require the following logic:
 - a. Map for input a variable based on DEPARTMENT. This will enable the routine to rebuild the list of sections for the current value of DEPARTMENT.
 - b. Map for input a variable based on SECTION. This will enable the routine to position the list to the current value of SECTION.
 - c. Set DEPARTMENT to the value of the input variable.
 - d. Clear the list SectDD
 - e. Add entries to the list from the SECTAB – Section code table, using DEPARTMENT as key.
 - f. If the passed variable for SECTION is not blank, set SECTION to the value of the input variable

Your completed code should look like the following:

```
Mthroutine Name(BuildDD2)
Define_Map For(*input) Class(#deptment) Name(#i_dept)
Define_Map For(*input) Class(#section) Name(#i_sect)
#deptment := #i_dept
Clr_List Named(#sectdd)
Select Fields(#sectdd) From_File(sectab) With_Key(#deptment)
Add_Entry To_List(#sectdd)
Endselect
If (#i_sect *NE *blanks)
#section := #i_sect
Endif
Endroutine
```

7. In this step you will make changes to your existing **Details** WebRoutine.
 - a. Add the lists DeptDD and SectDD to the web_map with a JSON attribute. Your code should look like the following:

```
Web_Map For(*BOTH) Fields(#EMPDATA (#deptdd *json) (#sectdd *json))
```

- b. At the end of the Details WebRoutine invoke the new method routines, passing the value of DEPARTMENT and SECTION as required. Your WebRoutine should look like the following. Unchanged code isn't shown.

```
WebRoutine Name(Details) Desc('Employee Details')
Web_Map For(*BOTH) Fields(#EMPDATA (#deptdd *json) (#sectdd *json))
....
....
Endcase
#com_owner.BuildDD1 I_Dept(#department)
#com_owner.BuildDD2 I_Dept(#department) I_Sect(#section)
Endroutine
```

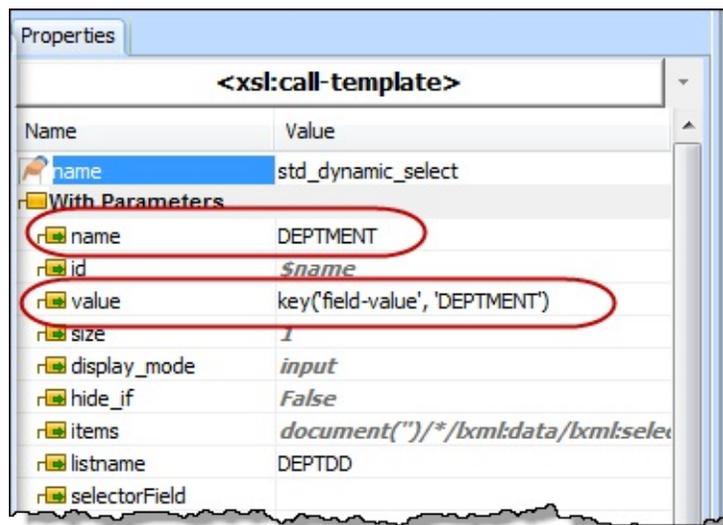
8. Compile your WAM.

Step 2. Add Dynamic Select Boxes to the Details Web Page

In this step you will add dynamic select boxes to replace the department and section code fields, and test your WAM.

1. Open the **Details** WebRoutine in the *Design* view. Drag and drop a Dynamic Select Box weblet onto the department and the section field input values.

With one of the Dynamic Select Boxes selected in the *Design* view, select the *Details* tab. Notice that the weblet has adopted the field name and value:



2. In the *Design* view, select the DEPARTMENT dynamic select box weblet and use the *Details* tab to define its properties as follows:

| Property | Value |
|--------------|-------------------|
| listname | DEPTDD |
| codeField | DEPARTMENT |
| captionField | DEPTDESC |

3. Select the Section dynamic select box weblet and define its properties as follows:

| Property | Value |
|----------|---------------|
| | SECTDD |

| | |
|--------------|----------------|
| Listname | |
| codeField | SECTION |
| captionField | SECDESC |

Note: All properties can be selected from a dropdown list.

4. Save your changes and test your WAM by running the **Begin** web page.

At this stage:

- Both the dropdown lists should be populated, the section's dropdown list containing only the sections for the current department.
- When employee details are displayed, the dropdown lists should display the correct value for the employee. You could use your WAM `iiiEmpUpdate` to check this.
- If you select a different section and update the employee, the update should be processed correctly.
- The section's dropdown list is populated once only, so if you change department, it will contain an incorrect list of values. You will correct this in the next step.

Step 3. Make the Sections Dropdown list dynamic

In this step you will define a new response WebRoutine which the AJAX enabled Dynamic Select box for sections will invoke.

1. Select the *Source* tab.
2. Create a new response WebRoutine, that will be invoked by the sections dynamic select box when DEPARTMENT changes. The requirements for this WebRoutine are:
 - The WebRoutine statement must have a Response() keyword with a value of *JSON
 - A web_map for input, field DEPARTMENT
 - A web_map for output of the list SectDD, defined as a *JSON list
 - Invoke the buildDD2 method routine passing DEPARTMENT and SECTION

Your code should look like the following:

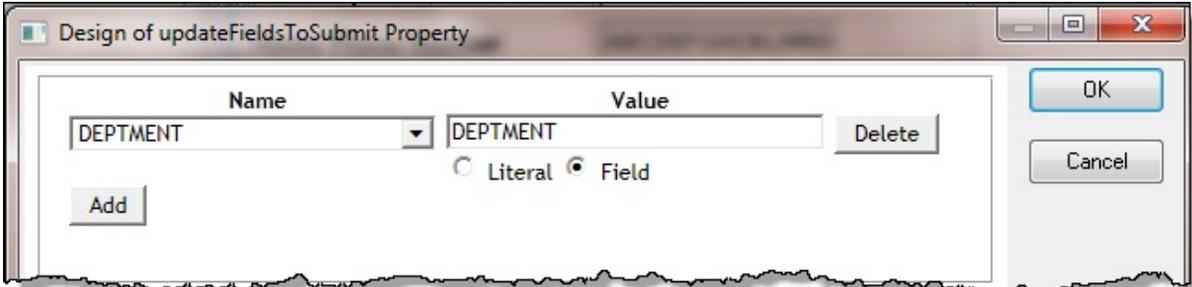
```
WebRoutine Name(USectDD) Response(*JSON)
Web_Map For(*input) Fields(#deptment)
Web_Map For(*output) Fields((#sectdd *JSON))
#com_owner.buildDD2 I_Dept(#deptment) I_Sect(#section)
Endroutine
```

3. Compile your WAM.
4. Open WebRoutine Details in the Design view.
5. Select the Section's dynamic select box and set up additional properties as follows:

```
updateOnFieldChange DEPARTMENT
updateWrName uSectDD
updateFieldsToSubmit Field: DEPARTMENT
Field Value: DEPARTMENT
```

6. Set the updateFieldsToSubmit property by selecting the *Value* column and clicking on the Ellipsis  button to open the *Design of...* property's dialog.

- a. Select a field *Name* of DEPARTMENT
- b. For *Value*, enter DEPARTMENT.
- c. Select the *Field* radio button.



7. Click *OK* to close the dialog.
8. Save your WAM.
9. Test your WAM by running the Begin webroutine. You should now be able to select a new department and notice that the sections dropdown list is refreshed.

In a later exercise, you will learn more about using the dynamic select box weblet.

Summary

Important Observations

- The Dynamic Select Box weblet is AJAX enabled. Its entries can be defined by a working list using a *JSON attribute.
- A dynamic select box can dynamically refresh its list of values by calling a response WebRoutine when a monitored field value changes. The response WebRoutine must have a Response(*JSON) keyword.

Tips & Techniques

- Method routines can be used in a Web Application Module.

What You Should Now Know

- How to implement dynamic select box weblets using a working list.
- How to setup a dynamic select box to refresh its list of values from the server.

WAM045 - A Dynamic Selector Dropdown list using a Select Field

Objectives

This example changes the implementation of the Dynamic Selector Dropdown list weblet for section code, by making use of a third field in the sections working list. The working list will now be defined as:

```
Def_List Name(#sectdd) Fields(#deptment #section #secdesc) Type(*Working
```

The list of sections will now be built once only and the selector field (DEPARTMENT) will enable the weblet to select the values to show, based on the value of field DEPARTMENT. A response WebRoutine for the Dynamic Selector for SECTIONS is no longer required.

The Dynamic Selector weblet for field SECTION will now to be set up to only display values that match the selector field, DEPARTMENT.

The list SECTDD will now contain all values from the table SECTAB. This technique works well if the total number of sections is small, as in this case. However if the possible list of department codes and section codes is large (1,000's for example rather than 100's), then the solution implemented in exercise WAM040 will be a better solution. As usual there is a trade off to consider. Is it better to output all values of section to the web page once, or to refresh the list of sections every time the department code changes? Bear in mind that this second approach is itself efficient because it uses AJAX techniques to only refresh the sections list and not the whole web page.

To demonstrate this technique you will complete the following :

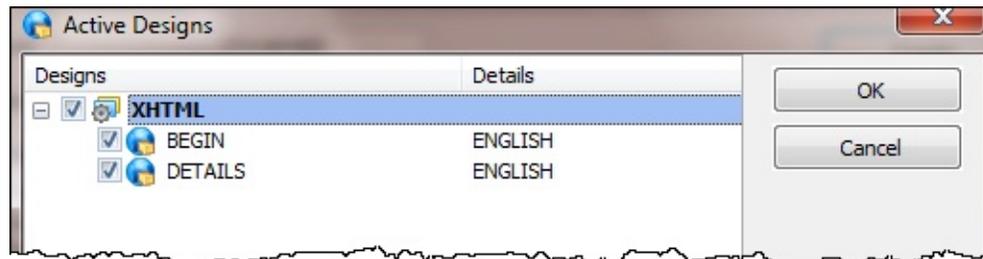
- [Step 1. Create WAM iiiDynamSelector – Dynamic Selector using Select Field](#)
- [Step 2. Setup the Dynamic Selector Dropdown list for Sections](#)
- [Summary](#)

Before You Begin

You should complete all preceding exercises.

Step 1. Create WAM iiiDynamSelector – Dynamic Selector using Select Field

1. Create WAM iiiDynamSelector – Dynamic Selector using Select Field by copying WAM iiiEmpUpdate_MK2 including the Active Designs:



2. Change the definition of the SECTDD list to include DEPARTMENT:
Def_List Name(#sectdd) Fields(**#department** #section #secdesc) Type(*Working
3. Remove the USect_DD WebRoutine, which is no longer required.
4. In the **Details** webroutine, move the code to invoke the BuildDD2 method routine to the position shown, at the end of the When (= M) logic. The moved code is shown in red. This routine will now be invoked once only.

```
WebRoutine Name(Details) Desc('Employee Details')
Web_Map For(*BOTH) Fields(#EMPDATA (#deptdd *json) (#sectdd *json))
Case (#STDREENTRY)
When ('= M')
Fetch Fields(#EMPDATA) From_File(PSLMST) With_Key(#EMPNO) Val_Ei
If_Status Is_Not(*OKAY)
Message Msgtxt('Employee not found')
Transfer Toroutine(Begin)
Endif
Change Field(#EMPNOW) To(#EMPNO)
```

#com_owner.BuildDD2 I_Dept(#department) I_Sect(#section)

5. Change the BuildDD2 method routine to match the following:
The Select now reads all records (no With_key(#department)).
An If / Endif block has been removed

The code shown in red has been added.

```
Mthroutine Name(BuildDD2)
Define_Map For(*input) Class(#deptment) Name(#i_dept)
Define_Map For(*input) Class(#section) Name(#i_sect)
Clr_List Named(#sectdd)
Select Fields(#sectdd) From_File(sectab)
Add_Entry To_List(#sectdd)
Endselect
Loc_Entry In_List(#sectdd) Where((#deptment = #i_dept) And (#section =
Endroutine
```

The LOC_ENTRY statement sets the values of DEPARTMENT and SECTION to the values retrieved for the current employee.

6. Compile your WAM.

Step 2. Setup the Dynamic Selector Dropdown list for Sections

1. Open the Details WebRoutine in the Design view.
2. Select the Dynamic Selector weblet for field SECTION and change its properties as shown:

| Property | Value |
|----------------------|-------------------|
| listname | SECTDD |
| selectorField | DEPARTMENT |
| selectorValueField | DEPARTMENT |
| codeField | SECTION |
| captionField | SECTDESC |
| updateOnFieldChange | DEPARTMENT |
| updateWrName | |
| updateFieldsToSubmit | |

Note that the updateWrName and updateFieldToSubmit properties no longer have a value.

3. Save your changes
4. Execute your WAM in the browser to test (run the begin WebRoutine).
 - a. Change department and section code values.
 - b. Redisplay the employee and ensure that the changes have been correctly processed.

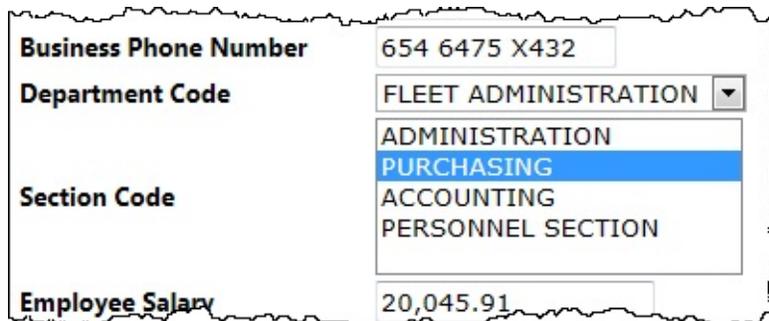
Summary

Important Observations

- As noted in the Objectives section, choosing the best technique to use, requires a good understanding of your database.

Tips & Techniques

- The Dynamic Select weblet may be used in a number of other ways. See the Web Application Modules guide for further details.
- For example, change the size property to 5, to display section codes as shown.



The image shows a screenshot of a web form with four fields. The 'Department Code' field is a dropdown menu currently showing 'FLEET ADMINISTRATION'. The 'Section Code' field is also a dropdown menu, currently showing 'PURCHASING' which is highlighted in blue. The other fields are text inputs: 'Business Phone Number' with the value '654 6475 X432' and 'Employee Salary' with the value '20,045.91'.

| | |
|-----------------------|----------------------|
| Business Phone Number | 654 6475 X432 |
| Department Code | FLEET ADMINISTRATION |
| Section Code | PURCHASING |
| Employee Salary | 20,045.91 |

- The weblet also supports multiple selections and return a list of fields to a WebRoutine.
- The updateFieldsToSubmit may be a list of field names, rather than a single value.

What You Should Know

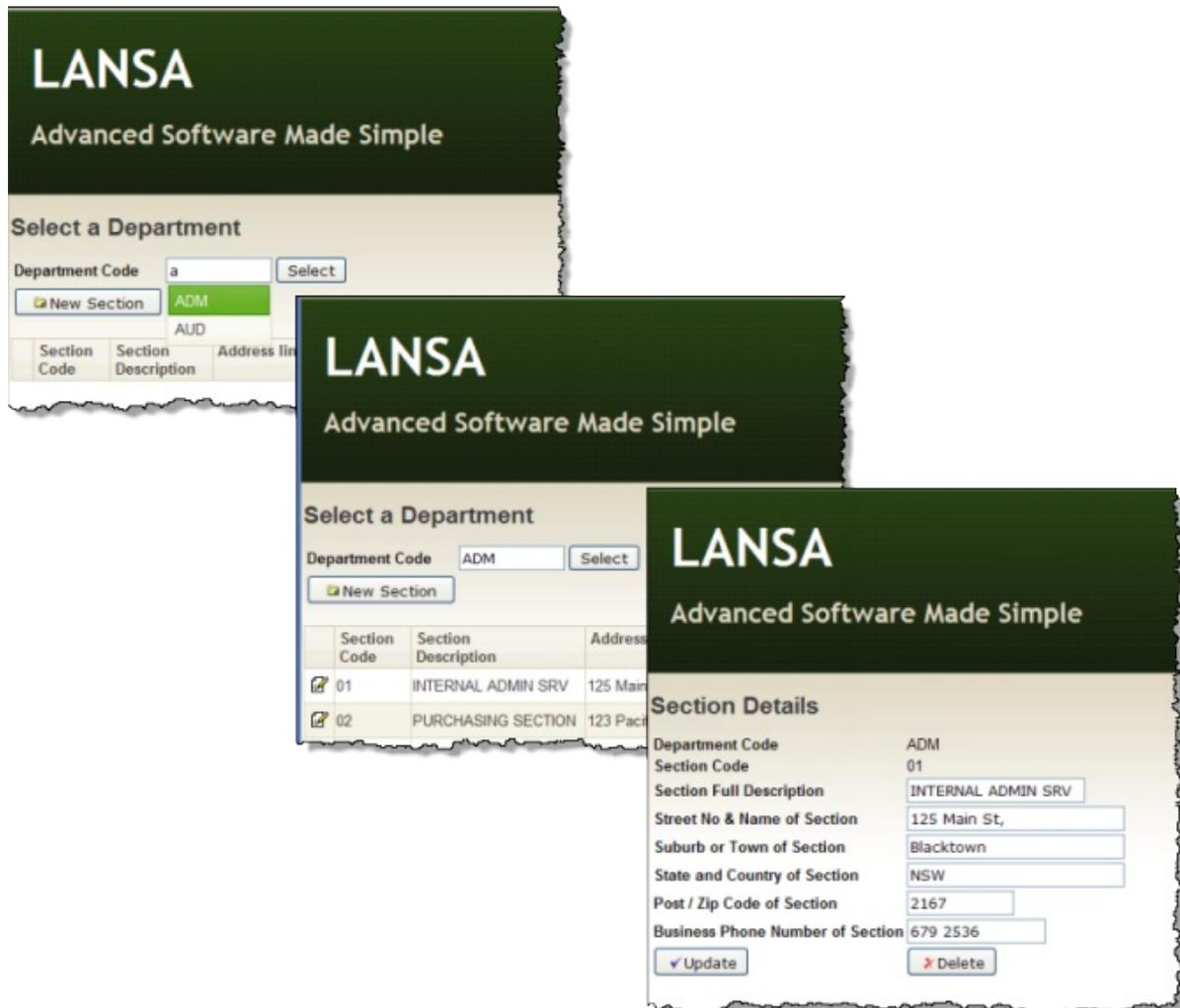
- You should now be aware that the Dynamic Selector weblet has a large number of features that may be used to enhance your user interface and the applications functionality.

WAM050 - A Section Maintenance Application

Objectives

- This exercise introduces an application that includes a list which is used to present a result set which enables selection of a single entry to maintain a record.
- To write a Section Maintenance WAM to display, update, delete and add sections. The main page will be based on a list of sections for the selected department.
- To show how lists can be used in WebRoutine web page.
- To implement an example of the AutoComplete weblet.

Your completed application will look like the following:



To achieve these objectives you will complete the following steps:

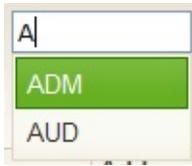
- [Step 1. Create iiiSecMaint - Section Maintenance WAM](#)
- [Step 2. Add a Details WebRoutine](#)
- [Step 3. Create iiiSecAdd - Add Section WAM](#)
- [Step 4. Complete the AddSect WebRoutine](#)
- [Step 5. Set up the 'New Section' button on the Begin page for iiSecMaint](#)
- [Summary](#)

Before You Begin

- In order to complete this exercise, you should have completed all the preceding exercises.

The AutoComplete Weblet

The AutoComplete weblet provides suggestions while you type into the field. The suggestions are provided by a WebRoutine using Ajax.



A *sourceWrName* property defines a called WebRoutine, which supports the AutoComplete weblet by returning a list as JSON data. The WebRoutine must have a *Response(*JSON)* keyword. The weblet calls this WebRoutine when you type into the input box.

A *listName*, *labelField* and *ValueField* define the list of values that the AutoComplete weblet displays.

Note: The AutoComplete weblet in this exercise will display matching department codes by reading the file DEPTAB using *Generic(*yes)*. This provides a simple introduction to implementing this weblet, but does not provide a realistic example. Since the file DEPTAB has a very small number of records a complete list of values shown in a combo box would be a better implementation. You will also use the Dynamic List Box that was part of the previous exercise.

Step 1. Create **iiiSecMaint** - Section Maintenance WAM

To design and develop your WAM applications, just like any other new application, you need to begin by focusing on:

- What is the business process?
- The user interface
- A new web application will require many WAMs and many web pages. Start by drawing up a plan of the web site and the navigation it will require.
- What WebRoutines will be required?
- What working lists, group_bys, working fields need to be defined?
- What weblets can be used to enhance the user interface?
- What working lists will be required to support these weblets?
- How do fields and lists need to be mapped into and out of each WebRoutine?
- In a new WAM create your WebRoutines and WEB_MAPS before coding your application logic
- When you add a new WebRoutine to a WAM, the XML/XSL will be built at compile time.
- Avoid creating very large WAMs with too many WebRoutines. Each time a WebRoutine is invoked from the browser, the WAM loads and then unloads. Many small WAMs is a much better design and will be easier to maintain.

1. Create a new WAM.

Name: **iiiSecMaint**

Description: **Section Maintenance**

Layout weblet: **iiilay01**

Consider the first two screen captures shown in the Objectives section that show the **Begin** WebRoutine in operation. This WebRoutine initially displays a list of sections that is empty and an input field for a department code. An AutoComplete weblet will replace the Department Code input field. A *Select* push button invokes the **Begin** WebRoutine that builds a list of sections for the department and displays this list.

Consider what working lists will be needed to handle this web page and what logic will be necessary?

a. Define a work field DEPT_IN based on DEPARTMENT, this will be the

department code input field.

- b. Define working list DEPTS, to support the AutoComplete weblet. The list contains the field DEPARTMENT only.
 - c. Define working list SECTLIST, to support the list of sections containing STDSELECT, SECTION, SECDESC, SECADDR1. Note: All fields apart from STDSELECT should have an *output attribute.
 - d. Map STDREENTRY globally as a hidden field.
2. Define a WebRoutine named *Begin*, based on the following pseudo code

Map field DEPT_IN for * both

Map for *output the Sections working list, SECTLIST

Case of field STDREENTRY

* Select Push Button

When = S

clear list SECTLIST

select fields in working list from the section table with the key
dept_in

add entry to working list

end of select loop

end of case loop

3. Define a response WebRoutine AutoComplete to build the list DEPTS to support the AutoComplete weblet

Define WebRoutine AutoComplete with a Response(*JSON)
keyword

Map DEPT_IN for input

Map list DEPTS for output as a *JSON list

Convert the first character of DEPT_IN to uppercase

clear list of departments

Select DEPARTMENT from the file DEPTAB with the key
DEPT_IN, with a Generic(*yes) keyword

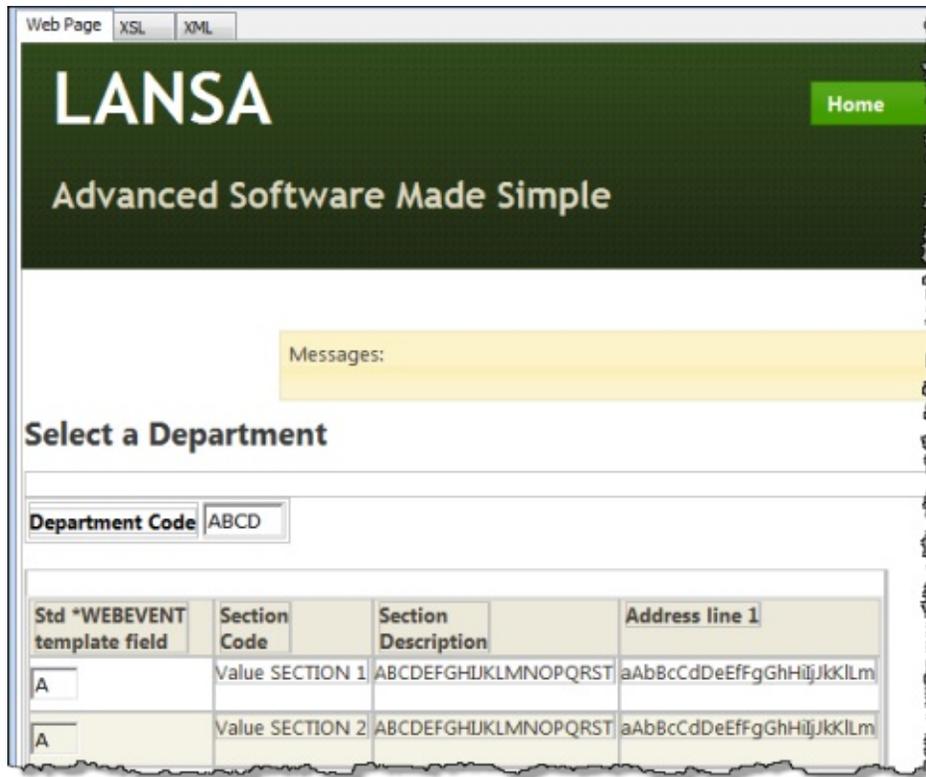
```
Add an entry to the department list
end select
end subroutine
```

Your completed code should now look like the following:

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_WAM) Layoutweblet('iilay01')
Define Field(#dept_in) Reffld(#deptment)
Def_List Name(#depts) Fields(#deptment) Type(*Working)
Def_List Name(#sectlist) Fields(#STDSELECT (#SECTION *out) (#SECDES
Web_Map For(*both) Fields((#stdrentry *hidden))
WebRoutine Name(Begin) Desc('Select a Department')
Web_Map For(*both) Fields(#dept_in #sectlist)
Case (#stdrentry)
When (= S)
Clr_List Named(#sectlist)
Select Fields(#sectlist) From_File(sectab) With_Key(#dept_in)
Add_Entry To_List(#sectlist)
Endselect
Endcase
Endroutine
WebRoutine Name(AutoComplete) Response(*JSON)
Web_Map For(*input) Fields(#dept_in)
Web_Map For(*output) Fields((#depts *json))
#dept_in := #dept_in.substring( 1, 1 ).upperCase
Clr_List Named(#depts)
Select Fields(#deptment) From_File(deptab) With_Key(#dept_in) Generic(*ye
Add_Entry To_List(#depts)
Endselect
Endroutine
End_Com
```

4. Compile your WAM and open the Begin WebRoutine in the Design view.

Your web page should look like the following:



5. Drag and drop an jQuery UI AutoComplete weblet onto the Department Code field. Select the Details tab and set up the AutoComplete properties:

| Property | Value |
|--------------|---------------------|
| sourceWrName | AutoComplete |
| Listname | DEPTS |
| valueField | DEPARTMENT |

6. If the *Select* column (field STDSELECT) is shown as an input field (i.e. it does not have a clickable image weblet field visualization defined in the Repository), drop a Clickable Image weblet into the field in the first column.
7. With the clickable image selected, select the Details tab and set up its properties:

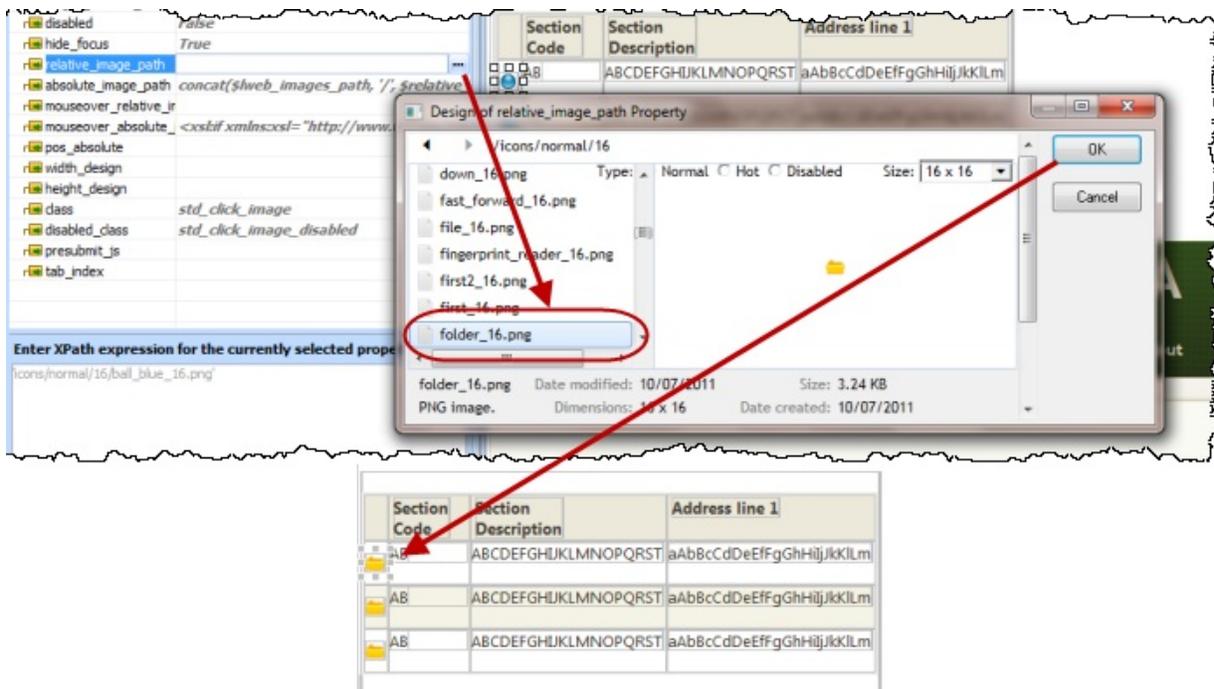
| Property | Value |
|------------------|----------------|
| currentrowhfield | SECTION |

| | |
|--------------------|-------------------------|
| currentrownumvalue | \$SECTION |
| Rentryvalue | S |
| tooltip | Select a Section |
| on_click_wrname | Details |

Note:

- To return a field value from a list define the value as \$FIELDNAME (upper case). i.e. \$SECTION in this case.
- Although 'SECTION' is an *output field in the list, you can return a field value via the clickable image weblet.
- The **Details** WebRoutine is not yet defined in your WAM, so you need to type in this value for the on_click_wrname property.

8. Select a clickable image weblet. Set the relative_image_path by clicking in the Value column, and then using the *Elipsis* button and select the **/normal/16** folder and then select any suitable image. See the example following:



9. Select the column heading "Std *WEBEVENT template field" and delete it.
10. If the field SECTION has a dropdown field visualization defined in the Repository, it will not be displayed in the list – since it is an *output field. If necessary, select the field SECTION in the list and use the context menu to select the option *Replace with an Output Field*.
11. Select anywhere in the table containing "Department Code" and the AutoComplete weblet. Use the context menu, and select the option Table Items / Add columns... to add one column to the right. Add a *push button* into this new column and remove the * placeholder characters from this cell.
Set up the push button properties:

| Property | Value |
|-------------------|-------------------------------|
| caption | Select |
| on_click_wrname | Begin |
| submitExtraFields | Field Name: STDREENTRY |
| | Literal Value: S |

Note: Whenever possible, set a weblet property value by selection from its dropdown list. Using a list of values provided by the editor when available, will help to minimize errors in your XSL.

12. Save your changes and run the web page in the browser. Your page should look like the following:

Browser address bar: http://localhost/CGI-BIN/lansaweb?wam=WEB045&webtrn=Begin&ml=LAN

LANSA

Advanced Software Made Simple

Select a Department

Department Code

| Section Code | Section Description | Address line 1 |
|--------------|---------------------|----------------------|
| 01 | INTERNAL ADMIN SRV | 125 Main St. |
| 02 | PURCHASING SECTION | 123 Pacific Highway, |
| 03 | ACCOUNTING SECTION | 252 Canterbury Road, |
| 04 | SALES & MARKETING | 121 Pitt Town Road |

Step 2. Add a Details WebRoutine

1. This WebRoutine will support display, update and delete of the selected Section record. Consider the logic that needs to be supported by this WebRoutine.
 - a. What fields and lists need to be mapped?
 - b. How to control the code to be executed each time this WebRoutine is invoked?
 - c. What work fields will be required to retain key values?
 - d. Whether to transfer control back to the **Begin** WebRoutine after a successful update or deletion?
2. At the WAM level, define a work field and a Group_by statement.
 - a. Define a work field, SECTW based on field SECTION. This will retain current value of SECTION.
 - b. Define a Group_by, SEC_DETL for all fields in the file SECTAB. Fields DEPARTMENT and SECTION should have a display attribute of *output.
3. Your new **Details** WebRoutine needs to do the following:
 - Map for *both the Group_by for Section fields and fields DEPT_IN and SECTW. Fields DEPT_IN and SECTW should be hidden fields.
 - Use field STDREENTRY to determine why the WebRoutine was called.
 - When called initially, fetch the Section record and save department and section code in work fields.
 - When called for update, set up department and section code from work fields and update the record.
 - If the update is successful, output a message and transfer to the **Begin** WebRoutine. Consider how the **Begin** routine should redisplay Sections for current department.
 - Output a message if the update is not successful.
 - When called for delete, set up department and section code from work fields, delete the record, if successful, output a message and transfer to **Begin** WebRoutine.
 - Output a message if the delete is not successful.

4. Create your WebRoutine based on the outline above.

Your complete code for the **Details** WebRoutine should look like the following:

```
WebRoutine Name(Details) Desc('Section Details')
Web_Map For(*BOTH) Fields(#SEC_DETL (#SECTW *HIDDEN) (#dept_in
Case Of_Field(#STDREENTRY)
* Initial call from clickable image
When Value_Is(= S)
Fetch Fields(#SEC_DETL) From_File(SECTAB) With_Key(#dept_in #SECTI
#dept_in := #DEPARTMENT
#SECTW := #SECTION
* Update button clicked
When Value_Is(= U)
* ensure that DEPARTMENT and SECTION can be redisplayed if a validation er
#DEPARTMENT := #dept_in
#SECTION := #SECTW
Update Fields(#SEC_DETL) In_File(SECTAB) With_Key(#dept_in #SECTW
If_Status Is(*OKAY)
#STDREENTRY := L
Message Msgtxt('Section changed')
Transfer Toroutine(BEGIN)
Else
Message Msgtxt('Error occurred on update')
Endif
* Delete button clicked
When Value_Is(= D)
Delete From_File(SECTAB) With_Key(#dept_in #SECTW) Val_Error(*NEXT
If_Status Is(*OKAY)
#STDREENTRY := S
Message Msgtxt('Section deleted')
Transfer Toroutine(BEGIN)
Else
Message Msgtxt('Error occurred on deletion')
Endif
Endcase
Endroutine
```

5. Recompile your WAM and open the Design view for the Details

WebRoutine. Your page should look like the following:

The screenshot shows a web form titled "Section Details" with the following fields and values:

| | |
|----------------------------------|---------------------------|
| Department Code | Value DEPARTMENT |
| Section Code | Value SECTION |
| Section Full Description | ABCDEFGHIJKLMNOPQR |
| Street No & Name of Section | aAbBcCdDeEfFgGhHiIjJkKlLm |
| Suburb or Town of Section | aAbBcCdDeEfFgGhHiIjJkKlLm |
| State and Country of Section | aAbBcCdDeEfFgGhHiIjJkKlLm |
| Post / Zip Code of Section | 123456 |
| Business Phone Number of Section | ABCDEFGHIJKLMN |

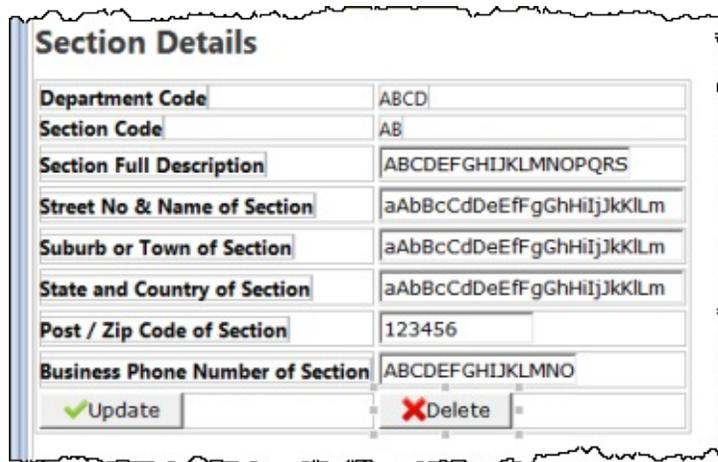
6. If the Department and Section code fields have a combo box field visualization weblet defined in the *Repository*, select each of them and use the context menu to Replace with output field. Fields with visualization weblets defined, will not display on the page in "output" mode.
7. Use the context menu to Add a row to the bottom of the table, and drop a Push Button with Image into each cell.
8. Set up the push button properties based on the following:

| Property | Value |
|---------------------|--|
| caption | Update |
| left_relative_image | icons/normal/16/check_mark_16.png |
| on_click_wname | Details |
| submitExtraFields | Field Name: STDREENTRY
Literal Value: U |
| caption | Delete |
| left_relative_image | icons/normal/16/cross_16.png |
| on_click_wname | Details |
| submitExtraFields | Field Name: STDREENTRY
Literal Value: D |

The `left_relative_image` and `submitExtraFields` properties should be selected using the *Ellipsis* button and the *Design of...* dialog.

9. Save your web page design.

Your completed design should look like the following:



The screenshot shows a web form titled "Section Details" with a light blue header. The form contains several input fields with labels and values, and two buttons at the bottom. The fields are:

| Field Label | Value |
|----------------------------------|-------------------------|
| Department Code | ABCD |
| Section Code | AB |
| Section Full Description | ABCDEFGHIJKLMNOPS |
| Street No & Name of Section | aAbBcCdDeEfGhHiIjJkKlLm |
| Suburb or Town of Section | aAbBcCdDeEfGhHiIjJkKlLm |
| State and Country of Section | aAbBcCdDeEfGhHiIjJkKlLm |
| Post / Zip Code of Section | 123456 |
| Business Phone Number of Section | ABCDEFGHIJKLMNO |

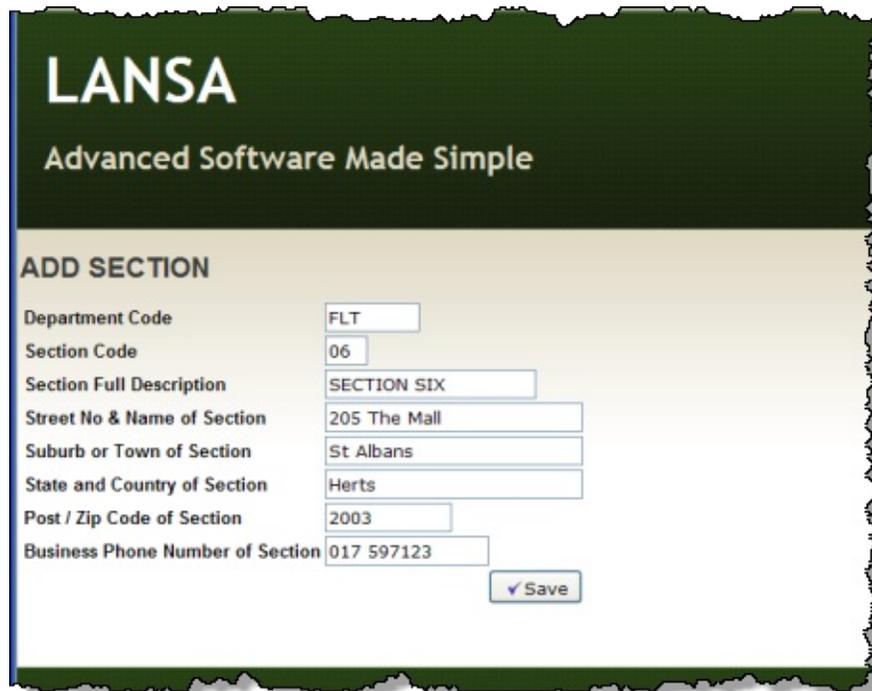
At the bottom of the form, there are two buttons: a green checkmark icon followed by the text "Update", and a red X icon followed by the text "Delete".

10. Re-test your WAM. Check what happens on a successful update or delete. Check what happens after an update with a validation error.

Step 3. Create iiiSecAdd - Add Section WAM

This simple logic could be incorporated into WAM iiiSecMaint, however, you will instead create a new WAM to illustrate building a multi-WAM application.

1. Review the following web page:



The screenshot shows a web form titled "ADD SECTION" from the LANSA application. The form is set against a dark green header with the LANSA logo and the tagline "Advanced Software Made Simple". The form fields are as follows:

| | |
|----------------------------------|--------------|
| Department Code | FLT |
| Section Code | 06 |
| Section Full Description | SECTION SIX |
| Street No & Name of Section | 205 The Mall |
| Suburb or Town of Section | St Albans |
| State and Country of Section | Herts |
| Post / Zip Code of Section | 2003 |
| Business Phone Number of Section | 017 597123 |

A "Save" button with a checkmark icon is located at the bottom right of the form.

- The AutoComplete weblet will support the department code input field
 - All other fields will be entered
 - A **New** button will be added to the **Begin** page for iiiSecMaint to call iiiSecAdd.
 - The Add Section WAM will return to Section Maintenance **Begin** WebRoutine when a section is successfully added.
 - Section Maintenance **Begin** WebRoutine will then list the sections for the department for which a section was added.
2. Create a new WAM **iiiSecAdd– Add Section** using *Layout Webletiiiilay01*. Consider what common definitions and logic you could copy from iiiSecMaint. What WEB_MAPs will be required in the 'AddSect' WebRoutine?
3. Create a basic **outline** for your WAM based on the following pseudo code

Define a work field DEPT_IN based on DEPARTMENT

Define a Group_by of all fields for the Section table. All fields should be input capable

Map field STDREENTRY for *both, as a *hidden field

- Define a WebRoutine "AddSect"

Map field DEPT_IN for *both

Map group_by for Section fields for *both, all fields should be input capable. DEPARTMENT should be a hidden field

End routine

Your code should now look like this:

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_WAM) Layoutweblet('iiilay01')
Define #DEPT_IN Reffld(#DEPARTMENT)
Group_By Name(#SECT_DETL) Fields((#DEPARTMENT *hidden) #SECTION
*
Web_Map For(*BOTH) Fields((#STDREENTRY *HIDDEN))
*
WebRoutine Name(AddSect) Desc('Add Section')
Web_Map For(*BOTH) Fields(#Dept_In #SECT_DETL)
* -----
* Add section logic goes here
* -----
Endroutine
End_Com
```

Step 4. Complete the AddSect WebRoutine

1. Your **AddSect** WebRoutine will handle the initial call from the **Begin** web page in **iiiSecMaint** and the call from the 'Save button' on the **AddSect** web page itself.

Create your logic for WebRoutine **AddSect** based on the following pseudo code:

CASE of STDREENTRY

* when called from Begin WebRoutine

When = N

- Change DEPARTMENT to DEPT_IN (value passed in from **iiiSecMaint**)
- Change SECTION to *null (value is passed in from **iiiSecMaint**)

* when Save button clicked

When = A

Change DEPARTMENT to DEPT_IN (the input value on the web page)

Insert to Section file

If Status is *OKAY

Output message

Change STDREENTRY to L

- Transfer to WebRoutine **iiiSecMaint.Begin**

Endif

Endcase

Your completed RDMLX code should look like the following:

```
Function Options(*DIRECT)
```

```
Begin_Com Role(*EXTENDS #PRIM_WAM) Layoutweblet('iiilay01')
```

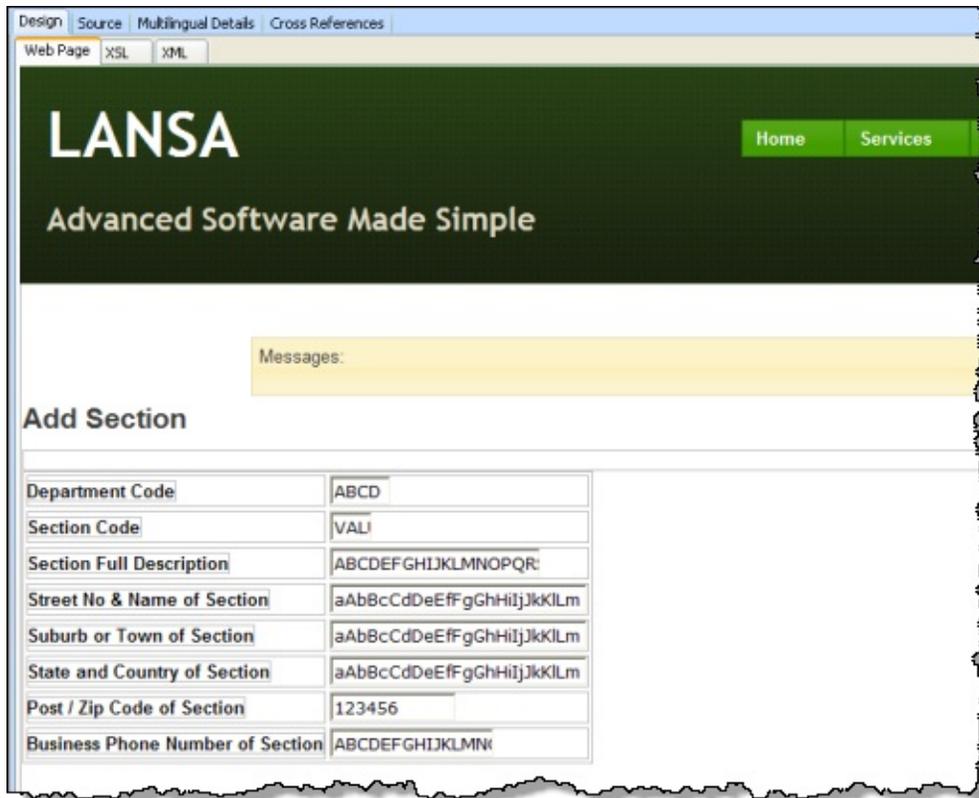
```
Group_By Name(#SECT_DETL) Fields((#DEPARTMENT *hidden) #SECTION
```

```
Define Field(#dept_in) Reffld(#department)
```

```
Web_Map For(*BOTH) Fields((#STDREENTRY *HIDDEN))
```

```
WebRoutine Name(ADDSECT) Desc('ADD SECTION')
Web_Map For(*BOTH) Fields(#dept_in #SECT_DETL)
Case Of_Field(#STDREENTRY)
* New button clicked in iiiSecMaint
When (= N)
#deptment := #dept_in
#SECTION := *NULL
* Add button clicked
When (= A)
#deptment := #dept_in
Insert Fields(#SECT_DETL) To_File(SECTAB) Val_Error(*NEXT)
#STDREENTRY := *NULL
If_Status Is(*OKAY)
Message Msgtxt('Section successfully added')
#stdreentry := L
Transfer Toroutine(#iiiSecMaint.BEGIN)
Endif
Endcase
Endroutine
End_Com
```

2. Compile your WAM and open in the Design view. It should look like the following:



- If necessary change the Department Code to an input field. Drop an AutoComplete weblet onto the department field value. Set up the weblet properties based on:

| Property | Value |
|---------------|---------------------|
| sourceWamName | iiiSecMaint |
| sourcewrName | AutoComplete |
| listName | DEPTS |
| valueField | DEPARTMENT |

Note that the AutoComplete weblet is calling the response WebRoutine you created in WAM **iiiSecMaint**.

- The Section field should be an input field, change it if necessary.
- Add a row to the bottom of the table and add a push button with image to the right hand cell. Make the cell right align. Set up the push button properties as

shown:

| Property | Value |
|---------------------|--|
| caption | Save |
| left_relative_image | icons/normal/16/check_mark_16.png |
| on_click_wname | AddSect |
| submitExtraFields | Field Name: STDREENTRY |
| | Literal Value: A |

6. Save these changes. Your completed AddSect page should look like the following:

Add Section

| | |
|----------------------------------|--|
| Department Code | ABCD |
| Section Code | VALI |
| Section Full Description | ABCDEFGHIJKLMNOPQR |
| Street No & Name of Section | aAbBcCdDeEfGhHiIjJkKlLm |
| Suburb or Town of Section | aAbBcCdDeEfGhHiIjJkKlLm |
| State and Country of Section | aAbBcCdDeEfGhHiIjJkKlLm |
| Post / Zip Code of Section | 123456 |
| Business Phone Number of Section | ABCDEFGHIJKLMN |
| | <input type="checkbox"/> <input checked="" type="checkbox"/> Save <input type="checkbox"/> |

Step 5. Set up the 'New Section' button on the Begin page for iiiSecMainti

1. Open the Begin WebRoutine in WAM iiiSecMaint in the *Design* view.
 - a. Add a new row to the table containing the Department AutoComplete weblet.
 - b. Set the left hand cell in the new row, to align left.
 - c. Add a button with image to the bottom left hand cell.

Hint: If you have a problem selecting inside the table, select the push button and move the cursor right, to position in the table cell. Enter two * characters. Now right click and use *Table Items / Add Rows...*
 - d. Delete the place holder characters.
2. Set up the New Section push button properties as shown:

| Property | Value |
|---------------------|--|
| Caption | New Section |
| left_relative_image | icons/normal/16/star_16.png |
| On_click_WamName | iiiSecAdd |
| On_click_wrname | AddSect |
| submitExtraFields | Field Name: STDREENTRY
Field Value: N |

- a. Adjust the width of the push button to display the caption as one line.
- b. Save your changes.

Your completed **Begin** page should look like the following:

Select a Department

| Section Code | Section Description | Address line 1 |
|--------------|---------------------|-------------------------|
| AB | ABCDEFGHIJKLMNQRST | aAbBcCdDeEfGhHiIjJkKlLm |
| AB | ABCDEFGHIJKLMNQRST | aAbBcCdDeEfGhHiIjJkKlLm |
| AB | ABCDEFGHIJKLMNQRST | aAbBcCdDeEfGhHiIjJkKlLm |

- Save these changes and run the **Begin** WebRoutine for WAM iiiSecMaint to test iiiSecAdd.
 - Adding a new section should return to the *Select a Department* web page and display a list of sections for the department concerned.
 - Trying to add a section with invalid or missing data should redisplay the *Add Section* web page, with error messages.

Summary

Important Observations

- A working list is displayed on the web page as a table
- Columns in the list may be input capable or output only.
- Columns in the list table may have weblets applied to them.
- Like fields, lists may be mapped for *output, *input or *both.
- The clickable image weblet can return a current row field and value and a STDREENTRY value to the WebRoutine it calls.
- A second WAM can be invoked simply by specifying the on_click_wamname property.
- The AutoComplete weblet can be implemented to invoke a response WebRoutine in an existing WAM, as used in exercise [WAM050 - A Section Maintenance Application](#).

Tips & Techniques

- Display, update and delete logic can easily be included in a single WebRoutine by using a CASE loop for field STDREENTRY
- Note that anchor weblets can also be used on fields in a browselist, to select a row – see later exercise.

What You Should Now Know

- How working lists are handled in the web page.
- How to select an entry from a browselist.
- How to design an application with a number of WAMs.

WAM055 - Using LANSAs Debug

Objectives

To learn how to use interactive debug with WAMs.

Initially this exercise assumes you are running your WAMs locally. A later step will demonstrate running debug as the WAM executes on the IBM i server.

To achieve these objectives you will complete the following:

- [Step 1. Get Started with Debug](#)
- [Step 2. Use Breakpoints](#)
- [Step 3. Use Break on Value Condition](#)
- [Step 4. Use Debug when the WAM is running on the server](#)
- [Summary](#)

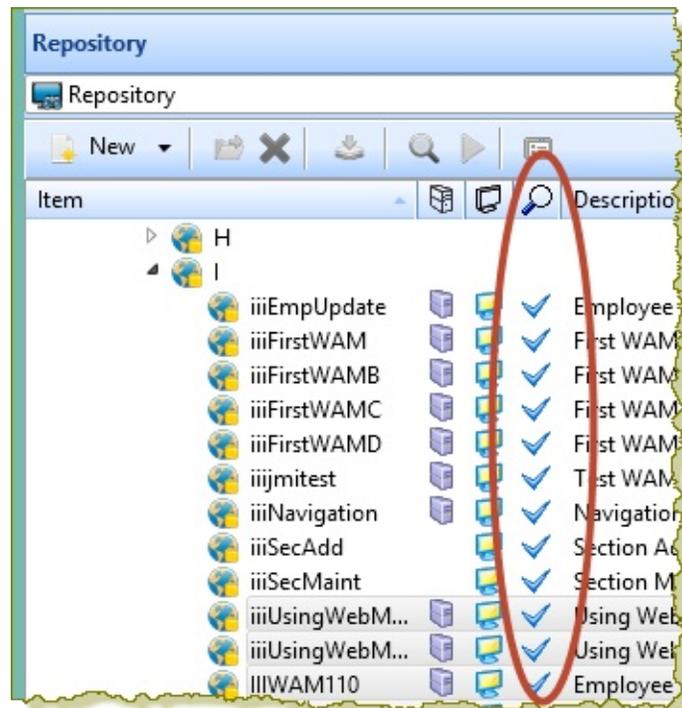
Before You Begin

- In order to complete this exercise, you must have completed exercise WAM050.

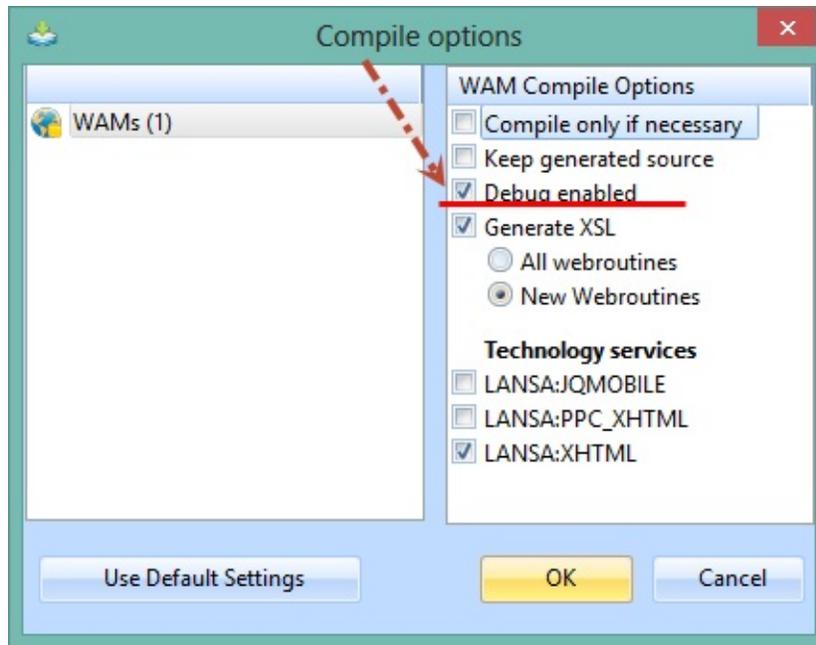
Step 1. Get Started with Debug

This exercise uses **iiiSecMaint - Section Maintenance**, which must be compiled with debug enabled.

Note: You can check if your WAM is debug enabled in the *Repository* tab or *Last Opened* tab in the LANSa Editor.

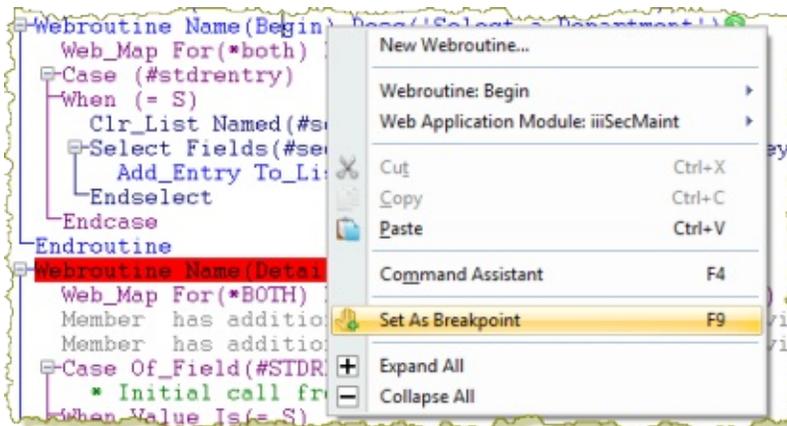


If necessary use the *Verify / Compile* menu option to recompile your WAM debug enabled.



Step 2. Use Breakpoints

1. Open WAM iiiSecMaint in the editor. Set a breakpoint on the **WebRoutine name(Begin)** statement. To set the breakpoint, select the line and then use **F9** or use the context mouse menu option *Set As Breakpoint* option. Breakpoint lines are highlighted in red.

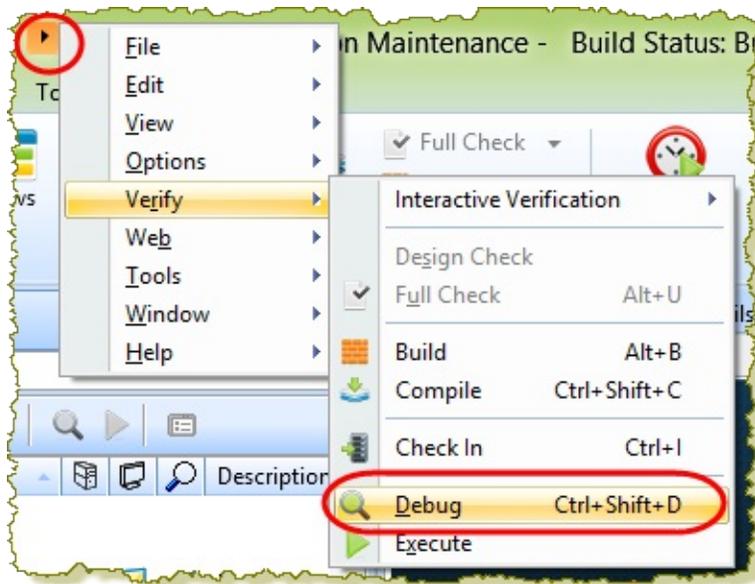


Your source code should now look like the following:

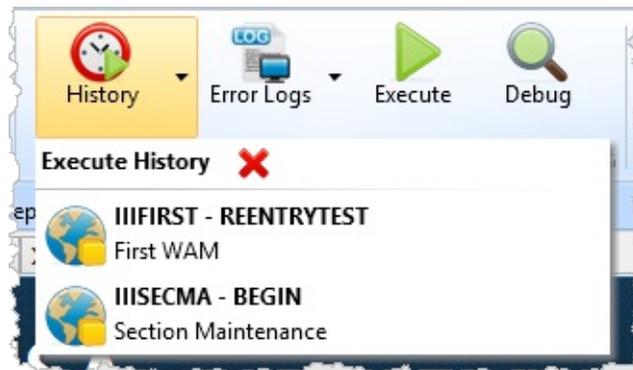


2. Open the WebRoutine **Begin** in *Design*. Run the WAM in debug mode using

 the **Debug Run in Debug** button on the *Home* ribbon, or use the *Verify / Debug* menu option from the *Toolbar Menu* button:



Note: You can use the *History* button on the Home ribbon, to run a WAM/WebRoutine executed earlier.



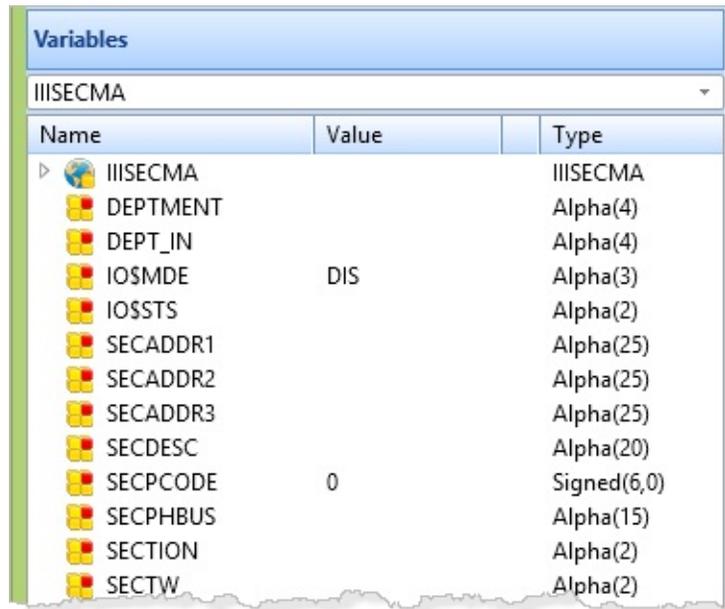
- When the WAM starts in debug mode in the browser, it will immediately switch focus to the LANSABuilder Editor, because WebRoutine **Begin** is run immediately, and it has a breakpoint defined, the breakpoint WebRoutine statement will be highlighted in yellow. This is the statement that is about to be executed as shown:

```

Web_Map For(*both) Fields((#stdreentry *hidden))
Webroutine Name(Begin) Desc('Select a Department')
Web_Map For(*both) Fields(#dept_in #sectlist)
Case (#stdreentry)
When (= S)
Clr_List Named(#sectlist)
Select Fields(#sectlist) From_File(sectab) With_Key(#dept_in)
Add_Entry To_List(#sectlist)
Endselect
Endcase
Endroutine

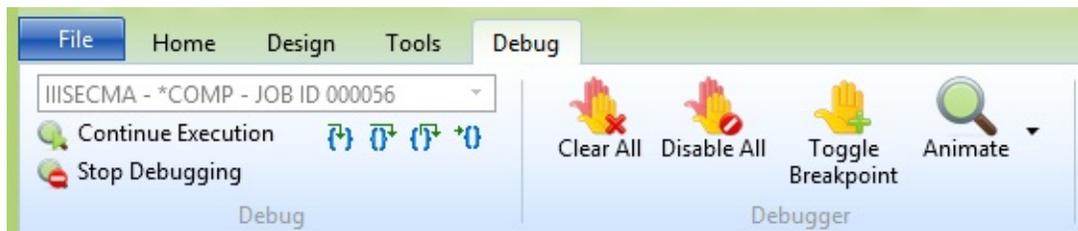
```

Note: in debug, program variables are shown on the *Variables* tab, in the left hand side pane, including working lists.

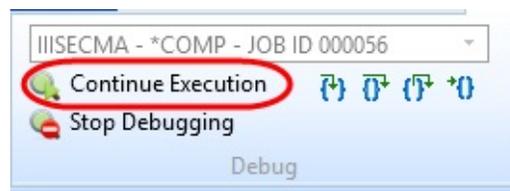


| Name | Value | Type |
|------------|-------|-------------|
| IIISECMA | | IIISECMA |
| DEPARTMENT | | Alpha(4) |
| DEPT_IN | | Alpha(4) |
| IOSMDE | DIS | Alpha(3) |
| IOSSTS | | Alpha(2) |
| SECADDR1 | | Alpha(25) |
| SECADDR2 | | Alpha(25) |
| SECADDR3 | | Alpha(25) |
| SECDESC | | Alpha(20) |
| SECPCODE | 0 | Signed(6,0) |
| SECPHBUS | | Alpha(15) |
| SECTION | | Alpha(2) |
| SECTW | | Alpha(2) |

A *Debug* ribbon is shown at the top of the editor:



- Use the *Continue Execution* button or F5, to run straight through this WebRoutine.

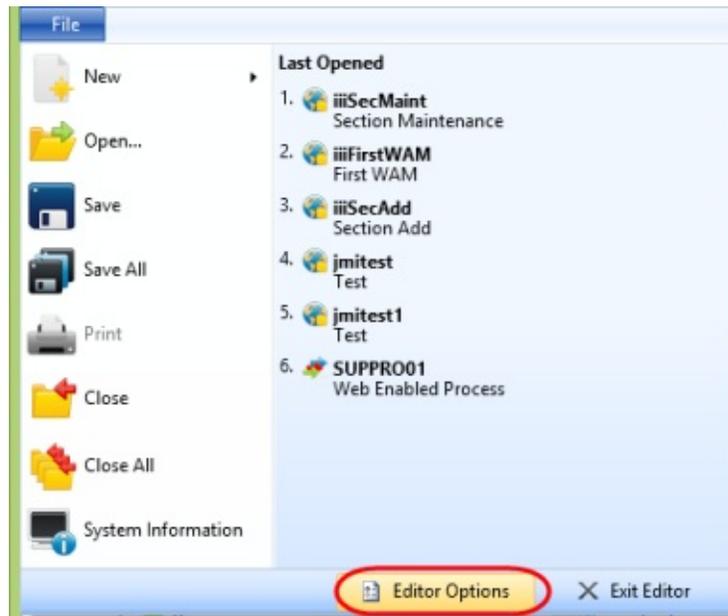


The initial page will be displayed in the web browser.

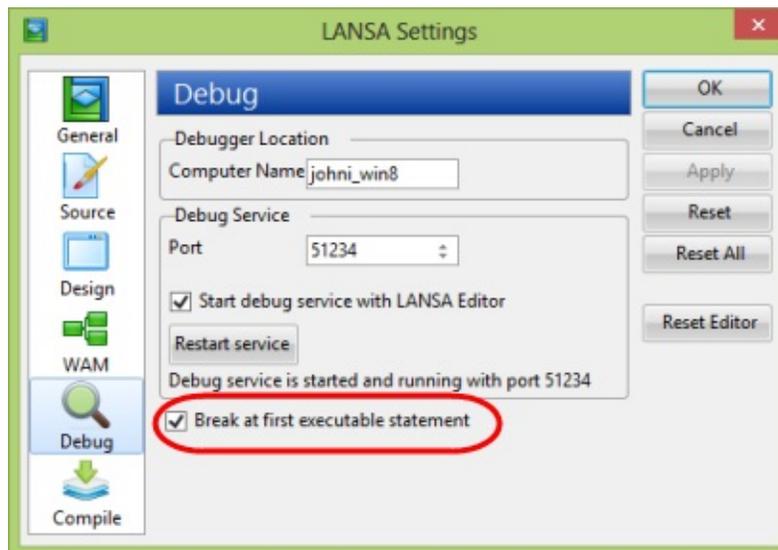
- Start typing into the department code input box and notice that debug stops at the AutoComplete WebRoutine:

```
Webroutine Name(AutoComplete) Response(*JSON)
Web_Map For(*input) Fields(#dept_in)
Web_Map For(*output) Fields((#depts *json))
#dept_in := #dept_in.substring( 1, 1 ).upperCase
Clr_List Named(#depts)
Select Fields(#deptment) From_File(deptab) With_Key(#dept_in) Get
Add_Entry To_List(#depts)
Endselect
Endroutine
```

This happened because Debug has a general setting to *Break at first executable statement*. To check this, select *Editor Settings* on the *File* menu:



c. Select the Debug icon.



- d. Uncheck the *Break at first executable statement* setting and click OK. Press F5 to continue running the WAM.
- e. With a valid department such as ADM or AUD entered, click the *Select* button. Focus will return to the WebRoutine statement for Begin in the editor and it will be highlighted in yellow as before.
- f.  Use the *Step Into* button or F8, to step through the WebRoutine one line at a time. If necessary use F5 to run through to the end of the Begin WebRoutine.
- g.  Use the *Run to Cursor* button to run all code up to the cursor position. To try this, press F5 to run through to the end of the WebRoutine. Enter a department code and click the *Select* button. In the Editor, position the cursor on the following statement in the Begin WebRoutine. Click the *Run to Cursor* button. You should run all code up to the SELECT statement and then stop.

SELECT FIELDS(#sec_list) FROM_FILE(sectab) WITH_KEY(#deptment

- h.  Use the *Step Out* button to run the code up to next breakpoint. In the editor set the SELECT statement to be a breakpoint and run the Begin WebRoutine again. Now use the *Step Out* button to execute the routine up to the SELECT statement. You could also have used the F5 key to achieve this. Use F8 to run around the SELECT loop. Notice the changing values of the variables shown in the *Variable* tab. That is, the value of variables DEPARTMENT, SECTION, SECDESC etc. If necessary now use F5 to run to the end of the WebRoutine.
- i.  Use the *Step Over* button to execute the procedure called by the current line and break at the line following the current line. *Step Over* is identical to *Step Into* except when the current statement contains a call to a procedure, *Step Over* executes this procedure as a unit and then steps to the next statement in the current procedure.

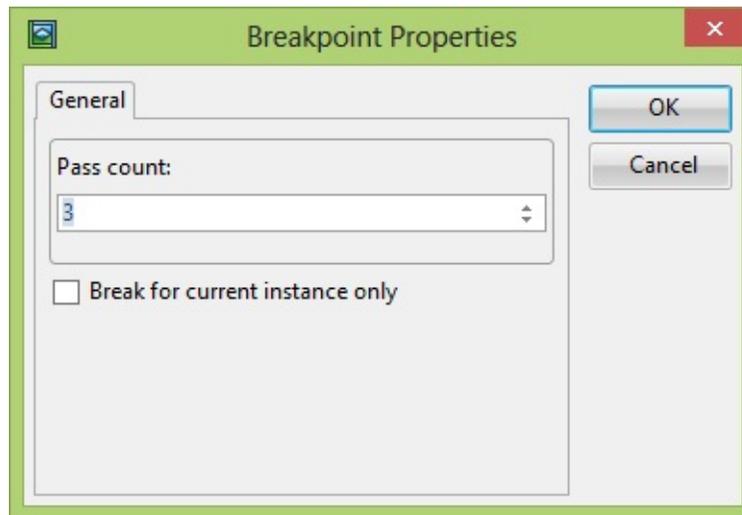


- j. Use the *Toggle Breakpoint* button on the *Debug* ribbon. In the web browser, enter a department code and click the *Select* button. In the LANSAs Editor, clear the breakpoint on the SELECT statement. To do this, select the SELECT line, and use the F9 key, or the *Toggle Breakpoint* button on the *Debug* ribbon to clear this breakpoint.

4. Set a breakpoint on the line

`ADD_ENTRY TO_LIST(#sec_list).`

Use the right mouse menu and select  *Breakpoint Properties*, and set the pass count to 3 and press OK.

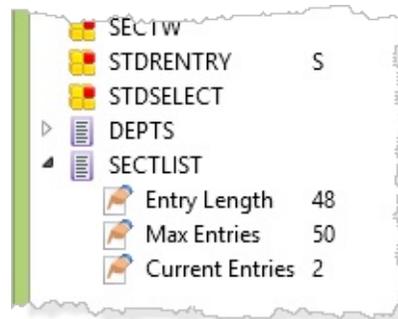


5. Remove the breakpoint from the Begin WebRoutine statement.

a. Press **F5** twice to run to the end of the WebRoutine.

b. When focus returns to the LANSAs Editor, expand the **SECT_LIST** entry in the *Variables* tab.

Note: The list SECT_LIST contain 2 entries currently.



Also notice the *Breakpoint* tab shows a passcount of 3 for this breakpoint. i.e. the `ADD_ENTRY` statement is about to be executed for the third time.

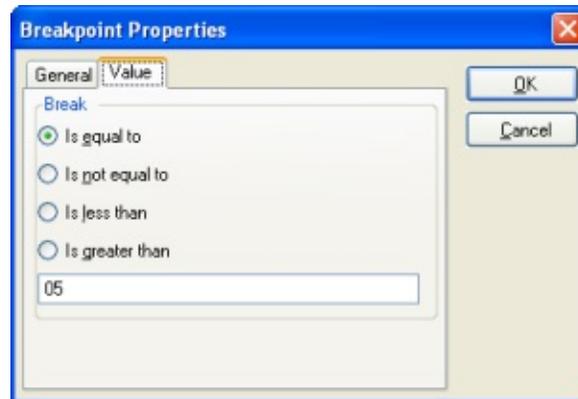
| Breakpoints | Pass Count | Hit Count | Line Number | Statement Text |
|-------------|------------|-----------|-------------|---------------------|
| IIISECMA | 0 | 1 | 14 | Webroutine Name(E |
| IIISECMA | 3 | 3 | 20 | Add_Entry To_List(# |

If the *Breakpoints* tab is not visible, select it from the *Home* ribbon /



Views gallery.

- d. Press **F5** to continue running. Note that each time the program breaks, 2 more entries have been added to the list (the third entry is about to be added).



5. Click OK to save the condition and press F5 to continue running the WAM. The breakpoint will next occur when SECTION has the chosen value.

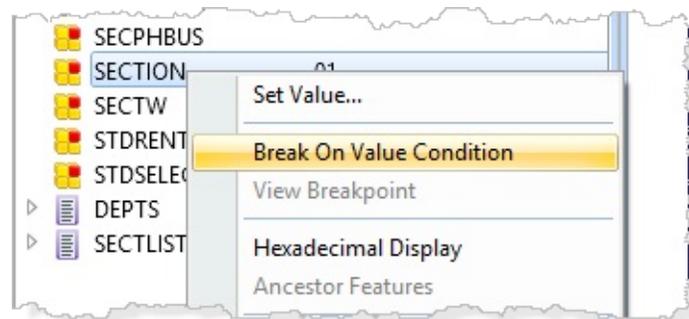
Step 3. Use Break on Value Condition

A breakpoint may have a variable value associated with it, meaning that that debug will break at this statement only when the value condition is true. To set a break on value condition you must first run the WAM in debug mode with a breakpoint set for the statement required. While running in debug. Select the *Variable* tab, select a variable and define a *Break on value* condition for this variable. This condition will be saved until it is removed. Continue running the WAM in debug mode and this statement will now break only when the variable's *Break on value condition* is true.

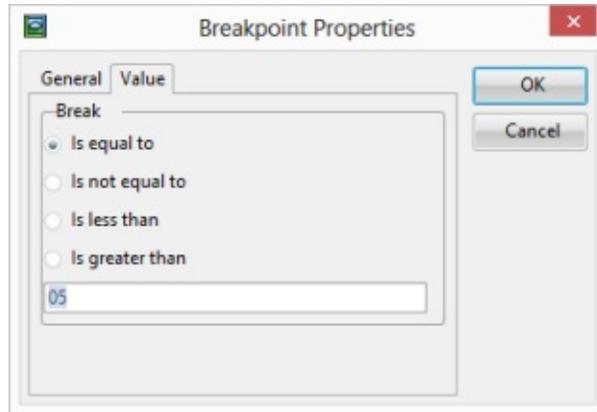
At this point your WAM iiiSecMaint should have a breakpoint defined on the statement:

Add_Entry To_List(#sectlist)

1. First run the WAM normally (not in debug mode) and review the list of section codes for the department you are using, for example ADM. Decide which section code value you want to break on, for example SECTION = 05.
2. Run the WAM again, this time in debug mode, enter a department code and click the Select button.
3. When debug breaks on the ADD_ENTRY statement select the field SECTION on the Variables tab and use the context menu to set the Break on Value condition:

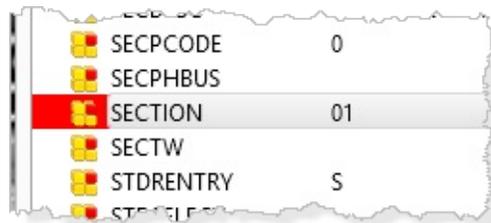


4. Set the condition to SECTION = 05 (or your chosen value).



5. Click OK to save the condition and press F5 to continue running the WAM. The breakpoint will next occur when SECTION has the chosen value.

The variable is highlighted on the *Variables* tab to show it has a *Break on Value* setting.



Step 4. Use Debug when the WAM is running on the server

To complete this step you must be using a Slave Workstation installation of Visual LANSA with a Master Repository on an IBM i server.

Debug with the WAM running on the server, requires communication from the developer's PC to the IBM i server and from the IBM i server to the developer's PC.

Visual LANSA Debug is a service that is started when VL starts. See *File / Editor Options* and select *Debug*:



This shows that the debug service is running on Port 51234.

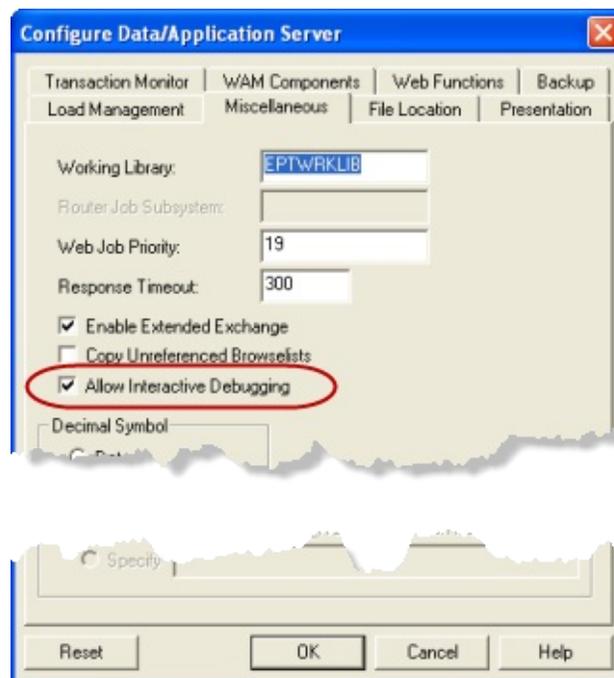
The IBM i server's communication to the developer's PC may rely on a locally defined Domain Name Server (DNS) to resolve the developer's PC Name to an IP address. Alternatively a routing entry must be defined for each developer's PC including the IP Address for the PC:

```
LC0ADMP407      Change Communications Routing Record

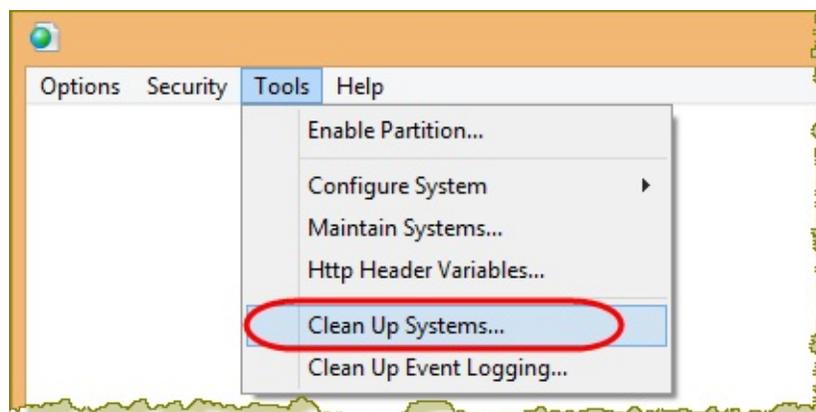
Partner LU name (Host) . . . . . A7800511
Fully qualified name . . . . . 10.44.10.51
-----
Communications method . . . . . SOCKET
Communications module . . . . .
Connection identifier . . . . . 4548
Packet size . . . . . 00000
TCP_NODELAY TCP/IP socket option . YES
Enable IPv6 . . . . . NO
```

To access this screen use the LANSA/CONFIGURE IBM i command and select, COMMS_EXTENSIONS followed by COMMS_ROUTING_RECORDS. See the *LANSA for iSeries User Guide* for detailed information.

LANSA Web running on the IBM i must be configured to allow interactive debug. This setting enables you to ensure that debug can only operate on your development or test system. The *LANSA Web Administrator* enables this setting to be enabled:

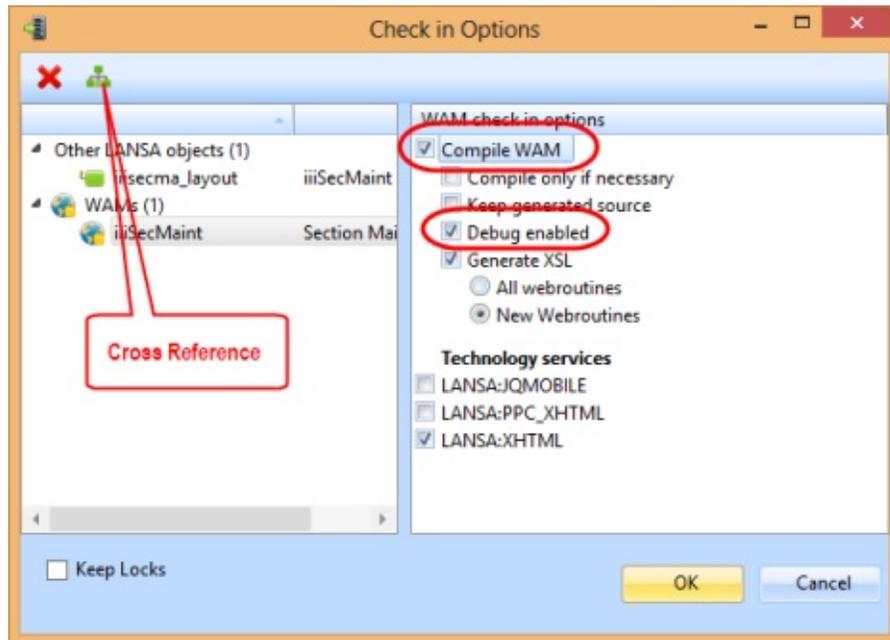


Restart LANSA Web after changing this setting. The Web Administrator Clean Up option will restart LANSA for the Web.



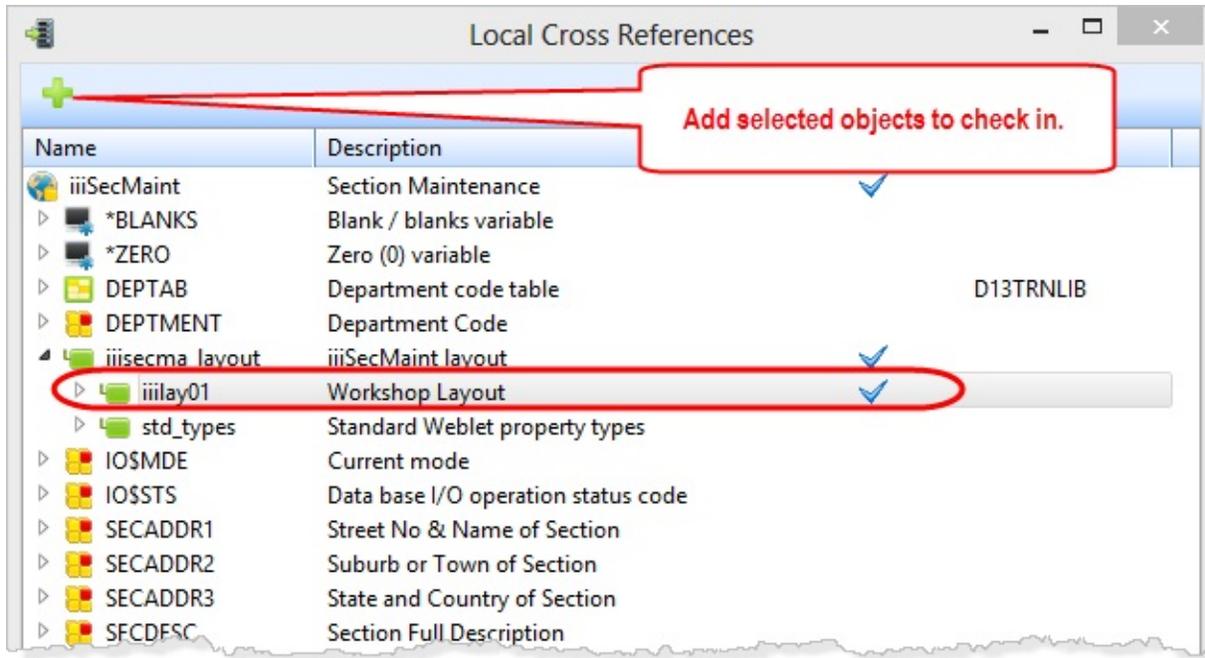
1. Check in your WAM **iiiSecMaint**. Its WAM layout, **iiisecma_layout** and the

common WAM layout **iiilay01** will be included automatically. Check this by selecting the WAM and using the *Cross Reference* dialog.



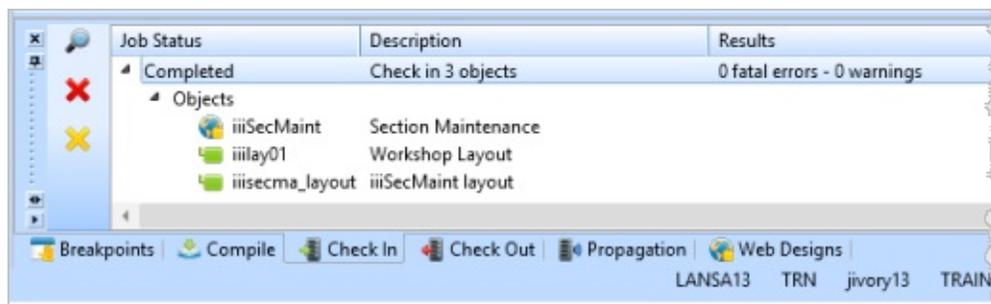
Make sure you check the options to compile the WAM and make it debug enabled.

2. In the *Local Cross References* dialog expand the WAM layout. iiSecma_layout.
 - a. Select the common layout, iiilay01 and click the green cross button as shown, to add it to the list of objects to be checked in.

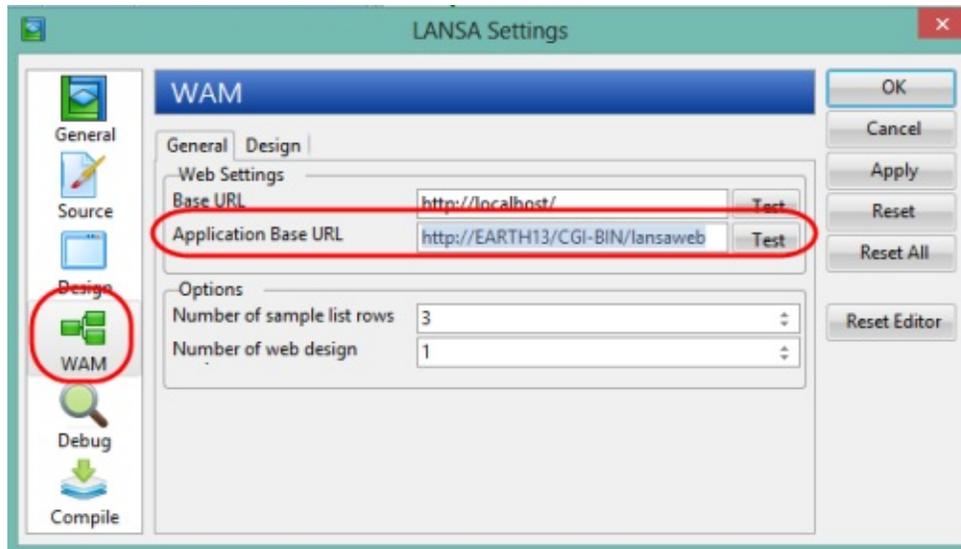


If required, you can select any other locally defined object and click the green cross button as shown to include these in the check in.

- Review the Check-in tab and ensure that your check-in and compile was successful.



- You now need to change your WAM execution settings, so that when you run the WAM from the Design view, it is run on the IBM i server.
 - From the File tab, open the *Editor Options* dialog and select the WAM settings.
 - Change the Application Base URL setting to point to the server name for your LANSAP IBM i system. Alternatively specify an IP Address.



- c. Use the *Test* button to check your entry is correct.
- d. Click OK to save your changes.

When you run your WAM it will now execute on the IBM i server.

5. From the *Design* view, Run your Begin WebRoutine to prove you can run the WAM on the IBM i server.
6. You should currently have the `ADD_ENTRY` statement set as a break point. Run your WAM again, this time in Debug mode. You will now be able to debug your WAM exactly as before, when you were running the WAM locally.

Summary

Important Observations

- Your Visual LANSA system, PC and LANSA for the Web must be correctly configured to support interactive debug of WAMs. Refer to the [Interactive Debugging](#) in the *Web Administration Guide* for set up details.

Tips and Techniques

- For debug support, make sure that the local Windows user, that is the default user for web jobs, is a member of the Group LANSA and Administrator
- In the LANSA Web Administrator for the local system, use the *Configure Data/Application Server* to make sure that the *Allow Interactive Debug* option is checked.
- For remote debug you need to use *LANSA Web Administrator* for the IBM i server to *Allow Interactive Debug*. If you have a local DNS then no other changes are needed. Without a local DNS you need to configure the *LANSA Listener* on the IBM I, adding an entry for each developer PC name and IP Address.
- Debug is supported via a Windows service running on the developer PC, which is started with Visual LANSA.

What You Should Know

- WAMs can be debugged interactively when run locally and also remotely when the WAM is run on the server (IBM i or Windows).

WAM060 - Employee Maintenance using Advanced Weblets

Objectives

To introduce you to a number of new weblets and techniques.

LANSA
Advanced Software Made Simple

Department: ADMINISTRATOR DEPTXX Section: INTERNAL ADMIN SRV Search

| Employ Number | Full Name | Post/zip Code | Home phone Number | Business Phone Number |
|---------------|---------------------|---------------|-------------------|-----------------------|
| A1001 | JONES, BEN | 2001 | 799 5268 | 798 0543 |
| A1012 | PAUL, PATRICK | 2147 | 687 1717 | 222 2222 |
| A1013 | PATTISON, GEORGE | 2012 | 222 2222 | 212 3569 |
| A1015 | WOODS, BRADLEY | 2030 | 430 1236 | 798 0543 |
| A1020 | DOUGLAS, ADAM PETER | 2147 | 674 5310 | 639 5188 |
| A1021 | MCCULLY, DAVID | 2153 | 762 1321 | 159 6845 |
| A1025 | ROBINSON, MARY | 2005 | 126 3598 | 456 1852 |
| A1027 | MORRISON, ALAN | 2007 | 148 2365 | 489 2485 |
| A1111 | VEREY, WARREN PETER | 2345 | 958 4567 | 957 3188 |
| A1404 | MRS BRICK, GILL | 2090 | 3343 333 | 334 444 |
| A1509 | REDFORD, ROBERT | 2060 | 5559966 | 9573188 |
| A2000 | BROWN, JOE | 2100 | 01 888 | 01 888 |
| A2002 | BROWN, GEORGE | 2000 | 020 1212 | 020 8899 |

LANSA
Advanced Software Made Simple

Department: ADMINISTRATOR DEPTXX Section: INTERNAL ADMIN SRV Search

| Employ Number | Full Name | Post/zip Code | Home phone Number | Business Phone Number |
|---------------|---------------------|---------------|-------------------|-----------------------|
| A1001 | JONES, BEN | 2001 | 799 5268 | 798 0543 |
| A1012 | PAUL, PATRICK | 2147 | 687 1717 | 222 2222 |
| A1013 | PATTISON, GEORGE | 2012 | 222 2222 | 212 3569 |
| A1015 | WOODS, BRADLEY | 2030 | 430 1236 | 798 0543 |
| A1020 | DOUGLAS, ADAM PETER | 2147 | 674 5310 | 639 5188 |
| A1021 | MCCULLY, DAVID | 2153 | 762 1321 | 159 6845 |
| A1025 | ROBINSON, MARY | 2005 | 126 3598 | 456 1852 |
| A1027 | MORRISON, ALAN | 2007 | 148 2365 | 489 2485 |
| A1111 | VEREY, WARREN PETER | 2345 | 958 4567 | 957 3188 |
| A1404 | MRS BRICK, GILL | 2090 | 3343 333 | 334 444 |
| A1509 | REDFORD, ROBERT | 2060 | 5559966 | 9573188 |
| A2000 | BROWN, JOE | 2100 | 01 888 | 01 888 |
| A2002 | BROWN, GEORGE | 2000 | 020 1212 | 020 8899 |

Details Skills

Employee Number: A1001
Employee Surname: JONES
Employee Given Name(s): BEN
Street No and Name: 144 Frog
Suburb or dept: PYMBLE.
State and Country: NSW.
Post / Zip Code: 2001
Home Phone Number: 799 5268
Business Phone Number: 798 0543

Save

- Dynamic Select Boxes for department and sections are linked. The sections combo box is repopulated when department changes.
- The Search button populates a list of employees on the left hand side.
- The page is divided into two resizable areas by a Vertical Splitter weblet.
- A Details WebRoutine is called by selecting the hyperlink on employee number (an Anchor weblet)
- The Details WebRoutine outputs to the area on the right hand side defined by a Nav Panel.

- A Tab Pages weblet enables employee details and a list of skills to be shown on the Navigation panel.
- A Save button on each tab page enables employee details or skills to be updated.
- A Dynamic select box weblet is also used in skill code column of the employee skills list. A *New Skill* button adds a blank skill entry at the top of the list and the *Save* button then inserts a new skill.

To achieve these objectives you will complete the following:

- [Step 1. Create WAM iiiEmpMaint – Employee Maintenance](#)
- [Step 2. Set up the ShowPage web page design](#)
- [Step 3. Complete the ShowPage web page design](#)
- [Step 4. Define the Details WebRoutine](#)
- [Step 5. Extend the Details WebRoutine for update](#)
- [Step 6. Extend the Details WebRoutine to add new employee skill](#)
- [Step 7. Control which Tab is redisplayed](#)
- [Step 8. Replace Date Acquired with a Date field \(Optional\).](#)
- [Step 9. Change Grade to a Dropdown list \(Optional\)](#)
- [Summary](#)

Before You Begin

Complete all preceding exercises in this workshop.

Step 1. Create WAM iiiEmpMaint – Employee Maintenance

1. Create a new WAM:

Name: iiiEmpMaint

Description: Employee Maintenance

Layout Weblet: iiilay01

2. Begin by defining the lists needed to support the main page WebRoutine, **ShowPage**.

- Define a working list, DEPTS for fields in the file DEPTAB
 - Define a working list, SECTS for section code and description from file SECTAB
- These lists will support Dynamic Select Box weblets for department and section codes.
- Define a working list, EMPLOYEES for fields EMPNO, FULLNAME, POSTCODE, PHONEBUS, and PHONEHME. All fields should be defined with an output attribute. This list of employees will be displayed on the left hand side of the vertical splitter weblet.
 - Define a Group_by, EMPS for fields EMPNO, SURNAME, GIVENAME, POSTCODE, PHONEBUS, PHONEHME
 - Map the field STDREENTRY for both, as a hidden field

Your code should look like the following:

```
Function Options(*direct)
Begin_Com Role(*EXTENDS #PRIM_WAM) Layoutweblet('iiilay01')
* Support web page ShowPage
Def_List Name(#depts) Fields(#deptment #deptdesc) Type(*Working)
Def_List Name(#sects) Fields(#section #secdesc) Type(*Working)
Def_List Name(#employs) Fields((#empno *out) (#fullname *out) (#postcode
Group_by Name(#emps) Fields(#empno #surname #givename #postcode #pho
Web_Map For(*both) Fields((#stdreentry *hidden))
End_Com
```

3. Define a ShowPage WebRoutine.

- Define a web_map for output for lists DEPTS, SECTS, mapped as JSON data and list EMPLOYEES.

- Define a web_map for both, for fields DEPARTMENT and SECTION
- Define a CASE loop for field STDREENTRY
- When = S
 - Select fields in EMPS from file PSLMST1 with key DEPARTMENT and SECTION
 - Fullname = Surname + Givenname
 - Add entry to EMPLOYS
 - End select
- End Case

Your code should look like the following:

```

WebRoutine Name(ShowPage)
Web_Map For(*output) Fields((#depts *json) (#sects *json) #employs)
Web_Map For(*both) Fields(#deptment #section)
Case (#stdreentry)
When (= S)
Select Fields(#emps) From_File(pslmst1) With_Key(#deptment #section)
#fullname := #surname + ', ' + #givenname
Add_Entry To_List(#employs)
Endselect
Endcase
Endroutine

```

4. Define a method routine **BuildDepts** to populate list DEPTS.

The routine should:

- Clear the list
- Select all entries from file DEPTAB, and add to list
- Position to the first entry

5. Define a method routine **BuildSects** to populate list SECTS.

The routine should:

- Define an input parameter, named i_dept, based on DEPARTMENT
- Clear the list
- Select entries from file SECTAB with a key of I_DEPT and add to list

- Position to the first entry

Your new code should look like the following:

```

Mthroutine Name(BuildDepts)
Define_Map For(*input) Class(#deptment) Name(#i_dept)
Clr_List Named(#depts)
Select Fields(#depts) From_File(deptab)
Add_Entry To_List(#depts)
Endselect
If (#i_dept = *blanks)
Get_Entry Number(1) From_List(#depts)
Else
#deptment := #i_dept
Endif
Endroutine
Mthroutine Name(BuildSects)
Define_Map For(*input) Class(#deptment) Name(#i_dept)
Clr_List Named(#septs)
Select Fields(#septs) From_File(sectab) With_Key(#i_dept)
Add_Entry To_List(#septs)
Endselect
Get_Entry Number(1) From_List(#septs)
Endroutine

```

6. Invoke these method routines at the end of the WebRoutine **ShowPage**

Your code should look like the following. New code is shown in red.

```

.....
#com_owner.BuildDepts I_Dept(#deptment)
#com_owner.buildsepts I_Dept(#deptment)
Endroutine

```

7. The Dynamic Select Box weblet will be set up to invoke a response WebRoutine to re-populate the list SECTS when field DEPARTMENT changes.

This will require a WebRoutine, UpdSects defined as follows:

- The WebRoutine must be defined with a Response() keyword with the value *JSON
- Map for input field DEPARTMENT

- Map for output the list SECTS as *JSON data
- Invoke the BuildSects method routine

Your code should look like the following:

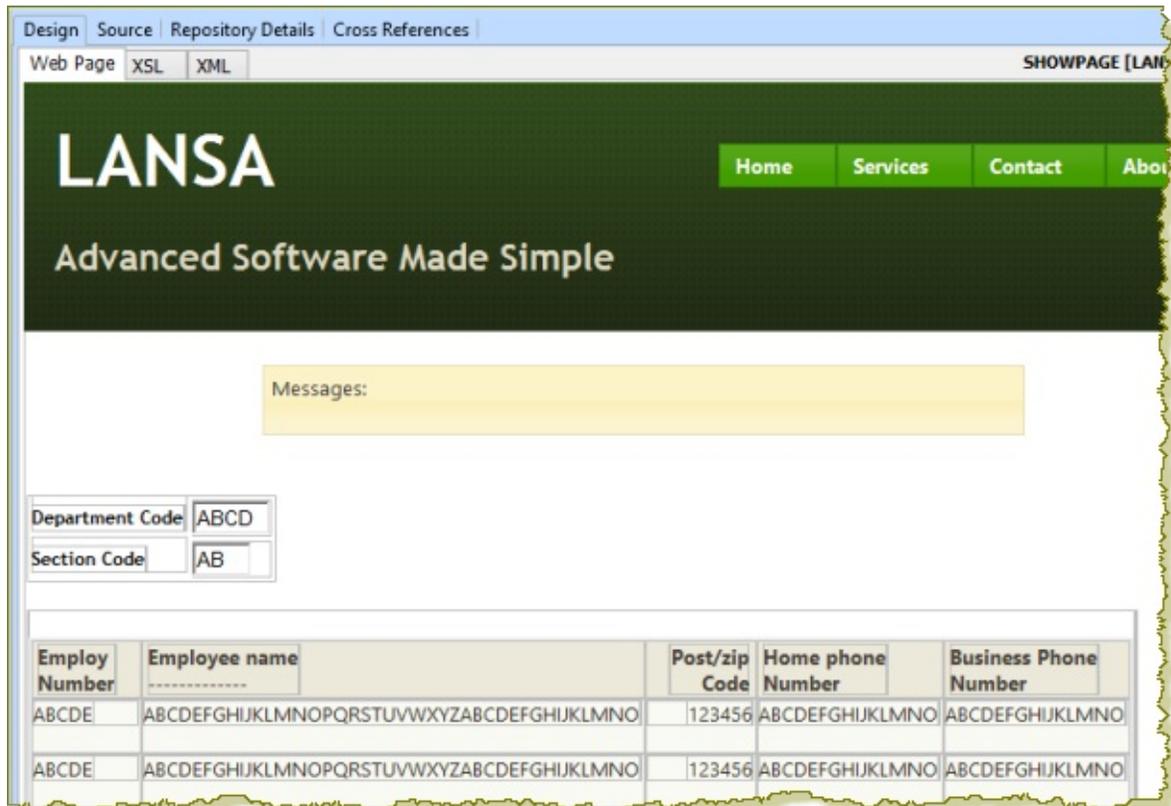
```
WebRoutine Name(updsects) Response(*JSON)
Web_Map For(*input) Fields(#deptment)
Web_Map For(*output) Fields((#sects *json))
#com_owner.BuildSects I_Dept(#deptment)
Endroutine
```

8. Compile your WAM.

Step 2. Set up the ShowPage web page design

1. Open the **ShowPage** WebRoutine in the *Design* view.

It should look like the following:



2. Click anywhere in the table containing department code and using the context menu select the *Table Items / add Columns...* option, to add 3 columns.
 - a. Move the section code and its label into the 3rd and 4th columns (top row), using drag and drop.
 - b. Drop a push button with image into the 5th cell, top row
 - c. Click in the bottom row, and use the context menu, *Table Items / Delete Row* to delete this row.
 - d. Click in the cell containing the *section code* label, if necessary use the cursor keys to ensure you are in the table cell (<td> tag) and use the *Details* tab to change its *class* to caption.
 - e. Select the *department code* label and delete it. Type Department: into the

cell

- f. Select the *section code* label and delete it. Type Section: into this cell.
- g. Select each of the new cells using the cursor keys, and delete the * place holder characters.

3. Select the push button and set up its properties on the Details tab:

| Property | Value |
|---------------------|------------------------------------|
| caption | Search |
| left_relative_image | icons/normal/16/zoom_16.png |
| on_click_wname | ShowPage |
| submitExtraFields | Field Name: STDREENTRY |
| | Literal Value: S |

- 4. Select the table and changes its *Align* property to center. You should be able to click on a corner of the table to select it.
- 5. Save your changes

Your page should look like the following:



6. Drag and drop a Dynamic Select Box onto the field DEPARTMENT, set up its properties as shown:

| Property | Value |
|-----------|-------------------|
| listname | DEPTS |
| codeField | DEPARTMENT |

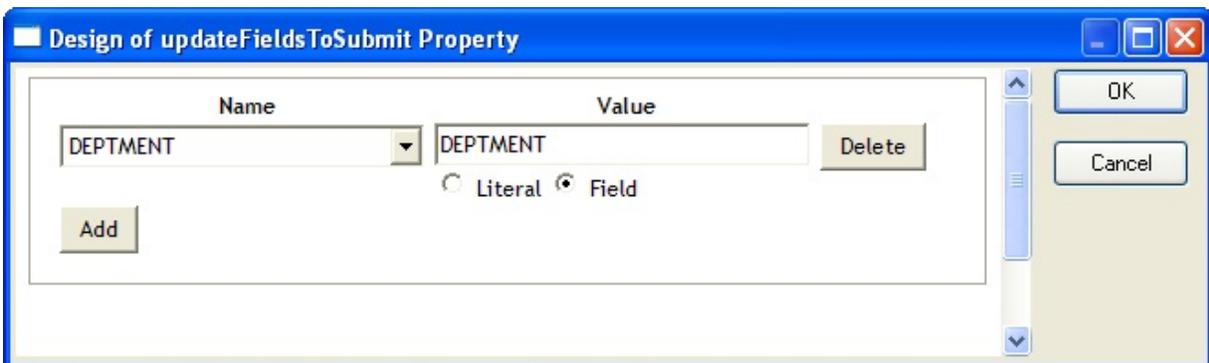
| | |
|--------------|-----------------|
| captionField | DEPTDESC |
|--------------|-----------------|

7. Drag and drop a Dynamic Select Box onto the field SECTION and set up its properties as shown:

| Property | Value |
|----------------------|--------------------------------|
| Listname | SECTS |
| codeField | SECTION |
| captionField | SECDESC |
| updateWrName | updsects |
| updateOnFieldChange | DEPARTMENT |
| updateFieldsToSubmit | Field Name: DEPARTMENT |
| | Value: Field DEPARTMENT |

Complete the *updateFieldsToSubmit* property by clicking on the Ellipsis  button in the value column, to open the *Design of...Property* dialog.

- Select DEPARTMENT in the *Name* column.
- Select the *Field* checkbox and enter **DEPARTMENT** in the *Value* column.



- Save your changes.
- Execute the **ShowPage** WebRoutine in the browser.

- Changing selected department should refresh the section's dropdown list.
- The Search button should populate the list of employees.

The screenshot shows the LANSA software interface. At the top, there is a dark green header with the text "LANSA" and "Advanced Software Made Simple". To the right of the header are two buttons: "Home" and "Serv". Below the header is a search form with two dropdown menus: "Department: Administration" and "Section: Internal Admin". To the right of these dropdowns is a green "Search" button. Below the search form is a table with the following data:

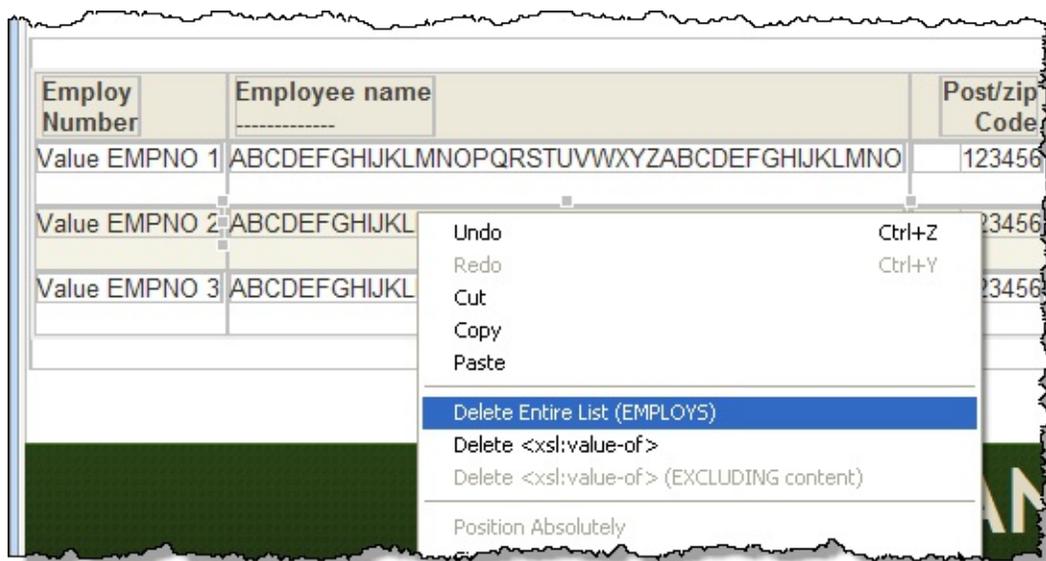
| Employ Number | Employee name | Post/zip Code | Home phone Number | Business Phone Number |
|---------------|-------------------|---------------|-------------------|-----------------------|
| A1001 | Jones, Shirley | 2001 | 799 5268 | 798 0543 |
| A1012 | Paul, Patrick | 2147 | 687 1717 | 222 2222 |
| A1013 | Pattinson, George | 2016 | 750 2562 | 212 3569 |
| A1015 | Woods, Bradley | 2030 | 450 1236 | 789 4562 |

Step 3. Complete the ShowPage web page design

In this step you will:

- Add a Vertical Splitter weblet to the page and move the employees list onto its left hand side.
- Add a Nav Panel to the right hand side of the vertical splitter and give this a name
- Add an Anchor weblet to the employee number column in the employees list and set the weblet up to invoke a **Details** WebRoutine and output to the nav panel.
- You will create the **Details** WebRoutine in the next step.

1. In this step you will temporarily remove the employees list from the page. With the **ShowPage** WebRoutine open in the Design view, select anywhere inside the employees list and use the context menu, to select the *Delete Entire List* option:



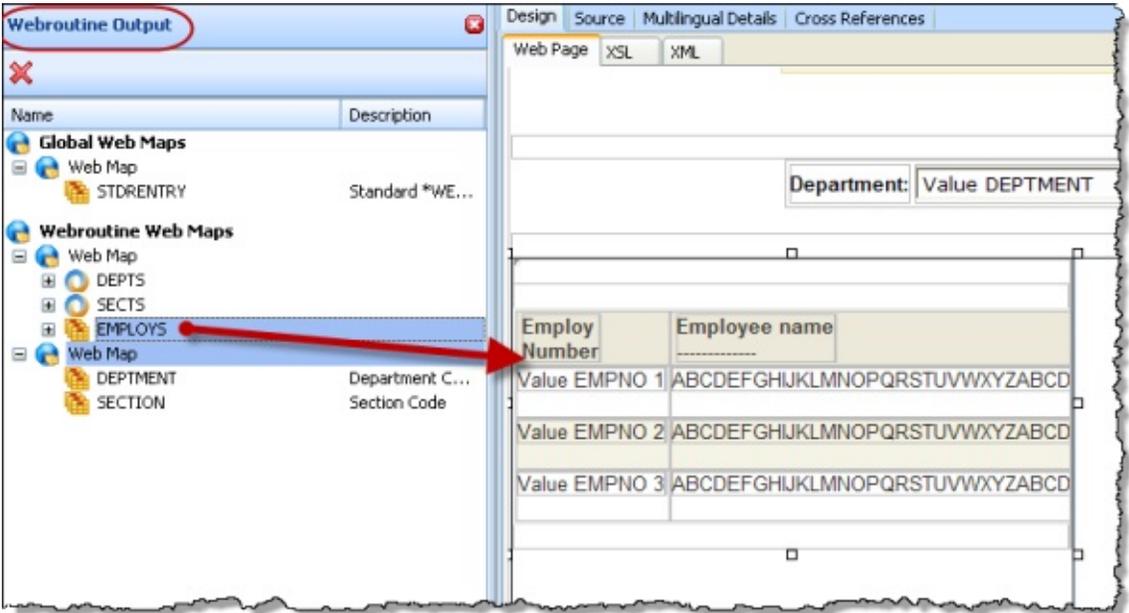
2. Drag and drop a Vertical Splitter onto the page.

Set up its properties as:

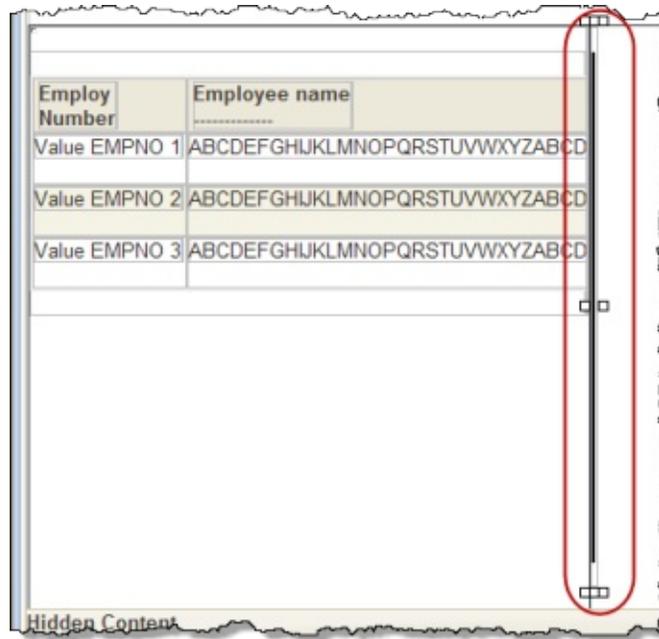
| Property | Value |
|----------|-------|
| Width | 100% |
| Height | 500px |
| | 40 |

Left_proportion_percent

3. Select the *WebRoutine Output* tab, and drag and drop the list EMPLOYS onto the left side of the Vertical Splitter:



- 4. Save your changes.
- 5. Drag and drop a Navigation Panel onto the right side of the vertical splitter.



With the Navigation Panel selected set up its properties:

| Property | Value |
|--------------|---------------|
| name | EmpDtl |
| border | none |
| border_width | 0px |
| height | 450px |

6. Drag and drop an Anchor weblet into the Employ Number column in the employees list. With the Anchor weblet selected, set up its properties as:

| Property | Value |
|--------------------|----------------|
| Currentrowhfield | EMPNO |
| Currentrownumvalue | \$EMPNO |
| Rentryvalue | D |
| On_click_wname | Details |
| target_window_name | EmpDtl |

7. Save your changes.

Step 4. Define the Details WebRoutine

The **Details** WebRoutine:

- Will be invoked by selecting an employee in the employees list, using the Anchor weblet (also known as a hyperlink). Output for the **Details** WebRoutine will be shown on the **ShowPage** web page.
 - Output will be displayed in the Navigation Panel on the right of the Vertical Splitter.
 - Will handle display and update of employee and employee skills data. It will also handle adding a new skill for the employee.
1. At the top of your WAM, begin by defining fields, lists or group_by that will be required to support the **Details** WebRoutine.
 - a. Define a working list, EMPSKLS for employee skills containing SKILCODE, GRADE, COMMENT, DATEACQ, DATEACQR and EMPNO . DATEACQR and EMPNO should be hidden.
 - b. Define a work field EMPNOW based on EMPNO
 - c. Define a group_by, EMPDATA containing fields EMPNOW, SURNAME, GIVENAME, ADDRESS1, ADDRESS2, ADDRESS3, POSTCODE, PHONEBUS and PHONEHME

Your code should look like the following:

```
Def_List Name(#empskls) Fields((#SKILCODE *out) #GRADE #COMMENT  
*  
Define Field(#empnow) Reffld(#empno)  
Group_By Name(#empdata) Fields((#empnow *out) #SURNAME #GIVENAI
```

2. Create a **Details** WebRoutine. Initially you will create a simple version of this WebRoutine and then extend it.
 - a. Map field EMPNO for both as a hidden field.
 - b. Map group_by, EMPDATA and list EMPSKLS for both. EMPSKLS should be mapped with a *private attribute. This means it will not be automatically shown on the web page.

Your code should look like the following:

```
WebRoutine Name(Details)  
Web_Map For(*both) Fields((#empno *hidden))
```

```
Web_Map For(*both) Fields(#empdata (#empskls *private))  
Endroutine
```

3. In this step you will add logic to your **Details** WebRoutine using a CASE loop for STDREENTRY (this is mapped globally as a hidden field).

Note the following:

- Field EMPNOW is shown on the Details web page as an output field. EMPNO is mapped as a hidden field and will be used for employee update. EMPNO is passed into the Details WebRoutine by the Anchor weblet.

Code your **Details** WebRoutine based on the following:

- Change EMPNOW to EMPNO
- Case loop on STDREENTRY
- When = D
 - Fetch fields in EMPDATA from file PSLMST with EMPNO
 - Clear list EMPSKLS
 - Select EMPSKLS from file PLSKSL with key EMPNO
 - Add entry to EMPSKLS
 - Endselect
- When = U
 - Update fields in EMPDATA in file PLSMT with key EMPNO. Go to next statement on validation error.
 - If status is not OK, issue an error message
 - Else
 - Issue an "employee changed" message including employee number.
 - Endcase

Your code should look like the following:

```
WebRoutine Name(Details)  
Web_Map For(*both) Fields((#empno *hidden))  
Web_Map For(*both) Fields(#empdata (#empskls *private))  
Web_Map For(*both) Fields((#skilcode *hidden))  
#empnow := #empno
```

```

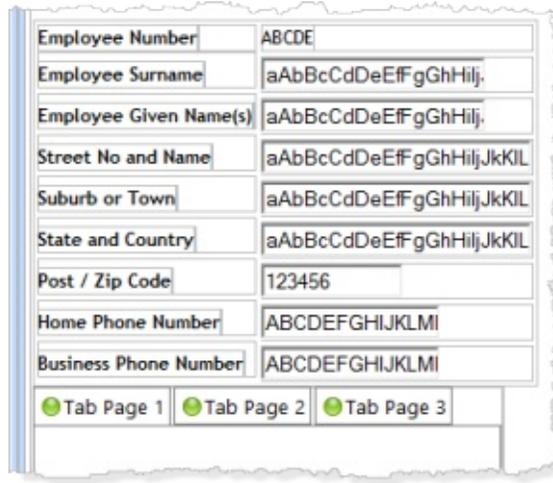
Case (#stdrentry)
When (= D)
Fetch Fields(#empdata) From_File(pslmst) With_Key(#empno)
Clr_List Named(#empskls)
Select Fields(#empskls) From_File(pslskl) With_Key(#empno)
Add_Entry To_List(#empskls)
Endselect
* update employee
When (= U)
Update Fields(#empdata) In_File(pslmst) With_Key(#empno) Val_Error(*next
If_Status Is_Not(*okay)
Message Msgtxt('Error occurred on employee update')
Else
Message Msgtxt(' Employee ' + #empno + ' was changed')
Endif
Endcase
Endroutine

```

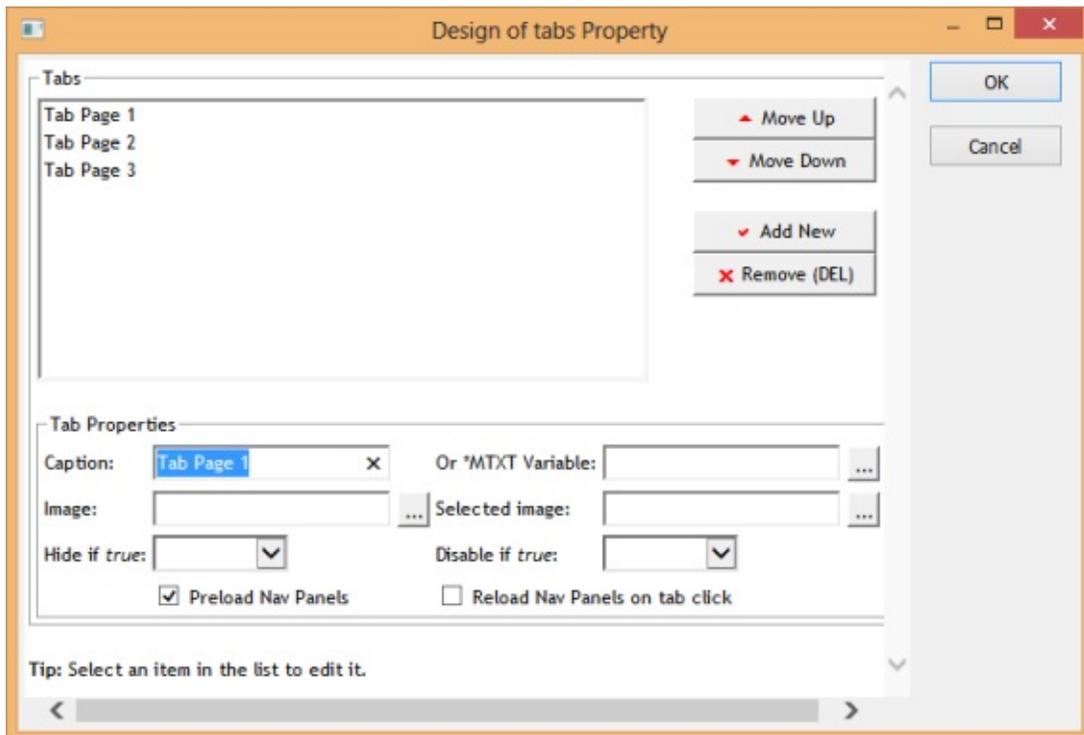
4. Compile your WAM and open the **Details** WebRoutine in the *Design* view.
Your web page should look like the following:

| | |
|------------------------|--------------------------|
| Employee Number | ABCDE |
| Employee Surname | ABCDEFGHIJKLMNOPQRS |
| Employee Given Name(s) | ABCDEFGHIJKLMNOPQRS |
| Street No and Name | aAbBcCdDeEfgGhHiIjJkKlLm |
| Suburb or Town | aAbBcCdDeEfgGhHiIjJkKlLm |
| State and Country | aAbBcCdDeEfgGhHiIjJkKlLm |
| Post / Zip Code | 123456 |
| Home Phone Number | ABCDEFGHIJKLMNO |
| Business Phone Number | ABCDEFGHIJKLMNO |

5. Drag and drop a *Tab Pages* weblet (not *Tab Pages (deprecated)*) onto the page.



a. Click on the Tab Pages weblet to select it and set up its *tabs* properties by clicking on the *Ellipsis* button, that will open the *Design of... Properties* dialog:



b. Select each tab page entry in the list and set up its properties:

| Property | Value |
|----------|-------|
| | |

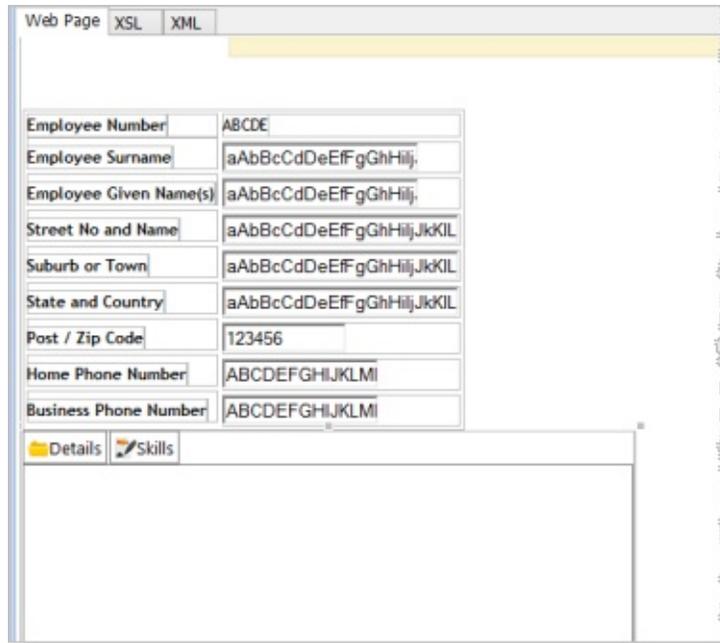
| | |
|---------|--------------------------------------|
| caption | Details |
| image | icons/normal/16/folder_16.png |

| Property | Value |
|-----------------|--|
| caption | Skills |
| image | icons/normal/16/contract_16.png |

- c. Delete Tab Page 3.
 - d. Save your changes by clicking OK.
6. Click on the *Tab Pages* weblet to select it and set its height and width properties:

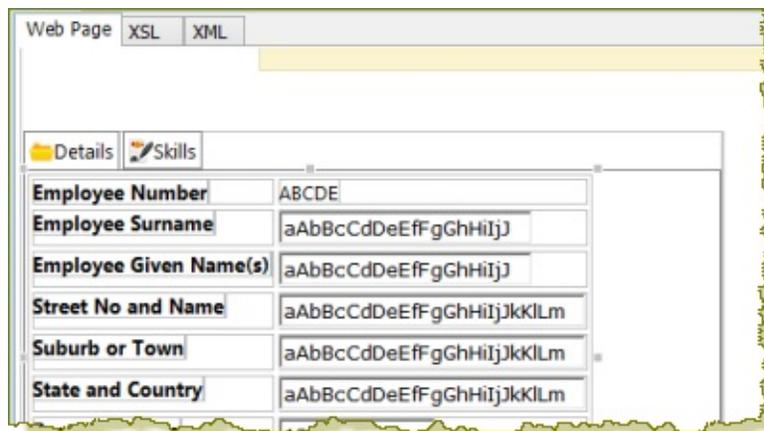
| Property | Value |
|-----------------|--------------|
| content_width | 450px |
| content_height | 400px |

7. Save your changes. Your page should now look like the following:



8. Click on a corner of the employees fields table to select it. Use the context menu to select *Cut*.
 - a. Click on the Tab Pages weblet to select it.
 - b. Click on the *Details* tab to ensure the correct tab sheet is selected.
 - c. Use the context menu to *Paste* the table into the tab folder *Details* tab.

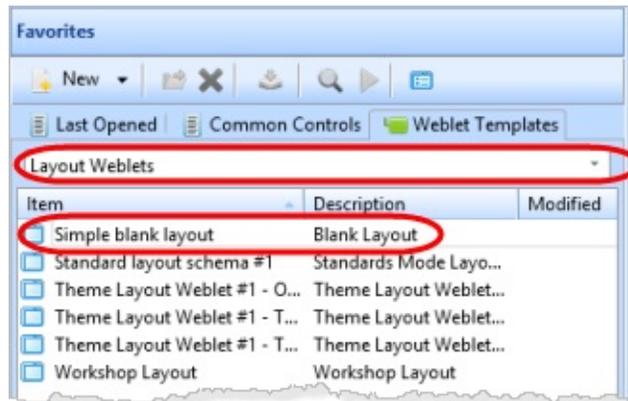
Your page should now look like the following:



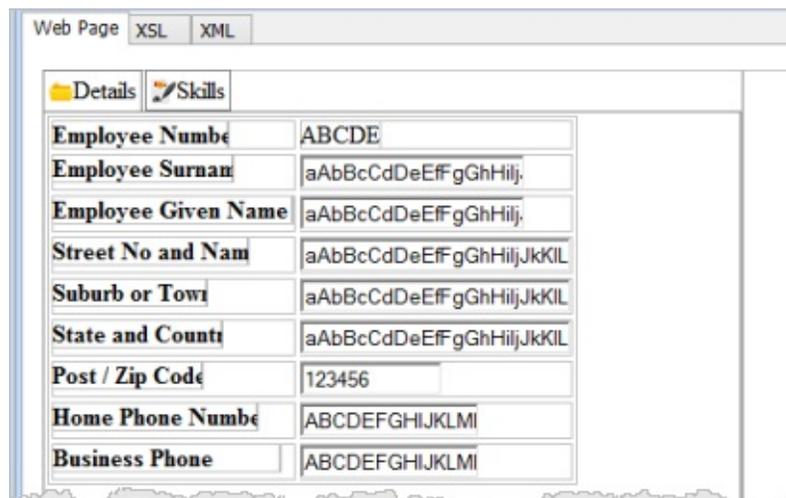
9. Save your changes.
10. Click on the Skills tab to select it and drag and drop a *jQuery Enabled Grid* onto the *Skills* tab page.
11. Select the Grid and set up its properties:

| Property | Value |
|----------|---------|
| Listname | EMPSKLS |

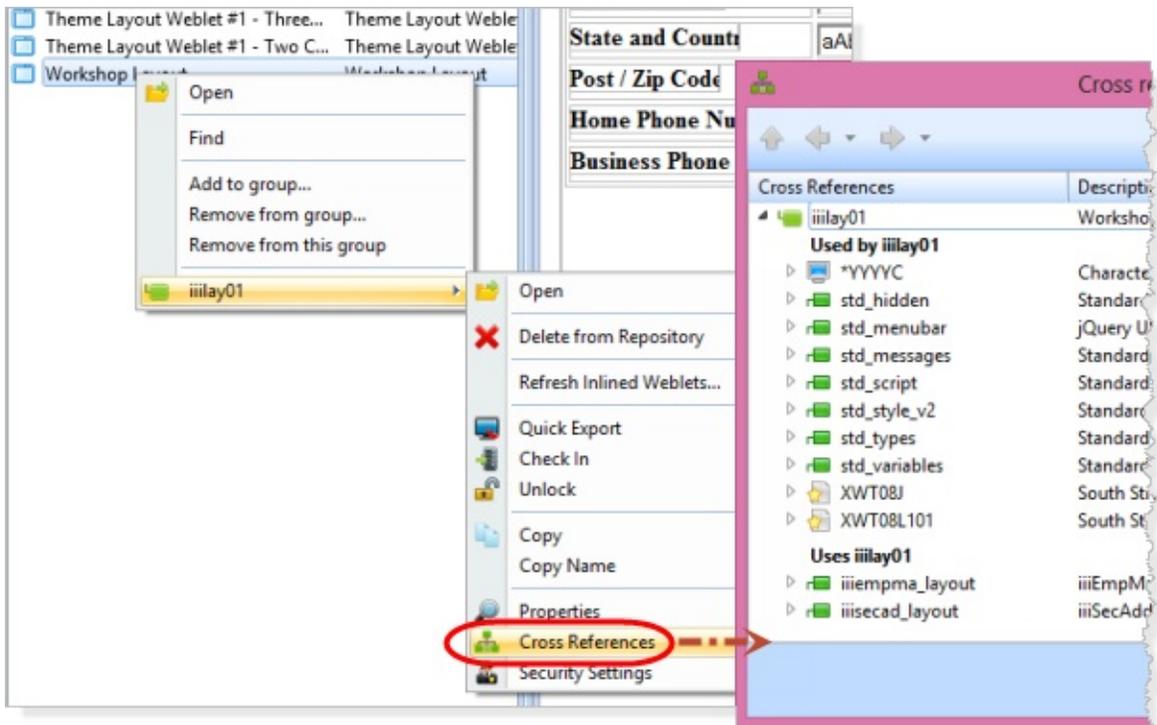
12. Click on the *Details* tab and save your changes. This will make *Details*, the default tab at run time.
13. The **Details** web page will be displayed inside the *Navigation Panel* on the **ShowPage** web page. The **Details** web page therefore requires a blank layout.
 - a. On the *Favorites / Weblet Templates* tab, using the dropdown at the top, select *Layout Weblets*:



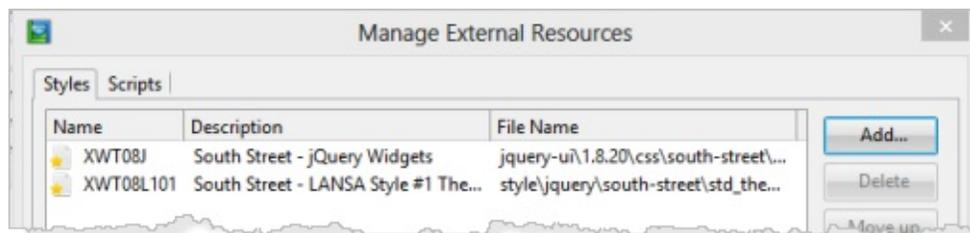
- b. Drag and drop the *Simple blank layout* onto the page. Your design should now look like the following:



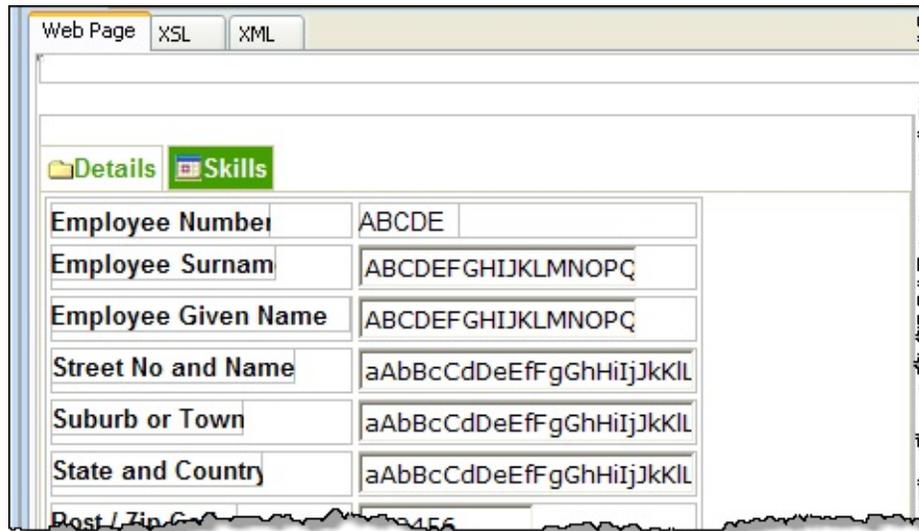
14. You need to ensure that this blank page adopts the same theme as your common layout **iiilay01**.
- On the *Favorites / Weblet Templates* tab, select *Layout Weblets* in the top dropdown list.
 - Right click on the layout *Workshop Layout* and use the context menu to select *Cross References*:



- Make a note of the *Style External Resources* being used. In this example these are XWT08J and XWT08L1.
Close this dialog.
- In the *Design* view for WebRoutine **Details**, on the Design ribbon, select the *External Resources* button. Use the *Add* button to add the two Styles required.



e) Your *Design* should now look like the following, but reflecting your chosen theme:



Note the Font used for the field labels.

Note also the background used for a non-selected tab. The color may not be shown in the *Design* view. Check your results in the next step.

15. Save your changes.

16. Execute the **ShowPage** WebRoutine in the browser to test your application:

- Selecting an employee should display details on the right side of the Vertical Splitter. The *Details* tab will be shown initially, as it was saved as the default.
- Selecting the *Skills* tab will display a grid of employee skills.
- Selecting another employee will display the *Details* tab again.
- Later you will enhance the application to remember the current tab and display it for the next employee selected.

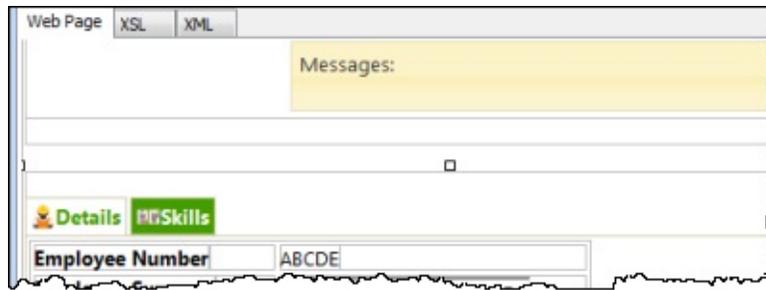
Step 5. Extend the Details WebRoutine for update

In this step you will:

- Make the **Details** tab handle update.
 - Extend the **Details** WebRoutine to handle update of Employee skills
 - Add a *Save* button to the *Skills* tab.
1. Open the **Details** WebRoutine in the *Design View*. The *Details* tab should be shown. If necessary select the *Details* tab and save your WAM to make this the default.
 2. On the *Details* tab, select anywhere in the table containing the fields and use the context menu to select *Table Items/Add Rows...* to add 1 row to the bottom of the table.
 3. Click in the left hand cell of the new row and set its *align* property to left.
 4. Drag and drop a *Push Button with Image* into the bottom left cell of the table. Set up the button as follows:

| Property | Value |
|---------------------|-----------------------------------|
| Caption | Save |
| left_relative_image | icons/normal/16/check_mark_16.png |
| on_click_wrname | Details |
| submitExtraFields | Field Name: STDREENTRY |
| | Literal value: U |

5. This step adds a *Message* weblet to this page:
 - a. Select the tab pages weblet. Move the cursor right and press enter. This should create a blank space above the tab pages weblet.
 - b. Drag and drop a *Messages* weblet into the space immediately above the *Tab Pages* weblet. Your design should now look like the following:



6. Select the Messages weblet and set its *target_window_name* property to **_top**. This will route messages from this page to the messages weblet on the page in which it is embedded.
Save your changes.
7. Retest your WAM. The update employee details logic has already been included in the **Details** WebRoutine. You should now be able to change employee data and save the changes. Try making an invalid change, such as a blank surname.
8. To handle update of employee skills the CASE loop in WebRoutine Details requires new logic.
 - The employee skills list EMPSKLS is mapped for both. The list should be read to process updates to the employee skills file PLSKLS.
 - The SKILCODE in EMPSKLS is mapped for *output. In order to update, this key value will need to be saved in a hidden field in the list.
 - When the list is read (SELECTLIST) the key value SKILCODE will be set from the hidden field.
 - The list contains a hidden field DATEACQR. This will be used to recognize list entries for existing records. This will be important when you extend the logic to handle insert of a new employee skill.
 - a. Define a work field SC based on field SKILCODE
 - b. Add field SC to list EMPSKLS as a hidden field. Your code should look like the following:
Define Field(#sc) Reffld(#skilcode)
 Def_List Name(#empskls) Fields((#SKILCODE *out) #GRADE
 #COMMENT #DATEACQ (#dateacqr *hidden) (#empno *hidden) (**#sc *hidden**)) Type(*Working) Entries(*max)
 - c. Change the existing logic which builds the employee skills list, to populate the hidden field SC. Your code should look like the following.

The new code is shown in red.

```
Select Fields(#empskls) From_File(pslskl) With_Key(#empno)
#sc := #skilcode
Add_Entry To_List(#empskls)
Endselect
```

d. Add new logic to update employee skills based on the following:

- When = S
- Read list EMPSKLS using SELECTLIST/ENDSELECT
- Change SC to SKILCODE
- If DATEACQR is not *zeroes

Update fields in list EMPSKLS in file PLSKL with key EMPNO and SKILCODE. Go to next line on validation error.

- End if
- End select
- If status is OK, issue message, employee 'Skills for nnnn were changed', end if, where nnnn is employee number.

Your code should look like the following:

```
When (= S)
Selectlist Named(#empskls)
#skilcode := #sc
If (#dateacqr *NE *zeroes)
Update Fields(#empskls) In_File(pslskl) With_Key(#empno #skilcode) Val_Er
Endif
Upd_Entry In_List(#empskls)
Endselect
If_Status Is(*okay)
Message Msgtxt('Skills for ' + #empno + ' were changed')
Endif
Endcase
```

8. Compile your WAM.
9. Open the **Details** WebRoutine in the *Design* view.
 - a. On the Skills tab, select the grid and move to the right using the cursor

keys and press enter. This will position the cursor immediately below the grid.

b. Use the context menu to insert a table with one row and two columns.

10. Click in each cell of the new table and change the align property to left.

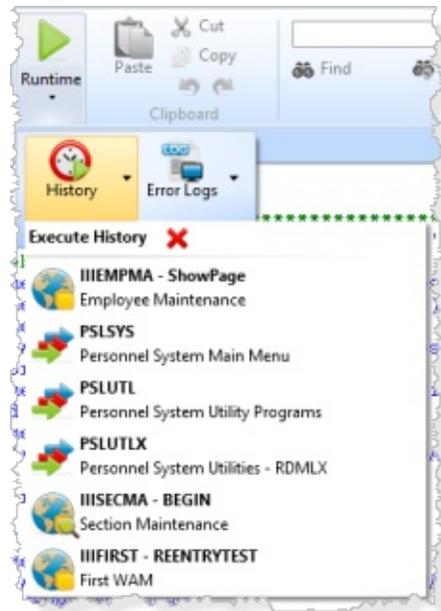
11. Drag and drop a Push Button with Image into the right hand cell and set it up as follows:

| Property | Value |
|---------------------|--|
| caption | Save |
| left_relative_image | icons/normal/16/check_mark_16.png |
| on_click_wname | Details |
| submitExtraFields | Field Name: STDREENTRY |
| | Literal Value: S |

12. Remove the * place holder characters from the right hand cell. Leave them in the left cell.

13. Click on the *Details* tab to select it and save your changes.

14. Re-test your WAM. Note that on the *Home* ribbon, you can use *History* button in the *Runtime* gallery, to re-run any VL component.



- You should be able to change any employee skills data and save changes with the *Save* button.
- Errors such as *Date Acquired = zero* or other validation errors will display messages in the messages weblet.
- After the Save button is processed the Details tab is redisplayed, because this is the default tab. You will fix this issue in a later enhancement.

Step 6. Extend the Details WebRoutine to add new employee skill

In this step you will extend the WebRoutine Details to insert a new employee skill.

- A *New Skill* push button on the Skills tab will refresh the grid with a blank first entry.
 - The SKILCODE will now be an input field in the list EMPSKLS and the hidden field SC will no longer be needed.
 - The SKILCODE column in the grid will be customized using a Dynamic select box.
 - The Dynamic select box will be populated with a list of skills from the table SKLTAB. A new method routine will build this list.
 - When the Skills Save button is processed, an employee skill will be inserted if the DATEACQR field is zero
1. Change the definition of list EMPSKLS so that SKILCODE is input capable, removing the hidden field SC. Your code should look like the following:

```
Def_List Name(#empskls) Fields(#SKILCODE #GRADE #COMMENT  
#DATEACQ (#dateacqr *hidden) (#empno *hidden)) Type(*Working)  
Entrys(*max)
```
 2. Delete the definition of field SC and remove all code which refers to it.
 3. Define a Group_by SKL_LIST for fields SKILCODE, GRADE, COMMENT, DATEACQ, DATAECQR. This will be used to clear employee skills list fields before adding a blank entry for insert.
 4. Define a working list SKILLS containing fields SKILCODE, SKILDESC. This list will mapped for output to populate the skill code Dynamic select box.

LANSA definition statements can be placed anywhere in your code. It is usual to place them at the top of the program code.

Your new code should look like the following:

```
Group_By Name(#skl_list) Fields(#skilcode  
#grade #comment #dateacq #dateacqr)  
Def_List Name(#skills) Fields(#skilcode #skildesc) Type(*Working)  
Entrys(*max)
```

5. Add a new When clause to the **Details** WebRoutine, that will handle a request from the *New Skill* button to add a blank entry as the first entry in the employee skills list. This will allow insert of a new employee skill.

The logic should be based on the following:

- When = N
 - Change SKL_LIST to default values
 - Add entry to EMPSKLS after *Start
 - Message 'Complete new skill in the first list entry'

Your new code should look like the following:

```
* add new top row to skills list
```

```
When (= N)
```

```
#skl_list := *default
```

```
Add_Entry To_List(#empskls) After(*START)
```

```
Message Msgtxt('Complete new skill in the first list entry')
```

6. When processing a list entry, a value of zero for Date Acquired (DATEACQR) means the entry is new.

Add the following new logic to the CASE loop, for when STDREENTRY = S.

New code is shown in red.

```
When (= S)
```

```
Selectlist Named(#empskls)
```

```
If (#dateacqr *NE *zeroes)
```

```
Update Fields(#empskls) In_File(pslskl) With_Key(#empno #skilcode) Val_Er
```

```
Else
```

```
Insert Fields(#empskls) To_File(pslskl) Val_Error(*next)
```

```
Endif
```

```
Endselect
```

```
If_Status Is(*okay)
```

```
Message Msgtxt('Skills for ' + #empno + ' were changed')
```

```
Endif
```

7. Create a method routine **BuildSkills** to populate the skills list SKILLS.
 - a. Clear the list SKILLS
 - b. Select all records from file SKLTAB and add entries to list SKILLS

c. Position to the first entry

Your code should look like the following:

```
Mthroutine Name(BuildSkills)
Clr_List Named(#skills)
Select Fields(#skills) From_File(skltab)
Add_Entry To_List(#skills)
Endselect
Get_Entry Number(1) From_List(#skills)
Endroutine
```

8. Add an output web_map for the SKILLS list to the **Details** WebRoutine as JSON data. Your code should look like the following:

```
Web_Map For(*output) Fields((#skills *JSON))
```

9. Invoke the BuildSkills method at the end of the **Details** WebRoutine. Your code should look like the following. New code is shown in red.

```
Endcase
```

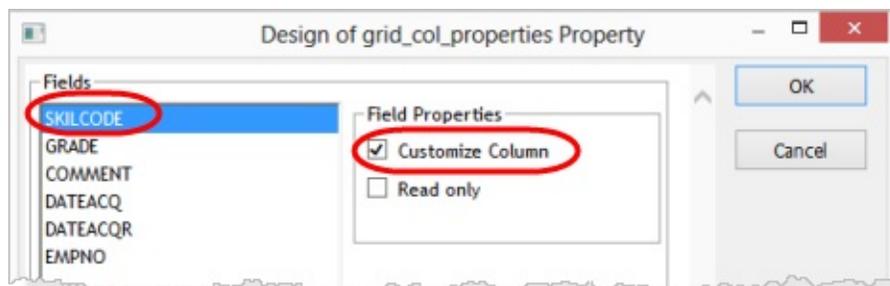
```
#com_owner.BuildSkills
```

```
Endroutine
```

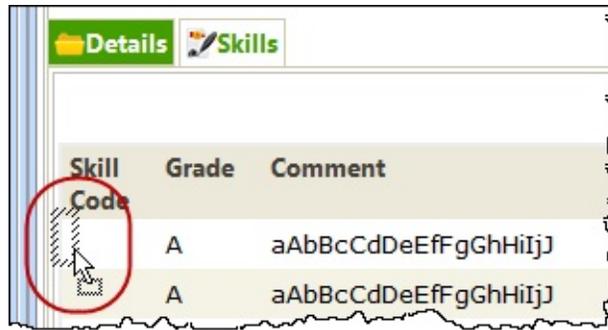
10. Compile your WAM.

11. Open the **Details** WebRoutine in the *Design* view.

12. Select the Skills tab, and select the Grid weblet. On the *Details* tab use the Ellipsis button for the *grid_col_properties* value to open the *Design of* dialog. With SKILCODE field selected, select the *Customize Column* check box and then click *OK* to close the dialog.



13. Drop a Dynamic select box into the Skill Code column. See the image below showing how the editor will highlight the column when the cursor is in the correct position.



14. Select the Dynamic select box and set up its properties as shown:

| Property | Value |
|--------------|-----------------|
| listName | SKILLS |
| codeField | SKILCODE |
| captionField | SKILDESC |

15. Add a push button with image weblet into the into the single row table below the grid. Set up the button properties as:

| Property | Value |
|---------------------|---------------------------------|
| caption | New Skill |
| left_relative_image | icons/normal/16/contract_16.png |
| on_click_wname | Details |
| submitExtraFields | Field Name: STDREENTRY |
| | Literal Value: N |

Adjust the width of the push button to display the caption as single line.

Remove the place holder characters from table cell.

16. Save your changes. Your design should look like the following:

| Skill Code | Grade | Comment | Date Skl Acquired | Acquire Empl Date Numl |
|------------|-------|--------------------|-------------------|------------------------|
| ABCDEFHIJ | A | aAbBcCdDeEffGhHiIj | 12/34/5 | 12/34/5 ABCD |
| ABCDEFHIJ | A | aAbBcCdDeEffGhHiIj | 12/34/5 | 12/34/5 ABCD |
| ABCDEFHIJ | A | aAbBcCdDeEffGhHiIj | 12/34/5 | 12/34/5 ABCD |

17. Retest your WAM. Select an employee and select the Skills tab. The Dynamic select box for skills should display the correct skill description in each row.

- Click the New Skill button to add a new row at the top of the skills grid.
- Select a skill and complete the grade, comment and date acquired column. Note that the date acquired is a six digit date in the format DD/MM/YY.
- Click the Save button to process the skills in the EMPSKLS list and update or insert to the employee skills file.
- Validation errors will display messages at the top of the page.

jl_bs@hotmail.com

Step 7. Control which Tab is redisplayed

At present the Details WebRoutine has no control over whether the Details or Skills tab is redisplayed. This can easily be achieved by introducing a field that is mapped to set the `tab_index` property.

1. Define a one character field `TABINDEX`.

```
Define Field(#tabindex) Type(*char) Length(1)
```

2. In the **Details** WebRoutine extend the `web_map` for `EMPNO` to include `TABINDEX` as a hidden field.

```
Web_Map For(*both) Fields((#empno *hidden) (#tabindex *hidden))
```

3. In the `CASE` loop, set the `TABINDEX` in each `When` clause, as follows:

```
When = D #Tabindex := '1'
```

```
When = U #Tabindex := '1'
```

```
When = N #Tabindex := '2'
```

```
When = S #Tabindex := '2'
```

4. Recompile your WAM.
5. Open the **Details** WebRoutine in the *Design* view. Select the Tab Pages weblet and set the `selected_tab_index_field` to `TABINDEX`. (Select from the dropdown).
6. Save your changes
7. Re-test your WAM. When working with the Skills tab, the Skills tab should now be redisplayed after the *New Skill* or the *Save* push button has been used.

Note: This is a limited implementation only, using simply the `tab_index_field` property. Using a hidden Nav Panel on each tab page, it is possible to return current tab index to a WebRoutine. This allows a design which supports selecting a different employee and always displaying the last tab page used (Details or Skills).

Step 8. Replace Date Acquired with a Date field (Optional).

In this step you will replace the Date Acquired column in the employee skills list (EMPSKLS) with field STD_DATEX. This is a Date type field and has a default visualization of a Date Picker weblet.

Note: The default Date Picker visualization for field STD_DATEX will be automatically implemented when the field is included on the page, or is a column in a list weblet.

When used as a column in the Grid weblet the default visualization is not recognised and you will need to add the jQuery UI DatePicker weblet to this column.

1. Change the definition of working list EMPSKLS as shown:

Changes are shown in red.

```
Def_List Name(#empskls) Fields(#SKILCODE #GRADE #COMMENT #std_
Entrys(*max)
```

Note: The virtual field DATEACQ will still be required in the list to update the employee skills record. The real field DATEACQR cannot be used for update.

2. In WebRoutine **Details**, when the employee skills list is populated, set up field STD_DATEX with the value of field DATEACQ:

Changes are shown in red.

```
When (= D)
#tabindex := '1'
Fetch Fields(#empdata) From_File(pslmst) With_Key(#empno)
Clr_List Named(#empskls)
Select Fields(#empskls) From_File(pslskl) With_Key(#empno)
#std_datex := #dateacq.asdate( DDMMYY )
Add_Entry To_List(#empskls)
Endselect
```

3. When a new list entry is added for insert, set up field STD_DATEX using current date, and give GRADE a default value (see [Step 9. Change Grade to a Dropdown list \(Optional\)](#)).

Changes are shown in red.

```
When (= N)
#tabindex := '2'
#skl_list := *default
```

```
#std_datex := #std_datex.Now
```

```
#grade := P
```

```
Add_Entry To_List(#empskls) After(*START)
```

```
Message Msgtxt('Complete new skill in top row')
```

4. When employee skills are updated, set up field DATEACQ:

Changes are shown in red.

```
When (= S)
```

```
#tabindex := '2'
```

```
Selectlist Named(#empskls)
```

```
If (#dateacqr *NE *zeroes)
```

```
#dateacq := #std_datex.asnumber( DDMMYY )
```

```
Update Fields(#empskls) In_File(pslskl) With_Key(#empno #skilcode)
```

```
Val_Error(*next)
```

```
If_Status Is_Not(*okay)
```

```
Message Msgtxt('Errors occurred on skills updates')
```

```
Endif
```

```
Else
```

```
Insert Fields(#empskls) To_File(pslskl) Val_Error(*next)
```

```
If_Status Is_Not(*okay)
```

```
Message Msgtxt('Insert failed')
```

```
Else
```

```
Message Msgtxt('Skills for ' + #empno + ' were changed')
```

```
Endif
```

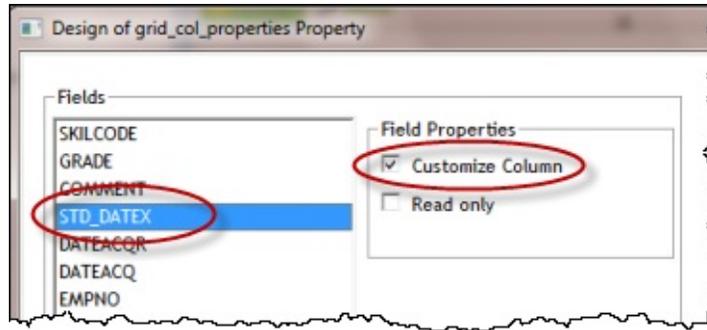
```
Endif
```

```
Endselect
```

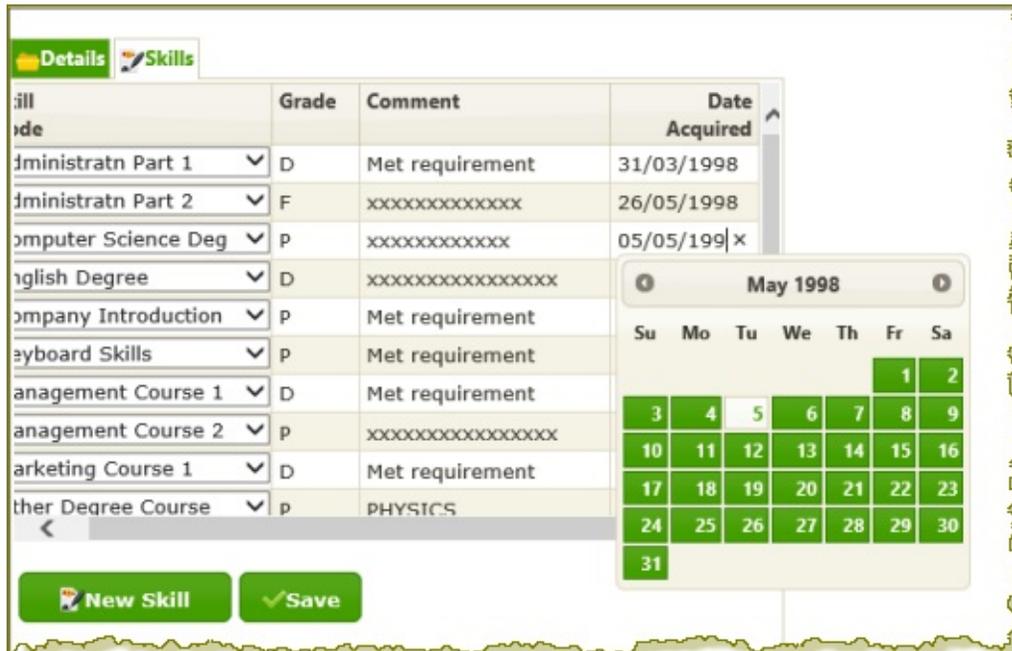
5. At the top of the program, override the column heading for STD_DATEX, with the following:

```
Override Field(#std_datex) Colhdg('Date' 'Acquired' '')
```

6. Compile your WAM.
7. Open the **Details** WebRoutine in the *Design* view and select the Skills tab.
 - a. Select the grid. Open the *Design of...* dialog using the ellipsis button for the *grid_column_properties* value. Select Customize Column for field STD_DATEX and click *OK*.



- b. On the *Weblet Templates* tab, select jQuery UI in the dropdown. Drop a jQuery UI Datepicker weblet into the new Date Acquired column.
 - d. With the jQuery UI Datepicker selected, on the Details tab, change the *dateFormat* to **dd/mm/yyyy**.
 - c. Select the Details tab page and save your changes.
8. Retest your WAM. Date Acquired in the Skills grid will now be displayed in dd/mm/ccyy format. Selecting a Date Acquired will display the calendar prompt.



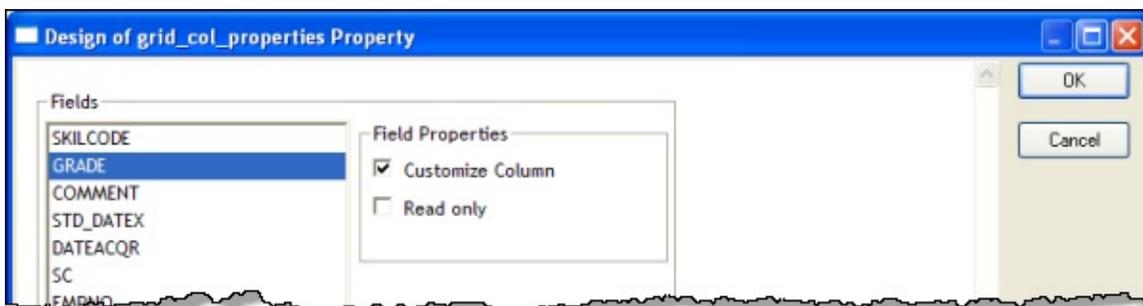
- 9. Change a Date Acquired and Save the changes. Redisplay the employee skills to show the date was updated correctly.
- 10. Add a new skill. The Date Acquired column should initially contain current

date.

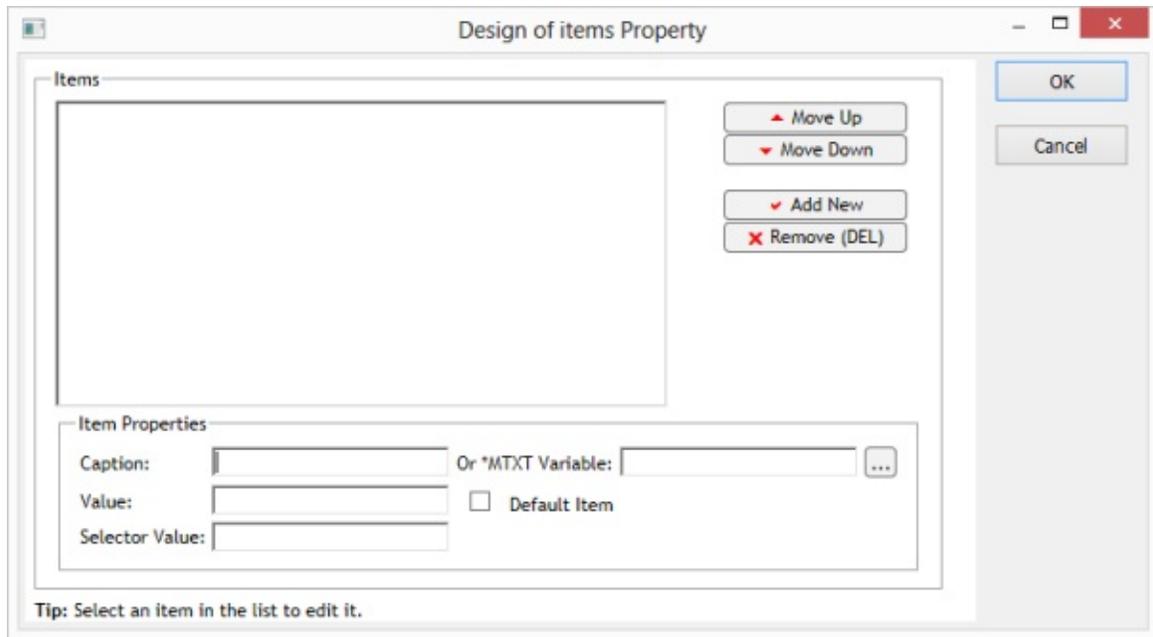
Step 9. Change Grade to a Dropdown list (Optional)

In this step you will replace the Grade column in the Skills grid with a combo box weblet and set it up with a hard coded list of values.

1. Open the **Details** WebRoutine in the Design view.
 - a. Select the Skills tab.
 - b. Select the Employee Skills Grid.
 - c. On the *Details* tab, select the `grid_col_properties` value and use the Ellipsis button to display the *Design of... Properties* dialog.



- d. Select the `GRADE` field and select the `Customize Column` checkbox. Click *OK* to close the dialog.
 - e. On the *Weblet Templates* tab, select `Standard Weblets` in the dropdown and drag and drop a `combo box weblet` into the `GRADE` column.
2. Select the `combo box weblet`, to set up its properties.
 - a. Leave the `name` property as `@id` This is correct for identifying a field in a `Grid weblet` (different to a list).
 - b. Note that the `value` property is `.` (a dot). This is correct and means value from current row/column.
 - c. Select the `items` property value and click the *Ellipsis* button to open the *Design of...* dialog.



d. In the Item Properties group box, set up a list of Captions and Values based on the following table:

| Caption | Value |
|----------------|--------------|
| Pass | P |
| Distinction | D |
| Merit | M |
| Fail | F |

Click *Add New* to add each entry to the list.

e. Click the *OK* button to save the changes.

Your design should look like the following:

| Skill Code | Grade | Comment | Date Skl Acquired |
|------------|-------|---------------------|-------------------|
| ABCDEFGHIJ | Pass | aAbBcCdDeEffGhHiIjJ | 12/34/56 |
| ABCDEFGHIJ | Pass | aAbBcCdDeEffGhHiIjJ | 12/34/56 |
| ABCDEFGHIJ | Pass | aAbBcCdDeEffGhHiIjJ | 12/34/56 |

3. Save the WAM.
4. Re-test your WAM. The Employee Skills grid should look like the following, displaying the correct caption for the GRADE fields:

| Skill Code | Grade | Comment | Date Skl Acquired |
|----------------------|-------------|-----------------|-------------------|
| Administratn Part 1 | Distinction | Met requirement | 25/03/ |
| Administratn Part 2 | Fail | | 02/05/ |
| Communications Degre | Distinction | | 04/05/ |
| Computer Science Deg | Pass | | 05/05/ |
| Economics Degree | Pass | | 01/05/ |
| Company Introduction | Pass | Met requirement | 05/02/ |
| Keyboard Skills | Pass | Met requirement | 05/02/ |
| Management Course 1 | Distinction | Met requirement | 15/03/ |
| Marketing Course 1 | Distinction | Met requirement | 25/03/ |
| Take Shorthand | Pass | | 03/05/ |

5. Change some grades and Save the changes. Redisplay skills for this employee number, to show that grade values were passed into the **Details** WebRoutine.

Summary

Important Information

- This WAM combines a number of weblets to build a Windows like interface. This may not be appropriate in all cases, for example this design would not suit a public website or a web site designed for an occasional user.
- The exercise demonstrates how a fairly complex WAM can be built and tested in stages.
- The Dynamic Select Box uses AJAX technology to call a response WebRoutine that refreshes its list of values.
- A Grid column may be customized using a field visualization weblet.
- The Navigation Panel enables one area of the web page to be refreshed (the Navigation panel is an iFrame in HTML terminology).

Tip & Techniques

- When a new field has been mapped for a WebRoutine, this can be reflected in the web page (actually the XSLT transformation that produces the XHTML) using the WebRoutine Output tab, and dropping the new field onto the page.
- The WebRoutine Output tab reflects the XML, which is always updated by a compile.
- jQuery weblets require lists to be mapped as *JSON data.

What I Should Know

- How to combine a number of weblets to construct a Windows-like interface.

WAM065 - Controlling List Output

Objectives

When you output a list to your web page, you should always consider whether there is a need to limit the number of entries that will be displayed.

If you output a large list:

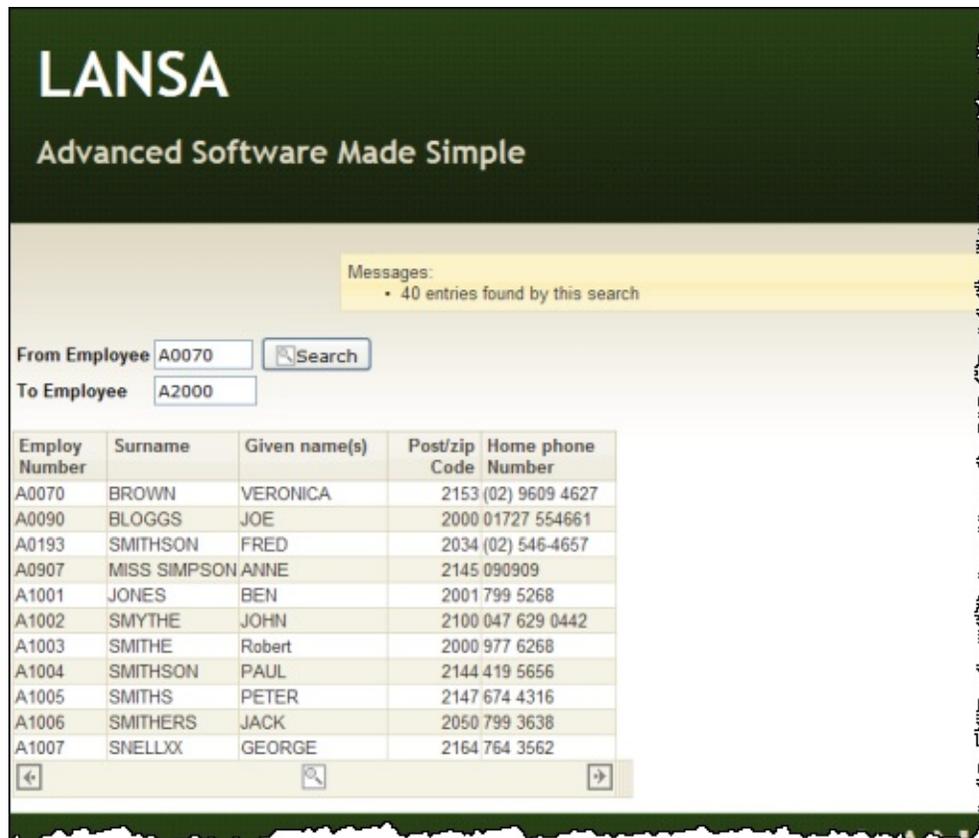
- The application may perform badly.
- The user will need to scroll down the page to find the entries he is interested in, or will need to scroll through the list or grid to find the entries he wants.

If you are an IBM i developer you will be familiar with writing output to sub-files, which the 5250 terminal is then able to scroll through. In the web, if you add 500 entries to the list, they will all be loaded to the page immediately.

This exercise demonstrates how the list paging weblet can be used together with program logic to load a list with one page of entries at a time. This provides one simple technique that could be used.

From your own experience of using web sites you will be familiar with a number of alternative techniques. For example a set of page links, which enable the user to jump to another page of results. These other techniques could also be implemented in a WAM.

When designing applications that may bring back a large set of results, you should always consider as many ways as possible for the user to limit his query to find only the entries he is looking for.



The employee enquiry application:

- Enables employee numbers to be selected using the AutoComplete weblet
- The search displays employees with a fixed page size of 10 entries
- The list_paging_weblet enables the user to page forward and backward to display all the employees found by the search

To achieve these objectives you will complete the following:

- [Step 1. Create WAM iiiEmpSearch – Employee Search](#)
- [Step 2. Add List Paging Images weblet](#)
- [Step 3. Add AutoComplete Weblets \(optional\)](#)
- [Summary](#)

Before You Begin

You should have completed all the preceding exercises in this workshop.

Step 1. Create WAM iiiEmpSearch – Employee Search

1. Create a new WAM

Name: iiiEmpSearch

Description: Employee Search

Weblet Template: iiilay01

2. Define the following work fields:

* Fields

Define Field(#empnof) Reffld(#empno) Desc('First entry on current page')

Define Field(#empnow) Reffld(#empno) Desc('Last entry on current page')

Define Field(#cur_page) Reffld(#std_num) Desc('Current page number')

Define Field(#empfrom) Reffld(#empno) Desc('From Employee')

Define Field(#empto) Reffld(#empno) Desc('To Employee')

Define Field(#total) Reffld(#std_count) Desc('Total entries this search')

3. Define the following working lists:

* lists

Def_List Name(#emplist) Fields((#empno *out) (#surname *out) (#givenname *out) (#initials *out))
Entrys(*max)

Def_List Name(#emps) Fields(#empno) Counter(#total) Type(*Working)
Entrys(*max)

4. Define the following global map.

* Global Maps

Web_Map For(*both) Fields((#stdrentry *hidden))

Many weblets return a value in the field STDRENTRY, so it usually needs to be mapped as a hidden field. You will add other fields to this web_map as you develop this WAM.

5. Create a WebRoutine search using the following code.

```
WebRoutine Name(search)
```

```
Web_Map For(*both) Fields(#empfrom #empto #emplist)
```

```
Case (#stdrentry)
```

```
When (= s)
```

```
#com_owner.browse I_Empfrom(#empfrom) I_Empto(#empto)
Endcase
Endroutine
```

EMPFROM and EMPTO will provide search values for employee number. The current page of results will be displayed by the list EMPLIST. A push button will return a STDREENTRY value of S, which will perform the method routine browse.

Ignore the error *Feature name browse is not a member.....* This routine is defined in the next step.

Important Note: This WAM, when completed, will comprise around 130 statements. To save time, much of this code is provided. Make sure you review each section of code as it is added, either at the time you are doing it or later. As with all programming tasks, you should approach WAM development a stage at a time, and test your initial code, before moving on to add additional logic.

6. Create the browse method routine using the following code:

```
Mthroutine Name(browse)
Define_Map For(*input) Class(#empno) Name(#i_empfrom)
Define_Map For(*input) Class(#empno) Name(#i_empto)
Clr_List Named(#emplist)
Select Fields(#emplist) From_File(pslmst) Where((#std_count <= 10) And (#e
Add_Entry To_List(#emplist)
If (#std_count = 1)
#empnof := #EMPNO
Endif
#empnow := #EMPNO
Endselect
#cur_page += 1
Endroutine
```

You will add more logic to this routine in a later step.

Review the browse routine, which:

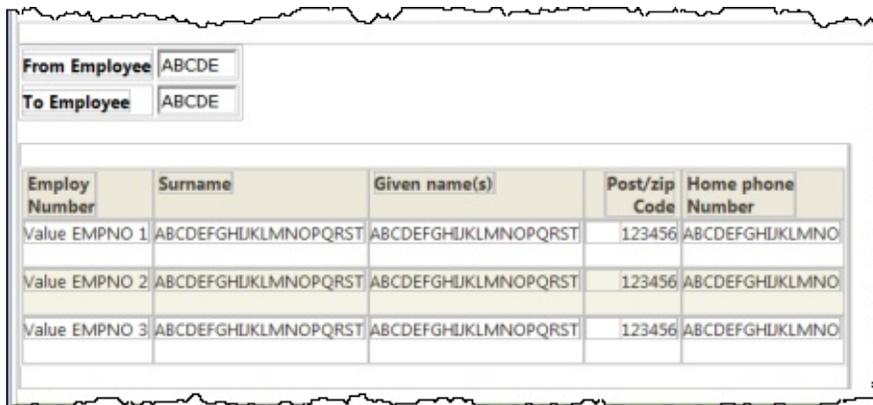
- Accepts two input values for EMPNO.
- Clears the working list EMPLIST. Note that STD_COUNT is the Counter() value for this list.

- Selects records from file PSLMST while STD_COUNT is less than or equal to 10 and the employee number is less than I_EMPTO.
- Reads the file using a startkey and an endwhere. The read will end when the Where() condition is false.
- The SELECT reads the next 10 records or stops when the end value for the search is reached.
- Adds entries to list EMPLIST
- Maintains EMPNOF as the first entry in the current list and EMPNOW as the last entry in the current list
- Increments current page (CUR_PAGE)

7. Add the following as hidden fields in the global web_map:
EMPNOF, EMPNOW and CUR_PAGE.

Note that this web_map maps the fields for *both. Their current value will be mapped back into each WebRoutine that is invoked from the web page.

8. Compile your WAM and open the search WebRoutine in the Design view. It should look like the following:



9. Select the employee number input box in the table containing employee numbers, and use the context menu *Table Items / Add Columns...* to add one column to the table.
10. Drop a *Push button with image* into the new top cell. Remove the placeholder characters.
11. Select the push button and set up the push button as follows, using the Details tab:

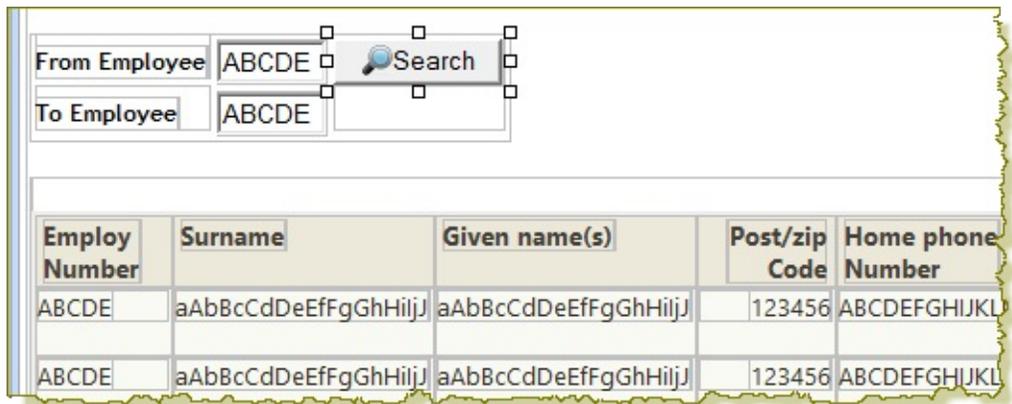


| Property | Value |
|--------------------------|------------------------------------|
| caption | Search |
| left_relative_image_path | icons/normal/16/zoom_16.png |
| on_click_wname | search |
| submitExtraFields | Field Name: STDREENTRY |
| | Literal Value: S |

12. Set the tab_index for the push button and employee number fields as follows

| Element | Tab_index |
|-------------|-----------|
| EMPFROM | 1 |
| EMPTO | 2 |
| Push Button | 3 |

13. Save your changes. Your web page should look like the following:



14. Execute your WAM in the browser. Enter search values for employee number such as A0070 and A2000. Your results should look like the following:

From Employee

 Search

To Employee

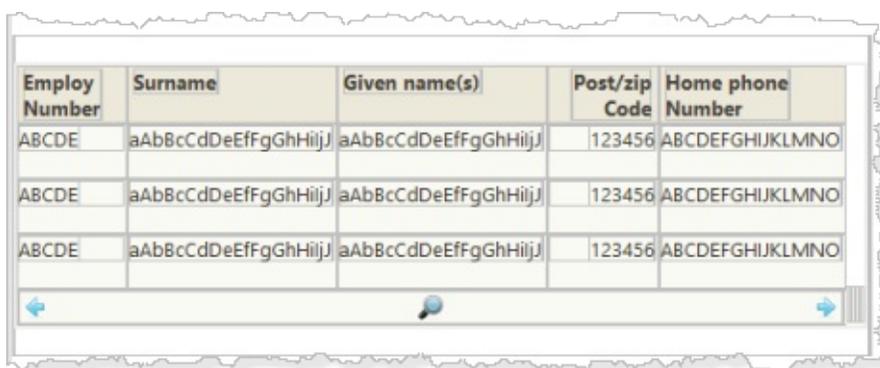
| Employ Number | Surname | Given name(s) | Post/zip Code | Home phone Number |
|---------------|--------------|----------------|---------------|-------------------|
| A0070 | BROWN | VERONICA | 2153 | (02) 9609 4627 |
| A0090 | BLOGGS | FRED JOHN ALAN | 2220 | 344-2234454545 |
| A0193 | SMITHSON | FRED | 2034 | (02) 546-4657 |
| A0907 | MISS SIMPSON | ANNE | 2145 | 090909 |
| A1001 | JONESxx | BEN | 2001 | 799 5268 |
| A1002 | SMYTHE | JOHN | 2100 | 047 629 0442 |
| A1003 | SMITHE | Robert | 2000 | 977 6268 |
| A1004 | SMITHSON | PAUL | 2144 | 419 5656 |
| A1005 | SMITHS | PETER | 2147 | 674 4316 |
| A1006 | SMITHERS | JACK | 2050 | 799 3638 |
| A1007 | SNELL | GEORGE | 2164 | 764 3562 |

Step 2. Add List Paging Images weblet

1. Click anywhere in the employee list and use the context menu, *Table Items / Add Rows...* to add one row to the bottom of this table.

Hint: Click in one of the columns such as Given Name and then use the context menu.

2. Select inside the left hand cell of this new row and use the *Details* tab to set its colspan to 5.
3. Drag a *List paging images* weblet into the new row. Your web page should look like the following:



The screenshot shows a table with five columns: Employ Number, Surname, Given name(s), Post/zip Code, and Home phone Number. The table contains three rows of data. A new row has been added at the bottom, which contains a list paging images weblet. The weblet is represented by a horizontal bar with a left arrow, a search icon, and a right arrow.

| Employ Number | Surname | Given name(s) | Post/zip Code | Home phone Number |
|-------------------|----------------------|----------------------|---------------|-------------------|
| ABCDE | aAbBcCdDeEfFgGhHiIjJ | aAbBcCdDeEfFgGhHiIjJ | 123456 | ABCDEFGHIJKLMNO |
| ABCDE | aAbBcCdDeEfFgGhHiIjJ | aAbBcCdDeEfFgGhHiIjJ | 123456 | ABCDEFGHIJKLMNO |
| ABCDE | aAbBcCdDeEfFgGhHiIjJ | aAbBcCdDeEfFgGhHiIjJ | 123456 | ABCDEFGHIJKLMNO |
| ← [Search Icon] → | | | | |

4. Save your changes.
5. Select the list paging images weblet and use the *Details* tab to set up its properties:

| Property | Value |
|-----------------|-------------------|
| prevcondfield | STDPREV |
| nextcondfield | STDMORE |
| Rentryfield | STDREENTRY |
| on_page_wname | page |
| on_search_wname | search |

You will observe that STDPREV, STDMORE and STDREENTRY are already defined. All these fields are defined in the repository. To use this weblet you

must ensure they are mapped appropriately.

The WebRoutine **page** is not yet defined, so you must type in this value.

6. Save your changes.

You'll find a detailed definition of all weblets in the *Web Application Modules* guide.

STDPREV and STDMORE are used to show or hide the next and previous images. STDPREV = Y will hide the previous image.

The weblet returns a value in STDREENTRY when the *more* or *previous* image is selected.

- The more image returns **M** in STDREENTRY
- The previous image returns **P** in STDREENTRY
- The search image returns **blank** in STDREENTRY.

7. Add STDPREV and STDMORE as hidden fields in the global web_map, which should now look like the following:

* Global Maps

```
Web_Map For(*both) Fields((#stdreentry *hidden) (#empnof *hidden)
(#empnow *hidden) (#cur_page *hidden) (#stdmore *hidden) (#stdprev
*hidden))
```

8. Create a page WebRoutine based on the following:

```
WebRoutine Name(page)
Web_Map For(*both) Fields(#empfrom #empto)
Web_Map For(*output) Fields(#emplist)
Case (#stdreentry)
When (= M)
#com_owner.browse I_Empfrom(#empnow) I_Empto(#empto)
When (= P)
#com_owner.previous I_Empfrom(#empnof) I_Empto(#empfrom)
Endcase
#stdreentry := D
Transfer Toroutine(search)
Endroutine
```

Ignore errors shown because the **previous** method routine does not yet exist.

Review the **page** WebRoutine which:

- Maps fields EMPFROM and EMPTO, in and out.
- Maps the list EMPLIST out.
- EMPNOF and EMPNOW represent the first and last entry in the list EMPLIST.
- For **more**, the **browse** method routine is invoked with EMPNOW as the from employee number value.
- For **previous**, the **previous** method routine is invoked with EMPNOF as the from employee number value.

9. Create the **previous** method routine based on the following:

```

Mthroutine Name(previous)
Define_Map For(*input) Class(#empno) Name(#i_empfrom)
Define_Map For(*input) Class(#empno) Name(#i_empto)
Clr_List Named(#emplist)
Select Fields(*all) From_File(pslmst) Where((#std_count <= 10) And (#empnc
Add_Entry To_List(#emplist) After(*start)
If (#std_count = 1)
#empnow := #EMPNO
Else
#empnof := #EMPNO
Endif
Endselect
#cur_page -= 1
If_Status Is(*beginfile)
Message Msgtxt('No more records')
#stdprev := *blank
Else
#stdprev #stdmore := Y
Endif
Endroutine

```

Review the **previous** method routine, which:

- Maps in from employee number and to employee number values.
- Clears the list EMPLIST
- Selects record from file PSLMST with a startkey, reading backwards, with an endwhere condition.

- The read ends when 10 record have been added to the list, or the current employee number is equal to or greater than the I_EMPTO value. This field contains the 'From Employee' value.
- EMPNOF and EMPNOW are maintained, allowing for the read being backwards.
- Current page number (CUR_PAGE) is decremented.
- Beginning of file is detected with a message.
- STDPREV and STDMORE are maintained, and control showing and hiding the *more* and *previous* images.

10. Extend the **browse** method routine to:

- Manage STDPREV and STDMORE
- Detect end of file
- Detect the employee number to value has been reached.

Your code should look like the following. **Existing** code is shown in red.

```

.....
Endselect
#cur_page += 1
If (#cur_page >= 2)
#stdprev := Y
Else
#stdprev := *blanks
Endif
If_Status Is(*endfile)
Message Msgtxt('No more records')
#stdmore := *blank
Endif
If (#empno >= #empto)
Add_Entry To_List(#emplist)
Message Msgtxt('End of Search')
#stdmore := *blank
Endif
Endroutine

```

11. In the **Search** WebRoutine add code to initialize the STDPREV and STDMORE fields.

Your code should look like the following. New code is shown in red.

```
When (= s)* Search Button#stdprev := *blank#stdmore :=  
Y#com_owner.browse I_Empfrom(#empfrom) I_Empto(#empto)
```

12. Compile and test your WAM in the browser.
 - a. With search values such as from A0070 to A2000 you should be able to page forward until A2000 is reached and then page back until A0070 is reached.
 - b. Perform a search from A0070 to A0090. The more and previous images should not be displayed.
13. Note that when the Search web page is initially displayed, the list images weblet is shown. To hide this when the page is initially displayed, make the following changes:
 - a. Add the field STD_COUNT to the global WEB_MAP as a hidden field
 - b. Define the hide_if property for the std_list_images as:

```
#STD_COUNT = 0
```

Add this logic in the Xpath expression editor for the hide_if property. Note that the field name must be in upper case.
14. Save your changes and re-test your WAM. The list images weblet should be hidden when the Search page is first displayed.

Step 3. Add AutoComplete Weblets (optional)

This step adds *Autocomplete* weblets for the To and From employee number fields. You may choose not to complete this step to save time.

At present, to use this enquiry you need to know suitable employee numbers. Adding an *AutoComplete* weblet for the from and to employee fields and a supporting response WebRoutine, will bring back and display a list of matching employee numbers as you type into the weblet.

1. Create the Empno_Prompt response WebRoutine based on the following code:

```
WebRoutine Name(Empno_Prompt) Response(*JSON)
Web_Map For(*input) Fields(#empno)
Web_Map For(*output) Fields((#emp_dd *json))
Def_List Name(#emp_dd) Fields(#empno #std_code) Counter(#std_count) Type
Clr_List Named(#emp_dd)
Select Fields(#emp_dd) From_File(pslmst) Where(#std_count <= 3) With_Key
#std_code := #empno
Add_Entry To_List(#emp_dd)
Endselect
Endroutine
```

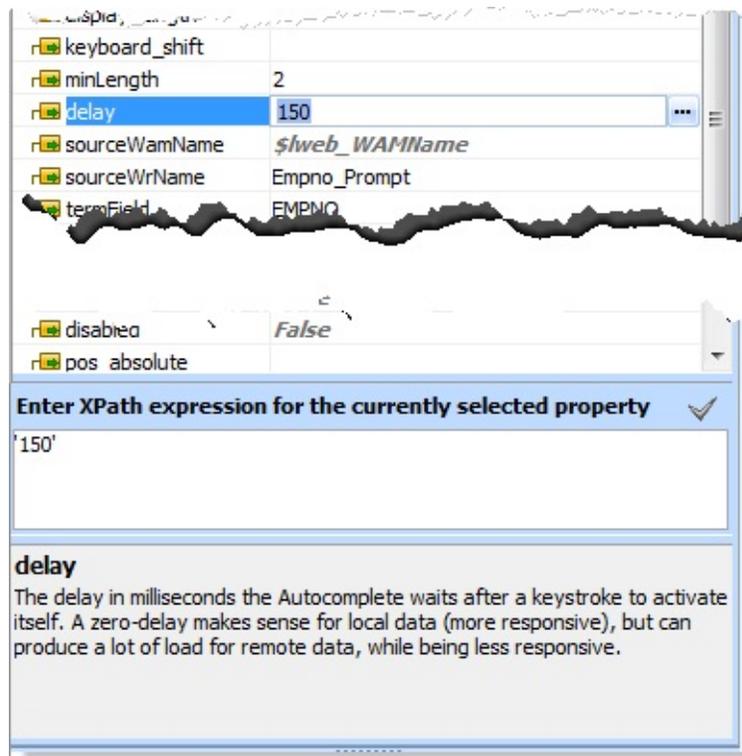
The Empno_Response WebRoutine:

- Must have the Response(*JSON) keyword on the WebRoutine statement. This routine is called by the AutoComplete weblet and returns a small list of employee numbers as JSON data.
 - Maps the field EMPNO for input
 - Maps the list EMP_DD for output as JSON data.
 - Defines a working list EMP_DD containing EMPNO. A second field has been added to ensure the fields are recognized by the *Details* tab dialog.
 - Clears the list EMP_DD
 - Selects up to 3 entries from the file PSLMST using EMPNO as a startkey
 - Adds entries to the list EMP_DD
 - Returns the list EMP_DD to the AutoComplete weblet.
2. Compile your WAM.
 3. Open the **search** WebRoutine in the *Design* view.

- Drop an AutoComplete weblet onto to the EMPFROM field and the EMPTO field.
- Set up both AutoComplete weblets as follows:

| Property | Value |
|--------------|--------------|
| minLength | 2 |
| Delay | 150 |
| sourceWrName | Empno_Prompt |
| termField | EMPNO |
| listName | EMP_DD |
| valueField | EMPNO |

For more information on all weblets see the *Web Application Modules Guide*. See also the help available for each property on the Details tab:



The **minLength** value is the characters to be typed before the WebRoutine is

called

The **delay** value is the number of milliseconds the weblet waits to activate itself after the last keystroke.

The **termField** is the value passed to the WebRoutine defined in **sourceWrName**.

The **listName** is the response list to be displayed as a dropdown list.

The **valueField** is the list value to be displayed.

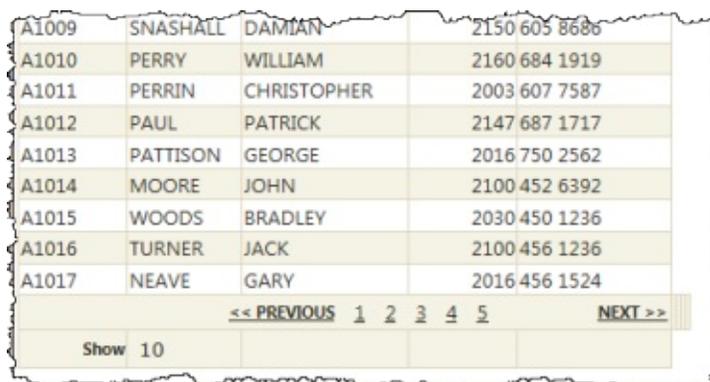
6. If necessary reduce the width of the AutoComplete input boxes to suit the field EMPNO (up to 5 characters).
7. Save your changes.
8. Execute your WAM in the browser and test the AutoComplete weblet. You should get the following results:



Summary

Important Information

- Other techniques you could use to control a large list of results, include showing a list of the page numbers available, as below, and enabling the user to jump to the page required.
- This example also enables the user to change the number of entries per page. One simple way to do this is to add an anchor to the page size value and use this to switch between two values such as 10 and 25.

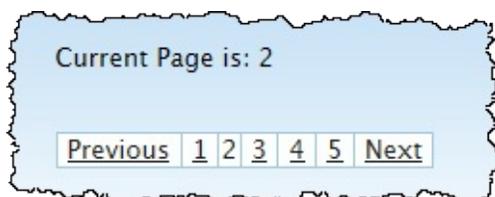


| | | | |
|-------|----------|-------------|---------------|
| A1009 | SNASHALL | DAMIAN | 2150 605 8686 |
| A1010 | PERRY | WILLIAM | 2160 684 1919 |
| A1011 | PERRIN | CHRISTOPHER | 2003 607 7587 |
| A1012 | PAUL | PATRICK | 2147 687 1717 |
| A1013 | PATTISON | GEORGE | 2016 750 2562 |
| A1014 | MOORE | JOHN | 2100 452 6392 |
| A1015 | WOODS | BRADLEY | 2030 450 1236 |
| A1016 | TURNER | JACK | 2100 456 1236 |
| A1017 | NEAVE | GARY | 2016 456 1524 |

<< PREVIOUS 1 2 3 4 5 NEXT >>

Show 10

- Another more sophisticated technique uses a working list, displayed as a single row, as in this example. This makes it easy to handle any number of pages and disable the link for the current page.



Tips & Techniques

- The technique used here will work well on large files.
- Of course your search criteria would likely be much more sophisticated than used here.
- The `SELECT_SQL` statement will enable you to read one or more files rapidly and also implement much more flexible search criteria. See also the free format version of `SELECT_SQL`.
- The *Autocomplete* weblet includes a cache property. If `cache = true`, the selection will be refined locally as you continue to type, rather than going

back to the server. For example the following initial response:

The screenshot shows a web form with two input fields: "From Employee" containing "A0" and "To Employee" containing "A0070". A green "Search" button is to the right. Below the "To Employee" field, a dropdown menu is open, listing employee numbers from A0090 to A1007. The table below the dropdown has columns for "Employ Number", "Surname", "Given name(s)", and "Pos".

| Employ Number | Surname | Given name(s) | Pos |
|---------------|---------|---------------|-----|
| A0090 | | | |
| A0193 | | | |
| A0907 | | | |
| A1001 | | | |
| A1002 | | | |
| A1003 | | | |
| A1004 | | | |
| A1005 | | | |
| A1006 | | | |
| A1007 | | | |

will be refined to the following as you continue to type into the input box, without making a second call to the server:

The screenshot shows the same web form, but now the "From Employee" field contains "A01" and the "To Employee" field contains "A0193". The dropdown menu is still open, but it is now filtered to show only the employee number "A0193". The table below the dropdown is partially visible, showing the same column headers as in the previous screenshot.

| Employ | Surname | Given name(s) | Pos |
|--------|---------|---------------|-----|
| A0193 | | | |

What You Should Know

- How to implement the list paging images weblet. You could also have used the list paging buttons weblet
- How to implement the AutoComplete weblet.

WAM070 - Hiding Techniques

Objectives

- To demonstrate different hiding techniques.

In this exercise, you will learn about a number of ways to conditionally hide objects on a page. The conditions will be based on a field value that you will set in the RDML. Some of the hiding techniques will only be available to certain types of objects, while one of the techniques can be applied to anything on the page. This requires the XSL Source to be manually edited.



To achieve these objectives, you will complete the following:

- [Step 1. Create a new WAM](#)
- [Step 2. Edit the HideMain WebRoutine web page](#)
- [Step 3. Apply the Conditional Hides](#)
- [Step 4. Test the WAM](#)
- [Summary](#)

Before You Begin

- In order to complete this exercise, you must first complete all the previous Exercises.

Step 1. Create a new WAM

1. Create a new **WAM**:

Name: **iiiHideTech**

Description: **Hiding Techniques**

Layout Weblet: **iiilay01**

2. In this step, you will add the RDMLX code to demonstrate different hiding techniques.

Use the DEFINE command to define the following work fields.

| Field | Type | Length | Default Value |
|-----------|-------|--------|---------------|
| CLASSHIDE | *char | 10 | *blanks |
| HIDEIF | *char | 10 | *blanks |
| XSLIF | *char | 10 | *blanks |

3. Create a WebRoutine named **HideMain** with a description of '**Hiding Techniques**'.

```
WebRoutine Name(HideMain) Desc('Hiding Techniques')
```

```
Endroutine
```

4. Add a WEB_MAP for *both. Include the fields you have just created as well as STDRETRY. Define all as hidden fields as follows:

```
Web_Map For(*BOTH) Fields((#CLASSHIDE *HIDDEN) (#HIDEIF *HIDD
```

5. Add a CASE loop based on STDRETRY for three different values A, B and C. Your code should look like the following:

```
Case Of_Field(#STDRETRY)
```

```
When Value_Is(= A)
```

```
If Cond(#CLASSHIDE = *BLANKS)
```

```
#CLASSHIDE := 'hidden'
```

```
Else
```

```
#CLASSHIDE := *BLANKS
```

```
Endif
When Value_Is(= B)
If Cond(#HIDEIF = *BLANKS)
#HIDEIF := 'HIDE'
Else
#HIDEIF := *BLANKS
Endif
When Value_Is(= C)
If Cond(#XSLIF = *BLANKS)
#XSLIF := 'HIDE'
Else
#XSLIF := *BLANKS
Endif
Endcase
```

The complete WAM source code can be found in [WAM 070. Appendix](#)

6. Compile your WAM

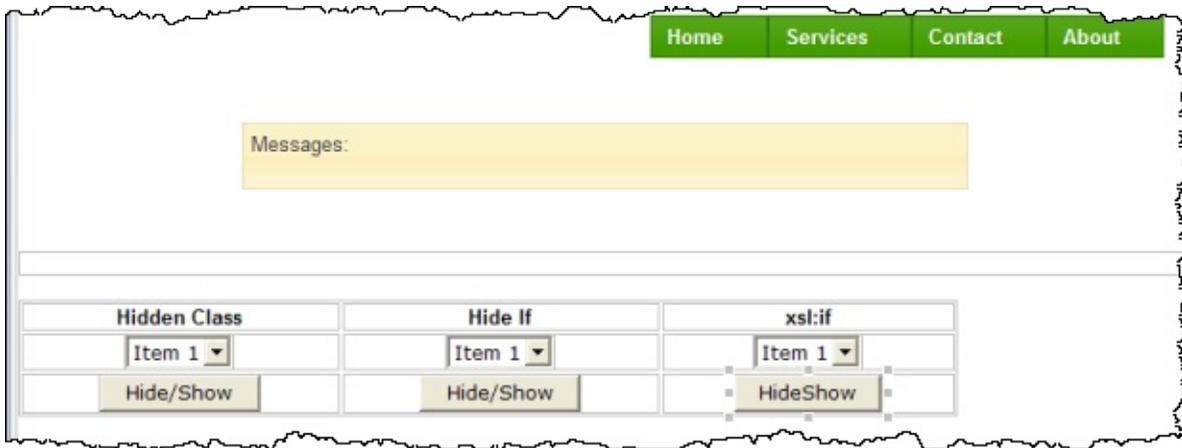
Step 2. Edit the HideMain WebRoutine web page

In this step, you will set up the web page for the HideMain WebRoutine in the Design view.

1. Open the **HideMain** WebRoutine in the *Design* view.
2. Use the context menu, *Insert HTML / Table* to add a 3 row by 3 column table to the page.
3. Select the table and set the table's width property to **80%**.
4. Set the align property of each individual cell to **center**.
5. Set the *class* property of each cell in the **top row** to **bold**.
6. Add text, to clarify the technique used, to each of the three cells in the top row. The headings should be **Hidden Class**, **hide_if**, and **xsl:if**.
7. Add a *Combo box* weblet to each of the three cells in the middle row.
8. Add a *Push button* weblet to each of the three cells in the bottom row.
9. Configure the three Hide/Show Push buttons using this table.

| | Property | Value |
|-----------------------|-------------------|--|
| | caption | Hide/Show |
| | on_click_wrname | HideMain |
| Table Column 1 | submitExtraFields | Field Name: STDREENTRY
Literal Value: A |
| Table Column 2 | submitExtraFields | Field Name: STDREENTRY
Literal Value: B |
| Table Column 3 | submitExtraFields | Field Name: STDREENTRY
Literal Value: C |

The *Design* view should appear something like the following:



10. Save your changes.

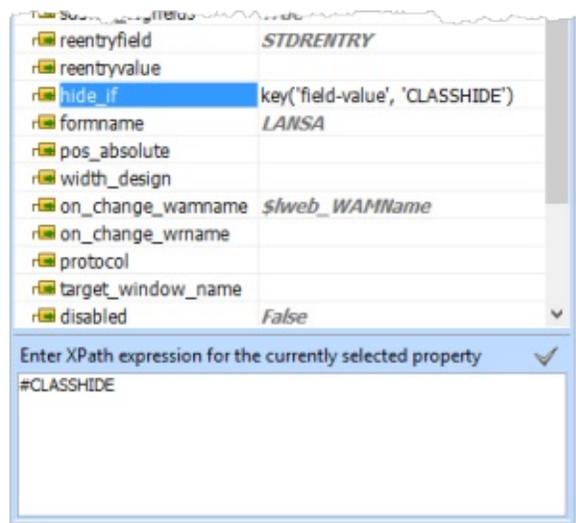
Step 3. Apply the Conditional Hides

In this step, you will conditional hide the dropdowns using different techniques.

1. Set the hidden class for the first dropdown list.

Set the first dropdown's *class* property to **#CLASSHIDE**. The RDML conditions this field to contain either *blanks or 'hidden'.

Note: Select the *class* property value and then use the *XPath editor* to enter **#CLASSHIDE**.

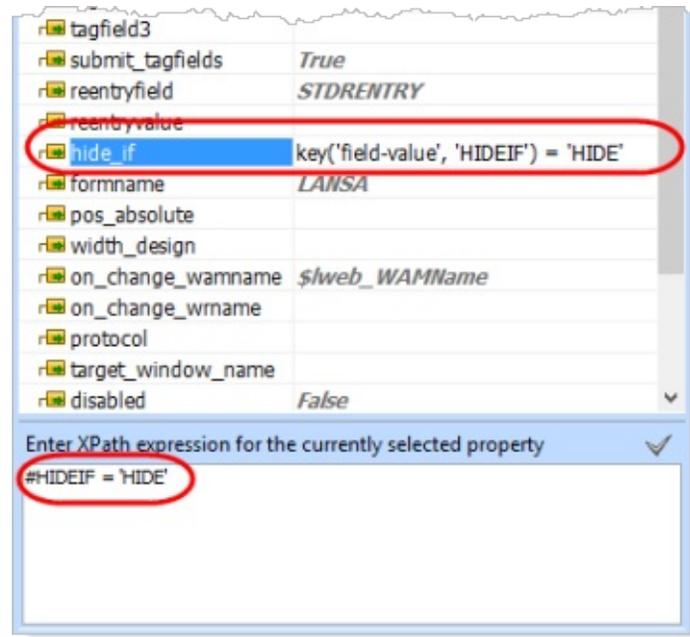


Click the icon to confirm this change. Note that the XPath editor has added the required XSL expression:

key('field-value', 'CLASSHIDE')

2. Set the hide_if condition for the second dropdown list.

Select the *hide_if* property value and use the *XPath expression* window to define the condition **#HIDEIF ='HIDE'**. Press the button to confirm this change.



Note: The field name must be entered in upper case.

Once again, the field is conditioned in the RDML, and when the condition specified in the *hide_if* property is true, the object will be hidden.

3. Set the **xsl:if**.

a. Select the third combo box.

b. Select the *XSL* tab. The block of code that generates the selected item will be highlighted.

You must enclose this **highlighted code** within the **xsl:if** tags. The syntax for the **xsl:if** is:

```
<xsl:if test="test">
  code to be hidden
</xsl:if>
```

Where *test* is the condition on which the code will be hidden.

c. Enclose the code for the third dropdown with the **xsl:if** condition. The condition to hide is when **#XSLIF** is 'HIDE', so *test* will be `key('field-value', 'XSLIF') != 'HIDE'`.

That is, output the combo box to the page if XSLIF is not equal to 'HIDE'.

The XSL code should look like the following. New code is shown in red.

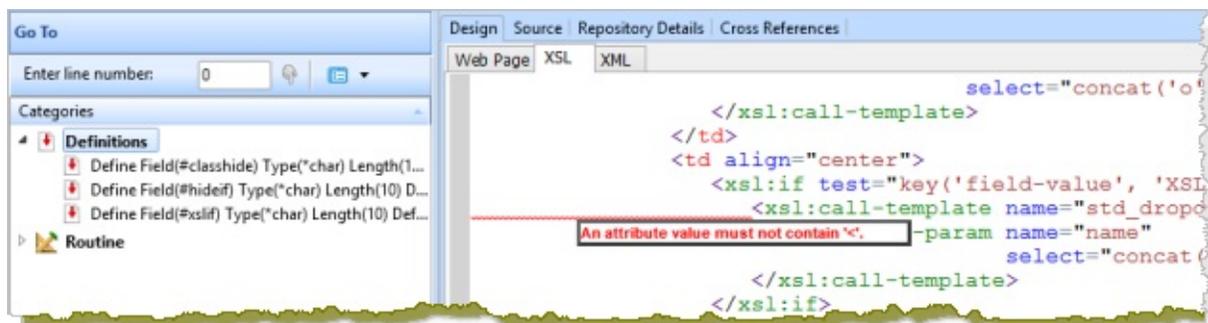
```
<xsl:if test="key('field-value', 'XSLIF') != 'HIDE'">
```

```
<xsl:call-template name="std_dropdown">
  <xsl:with-
param name="name" select="concat('o', position(), '_LANSA_28762')"/>
</xsl:call-template>
</xsl:if>
```

Using the XSL Editor

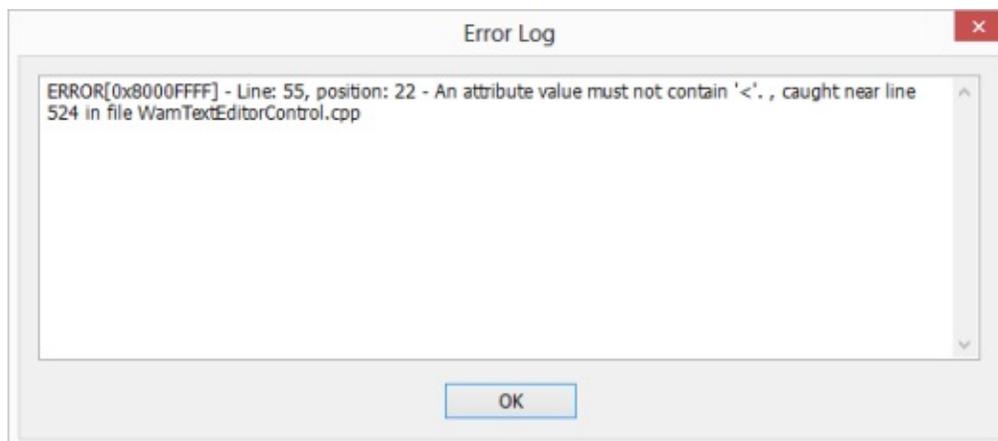
The editor has autocomplete functionality. As you type, press enter to select the prompted code. Once you complete the `<xsl:if` with the `>` character, the end if (`</xsl:if>`) will be generated. Move this after the `</xsl:call-template>`.

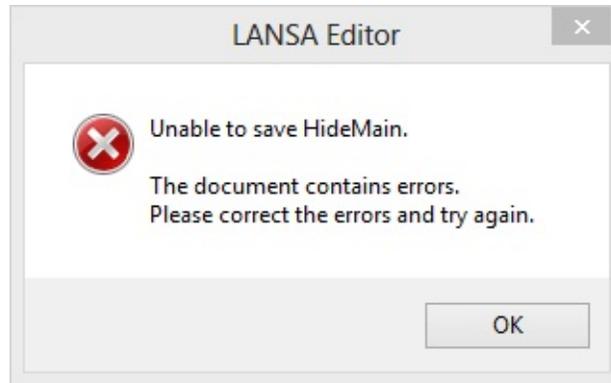
Note: If you click on the *Design* tab or try to save your XSL while your xsl changes contain errors, the errors will be reported on the *Go To* tab as shown in this example. In this case, there is a missing double quote at the end of the test condition.



Note that the line in error is highlighted.

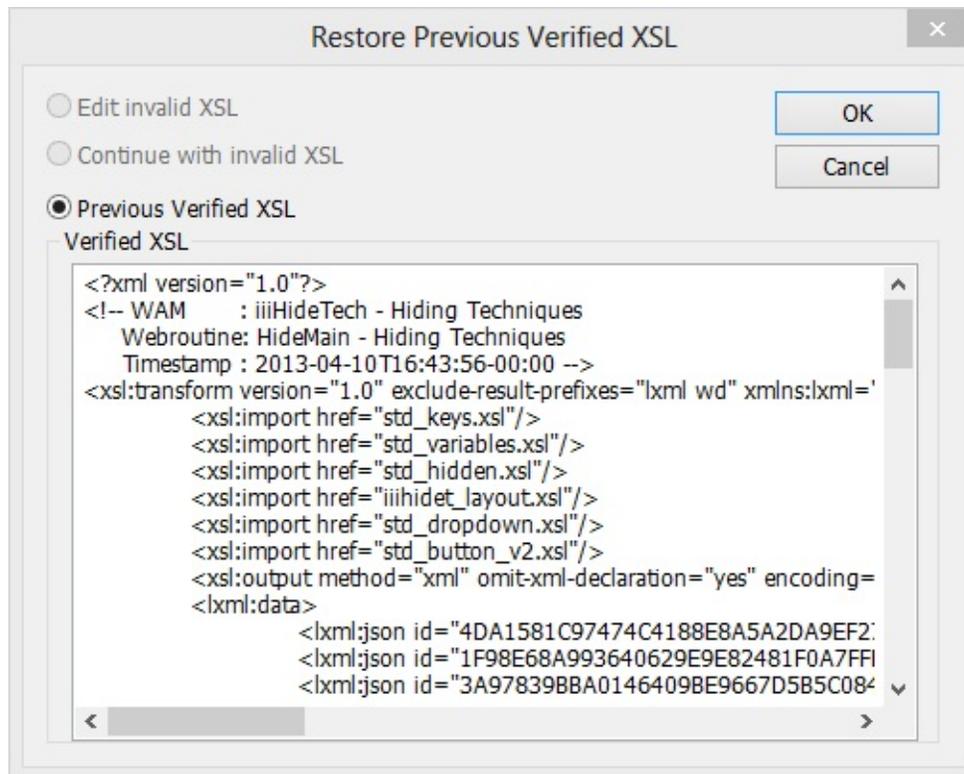
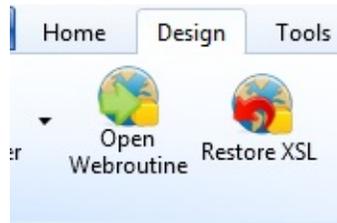
Attempts to save xsl with errors may also show these dialogs.





4. Press OK, to continue editing your XSL. In this case, as noted above a double quote at the end of the test condition is missing.

On the *Design* ribbon, you can use the Restore XSL dialogue to restore the previous verified XSL or continue editing.



HideMain web page.

Step 4. Test the WAM

In this step, you will test WAM.

1. Run WebRoutine **HideMain** in the browser.
2. Test all three buttons. Pressing each button will hide the object if it is visible, or make the object visible if it is hidden. The buttons will appear to function in the same way, but you know what is going on behind the scenes, and can see the different techniques in action.
3. If your hide / show logic is not working you will need to check both your RDML and on the *Design* tab, the definition of the push buttons and combo boxes. If necessary use debug to resolve any errors.
4. Close the browser.

Summary

Important Observations

- You can apply the hidden class to any object with a class property.
- You can set the hide_if property to hide weblinks.
- When an object does not have a class or hide_if property, you can still hide it from within the XSL code using an xsl:if.

Tips & Techniques

- Once the objects on the page are set up correctly (conditionally hidden based on a field) you will use the RDML to control the field to hide the object.

What I Should Know

- How to hide object using a hidden class, the hide_if property, and an xsl:if.

WAM 070. Appendix

Use the following RDMLX source code to create iiiHideTech in Step 1 of this exercise.

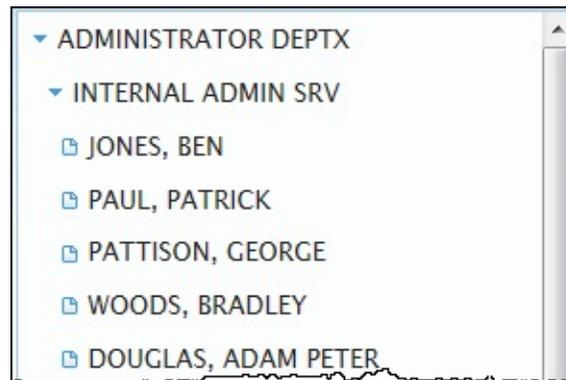
Replace the Layoutweblet() keyword with your common layout name.

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_WAM) Layoutweblet('iiilay01')
WebRoutine Name(HideMain)
Web_Map For(*BOTH) Fields((#CLASSHIDE *HIDDEN) (#HIDEIF *HIDD
Define Field(#CLASSHIDE) Type(*char) Length(10)
Define Field(#HIDEIF) Type(*char) Length(10)
Define Field(#XSLIF) Type(*char) Length(10)
Case Of_Field(#STDREENTRY)
When Value_Is(= A)
If Cond(#CLASSHIDE = *BLANKS)
#CLASSHIDE := 'hidden'
Else
#CLASSHIDE := *BLANKS
Endif
When Value_Is(= B)
If Cond(#HIDEIF = *BLANKS)
#HIDEIF := 'HIDE'
Else
#HIDEIF := *BLANKS
Endif
When Value_Is(= C)
If Cond(#XSLIF = *BLANKS)
#XSLIF := 'HIDE'
Else
#XSLIF := *BLANKS
Endif
Endcase
Endroutine
End_Com
```

WAM075 - Using a Tree View Weblet

Objectives

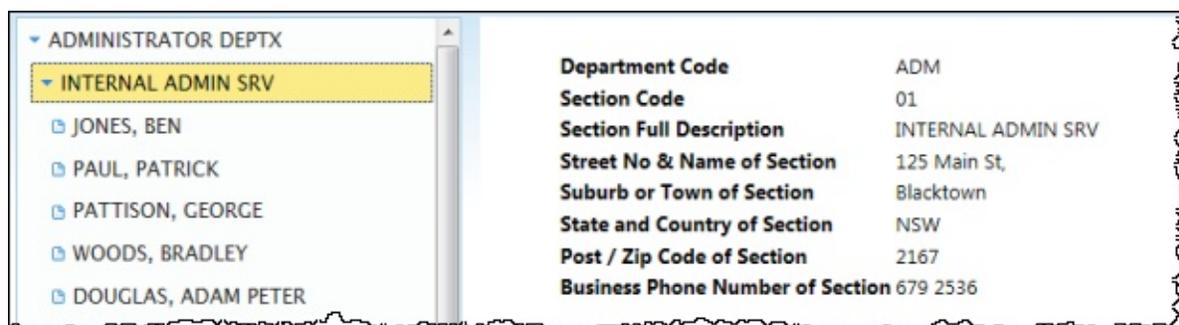
The Tree View weblet provides an expandable collapsible tree. It can be useful as a site navigation system or for visualizing complex hierarchical data.



The Tree View is filled with data from a working list. Similar to the VL Tree View, the source data may be unlevelled, where each entry has an id and specifies the id of its parent, or levelled, where the tree builds itself from a sorted list based on key columns in the list.

When using unlevelled source data, the list can be configured to use AJAX to request child entries from the server when a branch is opened by the user.

This exercise demonstrates how to build an expandable tree view. It will also show how to display detail data when an entry at each level is selected.



To achieve these objectives you will complete the following:

- [Step 1. Create WAM iiiTreeView – Using a Tree View Weblet](#)
- [Step 2. Make the Tree View Expand](#)
- [Step 3. Display Details for a Selected Department](#)
- [Step 4. Display Details for Sections and Employees](#)

- [Summary](#)

Before You Begin

You should complete all preceding exercises before starting this exercise.

Step 1. Create WAM **iiiTreeView** – Using a Tree View Weblet

Unlevelled List

In an unlevelled list each entry tells the Tree View exactly where it fits in the tree structure by specifying its parent ID. At a minimum, an unlevelled list must contain columns with the following data:

| Field | Tree View Property | Description |
|-----------|-----------------------------|---|
| ID | list_id_field | A unique ID string to identify the entry. |
| Parent ID | list_parent_id_field | The ID string of the parent entry. An empty string indicates a top level entry. |
| Caption | list_caption_field | The text to display for the entry. |

The Tree View processes the list entries in the supplied order and cannot add an entry to a parent that doesn't exist. It is your responsibility to ensure the list is sorted so that parent items come before their children and items at the same level are in display order.

Additional list fields may be used to control the tree view behaviour and appearance:

- List_image_field
- List_open_image_field
- List_is_selected_field
- List_is_expanded_field
- List_onselect_wamname_field
- List_onslect_wrname_field

Refer to [Tree View \(std_treeview_v2\) Properties](#) for further details.

Your WAM will build an initial list based on the department table (DEPTAB), so that initially the tree will contain a single level. You will later add routines to handle expanding the tree view at each level.

1. Create a new WAM:

Name: **iiiTreeView**

Description: **Using a Tree View Weblet**

Layout Weblet: **iiilay01**

2. Define the fields and lists required to support the tree view

```
Define Field(#listid) Reffld(#std_name)
Define Field(#listpid) Reffld(#std_name)
Define Field(#listcapt) Reffld(#std_desc)
Define Field(#onsubid) Reffld(#std_name)
Define Field(#onsublvl) Reffld(#std_code)
Define Field(#haschld) Reffld(#std_flag)
Define Field(#selwam) Reffld(#std_name)
Define Field(#selwrn) Reffld(#std_name)
Define Field(#currid) Reffld(#std_name)
Def_List Name(#emptree) Fields(#listid #listpid #listcapt #haschld #selwam #
Def_List Name(#ancestor) Fields(#listid) Type(*Working) Entries(3)
```

The list named **ancestor** will be explained in a later step.

3. Define a WebRoutine **deptview** which:

- Maps for both, the lists EMPTREE and ANCESTOR as JSON data
- Clears and builds the list EMPTREE reading all records from the department table
- Populates the list fields for each entry

Note:

- The parent id (field LISTPID) is blank for the top level in the tree view.
- The "has children" field (HASCHLD) is Y as this is the top level entry.
- WAM name and WebRoutine fields define the routine to call if this entry is selected

Your code should look like the following. Substitute your initials for **iii**.

```
WebRoutine Name(deptview)
Web_Map For(*both) Fields((#emptree *json))
Web_Map For(*both) Fields((#ancestor *json))
Clr_List Named(#emptree)
Select Fields(#deptment #deptdesc) From_File(deptab)
#listid := #deptment
#listcapt := #deptdesc
```

```

#listpid := *blanks
#haschld := Y
#selwam := iiiTreeView
#selwrn := DEPDET
Add_Entry To_List(#emptree)
Endselect
Endroutine

```

4. Compile the WAM and open the **deptview** WebRoutine in the *Design* view.
5. Drop a tree view weblet onto the page. Select the tree view weblet and drag the center right hand "handle" to make the tree view wider to allow room for three levels and descriptions to be displayed.
6. Set up the tree view properties as follows:

| Property | Value |
|------------------------|--|
| Listname | EMPTREE |
| item_image | icons/normal/16/operator_16.png *** |
| list_caption_field | LISTCAPT |
| list_id_field | LISTID |
| list_onselect_wamname | SELWAM |
| list_onselect_wrname | SELWRN |
| list_haschildren_field | HASCHLD |
| list_parent_id_field | LISTPID |

*** select the image using the Ellipsis button. Drill down by selecting the **normal** and **16 folders**.

7. Save your changes and run the WAM in the browser. Your tree view should display description for all departments:

▶ ADMINISTRATOR DEPTX

- ▶ INTERNAL AUDITING
- ▶ FLEET ADMINISTRATION
- ▶ GROUP ACCOUNTS
- ▶ INFORMATION SERVICES
- ▶ LEGAL DEPARTMENT
- ▶ MANAGEMNT INFORMATIO
- ▶ MARKETING DEPARTMENT
- ▶ SALES & DISTRIBUTION
- ▶ TRAVEL DEPARTMENT

Step 2. Make the Tree View Expand

In this step you will add a new **treeexpand** WebRoutine to handle expanding departments to add the department's sections and expanding sections to add employees for the selected section.

- The tree view weblet is AJAX enabled and will invoke the expand WebRoutine when an entry in the tree is selected.
- The response WebRoutine invoked by the tree view must have a WebRoutine statement with the keyword Response(*JSON)
- As before the lists are mapped as JSON data.
- Two additional fields must be mapped into the WebRoutine containing the level number being expanded (ONSUBLVL) and the id of the selected entry (ONSUBID).
- Depending on the value of field ONSUBLVL ,the WebRoutine should add entries from the section table (SECTAB) or the employees file (PSLMST).
- Table DEPTAB is keyed on DEPARTMENT
- Table SECTAB is keyed in DEPARTMENT and SECTION
- File PSLMST is keyed on EMPNO
- Field LISTID contains a unique id for each list entry. Its value must be constructed based on these key relationships.
- LISTID for sections = DEPARTMENT + SECTION
- LISTID for employees = EMPNO
- The parent id field LISTPID must be set using the above values.
- The list ancestors is returned by the tree view weblet and contains 1 to 3 entries, depending on the level being expanded. As its name suggests, an entry contains one field corresponding to the parent of the expanding entry.
- The WebRoutine must retrieve the appropriate entry in the ancestors list to construct the key(s) to read the file and expand the selected entry.
- For section entries, field LISTID contains the concatenated value of DEPARTMENT plus SECTION. The value of SECTION can be extracted from LISTID using the SUBSTRING intrinsic function with a start position calculated from the actual length of field DEPARTMENT. For example:

```
#std_num := (#deptment.CurChars + 1)
```

```
.....
```

```
#section := #listid.substring( #std_num )
```

1. Add the following code and then review its logic. Change **iii** to your initials.

```
Webroutine Name(treeexpand) Response(*json)
Web_Map For(*input) Fields((#ancestor *json))
Web_Map For(*output) Fields((#emptree *json))
Web_Map For(*input) Fields(#onsubid #onsublvl)
Clr_List Named(#emptree)
Case (#onsublvl)
When (= '1')
#deptment := #onsubid
Select Fields(#deptment #section #secdesc) From_File(sectab) With_Key(#de
#listid := #deptment + #section
#listcapt := #secdesc
#listpid := #deptment
#haschld := Y
#selwam := iiiTreeView
#selwrn := SECDET
Add_Entry To_List(#emptree)
Endselect
When (= '2')
Get_Entry Number(1) From_List(#ancestor)
#deptment := #listid
#std_num := (#deptment.CurChars + 1)
Get_Entry Number(2) From_List(#ancestor)
#section := #listid.substring( #std_num )
Clr_List Named(#emptree)
Select Fields(#empno #surname #givenname) From_File(pslmst1) With_Key(#c
#listid := #empno
#listcapt := #surname + ', ' + #givenname
#listpid := #deptment + #section
#haschld := N
#selwam := iiiTreeView
#selwrn := EMPDET
* #selwam #selwrn := *blanks
Add_Entry To_List(#emptree)
Endselect
Endcase
Endroutine
```

2. Compile your WAM and open the **deptview** WebRoutine in the *Design* view. Select the tree view and complete setting up its properties, as follows:

| Property | Value |
|------------------------|-------------------|
| onexpand_wname | TREEEXPAND |
| onsubmit_id_field | ONSUBID |
| onsubmit_level_field | ONSUBLVL |
| onsubmit_ancestor_list | ANCESTOR |

3. Save your changes and execute your WAM in the browser. Click the *Expand* icon to test the expanding. (Clicking the text to display the details for this level, will be completed in the next step.)

You should now be able to expand a department to add sections belonging to this department and then expand a section, adding employees belonging to this section.



Step 3. Display Details for a Selected Department

In this step you will add a WebRoutine to display details for the selected department. Details will be displayed on the right hand side of the web page, inside a Navigation panel. The Navigation panel enables output to be sent to this area of the page (in HTML terms the Navigation panel is an iFrame), without refreshing the whole page.

1. Create a WebRoutine **DepDet** to display department details.
 - Field ONSUBID should be mapped into the routine
 - Fields DEPARTMENT and DEPTDESC should be mapped for output with an output attribute
 - Fetch department fields based on the value of ONSUBID

Your code should look like the following

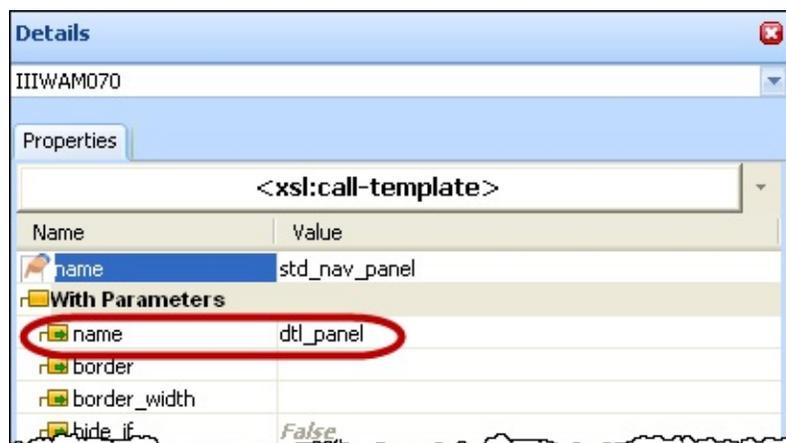
```
WebRoutine Name(DepDet)
Web_Map For(*input) Fields(#onsubid)
Web_Map For(*output) Fields((#deptment *out) (#deptdesc *out))
#deptment := #onsubid
Fetch Fields(*all) From_File(deptab) With_Key(#deptment)
Endroutine
```

2. Compile your WAM.
3. Open the **DeptView** WebRoutine in the *Design* view. In this step you will add a table with 1 row and 2 columns to the web page, and move the tree view into its left column.
 - a. Position your cursor to the right of the Tree View and press enter to insert a blank line below it.
 - b. Use the context menu to *Insert HTML / Table* with 1 row and two columns, below the tree view.
 - c. Select the tree view and use the context menu to *Cut* it.
 - d. Position the cursor in the left hand column of the table and use the context menu to *Paste* the tree view into it.
 - e. Save your changes. Your web page should now look like the following:



4. In this step you will add a Navigation panel weblet into the right hand cell of the table and set up the tree view to output details for a selected entry to this nav panel.
 - a. Drop a Navigation panel weblet into the right hand cell of the table.
 - b. Select the Navigation panel. Use the *Details* tab to change its name to `dtl_panel`.

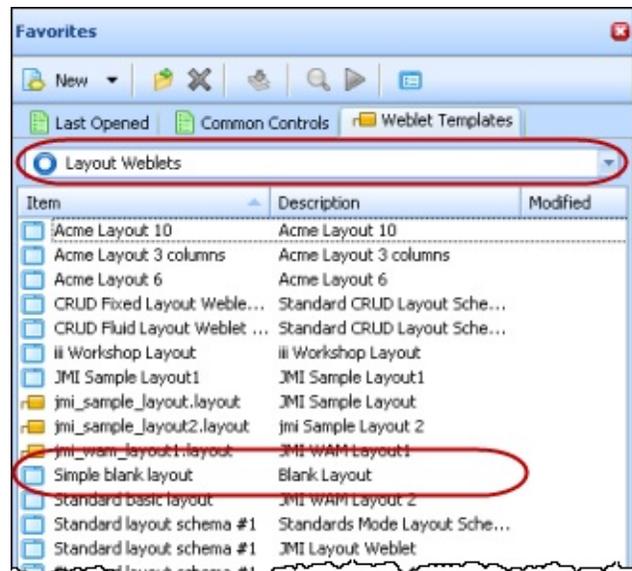
Note: You are changing the *name* property below the *With Parameters* heading:



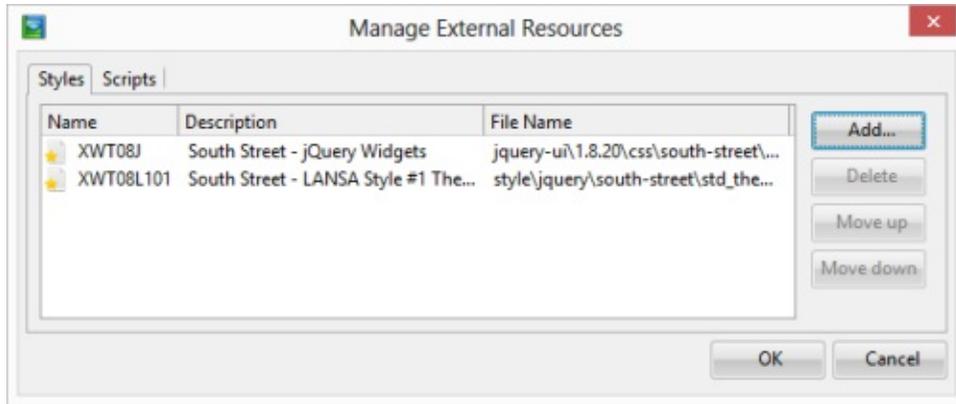
- c. Change the Navigation panel *border* property to **none**.
- d. Select the tree view. Using the *Details* tab, change its *target_window_name* property to `dtl_panel`.
- e. Change the tree view's *node_text_click* property to `Select`. This will select

an entry to display details when the entry's **text** is clicked.

- f. Remove the place holder characters from the table cells.
 - g. With the cursor positioned in the left hand table cell, change its *vAlign* property to top. This will position the tree view at the top of the table cell.
 - h. With the cursor positioned in the right hand table cell, change its *vAlign* property to top.
 - i. Save your changes.
5. In this step you will set up the web page for WebRoutine DepDet, by giving it a blank layout. This WebRoutine will be displayed within the Navigation panel on the **DeptView** web page and therefore does not require a layout.
- a. Open the **DepDet** WebRoutine in the Design view.
 - b. On the *Favorites / Weblet Templates* tab, select Layout Weblets in the top combo box:



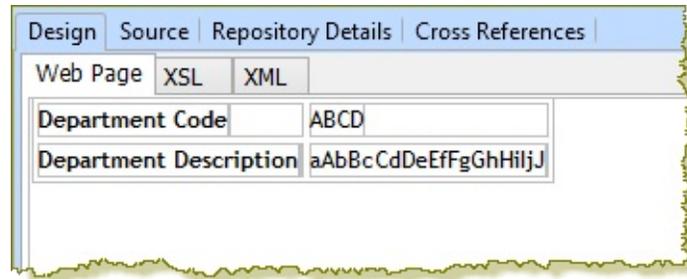
- c. Drag and drop the Simple blank layout onto the web page.
- d. From the *Design* ribbon select External Resources button. In the *Manage External Resources* dialog, add the Style external resources which are required by your chosen common standard layout.



This will add these style sheets into the XSL for this WebRoutines web page and ensure that its contents are consistent with your common layout theme.

Note: If necessary, find your common layout (iiilay01) and use the context menu Cross References option to find the external resources being used.

Your design should now look like the following:



6. Save your changes.

Step 4. Display Details for Sections and Employees

In this step you will complete the WAM by adding WebRoutines to display details for sections and employees.

1. Define a Group_by SECDATA for fields SECTION, SECDESC, SECADDR1, SECADDR2 SECADDR3, SECPCODE and SECPHBUS. All fields should have an output attribute.
2. Create a new WebRoutine **SecDet**, based on the following:
 - Map for input field ONSUBID and list ANCESTOR
 - Map for output the Group_by SECDATA
 - Retrieve the first entry from list ANCESTOR and set the value of DEPARTMENT from LISTID
 - Calculate STD_NUM based on the actual length of field DEPARTMENT + 1
 - Assign SECTION by substringing from ONSUBID, starting from STD_NUM
 - Fetch department fields with the key DEPARTMENT and SECTION

Your code should look like the following:

```
Webroutine Name(SecDet)
Web_Map For(*input) Fields(#onsubid #ancestor)
Web_Map For(*output) Fields(#secddata)
Get_Entry Number(1) From_List(#ancestor)
#deptment := #listid
#std_num := (#deptment.CurSize + 1)
#section := #onsubid.substring( #std_num )
Fetch Fields(*all) From_File(sectab) With_Key(#deptment #section)
Endroutine
```

3. Compile the WAM.
4. Define a Group_by **EMPDATA** for fields EMPNO, SURNAME, GIVENAME, ADDRESS1, ADDRESS2, ADDRESS3, POSTCODE, PHONEHME, PHONEBUS, DEPARTMENT, SECTION, SALARY, STARTDTE, TERMDATE. All fields should have an output attribute.
5. Define a working list **SKILLS**, for fields SKILCODE, SKILDESC, GRADE, COMMENT, DATEACQ. All fields should have an output attribute.

6. Open the WebRoutine **SecDet** in the *Design* view. Drop the Simple blank layout onto the page.
7. Use the *Web / Manage External Resources* menu option to give the web page the theme styles to match your common layout.
8. Save your changes.
9. Create a new WebRoutine **EmpDet**.
 - Map field ONSUBID and list ANCESTOR for input
 - Map Group_by and list SKILLS for output
 - Assign EMPNO to the value of ONSUBID
 - Fetch employee data from file PSLMST using key EMPNO
 - Clear the list SKILLS
 - Build the list SKILLS from file PSLSKL with key EMPNO
 - Fetch SKILDESC for each employee skill with key SKILCODE

Your code should look like the following:

```

Webroutine Name(EmpDet)
Web_Map For(*input) Fields(#onsubid #ancestor)
Web_Map For(*output) Fields(#empdata #skills)
#empno := #onsubid
Fetch Fields(#empdata) From_File(pslmst) With_Key(#empno)
Clr_List Named(#skills)
Select Fields(#skills) From_File(pslskl) With_Key(#empno)
Fetch Fields(#skildesc) From_File(skltab) With_Key(#SKILCODE)
Add_Entry To_List(#skills)
Endselect
Endroutine

```

10. Compile the WAM.
11. Open the **EmpDet** WebRoutine in the *Design* view. Drop the Simple blank layout onto the web page.
12. On the *Design* ribbon use the *External Resources* button, then *Manage External Resources* dialog to add the Style external resources to match those used in your common layout.
13. Open the **SecDet** WebRoutine in the *Design* view. Drop the Simple blank layout onto the web page.

14. On the *Design* ribbon use the *External Resources* button, then *Manage External Resources* dialog to add the *Style* external resources to match those used in your common layout.
15. Save your changes.
16. Test your WAM by running the **DeptView** WebRoutine in the browser. You should be able to expand the tree view and select a department, a section or an employee to display its details.
17. You probably found that the Navigation panel needed more space to display employee details.
 - a. Open the **DeptView** WebRoutine in the *Design* view
 - b. Select the table containing the tree view and Navigation panel. Do this by selecting a corner of the table. Alternatively, click anywhere inside the table and use the *Outline* tab to locate and select the table.
 - c. Drag the right hand edge of the table to the right.
 - d. Drag the bottom of edge of the table down.
 - e. If you examine the *Style* properties for the table you will see its current size in pixels (for example, width: 800px and height: 560px).
 - f. You can also hover over the center border and drag this to the left to match the size of the tree view weblet.
 - g. If necessary adjust the size of the Navigation panel to use the space now available
 - h. Save your changes
 - i. Re-test your WAM.

Summary

Important Information

- This exercise introduces the basics of building an unlevelled tree, handling its expansion and displaying details for any level. See the *Web Application Modules* guide for more information.
- Other fields in the tree working list may be defined to control open and closed images, and whether an entry is currently selected or expanded.
- The tree view is supported in all LANSAs supported browsers.

Tips & Techniques

- A tree view could be used as an application menu.

What You Should Know

- How to set up and use a tree view weblet for enquiry purposes.

WAM080 - Session Management

Objectives

This exercise initially shows how session management operates using a single WAM.

WAM **iiiSessionMng** demonstrates how various WebRoutines within the same WAM can manipulate and share data that is automatically stored on the server. There are a number of key concepts to understand when implementing session management.

- Session management must be enabled at the WAM level. For example
`Begin_Com Role(*EXTENDS #PRIM_WAM) Layoutweblet('iilay01') Sessio`
- At least one WebRoutine must have an `Onentry()` keyword of `*sessionstatus_none`. This enables the WebRoutine to execute in order to activate a session.
- Other WebRoutines have a default `Onentry()` keyword of `*sessionstatus_of_wam`. If sessions are enabled for the WAM, then a session must be active for this WebRoutine to run.
- The WAM should contain an event handling routine for `#com_owner.sessioninvalid` that determines what happens when a session is invalid or expired. Usually this will transfer to the "login" WebRoutine to force a session to be established.
- Fields and lists to be stored on the server are defined via a special `web_map`. For example the following defines two working lists that are to be stored as persistent data:

```
Web_Map For(*none) Fields(#empsave #empdata) Options(*PERSIST)
```

The `For(*none)` keyword value means the fields are not mapped to and from the web page.

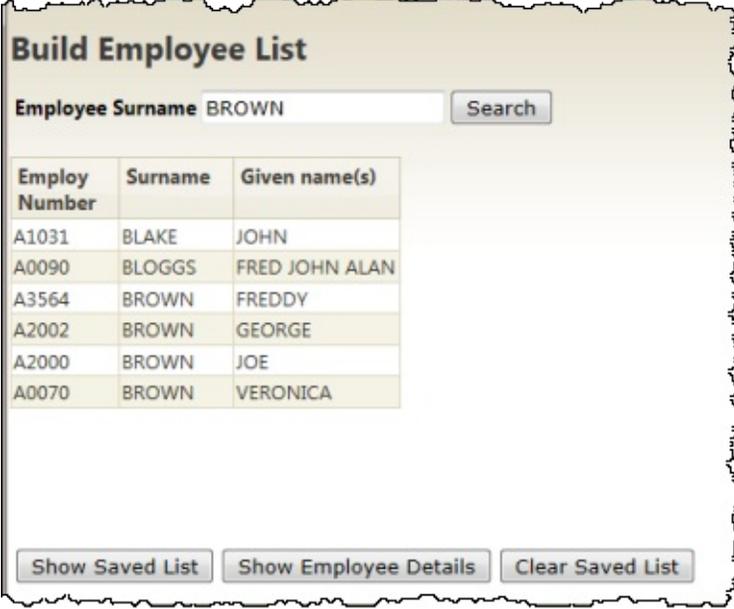
- As long as the session is active, each time the WAM starts, the persistent fields and lists are restored.
- Each time the WAM ends, the persistent fields and lists are saved.
- It's important to understand that as each WebRoutine executes, it shares the same persistent fields and lists and their current values as established by the last WebRoutine to execute.

For more details refer to [WAM Session Management](#).

- A second WAM will be developed that shares the session established by iiiWAM070. This will demonstrate how more than one WAM may share a session and the persistent fields and lists which session management enables.
- WAMs share a session by having a BEGIN_COM statement that declares Sessionstatus active and has a common Session groupname. For example:
Begin_Com Role(*EXTENDS #PRIM_WAM) Layoutweblet('iiilay01') Sessio

Description of WAM iiiSessionMng

The **Search** WebRoutine web page will look like the following:



| Employ Number | Surname | Given name(s) |
|---------------|---------|----------------|
| A1031 | BLAKE | JOHN |
| A0090 | BLOGGS | FRED JOHN ALAN |
| A3564 | BROWN | FREDDY |
| A2002 | BROWN | GEORGE |
| A2000 | BROWN | JOE |
| A0070 | BROWN | VERONICA |

- The WebRoutine search will build and display a list of employees, searching the file by surname. Each search adds entries to a working list that is displayed and a second persistent working list which is stored on the server.
- WebRoutine showsave is called by a push button on the search web page. This second routine builds and displays a list from the **saved list**.
- The Clear Saved List pushbutton invokes the **search** WebRoutine to clear the saved list.
- Other functionality will be added in later steps.

To meet these objectives you will complete the following:

- [Step 1. Create Session Management 1 WAM](#)
- [Step 2. Retrieve and Store Employee Details](#)

- [Step 3. Create Session Management 2 WAM](#)
- [Step 4. Test the Session Management Application](#)
- [Summary](#)

Before You Begin

You should complete all preceding exercises in this workshop.

Step 1. Create Session Management 1 WAM

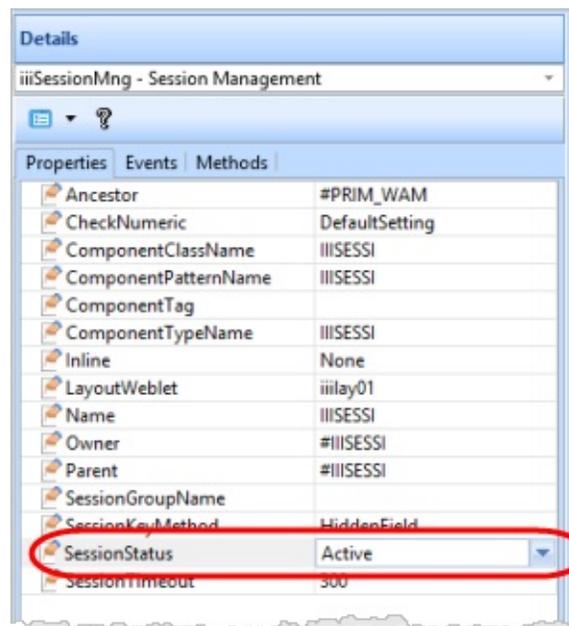
1. Create a new WAM:

Name: **iiiSessionMng**

Description: **Session Management 1**

Layout Weblet: **iiilay01**

2. Press F7 to display the WAM properties. To enable session management, set *SessionStatus* to **Active**.



3. Define the following lists and global web maps:

* list of employees to be saved on the server

```
Def_List Name(#empsave) Fields(#empno #surname #givename) Counter(#stdselect)
```

* latest search list of employees

```
Def_List Name(#empnew) Fields((#empno *out) (#surname *out) (#givename *out))
```

* display current saved list of employees.

```
Def_List Name(#empdisp) Fields(#stdselect (#empno *out) (#surname *out) (#givename *out))
```

* Map persistent data

```
Web_Map For(*none) Fields(#empsave) Options(*PERSIST)
```

* Map common return field from weblets

```
Web_Map For(*input) Fields((#stdrentry *hidden))
```

Note that working list EMPSAVE is mapped as persistent data.

4. Create three WebRoutines based on the following:

```
* Initialize WebRoutine sets sessionstatus active
WebRoutine Name(init) Onentry(*SESSIONSTATUS_NONE)
#com_owner.sessionstatus := active
Message Msgtxt('Session is now active')
Transfer Toroutine(search)
Endroutine
* Perform search and display results.
WebRoutine Name(search) Desc('Build a list of employees')
Web_Map For(*both) Fields(#surname)
Web_Map For(*output) Fields(#empnew)
Endroutine
* Load and display a list, from the saved list
WebRoutine Name(showsave) Desc('Show saved list of employees')
Web_Map For(*output) Fields(#empdisp)
Endroutine
```

Note that as outlined under *Objectives* above, the init WebRoutine may be executed before a session is active.

5. Add the initial logic to WebRoutine **search**.

- when STDREENTRY is S
- This should clear the list EMPNEW, which is built by each search.
- Ensure the surname is not blank
- Build the list EMPLIST by reading the logical file PSLMST2, with a key of surname, with generic(*yes).
- Add entries to both EMPNEW and the saved list EMPSAVE.

Your code should look like the following:

```
Case (#stdreentry)
When (= S)
Clr_List Named(#empnew)
Begincheck
Valuecheck Field(#surname) With_List(*BLANK) In_List(*ERROR) Not_Inli
Endcheck
Select Fields(#empsave) From_File(pslmst2) With_Key(#surname) Nbr_Keys(
Add_Entry To_List(#empnew)
```

```
Add_Entry To_List(#empsave)
Endselect
Endcase
```

6. Add the initial logic to the **showsave** WebRoutine

- Output the list EMPDISP which is loaded from the save list EMPSAVE.
- When STDREENTRY is L
- If the saved list counter value (STD_COUNT) is greater than 1
 - Clear the list EMPDISP
 - Read all records from EMPSAVE using SELECTLIST
 - Add entries to EMPDISP
- Else
 - Output message 'Saved list of employees is empty'
 - Transfer to **search** WebRoutine.

Your code should look like the following:

```
Case (#stdreentry)
When (= L)
If (#std_count > 1)
Clr_List Named(#empdisp)
Selectlist Named(#empsave)
Add_Entry To_List(#empdisp)
Endselect
Else
Message Msgtxt('Saved employee list is empty')
Transfer Toroutine(search)
Endif
Endcase
```

7. Add an event handling routine for session invalid to transfer to the search **init** WebRoutine. Your code should look like the following:

```
Evtroutine Handling(#COM_OWNER.sessioninvalid)
Message Msgtxt('Session Management must be active')
Transfer Toroutine(init)
Endroutine
```

8. Compile the WAM.
9. Open the **Search** WebRoutine in the *Design* view. Add a column to the table containing employee surname. Drop a push button into the new column. Set up the button properties:

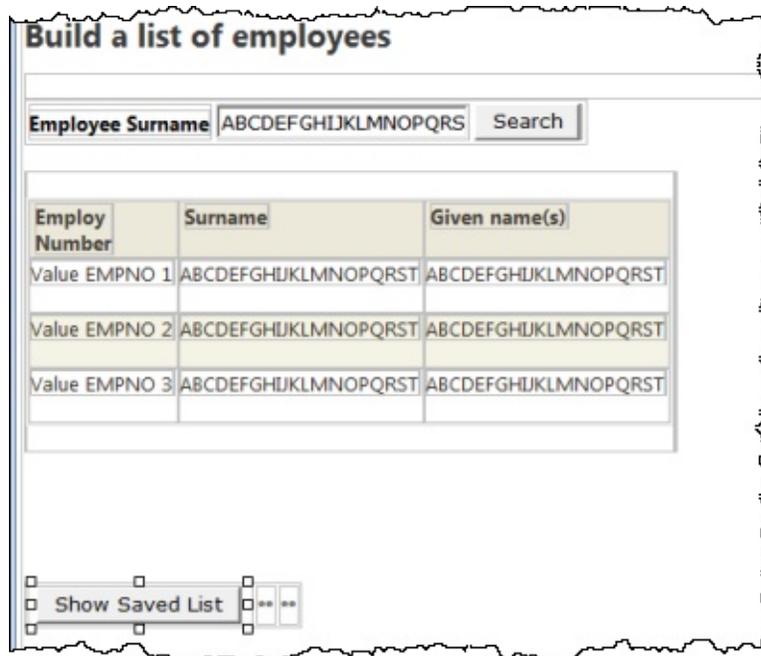
| Property | Value |
|-------------------|-------------------------------|
| caption | Search |
| on_click_wrname | Search |
| submitExtraFields | Field Name: STDREENTRY |
| | Literal Value: S |

10. Add a few blank lines below the table containing the employees list and insert a table with 1 row and 3 columns. Drop a push button into the left hand column and set up the button properties:

| Property | Value |
|-------------------|-------------------------------|
| Caption | Show Saved List |
| On_click_wrname | Showsave |
| submitExtraFields | Field Name: STDREENTRY |
| | Literal Value: L |

Save your changes.

Your page should look like the following:



11. Open the **showsave** WebRoutine in the *Design* view. Select the list, move the cursor right and press enter to create a blank line below the list. Drop a push button below the list and set up its properties:

| Property | Value |
|-------------------|-------------------------------|
| Caption | Return |
| On_click_wname | Search |
| submitExtraFields | Field Name: STDREENTRY |
| | Literal Value: M |

Your web page should look like the following:

Show saved list of employees

| Std *WEBEVENT
template field | Employ
Number | Surname | Given name(s) |
|---------------------------------|------------------|----------------------|----------------------|
| A | Value EMPNO 1 | ABCDEFGHIJKLMNOPQRST | ABCDEFGHIJKLMNOPQRST |
| A | Value EMPNO 2 | ABCDEFGHIJKLMNOPQRST | ABCDEFGHIJKLMNOPQRST |
| A | Value EMPNO 3 | ABCDEFGHIJKLMNOPQRST | ABCDEFGHIJKLMNOPQRST |

Return

12. Save your changes.

13. Execute your WAM in the browser by running any WebRoutine. Control will pass to the **init** WebRoutine which will enable session management, and transfer to the **search** WebRoutine.

You should be able to see the following results:

- A list of employees based on a search value such as B or S.
- Display the current saved list in WebRoutine **showpage**.
- Return to the WebRoutine **search**.
- Perform a different search and display the result of both searches in the **showpage** WebRoutine.
- Restart the WAM from the **init** WebRoutine and immediately display the saved list.

Q. Why is the list empty?

A. The persistent data is only restored if the session is already active. Running WebRoutine **init** starts a new session.

Step 2. Retrieve and Store Employee Details

- In this step you will extend the WebRoutine **showsave**, to fetch employee details. This will be invoked via a clickable image weblet in the list EMPDISP output by WebRoutine **ShowSave**.
- Employee details will be stored in a new working list EMPDATA that will hold one entry and will also be mapped as persistent data.
- A new WebRoutine **showemp** will display the stored employee details.
- The **search** WebRoutine will be extended to handle clearing the lists EMPSAVE and EMPDATA.

1. Define a working list EMPDATA for employee details:

```
Def_List Name(#empdata) Fields(#EMPNO #SURNAME #GIVENAME #AD
```

2. Define a group by to display employee fields

```
Group_By Name(#empgrp) Fields((#EMPNO *out) (#SURNAME *out) (#GI
```

3. Extend the web_map for persistent data to include list EMPDATA.

```
Web_Map For(*none) Fields(#empsave #empdata) Options(*PERSIST)
```

4. Extend case loop in the search WebRoutine to clear the saved lists.

```
When (= C)  
Clr_List Named(#empsave)  
Clr_List Named(#empdata)  
Message Msgtxt('Saved employee list was cleared')
```

5. Extend the case loop in the **showsave** WebRoutine to fetch employee data and add an entry to the list EMPDATA.

```
When (= D)  
Clr_List Named(#empdata)  
Fetch Fields(#empgrp) From_File(pslmst) With_Key(#empno) Val_Error(*nex  
If_Status Is(*okay)  
Add_Entry To_List(#empdata)  
Message Msgtxt('Employee details saved')  
Endif  
Transfer Toroutine(search)
```

- In the **showsave** WebRoutine, add a web_map for input for field EMPNO. This value will be passed in for the selected row, by the clickable image.

```
Web_Map For(*input) Fields(#empno)
```

- Create a new WebRoutine **showemp** to retrieve the one entry from the list EMPDATA or transfer to the **search** WebRoutine.

```
WebRoutine Name(showemp) Desc('Show Saved Employee')
Web_Map For(*output) Fields(#empgrp)
If (#listcount = 1)
Get_Entry Number(1) From_List(#empdata)
Else
Message Msgtxt('Employee details not available')
Transfer Toroutine(search)
Endif
Endroutine
```

- Compile your WAM.
- Open the search WebRoutine in the Design view.
- Add a push button into the table at the bottom of the page. Set up the button properties:

| Property | Value |
|-------------------|-------------------------------|
| caption | Show Employee Details |
| on_click_wrname | showemp |
| submitExtraFields | Field Name: STDREENTRY |
| | Literal Value: D |

- Add a third push button into the table at the bottom of the page and setup the button properties:

| Property | Value |
|----------|--------------------------|
| | Clear Saved Lists |

| | |
|-------------------|-------------------------------|
| caption | |
| on_click_wname | search |
| submitExtraFields | Field Name: STDREENTRY |
| | Literal Value: C |

12. Remove the place holder characters from this table.
13. Save your changes.
14. Open the **showsave** WebRoutine in the Design view.
15. Select and delete the column heading for the first column.
16. Drop a clickable image into the first column (field STDSELECT). Set up the clickable image properties:

| Property | Value |
|--------------------|-----------------|
| currentrowhfield | EMPNO |
| Currentrownumvalue | \$EMPNO |
| Reentryvalue | D |
| On_click_wname | showsave |

17. Save your changes.
18. Open the **showemp** WebRoutine in the *Design* view. Drop a push button onto the page below the list. Set up the button properties:

| Property | Value |
|-------------------|-------------------------------|
| Caption | Return to Search |
| On_click_wname | search |
| submitExtraFields | Field Name: STDREENTRY |
| | Literal Value: M |

19. Save your changes.

20. Test your WAM.

a. You should now get the following results:

- Retrieve and display employee search results
- Display saved list of employees
- Select an employee in the saved list using the clickable image
- Display selected employee and return to the search page
- Display the saved employee data
- Clear both saved employee list and employee data

b. Start your WAM from any WebRoutine except WebRoutine init. You should be transferred to the **init** WebRoutine to establish a session, and then transferred to the **search** WebRoutine, displaying suitable messages.

21. In your WAM source, use F7 to display WAM properties on the *Details* tab. Change the *Session Timeout* from its default value (**300** seconds) to **10** and recompile your WAM.

22. Test your WAM.

Perform a search and display the saved list. This should be displayed. However, due to the very short time out (10 seconds) by the time you return to the search web page, the WAM will have timed out, giving appropriate messages. If you then immediately display the saved employee list, there will be no entries.

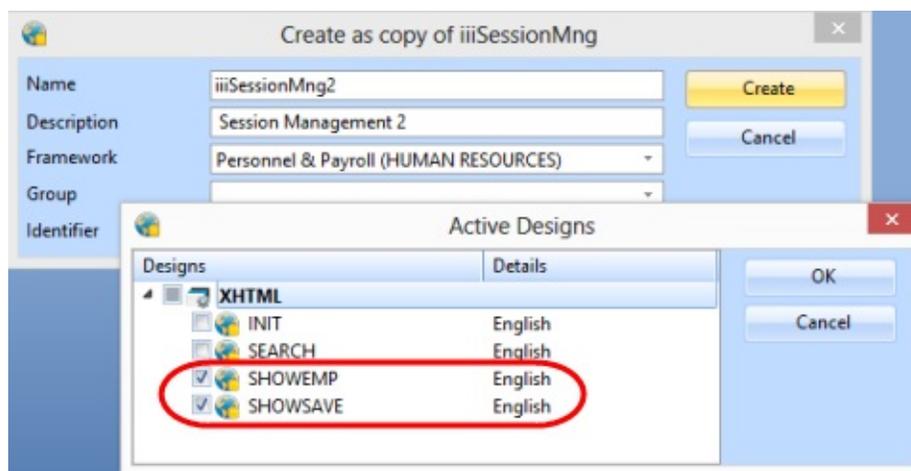
Note: The session timeout value is the "wait time" for the response from the client (the browser). So with a session timeout of 10 seconds, whenever you delay for more than 10 seconds, the session will time out. If on the other hand you keep sending requests to the server within the 10 second time out window, the session will never time out.

Persistent data will only be retrieved if the session is active. If a new session is established, due to time out, there is currently no persistent data for that session.

Step 3. Create Session Management 2 WAM

In this step you will create a second WAM and enable session management in both WAMs so that they share a common Session GroupName. This makes both WAMs part of the same session. The new WAM will contain the WebRoutines showsave and showemp and the session invalid event handling routine.

1. Create WAM `iiiSessionMng2` by copying WAM `iiiSessionMng`. Perform the copy step as follows:
 - a. Select the WAM `iiiSessionMng` on the Favorites / Last Opened tab and *Copy* it using the context menu.
 - b. In the *Active Designs* dialog, select only `SHOWEMP` and `SHOWSAVE`.



2. Delete the WebRoutines **INIT** and **SEARCH** from the new WAM.
3. Change four transfer commands to specify the WAM name and WebRoutine name, for example:
Transfer Toroutine(#iiiSessionMng.search)
4. Press F7 to display the WAM properties on the *Details* tab. Change the Session GroupName to `IIIGROUP`.
Set the Session Timeout value to 300.
5. Compile WAM **iiiSessionMng2**.
6. Open the `showsave` WebRoutine in the *Design* view and give the *Return* push button an `on_click_wamname` property of `iiiSessionMng`.
7. Open the **showemp** WebRoutine in the *Design* view and give the *Return to Search* push button an `on_click_wamname` property of `iiiSessionMng`.

8. Save your changes.
9. If necessary open WAM **iiiSessionMng** in the editor.
10. Open the search WebRoutine in the *Design* view. Make the following changes:

| Push Button | Property | Value |
|-----------------------|------------------|-----------------------|
| Show Saved List | on_click_wamname | iiiSessionMng2 |
| Show Employee Details | on_click_wamname | iiiSessionMng |

- 11 Save your changes.
12. Press F7 to display the WAM properties on the Details tab. Change the Session GroupName to IIIGROUP.
Ensure the Session Timeout value to 300.
13. Compile WAM **iiiSessionMng**.

Step 4. Test the Session Management Application

You can now test the application by executing any WebRoutine in either of the two WAMs. The session invalid event handling routine will transfer to the init WebRoutine in iiiSessionMng, which will make the session active.

- The application should behave exactly the same as before. The show saved list, retrieve employee details and display saved employee details logic is now performed by WAM iiiSessionMng2.

Summary

Important Observations

- Session Management and persistent data provide powerful and easy to use functionality.
- Applications that handle sensitive data can be made to time out after a given wait time
- Your application could establish a common session when the user logs in.
- Common data that is established at log in could be made available to all WAMs via persistent data.
- Data required in a later step can be stored as persistent data.
- Persistent data involves additional database I/Os. Consider carefully how much data needs to be stored and restored as persistent data.

Tips & Techniques

- Persistent data is secure because it is stored on the server.
- Persistent data is secure and avoids the need to map data as hidden in and out of the web page which may introduce opportunities to "hack" your application.
- Persistent data is only available for the duration of the session. If you need to make the data permanent, then your application must store it in a database.
- Before implementing Session Management in your own applications, see the Web Application Modules guide for more detailed information on Session Management

What You Should Know

- How to implement session management.

WAM085 - Enhancing the User Interface

Objectives

- To demonstrate a number of weblets which enhance the user interface provided by your web page.
- To illustrate standard field visualization weblets, which are automatically added for certain field types.
- To demonstrate some of the properties of these field visualization weblets

To meet these objectives you will complete the following steps:

- [Step 1. Create Repository Field Definitions](#)
- [Step 2. Create Employee Number AutoComplete WAM](#)
- [Step 3. Create WAM iiiEnhancedUI – Enhancing the Interface](#)
- [Step 4. Define Work Fields and Lists](#)
- [Step 5. Complete WAM RDMLX](#)
- [Step 6. Design the web pages](#)
- [Step 7. Test the WAM](#)
- [Step 8. Improve the ShowPage Page Design](#)
- [Step 9. Insert a fieldset around each table](#)
- [Summary](#)

Before You Begin

You should first complete all preceding exercises in this workshop

Step 1. Create Repository Field Definitions

Field Visualizations

Fields defined in the Repository may also have *Field Visualizations* defined.

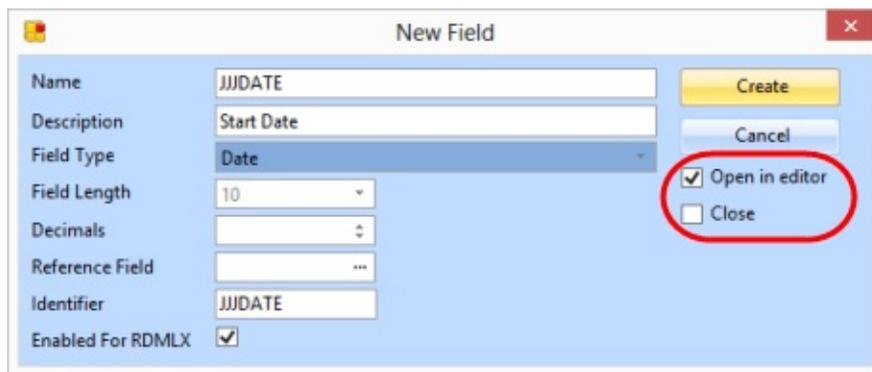
- These visualizations are able to define the field's appearance in Windows desktop applications and / or in web applications built using WAMs.
- For web, field visualizations are weblets.
- Fields may have a number of visualizations for Windows and web.
- One field visualization must be defined as the default visualization, which will be the appearance option that is automatically selected.
- For web, just one weblet visualization should be defined, since at design time there is no way to switch between visualizations, except by deleting one and adding another.
- Fields may have a *static picklist* defined which defines a set of descriptions (or captions) and values. Weblets such as a combo box or radio button group will use this picklist to populate the selection options in the web page.
- Field visualizations are compiled into the web page at design time. A change to the definition of a field's visualization in the *Repository*, for example changing a radio group to a combo box, or adding or removing entries in the picklist, can only be reflected in the web page by removing that field from the web page and re-adding it. It follows that field visualizations defined in the *Repository* with a static picklist, are only useful for very static data that is unlikely to ever change, such as "male or female" or "married, single or divorced".
- Note also that field types such as Date, DateTime and String will probably always benefit from having a weblet such as datepicker, so their automatic visualization definitions in the Repository are always useful.
- There are other field visualization options that only apply to Windows desktop applications, that are not discussed further here.
- See the *Visual LANSA Developer Guide* for more information on field visualizations.

1. Create new field definitions in the Repository:

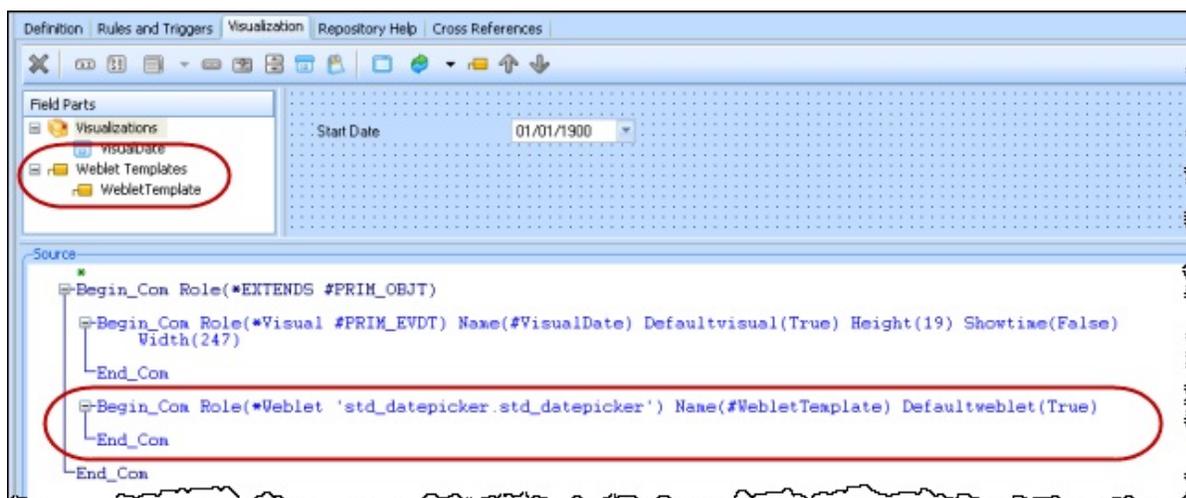
| Name | Description | Type | Length | Input Attribute |
|-------------|--------------------|-------------|---------------|------------------------|
| iiiDATE | Start Date | Date | 10 | |

| | | | | |
|-----------|------------------|--------------|-----|----|
| iiiDTETME | Last Updated | DateTime | 25 | |
| iiiNOTE | Comments | String | 512 | LC |
| iiiGENDER | Gender | Alphanumeric | 1 | |
| iiiTIME | Create Time | Time | 8 | |
| iiiCUREMP | Current Employee | Alphanumeric | 1 | |

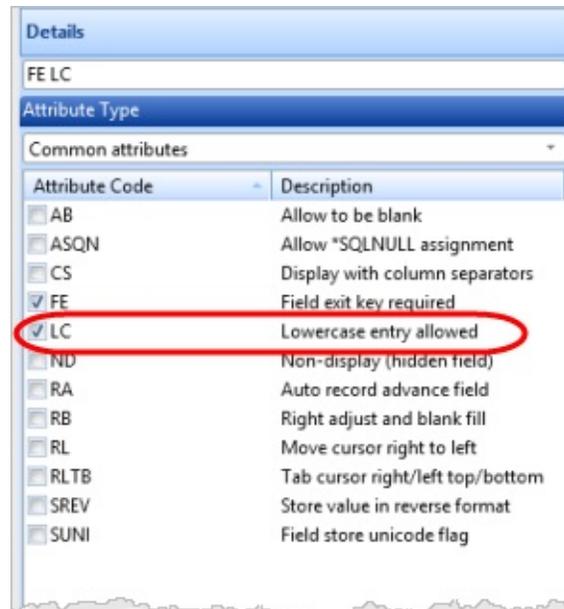
Use the New Field dialog, to open each field in the Editor and for the dialog to remain open. Fields will be automatically flagged as RDMLX when a type such as Date is selected:



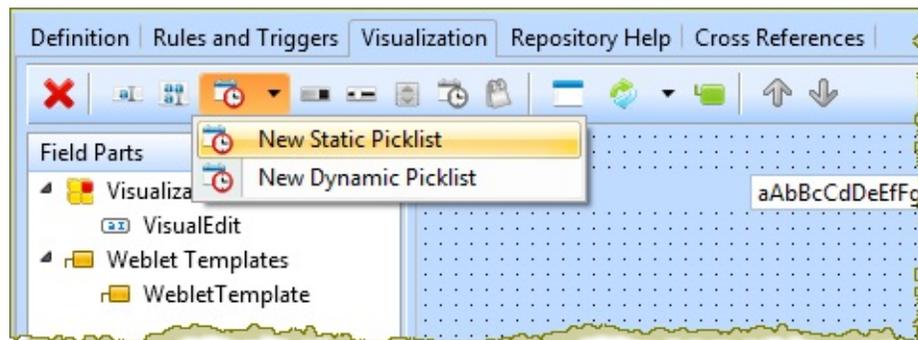
- Review the *Field Visualization* for the fields iiiDATE, iiiDTETME, iiiTIME and iiiNOTE. They each have been given a suitable weblet visualization.



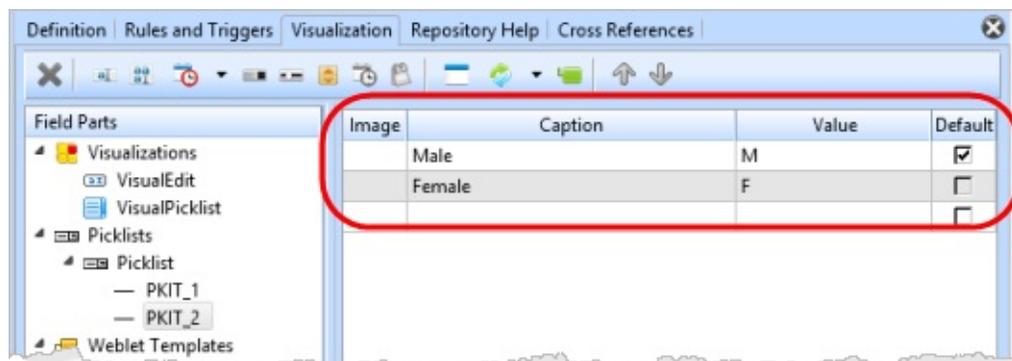
- Ensure that the *Input* attribute of field iiiNOTE allows lower case:



4. Switch to the *Field Visualization* definition for field iiiGENDER. Add a Static Picklist using the toolbar button:

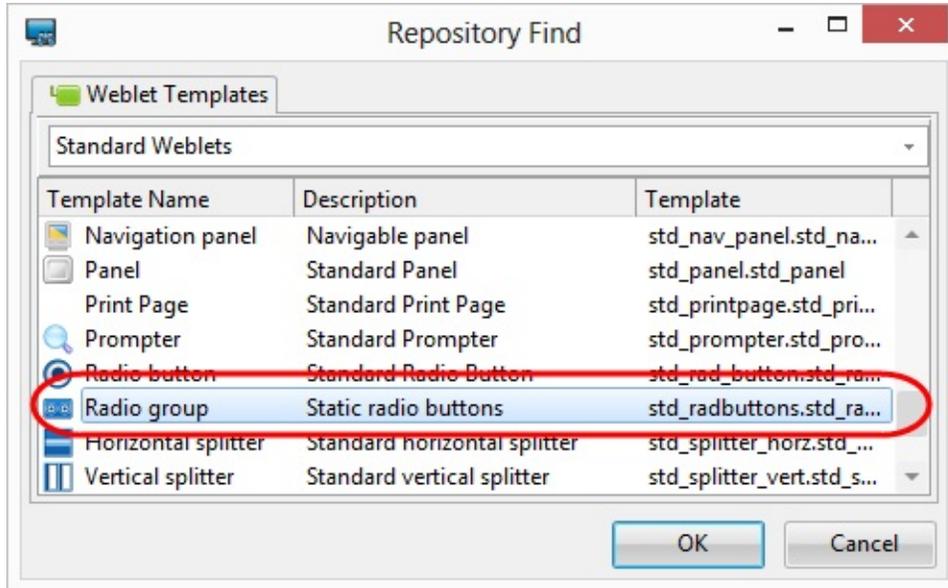


5. Define the static picklist with two entries as shown:



6. Open the *Repository Find* dialog using the  *insert Weblet Template* toolbar button.

- a. If necessary, in the *Repository Find* dialog, select *Standard Weblets* in the dropdown.
- b. Add a *Radio Group* weblet.



- 7. Save the field definition.
- 8. Switch to the *Visualization* tab for field iiiCUREMP. Use the  toolbar button to add a *Checkbox* weblet visualization.

The *Checkbox* weblet will return, by default, values of Y and N which are correct values for field iiiCUREMP. In this case a static picklist is not required.
- 9. Save the field definition.

Step 2. Create Employee Number AutoComplete WAM

This step illustrates how you can create a small response WAM to service any web page that uses an AutoComplete weblet for field employee number (EMPNO).

1. Create a new WAM:

Name: **iiiEmpAutoCmpl**

Description: **Employee Number AutoComplete**

Layout Weblet: **iiilay01**

2. Copy the following WebRoutine from WAM **iiiEmpSearch**.

```
WebRoutine Name(Empno_Prompt) Response(*JSON)
Web_Map For(*input) Fields(#empno)
Web_Map For(*output) Fields((#emp_dd *json))
Def_List Name(#emp_dd) Fields(#empno #std_code) Counter(#std_count) Typ
Clr_List Named(#emp_dd)
Select Fields(#emp_dd) From_File(pslmst) Where(#std_count <= 3) With_Key
#std_code := #empno
Add_Entry To_List(#emp_dd)
Endselect
Endroutine
```

3. Compile the new WAM.

Step 3. Create WAM **iiiEnhancedUI** – Enhancing the Interface

This WAM will use the new fields that you have just created. It will also build on the previous exercise, by enabling session management and saving a working list of each set of data entered.

This WAM will operate as follows:

- Executes WebRoutine **begin** to start session management and transfers to the **select** WebRoutine.
- The **select** WebRoutine requests input of an employee number. The **showpage** WebRoutine is invoked via a push button.
- The **select** WebRoutine also supports a push button to clear the saved list (SAVLIST).
- The **select** WebRoutine includes a push button to invoke the **showlist** WebRoutine.
- The **showpage** WebRoutine fetches employee data and displays all fields (except employee number) for input
- A Save button invokes the showpage WebRoutine to save the data to the working list SAVLIST and transfers back to the **select** WebRoutine.
- The **showlist** WebRoutine loads the saved list (SAVLIST) into the list EMPLIST and displays this list.

1. Create a new WAM:

Name: **iiiEnhancedUI**

Description: **Enhancing the Interface**

Layout weblet: **iiilay01**

2. Create the following four WebRoutines:

* Start session and transfer to select

Webroutine Name(begin) Onentry(*sessionstatus_none)

Endroutine

* Request an employee number

Webroutine Name(select) Desc('Select an Employee')

Endroutine

* Fetch employee and accept input

Webroutine Name(showpage) Desc('Enter employee details')
Endroutine

* Display the saved working list

Webroutine Name(showlist) Desc('Saved list of employee data')
Endroutine

3. Note that the **begin** WebRoutine has an **onentry()** keyword of ***sessionstatus_none**. This will be the first WebRoutine executed to start session management.

Use the *F7* key to display the WAM properties on the *Details* tab and enable session management:

The screenshot shows a software development environment with two main panes. The left pane, titled 'Details', shows the 'Properties' tab for a component named 'EnhancedUI - Enhancing the User Interface'. The 'SessionStatus' property is set to 'Active' and is circled in red. The right pane, titled 'Source', shows the source code for a component named 'STD_WAM'. The code includes a 'Function Options(*DIRECT)' block and a 'Begin_Com Role(*EXTENDS #PRIM_WAM)' block. The 'Begin_Com Role' block is circled in red and contains the following code:

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_WAM)
  Layoutweblet('iiilay01') Sessionstatus(Active)
  Define Field(#marstat) Type(*char) Length(1)
    Desc('Marital Status') Default('M')
  Define Field(#onleave) Reffld(#std_flag)
    Desc('On Leave') Colhdg('On' 'Leave')
    Default('N')
  Define Field(#review) Type(*string) Length(512)
    Desc('Last Review Notes') Input_Atr(LC)
  Define Field(#termdte) Type(*date)
    Desc('Terminate Date') Input_Atr(ASQN ISO)
    Default(*SQLNULL)
  Define Field(#empno) Reffld(#empno)
  Def_List Name(#savlist)
  Fields(#empno #GIVENAME #SURNAME #IIIDATE
    #IIIDTETIME #IIITIME #IIICUREMP #IIIGENDER
    #IIINOTE #marstat #onleave #review #termdte
```

Step 4. Define Work Fields and Lists

1. In this step you will define additional fields that will be included on the data entry web page **ShowPage**. The UI for these fields will be manually set up using weblets.

Define the following additional fields at the top of your WAM definition:

```
Define Field(#marstat) Type(*char) Length(1) Desc('Marital Status')
Default('M')
Define Field(#onleave) Reffld(#std_flag) Desc('On Leave') Colhdg('On'
'Leave') Default('N')
Define Field(#review) Type(*string) Length(512) Desc('Last Review Notes')
Input_Atr(LC)
Define Field(#termdte) Type(*date) Desc('Terminate Date') Input_Atr(ASQN
ISO) Default(*SQLNULL)
```

2. Define the following work field and working lists. Ensure that you change **iii** to your initials.

```
Define Field(#empnow) Reffld(#empno)
Def_List Name(#savlist) Fields(#empno #GIVENAME #SURNAME #IIIDAT
Def_List Name(#emplist) Fields((#empno *out) (#GIVENAME *out) (#SURNAME
```

Note that fields in list EMPLIST all have an ***output** attribute.

3. Define the following global web_maps:

```
Web_Map For(*both) Fields((#stdrentry *hidden))
Web_Map For(*none) Fields(#savlist) Options(*persist)
```

Note that the list SAVLIST is mapped ***none**, meaning it is not mapped out to the web page. It has an `Options()` keyword of ***persist**. With session management enabled, this list will be saved each time the WAM ends, and will be restored before the first WebRoutine is executed. i.e. every time something starts to run in the WAM.

4. Save your changes.

Step 5. Complete WAM RDMLX

1. The **begin** WebRoutine will be used to start session management and transfer to the **select** WebRoutine. Add the following code to the **begin** WebRoutine:

```
#com_owner.sessionstatus := active  
Message Msgtxt('Session is active')  
Transfer Toroutine(select)
```

2. The **select** WebRoutine display the employee number for input. It also supports push button to clear the saved list. Add the following code to the **select** WebRoutine. Ensure that you change **iii** to your initials.

```
Web_Map For(*output) Fields(#empno)  
Case (#stdrentry)  
When (= C)  
Clr_List Named(#savlist)  
Endcase
```

3. The **showpage** WebRoutine displays all employee fields for input. Employee number is an output field. Work field EMPNOW is mapped as a hidden field to store current employee number.

When initially invoked from the **select** WebRoutine, the employee fields are retrieved.

When invoked from the Save push button, an entry is added to the saved list.

Add the following code to the **showpage** WebRoutine. Change **iii** to your initials.

```
Web_Map For(*both) Fields((#empno *output) #surname #givenname #iiidate #  
Case (#stdrentry)  
* Invoked from select WebRoutine  
When (= S)  
#empnow := #empno  
Fetch Fields(#surname #givenname) From_File(pslmst) With_Key(#empno)  
* Save push button  
When (= U)  
#empno := #empnow  
Add_Entry To_List(#savlist)  
Transfer Toroutine(select)
```

Endcase

4. The **showlist** WebRoutine is invoked via a push button on the select web page. This routine clears the list EMPLIST and populates it from the current saved list SAVLIST. Add the following code to the **showlist** WebRoutine.

```
Web_Map For(*output) Fields(#emplist)
Clr_List Named(#emplist)
Selectlist Named(#savlist)
Add_Entry To_List(#emplist)
Endselect
```

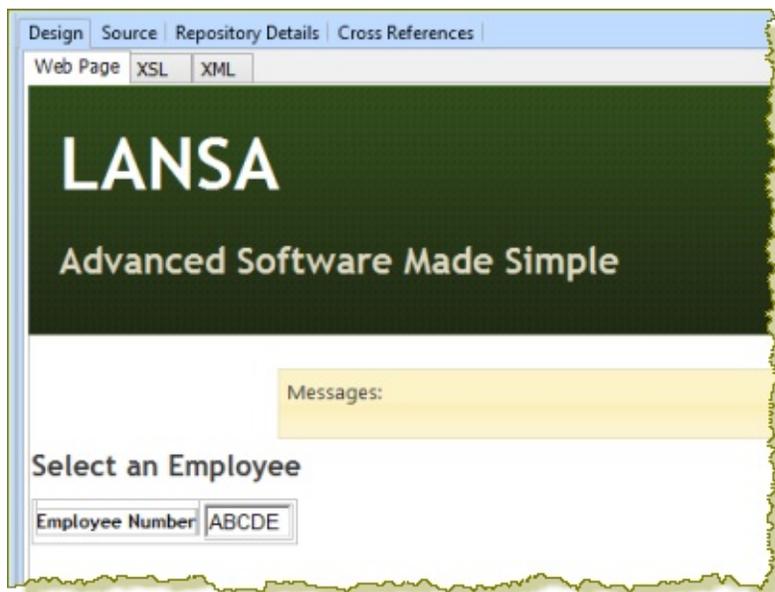
5. Add an event handling routine for invalid session. This will transfer to the **begin** WebRoutine if the WAM is invoked with an invalid or expired session status. The **begin** WebRoutine executes with ***sessionstatus_none**. Add the following event routine:

```
Evtoutine Handling(#com_owner.sessionInvalid)
Message Msgtxt('Session has expired')
Transfer Toroutine(begin)
Endroutine
```

6. Compile the WAM.

Step 6. Design the web pages

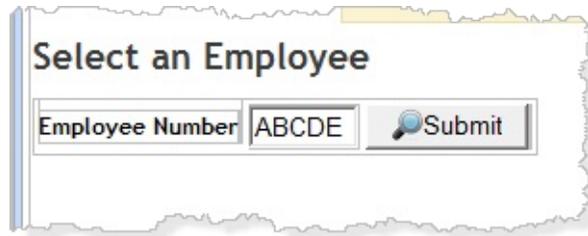
1. Open the **select** WebRoutine in the Design view. It should look like the following:



2. Add a column to the table containing employee number and drop a *Push button with image* weblet into the new cell. Delete the placeholder characters. Set up the push button as follows:

| Property | Value |
|--------------------------|---------------------------------|
| caption | Submit |
| left_relative_image_path | icons/normal/16/zoom.png |
| on_click_wname | Showpage |
| submitExtraFields | Field Name: STDREENTRY |
| | Literal Value: S |

Your page should look like the following:



- Drop an *jQuery UI AutoComplete* weblet onto the Employee Number field and set up its properties as follows:

| Property | Value |
|---------------|-----------------------|
| sourceWamName | iiiEmpAutoCmpl |
| sourceWrName | Empno_Prompt |
| termField | EMPNO |
| listName | EMP_DD |
| valueField | EMPNO |

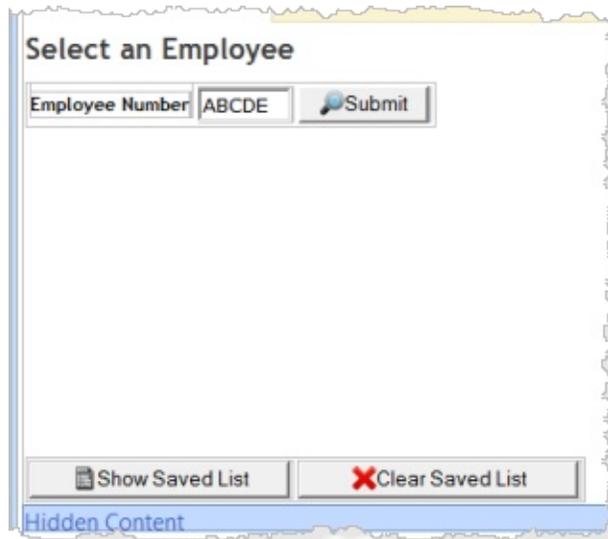
Note that if the WAM **iiiEmpAutoCmpl** is open in the editor, you will be able to select all of these values from a dropdown.

- Use the context menu to Insert HTML / Table with one row and two columns. Use the cursor keys to move to the left of this new table and press *Enter* a number of times to position this table towards the bottom of the central area of the screen.
- Add a push button with images into each cell of the new table. Set up the push buttons as follows:

| Property | Value |
|--------------------------|--|
| Caption | Show Saved List |
| left_relative_image_path | /icons/normal/16/blacklist_16.png |
| on_click_wrname | Showlist |

| Property | Value |
|--------------------------|--------------------------------------|
| Caption | Clear Saved List |
| left_relative_image_path | /icons/normal/16/cross_16.png |
| on_click_wname | Select |
| submitExtraFields | Field Name: STDREENTRY |
| | Literal Value: C |

Your web page should look like the following:



- Save your changes.
- Open the **ShowPage** WebRoutine in the design view. Your web page should look like the following:

Design | Source | Repository Details | Cross References

Web Page | XSL | XML

LANSA

Home | Services

Advanced Software Made Simple

Messages:

Enter employee details

| | |
|------------------------|---|
| Employee Number | ABCDE |
| Employee Surname | aAbBcCdDeEfFgGhHiIj. |
| Employee Given Name(s) | aAbBcCdDeEfFgGhHiIj. |
| Start Date | 01/01/1900 |
| Last Updated | 01/01/1900 01:00:00 |
| Comments | aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZaAb |
| Gender | <input type="radio"/> Male <input type="radio"/> Female |
| Current Employee | <input type="checkbox"/> Caption |
| Create Time | 00:00:00 |
| Marital Status | A |
| On Leave | A |
| Last Review Notes | aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZaAb |
| Terminate Date | 01/01/1900 |

- a. Note that Start Date, Last Updated, Comments, Gender, Current Employee and Create Time are all shown as weblents. For example click on the Start Date field and select the Details tab. Note that this is the std_datepicker weblent.

These weblents are all defined as weblent field visualizations in the *Repository*.

- b. Select **Last Review Notes** and then **Terminate Date** and check their definition using the Details tab. Notice that they have also been defined in the web page as weblents (std_varchar and std_DatePicker respectively) based on their field type. These are fields you defined in the WAM.
- c. The single character (Alphanumeric) fields **Marital Status** and **On Leave** are input fields. In the *Design* view you will replace these with weblents.

8. Select the field **Start Date** and set up its properties as follows:

| Property | Value |
|-------------------|-------------|
| changeMonth | True |
| changeYear | True |
| showOtherMonths | True |
| selectOtherMonths | True |

You will be able to see the effect of these settings once you test the WAM.

Change the *dateFormat* property to be correct for your region. **mm/dd/yyyy** is the default value.

Also change the *dateFormat* for fields **Terminate Date** and **Last Updated** to suit your region.

9. Select the **Comments** field and set up its properties as follows:

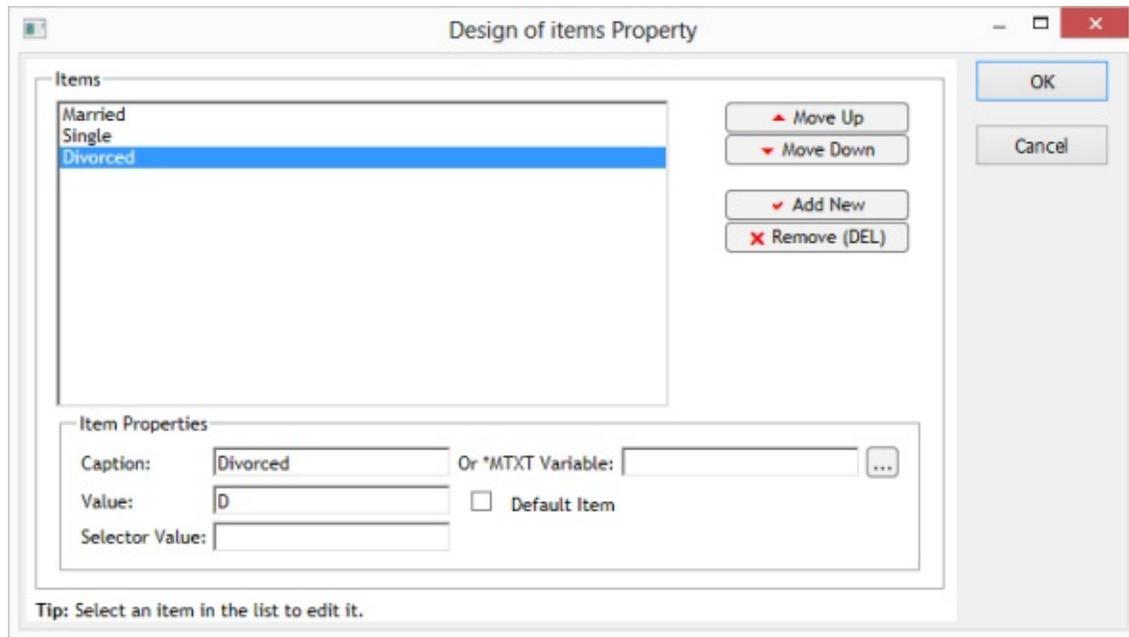
| Property | Value |
|-----------|-------------|
| Type | memo |
| word_wrap | soft |

Note that the **Comments** field is now a memo box.

10. Select the current Employee checkbox and change its caption to ". The caption should now be blank.

Note that the *oncode* and *offcode* properties for the check box weblet are values of **Y** and **N** which are correct for this field. These values could be changed if necessary.

11. Drop a *Radio Group* weblet onto the field **Marital Status**. On the *Details* tab, in the *items* property's value column, use the *Ellipsis* button to open the *Design of...* dialog to define three captions and values as shown:



| Caption | Value |
|----------|----------|
| Married | M |
| Single | S |
| Divorced | D |

12. Drop a *Check Box* weblet onto the **On Leave** field and set its caption to ".
Once again the default *oncode* and *offcode* values are suitable for this field.
13. Select the **Last Review Notes** field (*std_varchar* weblet) and set its properties as above for the Comments field.
14. Select the **Terminate Date** field (*std_DatePicker* weblet) and set its properties as above for Start Date
15. Save your changes.
16. Drop a *Push Button with Image* weblet into the bottom right of the table, next to the Terminate Date field. Set up its properties as follows:

| Property | Value |
|----------|-------------|
| Caption | Save |
| | |

| | |
|---------------------|-----------------------------------|
| left_relative_image | icons/normal/16/check_mark_16.png |
| on_click_wrname | Showpage |
| submitExtraFields | Field Name: STDREENTRY |
| | Literal Value: U |

17. In this step you will add a link to the **Show Saved List** page to return to the Select Employee Number page.

- a. Open the **showlist** WebRoutine in the Design view.
- b. Drop a *Push Button with Images* onto the page, below the list.
- c. Set up the push button properties as follows:

| Property | Value |
|---------------------|--------------------------|
| caption | Select Employee |
| left_relative_image | icons/normal/16/zoom.png |
| on_click_wrname | Select |
| submitExtraFields | Field Name: STDREENTRY |
| | Literal Value: T |

Note: The *dateFormat = Auto* property for the DatePicker weblet, will be correct for most applications. The date format will be determined by the partition's ISO Language setting. For the UK this should be **en-GB** since **en** will give DD/MM/YYYY.

18. Save your changes.

Step 7. Test the WAM

The WAM should be executed by calling the begin WebRoutine. However, note that calling any WebRoutine initially, will transfer control to the begin WebRoutine, via the session invalid event handling routine.

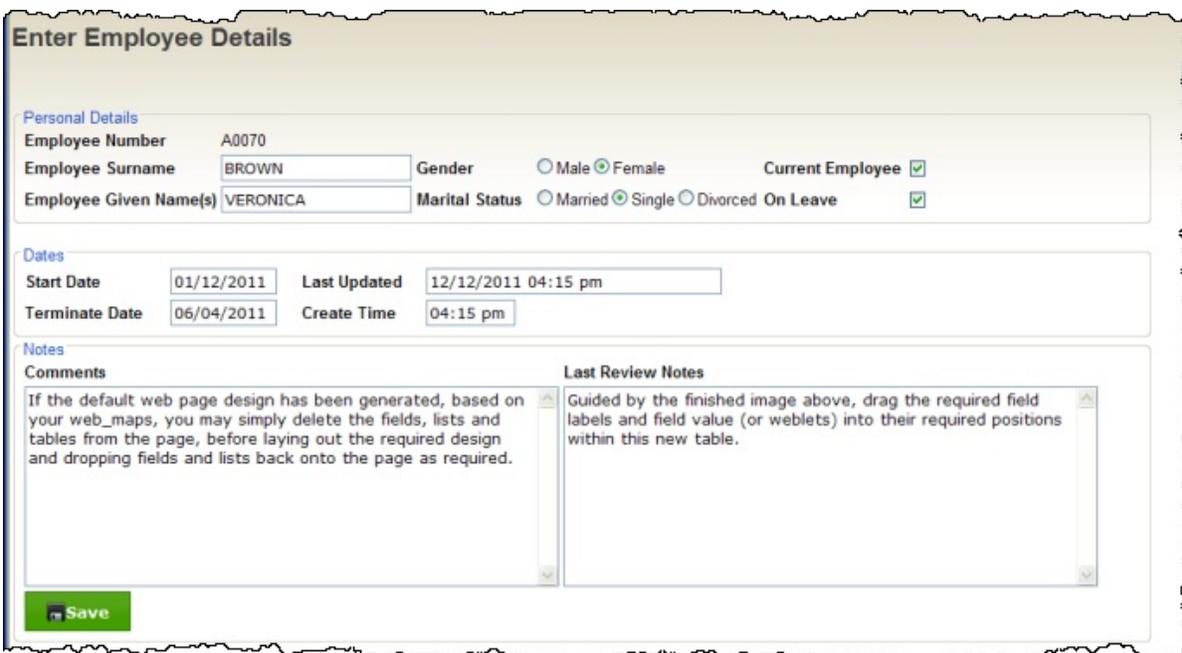
- On the *Select an Employee* page, you can enter a valid employee number by typing a character and selecting from the response in the AutoComplete weblet. Note that all employee numbers begin with 'A' followed by a four digit number. The weblet is set up to invoke the response WebRoutine with a delay of 150ms after you stop typing and after 2 characters have been entered.
- Data can then be entered on the Enter Employee Data web page. Note that initially this page will contain employee number, surname and given name only. You can enter other data via the weblets that support these field values.
- Note that the Start Date DatePicker allows year and month to be selected, within a 10 year range of current date. These limits can be adjusted by changing the weblet properties.
- The Comments std_varchar weblet applies word wrap as you type beyond the width of the control.
- Check box and Radio Group return values according to your selection.
- The Save button will return to the Select an Employee web page.
- Enter data for more than one employee.
- On the *Select an Employee* page, use the *Show Saved List* button to display the current saved list of entries, via the **showlist** WebRoutine.
- On the *Select an Employee* page, use the *Clear Saved List* to clear the saved list and then repeat your testing

Step 8. Improve the ShowPage Page Design

In this step you will change the layout of the *Enter Employee Details* page, by moving the fields into logical groups on the page. You will also insert `<fieldset>` tags around each group of fields.

- The fieldset tag displays a "group box" around an area of the screen. The fieldset tag may include a `<legend>` tag that defines a caption displayed at the top left corner of the box.
- The fieldset tag must enclose the complete table, if a table is being used to lay out a number of fields and labels.
- The fieldset tags may not enclose a number of rows within a table.
- For more information on all HTML and related topics, see www.w3schools.com

The finished page design (showpage) will look like the following:



The screenshot shows a web page titled "Enter Employee Details". It is divided into three main sections:

- Personal Details:** Contains fields for Employee Number (A0070), Employee Surname (BROWN), Employee Given Name(s) (VERONICA), Gender (radio buttons for Male and Female, with Female selected), Marital Status (radio buttons for Married, Single, and Divorced, with Single selected), Current Employee (checkbox checked), and On Leave (checkbox checked).
- Dates:** Contains fields for Start Date (01/12/2011), Last Updated (12/12/2011 04:15 pm), Terminate Date (06/04/2011), and Create Time (04:15 pm).
- Notes:** Contains two text areas: "Comments" and "Last Review Notes". The "Comments" area contains a message about the default web page design. The "Last Review Notes" area contains a message about dragging field labels and values into their required positions.

A green "Save" button is located at the bottom left of the form.

To redesign this web page, follow the details steps provided. However, bear in mind that other approaches could have been followed:

- If the WAM is initially compiled **before** any web_maps are defined, the web page will be blank.
- Recompiling the WAM when all the required web_maps have been defined, will list the field and list mappings on the *WebRoutine Output* tab. The page

may then be defined in the *Design* view, by dragging and dropping fields into their required locations.

- If the default web page design has been generated, based on your web_maps, you may simply delete the fields, lists and tables from the page, before laying out the required design and dropping fields and lists back onto the page as required.
- In this exercise, you will insert new elements into the page (tables) and drag and drop fields and labels on the page in the *Design* view to achieve the result required.
- As you work through this part of the exercise, refer to the image above for guidance on the end result required.

1. Open the **showpage** WebRoutine in the *Design* view.

2. In this step you will create the *Personal Details* area containing employee number, surname, given name, gender, marital status, current employee and on leave fields.

Insert a table at the bottom of the page with 3 rows and 6 columns

3. Guided by the finished image above, drag the required field labels and field value (or weblets) into their required positions within this new table.

Note: You may find an easier approach is to *Cut* (Ctrl+X) each element from the existing table and then *Paste* (CTRL+V) it into the new table.

4. In the new table, select each table cell (that is, the `<td>` tag) containing a *field label* and set its *class* property to **caption**.

5. Remove the place holder characters where necessary.

Your page should now look like the following:

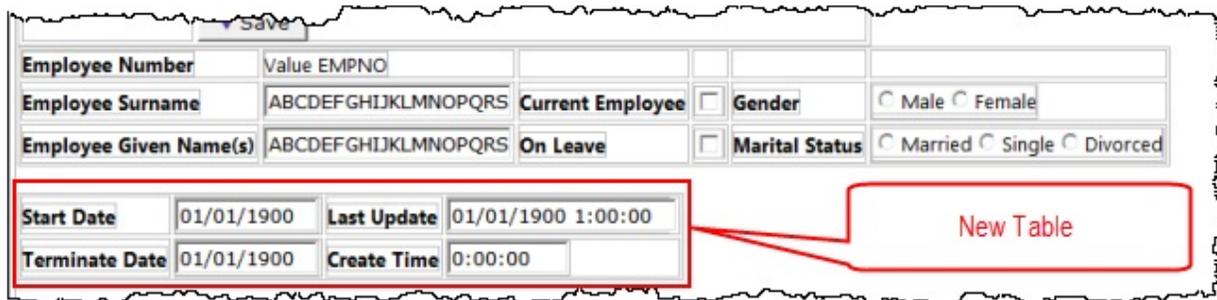


6. Save your changes.

7. Click in the top row of the original table, which is now empty, and use the

context menu to use *Table Items / Delete Rows...* to delete **3** rows. Repeat this step as necessary in the other empty rows.

8. Insert another new table at the bottom of the page, with 2 rows and 4 columns. Drag the date and time fields and labels into the new table, or *Cut* and *Paste* each item.
9. Select each cell containing a label and change its *class* to **caption**.
10. Remove the placeholder characters. Your page should now look like the following:



The screenshot shows a web form with several input fields. A new table is highlighted with a red border and a callout box labeled "New Table". The table has 2 rows and 4 columns. The first row contains "Start Date" (01/01/1900), "Last Update" (01/01/1900 1:00:00), and two empty cells. The second row contains "Terminate Date" (01/01/1900), "Create Time" (0:00:00), and two empty cells. The form also includes fields for "Employee Number", "Employee Surname", "Employee Given Name(s)", "Current Employee", "Gender", "On Leave", and "Marital Status".

| | | | | |
|------------------------|-------------------|------------------|--------------------------|--|
| Employee Number | Value EMPNO | | | |
| Employee Surname | ABCDEFGHIJKLMNOPS | Current Employee | <input type="checkbox"/> | Gender <input type="radio"/> Male <input type="radio"/> Female |
| Employee Given Name(s) | ABCDEFGHIJKLMNOPS | On Leave | <input type="checkbox"/> | Marital Status <input type="radio"/> Married <input type="radio"/> Single <input type="radio"/> Divorced |
| Start Date | 01/01/1900 | Last Update | 01/01/1900 1:00:00 | |
| Terminate Date | 01/01/1900 | Create Time | 0:00:00 | |

11. Once again make the original table smaller by deleting the empty rows. This step is optional, since you will eventually delete this whole table. Deleting rows makes it easier to drag items to the new table.
12. Insert a new table at the bottom of the page with 3 rows and 2 columns. Drag the comment and notes fields and labels into this new table. Review the image in 16. before you complete this step.
13. Select each cell containing a label and change its *class* to **caption**. Remove the place holder characters.
14. Add some ****** characters to the original table, so that it remains easily visible and then drag the Save button into the bottom left hand cell.
15. Drag the Save button into the bottom left cell of the new table.
16. Save your changes. Your page should look like the following:

| | | | | | |
|--|---------------------|------------------|--|----------------|---|
| Employee Number | ABCDE | | | | |
| Employee Surname | aAbBcCdDeEfFgGhHiIj | Current Employee | <input type="checkbox"/> | Gender | <input type="radio"/> Male <input type="radio"/> Female |
| Employee Given Name(s) | aAbBcCdDeEfFgGhHiIj | On Leave | <input type="checkbox"/> | Marital Status | <input type="radio"/> Married <input type="radio"/> Single <input type="radio"/> Divorced |
| Start Date | 01/01/1900 | Terminate Date | 01/01/1900 | | |
| Last Updated | 01/01/1900 01:00:00 | Create Time | 00:00:00 | | |
| Comments | | | Last Review Notes | | |
| aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ
aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ
aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ
aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ | | | aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ
aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ
aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ
aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ | | |
| <input type="checkbox"/> Save | | | | | |

17. Select the corner of the original table, to select the whole table and delete it.
18. Save your changes.

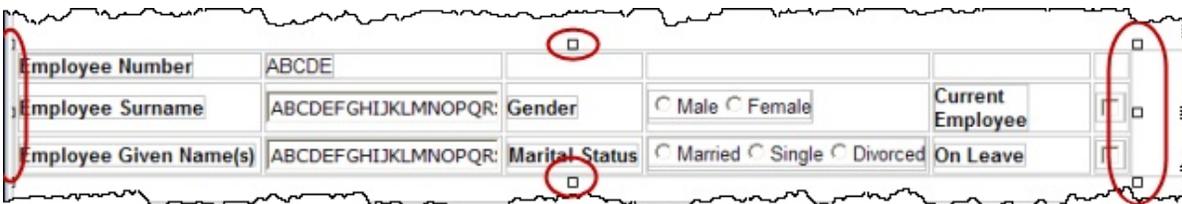
Step 9. Insert a fieldset around each table

In this step you will insert a fieldset around each table. This will display a round cornered group box around each area. You will also add a legend tag in each fieldset to display a caption.

Lastly, in the *Design* view you will insert a div around each fieldset and size it. You are doing this to size the fieldset that will otherwise have the same width as its container. It is possible to size the fieldset by giving it a class and using CSS, but a bug in IE would then display the fieldset with square corners. Adding the div avoids this issue.

1. Select the top table containing employee details. Select a corner of the table. Alternatively, select anywhere in the table and use the *Outline* tab, to select the table itself.

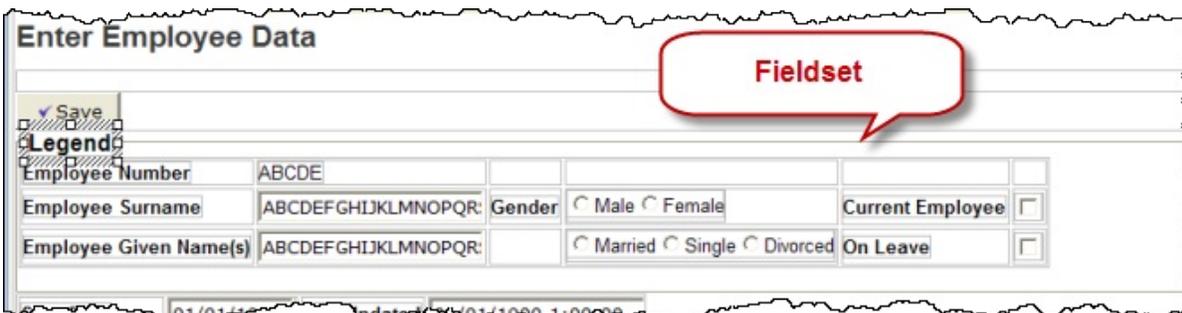
The Design view should now look like the following:



A screenshot of a web form in Design view. The form contains a table with the following fields: Employee Number (text), Employee Surname (text), Employee Given Name(s) (text), Gender (radio buttons for Male and Female), Marital Status (radio buttons for Married, Single, and Divorced), Current Employee (checkbox), and On Leave (checkbox). The table is surrounded by a jagged, torn-paper-like border. Small square handles are visible at the corners of the table, with red circles highlighting the top-right and bottom-left handles.

Note the "handles" around the table that is currently selected.

2. Use the right mouse button over one of these "handles" and select *Insert HTML / Fieldset* to insert a fieldset around this table. Your design should look like the following:



A screenshot of the same web form in Design view. A red speech bubble labeled "Fieldset" points to a red-bordered box that has been inserted around the table. The table fields are the same as in the previous screenshot. Above the table, there is a "Legend" text area with a "Save" button. The form is still surrounded by the jagged, torn-paper-like border.

3. Click in the Legend text area and replace it with **Personal Details..**
4. Select the table containing Start Date and insert a fieldset around this table, with and replace Legend text with **Dates.**
5. Select the table containing Comments and insert a fieldset around this table

with the legend **Comments**.

6. Save your changes.

| Details | | | | | |
|------------------------|---------------------|------------------|--------------------------|----------------|---|
| Employee Number | Value EMPNO | | | | |
| Employee Surname | ABCDEFGHIJKLMNOPQRS | Current Employee | <input type="checkbox"/> | Gender | <input type="radio"/> Male <input type="radio"/> Female |
| Employee Given Name(s) | ABCDEFGHIJKLMNOPQRS | On Leave | <input type="checkbox"/> | Marital Status | <input type="radio"/> Married <input type="radio"/> Single <input type="radio"/> Divorced |

7. Run your application to see the results in the browser. Enter some data. Your page should look like the following:

LANSA
Advanced Software Made Simple

Home Services Contact About

Enter employee details

Personal Details

Employee Number: A0070

Employee Surname: BROWN Current Employee: Gender: Male Female

Employee Given Name(s): VERONICA On Leave: Marital Status: Married Single Divorced

Dates

Start Date: 03/04/2013 Terminate Date: 30/04/2013

Last Updated: 12/04/2013 05:44 pm Create Time: 05:44 pm

Comments

Comments

consequat. Ut magna pulvinar ac at nisl. Mauris augue eu. Odio commodo lacinia habitasse nec urna imperdiet pede sapien in amet vitae porttitor adipiscing arcu sit condimentum sed sed ultrices vestibulum. Aliquam elit sequi molestie nibh per. Eget hac amet libero lorem eget. Amet nec nisl. Amet ut a. Turpis suspendisse enim. Pede tempora ipsum nam a nihil sit ut massa vel arcu suspendisse. Nunc eget orci. Non quia lacus. Consequat mollis consectetur. Tempor dignissim ultricies. Felis interdum dapibus lacus accumsan fermentum.

Last Review Notes

Orci augue vel. Quis eum lectus. Consectetur lacus enim quisque ac tincidunt fringilla semper nonummy. Neque dui lacinia. Aliquet nec eros ut non per. Faucibus ut adipiscing aliquam porttitor purus a itaque eget. Mauris vel pede sem in quo. Purus at leo posuere venenatis donec erat dis mauris nullam ut vestibulum.

Save

Notice that the group boxes span the full width of the web page. The screen shot has minimized the page width. With the browser maximised on a large screen the blank areas on the right look a little unfinished.

The web site <http://lorem-ipsam.perbang.dk/> may be used to generate 'mock latin' text to fill up your memo boxes.

8. With the **showpage** WebRoutine open in the *Design* view, select each fieldset border and use the context menu to *Insert* a **Div**. Check you have selected the fieldset by looking at the *Properties* sheet on the *Details* tab. It should display **<fieldset>** at the top. Use the context menu to *Insert HTML / DIV* around the fieldset.
9. With the Div selected use the *Details* tab, to expand the *Style* properties and change the *width* to **80%**.
10. Repeat the last two steps for the other two fieldsets.
11. Save your changes.
12. Run the application. Your web page should now look like the following:

Enter Employee Data

Details

Employee Number: A0070

Employee Surname: BROWN Current Employee: Gender: Male Female

Employee Given Name(s): VERONICA On Leave: Marital Status: Married Single Divorced

Dates

Start Date: 31/01/2001 Last Update: 25/08/2011 10:53 am

Terminate Date: 10/08/2006 Create Time: 10:53 am

Notes

Comments

In euismod signiferumque nam, graecis assueverit id nec. Viderer dolores has ne, ad eam solum choro. Usu at nibh praesent. Mollis tractatos te vix, ex eos tota aperiam. Ne partem indoctum his, his perfecto iudicabit an. Reque utamur facilis his an. Id omnes soluta dissentiet pro.

Last Review Notes

Lorem ipsum dolor sit amet, quis persius insolens id sed. Simul zril intellegat in duo. Enim blandit per ei, et rebum decore mea, an vix sale munere. Quem saperet honestatis at his. Falli debet necessitatibus sit et, cum wisi elltr no. Occurreret prodesset est id, mea ea oratio audire.

Summary

Important Information

- Only weblet field visualizations that are likely to change **very infrequently**, should be defined in the Repository
- Field types Date, DateTime, Time and String will be given weblet visualizations by default.
- The *Design* view provides a powerful graphical editor, that enables the developer to rapidly refine the page design and layout.
- You will always require some knowledge of HTML and style sheets (CSS) to be able to work with web page design.
- Excellent tutorials for all web languages are available at www.w3schools.com
- If your application is aimed at the consumer, you will probably keep the page design much simpler than this example, possibly spread over two or three pages.

Tips & Techniques

- This exercise has demonstrated some aspects of some of the weblets available. The Web Application Modules guide provides more detailed information for all of the weblets.
- You should be aware that fields supported by weblets should always be defined with a suitable default value. The weblet only returns a value when the user selects a value. e.g. only when the user checks or un-checks a check box, does the weblet return a value.

What You Should Know

- How to make use of the available weblets to enhance the user interface.

WAM090 - Using a List Row Weblet

Objectives

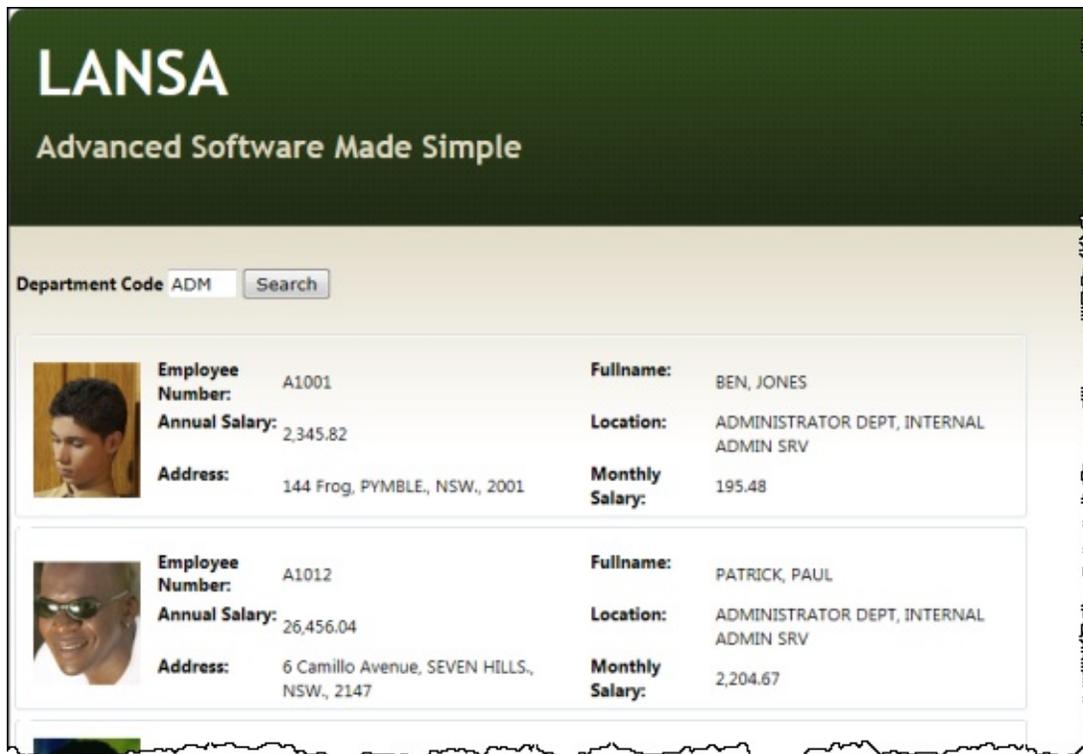
In the exercises which you have completed up to now, lists have always been formatted as a simple table with column headings and data shown in columns. This exercise will demonstrate how you can present a list as a series of formatted rows. Your list output can in fact be formatted almost any way you wish. Achieving the result you want, will depend on your skill with HTML and XSL transformation.

This exercise provides a custom weblet (`iii_ListRow`) for you to use. Although you will not be examining its XSL and HTML, it will demonstrate how list data can be treated in a different way.

The List Row Weblet (`iii_ListRow`)

The main concept to keep in mind in this exercise, is that list output consists of a repeating set of data. The row weblet `iii_ListRow` allows you to associate a working list with the weblet and then assign the fields within this list to parameters for each row. The working list needs to contain:

- The data fields – the values you want to output
- Fields holding the labels (or captions) for each of the output fields.
- A field containing a path and file name for a small employee image.
- Each row is wrapped in a `<fieldset>` tag. The list also contains a field and label field for the `<legend>` tag that provides the caption at the top left corner of the group box that the fieldset provides.
- The WebRoutine should map the working list with a `*private` attribute, since the List Row weblet will format the output.



The WAM will be a simple enquiry that lists employees by department.

To complete this exercise, complete the following:

- [Step 1. Create the List Row Weblet – iii_ListRow](#)Row – List Row Weblet
- [Step 2. Create WAM iii UseListRowWeblet](#)UseListRowWeblet
- [Step 3. Set Up the Web Page](#)
- [Summary](#)

Before You Begin

Complete the preceding exercises in this workshop.

Step 3 of this WAM requires a set of small images for employees. If you haven't already done so, you can download these from the documentation page of the LANSA web site as described in [WAM チュートリアル \(英語\)](#) following the list of tutorials.

Copy the images from the PHOTOS folder into to your web server /images path, such as: C:\Program Files (x86)\LANSA\WebServer\images. If you are running your WAMs on the IBM i, copy this folder to: /LANSA_<PGMLIB>/webserver/images where <PGMLIB> is your LANSA program library.

Step 1. Create the List Row Weblet – iii_ListRow

1. Create a new weblet as shown:



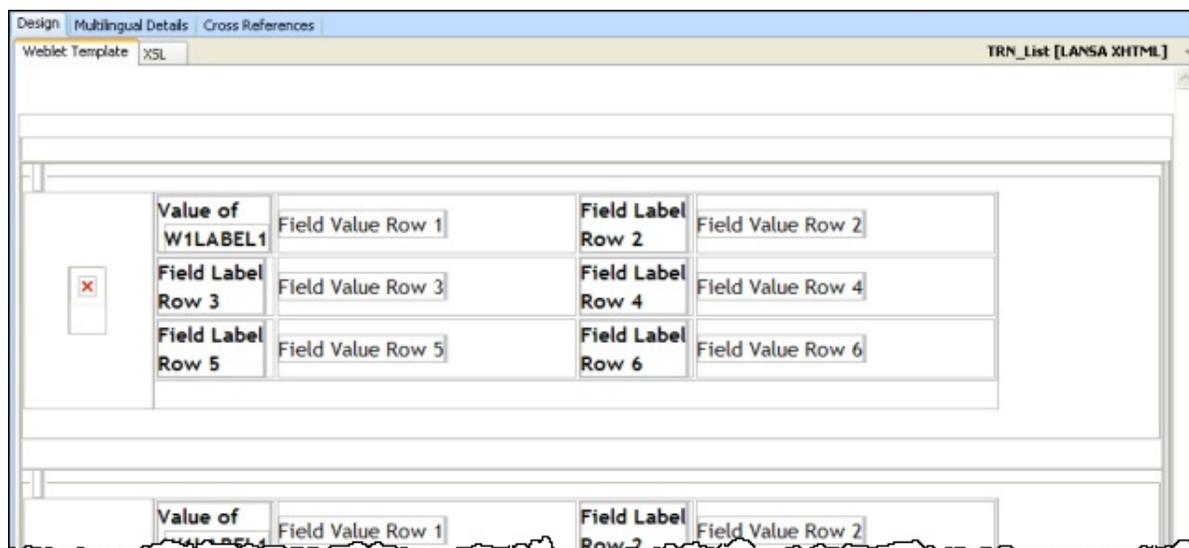
The 'New Weblet' dialog box is shown with the following fields and values:

| | | |
|---------------|--------------------------|--------|
| Name | iii_ListRow | Create |
| Description | Training List Row Weblet | |
| Weblet Group | Custom Weblets | Cancel |
| Layout Weblet | <input type="checkbox"/> | |

Give the weblet a *Weblet Group* of **Custom Weblets**.

If the group *Custom Weblets* does not already exist, type the name into the Weblet Group combo box and a new group will be created.

2. Copy the weblet XSL code from [WAM090. Appendix](#), at the end of this exercise and **replace** the default code in your new weblet.
3. Save the new weblet definition.
4. Select the *XSL* tab and use the *Replace* dialog to replace all occurrences of **iii_ListRow** with the same name using your initials in place of **iii**. There should be 5 occurrences.
5. Save your changes.
6. Select the *Design* view. Your weblet design should look like the following:



A row is defined with an image on the left hand side and data fields and label

fields arranged in three rows with four columns.

Step 2. Create WAM iii UseListRowWeblet

1. Create a new WAM:

Name: **iiiUseListRowWeblet**

Description: **Using a List Row Weblet**

Layout Weblet: **iiilay01**

2. Add the following definitions after the Begin_Com:

```
Define Field(#fulladdr) Reffld(#std_textl)
Define Field(#location) Reffld(#std_textl)
Define Field(#DESC1) Reffld(#std_desc) Default("Name: ")
Define Field(#DESC2) Reffld(#std_desc) Default("Location: ")
Define Field(#DESC3) Reffld(#std_desc) Default("Address: ")
Define Field(#DESC4) Reffld(#std_desc) Default("Start Date: ")
Define Field(#DESC5) Reffld(#std_desc) Default("Monthly Salary: ")
Define Field(#DESC6) Reffld(#std_desc) Default("Annual Salary: ")
Define Field(#DESC7) Reffld(#std_desc) Default("Employee Number: ")
*
Group_By Name(#empdata) Fields(#empno #SURNAME #GIVENAME #AD
Def_List Name(#emplist) Fields(#empno #fullname #fulladdr #location #SAL.
*
Web_Map For(*both) Fields((#stdrentry *hidden))
WebRoutine Name(list)
Web_Map For(*both) Fields(#deptment) /* comment */
Web_Map For(*output) Fields((#emplist *private))
Case (#stdrentry)
When (= S) /* Build list of employees */
Clr_List Named(#emplist)
Select Fields(#empdata) From_File(pslmst1) With_Key(#deptment) Nbr_Keys
Fetch Fields(#deptdesc) From_File(deptab) With_Key(#deptment) Keep_Last(
Fetch Fields(#secdesc) From_File(sectab) With_Key(#deptment #section) Kee
#std_textl := 'PHOTOS/' + #EMPNO + '.jpg'
#fullname := #givenname + ', ' + #surname
#fulladdr := #address1 + ', ' + #address2 + ', ' + #address3 + ', ' + #postcode.ass
#location := #deptdesc + ', ' + #secdesc
Add_Entry To_List(#emplist)
Endselect
Otherwise
```

Message Msgtxt('Enter a department code and Search')

Endcase

Endroutine

3. Review the supplied code.

The only unusual part of this WAM is that it defines a number of work fields used to define the labels required for the fields to be output in each row.

The working list EMPLIST contains the required fields for each employee, selected from the logical file PSLMST1 with a key of DEPARTMENT.

4. Compile the WAM.

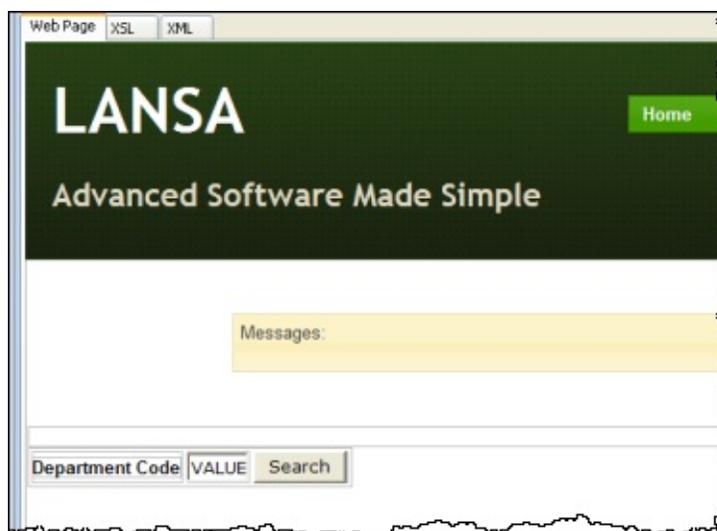
Step 3. Set Up the Web Page

1. Open the **list** WebRoutine in the *Design* view.
2. Add a column to the table containing the department code.
3. Drop a Push button into the new cell and set up its properties:

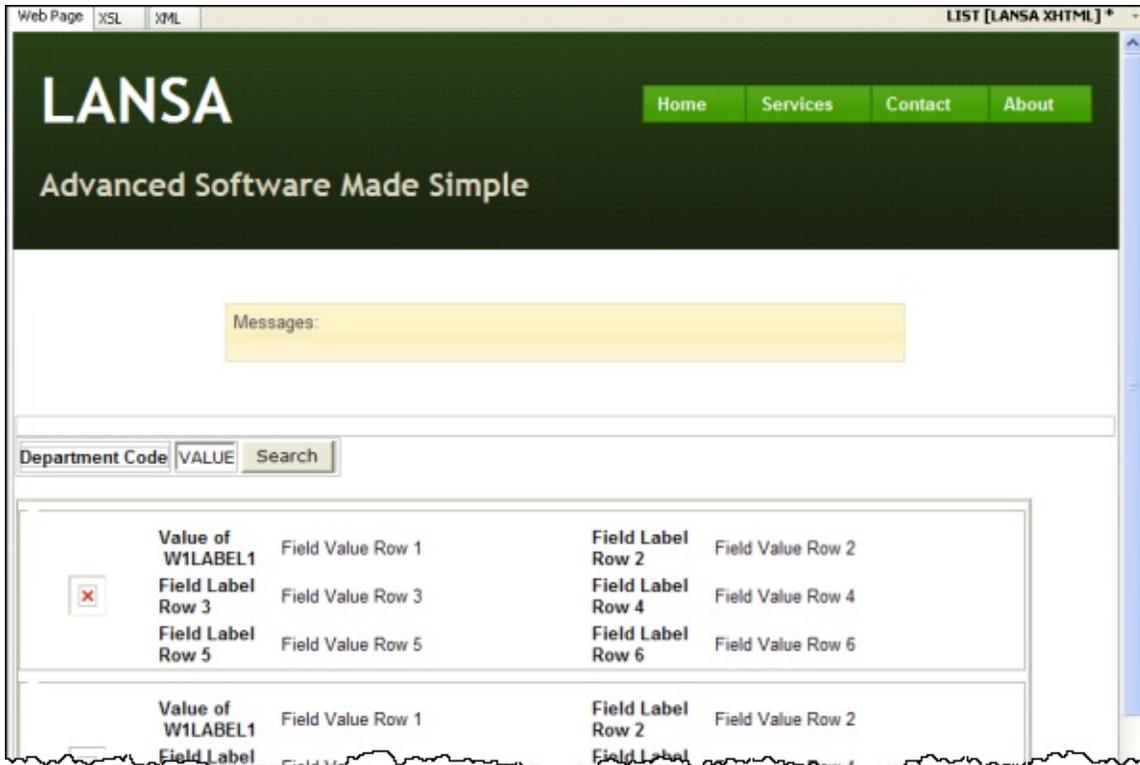
| Property | Value |
|-------------------|-------------------------------|
| Caption | Search |
| On_click_wrname | list |
| submitExtraFields | Field name: STDREENTRY |
| | Literal Value: S |

4. Save your changes.

Your page should look like the following:



5. On the *Favorites / Weblet Templates* tab, select *Custom Weblets* in the top combo box. Drop the *Training List Row Weblet* onto the page. Your page should look like the following:



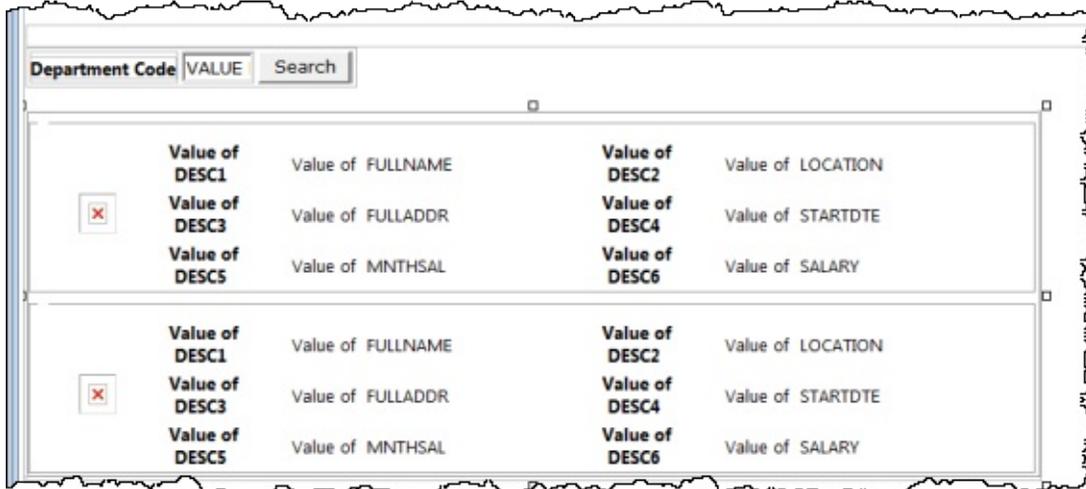
6. Select the List Row Weblet, Select the *Details* tab and set up its properties as follows:

| Property | Value |
|--------------------|-----------------|
| Listname | EMPLIST |
| List_field_label_1 | DESC1 |
| List_field_value_1 | FULLNAME |
| List_field_label_2 | DESC2 |
| List_field_value_2 | LOCATION |
| List_field_label_3 | DESC3 |
| List_field_value_3 | FULLADDR |
| List_field_label_4 | DESC4 |
| List_field_value_4 | STARTDTE |
| List_field_label_5 | DESC5 |
| List_field_value_5 | MNTHSAL |

| | |
|--------------------|------------------|
| List_field_label_6 | DESC6 |
| List_field_value_6 | SALARY |
| List_field_label_7 | DESC7 |
| List_field_value_7 | EMPNO |
| List_image_field | STD_TEXTL |

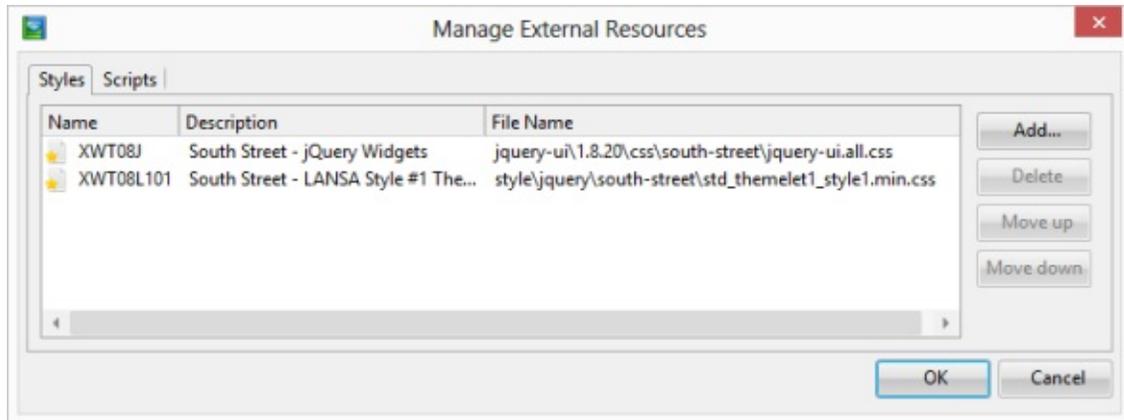
7. Save your changes.

Your design should look like the following:



8. This step will give the list row weblet the correct *External Resources* to match your selected page layout theme. See *WAM060 – Employee Maintenance / Step 4. Define the Details Webroutine* where you used *Cross References* for your layout weblet to find the *External Resources* used by your chosen theme.

With the *Design* view open, select the *Design ribbon / External Resources* button. Knowing your external resources (the cascading style sheets) used by your theme, delete the default files used by the code provided and add the files for your theme.



Save your changes.

9. This step requires a set of small images for employees. If you haven't already done so, you can download these now following the instructions in *Before You Begin* at the beginning of this exercise.
10. Test your WAM and list employees for department code ADM. At the moment your list will be displayed on the left hand side of the page.
11. In the *Design* view select the List Row weblet and use the context menu to *Insert / Div*. With the div selected, change its *align* property to **center**. Save your changes.
12. Test your WAM. The list will now be displayed in the center of the web page.

Summary

Important Observations

- Lists may be presented in any format by making use of the ability to create your own weblets. This will require good HTML knowledge and a working knowledge of XSLT.
- With some modifications the example given here could be used as part of a shopping application.
- Later exercises will cover creating your own simple weblet.
- Once you have completed *WAM105 – Create Your Own Weblet* and *WAM110 – Create Your Own Layout*, you should review the List Row weblet provided for this exercise. Although it is reasonably complex, once you understand the structure of weblets, you should be able to follow its logic by examining its different sections.

Tips & Techniques

- Lists can also be formatting as a single row. This is useful way to present a variable number of options

What You Should Know

- Custom weblets provide a highly flexible way to enhance the user interface of your web applications. Although the investment in creating and testing a custom weblet may be quite high, as a reusable component, it will pay off many times and simplify work for other developers.

WAM090. Appendix

List Row Weblet iii_ListRow (XSL)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- LANSA for the Web -->
<xsl:transform version="1.0" exclude-result-prefixes="lxml wd"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:lxml="http://www.lansa.com/2002/XML/Runtime-Data"
xmlns:wd="http://www.lansa.com/2002/XSL/Weblet-Design"
xmlns="http://www.w3.org/1999/xhtml">
<xsl:import href="std_types.xsl" />
<xsl:import href="std_keys.xsl" />
<xsl:import href="std_anchor.xsl" />
<xsl:output method="xml" omit-xml-declaration="yes" encoding="UTF-8"
indent="no" />
<wd:external-resources>
<wd:style name="XWT08J" />
<wd:style name="XWT08L101" />
</wd:external-resources>
<wd:definition>
<wd:group name="Custom Weblets" />
</wd:definition>
<lxml:data>
<lxml:list name="">
<lxml:list-header>
<lxml:header name="W1LABEL1">
<lxml:heading-1>List Field Label 1</lxml:heading-1>
<lxml:heading-2 />
<lxml:heading-3 />
</lxml:header>
<lxml:header name="W1FIELD1">
<lxml:heading-1>List Field 1</lxml:heading-1>
<lxml:heading-2 />
<lxml:heading-3 />
</lxml:header>
<lxml:header name="W1LABEL2">
<lxml:heading-1>List Field Label 2</lxml:heading-1>
<lxml:heading-2 />
```

```
<lxml:heading-3 />
</lxml:header>
<lxml:header name="W1FIELD2">
<lxml:heading-1>List Field 2</lxml:heading-1>
<lxml:heading-2 />
<lxml:heading-3 />
</lxml:header>
<lxml:header name="W1LABEL3">
<lxml:heading-1>List Field Label 3</lxml:heading-1>
<lxml:heading-2 />
<lxml:heading-3 />
</lxml:header>
<lxml:header name="W1FIELD3">
<lxml:heading-1>List Field 3</lxml:heading-1>
<lxml:heading-2 />
<lxml:heading-3 />
</lxml:header>
<lxml:header name="W1LABEL4">
<lxml:heading-1>List Field Label 4</lxml:heading-1>
<lxml:heading-2 />
<lxml:heading-3 />
</lxml:header>
<lxml:header name="W1FIELD4">
<lxml:heading-1>List Field 4</lxml:heading-1>
<lxml:heading-2 />
<lxml:heading-3 />
</lxml:header>
</lxml:list-header>
<lxml:list-entries>
<lxml:entry>
<lxml:column name="W1LABEL1"
id="SAMPLE_LIST.0001.W1LABEL1">Field 1 Label Col 2</lxml:column>
<lxml:column name="W1FIELD1"
id="SAMPLE_LIST.0001.W1FIELD1">Field 1 Value Col 2</lxml:column>
<lxml:column name="W1LABEL2"
id="SAMPLE_LIST.0001.W1LABEL2">Field 2 Label Col 3</lxml:column>
<lxml:column name="W1FIELD2"
id="SAMPLE_LIST.0001.W1FIELD2">Field 2 Value Col 3</lxml:column>
<lxml:column name="W1LABEL3"
```

id="SAMPLE_LIST.0001.W1LABEL3">Field 3 Label Col 2</lxml:column>
<lxml:column name="W1FIELD3"
id="SAMPLE_LIST.0001.W1FIELD3">Field 3 Value Col 2</lxml:column>
<lxml:column name="W1LABEL4"
id="SAMPLE_LIST.0001.W1LABEL4">Field 4 Label Col 3</lxml:column>
<lxml:column name="W1FIELD4"
id="SAMPLE_LIST.0001.W1FIELD4">Field 4 Value Col 3</lxml:column>
<lxml:column name="W1LABEL5"
id="SAMPLE_LIST.0001.W1LABEL3">Field 5 Label Col 2</lxml:column>
<lxml:column name="W1FIELD5"
id="SAMPLE_LIST.0001.W1FIELD3">Field 5 Value Col 2</lxml:column>
<lxml:column name="W1LABEL6"
id="SAMPLE_LIST.0001.W1LABEL4">Field 6 Label Col 3</lxml:column>
<lxml:column name="W1FIELD6"
id="SAMPLE_LIST.0001.W1FIELD4">Field 6 Value Col 3</lxml:column>
<lxml:column name="W1IMAGE"
id="SAMPLE_LIST.0001.W1IMAGE">IMAGE_FILE</lxml:column>
<lxml:column name="W1LABEL7"
id="SAMPLE_LIST.0002.W1LABEL7">Field 7 Label
Legend</lxml:column>
<lxml:column name="W1FIELD7"
id="SAMPLE_LIST.0002.W1FIELD7">Field 7 Value
Legend</lxml:column>
</lxml:entry>
<lxml:entry>
<lxml:column name="W1LABEL1"
id="SAMPLE_LIST.0001.W1LABEL1">Field 1 Label Col 2</lxml:column>
<lxml:column name="W1FIELD1"
id="SAMPLE_LIST.0001.W1FIELD1">Field 1 Value Col 2</lxml:column>
<lxml:column name="W1LABEL2"
id="SAMPLE_LIST.0001.W1LABEL2">Field 2 Label Col 3</lxml:column>
<lxml:column name="W1FIELD2"
id="SAMPLE_LIST.0001.W1FIELD2">Field 2 Value Col 3</lxml:column>
<lxml:column name="W1LABEL3"
id="SAMPLE_LIST.0001.W1LABEL3">Field 3 Label Col 2</lxml:column>
<lxml:column name="W1FIELD3"
id="SAMPLE_LIST.0001.W1FIELD3">Field 3 Value Col 2</lxml:column>
<lxml:column name="W1LABEL4"
id="SAMPLE_LIST.0001.W1LABEL4">Field 4 Label Col 3</lxml:column>

```
<!xml:column name="W1FIELD4"
id="SAMPLE_LIST.0001.W1FIELD4">Field 4 Value Col 3</!xml:column>
<!xml:column name="W1LABEL5"
id="SAMPLE_LIST.0001.W1LABEL3">Field 5 Label Col 2</!xml:column>
<!xml:column name="W1FIELD5"
id="SAMPLE_LIST.0001.W1FIELD3">Field 5 Value Col 2</!xml:column>
<!xml:column name="W1LABEL6"
id="SAMPLE_LIST.0001.W1LABEL4">Field 6 Label Col 3</!xml:column>
<!xml:column name="W1FIELD6"
id="SAMPLE_LIST.0001.W1FIELD4">Field 6 Value Col 3</!xml:column>
<!xml:column name="W1IMAGE"
id="SAMPLE_LIST.0001.W1IMAGE">IMAGE_FILE</!xml:column>
<!xml:column name="W1LABEL7"
id="SAMPLE_LIST.0002.W1LABEL7">Field 7 Label
Legend</!xml:column>
<!xml:column name="W1FIELD7"
id="SAMPLE_LIST.0002.W1FIELD7">Field 7 Value
Legend</!xml:column>
<!-- <!xml:column name="W1LABEL8"
id="SAMPLE_LIST.0002.W1LABEL8">Field Label Row 8</!xml:column> -
->
</!xml:entry>
</!xml:list-entries>
</!xml:list>
</!xml:data>
<wd:template name="iii_ListRow">
<wd:description icon="icons/std_grid.ico">
<wd:name lang="ENG">Training List Row Weblet</wd:name>
</wd:description>
<wd:param name="TRN List">
<wd:tip lang="ENG">The name of the weblet.</wd:tip>
</wd:param>
<wd:param name="listname">
<wd:tip lang="ENG">The name of the list to use to populate the cells in the
grid.</wd:tip>
</wd:param>
<wd:param name="List_field_label_1">
<wd:tip lang="ENG">The name of the field in the list to use as a label in list
entry, Row 1, Col 2.</wd:tip>
```

</wd:param>
<wd:param name="List_field_value_1">
<wd:tip lang="ENG">The name of the list field to show in list entry, Row 1, Col 2.</wd:tip>
</wd:param>
<wd:param name="List_field_label_2">
<wd:tip lang="ENG">The name of the field in the list to use as a label in list entry, Row 1, Col 2.</wd:tip>
</wd:param>
<wd:param name="List_field_value_2">
<wd:tip lang="ENG">The name of the list field to show in list entry, Row 1, Col 2.</wd:tip>
</wd:param>
<wd:param name="List_field_label_3">
<wd:tip lang="ENG">The name of the field in the list to use as a label in list entry, Row 2, Col 2.</wd:tip>
</wd:param>
<wd:param name="List_field_value_3">
<wd:tip lang="ENG">The name of the list field to show in list entry, Row 2, Col 2.</wd:tip>
</wd:param>
<wd:param name="List_field_label_4">
<wd:tip lang="ENG">The name of the field in the list to use as a label in list entry, Row 2, Col 3.</wd:tip>
</wd:param>
<wd:param name="List_field_value_4">
<wd:tip lang="ENG">The name of the list field to show in list entry, Row 2, Col 3.</wd:tip>
</wd:param>
<wd:param name="List_field_label_5">
<wd:tip lang="ENG">The name of the field in the list to use as a label in list entry, Row 3, Col 2.</wd:tip>
</wd:param>
<wd:param name="List_field_value_5">
<wd:tip lang="ENG">The name of the list field to show in list entry, Row 3, Col 2.</wd:tip>
</wd:param>
<wd:param name="List_field_label_6">
<wd:tip lang="ENG">The name of the field in the list to use as a label in list

```
entry, Row 3, Col 3.</wd:tip>
</wd:param>
<wd:param name="List_field_value_6">
<wd:tip lang="ENG">The name of the list field to show in list entry, Row 3,
Col 3.</wd:tip>
</wd:param>
<wd:param name="List_field_label_7">
<wd:tip lang="ENG">The name of the field in the list to use as a label in the
fieldset Legend.</wd:tip>
</wd:param>
<wd:param name="List_field_value_7">
<wd:tip lang="ENG">The name of the list field to show in t.</wd:tip>
</wd:param>
<wd:param name="even_row_class">
<wd:tip lang="ENG">The Cascading Stylesheet class for even row entries in
the list.</wd:tip>
</wd:param>
<wd:param name="odd_row_class">
<wd:tip lang="ENG">The Cascading Stylesheet class for odd row entries in
the list.</wd:tip>
</wd:param>
</wd:template>
```

```
<xsl:template name="iii_ListRow">
<xsl:param name="listname" wd:type="std:list_name_out" />
<xsl:param name="List_field_label_1" select=""W1LABEL1""
wd:type="std:list_field_name[list=$listname]" />
<xsl:param name="List_field_value_1" select=""W1FIELD1""
wd:type="std:list_field_name[list=$listname]" />
<xsl:param name="List_field_label_2" select=""W1LABEL2""
wd:type="std:list_field_name[list=$listname]" />
<xsl:param name="List_field_value_2" select=""W1FIELD2""
wd:type="std:list_field_name[list=$listname]" />
<xsl:param name="List_field_label_3" select=""W1LABEL3""
wd:type="std:list_field_name[list=$listname]" />
<xsl:param name="List_field_value_3" select=""W1FIELD3""
wd:type="std:list_field_name[list=$listname]" />
<xsl:param name="List_field_label_4" select=""W1LABEL4""
wd:type="std:list_field_name[list=$listname]" />
```

```

<xsl:param name="List_field_value_4" select=""W1FIELD4""
wd:type="std:list_field_name[list=$listname]" />
<xsl:param name="List_field_label_5" select=""W1LABEL5""
wd:type="std:list_field_name[list=$listname]" />
<xsl:param name="List_field_value_5" select=""W1FIELD5""
wd:type="std:list_field_name[list=$listname]" />
<xsl:param name="List_field_label_6" select=""W1LABEL6""
wd:type="std:list_field_name[list=$listname]" />
<xsl:param name="List_field_value_6" select=""W1FIELD6""
wd:type="std:list_field_name[list=$listname]" />
<xsl:param name="List_field_label_7" select=""W1LABEL7""
wd:type="std:list_field_name[list=$listname]" />
<xsl:param name="List_field_value_7" select=""W1FIELD7""
wd:type="std:list_field_name[list=$listname]" />
<xsl:param name="List_image_field" select=""W1IMAGE""
wd:type="std:list_field_name[list=$listname]" />
<xsl:param name="even_row_class" select=""even_row""
wd:type="std:css_style_class[tagName='table']" />
<xsl:param name="odd_row_class" select=""odd_row""
wd:type="std:css_style_class[tagName='table']" />
<xsl:param name="hide_if" select=""false()"" wd:type="std:boolean"
wd:tip_id="" />
<input type=""hidden"" name=""RESULTS.."
value=""{ count(/lxml:data/lxml:lists/lxml:list[@name=$listname]/lxml:list-
entries/lxml:entry[1])}"" />
<xsl:if test=""$listname != "" and not($lweb_design_mode)"">
<table border=""0"" cellspacing=""0"" cellpadding=""0"">
<tr>
<td>
<xsl:for-each
select=""/lxml:data/lxml:lists/lxml:list[@name=$listname]/lxml:list-
entries/lxml:entry">
<xsl:call-template name=""iii_ListRow.private"">
<xsl:with-param name=""listrow"" select=""." />
<xsl:with-param name=""listname"" select=""$listname"" />
<xsl:with-param name=""List_field_label_1""
select=""$List_field_label_1"" />
<xsl:with-param name=""List_field_value_1""
select=""$List_field_value_1"" />

```

```
<xsl:with-param name="List_field_label_2"
select="$List_field_label_2" />
<xsl:with-param name="List_field_value_2"
select="$List_field_value_2" />
<xsl:with-param name="List_field_label_3"
select="$List_field_label_3" />
<xsl:with-param name="List_field_value_3"
select="$List_field_value_3" />
<xsl:with-param name="List_field_label_4"
select="$List_field_label_4" />
<xsl:with-param name="List_field_value_4"
select="$List_field_value_4" />
<xsl:with-param name="List_field_label_5"
select="$List_field_label_5" />
<xsl:with-param name="List_field_value_5"
select="$List_field_value_5" />
<xsl:with-param name="List_field_label_6"
select="$List_field_label_6" />
<xsl:with-param name="List_field_value_6"
select="$List_field_value_6" />
<xsl:with-param name="List_field_label_7"
select="$List_field_label_7" />
<xsl:with-param name="List_field_value_7"
select="$List_field_value_7" />
<xsl:with-param name="List_image_field"
select="$List_image_field" />
<xsl:with-param name="even_row_class"
select="$even_row_class" />
<xsl:with-param name="odd_row_class"
select="$odd_row_class" />
<xsl:with-param name="hide_if" select="$hide_if" />
</xsl:call-template>
</xsl:for-each>
</td>
</tr>
</table>
</xsl:if>
<!-- =====
>
```

```
<xsl:if test="not($hide_if) or $lweb_design_mode">
<xsl:if test="$lweb_design_or_preview">
<xsl:call-template name="std_script_reference.private" />
<xsl:call-template name="std_style_reference.private">
<xsl:with-param name="caller_name" select="'std_button.xml'" />
</xsl:call-template>
</xsl:if>
<!-- ===== -->
<xsl:if test="$listname = " or $lweb_design_mode">
<table cellspacing="0" cellpadding="0" border="1">
<tbody>
<tr>
<td>
<xsl:for-each select="document(')/*//lxml:data/lxml:list/lxml:list-
entries/lxml:entry">
<xsl:call-template name="iii_ListRow.private">
<xsl:with-param name="listrow" select="." />
<xsl:with-param name="listname" select="$listname" />
<xsl:with-param name="List_field_label_1"
select="$List_field_label_1" />
<xsl:with-param name="List_field_value_1"
select="$List_field_value_1" />
<xsl:with-param name="List_field_label_2"
select="$List_field_label_2" />
<xsl:with-param name="List_field_value_2"
select="$List_field_value_2" />
<xsl:with-param name="List_field_label_3"
select="$List_field_label_3" />
<xsl:with-param name="List_field_value_3"
select="$List_field_value_3" />
<xsl:with-param name="List_field_label_4"
select="$List_field_label_4" />
<xsl:with-param name="List_field_value_4"
select="$List_field_value_4" />
<xsl:with-param name="List_field_label_5"
select="$List_field_label_5" />
<xsl:with-param name="List_field_value_5"
select="$List_field_value_5" />
<xsl:with-param name="List_field_label_6"
```

```

select="$List_field_label_6" />
<xsl:with-param name="List_field_value_6"
select="$List_field_value_6" />
<xsl:with-param name="List_image_field"
select="$List_image_field" />
<xsl:with-param name="even_row_class"
select="$even_row_class" />
<xsl:with-param name="odd_row_class"
select="$odd_row_class" />
<xsl:with-param name="hide_if" select="$hide_if" />
</xsl:call-template>
</xsl:for-each>
</td>
</tr>
</tbody>
</table>
</xsl:if>
</xsl:if>
</xsl:template>

```

```

<xsl:template name="iii_ListRow.private">
<xsl:param name="listrow" select="." />
<xsl:param name="listname" />
<xsl:param name="List_field_label_1" />
<xsl:param name="List_field_value_1" />
<xsl:param name="List_field_label_2" />
<xsl:param name="List_field_value_2" />
<xsl:param name="List_field_label_3" />
<xsl:param name="List_field_value_3" />
<xsl:param name="List_field_label_4" />
<xsl:param name="List_field_value_4" />
<xsl:param name="List_field_label_5" />
<xsl:param name="List_field_value_5" />
<xsl:param name="List_field_label_6" />
<xsl:param name="List_field_value_6" />
<xsl:param name="List_field_label_7" />
<xsl:param name="List_field_value_7" />
<xsl:param name="List_image_field" />
<xsl:param name="even_row_class" />

```

```

<xsl:param name="odd_row_class" />
<!-- xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx -->
<fieldset>
<legend class="caption">
<b>
<xsl:value-of select="lxml:column[@name=$List_field_label_7]" />&#160;
</b>
<xsl:value-of select="lxml:column[@name=$List_field_value_7]" />
</legend>
<table cellpadding="0" cellspacing="0" width="750" border="0"
__evenrc="{ $seven_row_class}" __oddrcc="{ $odd_row_class}">
<xsl:attribute name="class">
<xsl:choose>
<xsl:when test="position() mod 2 = 0">
<xsl:value-of select="$seven_row_class" />
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="$odd_row_class" />
</xsl:otherwise>
</xsl:choose>
</xsl:attribute>
<tbody>
<tr>
<td style="height: 20px;" valign="middle" width="100"
align="center">
<a class="thumbnail">
<span>

<br />
</span>
</a>
</td>
<td>
<table>
<tbody>
<tr>
<td class="caption" valign="middle" width="10%">
<xsl:choose>
<xsl:when test="$lweb_design_mode">

```

```

<xsl:choose>
<xsl:when test="$List_field_value_1 = 'W1LABEL1'">
<xsl:value-of select="lxml:column[@name=$List_field_label_1]" />
</xsl:when>
<xsl:otherwise>Value of &#160;<xsl:value-of select="$List_field_label_1" />
</xsl:otherwise>
</xsl:choose>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="lxml:column[@name=$List_field_label_1]" />
</xsl:otherwise>
</xsl:choose>
</td>
<td valign="middle" width="25%" align="left">
<xsl:choose>
<xsl:when test="$lweb_design_mode">
<xsl:choose>
<xsl:when test="$List_field_value_1 = 'W1FIELD1'">
<xsl:value-of select="lxml:column[@name=$List_field_value_1]" />
</xsl:when>
<xsl:otherwise>Value of &#160;<xsl:value-of select="$List_field_value_1"
/>
</xsl:otherwise>
</xsl:choose>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="lxml:column[@name=$List_field_value_1]" />
</xsl:otherwise>
</xsl:choose>
</td>
<td class="caption" valign="middle" width="10%">
<xsl:choose>
<xsl:when test="$lweb_design_mode">
<xsl:choose>
<xsl:when test="$List_field_label_2 = 'W1LABEL2'">
<xsl:value-of select="lxml:column[@name=$List_field_label_2]" />
</xsl:when>
<xsl:otherwise>Value of &#160;<xsl:value-of select="$List_field_label_2" />
</xsl:otherwise>

```

```

</xsl:choose>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="lxml:column[@name=$List_field_label_2]" />
</xsl:otherwise>
</xsl:choose>
</td>
<td valign="middle" width="25%" align="left">
<xsl:choose>
<xsl:when test="$lweb_design_mode">
<xsl:choose>
<xsl:when test="$List_field_value_2 = 'W1FIELD2'">
<xsl:value-of select="lxml:column[@name=$List_field_value_2]" />
</xsl:when>
<xsl:otherwise>Value of &#160;<xsl:value-of select="$List_field_value_2"
/>
</xsl:otherwise>
</xsl:choose>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="lxml:column[@name=$List_field_value_2]" />
</xsl:otherwise>
</xsl:choose>
</td>
</tr>
<tr>
<td class="caption" valign="middle" width="10%">
<xsl:choose>
<xsl:when test="$lweb_design_mode">
<xsl:choose>
<xsl:when test="$List_field_label_3 = 'W1LABEL3'">
<xsl:value-of select="lxml:column[@name=$List_field_label_3]" />
</xsl:when>
<xsl:otherwise>Value of &#160;<xsl:value-of select="$List_field_label_3" />
</xsl:otherwise>
</xsl:choose>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="lxml:column[@name=$List_field_label_3]" />

```

```

</xsl:otherwise>
</xsl:choose>
</td>
<td valign="middle" width="25%" align="left">
<xsl:choose>
<xsl:when test="$lweb_design_mode">
<xsl:choose>
<xsl:when test="$List_field_value_3 = 'W1FIELD3'">
<xsl:value-of select="lxml:column[@name=$List_field_value_3]" />
</xsl:when>
<xsl:otherwise>Value of &#160;<xsl:value-of select="$List_field_value_3"
/>
</xsl:otherwise>
</xsl:choose>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="lxml:column[@name=$List_field_value_3]" />
</xsl:otherwise>
</xsl:choose>
</td>
<td class="caption" valign="middle" width="10%">
<xsl:choose>
<xsl:when test="$lweb_design_mode">
<xsl:choose>
<xsl:when test="$List_field_label_4 = 'W1LABEL4'">
<xsl:value-of select="lxml:column[@name=$List_field_label_4]" />
</xsl:when>
<xsl:otherwise>Value of &#160;<xsl:value-of select="$List_field_label_4" />
</xsl:otherwise>
</xsl:choose>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="lxml:column[@name=$List_field_label_4]" />
</xsl:otherwise>
</xsl:choose>
</td>
<td valign="middle" width="25%" align="left">
<xsl:choose>
<xsl:when test="$lweb_design_mode">

```

```

<xsl:choose>
<xsl:when test="$List_field_value_4 = 'W1FIELD4'">
<xsl:value-of select="lxml:column[@name=$List_field_value_4]" />
</xsl:when>
<xsl:otherwise>Value of &#160;<xsl:value-of select="$List_field_value_4"
/>
</xsl:otherwise>
</xsl:choose>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="lxml:column[@name=$List_field_value_4]" />
</xsl:otherwise>
</xsl:choose>
</td>
</tr>
<tr>
<td class="caption" valign="middle" width="10%">
<xsl:choose>
<xsl:when test="$lweb_design_mode">
<xsl:choose>
<xsl:when test="$List_field_label_5 = 'W1LABEL5'">
<xsl:value-of select="lxml:column[@name=$List_field_label_5]" />
</xsl:when>
<xsl:otherwise>Value of &#160;<xsl:value-of select="$List_field_label_5" />
</xsl:otherwise>
</xsl:choose>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="lxml:column[@name=$List_field_label_5]" />
</xsl:otherwise>
</xsl:choose>
</td>
<td valign="middle" width="25%" align="left">
<xsl:choose>
<xsl:when test="$lweb_design_mode">
<xsl:choose>
<xsl:when test="$List_field_value_5 = 'W1FIELD5'">
<xsl:value-of select="lxml:column[@name=$List_field_value_5]" />
</xsl:when>

```

```

<xsl:otherwise>Value of &#160;<xsl:value-of select="$List_field_value_5"
/>
</xsl:otherwise>
</xsl:choose>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="lxml:column[@name=$List_field_value_5]" />
</xsl:otherwise>
</xsl:choose>
</td>
<td class="caption" valign="middle" width="10%">
<xsl:choose>
<xsl:when test="$lweb_design_mode">
<xsl:choose>
<xsl:when test="$List_field_label_6 = 'W1LABEL6'">
<xsl:value-of select="lxml:column[@name=$List_field_label_6]" />
</xsl:when>
<xsl:otherwise>Value of &#160;<xsl:value-of select="$List_field_label_6" />
</xsl:otherwise>
</xsl:choose>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="lxml:column[@name=$List_field_label_6]" />
</xsl:otherwise>
</xsl:choose>
</td>
<td valign="middle" width="25%" align="left">
<xsl:choose>
<xsl:when test="$lweb_design_mode">
<xsl:choose>
<xsl:when test="$List_field_value_6 = 'W1FIELD6'">
<xsl:value-of select="lxml:column[@name=$List_field_value_6]" />
</xsl:when>
<xsl:otherwise>Value of &#160;<xsl:value-of select="$List_field_value_6"
/>
</xsl:otherwise>
</xsl:choose>
</xsl:when>
<xsl:otherwise>

```

```
<xsl:value-of select="lxml:column[@name=$List_field_value_6]" />
</xsl:otherwise>
</xsl:choose>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
</tbody>
</table>
</fieldset>
</xsl:template>
</xsl:transform>
```

WAM095 - LOB Data Types and Stream Files

Objectives

- To demonstrate how to develop a WAM with a WebRoutine, that uses the Response() parameter to output a file to the browser.

In order to complete these objectives, you will follow one of two paths:

1. If you are **using the VLF application and DXDOCS** file for employee documents you should do the following:
 - [Step 1. Install Required Documents](#)
 - [Step 2. Set up Documents for an Employee](#)
 - [Step 3. Create WAM to Display Employee Documents](#)
 - [Step 4. Enhance Appearance of the Documents List \(Optional\)](#)
 - [Step 5. Set up the Documents List](#)
 - [Step 6. Test your Enhanced WAM](#)
 - [Summary](#)
2. If you are **NOT** using the VLF application and DXDOCS file you carry out these steps:
 - [Step 1. Install Required Documents](#)
 - [Step 3a. Create WAM to Display Employee Documents](#)
 - [Step 4. Enhance Appearance of the Documents List \(Optional\)](#)
 - [Step 5a. Set up the Documents List](#)
 - [Step 6. Test your Enhanced WAM](#)
 - [Summary](#)

Before You Begin

WAMs allow you to serve stream files that:

- You don't want to store on your Web server
- Documents whose contents are stored in your application data base.
- Documents that are created on demand.

You can serve the contents of LOB data types (BLOBs and CLOBs) and stream files located in your Application Server with special WebRoutines.

```
Webroutine Name(SEND_SAMPLE) Response(#HTTPR) Desc('Sample Document')  
  * MYPATH is the directory where the sample documents are stored  
  #HTTPR.ContentFile := #MYPATH + 'sample'  
Endroutine
```

The WebRoutine can contain RDMLX code to create the contents of the file or determine which file to send. The only requirement is that you set the ContentFile to the file name that you want to serve.

Step 1. Install Required Documents

In this exercise you will need some files from the zip file described in *Before you Begin* at the beginning of [WAM チュートリアル \(英語\)](#).

1. You will need the following files to complete this exercise:

- V_Brown_CV.pdf
- Employee_Confidentiality_Agreement.pdf
- A0070_Details.doc
- A0070_Holidays_2013.xls
- A0070_Notes.txt
- A0070_Presentation.ppt
- MYDOCS.txt

The file MYDOCS.txt contains a list of the files to be displayed, which will be read by your WAM if you are not using the VLF application to store images in file DXDOCS.

Copy these files from the Extra Files to the folder: C:\Program Files (x86)\LANSA\

IBM i

- a. If you are running your WAMs on the IBM i, copy the file MYDOCS.txt into an IFS folder such as /LANSA_d13pgmlib/ where d13pgmlib is your LANSA program library.
- b. Copy all other above files into the IFS folder, such as:
LANSA_d13pgmlib/webserver/images/
- c. Change your RDML code to use the first path to TRANSFORM_FILE, and the second path to retrieve the files to send to the web server.
[WAM095. Appendix A](#) contains sample code for both Windows and IBM i execution.

Alternatively, use any files that you have available that will be recognized by windows and can be opened in the browser. Note, that in this case you will need to create a simple text file MYDOCS.txt containing a list of the files you wish to display.

2. A later step requires a set of small gif images for each file type, which will be displayed as a clickable image in the first column of the employee documents list.

Windows

These files are supplied in the Extra Files zip file. Copy the files from the File_GIFS folder to to:

c:\Program Files (x86)\LANSA\WebServer\Images

| | | |
|----------|----------|---------|
| doc.gif | pdf.gif | txt.gif |
| htm.gif | ppt.gif | xls.gif |
| html.gif | text.gif | |

IBM i

If you are running your WAM on the IBM i, copy the above gif files to the IFS for example to:

/LANSA_d13pgmlib/webserver/images/ where d13pgmlib is the name of your program library.

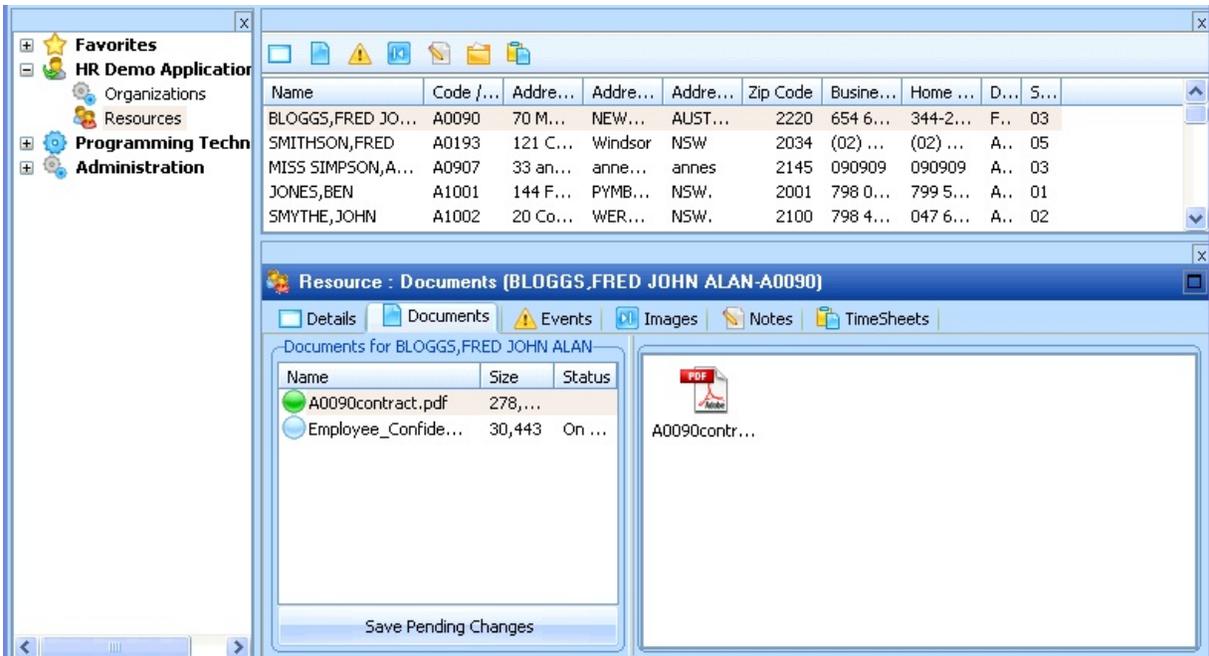
Scenario

This exercise may be completed in two ways:

1. Setting up documents for an employee in the file DXDOCS using the demonstration Visual Frameworks application. Continue immediately below to follow this approach
2. Reading an employee record and setting up a working list containing the names of images associated with the selected employee. To follow this approach, go directly to [Step 3a. Create WAM to Display Employee Documents](#).

The following assumes that you have Visual Frameworks (VLF) installed in the partition being used for the WAM training tutorials.

The file DXDOCS is maintained by the *Documents* command handler in the shipped VLF *HR Demo Application* for the business object, *Resources*.



You will use this application to set up documents for at least one employee. Your WAM will then enable these documents to be displayed in the browser. If you are testing your WAM applications locally (on your development PC), you can start the Visual Framework and work offline. You will be updating the DXDOCS file locally.

If you are creating WAM applications to run on an IBM i server, your Visual Framework must be configured to logon to the IBM i server. Your documents will then be stored in the file DXDOCS on the server.

You can find more details on how the Document command handler works (reusable part DX_DOCS) in the comments included in the component source RDML.

The WAM you will build in this exercise, `iiiDspEmpDocs` – Display Employee Documents, is a simple WAM that displays details for a single employee and a list of associated documents retrieved from file DXDOCS. When a document is selected it will be displayed in a new browser window.

File DXDOCS

Review the DXDOCS file definition in the *Repository*. Note that the file keys include all possible VLF instance list key fields. The highest level key contains the business object name. In this case, business object name will be `DEM_ORG_SEC_EMP`. See the VLF properties sheet for the *Resources* business object. The lowest level file key contains the document file name. For the business object *Resources*, the file key will be:

| Field | Value |
|-----------|-----------------|
| DF_ELOID | DEM_ORG_SEC_EMP |
| DF_ELKEY1 | #DEPARTMENT |
| DF_ELKEY2 | #SECTION |
| DF_ELKEY3 | #EMPNO |
| DF_ELFNAM | File Name |

All keys not shown in the table will be blank.

Understanding BLOBs

Refer to the *Technical Guide* for a detailed explanation of how to use the BLOB data type.

BLOB is a variable-length binary field of undefined maximum length.

The most common operation with BLOBs are saving files into the database and retrieving them so they can be viewed/edited/etc. In RDMLX, BLOB fields are manipulated as filenames.

Following is an example of saving a JPG as a BLOB field in a database file:

```
#MYBLOB := 'C:\temp\mypicture.jpg'
UPDATE FIELDS(#MYBLOB) IN_FILE(FILE1)
```

In RDMLX when a BLOB or CLOB field is used, keep in mind that the field contains a filename, not the actual data in the object. In RDMLX, LANSALOB fields will be manipulated as filenames. It is only in database IO commands that the BLOB or CLOB actual data itself is handled by reading from or writing to the named file.

Rather than the default property **.Value**, fields of type BLOB have a default property called **.FileName** to clearly indicate that changing the "value" of the field is actually changing its default property which is a file name property.

When BLOB and CLOB data is read from the database, files are automatically created in the directory structures under the LPTH= directory (for more information, refer to [Standard X_RUN Parameters](#) in the *Technical Reference*

Guide).

You can use the Documents command handler to store any type of document for an employee. For example, PDF, DOC, XLS, TXT etc.

Step 2. Set up Documents for an Employee

Before You Begin

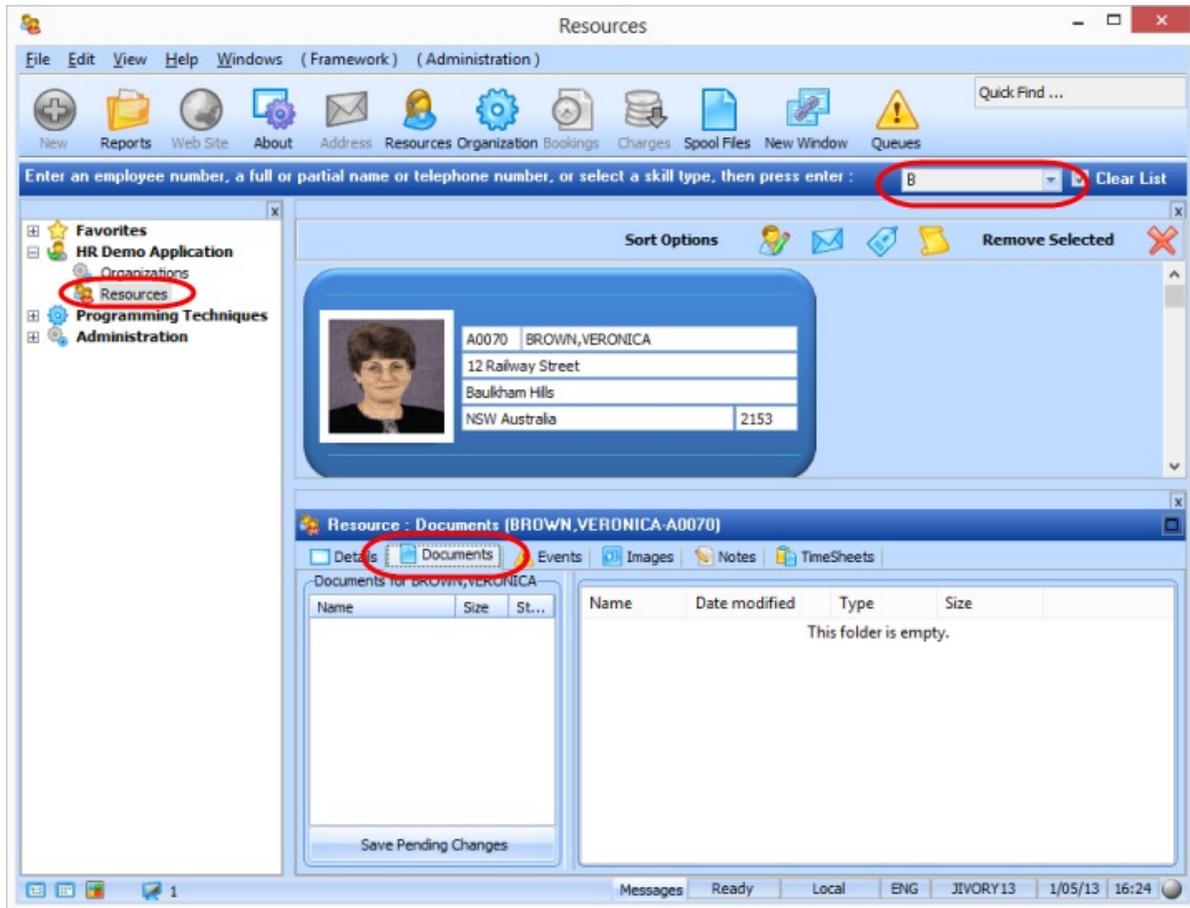
1. Start the Visual LANSa Framework (VLF).

If this is the first time VLF has been run, the shipped framework (vf_sy0001_system.xml) will be run by default. If necessary, in the following dialog select the *Open Latest Demonstration Version* checkbox to select the latest shipped framework:

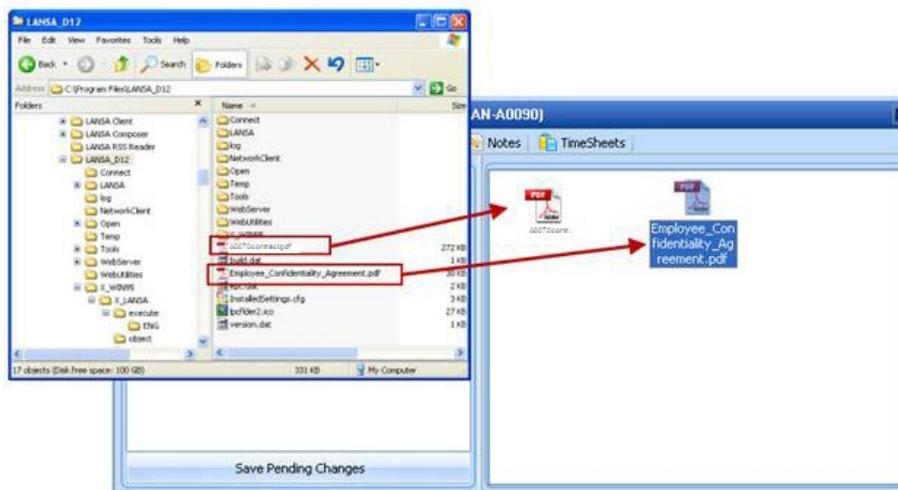


2. Select the *HR Demo Application* and then select the *Resources* business object. Enter **B** in the mini filter in the toolbar and press *Enter* to display employees in the Instance list. Select employee BROWN, VERONICA, A0070.

Select the *Documents* tab for this employee.



3. Open *Windows Explorer* and navigate to folder `c:\Program Files\LANSA`. Drag the sample files (doc, txt, ppt, and pdf) into the right hand *Documents* panel as shown:



4. Select the *Save Pending Changes* button to save the documents to the file

DXDOCS.

Step 3. Create WAM to Display Employee Documents

1. Create a new WAM:

Name: **iiiDspEmpDocs**

Description: **Display Employee Documents**

Layout Weblet: **iiilay01**

2. Define your WAM based on the following logic:

- Map field STDREENTRY for *both as a *hidden field
- Define a Group_by, EMPDATA for fields EMPNO, SURNAME, GIVENAME, ADDRESS1, ADDRESS2, ADDRESS3, POSTCODE, DEPARTMENT and SECTION.
- Define a working list DOCLIST, containing fields DF_ELFNAM and PRIFILRRN.
PRIFILRRN should be a hidden field.
- Define a WebRoutine BEGIN
 - Map field EMPNO for *both
 - Map fields FULLNAME, ADDRESS1, ADDRESS2, ADDRESS3, POSTCODE, DEPARTMENT and SECTION and list DOCLIST for *output. All fields should have a display attribute of *output.
 - If STDREENTRY = S
 - Clear list DOCLIST
 - Fetch fields for group_by EMPDATA from file PSLMST with the key EMPNO
 - FULLNAME = GIVENAME + ', ' + SURNAME
 - Select field DF_ELFNAM from the file DXDOCS with the key 'DEM_ORG_SEC_EMP', DEPARTMENT, SECTION, EMPNO and return relative record number to PRIFILRRN
 - Add entry to DOCLIST
 - End of select
 - End of if
- End of routine

- Define a WebRoutine SEND_DOCUMENT with a Response keyword of #HTTPR
- Map for *input field PRIFILRRN
- Fetch field DX_ELLOB from the file DXDOCS with the relative record number PRIFILRRN.
- Set #HTTPR property ContentFile to #DXELLOB.FileName
- End of routine

Note: The field DX_ELLOB returns the file into a local temporary directory, using an 8.3 filename. For example:

C:\DOCUME~1\pcuser\LOCALS~1\Temp\lobuser\pcuser\5152\dxdoc

This path may be defined by the LPTH=directory run time parameter. If the LPTH parameter is not defined, the path used will be the TPTH setting. The default value for this path is the user's temporary path, for example:

C:\Users\John\AppData\Local\Temp

This can be quickly found using %temp% in the Explorer address bar.

For further information, refer to the [Standard X_RUN Parameters](#) in the *Technical Reference Guide*.

3. Compile **iiiDspEmpDocs** and open the WebRoutine **BEGIN** in the *Design* view. It should look like the following:

The screenshot shows a web form design view with the following elements:

- Employee Information Fields:**
 - Employee Number: VALUE E
 - Employee full name: ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNO
 - Street No and Name: aAbBcCdDeEfgGhHijjklLm
 - Suburb or dept: aAbBcCdDeEfgGhHijjklLm
 - State and Country: aAbBcCdDeEfgGhHijjklLm
 - Post / Zip Code: 123456
 - Department Code: Value DEPARTMENT
 - Section Code: Value SECTION
- Document Table:**

| Document long file | Primary RRN |
|---|-------------|
| ABCDEFGHIJKLMN...QRSTUVWXYZABCDEFGHIJKLMN...XYZ | |
| ABCDEFGHIJKLMN...QRSTUVWXYZABCDEFGHIJKLMN...XYZ | |
- Hidden Content:** A section labeled "Hidden Content" is visible at the bottom of the form.

4. Drop a *Push Button* weblet alongside the employee number field. Set up the push button properties:
-

| Property | Value |
|-------------------|-------------------------------|
| Caption | Details |
| On_click_wname | BEGIN |
| submitExtraFields | Field Name: STDREENTRY |
| | Literal value: S |

5. Save your changes.

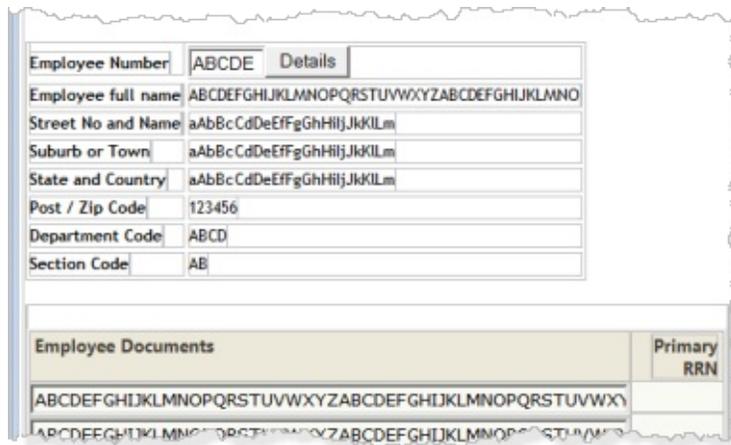
6. Select the column heading "Document long file" and delete it.

Type in a new the column heading "Employee Documents".

You may need to click somewhere else in the layout, to refresh the column heading with your changes.

7. Save your changes.

Your design should look like the following:



8. Drop an *Anchor* weblet into the file name column of the list (the left hand column). Ignore the increase in width of this column. At run time it will display with the width of the actual file names.

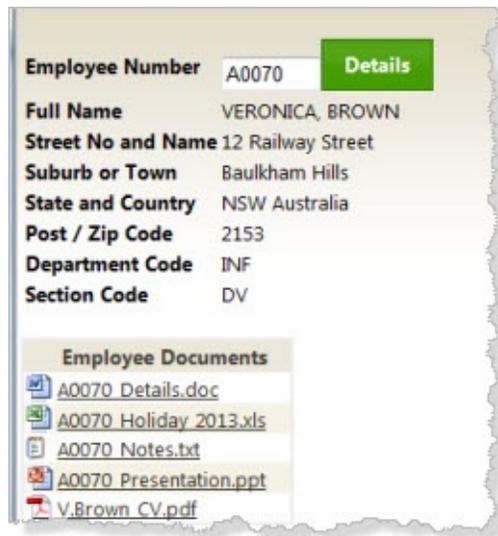
Set up the *Anchor* weblet properties:

| Property | Value |
|------------------|------------------|
| Currentrowhfield | PRIFILRRN |

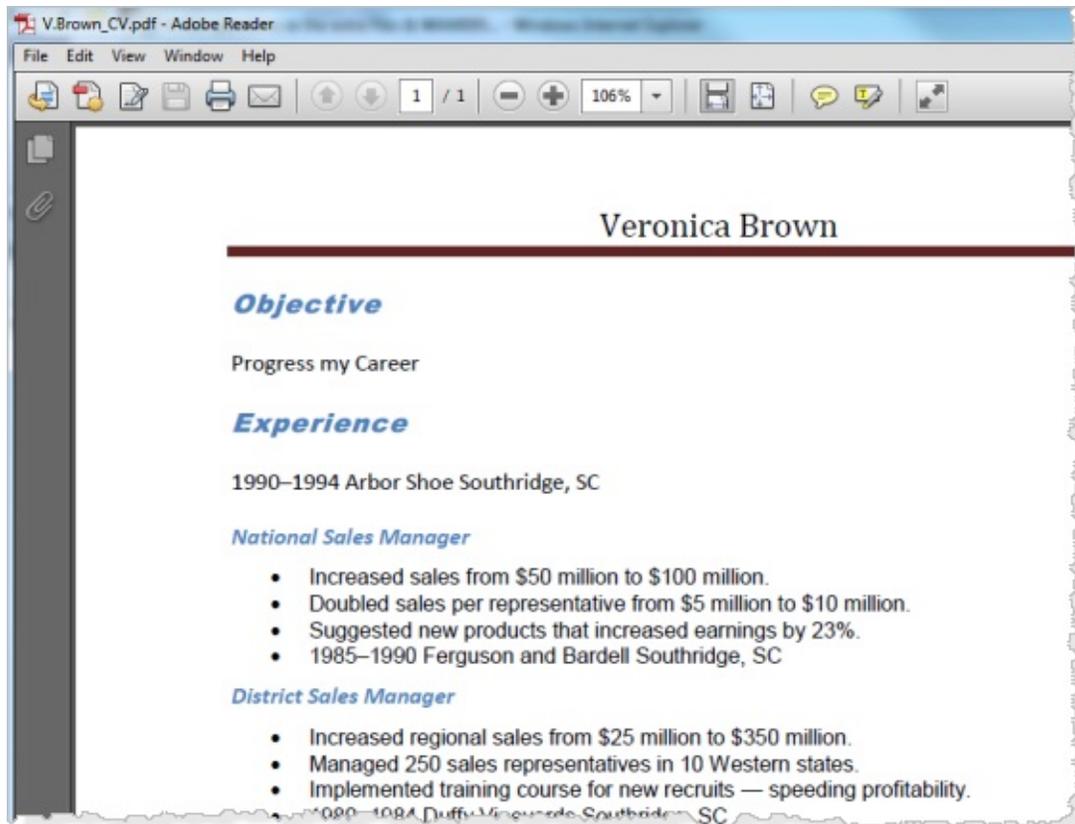
| | |
|--------------------|----------------------|
| Currentrownumval | \$PRIFILRRN |
| On_click_wname | SEND_DOCUMENT |
| Show_in_new_window | True |

Save your changes.

9. Execute and test your WAM in the browser.
 - a. Enter employee number A0070 and select the *Details* push button. Your web page should look like the following:



- b. Select one of the documents shown in the list. The document should be displayed in a new browser window.



Continue at [Step 4. Enhance Appearance of the Documents List \(Optional\)](#)

Step 3a. Create WAM to Display Employee Documents

Follow this step if you did not set up documents for an employee using the demonstration VLF application, in the file DXDOCS.

This WAM will display a fixed list of documents for an employee, based on a data from a supplied text file. To simplify building the list of documents, the WAM reads a text file MYDOCS.txt to populate a working list. Another working list is populated for display. A later step will add another column to this list which will contain a clickable image.

1. Create a new WAM:

*Name:***iiiDspEmpDocs**

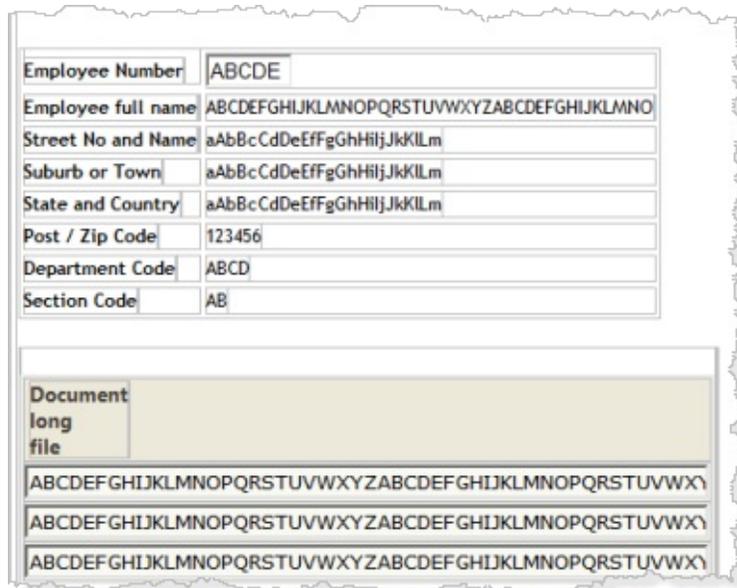
*Description:***Display Employee Documents**

*Layout Weblet:***iiilay01**

2. Create your WAM based on the following logic:

- Map field STDREENTRY for *both as a *hidden field
- Define a Group_By, EMPDATA for fields, EMPNO, SURNAME, GIVENAME, ADDRESS1, ADDRESS2, ADDRESS3, POSTCODE, DEPARTMENT and SECTION.
- Define a working list DOCLIST containing fields DF_ELFNAM
- Define a working list MYDOCS containing field STD_TEXTL
- Define a work field RETCODE with reference field IO\$STS
- Define a webroutine BEGIN
- Map field EMPNO for *both
- Map fields FULLNAME, ADDRESS1, ADDRESS2, ADDRESS3, POSTCODE, DEPARTMENT and SECTION for *output. All fields should have a display attribute of *output
- If STDREENTRY = S
- Clear list DOCLIST
- Clear list MYDOCS
- Fetch group_by EMPDATA from file PSLMST with the key EMPNO
- Use the BIF transform_file to populate the list MYDOCS from the file "C:\Program Files (x86)\LANSA\MYDOCS.txt"
- Selectlist MYDOCS

- Change DF_ELFNAM to STD_TEXTL
 - Add an entry to DOCLIST
 - Endselect
 - Endif
3. Define a WebRoutine SEND_DOCUMENT with a Response keyword of #HTTPR
- Map field DF_ELFNAM for *input
 - Set #HTTPR, property ContentFile to "C:\Program Files (x86)\LANSA\" + DF_ELFNAM
 - Endroutine
- Sample code is supplied in [WAM095. Appendix B](#).
4. Compile your WAM. Open the WebRoutine **BEGIN** in the *Design* view, it should look like the following:



5. Drop a *Push Button* weblet alongside the employee number field. Set up the push button properties:

| Property | Value |
|-------------------|------------|
| Caption | Details |
| On_Click_wrname | BEGIN |
| submitExtraFields | STDREENTRY |

| |
|------------------|
| Literal Value: S |
|------------------|

6. Save your changes.
7. Select the column heading "Document long file" and delete it.
Type in a new column heading - **Employee Documents**.
You may need to click on another part of the layout to refresh the column heading.
8. Drop an *Anchor* weblet into the list's file name column. Ignore the increase in width of the column. At run time it will be displayed with the width of the actual file names.

Set up the Anchor weblet properties:

| Property | Value |
|--------------------|---------------|
| Currentrowhfield | DF_ELFNAM |
| Currentrownumval | \$DF_ELFNAM |
| On_click_wname | SEND_DOCUMENT |
| Show_in_new_window | True |

Your design should look like the following:

| | | |
|--|---|---------|
| Employee Number | ABCDE | Details |
| Employee full name | ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNO | |
| Street No and Name | aAbBcCdDeEfFgGhHijJkKlLm | |
| Suburb or Town | aAbBcCdDeEfFgGhHijJkKlLm | |
| State and Country | aAbBcCdDeEfFgGhHijJkKlLm | |
| Post / Zip Code | 123456 | |
| Department Code | ABCD | |
| Section Code | AB | |
| | | |
| | | |
| | | |
| Employee Documents | | |
| ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOQRSTUVWXYZABCDEFGHI | | |
| ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOQRSTUVWXYZABCDEFGHI | | |

9. Save your changes.

10. Execute your WAM in the browser. Any employee number may be entered (A0070, A0090, A1234 etc). The WAM will always display the employee details and a fixed list of documents.

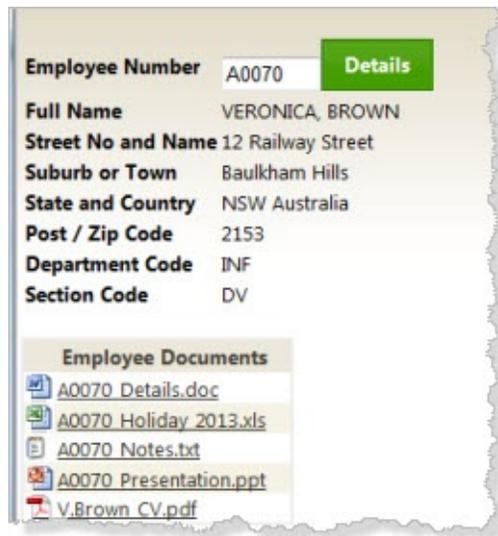
Click on one of the documents to display it. The way that the browser and Windows handles the request will depend on the version of browser and the version of Windows being used. For example, in Windows 7 with MS Office installed and IE10, the txt and pdf files are displayed in a new browser tab.

The Office documents prompt to be displayed in the required Office program.

Continue with *Step 4. Enhance Appearance of the Documents List (Optional)*

Step 4. Enhance Appearance of the Documents List (Optional)

This step adds a column containing a suitable image to the list of documents, as shown:



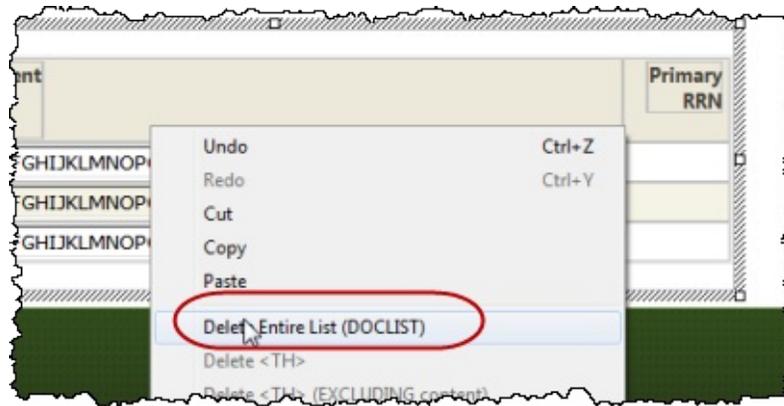
1. Extend WAM **iiiDspEmpDocs** as follows:
 - a. Define a character work field FILENAME, length 8.
 - b. Add field FILENAME as the first field in working list DOCLIST.
2. The field FILENAME needs to be populated with the name of the appropriate image to display for each BLOB, depending on its file type. The field DF_ELFNAM already contains the long file name for each BLOB (document).

Extend the **BEGIN** WebRoutine as shown. Within the SELECT loop add the following logic, before the ADD_ENTRY:

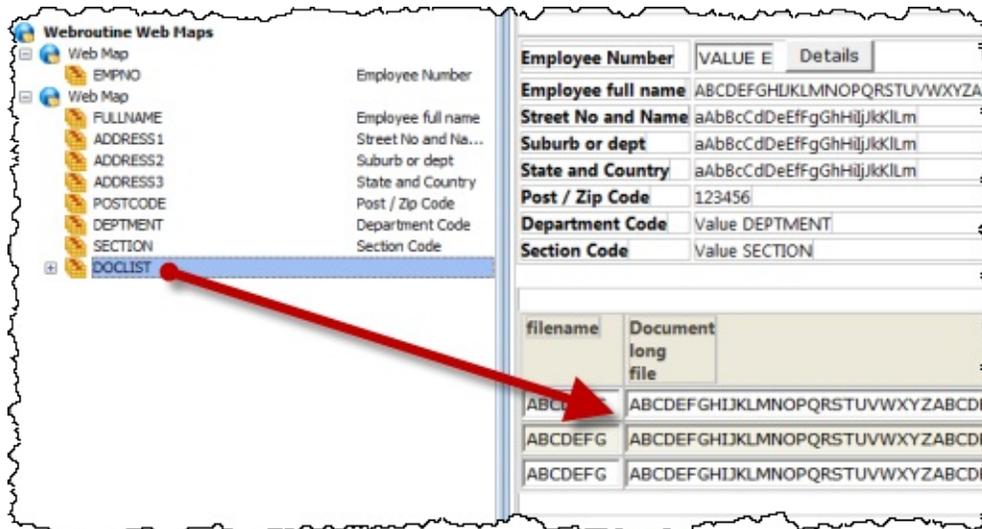
- #std_count := #df_elfnam.LastPositionOf('.')
 #std_texts := #df_elfnam.substring((#std_count + 1), 4)
- #filename := #std_texts.trim + '.gif'

See [WAM095. Appendix C](#) for the changed code, if required.

3. Recompile your WAM.
4. Open the WebRoutine **BEGIN** in the *Design* view. Select the **Employee Documents** list and use the context menu to *Delete Entire List*.



5. Select the *WebRoutine Output* tab and drag the list, **DOCLIST**, back onto the page.



6. Delete the column heading text for Filename. Delete the column heading, **Document long file** and replace it with **Employee Documents**.

If you are not using the VLF and the DXDOCS file, continue at [Step 5a. Set up the Documents List](#) otherwise continue at [Step 5. Set up the Documents List](#).

Step 5. Set up the Documents List

Follow this step when the DXDOCS file **is being used**.

1. Drag and drop a *Clickable Image* weblet into the first column (filename) and set up the weblet as follows:

| Property | Value |
|---------------------|----------------------------------|
| Currentrowhfield | PRIFILRRN |
| Currentrownumvalue | \$PRIFILRRN |
| Rentryvalue | S |
| Tooltip | Select image to display document |
| on_click_wrname | SEND_DOCUMENT |
| show_in_new_window | True |
| relative_image_path | \$FILENAME |

2. Drag and drop an *Anchor* weblet into the second column (DF_ELFNAM) and set up the weblet as follows:

| Property | Value |
|--------------------|---------------|
| currentrowhfield | PRIFILRRN |
| currentrownumvalue | \$PRIFILRRN |
| rentryvalue | S |
| on_click_wrname | SEND_DOCUMENT |
| show_in_new_window | True |

3. Save your changes

Step 5a. Set up the Documents List

Follow this step when you are **NOT** using the DXDOCS file.

1. Drag and drop a *Clickable Image* weblet into the first column (filename) and set up the weblet as follows:

| Property | Value |
|---------------------|----------------------------------|
| currentrowhfield | DF_ELFNAM |
| currentrownumvalue | \$DF_ELFNAM |
| rentryvalue | S |
| tooltip | Select image to display document |
| on_click_wrname | SEND_DOCUMENT |
| show_in_new_window | True |
| relative_image_path | \$FILENAME |

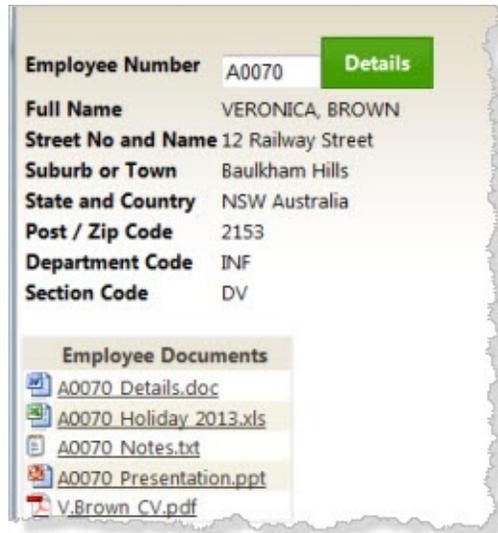
2. Drag and drop an *Anchor* weblet into the second column (DF_ELFNAM) and set up the weblet as follows:

| Property | Value |
|--------------------|---------------|
| currentrowhfield | DF_ELFNAM |
| currentrownumvalue | \$DF_ELFNAM |
| rentryvalue | S |
| on_click_wrname | SEND_DOCUMENT |
| show_in_new_window | True |

3. Save your changes.

Step 6. Test your Enhanced WAM

1. Test your application for employee A0070. If you have saved other types of document for this employee, your results will look like this:



If you are not using the DXDOCS file, your WAM outputs a fixed list of documents for an employee.

2. You should now be able to click on the hyperlink in the Employee Documents column or the clickable image in the first column to display a document.

Summary

Important Observations

- Standard Windows documents such as Adobe Acrobat (PDF), Word (DOC) and Excel (XLS) can easily be displayed by a WAM WebRoutine using the Response(#HTTTPR) parameter.
- You should read the relevant sections in the *Web Application Module Guide* and *Technical Reference* before implementing files storing BLOB and CLOB data.

What I Should Know

- How to write a WebRoutine that is using the Response() parameter to output a document.
- How to set up a list column with dynamic images

WAM095. Appendix A

Using DXDOCS File - Sample RDMLX for iiiDspEmpDocs

This WAM can be run locally or on IBM i.

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_WAM)
Web_Map For(*BOTH) Fields((#stdrentry *hidden))
Group_By Name(#empdata) Fields(#EMPNO #SURNAME #GIVENAME #A

Def_List Name(#doclist) Fields(#df_elfnam (#prifilrrn *hidden)) Type(*Worki

Webroutine Name(Begin)
Web_Map For(*both) Fields(#empno)
Web_Map For(*output) Fields((#fullname *out) (#address1 *out) (#address2 *
If (#stdrentry = S)
Clr_List Named(#doclist)
Fetch Fields(#empdata) From_File(pslmst) With_Key(#empno)
#fullname := #givename + ', ' + #surname
Select Fields(#df_elfnam) From_File(dxdocs) With_Key('DEM_ORG_SEC_E
Add_Entry To_List(#doclist)
Endselect
Endif
Endroutine

Webroutine Name(SEND_DOCUMENT) Response(#HTTPR) Desc('Sample I

Web_Map For(*input) Fields(#PRIFILRRN)

Fetch Fields(#df_elfnam #dx_elblob) From_File(dxdocs) With_Rrn(#prifilrrn)
#HTTPR.ContentFile := #DX_ELLOB.FileName

Endroutine

End_Com
```

WAM095. Appendix B

Sample RDMLX for iiiDspEmpDocs, when not using DXDOCS file.

This WAM to be run locally

```
Begin_Com Role(*EXTENDS #PRIM_WAM) Layoutweblet('iiilay01')
Web_Map For(*BOTH) Fields((#stdrentry *hidden))
Group_By Name(#empdata) Fields(#EMPNO #SURNAME #GIVENAME
#ADDRESS1 #ADDRESS2 #ADDRESS3 #POSTCODE #deptment #section)
```

```
Def_List Name(#doclist) Fields(#df_elfnam) Type(*Working)
Def_List Name(#mydocs) Fields(#std_textl) Type(*working)
```

```
Webroutine Name(Begin)
Web_Map For(*both) Fields(#empno)
Web_Map For(*output) Fields((#fullname *out) (#address1 *out) (#address2
*out) (#address3 *out) (#postcode *out) (#deptment *out) (#section *out)
#doclist)
Define Field(#retcode) Reffld(#io$sts)
If (#stdrentry = S)
Clr_List Named(#doclist)
Fetch Fields(#empdata) From_File(pslmst) With_Key(#empno)
#fullname := #givename + ', ' + #surname
Use Builtin(transform_file) With_Args(#mydocs 'c:\Program Files
(x86)\LANSA13\WAM095_DOCS.txt') To_Get(#retcode)
Selectlist Named(#mydocs)
#df_elfnam := #std_textl.trim
Add_Entry To_List(#doclist)
Endselect
Endif
Endroutine
```

```
Webroutine Name(SEND_DOCUMENT) Response(#HTTPR) Desc('Sample
Document')
Web_Map For(*input) Fields(#DF_ELFNAM)
#HTTPR.ContentFile := "C:\Program Files (x86)\LANSA13\" +
#DF_ELFNAM
Endroutine
```

End_Com

This WAM to be run on the IBM i

```
Begin_Com Role(*EXTENDS #PRIM_WAM) Layoutweblet('iilay01')
Web_Map For(*BOTH) Fields((#stdrentry *hidden))
Group_By Name(#empdata) Fields(#EMPNO #SURNAME #GIVENAME
#ADDRESS1 #ADDRESS2 #ADDRESS3 #POSTCODE #deptment #section)
Def_List Name(#doclist) Fields(#df_elfnam) Type(*Working)
Def_List Name(#mydocs) Fields(#std_textl) Type(*working)
Webroutine Name(Begin)
Web_Map For(*both) Fields(#empno)
Web_Map For(*output) Fields((#fullname *out) (#address1 *out) (#address2
*out) (#address3 *out) (#postcode *out) (#deptment *out) (#section *out)
#doclist)
Define Field(#retcode) Reffld(#io$sts)
If (#stdrentry = S)
Clr_List Named(#doclist)
Fetch Fields(#empdata) From_File(pslmst) With_Key(#empno)
#fullname := #givename + ', ' + #surname
Use Builtin(transform_file) With_Args(#mydocs
'/LANSA_d13pgmlib/MYDOCS.txt') To_Get(#retcode)

Selectlist Named(#mydocs)
#df_elfnam := #std_textl.trim
Add_Entry To_List(#doclist)
Endselect
Endif
Endroutine
Webroutine Name(SEND_DOCUMENT) Response(#HTTPR) Desc('Sample
Document')
Web_Map For(*input) Fields(#DF_ELFNAM)
#HTTPR.ContentFile := "/LANSA_d13pgmlib/webserver/images/" +
#DF_ELFNAM

Endroutine
End_Com
```

WAM095. Appendix C

Sample RDMLX for Enhanced Documents List

Using file DXDOCS

Changes are highlighted in red italics.

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_WAM)
Web_Map For(*BOTH) Fields((#stdrentry *hidden))
Group_By Name(#empdata) Fields(#EMPNO #SURNAME #GIVENAME
#ADDRESS1 #ADDRESS2 #ADDRESS3 #POSTCODE #deptment #section)
```

```
Def_List Name(#doclist) Fields(#filename #df_elfnam (#prifilrrn *hidden))
Type(*Working)
Define Field(#filename) Type(*char) Length(8)
Webroutine Name(Begin)
Web_Map For(*both) Fields(#empno)
Web_Map For(*output) Fields((#fullname *out) (#address1 *out) (#address2
*out) (#address3 *out) (#postcode *out) (#deptment *out) (#section *out)
#doclist)
If (#stdrentry = S)
Clr_List Named(#doclist)
Fetch Fields(#empdata) From_File(pslmst) With_Key(#empno)
#fullname := #givename + ', ' + #surname
Select Fields(#df_elfnam) From_File(dxdocs)
With_Key('DEM_ORG_SEC_EMP' #deptment #section #empno)
Return_Rrn(#PRIFILRRN)
#std_count := #df_elfnam.LastPositionOf('.')
#std_texts := #df_elfnam.substring( (#std_count + 1), 4 )
#filename := #std_texts.trim + '.gif'
Add_Entry To_List(#doclist)
Endselect
Endif
Endroutine
```

```
Webroutine Name(SEND_DOCUMENT) Response(#HTTPR) Desc('Sample
Document')
Web_Map For(*input) Fields(#PRIFILRRN)
```

```
Fetch Fields(#df_elfnam #dx_elblob) From_File(dxdocs) With_Rrn(#prifilrn)
#HTTPR.ContentFile := #DX_ELLOB.FileName
Endroutine
End_Com
```

Not Using DXDOCS File

WAM running locally

Changes are highlighted in red italics.

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_WAM) Layoutweblet('iiilay01')
Web_Map For(*BOTH) Fields((#stdrentry *hidden))
Group_By Name(#empdata) Fields(#EMPNO #SURNAME #GIVENAME
#ADDRESS1 #ADDRESS2 #ADDRESS3 #POSTCODE #deptment #section)
```

```
Def_List Name(#doclist) Fields(#filename #df_elfnam) Type(*Working)
Def_List Name(#mydocs) Fields(#std_textl) Type(*working)
```

```
Define Field(#filename) Type(*char) Length(8)
Webroutine Name(Begin)
Web_Map For(*both) Fields(#empno)
Web_Map For(*output) Fields((#fullname *out) (#address1 *out) (#address2
*out) (#address3 *out) (#postcode *out) (#deptment *out) (#section *out)
#doclist)
Define Field(#retcode) Reffld(#io$sts)
If (#stdrentry = S)
Clr_List Named(#doclist)
Fetch Fields(#empdata) From_File(pslmst) With_Key(#empno)
#fullname := #givename + ', ' + #surname
Use Builtin(transform_file) With_Args(#mydocs 'c:\Program Files
(x86)\LANSA13\MYDOCS.txt') To_Get(#retcode)
Selectlist Named(#mydocs)
#df_elfnam := #std_textl.trim
#std_count := #df_elfnam.LastPositionOf( '.' )
#std_texts := #df_elfnam.substring( (#std_count + 1), 4 )
#filename := #std_texts.trim + '.gif'
```

```
Add_Entry To_List(#doclist)
```

Endselect
Endif
Endroutine

Webroutine Name(SEND_DOCUMENT) Response(#HTTPR) Desc('Sample Document')
Web_Map For(*input) Fields(#DF_ELFNAM)
#HTTPR.ContentFile := "C:\Program Files (x86)\LANSA13\" +
#DF_ELFNAM
Endroutine
End_Com

WAM running on IBM i

Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_WAM)
Web_Map For(*BOTH) Fields((#stdrentry *hidden))
Group_By Name(#empdata) Fields(#EMPNO #SURNAME #GIVENAME
#ADDRESS1 #ADDRESS2 #ADDRESS3 #POSTCODE #deptment #section)

Def_List Name(#doclist) Fields(#filename #df_elfnam (#prifilrrn *hidden))
Type(*Working)
Define Field(#filename) Type(*char) Length(8)
Webroutine Name(Begin)
Web_Map For(*both) Fields(#empno)
Web_Map For(*output) Fields((#fullname *out) (#address1 *out) (#address2
*out) (#address3 *out) (#postcode *out) (#deptment *out) (#section *out)
#doclist)
If (#stdrentry = S)
Clr_List Named(#doclist)
Fetch Fields(#empdata) From_File(pslmst) With_Key(#empno)
#fullname := #givename + ', ' + #surname
Select Fields(#df_elfnam) From_File(dxdocs)
With_Key('DEM_ORG_SEC_EMP' #deptment #section #empno)
Return_Rrn(#PRIFILRRN)
#std_count := #df_elfnam.LastPositionOf('.')
#std_texts := #df_elfnam.substring((#std_count + 1), 4)
#filename := #std_texts.trim + '.gif'

```
Add_Entry To_List(#doclist)
Endselect
Endif
Endroutine
```

```
Webroutine Name(SEND_DOCUMENT) Response(#HTTPR) Desc('Sample
Document')
Web_Map For(*input) Fields(#PRIFILRRN)
Fetch Fields(#df_elfnam #dx_elblob) From_File(dxdocs) With_Rrn(#prifilrrn)
#HTTPR.ContentFile := #DX_ELBLOB.FileName
Endroutine
End_Com
```

WAM100 - Using Cascading Style Sheets

Objectives

To learn how your own style sheet can be set up and used to control the appearance of specific elements of your web pages.

During this exercise you will use: **Microsoft Internet Explorer, Developer Tools** (included in IE8, 9 and 10) that enable the structure of a web page and its style rules to be explored in detail.

In order to complete this exercise you will complete the following:

- Review [What are Cascading Style Sheets?](#) and [What CSS files are loaded and how do I add my own?](#)
- [Step 1. Create WAM iii Using CSS](#)
- [Step 2. Create new Style Sheet](#)
- [Step 3. Create an External Resource](#)
- [Step 4. Apply Style Sheet to WAM iii Using CSS](#)
- [Step 5. Apply External Resource to the Common Layout](#)
- [Step 6. Make the Style Sheet specific to lists named EMPLIST](#)
- [Step 7. Highlight a Column](#)
- [Summary](#)

Before You Begin

This exercise does not depend on knowledge gained from all preceding exercises. The following exercises should have been completed:

- [WAM005 - Create Your First WAM](#)
- [WAM010 - Using WEB_MAPs](#)
- [WAM015 - Working Lists](#)
- [WAM020 - WAM Navigation](#)
- [WAM030 - Employee Enquiry](#)

What are Cascading Style Sheets?

A Cascading Style Sheet tells the browser how to display page elements. Cascading Style Sheet information determines things like the fonts and color schemes, DHTML effects, alignment, border size and color, but may also be used to define images and other features related to the interface. These properties can be assigned to individual elements identified by an ID, or groups

of elements identified by type, location and class.

Many of the shipped weblots include style (or class) properties. The default style applied to a property, and the full set of styles available in the dropdown list associated with these properties, relate directly back to the CSS file referenced on the WAM's related layout.

For full information see <http://www.w3schools.com/css>

Style sheet files (CSS) are simply text and can be edited with any text editor such as Notepad. If you are working with style sheets it's a good idea to obtain a specialist style sheet editor. There are a number available, *TopStyle* is one example. They will make your editing faster and more accurate and make it easier to navigate through and manage a large number of styles defined in a style sheet. They will also help you to more rapidly learn the options available when defining styles.

What CSS files are loaded and how do I add my own?

The `std_style_v2` weblot takes care of creating all the `<link>` tags needed to load the CSS files so you need to include it in the `<head>` section of your layouts.

The `std_style_v2` weblot always loads `std_style.min.css` into every layout. This defines the non-theme related properties of all LANSA supplied weblots.

It then loads any CSS files defined by its `theme_css_filename` and `css_files` properties. These properties are provided for backwards compatibility with layouts built with older versions of the weblot. For new layouts, you should specify 'none' in `theme_css_filename` and use External Resources to define additional CSS files you want to include.

Next, it adds any CSS files defined as *External Resources* referenced in the webroutine, layout or weblots used by the webroutine.

Finally, the `std_style_v2` weblot loads a stylesheet defined by the variable `$lweb_std_css_language_overlay`. This variable is defined in the [std_locale](#) weblot and provides a means to apply language specific CSS modifications.

For more detail on this topic see the *Web Application Modules* guide.

What Cascading Style Sheets are available?

The main CSS stylesheets are in the main style directory under the images directory.

The themed CCS stylesheets are under the jQuery subdirectory—under a subdirectory named after the theme.

See the Web Application Modules guide for a full list of the other stylesheets

available to WAMs.

Cascading Style Sheets in Action

For an appreciation of how the shipped style sheets combine, this image is the begin WebRoutine in WAM *iiiSecMaint – Section Maintenance* with the themelet stylesheets removed:



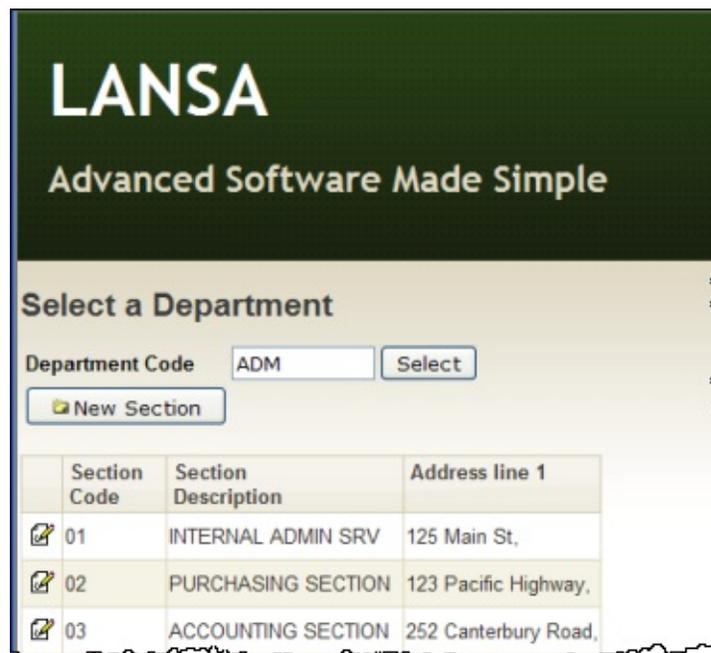
LANSA
Advanced Software Made Simple
[Home](#) [Services](#) [Contact](#) [About](#)

Select a Department

Department Code

| Section Code | Section Description | Address line 1 |
|--------------|---------------------|----------------------|
| 01 | INTERNAL ADMIN SRV | 125 Main St, |
| 02 | PURCHASING SECTION | 123 Pacific Highway, |
| 03 | ACCOUNTING SECTION | 252 Canterbury Road, |

This image is the **begin** WebRoutine with the *SouthStreet theme* style sheets applied:



LANSA
Advanced Software Made Simple

Select a Department

Department Code

| Section Code | Section Description | Address line 1 |
|--------------|---------------------|----------------------|
| 01 | INTERNAL ADMIN SRV | 125 Main St, |
| 02 | PURCHASING SECTION | 123 Pacific Highway, |
| 03 | ACCOUNTING SECTION | 252 Canterbury Road, |

Clearly, to change the appearance of any element on your web page, you need to

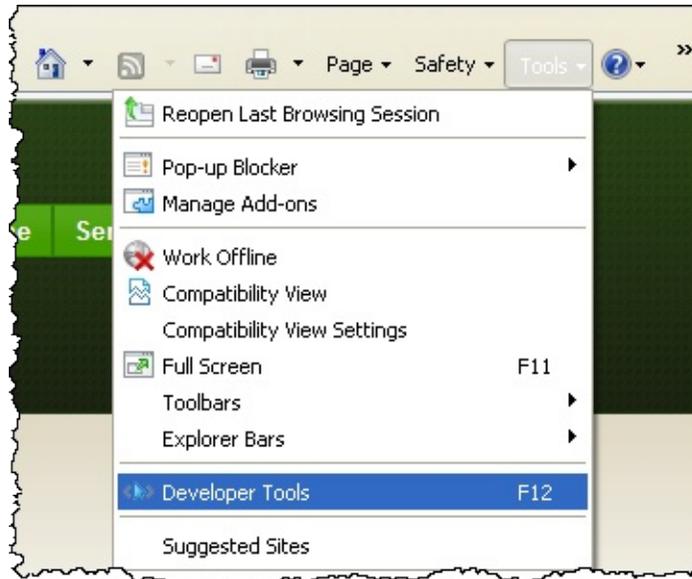
define a style sheet that overrides the styles defined by your chosen theme.

Use CSS with Lists and Grids

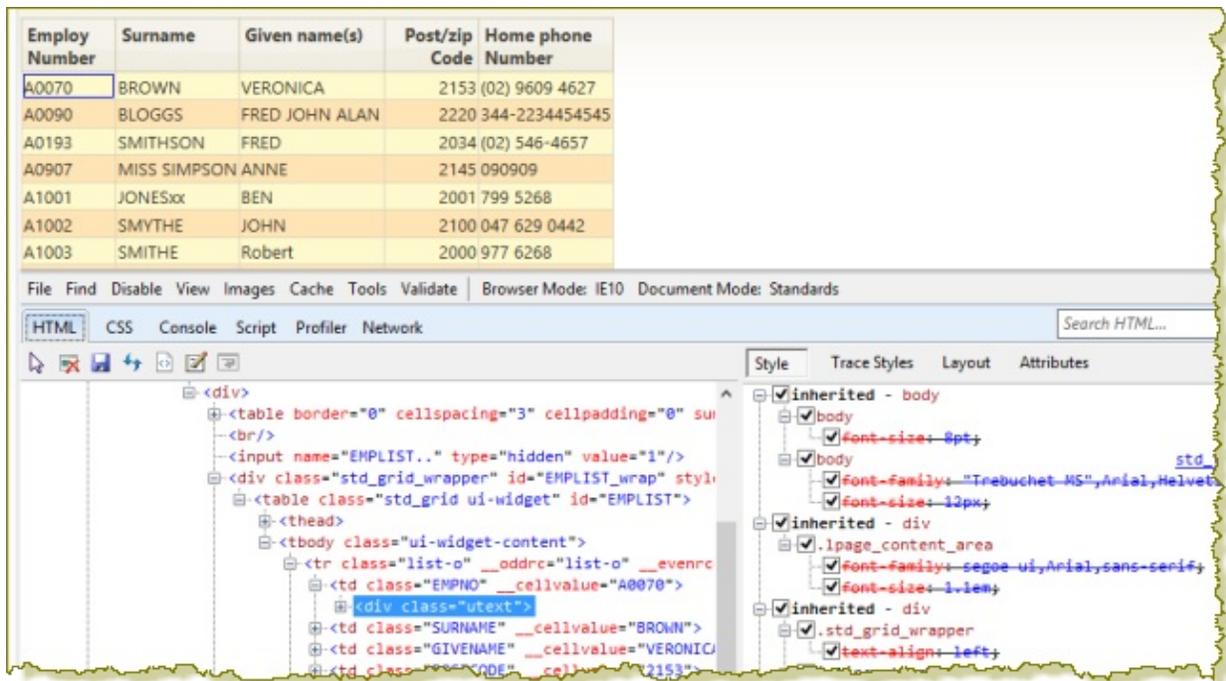
This exercise will demonstrate how to change the appearance of elements in a list. The exercise is about how to identify the elements you wish to change and then implement these changes with your own style sheet.

To use CSS effectively, you must first understand how the how the screen element you are interested in is constructed.

If you are using Internet Explorer 8, 9 or 10 you already have Developer Tools that may be started from the browser Tools menu, or by using F12. Tools like this are essential for understanding how your page is constructed. The *Mozilla Firefox* browser has a similar optional tool known as *Firebug*.



This screen picture shows *Developer Tools* running with a WAM which displays a list of employees:



Note that the *Select element by click*  icon in Developer Tools can be used to select the element on the page that you want to examine.

In the above example the table containing the employee list has been selected. The *Developer Tools* then displays the HTML, attributes and styles for the selected element.

Note: *Developer Tools* also has good display source features that will help you to see in more detail how part of the screen is defined. You will use these later in this exercise.

Using *Developer Tools* and selecting the table containing the list, shows that the browselist EMPLST table is defined as:

```
<DIV style="WIDTH: 617px" id=EMPLIST_wrap class=std_grid_wrapper>
<TABLE style="CURSOR: default" id=EMPLIST class="std_grid ui-
widget">
<THEAD>
<TR class="list-h ui-widget-header">
<TH class="utext EMPNO std_grid_sort_indicator" __allowsort="true" __moc
Employ
<BR>
Number
<DIV class=std_grid_cell_sizer><!--.--><!------>
</DIV>
```

```

</TH>
.....
<TBODY class=ui-widget-content>
<TR class=list-o __evenrc="list-e" __oddr="list-o">
<TD class=EMPNO __cellValue="A1003">
<INPUT id=EMPLIST.0001.EMPNO class=utext onchange="return isValidTe:
</TD>
<TD class=GIVENAME __cellValue="Robert">
<DIV class=utext>Robert<!--></DIV>
</TD>
.....
</TR>
.....
</TBODY>
</TABLE>

```

For clarity, some detail has been omitted from the above code.

Some points to note:

- A list is wrapped by a DIV with the *class* of **std_grid_wrapper**.
- The DIV also has an *id* of **<listname>_wrap**, where **<listname>** is the name of the list in the RDMLX.
- For a grid, the wrapper DIV also has a *class* of **std_grid_wrapper**. The *id* is **LANSA_<gridname>_wrap** where **<gridname>** is the name assigned to the grid in the Design view.
- The DIV wrapper provides the size and position for the grid, drawing any scrollbars as necessary.
- In a list, the row *class* names are alternated between **list-o** and **list-e**.
- For a grid, the *class* names for odd and even rows are defined by the grid properties **odd_row_class** and **even_row_class**, which have default values of *odd_row* and *even_row*.
- A list's table cells (<td> tags) are given a *class* name of the field name. This provides a way to apply styles to specific columns.
- Input fields are given a class that represents the data type of the field. Fields of type alpha, char or string will be "text", "utext" or "ltext" depending on the input case of the field. Fields of type packed, signed, integer, float, date, time or datetime will have a class name of "number". Boolean fields will be

"ltext" and all other fields will be "text".

The best way to understand how it all fits together is to look at a few examples.

Step 1. Create WAM iii Using CSS

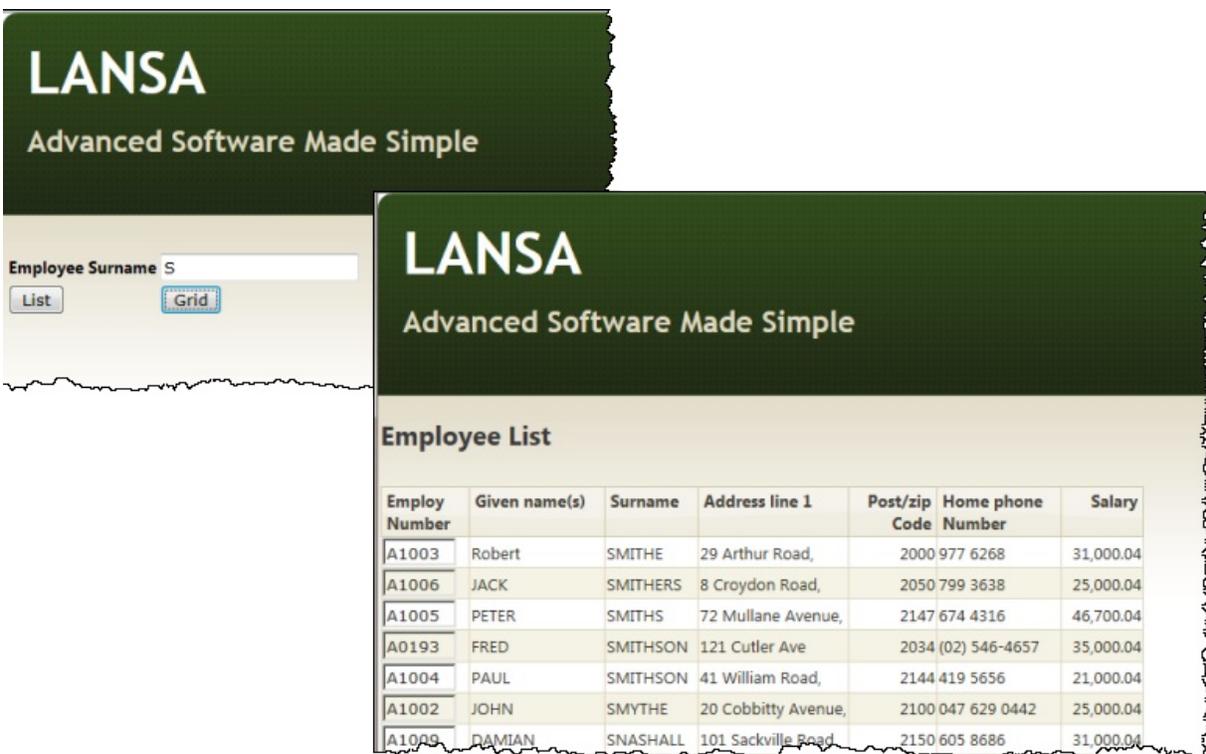
1. Create a new WAM

Name: **iiiUsingCSS**

Description: **Using CSS**

Layout weblet: **iiilay01**

Replace the default RDMLX code with the source provided in [WAM100. Appendix](#). It is a simple WAM that looks like the following. It displays a list of employees as a browse list or a grid.



2. Compile your WAM. Open the **begin** WebRoutine in the design view:

- Extend the table containing Employee Surname by adding a row.
- Add a push button to each new cell and set up the push button properties as follows:

| | |
|-----------------|----------------|
| Property | Value |
| Caption | List |
| on_click_wname | Emplist |

Caption **Grid**

on_click_wname **empgrid**

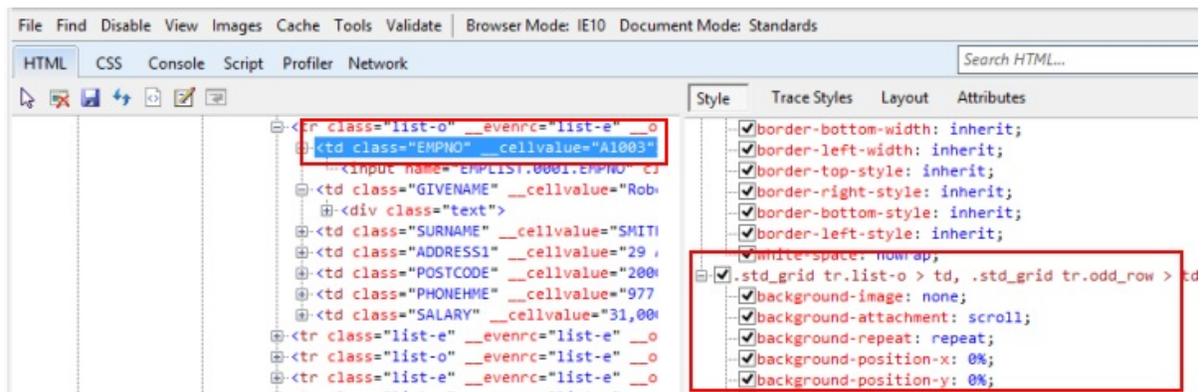
Save your changes.

3. Open the **empgrid** WebRoutine in the *Design* view. Add a *Grid* weblet to the page and link it to list EMPLIST.
4. Save your changes.
5. Run the **begin** WebRoutine in the web browser.
 - a. Enter a partial surname such as "S" or "B"
 - b. Click the *List* push button to display a list of employees.
 - c. Click the *Grid* push button to display a grid of employees.

Step 2. Create new Style Sheet

In this step you will create a style sheet to control the background color for odd and even rows in a list. To do this you first need to understand how the background color is controlled at the moment.

1. Execute WAM iiiUsingCSS in the browser and run *IE Developer Tools (F12)*.
 - a. Select the "Select" tool by clicking on the  on the toolbar.
 - b. Click anywhere on a list row.
 - c. Using the *HTML* view, select a TD tag, and scroll down the Style view to find the CSS that controls the background.



The CSS selector **.std_grid TR.list-o > TD** applies to all TD tags within an odd row, in a list, which is a table with a class of std_grid. The CSS is structured this way, due to the need for cross browser compatibility.

2. In Notepad use *File/New* to start a **new** document and add the following code.

```
.std_grid tr.list-o > td
{
    background-color: #fffacd;
}
.std_grid tr.list-e > td
{
    background-color: #ffe4b5;
}
```

3. Save your style sheet as **iii_style.css**, where **iii** are your initials. Use *Save as*

Type: All Files to save the file with the css extension.

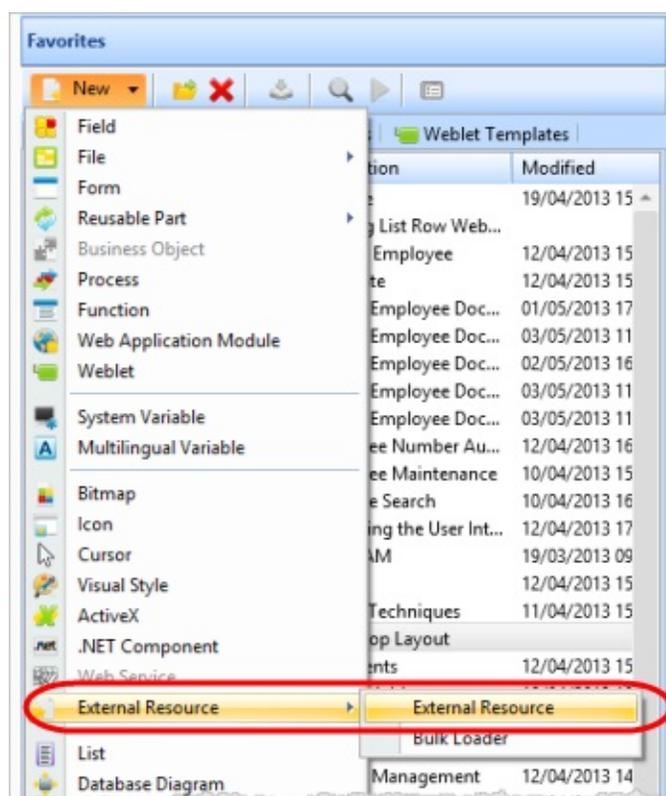
Make sure you save it to: **C:\Program
Files\LANSA\WebServer\Images\style.**

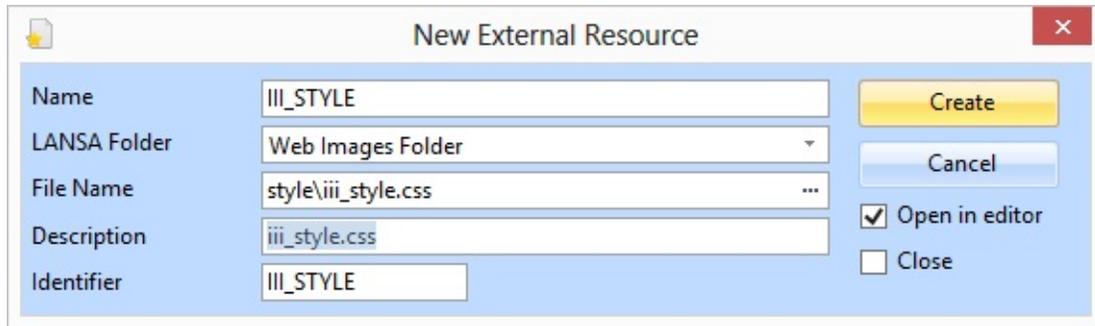
Leave Notepad open.

Step 3. Create an External Resource

- In this step you will define an *External Resource* using this style sheet and initially apply this to your WAM layout for iiiUsingCSS only.
- This will mean that style sheet iii_style.css will be applied only to all WebRoutines for the WAM iiiUsingCSS.
- In a later step you will apply the *External Resource* to the common layout **iiilay01**. You will see how it is then applied to all WAMs, sharing this common layout.
- In a later step you will also change your style sheet so that it targets only lists with an id of EMPLIST.

1. On the *Favorites* tab, select *External Resource / External Resource* from the *New* button.





2. Complete the details as shown in the table, but note the following steps:
 - a. Begin by selecting the *File Name* using the *Ellipsis* button and the *Open* dialog to select the new style sheet you saved in the `\style` folder. The *LANSA Folder* and *Description* will also be automatically completed.
 - c. Uncheck the options to *Open in the Editor*, check the *Close checkbox*.

Name **III_STYLE**

LANSA Folder **Web Images Folder**

Filename **Style\iii_style.css**

Description **iii_style.css**

- d. Click *Create* to save your *External Resource* definition.

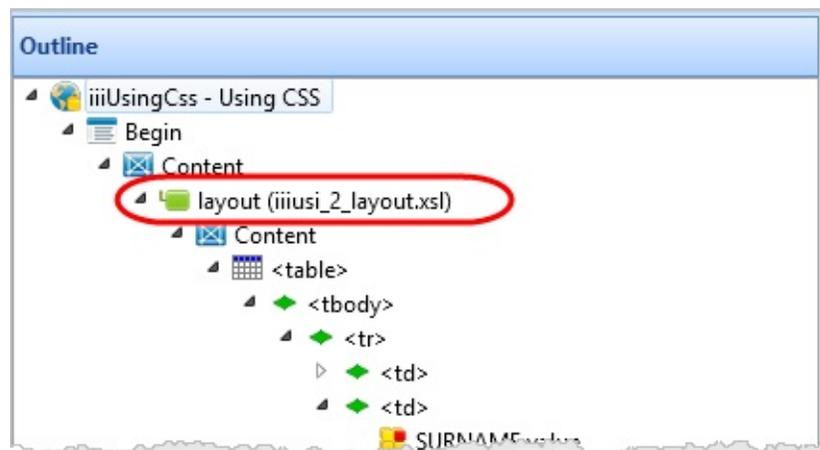
You have created an External Resource entry in the Repository, which can now be used to apply this style sheet to a layout.

Step 4. Apply Style Sheet to WAM `iiiUsingCSS`

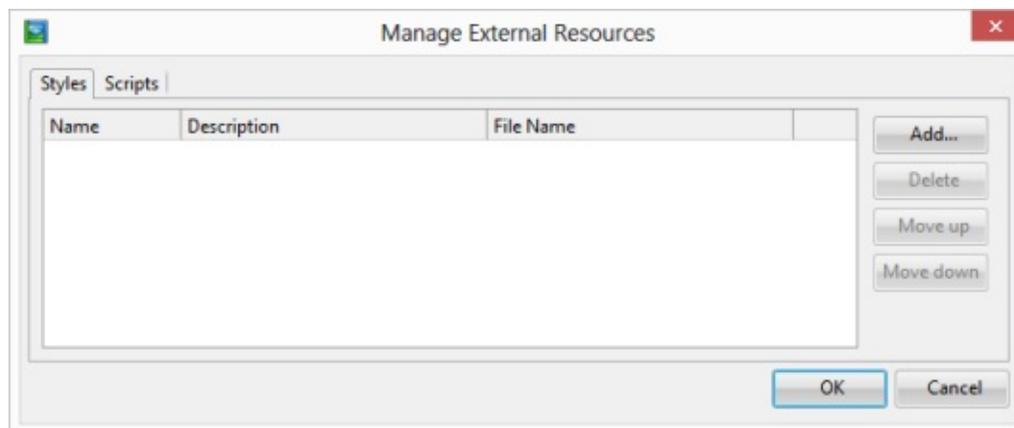
In this step you will open your WAM layout in the editor and add your *External Resource* to this layout. This will apply the style sheet `iii_style.css` to all WebRoutines in this WAM only.

1. On the *Outline* tab, open your WAM layout in the *Design* view by double clicking on the WAM layout item.

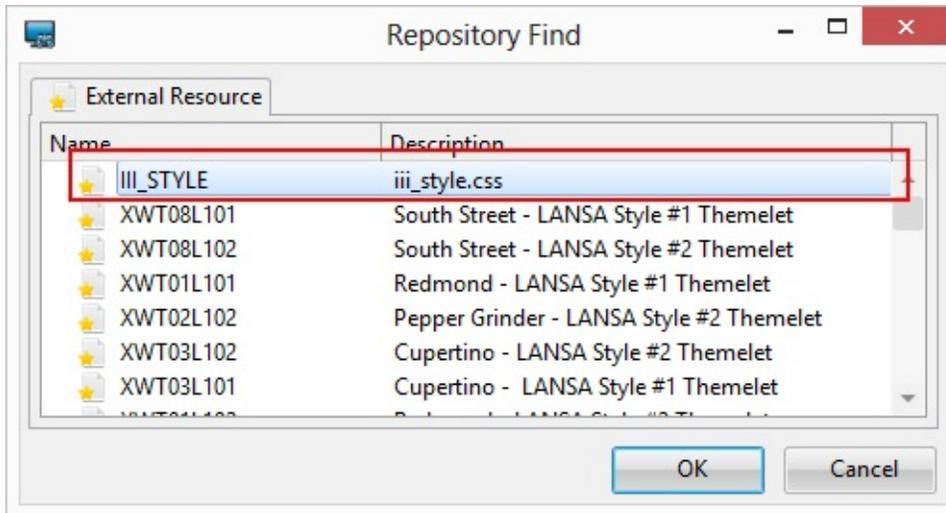
Note: The WAM layout is named using the WAM Identifier for example `iiiusi_2_layout.xml`. Your name may be slightly different.



2. Select the *Design* ribbon and click the *External Resources* button, to open the *Manage External Resources* dialog



3. Click the Add button, select your external resource and click *OK*. Click *OK* to close the *Manage External Resources* dialog.



4. Save the WAM layout.

The editor has added an entry to the WAM layout XSL to apply the style sheet defined in this external resource to the WAM layout.

```
<xsl:import href="iiilay01.xsl" />
<xsl:import href="std_types.xsl" />
<xsl:output method="xml" omit-xml-declaration="yes" indent="no" />
<wd:external-resources>
  <wd:style name="III_STYLE" />
</wd:external-resources>
<wd:definition>
  <wd:group name="WAM Layouts" />
</wd:definition>
```

5. Close the WAM layout.

6. Execute the WAM **iiiUsingCSS**, WebRoutine **begin** in the browser.

a. Enter a partial surname and display the List page for the employees

Employee List

| Employ Number | Given name(s) | Surname | Address line 1 | Post/zip Code | Home phone Number | Salary |
|---------------|----------------|---------|-------------------|---------------|-------------------|-----------|
| A1031 | JOHN | BLAKE | 3 Woodbury Road | 2100 | (02) 9668 9235 | 60,725.00 |
| A0090 | FRED JOHN ALAN | BLOGGS | 70 MAIN STREET | 2220 | 344-2234454545 | 20,045.91 |
| A3564 | FREDDY | BROWN | 121 SMITH STREET | 2153 | (02) 567-6758 | 30,000.00 |
| A0070 | VERONICA | BROWN | 12 Railway Street | 2153 | (02) 9609 4627 | 50,125.00 |

- b. Use the browser back button to return to the begin page, enter a partial surname and display the Grid page:

Employee Grid

| Employ Number | Given name(s) | Surname | Address line 1 | Post/zip Code | Home phone Number | Salary |
|---------------|----------------|---------|-------------------|---------------|-------------------|-----------|
| A1031 | JOHN | BLAKE | 3 Woodbury Road | 2100 | (02) 9668 9235 | 60,725.00 |
| A0090 | FRED JOHN ALAN | BLOGGS | 70 MAIN STREET | 2220 | 344-2234454545 | 20,045.91 |
| A3564 | FREDDY | BROWN | 121 SMITH STREET | 2153 | (02) 567-6758 | 30,000.00 |
| A0070 | VERONICA | BROWN | 12 Railway Street | 2153 | (02) 9609 4627 | 50,125.00 |

Note: The cascading style sheet has not been applied to the grid. At the moment it defines alternate background colors for a list only.

Step 5. Apply External Resource to the Common Layout

In this step you will open the common layout `iiilay01` in the editor and apply your *External Resource* to this layout and test the results.

1. Open the layout **`iiilay01`** in the editor. You could do this by opening it directly from the *Last Opened* tab on the *Favorites* tab, or by locating it on the *Repository* tab under *Weblets*.
2. With the common layout **`iiilay01`** open in the editor, as before select the *Design* ribbon and use *External Resources / Manage External Resources* dialog to *Add* the external resource `III_STYLE` to the layout.
3. Save the common layout and close it.
4. Open the WAM layout for WAM **`iiiUsingCSS`** and remove the external resource from this layout.
5. Close the WAM layout.
6. Execute the **`begin`** WebRoutine in the browser for WAM **`iiiUsingCSS`**. You should obtain the same results as before. Your style sheet for list alternate rows is applied to the list on the List page and is not applied to the grid on the Grid page.
7. Open the WAM **`iiiEmpSearch`** in the editor and execute the **`Search`** WebRoutine in the browser. Enter suitable employee numbers to display a list of employees. Note that the new cascading style sheet has been applied to the list.

All WAMs which were defined using the common layout **`iiilay01`** will have the new cascading style sheet applied. Any web page containing a list will have the CSS applied giving new alternate row background colors.

Step 6. Make the Style Sheet specific to lists named EMPLIST

As currently defined, the background color changes in your cascading style sheet, apply to all lists.

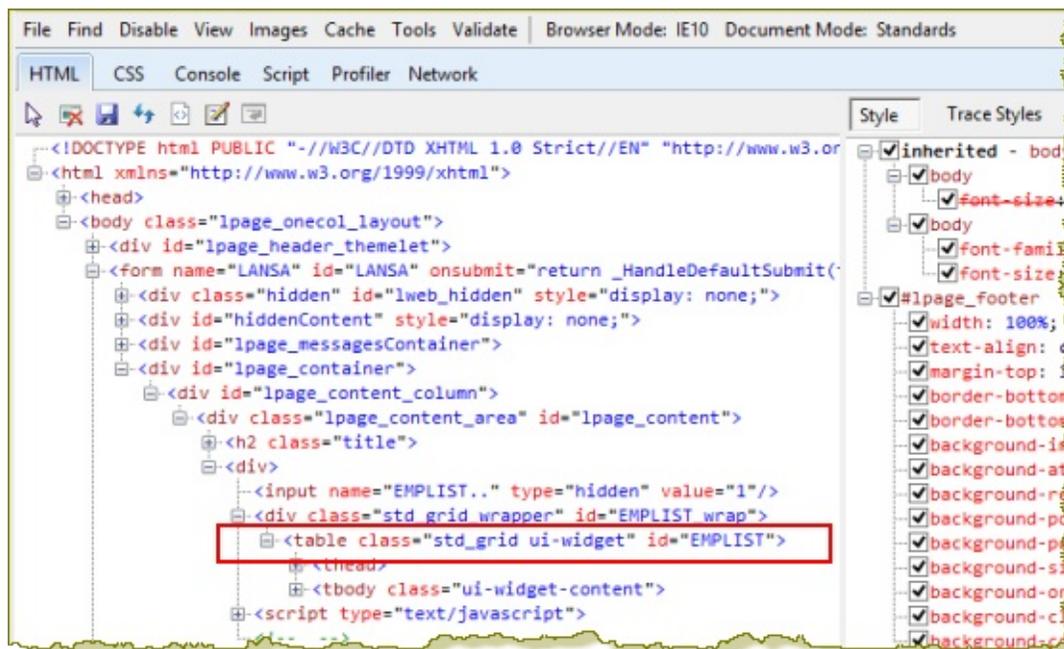
In this step you will make the style sheet specific to lists named EMPLIST.

1. Run the **begin** WebRoutine in the browser for WAM iiiUsingCSS and display a list of employees.

Use the IE Developer tools, as before to select the List (click on the edge of the List).

Expand the tree on the HTML tab, to show the <table....> tag for the list table.

Note this has an *id* of **EMPLIST**. It is given an *id* equal to the list name defined in the RDML.



2. Switch to Notepad, where you should still have your style sheet file open. Change the code to the following and save the file:

```
TABLE#EMPLIST tr.list-o > TD
{
    background-color: #fffacd;
```

```
}  
TABLE#EMPLIST tr.list-e > TD  
{  
  background-color: #ffe4b5;  
}
```

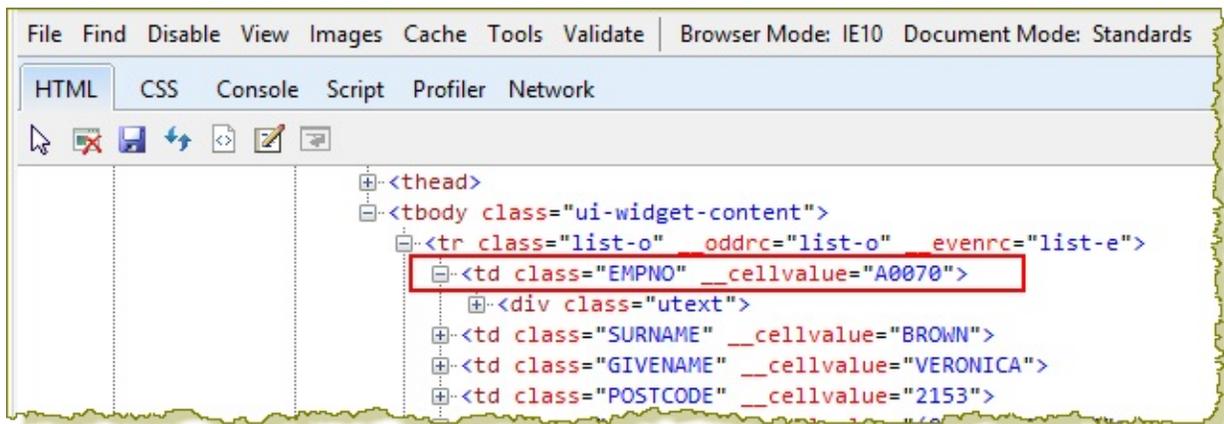
Your styles for odd and even background colors are now defined for lists named EMPLIST only.

11. Run WAM iiiUsingCSS and display the list of employees, which should reflect your style sheet.
12. Run WAM iiiSecMaint to display sections for a department. This list should not reflect your stylesheet, which is now specific to a list with an *id* of **EMPLIST**.

Step 7. Highlight a Column

In this step, you will investigate how a specific column can be identified and then create style sheet entry to change the background color for this column only.

1. Run the WAM `iiiUsingCSS` to display a list of employees. Using *IE Developer Tools* select the first input field in the first column (Employee Number).



Note that the `<td>` tags for each column have a class equal to the field name. In the first column this is **EMPNO**.

2. Add this code to your style sheet and save the changes:

```
TABLE#EMPLIST tr.list-o > TD.EMPNO
{
  background-color: #d78700;
}
TABLE#EMPLIST tr.list-e > TD.EMPNO
{
  background-color: #d78700;
}
```

This will override the background color for odd and even rows in table cells with a class of **EMPNO**.

3. Run your WAM `iiiUsingCSS` and display a list of employees. Your results should look like the following:

Employee List

| Employ Number | Given name(s) | Surname | Address line 1 | Posi
C |
|---------------|----------------|---------|-------------------|-----------|
| A1031 | JOHN | BLAKE | 3 Woodbury Road | |
| A0090 | FRED JOHN ALAN | BLOGGS | 70 MAIN STREET | |
| A3564 | FREDDY | BROWN | 121 SMITH STREET | |
| A0070 | VERONICA | BROWN | 12 Railway Street | |

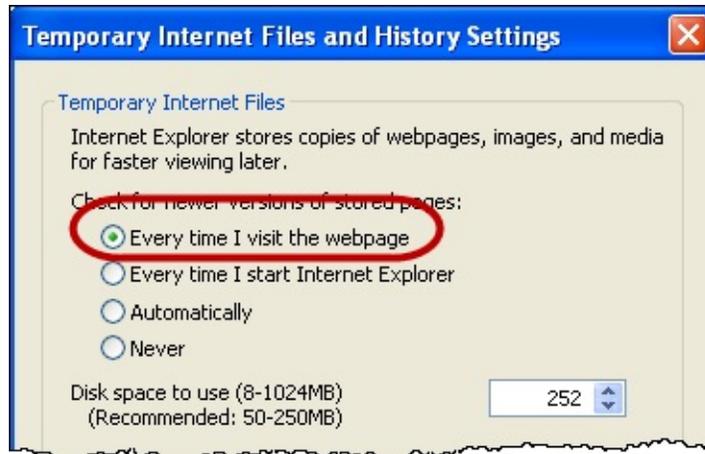
Summary

Important Observations

- You should be able to extend this exercise to make other changes to your list's appearance – for example borders, or to other elements on your web pages.
- As long as you are applying a custom CSS over the default, you can't break anything, so feel free to experiment.
- Here you have used a supplied themelet to set the overall appearance of your application. You may want to create your own themelet which replaces the supplied examples.

Tips & Techniques

- If you have not worked with CSS before, take a look at the tutorials at www.w3schools.com
- You may also want to take a look at these articles on CSS Selectors: http://www.456bereastreet.com/archive/200509/css_21_selectors_part_1/
- With the exception of the `tr.list-o` and `tr.list-e` styles shown earlier, the default selectors for most grid related styles start with the `.std_grid` class selector. This makes them easier to find in the CSS file and reduces the chances of accidental conflicts with styles used elsewhere (the `tr.list-o` and `tr.list-e` are defined as they are for backwards compatibility reasons).
- In the event of a conflict, the style with the more specific selector will take priority. For example, the default style for grid table cells is defined with `".std_grid tbody td"`. It will override any conflicting properties defined with `".std_grid td"`. So, if a style isn't working as expected, try making it more specific
- When working on web application development ensure that your browser settings check for newer versions of stored pages "Every time I visit the webpage".



- When changing entries in a style sheet, be aware that you may have problems with cached versions. Clear your browser history regularly.

What You Should Know

- The essential rules for creating and applying style sheets
- How to use the IE Developer Tool to understand the construction of your web pages at a detailed level.

WAM100. Appendix

Use the following RDMLX source code to create iiiUsingCSS in Step 1, of this exercise.

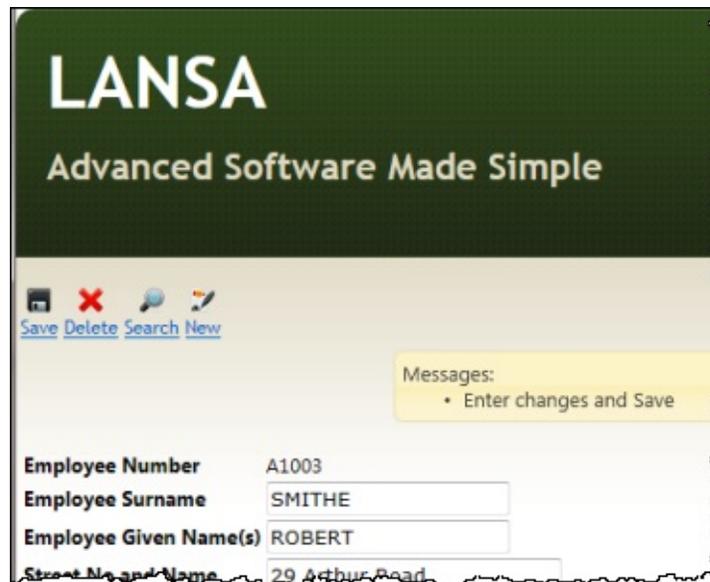
```
Def_List Name(#emplist) Fields(#empno (#givenname *out) (#surname *out) (#
Define Field(#hidesave) Type(*char) Length(1)
Define Field(#hidedel) Type(*char) Length(1)
Define Field(#hidenew) Type(*char) Length(1)
Define Field(#hidesrch) Type(*char) Length(1)
Define Field(#empnow) Reffld(#empno)
Web_Map For(*both) Fields((#stdrentry *hidden) (#empnow *hidden))
WebRoutine Name(Begin)
Web_Map For(*output) Fields(#surname)
#hidedel #hidesave #hidesrch := Y
Endroutine
WebRoutine Name(empgrid) Desc('Employee Grid')
Web_Map For(*input) Fields(#surname)
Web_Map For(*output) Fields((#emplist *private))
#hidedel #hidesave := Y
Execute Subroutine(bldlist)
Endroutine
WebRoutine Name(emplist) Desc('Employee List')
Web_Map For(*input) Fields(#surname)
Web_Map For(*output) Fields(#emplist)
#hidedel #hidesave := Y
Execute Subroutine(bldlist)
Endroutine
Subroutine Name(bldlist)
Clr_List Named(#emplist)
Select Fields(#emplist) From_File(pslmst2) With_Key(#surname) Generic(*ye
Add_Entry To_List(#emplist)
Endselect
Endroutine
```

WAM105 - Create Your Own Weblet

Objectives

To create Toolbar Menu Item weblet and a Toolbar weblet. A simple WAM application will then be used to test the Toolbar weblet.

The finished application will look like the following:



In order to complete this exercise, you must complete the following:

- Review [Weblets](#)
- [Step 1. Create Toolbar Menu Item Weblet](#)
- [Step 2. Create Toolbar Weblet](#)
- [Step 3. Complete Definition of Toolbar Menu Item Weblet](#)
- [Step 4. Setup iii_toolbar_menuitem Properties in iii_toolbar](#)
- [Step 5. Apply Toolbar Weblet to an Employee Maintenance WAM](#)
- [Summary](#)

Before You Begin

This exercise does not depend on knowledge gained from all preceding exercises. It is recommended that the following have been completed:

[WAM005 - Create Your First WAM](#)

[WAM010 - Using WEB_MAPs](#)

[WAM015 - Working Lists](#)

WAM020 - WAM Navigation

Weblets

Layouts are a type of Weblet. They allow you to customize the overall look and feel of a web site or web application.

Weblets are also building blocks of your WAM HTML page. They can be categorized into 2 main groups.

- Primitive Weblets typically have a 1-to-1 relationship with an HTML Tag/Element (eg: Push Button, Checkbox, Combo Box, Anchor, Clickable Image, an Input box, etc)
- Composite Weblets provide additional functionality combining Javascript, CSS, Primitive Weblets and HTML (eg: Grid, Tabsheets, etc)

XSL Templates

The `<xsl:template name="my_template_name"> </xsl:template>` element is very similar to the SUBROUTINE command used to define Subroutines in LANSAs.

Parameters can be received by a template by defining

`<xsl:param name="param_name" />` elements within the template.

Calling XSL Templates

The `<xsl:call-template name="layout-form.private">` element is very similar to the EXECUTE command used to call subroutines in LANSAs.

Parameters can be passed on the call using the

`<xsl:with-param name="param_name" select="param_value"/>` element.

To learn more about XSLT, see <http://www.w3schools.com/xsl/default.asp>

Step 1. Create Toolbar Menu Item Weblet

In this step you will create a Toolbar Menu Item from which the toolbar will be built. If you have some basic HTML knowledge, then you will know that a menu item is essentially an `<a>` anchor tag that can have an href, image and alternate text elements etc associated with it. You should also know that in a WAM application an "href" will usually call a JavaScript function to call a WAM / WebRoutine.

1. Create a new weblet select *New / Weblet* in the Visual LANSA Editor.
 - a. In Name and Description, replace iii with your initials.



- b. Select **Custom Weblets** as the *Weblet Group*.
- c. Press *Create* to create your weblet and the **Custom Weblets** group. The weblet will open in the *Design* view.

The XSL Source for your weblet should currently look like the following:

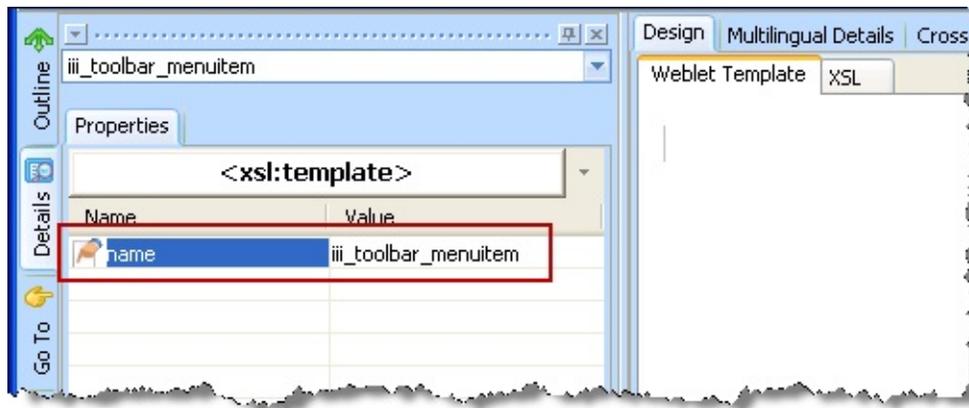
```

Design | Multilingual Details | Cross References
Weblet Template | XSL | iii_toolbar_menuitem [LANSA]
<?xml version="1.0" encoding="UTF-8"?>
<!-- (c) 2003, 2011 LANSA -->
<!-- XHTML weblet -->
<!-- $Revision:: 3 $ -->
<xsl:transform version="1.0" exclude-result-prefixes="lxml wd tsml"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:lxml="http://www.lansa.com/2002/XML/Runtime-Data"
  xmlns:wd="http://www.lansa.com/2002/XSL/Weblet-Design"
  xmlns:tsml="http://www.lansa.com/2002/XML/Generation-Metadata"
  xmlns="http://www.w3.org/1999/xhtml">
  <xsl:import href="std_types.xsl" />
  <xsl:output method="xml" omit-xml-declaration="yes" encoding="UTF-8"
    indent="no" />
  <wd:external-resources />
  <wd:definition>
    <wd:use-design-layout />
    <wd:group name="Custom Weblets" />
  </wd:definition>

  <xsl:template name="unnamed">
    <!-- Give your template appropriate name and type in your XSL here -->
  </xsl:template>
</xsl:transform>

```

2. Select the *Weblet Template* tab and use the *Details* tab to enter a *name* of **iii_toolbar_menuitem**. Replace **iii** with your initials.



Save your changes.

3. You will now add the skeleton HTML code for the anchor tag, inside the `<xsl:template>...</xsl:template>`, as follows:

Copy the following code `<a href="jav..... code and paste it immediately following the lines:`

New code is highlighted in red and italic.

```
<xsl:template name="iii_toolbar_menuitem">
  <!-- Give your template an appropriate name and type in your XSL here --
>
<a href="javascript:">    
    Menu Text
  </span>
</a>
```

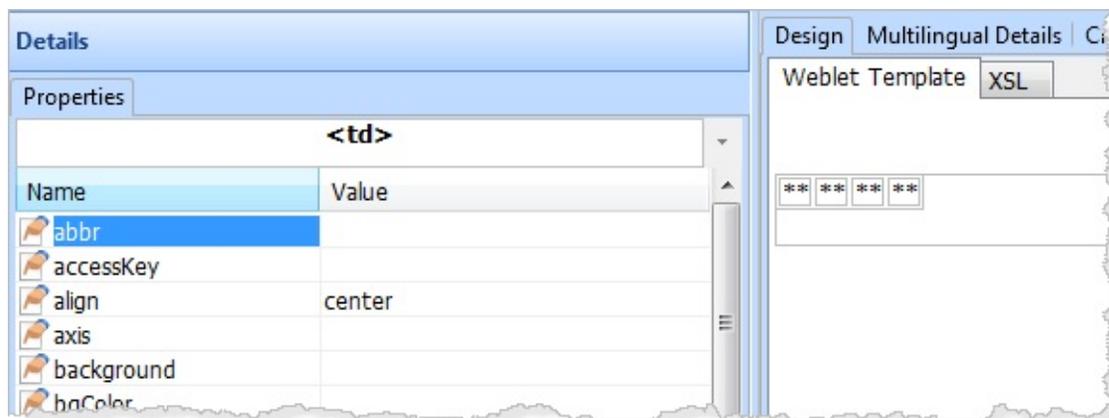
In later steps you will complete this outline for the anchor tag code.

4. Click on the Save  button on the editor *Toolbar* to save your changes.

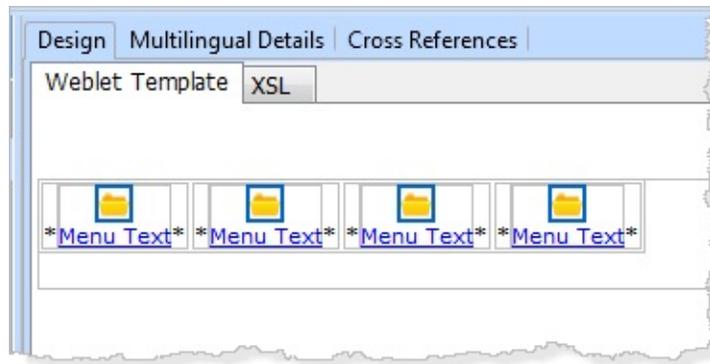
Step 2. Create Toolbar Weblet

In this step you will create a second weblet to build your Toolbar weblet. This simply consists of a single row table with 4 cells.

1. As before, create a new weblet using *New / Weblet*, *Name* = **iii_toolbar**, *Description* = **Toolbar**, *Group* = **Custom Weblets**. Replace **iii** with your initials.
2. On the *Details* tab change the weblet *name* to **iii_toolbar**.
3. In the *Design* view use the context menu to *Insert HTML / Table* with 1 row and 4 columns.
4. Save your changes.
5. Select each cell in the toolbar table and use the *Details* tab to set its *align* property to **center**.



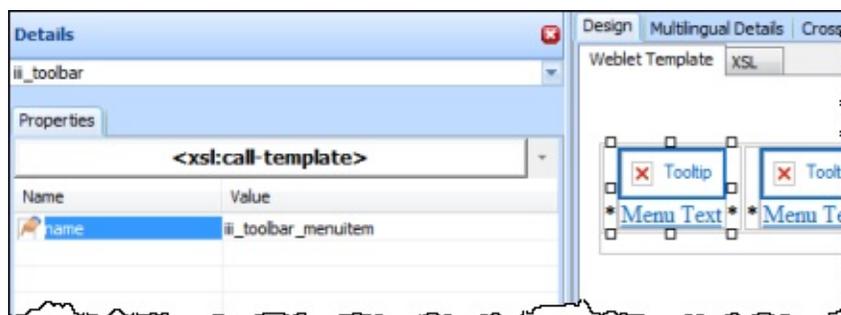
6. Switch to the *Favorites / Weblet Template* tab, find your *iii_toolbar_menuitem* weblet. First select **Custom Weblets** in the dropdown. If necessary use the *View / Refresh Repository* menu option to refresh the view in the Weblets Template tab. Drag and drop a *iii_toolbar_menuitem* into each of the four cells in the toolbar. Your toolbar should now look like the following in the Weblet Template design view.



7. Use the cursor keys to position into each cell and delete the * place holder characters.

Save your changes.

8. Select one of the toolbar menu items and select the *Details* tab. Notice that at present, the menu items do not have any properties that can be set.



Step 3. Complete Definition of Toolbar Menu Item Weblet

In this step you will complete the coding of the toolbar menu item weblet. You will add code to:

- Define weblet parameters
- Define elements of the anchor tag (for example, IMG ALT) as XSL variables
- Complete JavaScript for HREF to set reentry field value and call wam / WebRoutine
- Condition the <A> HREF and IMG tags based on a \$hide_if variable
- Add weblet parameter tooltips.

1. Switch to the iii_toolbar_menuitem or open it in the editor if necessary. Select the XSL tab.
2. Immediately following the **<xsl:template name="iii_toolbar_menuitem">** tag, paste in the following code to define the weblet parameters. Review the comments for each parameter to see where it will be used.

```
<!-- Used to set the Menu Text on the toolbar image -->
  <xsl:param name="menu_text" wd:type="std:mtxt_variable" select="Capt
  <!-- Used to set the image use for the toolbar Icon -->
  <xsl:param name="menu_image" wd:type="std:html_img_relative"
    select="/icons/normal/16/folder_16.png" />
  <!-- Used to set the ALT tag on the toolbar IMG tag -->
  <xsl:param name="tooltip_text" wd:type="std:mtxt_variable"
    select="Caption" />
  <!-- Used to set the Rentry Field Name when the toolbar Icon is clicked --
  >
  <xsl:param name="reentryfield"
    wd:type="std:field_name_in[wam=$on_click_wamname]
    [webrtn=$on_click_wrname]"
    select="STDREENTRY" wd:tip_id="" />
  <!-- Used to set the Rentry Field Value when the toolbar Icon is clicked-->
  <xsl:param name="reentryvalue" select="M" wd:tip_id="" />
  <!-- Used to set the Menu Text on the toolbar image -->
  <xsl:param name="hide_if" wd:type="std:boolean" select="false()"
    wd:tip_id="" />
  <!--
```

```

- Used to specify the WAMNAME to call when toolbar Icon is clicked -->
  <!-- It will default to the current WAM if no value is specified -->
  <xsl:param name="on_click_wamname" wd:type="std:wam"
    select="/lxml:data/lxml:context/lxml:webapplication" wd:tip="" />
  <!-- Used to specify the WebRoutine to call when toolbar Icon is clicked --
>
  <xsl:param name="on_click_wrname"
    wd:type="std:webroutine[wam=$on_click_wamname]" wd:tip="" />

```

This block of code defines the *parameters* that can be passed into the template when the toolbar menu item template is called, in a similar way to calling a subroutine. Once defined these become the *properties* that can be set in the *Design* view for a web page that uses this weblet.

For example: a parameter, named *menu_text*, has a default value of '*Caption*'. In the completed anchor tag code following, note that the variable **\$menu_text** is used as the caption text below the toolbar item image.

3. Save your changes.

4. You will now complete the code for the <A> anchor tag. Copy the following code and paste it to **replace** the skeleton code for the <A> tag, which you placed there earlier.

```

<a href="javascript:InsertHidden(document.LANSA,'{$sreentryfield}','{$sreentr
  
  <br />
  <span class="std_menuitem">
    <xsl:value-of select="$menu_text" />
  </span>
</a>

```

Note the following points about these changes:

- The value for the alt tag has been replaced with a variable \$stooltip_text
- The value for image file name in the src tag has been replaced with a variable \$menu_image
- The menu text inside the span tag is now generated by an <xsl:value-of which outputs the value of variable \$menu_text
- The href for the A tag has been defined with JavaScript code that passes

\$reentryfield and \$reentryvalue variables to the InsertHidden function

- The href code also runs the HandleEvent function passing \$on_click_wamname and \$on_click_wrname variables.

5. Save your changes

6. In this step you will add xsl code to condition the anchor tag, based on the **\$hide_if** parameter.

Note: The `<xsl:if>` element must have an `</xsl:if>` end tag. The `<xsl:if>` must surround the entire `<A HREF>` tag. Add the highlighted code only:

```
<xsl:if test="not($hide_if)">
  <a href="javascript: . . . . .
  </a>
</xsl:if>
```

Hint: The XSL editor autocomplete function will generate the `</xsl:if>` when you complete the beginning tag. Move this to the required position after the `` tag.

The variable **\$hide_if** is a Boolean, with a default value of 'false' as shown in the parameter definitions.

7. In this step you will add Weblet Parameter tooltips by copying in the following code, following the `</wd:definition>` end tag.

```
<wd:template name="iii_toolbar_menuitem">
  <wd:description icon="icons/userdefn.ico">
    <wd:name lang="ENG">iii Toolbar Menu Item</wd:name>
  </wd:description>
  <wd:param name="menu_text">
    <wd:tip lang="ENG">Menu Text to display below the image on the toolt
  </wd:param>
  <wd:param name="menu_image">
    <wd:tip lang="ENG">Image to display on the toolbar menu item</wd:tip
  </wd:param>
  <wd:param name="tooltip_text">
    <wd:tip lang="ENG">Tooltip text to display on the toolbar menu item</\
  </wd:param>
</wd:template>
```

Correct <the wd:template name="iii_toolbar_menuitem"> replacing iii with your initials.

Note:

- Tags such as **<wd:template name="iii_toolbar_menu_item">** have a namespace of **wd**. They are LANSAs defined weblet design tags, defined by the standard that is referenced at the top of the weblet XSL. See: **xmlns:wd=http://www.lansa.com/2002/XSL/Weblet-Design**.
- Standard XSL tags have a namespace of **xsl**, for example, **<xsl:if.....>**
- The **<wd:template</wd:template>** code is used by the Design view to define the icon and description used in the list of weblets, and the tooltip text for the weblet parameters.

8. Look towards the top of your toolbar XSL to find the statement:

```
<xsl:import href="std_types.xsl" />
```

Add the following line:

```
<xsl:import href="std_keys.xsl" />
```

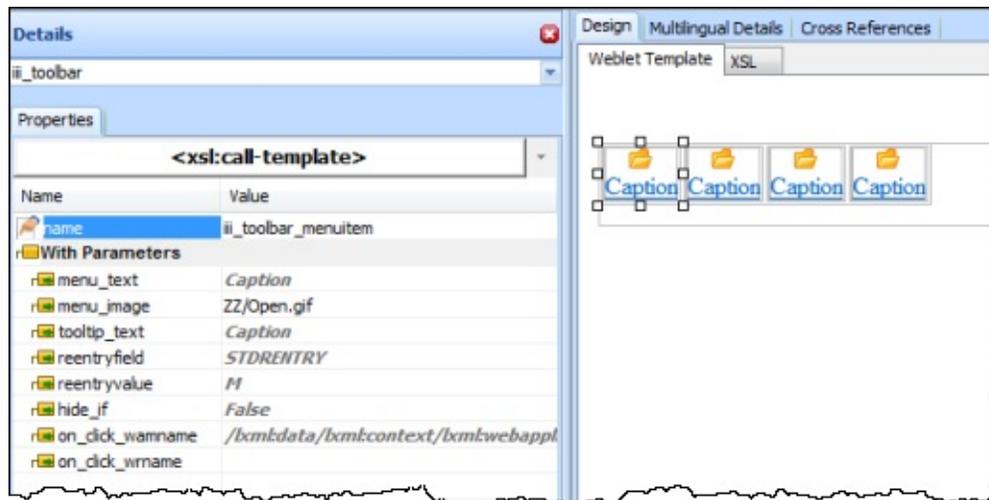
The **std_keys** XSL defines **xsl:key**'s such as "field-caption" and "field-value" that are used during transformation to extract data from the Data XML output via the WebRoutine.

9. Save your changes to the Toolbar Menu Item weblet.

Step 4. Setup iii_toolbar_menuitem Properties in iii_toolbar

In this step you will complete the definition of the toolbar by setting up the properties for each of the four menu items.

1. If necessary, open your iii_toolbar_menu in the editor.
2. In the Design view select the first menu item. The Details tab should now contain properties that can be set for each menu item.



3. Click on each of the 4 **Menu items** in order and set their **weblet properties** in the **Details** tab as follows:.

On the *Details Tab*, make sure to take note of the tooltip/help text provided for each weblet property. Some of these tooltips are shipped with LANSAs for standard weblet parameter types like WAMName and WebRoutine name but other custom properties like Menu Text also have tooltip text that was added to the weblet definition.

| <i>First Menu Item</i> | |
|------------------------|---------------------------------|
| <i>Property Name</i> | <i>Value</i> |
| menu_text | Save |
| menu_image | icons/normal/16/diskette_16.png |
| tooltip_text | Save a changed or new employee |

| | |
|------------------|--------------------------------|
| reentryfield | STDREENTRY (the default value) |
| reentryvalue | S |
| hide_if | #HIDESAVE='Y' |
| on_click_wamname | Leave as default (current WAM) |
| on_click_wrname | #WRNAME |

| | |
|-------------------------|--------------------------------|
| <i>Second Menu Item</i> | |
| <i>Property Name</i> | <i>Value</i> |
| menu_text | Delete |
| menu_image | icons/normal/16/cross_16.png |
| tooltip_text | Deletes the current employee |
| reentryfield | STDREENTRY (the default value) |
| reentryvalue | D |
| Hide_if | #HIDEDEL='Y' |
| on_click_wamname | Leave as default (current WAM) |
| on_click_wrname | Maint |

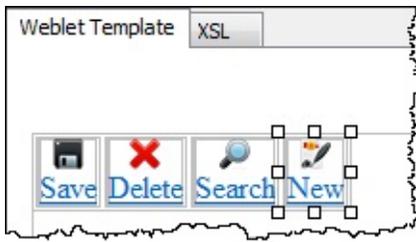
| | |
|------------------------|--------------|
| <i>Third Menu Item</i> | |
| <i>Property Name</i> | <i>Value</i> |
| menu_text | Search |

| | |
|------------------|--------------------------------|
| menu_image | icons/normal/16/zoom_16.png |
| tooltip_text | Switch to the Search web page |
| reentryfield | STDREENTRY (the default value) |
| reentryvalue | M |
| Hide_if | #HIDESRCH='Y' |
| on_click_wamname | Leave as default (current WAM) |
| on_click_wrname | Begin |

| | |
|-------------------------|--|
| <i>Fourth Menu Item</i> | |
| <i>Property Name</i> | <i>Value</i> |
| menu_text | New |
| menu_image | icons/normal/16/contract_16.png |
| tooltip_text | Switch to the create new employee web page |
| reentryfield | STDREENTRY (the default value) |
| reentryvalue | N |
| hide_if | #HIDENEW='Y' |
| on_click_wamname | Leave as default (current WAM) |
| on_click_wrname | New |

Use the Ellipsis button to find an image for the menu_image value.

4. Save your changes. You have now completed the definition of your toolbar weblet, which should look like the following:



Step 5. Apply Toolbar Weblet to an Employee Maintenance WAM

In this step you will create a new WAM based on supplied RDMLX code, set up the web page for each WebRoutine and then add the toolbar to the WAM layout.

1. Create a new WAM **iiiEmpMaint_TB – Employee Maintenance with Toolbar**, using Weblet Template **iiilay01**, based on the code supplied in [WAM 105. Appendix](#).
2. Compile the WAM.
3. Examine the WAM code and note the following:
 - Fields HIDEDEL, HIDESRCH, HIDENEW, HIDESAVE are globally mapped as hidden fields – these fields control whether the tool bar icons are shown.
 - Field WRNAME is also mapped globally as a hidden field. This is used to control which WebRoutine the Save button calls.
 - The WebRoutines set the value of the "HIDExxx" fields as appropriate. For example the Begin WebRoutine shows only the New toolbar icon.
4. Open the **begin** WebRoutine in the *Design* view, add a row to the table and add pushbuttons with a *caption* of **List** and **Grid** and with *on_click_wrname* values of **emplist** and **empgrid** respectively. These buttons do not require a *submitFieldValues* property.



5. Open the **emplist** WebRoutine in the *Design* view. Drop an *Anchor* weblet onto the employee number field in the first list column. Set up the *Anchor* weblet properties as shown:

| Property | Value |
|------------------|----------------|
| currentrowhfield | EMPNO |
| currentrownumval | \$EMPNO |
| Reentryvalue | M |

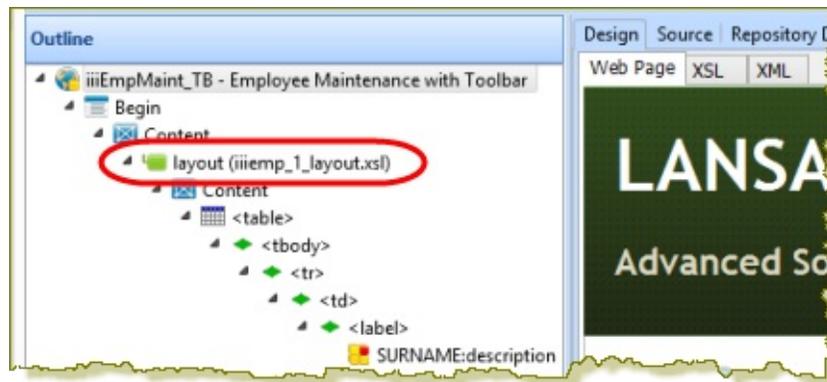
on_click_wname **maint**

Your list should now look like the following:

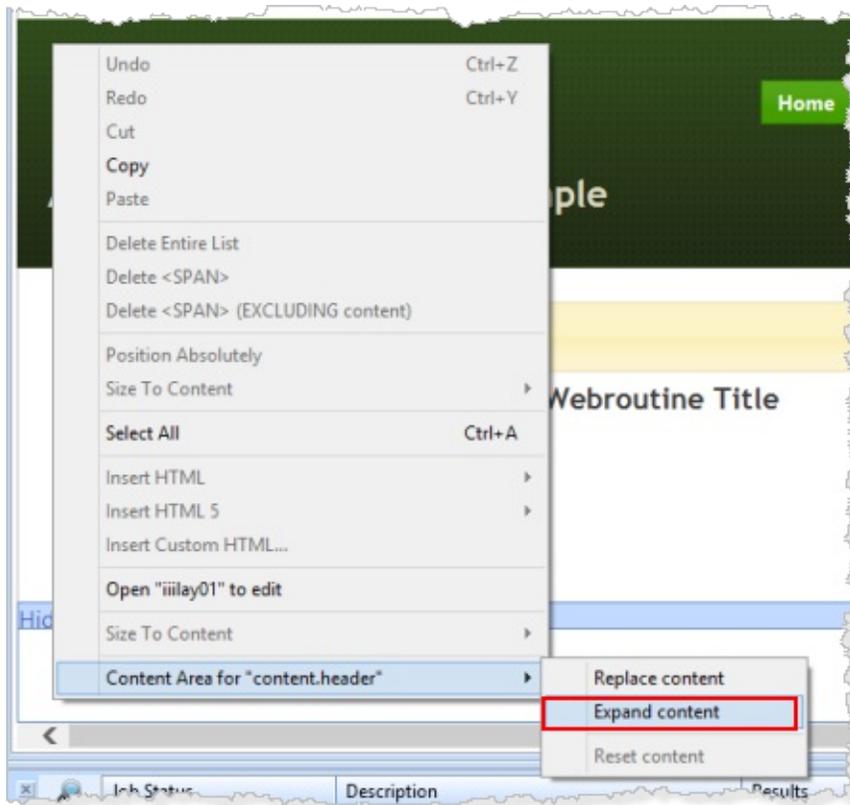
| Employ Number | Given name(s) | Surname | Dept Code | Section Code | Salary |
|---------------|----------------------|----------------------|--------------------|-----------------|----------------|
| Value EMPNO 1 | ABCDEFGHIJKLMNOPQRST | ABCDEFGHIJKLMNOPQRST | Value DEPARTMENT 1 | Value SECTION 1 | 123,456,789.12 |
| Value EMPNO 2 | ABCDEFGHIJKLMNOPQRST | ABCDEFGHIJKLMNOPQRST | Value DEPARTMENT 2 | Value SECTION 2 | 123,456,789.12 |
| Value EMPNO 3 | ABCDEFGHIJKLMNOPQRST | ABCDEFGHIJKLMNOPQRST | Value DEPARTMENT 3 | Value SECTION 3 | 123,456,789.12 |

6. Save your changes.
7. Select the  *Outline* tab. Double click on the layout **iiiempma_layout.xml** to open it.

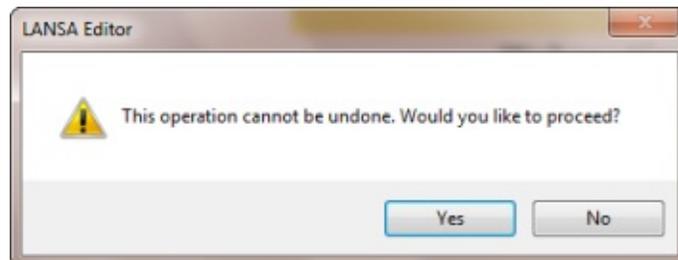
Note: The layout name uses the WAM **Identifier**, which may have a different value in your Visual LANSA.



8. With the wam layout open in the editor, click on the header area and using the context menu, select *Content Area for "content.header"* and *Expand content*.



In the *Confirm* dialog, click *OK* to confirm your change:



The layout will be redisplayed, with the change area highlighted in red.



9. Select the Add Content area. On the *Weblet Templates* tab, select *Custom Weblets* and drop your **iii_toolbar** weblet onto the layout, inside red **Add content** area. This is actually a span tag, with inline styles set, giving a red background and yellow text. Your layout will now look like the following:



10. Click on the Add content area to select it, and use the *Details* tab to clear its inline styles. This will remove the red background, yellow color and black borders. It also has a margin of 10px which could be retained, if required.

Delete the **** Add content** text from inside the span, leaving your just your toolbar weblet.

Your page should look like the following:



11. Save and close the wam layout. You have added the toolbar weblet into this WAM layout only. If it was required in all your WAMs you could have added it into your common layout, **iiilay01**.
12. Execute your WAM in the browser by running the **Begin** WebRoutine. Test the operation of the toolbar. The toolbar items will be hidden when not required. For example, the **Begin** WebRoutine will display only the *New* toolbar item.

Summary

Important Observations

- Once you understand the elements of a weblet's code, simple weblets can be developed quite easily.
- More complex weblets will require good XSL, HTML and possibly Javascript knowledge in order to design and create them.

Tips and Techniques

- Develop a weblet in small steps with frequent testing.
- It is often useful to add text between tags that will highlight areas in your prototype design.

What I should now know

- XSL tags with the wd namespace (for example, <wd:template name= . . . >) are LANSa weblet design tags which are used by the *Design* view.

WAM 105. Appendix

Use the following RDMLX source code to create iiiEmpMaint_TB in Step 5 of this exercise.

```
DEF_LIST NAME(#emps) FIELDS(#empno (#givename *out) (#surname *ou
DEFINE FIELD(#hidesave) TYPE(*char) LENGTH(1)
DEFINE FIELD(#hidedel) TYPE(*char) LENGTH(1)
DEFINE FIELD(#hidenew) TYPE(*char) LENGTH(1)
DEFINE FIELD(#hidesrch) TYPE(*char) LENGTH(1)
DEFINE FIELD(#empnow) REFFLD(#empno)
Define Field(#wrname) Type(*char) Length(50)
GROUP_BY NAME(#empmnt) FIELDS(#SURNAME #GIVENAME #ADDI
GROUP_BY NAME(#empadd) FIELDS(#EMPNO #SURNAME #GIVENAM
WEB_MAP FOR(*output) FIELDS((#hidesave *hidden) (#hidedel *hidden) (#
WEB_MAP FOR(*both) FIELDS((#stdrentry *hidden) (#empnow *hidden))
WebRoutine NAME(Begin)
WEB_MAP FOR(*output) FIELDS(#surname)
#hidedel #hidesave #hidesrch := Y
ENDROUTINE
WebRoutine NAME(empgrid)
WEB_MAP FOR(*input) FIELDS(#surname)
WEB_MAP FOR(*output) FIELDS((#emps *private))
#hidedel #hidesave := Y
EXECUTE SUBROUTINE(bldlist)
ENDROUTINE
WebRoutine NAME(emplist) DESC('Employee List')
WEB_MAP FOR(*input) FIELDS(#surname)
WEB_MAP FOR(*output) FIELDS(#emps)
#hidedel #hidesave := Y
EXECUTE SUBROUTINE(bldlist)
ENDROUTINE
WebRoutine NAME(maint)
WEB_MAP FOR(*both) FIELDS((#empno *out) #empmnt)
#hidedel #hidesave #hidesrch #hidenew := N
#wrname := 'maint'
CASE (#stdrentry)
WHEN (= S)
```

```

UPDATE FIELDS(#empmnt) IN_FILE(pslmst) WITH_KEY(#empnow) VAL_
IF_STATUS IS(*OKAY)
MESSAGE MSGTXT('Employee changed')
TRANSFER TOROUTINE(begin)
ENDIF
#EMPNO := #EMPNOW
WHEN (= D)
DELETE FROM_FILE(pslmst) WITH_KEY(#empnow) VAL_ERROR(*next)
IF_STATUS IS(*OKAY)
MESSAGE MSGTXT('Employee deleted')
TRANSFER TOROUTINE(begin)
ENDIF
#EMPNO := #EMPNOW
OTHERWISE
FETCH FIELDS(#empmnt) FROM_FILE(pslmst) WITH_KEY(#empno)
#empnow := #empno
MESSAGE MSGTXT('Enter changes and Save')
ENDCASE
ENDROUTINE
WebRoutine NAME(new)
WEB_MAP FOR(*both) FIELDS(#empadd)
#hidedel #hidenew := Y
CASE (#stdrentry)
WHEN (= N)
#empadd := *default
MESSAGE MSGTXT('Enter details and Save')
WHEN (= S)
INSERT FIELDS(#empadd) TO_FILE(pslmst) VAL_ERROR(*next)
IF_STATUS IS(*OKAY)
MESSAGE MSGTXT('New Employee added')
ENDIF
ENDCASE
#wrname := 'new'
ENDROUTINE
SUBROUTINE NAME(bldlist)
CLR_LIST NAMED(#emps)
SELECT FIELDS(#emps) FROM_FILE(pslmst2) WITH_KEY(#surname) GE
ADD_ENTRY TO_LIST(#emps)
IF (#listcount = 15)

```

```
MESSAGE MSGTXT('First 15 entries shown only')  
LEAVE  
ENDIF  
ENDSELECT  
ENDROUTINE
```

WAM110 - Create Your Own Layout Weblet

Objectives

As you have already seen, the Web Application Layout Manager Wizard enables you to create your own layout based on one of the supplied designs and themes. Bear in mind that the appearance of this style of these layouts could be considerably modified by simply changing the CSS associated with it. This standard layout can then be applied to each WAM you create.

You may require your WAM layouts to closely resemble your company web site standards and appearance. If you are building a business to consumer application, then this will certainly be the case. This exercise demonstrates how you can start from your own layout and embed this within a layout weblet so that it can be applied to your WAM application.

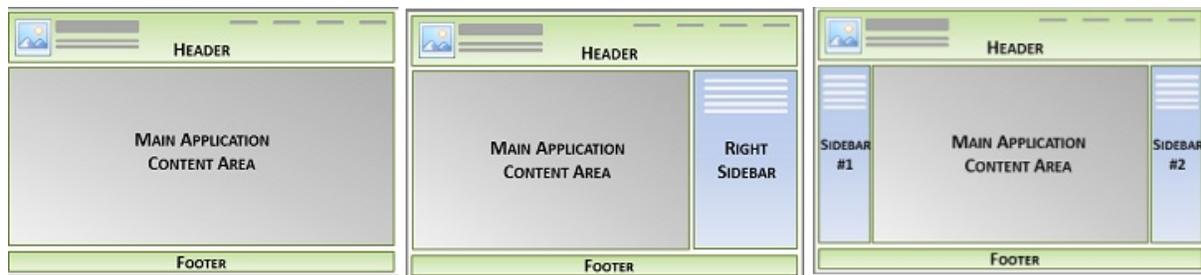
What is a Layout?

A WAM layout weblet is used to give structure to the web page associated with a webroutine and to interface with any documents referenced in the layout definition for functional or aesthetic values.

By default, each WAM has an associated WAM layout weblet, which is used as the basis for any presentation associated with the WAM's webroutines. A single WAM layout is generated for each WAM regardless of how many webroutines are defined within the WAM. If your web application includes multiple WAMs, the same layout can be applied to all the WAMs in your application. This way, you can guarantee a consistent interface.

As a visual element, a WAM layout typically provides the structure for any resulting web page. In this role, a WAM layout can define any titles, menus, message presentation or logos to be displayed. The WAM layout also controls the Cascading Style Sheet to be applied.

Your layout could have literally any appearance, but with transactional systems there will always be some kind of main content area, for example:



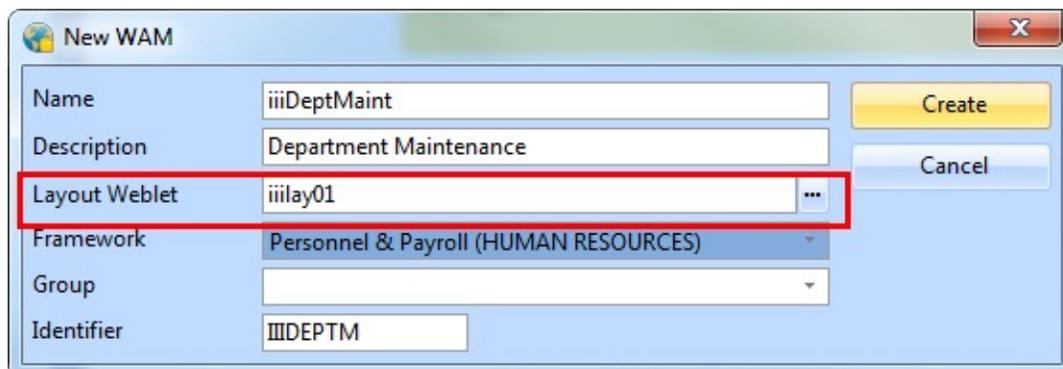
Contrary to what the name suggests, a WAM layout does not have to be made of visual elements - although it usually is.

The non-visual elements of a layout include references to XSL documents for:

- Standard variables
- Standard data types
- Style
- JavaScript
- Default hidden fields

What is a Layout Weblet?

A layout weblet is simply a special kind of weblet which contains the XSL and XHTML which together with appropriate cascading style sheets, defines the content and appearance of a web page. Once you have created and tested your own layout weblet, you can use it as a common layout when creating each WAM which makes up your application.



The screenshot shows a 'New WAM' dialog box with the following fields and values:

| Field | Value |
|---------------|---------------------------------------|
| Name | iiiDeptMaint |
| Description | Department Maintenance |
| Layout Weblet | iiiIay01 |
| Framework | Personnel & Payroll (HUMAN RESOURCES) |
| Group | |
| Identifier | IIIIDPTM |

What do Layouts Determine / Control?

The layout is a key element in the generated webroutine presentation. It ensures that a consistent interface is available across WebRoutines.

When you view your WebRoutine in the LANSA Editor's Outline tab, the layout weblet is generally at the highest level in the outline tree. This indicates that all weblets below the layout in the tree can refer to the documents specified in the layout weblet.

Some of the things controlled by layouts include:

- The appearance of any menus
- Available menu options
- The appearance of a message box
- The Cascading Style Sheet to be applied

- Access to common JavaScript functions
- Definition of any global hidden fields
- Standard variable definitions which may be referenced in other weblets
- Whether a visual layout should be applied.

Of course if you define your own layout, you can decide what common elements need to be included in the interface.

How is a WAM Layout assigned to a WAM?

A WAM-specific layout weblet is automatically generated for a WAM the first time it is built or compiled unless one already exists.

By default, when XSL is generated, the processor checks if a WAM-specific layout weblet already exists for the WAM. If a WAM layout does not exist, a new WAM layout weblet is generated and stored in the repository where the name is composed of the WAM Identifier followed by "_layout". After it has been generated, your WAM-specific layout weblet is referenced by all the webrouines in the associated WAM. Any changes to the WAM-specific layout weblet will be reflected in all of the WAM's webrouines.

The *Generate XSL* options on a WAM compilation do not regenerate the WAM-specific layout. A WAM-specific layout is generated only once. Any subsequent modifications to the WAM-specific layout, or the assignment of a different layout, must be performed in the LANSAs Editor.

How Do I Create My Own Site Layout?

As you have already seen in exercise WAM025, you can use the *Web Application Layout Manager Wizard* to create a common layout based on one of the three main designs supplied, and choose one of nine themes to control background and foreground colours and fonts used.

To create a layout weblet from scratch, select the Weblet option in the LANSAs Editor's New toolbar button dropdown list. In the dialog select the Layout Weblet option to create a Layout weblet.

Alternatively, you can copy one of the shipped layout weblets as a basis to creating your own.

In this exercise you will begin by creating your "company web page" layout outside of LANSAs, together with a style sheet to control its appearance. You will then build a layout template based on this company layout and test it by building a simple WAM which uses it.

To achieve these objectives, you will complete the following:

- Step 1. Create a Simple Company Test Layout
- Step 2. Create a Layout Template
- Step 3. Refine Layout Weblet Definition
- Step 4. Test the New Layout
- Step 5. Review Structure of Layout XSL and HTML
- Summary

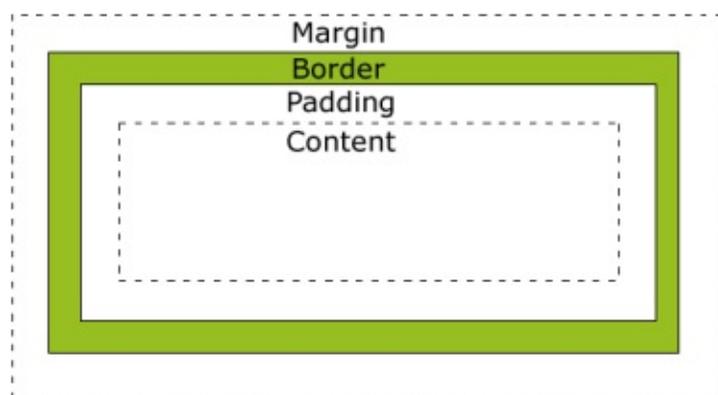
Step 1. Create a Simple Company Test Layout

The layout you will define will be constructed using DIVs and CSS. You could also define a page layout using tables. If you search the Internet on this topic, you will find a number of forums where the pro's and con's are discussed at great length!

The CSS Box Model

It's important to understand this concept. See www.w3schools.com for more information.

The image following illustrates the box model:



The different parts are:

- **Margin** - Clears an area around the border. The margin does not have a background color, it is completely transparent
- **Border** - A border that goes around the padding and content. The border is affected by the background color of the box
- **Padding** - Clears an area around the content. The padding is affected by the background color of the box
- **Content** - The content of the box, where text and images appear

In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

If the following is the CSS applied to an HTML element:

```
width:250px;  
padding:10px;  
border:5px solid gray;  
margin:10px;
```

The total width of the element is 300px, calculated as:

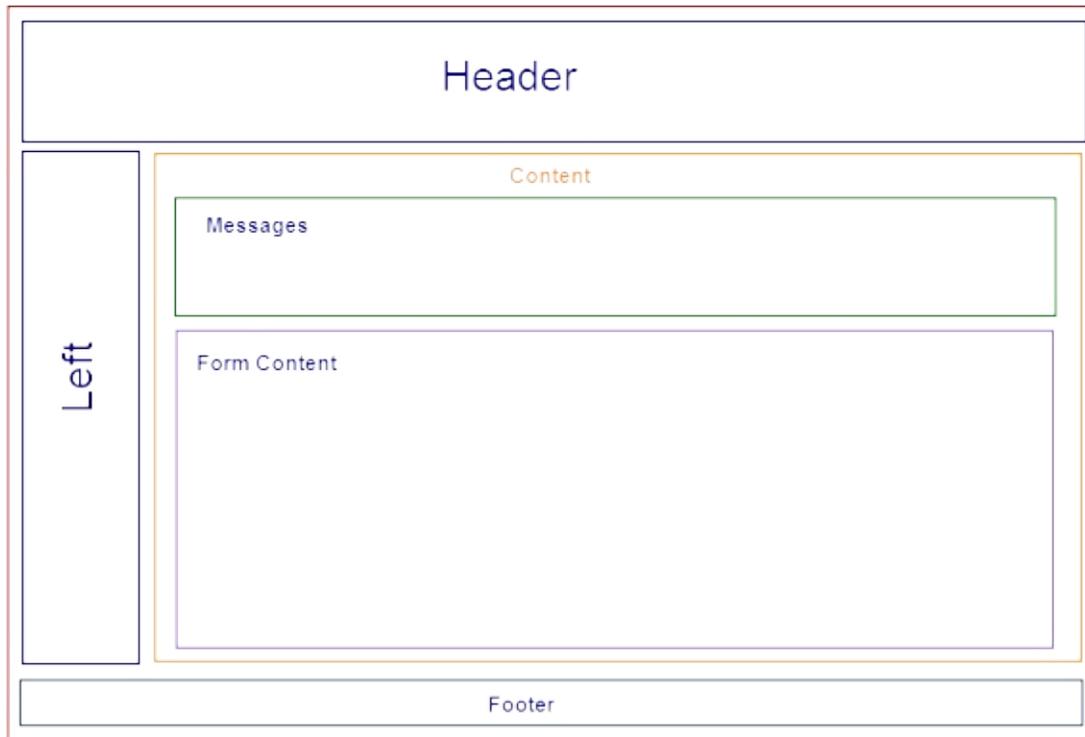
250px (width)
+ 20px (left and right padding)
+ 10px (left and right border)
+ 20px (left and right margin)
= 300px

Acme Company Layout Design

The completed layout will look like the following:



This layout will be constructed as follows:



Each area is defined by a DIV with a unique id. A stylesheet (CSS) will define the position, size and appearance of each DIV. Other content is defined within the appropriate DIV. The content area DIVs are nested so that content, contains messages and form content.

1. Create a new folder in \My Documents called **iii_layout**.
2. Copy the following code into Notepad and save it into \My Documents\iii_layout as a file name **iii_acme_layout.html**.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title> LANSA | Training | cross browser fixed header/footer/left column
layout scrolling middle area | WAMs | </title>
<link REL="stylesheet" TYPE="text/css" href="iii_acme_style.css">
</head>

<body class="acme_layout">
<div id="acme_header">Heading</div>
```

```
<div id="acme_footer">footer</div>
```

```
<div id="acme_sidebar"> <div style="padding-top:400px">Left Panel</div>  
</div>
```

```
<div id="acme_content"> <h2>Content Area</h2>
```

```
<div id="acme_messagesContainer">
```

```
<h2>Messages</h2>
```

```
</div>
```

```
<h2>Form Content</h2>
```

```
</div>
```

```
</body>
```

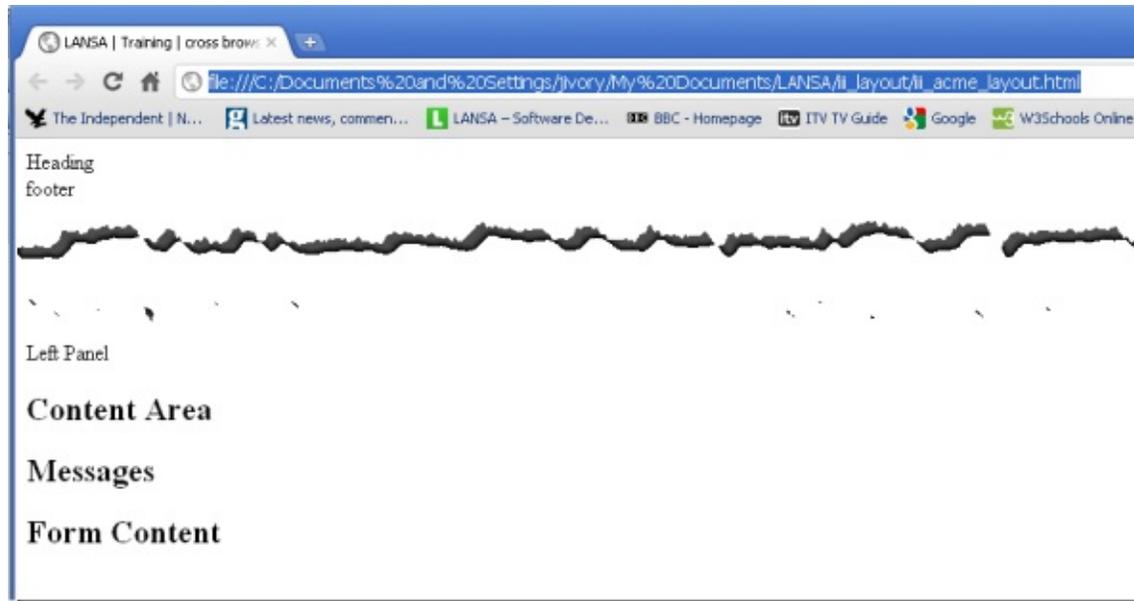
```
</html>
```

3. Change the style sheet named in this layout as **iii_acme_style.css** to use your initials.

Use the *Save as type***All** to save the file with the correct extension.

4. Review the contents of this HTML file.
 - a. The `<!--DOCTYPE` puts IE into standards mode.
 - b. Inside the `<HTML` tag the page content is declared as XHTML and the primary language is English (EN).
 - c. the character set for the document is declared inside the `<meta` tag.
 - d. The `<link` tag declares a style sheet file (CSS) to be referenced by this HTML document. As this does not currently exist, your web page will have no formatting, except browser defaults.
 - e. The web page content is defined as DIVs within the `<body></body>` tags.
 - f. In each DIV, text wrapped by `<h2>` tags identifies each area.
 - g. Each DIV has an id, for example `<div id="acme_content_column"></div>`
 - i. The messages DIV is defined inside the content div.
5. Use Windows Explorer to navigate to `\My Documents\iii_layout` and double click on the file `iii_acme_layout.html` to open it in the default browser.

Your web page will currently look like the following:



Leave your web page open in the browser.

6. Copy the following code into Notepad and use the *Save as type:All* option to save it as **iii_acme_style.css**, to folder `\My Documents\iii_layout`.

```
#acme_content
{
overflow: auto;
position:absolute;
z-index:6;
top:160px;
bottom:50px;
left:200px;
right:0;
border: solid;
border-color: red;
padding: 10px;
}
html {
height:100%;
max-height:100%;
padding:0;
margin:0;
border:0;
font-family:georgia, palatino linotype, times new roman, serif;
overflow: hidden;
```

```
}
body
{
height:100%;
max-height:100%;
overflow:hidden;
padding:0;
margin:0;
border:0;
border: none;
}
#acme_header
{position:absolute;
margin:0;
top:0;
left:0;
display:block;
width:100%;
height:160px;
z-index:5;
overflow:hidden;
font-family:georgia, palatino linotype, times new roman, serif;
border : solid;
border-color: fuchsia;
}
#acme_footer
{
position:absolute;
margin:0;
bottom:0;
left:0;
display:block;
width:100%;
height:50px;
font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
z-index:5;
overflow:hidden;
border: solid;
border-color: aqua;
```

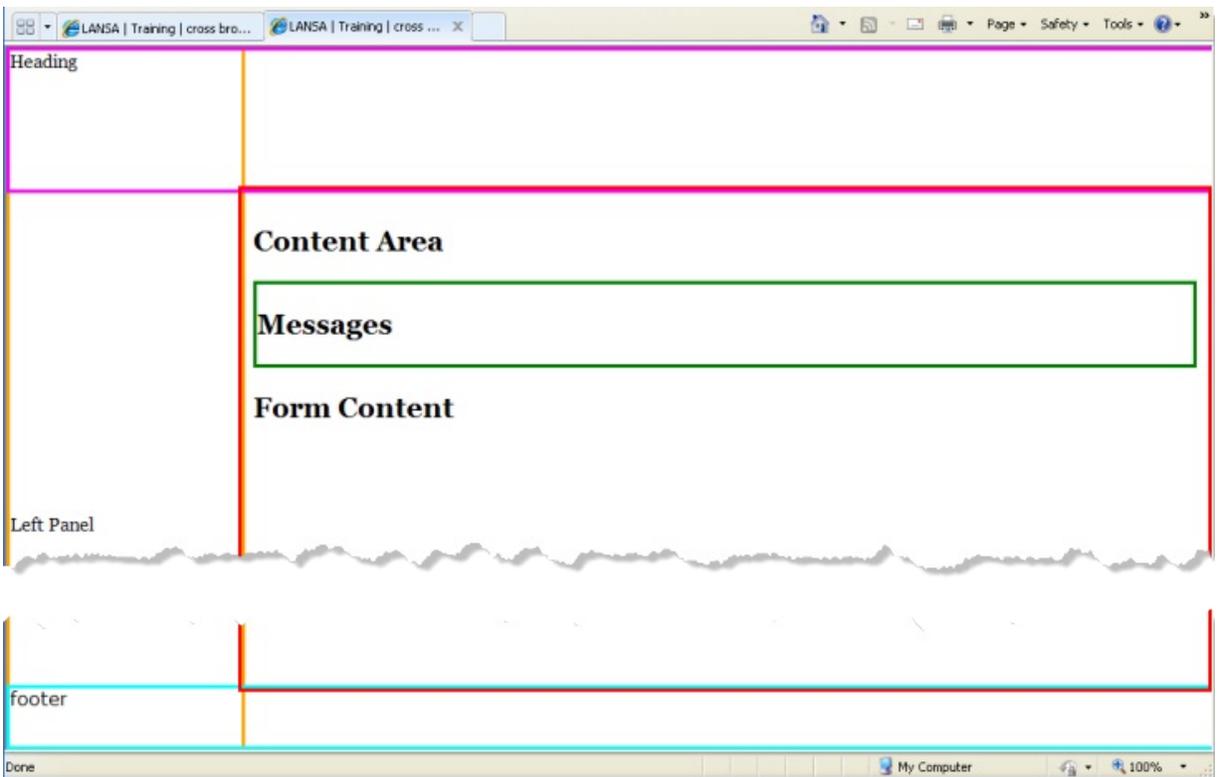
```
}
#acme_sidebar
{
position:absolute;
left:0;
top:0;
bottom: 0;
width:200px;
height : 100%;
z-index:4;
border: solid;
border-color: orange;
}
#acme_content p
{
padding:10px;
}
.bold {font-size:1.2em; font-weight:bold;}
```

```
dd {display:none;}
.p1
{
font-size: .4em;
color: white;
}
#acme_messagesContainer
{
top:0px;
min-height: 50px;
left: 0px;
overflow: auto;
z-index: 6;
border: solid;
border-color: green;
}
```

Notepad is not an ideal editor for CSS files. For your own work, we recommend you use a proper stylesheet editor such as *TopStyle*.

Leave the CSS file open in Notepad.

7. Review the styles sheet, which contains:
 - a. Styles for each id used in your web page, for example `#acme_content{overflow : auto; }`
 - b. The head, foot, sidebar and content DIVs are all position absolutely.
 - c. **acme_head** is positioned at the top of the page (top : 0; left : 0;)
 - d. **acme_content** is positioned below **acme_head** and to the right of **acme_sidebar** (top : 120px; left 200px;)
 - e. The acme_head, acme_foot, acme_sidebar, acme_content and acme_messagesContainer DIVs each have a solid coloured border (just for the moment, to make their position visible).
 - f. acme_content has a setting of overflow : auto. This will provide a scrollbar if the content exceeds the space available.
8. Refresh your web page which should still be open in the browser. It should look like the following:



9. Make the following changes to your Stylesheet file (CSS)
 - a. Remove the border and border-color styles from acme_head, acme_foot, acme_sidebar, acme_content and acme_messagesContainer.

b. Make the following additions to the styles shown:

```
#acme_content
{
Background : #fafad2;
Color : #7b68ee;
}
#acme_header
{
Background : #2ea129;
Color : #fff;
}
#acme_footer
{
Background : #2ea129;
Color : #fff;
}
#acme_sidebar
{
Background : #a19c29;
Color : #fff;
}
#acme_messagesContainer
{
Background : #ffffbb;
Color : black;
}
```

c. Save your style sheet.

10. Refresh your web page in the browser. It should now look like the following:



11. Go to the following web site: <http://generator.lorem-ipsuam.info/>
 - a. Using Notepad to edit your web page HTML document, copy and paste text from the lorem-ipsuam web site into the **messages** DIV and the **content** DIV below the messages DIV. Paste enough text to overflow the content area in the browser.
 - b. Remove the Content Area text (and H2 tags) from the top of the content DIV.
 - c. Refresh your web page in the browser. It should now look like the following:



Note that the content area has been given a vertical scroll bar.

Step 2. Create a Layout Template

In this step you will create a new layout weblet. You will see that the "create new layout weblet" function provides a default layout. You will remove much of this content and replace it with code based on the test layout which you created in Step 1.

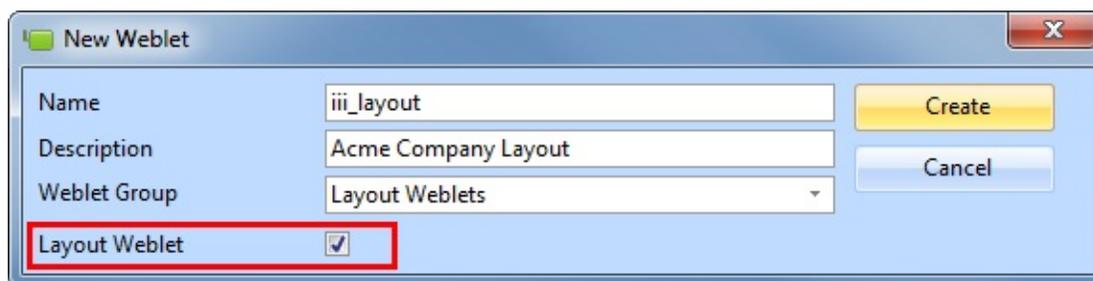
1. Use the *New / Weblet* dialog to create a layout weblet.

Name: **iii_layout**

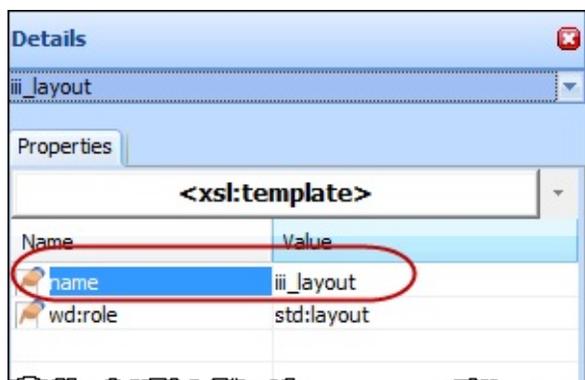
Description: **Acme Company Layout**

Weblet Group: **Layout Weblets**

Select the *Layout Weblet* checkbox.



2. The new layout weblet will open in the editor. In the *Design* view, click above the top of the layout and press F7. Select the *Details* tab to display the layout weblet properties. Change the name to **iii_layout**.



Save the change.

3. Select the *XSL* tab and change the `<wd:template name` to **iii_layout**.

```
<wd:template name="iii_layout">
  <wd:description icon="icons/std_layout.ico">
    <wd:name lang="ENG">New Layout</wd:name>
  </wd:description>
</wd:template>
<!-- Change your layout weblet name -->

<xsl:template name="iii_layout" wd:role="std:layout">
  <xsl:param name="window_title"
    select="$lweb_context/lxml:webapplication-title"
```

4. Change the <wd:description. . . <wd:name to **Acme Layout**.

```
<wd:template name="iii_layout">
  <wd:description icon="icons/std_layout.ico">
    <wd:name lang="ENG">Acme Layout</wd:name>
  </wd:description>
</wd:template>
```

5. As you continue to edit the XSL, regularly save your changes.

6. As you edit the XSL, the syntax will be checked, and any errors highlighted, for example:

```
<wd:definition>
  <wd:default-theme>
    <wd:style name="XWT01J" />
    <wd:style name="XWT01L" />
  </wd:default-theme>
  <wd:group name="Layout Weblets" />
</wd:definition>
<wd:template name="iii_layout">
  <wd:description icon="icons/std_layout.ico">
    <wd:name lang="ENG">iii Acme Layout</wd:name>
  </wd:description>
</wd:template>
<!-- Change your layout weblet name -->
```

Missing <

7. Delete the inline style declaration <style></style>. Delete the style tags and everything between them.

This block of code begins:

```
<style type="text/css">
#lpage_container {
```

8. Towards the end of the XSL there are 7 <xsl:templates. . .> blocks of code, which need to be deleted. They begin:

```
<xsl:template match="/lxml:data">
<xsl:template match="/lxml:data" mode="content.sidebar1">
<xsl:template match="/lxml:data" mode="content.sidebar2">
```

```
<xsl:template match="/lxml:data" mode="content.header">
<xsl:template match="/lxml:data" mode="content.navigation">
<xsl:template match="/lxml:data" mode="content.footer">
<xsl:template match="/lxml:data" mode="content.hidden">
```

Delete from the lines listed, up to their end tag `</xsl:template>` and everything inside the templates.

Note: Make sure you do not delete the `</xsl:transform>` at the end of the document.

9. Save your changes.

10. Find the code shown and mark it for retention by adding the comment lines shown around it.

New code is highlighted in red, italic.

Hint: Search (Ctrl+F) for the first line shown using `lpage_content_column`. The other lines shown follow immediately after it.

```
<!-- KEEP -->
<div id="lpage_content_column">
<div id="lpage_content" class="lpage_content_area">
<xsl:if test="$show_title">
<h2 class="title">
<xsl:value-of select="$title_text" />
</h2>
</xsl:if>
<div wd:content="content">
<xsl:apply-templates select="*" />
</div>
</div>
</div>
<!-- END KEEP -->
```

11. Above the code shown in 10. locate the following line and delete it, and everything up to your first `<!--keep -->` comment line.

```
<div id="lpage_header" wd:content="content.header">
```

Hint: Move the cursor to the top of the XSL document and search (Ctrl+F) for `lpage_header`.

12. The last step will have highlighted as an error, the `</div>` tag below your `<!--end keep-->` comment line. Delete this `</div>` and everything before:

```
</xsl:element>
```

Note: Do not delete the line `</xsl:element>`.

13. Save your changes.
14. With your test web page layout open in Notepad (created in Step 1), copy all the code inside the `<body></body>` tags (not including the body tags) and paste in into your new WAM layout. Paste it inside the `<body></body>` tags, **below** the `</xsl:if>`. Your code should look like the following.

The inserted code is shown in red.

```
</xsl:if>
```

```
<div id="acme_header">Heading</div>
<div id="acme_footer">footer</div>
<div id="acme_sidebar">
<div style="padding-top:400px">Left Panel</div>
</div>
<div id="acme_content">
<div id="acme_messagesContainer">
<h2>Messages</h2>
</div>
<h2>Form Content</h2>
</div>
```

```
<!-- keep -->
```

15. Change the *class* of the `<body>` tag to **acme_layout**. This line should now look like the following:

```
<body class="acme_layout">
```

16. Move the code (cut and paste) shown in red below, from the block of xsl which you commented to keep, into the position shown, immediately before the `<h2>Form Content</h2>` line.

The moved code is shown in red.

```
div id="acme_header">Heading</div>
<div id="acme_footer">footer</div>
<div id="acme_sidebar">
<div style="padding-top:400px">Left Panel</div>
</div>
<div id="acme_content">
<div id="acme_messagesContainer">
<h2>Messages</h2>
</div>
```

```
<xsl:if test="$show_title">
<h2 class="title">
<xsl:value-of select="$title_text" />
</h2>
</xsl:if>
```

```
<h2>Form Content</h2>
</div>
```

17. Move the code shown in red below from the xsl you commented to keep, into the position shown, after the `<h2>Form Content</h2>` line. The moved code is shown in red.

```
<div id="acme_header">Heading</div>
<div id="acme_footer">footer</div>
<div id="acme_sidebar">
<div style="padding-top:400px">Left Panel</div>
</div>
<div id="acme_content">
<div id="acme_messagesContainer">
<h2>Messages</h2>
</div>
<xsl:if test="$show_title">
<h2 class="title">
<xsl:value-of select="$title_text" />
</h2>
</xsl:if>
<h2>Form Content</h2>
```

```
<xsl:apply-templates select="*" />
```

</div>

18. Delete the remaining saved xsl. i.e everything which now remains within the comments <!--keep --> and <!--end keep _-->.
19. Save your changes. At this point you have your basic layout defined. No stylesheet is associated with it except for the styles defined by the standard_style.xsl weblet.

Step 3. Refine Layout Weblet Definition

In this step you will define a style *External Resource* for your new style sheet, and add the external resource to your layout.

1. You created a style sheet (iii_acme_style.css) for the new layout in *Step 1*.
 - a. Copy your iii_acme_style.css from \My Documents\iii_layout to your Visual LANSA web server \images\style folder. The path name should be similar to:

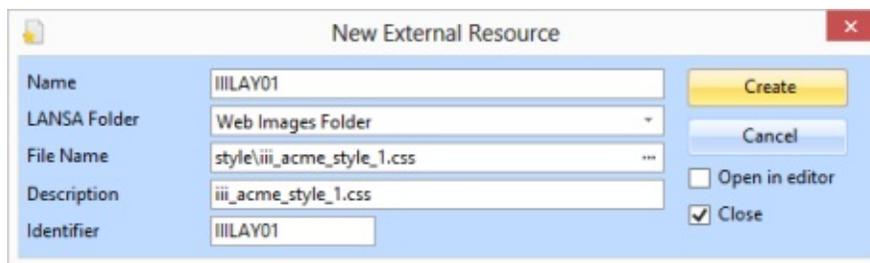
C:\Program Files (x86)\LANSA\WebServer\Images\style

On the IBM i, copy to the web server /images/style folder for the LANSA installation which you are using. The path name will be similar to:

LANSA_<PGMLIB>/webserver/images/style

Where <PGMLIB> is your LANSA program library.

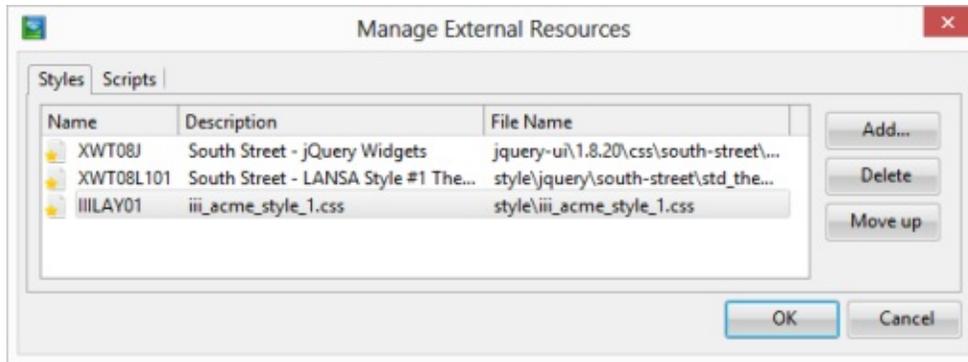
- b. Create an *External Resource* for this style sheet using the *New / External Resource / External Resource* option.



Enter the *External Resource Name*: **IIILAY01** and use the *Ellipsis* button for the *File Name* to select your style sheet file. The *LANSA Folder* and *Description* will be automatically filled.

- c. Click the *Create* button to save the new External Resource definition. You do not need to open it in the editor.
2. With your layout weblet open in the *Design view*, add the External Resource using the *Web / Manage External Resource* menu option.

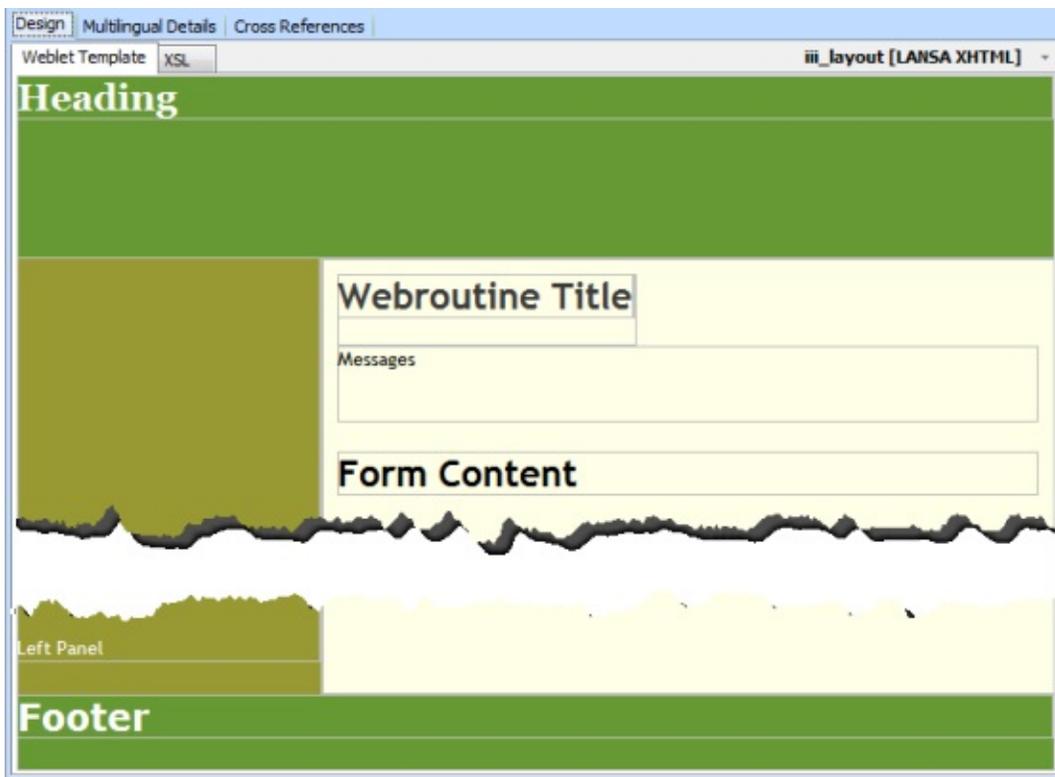
At the same time also add External Resources for styles XWT08J and XWT08L101. These will add style sheets to set fonts, colors and background colors for your page contents to blend with the green header and footer being used by the layout.



3. Delete the default theme external resources which were part of the default layout.

```
<wd:style name="XWT01J" />
<wd:style name="XWT01L" />
```

4. Save your changes. Your layout should now look like the following:



5. Remove the following text:

- a. Delete this line, which was defined with a <div> which includes an inline style, in order to position the text where it would be shown and not hidden:

```
<div style="padding-top:400px">Left Panel</div>
```

b. Delete the following text, including any header tags associated with it (e.g. `<h2></h2>`)

Messages

Form Content

6. In the `acme_header` DIV replace the "Heading" text with the following. This may be copied from *WAM Tutorials* in the *Web Application Modules* guide.

```
<span style="padding-left: 10px">  
<h1>Acme IT Services</h1>  
</span>  
<br />  
<span style="padding-left: 10px">  
<h2>Software for Business</h2>  
</span>
```

7. In the `acme_footer` DIV replace the "footer" text with:

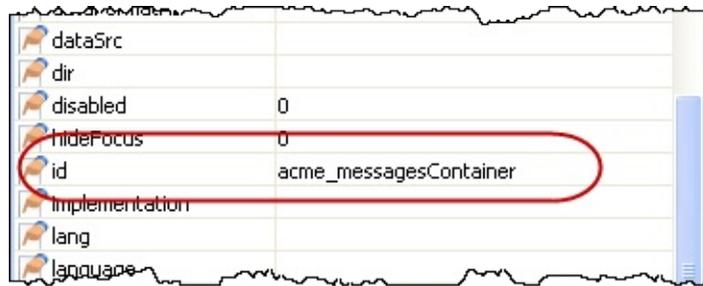
```
<span style="padding-left: 10px; padding-top:5px">© Acme Ltd 2011</span>
```

These changes use the `` tag to position the text using inline styles. A better solution would be to give these span tags an `id` and control the appearance via the external stylesheet.

8. Save your changes and review your layout in the *Design View*.
9. The DIV `acme_messagesContainer` is shown in the *Design* view as an outline box, above the *webroutine* title text. Select this area:



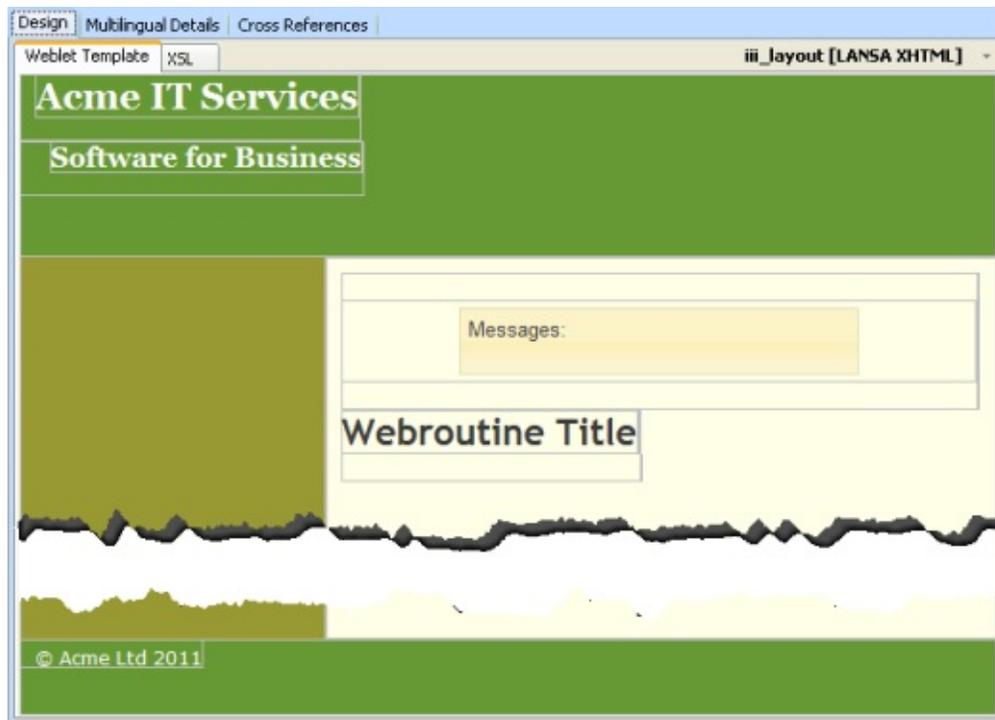
You can confirm you have the correct element selected by examining the *Details* tab, and checking the *id*.



From the *Standard Weblets* group, drop a messages weblet into the acme_messagesContainer DIV.

10. Save your changes

Your design should look like the following:



11. On the *Favorites/Weblet Templates* tab, select jQuery UI group from the top combo box.

The acme_header DIV is the top green box containing the text "Acme IT Services"

- a. Select the acme_header DIV
- b. Drop a *jQuery UI Menubar* into the **acme_header** DIV.
- c. Select the menu bar, and use the right mouse on one of the "handles" to *Insert HTML / Div*. This will insert a DIV around the menu bar, which can

then be given an *id*. An addition to the style sheet for this id, will then position the menu bar.

- d. With the new DIV selected, select the *Details* tab and change the *id* to **lpage_navBar**.
- e. Add the following style to your style sheet *iii_acme_style.css* and save the style file. This will override the LANSA styles for the **idlpage_navBar**.

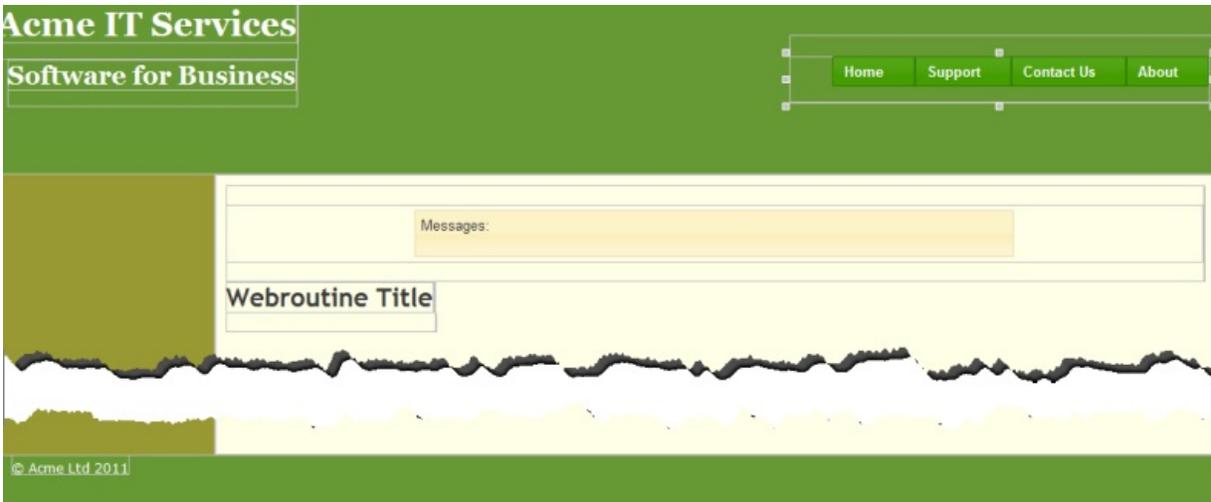
```
#lpage_navBar
{
min-width: 396px;
min-height: 44px;
position: absolute;
top: 10px;
left: 550px;
}
```

- d. Save the changes to your layout *iii_layout*.
12. With the menubar selected, set up the *menuitems* property on the *Details* sheet, using the *Ellipsis* button button to open the *Design of...* dialog. Define a top level menu only as:

Menu Item Action URL

Home	http://www.lansa.com
Support	http://www.lansa.com/support/index.htm
Contact Us	http://www.lansa.com/about/contactus.htm
About	http://www.lansa.com/about/index.htm

13. Save your changes. Your design should look like the following:



14. Your layout `xsl:template` currently contains some parameters which will not be used. Delete the following code:

```
<xsl:param name="width_type" select=""fluid""  
wd:type="std:layout_width_type" />  
<xsl:param name="width" select=""1000px"" wd:type="std:css_length_unit" />  
<xsl:param name="sidebar1_width" select=""20%"" wd:tip_id=""  
wd:type="std:css_length_unit" />  
<xsl:param name="content_width" select=""50%"" wd:tip_id=""  
wd:type="std:css_length_unit" />  
<xsl:param name="sidebar2_width" select=""30%"" wd:tip_id=""  
wd:type="std:css_length_unit" />
```

Step 4. Test the New Layout

1. Create a new WAM:

Name: **iiiTestLayout**

Description: **Test layout iii_layout**

Layout weblet: **iii_layout**

2. Add the following RDMLX code after the Begin_com and compile the WAM.

```
Define Field(#empnow) Reffld(#empno)
Def_List Name(#empskills) Fields(#SKILCODE #GRADE #COMMENT #D/
Web_Map For(*both) Fields((#stdrentry *hidden))
Webroutine Name(begin) Desc('Employee Details and Skills')
Web_Map For(*both) Fields((#EMPNO *out) #SURNAME #GIVENAME #A
Case (#stdrentry)
When (= U)
#empno := #empnow
Update Fields(*all) In_File(pslmst) With_Key(#empno)
Otherwise
Select Fields(*all) From_File(pslmst)
Leave
Endselect
#empnow := #empno
Clr_List Named(#empskills)
Select Fields(#empskills) From_File(pslskl) With_Key(#empno)
Add_Entry To_List(#empskills)
Endselect
Endcase
Endroutine
```

3. Open the **begin** WebRoutine in the *Design* view.
 - a. Add a column to the employee fields table.
 - b. Drop a push button into the top of the new column
 - c. Remove the place holder characters.
 - d. Select the push button and set up its properties as:

Property	Value
----------	-------

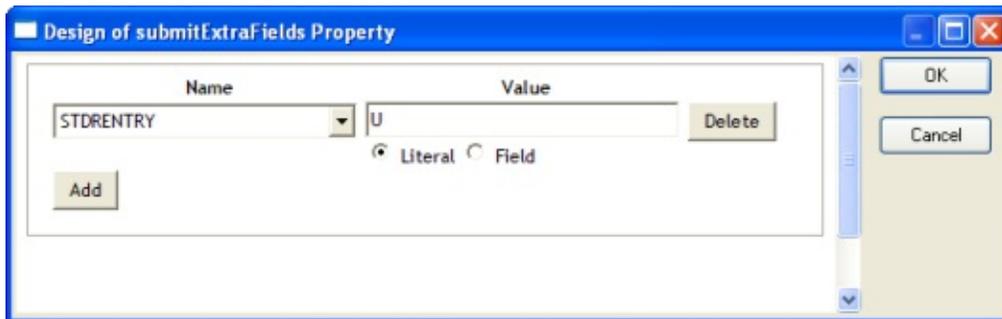
Caption **Save**

On_click_wname **Begin**

submitExtraFields **Field Name: STDREENTRY**

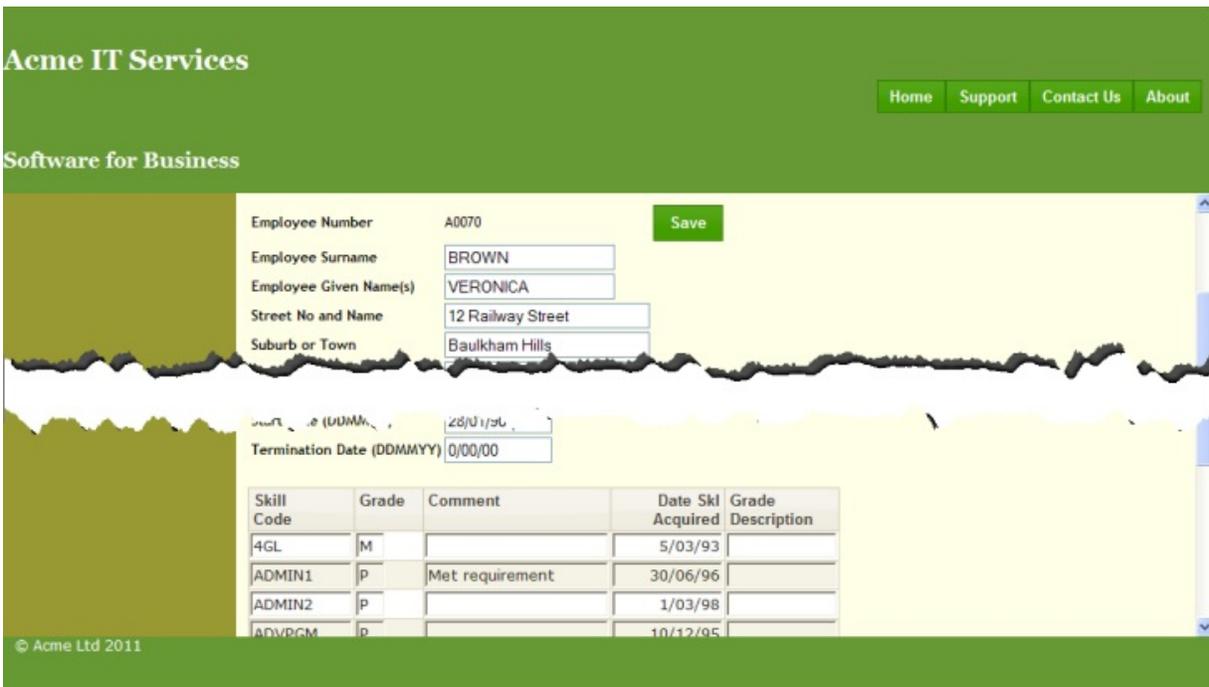
Literal Value: U

Complete the submitExtraFields property using the Ellipsis button:



e. Save your changes.

f. Execute the WAM in the browser. Your web page should look like the following:



4. Clear the surname and given name fields and click the Save button.

Validation errors should be displayed in the messages weblet at the top of the content area, as shown:



The screenshot shows a web form titled "Employee Details and Skills" with a yellow background. At the top, a "Messages:" box contains two error messages: "A surname is required" and "Given name(s) must be specified". Below the messages, the form fields are: "Employee Number" with the value "A0070", "Employee Surname" with an empty text box, and "Employee Given Name(s)" with an empty text box. A green "Save" button is located to the right of the "Employee Number" field. The form is framed by a decorative, torn-edge border.

5. Try out the menu bar. Notice that the menu items appearance changes for hover and selection. This is controlled by the theme style sheet defined by the external resource XWT08L101.

Step 5. Review Structure of Layout XSL and HTML

This step analyses the structure of the layout XSL, to explain what the main parts of the program are doing.

1. WAMs use the XSL Transformation language (XSLT) to process the XML output by a WebRoutine to produce a web page – an XHTML document. You can learn more about XSL at the www.w3schools.com web site.

The first few lines of code, define the XSL namespace standards used in the document.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (c) 2003, 2011 LANSA          -->
<!-- XHTML WAM Layout           -->
<!-- $Revision:: 12              $ -->
<xsl:transform version="1.0" exclude-result-prefixes="lxml wd tsm1"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:lxml="http://www.lansa.com/2002/XML/Runtime-Data"
  xmlns:wd="http://www.lansa.com/2002/XSL/Weblet-Design"
  xmlns:tsml="http://www.lansa.com/2002/XML/Generation-
Metadata"
  xmlns="http://www.w3.org/1999/xhtml">
```

XSL code such as `<xsl:if . . .>` has a namespace of **xsl** and is a standard defined by the World Wide Web consortium.

XSL code such as `<wd:external-resources. . .>` has a namespace of **wd** and is LANSA defined standard.

2. The `xsl:import` statements, import other weblets into this layout:

```
<!-- Standard imports          -->
  <!-- Import the weblet XSL files used by your layout here  -->
  <!-- e.g.                    -->
  <xsl:import href="std_variables.xsl" />
  <xsl:import href="std_types.xsl" />
  <xsl:import href="std_hidden.xsl" />
  <xsl:import href="std_style_v2.xsl" />
  <xsl:import href="std_script.xsl" />
  <xsl:import href="std_messages.xsl" />
  <xsl:import href="std_menubar.xsl" />
  <xsl:output method="xml" omit-xml-declaration="yes" encoding="UTF-8"
```

```
indent="no" doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd" />
```

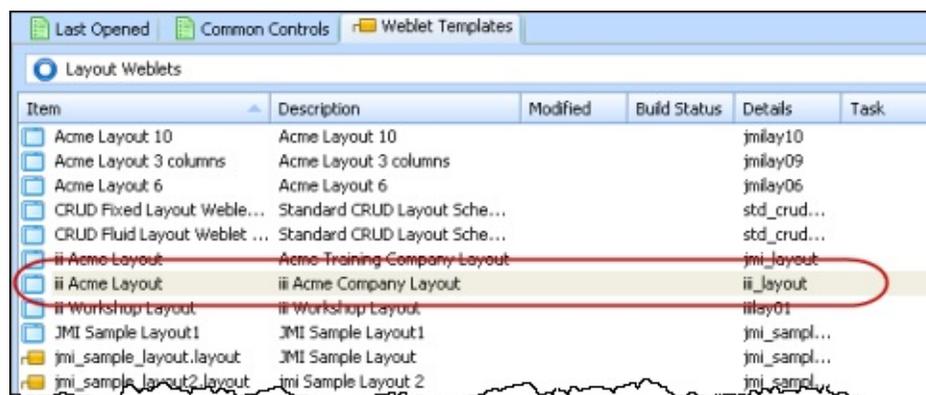
When the messages and menu bar weblets were added to this layout in the graphical editor, import statements for `std_messages.xsl` and `std_menubar.xls` were added.

3. The XSL statements which begin **wd:** are LANSAs weblet design statements which are used within LANSAs graphical web page editor (actually an XSL editor). **wd:** is a LANSAs namespace.

```
<wd:external-resources>
  <wd:style name="XWT08J" />
  <wd:style name="XWT08L101" />
  <wd:style name="IIILAY01" />
</wd:external-resources>
<wd:definition>
  <wd:default-theme />
  <wd:group name="Layout Weblets" />
</wd:definition>
<wd:template name="iii_layout">
  <wd:description icon="icons/std_layout.ico">
    <wd:name lang="ENG">iii Acme Layout</wd:name>
  </wd:description>
</wd:template>
```

The external resources will define links to external style sheets to be used by web pages based on this layout.

The above tags also provide a name, description, group name and icon to be used for this weblet in the Repository.

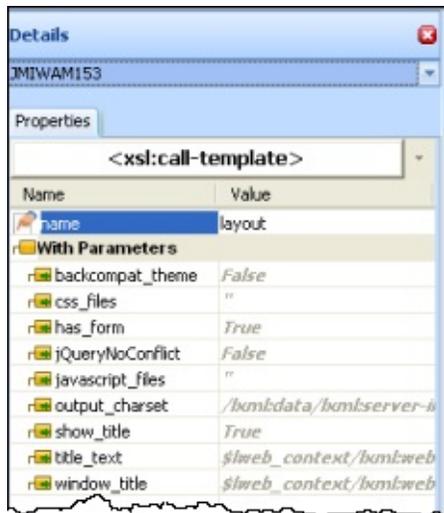


Item	Description	Modified	Build Status	Details	Task
Acme Layout 10	Acme Layout 10			jmlay10	
Acme Layout 3 columns	Acme Layout 3 columns			jmlay09	
Acme Layout 6	Acme Layout 6			jmlay06	
CRUD Fixed Layout Weblet...	Standard CRUD Layout Sche...			std_crud...	
CRUD Fluid Layout Weblet ...	Standard CRUD Layout Sche...			std_crud...	
ii Acme Layout	Acme Training Company Layout			jmi_layout	
iii Acme Layout	iii Acme Company Layout			ii_layout	
ii Workshop Layout	ii Workshop Layout			iiay01	
JMI Sample Layout1	JMI Sample Layout1			jmi_sampl...	
jmi_sample_layout.layout	JMI Sample Layout			jmi_sampl...	
jmi_sample_layout2.layout	jmi Sample Layout 2			jmi_sampl...	

4. xsl:templates are like subroutines. They can be called, passing parameters into the template.

```
<xsl:template name="iii_layout" wd:role="std:layout">
  <xsl:param name="window_title"
    select="$lweb_context/lxml:webapplication-title"
    wd:type="std:mtxt_variable" wd:tip_id="" />
  <xsl:param name="has_form" select="true()" wd:type="std:boolean"
    wd:tip_id="" />
  <xsl:param name="show_title" select="true()" wd:type="std:boolean"
    wd:tip_id="" />
  <xsl:param name="title_text" select="$lweb_context/lxml:webroutine-
title"
    wd:type="std:mtxt_variable" wd:tip_id="" />
  <xsl:param name="javascript_files" select=""" wd:tip_id="" />
  <xsl:param name="jQueryNoConflict" select="false()" wd:type="std:boole
    wd:tip_id="" />
  <xsl:param name="css_files" select=""" wd:tip_id="" />
  <xsl:param name="output_charset"
    select="/lxml:data/lxml:server-instructions/lxml:client-charset"
    wd:tip_id="" />
  <xsl:param name="backcompat_theme" select="false()" wd:type="std:boo
    wd:tip_id="" />
```

When you compile a WAM using this Layout Weblet, the generated WAM layout uses the layout weblet specified. When you then design a WebRoutines's web page, these parameters will be shown on the *Details* tab and could be set at design time and or at runtime.



5. The XHTML definition begins here. Note that an **xsl:if** is testing the value of an environment language variable (\$lweb_ISO_language). Further down another **xsl:if** refers to **\$window_title** which is the value of a boolean input parameter.

```

<html>
  <xsl:if test="$lweb_ISO_language != "">
    <xsl:attribute name="xml:lang">
      <xsl:value-of select="$lweb_ISO_language" />
    </xsl:attribute>
    <xsl:attribute name="lang">
      <xsl:value-of select="$lweb_ISO_language" />
    </xsl:attribute>
  </xsl:if>
  <head>
    <title>
      <xsl:value-of select="$window_title" />
    </title>
    <xsl:choose>
      <xsl:when test="$backcompat_theme">
        <xsl:call-template name="style">
          <xsl:with-param name="theme_css_filename" select="" />
          <xsl:with-param name="css_files" select="$css_files" />
        </xsl:call-template>
      </xsl:when>
      <xsl:otherwise>
        <xsl:call-template name="style">

```

```

        <xsl:with-param name="theme_css_filename" select=""none"" />
        <xsl:with-param name="css_files" select="$css_files" />
    </xsl:call-template>
</xsl:otherwise>
</xsl:choose>
<xsl:call-template name="script">
    <xsl:with-param name="javascript_files"
        select="$javascript_files" />
    <xsl:with-param name="trap_script_errors" select="false()" />
    <xsl:with-param name="jQueryNoConflict"
        select="$jQueryNoConflict" />
</xsl:call-template>
</head>

```

The above code defines the contents of the head area of the web page. The XSL logic is generating XHTML based on the parameters provided at runtime.

6. Your page content is output within the HTML <body</body> tags.

```

<body class="acme_layout">
    <xsl:variable name="containerName">
        <xsl:choose>
            <xsl:when test="$has_form">
                <xsl:text>form</xsl:text>
            </xsl:when>
            <xsl:otherwise>
                <xsl:text>div</xsl:text>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:variable>
    <xsl:element name="{ $containerName }">
        <xsl:attribute name="id">lpage_container</xsl:attribute>
        <xsl:if test="$has_form">
            <xsl:attribute name="onsubmit">return _HandleDefaultSubmit(this
</xsl:attribute>
            <xsl:attribute name="method">post</xsl:attribute>
            <xsl:attribute name="name">LANSA</xsl:attribute>
            <xsl:attribute name="action">
                <xsl:value-of select="$lweb_context/lxml:action-request" />?
            </xsl:attribute>

```

```
<xsl:call-template name="hidden_fields" />
</xsl:if>
```

The `xsl:choose` logic (like an RDML CASE loop) outputs a `<form>` tag or a `<div>` depending on the variable `$has_form`. **Has_form** is a property of the layout, and is shown on the *Properties* sheet on the *Design* tab.

7. This section is largely the code which you provided and represents the design of your web page, when combined with the necessary stylesheet.

```
<div id="acme_header">
  <span style="margin-left:10px; padding-top:0px">
    <h1>Acme IT Services</h1>
  </span>
  <br />
  <span style="margin-left: 20px">
    <h2>Software for Business</h2>
  </span>
  <div id="lpage_navBar">
    <xsl:call-template name="std_menubar">
      <xsl:with-param name="menu_items"
        select="document('')/*/*/lxml:data/lxml:menu[@id='E4
      <xsl:with-param name="name"
        select="concat('o', position(), '_LANSA_17914')"/>
      <!-- <xsl:with-param name="listname" select="?" /> -->
      <!-- <xsl:with-param name="orientation" select="?" /> -->
      <!-- <xsl:with-param name="show_arrows" select="?" /> -->
      <!-- <xsl:with-
param name="submit_selected_to" select="?" /> -->
    </xsl:call-template>
  </div>
</div>
<div id="acme_footer">
  <span style="margin-left: 10px; padding-
top:5px">© Acme Ltd 2011</span>
</div>
<div id="acme_sidebar"></div>
<div id="acme_content">
  <div id="acme_messagesContainer">
    <xsl:call-template name="messages">
```

```
        <!-- <xsl:with-  
param name="target_window_name" select="?" /> -->  
        </xsl:call-template>  
    </div>  
    <xsl:if test="$show_title">  
        <h2 class="title">  
            <xsl:value-of select="$title_text" />  
        </h2>  
    </xsl:if>  
    <xsl:apply-templates select="*" />  
    </div>  
</xsl:element>  
</body>
```

This code also includes the menu bar and messages weblets which you added using the graphical editor.

The `<xsl:apply-templates select="*" />` outputs your WebRoutine fields and lists, together with output from any weblets which were added to the page, such as *Push Buttons*.

Summary

Important Observations

- You have created your own company layout which can be used as the Layout Weblet when creating your application WAMs
- Any changes you make to this common layout, will affect all WAM layouts which are based on this common layout.
- The shipped layout designs demonstrate how the same design can be significantly altered simply by changing the style sheets associated with the design. Consider this option, before embarking on development of a completely new layout.

Tips & Techniques

- With the design tools now available, such as the *Microsoft IE Developer Toolbar* and a professional style sheet editor you can rapidly understand the detailed design of any web page you may want to adopt.

What You Should Know

- How to create your own layout weblet.
- How to use CSS to control page layout which is based on DIVs.

WAM115 - Check in WAMs to IBM i

Objectives

- To check in and compile some of your exercise WAMs and their layouts to the IBM i
- To run the WAMs on the IBM i

This exercise is optional and will depend on the training facilities being used. Your Visual LANSAs must be installed as a Slave Workstation to the IBM i LANSAs Master.

All the files and most of the fields used in this workshop will already exist on the IBM i, in the partition being used for training.

In order to complete this exercise you will perform the following:

- [Step 1. Check in a WAM and its Layout](#)
- [Step 2. Run a WAM on the IBM i](#)
- [Step 3. Run a WAM in debug on the IBM i](#)
- [Summary](#)

Before You Begin

This exercise assumes that you have completed all preceding exercises.

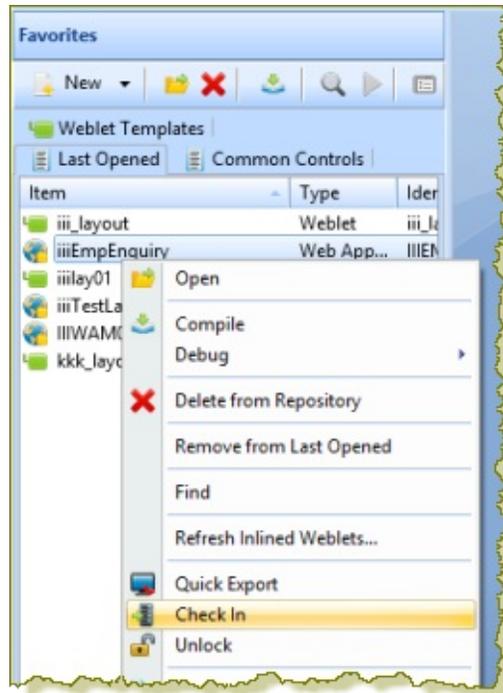
Step 1. Check in a WAM and its Layout

To compile and run your WAM on the IBM i, you will need to ensure that you have also checked in any dependent objects.

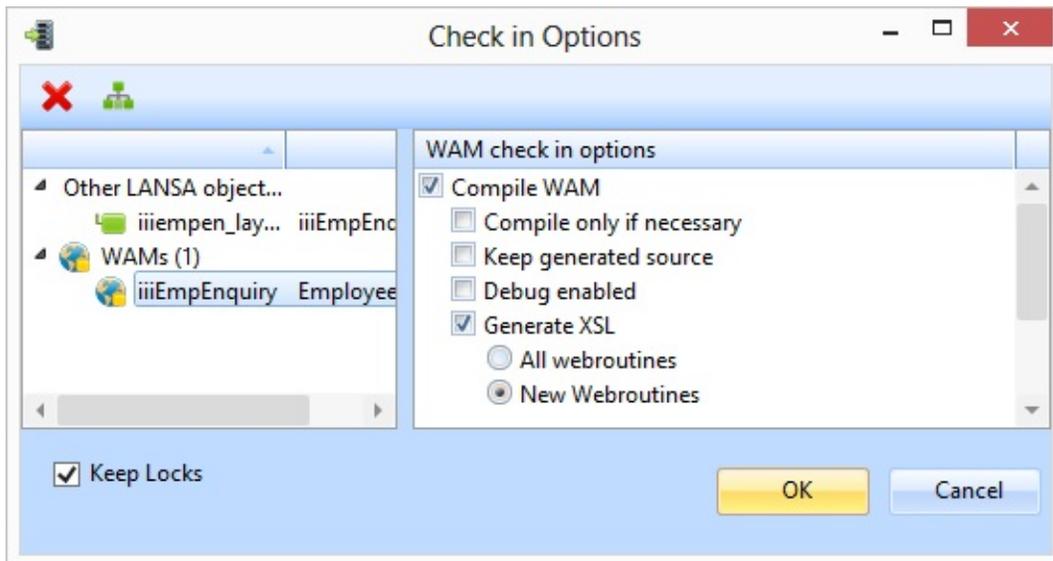
- Checking in a WAM also checks in the web page definition for each WebRoutine. The WAM layout will also be automatically checked in.
- You only need to check in a WAM layout once, or when it has been changed. You will find that LANSa V13 always checks in the WAM layout when you check in the WAM.
- When checking in a WAM, you should use the Cross References dialog to select and check in any dependent objects.
- If your WAM layout is based on a common company layout, you will check this layout in, the first time you check in a WAM which uses it, and whenever this common layout has been changed.
- Other definitions which you may need to consider are any custom weblets or External Resources which your WAM is using.
- You can check in using the *context menu* from the *Repository* tab, or from the *Last Opened* tab.
- With the WAM open in the editor, you can also check in from the *Home ribbon / Remote Systems/ Check in* button:



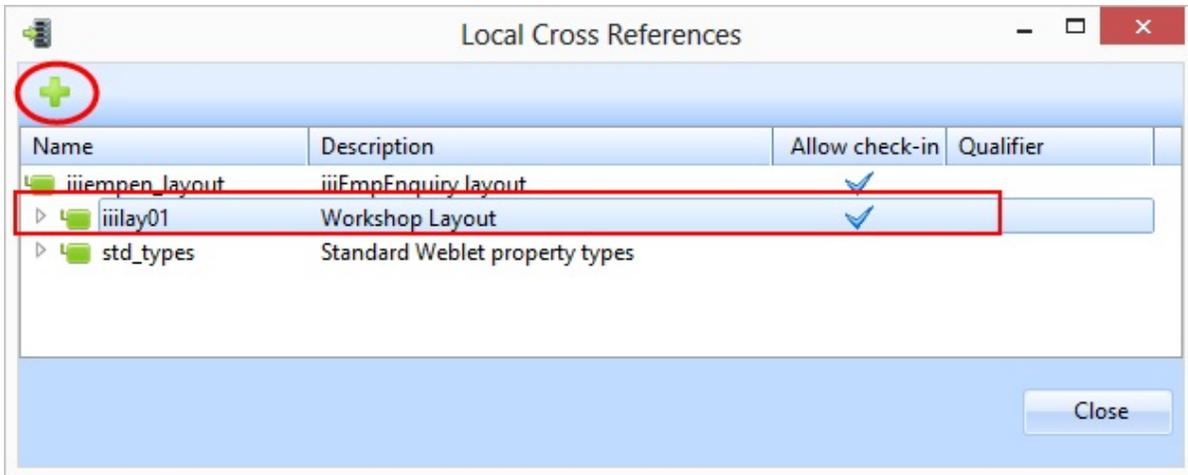
1. On the *Repository* tab find WAM iiiEmpEnquiry under *Web / Web Application Modules*. Use the context menu to *Check In*:



The *Check In Options* dialog opens.

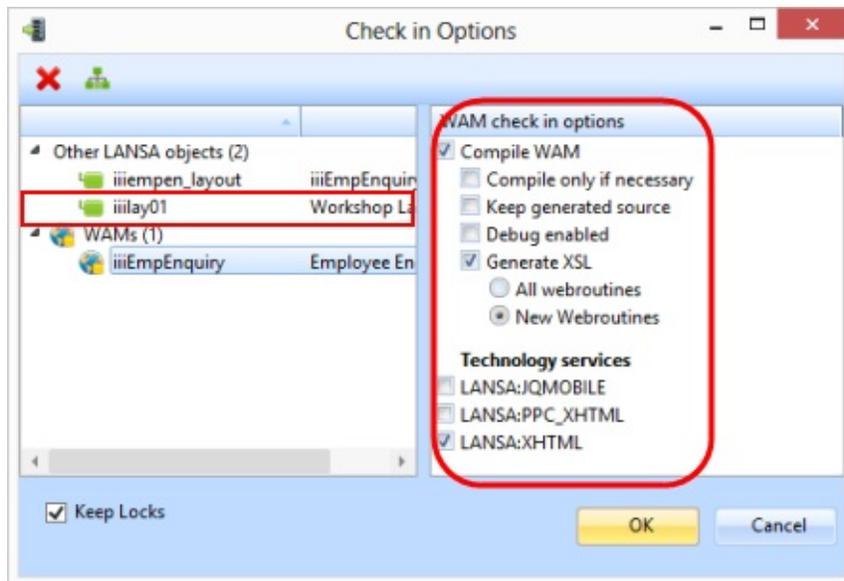


2. Select the WAM layout to enable the  *Cross References* toolbar button. Click the toolbar button to open the *Local Cross References* dialog:



Select the common layout **iiilay01** and click the  *Add for check in* button, to add it to the check in list of objects.

3. Click the *Close* button to close this dialog. Your check in dialog should look like the following:



Note: The common layout iiilay01 has been added to the objects to be check in.

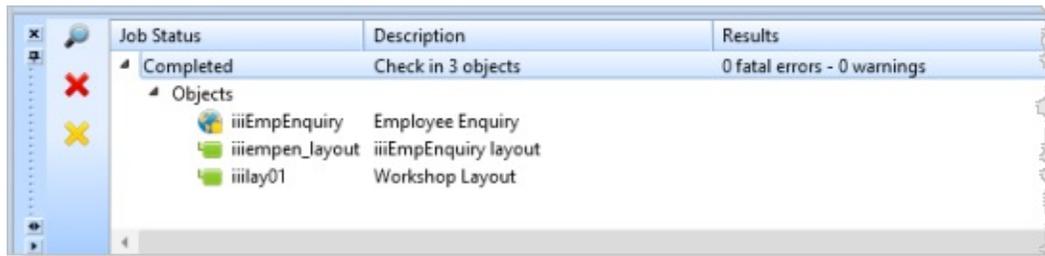
Ensure that the *Check In* options are as shown.

When you check in a WAM for compile the following steps occur:

- A C program for the WAM RDMLX code is generated and checked in.
- A compile job is submitted on the IBM i.
- The XSL Transformations for all WebRoutines are checked in for the

Technology Services which are selected in the check in options.

- The WAM layout is checked in.
 - As noted, you need to also check in any other dependent objects as appropriate.
 - Your new WAM is locked to a Task Id. Usually this will be the Task Id you selected when you logged in the Visual LANSAs. The lock is applied using your PC Name. If you select the *Keep Locks* option shown in the above dialog, Visual LANSAs will keep the WAM locked using Task Id and PC Name and you could continue developing the WAM. See the *Visual LANSAs Administrator's Guide* for further details on this subject.
4. Confirm the check in by clicking the *OK* button. The *Check in* information area will appear at the foot of the Editor window as shown in this example.

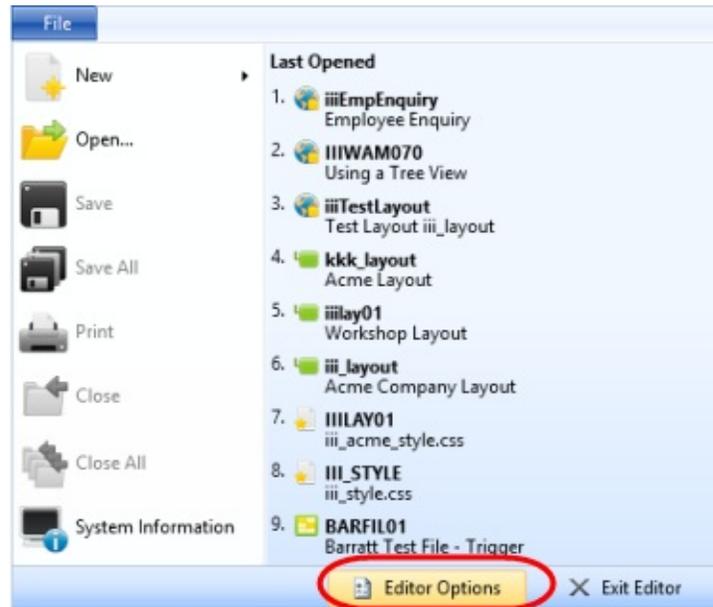


Job Status	Description	Results
Completed	Check in 3 objects	0 fatal errors - 0 warnings
Objects		
iiiEmpEnquiry	Employee Enquiry	
iiiempen_layout	iiiEmpEnquiry layout	
iiilay01	Workshop Layout	

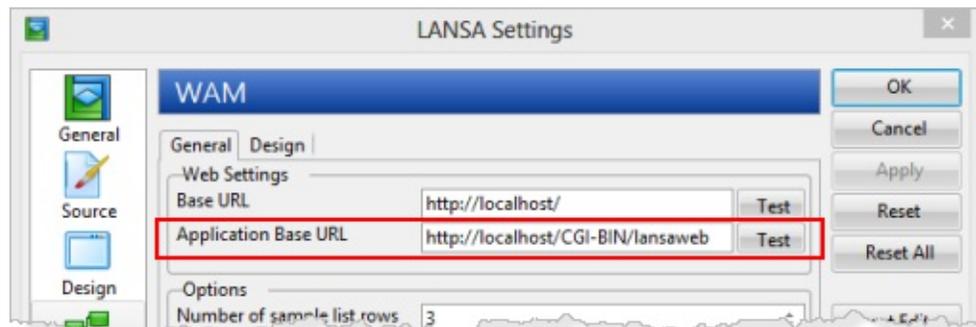
5. Review the check in information to ensure that no errors have occurred.

Step 2. Run a WAM on the IBM i

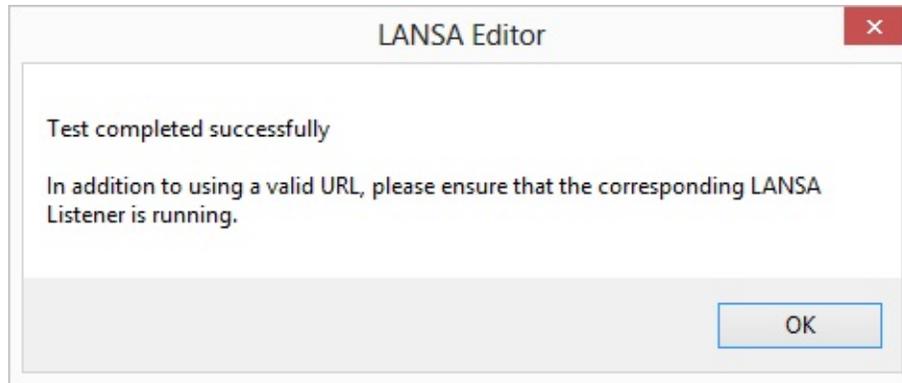
1. Open WAM iiiEmpEnquiry in the editor. Open the **begin** WebRoutine in the *Design* view.
2. Use the *Editor Options* from the *File* menu.



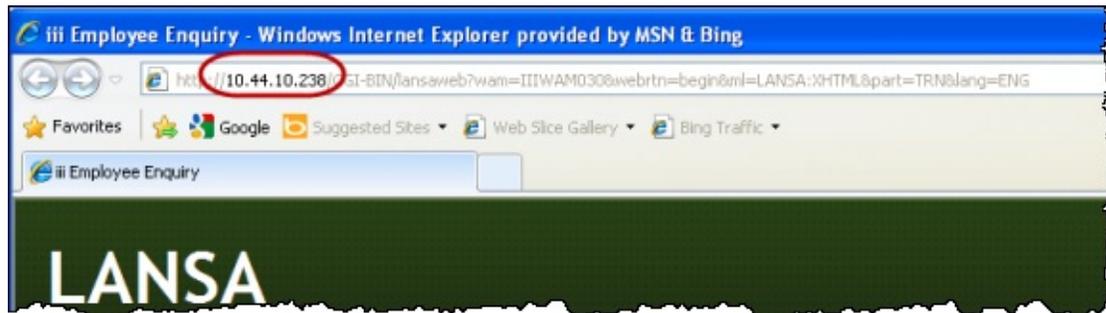
Select the *WAM* icon:



3. Change the *Application Base URL* setting to point to the IBM i server. This could be specified as a domain name or an IP Address.
4. Test the definition by clicking the *Test* button:



5. Close the settings dialog by clicking the *OK* button.
6. With the WAM *iiiEmpEnquiry* open in the editor, open the **begin** WebRoutine in the *Design* view and *Run* in the browser. Your WAM will be run on the IBM i.

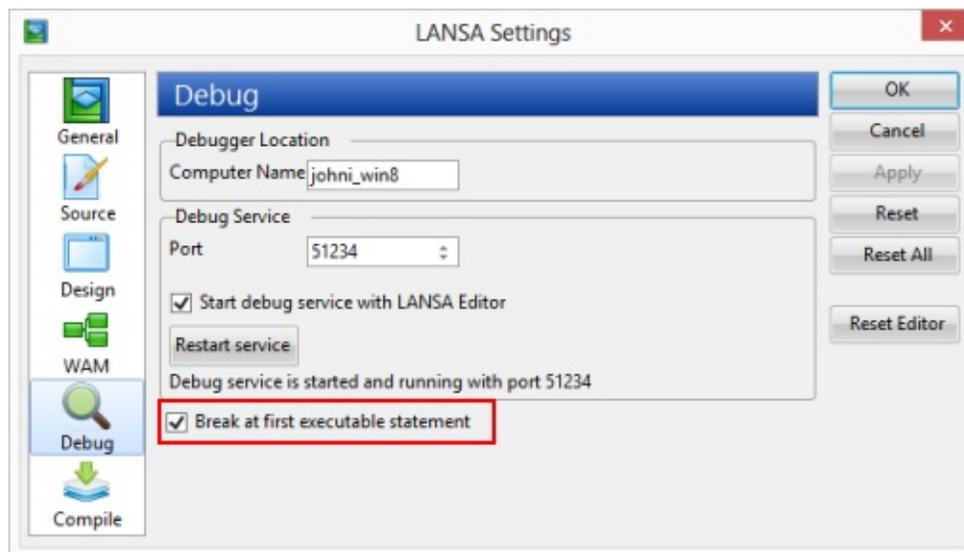


Step 3. Run a WAM in debug on the IBM i

With your WAM checked in and compiled on the IBM i server, you can now run it in debug mode from Visual LANSA, while it is running on the IBM i server.

The *Base Application URL* settings must point to your IBM i server.

1. You should have WAM iiiEmpEnquiry open in the editor, with the **begin** WebRoutine open in the *Design* view. Check the debug settings from the *File / Editor Options*, select the *Debug* icon.

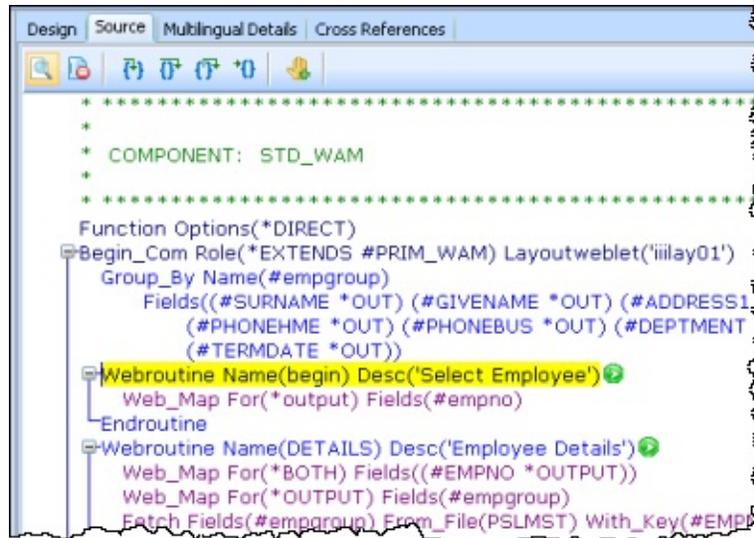


Ensure the *Break at first executable statement* option is selected. This means that initially you do not need to set any breakpoints in the program.



2. Run the begin WebRoutine using the  toolbar button.

The WebRoutine will execute in debug mode with the WAM executing on the IBM i server.



You can now continue to set breakpoints and use other debug features.

3. Refer to the exercise [WAM055 - Using LANSAs Debug](#), for further details if required.

Summary

Important Observations

- In this exercise you selected a group of objects for check in by name. You can also select objects to check in:
- By Task
- By building a static Editor List or maintaining a dynamic editor list for your changes
- By sorting the Repository tab by Date Modified and selecting changed objects
- By reference to your Last Opened tab
- If you have changed design of a WebRoutine's web page, simply check in the WAM again. No recompile will be necessary
- If you have changed a common layout weblet, check this in and the WAM's which use it will include it at run time.
- If you have changed cascading styles, check in the *External Resource* which defines this style file.

Tips & Techniques

- Remember that the details of a Field Visualization weblet are locked into the XSL when the WebRoutine web page is created and saved. If a field visualization has been changed, from radio button group to dropdown for example, checking in the WAM will make your application will behave correctly without checking in the changed field visualization details. However, you should check the field in, to maintain a consistent application definition in the Master Repository.
- Remember, the web server is waiting for a response from your WAM, so the web server could time out if you spend too long stepping through code in debug.

What You Should Know

- How to check in your WAM and related definitions
- How to run WAMs on the IBM i server from Visual Lansa.

WAM120 - Using the Menu Bar Weblet

Objectives

- To demonstrate how to set up the jQuery UI Menubar weblet to provide a fixed application menu.



- The menu will link to four WAMs which handle employee enquiry and update.

To achieve this objective you will complete the following:

- [Step 1. Define the Applications Menu](#)
- [Step 2. Test the Applications Menu](#)
- [Summary](#)

Before You Begin

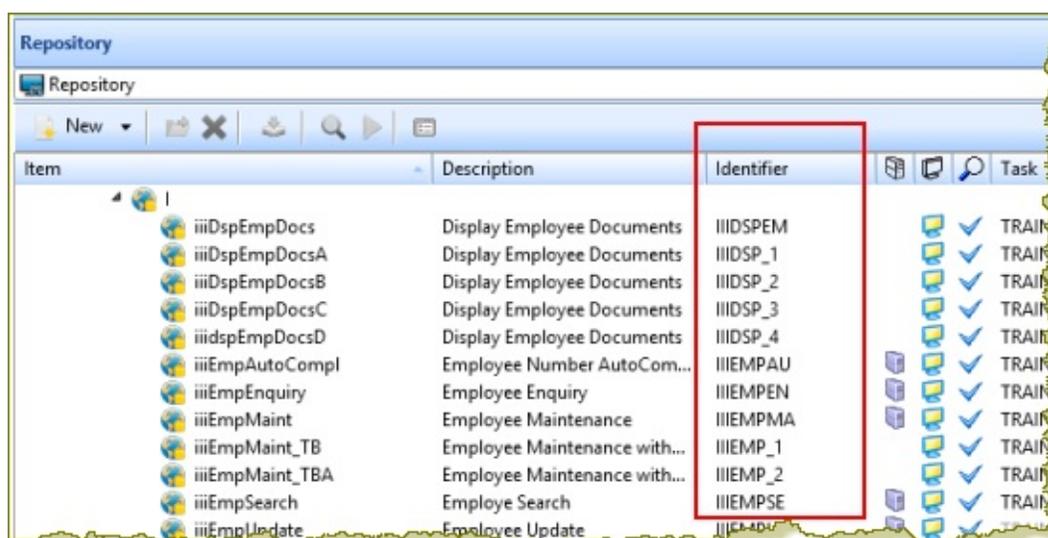
You should complete exercises WAM025, WAM030, WAM035 WAM040 and WAM045 which create and then use the common layout **iiilay01**.

Step 1. Define the Applications Menu

The menu needs to be setup with WAM name and WebRoutine name for four WAMs. For the first four WAMs using the common layout, the information is as follows. Replace **iii** with your initials.

WAM Name	WebRoutine Name	Menu Caption
iiiEmpEnquiry	begin	Enquiry
iiiEmpUpdate	begin	Update
iiiEmpUpdate_MK2	begin	Update with Dropdowns
iiiDynamSelector	begin	Update with Dropdown using Selector

0. Before you continue with this exercise, display your WAMs on the *Repository* tab and note the Identifiers for the above WAMs:

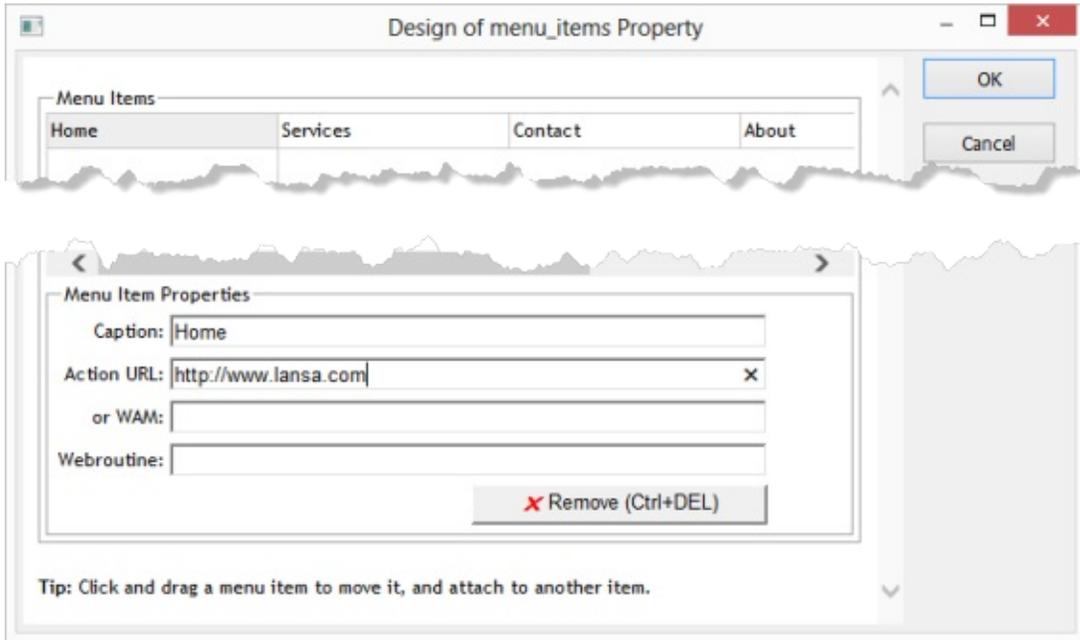


Visual LANS A allocates a unique 10 character identifier to each component you create (if Long Names is enabled at partition level). When calling the WAM directly in a URL, the Identifier, not the long names must be used.

1. Open layout **iiilay01** in the editor. In exercise WAM025 you added the

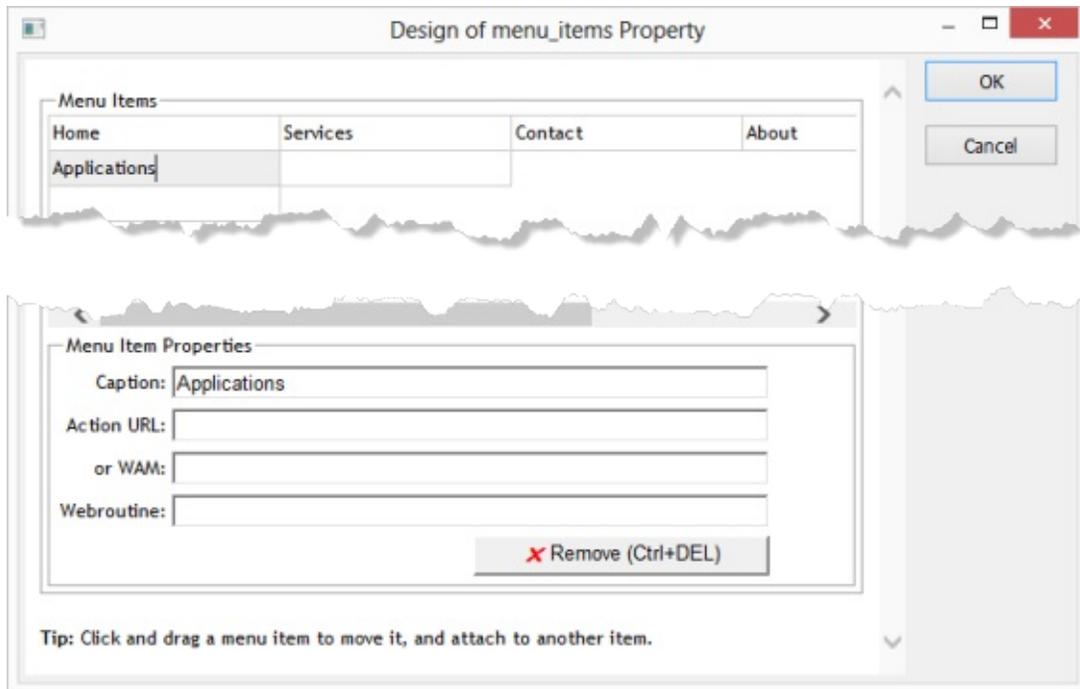
jQuery UI Menubar weblet to this layout and set up links to a number of LANSA website pages.

2. Select the menu bar weblet and use the Ellipsis button for the *menu_items* property to open the *Design of...* dialog, which should look like the following:

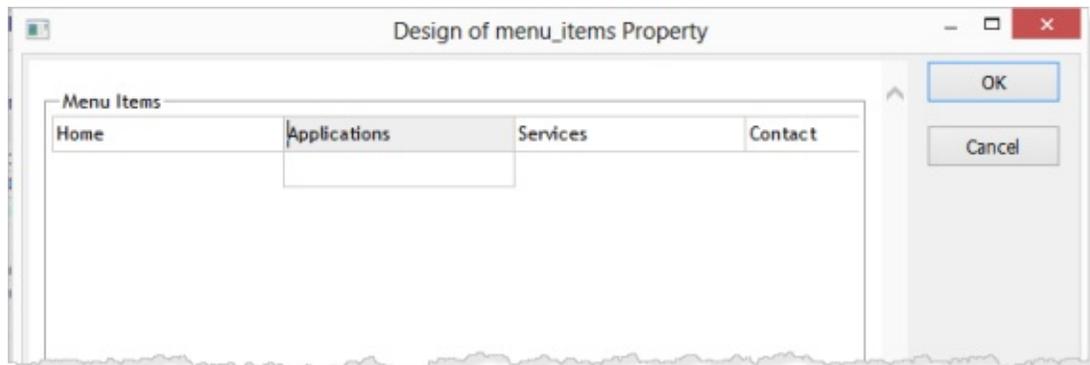


3. In this step you will define an *Applications* top level menu item, underneath menu item *Home* and then drag it into the top level, between *Home* and *Services*.

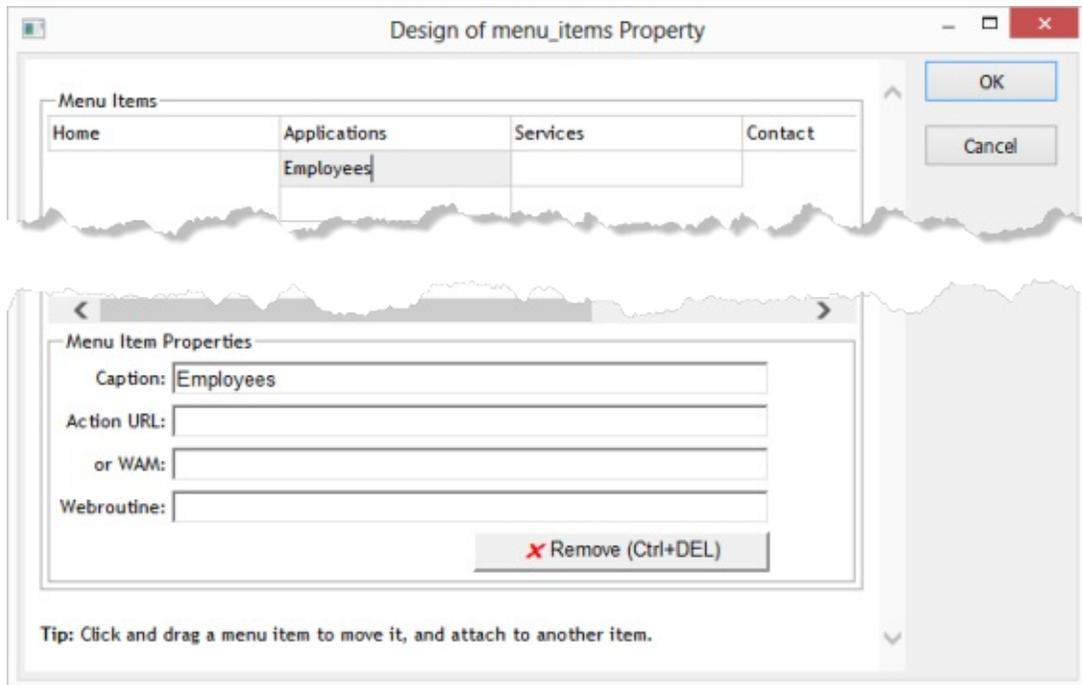
Click in the menu item below *Home* and in the Menu Item Properties, enter *Applications*. The *Design of menu_items Property* dialog should now look like the following:



- Click on the *Application* menu item and hold down the left mouse button to drag *Applications* into position between *Home* and *Services*. Position the cursor on the left hand side of *Services* when you release the left mouse button. Your design should look like the following:



- Click on the menu item *Applications* to display the sub-menu item underneath it. Enter **Employees** in the Menu Item Properties *Caption*. Your design should look like the following:



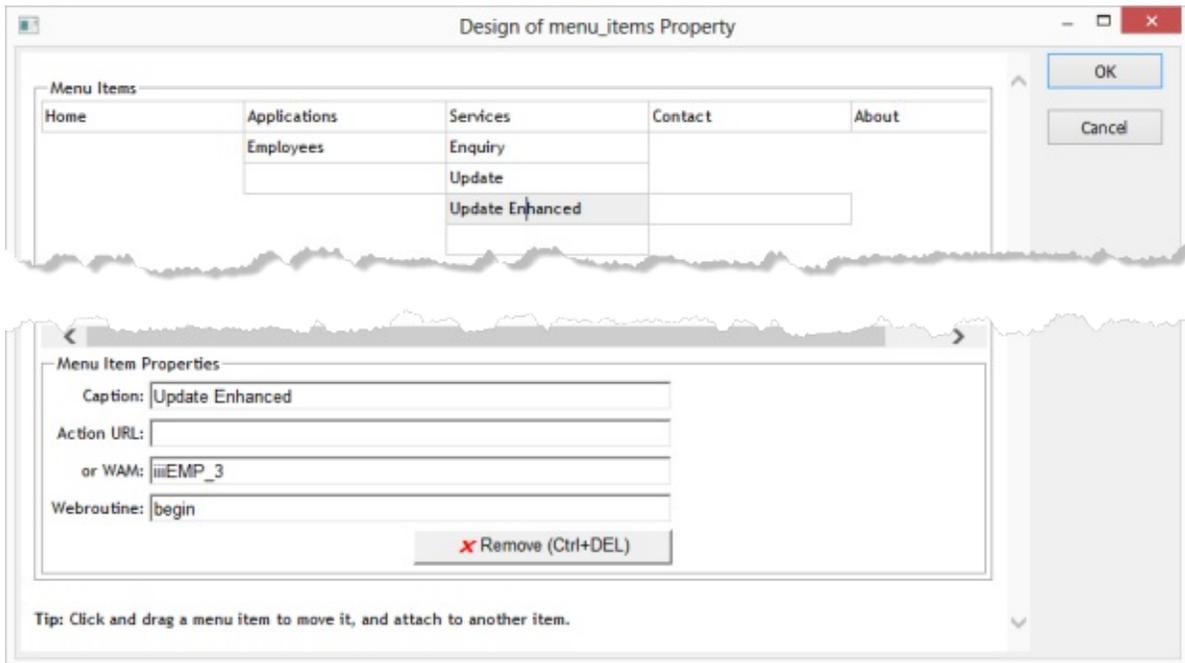
6. Click in the menu item to the right of *Employees*. This means you will be defining a sub-menu for *Employees*, which will be shown during the design step, under *Services*. Define *Enquiry* as:

Property	Value
Caption	Enquiry
WAM	iiiEMPEN
WebRoutine	begin

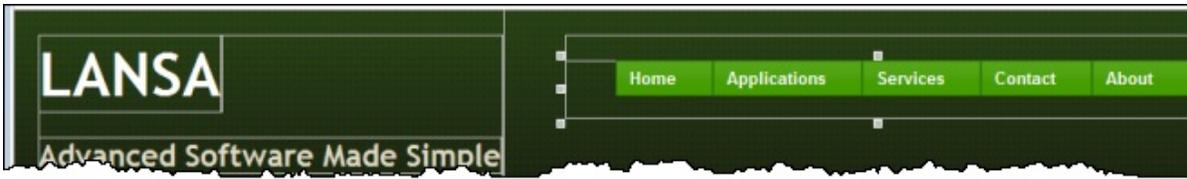
Note: Your WAM Identifier may well be different to the example shown above.
Your design should look like the following:



7. Continue by defining the next three sub-menu items for *Employees*, in the column under *Services*. Refer to the table in 1. above for details. Your dialog should now look like the following:



8. Click *OK* to save the changes and close the dialog. Your menu should look like the following:



9. Save your layout.

Step 2. Test the Applications Menu

1. Open WAM iiiEmpEnquiry in the editor and open the **begin** WebRoutine in the *Design* view. Run the WAM in the browser.
2. Use the Applications menu to run the *Employee Update* WAM:



3. Test all four menu sub-items to ensure that the links have been correctly defined.

Summary

Important Observations

- The menu bar weblet provides the functionality of a menu bar that can invoke other web pages including other webrouines.
- The menu bar can be arranged horizontally or vertically and the top level menu items can cause further menus to pop-up as the mouse moves over them.

Tips & Techniques

- The menu items can be defined in the Webroutine design using the menu item designer or they can be supplied at runtime in an RDMLX list.
- The weblet is based on the jQuery UI menu widget and requires jQuery and jQuery UI to operate (the weblet will automatically add the required external resources to the output HTML).

What Your Should Know

- How to define a static menu using the jQuery UI Menubar weblet.

WAM125 - Define a Dynamic Menu

Objectives

The jQuery UI Menubar can be defined dynamically using a working list. This exercise demonstrates how a login WAM could establish a custom menu for each group of users.

- A login WAM will handle two users ids ADMIN and USER.
- Only the ADMIN user will have access to the *Applications* menu

To meet these requirements the login WAM requires the following:

- Session Management must be enabled only if the login is valid
- The lists MNUSAVE and MNULIST which define the menu items, must be built with menu sub-items for *Applications*, only for the user ADMIN.
- All WAMs using this menu must be part of the same session group
- WAMs using this menu must transfer to the login WAM if a session is not active
- A working list must be output with fields which support the jQuery UI Menubar weblet.
- Two versions of the menu list are required:
- MNUSAVE will be the persistent version which is saved and restored by all WAMs using this menu.
- MNULIST will be the output version of the list which defines the menubar weblet menu items.
- The menu bar in the common layout **iiilay01** must be set up to use the menu list, MNULIST

To meet these objectives you will complete the following:

- [Step 1. Create the Login WAM](#)
- [Step 2. Redefine the Menubar in Layout iiilay01](#)
- [Step 3. Test your Login WAM](#)
- [Step 4. Make Application WAMs part of Session](#)
- [Step 5. Test the Applications Menu](#)
- [Step 6. Implement Menu for all Employee WAMs \(Optional\)](#)
- [Summary](#)

Step 1. Create the Login WAM

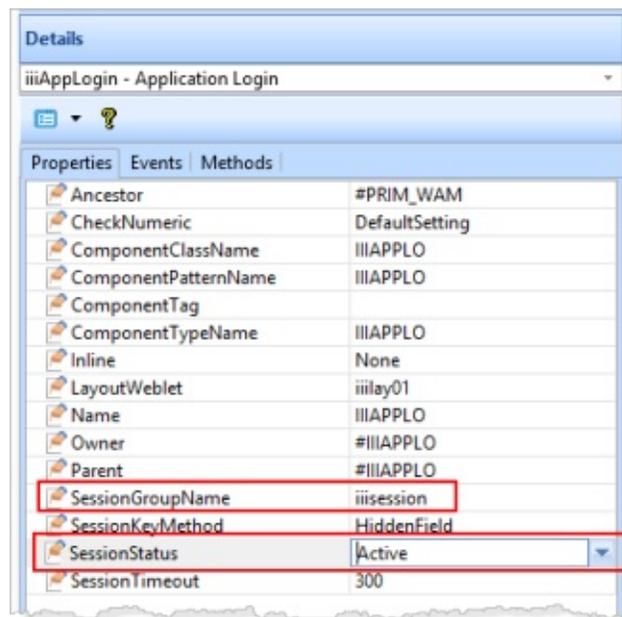
1. Create a new WAM with:

Name: **iiiAppLogin**

Description: **Application Login**

Layout weblet: **iiilay01**

2. Press F7 to display WAM properties. Use the *Details* tab to make *SessionStatus* active and define a *SessionGroupName* of **iiisession**.



3. Add the following definitions of fields and lists which will define the menu

```
Define Field(#mnuitmid) Type(*string) Length(3)
```

```
Define Field(#prtmitmid) Type(*string) Length(3)
```

```
Define Field(#mnucapt) Type(*char) Length(30) Input_Atr(LC)
```

```
Define Field(#mnuurl) Type(*string) Length(256)
```

```
Define Field(#mnuwam) Type(*char) Length(9)
```

```
Define Field(#mnuwrnme) Type(*char) Length(30)
```

```
Def_List Name(#mnusave) Fields(#mnuitmid #prtmitmid #mnucapt #mnuurl #m
```

```
Def_List Name(#mnulist) Fields(#mnuitmid #prtmitmid #mnucapt #mnuurl #mn
```

4. Add the following WEB_MAPs:

```
Web_Map For(*output) Fields((#mnulist *private))
```

```
Web_Map For(*none) Fields(#mnusave) Options(*persist)
```

```
Web_Map For(*both) Fields((#stdrentry *hidden))
```

Note the way that the map for list MNUSAVE is defined. This is the persistent version of the menu list which will be shared by all WAMs making up this application.

5. Define a WebRoutine **login**, which has an onentry parameter of *sessionstatus_none. This WebRoutine will establish a session for valid users and execute the subroutines to build the menu list.

```
Webroutine Name(login) Onentry(*SESSIONSTATUS_NONE)
Web_Map For(*both) Fields(#userid #passwd)
Message Msgtxt('Enter your user id and password')
Endroutine
```

Note that the fields USERID and PASSWD are mapped for *both. If these fields do not exist in the Repository, define them in your WAM as:

```
Define Field(#USERID) Type(*char) Length(10)
Define Field(#PASSWD) Type(*char) Length(10) Input_Atr(ND)
```

6. Define a method routine **AddToMenu** which will add each entry to the menu list MNUSAVE. It requires six input parameters which will define the required menu field entries. You will find this information in the *Web Applications Modules Guide / 8.1.18 Menubar (std_menubar)*. Your code should look like the following:

```
Mthroutine Name(AddToMenu)
Define_Map For(*input) Class(#STD_STRNG) Name(#itmid)
Define_Map For(*input) Class(#STD_STRNG) Name(#pitmid)
Define_Map For(*input) Class(#STD_DESC) Name(#caption)
Define_Map For(*input) Class(#STD_STRNG) Name(#URL)
Define_Map For(*input) Class(#STD_DESCS) Name(#WAM)
Define_Map For(*input) Class(#STD_DESCS) Name(#WRNAME)
#mnuitmid := #itmid
#prtmitmid := #pitmid
#mnucapt := #caption
#mnuurl := #URL
#mnuwam := #WAM
#mnuwrnme := #WRNAME
Add_Entry To_List(#mnusave)
```

Endroutine

If necessary add field STD_STRNG to the Repository as a STRING field, length 512.

7. Define a subroutine **buildmenu** which will add the standard menu entries which will be available for all users. This subroutine will be invoked once when the user logs in, so it should begin by clearing MNUSAVE. It should call the method routine AddToMenu to define and add each menu entry. Your code should look like the following:

```
Subroutine Name(buildmenu)
Clr_List Named(#mnusave)
* Home
#com_owner.addtomenu Itmid('1') Pitmid("") Caption('Home') Url('http://www.
* Applications
#com_owner.addtomenu Itmid('2') Pitmid("") Caption('Applications') Url("") Wa
* Support
#com_owner.addtomenu Itmid('3') Pitmid("") Caption('Support') Url('http://www
* Contact Us
#com_owner.addtomenu Itmid('4') Pitmid("") Caption('Contact Us') Url('http://v
* About
#com_owner.addtomenu Itmid('5') Pitmid("") Caption('About') Url('http://www.
*
Endroutine
```

8. Define a subroutine buildapps which will only be executed for the ADMIN user, which adds entries for the Employees menu and sub-menu items. It should not clear the list MNUSAVE. Your code should look like the following.

As before you must look in the *Repository* for your WAMs and note their Identifiers and replace the values shown in the code below..

```
Subroutine Name(buildapps)
* Employees
#com_owner.addtomenu Itmid('50') Pitmid('2') Caption('Employees') Url("")
Wam("") Wrname("")
* enquiry
#com_owner.addtomenu Itmid('51') Pitmid('50') Caption('Enquiry') Url("")
Wam(IIIEMPEN) Wrname(BEGIN)
```

```

* Update
#com_owner.addtomenu Itmid('52') Pitmid('50') Caption('Update') Url("")
Wam(IIIEMPUP) Wrname(BEGIN)
* Update with Dropdowns
#com_owner.addtomenu Itmid('53') Pitmid('50') Caption('Update with
Dropdowns') Url("") Wam(IIIEMP_3) Wrname(BEGIN)
* Update using Dropdown with Selector
#com_owner.addtomenu Itmid('54') Pitmid('50') Caption('Update using
Dropdown with selector') Url("") Wam(IIIEMP_4) Wrname(BEGIN)
Endroutine

```

Note: All menu entries must have a unique id. Sub-menu items must also have the correct parent id.

9. Define a new WebRoutine **welcome**. The **login** WebRoutine will transfer to this after a valid login. Welcome will clear and populate the output menu list MNULIST and display "welcome" messages. Your code should look like the following:

```

Webroutine Name(welcome)
Web_Map For(*input) Fields(#userid)
Clr_List Named(#mnulist)
Selectlist Named(#mnusave)
Add_Entry To_List(#mnulist)
Endselect
Message Msgtxt('Welcome ' + #USERID) Type(*STATUS)
Message Msgtxt('You are logged into the Personnel Applications System') Typ
Endroutine

```

Note that the USERID is mapped into this WebRoutine so that it's available in the welcome message.

10. In this step you will complete the **login** WebRoutine.

The login button will be set up to return STDREENTRY with a value of L.

A CASE loop for field USERID will check password, make the session status active, and execute one or both build menu subroutines. It will then transfer to the **welcome** WebRoutine. Your completed **login** WebRoutine should look like the following:

```

Webroutine Name(login) Onentry(*SESSIONSTATUS_NONE)

```

```

Web_Map For(*both) Fields(#userid #passwd)
Message Msgtxt('Enter your user id and password')
If (#stdrentry = L)
Case (#userid)
When (= ADMIN)
If (#passwd = '14MIN')
#com_owner.sessionstatus := active
Execute Subroutine(buildmenu)
Execute Subroutine(buildapps)
Transfer Toroutine(welcome)
Else
Message Msgtxt('Password incorrect for this user')
Endif
When (= USER)
If (#passwd = 'US5R')
#com_owner.sessionstatus := active
Execute Subroutine(buildmenu)
Transfer Toroutine(welcome)
Else
Message Msgtxt('Password incorrect for this user')
Endif
Endcase
Endif
Endroutine

```

11. Compile your WAM.

12. Open the **login** WebRoutine in the *Design* view.

- a. Add a row to the bottom of the fields table
- b. Add a push button with image to the bottom right hand cell
- c. Set up the push button as follows:

Property	Value
caption	Log In
left_relative_image	icons/normal/16/user_16.png
on_click_wrname	Login

submitExtraFields

Field Name: STDREENTRY

Literal Value: L

13. Save your changes.

Step 2. Redefine the Menubar in Layout **iiilay01**

1. Open the layout **iiilay01** in the editor.
2. Select the menu bar weblet.
3. Open the *Design of ...* dialog for the *menu_items* property and delete all the hard coded menu items. Click *OK* to save the changes.
4. Define the *listname* property as **MNULIST**.
5. Save your changes.

Step 3. Test your Login WAM

1. Make a note of the userid and password values which have been hard coded. Execute the login WebRoutine and login as ADMIN. The menu bar should look like the following:



2. If you are experiencing problems, try using debug to investigate.
3. Run the **Login** WebRoutine again and log in as USER. The menu bar should display the *Applications* top level menu item, but no sub-menu items will be shown.

Step 4. Make Application WAMs part of Session

In this step you will enable session management in WAM `iiiEmpEnquiry` and make it share the session group `IISESSION`. The WAM will need to map the saved and output menu lists in the same way as already implemented in the log in WAM `iiiAppLogin`.

1. Open WAM `iiiEmpEnquiry` in the editor. Press F7 to display the WAM properties and make `SessionStatus` **active** and `SessionGroupName`, **IISESSION**.
2. Copy and paste the following field, list and `web_maps` statements from `iiiAppLogin`. In a real application the menu fields would be defined in the *Repository*.

```
Define Field(#mnuitmid) Type(*string) Length(3)
Define Field(#prtmitid) Type(*string) Length(3)
Define Field(#mnucapt) Type(*char) Length(30) Input_Atr(LC)
Define Field(#mnuurl) Type(*string) Length(256)
Define Field(#mnuwam) Type(*char) Length(9)
Define Field(#mnuwrnme) Type(*char) Length(30)
Def_List Name(#mnusave) Fields(#mnuitmid #prtmitid #mnucapt #mnuurl #m
Def_List Name(#mnulist) Fields(#mnuitmid #prtmitid #mnucapt #mnuurl #mn
Web_Map For(*output) Fields((#mnulist *private))
Web_Map For(*none) Fields(#mnusave) Options(*persist)
```

3. Add an event handling routine for `SessionInvalid`. This will transfer to WAM **iiiAppLogin** and WebRoutine **login** if any WebRoutine is invoked without a session being established or after a session has timed out. Change **iii** to your initials.

```
EVTROUTINE HANDLING(#COM_OWNER.SessionInvalid) OPTIONS(*N
TRANSFER TOROUTINE(#iiiAppLogin.login)
ENDROUTINE
```

4. Change the initial WebRoutine in **iiiEmpEnquiry**, in this case it is WebRoutine **begin**, to clear list `MNULIST` and populate it from `MNUSAVE`. Your WebRoutine **begin** code should now look like the following. Changes are shown in red.

```
Webroutine Name(begin) Desc('Select Employee')
```

```
Web_Map For(*output) Fields(#empno)
Clr_List Named(#mnulist)
Selectlist Named(#mnusave)
Add_Entry To_List(#mnulist)
Endselect
Endroutine
```

Remember list MNUSAVE is automatically restored when the WAM runs, because session management is active and the list is mapped as persistent data. Both field values and lists may be defined as persistent data.

5. Compile iiiEmpEnquiry.

Step 5. Test the Applications Menu

1. Execute the **login** WebRoutine in `iiiAppLogin` in the browser. Log in as user ADMIN.
2. From the menu bar select *Applications / Employees / Enquiry* to execute WebRoutine **begin** in WAM **iiiEmpEnquiry**. With the *Enquiry* WAM running check that the *Applications* menu is shown. Close the browser.
3. Execute the **begin** WebRoutine in **iiiEmpEnquiry** in the browser. Your application should transfer to the log in WAM so that a session can be established. The transfer is handled via the event handling routine for `SessionInvalid`.

Step 6. Implement Menu for all Employee WAMs (Optional)

1. If the changes made in [Step 4. Make Application WAMs part of Session](#) are applied to WAMs `iiiEmpUpdate`, `iiiEmpUpdate_MK2` and `iiiDynamSelector` then all four Employee applications may be run from the menu and will display the correct menu.

Summary

Important Observations

- In a real application you could define login authority in a database file with a small application to manage them.

Tips & Techniques

- The menu bar has a *submit_selected_to* property, which will send the *menu id* in the defined field to the WebRoutine which is invoked. This could enable the WebRoutine to behave in a specific way when it is first invoked from the menu.

What You Should Know

- How to implement the jQuery UI Menubar using a working list.

WAM130 - Output a Web Page to a File

Objectives

Running a WAM using the X_RUN command, enables the WebRoutine output to be saved to a file. This enables permanent (instead of dynamic) web pages to be created from your application. This may be an advantage when seeking to optimize search engines searches over your public web site. It can also be used as a method of saving results which will be fixed for a period of time (for example monthly statistics) rather than repeatedly running a WAM to calculate the same set of results for each user enquiry.

Output to a file from a WebRoutine is supported for Windows, IBM i and Linux servers but you need to be aware of differences with the X_RUN command parameters used. See a [Saving a WAM's Output to a File](#) for more details.

Since the WebRoutine output goes directly to an HTML file, it can only contain output generated by the WebRoutine.

To demonstrate WAM output to a file on Windows and IBM i (if available) you will complete the following:

- [Step 1. Output Employee Enquiry to a File](#)
- [Step 2. Run WAM to output to a file in Windows](#)
- [Step 3. Run WAM to output a file on IBM i](#)
- [Summary](#)

Before You Begin

Complete the introductory exercises, WAM005, WAM010, WAM015 and WAM020 before starting this exercise.

Step 1. Output Employee Enquiry to a File

This exercise uses a simple enquiry WAM. When running in the browser, an employee number is requested via WebRoutine **begin**. A second WebRoutine **details** is invoked which displays employee data.

Using X_RUN the details WebRoutine is invoked, passing employee number as a parameter on the X_RUN, the web page containing data for the requested employee can be written to a file.

To execute a WAM, the parameters required by the X_RUN command for Windows are as follows:

Argument Value

PROC	*WAMSP is the value to activate the "output to file" function
WMOD	WAM Name
WRTN	WebRoutine Name
WAML	Mark up language, e.g. LANSA:HTML
PART	Partition
LANG	Language, e.g ENG
USER	User. Optional for some platforms
WASP	Output file path where the output will be saved. Format depends on platform. e.g. for Windows c:\temp\wam_output.html

The format of the X_RUN command for Windows is:

```
X_RUN PROC=*WAMSP WMOD=IIIEmpEnqToFile WRTN=DETAILS . . .
```

The X_RUN command does not have a parameter to input a field and value, but it does have a user defined parameter UDEF, which is a 256 long character field. If the UDEF parameter is used, the WebRoutine will require a simple modification to retrieve the UDEF value using the Built-in Function

GET_SESSION_VALUE. If the value returned by the BIF is non blank, the DETAILS WebRoutine can retrieve employee data using the session value as employee number.

1. Create a new Employee Enquiry WAM by copying the following code.

Name: iiiEmpEnqToFile

Description: Employee Enquiry to File

Layout Template: iiilay01

Function Options(*DIRECT)

Begin_Com Role(*EXTENDS #PRIM_WAM)

Group_By Name(#empgroup) Fields((#SURNAME *OUT) (#GIVENAME *(

Webroutine Name(begin) Desc('Select Employee')

Web_Map For(*output) Fields(#empno)

Endroutine

Webroutine Name(DETAILS) Desc('Employee Details')

Web_Map For(*BOTH) Fields((#EMPNO *OUTPUT))

Web_Map For(*OUTPUT) Fields(#empgroup)

Fetch Fields(#empgroup) From_File(PSLMST) With_Key(#EMPNO) Val_Err

If_Status Is_Not(*OKAY)

Message Msgtxt('Employee not found')

Endif

Endroutine

End_Com

2. Add logic to the details WebRoutine to retrieve session value and if non blank, use as employee number.

Add the following code in WebRoutine details, immediately **before** the FETCH command.

Define Field(#retcode) Type(*char) Length(2)

Use Builtin(get_session_value) With_Args(UDEF) To_Get(#STD_QSEL #retc

If (#std_qsel *NE *blanks)

#empno := #std_QSEL.trim

Endif

3. Compile your WAM.
4. Open the **begin** WebRoutine in the design view.

- a. Add a right hand column to the table containing employee number.
 - b. Drag and drop a push button weblet into the new column.
 - c. Change the *caption* to **Select**.
 - d. Set the push button *on_click_wrname* property to invoke the **Details** WebRoutine.
 - e. The *submitExtraFields* property does not need to be specified.
 - f. Remove the place holder characters.
 - g. Save your changes.
5. Run the **begin** WebRoutine in the browser and ensure that the **details** WebRoutine executes normally when invoked from the **begin** web page. Enter an employee number such as A0090, A0070 or A1001.

Step 2. Run WAM to output to a file in Windows

This step will run the **details** WebRoutine in Windows using the X_RUN command and pass in the employee number using the UDEF, a user defined run time parameter.

The X_RUN.EXE program is in the Visual LANSA \execute folder, for example:

```
C:\Program Files\LANSA_D12\X_WIN95\X_LANSA\execute
```

You could add this path to your Windows Path environment variable. In which case the X_RUN program will be found, if it is run from a simple batch file. For this exercise you will create a DOS batch file using Notepad and set up the required path.

1. Open Notepad and add the following code:

```
cd\  
cd program files\lansa\x_win95\x_lansa\execute  
x_run PROC=*WAMSP WMOD=iiiEMP_4 WRTN=DETAILS WAML=LAN
```

Important: On the *Favorites / Last Opened* tab find your WAM iiiEmpEnqToFile and find its Identifier. This must be used in the X_RUN (i.e. replace WMOD=IIIEMP_4 in the above code).

If necessary change the change directory command (cd) to reflect your Visual LANSA installed path.

Review the X_RUN command carefully and correct it for your Visual Lansa path, partition name, user and output HTML file name.

The specific X_RUN parameters you may need to change are:

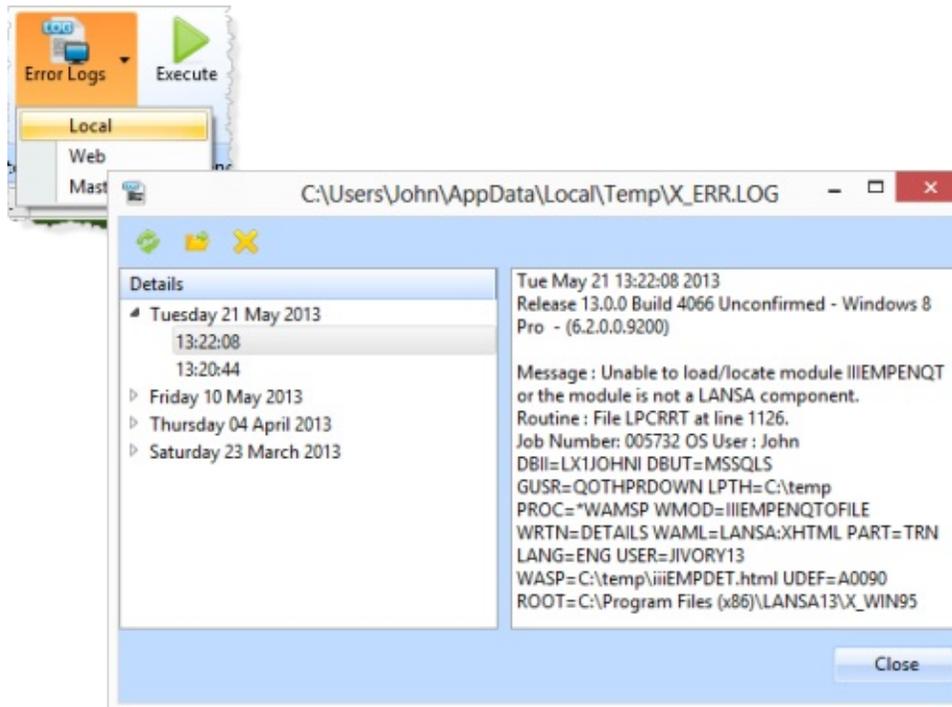
- WMOD – WAM Identifier
- PART – partition name
- USER – user name (use your Visual LANSA profile name)
- WASP – The output path and file name. Use your initials
- UDEF – The employee number

Create a c:\temp folder if necessary.

2. Use the *Save as Type:All* to save the file as iiiWAMOUT.bat in folder c:\temp

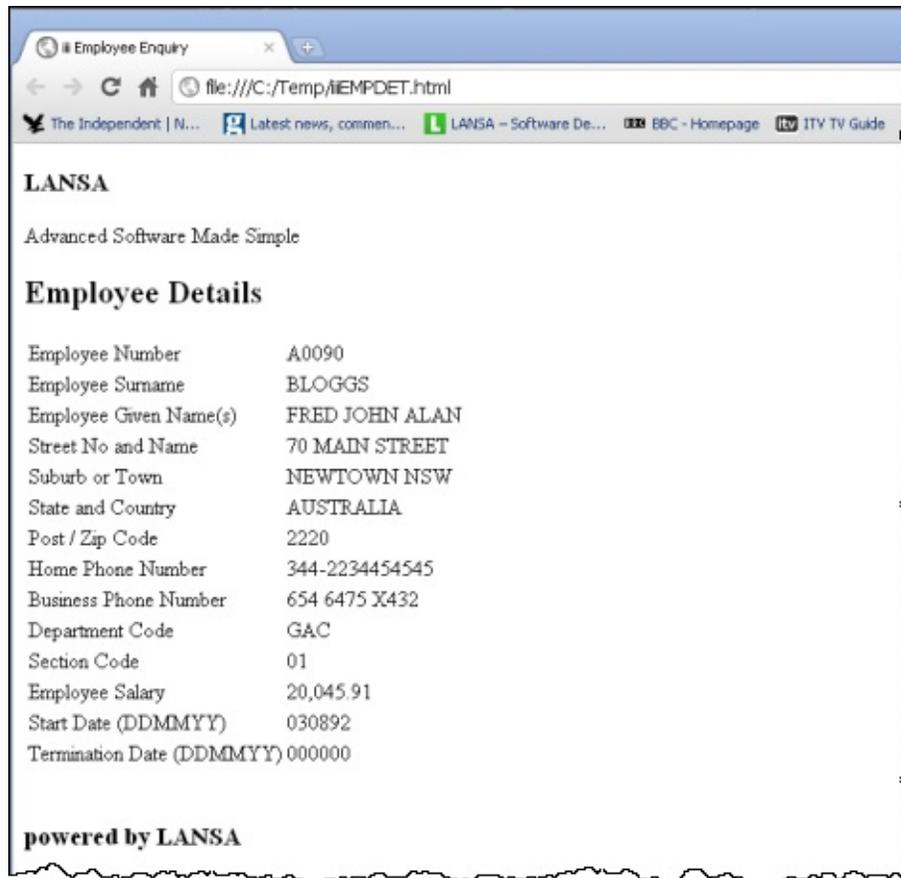
3. Navigate to c:\temp with Windows Explorer and double click on iiiWAMOUT.bat to run it. The DOS command prompt should briefly open and close.
4. Check folder c:\temp for the output file iiiEMPDET.html.
5. If your run does not output the file iiiEMPDET.html, check the batch file carefully for errors and try again.

When running your WAM locally, you can also check for errors in the *Visual Lansa Error Log*.



See also the *Web Runtime Error Log* from the *Error Logs* menu.

6. When your WAM has created the output file, double click on it to open it directly in your PC's default browser. It should look like the following:



No cascading style sheets have been applied.

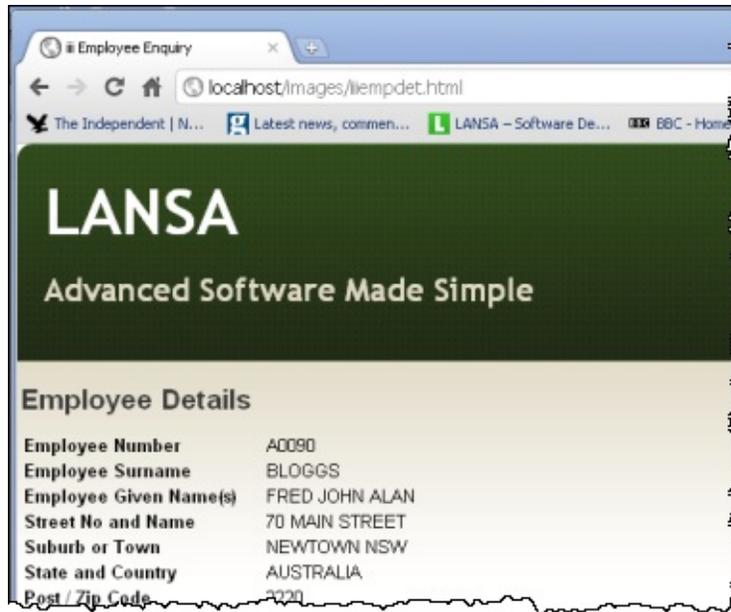
7. Copy the output file `iiiEMPDET.html` to your Visual LANS web server\images path, such as:

`C:\Program Files\LANS\WebServer\Images`

8. Open the file by putting the following URL into your browser address bar:

`http://localhost/images/iiiempdet.html`

Your web page should now be displayed correctly formatted, because the cascading style sheets have been applied. See below:



Step 3. Run WAM to output a file on IBM i

1. Check in and compile WAM iiiEmpEnqToFile and its layout. Also check in the common layout iiilay01, if it has not been checked in before. Review the *Check In* tab to ensure the Check In was successful
2. Start an emulator session. Log on with your Visual LANSA user id and password. Call the command processor with

CALL QCMD

Display a full screen by pressing F11.

3. Enter the following command:

```
LANSA REQUEST(X_RUN) PARM01('PROC=*WAMSP')  
PARM02('WMOD=IIIEMP_4') PARM03('WRTN=DETAILS')  
PARM04('WAML=LANSA.XHTML') PARM05('PART=TRN')  
PARM06('LANG=ENG') PARM07('/WASP=/TMP/IIIEMPDET.HTML')  
PARM08('UDEF=A0090')
```

Correct the call command for your WAM Identifier name, partition name and user name. Check that your IBM i has a /tmp folder in the IFS or use a suitable alternative folder.

Press Enter to run the WAM.

4. Use WRKLNK to view the /tmp folder and find your output HTML file.

Note: Your call command should have run successfully because your library list was correct, so that the X_RUN program was found. You logged on a LANSA developer and therefore had the correct library list.

If you were creating a CL program to run this job, you would need to establish the correct library list. For example:

```
ADDLIB <PGMLIB>  
LANSA REQUEST(X_RUN) PARM01('PROC=*WAMSP')  
PARM02('WMOD=IIIEMP_4') PARM03('WRTN=DETAILS')  
PARM04('WAML=LANSA.XHTML') PARM05('PART=TRN')  
PARM06('LANG=ENG') PARM07('/WASP=/TMP/IIIEMPDET.HTML')  
PARM08('UDEF=A0090')
```

where: <PGMLIB> is your LANSA program library.

You could also consider error handling in the modified **details** WebRoutine.

What to do if the UDEF variable is non-blank but the FETCH employee record

is unsuccessful?

Once again, if you have access to the /tmp folder from Windows explorer the iiiempdet.html can be opened in the browser, but it will be unformatted.

5. Copy the output HTML file to the appropriate IBM i web server /images folder. For example:

```
CPY OBJ('/tmp/iiiempdet.html')  
TODIR('/lansa_<PGMLIB>/webserver/www/htdocs/  
s/images') REPLACE(*YES)
```

where <PGMLIB> is your LANSAs program library. You may need to make other adjustments for your IFS set up.

6. Put the following URL into your browser address bar:

<http://10.44.10.238/images/iiiempdet.html>

where: 10.44.10.238 is your IBM i IP Address (or use its domain name). Your saved web page should now be displayed correctly formatted as the required style sheets (CSS) will have been included.

Summary

Important Observations

- For further details on using this facility see the [Saving a WAM's Output to a File](#).

Tips & Techniques

- The user defined parameter accepted by X_RUN is a 256 long character string. You could input a more complex key (DEPARTMENT and SECTION to identify a section record for example) by concatenating them into UDEF. The WebRoutine would then split the UDEF variable into its known parts.

What You Should Know

- How to implement the *Save WAM output to a file* feature.

WAM135 - Using the Google Static Maps API

Objective

This exercise provides a very simple example of using address information to display a street map using the Google Static Map API. The Google Maps facility can be used in many more sophisticated ways. This example is intended to illustrate that provided you have good address data, a static map can quite easily be displayed.

The information to build this example was taken from:

http://code.google.com/apis/maps/documentation/staticmaps/#quick_example

See the following URL for full documentation:

<http://code.google.com/apis/maps/documentation/staticmaps/>

The supplied example shows how the following `` tag will display a street map around the Brooklyn Bridge, New York.

```
<img src=http://maps.googleapis.com/maps/api/staticmap?
center=Brooklyn+Bridge,New+York,NY&zoom=14&size=512x512&maptype
&markers=color:blue%7Clabel:S%7C40.702147,-74.015794&markers=color:;
&markers=color:red%7Ccolor:red%7Clabel:C%7C40.718217,-73.998284&sei
```

A simple employee enquiry will display a static street map for the employee's address.

To meet this objective you will complete the following:

- [Step 1. Create an Employee Enquiry WAM](#)
- [Step 2. Add logic to set up URL to Google Map Service](#)
- [Summary](#)

Before You Begin

Complete the introductory exercises, WAM005, WAM010, WAM015 and WAM020.

Step 1. Create an Employee Enquiry WAM

1. Create a new WAM with:

Name: **iiiEmpEnqMap**

Description: **Employee Enquiry with Map**

Layout Template: **iiilay01**

Create the WAM based on the following code:

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_WAM) Layoutweblet('iiilay01')
Group_By Name(#empgroup) Fields((#SURNAME *OUT) (#GIVENAME *(
Define #ADDRESS Type(*string) Length(1000)
Webroutine Name(begin) Desc('Select Employee')
Web_Map For(*output) Fields(#empno)
Endroutine
Webroutine Name(DETAILS) Desc('Employee Details')
Web_Map For(*BOTH) Fields((#EMPNO *OUTPUT)
(#ADDRESS *hidden))
Web_Map For(*OUTPUT) Fields(#empgroup)
Fetch Fields(#empgroup) From_File(PSLMST) With_Key(#EMPNO) Val_Err
If_Status Is_Not(*OKAY)
Message Msgtxt('Employee not found')
Transfer Toroutine(begin)
Endif
Endroutine
End_Com
```

2. Compile the WAM.
3. Add a push button to the begin web page.
 - a. Open the **begin** WebRoutine in the *Design* view.
 - b. Add a column to the fields table.
 - c. Add a push button in the new column.
 - d. Set up the button with a caption of **Details** and an *on_click_wrname* of **details**.
 - e. Set up the *submitExtraFields* property to submit the field **EMPNO** with

the **Field Value**.

- e. Remove the * placeholder characters.
 - f. Save your changes.
4. Open the **Details** WebRoutine in the *Design* view.
- a. Add a column to the fields table
 - b. With the top row in the new column selected, set its *rowspan* property to **14**.
 - c. Click inside the new column and use the context menu to *Insert HTML / Div*. Your design should look like the following:



The screenshot shows a web form titled "Employee Details" with a table of fields and values. The fields are: Employee Number, Employee Surname, Employee Given Name(s), Street No and Name, Suburb or Town, State and Country, Post / Zip Code, Home Phone Number, Business Phone Number, Department Code, Section Code, Employee Salary, Start Date (DDMMYY), and Termination Date (DDMMYY). The values are: ABCDE, ABCDEFGHIJKLMNOPQRST, ABCDEFGHIJKLMNOPQRST, aAbBcCdDeEfGhHiIjJkKlLm, aAbBcCdDeEfGhHiIjJkKlLm, aAbBcCdDeEfGhHiIjJkKlLm, 123456, ABCDEFGHIJKLMNO, ABCDEFGHIJKLMNO, Value DEPARTMENT, AB, 123,456,789.12, 12/34/56, and 12/34/56. A vertical scrollbar is visible on the right side of the table.

Field	Value
Employee Number	ABCDE
Employee Surname	ABCDEFGHIJKLMNOPQRST
Employee Given Name(s)	ABCDEFGHIJKLMNOPQRST
Street No and Name	aAbBcCdDeEfGhHiIjJkKlLm
Suburb or Town	aAbBcCdDeEfGhHiIjJkKlLm
State and Country	aAbBcCdDeEfGhHiIjJkKlLm
Post / Zip Code	123456
Home Phone Number	ABCDEFGHIJKLMNO
Business Phone Number	ABCDEFGHIJKLMNO
Department Code	Value DEPARTMENT
Section Code	AB
Employee Salary	123,456,789.12
Start Date (DDMMYY)	12/34/56
Termination Date (DDMMYY)	12/34/56

- d. With the DIV still selected, select the *Details* tab. Expand its *Style* property and set the following properties:

Property	Value
Height	340px
Margin-left	50px
Width	340px

Your design should look like the following:

Employee Details

Employee Number	ABCDE	
Employee Surname	ABCDEFGHJKLMNOPQRST	
Employee Given Name(s)	ABCDEFGHJKLMNOPQRST	
Street No and Name	aAbBcCdDeEffGhHiIjJkKlLm	
Suburb or Town	aAbBcCdDeEffGhHiIjJkKlLm	
State and Country	aAbBcCdDeEffGhHiIjJkKlLm	
Post / Zip Code	123456	
Home Phone Number	ABCDEFGHJKLMNOP	
Business Phone Number	ABCDEFGHJKLMNOP	
Department Code	Value DEPTMENT	
Section Code	AB	
Employee Salary	123,456,789.12	
Start Date (DDMMYY)	12/34/56	
Termination Date (DDMMYY)	12/34/56	

- e. Delete the * place holder characters from this column.
- f. Save your changes.
- g. Drop a *clickable image* weblet into the DIV.
- h. Select the *clickable image* and set its *absolute_image_path* to **#ADDRESS**. Enter **#ADDRESS** in the *XPath Expression* window.
- i. Save your changes.

Step 2. Add logic to set up URL to Google Map Service

In this step you will extend the **Details** WebRoutine to use the employee address fields to set up the URL string which will be output as the #ADDRESS field.

Review the example Google URL provided in *Objectives* above, and note the Brooklyn Bridge address values in that string.

An employee address such as:

70 MAIN STREET (field ADDRESS1)

NEWTOWN NSW (field ADDRESS2)

AUSTRALIA (field ADDRESS3)

Must be used to produce a string inside the Google URL such as:

MAIN+STREET,NEWTOWN,NSW,AUSTRALIA

Your code needs to do the following:

- Remove house number
- Remove leading spaces
- Add + between words in an address line
- Add a comma between elements in field ADDRESS2

1. Define two work fields at WAM level (below the Begin_com):

Define Field(#addr1) Reffld(#address1)

Define Field(#addr2) Reffld(#address2)

2. In the **Details** WebRoutine add the following logic, immediately after the FETCH statement.

a. Remove numbers and first space character from ADDRESS1

```
#addr1 := #address1.removecharacters( "1234567890" ).remove( " " )
```

b. Replace space character (between words) with a +

```
#addr1 := #addr1.replace( " ", "+" )
```

c. Replace space character in ADDRESS2 with comma.

```
#addr2 := #address2.replace( " ", "," )
```

d. Define the field ADDRESS as the complete Google URL

```
#ADDRESS := 'http://maps.googleapis.com/maps/api/staticmap?center=' +  
#addr1 + ',' + #addr2 + ',' + #address3 +  
'&zoom=14&size=340x340&maptype=roadmap&markers=color:blue%7C' +  
#addr1 + ',' + #addr2 + ',' + #address3 + '&sensor=false'
```

Note that the size value in the URL has been changed to 340x340.

All other values use the supplied example values for example, for color. These could all be adjusted by reference to the Google web site for this service.

3. Compile and test your WAM. Your details web page should now look like the following:

Employee Details	
Employee Number	A0090
Employee Surname	BLOGGS
Employee Given Name(s)	FRED JOHN ALAN
Street No and Name	70 MAIN STREET
Suburb or Town	NEWTOWN NSW
State and Country	AUSTRALIA
Post / Zip Code	2220
Home Phone Number	344-2234454545
Business Phone Number	654 6475 X432
Department Code	
Section Code	
Employee Salary	20,045.91
Start Date (DDMMYY)	030892
Termination Date (DDMMYY)	000000



Summary

Important Information

- You should read the terms and conditions for this service.

Tips & Techniques

- There are many services like this example which can be used to enhance your web applications.

What You Should Know

- How to implement the Google Static Map service.

付録A. XSLとXMLの準拠

LANSAs for the WebのWAMは、W3Cの以下のXML規格に準拠していません。

- XML 1.0規格：<http://www.w3.org/TR/REC-xml/>
- XMLのネームスペース：<http://www.w3.org/TR/REC-xml-names/>
- XSLT 1.0規格：<http://www.w3.org/TR/xslt>
- XPath 1.0規格：<http://www.w3.org/TR/xpath>

付録B. WAM XML構造

実行時のWebroutine呼び出しにより生成されるXMLドキュメントには、標準の形式があります。Webroutineからのすべてのフィールドやリストなどの出力は、このXMLドキュメントに定義されます。このXMLドキュメントはWebroutine XSLスタイルシートを使った変換処理の入力として使用され、最終的なプレゼンテーション出力が生成されます。異なる特定のプレゼンテーション形式に正しく変換するためには、このXMLドキュメントが広く知られた標準の形式になっている必要があります。このXMLドキュメントは、以下のセクションに分かれています。

コンテキスト・セクション

このXMLドキュメントのコンテキスト・セクションには、Webroutineに関するコンテキスト情報が含まれています。WAMの名前、Webroutineの名前、Webroutineのタイトルなどの項目がこのセクションに含まれます。

オプション・セクション

オプション・セクションには、Webroutineの各種のオプションが含まれています。これらのオプションは変更可能で、特定の検証機能や表示機能を有効にするか無効にするかを設定できます。

メッセージ・セクション

メッセージ・セクションには、実行時にRDMLのMESSAGEコマンドを使用して出力するメッセージが含まれています。

フィールド・セクション

フィールド・セクションには、WebroutineのWEB_MAPステートメントで送信フィールドとして指定されているフィールドが含まれています。実行時に、送信用のフィールドの値がこのセクションに追加され、プレゼンテーション出力に変換されます。さらに、これらのフィールドのキャプション、説明や見出しも、実行時と設計時にこのセクションに追加されます。

リスト・セクション

リスト・セクションには、WebroutineのWEB_MAPステートメントで送信用のリストとして指定されているリストが含まれています。各リストの各フィールド列の見出しと各リスト行のフィールド値が、実行時にこのセクションに追加されます。見出しは、実行時の実際の行の値は含まれないものの、設計時にもこのセクションに含まれます。生成されたり

ストは、インラインされる場合は、属性がinline="true"となります。
JSONリストはCDATAセクションとして送信されます。JSONリストは通常JavaScriptにより実行されます。

XMLドキュメントの例

XMLドキュメントの例を以下に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Web application :PSLSYSWAM      Test      -->
<!-- Webroutine   :EmployeeEntry    Add an Employee      -->
<!-- Created by user :PCXUSER      -->
<!-- Timestamp    :2011-08-26T16:45:00      -->
<lxml:data xmlns:lxml="http://www.lansa.com/2002/XML/Runtime-Data">
  <lxml:context>
    <lxml:user-id>PCXUSER</lxml:user-id>
    <lxml:webapplication>PSLSYSWAM</lxml:webapplication>
    <lxml:webapplication-title>Test</lxml:webapplication-title>
    <lxml:webroutine>EmployeeEntry</lxml:webroutine>
    <lxml:webroutine-title>Add an Employee</lxml:webroutine-title>
    <lxml:service-name>PSLSYSWAM_EmployeeEntry</lxml:service-
name>
    <lxml:partition>DEM</lxml:partition>
    <lxml:language iso-lang="en">ENG</lxml:language>
    <lxml:images-path>/IMAGES</lxml:images-path>
    <lxml:action-request>/CGI-BIN/lansaweb</lxml:action-request>
  </lxml:context>
  <lxml:options>
    <lxml:option name="DBCS">>false</lxml:option>
    <lxml:option name="align-right">>true</lxml:option>
    <lxml:option name="check-numeric">>true</lxml:option>
    <lxml:option name="debug" />
    <lxml:option name="trace" />
    <lxml:option name="task" />
  </lxml:options>
  <lxml:external-resources>
    <lxml:script name="XWJQC" charset="iso-8859-
1" location="header">jquery/js/jquery 1.5.1.min.js</lxml:script>
    <lxml:script name="XWJQUI" charset="iso-8859-
1" location="header">jquery/js/jquery-ui-1.8.10.all.min.js</lxml:script>
```



```
<lxml:value />
</lxml:field>
<lxml:field name="ADDRESS1">
  <lxml:caption>
    <lxml:label>Address 1.....</lxml:label>
    <lxml:description>Street No and Name</lxml:description>
    <lxml:heading-1>Address line 1</lxml:heading-1>
    <lxml:heading-2 />
    <lxml:heading-3 />
  </lxml:caption>
  <lxml:value />
</lxml:field>
<lxml:field name="ADDRESS2">
  <lxml:caption>
    <lxml:label>Address 2.....</lxml:label>
    <lxml:description>Suburb or Town</lxml:description>
    <lxml:heading-1>Address line 2</lxml:heading-1>
    <lxml:heading-2 />
    <lxml:heading-3 />
  </lxml:caption>
  <lxml:value />
</lxml:field>
<lxml:field name="ADDRESS3">
  <lxml:caption>
    <lxml:label>Country</lxml:label>
    <lxml:description>State and Country</lxml:description>
    <lxml:heading-1>Country</lxml:heading-1>
    <lxml:heading-2 />
    <lxml:heading-3 />
  </lxml:caption>
  <lxml:value />
</lxml:field>
<lxml:field name="POSTCODE">
  <lxml:caption>
    <lxml:label>Post/zip code..</lxml:label>
    <lxml:description>Post / Zip Code</lxml:description>
    <lxml:heading-1>Post/zip</lxml:heading-1>
    <lxml:heading-2>Code</lxml:heading-2>
    <lxml:heading-3 />
  </lxml:caption>
  <lxml:value />
</lxml:field>
```

```
</xml:caption>
<xml:value />
</xml:field>
<xml:field name="PHONEHME">
  <xml:caption>
    <xml:label>Home phone.....</xml:label>
    <xml:description>Home Phone Number</xml:description>
    <xml:heading-1>Home phone</xml:heading-1>
    <xml:heading-2>Number</xml:heading-2>
    <xml:heading-3 />
  </xml:caption>
  <xml:value />
</xml:field>
<xml:field name="PHONEBUS">
  <xml:caption>
    <xml:label>Business ph....</xml:label>
    <xml:description>Business Phone Number</xml:description>
    <xml:heading-1>Business Phone</xml:heading-1>
    <xml:heading-2>Number</xml:heading-2>
    <xml:heading-3 />
  </xml:caption>
  <xml:value />
</xml:field>
<xml:field name="STARTDTER">
  <xml:caption>
    <xml:label>Start date.....</xml:label>
    <xml:description>Start date (YYMMDD)</xml:description>
    <xml:heading-1>Start</xml:heading-1>
    <xml:heading-2>Date</xml:heading-2>
    <xml:heading-3 />
  </xml:caption>
  <xml:value />
</xml:field>
<xml:field name="TERMDATER">
  <xml:caption>
    <xml:label>Term. date.....</xml:label>
    <xml:description>Termination Date (YYMMDD)
</xml:description>
    <xml:heading-1>Term.</xml:heading-1>
```

```

    <lxml:heading-2>Date</lxml:heading-2>
    <lxml:heading-3 />
  </lxml:caption>
  <lxml:value />
</lxml:field>
<lxml:field name="DEPARTMENT">
  <lxml:caption>
    <lxml:label>Department.....</lxml:label>
    <lxml:description>Department Code</lxml:description>
    <lxml:heading-1> Dept</lxml:heading-1>
    <lxml:heading-2> Code</lxml:heading-2>
    <lxml:heading-3 />
  </lxml:caption>
  <lxml:value />
</lxml:field>
<lxml:field name="SECTION">
  <lxml:caption>
    <lxml:label>Section.....</lxml:label>
    <lxml:description>Section Code</lxml:description>
    <lxml:heading-1> Section</lxml:heading-1>
    <lxml:heading-2> Code</lxml:heading-2>
    <lxml:heading-3 />
  </lxml:caption>
  <lxml:value />
</lxml:field>
<lxml:field name="SALARY">
  <lxml:caption>
    <lxml:label>Salary.....</lxml:label>
    <lxml:description>Employee Salary</lxml:description>
    <lxml:heading-1>Salary</lxml:heading-1>
    <lxml:heading-2 />
    <lxml:heading-3 />
  </lxml:caption>
  <lxml:value />
</lxml:field>
<lxml:field name="MNTHSAL">
  <lxml:caption>
    <lxml:label>Monthly Salary</lxml:label>
    <lxml:description>Monthly Salary</lxml:description>

```

```

        </xml:heading-1>Monthly</xml:heading-1>
        <xml:heading-2>Salary</xml:heading-2>
        <xml:heading-3 />
    </xml:caption>
    <xml:value />
</xml:field>
<xml:field name="STARTDTE">
    <xml:caption>
        <xml:label>Start date.....</xml:label>
        <xml:description>Start Date (DDMMYY)</xml:description>
        <xml:heading-1>Start</xml:heading-1>
        <xml:heading-2>Date</xml:heading-2>
        <xml:heading-3 />
    </xml:caption>
    <xml:value />
</xml:field>
<xml:field name="TERMDATE">
    <xml:caption>
        <xml:label>Term. date.....</xml:label>
        <xml:description>Termination Date (DDMMYY)
</xml:description>
        <xml:heading-1>Term.</xml:heading-1>
        <xml:heading-2>Date</xml:heading-2>
        <xml:heading-3 />
    </xml:caption>
    <xml:value />
</xml:field>
</xml:fields>
<xml:lists>
    <xml:list name="DEPTLIST" row-count="5">
        <xml:list-header>
            <xml:header name="DEPARTMENT">
                <xml:heading-1> Dept</xml:heading-1>
                <xml:heading-2> Code</xml:heading-2>
                <xml:heading-3 />
            </xml:header>
            <xml:header name="DEPTDESC">
                <xml:heading-1>Department</xml:heading-1>
                <xml:heading-2>Description</xml:heading-2>

```

```

        <|xml:heading-3 />
    </|xml:header>
</|xml:list-header>
<|xml:list-entries>
    <|xml:entry>
        <|xml:column name="DEPARTMENT" id="DEPTLIST.0001.DEPTM
        <|xml:column name="DEPTDESC" id="DEPTLIST.0001.DEPTDI
    </|xml:entry>
    <|xml:entry>
        <|xml:column name="DEPARTMENT" id="DEPTLIST.0002.DEPTM
        <|xml:column name="DEPTDESC" id="DEPTLIST.0002.DEPTDI
    </|xml:entry>
    <|xml:entry>
        <|xml:column name="DEPARTMENT" id="DEPTLIST.0003.DEPTM
        <|xml:column name="DEPTDESC" id="DEPTLIST.0003.DEPTDI
    </|xml:entry>
    <|xml:entry>
        <|xml:column name="DEPARTMENT" id="DEPTLIST.0004.DEPTM
        <|xml:column name="DEPTDESC" id="DEPTLIST.0004.DEPTDI
    </|xml:entry>
    <|xml:entry>
        <|xml:column name="DEPARTMENT" id="DEPTLIST.0005.DEPTM
        <|xml:column name="DEPTDESC" id="DEPTLIST.0005.DEPTDI
    </|xml:entry>
</|xml:list-entries>
</|xml:list>
<|xml:list name="SECTLIST" row-count="5">
    <|xml:list-header>
        <|xml:header name="SECTION">
            <|xml:heading-1> Section</|xml:heading-1>
            <|xml:heading-2> Code</|xml:heading-2>
            <|xml:heading-3 />
        </|xml:header>
        <|xml:header name="SECDESC">
            <|xml:heading-1>Section</|xml:heading-1>
            <|xml:heading-2>Description</|xml:heading-2>
            <|xml:heading-3 />
        </|xml:header>
    </|xml:list-header>

```

```

<lxml:list-entries>
  <lxml:entry>
    <lxml:column name="SECTION" id="SECTLIST.0001.SECTION"
    <lxml:column name="SECDESC" id="SECTLIST.0001.SECDESC"
  </lxml:entry>
  <lxml:entry>
    <lxml:column name="SECTION" id="SECTLIST.0002.SECTION"
    <lxml:column name="SECDESC" id="SECTLIST.0002.SECDESC"
  </lxml:entry>
  <lxml:entry>
    <lxml:column name="SECTION" id="SECTLIST.0003.SECTION"
    <lxml:column name="SECDESC" id="SECTLIST.0003.SECDESC"
  </lxml:entry>
  <lxml:entry>
    <lxml:column name="SECTION" id="SECTLIST.0004.SECTION"
    <lxml:column name="SECDESC" id="SECTLIST.0004.SECDESC"
  </lxml:entry>
  <lxml:entry>
    <lxml:column name="SECTION" id="SECTLIST.0005.SECTION"
    <lxml:column name="SECDESC" id="SECTLIST.0005.SECDESC"
  </lxml:entry>
</lxml:list-entries>
</lxml:list>
<lxml:json-list name="LIST01"><![CDATA[{ "list":{
  "LIST01":{"header":[
    {"name":"DEPARTMENT","heading-2":"Dept","heading-3":"Code"},
    {"name":"DEPTDESC","heading-2":"Department","heading-
3":"Description"},
    {"name":"PCK105","heading-1":"Packed","heading-2":"
(10,","heading-3":"5)"},
    {"name":"DAT01","heading-2":"Date","heading-3":"field"},
    {"name":"BOOL1","heading-2":"Boolean","heading-3":"Field"},
    {"name":"FLT01","heading-2":"Float","heading-3":"field"},
    {"name":"FLT04","heading-2":"Float","heading-3":"4"},
    {"name":"INT01","heading-1":"Integer","heading-
2":"field","heading-3":"1"},
    {"name":"INT02","heading-1":"Integer","heading-
2":"field","heading-3":"2"}]},
  "entries":[

```

```
["ADM","Admin > Dept",23456.78900,"2011-08-
29",true,+1.2345670000000000E+004,+1.234567E+004,12345,12345],
["SD","Sales\" & Dist",65432.12340,"2011-08-
29",false,+1.2345670000000000E+004,+1.234567E+004,12345,12345]
]]}
]]>
</xml:json-list>
</xml:lists>
</xml:data>
```

付録C. 廃止されたウェブレット (英語)

When a weblet is enhanced, best efforts are made to ensure that it remains backwards compatible so that existing WAMs continue to look and behave as they always did. Sometimes, changes in the way browsers behave, specific implementation requirements of the new features or other technical restrictions make it impossible to implement new features or fixes and maintain backwards compatibility.

When this occurs, a new weblet is created and the old weblet is left unchanged. The new weblet will usually have the same display name but will have a different XSLT template name (often with something like "_v2" appended to the name).

Weblets that have been deprecated are still shipped so that existing WAMs continue to work as before but they are normally removed from the Weblet Templates repository display. You can make them visible in the list by turning on the *Show deprecated Weblets* option in the XSL tab of the LANSAs Settings dialog.

If the changes to the weblet properties or behavior is significant then the documentation for the deprecated weblet is retained in this section.

Weblets in this category include:

Weblet name	Description
Attachment Panel (std_attachment_panel)	Panel with five areas where content can be dropped, Left, Top, Right, Center, and Bottom. Each of these has attachment layout manager behavior. Contents can be inserted or other weblets dropped into any of the five areas. Dropped weblets are sized according to attachment layout manager rules when they are dropped.
Date (std_date)	A text input box that supports the display, entry, prompting and validation of dates.
DateTime (std_datetime)	A text input box that supports the display, entry, prompting and validation of date and/or time values.
Dynamic HTML menu bar (std_dhtml_menu)	DHTML Multilevel Menu.

Push Button
(std_button) & Push
Button with Images
(std_image_button)

A button with images. Images can be on the left or right, or both, of the caption.

Time (std_time)

A text input box with added features to support the display, entry and validation of times.

Tree view
(std_treeview)

Tree control, Internet Explorer version only.

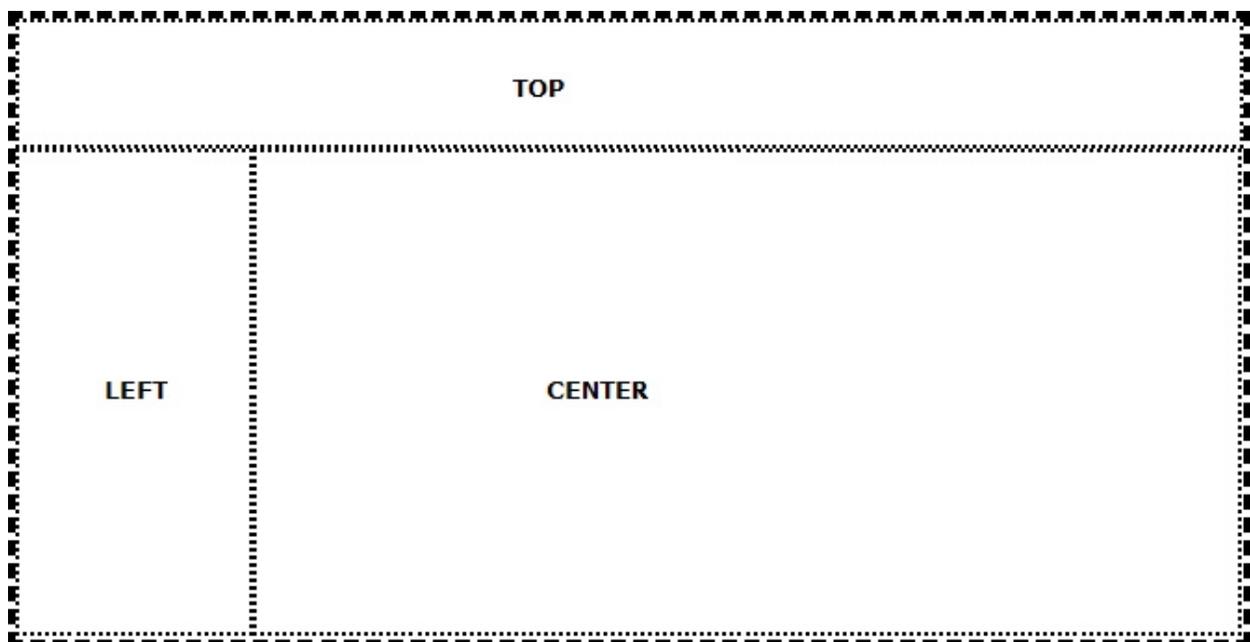
Tree view target
(std_treeview_target)

Panel that is a target of tree control selection actions.

Attachment panel (std_attachment_panel)

The Attachment panel weblet provides a panel with five areas where content can be dropped: left, top, right, center, and bottom. Each of these has attachment layout manager behavior. Contents can be inserted or other weblets dropped into any of the five areas. Dropped weblets are sized according to attachment layout manager rules when they are dropped.

The following is an example of the appearance of a nearly empty attachment panel. In this example, just three of the attachment areas have been used. A thick dashed border has been specified for the attachment panel and thin dotted borders for the panels used in the three areas. The borders have been used for clarity in this example – you do not have to use visible borders and you may not wish to in your applications. Remember you can drag and drop other weblets (such as input boxes, check boxes and push buttons) onto each of the layout areas.



The attachment panel is one of a number of weblets that you can use to aid the creation of a consistent and visually appealing layout for your web pages. You may also wish to review the horizontal and vertical splitters and the panel and navigation panel weblets. This weblet (the attachment panel) is static – the user is not able to resize or otherwise manipulate the size and position of the panels that it contains.

QuickStart- Attachment panel

To use the attachment panel you can follow these steps:

1. Click on the Weblets tab, select Standard Weblets from the drop-down list near the top and locate the Attachment panel weblet.
2. Drag and drop the weblet onto your page. Make sure the weblet is selected and then click on the Details tab. Set any properties required for the attachment panel, such as borders.
3. Now you can drag and drop or otherwise insert content into the required panes or layout areas. You may find it easiest to drag and drop the Panel weblet into each of the five layout areas that you wish to use. You can then more easily size those panels and insert other weblets onto those child panels.

name

The name the weblet is identified with. Normally, you would leave this as the default and let LANSA use its own internal naming convention. However, you may want to use your own name if using JavaScript or XSL that references the weblet.

Default value

An automatically generated, unique identifier.

Valid values

Single-quoted text.

panes

An XML node set specifying a set of panes to show. This is a system generated value set up when you drag the attachment panel onto the design view. You cannot modify the value of this property.

Default value

`document(")/*/lxml:data/lxml:panes[@id='<unique id>']` (this is equivalent to the current pane where the unique id is an automatically generated identifier.)

Valid values

Not Applicable. (This value is system maintained.)

border

The border style for the outer boundary of the weblet. For example 'dashed'.

Default value

Blank (no border is shown).

Valid values

Click the dropdown button next to this property in the property sheet to select one of the pre-defined border styles. Note that 'window-inset' is only supported by Internet Explorer.

border_width

The width of the border for the outer boundary of the weblet. This property is ignored unless a border style is selected for the border property

Default value

Blank. If a border style is selected, this default is equivalent to the 'medium' selection.

Valid values

Click the dropdown button next to this property in the property sheet to select one of the following values:

'medium'

'thick'

'thin'

hide_if

An expression which, if evaluated to be True, will hide the weblet.

Default value

False() (that is, the weblet will always be shown)

Valid values

Any valid XPath expression that returns a Boolean value.

**class_top, class_left, class_center,
class_right, class_bottom**

These properties specify the Cascading Style Sheet (CSS) class names for the five layout areas of the weblet.

Default value

The name of the shipped class for the corresponding layout area of the weblet.

Valid values

Any valid class name from the Cascading Style Sheet, in single quotes. A list of available classes can be selected from by clicking the corresponding dropdown button in the property sheet.

pos_absolute

The absolute position of the weblet on the web page. Note that 'Position Absolutely' must be selected from the weblet's right-click menu for this property to be used. The property will usually be set in pixels by dragging and dropping the weblet.

Default value

Blank (not positioned).

Valid values

Valid 'left' and 'top' coordinates, in valid units of measurement, in single quotes.

Example

In this example, Position Absolutely has been enabled for the weblet and the weblet was positioned as required in the Design view of the LANSAs Editor. This resulted in the value shown for the pos_absolute_design property.

 pos_absolute_design	'position:absolute;left: 324pt; top: 162.72pt;'
---	---

width

The width of the weblet on the web page.

Usually you would set the height and width of the weblet by dragging the grab-handles around the weblet in the Design view of the LANSAs Editor. Doing so updates the value of the width-design and height_design properties. However you can directly edit the property values if required.

Default value

" (this specifies that the attachment panel will use the full width available in the containing element).

Valid values

A width, in a valid unit of measurement, in single quotes.

height

The height of the weblet on the web page.

Usually you would set the height and width of the weblet by dragging the grab-handles around the weblet in the Design view of the LANSAs Editor. Doing so updates the value of the width-design and height_design properties. However you can directly edit the property values if required.

Default value

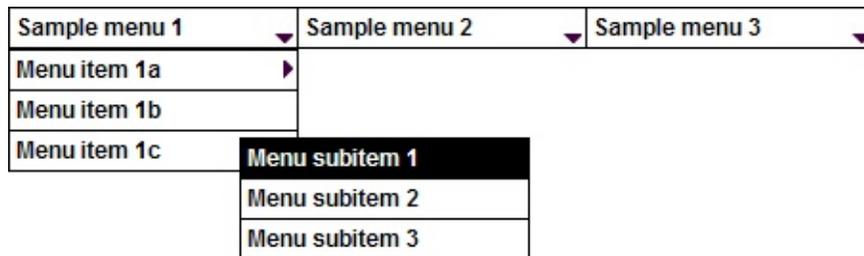
'250pt'

Valid values

A height, in a valid unit of measurement, in single quotes.

Dynamic HTML menu bar (std_dhtml_menu)

The Dynamic HTML menu bar weblet provides the functionality of a menu bar that can invoke other web pages including other web routines. The menu bar can be arranged horizontally or vertically and the top level menu items can cause further menus to pop-up as the mouse moves over them. This is what the menu bar weblet looks like when arranged horizontally – in this example, two levels of popup menu are shown:



The weblet provides just four properties that affect its orientation and size. The menu items themselves are specified using the menu item designer.

QuickStart - Dynamic HTML menu bar

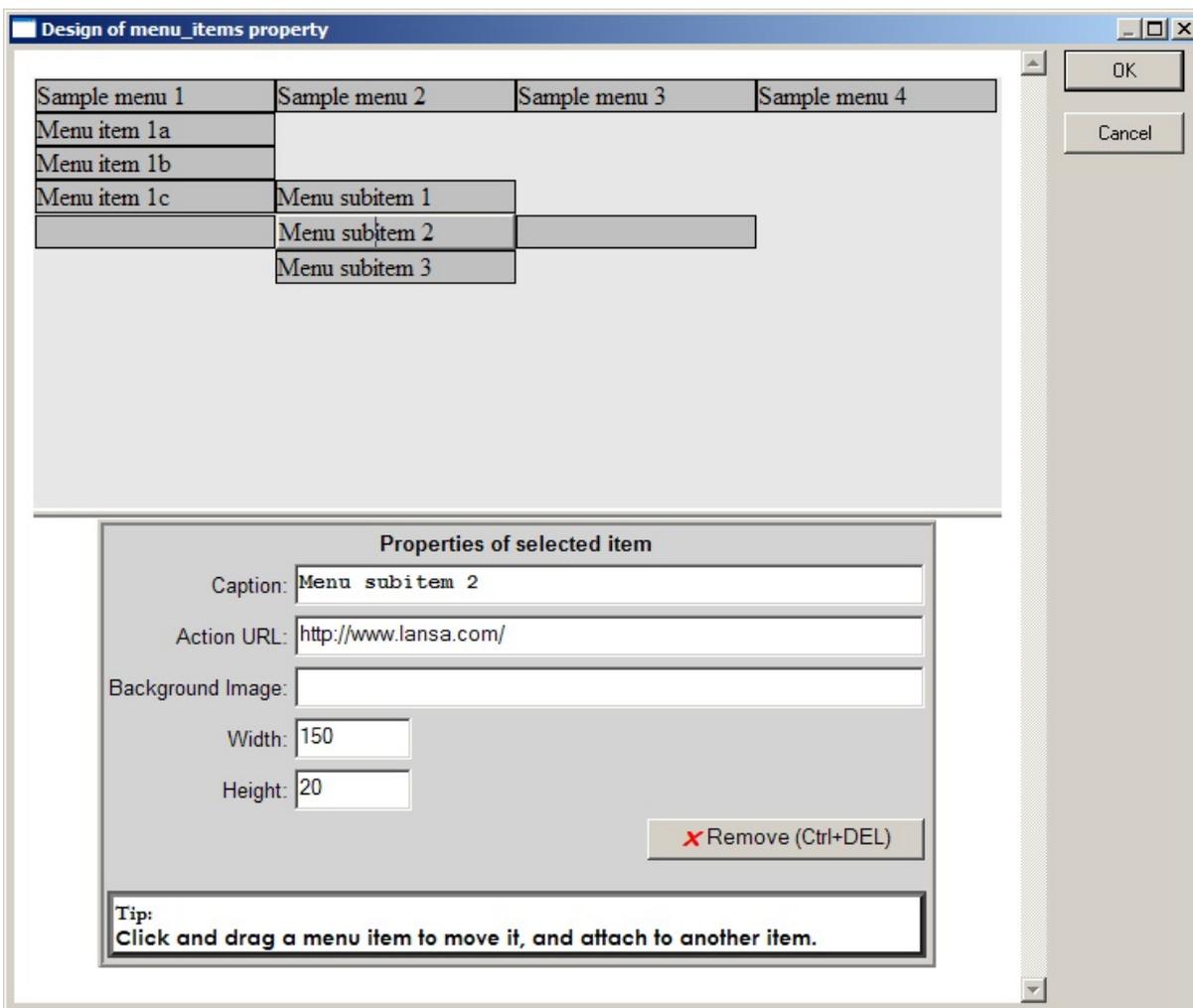
To use the Dynamic HTML menu bar weblet, open your webroutine in the LANSAs Editor and follow these steps:

1. Click on the Weblets tab, select Standard Weblets from the drop-down list near the top and locate the Dynamic HTML menu bar weblet.
2. Drag and drop the weblet onto your page in the Design view. Make sure the weblet on the page is selected and then click on the Details tab.
3. Click the ellipsis button next to the menu_items property to open the menu items designer and define your menu items.

Using the menu item designer

To open the menu item designer, follow these steps:

1. Make sure the Dynamic HTML menu bar weblet on your page is selected and then click on the Details tab.
2. Move the mouse pointer over the menu_items property in the Details tab. An ellipsis should appear next to the property value. Click the ellipsis button to open the menu items designer. A window like the one shown below appears.



The top-half of the window shows a representation of the current state of the menu. Note that this is always shown in horizontal orientation irrespective of the current value of the orientation property. In this part of the window, you can:

- click on items to complete their details in the lower half;
- drag and drop items to rearrange them;

- add new items by clicking on the blank items shown adjacent to the currently selected item;
- remove items by selecting them and clicking the Remove button.

In the bottom-half of the window you can specify the details for the selected item as follows:

Caption

Specifies the text that appears on the face of the menu item. You can also enter the text directly on the face of a menu item by clicking on it in the top-half of the window and typing.

Action URL

Specifies a URL that the menu item will navigate to when clicked. You can specify a complete URL or one that is relative to the current page. The URL can invoke another webroutine by specifying an appropriate URL.

Background Image

Specifies the path and file name of an image to be displayed as background to the menu item.

Width

Specifies the width in pixels of the menu item. See Understanding menu bar and menu item width and height for information on how this value and the weblet width property determine the menu bar and item width.

Height

Specifies the height in pixels of the menu item. See Understanding menu bar and menu item width and height for information on how this value and the weblet height property determine the menu bar and item height.

Understanding menu bar and menu item width and height

The Dynamic HTML Menu bar weblet has width and height properties. In addition, you can specify the width and height of individual menu items in the menu item designer. Following is a summary of how these values work together to determine the width and height of the menu bar, the top-level items and the items in pop-up menus. The effect of these width and height values can also vary according to the chosen value for the orientation property of the weblet.

Width

For menu items in pop-up menus, the width is determined by the width specified in the menu item designer for the first item in that pop-up. All items in the pop-up have the same width. The value specified for the width property of the weblet does not affect the width of these items.

For the top-level menu items (those that are shown statically on the page), the width is determined as follows:

If a value is specified for the weblet width property, then it applies to all top-level menu items (it overrides the width that may have been specified in the menu item designer for individual items).

Otherwise, the width depends on the orientation of the menu bar as follows:

- When in horizontal orientation ('top' is specified for the orientation property) the widths specified for individual top-level items is respected (that is, they can be different).
- When in vertical orientation ('left' or 'right' is specified for the orientation property) the width specified for the first top-level item applies to all the top-level items (that is, they are all the same width).

Height

For menu items in pop-up menus, the height specified for individual items applies. That is the items can have different heights, both within one pop-up menu and across different pop-up menus. The value specified for the height property of the weblet does not affect the height of these items.

For the top-level menu items (those that are shown statically on the page), the menu item height depends on the orientation of the menu bar as follows:

- When in vertical orientation ('left' or 'right' is specified for the orientation property) the heights specified for individual top-level items is respected (that is, they can be different).

- When in horizontal orientation ('top' is specified for the orientation property) the height specified for the first top-level item applies to all the top-level items (that is, they are all the same height).

If a value is specified for the height property of the weblet, it does not alter the apparent height of the menu items (that is the dimensions of the visible boundary of the menu item). In other words, the top-level menu items appear to be the same size irrespective of the value of the height property. Instead, the height property specifies the vertical space reserved for the menu bar – that is, it affects the vertical spacing between the menu bar and following page elements.

By increasing the height property value you can increase the space between the menu bar and following page elements.

By decreasing the height property value you can decrease this space, even to the point that the menu bar can apparently overlap following page elements in some circumstances.

If no value is specified for the height property, the weblet allocates a default amount of space according to the height of the first or all top-level menu items, depending upon the orientation of the menu bar.

Properties - Dynamic HTML menu bar

The Dynamic HTML menu bar weblet's properties are:

[menu_items](#)

[height](#)

[orientation](#)

[width](#)

menu_items

An XML nodeset that specifies the menu items. This is a system generated value set up when you drag the menu onto the design view.

Do not directly edit the value shown. Instead, click the ellipsis button to open the menu items designer. Refer to Using the menu item designer for more information.

Default value

`document(")/*/lxml:data/lxml:menu[@id='<unique id>']`

where the <unique id> is an automatically generated identifier.

Valid values

Not Applicable (this value is system generated and should not be modified).

orientation

The orientation of the menu. This determines the positioning of the top-level menu items relative to each other and the direction or relative location that pop-up menus appear.

Default value

'top'

Valid values

Click the dropdown button next to this property in the property sheet to select one of the following values:

'top' The top-level menu items are arranged horizontally and first-level pop-up menus appear below the corresponding top-level menu item. Suitable for use as a horizontal menu bar across or near the top of the page.

'left' The top-level menu items are arranged vertically and first-level pop-up menus appear to the right of the corresponding top-level menu item. Suitable for use as a vertical menu bar on the left of the page.

'right' The top-level menu items are arranged vertically and first-level pop-up menus appear to the left of the corresponding top-level menu item. Suitable for use as a vertical menu bar on the right of the page.

height

The height of the weblet on the web page. See Understanding menu bar and menu item width and height for information on how this property and the menu item height specified in the menu item designer determine the menu bar and item height.

Default value

Blank (See Understanding menu bar and menu item width and height).

Valid values

A height in pixels.

width

The width of the top-level menu items on the web page. See Understanding menu bar and menu item width and height for information on how this property and the menu item width specified in the menu item designer determine the menu bar and item width.

Default value

Blank (See Understanding menu bar and menu item width and height).

Valid values

A width in pixels.

Push Button (std_button) & Push Button with Images (std_image_button)

The Push Button webllets provide windows-like push buttons for your web page. They look like this:



QuickStart- Push Button & Push Button with Images

To add a push button to your web page:

1. Click on the Weblets tab, select Standard Weblets from the drop-down list near the top and locate either of the Push Button weblets.
2. Drag and drop the required weblet onto the web page. Click on the Details tab.
3. Set the caption to specify the text to be displayed on the button. In the case of the Push Button with Images, set the appropriate image properties. A left-hand side image and a right-hand side image can be set.
4. Set the `on_click_wrname` properties to the name of the webroutine to be invoked when the button is clicked. If the webroutine is in a different WAM to the current webroutine then you will also need to set the `on_click_wamname` property.

Properties - Push Button & Push Button with Images

All these properties are common to both button weblets except for those indicated as *std_image_button only*.

caption	left_image_height	right_absolute_image_path
class	left_relative_image_path	right_image_class
currentrowhfield	mouseover_class	right_image_height
currentrownumval	name	right_relative_image_path
default_button	on_click_wamname	show_in_new_window
disabled	on_click_wrname	tab_index
formname	pos_absolute_design	target_window_name
height_design	presubmit_js	text_class
hide_if	protocol	title
left_absolute_image_path	reentryfield	width_design
left_image_class	reentryvalue	

name

The name of the weblet. Normally, you would leave this as the default and let LANSA use its own internal naming convention. However, you may want to use your own name if using JavaScript or XSL that references the weblet.

Default value

`concat('o', position(), '_LANSA_n')` – this is the internal name given to the weblet by LANSA.

Valid values

A name enclosed in single quotes.

caption

The caption for the weblet.

Default value

'Caption'

Valid values

Single-quoted text or the name of a multilingual text variable (the corresponding ellipses button in the property sheet can be clicked to choose one from a list).

currentrowhfield

The field name to be used to post to the target webroutine the value that is specified in the currentrownumval property. The field name should be in single quotes.

See the description of the currentrownumval property for further information.

Default value

'STDROWNUM'

Valid values

Single-quoted text.

Example

This example specifies the field name DEPTLINK as the field name to be used to post the value to the target webroutine. The target webroutine would need to have field DEPTLINK in its WEB_MAP for *BOTH or for *INPUT in order to receive the value:

 currentrowhfield	'DEPTLINK'
--	------------

currentrownumval

The value to post to the target webroutine in the field specified in the `currentrowhfield` property. If that field is alphanumeric, the value must be specified in single quotes. If it is numeric, the value can be specified with or without quotes.

This property is used in conjunction with the `currentrowhfield` property to describe how to post values to a target webroutine. These two pieces of information are required to accomplish this:

1. `currentrowhfield`: the field name that the target webroutine uses to refer to the information
2. `currentrownumval`: a literal value or a field name in this (the source) webroutine that contains the necessary information

Note: Despite the name of the property being *currentrownumval*, the field name specified in `currentrownumval` is not required to be a numeric field.

Default value

`position()`

Valid values

Single-quoted text or the name of a field, system variable or multilingual text variable.

left_relative_image_path

std_image_button only.

The path and name, relative to the images directory, of the image to be displayed on the left of the weblet. If specified, the `left_absolute_image_path` property should be left blank.

Default value

'ball_red.gif'

Valid values

The path and name of an image, relative to the images directory, enclosed in single quotes. An image can be chosen from a prompter by clicking the corresponding ellipses button in the property sheet.

left_absolute_image_path

std_image_button only.

The path and name of the image to be displayed on the left of the weblet. If specified, the `left_relative_image_path` property should be left blank.

Default value

Blank – the default is to use the image specified in the `left_relative_image_path` property.

Valid values

The path and name of an image enclosed in single quotes.

left_image_height

std_image_button only.

The height of the image on the left of the weblet.

Default value

'12pt'

Valid values

A height, in a valid unit of measurement, enclosed in single quotes.

right_relative_image_path

std_image_button only.

The path and name, relative to the images directory, of the image to be displayed on the right of the weblet. If specified, the `right_absolute_image_path` property should be left blank.

Default value

Blank – by default, buttons do not display an image on the right.

Valid values

The path and name of an image, relative to the images directory, enclosed in single quotes. An image can be chosen from a prompter by clicking the corresponding ellipses button in the property sheet.

right_absolute_image_path

std_image_button only.

The path and name of the image to be displayed on the right of the weblet. If specified, the `right_relative_image_path` property should be left blank.

Default value

Blank – the default is to use the image specified in the `right_relative_image_path` property, if specified.

Valid values

The path and name of an image, enclosed in single quotes.

right_image_height

std_image_button only.

The height of the image on the right of the weblet.

Default value

'12pt'

Valid values

A height, in a valid unit of measurement, enclosed in single quotes.

reentryfield

The field name to be used to post to the WAM the value that is specified in the reentryvalue property. The field name should be in single quotes.

Default value

'STDREENTRY'

Valid values

Any repository- or WAM-defined field name. A list of known field names is available by clicking the corresponding dropdown button in the property sheet.

reentryvalue

The value to post into the field specified in the reentryfield property. If that field is alphanumeric, the value must be specified in single quotes. If it is numeric, the value can be specified with or without quotes.

Default value

'M'

Valid values

Any appropriate literal.

hide_if

An expression which, if evaluated to be True, will hide the weblet.

Default value

False() (that is, the weblet will always be shown)

Valid values

Any valid XPath expression that returns a Boolean value.

Example

This example will hide the weblet if field #STD_FLAG is equal to 'X'. The expression should be entered, and is shown when the property has focus, as follows:

 hide_if	#STD_FLAG = 'X'
---	-----------------

When the property loses focus, the expression is shown as follows:

 hide_if	key('field-value', 'STD_FLAG') = 'X'
---	--------------------------------------

formname

The name of the HTML form that is posted to the server.

Default value

'LANSA'

Valid values

A name for the form, in single quotes. A list of known form names is available by clicking the corresponding dropdown button in the property sheet.

pos_absolute_design

The absolute position of the weblet on the web page. Note that 'Position Absolutely' must be selected from the weblet's right-click menu for this property to be used. The property will usually be set in pixels by dragging and dropping the weblet.

Default value

Blank (not positioned).

Valid values

Valid 'left' and 'top' coordinates, in valid units of measurement, in single quotes.

width_design

The width of the weblet on the web page.

Default value

Blank (weblet uses its default width).

Valid values

A width, in a valid unit of measurement, in single quotes.

height_design

The height of the weblet on the web page.

Default value

Blank (weblet uses its default height).

Valid values

A height, in a valid unit of measurement, in single quotes.

on_click_wamname

The name of the WAM to be invoked when the weblet is clicked.

Default value

\$lweb_WAMName (this is equivalent to the current WAM).

Valid values

The name of a WAM in single quotes. A list of known WAMs can be displayed by clicking the corresponding dropdown button on the property sheet.

on_click_wrname

The name of the Webroutine to be invoked when the weblet is clicked.

Default value

Blank – a Webroutine name must be specified.

Valid values

The name of a Webroutine in single quotes. The Webroutine must exist in the WAM specified in the on_click_wamname property. A list of known Webroutines can be displayed by clicking the corresponding dropdown button on the property sheet.

protocol

The protocol (for example, http:// or https://) that should be used for navigation to the Webroutine specified in the on_click_wname property.

Default value

Blank. This is equivalent to the current protocol being used.

Valid values

A valid protocol, in single quotes. This is usually 'http:' or 'https:'.

show_in_new_window

A Boolean property, the result of which determines whether response HTML for the weblet should be shown in a new browser window.

Default value

false() – response HTML is shown in the current browser window.

Valid values

true(), false() or a valid expression.

target_window_name

The name of the window, or frame, in which response HTML will be shown.

Default value

Blank – response HTML will be shown in the current window.

Valid values

The name of a window or frame, in single quotes. A list of known windows and frames can be displayed by clicking on the corresponding dropdown button in the property sheet, or a unique name can be entered.

'_blank' will launch in a new window

'_media' will launch a media panel in the current window

'_search' will launch a search panel in the current window

'_parent' will launch in the parent window (usually the current window)

'_top' will launch in the top window (usually the current window)

Note that _search and _media are supported by Internet Explorer 6 only.

disabled

A Boolean property, the result of which determines whether the weblet appears enabled or disabled.

Default value

Blank – equivalent to False (that is, the weblet will always be enabled).

Valid values

true(), false() or a valid expression.

text_class

std_image_button only.

The Cascading Style Sheet (CSS) class name of the text of the weblet.

Default value

The name of the shipped text class for the weblet.

Valid values

Any valid class name from the Cascading Style Sheet, in single quotes. A list of available classes can be selected from by clicking the corresponding dropdown button in the property sheet.

title

Text to be displayed as a Tool Tip for the weblet when the mouse is hovered over it.

Default value

Blank – no Tool Tip text will be displayed.

Valid values

Single-quoted text or the name of a multilingual text variable (the corresponding ellipses button in the property sheet can be clicked to choose one from a list).

class

The Cascading Style Sheet class name of the weblet.

Default value

'std_button' or 'std_image_button' - The name of the default shipped class for the weblet.

Valid values

Any valid class name from the Cascading Style Sheet, in single quotes. A list of available classes can be selected from by clicking the corresponding dropdown button in the property sheet.

mouseover_class

The Cascading Style Sheet class name of the weblet when the mouse is moved over it.

Default value

'std_button_mouseover' - The name of the default shipped mouseover class for the weblet.

Valid values

Any valid class name from the Cascading Style Sheet, in single quotes. A list of available classes can be selected from by clicking the corresponding dropdown button in the property sheet.

left_image_class

std_image_button only.

The Cascading Style Sheet class name of the left image.

Default value

Blank – the image is displayed without the application of a style.

Valid values

Any valid class name from the Cascading Style Sheet, in single quotes. A list of available classes can be selected from by clicking the corresponding dropdown button in the property sheet.

right_image_class

std_image_button only.

The Cascading Style Sheet class name of the right image.

Default value

Blank – the image is displayed without the application of a style.

Valid values

Any valid class name from the Cascading Style Sheet, in single quotes. A list of available classes can be selected from by clicking the corresponding dropdown button in the property sheet.

presubmit_js

JavaScript code to be run prior to the submission of the form.

Default value

Blank. No JavaScript is run.

Valid values

Any valid JavaScript function, or JavaScript code followed by a semicolon (;).

If you want to execute the presubmit JavaScript only, without running the JavaScript that submits the request (thus canceling the onclick event), append **return false;** to your presubmit JavaScript.

Example

The following example shows a message box:

```
presubmit_js 'alert("Hello world!");'
```

The following example shows a message box and cancels the submit JavaScript:

```
presubmit_js 'alert("Hello world!"); return false;'
```

tab_index

Determines the tab order of the weblet on the form. The tab_index property value determines the tab order as follows:

1. Objects with a positive tab_index are selected in increasing tab_index order (and in source order to resolve duplicates).
2. Objects with a tab_index of zero or blank (the default) are selected in source order.
3. Objects with a negative tab_index are omitted from the tabbing order. Note that this behavior is not defined in the HTML specifications and is only supported by Internet Explorer and Firefox.

Default value

Blank. The weblet is selected in source order.

Valid values

Blank or a valid numeric value.

default_button

A Boolean property, the result of which determines whether the button is the default button for the form. Only one button on the form can be the default button – setting to True will set all other buttons to False.

Default value

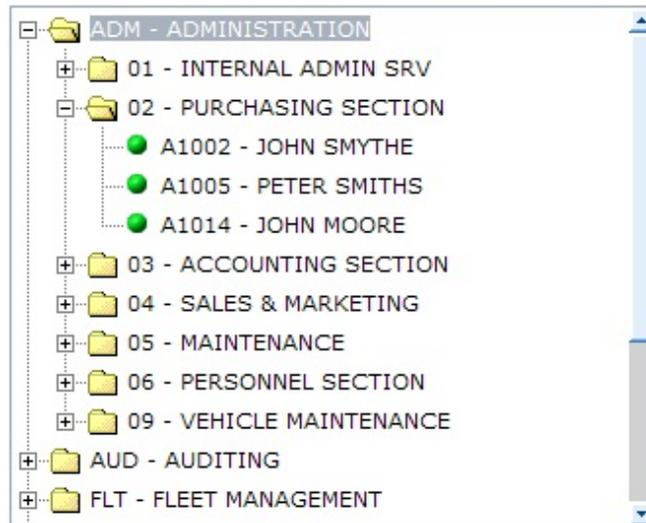
Blank - Equivalent to False.

Valid values

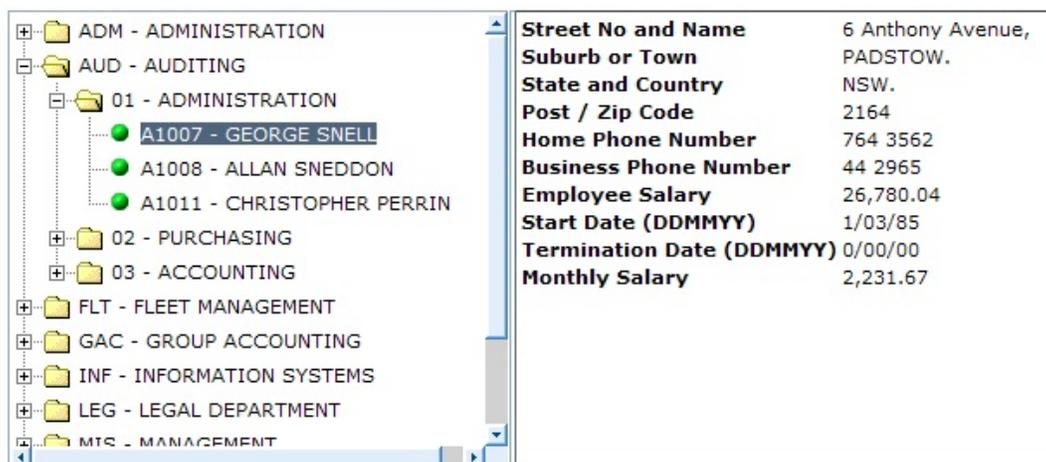
true(), false() or a valid expression.

Tree View (std_treeview)

The Tree View weblets provides a windows-like tree view on your web page. It can be used to produce something similar to this example:



It can also be used in conjunction with the Tree View Target (std_treeview_target) weblet, which responds to selection events from the tree view and can be used to display detail information. For example, the Tree View Target weblet can be seen displaying employee information to the right of the Tree View below:



QuickStart - Tree View

Refer to [An In-Depth Look at the Tree View Weblet](#).

Properties - Tree View

The Tree View weblet's properties are:

bg_color	list_is_selected_field	listname
default_style	list_onselect_wname_field	listname_of_parents_of_se
folder_closed_image	list_open_image_field	name
folder_open_image	list_parent_id_field	onexpand_wamname
formname	list_selected_style_field	pos_absolute_design
item_image	list_style_field	selected_style
list_caption_field	list_subitem_group_field	width_design
list_haschildren_field	list_tag_field	xmlid
list_image_field	list_type_field	xmltyped
list_is_expanded_field		

name

The name of the weblet. Normally, you would leave this as the default and let LANSAs use its own internal naming convention. When using this weblet in conjunction with the Tree View Target (this would normally be the case), it is recommended that a name be entered, as the Tree View Target will be required to reference it. Using this name will be clearer than using the LANSAs-generated name.

Default value

`concat('oTree', ancestor-or-self::lxml:list/@name,position())` – this is the internal name given to the tree view by LANSAs.

Valid values

A name in single quotes.

formname

The name of the HTML form that is posted to the server.

Default value

'LANSA'

Valid values

A name for the form, in single quotes. A list of known form names is available by clicking the corresponding dropdown button in the property sheet.

xmlid

The identifier of the XML Data Island containing tree items. Leave blank to use the default id.

Default value

`concat('xmltree_', $name)`

Valid values

A name for the data island, in single quotes, or a valid JavaScript function that will produce a valid name (the default name uses the concat function).

xmltyped

The identifier of the XML Data Island containing tree item types. Leave blank to use the default id.

Default value

`concat('xmltreetype_', $name)`

Valid values

A name for the data island, in single quotes, or a valid JavaScript function that will produce a valid name (the default name uses the concat function).

folder_closed_image

The path and file name, relative to the images directory, of an image that represents closed tree nodes.

Default value

'folder.gif'

Valid values

The path and name of an image, relative to the images directory, enclosed in single quotes. An image can be chosen from a prompter by clicking the corresponding ellipses button in the property sheet.

folder_open_image

The path and file name, relative to the images directory, of an image that represents open tree nodes.

Default value

'folderopen.gif'

Valid values

The path and name of an image, relative to the images directory, enclosed in single quotes. An image can be chosen from a prompter by clicking the corresponding ellipses button in the property sheet.

item_image

The path and file name, relative to the images directory, of an image to represent a leaf node of the tree.

Default value

'ball_grn.gif'

Valid values

The path and name of an image, relative to the images directory, enclosed in single quotes. An image can be chosen from a prompter by clicking the corresponding ellipses button in the property sheet.

listname

The name of the working list that contains the items used to populate the weblet.

Default value

Blank. A valid list name must be entered.

Valid values

The name of a valid working list, in single quotes. A list of valid list names can be chosen from by clicking the corresponding dropdown button in the property sheet.

list_caption_field

The name of the field in the listname working list that contains tree item captions.

Default value

Blank. A valid field name from the listname working list must be specified.

Valid values

The name of a valid field, in single quotes. A list of fields in the listname working list can be chosen from by clicking the corresponding dropdown button in the property sheet.

list_type_field

The name of the field in the listname working list that contains tree item types. Leave blank if not using types.

Default value

Blank. A valid field name from the listname working list should be specified if using types.

Valid values

The name of a valid field, in single quotes. A list of fields in the listname working list can be chosen from by clicking the corresponding dropdown button in the property sheet.

list_image_field

The name of the field in the listname working list that contains a tree item's image path and file name, relative to the images directory. Leave blank if using default images.

Default value

Blank. Default images are used.

Valid values

The name of a valid field, in single quotes. A list of fields in the listname working list can be chosen from by clicking the corresponding dropdown button in the property sheet.

list_open_image_field

The name of the field in the listname working list that contains a tree item's image path and file name, relative to the images directory, that represents an expanded node. Leave blank if using default images.

Default value

Blank. Default images are used.

Valid values

The name of a valid field, in single quotes. A list of fields in the listname working list can be chosen from by clicking the corresponding dropdown button in the property sheet.

list_tag_field

The name of the field in the listname working list that contains item tags. This is the non-visible, unique identifier of the tree item that can be used to identify it when selected or expanded.

Default value

\$list_caption_field. The field used to store the tag information is the same field used for the caption.

Valid values

The name of a valid field, in single quotes. A list of fields in the listname working list can be chosen from by clicking the corresponding dropdown button in the property sheet.

list_onselect_wrname_field

The name of the field in the listname working list that contains the name of the Webroutine in the current WAM that is to be invoked when a tree item is selected.

Default value

Blank. A Webroutine will not be invoked when a tree item is selected.

Valid values

The name of a valid field, in single quotes. A list of fields in the listname working list can be chosen from by clicking the corresponding dropdown button in the property sheet.

list_haschildren_field

The name of the field in the listname working list that determines whether a tree item has child items.

Default value

'STD_CODE'

Valid values

The name of a valid field, in single quotes, that will contain a:

'Y' (the tree item has child items) or an

'N' (the tree item does not have child items).

list_subitem_group_field

The name of the field in the listname working list that will contain the depth (or level) of a selected item from the root. For example, a direct child of the root will have a depth of 2.

Default value

'STD_LEVEL'

Valid values

The name of a valid numeric field, in single quotes.

list_is_selected_field

The name of the field in the listname working list, the value of which will determine if a tree item should be selected when displayed.

Default value

Blank. Tree items cannot be pre-selected.

Valid values

The name of the field in the working list that will contain a value of 'True' if an item in the tree should be selected. If set to 'Freeze', the item will be selected but the associated Tree View Target weblet (if applicable) will not be reloaded.

list_is_expanded_field

The name of the field in the working list that will control a tree item's expanded state.

Default value

Blank. A tree item's expanded state cannot be controlled.

Valid values

The name of the field in the working list that will contain a value of 'True' if an item should be expanded, and 'False' if it should not.

list_parent_id_field

The name of the field in the working list that will contain the identifier of the parent of the tree item.

Default value

Blank. A tree item's parent can only be identified by using the listname_of_parents_of_selected property.

Valid values

The name of the field in the working list that will contain the identifier of the parent of the tree item.

list_style_field

The name of the field in the working list that controls a tree item's style. This allows the style of the child items to vary from the parent.

Default value

Blank. The tree item adopts the default style.

Valid values

The name of the field in the working list that will contain the Cascading Style Sheet style of the tree item.

list_selected_style_field

The name of the field in the working list that controls a selected tree item's style.

Default value

Blank. The selected tree item adopts the default style.

Valid values

The name of the field in the working list that will contain the Cascading Style Sheet style of the tree item when it is selected.

onexpand_wamname

The name of the WAM whose Webroutine will be invoked when an item in the tree is expanded.

Default value

Blanks. The current WAM will be invoked.

Valid values

The name of a WAM, in single quotes. A selection can be made from a list of known WAMs by clicking on the corresponding dropdown button in the property sheet.

onexpand_wrname

The name of the Webroutine that will be invoked when an item in the tree is expanded.

Default value

Blank. The current Webroutine is the default.

Valid values

The name of a valid Webroutine, in single quotes. A selection can be made from a list of valid Webroutines by clicking on the corresponding dropdown button in the property sheet.

listname_of_parents_of_selected

The name of the working list returned to the WAM that will contain a list of parent identifiers for the currently selected or expanded tree item.

Default value

Blank. A list of parent identifiers is not passed to the WAM.

Valid values

The name of a valid working list, in single quotes. A selection can be made from a list of valid working lists by clicking on the corresponding dropdown button in the property sheet.

pos_absolute_design

The absolute position of the weblet on the web page. Note that 'Position Absolutely' must be selected from the weblet's right-click menu for this property to be used. The property will usually be set in pixels by dragging and dropping the weblet.

Default value

Blank (not positioned).

Valid values

Valid 'left' and 'top' coordinates, in valid units of measurement, in single quotes.

width_design

The width of the weblet on the web page.

Default value

Blank (weblet uses its default width).

Valid values

A width, in a valid unit of measurement, in single quotes.

height_design

The height of the weblet on the web page.

Default value

Blank (weblet uses its default height).

Valid values

A height, in a valid unit of measurement, in single quotes.

class

The Cascading Style Sheet class name of the weblet.

Default value

'std_treeview' - The name of the shipped class for the weblet.

Valid values

Any valid class name from the Cascading Style Sheet, in single quotes. A list of available classes can be selected from by clicking the corresponding dropdown button in the property sheet.

bg_color

The background color of the weblet.

Default value

Blank. The background color will be the same as the web page.

Valid values

A valid color, either in #RRGGBB format or a valid color name, in single quotes. A color can be chosen from a palette by clicking on the corresponding ellipses button in the property sheet.

Example

Both of the following examples will set the background color to red:

 bg_color	#FF0000'
--	----------

or

 bg_color	'red'
--	-------

default_style

The Cascading Style Sheet inline style of the tree control items.

Default value

Blank. The default style of the Cascading Style Sheet is used.

Valid values

A valid Cascading Style Sheet style name, in single quotes.

selected_style

The Cascading Style Sheet inline style of the selected tree control items.

Default value

Blank. The default style of the Cascading Style Sheet is used.

Valid values

A valid Cascading Style Sheet style name, in single quotes.

An In-Depth Look at the Tree View Weblet

The tree view weblet provides a windows-like tree view control for use by your WAM application. Following is a description of how it should be used. This description is rather long, and has been divided into these sections:

[About the Tree View Examples](#)

[What You Need to Know](#)

[When a Webroutine is Invoked](#)

[Accessing Key Information](#)

[The Tree View Target Weblet](#)

[Using a Navigation Panel](#)

[A Closer Look at WAMEX51](#)

[Invoking the WAM](#)

[Building the Tree View](#)

[The Role of the Tree View Target Weblet](#)

[Rebuilding the Tree View](#)

[Processing Expanding Tree Items](#)

[Accessing Ancestor Information](#)

As with all weblet controls that display a list of information, the Tree View Weblet is primarily driven by a working list in your WAM. Each entry in the working lists represents a tree item. Fields in the working list control the tree view item's appearance and behavior. These include the hidden key information of the tree item, its caption, what to do when the item receives focus and what to do when the tree item is expanded.

Depending on how you wish to utilize the tree view, different techniques are used. These are documented in What You Need to Know.

About the Tree View Examples

The Tree View examples supplied are WAMEX50 and WAMEX51.

They are both simple examples of a tree that contains Department branches which, when expanded, show Section branches belonging to the Department, which, in turn, show Employees belonging to the Section.

Both examples use a Vertical Splitter weblet. This allows the Tree View weblet to be displayed on the left of the web page and details for a selected tree item to be displayed, in a Tree View Target weblet, on the right of the web page.

WAM01 is an enquiry-only example, whereas WAMEX51 allows the update of the selected department's description. Due to the updating of the tree, WAMEX51 uses additional Weblets and slightly different coding techniques to those of WAMEX50.

Both examples are intended to give you a good introduction to the Tree View weblet and how it interacts with other, related weblets, as well as the underlying WAM code.

What You Need to Know

The Tree Working List

In all cases, you will need a working list. Remember that each entry of the working list represents an item in the tree view. The working list should contain fields that will populate the list_* properties of the weblet. Refer to Properties - Tree View for a list of these. As a minimum, however, the working list should contain the following:

- A field to hold a caption for the tree item It can be any free-format text string that describes the tree item In the example WAMs, this is the Department description, the Section description, or the name of an Employee.
- A field to hold the key, or tag, of the tree item In the example WAMs, this field holds either the Department Code, Section Code, or Employee Number Composite keys are derived by querying another working list that holds ancestor items of a given item This is discussed in the Accessing Key Information section.
- A field to hold the name of the Webroutine to be invoked when a tree item is selected When a tree item is selected, you will usually want to perform some action, such as display the details associated with the selected item In the example WAMs, a Webroutine is invoked to display details of Departments, Sections and Employees Refer to the When a Webroutine is Invoked section for more information.
- A Sub-Item Group field This numeric field holds the nesting level of the tree item from its root item In the example WAMs, Departments have a level of 1, sections 2 and employees 3.
- A HasChildren field This is a single-byte field that should contain a Y or N. This denotes whether or not the tree item is a branch that can be expanded in order to display dependant information In the example WAMs, each Department has dependant Sections, so this field is set to Y Likewise, Sections have dependant Employees, so this field is also set to Y Employees,

however, do not have any dependant information, so this field is set to N.

The Parent Items Working List

In most, if not all cases, your tree view will have multiple levels and, usually, will contain data that follows a parent/child pattern. In our examples, this is true of the Department/Sections/Employees data.

The Parent Items working list is passed into the WAM, providing a list of ancestors of an item that has been selected or is expanding. As such, it contains just one field, which must match the field used in the Tree View working list to hold a tree item's key information. More information about using this working list can be found later in this document.

When a Webroutine is Invoked

When a Tree Item is Selected

The `list_onselect_wname_field` property of the tree view points to a field that contains the name of the Webroutine that is to be invoked when the corresponding item in the tree is selected. In the example WAMs, this is set to `ShowDepartmentDetail` for each Department entry.

When this Webroutine is invoked, certain information is passed into it:

The key information of the selected tree item. This is the field that is specified in the `list_tag_field` property. This can be used to access additional or related information that is relevant to the tree item. A `Web_map` to receive this field is required at all times.

The working list that contains the identifiers of the selected tree item's parents or ancestors. This is the list specified in the `listname_of_parents_of_selected` property. See the Accessing Key Information section for more information.

When a Tree Item is Expanded

The `onexpand_wname` property of the Tree View can be used to build the contents of a branch when it is expanded. In the example WAMs, when a Department or Section branch is expanded, the `TreeExpanding` Webroutine is invoked.

As well as the key information of the expanding tree item, the level of the tree item is passed into this Webroutine, along with the tree list itself and the list of parents of the expanding tree item. Based on the level of the expanding tree item, the appropriate method is invoked to add child items to the tree for the selected parent item.

So, a level of 1 indicates that a Department tree item is being expanded, and so

Sections belonging to that Department are added to the tree. A level of 2 indicates that a Section tree item is being expanded, and so Employees are added.

Accessing Key Information

Because a tree item can have parent tree items, a working list to hold this information is required. It should be defined as containing just the field that is specified in the `list_tag_field` property of the tree.

It must be specified as a `Web_map` for the Webroutine that is used to display the main tree view working list. In WAMEX50, the `ShowPage` Webroutine has a `Web_map` specified for the list.

When passed into the WAM, it will have an entry for each of the selected item's parent items. Each entry contains the key information of the parent item. For tree items that do not have any parents, the list will be empty.

A `Web_map` to receive this list is only required if you wish to access information in it. This would typically be required for the selection Webroutine and the Tree Expanding Webroutine.

In the example WAMs, when a Section is selected, this list is used to ascertain the key value of the Department to which the Section belongs.

The Tree View Target Weblet

Usually, you will want to display additional information relating to a tree item that has been selected.

In the example WAMS, when a Department is selected, the `ShowDepartmentDetail` Webroutine is invoked to show additional information for the Department.

There must be an area on the web page for this information to be shown. This is what the Tree View Target weblet is used for. When a tree item is selected, the Tree View Target associated with the tree (via the Tree View Target's `treeview_name` property) becomes active, or gets focus. The output from the Webroutine invoked when the tree item is selected is thus directed to the Tree View Target.

For more information on the Tree View Target (`std_treeview_target`)weblet, refer to its documentation.

Using a Navigation Panel

In WAMEX50, which is a simple enquiry, the Tree View and Tree View Target weblets are displayed by the `ShowPage` Webroutine. This works fine for enquiry

purposes.

It becomes slightly more complex, however, if an update function for the details that are displayed in the Tree View Target weblet is introduced. For example, if the user changes the Department description and presses an update push button, you would expect the Tree View to be updated with the new description.

In order to get this to happen, the Tree View must be displayed in a different way to that shown in WAMEX50. Here, the Tree View is directly placed in the left-hand portion of the Vertical Splitter.

In WAMEX51, the left-hand portion of the Vertical Splitter contains a navigation panel. This panel is set to navigate to the DepartmentTree Webroutine. This is the Webroutine that contains the Tree View weblet. Separating it out like this means the Tree View can be easily refreshed on the web page, as it is, in effect, in its own sub-page of the main web page.

A Closer Look at WAMEX51

To help reinforce the techniques described, this section contains sections of the WAM code, along with property settings of the Tree View and associated weblets.

The List Definitions

The RDML code below shows the field and list definitions that are used by the Tree View weblet.

```
Define Field(#TreeID) Type(*CHAR) Length(256)
Define Field(#TreeCapt) Type(*CHAR) Length(256)
Define Field(#TreeLvl) RefId(#STD_LEVEL)
Define Field(#HasKids) Type(*CHAR) Length(1)
Define Field(#DetailWR) Type(*CHAR) Length(256)
Define Field(#Selected) Type(*CHAR) Length(10)

Def_List Name(#tvDepts) Fields(#TreeID #TreeCapt #TreeLvl #HasKids #DetailWR #Selected) Type(*Working)
  Entrys(9999)

Def_List Name(#Ancestors) Fields(#TreeID) Type(*Working)
```

The tvDepts working list is the list used to populate the Tree View. The function of its fields is as follows:

- TreeID, used to hold the key information for the tree item.
- TreeCapt, used to hold the caption for the tree item.
- TreeLvl, used to return the selected tree item's level to the WAM.
- HasKids, used to denote whether the tree item has children.
- DetailsWR, used to hold the name of the Webroutine to be invoked when the

tree item is selected.

- Selected, used to control whether a tree item should be selected.

The Ancestors list is used to receive parent tree item key information from the Tree View weblet for the selected tree item.

Right-click on the DepartmentTree Webroutine and select the LANSA Editor option. Once the LANSA Editor has opened, click on the tree itself and then select the Details tab. The properties of the tree view will be shown, as follows:

With Parameters	
name	'DeptTree'
formname	'LANSA'
xmlid	<i>concat('xmlTree_', \$name)</i>
xmltypeid	<i>concat('xmlTreeType_', \$name)</i>
folder_closed_image	'folder.gif'
folder_open_image	'folderopen.gif'
item_image	'person.jpg'
listname	'TVDEPTS'
list_caption_field	'TREECAPT'
list_type_field	
list_image_field	
list_open_image_field	
list_tag_field	'TREEID'
list_onselect_wname_field	'DETAILWR'
list_haschildren_field	'HASKIDS'
list_subitem_group_field	'TREELVL'
list_is_selected_field	'SELECTED'
list_is_expanded_field	
list_parent_id_field	
list_style_field	
list_selected_style_field	
onexpand_wamname	<i>/bxml:data/bxml:context/</i>
onexpand_wname	'TreeExpanding'
listname_of_parents_of_selected	'ANCESTORS'

Note the listname property contains the name of the tree view working list, tvDepts. Note also how the list_*, onexpand_wname and listname_of_parents_of_selected properties relate back to fields in tvDepts.

Invoking the WAM

The entry point of the WAM (that which should be used to execute the WAM via a browser URL or from another WAM) is the ViewDepartments Webroutine. It is designed to be the only entry point, to be executed only once. Any initialization logic for the WAM could be placed here. See below:

```

Webroutine Name(ViewDepartments) Onentry(*SESSIONSTATUS_NONE)

  * set the session status to active - this will ensure that session data is written out at the end of the routine.
  #com_owner.SessionStatus := Active

  * show the web page.
  Transfer Toroutine(ShowPage)

Endroutine

```

In this example, the only thing it has to do is set the Session Status to Active. This controls the writing out and reading in of any persistent session data when executing Webroutines in the WAM.

The ShowPage Webroutine is then executed. Again, this is designed to be the only place in the WAM used to display the web page. In this example, no Web_maps specify what is to be displayed – all such information is handled by other Webroutines, as you will see. Non-field and list data has been specified via the LANSAs Editor, and so the ShowPage Webroutine shows as being 'empty'.

Building the Tree View

1. Right-click on the ShowPage Webroutine and select the LANSAs Editor option.
2. Click on the left-hand side of the Vertical Splitter and select the Details tab to display the Navigation Panel's properties, as shown below:

With Parameters	
name	'TreeView'
border	
border_width	
hide_if	<i>false()</i>
pos_absolute	<i>'left:0pt;top:0pt;'</i>
width	<i>'100%'</i>
height	<i>'100%'</i>
size_panel_to_content	<i>false()</i>
size_panel_to_content_axis	<i>'both'</i>
scrolling	
class	<i>'std_nav_panel'</i>
transparent	<i>true()</i>
nav_url	
formname	
nav_wamname	<i>/bxml:data/bxml:context/ba</i>
nav_wrname	'DepartmentTree'
protocol	

Take note of the nav_wrname property. This is the Webroutine that is invoked whenever the Navigation Panel is displayed. In this example, it

points to the DepartmentTree Webroutine, which is used to build the tree view.

3. Close the XML editor and look at the DepartmentTree Webroutine in the WAM source, as shown below:

```
Webroutine Name(DepartmentTree)
  Web_Map For(*both) Fields((#tvDepts *PRIVATE))
  Web_Map For(*both) Fields((#Ancestors *PRIVATE))

  * build the tree if requested (this will happen the first time through).
  If (#BuildTree)
    #com_owner.BuildDepartmentList

    #BuildTree := False
  Endif
Endroutine
```

Note the Web_map definitions. Because this Webroutine is used to display the tree view, it has Web_maps for the working list that represents the tree, along with the working list that is used to hold ancestor items for a selected tree item. Wherever the tree view working list is specified as a Web_map, the ancestor list must also be specified as a Web_map. If it isn't, your WAM won't work correctly.

BuildTree is a Boolean field that is used to control the building of the tree view. Its default (in its field definition, at the top of the source) is True, which means that when this Webroutine is invoked for the first time, by the Navigation Panel, the tree view will be built. BuildTree is then set to False, ensuring that the tree is not rebuilt on subsequent invocations.

BuildDepartmentList method:

```
Mthroutine Name(BuildDepartmentList)

  Clr_List Named(#tvDepts)

  Select Fields(#DEPARTMENT #DEPTDESC) From_File(DEPTAB)

    #com_owner.AddListEntry I_Caption(#DEPARTMENT.BlankConcat(' ', #DEPTDESC)) I_Identifier(#DEPARTMENT)
    I_Level(1) I_Haschildren(Y) I_Detailwebroutine(ShowDepartmentDetail)
  Endselect
Endroutine
```

Note the setting for each entry to be added to the Tree View working list. Refer

to List Definitions in [A Closer Look at WAMEX51](#) for a full explanation of these.

The Role of the Tree View Target Weblet

1. Right-click on the ShowPage Webroutine again and select the LANSAs Editor option.
2. Click on the right-hand side of the Vertical Splitter and then click the Details tab. The properties for the Tree View Target will be displayed.

Note the `treeview_name` property points to the name of the Tree View weblet as defined in the DepartmentTree Webroutine. This indicates that the Tree View Target will receive selection events from the tree view. Effectively, it will become the active, target portion of the web page when something is selected in the tree.

This, in combination with the `list_onselect_wrname_field` property of the tree view, will display details of a selected Department, Section or Employee.

3. Close the LANSAs Editor.

DepartmentDetail Webroutine in the WAM source:

```
Webroutine Name>ShowDepartmentDetail)
  Web_Map For(*input) Fields(#TreeID)
  Web_Map For(*output) Fields((#DEPARTMENT *HIDDEN) #DEPTDESC)

  #DEPARTMENT := #TreeID

  #SelID := #DEPARTMENT

  Fetch Fields(#DEPTDESC) From_File(DEPTAB) With_Key(#DEPARTMENT)

Endroutine
```

This is the Webroutine that is invoked when a Department is selected in the tree view. Note the `*input Web_map`. This is the field specified against the `list_tag_field` property of the list view weblet. It contains the identifier, or key, of the selected tree item.

The `SelID` field is defined as a persistent session field (`Web_map` with `*NONE` and `*PERSIST`) and is used to hold the selected key on multiple invocations of the WAM. This is used when rebuilding the tree.

When the Webroutine ends, because the active portion of the web page is the Tree View Target, the output from the Webroutine is directed to it, so you see the department details in the right-hand portion of the Vertical Splitter.

Rebuilding the Tree View

1. Right-click on the ShowDepartmentDetail Webroutine and select the LANSA Editor option. Note that, as well as the department description, an Update push button is displayed. Click on it and select the Details tab. Its properties will be displayed, as follows:

With Parameters	
name	concat('o', position(), '_LANSA_5927')
caption	'Update'
reentryfield	'STDREENTRY'
reentryvalue	'M'
hide_if	false()
formname	'LANSA'
pos_absolute_design	'left:0pt;top:0pt;'
width_design	'0pt'
height_design	'0pt'
on_click_wamname	/bxml:data/bxml:context/bxml:weba
on_click_wrname	'UpdateDepartment'
protocol	
show_in_new_window	false()
target_window_name	'TreeView'

Note the on_click_wrname and target_window_name properties. The on_click_wrname property contains the name of the Webroutine to invoke when the push button is clicked.

The target_window_name property is used to direct the output of the WAM to a specified window. In effect, this becomes the active portion of the web page. In this example, the Navigation Panel will be the target window.

2. Close the LANSA Editor.

UpdateDepartment Webroutine in the WAM source:

```
Webroutine Name(UpdateDepartment)
  Web_Map For(*input) Fields(#DEPARTMENT #DEPTDESC)

  Update Fields(#DEPTDESC) In_File(DEPTAB) With_Key(#DEPARTMENT)

  * force the tree to be rebuilt.
  #Buildtree := True

  * rebuild and re-display the tree.
  Transfer Toroutine(DepartmentTree)
Endroutine
```

Note that the BuildTree boolean field is set to True, indicating that the tree should be rebuilt. Control is then transferred to the DepartmentTree Webroutine, from where the tree view is rebuilt. Have another look at the AddListEntry

method:

```
Mthroutine Name(AddListEntry)
  Define_Map For(*input) Class(#PRIM_ALPH) Name(#_Caption)
  Define_Map For(*input) Class(#PRIM_ALPH) Name(#_Identifier)
  Define_Map For(*input) Class(#PRIM_NMBR) Name(#_Level)
  Define_Map For(*input) Class(#PRIM_ALPH) Name(#_HasChildren)
  Define_Map For(*input) Class(#PRIM_ALPH) Name(#_DetailWebroutine)

  #TreeCapt := #_Caption
  #TreeID := #_Identifier
  #TreeLvl := #_Level
  #HasKids := #_HasChildren
  #DetailWR := #_DetailWebroutine

  If (#TreeID *EQ #SelID)
    #SELECTED := Freeze
    #SelID := *BLANKS
  Else
    #SELECTED := *BLANKS
  Endif

  Add_Entry To_List(#tvDepts)
Endroutine
```

Note the If/Else/Endif code. Remember the SelID field? If the entry being added to the tree is the same one that has just been updated, the SELECTED field is set to 'freeze'. SELECTED is a field in the tree view working list which drives the list_is_selected_field property of the tree view. Setting it to freeze does two things: it pre-selects the tree view item and it stops the TreeViewTarget from being reloaded.

Processing Expanding Tree Items

Open the DepartmentTree Webroutine in the LANSAs Editor and select the tree view weblet. Click the Details tab to display its properties. Note the onexpand_wrname property is set to TreeExpanding. This is the Webroutine to be invoked when an expander (+) of a tree item is clicked on. Close the LANSAs Editor and have a look at the TreeExpanding Webroutine in the WAM source:

```

Webroutine Name(TreeExpanding)
  Web_Map For(*input) Fields(#TreeID)
  Web_Map For(*input) Fields(#TreeLvl)
  Web_Map For(*both) Fields((#tvDepts *PRIVATE))
  Web_Map For(*input) Fields((#Ancestors *PRIVATE))

  Case (#TreeLvl)
    * a department is expanding - add sections to list.
  When (*EQ 1)

    #com_owner.BuildSectionsList( #TreeID )

    * a section is expanding - add employees to list.
  When (*EQ 2)

    #SECTION := #TreeID
    #DEPARTMENT := #com_owner.GetAncestor( 1 )

    #com_owner.BuildEmployeesList( #DEPARTMENT, #SECTION )

  Endcase

  * re-display the tree.
  Transfer Toroutine(DepartmentTree)

Endroutine

```

Note the *input field Web_maps: the TreeID field, which holds the key information of the expanding tree item, and the TreeLvl field, which indicates the level of the expanding tree item. The tree view working list is also passed in, along with the list of ancestors of the expanding tree item. Note that the tree view working list is specified as *both – it will be passed to the DepartmentTree Webroutine at the end of the TreeExpanding routine. Also, the ANCESTORS list is *input – it is only needed by this routine.

The Case statement determines what should be added to the tree view working list. If the level is 1, it means a Department tree item is being expanded, so Sections need to be added. If it's 2, a Section is being expanded and Employees need to be added. Once entries have been added to the tree working list, control is transferred back to the DepartmentTree Webroutine which displays the tree.

In [Accessing Ancestor Information](#), the focus is on what happens if a Section tree item is being expanded in order to show what you need to do to retrieve parent key information.

Accessing Ancestor Information

Refer back to [Processing Expanding Tree Items](#). When processing level 2, the SECTION field is set to the incoming identifier (this was set when the Sections

were added to the tree view working list in the BuildSectionsList method). Of course, a Section has a parent of Department.

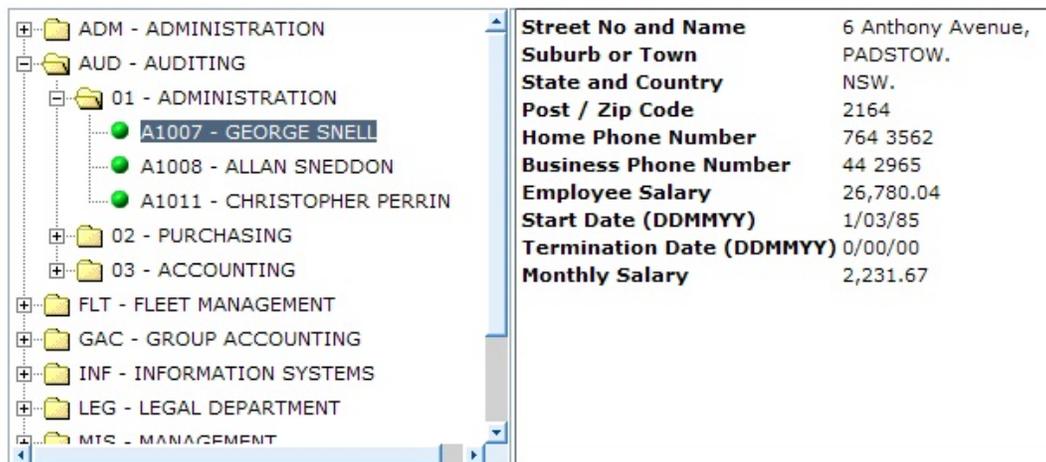
In order to determine the Department to which the Section being expanded belongs, the ANCESTORS working list is used. This list contains multiple entries of the TreeID field. A Section only has **one** parent, so the ancestor working list will contain **one** entry. In this example, the GetAncestor method is executed to retrieve the key value of the Section's parent.

If the tree contained more levels, the ANCESTOR working list would contain more than one entry for a level three or higher item, and so multiple Get_entries could be used to build up the full list of parent keys.

Tree View Target (std_treeview_target)

The Tree View Target weblet is a container-type control used in conjunction with the Tree View weblet. It responds to selection events from the Tree View and, as such, can be used to display information based on that selection.

The right-hand portion of the image below shows it in action (showing information for the selected employee in the tree view):



The screenshot displays a Tree View Target weblet. On the left, a tree structure shows a hierarchy of folders: ADM - ADMINISTRATION, AUD - AUDITING, 01 - ADMINISTRATION, 02 - PURCHASING, 03 - ACCOUNTING, FLT - FLEET MANAGEMENT, GAC - GROUP ACCOUNTING, INF - INFORMATION SYSTEMS, LEG - LEGAL DEPARTMENT, and MTS - MANAGEMENT. Under the '01 - ADMINISTRATION' folder, three employees are listed: A1007 - GEORGE SNELL (selected), A1008 - ALLAN SNEDDON, and A1011 - CHRISTOPHER PERRIN. On the right, a table displays the details for the selected employee, A1007 - GEORGE SNELL.

Street No and Name	6 Anthony Avenue,
Suburb or Town	PADSTOW.
State and Country	NSW.
Post / Zip Code	2164
Home Phone Number	764 3562
Business Phone Number	44 2965
Employee Salary	26,780.04
Start Date (DDMMYY)	1/03/85
Termination Date (DDMMYY)	0/00/00
Monthly Salary	2,231.67

QuickStart - Tree View Target

An example of how the Tree View Target weblet interacts with the Tree View weblet is included in [An In-Depth Look at the Tree View Weblet](#).

Properties - Tree View Target

The Tree View Target weblet's properties are:

<code>treeview_name</code>	<code>wamname</code>	<code>tag_fieldname_alias</code>
<code>formname</code>	<code>wrname</code>	<code>resize_to_content</code>
<code>pos_absolute_design</code>	<code>reentryfield</code>	<code>class</code>
<code>width_design</code>	<code>reentryvaluereentryfield</code>	<code>bg_colortreeview_name</code>
<code>height_designbg_color</code>		

treeview_name

The name of the tree view weblet this tree view target is to be associated with. Selection events fired by the tree view will be heard by this target. The result of this is that the Tree View Target becomes the 'active' panel, and will have content directed to it by the next Webroutine that is run.

Default value

`concat('oTree', ancestor-or-self::lxml:list/@name,position())` – this is the internal name given to the tree view by LANSA.

Valid values

A name for the form, in single quotes. A list of known form names is available by clicking the corresponding dropdown button in the property sheet.

formname

The name of the HTML form that is posted to the server.

Default value

'LANSA'

Valid values

A name for the form, in single quotes. A list of known form names is available by clicking the corresponding dropdown button in the property sheet.

pos_absolute_design

The absolute position of the weblet on the web page. Note that 'Position Absolutely' must be selected from the weblet's right-click menu for this property to be used. The property will usually be set in pixels by dragging and dropping the weblet.

Default value

Blank (not positioned).

Valid values

Valid 'left' and 'top' coordinates, in valid units of measurement, in single quotes.

width_design

The width of the weblet on the web page.

Default value

'100%' (this is equivalent to the weblet adopting the width of its container).

Valid values

A width, in a valid unit of measurement, in single quotes.

height_design

The height of the weblet on the web page.

Default value

Blank (weblet uses its default height).

Valid values

A height, in a valid unit of measurement, in single quotes.

wamname

The name of the WAM to be invoked when an item in the associated tree view is selected.

Default value

Blank – the current WAM is invoked, but only if the wrname property is specified.

Valid values

A valid WAM name, in single quotes. To choose from a list of known WAMs, click the corresponding dropdown button in the property sheet.

wrname

The name of the Webroutine to be invoked when an item in the associated tree view is selected.

Default value

Blank – the current WAM is invoked, but only if the wrname property is specified.

Valid values

A valid WAM name, in single quotes. To choose from a list of known WAMs, click the corresponding dropdown button in the property sheet.

reentryfield

The field name to be used to post to the WAM the value that is specified in the reentryvalue property. The field name should be in single quotes.

Default value

'STDREENTRY'

Valid values

Any repository- or WAM-defined field name. A list of known field names is available by clicking the corresponding dropdown button in the property sheet.

reentryvalue

The value to post into the field specified in the reentryfield property. If that field is alphanumeric, the value must be specified in single quotes. If it is numeric, the value can be specified with or without quotes.

Default value

'D'

Valid values

Any appropriate literal.

tag_fieldname_alias

When a Webroutine is invoked to navigate to a page for this panel, the field posted to it is the tag field, as specified in the list_tag_field property of the associated tree view. If the field required by the target Webroutine is different to the tag field name, it can be specified here.

Default value

Blank – the field name specified in the list_tag_field property of the associated tree view is used.

Valid values

A valid field name, in single quotes. Click the corresponding dropdown button in the property sheet to choose from a list of known field names.

resize_to_content

A Boolean property that indicates whether the panel will be resized to the content size of the page navigated to.

Default value

true() – the panel will be resized.

Valid values

true(), false() or a valid expression.

class

The Cascading Style Sheet class name of the weblet.

Default value

'std_treeview_target' - The name of the shipped class for the weblet.

Valid values

Any valid class name from the Cascading Style Sheet, in single quotes. A list of available classes can be selected from by clicking the corresponding dropdown button in the property sheet.

bg_color

The background color of the weblet.

Default value

Blank. The background color will be the same as the web page.

Valid values

A valid color, either in #RRGGBB format or a valid color name , in single quotes. A color can be chosen from a palette by clicking on the corresponding ellipses button in the property sheet.

Example

Both of the following examples will set the background color to red:

 bg_color	#FF0000'
--	----------

or

 bg_color	'red'
--	-------

Date (std_date)

QuickStart - Date Properties - Date

The date weblet provides a text input box control with added features to support the display, entry, prompting and validation of dates. It broadly corresponds to the `<input type="text">` HTML element.

An example of the weblet is shown below. In this example, the (optional) calendar prompt button has been clicked and the calendar prompt window is visible.



The date weblet is best used with fields of type date. If you use this type, the data will automatically be passed in the format expected by the weblet. You can use the date weblet with fields of other numeric types such as packed or signed, but it is your responsibility to ensure the numeric value is formatted in the correct ISO format expected by the date weblet. For example, you could use a signed (8, 0) field containing a date in YYYYMMDD format with an edit word ('0 - - ') to format it as an ISO date format.

QuickStart - Date

For date fields that have `std_date` as their default visualization, you do not need to manually add them to your web page. Simply include your date fields in your `web_map` or in a list that is present in your `web_map` and they will be visualized using the date weblet. Similarly fields of type time and of type datetime will be visualized using the time (`std_time`) and datetime (`std_datetime`) weblets.

If you do need to add the date weblet to your page manually, simply drag the date field from the Fields tab onto your page. Alternatively, open the XSL for your webroutine in the LANSAs Editor and follow these steps:

1. Click on the Weblets tab, select Standard Field Visualization from the drop-down list near the top and locate the Date weblet.
2. Drag the weblet onto your page in the Design view. Click on the weblet and then click on the Details tab.
3. Set the name and value properties as required to associate the weblet with the required field in your webroutines `web_map`.
4. You may also wish to set the `date_mask` property as required.

Properties - Date

The Date weblet's properties are:

allow_sqlnull	display_mode	read_only
button_image	hide_calendar	tab_index
button_title	hide_if	title
class	name	value
date_mask	onchange_script	width
disabled	pos_absolute	

name

The name the weblet is identified with. If the weblet visualizes a field, this is the name of the field. Normally, you would leave this as the default and let LANSA use its own internal naming convention. However, you may want to use your own name if using JavaScript or XSL that references the weblet.

Default value

Where the weblet visualizes a field the default name is the field name or combines the field name with a row number (for fields in a list). Otherwise the default name is an automatically generated, unique identifier.

Valid values

Single-quoted text.

value

The value to set the weblet to. If the weblet visualizes a field, this will identify the field whose value is to be shown.

Default value

No default value applies – for most uses of this weblet you must specify a field whose value is to be represented by the input box and/or that is used to receive the contents of the input box.

Valid values

Single-quoted text or the name of a field, system variable or multilingual text variable.

Example

This shows how the value is specified when the weblet visualizes a field:



A screenshot of a property editor for a weblet. The property name is 'value' and its value is '#V1DAT04'. The editor has a blue header bar with the property name and a white input field with the value.

When the property loses focus, the value is shown as follows:



A screenshot of a property editor for a weblet. The property name is 'value' and its value is 'key('field-value', 'V1DAT04')'. The editor has a white header bar with the property name and a white input field with the value.

display_mode

Controls whether the weblet accepts input, displays output or is hidden.

Default value

Blank (equivalent to 'input').

Valid values

Literal values 'input', 'output' or 'hidden'. A list of allowable values is available by clicking the corresponding dropdown button in the property sheet. Alternately, you may enter the name of a field, system variable or multilingual variable that will contain one of the allowable values at run-time.

hide_if

An expression which, if evaluated to be True, will hide the weblet.

Default value

False() (that is, the weblet will always be shown)

Valid values

Any valid XPath expression that returns a Boolean value.

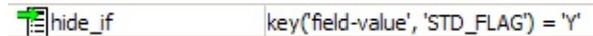
Example

This example will hide the weblet if field #STD_FLAG is equal to 'Y'. The expression should be entered in this form:



A screenshot of a weblet property editor. The property name 'hide_if' is highlighted in a dark blue box. The value entered is '#STD_FLAG = 'Y''. A small icon of a document with a green checkmark is visible to the left of the property name.

When the property loses focus, the expression is shown as follows:



A screenshot of a weblet property editor. The property name 'hide_if' is highlighted in a dark blue box. The value entered is 'key('field-value', 'STD_FLAG') = 'Y''. A small icon of a document with a green checkmark is visible to the left of the property name.

allow_sqlnull

A Boolean property which determines if the date value can be left blank.

Note: This property must be consistent with the fields's repository definition (ASQN attribute).

Default value

false(). If the weblet is dropped over a field, it defaults to the ASQN attribute of the field's repository definition.

Valid values

true(), false() or a valid expression.

date_mask

Specifies the format or mask used to display and enter the date for the weblet. This is a string containing a number of format specifiers that tell the weblet how to format the date.

See the DateTime weblet for a full list of valid format specifiers.

Note: this specifies the presentation format the weblet uses. The input and output date received from and returned to the webroutine are always in ISO format. If you choose a different presentation format by setting this property, the weblet will convert to and from the internal representation as required.

Default value

'YYYY-MM-DD'

Valid values

Any string containing valid format specifiers.

button_image

The path and file name, relative to the images virtual directory, of the image to display on the calendar prompt button.

Default value

'fp_im003.gif' (this image is shipped with LANSA).

Valid values

The path and name of an image, relative to the images directory, enclosed in single quotes. An image can be chosen from a prompter by clicking the corresponding ellipses button in the property sheet.

tab_index

Determines the tab order of the weblet on the form. The `tab_index` property value determines the tab order as follows:

1. Objects with a positive `tab_index` are selected in increasing `tab_index` order (and in source order to resolve duplicates).
2. Objects with a `tab_index` of zero or blank (the default) are selected in source order.
3. Objects with a negative `tab_index` are omitted from the tabbing order. Note that this behavior is not defined in the HTML specifications and is only supported by Internet Explorer and Firefox.

Default value

Blank. The weblet is selected in source order.

Valid values

Blank or a valid numeric value.

Example

 tab_index	3
---	---

title

Specifies text for the weblet that may display as tip text as the mouse moves over the weblet.

Default value

Blank – no tip text will be displayed.

Valid values

Single-quoted text or the name of a multilingual text variable (the corresponding ellipses button in the property sheet can be clicked to choose one from a list).

button_title

Specifies text for the calendar button (if shown) that may display as tip text as the mouse moves over the button.

Default value

Blank – the text specified for the title property is used.

Valid values

Single-quoted text or the name of a multilingual text variable (the corresponding ellipses button in the property sheet can be clicked to choose one from a list).

read_only

A boolean property, the result of which determines whether the content of the weblet is read-only (that is, the user cannot modify the content).

Default value

Blank – equivalent to False (that is, the user can modify the contents).

Valid values

true(), false() or a valid expression.

Example

This example will set the weblet to read-only if field #STD_FLAG is equal to 'Y'. The expression should be entered in this form:



A screenshot of a property editor interface. On the left, there is a dropdown menu with 'read_only' selected. To the right of this menu is a text input field containing the expression '#STD_FLAG = 'Y''. The entire input area is enclosed in a thin border.

When the property loses focus, the expression is shown as follows:



A screenshot of a property editor interface, similar to the one above. The dropdown menu on the left still shows 'read_only'. The text input field to the right now contains the expression 'key('field-value', 'STD_FLAG') = 'Y''. The input area is enclosed in a thin border.

disabled

A boolean property, the result of which determines whether the weblet appears enabled or disabled.

Default value

Blank – equivalent to False (that is, the weblet will always be enabled).

Valid values

true(), false() or a valid expression.

hide_calendar

A boolean property, the result of which determines whether the calendar button is shown for the weblet.

Default value

Blank – equivalent to false (that is, the calendar button will be shown).

Valid values

true(), false() or a valid expression.

class

The Cascading Style Sheet (CSS) class name of the weblet.

Default value

The name of the shipped class for the weblet.

Valid values

Any valid class name from the Cascading Style Sheet, in single quotes. A list of available classes can be selected from by clicking the corresponding dropdown button in the property sheet.

pos_absolute

The absolute position of the weblet on the web page. Note that 'Position Absolutely' must be selected from the weblet's right-click menu for this property to be used. The property will usually be set in pixels by dragging and dropping the weblet.

Default value

Blank (this is equivalent to the weblet being positioned relatively).

Valid values

Valid 'left' and 'top' coordinates, in valid units of measurement, in single quotes.

width

The width of the weblet on the web page. The weblet will reserve a minimum width based on the data to be displayed.

Usually you would set the width of the weblet by dragging the grab-handles around the weblet in the Design view of the LANSAs Editor. Doing so updates the value of the width property. However you can directly edit the property value if required.

Default value

Blank (this is equivalent to the weblet adopting its default width).

Valid values

A width, in a valid unit of measurement, in single quotes.

onchange_script

JavaScript code to be run when the input box loses focus after the text has been changed. JavaScript statements must be terminated by a semicolon.

Default value

Blank. No JavaScript is run.

Valid values

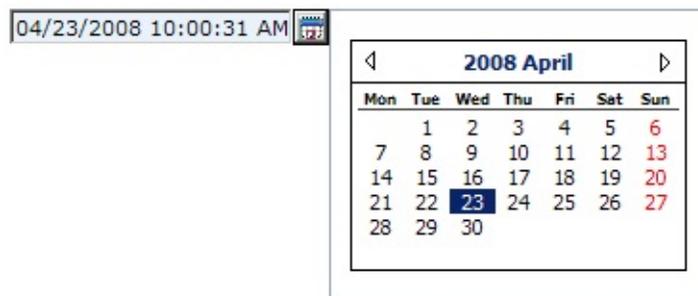
Any valid JavaScript statement(s).

DateTime (std_datetime)

[QuickStart - DateTime Properties - DateTime](#)

The datetime weblet provides a text input box control with added features to support the display, entry, prompting and validation of date and/or time values. It broadly corresponds to the `<input type="text">` HTML element.

An example of the weblet is shown below. In this example, the (optional) calendar prompt button has been clicked and the calendar prompt window is visible:



The datetime weblet is used to display and receive input for fields containing dates, times or datetimes. If your field contains only a date or only a time you may prefer to use one of the specialized weblets that are based on this weblet: `std_date` or `std_time`.

The datetime weblet is best used with fields of date, time or datetime data types. If you use these types, the data will automatically be passed in the format expected by the weblet. You can use the datetime weblet with fields of other numeric types such as packed or signed, but it is your responsibility to ensure the numeric value is formatted in the correct ISO format expected by the datetime weblet. For example, you could use a signed (14, 0) field containing a date and time in YYYYMMDDHHMMSS format with an edit word ('0 - - & : : ') to format it as an ISO date and time format.

QuickStart - DateTime

For datetime fields that have datetime weblet as their the default visualization, you usually do not need to manually add it to your web page. Simply include your datetime fields in your web_map or in a list that is present in your web_map and they will be visualized using the datetime weblet. Similarly fields of type date and of type time will be visualized using the date (std_date) and time (std_time) weblets.

If you do need to add the datetime weblet to your page manually, simply drag the datetime field from the Fields tab onto your page. Alternatively, open the XSL for your webroutine in the LANSA Editor and follow these steps:

1. Click on the Weblets tab, select Standard Field Visualization from the drop-down list near the top and locate the Datetime weblet.
2. Drag the weblet onto your page in the Design view. Click on the weblet and then click on the Details tab.
3. Set the name and value properties as required to associate the weblet with the required field in your webroutines web_map.
4. You may also wish to set the size and date_mask properties as required.

Properties - DateTime

The DateTime weblet's properties are:

allow_sqlnull	display_in_utc	read_only
button_image	hide_calendar	size
button_title	hide_if	tab_index
class	input_type	time_mask
date_mask	name	title
disabled	onchange_script	value
display_mode	pos_absolute	width

name

The name the weblet is identified with. If the weblet visualizes a field, this is the name of the field. Normally, you would leave this as the default and let LANSA use its own internal naming convention. However, you may want to use your own name if using JavaScript or XSL that references the weblet.

Default value

Where the weblet visualizes a field the default name is the field name or combines the field name with a row number (for fields in a list). Otherwise the default name is an automatically generated, unique identifier.

Valid values

Single-quoted text.

value

The value to set the weblet to. If the weblet visualizes a field, this will identify the field whose value is to be shown.

Default value

No default value applies – for most uses of this weblet you must specify a field whose value is to be represented by the input box and/or that is used to receive the contents of the input box.

Valid values

Single-quoted text or the name of a field, system variable or multilingual text variable.

Example

This shows how the value is specified when the weblet visualizes a field:



A screenshot of a property editor for a weblet. The property name is 'value' and its value is '#V1DAT04'. The editor has a dark blue header bar with the property name and a white input field with the value.

When the property loses focus, the value is shown as follows:



A screenshot of a property editor for a weblet. The property name is 'value' and its value is 'key('field-value', 'V1DAT04')'. The editor has a light gray header bar with the property name and a white input field with the value.

display_mode

Controls whether the weblet accepts input, displays output or is hidden.

Default value

Blank (equivalent to 'input').

Valid values

Literal values 'input', 'output' or 'hidden'. A list of allowable values is available by clicking the corresponding dropdown button in the property sheet. Alternately, you may enter the name of a field, system variable or multilingual variable that will contain one of the allowable values at run-time.

size

The size of the weblet data in characters. – the browser sizes the input box according to the number of characters specified, but will reserve a minimum width based on the data to be displayed. Sizing the weblet by dragging the grab handles (or manually specifying the width property) can increase the width but not reduce it beyond the minimum that the weblet determines.

Default value

24.

Valid values

A numeric value. Usually you set this according to the field definition. If the weblet is to display only a date, specify 10. If the weblet is to display only a time, specify 11 or greater. If the weblet is to display a date and time, specify 22 or greater. When a time is included in the data, remember to allow room for the AM/PM indicator.

hide_if

An expression which, if evaluated to be True, will hide the weblet.

Default value

False() (that is, the weblet will always be shown)

Valid values

Any valid XPath expression that returns a Boolean value.

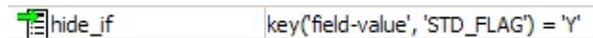
Example

This example will hide the weblet if field #STD_FLAG is equal to 'Y'. The expression should be entered in this form:



A screenshot of a weblet property editor. The property name 'hide_if' is highlighted in a dark blue box. To its right, the value '#STD_FLAG = 'Y'' is entered in a text field.

When the property loses focus, the expression is shown as follows:



A screenshot of a weblet property editor. The property name 'hide_if' is highlighted in a dark blue box. To its right, the value 'key('field-value', 'STD_FLAG') = 'Y'' is entered in a text field.

input_type

Specifies whether the weblet displays and receives time, date or datetime data.

If the weblet is to display and receive time data only, then you may prefer to use the Time (std_time) weblet. If the weblet is to display and receive date data only, then you may prefer to use the Date (std_date) weblet instead. Both of these weblets are a specialization of the Datetime weblet.

Default value

'datetime'

Valid values

'timeonly', 'dateonly' or 'datetime'. A list of the valid values may be displayed by clicking the corresponding dropdown button in the property sheet.

display_in_utc

A Boolean property which determines if the datetime displays the datetime's UTC value or the datetime's local value.

Default value

false(). If the weblet is dropped over a field, it defaults to the DUTC attribute of the field's repository definition.

Valid values

true(), false() or a valid expression.

allow_sqlnull

A Boolean property which determines if the datetime value can be left blank.

Note: This property must be consistent with the fields's repository definition (ASQN attribute).

Default value

false(). If the weblet is dropped over a field, it defaults to the ASQN attribute of the field's repository definition.

Valid values

true(), false() or a valid expression.

date_mask

Specifies the format or mask used to display and enter the date for the weblet. This is a string that contains a number of format specifiers that tell the weblet how to format the date.

Valid format specifiers are:

Format Specifier	Description
YYYY	Represents the year as a four-digit number.
YY	Represents the year as a two-digit number.
MMMM	Represents the name of the month as defined in <code>std_script_messages.js</code>
MM	Represents the month as a number from 01 through 12. A single-digit month is formatted with a leading zero.
M	Represents the month as a number from 1 through 12.
DDD	Represents the name of the day of the week as defined in <code>std_script_messages.js</code>
DD	Represents a day of the month from 1 - 31. A single digit day is formatted with a leading zero.
D	Represents a day of the month from 1 - 31.

Note: This specifies the presentation format the weblet uses. The input and output date received from and returned to the webroutine are always in ISO format. If you choose a different presentation format by setting this property, the weblet will convert to and from the internal representation as required.

Default value

'YYYY-MM-DD'

Valid values

Any string containing valid format specifiers

time_mask

Specifies the format or mask used to display and enter the time for the weblet. This is a string containing a number of format specifiers that tell the weblet how to format the time.

Valid format specifiers are:

Format Specifier	Description
-------------------------	--------------------

HH	Represents the hour as a number from 0 through 23, that is, the hour as represented by a zero-based 24-hour clock that counts the hours since midnight. A single-digit hour is formatted with a leading zero.
H	Represents the hour as a number from 0 through 23, that is, the hour as represented by a zero-based 24-hour clock that counts the hours since midnight.
hh	Represents the hour as a number from 1 through 12, that is, the hour as represented by a 12-hour clock that counts the whole hours since midnight or noon. A single-digit hour is formatted with a leading zero.
h	Represents the hour as a number from 1 through 12, that is, the hour as represented by a 12-hour clock that counts the whole hours since midnight or noon.
mm	Represents the minute as a number from 0 through 59. A single-digit minute is formatted with a leading zero.
m	Represents the minute as a number from 0 through 59.
ss	Represents the seconds as a number from 00 through 59. A single-digit second is formatted with a leading zero.
s	Represents the seconds as a number from 00 through 59.
sss	Represents the milliseconds as a number from 000 through 999 All values are represented as three digits.
t	Represents A.M. or P.M.

Note: This specifies the presentation format the weblet uses. The input and output date received from and returned to the webroutine are always in ISO format. If you choose a different presentation format by setting this property, the weblet will convert to and from the internal representation as required.

Default value

'HH:mm:ss'

Valid values

Any string containing valid format specifiers

button_image

The path and file name, relative to the images virtual directory, of the image to display on the calendar prompt button.

Default value

'fp_im003.gif' (this image is shipped with LANSA).

Valid values

The path and name of an image, relative to the images directory, enclosed in single quotes. An image can be chosen from a prompter by clicking the corresponding ellipses button in the property sheet.

tab_index

Determines the tab order of the weblet on the form. The `tab_index` property value determines the tab order as follows:

1. Objects with a positive `tab_index` are selected in increasing `tab_index` order (and in source order to resolve duplicates).
2. Objects with a `tab_index` of zero or blank (the default) are selected in source order.
3. Objects with a negative `tab_index` are omitted from the tabbing order. Note that this behavior is not defined in the HTML specifications and is only supported by Internet Explorer and Firefox.

Default value

Blank. The weblet is selected in source order.

Valid values

Blank or a valid numeric value.

title

Specifies text for the weblet that may display as tip text as the mouse moves over the weblet.

Default value

Blank – no tip text will be displayed.

Valid values

Single-quoted text or the name of a multilingual text variable (the corresponding ellipses button in the property sheet can be clicked to choose one from a list).

button_title

Specifies text for the calendar button (if shown) that may display as tip text as the mouse moves over the button.

Default value

Blank – the text specified for the title property is used.

Valid values

Single-quoted text or the name of a multilingual text variable (the corresponding ellipses button in the property sheet can be clicked to choose one from a list).

read_only

A boolean property, the result of which determines whether the content of the weblet is read-only (that is, the user cannot modify the content).

Default value

Blank – equivalent to False (that is, the user can modify the contents).

Valid values

true(), false() or a valid expression.

Example

This example will set the weblet to read-only if field #STD_FLAG is equal to 'Y'. The expression should be entered in this form:



A screenshot of a property editor interface. On the left, there is a dropdown menu with 'read_only' selected. To the right of this menu is a text input field containing the expression '#STD_FLAG = 'Y''. The entire input area is enclosed in a thin border.

When the property loses focus, the expression is shown as follows:



A screenshot of a property editor interface, similar to the one above. The dropdown menu on the left still shows 'read_only'. The text input field to the right now contains the expression 'key('field-value', 'STD_FLAG') = 'Y''. The input area is enclosed in a thin border.

disabled

A boolean property, the result of which determines whether the weblet appears enabled or disabled.

Default value

Blank – equivalent to False (that is, the weblet will always be enabled).

Valid values

true(), false() or a valid expression.

hide_calendar

A boolean property, the result of which determines whether the calendar button is shown for the weblet.

Default value

Blank – equivalent to false (that is, the calendar button will be shown if the weblet displays a date or datetime).

Valid values

true(), false() or a valid expression.

class

The Cascading Style Sheet (CSS) class name of the weblet.

Default value

The name of the shipped class for the weblet.

Valid values

Any valid class name from the Cascading Style Sheet, in single quotes. A list of available classes can be selected from by clicking the corresponding dropdown button in the property sheet.

pos_absolute

The absolute position of the weblet on the web page. Note that 'Position Absolutely' must be selected from the weblet's right-click menu for this property to be used. The property will usually be set in pixels by dragging and dropping the weblet.

Default value

Blank (this is equivalent to the weblet being positioned relatively).

Valid values

Valid 'left' and 'top' coordinates, in valid units of measurement, in single quotes.

width

The width of the weblet on the web page.

Usually you would set the width of the weblet by dragging the grab-handles around the weblet in the Design view of the LANSAs Editor. Doing so updates the value of the width property. However you can directly edit the property value if required.

The width can also be affected by the size property, but in any event, the weblet will reserve a minimum width based on the data to be displayed.

Default value

Blank (this is equivalent to the weblet adopting its default width).

Valid values

A width, in a valid unit of measurement, in single quotes.

onchange_script

JavaScript code to be run when the input box loses focus after the text has been changed. JavaScript statements must be terminated by a semicolon.

Default value

Blank. No JavaScript is run.

Valid values

Any valid JavaScript statement(s).

Time (std_time)

[QuickStart - Time Properties - Time](#)

The time weblet provides a text input box control with added features to support the display, entry and validation of times. It broadly corresponds to the `<input type="text">` HTML element.

An example of the weblet is shown below (for clarity it is shown with its label):

Sample time

The time weblet is best used with fields of type time. If you use this type, the data will automatically be passed in the format expected by the weblet. You can use the time weblet with fields of other numeric types such as packed or signed, but it is your responsibility to ensure the numeric value is formatted in the correct ISO format expected by the time weblet. For example, you could use a signed (6, 0) field containing a time in HHMMSS format with an edit word ('0 : : ') to format it as an ISO time format.

QuickStart - Time

For time fields for which this weblet is the default visualization, you usually do not need to manually add it to your web page. Simply include your time fields in your web_map or in a list that is present in your web_map and they will be visualized using the time weblet. Similarly fields of type date and of type datetime will be visualized using the date (std_date) and datetime (std_datetime) weblets.

If you do need to add the time weblet to your page manually, simply drag the time field from the Fields tab onto your page. Alternatively, open the XSL for your webroutine in the LANSA Editor and follow these steps:

1. Click on the Weblets tab, select Standard Field Visualization from the drop-down list near the top and locate the Time weblet.
2. Drag the weblet onto your page in the Design view. Click on the weblet and then click on the Details tab.
3. Set the name and value properties as required to associate the weblet with the required field in your webroutines web_map.

Properties - Time

The Time weblet's properties are:

allow_sqlnull	name	time_mask
class	onchange_script	title
disabled	pos_absolute	value
display_mode	read_only	width
hide_if	tab_index	

name

The name the weblet is identified with. If the weblet visualizes a field, this is the name of the field. Normally, you would leave this as the default and let LANSA use its own internal naming convention. However, you may want to use your own name if using JavaScript or XSL that references the weblet.

Default value

Where the weblet visualizes a field the default name is the field name or combines the field name with a row number (for fields in a list). Otherwise the default name is an automatically generated, unique identifier.

Valid values

Single-quoted text.

value

The value to set the weblet to. If the weblet visualizes a field, this will identify the field whose value is to be shown.

Default value

No default value applies – for most uses of this weblet you must specify a field whose value is to be represented by the input box and/or that is used to receive the contents of the input box.

Valid values

Single-quoted text or the name of a field, system variable or multilingual text variable.

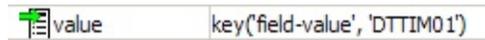
Example

This shows how the value is specified when the weblet visualizes a field:



A screenshot of a property editor for a weblet. The 'value' property is highlighted in blue, and its value is set to '#DTTIM01'.

When the property loses focus, the value is shown as follows:



A screenshot of a property editor for a weblet. The 'value' property is highlighted in blue, and its value is set to 'key('field-value', 'DTTIM01')'.

time_mask

Specifies the format or mask used to display and enter the time for the weblet. This is a string containing a number of format specifiers that tell the weblet how to format the date.

See the DateTime weblet for a full list of valid format specifiers.

Note: this specifies the presentation format the weblet uses. The input and output date received from and returned to the webroutine are always in ISO format. If you choose a different presentation format by setting this property, the weblet will convert to and from the internal representation as required.

Default value

'HH:mm:ss'

Valid values

Any string containing valid format specifiers.

display_mode

Controls whether the weblet accepts input, displays output or is hidden.

Default value

Blank (equivalent to 'input').

Valid values

Literal values 'input', 'output' or 'hidden'. A list of allowable values is available by clicking the corresponding dropdown button in the property sheet. Alternately, you may enter the name of a field, system variable or multilingual variable that will contain one of the allowable values at run-time.

hide_if

An expression which, if evaluated to be True, will hide the weblet.

Default value

False() (that is, the weblet will always be shown)

Valid values

Any valid XPath expression that returns a Boolean value.

Example

This example will hide the weblet if field #STD_FLAG is equal to 'Y'. The expression should be entered in this form:



A screenshot of a weblet property editor. The property name 'hide_if' is highlighted in a dark blue box. The value entered is '#STD_FLAG = 'Y''. A small icon of a document with a green checkmark is visible to the left of the property name.

When the property loses focus, the expression is shown as follows:



A screenshot of a weblet property editor. The property name 'hide_if' is highlighted in a dark blue box. The value entered is 'key('field-value', 'STD_FLAG') = 'Y''. A small icon of a document with a green checkmark is visible to the left of the property name.

allow_sqlnull

A Boolean property which determines if the time value can be left blank.

Note: This property must be consistent with the fields's repository definition (ASQN attribute).

Default value

false(). If the weblet is dropped over a field, it defaults to the ASQN attribute of the field's repository definition.

Valid values

true(), false() or a valid expression.

tab_index

Determines the tab order of the weblet on the form. The tab_index property value determines the tab order as follows:

1. Objects with a positive tab_index are selected in increasing tab_index order (and in source order to resolve duplicates).
2. Objects with a tab_index of zero or blank (the default) are selected in source order.
3. Objects with a negative tab_index are omitted from the tabbing order. Note that this behavior is not defined in the HTML specifications and is only supported by Internet Explorer and Firefox.

Default value

Blank. The weblet is selected in source order.

Valid values

Blank or a valid numeric value.

title

Specifies text for the weblet that may display as tip text as the mouse moves over the weblet.

Default value

Blank – no tip text will be displayed.

Valid values

Single-quoted text or the name of a multilingual text variable (the corresponding ellipses button in the property sheet can be clicked to choose one from a list).

read_only

A Boolean property, the result of which determines whether the content of the weblet is read-only (that is, the user cannot modify the content).

Default value

Blank – equivalent to False (that is, the user can modify the contents).

Valid values

true(), false() or a valid expression.

Example

This example will set the weblet to read-only if field #STD_FLAG is equal to 'Y'. The expression should be entered in this form:



A screenshot of a property editor interface. On the left, there is a dropdown menu with 'read_only' selected. To the right of the dropdown is a text input field containing the expression '#STD_FLAG = 'Y''.

When the property loses focus, the expression is shown as follows:



A screenshot of a property editor interface. On the left, there is a dropdown menu with 'read_only' selected. To the right of the dropdown is a text input field containing the expression 'key('field-value', 'STD_FLAG') = 'Y''.

disabled

A boolean property, the result of which determines whether the weblet appears enabled or disabled.

Default value

Blank – equivalent to False (that is, the weblet will always be enabled).

Valid values

true(), false() or a valid expression.

class

The Cascading Style Sheet (CSS) class name of the weblet.

Default value

The name of the shipped class for the weblet.

Valid values

Any valid class name from the Cascading Style Sheet, in single quotes. A list of available classes can be selected from by clicking the corresponding dropdown button in the property sheet.

pos_absolute

The absolute position of the weblet on the web page. Note that 'Position Absolutely' must be selected from the weblet's right-click menu for this property to be used. The property will usually be set in pixels by dragging and dropping the weblet.

Default value

Blank (this is equivalent to the weblet being positioned relatively).

Valid values

Valid 'left' and 'top' coordinates, in valid units of measurement, in single quotes.

width

The width of the weblet on the web page. The weblet will reserve a minimum width based on the data to be displayed.

Usually you would set the width of the weblet by dragging the grab-handles around the weblet in the Design view of the LANSAs Editor. Doing so updates the value of the width property. However you can directly edit the property value if required.

Default value

Blank (this is equivalent to the weblet adopting its default width).

Valid values

A width, in a valid unit of measurement, in single quotes.

onchange_script

JavaScript code to be run when the input box loses focus after the text has been changed. JavaScript statements must be terminated by a semicolon.

Default value

Blank. No JavaScript is run.

Valid values

Any valid JavaScript statement(s).