NanoXLSX is a small .NET / C# library to create and read XLSX files (Microsoft Excel 2007 or newer) in an easy and native way

- **No dependencies** (*
  - No need for an installation of Microsoft Office
  - No need for Office interop libraries
  - No need for 3rd party libraries
  - No need for an installation of the Microsoft Open Office XML SDK (OOXML)

See the Change Log for recent updates.

### Namespaces

<table>
<thead>
<tr>
<th>Namespace</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NanoXLSX</td>
<td>NanoXLSX is a library to generate and read XLSX files in an easy and native way</td>
</tr>
<tr>
<td>NanoXLSX.Exception</td>
<td></td>
</tr>
<tr>
<td>NanoXLSX.LowLevel</td>
<td></td>
</tr>
<tr>
<td>NanoXLSX.Style</td>
<td></td>
</tr>
</tbody>
</table>

Copyright Raphael Stoeckli © 2018
NanoXLSX is a library to generate and read XLSX files in an easy and native way

**Classes**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell</td>
<td>Class representing a cell of a worksheet</td>
</tr>
<tr>
<td>CellBasicFormulas</td>
<td>Class for handling of basic Excel formulas</td>
</tr>
<tr>
<td>Metadata</td>
<td>Class representing the metadata of a workbook</td>
</tr>
<tr>
<td>Workbook</td>
<td>Class representing a workbook</td>
</tr>
<tr>
<td>WorkbookShortener</td>
<td>Class to provide access to the current worksheet with a shortened syntax. Note: The WS object can be null if the workbook was created without a worksheet. The object will be available as soon as the current worksheet is defined</td>
</tr>
<tr>
<td>Worksheet</td>
<td>Class representing a worksheet of a workbook</td>
</tr>
<tr>
<td>WorksheetColumn</td>
<td>Class representing a</td>
</tr>
</tbody>
</table>
column of a worksheet

## Structures

<table>
<thead>
<tr>
<th>Structure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CellAddress</td>
<td>Struct representing the cell address as column and row (zero based)</td>
</tr>
<tr>
<td>CellRange</td>
<td>Struct representing a cell range with a start and end address</td>
</tr>
</tbody>
</table>

## Enumerations

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CellAddressType</td>
<td>Enum for the referencing style of the address</td>
</tr>
<tr>
<td>CellCellType</td>
<td>Enum defines the basic data types of a cell</td>
</tr>
<tr>
<td>WorksheetCellDirection</td>
<td>Enum to define the direction when using AddNextCell method</td>
</tr>
<tr>
<td>WorksheetSheetProtectionValue</td>
<td>Enum to define the possible protection types when protecting a worksheet</td>
</tr>
</tbody>
</table>
# Cell Class

Class representing a cell of a worksheet

## Inheritance Hierarchy

- System\Object
- NanoXLSXCell

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```
public class Cell : IComparable<Cell>
```

The `Cell` type exposes the following members.

## Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell</td>
<td>Default constructor. Cells created with this constructor do not have a link to a worksheet initially</td>
</tr>
<tr>
<td>Cell(Object, CellCellType)</td>
<td>Constructor with value and cell type. Cells created with this constructor do not have a link to a worksheet initially</td>
</tr>
<tr>
<td>Cell(Object, CellCellType, String)</td>
<td>Constructor with value, cell type and address. The worksheet reference is set to null and must</td>
</tr>
</tbody>
</table>
be assigned later

Cell(Object, CellCellType, Int32, Int32, Worksheet) Constructor with value, cell type, row number, column number and the link to a worksheet

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CellAddress</td>
<td>Gets or sets the combined cell Address as string in the format A1 - XFD1048576</td>
</tr>
<tr>
<td>CellAddress2</td>
<td>Gets or sets the combined cell Address as Address object</td>
</tr>
<tr>
<td>CellStyle</td>
<td>Gets the assigned style of the cell</td>
</tr>
<tr>
<td>ColumnNumber</td>
<td>Gets or sets the number of the column (zero-based)</td>
</tr>
<tr>
<td>DataType</td>
<td>Gets or sets the type of the cell</td>
</tr>
<tr>
<td>RowNumber</td>
<td>Gets or sets the number of the row (zero-based)</td>
</tr>
<tr>
<td>Value</td>
<td>Gets or sets the value of the cell (generic object type)</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>WorksheetReference</td>
<td>Gets or sets the parent worksheet reference</td>
</tr>
<tr>
<td><strong>Top</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Methods</strong></td>
<td></td>
</tr>
<tr>
<td><strong>compareTo</strong></td>
<td>Implemented CompareTo method</td>
</tr>
<tr>
<td><strong>ConvertArray&lt;T&gt;</strong></td>
<td>Converts a List of supported objects into a list of cells</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations</td>
</tr>
<tr>
<td></td>
<td>before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetCellRange(String)</strong></td>
<td>Gets a list of cell addresses from a</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>GetCellRange(String, String)</code></td>
<td>Get a list of cell addresses from a cell range</td>
</tr>
<tr>
<td><code>GetCellRange(CellAddress, CellAddress)</code></td>
<td>Get a list of cell addresses from a cell range</td>
</tr>
<tr>
<td><code>GetCellRange(Int32, Int32, Int32, Int32)</code></td>
<td>Get a list of cell addresses from a cell range</td>
</tr>
<tr>
<td><code>GetHashCode</code></td>
<td>Serves as a hash function for a particular type. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>GetType</code></td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>RemoveStyle</code></td>
<td>Removes the assigned style from the cell</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ResolveCellAddress</td>
<td>Gets the address of a cell by the column and row number (zero based)</td>
</tr>
<tr>
<td>ResolveCellCoordinate(String)</td>
<td>Gets the column and row number (zero based) of a cell by the address</td>
</tr>
<tr>
<td>ResolveCellCoordinate(String, Int32, Int32)</td>
<td>Gets the column and row number (zero based) of a cell by the address</td>
</tr>
<tr>
<td>ResolveCellRange</td>
<td>Resolves a cell range from the format like A1:B3 or AAD556:AAD1000</td>
</tr>
<tr>
<td>ResolveCellType</td>
<td>Method resets the Cell type and tries to find the actual type. This is used if a Cell was created with the CellType DEFAULT. CellTypes FORMULA and EMPTY will skip this method</td>
</tr>
<tr>
<td>ResolveColumn</td>
<td>Gets the column number from the column address (A</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------------------------------------------------------</td>
</tr>
<tr>
<td>ResolveColumnAddress</td>
<td>Gets the column address (A - XFD)</td>
</tr>
<tr>
<td>SetCellLockedState</td>
<td>Sets the lock state of the cell</td>
</tr>
<tr>
<td>SetStyle</td>
<td>Sets the style of the cell</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
# Cell Constructor

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell</td>
<td>Default constructor. Cells created with this constructor do not have a link to a worksheet initially</td>
</tr>
<tr>
<td>Cell(Object, CellObject, String)</td>
<td>Constructor with value, cell type and address. The worksheet reference is set to null and must be assigned later</td>
</tr>
<tr>
<td>Cell(Object, CellObject, Int32, Int32, Worksheet)</td>
<td>Constructor with value, cell type, row number, column number and the link to a worksheet</td>
</tr>
</tbody>
</table>

## See Also

Reference
Cell Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Cell Constructor

Default constructor. Cells created with this constructor do not have a link to a worksheet initially

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public Cell()
```

**See Also**

Reference  
Cell Class  
Cell Overload  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Cell Constructor (Object, CellCellType)

Constructor with value and cell type. Cells created with this constructor do not have a link to a worksheet initially

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

C#

```csharp
public Cell(
    Object value,
    CellCellType type
)
```

Parameters

- **value**
  - Type: `System.Object`
  - Value of the cell

- **type**
  - Type: `NanoXLSXCellCellType`
  - Type of the cell

See Also

Reference
- Cell Class
- Cell Overload
- NanoXLSX Namespace
Cell Constructor (Object, CellCellType, String)

Constructor with value, cell type and address. The worksheet reference is set to null and must be assigned later.

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public Cell(
    Object value,
    CellCellType type,
    string address
)
```

### Parameters

**value**
- Type: `SystemObject`
  - Value of the cell

**type**
- Type: `NanoXLSXCellCellType`
  - Type of the cell

**address**
- Type: `SystemString`
  - Address of the cell

### See Also

Reference
| Cell Class | Cell Overload | NanoXLSX Namespace |

Copyright Raphael Stoeckli © 2018
Cell Constructor (Object, CellCellType, Int32, Int32, Worksheet)

Constructor with value, cell type, row number, column number and the link to a worksheet

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Cell(
    Object value,
    CellCellType type,
    int column,
    int row,
    Worksheet reference
)
```

### Parameters

**value**
- Type: SystemObject  
  Value of the cell

**type**
- Type: NanoXLSXCellCellType  
  Type of the cell

**column**
- Type: SystemInt32  
  Column number of the cell (zero-based)
**row**
Type: `SystemInt32`
Row number of the cell (zero-based)

**reference**
Type: `NanoXLSXWorksheet`
Referenced worksheet which contains the cell

See Also

Reference
Cell Class
Cell Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
# Cell Properties

The `Cell` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CellAddress</code></td>
<td>Gets or sets the combined cell Address as string in the format A1 - XFD1048576</td>
</tr>
<tr>
<td><code>CellAddress2</code></td>
<td>Gets or sets the combined cell Address as Address object</td>
</tr>
<tr>
<td><code>CellStyle</code></td>
<td>Gets the assigned style of the cell</td>
</tr>
<tr>
<td><code>ColumnNumber</code></td>
<td>Gets or sets the number of the column (zero-based)</td>
</tr>
<tr>
<td><code>DataType</code></td>
<td>Gets or sets the type of the cell</td>
</tr>
<tr>
<td><code>RowNumber</code></td>
<td>Gets or sets the number of the row (zero-based)</td>
</tr>
<tr>
<td><code>Value</code></td>
<td>Gets or sets the value of the cell (generic object type)</td>
</tr>
<tr>
<td><code>WorksheetReference</code></td>
<td>Gets or sets the parent worksheet reference</td>
</tr>
</tbody>
</table>
See Also

Reference
Cell Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellAddress Property

Gets or sets the combined cell Address as string in the format A1 - XFD1048576

**Namespace:** NanoXSX  
**Assembly:** NanoXSX (in NanoXSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```c#
public string CellAddress { get; set; }
```

**Property Value**
Type: **String**

## See Also

**Reference**
- Cell Class
- NanoXSX Namespace

Copyright Raphael Stoeckli © 2018
CellCellAddress2 Property

Gets or sets the combined cell Address as Address object

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public CellAddress CellAddress2 { get; set; }
```

Property Value

Type: CellAddress

### See Also

Reference

- Cell Class  
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellCellStyle Property

Gets the assigned style of the cell

**Namespace:** NanoLSX  
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleCellStyle { get; }
```

### Property Value

Type: **Style**

### See Also

**Reference**
- **Cell Class**
- **NanoLSX Namespace**

Copyright Raphael Stoeckli © 2018
CellColumnNumber Property

Gets or sets the number of the column (zero-based)

**Namespace:** NanoXLSX

**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public int ColumnNumber { get; set; }
```

### Property Value

Type: **Int32**

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws a RangeException if the column number is out of range</td>
</tr>
</tbody>
</table>

### See Also

**Reference**
- Cell Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellDataType Property

Gets or sets the type of the cell

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public CellCellType DataType { get; set; }
```

**Property Value**  
Type: `CellCellType`

### See Also

- **Reference**  
  - Cell Class  
  - NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellRowNumber Property

Gets or sets the number of the row (zero-based)

**Namespace:** NanoXLXS

**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public int RowNumber { get; set; }
```

### Property Value

Type: Int32

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws a RangeException if the row number is out of range</td>
</tr>
</tbody>
</table>

### See Also

**Reference**

- Cell Class
- NanoXLXS Namespace

Copyright Raphael Stoeckli © 2018
CellValue Property

Gets or sets the value of the cell (generic object type)

**Namespace:** NanoXLsx  
**Assembly:** NanoXLsx (in NanoXLsx.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public Object Value { get; set; }
```

### Property Value

Type: `Object`

## See Also

**Reference**  
Cell Class  
NanoXLsx Namespace

Copyright Raphael Stoeckli © 2018
CellWorksheetReference Property

Gets or sets the parent worksheet reference

**Namespace:** NanoLSX  
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Worksheet WorksheetReference { get; set; }
```

Property Value  
Type: Worksheet

### See Also

Reference  
Cell Class  
NanoLSX Namespace

Copyright Raphael Stoeckli © 2018
# Cell Methods

The **Cell** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CompareTo</strong></td>
<td>Implemented CompareTo method</td>
</tr>
<tr>
<td><strong>ConvertArrayT</strong></td>
<td>Converts a List of supported objects into a list of cells</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetCellRange(String)</td>
<td>Gets a list of cell addresses from a cell range (format A1:B3 or AAD556:AAD1000)</td>
</tr>
<tr>
<td>GetCellRange(String, String)</td>
<td>Get a list of cell addresses from a cell range</td>
</tr>
<tr>
<td>GetCellRange(CellAddress, CellAddress)</td>
<td>Get a list of cell addresses from a cell range</td>
</tr>
<tr>
<td>GetCellRange(Int32, Int32, Int32, Int32)</td>
<td>Get a list of cell addresses from a cell range</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type.</td>
</tr>
<tr>
<td>(Inherited from Object.)</td>
<td></td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>(Inherited from Object.)</td>
<td></td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td>(Inherited from Object.)</td>
<td></td>
</tr>
<tr>
<td>RemoveStyle</td>
<td>Removes the assigned style from</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ResolveCellAddress</td>
<td>Gets the address of a cell by the column and row number (zero based)</td>
</tr>
<tr>
<td>ResolveCellCoordinate(String)</td>
<td>Gets the column and row number (zero based) of a cell by the address</td>
</tr>
<tr>
<td>ResolveCellCoordinate(String, Int32, Int32)</td>
<td>Gets the column and row number (zero based) of a cell by the address</td>
</tr>
<tr>
<td>ResolveCellRange</td>
<td>Resolves a cell range from the format like A1:B3 or AAD556:AAD1000</td>
</tr>
<tr>
<td>ResolveCellType</td>
<td>Method resets the Cell type and tries to find the actual type. This is used if a Cell was created with the CellType DEFAULT. CellTypes FORMULA and EMPTY will skip this method</td>
</tr>
<tr>
<td>ResolveColumn</td>
<td>Gets the column</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResolveColumnAddress</td>
<td>Gets the column address (A - XFD)</td>
</tr>
<tr>
<td>SetCellLockedState</td>
<td>Sets the lock state of the cell</td>
</tr>
<tr>
<td>SetStyle</td>
<td>Sets the style of the cell</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

See Also

Reference
Cell Class
NanoXLSX Namespace
CellCompareTo Method

Implemented CompareTo method

**Namespace:** NanoXL SX
**Assembly:** NanoXL SX (in NanoXL SX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public int CompareTo(
    Cell other
)
```

### Parameters

*other*
Type: NanoXL SXCCell
Object to compare

### Return Value
Type: Int32
0 if values are the same, -1 if this object is smaller, 1 if it is bigger

### Implements
IComparableTCompareTo(T)

### See Also

**Reference**
Cell Class
NanoXL SX Namespace

Copyright Raphael Stoeckli © 2018
CellConvertArray\(T\) Method

Converts a List of supported objects into a list of cells

**Namespace:** NanoXL SX

**Assembly:** NanoXL SX (in NanoXL SX.dl l) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public static List<Cell> ConvertArray\(T\)(
    List\(<T>\) list
)
```

**Parameters**

*list*

- Type: `System.Collections.Generic.List\(T\)`
  - List of generic objects

**Type Parameters**

*T*

- Generic data type

**Return Value**

- Type: `ListCell`
  - List of cells

**See Also**

**Reference**

- Cell Class
  - NanoXL SX Namespace
CellGetCellRange Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCellRange(String)</td>
<td>Gets a list of cell addresses from a cell range (format A1:B3 or AAD556:AAD1000)</td>
</tr>
<tr>
<td>GetCellRange(String, String)</td>
<td>Get a list of cell addresses from a cell range</td>
</tr>
<tr>
<td>GetCellRange(CellAddress, CellAddress)</td>
<td>Get a list of cell addresses from a cell range</td>
</tr>
<tr>
<td>GetCellRange(Int32, Int32, Int32)</td>
<td>Get a list of cell addresses from a cell range</td>
</tr>
</tbody>
</table>

See Also

Reference
- Cell Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellGetCellRange Method (String)

Gets a list of cell addresses from a cell range (format A1:B3 or AAD556:AAD1000)

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public static List<CellAddress> GetCellRange(
    string range
)
```

**Parameters**

- **range**  
  Type: `System.String`  
  Range to process

**Return Value**

- Type: `List<CellAddress>`  
  List of cell addresses

**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FormatException</code></td>
<td>Throws a <code>FormatException</code> if a part of the passed range is malformed</td>
</tr>
<tr>
<td><code>RangeException</code></td>
<td>Throws an <code>RangeException</code> if the range is</td>
</tr>
</tbody>
</table>
out of range (A-XFD and 1 to 1048576)

See Also

Reference
Cell Class
GetCellRange Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellGetCellRange Method (String, String)

Get a list of cell addresses from a cell range

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static List<CellAddress> GetCellRange(
    string startAddress,
    string endAddress
)
```

### Parameters

**startAddress**  
Type: System.String  
Start address as string in the format A1 - XFD1048576

**endAddress**  
Type: System.String  
End address as string in the format A1 - XFD1048576

### Return Value

Type: List<CellAddress>  
List of cell addresses

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Throws a ArgumentException if a part of the start or end address is invalid</td>
</tr>
</tbody>
</table>
passed range is malformed

**RangeException**  Throws an RangeException if the range is out of range (A-XFD and 1 to 1048576)

### See Also

Reference
- **Cell Class**
- **GetCellRange Overload**
- **NanoXLSX Namespace**

Copyright Raphael Stoeckli © 2018
CellGetCellRange Method
(CellAddress, CellAddress)

Get a list of cell addresses from a cell range

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static List<CellAddress> GetCellRange(
    CellAddress startAddress,
    CellAddress endAddress
)
```

**Parameters**

- **startAddress**
  - Type: NanoXLSXCellAddress
  - Start address

- **endAddress**
  - Type: NanoXLSXCellAddress
  - End address

**Return Value**

- Type: List<CellAddress>
  - List of cell addresses

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Throws a ArgumentException if a part of the</td>
</tr>
</tbody>
</table>
passed addresses is malformed

**RangeException**  Throws an RangeException if the value of one passed address is out of range (A-XFD and 1 to 1048576)

## See Also

Reference

- Cell Class
- GetCellRange Overload
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellGetCellRange Method (Int32, Int32, Int32, Int32)

Get a list of cell addresses from a cell range

**Namespace:** NanoXL SX

**Assembly:** NanoXL SX (in NanoXL SX.dll) Version: 1.2.3.0 (1.2.3)

▲ Syntax

```csharp
public static List<CellAddress> CellGetCellRange(
    int startColumn,
    int startRow,
    int endColumn,
    int endRow
)
```

Parameters

**startColumn**
- Type: `SystemInt32`
- Start column (zero based)

**startRow**
- Type: `SystemInt32`
- Start row (zero based)

**endColumn**
- Type: `SystemInt32`
- End column (zero based)

**endRow**
- Type: `SystemInt32`
- End row (zero based)
Return Value
Type: ListCellAddress
List of cell addresses

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the value of one passed address parts is out of range (A-XFD and 1 to 1048576)</td>
</tr>
</tbody>
</table>

See Also

Reference
Cell Class
GetCellRange Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellRemoveStyle Method

Removes the assigned style from the cell

**Namespace:** NanoXLXS

**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void RemoveStyle()
```

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws a StyleException if the style cannot be referenced</td>
</tr>
</tbody>
</table>

### See Also

- Reference
  - Cell Class
  - NanoXLXS Namespace

Copyright Raphael Stoeckli © 2018
CellResolveCellAddress Method

Gets the address of a cell by the column and row number (zero based)

Namespace: NanoXL SX
Assembly: NanoXL SX (in NanoXL SX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public static string ResolveCellAddress(
    int column,
    int row,
    CellAddressType type = CellAddressType.Default
)
```

Parameters

*column*
  - Type: SystemInt32
  - Column number of the cell (zero-based)

*row*
  - Type: SystemInt32
  - Row number of the cell (zero-based)

*type (Optional)*
  - Type: NanoXL SXCellAddressType
  - Optional referencing type of the address

Return Value

Type: String
Cell Address as string in the format A1 - XFD1048576. Depending on the type, Addresses like '$A55', 'B$2' or '$A$5' are possible outputs
# Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RangeException</code></td>
<td>Throws an <code>RangeException</code> if the start or end address was out of range</td>
</tr>
</tbody>
</table>

## See Also

**Reference**
- Cell Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
# CellResolveCellCoordinate Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ 心 ] ResolveCellCoordinate(String)</td>
<td>Gets the column and row number (zero based) of a cell by the address</td>
</tr>
<tr>
<td>![ 心 ] ResolveCellCoordinate(String, Int32, Int32)</td>
<td>Gets the column and row number (zero based) of a cell by the address</td>
</tr>
</tbody>
</table>

## See Also

- Reference
  - Cell Class
  - NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellResolveCellCoordinate Method (String)

Gets the column and row number (zero based) of a cell by the address

**Namespace:** NanoLSX  
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public static CellAddress ResolveCellCoordinate(
    string address
)
```

**Parameters**

`address`
- Type: `System.String`  
  Address as string in the format A1 - XFD1048576

**Return Value**

Type: `CellAddress`  
Struct with row and column

**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormatException</td>
<td>Throws a FormatException if the passed address is malformed</td>
</tr>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the value of the passed address is out of range (A-XFD</td>
</tr>
</tbody>
</table>
See Also

Reference
Cell Class
ResolveCellCoordinate Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellResolveCellCoordinate Method (String, Int32, Int32)

Gets the column and row number (zero based) of a cell by the address

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public static void ResolveCellCoordinate(
    string address,
    out int column,
    out int row
)
```

### Parameters

- **address**
  - Type: **System.String**
  - Address as string in the format A1 - XFD1048576

- **column**
  - Type: **System.Int32**
  - Column number of the cell (zero-based) as out parameter

- **row**
  - Type: **System.Int32**
  - Row number of the cell (zero-based) as out parameter

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FormatException | Throws a FormatException if the range address was malformed
---|---
RangeException | Throws an RangeException if the row or column number was out of range

See Also

Reference
Cell Class
ResolveCellCoordinate Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellResolveCellRange Method

Resolves a cell range from the format like A1:B3 or AAD556:AAD1000

**Namespace:**  NanoXLXS
**Assembly:**  NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public static CellRange ResolveCellRange(
        string range
    )
```

### Parameters

**range**  
Type:  `System.String`  
Range to process

### Return Value

Type:  `CellRange`  
Range object

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FormatException</code></td>
<td>Throws a FormatException if the start or end address was malformed</td>
</tr>
<tr>
<td><code>RangeException</code></td>
<td>Throws an RangeException if the range is out of range (A-XFD and 1 to 1048576)</td>
</tr>
</tbody>
</table>
See Also

Reference
Cell Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellResolveCellType Method

Method resets the Cell type and tries to find the actual type. This is used if a Cell was created with the CellType DEFAULT. CellTypes FORMULA and EMPTY will skip this method.

**Namespace:** NanoXL SX

**Assembly:** NanoXL SX (in NanoXL SX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void ResolveCellType()
```

### See Also

Reference
- Cell Class
- NanoXL SX Namespace

Copyright Raphael Stoeckli © 2018
CellResolveColumn Method

Gets the column number from the column address (A - XFD)

**Namespace:** NanoXLX

**Assembly:** NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static int ResolveColumn(
    string columnAddress
)
```

### Parameters

- **columnAddress**
  
  Type: System.String
  
  Column address (A - XFD)

### Return Value

Type: Int32

Column number (zero-based)

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Throws an ArgumentException if the passed address was out of range</td>
</tr>
</tbody>
</table>

### See Also

Reference
CellResolveColumnAddress Method

Gets the column address (A - XFD)

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```
public static string ResolveColumnAddress(
    int columnNumber
)
```

Parameters

`columnNumber`
- Type: `System.Int32`
- Column number (zero-based)

Return Value

Type: `String`
- Column address (A - XFD)

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RangeException</code></td>
<td>Throws an RangeException if the passed column number was out of range</td>
</tr>
</tbody>
</table>

See Also
Reference
Cell Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellSetCellLockedState Method

Sets the lock state of the cell

**Namespace:** NanoXL SX  
**Assembly:** NanoXL SX (in NanoXL SX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public void SetCellLockedState(
    bool isLocked,
    bool isHidden
)
```

**Parameters**

- **isLocked**  
  Type: `System.Boolean`  
  If true, the cell will be locked if the worksheet is protected

- **isHidden**  
  Type: `System.Boolean`  
  If true, the value of the cell will be invisible if the worksheet is protected

**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws an UndefinedStyleException if the style used to lock cells cannot be referenced</td>
</tr>
</tbody>
</table>

**Remarks**
The listed exception should never happen because the mentioned style is internally generated

See Also

Reference
Cell Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellSetStyle Method

Sets the style of the cell

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public Style SetStyle(
    Style style
)
```

### Parameters

**style**
- Type: NanoXLSX.StyleStyle
  - Style to assign

### Return Value

- Type: Style
- If the passed style already exists in the workbook, the existing one will be returned, otherwise the passed one

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws an StyleException if the style cannot be referenced or no style was defined</td>
</tr>
</tbody>
</table>

### See Also
Reference

Cell Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellAddress Structure

Struct representing the cell address as column and row (zero based)

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public struct Address
```

The **CellAddress** type exposes the following members.

### Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CellAddress(String, CellAddressType)</td>
<td>Constructor with address as string</td>
</tr>
<tr>
<td>CellAddress(Int32, Int32, CellAddressType)</td>
<td>Constructor with row and column as arguments</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals(Object)</td>
<td>Indicates whether this instance and a specified object are equal.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Equals(CellAddress)</td>
<td>Compares two addresses whether they are equal</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetAddress</td>
<td>Returns the combined Address</td>
</tr>
<tr>
<td>GetColumn</td>
<td>Gets the column address (A - XFD)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns the hash code for this instance. (Inherited from ValueType.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Overwritten ToString method (Overrides ValueType.ToString.)</td>
</tr>
</tbody>
</table>

**Top**

**Fields**
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
<td>Column number (zero based)</td>
</tr>
<tr>
<td>Row</td>
<td>Row number (zero based)</td>
</tr>
<tr>
<td>Type</td>
<td>Referencing type of the address</td>
</tr>
</tbody>
</table>

See Also

Reference

NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Address Constructor

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CellAddress(String, CellAddressType)</code></td>
<td>Constructor with address as string</td>
</tr>
<tr>
<td><code>CellAddress(Int32, Int32, CellAddressType)</code></td>
<td>Constructor with row and column as arguments</td>
</tr>
</tbody>
</table>

See Also

Reference

CellAddress Structure
NanoXLSX Namespace
CellAddress Constructor (String, CellAddressType)

Constructor with address as string

Namespace: NanoLSX
Assembly: NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

▶ Syntax

```
public Address(
    string address,
    CellAddressType type = CellAddressType.Default)
```

Parameters

address

Type: System.String
Address string (e.g. 'A1:B12')

type (Optional)

Type: NanoLSX.CellAddressType
Optional referencing type of the address

▶ See Also

Reference
CellAddress Structure
CellAddress Overload
NanoLSX Namespace

Copyright Raphael Stoeckli © 2018
NanoXL SX Library Documentation
CellAddress Constructor (Int32, Int32, CellAddressType)

Constructor with row and column as arguments

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Address(
    int column,
    int row,
    CellAddressType type = CellAddressType.Default
)
```

### Parameters

- **column**
  - Type: `System.Int32`
  - Column number (zero based)

- **row**
  - Type: `System.Int32`
  - Row number (zero based)

- **type** *(Optional)*
  - Type: `NanoXLSXCellAddressType`
  - Optional referencing type of the address

### See Also

Reference  
*CellAddress Structure*
CellAddress Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
## Address Methods

The **CellAddress** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals(Object)</td>
<td>Indicates whether this instance and a specified object are equal. <em>(Inherited from <strong>ValueType</strong>.)</em></td>
</tr>
<tr>
<td>Equals(CellAddress)</td>
<td>Compares two addresses whether they are equal</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. <em>(Inherited from <strong>Object</strong>.)</em></td>
</tr>
<tr>
<td>GetAddress</td>
<td>Returns the combined Address</td>
</tr>
<tr>
<td>GetColumn</td>
<td>Gets the column address <em>(A - XFD)</em></td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns the hash code for this instance. <em>(Inherited from <strong>ValueType</strong>.)</em></td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Overwritten <code>ToString</code> method (Overrides <code>ValueType.ToString</code>).</td>
</tr>
</tbody>
</table>

**See Also**

Reference
- CellAddress Structure
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
AddressEquals Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals(Object)</td>
<td>Indicates whether this instance and a specified object are equal.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from ValueType.)</td>
</tr>
<tr>
<td>Equals(CellAddress)</td>
<td>Compares two addresses whether they are equal</td>
</tr>
</tbody>
</table>

See Also

Reference
CellAddress Structure
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellAddressEquals Method (CellAddress)

Compares two addresses whether they are equal

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public bool Equals(
    CellAddress o
)
```

**Parameters**

- `o`  
  Type: NanoXLSX.CellAddress  
  Other address

**Return Value**

Type: Boolean  
True if equal

**See Also**

Reference  
CellAddress Structure  
Equals Overload  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellAddress

GetAddress Method

Returns the combined Address

**Namespace:** NanoXLSX

**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string GetAddress()
```

### Return Value

Type: **String**
Address as string in the format A1 - XFD1048576

### See Also

Reference

- CellAddress Structure
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellAddressGetMethod

Gets the column address (A - XFD)

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string GetColumn()
```

### Return Value

**Type:** String  
Column address as letter(s)

### See Also

**Reference**  
CellAddress Structure  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellAddressToString Method

Overwritten ToString method

**Namespace:** NanoLSX  
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public override string ToString()
```

### Return Value

**Type:** String  
Returns the cell address (e.g. 'A15')

### See Also

**Reference**  
CellAddress Structure  
NanoLSX Namespace

Copyright Raphael Stoeckli © 2018
# Address Fields

The `CellAddress` type exposes the following members.  

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
<td>Column number (zero based)</td>
</tr>
<tr>
<td>Row</td>
<td>Row number (zero based)</td>
</tr>
<tr>
<td>Type</td>
<td>Referencing type of the address</td>
</tr>
</tbody>
</table>

## See Also

Reference  
- `CellAddress Structure`  
- NanoXLSX Namespace  

Copyright Raphael Stoeckli © 2018
CellAddressColumn Field

Column number (zero based)

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

⚠️ **Syntax**

```
public int Column
```

Field Value  
Type: **Int32**

⚠️ **See Also**

- Reference
  - CellAddress Structure
  - NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellAddressRow Field

Row number (zero based)

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public int Row
```

Field Value
Type: Int32

See Also

Reference
CellAddress Structure
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellAddressType Field

Referencing type of the address

**Namespace:** NanoLSX  
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public CellAddressType Type
```

### Field Value

Type: `CellAddressType`

### See Also

- Reference
  - `CellAddress Structure`
  - `NanoLSX Namespace`

Copyright Raphael Stoeckli © 2018
CellAddressType Enumeration

Enum for the referencing style of the address

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```
public enum AddressType
```

Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>0</td>
<td>Default behavior (e.g. 'C3')</td>
</tr>
<tr>
<td>FixedRow</td>
<td>1</td>
<td>Row of the address is fixed (e.g. 'C$3')</td>
</tr>
<tr>
<td>FixedColumn</td>
<td>2</td>
<td>Column of the address is fixed (e.g. '$C3')</td>
</tr>
<tr>
<td>FixedRowAndColumn</td>
<td>3</td>
<td>Row and column of the address is fixed (e.g. '$C$3')</td>
</tr>
</tbody>
</table>

See Also
Reference
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulas Class

Class for handling of basic Excel formulas

Inheritance Hierarchy

System\Object NanoXLSXCellBasicFormulas

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

C# public static class BasicFormulas

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![S] Average(CellRange)</td>
<td>Returns a cell with a average formula</td>
</tr>
<tr>
<td>![S] Average(Worksheet, CellRange)</td>
<td>Returns a cell with a average formula</td>
</tr>
<tr>
<td>![S] Ceil(CellAddress, Int32)</td>
<td>Returns a cell with a ceil formula</td>
</tr>
<tr>
<td>![S] Ceil(Worksheet, CellAddress, Int32)</td>
<td>Returns a cell with a ceil formula</td>
</tr>
<tr>
<td>![S] Floor(CellAddress, Int32)</td>
<td>Returns a cell with a floor formula</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Floor(Worksheet,</td>
<td>Returns a cell with a floor formula</td>
</tr>
<tr>
<td>CellAddress, Int32)</td>
<td></td>
</tr>
<tr>
<td>GetBasicFormula</td>
<td>Function to generate a basic Excel function with one cell range as parameter</td>
</tr>
<tr>
<td></td>
<td>and an optional post argument</td>
</tr>
<tr>
<td>GetVLookup</td>
<td>Function to generate a Vlookup as Excel function</td>
</tr>
<tr>
<td>Max(CellRange)</td>
<td>Returns a cell with a max formula</td>
</tr>
<tr>
<td>Max(Worksheet, CellRange)</td>
<td>Returns a cell with a max formula</td>
</tr>
<tr>
<td>Median(CellRange)</td>
<td>Returns a cell with a median formula</td>
</tr>
<tr>
<td>Median(Worksheet,</td>
<td>Returns a cell with a median formula</td>
</tr>
<tr>
<td>CellRange)</td>
<td></td>
</tr>
<tr>
<td>Min(CellRange)</td>
<td>Returns a cell with a min formula</td>
</tr>
<tr>
<td>Min(Worksheet, CellRange)</td>
<td>Returns a cell with a min formula</td>
</tr>
<tr>
<td>Round(CellAddress,</td>
<td>Returns a cell with a round formula</td>
</tr>
<tr>
<td>Int32)</td>
<td></td>
</tr>
<tr>
<td>Round(Worksheet,</td>
<td>Returns a cell with a round formula</td>
</tr>
<tr>
<td>CellAddress, Int32)</td>
<td></td>
</tr>
<tr>
<td>Sum(CellRange)</td>
<td>Returns a cell with a sum formula</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>Sum(Worksheet, CellRange)</code></td>
<td>Returns a cell with a sum formula</td>
</tr>
<tr>
<td><code>VLookup(Object, CellRange, Int32, Boolean)</code></td>
<td>Function to generate a Vlookup as Excel function</td>
</tr>
<tr>
<td><code>VLookup(CellAddress, CellRange, Int32, Boolean)</code></td>
<td>Function to generate a Vlookup as Excel function</td>
</tr>
<tr>
<td><code>VLookup(Object, Worksheet, CellRange, Int32, Boolean)</code></td>
<td>Function to generate a Vlookup as Excel function</td>
</tr>
<tr>
<td><code>VLookup(Worksheet, CellAddress, Worksheet, CellRange, Int32, Boolean)</code></td>
<td>Function to generate a Vlookup as Excel function</td>
</tr>
</tbody>
</table>

**See Also**

- Reference
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
## BasicFormulas Methods

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average(CellRange)</td>
<td>Returns a cell with a average formula</td>
</tr>
<tr>
<td>Average(Worksheet, CellRange)</td>
<td>Returns a cell with a average formula</td>
</tr>
<tr>
<td>Ceil(CellAddress, Int32)</td>
<td>Returns a cell with a ceil formula</td>
</tr>
<tr>
<td>Ceil(Worksheet, CellAddress, Int32)</td>
<td>Returns a cell with a ceil formula</td>
</tr>
<tr>
<td>Floor(CellAddress, Int32)</td>
<td>Returns a cell with a floor formula</td>
</tr>
<tr>
<td>Floor(Worksheet, CellAddress, Int32)</td>
<td>Returns a cell with a floor formula</td>
</tr>
<tr>
<td>GetBasicFormula</td>
<td>Function to generate a basic Excel function with one cell range as parameter and an optional post argument</td>
</tr>
<tr>
<td>GetVLookup</td>
<td>Function to generate a Vlookup as Excel function</td>
</tr>
<tr>
<td>Max(CellRange)</td>
<td>Returns a cell with a max formula</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Max(Worksheet, CellRange)</td>
<td>Returns a cell with a max formula</td>
</tr>
<tr>
<td>Median(CellRange)</td>
<td>Returns a cell with a median formula</td>
</tr>
<tr>
<td>Median(Worksheet, CellRange)</td>
<td>Returns a cell with a median formula</td>
</tr>
<tr>
<td>Min(CellRange)</td>
<td>Returns a cell with a min formula</td>
</tr>
<tr>
<td>Min(Worksheet, CellRange)</td>
<td>Returns a cell with a min formula</td>
</tr>
<tr>
<td>Round(CellAddress, Int32)</td>
<td>Returns a cell with a round formula</td>
</tr>
<tr>
<td>Round(Worksheet, CellAddress, Int32)</td>
<td>Returns a cell with a round formula</td>
</tr>
<tr>
<td>Sum(CellRange)</td>
<td>Returns a cell with a sum formula</td>
</tr>
<tr>
<td>Sum(Worksheet, CellRange)</td>
<td>Returns a cell with a sum formula</td>
</tr>
<tr>
<td>VLookup(Object, CellRange, Int32, Boolean)</td>
<td>Function to generate a Vlookup as Excel function</td>
</tr>
<tr>
<td>VLookup(CellAddress, CellRange, Int32, Boolean)</td>
<td>Function to generate a Vlookup as Excel function</td>
</tr>
<tr>
<td>VLookup(Object, Worksheet, CellRange, Int32,</td>
<td>Function to generate a Vlookup as Excel function</td>
</tr>
</tbody>
</table>
**VLookup(Worksheet, CellAddress, Worksheet, CellRange, Int32, Boolean)**

Function to generate a Vlookup as Excel function

---

**Top**

**See Also**

Reference

- CellBasicFormulas Class
- NanoXLSX Namespace

---

Copyright Raphael Stoeckli © 2018
BasicFormulas Average Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average(CellRange)</td>
<td>Returns a cell with a average formula</td>
</tr>
<tr>
<td>Average(Worksheet, CellRange)</td>
<td>Returns a cell with a average formula</td>
</tr>
</tbody>
</table>

See Also

Reference
CellBasicFormulas Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulasAverage Method (CellRange)

Returns a cell with a average formula

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

▶ Syntax

```csharp
public static Cell Average(
    CellRange range
)
```

**Parameters**

*range*  
Type: NanoXLSXCellRange  
Cell range to apply the average operation to

**Return Value**

Type: Cell  
Prepared Cell object, ready to be added to a worksheet

▶ See Also

Reference  
CellBasicFormulas Class  
Average Overload  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulasAverage
Method (Worksheet, CellRange)

Returns a cell with a average formula

**Namespace:** NanoXLSX
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Cell Average(
    Worksheet target,
    CellRange range
)
```

### Parameters

*target*
- Type: NanoXLSXWorksheet
- Target worksheet of the average operation. Can be null if on the same worksheet

*range*
- Type: NanoXLSXCellRange
- Cell range to apply the average operation to

### Return Value

Type: Cell
- Prepared Cell object, ready to be added to a worksheet

### See Also

Reference
BasicFormulas

Ceil Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ceil(CellAddress, Int32)</td>
<td>Returns a cell with a ceil formula</td>
</tr>
<tr>
<td>Ceil(Worksheet, CellAddress, Int32)</td>
<td>Returns a cell with a ceil formula</td>
</tr>
</tbody>
</table>

See Also

Reference
CellBasicFormulas Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
CellBasicFormulas.Ceil Method (CellAddress, Int32)

Returns a cell with a ceil formula

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public static Cell Ceil(
    CellAddress address,
    int decimals
)
```

### Parameters

- **address**  
  Type: NanoXLSX.CellAddress  
  Address to apply the ceil operation to

- **decimals**  
  Type: System.Int32  
  Number of decimals (digits)

### Return Value

Type: Cell  
Prepared Cell object, ready to be added to a worksheet

## See Also

Reference  
CellBasicFormulas Class
Ceil Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulasCeil Method (Worksheet, CellAddress, Int32)

Returns a cell with a ceil formula

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Cell Ceil(
    Worksheet target,
    CellAddress address,
    int decimals
)
```

### Parameters

- **target**
  - Type: NanoXLSXWorksheet
  - Target worksheet of the ceil operation. Can be null if on the same worksheet

- **address**
  - Type: NanoXLSXCellAddress
  - Address to apply the ceil operation to

- **decimals**
  - Type: SystemInt32
  - Number of decimals (digits)

### Return Value

- Type: Cell
  - Prepared Cell object, ready to be added to a worksheet
See Also

Reference

CellBasicFormulas Class
Ceil Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
BasicFormulas.Floor Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floor(CellAddress, Int32)</td>
<td>Returns a cell with a floor formula</td>
</tr>
<tr>
<td>Floor(Worksheet, CellAddress, Int32)</td>
<td>Returns a cell with a floor formula</td>
</tr>
</tbody>
</table>

See Also

Reference
CellBasicFormulas Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulas.Floor Method (CellAddress, Int32)

Returns a cell with a floor formula

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Cell Floor(
    CellAddress address,
    int decimals
)
```

### Parameters

- **address**
  - Type: NanoXLSXCellAddress
  - Address to apply the floor operation to

- **decimals**
  - Type: SystemInt32
  - Number of decimals (digits)

### Return Value

Type: Cell
- Prepared Cell object, ready to be added to a worksheet

### See Also

- Reference  
  CellBasicFormulas Class
CellBasicFormulasFloor Method
(Worksheet, CellAddress, Int32)

Returns a cell with a floor formula

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public static Cell Floor(
    Worksheet target,
    CellAddress address,
    int decimals
)
```

**Parameters**

*target*  
Type: NanoXLSXWorksheet  
Target worksheet of the floor operation. Can be null if on the same worksheet

*address*  
Type: NanoXLSXCellAddress  
Address to apply the floor operation to

*decimals*  
Type: SystemInt32  
Number of decimals (digits)

**Return Value**  
Type: Cell  
Prepared Cell object, ready to be added to a worksheet
See Also

Reference
CellBasicFormulas Class
Floor Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulas.GetBasicFormula Method

Function to generate a basic Excel function with one cell range as parameter and an optional post argument

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
private static Cell GetBasicFormula(
    Worksheet target,  
    CellRange range,  
    string functionName,  
    string postArg
)
```

### Parameters

**target**  
Type: NanoXLSX.Worksheet  
Target worksheet of the cell reference. Can be null if on the same worksheet

**range**  
Type: NanoXLSX.CellRange  
Main argument as cell range. If applied on one cell, the start and end address are identical

**functionName**  
Type: System.String  
Internal Excel function name

**postArg**
**Type:** SystemString
Optional argument

**Return Value**
Type: Cell
Prepared Cell object, ready to be added to a worksheet

See Also

**Reference**
CellBasicFormulas Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulas.GetVLookup Method

Function to generate a Vlookup as Excel function

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private static Cell GetVLookup(
    Worksheet queryTarget,
    CellAddress address,
    Object number,
    Worksheet rangeTarget,
    CellRange range,
    int columnIndex,
    bool exactMatch,
    bool numericLookup
)
```

### Parameters

**queryTarget**
- Type: NanoXLSXWorksheet  
  Target worksheet of the query argument. Can be null if on the same worksheet

**address**
- Type: NanoXLSXCellAddress  
  In case of a reference lookup, query address of a cell as string

**number**
- Type: SystemObject  
  In case of a numeric lookup, number for the lookup
**rangeTarget**
Type: NanoXLSXWorksheet
Target worksheet of the matrix. Can be null if on the same worksheet

**range**
Type: NanoXLSXCellRange
Matrix of the lookup

**columnIndex**
Type: SystemInt32
Column index of the target column (1 based)

**exactMatch**
Type: SystemBoolean
If true, an exact match is applied to the lookup

**numericLookup**
Type: SystemBoolean
If true, the lookup is a numeric lookup, otherwise a reference lookup

**Return Value**
Type: Cell
Prepared Cell object, ready to be added to a worksheet

## See Also

**Reference**
CellBasicFormulas Class
NanoXLSX Namespace
BasicFormulas Max Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![S] Max(CellRange)</td>
<td>Returns a cell with a max formula</td>
</tr>
<tr>
<td>![S] Max(Worksheet, CellRange)</td>
<td>Returns a cell with a max formula</td>
</tr>
</tbody>
</table>

See Also

Reference
CellBasicFormulas Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulasMax Method (CellRange)

Returns a cell with a max formula

**Namespace:** NanoLSX  
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Cell Max(
    CellRange range
)
```

### Parameters

`range`
- Type: NanoLSXCellRange
  - Cell range to apply the max operation to

### Return Value

Type: Cell  
- Prepared Cell object, ready to be added to a worksheet

### See Also

Reference
- CellBasicFormulas Class  
- Max Overload  
- NanoLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulasMax Method (Worksheet, CellRange)

Returns a cell with a max formula

**Namespace:** NanoXLSX
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Cell Max(
    Worksheet target,
    CellRange range
)
```

### Parameters

- **target**
  - Type: NanoXLSXWorksheet
  - Target worksheet of the max operation. Can be null if on the same worksheet

- **range**
  - Type: NanoXLSXCellRange
  - Cell range to apply the max operation to

### Return Value

- Type: Cell
  - Prepared Cell object, ready to be added to a worksheet

### See Also

Reference
BasicFormulas Median Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Median(CellRange)</td>
<td>Returns a cell with a median formula</td>
</tr>
<tr>
<td>Median(Worksheet, CellRange)</td>
<td>Returns a cell with a median formula</td>
</tr>
</tbody>
</table>

See Also

Reference
CellBasicFormulas Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulasMedian Method (CellRange)

Returns a cell with a median formula

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public static Cell Median(
    CellRange range
)
```

**Parameters**

*range*  
Type: NanoXLSXCellRange  
Cell range to apply the median operation to

**Return Value**

Type: Cell  
Prepared Cell object, ready to be added to a worksheet

**See Also**

Reference  
CellBasicFormulas Class  
Median Overload  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulasMedian Method (Worksheet, CellRange)

Returns a cell with a median formula

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
public static Cell Median(
    Worksheet target,
    CellRange range
)
```

Parameters

target
Type: NanoXLSXWorksheet
Target worksheet of the median operation. Can be null if on the same worksheet

range
Type: NanoXLSXCellRange
Cell range to apply the median operation to

Return Value
Type: Cell
Prepared Cell object, ready to be added to a worksheet

See Also

Reference
BasicFormulas Min Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min(CellRange)</td>
<td>Returns a cell with a min formula</td>
</tr>
<tr>
<td>Min(Worksheet, CellRange)</td>
<td>Returns a cell with a min formula</td>
</tr>
</tbody>
</table>

See Also

Reference
CellBasicFormulas Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulas.Min Method (CellRange)

Returns a cell with a min formula

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Cell Min(
    CellRange range
)
```

### Parameters

- **range**  
  Type: NanoXLSX.CellRange  
  Cell range to apply the min operation to

### Return Value

- Type: Cell  
  Prepared Cell object, ready to be added to a worksheet

### See Also

- Reference  
  CellBasicFormulas Class  
  Min Overload  
  NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulas.Min Method (Worksheet, CellRange)

Returns a cell with a min formula

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public static Cell Min(
    Worksheet target,
    CellRange range
)
```

## Parameters

**target**
- Type: NanoXLSX.Worksheet
  - Target worksheet of the min operation. Can be null if on the same worksheet

**range**
- Type: NanoXLSX.CellRange
  - Cell range to apply the median operation to

## Return Value
- Type: Cell
  - Prepared Cell object, ready to be added to a worksheet

## See Also

Reference
CellBasicFormulas Class
Min Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
BasicFormulas.Round Method

▲ Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round(CellAddress, Int32)</td>
<td>Returns a cell with a round formula</td>
</tr>
<tr>
<td>Round(Worksheet, CellAddress, Int32)</td>
<td>Returns a cell with a round formula</td>
</tr>
</tbody>
</table>

See Also

Reference

CellBasicFormulas Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulas.Round Method (CellAddress, Int32)

Returns a cell with a round formula

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public static Cell Round(
    CellAddress address,
    int decimals
)
```

Parameters

- **address**
  - Type: NanoXLSX.CellAddress
  - Address to apply the round operation to

- **decimals**
  - Type: System.Int32
  - Number of decimals (digits)

Return Value

- Type: Cell
  - Prepared Cell object, ready to be added to a worksheet

See Also

- Reference
  - CellBasicFormulas Class
CellBasicFormulasRound Method (Worksheet, CellAddress, Int32)

Returns a cell with a round formula

**Namespace:** NanoLSX
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Cell Round(
    Worksheet target,
    CellAddress address,
    int decimals
)
```

### Parameters

**target**
- Type: `NanoLSX.Worksheet`
- Target worksheet of the round operation. Can be null if on the same worksheet.

**address**
- Type: `NanoLSX.CellAddress`
- Address to apply the round operation to

**decimals**
- Type: `System.Int32`
- Number of decimals (digits)

### Return Value
Type: **Cell**
Prepared Cell object, ready to be added to a worksheet

**See Also**

**Reference**
- CellBasicFormulas Class
- Round Overload
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
BasicFormulasSum Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Σ] Sum(CellRange)</td>
<td>Returns a cell with a sum formula</td>
</tr>
<tr>
<td>![Σ] Sum(Worksheet, CellRange)</td>
<td>Returns a cell with a sum formula</td>
</tr>
</tbody>
</table>

See Also

Reference
CellBasicFormulas Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulas.Sum Method (CellRange)

Returns a cell with a sum formula

**Namespace:** NanoLSX
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public static Cell Sum(
    CellRange range
)
```

**Parameters**

*range*
- Type: NanoLSX<CellRange>
- Cell range to get a sum of

**Return Value**

- Type: Cell
- Prepared Cell object, ready to be added to a worksheet

**See Also**

- Reference
  - CellBasicFormulas Class
  - Sum Overload
  - NanoLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulasSum Method (Worksheet, CellRange)

Returns a cell with a sum formula

**Namespace:** NanoXLSX

**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Cell Sum(
    Worksheet target,
    CellRange range
)
```

### Parameters

**target**
- Type: NanoXLSXWorksheet
- Target worksheet of the sum operation. Can be null if on the same worksheet

**range**
- Type: NanoXLSXCellRange
- Cell range to get a sum of

### Return Value

- Type: Cell
- Prepared Cell object, ready to be added to a worksheet

### See Also

Reference
BasicFormulas

VLookup Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLookup(Object, CellRange, Int32, Boolean)</td>
<td>Function to generate a Vlookup as Excel function</td>
</tr>
<tr>
<td>VLookup(CellAddress, CellRange, Int32, Boolean)</td>
<td>Function to generate a Vlookup as Excel function</td>
</tr>
<tr>
<td>VLookup(Object, Worksheet, CellRange, Int32, Boolean)</td>
<td>Function to generate a Vlookup as Excel function</td>
</tr>
<tr>
<td>VLookup(Worksheet, CellAddress, Worksheet, CellRange, Int32, Boolean)</td>
<td>Function to generate a Vlookup as Excel function</td>
</tr>
</tbody>
</table>

See Also

Reference
- CellBasicFormulas Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulas.VLookup Method (Object, CellRange, Int32, Boolean)

Function to generate a Vlookup as Excel function

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Cell VLookup(
    Object number,
    CellRange range,
    int columnIndex,
    bool exactMatch
)
```

### Parameters

- **number**  
  Type: SystemObject  
  Numeric value for the lookup. Valid types are int, long, float and double

- **range**  
  Type: NanoXLSX.CellRange  
  Matrix of the lookup

- **columnIndex**  
  Type: System.Int32  
  Column index of the target column (1 based)

- **exactMatch**
Type: SystemBoolean
If true, an exact match is applied to the lookup

Return Value
Type: Cell
Prepared Cell object, ready to be added to a worksheet

See Also

Reference
CellBasicFormulas Class
VLookup Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulasVLookup Method (CellAddress, CellRange, Int32, Boolean)

Function to generate a Vlookup as Excel function

**Namespace:** NanoLSXX
**Assembly:** NanoLSX (in NanoLSXx.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Cell VLookup(
    CellAddress address,
    CellRange range,
    int columnIndex,
    bool exactMatch
)
```

### Parameters

- **address**
  - Type: **NanoLSX.CellAddress**
  - Query address of a cell as string as source of the lookup
- **range**
  - Type: **NanoLSX.CellRange**
  - Matrix of the lookup
- **columnIndex**
  - Type: **System.Int32**
  - Column index of the target column (1 based)
- **exactMatch**
  - Type: **System.Boolean**
If true, an exact match is applied to the lookup

Return Value
Type: Cell
Prepared Cell object, ready to be added to a worksheet

See Also

Reference
CellBasicFormulas Class
VLookup Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellBasicFormulasVLookup Method (Object, Worksheet, CellRange, Int32, Boolean)

Function to generate a Vlookup as Excel function

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public static Cell VLookup(
    Object number,
    Worksheet rangeTarget,
    CellRange range,
    int columnIndex,
    bool exactMatch
)
```

**Parameters**

- **number**
  - Type: SystemObject
  - Numeric value for the lookup. Valid types are int, long, float and double

- **rangeTarget**
  - Type: NanoXLSXWorksheet
  - Target worksheet of the matrix. Can be null if on the same worksheet

- **range**
  - Type: NanoXLSXCellRange
Matrix of the lookup

\textit{columnIndex}
Type: \texttt{SystemInt32}
Column index of the target column (1 based)

\textit{exactMatch}
Type: \texttt{SystemBoolean}
If true, an exact match is applied to the lookup

Return Value
Type: \texttt{Cell}
Prepared Cell object, ready to be added to a worksheet

\textbf{See Also}

Reference
\texttt{CellBasicFormulas Class}
\texttt{VLookup Overload}
\texttt{NanoXLSX Namespace}

Copyright Raphael Stoeckli © 2018
CellBasicFormulasVLookup Method (Worksheet, CellAddress, Worksheet, CellRange, Int32, Boolean)

Function to generate a Vlookup as Excel function

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public static Cell VLookup(
    Worksheet queryTarget,
    CellAddress address,
    Worksheet rangeTarget,
    CellRange range,
    int columnIndex,
    bool exactMatch
)
```

**Parameters**

- **queryTarget**
  - Type: NanoXLSXWorksheet  
  - Target worksheet of the query argument. Can be null if on the same worksheet

- **address**
  - Type: NanoXLSXCellAddress  
  - Query address of a cell as string as source of the lookup

- **rangeTarget**
Type: **NanoXLSXWorksheet**  
Target worksheet of the matrix. Can be null if on the same worksheet.

*range*  
Type: **NanoXLSXCellRange**  
Matrix of the lookup.

*columnIndex*  
Type: **SystemInt32**  
Column index of the target column (1 based)

*exactMatch*  
Type: **SystemBoolean**  
If true, an exact match is applied to the lookup.

**Return Value**  
Type: **Cell**  
Prepared Cell object, ready to be added to a worksheet.

**See Also**

**Reference**  
[CellBasicFormulas Class](#)  
[VLookup Overload](#)  
[NanoXLSX Namespace](#)
CellCellType Enumeration

Enum defines the basic data types of a cell

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
public enum CellType
```

Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRING</td>
<td>0</td>
<td>Type for single characters and strings</td>
</tr>
<tr>
<td>NUMBER</td>
<td>1</td>
<td>Type for all numeric types (long, integer and float and double)</td>
</tr>
<tr>
<td>DATE</td>
<td>2</td>
<td>Type for dates and times (Note: Dates before 1900-01-01 are not allowed)</td>
</tr>
<tr>
<td>BOOL</td>
<td>3</td>
<td>Type for boolean</td>
</tr>
<tr>
<td>FORMULA</td>
<td>4</td>
<td>Type for Formulas (The cell will be handled differently)</td>
</tr>
<tr>
<td>EMPTY</td>
<td>5</td>
<td>Type for empty cells. This type is only used for</td>
</tr>
</tbody>
</table>
merged cells (all cells except the first of the cell range)

| DEFAULT | 6 | Default Type, not specified |

See Also

Reference
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellRange Structure

Struct representing a cell range with a start and end address

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public struct Range
```

The CellRange type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CellRange(String)</td>
<td>Constructor with a range string as argument</td>
</tr>
<tr>
<td>CellRange(CellAddress,</td>
<td>Constructor with addresses as arguments</td>
</tr>
<tr>
<td>CellAddress)</td>
<td></td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Indicates whether this instance and a specified</td>
</tr>
</tbody>
</table>
object are equal. (Inherited from `ValueType`.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Returns the hash code for this instance. (Inherited from <code>ValueType</code>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Overwritten ToString method (Overrides <code>ValueType.ToString</code>.)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Top</th>
</tr>
</thead>
</table>

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EndAddress</strong></td>
<td>End address of the range</td>
</tr>
<tr>
<td><strong>StartAddress</strong></td>
<td>Start address of the range</td>
</tr>
</tbody>
</table>

Top
See Also

Reference
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Range Constructor

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CellRange(String)</td>
<td>Constructor with a range string as argument</td>
</tr>
<tr>
<td>CellRange(CellAddress, CellAddress)</td>
<td>Constructor with addresses as arguments</td>
</tr>
</tbody>
</table>

See Also

Reference
CellRange Structure
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellRange Constructor (String)

Constructor with a range string as argument

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Range(string range)
```

### Parameters

- **range**
  - Type: System.String
  - Address range (e.g. 'A1:B12')

### See Also

- Reference
  - CellRange Structure
  - CellRange Overload
  - NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellRange Constructor (CellAddress, CellAddress)

Constructor with addresses as arguments

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Range(
    CellAddress start,
    CellAddress end
)
```

### Parameters

**start**  
Type: NanoXLSXCellAddress  
Start address of the range

**end**  
Type: NanoXLSXCellAddress  
End address of the range

### See Also

**Reference**  
CellRange Structure  
CellRange Overload  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Range Methods

The **CellRange** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Equals</code></td>
<td>Indicates whether this instance and a specified object are equal.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>ValueType</code>.)</td>
</tr>
<tr>
<td><code>Finalize</code></td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>GetHashCode</code></td>
<td>Returns the hash code for this instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>ValueType</code>.)</td>
</tr>
<tr>
<td><code>GetType</code></td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>ToString</code></td>
<td>Overwritten <code>ToString</code> method (Overrides <code>ValueTypeToString</code>.)</td>
</tr>
</tbody>
</table>
See Also

Reference
CellRange Structure
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellRangeToString Method

Overwritten ToString method

**Namespace:** NanoLSX

**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public override string ToString()
```

### Return Value

**Type:** String

Returns the range (e.g. 'A1:B12')

### See Also

**Reference**

- CellRange Structure
- NanoLSX Namespace

Copyright Raphael Stoeckli © 2018
Range Fields

The **CellRange** type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EndAddress</td>
<td>End address of the range</td>
</tr>
<tr>
<td>StartAddress</td>
<td>Start address of the range</td>
</tr>
</tbody>
</table>

### See Also

**Reference**

- CellRange Structure
- NanoXLSX Namespace

Copyright Raphael Sto Eckli © 2018
CellRangeEndAddress Field

End address of the range

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public CellAddress EndAddress
```

### Field Value

Type: `CellAddress`

### See Also

**Reference**
- CellRange Structure
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
CellRangeStartAddress Field

Start address of the range

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public CellAddress StartAddress
```

Field Value  
Type: `CellAddress`

### See Also

**Reference**  
*CellRange Structure*  
*NanoXLSX Namespace*  

Copyright Raphael Stoeckli © 2018
Metadata Class

Class representing the metadata of a workbook

Inheritance Hierarchy

- System
- Object
- NanoXLXIMetadata

Namespace: NanoXLX
Assembly: NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
public class Metadata
```

The Metadata type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metadata</td>
<td>Default constructor</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Gets or sets the application which created the workbook. Default is NanoXLX</td>
</tr>
<tr>
<td><strong>ApplicationVersion</strong></td>
<td>Gets or sets the version of the creation application. Default is the library version of NanoXLSX. Use the format <code>xxxxx.yyyyy</code> (e.g. 1.0 or 55.9875) in case of a custom value.</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Category</strong></td>
<td>Gets or sets the category of the document. There are no predefined values or restrictions about the content of this field</td>
</tr>
<tr>
<td><strong>Company</strong></td>
<td>Gets or sets the company owning the document. This value is for organizational purpose. Add more than one manager by using the semicolon (;) between the words</td>
</tr>
<tr>
<td><strong>ContentStatus</strong></td>
<td>Gets or sets the status of the document. There are no predefined values or restrictions about the content of this field</td>
</tr>
<tr>
<td><strong>Creator</strong></td>
<td>Gets or sets the creator of the workbook. Add more than one creator by using the semicolon (;) between the authors</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Gets or sets the description of the document or comment about it</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HyperlinkBase</td>
<td>Gets or sets the hyper-link base of the document.</td>
</tr>
<tr>
<td>Keywords</td>
<td>Gets or sets the keywords of the workbook. Separate particular keywords with semicolons (;)</td>
</tr>
<tr>
<td>Manager</td>
<td>Gets or sets the responsible manager of the document. This value is for organizational purpose.</td>
</tr>
<tr>
<td>Subject</td>
<td>Gets or sets the subject of the workbook</td>
</tr>
<tr>
<td>Title</td>
<td>Gets or sets the title of the workbook</td>
</tr>
<tr>
<td>UseColorMRU</td>
<td>Gets or sets the whether custom defined colors (in styles) will be added as recent colors (MRU). If true. MRU information will be added</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckVersion</td>
<td>Checks the format of the passed version string</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td>ParseVersion</td>
<td>Method to parse a common version (major.minor.revision.build) into the compatible format (major.minor). The minimum value is 0.0 and the maximum value is 99999.99999. The minor, revision and build number are joined if possible. If the number is too long, the additional characters will be removed from the right side down to five characters (e.g. 785563 will be 78556)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that</td>
</tr>
</tbody>
</table>
represents the current object.
(Inherited from Object.)

See Also

Reference
NanoXLSX Namespace
Metadata Constructor

Default constructor

**Namespace:** NanoXLSX
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public Metadata()
```

### See Also

Reference
- Metadata Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
# Metadata Properties

The **Metadata** type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Gets or sets the application which created the workbook. Default is NanoXLSX</td>
</tr>
<tr>
<td>ApplicationVersion</td>
<td>Gets or sets the version of the creation application. Default is the library version of NanoXLSX. Use the format xxxx.yyyyy (e.g. 1.0 or 55.9875) in case of a custom value.</td>
</tr>
<tr>
<td>Category</td>
<td>Gets or sets the category of the document. There are no predefined values or restrictions about the content of this field</td>
</tr>
<tr>
<td>Company</td>
<td>Gets or sets the company owning the document. This value is for organizational purpose. Add more than one manager by using the semicolon (;) between the words</td>
</tr>
<tr>
<td>ContentStatus</td>
<td>Gets or sets the status of the</td>
</tr>
</tbody>
</table>
document. There are no predefined values or restrictions about the content of this field

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creator</td>
<td>Gets or sets the creator of the workbook. Add more than one creator by using the semicolon (;) between the authors.</td>
</tr>
<tr>
<td>Description</td>
<td>Gets or sets the description of the document or comment about it.</td>
</tr>
<tr>
<td>HyperlinkBase</td>
<td>Gets or sets the hyper-link base of the document.</td>
</tr>
<tr>
<td>Keywords</td>
<td>Gets or sets the keywords of the workbook. Separate particular keywords with semicolons (;).</td>
</tr>
<tr>
<td>Manager</td>
<td>Gets or sets the responsible manager of the document. This value is for organizational purpose.</td>
</tr>
<tr>
<td>Subject</td>
<td>Gets or sets the subject of the workbook.</td>
</tr>
<tr>
<td>Title</td>
<td>Gets or sets the title of the workbook.</td>
</tr>
<tr>
<td>UseColorMRU</td>
<td>Gets or sets the whether custom defined colors (in styles) will be added as recent colors (MRU). If true.</td>
</tr>
</tbody>
</table>
MRU information will be added

See Also

Reference
- Metadata Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
**MetadataApplication Property**

Gets or sets the application which created the workbook. Default is NanoXLSX

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public string Application { get; set; }
```

Property Value  
Type: String

### See Also

- Reference  
  - Metadata Class  
  - NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
**MetadataApplicationVersion Property**

Gets or sets the version of the creation application. Default is the library version of NanoXLSX. Use the format xxxxx.yyyyy (e.g. 1.0 or 55.9875) in case of a custom value.

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public string ApplicationVersion { get; set; }
```

Property Value  
Type: String

**See Also**

Reference  
Metadata Class  
NanoXLSX Namespace
**MetadataCategory Property**

Gets or sets the category of the document. There are no predefined values or restrictions about the content of this field.

**Namespace:** NanoXLXS

**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string Category { get; set; }
```

**Property Value**

Type: **String**

### See Also

**Reference**

- [Metadata Class](#)
- [NanoXLXS Namespace](#)

Copyright Raphael Stoeckli © 2018
MetadataCompany Property

Gets or sets the company owning the document. This value is for organizational purpose. Add more than one manager by using the semicolon (;) between the words

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string Company { get; set; }
```

Property Value

Type: String

### See Also

Reference

- Metadata Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
MetadataContentStatus Property

Gets or sets the status of the document. There are no predefined values or restrictions about the content of this field

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string ContentStatus { get; set; }
```

**Property Value**  
**Type:** `String`

### See Also

**Reference**  
- [Metadata Class](#)  
- [NanoXLSX Namespace](#)

Copyright Raphael Stoeckli © 2018
MetadataCreator Property

 Gets or sets the creator of the workbook. Add more than one creator by using the semicolon (;) between the authors

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```
public string Creator { get; set; }
```

Property Value

Type: String

See Also

Reference
   Metadata Class
   NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Metadata Description Property

Gets or sets the description of the document or comment about it

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string Description { get; set; }
```

**Property Value**  
**Type:** String

### See Also

**Reference**  
- Metadata Class  
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
MetadataHyperlinkBase Property

Gets or sets the hyper-link base of the document.

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string HyperlinkBase { get; set; }
```

### Property Value

Type: `String`

### See Also

Reference  
**Metadata Class**  
**NanoXLSX Namespace**

Copyright Raphael Stoeckli © 2018
MetadataKeywords Property

Gets or sets the keywords of the workbook. Separate particular keywords with semicolons (;)

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string Keywords { get; set; }
```

**Property Value**

Type: String

### See Also

**Reference**
- Metadata Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
MetadataManager Property

Gets or sets the responsible manager of the document. This value is for organizational purpose.

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public string Manager { get; set; }
```

Property Value
Type: String

See Also

Reference
Metadata Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
# MetadataSubject Property

Gets or sets the subject of the workbook

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public string Subject { get; set; }
```

## Property Value

Type: `String`

## See Also

Reference  
- Metadata Class  
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
MetadataTitle Property

Gets or sets the title of the workbook

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string Title { get; set; }
```

**Property Value**

**Type:** String

### See Also

**Reference**
- Metadata Class
- NanoXLSX Namespace

---

Copyright Raphael Stoeckli © 2018
MetadataUseColorMRU Property

Gets or sets the whether custom defined colors (in styles) will be added as recent colors (MRU). If true, MRU information will be added

**Namespace:** NanoXLXS

**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public bool UseColorMRU { get; set; }
```

**Property Value**

Type: Boolean

### See Also

**Reference**

- Metadata Class
- NanoXLXS Namespace

Copyright Raphael Stoeckli © 2018
# Metadata Methods

The Metadata type exposes the following members.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckVersion</td>
<td>Checks the format of the passed version string</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ParseVersion</strong></td>
<td>Method to parse a common version (major.minor.revision.build) into the compatible format (major.minor). The minimum value is 0.0 and the maximum value is 99999.99999. The minor, revision and build number are joined if possible. If the number is too long, the additional characters will be removed from the right side down to five characters (e.g. 785563 will be 78556)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>
NanoXLSX Library Documentation
Metadata

**CheckVersion Method**

Checks the format of the passed version string.

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private void CheckVersion()
```

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormatException</td>
<td>Throws a FormatException if the version string is malformed</td>
</tr>
</tbody>
</table>

### See Also

Reference  
Metadata Class  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Metadata ParseVersion Method

Method to parse a common version (major.minor.revision.build) into the compatible format (major.minor). The minimum value is 0.0 and the maximum value is 999999.99999. The minor, revision and build number are joined if possible. If the number is too long, the additional characters will be removed from the right side down to five characters (e.g. 785563 will be 78556).

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static string ParseVersion(
    int major,
    int minor,
    int build,
    int revision
)
```

### Parameters

- **major**
  - Type: `SystemInt32`
  - Major number from 0 to 99999

- **minor**
  - Type: `SystemInt32`
  - Minor number

- **build**
  - Type: `SystemInt32`
  - Build number

- **revision**
  - Build number
Type: `System.Int32`
Revision number

Return Value
Type: `String`
Formatted version number (e.g. 1.0 or 55.987)

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FormatException</code></td>
<td>Throws a <code>FormatException</code> if the major number is too long or one of the numbers is negative</td>
</tr>
</tbody>
</table>

### See Also

Reference
- `Metadata Class`
- `NanoXLSX Namespace`

Copyright Raphael Stoeckli © 2018
Workbook Class

Class representing a workbook

Inheritance Hierarchy

- System
  - Object
  - NanoXLSXWorkbook

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public class Workbook
```

The Workbook type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workbook(Boolean)</td>
<td>Constructor with additional parameter to create a default worksheet. This constructor can be used to define a workbook that is saved as stream</td>
</tr>
<tr>
<td>Workbook(String)</td>
<td>Constructor with additional parameter to create a default worksheet with the specified name. This constructor can be used to</td>
</tr>
</tbody>
</table>
define a workbook that is saved as stream

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentWorksheet</td>
<td>Gets the current worksheet</td>
</tr>
<tr>
<td>Filename</td>
<td>Gets or sets the filename of the workbook</td>
</tr>
<tr>
<td>LockStructureIfProtected</td>
<td>Gets whether the structure are locked if workbook is protected</td>
</tr>
<tr>
<td>LockWindowsIfProtected</td>
<td>Gets whether the windows are locked if workbook is protected</td>
</tr>
<tr>
<td>SelectedWorksheet</td>
<td>Gets the selected worksheet. The</td>
</tr>
</tbody>
</table>
selected worksheet is not the current worksheet while design time but the selected sheet in the output file

<table>
<thead>
<tr>
<th>Styles</th>
<th>Gets the style manager of this workbook</th>
</tr>
</thead>
<tbody>
<tr>
<td>UseWorkbookProtection</td>
<td>Gets or sets whether the workbook is protected</td>
</tr>
<tr>
<td>WorkbookMetadata</td>
<td>Meta data object of the workbook</td>
</tr>
<tr>
<td>WorkbookProtectionPassword</td>
<td>Gets the password used for workbook protection</td>
</tr>
<tr>
<td>Worksheets</td>
<td>Gets the list of worksheets in the workbook</td>
</tr>
<tr>
<td>WS</td>
<td>Gets the shortener object for the current worksheet</td>
</tr>
</tbody>
</table>

Top
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddStyle</td>
<td>Adds a style to the style manager</td>
</tr>
<tr>
<td>AddStyleComponent</td>
<td>Adds a style component to a style</td>
</tr>
<tr>
<td>AddWorksheet(String)</td>
<td>Adding a new Worksheet. The new worksheet will be defined as current worksheet</td>
</tr>
<tr>
<td>AddWorksheet(Worksheet)</td>
<td>Adding a new Worksheet. The new worksheet will be defined as current worksheet</td>
</tr>
<tr>
<td>AddWorksheet(String, Boolean)</td>
<td>Adding a new Worksheet with a sanitizing option. The new worksheet will be defined as current worksheet</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the</td>
</tr>
</tbody>
</table>
Finalize

Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)

GetHashCode

Serves as a hash function for a particular type. (Inherited from Object.)

GetType

Gets the Type of the current instance. (Inherited from Object.)

Init

Init method called in the constructors

Load(Stream)

Loads a workbook from
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load(String)</td>
<td>Loads a workbook from a file</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>RemoveStyle(String)</td>
<td>Removes the defined style from the style sheet of the workbook</td>
</tr>
<tr>
<td>RemoveStyle(Style)</td>
<td>Removes the passed style from the style sheet</td>
</tr>
<tr>
<td>RemoveStyle(String, Boolean)</td>
<td>Removes the defined style from the style sheet of the workbook</td>
</tr>
<tr>
<td>RemoveStyle(Style, Boolean)</td>
<td>Removes the defined style from the style sheet of the workbook</td>
</tr>
<tr>
<td>RemoveWorksheet</td>
<td>Removes the defined</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ResolveMergedCells</td>
<td>Method to resolve all merged cells in all worksheets. Only the value of the very first cell of the locked cells range will be visible. The other values are still present (set to EMPTY) but will not be stored in the worksheet.</td>
</tr>
<tr>
<td>Save</td>
<td>Saves the workbook</td>
</tr>
<tr>
<td>SaveAs</td>
<td>Saves the workbook with the defined name</td>
</tr>
<tr>
<td>SaveAsStream</td>
<td>Save the workbook to a writable stream</td>
</tr>
<tr>
<td>SetCurrentWorksheet</td>
<td>Sets the current worksheet</td>
</tr>
<tr>
<td>SetSelectedWorksheet(Int32)</td>
<td>Sets the selected worksheet in</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>SetSelectedWorksheet(Worksheet)</strong></td>
<td>Sets the selected worksheet in the output workbook.</td>
</tr>
<tr>
<td><strong>SetWorkbookProtection</strong></td>
<td>Sets or removes the workbook protection. If <code>protectWindows</code> and <code>protectStructure</code> are both false, the workbook will not be protected.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference

NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
# Workbook Constructor

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workbook(Boolean)</td>
<td>Constructor with additional parameter to create a default worksheet. This constructor can be used to define a workbook that is saved as stream</td>
</tr>
<tr>
<td>Workbook(String)</td>
<td>Constructor with additional parameter to create a default worksheet with the specified name. This constructor can be used to define a workbook that is saved as stream</td>
</tr>
<tr>
<td>Workbook(String, String)</td>
<td>Constructor with filename and the name of the first worksheet</td>
</tr>
<tr>
<td>Workbook(String, String, Boolean)</td>
<td>Constructor with filename and the name of the first worksheet</td>
</tr>
</tbody>
</table>

## See Also

Reference
- Workbook Class
Workbook Constructor (Boolean)

Constructor with additional parameter to create a default worksheet. This constructor can be used to define a workbook that is saved as stream.

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
public Workbook(
    bool createWorkSheet
)
```

Parameters

`createWorkSheet`
Type: `System.Boolean`
If true, a default worksheet with the name 'Sheet1' will be created and set as current worksheet.

See Also

Reference
Workbook Class
Workbook Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
Workbook Constructor (String)

Constructor with additional parameter to create a default worksheet with the specified name. This constructor can be used to define a workbook that is saved as stream

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Workbook(  
    string sheetName
  )
```

### Parameters

**sheetName**  
Type: `System.String`  
Filename of the workbook. The name will be sanitized automatically according to the specifications of Excel

### See Also

**Reference**  
Workbook Class  
Workbook Overload  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Workbook Constructor (String, String)

Constructor with filename and the name of the first worksheet.

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public Workbook(
    string filename,
    string sheetName
)
```

### Parameters

- **filename**  
  Type: System.String  
  Filename of the workbook. The name will be sanitized automatically according to the specifications of Excel.

- **sheetName**  
  Type: System.String  
  Name of the first worksheet. The name will be sanitized automatically according to the specifications of Excel.

## See Also

- Reference
  - Workbook Class
  - Workbook Overload
  - NanoXLSX Namespace
NanoXLSX Library Documentation
Workbook Constructor (String, String, Boolean)

Constructor with filename and the name of the first worksheet

**Namespace:** NanoXLX
**Assembly:** NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

▶ Syntax

```
public Workbook(
    string filename,
    string sheetName,
    bool sanitizeSheetName
)
```

Parameters

- **filename**
  - Type: **System.String**
  - Filename of the workbook

- **sheetName**
  - Type: **System.String**
  - Name of the first worksheet

- **sanitizeSheetName**
  - Type: **System.Boolean**
  - If true, the name of the worksheet will be sanitized automatically according to the specifications of Excel

▶ See Also

Reference
Workbook Properties

The **Workbook** type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentWorksheet</td>
<td>Gets the current worksheet</td>
</tr>
<tr>
<td>Filename</td>
<td>Gets or sets the filename of the workbook</td>
</tr>
<tr>
<td>LockStructureIfProtected</td>
<td>Gets whether the structure are locked if workbook is protected</td>
</tr>
<tr>
<td>LockWindowsIfProtected</td>
<td>Gets whether the windows are locked if workbook is protected</td>
</tr>
<tr>
<td>SelectedWorksheet</td>
<td>Gets the selected worksheet. The selected worksheet is not the current worksheet while design time but the selected</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Styles</td>
<td>Gets the style manager of this workbook</td>
</tr>
<tr>
<td>UseWorkbookProtection</td>
<td>Gets or sets whether the workbook is protected</td>
</tr>
<tr>
<td>WorkbookMetadata</td>
<td>Meta data object of the workbook</td>
</tr>
<tr>
<td>WorkbookProtectionPassword</td>
<td>Gets the password used for workbook protection</td>
</tr>
<tr>
<td>Worksheets</td>
<td>Gets the list of worksheets in the workbook</td>
</tr>
<tr>
<td>WS</td>
<td>Gets the shortener object for the current worksheet</td>
</tr>
</tbody>
</table>

See Also

Reference

Workbook Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookCurrentWorksheet Property

Gets the current worksheet

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
public Worksheet CurrentWorksheet { get; }
```

Property Value
Type: Worksheet

See Also

Reference
Workbook Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookFilename Property

Gets or sets the filename of the workbook

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string Filename { get; set; }
```

### Property Value

Type: `String`

### See Also

Reference  
Workbook Class  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookLockStructureIfProtected Property

Gets whether the structure are locked if workbook is protected

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public bool LockStructureIfProtected { get; }
```

**Property Value**

Type: Boolean

**See Also**

Reference
*Workbook Class*
*NanoXLSX Namespace*

Copyright Raphael Stoeckli © 2018
WorkbookLockWindowsIfProtected Property

Gets whether the windows are locked if workbook is protected

Namespace: NanoXLX
Assembly: NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public bool LockWindowsIfProtected { get; }
```

Property Value
Type: Boolean

See Also

Reference
Workbook Class
NanoXLX Namespace

Copyright Raphael Stoeckli © 2018
Workbook

SelectedWorksheet Property

Gets the selected worksheet. The selected worksheet is not the current worksheet while design time but the selected sheet in the output file.

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public int SelectedWorksheet { get; }
```

Property Value

Type: Int32

See Also

Reference
Workbook Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookStyles Property

Gets the style manager of this workbook

**Namespace:** NanoLSX
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleManager Styles { get; }
```

### Property Value

Type: `StyleManager`

### See Also

**Reference**
- Workbook Class
- NanoLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookUseWorkbookProtection Property

Get or sets whether the workbook is protected

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public bool UseWorkbookProtection { get; set; }
```

**Property Value**  
Type: Boolean

### See Also

**Reference**  
Workbook Class  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookWorkbookMetadata Property

Metadata object of the workbook

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Metadata WorkbookMetadata { get; set; }
```

Property Value

Type: Metadata

### See Also

Reference

- Workbook Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
WorkbookProtectionPassword Property

Gets the password used for workbook protection

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string WorkbookProtectionPassword { get; }
```

**Property Value**

**Type:** String

### See Also

- **Reference**
  - Workbook Class
  - NanoXLSX Namespace

Copyright Raphaël Stoeckli © 2018
Workbook.Worksheets Property

Gets the list of worksheets in the workbook

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public List<Worksheet> Worksheets { get; }
```

**Property Value**  
Type: List<Worksheet>

**See Also**

- Reference  
  Workbook Class  
  NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookWS Property

Gets the shortener object for the current worksheet

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public WorkbookShortener WS { get; }
```

### Property Value

Type: WorkbookShortener

### See Also

- **Reference**
  - Workbook Class
  - NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
## Workbook Methods

The *Workbook* type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddStyle</td>
<td>Adds a style to the style manager</td>
</tr>
<tr>
<td>AddStyleComponent</td>
<td>Adds a style component to a style</td>
</tr>
<tr>
<td>AddWorksheet(String)</td>
<td>Adding a new Worksheet. The new worksheet will be defined as current worksheet</td>
</tr>
<tr>
<td>AddWorksheet(Worksheet)</td>
<td>Adding a new Worksheet. The new worksheet will be defined as current worksheet</td>
</tr>
<tr>
<td>AddWorksheet(String, Boolean)</td>
<td>Adding a new Worksheet with a sanitizing option. The new worksheet will</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified <em>Object</em> is equal to the current <em>Object</em>. (Inherited from <em>Object.</em>)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <em>Object.</em>)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from <em>Object.</em>)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <em>Type</em> of the current instance. (Inherited from <em>Object.</em>)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Init</td>
<td>Init method called in the constructors</td>
</tr>
<tr>
<td>Load(Stream)</td>
<td>Loads a workbook from a stream</td>
</tr>
<tr>
<td>Load(String)</td>
<td>Loads a workbook from a file</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>RemoveStyle(String)</td>
<td>Removes the defined style from the style sheet of the workbook</td>
</tr>
<tr>
<td>RemoveStyle(Style)</td>
<td>Removes the passed style from the style sheet</td>
</tr>
<tr>
<td>RemoveStyle(String, Boolean)</td>
<td>Removes the defined style from the style sheet of the workbook</td>
</tr>
<tr>
<td>RemoveStyle(Style, Boolean)</td>
<td>Removes the defined style</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RemoveWorksheet</td>
<td>Removes the defined worksheet from the style sheet of the workbook</td>
</tr>
<tr>
<td>ResolveMergedCells</td>
<td>Method to resolve all merged cells in all worksheets. Only the value of the very first cell of the locked cells range will be visible. The other values are still present (set to EMPTY) but will not be stored in the worksheet.</td>
</tr>
<tr>
<td>Save</td>
<td>Saves the workbook</td>
</tr>
<tr>
<td>SaveAs</td>
<td>Saves the workbook with the defined name</td>
</tr>
<tr>
<td>SaveAsStream</td>
<td>Save the workbook to a writable stream</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SetCurrentWorksheet</td>
<td>Sets the current worksheet</td>
</tr>
<tr>
<td>SetSelectedWorksheet(Int32)</td>
<td>Sets the selected worksheet in the output workbook</td>
</tr>
<tr>
<td>SetSelectedWorksheet(Worksheet)</td>
<td>Sets the selected worksheet in the output workbook</td>
</tr>
<tr>
<td>SetWorkbookProtection</td>
<td>Sets or removes the workbook protection. If protectWindows and protectStructure are both false, the workbook will not be protected</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

See Also

Top
Reference

Workbook Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookAddStyle Method

Adds a style to the style manager

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public Style AddStyle(Style style)
```

**Parameters**

- **style**
  - Type: NanoXLSX.StyleStyle
  - Style to add

**Return Value**

- Type: Style
  - Returns the managed style of the style manager

**See Also**

- Reference
  - Workbook Class
  - NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookAddStyleComponent Method

Adds a style component to a style

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public Style AddStyleComponent(
    Style baseStyle,
    AbstractStyle newComponent
)
```

### Parameters

- **baseStyle**  
  Type: NanoXLSX.StyleStyle  
  Style to append a component

- **newComponent**  
  Type: NanoXLSX.StyleAbstractStyle  
  Component to add to the baseStyle

### Return Value

Type: Style  
Returns the managed style of the style manager

### See Also

Reference  
Workbook Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
# WorkbookAddWorksheet Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddWorksheet(String)</td>
<td>Adding a new Worksheet. The new worksheet will be defined as current worksheet</td>
</tr>
<tr>
<td>AddWorksheet(Worksheet)</td>
<td>Adding a new Worksheet. The new worksheet will be defined as current worksheet</td>
</tr>
<tr>
<td>AddWorksheet(String, Boolean)</td>
<td>Adding a new Worksheet with a sanitizing option. The new worksheet will be defined as current worksheet</td>
</tr>
</tbody>
</table>

## See Also

- **Reference**
  - Workbook Class
  - NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookAddWorksheet Method (String)

Adding a new Worksheet. The new worksheet will be defined as current worksheet

Namespace: NanoXLSP
Assembly: NanoXLSP (in NanoXLSP.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public void AddWorksheet(
    string name
)
```

Parameters

- `name`
  - Type: SystemString
  - Name of the new worksheet

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorksheetException</td>
<td>Throws a WorksheetNameAlreadyExistsException if the name of the worksheet already exists</td>
</tr>
<tr>
<td>FormatException</td>
<td>Throws a FormatException if the name contains illegal characters or is out of</td>
</tr>
</tbody>
</table>
See Also

Reference

Workbook Class
AddWorksheet Overload
NanoXLSX Namespace
WorkbookAddWorksheet Method (Worksheet)

Adding a new Worksheet. The new worksheet will be defined as current worksheet

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void AddWorksheet(
    Worksheet worksheet
)
```

### Parameters

- **worksheet**  
  Type: NanoXLSXWorksheet  
  Prepared worksheet object

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorksheetException</td>
<td>WorksheetException is thrown if the name of the worksheet already exists</td>
</tr>
<tr>
<td>FormatException</td>
<td>FormatException is thrown if the worksheet name contains illegal characters or is out of range (length between 1 and 31)</td>
</tr>
</tbody>
</table>
See Also

Reference
Workbook Class
AddWorksheet Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookAddWorksheet Method (String, Boolean)

Adding a new Worksheet with a sanitizing option. The new worksheet will be defined as current worksheet

**Namespace:** NanoXLXSX  
**Assembly:** NanoXSX (in NanoXSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void AddWorksheet(
    string name,
    bool sanitizeSheetName
)
```

### Parameters

- **name**
  - Type: `System.String`  
  - Name of the new worksheet

- **sanitizeSheetName**
  - Type: `System.Boolean`
  - If true, the name of the worksheet will be sanitized automatically according to the specifications of Excel

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorksheetException</td>
<td>WorksheetException is thrown if the name of the worksheet already exists</td>
</tr>
</tbody>
</table>
and sanitizeSheetName is false

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormatException</td>
<td>FormatException is thrown if the worksheet name contains illegal characters or is out of range (length between 1 and 31) and sanitizeSheetName is false</td>
</tr>
</tbody>
</table>

**See Also**

**Reference**
- Workbook Class
- AddWorksheet Overload
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookInit Method

Init method called in the constructors

**Namespace:** NanoXLSX
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private void Init()
```

### See Also

Reference
Workbook Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookLoad Method

▲ Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load(Stream)</td>
<td>Loads a workbook from a stream</td>
</tr>
<tr>
<td>Load(String)</td>
<td>Loads a workbook from a file</td>
</tr>
</tbody>
</table>

Top

▲ See Also

Reference
Workbook Class
NanoXLXSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookLoad Method (Stream)

Loads a workbook from a stream

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
public static Workbook Load(
    Stream stream
)
```

Parameters

stream
- Type: System.IOStream
  - Stream containing the workbook

Return Value
- Type: Workbook
  - Workbook object

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>Throws IOException in case of an error</td>
</tr>
</tbody>
</table>

See Also

Reference
Workbook Class
WorkbookLoad Method (String)

Loads a workbook from a file

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Workbook Load(
    string filename
)
```

### Parameters

**filename**
- Type: `System.String`  
  - Filename of the workbook

### Return Value
- Type: `Workbook`  
  - Workbook object

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>Throws IOException in case of an error</td>
</tr>
</tbody>
</table>

### See Also

Reference
- Workbook Class
WorkbookRemoveStyle Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoveStyle(String)</td>
<td>Removes the defined style from the style sheet of the workbook</td>
</tr>
<tr>
<td>RemoveStyle(Style)</td>
<td>Removes the passed style from the style sheet</td>
</tr>
<tr>
<td>RemoveStyle(String, Boolean)</td>
<td>Removes the defined style from the style sheet of the workbook</td>
</tr>
<tr>
<td>RemoveStyle(Style, Boolean)</td>
<td>Removes the defined style from the style sheet of the workbook</td>
</tr>
</tbody>
</table>

See Also

Reference
Workbook Class
NanoXLSX Namespace
WorkbookRemoveStyle Method (String)

Removes the defined style from the style sheet of the workbook

**Namespace:**  NanoLSX  
**Assembly:**  NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

C#  

```csharp
public void RemoveStyle(
    string styleName
)
```

### Parameters

`styleName`  
Type:  `SystemString`  
Name of the style to be removed

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws an StyleException if the style was not found in the style collection (could not be referenced)</td>
</tr>
</tbody>
</table>

### See Also

Reference  
*Workbook Class*
WorkbookRemoveStyle Method (Style)

Removes the passed style from the style sheet

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public void RemoveStyle(
    Style style
)
```

### Parameters

**style**  
Type: NanoXLSX.StyleStyle  
Style to remove

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws an StyleException if the style was not found in the style collection (could not be referenced)</td>
</tr>
</tbody>
</table>

### See Also

Reference  
Workbook Class
WorkbookRemoveStyle Method
(String, Boolean)

Removes the defined style from the style sheet of the workbook

**Namespace:** NanoXLSX
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public void RemoveStyle(
    string styleName,
    bool onlyIfUnused
)
```

### Parameters

- **styleName**
  - Type: System.String
  - Name of the style to be removed

- **onlyIfUnused**
  - Type: System.Boolean
  - If true, the style will only be removed if not used in any cell

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws an UndefinedStyleException if the style was not found in the style collection (could not be referenced)</td>
</tr>
</tbody>
</table>
See Also

Reference
  Workbook Class
  RemoveStyle Overload
  NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookRemoveStyle Method (Style, Boolean)

Removes the defined style from the style sheet of the workbook

**Namespace:** NanoLSX  
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void RemoveStyle(
    Style style,
    bool onlyIfUnused
)
```

### Parameters

- **style**
  
  Type: NanoLSX.Style
  
  Style to remove

- **onlyIfUnused**
  
  Type: System.Boolean
  
  If true, the style will only be removed if not used in any cell

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws an StyleException if the style was not found in the style collection (could not be referenced)</td>
</tr>
</tbody>
</table>
See Also

Reference
Workbook Class
RemoveStyle Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
WorkbookRemoveWorksheet Method

Removes the defined worksheet

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```c#
public void RemoveWorksheet(
    string name
)
```

**Parameters**

*name*

Type: `System.String`
Name of the worksheet

**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorksheetException</td>
<td>Throws a UnknownWorksheetException if the name of the worksheet is unknown</td>
</tr>
</tbody>
</table>

**See Also**

Reference
- Workbook Class
- NanoXLSX Namespace
WorkbookResolveMergedCells Method

Method to resolve all merged cells in all worksheets. Only the value of the very first cell of the locked cells range will be visible. The other values are still present (set to EMPTY) but will not be stored in the worksheet.

**Namespace:** NanoXLSEX
**Assembly:** NanoXLSEX (in NanoXLSEX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void ResolveMergedCells()
```

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws an StyleException if one of the styles of the merged cells cannot be referenced or is null</td>
</tr>
</tbody>
</table>

### See Also

- Reference
  - Workbook Class
  - NanoXLSEX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookSave Method

Saves the workbook

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public void Save()
```

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>Throws IOException in case of an error</td>
</tr>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the start or end address of a handled cell range was out of range</td>
</tr>
<tr>
<td>FormatException</td>
<td>Throws a FormatException if a handled date cannot be translated to (Excel internal) OADate</td>
</tr>
<tr>
<td>StyleException</td>
<td>Throws an StyleException if one of the styles of the workbook cannot be referenced or is null</td>
</tr>
</tbody>
</table>

### See Also

Reference  
Workbook Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookSaveAs Method

Saves the workbook with the defined name

**Namespace:** NanoXLSSX  
**Assembly:** NanoXLSSX (in NanoXLSSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public void SaveAs(
    string filename
)
```

### Parameters

- **filename**
  - Type: `System.String`
  - filename of the saved workbook

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>Throws IOException in case of an error</td>
</tr>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the start or end address of a handled cell range was out of range</td>
</tr>
<tr>
<td>FormatException</td>
<td>Throws a FormatException if a handled date cannot be translated to (Excel internal) OADate</td>
</tr>
<tr>
<td>StyleException</td>
<td>Throws an StyleException if one of the</td>
</tr>
</tbody>
</table>
styles of the workbook cannot be referenced or is null

See Also

Reference
Workbook Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookSaveAsStream Method

Save the workbook to a writable stream

**Namespace:**  NanoXL SX  
**Assembly:**  NanoXL SX (in NanoXL SX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public void SaveAsStream(
    Stream stream
)
```

### Parameters

**stream**
- Type: `System.IOStream`
- Writable stream

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>Throws IOException in case of an error</td>
</tr>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the start or end address of a handled cell range was out of range</td>
</tr>
<tr>
<td>FormatException</td>
<td>Throws a FormatException if a handled date cannot be translated to (Excel internal) OADate</td>
</tr>
<tr>
<td>StyleException</td>
<td>Throws an StyleException if one of the styles of the workbook cannot be referenced or is null</td>
</tr>
</tbody>
</table>

See Also

Reference
- Workbook Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookSetCurrentWorksheet Method

Sets the current worksheet

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Worksheet SetCurrentWorksheet(
    string name
)
```

### Parameters

**name**  
Type: System.String  
Name of the worksheet

### Return Value

Type: Worksheet  
Returns the current worksheet

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorksheetException</td>
<td>Throws a MissingReferenceException if the name of the worksheet is unknown</td>
</tr>
</tbody>
</table>

### See Also
Reference

Workbook Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Workbook\SetSelectedWorksheet Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetSelectedWorksheet(Int32)</td>
<td>Sets the selected worksheet in the output workbook</td>
</tr>
<tr>
<td>SetSelectedWorksheet(Worksheet)</td>
<td>Sets the selected worksheet in the output workbook</td>
</tr>
</tbody>
</table>

See Also

Reference
Workbook Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookSetSelectedWorksheet Method (Int32)

Sets the selected worksheet in the output workbook

**Namespace:** NanoXLX
**Assembly:** NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void SetSelectedWorksheet(
    int worksheetIndex
)
```

### Parameters

- **worksheetIndex**
  - Type: `System.Int32`
  - Zero-based worksheet index

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws a OutOfRangeException if the index of the worksheet is out of range</td>
</tr>
</tbody>
</table>

### Remarks

This method does not set the current worksheet while design time. Use SetCurrentWorksheet instead for this
See Also

Reference

Workbook Class
SetSelectedWorksheet Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookSetSelectedWorksheet Method (Worksheet)

Sets the selected worksheet in the output workbook

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
C#

public void SetSelectedWorksheet(
    Worksheet worksheet
)
```

### Parameters

- `worksheet`  
  Type: NanoXLSXWorksheet  
  Worksheet object (must be in the collection of worksheets)

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorksheetException</td>
<td>Throws a UnknownWorksheetException if the worksheet was not found in the worksheet collection</td>
</tr>
</tbody>
</table>

### Remarks

This method does not set the current worksheet while design time. Use SetCurrentWorksheet instead for this
See Also

Reference
Workbook Class
SetSelectedWorksheet Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookSetWorkbookProtection Method

Sets or removes the workbook protection. If protectWindows and protectStructure are both false, the workbook will not be protected

**Namespace:** NanoXLXSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public void SetWorkbookProtection(
    bool state,
    bool protectWindows,
    bool protectStructure,
    string password
)
```

### Parameters

- **state**
  - Type: `SystemBoolean`
  - If true, the workbook will be protected, otherwise not

- **protectWindows**
  - Type: `SystemBoolean`
  - If true, the windows will be locked if the workbook is protected

- **protectStructure**
  - Type: `SystemBoolean`
  - If true, the structure will be locked if the workbook is protected

- **password**
  - Type: `SystemString`
  - Optional password. If null or empty, no password will be set in
case of protection

See Also

Reference

Workbook Class
NanoXLXSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookShortener Class

Class to provide access to the current worksheet with a shortened syntax. Note: The WS object can be null if the workbook was created without a worksheet. The object will be available as soon as the current worksheet is defined.

Inheritance Hierarchy

- System
- Object
- NanoXLX
- WorkbookShortener

Namespace: NanoXLX
Assembly: NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
public class Shortener
```

The WorkbookShortener type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorkbookShortener</td>
<td>Default constructor</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Down</td>
<td>Moves the cursor one row</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Down(Int32)</strong></td>
<td>Moves the cursor the number of defined rows down</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Formula(String)</strong></td>
<td>Sets a formula into the current cell and moves the cursor to the next cell (column or row depending on the defined cell direction)</td>
</tr>
<tr>
<td><strong>Formula(String, Style)</strong></td>
<td>Sets a formula with style into the current cell and moves the cursor to the next cell (column or row depending on the defined cell direction)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>NullCheck</strong></td>
<td>Internal method to check whether the worksheet is null.</td>
</tr>
<tr>
<td><strong>Right</strong></td>
<td>Moves the cursor one column to the right.</td>
</tr>
<tr>
<td><strong>Right(Int32)</strong></td>
<td>Moves the cursor the number of defined columns to the right.</td>
</tr>
<tr>
<td><strong>SetCurrentWorksheet</strong></td>
<td>Sets the worksheet accessed by the shortener.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object.</td>
</tr>
<tr>
<td><strong>Value(Object)</strong></td>
<td>Sets a value into the current cell and moves the cursor to the next cell</td>
</tr>
<tr>
<td></td>
<td>(column or row depending on the defined cell direction).</td>
</tr>
<tr>
<td><strong>Value(Object, Style)</strong></td>
<td>Sets a value with style into the current cell and moves the cursor to the</td>
</tr>
<tr>
<td></td>
<td>next cell (column or row depending on the defined cell).</td>
</tr>
</tbody>
</table>
See Also

Reference

NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookShortener Constructor

Default constructor

**Namespace:** NanoXLXS
**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Shortener()
```

### See Also

Reference
- WorkbookShortener Class
- NanoXLXS Namespace
## Shortener Methods

The **WorkbookShortener** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Down" /></td>
<td><strong>Down</strong> Moves the cursor one row down</td>
</tr>
<tr>
<td>![Down(Int32)]</td>
<td><strong>Down(Int32)</strong> Moves the cursor the number of defined rows down</td>
</tr>
<tr>
<td><img src="image" alt="Equals" /></td>
<td><strong>Equals</strong> Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><img src="image" alt="Finalize" /></td>
<td><strong>Finalize</strong> Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>![Formula(String)]</td>
<td><strong>Formula(String)</strong> Sets a formula into the current cell and moves the cursor to the next cell (column or row depending on the defined cell direction)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Formula(String, Style)</td>
<td>Sets a formula with style into the current cell and moves the cursor to the next cell (column or row depending on the defined cell direction)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>NullCheck</td>
<td>Internal method to check whether the worksheet is null</td>
</tr>
<tr>
<td>Right</td>
<td>Moves the cursor one column to the right</td>
</tr>
<tr>
<td>Right(Int32)</td>
<td>Moves the cursor the number of defined columns to the right</td>
</tr>
<tr>
<td>SetCurrentWorksheet</td>
<td>Sets the worksheet accessed by the shortener</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Value(Object)</td>
<td>Sets a value into the current cell and moves the cursor to the next cell (column or row depending on the defined cell direction)</td>
</tr>
<tr>
<td>Value(Object, Style)</td>
<td>Sets a value with style into the current cell and moves the cursor to the next cell (column or row depending on the defined cell direction)</td>
</tr>
</tbody>
</table>

See Also

Reference
- WorkbookShortener Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Shortener Down Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Down</td>
<td>Moves the cursor one row down</td>
</tr>
<tr>
<td>Down(Int32)</td>
<td>Moves the cursor the number of defined rows down</td>
</tr>
</tbody>
</table>

### See Also

Reference
- WorkbookShortener Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookShortenerDown Method

Moves the cursor one row down

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void Down()
```

### See Also

Reference
- WorkbookShortener Class
- Down Overload
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookShortenerDown Method (Int32)

Moves the cursor the number of defined rows down

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public void Down(
    int numberOfRows
)
```

### Parameters

**numberOfRows**  
Type: SystemInt32  
Number of rows to move

### See Also

Reference  
WorkbookShortener Class  
Down Overload  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
ShortenerFormula Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Formula(String)</code></td>
<td>Sets a formula into the current cell and moves the cursor to the next cell (column or row depending on the defined cell direction)</td>
</tr>
<tr>
<td><code>Formula(String, Style)</code></td>
<td>Sets a formula with style into the current cell and moves the cursor to the next cell (column or row depending on the defined cell direction)</td>
</tr>
</tbody>
</table>

See Also

Reference

- WorkbookShortener Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookShortenerFormula Method (String)

Sets a formula into the current cell and moves the cursor to the next cell (column or row depending on the defined cell direction)

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void Formula(
    string formula
)
```

### Parameters

- **formula**  
  Type: System.String  
  Formula to set

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorksheetException</td>
<td>Throws a WorksheetException if no worksheet was defined</td>
</tr>
</tbody>
</table>

### See Also

Reference  
WorkbookShortener Class
WorkbookShortenerFormula Method (String, Style)

Sets a formula with style into the current cell and moves the cursor to the next cell (column or row depending on the defined cell direction)

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void Formula(
    string formula,
    Style style
)
```

### Parameters

- **formula**
  - Type: `System.String`  
  - Formula to set

- **style**
  - Type: `NanoXLSX.StyleStyle`  
  - Style to apply

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorksheetException</td>
<td>Throws a WorksheetException if no worksheet was defined</td>
</tr>
</tbody>
</table>
See Also

Reference
WorkbookShortener Class
Formula Overload
NanoXLNSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookShortenerNullCheck Method

Internal method to check whether the worksheet is null

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
private void NullCheck()
```

See Also

Reference
WorkbookShortener Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
# ShortenerRight Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Right</td>
<td>Moves the cursor one column to the right</td>
</tr>
<tr>
<td>Right(Int32)</td>
<td>Moves the cursor the number of defined columns to the right</td>
</tr>
</tbody>
</table>

## See Also

Reference
- WorkbookShortener Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookShortenerRight Method

Moves the cursor one column to the right

**Namespace:** NanoXL SX
**Assembly:** NanoXL SX (in NanoXL SX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void Right()
```

### See Also

Reference
- WorkbookShortener Class
- Right Overload
- NanoXL SX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookShortenerRight Method (Int32)

Moves the cursor the number of defined columns to the right

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

▲ Syntax

```csharp
public void Right(
    int numberOfColumns
)
```

Parameters

`numberOfColumns`  
Type: System.Int32  
Number of columns to move

▲ See Also

Reference  
WorkbookShortener Class  
Right Overload  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookShortenerSetCurrentWorksheet Method

Sets the worksheet accessed by the shortener

**Namespace:** NanoXL

**Assembly:** NanoXL (in NanoXL.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public void SetCurrentWorksheet(
    Worksheet worksheet
)
```

### Parameters

- **worksheet**
  - Type: NanoXLWorkbook
  - Current worksheet

### See Also

- **Reference**
  - WorkbookShortener Class
  - NanoXL Namespace

Copyright Raphael Stoeckli © 2018
# ShortenerValue Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value(Object)</td>
<td>Sets a value into the current cell and moves the cursor to the next cell (column or row depending on the defined cell direction)</td>
</tr>
<tr>
<td>Value(Object, Style)</td>
<td>Sets a value with style into the current cell and moves the cursor to the next cell (column or row depending on the defined cell direction)</td>
</tr>
</tbody>
</table>

## See Also

Reference
- Workbook
- Shortener Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorkbookShortener.Value Method (Object)

Sets a value into the current cell and moves the cursor to the next cell (column or row depending on the defined cell direction)

**Namespace:** NanoXLX
**Assembly:** NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public void Value(
    Object value
)
```

#### Parameters

- **value**
  - Type: `System.Object`
  - Value to set

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorksheetException</td>
<td>Throws a WorksheetException if no worksheet was defined</td>
</tr>
</tbody>
</table>

### See Also

Reference
- WorkbookShortener Class
WorkbookShortenerValue Method (Object, Style)

Sets a value with style into the current cell and moves the cursor to the next cell (column or row depending on the defined cell direction)

**Namespace:** NanoXLX

**Assembly:** NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public void Value(
    Object value,
    Style style
)
```

### Parameters

- **value**
  - Type: `System.Object`
  - Value to set

- **style**
  - Type: `NanoXLX.StyleStyle`
  - Style to apply

## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>WorksheetException</code></td>
<td>Throws a WorksheetException if no worksheet was defined</td>
</tr>
</tbody>
</table>
See Also

Reference
WorkbookShortener Class
Value Overload
NanoXLSX Namespace
Worksheet Class

Class representing a worksheet of a workbook

Inheritance Hierarchy

- SystemObject
- NanoXLSXWorksheet

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
public class Worksheet
```

The `Worksheet` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worksheet</td>
<td>Default Constructor</td>
</tr>
<tr>
<td>Worksheet(String, Int32, Workbook)</td>
<td>Constructor with name and sheet ID</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoFilterRange</td>
<td>Gets the range of the</td>
</tr>
</tbody>
</table>
auto-filter. Wrapped to Nullable to provide null as value. If null, no auto-filter is applied

<table>
<thead>
<tr>
<th>Cells</th>
<th>Gets the cells of the worksheet as dictionary with the cell address as key and the cell object as value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns</td>
<td>Gets all columns with non-standard properties, like auto filter applied or a special width as dictionary with the zero-based column index as key and the column object as value</td>
</tr>
<tr>
<td>CurrentCellDirection</td>
<td>Gets or sets the direction when using AddNextCell method</td>
</tr>
<tr>
<td>DefaultColumnWidth</td>
<td>Gets or sets the default column width</td>
</tr>
<tr>
<td>DefaultRowHeight</td>
<td>Gets or sets the default Row height</td>
</tr>
<tr>
<td>HiddenRows</td>
<td>Gets the hidden rows as dictionary with the zero-based row number as key and a boolean as value. True indicates hidden, false visible.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>MergedCells</strong></td>
<td>Gets the merged cells (only references) as dictionary with the cell address as key and the range object as value</td>
</tr>
<tr>
<td><strong>RowHeights</strong></td>
<td>Gets defined row heights as dictionary with the zero-based row number as key and the height (float from 0 to 409.5) as value</td>
</tr>
<tr>
<td><strong>SelectedCells</strong></td>
<td>Gets the cell range of selected cells of this worksheet. Null if no cells are selected</td>
</tr>
<tr>
<td><strong>SheetID</strong></td>
<td>Gets or sets the internal ID of the sheet</td>
</tr>
<tr>
<td><strong>SheetName</strong></td>
<td>Gets or sets the name of the worksheet</td>
</tr>
<tr>
<td><strong>SheetProtectionPassword</strong></td>
<td>Gets the password used for sheet protection</td>
</tr>
<tr>
<td><strong>SheetProtectionValues</strong></td>
<td>Gets the list of SheetProtectionValues. These values define the allowed actions if the worksheet is protected</td>
</tr>
<tr>
<td><strong>UseSheetProtection</strong></td>
<td>Gets or sets whether the worksheet is protected</td>
</tr>
</tbody>
</table>
protected. If true, protection is enabled

| WorkbookReference | Gets or sets the Reference to the parent Workbook |

**Top**

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddAllowedActionOnSheetProtection</td>
<td>Method to add allowed actions if the worksheet is protected. If one or more values are added, UseSheetProtection will be set to true</td>
</tr>
<tr>
<td>AddCell(Object, String)</td>
<td>Adds an object to the defined cell address. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted</td>
</tr>
<tr>
<td>AddCell(Object, Int32, Int32)</td>
<td>Adds an object to the defined cell</td>
</tr>
</tbody>
</table>
address. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted

AddCell(Object, String, Style) Adds an object to the defined cell address. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted

AddCell(Object, Int32, Int32, Style) Adds an object to the defined cell address. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddCellFormula(String, String)</td>
<td>Adds a cell formula as string to the defined cell address</td>
</tr>
<tr>
<td>AddCellFormula(String, Int32, Int32)</td>
<td>Adds a cell formula as string to the defined cell address</td>
</tr>
<tr>
<td>AddCellFormula(String, String, Style)</td>
<td>Adds a cell formula as string to the defined cell address</td>
</tr>
<tr>
<td>AddCellFormula(String, Int32, Int32, Style)</td>
<td>Adds a cell formula as string to the defined cell address</td>
</tr>
<tr>
<td>AddCellRange(ListObject, String)</td>
<td>Adds a list of object values to a defined cell range. If the type of the a particular value does not match with one of the supported data types, it will be casted to a String. Prepared objects of the type Cell will not be casted but adjusted</td>
</tr>
<tr>
<td>AddCellRange(ListObject, CellAddress, CellAddress)</td>
<td>Adds a list of object values to a defined cell range. If the type of the a particular value does not match with one of the</td>
</tr>
</tbody>
</table>
If the type of the a particular value does not match with one of the supported data types, it will be casted to a String. Prepared objects of the type Cell will not be casted but adjusted.

### AddCellRange(ListObject, String, Style)

Adds a list of object values to a defined cell range. If the type of the a particular value does not match with one of the supported data types, it will be casted to a String. Prepared objects of the type Cell will not be casted but adjusted.

### AddCellRange(ListObject, CellAddress, CellAddress, Style)

Adds a list of object values to a defined cell range. If the type of the a particular value does not match with one of the supported data types, it will be casted to a String. Prepared objects of the type Cell will not be casted but adjusted.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddCellRangeInternalT</td>
<td>Internal function to add a generic list of value to the defined cell range</td>
</tr>
<tr>
<td>AddHiddenColumn(Int32)</td>
<td>Sets the defined column as hidden</td>
</tr>
<tr>
<td>AddHiddenColumn(String)</td>
<td>Sets the defined column as hidden</td>
</tr>
<tr>
<td>AddHiddenRow</td>
<td>Sets the defined row as hidden</td>
</tr>
<tr>
<td>AddNextCell(Object)</td>
<td>Adds an object to the next cell position. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted.</td>
</tr>
<tr>
<td>AddNextCell(Object, Style)</td>
<td>Adds an object to the next cell position. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted.</td>
</tr>
<tr>
<td>Method/Property</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>AddNextCell</strong>&lt;br&gt;(Cell, Boolean, Style)**</td>
<td>Method to insert a generic cell to the next cell position</td>
</tr>
<tr>
<td><strong>AddNextCellFormula</strong>&lt;br&gt;(String)**</td>
<td>Adds a formula as string to the next cell position</td>
</tr>
<tr>
<td><strong>AddNextCellFormula</strong>&lt;br&gt;(String, Style)**</td>
<td>Adds a formula as string to the next cell position</td>
</tr>
<tr>
<td><strong>CastValue</strong></td>
<td>Method to cast a value or align an object of the type Cell to the context of the worksheet</td>
</tr>
<tr>
<td><strong>ClearActiveStyle</strong></td>
<td>Clears the active style of the worksheet. All later added calls will contain no style unless another active style is set</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.&lt;br&gt;(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources</td>
</tr>
</tbody>
</table>
and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCell(CellAddress)</td>
<td>Gets the cell of the specified address</td>
</tr>
<tr>
<td>GetCell(Int32, Int32)</td>
<td>Gets the cell of the specified column and row number (zero-based)</td>
</tr>
<tr>
<td>GetCurrentColumnNumber</td>
<td>Gets the current column number (zero based)</td>
</tr>
<tr>
<td>GetCurrentRowNumber</td>
<td>Gets the current row number (zero based)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetLastAddress</td>
<td>Gets the last existing row or column number of the current worksheet (zero-based)</td>
</tr>
<tr>
<td>GetLastColumnNumber</td>
<td>Gets the last</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>GetLastRowNumber</td>
<td>Gets the last existing row number in the current worksheet (zero-based)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GoToNextColumn</td>
<td>Moves the current position to the next column</td>
</tr>
<tr>
<td>GoToNextColumn(Int32)</td>
<td>Moves the current position to the next column with the number of cells to move</td>
</tr>
<tr>
<td>GoToNextRow</td>
<td>Moves the current position to the next row (use for a new line)</td>
</tr>
<tr>
<td>GoToNextRow(Int32)</td>
<td>Moves the current position to the next row with the number of cells to move (use for a new line)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HasCell(CellAddress)</td>
<td>Gets whether the specified address exists in the worksheet. Existing means that a value was stored at the address</td>
</tr>
<tr>
<td>HasCell(Int32, Int32)</td>
<td>Gets whether the specified address exists in the worksheet. Existing means that a value was stored at the address</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>MergeCells(String)</td>
<td>Merges the defined cell range</td>
</tr>
<tr>
<td>MergeCells(CellRange)</td>
<td>Merges the defined cell range</td>
</tr>
<tr>
<td>MergeCells(CellAddress, CellAddress)</td>
<td>Merges the defined cell range</td>
</tr>
<tr>
<td>RecalculateAutoFilter</td>
<td>Method to recalculate the auto filter (columns) of this worksheet. This is an internal method. There is no</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RecalculateColumns</td>
<td>Method to recalculate the collection of columns of this worksheet. This is an internal method. There is no need to use it. It must be public to require access from the LowLevel class.</td>
</tr>
<tr>
<td>RemoveAutoFilter</td>
<td>Removes auto filters from the worksheet.</td>
</tr>
<tr>
<td>RemoveCell(String)</td>
<td>Removes a previous inserted cell at the defined address.</td>
</tr>
<tr>
<td>RemoveCell(Int32, Int32)</td>
<td>Removes a previous inserted cell at the defined address.</td>
</tr>
<tr>
<td>RemoveHiddenColumn(Int32)</td>
<td>Sets a previously defined, hidden column as visible again.</td>
</tr>
<tr>
<td>RemoveHiddenColumn(String)</td>
<td>Sets a previously defined, hidden column as visible again.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>RemoveHiddenRow</td>
<td>Sets a previously defined, hidden row as visible again</td>
</tr>
<tr>
<td>RemoveMergedCells</td>
<td>Removes the defined merged cell range</td>
</tr>
<tr>
<td>RemoveSelectedCells</td>
<td>Removes the cell selection of this worksheet</td>
</tr>
<tr>
<td>SanitizeWorksheetName</td>
<td>Sanitizes a worksheet name</td>
</tr>
<tr>
<td>SetActiveStyle</td>
<td>Sets the active style of the worksheet. This style will be assigned to all later added cells</td>
</tr>
<tr>
<td>SetAutoFilter(String)</td>
<td>Sets the column auto filter within the defined column range</td>
</tr>
<tr>
<td>SetAutoFilter(Int32, Int32)</td>
<td>Sets the column auto filter within the defined column range</td>
</tr>
<tr>
<td>SetColumnHiddenState</td>
<td>Sets the defined column as hidden or visible</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SetColumnWidth(Int32, Single)</td>
<td>Sets the width of the passed column number (zero-based)</td>
</tr>
<tr>
<td>SetColumnWidth(String, Single)</td>
<td>Sets the width of the passed column address</td>
</tr>
<tr>
<td>SetCurrentCellAddress(String)</td>
<td>Set the current cell address</td>
</tr>
<tr>
<td>SetCurrentCellAddress(Int32, Int32)</td>
<td>Set the current cell address</td>
</tr>
<tr>
<td>SetCurrentColumnNumber</td>
<td>Sets the current column number (zero based)</td>
</tr>
<tr>
<td>SetCurrentRowNumber</td>
<td>Sets the current row number (zero based)</td>
</tr>
<tr>
<td>SetRowHeight</td>
<td>Sets the height of the passed row number (zero-based)</td>
</tr>
<tr>
<td>SetRowHiddenState</td>
<td>Sets the defined row as hidden or visible</td>
</tr>
<tr>
<td>SetSelectedCells(String)</td>
<td>Sets the selected cells on this worksheet</td>
</tr>
<tr>
<td>SetSelectedCells(CellRange)</td>
<td>Sets the selected cells on this worksheet</td>
</tr>
</tbody>
</table>
worksheet

SetSelectedCells(CellAddress, CellAddress)  
Sets the selected cells on this worksheet

SheetName  
Validates and sets the worksheet name

SetSheetName  
Sets the name of the sheet

SetSheetProtectionPassword  
Sets or removes the password for worksheet protection. If set, UseSheetProtection will also be set to true

ToString  
Returns a string that represents the current object. (Inherited from Object.)

WorksheetExists  
Checks whether a worksheet with the given name exists

**Top**

**Fields**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT_COLUMN_WIDTH</td>
<td>Default column</td>
</tr>
<tr>
<td>Symbol</td>
<td>Value</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>☐</td>
<td>DEFAULT_ROW_HEIGHT</td>
</tr>
<tr>
<td>☐</td>
<td>MAX_COLUMN_NUMBER</td>
</tr>
<tr>
<td>☐</td>
<td>MAX_COLUMN_WIDTH</td>
</tr>
<tr>
<td>☐</td>
<td>MAX_ROW_HEIGHT</td>
</tr>
<tr>
<td>☐</td>
<td>MAX_ROW_NUMBER</td>
</tr>
<tr>
<td>☐</td>
<td>MIN_COLUMN_NUMBER</td>
</tr>
<tr>
<td>☐</td>
<td>MIN_COLUMN_WIDTH</td>
</tr>
<tr>
<td>☐</td>
<td>MIN_ROW_HEIGHT</td>
</tr>
<tr>
<td>☐</td>
<td>MIN_ROW_NUMBER</td>
</tr>
</tbody>
</table>
See Also

Reference
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet Constructor

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worksheet</td>
<td>Default Constructor</td>
</tr>
<tr>
<td>Worksheet(String, Int32, Workbook)</td>
<td>Constructor with name and sheet ID</td>
</tr>
</tbody>
</table>

## See Also

Reference
- Worksheet Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet Constructor

Default Constructor

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Worksheet()
```

### See Also

**Reference**
- Worksheet Class
- Worksheet Overload
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet Constructor (String, Int32, Workbook)

Constructor with name and sheet ID

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

⚠️ Syntax

```csharp
public Worksheet(
    string name,
    int id,
    Workbook reference
)
```

Parameters

- **name**
  - Type: System.String
  - Name of the worksheet
- **id**
  - Type: System.Int32
  - ID of the worksheet (for internal use)
- **reference**
  - Type: NanoXLSXWorkbook
  - Reference to the parent Workbook

⚠️ See Also

Reference  
Worksheet Class
Worksheet Properties

The **Worksheet** type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AutoFilterRange</strong></td>
<td>Gets the range of the auto-filter. Wrapped to Nullable to provide null as value. If null, no auto-filter is applied.</td>
</tr>
<tr>
<td><strong>Cells</strong></td>
<td>Gets the cells of the worksheet as dictionary with the cell address as key and the cell object as value.</td>
</tr>
<tr>
<td><strong>Columns</strong></td>
<td>Gets all columns with non-standard properties, like auto filter applied or a special width as dictionary with the zero-based column index as key and the column object as value.</td>
</tr>
<tr>
<td><strong>CurrentCellDirection</strong></td>
<td>Gets or sets the direction when using AddNextCell method.</td>
</tr>
<tr>
<td><strong>DefaultColumnWidth</strong></td>
<td>Gets or sets the default column width.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DefaultRowHeight</td>
<td>Gets or sets the default Row height</td>
</tr>
<tr>
<td>HiddenRows</td>
<td>Gets the hidden rows as dictionary with the zero-based row number as key and a boolean as value. True indicates hidden, false visible.</td>
</tr>
<tr>
<td>MergedCells</td>
<td>Gets the merged cells (only references) as dictionary with the cell address as key and the range object as value</td>
</tr>
<tr>
<td>RowHeights</td>
<td>Gets defined row heights as dictionary with the zero-based row number as key and the height (float from 0 to 409.5) as value</td>
</tr>
<tr>
<td>SelectedCells</td>
<td>Gets the cell range of selected cells of this worksheet. Null if no cells are selected</td>
</tr>
<tr>
<td>SheetID</td>
<td>Gets or sets the internal ID of the sheet</td>
</tr>
<tr>
<td>SheetName</td>
<td>Gets or sets the name of the worksheet</td>
</tr>
<tr>
<td>SheetProtectionPassword</td>
<td>Gets the password</td>
</tr>
<tr>
<td><strong>SheetProtectionValues</strong></td>
<td>Gets the list of SheetProtectionValues. These values define the allowed actions if the worksheet is protected</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>UseSheetProtection</strong></td>
<td>Gets or sets whether the worksheet is protected. If true, protection is enabled</td>
</tr>
<tr>
<td><strong>WorkbookReference</strong></td>
<td>Gets or sets the Reference to the parent Workbook</td>
</tr>
</tbody>
</table>

**See Also**

- Reference
- [Worksheet Class](#)
- [NanoXLSX Namespace](#)

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
WorksheetAutoFilterRange Property

Gets the range of the auto-filter. Wrapped to Nullable to provide null as value. If null, no auto-filter is applied

**Namespace:** NanoXLXS
**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Nullable<CellRange> AutoFilterRange { get; }
```

**Property Value**
Type: Nullable<CellRange>

### See Also

Reference
- Worksheet Class
- NanoXLXS Namespace
WorksheetCells Property

Gets the cells of the worksheet as dictionary with the cell address as key and the cell object as value

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public Dictionary<string, Cell> Cells { get; }
```

**Property Value**

Type: `Dictionary<String, Cell>`

### See Also

**Reference**

- Worksheet Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetColumns Property

Gets all columns with non-standard properties, like auto filter applied or a special width as dictionary with the zero-based column index as key and the column object as value.

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Dictionary<int, WorksheetColumn> Columns
```

**Property Value**  
Type: `Dictionary<int, WorksheetColumn>`

### See Also

- **Reference**  
  - Worksheet Class  
  - NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetCurrentCellDirection

Property

Gets or sets the direction when using AddNextCell method

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

C#

```csharp
public WorksheetCellDirection CurrentCellDirection
```

Property Value

Type: WorksheetCellDirection

See Also

Reference

Worksheet Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetDefaultColumnWidth Property

Gets or sets the default column width

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public float DefaultColumnWidth { get; set; }
```

### Property Value

**Type:** Single

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Throws a ArgumentException exception if the passed width is out of range (set)</td>
</tr>
</tbody>
</table>

### See Also

Reference  
**Worksheet Class**  
**NanoXLSX Namespace**

Copyright Raphael Stoeckli © 2018
WorksheetDefaultRowHeight Property

Gets or sets the default Row height

**Namespace:** NanoXLXS
**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public float DefaultRowHeight { get; set; }
```

### Property Value

Type: **Single**

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws a RangeException exception if the passed height is out of range (set)</td>
</tr>
</tbody>
</table>

### See Also

Reference

- Worksheet Class
- NanoXLXS Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
WorksheetHiddenRows Property

Gets the hidden rows as dictionary with the zero-based row number as key and a boolean as value. True indicates hidden, false visible.

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public Dictionary<int, bool> HiddenRows { get; }
```

Property Value
Type: Dictionary<Int32, Boolean>

Remarks
Entries with the value false are not affecting the worksheet. These entries can be removed

See Also

Reference
Worksheet Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetMergedCells Property

Gets the merged cells (only references) as dictionary with the cell address as key and the range object as value

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public Dictionary<string, CellRange> MergedCells
```

**Property Value**  
Type: `Dictionary<string, CellRange>`

### See Also

- Reference  
  Worksheet Class  
  NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetRowHeights Property

Gets defined row heights as dictionary with the zero-based row number as key and the height (float from 0 to 409.5) as value

Namespace: NanoXLSX  
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```
public Dictionary<int, float> RowHeights { get; }
```

Property Value
Type: DictionaryInt32, Single

See Also

Reference
Worksheet Class  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
WorksheetSelectedCells Property

Gets the cell range of selected cells of this worksheet. Null if no cells are selected

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Nullable<CellRange> SelectedCells { get; }
```

**Property Value**

Type: NullableCellRange

### See Also

Reference

Worksheet Class  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetSheetID Property

Gets or sets the internal ID of the sheet

**Namespace:** NanoLSX
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public int SheetID { get; set; }
```

### Property Value

Type: Int32

### See Also

Reference
- Worksheet Class
- NanoLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetSheetName Property

Gets or sets the name of the worksheet

**Namespace:** NanoXLSPX

**Assembly:** NanoXLSPX (in NanoXLSPX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public string SheetName { get; set; }
```

### Property Value

Type: String

### See Also

Reference

- Worksheet Class
- NanoXLSPX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetSheetProtectionPassword Property

Gets the password used for sheet protection

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string SheetProtectionPassword { get; }
```

Property Value  
Type: String

### See Also

Reference  
Worksheet Class  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetSheetProtectionValues Property

Gets the list of SheetProtectionValues. These values define the allowed actions if the worksheet is protected

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public List<WorksheetSheetProtectionValue> SheetProtectionValues
```

### Property Value

Type: `List<WorksheetSheetProtectionValue>`

## See Also

- Reference: [Worksheet Class](#)
- Namespace: [NanoXLSX Namespace](#)

Copyright Raphael Stoeckli © 2018
WorksheetUseSheetProtection Property

Gets or sets whether the worksheet is protected. If true, protection is enabled.

**Namespace:** NanoXLX
**Assembly:** NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public bool UseSheetProtection { get; set; }
```

**Property Value**

Type: Boolean

**See Also**

Reference
- Worksheet Class
- NanoXLX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetWorkbookReference Property

Gets or sets the Reference to the parent Workbook

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public Workbook WorkbookReference { get; set; }
```

**Property Value**  
Type: Workbook

### See Also

- Reference  
  - Worksheet Class  
  - NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet Methods

The **Worksheet** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddAllowedActionOnSheetProtection</strong></td>
<td>Method to add allowed actions if the worksheet is protected. If one or more values are added, UseSheetProtection will be set to true</td>
</tr>
<tr>
<td><strong>AddCell(Object, String)</strong></td>
<td>Adds an object to the defined cell address. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted</td>
</tr>
<tr>
<td><strong>AddCell(Object, Int32, Int32)</strong></td>
<td>Adds an object to the defined cell address. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted</td>
</tr>
</tbody>
</table>
of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddCell(Object, String, Style)</td>
<td>Adds an object to the defined cell address. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted</td>
</tr>
<tr>
<td>AddCell(Object, Int32, Int32, Style)</td>
<td>Adds an object to the defined cell address. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted</td>
</tr>
<tr>
<td>AddCellFormula(String, String)</td>
<td>Adds a cell formula as string to the defined cell address</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>AddCellFormula(String, Int32, Int32)</strong></td>
<td>Adds a cell formula as string to the defined cell address</td>
</tr>
<tr>
<td><strong>AddCellFormula(String, String, Style)</strong></td>
<td>Adds a cell formula as string to the defined cell address</td>
</tr>
<tr>
<td><strong>AddCellFormula(String, Int32, Int32, Style)</strong></td>
<td>Adds a cell formula as string to the defined cell address</td>
</tr>
<tr>
<td><strong>AddCellRange(ListObject, String)</strong></td>
<td>Adds a list of object values to a defined cell range. If the type of the a particular value does not match with one of the supported data types, it will be casted to a String. Prepared objects of the type Cell will not be casted but adjusted</td>
</tr>
<tr>
<td><strong>AddCellRange(ListObject, CellAddress, CellAddress)</strong></td>
<td>Adds a list of object values to a defined cell range. If the type of the a particular value does not match with one of the supported data types, it will be casted to a String. Prepared objects of the type Cell will not be casted but adjusted</td>
</tr>
</tbody>
</table>
### AddCellRange(ListObject, String, Style)

Add a list of object values to a defined cell range. If the type of the a particular value does not match with one of the supported data types, it will be casted to a String. Prepared objects of the type Cell will not be casted but adjusted.

<table>
<thead>
<tr>
<th>AddCellRange(ListObject, CellAddress, CellAddress, Style)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adds a list of object values to a defined cell range. If the type of the a particular value does not match with one of the supported data types, it will be casted to a String. Prepared objects of the type Cell will not be casted but adjusted.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AddCellRangeInternalT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal function to add a generic list of value to the defined</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>AddHiddenColumn(Int32)</td>
</tr>
<tr>
<td>AddHiddenColumn(String)</td>
</tr>
<tr>
<td>AddHiddenRow</td>
</tr>
<tr>
<td>AddNextCell(Object)</td>
</tr>
<tr>
<td>AddNextCell(Object, Style)</td>
</tr>
<tr>
<td>AddNextCell(Cell, Boolean, Style)</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>AddNextCellFormula(String)</td>
</tr>
<tr>
<td>AddNextCellFormula(String, Style)</td>
</tr>
<tr>
<td>CastValue</td>
</tr>
<tr>
<td>ClearActiveStyle</td>
</tr>
<tr>
<td>Equals</td>
</tr>
<tr>
<td>Finalize</td>
</tr>
</tbody>
</table>
reclaimed by garbage collection. (Inherited from **Object**.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetCell(CellAddress)</strong></td>
<td>Gets the cell of the specified address</td>
</tr>
<tr>
<td><strong>GetCell(Int32, Int32)</strong></td>
<td>Gets the cell of the specified column and row number (zero-based)</td>
</tr>
<tr>
<td><strong>GetCurrentColumnNumber</strong></td>
<td>Gets the current column number (zero based)</td>
</tr>
<tr>
<td><strong>GetCurrentRowNumber</strong></td>
<td>Gets the current row number (zero based)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetLastAddress</strong></td>
<td>Gets the last existing row or column number of the current worksheet (zero-based)</td>
</tr>
<tr>
<td><strong>GetLastColumnNumber</strong></td>
<td>Gets the last existing column number in the current worksheet</td>
</tr>
</tbody>
</table>
### Function Descriptions

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetLastRowNumber</strong></td>
<td>Gets the last existing row number in the current worksheet (zero-based)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>)</td>
</tr>
<tr>
<td><strong>GoToNextColumn</strong></td>
<td>Moves the current position to the next column</td>
</tr>
<tr>
<td><strong>GoToNextColumn(Int32)</strong></td>
<td>Moves the current position to the next column with the number of cells to move</td>
</tr>
<tr>
<td><strong>GoToNextRow</strong></td>
<td>Moves the current position to the next row (use for a new line)</td>
</tr>
<tr>
<td><strong>GoToNextRow(Int32)</strong></td>
<td>Moves the current position to the next row with the number of cells to move (use for a new line)</td>
</tr>
<tr>
<td><strong>HasCell(CellAddress)</strong></td>
<td>Gets whether the specified address</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HasCell(Int32, Int32)</td>
<td>Gets whether the specified address exists in the worksheet. Existing means that a value was stored at the address.</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>MergeCells(String)</td>
<td>Merges the defined cell range</td>
</tr>
<tr>
<td>MergeCells(CellRange)</td>
<td>Merges the defined cell range</td>
</tr>
<tr>
<td>MergeCells(CellAddress, CellAddress)</td>
<td>Merges the defined cell range</td>
</tr>
<tr>
<td>RecalculateAutoFilter</td>
<td>Method to recalculate the auto filter (columns) of this worksheet. This is an internal method. There is no need to use it. It must be public to require access from</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RecalculateColumns</td>
<td>Method to recalculate the collection of columns of this worksheet. This is an internal method. There is no need to use it. It must be public to require access from the LowLevel class.</td>
</tr>
<tr>
<td>RemoveAutoFilter</td>
<td>Removes auto filters from the worksheet.</td>
</tr>
<tr>
<td>RemoveCell(String)</td>
<td>Removes a previous inserted cell at the defined address.</td>
</tr>
<tr>
<td>RemoveCell(Int32, Int32)</td>
<td>Removes a previous inserted cell at the defined address.</td>
</tr>
<tr>
<td>RemoveHiddenColumn(Int32)</td>
<td>Sets a previously defined, hidden column as visible again.</td>
</tr>
<tr>
<td>RemoveHiddenColumn(String)</td>
<td>Sets a previously defined, hidden column as visible again.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RemoveHiddenRow</td>
<td>Sets a previously defined, hidden row as visible again</td>
</tr>
<tr>
<td>RemoveMergedCells</td>
<td>Removes the defined merged cell range</td>
</tr>
<tr>
<td>RemoveSelectedCells</td>
<td>Removes the cell selection of this worksheet</td>
</tr>
<tr>
<td>SanitizeWorksheetName</td>
<td>Sanitizes a worksheet name</td>
</tr>
<tr>
<td>SetActiveStyle</td>
<td>Sets the active style of the worksheet. This style will be assigned to all later added cells</td>
</tr>
<tr>
<td>SetAutoFilter(String)</td>
<td>Sets the column auto filter within the defined column range</td>
</tr>
<tr>
<td>SetAutoFilter(Int32, Int32)</td>
<td>Sets the column auto filter within the defined column range</td>
</tr>
<tr>
<td>SetColumnHiddenState</td>
<td>Sets the defined column as hidden or visible</td>
</tr>
<tr>
<td>SetColumnWidth(Int32, Single)</td>
<td>Sets the width of the passed column number (zero-</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SetColumnWidth(String, Single)</td>
<td>Sets the width of the passed column address</td>
</tr>
<tr>
<td>SetCurrentCellAddress(String)</td>
<td>Set the current cell address</td>
</tr>
<tr>
<td>SetCurrentCellAddress(Int32, Int32)</td>
<td>Set the current cell address</td>
</tr>
<tr>
<td>SetCurrentColumnNumber</td>
<td>Sets the current column number (zero based)</td>
</tr>
<tr>
<td>SetCurrentRowNumber</td>
<td>Sets the current row number (zero based)</td>
</tr>
<tr>
<td>SetRowHeight</td>
<td>Sets the height of the passed row number (zero-based)</td>
</tr>
<tr>
<td>SetRowHiddenState</td>
<td>Sets the defined row as hidden or visible</td>
</tr>
<tr>
<td>SetSelectedCells(String)</td>
<td>Sets the selected cells on this worksheet</td>
</tr>
<tr>
<td>SetSelectedCells(CellRange)</td>
<td>Sets the selected cells on this worksheet</td>
</tr>
<tr>
<td>SetSelectedCells(CellAddress,</td>
<td>Sets the selected</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>CellAddress</code></td>
<td>cells on this worksheet</td>
</tr>
<tr>
<td><code>SetSheetName</code></td>
<td>Validates and sets the worksheet name</td>
</tr>
<tr>
<td><code>SetSheetName</code></td>
<td>Sets the name of the sheet</td>
</tr>
<tr>
<td><code>SetSheetProtectionPassword</code></td>
<td>Sets or removes the password for worksheet protection. If set, UseSheetProtection will be also set to true</td>
</tr>
<tr>
<td><code>ToString</code></td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td><code>WorksheetExists</code></td>
<td>Checks whether a worksheet with the given name exists</td>
</tr>
</tbody>
</table>

### See Also

**Reference**
- **Worksheet Class**
- **NanoXLSX Namespace**

Copyright Raphael Stoeckli © 2018
WorksheetAddAllowedActionOnSheetProtection Method

Method to add allowed actions if the worksheet is protected. If one or more values are added, UseSheetProtection will be set to true.

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void AddAllowedActionOnSheetProtection(
    WorksheetSheetProtectionValue typeOfProtection
)
```

### Parameters

- **typeOfProtection**  
  Type: NanoXLSXWorksheetSheetProtectionValue  
  Allowed action on the worksheet or cells

### See Also

- Reference  
  - Worksheet Class  
  - NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
# Worksheet AddCell Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddCell(Object, String)</td>
<td>Adds an object to the defined cell address. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted</td>
</tr>
<tr>
<td>AddCell(Object, Int32, Int32)</td>
<td>Adds an object to the defined cell address. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted</td>
</tr>
<tr>
<td>AddCell(Object, String, Style)</td>
<td>Adds an object to the defined cell address. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted</td>
</tr>
<tr>
<td>AddCell(Object, Int32, Int32, Style)</td>
<td>Adds an object to the defined cell address. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted</td>
</tr>
</tbody>
</table>
prepared object of the type Cell will not be casted but adjusted

See Also

Reference

Worksheet Class

NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetAddCell Method (Object, String)

Adds an object to the defined cell address. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void AddCell(
    Object value,
    string address
)
```

### Parameters

- **value**  
  Type: SystemObject  
  Unspecified value to insert

- **address**  
  Type: SystemString  
  Cell address in the format A1 - XFD1048576

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws an UndefinedStyleException if the</td>
</tr>
</tbody>
</table>
active style cannot be referenced while creating the cell

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RangeException</strong></td>
<td>Throws a RangeException if the passed cell address is out of range</td>
</tr>
<tr>
<td><strong>FormatException</strong></td>
<td>Throws a FormatException if the passed cell address is malformed</td>
</tr>
</tbody>
</table>

**Remarks**

Recognized are the following data types: Cell (prepared object), string, int, double, float, long, DateTime, bool. All other types will be casted into a string using the default ToString() method.

**See Also**

Reference
Worksheet Class
AddCell Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet AddCell Method (Object, Int32, Int32)

Adds an object to the defined cell address. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted.

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void AddCell(  
    Object value,  
    int columnAddress,  
    int rowAddress
)
```

### Parameters

- **value**
  - Type: `SystemObject`
  - Unspecified value to insert
- **columnAddress**
  - Type: `SystemInt32`
  - Column number (zero based)
- **rowAddress**
  - Type: `SystemInt32`
  - Row number (zero based)

### Exceptions
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws an UndefinedStyleException if the active style cannot be referenced while creating the cell</td>
</tr>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the passed cell address is out of range</td>
</tr>
</tbody>
</table>

Remarks

Recognized are the following data types: Cell (prepared object), string, int, double, float, long, DateTime, bool. All other types will be casted into a string using the default ToString() method

See Also

Reference
Worksheet Class
AddCell Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet AddCell Method
(Object, String, Style)

Adds an object to the defined cell address. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted.

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public void AddCell(
    Object value,
    string address,
    Style style
)
```

### Parameters

- **value**  
  - Type: `SystemObject`  
  - Unspecified value to insert

- **address**  
  - Type: `SystemString`  
  - Cell address in the format A1 - XFD1048576

- **style**  
  - Type: `NanoXLSX.StyleStyle`  
  - Style to apply on the cell

###Exceptions
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws an UndefinedStyleException if the passed style is malformed</td>
</tr>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the passed cell address is out of range</td>
</tr>
<tr>
<td>FormatException</td>
<td>Throws a FormatException if the passed cell address is malformed</td>
</tr>
</tbody>
</table>

**Remarks**

Recognized are the following data types: Cell (prepared object), string, int, double, float, long, DateTime, bool. All other types will be casted into a string using the default ToString() method.

**See Also**

Reference
- Worksheet Class
- AddCell Overload
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetAddCell Method
(Object, Int32, Int32, Style)

Adds an object to the defined cell address. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted

**Namespace:** NanoXL SX
**Assembly:** NanoXL SX (in NanoXL SX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```c#
public void AddCell(
    Object value,
    int columnAddress,
    int rowAddress,
    Style style
)
```

**Parameters**

*value*
  - Type: **SystemObject**
  - Unspecified value to insert

*columnAddress*
  - Type: **SystemInt32**
  - Column number (zero based)

*rowAddress*
  - Type: **SystemInt32**
  - Row number (zero based)

*style*
Type: NanoXLSX.StyleStyle
Style to apply on the cell

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws an UndefinedStyleException if the passed style is malformed</td>
</tr>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the passed cell address is out of range</td>
</tr>
</tbody>
</table>

Remarks

Recognized are the following data types: Cell (prepared object), string, int, double, float, long, DateTime, bool. All other types will be casted into a string using the default ToString() method.

See Also

Reference
Worksheet Class
AddCell Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
# WorksheetAddCellFormula Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddCellFormula(String, String)</td>
<td>Adds a cell formula as string to the defined cell address</td>
</tr>
<tr>
<td>AddCellFormula(String, Int32, Int32)</td>
<td>Adds a cell formula as string to the defined cell address</td>
</tr>
<tr>
<td>AddCellFormula(String, String, Style)</td>
<td>Adds a cell formula as string to the defined cell address</td>
</tr>
<tr>
<td>AddCellFormula(String, Int32, Int32, Style)</td>
<td>Adds a cell formula as string to the defined cell address</td>
</tr>
</tbody>
</table>

## See Also

- Reference
- Worksheet Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetAddCellFormula Method (String, String)

Adds a cell formula as string to the defined cell address

**Namespace:** NanoXLXS
**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void AddCellFormula(
    string formula,
    string address
)
```

### Parameters

- **formula**
  - Type: `System.String`
  - Formula to insert

- **address**
  - Type: `System.String`
  - Cell address in the format A1 - XFD1048576

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StyleException</code></td>
<td>Throws an UndefinedStyleException if the active style cannot be referenced while creating the cell</td>
</tr>
<tr>
<td><code>RangeException</code></td>
<td>Throws an RangeException if the passed</td>
</tr>
</tbody>
</table>
cell address is out of range

FormatException

Throws a FormatException if the passed cell address is malformed

See Also

Reference
Worksheet Class
AddCellFormula Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetAddCellFormula Method (String, Int32, Int32)

Adds a cell formula as string to the defined cell address

**Namespace:** NanoXL SX  
**Assembly:** NanoXL SX (in NanoXL SX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void AddCellFormula(
    string formula,
    int columnAddress,
    int rowAddress
)
```

### Parameters

- **formula**
  - Type: **SystemString**
  - Formula to insert

- **columnAddress**
  - Type: **SystemInt32**
  - Column number (zero based)

- **rowAddress**
  - Type: **SystemInt32**
  - Row number (zero based)

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws an UndefinedStyleException if the active style cannot be referenced while creating the cell</td>
</tr>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the passed cell address is out of range</td>
</tr>
</tbody>
</table>

**See Also**

Reference

- Worksheet Class
- AddCellFormula Overload
- NanoXLXSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetAddCellFormula Method (String, String, Style)

Adds a cell formula as string to the defined cell address

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

▲ Syntax

```csharp
public void AddCellFormula(
    string formula,
    string address,
    Style style
)
```

**Parameters**

*formula*  
Type: SystemString  
Formula to insert

*address*  
Type: SystemString  
Cell address in the format A1 - XFD1048576

*style*  
Type: NanoXLSX.StyleStyle  
Style to apply on the cell

▲ Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
</table>

(Copy)
<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws an UndefinedStyleException if the passed style was malformed</td>
</tr>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the passed cell address is out of range</td>
</tr>
<tr>
<td>FormatException</td>
<td>Throws a FormatException if the passed cell address is malformed</td>
</tr>
</tbody>
</table>

See Also

Reference
- Worksheet Class
- AddCellFormula Overload
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetAddCellFormula Method (String, Int32, Int32, Style)

Adds a cell formula as string to the defined cell address

**Namespace:** NanoXLXS
**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void AddCellFormula(
    string formula,
    int columnAddress,
    int rowAddress,
    Style style
)
```

**Parameters**

- **formula**
  - Type: System.String
  - Formula to insert

- **columnAddress**
  - Type: System.Int32
  - Column number (zero based)

- **rowAddress**
  - Type: System.Int32
  - Row number (zero based)

- **style**
  - Type: NanoXLXS.Style
"
Style to apply on the cell

## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StyleException</strong></td>
<td>Throws an UndefinedStyleException if the active style cannot be referenced while creating the cell</td>
</tr>
<tr>
<td><strong>RangeException</strong></td>
<td>Throws an RangeException if the passed cell address is out of range</td>
</tr>
</tbody>
</table>

## See Also

Reference
- Worksheet Class
- AddCellFormula Overload
- NanoXLSX Namespace

Copyright Raphaël Stoeckli © 2018
WorksheetAddCellRange Method

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddCellRange(ListObject, String)</td>
<td>Adds a list of object values to a defined cell range. If the type of the a particular value does not match with one of the supported data types, it will be casted to a String. Prepared objects of the type Cell will not be casted but adjusted</td>
</tr>
<tr>
<td>AddCellRange(ListObject, CellAddress, CellAddress)</td>
<td>Adds a list of object values to a defined cell range. If the type of the a particular value does not match with one of the supported data types, it will be casted to a String. Prepared objects of the type Cell will not be casted but adjusted</td>
</tr>
</tbody>
</table>
AddCellRange(ListObject, CellAddress, CellAddress, Style)

Adds a list of object values to a defined cell range. If the type of the a particular value does not match with one of the supported data types, it will be casted to a String. Prepared objects of the type Cell will not be casted but adjusted.
WorksheetAddCellRange Method (ListObject, String)

Adds a list of object values to a defined cell range. If the type of the a particular value does not match with one of the supported data types, it will be casted to a String. Prepared objects of the type Cell will not be casted but adjusted

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public void AddCellRange(
    List<Object> values,
    string cellRange
)
```

Parameters

**values**
Type: System.Collections.Generic.ListObject
List of unspecified objects to insert

**cellRange**
Type: System.String
Cell range as string in the format like A1:D1 or X10:X22

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the number of</td>
</tr>
</tbody>
</table>


cells resolved from the range differs from the number of passed values

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StyleException</strong></td>
<td>Throws an UndefinedStyleException if the active style cannot be referenced while creating the cells</td>
</tr>
<tr>
<td><strong>FormatException</strong></td>
<td>Throws a FormatException if the passed cell range is malformed</td>
</tr>
</tbody>
</table>

**Remarks**

The data types in the passed list can be mixed. Recognized are the following data types: Cell (prepared object), string, int, double, float, long, DateTime, bool. All other types will be casted into a string using the default ToString() method

**See Also**

Reference
Worksheet Class
AddCellRange Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetAddCellRange Method (ListObject, CellAddress, CellAddress)

Adds a list of object values to a defined cell range. If the type of the a particular value does not match with one of the supported data types, it will be casted to a String. Prepared objects of the type Cell will not be casted but adjusted

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void AddCellRange(
    List<object> values,
    NanoXLSX.CellAddress startAddress,
    NanoXLSX.CellAddress endAddress
)
```

### Parameters

**values**
- Type: `System.Collections.Generic.ListObject`  
  - List of unspecified objects to insert

**startAddress**
- Type: `NanoXLSX.CellAddress`  
  - Start address

**endAddress**
- Type: `NanoXLSX.CellAddress`  
  - End address
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RangeException</code></td>
<td>Throws an <code>RangeException</code> if the number of cells resolved from the range differs from the number of passed values</td>
</tr>
<tr>
<td><code>StyleException</code></td>
<td>Throws an <code>UndefinedStyleException</code> if the active style cannot be referenced while creating the cells</td>
</tr>
</tbody>
</table>

## Remarks

The data types in the passed list can be mixed. Recognized are the following data types: string, int, double, float, long, DateTime, bool. All other types will be casted into a string using the default `ToString()` method.

## See Also

Reference

- [Worksheet Class](#)
- [AddCellRange Overload](#)
- [NanoXLSX Namespace](#)
WorksheetAddCellRange
Method (ListObject, String, Style)

Adds a list of object values to a defined cell range. If the type of the a particular value does not match with one of the supported data types, it will be casted to a String. Prepared objects of the type Cell will not be casted but adjusted

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public void AddCellRange(
    List<object> values,
    string cellRange,
    Style style
)
```

Parameters

values
Type: System.Collections.Generic.ListObject
List of unspecified objects to insert

cellRange
Type: System.String
Cell range as string in the format like A1:D1 or X10:X22

style
Type: NanoXLSX.StyleStyle
Style to apply on the all cells of the range

Exceptions
### Exception | Condition
--- | ---
RangeException | Throws an RangeException if the number of cells resolved from the range differs from the number of passed values

StyleException | Throws an UndefinedStyleException if the passed style is malformed

FormatException | Throws a FormatException if the passed cell range is malformed

### Remarks

The data types in the passed list can be mixed. Recognized are the following data types: Cell (prepared object), string, int, double, float, long, DateTime, bool. All other types will be casted into a string using the default ToString() method

### See Also

Reference
- Worksheet Class
- AddCellRange Overload
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetAddCellRange Method (ListObject, CellAddress, CellAddress, Style)

Adds a list of object values to a defined cell range. If the type of the a particular value does not match with one of the supported data types, it will be casted to a String. Prepared objects of the type Cell will not be casted but adjusted

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void AddCellRange(
    List<Object> values,
    CellAddress startAddress,
    CellAddress endAddress,
    Style style
)
```

### Parameters

- **values**
  Type: System.Collections.Generic.ListObject  
  List of unspecified objects to insert
- **startAddress**
  Type: NanoXLSXCellAddress  
  Start address
- **endAddress**
  Type: NanoXLSXCellAddress
End address

**style**

Type: `NanoXLSX.StyleStyle`

Style to apply on the all cells of the range

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RangeException</strong></td>
<td>Throws an <code>RangeException</code> if the number of cells resolved from the range differs from the number of passed values</td>
</tr>
<tr>
<td><strong>StyleException</strong></td>
<td>Throws an <code>UndefinedStyleException</code> if the passed style is malformed</td>
</tr>
</tbody>
</table>

### Remarks

The data types in the passed list can be mixed. Recognized are the following data types: Cell (prepared object), string, int, double, float, long, DateTime, bool. All other types will be casted into a string using the default `ToString()` method

### See Also

- Reference
  - Worksheet Class
  - AddCellRange Overload
  - NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetAddCellRangeInternal<T> Method

Internal function to add a generic list of value to the defined cell range

**Namespace:** NanoXLSSX  
**Assembly:** NanoXLSSX (in NanoXLSSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private void AddCellRangeInternal<T>(
    List<T> values,
    CellAddress startAddress,
    CellAddress endAddress,
    Style style
)
```

### Parameters

- **values**
  - Type: System.Collections.Generic.List<T>
  - List of values

- **startAddress**
  - Type: NanoXLSSX.CellAddress
  - Start address

- **endAddress**
  - Type: NanoXLSSX.CellAddress
  - End address

- **style**
  - Type: NanoXLSSX.Style
  - Style to apply on the all cells of the range
Type Parameters

$T$

Data type of the generic value list

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RangeException</code></td>
<td>Throws an <code>RangeException</code> if the number of cells differs from the number of passed values</td>
</tr>
<tr>
<td><code>StyleException</code></td>
<td>Throws an <code>StyleException</code> if the active style cannot be referenced while creating the cells</td>
</tr>
</tbody>
</table>

Remarks

The data types in the passed list can be mixed. Recognized are the following data types: Cell (prepared object), string, int, double, float, long, DateTime, bool. All other types will be casted into a string using the default `ToString()` method

See Also

Reference
Worksheet Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetAddHiddenColumn Method

▲ Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddHiddenColumn(Int32)</td>
<td>Sets the defined column as hidden</td>
</tr>
<tr>
<td>AddHiddenColumn(String)</td>
<td>Sets the defined column as hidden</td>
</tr>
</tbody>
</table>

▲ See Also

Reference
Worksheet Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetAddHiddenColumn Method (Int32)

Sets the defined column as hidden

**Namespace:**  NanoXLSX  
**Assembly:**  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void AddHiddenColumn(
    int columnNumber
)
```

### Parameters

- **columnName**
  - Type: `System.Int32`
  - Column number to hide on the worksheet

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RangeException</code></td>
<td>Throws a RangeException if the passed column number is out of range</td>
</tr>
</tbody>
</table>

### See Also

- Reference
  - Worksheet Class
  - AddHiddenColumn Overload
Worksheet AddHiddenColumn Method (String)

Sets the defined column as hidden

**Namespace:** NanoXSX  
**Assembly:** NanoXSX (in NanoXSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void AddHiddenColumn(
    string columnAddress
)
```

### Parameters

- **columnAddress**
  - Type: `System.String`
  - Column address to hide on the worksheet

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RangeException</code></td>
<td>Throws an <code>RangeException</code> if the passed column address is out of range</td>
</tr>
</tbody>
</table>

### See Also

Reference
- Worksheet Class
- AddHiddenColumn Overload
WorksheetAddHiddenRow Method

Sets the defined row as hidden

Namespace: NanoXLXS
Assembly: NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public void AddHiddenRow(int rowNumber)
```

Parameters

`rowNumber`  
Type: SystemInt32  
Row number to hide on the worksheet

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the passed row number is out of range</td>
</tr>
</tbody>
</table>

See Also

Reference

Worksheet Class
NanoXLXS Namespace
## WorksheetAddNextCell Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddNextCell(Object)</td>
<td>Adds an object to the next cell position. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted</td>
</tr>
<tr>
<td>AddNextCell(Object, Style)</td>
<td>Adds an object to the next cell position. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted</td>
</tr>
<tr>
<td>AddNextCell(Cell, Boolean, Style)</td>
<td>Method to insert a generic cell to the next cell position</td>
</tr>
</tbody>
</table>

### See Also

**Reference**
- Worksheet Class
- NanoLSX Namespace
WorksheetAddNextCell Method (Object)

Adds an object to the next cell position. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public void AddNextCell(
    Object value
)
```

**Parameters**

- **value**  
  Type: SystemObject  
  Unspecified value to insert

**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws a RangeException if the next cell is out of range (on row or column)</td>
</tr>
</tbody>
</table>

**Remarks**
Recognized are the following data types: Cell (prepared object), string, int, double, float, long, DateTime, bool. All other types will be casted into a string using the default ToString() method

See Also

Reference
Worksheet Class
AddNextCell Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet AddNextCell Method (Object, Style)

Adds an object to the next cell position. If the type of the value does not match with one of the supported data types, it will be casted to a String. A prepared object of the type Cell will not be casted but adjusted

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void AddNextCell(
    Object value,
    Style style
)
```

### Parameters

**value**
- Type: `SystemObject`
- Unspecified value to insert

**style**
- Type: `NanoXLSX.StyleStyle`
- Style object to apply on this cell

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RangeException</code></td>
<td>Throws a <code>RangeException</code> if the next cell is</td>
</tr>
<tr>
<td>out of range (on row or column)</td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>StyleException</strong></td>
<td>Throws a StyleException if the default style was malformed</td>
</tr>
</tbody>
</table>

## Remarks

Recognized are the following data types: Cell (prepared object), string, int, double, float, long, DateTime, bool. All other types will be casted into a string using the default ToString() method

## See Also

Reference
- Worksheet Class
- AddNextCell Overload
- NanoXLSX Namespace
WorksheetAddNextCell Method (Cell, Boolean, Style)

Method to insert a generic cell to the next cell position

**Namespace:** NanoXLsx  
**Assembly:** NanoXLsx (in NanoXLsx.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
private void AddNextCell(
    Cell cell,
    bool incremental,
    Style style
)
```

### Parameters

- **cell**
  - Type: NanoXLsxCell  
  - Cell object to insert

- **incremental**
  - Type: SystemBoolean  
  - If true, the address value (row or column) will be incremented, otherwise not

- **style**
  - Type: NanoXLsx.StyleStyle  
  - If not null, the defined style will be applied to the cell, otherwise no style or the default style will be applied

### Exceptions
### Exception Condition

| StyleException | Throws a StyleException if the default style was malformed or if the active style cannot be referenced |

### Remarks

Recognized are the following data types: string, int, double, float, long, DateTime, bool. All other types will be casted into a string using the default ToString() method.

### See Also

Reference
- Worksheet Class
- AddNextCell Overload
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
# Worksheet>AddNextCellFormula Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AddNextCellFormula(String)</code></td>
<td>Adds a formula as string to the next cell position</td>
</tr>
<tr>
<td><code>AddNextCellFormula(String, Style)</code></td>
<td>Adds a formula as string to the next cell position</td>
</tr>
</tbody>
</table>

## See Also

Reference

- Worksheet Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetAddNextCellFormula Method (String)

Adds a formula as string to the next cell position

**Namespace:** NanoXLSSX  
**Assembly:** NanoXLSSX (in NanoXLSSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void AddNextCellFormula(
    string formula
)
```

### Parameters

*formula*

Type: System.String

Formula to insert

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StyleException</strong></td>
<td>Throws an UndefinedStyleException if the active style cannot be referenced while creating the cell</td>
</tr>
<tr>
<td><strong>RangeException</strong></td>
<td>Trows a RangeException if the next cell is out of range (on row or column)</td>
</tr>
</tbody>
</table>

### See Also
Reference
Worksheet Class
AddNextCellFormula Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet AddNextCellFormula Method (String, Style)

Adds a formula as string to the next cell position

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void AddNextCellFormula(
    string formula,
    Style style
)
```

### Parameters

- **formula**
  - Type: System.String
  - Formula to insert

- **style**
  - Type: NanoXLSX.Style
  - Style to apply on the cell

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws an UndefinedStyleException if the active style cannot be referenced while creating the cell</td>
</tr>
<tr>
<td>RangeException</td>
<td>Trows a RangeException if the next cell is</td>
</tr>
</tbody>
</table>
out of range (on row or column)

See Also

Reference
Worksheet Class
AddNextCellFormula Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetCastValue Method

Method to cast a value or align an object of the type Cell to the context of the worksheet

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private Cell CastValue(
    Object value,
    int column,
    int row
)
```

### Parameters

- **value**
  - Type: `SystemObject`
  - Unspecified value or object of the type Cell

- **column**
  - Type: `SystemInt32`
  - Column index

- **row**
  - Type: `SystemInt32`
  - Row index

### Return Value

- Type: `Cell`
  - Cell object

### See Also
Reference
Worksheet Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetClearActiveStyle Method

Clears the active style of the worksheet. All later added calls will contain no style unless another active style is set

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void ClearActiveStyle()
```

### See Also

Reference  
 Worksheet Class  
 NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet GetCell Method

➤ Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCell(CellAddress)</td>
<td>Gets the cell of the specified address</td>
</tr>
<tr>
<td>GetCell(Int32, Int32)</td>
<td>Gets the cell of the specified column and row number (zero-based)</td>
</tr>
</tbody>
</table>

➤ See Also

Reference
Worksheet Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetGetCell Method (CellAddress)

Gets the cell of the specified address

**Namespace:** NanoXL SX  
**Assembly:** NanoXL SX (in NanoXL SX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public Cell GetCell(
    CellAddress address
)
```

### Parameters

**address**
- Type: NanoXL SXCellAddress
- Address of the cell

### Return Value

**Type:** Cell  
Cell object

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorksheetException</td>
<td>Throws a WorksheetException if the cell was not found on the cell table of this worksheet</td>
</tr>
</tbody>
</table>
See Also

Reference

Worksheet Class
GetCell Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetGetCell Method (Int32, Int32)

Gets the cell of the specified column and row number (zero-based)

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public Cell GetCell(
    int columnNumber,
    int rowNumber
)
```

Parameters

- `columnNumber`
  - Type: System.Int32
  - Column number of the cell

- `rowNumber`
  - Type: System.Int32
  - Row number of the cell

Return Value

Type: Cell
Cell object

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorksheetException</td>
<td>Trows a WorksheetException if the cell</td>
</tr>
</tbody>
</table>
was not found on the cell table of this worksheet

See Also

Reference

Worksheet Class
GetCell Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetGetCurrentColumnNumber Method

Gets the current column number (zero based)

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public int GetCurrentColumnNumber()
```

### Return Value

Type: **Int32**  
Column number (zero-based)

### See Also

Reference  
Worksheet Class  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetGetCurrentRowNumber Method

Gets the current row number (zero based)

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public int GetCurrentRowNumber()
```

### Return Value

Type: **Int32**  
Row number (zero-based)

### See Also

**Reference**  
- Worksheet Class  
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetGetLastAddress Method

Gets the last existing row or column number of the current worksheet (zero-based)

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private int GetLastAddress(
    bool column
)
```

### Parameters

**column**
Type: SystemBoolean
If true, the output will be the last column, otherwise the last row

### Return Value
Type: Int32
Last row or column number (zero-based)

### See Also

**Reference**  
Worksheet Class  
NanoXLSX Namespace
WorksheetGetLastColumnNumber Method

Gets the last existing column number in the current worksheet (zero-based)

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public int GetLastColumnNumber()
```

**Return Value**

Type: `Int32`

Zero-based column number. In case of a empty worksheet, -1 will be returned

**See Also**

Reference

- Worksheet Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
WorksheetGetLastRowNumber Method

Gets the last existing row number in the current worksheet (zero-based)

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public int GetLastRowNumber()
```

### Return Value

Type: **Int32**  
Zero-based row number. In case of a empty worksheet, -1 will be returned

### See Also

*Reference*  
**Worksheet Class**  
**NanoXLSX Namespace**

Copyright Raphael Stoeckli © 2018
# WorksheetGoToNextColumn Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![icon] GoToNextColumn</td>
<td>Moves the current position to the next column</td>
</tr>
<tr>
<td>![icon] GoToNextColumn(Int32)</td>
<td>Moves the current position to the next column with the number of cells to move</td>
</tr>
</tbody>
</table>

## See Also

- Reference
  - Worksheet Class
  - NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetGoToNextColumn Method

Moves the current position to the next column

**Namespace:** NanoXLXS

**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public void GoToNextColumn()
```

### See Also

Reference

- Worksheet Class
- GoToNextColumn Overload
- NanoXLXS Namespace

Copyright Raphael Stoeckli © 2018
WorksheetGoToNextColumn Method (Int32)

Moves the current position to the next column with the number of cells to move

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public void GoToNextColumn(
    int numberOfColumns
)
```

**Parameters**

- `numberOfColumns`  
  **Type:** SystemInt32  
  Number of columns to move

### See Also

Reference

- Worksheet Class
- GoToNextColumn Overload
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet GoToNextRow Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GoToNextRow</td>
<td>Moves the current position to the next row (use for a new line)</td>
</tr>
<tr>
<td>GoToNextRow(Int32)</td>
<td>Moves the current position to the next row with the number of cells to move (use for a new line)</td>
</tr>
</tbody>
</table>

## See Also

Reference
- Worksheet Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
WorksheetGoToNextRow Method

Moves the current position to the next row (use for a new line)

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public void GoToNextRow()
```

## See Also

Reference
- Worksheet Class
- GoToNextRow Overload  
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet GoToNextRow Method (Int32)

Moves the current position to the next row with the number of cells to move (use for a new line)

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void GoToNextRow(int numberOfRows)
```

**Parameters**

`numberOfRows`  
Type: SystemInt32  
Number of rows to move

### See Also

- Worksheet Class  
- GoToNextRow Overload  
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
## Worksheet HasCell Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasCell(CellAddress)</td>
<td>Gets whether the specified address exists in the worksheet. Existing means that a value was stored at the address</td>
</tr>
<tr>
<td>HasCell(Int32, Int32)</td>
<td>Gets whether the specified address exists in the worksheet. Existing means that a value was stored at the address</td>
</tr>
</tbody>
</table>

### See Also

- **Reference**
  - Worksheet Class
  - NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
WorksheetHasCell Method (CellAddress)

 Gets whether the specified address exists in the worksheet. Existing means that a value was stored at the address

 **Namespace:** NanoXLSX  
 **Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

 ### Syntax

```csharp
public bool HasCell(
    CellAddress address
)
```

**Parameters**

`address`  
Type: NanoXLSXCellAddress  
Address to check

**Return Value**

Type: Boolean  
`true` if the cell exists, otherwise `false`.

### See Also

**Reference**  
Worksheet Class  
HasCell Overload  
NanoXLSX Namespace
WorksheetHasCell Method (Int32, Int32)

Gets whether the specified address exists in the worksheet. Existing means that a value was stored at the address

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public bool HasCell(
    int columnNumber,
    int rowNumber
)
```

**Parameters**

- **columnName**  
  Type: SystemInt32  
  Column number of the cell to check (zero-based)

- **rowNumber**  
  Type: SystemInt32  
  Row number of the cell to check (zero-based)

**Return Value**

Type: Boolean  
true if the cell exists, otherwise false.

**See Also**

Reference
Worksheet Class
HasCell Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetMergeCells Method

⚠️ Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MergeCells(String)</td>
<td>Merges the defined cell range</td>
</tr>
<tr>
<td>MergeCells(CellRange)</td>
<td>Merges the defined cell range</td>
</tr>
<tr>
<td>MergeCells(CellAddress, CellAddress)</td>
<td>Merges the defined cell range</td>
</tr>
</tbody>
</table>

See Also

Reference
Worksheet Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetMergeCells Method (String)

Merges the defined cell range

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string MergeCells(
    string cellRange
)
```

### Parameters

**cellRange**
- **Type:** System.String
- Range to merge (e.g. 'A1:B12')

### Return Value

**Type:** String  
Returns the validated range of the merged cells (e.g. 'A1:B12')

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the passed cell range is out of range</td>
</tr>
<tr>
<td>FormatException</td>
<td>Throws a FormatException if the passed cell range is malformed</td>
</tr>
</tbody>
</table>
See Also

Reference
Worksheet Class
MergeCells Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetMergeCells Method (CellRange)

Merges the defined cell range

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public string MergeCells(
    CellRange cellRange
)
```

### Parameters

- **cellRange**  
  - Type: NanoXLSXCellRange  
  - Range to merge

### Return Value

- **Type:** String  
  - Returns the validated range of the merged cells (e.g. 'A1:B12')

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the passed cell range is out of range</td>
</tr>
</tbody>
</table>

### See Also
Reference
Worksheet Class
MergeCells Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetMergeCells Method (CellAddress, CellAddress)

Merges the defined cell range

**Namespace:** NanoXLXS
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string MergeCells(
    CellAddress startAddress,
    CellAddress endAddress
)
```

### Parameters

- **startAddress**
  - Type: NanoXLSXCellAddress
  - Start address of the merged cell range

- **endAddress**
  - Type: NanoXLSXCellAddress
  - End address of the merged cell range

### Return Value

- Type: String
  - Returns the validated range of the merged cells (e.g. 'A1:B12')

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentNullException</td>
<td>Throws an ArgumentException if one of the</td>
</tr>
</tbody>
</table>
passed cell addresses is out of range

See Also

Reference
Worksheet Class
MergeCells Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetRecalculateAutoFilter Method

Method to recalculate the auto filter (columns) of this worksheet. This is an internal method. There is no need to use it. It must be public to require access from the LowLevel class

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
public void RecalculateAutoFilter()
```

See Also

Reference
  Worksheet Class
  NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetRecalculateColumns Method

Method to recalculate the collection of columns of this worksheet. This is an internal method. There is no need to use it. It must be public to require access from the LowLevel class

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public void RecalculateColumns()
```

See Also

Reference
Worksheet Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetRemoveAutoFilter Method

Removes auto filters from the worksheet

**Namespace:** NanoXLXS
**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```c#
public void RemoveAutoFilter()
```

**See Also**

Reference
- Worksheet Class
- NanoXLXS Namespace

Copyright Raphael Stoeckli © 2018
## Worksheet\RemoveCell Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoveCell(String)</td>
<td>Removes a previous inserted cell at the defined address</td>
</tr>
<tr>
<td>RemoveCell(Int32, Int32)</td>
<td>Removes a previous inserted cell at the defined address</td>
</tr>
</tbody>
</table>

### See Also

- Reference
  - Worksheet Class
  - NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetRemoveCell Method (String)

Removes a previous inserted cell at the defined address

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public bool RemoveCell(
    string address
)
```

### Parameters

*address*  
Type: `System.String`  
Cell address in the format A1 - XFD1048576

### Return Value

Type: `Boolean`  
Returns true if the cell could be removed (existed), otherwise false (did not exist)

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RangeException</code></td>
<td>Throws an <code>RangeException</code> if the passed cell address is out of range</td>
</tr>
<tr>
<td><code>FormatException</code></td>
<td>Throws a <code>FormatException</code> if the passed cell address is not valid</td>
</tr>
</tbody>
</table>


address is malformed

See Also

Reference
Worksheet Class
RemoveCell Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetRemoveCell Method (Int32, Int32)

Removes a previous inserted cell at the defined address

**Namespace:** NanoXL SX  
**Assembly:** NanoXL SX (in NanoXL SX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public bool RemoveCell(  
    int columnAddress,  
    int rowAddress
)
```

### Parameters

- **columnAddress**  
  Type: SystemInt32  
  Column number (zero based)

- **rowAddress**  
  Type: SystemInt32  
  Row number (zero based)

### Return Value

Type: Boolean  
Returns true if the cell could be removed (existed), otherwise false (did not exist)

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
</table>

**RangeException**  Throws an `RangeException` if the passed cell address is out of range

## See Also

Reference
- Worksheet Class
- RemoveCell Overload
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet RemoveHiddenColumn Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoveHiddenColumn(Int32)</td>
<td>Sets a previously defined, hidden column as visible again</td>
</tr>
<tr>
<td>RemoveHiddenColumn(String)</td>
<td>Sets a previously defined, hidden column as visible again</td>
</tr>
</tbody>
</table>

## See Also

Reference
 Worksheet Class
 NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetRemoveHiddenColumn Method (Int32)

Sets a previously defined, hidden column as visible again

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

**C#**

```csharp
public void RemoveHiddenColumn(
    int columnNumber
)
```

### Parameters

*columnNumber*
- Type: `System.Int32`
- Column number to make visible again

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Throws an ArgumentException if the passed column number is out of range</td>
</tr>
</tbody>
</table>

### See Also

**Reference**
- Worksheet Class
- RemoveHiddenColumn Overload
Worksheet RemoveHiddenColumn Method (String)

Sets a previously defined, hidden column as visible again

**Namespace:** NanoXLX
**Assembly:** NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

C#

```csharp
public void RemoveHiddenColumn(
    string columnAddress
)
```

### Parameters

- **columnAddress**
  - Type: System.String
  - Column address to make visible again

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the column address out of range</td>
</tr>
</tbody>
</table>

### See Also

- Reference
- Worksheet Class
- RemoveHiddenColumn Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetRemoveHiddenRow Method

Sets a previously defined, hidden row as visible again

**Namespace:** NanoXL SX  
**Assembly:** NanoXL SX (in NanoXL SX.dll) Version: 1.2.3.0 (1.2.3)

⚠️ Syntax

```
public void RemoveHiddenRow(
    int rowNumber
)
```

**Parameters**

*rowNumber*
  - Type: SystemInt32
  - Row number to hide on the worksheet

⚠️ Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the passed row number is out of range</td>
</tr>
</tbody>
</table>

⚠️ See Also

Reference

Worksheet Class  
NanoXL SX Namespace
WorksheetRemoveMergedCells Method

Removes the defined merged cell range

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

⚠️ Syntax

```csharp
public void RemoveMergedCells(
    string range
)
```

Parameters

*range*

   - Type: **System.String**
   - Cell range to remove the merging

⚠️ Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws a UnknownRangeException if the passed cell range was not merged earlier</td>
</tr>
</tbody>
</table>

⚠️ See Also

Reference

- Worksheet Class
- NanoXLSX Namespace
WorksheetRemoveSelectedCells Method

Removes the cell selection of this worksheet

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

```csharp
public void RemoveSelectedCells()
```

**See Also**

Reference  
Worksheet Class  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetSanitizeWorksheetName Method

Sanitizes a worksheet name

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static string SanitizeWorksheetName(
    string input,
    Workbook workbook
)
```

### Parameters

- **input**
  - Type: System.String  
  - Name to sanitize

- **workbook**
  - Type: NanoXLSXWorkbook  
  - Workbook reference

### Return Value

- Type: String  
  - Name of the sanitized worksheet

### See Also

- Reference: Worksheet Class
WorksheetSetActiveStyle Method

Sets the active style of the worksheet. This style will be assigned to all later added cells

Namespace: NanoXLX
Assembly: NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
public void SetActiveStyle(
    Style style
)
```

Parameters

*style*
Type: NanoXLX.Style
Style to set as active style

See Also

Reference
Worksheet Class
NanoXLX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet SetAutoFilter Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetAutoFilter(String)</td>
<td>Sets the column auto filter within the defined column range</td>
</tr>
<tr>
<td>SetAutoFilter(Int32, Int32)</td>
<td>Sets the column auto filter within the defined column range</td>
</tr>
</tbody>
</table>

See Also

Reference
Worksheet Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetSetAutoFilter Method (String)

Sets the column auto filter within the defined column range

**Namespace:**  NanoXLX
**Assembly:**  NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void SetAutoFilter(
    string range
)
```

### Parameters

**range**
Type: `System.String`
Range to apply auto filter on. The range could be 'A1:C10' for instance. The end row will be recalculated automatically when saving the file

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RangeException</code></td>
<td>Throws an <code>RangeException</code> if the passed range out of range</td>
</tr>
<tr>
<td><code>FormatException</code></td>
<td>Throws an <code>FormatException</code> if the passed range is malformed</td>
</tr>
</tbody>
</table>
See Also

Reference
- Worksheet Class
- SetAutoFilter Overload
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetSetAutoFilter Method (Int32, Int32)

Sets the column auto filter within the defined column range

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#  
public void SetAutoFilter(
    int startColumn,
    int endColumn
)
```

### Parameters

- **startColumn**  
  Type: `System.Int32`  
  Column number with the first appearance of an auto filter drop down  
- **endColumn**  
  Type: `System.Int32`  
  Column number with the last appearance of an auto filter drop down

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RangeException</strong></td>
<td>Throws an RangeException if the start or end address out of range</td>
</tr>
</tbody>
</table>

See Also

Reference

Worksheet Class
SetAutoFilter Overload
NanoXLSX Namespace
WorksheetSetColumnHiddenState Method

Sets the defined column as hidden or visible

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private void SetColumnHiddenState(
    int columnNumber,
    bool state
)
```

### Parameters

- **columnNumber**  
  Type: SystemInt32  
  Column number to hide on the worksheet

- **state**  
  Type: SystemBoolean  
  If true, the column will be hidden, otherwise be visible

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the column number out of range</td>
</tr>
</tbody>
</table>

### See Also
## WorksheetSetColumnWidth Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetColumnWidth(Int32, Single)</td>
<td>Sets the width of the passed column number (zero-based)</td>
</tr>
<tr>
<td>SetColumnWidth(String, Single)</td>
<td>Sets the width of the passed column address</td>
</tr>
</tbody>
</table>

### See Also

**Reference**
- Worksheet Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetSetColumnWidth Method (Int32, Single)

Sets the width of the passed column number (zero-based)

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void SetColumnWidth(
    int columnNumber,
    float width
)
```

### Parameters

- **columnNumber**
  - Type: `SystemInt32`
  - Column number (zero-based, from 0 to 16383)

- **width**
  - Type: `SystemSingle`
  - Width from 0 to 255.0

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
</table>
| `RangeException`| Throws an `RangeException`:
  a) If the passed column number is out of range
  b) If the column width is out of range (0 - |
See Also

Reference
Worksheet Class
SetColumnWidth Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
**WorksheetSetColumnWidth Method (String, Single)**

Sets the width of the passed column address

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void SetColumnWidth(  
    string columnAddress,  
    float width
)
```

#### Parameters

- **columnAddress**  
  Type: `System.String`  
  Column address (A - XFD)

- **width**  
  Type: `System.Single`  
  Width from 0 to 255.0

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
</table>
| `RangeException` | Throws an `RangeException`:
|                  | a) If the passed column address is out of range                            |
|                  | b) if the column width is out of range (0 - |
See Also

Reference
Worksheet Class
SetColumnWidth Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
# WorksheetSetCurrentCellAddress Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SetCurrentCellAddress(String)</code></td>
<td>Set the current cell address</td>
</tr>
<tr>
<td><code>SetCurrentCellAddress(Int32, Int32)</code></td>
<td>Set the current cell address</td>
</tr>
</tbody>
</table>

## See Also

Reference
- Worksheet Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetSetCurrentCellAddress Method (String)

Set the current cell address

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void SetCurrentCellAddress(
    string address
)
```

### Parameters

- **address**
  - Type: `System.String`  
  - Cell address in the format A1 - XFD1048576

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the passed cell address is out of range</td>
</tr>
<tr>
<td>FormatException</td>
<td>Throws a FormatException if the passed cell address is malformed</td>
</tr>
</tbody>
</table>

### See Also
Reference

Worksheet Class
SetCurrentCellAddress Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetSetCurrentCellAddress Method (Int32, Int32)

Set the current cell address

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
c# Copy

public void SetCurrentCellAddress(
    int columnAddress,
    int rowAddress
)
```

### Parameters

- **columnAddress**
  - Type: `System.Int32`
  - Column number (zero based)

- **rowAddress**
  - Type: `System.Int32`
  - Row number (zero based)

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if one of the passed cell addresses is out of range</td>
</tr>
</tbody>
</table>

### See Also
Reference
Worksheet Class
setCurrentCellAddress Overload
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetSetCurrentColumnNumber Method

Sets the current column number (zero based)

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void SetCurrentColumnNumber(int columnNumber)
```

### Parameters

- `columnNumber`  
  Type: `System.Int32`  
  Column number (zero based)

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RangeException</code></td>
<td>Throws a <code>RangeException</code> if the number is out of the valid range. Range is from 0 to 16383 (16384 columns)</td>
</tr>
</tbody>
</table>

### See Also

Reference  
**Worksheet Class**
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetSetCurrentRowNumber Method

Sets the current row number (zero based)

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public void SetCurrentRowNumber(
    int rowNumber
)
```

### Parameters

- **rowNumber**  
  Type: System.Int32  
  Row number (zero based)

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws an RangeException if the number is out of the valid range. Range is from 0 to 1048575 (1048576 rows)</td>
</tr>
</tbody>
</table>

### See Also

Reference  
Worksheet Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet SetRowHeight Method

Sets the height of the passed row number (zero-based)

**Namespace:** NanoXLXS
**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void SetRowHeight(
    int rowNumber,
    float height
)
```

**Parameters**

*rowNumber*

Type: SystemInt32
Row number (zero-based, 0 to 1048575)

*height*

Type: SystemSingle
Height from 0 to 409.5

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws an RangeException:</td>
</tr>
<tr>
<td></td>
<td>a) If the passed row number is out of range</td>
</tr>
<tr>
<td></td>
<td>b) If the row height is out of range (0 - 409.5)</td>
</tr>
</tbody>
</table>
See Also

Reference
Worksheet Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetSetRowHiddenState Method

Sets the defined row as hidden or visible

**Namespace:** NanoXLX
**Assembly:** NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private void SetRowHiddenState(
    int rowNumber,
    bool state
)
```

**Parameters**

- **rowNumber**
  - Type: `SystemInt32`
  - Row number to make visible again

- **state**
  - Type: `SystemBoolean`
  - If true, the row will be hidden, otherwise visible

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RangeException</code></td>
<td>Throws an <code>RangeException</code> if the passed row number was out of range</td>
</tr>
</tbody>
</table>

### See Also
### WorksheetSetSelectedCells Method

#### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SetSelectedCells(String)</code></td>
<td>Sets the selected cells on this worksheet</td>
</tr>
<tr>
<td><code>SetSelectedCells(CellRange)</code></td>
<td>Sets the selected cells on this worksheet</td>
</tr>
<tr>
<td><code>SetSelectedCells(CellAddress, CellAddress)</code></td>
<td>Sets the selected cells on this worksheet</td>
</tr>
</tbody>
</table>

#### See Also

Reference

- [Worksheet Class](#)
- [NanoXLSX Namespace](#)
WorksheetSetSelectedCells Method (String)

Sets the selected cells on this worksheet

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public void SetSelectedCells(
    string range
)
```

### Parameters

- **range**  
  Type: `System.String`  
  Cell range to select

### See Also

- Reference  
  Worksheet Class  
  SetSelectedCells Overload  
  NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetSetSelectedCells Method (CellRange)

Sets the selected cells on this worksheet

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void SetSelectedCells(
    CellRange range
)
```

**Parameters**

`range`
- Type: NanoXLSXCellRange
- Cell range to select

### See Also

Reference
- Worksheet Class
- SetSelectedCells Overload
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetSetSelectedCells Method (CellAddress, CellAddress)

Sets the selected cells on this worksheet

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

C#  
```
public void SetSelectedCells(
    CellAddress startAddress,
    CellAddress endAddress
)
```

### Parameters

- **startAddress**  
  Type: NanoXLSXCellAddress  
  Start address of the range

- **endAddress**  
  Type: NanoXLSXCellAddress  
  End address of the range

### See Also

Reference  
Worksheet Class  
SetSelectedCells Overload  
NanoXLSX Namespace
WorksheetSetSheetname Method

Validates and sets the worksheet name

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void SetSheetname(
    string name
)
```

### Parameters

- **name**
  
  Type: System.String

  Name to set

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormatException</td>
<td>Throws a FormatException if the sheet name is too long (max. 31) or contains illegal characters [ ] * ? /</td>
</tr>
</tbody>
</table>

### See Also

Reference

Worksheet Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetSetSheetName Method

Sets the name of the sheet

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void SetSheetName(
    string name,
    bool sanitize
)
```

### Parameters

- **name**
  - Type: `System.String`  
  - Name of the sheet

- **sanitize**
  - Type: `System.Boolean`  
  - If true, the filename will be sanitized automatically according to the specifications of Excel

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorksheetException</td>
<td>WorksheetException Thrown if no workbook is referenced. This information is necessary to determine whether the</td>
</tr>
</tbody>
</table>
name already exists

See Also

Reference
Worksheet Class
NanoXSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetSetSheetProtectionPassword Method

Sets or removes the password for worksheet protection. If set, UseSheetProtection will be also set to true

Namespace: NanoXLXS
Assembly: NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```
public void SetSheetProtectionPassword(
    string password
)
```

Parameters

password
Type: System.String
Password (UTF-8) to protect the worksheet. If the password is null or empty, no password will be used

See Also

Reference
Worksheet Class
NanoXLXS Namespace

Copyright Raphael Stoeckli © 2018
Worksheet Exists Method

Checks whether a worksheet with the given name exists

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
private static bool WorksheetExists(
    string name,
    Workbook workbook
)
```

**Parameters**

- **name**
  - Type: `System.String`  
  - Name to check

- **workbook**
  - Type: `NanoXLSXWorkbook`  
  - Workbook reference

**Return Value**

- Type: `Boolean`  
  - True if the name exits, otherwise false

### See Also

Reference  
**Worksheet Class**
# Worksheet Fields

The **Worksheet** type exposes the following members.

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT_COLUMN_WIDTH</td>
<td>Default column width as constant</td>
</tr>
<tr>
<td>DEFAULT_ROW_HEIGHT</td>
<td>Default row height as constant</td>
</tr>
<tr>
<td>MAX_COLUMN_NUMBER</td>
<td>Maximum column number (zero-based) as constant</td>
</tr>
<tr>
<td>MAX_COLUMN_WIDTH</td>
<td>Maximum column width as constant</td>
</tr>
<tr>
<td>MAX_ROW_HEIGHT</td>
<td>Maximum row height as constant</td>
</tr>
<tr>
<td>MAX_ROW_NUMBER</td>
<td>Maximum row number (zero-based) as constant</td>
</tr>
<tr>
<td>MIN_COLUMN_NUMBER</td>
<td>Minimum column number (zero-based) as constant</td>
</tr>
<tr>
<td><strong>MIN_COLUMN_WIDTH</strong></td>
<td>Minimum column width as constant</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td><strong>MIN_ROW_HEIGHT</strong></td>
<td>Minimum row height as constant</td>
</tr>
<tr>
<td><strong>MIN_ROW_NUMBER</strong></td>
<td>Minimum row number (zero-based) as constant</td>
</tr>
</tbody>
</table>

See Also

Reference
Worksheet Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet

**DEFAULT_COLUMN_WIDTH Field**

Default column width as constant

**Namespace:** NanoXLX

**Assembly:** NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public const float DEFAULT_COLUMN_WIDTH = 10f
```

Field Value

Type: `Single`

### See Also

Reference

- Worksheet Class
- NanoXLX Namespace
Default row height as constant

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public const float DEFAULT_ROW_HEIGHT = 15f
```

**Field Value**  
**Type:** Single

### See Also

**Reference**  
  - Worksheet Class  
  - NanoXLSX Namespace
WorksheetMAX_COLUMN_NUMBENumber Field

Maximum column number (zero-based) as constant

**Namespace:**  NanoXLSX  
**Assembly:**  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```
public const int MAX_COLUMN_NUMBER = 16383
```

**Field Value**  
Type:  **Int32**

**See Also**

Reference  
Worksheet Class  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet MAX_COLUMN_WIDTH Field

Maximum column width as constant

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public const float MAX_COLUMN_WIDTH = 255f
```

**Field Value**  
**Type:** Single

### See Also

**Reference**  
Worksheet Class  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet MAX_ROW_HEIGHT Field

Maximum row height as constant

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```
public const float MAX_ROW_HEIGHT = 409.5f
```

Field Value  
Type: Single

## See Also

Reference  
**Worksheet Class**  
**NanoXLSX Namespace**

Copyright Raphael Stoeckli © 2018
WorksheetMAX_ROW_NUMBER Field

Maximum row number (zero-based) as constant

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public const int MAX_ROW_NUMBER = 1048575
```

**Field Value**  
Type: Int32

### See Also

**Reference**  
- Worksheet Class  
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet MIN_COLUMN_NUMBER Field

Minimum column number (zero-based) as constant

**Namespace:** NanoXLX

**Assembly:** NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public const int MIN_COLUMN_NUMBER = 0
```

Field Value
Type: Int32

### See Also

Reference
- Worksheet Class
- NanoXLX Namespace

Copyright Raphael Stoeckli © 2018
Worksheet MIN_COLUMN_WIDTH Field

Minimum column width as constant

Namespace: NanoXLSX  
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public const float MIN_COLUMN_WIDTH = 0f
```

Field Value
Type: Single

See Also

Reference
Worksheet Class  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetMIN_ROW_HEIGHT Field

Minimum row height as constant

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public const float MIN_ROW_HEIGHT = 0f
```

Field Value  
Type: **Single**

### See Also

Reference  
**Worksheet Class**  
**NanoXLSX Namespace**

Copyright Raphael Stoeckli © 2018
Worksheet MIN_ROW_NUMBER Field

Minimum row number (zero-based) as constant

**Namespace:** NanoXLXS
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```c#
public const int MIN_ROW_NUMBER = 0
```

Field Value
Type: Int32

## See Also

Reference
- Worksheet Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetCellDirection Enumeration

Enum to define the direction when using AddNextCell method

**Namespace:** NanoLSX  
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public enum CellDirection
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColumnToColumn</td>
<td>0</td>
<td>The next cell will be on the same row (A1,B1,C1...)</td>
</tr>
<tr>
<td>RowToRow</td>
<td>1</td>
<td>The next cell will be on the same column (A1,A2,A3...)</td>
</tr>
</tbody>
</table>

### See Also

Reference  
NanoLSX Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
WorksheetColumn Class

Class representing a column of a worksheet

Inheritance Hierarchy

- `SystemObject`
- `NanoXLSXWorksheetColumn`

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public class Column
```

The `WorksheetColumn` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Constructor icon] WorksheetColumn</td>
<td>Default constructor</td>
</tr>
<tr>
<td>![Constructor icon] WorksheetColumn(Int32)</td>
<td>Constructor with column number</td>
</tr>
<tr>
<td>![Constructor icon] WorksheetColumn(String)</td>
<td>Constructor with column address</td>
</tr>
</tbody>
</table>

Properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColumnAddress</td>
<td>Column address (A to XFD)</td>
</tr>
<tr>
<td>HasAutoFilter</td>
<td>If true, the column has auto filter applied, otherwise not</td>
</tr>
<tr>
<td>IsHidden</td>
<td>If true, the column is hidden, otherwise visible</td>
</tr>
<tr>
<td>Number</td>
<td>Column number (0 to 16383)</td>
</tr>
<tr>
<td>Width</td>
<td>Width of the column</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------------------------</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference
NanoXLSX Namespace
Column Constructor

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorksheetColumn</td>
<td>Default constructor</td>
</tr>
<tr>
<td>WorksheetColumn(Int32)</td>
<td>Constructor with column number</td>
</tr>
<tr>
<td>WorksheetColumn(String)</td>
<td>Constructor with column address</td>
</tr>
</tbody>
</table>

See Also

Reference
- WorksheetColumn Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetColumn Constructor

Default constructor

**Namespace:** NanoLSX  
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public Column()
```

### See Also

Reference

 WorksheetColumn Class  
 WorksheetColumn Overload  
 NanoLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetColumn Constructor (Int32)

Constructor with column number

Namespace: NanoXL
Assembly: NanoXL (in NanoXL.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
public Column(
    int columnCoordinate
)
```

Parameters

`columnCoordinate`
- Type: SystemInt32
- Column number (zero-based, 0 to 16383)

See Also

Reference
- WorksheetColumn Class
- WorksheetColumn Overload
- NanoXL Namespace

Copyright Raphael Stoeckli © 2018
WorksheetColumn Constructor (String)

Constructor with column address

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Column(
    string columnAddress
)
```

### Parameters

`columnAddress`  
Type: `System.String`  
Column address (A to XFD)

### See Also

Reference
- WorksheetColumn Class
- WorksheetColumn Overload
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
# Column Properties

The `WorksheetColumn` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColumnAddress</td>
<td>Column address (A to XFD)</td>
</tr>
<tr>
<td>HasAutoFilter</td>
<td>If true, the column has auto filter applied, otherwise not</td>
</tr>
<tr>
<td>IsHidden</td>
<td>If true, the column is hidden, otherwise visible</td>
</tr>
<tr>
<td>Number</td>
<td>Column number (0 to 16383)</td>
</tr>
<tr>
<td>Width</td>
<td>Width of the column</td>
</tr>
</tbody>
</table>

## See Also

Reference

- `WorksheetColumn Class`
- `NanoXLSX Namespace`

Copyright Raphael Stoeckli © 2018
WorksheetColumn::ColumnAddress Property

Column address (A to XFD)

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string ColumnAddress { get; set; }
```

Property Value  
Type: **String**

### See Also

Reference  
WorksheetColumn Class  
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
**WorksheetColumnHasAutoFilter Property**

If true, the column has auto filter applied, otherwise not

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public bool HasAutoFilter { get; set; }
```

**Property Value**
Type: Boolean

**See Also**

Reference  
**WorksheetColumn Class**  
**NanoXLSX Namespace**

Copyright Raphael Stoeckli © 2018
WorksheetColumn.IsHidden Property

If true, the column is hidden, otherwise visible

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public bool IsHidden { get; set; }
```

Property Value
Type: **Boolean**

### See Also

Reference
- WorksheetColumn Class
- NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetColumnNumber Property

Column number (0 to 16383)

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public int Number { get; set; }
```

Property Value
Type: Int32

See Also

Reference
WorksheetColumn Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetColumnGetWidth Property

Width of the column

Namespace: NanoXLSX
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

- **Syntax**

```c#
public float Width { get; set; }
```

- Property Value
  - Type: Single

- **See Also**

  Reference
  - WorksheetColumn Class
  - NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
Column Methods

The *WorksheetColumn* type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified <em>Object</em> is equal to the current <em>Object</em>. (Inherited from <em>Object.</em>)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <em>Object.</em>)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from <em>Object.</em>)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <em>Type</em> of the current instance. (Inherited from <em>Object.</em>)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <em>Object</em>. (Inherited from <em>Object.</em>)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from <em>Object.</em>)</td>
</tr>
</tbody>
</table>
See Also

Reference
WorksheetColumn Class
NanoXLSX Namespace

Copyright Raphael Stoeckli © 2018
WorksheetSheetProtectionValue Enumeration

Enum to define the possible protection types when protecting a worksheet

**Namespace:** NanoXLSX  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public enum SheetProtectionValue
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objects</td>
<td>0</td>
<td>If selected, the user can edit objects if the worksheets is protected</td>
</tr>
<tr>
<td>scenarios</td>
<td>1</td>
<td>If selected, the user can edit scenarios if the worksheets is protected</td>
</tr>
<tr>
<td>formatCells</td>
<td>2</td>
<td>If selected, the user can format cells if the</td>
</tr>
<tr>
<td>Feature</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>formatColumns</td>
<td>3</td>
<td>If selected, the user can format columns if the worksheets is protected</td>
</tr>
<tr>
<td>formatRows</td>
<td>4</td>
<td>If selected, the user can format rows if the worksheets is protected</td>
</tr>
<tr>
<td>insertColumns</td>
<td>5</td>
<td>If selected, the user can insert columns if the worksheets is protected</td>
</tr>
<tr>
<td>insertRows</td>
<td>6</td>
<td>If selected, the user can insert rows if the worksheets is protected</td>
</tr>
<tr>
<td>insertHyperlinks</td>
<td>7</td>
<td>If selected, the user can insert hyperlinks if the worksheets is protected</td>
</tr>
<tr>
<td>deleteColumns</td>
<td>8</td>
<td>If selected, the user can delete columns if the worksheets is protected</td>
</tr>
<tr>
<td>Feature</td>
<td>ID</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>----</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>deleteRows</td>
<td>9</td>
<td>If selected, the user can delete rows if the worksheets is protected</td>
</tr>
<tr>
<td>selectLockedCells</td>
<td>10</td>
<td>If selected, the user can select locked cells if the worksheets is protected</td>
</tr>
<tr>
<td>sort</td>
<td>11</td>
<td>If selected, the user can sort cells if the worksheets is protected</td>
</tr>
<tr>
<td>autoFilter</td>
<td>12</td>
<td>If selected, the user can use auto filters if the worksheets is protected</td>
</tr>
<tr>
<td>pivotTables</td>
<td>13</td>
<td>If selected, the user can use pivot tables if the worksheets is protected</td>
</tr>
<tr>
<td>selectUnlockedCells</td>
<td>14</td>
<td>If selected, the user can select unlocked cells if the worksheets is protected</td>
</tr>
</tbody>
</table>
# NanoXLSX.Exception Namespace

[Missing <summary> documentation for "N:NanoXLSX.Exception"]

## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormatException</td>
<td>Class for exceptions regarding format error incidents</td>
</tr>
<tr>
<td>IOException</td>
<td>Class for exceptions regarding stream or save error incidents</td>
</tr>
<tr>
<td>RangeException</td>
<td>Class for exceptions regarding range incidents (e.g. out-of-range)</td>
</tr>
<tr>
<td>StyleException</td>
<td>Class for exceptions regarding Style incidents</td>
</tr>
<tr>
<td>WorksheetException</td>
<td>Class for exceptions regarding worksheet incidents</td>
</tr>
</tbody>
</table>
FormatException Class

Class for exceptions regarding format error incidents

Inheritance Hierarchy

System\Object  System\Exception
NanoXLSX.\Exception\FormatException

Namespace:  NanoXLSX.\Exception
Assembly:  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
[SerializableAttribute]
public class FormatException : Exception
```

The `FormatException` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FormatException</code></td>
<td>Default constructor</td>
</tr>
<tr>
<td><code>FormatException(String)</code></td>
<td>Constructor with passed message</td>
</tr>
<tr>
<td><code>FormatException(String, String)</code></td>
<td>Constructor with passed message</td>
</tr>
<tr>
<td><code>FormatException(String, String, Exception)</code></td>
<td>Constructor with passed message and inner exception</td>
</tr>
</tbody>
</table>
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td>ExceptionTitle</td>
<td>Gets or sets the title of the exception</td>
</tr>
<tr>
<td>HelpLink</td>
<td>Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td>HResult</td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td>InnerException</td>
<td>Gets the Exception instance that caused the current exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td>Message</td>
<td>Gets a message that describes the current exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td>Source</td>
<td>Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)</td>
</tr>
</tbody>
</table>
**StackTrace**
Gets a string representation of the immediate frames on the call stack.
(Inherited from Exception.)

**TargetSite**
Gets the method that throws the current exception.
(Inherited from Exception.)

## Top

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetBaseException</strong></td>
<td>When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
</tbody>
</table>
**GetObjectData**
When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)

**GetTypeInfo**
Gets the runtime type of the current instance. (Inherited from Exception.)

**MemberwiseClone**
Creates a shallow copy of the current Object. (Inherited from Object.)

**ToString**
Creates and returns a string representation of the current exception. (Inherited from Exception.)

---

**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SerializeObjectState</td>
<td>Occurs when an exception is serialized to create an exception state object that contains serialized data about the exception. (Inherited from Exception.)</td>
</tr>
</tbody>
</table>

---

**See Also**
## FormatException Constructor

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FormatException</code></td>
<td>Default constructor</td>
</tr>
<tr>
<td><code>FormatException(String)</code></td>
<td>Constructor with passed message</td>
</tr>
<tr>
<td><code>FormatException(String, String)</code></td>
<td>Constructor with passed message</td>
</tr>
<tr>
<td><code>FormatException(String, String, Exception)</code></td>
<td>Constructor with passed message and inner exception</td>
</tr>
</tbody>
</table>

### See Also

- **Reference**
  - `FormatException Class`
  - `NanoXLSX.Exception Namespace`

Copyright Raphael Stoeckli © 2018
FormatException Constructor

Default constructor

**Namespace:** NanoXLXS.Exception  
**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public FormatException()
```

### See Also

**Reference**
- FormatException Class
- FormatException Overload
- NanoXLXS.Exception Namespace

Copyright Raphael Stoeckli © 2018
FormatException Constructor (String)

Constructor with passed message

**Namespace:** NanoXLXS.Exception  
**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public FormatException(  
    string message  
)
```

**Parameters**

*message*
- Type: `System.String`
- Message of the exception

### See Also

**Reference**
- Exception Class
- FormatException Overload
- NanoXLXS.Exception Namespace

Copyright Raphael Stoeckli © 2018
FormatException Constructor (String, String)

Constructor with passed message

**Namespace:** NanoXLSX.Exception  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
publicFormatException(
    string title,
    string message
)
```

**Parameters**

- **title**  
  Type: System.String  
  Title of the exception

- **message**  
  Type: System.String  
  Message of the exception

### See Also

- Reference  
  *FormatException Class  
  *FormatException Overload  
  *NanoXLSX.Exception Namespace
FormatException Constructor (String, String, Exception)

Constructor with passed message and inner exception

**Namespace:** NanoXLXS.Exception  
**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

▲ Syntax

```csharp
public FormatException(
    string title,
    string message,
    Exception inner
)
```

Parameters

- **title**  
  Type: System.String  
  Title of the exception

- **message**  
  Type: System.String  
  Message of the exception

- **inner**  
  Type: System.Exception  
  Inner exception

▲ See Also

Reference  
FormatException Class
The `FormatException` type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from <code>Exception</code>.)</td>
</tr>
<tr>
<td>ExceptionTitle</td>
<td>Gets or sets the title of the exception</td>
</tr>
<tr>
<td>HelpLink</td>
<td>Gets or sets a link to the help file associated with this exception.        (Inherited from <code>Exception</code>.)</td>
</tr>
<tr>
<td>HResult</td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from <code>Exception</code>.)</td>
</tr>
<tr>
<td>InnerException</td>
<td>Gets the <code>Exception</code> instance that caused the current exception.             (Inherited from <code>Exception</code>.)</td>
</tr>
<tr>
<td>Message</td>
<td>Gets a message that describes the current exception.                         (Inherited from <code>Exception</code>.)</td>
</tr>
<tr>
<td>Source</td>
<td>Gets or sets the name of the</td>
</tr>
</tbody>
</table>
application or the object that causes the error.  
(Inherited from Exception.)

| StackTrace | Gets a string representation of the immediate frames on the call stack.  
(Inherited from Exception.) |
|------------|--------------------------------------------------------------------------|
| TargetSite | Gets the method that throws the current exception.  
(Inherited from Exception.) |

See Also

Reference

FormatException Class
NanoLSX.Exception Namespace

Copyright Raphael Stoeckli © 2018
FormatExceptionExceptionTitle
Property

Gets or sets the title of the exception

Namespace: NanoXLSX.Exception
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public string ExceptionTitle { get; set; }
```

Property Value
Type: String

See Also

Reference
FormatException Class
NanoXLSX.Exception Namespace

Copyright Raphael Stoeckli © 2018
FormatException Methods

The `FormatException` type exposes the following members.

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Equals**         | Determines whether the specified `Object` is equal to the current `Object`.  
                        (Inherited from `Object`.)                                                                                                           |
| **Finalize**       | Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.  
                        (Inherited from `Object`.)                                                                                                           |
| **GetBaseException** | When overridden in a derived class, returns the `Exception` that is the root cause of one or more subsequent exceptions.  
                        (Inherited from `Exception`.)                                                                                                          |
| **GetHashCode**   | Serves as a hash function for a particular type.  
                        (Inherited from `Object`.)                                                                                                           |
| **GetObjectData** | When overridden in a derived class, sets the `SerializationInfo` with information about the                                             |
### See Also

#### Reference
- [FormatException Class](#)
- [NanoXLXS.Exception Namespace](#)

**Copyright Raphael Stoeckli © 2018**
FormatException Events

The `FormatException` type exposes the following members.

**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SerializeObjectState</td>
<td>Occurs when an exception is serialized to create an exception state object that contains serialized data about the exception. (Inherited from <code>Exception</code>.)</td>
</tr>
</tbody>
</table>

See Also

Reference

- `FormatException Class`
- `NanoXLX.Exception Namespace`

Copyright Raphael Stoeckli © 2018
IOException Class

Class for exceptions regarding stream or save error incidents

Inheritance Hierarchy

System
  SystemObject
    SystemException
      NanoXL SX.Exception
        IOException

Namespace: NanoXL SX.Exception
Assembly: NanoXL SX (in NanoXL SX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

[C#]

```csharp
[SerializableAttribute]
public class IOException : Exception
```

The IOException type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>Default constructor</td>
</tr>
<tr>
<td>IOException(String, String)</td>
<td>Constructor with passed message</td>
</tr>
<tr>
<td>IOException(String, String, Exception)</td>
<td>Constructor with passed message and inner exception</td>
</tr>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td>ExceptionTitle</td>
<td>Gets or sets the title of the exception</td>
</tr>
<tr>
<td>HelpLink</td>
<td>Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td>HResult</td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td>InnerException</td>
<td>Gets the Exception instance that caused the current exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td>Message</td>
<td>Gets a message that describes the current exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td>Source</td>
<td>Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)</td>
</tr>
<tr>
<td>StackTrace</td>
<td>Gets a string representation of the immediate frames on the call</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>TargetSite</td>
<td>Gets the method that throws the current exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>Methods</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetObjectData</td>
<td>When overridden in a derived class, sets the</td>
</tr>
</tbody>
</table>
SerializationInfo with information about the exception.  
(Inherited from Exception.)

**GetType**  
Gets the runtime type of the current instance.  
(Inherited from Exception.)

**MemberwiseClone**  
Creates a shallow copy of the current Object.  
(Inherited from Object.)

**ToString**  
Creates and returns a string representation of the current exception.  
(Inherited from Exception.)

---

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| SerializeObjectState  | Occurs when an exception is serialized to create an exception state object that contains serialized data about the exception. 
                       | (Inherited from Exception.)                                                                                                                  |

---

### See Also

Reference  
NanoXLSX.Exception Namespace
## IOException Constructor

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>Default constructor</td>
</tr>
<tr>
<td>IOException(String, String)</td>
<td>Constructor with passed message</td>
</tr>
<tr>
<td>IOException(String, String, Exception)</td>
<td>Constructor with passed message and inner exception</td>
</tr>
</tbody>
</table>

### See Also

**Reference**
- IOException Class
- NanoXLSX.Exception Namespace

(80x685) IOException	Constructor
(94x618) Overload	List
(184x585) Name
(321x585) Description
(184x556) IOException
(321x556) Default	constructor
(184x523) IOException(String, String)
(321x523) Constructor	with	passed
message
(184x474) IOException(String, String, Exception)
(321x474) Constructor	with	passed
message	and	inner exception
(94x366) Copyright	Raphael	Stoeckli	©	2018
IOException Constructor

Default constructor

**Namespace:** NanoXLSX.Exception  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public IOException()
```

### See Also

Reference
- IOException Class
- IOException Overload
- NanoXLSX.Exception Namespace

Copyright Raphael Stoeckli © 2018
IOException Constructor (String, String)

Constructor with passed message

**Namespace:** NanoXLSX.Exception  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public IOException(
    string title,
    string message
)
```

### Parameters

- **title**
  - Type: `System.String`
  - Title of the exception
- **message**
  - Type: `System.String`
  - Message of the exception

### See Also

- Reference
  - IOException Class
  - IOException Overload
  - NanoXLSX.Exception Namespace

Copyright Raphael Stoeckli © 2018
IOException Constructor (String, String, Exception)

Constructor with passed message and inner exception

**Namespace:** NanoLSX.Exception  
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public IOException(
    string title,
    string message,
    Exception inner
)
```

### Parameters

- **title**
  - Type: SystemString  
  - Title of the exception

- **message**
  - Type: SystemString  
  - Message of the exception

- **inner**
  - Type: SystemException  
  - Inner exception

### See Also

- Reference
- IOException Class
# IOException Properties

The `IOException` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from <code>Exception</code>.)</td>
</tr>
<tr>
<td>ExceptionTitle</td>
<td>Gets or sets the title of the exception.</td>
</tr>
<tr>
<td>HelpLink</td>
<td>Gets or sets a link to the help file associated with this exception. (Inherited from <code>Exception</code>.)</td>
</tr>
<tr>
<td>HResult</td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from <code>Exception</code>.)</td>
</tr>
<tr>
<td>InnerException</td>
<td>Gets the <code>Exception</code> instance that caused the current exception. (Inherited from <code>Exception</code>.)</td>
</tr>
<tr>
<td>Message</td>
<td>Gets a message that describes the current exception. (Inherited from <code>Exception</code>.)</td>
</tr>
<tr>
<td>Source</td>
<td>Gets or sets the name of the</td>
</tr>
</tbody>
</table>
application or the object that causes the error. (Inherited from `Exception`.)

<table>
<thead>
<tr>
<th>StackTrace</th>
<th>Gets a string representation of the immediate frames on the call stack. (Inherited from <code>Exception</code>.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetSite</td>
<td>Gets the method that throws the current exception. (Inherited from <code>Exception</code>.)</td>
</tr>
</tbody>
</table>

**See Also**

**Reference**
- `IOException` Class
- `NanoXLSX.Exception` Namespace

Copyright Raphael Stoekli © 2018
IOException

Property

Gets or sets the title of the exception

Namespace: NanoXLSX.Exception
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
public string ExceptionTitle { get; set; }
```

Property Value

Type: String

See Also

Reference
 IOException Class
 NanoXLSX.Exception Namespace

Copyright Raphael Stoeckli © 2018
## IOException Methods

The `IOException` type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="false" alt="" /> Equals</td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><img src="false" alt="" /> Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><img src="false" alt="" /> GetBaseException</td>
<td>When overridden in a derived class, returns the <code>Exception</code> that is the root cause of one or more subsequent exceptions. (Inherited from <code>Exception</code>.)</td>
</tr>
<tr>
<td><img src="false" alt="" /> GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><img src="false" alt="" /> GetObjectData</td>
<td>When overridden in a derived class, sets the <code>SerializationInfo</code> with information about the</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Get Type</strong></td>
<td>Gets the runtime type of the current instance.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td><strong>To-String</strong></td>
<td>Creates and returns a string representation of the current exception.</td>
</tr>
</tbody>
</table>

### See Also

- Reference
- `IOException Class`
- `NanoXLSX.Exception Namespace`

Copyright Raphael Stoeckli © 2018
IOException Events

The IOException type exposes the following members.

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SerializeObjectState</td>
<td>Occurs when an exception is serialized to create an exception state object that contains serialized data about the exception. (Inherited from Exception.)</td>
</tr>
</tbody>
</table>

See Also

Reference
IOException Class
NanoXLSX.Exception Namespace
RangeException Class

Class for exceptions regarding range incidents (e.g. out-of-range)

Inheritance Hierarchy

- System
  - Object
  - System
    - Exception
      - NanoXL Exception
        - RangeException

Namespace: NanoXL.Exception
Assembly: NanoXL (in NanoXL.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
[SerializableAttribute]
public class RangeException : Exception
```

The `RangeException` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RangeException</code></td>
<td>Default constructor</td>
</tr>
<tr>
<td><code>RangeException(String, String)</code></td>
<td>Constructor with passed message</td>
</tr>
</tbody>
</table>

Properties
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data</strong></td>
<td>Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>ExceptionTitle</strong></td>
<td>Gets or sets the title of the exception</td>
</tr>
<tr>
<td><strong>HelpLink</strong></td>
<td>Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>HResult</strong></td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>InnerException</strong></td>
<td>Gets the Exception instance that caused the current exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>Message</strong></td>
<td>Gets a message that describes the current exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>StackTrace</strong></td>
<td>Gets a string representation of the immediate frames on the call stack. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>TargetSite</strong></td>
<td>Gets the method that throws the current exception.</td>
</tr>
</tbody>
</table>
(Inherited from Exception.)

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ ] Equals</td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>![ ] Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>![ ] GetBaseException</td>
<td>When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)</td>
</tr>
<tr>
<td>![ ] GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td>![ ] GetObjectData</td>
<td>When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)</td>
</tr>
</tbody>
</table>
### GetType
Gets the runtime type of the current instance.
(Inherited from Exception.)

### MemberwiseClone
Creates a shallow copy of the current Object.
(Inherited from Object.)

### ToString
Creates and returns a string representation of the current exception.
(Inherited from Exception.)

---

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SerializeObjectState</td>
<td>Occurs when an exception is serialized to create an exception state object that contains serialized data about the exception. (Inherited from Exception.)</td>
</tr>
</tbody>
</table>

---

## See Also

Reference
NanoXLSX.Exception Namespace

---

Copyright Raphael Stoeckli © 2018
RangeException Constructor

- **Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Default constructor</td>
</tr>
<tr>
<td>RangeException(String, String)</td>
<td>Constructor with passed message</td>
</tr>
</tbody>
</table>

**See Also**

Reference
- RangeException Class
- NanoXLSX.Exception Namespace

Copyright Raphael Stoeckli © 2018
RangeException Constructor

Default constructor

**Namespace:** NanoXLSX.Exception  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public RangeException()
```

### See Also

**Reference**
- RangeException Class  
- RangeException Overload  
- NanoXLSX.Exception Namespace
RangeException Constructor (String, String)

Constructor with passed message

**Namespace:** NanoXLSX.Exception  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

⚠️ Syntax

```csharp
public RangeException(
    string title,
    string message
)
```

Parameters

*title*  
Type: System.String  
Title of the exception

*message*  
Type: System.String  
Message of the exception

⚠️ See Also

Reference  
RangeException Class  
RangeException Overload  
NanoXLSX.Exception Namespace

Copyright Raphael Stoeckli © 2018
## RangeException Properties

The **RangeException** type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Gets a collection of key/value pairs that provide additional user-defined information about the exception.</td>
</tr>
<tr>
<td></td>
<td><em>(Inherited from Exception.)</em></td>
</tr>
<tr>
<td>ExceptionTitle</td>
<td>Gets or sets the title of the exception</td>
</tr>
<tr>
<td>HelpLink</td>
<td>Gets or sets a link to the help file associated with this exception. <em>(Inherited from Exception.)</em></td>
</tr>
<tr>
<td>HResult</td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. <em>(Inherited from Exception.)</em></td>
</tr>
<tr>
<td>InnerException</td>
<td>Gets the Exception instance that caused the current exception. <em>(Inherited from Exception.)</em></td>
</tr>
<tr>
<td>Message</td>
<td>Gets a message that describes the current exception. <em>(Inherited from Exception.)</em></td>
</tr>
<tr>
<td>Source</td>
<td>Gets or sets the name of the</td>
</tr>
</tbody>
</table>
application or the object that causes the error.  
(Inherited from Exception.)

| StackTrace | Gets a string representation of the immediate frames on the call stack.  
(Inherited from Exception.) |
|-----------------|----------------------------------------------------------------------------------|
| TargetSite | Gets the method that throws the current exception.  
(Inherited from Exception.) |

**See Also**

- **Reference**
  - `RangeException Class`
  - `NanoXLSX.Exception Namespace`

Copyright Raphael Stoeckli © 2018
RangeException

Property

Gets or sets the title of the exception

**Namespace:** NanoLSX.Exception

**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

⚠️ Syntax

```c#
public string ExceptionTitle { get; set; }
```

Property Value

Type: String

⚠️ See Also

Reference

RangeException Class

NanoLSX.Exception Namespace

Copyright Raphael Stoeckli © 2018
## RangeException Methods

The `RangeException` type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>When overridden in a derived class, returns the <code>Exception</code> that is the root cause of one or more subsequent exceptions. (Inherited from <code>Exception</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetObjectData</td>
<td>When overridden in a derived class, sets the <code>SerializationInfo</code> with information about the</td>
</tr>
</tbody>
</table>
exception.  
(Inherited from Exception.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetType</td>
<td>Gets the runtime type of the current instance. (Inherited from Exception.)</td>
</tr>
</tbody>
</table>
| MemberwiseClone | Creates a shallow copy of the current Object.  
(Inherited from Object.) |
| ToString        | Creates and returns a string representation of the current exception.  
(Inherited from Exception.) |

See Also

Reference

RangeException Class
NanoXLSX.Exception Namespace
RangeException Events

The `RangeException` type exposes the following members.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SerializeObjectState</td>
<td>Occurs when an exception is serialized to create an exception state object that contains serialized data about the exception. (Inherited from <code>Exception</code>.)</td>
</tr>
</tbody>
</table>

See Also

Reference

- `RangeException Class`
- `NanoXLSX.Exception Namespace`
StyleException Class

Class for exceptions regarding Style incidents

Inheritance Hierarchy

- SystemObject
- SystemException
- NanoXLSX.Exception
  - StyleException

Namespace: NanoXLSX.Exception
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
[SerializableAttribute]
public class StyleException : Exception
```

The `StyleException` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![StyleException]</td>
<td>Default constructor</td>
</tr>
<tr>
<td>![StyleException(String, String)]</td>
<td>Constructor with passed message</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top</td>
<td></td>
</tr>
<tr>
<td><strong>Data</strong></td>
<td>Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td>----------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ExceptionTitle</strong></td>
<td>Gets or sets the title of the exception</td>
</tr>
<tr>
<td><strong>HelpLink</strong></td>
<td>Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>HResult</strong></td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>InnerException</strong></td>
<td>Gets the Exception instance that caused the current exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>Message</strong></td>
<td>Gets a message that describes the current exception. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>StackTrace</strong></td>
<td>Gets a string representation of the immediate frames on the call stack. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>TargetSite</strong></td>
<td>Gets the method that throws the current exception.</td>
</tr>
</tbody>
</table>
(Inherited from Exception.)

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetBaseException</strong></td>
<td>When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetObjectData</strong></td>
<td>When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)</td>
</tr>
</tbody>
</table>
GetType

Gets the runtime type of the current instance.
(Inherited from Exception.)

MemberwiseClone

Creates a shallow copy of the current Object.
(Inherited from Object.)

ToString

Creates and returns a string representation of the current exception.
(Inherited from Exception.)

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SerializeObjectState</td>
<td>Occurs when an exception is serialized to create an exception state object that contains serialized data about the exception.</td>
</tr>
</tbody>
</table>
# StyleException Constructor

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Default constructor</td>
</tr>
<tr>
<td>StyleException(String, String)</td>
<td>Constructor with passed message</td>
</tr>
</tbody>
</table>

## See Also

Reference
- StyleException Class
- NanoXLSX.Exception Namespace

Copyright Raphael Stoeckli © 2018
StyleException Constructor

Default constructor

**Namespace:** NanoXLX.Exception  
**Assembly:** NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleException()
```

### See Also

Reference  
- StyleException Class  
- StyleException Overload  
- NanoXLX.Exception Namespace

Copyright Raphael Stoeckli © 2018
StyleException Constructor (String, String)

Constructor with passed message

**Namespace:** NanoXLXS.Exception  
**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleException(
    string title,
    string message
)
```

### Parameters

- **title**
  - Type: `System.String`  
  - Title of the exception

- **message**
  - Type: `System.String`  
  - Message of the exception

### See Also

- Reference
  - `StyleException` Class
  - `StyleException` Overload
  - `NanoXLXS.Exception` Namespace

Copyright Raphael Stoeckli © 2018
StyleException Properties

The **StyleException** type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from <strong>Exception</strong>.)</td>
</tr>
<tr>
<td>ExceptionTitle</td>
<td>Gets or sets the title of the exception</td>
</tr>
<tr>
<td>HelpLink</td>
<td>Gets or sets a link to the help file associated with this exception. (Inherited from <strong>Exception</strong>.)</td>
</tr>
<tr>
<td>HResult</td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from <strong>Exception</strong>.)</td>
</tr>
<tr>
<td>InnerException</td>
<td>Gets the <strong>Exception</strong> instance that caused the current exception. (Inherited from <strong>Exception</strong>.)</td>
</tr>
<tr>
<td>Message</td>
<td>Gets a message that describes the current exception. (Inherited from <strong>Exception</strong>.)</td>
</tr>
<tr>
<td>Source</td>
<td>Gets or sets the name of the</td>
</tr>
</tbody>
</table>
application or the object that causes the error.
(Inherited from Exception.)

<table>
<thead>
<tr>
<th>StackTrace</th>
<th>Gets a string representation of the immediate frames on the call stack. (Inherited from Exception.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetSite</td>
<td>Gets the method that throws the current exception. (Inherited from Exception.)</td>
</tr>
</tbody>
</table>

See Also

Reference
StyleException Class
NanoXLSX.Exception Namespace
**StyleException**

Property

Gets or sets the title of the exception

**Namespace:** NanoXSX.Exception  
**Assembly:** NanoXSX (in NanoXSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string ExceptionTitle { get; set; }
```

Property Value

Type: String

### See Also

Reference

- StyleException Class
- NanoXSX.Exception Namespace

Copyright Raphael Stoeckli © 2018
## StyleException Methods

The **StyleException** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetBaseException</strong></td>
<td>When overridden in a derived class, returns the <strong>Exception</strong> that is the root cause of one or more subsequent exceptions. (Inherited from <strong>Exception</strong>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetObjectData</strong></td>
<td>When overridden in a derived class, sets the <strong>SerializationInfo</strong> with information about the</td>
</tr>
</tbody>
</table>
### See Also

**Reference**

- **StyleException Class**
- **NanoXLXS.Exception Namespace**

Copyright Raphael Stoeckli © 2018
StyleException Events

The StyleException type exposes the following members.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Image" alt="" /> SerializeObjectState</td>
<td>Occurs when an exception is serialized to create an exception state object that contains serialized data about the exception. (Inherited from Exception.)</td>
</tr>
</tbody>
</table>

Top

### See Also

Reference
- StyleException Class
- NanoLSX.Exception Namespace

Copyright Raphael Stoeckli © 2018
WorksheetException Class

Class for exceptions regarding worksheet incidents

Inheritance Hierarchy

SystemObject  SystemException
    NanoXLSX.ExceptionWorksheetException

Namespace: NanoXLSX.Exception
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
[SerializableAttribute]
public class WorksheetException : Exception
```

The WorksheetException type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorksheetException</td>
<td>Default constructor</td>
</tr>
<tr>
<td>WorksheetException(String, String)</td>
<td>Constructor with passed message</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Data</strong></td>
<td>Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from <strong>Exception</strong>.)</td>
</tr>
<tr>
<td><strong>ExceptionTitle</strong></td>
<td>Gets or sets the title of the exception</td>
</tr>
<tr>
<td><strong>HelpLink</strong></td>
<td>Gets or sets a link to the help file associated with this exception. (Inherited from <strong>Exception</strong>.)</td>
</tr>
<tr>
<td><strong>HRESULT</strong></td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from <strong>Exception</strong>.)</td>
</tr>
<tr>
<td><strong>InnerException</strong></td>
<td>Gets the <strong>Exception</strong> instance that caused the current exception. (Inherited from <strong>Exception</strong>.)</td>
</tr>
<tr>
<td><strong>Message</strong></td>
<td>Gets a message that describes the current exception. (Inherited from <strong>Exception</strong>.)</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>Gets or sets the name of the application or the object that causes the error. (Inherited from <strong>Exception</strong>.)</td>
</tr>
<tr>
<td><strong>StackTrace</strong></td>
<td>Gets a string representation of the immediate frames on the call stack. (Inherited from <strong>Exception</strong>.)</td>
</tr>
<tr>
<td><strong>TargetSite</strong></td>
<td>Gets the method that throws the current exception.</td>
</tr>
</tbody>
</table>
### Top Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetBaseException</strong></td>
<td>When overridden in a derived class, returns the <code>Exception</code> that is the root cause of one or more subsequent exceptions. (Inherited from <code>Exception</code>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetObjectData</strong></td>
<td>When overridden in a derived class, sets the <code>SerializationInfo</code> with information about the exception. (Inherited from <code>Exception</code>.)</td>
</tr>
</tbody>
</table>
### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SerializeObjectState</td>
<td>Occurs when an exception is serialized to create an exception state object that contains serialized data about the exception. (Inherited from Exception.)</td>
</tr>
</tbody>
</table>

### See Also

- Reference
  - NanoXLSX.Exception Namespace

Copyright Raphael Stoeckli © 2018
WorksheetException Constructor

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorksheetException</td>
<td>Default constructor</td>
</tr>
<tr>
<td>WorksheetException(String, String)</td>
<td>Constructor with passed message</td>
</tr>
</tbody>
</table>

### See Also

Reference

WorksheetException Class
NanoXLSX.Exception Namespace

Copyright Raphael Stoeckli © 2018
WorksheetException Constructor

Default constructor

**Namespace:** NanoXSX.Exception  
**Assembly:** NanoXSX (in NanoXSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public WorksheetException()
```

### See Also

Reference
- WorksheetException Class
- WorksheetException Overload
- NanoXSX.Exception Namespace

Copyright Raphael Stoeckli © 2018
WorksheetException Constructor (String, String)

Constructor with passed message

**Namespace:** NanoLSX.Exception  
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

⚠️ Syntax

```csharp
public WorksheetException(  
    string title,  
    string message
)
```

Parameters

- **title**
  - Type: SystemString
  - Title of the exception

- **message**
  - Type: SystemString
  - Message of the exception

⚠️ See Also

- Reference
  - WorksheetException Class
  - WorksheetException Overload
  - NanoLSX.Exception Namespace

Copyright Raphael Stoeckli © 2018
## WorksheetException Properties

The **WorksheetException** type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data</strong></td>
<td>Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from <strong>Exception.</strong>)</td>
</tr>
<tr>
<td><strong>ExceptionTitle</strong></td>
<td>Gets or sets the title of the exception</td>
</tr>
<tr>
<td><strong>HelpLink</strong></td>
<td>Gets or sets a link to the help file associated with this exception. (Inherited from <strong>Exception.</strong>)</td>
</tr>
<tr>
<td><strong>HResult</strong></td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from <strong>Exception.</strong>)</td>
</tr>
<tr>
<td><strong>InnerException</strong></td>
<td>Gets the <strong>Exception</strong> instance that caused the current exception. (Inherited from <strong>Exception.</strong>)</td>
</tr>
<tr>
<td><strong>Message</strong></td>
<td>Gets a message that describes the current exception. (Inherited from <strong>Exception.</strong>)</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>Gets or sets the name of the</td>
</tr>
</tbody>
</table>
application or the object that causes the error.
(Inherited from Exception.)

| StackTrace | Gets a string representation of the immediate frames on the call stack.
|-------------|------------------------------------------------------------------|
| TargetSite  | Gets the method that throws the current exception.
|             | (Inherited from Exception.)                                      |

See Also

Reference

WorksheetException Class
NanoXLSX.Exception Namespace

Copyright Raphael Stoeckli © 2018
WorksheetExceptionExceptionTitle Property

Gets or sets the title of the exception

**Namespace:** NanoXLSX.Exception  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string ExceptionTitle { get; set; }
```

Property Value  
Type: **String**

### See Also

- Reference  
  - WorksheetException Class  
  - NanoXLSX.Exception Namespace

Copyright Raphael Stoeckli © 2018
## WorksheetException Methods

The `WorksheetException` type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetBaseException</strong></td>
<td>When overridden in a derived class, returns the <code>Exception</code> that is the root cause of one or more subsequent exceptions. (Inherited from <code>Exception</code>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetObjectData</strong></td>
<td>When overridden in a derived class, sets the <code>SerializationInfo</code> with information about the</td>
</tr>
</tbody>
</table>
exception.  
(Inherited from Exception.)

**GetType**  
Gets the runtime type of the current instance.  
(Inherited from Exception.)

**MemberwiseClone**  
Creates a shallow copy of the current Object.  
(Inherited from Object.)

**ToString**  
Creates and returns a string representation of the current exception.  
(Inherited from Exception.)

---

**See Also**

**Reference**
WorksheetException Class  
NanoXLSX.Exception Namespace

---

Copyright Raphael Stoeckli © 2018
# WorksheetException Events

The `WorksheetException` type exposes the following members.

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SerializeObjectState</td>
<td>Occurs when an exception is serialized to create an exception state object that contains serialized data about the exception. (Inherited from <code>Exception</code>.)</td>
</tr>
</tbody>
</table>

## See Also

**Reference**
- `WorksheetException Class`
- `NanoXLSX.Exception Namespace`

Copyright Raphael Stoeckli © 2018
## NanoXLSX.LowLevel Namespace

[Missing <summary> documentation for "N:NanoXLSX.LowLevel"]

### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Image] LowLevel</td>
<td>Class for low level handling (XML, formatting, packing)</td>
</tr>
<tr>
<td>![Image] LowLevelDocumentPath</td>
<td>Class to manage XML document paths</td>
</tr>
<tr>
<td>![Image] LowLevelSortedMap</td>
<td>Class to manage key value pairs (string / string). The entries are in the order how they were added</td>
</tr>
<tr>
<td>![Image] Reader</td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>SharedStringsReader</strong></td>
<td>Class representing a reader for the shared strings table of XLSX files</td>
</tr>
<tr>
<td><strong>WorkbookReader</strong></td>
<td>Class representing a reader to decompile a workbook in an XLSX files</td>
</tr>
<tr>
<td><strong>WorksheetReader</strong></td>
<td>Class representing a reader for worksheets of XLSX files</td>
</tr>
<tr>
<td><strong>WorksheetReaderCellResolverTuple</strong></td>
<td></td>
</tr>
</tbody>
</table>
LowLevel Class

Class for low level handling (XML, formatting, packing)

Inheritance Hierarchy

- SystemObject
  - NanoXLSX.LowLevel

Namespace:  NanoXLSX.LowLevel
Assembly:  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
internal class LowLevel
```

The `LowLevel` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="low-level.png" alt=" LowLevel" /></td>
<td>LowLevel</td>
</tr>
<tr>
<td><img src="low-level.png" alt=" LowLevel" /></td>
<td>LowLevel Constructor with defined workbook object</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="intercept-documents.png" alt=" InterceptDocuments" /></td>
<td>Gets or set whether XML</td>
</tr>
</tbody>
</table>
documents are intercepted during creation

**InterceptedDocuments**  Gets the intercepted documents if interceptDocuments is set to true

---

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppendXmlTag</td>
<td>Method to append a simple XML tag with an enclosed value to the passed StringBuilder</td>
</tr>
<tr>
<td>AppendXmlToPackagePart</td>
<td>Writes raw XML strings into the passed Package Part</td>
</tr>
<tr>
<td>CreateAppPropertiesDocument</td>
<td>Method to create the app-properties (part of metadata) as a raw XML string</td>
</tr>
<tr>
<td>CreateAppString</td>
<td>Method to create the XML string for the app-properties document</td>
</tr>
<tr>
<td>CreateColsString</td>
<td>Method to create the columns XML string. This is used to define the width of columns</td>
</tr>
<tr>
<td>CreateCorePropertiesDocument</td>
<td>Method to create the core-properties (part of metadata) as a raw XML string</td>
</tr>
<tr>
<td>CreateCorePropertiesString</td>
<td>Method to create the XML string for the core-properties document</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CreateMergedCellsString</td>
<td>Method to create the merged cells string of the passed worksheet</td>
</tr>
<tr>
<td>CreateMruColorsString</td>
<td>Method to create the XML string for the color-MRU part of the style sheet document (recent colors)</td>
</tr>
<tr>
<td>CreateRowString</td>
<td>Method to create a row string</td>
</tr>
<tr>
<td>CreateSharedStringsDocument</td>
<td>Method to create shared strings as raw XML string</td>
</tr>
<tr>
<td>CreateSheetProtectionString</td>
<td>Method to create the protection string of the passed worksheet</td>
</tr>
<tr>
<td>CreateStyleBorderString</td>
<td>Method to create the XML string for the border part of the style sheet document</td>
</tr>
<tr>
<td>CreateStyleFillString</td>
<td>Method to create the XML string for the fill part of the style sheet document</td>
</tr>
<tr>
<td>CreateStyleFontString</td>
<td>Method to create the XML string for the font part of the style sheet document</td>
</tr>
<tr>
<td>CreateStyleNumberFormatString</td>
<td>Method to create the XML string for the number format part of the style sheet document</td>
</tr>
<tr>
<td>CreateStyleSheetDocument</td>
<td>Method to create a style sheet as raw XML string</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CreateStyleXfsString</td>
<td>Method to create the XML for the Xf part of the stylesheet document</td>
</tr>
<tr>
<td>CreateWorkbookDocument</td>
<td>Method to create a workbook as a raw XML string</td>
</tr>
<tr>
<td>CreateWorksheetPart</td>
<td>Method to create a worksheet part as a raw XML string</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>EscapeXmlAttributeChars</td>
<td>Method to escape XML characters in an XML attribute</td>
</tr>
<tr>
<td>EscapeXmlChars</td>
<td>Method to escape XML characters between two XML tags</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform cleanup operations</td>
</tr>
<tr>
<td></td>
<td>before being reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GeneratePasswordHash</td>
<td>Method to generate an Excel internal password hash to protect workbooks or</td>
</tr>
<tr>
<td></td>
<td>worksheets. This method is derived from the c++ implementation by Kohei</td>
</tr>
<tr>
<td></td>
<td>Yoshida (<a href="http://kohei.us/2008/01/excel-sheet-protection-password-hash/">http://kohei.us/2008/01/excel-sheet-protection-password-hash/</a>)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetOADateTimeString</td>
<td>Method to convert a date and time into the internal Excel time format (OAdate)</td>
</tr>
<tr>
<td>GetSortedSheetData</td>
<td>Method to sort the cells of a worksheet as preparation for an XML document</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Save</td>
<td>Method to save the workbook</td>
</tr>
<tr>
<td>SaveAsStream</td>
<td>Method to save the workbook as a stream</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

**Remarks**

This class is only for internal use. Use the high level API (e.g. class Workbook) to manipulate data and create Excel files.

**See Also**
LowLevel Constructor

Constructor with defined workbook object

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public LowLevel(
    Workbook workbook
)
```

### Parameters

`workbook`  
Type: NanoXLXWorkbook  
Workbook to process

### See Also

Reference  
LowLevel Class  
NanoXLSX.LowLevel Namespace
LowLevel Constructor

Initializes the static fields of the LowLevel class

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

C# static LowLevel()

### See Also

Reference
LowLevel Class  
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevel Properties

The **LowLevel** type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InterceptDocuments</td>
<td>Gets or set whether XML documents are intercepted during creation</td>
</tr>
<tr>
<td>InterceptedDocuments</td>
<td>Gets the intercepted documents if interceptDocuments is set to true</td>
</tr>
</tbody>
</table>

See Also

Reference

LowLevel Class
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelInterceptDocuments Property

Gets or set whether XML documents are intercepted during creation

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public bool InterceptDocuments { get; set; }
```

Property Value  
Type: Boolean

### See Also

Reference  
LowLevel Class  
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelInterceptedDocuments Property

Gets the intercepted documents if interceptDocuments is set to true

**Namespace:** NanoXL SX.LowLevel  
**Assembly:** NanoXL SX (in NanoXL SX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Dictionary<string, XmlDocument> InterceptedDocuments
```

**Property Value**  
Type: `Dictionary<String, XmlDocument>`

### See Also

**Reference**  
- LowLevel Class  
- NanoXL SX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
## LowLevel Methods

The **LowLevel** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppendXmlTag</td>
<td>Method to append a simple tag with an enclosed value to the passed StringBuilder</td>
</tr>
<tr>
<td>AppendXmlToPackagePart</td>
<td>Writes raw XML strings into the passed Package Part</td>
</tr>
<tr>
<td>CreateAppPropertiesDocument</td>
<td>Method to create the app-properties (part of metadata) as raw XML string</td>
</tr>
<tr>
<td>CreateAppString</td>
<td>Method to create the XML string for the app-properties document</td>
</tr>
<tr>
<td>CreateColsString</td>
<td>Method to create the columns as XML string. This is used to define the width of columns</td>
</tr>
<tr>
<td>CreateCorePropertiesDocument</td>
<td>Method to create the core-properties (part of metadata) as raw XML string</td>
</tr>
<tr>
<td>CreateCorePropertiesString</td>
<td>Method to create the XML string for the core-properties document</td>
</tr>
<tr>
<td>CreateMergedCellsString</td>
<td>Method to create the merged cells string of the passed</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CreateMruColorsString</td>
<td>Method to create the XML for the color-MRU part of the style sheet document (recent colors)</td>
</tr>
<tr>
<td>CreateRowString</td>
<td>Method to create a row string</td>
</tr>
<tr>
<td>CreateSharedStringsDocument</td>
<td>Method to create shared strings as raw XML string</td>
</tr>
<tr>
<td>CreateSheetProtectionString</td>
<td>Method to create the protection string of the passed worksheet</td>
</tr>
<tr>
<td>CreateStyleBorderString</td>
<td>Method to create the XML for the border part of the style sheet document</td>
</tr>
<tr>
<td>CreateStyleFillString</td>
<td>Method to create the XML for the fill part of the style sheet document</td>
</tr>
<tr>
<td>CreateStyleFontString</td>
<td>Method to create the XML for the font part of the style sheet document</td>
</tr>
<tr>
<td>CreateStyleNumberFormatString</td>
<td>Method to create the XML for the number format part of the style sheet document</td>
</tr>
<tr>
<td>CreateStyleSheetDocument</td>
<td>Method to create a style sheet as raw XML string</td>
</tr>
<tr>
<td>CreateStyleXfsString</td>
<td>Method to create the XML for the Xf part of the style sheet document</td>
</tr>
<tr>
<td>Method/Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CreateWorkbookDocument</td>
<td>Method to create a workbook as a raw XML string</td>
</tr>
<tr>
<td>CreateWorksheetPart</td>
<td>Method to create a worksheet part as a raw XML string</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>EscapeXmlAttributeChars</td>
<td>Method to escape XML characters in an XML attribute</td>
</tr>
<tr>
<td>EscapeXmlChars</td>
<td>Method to escape XML characters between two tags</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform cleanup operations</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GeneratePasswordHash</td>
<td>Method to generate an Excel internal password hash to protect workbooks or</td>
</tr>
<tr>
<td></td>
<td>worksheets. This method is derived from the C++ implementation by Kohei</td>
</tr>
<tr>
<td></td>
<td>Yoshida (<a href="http://kohei.us/2008/01/excel-sheet-protection-password-hash/">http://kohei.us/2008/01/excel-sheet-protection-password-hash/</a>)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetOADateTimeString</td>
<td>Method to convert a date and time into the internal Excel time format (OAdate)</td>
</tr>
<tr>
<td>GetSortedSheetData</td>
<td>Method to sort the cells of a worksheet as preparation for the XML document</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>)</td>
</tr>
<tr>
<td>Save</td>
<td>Method to save the workbook</td>
</tr>
<tr>
<td>SaveAsStream</td>
<td>Method to save the workbook as a stream</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from <code>Object</code>)</td>
</tr>
</tbody>
</table>

**See Also**

- Reference
  - LowLevel Class
  - NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelAppendXmlTag Method

Method to append a simple XML tag with an enclosed value to the passed StringBuilder

**Namespace:** NanoXSX.LowLevel

**Assembly:** NanoXSX (in NanoXSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
private bool AppendXmlTag(
    StringBuilder sb,
    string value,
    string tagName,
    string nameSpace
)
```

**Parameters**

- **sb**
  - Type: `System.Text.StringBuilder`
  - StringBuilder to append

- **value**
  - Type: `System.String`
  - Value of the XML element

- **tagName**
  - Type: `System.String`
  - Tag name of the XML element

- **nameSpace**
  - Type: `System.String`
  - Optional XML name space. Can be empty or null

**Return Value**
Type: **Boolean**
Returns false if no tag was appended, because the value or tag name was null or empty

### See Also

**Reference**
- LowLevel Class
- NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelAppendXmlToPackagePart Method

Writes raw XML strings into the passed Package Part

**Namespace:** NanoXLSX.LowLevel

**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

▲ Syntax

```csharp
private void AppendXmlToPackagePart(
    string doc,
    PackagePart pp,
    string title
)
```

▲ Parameters

**doc**
Type: `System.String`
Document as raw XML string

**pp**
Type: `System.IO.Packaging.PackagePart`
Package part to append the XML data

**title**
Type: `System.String`
Title for interception / debugging purpose

▲ Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
</table>

IOException throws an IOException if the XML data could not be written into the Package Part

See Also

Reference
LowLevel Class
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelCreateAppPropertiesDocument Method

Method to create the app-properties (part of meta data) as raw XML string

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
private string CreateAppPropertiesDocument()
```

### Return Value

**Type:** String  
Raw XML string

## See Also

**Reference**  
LowLevel Class  
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelCreateAppString Method

Method to create the XML string for the app-properties document

**Namespace:** NanoXLXSX.LowLevel  
**Assembly:** NanoXLXSX (in NanoXLXSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
private string CreateAppString()
```

### Return Value

**Type:** String  
String with formatted XML data

### See Also

**Reference**  
LowLevel Class  
NanoXLXSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelCreateColsString Method

Method to create the columns as XML string. This is used to define the width of columns

Namespace: NanoXLsx.LowLevel
Assembly: NanoXLsx (in NanoXLsx.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
private string CreateColsString(
    Worksheet worksheet
)
```

Parameters

worksheet
Type: NanoXLsxWorksheet
Worksheet to process

Return Value
Type: String
String with formatted XML data

See Also

Reference
LowLevel Class
NanoXLsx.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelCreateCorePropertiesDocument Method

Method to create the core-properties (part of meta data) as raw XML string

Namespace: NanoXLSX.LowLevel
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```
private string CreateCorePropertiesDocument()
```

Return Value
Type: String
Raw XML string

See Also

Reference
LowLevel Class
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelCreateCorePropertiesString Method

Method to create the XML string for the core-properties document

Namespace: NanoXLSX.LowLevel
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
private string CreateCorePropertiesString()
```

Return Value

Type: String
String with formatted XML data

See Also

Reference
LowLevel Class
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelCreateMergedCellsString Method

Method to create the merged cells string of the passed worksheet

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private string CreateMergedCellsString(
    Worksheet sheet
)
```

### Parameters

**sheet**  
Type: NanoXLSXWorksheet  
Worksheet to process

### Return Value

Type: String  
Formatted string with merged cell ranges

### See Also

Reference  
LowLevel Class  
NanoXLSX.LowLevel Namespace

Copyright Raphaël Stoeckli © 2018
LowLevelCreateMruColorsString Method

Method to create the XML string for the color-MRU part of the style sheet document (recent colors)

Namespace:   NanoXLSX.LowLevel
Assembly:   NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```
private string CreateMruColorsString()
```

Return Value

Type:   String
String with formatted XML data

See Also

Reference

LowLevel Class
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelCreateRowString Method

Method to create a row string

**Namespace:** NanoXLSX.LowLevel
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private string CreateRowString(
    List<Cell> columnFields,
    Worksheet worksheet
)
```

### Parameters

- **columnFields**
  - Type: `System.Collections.Generic.List<Cell>`
  - List of cells
- **worksheet**
  - Type: `NanoXLSXWorksheet`
  - Worksheet to process

### Return Value

- Type: `String`
  - Formatted row string

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FormatException</code></td>
<td>Throws a <code>FormatException</code> if a handled date</td>
</tr>
</tbody>
</table>
cannot be translated to (Excel internal)
OADate

See Also

Reference
LowLevel Class
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelCreateSharedStringsDocument Method

Method to create shared strings as raw XML string

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
private string CreateSharedStringsDocument()
```

**Return Value**

Type: String  
Raw XML string

## See Also

Reference
LowLevel Class  
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelCreateSheetProtectionString Method

Method to create the protection string of the passed worksheet

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
private string CreateSheetProtectionString(
    Worksheet sheet
)
```

**Parameters**

*sheet*
  
  Type: NanoXLSXWorksheet  
  Worksheet to process

**Return Value**

Type: String  
Formatted string with protection statement of the worksheet

**See Also**

Reference
LowLevel Class  
NanoXLSX.LowLevel Namespace
LowLevelCreateStyleBorderString Method

Method to create the XML string for the border part of the style sheet document

**Namespace:** NanoXLXSX.LowLevel  
**Assembly:** NanoXLXSX (in NanoXLXSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private string CreateStyleBorderString()
```

### Return Value

**Type:** String  
String with formatted XML data

### See Also

**Reference**

[LowLevel Class](#)  
[NanoXLXSX.LowLevel Namespace](#)

Copyright Raphael Stoeckli © 2018
LowLevelCreateStyleFillString Method

Method to create the XML string for the fill part of the style sheet document

**Namespace:** NanoXLSX.LowLevel
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private string CreateStyleFillString()
```

### Return Value

Type: **String**
String with formatted XML data

### See Also

**Reference**
LowLevel Class
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelCreateStyleFontString Method

Method to create the XML string for the font part of the style sheet document

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private string CreateStyleFontString()
```

Return Value

Type: **String**  
String with formatted XML data

### See Also

Reference  
LowLevel Class  
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelCreateStyleNumberFormatString Method

Method to create the XML string for the number format part of the style sheet document

Namespace:  NanoXLSX.LowLevel
Assembly:  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```
private string CreateStyleNumberFormatString()
```

Return Value
Type:  String
String with formatted XML data

See Also

Reference
LowLevel Class
NanoXLSX.LowLevel Namespace
LowLevelCreateStyleSheetDocument Method

Method to create a style sheet as raw XML string

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
private string CreateStyleSheetDocument()
```

### Return Value

Type: String
Raw XML string

## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws an StyleException if one of the styles cannot be referenced or is null</td>
</tr>
</tbody>
</table>

## Remarks

The UndefinedStyleException should never happen in this state if the internally managed style collection was not tampered.

## See Also

Reference  
LowLevel Class
NanoXLSX Library Documentation
LowLevelCreateStyleXfsString Method

Method to create the XML string for the Xf part of the style sheet document

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private string CreateStyleXfsString()
```

**Return Value**

Type: **String**  
String with formatted XML data

### See Also

**Reference**

- LowLevel Class
- NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelCreateWorkbookDocument Method

Method to create a workbook as raw XML string

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

C#  
```csharp
private string CreateWorkbookDocument()
```

## Return Value

Type: String  
Raw XML string

## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeException</td>
<td>Throws an OutOfRangeException if an address was out of range</td>
</tr>
</tbody>
</table>

## See Also

Reference  
LowLevel Class  
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelCreateWorksheetPart Method

Method to create a worksheet part as a raw XML string

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
private string CreateWorksheetPart(
    Worksheet worksheet
)
```

**Parameters**

- `worksheet`
  - Type: NanoXLSXWorksheet  
  - worksheet object to process

**Return Value**

- Type: String  
  - Raw XML string

**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormatException</td>
<td>Throws a FormatException if a handled date cannot be translated to (Excel internal) OADate</td>
</tr>
</tbody>
</table>
See Also

Reference
LowLevel Class
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelEscapeXmlAttributeChars Method

Method to escape XML characters in an XML attribute

Namespace: NanoXLSX.LowLevel
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public static string EscapeXmlAttributeChars(
    string input
)
```

Parameters

`input`
Type: System.String
Input string to process

Return Value
Type: String
Escaped string

See Also

Reference
LowLevel Class
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelEscapeXmlChars Method

Method to escape XML characters between two XML tags

**Namespace:** NanoXLXS.LowLevel  
**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public static string EscapeXmlChars(
    string input
)
```

### Parameters

**input**
Type: `System.String`  
Input string to process

### Return Value

Type: `String`  
Escaped string

### Remarks

Note: The XML specs allow characters up to the character value of 0x10FFFF. However, the C# char range is only up to 0xFFFF. NanoXLXS will neglect all values above this level in the sanitizing check. Illegal characters like 0x1 will be replaced with a white space (0x20)
See Also

Reference
LowLevel Class
NanoXLXSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelGeneratePasswordHash Method

Method to generate an Excel internal password hash to protect workbooks or worksheets
This method is derived from the c++ implementation by Kohei Yoshida
(http://kohei.us/2008/01/18/excel-sheet-protection-password-hash/)

**Namespace:** NanoXLSX.LowLevel
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public static string GeneratePasswordHash(
    string password
)
```

Parameters

**password**
Type: System.String
Password string in UTF-8 to encrypt

Return Value
Type: String
16 bit hash as hex string

Remarks

WARNING! Do not use this method to encrypt 'real' passwords or data outside from NanoXLSX. This is only a minor security feature. Use a proper cryptography method instead.
See Also

Reference

LowLevel Class
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelGetOADateTimeString Method

Method to convert a date or date and time into the internal Excel time format (ODate)

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static string GetOADateTimeString(
    DateTime date,
    CultureInfo culture
)
```

### Parameters

- **date**
  - Type: `System.DateTime`
  - Date to process

- **culture**
  - Type: `System.Globalization.CultureInfo`
  - CultureInfo for proper formatting of the decimal point

### Return Value

- Type: `String`
  - Date or date and time as Number

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
</table>


**Remarks**

OA Date format starts at January 1st 1900 (actually 00.01.1900). Dates beyond this date cannot be handled by Excel under normal circumstances and will throw a FormatException.

**See Also**

- Reference
- LowLevel Class
- NanoXLSX.LowLevel Namespace
LowLevelGetSortedSheetData Method

Method to sort the cells of a worksheet as preparation for the XML document

**Namespace:** NanoXL SX.LowLevel  
**Assembly:** NanoXS X (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private List<List<Cell>> GetSortedSheetData(
    Worksheet sheet
)
```

### Parameters

**sheet**
Type: NanoXS XWorksheet  
Worksheet to process

### Return Value
Type: List<List<Cell>>  
Two dimensional array of Cell objects

### See Also

Reference
LowLevel Class  
NanoXLSX.LowLevel Namespace
LowLevelSave Method

Method to save the workbook

**Namespace:** NanoLSX.LowLevel  
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public void Save()
```

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>Throws IOException in case of an error</td>
</tr>
<tr>
<td>RangeException</td>
<td>Throws an OutOfRangeException if the start or end address of a handled cell range was out of range</td>
</tr>
<tr>
<td>FormatException</td>
<td>Throws a FormatException if a handled date cannot be translated to (Excel internal) OADate</td>
</tr>
<tr>
<td>StyleException</td>
<td>Throws a StyleException if one of the styles of the workbook cannot be referenced or is null</td>
</tr>
</tbody>
</table>

### Remarks

The StyleException should never happen in this state if the internally managed style collection was not tampered.
See Also

Reference

LowLevel Class
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelSaveAsStream Method

Method to save the workbook as stream

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public void SaveAsStream(
    Stream stream
)
```

### Parameters

- **stream**
  - **Type:** System.IO.Stream
  - Writable stream as target

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>Throws IOException in case of an error</td>
</tr>
<tr>
<td>RangeException</td>
<td>Throws an OutOfRangeException if the start or end address of a handled cell range was out of range</td>
</tr>
<tr>
<td>FormatException</td>
<td>Throws a FormatException if a handled date cannot be translated to (Excel internal) OADate</td>
</tr>
<tr>
<td>StyleException</td>
<td>Throws an StyleException if one of the</td>
</tr>
</tbody>
</table>
styles of the workbook cannot be referenced or is null

Remarks

The StyleException should never happen in this state if the internally managed style collection was not tampered.

See Also

Reference
LowLevel Class
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelDocumentPath Class

Class to manage XML document paths

Inheritance Hierarchy

```
SystemObject   NanoXMLX.LOWLEVELLowLevelDocumentPath
```

Namespace:  NanoXMLX.LowLevel
Assembly:  NanoXMLX (in NanoXMLX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```
public class DocumentPath
```

The LowLevelDocumentPath type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LowLevelDocumentPath</td>
<td>Default constructor</td>
</tr>
<tr>
<td>LowLevelDocumentPath(String, String)</td>
<td>Constructor with defined file name and path</td>
</tr>
</tbody>
</table>

Properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetFullPath</td>
<td>Method to return the full path of the document</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that</td>
</tr>
</tbody>
</table>
represents the current object. (Inherited from \texttt{Object}.)

\section*{See Also}

Reference

\texttt{NanoXLSX.LowLevel Namespace}

Copyright Raphael Stoeckli © 2018
## DocumentPath Constructor

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LowLevelDocumentPath</td>
<td>Default constructor</td>
</tr>
<tr>
<td>LowLevelDocumentPath(String, String)</td>
<td>Constructor with defined file name and path</td>
</tr>
</tbody>
</table>

### See Also

**Reference**
- LowLevelDocumentPath Class  
- NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelDocumentPath Constructor

Default constructor

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public DocumentPath()
```

### See Also

Reference

- LowLevelDocumentPath Class
- LowLevelDocumentPath Overload
- NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelDocumentPath Constructor (String, String)

Constructor with defined file name and path

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

▲ Syntax

```csharp
public DocumentPath(
    string filename,
    string path
)
```

Parameters

- **filename**
  - Type: System.String
  - File name of the document

- **path**
  - Type: System.String
  - Path of the document

▲ See Also

**Reference**  
LowLevelDocumentPath Class  
LowLevelDocumentPath Overload  
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
DocumentPath Properties

The `LowLevelDocumentPath` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename</td>
<td>File name of the document</td>
</tr>
<tr>
<td>Path</td>
<td>Path of the document</td>
</tr>
</tbody>
</table>

### See Also

Reference
- `LowLevelDocumentPath Class`
- `NanoXLSX.LowLevel Namespace`

Copyright Raphael Stoeckli © 2018
LowLevelDocumentPathFilename Property

File name of the document

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string Filename { get; set; }
```

**Property Value**

Type: **String**

### See Also

Reference

LowLevelDocumentPath Class  
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelDocumentPathPath Property

Path of the document

**Namespace:** NanoXLSP.LowLevel  
**Assembly:** NanoXLSP (in NanoXLSP.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```c#
public string Path { get; set; }
```

### Property Value

**Type:** String

### See Also

**Reference**
- LowLevelDocumentPath Class  
- NanoXLSP.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
## DocumentPath Methods

The `LowLevelDocumentPath` type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetFullPath</td>
<td>Method to return the full path of the document</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>
**ToString**

Returns a string that represents the current object. (Inherited from **Object**.)

---

**See Also**

Reference

LowLevelDocumentPath Class
NanoXLSX.LowLevel Namespace
LowLevelDocumentPath.GetFullPath Method

Method to return the full path of the document

**Namespace:** NanoXLSX.LowLevel
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string GetFullPath()
```

### Return Value

Type: **String**  
Full path

### See Also

Reference

LowLevelDocumentPath Class  
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelSortedMap Class

Class to manage key value pairs (string / string). The entries are in the order how they were added

Inheritance Hierarchy

- **SystemObject**  NanoXLSX.LowLevelLowLevelSortedMap

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public class SortedMap
```

The `LowLevelSortedMap` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LowLevelSortedMap</td>
<td>Default constructor</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Number of map entries</td>
</tr>
<tr>
<td>Item</td>
<td>Indexer to get the specific value by the</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Add</td>
<td>Adds a key value pair to the map. If the key already exists, only its index will be returned</td>
</tr>
<tr>
<td>ContainsKey</td>
<td>Gets whether the specified key exists in the map</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
</tbody>
</table>
(Inherited from `Object`.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>

See Also

Reference

*NanoXLSX.LowLevel* Namespace

Copyright Raphael Stoeckli © 2018
LowLevelSortedMap Constructor

Default constructor

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public SortedMap()
```

### See Also

**Reference**
- LowLevelSortedMap Class
- NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
## SortedMap Properties

The **LowLevelSortedMap** type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Number of map entries</td>
</tr>
<tr>
<td>Item</td>
<td>Indexer to get the specific value by the key</td>
</tr>
<tr>
<td>Keys</td>
<td>Gets the keys of the map as list</td>
</tr>
<tr>
<td>Values</td>
<td>Gets the values of the map as values</td>
</tr>
</tbody>
</table>

### See Also

Reference

- **LowLevelSortedMap Class**
- **NanoXLSX.LowLevel Namespace**

---

Copyright Raphael Stoeckli © 2018
LowLevelSortedMap Count Property

Number of map entries

**Namespace:** NanoXLXS.LowLevel  
**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

⚠️ Syntax

```
public int Count { get; }
```

Property Value

Type: **Int32**

⚠️ See Also

Reference  
LowLevelSortedMap Class  
NanoXLXS.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
LowLevelSortedMapItem Property

Indexer to get the specific value by the key

**Namespace:** NanoXL SX.LowLevel  
**Assembly:** NanoXL SX (in NanoXL SX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public string this[string key]
{
    get;
}
```

### Parameters

**key**
- Type: **System.String**
- Key to corresponding value. Returns null if not found

### Property Value

Type: **String**

### See Also

**Reference**
- [LowLevelSortedMap Class](#)
- [NanoXL SX.LowLevel Namespace](#)
LowLevelSortedMap.Keys Property

Gets the keys of the map as list

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public List<string> Keys { get; }
```

### Property Value

Type: List<String>

### See Also

Reference
- LowLevelSortedMap Class
- NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
**LowLevelSortedMapValues Property**

Gets the values of the map as values

**Namespace:** NanoXLXSX.LowLevel  
**Assembly:** NanoXLXSX (in NanoXLXSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public List<string> Values { get; }
```

Property Value  
Type: List<String>

### See Also

Reference  
LowLevelSortedMap Class  
NanoXLXSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
# SortedMap Methods

The **LowLevelSortedMap** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a key value pair to the map. If the key already exists, only its index will be returned</td>
</tr>
<tr>
<td>ContainsKey</td>
<td>Gets whether the specified key exists in the map</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
</tbody>
</table>
**MemberwiseClone**  Creates a shallow copy of the current Object.  
(Inherited from Object.)

**ToString**  Returns a string that represents the current object.  
(Inherited from Object.)

---

**See Also**

Reference

LowLevelSortedMap Class  
NanoXLSX.LowLevel Namespace

---

Copyright Raphael Stoeckli © 2018
LowLevelSortedMapAdd Method

Adds a key value pair to the map. If the key already exists, only its index will be returned.

**Namespace:** NanoXLSX.LowLevel
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public int Add(
    string key,
    string value
)
```

### Parameters

- **key**
  - Type: `System.String`
  - Key of the tuple

- **value**
  - Type: `System.String`
  - Value of the tuple

### Return Value

- Type: `Int32`
  - Position of the tuple in the map as index (zero-based)

### See Also

- Reference
  - LowLevelSortedMap Class
  - NanoXLSX.LowLevel Namespace
LowLevelSortedMapContainsKey Method

Gets whether the specified key exists in the map

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

⚠️ Syntax

```csharp
public bool ContainsKey(
    string key
)
```

**Parameters**

- **key**  
  Type: `System.String`  
  Key to check

**Return Value**

- Type: `Boolean`  
  True if the entry exists, otherwise false

⚠️ See Also

- Reference  
  LowLevelSortedMap Class  
  NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
Reader Class

[Missing <summary> documentation for "T:NanoXLSX.LowLevel.Reader"]

▲ Inheritance Hierarchy

System → Object → NanoXLSX.LowLevelReader

Namespace: NanoXLSX.LowLevel
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

▲ Syntax

```csharp
public class Reader
```

The Reader type exposes the following members.

▲ Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reader(Stream)</td>
<td>Constructor with stream as parameter</td>
</tr>
<tr>
<td>Reader(String)</td>
<td>Constructor with file path as parameter</td>
</tr>
</tbody>
</table>

▲ Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <em>Object</em> is equal to the current <em>Object</em>. (Inherited from <em>Object.</em>)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <em>Object.</em>)</td>
</tr>
<tr>
<td><strong>GetEntryStream</strong></td>
<td>Gets the memory stream of the specified file in the archive (XLSX file)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. (Inherited from <em>Object.</em>)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <em>Type</em> of the current instance. (Inherited from <em>Object.</em>)</td>
</tr>
<tr>
<td><strong>GetWorkbook</strong></td>
<td>Resolves the workbook with all worksheets from the loaded file</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <em>Object</em>. (Inherited from <em>Object.</em>)</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Reads the XLSX file from a file path or a file stream</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from <em>Object.</em>)</td>
</tr>
</tbody>
</table>
See Also

Reference
NanoXLSX.LowLevel Namespace
Reader Constructor

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reader(Stream)</td>
<td>Constructor with stream as parameter</td>
</tr>
<tr>
<td>Reader(String)</td>
<td>Constructor with file path as parameter</td>
</tr>
</tbody>
</table>

See Also

Reference
- Reader Class
- NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
Reader Constructor (Stream)

Constructor with stream as parameter

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

[C#]
```csharp
public Reader(  
    Stream stream  
)
```

### Parameters

**stream**

Type: `System.IO.Stream`  
Stream of the xlsx file to load

### See Also

- Reference
  - Reader Class
  - Reader Overload
  - NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
Reader Constructor (String)

Constructor with file path as parameter

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public Reader(
    string path
)
```

**Parameters**

`path`

Type: `System.String`
File path of the xlsx file to load

**See Also**

Reference
- Reader Class
- Reader Overload
- NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
## Reader Methods

The **Reader** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetEntryStream</td>
<td>Gets the memory stream of the specified file in the archive (XLSX file)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetWorkbook</td>
<td>Resolves the workbook with all worksheets from the loaded file</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Reads the XLSX file from a file path or a file stream</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
</tbody>
</table>

**See Also**

- Reference
- Reader Class
- NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
**ReaderGetEntryStream Method**

Gets the memory stream of the specified file in the archive (XLSX file)

**Namespace:** NanoXLXSX.LowLevel  
**Assembly:** NanoXLXSX (in NanoXLXSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private MemoryStream GetEntryStream(
    string name,
    ZipArchive archive
)
```

### Parameters

**name**  
Type: `System.String`  
Name of the XML file within the XLSX file

**archive**  
Type: `System.IO.Compression.ZipArchive`  
Zip file (XLSX)

### Return Value

Type: `MemoryStream`  
MemoryStream object of the specified file

### See Also

- Reference
  - Reader Class
  - NanoXLXSX.LowLevel Namespace
ReaderGetWorkbook Method

Resolves the workbook with all worksheets from the loaded file

**Namespace:**   NanoXLXS.LowLevel
**Assembly:**   NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Workbook GetWorkbook()
```

**Return Value**

Type: Workbook
Workbook object

### See Also

**Reference**

Reader Class
NanoXLXS.LowLevel Namespace
### Reader Read Method

Reads the XLSX file from a file path or a file stream

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

#### Syntax

```csharp
public void Read()
```

#### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>Throws IOException in case of an error</td>
</tr>
</tbody>
</table>

#### See Also

- **Reference**  
  - Reader Class  
  - NanoXLSX.LowLevel Namespace

---

Copyright Raphael Stoeckli © 2018
SharedStringsReader Class

Class representing a reader for the shared strings table of XLSX files

Inheritance Hierarchy

- System
  - Object
  - NanoXLSX.LowLevel.SharedStringsReader

Namespace: NanoXLSX.LowLevel
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public class SharedStringsReader
```

The `SharedStringsReader` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SharedStringsReader</td>
<td>Default constructor</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasElements</td>
<td>Gets whether the workbook contains shared strings</td>
</tr>
<tr>
<td>SharedStrings</td>
<td>List of shared string entries</td>
</tr>
</tbody>
</table>
### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetString</strong></td>
<td>Gets the value of the shared string table by its index</td>
</tr>
<tr>
<td><strong>GetTextToken</strong></td>
<td>Function collects text tokens recursively in case of a split by formatting</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Reads the XML file form the</td>
</tr>
</tbody>
</table>
passed stream and processes the shared strings table

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ToString</code></td>
<td>Returns a string that represents the current object. (Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>

**See Also**

- Reference
  - `NanoXLSX.LowLevel Namespace`

Copyright Raphael Stoeckli © 2018
SharedStringsReader Constructor

Default constructor

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public SharedStringsReader()
```

### See Also

**Reference**  
SharedStringsReader Class  
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
SharedStringsReader Properties

The `SharedStringsReader` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasElements</td>
<td>Gets whether the workbook contains shared strings</td>
</tr>
<tr>
<td>SharedStrings</td>
<td>List of shared string entries</td>
</tr>
</tbody>
</table>

## See Also

Reference

- `SharedStringsReader Class`
- `NanoXLSX.LowLevel Namespace`
SharedStringsReader.HasElements Property

Gets whether the workbook contains shared strings

**Namespace:** NanoXLXS.LowLevel  
**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public bool HasElements { get; }
```

**Property Value**

Type: Boolean  
True if at least one shared string object exists in the workbook

**See Also**

Reference  
SharedStringsReader Class  
NanoXLXS.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
SharedStringsReader

Property

List of shared string entries

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public List<string> SharedStrings { get; private set; }
```

### Property Value

Type: **ListString**  
String entry, sorted by its internal index of the table

### See Also

Reference
- **SharedStringsReader Class**
- **NanoXLSX.LowLevel Namespace**
**SharedStringsReader Methods**

The **SharedStringsReader** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>GetString</td>
<td>Gets the value of the shared string table by its index.</td>
</tr>
<tr>
<td>GetTextToken</td>
<td>Function collects text tokens recursively in case of a split by formatting</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td>Read</td>
<td>Reads the XML file form the passed stream and processes the shared strings table</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object.</td>
</tr>
</tbody>
</table>

**See Also**

Reference

[SharedStringsReader Class](#)

[NanoXLSX.LowLevel Namespace](#)
**SharedStringsReader.GetString Method**

Gets the value of the shared string table by its index

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string GetString(
    int index
)
```

### Parameters

- **index**
  
  Type: `System.Int32`  
  
  Index of the stared string entry

### Return Value

- **Type:** `String`  
  
  Determined shared string value. Returns null in case of a invalid index

### See Also

**Reference**  
- [SharedStringsReader Class](#)  
- [NanoXLSX.LowLevel Namespace](#)
SharedStringsReaderGetTextToken Method

Function collects text tokens recursively in case of a split by formatting

Namespace: NanoXLSX.LowLevel
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
private void GetTextToken(
    XmlNode node,
    ref StringBuilder sb
)
```

Parameters

`node`
Type: System.Xml.XmlNode
Root node to process

`sb`
Type: System.Text.StringBuilder
StringBuilder reference

See Also

Reference
SharedStringsReader Class
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
SharedStringsReader.Read Method

Reads the XML file from the passed stream and processes the shared strings table

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public void Read(
    MemoryStream stream
)
```

### Parameters

*stream*
- Type: `System.IO.MemoryStream`  
  Stream of the XML file

## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>Throws IOException in case of an error</td>
</tr>
</tbody>
</table>

## See Also

- Reference
  - SharedStringsReader Class
  - NanoXLSX.LowLevel Namespace
WorkbookReader Class

Class representing a reader to decompile a workbook in an XLSX files

Inheritance Hierarchy

- `System.Object`
- `NanoXLSX.LowLevel.WorkbookReader`

Namespace: `NanoXLSX.LowLevel`
Assembly: `NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)`

Syntax

```csharp
public class WorkbookReader
```

The `WorkbookReader` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>WorkbookReader</code></td>
<td>Default constructor</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>WorksheetDefinitions</code></td>
<td>Dictionary of worksheet definitions. The key is the worksheet number and the value is the worksheet</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetWorkbookInformation</strong></td>
<td>Finds the workbook information recursively</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
</tbody>
</table>
Read
Reads the XML file form the passed stream and processes the workbook information

ToString
Returns a string that represents the current object. (Inherited from Object.)

See Also
Reference
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
WorkbookReader Constructor

Default constructor

Namespace: NanoXLSX.LowLevel
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```
public WorkbookReader()
```

See Also

Reference

WorkbookReader Class
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
# WorkbookReader Properties

The **WorkbookReader** type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Worksheet Definitions" /></td>
<td>Dictionary of worksheet definitions. The key is the worksheet number and the value is the worksheet name</td>
</tr>
</tbody>
</table>

## See Also

Reference
- **WorkbookReader Class**
- **NanoXLSX.LowLevel Namespace**

Copyright Raphael Stoeckli © 2018
WorkbookReaderWorksheetDefinitions

Property

Dictionary of worksheet definitions. The key is the worksheet number and the value is the worksheet name

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Dictionary<int, string> WorksheetDefinitions
```

### Property Value

**Type:** Dictionary

Dictionary with worksheet definitions

### See Also

**Reference**

WorkbookReader Class  
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
# WorkbookReader Methods

The WorkbookReader type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type.</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>GetWorkbookInformation</strong></td>
<td>Finds the workbook information recursively</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy</td>
</tr>
</tbody>
</table>
of the current Object.
(Inherited from Object.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>Reads the XML file form the passed stream and processes the workbook information</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

See Also

Reference
WorkbookReader Class
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
WorkbookReader.GetWorkbookInformation Method

Finds the workbook information recursively

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private void GetWorkbookInformation(
    XmlNode node
)
```

### Parameters

- **node**
  - Type: `System.Xml.XmlNode`
  - Root node to check

### See Also

**Reference**
- WorkbookReader Class
- NanoXLSX.LowLevel Namespace

---

Copyright Raphael Stoeckli © 2018
WorkbookReader Read Method

Reads the XML file form the passed stream and processes the workbook information

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void Read(
    MemoryStream stream
)
```

### Parameters

`stream`  
Type: System.IO.MemoryStream  
Stream of the XML file

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>Throws IOException in case of an error</td>
</tr>
</tbody>
</table>

### See Also

**Reference**
- WorkbookReader Class
- NanoXLSX.LowLevel Namespace
WorksheetReader Class

Class representing a reader for worksheets of XLSX files

Inheritance Hierarchy

```
SystemObject  NanoXLSX.LowLevelWorksheetReader
```

Namespace:  NanoXLSX.LowLevel
Assembly:  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
public class WorksheetReader
```

The `WorksheetReader` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorksheetReader</td>
<td>Constructor with parameters</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Gets the data of the worksheet as Dictionary of cell address-cell object tuples</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Name</td>
<td>Gets the name of the worksheet</td>
</tr>
<tr>
<td>WorksheetNumber</td>
<td>Gets the number of the worksheet</td>
</tr>
</tbody>
</table>

**Top**

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetAttribute</td>
<td>Gets the attribute with the passed name.</td>
</tr>
<tr>
<td>GetBooleanValue</td>
<td>Parses the boolean value of a raw cell.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetIntValue</td>
<td>Parses the integer value of a raw cell.</td>
</tr>
<tr>
<td>GetNumericValue</td>
<td>Parses the numeric (double)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>GetRow</strong></td>
<td>Gets a row as list of cell objects</td>
</tr>
<tr>
<td><strong>GetRowCount</strong></td>
<td>Gets the number of rows</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>HasColumn</strong></td>
<td>Gets whether the specified column exists in the data</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Reads the XML file form the passed stream and processes the worksheet data</td>
</tr>
<tr>
<td><strong>ResolveCellData</strong></td>
<td>Resolves the data of a read cell, transforms it into a cell object and adds it to the data</td>
</tr>
<tr>
<td><strong>RowHasColumns</strong></td>
<td>Gets whether the passed row (represented as list of cell objects) contains the specified column numbers</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
See Also

Reference
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
WorksheetReader Constructor

Constructor with parameters

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public WorksheetReader(  
    SharedStringsReader sharedStrings,  
    string name,  
    int number
)
```

### Parameters

- **sharedStrings**  
  Type: NanoXLSX.LowLevel.SharedStringsReader  

- **name**  
  Type: System.String  

- **number**  
  Type: System.Int32  

### See Also

Reference
## WorksheetReader Properties

The **WorksheetReader** type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Image] Data</td>
<td>Gets the data of the worksheet as Dictionary of cell address-cell object tuples</td>
</tr>
<tr>
<td>![Image] Name</td>
<td>Gets the name of the worksheet</td>
</tr>
<tr>
<td>![Image] WorksheetNumber</td>
<td>Gets the number of the worksheet</td>
</tr>
</tbody>
</table>

### See Also

- **Reference**
  - WorksheetReader Class
  - NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
WorksheetReaderData Property

Gets the data of the worksheet as Dictionary of cell address-cell object tuples

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Dictionary<string, Cell> Data { get; private set; }
```

### Property Value

Type: Dictionary<String, Cell>  
Dictionary of cell address-cell object tuples

### See Also

Reference  
WorksheetReader Class  
NanoXLSX.LowLevel Namespace

Copyright Raphaël Stoeckli © 2018
WorksheetReaderName Property

Gets the name of the worksheet

**Namespace:** NanoXLXSX.LowLevel  
**Assembly:** NanoXLXSX (in NanoXLXSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public string Name { get; private set; }
```

### Property Value

**Type:** String  
**Name of the worksheet**

### See Also

**Reference**  
WorksheetReader Class  
NanoXLXSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
WorksheetReader.WorksheetNumber Property

Gets the number of the worksheet

**Namespace:** NanoXLXS.LOWLEVEL  
**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public int WorksheetNumber { get; private set; }
```

**Property Value**

- **Type:** Int32
- Number of the worksheet

### See Also

**Reference**

- WorksheetReader Class
- NanoXLXS.LOWLEVEL Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
## WorksheetReader Methods

The **WorksheetReader** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetAttribute</strong></td>
<td>Gets the attribute with the passed name.</td>
</tr>
<tr>
<td><strong>GetBooleanValue</strong></td>
<td>Parses the boolean value of a raw cell</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetIntValue</strong></td>
<td>Parses the integer value of a raw cell</td>
</tr>
<tr>
<td><strong>GetNumericValue</strong></td>
<td>Parses the numeric (double) value of a raw cell</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetRow</td>
<td>Gets a row as list of cell objects</td>
</tr>
<tr>
<td>GetRowCount</td>
<td>Gets the number of rows</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>HasColumn</td>
<td>Gets whether the specified column exists in the data</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Read</td>
<td>Reads the XML file from the passed stream and processes the worksheet data</td>
</tr>
<tr>
<td>ResolveCellData</td>
<td>Resolves the data of a read cell, transforms it into a cell object and adds it to the data</td>
</tr>
<tr>
<td>RowHasColumns</td>
<td>Gets whether the passed row (represented as list of cell objects) contains the specified column numbers</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>

**See Also**
Reference
WorksheetReader Class
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
WorksheetReader

Get Attribute Method

Gets the attribute with the passed name.

**Namespace:** NanoXLXSX.LowLevel  
**Assembly:** NanoXLXSX (in NanoXLXSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
private string GetAttribute(
    string targetName,  
    XmlNode node,  
    string defaultValue
)
```

### Parameters

- **targetName**
  - Type: **System.String**
  - Name of the target attribute

- **node**
  - Type: **System.Xml.XmlNode**
  - XML node that contains the attribute

- **defaultValue**
  - Type: **System.String**
  - Default value if the attribute was not found

### Return Value

- **Type:** **String**
  - Attribute value as string or default value if not found (can be null)
See Also

Reference

WorksheetReader Class
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
WorksheetReader.GetBooleanValue Method

Parses the boolean value of a raw cell

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private static WorksheetReaderCellResolverTuple GetBooleanValue (string raw)
```

### Parameters

**raw**
- Type: `System.String`  
- Raw value as string

### Return Value

- Type: `WorksheetReaderCellResolverTuple`  
- `CellResolverTuple` with information about the validity and resolved data

### See Also

**Reference**  
- `WorksheetReader Class`  
- `NanoXLSX.LowLevel Namespace`

Copyright Raphael Stoeckli © 2018
WorksheetReaderGetIntValue Method

Parses the integer value of a raw cell

**Namespace:** NanoXLXS.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

▲ Syntax

```csharp
private static WorksheetReaderCellResolverTuple (string raw)
```

**Parameters**

`raw`  
Type: `System.String`  
Raw value as string

**Return Value**

Type: `WorksheetReaderCellResolverTuple`  
CellResolverTuple with information about the validity and resolved data

▲ See Also

**Reference**  
WorksheetReader Class  
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
WorksheetReader.GetNumericValue Method

Parses the numeric (double) value of a raw cell

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
private static WorksheetReaderCellResolverTuple GetNumericValue (  
    string raw
)
```

Parameters

`raw`

Type: `System.String`  
Raw value as string

Return Value

Type: `WorksheetReaderCellResolverTuple`  
CellResolverTuple with information about the validity and resolved data

See Also

Reference  
WorksheetReader Class  
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
WorksheetReader.GetRow Method

Gets a row as list of cell objects

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public List<Cell> GetRow(int rowNumber)
```

### Parameters

* **rowNumber**
  - Type: `System.Int32`
  - Row number

### Return Value

- **Type:** `List<Cell>`
- List of cell objects

### See Also

- Reference
  - WorksheetReader Class
  - NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
WorksheetReader.GetRowCount Method

Gets the number of rows

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public int GetRowCount()
```

**Return Value**
Type: **Int32**  
Number of rows

### See Also

**Reference**
- **WorksheetReader Class**
- **NanoXLSX.LowLevel Namespace**

Copyright Raphael Stoeckli © 2018
WorksheetReader HasColumn Method

Gets whether the specified column exists in the data

**Namespace:** NanoXLXS.LownLevel
**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public bool HasColumn(
    string columnAddress
)
```

### Parameters

- **columnAddress**
  - Type: **System.String**
  - Column address as string

### Return Value

- Type: **Boolean**
- Column address as string

### See Also

- **Reference**
  - WorksheetReader Class
  - NanoXLXS.LownLevel Namespace

Copyright Raphael Stoeckli © 2018
WorksheetReader.Read Method

Reads the XML file form the passed stream and processes the worksheet data

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public void Read(MemoryStream stream)
```

### Parameters

**stream**  
Type: System.IO.MemoryStream  
Stream of the XML file

## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>Throws IOException in case of an error</td>
</tr>
</tbody>
</table>

## See Also

**Reference**  
WorksheetReader Class  
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
WorksheetReaderResolveCellData Method

Resolves the data of a read cell, transforms it into a cell object and adds it to the data

Namespace:  NanoXLXSX.LowLevel
Assembly:  NanoXLXSX (in NanoXLXSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
private void ResolveCellData(
    string address,
    string type,
    string value,
    string style,
    string formula
)
```

Parameters

address
Type: SystemString
Address of the cell

type
Type: SystemString
Expected data type

value
Type: SystemString
Raw value as string

style
Type: SystemString
Style definition as string (can be null)

**formula**
Type: **SystemString**
Formula as string (can be null; data type determines whether value or formula is used)

See Also

Reference
- WorksheetReader Class
- NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
WorksheetReaderRowHasColumns Method

Gets whether the passed row (represented as list of cell objects) contains the specified column numbers

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public bool RowHasColumns(
    List<Cell> cells,
    int[] columnNumbers
)
```

### Parameters

- **cells**  
  Type: `System.Collections.Generic.List<Cell>`  
  List of cell objects to check.

- **columnNumbers**  
  Type: `System.Int32`  
  Array of column numbers

### Return Value

Type: `Boolean`  
True if all column numbers were found, otherwise false

### See Also

- Reference
WorksheetReaderCellResolverTuple


Inheritance Hierarchy

- System
- Object
- NanoXLSX.LowLevelWorksheetReaderCellResolverTuple

Namespace: NanoXLSX.LowLevel
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
private class CellResolverTuple
```

The WorksheetReaderCellResolverTuple type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorksheetReaderCellResolverTuple</td>
<td>Default constructor with parameters</td>
</tr>
</tbody>
</table>

Properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Gets the data as object</td>
</tr>
<tr>
<td>DataType</td>
<td>Gets the type of the cell</td>
</tr>
<tr>
<td>IsValid</td>
<td>Gets whether the cell is valid</td>
</tr>
</tbody>
</table>

**Top**

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified <em>Object</em> is equal to the current <em>Object</em>. (Inherited from <em>Object</em>.)*</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <em>Object</em>.)*</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from <em>Object</em>.)*</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <em>Type</em> of the current instance. (Inherited from <em>Object</em>.)*</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <em>Object</em>. (Inherited from <em>Object</em>.)*</td>
</tr>
</tbody>
</table>
**ToString**

Returns a string that represents the current object. (Inherited from **Object**.)

---

**See Also**

Reference

NanoXLSX.LowLevel Namespace
WorksheetReaderCellResolverTuple Constructor

Default constructor with parameters

**Namespace:** NanoXLXS.LowLevel  
**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public CellResolverTuple(  
    bool isValid,  
    Object data,  
    Type type  
)
```

## Parameters

**isValid**  
Type: System.Boolean  
If true, the resolved cell contains valid data

**data**  
Type: System.Object  
Data object.

**type**  
Type: System.Type  
Type of the cell

## See Also

Reference  
WorksheetReaderCellResolverTuple Class
CellResolverTuple Properties

The WorksheetReaderCellResolverTuple type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Gets the data as object</td>
</tr>
<tr>
<td>DataType</td>
<td>Gets the type of the cell</td>
</tr>
<tr>
<td>IsValid</td>
<td>Gets whether the cell is valid</td>
</tr>
</tbody>
</table>

See Also

Reference
WorksheetReaderCellResolverTuple Class
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
WorksheetReaderCellResolverTuple Property

Gets the data as object

**Namespace:** NanoXLSX.LowLevel  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Object Data { get; private set; }
```

### Property Value

Type: `Object`  
Generic object

### See Also

Reference  
WorksheetReaderCellResolverTuple Class  
NanoXLSX.LowLevel Namespace
WorksheetReaderCellResolverTuple Property

Gets the type of the cell

**Namespace:** NanoXLSX.LowLevel

**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Type DataType { get; private set; }
```

**Property Value**

Type: **Type**

Data type

### See Also

**Reference**

WorksheetReaderCellResolverTuple Class  
NanoXLSX.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
WorksheetReaderCellResolverTuple

Property

Gets whether the cell is valid

**Namespace:** NanoXLXS.LowLevel

**Assembly:** NanoXLXS (in NanoXLXS.dll) Version: 1.2.3.0 (1.2.3)

#### Syntax

```c#
public bool IsValid { get; private set; }
```

Property Value

Type: **Boolean**

True if valid, otherwise false

#### See Also

**Reference**

- WorksheetReaderCellResolverTuple Class
- NanoXLXS.LowLevel Namespace

Copyright Raphael Stoeckli © 2018
CellResolverTuple Methods

The **WorksheetReaderCellResolverTuple** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object.</td>
</tr>
</tbody>
</table>
See Also

Reference
WorksheetReaderCellResolverTuple Class
NanoXLSX.LowLevel Namespace
# NanoXLSX.Style Namespace

[Missing `<summary>` documentation for "N: NanoXLSX.Style"]

## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbstractStyle</td>
<td>Class represents an abstract style component</td>
</tr>
<tr>
<td>AbstractStyleAppendAttribute</td>
<td>Attribute designated to control the copying of style properties</td>
</tr>
<tr>
<td>Style</td>
<td>Class representing a Style with sub classes within a style sheet. An instance of this class is only a container for the different sub-classes. These sub-classes contain the actual styling information.</td>
</tr>
<tr>
<td>StyleBasicStyles</td>
<td>Factory class with the most important predefined styles</td>
</tr>
<tr>
<td>StyleBorder</td>
<td>Class representing</td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>StyleCellXf</td>
<td>Class representing an XF entry. The XF entry is used to make reference to</td>
</tr>
<tr>
<td></td>
<td>other style instances like Border or Fill and for the positioning of the</td>
</tr>
<tr>
<td></td>
<td>cell content</td>
</tr>
<tr>
<td>StyleFill</td>
<td>Class representing a Fill (background) entry. The Fill entry is used to</td>
</tr>
<tr>
<td></td>
<td>define background colors and fill patterns</td>
</tr>
<tr>
<td>StyleFont</td>
<td>Class representing a Font entry. The Font entry is used to define text</td>
</tr>
<tr>
<td></td>
<td>formatting</td>
</tr>
<tr>
<td>StyleNumberFormat</td>
<td>Class representing a NumberFormat entry. The NumberFormat entry is used to</td>
</tr>
<tr>
<td></td>
<td>define cell formats like currency or date</td>
</tr>
</tbody>
</table>
StyleManager

Class representing a style manager to maintain all styles and its components of a workbook

Enumerations

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleBasicStylesStyleEnum</td>
<td>Enum with style selection</td>
</tr>
<tr>
<td>StyleBorderStyleValue</td>
<td>Enum for the border style</td>
</tr>
<tr>
<td>StyleCellXfHorizontalAlignValue</td>
<td>Enum for the horizontal alignment of a cell</td>
</tr>
<tr>
<td>StyleCellXfTextBreakValue</td>
<td>Enum for text break options</td>
</tr>
<tr>
<td>StyleCellXfTextDirectionValue</td>
<td>Enum for the general text alignment direction</td>
</tr>
<tr>
<td>StyleCellXfVerticalAlignValue</td>
<td>Enum for the vertical alignment of a cell</td>
</tr>
<tr>
<td>Enum</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td><strong>StyleFillFillType</strong></td>
<td>Enum for the type of the color</td>
</tr>
<tr>
<td><strong>StyleFillPatternValue</strong></td>
<td>Enum for the pattern values</td>
</tr>
<tr>
<td><strong>StyleFontSchemeValue</strong></td>
<td>Enum for the font scheme</td>
</tr>
<tr>
<td><strong>StyleFontVerticalAlignValue</strong></td>
<td>Enum for the vertical alignment of the text from base line</td>
</tr>
<tr>
<td><strong>StyleNumberFormatFormatNumber</strong></td>
<td>Enum for predefined number formats</td>
</tr>
</tbody>
</table>

Copyright Raphael Stoeckli © 2018
AbstractStyle Class

Class represents an abstract style component

**Inheritance Hierarchy**

- `System.Object`
- `NanoXLSX.Style.AbstractStyle`

More...

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public abstract class AbstractStyle : IComparable, IEquatable<AbstractStyle>
```

The `AbstractStyle` type exposes the following members.

**Constructors**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AbstractStyle</code></td>
<td>Initializes a new instance of the <code>AbstractStyle</code> class</td>
</tr>
</tbody>
</table>

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Hash</code></td>
<td>Gets the unique hash of the object</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InternalID</td>
<td>Gets or sets the internal ID for sorting purpose in the Excel style document (nullable)</td>
</tr>
<tr>
<td>CalculateHash</td>
<td>Abstract method definition to calculate the hash of the component</td>
</tr>
<tr>
<td>CastValue</td>
<td>Method to cast values of the components to string values for the hash calculation (protected/internal static method)</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Method to compare two objects for sorting purpose</td>
</tr>
<tr>
<td>Copy</td>
<td>Abstract method to copy a component (dereferencing)</td>
</tr>
<tr>
<td>CopyPropertiesT</td>
<td>Internal method to copy altered properties from a source object. The decision whether a property is copied is dependent on a untouched reference object</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Equals(Object)</td>
<td>Determines whether the specified <a href="#">Object</a> is equal to the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td>Equals(AbstractStyle)</td>
<td>Method to compare two objects for sorting purpose</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <a href="#">Type</a> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <a href="#">Object</a>.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <a href="#">Object</a>.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference
NanoXLSX.Style Namespace

Inheritance Hierarchy

System.Object  
   NanoXLSX.StyleAbstractStyle  
      NanoXLSX.StyleStyle  
      NanoXLSX.StyleStyleBorder  
      NanoXLSX.StyleStyleCellXf  
      NanoXLSX.StyleStyleFill  
      NanoXLSX.StyleStyleFont  
      NanoXLSX.StyleStyleNumberFormat

Copyright Raphael Stoeckli © 2018
AbstractStyle Constructor

Initializes a new instance of the AbstractStyle class

**Namespace:** NanoXLSX.Style
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
protected AbstractStyle()
```

### See Also

Reference
- AbstractStyle Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
AbstractStyle Properties

The AbstractStyle type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hash</td>
<td>Gets the unique hash of the object</td>
</tr>
<tr>
<td>InternalID</td>
<td>Gets or sets the internal ID for sorting purpose in the Excel style document (nullable)</td>
</tr>
</tbody>
</table>

## See Also

Reference

- AbstractStyle Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
AbstractStyleHash Property

Gets the unique hash of the object

**Namespace:** NanoXLXSX.Style  
**Assembly:** NanoXLXSX (in NanoXLXSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public string Hash { get; }
```

### Property Value

Type: **String**

### See Also

**Reference**  
AbstractStyle Class  
NanoXLXSX.Style Namespace

Copyright Raphael Stoeckli © 2018
AbstractStyleInternalID Property

Gets or sets the internal ID for sorting purpose in the Excel style document (nullable)

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public Nullable<int> InternalID { get; set; }
```

Property Value
Type: NullableInt32

See Also

Reference
AbstractStyle Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
AbstractStyle Methods

The **AbstractStyle** type exposes the following members.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalculateHash</td>
<td>Abstract method definition to calculate the hash of the component</td>
</tr>
<tr>
<td>CastValue</td>
<td>Method to cast values of the components to string values for the hash calculation (protected/internal static method)</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Method to compare two objects for sorting purpose</td>
</tr>
<tr>
<td>Copy</td>
<td>Abstract method to copy a component (dereferencing)</td>
</tr>
<tr>
<td>CopyPropertiesT</td>
<td>Internal method to copy altered properties from a source object. The decision whether a property is copied is dependent on a untouched reference object</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Equals(Object)</td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Equals(AbstractStyle)</td>
<td>Method to compare two objects for sorting purpose</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

See Also

Reference
AbstractStyleCalculateHash Method

Abstract method definition to calculate the hash of the component

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

#### Syntax

```csharp
public abstract string CalculateHash()
```

**Return Value**

**Type:** String  
Returns the hash of the component as string

#### See Also

Reference

- AbstractStyle Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
AbstractStyleCastValue Method

Method to cast values of the components to string values for the hash calculation (protected/internal static method)

Namespace: NanoXLXSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
protected static void CastValue(
    Object o,
    ref StringBuilder sb,
    Nullable<char> delimiter
)
```

Parameters

- **o**
  Type: SystemObject
  Value to cast

- **sb**
  Type: System.TextStringBuilder
  StringBuilder reference to put the casted object in

- **delimiter**
  Type: SystemNullableChar
  Delimiter character to append after the casted value

See Also

Reference
AbstractStyle Class
NanoXLSX.Style Namespace
AbstractStyle CompareTo Method

Method to compare two objects for sorting purpose

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public int CompareTo(
    AbstractStyle other
)
```

### Parameters

*other*

Type: NanoXLSX.StyleAbstractStyle  
Other object to compare with this object

### Return Value

Type: Int32  
-1 if the other object is bigger. 0 if both objects are equal. 1 if the other object is smaller.

### Implements

IComparable<T> CompareTo(T)

### See Also

**Reference**  
AbstractStyle Class  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
AbstractStyleCopy Method

Abstract method to copy a component (dereferencing)

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public abstract AbstractStyle Copy()
```

**Return Value**

*Type:* AbstractStyle  
Returns a copied component

### See Also

**Reference**  
AbstractStyle Class  
NanoXLSX.Style Namespace
AbstractStyleCopyProperties<T> Method

Internal method to copy altered properties from a source object. The decision whether a property is copied is dependent on a untouched reference object.

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
internal void CopyProperties<T>(
    T source,
    T reference
)
where T : AbstractStyle
```

Parameters

- **source**
  - Type: `T`
  - Source object with properties to copy
- **reference**
  - Type: `T`
  - Reference object to decide whether the properties from the source objects are altered or not

Type Parameters

- `T`
  - Style or sub-class of Style that extends AbstractStyle
See Also

Reference

AbstractStyle Class
NanoXLX.Style Namespace

Copyright Raphael Stoeckli © 2018
# AbstractStyleEquals Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Equals(Object)</code></td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>Equals(AbstractStyle)</code></td>
<td>Method to compare two objects for sorting purpose</td>
</tr>
</tbody>
</table>

## See Also

Reference
- AbstractStyle Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
AbstractStyle.Equals Method (AbstractStyle)

Method to compare two objects for sorting purpose

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public bool Equals(
    AbstractStyle other
)
```

### Parameters

**other**
- Type: NanoXLSX.StyleAbstractStyle
- Other object to compare with this object

### Return Value
- Type: Boolean
- True if both objects are equal, otherwise false

### Implements
- IEquatable<T> Equals(T)

### See Also

Reference
- AbstractStyle Class
- Equals Overload
- NanoXLSX.Style Namespace
AbstractStyleAppendAttribute Class

Attribute designated to control the copying of style properties

Inheritance Hierarchy

System\Object  System\Attribute
NanoXLSX.\Style\AbstractStyleAppendAttribute

Namespace: NanoXLSX.\Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

C#  

```csharp
public class AppendAttribute : Attribute
```

The AbstractStyleAppendAttribute type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbstractStyleAppendAttribute</td>
<td>Default constructor</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignore</td>
<td>Indicates whether the property</td>
</tr>
</tbody>
</table>
annotated with the attribute is ignored during the copying of properties

<table>
<thead>
<tr>
<th>NestedProperty</th>
<th>Indicates whether the property annotated with the attribute is a nested property. Nested properties are ignored but during the copying of properties but can be broken down to its sub-properties</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>TypId</th>
<th>When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)</th>
</tr>
</thead>
</table>

**Top**

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns the hash code for this instance. (Inherited from Attribute.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>IsDefaultAttribute</td>
<td>When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from <strong>Attribute</strong>.)</td>
</tr>
<tr>
<td>Match</td>
<td>When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from <strong>Attribute</strong>.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference

NanoXLSX.Style Namespace

SystemAttribute
NanoXLSX Library Documentation
AbstractStyleAppendAttribute

Constructor

Default constructor

**Namespace:** NanoXLXSX.Style
**Assembly:** NanoXLXSX (in NanoXLXSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public AppendAttribute()
```

### See Also

**Reference**
- AbstractStyleAppendAttribute Class
- NanoXLXSX.Style Namespace

Copyright Raphael Stoeckli © 2018
# AppendAttribute Properties

The `AbstractStyleAppendAttribute` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignore</td>
<td>Indicates whether the property annotated with the attribute is ignored during the copying of properties</td>
</tr>
<tr>
<td>NestedProperty</td>
<td>Indicates whether the property annotated with the attribute is a nested property. Nested properties are ignored but during the copying of properties but can be broken down to its sub-properties</td>
</tr>
<tr>
<td>TypId</td>
<td>When implemented in a derived class, gets a unique identifier for this <code>Attribute</code>. (Inherited from <code>Attribute</code>.)</td>
</tr>
</tbody>
</table>

See Also

Reference

- `AbstractStyleAppendAttribute Class`
- `NanoXLSX.Style Namespace`
AbstractStyleAppendAttributeIgnore Property

Indicates whether the property annotated with the attribute is ignored during the copying of properties

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public bool Ignore { get; set; }
```

**Property Value**

Type: `Boolean`

- `true` if ignored, otherwise `false`.

### See Also

- **Reference**
  - AbstractStyleAppendAttribute Class
  - NanoXLSX.Style Namespace
AbstractStyleAppendAttributeNested Property

Indicates whether the property annotated with the attribute is a nested property. Nested properties are ignored but during the copying of properties but can be broken down to its sub-properties

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public bool NestedProperty { get; set; }
```

**Property Value**
- Type: `Boolean`
- `true` if a nested property, otherwise `false`.

### See Also

- Reference
  - AbstractStyleAppendAttribute Class
  - NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
## AppendAttribute Methods

The **AbstractStyleAppendAttribute** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Returns the hash code for this instance. (Inherited from Attribute.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>IsDefaultAttribute</strong></td>
<td>When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Match</strong></td>
<td>When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from <strong>Attribute</strong>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference

- AbstractStyle
- AppendAttribute
- Class

**NanoXLSX.Style Namespace**
Style Class

Class representing a Style with sub classes within a style sheet. An instance of this class is only a container for the different sub-classes. These sub-classes contain the actual styling information.

Inheritance Hierarchy

SystemObject NanoXLSX.StyleAbstractStyle NanoXLSX.Style

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public class Style : AbstractStyle
```

The Style type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Style</td>
<td>Default constructor</td>
</tr>
<tr>
<td>Style(String)</td>
<td>Constructor with parameters</td>
</tr>
<tr>
<td>Style(String, Int32, Boolean)</td>
<td>Constructor with parameters (internal use)</td>
</tr>
</tbody>
</table>

Top
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentBorder</td>
<td>Gets or sets the current Border object of the style</td>
</tr>
<tr>
<td>CurrentCellXf</td>
<td>Gets or sets the current CellXf object of the style</td>
</tr>
<tr>
<td>CurrentFill</td>
<td>Gets or sets the current Fill object of the style</td>
</tr>
<tr>
<td>CurrentFont</td>
<td>Gets or sets the current Font object of the style</td>
</tr>
<tr>
<td>CurrentNumberFormat</td>
<td>Gets or sets the current NumberFormat object of the style</td>
</tr>
<tr>
<td>Hash</td>
<td>Gets the unique hash of the object (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>InternalID</td>
<td>Gets or sets the internal ID for sorting purpose in the Excel style document (nullable) (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>IsInternalStyle</td>
<td>Gets whether the style is system internal. Such styles are not meant to</td>
</tr>
</tbody>
</table>
### Name

Gets or sets the name of the style. If not defined, the automatically calculated hash will be used as name.

### StyleManagerReference

Sets the reference of the style manager.

---

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Append</td>
<td>Appends the specified style parts to the current one. The parts can be instances of sub-classes like Border or CellXf or a Style instance. Only the altered properties of the specified style or style part that differs from a new / untouched style instance will be appended. This enables method chaining.</td>
</tr>
<tr>
<td>CalculateHash</td>
<td>Override method to calculate the hash of this component (Overrides AbstractStyleCalculateHash.)</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Method to compare two objects for sorting purpose (Inherited from</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Copy</td>
<td>Method to copy the current object to a new one without casting (Overrides AbstractStyleCopy.)</td>
</tr>
<tr>
<td>CopyPropertiesT</td>
<td>Internal method to copy altered properties from a source object. The decision whether a property is copied is dependent on a untouched reference object (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>CopyStyle</td>
<td>Method to copy the current object to a new one with casting</td>
</tr>
<tr>
<td>Equals(Object)</td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Equals(AbstractStyle)</td>
<td>Method to compare two objects for sorting purpose (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
GetHashCode

Serves as a hash function for a particular type.
(Inherited from Object.)

GetType

Gets the Type of the current instance.
(Inherited from Object.)

MemberwiseClone

Creates a shallow copy of the current Object.
(Inherited from Object.)

ReorganizeStyle

Method to reorganize / synchronize the components of this style

ToString

Override toString method
(Overrides Object.ToString.)

See Also

Reference
NanoXLSX.Style Namespace
# Style Constructor

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Style</td>
<td>Default constructor</td>
</tr>
<tr>
<td>Style(String)</td>
<td>Constructor with parameters</td>
</tr>
<tr>
<td>Style(String, Int32, Boolean)</td>
<td>Constructor with parameters (internal use)</td>
</tr>
</tbody>
</table>

## See Also

- **Reference**
  - **Style Class**
  - **NanoXLSX.Style Namespace**

Copyright Raphael Stoeckli © 2018
Style Constructor

Default constructor

**Namespace:** NanoXLXSX.Style  
**Assembly:** NanoXLXSX (in NanoXLXSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Style()
```

### See Also

Reference
- Style Class  
- Style Overload  
- NanoXLXSX.Style Namespace

Copyright Raphael Stoeckli © 2018
Style Constructor (String)

Constructor with parameters

**Namespace:**  NanoXLSX.Style  
**Assembly:**  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Style(string name)
```

### Parameters

- **name**
  - Type: `System.String`
  - Name of the style

### See Also

- **Reference**
  - Style Class
  - Style Overload
  - NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
Style Constructor (String, Int32, Boolean)

Constructor with parameters (internal use)

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Style(
    string name,
    int forcedOrder,
    bool internalStyle
)
```

### Parameters

- **name**
  - Type: **SystemString**
  - Name of the style

- **forcedOrder**
  - Type: **SystemInt32**
  - Number of the style for sorting purpose. Style will be placed to this position (internal use only)

- **internalStyle**
  - Type: **SystemBoolean**
  - If true, the style is marked as internal

### See Also

Reference
Style Class
Style Overload
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
## Style Properties

The `Style` type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentBorder</td>
<td>Gets or sets the current Border object of the style</td>
</tr>
<tr>
<td>CurrentCellXf</td>
<td>Gets or sets the current CellXf object of the style</td>
</tr>
<tr>
<td>CurrentFill</td>
<td>Gets or sets the current Fill object of the style</td>
</tr>
<tr>
<td>CurrentFont</td>
<td>Gets or sets the current Font object of the style</td>
</tr>
<tr>
<td>CurrentNumberFormat</td>
<td>Gets or sets the current NumberFormat object of the style</td>
</tr>
<tr>
<td>Hash</td>
<td>Gets the unique hash of the object (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>InternalID</td>
<td>Gets or sets the internal ID for sorting purpose in the Excel style document (nullable)</td>
</tr>
<tr>
<td><strong>IsInternalStyle</strong></td>
<td>Gets whether the style is system internal. Such styles are not meant to be altered</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Gets or sets the name of the style. If not defined, the automatically calculated hash will be used as name</td>
</tr>
<tr>
<td><strong>StyleManagerReference</strong></td>
<td>Sets the reference of the style manager</td>
</tr>
</tbody>
</table>

**See Also**

Reference
- **Style Class**
- **NanoXLSX.Style Namespace**

Copyright Raphael Stoeckli © 2018
StyleCurrentBorderStyle Property

Gets or sets the current Border object of the style

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleBorder CurrentBorder { get; set; }
```

**Property Value**

Type: **StyleBorder**

### See Also

**Reference**

- **Style Class**
- **NanoXLSX.Style Namespace**

Copyright Raphael Stoeckli © 2018
StyleCurrentCellXf Property

Gets or sets the current CellXf object of the style

**Namespace:**  NanoXLSX.Style  
**Assembly:**  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleCellXf CurrentCellXf { get; set; }
```

**Property Value**

Type:  `StyleCellXf`

### See Also

**Reference**

- `Style Class`
- `NanoXLSX.Style Namespace`

---

Copyright Raphael Stoeckli © 2018
StyleCurrentFill Property

Gets or sets the current Fill object of the style

**Namespace:** NanoLSX.Style  
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

C#

```csharp
public StyleFill CurrentFill { get; set; }
```

### Property Value

Type: StyleFill

### See Also

Reference

Style Class  
NanoLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleCurrentFont Property

Gets or sets the current Font object of the style

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleFont CurrentFont { get; set; }
```

### Property Value

Type: `StyleFont`

### See Also

Reference
- **Style Class**
- **NanoXLSX.Style Namespace**

Copyright Raphael Stoeckli © 2018
StyleCurrentNumberFormat Property

Gets or sets the current NumberFormat object of the style

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleNumberFormat CurrentNumberFormat { get; }
```

Property Value

Type: **StyleNumberFormat**

### See Also

Reference  
**Style Class**  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleIsInternalStyle Property

Gets whether the style is system internal. Such styles are not meant to be altered

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public bool IsInternalStyle { get; }
```

Property Value
Type: Boolean

See Also

Reference
Style Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
StyleName Property

Gets or sets the name of the style. If not defined, the automatically calculated hash will be used as name

**Namespace:**  NanoXLSX.Style  
**Assembly:**  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```
public string Name { get; set; }
```

### Property Value

**Type:** String

## See Also

**Reference**

- **Style Class**  
- **NanoXLSX.Style Namespace**

Copyright Raphael Stoeckli © 2018
Property

Sets the reference of the style manager

**Namespace:**  NanoXLSX.Style  
**Assembly:**  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#  
public StyleManager StyleManagerReference { set;  
```

**Property Value**

Type: **StyleManager**

### See Also

Reference
Style Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
# Style Methods

The **Style** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Append</td>
<td>Appends the specified style parts to the current one. The parts can be instances of sub-classes like Border or CellXf or a Style instance. Only the altered properties of the specified style or style part that differs from a new / untouched style instance will be appended. This enables method chaining.</td>
</tr>
<tr>
<td>CalculateHash</td>
<td>Override method to calculate the hash of this component (Overrides AbstractStyleCalculateHash.)</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Method to compare two objects for sorting purpose (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Copy</td>
<td>Method to copy the current object to a new one without casting (Overrides AbstractStyleCopy.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CopyPropertiesT</td>
<td>Internal method to copy altered properties from a source object. The decision whether a property is copied is dependent on a untouched reference object. (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>CopyStyle</td>
<td>Method to copy the current object object to a new one with casting.</td>
</tr>
<tr>
<td>Equals(Object)</td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Equals(AbstractStyle)</td>
<td>Method to compare two objects for sorting purpose. (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>ReorganizeStyle</td>
<td>Method to reorganize / synchronize the components of this style</td>
</tr>
<tr>
<td>ToString</td>
<td>Override toString method</td>
</tr>
<tr>
<td></td>
<td>(Overrides ObjectToString.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference

Style Class

NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleAppend Method

Appends the specified style parts to the current one. The parts can be instances of sub-classes like Border or CellXf or a Style instance. Only the altered properties of the specified style or style part that differs from a new / untouched style instance will be appended. This enables method chaining.

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Style Append(
    AbstractStyle styleToAppend
)
```

### Parameters

**styleToAppend**  
Type: NanoXLSX.StyleAbstractStyle  
The style to append or a sub-class of Style

### Return Value

Type: Style  
Current style with appended style parts

### See Also

Reference  
Style Class  
NanoXLSX.Style Namespace
StyleCalculateHash Method

Override method to calculate the hash of this component

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public override sealed string CalculateHash()
```

### Return Value

**Type:** String  
Calculated hash as string

## See Also

**Reference**  
Style Class  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleCopy Method

Method to copy the current object to a new one without casting

**Namespace:** NanoXLSX.Style
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public override AbstractStyle Copy()
```

#### Return Value

**Type:** AbstractStyle
Copy of the current object without the internal ID

### See Also

**Reference**
- Style Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleCopyStyle Method

Method to copy the current object to a new one with casting

Namespace: NanoXLXSX.Style
Assembly: NanoXLXSX (in NanoXLXSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```
public Style CopyStyle()
```

Return Value
Type: Style
Copy of the current object without the internal ID

See Also

Reference
Style Class
NanoXLXSX.Style Namespace

Copyright Raphael Stoeckli © 2018
Style

ReorganizeStyle Method

Method to reorganize / synchronize the components of this style

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
private void ReorganizeStyle()
```

### See Also

Reference

- Style Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
**StyleToString Method**

Override toString method

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public override string ToString()
```

### Return Value

Type: String  
String of a class instance

### See Also

**Reference**  
Style Class  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
StyleBasicStyles Class

Factory class with the most important predefined styles

Inheritance Hierarchy

- SystemObject
- NanoXLSX.Style
- Style

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public static class BasicStyles
```

The StyleBasicStyles type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bold</td>
<td>Gets the bold style</td>
</tr>
<tr>
<td>BoldItalic</td>
<td>Gets the bold and italic style</td>
</tr>
<tr>
<td>BorderFrame</td>
<td>Gets the border frame style</td>
</tr>
<tr>
<td>BorderFrameHeader</td>
<td>Gets the border style for header cells</td>
</tr>
<tr>
<td>DateFormat</td>
<td>Gets the date format style</td>
</tr>
<tr>
<td>DottedFill_0_125</td>
<td>Gets the special pattern fill</td>
</tr>
</tbody>
</table>
style (for compatibility)

<table>
<thead>
<tr>
<th></th>
<th>DoubleUnderline</th>
<th>Gets the double underline style</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Italic</td>
<td>Gets the italic style</td>
</tr>
<tr>
<td></td>
<td>MergeCellStyle</td>
<td>Gets the style used when merging cells</td>
</tr>
<tr>
<td></td>
<td>RoundFormat</td>
<td>Gets the round format style</td>
</tr>
<tr>
<td></td>
<td>Strike</td>
<td>Gets the strike style</td>
</tr>
<tr>
<td></td>
<td>Underline</td>
<td>Gets the underline style</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ColorizedBackground</strong></td>
<td>Gets a style to colorize the background of a cell</td>
</tr>
<tr>
<td><strong>ColorizedText</strong></td>
<td>Gets a style to colorize the text of a cell</td>
</tr>
<tr>
<td><strong>Font</strong></td>
<td>Gets a style with a user defined font</td>
</tr>
<tr>
<td><strong>GetStyle</strong></td>
<td>Method to maintain the styles and to create singleton instances</td>
</tr>
</tbody>
</table>
### BasicStyles Properties

The `StyleBasicStyles` type exposes the following members.

#### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bold</td>
<td>Gets the bold style</td>
</tr>
<tr>
<td>BoldItalic</td>
<td>Gets the bold and italic style</td>
</tr>
<tr>
<td>BorderFrame</td>
<td>Gets the border frame style</td>
</tr>
<tr>
<td>BorderFrameHeader</td>
<td>Gets the border style for header cells</td>
</tr>
<tr>
<td>DateFormat</td>
<td>Gets the date format style</td>
</tr>
<tr>
<td>DottedFill_0_125</td>
<td>Gets the special pattern fill style (for compatibility)</td>
</tr>
<tr>
<td>DoubleUnderline</td>
<td>Gets the double underline style</td>
</tr>
<tr>
<td>Italic</td>
<td>Gets the italic style</td>
</tr>
<tr>
<td>MergeCellStyle</td>
<td>Gets the style used when merging cells</td>
</tr>
<tr>
<td>RoundFormat</td>
<td>Gets the round format style</td>
</tr>
<tr>
<td>Strike</td>
<td>Gets the strike style</td>
</tr>
<tr>
<td>Underline</td>
<td>Gets the underline style</td>
</tr>
</tbody>
</table>
See Also

Reference
StyleBasicStyles Class
NanoXLSX.Style Namespace
StyleBasicStylesBold Property

Gets the bold style

**Namespace:**  NanoXLSX.Style  
**Assembly:**  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Style Bold { get; }
```

### Property Value

Type:  **Style**

### See Also

**Reference**  
StyleBasicStyles Class  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBasicStylesBoldItalic Property

Gets the bold and italic style

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public static Style BoldItalic { get; }
```

Property Value

Type: Style

### See Also

Reference

- StyleBasicStyles Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
**StyleBasicStyles.BorderFrame Property**

Gets the border frame style

**Namespace:** NanoLSX.Style  
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public static Style BorderFrame { get; }
```

Property Value  
Type: **Style**

## See Also

*Reference*
- StyleBasicStyles Class  
- NanoLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBasicStylesBorderFrameHeader Property

Gets the border style for header cells

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

⚠ **Syntax**

```csharp
public static Style BorderFrameHeader { get; }
```

Property Value
Type: Style

⚠ **See Also**

Reference
- StyleBasicStyles Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBasicStylesDateFormat Property

Gets the date format style

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Style DateFormat { get; }
```

**Property Value**  
**Type:** Style

### See Also

**Reference**  
- StyleBasicStyles Class  
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBasicStylesDottedFill_0_125 Property

Gets the special pattern fill style (for compatibility)

**Namespace:** NanoXLSX.Style
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public static Style DottedFill_0_125 { get; }
```

Property Value
Type: Style

## See Also

Reference
**StyleBasicStyles Class**
**NanoXLSX.Style Namespace**

Copyright Raphael Stoeckli © 2018
StyleBasicStyles.DoubleUnderline Property

Gets the double underline style

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public static Style DoubleUnderline { get; }
```

Property Value
Type: Style

### See Also

Reference
- StyleBasicStyles Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBasicStylesItalic Property

Gets the italic style

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Style Italic { get; }
```

### Property Value

Type: Style

### See Also

**Reference**  
StyleBasicStyles Class  
NanoXLSX.Style Namespace
StyleBasicStylesMergeCellStyleStyle

Gets the style used when merging cells

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Style MergeCellStyle { get; }
```

**Property Value**

**Type:** Style

### See Also

Reference
- StyleBasicStyles Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBasicStyles.RoundFormat Property

Gets the round format style

**Namespace:** NanoXLSX.Style
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Style RoundFormat { get; }
```

**Property Value**
Type: **Style**

### See Also

Reference
- **StyleBasicStyles Class**
- **NanoXLSX.Style Namespace**
NanoXLSX Library Documentation
StyleBasicStylesStrike Property

Gets the strike style

**Namespace:**  NanoXLSX.Style  
**Assembly:**  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Style Strike { get; }
```

**Property Value**

Type: **Style**

### See Also

**Reference**

- **StyleBasicStyles Class**
- **NanoXLSX.Style Namespace**

Copyright Raphael Stoeckli © 2018
StyleBasicStylesUnderline Property

Gets the underline style

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Style Underline { get; }
```

Property Value

Type: Style

### See Also

Reference

- StyleBasicStyles Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
BasicStyles Methods

The `StyleBasicStyles` type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColorizedBackground</td>
<td>Gets a style to colorize the background of a cell</td>
</tr>
<tr>
<td>ColorizedText</td>
<td>Gets a style to colorize the text of a cell</td>
</tr>
<tr>
<td>Font</td>
<td>Gets a style with a user defined font</td>
</tr>
<tr>
<td>GetStyle</td>
<td>Method to maintain the styles and to create singleton instances</td>
</tr>
</tbody>
</table>

See Also

Reference

- `StyleBasicStyles Class`
- `NanoXLSX.Style Namespace`

Copyright Raphael Stoeckli © 2018
# StyleBasicStylesColorizedBackground Method

Gets a style to colorize the background of a cell

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public static Style ColorizedBackground(string rgb)
```

## Parameters

`rgb`  
Type: System.String  
RGB code in hex format (e.g. FF00AC). Alpha will be set to full opacity (FF)

## Return Value

Type: Style  
Style with background color definition

## See Also

**Reference**  
StyleBasicStyles Class  
NanoXLSX.Style Namespace
Style Basic Styles Colorized Text Method

Gets a style to colorize the text of a cell

**Namespace:** NanoLSX.Style

**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Style ColorizedText(
    string rgb
)
```

### Parameters

**rgb**

- **Type:** System.String
  - RGB code in hex format (e.g. FF00AC). Alpha will be set to full opacity (FF)

### Return Value

- **Type:** Style
  - Style with font color definition

### See Also

- **Reference**
  - Style Basic Styles Class
  - NanoLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
**StyleBasicStyles.Font Method**

Gets a style with a user defined font

**Namespace:** NanoXLX.Style  
**Assembly:** NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static Style Font(
    string fontName,
    int fontSize = 11,
    bool isBold = false,
    bool isItalic = false
)
```

### Parameters

- **fontName**
  - Type: `System.String`
  - Name of the font

- **fontSize (Optional)**
  - Type: `System.Int32`
  - Size of the font in points (optional; default 11)

- **isBold (Optional)**
  - Type: `System.Boolean`
  - If true, the font will be bold (optional; default false)

- **isItalic (Optional)**
  - Type: `System.Boolean`
  - If true, the font will be italic (optional; default false)

### Return Value

Type: `Style`
Style with font definition

Remarks

The font name as well as the availability of bold and italic on the font cannot be validated by NanoXLSX. The generated file may be corrupt or rendered with a fall-back font in case of an error.

See Also

Reference

StyleBasicStyles Class
NanoXLSX.Style Namespace
StyleBasicStylesGetStyle Method

Method to maintain the styles and to create singleton instances

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
private static Style GetStyle(
    StyleBasicStylesStyleEnum value
)
```

Parameters

`value`
Type: NanoXLSX.StyleBasicStylesStyleEnum
Enum value to maintain

Return Value
Type: Style
The style according to the passed enum value

See Also

Reference
StyleBasicStyles Class
NanoXLSX.Style Namespace

Copyright Raphaël Stoeckli © 2018
StyleBasicStylesStyleEnum Enumeration

Enum with style selection

**Namespace:** NanoXLSX.Style
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
private enum StyleEnum
```

**Members**

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bold</td>
<td>0</td>
<td>Format text bold</td>
</tr>
<tr>
<td>italic</td>
<td>1</td>
<td>Format text italic</td>
</tr>
<tr>
<td>boldItalic</td>
<td>2</td>
<td>Format text bold and italic</td>
</tr>
<tr>
<td>underline</td>
<td>3</td>
<td>Format text with an underline</td>
</tr>
<tr>
<td>doubleUnderline</td>
<td>4</td>
<td>Format text with a double underline</td>
</tr>
<tr>
<td>strike</td>
<td>5</td>
<td>Format text with a strike-through</td>
</tr>
<tr>
<td>dateFormat</td>
<td>6</td>
<td>Format number</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>roundFormat</code></td>
<td>7</td>
<td>Rounds number as an integer</td>
</tr>
<tr>
<td><code>borderFrame</code></td>
<td>8</td>
<td>Format cell with a thin border</td>
</tr>
<tr>
<td><code>borderFrameHeader</code></td>
<td>9</td>
<td>Format cell with a thin border and a thick bottom line as header cell</td>
</tr>
<tr>
<td><code>dottedFill_0_125</code></td>
<td>10</td>
<td>Special pattern fill style for compatibility purpose</td>
</tr>
<tr>
<td><code>mergeCellStyle</code></td>
<td>11</td>
<td>Style to apply on merged cells</td>
</tr>
</tbody>
</table>

⚠️ See Also

Reference

NanoXLX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBorder Class

Class representing a Border entry. The Border entry is used to define frames and cell borders.

Inheritance Hierarchy

- SystemObject
- NanoXLSX.StyleAbstractStyle
- NanoXLSX.StyleStyleBorder

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

C#

```csharp
public class Border : AbstractStyle
```

The StyleBorder type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleBorder</td>
<td>Default constructor</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BottomColor</td>
<td>Gets or sets the color code of the bottom border. The value is</td>
</tr>
</tbody>
</table>
expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BottomStyle</strong></td>
<td>Gets or sets the style of bottom cell border</td>
</tr>
<tr>
<td><strong>DiagonalColor</strong></td>
<td>Gets or sets the color code of the diagonal lines. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF</td>
</tr>
<tr>
<td><strong>DiagonalDown</strong></td>
<td>Gets or sets whether the downwards diagonal line is used. If true, the line is used</td>
</tr>
<tr>
<td><strong>DiagonalStyle</strong></td>
<td>Gets or sets the style of the diagonal lines</td>
</tr>
<tr>
<td><strong>DiagonalUp</strong></td>
<td>Gets or sets whether the upwards diagonal line is used. If true, the line is used</td>
</tr>
<tr>
<td><strong>Hash</strong></td>
<td>Gets the unique hash of the object (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td><strong>InternalID</strong></td>
<td>Gets or sets the internal ID for sorting purpose in the Excel style document (nullable) (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td><strong>LeftColor</strong></td>
<td>Gets or sets the color code of the left border. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LeftStyle</td>
<td>Gets or sets the style of left cell border</td>
</tr>
<tr>
<td>RightColor</td>
<td>Gets or sets the color code of the right border. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF</td>
</tr>
<tr>
<td>RightStyle</td>
<td>Gets or sets the style of right cell border</td>
</tr>
<tr>
<td>TopColor</td>
<td>Gets or sets the color code of the top border. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF</td>
</tr>
<tr>
<td>TopStyle</td>
<td>Gets or sets the style of top cell border</td>
</tr>
<tr>
<td>CalculateHash</td>
<td>Override method to calculate the hash of this component (internal method) (Overrides AbstractStyle.CalculateHash.)</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Method to compare two objects for sorting purpose (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Copy</td>
<td>Method to copy the current object to a new one without casting (Overrides AbstractStyleCopy.)</td>
</tr>
<tr>
<td>CopyBorder</td>
<td>Method to copy the current object to a new one with casting</td>
</tr>
<tr>
<td>CopyPropertiesT</td>
<td>Internal method to copy altered properties from a source object. The decision whether a property is copied is dependent on a untouched reference object (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Equals(Object)</td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Equals(AbstractStyle)</td>
<td>Method to compare two objects for sorting purpose (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for</td>
</tr>
</tbody>
</table>
a particular type.  
(Inherited from Object.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetStyleName</td>
<td>Gets the border style name from the enum</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>IsEmpty</td>
<td>Method to determine whether the object has no values but the default values (means: is empty and must not be processed)</td>
</tr>
</tbody>
</table>
| MemberwiseClone | Creates a shallow copy of the current Object.  
(Inherited from Object.) |
| ToString    | Override toString method  
(Overrides ObjectToString.) |

See Also

Reference
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBorder Constructor

Default constructor

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public border()
```

### See Also

- Reference
  - **StyleBorder Class**
  - **NanoXLSX.Style Namespace**

Copyright Raphael Stoeckli © 2018
Border Properties

The `StyleBorder` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Image] BottomColor</td>
<td>Gets or sets the color code of the bottom border. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF</td>
</tr>
<tr>
<td>![Image] BottomStyle</td>
<td>Gets or sets the style of bottom cell border</td>
</tr>
<tr>
<td>![Image] DiagonalColor</td>
<td>Gets or sets the color code of the diagonal lines. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF</td>
</tr>
<tr>
<td>![Image] DiagonalDown</td>
<td>Gets or sets whether the downwards diagonal line is used. If true, the line is used</td>
</tr>
<tr>
<td>![Image] DiagonalStyle</td>
<td>Gets or sets the style of the diagonal lines</td>
</tr>
<tr>
<td>![Image] DiagonalUp</td>
<td>Gets or sets whether the upwards diagonal line is used. If true, the line is used</td>
</tr>
<tr>
<td>![Image] Hash</td>
<td>Gets the unique hash of the</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>InternalID</strong></td>
<td>Gets or sets the internal ID for sorting purpose in the Excel style document (nullable) (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td><strong>LeftColor</strong></td>
<td>Gets or sets the color code of the left border. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF</td>
</tr>
<tr>
<td><strong>LeftStyle</strong></td>
<td>Gets or sets the style of left cell border</td>
</tr>
<tr>
<td><strong>RightColor</strong></td>
<td>Gets or sets the color code of the right border. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF</td>
</tr>
<tr>
<td><strong>RightStyle</strong></td>
<td>Gets or sets the style of right cell border</td>
</tr>
<tr>
<td><strong>TopColor</strong></td>
<td>Gets or sets the color code of the top border. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF</td>
</tr>
<tr>
<td><strong>TopStyle</strong></td>
<td>Gets or sets the style of top cell border</td>
</tr>
</tbody>
</table>

**Top**
See Also

Reference
StyleBorderStyle
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBorderBottomColor Property

Gets or sets the color code of the bottom border. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF.

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

public string BottomColor { get; set; }

Property Value
Type: String

See Also

Reference
StyleBorder Class
NanoXLSX.Style Namespace
StyleBorderStyle Property

Gets or sets the style of bottom cell border

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleBorderStyleStyleValue BottomStyle { get; set; }
```

Property Value  
Type: **StyleBorderStyleStyleValue**

### See Also

Reference  
**StyleBorderStyle Class**  
**NanoXLSX.Style Namespace**

Copyright Raphael Stoeckli © 2018
StyleBorderStyleDiagonalColor Property

Gets or sets the color code of the diagonal lines. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public string DiagonalColor { get; set; }
```

Property Value
Type: String

See Also

Reference
StyleBorderStyle Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBorderDiagonalDown Property

Gets or sets whether the downwards diagonal line is used. If true, the line is used.

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public bool DiagonalDown { get; set; }
```

Property Value

Type: Boolean

## See Also

Reference
- StyleBorder Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBorderStyle DiagonalStyle Property

Gets or sets the style of the diagonal lines

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleBorderStyle StyleValue DiagonalStyle { get; }
```

#### Property Value

Type: StyleBorderStyle

### See Also

Reference

- StyleBorderStyle Class
- NanoXLSX.Style Namespace
StyleBorderDiagonalUp Property

Gets or sets whether the upwards diagonal line is used. If true, the line is used.

**Namespace:** NanoLSX.Style  
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public bool DiagonalUp { get; set; }
```

### Property Value

Type: Boolean

## See Also

Reference  
- StyleBorder Class  
- NanoLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBorderStyle Property

Gets or sets the color code of the left border. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF

Namespace:  NanoXLSX.Style
Assembly:  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public string LeftColor { get; set; }
```

Property Value
Type:  String

See Also

Reference
StyleBorderStyle Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBorderLeftStyle Property

Gets or sets the style of left cell border

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleBorderStyleValue LeftStyle { get; set; }
```

**Property Value**  
Type: `StyleBorderStyleValue`  

### See Also

- Reference  
  - **StyleBorder Class**  
  - **NanoXLSX.Style Namespace**

Copyright Raphael Stoeckli © 2018
StyleBorderStyle RightColor Property

Gets or sets the color code of the right border. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public string RightColor { get; set; }
```

Property Value
Type: String

See Also

Reference
StyleBorderStyle Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBorderRightStyle Property

Gets or sets the style of right cell border

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleBorderStyleStyleValue RightStyle { get; set; }
```

### Property Value

Type: `StyleBorderStyleStyleValue`

### See Also

Reference  
**StyleBorder Class**  
**NanoXLSX.Style Namespace**

---

Copyright Raphaël Stoeckli © 2018
StyleBorderStyleTopColor Property

Gets or sets the color code of the top border. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```
public string TopColor { get; set; }
```

Property Value
Type: String

See Also

Reference
StyleBorderStyle Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBorderStyleTopStyle Property

Gets or sets the style of top cell border

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleBorderStyleStyleValue TopStyle { get; set; }
```

**Property Value**

Type: StyleBorderStyleStyleValue

### See Also

**Reference**

StyleBorderStyle Class  
NanoXLSX.Style Namespace

---

Copyright Raphael Stoeckli © 2018
Border Methods

The `StyleBorder` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalculateHash</td>
<td>Override method to calculate the hash of this component (internal method)</td>
</tr>
<tr>
<td></td>
<td>(Overrides <code>AbstractStyleCalculateHash</code>.)</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Method to compare two objects for sorting purpose (Inherited from <code>AbstractStyle</code>.</td>
</tr>
<tr>
<td>Copy</td>
<td>Method to copy the current object to a new one without casting</td>
</tr>
<tr>
<td></td>
<td>(Overrides <code>AbstractStyleCopy</code>.)</td>
</tr>
<tr>
<td>CopyBorder</td>
<td>Method to copy the current object to a new one with casting</td>
</tr>
<tr>
<td>CopyPropertiesT</td>
<td>Internal method to copy altered properties from a source object. The decision whether a property is copied is dependent on a untouched reference object</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Equals(Object)</td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Equals(AbstractStyle)</td>
<td>Method to compare two objects for sorting purpose (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetStyleName</td>
<td>Gets the border style name from the enum</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>IsEmpty</td>
<td>Method to determine whether the object has no values but the default values (means: is empty and must not be processed)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of</td>
</tr>
</tbody>
</table>
the current **Object**.
(Inherited from **Object**.)

| **ToString** | Override toString method
(Overrides **ObjectToString**.) |

**See Also**

**Reference**
- **StyleBorder Class**
- **NanoXLSX.Style Namespace**
StyleBorderCalculateHash Method

Override method to calculate the hash of this component (internal method)

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public override string CalculateHash()
```

### Return Value

Type: **String**
Calculated hash as string

### See Also

Reference  
StyleBorder Class  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBorderCopy Method

Method to copy the current object to a new one without casting

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public override AbstractStyle Copy()
```

Return Value
Type: AbstractStyle
Copy of the current object without the internal ID

## See Also

Reference
StyleBorder Class
NanoXLSX.Style Namespace
StyleBorderCopyBorder Method

Method to copy the current object to a new one with casting

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleBorder CopyBorder()
```

### Return Value

Type: StyleBorder  
Copy of the current object without the internal ID

### See Also

**Reference**  
StyleBorder Class  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBorderStyleGetName Method

Gets the border style name from the enum

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public static string GetStyleName(
    StyleBorderStyleValue style
)
```

**Parameters**

*style*  
Type: NanoXLSX.StyleStyleBorderStyleValue  
Enum to process

**Return Value**

Type: String  
The valid value of the border style as String

**See Also**

Reference  
StyleBorderStyle Class  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBorderStyleIsEmpty Method

Method to determine whether the object has no values but the default values (means: is empty and must not be processed)

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public bool IsEmpty()
```

Return Value  
Type: **Boolean**  
True if empty, otherwise false

### See Also

**Reference**  
- StyleBorderStyle Class  
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleBorderStyleToString Method

Override toString method

**Namespace:** NanoXLSX.Style

**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public override string ToString()
```

**Return Value**

**Type:** String

String of a class

### See Also

**Reference**

StyleBorderStyle Class

NanoXLSX.Style Namespace
StyleBorderStyleValue Enumeration

Enum for the border style

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public enum StyleValue
```

**Members**

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>0</td>
<td>no border</td>
</tr>
<tr>
<td>hair</td>
<td>1</td>
<td>hair border</td>
</tr>
<tr>
<td>dotted</td>
<td>2</td>
<td>dotted border</td>
</tr>
<tr>
<td>dashDotDot</td>
<td>3</td>
<td>dashed border with double-dots</td>
</tr>
<tr>
<td>dashDot</td>
<td>4</td>
<td>dash-dotted border</td>
</tr>
<tr>
<td>dashed</td>
<td>5</td>
<td>dashed border</td>
</tr>
<tr>
<td>thin</td>
<td>6</td>
<td>thin border</td>
</tr>
<tr>
<td>mediumDashDotDot</td>
<td>7</td>
<td>medium-dashed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>-----</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>slantDashDot</td>
<td>8</td>
<td>slant dash-dotted border</td>
</tr>
<tr>
<td>mediumDashDot</td>
<td>9</td>
<td>medium dash-dotted border</td>
</tr>
<tr>
<td>mediumDashed</td>
<td>10</td>
<td>medium dashed border</td>
</tr>
<tr>
<td>medium</td>
<td>11</td>
<td>medium border</td>
</tr>
<tr>
<td>thick</td>
<td>12</td>
<td>thick border</td>
</tr>
<tr>
<td>s_double</td>
<td>13</td>
<td>double border</td>
</tr>
</tbody>
</table>

**See Also**

*Reference*
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
**StyleCellXf Class**

Class representing an XF entry. The XF entry is used to make reference to other style instances like Border or Fill and for the positioning of the cell content.

**Inheritance Hierarchy**

```
SystemObject NanoXLXSX.StyleAbstractStyle NanoXLXSX.StyleAbstractStyle
NanoXLXSX.StyleStyleCellXf
```

**Namespace:** NanoXLXSX.Style  
**Assembly:** NanoXLXSX (in NanoXLXSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```
public class CellXf : AbstractStyle
```

The **StyleCellXf** type exposes the following members.

**Constructors**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleCellXf</td>
<td>Default constructor</td>
</tr>
</tbody>
</table>

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment</td>
<td>Gets or sets the text break</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ForceApplyAlignment</strong></td>
<td>Gets or sets whether the applyAlignment property (used to merge cells) will be defined in the XF entry of the style. If true, applyAlignment will be defined.</td>
</tr>
<tr>
<td><strong>Hash</strong></td>
<td>Gets the unique hash of the object (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td><strong>Hidden</strong></td>
<td>Gets or sets whether the hidden property (used for protection or hiding of cells) will be defined in the XF entry of the style. If true, hidden will be defined.</td>
</tr>
<tr>
<td><strong>HorizontalAlign</strong></td>
<td>Gets or sets the horizontal alignment of the style.</td>
</tr>
<tr>
<td><strong>InternalID</strong></td>
<td>Gets or sets the internal ID for sorting purpose in the Excel style document (nullable) (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td><strong>Locked</strong></td>
<td>Gets or sets whether the locked property (used for locking / protection of cells or worksheets) will be defined in the XF entry of the style. If true, locked will</td>
</tr>
</tbody>
</table>
TextDirection

Gets or sets the direction of the text within the cell

TextRotation

Gets or sets the text rotation rotation in degrees (from +90 to -90)

VerticalAlign

Gets or sets the vertical alignment of the style

Top

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalculateHash</td>
<td>Override method to calculate the hash of this component (internal method) (Overrides AbstractStyleCalculateHash.)</td>
</tr>
<tr>
<td>CalculateInternalRotation</td>
<td>Method to calculate the internal text rotation. The text direction and rotation are handled internally by the text rotation value</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Method to compare two objects for sorting purpose (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Copy</td>
<td>Method to copy the current object to a new one without casting</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>CopyCellXf</code></td>
<td>Method to copy the current object to a new one with casting</td>
</tr>
<tr>
<td><code>CopyPropertiesT</code></td>
<td>Internal method to copy altered properties from a source object. The decision whether a property is copied is dependent on a untouched reference object (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td><code>Equals(Object)</code></td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td><code>Equals(AbstractStyle)</code></td>
<td>Method to compare two objects for sorting purpose (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td><code>Finalize</code></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><code>GetHashCode</code></td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td><code>GetType</code></td>
<td>Gets the Type of the current</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Override toString method (Overrides Object.ToString.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleCellXf Constructor

Default constructor

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public CellXf()
```

### See Also

**Reference**  
*StyleCellXf Class*  
*NanoXLSX.Style Namespace*

Copyright Raphael Stoeckli © 2018
CellXf Properties

The `StyleCellXf` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment</td>
<td>Gets or sets the text break options of the style</td>
</tr>
<tr>
<td>ForceApplyAlignment</td>
<td>Gets or sets whether the applyAlignment property (used to merge cells) will be defined in the XF entry of the style. If true, applyAlignment will be defined</td>
</tr>
<tr>
<td>Hash</td>
<td>Gets the unique hash of the object (Inherited from <code>AbstractStyle</code>.)</td>
</tr>
<tr>
<td>Hidden</td>
<td>Gets or sets whether the hidden property (used for protection or hiding of cells) will be defined in the XF entry of the style. If true, hidden will be defined</td>
</tr>
<tr>
<td>HorizontalAlign</td>
<td>Gets or sets the horizontal alignment of the style</td>
</tr>
<tr>
<td>InternalID</td>
<td>Gets or sets the internal ID</td>
</tr>
</tbody>
</table>
for sorting purpose in the Excel style document (nullable) (Inherited from AbstractStyle.)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locked</td>
<td>Gets or sets whether the locked property (used for locking / protection of cells or worksheets) will be defined in the XF entry of the style. If true, locked will be defined.</td>
</tr>
<tr>
<td>TextDirection</td>
<td>Gets or sets the direction of the text within the cell</td>
</tr>
<tr>
<td>TextRotation</td>
<td>Gets or sets the text rotation in degrees (from +90 to -90)</td>
</tr>
<tr>
<td>VerticalAlign</td>
<td>Gets or sets the vertical alignment of the style</td>
</tr>
</tbody>
</table>

See Also

Reference
- StyleCellXf Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleCellXfAlignment Property

Gets or sets the text break options of the style

**Namespace:** NanoLSX.Style  
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleCellXfTextBreakValue Alignment { get; }
```

**Property Value**  
Type: StyleCellXfTextBreakValue

### See Also

Reference  
StyleCellXf Class  
NanoLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleCellXfForceApplyAlignment Property

Gets or sets whether the applyAlignment property (used to merge cells) will be defined in the XF entry of the style. If true, applyAlignment will be defined

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

C# copy

```csharp
public bool ForceApplyAlignment { get; set; }
```

Property Value
Type: Boolean

See Also

Reference
StyleCellXf Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleCellXfHidden Property

Gets or sets whether the hidden property (used for protection or hiding of cells) will be defined in the XF entry of the style. If true, hidden will be defined

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```
public bool Hidden { get; set; }
```

Property Value  
Type: **Boolean**

## See Also

Reference  
StyleCellXf Class  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleCellXfHorizontalAlign

Property

Gets or sets the horizontal alignment of the style

Namespace: NanoLSX.Style
Assembly: NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c# public StyleCellXfHorizontalAlignValue Horizontal```

Property Value

Type: StyleCellXfHorizontalAlignValue

See Also

Reference

StyleCellXf Class
NanoLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleCellXfLocked Property

Gets or sets whether the locked property (used for locking / protection of cells or worksheets) will be defined in the XF entry of the style. If true, locked will be defined.

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public bool Locked { get; set; }
```

Property Value  
Type: Boolean

### See Also

**Reference**  
StyleCellXf Class  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleCellXfTextDirection Property

Gets or sets the direction of the text within the cell

**Namespace:** NanoXLSX.Style

**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public StyleCellXfTextDirectionValue TextDirection
```

### Property Value

Type: StyleCellXfTextDirectionValue

### See Also

Reference
- StyleCellXf Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleCellXf.TextRotation Property

Gets or sets the text rotation in degrees (from +90 to -90)

**Namespace:** NanoXL SX.Style  
**Assembly:** NanoXL SX (in NanoXL SX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public int TextRotation { get; set; }
```

**Property Value**

Type: **Int32**

**See Also**

**Reference**

- StyleCellXf Class
- NanoXL SX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleCellXfVerticalAlign Property

Gets or sets the vertical alignment of the style

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public StyleCellXfVerticalAlignValue VerticalAlign
```

### Property Value

Type: `StyleCellXfVerticalAlignValue`

### See Also

**Reference**
- `StyleCellXf Class`
- `NanoXLSX.Style Namespace`

Copyright Raphael Stoeckli © 2018
# CellXf Methods

The `StyleCellXf` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalculateHash</td>
<td>Override method to calculate the hash of this component (internal method) (Overrides <code>AbstractStyleCalculateHash</code>.)</td>
</tr>
<tr>
<td>CalculateInternalRotation</td>
<td>Method to calculate the internal text rotation. The text direction and rotation are handled internally by the text rotation value</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Method to compare two objects for sorting purpose (Inherited from <code>AbstractStyle</code>.)</td>
</tr>
<tr>
<td>Copy</td>
<td>Method to copy the current object to a new one without casting (Overrides <code>AbstractStyleCopy</code>.)</td>
</tr>
<tr>
<td>CopyCellXf</td>
<td>Method to copy the current object to a new one with casting</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>CopyProperties&lt;T&gt;</strong></td>
<td>Internal method to copy altered properties from a source object. The decision whether a property is copied is dependent on a untouched reference object (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td><strong>Equals(Object)</strong></td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Equals(AbstractStyle)</strong></td>
<td>Method to compare two objects for sorting purpose (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
ToString

Override toString method
(Overrides ObjectToString.)

See Also

Reference
StyleCellXf Class
NanoXLSX.Style Namespace
StyleCellXfCalculateHash Method

Override method to calculate the hash of this component (internal method)

**Namespace:** NanoXLSX.Style

**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public override string CalculateHash()
```

**Return Value**

- **Type:** String
- Calculated hash as string

### See Also

- **Reference**
  - StyleCellXf Class
  - NanoXLSX.Style Namespace
StyleCellXfCalculateInternalRotation Method

Method to calculate the internal text rotation. The text direction and rotation are handled internally by the text rotation value.

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public int CalculateInternalRotation()
```

Return Value

Type: Int32
Returns the valid rotation in degrees for internal uses (LowLevel)

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormatException</td>
<td>Throws a FormatException if the rotation angle (-90 to 90) is out of range</td>
</tr>
</tbody>
</table>

See Also

Reference

StyleCellXf Class
NanoXLSX.Style Namespace
StyleCellXfCopy Method

Method to copy the current object to a new one without casting

Namespace: NanoLSX.Style
Assembly: NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public override AbstractStyle Copy()
```

Return Value
Type: AbstractStyle
Copy of the current object without the internal ID

See Also

Reference
StyleCellXf Class
NanoLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
StyleCellXfCopyCellXf Method

Method to copy the current object to a new one with casting

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleCellXf CopyCellXf();
```

**Return Value**

Type: **StyleCellXf**  
Copy of the current object without the internal ID

### See Also

**Reference**

- **StyleCellXf Class**
- **NanoXLSX.Style Namespace**
StyleCellXfToString Method

Override toString method

**Namespace:** NanoXLSX.Style
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public override string ToString()
```

Return Value
- Type: **String**
- String of a class instance

See Also

Reference
- **StyleCellXf Class**
- NanoXLSX.Style Namespace
NanoXLSX Library Documentation
StyleCellXfHorizontalAlignmentValue Enumeration

Enum for the horizontal alignment of a cell

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public enum HorizontalAlignValue
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>left</td>
<td>0</td>
<td>Content will be aligned left</td>
</tr>
<tr>
<td>center</td>
<td>1</td>
<td>Content will be aligned in the center</td>
</tr>
<tr>
<td>right</td>
<td>2</td>
<td>Content will be aligned right</td>
</tr>
<tr>
<td>fill</td>
<td>3</td>
<td>Content will fill up the cell</td>
</tr>
<tr>
<td>justify</td>
<td>4</td>
<td>justify alignment</td>
</tr>
<tr>
<td>general</td>
<td>5</td>
<td>General alignment</td>
</tr>
<tr>
<td>centerContinuous</td>
<td>6</td>
<td>Center continuous</td>
</tr>
<tr>
<td>alignment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>distributed</td>
<td>7</td>
<td>Distributed alignment</td>
</tr>
<tr>
<td>none</td>
<td>8</td>
<td>No alignment. The alignment will not be used in a style</td>
</tr>
</tbody>
</table>

See Also

Reference

NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleCellXfTextBreakValue Enumeration

Enum for text break options

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

C#

```csharp
public enum TextBreakValue
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wrapText</td>
<td>0</td>
<td>Word wrap is active</td>
</tr>
<tr>
<td>shrinkToFit</td>
<td>1</td>
<td>Text will be resized to fit the cell</td>
</tr>
<tr>
<td>none</td>
<td>2</td>
<td>Text will overflow in cell</td>
</tr>
</tbody>
</table>

### See Also

Reference  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleCellXfTextDirectionValue Enumeration

Enum for the general text alignment direction

**Namespace:**  NanoXLSX.Style  
**Assembly:**  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

⚠️ Syntax

```csharp
public enum TextDirectionValue
```

⚠️ Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>horizontal</td>
<td>0</td>
<td>Text direction is horizontal (default)</td>
</tr>
<tr>
<td>vertical</td>
<td>1</td>
<td>Text direction is vertical</td>
</tr>
</tbody>
</table>

⚠️ See Also

Reference  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
**StyleCellXfVerticalAlignValue Enumeration**

Enum for the vertical alignment of a cell

**Namespace:** NanoLSX.Style

**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public enum VerticalAlignValue
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bottom</td>
<td>0</td>
<td>Content will be aligned on the bottom (default)</td>
</tr>
<tr>
<td>top</td>
<td>1</td>
<td>Content will be aligned on the top</td>
</tr>
<tr>
<td>center</td>
<td>2</td>
<td>Content will be aligned in the center</td>
</tr>
<tr>
<td>justify</td>
<td>3</td>
<td>justify alignment</td>
</tr>
<tr>
<td>distributed</td>
<td>4</td>
<td>Distributed alignment</td>
</tr>
<tr>
<td>none</td>
<td>5</td>
<td>No alignment. The alignment will not be used in a style</td>
</tr>
</tbody>
</table>
See Also

Reference
NanoXLSX.Style Namespace
StyleFill Class

Class representing a Fill (background) entry. The Fill entry is used to define background colors and fill patterns.

Inheritance Hierarchy

```
SystemObject  NanoXLSX.StyleAbstractStyle
           NanoXLSX.StyleStyleFill
```

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```
public class Fill : AbstractStyle
```

The StyleFill type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleFill</td>
<td>Default constructor</td>
</tr>
<tr>
<td>StyleFill(String, StyleFillFillType)</td>
<td>Constructor with color value and fill type</td>
</tr>
<tr>
<td>StyleFill(String, String)</td>
<td>Constructor with foreground and background color</td>
</tr>
</tbody>
</table>
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BackgroundColor</strong></td>
<td>Gets or sets the background color of the fill. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF</td>
</tr>
<tr>
<td><strong>ForegroundColor</strong></td>
<td>Gets or sets the foreground color of the fill. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF</td>
</tr>
<tr>
<td><strong>Hash</strong></td>
<td>Gets the unique hash of the object (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td><strong>IndexedColor</strong></td>
<td>Gets or sets the indexed color (Default is 64)</td>
</tr>
<tr>
<td><strong>InternalID</strong></td>
<td>Gets or sets the internal ID for sorting purpose in the Excel style document (nullable) (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td><strong>PatternFill</strong></td>
<td>Gets or sets the pattern type of the fill (Default is none)</td>
</tr>
</tbody>
</table>

## Methods
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalculateHash</td>
<td>Override method to calculate the hash of this component (Overrides AbstractStyle.CalculateHash.)</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Method to compare two objects for sorting purpose (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Copy</td>
<td>Method to copy the current object to a new one without casting (Overrides AbstractStyle.Copy.)</td>
</tr>
<tr>
<td>CopyFill</td>
<td>Method to copy the current object to a new one with casting</td>
</tr>
<tr>
<td>CopyPropertiesT</td>
<td>Internal method to copy altered properties from a source object. The decision whether a property is copied is dependent on an untouched reference object (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Equals(Object)</td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Equals(AbstractStyle)</td>
<td>Method to compare two objects for sorting purpose (Inherited from AbstractStyle.)</td>
</tr>
</tbody>
</table>
Finalize

Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.
(Inherited from \texttt{Object}.)

GetHashCode

Serves as a hash function for a particular type.
(Inherited from \texttt{Object}.)

GetPatternName

Gets the pattern name from the enum

GetType

Gets the \texttt{Type} of the current instance.
(Inherited from \texttt{Object}.)

MemberwiseClone

Creates a shallow copy of the current \texttt{Object}.
(Inherited from \texttt{Object}.)

SetColor

Sets the color and the depending fill type

ToString

Override toString method
(Overrides \texttt{ObjectToString}.)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{DEFAULTCOLOR}</td>
<td>Default Color (foreground or background) as constant</td>
</tr>
</tbody>
</table>
See Also

Reference
NanoXLSX.Style Namespace
Fill Constructor

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StyleFill</code></td>
<td>Default constructor</td>
</tr>
<tr>
<td><code>StyleFill(String, StyleFillFillType)</code></td>
<td>Constructor with color value and fill type</td>
</tr>
<tr>
<td><code>StyleFill(String, String)</code></td>
<td>Constructor with foreground and background color</td>
</tr>
</tbody>
</table>

**See Also**

Reference

- [StyleFill Class](#)
- [NanoXLSX.Style Namespace](#)
StyleFill Constructor

Default constructor

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public Fill()
```

### See Also

**Reference**
- StyleFill Class
- StyleFill Overload
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFill Constructor (String, StyleFillFillFillType)

Constructor with color value and fill type

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
public Fill(
    string value,
    StyleFillFillFillType filltype
)
```

Parameters

'value'
Type: System.String
Color value

'filltype'
Type: NanoXLSX.StyleStyleFillFillFillType
Fill type (fill or pattern)

See Also

Reference
StyleFill Class
StyleFill Overload
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFill Constructor (String, String)

Constructor with foreground and background color

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public Fill(
    string foreground,
    string background
)
```

### Parameters

*foreground*

- Type: `System.String`
- Foreground color of the fill

*background*

- Type: `System.String`
- Background color of the fill

## See Also

Reference

- [StyleFill Class](#)
- [StyleFill Overload](#)
- [NanoXLSX.Style Namespace](#)
# Fill Properties

The **StyleFill** type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BackgroundColor</strong></td>
<td>Gets or sets the background color of the fill. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF</td>
</tr>
<tr>
<td><strong>ForeColor</strong></td>
<td>Gets or sets the foreground color of the fill. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF</td>
</tr>
<tr>
<td><strong>Hash</strong></td>
<td>Gets the unique hash of the object (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td><strong>IndexedColor</strong></td>
<td>Gets or sets the indexed color (Default is 64)</td>
</tr>
<tr>
<td><strong>InternalID</strong></td>
<td>Gets or sets the internal ID for sorting purpose in the Excel style document (nullable) (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td><strong>PatternFill</strong></td>
<td>Gets or sets the pattern type of the fill (Default is none)</td>
</tr>
</tbody>
</table>
See Also

Reference
- StyleFill Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFillBackgroundColor

Property

Gets or sets the background color of the fill. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

C#

```csharp
public string BackgroundColor { get; set; }
```

Property Value

Type: String

See Also

Reference

StyleFill Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFillForegroundColor Property

Gets or sets the foreground color of the fill. The value is expressed as hex string with the format AARRGGBB. AA (Alpha) is usually FF

**Namespace:**  NanoXLSX.Style  
**Assembly:**  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public string ForegroundColor { get; set; }
```

**Property Value**
- **Type:** String

### See Also

- **Reference**
  - StyleFill Class
  - NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFillIndexedColor Property

Gets or sets the indexed color (Default is 64)

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public int IndexedColor { get; set; }
```

### Property Value

Type: Int32

### See Also

- Reference
  - StyleFill Class
  - NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFillPatternFill Property

Gets or sets the pattern type of the fill (Default is none)

**Namespace:** NanoXLSX.Style

**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleFillPatternValue PatternFill { get; set; }
```

### Property Value

Type: `StyleFillPatternValue`

### See Also

**Reference**
- StyleFill Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
Fill Methods

The **StyleFill** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalculateHash</td>
<td>Override method to calculate the hash of this component (Overrides AbstractStyleCalculateHash.)</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Method to compare two objects for sorting purpose (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Copy</td>
<td>Method to copy the current object to a new one without casting (Overrides AbstractStyleCopy.)</td>
</tr>
<tr>
<td>CopyFill</td>
<td>Method to copy the current object to a new one with casting</td>
</tr>
<tr>
<td>CopyPropertiesT</td>
<td>Internal method to copy altered properties from a source object. The decision whether a property is copied is dependent on a untouched reference object (Inherited from</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Equals(Object)</td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Equals(AbstractStyle)</td>
<td>Method to compare two objects for sorting purpose (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetPatternName</td>
<td>Gets the pattern name from the enum</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>SetColor</td>
<td>Sets the color and the depending fill type</td>
</tr>
<tr>
<td>ToString</td>
<td>Override toString method</td>
</tr>
</tbody>
</table>
See Also

Reference
StyleFill Class
NanoXLSX.Style Namespace
StyleFillCalculateHash Method

Override method to calculate the hash of this component

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public override string CalculateHash()
```

Return Value
Type: String
Calculated hash as string

See Also

Reference
StyleFill Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFillCopy Method

Method to copy the current object to a new one without casting

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public override AbstractStyle Copy()
```

**Return Value**

Type: AbstractStyle  
Copy of the current object without the internal ID

### See Also

**Reference**
- StyleFill Class  
- NanoXLSX.Style Namespace
StyleFillCopyFill Method

Method to copy the current object to a new one with casting

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleFill CopyFill()
```

### Return Value

Type: **StyleFill**  
Copy of the current object without the internal ID

### See Also

**Reference**
- **StyleFill Class**
- **NanoXLSX.Style Namespace**
StyleFillGetPatternName Method

Gets the pattern name from the enum

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public static string GetPatternName(  
    StyleFillPatternValue pattern
)
```

### Parameters

- **pattern**
  - Type: NanoXLSX.StyleStyleFillPatternValue
    Enum to process

### Return Value

- Type: String
  - The valid value of the pattern as String

### See Also

- Reference
  - StyleFill Class
  - NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
# StyleFillSetColor Method

Sets the color and the depending fill type

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public void SetColor(
    string value,
    StyleFillFillType filltype
)
```

### Parameters

- **value**
  - Type: System.String  
  - color value

- **filltype**
  - Type: NanoXLSX.StyleStyleFillFillType  
  - fill type (fill or pattern)

## See Also

- **Reference**
  - StyleFill Class  
  - NanoXLSX.Style Namespace

---

Copyright Raphael Stoeckli © 2018
StyleFillToString Method

Override toString method

**Namespace:** NanoXLSX.Style
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public override string ToString()
```

### Return Value

Type: **String**
String of a class

### See Also

**Reference**
- StyleFill Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
Fill Fields

The `StyleFill` type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DEFAULTCOLOR</code></td>
<td>Default Color (foreground or background) as constant</td>
</tr>
</tbody>
</table>

### See Also

Reference
- `StyleFill Class`
- `NanoXLSX.Style Namespace`

Copyright Raphael Stoeckli © 2018
StyleFill\n\nDEFAULTCOLOR Field

Default Color (foreground or background) as constant

**Namespace:** NanoLSX.Style  
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
public const string DEFAULTCOLOR = "FF000000"
```

Field Value  
Type: String

See Also

Reference  
StyleFill Class  
NanoLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
**StyleFillFillType Enumeration**

Enum for the type of the color

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public enum FillType
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>patternColor</td>
<td>0</td>
<td>Color defines a pattern color</td>
</tr>
<tr>
<td>fillColor</td>
<td>1</td>
<td>Color defines a solid fill color</td>
</tr>
</tbody>
</table>

### See Also

Reference  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFillPatternValue Enumeration

Enum for the pattern values

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public enum PatternValue
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>0</td>
<td>No pattern (default)</td>
</tr>
<tr>
<td>solid</td>
<td>1</td>
<td>Solid fill (for colors)</td>
</tr>
<tr>
<td>darkGray</td>
<td>2</td>
<td>Dark gray fill</td>
</tr>
<tr>
<td>mediumGray</td>
<td>3</td>
<td>Medium gray fill</td>
</tr>
<tr>
<td>lightGray</td>
<td>4</td>
<td>Light gray fill</td>
</tr>
<tr>
<td>gray0625</td>
<td>5</td>
<td>6.25% gray fill</td>
</tr>
<tr>
<td>gray125</td>
<td>6</td>
<td>12.5% gray fill</td>
</tr>
</tbody>
</table>

### See Also
StyleFont Class

Class representing a Font entry. The Font entry is used to define text formatting.

Inheritance Hierarchy

```
SystemObject    NanoXLSX.StyleAbstractStyle
    NanoXLSX.StyleStyleFont
```

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```
public class Font : AbstractStyle
```

The StyleFont type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleFont</td>
<td>Default constructor</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bold</td>
<td>Gets or sets whether the font is bold. If true, the font is declared bold.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Charset</strong></td>
<td>Gets or sets the char set of the Font (Default is empty)</td>
</tr>
<tr>
<td><strong>ColorTheme</strong></td>
<td>Gets or sets the font color theme (Default is 1)</td>
</tr>
<tr>
<td><strong>ColorValue</strong></td>
<td>Gets or sets the font color (default is empty)</td>
</tr>
<tr>
<td><strong>DoubleUnderline</strong></td>
<td>Gets or sets whether the font has a double underline. If true, the font is declared with a double underline</td>
</tr>
<tr>
<td><strong>Family</strong></td>
<td>Gets or sets the font family (Default is 2)</td>
</tr>
<tr>
<td><strong>Hash</strong></td>
<td>Gets the unique hash of the object (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td><strong>InternalID</strong></td>
<td>Gets or sets the internal ID for sorting purpose in the Excel style document (nullable) (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td><strong>IsDefaultFont</strong></td>
<td>Gets whether the font is equals the default font</td>
</tr>
<tr>
<td><strong>Italic</strong></td>
<td>Gets or sets whether the font is italic. If true, the font is declared as italic</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Gets or sets the font name (Default is Calibri)</td>
</tr>
</tbody>
</table>
### Scheme
Gets or sets the font scheme (Default is minor)

### Size
Gets or sets the font size. Valid range is from 8 to 75

### Strike
Gets or sets whether the font is struck through. If true, the font is declared as strike-through

### Underline
Gets or sets whether the font is underlined. If true, the font is declared as underlined

### VerticalAlign
Gets or sets the alignment of the font (Default is none)

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalculateHash</td>
<td>Override method to calculate the hash of this component (Overrides AbstractStyleCalculateHash.)</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Method to compare two objects for sorting purpose (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Copy</td>
<td>Method to copy the current object to a new one without casting (Overrides AbstractStyle.Copy.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CopyFont</td>
<td>Method to copy the current object to a new one with casting</td>
</tr>
<tr>
<td>CopyPropertiesT</td>
<td>Internal method to copy altered properties from a source object. The decision whether a property is copied is dependent on a untouched reference object (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Equals(Object)</td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Equals(AbstractStyle)</td>
<td>Method to compare two objects for sorting purpose (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
</tbody>
</table>
MemberwiseClone

Creates a shallow copy of the current Object.
(Inherited from Object.)

ToString

Override toString method
(Overrides Object.ToString.)

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULTFONT</td>
<td>Default font family as constant</td>
</tr>
</tbody>
</table>

See Also

Reference
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFont Constructor

Default constructor

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Font()
```

### See Also

Reference  
**StyleFont Class**  
**NanoXLSX.Style Namespace**

Copyright Raphael Stoeckli © 2018
# Font Properties

The **StyleFont** type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bold</td>
<td>Gets or sets whether the font is bold. If true, the font is declared as bold</td>
</tr>
<tr>
<td>Charset</td>
<td>Gets or sets the char set of the Font (Default is empty)</td>
</tr>
<tr>
<td>ColorTheme</td>
<td>Gets or sets the font color theme (Default is 1)</td>
</tr>
<tr>
<td>ColorValue</td>
<td>Gets or sets the font color (default is empty)</td>
</tr>
<tr>
<td>DoubleUnderline</td>
<td>Gets or sets whether the font has a double underline. If true, the font is declared with a double underline</td>
</tr>
<tr>
<td>Family</td>
<td>Gets or sets the font family (Default is 2)</td>
</tr>
<tr>
<td>Hash</td>
<td>Gets the unique hash of the object (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>InternalID</td>
<td>Gets or sets the internal ID for sorting purpose in the Excel</td>
</tr>
</tbody>
</table>
style document (nullable)
(Inherited from AbstractStyle.)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsDefaultFont</td>
<td>Gets whether the font is equals the default font</td>
</tr>
<tr>
<td>Italic</td>
<td>Gets or sets whether the font is italic. If true, the font is declared as italic</td>
</tr>
<tr>
<td>Name</td>
<td>Gets or sets the font name (Default is Calibri)</td>
</tr>
<tr>
<td>Scheme</td>
<td>Gets or sets the font scheme (Default is minor)</td>
</tr>
<tr>
<td>Size</td>
<td>Gets or sets the font size. Valid range is from 8 to 75</td>
</tr>
<tr>
<td>Strike</td>
<td>Gets or sets whether the font is struck through. If true, the font is declared as strike-through</td>
</tr>
<tr>
<td>Underline</td>
<td>Gets or sets whether the font is underlined. If true, the font is declared as underlined</td>
</tr>
<tr>
<td>VerticalAlign</td>
<td>Gets or sets the alignment of the font (Default is none)</td>
</tr>
</tbody>
</table>

**See Also**

Reference
Use the `Style.Font` property to configure font properties for a style. The `Font` class provides methods for configuring font properties such as size, style, and color. The `Font` class also includes methods for setting font effects such as underline and strikethrough. The `Font` class allows you to customize the appearance of text in a style document. When a style is applied to text, the `Font` class settings are used to determine the appearance of the text. If a font property is not set, the value is inherited from the parent style. If a font property is not defined for a style, the value is taken from the default value of the style. The `Font` class provides a way to customize the appearance of text in a style document. The `Font` class allows you to set the font size, style, color, and effects such as underline and strikethrough. The `Font` class also includes methods for setting font properties such as size, style, color, and effects. The `Font` class allows you to customize the appearance of text in a style document. When a style is applied to text, the `Font` class settings are used to determine the appearance of the text. If a font property is not set, the value is inherited from the parent style. If a font property is not defined for a style, the value is taken from the default value of the style. The `Font` class provides a way to customize the appearance of text in a style document.
StyleFontBold Property

Gets or sets whether the font is bold. If true, the font is declared as bold.

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public bool Bold { get; set; }
```

**Property Value**  
Type: Boolean

### See Also

**Reference**  
StyleFont Class  
NanoXLSX.Style Namespace
StyleFontCharset Property

GETS or SETS the char set of the Font (Default is empty)

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public string Charset { get; set; }
```

Property Value

Type: String

See Also

Reference
StyleFont Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFontColorTheme Property

Gets or sets the font color theme (Default is 1)

Namespace:  NanoXLSX.Style
Assembly:  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public int ColorTheme { get; set; }
```

Property Value
Type: Int32

See Also

Reference
StyleFont Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
**StyleFontColorValue Property**

Gets or sets the font color (default is empty)

**Namespace:** NanoLSX.Style  
**Assembly:** NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string ColorValue { get; set; }
```

**Property Value**

Type: `String`

### See Also

**Reference**
- StyleFont Class
- NanoLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFontDoubleUnderline

Property

Gets or sets whether the font has a double underline. If true, the font is declared with a double underline.

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```
public bool DoubleUnderline { get; set; }
```

Property Value

Type: Boolean

See Also

Reference
- StyleFont Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFontFamily Property

Gets or sets the font family (Default is 2)

**Namespace:** NanoXLexX.Style  
**Assembly:** NanoXLexX (in NanoXLexX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string Family { get; set; }
```

**Property Value**  
Type: **String**

### See Also

**Reference**  
*StyleFont Class*  
*NanoXLexX.Style Namespace*

Copyright Raphael Stoeckli © 2018
**StyleFont.IsDefaultFont Property**

Gets whether the font is equals the default font

**Namespace:**  NanoXLSX.Style  
**Assembly:**  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public bool IsDefaultFont { get; }
```

**Property Value**

Type:  Boolean

### See Also

**Reference**

- StyleFont Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFontItalic Property

Gets or sets whether the font is italic. If true, the font is declared as italic.

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public bool Italic { get; set; }
```

Property Value  
Type: Boolean

### See Also

- Reference  
  - StyleFont Class  
  - NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFontName Property

Gets or sets the font name (Default is Calibri)

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string Name { get; set; }
```

Property Value  
Type: **String**

### See Also

**Reference**  
*StyleFont Class*  
*NanoXLSX.Style Namespace*

Copyright Raphael Stoeckli © 2018
StyleFontScheme Property

Gets or sets the font scheme (Default is minor)

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public StyleFontSchemeValue Scheme { get; set; }
```

Property Value
Type: StyleFontSchemeValue

See Also

Reference
StyleFont Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFontGetSize Property

Gets or sets the font size. Valid range is from 8 to 75

**Namespace:** NanoXLSSX.Style  
**Assembly:** NanoXLSSX (in NanoXLSSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

C#  
```
public int Size { get; set; }
```

### Property Value

Type: **Int32**

### See Also

Reference  
- **StyleFont Class**  
- NanoXLSSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFontStrike Property

Gets or sets whether the font is struck through. If true, the font is declared as strike-through

**Namespace:** NanoXLSX.Style
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```c#
public bool Strike { get; set; }
```

Property Value
Type: Boolean

## See Also

Reference
- StyleFont Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFontUnderline Property

Gets or sets whether the font is underlined. If true, the font is declared as underlined.

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public bool Underline { get; set; }
```

**Property Value**  
Type: Boolean

### See Also

- **Reference**  
  - StyleFont Class  
  - NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFontVerticalAlign Property

Gets or sets the alignment of the font (Default is none)

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleFontVerticalAlignValue VerticalAlign
```

### Property Value

**Type:** StyleFontVerticalAlignValue

### See Also

- **Reference**  
  - StyleFont Class  
  - NanoXLSX.Style Namespace

---

Copyright Raphael Stoeckli © 2018
Font Methods

The **StyleFont** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalculateHash</td>
<td>Override method to calculate the hash of this component (Overrides AbstractStyleCalculateHash.)</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Method to compare two objects for sorting purpose (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Copy</td>
<td>Method to copy the current object to a new one without casting (Overrides AbstractStyleCopy.)</td>
</tr>
<tr>
<td>CopyFont</td>
<td>Method to copy the current object to a new one with casting</td>
</tr>
<tr>
<td>CopyPropertiesT</td>
<td>Internal method to copy altered properties from a source object. The decision whether a property is copied is dependent on a untouched reference object (Inherited from)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>AbstractStyle.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Equals(Object)</strong></td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>(Inherited from Object.)</td>
<td></td>
</tr>
<tr>
<td><strong>Equals(AbstractStyle)</strong></td>
<td>Method to compare two objects for sorting purpose</td>
</tr>
<tr>
<td>(Inherited from AbstractStyle.)</td>
<td></td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>(Inherited from Object.)</td>
<td></td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type.</td>
</tr>
<tr>
<td>(Inherited from Object.)</td>
<td></td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>(Inherited from Object.)</td>
<td></td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td>(Inherited from Object.)</td>
<td></td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Override toString method</td>
</tr>
<tr>
<td>(Overrides Object.ToString.)</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**
Reference
StyleFont Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFontCalculateHash Method

Override method to calculate the hash of this component

**Namespace:** NanoXLSX.Style
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public override string CalculateHash()
```

### Return Value
Type: String
Calculated hash as string

### See Also

Reference
StyleFont Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFontCopy Method

Method to copy the current object to a new one without casting

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public override AbstractStyle Copy()
```

**Return Value**

Type: AbstractStyle  
Copy of the current object without the internal ID

### See Also

**Reference**

- StyleFont Class
- NanoXLSX.Style Namespace
StyleFontCopyFont Method

Method to copy the current object to a new one with casting

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public StyleFont CopyFont()
```

**Return Value**

Type: **StyleFont**  
Copy of the current object without the internal ID

**See Also**

**Reference**  
[StyleFont Class](#)  
[NanoXLSX.Style Namespace](#)

Copyright Raphael Stoeckli © 2018
StyleFontToString Method

Override toString method

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

▲ Syntax

```csharp
public override string ToString()
```

Return Value
Type: String
String of a class

▲ See Also

Reference
StyleFont Class  
NanoXLSX.Style Namespace
Font Fields

The `StyleFont` type exposes the following members.

**Fields**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULTFONT</td>
<td>Default font family as constant</td>
</tr>
</tbody>
</table>

**See Also**

Reference
- `StyleFont Class`
- `NanoXLSX.Style Namespace`

Copyright Raphael Stoeckli © 2018
# StyleFontDEFAULTFONT Field

Default font family as constant

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```
public const string DEFAULTFONT = "Calibri"
```

**Field Value**  
**Type:** String

## See Also

Reference  
**StyleFont Class**  
**NanoXLSX.Style Namespace**

Copyright Raphael Stoeckli © 2018
Style Font Scheme Value
Enumeration

Enum for the font scheme

**Namespace:** NanoXLSX.Style
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public enum SchemeValue
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>major</td>
<td>0</td>
<td>Font scheme is major</td>
</tr>
<tr>
<td>minor</td>
<td>1</td>
<td>Font scheme is minor (default)</td>
</tr>
<tr>
<td>none</td>
<td>2</td>
<td>No Font scheme is used</td>
</tr>
</tbody>
</table>

### See Also

Reference
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleFontVerticalAlignValue
 Enumeration

Enum for the vertical alignment of the text from base line

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public enum VerticalAlignValue
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subscript</td>
<td>0</td>
<td>Text will be rendered as subscript</td>
</tr>
<tr>
<td>superscript</td>
<td>1</td>
<td>Text will be rendered as superscript</td>
</tr>
<tr>
<td>none</td>
<td>2</td>
<td>Text will be rendered normal</td>
</tr>
</tbody>
</table>

### See Also

Reference  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleNumberFormat Class

Class representing a NumberFormat entry. The NumberFormat entry is used to define cell formats like currency or date.

Inheritance Hierarchy

- SystemObject
  - NanoXLSX.StyleAbstractStyle
    - NanoXLSX.StyleStyleNumberFormat

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public class NumberFormat : AbstractStyle
```

The StyleNumberFormat type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleNumberFormat</td>
<td>Default constructor</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustomFormatCode</td>
<td>Gets or sets the custom format code in the notation</td>
</tr>
</tbody>
</table>
of Excel

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustomFormatID</td>
<td>Gets or sets the format number of the custom format. Must be higher or equal then predefined custom number (164)</td>
</tr>
<tr>
<td>Hash</td>
<td>Gets the unique hash of the object (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>InternalID</td>
<td>Gets or sets the internal ID for sorting purpose in the Excel style document (nullable) (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>IsCustomFormat</td>
<td>Gets whether the number format is a custom format (higher or equals 164). If true, the format is custom</td>
</tr>
<tr>
<td>Number</td>
<td>Gets or sets the format number. Set this to custom (164) in case of custom number formats</td>
</tr>
</tbody>
</table>

Top

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalculateHash</td>
<td>Override method to calculate the hash of this component</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>CompareTo</td>
<td>Method to compare two objects for sorting purpose (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Copy</td>
<td>Method to copy the current object to a new one without casting (Overrides AbstractStyleCopy.)</td>
</tr>
<tr>
<td>CopyNumberFormat</td>
<td>Method to copy the current object to a new one with casting</td>
</tr>
<tr>
<td>CopyPropertiesT</td>
<td>Internal method to copy altered properties from a source object. The decision whether a property is copied is dependent on a untouched reference object (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Equals(Object)</td>
<td>Determines whether the specified Object is equal to the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Equals(AbstractStyle)</td>
<td>Method to compare two objects for sorting purpose (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free</td>
</tr>
</tbody>
</table>
resources and perform other cleanup operations before it is reclaimed by garbage collection.
(Inherited from Object.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Override toString method (Overrides Object.ToString.)</td>
</tr>
</tbody>
</table>

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMFORMAT_START_NUMBER</td>
<td>Start ID for custom number formats as constant</td>
</tr>
</tbody>
</table>

### See Also
StyleNumberFormat Constructor

Default constructor

Namespace: NanoXSX.Style
Assembly: NanoXSX (in NanoXSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public NumberFormat()
```

See Also

Reference
- StyleNumberFormat Class
- NanoXSX.Style Namespace

Copyright Raphael Stoeckli © 2018
NumberFormat Properties

The `StyleNumberFormat` type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustomFormatCode</td>
<td>Gets or sets the custom format code in the notation of Excel</td>
</tr>
<tr>
<td>CustomFormatID</td>
<td>Gets or sets the format number of the custom format. Must be higher or equal then predefined custom number (164)</td>
</tr>
<tr>
<td>Hash</td>
<td>Gets the unique hash of the object (Inherited from <code>AbstractStyle</code>.)</td>
</tr>
<tr>
<td>InternalID</td>
<td>Gets or sets the internal ID for sorting purpose in the Excel style document (nullable) (Inherited from <code>AbstractStyle</code>.)</td>
</tr>
<tr>
<td>IsCustomFormat</td>
<td>Gets whether the number format is a custom format (higher or equals 164). If true, the format is custom</td>
</tr>
</tbody>
</table>
**Number**

Gives or sets the format number. Set this to custom (164) in case of custom number formats.

---

**See Also**

- **Reference**
  - [StyleSheetFormat Class](#)
  - [NanoXLSX.Style Namespace](#)

---

Copyright Raphael Stoeckli © 2018
StyleNumberFormatCustomFormatCode Property

Gets or sets the custom format code in the notation of Excel

**Namespace:**  NanoXLSX.Style  
**Assembly:**  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public string CustomFormatCode { get; set; }
```

Property Value  
Type:  String

### See Also

Reference  
StyleNumberFormat Class  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleNumberFormatCustomFormatId Property

Gets or sets the format number of the custom format. Must be higher or equal then predefined custom number (164)

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public int CustomFormatID { get; set; }
```

#### Property Value

Type: `Int32`

### See Also

- Reference
  - `StyleNumberFormat Class`
  - `NanoXLSX.Style Namespace`

Copyright Raphael Stoeckli © 2018
StyleNumberFormat.IsCustomFormat Property

Gets whether the number format is a custom format (higher or equals 164). If true, the format is custom.

**Namespace:** NanoXLXSX.Style  
**Assembly:** NanoXLXSX (in NanoXLXSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public bool IsCustomFormat { get; }
```

**Property Value**

Type: Boolean

### See Also

**Reference**

- StyleNumberFormat Class
- NanoXLXSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleNumberFormatNumber Property

Gets or sets the format number. Set this to custom (164) in case of custom number formats

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public StyleNumberFormatNumber Number { get }
```

**Property Value**  
Type: `StyleNumberFormatNumber`

## See Also

**Reference**  
StyleNumberFormat Class  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
## NumberFormat Methods

The `StyleNumberFormat` type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CalculateHash</code></td>
<td>Override method to calculate the hash of this component (Overrides <code>AbstractStyleCalculateHash</code>.)</td>
</tr>
<tr>
<td><code>CompareTo</code></td>
<td>Method to compare two objects for sorting purpose (Inherited from <code>AbstractStyle</code>.)</td>
</tr>
<tr>
<td><code>Copy</code></td>
<td>Method to copy the current object to a new one without casting (Overrides <code>AbstractStyleCopy</code>.)</td>
</tr>
<tr>
<td><code>CopyNumberFormat</code></td>
<td>Method to copy the current object to a new one with casting</td>
</tr>
<tr>
<td><code>CopyPropertiesT</code></td>
<td>Internal method to copy altered properties from a source object. The decision whether a property is copied is dependent on a untouched reference object (Inherited from...</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>AbstractStyle.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Equals(Object)</strong></td>
<td>Determines whether the specified Object is equal to the current Object.     (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Equals(AbstractStyle)</strong></td>
<td>Method to compare two objects for sorting purpose (Inherited from AbstractStyle.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Override toString method (Overrides ObjectToString.)</td>
</tr>
</tbody>
</table>

**See Also**
Reference

StyleNumberFormat Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleNumberFormatCalculateHash Method

Override method to calculate the hash of this component

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
public override string CalculateHash()
```

**Return Value**

**Type:** String  
Calculated hash as string

## See Also

Reference

- **StyleNumberFormat Class**
- **NanoXLSX.Style Namespace**

Copyright Raphael Stoeckli © 2018
StyleNumberFormatCopy Method

Method to copy the current object to a new one without casting

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public override AbstractStyle Copy()
```

Return Value
Type: AbstractStyle
Copy of the current object without the internal ID

See Also

Reference
StyleNumberFormat Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleNumberFormatCopyNumberFormat Method

Method to copy the current object to a new one with casting

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleNumberFormat CopyNumberFormat()
```

**Return Value**

Type: `StyleNumberFormat`  
Copy of the current object without the internal ID

### See Also

**Reference**

- `StyleNumberFormat Class`
- `NanoXLSX.Style Namespace`

Copyright Raphael Stoeckli © 2018
StyleNumberFormatToString Method

Override toString method

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public override string ToString()
```

**Return Value**
- Type: `String`
- String of a class

### See Also

Reference
- StyleNumberFormat Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
NumberFormat Fields

The `StyleNumberFormat` type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMFORMAT_START_NUMBER</td>
<td>Start ID for custom number formats as constant</td>
</tr>
</tbody>
</table>

See Also

Reference
- `StyleNumberFormat Class`
- `NanoXLSX.Style Namespace`
**StyleNumberFormatCUSTOMFORMATION Field**

Start ID for custom number formats as constant

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public const int CUSTOMFORMAT_START_NUMBER = 124
```

Field Value  
Type: Int32

### See Also

Reference  
StyleNumberFormat Class  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleNumberFormat

Enumeration

Enum for predefined number formats

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public enum FormatNumber
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>0</td>
<td>No format / Default</td>
</tr>
<tr>
<td>format_1</td>
<td>1</td>
<td>Format: 0</td>
</tr>
<tr>
<td>format_2</td>
<td>2</td>
<td>Format: 0.00</td>
</tr>
<tr>
<td>format_3</td>
<td>3</td>
<td>Format: #,##0</td>
</tr>
<tr>
<td>format_4</td>
<td>4</td>
<td>Format: #,##0.00</td>
</tr>
<tr>
<td>format_5</td>
<td>5</td>
<td>Format: $#,##0_);($#,##0)</td>
</tr>
<tr>
<td>format_6</td>
<td>6</td>
<td>Format: $#,##0_);<a href="$#,##0">Red</a></td>
</tr>
<tr>
<td>format_7</td>
<td>7</td>
<td>Format: $#,##0.00_);</td>
</tr>
<tr>
<td>Format</td>
<td>Precision</td>
<td>Format Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>format_8</td>
<td>8</td>
<td>Format: $#,##0.00_);[Red] ($#,##0.00)</td>
</tr>
<tr>
<td>format_9</td>
<td>9</td>
<td>Format: 0%</td>
</tr>
<tr>
<td>format_10</td>
<td>10</td>
<td>Format: 0.00%</td>
</tr>
<tr>
<td>format_11</td>
<td>11</td>
<td>Format: 0.00E+00</td>
</tr>
<tr>
<td>format_12</td>
<td>12</td>
<td>Format: # ?/?</td>
</tr>
<tr>
<td>format_13</td>
<td>13</td>
<td>Format: # ??/??</td>
</tr>
<tr>
<td>format_14</td>
<td>14</td>
<td>Format: m/d/yyyy</td>
</tr>
<tr>
<td>format_15</td>
<td>15</td>
<td>Format: d-mmm-yy</td>
</tr>
<tr>
<td>format_16</td>
<td>16</td>
<td>Format: d-mmm</td>
</tr>
<tr>
<td>format_17</td>
<td>17</td>
<td>Format: mmm-yy</td>
</tr>
<tr>
<td>format_18</td>
<td>18</td>
<td>Format: mm AM/PM</td>
</tr>
<tr>
<td>format_19</td>
<td>19</td>
<td>Format: h:mm:ss AM/PM</td>
</tr>
<tr>
<td>format_20</td>
<td>20</td>
<td>Format: h:mm</td>
</tr>
<tr>
<td>format_21</td>
<td>21</td>
<td>Format: h:mm:ss</td>
</tr>
<tr>
<td>format_22</td>
<td>22</td>
<td>Format: m/d/yyyy h:mm</td>
</tr>
<tr>
<td>format_37</td>
<td>37</td>
<td>Format: #,##0_);(#,##0)</td>
</tr>
<tr>
<td>format_38</td>
<td>38</td>
<td>Format: #,##0_);[Red] (#,##0)</td>
</tr>
<tr>
<td>format_39</td>
<td>39</td>
<td>Format: #,##0.00_);</td>
</tr>
<tr>
<td>#</td>
<td>Format</td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>---</td>
<td>----------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| 40 | #,##0.00 | Format: #,##0.00
| 45 | mm:ss   | Format: mm:ss   |
| 46 | [h]:mm:ss | Format: [h]:mm:ss |
| 47 | mm:ss.0 | Format: mm:ss.0 |
| 48 | ##0.0E+0 | Format: ##0.0E+0 |
| 49 | #       | Format: #       |
| 164 | Custom | Custom Format (ID 164 and higher) |

**See Also**

Reference

NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManager Class

Class representing a style manager to maintain all styles and its components of a workbook

Inheritance Hierarchy

System.Object  NanoXLSX.StyleStyleManager

Namespace:  NanoXLSX.Style
Assembly:  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public class StyleManager
```

The `StyleManager` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleManager</td>
<td>Default constructor</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddStyle</td>
<td>Adds a style component to the manager</td>
</tr>
<tr>
<td>Method/Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AddStyleComponent(AbstractStyle)</td>
<td>Adds a style component to the manager</td>
</tr>
<tr>
<td>AddStyleComponent(AbstractStyle, Nullable Int32)</td>
<td>Adds a style component to the manager with an ID</td>
</tr>
<tr>
<td>CleanupStyleComponents</td>
<td>Method to cleanup style components in the style manager</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetBorderByHash</td>
<td>Gets a border</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetBorders</td>
<td>Gets all borders of the style manager</td>
</tr>
<tr>
<td>GetBorderStyleNumber</td>
<td>Gets the number of borders in the style manager</td>
</tr>
<tr>
<td>GetCellXfByHash</td>
<td>Gets a cellXf by its hash</td>
</tr>
<tr>
<td>GetCellXfs</td>
<td>Gets all cellXfs of the style manager</td>
</tr>
<tr>
<td>GetCellXfStyleNumber</td>
<td>Gets the number of cellXfs in the style manager</td>
</tr>
<tr>
<td>GetComponentByHash</td>
<td>Gets a component by its hash</td>
</tr>
<tr>
<td>GetFillByHash</td>
<td>Gets a fill by its hash</td>
</tr>
<tr>
<td>GetFills</td>
<td>Gets all fills of the style manager</td>
</tr>
<tr>
<td>GetFillStyleNumber</td>
<td>Gets the number of fills in the style manager</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetFontByHash</td>
<td>Gets a font by its hash</td>
</tr>
<tr>
<td>GetFonts</td>
<td>Gets all fonts of the style manager</td>
</tr>
<tr>
<td>GetFontStyleNumber</td>
<td>Gets the number of fonts in the style manager</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetNumberFormatByHash</td>
<td>Gets a numberFormat by its hash</td>
</tr>
<tr>
<td>GetNumberFormats</td>
<td>Gets all numberFormats of the style manager</td>
</tr>
<tr>
<td>GetNumberFormatStyleNumber</td>
<td>Gets the number of numberFormats in the style manager</td>
</tr>
<tr>
<td>GetStyleByHash</td>
<td>Gets a style by its hash</td>
</tr>
<tr>
<td>GetStyleByName</td>
<td>Gets a style by</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetStyleNumber</td>
<td>Gets the number of styles in the style manager</td>
</tr>
<tr>
<td>GetStyles</td>
<td>Gets all styles of the style manager</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>IsUsedByStyle</td>
<td>Checks whether a style component in the style manager is used by a style</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>RemoveStyle</td>
<td>Removes a style and all its components from the style manager</td>
</tr>
<tr>
<td>Reorganize</td>
<td>Method to</td>
</tr>
</tbody>
</table>
reorganize / reorder a list of style components

ToString

Returns a string that represents the current object. (Inherited from Object.)

Field:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BORDERPREFIX</td>
<td>Prefix for the hash calculation of border styles</td>
</tr>
<tr>
<td>CELLXFPREFIX</td>
<td>Prefix for the hash calculation of cellXf styles</td>
</tr>
<tr>
<td>FILLPREFIX</td>
<td>Prefix for the hash calculation of fill styles</td>
</tr>
<tr>
<td>FONTPREFIX</td>
<td>Prefix for the hash calculation of font styles</td>
</tr>
<tr>
<td>NUMBERFORMATPREFIX</td>
<td>Prefix for the hash calculation of number format styles</td>
</tr>
</tbody>
</table>
STYLEPREFIX

Prefix for the hash calculation of styles

See Also

Reference
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManager Constructor

Default constructor

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleManager()
```

### See Also

Reference

- StyleManager Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
# StyleManager Methods

The `StyleManager` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddStyle</td>
<td>Adds a style component to the manager</td>
</tr>
<tr>
<td>AddStyleComponent(AbstractStyle)</td>
<td>Adds a style component to the manager</td>
</tr>
<tr>
<td>AddStyleComponent(AbstractStyle, NullableInt32)</td>
<td>Adds a style component to the manager with an ID</td>
</tr>
<tr>
<td>CleanupStyleComponents</td>
<td>Method to cleanup style components in the style manager</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetBorderByHash</strong></td>
<td>Gets a border by its hash</td>
</tr>
<tr>
<td><strong>GetBorders</strong></td>
<td>Gets all borders of the style manager</td>
</tr>
<tr>
<td><strong>GetBorderStyleNumber</strong></td>
<td>Gets the number of borders in the style manager</td>
</tr>
<tr>
<td><strong>GetCellXfByHash</strong></td>
<td>Gets a cellXf by its hash</td>
</tr>
<tr>
<td><strong>GetCellXfs</strong></td>
<td>Gets all cellXfs of the style manager</td>
</tr>
<tr>
<td><strong>GetCellXfStyleNumber</strong></td>
<td>Gets the number of cellXfs in the style manager</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetComponentByHash</td>
<td>Gets a component by its hash</td>
</tr>
<tr>
<td>GetFillByHash</td>
<td>Gets a fill by its hash</td>
</tr>
<tr>
<td>GetFills</td>
<td>Gets all fills of the style manager</td>
</tr>
<tr>
<td>GetFillStyleNumber</td>
<td>Gets the number of fills in the style manager</td>
</tr>
<tr>
<td>GetFontByHash</td>
<td>Gets a font by its hash</td>
</tr>
<tr>
<td>GetFonts</td>
<td>Gets all fonts of the style manager</td>
</tr>
<tr>
<td>GetFontStyleNumber</td>
<td>Gets the number of fonts in the style manager</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetNumberFormatByHash</td>
<td>Gets a numberFormat by its hash</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetNumberFormats</td>
<td>Gets all numberFormats of the style manager</td>
</tr>
<tr>
<td>GetNumberFormatStyleNumber</td>
<td>Gets the number of numberFormats in the style manager</td>
</tr>
<tr>
<td>GetStyleByHash</td>
<td>Gets a style by its hash</td>
</tr>
<tr>
<td>GetStyleByName</td>
<td>Gets a style by its name</td>
</tr>
<tr>
<td>GetStyleNumber</td>
<td>Gets the number of styles in the style manager</td>
</tr>
<tr>
<td>GetStyles</td>
<td>Gets all styles of the style manager</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>IsUsedByStyle</td>
<td>Checks whether a style component in the style manager is</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RemoveStyle</td>
<td>Removes a style and all its components from the style manager</td>
</tr>
<tr>
<td>Reorganize</td>
<td>Method to reorganize / reorder a list of style components</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference
- StyleManager Class
- NanoXLSX.Style Namespace
**StyleManager.AddStyle Method**

Adds a style component to the manager

**Namespace**: NanoLSX.Style

**Assembly**: NanoLSX (in NanoLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Style AddStyle(
    Style style
)
```

**Parameters**

- **style**
  - Type: NanoLSX.Style
  - Style to add

**Return Value**

- Type: Style
  - Added or determined style in the manager

### See Also

**Reference**

- StyleManager Class
- NanoLSX.Style Namespace
StyleManager AddStyleComponent Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="AbstractStyle" alt="AddStyleComponent" /></td>
<td>Adds a style component to the manager</td>
</tr>
<tr>
<td>![AddStyleComponent](AbstractStyle, NullableInt32)</td>
<td>Adds a style component to the manager with an ID</td>
</tr>
</tbody>
</table>

Top

See Also

Reference

StyleManager Class
NanoXLStyle Namespace

Copyright Raphael Stoeckli © 2018
StyleManager

AddStyleComponent Method (AbstractStyle)

Adds a style component to the manager

**Namespace:** NanoXSX.Style
**Assembly:** NanoXSX (in NanoXSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private string AddStyleComponent(
    AbstractStyle style
)
```

### Parameters

**style**
- Type: NanoXSX.StyleAbstractStyle
- Component to add

### Return Value

Type: String
- Hash of the added or determined component

### See Also

**Reference**
- StyleManager Class
- AddStyleComponent Overload
- NanoXSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManager

AddStyleComponent Method (AbstractStyle, Nullable<Int32>)

Adds a style component to the manager with an ID

**Namespace:** NanoXLSX.Style
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
private string AddStyleComponent(
    AbstractStyle style,
    Nullable<int> id
)
```

### Parameters

**style**
- Type: NanoXLSX.StyleAbstractStyle
  - Component to add

**id**
- Type: System.Nullable<Int32>
  - Id of the component

### Return Value

Type: String
- Hash of the added or determined component

### See Also
Reference

StyleManager Class
AddStyleComponent Overload
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManager

CleanupStyleComponents

Method

Method to cleanup style components in the style manager

Namespace: NanoXLXSX.Style
Assembly: NanoXLXSX (in NanoXLXSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```c#
private void CleanupStyleComponents()
```

See Also

Reference
StyleManager Class
NanoXLXSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManager.GetBorderByHash Method

Gets a border by its hash

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleBorder GetBorderByHash(
    string hash
)
```

### Parameters

- **hash**
  - Type: `System.String`  
  - Hash of the border

### Return Value

- Type: `StyleBorder`  
- Determined border

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws a StyleException if the border was not found in the style manager</td>
</tr>
</tbody>
</table>

### See Also
Reference

**StyleManager Class**
**NanoXLSX.Style Namespace**

Copyright Raphael Stoeckli © 2018
**StyleManager.GetBorders Method**

Gets all borders of the style manager

**Namespace:** NanoXLXSX.Style  
**Assembly:** NanoXLXSX (in NanoXLXSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleBorderStyle[] GetBorders()
```

### Return Value

Type: **StyleBorderStyle**  
Array of borders

### See Also

Reference

- **StyleManager Class**  
- NanoXLXSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManager.GetBorderStyleNumber Method

Gets the number of borders in the style manager

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public int GetBorderStyleNumber()
```

**Return Value**

Type: Int32  
Number of stored borders

### See Also

**Reference**
- StyleManager Class  
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManager.GetCellXfByHash Method

Gets a cellXf by its hash

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleCellXf GetCellXfByHash(  
    string hash
)
```

### Parameters

- **hash**
  - Type: System.String
  - Hash of the cellXf

### Return Value

- Type: StyleCellXf
  - Determined cellXf

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws a.StyleException if the cellXf was not found in the style manager</td>
</tr>
</tbody>
</table>

### See Also
**StyleManagerGetCellXfs Method**

Gets all cellXfs of the style manager

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleCellXf[] GetCellXfs()
```

### Return Value

Type: `StyleCellXf`  
Array of cellXfs

### See Also

Reference  
- StyleManager Class  
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
**StyleManager.GetCellXfStyleNumber**

Gets the number of cellXfs in the style manager

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public int GetCellXfStyleNumber()
```

### Return Value

Type: **Int32**  
Number of stored cellXfs

### See Also

**Reference**  
[StyleManager Class]  
[NanoXLSX.Style Namespace]

---

Copyright Raphael Stoeckli © 2018
StyleManagerGetComponentByHash Method

Gets a component by its hash

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

## Syntax

```csharp
private AbstractStyle GetComponentByHash(
    ref List<AbstractStyle> list,
    string hash
)
```

### Parameters

- **list**
  - Type: `System.Collections.Generic.List<AbstractStyle>`
  - List to check
- **hash**
  - Type: `System.String`
  - Hash of the component

### Return Value

- Type: `AbstractStyle`
  - Determined component. If not found, null will be returned

## See Also

- Reference
  - StyleManager Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManagerGetFillByHash Method

Gets a fill by its hash

**Namespace:** NanoXLX.Style  
**Assembly:** NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```c#
public StyleFill GetFillByHash(
    string hash
)
```

**Parameters**

`hash`  
Type: `System.String`  
Hash of the fill

**Return Value**

Type: `StyleFill`  
Determined fill

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StyleException</code></td>
<td>Throws a <code>StyleException</code> if the fill was not found in the style manager</td>
</tr>
</tbody>
</table>

### See Also
Reference

StyleManager Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManager.GetFills Method

Gets all fills of the style manager

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public StyleFill[] GetFills()
```

Return Value
Type: StyleFill
Array of fills

See Also

Reference
StyleManager Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManager.GetFillStyleNumber Method

Gets the number of fills in the style manager

**Namespace:** NanoXLSX.Style
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

**Syntax**

```csharp
public int GetFillStyleNumber()
```

**Return Value**
- **Type:** Int32
- Number of stored fills

**See Also**

Reference
- StyleManager Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManager.GetFontByHash Method

Gets a font by its hash

**Namespace:** NanoXLX.Style  
**Assembly:** NanoXLX (in NanoXLX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleFont GetFontByHash(  
    string hash
)
```

### Parameters

**hash**
- **Type:** System.String
- **Hash of the font**

### Return Value

**Type:** StyleFont  
**Determined font**

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws a StyleException if the font was not found in the style manager</td>
</tr>
</tbody>
</table>

### See Also
Reference

StyleManager Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManager.GetFonts Method

Gets all fonts of the style manager

**Namespace:**  NanoXLSX.Style  
**Assembly:**  NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public StyleFont[] GetFonts()
```

### Return Value

Type:  `StyleFont`  
Array of fonts

### See Also

Reference
- StyleManager Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManagerGetFontStyleNumber Method

Gets the number of fonts in the style manager

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
public int GetFontStyleNumber()
```

Return Value
Type: Int32
Number of stored fonts

See Also

Reference
StyleManager Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
**StyleManagerGetNumberFormatByHash Method**

Gets a numberFormat by its hash

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public StyleNumberFormat GetNumberFormatByHash(
    string hash
)
```

### Parameters

- **hash**
  - Type: System.String  
  - Hash of the numberFormat

### Return Value

- Type: StyleNumberFormat  
  - Determined numberFormat

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws a StyleException if the numberFormat was not found in the style manager</td>
</tr>
</tbody>
</table>

### See Also
Reference
StyleManager Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManager.GetNumberFormats Method

Gets all numberFormats of the style manager

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public StyleNumberFormat[] GetNumberFormats()
```

### Return Value

Type: **StyleNumberFormat**  
Array of numberFormats

### See Also

**Reference**
- **StyleManager Class**
- **NanoXLSX.Style Namespace**

Copyright Raphael Stoeckli © 2018
StyleManagerGetNumberFormatStyleNumber Method

Gets the number of numberFormats in the style manager

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public int GetNumberFormatStyleNumber()
```

**Return Value**

Type: Int32  
Number of stored numberFormats

### See Also

Reference  
[StyleManager Class](#)  
[NanoXLSX.Style Namespace](#)

Copyright Raphael Stoeckli © 2018
StyleManager

GetStyleByHash Method

Gets a style by its hash

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Style GetStyleByHash(
    string hash
)
```

### Parameters

**hash**
- Type: **System.String**
- Hash of the style

### Return Value

Type: **Style**
- Determined style

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws a StyleException if the style was not found in the style manager</td>
</tr>
</tbody>
</table>

### See Also
Reference

StyleManager Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManager.GetStyleByName Method

Gets a style by its name

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Style GetStyleByName(
    string name
)
```

### Parameters

- **name**
  
  Type: System.String  
  Name of the style

### Return Value

- Type: Style  
  Determined style

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws a StyleException if the style was not found in the style manager</td>
</tr>
</tbody>
</table>

### See Also
Reference
StyleManager Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManager

GetStyleNumber Method

Gets the number of styles in the style manager

**Namespace:** NanoXLSX.Style

**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public int GetStyleNumber()
```

### Return Value

Type: `Int32`

Number of stored styles

### See Also

Reference

- StyleManager Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManager.GetStyles Method

Gets all styles of the style manager

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public Style[] GetStyles()
```

**Return Value**

Type: Style

Array of styles

### See Also

Reference

- StyleManager Class
- NanoXLSX.Style Namespace
StyleManagerIsUsedByStyle Method

Checks whether a style component in the style manager is used by a style

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

C#

```csharp
private bool IsUsedByStyle(
    AbstractStyle component
)
```

Parameters

component
Type: NanoXLSX.StyleAbstractStyle
Component to check

Return Value
Type: Boolean
If true, the component is in use

See Also

Reference
StyleManager Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManager.RemoveStyle Method

Removes a style and all its components from the style manager

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public void RemoveStyle(
    string styleName
)
```

### Parameters

**styleName**
- **Type:** System.String
- Name of the style to remove

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleException</td>
<td>Throws a StyleException if the style was not found in the style manager</td>
</tr>
</tbody>
</table>

### See Also

Reference
- StyleManager Class  
- NanoXLSX.Style Namespace
StyleManagerReorganize Method

Method to reorganize / reorder a list of style components

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```csharp
private void Reorganize(
    ref List<AbstractStyle> list
)
```

Parameters

`list`
Type: System.Collections.Generic.List<AbstractStyle>
List to reorganize as reference

See Also

Reference
StyleManager Class
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
# StyleManager Fields

The `StyleManager` type exposes the following members.

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BORDERPREFIX</code></td>
<td>Prefix for the hash calculation of border styles</td>
</tr>
<tr>
<td><code>CELLXFPREFIX</code></td>
<td>Prefix for the hash calculation of cellXf styles</td>
</tr>
<tr>
<td><code>FILLPREFIX</code></td>
<td>Prefix for the hash calculation of fill styles</td>
</tr>
<tr>
<td><code>FONTPREFIX</code></td>
<td>Prefix for the hash calculation of font styles</td>
</tr>
<tr>
<td><code>NUMBERFORMATPREFIX</code></td>
<td>Prefix for the hash calculation of number format styles</td>
</tr>
<tr>
<td><code>STYLEPREFIX</code></td>
<td>Prefix for the hash calculation of styles</td>
</tr>
</tbody>
</table>

## See Also
StyleManagerBORDERPREFIX Field

Prefix for the hash calculation of border styles

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

⚠️ Syntax

```csharp
public const string BORDERPREFIX = "borders@"
```

Field Value  
Type: String

⚠️ See Also

Reference  
StyleManager Class  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManagerCELLXFPPREFIX Field

Prefix for the hash calculation of cellXf styles

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public const string CELLXFPPREFIX = "/cellXf@"
```

**Field Value**  
**Type:** String

### See Also

Reference  
StyleManager Class  
NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManagerFILLPREFIX Field

Prefix for the hash calculation of fill styles

Namespace: NanoXLSX.Style
Assembly: NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

Syntax

```
public const string FILLPREFIX = "/fill@
```

Field Value
Type: String

See Also

Reference
StyleManager Class
NanoXLSX.Style Namespace
StyleManager.FONTPREFIX Field

Prefix for the hash calculation of font styles

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```csharp
public const string FONTPREFIX = "/font@
```

**Field Value**

**Type:** String

### See Also

**Reference**

- StyleManager Class
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
StyleManager.NUMBERFORMATPREFIX Field

Prefix for the hash calculation of number format styles

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

### Syntax

```
public const string NUMBERFORMATPREFIX = "/numberFormat/
```

**Field Value**  
**Type:** String

### See Also

**Reference**  
- StyleManager Class  
- NanoXLSX.Style Namespace

Copyright Raphael Stoeckli © 2018
NanoXLSX Library Documentation
StyleManager STYLEPREFIX Field

Prefix for the hash calculation of styles

**Namespace:** NanoXLSX.Style  
**Assembly:** NanoXLSX (in NanoXLSX.dll) Version: 1.2.3.0 (1.2.3)

⚠️ **Syntax**

```c#
public const string STYLEPREFIX = "style=
```

Field Value  
Type: String

⚠️ **See Also**

Reference  
StyleManager Class  
NanoXLSX.Style Namespace