

# Documenting Web Projects

This topic contains the following sections:

- [Documenting Web Application and Web Service Application Projects](#)
- [Documenting Website and Web Service Website Projects](#)
- [Automating the Process](#)
- [Usage Notes and Known Issues](#)
- [How The Code Providers Work](#)
- [Conclusion](#)
- [See Also](#)

A question that comes up with some regularity is how to produce a help file from the XML comments in the code for a web project using the [Sandcastle Help File Builder](#). There are two types of web project: a *web application* project and a *website* project (both include a web service and an AJAX-enabled variant of the project type). A web application project is the simplest to document as it generates a single assembly with a fixed name much like any other non-web project. A website project, however, requires some extra effort to document and has a number of limitations:

- It does not necessarily generate a single assembly.
- It only generates assemblies when you actually publish the project.
- The assembly names may vary from build to build depending on the options you use when publishing the website.
- There is no project-level option to produce XML comments files. Instead, you have to add a section to the *Web.config* file to produce them.
- Even after adding the option to produce XML comments files, there are limitations:
  - If you specify a single comments filename, only the comments from the last assembly compiled will be saved. The VB.NET compiler has a [/doc+](#) option to work around this issue. However, the C# compiler does not so this makes it

impossible to fully document a C# website project.

- The comments files produced using the VB.NET [/doc+](#) option are not placed in the folder to which the project was published. Instead, they are left in the system's temporary ASP.NET folder for the related .NET version (i.e. `C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319\Temporary ASP.NET files`) so you must locate and retrieve them after each build.

To work around the limitations present when documenting website projects, you can use the supplied custom code providers (one for C# and one for VB.NET). By using these code providers, you can specify a folder to which the XML comments files should be copied. This saves you from having to locate and copy them manually and solves the issue of the C# compiler not being able to output one comments file per assembly. Below are instructions for documenting each type of web project and information about how the code providers work.

#### Note

The code providers assembly can be found in the `.\Extras` folder under the root help file builder installation folder (typically `C:\Program Files (x86)\EWSoftware\Sandcastle Help File Builder`).

## ▪ Documenting Web Application and Web Service Application Projects

As noted, web applications are the easiest to document and do not require the use of the custom code providers. These projects are created in Visual Studio by selecting **File | New | Project** and then selecting **ASP.NET Web Application** or **ASP.NET Web Service Application** from the **Web** category of your preferred language. Note that these options are only available in Visual Studio 2005 after you install Service Pack 1.

You must enable the XML comments option in your application projects so that an XML comments file is produced when the projects are built. This is done as follows:

1. In the Solution Explorer, right click on the project and select **Properties**.
2. Select the **Build** property page (C#) or the **Compile** property

page (VB.NET).

3. The XML comments filename is a per configuration option. As such, you can either select the **All Configurations** option at the top of the page to set the XML comments options for all configurations at the same time or select each configuration individually and repeat the next step for each one.
4. For C#, in the **Output** section, check the checkbox next to the **XML documentation file** text box and specify a name for the XML comments file. Although not required, a common convention is to name the XML comments file after the related assembly except with a *.xml* extension. The assembly name can be found on the **Application** property page.

For VB.NET, just check the **Generate XML documentation file** checkbox. It names the file after the assembly automatically.

5. If you have a solution with multiple web application or web service application projects that need to be documented, repeat the above steps for each project in the solution. If using the Sandcastle Help File Builder, be sure to give each project's XML comments file a unique name as they are copied to a common location for the build. Identically named files would overwrite each other and the documentation in the overwritten files would not appear in the help file.

Once the above has been done, Visual Studio will create the XML comments file each time the project is built. It is placed in the *.\bin* folder along with the assembly. These can be used in conjunction with your preferred documentation tool to produce a help file.

## ▪ Documenting Website and Web Service Website Projects

These projects are created in Visual Studio by selecting **File | New | Web Site** and then selecting **ASP.NET Web Site** or **ASP.NET Web Service** from the template dialog box. In order to create XML comments files, you must add the custom code providers to the *Web.config* file and publish the website. This is done as follows:

1. If your project does not yet contain one, add a *Web.config* configuration file.
2. Add the following `<system.codedom>` section to the

<configuration> section as shown in the example below. You only need to add the <compiler> element for the language that you are using but it will not hurt anything if they are both present.

**Example Code Provider Configurations** [Copy](#)

```
<configuration>
  <system.codedom>
    <compilers>
      <!-- For C# -->
      <compiler language="c#;cs;csharp" exten:
        compilerOptions="/docpath:C:\Publish\I
        type="EWSoftware.CodeDom.CSharpCodePr
          EWSoftware.CodeDom, Version=1.1
          PublicKeyToken=d633d7d5b41cbb65
        <!-- NOTE: Change version value as nee
        <providerOption name="CompilerVersion
      </compiler>

      <!-- For VB.NET -->
      <compiler language="vb;vbs;visualbasic;"'
        compilerOptions="/docpath:C:\Publish\I
        type="EWSoftware.CodeDom.VBCodeProvid
          EWSoftware.CodeDom, Version=1.1
          PublicKeyToken=d633d7d5b41cbb65
        <!-- NOTE: Change version value as nee
        <providerOption name="CompilerVersion
      </compiler>
    </compilers>
  </system.codedom>
</configuration>
```

Instead of a /doc option, the custom code providers use a custom /docpath option in the compilerOptions attribute that specifies the fully qualified folder name into which the XML comments files are placed. The folder will be created by the code provider if it does not exist. Any other compiler options that you might need can also be specified in the attribute value separated by a space.

3. In order to use the code providers, the ASP.NET compiler must be able to find them. You can either copy the **EWSoftware.CodeDom.dll** assembly into your project's `.\bin` folder or you can register it in the Global Assembly Cache (GAC) so that it is available to any project. To do so, use the **gacutil** command line tool as follows replacing the path to the assembly to match the installed location on your system:

[Copy](#)

```
gacutil -i \DotNet\Web\WebCodeProviders\Bin\R
```

To remove the assembly from the Global Assembly Cache, use the **gacutil** command line tool as follows:

[Copy](#)

```
gacutil -u EWSoftware.CodeDom
```

You can find a copy of the tool in the Visual Studio installation folder in the `.\\SDK\\v2.0\\bin` sub-folder.

Once the above has been done, you can publish your website. Right click on the project in the Solution Explorer, select **Publish Web Site**, set the options, and click OK to publish it. When finished, look in the location you specified in the `/docpath` option and you will find one XML comments file for each assembly that was built for your website. These can be used in conjunction with the assemblies and your preferred documentation tool to produce a help file.

As an alternative to publishing the website, you can use a web deployment project. Add-ins are available at the following locations:

- [Visual Studio 2005 Web Deployment Add-In](#)
- [Visual Studio 2008 Web Deployment Add-In](#)

The [Sandcastle Help File Builder](#) version 1.8.0.0 and later supports wildcard documentation sources. As such, just add one entry with the path set to `[appPath]\\bin\\App_*.dll` where `[appPath]` is the path to the published website and another entry with the path set to `[docPath]\\App_*.xml` where `[docPath]` is the path to the XML comments files.

Since the code providers are not needed after the website is published, you can delete the `system.codedom` section from the published `Web.Config` file and delete the assembly from the `.\\bin`

folder if you added it to the project for publishing.

## ▪ Automating the Process

Using the **aspnet\_compiler** tool, it is possible to automate the publishing of a website project. The tool is located in the .NET Framework folder. For .NET 2.0 through 3.5, this is usually C:\Windows\Microsoft.NET\Framework\v2.0.50727. For .NET 4, this is usually C:\Windows\Microsoft.NET\Framework\v4.0.30319. To run it, issue a command similar to the following (lines wrapped for display purposes):

[Copy](#)

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\asp  
-f -v /MyWebsite  
-p C:\Path\To\MyWebsite C:\Publish\MyWebsite
```

The **-f** option tells it to delete any existing content in the target folder (C:\Publish\MyWebsite in the example above). The **-v [path]** option tells it the virtual path of the website to compile (i.e. the path you use to load it in the web browser). The **-p [path]** option tells it the physical location of the website related to the virtual path to compile. Using a script, you could for example publish the website, build the help file, replace the *Web.config* in the published website with one that has had the **system.codedom** section removed, and delete the code provider assembly if it was copied locally. This is left as an exercise for the reader. The web deployment add-ins noted above support post-build events so they may provide better support for such tasks.

## ▪ Usage Notes and Known Issues

The following are some known issues and things to be aware of when using the custom code providers.

- If any part of the XML comments file path contains spaces, enclose it in " entities (i.e. `compilerOptions="/docpath:"C:\My Path With Spaces"";`).
- Do not use a relative path in the **/docpath** option. Always use a fully qualified path. When using the project's publish option,

the ASP.NET compiler always runs in the .NET Framework folder so your XML comments files will end up in a location relative to it rather than your project's folder if you do.

- You can specify any folder except one that is under the location to which you publish the website. The act of publishing the website clears the destination folder and, since the comments files are produced first, you will lose them if you place them under it.
- The code providers will delete any "App\_\*.xml" files from the specified folder on the first use. This prevents lots of files from accumulating in the comments file folder that contain duplicate information when not using fixed names. It also means that you should publish each website's comments files to a unique folder so that they are not lost if you are publishing and documenting multiple projects.
- Unless you are using fixed names when publishing, the assembly and comments filenames will change each time you publish the website. As such, use wildcards (App\_\*.dll and App\_\*.xml) to specify the files to use in your preferred documentation tool.
- When using the VB.NET custom code provider, the commonly imported namespaces such as `System` and `Microsoft.VisualBasic` are not included automatically for some unknown reason. The custom code provider will add a `/imports` command line option automatically for most of the common namespaces to work around the problem. If publishing fails due to undefined types when using the custom code providers, simply add a `/imports:[Namespace]` option to the `compilerOptions` attribute where "[Namespace]" is the missing namespace. You can generally identify namespaces that may cause problems by checking `Web.Config` for `<namespace>` elements in the `<pages>` configuration section. See below for a list of the common namespaces added automatically by the VB.NET custom code provider. Another workaround to this issue is to add `Imports` statements to the code for each of the required namespaces rather than relying on the automatically included references.
- Related to the item above, when the custom code providers are active in the `Web.Config` file, you may see lots of errors indicated by the red squiggly underline related to missing types while viewing VB.NET code. During development, you can comment out the custom code providers so that this does not happen. As noted, the other workaround is to add an `Imports`

statement to the code that includes the missing namespace that contains the types.

## How The Code Providers Work

The custom code providers are actually quite simple. Both the C# and VB.NET custom code providers are identical with the exception of their base class and an extra set of compiler options passed by the VB.NET version. The C# version is shown below.

Example Code Provider

Copy

```
public class CSharpCodeProviderWithDocs : CSharpCodeProvider
{
    public override CompilerResults CompileAssemblyFromDom(
        CompilerParameters options,
        params CodeCompileUnit[] compilationUnits)
    {
        CodeProviderHelper.ReplaceDocPathOption(options);
        return base.CompileAssemblyFromDom(options,
            compilationUnits);
    }

    public override CompilerResults CompileAssemblyFromFile(
        CompilerParameters options, params string[])
    {
        CodeProviderHelper.ReplaceDocPathOption(options);
        return base.CompileAssemblyFromFile(options);
    }

    public override CompilerResults CompileAssemblyFromSource(
        CompilerParameters options, params string[])
    {
        CodeProviderHelper.ReplaceDocPathOption(options);
        return base.CompileAssemblyFromSource(options);
    }
}
```

The derived version simply overrides three key methods and, prior to

calling the base implementation, calls the `CodeProviderHelper.ReplaceDocPathOption` method which takes care of updating the compiler options to produce a unique XML comments file for the assembly in the specified folder. It is shown below.

#### Replace Doc Path Option Method

Copy

```
internal static void ReplaceDocPathOption(
    CompilerParameters options,
    string[] additionalOptions)
{
    Match m;
    string docPath, docFile;

    // Replace /docpath with /doc
    if(!String.IsNullOrEmpty(options.CompilerOptions))
    {
        m = reDocPathOpt.Match(options.CompilerOptions);

        if(m.Success)
        {
            docPath = m.Groups[1].Value.Replace('
                String.Empty);

            if(!Directory.Exists(docPath))
            {
                Directory.CreateDirectory(docPath);
                docFilesPurged = true;
            }
            else
                if(!docFilesPurged)
                {
                    // Purge the comments files if
                    // on the first call.
                    foreach(string file in Directory.GetFiles(
                        docPath, "App_*.xml"))
                        File.Delete(file);

                    docFilesPurged = true;
                }
        }
    }
}
```

```

        }

        docFile = Path.Combine(docPath, Path.
            Path.ChangeExtension(options.Output
                ".xml")));

        options.CompilerOptions = String.Format(
            "{0} /doc:{1}\\"", reDocPathOpt.
            options.CompilerOptions, String.E
        }
    }

    // Append the other options
    if(additionalOptions != null && additionalOptions != "")
        options.CompilerOptions = String.Concat(
            options.CompilerOptions, " ", String.
            additionalOptions));
}

```

A regular expression is used to locate the `/docpath` option. If found, it extracts the folder name and either creates it if it does not exist or deletes all existing `App_*.xml` files in it the first time it is called. It then combines the path with the output assembly's name and a `.xml` extension and replaces the `/docpath` option with a `/doc` command line option. If any additional options are passed to the method, they are also appended to the command line options. The VB.NET code provider passes in an extra `/imports` command line option to specify the following default import namespaces to work around the issue noted earlier:

- System
- System.Collections
- System.Collections.Generic
- System.Collections.ObjectModel
- System.Configuration
- System.Data

- System.Web
- System.Web.Configuration
- System.Web.UI
- System.Web.UI.HtmlControls
- System.Web.UI.WebControls
- System.Web.Util
- System.Xml
- Microsoft.VisualBasic

## ◀ Conclusion

By using the custom code providers and the instructions above, you can now easily produce XML comments files for website projects in order to create help files for them.

## ◀ See Also

Reference

[CSharpCodeProviderWithDocs](#)

[VBCodeProviderWithDocs](#)

Other Resources

[Sandcastle Help File Builder](#)

---

[Copyright © 2008-2015, Eric Woodruff, All rights reserved](#)

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# Version History

The topics in this section describe the various changes made to the Web Code Providers library over the life of the project.

## ► Version History

Select a version below to see a description of its changes.

- [Version 1.1.0.0](#)
- [Version 1.0.0.0](#)

---

Copyright © 2008-2015, Eric Woodruff, All rights reserved

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# Version 1.1.0.0

Version 1.1.0.0 was released on February 3rd, 2009.

## ▪ Changes in This Release

- Added constructors to support `providerOption` elements in the configuration. These are required for .NET 3.5 support.

### Important

Be sure to update the version number to 1.1.0.0 in the `compiler` element in any `Web.config` files that use the custom code providers.

## ▪ See Also

[Other Resources](#)

[Version History](#)

---

[Copyright © 2008-2015, Eric Woodruff, All rights reserved](#)

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# Version 1.0.0.0

Version 1.0.0.0 was released on May 15th, 2008.

## ► Changes in This Release

- Initial release.

## ► See Also

[Other Resources](#)

[Version History](#)

---

Copyright © 2008-2015, Eric Woodruff, All rights reserved

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# EWSoftware.CodeDom Namespace

Custom code providers for website projects that can generate XML comments files for all code in the project.

## ► Classes

Class	Description
 <a href="#">CSharpCodeProviderWithDocs</a>	This is a custom C# code provider that is able to output an individual XML comments file for each unit that is compiled to a folder of your choice.
 <a href="#">VBCodeProviderWithDocs</a>	This is a custom VB.NET code provider that is able to output an individual XML comments file for each unit that is compiled to a folder of your choice.

---

Copyright © 2008-2015, Eric Woodruff, All rights reserved

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# CSharpCodeProviderWithDocs Class

This is a custom C# code provider that is able to output an individual XML comments file for each unit that is compiled to a folder of your choice.

## ► Inheritance Hierarchy

```
SystemObject SystemMarshalByRefObject
  System.ComponentModel.Component
    System.CodeDom.CompilerCodeDomProvider
      Microsoft.CSharpCSharpCodeProvider
        EWSoftware.CodeDomCSharpCodeProviderWithDocs
```

**Namespace:** [EWSoftware.CodeDom](#)

**Assembly:** EWSoftware.CodeDom (in EWSoftware.CodeDom.dll)

Version: 2015.6.7.0

## ► Syntax

C#    VB    C++    F#

Copy

```
public class CSharpCodeProviderWithDocs : CSharpC
```

The [CSharpCodeProviderWithDocs](#) type exposes the following members.

## ► Constructors

Name	Description
 <a href="#">CSharpCodeProviderWithDocs</a>	Default

construc



[CSharpCodeProviderWithDocs\(IDictionaryString, String\)](#)

This  
construc  
is passe  
provider  
options  
dictionai

[Top](#)

## Properties

	Name	Description
	<a href="#">CanRaiseEvents</a>	Gets a value indicating whether the component can raise an event. (Inherited from <a href="#">Component</a> .)
	<a href="#">Container</a>	Gets the <a href="#">.IContainer</a> that contains the <a href="#">Component</a> . (Inherited from <a href="#">Component</a> .)
	<a href="#">DesignMode</a>	Gets a value that indicates whether the <a href="#">Component</a> is currently in design mode. (Inherited from <a href="#">Component</a> .)
	<a href="#">Events</a>	Gets the list of event handlers that are attached to this <a href="#">Component</a> . (Inherited from <a href="#">Component</a> .)
	<a href="#">FileExtension</a>	Gets the file name extension to use when creating source code files. (Inherited from

[CSharpCodeProvider.\)](#)

	<a href="#">LanguageOptions</a>	Gets a language features identifier. (Inherited from <a href="#">CodeDomProvider</a> .)
	<a href="#">Site</a>	Gets or sets the <a href="#">ISite</a> of the <a href="#">Component</a> . (Inherited from <a href="#">Component</a> .)

[Top](#)

## Methods

Name	Description
 <a href="#">CompileAssemblyFromDom</a>	Compiles an assembly based on the code contained in the specified <code>CodeDomProvider</code> using the specified compiler settings. (Overrides <a href="#">CodeDomProviderCompiler.CompileCodeCompileUnit</a> .)
 <a href="#">CompileAssemblyFromFile</a>	Compiles an assembly from files, using the specified <code>CodeDomProvider</code> and <code>String</code> . (Overrides <a href="#">CodeDomProviderCompiler.CompileString</a> .)
 <a href="#">CompileAssemblyFromSource</a>	Compiles an assembly from source code, using the specified <code>CodeDomProvider</code> and <code>String</code> . (Overrides <a href="#">CodeDomProviderCompiler.CompileString</a> .)
 <a href="#">CreateCompiler</a>	<b>Obsolete.</b> Gets an instance of the <code>CSharpCodeProvider</code> .

(Inherited from [CSharpCode](#))

≡	<a href="#">CreateEscapedIdentifier</a>	Creates an escaped identifier. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">CreateGenerator</a>	<b>Obsolete.</b> Gets an instance of the <a href="#">CodeDomProvider</a> class. (Inherited from <a href="#">CSharpCode</a> )
≡	<a href="#">CreateGenerator(TextWriter)</a>	When overridden in a derived class, creates a code generator using the specified <a href="#">TextWriter</a> . (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">CreateGenerator(String)</a>	When overridden in a derived class, creates a code generator using the specified file name. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">CreateObjRef</a>	Creates an object that can be marshaled to another thread or process to generate a proxy used to access the object. (Inherited from <a href="#">MarshalByObject</a> )
≡	<a href="#">CreateParser</a>	<b>Obsolete.</b> When overridden in a derived class, creates a code parser. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">CreateValidIdentifier</a>	Creates a valid identifier. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">Dispose</a>	Releases all resources used by the <a href="#">CodeDomProvider</a> object. (Inherited from <a href="#">Component</a> )
≡	<a href="#">Dispose(Boolean)</a>	Releases the unmanaged resources used by the <a href="#">CodeDomProvider</a> object and optionally releases the managed resources. (Inherited from <a href="#">Component</a> )
≡	<a href="#">Equals</a>	Determines whether the current object is equal to another object of the same type. (Inherited from <a href="#">Object</a> .)

	<a href="#">Finalize</a>	Releases unmanaged resources used by the <a href="#">CodeDomProvider</a> before the <a href="#">CodeDomProvider</a> is garbage collected. (Inherited from <a href="#">Component</a> )
≡	<a href="#">GenerateCodeFromCompileUnit</a>	Generates code for the specified <a href="#">CodeDOM</a> compilation using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GenerateCodeFromExpression</a>	Generates code for the specified <a href="#">CodeDOM</a> expression using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GenerateCodeFromMember</a>	Generates code for the specified writer and code generator. (Inherited from <a href="#">CSharpCodeProvider</a> )
≡	<a href="#">GenerateCodeFromNamespace</a>	Generates code for the specified <a href="#">CodeDOM</a> namespace using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GenerateCodeFromStatement</a>	Generates code for the specified <a href="#">CodeDOM</a> statement using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GenerateCodeFromType</a>	Generates code for the specified <a href="#">CodeDOM</a> type declaration using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GetConverter</a>	Gets a <a href="#">TypeConverter</a> for the specified type. (Inherited from <a href="#">CSharpCodeProvider</a> )
≡	<a href="#">GetHashCode</a>	Serves as a hash function. (Inherited from <a href="#">Object</a> .)

≡	<a href="#">GetLifetimeService</a>	Retrieves the current lifetime policy for this instance. (Inherited from <a href="#">MarshalBase</a> )
≡	<a href="#">GetService</a>	Returns an object that represents the <a href="#">Component</a> or by its <a href="#">CodeDomObject</a> . (Inherited from <a href="#">Component</a> )
≡	<a href="#">GetType</a>	Gets the <a href="#">Type</a> of the current instance. (Inherited from <a href="#">Object</a> .)
≡	<a href="#">GetTypeOutput</a>	Gets the type indicated by the <a href="#">CodeDomObject</a> . (Inherited from <a href="#">CodeDomObject</a> )
≡	<a href="#">InitializeLifetimeService</a>	Obtains a lifetime service instance. (Inherited from <a href="#">MarshalBase</a> )
≡	<a href="#">IsValidIdentifier</a>	Returns a value that indicates whether the identifier for the current language is valid. (Inherited from <a href="#">CodeDomObject</a> )
≡	<a href="#">MemberwiseClone</a>	Creates a shallow copy of the current <a href="#">CodeDomObject</a> . (Inherited from <a href="#">Object</a> .)
≡	<a href="#">MemberwiseClone(Boolean)</a>	Creates a shallow copy of the current <a href="#">CodeDomObject</a> , using the specified <a href="#">Boolean</a> value to indicate whether the clone operation is a memberwise clone. (Inherited from <a href="#">MarshalBase</a> )
≡	<a href="#">Parse</a>	Compiles the code read from the <a href="#">CodeCompileUnit</a> . (Inherited from <a href="#">CodeDomObject</a> )
≡	<a href="#">Supports</a>	Returns a value indicating whether support is provided. (Inherited from <a href="#">CodeDomObject</a> )
≡	<a href="#">ToString</a>	Returns a <a href="#">String</a> containing a string representation of the current <a href="#">CodeDomObject</a> . This method should not be overridden. (Inherited from <a href="#">Object</a> )

(Inherited from [Component](#))

[Top](#)

## Events

	Name	Description
⚡	<a href="#">Disposed</a>	Occurs when the component is disposed by a call to the <a href="#">Dispose</a> method. (Inherited from <a href="#">Component</a> .)

[Top](#)

## Remarks

This changes the default behavior which overwrites the comments file on each invocation and dumps it into the temporary ASP.NET compilation folder.

A `/docpath: [path]` option should be added to the `compilerOptions` attribute in the `Web.config` file to specify the path to which the XML comments files will be written. The filenames will match the assembly names generated by the compiler.

## Examples

### Example Compiler Configuration

[Copy](#)

```
<configuration>
  <system.codedom>
    <compilers>
      <!-- For C# -->
      <compiler language="c#;cs;csharp" extensi
        compilerOptions="/docpath:C:\Publish\Do
        type="EWSoftware.CodeDom.CSharpCodePro
          EWSoftware.CodeDom, Version=1.0.0.0
          PublicKeyToken=d633d7d5b41cbb65" />
```

```
</compilers>
</system.codedom>
</configuration>
```

## ▲ See Also

Reference

[EWSOFTWARE.CODEDOM NAMESPACES](#)

---

[COPYRIGHT © 2008-2015, ERIC WOODRUFF, ALL RIGHTS RESERVED](#)

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# CSharpCodeProviderWithDocs Constructor

There are two overloads for the constructor

## ► Overload List

Name	Description
 <a href="#">CSharpCodeProviderWithDocs</a>	Default constructor
 <a href="#">CSharpCodeProviderWithDocs(IDictionary&lt;String, String&gt;)</a>	This constructor is passed provider options dictionary

[Top](#)

## ► See Also

[Reference](#)

[CSharpCodeProviderWithDocs Class](#)

[EWSoftware.CodeDom Namespace](#)

---

Copyright © 2008-2015, Eric Woodruff, All rights reserved

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# CSharpCodeProviderWithDocs Constructor

Default constructor

**Namespace:** [EWSoftware.CodeDom](#)

**Assembly:** EWSoftware.CodeDom (in EWSoftware.CodeDom.dll)

Version: 2015.6.7.0

## ► Syntax

[C#](#)   [VB](#)   [C++](#)   [F#](#)

[Copy](#)

```
public CSharpCodeProviderWithDocs()
```

## ► See Also

Reference

[CSharpCodeProviderWithDocs Class](#)

[CSharpCodeProviderWithDocs Overload](#)

[EWSoftware.CodeDom Namespace](#)

---

Copyright © 2008-2015, Eric Woodruff, All rights reserved

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# CSharpCodeProviderWithDocs

## Constructor (IDictionary<String, String>)

This constructor is passed a provider options dictionary

**Namespace:** [EWSoftware.CodeDom](#)

**Assembly:** EWSoftware.CodeDom (in EWSoftware.CodeDom.dll)

Version: 2015.6.7.0

### ► Syntax

[C#](#)   [VB](#)   [C++](#)   [F#](#)

[Copy](#)

```
public CSharpCodeProviderWithDocs(  
    IDictionary<string, string> providerOptions  
)
```

### Parameters

*providerOptions*

Type: [System.Collections.GenericIDictionary<String, String>](#)

The provider options

### ► See Also

Reference

[CSharpCodeProviderWithDocs Class](#)

[CSharpCodeProviderWithDocs Overload](#)

[EWSoftware.CodeDom Namespace](#)

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# CSharpCodeProviderWithDocs Properties

The [CSharpCodeProviderWithDocs](#) type exposes the following members.

## Properties

	Name	Description
	<a href="#">CanRaiseEvents</a>	Gets a value indicating whether the component can raise an event. (Inherited from <a href="#">Component</a> .)
	<a href="#">Container</a>	Gets the <a href="#">.IContainer</a> that contains the <a href="#">Component</a> . (Inherited from <a href="#">Component</a> .)
	<a href="#">DesignMode</a>	Gets a value that indicates whether the <a href="#">Component</a> is currently in design mode. (Inherited from <a href="#">Component</a> .)
	<a href="#">Events</a>	Gets the list of event handlers that are attached to this <a href="#">Component</a> . (Inherited from <a href="#">Component</a> .)
	<a href="#">FileExtension</a>	Gets the file name extension to use when creating source code files. (Inherited from <a href="#">CSharpCodeProvider</a> .)



[LanguageOptions](#) Gets a language features identifier.

(Inherited from [CodeDomProvider](#).)



[Site](#)

Gets or sets the [ISite](#) of the [Component](#).

(Inherited from [Component](#).)

---

[Top](#)

## ▲ See Also

[Reference](#)

[CSharpCodeProviderWithDocs Class](#)

[EWSoftware.CodeDom Namespace](#)

---

[Copyright © 2008-2015, Eric Woodruff, All rights reserved](#)

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# CSharpCodeProviderWithDocs Methods

The [CSharpCodeProviderWithDocs](#) type exposes the following members.

## ▪ Methods

Name	Description
<a href="#">CompileAssemblyFromDom</a>	Compiles an assembly based on the code contained in the specified file, using the specified compiler settings. (Overrides <a href="#">CodeDomProviderCompileAssemblyFromDom(String)</a> .)
<a href="#">CompileAssemblyFromFile</a>	Compiles an assembly from files, using the specified compiler settings. (Overrides <a href="#">CodeDomProviderCompileAssemblyFromFile(String)</a> .)
<a href="#">CompileAssemblyFromSource</a>	Compiles an assembly from source code, using the specified compiler settings. (Overrides <a href="#">CodeDomProviderCompileAssemblyFromSource(String)</a> .)
<a href="#">CreateCompiler</a>	<b>Obsolete.</b> Gets an instance of the <a href="#">CodeDomProvider</a> . (Inherited from <a href="#">CSharpCodeProvider</a> .)
<a href="#">CreateEscapedIdentifier</a>	Creates an escaped identifier.

(Inherited from [CodeDomProvider](#))

≡	<a href="#">CreateGenerator</a>	<b>Obsolete.</b> Gets an instance of the <a href="#">CodeDomProvider</a> class. (Inherited from <a href="#">CSharpCodeProvider</a> )
≡	<a href="#">CreateGenerator(TextWriter)</a>	When overridden in a derived class, creates a generator using the specified <a href="#">TextWriter</a> . (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">CreateGenerator(String)</a>	When overridden in a derived class, creates a generator using the specified file name. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">CreateObjRef</a>	Creates an object that can be marshaled to another application to generate a proxy used to access the object. (Inherited from <a href="#">MarshalByObject</a> )
≡	<a href="#">CreateParser</a>	<b>Obsolete.</b> When overridden in a derived class, creates a parser. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">CreateValidIdentifier</a>	Creates a valid identifier. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">Dispose</a>	Releases all resources used by the <a href="#">CodeDomProvider</a> class. (Inherited from <a href="#">Component</a> )
💡	<a href="#">Dispose(Boolean)</a>	Releases the unmanaged resources used by the <a href="#">CodeDomProvider</a> class and optionally releases the managed resources. (Inherited from <a href="#">Component</a> )
≡	<a href="#">Equals</a>	Determines whether the current <a href="#">CodeDomProvider</a> object is equal to a specified object. (Inherited from <a href="#">Object</a> .)
💡	<a href="#">Finalize</a>	Releases unmanaged resources used by the <a href="#">CodeDomProvider</a> class and performs other cleanup operations before the <a href="#">CodeDomProvider</a> object is destroyed. (Inherited from <a href="#">Component</a> )

≡	<a href="#">GenerateCodeFromCompileUnit</a>	Generates code for the specified (CodeDOM) compilation using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GenerateCodeFromExpression</a>	Generates code for the specified (CodeDOM) expression using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GenerateCodeFromMember</a>	Generates code for the specified writer and code generator. (Inherited from <a href="#">CSharpCodeProvider</a> )
≡	<a href="#">GenerateCodeFromNamespace</a>	Generates code for the specified (CodeDOM) namespace using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GenerateCodeFromStatement</a>	Generates code for the specified (CodeDOM) statement using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GenerateCodeFromType</a>	Generates code for the specified (CodeDOM) type declaration using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GetConverter</a>	Gets a <a href="#">TypeConverter</a> for the current type. (Inherited from <a href="#">CSharpCodeProvider</a> )
≡	<a href="#">GetHashCode</a>	Serves as a hash function. (Inherited from <a href="#">Object</a> .)
≡	<a href="#">GetLifetimeService</a>	Retrieves the current lifetime policy for this instance. (Inherited from <a href="#">MarshalByValueContainer</a> )

---

💡	<a href="#">GetService</a>	Returns an object that represents the <a href="#">Component</a> or by its <a href="#">Contract</a> . (Inherited from <a href="#">Component</a> )
💡	<a href="#">GetType</a>	Gets the <a href="#">Type</a> of the current instance. (Inherited from <a href="#">Object</a> .)
💡	<a href="#">GetTypeOutput</a>	Gets the type indicated by the <a href="#">CodeDomProvider</a> .
💡	<a href="#">InitializeLifetimeService</a>	Obtains a lifetime service for the current instance. (Inherited from <a href="#">MarshalByRefObject</a> )
💡	<a href="#">IsValidIdentifier</a>	Returns a value that indicates whether the identifier for the current language is valid.
💡	<a href="#">MemberwiseClone</a>	Creates a shallow copy of the current instance. (Inherited from <a href="#">Object</a> .)
💡	<a href="#">MemberwiseClone(Boolean)</a>	Creates a shallow copy of the current instance, using the specified Boolean value to determine whether to copy the fields of the current instance.
💡	<a href="#">Parse</a>	Compiles the code read from the <a href="#">CodeCompileUnit</a> . (Inherited from <a href="#">CodeDomObject</a> )
💡	<a href="#">Supports</a>	Returns a value indicating whether the current instance supports the specified feature.
💡	<a href="#">ToString</a>	Returns a <a href="#">String</a> containing a standard representation of the current instance. This method should not be overridden.

---

[Top](#)

## ▲ See Also

Reference

[CSharpCodeProviderWithDocs Class](#)

[EWSOFTWARE.CODEDOM Namespace](#)

---

Copyright © 2008-2015, Eric Woodruff, All rights reserved

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# CSharpCodeProviderWithDocsCom Method

Compiles an assembly based on the [System.CodeDom](#) trees contained in the specified array of [CodeCompileUnit](#) objects, using the specified compiler settings.

**Namespace:** [EWSoftware.CodeDom](#)

**Assembly:** EWSoftware.CodeDom (in EWSoftware.CodeDom.dll)

Version: 2015.6.7.0

## ► Syntax

[C#](#)   [VB](#)   [C++](#)   [F#](#)

[Copy](#)

```
public override CompilerResults CompileAssemblyFromCompilerParameters options,
params CodeCompileUnit[] compilationUnits
)
```

### Parameters

*options*

Type: [System.CodeDom.Compiler.CompilerParameters](#)

A [CompilerParameters](#) object that indicates the settings for the compilation.

*compilationUnits*

Type: [System.CodeDom.CodeCompileUnit](#)

An array of type [CodeCompileUnit](#) that indicates the code to compile.

### Return Value

Type: [CompilerResults](#)

A [CompilerResults](#) object that indicates the results of the compilation.

## Exceptions

Exception	Condition
<a href="#">NotImplementedException</a>	Neither this method nor the <a href="#">CreateCompiler</a> method is overridden in a derived class.

## See Also

### Reference

[CSharpCodeProviderWithDocs Class](#)  
[EWSoftware.CodeDom Namespace](#)

---

Copyright © 2008-2015, Eric Woodruff, All rights reserved

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# CSharpCodeProviderWithDocsCom Method

Compiles an assembly from the source code contained in the specified files, using the specified compiler settings.

**Namespace:** [EWSOFTWARE.CODEDOM](#)

**Assembly:** EWSOFTWARE.CODEDOM (in EWSOFTWARE.CODEDOM.dll)

Version: 2015.6.7.0

## ► Syntax

[C#](#)   [VB](#)   [C++](#)   [F#](#)

[Copy](#)

```
public override CompilerResults CompileAssemblyFrom
    CompilerParameters options,
    params string[] fileNames
)
```

## Parameters

*options*

Type: [System.CodeDom.CompilerCompilerParameters](#)

A [CompilerParameters](#) object that indicates the settings for the compilation.

*fileNames*

Type: [SystemString](#)

An array of the names of the files to compile.

## Return Value

Type: [CompilerResults](#)

A [CompilerResults](#) object that indicates the results of compilation.

## ◀ Exceptions

Exception	Condition
<a href="#">NotImplementedException</a>	Neither this method nor the <a href="#">CreateCompiler</a> method is overridden in a derived class.

## ◀ See Also

### Reference

[CSharpCodeProviderWithDocs Class](#)

[EWSOFTWARE.CODEDOM Namespace](#)

---

Copyright © 2008-2015, Eric Woodruff, All rights reserved

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# CSharpCodeProviderWithDocsCom Method

Compiles an assembly from the specified array of strings containing source code, using the specified compiler settings.

**Namespace:** [EWSoftware.CodeDom](#)

**Assembly:** EWSoftware.CodeDom (in EWSoftware.CodeDom.dll)

Version: 2015.6.7.0

## ► Syntax

[C#](#)   [VB](#)   [C++](#)   [F#](#)

[Copy](#)

```
public override CompilerResults CompileAssemblyFr
    CompilerParameters options,
    params string[] sources
)
```

## Parameters

*options*

Type: [System.CodeDom.CompilerCompilerParameters](#)

A [CompilerParameters](#) object that indicates the compiler settings for this compilation.

*sources*

Type: [SystemString](#)

An array of source code strings to compile.

## Return Value

Type: [CompilerResults](#)

A [CompilerResults](#) object that indicates the results of compilation.

## ◀ Exceptions

Exception	Condition
<a href="#">NotImplementedException</a>	Neither this method nor the <a href="#">CreateCompiler</a> method is overridden in a derived class.

## ◀ See Also

### Reference

[CSharpCodeProviderWithDocs Class](#)

[EWSOFTWARE.CODEDOM Namespace](#)

---

Copyright © 2008-2015, Eric Woodruff, All rights reserved

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# CSharpCodeProviderWithDocs Events

The [CSharpCodeProviderWithDocs](#) type exposes the following members.

## ▪ Events

	Name	Description
	<a href="#">Disposed</a>	Occurs when the component is disposed by a call to the <a href="#">Dispose</a> method. (Inherited from <a href="#">Component</a> .)

[Top](#)

## ▪ See Also

### Reference

[CSharpCodeProviderWithDocs Class](#)  
[EWSoftware.CodeDom Namespace](#)

---

Copyright © 2008-2015, Eric Woodruff, All rights reserved

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# VBCODEPROVIDERWITHDOCS Class

This is a custom VB.NET code provider that is able to output an individual XML comments file for each unit that is compiled to a folder of your choice.

## ► Inheritance Hierarchy

```
SystemObject SystemMarshalByRefObject
  System.ComponentModel.Component
    System.CodeDom.CompilerCodeDomProvider
      Microsoft.VisualBasic.VBCodeProvider
        EWSoftware.CodeDomVBCODEProviderWithDocs
```

**Namespace:** [EWSoftware.CodeDom](#)

**Assembly:** EWSoftware.CodeDom (in EWSoftware.CodeDom.dll)

Version: 2015.6.7.0

## ► Syntax

C#    VB    C++    F#

[Copy](#)

```
public class VBCODEProviderWithDocs : VBCodeProv
```

The [VBCODEProviderWithDocs](#) type exposes the following members.

## ► Constructors

Name	Description
<a href="#">VBCODEProviderWithDocs</a>	Default constructor
<a href="#">VBCODEProviderWithDocs(IDictionaryString, This)</a>	

`String)` constructor is passed a provider options dictionary

[Top](#)

## Properties

Name	Description
 <a href="#">CanRaiseEvents</a>	Gets a value indicating whether the component can raise an event. (Inherited from <a href="#">Component</a> .)
 <a href="#">Container</a>	Gets the <a href="#">IContainer</a> that contains the <a href="#">Component</a> . (Inherited from <a href="#">Component</a> .)
 <a href="#">DesignMode</a>	Gets a value that indicates whether the <a href="#">Component</a> is currently in design mode. (Inherited from <a href="#">Component</a> .)
 <a href="#">Events</a>	Gets the list of event handlers that are attached to this <a href="#">Component</a> . (Inherited from <a href="#">Component</a> .)
 <a href="#">FileExtension</a>	Gets the file name extension to use when creating source code files. (Inherited from <a href="#">VBCODEProvider</a> .)
 <a href="#">LanguageOptions</a>	Gets a language features

identifier.  
(Inherited from [VBCodeProvider](#).)



### Site

Gets or sets the [ISite](#) of the [Component](#).  
(Inherited from [Component](#).)

[Top](#)

## Methods

Name	Description
<a href="#">CompileAssemblyFromDom</a>	Compiles an assembly based on the code contained in the specified <code>CodeDomProvider</code> , using the specified compiler settings. (Overrides <a href="#">CodeDomProviderCompiler.CompileAssemblyFromDom</a> .)
<a href="#">CompileAssemblyFromFile</a>	Compiles an assembly from files, using the specified <code>CodeDomProvider</code> and <code>String</code> . (Overrides <a href="#">CodeDomProviderCompiler.CompileAssemblyFromFile</a> .)
<a href="#">CompileAssemblyFromSource</a>	Compiles an assembly from source code, using the specified <code>CodeDomProvider</code> and <code>String</code> . (Overrides <a href="#">CodeDomProviderCompiler.CompileAssemblyFromSource</a> .)
<a href="#">CreateCompiler</a>	<b>Obsolete.</b> Gets an instance of the <code>CodeDomProvider</code> . (Inherited from <a href="#">VBCodeProvider</a> .)
<a href="#">CreateEscapedIdentifier</a>	Creates an escaped identifier.

(Inherited from [CodeDomProvider](#))

≡	<a href="#">CreateGenerator</a>	<b>Obsolete.</b> Gets an instance of the <a href="#">CodeGenerator</a> class. (Inherited from <a href="#">VBCodeProvider</a> )
≡	<a href="#">CreateGenerator(TextWriter)</a>	When overridden in a derived class, creates a generator using the specified <a href="#">TextWriter</a> . (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">CreateGenerator(String)</a>	When overridden in a derived class, creates a generator using the specified file name. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">CreateObjRef</a>	Creates an object that can be marshaled to another application to generate a proxy used to access the object. (Inherited from <a href="#">MarshalByObject</a> )
≡	<a href="#">CreateParser</a>	<b>Obsolete.</b> When overridden in a derived class, creates a parser. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">CreateValidIdentifier</a>	Creates a valid identifier. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">Dispose</a>	Releases all resources used by the <a href="#">CodeDomProvider</a> . (Inherited from <a href="#">Component</a> )
≡	<a href="#">Dispose(Boolean)</a>	Releases the unmanaged resources used by the <a href="#">CodeDomProvider</a> , optionally releases the managed resources. (Inherited from <a href="#">Component</a> )
≡	<a href="#">Equals</a>	Determines whether the current <a href="#">CodeDomProvider</a> is equal to a specified object. (Inherited from <a href="#">Object</a> .)
≡	<a href="#">Finalize</a>	Releases unmanaged resources used by the <a href="#">CodeDomProvider</a> and performs other cleanup operations before the <a href="#">CodeDomProvider</a> is reclaimed by the garbage collector. (Inherited from <a href="#">Component</a> )

≡	<a href="#">GenerateCodeFromCompileUnit</a>	Generates code for the specified (CodeDOM) compilation using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GenerateCodeFromExpression</a>	Generates code for the specified (CodeDOM) expression using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GenerateCodeFromMember</a>	Generates code for the specified writer and code generator. (Inherited from <a href="#">VBCodemProvider</a> )
≡	<a href="#">GenerateCodeFromNamespace</a>	Generates code for the specified (CodeDOM) namespace using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GenerateCodeFromStatement</a>	Generates code for the specified (CodeDOM) statement using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GenerateCodeFromType</a>	Generates code for the specified (CodeDOM) type declaration using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GetConverter</a>	Gets a <a href="#">TypeConverter</a> for the current type. (Inherited from <a href="#">VBCodemProvider</a> )
≡	<a href="#">GetHashCode</a>	Serves as a hash function. (Inherited from <a href="#">Object</a> .)
≡	<a href="#">GetLifetimeService</a>	Retrieves the current lifetime policy for this instance. (Inherited from <a href="#">MarshalByRefObject</a> )

---

💡	<a href="#">GetService</a>	Returns an object that represents the <a href="#">Component</a> or by its <a href="#">Contract</a> . (Inherited from <a href="#">Component</a> )
💡	<a href="#">GetType</a>	Gets the <a href="#">Type</a> of the current instance. (Inherited from <a href="#">Object</a> .)
💡	<a href="#">GetTypeOutput</a>	Gets the type indicated by the <a href="#">CodeDomProvider</a> .
💡	<a href="#">InitializeLifetimeService</a>	Obtains a lifetime service for the current component instance. (Inherited from <a href="#">MarshalByRefObject</a> )
💡	<a href="#">IsValidIdentifier</a>	Returns a value that indicates whether the identifier for the current language is valid.
💡	<a href="#">MemberwiseClone</a>	Creates a shallow copy of the current instance. (Inherited from <a href="#">Object</a> .)
💡	<a href="#">MemberwiseClone(Boolean)</a>	Creates a shallow copy of the current instance. (Inherited from <a href="#">MarshalByRefObject</a> )
💡	<a href="#">Parse</a>	Compiles the code read from the <a href="#">CodeCompileUnit</a> . (Inherited from <a href="#">CodeDomObject</a> )
💡	<a href="#">Supports</a>	Returns a value indicating whether the current component supports the provided feature.
💡	<a href="#">ToString</a>	Returns a <a href="#">String</a> containing a standard representation of the object. This method should not be overridden. (Inherited from <a href="#">Component</a> )

---

[Top](#)

## Events

Name	Description
⚡ <a href="#">Disposed</a>	Occurs when the component is disposed by a call to the <a href="#">Dispose</a> method. (Inherited from <a href="#">Component</a> .)

[Top](#)

## Remarks

This changes the default behavior which overwrites the comments file on each invocation and/or dumps them into the temporary ASP.NET compilation folder.

A `/docpath:[path]` option should be added to the `compilerOptions` attribute in the `Web.config` file to specify the path to which the XML comments files will be written. The filenames will match the assembly names generated by the compiler.

## Examples

### Example Compiler Configuration

[Copy](#)

```
<configuration>
  <system.codedom>
    <compilers>
      <!-- For VB.NET -->
      <compiler language="vb;vbs;visualbasic;vt"
                 compilerOptions="/docpath:C:\Publish\Doc
                 type="EWSoftware.CodeDom.VBCodeProvider
                               EWSoftware.CodeDom, Version=1.0.0.0
                               PublicKeyToken=d633d7d5b41cbb65" />
    </compilers>
  </system.codedom>
</configuration>
```

## ► See Also

Reference

[EWSOFTWARE.CODEDOM NAMESPACE](#)

---

Copyright © 2008-2015, Eric Woodruff, All rights reserved

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# VBCODEPROVIDERWITHDOCS Constructor

There are two overloads for the constructor

## ► Overload List

Name	Description
 <a href="#">VBCODEPROVIDERWITHDOCS</a>	Default constructor
 <a href="#">VBCODEPROVIDERWITHDOCS(IDICTIONARYSTRING, String)</a>	This constructor is passed a provider options dictionary

[Top](#)

## ► See Also

[Reference](#)

[VBCODEPROVIDERWITHDOCS Class](#)

[EWSOFTWARE.CODEDOM Namespace](#)

---

Copyright © 2008-2015, Eric Woodruff, All rights reserved

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# VBCODEProviderWithDocs Constructor

Default constructor

**Namespace:** [EWSoftware.CodeDom](#)

**Assembly:** EWSoftware.CodeDom (in EWSoftware.CodeDom.dll)

Version: 2015.6.7.0

## ► Syntax

[C#](#)   [VB](#)   [C++](#)   [F#](#)

[Copy](#)

```
public VBCODEProviderWithDocs()
```

## ► See Also

### Reference

[VBCODEProviderWithDocs Class](#)

[VBCODEProviderWithDocs Overload](#)

[EWSoftware.CodeDom Namespace](#)

---

Copyright © 2008-2015, Eric Woodruff, All rights reserved

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# VBCODEPROVIDERWITHDOCS Constructor (IDictionaryString, String)

This constructor is passed a provider options dictionary

**Namespace:** [EWSoftware.CodeDom](#)

**Assembly:** EWSoftware.CodeDom (in EWSoftware.CodeDom.dll)

Version: 2015.6.7.0

## ► Syntax

[C#](#)   [VB](#)   [C++](#)   [F#](#)

[Copy](#)

```
public VBCODEPROVIDERWITHDOCS(  
    IDictionary<string, string> providerOptions  
)
```

## Parameters

*providerOptions*

Type: [System.Collections.GenericIDictionaryString, String](#)

The provider options

## ► See Also

### Reference

[VBCODEPROVIDERWITHDOCS Class](#)

[VBCODEPROVIDERWITHDOCS Overload](#)

[EWSoftware.CodeDom Namespace](#)

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# VBCodeProviderWithDocs Properties

The [VBCodeProviderWithDocs](#) type exposes the following members.

## Properties

	Name	Description
	<a href="#">CanRaiseEvents</a>	Gets a value indicating whether the component can raise an event. (Inherited from <a href="#">Component</a> .)
	<a href="#">Container</a>	Gets the <a href="#">.IContainer</a> that contains the <a href="#">Component</a> . (Inherited from <a href="#">Component</a> .)
	<a href="#">DesignMode</a>	Gets a value that indicates whether the <a href="#">Component</a> is currently in design mode. (Inherited from <a href="#">Component</a> .)
	<a href="#">Events</a>	Gets the list of event handlers that are attached to this <a href="#">Component</a> . (Inherited from <a href="#">Component</a> .)
	<a href="#">FileExtension</a>	Gets the file name extension to use when creating source code files. (Inherited from <a href="#">VBCodeProvider</a> .)



[LanguageOptions](#) Gets a language features identifier.  
(Inherited from [VBCodeProvider](#).)



[Site](#) Gets or sets the [ISite](#) of the [Component](#).  
(Inherited from [Component](#).)

---

[Top](#)

## ▲ See Also

### Reference

[VBCodeProviderWithDocs Class](#)  
[EWSOFTWARE.CODEDOM Namespace](#)

---

Copyright © 2008-2015, Eric Woodruff, All rights reserved

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# VBCODEPROVIDERWITHDOCS Methods

The [VBCODEPROVIDERWITHDOCS](#) type exposes the following members.

## ▪ Methods

Name	Description
<a href="#">CompileAssemblyFromDom</a>	Compiles an assembly based on the code contained in the specified CodeDomProvider object, using the specified compiler settings. (Overrides <a href="#">CodeDomProvider.CompileAssemblyFromDom(CodeCompileUnit)</a> .)
<a href="#">CompileAssemblyFromFile</a>	Compiles an assembly from files, using the specified compiler settings. (Overrides <a href="#">CodeDomProvider.CompileAssemblyFromFile(String)</a> .)
<a href="#">CompileAssemblyFromSource</a>	Compiles an assembly from source code, using the specified compiler settings. (Overrides <a href="#">CodeDomProvider.CompileAssemblyFromSource(String)</a> .)
<a href="#">CreateCompiler</a>	<b>Obsolete.</b> Gets an instance of the <a href="#">VBCODEP</a> class. (Inherited from <a href="#">VBCODEP</a> .)
<a href="#">CreateEscapedIdentifier</a>	Creates an escaped identifier.

(Inherited from [CodeDomProvider](#))

≡	<a href="#">CreateGenerator</a>	<b>Obsolete.</b> Gets an instance of the <a href="#">CodeDomProvider</a> class. (Inherited from <a href="#">VBCodeProvider</a> )
≡	<a href="#">CreateGenerator(TextWriter)</a>	When overridden in a derived class, creates a generator using the specified <a href="#">TextWriter</a> . (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">CreateGenerator(String)</a>	When overridden in a derived class, creates a generator using the specified file name. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">CreateObjRef</a>	Creates an object that can be marshaled to another application to generate a proxy used to access the object. (Inherited from <a href="#">MarshalByObject</a> )
≡	<a href="#">CreateParser</a>	<b>Obsolete.</b> When overridden in a derived class, creates a parser. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">CreateValidIdentifier</a>	Creates a valid identifier. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">Dispose</a>	Releases all resources used by the <a href="#">CodeDomProvider</a> class. (Inherited from <a href="#">Component</a> )
≡	<a href="#">Dispose(Boolean)</a>	Releases the unmanaged resources used by the <a href="#">CodeDomProvider</a> class and optionally releases the managed resources. (Inherited from <a href="#">Component</a> )
≡	<a href="#">Equals</a>	Determines whether the current <a href="#">CodeDomProvider</a> object is equal to a specified object. (Inherited from <a href="#">Object</a> .)
≡	<a href="#">Finalize</a>	Releases unmanaged resources used by the <a href="#">CodeDomProvider</a> class and performs other cleanup operations before the <a href="#">CodeDomProvider</a> object is destroyed. (Inherited from <a href="#">Component</a> )

≡	<a href="#">GenerateCodeFromCompileUnit</a>	Generates code for the specified (CodeDOM) compilation using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GenerateCodeFromExpression</a>	Generates code for the specified (CodeDOM) expression using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GenerateCodeFromMember</a>	Generates code for the specified writer and code generator. (Inherited from <a href="#">VBCodemProvider</a> )
≡	<a href="#">GenerateCodeFromNamespace</a>	Generates code for the specified (CodeDOM) namespace using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GenerateCodeFromStatement</a>	Generates code for the specified (CodeDOM) statement using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GenerateCodeFromType</a>	Generates code for the specified (CodeDOM) type declaration using the specified options. (Inherited from <a href="#">CodeDomProvider</a> )
≡	<a href="#">GetConverter</a>	Gets a <a href="#">TypeConverter</a> for the current type. (Inherited from <a href="#">VBCodemProvider</a> )
≡	<a href="#">GetHashCode</a>	Serves as a hash function. (Inherited from <a href="#">Object</a> .)
≡	<a href="#">GetLifetimeService</a>	Retrieves the current lifetime policy for this instance. (Inherited from <a href="#">MarshalByValueContainer</a> )

---

💡	<a href="#">GetService</a>	Returns an object that represents the <a href="#">Component</a> or by its <a href="#">Contract</a> . (Inherited from <a href="#">Component</a> )
💡	<a href="#">GetType</a>	Gets the <a href="#">Type</a> of the current instance. (Inherited from <a href="#">Object</a> .)
💡	<a href="#">GetTypeOutput</a>	Gets the type indicated by the <a href="#">CodeDomProvider</a> .
💡	<a href="#">InitializeLifetimeService</a>	Obtains a lifetime service for the current component instance. (Inherited from <a href="#">MarshalByRefObject</a> )
💡	<a href="#">IsValidIdentifier</a>	Returns a value that indicates whether the identifier for the current language is valid.
💡	<a href="#">MemberwiseClone</a>	Creates a shallow copy of the current <a href="#">Object</a> . (Inherited from <a href="#">Object</a> .)
💡	<a href="#">MemberwiseClone(Boolean)</a>	Creates a shallow copy of the current <a href="#">Object</a> , using the specified <a href="#">Boolean</a> value to indicate whether the copy is a memberwise or a deep clone.
💡	<a href="#">Parse</a>	Compiles the code read from the <a href="#">CodeCompileUnit</a> . (Inherited from <a href="#">CodeDomObject</a> )
💡	<a href="#">Supports</a>	Returns a value indicating whether the current component supports the provided feature.
💡	<a href="#">ToString</a>	Returns a <a href="#">String</a> containing a standard representation of the current component. This method should not be overridden.

---

[Top](#)

## ▲ See Also

### Reference

[VBCodeProviderWithDocs Class](#)  
[EWSOFTWARE.CODEDOM Namespace](#)

---

Copyright © 2008-2015, Eric Woodruff, All rights reserved

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# VBCODEPROVIDERWITHDOCSCOMPILEA METHOD

Compiles an assembly based on the [System.CodeDom](#) trees contained in the specified array of [CodeCompileUnit](#) objects, using the specified compiler settings.

**Namespace:** [EWSoftware.CodeDom](#)

**Assembly:** EWSoftware.CodeDom (in EWSoftware.CodeDom.dll)

Version: 2015.6.7.0

## ► Syntax

C#    VB    C++    F#

Copy

```
public override CompilerResults CompileAssemblyFromCompilerParameters(CompilerParameters options,
    params CodeCompileUnit[] compilationUnits)
```

### Parameters

*options*

Type: [System.CodeDom.CompilerCompilerParameters](#)

A [CompilerParameters](#) object that indicates the settings for the compilation.

*compilationUnits*

Type: [System.CodeDomCodeCompileUnit](#)

An array of type [CodeCompileUnit](#) that indicates the code to compile.

### Return Value

Type: [CompilerResults](#)

A [CompilerResults](#) object that indicates the results of the compilation.

## Exceptions

Exception	Condition
<a href="#">NotImplementedException</a>	Neither this method nor the <a href="#">CreateCompiler</a> method is overridden in a derived class.

## See Also

### Reference

[VBCodeProviderWithDocs Class](#)  
[EWSoftware.CodeDom Namespace](#)

---

Copyright © 2008-2015, Eric Woodruff, All rights reserved

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# VBCODEPROVIDERWITHDOCSCOMPILEA METHOD

Compiles an assembly from the source code contained in the specified files, using the specified compiler settings.

**Namespace:** EWSOFTWARE.CODEDOM

**Assembly:** EWSOFTWARE.CODEDOM (in EWSOFTWARE.CODEDOM.dll)

Version: 2015.6.7.0

## ► Syntax

C#    VB    C++    F#

Copy

```
public override CompilerResults CompileAssemblyFr
    CompilerParameters options,
    params string[] fileNames
)
```

## Parameters

*options*

Type: [System.CodeDom.CompilerCompilerParameters](#)

A [CompilerParameters](#) object that indicates the settings for the compilation.

*fileNames*

Type: [SystemString](#)

An array of the names of the files to compile.

## Return Value

Type: [CompilerResults](#)

A [CompilerResults](#) object that indicates the results of compilation.

## ▪ Exceptions

Exception	Condition
<a href="#">NotImplementedException</a>	Neither this method nor the <a href="#">CreateCompiler</a> method is overridden in a derived class.

## ▪ See Also

### Reference

[VBCodeProviderWithDocs Class](#)

[EWSOFTWARE.CODEDOM Namespace](#)

---

Copyright © 2008-2015, Eric Woodruff, All rights reserved

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# VBCODEPROVIDERWITHDOCSCOMPILEA METHOD

Compiles an assembly from the specified array of strings containing source code, using the specified compiler settings.

**Namespace:** [EWSoftware.CodeDom](#)

**Assembly:** EWSoftware.CodeDom (in EWSoftware.CodeDom.dll)

Version: 2015.6.7.0

## ► Syntax

C#    VB    C++    F#

[Copy](#)

```
public override CompilerResults CompileAssemblyFr
    CompilerParameters options,
    params string[] sources
)
```

### Parameters

*options*

Type: [System.CodeDom.CompilerCompilerParameters](#)

A [CompilerParameters](#) object that indicates the compiler settings for this compilation.

*sources*

Type: [SystemString](#)

An array of source code strings to compile.

### Return Value

Type: [CompilerResults](#)

A [CompilerResults](#) object that indicates the results of compilation.

## ▪ Exceptions

Exception	Condition
<a href="#">NotImplementedException</a>	Neither this method nor the <a href="#">CreateCompiler</a> method is overridden in a derived class.

## ▪ See Also

### Reference

[VBCodeProviderWithDocs Class](#)  
[EWSOFTWARE.CODEDOM Namespace](#)

---

Copyright © 2008-2015, Eric Woodruff, All rights reserved

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)

# VBCODEProviderWithDocs Events

The [VBCODEProviderWithDocs](#) type exposes the following members.

## Events

	Name	Description
	<a href="#">Disposed</a>	Occurs when the component is disposed by a call to the <a href="#">Dispose</a> method. (Inherited from <a href="#">Component</a> .)

[Top](#)

## See Also

### Reference

[VBCODEProviderWithDocs Class](#)

[EWSoftware.CodeDom Namespace](#)

---

[Copyright © 2008-2015, Eric Woodruff, All rights reserved](#)

Send comments on this topic to [Eric@EWoodruff.us](mailto:Eric@EWoodruff.us)