
ComponentOne

Tabs for ASP.NET Wijmo

Copyright © 2012 ComponentOne LLC. All rights reserved.

Corporate Headquarters

ComponentOne LLC

201 South Highland Avenue
3rd Floor
Pittsburgh, PA 15206 • USA

Internet: info@ComponentOne.com

Web site: <http://www.componentone.com>

Sales

E-mail: sales@componentone.com

Telephone: 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

Trademarks

The ComponentOne product name is a trademark and ComponentOne is a registered trademark of ComponentOne LLC. All other trademarks used herein are the properties of their respective owners.

Warranty

ComponentOne warrants that the original CD (or diskettes) are free from defects in material and workmanship, assuming normal use, for a period of 90 days from the date of purchase. If a defect occurs during this time, you may return the defective CD (or disk) to ComponentOne, along with a dated proof of purchase, and ComponentOne will replace it at no charge. After 90 days, you can obtain a replacement for a defective CD (or disk) by sending it and a check for \$25 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original CD (or disks) set forth here, ComponentOne makes no other warranties, express or implied. Every attempt has been made to ensure that the information contained in this manual is correct as of the time it was written. We are not responsible for any errors or omissions. ComponentOne's liability is limited to the amount you paid for the product. ComponentOne is not liable for any special, consequential, or other damages for any reason.

Copying and Distribution

While you are welcome to make backup copies of the software for your own use and protection, you are not permitted to make copies for the use of anyone else. We put a lot of time and effort into creating this product, and we appreciate your support in seeing that it is used by licensed users only.

This manual was produced using [ComponentOne Doc-To-Help™](#).

Table of Contents

ComponentOne Tabs for ASP.NET Wijmo Overview	1
Installing Studio for ASP.NET Wijmo	1
Studio for ASP.NET Wijmo Setup Files	1
System Requirements	2
Uninstalling Studio for ASP.NET Wijmo	2
Deploying your Application in a Medium Trust Environment	2
End-User License Agreement	6
Licensing FAQs	6
What is Licensing?	6
How does Licensing Work?	6
Common Scenarios	7
Troubleshooting	9
Technical Support	10
Redistributable Files	11
About This Documentation	11
Namespaces	12
Creating an ASP.NET Project	13
Adding the Studio for ASP.NET Wijmo Components to a Project	14
ComponentOne ASP.NET Tabs for ASP.NET Wijmo Migration Guide	15
Prerequisites	15
Getting Started	15
Migration details	15
Key Features	17
Wijmo Top Tips	18
Tabs for ASP.NET Wijmo Quick Start	18
Step 1 of 3: Adding C1Tabs to the Page	18
Step 2 of 3: Working with the C1Tabs Designer Form	18
Step 3 of 3: Adding Content to the C1Tabs Control	19
C1Tabs Design-Time Support	20
C1Tabs Smart Tag	20

C1Tabs Context Menu	21
C1Tabs Designer Form	22
C1Tabs Elements	26
C1Tabs Themes	26
Tabstrip Direction	29
C1Tabs Behavior	30
Keyboard Access	30
ToolTips	30
Selected Index	31
Tabs for ASP.NET Wijmo Task-Based Help	31
Working with Themes	31
Using a Built-In Theme	31
Using a Custom Theme	32
Adding Tab Pages to the C1Tabs Control	34
Creating a C1Tabs Control in Code	35
Adding and Manipulating Tab Page Content	37
Adding Controls to C1Tabs	37
Adding Text to a C1Tabs Tab Page	38
Changing the Alignment	39
Changing the Selected Index	40
Tabs for ASP.NET Wijmo Client-Side Reference	41
Using the Wijmo CDN	41

ComponentOne Tabs for ASP.NET Wijmo Overview

Easily organize and navigate Web content with **ComponentOne Tabs™** for **ASP.NET Wijmo**. Host full pages of content on each tab.

For a list of the latest features added to **ComponentOne Studio for ASP.NET Wijmo**, visit [What's New in Studio for ASP.NET Wijmo](#).



Getting Started

- [Quick Start](#) (page 18)
- [Design-Time Support](#) (page 20)
- [Task-Based Help](#) (page 31)

Installing Studio for ASP.NET Wijmo

The following sections provide helpful information on installing **ComponentOne Studio for ASP.NET Wijmo**:

Studio for ASP.NET Wijmo Setup Files

The **ComponentOne Studio for ASP.NET Wijmo** installation program will create the following directory: C:\Program Files\ComponentOne\Studio for ASP.NET Wijmo. This directory contains the following subdirectories:

Bin	Contains copies of all binaries (DLLs, Exes) in the ComponentOne Visual Studio ASP.NET Wijmo package.
Wijmo	Contains files (at least a readme.txt) related to the product.

The **ComponentOne Studio for ASP.NET Wijmo Help Setup** program installs integrated Microsoft Help Viewer help to the C:\Program Files\ComponentOne\Studio for ASP.NET Wijmo directory in the following folder:

HelpViewer	Contains Microsoft Help Viewer Visual Studio 2010 integrated documentation for all Studio components.
-------------------	---

Samples

Samples for the product are installed in the **ComponentOne Samples** folder by default. The path of the **ComponentOne Samples** directory is slightly different on Windows XP and Windows 7/Vista machines:

Windows XP path: C:\Documents and Settings\\My Documents\ComponentOne Samples

Windows 7/Vista path: C:\Users\\Documents\ComponentOne Samples

The **ComponentOne Samples** folder contains the following subdirectories:

Common	Contains support and data files that are used by many of the demo programs.
---------------	---

Studio for ASP.NET Wijmo Contains a readme.txt file and the folders that make up the Control Explorer and other samples.

Samples can be accessed from the **ComponentOne Sample Explorer**. To view samples, on your desktop, click the **Start** button and then click **All Programs | ComponentOne | Studio for ASP.NET Wijmo | Control Explorer**.

System Requirements

System requirements for **ComponentOne Studio for ASP.NET Wijmo** components include the following:

- Operating Systems:** Windows Server® 2003
Windows Server 2008
Windows XP SP2
Windows Vista™
Windows 7
- Web Server:** Microsoft Internet Information Services (IIS) 6.0 or later
- Environments:** .NET Framework 3.0 or later
Visual Studio 2008 or later
Internet Explorer 6.0 or later
Firefox® 2.0 or later
Safari® 2.0 or later

Uninstalling Studio for ASP.NET Wijmo

To uninstall **Studio for ASP.NET Wijmo**:

1. Open the **Control Panel** and select the **Add or Remove Programs (Programs and Features in Vista/Windows 7)**.
2. Select **ComponentOne Studio for ASP.NET Wijmo** and click the **Remove** button.
3. Click **Yes** to remove the program.

To uninstall **Studio for ASP.NET Wijmo** integrated help:

1. Open the Control Panel and select **Add or Remove Programs** (Programs and Features in Windows 7/Vista).
2. Select ComponentOne Studio for ASP.NET Wijmo Help and click the Remove button.
3. Click **Yes** to remove the integrated help.

Deploying your Application in a Medium Trust Environment

Depending on your hosting choice, you may need to deploy your Web site or application in a medium trust environment. Often in a shared hosting environment, medium trust is required. In a medium trust environment several permissions are unavailable or limited, including OleDbPermission, ReflectionPermission, and FileIOPermission. You can configure your Web.config file to enable these permissions.

Note: ComponentOne controls will not work in an environment where reflection is not allowed.

ComponentOne ASP.NET Wijmo controls include the AllowPartiallyTrustedCallers() assembly attribute and will work under the medium trust level with some changes to the Web.config file. Since this requires some control over

the Web.config file, please check with your particular host to determine if they can provide the rights to override these security settings.

Modifying or Editing the Config File

In order to add permissions, you can edit the existing web_mediumtrust.config file or create a custom policy file based on the medium trust policy. If you modify the existing web_mediumtrust.config file, all Web applications will have the same permissions with the permissions you have added. If you want applications to have different permissions, you can instead create a custom policy based on medium trust.

Edit the Config File

In order to add permissions, you can edit the existing web_mediumtrust.config file. To edit the existing web_mediumtrust.config file, complete the following steps:

1. Locate the medium trust policy file web_mediumtrust.config located by default in the %windir%\Microsoft.NET\Framework\{Version}\CONFIG directory.
2. Open the web_mediumtrust.config file.
3. Add the permissions that you want to grant. For examples, see [Adding Permissions](#) (page 4).

Create a Custom Policy Based on Medium Trust

In order to add permissions, you can create a custom policy file based on the medium trust policy. To create a custom policy file, complete the following steps:

1. Locate the medium trust policy file web_mediumtrust.config located by default in the %windir%\Microsoft.NET\Framework\{Version}\CONFIG directory.
2. Copy the web_mediumtrust.config file and create a new policy file in the same directory.
Give the new a name that indicates that it is your variation of medium trust; for example, AllowReflection_Web_MediumTrust.config.
3. Add the permissions that you want to grant. For examples, see [Adding Permissions](#) (page 4).
4. Enable the custom policy file on your application by modifying the following lines in your web.config file under the `<system.web>` node:

```
<system.web>
<trust level="CustomMedium" originUrl="" />

  <securityPolicy>
    <trustLevel name="CustomMedium"
policyFile="AllowReflection_Web_MediumTrust.config" />
  </securityPolicy>
  ...
</system.web>
```

Note: Your host may not allow trust level overrides. Please check with your host to see if you have these rights.

Allowing Deserialization

To allow the deserialization of the license added to App_Licenses.dll by the Microsoft IDE, you should add the SerializationFormatter flag to security permission to the Web.config file. Complete the steps in the [Modifying or Editing the Config File](#) (page 3) topic to create or modify a policy file before completing the following.

Add the `SerializationFormatter` flag to the `<IPermission class="SecurityPermission">` tag so that it appears similar to the following:

```
<NamedPermissionSets>
  <PermissionSet
    class="NamedPermissionSet"
```

```

    version="1"
    Name="ASP.Net">
      <IPermission
        class="SecurityPermission"
        version="1"
        Flags="Assertion, Execution, ControlThread,
ControlPrincipal, RemotingConfiguration, SerializationFormatter"/>
      ...
    </PermissionSet>
  </NamedPermissionSets>

```

Adding Permissions

You can add permission, including ReflectionPermission, OleDbPermission, and FileIOPermission, to the web.config file. Note that ComponentOne controls will not work in an environment where reflection is not allowed. Complete the steps in the [Modifying or Editing the Config File](#) (page 3) topic to create or modify a policy file before completing the following.

ReflectionPermission

By default ReflectionPermission is not available in a medium trust environment. ComponentOne ASP.NET Wijmo controls require reflection permission because LicenseManager.Validate() causes a link demand for full trust.

To add reflection permission, complete the following:

1. Open the web_mediumtrust.config file or a file created based on the web_mediumtrust.config file.
2. Add the following `<SecurityClass>` tag after the `<SecurityClasses>` tag so that it appears similar to the following:

```

<SecurityClasses>
  <SecurityClass Name="ReflectionPermission"
Description="System.Security.Permissions.ReflectionPermission,
mscorlib, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089"/>
  ...
</SecurityClasses>

```

3. Add the following `<IPermission>` tag after the `<NamedPermissionSets>` tag so it appears similar to the following:

```

<NamedPermissionSets>
  <PermissionSet class="NamedPermissionSet" version="1"
Name="ASP.Net">
    <IPermission
      class="ReflectionPermission"
      version="1"
      Flags="ReflectionEmit,MemberAccess" />
    ...
  </PermissionSet>
</NamedPermissionSets>

```

4. Save and close the web_mediumtrust.config file.

OleDbPermission

By default OleDbPermission is not available in a medium trust environment. This means you cannot use the ADO.NET managed OLE DB data provider to access databases. If you wish to use the ADO.NET managed OLE DB data provider to access databases, you must modify the web_mediumtrust.config file.

To add OleDbPermission, complete the following steps:

1. Open the web_mediumtrust.config file or a file created based on the web_mediumtrust.config file.

2. Add the following `<SecurityClass>` tag after the `<SecurityClasses>` tag so that it appears similar to the following:

```
<SecurityClasses>
  <SecurityClass Name="OleDbPermission"
  Description="System.Data.OleDb.OleDbPermission, System.Data,
  Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
  ...
</SecurityClasses>
```

3. Add the following `<IPermission>` tag after the `<NamedPermissionSets>` tag so it appears similar to the following:

```
<NamedPermissionSets>
  <PermissionSet class="NamedPermissionSet" version="1"
  Name="ASP.Net">
    <IPermission class="OleDbPermission" version="1"
  Unrestricted="true"/>
    ...
  </PermissionSet>
</NamedPermissionSets>
```

4. Save and close the `web_mediumtrust.config` file.

FileIOPermission

By default, FileIOPermission is not available in a medium trust environment. This means no file access is permitted outside of the application's virtual directory hierarchy. If you wish to allow additional file permissions, you must modify the `web_mediumtrust.config` file.

To modify FileIOPermission to allow read access to a specific directory outside of the application's virtual directory hierarchy, complete the following steps:

1. Open the `web_mediumtrust.config` file or a file created based on the `web_mediumtrust.config` file.
2. Add the following `<SecurityClass>` tag after the `<SecurityClasses>` tag so that it appears similar to the following:

```
<SecurityClasses>
  <SecurityClass Name="FileIOPermission"
  Description="System.Security.Permissions.FileIOPermission, mscorlib,
  Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
  ...
</SecurityClasses>
```

3. Add the following `<IPermission>` tag after the `<NamedPermissionSets>` tag so it appears similar to the following:

```
<NamedPermissionSets>
  <PermissionSet class="NamedPermissionSet" version="1"
  Name="ASP.Net">
    ...
    <IPermission class="FileIOPermission" version="1"
  Read="C:\SomeDir;$AppDir$" Write="$AppDir$" Append="$AppDir$"
  PathDiscovery="$AppDir$" />
    ...
  </PermissionSet>
</NamedPermissionSets>
```

4. Save and close the `web_mediumtrust.config` file.

End-User License Agreement

All of the ComponentOne licensing information, including the ComponentOne end-user license agreements, frequently asked licensing questions, and the ComponentOne licensing model, is available online at <http://www.componentone.com/SuperPages/Licensing/>.

Licensing FAQs

This section describes the main technical aspects of licensing. It may help the user to understand and resolve licensing problems he may experience when using ComponentOne .NET and ASP.NET products.

What is Licensing?

Licensing is a mechanism used to protect intellectual property by ensuring that users are authorized to use software products.

Licensing is not only used to prevent illegal distribution of software products. Many software vendors, including ComponentOne, use licensing to allow potential users to test products before they decide to purchase them.

Without licensing, this type of distribution would not be practical for the vendor or convenient for the user. Vendors would either have to distribute evaluation software with limited functionality, or shift the burden of managing software licenses to customers, who could easily forget that the software being used is an evaluation version and has not been purchased.

How does Licensing Work?

ComponentOne uses a licensing model based on the standard set by Microsoft, which works with all types of components.

Note: The **Compact Framework** components use a slightly different mechanism for run-time licensing than the other ComponentOne components due to platform differences.

When a user decides to purchase a product, he receives an installation program and a Serial Number. During the installation process, the user is prompted for the serial number that is saved on the system. (Users can also enter the serial number by clicking the **License** button on the **About Box** of any ComponentOne product, if available, or by rerunning the installation and entering the serial number in the licensing dialog box.)

When a licensed component is added to a form or Web page, Visual Studio obtains version and licensing information from the newly created component. When queried by Visual Studio, the component looks for licensing information stored in the system and generates a run-time license and version information, which Visual Studio saves in the following two files:

- An assembly resource file which contains the actual run-time license
- A "licenses.licx" file that contains the licensed component strong name and version information

These files are automatically added to the project.

In WinForms and ASP.NET 1.x applications, the run-time license is stored as an embedded resource in the assembly hosting the component or control by Visual Studio. In ASP.NET 2.x applications, the run-time license may also be stored as an embedded resource in the App_Licenses.dll assembly, which is used to store all run-time licenses for all components directly hosted by WebForms in the application. Thus, the App_licenses.dll must always be deployed with the application.

The licenses.licx file is a simple text file that contains strong names and version information for each of the licensed components used in the application. Whenever Visual Studio is called upon to rebuild the application resources, this file is read and used as a list of components to query for run-time licenses to be embedded in the appropriate assembly resource. Note that editing or adding an appropriate line to this file can force Visual Studio to add run-time licenses of other controls as well.

Note that the licenses.licx file is usually not shown in the Solution Explorer; it appears if you press the **Show All Files** button in the Solution Explorer's Toolbox, or from Visual Studio's main menu, select **Show All Files** on the **Project** menu.

Later, when the component is created at run time, it obtains the run-time license from the appropriate assembly resource that was created at design time and can decide whether to simply accept the run-time license, to throw an exception and fail altogether, or to display some information reminding the user that the software has not been licensed.

All ComponentOne products are designed to display licensing information if the product is not licensed. None will throw licensing exceptions and prevent applications from running.

Common Scenarios

The following topics describe some of the licensing scenarios you may encounter.

Creating components at design time

This is the most common scenario and also the simplest: the user adds one or more controls to the form, the licensing information is stored in the licenses.licx file, and the component works.

Note that the mechanism is exactly the same for Windows Forms and Web Forms (ASP.NET) projects.

Creating components at run time

This is also a fairly common scenario. You do not need an instance of the component on the form, but would like to create one or more instances at run time.

In this case, the project will not contain a licenses.licx file (or the file will not contain an appropriate run-time license for the component) and therefore licensing will fail.

To fix this problem, add an instance of the component to a form in the project. This will create the licenses.licx file and things will then work as expected. (The component can be removed from the form after the licenses.licx file has been created).

Adding an instance of the component to a form, then removing that component, is just a simple way of adding a line with the component strong name to the licenses.licx file. If desired, you can do this manually using notepad or Visual Studio itself by opening the file and adding the text. When Visual Studio recreates the application resources, the component will be queried and its run-time license added to the appropriate assembly resource.

Inheriting from licensed components

If a component that inherits from a licensed component is created, the licensing information to be stored in the form is still needed. This can be done in two ways:

- Add a LicenseProvider attribute to the component.

This will mark the derived component class as licensed. When the component is added to a form, Visual Studio will create and manage the licenses.licx file, and the base class will handle the licensing process as usual. No additional work is needed. For example:

```
[LicenseProvider(typeof(LicenseProvider))]
class MyGrid: C1.Win.C1FlexGrid.C1FlexGrid
{
    // ...
}
```

- Add an instance of the base component to the form.

This will embed the licensing information into the licenses.licx file as in the previous scenario, and the base component will find it and use it. As before, the extra instance can be deleted after the licenses.licx file has been created.

Please note, that C1 licensing will not accept a run-time license for a derived control if the run-time license is embedded in the same assembly as the derived class definition, and the assembly is a DLL. This restriction is necessary to prevent a derived control class assembly from being used in other applications without a design-time license. If you create such an assembly, you will need to take one of the actions previously described create a component at run time.

Using licensed components in console applications

When building console applications, there are no forms to add components to, and therefore Visual Studio won't create a licenses.licx file.

In these cases, create a temporary Windows Forms application and add all the desired licensed components to a form. Then close the Windows Forms application and copy the licenses.licx file into the console application project.

Make sure the licenses.licx file is configured as an embedded resource. To do this, right-click the licenses.licx file in the Solution Explorer window and select **Properties**. In the Properties window, set the **Build Action** property to **Embedded Resource**.

Using licensed components in Visual C++ applications

There is an issue in VC++ 2003 where the licenses.licx is ignored during the build process; therefore, the licensing information is not included in VC++ applications.

To fix this problem, extra steps must be taken to compile the licensing resources and link them to the project. Note the following:

1. Build the C++ project as usual. This should create an .exe file and also a licenses.licx file with licensing information in it.
2. Copy the licenses.licx file from the app directory to the target folder (Debug or Release).
3. Copy the C1Lc.exe utility and the licensed dlls to the target folder. (Don't use the standard lc.exe, it has bugs.)
4. Use C1Lc.exe to compile the licenses.licx file. The command line should look like this:

```
c1lc /target:MyApp.exe /complist:licenses.licx  
/i:C1.Win.C1FlexGrid.dll
```
5. Link the licenses into the project. To do this, go back to Visual Studio, right-click the project, select properties, and go to the Linker/Command Line option. Enter the following:

```
/ASSEMBLYRESOURCE:Debug\MyApp.exe.licenses
```
6. Rebuild the executable to include the licensing information in the application.

Using licensed components with automated testing products

Automated testing products that load assemblies dynamically may cause them to display license dialog boxes. This is the expected behavior since the test application typically does not contain the necessary licensing information, and there is no easy way to add it.

This can be avoided by adding the string "C1CheckForDesignLicenseAtRuntime" to the AssemblyConfiguration attribute of the assembly that contains or derives from ComponentOne controls. This attribute value directs the ComponentOne controls to use design-time licenses at run time.

For example:

```
#if AUTOMATED_TESTING  
    [AssemblyConfiguration("C1CheckForDesignLicenseAtRuntime")]  
#endif  
public class MyDerivedControl : C1LicensedControl  
{  
    // ...  
}
```

```
}
```

Note that the AssemblyConfiguration string may contain additional text before or after the given string, so the AssemblyConfiguration attribute can be used for other purposes as well. For example:

```
[AssemblyConfiguration("C1CheckForDesignLicenseAtRuntime,BetaVersion")]
```

THIS METHOD SHOULD ONLY BE USED UNDER THE SCENARIO DESCRIBED. It requires a design-time license to be installed on the testing machine. Distributing or installing the license on other computers is a violation of the EULA.

Troubleshooting

We try very hard to make the licensing mechanism as unobtrusive as possible, but problems may occur for a number of reasons.

Below is a description of the most common problems and their solutions.

I have a licensed version of a ComponentOne product but I still get the splash screen when I run my project.

If this happens, there may be a problem with the licenses.licx file in the project. It either doesn't exist, contains wrong information, or is not configured correctly.

First, try a full rebuild (**Rebuild All** from the Visual Studio **Build** menu). This will usually rebuild the correct licensing resources.

If that fails follow these steps:

1. Open the affected project.
2. Select an instance of the updated component.
3. In the Visual Studio Properties window, change any property. It does not matter which property you change; you can change it back to the previous value.
4. Rebuild the project using the **Rebuild All** option (not just **Rebuild**) and run the solution.

Alternative 1: Follow these steps:

1. Open a new Visual Studio.NET project.
2. Add the updated component to the form.
3. Compile and run the new project.
4. Open the licenses.licx file in the new project.
5. Copy the line that starts with the namespace of the updated component (for example, C1.Win.C1Report) and ends with a public key token.
6. Open the existing, incorrectly licensed project.
7. Open the licenses.licx file in the new project.
8. Paste the line from step 5 into this file (replace the old licensing information with the new).
9. Rebuild the project using the **Rebuild All** option (not just **Rebuild**) and run the solution.

Alternative 2: Follow these steps:

1. Open the affected project.
2. Delete the licenses.licx file from the project.
3. Add a new instance of the updated component to the form.
4. Rebuild and run the solution. The nag screen should not appear.

5. Remove the newly added component from the form.

Try each of these options multiple times, if necessary. If that still does not help, are you creating any of the controls in code rather than design-time? If so, you must add an entry for the control in the licenses.licx file (see <http://helpcentral.componentone.com/PrintableView.aspx?ID=1886> for more information). Also if this is a website, as opposed to an ASP.NET web application, please try right-clicking the licenses.licx file and selecting "Build Runtime Licenses" from the context menu.

I have a licensed version of a ComponentOne product on my Web server but the components still behave as unlicensed.

There is no need to install any licenses on machines used as servers and not used for development.

The components must be licensed on the development machine, therefore the licensing information will be saved into the executable (.exe or .dll) when the project is built. After that, the application can be deployed on any machine, including Web servers.

For ASP.NET 2.x applications, be sure that the App_Licenses.dll assembly created during development of the application is deployed to the bin application bin directory on the Web server.

If your ASP.NET application uses WinForms user controls with constituent licensed controls, the run-time license is embedded in the WinForms user control assembly. In this case, you must be sure to rebuild and update the user control whenever the licensed embedded controls are updated.

I downloaded a new build of a component that I have purchased, and now I'm getting the splash screen when I build my projects.

Make sure that the serial number is still valid. If you licensed the component over a year ago, your subscription may have expired. In this case, you have two options:

Option 1 – Renew your subscription to get a new serial number.

If you choose this option, you will receive a new serial number that you can use to license the new components (from the installation utility or directly from the **About Box**).

The new subscription will entitle you to a full year of upgrades and to download the latest maintenance builds directly from <http://prerelease.componentone.com/>.

Option 2 – Continue to use the components you have.

Subscriptions expire, products do not. You can continue to use the components you received or downloaded while your subscription was valid.

Technical Support

ComponentOne offers various support options. For a complete list and a description of each, visit the ComponentOne Web site at <http://www.componentone.com/SuperProducts/SupportServices/>.

Some methods for obtaining technical support include:

- **Online Resources**
ComponentOne provides customers with a comprehensive set of technical resources in the form of FAQs, samples and videos, Version Release History, searchable Knowledge base, searchable Online Help and more. We recommend this as the first place to look for answers to your technical questions.
- **Online Support via our Incident Submission Form**
This online support service provides you with direct access to our Technical Support staff via an [online incident submission form](#). When you submit an incident, you'll immediately receive a response via e-mail confirming that you've successfully created an incident. This email will provide you with an Issue Reference ID and will provide you with a set of possible answers to your question from our

Knowledgebase. You will receive a response from one of the ComponentOne staff members via e-mail in 2 business days or less.

- **Product Forums**
ComponentOne's [product forums](#) are available for users to share information, tips, and techniques regarding ComponentOne products. ComponentOne developers will be available on the forums to share insider tips and technique and answer users' questions. Please note that a ComponentOne User Account is required to participate in the ComponentOne Product Forums.
- **Installation Issues**
Registered users can obtain help with problems installing ComponentOne products. Contact technical support by using the [online incident submission form](#) or by phone (412.681.4738). Please note that this does not include issues related to distributing a product to end-users in an application.
- **Documentation**
Microsoft integrated ComponentOne documentation can be installed with each of our products, and documentation is also available online. If you have suggestions on how we can improve our documentation, please email the [Documentation team](#). Please note that e-mail sent to the [Documentation team](#) is for documentation feedback only. [Technical Support](#) and [Sales](#) issues should be sent directly to their respective departments.

Note: You must create a ComponentOne Account and register your product with a valid serial number to obtain support using some of the above methods.

Redistributable Files

ComponentOne Studio for ASP.NET Wijmo is developed and published by ComponentOne LLC. You may use it to develop applications in conjunction with Microsoft Visual Studio or any other programming environment that enables the user to use and integrate the control(s). You may also distribute, free of royalties, the following Redistributable Files with any such application you develop to the extent that they are used separately on a single CPU on the client/workstation side of the network:

- C1.Web.Wijmo.Controls.3.dll
- C1.Web.Wijmo.Controls.Design.3.dll
- C1.Web.Wijmo.Controls.4.dll
- C1.Web.Wijmo.Controls.Design.4.dll
- C1.Web.Wijmo.Extenders.3.dll
- C1.Web.Wijmo.Extenders.4.dll
- C1.C1Report.2.dll
- C1.C1Report.4.dll

Site licenses are available for groups of multiple developers. Please contact Sales@ComponentOne.com for details.

About This Documentation

Acknowledgements

Microsoft, Windows, Windows Vista, Visual Studio, and Microsoft Expression are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Firefox is a registered trademark of the Mozilla Foundation.

Safari is a registered trademark of Apple Inc.

ComponentOne

If you have any suggestions or ideas for new features or controls, please call us or write:

Corporate Headquarters

ComponentOne LLC

201 South Highland Avenue

3rd Floor

Pittsburgh, PA 15206 • USA

412.681.4343

412.681.4384 (Fax)

<http://www.componentone.com>

ComponentOne Doc-To-Help

This documentation was produced using [ComponentOne Doc-To-Help® Enterprise](#).

Namespaces

Namespaces organize the objects defined in an assembly. Assemblies can contain multiple namespaces, which can in turn contain other namespaces. Namespaces prevent ambiguity and simplify references when using large groups of objects such as class libraries.

The general namespace for ComponentOne Web products is **C1.Web**. The following code fragment shows how to declare a **C1InputMask** using the fully qualified name for this class:

- Visual Basic
`Dim Tabs As C1.Web.Wijmo.Controls.C1Tabs.C1Tabs`
- C#
`C1.Web.Wijmo.Controls.C1Tabs.C1Tabs Tabs;`

Namespaces address a problem sometimes known as *namespace pollution*, in which the developer of a class library is hampered by the use of similar names in another library. These conflicts with existing components are sometimes called *name collisions*.

Fully qualified names are object references that are prefixed with the name of the namespace where the object is defined. You can use objects defined in other projects if you create a reference to the class (by choosing Add Reference from the Project menu) and then use the fully qualified name for the object in your code.

Fully qualified names prevent naming conflicts because the compiler can always determine which object is being used. However, the names themselves can get long and cumbersome. To get around this, you can use the Imports statement (**using** in C#) to define an alias — an abbreviated name you can use in place of a fully qualified name. For example, the following code snippet creates aliases for two fully qualified names, and uses these aliases to define two objects:

- Visual Basic
`Imports C1Tabs = C1.Web.UI.Controls.C1Tabs.C1Tabs
Imports MyTabs = MyProject.Objects.C1Tabs.C1Tabs

Dim wm1 As C1Tabs
Dim wm2 As MyTabs`
- C#
`using C1Tabs = C1.Web.UI.Controls.C1Tabs.C1Tabs;
using MyTabs = MyProject.Objects.C1Tabs.C1Tabs;

C1Tabs wm1;
MyTabs wm2;`

If you use the **Imports** statement without an alias, you can use all the names in that namespace without qualification provided they are unique to the project.

Creating an ASP.NET Project

ComponentOne Studio for ASP.NET Wijmo requires Visual Studio 2008 or later and .NET Framework 3.0 or later for your Web applications. **Studio for ASP.NET Wijmo** does not require AJAX extensions; however, you can install them if you want to use AJAX in your project. See the following table for more details on installing AJAX extensions.

Visual Studio 2010	You can build Ajax-enabled ASP.NET projects by downloading the AJAX Control Toolkit from CodePlex and dragging-and-dropping the controls from the Visual Studio Toolbox onto an ASP.NET Web Forms page.
Visual Studio 2008, .NET Framework 3.5	You can easily create an AJAX-enabled ASP.NET project without installing separate add-ins because the framework has a built-in AJAX library and controls.
Visual Studio 2008, .NET Framework 3.0	You must install the ASP.NET AJAX Extensions 1.0, which can be found at http://www.asp.net/ajax/downloads/archive/ . Then you can create an AJAX 1.0-Enabled ASP.NET 2.0 Web site or application.

The following topics explain how to create projects in Visual Studio 2010 and 2008.

- [Creating a Web Site/Application Project in Visual Studio 2010](#) 📌

To create a new Web site/application project in Visual Studio 2010, complete the following steps.

1. If you are creating an AJAX project, download the AJAX Control Toolkit from [CodePlex](#) and extract the files.
2. From the **File** menu, select **New | Web Site/Project**. The New Web Site/New Project dialog box opens.
3. Select a .NET Framework in the upper right corner. Note that if you choose .NET Framework 3.0, you must install the [extensions](#) first.
4. Under **Project Types**, choose either **Visual Basic** or **Visual C#** and then select **Web**. Note that one of these options may be located under **Other Languages**.
5. Select a language, and in the list of templates, select **ASP.NET Web Site/Application**.
6. Specify a location and then click **OK**.

Note: The Web server must have IIS version 6 or later and the .NET Framework installed on it. If you have IIS on your computer, you can specify http://localhost for the server.

A new Web project is created at the root of the Web server you specified.

7. If you are creating an AJAX project, right-click the Toolbox (create a new tab if you like), select **Choose Items** and browse to find the **AjaxControlToolkit.dll**. You can begin dragging toolkit controls to your page. Note that if you do not see the toolkit controls in the Toolbox, make sure you selected the .NET Framework that corresponds with the version of the toolkit you downloaded.

- [Creating a Web Site/Application Project in Visual Studio 2008](#) 📌

To create a Web site/application project in Visual Studio 2008, complete the following steps:

1. From the **File** menu, select **New | Web Site/Project**. The New Web Site/New Project dialog box opens.
2. Select .NET Framework 3.5 or 3.0 in the upper right corner. Note that if you choose .NET Framework 3.0, you must install the [extensions](#) first.

3. Select a language, and in the list of templates, select **ASP.NET Web Site/Application** or **AJAX 1.0-Enabled ASP.NET 2.0 Web Site/Application**.
4. Specify a location and then click **OK**.

Note: The Web server must have IIS version 6 or later and the .NET Framework installed on it. If you have IIS on your computer, you can specify `http://localhost` for the server.

A new Web project is created at the root of the Web server you specified.

Adding the Studio for ASP.NET Wijmo Components to a Project

When you open Visual Studio, you will notice a **ComponentOne Studio for ASP.NET Wijmo Projects** tab containing the ComponentOne controls that have automatically been added to the Toolbox.

Note that you can manually add ComponentOne controls to the Toolbox at a later time.

Manually Adding the Studio for ASP.NET Wijmo controls to the Toolbox

When you install **ComponentOne Studio for ASP.NET Wijmo**, the following Input for ASP.NET Wijmo components will appear in the Visual Studio Toolbox customization dialog box.

To manually add the Studio for ASP.NET Wijmo controls to the Visual Studio Toolbox:

1. Open the Visual Studio IDE (Microsoft Development Environment). Make sure the Toolbox is visible (select **Toolbox** in the **View** menu if necessary) and right-click it to open the context menu.
2. To make the Studio for ASP.NET Wijmo components appear on their own tab in the Toolbox, select **Add Tab** from the context menu and type in the tab name, Studio for ASP.NET Wijmo, for example.
3. Right-click the tab where the component is to appear and select **Choose Items** from the context menu. The **Choose Toolbox Items** dialog box opens.
4. In the dialog box, select the **.NET Framework Components** tab. Sort the list by Namespace (click the **Namespace** column header) and check the check boxes for all components belonging to namespace `C1.Web.Wijmo.Controls.C1Input`. Note that there may be more than one component for each namespace.
5. Click **OK** to close the dialog box. The controls are added to the Visual Studio Toolbox.

Adding Studio for ASP.NET Wijmo Controls to the Form

To add **Studio for ASP.NET Wijmo** controls to a form:

1. Add them to the Visual Studio Toolbox.
2. Double-click each control or drag it onto your form.

Adding a Reference to the Assembly

To add a reference to the `C1.Web.Wijmo.Controls.3` or `C1.Web.Wijmo.Controls.4` assembly:

1. Select the **Add Reference** option from the **Website** menu of your Web Site project or from the **Project** menu of your Web Application project.
2. Select the most recent version of the **ComponentOne Studio for ASP.NET Wijmo** assembly from the list on the **NET** tab or browse to find the `C1.Web.Wijmo.Controls.3.dll` or `C1.Web.Wijmo.Controls.4.dll` file and click **OK**.
3. Select the **Form1.vb** tab or go to **View | Code** to open the Code Editor. At the top of the file, add the following **Imports** directive (**using** in C#):

```
Imports C1.Web.Wijmo.Controls
```

Note: This makes the objects defined in the **C1.Web.Wijmo.Controls.3(4)** assembly visible to the project. See [Namespaces](#) (page 12) for more information.

ComponentOne ASP.NET Tabs for ASP.NET Wijmo Migration Guide

ComponentOne is constantly making efforts to be the leading edge for any .net controls. Our new Wijmo controls for ASP.net are precisely that. ASP.NET Wijmo contains some exciting advances visually, and it also simplifies the development processes overall. Based upon the new Wijmo framework, these ASP.NET controls have been completely re-engineered from the ground up. The new controls leverage the latest technologies available to create the ultimate development experience including:

1. **Improved Performance:** Fully extensible client-side and server-side object models with faster load times.
2. **Latest Standards Support:** CSS3 and HTML5 compliance.
3. **Major Browser Support:** Internet Explorer, Firefox, Chrome and Safari.
4. **Themes:** Built-in premium CSS3 themes and all Controls are compatible with jQuery UI Themroller.
5. Tightly integrated with jQuery.

The C1Tabs control is one of the new controls included in our ASP.net Wijmo line up. Let's take a look at the procedures involved in migrating over to our latest controls and delve into some details about some basic differences between ASP.NET Wijmo controls and their older ASP.NET AJAX counterparts along the way.

Prerequisites

To get started in this migration you will need our ASP.net Wijmo controls. These can be downloaded from <http://www.componentone.com/SuperProducts/StudioASPNET> by clicking the orange "Download Free Trial" button.

Getting Started

To get things started, you will create a new ASP.NET Web application using ComponentOne's previous ASP.NET AJAX controls. Below are links to the quick-start guides where you will receive directions to create a website with these controls and visual aid as to what the final product will look like. Once each control is set up on its respective Web application, we can then begin to delve into the migration details for our ASP.net Wijmo controls. During the migration process, we will note changes between the Palomino and Wijmo controls.

C1TabsControl

Follow this quick-start to create the C1Tabs using our ASP.net C1TabControl control:

<http://helpcentral.componentone.com/nethelp/c1tabcontrol/tabcontrolforaspnetajaxquickstart.htm>

Migration details

Once you have completed each of quick start guides you can begin making changes to your respective programs as migration to the new ASP.NET Wijmo controls. The first thing you will want to do is add the new ASP.net Wijmo assembly references. From the menu bar, select project and then select **Add Reference**. Click the **Browse...** button and navigate to: =

For .NET 4.0:

- *For 32bit Machines:* C:\Program Files\ComponentOne\Studio for ASP.NET Wijmo\bin\v4
- *For 64bit Machines:* C:\Program Files (x86)\ComponentOne\Studio for ASP.NET Wijmo\bin\v4

Select **C1.Web.Wijmo.Controls.4** and click **Open**.


For .NET 3.5:

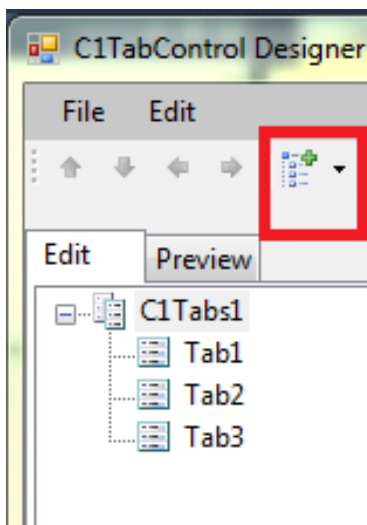
- *For 32bit Machines:* C:\Program Files\ComponentOne\Studio for ASP.NET Wijmo\bin\v3

- For 64bit Machines: C:\Program Files (x86)\ComponentOne\Studio for ASP.NET Wijmo\bin\v3

Select the **C1.Web.Wijmo.Controls.3** and click **Open**. Then, in the **Add Reference to...** dialog box, click **Add**. In your solution, open up your page where you have your ASP.NET control in Design view.

C1TabsControl to C1Tabs

Add the **C1Tabs** control by dragging the control to your ASP.net page. This will display a content box in which your tabs will display viewable content for each individual tab. To begin, click the smart tag  and click the **C1Tabs.SmartTag.Designer** link item. Similar to the **C1TabsControl** from our older ASP.NET controls, the **C1TabControl Designer** dialog box contains an edit tab, which you can use to manipulate all the properties, and a **Preview** tab where you can view the changes you've made. In the **Edit** tab use the add child item button to add some **C1TabItems** to your **C1Tab** control. Add as many tabs as you would like. In my example I have added three tabs. When you are finished, click **OK** to save the changes that you have made.



In Design view, you will be able to select each tab and edit the content within the contents pane on the control. Shown below are my three tabs with themes Cobalt and Midnight themes respectively.

The default .aspx code for the Wijmo control is slightly different than its palomino control. Here are some examples of the differences:

Palomino	Wijmo
<pre><cc2:C1TabControl ID="C1Tab Control1"runat="server" VisualStylePath=~\C1WebCon trols/VisualStyles"></pre>	<pre><wijmo:C1Tabs ID="C1Tabs1" runat="server"></pre>
<pre><TabPage> <cc2:C1TabPage runat="ser ver" Text="C1TabPage1"></cc2:C1T abPage></pre>	<pre><Pages> <wijmo:C1TabPage runat="server" Text="Tab1" ID="Tab1"></wijmo:C1T abPage> <wijmo:C1TabPage runat="server"</pre>

<pre> <cc2:C1TabPage runat="server" Text="C1TabPage2"></cc2:C1T abPage> <cc2:C1TabPage runat="ser ver" Text="C1TabPage3"></cc2:C1T abPage> </TabPage> </pre>	<pre> Text="Tab2" ID="Tab2"></wijmo:C1T abPage> <wijmo:C1TabPage runat="server" Text="Tab3" ID="Tab3"></wijmo:C1T abPage> </Pages> </pre>
--	--

Key Features

The following are some of the main features of C1Tabs that you may find useful:

- **Tab Directions**
The C1Tabs is oriented top-horizontal by default. You can easily change the direction to right, left, or bottom
- **Host External Content**
Host external content in a page; display the content of another Web page in your project or the content of a website outside of your project.
- **Scrollable**
When there are too many tabs to fit along the width of the tab, the tabs will scroll instead of wrapping if the scrollable option is set to true.
- **Add/remove Tabs**
Dynamically add tabs using the **addTab** function. In addition, you may include a function that will allow each tab to close by clicking on an 'x' on the tabs.
- **Animation**
Add transition effects when end-users navigate between tab pages. Select from built-in animations or create your own custom effects.
- **Theming**
With just a click of the SmartTag, change the tabs' look by selecting one of the 5 premium themes (Midnight, Aristo, Rocket, Cobalt, and Sterling). Optionally, use ThemeRoller from jQuery UI to create a customized theme!
- **CSS Support**
Use a cascading style sheet (CSS) style to define custom skins. CSS support allows you to match the tab control to your organization's standards.

Wijmo Top Tips

The following tips may help you troubleshoot when working with Studio for ASP.NET Wijmo.

Tip 1: Prevent poor page rendering in quirks mode by editing the meta tag to fix rendering.

If a user's browser is rendering a page in quirks mode, widgets and controls may not appear correctly on the page. This is indicated by a broken page icon in the address bar. In **Compatibility View**, the browser uses an older rendering engine.



Users can set this view that causes the issue. To prevent rendering in quirks mode, you can force the page to render with the latest browser. Add the following meta tag to the header of the page:

```
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
```

Tabs for ASP.NET Wijmo Quick Start

In this quick start guide, you will explore the functionality of the **C1Tabs** control by creating a simple paged area with navigation controls within a Web site.

Step 1 of 3: Adding C1Tabs to the Page

In this step you will begin by adding the **C1Tabs** control to the page.

To begin, complete the following steps:

1. Create a new ASP.NET Website project.
2. Click the **Design** tab located below the Document window to switch to Design view.
3. Navigate to the Visual Studio Toolbox and double-click **C1Tabs** to add the control to the page.

The Web page contains the empty **C1Tabs** control.

Step 2 of 3: Working with the C1Tabs Designer Form

The **C1Tabs Designer Form** allows you to easily customize the **C1Tabs** control and each **C1TabPage** you choose to include. In this step you will add three **C1TabPage**s to the **C1Tabs** control and change its behavior using the **C1Tabs Designer Form**.

Complete the following steps:

1. Click **C1Tabs**'s smart tag to open the **C1Tabs Tasks** menu and select **C1Tabs.SmartTag.Designer**.
The **C1Tabs Designer Form** opens.
2. In the **C1Tabs Designer Form**, click the **Add Child Item** button to add a **C1TabPage** to the control.
3. Click the **Add Child Item** button again to add another **C1TabPage** to the **C1Tabs** control.

4. Select **C1Tabs1** and set the following properties in the properties grid.
 - Set the **Height** property to **300px**.
 - Set the **Width** property to **400px**.
5. Click **OK** to save and close the **C1Tabs Designer Form** and observe that the C1Tabs control has two tabs.



In the next step, you will add content to the **C1Tabs** control and customize its appearance.

Step 3 of 3: Adding Content to the C1Tabs Control

In this step you will add content to the **C1Tabs** control. Adding content to the control is as simple as clicking in the control's body and typing text or adding controls from the Toolbox.

Complete the following to add standard controls and text content to pages in the **C1Tabs** control:

1. Click in the content area of the **C1Tabs** control and type "This is tab page 1."

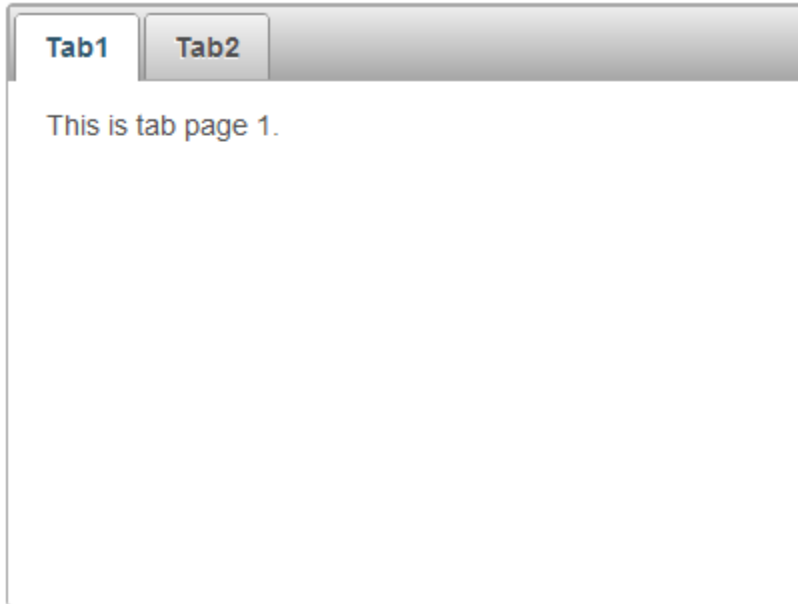
The text content is added to the first page of the **C1Tabs** control.

2. Switch to Source view to observe the structure of the C1Tabs control, and add the following `<asp:Calendar>` tag within the `<wijmo:C1TabsStep>` tag for the second step so it appears as follows:

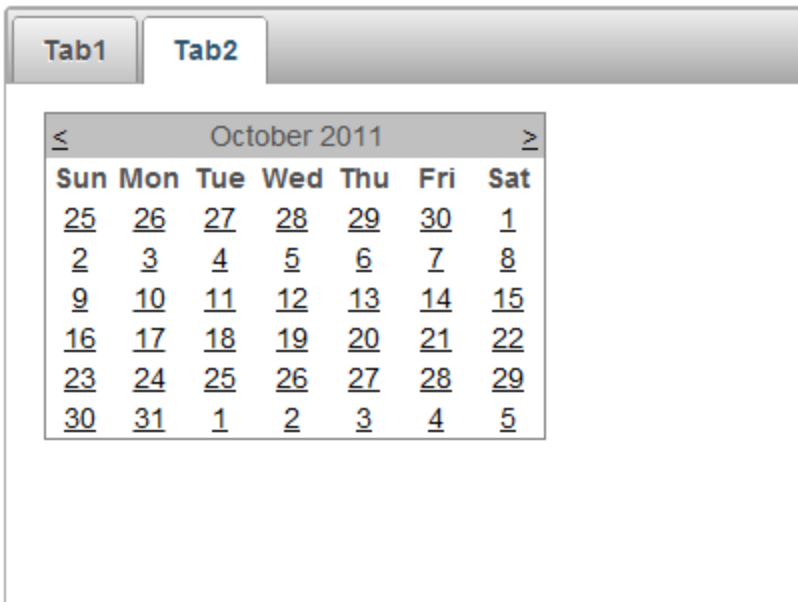
```
<wijmo:C1TabPage Text="Tab2" ID="Tab2" runat="server">  
    <asp:Calendar ID="Calendar1" runat="server"></asp:Calendar>  
</wijmo:C1TabPage>
```

This adds a **Calendar** control to the second page of the C1Tabs control.

3. Press F5 to run the application and observe that the C1Tabs control appears with its first tab page in focus:



4. Click **Tab2** and observe that its second tab page comes into focus:



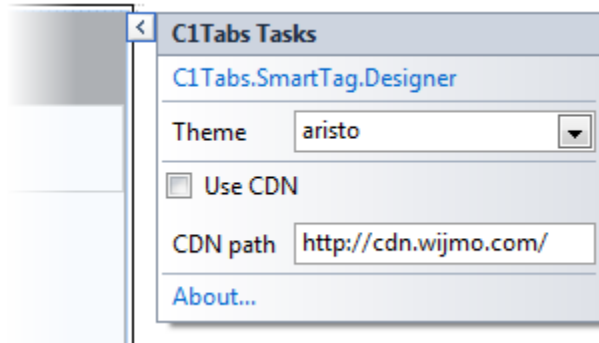
C1Tabs Design-Time Support

The following sections describe how to use **C1Tabs**' design-time environment to configure the C1Tabs control.

C1Tabs Smart Tag

The C1Tabs control includes a smart tag (📌) in Visual Studio. A smart tag represents a shortcut tasks menu that provides the most commonly used properties in C1Tabs.

To access the **C1Tabs Tasks** menu, click the smart tag in the upper-right corner of the C1Tabs control.



The **C1Tabs Tasks** menu operates as follows:

- **C1Tabs.SmartTag.Designer**
 - Clicking **C1Tabs.SmartTag.Designer** opens the **C1Tabs Designer Form** where you can quickly configure **C1Tabs**'s elements without having to scroll through its **Properties** window. You can load and save the control's content and can add **C1Tabs** to the control.
- **Theme**
 - Sets the theme to one of the built-in skins.
- **Use CDN**
 - Determines whether the control is using the CDN for the client-side reference.
- **CDN Path**
 - The path to the CDN library you are using.
- **About**
 - Clicking **About** reveals the **About ComponentOne** dialog box. This dialog box displays the version number and licensing information for the ComponentOne product.

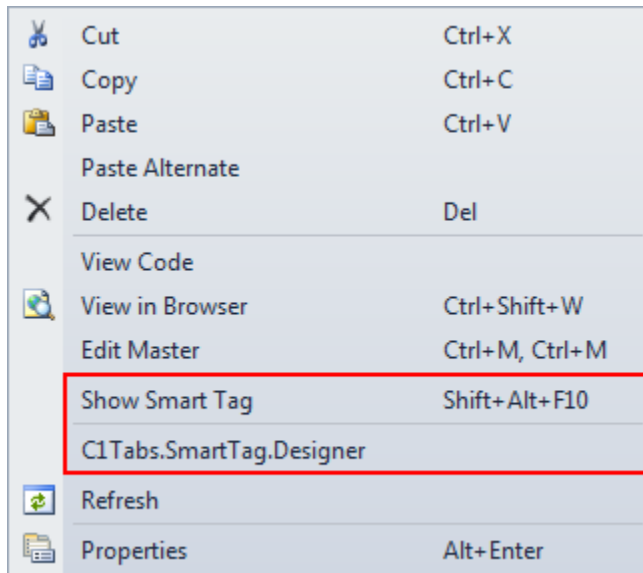
To access the **C1Tabs Collection Editor**:

1. Click the smart tag in the upper-right corner of the C1Tabs control to open the **C1Tabs Tasks** menu.
2. Select **Tabs Designer**.
3. Select the C1Tabs control.
4. Click the **ellipsis** button next to the C1Tabs property under the **Misc** category in the properties pane. The **C1Tabs Collection Editor** appears.

C1Tabs Context Menu

C1Tabs has additional commands available on the context menu that Visual Studio provides for all .NET and ASP.NET controls.

Right-click anywhere on the C1Tabs control to display the C1Tabs context menu:



The context menu commands operate as follows:

- **Show Smart Tag**
Clicking **Show Smart Tag** opens the **C1Tabs Tasks** menu.
- **Tabs Designer**
Clicking **Tabs Designer** opens the **C1Tabs Designer Form**, where **C1TabPage**s can be added, removed, and reordered. You can preview the C1Tabs here, as well as set a variety of properties defining the appearance, behavior, and more for each C1TabPage.

C1Tabs Designer Form

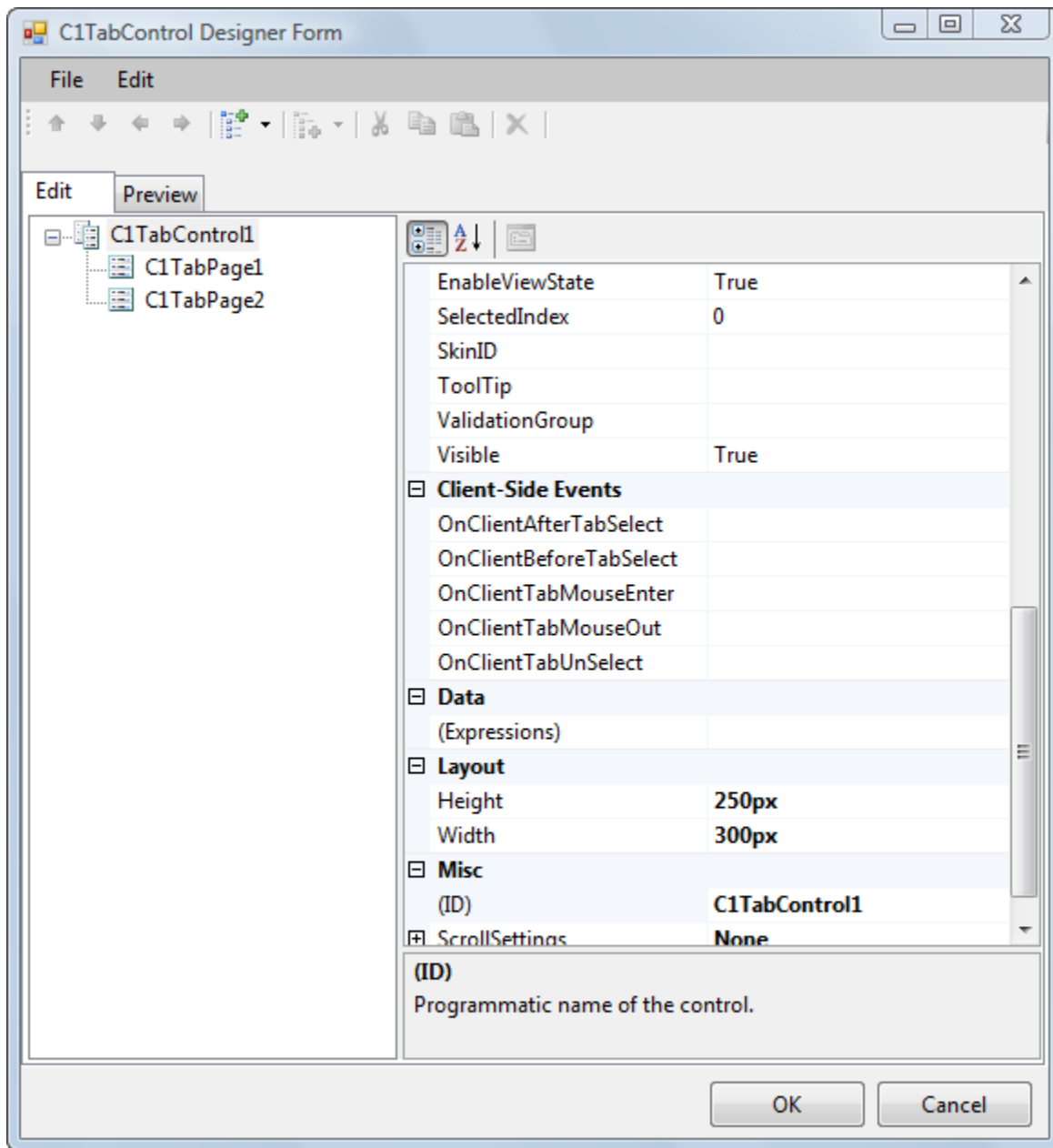
The **C1Tabs Designer Form** is **C1Tabs**' designer for editing its properties, as well as the **C1TabPage**s' properties. The **C1Tabs Designer Form** is similar to the Properties window as it allows you to modify the control visually. However, it allows you to: select the C1Tabs and **C1TabPage**s, set their properties, manipulate the C1TabPage locations, and then preview the appearance of the C1Tabs control, all within the form.

In this topic you will become familiar with the **C1Tabs Designer Form**'s design interface so you can use the commands within it to edit C1Tabs with minimal effort and time.

To open the **C1Tabs Designer Form**, click the C1Tabs smart tag and select the **Tabs Designer** link from the **C1Tabs Tasks** menu.

Exploring the C1Tabs Designer Form

The **C1Tabs Designer Form** contains a menu, toolbar, **Edit** tab, **Preview** tab, and properties pane.



- **C1Tabs Designer Form Menu**





The **C1Tabs Designer Form** menu contains the following menu items and subitems:

Menu Item	Submenu Item	Description
File	Exit	Closes the C1Tabs Designer Form .
Edit	Insert Item	Inserts a C1TabPage at the specified place in the list of tabs and separators.
	Add Child	Adds a C1TabPage as a child of the selected C1Tabs.
	Cut	Cuts the selected C1TabPage to be moved in the list of

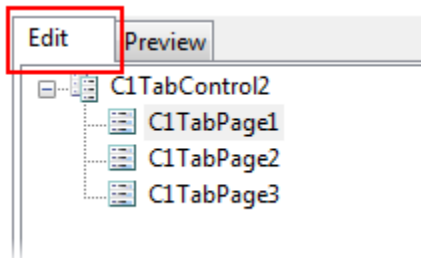
		items.
	Copy	Copies the selected C1TabPage.
	Paste	Pastes a C1TabPage at the specified location in the list of items.
	Delete	Removes the selected C1TabPage.
	Rename	Allows you to change the name of the C1TabPage.

- **C1Tabs Designer Form Toolbar**

The table below describes each button:

Button	Name	Description
	Move Item Up	Moves the selected C1TabPage up the list of items.
	Move Item Down	Moves the selected C1TabPage down the list of items.
	Move Item Left	Moves the selected C1TabPage to the left in the hierarchy.
	Move Item Right	Moves the selected C1TabPage to the right in the hierarchy.
	Add Child Item	Inserts a C1TabPage as a child of the C1Tabs control.
	Insert Item	Inserts a C1TabPage at the specified location in the list of items.
	Cut	Cuts the selected C1TabPage to be moved in the list of items.
	Copy	Copies the selected C1TabPage.
	Paste	Pastes a C1TabPage at the specified location in the list of items.
	Delete	Removes the selected C1TabPage.

- **Edit Tab**

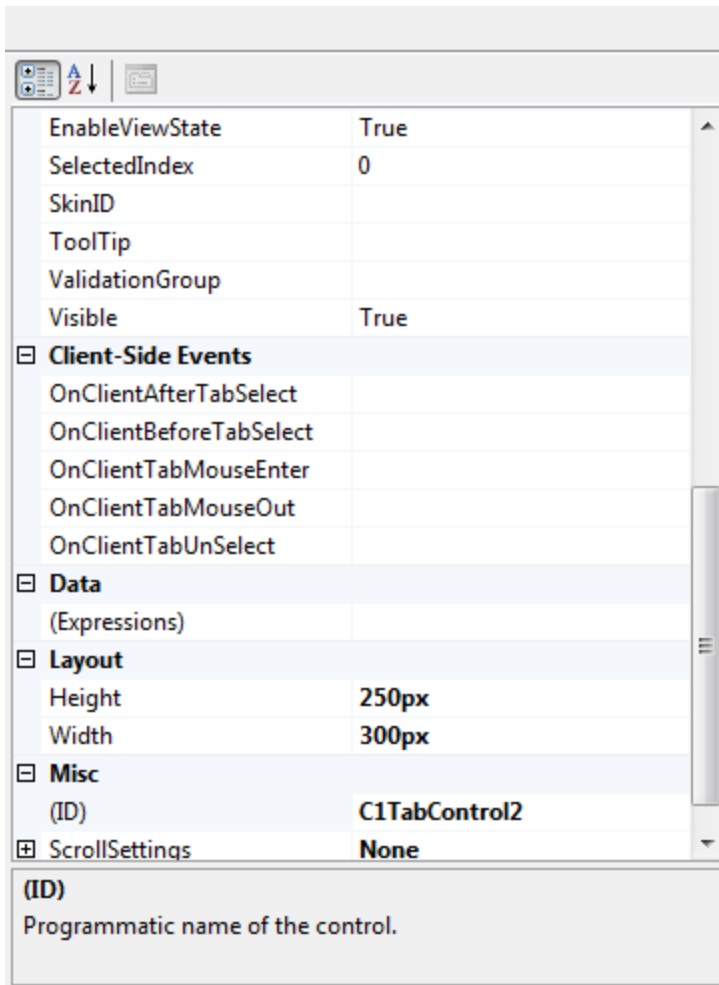


Click the **Edit** tab and select the C1Tabs control or the desired C1TabPage for which you would like to manipulate or adjust the properties.

- **Preview Tab**

Click the **Preview** tab for a preview of what the C1Tabs control will look like.

- **Properties Pane**



The **C1Tabs Designer Form** properties pane is almost identical to the Visual Studio Properties window. Simply select a C1Tabs or C1TabPage and set the desired properties here.

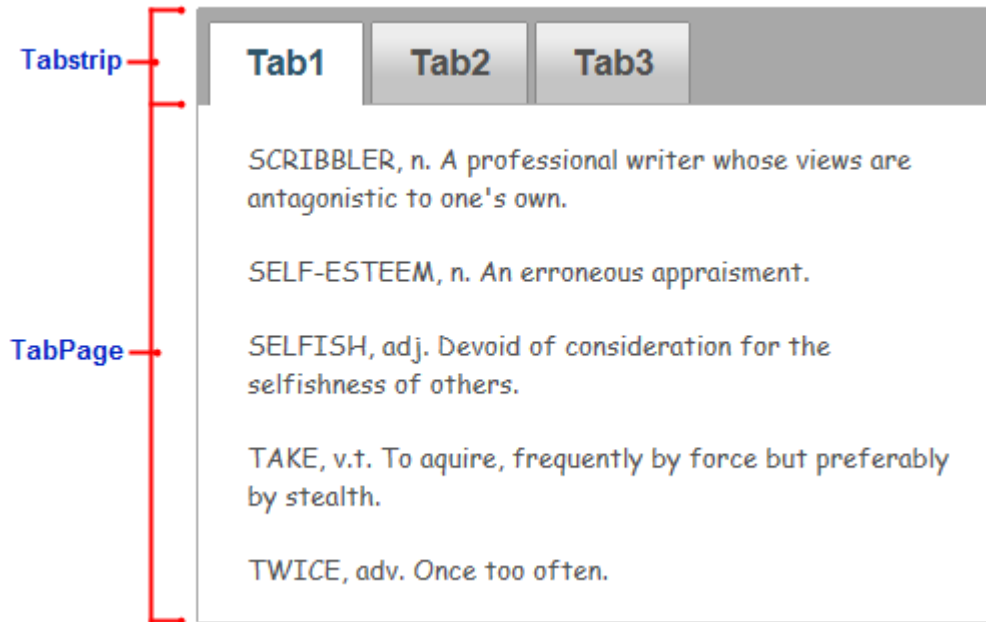
- **Command Buttons**

The command buttons are summarized in the following table:

Button	Description
OK	Clicking OK applies the new settings to the C1Tabs control.
Cancel	Clicking Cancel closes the C1Tabs Designer Form , cancelling the new settings and applying the default settings to the C1Tabs control.

C1Tabs Elements

A **C1Tabs** is essentially a combination of a wizard control and a tab strip. **C1Tabs** is used to display a collection of pages (represented by the `C1TabPage` class) one at a time. Navigation for a **C1Tabs** control is handled by an integrated tabstrip.



Tabstrip

The **C1Tabs** tabstrip is used to navigate through the pages of the control. Each tab is associated with one **C1TabPage**. The tabstrip can be aligned to the top, bottom, left, or right of the control.

Tab Page

The tab page of the **C1Tabs** can hold a variety of elements, such as formatted text, controls, tables, and more. Arbitrary controls can also be added to **C1Tabs** simply by declaring the server control within the **C1TabPage** tag. For example, the following markup adds the `Button` server control inside the first page of the **C1Tabs**:

```
<cc1:C1TabPage ID="C1TabPage01" runat="server">  
  <asp:Button ID="Button1" runat="server" Text="Button" />  
</cc1:C1TabPage>
```

C1Tabs Themes

The **C1Tabs** control contains five built-in themes. When one of these themes is selected, all other ASP.NET Wijmo studio controls on the page will be skinned accordingly. The themes will appear on the **C1Tabs** control as follows:

Arctic

Tab1	Tab2	Tab3
<p>Beef meatloaf jerky shank salami. Tongue cow drumstick, salami pork pig bacon. Beef ribs rump short ribs spare ribs boudin meatloaf. Jowl sausage flank chicken sirloin chuck, pork belly bresaola pig beef cow short ribs drumstick fatback. Sausage tri-tip beef spare ribs chicken hamburger. Flank shankle biltong chuck chicken t-bone. Biltong jerky boudin drumstick, chuck short loin flank t-bone ground round fatback beef short ribs.</p>		

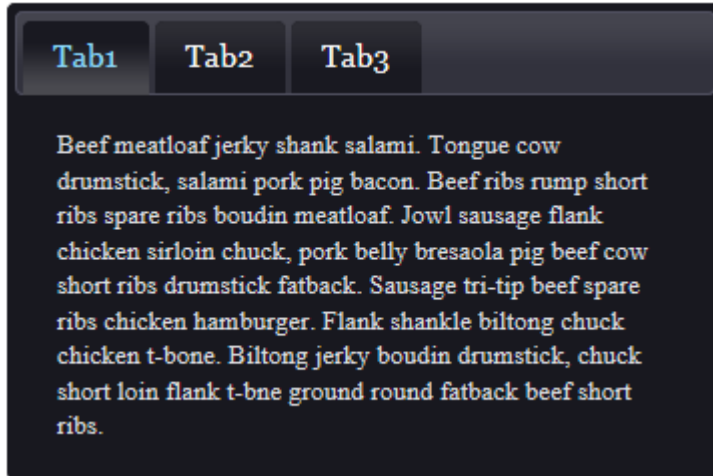
Aristo

Tab1	Tab2	Tab3
<p>Beef meatloaf jerky shank salami. Tongue cow drumstick, salami pork pig bacon. Beef ribs rump short ribs spare ribs boudin meatloaf. Jowl sausage flank chicken sirloin chuck, pork belly bresaola pig beef cow short ribs drumstick fatback. Sausage tri-tip beef spare ribs chicken hamburger. Flank shankle biltong chuck chicken t-bone. Biltong jerky boudin drumstick, chuck short loin flank t-bone ground round fatback beef short ribs.</p>		

Cobalt

Tab1	Tab2	Tab3
<p>Beef meatloaf jerky shank salami. Tongue cow drumstick, salami pork pig bacon. Beef ribs rump short ribs spare ribs boudin meatloaf. Jowl sausage flank chicken sirloin chuck, pork belly bresaola pig beef cow short ribs drumstick fatback. Sausage tri-tip beef spare ribs chicken hamburger. Flank shankle biltong chuck chicken t-bone. Biltong jerky boudin drumstick, chuck short loin flank t-bone ground round fatback beef short ribs.</p>		

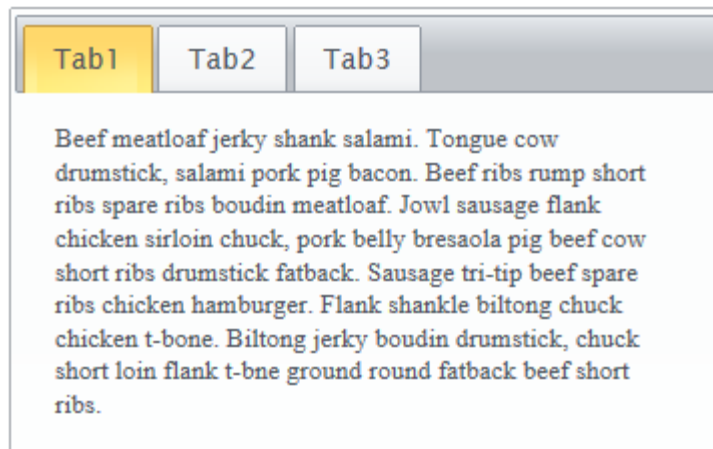
Midnight



Rocket



Sterling


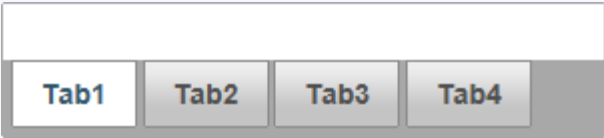
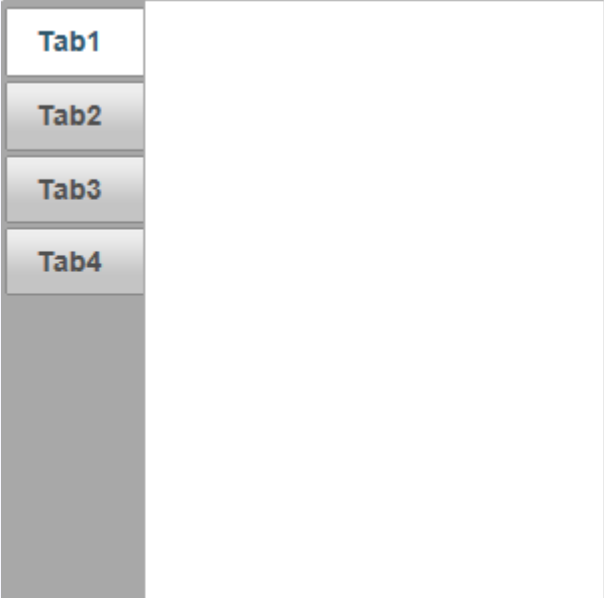


To set the theme of the C1Tabs control, simply set its Theme property to one of the built-in themes.

Tabstrip Direction

The tabstrip element of a **CITabs** can appear in one of four areas of the control: right, left, top, or bottom. The position of the tabstrip is handled by the **Direction** property.

Please see the images below for an illustration of each setting:

Direction	Example
Top	 The image shows a horizontal tabstrip at the top of a control. It contains four tabs labeled 'Tab1', 'Tab2', 'Tab3', and 'Tab4'. 'Tab1' is the active tab, highlighted in white with a dark border. The other tabs are in a grey state. Below the tabs is a large white rectangular area representing the content of the selected tab.
Bottom	 The image shows a horizontal tabstrip at the bottom of a control. It contains four tabs labeled 'Tab1', 'Tab2', 'Tab3', and 'Tab4'. 'Tab1' is the active tab, highlighted in white with a dark border. The other tabs are in a grey state. Above the tabs is a large white rectangular area representing the content of the selected tab.
Left	 The image shows a vertical tabstrip on the left side of a control. It contains four tabs labeled 'Tab1', 'Tab2', 'Tab3', and 'Tab4'. 'Tab1' is the active tab, highlighted in white with a dark border. The other tabs are in a grey state. To the right of the tabs is a large white rectangular area representing the content of the selected tab.



C1Tabs Behavior

The following topics provide information about the C1Tabs control's behavioral features. Some of these features affect how the control acts when loaded, whereas others affect the users' interactions with the control.

Keyboard Access

Tabs for ASP.NET AJAX features keyboard support for the C1Tabs control and each of its tab pages. You can enable this feature for the whole tabstrip by setting the **C1Tabs.AccessKey** property to an access key, or you can enable this feature for individual tabs by setting the **C1TabPage.AccessKey** property to an access key. Once the **AccessKey** property is set, you can access an element by pressing the ALT key and the access key simultaneously on your keyboard.

ToolTips

You can use the **ToolTip** property to create a user-friendly interface. ToolTips are graphic user interface elements that are used to provide users with information or instructions regarding elements of a user interface. When users hover over the interface element with their cursor, a box will appear with the additional information.

ToolTips can be applied to each tab page of a C1Tabs control by setting the **ToolTip** property to a string. If you wanted to set a tooltip for the first tab of the control, you would use the following code:

- Visual Basic

```
TabPage1.ToolTip = "Hello World!"
```

- C#

```
TabPage1.ToolTip = "Hello World!";
```

You can also set the **C1TabPage.ToolTip** property in Design view or in Source view.

Selected Index

The C1Tabs control's tab pages follow a zero-based index, meaning that the index of the first tab page is zero. By default, the Selected property will be set to zero ("0"), and the first tab will be selected at run-time. To change which tab and tab page is selected at run-time, set the Selected property to a different number in the index. For example, if you have four tabs and want the last one to be selected at run-time, you would set the Selected property to 3.

Tabs for ASP.NET Wijmo Task-Based Help

The task-based help section assumes that you are familiar with programming in the Visual Studio ASP.NET environment and have a general understanding of the **Tabs for ASP.NET Wijmo** control.

Each topic provides a solution for specific tasks using the C1Tabs control. By following the steps outlined in each topic, you will be able to create projects using a variety of C1Tabs features.

Each task-based help topic also assumes that you have created a new AJAX-enabled ASP.NET project.

ComponentOne Tabs for ASP.NET Wijmo requires you to create an ASP.NET AJAX-Enabled project so that Microsoft ASP.NET AJAX Extensions and a **ScriptManager** control are included in your project before the C1Tabs control is placed on the page. This allows you to take advantage of ASP.NET AJAX and certain features such as partial-page rendering and client-script functionality of the Microsoft AJAX Library.

Working with Themes


The topics in this section illustrate how to utilize built-in themes and custom themes.

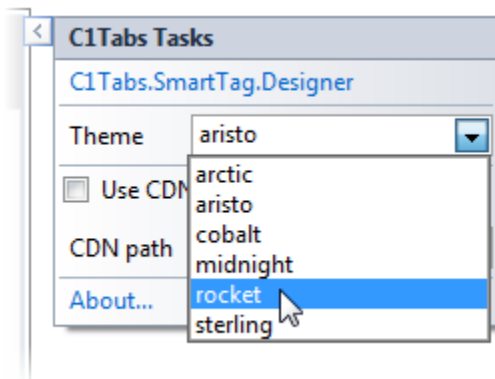
Using a Built-In Theme

A C1Tabs control has six embedded themes that you can apply with just a few clicks. This topic illustrates how to change the theme in Design view, in Source view, and in code. For more information on themes, see [C1Tabs Themes](#) (page 26).

Changing the Theme in Design View

Complete the following steps:

1. Click the **C1Tabs** smart tag () to open the **C1Tabs Tasks** menu.
2. Click the **Theme** drop-down arrow and select a theme from the list. For this example, select **rocket**.



The **rocket** theme is applied to the C1Tabs control.

Changing the Theme in Source View

To change the theme of your **C1Tabs** in Source view, add `VisualStyle="rocket"` to the `<wijmo:C1Tabs>` tag so that it resembles the following:

```
<wijmo:C1Tabs ID="C1Tabs1" runat="server" Theme="rocket"/>
```

Changing the Theme in Code

Complete the following steps:

1. Import the following namespace into your project:

- Visual Basic
`Imports C1.Web.Wijmo.Controls`

- C#
`using C1.Web.Wijmo.Controls;`

2. Add the following code, which sets the Theme property, to the **Page_Load** event:

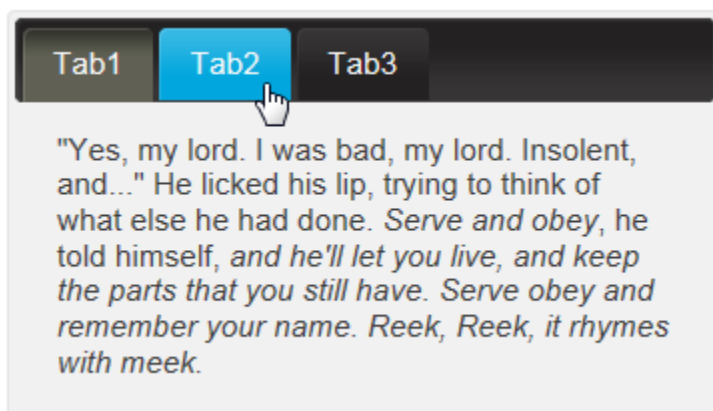
- Visual Basic
`C1Tabs1.Theme = "rocket"`

- C#
`C1Tabs1.Theme = "rocket";`

3. Run the program.

✔ This topic illustrates the following:

The following image shows a **C1Tabs** control with the **rocket** theme:



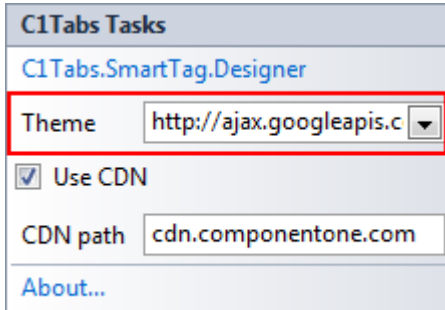
Using a Custom Theme

Tabs for ASP.NET Wijmo provides six built-in themes, but if you prefer to use a different theme, you can choose an existing theme using a CDN URL or create your own custom theme with the jQuery ThemeRoller Web application. We will use C1Tabs in the following examples.

Using a CDN URL

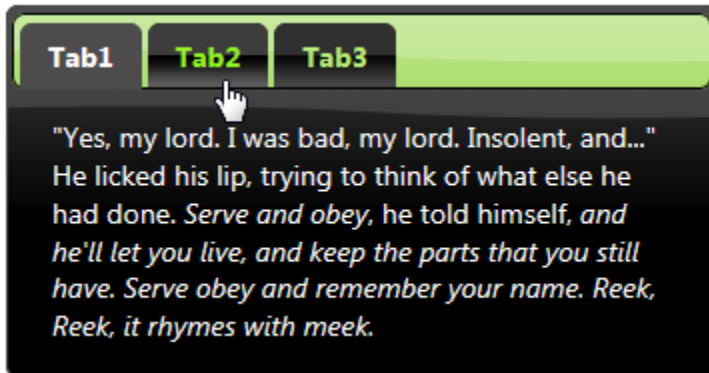
Complete the following steps:

1. Click the C1Tabs smart tag to open the **C1Tabs Tasks** menu.
2. Select the **Use CDN** check box.
3. In the **Theme** property, enter a CDN URL to specify the theme; CDN URLs can be found at <http://blog.jqueryui.com/2011/06/jquery-ui-1-8-14/>. In this example, we'll use the *trontastic* theme: <http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.14/themes/trontastic/jquery-ui.css>.



This theme setting is stored in the `<appSettings>` of the `Web.config` file. In the Solution Explorer, double-click the `Web.config` file. Notice the `<appSettings>` tag contains a `WijmoTheme` key and value; this is where the CDN URL you added is specified.

4. Run the project and notice the theme is applied to C1Tabs.

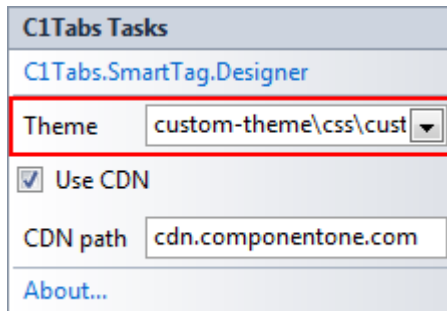


Using jQuery ThemeRoller

Complete the following steps:

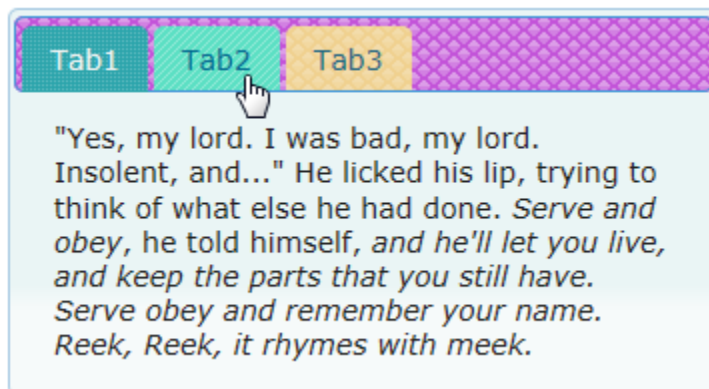
1. Go to <http://jqueryui.com/themeroller/>.
2. On the **Roll Your Own** tab, change the settings to create a custom theme; you can customize fonts, colors, backgrounds, borders, and more. Or click the **Gallery** tab and select an existing theme.
3. Click the **Download** button and then click **Download** again on the **Build Your Download** page.
4. Save and unzip the theme .zip file to a folder within your Visual Studio project folder. In this example, we created a `customtheme` folder.
5. In the Solution Explorer, click **Show All Files** and then right-click the `customtheme` folder and select **Include in Project**.

- Click the C1Tabs smart tag to open the **Tasks** menu.
- Select the **Use CDN** check box.
- In the **Theme** property, enter the path to your custom theme .css; for example, **custom-theme\css\custom-theme/jquery-ui-1.8.15.custom.css**.



This theme setting is stored in the `<appSettings>` of the **Web.config** file. In the Solution Explorer, double-click the **Web.config** file. Notice the `<appSettings>` tag contains a **WijmoTheme** key and value; this is where the custom theme you added is specified.

- Run the project and notice the theme is applied to C1Tabs.



Adding Tab Pages to the C1Tabs Control

This topic illustrates how to add tab pages to a C1Tabs control in Design view, in Source view, and in code.

In Design View

Complete the following steps:

- Click the smart tag to open the **C1Tabs Tasks** menu. Select **C1Tabs.SmartTag.Designer**. The **C1Tabs Designer Form** dialog box opens.
- Click the **Add Child Item** Button to add a C1TabPage to the C1Tabs control. It will appear in the treeview as "Tab1".
- Click **OK** to close the **C1Tabs Design Form** dialog box.

In Source View

Add the following markup between the `<wijmo:C1TabPage>` tags:

```
<wijmo:C1TabPage ID="Tab1" runat="server" Text="Tab1">
</wijmo:C1TabPage>
```

In Code

Complete the following steps:

1. Import the following namespace into your project:

- Visual Basic
`Imports C1.Web.Wijmo.Controls.C1Tabs`

- C#
`using C1.Web.Wijmo.Controls.C1Tabs;`

2. Add the following code to the `Page_Load` event:

- Visual Basic
`Dim TabPage1 As New C1TabPage()
TabPage1.Text = "TabPage1"
C1Tabs1.Controls.Add(TabPage1)`

- C#
`C1TabPage TabPage1 = new C1TabPage();
TabPage1.Text = "TabPage1";
C1Tabs1.Controls.Add(TabPage1);`

Creating a C1Tabs Control in Code

In some instances, you may want to add a `C1Tabs` control to your project in code. In this topic, you will learn how to create a `C1Tabs` control with three `C1TabPage` objects using C# and Visual Basic code.

Complete the following steps:

1. Add a **PlaceHolder** control to your page.
2. In Design view, double-click the page to add a **Page_Load** event to the project and to switch to the code editor.
3. Import the following namespace into your project:

- Visual Basic
`Imports C1.Web.Wijmo.Controls.C1Tabs`

- C#
`using C1.Web.Wijmo.Controls.C1Tabs;`

4. Create the `C1Tabs` object, set its **Width** and **Height**, and then add it to your project by placing the following code in the **Page_Load** event:

- Visual Basic
`Dim NewTabs As C1Tabs = New C1Tabs()
NewTabs.Width = 300
NewTabs.Height = 200
Placeholder1.Controls.Add(NewTabs)`

- C#
`C1Tabs NewTabs = new C1Tabs();
NewTabs.Width = 300;`

```
NewTabs.Height = 200;
Placeholder1.Controls.Add(NewTabs);
```

5. Create three C1TabPage objects and add them to the C1Tabs. This code should also be added to the **Page_Load** event.

- Visual Basic

```
'Create three C1TabPage objects
Dim C1TabPage1 As C1TabPage = New C1TabPage()
Dim C1TabPage2 As C1TabPage = New C1TabPage()
Dim C1TabPage3 As C1TabPage = New C1TabPage()

'Set the TabPages' 'Text' Property
C1TabPage1.Text = "C1TabPage1"
C1TabPage2.Text = "C1TabPage2"
C1TabPage3.Text = "C1TabPage3"

'Add the three C1TabPage objects to the C1Tabs
NewTabs.Controls.Add(C1TabPage1)
NewTabs.Controls.Add(C1TabPage2)
NewTabs.Controls.Add(C1TabPage3)
```

- C#

```
//Create three C1TabPage objects
C1TabPage C1TabPage1 = new C1TabPage();
C1TabPage C1TabPage2 = new C1TabPage();
C1TabPage C1TabPage3 = new C1TabPage();

//Set the TabPages' 'Text' Property'
C1TabPage1.Text = "C1TabPage1";
C1TabPage2.Text = "C1TabPage2";
C1TabPage3.Text = "C1TabPage3";

//Add the three C1Tab objects to the C1Tabs
NewTabs.Controls.Add(C1TabPage1);
NewTabs.Controls.Add(C1TabPage2);
NewTabs.Controls.Add(C1TabPage3);
```

6. Run the program.

 **This Topic Illustrates the Following:**

When your project is run, your C1Tabs control will resemble the following image:



Adding and Manipulating Tab Page Content

A C1Tabs control's tab pages can contain arbitrary controls, display text, and display external content. The next three topics will instruct you on adding content to the pages of the C1Tabs control

Adding Controls to C1Tabs

You can add arbitrary controls to each of your **C1Tabs**'s tab pages using a simple drag-and-drop operation, XHTML, or code. This topic illustrates how to add a standard **Button** control to a tab page. This topic assumes you have added at least one tab page to the control (see [Adding Tab Pages to the C1Tabs Control](#) (page 34)).

In Design View

Complete the following steps:

1. In the designer, select the tab you wish to add the control to. To select a tab, select the C1Tabs control and then click on the tab.
2. Select a **Button** control from the Visual Studio Toolbox and then drag it onto the C1TabPage.

In Source View

Complete the following steps:

1. Locate the `<wijmo:C1TabPage>` tag for the tab page you wish to add the control to and place the following tag between them:
2. Run the program and observe that a **Button** control appears on the tab page.

In Code

Complete the following steps:

1. Create a **Button** control and add text to it by entering the following code to the **Page_Load** event:

- Visual Basic

```
Dim nuButton As Button = New Button()  
nuButton.Text = "Hello World!"
```

- C#

```
Button nuButton = new Button();  
nuButton.Text = "Hello World!";
```

2. Add the **Button** control to the tab page:

- Visual Basic

```
C1TabPage1.Controls.Add(nuButton)
```

- C#

```
C1TabPage1.Controls.Add(nuButton);
```

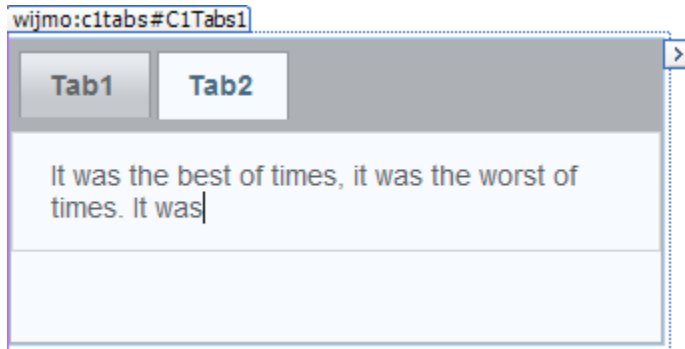
3. Run the program and observe that a **Button** control appears on the tab page.

Adding Text to a C1Tabs Tab Page

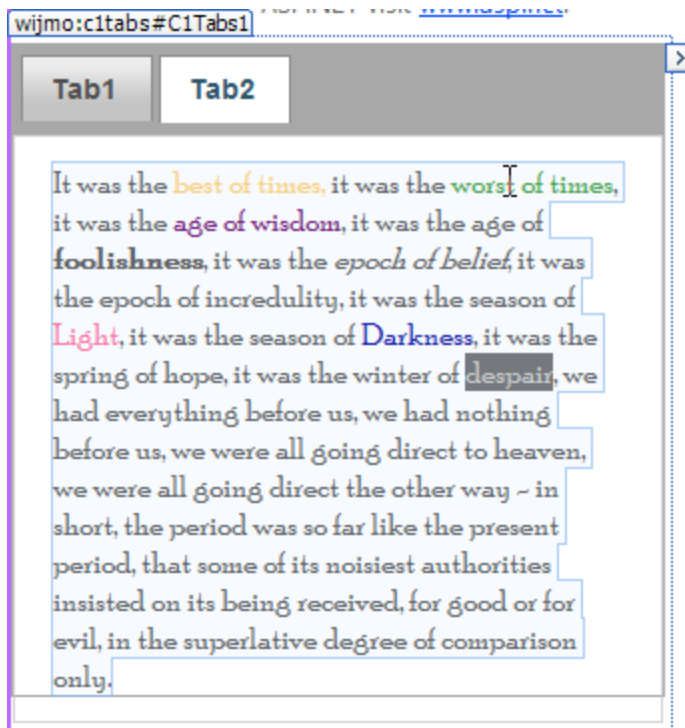
In this topic, you will learn how to add text to a C1Tabs control using the designer and XHTML markup.

In Design View

To add text to a tab page, simply place your cursor inside the tab page and type (or copy) text onto the tab page.



Once you've added text to the tab page, you can use Visual Studio's Formatting toolbar (to view this toolbar, use the following path: **View** | **Toolbars** | **Formatting**) to format the text. The image below shows a C1TabPage with formatted text:



In Source View

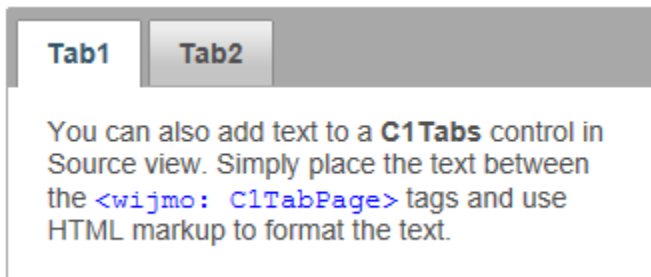
You can add text to a C1TabPage in Source view by placing text between the `<wijmo:C1TabPage>` tags. To format the text, you will use XHTML markup.

To add text to a C1TabPage in Source view, follow these steps:

1. Switch to Source view and paste the following text and XHTML tags between the `<wijmo:C1TabPage>` tags:

```
You can also add text to a <b>C1Tabs</b> control in Source view. Simply place the text between the <span style="color: #0000ff; font-family: Courier New">&lt;wijmo: C1TabPage&gt;</span> tags and use HTML markup to format the text.
```

2. Click the **Design** tab to return to Design view and observe that text has been added to the C1TabPage of the C1Tabs control. Your result will resemble the following image:

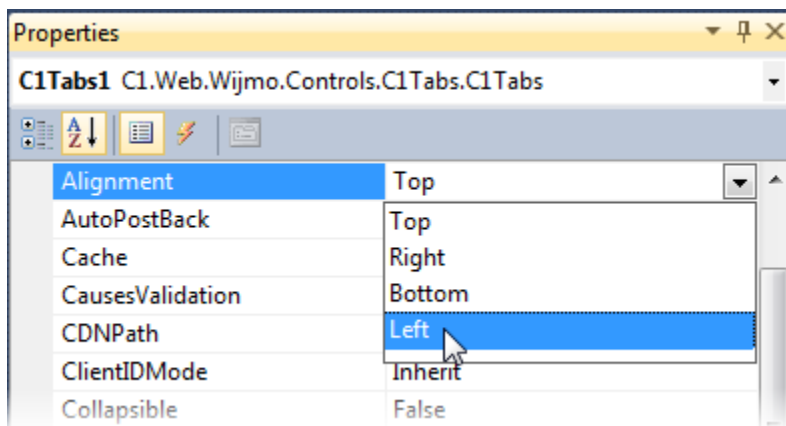


Changing the Alignment

The default orientation of a C1Tabs control is horizontal and located at the top of the control, but you can easily change that orientation to bottom, left, or right by setting the Alignment property. The following instructions explain how to change the orientation of your tabstrip in Design view, in Source view, and in code. For more information, see [Tabstrip Direction](#) (page 29).

In Design View

In the Properties window, make sure that your C1Tabs control is selected, click the Alignment property's dropdown arrow, and select **Top**, **Right**, **Bottom**, or **Left** from the list. For this example, set it to **Left**.



In Source View

In Source view, add `Alignment="Right"` to the `<wijmo:C1Tabs>` tag so that your XHTML resembles the following:

```
<wijmo:C1Tabs ID="C1Tabs1" runat="server" Alignment="Right" >
```

In Code

Complete the following steps:

1. Import the following namespace into your project:

- Visual Basic
`Imports C1.Web.Wijmo.Controls.C1Tabs`

- C#
`using C1.Web.Wijmo.Controls.C1Tabs;`

2. Add the following code to the **Page_Load** event:

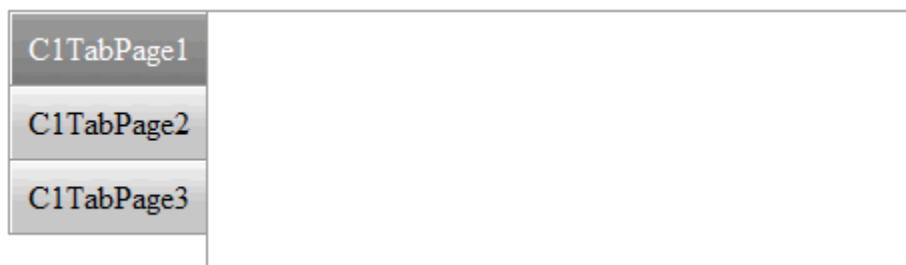
- Visual Basic
`C1Tabs1.Alignment = Alignment.Left`

- C#
`C1Tabs1.Alignment = Alignment.Left;`

3. Run the program.

✔ This Topic Illustrates the Following:

The following image depicts a C1Tabs control that has a tabstrip oriented to the left.



Changing the Selected Index

The Selected property of a C1Tabs control can be used to determine which tab will be selected when your project is run. The following topic shows you how to set this property in Design view, in Source view, and in code.

In Design View

Complete the following steps:

1. Click C1Tabs's smart tag (🔗) to open the **C1Tabs Tasks** menu and select **Tabs Designer**.
The **C1Tabs Designer Form** appears.
2. Use the **Add Child Item** button (🛠️) to add three pages to your C1Tabs.
3. Select C1Tabs from the treeview to reveal its list of properties.

4. Locate the Selected property and set its value to "1".

Note: Observe that the default value of the **SelectedIndex** property is 0. If you had kept this setting, your **C1Tabs** would have loaded with the first tab, **C1TabPage1**, selected.

5. Press **OK** and then run the project. Observe that the second tab page, **C1TabPage2**, is selected at run-time.

In Source View

Add `Selected="1"` to the `<wijmo:C1Tabs>` tag. The resulting XHTML should resemble the following:

```
<wijmo:C1Tabs ID="C1Tabs1" runat="server" Height="29px"
VisualStyle="Office2007Blue"
        VisualStylePath="~/C1WebControls/VisualStyles"
SelectedIndex="1">
```

In Code

Complete the following steps:

1. Add the following code to the **Page_Load** event:

- Visual Basic

```
C1Tabs1.SelectedIndex = 1
```

- C#

```
C1Tabs1.SelectedIndex = 1;
```

2. Press F5 to run the project.

Tabs for ASP.NET Wijmo Client-Side Reference

As part of the amazing [ComponentOne Web stack](#), the Wijmo jQuery UI widgets are optimized for client-side Web development and utilize the power of jQuery for superior performance and ease of use.

The ComponentOne Wijmo website at <http://wijmo.com/widgets/> provides everything you need to know about Wijmo widgets, including demos and samples, documentation, theming examples, support and more.

The client-side documentation provides an overview, sample markup, options, events, and methods for each widget. To get started with client-side Web development for **Tabs for ASP.NET Wijmo**, click one of the external links to view our Wijmo wiki documentation. Note that the **Overview** topic for each of the widgets applies mainly to the widget, not to the server-side ASP.NET Wijmo control.

- <http://wijmo.com/wiki/index.php/Tabs - Options>
- <http://wijmo.com/wiki/index.php/Tabs - Events>
- <http://wijmo.com/wiki/index.php/Tabs - Methods>

Using the Wijmo CDN

You can easily load the client-side Wijmo widgets into your web page using a Content Delivery Network (CDN). CDN makes it quick and easy to use external libraries, and deploy them to your users. A CDN is a network of computers around the world that host content. Ideally, if you're in the United States and you access a webpage using a CDN, you'll get your content from a server based in the US. If you're in India or China, and you access the SAME webpage, the content will come from a server a little closer to your location.

When web browsers load content, they commonly will check to see if they already have a copy of the file cached. By using a CDN, you can benefit from this. If a user had previously visited a site using the same CDN, they will already have a cached version of the files on their machine. Your page will load quicker since it doesn't need to re-download your support content.

Wijmo has had CDN support from the very beginning. You can find the CDN page at <http://wijmo.com/downloads/cdn/>. The markup required for loading Wijmo into your page looks similar to this:

```
<!--jQuery References-->
<script src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-1.7.1.min.js"
type="text/javascript"></script>
<script src="http://ajax.aspnetcdn.com/ajax/jquery.ui/1.8.17/jquery-
ui.min.js" type="text/javascript"></script>
<!--Theme-->
<link href="http://cdn.wijmo.com/themes/rocket/jquery-wijmo.css"
rel="stylesheet" type="text/css" title="rocket-jqueryui" />
<!--Wijmo Widgets CSS-->
<link href="http://cdn.wijmo.com/jquery.wijmo-complete.all.2.0.0.min.css"
rel="stylesheet" type="text/css" />
<!--Wijmo Widgets JavaScript-->
<script src="http://cdn.wijmo.com/jquery.wijmo-open.all.2.0.0.min.js"
type="text/javascript"></script>
<script src="http://cdn.wijmo.com/jquery.wijmo-complete.all.2.0.0.min.js"
type="text/javascript"></script>
```

In this markup, you'll notice that some of the .js files are labeled as *.min.js. These files have been minified - in other words, all unnecessary characters have been removed - to make the pages load faster. You will also notice that there are no references to individual .js files. The JavaScript for all widgets, CSS, and jQuery references have been combined into one file, respectively, such as wijmo-complete.2.0.0.min.js. If you want to link to individual .js files, see the **Dependencies** topic for each widget.

With the **ComponentOne Studio for ASP.NET Wijmo** controls, you can click the **Use CDN** checkbox in the control's **Tasks** menu and specify the **CDN path** if you want to access the client-side widgets.

