

---

ComponentOne

# Splitter for ASP.NET Wijmo

Copyright © 2012 ComponentOne LLC. All rights reserved.

*Corporate Headquarters*

**ComponentOne LLC**

201 South Highland Avenue  
3<sup>rd</sup> Floor  
Pittsburgh, PA 15206 • USA

**Internet:** [info@ComponentOne.com](mailto:info@ComponentOne.com)

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

**Sales**

E-mail: [sales@componentone.com](mailto: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 Splitter 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 .....	12
Namespaces .....	12
Creating an ASP.NET Project .....	13
Adding the Splitter for ASP.NET Wijmo Components to a Project .....	14
Key Features .....	15
Wijmo Top Tips .....	15
Splitter for ASP.NET Wijmo Quick Start .....	16
Step 1 of 4: Adding C1Splitter to the Page .....	16
Step 2 of 4: Changing the Behavior and Appearance of the C1Splitter Control .....	17
Step 3 of 4: Adding Content to the C1Splitter Control .....	18
Step 4 of 4: Manipulating the C1Splitter Control at Run Time .....	19
Design-Time Support .....	21
C1Splitter Smart Tag .....	21
C1Splitter Context Menu .....	22
C1Splitter Elements .....	23

Splitter for ASP.NET Wijmo Appearance and Behavior .....	24
Themes .....	24
Splitter Bar Position .....	26
Splitter Bar Animation Effects .....	26
Animation Effect Descriptions .....	26
Animation Effect Duration .....	28
Panel Layout .....	28
Collapsible and Expandable Panels .....	28
Panel Scrolling .....	29
Panel Previewing .....	29
Split Types .....	30
Horizontal Split .....	30
Vertical Split .....	31
Compound Split .....	31
Full-Size Split .....	32
Splitter for ASP.NET AJAX Task-Based Help .....	33
Adding Content to the Splitter Panels .....	33
Adding Arbitrary Controls to C1Splitter .....	33
Adding Text to a Splitter Panel .....	35
Changing the Appearance of a C1Splitter Control .....	36
Changing the Theme .....	36
Changing the Theme to a Custom Theme .....	38
Changing Splitter Bar Location .....	40
Creating Different Split Types .....	41
Creating a Horizontal Split .....	41
Creating a Nested Split .....	42
Creating a Full-Size Split .....	44
Setting C1Splitter Behaviors .....	45
Setting a Minimum Size for a Splitter Panel .....	45
Setting a Collapsed Splitter Panel .....	46
Using Animation Effects .....	47
Using the Ghost Effect .....	48
Splitter for ASP.NET Wijmo Client-Side Reference .....	50
Using the Wijmo CDN .....	51

# ComponentOne Splitter for ASP.NET Wijmo Overview

Create a professional and polished website with the help of **ComponentOne Splitter™ for ASP.NET Wijmo**. This container control features a movable and collapsible bar that divides a container's display area into two resizable panels. Splitters are able to be nested indefinitely, providing you infinite possibilities in UI design.

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

- [Splitter for ASP.NET Wijmo Quick Start](#) (page 16)
- [Design-Time Support](#) (page 21)
- [C1Splitter Elements](#) (page 23)
- [Task-Based Help](#) (page 33)

## 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:

<b>Bin</b>	Contains copies of all binaries (DLLs, Exes) in the ComponentOne Visual Studio ASP.NET Wijmo package.
<b>Wijmo</b>	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:

<b>HelpViewer</b>	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\<<username>\Documents\ComponentOne Samples

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

<b>Common</b>	Contains support and data files that are used by many of the demo programs.
<b>Studio for ASP.NET Wijmo</b>	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:

<b>Operating Systems:</b>	Windows Server® 2003 Windows Server 2008 Windows XP SP2 Windows Vista™ Windows 7
<b>Web Server:</b>	Microsoft Internet Information Services (IIS) 6.0 or later
<b>Environments:</b>	.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](mailto: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

3<sup>rd</sup> 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 MaskedInput As C1.Web.Wijmo.Controls.C1Input.C1InputMask
```

- C#

```
C1.Web.Wijmo.Controls.C1Input.C1InputMask MaskedInput;
```

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 C1InputMask = C1.Web.UI.Controls.C1Input.C1InputMask  
Imports MyInputMask = MyProject.Objects.C1Input.C1InputMask
```

```
Dim wm1 As C1InputMask
Dim wm2 As MyInputMask
```

- **C#**

```
using C1InputMask = C1.Web.UI.Controls.C1Input.C1InputMask;
using MyInputMask = MyProject.Objects.C1Input.C1InputMask;

C1InputMask wm1;
MyInputMask 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 <a href="#">CodePlex</a> 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 <a href="http://www.asp.net/ajax/downloads/archive/">http://www.asp.net/ajax/downloads/archive/</a> . 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 Splitter 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 **C1Splitter** component 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.C1Splitter. 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.

# Key Features

The following are some of the main features of C1Splitter that you may find useful:

- **Expand and Collapse Panels**

Just set one property to expand or collapse panels. Add visual effects such as images and mouse over styles to the splitter bar to represent a collapsed or expanded panel. See [Collapsible and Expandable Panels](#) (page 28) for more information.

- **Unlimited Nesting**

Organize massive amounts of data into one page using **Splitter for ASP.NET Wijmo's** nesting feature. **Splitter for ASP.NET Wijmo** enables you to nest multiple splits of any orientation type as well accommodate the resizing for the nested splitters when you resize the panel. See [Compound Split](#) for more information.

- **Full Split**

You can create a full-size splitter by setting the **fullSplit** option to **True**. Resize your Web browser and observe how the wijsplitter widget expands or contracts fluidly.

- **Theming**

With just a click of the SmartTag, change the splitter's look by selecting one of the 6 premium themes (Arctic, Midnight, Aristo, Rocket, Cobalt, and Sterling). Optionally, use ThemeRoller from jQuery UI to create a customized theme!

# 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" />
```

# Splitter for ASP.NET Wijmo Quick Start

The goal of this quick start guide is to get you acquainted with **Splitter for ASP.NET Wijmo**. In the first step of this Quick Start guide, you will add a C1Splitter control to your Web project. This quick start guide will also explain how to set common properties of C1Splitter, add arbitrary content to its panels, and how to manipulate the control at run time.

## Step 1 of 4: Adding C1Splitter to the Page

In this step, you will create a Web site project and add a C1Splitter control to it.

Complete the following steps:

1. Begin by creating an ASP.NET AJAX-Enabled Web Site. Note that as you've created an ASP.NET AJAX-Enabled Web Site, a **ScriptManager** control initially appears on the page.
2. While in Design view, navigate to the Visual Studio Toolbox and double-click the **C1Splitter** control to add it to your form.

The **C1Splitter** appears with two empty panes and a splitter bar:



In the next step, you will modify the behavior and appearance of the C1Splitter control.

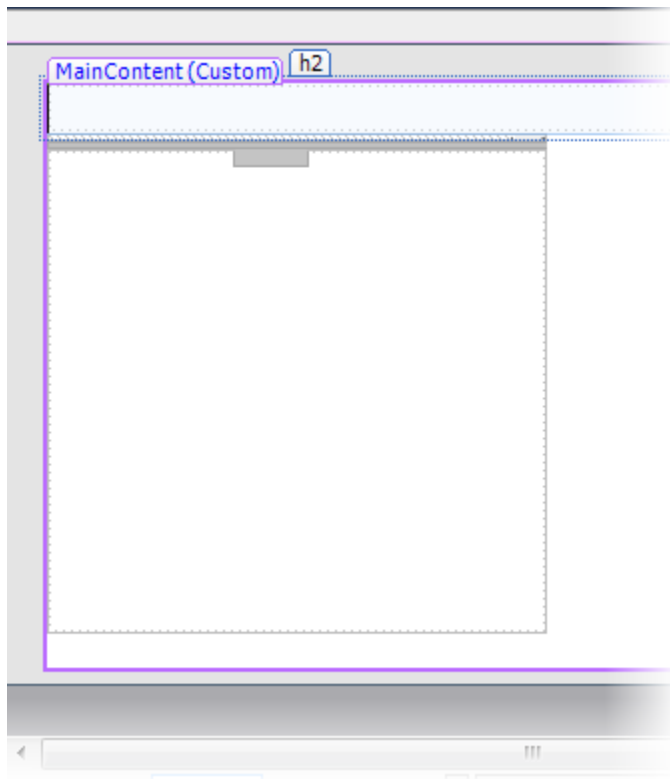
## Step 2 of 4: Changing the Behavior and Appearance of the C1Splitter Control

In this step, you will customize the appearance and behavior of the C1Splitter control.

Complete the following steps:

1. With the C1Splitter control selected on the Web page, set the following properties in the **Properties** window:
  - Set the Width property to "250px".
  - Set the Height property to "250px".
  - Set the SplitterDistance to "75px".
  - Set the Orientation property to **Horizontal**.
2. Expand the **Panel1** node to reveal its list of properties and then set the Collapsed property to **True**; this will set the top panel to be collapsed upon page load.
3. Expand the **Panel2** node to reveal its list of properties and then set the MinSize property to "82". This will prevent the bottom panel from being resized to a height of less than 82 pixels at run time.

The appearance settings will be updated at design time and your project will resemble the following:



In the next step, you'll add content to the C1Splitter control.

## Step 3 of 4: Adding Content to the C1Splitter Control

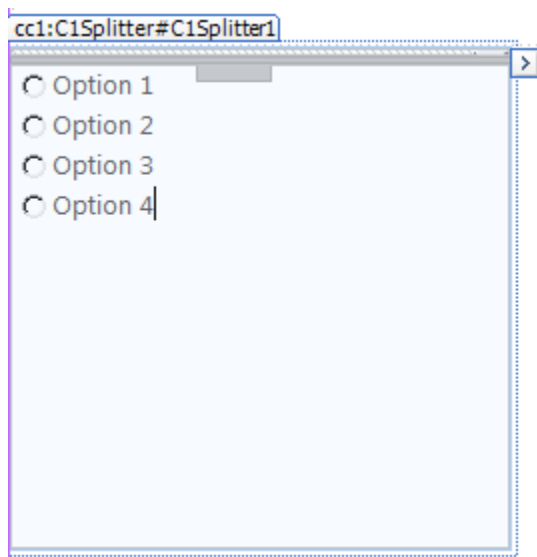
In this step, you will add content to the C1Splitter using both the Designer and markup. You can easily drop controls from the Visual Studio Toolbox into the panels, or you can achieve the same result by switching to Source view and adding the elements in markup code.

Complete the following steps:

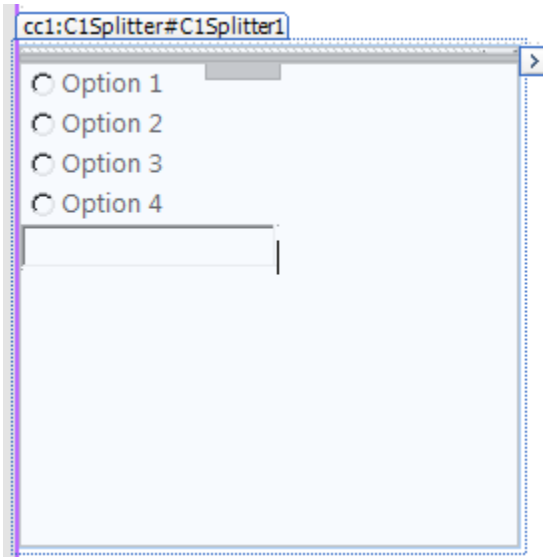
1. Click the **Source** tab to enter Source view and enter the following markup between the `<Panel2>` tags:

```
<ContentTemplate>
  <asp:RadioButton ID="RadioButton1" runat="server" Text="Option 1" />
  <br />
  <asp:RadioButton ID="RadioButton2" runat="server" Text="Option 2" />
  <br />
  <asp:RadioButton ID="RadioButton3" runat="server" Text="Option 3" />
  <br />
  <asp:RadioButton ID="RadioButton4" runat="server" Text="Option 4" />
</ContentTemplate>
```

2. Click the **Design** tab to return to Design view and observe that four radio buttons have been added to **Panel2**. The result resembles the following image:



3. In **Panel2**, place your cursor after **Option 4** and then press ENTER.
4. Navigate to the Visual Studio Toolbox and double-click the **TextBox** icon to add a **TextBox** control to the panel. The result will resemble the following:



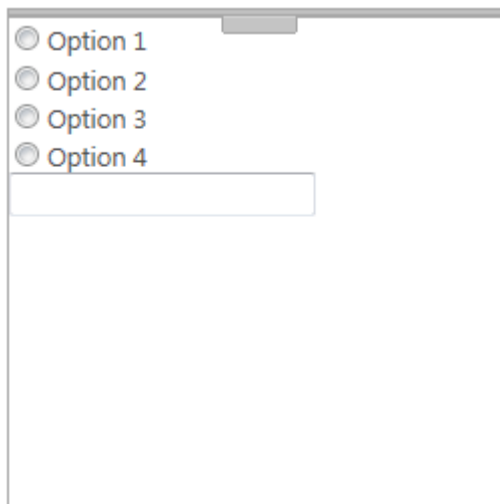
In the next step, you'll run the program and walk through some of the behavioral features of the C1Splitter control.

## Step 4 of 4: Manipulating the C1Splitter Control at Run Time

Now that you've customized the C1Splitter and added content to it, you are going to run the project and observe some of the changes that you've made to the control.

Complete the following steps:

1. Press **F5** to build the project. The **C1Splitter** control appears similar to the image below:

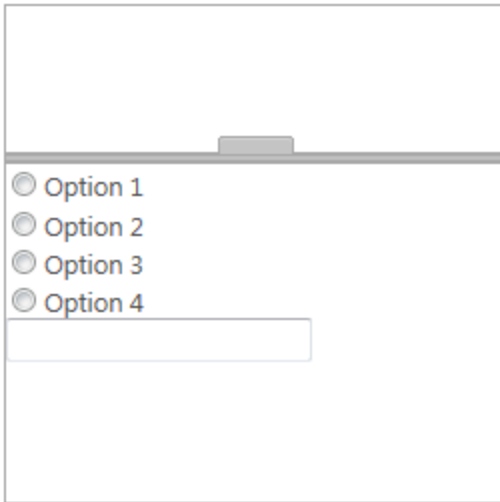


Observe that only **Panel2** appears; **Panel1** is collapsed, which is what you specified when you set the Collapsed property to **True**.

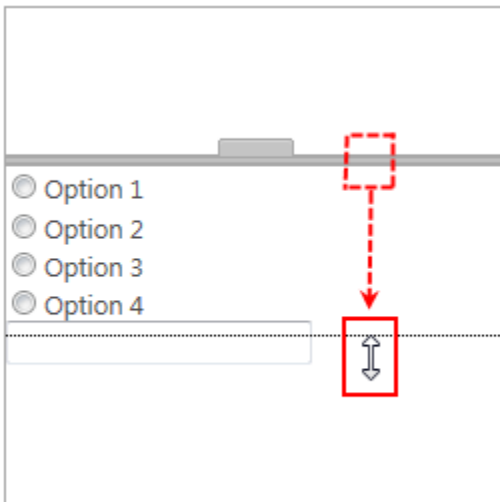
2. Click the expander button.



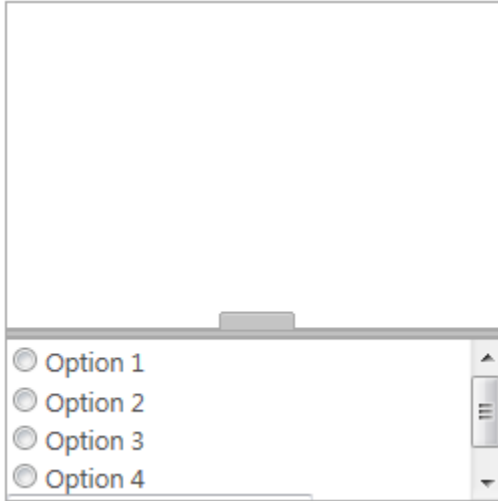
**Panel1** expands into view.



3. Click the splitter bar to activate it and then attempt to drag the splitter bar to the bottom of the control.



As you've probably noticed by now – and as you can see in the image above – you can't expand the bar to the bottom of the control. This is because you set the `MinSize` to **82** in step 2 so that users wouldn't be able to minimize **Panel2** to a height of less than 82 pixels. Thus, upon releasing your cursor, the result will resemble the following image:



### ✔ What You've Accomplished

Congratulations! You have successfully completed the C1Splitter quick start. In this topic, you added a C1Splitter control to your Web page, customized its behavior and appearance, added content to its panels, and manipulated the control at run time.

## Design-Time Support

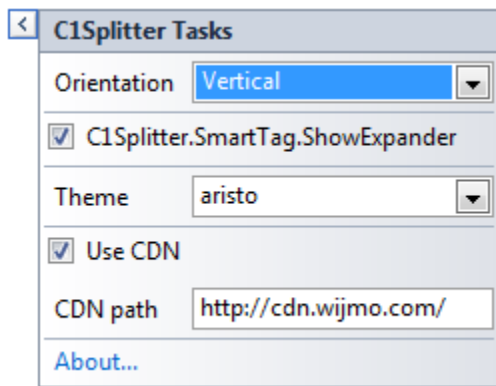
C1Splitter provides customized context menus, smart tags, and a designer that offers rich design-time support and simplifies working with the object model.

The following topics describe how to use **C1Splitter**'s design-time environment to configure **C1Splitter**'s controls.

### C1Splitter Smart Tag

In Visual Studio, each control in **Splitter for ASP.NET Wijmo** includes a smart tag. A smart tag represents a short-cut tasks menu that provides the most commonly used properties in each control.

To access the **C1Splitter Tasks** menu, click the smart tag (☰) in the upper right corner of the **C1Splitter** control. This will open the **C1Splitter Tasks** menu.



The **C1Splitter Tasks** menu operates as follows:

- **Orientation Drop-Down List**

Selecting the **Orientation** drop-down box provides different splitter bar orientations (**Vertical** and **Horizontal**). By default, this is set to **Vertical**.

- **ShowExpander Check Box**

The **ShowExpander** check box allows you to choose whether the splitter bar's expander button is displayed. By default, the **ShowExpander** check box is checked, meaning the button is displayed.

- **Theme Drop-Down List**

You can select one of the five built-in Studio for ASP.NET Wijmo themes from this box to apply the theme to the control.

- **Use CDN check box**

Determines whether the control references the client-side library at a CDN site.

- **CDN Path text box**

The location of the control's client-side library on the CDN.

- **About**

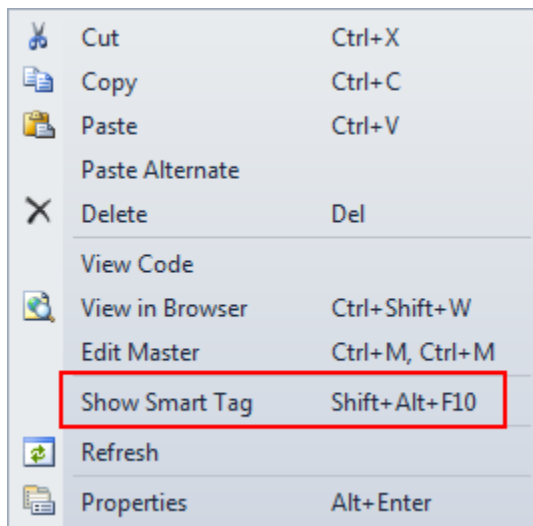
Clicking **About** reveals the About ComponentOne dialog box. This dialog box displays the version number and licensing information for the ComponentOne product.

## C1Splitter Context Menu

**C1Splitter** has designer verbs displayed in the shortcut menu or context menu associated with the **C1Splitter** and **C1UpdateSplitter** controls.

### C1Splitter Context Menu

Right-click anywhere on the **C1Splitter** control to display its context menu.



The following command has been added to this context menu by **Splitter for ASP.NET Wijmo**:

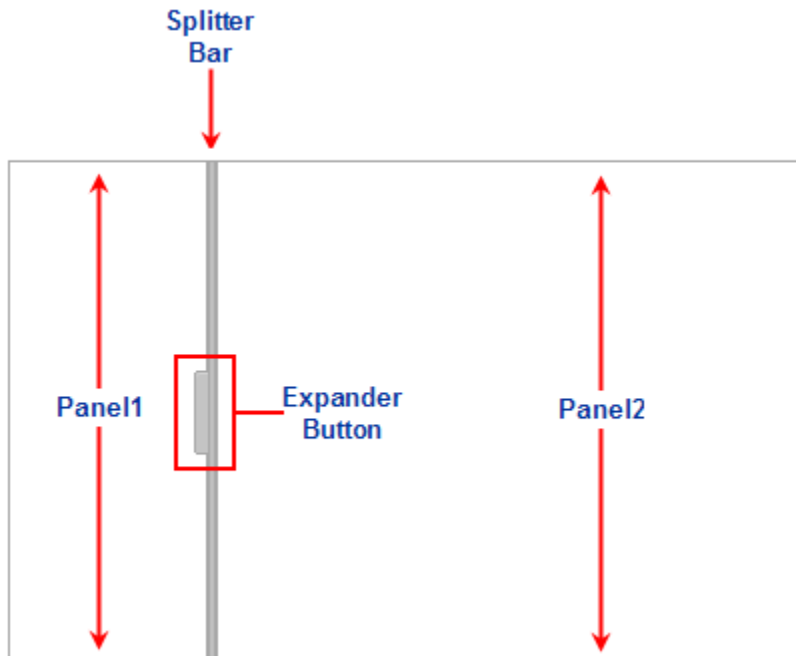
### Show Smart Tag

Click **Show Smart Tag** to open the **C1Splitter Tasks** menu.

# C1Splitter Elements

The C1Splitter container control consists of two basic objects: a SplitterPanel, and an expander button. The SplitterPanel object defines the appearance and behavior for Panel1 and Panel2.

The following image labels the elements of the splitter on a default vertical C1Splitter control:



## Splitter Panels

C1Splitter consists of two panels separated by a splitter bar. The panels appear to the left and right of the splitter bar for vertical splits and top and bottom for horizontal splits. C1Splitter refers to the left/top panel in the designer as Panel1 and the right/bottom panel as Panel2. In the designer, you can control each panel's appearance and behavior through Panel1 and Panel2 properties. Both panels contain the same properties from the SplitterPanel object. You can apply different behaviors and styles to each panel since you can set each panel individually.

To achieve these customizations, you can use any of the properties in the SplitterPanel object:

Property	Description
Collapsed	Gets or sets a value determining whether the panel is collapsed or expanded.
MinSize	Gets or sets the minimum size of a splitter panel.
ScrollBars	Gets or sets the type of scroll bars to display for splitter panel. There are four options: None, Horizontal, Vertical, Both, and Auto.

In the object model, both panels are referred as **SplitterPanel**. The SplitterPanel object contains properties and methods for **Panel1** and **Panel2**.

# Splitter for ASP.NET Wijmo

## Appearance and Behavior

The following section details the appearance and behavior properties used to control the style and behavior of the C1Splitter control.

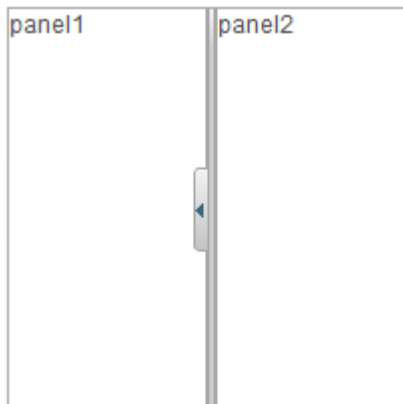
### Themes

The C1Splitter 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 C1Splitter control as follows:

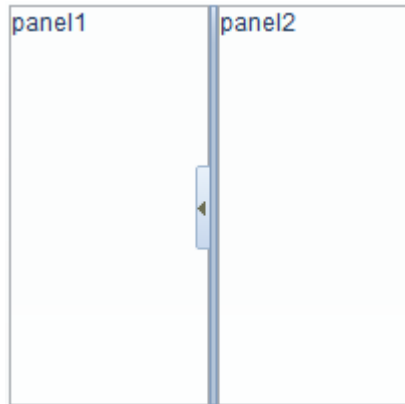
Arctic



Aristo



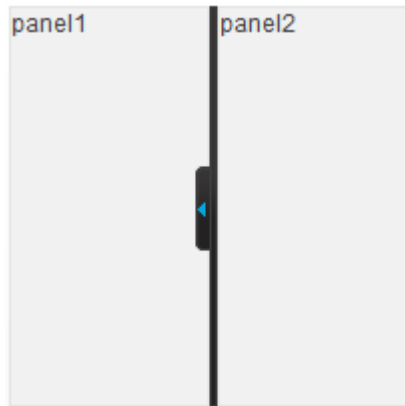
Cobalt



Midnight



Rocket



Sterling



To set the theme of the C1Splitter control, simply set its Theme property to one of the built-in themes.

## Splitter Bar Position

You can determine the location of the splitter, in pixels, from the left or top edge of the C1Splitter through the SplitterDistance property. This property is also useful when determining the splitter bar's position at run time.

## Splitter Bar Animation Effects

The following topics provide information about **Splitter for ASP.NET Wijmo**'s animation and transition effects.

### Animation Effect Descriptions

**C1Splitter** contains thirty-one built-in animation effects that change the reaction of the splitter bar when it is moved. The default transition is **EaseLinear**, but you can set it to another effect using the AnimateEasing property.

The table below describes each animation effect:

Name	Description
FadeIn	Expands body of the control so that it appears to fade in.
FadeOut	Collapses the body of the control, so that it appears to fade out.
ScrollInFromTop	Expands the body of the control, scrolling into view from the top.
ScrollInFromRight	Expands the body of the control, scrolling into view from the right.
ScrollInFromBottom	Expands the body of the control, scrolling into view from the bottom.
ScrollInFromLeft	Expands the body of the control, scrolling into view from the left.
ScrollOutToTop	Collapses the body of the control, scrolling out of view to the top.
ScrollOutToRight	Collapses the body of the control, scrolling out of view to the right.
ScrollOutToBottom	Collapses the body of the control, scrolling out of view to the bottom.
ScrollOutToLeft	Collapses the body of the control, scrolling out of view to the left.
Fold	Collapses the body of control vertically and then horizontally so it appears to unfold.
UnFold	Expands the body of control horizontally and then vertically so it appears to unfold.
OpenVertically	Expands the body of control vertically from the center of the body area.
CloseVertically	Collapses the body of control vertically from the center of the body area.
OpenHorizontally	Expands the body of control horizontally from the center of the body area.
CloseHorizontally	Collapses the body of control horizontally from the center of the body area.
Shake	Expands or Collapses the body of control with a horizontal shaking motion.
Bounce	Expands or Collapses the body of control with a vertical bouncing motion.
DropInFromTop	Expands the body of the control from below the control to the top.
DropInFromRight	Expands the body of the control from the left of the control to the right.
DropInFromBottom	Expands the body of the control from above the control to the bottom.
DropInFromLeft	Expands the body of the control from the right of the control to the left.
DropOutToTop	Collapses the body of the control out to above the control.

DropOutToRight	Collapses the body of the control out to the right of the control.
DropOutToBottom	Collapses the body of the control out to below the control.
DropOutToLeft	Collapses the body of the control out to the left of the control.

## Animation Effect Duration

You can set the length of **C1Splitter**'s animation effect takes using the **Duration** property. The unit of time used for specifying animation effect duration is in milliseconds, and the default setting for the **Duration** property is **500** milliseconds (or half a second). Increase this value for longer animation effect, and decrease this number for a shorter animation effect.

## Panel Layout

C1Splitter's panels show WYSIWYG views so that you can view the result without having to run the project. With the WYSIWYG designer interface C1Splitter supports, it is simple to arrange child controls in the containers of the panels because C1Splitter displays them as it would at run time.

You can add as many child controls to each panel by dragging and dropping each control into the desired panel. When you add a child control to the C1Splitter, it places the child control in the top left corner of the panel by default.

Panels can be selected on the Web form by clicking anywhere inside the respective panel's rectangular box.

## Collapsible and Expandable Panels

You can use the Collapsed property to specify a collapsed or expanded panel. The panels in C1Splitter can easily be collapsed or expanded by setting their respective Collapsed properties to **True** or **False**. At run time, the panel can be expanded by clicking on the expander button.

Note: Only one panel can be collapsed at a time.

The following image illustrates Panel1's Collapsed property set to **True**:



## Panel Scrolling

C1Splitter provides different types of scrollbars to use for the panels. You can specify whether you would like horizontal or vertical scrollbars to appear on the panel through the ScrollBars property. The ScrollBars property includes the following values for you to choose from: **None**, **Horizontal**, **Vertical**, **Both**, and **Auto**. **Auto**, the default setting for the ScrollBars property, enables C1Splitter to automatically add vertical and/or horizontal scrollbars when the content information is larger than the panel's size. If you prefer not to use scrollbars, then you can set both panels' ScrollBars property to **None**.

Each panel may use a different type of scrollbar. This is beneficial when each panel contains different controls that take up more horizontal space and vice-versa in the panel. This also gives you more flexibility in how you set up the layout in your splitter panels.

The following image illustrates a **C1Splitter** with a vertical scrollbar in the first panel and a horizontal and vertical scrollbar in the second.



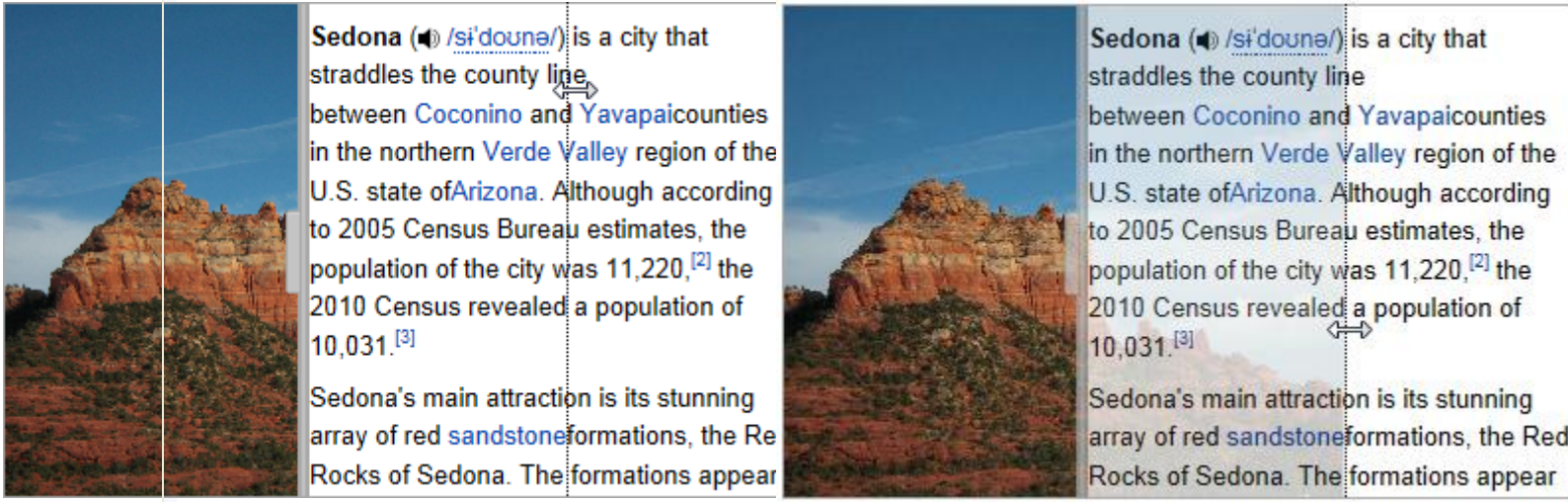
## Panel Previewing

C1Splitter contains a property, Ghost, which determines whether or not a preview of a panel's contents will be displayed when users drag the splitter bar to resize one of the panels. When Ghost is set to **True**, users will see a translucent preview of a panel's content while they are moving the splitter bar. When Ghost is set to **False**, users will only see a dotted line indicating the placement of the splitter bar; the content of the panel will not be revealed until after the splitter bar is released. The Ghost property is set to **False** by default.

The table below illustrates the two settings of the Ghost property:

Ghost = False

Ghost = True



## Split Types

The default **CISplitter** has a simple vertical layout. The vertical layout has a left and right panel separated by a divider. The divider is referred to as the bar.

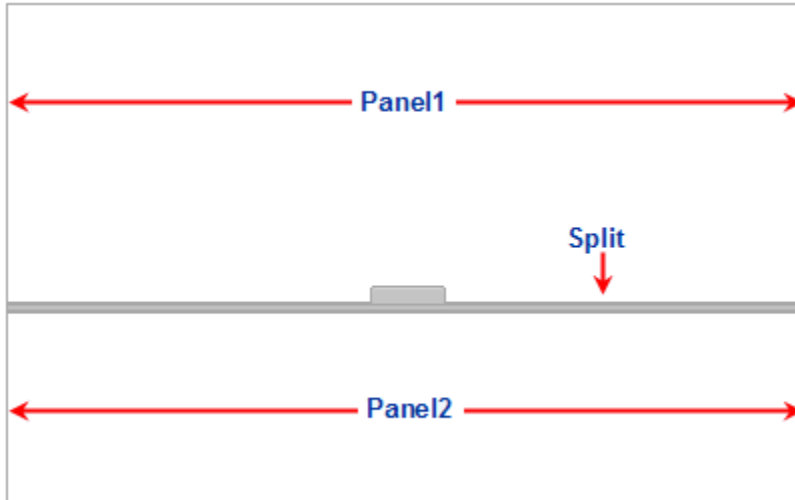
**CISplitter** contains four basic types of splits:

- Horizontal Split
- Vertical Split
- Compound Split
- Full Split

### Horizontal Split

A horizontal split divides the panels into two or more rows and is represented in the Web page by one or more **CISplitter** bars.

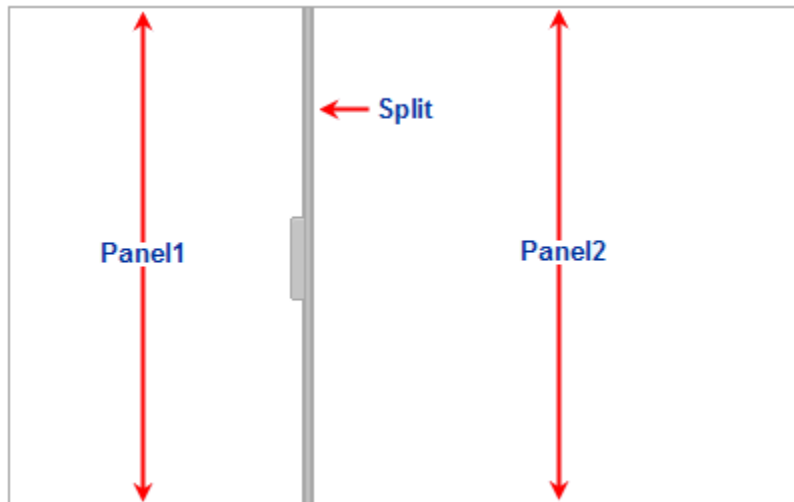
The following image illustrates a horizontal split:



## Vertical Split

A vertical split is the default split type for C1Splitter. It divides the panels into two or more columns and is represented in the Web page by one or more C1Splitter bars.

The following image illustrates a vertical split:

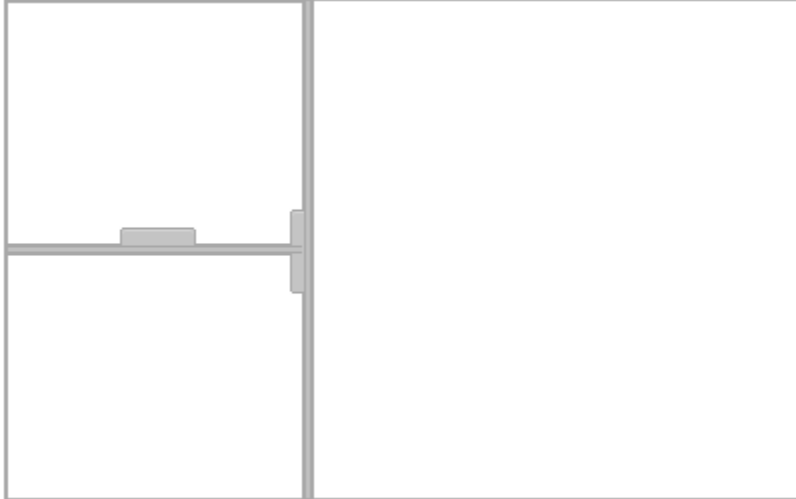


## Compound Split

A compound split is a nested split, meaning that the initial C1Splitter control contains one or more C1Splitter controls. A compound split can contain two or more vertical splitters, two or more horizontal splitters, or a combination of vertical and horizontal splitters.

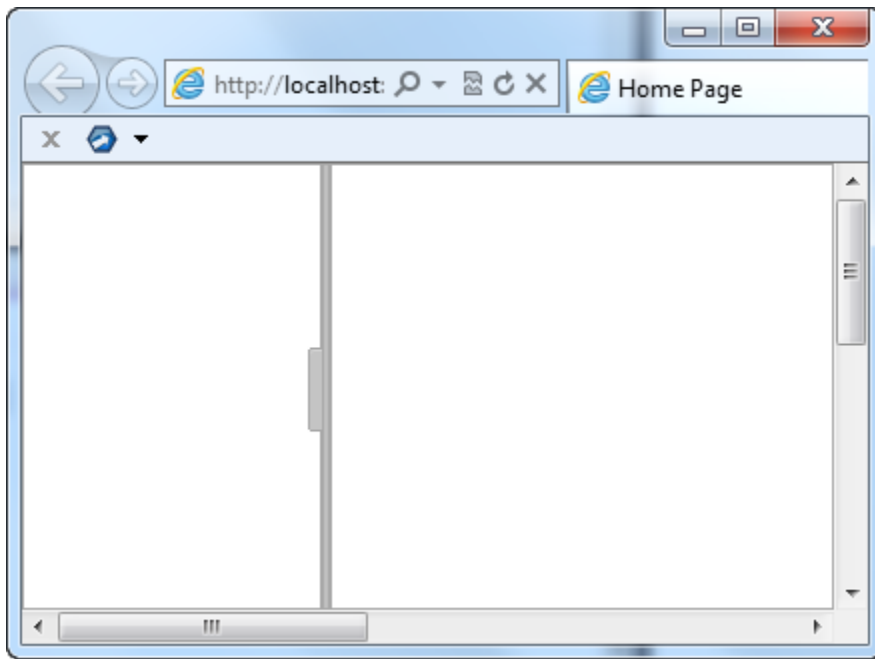
A compound split can be created by directly dropping a child C1Splitter into the panel of the parent C1Splitter.

The most common usage of a compound splitter is full-cover nesting. To create full cover nesting, set the nested splitter control's width or height to 100% to make it fully extend to the size of the parent's panel. The following image illustrates a full-cover nesting in a compound splitter:



## Full-Size Split

A full-size split is a horizontal or vertical split that stretches to fill the content area of a Web browser. The following image illustrates a full-size splitter in an Internet Explorer browser:



To create a full-size split, set the **CISplitter.FullSplit** property to **True**.

# Splitter for ASP.NET AJAX 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 **ComponentOne Splitter** control.

Each topic provides a solution for specific tasks using the C1Splitter control. By following the steps outlined in each topic, you will be able to create projects using a variety of **C1Splitter** features.

This task-based help section assumes that you have created a new AJAX-enabled ASP.NET project.

**ComponentOne Splitter for ASP.NET AJAX** 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 C1Splitter 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.

## Adding Content to the Splitter Panels

A C1Splitter control can hold arbitrary controls or display text. The following topics will instruct you on adding content to the pages of your C1Splitter control.

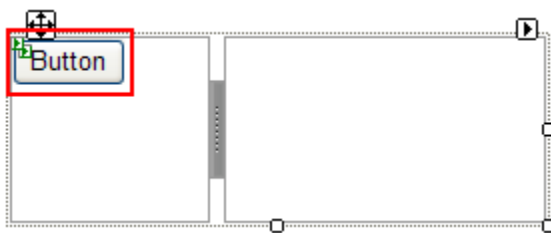
### Adding Arbitrary Controls to C1Splitter

You can add arbitrary controls to each panel of the C1Splitter control using a simple drag-and-drop operation or HTML. In this topic, you will add a **Button** control to **Panel1** and a **TextBox** control to **Panel2**.

#### In Design View

Complete the following steps:

1. Add a C1Splitter control to your Web project.
2. Select a **Button** control from the Visual Studio Toolbox and drag it into the **Panel1**.



3. Select a **TextBox** control from the Visual Studio Toolbox and drag it into **Panel2**.



## In Source View

Complete the following steps:

1. Add a C1Splitter control to your Web project.
2. Click the **Source** tab to enter Source view.
3. Locate the `<Panel1>` tags and place the following tag between them:

```
<ContentTemplate>
    <asp:Button ID="Button1" runat="server" Text="Button" />
</ContentTemplate>
```

4. Locate the `<Panel2>` tags and place the following tag between them:

```
<ContentTemplate>
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
</ContentTemplate>
```

## In Code

Complete the following steps:

1. Add a C1Splitter control to your Web project.
2. Select a **PlaceHolder** control from the Visual Studio Toolbox and drag it into the **Panel1**.  
**PlaceHolder1** appears in **Panel1**.
3. Select a **PlaceHolder** control from the Visual Studio Toolbox and drag it into the **Panel2**.  
**PlaceHolder2** appears in **Panel2**.
4. In the Solution Explorer window, right-click on the project and select **View Code** to enter the code editor.
5. 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!";
```

6. Create a **TextBox** control:

- Visual Basic

```
Dim nuTextBox As TextBox = New TextBox()
```

- C#

```
TextBox nuTextBox = new TextBox();
```

7. Add the **Button** control to the **PlaceHolder1**:

- Visual Basic

```
Placeholder1.Controls.Add (nuButton)
```

- C#

```
Placeholder1.Controls.Add (nuButton) ;
```

8. Add the **TextBox** control to **Placeholder2**:

- Visual Basic

```
Placeholder2.Controls.Add (nuTextBox)
```

- C#

```
Placeholder2.Controls.Add (nuTextBox) ;
```

9. Run the program.

### ✔ This Topic Illustrates the Following:

The following graphic depicts a **C1Splitter** control with a **Button** control in **Panel1** and a **TextBox** control in **Panel2**.

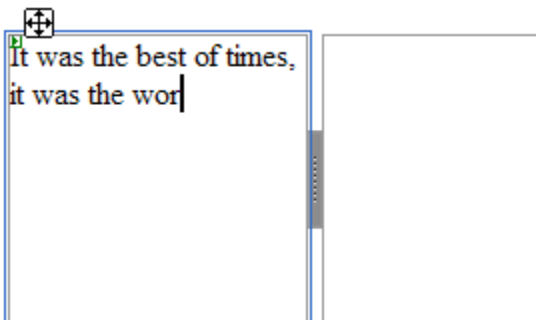


### Adding Text to a Splitter Panel

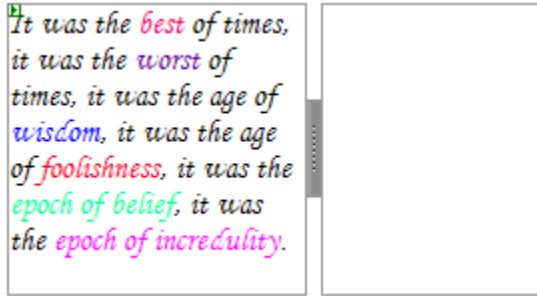
In this topic, you will learn how to add text to a **C1Splitter** control using the designer and HTML markup.

#### In Design View

To add text to a panel, simply place your cursor inside the panel and type (or copy) the text into the panel.



Once you've added text to the 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 features a **C1PageView** with formatted text:



### In Source View

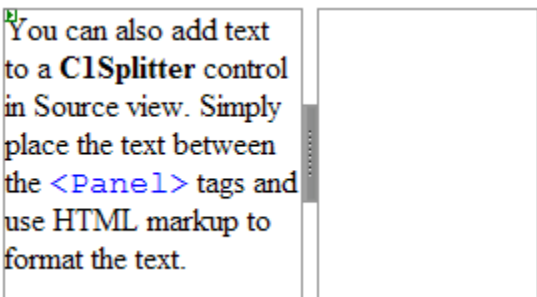
You can add text to a C1Splitter panel in Source view by placing text between the `<Panel1>` or `<Panel2>` tags. To format the text, you would use HTML markup.

Complete the following steps:

1. Add a C1Splitter to your project.
2. Switch to Source view and paste the following markup and text between the `<Panel1>` tags:

```
<ContentTemplate>
You can also add text to a <b>C1Splitter</b> control in Source
view. Simply place the text between the <span style="color:
#0000ff; font-family: Courier New">&lt;Panel&gt;</span> tags and
use HTML markup to format the text.
</ContentTemplate>
```

3. Click the **Design** tab to enter Design view and observe that text has been added to **Panel1** of your C1Splitter control. The result will resemble the following image:



## Changing the Appearance of a C1Splitter Control

The following topics detail how to modify the appearance of a C1Splitter control using the Designer, HTML, and code.

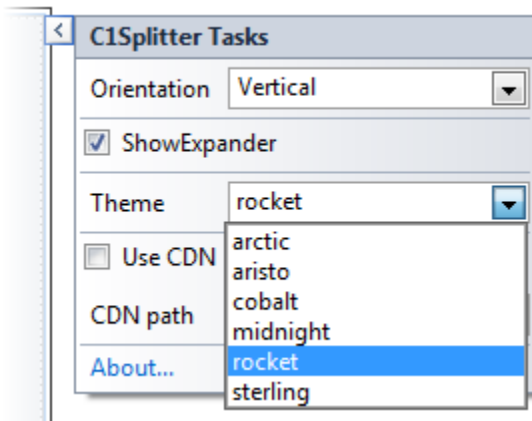
### Changing the Theme

A C1Splitter 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 [Themes](#) (page 24).

### Changing the Theme in Design View

Complete the following steps:

1. Click the **C1Splitter** smart tag (🔗) to open the **C1Splitter 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 C1Splitter control.

### Changing the Theme in Source View

To change the theme of your **C1Splitter** in Source view, add `Theme="rocket"` to the `<wijmo:C1Splitter>` tag so that it resembles the following:

```
<wijmo:C1Splitter ID="C1Splitter1" 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  
`C1Splitter1.Theme = "rocket"`
  - C#  
`C1Splitter1.Theme = "rocket";`
3. Run the program.

### ✔ This topic illustrates the following:

The following image shows a **C1Splitter** control with the **rocket** theme:



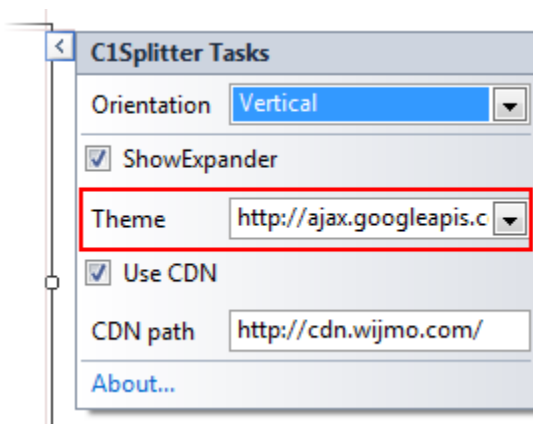
## Changing the Theme to a Custom Theme

**Splitter 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 C1Splitter in the following examples.

### Using a CDN URL

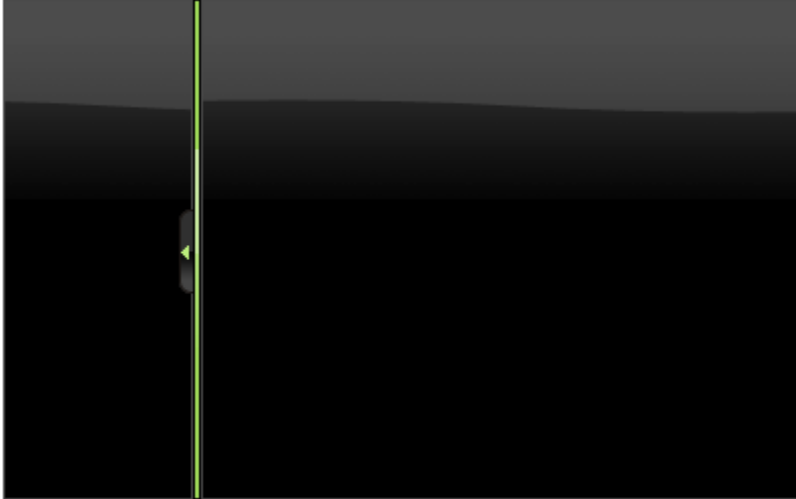
Complete the following steps:

1. Click the C1Splitter smart tag to open the **Tasks** menu.
2. 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.

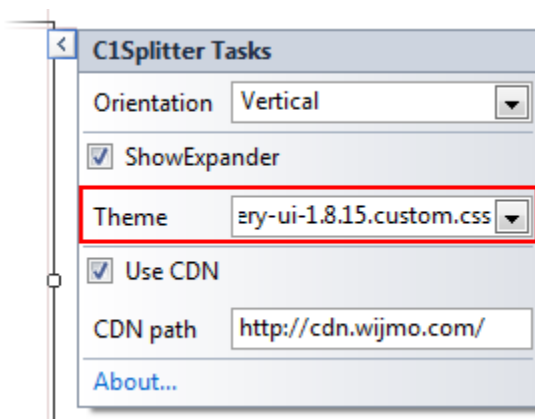
3. Run the project and notice the theme is applied to C1Splitter.



### Using jQuery ThemeRoller

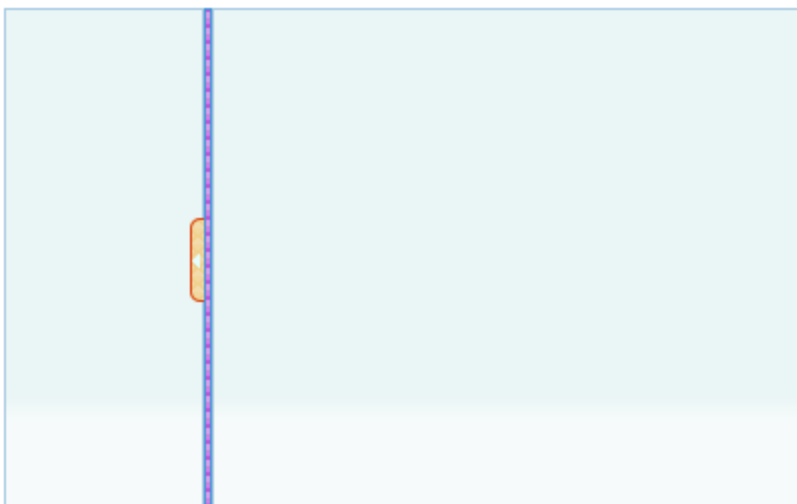
Complete the following steps:

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**.
6. Click the C1Splitter smart tag to open the **Tasks** menu.
7. 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.

8. Run the project and notice the theme is applied to C1Splitter.



## Changing Splitter Bar Location

The default location of the splitter bar is 100 pixels from the left for a vertical split and 100 pixels from the top for a horizontal split. You can adjust the initial location of the splitter bar using the `SplitterDistance` property. In this topic, you will learn how to set the `SplitterDistance` property in Design view, in Source view, and in code.

### In Design View

Complete the following steps:

1. Add **C1Splitter** to the Web form.
2. Right-click on the control and select **Properties**.
3. In the Properties window, locate the `SplitterDistance` property and specify a number to represent the location of the splitter bar from the left edge of the splitter bar. For this example, we'll set it to "250".
4. Run the program.

### In Source View

To set the splitter bar location, place `SplitterDistance="250"` within the `<wijmo:C1Splitter>` tag. Once the `SplitterDistance` property has been set, the markup will resemble the following:

```
<wijmo:C1Splitter ID="C1Splitter1" runat="server" SplitterDistance="250">
```

### In Code

Complete the following steps:

1. Import the following namespace into your project:

- Visual Basic

```
Imports C1.Web.Wijmo.Controls.C1Splitter
```

- C#

```
using C1.Web.Wijmo.Controls.C1Splitter;
```

2. Add the following code to the **Page\_Load** event to set the `SplitterDistance` property:

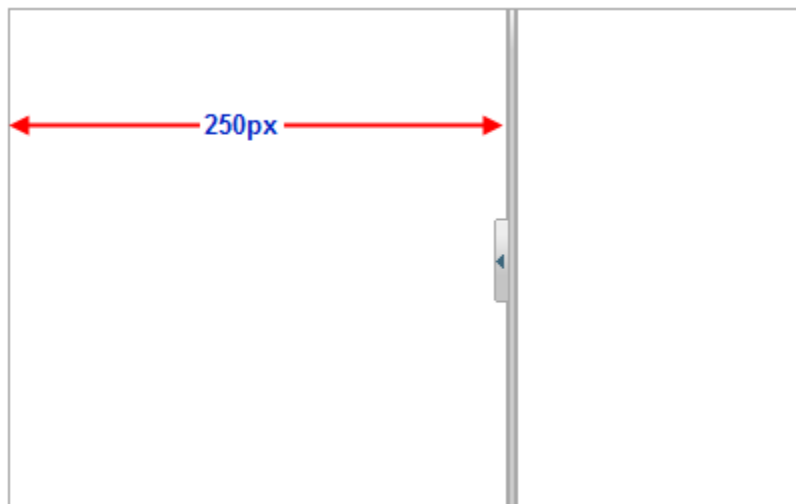
- Visual Basic  
`C1Splitter1.SplitterDistance = 250`

- C#  
`C1Splitter1.SplitterDistance = 250;`

3. Run the program.

✔ **This Topic Illustrates the Following:**

The following image depicts a C1Splitter with a splitter set 50 pixels from the left side of the control:



## Creating Different Split Types

There are four types of splits that can be created with the **C1Splitter** control: horizontal split, vertical split, compound split, and full-size split. This section contains procedures for creating each type of split.

For more information on split types, see [Split Types](#) (page 30).

### Creating a Horizontal Split

Creating a horizontal split is as simple as setting one property. In this topic, you'll learn how to set the Orientation property in Design view, Source view, and in code.

For more information on horizontal splits, see [Horizontal Split](#) (page 30).

#### In Design View

Complete the following steps:

1. Add the C1Splitter control to the form.
2. In the Properties window, set the splitter's Orientation to **Horizontal**.

#### In Source View

To create a horizontal split, place `Orientation="Horizontal"` within the `<wijmo:C1Splitter>` tag. Once the Orientation property has been set, the markup will resemble the following:

```
<wijmo:C1Splitter ID="C1Splitter1" runat="server" Height="212px"
Orientation="Horizontal"
Width="221px">
```

## In Code

Complete the following steps:

1. Import the following namespaces into your project:

- Visual Basic

```
Imports C1.Web.Wijmo.Controls.C1Splitter
```

- C#

```
using C1.Web.Wijmo.Controls.C1Splitter;
```

2. Add the following code, which sets the Orientation property, to the **Page\_Load** event:

- Visual Basic

```
C1Splitter1.Orientation = C1.Web.Wijmo.Controls.Orientation.Horizontal
```

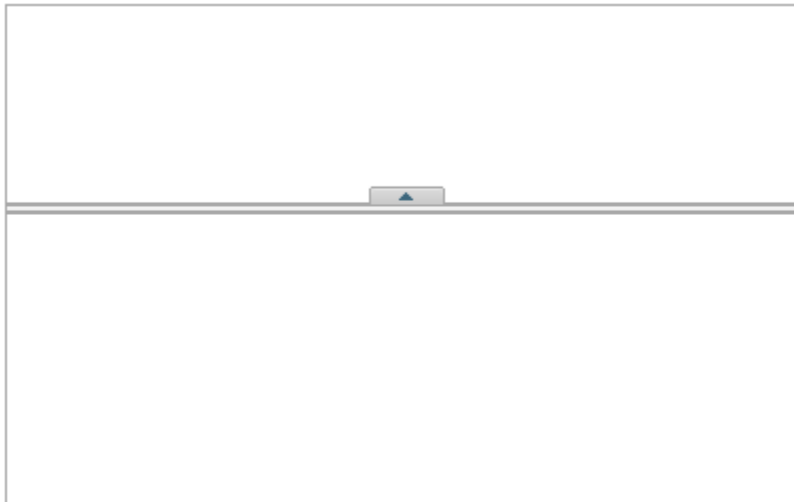
- C#

```
C1Splitter1.Orientation = C1.Web.Wijmo.Controls.Orientation.Horizontal;
```

3. Run the program.

### ✔ This Topic Illustrates the Following:

The splitter bar is now horizontal. The result of this topic will resemble the following image:



## Creating a Nested Split

You can use combine multiple C1Splitter objects to create nested splits. In this topic, you will learn how to nest a horizontal split within the first panel of a vertical split.

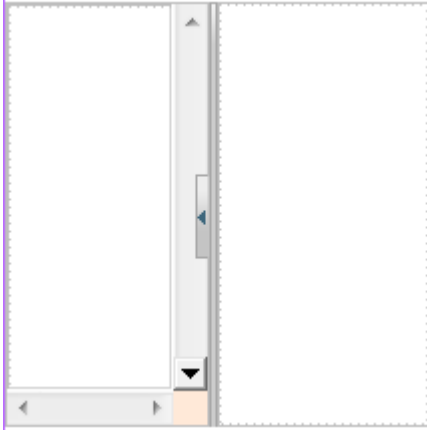
For more information on compound splits, see [Compound Split](#) (page 31).

Complete the following steps:

1. Add the C1Splitter control to the form.

The default vertical split layout appears for **C1Splitter**.

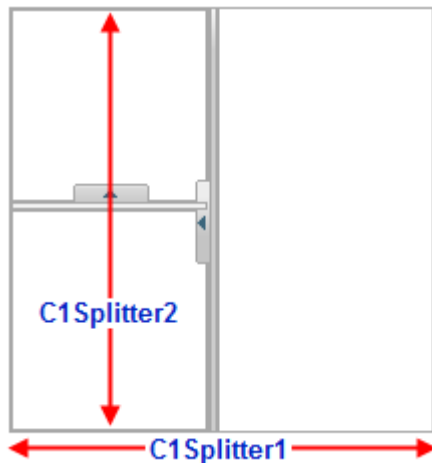
2. Select **C1Splitter1**, navigate to the Properties window, and set both the Height and Width properties to "212".
3. Drag another **C1Splitter** control from the Visual Studio Toolbox and drop it into the right panel (**Panel1**) of **C1Splitter1**. **C1Splitter2** is added to right panel. Observe that vertical and horizontal scrollbars appear to accommodate for the large size of the control.



4. Now you will need to resize **C1Splitter2** so that it fits in the right panel of **C1Splitter1**. You can do this using either of the following techniques:
  - Select **C1Splitter2** and navigate to the Properties window. Adjust the Width property to "100" and its Height property to "210".
  - OR
  - Starting from the bottom-right corner of **C1Splitter2**, drag **C1Splitter2**'s container leftward and upward until it fits snugly into the left panel (**Panel1**) of **C1Splitter1**.
5. Once **C1Splitter2** is positioned correctly in the right panel, set its Orientation property to **Horizontal**.

✔ **This Topic Illustrates the Following:**

The following image displays **C1Splitter1** nested within the left panel of **C1Splitter2**:



## Creating a Full-Size Split

A full-size split is a horizontal or vertical split that stretches to fill the content area of a Web browser. You can create a full-size split by setting one property: `FullSplit`. In this topic, you'll learn how to set the `FullSplit` property in Design view, Source view, and in code.

For more information on full-size splits, see [Full-Size Split](#) (page 32).

### In Design View

Complete the following steps:

1. Add the `C1Splitter` control to the form.
2. In the Properties window, set the splitter's `FullSplit` property to **True**.
3. Run the program and observe that the control expands to the width and height of your Web browser.

### In Source View

Complete the following steps:

1. Click the Source button to enter Source view.
2. Place `FullSize="True"` within the `<wijmo:C1Splitter>` tag so that the markup resembles the following:

```
<wijmo:C1Splitter ID="C1Splitter1" runat="server" Height="212px"
FullSize="True"
Width="221px"
```
3. Run the program and observe that the control expands to the width and height of your Web browser.

### In Code

Complete the following steps:

1. Import the following namespace into your project:
  - Visual Basic

```
Imports C1.Web.Wijmo.Controls.C1Splitter
```
  - C#

```
using C1.Web.Wijmo.Controls.C1Splitter;
```
2. Add the following code, which sets the `Orientation` property, to the **Page\_Load** event:
  - Visual Basic

```
C1Splitter1.FullSplit = True
```
  - C#

```
C1Splitter1.FullSplit = true;
```
3. Run the program and observe that the control expands to the width and height of your Web browser.

# Setting C1Splitter Behaviors

The C1Splitter control has a list of properties that affect how the control behaves at run time. Some of the properties affect how the control acts when loaded, whereas others affect the users' interactions with the control. The following topics will instruct you on how to modify the run-time actions of the control.

## Setting a Minimum Size for a Splitter Panel

In some instances, you might want to keep users from resizing a panel past a certain point. In **Panel1**, for example, you may have a stack of buttons that you want visible at all times. When confronted with that sort of situation, you can use the MinSize property to specify, in pixels, the size of the area that you don't want users to drag past. In this topic, you will learn how to set the MinSize property in Design view, in Source view, and in Code.

### In Design View

Complete the following steps:

1. Add **C1Splitter** to the Web form.
2. Right-click on the control and select **Properties**. In the Properties window, expand the **Panel1** node.
3. Set the MinSize property for **Panel1** to "30".
4. Save and run your project.

### In Source View

To make **Panel1** a fixed-size panel, place `<Panel1 MinSize="30"></Panel1>` between the `<wijmo:C1Splitter>` and `</wijmo:C1Splitter>` tags. Once the MinSize property has been set, the markup will resemble the following:

```
<wijmo:C1Splitter ID="C1Splitter6" runat="server" Height="251px" Width="217px"
SplitterDistance="50">
  <Panel1 MinSize="30">
  </Panel1>
</wijmo:C1Splitter>
```

### In Code

Complete the following steps:

1. Import the following namespace into your project:

- Visual Basic

```
Imports C1.Web.Wijmo.Controls.C1Splitter
```

- C#

```
using C1.Web.Wijmo.Controls.C1Splitter;
```

2. Add the following code, which sets the MinSize property, to the **Page\_Load** event:

- Visual Basic

```
C1Splitter1.Panel1.MinSize = 30
```

- C#

```
C1Splitter1.Panel1.MinSize = 30;
```

3. Run the program.

✔ **This Topic Illustrates the Following:**

Once you've built the project, drag the splitter bar to the left and observe that it sticks at 30 pixels.

## Setting a Collapsed Splitter Panel

To create a collapsed panel, use the Collapsed property. In this topic, you will learn how to set the Collapsed property in Design view, in Source view, and in code. You can collapse either panel, but this topic will illustrate this setting using **Panel1**.

For more information on collapsed and expanded panels, see [Collapsible and Expandable Panels](#) (page 28).

### In Design View

Complete the following steps:

1. Add C1Splitter to the Web form.
2. Right-click on the control and select **Properties**. In the Properties window, expand the **Panel1** node and locate the Collapsed property.
3. Set the Collapsed property to **True**.

### In Source View

To make Panel1 a collapsed panel, place `<Panel1 Collapsed="True"></Panel1>` between the `<wijmo:C1Splitter>` and `</wijmo:C1Splitter>` tags. The final markup will resemble the following:

```
<wijmo:C1Splitter ID="C1Splitter1" runat="server">
  <Panel1 Collapsed=True>
</Panel1>
</wijmo:C1Splitter>
```

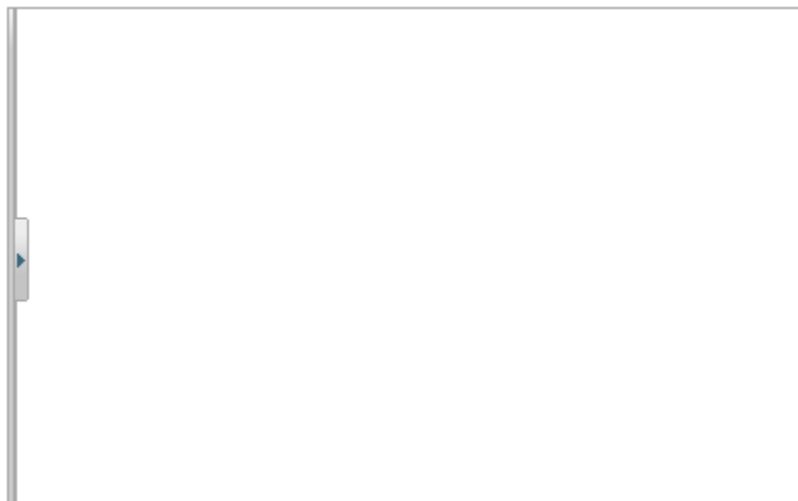
### In Code

Complete the following steps:

1. Import the following namespace into your project:
  - Visual Basic  
`Imports C1.Web.Wijmo.Controls.C1Splitter`
  - C#  
`using C1.Web.Wijmo.Controls.C1Splitter;`
2. Add the following code, which sets the Collapsed property, to the **Page\_Load** event:
  - Visual Basic  
`C1Splitter1.Panel1.Collapsed = True`
  - C#  
`C1Splitter1.Panel1.Collapsed = true;`
3. Run the program.

- ✔ This topic illustrates the following:

Panel1 is collapsed.



To expand the panel, simply click the expand button.

## Using Animation Effects

C1Splitter contains thirty-one transition effects that allow you to customize interaction with the control. In this topic, you will set the Easing and **Duration** properties to create an animation effect that occurs when the splitter bar is moved. This topic illustrates how to set each of these properties in Design view, in Source view, and in code.

### In Design View

Complete the following steps:

1. Select C1Splitter on the Web page and then navigate to the Properties window.
2. Expand the ResizeSettings node and then expand the AnimationOptions node. Set the following properties:
  - Set the Easing property to **EaseOutBounce**. This property determines the animation transition effect.
  - Set the **AnimationDuration** property to **1000**. This will lengthen the duration of the animation effect, guaranteeing that you will notice the effect when you run the program.
3. Run the program and then use your cursor to drag the splitter bar. Release the splitter bar and observe that it bounces for a few seconds before settling into a resting state.

### In Source View

In Source view place `<ResizeSettings AnimationDuration="1500" Easing="EaseOutBounce" />` between the `<wijmo:C1Splitter>` and `</wijmo:C1Splitter>` tags so that the markup appears similar to the following:

```
<wijmo:C1Splitter ID="C1Splitter1" runat="server" Height="298px" ">  
  <Bar Width="14" />  
  <Panel2>  
</Panel2>  
  <Panel1>
```

```
</Panel1>  
    <ResizeSettings AnimationDuration="1000" Easing="EaseOutBounce" />  
</wijmo:C1Splitter>
```

Run the program and then use your cursor to drag the splitter bar. Release the splitter bar and observe that it bounces for a few seconds before settling into a resting state.

### In Code

Complete the following steps:

1. Import the following namespace into your project:

- Visual Basic  
`Imports C1.Web.UI`

- C#  
`using C1.Web.UI;`

2. Set the duration of the animation:

- Visual Basic  
`C1Splitter1.ResizeSettings.AnimationOptions.AnimationDuration = 1000`

- C#  
`C1Splitter1.ResizeSettings.AnimationOptions.AnimationDuration = 1000;`

3. Select an animation transition effect:

- Visual Basic  
`C1Splitter1.ResizeSettings.AnimationOptions.Easing =  
Easing.EaseOutBounce`

- C#  
`C1Splitter1.ResizeSettings.AnimationOptions.Easing =  
Easing.EaseOutBounce;`

4. Run the program and then use your cursor to drag the splitter bar. Release the splitter bar and observe that it bounces for a few seconds before settling into a resting state.

### Using the Ghost Effect

By default, users will only see a dotted line indicating the placement of the splitter bar when they attempt to resize panes. You can, however, set up `C1Splitter` so that a preview of the pane is shown as the user slides the splitter bar. To achieve this effect, you set the `Ghost` property to **True**. In this topic, you will learn how to set the `Ghost` property in Design view, in Source view, and in code.

For more information on the `Ghost` property, see [Panel Previewing](#) (page 29).

### In Design View

Complete the following steps:

1. Add a `C1Splitter` control to your Web project.

2. [Add text to Panel1 and Panel2](#) (page 35) of the C1Splitter control.
3. Right-click the C1Splitter control to open its context menu, then select **Properties**.  
The Properties window appears with C1Splitter's property list in focus.
4. Expand the **ResizeSettings** node.  
The properties of the ResizeSettings class are revealed.
5. Set the Ghost property to **True**.
6. Run the program and resize Panel1 by moving the splitter bar to the right. As you move the splitter bar, observe that the text in Panel1 expands to fit the area that you are sizing and overlaps the text of the second panel.

### In Source View

Complete the following steps:

1. Add a C1Splitter control to your Web project.
2. [Add text to Panel1 and Panel2](#) (page 35) of the C1Splitter control.
3. Click the **Source** button to enter Source view.
4. In Source view, place `<ResizeSettings Ghost="True" />` between the `<wijmo:C1Splitter>` and `</wijmo:C1Splitter>` tags so that the markup appears similar to the following:

```
<wijmo:C1Splitter ID="C1Splitter1" runat="server" Height="150px"
Width="250px">
  <ResizeSettings Ghost="True" />
</wijmo:C1Splitter>
```

5. Run the program and resize Panel1 by moving the splitter bar to the right. As you move the splitter bar, observe that the text in Panel1 expands to fit the area that you are sizing and overlaps the text of the second panel.

### In Code

Complete the following steps:

1. Add a C1Splitter control to your Web project.
2. [Add text to Panel1 and Panel2](#) (page 35) of the C1Splitter control.
3. Double-click the Web page to enter Code view. Observe that a **Page\_Load** event has been added to the page.
4. Import the following namespace into your project:
  - Visual Basic  
`Imports C1.Web.Wijmo.Controls.C1Splitter`
  - C#  
`using C1.Web.Wijmo.Controls.C1Splitter;`
5. Set the Ghost property to **True** by placing the following code in the **Page\_Load** event:

```
Visual Basic
C1Splitter1.ResizeSettings.Ghost = True
```

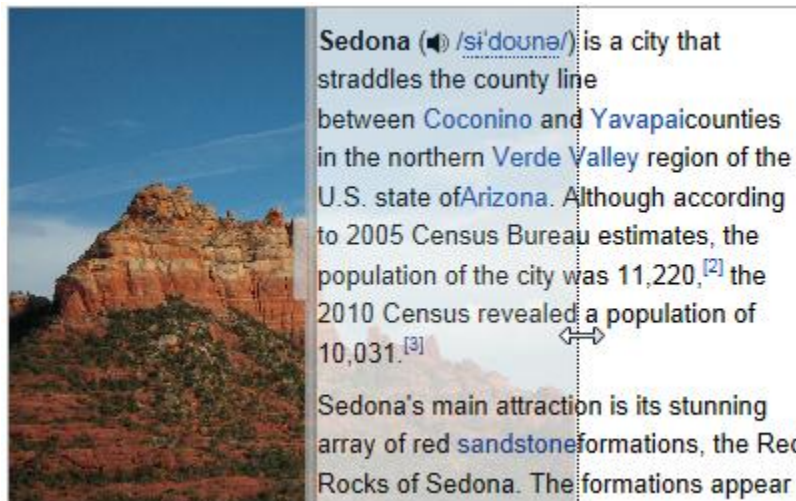
- C#

```
C1Splitter1.ResizeSettings.Ghost = true;
```

6. Run the program resize Panel1 by moving the splitter bar to the right. As you move the splitter bar, observe that the text in Panel1 expands to fit the area that you are sizing and overlaps the text of the second panel.

✔ **This Topic Illustrates the Following:**

The images below, Figure 1 and Figure 2, illustrate the final result of this topic. Figure 1 shows the translucent preview that is created when the Ghost property is set to **True**; Figure 2 depicts what the control looks like after the splitter bar is released and **Panel1** is resized. Note that the text in Figure 2 adopts the form that was revealed in the preview.



For a visual comparison of the Ghost property's settings, see [Panel Previewing](#) (page 29).

## Splitter 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 **Splitter 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/Splitter - Options>
- <http://wijmo.com/wiki/index.php/Splitter - Events>

- <http://wijmo.com/wiki/index.php/Splitter - 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.

**CSAccordion Tasks**

Choose Data Source: (None) ▾

ExpandDirection: Bottom ▾

**Edit Pages**

Add new page

Remove selected page

SelectedIndex: 0

Theme: aristo ▾

Use CDN

CDN path:

[About...](#)