

---

ComponentOne

# **TreeView 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 TreeView 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 .....	11
Redistributable Files.....	11
About This Documentation .....	12
Namespaces.....	12
Creating an ASP.NET Project .....	13
Adding the TreeView for ASP.NET Wijmo Components to a Project .....	14
ComponentOne TreeView for ASP.NET Wijmo Migration Guide.....	15
Prerequisites.....	16
Getting Started.....	16
Migration details.....	16
C1TreeView Migration.....	16
Wijmo Top Tips.....	19
Key Features.....	21
TreeView for ASP.NET Wijmo Quick Start .....	21
Step 1 of 3: Adding C1TreeView to the Page.....	22
Step 2 of 3: Creating a TreeView Using the Designer.....	22
Step 3 of 3: Running the Project .....	23
Design-Time Support.....	25

C1TreeView Smart Tag.....	25
C1TreeViewNodeBinding Collection Editor .....	26
TreeView Designer Form.....	27
Exploring the TreeView Designer Form .....	28
How to Use the Designer .....	30
TreeView Structure and Elements.....	32
TreeView Creation .....	33
Static TreeView Creation.....	34
Dynamic TreeView Creation .....	35
TreeView Appearance and Behavior .....	38
C1TreeView Themes.....	38
Check Boxes.....	40
Drag and Drop Nodes.....	41
Load on Demand .....	42
Node Selection.....	42
Node Navigation.....	42
C1TreeView CSS Selectors .....	43
TreeView for ASP.NET Wijmo Task-Based Help .....	45
Creating and Configuring Check Box Nodes.....	45
Creating Node Check Boxes.....	45
Preventing Child Nodes from Being Automatically Checked.....	45
Working with Themes.....	47
Using a Built-In Theme .....	47
Using a Custom Theme.....	48
Working with C1TreeView CSS Selectors.....	50
Adding a Top-Level Node to a TreeView.....	51
Adding a Child Node to a TreeView Node.....	51
Populating C1TreeView with a Site Map .....	53
Populating C1TreeView with XML.....	55
Saving and Loading a C1TreeView from XML.....	57
Setting the Auto Collapse Property.....	58
Setting C1TreeView to Open on Hover .....	61
Setting C1TreeView Node Icons.....	61
Setting C1TreeView Properties to Allow Drag-and-Drop Behaviors .....	64
Drag-and-drop Behaviors Within One Tree Structure .....	64
Drag-and-drop Behaviors Between Two Tree Structures.....	65

TreeView for ASP.NET Wijmo Client-Side Reference .....	73
Using the Wijmo CDN .....	73



# ComponentOne TreeView for ASP.NET Wijmo Overview

Present items in a hierarchical tree structure with **ComponentOne TreeView™ for ASP.NET Wijmo**. The **C1TreeView** control supports expand/collapse animations, stylish themes, and the ever popular drag-and-drop functionality.

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

- [TreeView for ASP.NET Wijmo Quick Start](#) (page 21)
- [TreeView Appearance and Behavior](#) (page 38)
- [Task-Based Help](#) (page 45)

## 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:

### Samples

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

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

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

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

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

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

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

## System Requirements

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

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

## Uninstalling Studio for ASP.NET Wijmo

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

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

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

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

## Deploying your Application in a Medium Trust Environment

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

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

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

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

### **Modifying or Editing the Config File**

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

#### **Edit the Config File**

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

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

#### **Create a Custom Policy Based on Medium Trust**

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

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

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

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

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

### **Allowing Deserialization**

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

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

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

## Adding Permissions

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

### ReflectionPermission

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

To add reflection permission, complete the following:

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

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

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

```
<NamedPermissionSets>
  <PermissionSet class="NamedPermissionSet" version="1"
Name="ASP.Net">
  <IPermission
    class="ReflectionPermission"
```

```

        version="1"
        Flags="ReflectionEmit,MemberAccess" />
    ...
</PermissionSet>
</NamedPermissionSets>

```

4. Save and close the web\_mediumtrust.config file.

### OleDbPermission

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

To add OleDbPermission, complete the following steps:

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

```

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

```

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

```

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

```

4. Save and close the web\_mediumtrust.config file.

### FileIOPermission

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

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

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

```

<SecurityClasses>

```

```

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

```

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

```

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

```

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

## End-User License Agreement

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

## Licensing FAQs

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

### What is Licensing?

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

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

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

### How does Licensing Work?

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

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

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

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

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

These files are automatically added to the project.

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

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

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

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

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

## **Common Scenarios**

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

### ***Creating components at design time***

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

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

### ***Creating components at run time***

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

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

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

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

### ***Inheriting from licensed components***

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

- Add a LicenseProvider attribute to the component.

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

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

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

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

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

### ***Using licensed components in console applications***

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

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

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

### ***Using licensed components in Visual C++ applications***

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

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

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

4. Use CILc.exe to compile the licenses.licx file. The command line should look like this:

```
cilc /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 may not exist, it may contain incorrect information, or it may not be 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.
5. Paste the line from step 5 into this file (replace the old licensing information with the new).
6. 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 **C1TreeView** using the fully qualified name for this class:

- Visual Basic

```
Dim MaskedInput As C1.Web.Wijmo.Controls.C1TreeView
```

- C#

```
C1.Web.Wijmo.Controls.C1TreeView;
```

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 C1TreeView = C1.Web.UI.Controls.C1TreeView
Imports MyTreeView = MyProject.Objects.C1TreeView

Dim wm1 As C1TreeView
Dim wm2 As MyTreeView
```

- C#

```
using C1TreeView = C1.Web.UI.Controls.C1TreeView;
using MyTreeView = MyProject.Objects.C1TreeView;

C1TreeView wm1;
MyTreeView 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 TreeView for ASP.NET Wijmo Components to a Project

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

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

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

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

- C1TreeView

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.C1TreeView. Note that there may be more than one component for each namespace.
5. Click **OK** to close the dialog box. The controls are added to the Visual Studio Toolbox.

### Adding Studio for ASP.NET Wijmo Controls to the Form

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

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

### Adding a Reference to the Assembly

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

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

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

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

## ComponentOne TreeView for ASP.NET Wijmo Migration Guide

ComponentOne has recently released its latest and greatest in ASP.NET controls, ASP.NET Wijmo. In the spotlight now is ComponentOne's C1TreeView control. If you have used the older control (ASP.NET AJAX) then you will know of some of the cool features that will come packaged. The Wijmo control features:

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

If you have used the ASP.NET Ajax control and are interested in moving over to the new Wijmo control, this will be a good read. Let's take a look at the procedures involved in migrating over to our latest controls and delve into some details about some basic differences between ASP.NET Wijmo controls and their older ASP.NET AJAX controls along the way.

## Prerequisites

To get started in this migration you will need our ASP.NET Wijmo controls. These can be downloaded [here](#) by clicking the orange "Download Free Trial" button.

## Getting Started

To get things started, you should create a new ASP.NET Web Application using ComponentOne's older AJAX controls. Below is the link to the quick-start guide where you will receive directions to create a web page with a treeview control and visual aid as to what the final product will look like. Once the control is set up on your web application, we can then begin to delve into the migration details for our ASP.NET Wijmo controls. During the migration process, we will note changes between the AJAX and Wijmo controls.

Follow [this quick-start to create the C1TreeView](#) using our ASP.net Ajax control.

## Migration details

Once you have completed the quick-start guide you can begin migrating to the new ASP.NET Wijmo controls. The first thing you will want to do is add the new ASP.NET Wijmo assembly references. Go to **Project** from the menu and select **Add Reference**. Click the **Browse...** button and navigate to:

### For .net 4.0

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

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

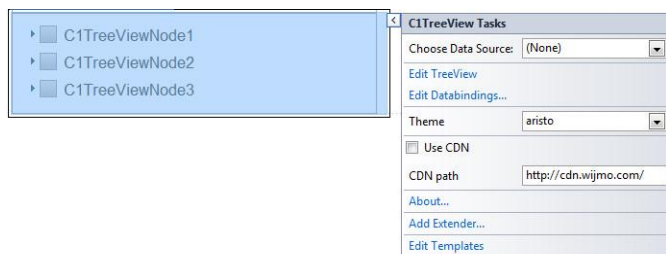
### For .net 3.5

- *For 32bit Machines:* C:\Program Files\ComponentOne\Studio for ASP.NET Wijmo\bin\v3
- *For 64bit Machines:* C:\Program Files (x86)\ComponentOne\Studio for ASP.NET Wijmo\bin\v3

Select the C1.Web.Wijmo.Controls.3 and click "Open".

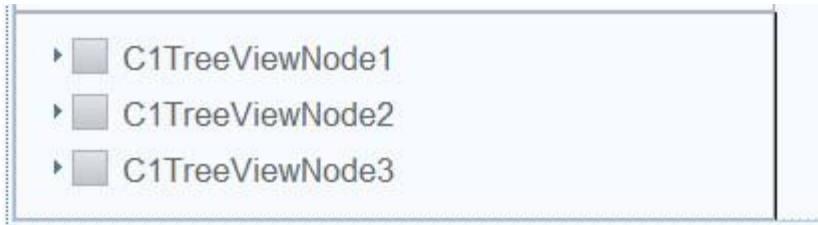
Then in the **Add Reference to...** dialog box, click **Add**. In your solution, open up your page where you have your ASP.NET control in Design view.

## C1TreeView Migration



To add the C1TreeView control, simply drag the C1TreeView from the Toolbox onto the page. This will display a C1TreeView box without any nodes.

You can use the **C1TreeView Designer Form** dialog box to add nodes and edit each of the nodes properties by clicking the **Edit TreeView** link. All data binding is performed the same way as with the original ASP.NET control. The C1TreeView nodes will populate accordingly based on your data source or static declaration as seen below:



If you created panes in code and wanted to move those items from your old ASP.NET AJAX C1TreeView control to your new ASP.NET Wijmo C1TreeView control, you could do so by manipulating the code. The treeview markup is very similar. The hierarchical markup is represented in both the AJAX and Wijmo controls as follows:

- <TreeView>
- <Nodes>
- <TreeNode>

You would then need to find and replace every instance of “cc1” to “wijmo”.

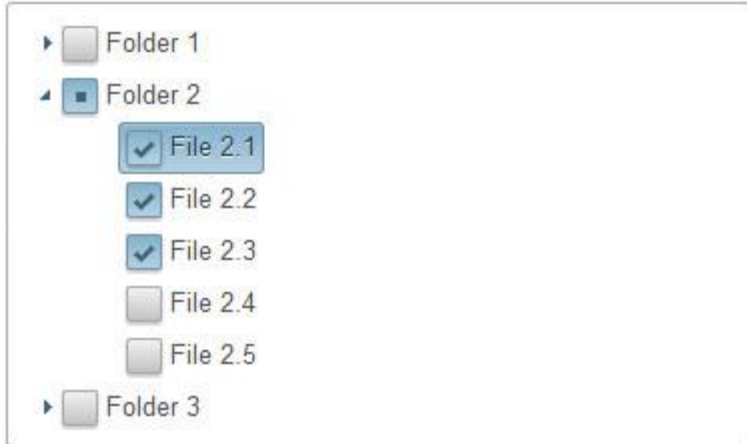
The **C1TreeView Designer Form** dialog box can be used to edit the C1TreeView control. The properties are very similar, which makes it easier to move over. Some of the changes can be noted below.

## Wijmo

(Expressions)	
(ID)	C1TreeView2
AccessKey	
AllowDrag	True
AllowDrop	True
AllowEdit	False
AllowSorting	True
AllowTriState	True
AutoCheckNodes	True
AutoCollapse	False
AutoPostBack	False
CDNPath	http://cdn.wijmo.com/
ClientIDMode	Inherit
CollapseAnimation	C1.Web.Wijmo.Controls.Animation
CollapseDelay	0
CssClass	
DataBindings	(Collection)
DataBindStartLevel	-1
DataSourceID	
Enabled	True
EnableTheming	True
EnableViewState	True
ExpandAnimation	C1.Web.Wijmo.Controls.Animation
ExpandCollapseHoverUsed	False
ExpandDelay	0
Height	
LoadOnDemand	False
OnClientNodeCheckChanged	
OnClientNodeClick	
OnClientNodeCollapsed	
OnClientNodeDragging	
OnClientNodeDragStarted	
OnClientNodeDropped	
OnClientNodeExpanded	
OnClientNodeMouseOut	
OnClientNodeMouseOver	
OnClientNodeTextChanged	
OnClientSelectedNodeChanged	
SelectedNodes	(Collection)
ShowCheckBoxes	True
ShowExpandCollapse	True
SkinID	
TabIndex	0
Theme	aristo
ToolTip	
UseCDN	False
ViewStateMode	Inherit
Visible	True
Width	

## Ajax

(Expressions)	
(ID)	C1TreeView4
AccessKey	
AllowDragDrop	False
AllowEdit	False
AllowSorting	False
AllowTriState	False
AutoCheckNodes	False
AutoCollapse	False
AutoPostBack	False
C1WebControlsPath	~/C1WebControls/
ClientIDMode	Inherit
CollapseAnimation	None
CollapseDelay	100
CollapseDuration	500
CollapseEasing	EaseLinear
CssClass	
DataBindings	(Collection)
DataBindStartLevel	-1
DataSourceID	
DisplayVisible	True
Enabled	True
EnableTheming	True
EnableViewState	True
ExpandAnimation	None
ExpandCollapseHoverUsed	False
ExpandDelay	0
ExpandDepth	-2
ExpandDuration	500
ExpandEasing	EaseLinear
Height	
LoadOnDemand	False
OnClientNodeCheckChanged	
OnClientNodeClicked	
OnClientNodeCollapsed	
OnClientNodeDragging	
OnClientNodeDragStarted	
OnClientNodeDropped	
OnClientNodeExpanded	
OnClientNodeMouseOut	
OnClientNodeMouseOver	
OnClientNodeTextChanged	
OnClientSelectedNodesChanged	
SelectedNodes	(Collection)
ShowCheckBoxes	False
ShowExpandCollapse	True
ShowLines	True
SkinID	
TabIndex	0
ToolTip	
UseEmbeddedjQuery	True
UseEmbeddedVisualStyles	True
ViewStateMode	Inherit
VisualStyle	ArcticFox
VisualStylePath	~/C1WebControls/VisualStyles
Width	



The C1TreeView control provides check box support by simply setting the **showCheckBox** property to **True**. This will create your C1TreeView with checkable nodes. Setting the **AllowDrag** and **AllowDrop** properties will allow you to maneuver your nodes as you wish. These properties also allow you to drag nodes from one tree view to another. Add keyboard accessibility support to give the C1TreeView control focus with a specified key combination. This enables end-users to use the keyboard arrow keys to navigate through the C1TreeView items. Make the C1TreeView look pretty using the packaged Wijmo themes. Keep in mind that you could easily change the appearance of your control by setting the theme to one of 30 pre-existing themes. These themes can be easily styled with ThemeRoller and of course you could create your own! [This and many other features can be found at the Wijmo TreeView control product page.](#)

## 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" />
```



# Key Features

The C1TreeView control contains the following key features:

- **Data Binding Support**

Bind the C1TreeView control to a data source – you can bind to an XML or SiteMap data source, or you can even read data from an Access data source and create the C1TreeView hierarchy dynamically.
- **Drag-and-drop Nodes**

You can drag-and-drop C1TreeViewNodes on nodes, in between nodes, or from one tree to another tree. Visual cues, such as a vertical gray line, are used to show you where the C1TreeViewNode is going to be dropped.
- **Check Box Support**

Node items can be implemented as regular check boxes. This enables end-users to check or uncheck the boxes to select or unselect the corresponding nodes. When the check boxes are enabled for the C1TreeView you can create an action when the status of a check box changes between posts.
- **Animation**

C1TreeView supports expand and collapse animation effects. Typically collapsing treeview items use animations that scroll in, fade in, fold in, close or drop in and expanding treeview items use animations that scroll out, fade out, fold out, open, or drop out. You also have the flexibility to specify transition effects and how long the animation lasts.
- **Templates Support**

Change the tree view's appearance with the built-in template editor. Add your own elements, including text, images, and controls such as buttons, to various nodes.
- **Keyboard Support**

Add access key support to give the C1TreeView control focus with a chosen key combination. This enables end-users to use the keyboard arrow keys to navigate through the treeview items.
- **Theming**

With just a click of the SmartTag, change the treeview's look by selecting one of the 5 premium themes (Midnight, Aristo, Rocket, Cobalt, and Sterling). Optionally, useThemeRoller from jQuery UI to create a customized theme!
- **CSS Support**

Use a cascading style sheet (CSS) style to define custom skins. CSS support allows you to match the treeview to your organization's standards.

## TreeView for ASP.NET Wijmo Quick Start

In this quick start you will learn how to create a root and child items for the **C1TreeView** control, apply a visual style, drag-and-drop treeview nodes, enable check boxes, and bind to an XMLDataSource.

## Step 1 of 3: Adding C1TreeView to the Page

In this lesson you will learn how to create a new ASP.NET Web site and add a **C1TreeView** control to your project.

To begin the Quick Start, complete the following steps:

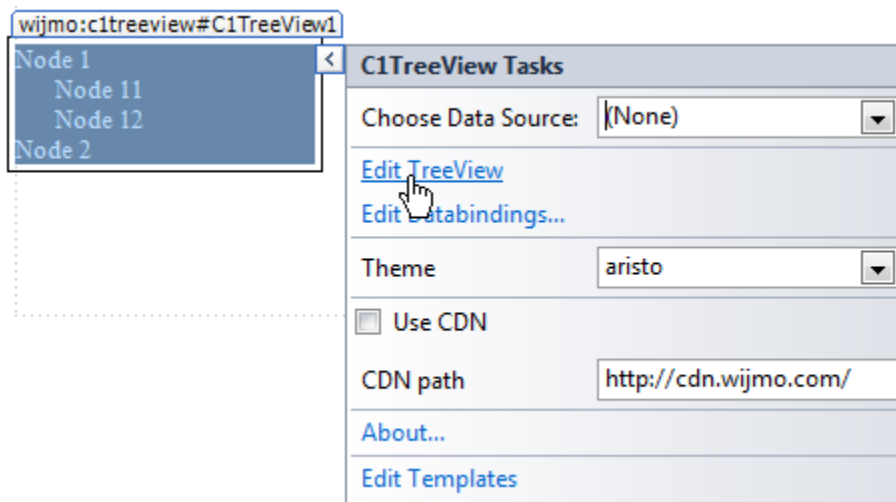
1. Begin by creating a new ASP.NET Web Site.
2. While in Design view navigate to the Visual Studio Toolbox and double-click the **C1TreeView** icon to add the **C1TreeView** control to your page.

The page will appear similar to the following:

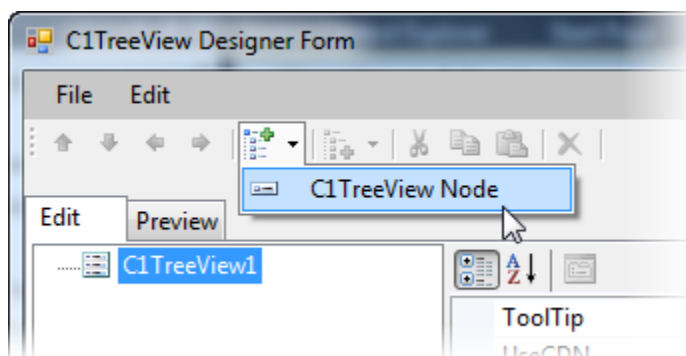
## Step 2 of 3: Creating a TreeView Using the Designer

This lesson will show you how to create root and child nodes, apply a visual style, and display check boxes next to the nodes.

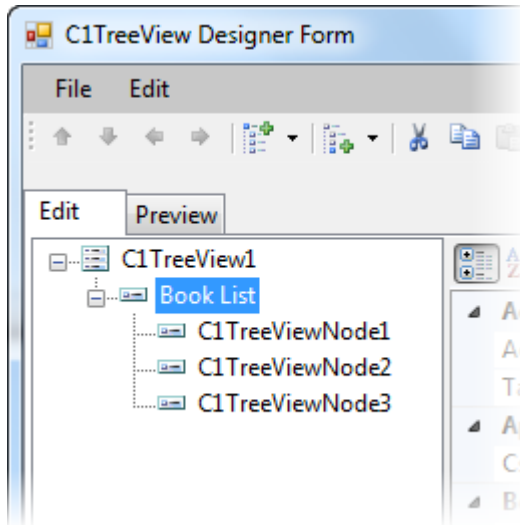
1. Select the **C1TreeView** control and click on the smart tag to open its **Tasks** menu.
2. Select **Edit TreeView** from **C1TreeView Tasks** menu to open the designer.



3. Right-click on the **C1TreeView** item and select **Add Child | C1TreeView Node** to add the root to the **C1TreeView** control. Set the **C1TreeViewNode1**'s **Text** property to "Book List".



4. Right-click on **Book List** and select **Add Child** to create a child for the root node. Repeat this two more times. Three child nodes will exist under the Book List.



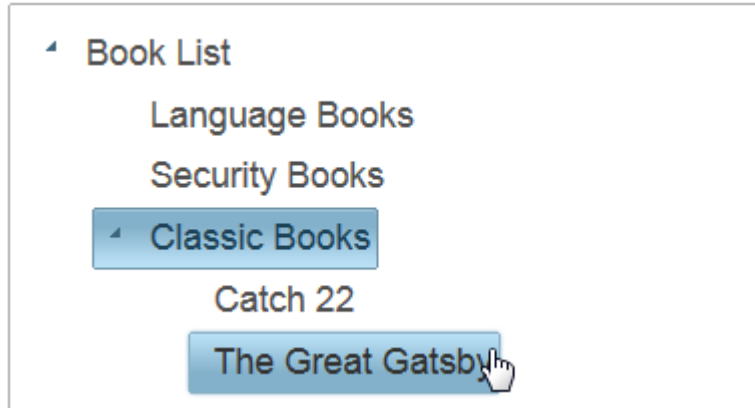
5. Select the first node under the Book List and set its **Text** property to "Language Books".
6. Select the second node under the Book List and set its **Text** property to "Security Books".
7. Select the third node under the Book List and set its **Text** property to "Classic Books".
8. Right-click on the **Classic Books** node and select **Add Child** to create a child for the Classic Books node. Repeat this to create two **C1TreeViewNodes** under the **Classic Books** node.
9. Select the first node under the **Classic Books** node and set its **Text** property to "The Great Gatsby".
10. Select the second node under the **Classic Books** node and set its **Text** property to "Catch-22".
11. Right-click on **Book List** and select **Add Child** to add to add a child node.

In this step, you nodes and child nodes to the **C1TreeView** control. In the next step, you'll run the project and see the results of this quick start.

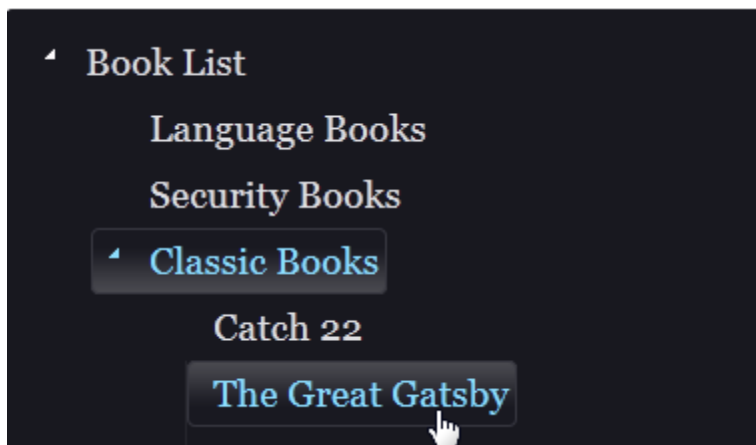
## Step 3 of 3: Running the Project

In this step, you'll run the project and see the results of this quick start.

1. Save and run your project and observe the following:
  - The root node, Book List, isn't expanded since the Expanded property was set to **False** by default.
  - Expand the **Book List** node and notice the child nodes you created.
  - Notice the theme is the default theme, Aristo.



2. Go back to your project in the design view and click the **CITreeView**'s smart tag to open the **CITreeView Tasks** list. Click the **Theme** drop-down arrow and select **midnight** from the list.
3. Select the **CITreeView** item and set the **VisualStyle** property to **Vista**.
4. Save and run your project and notice the new theme, midnight, is applied to **CITreeView**.



5. Go back to your project in the Design view and open the **TreeView Designer Form**.
6. Select the **CITreeView** item and in the Properties window, set its behavior properties to the following:
  - AllowDrag to **True**
  - AllowDrop to **True**.
  - ShowCheckBoxes to **True**
7. Click **OK** to save and close the designer.
8. Run the project and observe the following:
  - Expand the Book List node and notice the check boxes next to each treeview node.
  - Select any of the treeview nodes and drag it to a new location.

# Design-Time Support

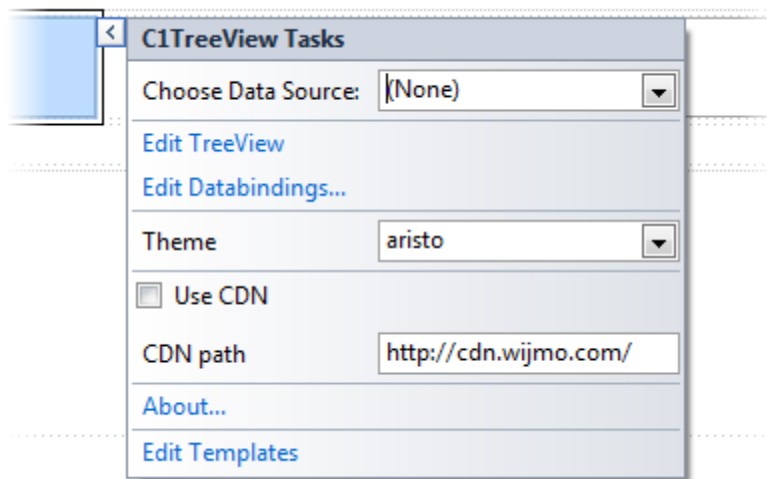
The following sections describe how to use **C1TreeView**'s design-time environment to configure the **C1TreeView** control.

## C1TreeView Smart Tag

The **C1TreeView** control includes a smart tag in Visual Studio. A smart tag represents a short-cut tasks menu that provides the most commonly used properties in **C1TreeView**.

The **C1TreeView** control provides quick and easy access to the **TreeView Designer Form** and common properties through its smart tag.

To access the **C1TreeView Tasks** menu, click on the smart tag in the upper-right corner of the **C1TreeView** control. This will open the **C1TreeView Tasks** menu.



The **C1TreeView Tasks** menu operates as follows:

- **Choose Data Source**  
Clicking on the **Choose Data Source** item opens a drop-down list where you can choose an existing data source or select a new data source to bind to.
- **Edit TreeView**  
Clicking on the **Edit TreeView** item opens the **TreeView Designer Form** where you can quickly configure **C1TreeView**'s elements without having to scroll through its Properties window. Here you can add, remove, and re-order **C1TreeViewNodes** as well as set a variety of properties defining their appearance, behavior, and more. For more information on the **TreeView Designer Form**, see [TreeView Designer Form](#) (page 27).
- **Edit DataBindings**  
Clicking on the **Edit Databindings** item opens the **Bindings Collection Editor** dialog box where you can add and remove bindings and edit properties.
- **Theme**

Clicking the **Theme** drop-down arrow enables you to select from different built-in visual styles. See [C1TreeView Themes](#) (page 38) for more information.

- **Use CDN**

Determines whether the control is using the CDN for the client-side reference.

- **CDN Path**

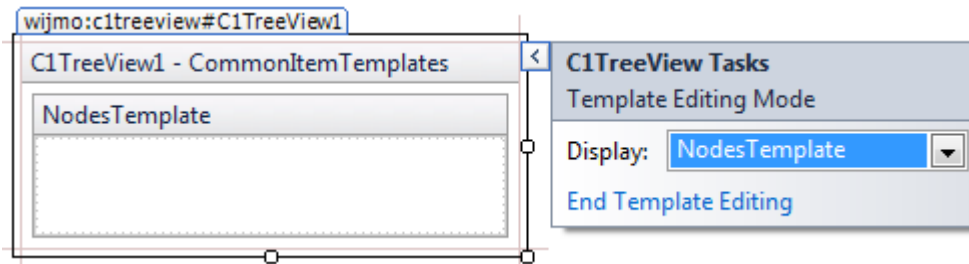
The path to the CDN library you are using.

- **About**

Clicking on the **About** item displays the About dialog box, which is helpful in finding the version number of **Treeview for ASP.NET** and online resources.

- **Edit Templates**

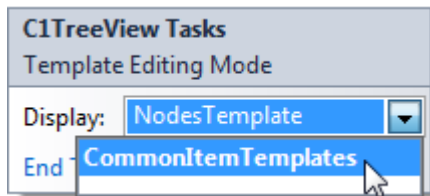
Clicking on the **Edit Templates** item switches the **C1TreeView** control to Template Editing Mode:



In Template Editing Mode, the **C1TreeView Tasks** menu appears with different options:

- **Display**

Selecting the **Display** drop-down arrow will open a list of template areas that can be customized:



Select a template from this list to open that template to be edited.

- **End Template Editing**

Clicking the **End Template Editing** item will end Template Editing Mode and return you to the main **C1TreeView Tasks** menu.

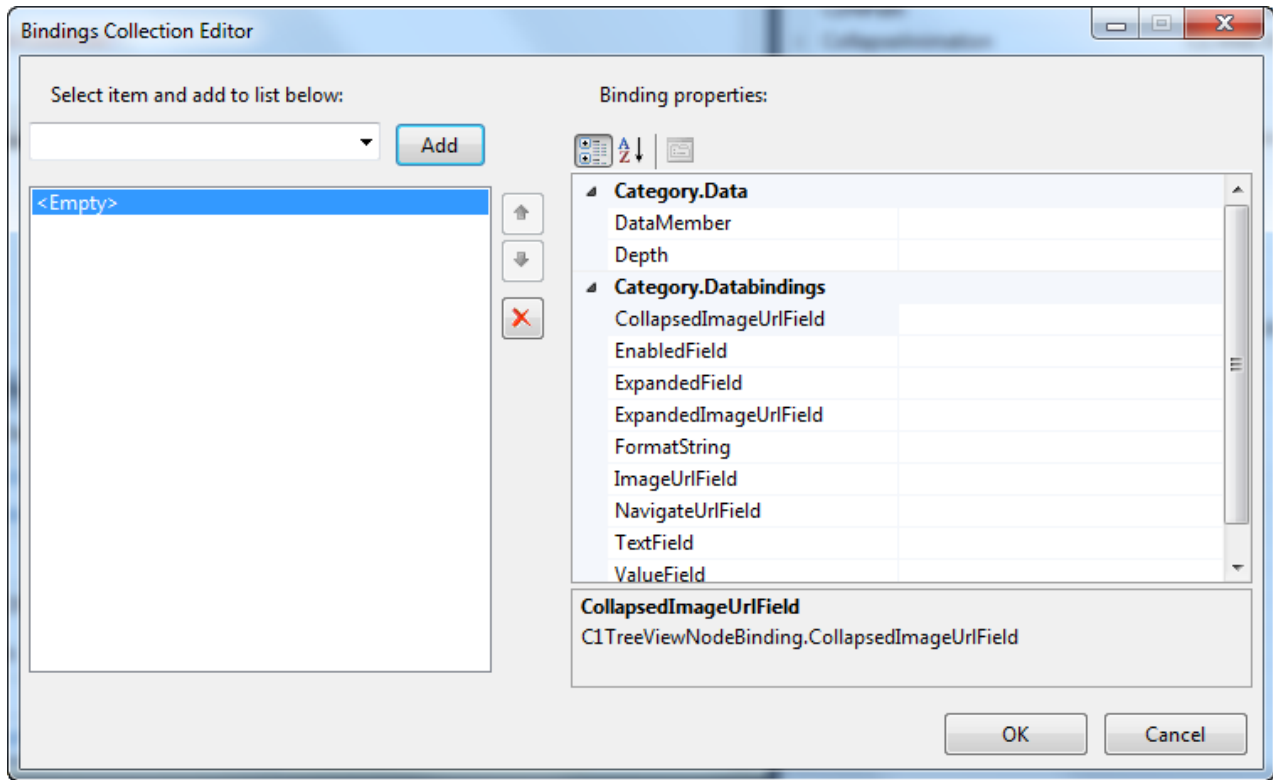
## C1TreeViewNodeBinding Collection Editor

The **C1TreeView** control includes a collection editor that allows you to add or remove databindings from the **C1TreeViewNodeCollection**, as well as specify binding properties.

There are two ways to access the **C1TreeViewNodeBinding Collection Editor**:

### From the C1TreeView Tasks menu:

1. Click the smart tag in the upper-right corner of the C1TreeView control to open the **C1TreeView Tasks** menu.
2. Select **Edit databindings**. The **Bindings Collection Editor** appears.



### From the TreeView Designer Form:

1. Click the smart tag in the upper-right corner of the C1TreeView control to open the **C1TreeView Tasks** menu.
2. Select **Edit TreeView**. The **TreeView Designer Form** appears.
3. With the **C1TreeView** control selected, click the **ellipsis** button next to the **DataBindings** property. The **C1TreeViewNodeBinding Collection Editor** appears. This dialog box, although it appears slightly different, is essentially the same as and contains the same properties as the **Bindings Collection Editor**.

## TreeView Designer Form

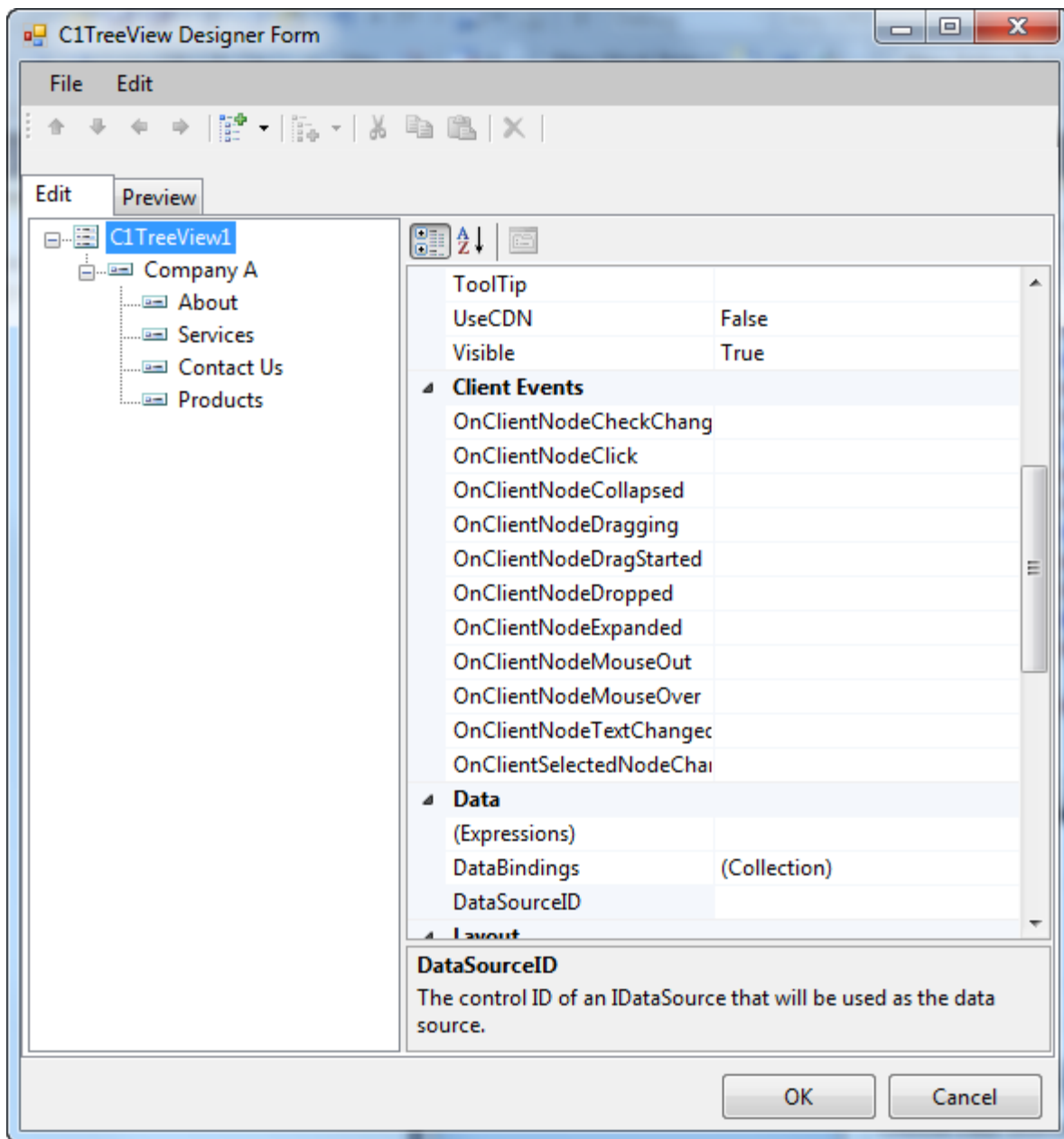
The **TreeView Designer Form** is **C1TreeView**'s designer for editing its properties, as well as the **C1TreeViewNode** properties. The **TreeView Designer Form** is similar to the Properties window as it allows programmers to modify the control visually. However, it allows you to select a **C1TreeViewNode**, set its properties, manipulate the nodes, and then preview the appearance of the C1TreeView control, all within the form.

In this topic you will become familiar with the **TreeView Designer Form**'s design interface so you can use the commands within it to edit C1TreeView with minimal effort and time.

To open the **TreeView Designer Form**, click the C1TreeView smart tag and select the **Edit TreeView** link from the **C1TreeView Tasks** menu:

## Exploring the TreeView Designer Form

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



### Edit Tab

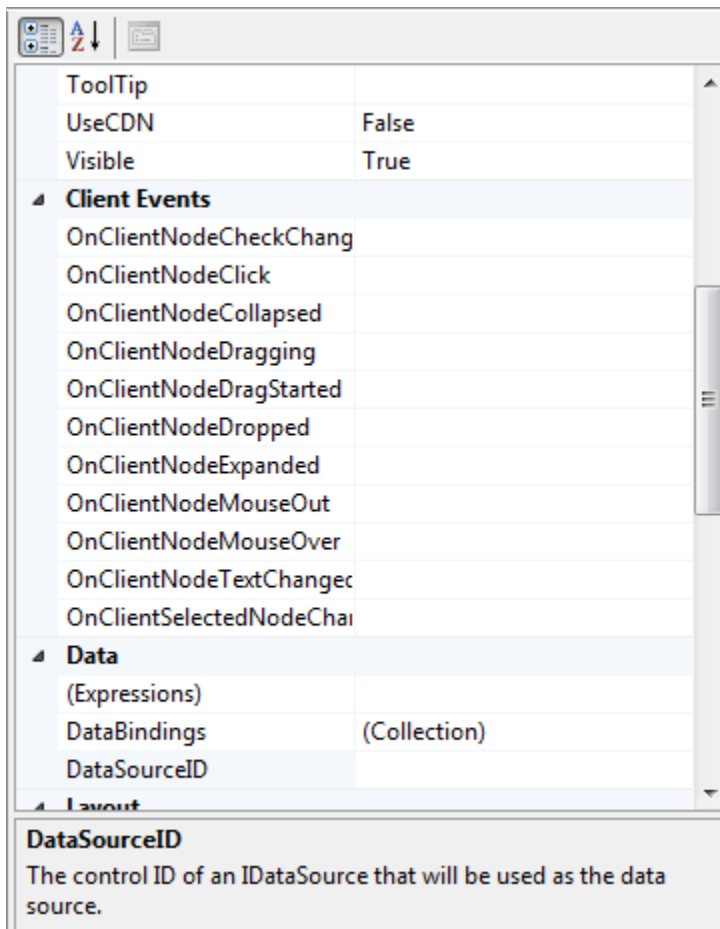
Click the **Edit** tab and select the C1TreeView control or the desired **C1TreeViewNode** for which you would like to manipulate or adjust the properties.

### Preview Tab

Click the **Preview** tab for a WYSIWYG preview of what the C1TreeView control will look like.

### Properties Pane

The **TreeView Designer Form** properties pane is almost identical to the Visual Studio Properties window. Simply select a **C1TreeNode** or the **C1TreeView** control and set the desired properties here.



### Command Buttons

The command buttons are summarized in the following table:

Button	Description
<b>OK</b>	Clicking <b>OK</b> applies the new settings to the <b>C1TreeView</b> control.
<b>Cancel</b>	Clicking <b>Cancel</b> closes the <b>TreeView Designer Form</b> , cancelling the new settings and applying the default settings to the <b>C1TreeView</b> control.

### TreeView Designer Form Menu

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

Menu Item	Submenu Item	Description
File	Load from XML	Load the formatting for a <b>C1TreeView</b> control from an .xml file.
	Save as XML	Save the current formatting of the <b>C1TreeView</b> control to an .xml file.
	Exit	Closes the <b>TreeView Designer Form</b> .

Edit	Insert Item	Inserts a new <b>C1TreeViewNode</b> at the specified place in the list of nodes.
	Add Child	Adds a new <b>C1TreeViewNode</b> as a child of the <b>C1TreeView</b> or of another <b>C1TreeViewNode</b> .
	Cut	Cuts the selected <b>C1TreeViewNode</b> to be moved in the list of nodes.
	Copy	Copies the selected <b>C1TreeViewNode</b> .
	Paste	Pastes a <b>C1TreeViewNode</b> at the specified location in the list of nodes.
	Delete	Removes the selected <b>C1TreeViewNode</b> .
	Rename	Allows you to change the name of the <b>C1TreeViewNode</b> .

### **TreeView Designer Form Toolbar**

The toolbar for the TreeView Designer Form appears like the following:



The table below describes each button in the toolbar:

Button	Name	Description
	Move Item Up	Moves the selected <b>C1TreeViewNode</b> up in the list of nodes.
	Move Item Down	Moves the selected <b>C1TreeViewNode</b> down in the list nodes.
	Move Item Left	Moves the selected <b>C1TreeViewNode</b> to the left in the hierarchy.
	Move Item Right	Moves the selected <b>C1TreeViewNode</b> to the right in the hierarchy.
	Add Child Item	Inserts a <b>C1TreeViewNode</b> as a child of the <b>C1TreeView</b> control or of another <b>C1TreeViewNode</b> .
	Insert Item	Inserts a <b>C1TreeViewNode</b> at the specified location in the list of nodes.
	Cut	Cuts the selected <b>C1TreeViewNode</b> to be moved in the list of nodes.
	Copy	Copies the selected <b>C1TreeViewNode</b> .
	Paste	Pastes a <b>C1TreeViewNode</b> at the specified location in the list of nodes.
	Delete	Removes the selected <b>C1TreeViewNode</b> .

### **How to Use the Designer**

The following topics illustrate how to use the **TreeView Designer Form** for several tasks.

#### **Deleting a C1TreeViewNode**

You can use one of the following three methods when deleting a **C1TreeViewNode** in the designer:

- **Deleting a child node through the shortcut menu**  
Right-click the **C1TreeNode** you wish to delete and select **Delete**.
- **Deleting a child node by pressing on the delete button**  
Select the node you wish to delete and click on the **Delete**.
- **Deleting a child node through the Edit menu.**  
Click the Edit menu and select **Delete**.

### ***Renaming the TreeNode in the Designer***

You can use one of the following three methods when renaming a **C1TreeNode** in the designer:

- **Pressing F3**
  - a. Select the **C1TreeNode** you wish to rename.
  - b. Press the F3 key and type the new name.
- **Selecting rename from the shortcut menu**
  - a. Right-click the **C1TreeNode** you wish to rename.
  - b. Select **Rename** from the context menu and type the new name.
- **Selecting rename from the Edit menu**
  - a. Select the **C1TreeNode** you wish to rename.
  - b. Click the Edit menu, select **Rename**, and enter the new name.

### ***Adding a Child Node***

You can use one of the following three methods when adding a child **C1TreeNode** in the designer:

- **Adding a child node through the shortcut menu**  
Right-click on the node you wish to add a child node and select **Add Child | C1TreeNode**.
- **Adding a child node by pressing on the add child button**  
Select the node you wish to add a child node and click on the **Add Child** button's drop-down arrow and select **C1TreeNode**.
- **Adding a child node through the Edit menu**  
Select the node you wish to add a child node and click on the **Edit** menu and select **Add Child | C1TreeNode**.

### ***Inserting a Node***

You can use one of the following three methods when inserting a **C1TreeNode** in the designer:

- **Inserting a node through the shortcut menu**  
Right-click on the **C1TreeNode** you wish to add a child node and select **Insert Item | C1TreeNode**.
- **Inserting a node by pressing on the Insert Node button**  
Select the **C1TreeNode** you wish to add a child node and click on the **Insert Item** button's drop-down arrow and select **C1TreeNode**.
- **Inserting a node using the Edit menu**

Select the `CITreeViewNode` you wish to add a child node and click on the **Edit** menu and select **Insert Item | CITreeViewNode**.

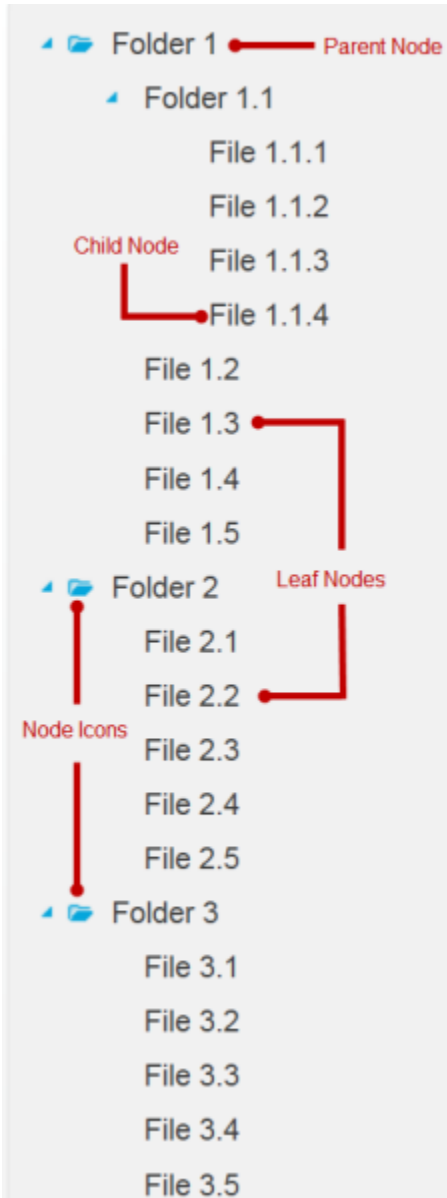
# TreeView Structure and Elements

**CITreeView** is a tree type Web control that displays a hierarchical tree structure. A tree structure is used to represent hierarchical data into a graphical form. This section will provide a visual and written overview of the structure and elements of **CITreeView**.

A tree contains one or more elements where each element is a node. A node can be a parent, child, or leaf node. The description for each type of node is as follows:

- Parent node is a node that contains other nodes.
- Child node is a node that is contained by another node.
- Leaf node is a node that does not contain child nodes.

Like the classic tree structure, the **CITreeView** contains one or more nodes that consist of parent, child, and leaf nodes. The parent, child, and leaf nodes are referred as **CITreeViewNodes**. The following image illustrates the nodes and structure of the **CITreeView** control.



**C1TreeView** is drawn like an inverted tree where the root appears first. The **C1TreeView** can contain one or more root nodes. If a node has child nodes it can be collapsed or expanded. The `ShowExpandCollapse` property allows nodes to be expanded or collapsed when set to true. Each node can have text and an image associated with it, may be edited, selected, or display check boxes depending upon the property settings for the **C1TreeView** and **C1TreeViewNode** objects. The tree node can be rendered as a hyperlink and have a URL associated with it.

## TreeView Creation

**C1TreeViewNodes** can be defined on your page or user control by using any of the following methods:

- Static creation using declarative syntax
- Dynamic creation using a constructor to create new instances of the **C1TreeViewNode** class.
- Data source creation through binding **C1TreeView** to a `SiteMapDataSource`, `XMLDataSource`, or an `AccessDataSource`.

## Static TreeView Creation

Each node in the Tree is represented by a name/value pair, defined by the text and value properties of `treenode`, respectively. The text of a node is rendered, whereas the value of a node is not rendered and is typically used as additional data for handling postback events.

A static menu is the simplest way to create the treeview structure.

You can use the **TreeView Designer Form** designer to build the treeview system or you can use declarative syntax in the .aspx file to specify the nodes.

To display static **C1TreeViewNodes** using the designer, open the **TreeView Designer Form** and add **C1TreeViewNodes** to the parent. The properties for each **C1TreeViewNode** can be modified directly in the designer. For more information about the menu designer, see [TreeView Designer Form](#) (page 27).

To display static **C1TreeViewNodes** using declarative syntax, first nest opening and closing `<Nodes>` tags between opening and closing tags of the **C1TreeView** control. Next, create the treeview structure by nesting `<asp:C1TreeViewNode>` elements between opening and closing `<Nodes>` tags. Each `<asp:C1TreeViewNode>` element represents a node in the control and maps to a **C1TreeViewNode** object.

Declarative syntax can be used to define the **C1TreeViewNodes** inline on your page.

For example:

```
<wijmo:C1TreeView ID="C1TreeView1" runat="server"
AllowSorting="False" AutoCollapse="False"
    VisualStyle="Default"
VisualStylePath="~/C1WebControls/C1TreeView/VisualStyles">
    <Nodes>
        <wijmo:C1TreeViewNode runat="server" Expanded="False"
Text="C1TreeViewNode">
            <Nodes>
                <wijmo:C1TreeViewNode runat="server"
Expanded="False" Text="C1TreeViewNode">
                    </wijmo:C1TreeViewNode>
                <wijmo:C1TreeViewNode runat="server"
Expanded="False" Text="C1TreeViewNode">
                    </wijmo:C1TreeViewNode>
                <wijmo:C1TreeViewNode runat="server"
Expanded="False" Text="C1TreeViewNode">
                    <Nodes>
                        <wijmo:C1TreeViewNode runat="server"
Expanded="False" Text="C1TreeViewNode">
                                </wijmo:C1TreeViewNode>
                        <wijmo:C1TreeViewNode runat="server"
Expanded="False" Text="C1TreeViewNode">
                                </wijmo:C1TreeViewNode>
                        <wijmo:C1TreeViewNode runat="server"
Expanded="False" Text="C1TreeViewNode">
                                </wijmo:C1TreeViewNode>
                    </Nodes>
                </wijmo:C1TreeViewNode>
            </Nodes>
        </wijmo:C1TreeViewNode>
    </Nodes>
</wijmo:C1TreeView>
```

## Dynamic TreeView Creation

Dynamic treeviews can be created on the server side or client side. When creating dynamic treeview on the server side, use a constructor to dynamically create a new instance of the **C1TreeViewNode** class. For client-side, the **CreateInstance** constructor can be used to dynamically create a new instance of the **C1TreeView** control. For example the follow script creates a new **C1TreeView** control on the client side:

```
var aTreeView = C1.Web.C1TreeView.createInstance ();
```

`document.body.appendChild(aTreeView.element);` **C1TreeView** or **C1TreeViewNode** constructors can be used to create a new instance of the **C1TreeView** or **C1TreeViewNode** class. Once the nodes are created, they can be added to the Node collection of a new node or treeview.

For example:

- Visual Basic

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
```

```
    'create an instance of the class
```

```
    Dim treeView As New C1TreeView()
```

```
    Placeholder1.Controls.Add(treeView)
```

```
    If Not Page.IsPostBack Then
```

```
        Dim P As New C1TreeViewNode()
```

```
        P.Text = "Products"
```

```
        P.Value = "PS"
```

```
        P.Expanded = True
```

```
        treeView.Nodes.Add(P)
```

```
        Dim Pr1 As New C1TreeViewNode()
```

```
        Pr1.Text = "Product 1"
```

```
        Pr1.Value = "Pr1"
```

```
        Pr1.Expanded = True
```

```
        P.Nodes.Add(Pr1)
```

```
        Dim Oview1 As New C1TreeViewNode()
```

```
        Oview1.Text = "Overview"
```

```
        Oview1.Value = "Oview1"
```

```
        Pr1.Nodes.Add(Oview1)
```

```
        Dim Down1 As New C1TreeViewNode()
```

```
        Down1.Text = "Downloads"
```

```
        Down1.Value = "Down1"
```

```
        Pr1.Nodes.Add(Down1)
```

```

Dim Supp1 As New C1TreeViewNode()
Supp1.Text = "Support"
Supp1.Value = "Supp1"
Pr1.Nodes.Add(Supp1)

Dim Pr2 As New C1TreeViewNode()
Pr2.Text = "Products 2"
Pr2.Value = "Pr2"
Pr2.Expanded = True
P.Nodes.Add(Pr2)

Dim Oview2 As New C1TreeViewNode()
Oview2.Text = "Overview"
Oview2.Value = "Oview2"
Pr2.Nodes.Add(Oview2)

Dim Down2 As New C1TreeViewNode()
Down2.Text = "Downloads"
Down2.Value = "Down2"
Pr2.Nodes.Add(Down2)

Dim Supp2 As New C1TreeViewNode()
Supp2.Text = "Support"
Supp2.Value = "Supp2"

Pr2.Nodes.Add(Supp2)
End If

```

End Sub

- **C#**

```

protected void Page_Load(object sender, EventArgs e)
{
    //create an instance of the class
    C1TreeView treeView = new C1TreeView();
    Placeholder1.Controls.Add(treeView);

    if (!Page.IsPostBack)

```

```
{

    C1TreeViewNode P = new C1TreeViewNode();
    P.Text = "Products";
    P.Value = "PS";
    P.Expanded = true;
    treeView.Nodes.Add(P);

    C1TreeViewNode Pr1 = new C1TreeViewNode();
    Pr1.Text = "Product 1";
    Pr1.Value = "Pr1";
    Pr1.Expanded = true;
    P.Nodes.Add(Pr1);

    C1TreeViewNode Oview1 = new C1TreeViewNode();
    Oview1.Text = "Overview";
    Oview1.Value = "Oview1";
    Pr1.Nodes.Add(Oview1);

    C1TreeViewNode Down1 = new C1TreeViewNode();
    Down1.Text = "Downloads";
    Down1.Value = "Down1";
    Pr1.Nodes.Add(Down1);

    C1TreeViewNode Supp1 = new C1TreeViewNode();
    Supp1.Text = "Support";
    Supp1.Value = "Supp1";
    Pr1.Nodes.Add(Supp1);

    C1TreeViewNode Pr2 = new C1TreeViewNode();
    Pr2.Text = "Products 2";
    Pr2.Value = "Pr2";
    Pr2.Expanded = true;
    P.Nodes.Add(Pr2);

    C1TreeViewNode Oview2 = new C1TreeViewNode();
    Oview2.Text = "Overview";
    Oview2.Value = "Oview2";
```

```

Pr2.Nodes.Add(Oview2);

C1TreeNode Down2 = new C1TreeNode();
Down2.Text = "Downloads";
Down2.Value = "Down2";
Pr2.Nodes.Add(Down2);

C1TreeNode Supp2 = new C1TreeNode();
Supp2.Text = "Support";
Supp2.Value = "Supp2";
Pr2.Nodes.Add(Supp2);

}

}

```

# TreeView Appearance and Behavior

The following topics describe the appearance and behavior of the **C1TreeView** control.

## C1TreeView Themes

The **C1TreeView** 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 **C1TreeView** control as follows:

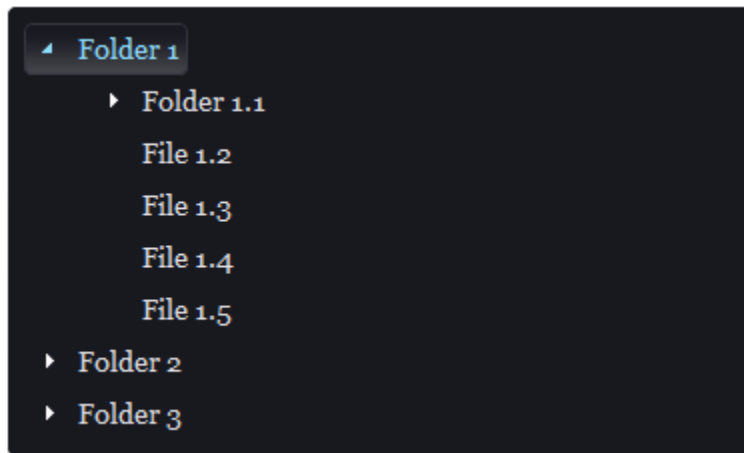
Aristo



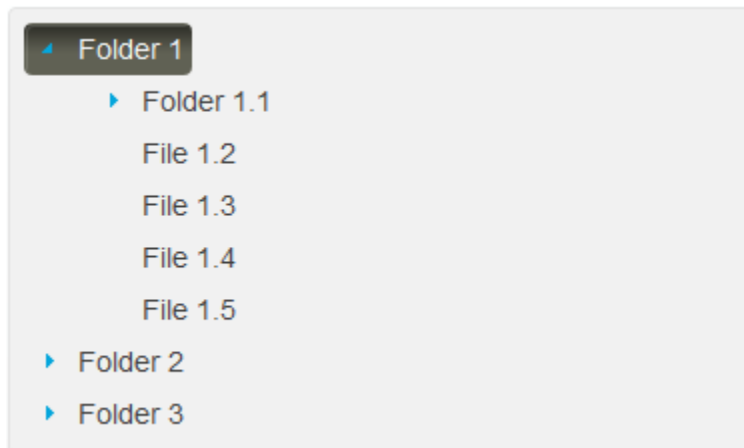
Cobalt



Midnight



Rocket



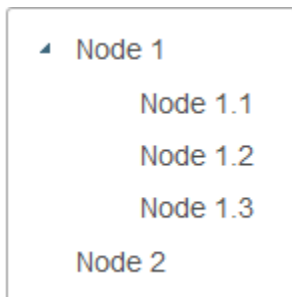
Sterling



To set the theme of the `C1TreeView` control, simply set its `Theme` property to one of the built-in themes.

## Check Boxes

You can display check boxes next to each `C1TreeViewNode` when `ShowCheckBoxes` is set to true. When the check boxes are enabled for the `C1TreeView` you can use the `NodeCheckChanged` to create an action whenever the status of a check box changes between posts. If you want to respond immediately to changes in the check boxes on the client without postback you can use the `OnClientNodeCheckChanged` server-side event property.



### Tri-State Checkboxes

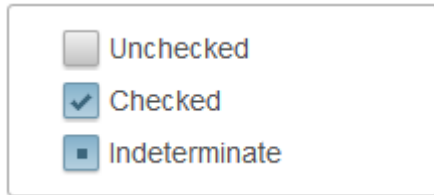
When the `ShowCheckBoxes` and `AllowTriState` properties are set to true you can use three types of checkbox states that appear next to the `C1TreeViewNode`.

The following table describes the three check box states and how it visually affects each checkbox next to the `C1TreeViewNodes`.

The property `CheckState` specifies the check state of `C1TreeViewNode`.

Checkbox State	Description
Indeterminate	A dark shaded gray box appears in the parent node when only a few of the child nodes are selected.
Checked	A checkmark appears in the parent node when all of its child nodes are selected.
Unchecked	An empty checkbox appears in the parent node when none of the parent node or child nodes are selected.

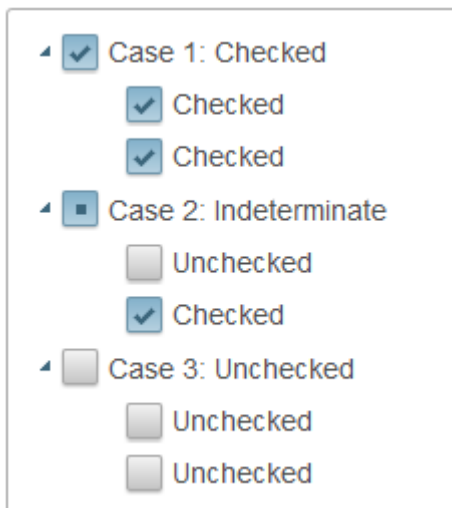
The following image displays each check box state for the **C1TreeView** control: indeterminate, checked, and unchecked.



### Tri-State's Effect on Child Nodes

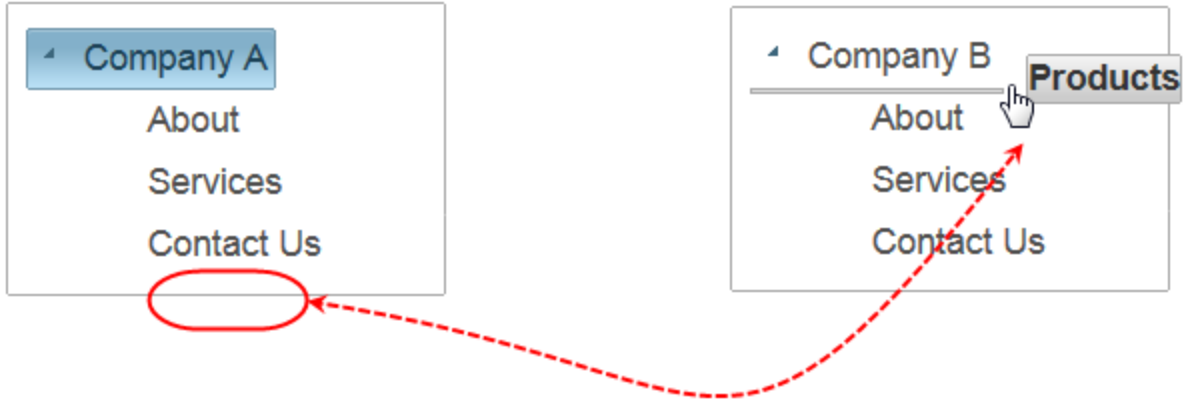
If **AllowTriState** is set to **True** and a node of **C1Treeview** has child nodes, its **CheckState** is determined by the **CheckState** of its children. There are three cases which are as follows:

- **Case 1**  
All of the **Checked** properties of child nodes are set to **True**, in which case the parent node's **Checked** property would be **True** and the **CheckState** is set to **Checked** automatically.
- **Case 2**  
Some of the child nodes' **Checked** properties are set to **True**, in which case the parent node's **Checked** property will be **True**, but the **CheckState** property will be **Indeterminate**.
- **Case 3**  
All child nodes' **Checked** properties are set to **False**, in which case the parent node's **Checked** property will be **False**, and the **CheckState** property will be **UnChecked**.



### Drag and Drop Nodes

You can drag-and-drop **C1TreeViewNodes** on nodes, in between nodes, or from one tree to another tree when the **AllowDrag** and **AllowDrop** properties are set to **True**. The following image shows a **C1TreeViewNode** being dragged from one **C1TreeView** to another **C1TreeView**. A vertical gray line is used as a visual cue to show you where the **C1TreeViewNode** is going to be dropped.



When a **CITreeViewNode** indicates a dropped node, it generates a **NodeDropped** server-side event. The event handler for the **NodeDropped** can perform a specific action by locating the dropped node. If you want to respond immediately when a node is being **dragged** or when it's dropped on the client without postback you can use the **OnClientNodeDragStarted**, **OnClientNodeDragging**, and **OnClientNodeDropped** server-side property events.

## Load on Demand

If you have a tree that contains many nodes and you only want vital information to be sent to the server you can set the **LoadOnDemand** and **AutoPostBack** properties to **True**.

## Node Selection

When you click on a node at run time it is automatically marked as selected. Clicking a node will raise the **SelectedNodesChanged** event to provide custom functionality. To have the nodes marked as selected without clicking them you can enable the **Selected** property. Multiple nodes can be selected at one time by holding down the control key while mouse clicking multiple nodes. To unselect a node, click on it again. The nodes are marked as selected in the following **CITreeView**:



## Node Navigation

**CITreeView** supports mouse and keyboard navigation.

### Navigating **CITreeViewNodes** using the mouse

The following table describes the actions and corresponding mouse commands when navigating through the **CITreeViewNodes**:

Action	Mouse Command
Expand a node	Click on the plus sign at the left of the node's name.
Collapse a node	Click on the minus sign at the left of the node's name.
Select a node	Click on the node's name.

### Navigating **CITreeViewNodes** using the keyboard

The following table describes the actions and their associated keys to use when navigating through **C1TreeViewNodes**:

Action	Keyboard Command
Expand a node	+ KEY
Collapse a node	- KEY
Move up a node	UP ARROW KEY
Move down a down	DOWN ARROW KEY
Select multiple nodes	MOUSE + CTRL KEY

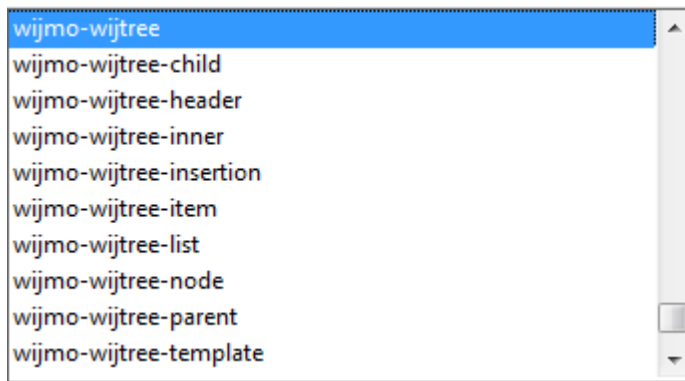
## C1TreeView CSS Selectors

You can style many C1TreeView elements using CSS to make their appearance unique. To make this customization easier, ComponentOne includes CSS selectors with each of its six built-in themes.

You can apply general CSS properties such as border, background, text, font, margin, padding, list, outline, and table to applicable CSS selectors.

For a list of common individual CSS selectors and grouped CSS selectors, select the C1TreeView control in your project and view the drop-down list next to the **CssClass** property in the Visual Studio Properties window.

C1TreeView **CSS** selectors begin with `wijmo-wjtree`:





# TreeView for ASP.NET Wijmo Task-Based Help

The task-based help section assumes that you are familiar with programming in the Visual Studio ASP.NET environment, and know how to use the C1TreeView control in general. Each topic provides a solution for specific tasks using the C1TreeView control. Each task-based help topic also assumes that you have created a new ASP.NET project.

## Creating and Configuring Check Box Nodes

This section contains several tasks that will help you create and configure check boxes for C1TreeViewNodes. For more information about check boxes, see [Check Boxes](#) (page 40).

### Creating Node Check Boxes

To create a C1TreeView filled with check box nodes, simply set the C1TreeView's ShowCheckBoxes property to True and all of the nodes contained within the control will adopt check boxes.

#### In Design View

Complete the following steps:

1. Click the smart tag to open the **C1TreeView Tasks** menu. Select **Edit TreeView**.  
The **C1TreeView Designer Form** dialog box opens.
2. In the designer treeview, select the C1TreeView control (**C1TreeView1** by default).
3. Navigate to the properties grid, locate the ShowCheckBoxes property, and set it to **True**.

#### In Source View

Add `ShowCheckBoxes = True` to the `<wijmo:C1TreeView>` tag so that the markup resembles the following:

```
<wijmo:C1TreeView ID="C1TreeView1" runat="server"
ShowCheckBoxes="True">
```

#### In Code

Add the following code snippet to the **Page\_Load** event:

- Visual Basic  

```
C1TreeView1.ShowCheckBoxes="True"
```
- C#  

```
C1TreeView1.ShowCheckBoxes="True";
```

## Preventing Child Nodes from Being Automatically Checked

When check boxes are utilized in a C1TreeView, clicking a parent node will automatically check all of its child nodes as well. To prevent this, you can set the C1TreeView's AutoCheckNodes property to **False**.

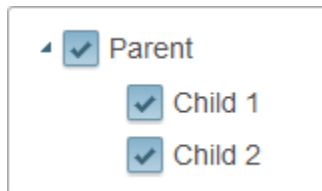
In this tutorial, you will create a C1TreeView control with one parent node and two child nodes. You will change the nodes to check boxes and then set the AutoCheckNodes property to **False** so the child nodes' Checked property settings will be independent of its parent node's Checked property setting.

Complete the following:

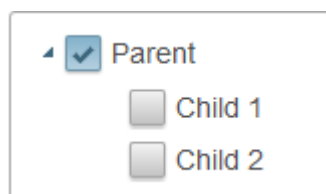
1. Click the **Source** tab to switch to Source view.
2. Add the following markup between the `<wijmo:C1TreeView>` tags:

```
<Nodes>
  <wijmo:C1TreeViewNode runat="server" Text="Parent">
    <Nodes>
      <wijmo:C1TreeViewNode runat="server" Text="Child 1">
      </wijmo:C1TreeViewNode>
      <wijmo:C1TreeViewNode runat="server" Text="Child 2">
      </wijmo:C1TreeViewNode>
    </Nodes>
  </wijmo:C1TreeViewNode>
</Nodes>
```

3. This markup creates a parent node with two child nodes.
4. Enable the check boxes by adding `ShowCheckBoxes="True"` to the `<wijmo:C1TreeView>` tag.
5. Press F5 to run the project and complete the following to see how the check boxes behave by default:
  - a. Expand the **Parent** node to reveal the TreeView hierarchy.
  - b. Click the **Parent** check box and observe that the child nodes are automatically selected.



- c. Terminate the program.
6. Click **Design** tab to enter Design view.
  7. In the Properties window, select **C1TreeView1** from the drop-down list and then set the `AutoCheckNodes` property to **False**.
  8. Press F5 to run the project and complete the following to see how the check boxes behave when the nodes are no longer auto-checked:
    - a. Expand the **Parent** node to reveal the TreeView hierarchy.
    - b. Check the **Parent** check box and observe that the child tabs don't get checked automatically.



**Tip:** If you check the **Child 1** check box and leave the **Child 2** check box, you'll notice that the **Parent** check box contains an "indeterminate" mark. If you want to prevent this from happening, set the C1TreeView control's AllowTriState property to **False**.

## Working with Themes


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

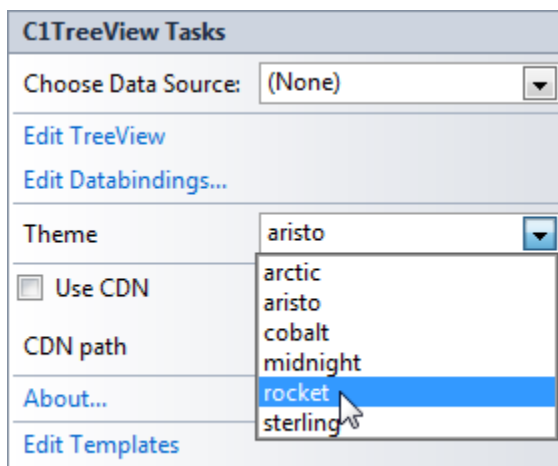
### Using a Built-In Theme

A C1TreeView 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 [C1TreeView Themes](#) (page 38).

### Changing the Theme in Design View

Complete the following steps:

1. Click the **C1TreeView** smart tag  to open the **C1TreeView 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 C1TreeView control.

### Changing the Theme in Source View

To change the theme of your **C1TreeView** in Source view, add `VisualStyle="rocket"` to the `<wijmo:C1TreeView>` tag so that it resembles the following:

```
<wijmo:C1TreeView ID="C1TreeView1" 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

```
C1TreeView1.Theme = "rocket"
```

- C#

```
C1TreeView1.Theme = "rocket";
```

3. Run the program.

✔ **This topic illustrates the following:**

The following image shows a **C1TreeView** control with the **rocket** theme:



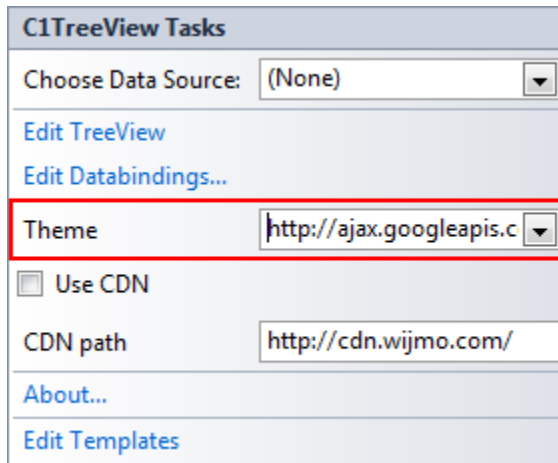
### Using a Custom Theme

**Tabs for ASP.NET Wijmo** provides six built-in themes, but if you prefer to use a different theme, you can choose an existing theme using a CDN URL or create your own custom theme with the jQuery ThemeRoller Web application. We will use C1TreeView in the following examples.

#### Using a CDN URL

Complete the following steps:

1. Click the C1TreeView smart tag to open the **C1TreeView Tasks** menu.
2. Select the **Use CDN** check box.
3. In the **Theme** property, enter a CDN URL to specify the theme; CDN URLs can be found at <http://blog.jqueryui.com/2011/06/jquery-ui-1-8-14/>. In this example, we'll use the *trontastic* theme: <http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.14/themes/trontastic/jquery-ui.css>.



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

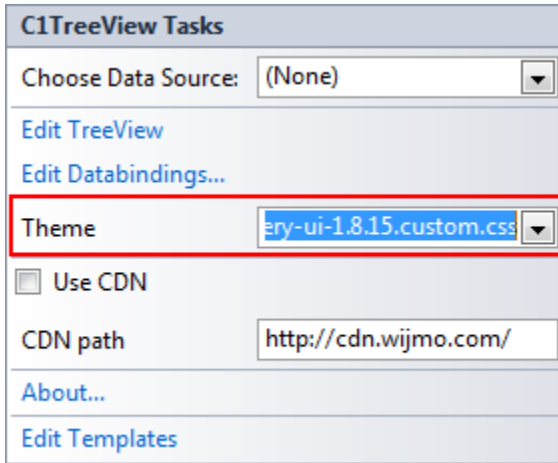
4. Run the project and notice the theme is applied to C1TreeView.



### Using jQuery ThemeRoller

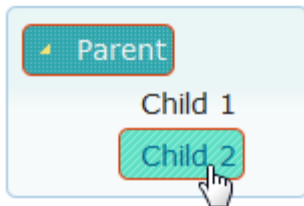
Complete the following steps:

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 C1TreeView smart tag to open the **Tasks** menu.
7. Select the **Use CDN** check box.
8. In the **Theme** property, enter the path to your custom theme .css; for example, `custom-theme/css/custom-theme/jquery-ui-1.8.15.custom.css`.



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

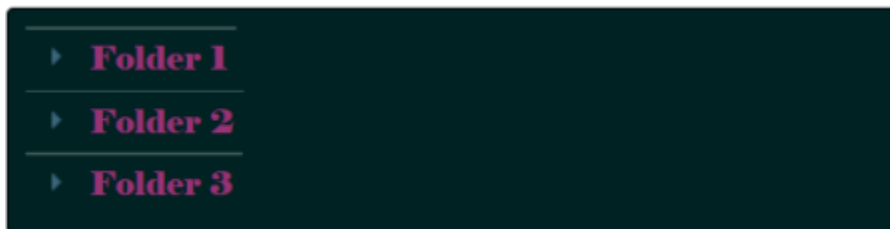
- Run the project and notice the theme is applied to C1TreeView.



## Working with C1TreeView CSS Selectors

C1TreeView allows you to fully customize the control's appearance using CSS selectors. This topic will walk you through setting CSS selectors to customize the appearance of C1TreeView.

- Begin in Design View and navigate to the Properties window for your C1TreeView control.
- Locate the `CssClass` property and use the drop-down menu to set it to `wijmo-wijtree`.
- Switch to your Source View and locate the first set of `<asp:Content>` tags on the page. Add `<style type="text/css"></style>` tags. These tags will allow you to add CSS styling.
- Add `.wijmo-wijtree { color: #993377; background: #02222; border-color: Black; font-family: Elephant; }` to the tags.
- Run your program. The C1TreeView control should appear as in the following image.




## Adding a Top-Level Node to a TreeView

This topic illustrates how to add a top-level node to a C1TreeView control in Design view, in Source view, and in code.

### In Design View

Complete the following steps:

1. Click the smart tag to open the **C1TreeView Tasks** menu. Select **Edit TreeView**.  
The **C1TreeView Designer Form** dialog box opens.
2. Click the **Add Child Item** button  to add a C1TreeViewNode to the C1TreeView control.
3. Click **OK** to close the **C1TreeView Design Form** dialog box.

### In Source View

Add the following markup between the <wijmo:C1TreeView> tags:

```
<wijmo:C1TreeViewNode ID="Node1" runat="server" Text="Node1">
</wijmo:C1TreeViewNode>
```

### In Code View

Complete the following steps:

1. Import the following namespace into your project:
  - Visual Basic  
`Imports C1.Web.Wijmo.Controls.C1TreeView`
  - C#  
`using C1.Web.Wijmo.Controls.C1TreeView;`
2. Add the following code to the **Page\_Load** event:
  - Visual Basic  
`Dim TreeViewNode1 As New C1TreeViewNode()  
C1TreeViewNode1.Text = "C1TreeViewNode1"  
C1TreeView1.Nodes.Add(C1TreeViewNode1)`
  - C#  
`C1TreeViewNode TreeViewNode1 = new C1TreeViewNode();  
C1TreeViewNode1.Text = "C1TreeViewNode1";  
C1TreeView1.Nodes.Add(C1TreeViewNode1);`
3. Run the program.

## Adding a Child Node to a TreeView Node

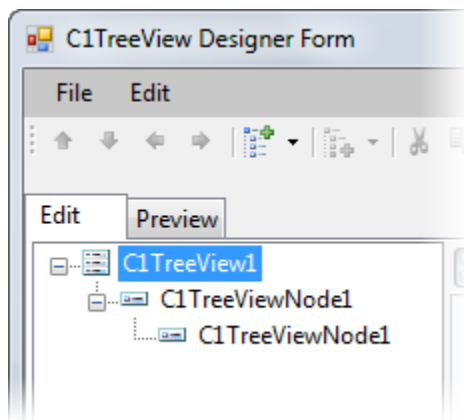
This topic illustrates how to add a child node to a C1TreeViewNode control in Design view, in Source view, and in code. This topic assumes that you have completed [Adding a Top-Level Node to a TreeView](#) (page 51).

### In Design View

Complete the following steps:

1. Click the smart tag to open the **C1TreeView Tasks** menu. Select **Edit TreeView**.  
The **C1TreeView Designer Form** dialog box opens.
2. Select the node you wish to add the child node to.

3. Click the **Add Child Item** button to add the child node to the node you select. The treeview of the designer form will resemble the following:



4. Click **OK** to close the **C1TreeView Design Form** dialog box.

#### In Source View

Add the following markup between the `<wijmo:C1TreeViewNode>` tags of the node you wish to add the child node to:

```
<Nodes>
  <wijmo:C1TreeViewNode ID="Node1" runat="server" Text="Node1">
    </wijmo:C1TreeViewNode>
  </Nodes>
```

#### In Code View

Complete the following steps:

4. Import the following namespace into your project:

- Visual Basic

```
Imports C1.Web.Wijmo.Controls.C1TreeView
```

- C#

```
using C1.Web.Wijmo.Controls.C1TreeView;
```

5. Add the following code to the **Page\_Load** event:

- Visual Basic

```
' Create first node and add it to the C1TreeView.
Dim C1TreeViewNode1 As New C1TreeViewNode()
C1TreeViewNode1.Text = "C1TreeViewNode1"
C1TreeView1.Nodes.Add(C1TreeViewNode1)

' Create the child node and add it to C1TreeViewNode1
Dim C1TreeViewNode2 As New C1TreeViewNode()
C1TreeViewNode2.Text = "C1TreeViewNode1"
C1TreeViewNode1.Nodes.Add(C1TreeViewNode2)
```

- C#

```
// Create first node and add it to the C1TreeView.
C1TreeViewNode C1TreeViewNode1 = new C1TreeViewNode();
```

```

C1TreeNode1.Text = "C1TreeNode1";
C1TreeView1.Nodes.Add(C1TreeNode1);

// Create the child node and add it to C1TreeNode1
C1TreeNode C1TreeNode2 = new C1TreeNode();
C1TreeNode2.Text = "C1TreeNode2";
C1TreeNode1.Nodes.Add(C1TreeNode2);

```

6. Run the program.

---

## Populating C1TreeView with a Site Map

This lesson illustrates how to populate a C1TreeView with site map data.

To create the Site Map and bind it to the **C1TreeView** control, complete the following:

1. In the Solution Explorer, right-click the project's name and select **Add New Item**.  
The **Add New Item** dialog box appears.
2. Select **Site Map** from the list of templates, and then click **Add** to add the new **Web.sitemap** page to the project.

The following default source code appears for the **Web.sitemap** file:

```

<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="" title="" description="">
    <siteMapNode url="" title="" description="" />
    <siteMapNode url="" title="" description="" />
  </siteMapNode>
</siteMap>

```

3. Replace the default data with the following data for the Web.sitemap file:

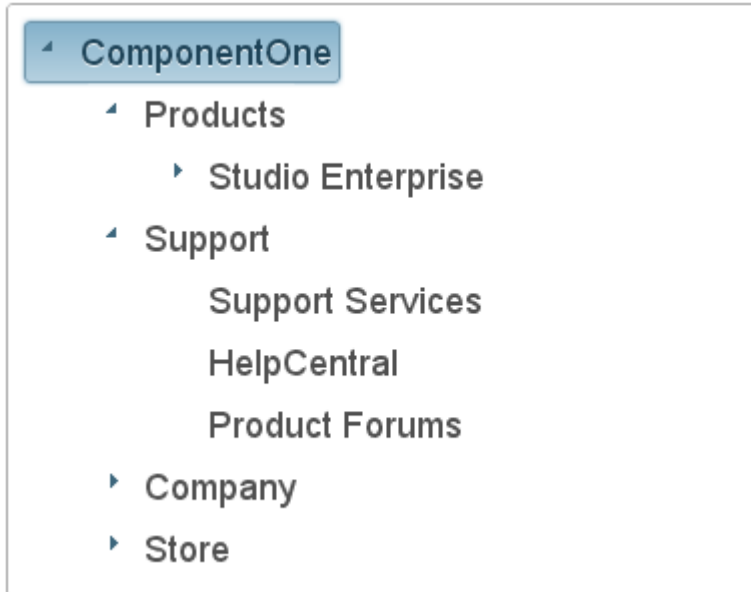
```

<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode title="ComponentOne">
    <siteMapNode title="Products">
      <siteMapNode title="Studio Enterprise">
        <siteMapNode title="Studio for WinForms" />
        <siteMapNode title="Studio for ASP.NET" />
        <siteMapNode title="Studio for WPF" />
        <siteMapNode title="Studio for Mobile" />
        <siteMapNode title="Studio for ActiveX" />
        <siteMapNode title="Studio for Silverlight" />
      </siteMapNode>
    </siteMapNode>
  </siteMapNode>

```

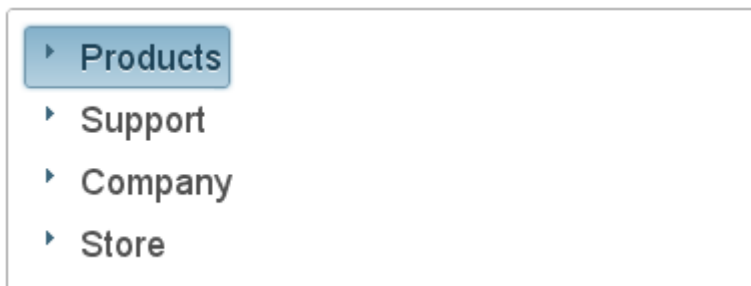
```
<siteMapNode title="Support">
  <siteMapNode title="Support Services" />
  <siteMapNode title="HelpCentral" />
  <siteMapNode title="Product Forums" />
</siteMapNode>
<siteMapNode title="Company">
  <siteMapNode title="About Us" />
  <siteMapNode title="Partners" />
  <siteMapNode title="Contact Us" />
  <siteMapNode title="Join Us" />
  <siteMapNode title="Press Center" />
  <siteMapNode title="Governance" />
</siteMapNode>
<siteMapNode title="Store">
  <siteMapNode title="Buy Now" />
  <siteMapNode title="Resellers" />
</siteMapNode>
</siteMapNode>
</siteMap>
```

4. Open **CITreeview** control's Tasks menu and click the **Choose Data Source** drop-down arrow. **Select New Data Source** to open the **Data Source Configuration Wizard**.
5. Select **Site Map** and click **OK**.  
**SiteMapDataSource1** is added to your project.
6. Press F5 to run the project and observe the following:  
The data from the **Web.sitemap** file is reflected in the **CITreeView** control.



Observe that the control opens with the top-level node, ComponentOne. In the next step, you'll learn how to remove the top-level node so that you just expose the second-level nodes in the C1TreeView.

7. Close the browser and return to the project.
8. In Design view, select **SiteMapDataSource** and, in the Properties window, set the **ShowStartingNode** to **False**.
9. Press F5 to run the project and observe that the top-level node has been removed.



## Populating C1TreeView with XML

This tutorial teaches you how installed templates, add the XML Data Source component to the Web site, assign it to the **C1TreeView** control, and then set the binding for the **C1TreeView**.

Complete the following steps:

1. From the Toolbox, double-click the C1TreeView icon to add the control to your project.
2. Create and prepare the XML file by completing the following steps:
3. Right-click the **App\_Data** in the Solution Explorer and select **Add New Item**. The **Add New Item** dialog box appears.
  - a. Select the XML File and rename it "TreeView\_Hierarchy.xml". The XML file opens.
  - b. In the XML view, add the following data to the **TreeViewHierarchy.xml** document:

```

<?xml version="1.0" encoding="utf-8" ?>
<root>
  <TreeViewNode Text="CDs">\
    <TreeViewNode Text="Back to Black"></TreeViewNode>
    <TreeViewNode Text="Frank"></TreeViewNode>
    <TreeViewNode Text="Nevermind"></TreeViewNode>
    <TreeViewNode Text="In Utero"></TreeViewNode>
  </TreeViewNode>
  <TreeViewNode Text="Cassette Tapes">
    <TreeViewNode Text="Bleach"></TreeViewNode>
    <TreeViewNode Text="Cheap Thrills"></TreeViewNode>
    <TreeViewNode Text="Dangerous"></TreeViewNode>
    <TreeViewNode Text="Bad"></TreeViewNode>
  </TreeViewNode>
  <TreeViewNode Text="Vinyl Records">
    <TreeViewNode Text="Axis: Bold as Love"></TreeViewNode>
    <TreeViewNode Text="Full Circle"></TreeViewNode>
    <TreeViewNode Text="Off The Wall"></TreeViewNode>
    <TreeViewNode Text="Other Voices"></TreeViewNode>
  </TreeViewNode>
</root>

```

4. Switch back to the Design view of the **.aspx page** and complete the following steps to create a new data source:
  - a. Click **CITreeView**'s smart tag to open the **CITreeView Tasks** menu and then, from the **Choose Data Source** drop-down list, select **New Data Source**.
  - b. The **Data Source Configuration Wizard** dialog box opens.
  - c. Select **XML File** and then click **OK**.

**XmlDataSource1** is added to the project.
5. Complete the following steps to configure the data source:
  - a. Click **CITreeView**'s smart tag to open the **CITreeView Tasks** menu; click **Configure Data Source**. The **Configure Data Source** dialog box opens.
  - b. In the XPath expression text field, enter "root/TreeViewNode". This will select all TreeViewNodes that are children of root so that the TreeViewNodes are the top-level nodes on the Web page.
  - c. Next to **Data file** text field, click **Browse** to open the **Select XML File** dialog box.
  - d. Select the **App\_Data** project folder , and then select **TreeView\_Hierarchy.xml** from the **Contents of folder** pane.
  - e. Click **OK** to close the **Select XML File** dialog box.

- f. Click **OK** to close the **Configure Data Source** dialog box.
6. Complete the following steps to bind the XML tags to the **C1TreeViewNodes**.
  - a. Click **C1TreeView**'s smart tag to open the **C1TreeView Tasks** menu and click **Edit Databindings**.  
**The Bindings Collection Editor** dialog box opens.
  - b. Click **Add** to add an empty binding to the project.
  - c. Set the binding's properties as follows:
    - Set the **DataMember** property to "TreeViewNode".
    - Set the **TextField** property to "Text".
  - d. Click **OK** to close the **Bindings Collection Editor**.
7. Press F5 to run the project.

Observe that the data from the **TreeView\_Hierarchy.xml** file is reflected in the **C1TreeView** control.



## Saving and Loading a C1TreeView from XML

The following tasks show you how to save your C1TreeView control as an .xml file and then load it into your project using the designer.

## Save the C1TreeView as XML

To save your tree as an XML file using the designer:

1. Click **C1TreeView**'s smart tag and select **Edit TreeView** to open the **C1TreeView Designer Form**.
2. Navigate to **File | Save as XML**.
3. Name the .xml file for your **C1TreeView** and browse to where you would like to save it.
4. Click **OK** to close the **TreeView Designer Form** dialog box.

## Load an Existing XML C1TreeView into your Project

To load the **C1TreeView** control you saved as an .xml file into your project:

1. Click **C1TreeView**'s smart tag and select **Edit TreeView** to open the **C1TreeView Designer Form**.
2. Open the **C1TreeView Designer Form**.
3. Navigate to **File | Load From XML** and click open to open the existing .xml file.

## Load an Existing XML C1TreeView in Code

To load the **C1TreeView** control you saved as an .xml file into your project:

1. Create an XML file for the C1TreeView structure.
2. Call the LoadLayout method to load the items, passing in the path to the file:
  - Visual Basic

```
C1TreeView1.LoadLayout("c:\\Visual Studio
2005\\WebSites\\LoadLayoutEX\\App_Data\\C1TreeViewControl.xml")
```
  - C#

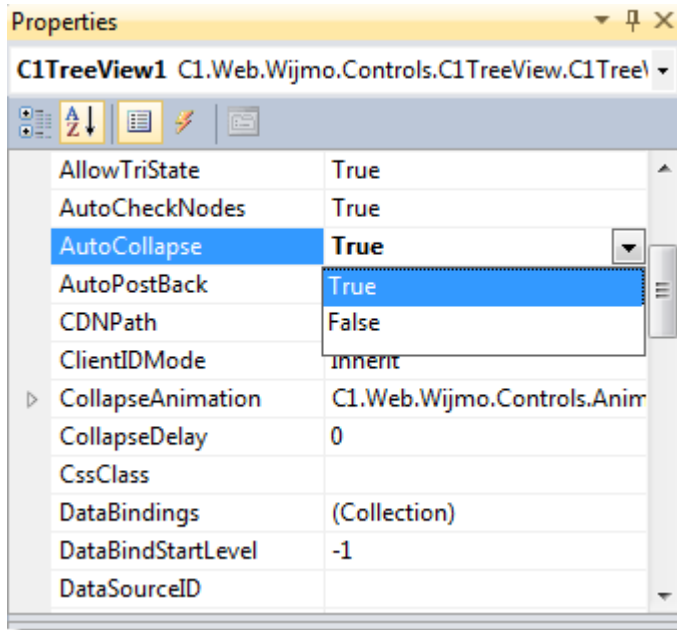
```
C1TreeView1.LoadLayout("c:\\Visual Studio
2005\\WebSites\\LoadLayoutEX\\App_Data\\C1TreeViewControl.xml");
```
3. Press F5 to run the program.

## Setting the Auto Collapse Property

Autocollapse causes all expanded nodes to collapse if another node is expanded. This task-based help topic will walk you through setting the Autocollapse property in Design View and in Source View.

### In Design View

1. In the Design View, navigate to the **C1TreeView** Properties window.
2. Locate the **AutoCollapse** property and use the drop-down menu to set the property to "true".



3. Press F5 to run your program. Note that when one node is expanded, any other expanded nodes will automatically collapse.

#### In Source View

1. In the Source View, add the following markup to create the nodes for C1TreeView.

```

<Nodes>
    <wijmo:C1TreeViewNode Text="Folder 1">
        <Nodes>
            <wijmo:C1TreeViewNode Text="Folder 1.1">
                <Nodes>
                    <wijmo:C1TreeViewNode Text="Folder 1.1.1">
                    </wijmo:C1TreeViewNode>
                    <wijmo:C1TreeViewNode Text="Folder 1.1.2">
                    </wijmo:C1TreeViewNode>
                    <wijmo:C1TreeViewNode Text="Folder 1.1.3">
                    </wijmo:C1TreeViewNode>
                    <wijmo:C1TreeViewNode Text="Folder 1.1.4">
                    </wijmo:C1TreeViewNode>
                </Nodes>
            </wijmo:C1TreeViewNode>
            <wijmo:C1TreeViewNode Text="Folder 1.2">
            </wijmo:C1TreeViewNode>
            <wijmo:C1TreeViewNode Text="Folder 1.3">
            </wijmo:C1TreeViewNode>
        </Nodes>
    </wijmo:C1TreeViewNode>

```

```

        <wijmo:C1TreeViewNode Text="Folder 1.4">
        </wijmo:C1TreeViewNode>
        <wijmo:C1TreeViewNode Text="Folder 1.5">
        </wijmo:C1TreeViewNode>
    </Nodes>
</wijmo:C1TreeViewNode>
<wijmo:C1TreeViewNode Text="Folder 2">
    <Nodes>
        <wijmo:C1TreeViewNode Text="Folder 2.1">
        </wijmo:C1TreeViewNode>
        <wijmo:C1TreeViewNode Text="Folder 2.2">
        </wijmo:C1TreeViewNode>
        <wijmo:C1TreeViewNode Text="Folder 2.3">
        </wijmo:C1TreeViewNode>
        <wijmo:C1TreeViewNode Text="Folder 2.4">
        </wijmo:C1TreeViewNode>
        <wijmo:C1TreeViewNode Text="Folder 2.5">
        </wijmo:C1TreeViewNode>
    </Nodes>
</wijmo:C1TreeViewNode>
<wijmo:C1TreeViewNode Text="Folder 3">
    <Nodes>
        <wijmo:C1TreeViewNode Text="Folder 3.1">
        </wijmo:C1TreeViewNode>
        <wijmo:C1TreeViewNode Text="Folder 3.2">
        </wijmo:C1TreeViewNode>
        <wijmo:C1TreeViewNode Text="Folder 3.3">
        </wijmo:C1TreeViewNode>
        <wijmo:C1TreeViewNode Text="Folder 3.4">
        </wijmo:C1TreeViewNode>
        <wijmo:C1TreeViewNode Text="Folder 3.5">
        </wijmo:C1TreeViewNode>
    </Nodes>
</wijmo:C1TreeViewNode>
</Nodes>

```

2. Add `Autocollapse="true"` to the `<wijmo:C1TreeView>` tags as in the following sample:

```
<wijmo:C1TreeView ID="C1TreeView1" runat="server" AutoCollapse="true">
```

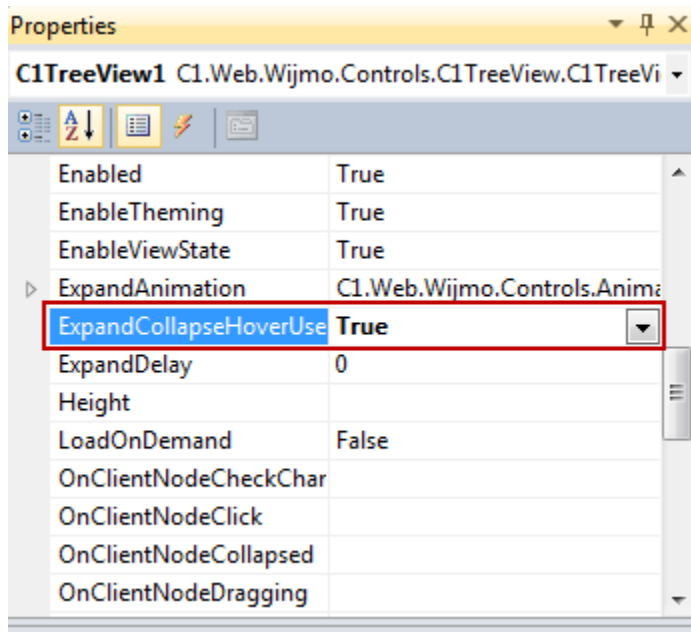
3. Press F5 to run your program. Note that when one node is expanded, any other expanded node will automatically collapse.

## Setting C1TreeView to Open on Hover

C1TreeView supports expanding nodes when they are hovered over as an alternative to expanding on a mouse click. This topic will walk you through setting the **ExpandCollapseHoverUsed** property.

### In Design View

1. In the Design View, navigate to the C1TreeView Properties window and locate the **ExpandCollapseHoverUsed** property.



2. Use the drop-down menu to set the property to "true".
3. Run your project. Note that the nodes will expand when hovered over with the mouse.

### In Source View

Add `ExpandCollapseHoverUse="true"` to the `<wijmo:C1TreeView>` tags to resemble the following sample:

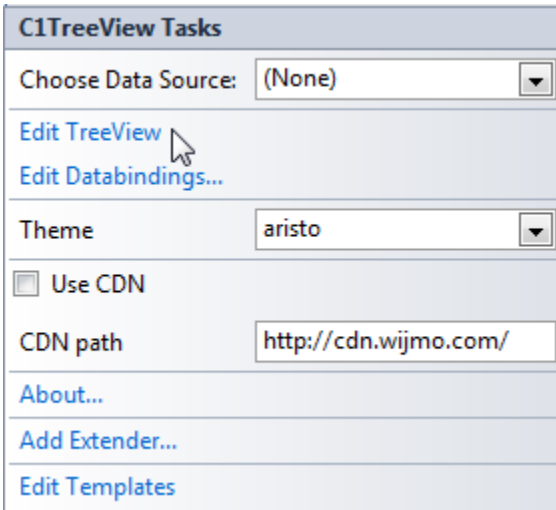
```
<wijmo:C1TreeView ID="C1TreeView1" runat="server" AutoCollapse="True"
    ExpandCollapseHoverUsed="True">
```

## Setting C1TreeView Node Icons

C1TreeView allows you to set node icons. In addition, you can switch icons based on C1TreeView's current state. This topic will walk you through setting the properties to display node icons and to change node icons based on C1TreeView's state.

### In Design View

1. Click the C1TreeView smart tag  to open the C1TreeView Tasks Menu.
2. Click **Edit TreeView** to open the C1TreeView Designer Form.



3. Select the first node, Folder 1, to display its properties in the Properties Window.
4. Locate the **CollapsedIconClass** property and set it to **ui-icon-folder-collapsed**.
5. Locate the **ExpandedIconClass** property and set it to **ui-icon-folder-expanded**.
6. Locate the **ItemIconClass** property and set it to **ui-icon-document**. The Designer Form Properties Window should resemble the following image:

Misc	
(ID)	
CollapsedIconClass	<b>ui-icon-folder-collapsed</b>
DisplayVisible	True
Expanded	False
ExpandedIconClass	<b>ui-icon-folder-open</b>
ItemIconClass	<b>ui-icon-document</b>
Selected	False
Text	<b>Folder 1</b>
Url	

7. Press F5 to run your project. Your C1TreeView control should resemble the following image.

- ▶
▶ Folder 1
  - ▶ Folder 1.1
    - Folder 1.2
    - Folder 1.3
    - Folder 1.4
    - Folder 1.5
  - ▶ Folder 2
  - ▶ Folder 3

### In Source View

1. In the Source View, locate the first set of `<wijmo:C1TreeNode>` tags, the **Text** property of which should read "Folder 1".

2. Insert the following markup to set the node icons:

```
CollapsedIconClass="ui-icon-folder-collapsed"
ExpandedIconClass="ui-icon-folder-open"
ItemIconClass="ui-icon-document"
```

3. Your `<wijmo:C1TreeNode>` tags should resemble the following sample.

```
<wijmo:C1TreeNode Text="Folder 1"
CollapsedIconClass="ui-icon-folder-collapsed"
ExpandedIconClass="ui-icon-folder-open" ItemIconClass="ui-icon-
document">
```

4. Locate the `<wijmo:C1TreeNode>` tags for "Folder 1.1" and insert the following markup.

```
CollapsedIconClass="ui-icon-folder-collapsed"
ExpandedIconClass="ui-icon-folder-open"
ItemIconClass="ui-icon-document"
```

5. Your `<wijmo:C1TreeNode>` tags should resemble the following.

```
<wijmo:C1TreeNode Text="Folder 1.1"
CollapsedIconClass="ui-icon-folder-collapsed"
ExpandedIconClass="ui-icon-folder-open" ItemIconClass="ui-icon-
document">
```

6. Locate the `<wijmo:C1TreeNode>` tags for "Folder 1.1.1" and add `ItemIconClass="ui-icon-document"` to the tags. Your `<wijmo:C1TreeNode>` tags should resemble the following sample.

```
<wijmo:C1TreeNode Text="Folder 1.1.1" ItemIconClass="ui-icon-
document">
```

7. Press F5 to run your program. Your C1TreeView control should resemble the following image.




## Setting C1TreeView Properties to Allow Drag-and-Drop Behaviors

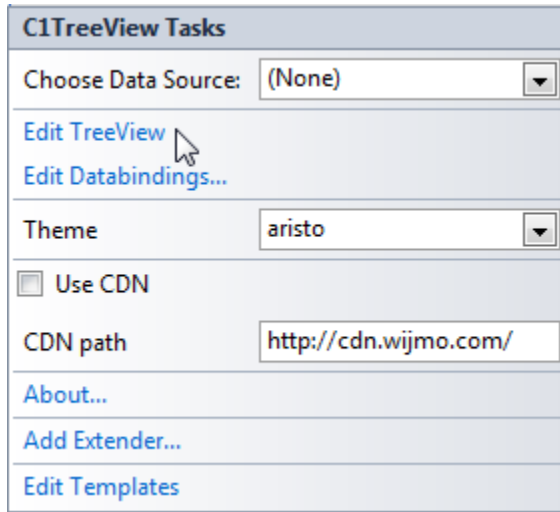
C1TreeView supports drag-and-drop behavior which allows the end-user to drag and drop nodes to rearrange the tree structure. Drag and drop node behavior can occur within one tree, or between two tree structures. This task-based help will walk you through setting the basic properties that allow drag-and-drop behavior, and the properties that allow end-users to drag and drop nodes between two tree structures.

### Drag-and-drop Behaviors Within One Tree Structure

This topic will walk you through setting C1TreeView properties to allow drag-and-drop behaviors within one tree structure.

#### In Design View

1. Click the C1TreeView smart tag  to open the C1TreeView Tasks Menu.
2. Click **Edit TreeView** to open the C1TreeView Designer Form.



3. Locate the **AllowDrag** and **AllowDrop** properties in the Designer Form Properties Window and use the drop-down menus to set both properties to "true".
4. Press F5 to run your program. Note that you are now able to rearrange the nodes.

#### In Source View

Add the following markup to the first set of `<wijmo:C1TreeView>` tags.

```
AllowDrag="True"
AllowDrop="True"
```

The `<wijmo:C1TreeView>` tags should resemble the following sample.

```
<wijmo:C1TreeView ID="C1TreeView1" runat="server" AllowDrag="True"
    AllowDrop="True">
```

Press F5 to run your program. Note that you are now able to rearrange the nodes.

### Drag-and-drop Behaviors Between Two Tree Structures

This topic will walk you through setting C1TreeView properties to allow drag-and-drop behaviors between two tree structures.

#### In Design View

1. In Design View, add two C1TreeView controls to your project and create several child nodes for each control.
2. Select the first C1TreeView control and navigate to the Properties window.
3. Locate the **AllowDrag** and **AllowDrop** properties and use the drop-down menus to set both to "true".
4. Select the second C1TreeView control and navigate to the Properties window.
5. Locate the **AllowDrop** property and use the drop-down menu to set it to "true". This will allow you to drag items from the first C1TreeView control and drop them into the second C1TreeView control at run time.
6. Press F5 to run your program. Note that you are able to drag nodes from the first tree and drop them into the second tree.

#### In Source View

1. Insert the following markup in the second set of `<asp:Content>` tags to create two C1TreeView controls.

```

<wijmo:C1TreeView ID="C1TreeView1" runat="server">
    <Nodes>
        <wijmo:C1TreeViewNode runat="server" CheckState="UnChecked"
NodeIndex="0"
            Owner="C1TreeView1" StaticKey="C1TreeView1_0"
Text="C1TreeViewNode6"
            TreeView="C1TreeView1">
        </wijmo:C1TreeViewNode>
        <wijmo:C1TreeViewNode runat="server" CheckState="UnChecked"
NodeIndex="0"
            Owner="C1TreeView1" StaticKey="C1TreeView1_1"
Text="C1TreeViewNode4"
            TreeView="C1TreeView1">
            <Nodes>
                <wijmo:C1TreeViewNode runat="server"
CheckState="UnChecked" NodeIndex="0"
                    Owner="" StaticKey="C1TreeView1_10"
Text="C1TreeViewNode1"
                    TreeView="C1TreeView1">
                </wijmo:C1TreeViewNode>
                <wijmo:C1TreeViewNode runat="server"
CheckState="UnChecked" NodeIndex="0"
                    Owner="" StaticKey="C1TreeView1_11"
Text="C1TreeViewNode2"
                    TreeView="C1TreeView1">
                </wijmo:C1TreeViewNode>
                <wijmo:C1TreeViewNode runat="server"
CheckState="UnChecked" NodeIndex="0"
                    Owner="" StaticKey="C1TreeView1_12"
Text="C1TreeViewNode3"
                    TreeView="C1TreeView1">
                </wijmo:C1TreeViewNode>
            </Nodes>
        </wijmo:C1TreeViewNode>
        <wijmo:C1TreeViewNode runat="server" CheckState="UnChecked"
NodeIndex="0"
            Owner="C1TreeView1" StaticKey="C1TreeView1_2"
Text="C1TreeViewNode5"
            TreeView="C1TreeView1">
        </wijmo:C1TreeViewNode>
        <wijmo:C1TreeViewNode runat="server" CheckState="UnChecked"
NodeIndex="0"

```

```

        Owner="C1TreeView1" StaticKey="C1TreeView1_3"
Text="C1TreeViewNode1"
        TreeView="C1TreeView1">
        <Nodes>
            <wijmo:C1TreeViewNode runat="server"
CheckState="Unchecked" NodeIndex="0"
                Owner="" StaticKey="C1TreeView1_30"
Text="C1TreeViewNode1"
                TreeView="C1TreeView1">
            </wijmo:C1TreeViewNode>
            <wijmo:C1TreeViewNode runat="server"
CheckState="Unchecked" NodeIndex="0"
                Owner="" StaticKey="C1TreeView1_31"
Text="C1TreeViewNode2"
                TreeView="C1TreeView1">
            </wijmo:C1TreeViewNode>
            <wijmo:C1TreeViewNode runat="server"
CheckState="Unchecked" NodeIndex="0"
                Owner="" StaticKey="C1TreeView1_32"
Text="C1TreeViewNode3"
                TreeView="C1TreeView1">
            </wijmo:C1TreeViewNode>
        </Nodes>
    </wijmo:C1TreeViewNode>
    <wijmo:C1TreeViewNode runat="server" CheckState="Unchecked"
NodeIndex="0"
        Owner="C1TreeView1" StaticKey="C1TreeView1_4"
Text="C1TreeViewNode2"
        TreeView="C1TreeView1">
    </wijmo:C1TreeViewNode>
    <wijmo:C1TreeViewNode runat="server" CheckState="Unchecked"
NodeIndex="0"
        Owner="C1TreeView1" StaticKey="C1TreeView1_5"
Text="C1TreeViewNode3"
        TreeView="C1TreeView1">
    <Nodes>
        <wijmo:C1TreeViewNode runat="server"
CheckState="Unchecked" NodeIndex="0"
                Owner="" StaticKey="C1TreeView1_50"
Text="C1TreeViewNode1"
                TreeView="C1TreeView1">
        </wijmo:C1TreeViewNode>

```

```

        <wijmo:C1TreeNode runat="server"
CheckState="Unchecked" NodeIndex="0"
                Owner="" StaticKey="C1TreeView1_51"
Text="C1TreeNode2"
                TreeView="C1TreeView1">
        </wijmo:C1TreeNode>
        <wijmo:C1TreeNode runat="server"
CheckState="Unchecked" NodeIndex="0"
                Owner="" StaticKey="C1TreeView1_52"
Text="C1TreeNode3"
                TreeView="C1TreeView1">
        </wijmo:C1TreeNode>
    </Nodes>
</wijmo:C1TreeNode>
    <wijmo:C1TreeNode runat="server" CheckState="Unchecked"
NodeIndex="0"
                Owner="C1TreeView1" StaticKey="C1TreeView1_6"
Text="C1TreeNode7"
                TreeView="C1TreeView1">
    </wijmo:C1TreeNode>
</Nodes>
</wijmo:C1TreeView>
<wijmo:C1TreeView ID="C1TreeView2" runat="server">
    <Nodes>
        <wijmo:C1TreeNode runat="server" CheckState="Unchecked"
NodeIndex="0"
                Text="C1TreeNode6">
        <Nodes>
            <wijmo:C1TreeNode runat="server"
CheckState="Unchecked" NodeIndex="0"
                    Text="C1TreeNode1">
            </wijmo:C1TreeNode>
            <wijmo:C1TreeNode runat="server"
CheckState="Unchecked" NodeIndex="0"
                    Text="C1TreeNode2">
            </wijmo:C1TreeNode>
        </Nodes>
    </wijmo:C1TreeNode>
    <wijmo:C1TreeNode runat="server" CheckState="Unchecked"
NodeIndex="0"
                Text="C1TreeNode7">

```

```

        </wijmo:C1TreeNode>
        <wijmo:C1TreeNode runat="server" CheckState="UnChecked"
NodeIndex="0"
            Text="C1TreeNode8">
            <Nodes>
                <wijmo:C1TreeNode runat="server"
CheckState="UnChecked" NodeIndex="0"
                    Text="C1TreeNode1">
                </wijmo:C1TreeNode>
                <wijmo:C1TreeNode runat="server"
CheckState="UnChecked" NodeIndex="0"
                    Text="C1TreeNode2">
                </wijmo:C1TreeNode>
                <wijmo:C1TreeNode runat="server"
CheckState="UnChecked" NodeIndex="0"
                    Text="C1TreeNode3">
                </wijmo:C1TreeNode>
                <wijmo:C1TreeNode runat="server"
CheckState="UnChecked" NodeIndex="0"
                    Text="C1TreeNode4">
                </wijmo:C1TreeNode>
            </Nodes>
        </wijmo:C1TreeNode>
        <wijmo:C1TreeNode runat="server" CheckState="UnChecked"
NodeIndex="0"
            Text="C1TreeNode1">
        </wijmo:C1TreeNode>
        <wijmo:C1TreeNode runat="server" CheckState="UnChecked"
NodeIndex="0"
            Text="C1TreeNode2">
        </wijmo:C1TreeNode>
        <wijmo:C1TreeNode runat="server" CheckState="UnChecked"
NodeIndex="0"
            Text="C1TreeNode3">
            <Nodes>
                <wijmo:C1TreeNode runat="server"
CheckState="UnChecked" NodeIndex="0"
                    Text="C1TreeNode1">
                </wijmo:C1TreeNode>
                <wijmo:C1TreeNode runat="server"
CheckState="UnChecked" NodeIndex="0"
                    Text="C1TreeNode2">
                </wijmo:C1TreeNode>
            </Nodes>
        </wijmo:C1TreeNode>

```

```

        Text="C1TreeViewNode2">
    </wijmo:C1TreeViewNode>
    <wijmo:C1TreeViewNode runat="server"
CheckState="UnChecked" NodeIndex="0"
        Text="C1TreeViewNode3">
    </wijmo:C1TreeViewNode>
    <wijmo:C1TreeViewNode runat="server"
CheckState="UnChecked" NodeIndex="0"
        Text="C1TreeViewNode4">
    </wijmo:C1TreeViewNode>
    <wijmo:C1TreeViewNode runat="server"
CheckState="UnChecked" NodeIndex="0"
        Text="C1TreeViewNode5">
    </wijmo:C1TreeViewNode>
    <wijmo:C1TreeViewNode runat="server"
CheckState="UnChecked" NodeIndex="0"
        Text="C1TreeViewNode6">
    </wijmo:C1TreeViewNode>
    <wijmo:C1TreeViewNode runat="server"
CheckState="UnChecked" NodeIndex="0"
        Text="C1TreeViewNode7">
    </wijmo:C1TreeViewNode>
    </Nodes>
</wijmo:C1TreeViewNode>
<wijmo:C1TreeViewNode runat="server" CheckState="UnChecked"
NodeIndex="0"
        Text="C1TreeViewNode4">
</wijmo:C1TreeViewNode>
<wijmo:C1TreeViewNode runat="server" CheckState="UnChecked"
NodeIndex="0"
        Text="C1TreeViewNode5">
</wijmo:C1TreeViewNode>
</Nodes>
</wijmo:C1TreeView>

```

2. Locate the first set of `<wijmo:C1TreeView>` tags and add `AllowDrag="True"` and `AllowDrop="True"`. The tag should resemble the following sample.

```

<wijmo:C1TreeView ID="C1TreeView1" runat="server" AllowDrag="True"
        AllowDrop="True">

```

3. Locate the second set of `<wijmo:C1TreeView>` tags and add `AllowDrop="True"`. The tag should resemble the following sample.

```
<wijmo:C1TreeView ID="C1TreeView2" runat="server" AllowDrop="True">
```

4. Press F5 to run your program. Note that you are now able to drag nodes from the first tree and drop them into the second tree.



# TreeView 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 **TreeView 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.

The following members pertain to the C1TreeView object:

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

The following members pertain to the C1TreeViewNode object:

- [http://wijmo.com/wiki/index.php/Tree#Options\\_2](http://wijmo.com/wiki/index.php/Tree#Options_2)

## 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.



