

---

ComponentOne

# Menu 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 Menu for ASP.NET Wijmo Overview.....	7
Installing Studio for ASP.NET Wijmo .....	7
Studio for ASP.NET Wijmo Setup Files.....	7
System Requirements .....	8
Uninstalling Studio for ASP.NET Wijmo.....	8
Deploying your Application in a Medium Trust Environment .....	9
End-User License Agreement .....	12
Licensing FAQs .....	12
What is Licensing?.....	12
How does Licensing Work?.....	13
Common Scenarios .....	13
Troubleshooting.....	15
Technical Support .....	17
Redistributable Files.....	18
About This Documentation .....	18
Namespaces.....	19
Creating an ASP.NET Project .....	20
Adding the Menu for ASP.NET Wijmo Components to a Project .....	21
ComponentOne Menu for ASP.NET WijmoMigration Guide.....	22
Prerequisites.....	22
Getting Started.....	22
Migration Details.....	22
Wijmo Top Tips.....	25
Key Features.....	26
Menu for ASP.NET Wijmo Quick Start.....	29
Step 1 of 3: Adding C1Menu to the Page.....	29
Step 2 of 3: Populating the Menu with a SiteMap .....	29
Step 3 of 3: Running the Project .....	30
Design-Time Support.....	31
C1Menu Smart Tag.....	31

C1Menu Designer Form .....	33
C1Menu Designer Form Context Menu .....	34
C1Menu Bindings Collection Editor.....	35
Menu Types.....	36
Top-level menu .....	36
Drop-down Menu .....	37
Group Menu .....	37
Flyout Menu .....	37
Sliding menu .....	38
Menu Creation.....	39
Static Menu Creation.....	40
Dynamic Menu Creation.....	41
Data Source Menu Creation.....	41
C1Menu Appearance.....	41
C1Menu Themes.....	41
Menu Item Icons.....	43
Templates.....	44
Menu Navigation and Shortcuts.....	45
C1Menu CSS Selectors .....	46
Menu for ASP.NET Wijmo Task-Based Help.....	47
Creating a C1Menu Control in Code.....	47
Working with Themes.....	47
Using a Built-In Theme .....	47
Using a Custom Theme.....	49
Working with Templates.....	51
Creating an Individual Item Template .....	51
Creating an ItemsTemplate .....	52
Creating a Child Items Template .....	53
Creating a Top-Level Item Template .....	55
Working with CSS Selectors .....	56
Customizing C1Menu Appearance with CSS Selectors.....	56
Customizing C1Menu Link Appearance with CSS Selectors .....	57
Adding a Top-Level Item to a Menu .....	58
Creating a Drop-Down Menu.....	59
Creating a Sliding Menu .....	60
Animating C1Menu .....	63

Changing Menu Item Triggers.....	64
C1Menu Item Functions.....	65
Dynamically Adding Items to C1Menu.....	71
Populating C1Menu with a Site Map .....	74
Populating C1Menu with XML.....	76
Saving and Loading a C1Menu from XML.....	78
Menu for ASP.NET Wijmo Client-Side Reference.....	81
Using the Wijmo CDN .....	81



# ComponentOne Menu for ASP.NET Wijmo Overview

Create multi-level menus with animation effects, image and check box items, interactive item scrolling, and more. You can even create a pop-up menu for context help within your application. **ComponentOne Menu™ for ASP.NET Wijmo** makes it easy with design-time support and client-side API.

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

Get started with the following topics:

- [Menu for ASP.NET Wijmo Quick Start](#) (page 29)
- [Menu Types](#) (page 36)
- [C1Menu Appearance](#) (page 41)
- [Task-Based Help](#) (page 47)

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

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

## System Requirements

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

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

## Uninstalling Studio for ASP.NET Wijmo

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

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

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

1. Open the **Control Panel** and select **Add or Remove Programs** (Programs and Features in Windows 7/Vista).
1. Select **ComponentOne Studio for ASP.NET Wijmo Help** and click the **Remove** button.
2. 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 10).

#### 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 10).
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 9) 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 9) 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 CILc.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 **C1Menu** using the fully qualified name for this class:

- Visual Basic

```
Dim MaskedInput As C1.Web.Wijmo.Controls.C1Menu
```

- C#

```
C1.Web.Wijmo.Controls.C1Menu;
```

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 C1Menu = C1.Web.UI.Controls.C1Menu
```

```
Imports MyMenu = MyProject.Objects.C1Menu
```

```
Dim wm1 As C1Menu
```

```
Dim wm2 As MyMenu
```

- C#

```
using C1Menu = C1.Web.UI.Controls.C1Menu;
```

```
using MyMenu = MyProject.Objects.C1Menu;
```

```
C1Menu wm1;
```

```
MyMenu 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 Menu 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 Menu for ASP.NET Wijmo components will appear in the Visual Studio Toolbox customization dialog box:

- C1Menu

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.C1Menu`. 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 19) for more information.

## ComponentOne Menu for ASP.NET WijmoMigration Guide

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

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

The **C1Menu** control is one of the new controls included in our ASP.net Wijmo line up. Let's take a look at the procedures involved in migrating over to our latest controls and delve into some details about some basic differences between ASP.NET Wijmo controls and their older ASP.NET AJAX counterparts along the way.

### Prerequisites

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

### Getting Started

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

Follow this quick-start to create the C1Menu control application:

[Menu for ASP.NET AJAX Quick Start](#)

### Migration Details

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

**For .NET 4.0:**

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


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

**For .NET 3.5:**

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

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

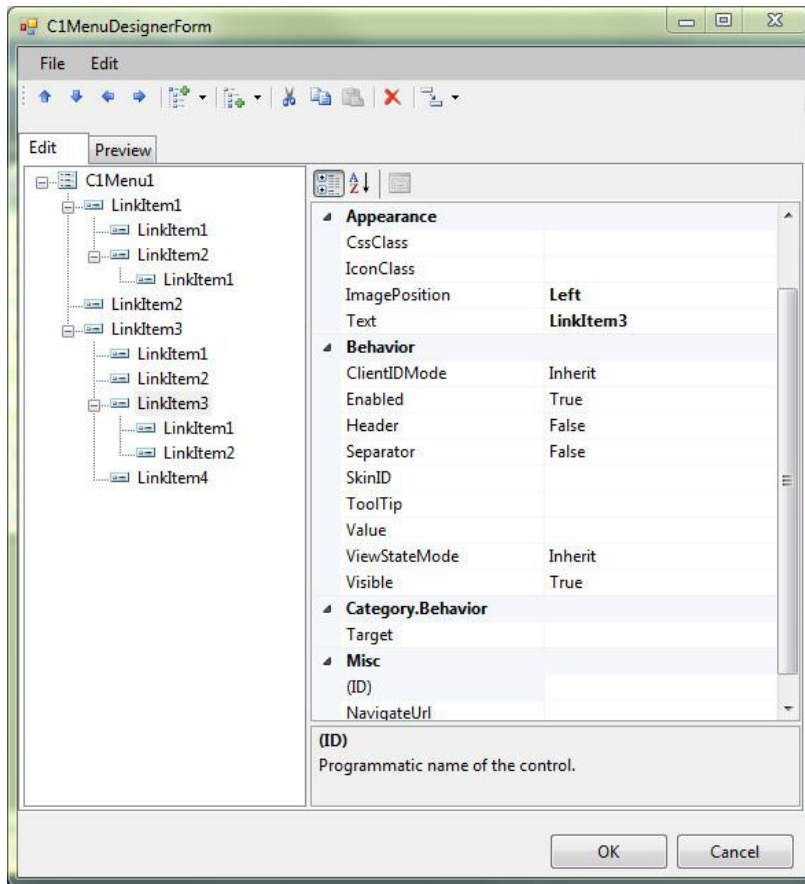
To add the new ASP.NET Wijmo control, simply drag the **C1Menu** from the Toolbox onto the page. This will display a **C1Menu** control without **C1MenuItems**.

You can use the **C1Menu Tasks** menu (to open this to this, simply click the control's smart tag ) to add items to the menu and edit each item's properties by clicking the Edit Menu link. All data binding is performed the same way as with the original ASP.NET **C1Menu**. However, if you created menu items in code and wanted to move those items from your old ASP.NET **C1Menu** to your new ASP.NET Wijmo **C1Menu** control, you could do so by manipulating the code. In Source view, you would copy the contents between the `<items>` tags that are within the `<ccl:c1menu>` tags and place them in your new `<Wijmo:C1Menu>` tags. You would then need to find and replace every instance of `ccl` to `wijmo`.

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

Palomino	Wijmo
<pre data-bbox="350 930 959 978">&lt;ccl:C1Menu ID="C1Menu1" runat="server" VisualStylePath=~\C1WebControls/VisualStyles"&gt;</pre>	<pre data-bbox="1138 930 1471 978">&lt;wijmo:C1Menu ID="C1Menu1" runat="server"&gt;</pre>
<p data-bbox="204 1062 883 1115">Palomino controls has multiple behavior properties that control animation. Example:</p> <pre data-bbox="350 1152 959 1272">&lt;ccl:C1Menu ID="C1Menu1" runat="server"ExpandAni mation="ScrollInFromTop" ExpandDelay="200" ExpandDuration="700" ExpandEas ing="EaseOutBack" VisualStylePath=~\C1WebControls/VisualStyles"&gt;</pre>	<p data-bbox="992 1062 1438 1115">Animations in Wijmo are controlled using block statements. Example:</p> <pre data-bbox="1138 1152 1471 1247">&lt;HideAnimation&gt; &lt;Animated Effect="fade"&gt; &lt;/Animated&gt; &lt;/HideAnimation&gt;</pre>

The **C1MenuDesignerForm** can be used to edit the ASP.NET Wijmo menu control.



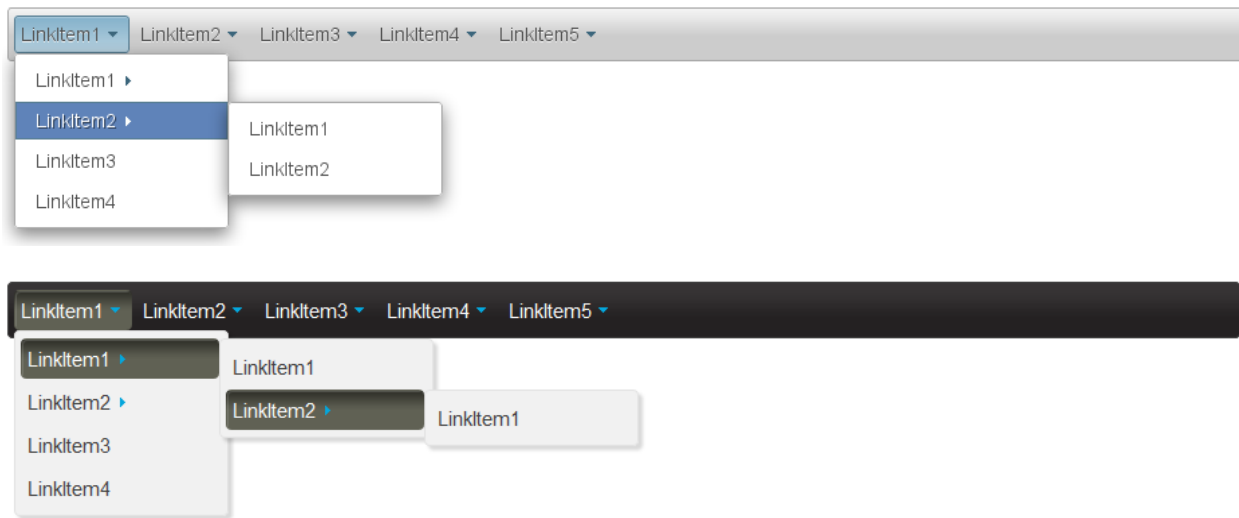
There are some differences between the server side properties for our previous ASP.NET AJAX Controls and our ASP.NET Wijmo controls. These changes are shown below. Some of these are actually simplified in the ASP.NET Wijmo version of the control, such as the **Animation** property.

Removed Properties		New Properties
AccessKey	EnableViewState	Animation
ClickToOpen	ExpandAnimation	BackLink
CollapseAnimation	ExpandDelay	BackLinkText
CollapseDelay	ExpandEasing	CrumbDefaultText
CollapseDuration	ExpandDuration	HideAnimation
CollapseEasing	FlowRight	HideDelay
ContextElementID	HoverElementID	MaxHeight
ContextMenu	NestedGroupCheckable	Mode
DisplayVisible	OnClientItemBlur	OnClientBlur
EmbeddedVisualStyles	OnClientItemChecked	OnClientSelect

Enabled	OnClientItemFocus	OnClientFoucs
EnableTheming	OnClientItemClick	Position
ScrollSettings	OnClientMouseDown	ShowAnimation
ToolTip	OnClientMouseOut	ShowDelay
Visible	OnClientMouseOver	SlidingAnimation
VisualStyle	OnClientItemUnChecked	TopLinkText
VisualStylePath	WindowCollisionDetection	Trigger
		TriggerEvent

You may want to note that the **Orientation** / **NestedGroupCheckable** properties for submenu items have been removed. The positions of sub menus are now set via the jQuery position utility.

You can play around with the **C1MenuDesigner** and adjust settings accordingly to your liking. When you are finished you will be left with a beautiful new **C1Menu** that has been built on web standards including AJAX, CSS, HTML5, and JQuery.



Keep in mind that you could easily change the appearance of your control by setting the theme to one of 30 pre-designed themes. These themes can be easily styled with ThemeRoller and of course you could create your own!

## 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 **CIMenu** control includes several unique features, including the following:

- **Vertical and Horizontal Menus**  
Menus and its submenus can be rendered either horizontally or vertically and include other layout options.
- **Scrolling**  
**CIMenu** can scroll top level menus, sub menus, and sub groups. You can set the scroll mode option to button click scrolling, button hover scrolling, edge hover scrolling, or scroll bars.
- **Menu Item Icons**  
Menu items can have their own icons. Choose from the many built-in icons or add your own icon to the menu.
- **Animation**  
Menu supports different expand and collapse animation effects. For example, fade in, scroll in from the top, open horizontally, bounce, and more.
- **Overlay Flash, ActiveX, and Windowed Objects**  
Unlike other menu controls, **CIMenu** can overlay any windowed objects, Flash, ActiveX and other standard and 3rd party components – so you're not limited in your website design.
- **Data Binding Support**  
Bind the **CIMenu** control to a data source – you can bind to an XML data source or SiteMap data source, or you can even read data from Access data source and create the **CIMenu** hierarchy dynamically.
- **Keyboard Support**  
Add access key support to give the **CIMenu** control focus with a chosen key combination. This enables end-users to use the keyboard arrow keys to navigate through the menu and menu items and the ENTER key to open a link in a menu item.
- **Theming**  
With just a click of the SmartTag, change the menu's look by selecting one of the 5 premium themes (Midnight, Aristo, Rocket, Cobalt, and Sterling). Optionally, use ThemeRoller from jQuery UI to create a customized theme!  
[http://stage.componentone.com/newimages/Products/ScreenShots/StudioASPNET/menu\\_themes.png](http://stage.componentone.com/newimages/Products/ScreenShots/StudioASPNET/menu_themes.png)
- **CSS Support**

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



# Menu for ASP.NET Wijmo Quick Start

The **C1Menu** Quick Start describes how to get started with the ASP.NET control, **C1Menu**. In the quick start you'll create an ASP.NET Web Site, add a **C1Menu** control to the page, apply different binding methods, add menu items and sub-menu items using the editor, change the menu's orientation, and more!

We have made it easy for you so you can jump to any topic you like since they don't follow chronological order.

## Step 1 of 3: Adding C1Menu to the Page

In this lesson you will learn how to create a new ASP.NET Web site and add a C1Menu control to your project. To begin the Quick Start, complete the following steps:

1. Begin by creating a new Web Site. Note that as you've created an AJAX-Enabled Web site, a **ScriptManager** control initially appears on the page.
2. While in Design view navigate to the Visual Studio Toolbox and double-click the **C1Menu** icon to add the **C1Menu** control to your page.

The page will appear similar to the following:



## Step 2 of 3: Populating the Menu with a SiteMap

In this section of the quick start, you'll learn how to bind C1Menu to a SiteMapDataSource.

To create an XML file and bind it to **C1Menu**, add the **XMLDataSource** component to the Web site, and then assign it to the **C1Menu** control.

1. Start a new **AJAX 1.0-Enabled ASP.NET 2.0 Web Site** project.
2. Right-click on the **App\_Data** in the Solution Explorer and select **Add New Item**. The **Add New Item** dialog box appears.
3. Select the XML File and rename it "Menu.xml".
4. Click on the **Add** button in the **Add New Item** dialog box.
5. Switch to the XML view and add the following data to **Menu.xml**:

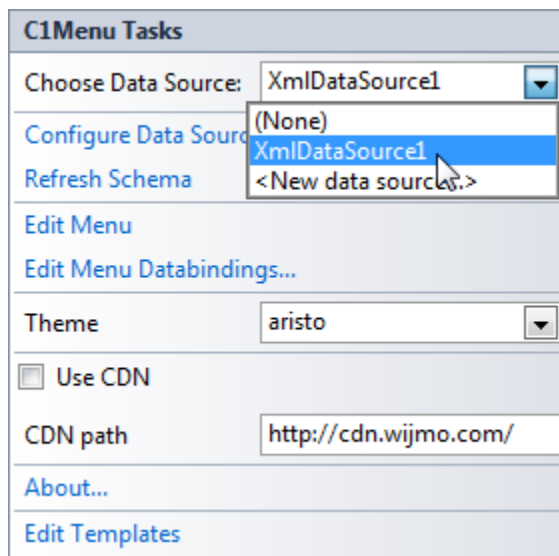
```
<?xml version="1.0" encoding="utf-8" ?>
<root>
  <menuitem Text="Home">
</menuitem>
  <menuitem Text="Products">
    <menuitem Text="Hardware">
</menuitem>
```

```

    <menuItem Text="Software">
    </menuItem>
</menuItem>
<menuItem Text="Services">
    <menuItem Text="Training">
    </menuItem>
    <menuItem Text="Consulting">
    </menuItem>
</root>

```

6. Switch back to the .aspx page and select the **Design** tab to switch to the design view.
7. Expand the **Data** node in the Visual Studio Toolbox and double-click on the **XmlDataSource** component to add it to the Web page.
8. Select the **XmlDataSource1** on the Web Page and navigate to the **Properties** window.
9. Click on the ellipsis button next to the **DataFile** property to open the **Select XML File** dialog box. Select **App\_Data** and click **Menu.xml** file. Click **OK** to add it to the **XmlDataSource1.DataFile** property.
10. Set the **XmlDataSource1.XPath** property to " root/menuitem".
11. Click the smart tag to open the **CIMenu Tasks** menu and select **XmlDataSource1** from the **Choose Data Source** drop-down listbox

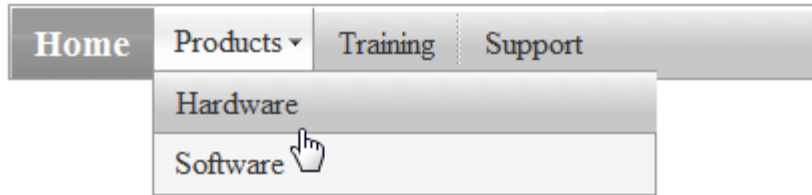


In the next step, you'll run the program and observe how the XML file populates the menu.

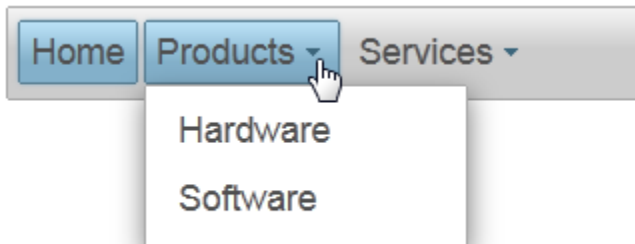
## Step 3 of 3: Running the Project

In this section of the quick start, you'll run the project and see the result of populating a **CIMenu** control with an XML file.

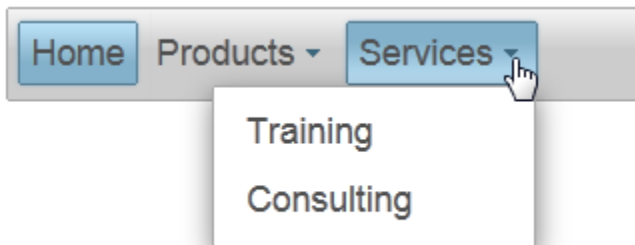
1. Save and build your project. Observe that there are three top-level menu items:



2. Hover your cursor over Products and observe that two items appear in its sub-menu:



3. Hover your cursor over Training and observe that two items appear in its sub-menu.



Congratulations! You have completed the Menu for ASP.NET Wijmo quick start.

## Design-Time Support

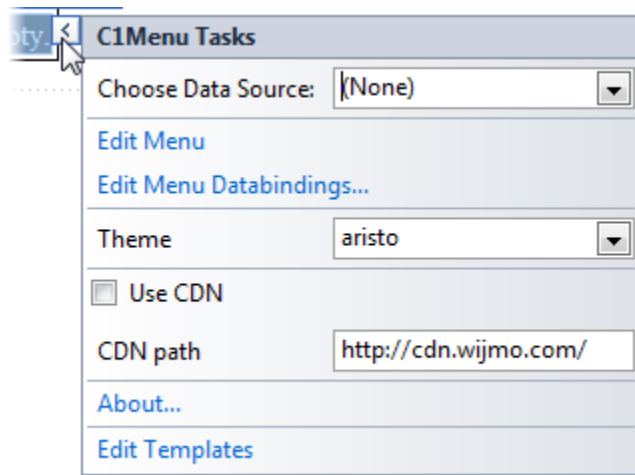
**C1Menu** provides smart tags, designer, and a bindings collection editor that offers rich design-time support and simplifies working with the object model.

The following topics describe how to use C1Menu's design-time environment to configure the C1Menu control.

### C1Menu Smart Tag

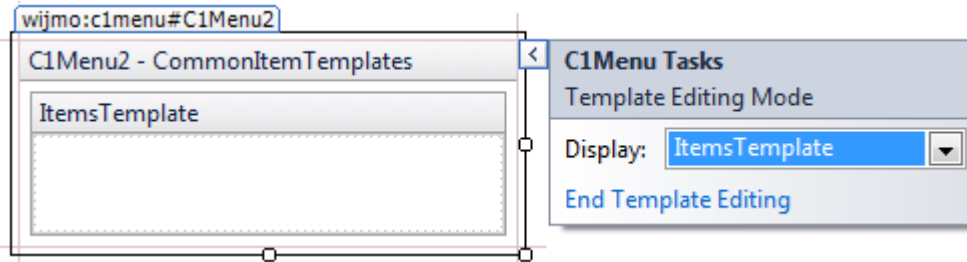
The **C1Menu** control includes a smart tag in Visual Studio. A smart tag represents a shortcut tasks menu that provides the most commonly used properties in **C1Menu**.

To access the **C1Menu Tasks** menu, click on the smart tag in the upper-right corner of the **C1Menu** control. This will open the **C1Menu Tasks** menu.



The **C1Menu Tasks** menu operates as follows:

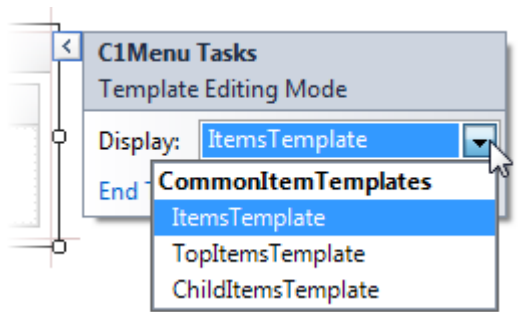
- **Choose Data Source**  
Clicking on the **Choose Data Source** item opens a drop-down list where you can choose an existing data source or select a new data source to bind to.
- **Edit Menu**  
Clicking on the **Edit Menu** item opens the **C1Menu Designer Form** where you can quickly configure **C1Menu**'s elements without having to scroll through its Properties window. You can load and save the control's content and can add LinkItem, Header, Group, and Separator elements. For more information on the **C1 Menu Designer Form**, see [C1Menu Designer Form](#) (page 33).
- **Edit Menu 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 skins.
- **About**  
Clicking on the **About** item displays the **About** dialog box, which is helpful in finding the version number of **Menu for ASP.NET** and online resources.
- **Edit Templates**  
Clicking on the **Edit Templates** item switches the **C1Menu** control to **Template Editing Mode**:



In Template Editing Mode, the **C1Menu 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.

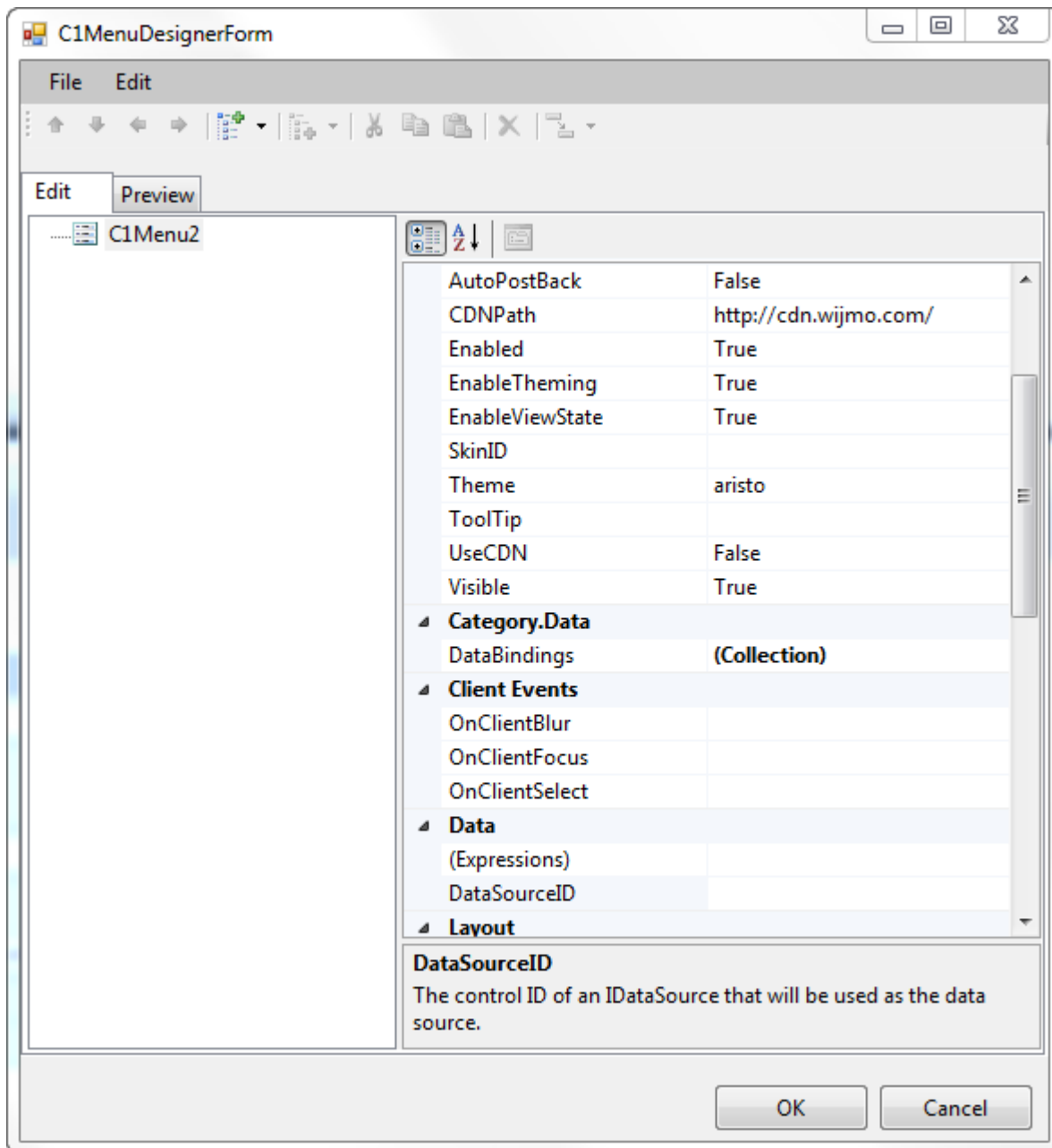
- **End Template Editing**

Clicking the **End Template Editing** item will end Template Editing Mode and return you to the main **C1Menu Tasks** menu.

## C1Menu Designer Form

The **C1Menu Designer Form** dialog box lets you quickly configure **C1Menu**'s elements without having to scroll through the control's Properties window. Using the **C1Menu Designer Form** you can add, manipulate, and delete **LinkItem**, **Header**, **Group**, and **Separator** elements in the **C1Menu** control and load and save the control's content.

To access the **C1Menu Designer Form** dialog box select the **Edit Menu** item from the **C1Menu Tasks** menu (see [C1Menu Smart Tag](#) (page 31) for details) or right-click on the **C1Menu** control at design time and select **Edit Menu**. The designer, with items added, looks similar to the following:



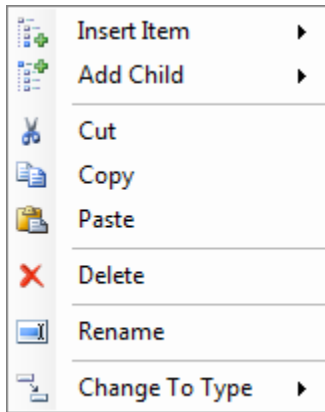
The **C1Menu Designer Form** includes an **Edit** tab and a **Preview** tab. The **Edit** tab, pictured above, consists of a left pane listing the order of added menu items. The right side of the **Edit** tab consists of a properties grid allowing you to quickly customize added items.

The **Preview** tab allows you to view the **C1Menu** control and quickly preview any changes you are making:





















### C1Menu Designer Form Context Menu

The **C1Menu Designer Form** context menu lets you quickly configure **C1Menu**'s elements. Using the **C1Menu Designer Form** you can load and save the control's content and can add and remove **LinkItem**, **Header**, **Group**, and **Separator** elements.

Access the **C1Menu Designer Form** context menu by right-clicking in the left pane of the **Edit** tab. The context menu will look similar to the following:



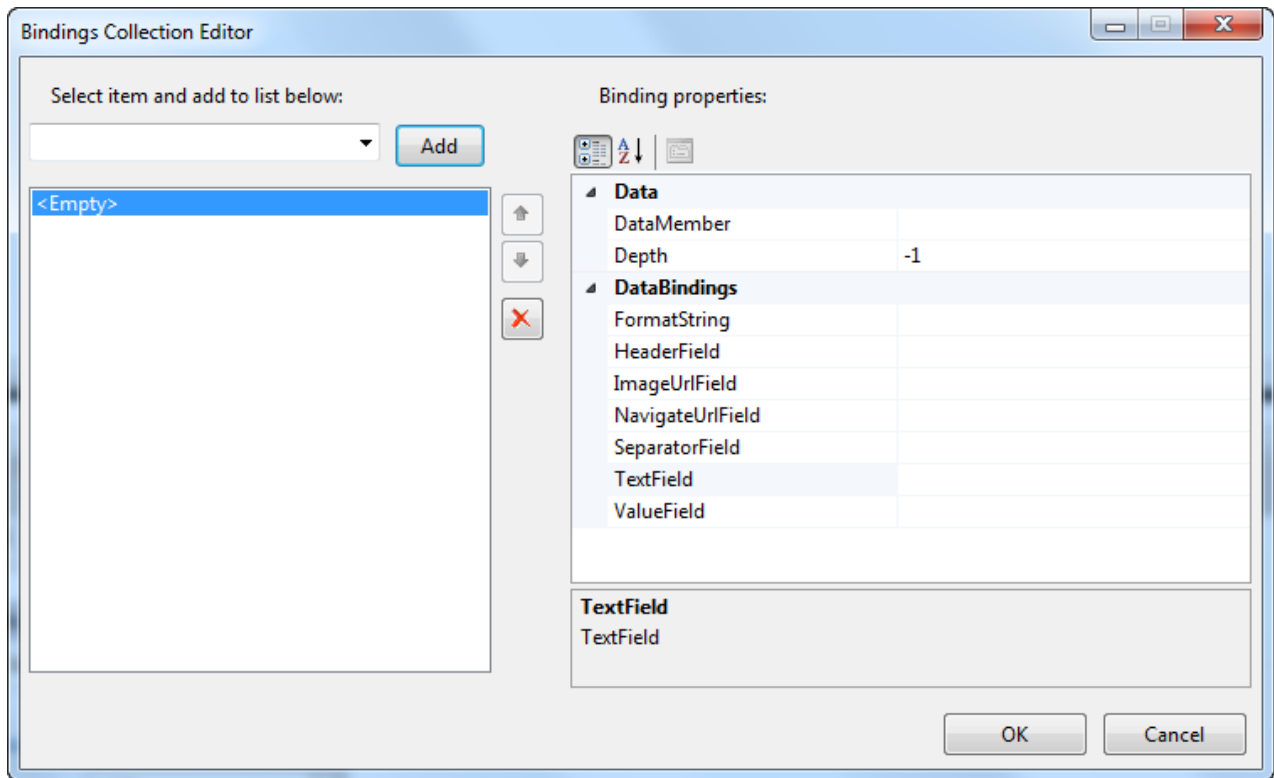
The following table describes the function of each item in the **C1Menu Designer Form** context menu.

Button	Name	Description
	Insert Item	Inserts a new item in the menu. Choices include the LinkItem (  ) , Header (  ) , Group (  ) , and Separator (  ) elements.
	Add Child	Inserts a child item under the current item. Choices include the LinkItem (  ) , Header (  ) , Group (  ) , and Separator (  ) elements.
	Cut	Cuts the currently selected item.
	Copy	Copies the currently selected item.
	Paste	Pastes a cut or copied item to the selected location.
	Delete	Deletes the currently selected item.
	Rename	Renames the currently selected item. Select this option and type in a new name for the selected item.
	Change to Type	Changes the current item to an item of a different type. Choices include the LinkItem (  ) , Header (  ) , Group (  ) , and Separator (  ) elements.

## C1Menu Bindings Collection Editor

The **Bindings Collection Editor** dialog box lets you easily define the relationship between a data item and the menu item it is bound to. You can access the **Bindings Collection Editor** dialog box by selecting the **ellipses** button next to the **C1Menu.DataBindings** property in the Properties window, or by selecting the **Edit Databindings** item from the **C1Menu Tasks** menu.

The **Bindings Collection Editor** dialog box will appear similar to the following:



The **Bindings Collection Editor** consists of a drop-down box where you can choose an existing item to add, a left-side list box listing added items, and a right-side properties grid where you can change the data and databinding properties for that item.

Note that any changes you make in the **Bindings Collection Editor** will be reflected in the `<DataBindings>` tag in the Source view, for example:

```
<cc1:clmenu id="C1Menu1" runat="server" datasourceid="SiteMapDataSource1"
visualstyle="Default" visualstylepath="~/C1WebControls/C1Menu/VisualStyles">
  <DataBindings>
    <cc1:C1MenuItemBinding DataMember="SiteMapNode" Depth="5"
NestedGroupHeight="" NestedGroupWidth="" />
  </DataBindings>
</cc1:clmenu>
```

## Menu Types

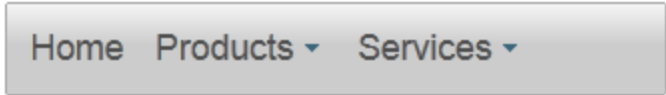
Typically top-level or submenus are used to create a navigation system, but the C1Menu control extends the menu functionality to create menus with groups so items can be organized into categories

This section details the common types of menus you can create using the C1Menu control.

### Top-level menu

The top level menu is the main menu. It consists of menu items which are arranged on a horizontal or vertical menu bar. Each menu item may or may not contain a list of submenu items. A top-level menu is always visible on the form. Typically top-level menus have 15 menu items or less.

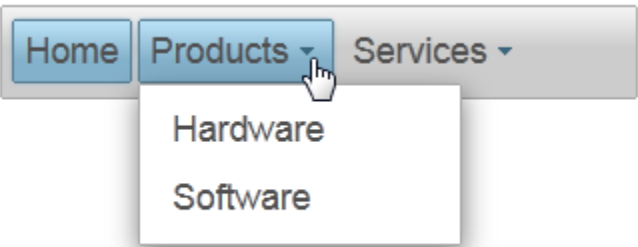
A typical top-level menu looks like the following:



## Drop-down Menu

The menu bar contains all of the drop-down menus and submenus in your application. Each menu in the menu bar is represented by its menu title.

Submenus, or drop-down menus, are additional menu items that appear within a menu item. Submenu items appear when you hover or click on the menu item on the top-level menu or submenu level. Depending on the style of the menu, an arrow or some other graphic is used to indicate when the menu item opens the submenu items. A submenu can have a submenu and so on.

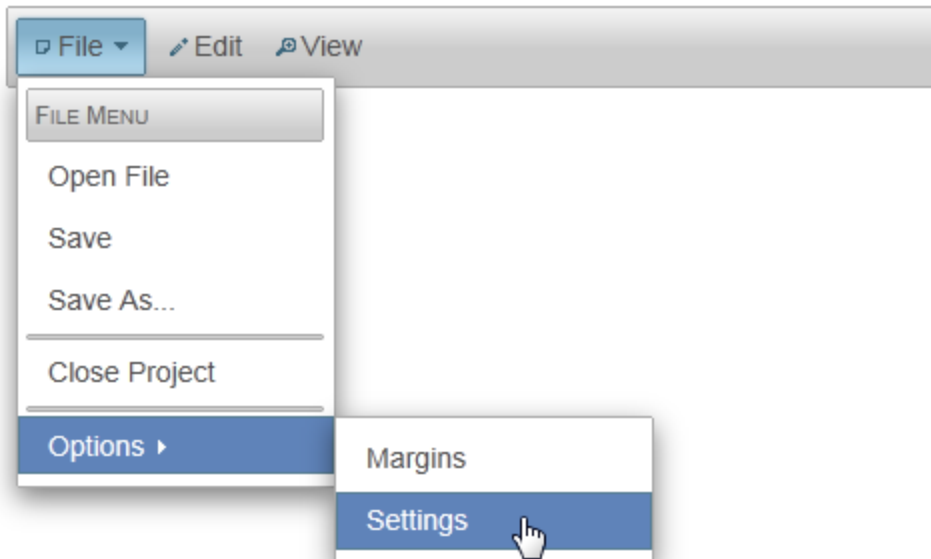


## Group Menu

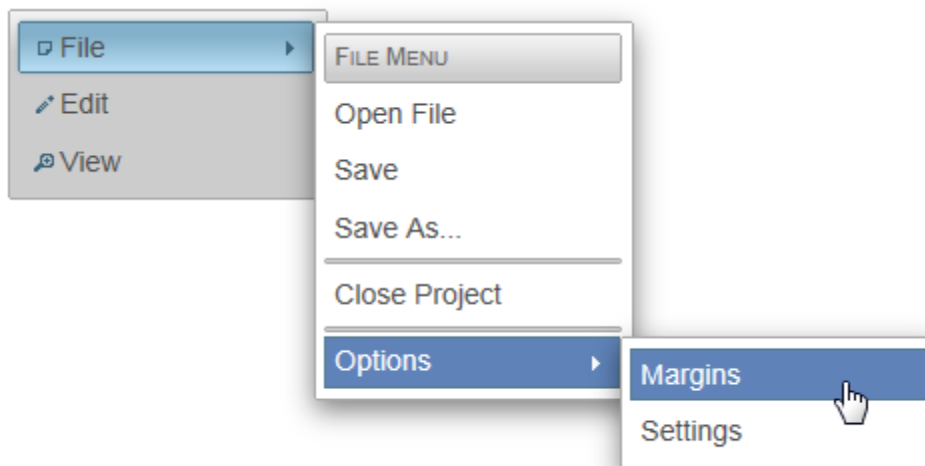
Grouped menus are multiple sets of menu items organized into one or more categories. The menu items in a group share a common function. Each group may include a heading item type that represents the category name for the group. Typically group menus appear as drop-down menus with at least one heading item to label the grouped menu items. Groupings can be separated by heading items or separator items.

## Flyout Menu

By default, the type of menu displayed is a flyout menu in a horizontal position. This is the basic menu that you see atop most applications.

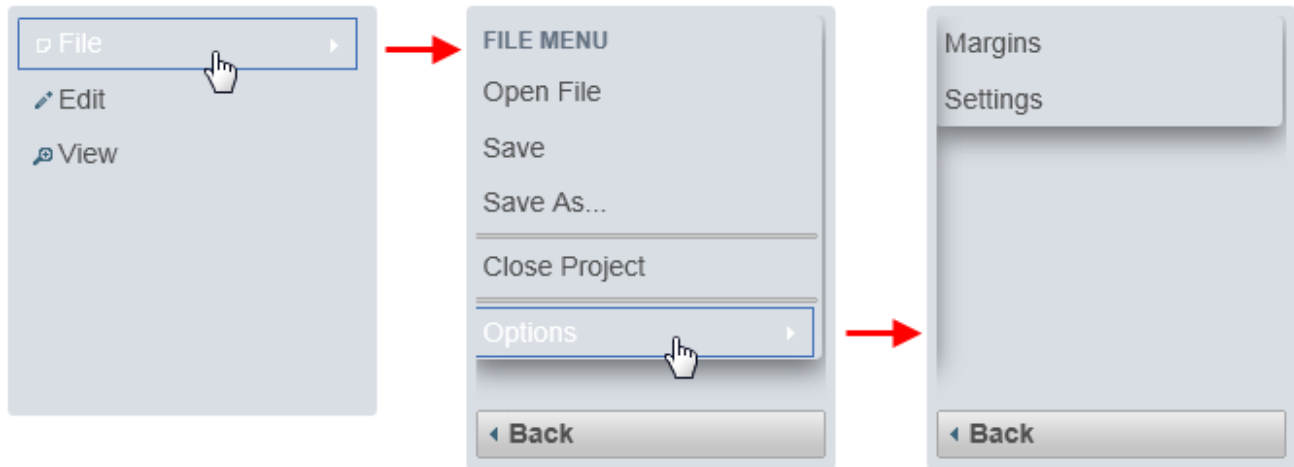


The flyout menu can also be created in a vertical position by setting the C1Menu control's Orientation property to Vertical.

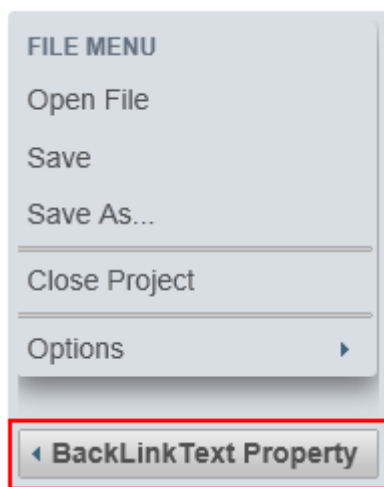


## Sliding menu

A sliding menu is a type of menu that is used primarily on smartphone devices. Instead of having a submenu fly out when the user clicks or hovers over a menu item, submenus will slide into view, replacing the previous menu.



Note that the above picture shows a "Back" button on the two submenus. A sliding menu, by default, will apply a "Back" navigation button to your submenus. If you'd like, you can change the string of the "Back" button by setting the C1Menu control's BackLinkText property.



You may also use breadcrumb navigation in lieu of the "Back" button by setting the C1Menu control's BackLink property to **False**.

## Menu Creation

A menu system can be created using one of the following methods:

- Static creation using declarative syntax
- Dynamic creation using a constructor to create new instances of the C1MenuItem class.
- Data source creation through binding C1Menu to a **SiteMapDataSource**, **XMLDataSource**, or an **AccessDataSource**.

## Static Menu Creation

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

You can use the **C1Menu Designer Form** designer to build the menu system or you can use declarative syntax in the .aspx file to specify the menu items. To display static menu items using declarative syntax, first nest opening and closing `<Item>` tags between opening and closing tags of the Menu control. Next, create the menu structure by nesting `<wijmo:C1MenuItem>` elements between opening and closing `<Items>` tags. Each `<wijmo:C1MenuItem>` element represents a menu item in the control and maps to a `C1MenuItem` object.

```
<wijmo:C1Menu ID="C1Menu1" runat="server">
  <HideAnimation>
  <Animated Effect="fade"></Animated>
  </HideAnimation>
    <Items>
      <wijmo:C1MenuItem runat="server" ImagePosition="Left"
Text="LinkItem1">
        <Items>
          <wijmo:C1MenuItem runat="server" ImagePosition="Left"
Text="LinkItem1">
            </wijmo:C1MenuItem>
          <wijmo:C1MenuItem runat="server" ImagePosition="Left"
Text="LinkItem2">
            </wijmo:C1MenuItem>
          </Items>
        </wijmo:C1MenuItem>
      <wijmo:C1MenuItem runat="server" ImagePosition="Left"
Text="LinkItem2">
        <Items>
          <wijmo:C1MenuItem runat="server" ImagePosition="Left"
Text="LinkItem1">
            </wijmo:C1MenuItem>
          <wijmo:C1MenuItem runat="server" ImagePosition="Left"
Text="LinkItem2">
            </wijmo:C1MenuItem>
          </Items>
        </wijmo:C1MenuItem>
      </Items>
    </wijmo:C1Menu>
```

## Dynamic Menu Creation

Dynamic menus can be created on the server side or client side. When creating dynamic menus on the server side, use a constructor to dynamically create a new instance of the `C1MenuItem` class. For client-side, the **CreateInstance** constructor can be used to dynamically create a new instance of the `C1Menu` control. For example the follow script creates a new menu control on the client side:

```
var aMenu = C1.Wijmo.Controls.C1Menu.createInstance ();  
document.body.appendChild(aMenu.element);
```

## Data Source Menu Creation

Menu items can be created from a hierarchal data source control such as an **XMLDataSource** or **SiteMapDataSource**. This allows you to update the menu items without having to edit code. Menu items can also be bound to a non-hierarchal control such as an **AccessDataSource** component.

See [Populating C1Menu with a Site Map](#) (page 74) and [Populating C1Menu with XML](#) (page 76) for tutorials about populating the `C1Menu` control with a data source.

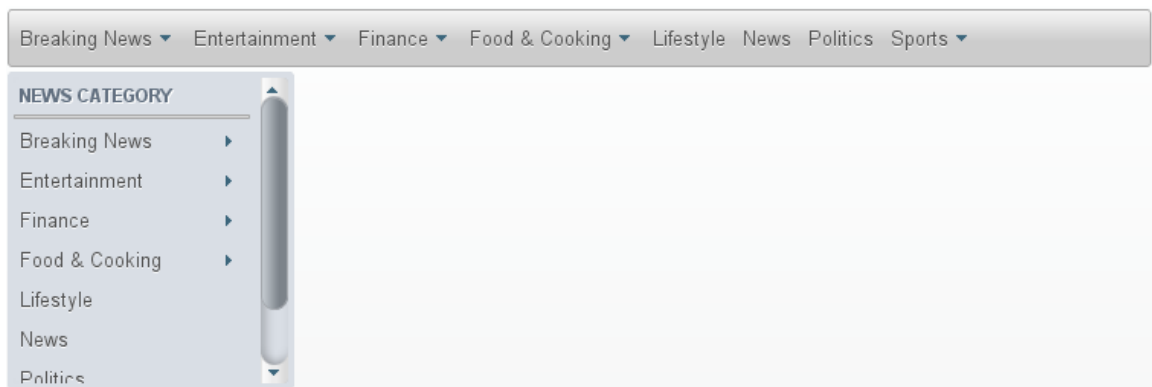
# C1Menu Appearance

The following topics illustrate features that will modify the appearance of the `C1Menu` control.

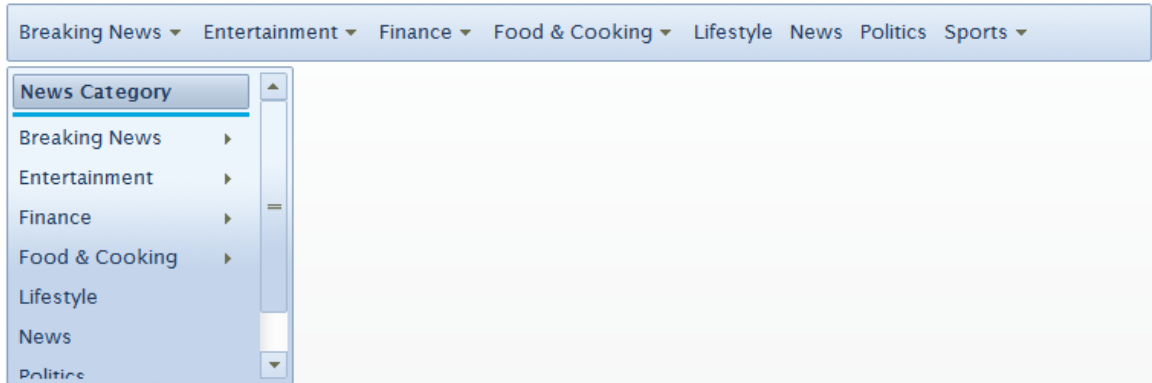
## C1Menu Themes

The `C1Menu` 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 `C1Menu` control as follows:

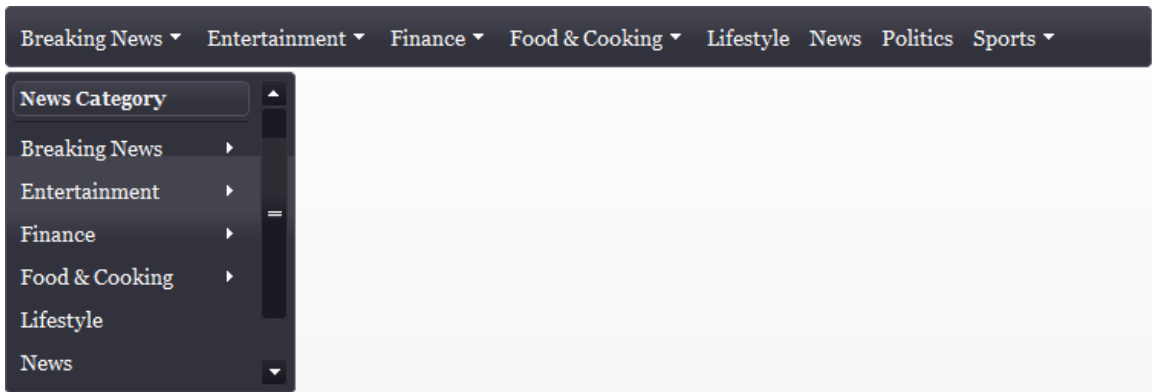
Aristo



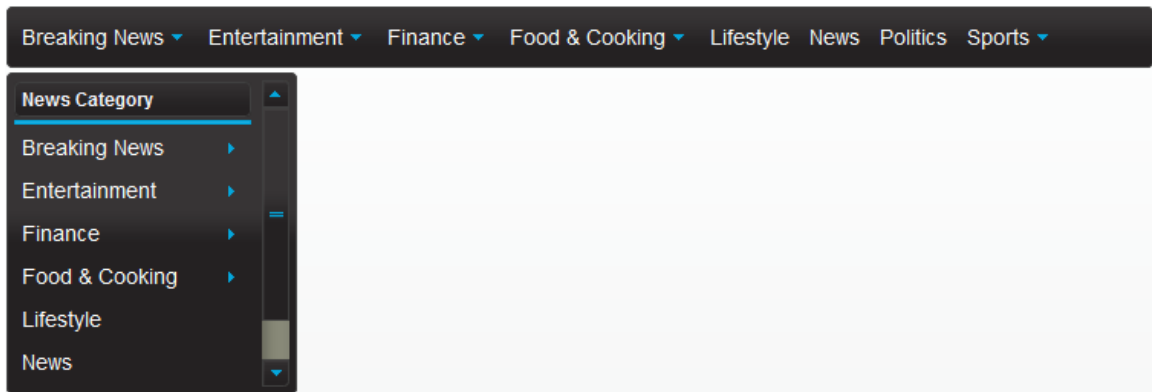
Cobalt



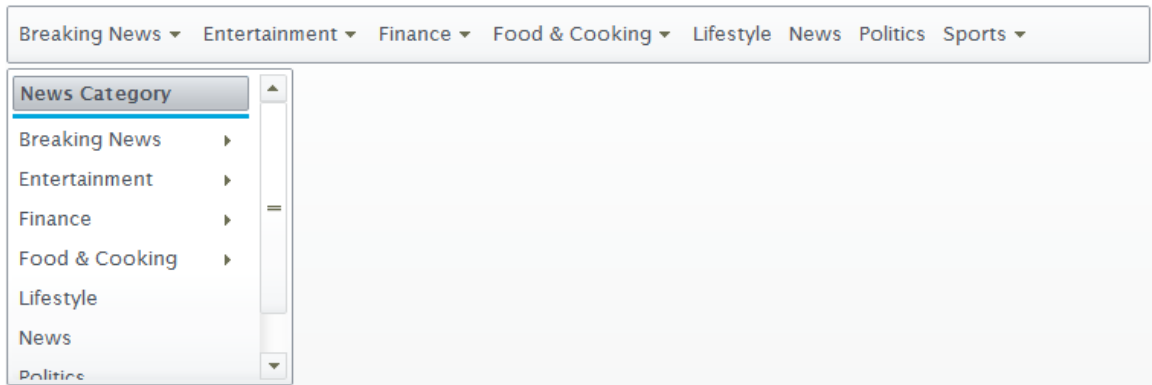
Midnight



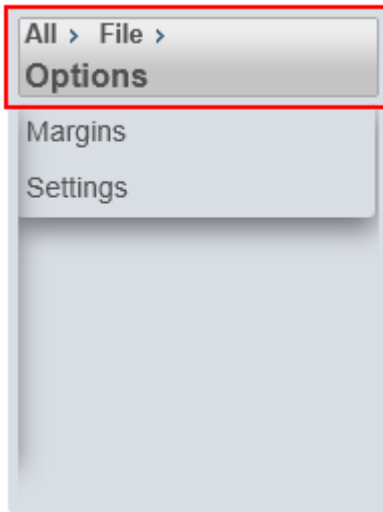
Rocket



Sterling



To set the theme of the C1Menu control, simply set its **Theme** property to one of the built-in themes.



## Menu Item Icons

You can easily add icons to individual menu items by setting that menu item's `IconClass` property. You can declare the jQuery UI "ui-icon" class and then add a second class to it describing the type of icon you'd want to use. The icon classes follow the following syntax:

```
ui-icon ui-icon-[icon name]
```

You can use any icon that's included in the jQuery UI set of framework icons. You can see a list of jQuery UI framework icons on the jQuery UI ThemeRoller page at <http://jqueryui.com/themeroller/>.

The default position for icons is to the left of a menu items text. If you'd like to change this, you can set the menu item's `ImagePosition` property to **Left**.

## Templates

C1Menu includes several types of templates, from the basic per-item template to global templates, such as the `ChildItemsTemplate`, which are used to control templates for entire portions of the control.

### Individual Templates

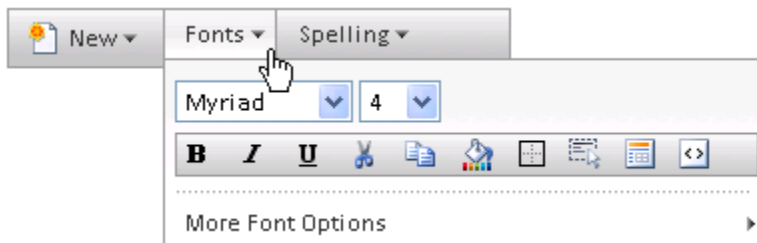
Templates can be used to embed HTML controls in a menu item, or they can be used for something as simple as formatting. These templates can be created in Source view by adding `<Template>` tags to individual menu items. For example:

```
<wijmo:C1MenuItem ID="C1MenuItem1" runat="server">
  <Template>
    <div>
      <a href="#" class="wijmo-wijmenu-text">New</a><span
class="wijmo-wijmenu-icon-right">Ctrl+N</span>
    </div>
  </Template>
</wijmo:C1MenuItem>
```

If a global template, such as the `ItemsTemplate`, is assigned to the **C1Menu** control, the individual template will override it.

For a tutorial concerning individual templates, see [Creating an Individual Item Template](#) (page 51).

### Global Templates



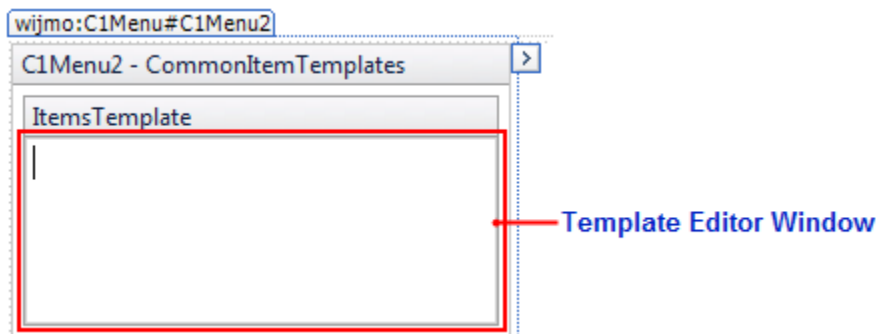
C1Menu also includes special template designers for customizing the top level menu items (the `TopItemsTemplate`), child menu items (the `ChildItemsTemplate`), or *all* menu items (the `ItemsTemplate`). Creating templates of different types allows you to control the design of all or even just a specific portion of your menu items. Templates are also useful as new item prototypes, which means that you can ensure dynamically added items adhere to design of other items in your menu.

The following **C1Menu** control has its `ChildItemsTemplate` customized to include an two HTML element, an `Input` (check box) control and a `Label` control, for each menu item:



### To access a template:

1. Click the C1Menu control's smart tag to open the **C1Menu Tasks** menu.  
**Template Editing Mode** engages.
2. Click the Display drop-down arrow and select the type of template (TopItemsTemplate, ChildItemsTemplate, or ItemsTemplate) from the drop-down list.
3. Add HTML elements or formatting to the template designer window. (See [Working with Templates](#) (page 51) for a tutorial.)



## Menu Navigation and Shortcuts

The two common approaches for navigating menu systems are:

- Point and click using the mouse
- Arrow keys using the keyboard

### Mouse Navigation

When the user clicks a menu item, the Menu control can either navigate to a linked Web page or simply post back to the server. If the `C1MenuItem.NavigateUrl` property of a menu item is set, the Menu control navigates to the linked page; otherwise, it posts the page back to the server for processing. By default, a linked page is displayed in the same window or frame as the Menu control. To display the linked content in a different window or frame, use the `C1MenuItem.Target` property of the Menu control.

### Keyboard Navigation

C1Menu supports access keys to enhance the user's navigation of your menu items. Access keys are used for navigating through the menu and typically use the ALT key plus the specified key.

You can add access keys to any of the menu items and submenu items using the **AccessKey** property. One character should be used when you assign access keys to menu items.

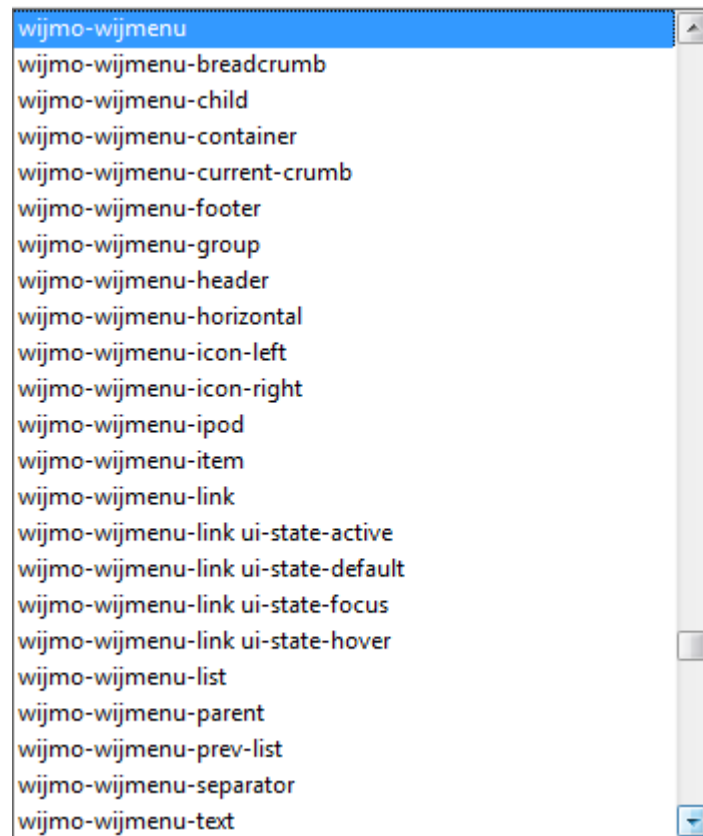
## C1Menu CSS Selectors

You can style many C1Menu 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 C1Menu control in your project and view the drop-down list next to the **CssClass** property in the Visual Studio Properties window.

C1Menu **CSS** selectors begin with wijmo-wijmenu:



# Menu 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 **C1Menu** control in general. Each topic provides a solution for specific tasks using the C1Menu control. Each task-based help topic also assumes that you have created a new ASP.NET project.

## Creating a C1Menu Control in Code

Creating a C1Menu control in code is an easy process. In this topic, you will add a **PlaceHolder** control to the page, add an import statement, add the C1Menu control, and add the control to the **PlaceHolder**.

Complete the following steps:

1. In Design View, navigate to the Visual Studio Toolbox and add a **PlaceHolder** control to the page.
2. Double-click the page to add a **Page\_Load** event and switch to Code view.
3. Add the following statement to the top of the Code Editor to import the appropriate namespace.

- Visual Basic

```
Imports C1.Web.Wijmo.Controls.C1Menu
```

- C#

```
using C1.Web.Wijmo.Controls.C1Menu;
```

4. Add the following code to create the C1Menu control and add the control to the Placeholder.

- Visual Basic

```
'Create a new C1Menu control  
Dim C1M As New C1Menu()  
  
'Add C1Menu to the Placeholder control  
Placeholder1.Controls.Add(C1M)
```

- C#

```
//Create a new C1Menu control  
C1Menu C1M = new C1Menu();  
  
//Add C1Menu to the Placeholder control  
Placeholder1.Controls.Add(C1M);
```

5. Press F5 to run your program. You have created a basic C1Menu control in code.

## Working with Themes


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

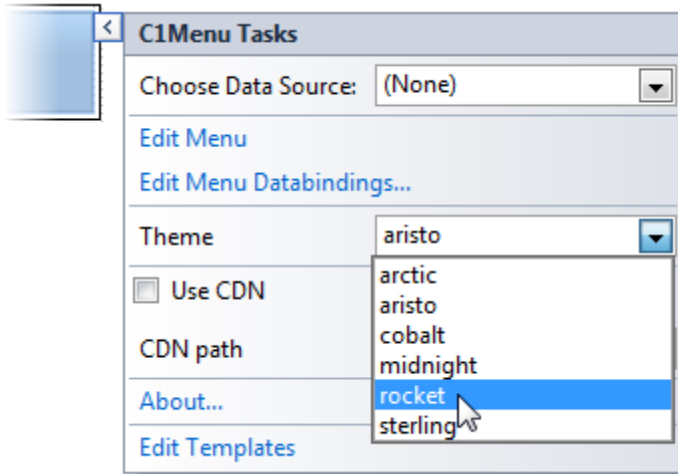
### Using a Built-In Theme

A C1Menu 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 [C1Menu Themes](#) (page 41).

## Changing the Theme in Design View

Complete the following steps:

1. Click the **C1Menu** smart tag  to open the **C1Menu 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 C1Menu control.

## Changing the Theme in Source View

To change the theme of your **C1Menu** in Source view, add `VisualStyle="rocket"` to the `<wijmo:C1Menu>` tag so that it resembles the following:

```
<wijmo:C1Menu ID="C1Menu1" runat="server" Theme="rocket"/>
```

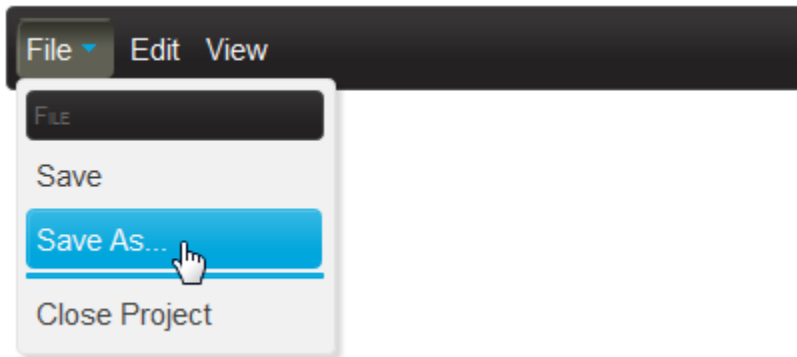
## Changing the Theme in Code

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  
`C1Menu1.Theme = "rocket"`
  - C#  
`C1Menu1.Theme = "rocket";`
3. Run the program.

✔ **This topic illustrates the following:**

The following image shows a **C1Menu** control with the **rocket** theme:



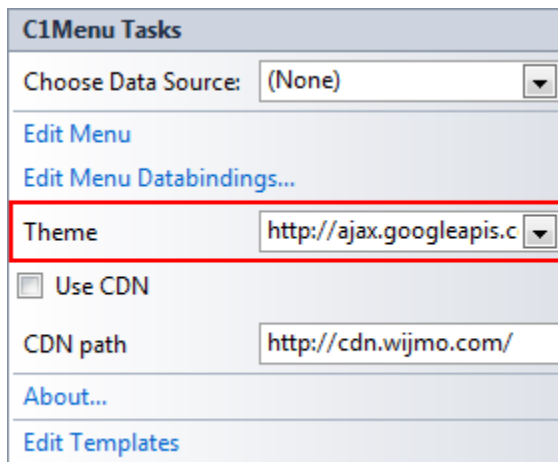
## Using a Custom Theme

**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 C1Menu in the following examples.

### Using a CDN URL

Complete the following steps:

1. Click the C1Menu smart tag to open the **C1Menu 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.

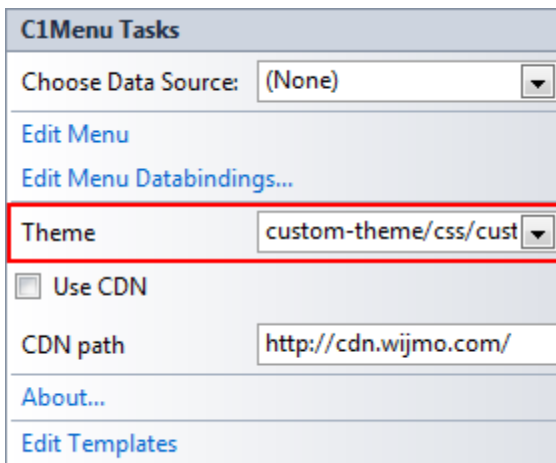
4. Run the project and notice the theme is applied to C1Menu.



## Using jQuery ThemeRoller

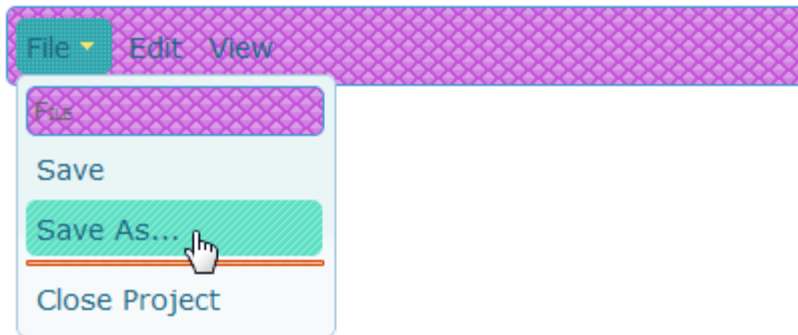
Complete the following steps:

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 C1Menu 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.

9. Run the project and notice the theme is applied to C1Menu.



## Working with Templates

This section contains tutorials that will teach you how to use the `ChildItemsTemplate`, the `ItemsTemplate`, and the `TopItemsTemplate`.

### Creating an Individual Item Template

In this tutorial, you will learn how to create a template for an individual menu item.

**Note:** Individual templates trump global templates, so adding a template to an individual item will prevent formats that you may have used in `ItemsTemplates`, `ChildItemsTemplates`, and `TopItemsTemplates`.

Complete the following steps:

1. Create a `C1Menu` control with one top-level menu item. Add two child items to that top-level menu item to create a submenu.
2. Switch to Source view and add the following markup between the `<wijmo:C1MenuItem ID="C1MenuItem1"...>` tags:

```
<Template>
  <div>
    <a href="#" class="wijmo-wijmenu-text">New</a><span
class="wijmo-wijmenu-icon-right">Ctrl+N</span>
  </div>
</Template>
```

3. Now add the following markup between the `<wijmo:C1MenuItem ID="C1MenuItem1"...>` tags:

```
<Template>
  <div>
    <a href="#" class="wijmo-wijmenu-text">Close</a><span
class="wijmo-wijmenu-icon-right">Ctrl+C</span>
  </div>
</Template>
```

- Press F5 to run the project. When the project loads, click the top-level menu item to open its submenu and observe that the two menu items have adopted the templates you specified.

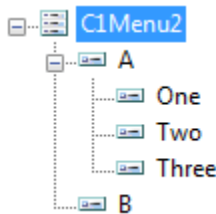



## Creating an ItemsTemplate

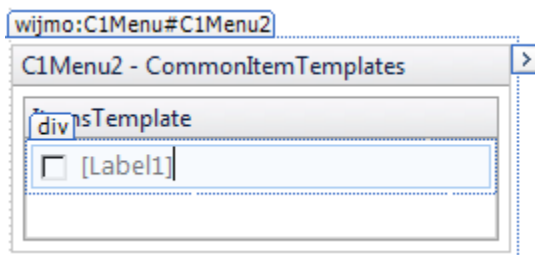
This tutorial will teach you how to create an ItemsTemplate template for the C1Menu control. The template, which will be the prototype for *all* items in the C1Menu control, will consist of a **Div** element, an **Input (checkbox)** control, and a **Label** control. The label control's **Text** property will be bound to the **Text** property of the individual menu items.


Complete the following steps:

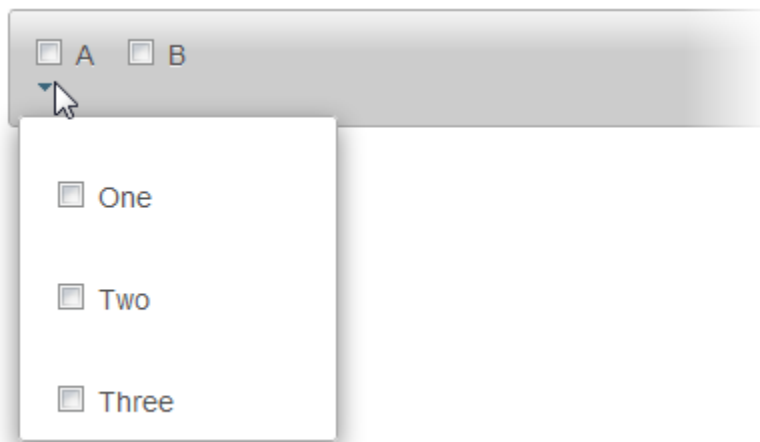
- Create a **C1Menu** control that has one two top-level menu items named "A" and "B". Give item "A" three child items and name them "One", "Two", and "Three".



- Click the C1Menu control's smart tag  and select **Edit Templates** from the **C1Menu Tasks** menu. The **C1Menu Tasks** menu changes to **Template Editing Mode**. The default mode listed in the **Description** drop-down list is ItemsTemplate, so there's no need to change it.
- Navigate to the Toolbox and, using a drag-and-drop operation, add a **Div** element to the **ChildItems** template.
- Navigate to the Toolbox and, using a drag-and-drop operation, add an **Input (Checkbox)** control and a **Label** control to the **Div** element. The template will look like this:




5. Select the **Label** element, navigate to the **Properties** window, and set the **Text** property to "<% #DataBinder.Eval(Container.DataItem, "Text") %>". This will bind the label's **Text** property to a data binding, which we will specify in a later step.
6. Click **End Template Editing** to close the template.
7. Create a data binding by completing the following steps:
  - a. Click the **C1Menu** control's smart tag  and select **Edit Menu Databindings...** from the **C1Menu Tasks** menu. Complete the following steps in the **Bindings Collection Editor** dialog box:
  - b. Click **Add** to add an <Empty> databinding to the project.
  - c. Set the <Empty> databinding's **TextField** property to **Text**.
  - d. Click **OK** to close the **Bindings Collection Editor** dialog box.
8. Run the project and select **A**. Observe that each menu item in the C1Menu control – top-level and child items - contains a check box and a label. Also observe that the label text is the same as the text you specified in step 1 of this tutorial because of the data bindings you specified in this tutorial.

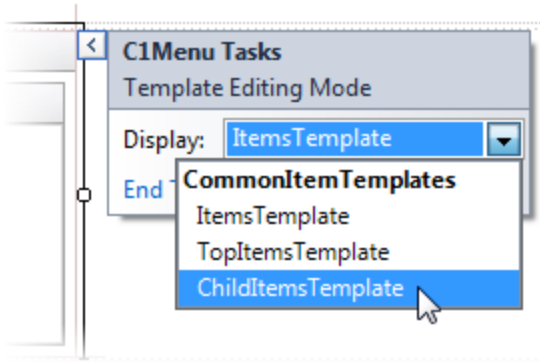


## Creating a Child Items Template

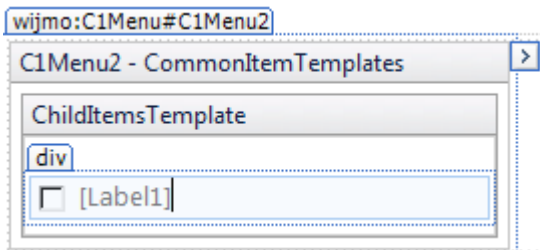
This tutorial will teach you how to create a ChildItemsTemplate template for the C1Menu control. The template, which will be the prototype for all items in the C1Menu control's various submenus, will consist of a **Div** element, an **Input (check box)** control, and a **Label** control. The label control's **Text** property will be bound to the **Text** property of the individual menu items.


Complete the following steps:

1. Create a **C1Menu** control that has one top-level menu item with three child items. Name the child items "One", "Two", and "Three".
2. Click the C1Menu control's smart tag  and select **Edit Templates** from the **C1Menu Tasks** menu. The **C1Menu Tasks** menu changes to **Template Editing Mode**.
3. Select **ChildTopItemsTemplate** from the **Display** drop-down list box.



4. Navigate to the Toolbox and, using a drag-and-drop operation, add a **Div** element to the **ChildItems** template.
5. Navigate to the Toolbox and, using a drag-and-drop operation, add an **Input (Checkbox)** control and a **Label** control to the **Div** element. The template will look like this:



6. Select the Label element, navigate to the Properties window, and set the Text property to "<% #DataBinder.Eval(Container.DataItem,"Text") %>". This will bind the label's Text property to a data binding, which we will specify in a later step.
7. Click End Template Editing to close the template.
8. Create a data binding by completing the following steps:
  - a. Click the **C1Menu** control's smart tag  and select **Edit Menu Databindings...** from the **C1Menu Tasks** menu. Complete the following steps in the **Bindings Collection Editor** dialog box:
    - b. Click Add to add an <Empty> databinding to the project.
    - c. Set the <Empty> databinding's **TextField** property to **Text**.
    - d. Click **OK** to close the **Bindings Collection Editor** dialog box.
9. Run the project and select LinkItem1. Observe that each menu item in the sub menu contains a check box and a label. Also observe that the label text is the same as the text you specified in step 1 of this tutorial; that is because of the data bindings.

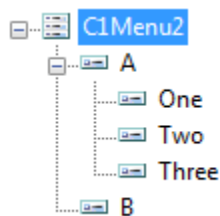



## Creating a Top-Level Item Template

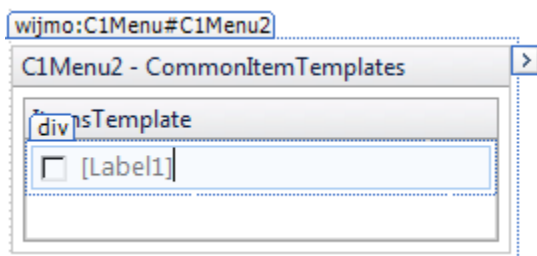
This tutorial will teach you how to create an `TopItemsTemplate` template for the `C1Menu` control. The template, which will be the prototype for *all* items in the `C1Menu` control, will consist of a **Div** element, an **Input (checkbox)** control, and a **Label** control. The label control's **Text** property will be bound to the **Text** property of the individual menu items.


Complete the following steps:

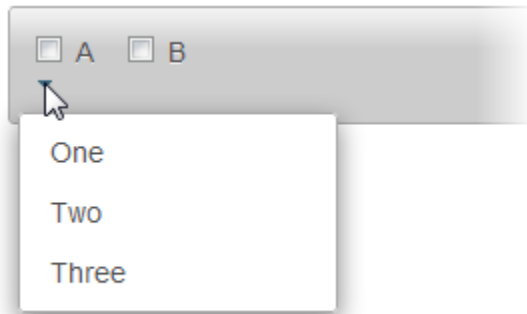
1. Create a **C1Menu** control that has one two top-level menu items named "A" and "B". Give item "A" three child items, and name them "One", "Two", and "Three".



2. Click the `C1Menu` control's smart tag  and select **Edit Templates** from the **C1Menu Tasks** menu. The **C1Menu Tasks** menu changes to **Template Editing Mode**. The default mode listed in the Description drop-down list is `TopItemsTemplate`, so there's no need to change it.
3. Navigate to the Toolbox and, using a drag-and-drop operation, add a **Div** element to the **ChildItems** template.
4. Navigate to the Toolbox and, using a drag-and-drop operation, add an **Input (Checkbox)** control and a **Label** control to the **Div** element. The template will look like this:



5. Select the **Label** element, navigate to the **Properties** window, and set the **Text** property to "<% #DataBinder.Eval(Container.DataItem, "Text") %>". This will bind the label's **Text** property to a data binding, which we will specify in a later step.
6. Click **End Template Editing** to close the template.
7. Create a data binding by completing the following steps:
  - a. Click the **CIMenu** control's smart tag  and select **Edit Menu Databindings...** from the **CIMenu Tasks** menu. Complete the following steps in the **Bindings Collection Editor** dialog box:
    - b. Click **Add** to add an <Empty> databinding to the project.
    - c. Set the <Empty> databinding's **TextField** property to **Text**.
    - d. Click **OK** to close the **Bindings Collection Editor** dialog box.
8. Run the project and select **A**. Observe that only the top-level menu items contain a check box and a label. Also observe that the label text on the top-level menu items is the same as the text you specified in step 1 of this tutorial because of the databindings you specified in this tutorial.



## Working with CSS Selectors

C1Menu allows full customization through supporting CSS. This section will teach you how to use the built-in CSS selectors to customize the C1Menu control. You can use single selectors or you can combine selectors to make the CSS more specific.

### Customizing C1Menu Appearance with CSS Selectors

You can use CSS selectors to customize the appearance of the **C1Menu** control in the Source View.

1. In the Source View, locate the first set of `<asp:Content>` tags. Add the following tags between the `<asp:Content>` `</asp:Content>` tags that will allow you to insert CSS styling.
 

```
<style type="text/css"> </style>
```
2. Use the following CSS selector to set the general appearance of the control.
 

```
.wijmo-wijmenu
```
3. Insert `{background: #339900; border-color: Blue; }` after the CSS selector. This will set the background to green with a blue border.
4. Insert `.wijmo-wijmenu-text {color: #992233; }` to set the color of the text for the control.

5. Insert `.wijmo-mijmenu-item {font-family: French Script MT; }` and `.wijmo-wijmenu-parent {font-family: Blackadder ITC; }` to set the fonts for menu items and parent menu items.
6. Press F5 to run your program. Note the changes that you have made to the appearance of the **C1Menu** control. It should appear as in the following image. Note that parent menu items and regular menu items appear in different font styles.



### Customizing C1Menu Link Appearance with CSS Selectors

**C1Menu** allows you to customize link appearance using CSS selectors. For this topic, you will learn how to use the CSS selectors to customize links on mouse hover.

1. In the Source View, locate the first set of `<asp:Content>` tags. Add the following tags between the `<asp:Content>` `</asp:Content>` tags that will allow you to insert CSS styling.
 

```
<style type="text/css"> </style>
```
2. Use the following CSS selector to set the general appearance of the control.
 

```
.wijmo-wijmenu a.wijmo-wijmenu-link:hover
{
font-family:Calibri; background: Pink; border-color: green; border-
style: solid;
}
```
3. Press F5 to run your program. Hover over one of the menu items; it should appear as in the following image.



## Adding a Top-Level Item to a Menu

This topic illustrates how to add a top-level menu item to a C1Menu 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 **C1Menu Tasks** menu. Select **Edit Menu**.

The **C1Menu Designer Form** dialog box opens.

2. Click the **Add Child Item** button  to add a C1MenuItem to the C1Menu control.
3. Click **OK** to close the **C1Menu Design Form** dialog box.

### In Source View

Add the following markup between the <wijmo:C1Menu> tags:

```
<wijmo:C1MenuItem ID="MenuItem1" runat="server" Text="LinkItem1">
</wijmo:C1MenuItem>
```

### In Code View

1. Complete the following steps: Import the following namespace into your project:

- Visual Basic
 

```
Imports C1.Web.Wijmo.Controls.C1Menu
```

- C#
 

```
using C1.Web.Wijmo.Controls.C1Menu;
```

2. Add the following code to the Page\_Load event:

- Visual Basic

```
Dim MenuItem1 As New C1MenuItem()  
MenuItem1.Text = "LinkItem1"  
C1Menu1.Items.Add(MenuItem1)
```

- C#

```
C1MenuItem MenuItem1 = new C1MenuItem();  
MenuItem1.Text = "LinkItem1";  
C1Menu1.Items.Add(MenuItem1);
```

3. Run the program.

## Creating a Drop-Down Menu

This topic illustrates the creation of a drop-down menu using Design view, Source view, and code. To add a drop-down menu, all you have to do is add one (or more) C1MenuItem as a child of a top-level C1MenuItem. This topic assumes that you have completed [Adding a Top-Level Item to a Menu](#) (page 58).

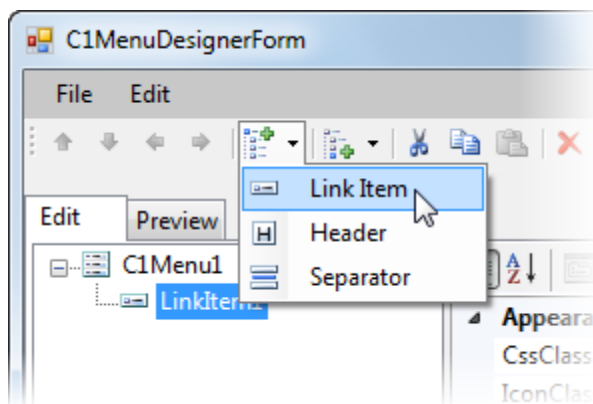
### In Design View

Complete the following steps:

1. Click the smart tag to open the **C1Menu Tasks** menu. Select **Edit Menu**.

The **C1Menu Designer Form** dialog box opens.

2. Select the menu item you wish to add the submenu to.
3. Click the **Add Child Item** drop-down arrow and select **Link Item** from the list to add a child item to the selected menu item..



4. Click **OK** to close the **C1Menu Design Form** dialog box.

### In Source View

Add the following markup between the <wijmo:C1MenuItem> tags of the item you wish to add the submenu to:

```
<Nodes>  
  <wijmo:C1MenuItem ID="Node1" runat="server" Text="LinkItem1">  
    </wijmo:C1MenuItem>  
</Nodes>
```

### In Code View

Complete the following steps:

1. Import the following namespace into your project:

- Visual Basic

```
Imports C1.Web.Wijmo.Controls.C1Menu
```

- C#

```
using C1.Web.Wijmo.Controls.C1Menu;
```

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

- Visual Basic

```
' Create first node and add it to the C1Menu.
```

```
Dim C1MenuItem1 As New C1MenuItem()
```

```
C1MenuItem1.Text = "LinkItem1"
```

```
C1Menu1.Items.Add(C1MenuItem1)
```

```
' Create the child node and add it to C1MenuItem1
```

```
Dim C1MenuItem2 As New C1MenuItem()
```

```
C1MenuItem1.Text = "LinkItem1"
```

```
C1MenuItem1.Items.Add(C1MenuItem2)
```

- C#

```
// Create first node and add it to the C1Menu.
```

```
C1MenuItem C1MenuItem1 = new C1MenuItem();
```

```
C1MenuItem1.Text = "LinkItem1";
```

```
C1Menu1.Items.Add(C1MenuItem1);
```

```
// Create the child node and add it to C1MenuItem1
```

```
C1MenuItem C1MenuItem2 = new C1MenuItem();
```

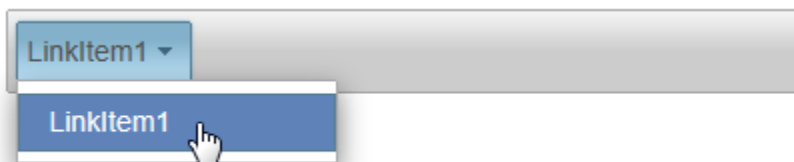
```
C1MenuItem2.Text = "LinkItem1";
```

```
C1MenuItem1.Items.Add(C1MenuItem2);
```

3. Run the program.

✔ **This topic illustrates the following:**

Creating a submenu is as easy as adding a child `C1MenuItem` to a parent `C1MenuItem`.



## Creating a Sliding Menu

**Menu for ASP.NET Wijmo** can be displayed as a traditional flyout menu or as a touch phone-like sliding menu. In this tutorial, you'll create a sliding menu with a breadcrumb header and an elastic sliding animation.

Complete the following steps:

1. In Source view, add the following markup between the `<wijmo:C1Menu>` and `</ wijmo:C1Menu>` tags to add menu items and submenus to the `C1Menu` control:

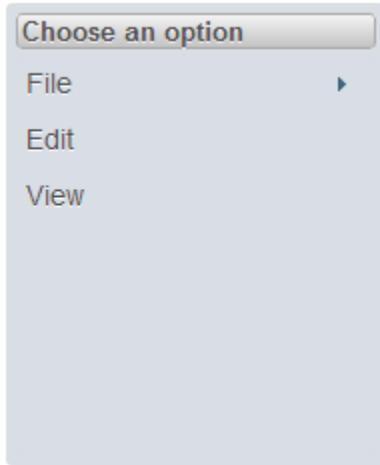
```

<Items>
    <wijmo:C1MenuItem ID="C1MenuItem1" runat="server"
ImagePosition="Left" Text="File">
        <Items>
<wijmo:C1MenuItem runat="server" Header="True" Text="File Menu"
StaticKey="sk1"
                ImagePosition="Left"
ID="C1MenuItem2"></wijmo:C1MenuItem>
                <wijmo:C1MenuItem runat="server" ImagePosition="Left"
Text="Open File">
                        </wijmo:C1MenuItem>
                <wijmo:C1MenuItem ID="C1MenuItem3" runat="server"
ImagePosition="Left" Text="Save">
                        </wijmo:C1MenuItem>
                <wijmo:C1MenuItem ID="C1MenuItem4" runat="server"
ImagePosition="Left" Text="Save As...">
                        </wijmo:C1MenuItem>
                <wijmo:C1MenuItem runat="server" ImagePosition="Left"
Separator="True">
                        </wijmo:C1MenuItem>
                <wijmo:C1MenuItem ID="C1MenuItem6" runat="server"
ImagePosition="Left" Text="Close Project">
                        </wijmo:C1MenuItem>
                <wijmo:C1MenuItem runat="server" ImagePosition="Left"
Separator="True">
                        </wijmo:C1MenuItem>
                <wijmo:C1MenuItem runat="server" ImagePosition="Left"
Text="Options">
                        <Items>
                                <wijmo:C1MenuItem runat="server"
ImagePosition="Left" Text="Margins">
                                        </wijmo:C1MenuItem>
                                <wijmo:C1MenuItem runat="server"
ImagePosition="Left" Text="Settings">
                                        </wijmo:C1MenuItem>
                                </Items>
                        </wijmo:C1MenuItem>
                </Items>
        </wijmo:C1MenuItem>
        <wijmo:C1MenuItem ID="C1MenuItem7" runat="server"
ImagePosition="Left" Text="Edit">
                </wijmo:C1MenuItem>
        <wijmo:C1MenuItem ID="C1MenuItem8" runat="server"
ImagePosition="Left" Text="View">
                </wijmo:C1MenuItem>
        </Items>

```

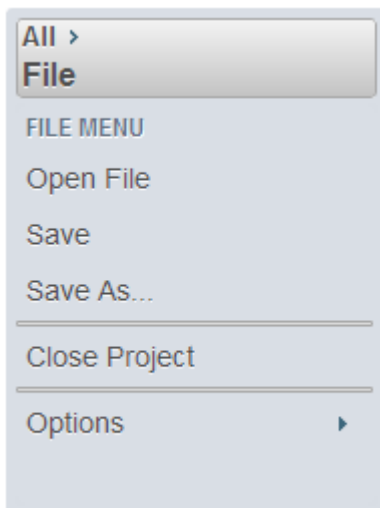
2. Switch to Design view.
3. Click the C1Menu control's smart tag to open the **C1Menu Tasks** menu. Click **Edit Menu**. The **C1MenuDesignerForm** dialog box opens.
4. With **C1Menu1** selected in the treeview, set the following properties:
  - Set the Mode property to **Sliding**. This is what changes C1Menu from a traditional flyout menu to a sliding phone-like sliding menu.
  - Set the BackLink property to **False**. This replaces the default "Back" link with breadcrumb links.

- Expand the SlidingAnimation node and set the **Easing** property to **EaseInOutElastic**. This sets the animation for the sliding menu.
5. Click **OK** to exit the **C1MenuDesignerForm** dialog box.
  6. Press **F5** to run your project. The project will look as follows:

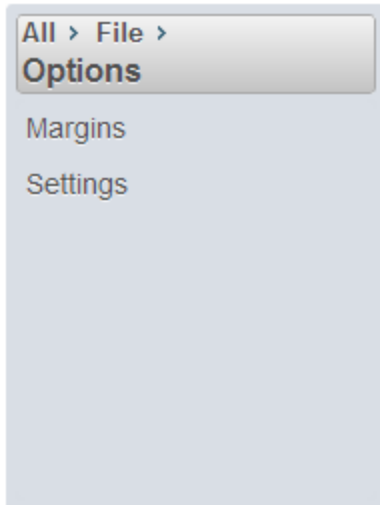


The "Choose an option" text at the top of the menu is a placeholder. It's where the breadcrumb links will appear as you navigate through the menu.

7. Click **File**. Did you notice the elastic animation as the new menu screen slid into view? Also observe that the breadcrumb bar says "All > File".



8. Click **Options** and observe that the breadcrumbs say "All > File > Options". To return to the File menu, click **File**. To return to the top-level menu, click **All**.

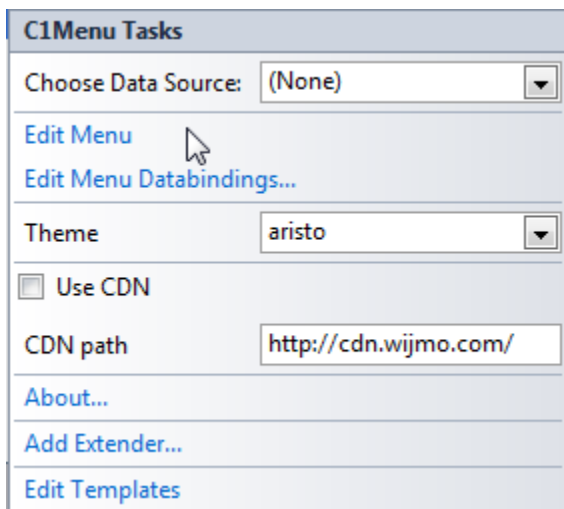


## Animating C1Menu

The **C1Menu** control supports animation effects. This topic illustrates changing the animation effects using either the **C1Menu Tasks** menu or the Source View.

### In Design View

1. Click the **C1Menu** smart tag to open the **C1Menu Tasks** menu.
2. Click the **Edit Menu** link to open the **C1Menu DesignerForm**.



3. Find the **Hide Animation** and **Show Animation** properties in the list. Use the arrows to expand the property nodes.

HideAnimation	C1.Web.Wijmo.Controls.An
▶ Animated	Effect: blind
Duration	400
Easing	Swing
Option	(Collection)
HideDelay	400
ShowAnimation	C1.Web.Wijmo.Controls.An
▶ Animated	Effect: bounce
Duration	400
Easing	Swing
Option	(Collection)
ShowDelay	400
SkinID	

4. Set the **HideAnimation > Animation > Effect** property to **blind** and the **ShowAnimation > Animation > Effect** property to **bounce**.
5. Click **OK** and then press F5 to run your program. Note the animation effects as you open and close submenus.

#### In Source View

1. In the Source View of your project, add the following markup between the `<wijmo:C1Menu>` tags.

```

<ShowAnimation>
    <Animated Effect="bounce" />
</ShowAnimation>
<HideAnimation>
    <Animated Effect="blind"></Animated>
</HideAnimation>

```

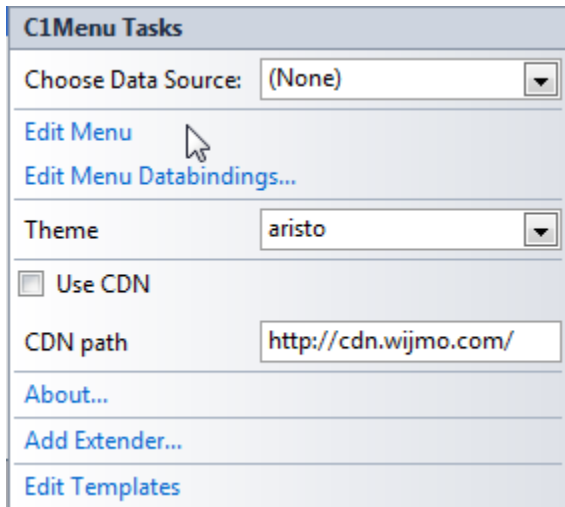
2. Press F5 to run your program. Hover your mouse over the submenus to see the animation effects.

## Changing Menu Item Triggers

The **C1Menu** control can respond to different trigger events to open a menu item. This topic illustrates how to set the **Trigger** and **TriggerEvent** properties to change the trigger that opens the menu items in either the Design View or the Source View.

#### In Design View

1. Click the **C1Menu** smart tag to open the **C1Menu Tasks** menu.
2. Click the **Edit Menu** link to open the **C1Menu DesignerForm**.



3. Locate the **Trigger** and **TriggerEvent** properties in the list.
4. Set the **Trigger** property to **.wijmo-wijmenu-item** and then use the **TriggerEvent** drop-down list to choose a **Trigger Event**. For this task, set the **Trigger Event** to **Rtclick** as in the following image.

▶ SlidingAnimation	C1.Web.Wijmo.Controls.SlideAnimation
Theme	cobalt
ToolTip	
Trigger	.wijmo-wijmenu-item
TriggerEvent	Rtclick
UseCDN	False

5. Click **OK** and then press F5 to run your application. The submenus will appear when you right-click on the menu item.

### In Source View

In Source View, add `TriggerEvent="Rtclick"` and `Trigger=".wijmo-wijmenu-item"` to the `<wijmo:C1Menu>` tags.

Press F5 to run your project. Note that you now need to right-click a menu item to make the submenu drop down.

## C1Menu Item Functions

The **C1Menu** control has an extensive client-side API. This topic illustrates how to call the client-side methods using the same patterns you would see in a jQuery UI using the Source View.

1. In Source View, add the following markup between the `<wijmo:C1Menu>` tags.
  - Markup to add

```
<Items>
```

```
<wijmo:C1MenuItem Text="MenuItem" runat="server">
</wijmo:C1MenuItem>
```

```

        <wijmo:C1MenuItem ID="C1MenuItem1" Text="Breaking News"
runat="server">
            <Items>
                <wijmo:C1MenuItem runat="server"
Header="true" Text="header2">
                    </wijmo:C1MenuItem>
                <wijmo:C1MenuItem runat="server"
Separator="true">
                    </wijmo:C1MenuItem>
                <wijmo:C1MenuItem runat="server"
Text="Entertainment">
                    </wijmo:C1MenuItem>
                <wijmo:C1MenuItem ID="C1MenuItem2"
runat="server" Text="Politics"
NavigateUrl="http://www.w3schools.com/tags/html5.asp">
                    </wijmo:C1MenuItem>
                <wijmo:C1MenuItem ID="C1MenuItem3"
runat="server" Text="A&E">
                    </wijmo:C1MenuItem>
                <wijmo:C1MenuItem ID="C1MenuItem4"
runat="server" Text="Sports">
                    </wijmo:C1MenuItem>
                <wijmo:C1MenuItem ID="C1MenuItem5"
runat="server" Text="Local">
                    </wijmo:C1MenuItem>
                <wijmo:C1MenuItem ID="C1MenuItem6"
runat="server" Text="Health">
                    </wijmo:C1MenuItem>
            </Items>
        </wijmo:C1MenuItem>
        <wijmo:C1MenuItem ID="C1MenuItem7" runat="server"
Text="Entertainment">
            <Items>
                <wijmo:C1MenuItem ID="C1MenuItem8"
runat="server" Text="Celebrity news">
                    </wijmo:C1MenuItem>
                <wijmo:C1MenuItem ID="C1MenuItem9"
runat="server" Text="Gossip">
                    </wijmo:C1MenuItem>
                <wijmo:C1MenuItem ID="C1MenuItem10"
runat="server" Text="Movies">
                    </wijmo:C1MenuItem>
            </Items>
        </wijmo:C1MenuItem>

```

```

        <wijmo:C1MenuItem ID="C1MenuItem11"
runat="server" Text="Music">
            <Items>
                <wijmo:C1MenuItem
ID="C1MenuItem12" runat="server" Text="Alternative">
                    </wijmo:C1MenuItem>
                <wijmo:C1MenuItem
ID="C1MenuItem13" runat="server" Text="Country">
                    </wijmo:C1MenuItem>
                <wijmo:C1MenuItem
ID="C1MenuItem14" runat="server" Text="Dance">
                    </wijmo:C1MenuItem>
                <wijmo:C1MenuItem
ID="C1MenuItem15" runat="server" Text="Electronica">
                    </wijmo:C1MenuItem>
                <wijmo:C1MenuItem
ID="C1MenuItem16" runat="server" Text="Metal">
                    </wijmo:C1MenuItem>
                <wijmo:C1MenuItem
ID="C1MenuItem17" runat="server" Text="Pop">
                    </wijmo:C1MenuItem>
                <wijmo:C1MenuItem
ID="C1MenuItem18" runat="server" Text="Rock">
                    </wijmo:C1MenuItem>
            <Items>
                <wijmo:C1MenuItem
ID="C1MenuItem19" runat="server" Text="Bands">
                    <Items>
                        <wijmo:C1MenuItem
ID="C1MenuItem22" runat="server" Text="Dokken">
                            </wijmo:C1MenuItem>
                    </Items>
                </wijmo:C1MenuItem>
            </Items>
                <wijmo:C1MenuItem
ID="C1MenuItem20" runat="server" Text="Fan Clubs">
                    </wijmo:C1MenuItem>
                <wijmo:C1MenuItem
ID="C1MenuItem21" runat="server" Text="Songs">
                    </wijmo:C1MenuItem>
            </Items>
        </wijmo:C1MenuItem>
    </Items>

```

```

        </wijmo:C1MenuItem>
        <wijmo:C1MenuItem ID="C1MenuItem23"
runat="server" Text="Slide shows">
        </wijmo:C1MenuItem>
        <wijmo:C1MenuItem ID="C1MenuItem24"
runat="server" Text="Red carpet">
        </wijmo:C1MenuItem>
    </Items>
</wijmo:C1MenuItem>
<wijmo:C1MenuItem ID="C1MenuItem25" Text="Finance"
runat="server">
    <Items>
        <wijmo:C1MenuItem ID="C1MenuItem26"
Text="Personal" runat="server">
            <Items>
                <wijmo:C1MenuItem
ID="C1MenuItem28" Text="Loans" runat="server">
                </wijmo:C1MenuItem>
                <wijmo:C1MenuItem
ID="C1MenuItem29" Text="Savings" runat="server">
                </wijmo:C1MenuItem>
                <wijmo:C1MenuItem
ID="C1MenuItem30" Text="Mortgage" runat="server">
                </wijmo:C1MenuItem>
                <wijmo:C1MenuItem
ID="C1MenuItem31" Text="Debt" runat="server">
                </wijmo:C1MenuItem>
            </Items>
        </wijmo:C1MenuItem>
        <wijmo:C1MenuItem ID="C1MenuItem27"
Text="Business" runat="server">
        </wijmo:C1MenuItem>
    </Items>
</wijmo:C1MenuItem>
<wijmo:C1MenuItem ID="C1MenuItem32" Text="Food &#38;
Cooking" runat="server">
    <Items>
        <wijmo:C1MenuItem ID="C1MenuItem33"
Text="Breakfast" runat="server">
        </wijmo:C1MenuItem>
        <wijmo:C1MenuItem ID="C1MenuItem34"
Text="Lunch" runat="server">

```

```

        </wijmo:C1MenuItem>
        <wijmo:C1MenuItem ID="C1MenuItem35"
Text="Dinner" runat="server">
        </wijmo:C1MenuItem>
        <wijmo:C1MenuItem ID="C1MenuItem36"
Text="Dessert" runat="server">
                <Items>
                        <wijmo:C1MenuItem
ID="C1MenuItem37" Text="Dump Cake" runat="server">
                                </wijmo:C1MenuItem>
                        <wijmo:C1MenuItem
ID="C1MenuItem38" Text="Doritos" runat="server">
                                </wijmo:C1MenuItem>
                        <wijmo:C1MenuItem
ID="C1MenuItem39" Text="Both please" runat="server">
                                </wijmo:C1MenuItem>
                </Items>
        </wijmo:C1MenuItem>
    </Items>
    </wijmo:C1MenuItem>
    <wijmo:C1MenuItem ID="C1MenuItem40" Text="Lifestyle"
runat="server">
    </wijmo:C1MenuItem>
    <wijmo:C1MenuItem ID="C1MenuItem41" Text="News"
runat="server">
    </wijmo:C1MenuItem>
    <wijmo:C1MenuItem ID="C1MenuItem42" Text="Politics"
runat="server">
    </wijmo:C1MenuItem>
    <wijmo:C1MenuItem ID="C1MenuItem43" Text="Sports"
runat="server">
    </wijmo:C1MenuItem>
    <wijmo:C1MenuItem ID="C1MenuItem44" Text="Novels"
runat="server">
    </wijmo:C1MenuItem>
    <wijmo:C1MenuItem ID="C1MenuItem45" Text="Magazine"
runat="server">
    </wijmo:C1MenuItem>
    <wijmo:C1MenuItem ID="C1MenuItem46" Text="Books"
runat="server">
    </wijmo:C1MenuItem>

```

```

        <wijmo:C1MenuItem ID="C1MenuItem47" Text="Education"
runat="server">
        </wijmo:C1MenuItem>
    </Items>

```

2. After the closing `<wijmo:C1Menu>` tag, insert the following markup to create the button controls.

```

<p>
    <input type="button" id="previous" value="previous" />
    <input type="button" id="next" value="next" />
    <input type="button" id="previousPage" value="previousPage" />
    <input type="button" id="nextPage" value="nextPage" />
</p>

```

3. Use the following script to call the client-side functions.

```

<script type="text/javascript">
    var count = 0;
    $(document).ready(function () {
        $("#previous").click(function () {
            $("#<%= Menu1.ClientID
%>").focus().clmenu("previous");
            count++;
        });
        $("#next").click(function () {
            $("#<%= Menu1.ClientID
%>").focus().clmenu("next");
            count++;
        });
        $("#previousPage").click(function () {
            if (count === 0) {
                $("#<%= Menu1.ClientID
%>").find(".wijmo-wijmenu-link:first").click();
            }
            $("#<%= Menu1.ClientID
%>").clmenu("previousPage");
            count++;
        });
        $("#nextPage").click(function () {
            if (count === 0) {
                $("#<%= Menu1.ClientID
%>").find(".wijmo-wijmenu-link:first").click();
            }
        });
    });

```

```

        $("#<%= Menu1.ClientID
    >").c1menu("nextPage");
        count++;
    });
});

</script>

```

4. Press F5 to run your program. The menu should resemble the following image.



## Dynamically Adding Items to C1Menu

This topic illustrates how to use the client-side controls to dynamically add items to **C1Menu**.

1. Go to the Source View and insert the following markup between the `<wijmo:C1Menu>` tags to populate the menu.

```

<Items>
    <wijmo:c1menuitem id="C1MenuItem1" runat="server" text="Menu
item">
    </wijmo:c1menuitem>
    <wijmo:c1menuitem id="C1MenuItem2" runat="server"
separator="true">
    </wijmo:c1menuitem>
    <wijmo:c1menuitem id="C1MenuItem3" runat="server"
text="Vertical" value="DynamicOrientationItem">
    <Items>
        <wijmo:C1MenuItem ID="C1MenuItem4" runat="server"
Text="Menu item">

```

```

        </wijmo:C1MenuItem>
        <wijmo:C1MenuItem ID="C1MenuItem5" runat="server"
Text="Menu item">
        </wijmo:C1MenuItem>
        <wijmo:C1MenuItem ID="C1MenuItem6" runat="server"
Text="Menu item">
        </wijmo:C1MenuItem>
        <wijmo:C1MenuItem ID="C1MenuItem7" runat="server"
Text="Menu item">
        </wijmo:C1MenuItem>
        <wijmo:C1MenuItem ID="C1MenuItem8" runat="server"
Text="Menu item">
        </wijmo:C1MenuItem>
    </Items>
</wijmo:clmenuitem>
    <wijmo:clmenuitem id="C1MenuItem9" runat="server" text="Menu
item">
    </wijmo:clmenuitem>
    <wijmo:clmenuitem id="C1MenuItem10" runat="server"
text="Menu item">
    </wijmo:clmenuitem>
</Items>

```

2. After the closing `</Items>` tag, insert the following markup to create the buttons that will control the add and remove functions.

```

<fieldset>
    <legend>Remove function</legend>
    <label for="tbSelector">
        Selector</label>
    <input type="text" id="tbSelector" />
    <label for="tbIndex">
        Index</label>
    <input type="text" id="tbIndex" />
    <input type="button" value="Remove" onclick="remove()" />
</fieldset>
<fieldset>
    <legend>Add function</legend>
    <label for="tbItem">
        item</label>
    <input type="text" id="tbItem" />
    <label for="tbAddSelector">

```

```

        Selector</label>
<input type="text" id="tbAddSelector" />
<label for="tbAddIndex">
    Index</label>
<input type="text" id="tbAddIndex" />
<input type="button" value="Add" onclick="add()" />
</fieldset>

```

3. Use the following script to initialize the buttons.

```

<script type="text/javascript">
    function remove() {
        var index, selector;
        if ($("#tbIndex").val() != "" &&
!isNaN($("#tbIndex").val())) {
            index = parseInt($("#tbIndex").val());
        }
        if ($("#tbSelector").val() != "") {
            selector = $("#tbSelector").val();
        }
        if (!selector && index != undefined) {
            selector = index;
            index = null;
        }
        $("#<%= Menu1.ClientID %>").clmenu("remove", selector,
index);
    }

    function add() {
        var index, selector, item;
        item = $("#tbItem").val();
        if ($("#tbAddIndex").val() != "" &&
!isNaN($("#tbAddIndex").val())) {
            index = parseInt($("#tbAddIndex").val());
        }
        if ($("#tbAddSelector").val() != "") {
            selector = $("#tbAddSelector").val();
        }
        if (!selecotr && index != undefined) {
            selector = index;
            index = null;
        }
    }

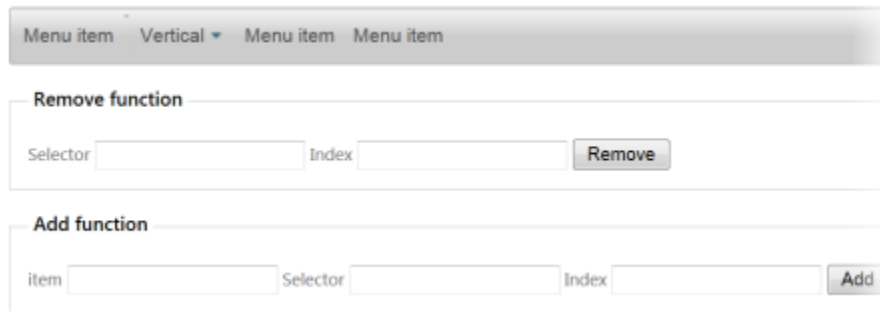
```

```

    }
    $("#<%= Menu1.ClientID %>").c1menu("add", item, selector,
index);
}
</script>

```

4. Press F5 to run your program. The **C1Menu** control should resemble the following image.



## Populating C1Menu with a Site Map

This lesson illustrates how to populate a C1Menu with site map data.

To create the Site Map and bind it to the **C1Menu** 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" />

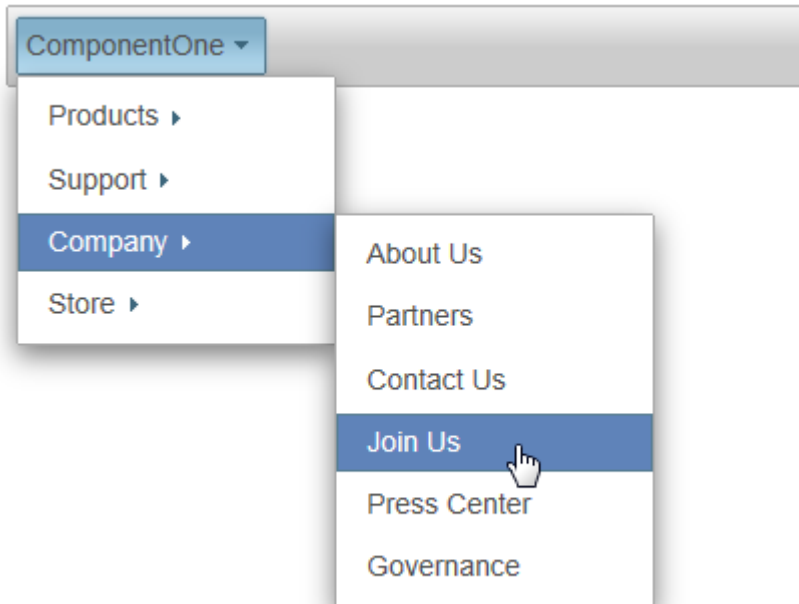
```

```

        <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 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>

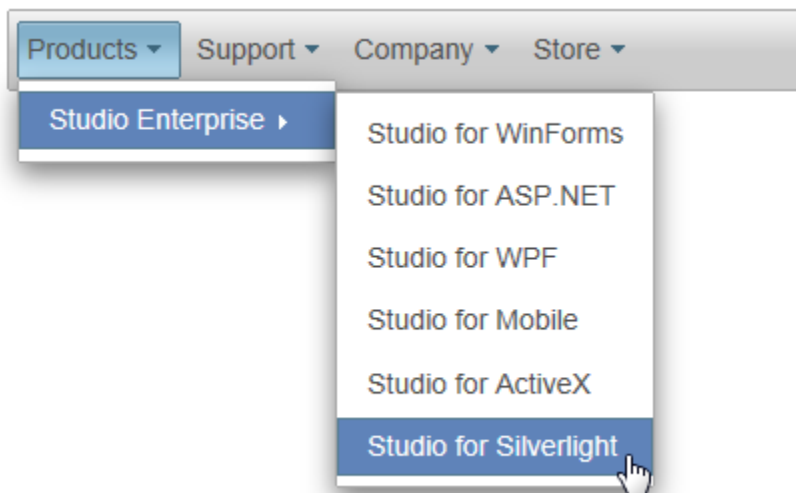
```

4. Open C1Menu 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 **C1Menu** 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 `C1Menu`.

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 C1Menu with XML

This tutorial teaches you how installed templates, add the XML Data Source component to the Web site, assign it to the `C1Menu` control, and then set the binding for the `C1Menu`.

Complete the following steps:

1. From the Toolbox, double-click the C1Menu 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 "Menu\_Hierarchy.xml". The XML file opens.
  - b. In the XML view, add the following data to the **Menu\_Hierarchy.xml** document:

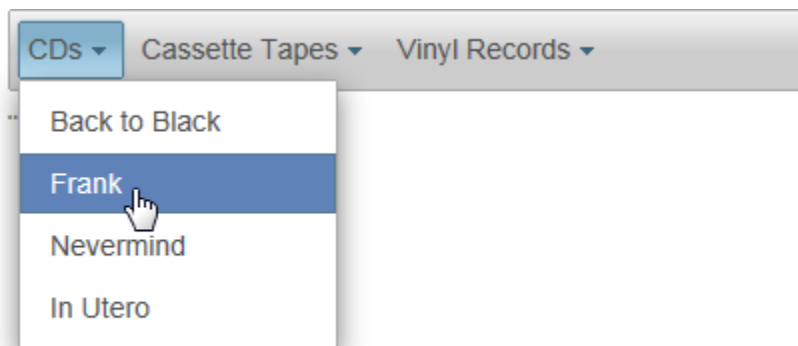
```
<?xml version="1.0" encoding="utf-8" ?>
<root>
  <MenuItem Text="CDs">\
    <MenuItem Text="Back to Black"></MenuItem>
    <MenuItem Text="Frank"></MenuItem>
    <MenuItem Text="Nevermind"></MenuItem>
    <MenuItem Text="In Utero"></MenuItem>
  </MenuItem>
  <MenuItem Text="Cassette Tapes">
    <MenuItem Text="Bleach"></MenuItem>
    <MenuItem Text="Cheap Thrills"></MenuItem>
    <MenuItem Text="Dangerous"></MenuItem>
    <MenuItem Text="Bad"></MenuItem>
  </MenuItem>
  <MenuItem Text="Vinyl Records">
    <MenuItem Text="Axis: Bold as Love"></MenuItem>
    <MenuItem Text="Full Circle"></MenuItem>
    <MenuItem Text="Off The Wall"></MenuItem>
    <MenuItem Text="Other Voices"></MenuItem>
  </MenuItem>
</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 **C1Menu**'s smart tag to open the **C1Menu 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 **C1Menu**'s smart tag to open the **C1Menu Tasks** menu; click **Configure Data Source**.  
The **Configure Data Source** dialog box opens.
  - b. In the XPath expression text field, enter "root/MenuItem". This will select all MenuItem's that are children of root so that the MenuItem's 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 **Menu\_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 **C1MenuItems**.
    - a. Click **C1Menu**'s smart tag to open the **C1Menu 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 "MenuItem".
      - Set the **TextField** property to "Text".
    - d. Click **OK** to close the **Bindings Collection Editor**:
  7. Press F5 to run the project.

✔ **This topic illustrates the following:**

As the project is running, click through the menu and observe that the data from the **Menu\_Hierarchy.xml** file is reflected in the **C1Menu** control.



## Saving and Loading a C1Menu from XML

The following tasks show you how to save your C1Menu control as an .xml file and then load it into your project using the designer.

### Save the C1Menu as XML

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

1. Click **C1Menu**'s smart tag and select **Edit Menu** to open the **C1Menu Designer Form**.

2. Navigate to File | Save as XML.
3. Name the .xml file for your **C1Menu** and browse to where you would like to save it.
4. Click **OK** to close the **Menu Designer Form** dialog box.

#### **Load an Existing XML C1Menu into your Project**

To load the **C1Menu** control you saved as an .xml file into your project:

1. Click **C1Menu**'s smart tag and select **Edit Menu** to open the **C1Menu Designer Form**.
2. Open the C1Menu Designer Form.
3. Navigate to **File | Load From XML** and click open to open the existing .xml file.

#### **Load an Existing XML C1Menu in Code**

To load the **C1Menu** control you saved as an .xml file into your project:

1. Create an XML file for the C1Menu structure.
2. Call the LoadLayout method to load the items, passing in the path to the file:
  - Visual Basic

```
C1Menu1.LoadLayout("c:\\Visual Studio
2005\\WebSites\\LoadLayoutEX\\App_Data\\C1MenuControl.xml")
```
  - C#

```
C1Menu1.LoadLayout("c:\\Visual Studio
2005\\WebSites\\LoadLayoutEX\\App_Data\\C1MenuControl.xml");
```
3. Press F5 to run the program.



# Menu 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 **Maps 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.

- [C1Menu options](#)
- [C1Menu events](#)
- [C1Menu methods](#)

## Using the Wijmo CDN

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

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

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

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

```
<script src="http://cdn.wijmo.com/jquery.wijmo-open.all.2.0.0.min.js"
type="text/javascript"></script>

<script src="http://cdn.wijmo.com/jquery.wijmo-
complete.all.2.0.0.min.js" type="text/javascript"></script>
```

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

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



