Cover Page

Table of Contents

Overview

ComboBox for ASP.NET Web Forms is a full-featured combo box control that combines an editable text box with an auto-searchable drop-down list.	**Getting Started** To get started, review the following topics: • Key Features • Quick Start • Samples

Help with ASP.NET Web Forms Edition

Getting Started

For information on installing **ComponentOne Studio ASP.NET Web Forms Edition**, licensing, technical support, namespaces and creating a project with the control, please visit Getting Started with ASP.NET Web Forms Edition.

Key Features

ComboBox for ASP.NET Web Forms provides the following unique key features:

- **Auto-searchable Drop-down list**
  Locate items quickly by typing the first few characters. ComboBox will automatically search the list and select the items for you as you type.
- **Load on Demand**
  Populate items dynamically from the server using AJAX. This helps to keep page sizes small and manageable.
- **Item Selection**
  The C1ComboBox control provides single and multiple selection modes allowing end-users to select one or several items in the drop-down list.
- **Resizable Drop-down List**
  The C1ComboBox control provides single and multiple selection modes allowing end-users to select one or several items in the drop-down list.
- **Theming**
  With just a click of th e SmartTag, change the bar charts look by selecting one of the 5 premium themes (Midnight, Aristo, Rocket, Cobalt, and Sterling). Optionally, use ThemeRoller from jQuery UI to create a customized theme.
- **CSS Support**
  Use a cascading style sheet (CSS) style to define custom skins. CSS support allows you to match the accordion to your organization's standards.

Quick Start

The C1ComboBox Quick Start describes how to get started with the ASP.NET control, **C1ComboBox**. In this quick start, you will create an ASP.NET application containing one **C1ComboBox** control and three **C1ComboBoxItems**. Declare one Label control in an .aspx file. Then create an event handler for the **SelectedItem** event which displays the value of the selected combobox item.

Step 1 of 4: Creating an Application

In this topic you will create an ASP.NET Edition web application and add a **C1ComboBox** control to the web form.

1. Begin by creating an ASP.NET Edition Web application. Note that if using Visual Studio 2008, you must add a **ScriptManager** control to the form. If using Visual Studio 2005, the **ScriptManager** control is automatically added to the form.
2. Add the following references to your project:
    1. C1.Web.Wijmo.Controls.4.dll
    2. C1.Web.Wijmo.Controls.Design.4.dll
    3. C1.C1Report.4.dll
3. Add the controls to the 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 Web Forms components appear within a tab in the Toolbox, select **Add Tab** from the context menu and type in the tab name (for example, Studio for ASP.NET Web Forms).
    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 Assembly Name (click the **Assembly Name** column header) and check the check boxes for all components corresponding to Assembly Name **C1.Web.Wijmo.Controls.x**.
    5. Click **OK** to close the dialog box. The controls are added to the Visual Studio Toolbox.
4. Add a new Web Form to your project. Right-click the project name in the Solution Explorer and select **Add | New Item**. Within the **Add New Item** dialog box select **Web Form** from the list of templates. Provide a name to the form.
5. While in Design view, navigate to the Visual Studio Toolbox and double-click the **C1ComboBox** control. This adds combobox to your form.
   Notice in the Source view of your form, markup for C1ComboBox gets added within tags inside the <body></body> tags.

Step 2 of 4: Adding Items to C1ComboBox

In this step, you will use the **C1ComboBox Editor** to add items to a **C1ComboBox** control that will appear at run time when you click on its dropdown arrow.

1. Click the C1ComboBox smart tag and select **Edit Items** from the **C1ComboBox Tasks** menu.
   The **C1ComboBox Editor** appears.
2. Click on the C1ComboxItem from the **Add Child Item** button three times to get three C1ComboBoxItems.
3. Set the values for the **Text** and **Value** properties for each C1ComboBoxItem to the following:
    - C1ComboItem1 **Text** property to Coach and **Value** property to "Coach Cabin"
    - C1ComboItem2 **Text** property to Business and **Value** property to "Business Cabin"
    - C1ComboItem3 **Text** property to First and **Value** property to "First Cabin"
4. Click **OK** to apply the changes to the **C1ComboBox** control and close the **C1ComboBox Editor**.

Step 3 of 4: Create an Event Handler for the Selected Item

In this topic the **selectedItem** event is triggered when an item in the combobox is selected.

1. Select the **Source** tab and add the following script for the **C1ComboBox1_OnClientChanged** function before the **<body>** tag:

   To write code in Client Side script

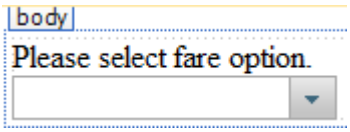
   ```
   <script id="scriptInit" type="text/javascript">
               function C1ComboBox1_OnClientChanged(e, data) {
                   var val = data.selectedItem.value;
                   $('#output').html('I selected the ' + val + ' airfare option. ');
               }
   </script>
   ```

2. Add a **Label** to the source page before the C1ComboBox tags so the **Label** appears above the **C1ComboBox**.

   To write code in Source View

   ```
    <label id="output">
         Please select fare option.</label>
   ```

   The label should appear like the following

3. Within the C1ComboBox tags assign the `onclientchanged="C1ComboBox1_OnClientChanged"` function to the onclientchanged property so it appears like the following:

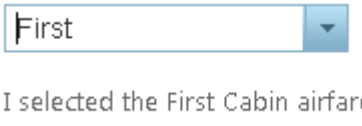
   To write code in Source View

   ```
   <cc1:C1ComboBox ID="C1ComboBox1" runat="server" Width="160px"
               onclientchanged="C1ComboBox1_OnClientChanged">
   ```

Step 4 of 4: Running the Project

Press **F5** to run the project and view the following:

- Select an item from the list.
  The value of the item selected appears updated in the text below.

  I selected the First Cabin airfare option.

Congratulations! You have successfully completed the **ComboBox for ASP.NET Web Forms Quick Start**.

Design-Time Support

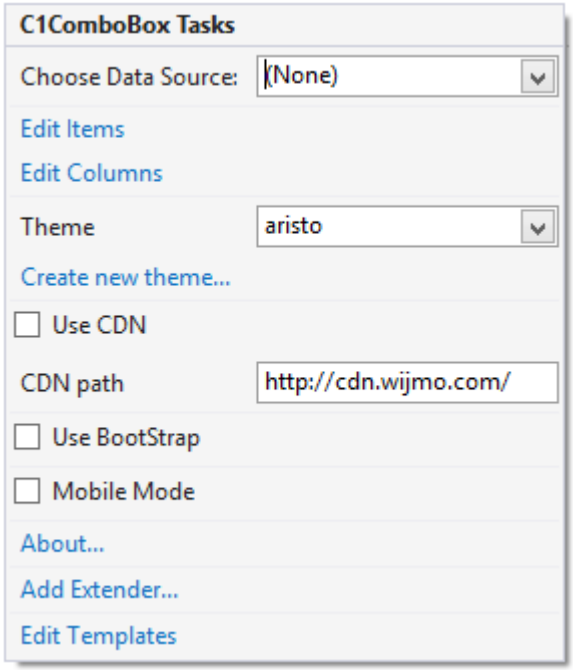
C1ComboBox provides customized context menus, smart tags, and a designer that offers rich design-time support and simplifies working with the object model.

The following sections describe how to use C1ComboBox design-time environment to configure the **C1ComboBox** control.

Smart Tag

In Visual Studio, the **C1ComboBox** control includes a smart tag. A smart tag represents a short-cut tasks menu that provides the most commonly used properties in **C1ComboBox**.

To access the **C1ComboBox Tasks** menu, click on the smart tag in the upper-right corner of the **C1ComboBox** control. This will open the **C1CombBox Tasks** menu.



The **C1ComboBox 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 Items
Clicking the **Edit Items** link item opens the **C1ComboBox** editor.

Edit Columns
Clicking the Edit Columns link item opens the **Columns Designer** editor.

Theme
The **Theme** drop-down box allows you to set the **Theme** property and change the C1ComboBox control's appearance to one of the five predefined themes. By default this is set to the **Aristo** theme. For more information about available visual styles, see Themes.

Create new theme
The **Create new theme** option opens **ThemeRoller for Visual Studio**. This allows you to create a customized theme without leaving your development environment. To find more information on using ThemeRoller in you application, see ThemeRoller for Visual Studio**.**

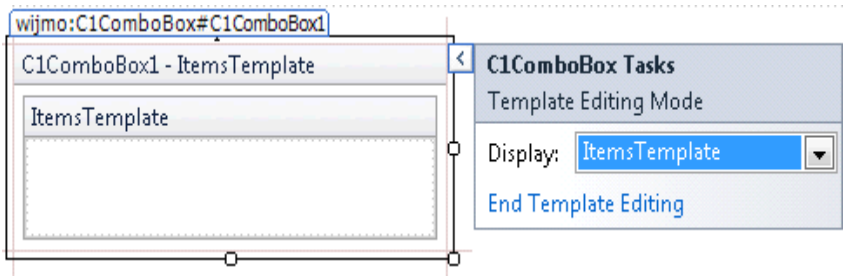
Use CDN
When the **Use CDN** checkbox is selected it loads the client resources from CDN. This is not selected by default.

CDN path
Displays the url path of the **CDN**.

Use Bootstrap
Selecting the **Use Bootstrap** option applies Bootstrap theming to your control. To find more information on using Bootstrap theming in your application, see Bootstrap theming.

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

Edit Templates
Clicking on the **Edit Templates** item switches the **C1ComboBox** control to **Template Editing Mode**:



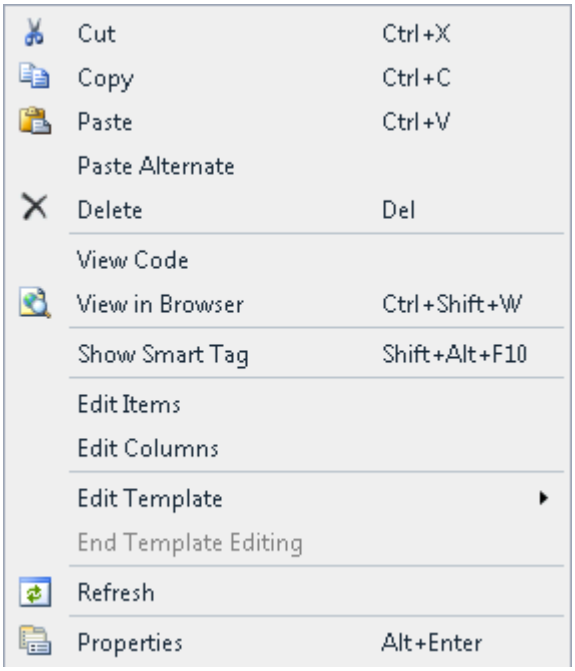
In **Template Editing Mode**, the **C1Combobox Tasks** menu appears with different options:

- **Display**
  Selecting the Display drop-down arrow will open the **Items Template** item that can be customized. Select the **Items Template** from this list to open the template to be edited.
- **End Template Editing**
  Clicking the **End Template Editing** item will end **Template Editing Mode** and return you to the main **C1ComboBox Tasks** menu.

Context Menu

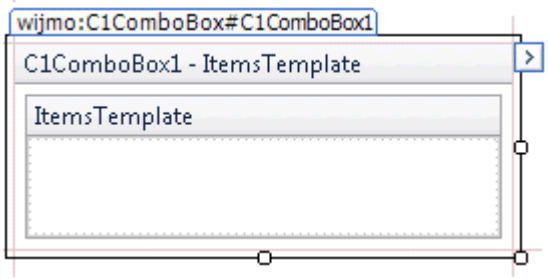
C1ComboBox has additional commands with each context menu that Visual Studio provides for all .NET and ASP.NET controls.

Right-click anywhere on the list to display the **C1ComboBox** context menu:



The context menu commands operate as follows:

Edit Items
Selecting the **Edit Items** menu item opens the **C1ComboBox** editor.
Edit Columns
Selecting the **Edit Columns** menu item opens the **Columns Designer** editor.
Edit Templates
Selecting Item Template from the Edit Templates menu switches the **C1ComboBox** control to **Template Editing Mode**.



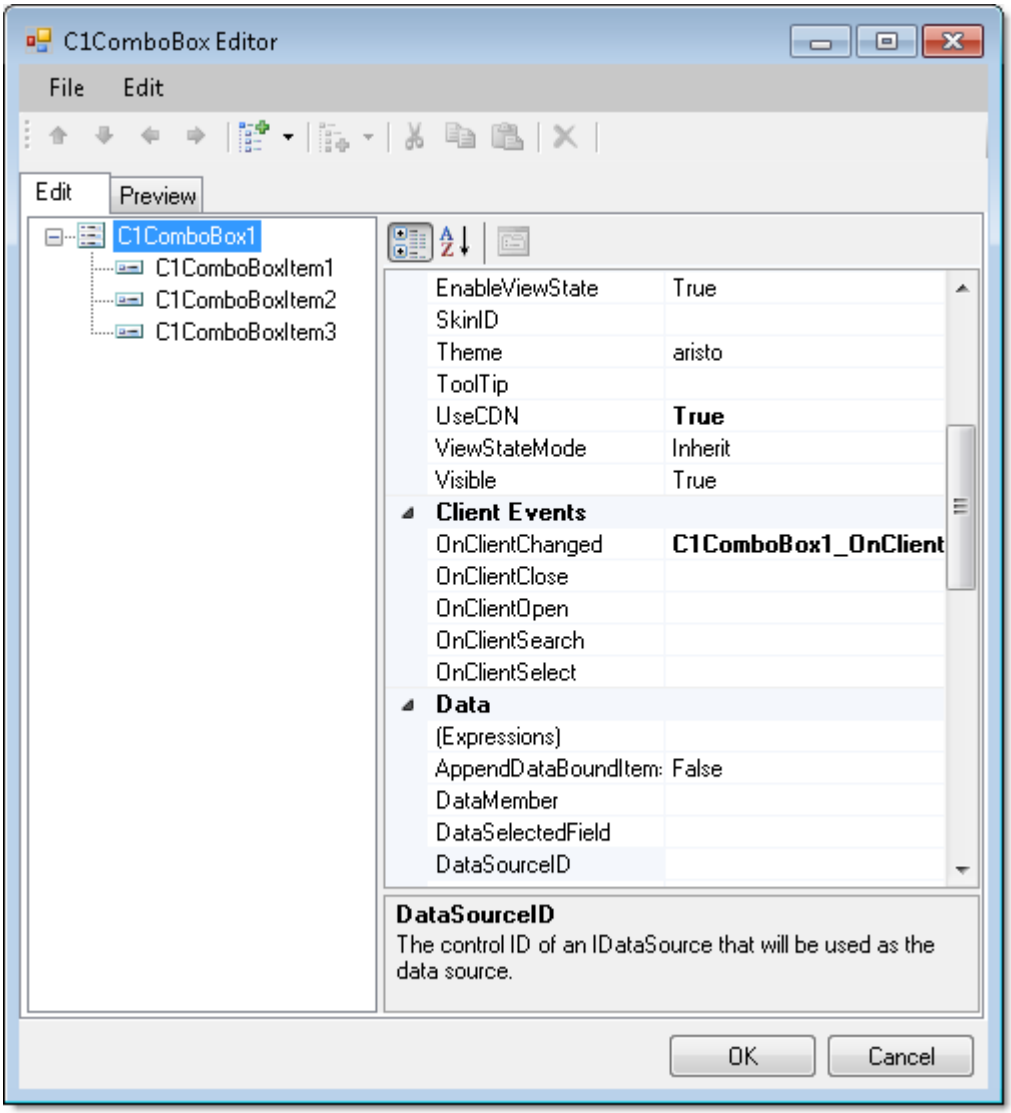
Collection Editors

C1ComboBox includes the following two collection editors to add/remove/modify C1CombBoxItems and columns.

- **C1ComboBox Editor**
- **Columns Designer**

C1ComboBox Editor

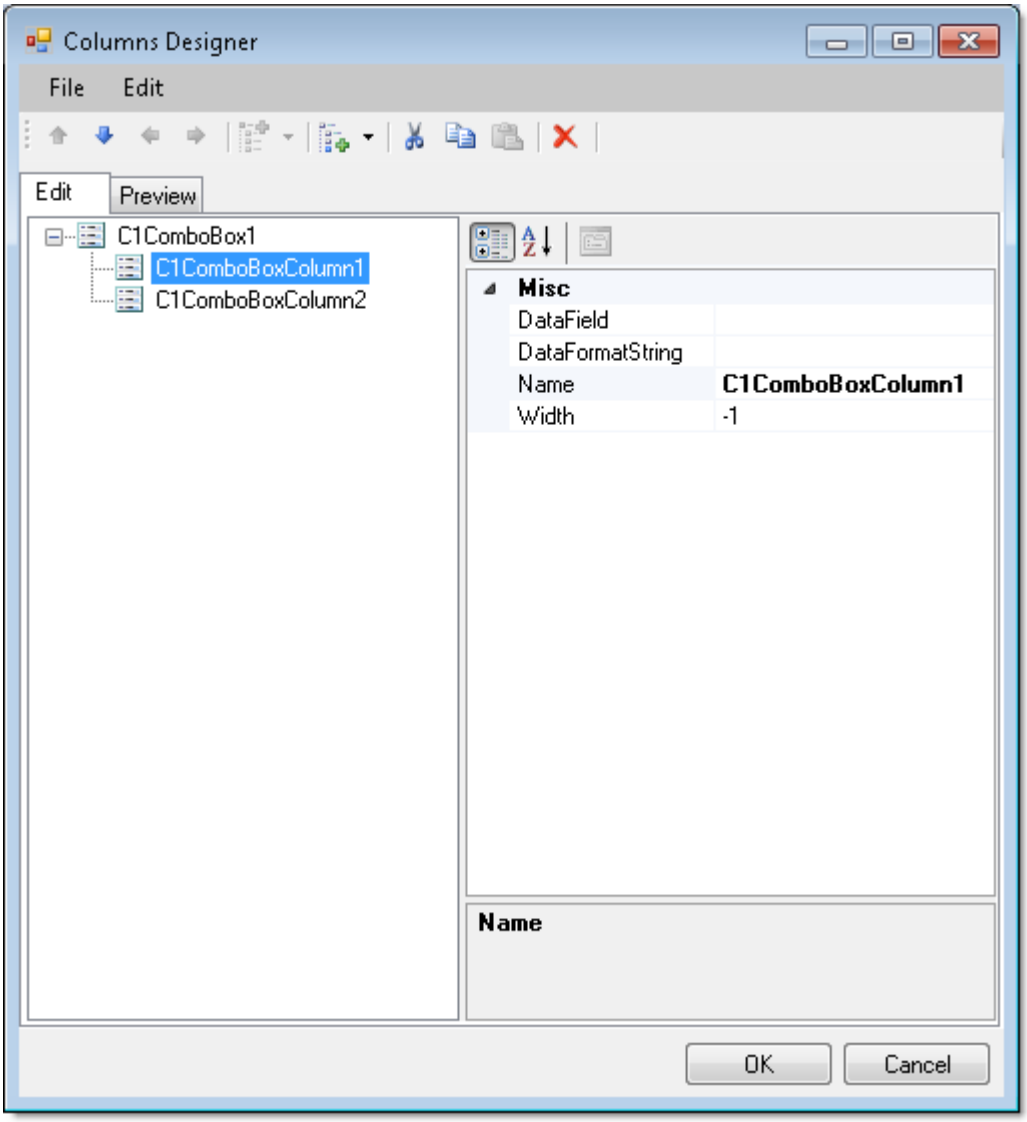
The **C1ComboBox Editor** allows the user to add/remove/modify C1CombBoxItems in the C1CombBox control.



To access the **C1ComboBox Editor**, right-click on the **C1ComboBox** control and select **Edit Items** from its context menu.

Columns Designer

The **Columns Designer** allows you add, remove, or modify columns in the C1ComboBox control.



To access the **Columns Designer**, right-click on the **C1ComboBox** control and select **Columns Designer** from its context menu.

ComboBox Appearance

The combobox's appearance is controlled by the built-in themes or the CSS styles. The combobox themes effects the appearance of all of the combobox elements such as the combobox items and columns.

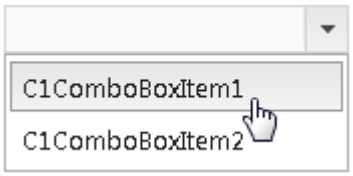
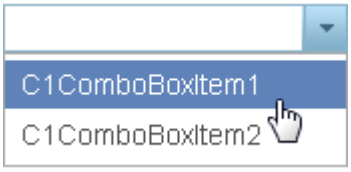
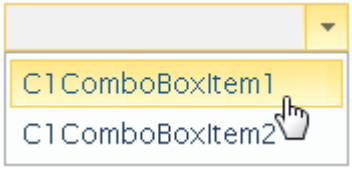
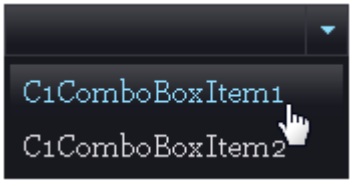
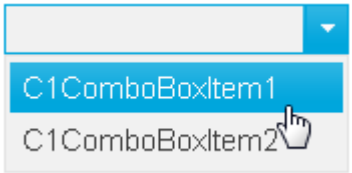
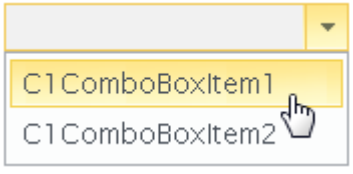
C1ComboBox is designed to make customization easy for you. You have endless possibilities in changing its default appearance.

You can apply CSS styles to modify C1ComboBoxs elements. C1ComboBox provides several built-in CSS Selectors that appear when you select the drop down arrow next to the **CSSClass** property

Themes

C1ComboBox provides five built-in visual styles for the control **Arctic**, **Aristo**, **Cobalt**, **Midnight**, and **Rocket**, and **Sterling** - that can be easily applied to the control by setting the **Theme** property.

The following table illustrates each of the six built-in themes with the hover style shown.

Theme	Appearance
Arctic	
Aristo (default)	
Cobalt	
Midnight	
Rocket	
Sterling	

ThemeRoller for Visual Studio

ThemeRoller for Visual Studio Quick start

Step 1Creating an Application

Step 2Design theme of

Step 3Execution of the application

ThemeRoller for Visual Studio Elements

Task Menu

New theme screen

Theme Roller

Bootstrap for ASP.NET Web Forms Quick Start

C1ComboBox CSS Selectors

You can style any element of the C1ComboBox using CSS styles to make their appearance truly unique. To make the customization process easier, ComponentOne includes CSS selectors for each of its six built-in themes.

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

For a list of common individual CSS selectors and grouped CSS selectors, select the **C1ComboBox** control in your project, and view the drop-down list next to the **CssClass** property in the **C1ComboBox Visual Studio Properties** window.

The following table details the common individual CSS selectors and grouped CSS selectors. You can combine the individual CSS selectors as a group to make the CSS selector more specific and strong. The grouped CSS selectors define styles for more than one element of the C1ComboBox.

CSS Selectors	Description
.wijmo-wijcombobox	Applies the style to the wijcombobox control.
.wijmo-wijcombobox-active-prev	Applies the style to the previous active wijcombobox.
.wijmo-wijcombobox-cell	Applies the style to the cell in the wijcombobox control

.wijmo-wijcombobox-input	Applies the style to the input wijcombobox.
.wijmo-wijcombobox-item ui-state-active	Applies the style to the active state unlisted item of the wijcombobox.
.wijmo-wijcombobox-item ui-state-hover	Applies the style to the hover state unlisted item of the wijcombobox.
.wijmo-wijcombobox-label	Applies the style to the label of the wijcombobox.
.wijmo-wijcombobox-list	Applies the style to the list of the wijcombobox.
.wijmo-wijcombobox-loading	Applies the style to the loading wijcombobox.
.wijmo-wijcombobox-multicolumn	Applies the style to the multicolumn wijcombobox.
.wijmo-wijcombobox-row	Applies the style to the wijcombobox row.
.wijmo-wijcombobox-rowheader	Applies the style to the rowheader of the wijcombobox.
.wijmo-wijcombobox-trigger	Applies the style to the triggered wijcombobox.
.wijmo-wijcombobox-ul	Applies the style to the unlisted wijcombobox.
.wijmo-wijcombobox-wrapper	Applies the style to the wrapper wijcombobox.

Working with the Client-Side ComboBox

C1ComboBox client side has a very rich client-side object model since most of its members are identical to the members in the server-side control.

When a C1ComboBox control is rendered, an instance of the client-side combobox will be created automatically. This means that you can enjoy the convenience of accessing properties and methods of the C1ComboBox control without having to postback to the server.

Using C1ComboBox/s client-side code, you can implement many features in your Web page without the need to send information to the Web server which takes time. Thus, using C1CombBox's client-side object model can increase the efficiency of your Web site.

Client-Side Events

C1ComboBox includes several client-side events that allow you to manipulate the combobox items in the C1ComboBox control when an action such as selecting the item, opening the drop-down list, or closing the drop-down list occurs.

Each of the client-side events requires two parameters: the **ID** that identifies the sender **C1ComboBox**.

You can use the sever-side properties, listed in the **Client Side Event** table, to specify the name of the JavaScript function that will respond to a particular client-side event. For example, to assign a JavaScript function called "CloseList" to respond to the closed drop-down list, you would set the OnClientClose property to "CloseList".

The following table lists the events that you can use in your client scripts. These properties are defined on the server side, but the actual events or the name you declare for each JavaScript function are defined on the client side.

Client Side Event table

Event Server-Side Property Name	Event Name	Description
OnClientChanged	Changed	A function called when the select item is changed
OnClientClose	Close	A function called when the drop-down list is closed.
OnClientOpen	Open	A function called when the drop-down list is opened.
OnClientSearch	Search	A function called before searching the list.
OnClientSelect	Select	A function called when any item in the list is selected.

Samples

Please be advised that this ComponentOne software tool is accompanied by various sample projects and/or demos, which may make use of other ComponentOne development tools included with ComponentOne Studio Enterprise.

C# Samples

The following pages within the **ControlExplorer** sample installed with **ASP.NET Web Forms Edition** detail the C1ComboBox control's functionality:

Sample	Description
Animation	This sample demonstrates the different animation effects and how you can manipulate their speed.
DataBinding	Demonstrates how to bind C1ComboBox to an XML or Access data source.
Dynamic	Demonstrates how to bind the combobox to the datatable.
LoadOnDemand	This sample demonstrates the different callback events and how you can populate items for c1combobox without postback their speed.
MultipleColumns	Shows how you can easily create multiple columns in the c1combobox by setting the columns option.
Overview	Displays an editable text box/drop-down list on the aspx page.
Position	This sample shows how you can change the position of the drop-down list for the c1combobox by setting the dropDownListPosition option.
Select	This sample illustrates how the select event is triggered when an item in the combobox is selected.

Task-Based Help

The task-based help assumes that you are familiar with programming in Visual Studio. By following the steps outlined in the Help, you will be able to create projects demonstrating a variety of **ComboBox for ASP.NET Web Forms** features and get a good sense of what C1ComboBox can do.

Adding a Custom Theme

ComboBox for ASP.NET Web Forms 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.

Using ThemeRoller for Visual Studio

The new **ThemeRoller for Visual Studio** makes designing beautiful themes for **ASP.NET Web Forms Edition** Controls easy. For more information on creating and editing a **ThemeRoller for Visual Studio** theme, please see ThemeRoller for Visual Studio.

Using a CDN URL

1. Click the **C1ComboBox** smart tag to open the Tasks menu.
2. Select **Use CDN**.
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 sunny theme: http://ajax.aspnetcdn.com/ajax/jquery.ui/1.8.14/themes/sunny/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 ComboBox.

Using jQuery ThemeRoller

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 **Themes** folder.
5. In the Solution Explorer, select the project name and click the refresh button so the **Themes** folder appears in your project.
6. Click the **C1ComboBox** smart tag to open the **Tasks** menu.
7. Select **Use CDN**.
8. Right-click on the **C1ComboBox** control and navigate to the **Theme** property in the Properties window and enter the path to your custom theme .css, for example, **Themes/css/custom-theme/jquery-ui-1.8.16.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 them theme is applied to **C1ComboBox**.

Databinding the ComboBox

Binding a database table to a combobox can be accomplished using the following steps:

1. Add the **ComboBox** control to your page.
2. Retrieve the data from the appropriate database table. In this example, we use the **C1Nwind.mdb** file so copy the **C1Nwind.mdb** from the **App_Data** folder in the Control Explorer sample to your project's **App_Data** folder.
3. Set C1ComboBox1's data properties to the following in your source page:
   - **DataSourceID** property to **AccessDataSource1**. This will set the **ID** of the data source.
   - **DataTextField** property to **UnitPrice**. This will set the field in the data source from which to load the text values.
   - **DataTextFormatString** propertyto **{0:C}**. This will apply the currency formatting.
   - **DataValueField** property to **OrderID**. This will load the **OrderID** field items into the combobox.
4. Add the following to your source page:

   To write code in Source View

```
    <asp:AccessDataSource ID="AccessDataSource1" runat="server"
DataFile="~/App_Data/C1NWind.mdb"
        SelectCommand="SELECT top 100 [UnitPrice], [OrderID] FROM [Order
Details]"></asp:AccessDataSource>
```

5. Set C1ComboBox's **Width** to **160**. This will allow more witdth space to accommodate the items in the dropdown box.
6. Save and run your project and observe the binding values will fill the dropdown list of **C1Combobox**.

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 **Combobox for ASP.NET Web Forms**, 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 Web Forms control.

- wijcombobox documentation
- wijcombobox API

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.

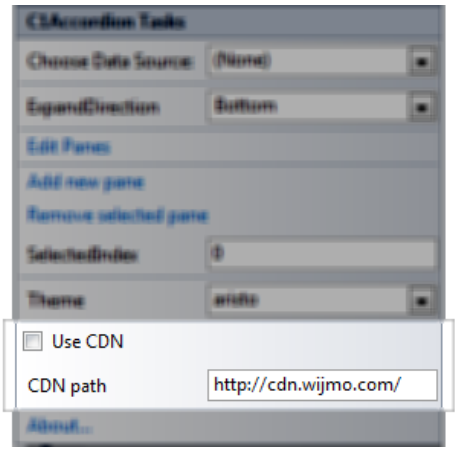
Wijmo has had CDN support from the very beginning. You can find the CDN page at http://wijmo.com/downloads/cdn/. The markup required for loading Wijmo into your page looks similar to this:

To write code in Source View

```
<!--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 **ASP.NET Web Forms Edition** 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.



>