**AutoComplete**

**AutoComplete Demonstration**

Type some characters in this textbox. The web service returns random words that start with the text you have typed.

AutoComplete is an ASP.NET AJAX extender that can be attached to any TextBox control, and will associate that control with a popup panel to display words that begin with the prefix typed into the textbox.

The dropdown with candidate words supplied by a web service is positioned on the bottom left of the text box.

In the sample above, the textbox is associated with an AutoCompleteExtender that pulls words that start with the contents of the textbox using a web service.

When you have typed more content than the specified minimum word length, a popup will show words or phrases starting with that value. Caching is turned on, so typing the same prefix multiple times results in only one call to the web service.

** AutoComplete Properties**

The textbox is linked with an AutoCompleteExtender which is initialized with this code. The*italic* properties are optional:

<ajaxToolkit:AutoCompleteExtender

runat="server"

ID="autoComplete1"

TargetControlID="myTextBox"

ServiceMethod="GetCompletionList"

*ServicePath*="AutoComplete.asmx"

*MinimumPrefixLength*="2"

*CompletionInterval*="1000"

*EnableCaching*="true"

*CompletionSetCount*="20"

*CompletionListCssClass*="autocomplete\_completionListElement"

*CompletionListItemCssClass*="autocomplete\_listItem"

*CompletionListHighlightedItemCssClass*="autocomplete\_highlightedListItem"

*DelimiterCharacters*=";, :">

*<Animations>*

*<OnShow> ... </OnShow>*

*<OnHide> ... </OnHide>*

*</Animations>*

</ajaxToolkit:AutoCompleteExtender>

**Calendar**

Calendar is an ASP.NET AJAX extender that can be attached to any ASP.NET TextBox control. It provides client-side date-picking functionality with customizable date format and UI in a popup control. You can interact with the calendar by clicking on a day to set the date, or the "Today" link to set the current date.

In addition, the left and right arrows can be used to move forward or back a month. By clicking on the title of the calendar you can change the view from Days in the current month, to Months in the current year. Another click will switch to Years in the current Decade. This action allows you to easily jump to dates in the past or the future from within the calendar control.

The page uses the culture setting **English (United States)** which was specified by the browser. The page properties have Culture="auto" and UICulture="auto" set to enable the same. See this [MSDN article](http://msdn2.microsoft.com/en-us/library/bz9tc508.aspx) for more information. The ScriptManager on this Calendar demo page has EnableScriptGlobalization="true" and EnableScriptLocalization="true".

** Calendar Properties**

The calendar associated with a button has been initialized with this code. The properties in*italic* are optional:

<ajaxToolkit:Calendar runat="server"

TargetControlID="Date1"

*CssClass*="ClassName"

*Format*="MMMM d, yyyy"

*PopupButtonID*="Image1" />

* **TargetControlID** - The ID of the TextBox to extend with the calendar.
* **CssClass** - Name of the CSS class used to style the calendar. See the Calendar Theming section for more information.
* **Format** - [Format string](http://msdn2.microsoft.com/en-us/library/bb79761a-ca08-44ee-b142-b06b3e2fc22b.aspx) used to display the selected date.
* **PopupButtonID** - The ID of a control to show the calendar popup when clicked. If this value is not set, the calendar will pop up when the textbox receives focus.
* **PopupPosition** - Indicates where the calendar popup should appear at the BottomLeft(default), BottomRight, TopLeft, TopRight, Left or Right of the TextBox.
* **SelectedDate** - Indicates the date the Calendar extender is initialized with.

**CascadingDropDown**

|  |  |
| --- | --- |
| Make |  |
| Model |  |
| Color |  |

[No response provided yet]

** CascadingDropDown Description**

CascadingDropDown is an ASP.NET AJAX extender that can be attached to an ASP.NET DropDownList control to get automatic population of a set of DropDownList controls. Each time the selection of one the DropDownList controls changes, the CascadingDropDown makes a call to a specified web service to retrieve the list of values for the next DropDownList in the set.

CascadingDropDown enables a common scenario in which the contents of one list depends on the selection of another list and does so without having to embed the entire data set in the page or transfer it to the client at all. All the logic about the contents of the set of DropDownList controls lives on the server in a web service. This web service can use any suitable method for storing and looking up the relevant data. The sample web service used here reads the data set from a simple hierarchical XML data file. The sample data file shows that the DropDownList items can have distinct names and value (values are optional in the sample). There is also a walkthrough showing how to use a [CascadingDropDown with a database](http://www.ajaxcontroltoolkit.com/Walkthrough/CCDWithDB.aspx).

** CascadingDropDown Properties**

The control above is initialized with this code. The *italic* properties are optional:

<ajaxToolkit:CascadingDropDown ID="CDD1" runat="server"

TargetControlID="DropDownList2"

Category="Model"

*PromptText*="Please select a model"

*LoadingText*="[Loading models...]"

*ServicePath*="CarsService.asmx"

ServiceMethod="GetDropDownContents"

*ParentControlID*="DropDownList1"

*SelectedValue*="SomeValue" />

* **TargetControlID** - The ID of the DropDownList to populate.
* **Category** - The name of the category this DropDownList represents.
* **PromptText** - Optional text to display before the user has selected a value from the DropDownList.
* **PromptValue** - Optional value set when PromptText is displayed.
* **EmptyText** - Optional text to display when the DropDownList has no data to display.
* **EmptyValue** - Optional value set when EmptyText is displayed.
* **LoadingText** - Optional text to display while the data for the DropDownList is being loaded.
* **ServicePath** - Path to a web service that returns the data used to populate the DropDownList. This property should be left null if ServiceMethod refers to a page method. The web service should be decorated with the System.Web.Script.Services.ScriptService attribute.
* **ServiceMethod** - Web service method that returns the data used to populate the DropDownList. The signature of this method must match the following:
* [System.Web.Services.WebMethod]
* [System.Web.Script.Services.ScriptMethod]
* public CascadingDropDownNameValue[] GetDropDownContents(

string knownCategoryValues, string category) { ... }

Note that you can replace "GetDropDownContents" with a naming of your choice, but the return type and parameter name and type must exactly match, including case.

* **ContextKey** - User/page specific context provided to an optional overload of the web method described by ServiceMethod/ServicePath. If the context key is used, it should have the same signature with an additional parameter named contextKey of type string:
* [System.Web.Services.WebMethod]
* [System.Web.Script.Services.ScriptMethod]
* public CascadingDropDownNameValue[] GetDropDownContents(

string knownCategoryValues, string category, string contextKey) { ... }

Note that you can replace "GetDropDownContents" with a name of your choice, but the return type and parameter name and type must exactly match, including case.

* **UseContextKey** - Whether or not the ContextKey property should be used. This will be automatically enabled if the ContextKey property is ever set (on either the client or the server). If the context key is used, it should have the same signature with an additional parameter named contextKey of type string (as described above).
* **ParentControlID** - Optional ID of the parent DropDownList that controls the contents of this DropDownList.
* **SelectedValue** - Optional value to select by default. This needs to exactly match the string representation of a value in the DropDownList.

**DropDown**

Hover over the text below and click to select an option:

Select your favorite exotic ice-cream flavor

** DropDown Description**

DropDown is an ASP.NET AJAX extender that can be attached to almost any ASP.NET control to provide a SharePoint-style drop-down menu. The displayed menu is merely another panel or control. In the above sample the drop-down is a Panel which contains LinkButtons. The drop-down is activated by left- or right-clicking the attached control. If the behavior is attached to a Hyperlink or LinkButton, clicking on the link itself will operate normally.

** DropDown Properties**

The control above is initialized with this code. The *italic* properties are optional:

<ajaxToolkit:DropDownExtender runat="server" ID="DDE"

TargetControlID="TextLabel"

*DropDownControlID*="DropPanel" />

* **TargetControlID** - The ID of the control which needs a drop-down.
* **DropDownControlID** - The ID of the control which will be displayed as the dropdown.
* **Animations** - Generic animations for the DropDown extender. See the [Using Animations](http://www.ajaxcontroltoolkit.com/Walkthrough/UsingAnimations.aspx) walkthrough and [Animation Reference](http://www.ajaxcontroltoolkit.com/Walkthrough/AnimationReference.aspx) for more details.
  + **OnShow** - The OnShow animation will be played each time the dropdown is displayed. The dropdown will be positioned correctly but hidden. The animation can use **<HideAction Visible="true" />** to display the dropdown along with any other visual effects.
  + **OnHide** - The OnHide animation will be played each time the dropdown is hidden.

**FilteredTextBox**

|  |  |
| --- | --- |
| Only digits are allowed here: |  |
| Only lower-case letters are allowed here: |  |
| Only math symbols (+,-,\*,/,=,.) and numbers: |  |
| No digits allowed in this textbox: |  |

** FilteredTextBox Description**

FilteredTextBox is an extender which prevents a user from entering invalid characters into a text box. Note that since this effect can be avoided by deactivating JavaScript, you should use this extender as a convenience for your users, but you must never expect that the data being sent to the server consists of "valid" chars only.

** FilteredTextBox Properties**

The math-symbols example above is initialized with this code:

<ajaxToolkit:FilteredTextBoxExtender ID="ftbe" runat="server"

TargetControlID="TextBox3"

FilterType="Custom, Numbers"

*ValidChars*="+-=/\*()." />

* **TargetControlID** - The ID of the text box for this extender to operate on
* **FilterType** - A the type of filter to apply, as a comma-separated combination of**Numbers**, **LowercaseLetters**, **UppercaseLetters**, and **Custom**. If Custom is specified, the ValidChars field will be used in addition to other settings such as Numbers.
* **FilterMode** - A the filter mode to apply, either **ValidChars** (default) or**InvalidChars**. If set to InvalidChars, FilterType must be set to Custom; if set to ValidChars, FilterType must contain Custom.
* **ValidChars** - A string consisting of all characters considered valid for the text field, if "Custom" is specified as the filter type. Otherwise this parameter is ignored.
* **InvalidChars** - A string consisting of all characters considered invalid for the text field, if "Custom" is specified as the filter type and "InvalidChars" as the filter mode. Otherwise this parameter is ignored.
* **FilterInterval** - An integer containing the interval (in milliseconds) in which the field's contents are filtered, defaults to 250.

**PasswordStrength**

Simple Complexity, Text Indicator  
  
2 more characters  
  
Average Complexity, Bar Indicator  
  
  
  
High Complexity, Text Indicator, Help Indicator  
  


**Note: Recent changes will cause the PasswordStrength control to render differently compared to previous releases. This will affect the padding attributes of the BarIndicator display type. To achieve the same style as previous releases, you will need to add the following to the style for that control.**

padding: 2px 2px 2px 2px;

** PasswordStrength Description**

PasswordStrength is an ASP.NET AJAX extender that can be attached to an ASP.NET TextBox control used for the entry of passwords. The PasswordStrength extender shows the strength of the password in the TextBox and updates itself as the user types the password. The indicator can display the strength of the password as a text message or with a progress bar indicator. The styling and position of both types of indicators is configurable. The required strength of the password is also configurable, allowing the page author to tailor the password strength requirements to their needs. The text messages that describe the current strength of the password can also be configured and their default values have localization support built-in. The second and third extenders' strings are being pulled from Toolkit resources files. We do not have strings for all languages currently so they may show non-localized values for some languages. A help indicator can be used to provide explicit instructions about what changes are required to achieve a strong password. The indicator is displayed when the user begins typing into the TextBox and is hidden from view once the TextBox loses focus.

** PasswordStrength Properties**

The control above is initialized with this code. The *italic* properties are optional:

<ajaxToolkit:PasswordStrength ID="PS" runat="server"

TargetControlID="TextBox1"

*DisplayPosition="RightSide"*

*StrengthIndicatorType="Text"*

*PreferredPasswordLength="10"*

*PrefixText="Strength:"*

*TextCssClass="TextIndicator\_TextBox1"*

*MinimumNumericCharacters="0"*

*MinimumSymbolCharacters="0"*

*RequiresUpperAndLowerCaseCharacters="false"*

*TextStrengthDescriptions="Very Poor;Weak;Average;Strong;Excellent"*

*TextStrengthDescriptionStyles="cssClass1;cssClass2;cssClass3;cssClass4;cssClass5*

*CalculationWeightings="50;15;15;20"* />

* **TargetControlID** - ID of the TextBox to attach to
* **DisplayPosition** - Positioning of the strength indicator relative to the target control
* **StrengthIndicatorType** - Strength indicator type (Text or BarIndicator)
* **PreferredPasswordLength** - Preferred length of the password
* **PrefixText** - Text prefixed to the display text when StrengthIndicatorType=Text
* **TextCssClass** - CSS class applied to the text display when StrengthIndicatorType=Text
* **MinimumNumericCharacters** - Minimum number of numeric characters
* **MinimumSymbolCharacters** - Minimum number of symbol characters (ex: $ ^ \*)
* **RequiresUpperAndLowerCaseCharacters** - Specifies whether mixed case characters are required
* **MinimumLowerCaseCharacters** - Only in effect if RequiresUpperAndLowerCaseCharacters property is true. Specifies the minimum number of lowercase characters required when requiring mixed case characters as part of your password strength considerations.
* **MinimumUpperCaseCharacters** - Only in effect if RequiresUpperAndLowerCaseCharacters property is true. Specifies the minimum number of uppercase characters required when requiring mixed case characters as part of your password strength considerations.
* **TextStrengthDescriptions** - List of semi-colon separated descriptions used when StrengthIndicatorType=Text (Minimum of 2, maximum of 10; order is weakest to strongest)
* **CalculationWeightings** - List of semi-colon separated numeric values used to determine the weighting of a strength characteristic. There must be 4 values specified which must total 100. The default weighting values are defined as 50;15;15;20. This corresponds to password length is 50% of the strength calculation, Numeric criteria is 15% of strength calculation, casing criteria is 15% of calculation, and symbol criteria is 20% of calculation. So the format is 'A;B;C;D' where A = length weighting, B = numeric weighting, C = casing weighting, D = symbol weighting.
* **BarBorderCssClass** - CSS class applied to the bar indicator's border when StrengthIndicatorType=BarIndicator
* **BarIndicatorCssClass** - CSS class applied to the bar indicator's inner bar when StrengthIndicatorType=BarIndicator
* **StrengthStyles** - List of semi-colon separated CSS classes that are used depending on the password's strength. This property will override the BarIndicatorCssClass / TextIndicatorCssClass property if present. The BarIndicatorCssClass / TextIndicatorCssClass property differs in that it attributes one CSS style to the BarIndicator or Text Strength indicator (depending on which type is chosen) regardless of password strength. This property will cause the style to change based on the password strength and also to the number of styles specified in this property. For example, if 2 styles are defined like StrengthStyles="style1;style2" then style1 is applied when the password strength is less than 50%, and style2 is applied when password strength is >= 50%. This property can have up to 10 styles.
* **HelpStatusLabelID** - Control ID of the label used to display help text
* **HelpHandleCssClass** - CSS class applied to the help element used to display a dialog box describing the password requirements
* **HelpHandlePosition** - Positioning of the help handle element relative to the target control

**ConfirmButton**

[Click Me](javascript:__doPostBack('ctl00$SampleContent$LinkButton',''))   
  
  
  
You clicked the Button at 11:43:26 AM.

** ConfirmButton Description**

ConfirmButton is a simple extender that catches clicks on a button (or any instance of a type derived from Button) and displays a message to the user. If the "OK" button is clicked, the button or link functions normally. If not, the click is trapped and the button will not perform its default submit behavior; optionally, a client script is executed if the OnClientCancel property is set. This is useful for delete links or anything else that requires confirmation from the user.

** ConfirmButton Properties**

The control above is initialized with this code. The *italic* properties are optional:

<ajaxToolkit:ConfirmButtonExtender ID="cbe" runat="server"

TargetControlID="LinkButton1"

ConfirmText="Are you sure you want to click this?"

*OnClientCancel*="CancelClick" />

* **TargetControlID** - The ID of the button or link for this extender to operate on.
* **ConfirmText** - The text to show when you want to confirm the click. (Note: HTML entities can be used here (ex: "&#10;" for new-line))
* **OnClientCancel** - The client-side script that executes when the cancel button is clicked in the confirm dialog.
* **ConfirmOnFormSubmit** - True if the confirm dialog should wait until just before the form submits to display. This is useful when ASP.NET validators are in use and the confirm should be shown only after all validators pass.
* **DisplayModalPopupID** - Optionally specifies the ID of a ModalPopup control to use for displaying the confirmation dialog (instead of window.confirm). When using DisplayModalPopupID, the following conditions must be met:
  + The ModalPopup must be configured to work against the same TargetControlID as the ConfirmButton (and should work properly if the ConfirmButton is disabled).
  + The ModalPopup must specify OkControlID and/or CancelControlID to identify the buttons corresponding to window.confirm's OK/Cancel buttons.
  + The ModalPopup must not specify OnOkScript or OnCancelScript.