CAMBRO

DbCombo.net

Table of contents

User Manual
Release v4.1 – January 2003

Introduction

A quick WebControl for picking from databases

DbCombo is a WebControl that enables you to replicate the functionality of an HTML drop-down, but for large volumes of data where page download times would suffer with a SELECT tag.

It has two modes:

- ‘Up-level-browser’ mode. Complex client scripting is used to ‘auto-complete’ the users entry. The client retrieves new records from the server on the fly.
- ‘Down-level-browser’ mode. Standard HTML 3.2 is used, and the postback architecture is utilized to facilitate searching of the data. A ‘search’ button searches, a ‘more…’ button gets more records, and a ‘select’ button selects the current record.

The control itself intelligently chooses between the two modes, based on the browser requesting the page. IE5+ Windows clients get the up-level version; all other browsers get the down-level version.

All textual elements used in the control are customisable, making it easy to use the control on international sites.

Quick Start: Visual Studio Drag & Drop

This Quick Start covers using Visual Studio to set up DbCombo on a web form, accessing the Northwind database (either Access or SQL Server).

1) IMPORTANT: Create a file called DbComboServer.aspx in your web application. Enter the following on the first line, and leave the rest of the file blank:

```
<%@ Page AutoEventWireup="false"
Inherits="Cambro.Web.DbCombo.ServerPage" %>
```

Note – This file should only contain the line above – do not include any other code! You should remove any code that Visual Studio adds.
This page doesn't need a code-behind file, so if one was created, you may delete it.

2) Add DbCombo to your Toolbox:

1. View > Toolbox
2. Right-click on the Toolbox and choose 'Customize Toolbox'
3. Click the .NET Framework Components tab at the top of the dialog box.
4. Scroll down until you find the item 'DbCombo' – click it's check box.
5. Click 'OK' to close the dialog box.

3) Add an instance of DbCombo to a web form:

1. Create a new Web Form in your project.
2. Drag the DbCombo component from the Toolbox onto the form's design window. An instance of the DbCombo component will appear in the design window, and be added to your code.
3. You can edit the properties of this instance using the Properties window (just like any other Toolbox component).
   The value of the RegistrationKey property should be the registration key you have been assigned. You may omit this property, but DbCombo will only function on the localhost domain. Please see the "My Purchases" page of www.dbcombo.net for your registration key.

4) Ensure your database is accessible:

If you wish to use SQL Server… this example assumes that SQL server is installed on the local machine, and that the **Northwind** database is accessible.

If you wish to use Access… this example assumes that the northwind.mdb Access database file is in the same directory as the aspx files.

5) Set up the Server Method:

For DbCombo to function, you need to add a **Server Method**. This is a section of code that goes in your web form's Codebehind, or within a <script runat=server> tag on your web form. The Server Method contains information about your database, and is called by DbCombo to retrieve the appropriate data form your data store.

We have included 4 different sample Server Method blocks below. These cover the SQL and Access databases, with versions provided in both C# and VB.NET. You can modify the example code below to point to any database.

C#, SQL Server:

```
[Cambro.Web.DbCombo.ResultsMethodAttribute(true)]
public static object
DbComboMethod(Cambro.Web.DbCombo.ServerMethodArgs args)
{

     DataSet dataset=new DataSet();

     SqlConnection conn = new SqlConnection("Data Source=(local);
Initial Catalog=Northwind; Integrated Security=SSPI;");

     SqlDataAdapter adapter = new SqlDataAdapter();

     adapter.SelectCommand = new SqlCommand("SELECT TOP "+args.Top+"
CompanyName AS DbComboText, CustomerID AS DbComboValue FROM
Customers WHERE CompanyName LIKE @Query ORDER BY CompanyName",
conn);


adapter.SelectCommand.Parameters.Add("@Query","%"+args.Query+"%");

     adapter.Fill(dataset);

     conn.Close();

     return dataset;

}
```

C#, Access:

```
[Cambro.Web.DbCombo.ResultsMethodAttribute(true)]
public static object
DbComboMethod(Cambro.Web.DbCombo.ServerMethodArgs args)
{
     DataSet dataset=new DataSet();

     OleDbConnection conn = new
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Password=;User
ID=;Data
Source="+HttpContext.Current.Server.MapPath("northwind.mdb")+";");

     OleDbDataAdapter adapter = new OleDbDataAdapter();

     adapter.SelectCommand = new OleDbCommand("SELECT TOP
"+args.Top+" CompanyName AS DbComboText, CustomerID AS DbComboValue
FROM Customers WHERE CompanyName LIKE @Query ORDER BY CompanyName",
conn);


adapter.SelectCommand.Parameters.Add("@Query","%"+args.Query+"%");

     adapter.Fill(dataset);

     conn.Close();

     return dataset;
}
```

VB.NET, SQL Server:

```
<Cambro.Web.DbCombo.ResultsMethodAttribute(true)> _
Public Shared Function DbComboMethod(args As
Cambro.Web.DbCombo.ServerMethodArgs) As Object

     Dim dataset as DataSet = New DataSet()

     Dim conn As SqlConnection = New SqlConnection("Data
Source=(local); Initial Catalog=Northwind;  Integrated
Security=SSPI;")

     Dim adapter As SqlDataAdapter = New SqlDataAdapter()

     adapter.SelectCommand = New SqlCommand("SELECT TOP " &
args.Top.ToString() & " CompanyName AS DbComboText, CustomerID AS
DbComboValue FROM Customers WHERE CompanyName LIKE @Query ORDER BY
CompanyName", conn)

     adapter.SelectCommand.Parameters.Add("@Query","%" & args.Query &
"%")

     adapter.Fill(dataset)

     conn.Close()

     return dataset

End Function
```

VB.NET, Access:

```
<Cambro.Web.DbCombo.ResultsMethodAttribute(true)> _
Public Shared Function DbComboMethod(args As
Cambro.Web.DbCombo.ServerMethodArgs) As Object

     Dim dataset as DataSet = New DataSet()

     Dim conn As OleDbConnection = New
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Password=;User
ID=;Data Source=" &
HttpContext.Current.Server.MapPath("northwind.mdb") & ";")

     Dim adapter As OleDbDataAdapter = New OleDbDataAdapter()

     adapter.SelectCommand = New OleDbCommand("SELECT TOP " &
args.Top.ToString() & " CompanyName AS DbComboText, CustomerID AS
DbComboValue FROM Customers WHERE CompanyName LIKE @Query ORDER BY
CompanyName", conn)

     adapter.SelectCommand.Parameters.Add("@Query","%" & args.Query &
"%")

     adapter.Fill(dataset)

     conn.Close()
     return dataset
End Function
```

Please note:
If you have put the above function in your .aspx page, you will need a couple of

lines at the top (under the <%@ Page ... directive) to import the right namespaces:

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<%@ Import Namespace="System.Data.OleDb" %>
```

If you have put the above function in your code-behind class, you will need a couple of lines at the top to import the right namespaces:

C#:

```
using System.Data;
using System.Data.SqlClient;
using System.Data.OleDb;
```

VB.NET:

```
Imports System.Data
Imports System.Data.SqlClient
Imports System.Data.OleDb
```

6) Navigate to the page in a browser, and you should see the DbCombo control.

Quick Start: Code-based

This Quick Start covers deploying DbCombo by writing code. In this Quick Start DbCombo is set up on a web form to access the Northwind database.

1) IMPORTANT Create a file called DbComboServer.aspx in your web application. Enter the following on the first line, and leave the rest of the file blank:

```
<%@ Page AutoEventWireup="false"
Inherits="Cambro.Web.DbCombo.ServerPage" %>
```

Note – This file should only contain the line above – do not include any other code!
This page doesn't need a code-behind file, so if one was created, you may delete it.

2) Paste the following text in a web-form of your choice (e.g. default.aspx) just below the <@Page … directive.

```
<%@ Register TagPrefix="DbCombo" Namespace="Cambro.Web.DbCombo"
Assembly="Cambro.Web.DbCombo" %>
```

Paste the following text into the body of the page:

```
<h1>My first DbCombo</h1>

<DbCombo:DbCombo
     Runat="server"
     ID="Combo1"
     RegistrationKey="..." />
```

The value of the RegistrationKey property should be the registration key you have been assigned. You may omit this property, but DbCombo will only function on the localhost domain. Please see the "My Purchases" page of www.dbcombo.net for your registration key.

3) This example assumes that either an SQL server database is installed on the local machine, and the Northwind database is accessible, or the northwind.mdb access database is in the same directory as the aspx files. You can modify the example code below to point to any database.

Dependant on your preferred language and database, paste the following into a <script runat=server> block, or to your code-behind:

C#, SQL Server:

```
[Cambro.Web.DbCombo.ResultsMethodAttribute(true)]
public static object
DbComboMethod(Cambro.Web.DbCombo.ServerMethodArgs args)
{

     DataSet dataset=new DataSet();

     SqlConnection conn = new SqlConnection("Data Source=(local);
Initial Catalog=Northwind; Integrated Security=SSPI;");

     SqlDataAdapter adapter = new SqlDataAdapter();

     adapter.SelectCommand = new SqlCommand("SELECT TOP "+args.Top+"
CompanyName AS DbComboText, CustomerID AS DbComboValue FROM
Customers WHERE CompanyName LIKE @Query ORDER BY CompanyName",
conn);


adapter.SelectCommand.Parameters.Add("@Query","%"+args.Query+"%");

     adapter.Fill(dataset);

     conn.Close();

     return dataset;

}
```

C#, Access:

```
[Cambro.Web.DbCombo.ResultsMethodAttribute(true)]
public static object
DbComboMethod(Cambro.Web.DbCombo.ServerMethodArgs args)
{
     DataSet dataset=new DataSet();

     OleDbConnection conn = new
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Password=;User
ID=;Data
Source="+HttpContext.Current.Server.MapPath("northwind.mdb")+";");

     OleDbDataAdapter adapter = new OleDbDataAdapter();

     adapter.SelectCommand = new OleDbCommand("SELECT TOP
"+args.Top+" CompanyName AS DbComboText, CustomerID AS DbComboValue
FROM Customers WHERE CompanyName LIKE @Query ORDER BY CompanyName",
conn);


adapter.SelectCommand.Parameters.Add("@Query","%"+args.Query+"%");

     adapter.Fill(dataset);

     conn.Close();

     return dataset;
}
```

VB.NET, SQL Server:

```
<Cambro.Web.DbCombo.ResultsMethodAttribute(true)> _
Public Shared Function DbComboMethod(args As
Cambro.Web.DbCombo.ServerMethodArgs) As Object

     Dim dataset as DataSet = New DataSet()

     Dim conn As SqlConnection = New SqlConnection("Data
Source=(local); Initial Catalog=Northwind;  Integrated
Security=SSPI;")

     Dim adapter As SqlDataAdapter = New SqlDataAdapter()

     adapter.SelectCommand = New SqlCommand("SELECT TOP " &
args.Top.ToString() & " CompanyName AS DbComboText, CustomerID AS
DbComboValue FROM Customers WHERE CompanyName LIKE @Query ORDER BY
CompanyName", conn)

     adapter.SelectCommand.Parameters.Add("@Query","%" & args.Query &
"%")

     adapter.Fill(dataset)

     conn.Close()

     return dataset

End Function
```

VB.NET, Access:

```
<Cambro.Web.DbCombo.ResultsMethodAttribute(true)> _
Public Shared Function DbComboMethod(args As
Cambro.Web.DbCombo.ServerMethodArgs) As Object

     Dim dataset as DataSet = New DataSet()

     Dim conn As OleDbConnection = New
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Password=;User
ID=;Data Source=" &
HttpContext.Current.Server.MapPath("northwind.mdb") & ";")

     Dim adapter As OleDbDataAdapter = New OleDbDataAdapter()

     adapter.SelectCommand = New OleDbCommand("SELECT TOP " &
args.Top.ToString() & " CompanyName AS DbComboText, CustomerID AS
DbComboValue FROM Customers WHERE CompanyName LIKE @Query ORDER BY
CompanyName", conn)

     adapter.SelectCommand.Parameters.Add("@Query","%" & args.Query &
"%")

     adapter.Fill(dataset)

     conn.Close()
     return dataset
End Function
```

Please note:
If you have put the above function in your .aspx page, you will need a couple of lines at the top (under the <%@ Page ... directive) to import the right namespaces:

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<%@ Import Namespace="System.Data.OleDb" %>
```

If you have put the above function in your code-behind class, you will need a couple of lines at the top to import the right namespaces:

C#:

```
using System.Data;
using System.Data.SqlClient;
using System.Data.OleDb;
```

VB.NET:

```
Imports System.Data
Imports System.Data.SqlClient
Imports System.Data.OleDb
```

4) Ensure that the Cambro.Web.DbCombo.dll file is in your bin directory, and if it isn't, create a Reference.

5) Navigate to the page in a browser, and you should see the DbCombo control.

Intellisense

To use HTML Intellisense, add the following attribute to an html tag that contains the DbCombo control (html, body, form, span etc.).

```
XMLNS:DbCombo="http://schemas.cambro.net/dbcombo"
```

Reference

The syntax for the ASPX WebControl is below:

```
<%@ Register TagPrefix="DbCombo" Namespace="Cambro.Web.DbCombo"
            Assembly="Cambro.Web.DbCombo" %>

<DbCombo:DbCombo Runat="server" ID="id" RegistrationKey="..." />
```

All the properties are optional except the standard `Runat` and `ID`. The `RegistrationKey` property should only be omitted if testing on the localhost domain.

Basic Properties

The properties below are used in the regular, day-to-day use of DbCombo.

Runat and ID [Pro, Lite, Free]

These are standard properties, required on all web controls.

string RegistrationKey [Pro, Lite, Free]

This is the registration key. If left blank, a default registration key is used that limits the use to the localhost domain. Your registration key is always available in the "My Purchases" page of www.dbcombo.net.

string ServerMethod [Pro, Lite, Free]

This string defines the method that is used to return results. There is more info about ServerMethod later in this document.

string Value [Pro, Lite, Free]

This is the value that is currently selected. The selected value is changed by selecting an item from the drop-down. You may set this in the ASPX page and access it at PostBack in your server code.

string Text [Pro, Lite, Free]

This is the text that is currently entered in the text box. You may set this in the ASPX page and access it at PostBack in your server code.

int TextBoxColumns [Pro, Lite, Free]

This changes the number of columns in the text entry text box. Default is 30.

int DropDownRows [Pro, Lite, Free]

This changes the number of rows visible in the selection drop-down. It also determines the number of rows that are retrieved initially when a new query is entered. Default is 10.

bool ForceDownLevel [Pro, Lite, Free]

This parameter, when set to true, forces the control in to down-level-browser mode. The default value is false.

bool Debug [Pro, Lite, Free]

This parameter is useful in debugging the server side method. When set to true, the raw results from the server method are displayed in a pop-up browser window. Default is false.

Intermediate properties, events and methods

These properties will be used in special circumstances.

string ServerType and string ServerAssembly [Pro]

DbCombo looks in the current aspx page and in it's codebehind for your ServerMethod. If your method is not in either of these places, you can specify where your ServerMethod is using the ServerType and ServerNamespace. ServerType should be the full type name, including the namespace and class name.
ServerAssembly is only needed if your ServerMethod is not in the same assembly as your codebehind.

string DataValueField / DataTextField [Pro]

This properties are names of the field in the dataset returned by the ServerMethod that will be used for the Value and Text properties of DbCombo. The default value of an empty string causes DbCombo to use the default of "DbComboValue" and "DbComboText".

Note: If you specify values for either of these properties, you must change the FieldSecurity property in your ServerMethod – e.g.:

```
[Cambro.Web.DbCombo.ResultsMethod(true)]
public static object DbComboMethod(
     Cambro.Web.DbCombo.ServerMethodArgs args)
{
     args.FieldSecurity.Value = FieldSecurity.AllFields;
     ...
```

The default value of FieldSecurity will only allow the default "DbComboValue" and "DbComboText" fields to be returned. This field is necessary because the request comes from an untrusted source.

string DataMember [Pro]

Use the DataMember property to specify a member from a multimember data source. For example, if you have a data source with more than one table, use the DataMember property to specify which table to bind to. Leave blank for the default. Remember if you change this from its default, you MUST change the ServerMethodEventArgs.DataMemberSecurity property in your ServerMethod.

Note: If you specify a value for this property, you must change the DataMemberSecurity property in your ServerMethod – e.g.:

```
[Cambro.Web.DbCombo.ResultsMethod(true)]
public static object DbComboMethod(
     Cambro.Web.DbCombo.ServerMethodArgs args)
{
     args.DataMemberSecurity.Value=DataMemberSecurity.AllDataMembers;
     ...
```

The default value of DataMemberSecurity will only allow the default DataMember to be returned. This is necessary because the request comes from an untrusted source.

string ServerDir [Pro, Lite, Free]

The server file – DbComboServer.aspx is assumed to be in the same directory as the control unless this parameter is set.

If the support file is in the root directory of the web site, use "/".
To specify the support directory relative to the **root directory** of the web site, use something like: "/myDir/myOtherDir/"
To specify the support directory relative to the **current directory** of the web site, use something like: "myOtherDir/"

A trailing '/' is added if not specified.

bool HideIntersectingSelectTags [Pro]

Select tags render on top of all objects in IE, sometimes causing the more button or status message to be obscured.
Setting this to true will temporarily hide all select tags that would intersect with DbCombo when the drop-down opens. The select tags are re-displayed when the drop-down is hidden.
This function is disabled by default because the process of hiding the SELECT tags may reduce performance on complex pages.

bool HideAllSelectTags [Pro]

Select tags render on top of all objects in IE, sometimes causing the more button or status message to be obscured.
Setting this to true will temporarily hide all select tags on the page when the drop-down opens. The select tags are re-displayed when the drop-down is hidden.
This provides far better performance than HideIntersectingSelectTags especially on large pages, because it doesn't have to evaluate the position of each SELECT tag.

bool AutoPostBack [Pro]

If this is set to true, the control will postback when either an item is clicked on, or enter is pressed on an item. Default is false.

bool ReQueryOnLoad [Pro]

This property determines if DbCombo will perform a re-query the next time it loads. It is initially set to false, but if a query is attempted and the page is posted back, it will become true. Set to true initially to force a re-query. (default: false).

bool ReQueryDisabled [Pro]

If this is set to true, the control will not perform a re-query on load. If set to false, the control will perform a re-query if necessary. Default is false.

string ReQueryText, int ReQueryRecords [Pro]

Upon postback, DbCombo uses this property to determine the state of the DbCombo before postback. The text of the actual query (what was last queried on, not what is in the Text property) is used for a re-query.
You can force a re-query on page load (without postback) by entering a value for this property, and the ReQueryRecords property.

Please note the value of ReQueryRecords must conform to:
`(2^n-1) x DropDownRows` (with n being any +ve integer)

int MaxLength [Pro]

This sets the maximum length of the text-box - if 0, then unlimited. Default is 0. Note: this only affects text input. If an item longer than the MaxLength is selected, it will be displayed in full.

short TabIndex [Pro]

This sets the tab index of the text box. The tab index of the up level search button is always set to -1. Tab indexes of other buttons follow in sequence from the main tab index. Defaults is 1.

int Latency [Pro]

This is the number of milliseconds that DbCombo waits after the last keypress before making a query. The default of 300 ensures that a query is not fired on every keypress, when the user types several characters. A lower latency figures may be useful when server load is not important, and when internet latency is minimal (e.g. intranet usage).
When Latency is set to -1, automatic searching on key presses is disabled.

event SelectedItemChanged (OnSelectedItemChanged) [Pro]

This event fires on the server when the selected item has changed since the last postback.

void Reset() [Pro]

This method resets the DbCombo. DbCombo will not perform a requery after this method is called.

bool SelectSingleItemOnEnter [Pro]

If there is only one item in the results and enter is pressed, this item will be selected if this property is set to true. (Default: true).

bool SelectSingleItemOnTab [Pro]

If there is only one item in the results and tab is pressed, this item will be selected if this property is set to true. (Default: true).

bool TabToNextFieldOnEnter [Pro]

When enter is pressed, DbCombo will simulate tab being pressed - e.g. skip to the next field. (Default: false).

ValidationProperties ValidationProperty [Pro]

DbCombo supports the built-in asp.net validation controls by exposing either the Text or Value property to them.
Possible values are:

- ValidationProperties.Text
- ValidationProperties.Value

Choosing the Text property will validate on the text entered in the text box.
Choosing the Value property (default) validates on the hidden, Value property.
This is useful because if an item from the results is not selected, the Value property will always be empty. A RequiredFieldValidator can used to ensure an item is selected.
Default is ValidationProperties.Value

ErrorBoxTypes ErrorBoxType [Pro]

This allows you to specify the way that DbCombo handles errors on the server.
Values are:

- ErrorBoxTypes.Auto
  If an error occurs on the server, DbCombo will display a pop-up dialogue detailing the error.
- ErrorBoxTypes.None
  If an error occurs on the server, DbCombo will ignore the error.
- ErrorBoxTypes.Custom
  If an error occurs on the server, DbCombo will display a pop-up dialogue with custom text in it. Adjust the CustomErrorText property to set the error text.

string ErrorBoxCustomText [Pro]

If ErrorBoxType = Custom, this string is used in the pop-up dialogue for all errors.

Advanced properties

These properties are not meant to be used in usual circumstances – care and extended testing should be employed when implementing them.

bool CloseResultsOnBlur [Pro]

If this is set to false, the results stay visible when the DbCombo looses focus. Default is true.

bool CloseResultsOnClick [Pro]

If this is set to false, the results stay visible when an item is clicked on. Default is true.

bool CloseResultsOnEnter [Pro]

If this is set to false, the results stay visible when the enter key is pressed. Default is true.

bool CloseResultsOnTab [Pro]

If this is set to false, the results stay visible when the tab key is pressed. Default is true.

bool ClearQueryOnUpLevelSearchButton [Pro]

If this is set to true, clicking on the UpLevel search button will cause the contents of the text box to be cleared before the query is executed.

int ResultsSpanTweakX, ResultsSpanTweakY [Pro]

The results span can be misaligned by an IE browser issue. Tweak these variables until the corner of the results span is aligned with that of the text box. *The alignment bug has been fixed, but I have left these properties active, as they may be useful in certain circumstances.*

boool InvertArrow [Pro, Lite, Free]

The Visual Studio.NET 2003 built-in browser fires the blur event of the DbCombo textbox when the image on the up-level search button is changed (from a down arrow to an up arrow). By default DbCombo will cancel the current query if this happens. If you specifically need DbCombo to work inside Visual Studio.NET 2003, set this to false. The arrow image on the button will not change while the results are open. Default is true.

Appearance properties

These properties may be used to alter and customise the appearance of DbCombo.

bool ShowUpLevelHelpButton [Pro]

This shows or hides the up-level help button. Default is false.

bool ShowDownLevelHelpButton [Pro]

This shows or hides the down-level help button. Default is true.

bool ShowUpLevelSearchButton [Pro]

This hides or displays the up-level search button. Default is true.

bool ShowDbComboLink [Pro, Lite]

This hides or displays the DbCombo logo icon and link to the DbCombo site. Default is true.

Styles [Pro]

There are several html elements may have their style and class attributes set. The table below details which controls are affected by which properties:

string UpLevelButtonStyle string UpLevelButtonClass	All the up-level buttons in the control (Except the 'More...' button in the status panel.)
string UpLevelMoreResultsButtonStyle string UpLevelMoreResultsButtonClass	The 'More...' button in the status panel.
string DownLevelButtonStyle string DownLevelButtonClass	All the down-level buttons in the control
string TextBoxStyle string TextBoxClass	The text box
string ResultsBackgroundSpanStyle string ResultsBackgroundSpanClass	The background of the results in up-level mode. This span encompasses the whole results drop-down, including the status bar and more-results button bar.
string ResultsInnerSpanStyle string ResultsInnerSpanClass	The background of the results in up-level mode. This span only contains the list-box (not the status bar, or more-button bar).
string ResultsSelectBoxStyle string ResultsSelectBoxClass	The list-box.
string StatusMessageStyle string StatusMessageClass	The status messages (loading, no results etc.)
string MoreResultsBarStyle string MoreResultsBarClass	This is the bar containing the more-results button.
string MainSpanStyle string MainSpanClass	The main <span> tag that surrounds the whole control.

Text elements [Pro]

All the textual elements can be modified for customisation or internationalisation:

string TextClearButton	This is the text that appears in the down-level ‘Clear’ button.
string TextDownLevelSearchButton	This is the text that appears in the down-level ‘Search’ button.
string TextUpLevelSearchButton	This text appears in the up-level search button if the button is in *Text* mode. (The button is in *Graphic* mode by default).
string TextLoading	The text that appears in the status message while DbCombo is loading more results.
string TextNoResults	The text that appears when there are no results to display (Default: "No results.")
string TextMoreButton	The text that appears on the 'More' button (Default: "More...")
string TextSelectButton	The text that appears on the 'Select' button (DownLevel only) (Default: "Select")
string TextHelpButton	The text that appears on the help button (Default: " ? ")
string TextUpLevelHelpBox	The text that appears in the UpLevel help box. Note - use "\n" for a new line, and remember to escape certain characters - e.g. """ must be written as "\"".
string TextDownLevelHelpBox	The text that appears in the DownLevel help box. Note - use "\n" for a new line, and remember to escape certain characters - e.g. """ must be written as "\"".
string TextAlt	The text that appears in the ALT attribute of the textbox. This is useful for screen-readers.

State functions

These properties are concerned with communicating page or server state to the ServerMethod.

string ClientStateFunction [Pro]

This client function enables you to pass information about the pages state back to your ServerMethod. This enables you to provide different results dependant on other form elements (e.g. drop-downs etc.)
IMPORTANT – you **must include the parenthesis** after the name of your function. You may optionally include parameters to pass to the function. This function must return a javascript 'Object' object. – e.g.:

```
function MyStateFunction()
{
     var state = new Object();
     state["key1"]="value1";
     state["another_key"]="another_value";
     return state;
}
```

This object is converted to a .NET 'Hashtable' object, and passed to your ServerMethod server function as the clientState parameter.

ADVANCED NOTE: You can inform DbCombo of a change to the page state by issuing the following javascript code:

```
if (typeof(DbComboServerExists)!='undefined')
    DbComboStateChanged('Combo1');
```

With 'Combo1' being the unique id of the DbCombo control.

event GetDownLevelState / OnGetDownLevelState [Pro]

This server function enables you to pass state in down-level browser mode. The function takes one parameter of type DbCombo (the issuing DbCombo object), and returns a Hashtable object. You should aim to replicate your javascript functionality, e.g.:

```
protected Hashtable MyDownLevelStateFunction()
{
     Hashtable state = new Hashtable();
     state["key1"]="value1";
     state["another_key"]="another_value";
     return state;
}
```

As this function is not static, you may access the Page object, and postback data.

Hashtable ServerState, string ServerStateSecretString [Pro]

This enables data to be securely transferred from server code to the

ServerMethod function. In your server code, set this to a Hashtable. You can use any serializable types as the keys and data. DbCombo serialises this Hashtable, and using this string and a secret key, creates a MD5 hash. This is round-tripped with each DbCombo request. The ServerMethod can verify the authenticity of the data in the Hashtable by repeating the HD5 hash and comparing the results with the client submitted hash. This process is secure so long as the secret key remains secret, and is random. You must provide your own secret key by setting the ServerStateSecretString property to a random string of your choice.
The Hashtable is presented to the ServerMethod as a SecureHashtable (Cambro.Web.DbCombo.SecureHashtable). This derives from Hashtable, and adds the Authenticate method. You MUST call this method with your secret string as a parameter to ensure that the ServerState is valid. A code sample is provided below:

```
protected void Page_Load(object o, EventArgs e)
{
    Hashtable hash = new Hashtable();
    hash.Add("UserName", HttpContext.Current.User.Identity.Name);
    Combo1.ServerState = hash;
    Combo1.ServerStateSecretString=
        "kdfsjgnskajh[apq2-4uhg465654435879mnbfdm cxs;q4";
}

[Cambro.Web.DbCombo.ResultsMethodAttribute(true)]
public static object
DbComboMethod(Cambro.Web.DbCombo.ServerMethodArgs args)
{
    string extraWhereClause = "";
    if (
        args.ServerState!=null &&
        args.ServerState.Authenticate(
            "kdfsjgnskajh[apq2-4uhg465654435879mnbfdm cxs;q4"
        ) &&
        args.ServerState["UserName"]!=null &&
        args.ServerState["UserName"].ToString().Length > 0
    )
    {
        extraWhereClause +=
            " AND ClientUserName = '" +

args.ServerState["UserName"].ToString().Replace("'","''") + "' ";
    }
    else
        throw new Exception("ServerState is coorrupt.");

    DataSet dataset=new DataSet();
    SqlConnection conn = new SqlConnection("[your-connection]");
    SqlDataAdapter adapter = new SqlDataAdapter();
    adapter.SelectCommand = new SqlCommand(@"
        SELECT TOP "+args.Top+@"
            ClientName AS DbComboText,
            ClientID AS DbComboValue
        FROM Orders
        WHERE
            ClientName LIKE @Query
            "+extraWhereClause+@"
        ORDER BY ProductName", conn);
    adapter.SelectCommand.Parameters.Add("@Query", args.Query+"%");
    adapter.Fill(dataset);
    conn.Close();
    return dataset;
}
```

Client functions

This section is concerned with interacting with Javascript on the client.

string ClientOnSelectFunction [Pro]

If specified, this javascript function is executed each time a DbCombo item is selected in the drop-down.
IMPORTANT – you **must NOT include the parenthesis** after the name of your function. Your function must accept two strings as parameters. The first is DbCombo's Value property; the second is the Text property. If we are **unselecting** from DbCombo, the value property is a blank string, and the Text property is set to whatever the textbox contains.

```
function OnSelectEmployee(Value, Text, SelectionType)
{
     if (Value!="")
     {
             document.all["myText"].value = Text;
             document.all["myValue"].value = Value;
     }
     else
     {
             document.all["myText"].value = "Not selected";
             document.all["myValue"].value = "Not selected";
     }
}
```

The SelectionType parameter shows how the event was generated – please see the table below:

SelectionType	Event
1	The user is scrolling down the list ClinetOnSelectFunction fires each time a new item is selected.
2	The user has pressed enter when an item is selected. The results disappear, and the item is selected.
3	The user has clicked on an item. The results disappear, and the item is selected.
4	The user has entered more text into the textbox. Another query will be run, and the selection is cleared. Nothing is selected.
5	The user has scrolled off the bottom or top of the drop-down, and cleared the current selection. Nothing is selected.
6	The user has pressed escape to clear the DbCombo, and hide the results. Nothing is selected.
7	DbCombo has evaluated the client state function, and found it to have changed. The current selection is cleared. Nothing is selected.
8	The user has entered an exact match to one of the data items. This item is selected.
9	In a 're-query on postback' or an 'auto-query on load', both the text and value properties have matched with one of the items. This item is selected.
10	The selection has been made by the DbComboSelectByValue JavaScript support function.
11	The selection has been made by the DbComboSelectByText JavaScript support function.

ServerMethod details

We’ve tried to make it as easy as possible to code DbCombo server query methods. The signature of the method must be:

```
[Cambro.Web.DbCombo.ResultsMethodAttribute(true)]
public static object xxx(Cambro.Web.DbCombo.ServerMethodArgs args){}
```

It’s important to remember that the custom attribute is always included. This is an essential security measure designed to ensure that only these methods can be executed by a web client.

The parameter *Cambro.Web.DbCombo.ServerMethodArgs* has the following properties.

ServerMethodArgs properties

string Query

This is the text that has been entered into the text box. You are free to use this in any way you want to produce results.

int Top

This specifies how many rows are required. Only a certain number of rows are transmitted to the client to enhance performance, so there is no need to shuttle extra results around internally.

Hashtable ClientState

This Hashtable is generated from the output of the client state function. If no client state function has been specified, or it returns an empty Object, this parameter will be null.

Cambro.Web.DbCombo.SecureHashtable ServerState

This SecureHashtable is generated from the server state. If no server state has been specified, this parameter will be null. See ‘ServerState’ in the reference section for an example.

bool UpLevel

This enables you to tell if the client is in up-level or down-level mode. If the browser is in up-level mode, this is true. If the browser is in down-level mode, this is false.

FieldSecurityClass FieldSecurity, ArrayList FieldSubset

The FieldSecurity.Value property holds the value of the FieldSecurity setting. If you specify values for either the DataValueField or DataTextField property, you must change the FieldSecurity setting. The default value of FieldSecurity will only allow the default “DbComboValue” and “DbComboText” fields to be returned. This field is necessary because the request comes from an untrusted source.

FieldSecurity can be set to:

FieldSecurity.Default
Most secure - only the default "DbComboValue" and "DbComboText" fields are available to the client.

FieldSecurity.IncludeFieldSubset
Only fields contained in the field subset are available to the client. Each field in the subset should be added to the ArrayList FieldSubset – e.g.:

```
[Cambro.Web.DbCombo.ResultsMethod(true)]
public static object DbComboMethod(
     Cambro.Web.DbCombo.ServerMethodArgs args)
{
     args.FieldSecurity.Value = FieldSecurity.IncludeFieldSubset;
     args.FieldSubset.Add("CustomerID");
     args.FieldSubset.Add("CustomerName");
     args.FieldSubset.Add("CustomerEmail");
     ...
```

FieldSecurity.ExcludeFieldSubset
All fields are available to the client EXCEPT those contained in the field subset. Each field in the subset should be added to the ArrayList FieldSubset – e.g.:

```
[Cambro.Web.DbCombo.ResultsMethod(true)]
public static object DbComboMethod(
     Cambro.Web.DbCombo.ServerMethodArgs args)
{
     args.FieldSecurity.Value = FieldSecurity.ExcludeFieldSubset;
     args.FieldSubset.Add("CustomerPassword");
     args.FieldSubset.Add("CustomerSalary");
     ...
```

FieldSecurity.AllFields
AllFields - least secure - all fields are available to the client.

DataMemberSecurityClass DataMemberSecurity, ArrayList DataMemberSubset

The DataMemberSecurity.Value property holds the value of the DataMemberSecurity setting. If you specify a value for the DataMember property, you must change the DataMemberSecurity setting. The default value of DataMemberSecurity will only allow the default DataMember to be returned. This is necessary because the request comes from an untrusted source.

DataMemberSecurity can be set to:

DataMemberSecurity.Default
Most secure - only the default DataMember is available to the client.

DataMemberSecurity.IncludeDataMemberSubset
Only DataMembers contained in the DataMember subset are available to the client. Each DataMember in the subset should be added to the ArrayList DataMemberSubset – e.g.:

```
[Cambro.Web.DbCombo.ResultsMethod(true)]
public static object DbComboMethod(
     Cambro.Web.DbCombo.ServerMethodArgs args)
```

```
{
     args.DataMemberSecurity.Value =
DataMemberSecurity.IncludeDataMemberSubset;
     args.DataMemberSubset.Add("PublicCustomerDetails");
     args.DataMemberSubset.Add("PublicEmployeeDetails");
     args.DataMemberSubset.Add("PhoneDirectory");
     ...
```

DataMemberSecurity.ExcludeFieldSubset
All DataMembers are available to the client EXCEPT those contained in the DataMember subset. Each DataMember in the subset should be added to the ArrayList DataMemberSubset – e.g.:

```
[Cambro.Web.DbCombo.ResultsMethod(true)]
public static object DbComboMethod(
     Cambro.Web.DbCombo.ServerMethodArgs args)
{
     args.DataMemberSecurity.Value =
DataMemberSecurity.ExcludeDataMemberSubset;
     args.DataMemberSubset.Add("PrivateCustomerDetails");
     args.DataMemberSubset.Add("EmployeeDetails");
     ...
```

DataMemberSecurity.AllDataMembers
AllDataMembers - least secure - all DataMembers are available to the client.

Return object

The return value is an object, and can be one of many types. Examples of supported types are:

- DataSet
- DataView
- SqlDataReader
- OleDbDataReader
- Collections of business ojects

I recommend a DataSet – nice and simple.

Important: Unless you specify values for DbCombo.DataValueField / DataTextField, the Value is extracted from a column called 'DbComboValue', and the Text is extracted from a column called 'DbComboText'. Your SQL query must therefore be something like:

```
SELECT Id AS DbComboValue, Name AS DbComboText FROM MyTable WHERE
...
```

Important note for DataReader users

Note: If using a SqlDataReader or OleDbDataReader, ensure you set CommandBehavior.CloseConnection when you call ExecuteReader(). This will ensure that the connection is closed after DbCombo finishes with it.
e.g.:

```
SqlConnection conn = new SqlConnection(connStr);
SqlCommand myCommand = new SqlCommand(sqlStr, conn);
myCommand.Connection.Open();
return myCommand.ExecuteReader(CommandBehavior.CloseConnection );
```

ServerMethod technical details

99% of the time you won’t need to know the following, but it is here for completeness.

You are not actually limited to the aforementioned types. When extracting results from a query object, the logic goes:

```
if the object is a DataSet, use the required table as specified
by DataMember

if the object can be reduced to an IList...
    ...move through it’s records using the IList interface
if the object is IEnumerable
    ...move through the records using it’s IEnumerator

    for each item
    {
        if the item is a DataRowView, IDataRecord or IDictionary
            use the values of the required columns specified
            by DataTextField and DataValueField
        else if the item is a string or primative value type
            use it’s string reprasentation
        else
            try to use reflection to determine the value of fields
            or properties with the names specified by DataTextField
            and DataValueField
    }
}
```

In this way you may also use an array of arrays, an array of dictionary objects, or any other construct that supports the above framework.

A simple construct that I recommend for testing is:

```
return new string[] {
     "your value1",
     "your value2",
     "your value3"
};
```

This will return the above values for both the Text and Value elements of the items. This does not require database access.
If you require the Text and Value properties to be different, a simple class is included for this purpose. Use the following syntax:

```
return new Cambro.Web.DbCombo.SimpleResult(
     "DbComboText",new string[]{"Text1","Text2","Text3"},
     "DbComboValue",new string[]{"Value1","Value2","Value3"}
);
```

You can replace the strings DbComboText and DbComboValue with your own column names is needed.

Note for non utf-8 encoding

Some users may experience problems when using non utf-8 content encoding settings. A work-around is explained below.

Create a subdirectory in your web application called 'utf-8'

Create a web.config file in that directory. It’s contents should be:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <system.web>
        <globalization
            requestEncoding="utf-8"
            responseEncoding="utf-8" />
    </system.web>
</configuration>
```

Move your DbComboServer.aspx into this directory.

Add the following to your DbCombo tags:

ServerDir="utf-8/"

JavaScript support functions

DbCombo (in up-level mode) has an extensive library of JavaScript functions, some of which are useful for interacting with DbCombo.

Please note that the JavaScript support is not intended to be accessed by script on your page, and thus is not covered by our support contract. It is only included here for advanced users that are happy to spend the time experimenting.

These JavaScript functions are global to the page, and may be called from your script.
They all take a standard parameter (designated 'c' below). This is a string representing the UniqueID of the DbCombo tag. In your script it is best to insert this with a databound block: <%#MyDbCombo.UniqueID%>. Note that you must execute the Page.DataBind() method to emit the code for these blocks.

bool DbComboServerExists

This global JavaScript variable is set to true when the JavaScript support file loads. Check the type of this variable (as demonstrated below) before trying to access any of the functions. If DbCombo is in down-level mode (e.g. on a Netscape platform), the JavaScript support file will not load, but your code may still execute and fail.

```
if (typeof(DbComboServerExists)!='undefined')
{
     [safe to access DbCombo Javascript support functions here]
}
```

string DbComboGetText(c)

This gets the text currently in the DbCombo text box.

string DbComboGetValue(c)

This gets the value of the item currently selected.

string DbComboGetQuery(c)

This gets the query that was last used to search. This may be different to the text in the textbox, because the text changes as items are selected; however the query only changes when another search is performed.

void DbComboStateChanged(c)

This informs the DbCombo that its client state may have changed. The DbCombo will re-evaluate its client state and if it is indeed found to have changed, it will re-query the database.

void DbComboSendRequestButton(c)

This function is fired when the up-level search button is clicked. It will perform a search or show the hidden results.

void DbComboClear(c)

This function completely clears a DbCombo, and resets all properties.

bool DbComboHiddenSearchAvailable(c)

This determines if a DbCombo has a result-set hidden.

void DbComboShowResults(c, bool resize)

This shows the results pane, including drop-down. The resize parameter determines whether DbCombo should resize the drop-down and associated elements. Ensure you check that there is a result-set hidden (with DbComboHiddenSearchAvailable) before you call this function.

void DbComboHideResults(c)

This will hide the results of a DbCombo.

void DbComboSetCaretToEnd(c)

This sets the caret (text cursor) to the end of the text-box of a DbCombo.

void DbComboSelectAll(c)

This selects all the text of a DbCombo.

void DbComboGetMoreResults(c)

This gets more results, equivalent to clicking the More button.

void DbComboChangeText(c, string text)

This function changes the text in the DbCombo, and initiates a new query.

bool DbComboSelectByValue(c, string value)

This function attempts to select an item from the DbCombo result set based on its Value property. This function returns true if the item has been found and false if the item was not found.

bool DbComboSelectByText(c, string text)

This function attempts to select an item from the DbCombo result set based on its Text property. This function returns true if the item has been found and false if the item was not found.