Web Developer's Core Component Library
Version 1.1

ASP Programmer's Manual

BreZaSoft, LLC

License Agreement

You should carefully read the following terms and conditions before using the Web Developer’s Core Component Library software. Your use of the Web Developer’s Core Component Library software indicates your acceptance of this license agreement. Do not use the Web Developer’s Core Component Library software if you do not agree with this license agreement.

Disclaimer of Warranty

The Web Developer’s Core Component Library software and the accompanying files are distributed and licensed “as is”. BreZaSoft, LLC disclaims all warranties, either express or implied, including, but not limited to implied warranties of merchantability and fitness for a particular purpose. Should the Web Developer’s Core Component Library software prove defective, the licensee assumes the risk of paying the entire cost of all necessary servicing, repair, or correction and any incidental or consequential damages. In no event will BreZaSoft, LLC be liable for any damages whatsoever (including without limitation damages for loss of business profits, business interruption, loss of business information and the like) arising out of the use or the inability to use the Web Developer’s Core Component Library software even if BreZaSoft, LLC has been advised of the possibility of such damages.

Copyright

The Web Developer’s Core Component Library software is protected by copyright, as well as other intellectual property laws and treaties. The Web Developer’s Core Component Library software is licensed not sold. Title to the Web Developer’s Core Component Library software shall at all times remain with BreZaSoft, LLC.

Demo Version

The demo version of the Web Developer’s Core Component Library software will produce acknowledgement messages in the output generated by it.

Subjected to the conditions in this license agreement:

- You may use the unmodified demo version of the Web Developer’s Core Component Library software without charge.
- You may redistribute the unmodified demo version of the Web Developer’s Core Component Library software, provided you do not charge for it.
- You may embed the unmodified demo version of the Web Developer’s Core Component Library software (or part of it), in a product and distribute the product, provided you do not charge for the product.

If you do not want the acknowledgement messages appearing in the output, or you want to embed the Web Developer's Core Component Library software (or part of it) in a product that is not free, you must purchase a commercial license to use the Component Library software from BreZaSoft, LLC. Please refer to BreZaSoft's web site at http://www.BreZaSoft.com for details.

Ordering Information

Please refer to the BreZaSoft, LLC Web site at http://www.BreZaSoft.com

Support Contact

Please refer to the BreZaSoft, LLC Web site at http://www.BreZaSoft.com

Table of Contents

Table of Figures

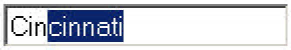
1

Introduction

Welcome to BreZaSoft's Web Developer's Core Component Library

BreZaSoft's Web Developer's Core Component Library classes are compiled COM objects. The ASP developer instantiates an instance of a Core Component class from their web page via a call to CreateObject. From there, they populate different properties and call a method that generates HTML/JavaScript code. The output generated by the Core Component Library is flexible, and provides a dynamic, professional look to your Web Applications. The Web Developer's Core Component Library consists of the following classes:

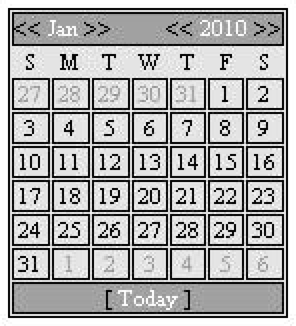
- AutoComplete Text Box (BZSCoreComp.AutoComplete)



Cincinnati

Figure 1 - Introduction to AutoComplete Output

- Embedded and Popup Calendar (BZSCoreComp.Calendar)

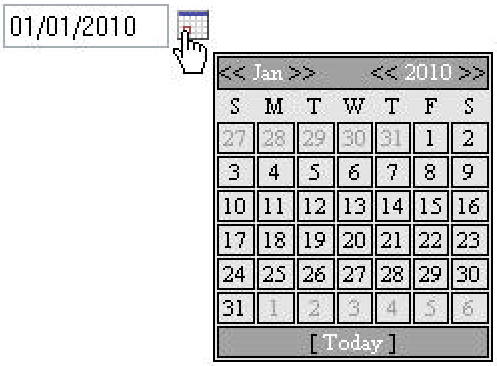


<< Jan >> << 2010 >>

S	M	T	W	T	F	S
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

[Today]

Figure 2 - Introduction to Embedded Calendar Output



01/01/2010

S	M	T	W	T	F	S
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

Figure 3 - Introduction to Popup Calendar Output

- Right-Click Popup Menu (BZSCoreComp.PopupMenu)

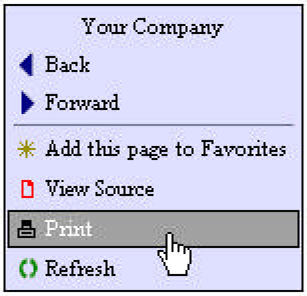


Figure 4 - Introduction to Right-Click Popup Menu Output

- Real Time Progress Bar (BZSCoreComp.ProgressBar)

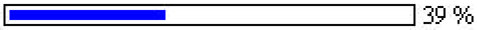


Figure 5 - Introduction to Progress Bar Output

- Tabbed Panels (BZSCoreComp.Tabs)

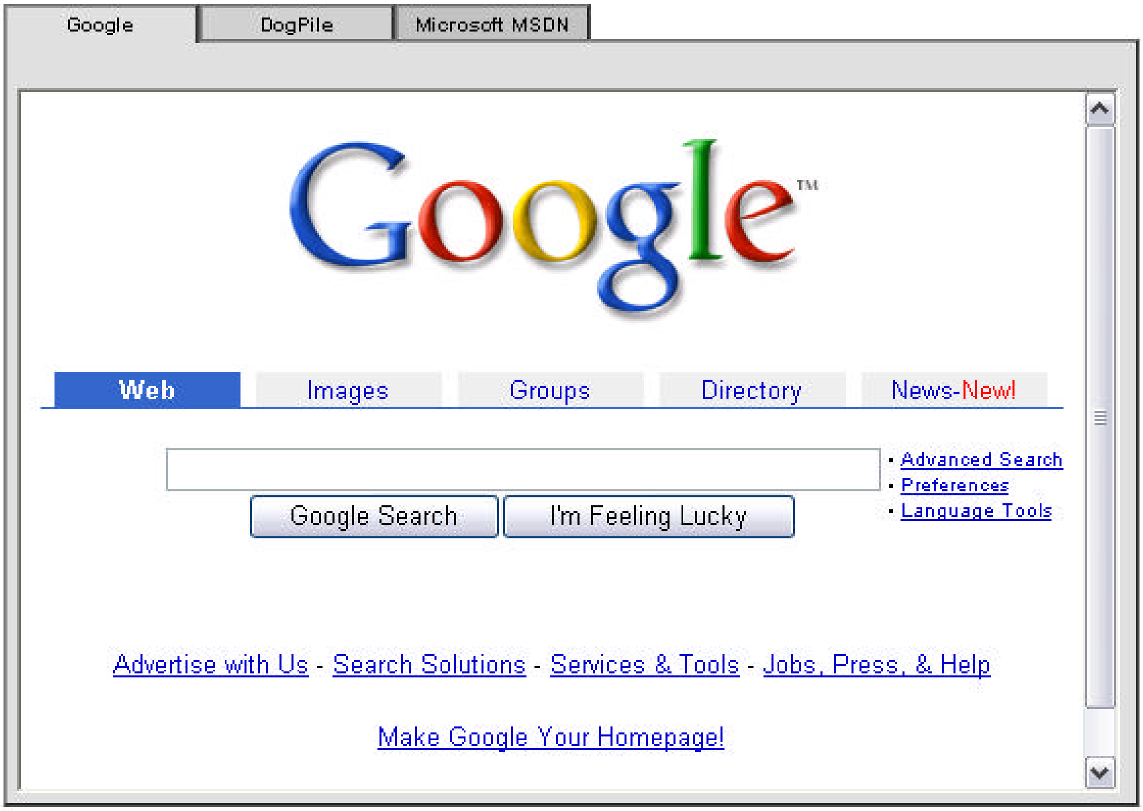
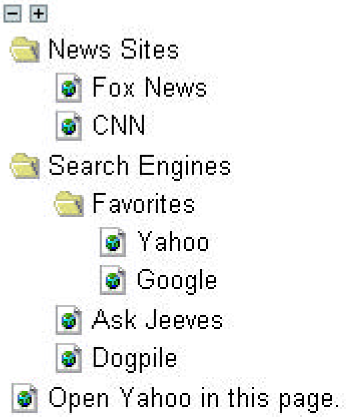


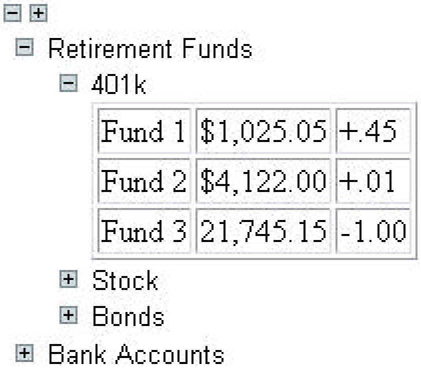
Figure 6 - Introduction to Tabs Output

- Expandable / Collapsible Tree



- News Sites
  - Fox News
  - CNN
- Search Engines
  - Favorites
    - Yahoo
    - Google
  - Ask Jeeves
  - Dogpile
- Open Yahoo in this page.

Figure 7 - Introduction to Tree Output (1)



- Retirement Funds
  - 401k

Fund 1	$1,025.05	+.45
Fund 2	$4,122.00	+.01
Fund 3	21,745.15	-1.00

  - Stock
  - Bonds
- Bank Accounts

Figure 8 - Introduction to Tree Output (2)

Supported Platforms

The Core Component Library is supported on Windows2000/NT/XP/ME/98. Output of the Web Developer's Core Component Library is certified on Microsoft's Internet Explorer 5.5+.

2

Installation

Setup

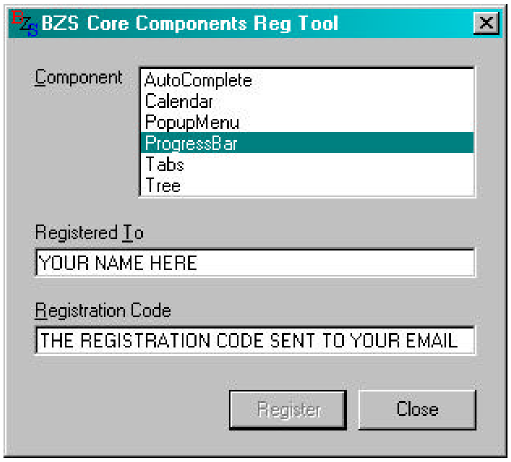
To install the Core Component Library, simply run the accompanying setup executable (setup.exe) file. This will launch the installer and guide you through the rest of the process.

Sample Projects

The Core component Library comes with a number of Sample Projects. These projects serve as good tutorials on how to use the Core Web Developer's Core Component Library software.

Installing the Component Library License

The Core Component Library is packaged with a License Installer which you can use to install your License Code. You do not need to install a license if you intend on using the Web Developer's Core Component Library software as a demo version only.



Component: ProgressBar

Registered To: YOUR NAME HERE

Registration Code: THE REGISTRATION CODE SENT TO YOUR EMAIL

Figure 9 - Registration Application

The Encrypted License Code will be emailed to you after purchasing a license from http://www.BreZaSoft.com

Testing the Web Developer’s Core Component Library Installation

The easiest way to verify that the Core Component Library was setup correctly is to simply run one of the Sample Projects packaged with the Library. If the sample runs correctly, it can be confirmed that the Core Component Library was installed successfully.

Troubleshooting the Web Developer’s Core Component Library Installation

When testing your installation of the Web Developer’s Core Component Library, please take the following into consideration:

- Ensure that the Windows account that IIS is running under has sufficient privileges to access the file that houses the Web Developer’s Core Component Library Classes (BZSCoreComp.dll)
- If you want the “Demo” acknowledgement not to appear in any/all output generated by the components, please register your license key with the “BZS Core Components Reg Tool,” which is installed along with the Web Developer’s Core Component Library.
- When entering information into the “BZS Core Components Reg Tool,” please ensure that you are typing the information exactly as it appears in your license documentation.

3

Coding with the AutoComplete Text Box

Overview

This class implements the AutoComplete Textbox. This functionality is common in Win32 applications, but is difficult to reproduce in a Web Browser. The AutoComplete class allows the ASP developer to link a common HTML <SELECT> List to a free form text box, which automatically finds matches within the linked <SELECT> List, and anticipates the completion of the text. The linked <SELECT> List can be displayed with the AutoComplete Text Box, or hidden to expose only the Auto Complete Text Box to the user. There is a simple and advanced sample for the Auto Complete Text Box. The code implementation below is the simple implementation.

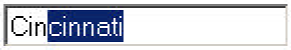
Code Implementation

```
<%@ Language=VBScript %>
<%
Option Explicit
Response.ExpiresAbsolute = 0

Dim AC
Set AC = CreateObject("BZSCoreComp.AutoComplete")

%>
<form>
<select name=selAutoComplete1 style="display:none">
      <option value=1>Cincinnati</option>
      <option value=2>New York</option>
      <option value=3>Los Angeles</option>
      <option value=4>Chicago</option>
</select>
<%

Response.Write AC.EmbedAutoComplete("AutoComplete1",
"selAutoComplete1", True, True)

Set AC = Nothing
%>
</form>
```

Figure 10 - Code Implementation for AutoComplete Text Box

Output

The output below displays how the Auto Complete Text Box anticipates the entered text from the user.



Cincinnati

Figure 11 - AutoComplete Textbox Output

Code Breakout and Explanation

- **Set AC = CreateObject("BZSCoreComp.AutoComplete")**
    - The Auto Complete Text Box is encapsulated in a COM object called “BZSCoreComp.AutoComplete”. This line simply creates an instance of the object so you can use the Auto Complete Text Box

- **<form>**
    - The Auto Complete Text Box Object will render and stream client side code to the browser. The code it creates must reside within <form> Tags.

- **<select name=selAutoComplete1 style="display:none">**
    - This is the HTML <SELECT> List that you will be linking the Auto Complete Text Box to. Here, we’ve used a style tag to hide the <SELECT> list from view. You do not have to hide the <SELECT> list if you wish to have it displayed.

- **<option value=1>Cincinnati</option>**
    - This is the first of four linked options in this example. You may add as many options as you would like.

- **Response.Write AC.EmbedAutoComplete("AutoComplete1", "selAutoComplete1", True, True)**
    - This line of code will create the output.
    - The first parameter (AutoComplete1) is the name of the <INPUT> the object should create. This name can be used in subsequent pages to extract the value typed by the user. You may also use the value of the linked <SELECT> list, as the code rendered by the Auto Complete Text Box will change the value of the linked select list automatically as the user types.
    - The second parameter (selAutoComplete1) is the name of the HTML <SELECT> list you are linking to.
    - The third parameter (True) tells the component if it should force the user to type a match within the <SELECT> list. If this is set to false, it will attempt to match the user input with an option from the list, but will allow the user to enter a free-form value as well.

  The last parameter (True) tells the component to link by Text. If this parameter were set to false, the user could type the value of the option, and the

    Auto Complete text box would associate the entered value with the text of the option.

- **Set AC = Nothing**
    - Garbage Collection– Releasing the object you created

Properties List

Property Name	Type	Data Type	Functionality
[No Properties]			

Figure 12 - AutoComplete Textbox Properties List

Methods List

Method Name	Parameter	Data Type	Functionality
EmbedAutoComplete	HTMLFieldNameToBuild	String	The name of the Auto Complete <INPUT> to be generated.
	HTMLSelectNameToLink	String	The name of the HTML <SELECT> List to link to.
	ForceAutoComplete	Boolean	Permits or restricts the user from entering text that does not match an option in the HTML select list.
	IsLinkedAsText	Boolean	Permits or restricts the user from entering the HTML <SELECT> List value instead of the HTML <SELECT> list’s text.

Figure 13 - AutoComplete Textbox Methods List

4

Coding with the Embedded and Popup Calendar

Overview

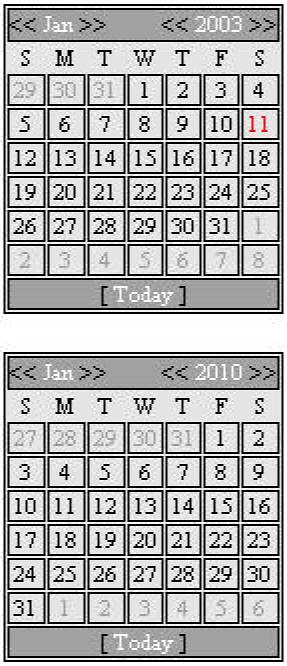
This class implements the Embedded and Popup Calendar control. The Embedded and Popup Calendars are a perfect tool for presenting your users with an easy way to enter a date and are widely used when setting up report criteria. The Embedded Calendar displays on the web page and remains displayed at all times. The Popup Calendar is displayed as a textbox, with an image next to it (the image is specified by the ASP developer via a property setting). When the user clicks on the image the calendar “pops up” next to the text box and fills the text box with whatever date the user clicked on. These calendars also relieve the ASP Developer from the task of validating the user input, as the calendar will ensure that any date chosen is a valid date.

Code Implementation (Embedded)

```
<%@ Language=VBScript %>
<%
Option Explicit
Response.ExpiresAbsolute = 0

%>
<script language="JavaScript">
      function cal1_OnClick(whichDate) {
            alert( "You clicked Calendar 1.  You selected " + whichDate
);
      }

      function cal2_OnClick(whichDate) {
            alert( "You clicked Calendar 2.  You selected " + whichDate
);
      }
</script>
<%
Dim Cal
Set Cal = CreateObject("BZSCoreComp.Calendar")

Cal.ActiveDate = Now
Response.Write Cal.DisplayEmbedded("cal1")

Response.Write "<br>"

Cal.ActiveDate = "1/1/2010"
Response.Write Cal.DisplayEmbedded("cal2")

Set Cal = Nothing
%>
```

Figure 14 - Code Implementation for Embedded Calendar

Output (Embedded)



<< Jan >> << 2003 >>

S	M	T	W	T	F	S
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

[Today]

<< Jan >> << 2010 >>

S	M	T	W	T	F	S
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

[Today]

Figure 15 - Embedded Calendar Output

Code Breakout and Explanation (Embedded)

- **<script language="JavaScript"> function cal1_OnClick(whichDate) {**
    - The Embedded Calendar requires that client side JavaScript be written to handle the "OnClick" event of the calendar (This script is not necessary with the Popup Calendar). The function written must be named with the following rules in mind.
        - The name of the function is case-sensitive
        - The name of the function must begin with the name of the calendar which it is responsible for handling.
        - The name of the function must end in "_OnClick(whichDate)", where "whichDate" is the value of the date passed in from the calendar. You may use any name you choose for this value.

- **`alert( "You clicked Calendar 1.  You selected " + whichDate )`**
    - We simply chose to display the date the user clicked in an "alert" box. You could do anything you wish here, including redirecting to a page with the date embedded in a querystring, etc.

- **Set Cal = CreateObject("BZSCoreComp.Calendar")**
    - The Embedded Calendar is encapsulated in a COM object called "BZSCoreComp.Calendar". This line simply creates an instance of the object so you can use the Embedded Calendar
- **Cal.ActiveDate = Now**
    - Here, you set the calendar's initial date. This is the date the calendar will show on initialization. The ActiveDate property will accept a VB Date data type, or a string representing a date, formatted as "mm/dd/yyyy".
- **Response.Write Cal.DisplayEmbedded("cal1")**
    - This line streams the client side script to the browser, creating the calendar. The calendar created in this line is named "cal1", as shown. You may create as many calendars on one page as you like, provided you give them unique names.
- **Set Cal = Nothing**
    - Garbage Collection – Releasing the object you created

Properties List (Embedded)

Property Name	Type	Data Type	Functionality
ActiveDate	Get/Let	Date or String	This sets the calendar's initial date; the date that displays when the page is first loaded
ImageURL	Get/Let	String	This is the image that displays next to the popup calendar. This property is not used for the Embedded Calendar. If it is set, it will be ignored

Figure 16 - Embedded Calendar Properties List

Methods List (Embedded)

Method Name	Parameter	Data Type	Functionality
DisplayEmbedded	CalName	String	The name of the calendar

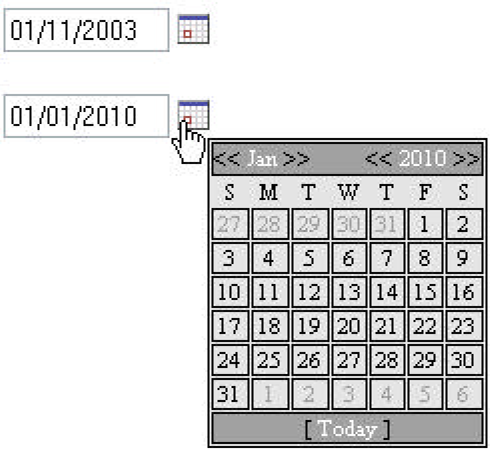
Figure 17 - Embedded Calendar Methods List

Code Implementation (Popup)

```
<%@ Language=VBScript %>
<%
Option Explicit

Dim Cal

Set Cal = CreateObject( "BZSCoreComp.Calendar" )

%><form><%
Cal.ImageURL = "Calendar.gif"

Cal.ActiveDate = Now
Response.Write Cal.DisplayPopup("CALENDAR_1")

Response.Write "<BR><BR>"

Cal.ActiveDate = "1/1/2010"
Response.Write Cal.DisplayPopup("CALENDAR_2")

%></form><%

Set Cal = Nothing
%>
```

Figure 18 - Code Implementation for Popup Calendar

Output (Popup)



01/11/2003

01/01/2010

<< Jan >> << 2010 >>

S	M	T	W	T	F	S
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

[Today]

Figure 19 - Popup Calendar Output

Code Breakout and Explanation (Popup)

- **Set Cal = CreateObject("BZSCoreComp.Calendar")**
    - The Popup Calendar is encapsulated in a COM object called "BZSCoreComp.Calendar". This line simply creates the object so you can use the Embedded Calendar.

- **%><form><%**
  - The output of the Popup Calendar must be contained within <form> tags.
- **Cal.ImageURL = "Calendar.gif"**
  - This line sets the image that is to be displayed next to the Popup Calendar's Textbox. This is what the user will click to display the calendar
- **Cal.ActiveDate = Now**
  - Here, you set the calendars initial date. This is the date the calendar will show on initialization. The "ActiveDate" property will accept a Date data type, or a string representing a date, formatted as "mm/dd/yyyy". If you do not set this property, the calendar will use the current date as its default value.
- **Response.Write Cal.DisplayPopup("CALENDAR_1")**
  - This line will stream client side code to the browser which will create the textbox and image for the Popup Calendar. The name of this particular calendar is "CALENDAR_1"
- **Set Cal = Nothing**
  - Garbage Collection– Releasing the object you created

Properties List (Popup)

Property Name	Type	Data Type	Functionality
ActiveDate	Get/Let	Date or String	This sets the calendars initial date… the date that displays when the page is first loaded
ImageURL	Get/Let	String	This is the image that displays next to the popup calendar.

Figure 20 - Popup Calendar Properties List

Methods List (Embedded)

Method Name	Parameter	Data Type	Functionality
DisplayPopup	CalName	String	The name of the calendar

Figure 21 - Popup Calendar Methods List

5

Coding with the Right-Click Popup Menu

Overview

This class implements the Right-Click Popup Menu. The Right-Click Popup Menu gives the ASP developer the ability to display his/her own context menu within Internet Explorer. This class gives your Web page a professional look and can provide a handy menu for the user to execute widely used functionality. This class is used frequently to help in preventing users from saving images from your Web page, or viewing the source of your Web page. The section below uses the “simple” Popup Menu example.

Please note: BreZaSoft, LLC does not guarantee that the implementation of the Right-Click Popup Menu will completely secure your images, client-side source, or any other information from your Web site. Savvy Internet users may still be able to access this information by other means.

Code Implementation

```
<%@ Language=VBScript %>
<%
Option Explicit

Dim Pop
Set Pop = CreateObject("BZSCoreComp.PopUpMenu")

Pop.AddMenuTitle "Your Company"
Pop.AddMenuItem "Back", "window.parent.history.back();", "Popup-
Back.gif"
Pop.AddMenuItem "Forward", "window.parent.history.forward();", "Popup-
Forward.gif"
Pop.AddHorizontalRule "Popup-HR.gif"
Pop.AddMenuItem "Add this page to Favorites",
"window.parent.external.AddFavorite(self.location.href,'Popup Menu
Example');", "Popup-Favorite.gif"
Pop.AddMenuItem "View Source", "window.parent.location='view-
source:'+window.parent.location.href;", "Popup-Source.gif"
Pop.AddMenuItem "Print", "window.parent.print();", "Popup-Print.gif"
Pop.AddMenuItem "Refresh", "window.parent.location.reload();", "Popup-
Refresh.gif"
Response.Write Pop.EmbedPopupMenu

Set Pop = Nothing
%>
```

Figure 22 - Code Implementation for Right-Click Popup Menu

Output

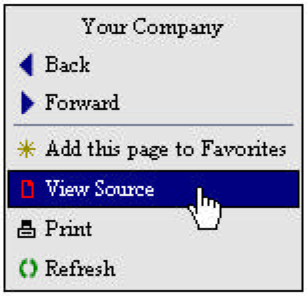


Figure 23 - Right-Click Popup Menu Output

Code Breakout and Explanation

- **Set Pop = CreateObject("BZSCoreComp.PopUpMenu")**
  - The Right-Click Popup Menu is encapsulated in a COM object called "BZSCoreComp.PopUpMenu". This line simply creates an instance of the object so you can use the Popup Menu
- **Pop.AddMenuTitle "Your Company"**
  - This line will display a title within the menu. The title is not clickable, and cannot be associated with any code.
- **Pop.AddMenuItem "Back", "window.parent.history.back();", "Popup-Back.gif"**
  - This is one of the nodes in the menu. The first parameter is what will be displayed in the menu to the user. In this case, the word "Back".
  - The second parameter is the JavaScript code to be executed when the user clicks this menu item. Please note that the popup menu is considered to be it's own window. To affect the page which called the Popup Menu, you must include "window.parent" in front of your JavaScript code.
  - The last parameter is the image that should display for this particular node. In this case, the image is called "Popup-Back.gif"
- **Response.Write Pop.EmbedPopupMenu**
  - This will stream the client-side code to the browser, and enable the Right-Click Popup Menu

Properties List

Property Name	Type	Data Type	Functionality
BGColor	Get/Let	String	This sets the background color of the Popup Menu. It can be the Hexadecimal representation, or a valid named color.
BorderColor	Get/Let	String	This sets the border color of the Popup Menu. It can be the Hexadecimal representation, or a valid named color.
OnMouseOut	Let	String	This represents the JavaScript code to be executed when the mouse pointer leaves a node.
OnMouseOver	Let	String	This represents the JavaScript code to be executed when the mouse pointer enters a node.
Width	Get/Let	Integer or String	The width of the menu. Note: the height of the menu is auto-calculated.

Figure 24 - Right-Click Popup Properties List

Methods List

Method Name	Parameter	Data Type	Functionality
AddHorizontalRule	ImagePath_1x1Pixel	String	This points to an image that will be stretched across the width of the menu to create a horizontal rule. This image should be a 1x1 pixel image. "Popup-HR.gif" is an image that comes packaged with the ASP examples. Feel free to use this image as your horizontal rule.
AddMenuItem	Display	String	This is the word (or words) to be displayed in the menu node.
	OnClick	String	The JavaScript code to be executed when the user clicks the node. This line of code will generally begin with "window.parent".
	ImagePath	String	The path to the image to be displayed next to the node in the menu.
AddMenuTitle	Display	String	The word (or words) to be displayed in the title of the menu. The title is not clickable, and cannot have JavaScript code associated with it.
EmbedPopupMenu			Embeds the popup menu in the browser.

Figure 25 - Right-Click Popup Methods List

6

Coding with the Progress Bar

Overview

This class implements the Progress Bar. The Progress Bar is a real time progress bar that mimics the progress bar controls familiar to Win32 programming for years. This class does a great job of informing your users of the progress of intensive processing. The Progress Bar is most widely used during the calculation of reports.

Code Implementation

```
<%
Option Explicit

Response.Buffer = False
Dim PBar
Dim dtEnd
Dim x
Dim Inc

Set PBar = CreateObject("BZSCoreComp.ProgressBar")
%>
<form name="frmTest">
<body>
This page demonstrates how you can use the ProgressBar Component.<br>
<%
PBar.Color = "0000FF"
PBar.Max = 200

Response.Write PBar.Show(True,True)

Randomize

While PBar.GetValue < PBar.Max
      Inc = CInt(rnd * 3) + 1
      dtEnd = DateAdd("s", .5, Now())
      Do While Now < dtEnd
      Loop
      Response.Write PBar.SetValue( PBar.GetValue + Inc )
      Response.Write "This was written after 1 iteration.<br>"
Wend
Response.Write PBar.HideOutput(False)

Set PBar = Nothing
%>
</body>
</form>
```

Figure 26 - Code Implementation for the Progress Bar

Output

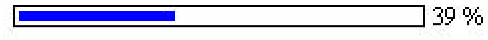


Figure 27 - Progress Bar Output

Code Breakout and Explanation

- **Response.Buffer = False**
    - This line is necessary for the progress bar to function properly. This line tells the server not to hold on to the output of the processing it is doing, but rather to stream it to the client as it processes it. Without this line of code, the JavaScript generated by the Progress Bar class will not be written to the browser until the very end, and the commands responsible for updating the Progress Bar will not execute. Implementation of the progress bar can hide this streamed output if you desire.
- **Set PBar = CreateObject("BZSCoreComp.ProgressBar")**
    - The Progress Bar is encapsulated in a COM object called "BZSCoreComp.ProgressBar". This line simply creates an instance of the object so you can use the Progress Bar
- **<form name="frmTest">**
    - The code generated by the Progress Bar class and streamed to the Browser must be contained within <Form> tags
- **PBar.Color = "0000FF"**
    - The Hexadecimal representation of the color of the bar
- **PBar.Max = 200**
    - This tells the progress bar what the maximum interval value of your processing is. In this case, we will iterate 200 times. Generally, the progress bar is used while iterating a Recordset. If this is the case, then the .Max property would be set to the Recordset's RecordCount. (If using ADO, be sure to set your Recordset's "CursorLocation" property to "adUseClient (2)", so the RecordCount property will reflect an accurate amount) Later, when you set the value of the progress bar, it will automatically calculate the percentage of processing complete based on the current value and max value of the bar. For example, if there were 10 iterations, you would set the .Max property to "10". Then, during the 6th iteration, the progress bar would display "60%"
- **Response.Write PBar.Show(True,True)**
    - This line of code will embed the Progress Bar into the page.
    - The first parameter (True) tells the progress bar to hide the output of the processing.
    - The second parameter (True) tells the progress bar to hide itself after the processing has completed.
- **Response.Write PBar.SetValue(PBar.GetValue + Inc)**
    - This sets the Progress Bar's current value inside the iteration. When this line is set, the progress bar will calculate the percentage of processing completed, then update itself to give that representation to the user.

    - Generally, if used within a Recordset, this value is set to the current pointer location of the Recordset.
- **Response.Write PBar.HideOutput(False)**
    - This tells the Progress bar to display the output which has been hidden during processing.
- **Set PBar = Nothing**
    - Garbage Collection– Releasing the object you created.

Properties List

Property Name	Type	Data Type	Functionality
Color	Get/Let	String	The color of the Progress Bar.
Max	Get/Let	Long	The maximum interval value of your processing.

Figure 28 - Progress Bar Properties List

Methods List

Method Name	Parameter	Data Type	Functionality
GetValue		Long	Returns the current value of the Progress Bar.
HideOutput	Hidden	Boolean	Tells the Progress Bar if the output of the processing of the Web page should be hidden or not.
SetValue	NewValue	Long	Tells the Progress Bar what its current iteration value is.
Show	HideOutputDuring	Boolean	Tells the Progress Bar if it should hide the output of the Web page during processing.
	HideProgressBarAfter	Boolean	Tells the Progress Bar if it should hide itself after processing.

Figure 29 - Progress Bar Methods List

7

Coding with the Tabbed Panels

Overview

This class implements the Tabbed Panels. The Tabs will automatically create an IFrame within each panel, and allow the ASP Developer to specify the URL of each frame. Adding Tabs to your Web page allows you to group information into one page, which would have generally been stored on multiple pages. Clicking on each tab will display the information contained within the panel, without the necessity to reload that information. The Tabs are frequently used in combination with the Tree Control to provide a powerful navigation system for your users. The below examples are taken from the simple Tabs example packaged with the Web Developers Core Component Library.

Code Implementation

```
<%@ Language=VBScript %>
<%
Option Explicit

Dim Tabs
Set Tabs = CreateObject("BZSCoreComp.Tabs")

Tabs.TabCount = 3
Tabs.PanelWidth = 580
Tabs.PanelHeight = 380
Tabs.TabWidth = 100
Tabs.LayerColor = "e0e0e0"

Tabs.TabName = "Google"
Response.Write Tabs.WriteTab("GoogleFrame", "http://www.google.com/")

Tabs.TabName = "DogPile"
Response.Write Tabs.WriteTab("DogPileFrame", "http://www.dogpile.com/")

Tabs.TabName = "Microsoft MSDN"
Response.Write Tabs.WriteTab("MicrosoftFrame",
"http://msdn.microsoft.com/")

Set Tabs = Nothing
%>
```

Figure 30 - Code Implementation for Tabbed Panels

Output

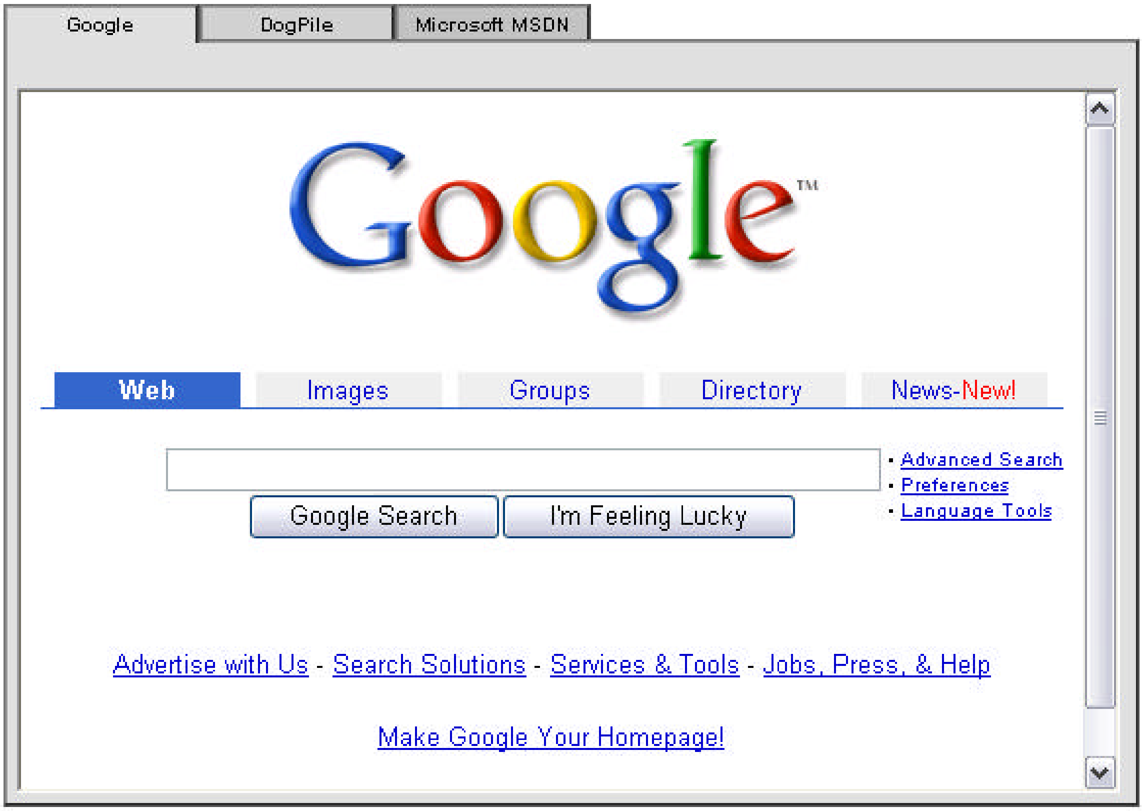


Figure 31 - Tabs Output

Code Breakout and Explanation

- **Set Tabs = CreateObject("BZSCoreComp.Tabs")**
    - The Tabs are encapsulated in a COM object called “BZSCoreComp.Tabs”. This line simply creates an instance of the object so you can use the Tabbed Panels.
- **Tabs.TabCount = 3**
    - The number of Tabs to be created.
- **Tabs.PanelWidth = 580**
    - The width of the panels in pixels.
- **Tabs.PanelHeight = 380**
    - The height of the panels in pixels.
- **Tabs.TabWidth = 100**
    - The width of the Tabs above the panels in pixels.
- **Tabs.LayerColor = "e0e0e0"**
    - The hexadecimal representation of the color of the layer.
- **Tabs.TabName = "Google"**
    - The name to be displayed on this particular Tab.

- **Response.Write Tabs.WriteTab("GoogleFrame", "http://www.google.com/")**
  - This adds the Tab to the page.
  - The first parameter tells the Tabs what the name of the frame inside the tab is called.
  - The second parameter tells the Tabs what the location of the frame should be.
- **Set Tabs = Nothing**
  - Garbage Collection– Releasing the object you created.

Properties List

Property Name	Type	Data Type	Functionality
IsVisible	Get/Let	Boolean	Displays or hides the Tab. This functionality can be used to give the impression of dynamically created tabs (The code to accomplish this requires advanced JavaScript, and is not included in this documentation).
JavaScriptOnClick	Let	String	The additional JavaScript code to be executed (if any) when the user clicks on a tab.
LayerColor	Get/Let	String	The hexadecimal representation of the color of the layer.
PanelHeight	Get/Let	Integer	The height of the panels in pixels.
PanelWidth	Get/Let	Integer	The width of the panels in pixels.
TabCount	Get/Let	Integer	The number of tabs and panels to be created.
TabName	Get/Let	String	The name displayed on the tab.
TabWidth	Get/Let	Integer	The width of the tab above the panel in pixels.

Figure 32 - Tabs Properties List

Methods List

Method Name	Parameter	Data Type	Functionality
MoveNextLayerColor			Automatically calculates a lighter shade, and sets the next panel color to that calculated value.
MovePreviousLayerColor			Automatically calculates a darker shade, and sets the next panel color to that calculated value.

Figure 33 - Tabs Methods List

8

Coding with the Tree Control

Overview

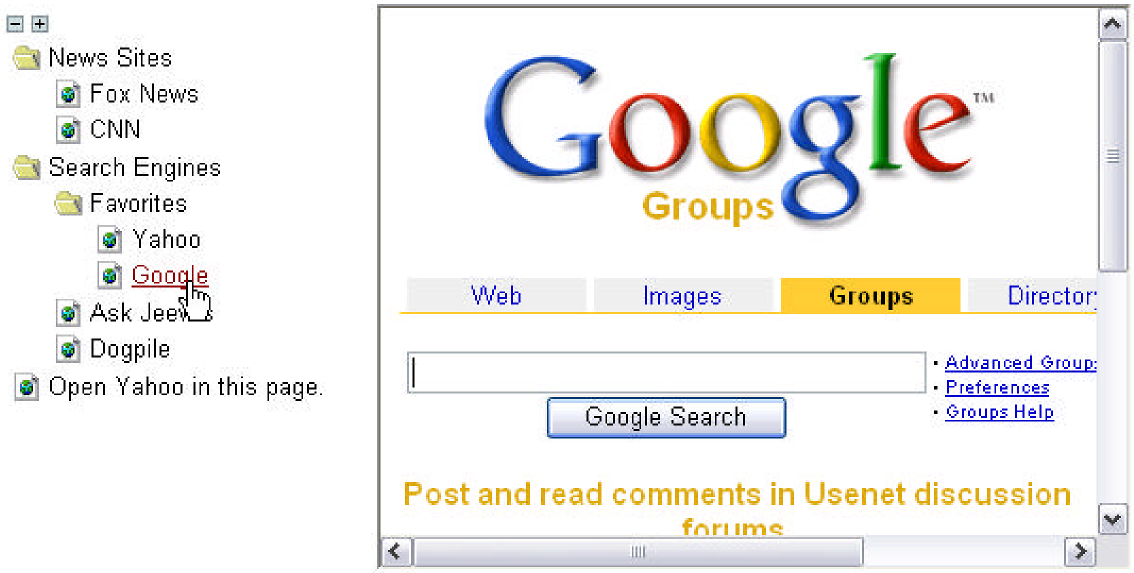
This class implements the Tree Control. The Tree control gives you the power create a tree view which can contain nodes and/or any HTML code or JavaScript you would like. You can create as many categories and sub-categories as you like. The Tree Control is most often used as a navigation tool, but has many other implementations as well. The following code implementation describes the simple example packaged with the Web Developers Core Component Library.

Code Implementation

```
<%@ Language=VBScript %>
<%
Option Explicit

%>
<STYLE>
A
{
    FONT-SIZE: 10pt;
    COLOR: #000000;
    FONT-FAMILY: Arial;
    TEXT-DECORATION: none
}
A:hover
{
    COLOR: #8b0000;
    TEXT-DECORATION: underline
}
</STYLE>
<body bgcolor="FFFFFF">
<table border=0 width=100%>
<tr>
<td bgcolor="FFFFFF" align=left valign=top width=20%>
<%
Dim Tree
Set Tree = CreateObject("BZSCoreComp.Tree")

Tree.CollapseAllImage = "Tree-Minus.gif"
Tree.ExpandAllImage = "Tree-Plus.gif"
Tree.CollapsedImage = "Tree-FolderClosed.gif"
Tree.ExpandedImage = "Tree-FolderOpen.gif"
Response.Write Tree.WriteExpandCollapse
Response.Write Tree.WriteMenuHeader("News Sites")
      Response.Write Tree.WriteNode("Fox News",
"http://www.FoxNews.com/", "treebody", "Tree-App.gif")
      Response.Write Tree.WriteNode("CNN", "http://www.CNN.com/",
"treebody", "Tree-App.gif")
Response.Write Tree.WriteMenuFooter()

Response.Write Tree.WriteMenuHeader("Search Engines")
      Response.Write Tree.WriteMenuHeader("Favorites")
            Response.Write Tree.WriteNode("Yahoo",
"http://www.yahoo.com/", "treebody", "Tree-App.gif")
            Response.Write Tree.WriteNode("Google",
"http://www.deja.com/", "treebody", "Tree-App.gif")
      Response.Write Tree.WriteMenuFooter()
      Response.Write Tree.WriteNode("Ask Jeeves",
"http://www.askjeeves.com/", "treebody", "Tree-App.gif")
      Response.Write Tree.WriteNode("Dogpile",
"http://www.dogpile.com/", "treebody", "Tree-App.gif")
Response.Write Tree.WriteMenuFooter()
Response.Write Tree.WriteNode("Open Yahoo in this page.",
"http://www.yahoo.com", "_self", "Tree-App.gif")
Set Tree = Nothing
%>
</td>
<td align=left valign=top width=80%>
      <iframe name="treebody" width=400 height=300
src="about:blank"></iframe>
</td>
</tr>
</table>
</body>
```

Figure 34 - Code Implementation for the Tree Control

Output



Post and read comments in Usenet discussion forums

Figure 35 - Tree Control Output

Code Breakout and Explanation

- **<STYLE>**
  - To improve the look of the Tree, we wrote Cascading Style Sheet code.
- **Set Tree = CreateObject("BZSCoreComp.Tree")**
  - The Tree Control is encapsulated in a COM object called.
  - "BZSCoreComp.Tree". This line simply creates an instance of the object so you can use the Tree Control.
- **Tree.CollapseAllImage = "Tree-Minus.gif"**
  - The image that the user will click on to collapse all nodes within the Tree. For this image to appear, you must have also called the "WriteExpandCollapse" method which will display the image at the top of the Tree.
- **Tree.ExpandAllImage = "Tree-Plus.gif"**
  - The image that the user will click to expand all nodes within the Tree. For this image to appear, you must have also called the "WriteExpandCollapse" method which will display the image at the top of the Tree.
- **Tree.CollapsedImage = "Tree-FolderClosed.gif"**
  - The image to represent a category which is closed.
- **Tree.ExpandedImage = "Tree-FolderOpen.gif"**
  - The image to represent a category which is open.
- **Response.Write Tree.WriteExpandCollapse**
  - This will display the images that the user will click to expand or collapse all nodes in the Tree. The images are displayed at the top of the tree.
- **Response.Write Tree.WriteMenuHeader("News Sites")**
  - This tells the tree to begin a new category, and to name it "News Sites". The image associated with this category is defined in the "CollapsedImage" and "ExpandedImage" properties.
- **Response.Write Tree.WriteNode("Fox News", "http://www.FoxNews.com/", "treebody", "Tree-App.gif")**
  - This writes a node within the category.
  - The first parameter is the name of the node.
  - The second parameter is the URL to go to when the user clicks the node.
    - If you want to associate JavaScript with the node, you may either do so by not using a "WriteNode" method for the contents of the node, or you can set the URL location to **#" onclick='[JavaScript code here]'**.
  - The third parameter is the target of the URL.
  - The fourth parameter is the image to display in the node.
- **Response.Write Tree.WriteMenuFooter()**
  - This signifies the end of a category.
- **Set Tree = Nothing**
  - Garbage Collection – Releasing the object you created.

Properties List

Property Name	Type	Data Type	Functionality
CollapseAllImage	Get/Let	String	The path of the image to display at the top of the tree. When clicked, this will collapse all nodes in the tree.
CollapsedImage	Get/Let	String	The image that represents a closed category.
ExpandAllImage	Get/Let	String	The path of the image to display at the top of the tree. When clicked, this will expand all nodes in the tree.
ExpandedImage	Get/Let	String	The image that represents an open category.
TextClass	Get/Let	String	If you have a cascading style sheet defined and included in your page (example: <LINK rel="stylesheet" type="text/css" href="../CSS/Default.css">), this is the class within that Style Sheet to use for the tree.
Width	Get/Let	Integer	The width of the Tree in pixels.

Figure 36 - Tree Control Properties List

Methods List

Method Name	Parameter	Data Type	Functionality
Initialize			Legacy code, used to initialize the tree. Calling this method is no longer necessary, but will not hurt execution of the code if called.
WriteExpandCollapse			Displays the images at the top of the Tree responsible for expanding and collapsing all nodes.
WriteMenuFooter			Method to signify the end of a category.
WriteMenuHeader	MenuName	String	Method to signify the beginning of a category. This will also display the Menu Name on the category node.
WriteNode	NodeName	String	Writes a node within a category. This is the name of the node.
	URI	String	The Universal Resource Indicator. The location to redirect to when the user clicks a node.
	Target	String	The frame name (or "_self") that will be redirected when the user clicks the node.
	ImagePath	String	The path to the image to be displayed on the node.

Figure 37 - Tree Control Methods List