WebClasses Make Registration Easy

Use VB6's new WebClass technology to easily build flexible and powerful Internet applications.

by Stan Schultes and Chris Barlow

WHAT YOU NEED

VB6 Professional or Enterprise Edition

NT Server with IIS 4 or Windows 95/98 with Personal Web Server V3

Access 97 Database

Microsoft Data Access Components 1.5 or 2.0

Earlier this year, we showed you how to write a registration application that allows a user to visit your Web page and register a software product ["Create a User Registration Class," *VBPJ* April 1998, and "Add E-Mail Registration to Your Server," *VBPJ* May 1998]. After registration, the app sends a serial number back to the user by e-mail, as well as a link to the Web server to continue the registration process. This VB5 application we demonstrated for you earlier creates an ActiveX DLL that runs on the Web server, and a companion Active Server Pages (ASP) script that determines the user's state and calls the DLL appropriately. Now VB6 offers a fresh approach to programming this type of application using a new technology called WebClasses.

WebClasses, in a nutshell, are ActiveX DLLs that run on a Web server. WebClasses allow hyperlinks on an HTML page in the client-side browser to fire events in the server DLL. The WebClass programming model is similar to the traditional VB model—instead of a form that contains controls, the WebClass application features a Web page that contains controls. The full VB event model is exposed to the developer on the server side, allowing a highly interactive style of Web development. In this column, we'll walk through the construction of a simple WebClass application, showing you how easy it is to get started.

To create a simple WebClass application, start VB6 and select IIS Application from the New Project dialog box. Name the project SimpleReg in the Properties window. Double-click on the WebClass1 designer in the Project Explorer window (you can see both the Properties window and Project Explorer through VB's View menu). In the Properties window, name the WebClass wcSimple, and enter SimpleReg in the NameInURL property. This creates a special startup file called SimpleReg.asp when the ActiveX DLL is compiled. Now save the project.

A WebClass application displays HTML on the user's browser through the use of HTML templates. Because VB6 doesn't include an HTML editor, you must create the templates outside VB (the VB6 DHTML Designer is completely unrelated to WebClasses). You can use any HTML editor, but because Visual Studio includes Visual InterDev (VID) 6.0, VID is a logical choice.

Make Visual InterDev the default HTML editor for Visual Basic by going to VB6's Tools | Options menu. On the Advanced tab, fill in the External HTML Editor box with this path to Visual InterDev: C:\Program Files\Microsoft Visual Studio\Common\IDE\IDE98\Devenv.exe. You'll notice that Visual InterDev creates copies of the template files used in your project. For example, if your template is called WebPage.htm, InterDev creates a copy named WebPage1.htm. Be aware of this if you edit your template with Notepad or another editor.

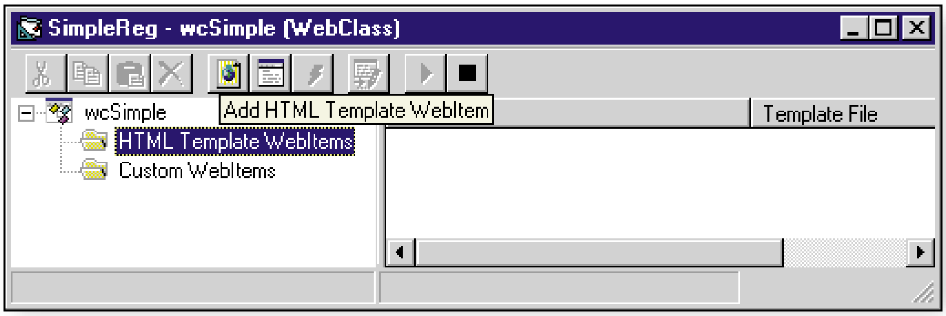

Figure 1 WebClass Designer. Add an HTML template to the project using the WebClass Designer. Either click on the Add Template button (above the ToolTip), or right-click on the HTML Template WebItems folder and select Add HTML Template… from the context menu.

In the sample code, the SimpleReg.htm file is a template created with Visual InterDev. Select the HTML Template WebItem folder in the left-hand pane of the WebClass designer. Add the SimpleReg template to the project by clicking on the Add HTML Template WebItem button and selecting SimpleReg.htm in the file dialog (see Figure 1). Name this new WebItem tplSimple. Double-click on the tplSimple WebItem to view the code for the WebClass. Find the WebClass_Start procedure and replace the generated code with this code to display the template when the WebClass loads:

```
Private Sub WebClass_Start()
'Display the main HTML template
   Session("Title") = "Enter Your " & _
      "Registration Information"
   tplSimple.WriteTemplate
End Sub
```

The WebClass_Start procedure is similar to the Form_Load procedure in a traditional VB application. Run the project and watch your browser display the template HTML page. Exit the browser and stop the VB application (this is the normal way of shutting down a WebClass application during the debug/development process).

In the WebClass designer, right-click on the tplSimple WebItem and select Edit HTML Template… from the context menu. You will see the template page in your default HTML editor. Looking at the source code, you can see this is a simple page with three special HTML elements called tags, formatted like this:

```
<WC@TAGTITLE>Title</WC@TAGTITLE>
```

The text portion of these tags ("Title") is replaced at run time by code you add to the WebClass. When the WriteTemplate method executes, it calls a special procedure named ProcessTag once for each tag found in the HTML template. Back in VB, double-click on the tplSimple WebItem to display the Code window. Select the tplSimple_ProcessTag procedure from the comboboxes at the top of the code window, and add this code:

```
Dim sHTM As String
Select Case TagName
Case "WC@TAGTITLE"
   TagContents = Session("Title")
Case "WC@TAGEMAIL"
   sHTM = "Enter Email Address:<br>"
   sHTM = sHTM & "<input type='text'"
   sHTM = sHTM & " name='Email'><br>"
   TagContents = sHTM
Case "WC@TAGNAME"
   sHTM = "Name:<br>"
   sHTM = sHTM & "<input type='text'"
   sHTM = sHTM & " name='Name'><br>"
   TagContents = sHTM
End Select
```

Passing back HTML code in the TagContents parameter makes the tag replacement. Set a breakpoint at the tplSimple.WriteTemplate line in the WebClass_Start procedure, and run your application. At the breakpoint, single-step the project using the F8 key to see the order of events. You can see the ProcessTag procedure is called three times—once for each tag in the template. What is really cool is that the procedure replaces the E-mail and Name tags at run time with HTML code that generates textboxes on the browser.

The ability to interactively debug your server-side event code illustrates one of the most powerful capabilities of WebClass application development with VB. In VB5 Web development, VBScript in an ASP script does a lot of the work. It is not possible to debug VBScript in an ASP script interactively—trial and error is unfortunately the only solution. With VB6, however, all server-side code runs in the WebClass, and you can interactively debug it using VB's excellent design environment.

Connecting an Event

The next step in this simple application is to add event-handling code for the Register Me button on the form. Back in VB's WebClass designer, click on the tplSimple WebItem. In the right-hand pane, you'll see a list of the HTML elements on the page to which you can attach events. Right-click on the Form1 tag and select the Connect to Custom Event menu item. Name the added event RegisterMe, and notice the event name in the Target column next to Form1. Take a quick look at the HTML template (right-click on tplSimple and select Edit HTML Template). Find the form element, which now looks like this:

```
<FORM method='post' _
   action=SimpleReg.ASP?WCI=tplSimple _
   &amp;WCE=RegisterMe&amp;WCU>
```

SimpleReg.asp is the special ActiveX DLL startup script. The values after the question mark indicate the WebClass Item (WCI) is tplSimple, the WebClass Event (WCE) is RegisterMe, and the WebClass URLData (WCU) is empty. When the user clicks on the Register Me button in the browser, this line makes the tplSimple_RegisterMe event fire in the ActiveX server DLL. In this case, the form action is connected to a server-side event, but you can set any hyperlink in the browser to fire an event on the server. WebClasses bring magic to Web application development—they extend VB's event-driven style of programming to browser-based applications.

Double-click on the RegisterMe event in the left-

RESOURCES

Chris Barlow's Getting Started column, "Create a User Registration Class," **VBPJ** April 1998.

Chris Barlow's Getting Started column, "Add E-Mail Registration to Your Server," **VBPJ** May 1998.

LINK

For an introduction to WebClasses, read the article by Sean Alexander of Microsoft at http://msdn.microsoft.com/developer/news/feature/vb6jun98/wbclsprmr.htm.

hand pane of VB's WebClass designer to show the Code window. Add this code to the event:

```
If Len(Request.Form("Email")) = 0 Then
   Session("Title") = "Please " & _
      "Enter an Email Address!"
   Session("Email") = ""
   Session("Name") = ""
Else
   Session("Title") = "Here is " & _
      "your Entry!"
   Session("Email") = _
      Request.Form("Email")
   Session("Name") = _
      Request.Form("Name")
End If
tplSimple.WriteTemplate
```

Request.Form is the standard way of returning posted data from a browser page. Request.Form sets several Session variables and redisplays the same template HTML page. Set a breakpoint on the If statement and run the project. Enter an e-mail address and name, and click on the RegisterMe button. You'll see the Title field update, but the textboxes appear empty because you didn't put a value into them when you wrote the template. Stop the project and change the tplSimple_ProcessTag procedure (download Listing 1 from DevX; see the Download Free Code box for details). Give the textboxes values based on the Session variables. Run the project again and see what happens.

Database Access Using ADO

Finally, add a way to store and retrieve records from a database. For simplicity, we used Microsoft Access 97 as the database. Create a new database in your source directory with Access and name it Register.mdb. Add a table with two fields, Email and Name, and save the table as Users. Choose OK when Access prompts you to create a unique index field. Put a copy of Register.mdb in the root directory of your C drive (c:\).

In the project, add a reference to Microsoft ActiveX Data Objects Library, which you can find under Project | References…. You can use either the ADO Library 1.5 or 2.0. If you don't have the ADO Library installed, you can download the MDAC 2.0 setup kit from Microsoft at http://www.microsoft.com/data/ado. Change the code in the tplSimple.RegisterMe procedure (download Listing 2 from DevX; see the Download Free Code box for details).

This code saves the form data into Session variables, then opens an ADO recordset and searches for the e-mail address (download Listing 3 from DevX).

If found, the code returns the record. If not found, it adds a new record. This function sets the connect string:

```
Private Function Connect() As String
'returns an ADO connect string
   Const kDB = "DBQ=c:\register.mdb;"
   Const kDrv = "Driver={Microsoft" & _
      "Access Driver (*.mdb)}"
   Connect = kDB & kDrv
End Function
```

Set a breakpoint in tplSimple_RegisterMe, run the project, and see how the data access code works. This simple example should show you how easy it is to program in ADO. Keep in mind that any "real" application you deploy in your company would include error handling. You would also normally use the registry to store variables such as connect string information, and you would need to provide a way to update existing records and delete unwanted records.

Once you compile the project, two things happen. First, the created ActiveX DLL registers on the development machine, as with any ActiveX project in VB. Second, a special startup ASP file is created with the name specified in the NameInURL property of the WebClass. In the sample application, this file is named SimpleReg.asp. Take a look at this script with Notepad. When the user enters the URL of this file in the browser, IIS executes this script to create an instance of the SimpleReg.wcSimple WebClass. Microsoft recommends you not change this ASP script in any way.

From the user's standpoint, the WebClass version of the registration application works similarly to the one shown in the May 1998 column. Much of this column's actual code is similar to that shown in the prior column, except all the code is now included in a WebClass, and the startup ASP file doesn't include any user code.

Heed Caution Before Rushing Out

You should now have a solid understanding of how WebClasses work from this real-world comparison of a before-and-after application. You still might not want to rush out and convert all your existing Web applications to WebClass projects, however. Converting an application requires a lot of time, so you should do this only if there is a tangible business benefit. WebClasses allow some applications to be built or extended in ways not possible before, for example.

In the end, Microsoft has given us a powerful new toolset to use in the development of Internet applications. While building Web applications with VB5 is certainly possible, you must resort to a lot of programming tricks to get the ASP scripts to interact properly with VB DLLs. With VB6, gone are the limitations of VBScript and the lack of a true event-driven programming environment. WebClasses bring the full power of VB to Web development—a good thing indeed. VBPJ

About the Authors

Stan Schultes is the lead developer at SunOpTech, where he is responsible for development and worldwide support of the ObjectBank and ObjectOrder products. Stan holds a degree in computer engineering from Purdue University. Reach Stan at Stan@VBExpert.com.

Chris Barlow is president and CEO of SunOpTech, a developer of manufacturing decision-support applications. Chris holds degrees from Harvard Business School and Dartmouth College, where he worked with Drs. Kemeny and Kurtz on the Basic language. Reach Chris at Chris@VBExpert.com.

DOWNLOAD FREE CODE

Download the code for this issue of **VBPJ free** from http://www.vbpj.com.

To get the free code for this entire issue, type **VBPJ1298** into the Locator+ field at the top right of the VBPJ home page. (You first need to register, for free, on DevX.) The free code for this article includes all code listings, plus the code to build a simple WebClass app.

To get the bonus code for this article, available to DevX Premier Club members, type **VBPJ1298GS** into the Locator+ field. The bonus code includes all the free code described above, plus a VB6 rewrite of the user registration application discussed in the April and May Getting Started columns. The authors converted the app to an event-driven, server-side application using WebClasses from a server-side ASP script with a helper DLL.