Create a User Registration Class

by Chris Barlow

Build a class so you can easily reuse user information in all your apps.

Often you want to capture information about the users of your application or Web site so you can contact them in the future. Because you probably want to capture the same registration information for each user, regardless of the application, you should create a Visual Basic class. You can easily reuse a class in all your applications.

You need Visual Basic 5.0 to create the ActiveX DLL and Active Data Objects (ADO) 1.5, and you need Access 97 for the database access. To test the Web-based registration application, you need access to a Web server. If Internet Information Server (IIS) running on Windows NT is not available, you can use the Microsoft Personal Web Server (PWS) running on Windows 95 to test the Active Server Pages (ASP).

Start Access and create a Registration.mdb database with a single Users table (see Figure 1). You can download the database, the Visual Basic class, and the ASP page from the free, Registered Level of The Development Exchange (see the Code Online box at the end of this column for details). The user's e-mail address makes for a good, unique key for the database table because each user has a different e-mail address from every other user in the world. I've simplified the table a bit for this example, but you might want to add more user fields or even another table to keep track of the products and applications registered for a particular user.

Start Visual Basic and create a new ActiveX DLL project. Change the Project name to Registration, and the Class1 name to cUser. Use this code to add the same database fields as public properties of the class. Add a property called DBPath to store the path to the registration database:

```
Public UserID As Long
Public Email As String
Public FirstName As String
Public LastName As String
Public Salutation As String
Public Address As String
Public City As String
Public State As String
Public PostalCode As String
Public Country As String
Public Organization As String
Public Title As String
Public WorkNumber As String
Public FaxNumber As String
Public Password As String
Public SerialNumber As String
Public Created As Date
Public Modified As Date
Public Expire As Date
Public Comments As String
Public DBPath As String
```

As you read more about object-oriented programming with Visual Basic, you'll learn that robust classes generally use public property procedures to give the developer

Users : Table

Field Name	Data Type
UserID	AutoNumber
Email	Text
FirstName	Text
LastName	Text
Salutation	Text
Address	Text
City	Text
State	Text
PostalCode	Text
Country	Text
Organization	Text
Title	Text
WorkNumber	Text
FaxNumber	Text
Password	Text
SerialNumber	Text
Created	Date/Time
Modified	Date/Time
Expire	Date/Time
Comments	Memo

FIGURE 1 ***User Database Table.*** *When you design this table in Access, set the Indexed property for the Email field to Yes (No Duplicates) to make that field a unique key for this table.*

Chris Barlow is president of SunOpTech, a developer of document management, decision-support, and supply chain applications, including the ObjectBank, ObjectOrder, and ObjectJob systems. He holds two U.S. Patents related to software for decentralized distributed asynchronous object-oriented and scheduling systems. Chris, who is a frequent speaker at VBITS, Tech•Ed, and DevDays and has been featured in two Microsoft videos, holds degrees from Harvard Business School and Dartmouth College. Reach Chris at Chris@VBExpert.com or through his Web server at www.VBExpert.com.

more control over the values of the class properties. However, you can use this simpler form of property definition in a basic class such as this. You need to add methods to the class to provide the functionality—to Clear the properties, to Add a new user, and to Load an existing user. You also need methods to check whether a user already exists in the database (Exists) and whether a particular user and password is valid (IsValid).

Using ADO to access the database works well from ASP pages; you'll find it's not much different than the DAO you've been using. ADO is part of Microsoft's Universal Data Access strategy to provide high-performance access to any data source, including relational and nonrelational databases, e-mail and file systems, text and graphics, custom business objects, and more. Built as a layer on top of Microsoft's OLE DB, the hierarchy of objects found in DAO is de-emphasized in the ADO model. You don't need to create a Workspace and then a Database object in order to open a recordset. With DAO, you generally open a database at one point in your program and leave it open while other procedures open and close recordsets. ADO, on the other hand, operates in a stateless manner, where each database access opens and closes the database and recordset together. ADO is ideal for data access from a Web site where a browser retrieves one page of information, then disconnects until the user needs another page of information. ADO even lets you create temporary, unattached recordsets. If you don't have ADO installed on your computer, you can download it from Microsoft's Web site at http://www.microsoft.com/data/ado.

CREATE THE CLASS METHODS

It is easy to add a record to a table with ADO. Open an ADODB.Recordset object, clear the recordset with the AddNew method, set the values of each field, and write the record with the Update method:

```
Public Function Add() As Boolean
Dim rsADO As New ADODB.Recordset
On Error GoTo AddErr
rsADO.Open "Users", Connect, _
   adOpenDynamic, adLockOptimistic
rsADO.AddNew
rsADO!Email = Email
rsADO!FirstName = FirstName
rsADO!LastName = LastName
rsADO!Salutation = Salutation
rsADO!Address = Address
rsADO!City = City
rsADO!State = State
rsADO!PostalCode = PostalCode
rsADO!Country = Country
rsADO!Organization = Organization
rsADO!Title = Title
rsADO!WorkNumber = WorkNumber
rsADO!FaxNumber = FaxNumber
rsADO!Password = Password
rsADO!SerialNumber = SerialNumber
rsADO!Created = Created
rsADO!Modified = Modified
rsADO!Expire = Expire
rsADO!Comments = Comments
rsADO.Update
```

Notice that the recordset has a Connect parameter that gives the path to the database and driver for the database. I added a private Property Get procedure to the class to build and return the connect string based on the DBPath property. This technique will be useful if you decide to expand the class to let the user set the database driver—for example, to use an ODBC database rather than an Access database:

ONCE YOU CREATE THE CLASS, IT'S EASY TO WRITE AN ASP PAGE TO USE IT.

```
Private Property Get Connect() As _
   String
Connect = "DBQ=" & DBPath & _
   ";DRIVER={Microsoft Access " & _
   "Driver (*.mdb)}"
End Property
```

The Load method is even easier. With an e-mail address as the argument, you construct a simple SQL statement and pass it into the Recordset Open method rather than the table name. If a record is found, load the class properties from the recordset and return True:

```
Public Function Load(sEmail$) As _
   Boolean
Dim SQL$
Dim rsADO As New ADODB.Recordset
SQL = "Select * from Users where " & _
   "Email = '" & sEmail & "'"
rsADO.Open SQL, Connect
If Not rsADO.EOF Then
   UserID = rsADO!UserID
   Email = rsADO!Email
   FirstName = rsADO!FirstName
   LastName = rsADO!LastName
   Salutation = rsADO!Salutation
   Address = rsADO!Address
   City = rsADO!City
   State = rsADO!State
   PostalCode = rsADO!PostalCode
   Country = rsADO!Country
   Organization = rsADO!Organization
   Title = rsADO!Title
   WorkNumber = rsADO!WorkNumber
   FaxNumber = rsADO!FaxNumber
   Password = rsADO!Password
   SerialNumber = rsADO!SerialNumber
   Created = rsADO!Created
   Modified = rsADO!Modified
   Expire = rsADO!Expire
   Comments = rsADO!Comments
   Load = True
End If
rsADO.Close
Set rsADO = Nothing
End Function
```

Using these methods as models, you should find it easy to complete the other methods. For example, IsValid accepts the e-mail address and password as arguments and constructs a SQL statement to retrieve the password for that e-mail address. If the password in the recordset matches the password argument, then the method returns True (for the complete User class code, download Listing 1 from the free, Registered Level of The Development Exchange; see the Code Online box for details):

```
Public Function IsValid(sEmail$, _
   Password$) As Boolean
Dim SQL$
Dim rsADO As New ADODB.Recordset
SQL = "Select Password from Users " * _
   "where Email = '" & sEmail & "'"
rsADO.Open SQL, Connect
If Not rsADO.EOF Then
   If StrComp(Password, rsADO(0), _
      vbTextCompare) = 0 Then IsValid _
      = True
End If
rsADO.Close
Set rsADO = Nothing
End Function
```

You can test your new class from Visual Basic by designing a simple test harness as a Standard EXE project that references your class. However, because this class will be used primarily from ASP pages, I created two special methods—EntryForm and DisplayForm—that return an HTML form ready to display within a Web page (see Figure 2). As you write more classes to use on your Web pages, you'll find that methods that create HTML are extremely useful. The EntryForm method takes two arguments: the Web page that the form will be passed to when the user clicks on the submit button,

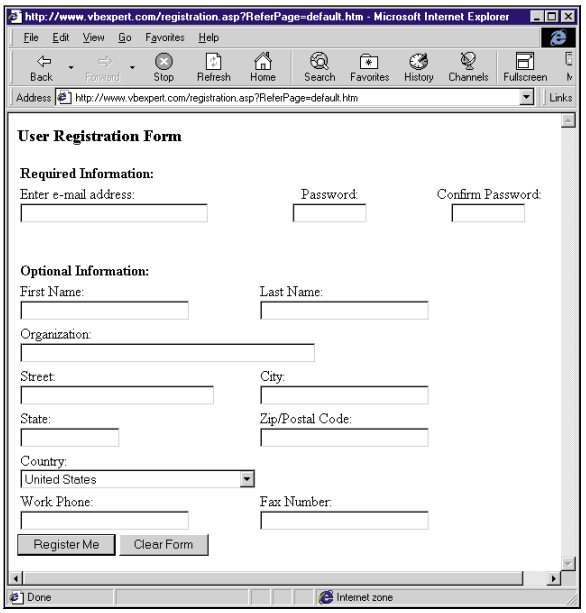

FIGURE 2 ***HTML Entry Form.*** *If you design class methods that return HTML like this form, you can spend your time coding in the Visual Basic IDE rather than struggling with the current state of script editors.*

and the submission method. The default for these arguments will post the form to an ASP file called Registration.asp. Note that you can use single quotes rather than double quotes in the HTML lines. In the first part of this method, notice that an HTML table is created with no borders in order to display the fields with the proper alignment (download Listing 2 from the free, Registered Level of DevX; see the Code Online box for details). The actual method is quite long because an HTML option box is filled with all valid Country names. The EntryForm method ends by adding the HTML buttons at the bottom of the form and returning the resulting HTML:

```
HTML = HTML & "<h2><input " & _
   "type='submit' name ='Action' " & _
   "value='Register Me'> <input" & _
   "type='reset' value='Clear " & _
   "Form'></h2>"
HTML = HTML & "</form>"
EntryForm = HTML
End Function
```

CREATE AN ACTIVE SERVER PAGE

Once you create the class, it's easy to write an ASP page to use this class (download Listing 3 from the free, Registered Level of DevX; see the Code Online box for details). You can try out my ASP page and take a look at the source code and complete class functions on my Web site at http://www.vbexpert.com/demo/registration.asp.

First, compile your class into an ActiveX DLL and register it on your Web server. Don't forget to copy the registration database to the server. You can use Visual InterDev or FrontPage to create the ASP page, but if you don't have those tools, simply use Notepad—it works fine.

Start your page by writing code to use the CreateObject method of the Server object to create an instance of your class. Set the DBPath property of your class to the location of the database. You can find out whether this page was called with a post method by checking the Request object's ServerVariables collection. If a form was posted to this page, call the ProcessPost procedure. Otherwise, call the EntryForm method of the class and use the Write method of the Response object to put the HTML for the form in the Web page:

```
<%
Set User = Server.CreateObject _
   ("Registration.cUser")
User.DBPath = "c:\registration.mdb"
If Request.ServerVariables _
   ("REQUEST_METHOD")="POST" Then
   Call ProcessPost
Else
   Response.Write User.EntryForm
End If
%>
```

Save this file as Registration.asp, and point to it with your browser—you should see the user registration entry form appear. Once you've got this much working, the rest is easy. The ProcessPost procedure is called when the user clicks on the Register Me button on the entry form. The logic for this procedure is straightforward. First, fill the class properties from the form using the GetFormFields procedure. This procedure, shown in Listing 3 on DevX, uses the Form collection of the Request object to retrieve the fields from the form:

```
User.Email = request.form("Email")
```

After getting the value from the form fields, the procedure checks whether the password matches what the user typed into the Confirm Password field on the form. If it doesn't match, set the HTML variable to an appropriate error message; otherwise, call the ProcessAdd procedure:

```
Sub ProcessPost()
GetFormFields
If User.Password <> request.form _
   ("PasswordC") Then
   HTML = "Password fields do not " & _
      "match - Please click the " & _
      "Back button and re-enter"
Else
   ProcessAdd
End If
End Sub
```

Finally, the ProcessAdd procedure calls a routine to load the class properties from the form fields, then calls the Add method of your user class to add the user to the database:

```
Sub ProcessAdd()
If User.Add Then
   Session("SerialNumber") = _
      User.SerialNumber
   HTML = HTML & _
      " Thank you for registering!<p>"
   HTML = HTML & _
      " Your Serial Number is: " & _
      User.SerialNumber & "<p>"
   HTML = HTML & "<a href='" & " _
      "Session("ReferPage") & "' _
      ">Finish Registration"
Else
   HTML = HTML & " User " & _
      User.UserID & " Not Added <p>" _
      & ProgVer
   HTML = HTML & "<a href='" & " _
      "Session("ReferPage") & "' _
      ">Finish Registration"
End If
End Sub
```

Download the ZIP file and start working with this registration class. Next month I'll show you how to expand this class to e-mail a user his or her serial number, along with a hyperlink to a download area on your Web site. ☒

Code Online

You can find all the code published in this issue of VBPJ on The Development Exchange (DevX) at http://www.windx.com. For details, please see "Get Extra Code in DevX's Premier Club" in Letters to the Editor.

Create a User Registration Class

Locator+ Codes

Listings for the entire issue, plus the listings for this column not printed due to space limitations, as well as the basic registration application, including the database, class, and ASP file (free Registered Level): VBPJ0498

✪ Listings for this article only, plus an enhanced registration application, including methods to update and delete user information (subscriber Premier Level) GS0498