Add E-Mail Registration to Your Server

by Chris Barlow

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

Click & Retrieve Source CODE!

Have you downloaded a software product over the Web recently? Often you're required to fill out a registration form and provide your e-mail address as your unique identifier. Companies want to capture information about users downloading their application so they can contact them in the future. Usually after entering the information, you can jump right to the download page. But how can the company be certain you've entered your e-mail address correctly? Some find it tempting to quickly enter any old name and address and get on with the download. However, more and more often, after you enter the registration information, you can't download the software immediately. Instead, the Web site displays a page telling you a code has been sent to your e-mail address with a link to the download page. This is a simple way to prove that you entered an accurate e-mail address without interfering with the download process. If you have your e-mail running, the e-mail message often arrives in your inbox within a few seconds, and you can proceed with the download.

Last month I showed you how to create a Visual Basic class that captures user-registration information into a database, then write an Active Server Page (ASP) application to use this class from a Web site. The class has properties to store user name and address information and methods to save and load the database records. You can download this class from the free, Registered Level of The Development Exchange (see the Code Online box at the end of this column for details). Use this class as a base to add a method to implement an e-mail registration application.

You need Visual Basic 5.0 to create the ActiveX DLL, and you need ActiveX Data Objects (ADO) 1.5 and Access 97 for the database access. To add the e-mail functionality, you need Active Messaging 1.1, which you can download from the Microsoft Web site at *ftp://ftp.microsoft.com/services/technet/samples/boes/bo/mailexch/exchange/appfarm/actmsg.exe*. 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 ASP pages. Start Visual Basic, load the Registration ActiveX DLL project, and change the Project name to EmailRegistration.

ADD THE SEND METHOD

Microsoft Active Messaging lets you easily send and receive e-mail from within a Visual Basic application. To send e-mail from this class, select the References menu item on the Project menu and add a reference to the Microsoft Active Messaging 1.1 Object Library.

The object model for Active Messaging is simple. The Session object is the starting point for all use of the Active Messaging components. Begin an Active Messaging session by creating a MAPI.Session object. All other objects are derived from the Session object. For example, create an outgoing message by adding a Message object to the Messages collection in the Outbox folder in the Session object.

Add this code to the Send method to dimension a Session object and a Message object, to create the MAPI Session, and to call the Logon method of the Session object. This logon automatically uses your Exchange or Outlook profile. Set up a default profile for your mail client so the logon dialogs don't appear on your Web server:

```
Public Function Send(Subject, Msg) _
   As Boolean
Dim oSession As Object
Dim oMessage As Message
```

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 http://www.VBExpert.com.

VBS

```
<script LANGUAGE="VBScript">
<!--
<%
ProgNme = "EmailRegistration.asp"
ProgVer = "1.01.10"
ProgInit = "CRB"

Dim HTML

sub ProcessPost()
  GetFormFields
  if User.Password <> request.form("PasswordC") then
     'stop and re-enter
     HTML = "Password fields do not match - " & _
        "Please click the Back button and re-enter"
  elseif request.form("Action") = "Update Me" then
     ProcessUpdate
  elseif User.Exists(User.Email) then
     'check to see if this email is valid in the system
     if User.IsValid(User.Email, User.Password) then
        ReturnDisplayForm
     else
        'stop and re-enter
        HTML = "Incorrect Password for this email " & _
           "address - Please click the Back " & _
           "button and re-enter"
     end if
  else
     ProcessAdd
  end if
end sub

sub ProcessUpdate()
  User.Modified = Now()
  if User.Update then
     Session("SerialNumber") = User.SerialNumber
     HTML = HTML & " Thank you for updating your " & _
        "information!<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 Updated <p>" & ProgVer
     HTML = HTML & "<a href='" & Session("ReferPage") _
        & "' >Finish Registration"
  end if
end sub

sub ProcessAdd()
  if User.Add then
     Session("SerialNumber") = User.SerialNumber
     HTML = HTML & " Thank you for registering!<p>"
     HTML = HTML & " Your Serial Number has been " & _
        "sent to your email address"
     Msg = "Your Serial Number is: " & _
        User.SerialNumber & vbcrlf
     Msg = Msg & "Please return this page " & _
        "and enter your serial number"
     User.Send "User Registration",Msg
     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

sub ReturnEntryForm
  HTML = User.EntryForm
end sub

sub ReturnDisplayForm
  User.Load(User.Email)
  HTML = User.DisplayForm
end sub

sub ValidateDownload
  'check Serial number and jump to download page
  Serial = Request.querystring("Serial")
  Email = request.querystring("Email")
  User.Load Email
  if User.SerialNumber = Serial then
     HTML=""
  else
     HTML = "Sorry! Invalid Serial Number!"
  end if
end sub

sub GetFormFields()
  User.Clear
  User.Email = request.form("Email")
  User.Password = request.form("Password")
  User.FirstName = request.form("FirstName")
  User.LastName = request.form("LastName")
  User.Address = request.form("Address")
  User.City = request.form("City")
  User.State = request.form("State")
  User.PostalCode = request.form("PostalCode")
  User.Country = request.form("Country")
  User.Organization = request.form("Organization")
  User.WorkNumber = request.form("WorkNumber")
  User.FaxNumber = request.form("FaxNumber")
  User.Password = request.form("Password")
  User.Comments = request.form("Comments")
end sub
%>

-->
</script>

<%
  server.scripttimeout = 999
  Set User = _
     Server.CreateObject("EmailRegistration.cUser")
  User.DBPath = "c:\registration.mdb"
  If Request.ServerVariables("REQUEST_METHOD")="POST" _
     then
     Call ProcessPost
  elseif len(Request.querystring("Serial")) > 0 then
     'check Serial number and jump to download page
     Call ValidateDownload
  else
     'save referring page
     Session("ReferPage") = _
        Request.querystring("ReferPage")
     Call ReturnEntryForm
  end if
  if len(HTML) then
     Response.Write HTML
  else
     Response.Redirect "download.asp"
  end if
%>
```

LISTING 1 ***EmailRegistration ASP Page.*** *An Active Server Page is still a real piece of source code that you or someone else will need to upgrade or debug sometime in the future. Get in the habit of specifying version information, as on line 5, so you can track the proper version.*

```
On Error Resume Next
Set oSession = _
   CreateObject("MAPI.Session")
If oSession Is Nothing Then Exit _
   Function
oSession.Logon
If Err Then Exit Function
```

Use the Add method of the Messages collection of the Outbox object to add a new outgoing message, and set the Subject and Text properties from the arguments of the Send method:

```
Set oMessage = _
   oSession.Outbox.Messages.Add
With oMessage
   .Subject = Subject
   .Text = Msg
End With
```

Finally, use the Add method of the Recipients collection of the Message object to add the Recipient object to this message, and set its Name and Type properties. Set the e-mail address from the Email property of the User class. The With…End With statement lets you work with the Recipient object without requiring you to dimension a variable to hold the Recipient object. Call the Recipient object's Resolve method to change the text name to a valid e-mail address. Send the message and log off the session:

```
With oMessage.Recipients.Add
   .Name = Email
   .Type = mapiTo
   .Resolve
End With
oMessage.Send
oSession.Logoff
Send = True
End Function
```

CREATE AN ACTIVE SERVER PAGE

Once you add the Send method to your User class, you can easily modify the Registration.asp ASP page to use the new method. Try the revised ASP page, and take a look at the source code and complete class functions on my Web site at http://www.vbexpert.com/demo/emailregistration.asp (see Listing 1).

The ASP page can create an entry form for user information, process the addition of a new user, and send an e-mail message with the new user's serial number. The ASP page also looks up an e-mail address and displays user information from the database, and it validates a user's serial number and presents a download page. The main block of code in the ASP page determines how the page was called and what step needs to be executed.

First, compile your EmailRegistration class into an ActiveX DLL, and register it on your Web server. Make sure your registration database is on the server in the proper folder.

When a browser requests the ASP page from your Web server, the server processes the main block of code within the comment marks (<%…%>) and creates the HTML page to return to the browser. From within the code block, you can call subroutines in the ASP page. Use the CreateObject method of the Server object to create an instance of your EmailRegistration.cUser class and to set the DBPath property of your class to the database location. To find out if the page was called with a post method, check the Request object's ServerVariables collection. If a form was posted to the page, you need to add a new user or look up an existing user by calling the ProcessPost procedure. If the page was called with a query string containing a serial number, you need to call the ValidateDownload procedure to make sure the serial number matches the e-mail address. Then present the page to download a file. Otherwise, call the EntryForm method of the class to create a form for the user to enter his or her registration information. Finally, use the Write method of the Response object to put the HTML for the form in the Web page:

```
<%
   server.scripttimeout = 999
   Set User = _
      Server.CreateObject _
      ("EmailRegistration.cUser")
   User.DBPath = "c:\registration.mdb"
   If Request.ServerVariables _
      ("REQUEST_METHOD")="POST" then
      Call ProcessPost
   elseif len(Request.querystring _
      ("Serial")) > 0 then
      'check serial number and jump to
      'download page
      Call ValidateDownload
   else
      'save referring page
      Session("ReferPage") = Request. _
         querystring("ReferPage")
      Call ReturnEntryForm
   end if
   if len(HTML) then
      Response.Write HTML
   else
      Response.Redirect "download.asp"
   end if
%>
```

Because you've wrapped the e-mail functionality with the cUser class of the EmailRegistration DLL, the ValidateDownload procedure is simple. Use the QueryString method of the Request object to retrieve the serial number and the e-mail address that was passed by the browser. Load the user information from the database, and check whether the serial numbers match. If they don't, display an appropriate error message:

```
sub ValidateDownload
   'check serial number and jump to
   'download page
   Serial = _
      Request.querystring("Serial")
   Email = request.querystring("Email")
   User.Load Email
   if User.SerialNumber = Serial then
      HTML=""
   else
      HTML = "Sorry! Invalid " & _
         "Serial Number!"
   end if
end sub
```

Save this file as EmailRegistration.asp, and point to it with your browser—you should see the user-registration entry form appear. Enter information for a new user, and you should see a message stating that the serial number has been sent to your e-mail address. Fire up your e-mail and click on the hyperlink in the e-mail message to return to the ASP page for validation of the download.

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.*

Add E-Mail Registration to Your Server

Locator+ Codes

Listings for the entire issue, plus a class you can use as a base to add a method to implement an e-mail registration application (free Registered Level): VBPJ0598

Listings for this article only, plus a class you can use as a base to add a method to implement an e-mail registration application, as well as the code for the e-mail registration DLL and ASP (subscriber Premier Level): GS0598