Upload Files with ASP.NET

Two ASP.NET classes remove the sting of uploading files to the Web server.

By Alok Mehta

TECHNOLOGIES

- ASP.NET
- Visual Studio .NET 2003

RESOURCE CD

- leftcol highlights
- leftcol highlights

DOWNLOAD

- Doc # xxx
- Subscribers can get this article's download at http://Advisor.com /Doc/xxx

These days, most Web applications and sites require users to upload files from the local computers to the server. You can find typical application of file upload features in content management Web sites and applications that allow customization based on the uploaded files (such as www.shutterfly.com, which prints high quality pictures of uploaded digital images).

There are several tedious solutions that let you upload files to the server after some struggle. The top two are:

File Transfer Protocol (FTP): File Transfer Protocol works fine in most cases, it often creates a discontinuity in an application user interface by requiring the user to initiate a separate Web browser session. It also can require a separate process to validate and parse the file.

Integrating third-party controls in traditional ASP: In classic ASP, file upload is an involved process in which you extract the content of the uploaded file from raw HTTP request data. In the end, many ASP programmers resort to a third-party component for file upload. Expensive third-party components are necessary to be able to receive files on the Web server.

Thanks to the extensive .NET Framework class library, file upload and download are much simpler in ASP.NET than in classic ASP. File upload in ASP.NET is as easy as retrieving an HTML form value. On the server side, ASP.NET offers this functionality by means of an HTML server control called HtmlInputFile, which manages the upload for you. On the browser side, this control behaves like the normal control you already know. In this article, I show you how to upload a file using ASP.NET using HtmlInputFile, an HTML control.

To perform file upload in ASP.NET, you must have two namespaces/classes: System.Web.UI.HtmlControls and System.Web. They're both part of System.Web.DLL. The first class represents an HTML control the user uses to select a file on the client side and upload it. The second represents the uploaded file itself, and the HtmlInputFile control obtains an instance of this class.

System.Web.UI.HtmlControls namespace

The System.Web.UI.HtmlControls class is easy to use. It consists of a collection of classes that let you create HTML server controls on a Web Forms page. HTML server controls run on the server and map directly to standard HTML tags supported by most browsers. This lets you control the HTML elements on a Web Forms page programmatically. Although System.Web.UI.HTMLControls offers several classes, you must have the HtmlInputFile class, which defines these four properties:

Accept: Comma-separated list of MIME types the user can select to upload.

MaxLength: Maximum length of the uploaded file *path*. This property has nothing to do with the uploaded file itself.

PostedFile: Returns a System.Web.HttpPostedFile object, representing the uploaded file.

Size: Width of the text box that displays the selected file. Again, this has nothing to do with the size of the uploaded file.

System.Web namespace

The System.Web class supplies classes and interfaces that enable browser-server

Alok Mehta is senior vice president and chief technology officer of AFS Technologies in Weston, Massachusetts. He recently completed his Ph.D. in Computer Science from Worcester Polytechnic Institute.

communication. This namespace includes the HTTPRequest class that provides extensive information about the current HTTP request, the HTTPResponse class that manages HTTP output to the client, and the HTTPServerUtility object that provides access to server-side utilities and processes. System.Web also includes classes for cookie manipulation, file transfer, exception information, and output cache control. Within System.Web class, you use the HttpPostedFile subclass, which represents an uploaded file. HttpPostedFile provides properties and methods to get information on an individual file and to read and save the file. You upload files in MIME multipart/form-data format and buffer them in server memory until you explicitly save them to disk. In addition, it has these four properties:

ContentLength: Length of the uploaded file.

ContentType: Content type of the uploaded file.

FileName: Client-side file path of the uploaded file. Note that browsers under Windows include the file path information, whereas browsers under Linux/Unix and Macintosh don't.

InputStream: a System.IO.Stream pointing to the uploaded file.

Try it yourself

The code in listings 1 and 2 demonstrates a Web form that's enabled to upload a file, in c:\Temp folder. (You can change this folder.) To make the code in this example work, follow these steps:

1. Open Visual Studio .NET.
2. Start a new ASP.NET Web application. You'll be prompted to name the new project; type "FileUpload".

ASP.NET creates a new project called "FileUpload". As a part of the automatic code generation feature, Visual Studio .NET creates several files within the new project.

3. Within the newly created project FileUpload, locate the file WebForm1.aspx and click on HTML tab. Copy the code from listing 1 into the HTML tab.

Listing 1: Client-side logic—??Needs caption??

```
<%@ Page Language="vb" AutoEventWireup="false"
Codebehind="WebForm1.aspx.vb"
Inherits="FileUpload.WebForm1"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML>
  <HEAD>
    <title>WebForm1</title>
    <meta name="GENERATOR" content="Microsoft Visual
    Studio.NET 7.0">
    <meta name="CODE_LANGUAGE" content="Visual Basic 7.0">
    <meta name="vs_defaultClientScript"
    content="JavaScript">
    <meta name="vs_targetSchema"

content="http://schemas.microsoft.com/intellisense/ie5">
  </HEAD>
  <body MS_POSITIONING="GridLayout">
    <%
    if page.ispostback=false then
    %>
    <form runat="server" enctype="multipart/form-data"
    ID="Form1">
    <input type="file" id="file1" runat="server"
    NAME="file1" style="Z-INDEX: 101; LEFT: 324px; POSITION:
    absolute; TOP: 117px">
    <asp:Button id="btn1" runat="server" text="Upload"
    onclick="upload" style="Z-INDEX: 102; LEFT: 325px;
    POSITION: absolute; TOP: 179px" />
<asp:Label id="Label1" style="Z-INDEX: 103; LEFT: 327px;
POSITION: absolute; TOP: 72px" runat="server"
Width="206px"
Height="32px">VB.Net File Upload Example</asp:Label>
    <asp:Label id="message" style="Z-INDEX: 104; LEFT:
    328px; POSITION: absolute; TOP: 151px" runat="server"
    Width="231px"></asp:Label>
    </form>
    <%
    end if
    %>
  </body>
</HTML>
```

4. In the project FileUpload, locate the file WebForm1.aspx.vb. Copy the code from listing 2 into this tab.

Listing 2: Server side logic—caption needed.

```
Imports System.IO

Public Class WebForm1

  Inherits System.Web.UI.Page

  'Bind the UI to Events

  Protected WithEvents btn1 As _
     System.Web.UI.WebControls.Button 'Button
  Protected WithEvents Label1 As _
     System.Web.UI.WebControls.Label 'Label
  Protected WithEvents message As _
     System.Web.UI.WebControls.Label 'Message
  Protected WithEvents file1 As _
   System.Web.UI.HtmlControls.HtmlInputFile
   'HTML INPUT FILE

#Region " Web Form Designer Generated Code "

  'This call is required by the Web Form Designer.
  <System.Diagnostics.DebuggerStepThrough()> Private Sub _
     InitializeComponent()

  End Sub

  Private Sub Page_Init(ByVal sender As System.Object, _
     ByVal e As System.EventArgs) Handles MyBase.Init
    InitializeComponent()
  End Sub

#End Region

  Public Sub Upload( _
     ByVal sender As Object, ByVal e As System.EventArgs)
'Define a constant path name which can easily be changed
  Const savePath = "C:\Temp\"

    If Not (file1.PostedFile Is Nothing) _
       Then 'do nothing if the filename  is blank
    Try
      Dim postedFile = file1.PostedFile 'HTMLINputFile
      Dim filename As String = _
         Path.GetFileName(postedFile.FileName)
         'Get FileName
      Dim contentType As String = _
         postedFile.ContentType 'Get Type
      Dim contentLength As Integer = _
         postedFile.ContentLength ' Get Length

      postedFile.SaveAs(savePath & filename) 'Save File
      'Create Message
      message.Text = postedFile.Filename & " uploaded" & _
         "<br>content type: " & contentType & _
         "<br>content length: " & _
         contentLength.ToString()
      Catch exc As Exception
        message.Text = "Failed uploading file"
      End Try
    End If

  End Sub
End Class
```

Continued

5. Save the project.
6. Using the menu "Build Solution," build the solution. Assuming Windows 2000, .NET, and Internet Information Server (IIS) are configured properly, this step creates a Web site called "FileUpload" on your local Web server (localhost). This new site lets you upload a file by typing http://localhost/FileUpload/WebForm1.aspx in the Web browser.

Listing 1 shows the user interface or the client-side logic. It contains these controls:

- Web form named "Form1".
- System.Web.UI.HtmlControls.HtmlInputFile control named "File1" for selecting a file to upload.
- Button control named "Btn1" to submit the upload. The button is bound to the Upload subroutine. In other words, pressing "Btn1" calls the Upload routine, which resides in WebForm1.aspx.vb.
- Label control named "Message" to display a message from the server (either an error message, if the upload fails, or file information, if the upload succeeds).

The server-side logic is shown in listing 2.

Using events, the user interface controls Btn1, Lable1, Message, and File1 are bound to their respective **??something missing??**. VS.NET automatically generates this code.

The statement <asp:Button id="btn1" runat="server" text="Upload" onclick="upload"> triggers the call to the subroutine Upload when Btn1 is pressed. Sub Upload is the most important part of the project. Here's how it works: If the filename you want to upload is blank, the file isn't uploaded. Using various properties and methods of HTMLInputFile control and the IO namespace library, you can determine the uploaded file's properties such as path, name, content, and length. Using SaveAs method of the HTMLInputFile class, the file is saved in c:\Temp\ folder. You can easily save this file into a designated user folder or some other place. The exception handler lets you capture any errors.

Wrap up

ASP.NET gives you an easy way to upload the files to the Web server using HTMLInputFile and HTTPPostedFile. When saving the file to the server, remember that the ASP.NET account on the Web server has the WRITE permission to the folder where you're saving the file. Moreover, the Web server, by default, has a limit of 4MB as uploading size of the file. If you want to upload a larger file, you'll have to change the machine.config files httpRuntime settings. You can easily modify the code in this article to enhance multiple file uploading. ADVISOR