ASP.NET: Uploading an Image and creating a thumbnail

During the course of this tutorial you will learn how to provide users with the facility to upload files to a remote server using an HTML form; specifically this tutorial will focus on uploading image files to an Images directory and saving a proportionally-sized thumbnail of the uploaded images in a Thumbs directory, the code presented here is able to identify if an image is landscape or portrait and adjust the dimensions of the generated thumbnail accordingly. This is a very common scenario on a typical e-commerce site where a product list page would show the thumbnail images and the detailed product info page would show the larger image. This tutorial demonstrates how to code the ASP.NET for uploading gif and jpg files.

In a production environment, where you want support for GIF, JPG, PGM, PNG, TGA, TIFF, PCX, or BMP files, or allow users to select compression settings, or enable user-friendly error messages, DMXzone have an extension called Smart Image Processor.NET which is highly recommended.

The HTML portion of the page

The HTML portion of the example page is pretty simple, it consists of an HTML form with a **runat="server"** attribute defined and its **enctype** attribute set to **multipart/form-data**, which tells the browser that its is to post the form data as binary data rather that plain text. Within the form is an input field with a type of **file**, which renders a text field equipped with a browse button that allows the end user to browse and select a file on their own computer that should be uploaded. This file field also has its **runat** attribute set to server to make it visible to our server side code. Finally, the form also contains an **asp:Button** control that renders the submit button for the form. All this is shown in the following code block.

```
<HTML>
 <HEAD>
 <title>Image Upload and thumbnail maker</title>
 </HEAD>
 <body>
 <form method="post" enctype="multipart/form-data" name="form1" id="form1"
runat="server">
  <p>
  <input name="File1" type="file" id="File1" runat="server">
  </p>
  <p>
  <asp:Button ID="Button1" Text="Upload" runat="server" />
  </p>
 </form>
 </body>
</HTML>
```

The Server Side Code

Here's the server side code responsible for saving the uploaded image and generating the thumbnail image:

```
C# (UploadCsharp.aspx)
void Page_Load(object sender, System.EventArgs e)
{
 if(Page.IsPostBack && File1.PostedFile != null)
 {
 SaveImage();
 }
}

void SaveImage()
```

```
{
 System.IO.FileInfo F = new System.IO.FileInfo(File1.PostedFile.FileName);
 if(F.Extension.ToLower() == ".gif" || F.Extension.ToLower() == ".jpg")
 {
 File1.PostedFile.SaveAs(Request.PhysicalApplicationPath + "\\Images\\" + F.Name);
 MakeThumbNail(F, 150, 80);
 }
 else
 {
 //warn user only gif or jpg files are allowed
 }
}

void MakeThumbNail(System.IO.FileInfo F, double MaxWidth, double MaxHeight)
{
 System.Drawing.Image OriginalImg = ¬
System.Drawing.Image.FromStream(File1.PostedFile.InputStream);
 System.Drawing.Size TheSize = new System.Drawing.Size(OriginalImg.Width, ¬
OriginalImg.Height);

 double sizer = 1;

 while((MaxWidth > -1 && TheSize.Width > MaxWidth) || (MaxHeight > -1 && ¬
TheSize.Height > MaxHeight))
            {
 if(MaxWidth > -1 && TheSize.Width > MaxWidth)
 {
  sizer = MaxWidth / TheSize.Width;
  TheSize.Width = Convert.ToInt32(TheSize.Width * sizer);
  TheSize.Height = Convert.ToInt32(TheSize.Height * sizer);
 }
 if(MaxHeight > -1 And TheSize.Height > MaxHeight)
 {
  sizer = MaxHeight / TheSize.Height;
  TheSize.Width = Convert.ToInt32(TheSize.Width * sizer);
  TheSize.Height = Convert.ToInt32(TheSize.Height * sizer);
 }
 }

 string SavePath = Request.PhysicalApplicationPath + "\\Thumbs\\" + F.Name;

 System.Drawing.Bitmap NewImg = new System.Drawing.Bitmap(OriginalImg, TheSize);
 OriginalImg.Dispose();

 if(System.IO.File.Exists(SavePath)){System.IO.File.Delete(SavePath);}

 switch(F.Extension.ToLower())
 {
 case ".jpg":
  NewImg.Save(SavePath, System.Drawing.Imaging.ImageFormat.Jpeg);
  break;
 case ".gif":
  NewImg.Save(SavePath, System.Drawing.Imaging.ImageFormat.Gif);
  break;
 }

 NewImg.Dispose();
```

```
}

VB (UploadVB.aspx)
Sub Page_Load(Sender As Object, e As System.EventArgs)
 If Page.IsPostBack And Not File1.PostedFile Is Nothing Then
 SaveImage()
 End If
End Sub

Sub SaveImage()
 Dim F As New System.IO.FileInfo(File1.PostedFile.FileName)
 If F.Extension.ToLower() = ".gif" Or F.Extension.ToLower() = ".jpg" Then
 File1.PostedFile.SaveAs(Request.PhysicalApplicationPath & "\Images\" + F.Name)
 MakeThumbNail(F, 150, 80)
 Else
 'warn user only gif or jpg files are allowed
 End If
End Sub

Sub MakeThumbNail(F As System.IO.FileInfo, MaxWidth As Double, MaxHeight As Double)
 Dim OriginalImg As System.Drawing.Image = ¬
System.Drawing.Image.FromStream(File1.PostedFile.InputStream)
 Dim TheSize As new System.Drawing.Size(OriginalImg.Width, OriginalImg.Height)

 Dim sizer As Double = 1

 Do While (MaxWidth > -1 And TheSize.Width > MaxWidth) Or (MaxHeight > -1 And ¬
TheSize.Height > MaxHeight)
 If MaxWidth > -1 And TheSize.Width > MaxWidth Then
  sizer = MaxWidth / TheSize.Width
  TheSize.Width = Convert.ToInt32(TheSize.Width * sizer)
  TheSize.Height = Convert.ToInt32(TheSize.Height * sizer)
 End If
 If MaxHeight > -1 And TheSize.Height > MaxHeight Then
  sizer = MaxHeight / TheSize.Height
  TheSize.Width = Convert.ToInt32(TheSize.Width * sizer)
  TheSize.Height = Convert.ToInt32(TheSize.Height * sizer)
 End If
 Loop

 Dim SavePath As String = Request.PhysicalApplicationPath & "\Thumbs\" & F.Name

 Dim NewImg As New System.Drawing.Bitmap(OriginalImg, TheSize)
 OriginalImg.Dispose()

 If System.IO.File.Exists(SavePath) Then System.IO.File.Delete(SavePath)

 Select Case F.Extension.ToLower()
 Case ".jpg"
  NewImg.Save(SavePath, System.Drawing.Imaging.ImageFormat.Jpeg)
 Case ".gif"
  NewImg.Save(SavePath, System.Drawing.Imaging.ImageFormat.Gif)
 End Select

 NewImg.Dispose()

End Sub
```

Now we will break the code down into individual lines and explain the purpose of each one.

C#
```
void Page_Load(object sender, System.EventArgs e)
```
VB
```
Sub Page_Load(Sender As Object, e As System.EventArgs)
```
This line is the procedure body of the **Page_Load** method which runs every time the page loads, any code placed within the Page_Load method is executed when the page loads.

C#
```
if(Page.IsPostBack && File1.PostedFile != null)
```
VB
```
If Page.IsPostBack And Not File1.PostedFile Is Nothing Then
```
This line is the first line within the Page_Load method, it checks two things, first it checks if the page **Page.IsPostBack** equals true indicating that the page load was the result of the form being submitted, secondly it checks that a file has been uploaded, if no file is uploaded the **File1.PostedFile** will equal **null**(**Nothing** in VB)

C#
```
SaveImage();
```
VB
```
SaveImage()
```
If both the checks performed by the previous line return true indicating that a file has been uploaded this line gets executed, this line simply calls the **SaveImage** method which we will look at next.

C#
```
void SaveImage()
```
VB
```
Sub SaveImage()
```
This line is the procedure body of the **SaveImage** method which contains the code used to save the uploaded image file to the server's file system.

C#
```
System.IO.FileInfo F = new System.IO.FileInfo(File1.PostedFile.FileName);
```
VB
```
Dim F As New System.IO.FileInfo(File1.PostedFile.FileName)
```
This line creates a new instance of the **FileInfo** class called **F**; the path of the uploaded image (**File1.PostedFile.FileName**) is passed as a parameter of the FileInfo class constructor. The FileInfo class is a useful tool that can be used to extract the file name from the full path and also get the file extension without us having to parse the information from the string directly.

For more information on the FileInfo class and other file system access tips see my tutorial Working with the Server's file system using ASP.NET.

C#
```
if(F.Extension.ToLower() == ".gif" || F.Extension.ToLower() == ".jpg")
```
VB

```
If F.Extension.ToLower() = ".gif" Or F.Extension.ToLower() = ".jpg" Then
```

this line used the F FileInfo class to check if the uploaded files extension matches **.gif** or **.jpg** indicating that the user has uploaded a *.gif or *.jpg file, we must perform this check to avoid errors that would arise if the user was to upload a different type of file. This example is constrained to supporting *.gif and *.jpg image types but you can add other types such as *.png if required, as long as the type is a supported image format of the .NET Framework, the following are natively supported:

Extension	Description
EMF	Enhanced Windows metafile.
Exif	Exchangeable Image Format.
GIF	Graphics Interchange Format.
ico	Windows icon.
JPEG	Joint Photographic Experts Group.
PNG	W3C Portable Network Graphics.
TIFF	Tag Image File Format.
WMF	Windows metafile.

C#
```
File1.PostedFile.SaveAs(Request.PhysicalApplicationPath + "\\Images\\" + F.Name);
```

VB
```
File1.PostedFile.SaveAs(Request.PhysicalApplicationPath & "\Images\" + F.Name)
```

This line saves the uploaded image to a directory called **Images** that should already exist in the site root - an error will occur if the directory does not exist. The physical path to the root of the site is obtained with **Request.PhysicalApplicationPath** and then the directory and image file name are appended to this value. Your site must be defined an IIS Application for this to work correctly.

C#
```
MakeThumbNail(F, 150, 80);
```

VB
```
MakeThumbNail(F, 150, 80)
```

This line called the **MakeThumbNail** method which creates and saves the thumbnail; we will look at this method shortly. Three parameters are passed with this call: the **F FileInfo** class, and the maximum width and height that the thumbnail can be. if you wish to ensure the width is constrained to a maximum possible size but are not concerned what size the height ends up at you can simply pass **-1** as the third parameter and the height will be resized proportionally with the height with no further checking performed. The same can be done to ignore the width value and have the constraints applied to the height by passing **-1** as the second parameter. If **-1** is passed for both dimensions the thumbnail will be the same size as the original image.

C#
```
else
{
 //warn user only gif or jpg files are allowed
}
```

VB
```
Else
 'warn user only gif or jpg files are allowed
End If
```

these lines are used to optionally provide feedback to the user when they upload a non *.gif or *.jpg file, the example code does not show feedback, as you may wish to Response.Write, set a Labels text or redirect to a different page in this scenario.

```
C#
void MakeThumbNail(System.IO.FileInfo F, double MaxWidth, double MaxHeight)

VB
Sub MakeThumbNail(F As System.IO.FileInfo, MaxWidth As Double, MaxHeight As Double)
```

This line is the procedure body of the **MakeThumbNail** method, this method contains the code used to generate and save the thumbnail file to the server's file system.

```
C#
System.Drawing.Image OriginalImg = ¬
System.Drawing.Image.FromStream(File1.PostedFile.InputStream);

VB
Dim OriginalImg As System.Drawing.Image = ¬
System.Drawing.Image.FromStream(File1.PostedFile.InputStream)
```

This line creates a new instance of the **System.Drawing.Image** class called **OriginalImg**; the binary image data passed to the Image class constructor is the **File1.PostedFile.InputStream** which returns the binary date representing the uploaded image which is still held in the server's memory.

```
C#
System.Drawing.Size TheSize = new System.Drawing.Size(OriginalImg.Width, ¬
OriginalImg.Height);

VB
Dim TheSize As new System.Drawing.Size(OriginalImg.Width, OriginalImg.Height)
```

This line creates a new instance of the **System.Drawing.Size** class called **TheSize** which is used to store width and height values used for resizing image files, the width and height of the OriginalImg image are passed into the class constructor as the starting size.

```
C#
double sizer = 1;

VB
Dim sizer As Double = 1
```

This line declares a variable called **sizer** which will be used during the resize procedure, the default value of this variable is set to 1.

```
C#
while((MaxWidth > -1 && TheSize.Width > MaxWidth) || (MaxHeight > -1 && TheSize.Height
> MaxHeight))

VB
Do While (MaxWidth > -1 And TheSize.Width > MaxWidth) Or (MaxHeight > -1 And
TheSize.Height > MaxHeight)
```

this line begins a loop that will repeated while the width or height of the TheSize class is more than the MaxWidth or MaxHeight values respectively, if -1 is passed for the MaxWidth or MaxHeight the associated dimension is ignored.

```
C#
if(MaxWidth > -1 && TheSize.Width > MaxWidth)

VB
If MaxWidth > -1 And TheSize.Width > MaxWidth Then
```

This line checks to see if the MaxWidth value is more than -1, if it is the Width of the TheSize class is checked to see if it's over the MaxWidth value.

```
C#
sizer = MaxWidth / TheSize.Width
TheSize.Width = Convert.ToInt32(TheSize.Width * sizer);
TheSize.Height = Convert.ToInt32(TheSize.Height * sizer);

VB
sizer = MaxWidth / TheSize.Width
TheSize.Width = Convert.ToInt32(TheSize.Width * sizer)
TheSize.Height = Convert.ToInt32(TheSize.Height * sizer)
```

if the Width of the TheSize class is over the MaxWidth value these lines are then executed. The first sets the sizer variable to the result of the MaxWidth value divided by the Width of the TheSize class, the next line sets the Width of the TheSize class to the result of multiplying the current Width of the TheSize class by the value of the sizer variable. Finally the Height of the TheSize class is set to the result of multiplying the current Width of the TheSize class by the value of the sizer variable converted to an Int32 value (a whole number). This process brings the width down to the MaxWidth value and brings the height down proportionally.

```
C#
if(MaxHeight > -1 And TheSize.Height > MaxHeight)

VB
If MaxHeight > -1 And TheSize.Height > MaxHeight Then
```

Now that the width has been checked and sized if necessary, the MaxHeight is checked to see if it's more than -1 and if it is the Height of the TheSize class is checked to ensure is not more than the MaxHeight value.

```
C#
sizer = MaxHeight / TheSize.Height;
TheSize.Width = Convert.ToInt32(TheSize.Width * sizer);
TheSize.Height = Convert.ToInt32(TheSize.Height * sizer);

VB
sizer = MaxHeight / TheSize.Height
TheSize.Width = Convert.ToInt32(TheSize.Width * sizer)
TheSize.Height = Convert.ToInt32(TheSize.Height * sizer)
```

If the height of the TheSize class is more than the MaxHeight value the sizer variable is set to the result of the MaxHeight divided by the current height of the TheSize class. Next, the Width of the TheSize class is set to the result of multiplying the Width of the TheSize class with the sizer variable converted to an Int32. Finally, the Height of the TheSize class is set to the result of multiplying the Height of the TheSize class with the sizer variable converted to an Int32. By resizing both dimensions each time, the proportions of the image will be retained.

```
C#
string SavePath = Request.PhysicalApplicationPath + "\\Thumbs\\" + F.Name;

VB
Dim SavePath As String = Request.PhysicalApplicationPath & "\Thumbs\" & F.Name
```

This line sets a string variable called **SavePath** to the result of calling **Request.PhysicalApplicationPath** to get the physical path to the site root and adding the directory name (Thumbs) and then the file name (**F.Name**) onto the end. This will result in the full path to the location the thumbnail should be saved at.

```
C#
System.Drawing.Bitmap NewImg = new System.Drawing.Bitmap(OriginalImg, TheSize);
```

```
VB
Dim NewImg As New System.Drawing.Bitmap(OriginalImg, TheSize)
```

This line creates a new instance of the **System.Drawing.Bitmap** class called **NewImg**. The parameters passed in the class constructor are the Image class **OriginalImg** that was created earlier, and the TheSize class we just re-dimensioned.

```
C#
OriginalImg.Dispose();

V
OriginalImg.Dispose()
```

This line purges the **OriginalImg** class from memory by calling its **Dispose** method as we have no further use for it now.

```
C#
if(System.IO.File.Exists(SavePath)){System.IO.File.Delete(SavePath);}

V
If System.IO.File.Exists(SavePath) Then System.IO.File.Delete(SavePath)
```

This line uses the **Exists** method of the **System.IO.File** class to check if the thumbnail image already exists. If it exists, it's deleted using the Delete method of the System.IO.File class. The reason for this is that when we later save the new thumbnail, an error will result if the file already exists.

```
C#
switch(F.Extension.ToLower())

V
Select Case F.Extension.ToLower()
```

This line begins a conditional **switch/Select Case** branch structure that contains one case element for each possible condition; here we are saying select the case element that matches the files extension converted to lower case.

```
C#
case ".jpg":
 NewImg.Save(SavePath, System.Drawing.Imaging.ImageFormat.Jpeg);
 break;

V
Case ".jpg"
 NewImg.Save(SavePath, System.Drawing.Imaging.ImageFormat.Jpeg)
```

These lines represent a case element which comprises of the case label, **.jpg** in this case (no pun intended), and the code to execute if this elements label matches the switch/select case condition. If this element matches the condition the thumbnail is saved as a *.jpg format using the **Save** method of the **NewImg** class object, passing the path to save the file to and the format to encode the image file (in this case **System.Drawing.Imaging.ImageFormat.Jpeg)** as parameters. The C# code for the case element ends with the break statement; this causes the code to pass over the rest of the case elements without executing their code, Visual Basic does not require this as it always jumps to the **End Select** after a case element has been processed.

```
C#
case ".gif":
 NewImg.Save(SavePath, System.Drawing.Imaging.ImageFormat.Gif);
 break;
```

```
VB
Case ".gif"
 NewImg.Save(SavePath, System.Drawing.Imaging.ImageFormat.Gif)
```

This is the case element for when the uploaded file is a ***.gif** file, in this case the thumbnail is saved as a *.gif file. This means the user can upload a *.jpg or *.gif file and you know the thumbnail will always be the same format as the uploaded image.

```
C#
NewImg.Dispose();

VB
NewImg.Dispose()
```

Now that the thumbnail has been saved, we no longer need the NewImg class object, so we call its Dispose method to reclaim the memory is using.

Summary

During the course of this tutorial you learned how to upload files using an HTML form, you also learned how to save the uploaded file to the servers file system, next you learned how to load an image into memory and resize it based on pre determined maximums, finally you learned how to save the new image to the server's file system.