Taking it to the Limit with Pure ASP upload 3.0: Multiple Images

In the first tutorial, "Taking it to the Limit with Pure ASP upload 3.0: Image Update Options" we learned to handle a single image in a form with text and image fields' In this tutorial delete multiple images—or handle a single on a page with multiple images, and set the database field or fields to null by toggling the skip empty fields feature of Pure ASP Upload 3. In this section we will learn to handle multiple images—including multiple version of the same image, e.g. thumbs and large views on a page which updates images and text fields and on a page with images only while setting the database fields to null and either skipping the empty field, or updating it when a new image is uploaded.

The methods presented in this tutorial, will help you take your web applications to the limit in power, performance, and efficiency.

Assumptions

Although this tutorial is part of a series, it stands on its own, though you may have a bit more trouble keeping up than you would if you read the first of the series. It assumes you know how to create a record set to populate your form with existing data, know how to apply the Dreamweaver Update Server Behaviour, and have some familiarity with the **file scripting object**. If you have never configured a connection string, created a record set, populated a form with data, applied an insert or update behaviour and successfully run the page then this tutorial is not for you. Of course you also need to know how to apply the Pure ASP Upload behaviour. The first thing we do is set up our form. So let's get started!

Tip: The File Scripting Object--often abbreviated fso--allows us to access the server file system.

Form Modifications

Since we're dealing with images it seems a good idea to display the existing image on the update page so the user can make a decision as to whether or not to replace or remove the image. To do this we will create a conditional region. The conditional region will display two elements: If the database image field has a value other than null, the first will be image itself. The second element is the **Delete Image?** check box. The conditional region will do two things. If there is an image, it will be displayed along with the Delete Image? checkbox. If there is no image, the message **No existing image** will be displayed. If the record set image field has an image specified and no image is found, then the dreaded red X will appear. On the front of end of the application, I prefer to use the **file scripting object** to determine if the specified image exists, before displaying it, thus avoiding the red X when the image cannot be found. You will be learning to use the file scripting object into today's tutorial.

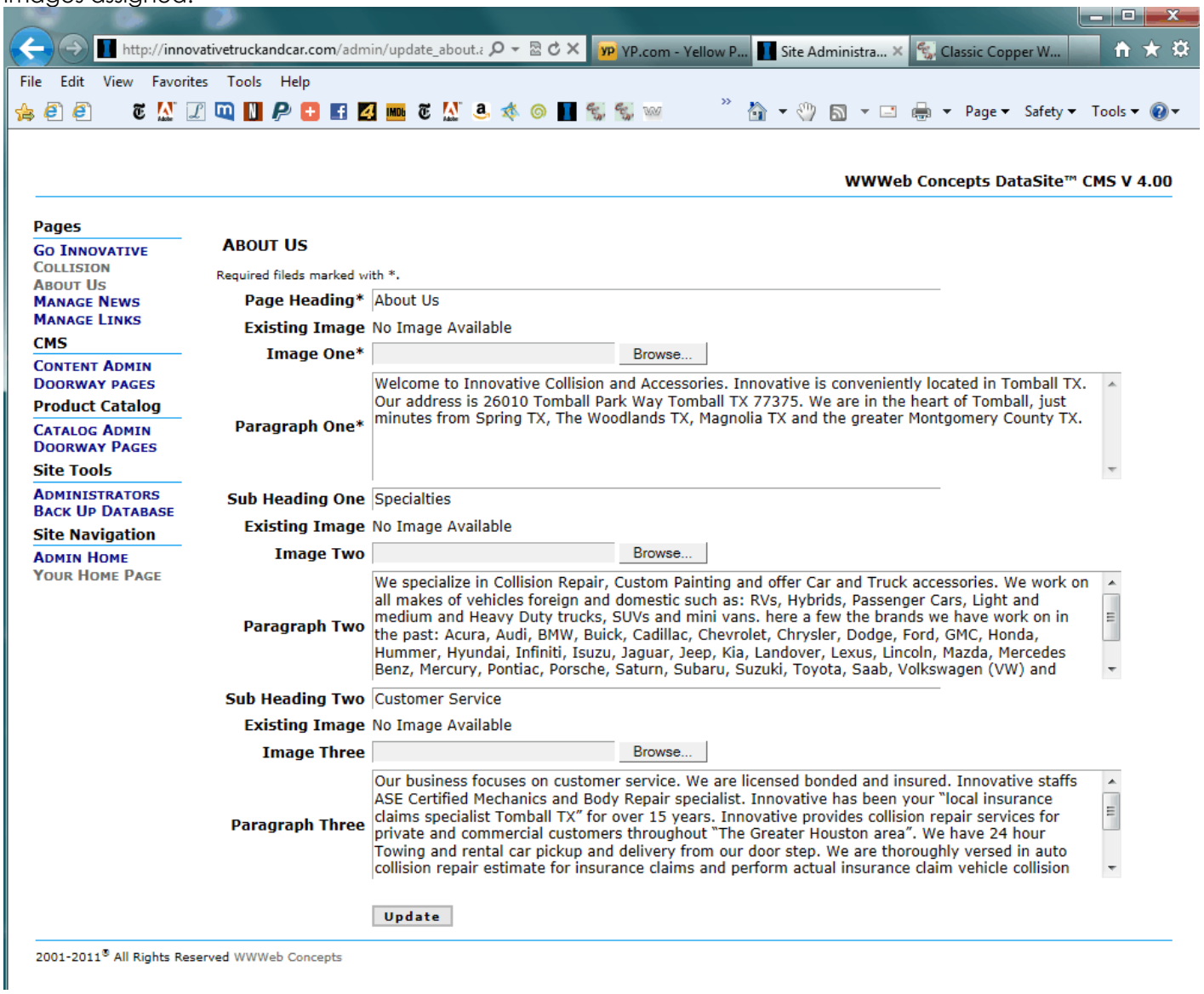
If you read the first tutorial, this will be familiar. We're going to display the existing images and a form check box to replace or remove the image in a conditional region. The conditional region will do two things. If there is an image, it will be displayed along with the form checkbox: **Delete Image?** If there is no image, the message **No existing image** will be displayed. If the record set image field has an image specified and no image is found, a red X will appear to alert the user of the missing image. (I prefer to use the **file scripting object** for displaying images on the site front end. See my tutorial, "Taking it to the Limit with Pure ASP upload 3.0: Image Update Options" for this technique.) This time, however, we are going to name our form fields in a numerical sequence: e.g. Delete_Image1, Delete_Image2, Delete_Image3. The page also has fields for a page heading, two page subheadings, and three paragraphs of text.

```
<tr>
  <td>
   <% If (rs_links.Fields.Item("P_Image").Value)<> "" Then%>
   <img src="../site_images/<%=(rs_page.Fields.Item("P_Image").Value)%>" alt="Existing
Image" /></td>
```

```
</tr>
<tr>
  <td colspan="2">Delete Image?</td>
  <td><input name="Delete_Image1" type="checkbox" id="Delete_Image1" value="Y" />
   <% Else%>No Image Available
   <% End If %>
  </td>
</tr>
```

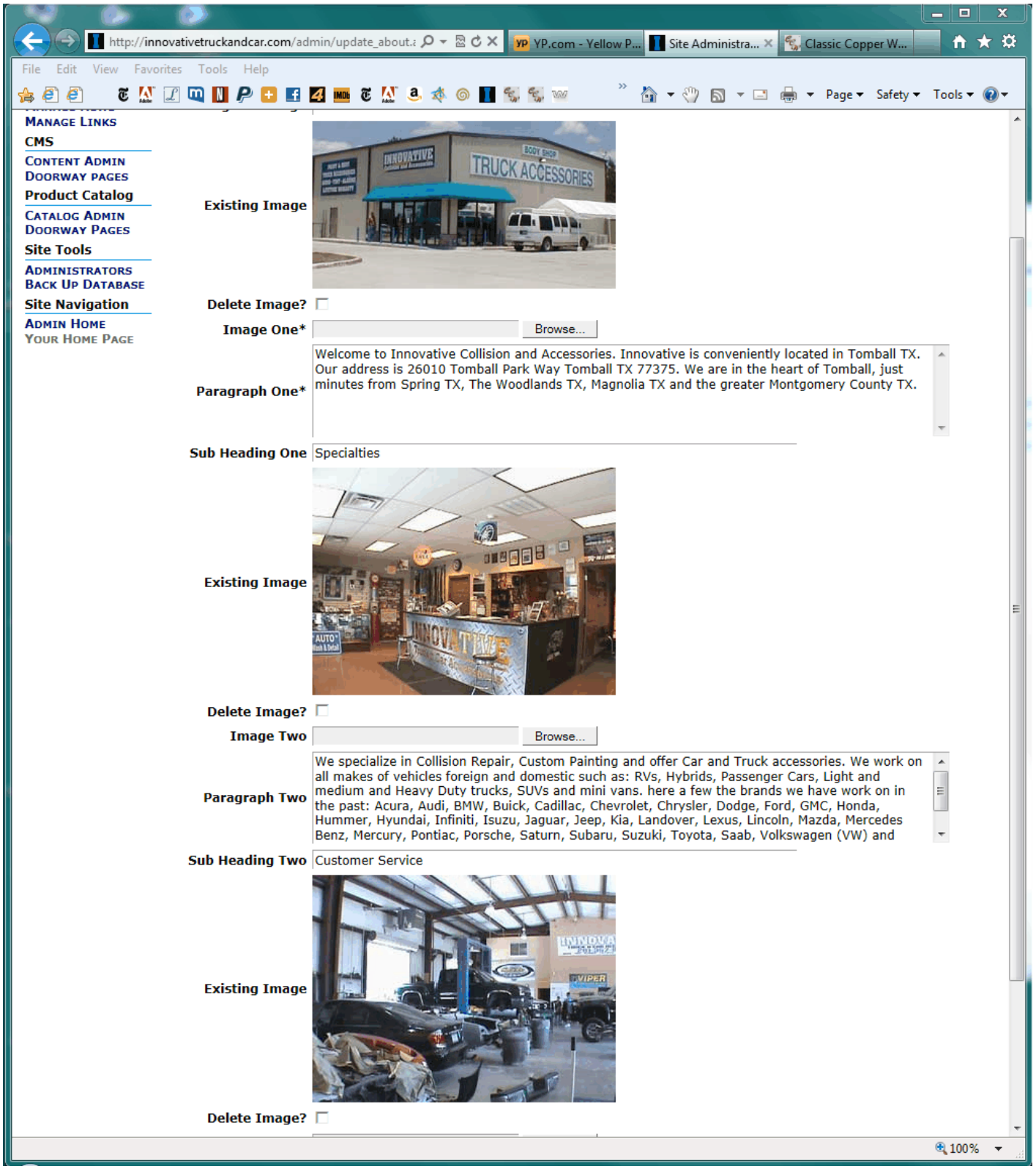
Form code block

Now that you have created your three conditional regions, your page should look like this when there are no images assigned.



Page without images 1

If you have already assigned images it should look like this.



Page with images 1

Toggling the Skip Empty Fields Feature

If you've read the first tutorial, this will be familiar. This time, however, we will be toggling multiple fields. First we need to apply the Dreamweaver update server behaviour and Pure ASP Upload. Next we apply the Pure ASP Upload 3.0 behaviour, and finally we're ready to customize the code. I'll try to keep the recap as brief as possible so those of you who have read the first tutorial don't spend too much time in familiar territory. At the same time, I will do my best to bring those who haven't read the first tutorial up to speed.

In the first tutorial I had readers switch to code view before they save the page after applying the pure upload behaviour. This is because pure ASP upload must modify the Dreamweaver update server behaviour. It uses the **generic Request()** Collection and the **Request.Form()** collection which cannot be used after calling a binary upload. Instead pure ASP upload uses **UploadFormRequest()** to get the necessary values.

Tip: The generic Request() and the Request.Form() collections cannot be called after calling a binary upload.

You should notice the update parameter or your image fields has changed. Pure ASP upload uses the **MM_IIF** to implement for the skip empty fields feature. As you may recall the MM_IIF enough returns one of two values true or false to identify empty fields. This is the heart of the skip empty fields feature. Basically, the image field parameters use the image form field value if an image has been selected. Otherwise, it uses the hidden field **upload_org_yourimagefieldname** created by the Pure ASP Upload extension. This contains the original image field value which is substituted for the empty upload form field, thus skipping the empty field. These are the parameters we going to use to toggle the skip empty fields feature. We do this by creating two parameters: one which skips the empty field using the MM_IIF and one that does not skip the empty field using the original Dreamweaver created parameter modified for upload. With an **if… then… else** statement and the "Delete Image?" check boxes as switches, we can control the input.

Tip: You'll find the MM_IIF implementation in the code block immediately above the Dreamweaver update server behaviour.

Now that we know what we are going to do, let's get to it. The modified Dreamweaver update server behaviour appears in the code block below. In this case the name of my image fields is named **P_Image(i)** where i is an integer. You should notice that my image fields are named in a numeric sequence. This becomes useful when we delete the images. I've commented the code before and after each of the toggle sections so you can easily identify them.

```
<%
If (CStr(UploadFormRequest("MM_update")) = "UpdateForm") Then
  If (Not MM_abortEdit) Then
    ' execute the update
    Dim MM_editCmd

    Set MM_editCmd = Server.CreateObject ("ADODB.Command")
    MM_editCmd.ActiveConnection = MM_datasource_STRING
    MM_editCmd.CommandText = "UPDATE About SET P_Heading1 = ?, P_Image1 = ?,
P_Paragraph1 = ?, P_Heading2 = ?, P_Image2 = ?, P_Paragraph2 = ?, P_Heading3 = ?,
P_Image3 = ?, P_Paragraph3 = ? WHERE P_ID = ?"
    MM_editCmd.Prepared = true
    MM_editCmd.Parameters.Append MM_editCmd.CreateParameter("param1", 202, 1, 50,
UploadFormRequest("P_Heading1")) ' adVarWChar

      'Toggle skip empty field behavior to remove existing image
      If CStr(UploadFormRequest("Delete_Image1")) <> "" Then
    MM_editCmd.Parameters.Append MM_editCmd.CreateParameter("param2", 202, 1, 100,
UploadFormRequest("P_Image1")) ' adVarWChar
```

```
      Else
      MM_editCmd.Parameters.Append MM_editCmd.CreateParameter("param2", 202, 1, 100,
MM_IIF(UploadFormRequest("P_Image1"), UploadFormRequest("P_Image1"),
UploadFormRequest("upload_org_P_Image1"))) ' adVarWChar
      End If
      'End toggle

    MM_editCmd.Parameters.Append MM_editCmd.CreateParameter("param3", 203, 1,
536870910, UploadFormRequest("P_Paragraph1")) ' adLongVarWChar
    MM_editCmd.Parameters.Append MM_editCmd.CreateParameter("param4", 202, 1, 50,
UploadFormRequest("P_Heading2")) ' adVarWChar

      'Toggle skip empty field behavior to remove existing image
      If CStr(UploadFormRequest("Delete_Image2")) <> "" Then
            MM_editCmd.Parameters.Append MM_editCmd.CreateParameter("param5", 202, 1,
100, UploadFormRequest("P_Image2")) ' adVarWChar
      Else
      MM_editCmd.Parameters.Append MM_editCmd.CreateParameter("param5", 202, 1, 100,
MM_IIF(UploadFormRequest("P_Image2"), UploadFormRequest("P_Image2"),
UploadFormRequest("upload_org_P_Image2"))) ' adVarWChar
      End If
      'End toggle

    MM_editCmd.Parameters.Append MM_editCmd.CreateParameter("param6", 203, 1,
536870910, UploadFormRequest("P_Paragraph2")) ' adLongVarWChar
    MM_editCmd.Parameters.Append MM_editCmd.CreateParameter("param7", 202, 1, 50,
UploadFormRequest("P Heading3")) ' adVarWChar

      'Toggle skip empty field behavior to remove existing image
      If CStr(UploadFormRequest("Delete_Image2")) <> "" Then
      MM_editCmd.Parameters.Append MM_editCmd.CreateParameter("param8", 202, 1, 50,
UploadFormRequest("P_Image3")) ' adVarWChar
      Else
      MM_editCmd.Parameters.Append MM_editCmd.CreateParameter("param8", 202, 1, 50,
MM_IIF(UploadFormRequest("P_Image3"), UploadFormRequest("P_Image3"),
UploadFormRequest("upload_org_P_Image3"))) ' adVarWChar
      End If
      'End toggle

      MM_editCmd.Parameters.Append MM_editCmd.CreateParameter("param9", 203, 1,
536870910, UploadFormRequest("P_Paragraph3")) ' adLongVarWChar
    MM_editCmd.Parameters.Append MM_editCmd.CreateParameter("param10", 5, 1, -1,
MM_IIF(UploadFormRequest("MM_recordId"), UploadFormRequest("MM_recordId"), null))
' adDouble
```

The line of code after the **Then** uses the Dreamweaver generated parameter code modified for upload. It does not use the MM_IIF implementation and does not skip the empty field. The line of code after the **Else** uses the parameter as modified by the pure ASP upload extension with the MM implementation and skips the empty field, using the value from the hidden field upload_org_P_Imagei), which as you recall holds the value of the original image. Now all we have to do is close the If… Then… Else I statement with the final **End If** and repeat the process for each image field.

Tip: The parameters both have the same name: param3 in the example. This is because only one parameter is used in the SQL statement depending upon the condition of the Delete Image? check box.

Now that we set up the code to toggle the skip empty fields feature all we have to do is handle the image delete's.

Deleting the Image

We need two functions from Marcellino Bommezijn's tutorial on deleting images when updating or deleting a record. The first function, **newFileSystemObject()**, creates the file scripting object, and the second function **fileExists** uses the newly created file scripting object to determine if the file exists. I usually place the functions in an include file with other miscellaneous functions. You can also place them in the page, provided it's before you call them. Immediately above the MM_IIF Implementation is a good place.

```
<%
Function newFileSystemObject()
set newFileSystemObject=Server.CreateObject("Scripting.FileSystemObject")
End Function

Function fileExists(aFileSpec)
fileExists=newFileSystemObject.fileExists(aFileSpec)
End Function
%>
```

In the last tutorial we delete only one image this time we're going to handle all three images using a **for.. next** loop and the sequential numerical names we gave our delete image checkbox our image fields and our image form fields. Before deleting the image(s) we use an **if… then….** Statement to step through our images.

Tip: the for…next loop can used to through numerical values where the beginning and ending number are known or to loop through an array with an unknown number of members using the upper bound `unbound` and lower bound `lbound` properties.

For each field we check two conditions using an if… then statement: Is the user removing the existing is the user uploading a new image. If either or both are true then we delete the existing image. But before we delete the image we need to create our file scripting object to determine if the image exists on the server's file system.

Next we create a new file scripting object to actually delete the image, set the folder path, and the image file name. We'll get the file name from the hidden form field, upload_org_image_field**,** created by Pure ASP Upload which holds the original image value. We then call the fileExists function to make sure we have the file to delete. If we try to execute a delete where there's no file the page will throw an error. If the file exists we use the **File.DeleteFile()** method to remove the image from the server's file system. We close our If… Then statement and set the file back to nothing. Notice that we're using our counter **i** to step through the fields numerically. So that the names passed become Delete_Image1, Delete_Image2, Delete_Image3, and so with each field.

```
      ' check to see if the user has removed or replaced an image and delete old i = 1
To 3
      If UploadFormRequest("P_Image" & i & "") <> "" Or
UploadFormRequest("Delete_Image" & i & "") <> "" Then
      'create file scripting object
         Set File = CreateObject("Scripting.FileSystemObject")
         ImageFolder = Server.MapPath("..\site_images\")
```

```
         ImagePath = ImageFolder & "\" & (UploadFormRequest("upload_org_P_Image" & i &
""))
         ' check if file exists and if true delete the file
         If fileExists(ImagePath) Then File.DeleteFile(ImagePath)
            Set File = Nothing
      End If
      Next
```

Finally the Dreamweaver generated update behaviour appends the query string to redirect to the desired page after the update.

```
    MM_editCmd.Execute
    MM editCmd.ActiveConnection.Close

    ' append the query string to the redirect URL
    Dim MM_editRedirectUrl
    MM_editRedirectUrl = "default.asp"
    If (UploadQueryString <> "") Then
      If (InStr(1, MM_editRedirectUrl, "?", vbTextCompare) = 0) Then
        MM_editRedirectUrl = MM_editRedirectUrl & "?" & UploadQueryString
      Else
        MM_editRedirectUrl = MM_editRedirectUrl & "&" & UploadQueryString
      End If
    End If
    Response.Redirect(MM_editRedirectUrl)
  End If
End If
%>
```

Associated Images

If you're using Smart Image Processor ASP 2 like I am, then you may have some associated images to delete as well. In fact I create three versions of each image for my products and product versions in the DataSite™ CMS V4.00: a thumbnail view, a regular view, and a large view. I use Advanced ToolTips and DMXzone Lightbox extensions to display the created images depending upon the situation inside my application. Deleting you additional images is really quite simple we just need to add capacity image call the fileExists function to make sure it's there in the line of code to delete the file. In code block below 'll notice I've added a second path, **ThumbImagePath**, and a third path, **LargeImagePath**.

```
      For i = 1 To 3
      If UploadFormRequest("P_LargeImage" & i & "") <> "" Or
UploadFormRequest("Delete_Image" & i & "") <> "" Then
      'create file scripting object
         Set File = CreateObject("Scripting.FileSystemObject")
         ImageFolder = Server.MapPath("..\product_images_large\")
         ImagePath = ImageFolder & "\" & (UploadFormRequest("upload_org_P_LargeImage" &
i & ""))
         ThumbImagePath = ImageFolder & "\thumb_" &
(UploadFormRequest("upload_org_P_LargeImage" & i & ""))
         LargeImagePath = ImageFolder & "\large_" &
(UploadFormRequest("upload_org_P_LargeImage" & i & ""))
         ' check if file exists and if true delete the file
         If fileExists(ImagePath) Then File.DeleteFile(ImagePath)
         If fileExists(ThumbImagePath) ThenFile.DeleteFile(ThumbImagePath)
         If fileExists(LargeImagePath) Then File.DeleteFile(LargeImagePath)
```

```
         Set File = Nothing
      End If
      Next
```

If you read the tutorial you should remember I used a multiline if… then statement is to determine if the fileExists and then delete it and added a closing End If. That was for clarity, in this more advanced tutorial I've used a single line if … then statement and there is no need for the closing End If. This method will keep your code a bit shorter.

Advanced Image Handling

In this portion of the tutorial, we'll look at an advanced case on page with images only with the fields sequentially numerically named. We will use a for… next loop to delete images and to step through the parameters. Sense you seen all this before you should have no trouble understanding the following code sample.

```
<%
If (CStr(UploadFormRequest("MM update")) = "UpdateForm") Then
  If (Not MM_abortEdit) Then
    ' execute the update
    Dim MM_editCmd

    Set MM_editCmd = Server.CreateObject ("ADODB.Command")
    MM_editCmd.ActiveConnection = MM_datasource_STRING
    MM_editCmd.CommandText = "UPDATE Products SET P_LargeImage1 = ?, P_LargeImage2 = ?,
P_LargeImage3 = ? WHERE P_ID = ?"
    MM_editCmd.Prepared = true

    ' Loop through and toggle skip empty field behavior to remove large images as
needed
      For image = 1 To 3
            If CStr(UploadFormRequest("Delete_Image" & image & "")) <> "" Then
                  MM_editCmd.Parameters.Append MM_editCmd.CreateParameter("param" &
image & "", 202, 1, 255, UploadFormRequest("P_LargeImage" & image & "")) ' adVarWChar
            Else
                  MM_editCmd.Parameters.Append MM_editCmd.CreateParameter("param" &
image & "", 202, 1, 255, MM_IIF(UploadFormRequest("P_LargeImage" & image & ""),
UploadFormRequest("P_LargeImage" & image & ""),
UploadFormRequest("upload_org_P_LargeImage" & image & ""))) ' adVarWChar
            End If 'End toggle
      Next
      MM_editCmd.Parameters.Append MM_editCmd.CreateParameter("param4", 5, 1, -1,
MM_IIF(UploadFormRequest("MM_recordId"), UploadFormRequest("MM_recordId"), null))
' adDouble

      ' check to see if the user has removed or replaced an image and delete old images
      For i = 1 To 3
      If UploadFormRequest("P_LargeImage" & i & "") <> "" Or
UploadFormRequest("Delete_Image" & i & "") <> "" Then
      'create file scripting object
         Set File = CreateObject("Scripting.FileSystemObject")
         ImageFolder = Server.MapPath("..\product_images_large\")
         ImagePath = ImageFolder & "\" & (UploadFormRequest("upload_org_P_LargeImage" &
i & ""))
         ThumbImagePath = ImageFolder & "\thumb_" &
(UploadFormRequest("upload_org_P_LargeImage" & i & ""))
```

```
        LargeImagePath = ImageFolder & "\large_" &
(UploadFormRequest("upload_org_P_LargeImage" & i & ""))
        ' check if file exists and if true delete the file
        If fileExists(ImagePath) Then File.DeleteFile(ImagePath)
        If fileExists(ThumbImagePath) Then File.DeleteFile(ThumbImagePath)
        If fileExists(LargeImagePath) Then File.DeleteFile(LargeImagePath)
            Set File = Nothing
      End If
      Next

      MM_editCmd.Execute
    MM_editCmd.ActiveConnection.Close

    ' append the query string to the redirect URL
    Dim MM_editRedirectUrl
    MM_editRedirectUrl = "category.asp?Category_ID=" &
(UploadFormRequest("Category_ID"))

      If (UploadQueryString <> "") Then
      If (InStr(1, MM_editRedirectUrl, "?", vbTextCompare) = 0) Then
        MM_editRedirectUrl = MM_editRedirectUrl & "?" & UploadQueryString
      Else
        MM_editRedirectUrl = MM_editRedirectUrl & "&" & UploadQueryString
      End If
    End If
    Response.Redirect(MM_editRedirectUrl)
  End If
End If
%>
```

Conclusion

In this tutorial we learned how to toggle the skip empty fields feature of DMXzone Pure ASP Upload 3. We learned to use the file scripting object and a for each… loop. We learned how to handle associated images that we may have created using DMXzone Smart Image Processor ASP 2. We also learned how to toggle the skip empty fields feature within a for each… loop using sequentially numerically named fields. Using the techniques you learned today you can you offer your CMS users all the image handling capacity they'll ever need.

If you want to learn more, check out the new WWWeb Concepts DataSite™ V4.00 CMS scheduled for release September 1, 2011 at WWWeb Concepts. The DataSite™ V4.00 CMS is a powerful out-of-the-box data-driven web CMS application created to provide Dreamweaver users with a powerful out-of-the-box CMS solution. The WWWeb Concepts DataSite™ V4.00 CMS is designed specifically for Dreamweaver users and features all Dreamweaver and popular Dreamweaver extensions' code which has been modified to increase the functionality and efficiency.