Uploading Files with MVC - Part 4

So far in this blog post series on uploading files with MVC you have learned to style the file upload control, to use a view model for data binding, and how to create a thumbnail from an uploaded image. In this post you learn to store the uploaded file in a folder on your server.

To accomplish this, there are a few changes you need to make to the application written thus far. You need to add a new setting to the Web.config file, add a property in the AppSettings method to read that setting, and a method to the view model to write the original and thumbnail files to disk. In the controller, call the new method in the view model to write the files. Finally, change the src attribute on image tags to use the path to the folder on the server where the files are located.

Web Configuration Settings

Instead of hard-coding the physical path to your web server, add the path in the configuration file for your web application. Open the Web.config file and add a new key/value pair in the <appSettings> section. The key name should be *uploadFolderName*. Set the value to */UploadedFiles/* as shown in the code snippet below.

```
<add key="uploadFolderName" value="/UploadedFiles/" />
```

Modify AppSettings Class

The AppSettings class contains one property for each key/value pair located in the <appSettings> section within the Web.config file. Since you added a new key/value pair for the folder where you wish to store the image files, add a property named *UploadFolderName* to the AppSettings class.

```
public static string UploadFolderName { get; set; }
```

Change the LoadDefaults() Method

Locate the LoadDefaults() method in the AppSettings class and add a new line of code to read the *uploadFolderName* value from the config file and set the *UploadFolderName* property in this class.

```
UploadFolderName =
  GetValue<string>("uploadFolderName", "/UploadedFiles/");
```

View Model

In the last blog post you created a thumbnail from the uploaded image. You need to create two file names to where you will store the original and the thumbnail images on the server. The FileUpload class already has two properties; *ServerUrl* and *ServerThumbnailUrl* to hold these two file names.

SetServerUrlProperties Method

Add a new method to your view model class to set these two properties. Use the *UploadFolderName* property from the AppSettings class and the date to create unique file names for the uploaded file. You do not necessarily have to add the date before the file name, but if you are just keeping these files around temporarily, adding the date provides you with a mechanism to easily delete old files. You could write a utility to delete all files older a specific date.

```
public void SetServerUrlProperties()
{
  string date;

  date = DateTime.Now.ToString("s").Replace(":", "-");

  FileUploadInfo.ServerUrl = AppSettings.UploadFolderName + date +
"-" + FileUploadInfo.FileName;

  FileUploadInfo.ServerThumbnailUrl = AppSettings.UploadFolderName +
date + "-thumbnail-" + FileUploadInfo.FileName;
}
```

SaveToFileSystem Method

Once the file names are created, add a method to perform the actual writing of the files to disk. Add the method shown below, named SaveToFileSystem(), to your view model class. This method calls the WriteAllBytes() method on the System.IO.File class to write both the *Contents* and *Thumbnail* properties to disk.

```
protected void SaveToFileSystem()
{
  try {

File.WriteAllBytes(HttpContext.Current.Server.MapPath(FileUploadInfo
.ServerUrl), FileUploadInfo.Contents);

File.WriteAllBytes(HttpContext.Current.Server.MapPath(FileUploadInfo
.ServerThumbnailUrl), FileUploadInfo.Thumbnail);
  }
  catch (Exception ex) {
    // TODO: Add exception publishing here
    System.Diagnostics.Debug.WriteLine(ex.ToString());
  }
}
```

Save Method

Now that you have the methods written to create the file names, and to write the actual files to disk, add a Save() method to the view model class. The Save() method sets the file information properties, creates the thumbnail, calls the SetServerUrlProperties() method, then finally calls the SaveToFileSystem() method.

```
public void Save()
{
  // Set file info properties from file upload control
  SetFileInfoProperties();

  // Create thumbnail
  CreateThumbnail();

  // Set Server URL properties
  SetServerUrlProperties();

  // Save on file system
  SaveToFileSystem();
}
```

Modify Controller

Now that you moved the code to set the file information properties and create the thumbnail within the Save() method, you can modify the Post method in the MVC controller to call that Save() method. Modify the Post method to look like the following.

```
[HttpPost]
public ActionResult Sample01(FileStorageViewModel vm)
{
  // Store onto file system
  vm.Save();

  // Look at properties of View Model
  System.Diagnostics.Debugger.Break();

  // TODO: Do something with the file data

  return View(vm);
}
```

Modify HTML to use Server URL

With the image now stored in a file on the server, modify the <img> tags on your web page to display the image from the actual files as opposed to the Base64 encoded string.

```
<div class="form-group">
  <img id="theImage"
       style="width: 100%;"
       src="@Model.FileUploadInfo.ServerUrl"
       title="@Model.FileUploadInfo.FileName" />
</div>
<div class="form-group">
  <img id="theThumbnail"
       src="@Model.FileUploadInfo.ServerThumbnailUrl"
       title="@Model.FileUploadInfo.FileName" />
</div>
```

If you have followed along and performed all the steps in this blog post, you should be able to run this page, upload an image, and see the original and thumbnail images displayed on the page. You will also find the files stored on your hard drive under the UploadedFiles folder.

Summary

In this blog post you learned to take the image uploaded and store it onto the file system on your web server. Sometimes you want to keep images around, so storing them on the file system is one mechanism to do so. In this blog post, you added a date and time to the file names written. This makes it easier to delete all files older than a specified date. Another option besides storing images on disk, is to store the images into an SQL Server table. Storing images into SQL Server is the subject of the next blog post in this series.

Sample Code

You can download the complete sample code at my website. http://www.pdsa.com/downloads. Choose "PDSA/Fairway Blog", then "Uploading Files using MVC - Part 4" from the drop-down.