Uploading Files with MVC - Part 5

So far in this blog post series on uploading files with MVC you learned to style the file upload control, use a view model for data binding, create a thumbnail from an uploaded image, and store files on the server's file system. In this post, you learn to store the uploaded file in an SQL Server table. If you have not done so already, please download the sample from Part 4 so you can follow along with this blog post.

Add FileUpload Table to SQL Server

To store the file information into SQL Server, add a table named FileUpload to an SQL Server database using the script shown below. This table is going to store the file contents as well as the file information. If you wish, you do not have to store the file contents, you can place the file on the server's file system, and just store the path and file name in this table. If you wish to do this, eliminate the *Contents* and *Thumbnail* fields from the SQL Script below.

```
CREATE TABLE FileUpload(
 FileUploadId int IDENTITY(1,1) NOT NULL
              CONSTRAINT PK_FileUpload PRIMARY KEY CLUSTERED,
 FileTitle nvarchar(100) NOT NULL,
 FileDescription nvarchar(500) NULL,
 FilePath nvarchar(255) NULL,
 FileName nvarchar(100) NULL,
 ContentLength int NULL,
 ContentType nvarchar(100) NULL,
 Contents varbinary(max) NULL,
 ServerUrl nvarchar(255) NULL,
 ServerThumbnailUrl nvarchar(255) NULL,
 Thumbnail varbinary(max) NULL
)
```

I have already built a local SQL Server database with this table. It is included in the sample download that accompanies this blog post.

Add Entity Framework

If you have not already done so, add the Entity Framework to your MVC application. Open the Web.config file and add a section named <connectionStrings> in the <appSettings> section if you don't already have it. Set the Server attribute to the name of your SQL Server machine name, and the Database attribute to the database name where you created the FileUpload table.

```
<connectionStrings>
  <add name="FileUploadDB"
       connectionString="Server=Localhost;
                         Database=FileUploadSample;
                         Integrated Security=Yes;
                         MultipleActiveResultSets=true"
       providerName="System.Data.SqlClient" />
</connectionStrings>
```

If you are using the local SQL Server database supplied with this blog post, you can use the following connection string. Just modify the path to the location of the FileUploadSample.mdf.

```
Server=(localdb)\MSSQLLocalDB;
Database=FileUploadSample;
AttachDbFilename=D:\Samples\FileUploadSample.mdf;
MultipleActiveResultSets=true
```

Add Database Context Class

To store data in SQL Server, you need a database context class. In the \Models folder add a new file named **FileUploadDB.cs**. Add the following code within this file.

```
using System.Data.Entity;

namespace FileUploadSamples.Models
{
  public partial class FileUploadDB : DbContext
  {
    public FileUploadDB() : base("name=FileUploadDB") { }

    public virtual DbSet<FileUpload> FileUploads { get; set; }
  }
}
```

The class inherits the DbContext class from the System.Data.Entity namespace. Create a constructor that calls the base class' constructor and pass in the name of the connection string you added to the connectionStrings element in the Web.config file. Add a property named *FileUploads* of the type DbSet<FileUpload>. A DbSet class is used to perform CRUD operations on a table via a class that has the appropriate EF attributes added. See Part 2 of this blog post for an example of the FileUpload class or look in the samples download.

Add Method to View Model

Add a new method to your view model class to save the file information to SQL Server. Name this new method SaveToSQLServer() and add the code shown below.

```
protected void SaveToSQLServer()
{
  FileUploadDB db = null;

  try {
    using (db = new FileUploadDB()) {
      // Add file upload info object
      db.FileUploads.Add(FileUploadInfo);
      // Store into SQL Server
      db.SaveChanges();
    }
  }
  catch (Exception ex) {
    System.Diagnostics.Debug.WriteLine(ex.ToString());
  }
}
```

The SaveToSQLServer() method creates an instance of the FileUploadDB database context class. It adds the new FileUpload object to the FileUploads collection and calls the SaveChanges() method. This method executes an INSERT statement into the database and saves the new file information into the FileUpload table. Call this new method from the Save() method in your view model.

Summary

In this blog post you created a table in SQL Server to hold the uploaded file's information. The columns in this table match the FileUpload class you created in Part 2 of this blog post series. You created an Entity Framework database context

class to perform the insert of the data from the FileUpload class into the FileUpload table.

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 5" from the drop-down.