14 Mar 2014

Single File Upload to Multiple File Upload in MVC

In this blog post you will learn how to take advantage of HTML5 in MVC to turn single file upload into multiple file upload functionality. Today almost every browser extended the support to HTML5 and in case any browser does not, it will still work as it was working before and upload single file at a time.

A month back I published same blog post using ASP.NET Web Forms here 'Single File Upload to Multiple File Upload ASP.NET in Web Forms'.

Let's go ahead and first explore how we can upload single file in the SQL Server database and then we will update the existing application to allow multiple file upload in the SQL Server database.

Single File Upload to SQL Server Database

To allow uploading file in MVC we need to create a form which will allow file uploading to server, W3C recommends using enctype="multipart/form-data" to enable this. So here is the view page.

Index.cshtml

This view page is using a view model 'MyViewModel' for its view binding also it uses enctype = multipart/form-data and model validation.

```
@model MvcFileUploadToDB.Models.MyViewModel

@{
    ViewBag.Title = "Home Page";
}

<div class="jumbotron">
    @using (Html.BeginForm(null, null, FormMethod.Post, new { enctype = "multipart/form-data" }))
    {
        <div>
            @Html.LabelFor(x => x.File)
            @Html.TextBoxFor(x => x.File, new { type = "file" })
            @Html.ValidationMessageFor(x => x.File)
        </div>
        <button type="submit">Upload File</button>
    }
</div>

<p>
    @Html.ActionLink("Download File", "Download")
</p>
```

MyViewModel.cs Model

In above view page we used a view model 'MyViewModel', look at the code given below. Please note 'MyViewModel' has a property 'File' which is of type 'HttpPostedFileBase'. HttpPostedFileBase class is an abstract class that contains the same members as the HttpPostedFile class. The HttpPostedFileBase class lets us create derived classes that are like the HttpPostedFile class, but we can customize and that work outside the ASP.NET pipeline.

```
namespace MvcFileUploadToDB.Models
{
    public class MyViewModel
    {
        [DisplayName("Select File to Upload")]
        public HttpPostedFileBase File { get; set; }
    }
```

CATEGORIES

.NET Ajax ARM ASM ASP. Azure Books Bug C C# C# Code Snippet Console App CPP CSS Data Communicat Ebook Elmah Entity Framewor HTML5 HTTP IIS Java Jav JharkhandGeeks jQuery JSO Media Service Microsoft MVP M Miscellaneous Modernizr MVC News-Event Node.js Office 2013 Open Source Oracle PDF People PowerShell Razor Review Security Silverlight SQL Server SSL TCP TFS Tip Training TypeScript Validation VB Virtual Machine Virtual Networ Basic 6 Visual Studio Visual St Visual Studio 2012 WebMatrix Windows Windows (WP) Windows8

POST LIST

▶ 2017 (4)
▶ 2016 (10)
▶ 2015 (46)
▼ 2014 (19)
Nov 2014 (1)
Oct 2014 (4)
Jul 2014 (3)
Jun 2014 (1)
May 2014 (1)
Mar 2014 (2)
Feb 2014 (5)
Jan 2014 (2)
▶ 2013 (63)
▶ 2012 (538)

```
    public class FileUploadDBModel
    {
        public int Id { get; set; }
        public string FileName { get; set; }
        public byte[] File { get; set; }
    }
}
```

It has also a model 'FileUploadDBModel' which is our actual business model.

HomeController.cs File

In above code we have a model FileUploadModel and this model will be used in POST version of action to communicate with backend business/database.

You can see both POST and GET version of actions below. In the GET version of action, we just returning the model to view. POST version of action will receive the MyViewModel as a parameter and after validation it will extract the posted values to fill the FileUploadDBModel properties and save in the database.

```
namespace MvcFileUploadToDB.Controllers
{
    public class HomeController : Controller
    {
        private FileUploadDBContext db = new FileUploadDBContext();

        public ActionResult Index()
        {
            var model = new MyViewModel();
            return View(model);
        }

        [HttpPost]
        public ActionResult Index(MyViewModel model)
        {
            if (!ModelState.IsValid)
            {
                return View(model);
            }

            FileUploadDBModel fileUploadModel = new FileUploadDBModel();

            byte[] uploadFile = new byte[model.File.InputStream.Length];
            model.File.InputStream.Read(uploadFile, 0, uploadFile.Length);

            fileUploadModel.FileName = model.File.FileName;
            fileUploadModel.File = uploadFile;

            db.FileUploadDBModels.Add(fileUploadModel);
            db.SaveChanges();

            return Content("File Uploaded.");
        }

        public ActionResult Download()
        {
            return View(db.FileUploadDBModels.ToList());
        }

        public FileContentResult FileDownload(int? id)
        {
            byte[] fileData;
            string fileName;

            FileUploadDBModel fileRecord = db.FileUploadDBModels.Find(id);

            fileData = (byte[])fileRecord.File.ToArray();
            fileName = fileRecord.FileName;

            return File(fileData, "text", fileName);
        }
```

```
    }
}
```

We have an action 'Download' which will return the list of uploaded files from the database. When user will click on download link associated with each files on the view, a method 'FileDownload' will receive the call and return the File type data and user will notice download progress in the browser.

As we are using Code First, there should be DbContext and DbSet, find them below.

Database Context

Here is my DbContext and DbSet.

```
namespace MvcFileUploadToDB.Models
{
    public class FileUploadDBContext : DbContext
    {

        public FileUploadDBContext()
            : base("name=FileUploadDBContext")
        {
        }

        public System.Data.Entity.DbSet<MvcFileUploadToDB.Models.FileUploadDBModel> FileUploadDBModels {
get; set; }

    }
}
```

In the above code we used 'FileUploadDBContext', this is basically our configuration string name. Let's look at that connection string as well.

```
<add name="FileUploadDBContext" connectionString="Data Source=(localdb)\v11.0; Initial
Catalog=DelContext-20140305142934; Integrated Security=True; MultipleActiveResultSets=True;
AttachDbFilename=|DataDirectory|DelContext-20140305142934.mdf"
  providerName="System.Data.SqlClient" />
```

Now in order to allow user download the upload files from SQL Server database we need a view.

Download.cshtml

This view page will list the uploaded files from the database with download link by using id unique property for each item/file in the list.

```
@model IEnumerable<MvcFileUploadToDB.Models.FileUploadDBModel>

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Upload New", "Index")
</p>

<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.FileName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.File)
        </th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
```

```
                @Html.DisplayFor(modelItem => item.FileName)
            </td>
            <td>
                @Html.ActionLink("Download", "Filedownload", new { id=item.Id})
            </td>
        </tr>
    }
</table>
```

Now, run the application you will see everything working. Using this application you will be able to upload one file at a time.

But what if we want to select multiple files and upload them at once. Let's learn this too, we are not going to make huge change in above code, just 2-3 quick change and done.

Multiple Files Upload to SQL Server Database

To allow multiple files uploading in MVC, we first need to change the view that will allow selecting multiple files. Because, in above approach we can't even select multiple files. In order to allow selecting multiple files we need to add 'multiple' attribute with the helper.

Index.cshtml

I made following changes to allow selecting multiple files in the same view page.

```
@Html.TextBoxFor(x => x.File, new { type = "file", multiple = "true" })
```

MyViewModel.cs Models

Updated the File property of MyViewModel to allow handling list of files.

```
public class MyViewModel
{
    [DisplayName("Select File to Upload")]
    public IEnumerable<HttpPostedFileBase> File { get; set; }
}
```

HomeController.cs File

In the Home Controller keep every actions as it is, just update POST version of Index method as given below.

```
[HttpPost]
public ActionResult Index(MyViewModel model)
{
    if (!ModelState.IsValid)
    {
        return View(model);
    }

    FileUploadDBModel fileUploadModel = new FileUploadDBModel();

    foreach (var item in model.File)
    {
        byte[] uploadFile = new byte[item.InputStream.Length];
        item.InputStream.Read(uploadFile, 0, uploadFile.Length);

        fileUploadModel.FileName = item.FileName;
        fileUploadModel.File = uploadFile;

        db.FileUploadDBModels.Add(fileUploadModel);
        db.SaveChanges();
    }

    return Content("File Uploaded.");
}
```

What we are doing here is looping through files available in model and uploading them one by one in the database.

Now, run the application you will see everything working and you will be able to select multiple files and upload them at once.

In case you want complete source code, it is available on Gist.

Single File Upload in SQL Server Database
Multiple Files Upload in SQL Server Database

Hope this helps.

About Author

Abhimanyu Kumar Vatsa, Microsoft MVP in ASP.NET/IIS | Author | Blogger, works as Senior Software Engineer at Knorish Framework Pvt. Ltd.

Know More

Connect him on Facebook | Twitter (@itorian) | YouTube

Posted by Abhimanyu Kumar Vatsa at 08:57 In Category ASP.NET, HTML5, MVC, SQL Server, Visual Studio

Comment using Facebook (0 comment(s)):

0 Comments Sort by **Oldest**

Add a comment...

Facebook Comments Plugin

Comment using Google Services (4 comments):

Ankit Shahra Tuesday, May 27, 2014 3:09:00 am

In multiple files upload in sql server database, In Controller foreach loop is not working...because foreach statement cannot operate on varible type HttpPostedFileBase.
It does not contain public definition for "Get Enumerator".

Reply

Anonymous Tuesday, September 16, 2014 11:25:00 pm

How can I get the source path of the file by using this code

Reply

code4developer Tuesday, October 07, 2014 11:49:00 pm

Its so nice post. its totally helpful thanks.

Reply

Anonymous Monday, April 20, 2015 12:00:00 am

Very useful article. But on Index page on column Attachment it is showing long line of numbers, how can change it to document name

Reply

Enter your comment...

Comment as: raliqalatopollo@ ▾ Sign out

Publish Preview Notify me