Simple File Upload – Step By Step

Assumptions:

- The table being used is **Titles**
- The field to hold our file is called **BookImage**
- The item being uploaded is an image – See **how to handle PDF s for linking at the end of this document** if that is what you are attempting to upload and display.
- The location in the site structure where images are stored is: ***(UI_Layer)*** **~/Content/Images/Books/**
- **NoImage.png** is the name of our ***default image***. It has been saved to the file location above.
- ***All test data*** (that involves images) in our database table either **has a valid image name** (the images has been saved to the folder above) **or is set to noImage.png**

To fully complete all steps to the file upload, you will need the following files open:

- Index.cshtml (if you wish to display an image there)
- Details.cshtml (you should DEFINITELY display the image there)
- Create.cshtml
- Edit.cshtml

Prep pieces of the UI. (index.cshtml & details.csthtml)

1) In the index.cshtml (if you are using it), locate the <td> that contains the **Html.DisplayFor(modelItem =>modelItem.BookImage)**←(***look for your table's field name***)
2) Comment it out (or delete) and replace it with an ****
3) This should connect your src value (mapping to where you saved your image files) to the database value (just the image name). As long as you have valid data and images saved per our Assumptions above, you should be able to run and test your index view

```
@foreach (var item in Model) {
    <tr>
        <td>
        <img src="@Url.Content("~/Content/Images/Books/" +item.BookImage)"
                    alt="@item.BookTitle" />
        </td>
```

4) Perform the same Activity in the details view. When you map the src value, you will use Model.BookTitle instead of item.BookTitle (as this is not in a foreach loop).

****

```
<dt>
    @Html.DisplayNameFor(model => model.BookImage)
</dt>
<dd>
    <img src="@Url.Content("~/Content/Images/Books/" +Model.BookImage)"
         alt="@Model.BookTitle" />
</dd>
```

5) **Test**

Create

1) Modify the **BeginForm()** at the top of the view to reflect the following ***(as it pertains to your ControllerName)***. This is going to the **Create/Post** of the ***Titles Controller in this example***

```
@using (Html.BeginForm("Create","Titles",FormMethod.Post,
    new {@enctype="multipart/form-data" }))
```

2) Add an <input type=”file” /> (replacing the existing **Html.DisplayFor()** for ***your*** image field

```
<div class="form-group">
    @Html.LabelFor(model => model.BookImage, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        <input type="file" name="coverImage" />
        @*@Html.EditorFor(model => model.BookImage, new { htmlAttributes = new { @class = "form-control" } })*@
        @*@Html.ValidationMessageFor(model => model.BookImage, "", new { @class = "text-danger" })*@
    </div>
</div>
```

3) Go to the **[HttpPost] Create Action** of your controller and **another parameter** for your **HttPPostedFileBase** ***(this name must match the name value of your input in the create view)*** (the bind list was truncated for this image example)

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include = "TitleID,ISBN,BookTitle,[
Title title, HttpPostedFileBase coverImage)
{   //HttpPostedFileBase is the datatype for the file.
    //the name of the variable must match the name attribute of the
    //input type=file in the UI
```

4) **INSIDE of the if(ModelState.IsValid)** code (**above whatever else is in there** add the following code. ***(Remember to change your variable names and values to match your project)***

```
//establish a variable for our default image
string imageName = "noImage.png";
//if a file was sent
if (coverImage != null)
{
    //reassign the variable to the filename sent over
    imageName = coverImage.FileName;

    //create a variable for the extension
    string ext = imageName.Substring(imageName.LastIndexOf('.'));

    //create a list of valid extensions
    string[] goodExts = {".jpg", ".jpeg", ".png", ".gif" };

    //if our extension is valid, assign a GUID as the name
    //and concatonate the extension
    if (goodExts.Contains(ext.ToLower()))
    {
        //save the file to the webserver
        coverImage.SaveAs(Server
            .MapPath("~/Content/Images/Books/" + imageName));
    }
    else
    {
        //otherwise default back to the default image
        imageName = "noImage.png";

    }
}
//No matter what send the file name to the database
//as the value for the BookImage property
title.BookImage = imageName;
```

5) **Test**

Edit

1) Modify the **BeginForm()** at the top of the view to reflect the following ***(as it pertains to your ControllerName)***. This is going to the **Edit/Post** of the ***Titles Controller in this example***
    a. You will need to add a HiddenFor() for your image/file in the event that one is not uploaded in the edit. You may want to put it with the existing one for the ID:

```
@using (Html.BeginForm("Edit","Titles",FormMethod.Post,
    new { @enctype = "multipart/form-data" }))
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.TitleID)
        @Html.HiddenFor(model => model.BookImage)
```

2) Add an <input type=”file” /> (replacing the existing **Html.EditorFor()** for ***your*** image field

```
<div class="form-group">
    @Html.LabelFor(model => model.BookImage, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        <input type="file" name="coverImage" />
        @*@Html.EditorFor(model => model.BookImage, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.BookImage, "", new { @class = "text-danger" })*@
    </div>
</div>
```

3) Go to the **[HttpPost] Edit Action** of your controller and **another parameter** for your **HttPPostedFileBase** ***(this name must match the name value of your input in the create view)*** (the bind list was truncated for this image example)

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include = "TitleID,
Title title, HttpPostedFileBase coverImage)
```

4) **INSIDE of the if(ModelState.IsValid)** code (**above whatever else is in there** add the following code. ***(Remember to change your variable names and values to match your project)***

```
//There is no Default Image here - if a new one is not provided
//the HiddenFor() in the view will be used (notes at the the end)
if (coverImage != null)
{
    //reassign the variable to the filename sent over
    string imageName = coverImage.FileName;

    //create a variable for the extension
    string ext = imageName.Substring(imageName.LastIndexOf('.'));

    //create a list of valid extensions
    string[] goodExts = { ".jpg", ".jpeg", ".png", ".gif" };

    //if our extension is valid, assign a GUID as the name
    //and concatonate the extension
    if (goodExts.Contains(ext.ToLower()))
    {
        //save the file to the webserver
        coverImage.SaveAs(Server
            .MapPath("~/Content/Images/Books/" + imageName));

        //Save to the database
        title.BookImage = imageName;
    }
    //If the image ext is not valid we default to the existing
    //image that is passed by the Html.HiddenFor() in the Edit
    //View
}

db.Entry(title).State = EntityState.Modified;
db.SaveChanges();
return RedirectToAction("Index");
```

5) **Test**

There are some subtle changes if you are uploading a **pdf that is to be viewed**:

Prep Pieces of the UI (index.cshtml & details.csthtml):

1) You will have **no <img> tag** to display, you will **instead have an <a>**:
   <a href="@Url.Content("~/Content/Images/PDFS/" + @item.FieldForPDF)"
   target="_blank">Click to View Document
   (Remember in the index view it will be item.Field and in the details view it will be Model.Field)

Create:

1) There are no changes the Create View
2) Instead of a no image available you will still want to care for someone not uploading a pdf you can do one of the following:
   a. Create a PDF that says No Document Provided (probably in large font) and name it noPDF.pdf and use it as your default
   b. If adding the file is REQUIRED, then if the HttpPostedFileBase object IS NULL, return the view with the (unfinished/incomplete object) and pass messaging that the file is required.
3) Instead of the string[] for goodExts (valid extensions), just ensure that your file extension is .pdf

Edit:

1) Instead of the string[] for goodExts (valid extensions), just ensure that your file extension is .pdf

Other Uses (Email)

1) In the case of email, you do not need to save the file to the web server
2) You will only need to add it to the attachments collection of the MailMessage object before you send the email

```
MailMessage m = new MailMessage();
//Your Mail Code here
//This with go with the other mail properties
m.IsBodyHtml = true;
m.Attachments.Add(yourVariableHere);
```