**Showing Uploaded files on visiting a page**

In the previous article we have seen how to display the uploaded image as soon as it is uploaded. In this article we will learn how to display the uploaded image on visiting a page.

In this article we will use the same sample application that we have been using for the previous articles.

In the previous article we made code changes to display the uploaded image as soon as it is uploaded. In this article we will continue from where we left off.

To show uploaded files on visiting a page, we will create a new nav bar item, controller action method and a new view item to display the list of all the uploaded files. In the controller action method, we will execute a select query to fetch all the files from the database table and pass it to the view to display it.

**Adding a new nav-bar item**

**\_Layout.cshtml** file

I have added a new navigation bar item named “Files” to the “**\_Layout.cshtml**” which is within “Views --> Shared” folder as shown below

“

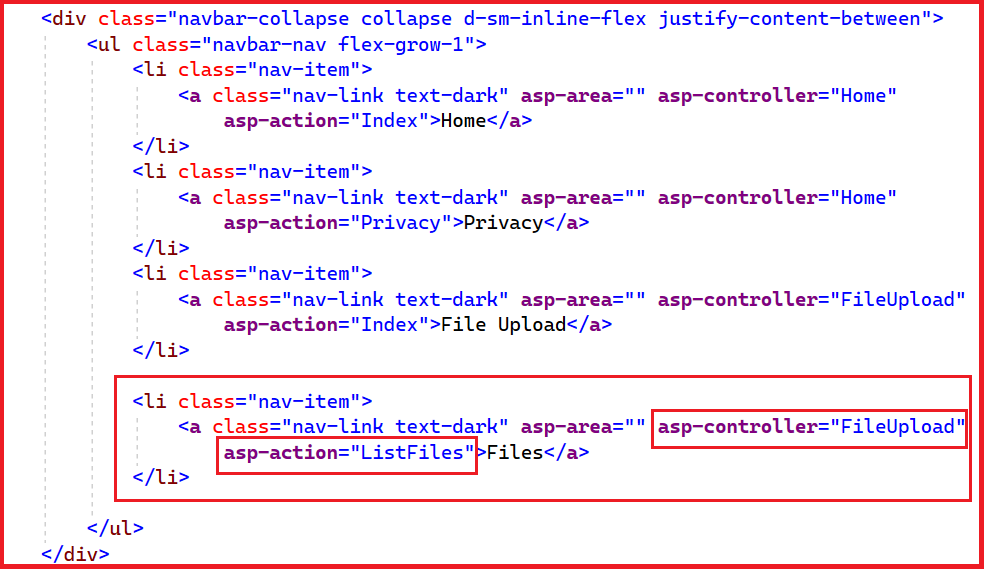
*<li class="nav-item">*

*<a class="nav-link text-dark" asp-area="" asp-controller="****FileUpload****"*

*asp-action="****ListFiles****">Files</a>*

*</li>*

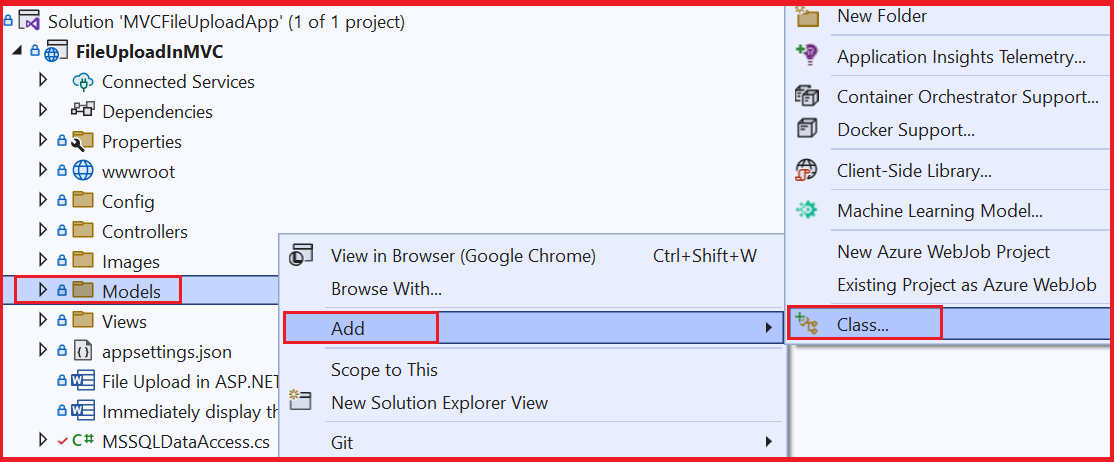
”



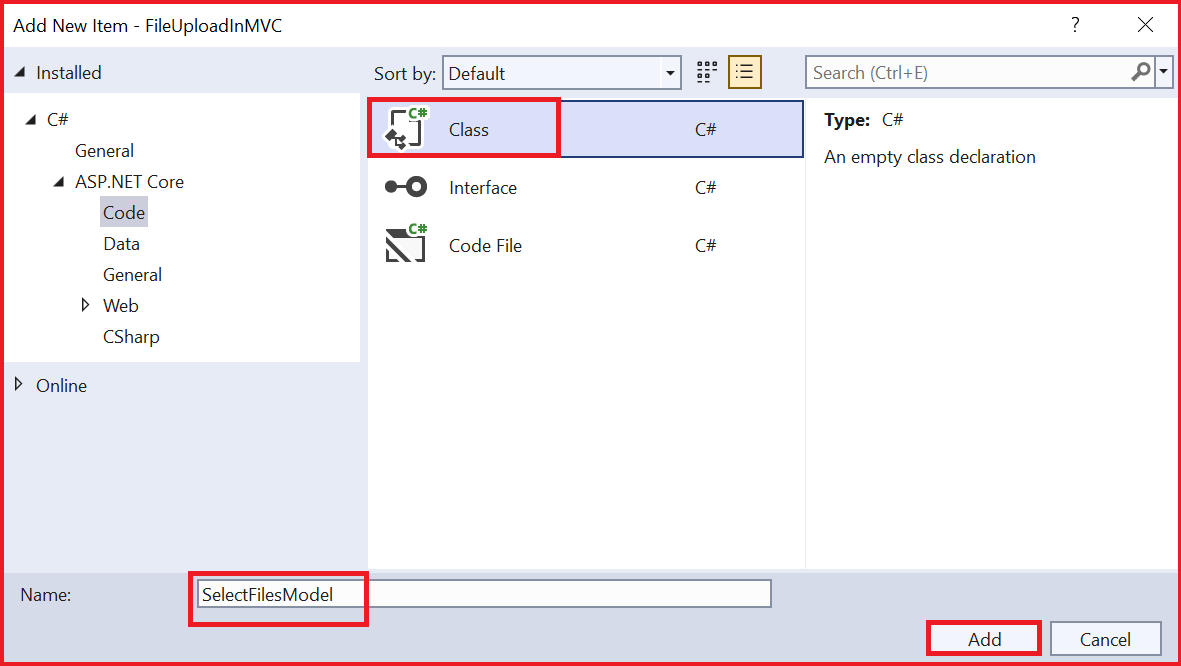
Notice that the “asp-controller” is “**FileUpload**” and “asp-action“ is “**ListFiles**”. So, on click of nav-bar item, the “ListFiles” action method in the FileUpload controller will be executed.

Adding a new model which is a C# equivalent of the “FileUpload” table which we have created in the previous article.

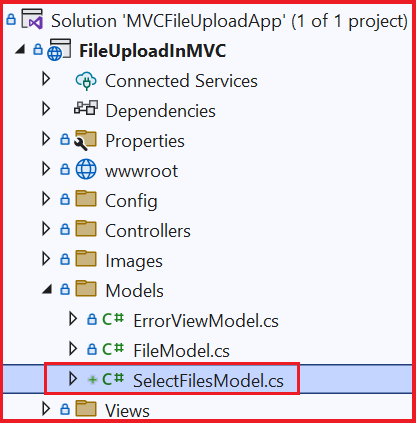
1. Right click on the “Models” folder, select “Add” and select “Class”



1. In the “Add New Item” window, give the name of the file as “SelectFilesModel” and then click on “Add” as shown below



1. “SelectFilesModel” class will be created as shown below



Add below properties to the **SelectFilesModel.cs** file

**SelectFilesModel.cs file**

“

*namespace FileUploadInMVC.Models*

*{*

*public class SelectFilesModel*

*{*

*public int Id { get; set; }*

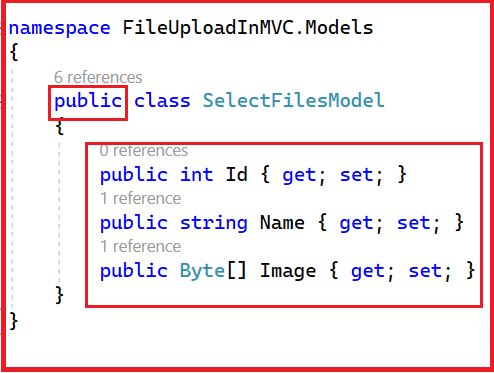
*public string Name { get; set; }*

*public Byte[] Image { get; set; }*

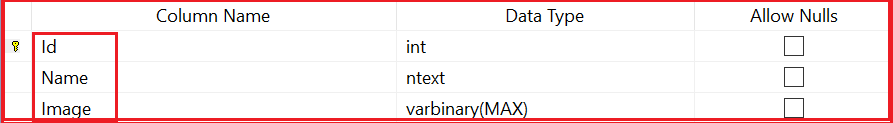
*}*

*}*

”



Notice that I have changed the access modifier for the class to “public”. Note that property names for “Id”, “Name” and “Image” must match exactly with the column names in the table



C# equivalent for SQL data types

|  |  |
| --- | --- |
| SQL Data type | C# equivalent data type |
| int | int |
| ntext | String |
| Varbinary(MAX) | Byte[] |

**Controller changes**

**FileUploadController.cs**

I have added a new action method named “**ListFiles**”. In this action method we make a call to “**GetAllFilesFromDatabaseAsync**” method defined in **MSSQLDataAccess.cs** file. The values returned from the “GetAllFilesFromDatabaseAsync” method will be passed to the “**Files**” View as shown below

“

*public async Task<IActionResult>* ***ListFiles****()*

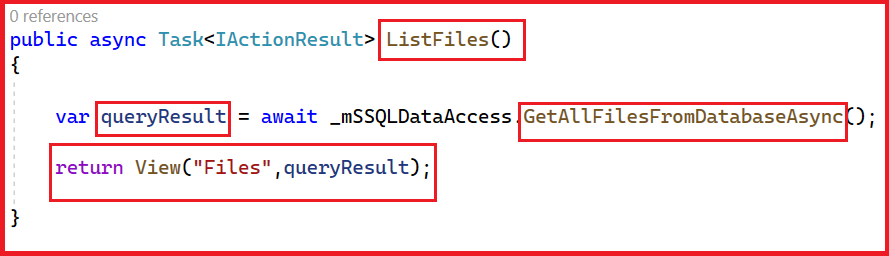
*{*

*var* ***queryResult*** *= await \_mSSQLDataAccess.****GetAllFilesFromDatabaseAsync****();*

*return View("Files",****queryResult****);*

*}*

”



Data access layer changes

**MSSQLDataAccess.cs** changes.

We have added a new method “**GetAllFilesFromDatabaseAsync**”. It executes a select query on the “FileUpload” table of the “sample” database which we have created in the previous article.

The select query fetches all the records from the “FileUpload” table. The query results will be automatically mapped to the “SelectFilesModel” by the dapper. The query result will be an IEnumerable of “SelectFilesModel” which will be returned to the controller action which then will be passed to the view.

“

*public async Task<IEnumerable<SelectFilesModel>> GetAllFilesFromDatabaseAsync()*

*{*

*try*

*{*

*using var conn = new SqlConnection(\_config.GetConnectionString("SQLDB"));*

*var query = "****SELECT \* FROM FileUpload****";*

*var queryResult = await conn.QueryAsync<****SelectFilesModel****>(query);*

*return queryResult;*

*}*

*catch (Exception)*

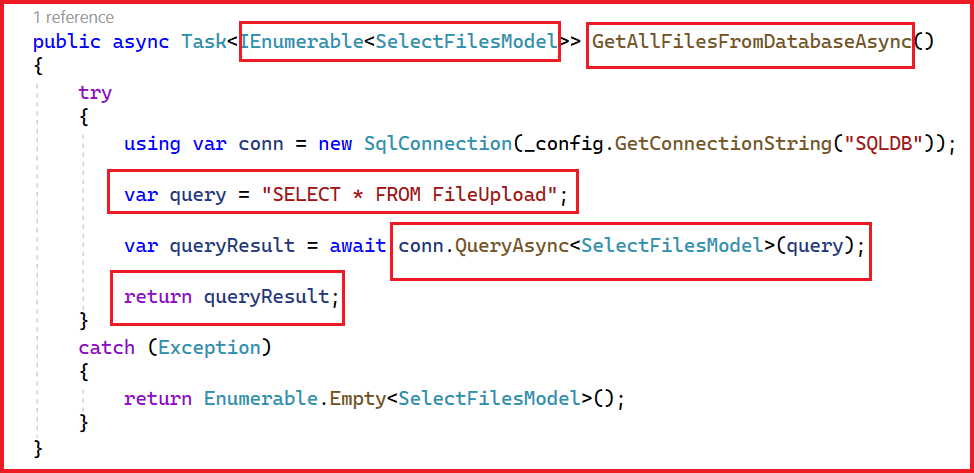
*{*

*return Enumerable.Empty<SelectFilesModel>();*

*}*

*}*

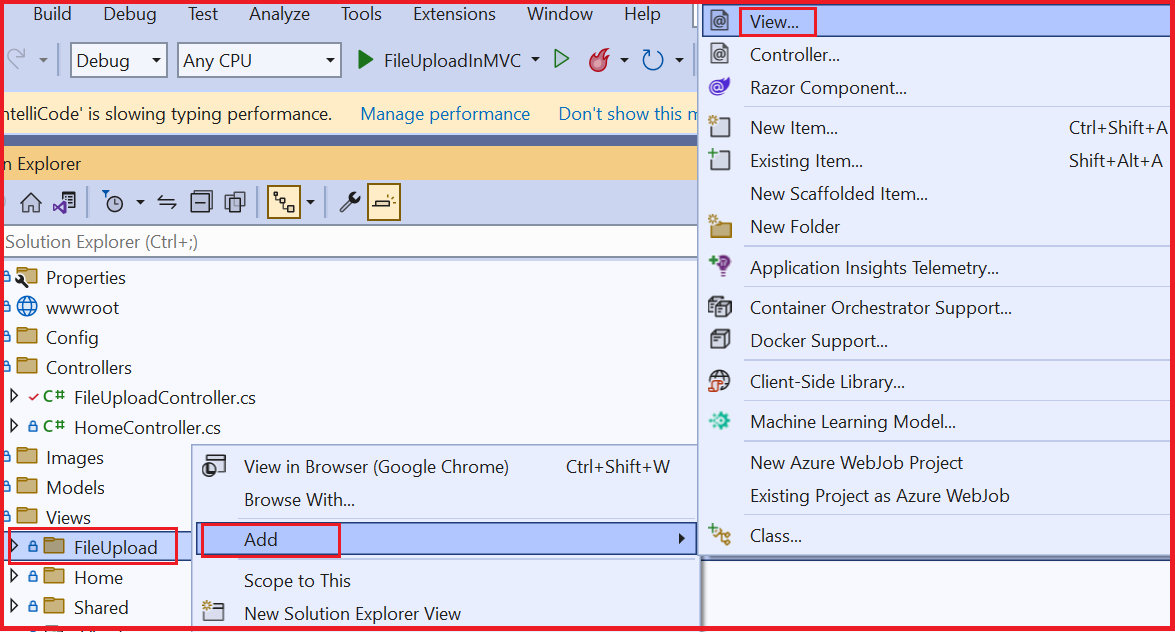
”



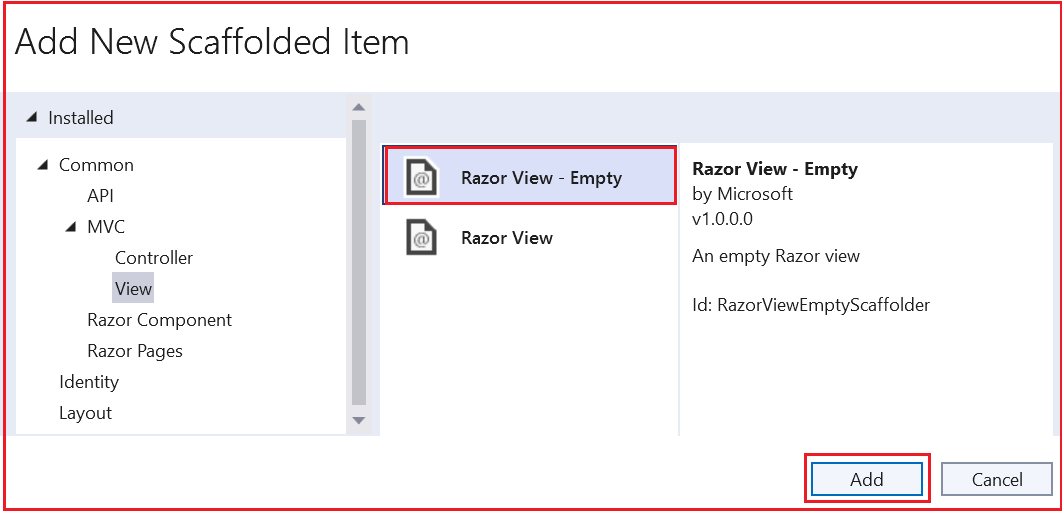
Notice that we are creating a new sql connection with the help of connection string configured in appsettings.json file like what we did in previous articles. Then we frame a select query to fetch all the records. We execute the query using QueryAsync method.

We will create a new view “Files.cshtml” under “Views” --> FileUpload folder

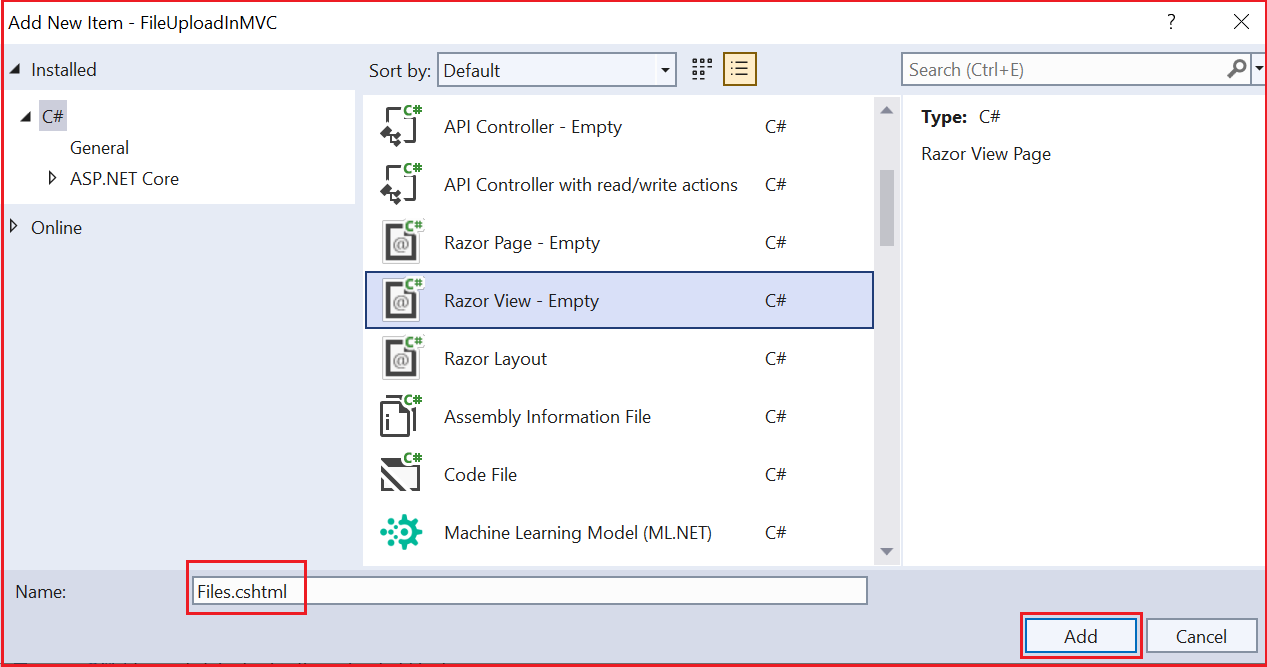
1. Right click on “FileUpload” folder, select “Add” and then select “View” as shown below



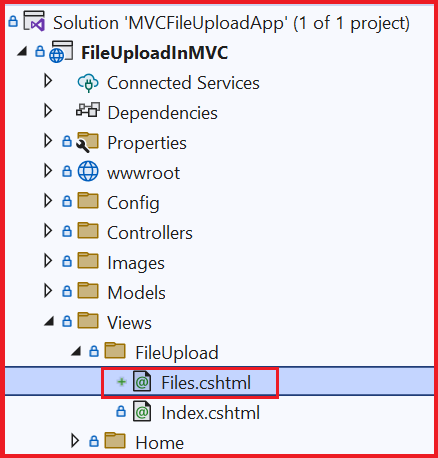
1. In the “Add New Scaffolded Item” window, select “Razor View - Empty” and click on “Add” as shown below



1. Give the file name as “Files.cshtml” and click on “Add” as shown below



1. “Files” view will be created as shown below



In the Files view, we iterate over the **IEnumerable<SelectFilesModel>** passed by the controller action and populate the file name and image on the html table. Displaying images is the same way as we did in the previous article. We use <img> control, convert the byte array image to base 64 string and set it as the source of the image.

We have used a few style classes to set the height and width of the image, to set the table content to center of the column and to add border for the table.

HTML table

“

*@model* ***IEnumerable<SelectFilesModel>***

*<div>*

*<h4>List of uploaded files</h4>*

*<****table*** *style="width:50%">*

*<tr>*

*<****th*** *class="****tableHead****">File Name</th>*

*<th class="****tableHead****">Image</th>*

*</tr>*

*@foreach (var file in @Model)*

*{*

*<tr>*

*<****td****>*

*@file.Name*

*</td>*

*<td>*

*<****img*** *class="****imageClass****" src="@String.Format("data:image/gif;base64,{0}",*

*Convert.ToBase64String(@file?.Image))">*

*</img>*

*</td>*

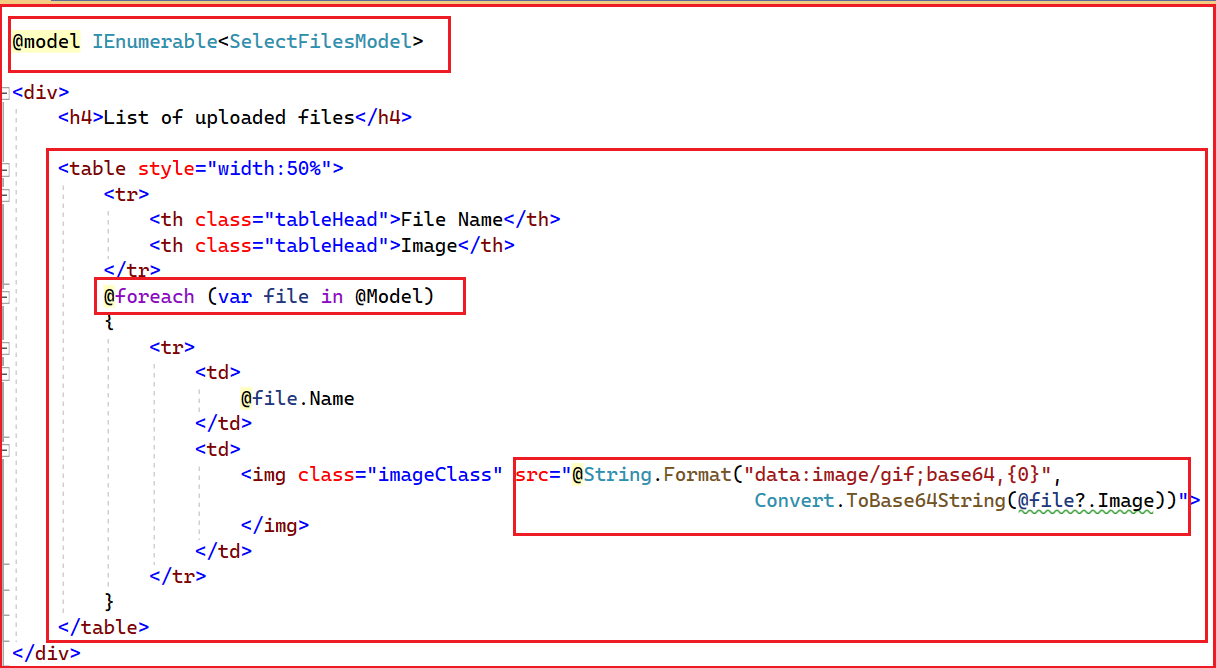
*</tr>*

*}*

*</table>*

*</div>*

”



**Style**

“

*<style>*

*.imageClass{*

*width:50%;*

*height:50%;*

*}*

*.tableHead{*

*width:50%;*

*}*

*table, th, td {*

*border: 1px solid black;*

*text-align: center;*

*}*

*</style>*

”



We have added three style classes

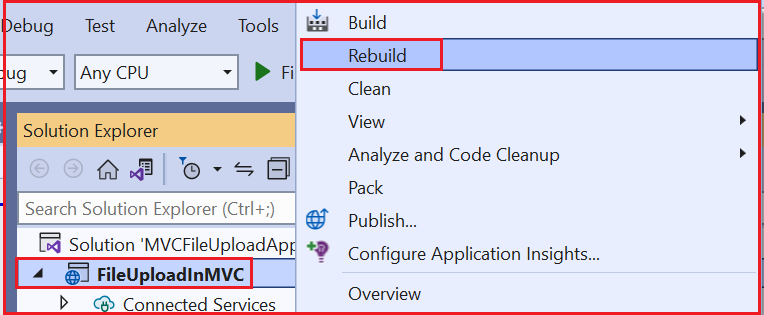
**imageClass :** This is used by the **image** (<img>) control. We set the width and height of the image to fit the view.

**tableHead :** This is used by the **table head** (<th>) control. We set the width of the table column

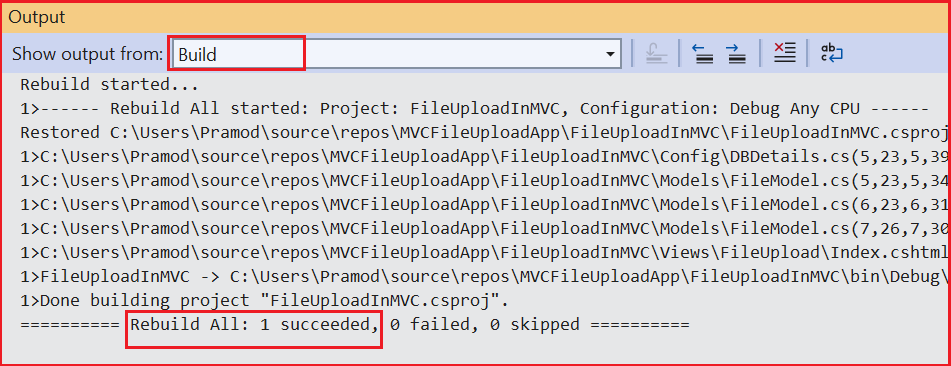
**table, th, td:** The styling applies to table, table header and value cell. Here we set the border for table and cells and set the text alignment in the table cell to center.

With all these changes in place, let’s rebuild the application and run.

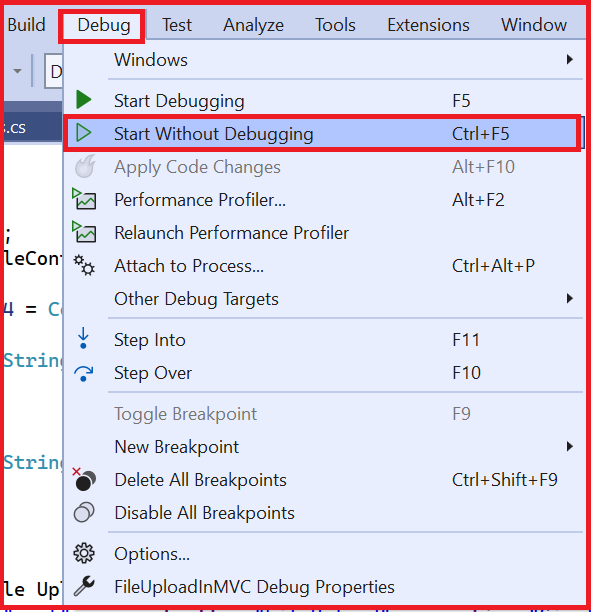
1. Right click on the project and select “Rebuild”



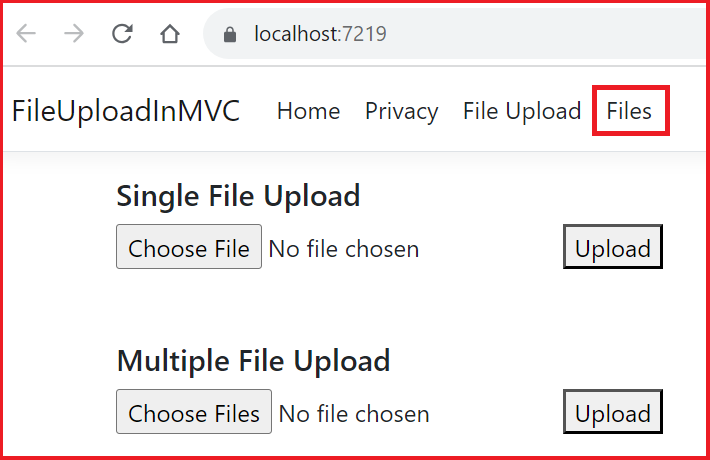
1. We can check the status of the build from the output window



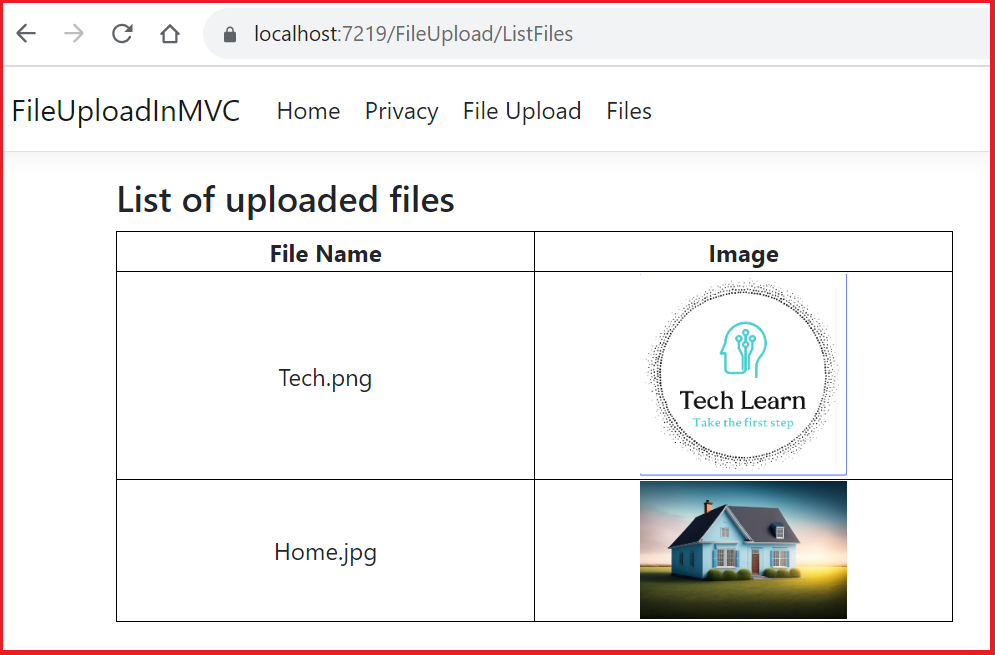
1. Run the application by selecting “Start Without Debugging” from “Debug” section



1. The application will display the “index” view of the Home Controller. Notice that the newly added nav-bar item “Files” is displayed as highlighted in the below screenshot

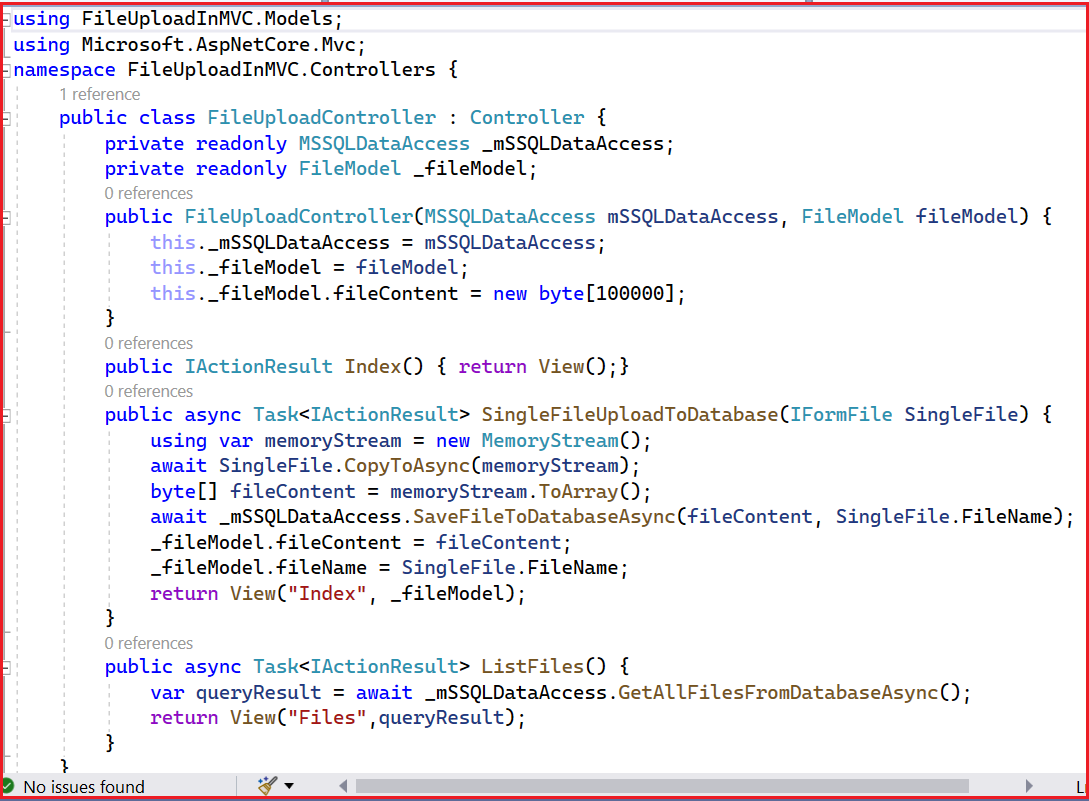


1. Click on the “Files” nav bar item to navigate to “Files” view of “File upload” controller.  
   Files view will be displayed as shown below

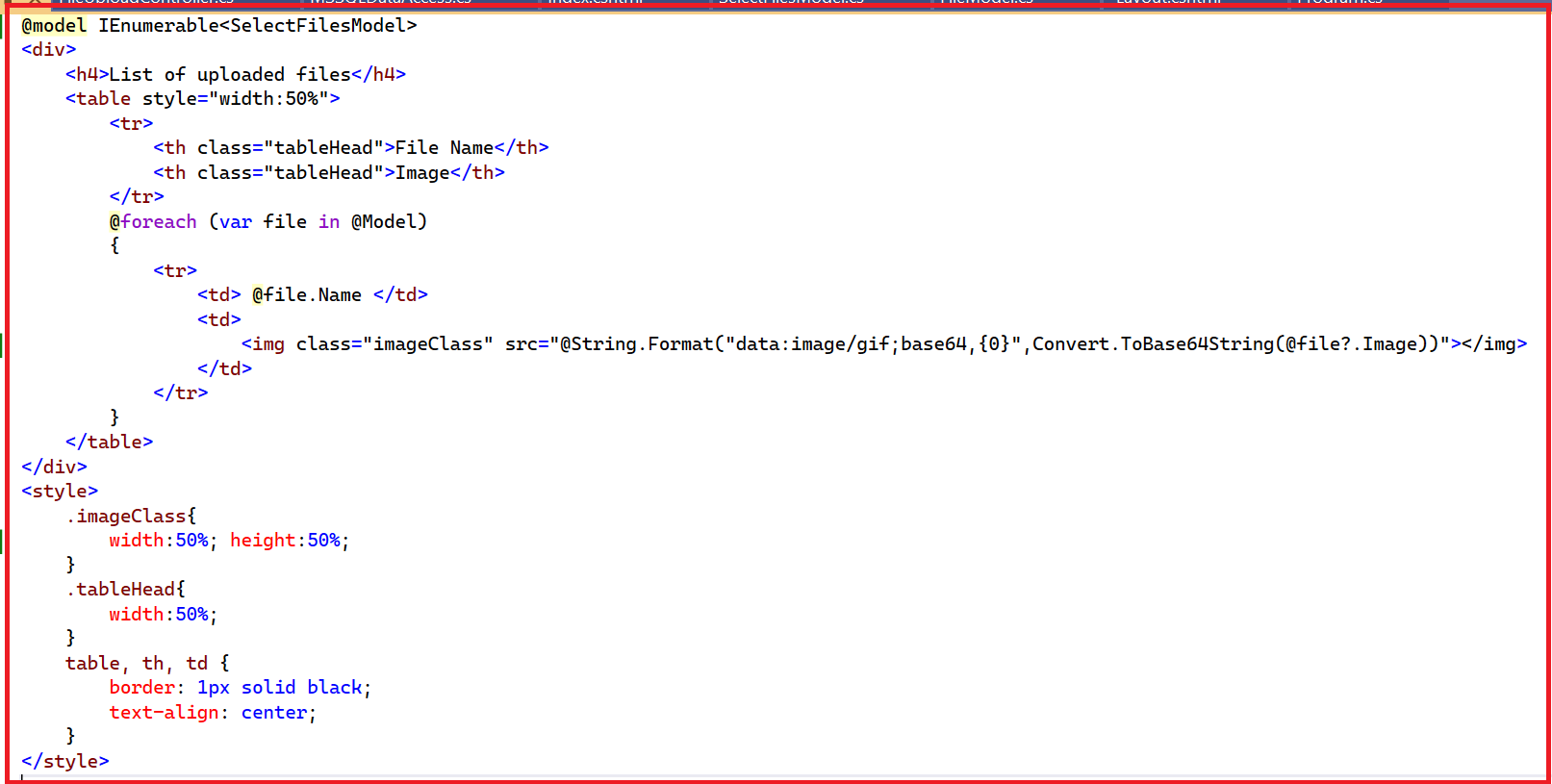


Notice that the file names and their corresponding images are displayed

Complete “FileUploadController.cs” file now looks like below



Complete “Files.cshtml” file now looks like below



In this article we have learnt how to display the uploaded file on loading a page. Let me know your thoughts on this in the comments section. In the next article we will discuss how to delete the file from the database