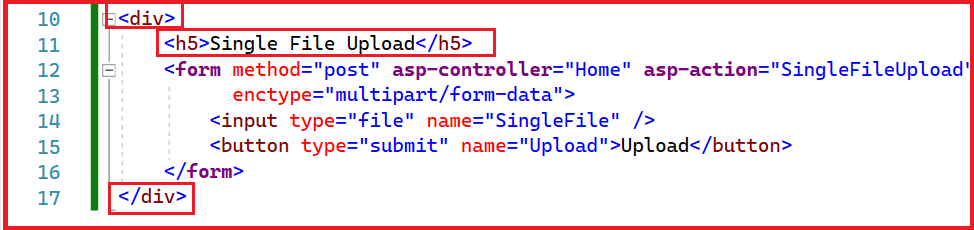
**Uploading multiple files using buffering in ASP.NET core MVC**

In the previous article we learnt about uploading a single file in asp.net core MVC application using buffering approach. In this article we will discuss how to upload multiple files in asp.net core MVC using the buffering approach.

Let’s continue with the same application which we created in the previous content.

To differentiate “Single File Upload” and “Multiple File Upload” examples, I have enclosed the form created in the “**Index.cshtml**” file in the previous example with <div> and <h5> tags as shown below

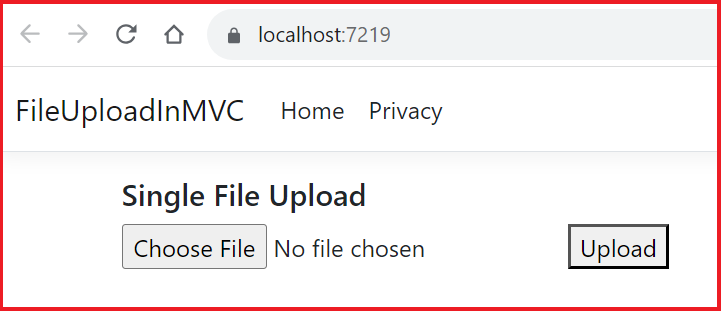


“

<h5>Single File Upload</h5>

”

The “Index” view now looks as shown below

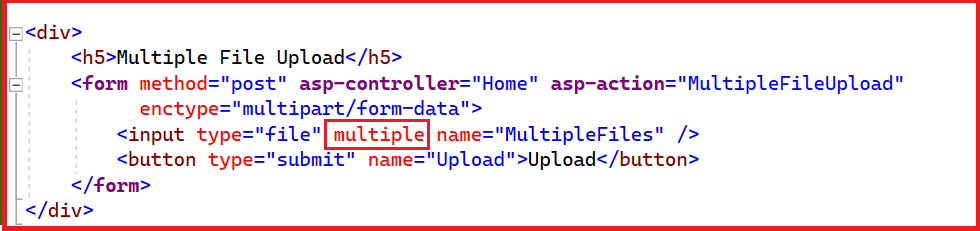


Let’s update the “**Index.cshtml**” file under Views => Home folder to add another form control within a div element for uploading multiple files. The div element contains a <h5> tag followed by form element. The form control encloses an input control of the type “file” and is followed by a button control of type submit.

The index.cshtml file containing both the form elements is as shown below. Added two <br> break tags in between form elements to have space between two form elements for better UI look as shown below



Let’s look in detail at the form element added for multiple file upload.



“

<div>

<h5>Multiple File Upload</h5>

<form method="post" asp-controller="Home" asp-action="**MultipleFileUpload**"

enctype="multipart/form-data">

<input type="file" **multiple** name="**MultipleFiles**" />

<button type="submit" name="Upload">Upload</button>

</form>

</div>

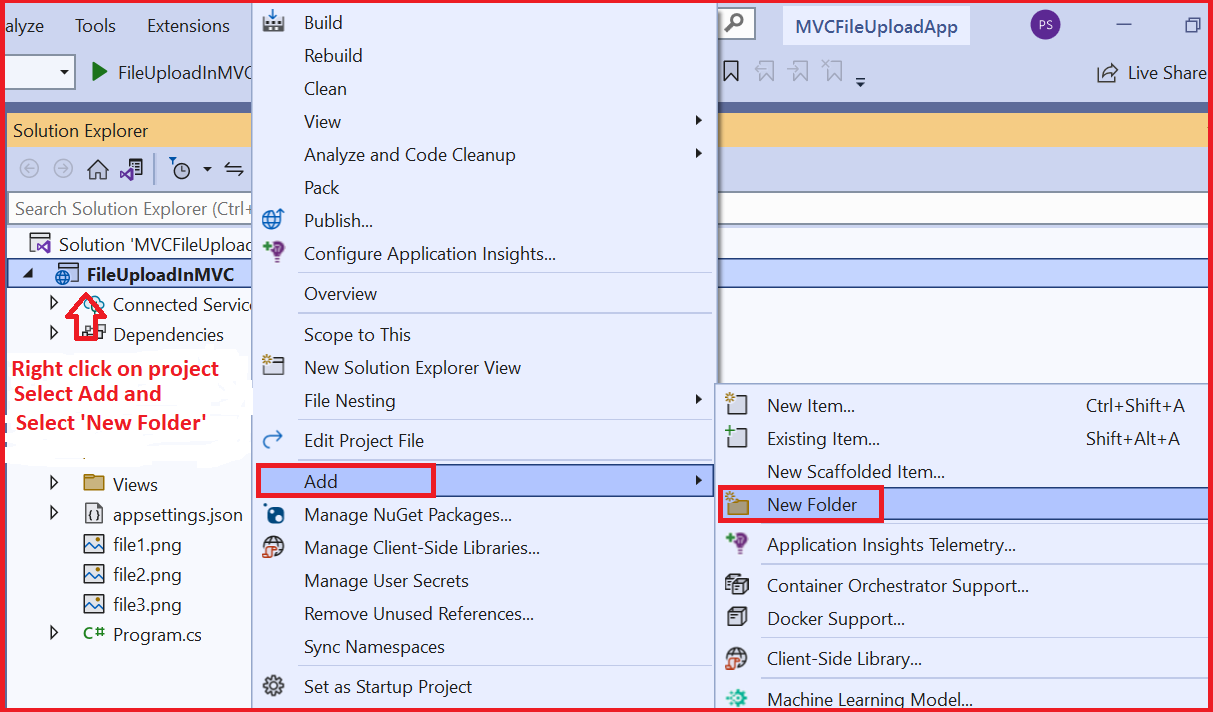
”

Notice that

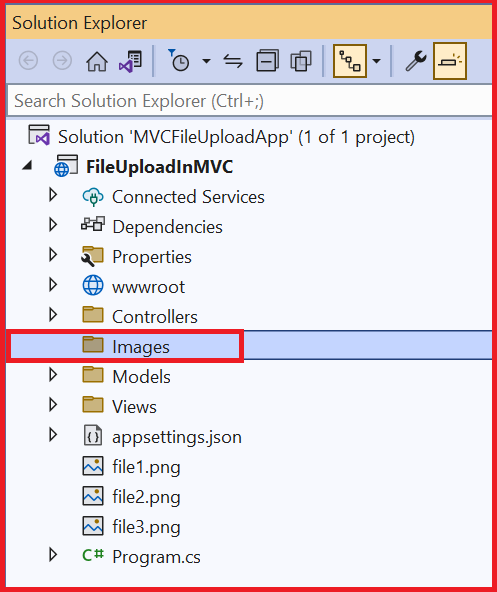
1. form control has method = post, asp-controller="**Home**" asp-action=" **MultipleFileUpload**" attributes.  
   On submitting the form, the “**MultipleFileUpload**” action method in the “**Home**” controller will be executed.
2. The name attribute of the file type input is **MultipleFiles**
3. There is a “**multiple**” attribute on the input element of type “file”.  
   This element is required for supporting multiple file uploads.

Let’s create a new folder “Images” within the project from Visual Studio. The uploaded files will be saved within this folder

1. Right click on the project and select “Add”, then select “New Folder”

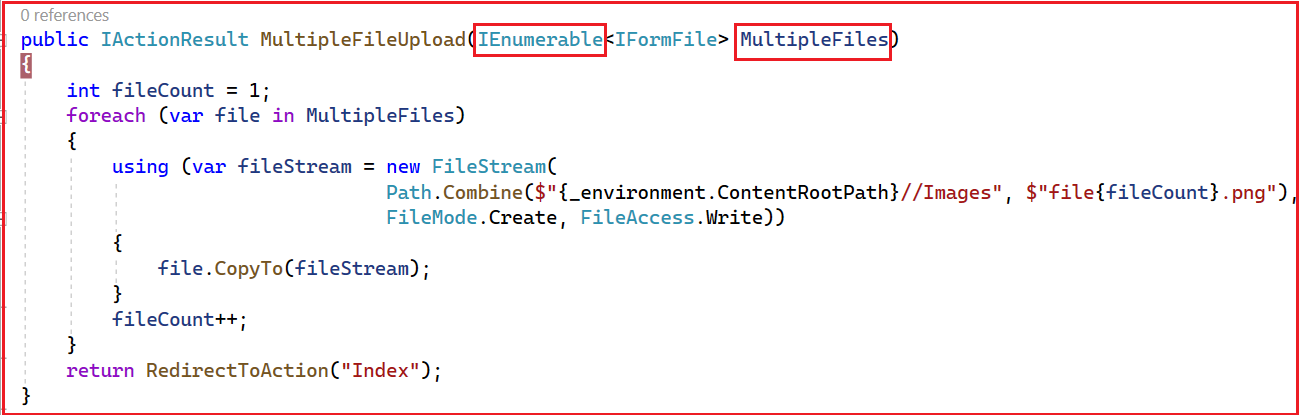


1. Give the name of the folder as “Images”, the folder will be created as shown below



##### **Modifying the Home Controller:**

Let’s update the “Home” controller to add a new action method “**MultipleFileUpload**”. In this method we iterate over all the files which are uploaded and save them one after another. We save the files to “Images” folder which we created in the previous step using the CopyTo(Stream) method of the IFormFile interface.



“

public IActionResult MultipleFileUpload(IEnumerable<IFormFile> MultipleFiles)

{

int fileCount = 1;

foreach (var file in MultipleFiles)

{

using (var fileStream = new FileStream(

Path.Combine($"{\_environment.ContentRootPath}//Images", $"file{fileCount}.png"),

FileMode.Create, FileAccess.Write))

{

file.CopyTo(fileStream);

}

fileCount++;

}

return RedirectToAction("Index");

}

”

**Notice that**

We have an input parameter of type “**IEnumerable<IFormFile>**” for the action method. The name of the input parameter is **MultipleFiles**. The name of the input parameter must match the name of the file type input control that we have added.

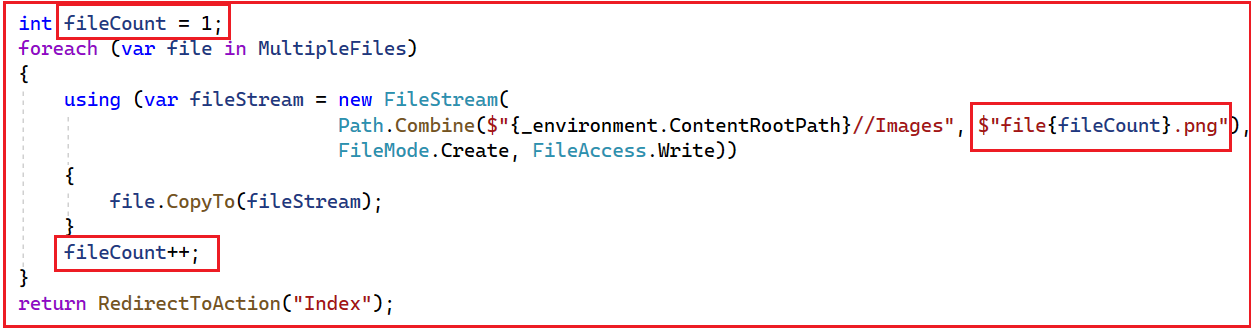


The input parameter “**MultipleFiles**” contains the list of all the uploaded files. We are iterating over the files one by one, and for each file we are creating a file stream that takes in the file path, file mode, and file access parameters.

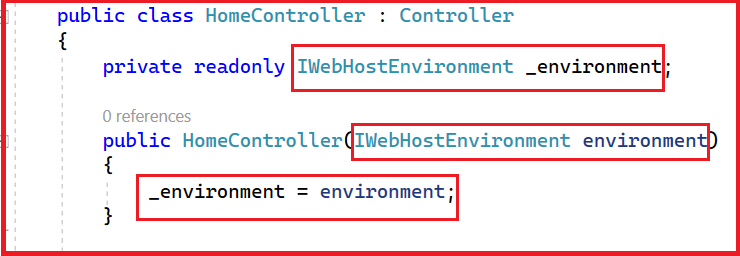
**Unique File Names**

To have unique file names for each file, we are creating a counter, an integer variable “**fileCount**”. Its initial value is 1 and we are incrementing it after processing each file.

For file name we are string interpolating “file” and fileCount variable value as highlighted below. So, the names of the saved files will be file1, file2, file3 ... etc.



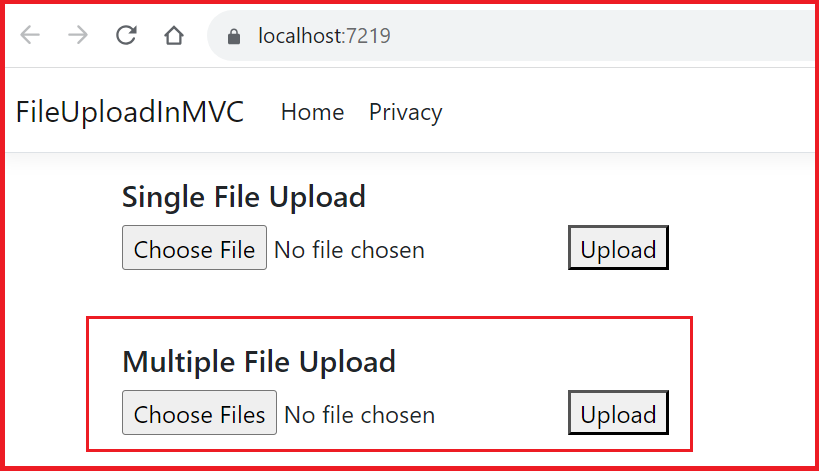
To get the content root path, we are using IWebHostEnvironment and getting the instance of it using dependency injection, as shown below.



To save the files in the images folder, we are appending folder name “Images” to the content root path as shown below

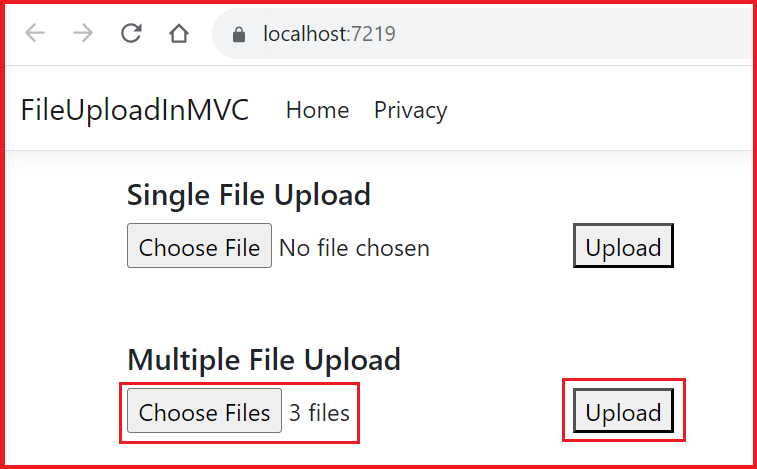


Now run the application, and the index view of the home controller will be displayed, as shown below. The index view has both “Single File Upload” form which we learnt in the previous article and the “Multiple File Upload” which we have added in this article.

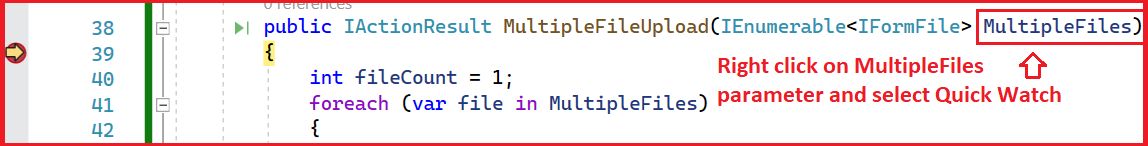


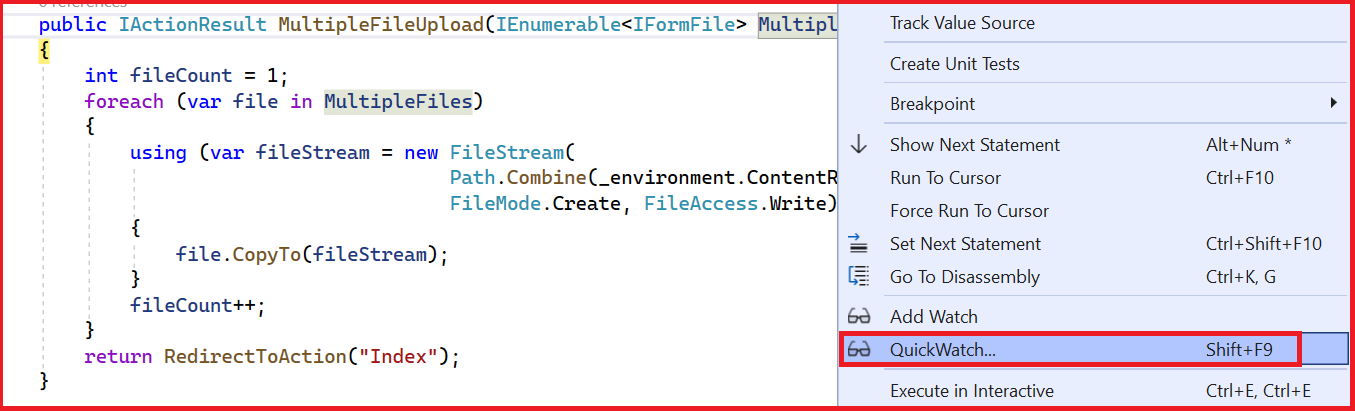
Let’s use the “Multiple File Upload” form to upload multiple files. Click on “Choose Files” and select multiple \*.png files.

After selecting the files, it will show the number of files chosen as shown below. Click on “Upload” button to process and save the uploaded files

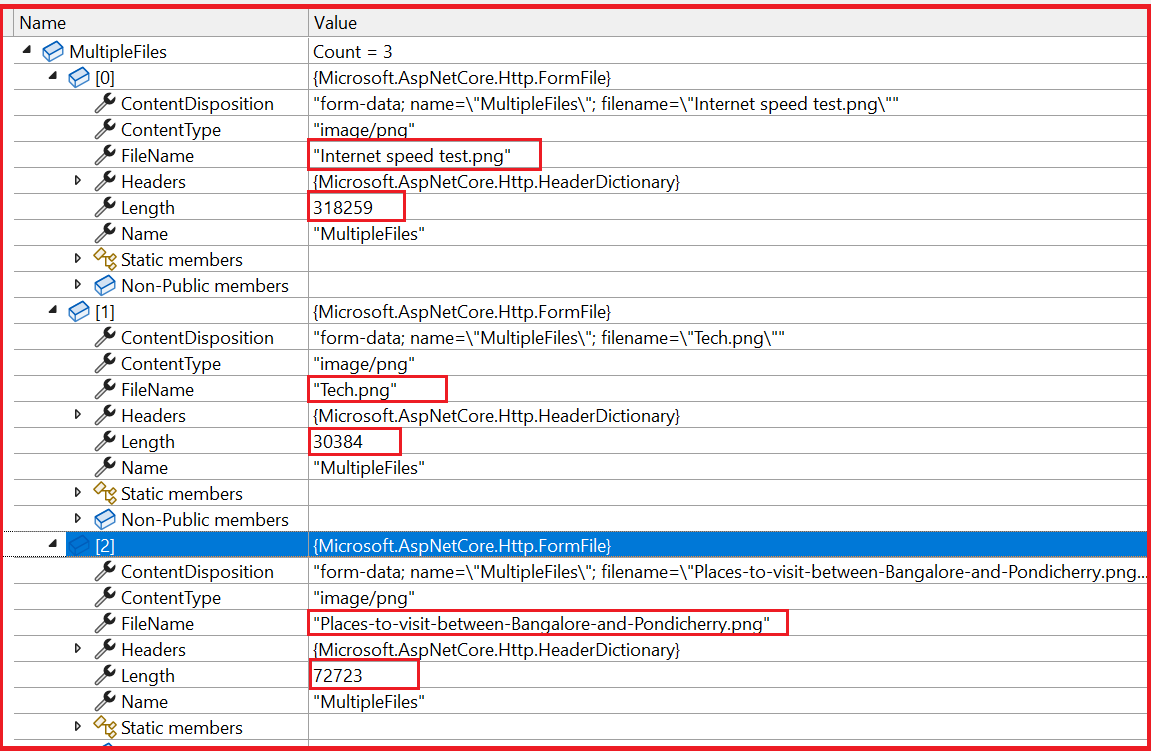


Please put a breakpoint at the entry of the **MultipleFileUpload** action method. Once you click on the Upload button, the **MultipleFileUpload** action method of the **HomeController** will be hit. Right-click on the **MultipleFiles** parameter and then select the QuickWatch option from the context menu, as shown in the below image.

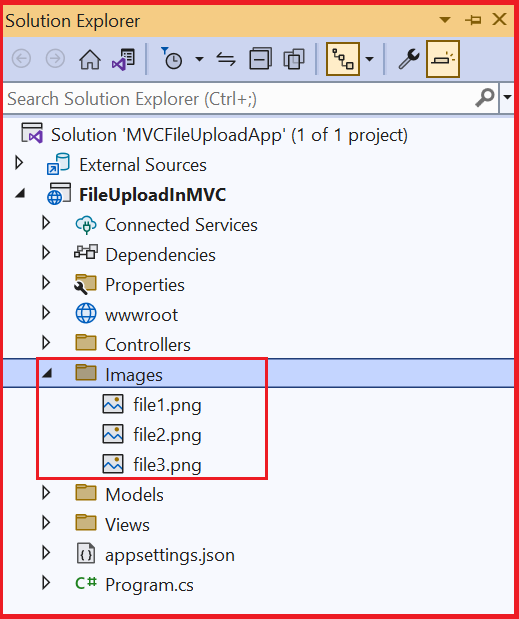




Notice it contains all the details about all the files we just uploaded, as shown below.



Click on F5 to continue with debugging. Once the file is copied to the specified path, the index page will be reloaded, and we should be able to see the uploaded file within the “Images” folder in the solution explorer, as shown below.



After saving the file, we re-direct the application to the index page.

