Lab 5 – Static Content and File Uploads

# Description

This lab allows you to start implementing other features of MVC: Static Content and File Uploads

# Estimated Time

This lab will take an estimated 2 hours to complete

# Deliverable

Deploy your website to Windows Azure and submit the link to Brightspace.

See “Brightspace -> Course Content -> Extra Materials -> Azure Usage” for information about deploying Azure Web Apps, Databases and Storage Accounts.

# Notes

* Be sure Visual Studio is up to date.
* Install SQL Server Management Studio Community Edition
* Follow along closely to the instructions!
* When using the <input> tag be sure to set the type to collect the proper values
* Use the example code found in Week 8 and Week 9 to complete this lab.
* Demo can be found at: <http://cst8359.hopto.org/lab5/>

# Create a new MVC Core project called ‘Lab5’

1. Open Visual Studio 2015
2. Click: File -> New -> Project
3. Click: Templates -> Visual C# -> Web, select ‘ASP.NET Core Web Application (.NET Core)’
4. Name the application ‘Lab5’, save the project in your desired location and click the button ‘OK’
5. Select the ‘Empty’ ASP.NET Core Template
6. Be sure to uncheck ‘Host in the cloud’
7. Click the ‘OK’ button.

# Configure your new Web Application

1. Modify the file ‘project.json’. Replace the section:

"dependencies": {

"Microsoft.NETCore.App": {

"version": "1.0.1",

"type": "platform"

},

"Microsoft.AspNetCore.Diagnostics": "1.0.0",

"Microsoft.AspNetCore.Server.IISIntegration": "1.0.0",

"Microsoft.AspNetCore.Server.Kestrel": "1.0.1",

"Microsoft.Extensions.Logging.Console": "1.0.0"

},

with:

"dependencies": {

"Microsoft.NETCore.App": {

"version": "1.0.1",

"type": "platform"

},

"Microsoft.AspNetCore.Diagnostics": "1.0.0",

"Microsoft.AspNetCore.Mvc": "1.0.1",

"Microsoft.AspNetCore.Mvc.TagHelpers": "1.0.1",

"Microsoft.AspNetCore.Server.IISIntegration": "1.0.0",

"Microsoft.AspNetCore.Server.Kestrel": "1.0.1",

"Microsoft.AspNetCore.StaticFiles": "1.0.0",

"Microsoft.Extensions.Logging.Console": "1.0.0",

"Microsoft.EntityFrameworkCore.SqlServer": "1.0.1",

"WindowsAzure.Storage": "7.2.1",

"Microsoft.Data.Edm": "5.7.0",

"Microsoft.Data.OData": "5.7.0",

"Microsoft.Data.Services.Client": "5.7.0"

},

1. Visual Studio should now update itself with the packages you need for this application to run.
2. Modify ‘Startup.cs’. Add the following lines to the method ‘ConfigureServices(IServiceCollection services)’”

var connection = @"user azure usage guide";

services.AddDbContext<PhotoDataContext>(options => options.UseSqlServer(connection));

services.AddMvc();

1. Modify ‘Startup.cs’. Replace the contents of ‘Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)’ with:

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseStaticFiles();

app.UseMvc(routes =>

{

routes.MapRoute(

name: "default",

template: "{controller=Home}/{action=Index}/{id?}");

});

# Create the ‘Controllers’ and ‘Views’ and ‘Models’ folders

1. Create a folder in your project called ‘Controllers’
2. Create a new Controller in a folder called ‘Home’
   1. NOTE: if you use ‘MVC Controller Class’ template from visual studio it will create all the basic code you need to have a valid controller
   2. To do so right click on the ‘Controllers’ folder, click ‘Add’, click ‘New Item’ and select the ‘MVC Controller Class’
3. Create a folder in your project called ‘Views’
4. At the root of this new ‘Views’ folder create a filed called ‘\_ViewImports.cshtml’
5. Add the following lines of code to the file ‘\_ViewImports.cshtml’

@using Lab5

@addTagHelper "\*, Microsoft.AspNetCore.Mvc.TagHelpers"

1. At the root of the ‘Views’ folder created a folder called ‘Home’
2. At the root of the project create a folder called ‘Models’

# Create the Model

1. At the root of the ‘Models’ folder create a file called ‘PhotoDataContext.cs’
   1. Use the example code located in Week 9 as an example.
   2. Make sure your context you include the Constructor and the DBSet to hold your photo objects.
2. At the root of the ‘Models’ folder create a file called ‘Photo.cs’
3. Add the following Properties to the ‘Photo.cs’ file
   1. int PhotoId
   2. string FileName
   3. string Url
4. Use SQL Server Management Studio and the following script to create the database:

USE [master]  
GO  
  
CREATE DATABASE [Lab5]  
GO  
  
USE [Lab5]  
CREATE TABLE Photos(  
 [PhotoId] [int] primary key IDENTITY(1,1) NOT NULL,  
 [FileName] [nvarchar](2048) NOT NULL,  
 [Url] [nvarchar](2048) NOT NULL  
)

# Create the Controller’s Actions and Views

1. Create the ‘HomeController’
2. In your HomeController create the ‘Index’ view
   1. NOTE: You can find an example of file uploads from the ‘File Uploads and Azure Storage’ (C#, Zip) file. Please be prepared to copy it exactly if need be.
   2. NOTE: You can find an example of ‘static web pages’ in the ‘Static Content (C#, Zip)’ file.
   3. The index view will be used to display and upload images collected by the user. Modify my code example to accept multiple files.
   4. Use CSS, in the form of static content, to change your output so it exactly matches my demo http://cst8359.hopto.org/lab5
3. Create a static web page called ‘Questions.html’
   1. NOTE: You can find an example of ‘static web pages’ in the ‘Static Content (C#, Zip)’ file.
   2. Display, using html, the answers to the following questions:
      1. List and describe 3 types of static content that can be used in MVC Core
      2. Where does static content belong in an MVC Core application?
      3. Explain how to link to a static page from a controller and/or view.

# Lastly

1. Be prepared to completely reuse (rip off) the code examples provided in the ‘Azure Storage’ and ‘Static Content’ projects.
2. Use the Azure Usage guide to get access to your storage account.