Lab 5 – Razor Pages, Static Content, Migrations, and File Uploads

Description

This lab allows you to start implementing other features of ASP.NET, Static Content, Migrations, and File Uploads in the context of Razor Pages app

Estimated Time

This lab will take an estimated 3 hours to complete

Notes

- Be sure Visual Studio is up to date.
- When using the <input> tag be sure to set the type to collect the proper values
- Demo can be found at: http://afrasialab5.azurewebsites.net/
    - Note, create, and delete buttons are intentionally disabled. Your submission should be functional though.

Create a new Razor Pages project called ‘Lab5’

1. Open Visual Studio 2022
2. Create new ASP.NET Core Empty called Lab5. Refer to Lab 4 if you forgot how to do that.
3. Install the following NuGet packages (refer to lab 4 if you forgot how to do this):

    a. “Microsoft.EntityFrameworkCore" v: 6.0.11 or newer (not versions 7.0.0 and above)
    b. "Microsoft.EntityFrameworkCore.Sqlite” v: 6.0.11 or newer (not versions 7.0.0 and above)
    c. "Microsoft.EntityFrameworkCore.Tools" v: 6.0.11 or newer (not versions 7.0.0 and above)
    d. “Microsoft.VisualStudio.Web.CodeGeneration.Design” v: 6.0.10 or newer (not versions 7.0.0 and above)
    e. “Microsoft.EntityFrameworkCore.SqlServer” v: 6.0.11 or newer (not versions 7.0.0 and above)
    f. “Azure.Storage.Blobs” v: 12.14.1 or newer

Configure your new Web Application

1. Open your project file, and comment out (or delete) the line: <Nullable>enable</Nullable>

2. At the root of the project create a folder called ‘Models’
3. At the root of the project create a folder called ‘Data’
4. Clear the content of the Program.cs file and replace it with the code at: https://gist.github.com/aarad-ac/a99b73d548db21b1151e7a16a172b2d4
5. Then replace /*LabName*/ with Lab5
6. Replace /* Define services here */ with the following code. Ignore errors around `AnswerImageDataContext` and fix the rest by adding the right ‘using’ statements (you are on your own):

```
          builder.Services.AddRazorPages();

var connection = builder.Configuration.GetConnectionString("DefaultDBConnection");
builder.Services.AddDbContext<AnswerImageDataContext>(options => options.UseSqlServer(connection));

var blobConnection = builder.Configuration.GetConnectionString("AzureBlobStorage");
builder.Services.AddSingleton(new BlobServiceClient(blobConnection));
```

7. Ignore the syntax errors around ` AnswerImageDataContext ` but fix the rest of them by adding the right ‘using’ statements (you are on your own)
8. Modify the Program class and replace /* Define routing here */ with the right routing method. You are on your own.
9. See the Azure SQL document in Brightspace -> extra materials to set up your db and find your database connection string
   a. Like lab 4, it is recommended that you start with a local db and then switch to Azure SQL as one of the final steps
10. See the Azure Storage document in Brightspace -> extra materials to set up your blob storage and find your database connection string
11. Modify both appsettings.Development.json and appsettings.json and add the following lines right before “Logging”, replacing `Red text` with your appropriate connection strings. Do not remove the quotes. Replace password placeholders with the right password where needed.

```
"ConnectionStrings": {
    "DefaultDBConnection": "AZURE SQL CONNECTION STRING GOES HERE",
    "AzureBlobStorage": "AZURE STORAGE CONNECTION STRING GOES HERE"
},
```

Create the ‘wwwroot’, ‘Pages’ and ‘Models’ folders

1. At the root create wwwroot folder and add a css file there. You can use the one I posted with the sample code
2. Create a folder in your project called ‘Pages’

3. At the root of this new ‘Pages’ folder, create a file called ‘_ViewImports.cshtml’
4. Add the following lines of code to the file ‘_ViewImports.cshtml’

```
@using Lab5
@namespace Lab5.Pages
@addTagHelper "*, Microsoft.AspNetCore.Mvc.TagHelpers"
```

5. Under Pages, create your layout in the folder called shared
6. At the root of this new ‘Pages’ folder, create a file called ‘_ViewStart.cshtml’
7. Add the following lines of code to the file ‘_ViewStart.cshtml’

```
@{
        Layout = "_Layout";
}
```

8. At the root of the ‘Pages’ folder created a Razor Page called ‘Index.cshtml’
    a. Right click on Pages
    b. Add -> Razor Pages
    c. Select “Razor Pages- Empty”
    d. Again, select “Razor Pages- Empty”
    e. Change the Page name, and click add
9. Create a Page called Error.cshtml under Pages. It should contain some error message

Create the Model

1. At the root of the ‘Models’ folder, create a file called ‘AnswerImage.cs’
2. Add the following Properties to the ‘AnswerImage.cs’ file
    a. int AnswerImageId
    b. string FileName
    c. string Url
    d. Question Question
        i. Note that the type is a Question. Question should be an enum that has two values “Earth” and “Computer”. If you need a sample code refer to EntityFramework sample and check how we define driver’s license.
    e. Add proper attributes to the entity properties you created above. Based on your knowledge of ASP.NET so far, you should be able to identify minimum requirements by now. The attributes should help with prober validation of model. You are on your own for this.
3. At the root of the ‘Data’ folder create a file called ‘AnswerImageDataContext.cs’
    a. Use the example code located in Week 9 slides or lab 4 for guidance.

    b. Make sure in your context you include the Constructor and the DBSet to hold your AnswerImage objects.

4. Add a migration for initial creation of database
    a. In the ‘Package Manager Console’ type: Add-Migration InitialCreate
    b. Hit ‘Enter’
5. Apply the migration to DB
    a. In the ‘Package Manager Console’ type: Update-Database
    b. Hit ‘Enter’
6. Validate your database is created, and the AnswerImage table has the desired columns
    a. You can do this on Azure or from Visual Studio
    b. If DB is not created, start to troubleshoot

Create the Pages

1. Under Pages, create a new folder called AnswerImages
2. Right click on AnswerImages folder, and Add New Scaffolded Item
    a. Select Razor Pages using Entity Framework (CRUD)
3. From the scaffolded pages, delete the Details and Edit pages
4. Open the page models for Create and Delete pages and add the following lines to the beginning of both classes.

```
 private readonly BlobServiceClient _blobServiceClient;
 private readonly string earthContainerName = "earthimages";
private readonly string computerContainerName = "computerimages";
```

5. For the Create and Delete page models, modify the constructor, add this parameter to their signatures:

```
        BlobServiceClient blobServiceClient
```

6. and add the following line to their body:

```
        _blobServiceClient = blobServiceClient;
```

7. On both Create and Delete page models, modify the OnPostAsync method to handle Uploading and Deleting the images respectively
    a. Review the lecture on file uploads and check the sample code for more hints
8. Modify the Create Page to enable the user to upload photos
9. Modify Razor code on Index and Delete pages to show the images as required

Publish

1. Follow the instructions provided in lab 4 for publishing

2. Additionally, make sure 'Storage' appears under 'Service Dependencies'. You do not need to configure it

Push to Git

1. Push your code to git
2. Note: No PASSWORDs should be pushed to Git. You can add your appsettings to git ignore, so they never get pushed to git.
    a. Just create a file called'`.gitignore'`
    b. And then add the following lines to it:

    appsettings.json

    appsettings.Development.json

Deliverable

1. Answer the following questions by pictures:
    a. What will the computers look like in 20 years?
    b. What will the earth look like in 20 years?
2. Upload your answers to your website. Now, there should be two photos in your AnswerImages/Index page. Delete any extra photos you might have there.
3. Submit to brightspace:
    a. Link to your Git repo.
    b. Link to your published website

Marking Scheme:

program	2
viewimports	1
viewstart	1
layout	2
index	1
Error Page	1
structure	1
model	6
context	2
migration	2
Create Page Model	6

Delete Page Model	3
Create Page	4
images uploaded	2
gitignore	3
working assignment	3
Total	40