Assignment 1 – Blog Engine

# Description

This assignment builds off the techniques learned in the first 4 labs of the course and allows you to get creative and create a Blog.

# Estimated Time

This lab will take an estimated 20 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

* Read the entire assignment before you get started.
* An SQL Script is provided. The script contains the SQL needed to generate your database.
* Use Lab 4 as a starting point.
* Be sure to have completed lab 4 before attempting this assignment. You will find it very difficult and waste a lot of time otherwise.
* No seriously, read the entire assignment first!
* The project/code named ‘ASP.NET MVC Core and the Entity Framework’ will be referred to as the ‘The Example’ from now on.
* Work in teams if need be. This application framework can be intimidating when you are starting out – use all the help you can get without blatantly cheating.
* Blatant cheaters will be prosecuted.
* I highly recommend using the ‘Bootstrap’ CSS framework. It will help provide basic styling to your application without being impossible to understand and implement.
* Demo provided at: <http://cst8359.hopto.org/assignment1>

# Create your database

1. Use the Azure Usage guide to create a new database on Windows Azure

# Create a new MVC Core project called ‘Assignment1’

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 ‘Assignment1’, 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.Server.IISIntegration": "1.0.0",

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

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

"Microsoft.AspNetCore.Session": "1.0.0",

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

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

"Microsoft.Extensions.Caching.Memory": "1.0.0",

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

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 = @"use Azure Usage guide";

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

services.AddMvc();

services.AddMemoryCache();

services.AddSession();

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

loggerFactory.AddConsole();

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseStaticFiles();

app.UseSession();

app.UseMvc(routes =>

{

routes.MapRoute(

name: "default",

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

});

# Create the ‘Home’ Controller

1. Create a folder in your project called ‘Controllers’
2. Create a new Controller in a folder called ‘Home’
   1. 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’

# Create the basic ‘Views’

1. Create a folder in your project called ‘Views’
2. In the ‘Views’ folder create a folder called ‘Home’
3. In the ‘Views’ folder create a folder called ‘Shared’
4. In the ‘Views’ folder create a View called ‘\_ViewImports.cshtml’
   1. Copy the contents of this same file from ‘The Example’
5. In the ‘Shared’ folder create a View called ‘\_Layout.cshtml’
   1. If you use ‘MVC View Layout Page’ template from visual studio it will create the basic file template for you.
   2. To do so right click on the ‘Shared’ folder, click ‘Add’, click ‘New Item’ and select the ‘MVC View Layout Page’ template.
6. In the ‘Home’ folder create a view called ‘Index.cshtml’ that uses the layout file.

# Create the ‘Model’

1. Create a folder in your project called ‘Models’
2. In the ‘Models’ folder create a class called ‘Assignment1DataContext’
   1. Use the contents of data context in ‘The Example’ as a guide.
   2. Note, you will need to add a ‘DBSet<T>’ collection for each of the tables in your model.
3. Create the Models for your application.
   1. The SQL script your ran against your server created a database with several tables. Use the columns created by the script to create your model objects
   2. Your application will have 4 Model classes
      1. Role
         1. RoleId of type int
         2. Name of type string
      2. User
         1. UserId of type int
         2. RoleId of type int and using the foreign key attribute
         3. FirstName of type string
         4. LastName of type string
         5. EmailAddress of type string
         6. Password of type string
      3. BlogPost
         1. BlogPostId of type int
         2. UserId of type int and using the foreign key attribute
         3. Title of type string
         4. Content of type string
         5. Posted of type DateTime
      4. Comment
         1. CommentId of type int
         2. BlogPostId of type int and using the foreign key attribute
         3. UserId of type int and using the foreign key attribute
         4. Content of type string
   3. Bonus points if you explore ‘Navigation Properties’ and implement them in your data context. (Hint, you will be able to write code like ‘user.BlogPosts’ or ‘blogPost.Comments’ and it will dramatically speed up development time)

# Application Logic

From here on in I will describe functionality and it will be up to you implement it the best you see fit.

## Views

1. \_Layout.cshtml
   1. Your layout file will have a link to both a ‘Register’ view and a ‘Login’ view.
   2. If the user is logged in display their First and Last name on the page somewhere.
   3. If the user is an administrator display a link to the ‘AddBlogPost’ view.
   4. Your layout file should contain your First Name, Last Name, Email Address and Student Number in the footer.
   5. I encourage you to create a theme using either custom CSS, or a framework like bootstrap.
2. Home Controller / Register
   1. This view will allow users to create ‘User’ accounts to be stored in your database.
   2. Create a form that creates a User model: FirstName, LastName, EmailAddress, Password and Role
   3. Once you have collected the data from the client, store it in the database and redirect the user to the Login view
   4. Note, I know this is not standard practice, but allow this screen to create both ‘Admin’ users and ‘General Users’. This is simply so I can test the application easily.
3. Home Controller / Login
   1. This view will allow the users to log in to your application.
   2. Create a form that accepts the users email address and password.
   3. Once the form is submitted authenticate the user. If the user is valid redirect them to the ‘Index’ view
   4. Note store the ‘UserId’ of the user in the Session so you can retrieve it for other subsequent views.
4. Home Controller / Index
   1. This view will list all the BlogPosts
   2. You will display ‘Title’, the first 100 characters of the ‘Content’ as a preview and the ‘Posted’ fields from each BlogPost object in your database
   3. The Title of each blog post will be a link to the ‘DisplayFullBlogPost’ view
   4. If the user is logged in as an administrator each blog post will display a link to the ‘EditBlogPost’ view as well as the other requirements above.
5. Home Controller / DisplayFullBlogPost
   1. This view will list the Title, the Content, the Posted value, the email address and full name of the users who created the post.
   2. If the user is logged in they should be able to comment on the blog post using a comments text box at the bottom of the screen.
   3. Display all comments associated with the blog post.
6. Home Controller / AddBlogPost
   1. This view will collect the data needed to create a blog post.
   2. Use a form to collect Title, Content, Posted (date time)
7. Home Controller / EditBlogPost
   1. This view will allow an administrator to modify the data of a selected blog post.
   2. Use a form to collect the modified Title, Content, Posted (date time) content.

# Bootstrap

Use your experience from Lab 4b to style the application using Bootstrap. There are many examples of Bootstrap themed blogs, yours should have similar functionality.

For example: <https://blackrockdigital.github.io/startbootstrap-blog-post/>

# Lastly

1. Ask for help. I am quick about responding to questions.
2. Be sure to get this done. Assignment 2 is going to be an expanded version of this application.
3. Don’t rush through it trying to meet a deadline – get it right the first time so you don’t have to kill yourself trying to get Assignment 2 done.
4. Be prepared to complete reuse (rip off) the code provided in ‘The Example’
5. I bet you didn’t read to here before starting!