Assignment 2 – Blog Engine Expanded

# Description

This assignment builds off the content you have learned throughout this course.

# Estimated Time

This lab will take an estimated 20 hours to complete

# Deliverable

Demo your completed application to your professor before the conclusion of the course. Please be prepared to demo your application on your laptop and be prepared to present your code if asked.

# 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 Assignment 1 as a starting point.
* Be sure to have completed lab 4 and lab 5 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.

# Create your database

1. Open the SQL Server Management Studio
2. Connect to your local database server. The host will either be ‘localhost’ or ‘localhost\SQLEXPRESS’. Use local authentication if configured, otherwise us the ‘sa’ account you had to set up while installing the server.
3. Once connected click: File -> Open -> File.
4. Navigate to the script provided: CreateAssignment2Database.sql
5. Execute the script
6. Using the object explorer expand the ‘Databases’ + symbol and be sure that your database was created correctly. (Note, you may have to right click on databases and click ‘Refresh’)

# Create a new MVC Core project called ‘Assignment2’

1. Create a copy of your completed Assignment 1 project and rename it Assignment2

# 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.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"

},

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"

"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 = @"Server=localhost;Database=Assignment2;Trusted\_Connection=True;MultipleActiveResultSets=true";

// if that fails try: var connection =

// @"Server=localhost\SQLEXPRESS;Database=Assignment2;Trusted\_Connection=True;MultipleActiveResultSets=true";

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 ‘Assignment2DataContext’
   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 6 Model classes
      1. BadWord
         1. BadWordId of type int
         2. Word of type string
      2. Role
         1. RoleId of type int
         2. Name of type string
      3. 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
         7. DateOfBirth of type DateTime
         8. City of type string
         9. Address of type string
         10. PostalCode of type string
         11. Country of type string
      4. BlogPost
         1. BlogPostId of type int
         2. UserId of type int and using the foreign key attribute
         3. Title of type string
         4. ShortDescription of type string
         5. Content of type string
         6. Posted of type DateTime
         7. IsAvailable of type bool
      5. 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
         5. Rating of type int
      6. Photo
         1. PhotoId of type int
         2. BlogPostId of type int and using the foreign key attribute
         3. Filename of type string
         4. Url 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 logged in display a link to the ‘EditProfile’ view.
   4. If the user is an administrator display a link to the ‘AddBlogPost’ view.
   5. If the user is an administrator display a link to the ‘ViewBadWords’ view.
   6. Your layout file should contain your First Name, Last Name, Email Address and Student Number in the footer.
   7. 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 and collects the following fields:
      1. FirstName, LastName, EmailAddress, Password, Role, DateOfBirth, City, Address, PostalCode and Country
   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 we can test the application easily.
3. Home Controller / EditProfile
   1. This view will a user to modify the details collected in the ‘Register’ view.
   2. This view can only be accessed after the user logs into the application.
4. 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.
5. Home Controller / Index
   1. This view will list all the BlogPosts
   2. You will display ‘Title’, the ShortDescription 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. If the user is logged in as an administrator each blog post will display a link to delete the blog post from the database.
6. Home Controller / DisplayFullBlogPost
   1. This view will list the Title, Content, Posted value, the email address and full name of the users who created the post and all the images associated with the blog 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.
      1. When a user posts a comment, the comment should be scanned for Bad Words listed in the BadWords table of your database. If a word is found replace that word content with \*\*\*\*\*
   3. Display all comments associated with the blog post.
7. Home Controller / AddBlogPost
   1. This view will collect the data needed to create a blog post.
   2. Use a form to collect Title, ShortDescription, Content, Posted (date time), IsAvailable
   3. This view can only be accessed after the user logs into the application.
8. 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, ShortDescription, Content, Posted (date time), IsAvailable content.
   3. This view will also allow a user to add or delete images for a blog post.
   4. This view can only be accessed after the user logs into the application.
9. Home Controller / ViewBadWords
   1. This view will allow an administrator to define a set of restricted words that will be ‘starred-out’ when a user posts a comment.
   2. This page will contain:
      1. A form at the top of the page to submit new ‘Bad Words’
   3. A list of the bad words that are currently in the database. The administrator should be able to delete bad words.
   4. This view can only be accessed after the user logs into the application.

# Blog Post Questions

1. To demo the assignment, you must research the following 5 Windows Azure technologies and create a blog post about each subject:
   1. Service Fabric
   2. Machine Learning
   3. Azure IoT Hub
   4. Table Storage
   5. Virtual Machines
2. Each blog post should be a detailed description of the Azure service

# 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 prepared to complete reuse (rip off) the code provided in ‘The Example’ and ‘File Uploads and Azure Storage’
3. I bet you didn’t read to here before starting… again.