Lab 6 – Web API

Description

This lab allows you to start implementing other features of MVC: Web API

Estimated Time

This lab will take an estimated 2 hours to complete

Notes

- Application Demo can be found at: http://afrasialab6.azurewebsites.net/
- I intentionally blocked all actions other than GET, but your service should have a working version for all the verbs.
- See "Brightspace -> Course Content -> Extra Materials -> Azure Usage" for information about deploying Azure Web Apps, Databases and Storage Accounts.

Create a new MVC Core project called 'Lab6'

1. Create new ASP.NET Core Empty Application called Lab6.
    a. Uncheck the HTTPS box
    b. Refer to previous labs if you forgot how to do this
2. 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. "Swashbuckle.AspNetCore" >= v:6.4.0 (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 'Data'
3. Under Data folder, create a class 'DbInitializer.cs':

a. Replace the class with:

```
public static class DbInitializer
 {
     public static void Initialize(StudentDbContext context)
     {
         context.Database.Migrate();

     if (context.Students.Any())
      {
          return;   // DB has been seeded
      }

      var students = new Student[]
      {
      new Student{FirstName="Carson",LastName="Alexander",Program="ICT"},
      new Student{FirstName="Meredith",LastName="Alonso",Program="ICT"},
      new Student{FirstName="Arturo",LastName="Anand",Program="ICT"},
      new Student{FirstName="Gytis",LastName="Barzdukas",Program="ICT"},
      };
      foreach (Student c in students)
      {
          context.Students.Add(c);
      }
         context.SaveChanges();

    }

}
```

Note: you are getting errors with '`StudentDbContext`' and '`Student`' at this point. Ignore it until you finish creating the model.

4. Clear the content of the Program.cs file and replace it with the code at: https://gist.github.com/aarad-ac/ed724e99a3dfae2256ab04a65ed9eb24
5. Ignore the syntax errors around ` `StudentDbContext` ` but fix the rest of them by adding the right 'using' statements (you are on your own)
6. See the Azure SQL document in Brightspace -> extra materials to set up your db and find your database connection string
7. 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.

```
"ConnectionStrings": {
    "DefaultConnection": "AZURE SQL CONNECTION STRING GOES HERE"
```

```
        },
```

8. Modify `Program.cs'. replace /* Define services here */ with the following code. Ignore errors around `StudentDbContext` and fix the rest by adding the right 'using' statements (you are on your own):

```
            var connection = builder.Configuration.GetConnectionString("DefaultConnection");
             builder.Services.AddDbContext<StudentDbContext>(options => options.UseSqlServer(connection));
             builder.Services.AddControllers();
    builder.Services.AddSwaggerGen(c =>
    {
        c.EnableAnnotations();
    });
```

9. Modify 'Program.cs'. Replace /* Define routing here */ with the appropriate code (you are on your own)

Create the 'Controllers' and 'Models' folders

1. Create a folder in your project called 'Controllers'
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 'Student.cs'
    a. This should define a model that has the following fields. Use appropriate attributes for the fields. You should already know how to do this. You are on your own.
        i. It should define an ID field of type Guid, that gets autogenerated by database. Refer to the Entity Framework lecture. You are on your own.
            1. Annotate the ID with this: `[SwaggerSchema(ReadOnly = true)]`
        ii. FirstName
        iii. LastName
        iv. Program
2. At the root of the 'Data' folder create a file called 'StudentDbContext.cs'
    a. You are on your own
3. Add migration for initial creation of database and apply it to database. If you forgot how to do this, refer to lab 5.
4. At this point all the compile errors you previously ignored should be resolved.

Scaffold the Controller's Actions

1. Right click on the Controllers folder, then select:

   a. Add > New scaffolded item..
   b. API Controller with actions using Entity Framework
   c. Scaffold Student model
2. In the Scaffolded controller, modify the action responding to PUT to return a Student when the request is successful
3. In your StudentsController annotate every action with the right ProducesResponseType annotations and explain why you added that annotation in a comment. For instance in 'Get' action

```
        [HttpGet]
        [ProducesResponseType(StatusCodes.Status200OK)] // returned when we return list of Students successfully
        [ProducesResponseType(StatusCodes.Status500InternalServerError)] // returned when there is an error in processing
the request
         public async Task<ActionResult<IEnumerable<Student>>> Get()
         {
               return Ok(await _context.Students.ToListAsync());
         }
```

Publish

1. Refer to previous labs for publishing instructions. Note that there is one additional step to the process. On the "API Management" set up step, check the "Skip this step" and hit finish.

Deliverable

Deploy your website to Windows Azure and push the code to GitHub. Submit both links to Brightspace.

Answer the following questions with your submission to Brightspace (as text submission):

1. Define each HTTP Status Code that you used in this project and explain what each code means.
2. Does the service you created in this assignment conform to all REST principals? Explain why.

Grading Scheme

model	6
context	2
migration	1
Controller	15
dbinitializer	2
q1	12
q2	5

gitignore	2
working assignment	5
total	50