W2D3 – Intro to Models in Razor Pages.

Getting started.

1. Create a new project called “RazorPages”
   1. We will be using this project for the rest of our sessions on Razor Pages.
2. Note the wwwRoot folder. This houses static content.
3. Note the pages folder. This houses our dynamic pages.
   1. In this folder note \_Layout.cshtml. this file allows all pages to be consistent.
   2. Start the page via ctrl+f5, and check home, about and contact. Note how these share a similar layout. This is the work of \_layout.cshtml.
4. Note Startup.cs. This is the file that configures the request pipeline.
   1. It will be where we set up databases and things like that
   2. For more information see: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/startup
5. Note Program.cs This is the file that runs our site.

Adding a Model

1. Right click “RazoPages” in the solution explorer
   1. Select add -> new folder.
   2. Name the folder “Models”.
2. Right click the “Models” folder and select add class.
   1. Name the class Students
   2. Add a public int called ID, with `{ get; set;}`
      1. This field is required by the DB as the primary key.
   3. Add a public string called FirstName
   4. Add a public string called LastName
   5. Add a public int called year.
   6. Make sure each of these has the “{get; set;}” at the end.
3. Add another class called “StudentContext”
   1. Add “using Microsoft.EntityFrameworkCore” to the top.
   2. Add “: DbContext” after “public class StudentContext”
      1. This makes the class derived from DbContext, which is contained within the Microsoft.EntityFramworkCore class
   3. Add a constructor.
      1. This is simply the name of the class and a couple of parenthesis
   4. Within the constructor’s parenthesis add DbContextOptions<StudentContext> options.
   5. After you need “: base(options);”
   6. You now need a public variable called “Student” of the class DbSet that’s of type Student.
      1. DbSet is a strongly typed array class, like how we strongly typed DbConextOptions with Student context, this is a similar idea.
   7. We’ve now created an entity set. In Microsoft’s entity framework terminology the entity set corresponds to the database table and the entity itself (in this case the student) corresponds to a row in that table.
4. Setting up the database;
   1. Add the following code to “appsettings.json”

"ConnectionStrings": {

"StudentContext": "Server=(localdb)\\mssql<localdb;Database=Student-1;Trust>ed\_Connection=True;MultipleActiveResultSets=true"

}

* 1. In Startup.cs at the top add “using Microsoft.EntityFrameworkCore;” and “using RazorPages.models;”
     1. These are needed for the code we are about to add.
  2. Now go down to the function called configure services.
     1. Add to services the DbContext.
     2. In the parentheses, you need options then a “=>” (pointer operator)
     3. Then you need options.useSqlServer(configuration.getConnectionString(“StudentContext”))
     4. This should set up things for the scaffold tooling to preform the initial migration.

1. Scafolding.
   1. Go to tools -> Nuget Package Manager -> Package Manager Console.
      1. Commands you need to know
      2. Install-package installs packages, we are going to use it to install the scaffolding tooling.
      3. Add-migration generates code and migrates things…
      4. Update-database updates the database
   2. In the package manager console type: Install-Package Microsoft.VisualStudio.Web.CodeGeneration.Design
   3. Then type in : Add-Migration Initial
   4. Next type Update-Database
   5. If you did this correctly you should now have a “migrations” folder in your solution explorer
   6. Now we need to open command prompt (it will be easier if you go into the folder where startup.cs is then press shift + left click or shift+ application menu key you should be able to open CMD, if not, open PS and then type CMD)
   7. In the directory that has your startup.cs file in it, type: dotnet aspnet-codegenerator razorpage -m Student -dc StudentContext -udl -outDir Pages\Students --referenceScriptLibraries
   8. Once that runs it will have created 5 files to edit and interact with the database
2. Testing
   1. Press ctrl+f5
      1. If anything is broken fix it.
      2. Common issues:
         1. if it won’t build, check your DbSet and check the autogenerated files for constancy
         2. if you are getting an sql error, open the sql explorer and double check that your database name is the same as the DbSet’s name.
         3. any other issues, ask me.
   2. When it launches, navigate to localhost:xxxxx/students (simply add the “/students” to the end of the url)
   3. create a new student.
   4. Repeat “c” a few times.
   5. Edit a couple of entries.
   6. View a few… then delete a few
   7. If all these changes stick, congrats it works… otherwise… ask me.

Questions:

1. What did you learn?
2. Do you see any inherent issues with this database?
3. How would you fix these issues?
4. Create your own datatype in a new project to test what you have learned.