# : Create QAForum App

## Objective

In this lab we use the understandings so far to create a web app using an EntityFramework database. The Web App is an online forum, so consists of Forums, Threads and Posts

This exercise will take around minutes.

|  |  |
| --- | --- |
| 1 | Using Visual Studio create a new ASP.NET Core Web App (Model-View-Controller) called “QAForum”.  Select the “.NET 6.0(Long Term Support)” framework and  Set Authentication Type to “Individual Accounts” and  Ensure there is a tick in the “Do not use top-level statements” checkbox. |
| 2 | Add a new folder called ViewModels to the QAForum project. |
| 3 | Add a new controller to Controllers folder. Call the controller ForumController, and use the MVC Controller – Empty template.   |  | | --- | | public class ForumController : Controller | |
| 4 | Right-click on the Index action, and add a view. Leave the template set to “Razor View – Empty” and just create a default view. We will update this to something more useful soon. |
| 5 | Let’s customise our website’s appearance. Open Views/Home/Index.cshtml. Change the line with the link so that it links to our forums:   |  | | --- | | <div class="text-center">  <h1 class="display-4">Welcome</h1>  <p>Please <a **asp-controller**="Forum" **asp-action**="Index">click here to see the Forums</a></p>  </div> |   Now, open Views/Shared/\_Layout.cshtml, and add a link in the navigation bar so that users can get to the forums:   |  | | --- | | <ul class="navbar-nav flex-grow-1">  <li class="nav-item">  <a class="nav-link text-dark"**asp-area**=""**asp-controller**="Home"**asp-action**="Index">Home</a>  </li>  <li class="nav-item">  <a class="nav-link text-dark"**asp-area**="" **asp-controller**="Forum" **asp-action**="Index">Forums</a>  </li>  <li class="nav-item">  <a class="nav-link text-dark"**asp-area**=""**asp-controller**="Home"**asp-action**="Privacy">Privacy</a>  </li>  </ul> | |
| 6 | |  | | --- | |  |   Press F5. When the home page appears, click on the link and the navbar item to make sure they appear to be working. Not much to see yet but hopefully there are no errors. |
| 7 | In the Assets folder we have provided the 3 model files in a Models folder. Drag and drop these into the Models folder of the QAForum app. |
| 8 | Take a few moments to have a look at these model files. They represent the data we will be dealing with and will need to persist into a database. |
| 9 | There is also an EF folder. Drag this onto your project and look at the ForumDbContext.  You will need to add another Nuget package. Right click Dependencies, choose “Manage Nuget Packages...”  On the Browse tab search for “Microsoft.EntityFrameworkCore.Proxies” and choose the version which matches the other EntityFramework Packages which are already installed. (It was version 6.0.14 in the “End” solution).  Install this package. |
| 10 | Now go to the Main method in the Program.cs file. Note how SqlServer is configured for the authentication (the “DefaultConnection”). Using this as your template, do the same for ForumDbContext, using the connection string name “Forum.Data”.So it looks like this:  var connectionString2 =builder.Configuration.GetConnectionString("Forum.Data") ?? throw new InvalidOperationException("Connection string 'Forum.Data' not found.");  builder.Services.AddDbContext<ForumDbContext>(options =>  options.UseSqlServer(connectionString2)); |
| 11 | Change “DefaultConnection” to “Forum.Users” |
| 12 | In appsettings.json, change the connection string name “DefaultConnection” to “Forum.Users” and its value to this:   |  | | --- | | "Forum.Users":"Server=.\\SqlExpress;Database=Forum.Users;Trusted\_Connection=True; MultipleActiveResultSets=true" |   NB the connection string here is only across 2 lines here because of the width constraint of an A4 page. Make sure when you copy this line that it does not have line breaks in it. |
| 13 | Copy this connection string and use it to add another connectionString and change the name to “Forum.Data”; change the database to “Forum.Data” |
| 14 | We will now use EntityFramework Migrations to generate the required Databases.  On the Tools menu go to Nuget Package Manager and click Package Manager Console. Create the Forum.Users database using the following command:-    Nb. This Database would be used should we want to add authentication capabilities to our app. This is beyond the scope of our course. |
| 15 | Add an initial migration for the Forum.Data database using this command. We will be using this database    This will generate code that will create the Forum.Data database and the required schema.  Next we are going to create and populate the DB with some sample data. |
| 16 | Add to the solution a C# Console App (.NET Core) called “SeedDatabase”. |
| 17 | In the SeedDatabase project, install the NuGet packages :  Microsoft.EntityFrameworkCore.SqlServer  Microsoft.EntityFrameworkCore.Tools  Again choose the versions that match the ones used before. 6.0.14 in the “End”.  And add a reference to QAForum.  Ensure all compiles. |
| 18 | In the Assets folder is a file Program.cs. Drag this over the one you have in the SeedDatabase project. This project will create the database for us by running the Entity Framework migrations. It will also add some sample data that we can use throughout the development process.  (Note that Entity Framework has a built-in technique for seeding data, which is not covered on this course, and also would not be suitable here, because this seed data is only intended to be used during development, and will not be included when we release our system. If you need to add data to your production database as a one-off process when the database is created, you can find out more at <https://docs.microsoft.com/en-us/ef/core/modeling/data-seeding>.) |
| 19 | Right-click on the SeedDatabase project, and select Set As Startup Project |
| 20 | Press F5 to run the SeedDatabase program – then, using SSMS, check that your Forum.Data database has Forum, Thread and Post records in it.  Change the startup program back to QAForum. |
| 21 | Now, let’s show some data. Modify the ForumController so that it gets an instance of the data context injected into it. Then pass the list of controllers into the view:   |  | | --- | | Public class ForumController : Controller  {  private readonly ForumDbContext context;  public ForumController(ForumDbContext context)  {  this.context = context;  }  public IActionResult Index()  {  return View(context.Forums);  }  } |   Then, right-click on the Index action, and select Add View. Choose Razor View (NOT the empty one this time) and the appropriate options… see if you can work out what they are before you look down. (You will need to confirm that you want to overwrite the existing view.) |
| 22 | Did you choose the List template, and the Forum class? If so, well done! Run the program, and check you can see the list of forums! |
| 23 | In an earlier lab, we saw that using a view-model gives us much more control over how a specific view looks. Let’s reinstate the view-model class from that earlier lab.  You will find the code in Assets/ViewModels/ForumViewModel.cs. Drag that into the ViewModels folder. It’s almost identical to the one from the previous lab, with the namespaces updated for this lab and with the “Date of latest thread” removed.  Modify the controller to use the view-model:   |  | | --- | | public IActionResult Index()  {  return View(ForumViewModel.FromForums(context.Forums));  } |   And then re-create the view, using the List template but the ForumViewModel class.  Run the application, and go to the Forum page: |
|  | We now have an app that communicating with our database. We’ll add the code for Threads and Posts later. But now it’s time to move on. |
|  |  |