Optional 17 : Deployment

Objective

Now you've built your website, you will want to deploy it.

ASP.Net Core supports a variety of deployment scenarios. We've already seen that your compiled code includes a basic built-in web server, for example, and we've also hosted our application in IIS Express for development and testing purposes.

However, IIS Express is intended only for development, and the built-in web server is very lightweight. For a production system, a fully-fledged web server allows a much better degree of management of the application. The most common deployment scenarios for ASP.Net applications are probably deploying to either IIS, or to an Azure App Service.

In this lab, you are going to deploy to IIS, and we'll take a look at some of the considerations that need to be taken into account during deployment.

This exercise will take around 30 minutes.

Referenced material

This exercise is based on material from the chapter "Deployment".

Preparing the Seed Data

- Open the 'Begin' solution in Visual Studio and compile (Shift Ctrl+B). The 'Begin' solution is identical to the 'End' solution from the previous lab.

- Up until now, we've been using a project called "SeedDatabase" to add some sample data to the database.

  This project is not suitable for real life, because a) it deletes the database before applying any changes, and b) it adds lots of data that was useful to help us develop the application, but won't be needed once we release the application.

  Instead, we want to seed the initial set of forums, but not the threads or the posts. And we will do this using Entity Framework's data seeding functionality.

- So, open /EF/ForumDbContext. Find the method OnModelCreating(). The method already contains code which describes the foreign keys in our database. At the end of this code, add the following. You can change the number and name of the forums if you want:

  ```
  protected override void OnModelCreating(ModelBuilder modelBuilder)
  {
      modelBuilder.Entity<Thread>()
          .HasOne(t => t.Forum)
          .WithMany(f => f.Threads)
          .OnDelete(DeleteBehavior.Restrict);

      modelBuilder.Entity<Post>()
          .HasOne(p => p.Thread)
  ```

```
        .WithMany(t => t.Posts)
        .OnDelete(DeleteBehavior.Restrict);

    modelBuilder.Entity<Forum>()
        .HasData
        (
            new Forum { ForumId = 1, Title = "ASP.NET MVC" },
            new Forum { ForumId = 2, Title = "ASP.NET AJAX" },
            new Forum { ForumId = 3, Title = "ASP.NET WebForms" },
            new Forum { ForumId = 4, Title = "jQuery" },
            new Forum { ForumId = 5, Title = "Silverlight" },
            new Forum { ForumId = 6, Title = "Visual Studio 2019" },
            new Forum { ForumId = 7, Title = "WPF" }
        );
}
```

Note that, when using the HasData() method, you have to provide the primary key, which is different to when you add data to the database in the normal way. HasData() is not intended to be used once the system is running – it is only to be used to seed data when deploying the application.

- Go to Tools, Nuget Package Manager, Package Manager Console. In the package manager console, enter the following command:

  ```
  add-migration SeedForums -context ForumDbContext
  ```

  Once the migration has been added, the file containing the migration is open. Have a look through the file – you'll see that the forum data is included in there, as part of the migration.

- Although we probably don't have any need for our SeedDatabase program anymore, it's worth noting that it won't work properly now, since it now seeds the same forums twice. It would be good practice to fix this, and it's a very easy fix. In SeedDatabase/Program.cs, add this line:

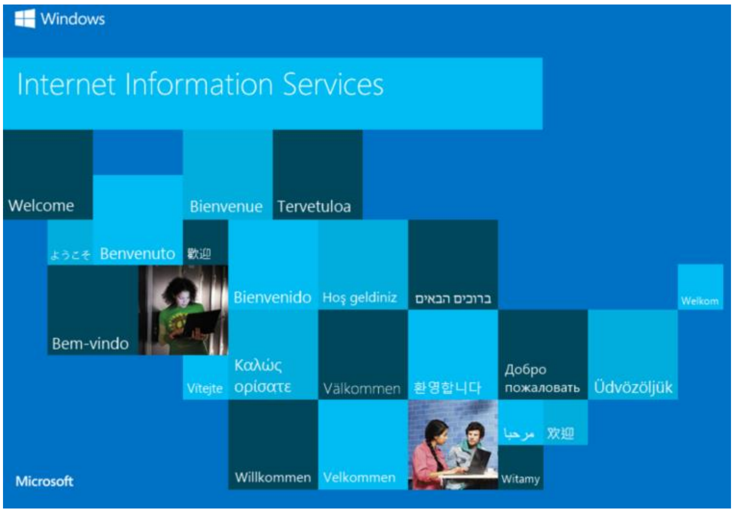
  ```
  static void Main(string[] args)
  {
      var optionsBuilder = new DbContextOptionsBuilder<ForumDbContext>();
      optionsBuilder.UseSqlServer(....);
      using(ForumDbContext ctx = new ForumDbContext(optionsBuilder.Options))
      {
          ctx.Database.EnsureDeleted();
          ctx.Database.Migrate();
          ctx.Forums.RemoveRange(ctx.Forums);
          InitForums(ctx);
      }
  }
  ```

This is the preferred way of seeding data into a live database. (It would be nice if we could seed the admin user the same way, but since the OnModelCreating() method does not have access to the user manager, we have to do that in the Main() method as shown in the previous chapter.)

Deploying to IIS

Next, we will deploy our application to IIS.

- Ensure that IIS is installed on your computer. You can do this by opening a web browser and going to http://localhost. You should see a screen like the following:



If you see an error message instead of this screen, you will need to install IIS. There are instructions for doing this at the end of this document.

Install the ASP.Net Core 3.1 Hosting Bundle. You can download the hosting bundle from the following URL:

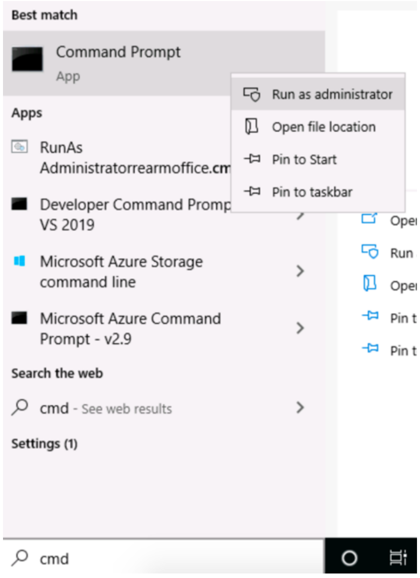
https://dotnet.microsoft.com/download/dotnet-core/thank-you/runtime-aspnetcore-3.1.3-windows-hosting-bundle-installer

The purpose of this bundle is to allow IIS to host ASP.Net Core applications.

- After installing the hosting bundle, IIS will need to be restarted. Use the following steps to restart it:
  - Click on the start menu, and type “cmd”
  - The “Command Prompt” app will be shown. Right-click on the icon for Command Prompt, and choose Run As Administrator:
  - In the command window that opens, type the following two commands:

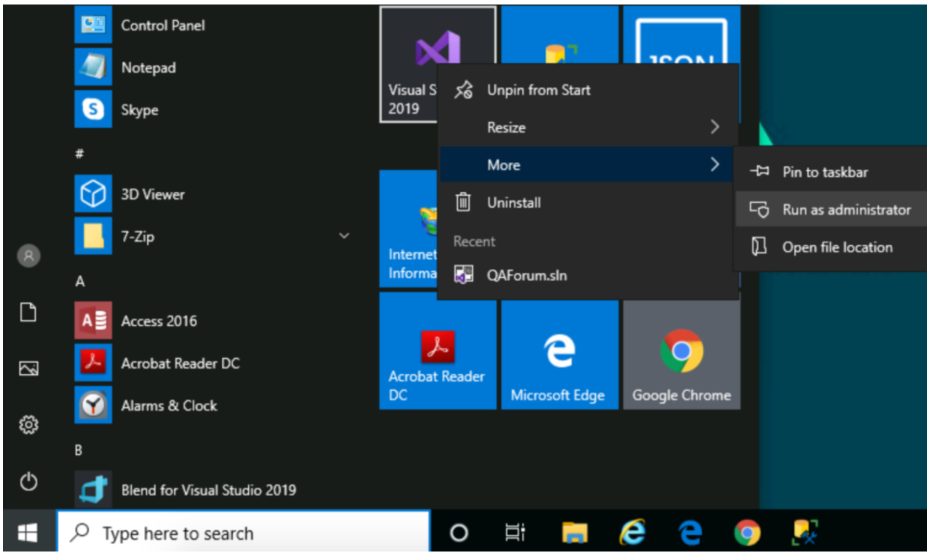
```
net stop was /y

net start w3svc
```



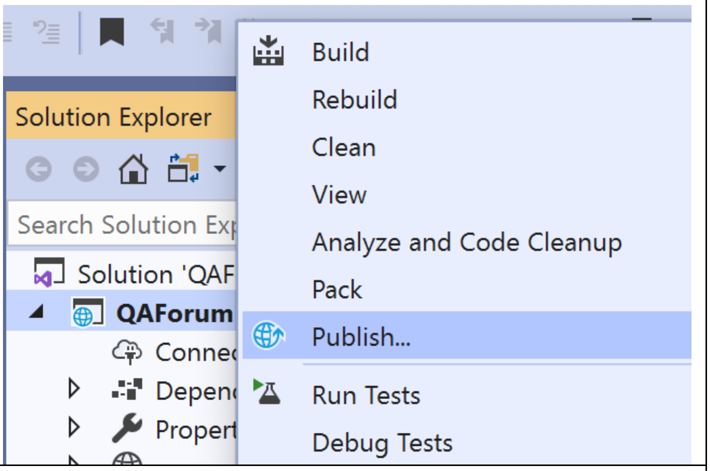
Now that we know that IIS is installed and ready for our .Net Core application, we need to prepare to deploy. This needs to be done as an administrator. So, close down Visual Studio.

Click on the Start menu, then right-click on Visual Studio 2019. Choose More, then Run As Administrator.



Once Visual Studio starts, open the QAForum project (it will probably be at the top of your Recents list, but otherwise you will need to navigate to it. Make sure you open the version from this lab, not from a previous lab!)

In the solution explorer, right-click on the QAForum project, and choose Publish:



At the next screen, you are asked to pick a publish target. Choose "IIS, FTP, etc", then click Create Profile.

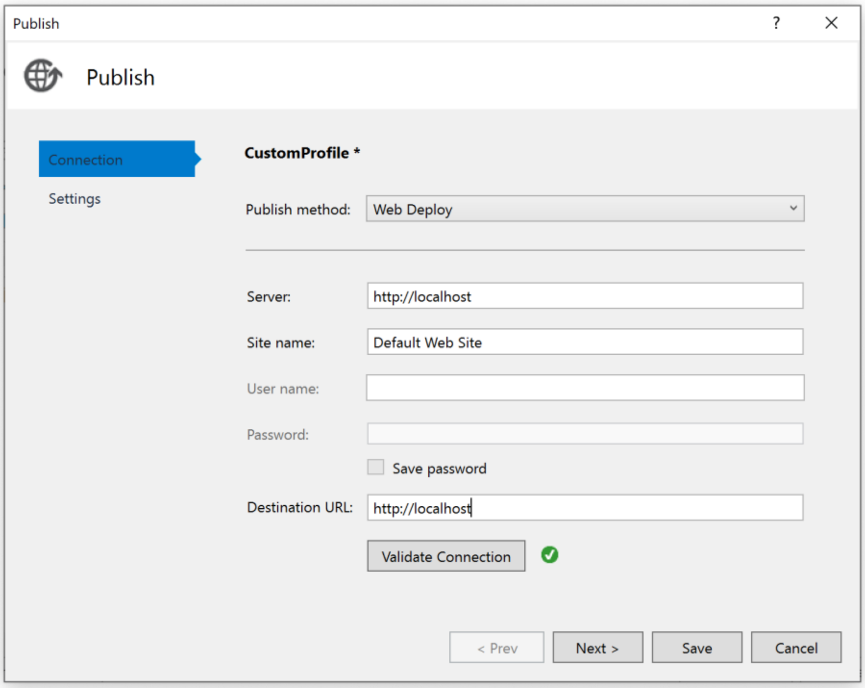
Ensure that the publish method is set to Web Deploy. Then enter the following details:

Server: http://localhost

Site name: Default Web Site

Destination URL: http://localhost

Click the Validate Connection button to make sure you've entered these details correctly:



Publish

CustomProfile *

Publish method: Web Deploy

Server: http://localhost

Site name: Default Web Site

User name:

Password:

□ Save password

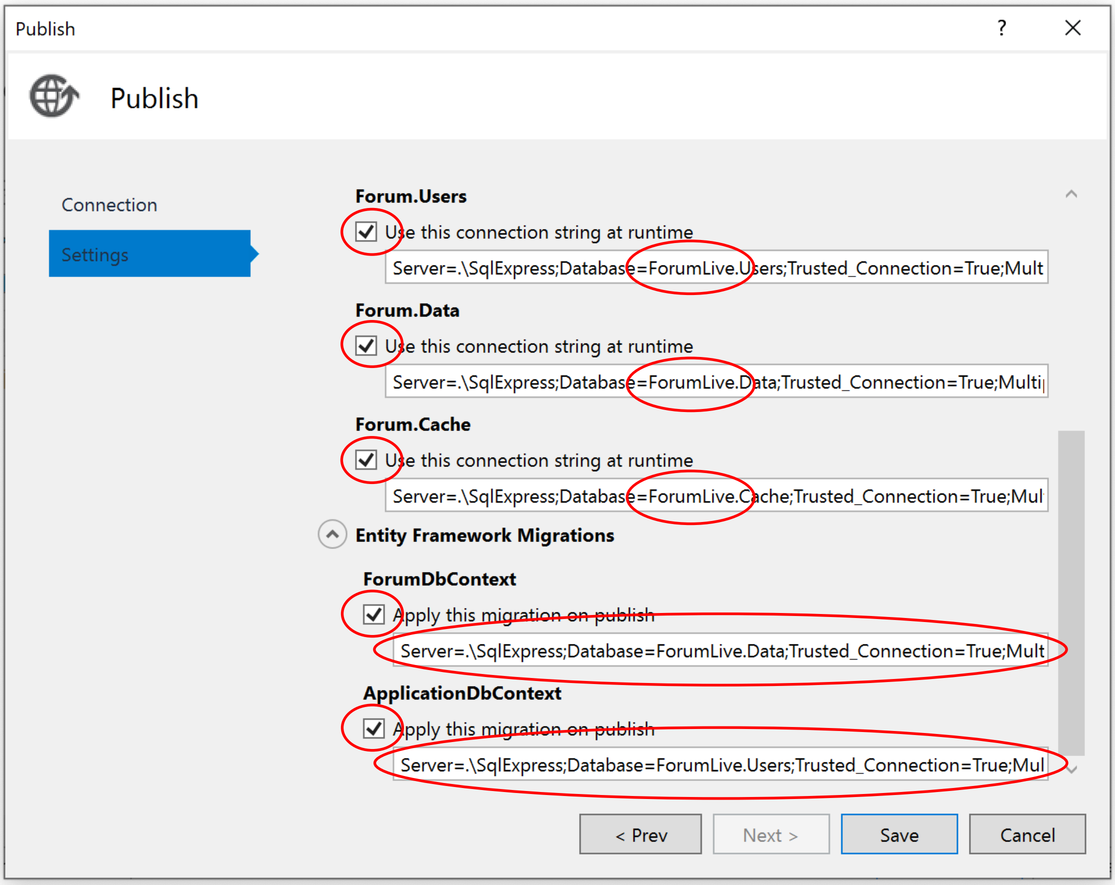
Destination URL: http://localhost

Click the Next button.

Wait for "Discovering Data Contexts" to complete if necessary. Then, expand the "Databases" option, and the "Entity Framework Migrations" option

Under the "Databases" option, you will see the three connection strings that are in our program's configuration file. One of those (the Cache) is not actually used, we set it up for a task in an earlier lab but then we removed the code that used it – but there's no harm updating it anyway.

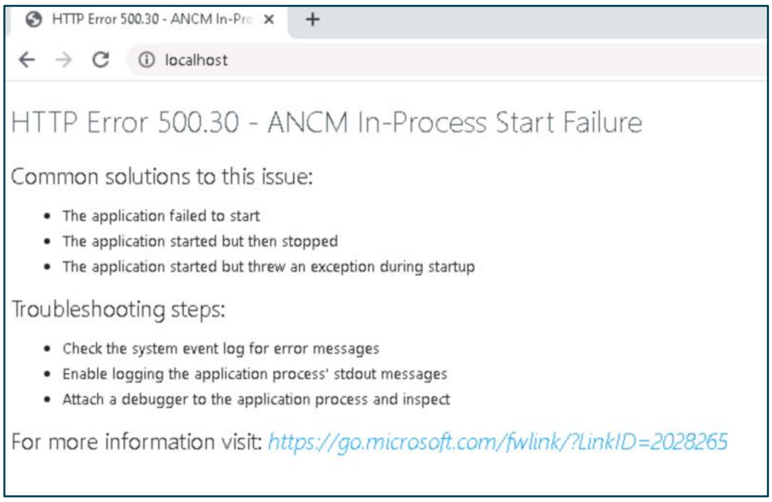
So, next to each of the three databases, tick the checkbox. This tells Visual Studio that we'd like the live version of our program to use a different connection string. Then, for each of the three databases, change the first part of the name from Forum to ForumLive, so that, for example, Forum.Users becomes ForumLive.Users.

	Now, under Entity Framework Migrations: Tick both boxes, to say that you want to apply the database migrations. (The migrations will create the databases, create the tables, and add the seed data.) Copy the connection string from “Forum.Data” in the Databases section to “ForumDbContext” in the Migrations section Copy the connection string from “Forum.Users” in the Databases section to “ApplicationDbContext” in the Migrations section
	This image shows what the screen should look like know, and indicates all the areas you should have changed:  Publish ? × Publish Connection Settings < Prev Next > Save Cancel **Forum.Users** ☑ Use this connection string at runtime Server=.\SqlExpress;Database=ForumLive.Users;Trusted_Connection=True;Mult **Forum.Data** ☑ Use this connection string at runtime Server=.\SqlExpress;Database=ForumLive.Data;Trusted_Connection=True;Multi **Forum.Cache** ☑ Use this connection string at runtime Server=.\SqlExpress;Database=ForumLive.Cache;Trusted_Connection=True;Mul **Entity Framework Migrations** **ForumDbContext** ☑ Apply this migration on publish Server=.\SqlExpress;Database=ForumLive.Data;Trusted_Connection=True;Mult **ApplicationDbContext** ☑ Apply this migration on publish Server=.\SqlExpress;Database=ForumLive.Users;Trusted_Connection=True;Mul
	Click Save to save this publish profile. You can now re-use the profile every time you need to publish your application.

On the next screen, click Publish.

Your program will be compiled, packaged up for publishing, and then sent to the web server. Then, the database migrations will be applied.

Once all of this is complete, a web browser will open on your website's home page… but you will see the following error:



HTTP Error 500.30 - ANCM In-Pro ×

localhost

HTTP Error 500.30 - ANCM In-Process Start Failure

Common solutions to this issue:

- The application failed to start
- The application started but then stopped
- The application started but threw an exception during startup

Troubleshooting steps:

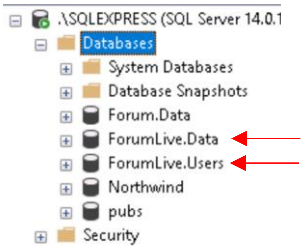
- Check the system event log for error messages
- Enable logging the application process' stdout messages
- Attach a debugger to the application process and inspect

For more information visit: https://go.microsoft.com/fwlink/?LinkID=2028265

The reason for the error is that the web server runs under a different user profile to the one we've been working with throughout the course, and that user profile does not have permission to access the databases. (The error actually happens in the Main() method, when trying to add the admin user – so before the application is running. That's why we're seeing an error message from IIS, rather than a message from the application.)

To fix this, open SQL Server Management Studio.

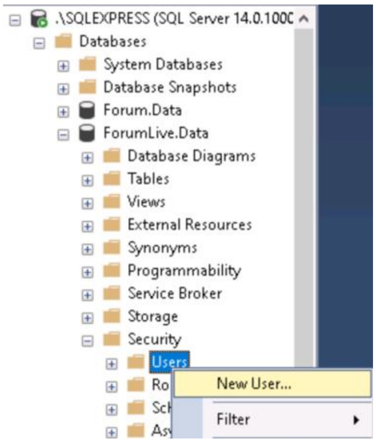
Check in SQL Server Management Studio that two new "live" databases have been created:



.\SQLEXPRESS (SQL Server 14.0.1
- Databases
  - System Databases
  - Database Snapshots
  - Forum.Data
  - ForumLive.Data
  - ForumLive.Users
  - Northwind
  - pubs
- Security

Expand ForumLive.Data, then expand Security.

Right-click on Users, and select New User.



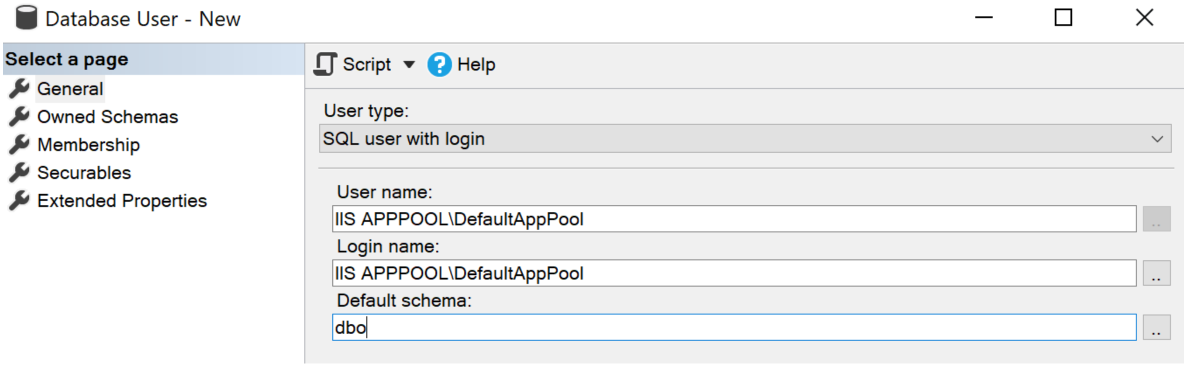
In the New User dialog, select a user type of "Windows User". Then, enter the following for both the user name and the login name. It must be entered accurately, because this is the name of the account which IIS is using:

IIS APPPOOL\DefaultAppPool

For the default schema, enter:

dbo

The screen should look like this:



User type: SQL user with login

User name: IIS APPPOOL\DefaultAppPool

Login name: IIS APPPOOL\DefaultAppPool

Default schema: dbo

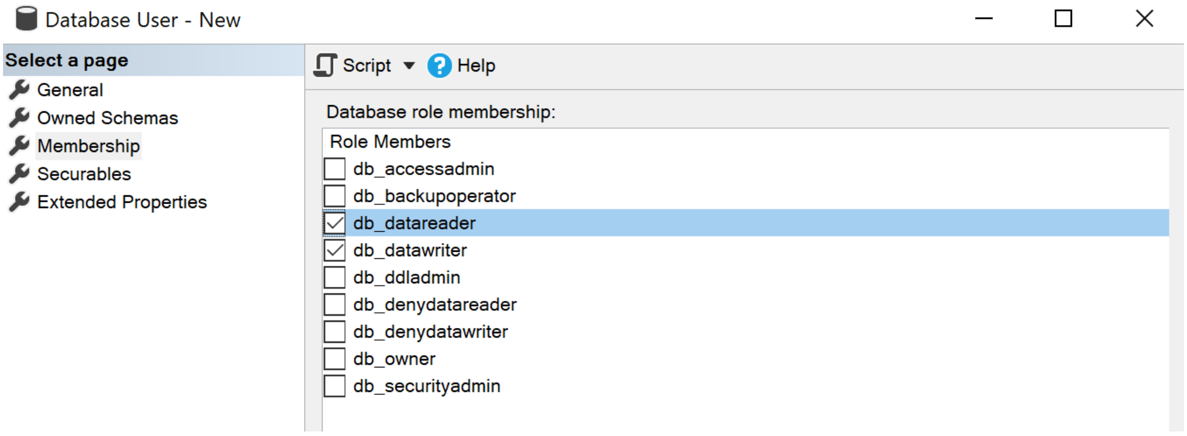
Click on the Membership section of the new user screen.

Put ticks in the boxes labelled:

db_datareader

db_datawriter

The screen should look like this:



Database role membership:

Role Members

☐ db_accessadmin
☐ db_backupoperator
☑ db_datareader
☑ db_datawriter
☐ db_ddladmin
☐ db_denydatareader
☐ db_denydatawriter
☐ db_owner
☐ db_securityadmin

Click the Ok button to save the user details.

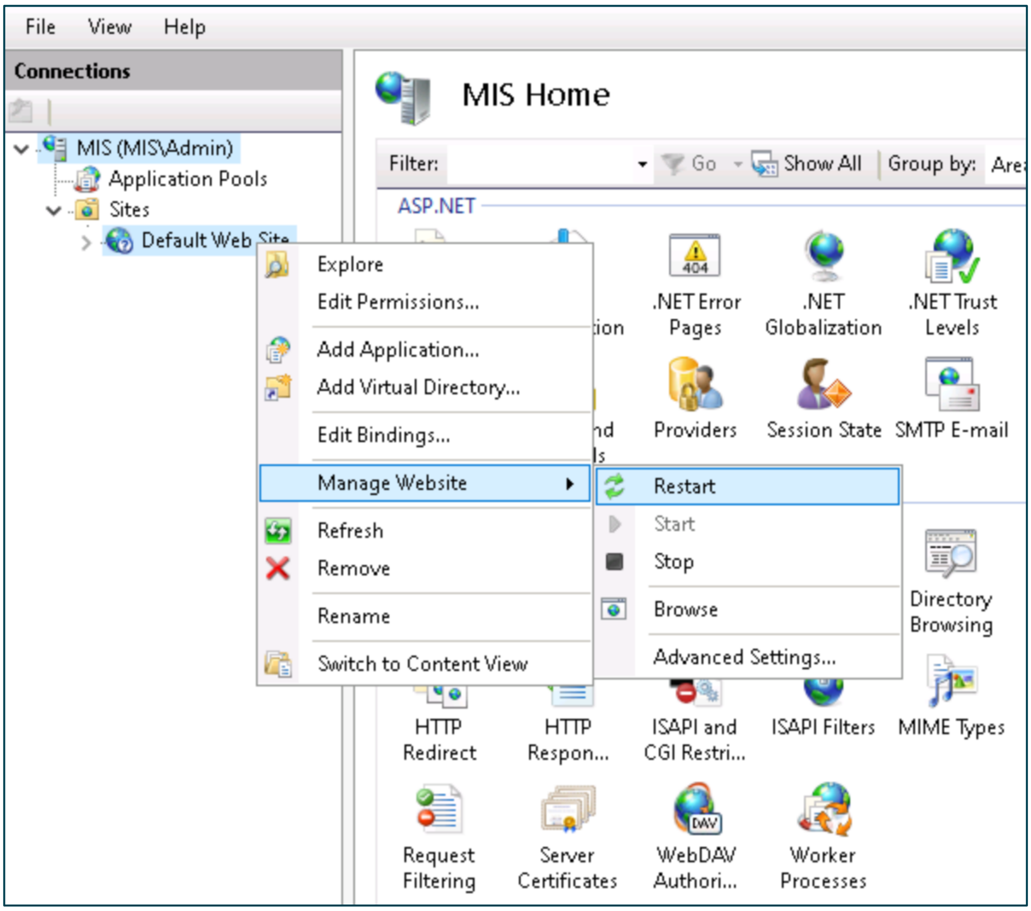
Then, repeat the previous two steps to create an identical user entry with the same permissions in the ForumLive.Users database.

You will only need to complete the last few steps the first time you deploy, when a new database is created. Once the database is created and the permissions have been set, they will remain set when you deploy again, even if your application includes new migrations that need to be applied.

IIS now needs to restart the application, otherwise it remains in its error state. Again, this will only need to be done the first time you deploy the application.

Click on the Start menu, and type “IIS”. Of the options presented, you should be able to see “Internet Information Services (IIS) Manager”, which you should now click on.

In the left-hand side of the application, expand the computer’s name. Then expand the “Sites” section. Right click on Default Web Site, and choose Manage Website, Restart:



	Now, refresh your web browser. (If you closed the web browser, open it again and visit http://localhost.com) Everything should be working! You can log in as admin@qaforum.com (the password is Pa$$w0rd), or you can create a new login. If you create a new login, you will need to go to C:\Users\Public to find the "confirmation e-mail" with the link for you to click to confirm your e-mail address. Look at the Forums list – you should see the forums that you seeded using Entity Framework. But there will be no threads or posts until you add some.

Congratulations, you now know how to deploy your software! Let's take a brief look at what happened to our settings file when we deployed the application.

Understanding Settings Transformations

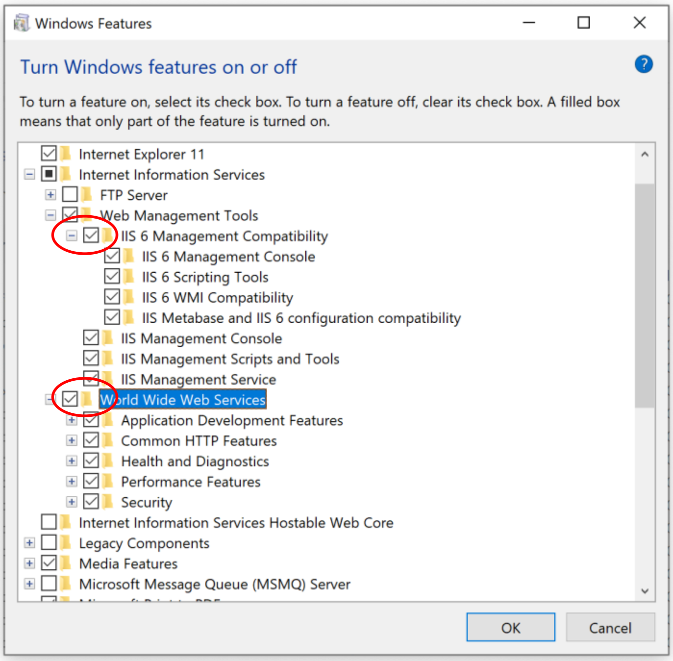
•	One last thing to do. Have a look in C:\Inetpub\wwwroot. This is the folder where your website files are located. Find the files that start with "appsettings…" There is a file called appsettings.production.json, which was not in our original project. Use whatever text editor you like to take a look in this file. Any settings in this file will override the standard settings, so long as the ASPNETCORE_ENVIRONMENT variable is set to "production". In here, you will find the new connection strings. This file was created as part of the deployment of your project.
•	While we're here, have a look at one more file: web.config. This file was not in our original project at all, but has been added as part of the deployment. It contains IIS configuration options. Many options can be set graphically through the IIS Manager, but further options can be set in this file if required.

That's the end of the course – well done!

Installing IIS

You should find that IIS is installed on your lab computer. But in case it isn't, or if you want to know how to install IIS on your own computer, we've provided instructions for installing it below:

	To install IIS, click the Start menu, and type "Control Panel". Open the control panel. Click "Programs and Features"

	Click “Turn Windows Features On or Off” From the options that are shown, click the + sign next to “Internet Information Services
	There will be three sections available, each with their own sub-sections. Go into the sub-sections for Web Management Tools and for World Wide Web Services, and turn on everything in both sub-sections. By the time you finish, your screen should look like this:  Windows Features Turn Windows features on or off To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on. ☑ Internet Explorer 11 ⊟ ■ Internet Information Services ⊞ □ FTP Server ⊟ ☑ Web Management Tools ⊟ ☑ IIS 6 Management Compatibility ☑ IIS 6 Management Console ☑ IIS 6 Scripting Tools ☑ IIS 6 WMI Compatibility ☑ IIS Metabase and IIS 6 configuration compatibility ☑ IIS Management Console ☑ IIS Management Scripts and Tools ☑ IIS Management Service ⊟ ☑ World Wide Web Services ⊞ ☑ Application Development Features ⊞ ☑ Common HTTP Features ⊞ ☑ Health and Diagnostics ⊞ ☑ Performance Features ⊞ ☑ Security □ Internet Information Services Hostable Web Core ⊞ □ Legacy Components ⊞ ☑ Media Features ⊞ □ Microsoft Message Queue (MSMQ) Server
	Click Ok, and wait for everything to install. You may need to reboot your computer.