SENSIBLE DEV

BOOTSTRAP FOR .NET DEVS

FRONT-END FRAMEWORK MASTERY

Matthew Sinex

Bootstrap for .NET Devs

Matthew Sinex

This book is for sale at http://leanpub.com/bootstrap4dotnet

This version was published on 2019-11-06

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

Introduction

Personally, I don't consider myself to be much of a designer, which is why I turn to front-end frameworks when developing my projects. Bootstrap is, of course, one of the most popular front-end frameworks, and integrates well with the .NET ecosystem.

However, Bootstrap and .NET work together in ways that aren't always obvious.

Also, new .NET projects ship with Bootstrap 3 instead of the recently updated Bootstrap 4. The built-in form scaffolds are also formatted with Bootstrap 3 styles instead of Bootstrap 4. This presents us with some issues, especially if you're trying to follow along with up-to-date tutorials on the framework.

But thankfully, we don't have to throw out Bootstrap 4, nor do we have to abandon the awesome .NET features, like building forms from model classes and built-in form validation. I'll guide you through how to use Bootstrap 4, and give you easy, copy-and-paste templates that you can use in your projects right away.

What You'll Learn

In **Chapter 1**, we'll set up our application and upgrade from Bootstrap 3 to Bootstrap 4.

Chapter 2 will cover Bootstrap forms and how they work with .NET form-building utilities. These differ from MVC 5 to .NET Core, so we'll look at those differences.

In **Chapter 3**, we'll discuss navigation menus and how to set the active menu item programmatically.

What You Won't Learn

I'm not going to walk through every feature of Bootstrap 4. I don't think it's really necessary to explain what's already explained wonderfully in the official documentation.

This applies to most components that are framework or platform agnostic. For example, to make a button larger, you add the `btn-lg` class:

```
<button class="btn btn-lg">Large button</button>
```

But this has nothing to do with .NET. It'd be the same if you were coding with Ruby on Rails, Symfony, or just plain old HTML and CSS in a text editor.

This book will only focus on the components that make a difference with ASP.NET features.

MVC 5 and .NET Core

Throughout this book, you'll see examples in both Visual Studio 2015 with MVC 5, and .NET Core in Visual Studio 2017.

Why both? While you probably won't be using the older version of Visual Studio for new projects, there are plenty of legacy applications around that were built in VS 2015. If your organization still uses it, you'll find these sections helpful.

There are many places where there isn't any difference between the two. However, there are enough differences, especially between Url Helpers and Razor Tags, where there's a need for separate examples.

Source Code

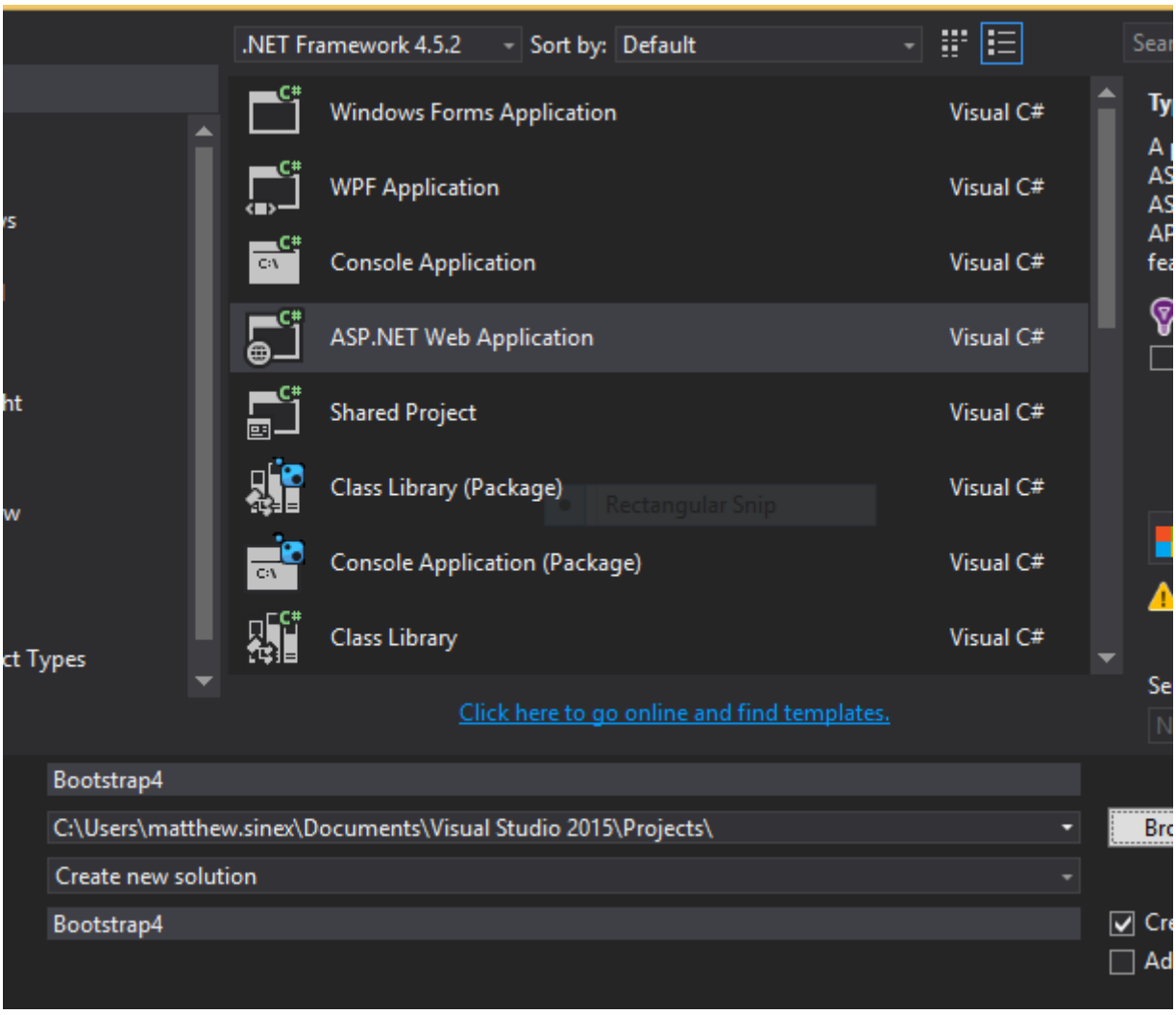
This book comes with the complete source code to every chapter. You can find the download link for the Visual Studio projects in Appendix A, at the end of the book.

Setting up Your Application

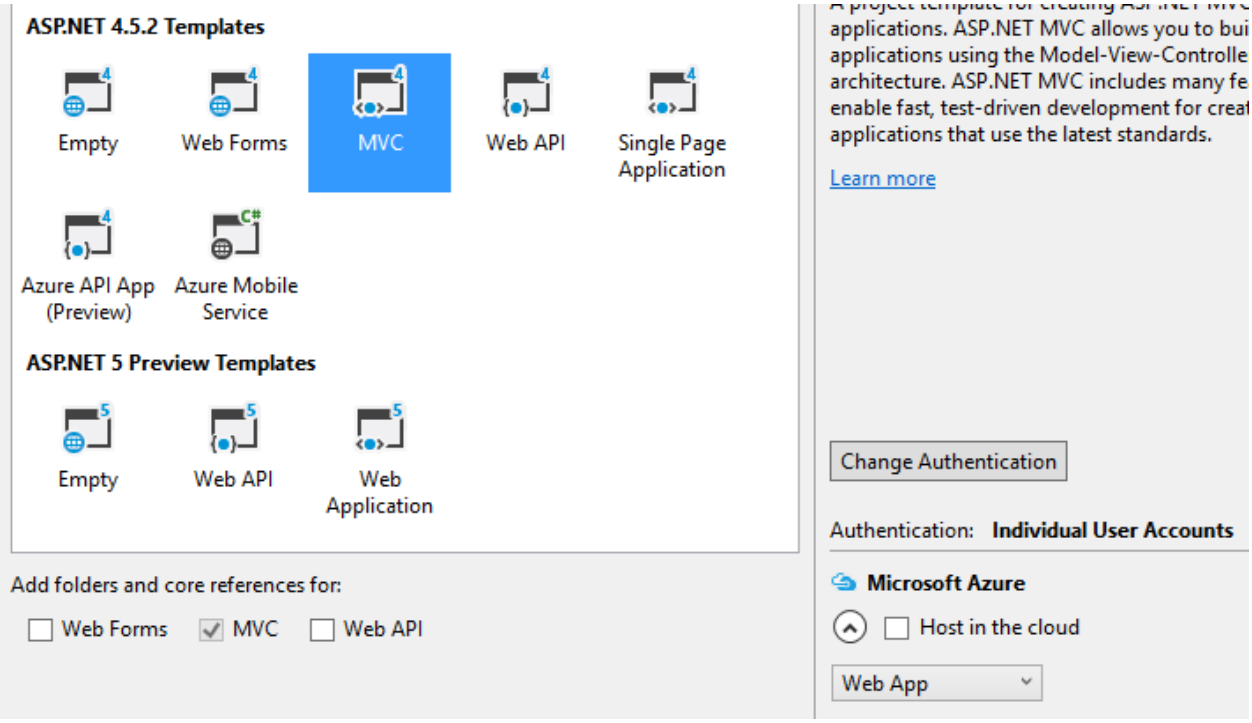
For our first steps, let's set up a basic application with Individual Account authentication.

MVC 5

Create a new web application and call it `Bootstrap4`.

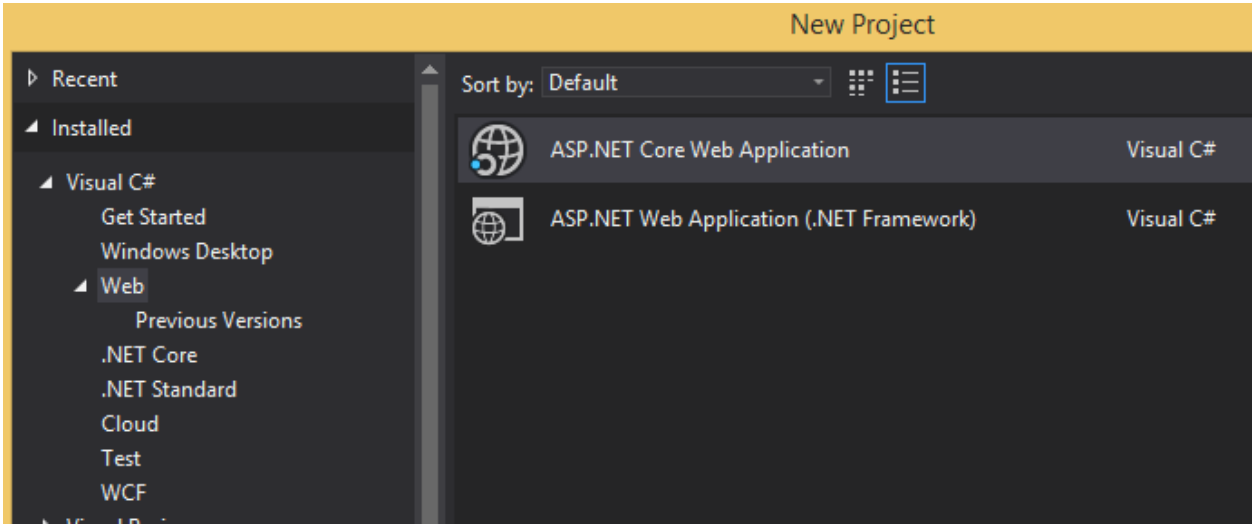


Select the **MVC** template, and **Individual User Accounts**.

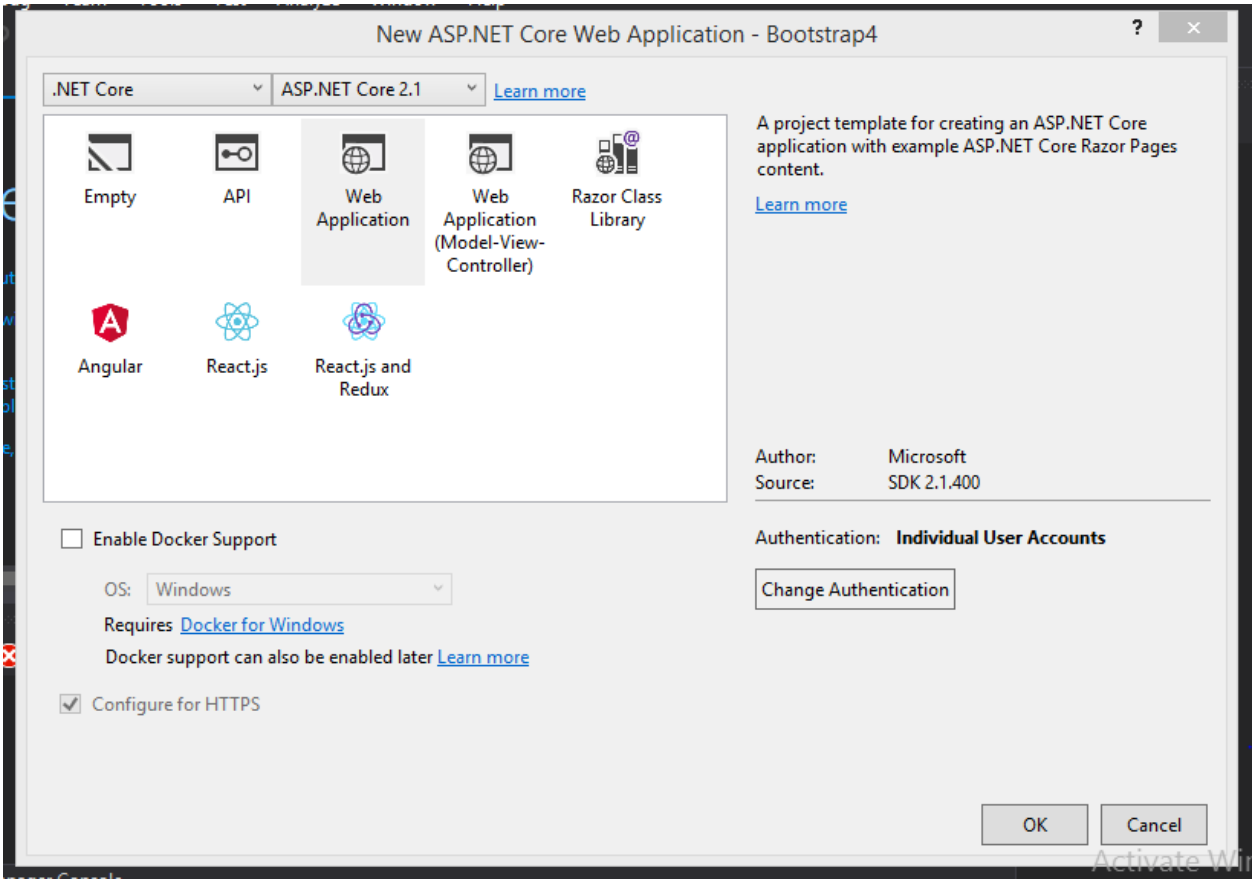


.NET Core

Create a new ASP.NET Core web application and call it `Bootstrap 4`.



On the next screen, choose **Web Application** and **Individual User Accounts**.



You can, if you want, choose a different authentication type, but I've chosen what I think is the most common option for most people.

Upgrading Bootstrap 3 to Bootstrap 4

Now that we have the project set up, let's upgrade our version of Bootstrap. By default, the project is set up with Bootstrap 3. We'll need to change the CSS and JavaScript references from Bootstrap 3 to Bootstrap 4.

Upgrading in MVC 5

One easy option is to install Bootstrap 4 from the Package Manager Console. Simply type this into the console and press `Enter`:

```
Install-Package bootstrap
```

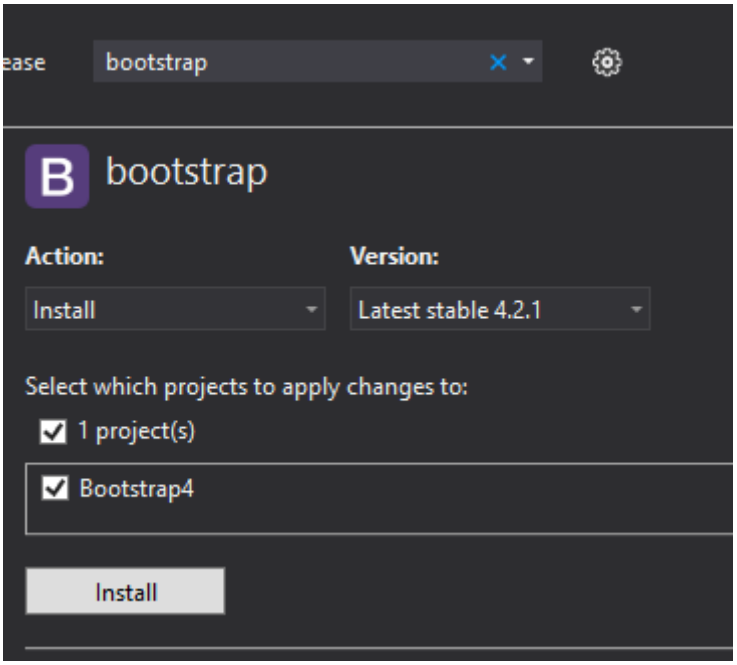
This will target the most recent version of Bootstrap. As of today, that's still major version 4, and that's all the specificity we really care about.

But if you wanted to target a specific version, you can, by adding the `version` flag:

```
Install-Package bootstrap -Version 4.1.1
```

Alternatively, you could click on **Tools** → **Nuget Package Manager** → **Manage Nuget Packages for Solution**...

Then, click on Browse and search for `bootstrap`.

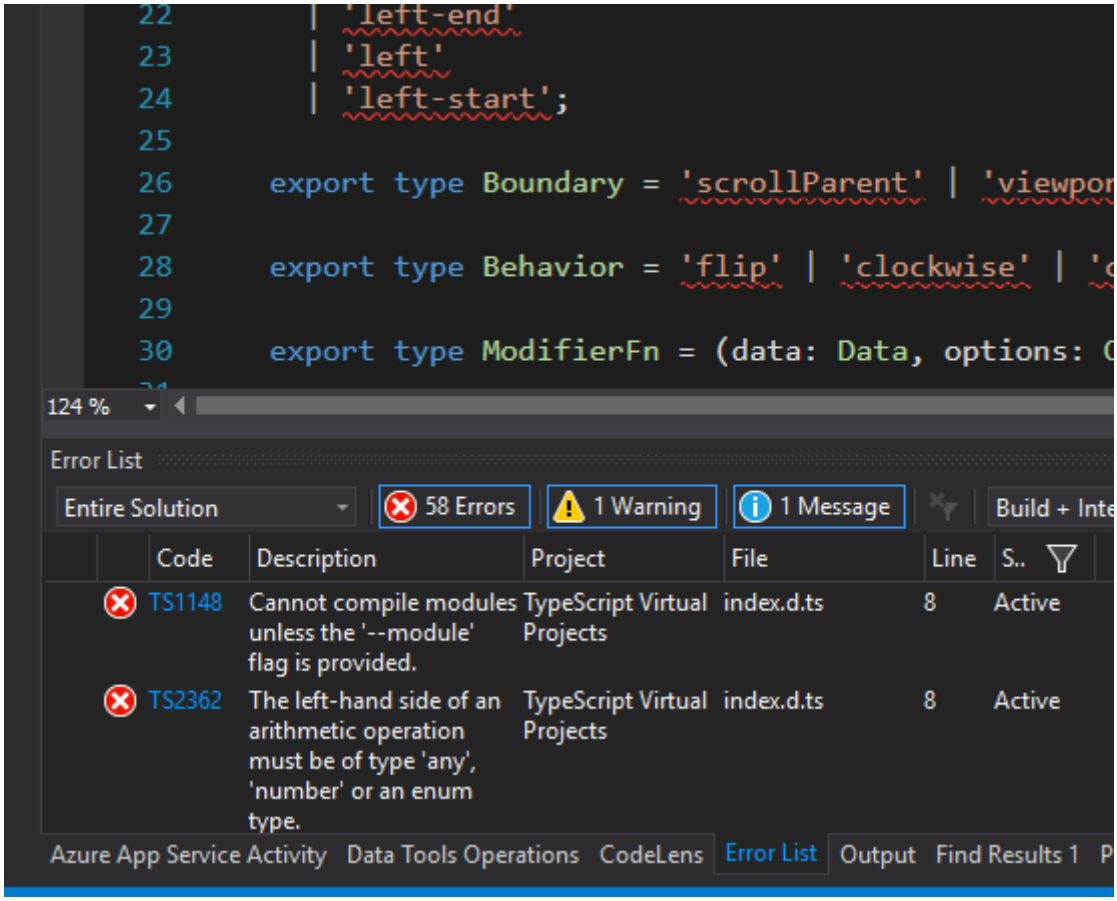


Click on **Install**.

Whichever way you do it, installing the nuget package will overwrite the **bootstrap.js** file in the **Scripts** folder and the **bootstrap.css** file in the **Content** folder. It will also add a number of other files to those two folders, most of which you don't need to worry about.

Fixing the Popper.js Error

Here's one snag you might run into. When you try to run the project locally, Intellisense might detect errors in the TypeScript file `index.d.ts`, found in the **Scripts** folder.



This is a TypeScript file for **Popper.js**, a library for managing tooltips and popovers. Compiling it will generate the appropriate JavaScript code.

Basically, the project folder is missing a TypeScript configuration file, which is where you would set the module flag.

You could mess around with that, but if you aren't going to change anything in Popper's source code, you don't need to.

The easiest way to fix this problem is to just exclude the file from your project.

Right-click on `index.d.ts` and select **Exclude From Project**.

Upgrading in .NET Core

Upgrading from Bootstrap 3 to 4 in .NET Core is a bit more complicated.

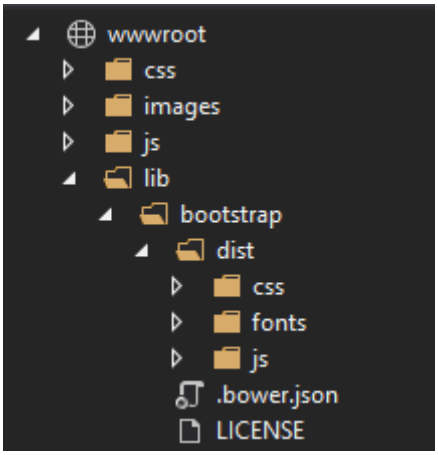
This is mostly because JavaScript dependencies are being handled more and more by client-side build systems, like Gulp or Webpack. Since developers are using these systems to manage dependencies anyway, .NET is starting to get out of the way.

In any case, here are two ways you can upgrade from Bootstrap 3 to 4.

Installing Manually from the Official Site

Head out to the official Bootstrap 4 site[1] and grab the compiled files from the download page.

Now we just need to overwrite the existing Bootstrap files. You can find these (and other JavaScript library files) in **wwwroot/lib/bootstrap/dist**.



In the zip archive you downloaded, just copy **bootstrap.css** to the **css** folder, and **bootstrap.js** to the **js** folder, overwriting the Bootstrap 3 files that already exist in your project.

That will take care of your local files.

Next, go to the layout file in **Pages/Shared/_Layout.cshtml**.

You'll notice that .NET Core has introduced Environment Tag Helpers in the layout. For example, you should see this block of code in the `head` of the page:

```
<environment exclude="Development">
  <!--- stylesheet links --->
</environment>
```

This indicates that if the environment is not development (i.e., production), the page will render those references.

In the head, you'll want to change the link to the latest version of **bootstrap.min.css**, and the link near the bottom of the page to the latest version of **bootstrap.min.js**.

So, for example, you'd want to change this CSS link:

```
href="https://ajax.aspnetcdn.com/ajax/bootstrap/3.3.7/css/bootstrap.min.css"
```

to this:

[1] https://getbootstrap.com/

```
href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css"
```

Be sure to also change the integrity hash if you're including that.

As of today, you can find the updated CDN links on this page[2]. Of course, keep in mind that versions will change often.

And that's the problem with this method. If you're happy with installing once and never updating Bootstrap 4 again, this will be fine. And, to be perfectly honest, I'd probably be happy doing this myself.

However, if you want a more robust system of updating your Bootstrap 4 files, you can use a task manager like Gulp to automatically pull the latest version and overwrite the files where we need to.

Using npm and Gulp

Now we're really getting into the weeds here, but I'm going to give you a quick crash course on how to use Node Package Manager and Gulp to install Bootstrap and put the files where they need to go.

Make sure you have npm installed. If you don't, head to the installation page[3].

Go back to your Visual Studio project. Right-click on the project, then select **Add** → **New Item**...

Search for `npm Configuration File` and click on `Add` to create a `package.json` file.

Go to the Package Manager Console and run the following two commands:

```
cd bootstrap4
npm install -S bootstrap
```

The first command will get you into the project subfolder. The second will install Bootstrap into the **node_modules** folder. You may need to click on the `Show All Files` button in the Solution Explorer to see it.

Whatever you do, though, **DO NOT** add the **node-modules** folder to your project. If you do, you'll end up adding thousands of unnecessary files as your project grows.

Next, we need to install Gulp and set it up to copy over the correct files from **node_modules** to **wwwroot**.

Install Gulp by running this in the Package Manager Console (make sure you're still in the project directory, not the solution directory):

```
npm install -S gulp
```

Now create a file in the root of the project and name it `gulpfile.js`.

In this file, we'll create a build pipeline to copy the files. Gulp is very powerful, and you can do a lot with it, but here we're keeping it simple.

gulpfile.js

[2] https://getbootstrap.com/docs/4.2/getting-started/download/
[3] https://www.npmjs.com/get-npm

```
const gulp = require('gulp');

const bootstrapStyles = [
  "node_modules/bootstrap/dist/css/bootstrap.min.css",
  "node_modules/bootstrap/dist/css/bootstrap.css"
];

const bootstrapScripts = [
  "node_modules/bootstrap/dist/js/bootstrap.min.js",
  "node_modules/bootstrap/dist/js/bootstrap.js"
];

gulp.task("build-bootstrap-css", function () {
  return gulp.src(bootstrapStyles)
    .pipe(
      gulp.dest('wwwroot/lib/bootstrap/dist/css')
    );
});

gulp.task("build-bootstrap-js", function () {
  return gulp.src(bootstrapScripts)
    .pipe(
      gulp.dest('wwwroot/lib/bootstrap/dist/js')
    );
});

gulp.task('default', gulp.series(
  'build-bootstrap-css', 'build-bootstrap-js'
));
```

So here's what's happening. We're defining the paths for the files we want to grab from **node_modules**. We then define two Gulp tasks, one to copy the CSS and one to copy the JavaScript files. Within each task, we define the destination path. Finally, we make a `default` task which will run our two tasks in a series.

Now let's make it so that this script runs whenever we build our Visual Studio project.

Edit your `csproj` file (in this case, `Bootstrap4.csproj`). Add the following lines inside the `Project` tags:

```
<Target Name="RunGulp" BeforeTargets="Build">
  <Exec Command="node_modules\.bin\gulp.cmd" />
</Target>
```

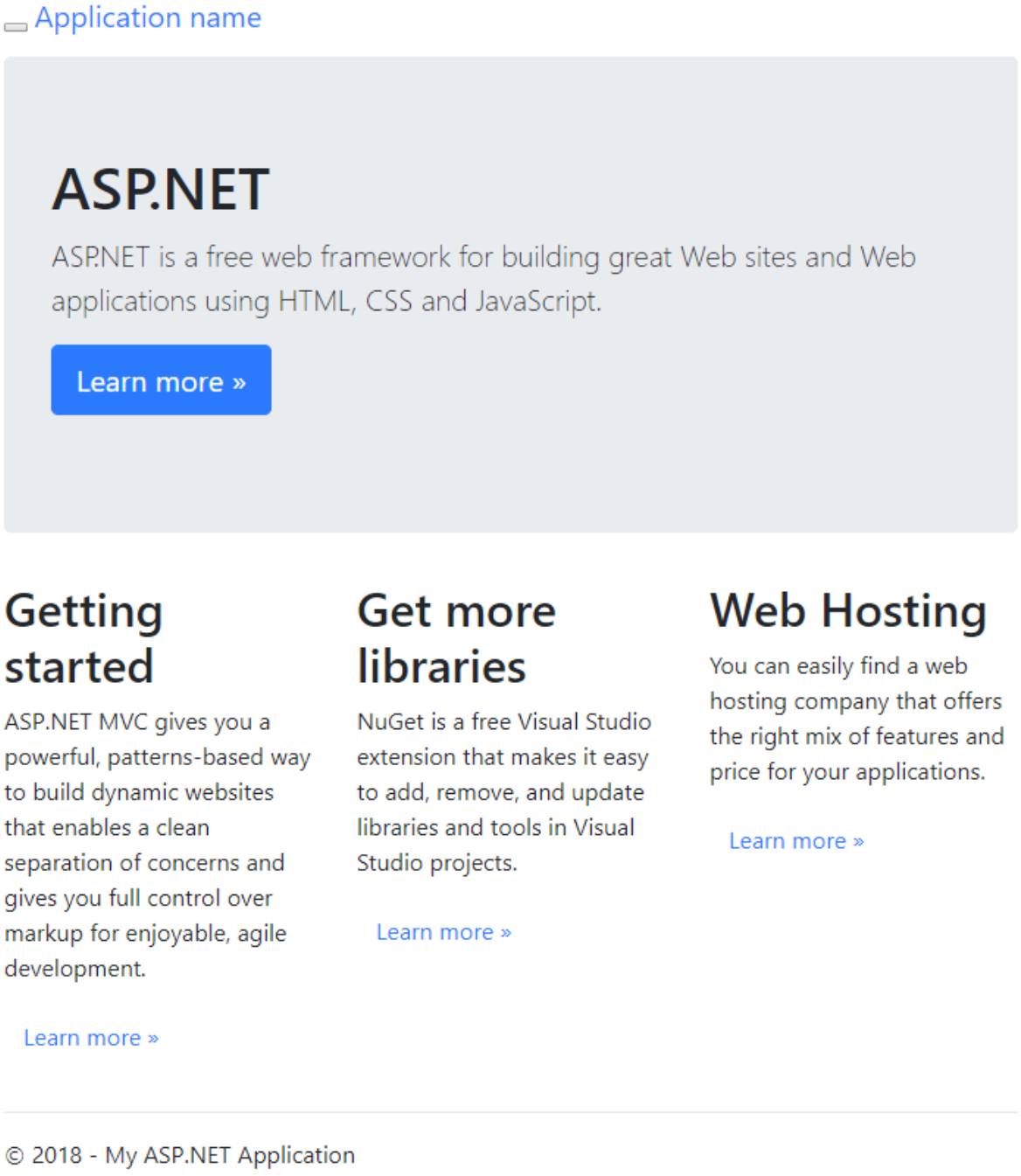
Every time you build the project, Gulp will copy over whatever Bootstrap files you've installed from npm.

Changing the Application Layout

Now that you've installed Bootstrap 4, let's debug the application locally and see what we have.

Hit `F5` or click on the green play button (with the name of your default browser) in the toolbar.

You should see something like this:



All right, we have a home page!

But, as you can probably notice, that navigation bar doesn't look right. In fact, it's downright broken. That's because Bootstrap 4 made some changes to how the navbar works, so we'll need to change our layout.

Navigate to the layout file in **Views** → **Shared** → **_Layout.cshtml**. We need to remove the div with

the class of `navbar` (the first element in the HTML body). Replace it with this:

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
</nav>
```

This will provide us with an empty navbar in a dark style, like the original Bootstrap 3 template. The `navbar-expand-lg` class adds responsive properties. By default, the navbar has a menu that expands vertically. With this class, you'll see a "normal" horizontal menu when the screen is wider than the large breakpoint (992px by default).

Next, we need to add a div with a class of `container` inside the nav. This will ensure some standardized widths and padding so that our navigation won't just stretch to the very ends of the screen.

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <div class="container">
  </div>
</nav>
```

Great! Now you can add in anything you want to the menu itself. In our case, to match the default ASP.NET layout, we're going to add the following four things:

1. A brand link (or application link)
2. A button to open and close the hamburger menu
3. A nav with our menu links
4. The LoginPartial that will render the login / logout links

So, let's get going. Keep in mind that each of these will be placed inside of the container div.

First, add the following markup for the brand link:

MVC 5

```
@Html.ActionLink(
  "Application name",
  "Index",
  "Home",
  new { area = "" },
  new { @class = "navbar-brand" }
)
```

.NET Core

```
 <a asp-page="/Index" class="navbar-brand">VetVisit</a>
```

The following button will expand the responsive menu when clicked:

```
<button type="button"
    class="navbar-toggler"
    data-toggle="collapse"
    data-target=".navbar-collapse"
    aria-expanded="false"
    aria-label="Toggle navigation"
>
  <span class="navbar-toggler-icon"></span>
</button>
```

Next, we can add our menu items into a div. We'll use the ActionLink Url Helpers (Tag Helpers in .NET Core) to render the links.

Also, you'll notice that we've added the `LoginPartial` as the last element in the div.

MVC 5

```
<div class="navbar-collapse collapse">
  <ul class="nav navbar-nav mr-auto">
    <li class="nav-item">
      @Html.ActionLink(
        "Home", "Index", "Home", null, new { @class = "nav-link" }
      )
    </li>
    <li class="nav-item">
      @Html.ActionLink(
        "About", "About", "Home", null, new { @class = "nav-link" }
      )
    </li>
    <li class="nav-item">
      @Html.ActionLink(
        "Contact", "Contact", "Home", null, new { @class = "nav-link" }
      )
    </li>
  </ul>
  @Html.Partial("_LoginPartial")
</div>
```

.NET Core

```
<div class="navbar-collapse collapse">
  <ul class="nav navbar-nav mr-auto">
    <li class="nav-item"><a asp-page="/Index" class="nav-link">Home</a></li>
    <li class="nav-item"><a asp-page="/About" class="nav-link">About</a></li>
    <li class="nav-item"><a asp-page="/Contact" class="nav-link">Contact</a></li>
  </ul>
  @await Html.PartialAsync("_LoginPartial")
</div>
```

Finally, we need to make a few changes to the LoginPartial. Obviously, if you chose a different authentication method aren't using it, you don't need to make this adjustment.

_LoginPartial.cshtml

MVC 5

```
@using Microsoft.AspNet.Identity
@if (Request.IsAuthenticated)
{
  using (Html.BeginForm(
    "LogOff", "Account", FormMethod.Post, new { id = "logoutForm" }
  ))
  {
  @Html.AntiForgeryToken()
  <ul class="nav navbar-nav">
    <li class="nav-item">
        @Html.ActionLink(
          "Hello " + User.Identity.GetUserName() + "!",
          "Index",
          "Manage",
          routeValues: null,
          htmlAttributes: new { title = "Manage", @class = "nav-link" }
        )
    </li>
    <li class="nav-item">
      <a
        class="nav-link"
        href="javascript:document.getElementById('logoutForm').submit()"
      >
        Log off
      </a>
    </li>
  </ul>
  }
```

```
}
else
{
  <ul class="nav navbar-nav">
    <li class="nav-item">
      @Html.ActionLink(
        "Register",
        "Register",
        "Account",
        routeValues: null,
        htmlAttributes: new { id = "registerLink", @class = "nav-link" }
      )
    </li>
    <li class="nav-item">
      @Html.ActionLink(
        "Log in",
        "Login",
        "Account",
        routeValues: null,
        htmlAttributes: new { id = "loginLink", @class = "nav-link" }
      )
    </li>
  </ul>
}
```

.NET Core

```
@if (SignInManager.IsSignedIn(User))
{
  <form
    asp-controller="Account"
    asp-action="Logout"
    method="post"
    id="logoutForm"
    class="navbar-right"
  >
    <ul class="nav navbar-nav">
      <li class="nav-item">
        <a asp-page="/Account/Manage/Index" title="Manage" class="nav-link">
          Hello @UserManager.GetUserName(User)!
        </a>
      </li>
      <li class="nav-item">
```

```
        <button type="submit" class="btn btn-link navbar-btn navbar-link">
          Log out
        </button>
      </li>
    </ul>
  </form>
}
else
{
  <ul class="nav navbar-nav">
    <li class="nav-item">
      <a asp-page="/Account/Register" class="nav-link">Register</a>
    </li>
    <li class="nav-item">
      <a asp-page="/Account/Login" class="nav-link">Log in</a>
    </li>
  </ul>
}
```

Adding the Style Render Section

In the future, we'll be adding stylesheets specific to a page. Because of this, we'll want to add a `RenderSection` to our layout.

Add the following style section to the head:

```
@RenderSection("styles", required: false)
```

.NET Core Environments in Site Layout

.NET Core makes use of different environments, like "Development" and "Production." The layout file includes several tags to specify which script and style files to use in each environment. As is, the local files are used in development, with the CDNs used in production (with the local files listed as a fallback). Our local copies of Bootstrap have been updated to version 4 already. We'll just want to update the URL of our CDNs. The exact link will differ depending on which version is current when you're creating the project. As of this writing, the current version is 4.1.1. You can find the URLs in a number of different places, but you should be able to find them easily enough on the Bootstrap home page in the "Getting Started" section.

In our example, you'll want to change the stylesheet references in the head to this:

```
<environment include="Development">
  <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />
  <link rel="stylesheet" href="~/css/site.css" />
</environment>
<environment exclude="Development">
  <link rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css"
    asp-fallback-href="~/lib/bootstrap/dist/css/bootstrap.min.css"
    asp-fallback-test-class="sr-only"
    asp-fallback-test-property="position"
    asp-fallback-test-value="absolute"
    integrity=
      "sha384-WskhaSGFgHYWDcbwN70/dfYBj47jz9qbsMId/iRN3ewGhXQFZCSftd1LZCfmhktB"
    crossorigin="anonymous"/>
  <link rel="stylesheet" href="~/css/site.min.css" asp-append-version="true" />
</environment>
```

And the script references to this:

```
<environment include="Development">
  <script src="~/lib/jquery/dist/jquery.js"></script>
  <script src="~/lib/bootstrap/dist/js/bootstrap.js"></script>
  <script src="~/js/site.js" asp-append-version="true"></script>
</environment>
<environment exclude="Development">
  <script src="https://code.jquery.com/jquery-3.3.1.min.js"
    asp-fallback-src="~/lib/jquery/dist/jquery.min.js"
    asp-fallback-test="window.jQuery"
    crossorigin="anonymous"
    integrity="sha256-2Kok7MbOyxpgUVvAk/HJ2jigOSYS2auK4Pfzbm7uH60=">
  </script>
  <script
    src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js"
    asp-fallback-src="~/lib/bootstrap/dist/js/bootstrap.min.js"
    asp-fallback-test="window.jQuery && window.jQuery.fn && window.jQuery.fn.modal"
    crossorigin="anonymous"
    integrity=
      "sha384-smHYKdLADwkXOn1EmN1qk/HfnUcbVRZyYmZ4qpPea6sjB/pTJ0euyQp0Mk8ck+5T">
  </script>
  <script src="~/js/site.min.js" asp-append-version="true"></script>
</environment>
```

Removing the Default styles

Your ASP.NET project has a default stylesheet. If you're using MVC 5, you'll find this in **Content/Site.css**. In .NET Core, this file is found in **wwwroot/css/site.css**.

While I appreciate the effort that Microsoft's put into these styles, they'll just trip us up as we move forward.

For example, in MVC 5, all inputs have a `max-width` of 280px. This is… fine, I guess? But it's not responsive and on some screens this will look too short.

My recommendation? Just delete all of the styles you find in this file.

Before you move on, **Site.css** should be empty.

Basic Entity Modeling

In order to do anything non-trivial with Bootstrap, like forms or modals, we need to create a model and a table in the database.

For the rest of this book, we'll pretend that we're building an app for a veterinarian's office to keep track of pet appointments.

Creating a Model

If your project doesn't have a **Models** folder, create one. In the **Models** folder, create a new C# class called `Visit`.

Visit.cs

```
public class Visit
{
  public int ID { get; set; }
  public string PetName { get; set; }
  public DateTime VisitDate { get; set; }
}
```

We're going to start with this extremely simple representation, and then improve and add as we go.

Next, we need to add our Visit class to the existing `DbContext`. Head to ***Models/IdentityModels.cs*** and find the `ApplicationDbContext` class. (If you're using .NET Core, this can be found in **Data/ApplicationDbContext.cs**).

Add the following line to the class:

```
public DbSet<Visit> Visits { get; set; }
```

In this case, the `ApplicationDbContext` class was already written for us since we choose "Individual Account" authentication when we began the project.

If you chose a different method, like "No Authentication," you'll have to make your own class that inherits from `DbContext`.

Here's an example of creating your own `DbContext` in both of our stacks:

MVC 5

In your **Models** folder, create a new file named `ApplicationContext.cs`.

```
public class ApplicationContext : DbContext
{
  public ApplicationContext() : base("name=default")
  {
  }
  public DbSet<Visit> Visits { get; set; }
}
```

You'll also need a connection string in your `web.config` named `default`.

```
<connectionStrings>
  <add
    name="default"
    connectionString="Server=(localdb)\mssqllocaldb;
      Database=TestDb;Trusted_Connection=True;ConnectRetryCount=0"
    providerName="System.Data.SqlClient"
  />
</connectionStrings>
```

.NET Core

Inside the **Models** folder, create a file called `ApplicationContext.cs`.

```
public class ApplicationContext : DbContext
{
  public ApplicationContext
  (DbContextOptions<ApplicationContext> options)
  : base(options)
  { }

  public DbSet<Visit> Visits { get; set; }
}
```

Now pop into `appsettings.json`. We'll be providing a connection string to a test database. If you chose Individual Account authentication earlier, this part is probably already set up for you.

```
{
  "ConnectionStrings": {
    "Default": "Server=(localdb)\\mssqllocaldb;Database=TestDb;
      Trusted_Connection=True;ConnectRetryCount=0"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

In `Startup.cs`, you'll now want to add the DbContext with this line at the end of the `ConfigureServices` method:

```
services.AddDbContext<ApplicationContext>(options =>
  options.UseSqlServer(Configuration.GetConnectionString("Default"))
);
```

Running Database Migrations

Once that's done, we need to create our database and tables. Go to the Package Manager Console in Visual Studio.

If you're using MVC 5, you'll need to run this command first:

```
enable-migrations
```

Then, for both MVC 5 and .NET Core, run the following commands, one after the other:

```
add-migration initial
update-database
```

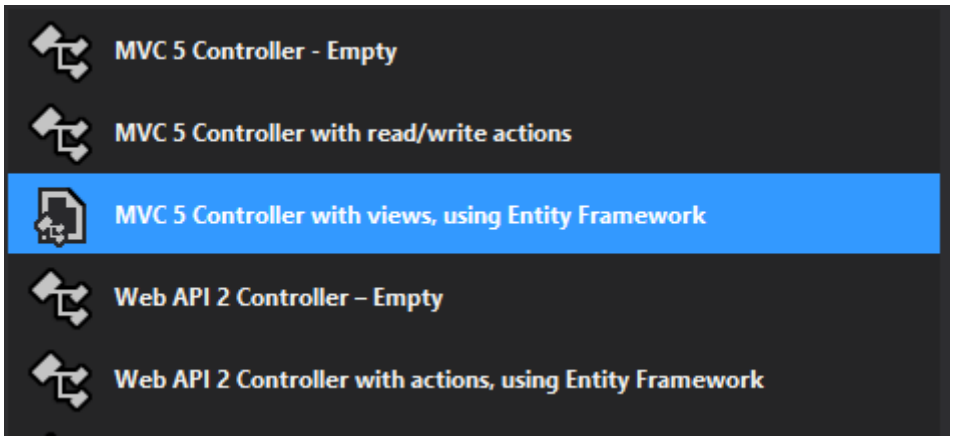
Great! We should have a working database now.

Next, we need to create CRUD pages. In MVC 5, this will take the form of Controllers, and in .NET Core we'll be using Razor Pages.

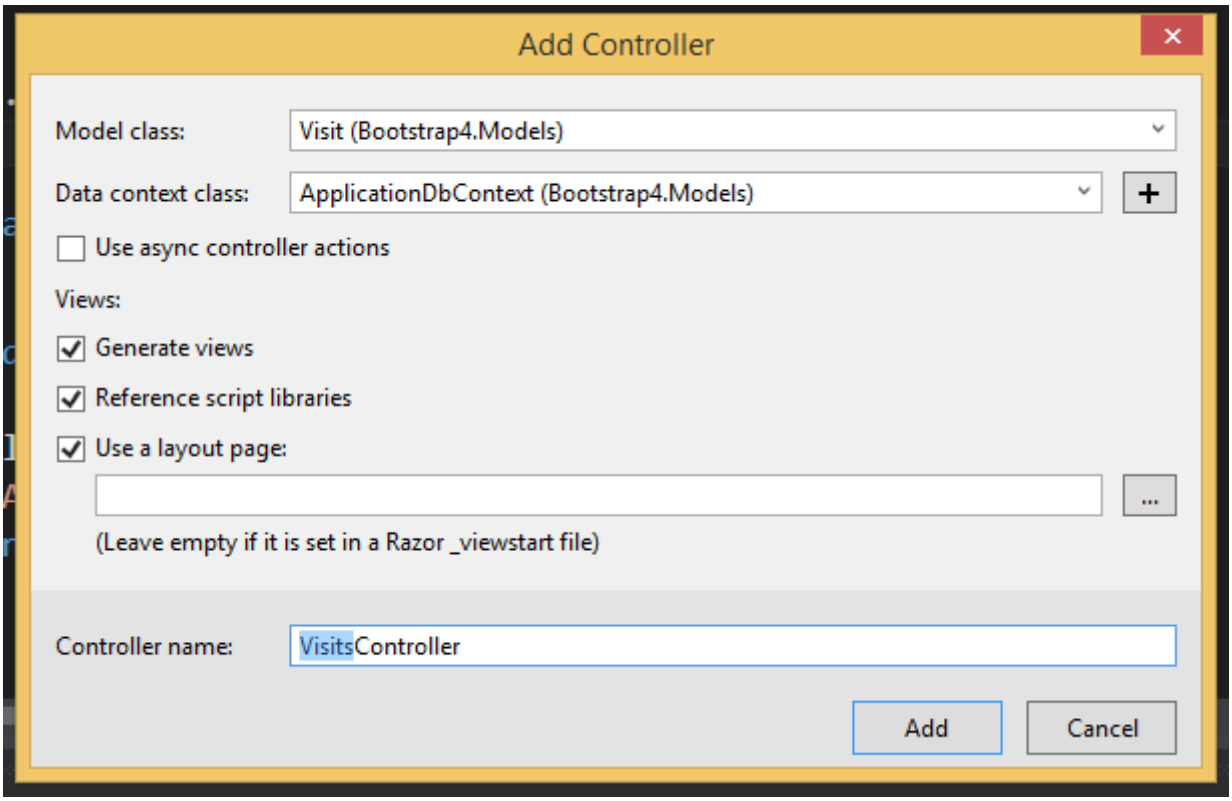
MVC 5

Right-click on the **Controllers** folder and select **Add** → **Controller**...

On the next dialog, choose **MVC 5 Controller with views, using Entity Framework**. Click on **Add**.



On the following dialog, choose the **Visit** model class and the **ApplicationDbContext** data context class.



Click on **Add** and Visual Studio will scaffold both your controller class (in the **Controller** folder)

and the Views (in **Views/Visits**).

.NET Core

In the **Pages** folder, create a subfolder called **Visits**. Right-click it and select:

Add → Razor Page... → Razor Pages Using Entity Framework (CRUD)

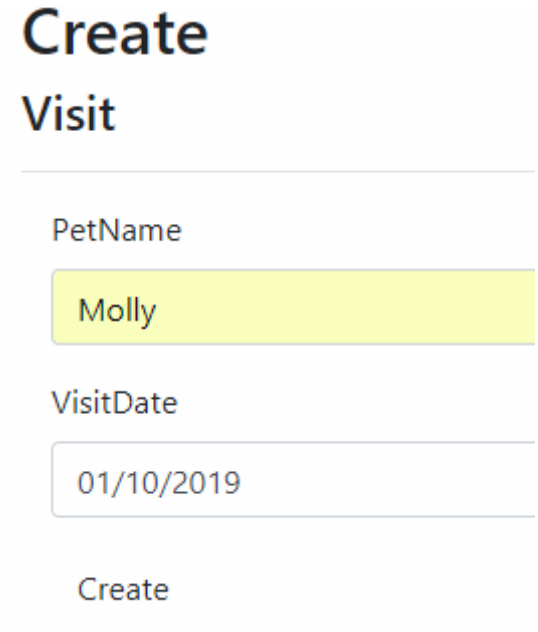
Select **Visit** for the model class and **ApplicationDbContext** for the Data context class. Click **Add** and you should see Visual Studio create your Razor pages.

Checking the Index View

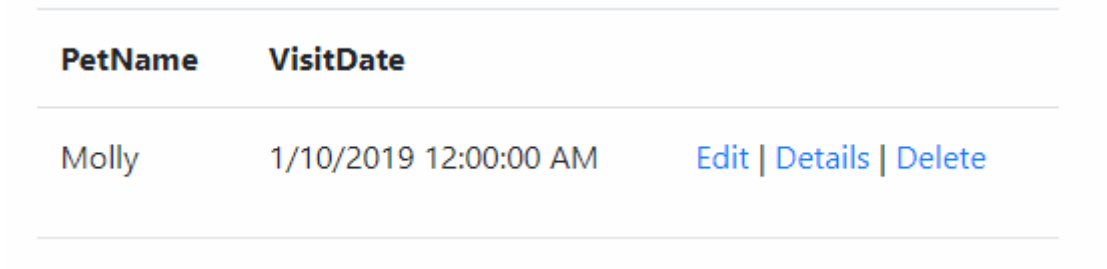
Now that we're done, debug the project locally and check that you can go to:

```
http://localhost:YOURPORT/Visits/Create
```

Try making a new visit and clicking **Create**.



This should redirect you to the Index view, where you'll see the visit you created.



Great! It works.

But, as I'm sure you've noticed, this is still very ugly. The text inputs are too wide, the heading names are written for a computer and not a human, and that date format is ***ugly***.

In the next few chapters, we'll start to fix all of that.

Thanks for Reading

You've reached the end of the sample chapters, but there's much more in the full book!

Check out Bootstrap for .NET Devs[4] right now to purchase the rest.

Be sure to check out the Sensible Dev website too: Sensible Dev[5]

– Matt

[4] https://leanpub.com/bootstrap4dotnet
[5] https://sensibledev.com

Simple Bootstrap Forms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Form Core Structure

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

MVC 5

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

.NET Core

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Adding Data Annotations

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Testing it Out

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Advanced Bootstrap Form Elements

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Text

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Date

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Single Checkbox

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Textarea

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Radio Button

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Number

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Range

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Tel

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Finishing Up the Form

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Adding the Edit, Details, and Delete Views

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Adding View Models

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Creating the View Model

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Updating the Controller / Razor Page

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Updating the Index View

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

AutoMapper

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Installation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Is AutoMapper Worth it?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Bootstrap Modal with AJAX Content

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Adding a Basic Bootstrap 4 Modal

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Creating a Web API to Serve AJAX Content

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Bootstrap Modal with AJAX Content

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Bootstrap Modal with Edit Capabilities

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Bootstrap Modal Form validation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Bootstrap Dynamic Pagination

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

When to Use Pagination

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Seeding Data

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Paginating Table Data

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Non-Paginated List without a Table

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Bootstrap Pagination

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Razor Page

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

More to come!

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.

Appendix A: Source Code

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/bootstrap4dotnet.