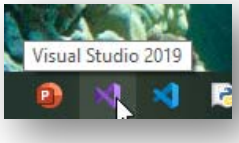
CISS 233 C# Programming

MVC Tutorial 2

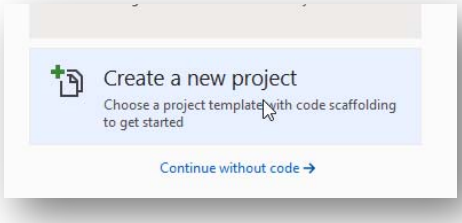
Adding additional Views and Controllers, Shared Layouts

In the last tutorial I went through creating an MVC application, adding a form in the View to communicate to the Controller and how to communicate between the Controller and View and back again. In this tutorial I'll cover writing additional Views/Controllers and implementing a common Layout so that all of your pages have a consistent look and feel.

I'm going to create a new project for this tutorial. It's going to have multiple Views and Controllers. To begin with we'll just work with the Home View/Controller and Layout. From there we'll add an About me View/Controller as well as a simple form View/Controller. We'll start talking about Models in the next module.



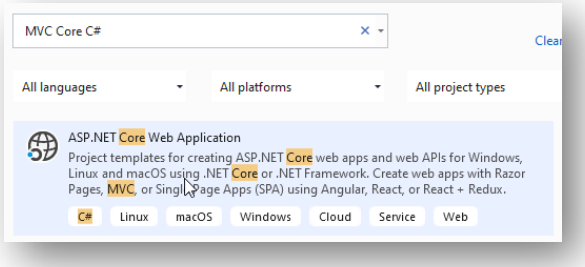
I'm going to open Visual Studio.

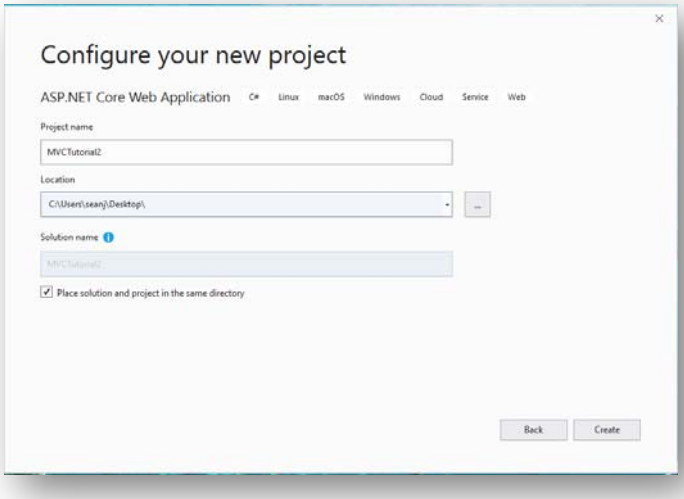


I'll choose to create a new project of type ASP .NET Core Web Application C#.

If it doesn't come up in your recently used project templates then just type MVC Core C# in the search box and you should be given options to choose from with the first one being the project type you're looking for.

Once you highlight the correct template then click next.



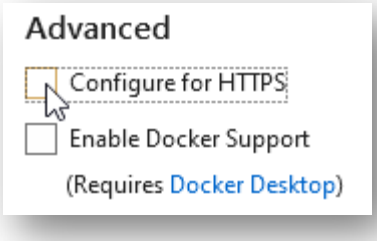


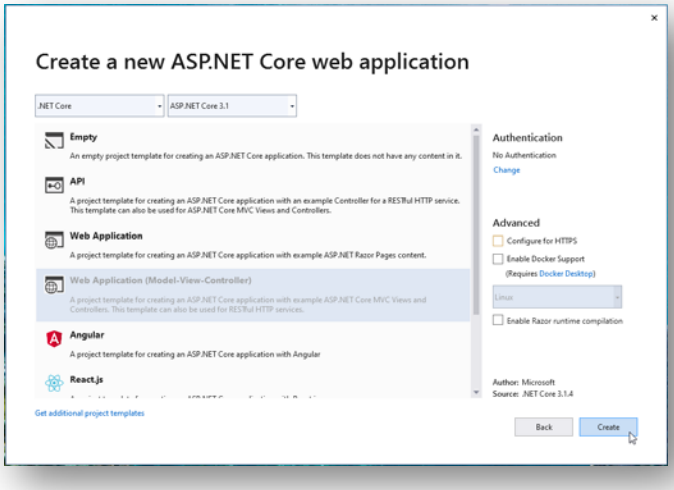
I'm going to name it MVCTutorial2 and just set its location as my desktop.

Be mindful of where the application is saved to. If you don't know then you can always open it up from your recent applications when you first open Visual Studio and then right click on the solution to see its properties and where it is located.

Click Create to create it.

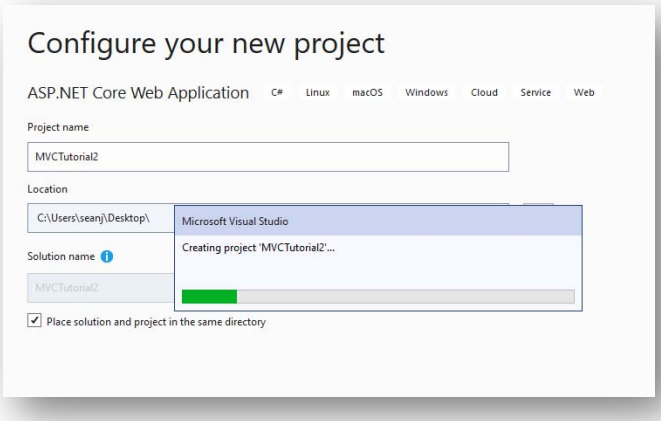
The next screen is the template types page. I'm going to choose Web Application (Model-View-Controller) and I'm also going to deselect "Configure for HTTPS".



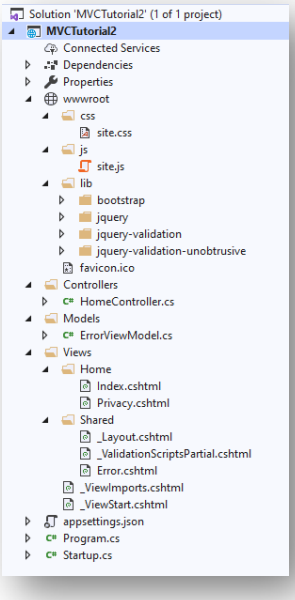


If this was an application that I'd be deploying to a website then I would probably select this but it isn't so I won't.

Click Create to finish and create your app.



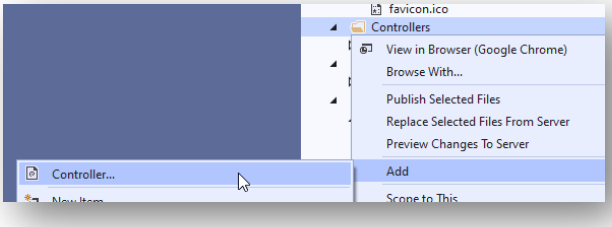
By default, when you choose Web Application (Model-View-Controller) you'll be given a wwwroot folder containing all of your common css, JavaScript and third-party libraries. You'll have a Controllers folder with a HomeController, a Models folder with an ErrorViewModel which we won't be discussing in this tutorial, and two Views folders. Those View folders, Home and Shared, each contain components for those Views.

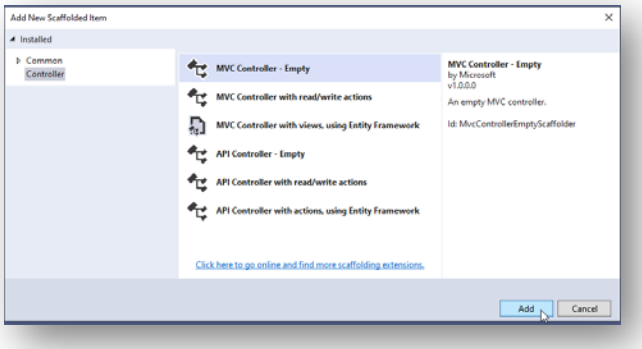


Shared isn't really a View as much as it has components that can be used in any Views you create. This includes the _Layout.cshtml, _ValidationScriptsPartial.cshtml, and Error.cshtml.

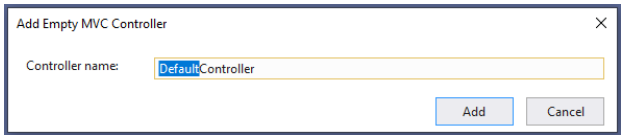
Let's go ahead and create a new Controller and its View for our AboutMe page.

I'll right-click on my Controllers folder. I'll select Add Controller.





Choose the MVC Controller – Empty



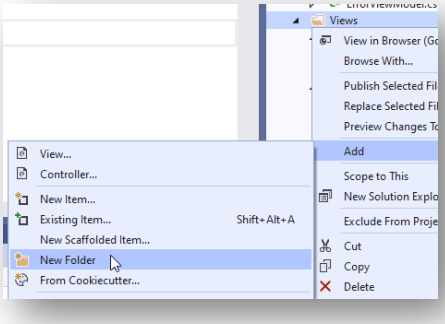
Change the Controller name to AboutMeController.

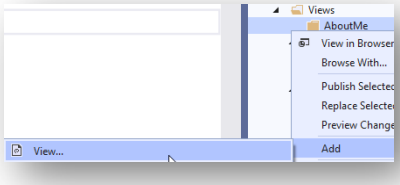
Next we'll add a View called AboutMe. A View unlike a Model usually has its own Folder. And inside of that folder is usually an index page that contains the code for the View.

Let's first create the folder that the new View will live in.

I can right-click on Views and click Add and New Folder.

I'll name it AboutMe





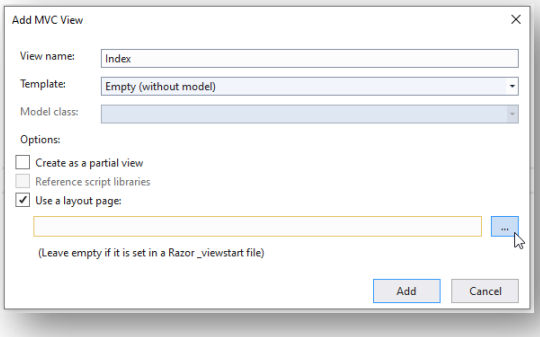
The View's name will be Index. It will be inside of the AboutMe folder.

Next, I'll name the View Index.

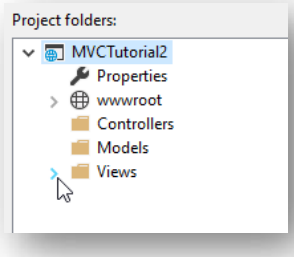
This View will be Empty (without model).

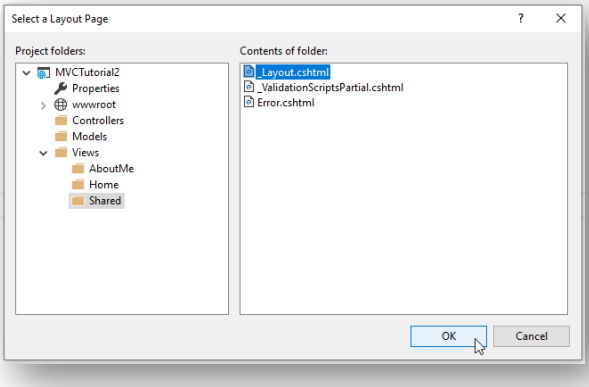
We will connect it to our common Layout.

By clicking the button with the three periods I can navigate to where I can connect the common Layout.

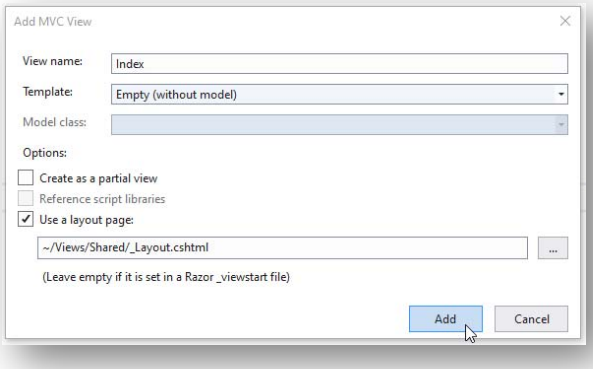


I'll choose the Views folder.



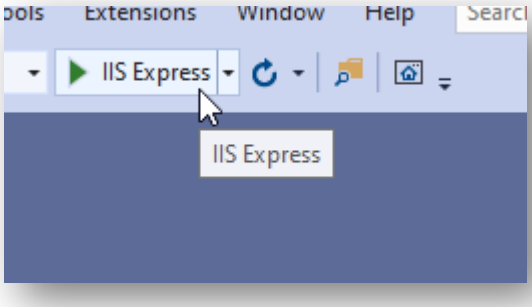


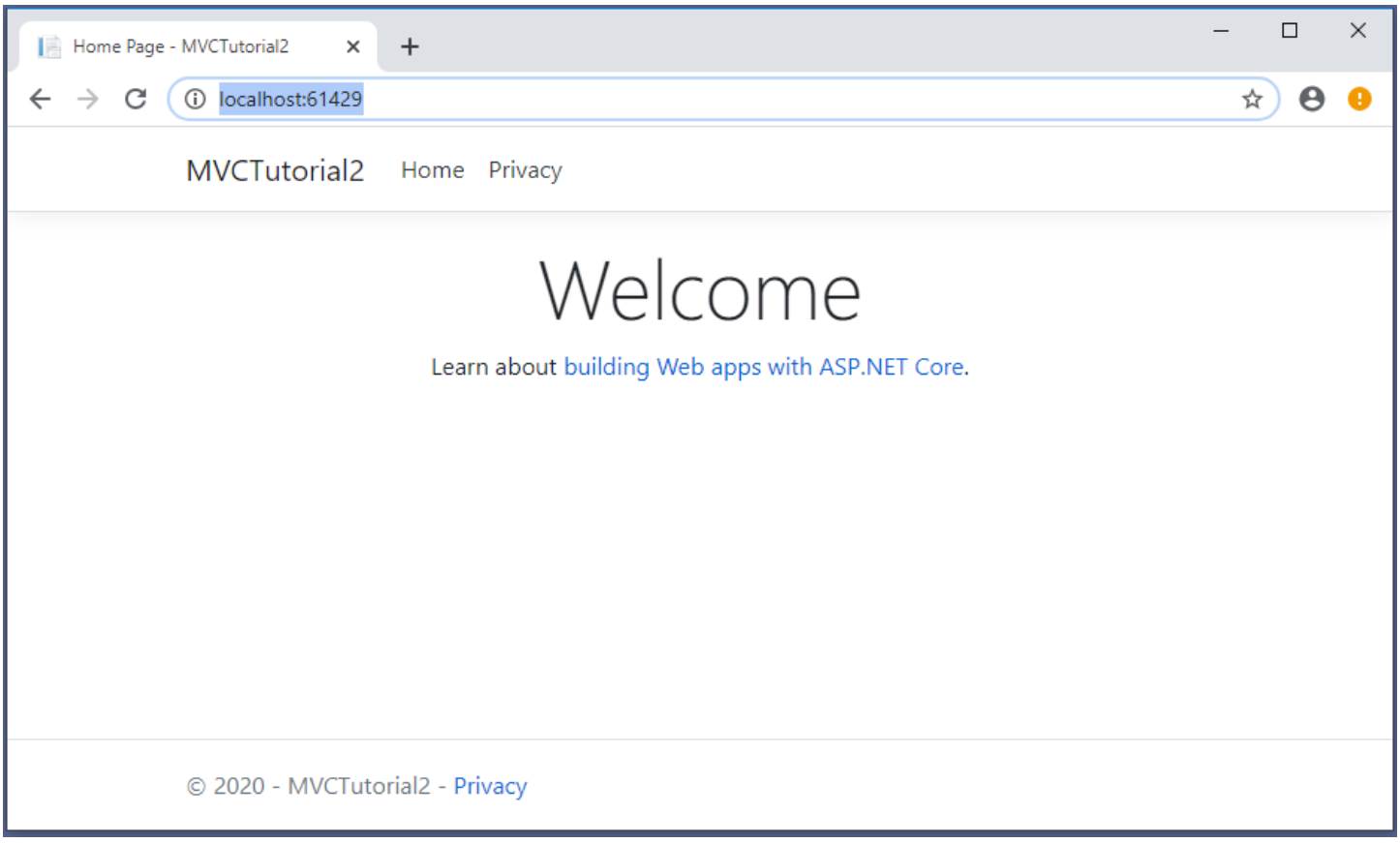
And inside of the Views folder Shared and then _Layout.cshtml.



Click Add to complete the creation of the Index View under the About Me folder in Views.

Let's run our project now. If you click on the green play button in the debug menu bar you will be able to start the project.





You'll see the Welcome screen for the app which is the Home View. In the menu bar you'll also see Home and Privacy. This menu, as well as the styling for the page are all common elements linked to in the shared Layouts page. Let's look at the Layouts page and make some changes.

Here is the code for the Layouts page.

_Layout.cshtml

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="utf-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>@ViewData["Title"] - MVCTutorial2</title>
7      <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
8      <link rel="stylesheet" href="~/css/site.css" />
9  </head>
10 <body>
11     <header>
12         <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
13             <div class="container">
14                 <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">MVCTutorial2</a>
15                 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-controls="navbarSupportedContent"
16                         aria-expanded="false" aria-label="Toggle navigation">
17                     <span class="navbar-toggler-icon"></span>
18                 </button>
19                 <div class="navbar-collapse collapse d-sm-inline-flex flex-sm-row-reverse">
20                     <ul class="navbar-nav flex-grow-1">
21                         <li class="nav-item">
22                             <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
23                         </li>
24                         <li class="nav-item">
25                             <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
26                         </li>
27                     </ul>
28                 </div>
29             </div>
30         </nav>
31     </header>
32     <div class="container">
33         <main role="main" class="pb-3">
34             @RenderBody()
35         </main>
36     </div>
37
38     <footer class="border-top footer text-muted">
39         <div class="container">
40             &copy; 2020 - MVCTutorial2 - <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
41         </div>
42     </footer>
43     <script src="~/lib/jquery/dist/jquery.min.js"></script>
44     <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
45     <script src="~/js/site.js" asp-append-version="true"></script>
46     @RenderSection("Scripts", required: false)
47 </body>
48 </html>
```

Notice first that there are some links which are the navigation in the page. These are line 22, line 25, and in the footer line 40. Notice also that line 34 @RenderBody() is what pulls the code in from the actual View pages linking them to this Layout Template.

Line 8 is a link to our common stylesheet site.css in wwwroot/CSS, and our common JavaScript is in site.js inside of wwwroot/js.

I'm never going to use Privacy so I'll remove those links and delete the cshtml file inside of my Home Controller. I'm first going to remove the anchor tag from line 40. I could leave it in there but I want to delete the privacy page.

This:

```
38  <footer class="border-top footer text-muted">
39      <div class="container">
40          &copy; 2020 - MVCTutorial2 - <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
41      </div>
42  </footer>
```

Becomes this:

```
38  <footer class="border-top footer text-muted">
39      <div class="container">
40          &copy; 2020 - MVCTutorial2
41      </div>
42  </footer>
```

I'm going to change the link on line 25 to reference the new AboutMe Controller/View.

This:

```
24                  <li class="nav-item">
25                      <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
26                  </li>
```

Becomes:

```
24                  <li class="nav-item">
25                      <a class="nav-link text-dark" asp-area="" asp-controller="AboutMe" asp-action="Index">About Me</a>
26                  </li>
```

I'm going to add some Code to AboutMe/Index.cshtml so I can see the changes when it runs.

```
1
2   @{
3       ViewData["Title"] = "Index";
4       Layout = "~/Views/Shared/_Layout.cshtml";
5   }
6
7   <h1>Index</h1>
8
9
```

Now will have this code:

```
1
2   @{
3       ViewData["Title"] = "About Me";
4       Layout = "~/Views/Shared/_Layout.cshtml";
5   }
6
7   <h1>About me</h1>
8
9   <h3>Sean O'Keefe</h3>
10  <h3>HVCC</h3>
```

I'm going to change the background color for the app. So in wwwroot/css/site.css I'll open that up and starting at line 61 I'm going to add a background-color of lightsalmon to the body tag.

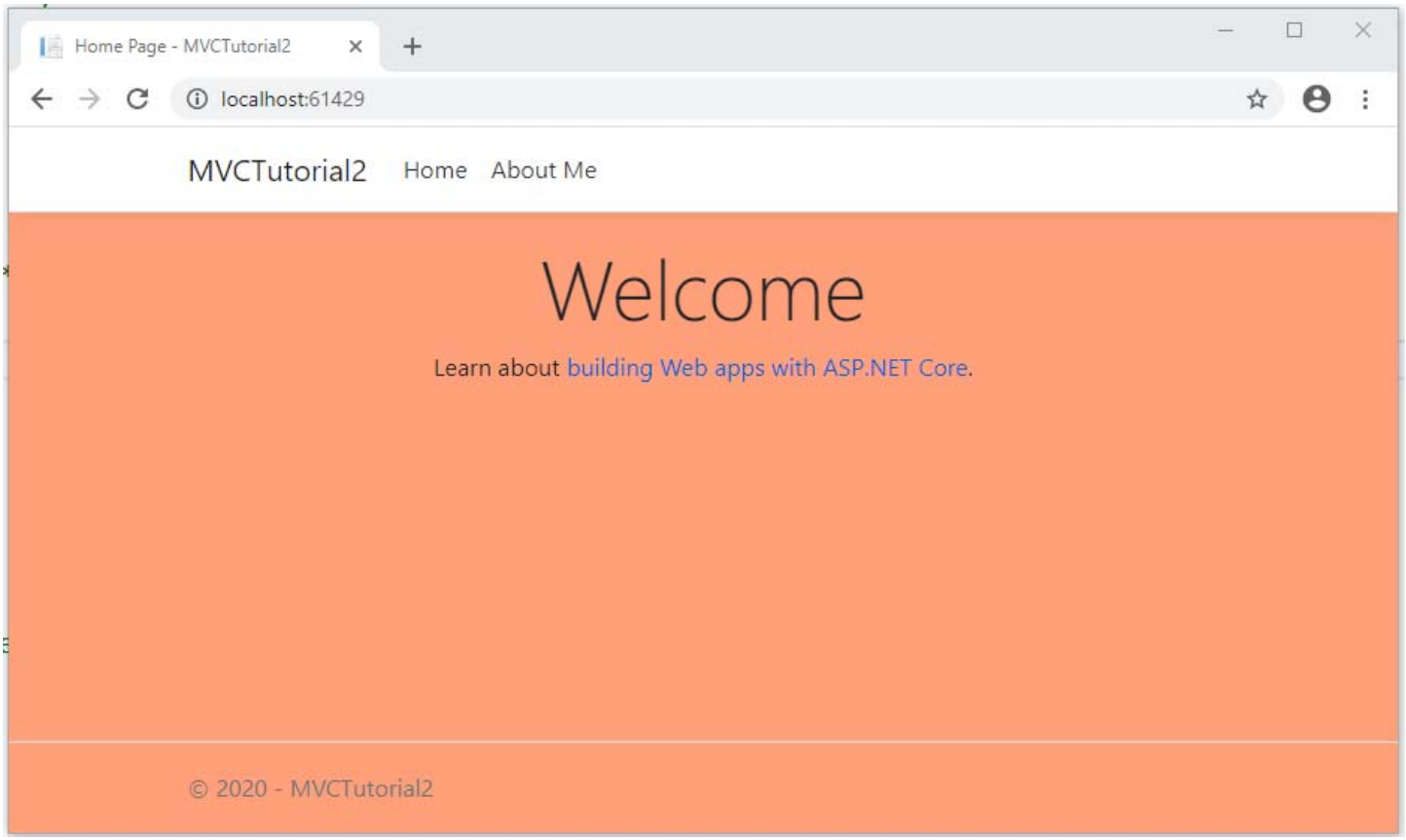
From this:

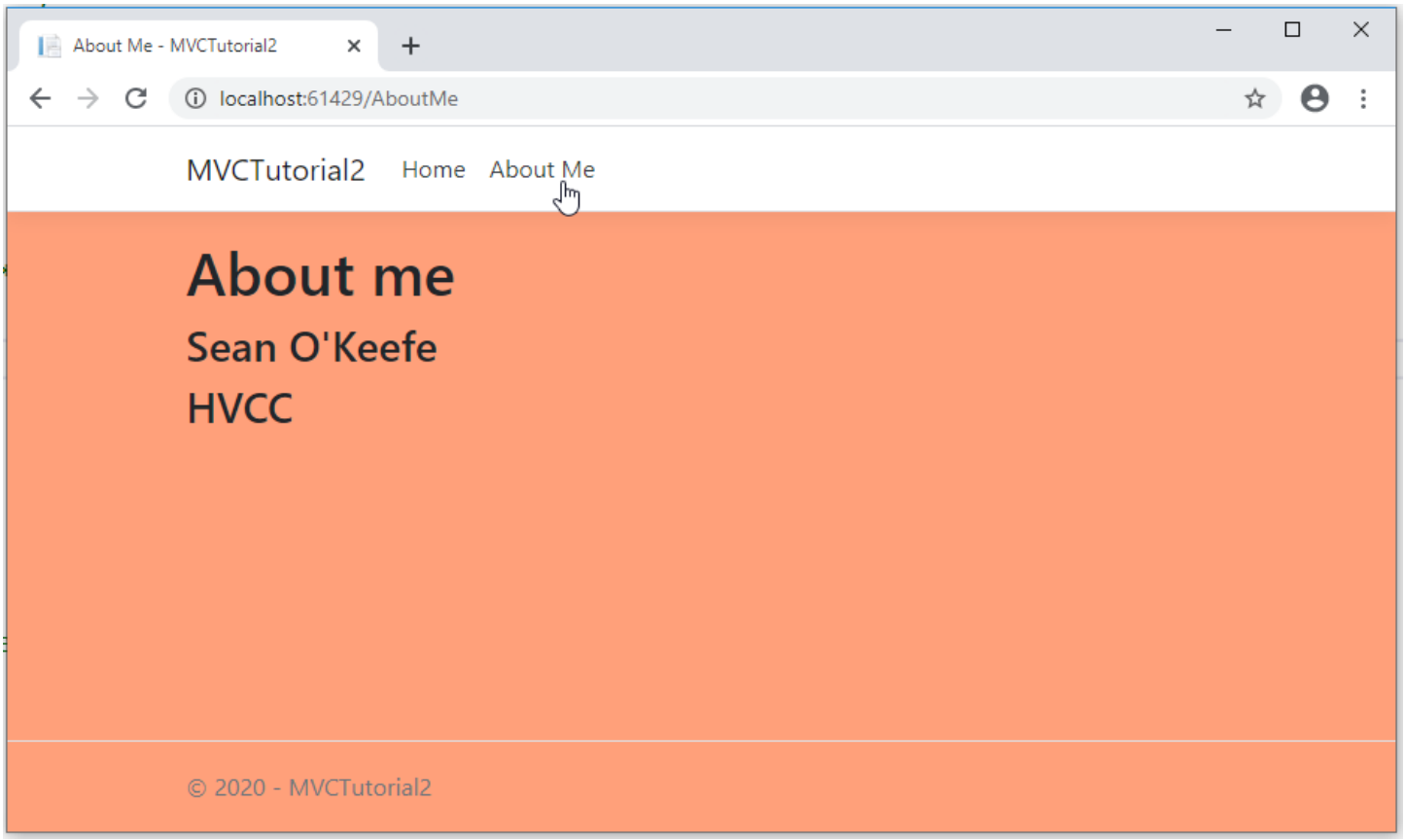
```
61  body {
62      /* Margin bottom by footer height */
63      margin-bottom: 60px;
64  }
```

To this:

```
61  body {
62      /* Margin bottom by footer height */
63      margin-bottom: 60px;
64      background-color:lightsalmon;
65  }
```

Now when I run it, I'll see the new page color as well as the link to the new page in the common Layout.


Home Page - MVCTutorial2
localhost:61429
MVCTutorial2
Home
About Me
Welcome
Learn about building Web apps with ASP.NET Core.
© 2020 - MVCTutorial2


About Me - MVCTutorial2
localhost:61429/AboutMe
MVCTutorial2
Home
About Me
About me
Sean O'Keefe
HVCC
© 2020 - MVCTutorial2

I'll create one more page and that's a form to get information from the user and display it in a results page. For this I'll create One Controller called InfoFormController and a View folder called InfoForm with two separate views, one called Index where the form will be and another called results which will show the results from the form. I'll also create one more link in the navigation in shared/Layout so that we can get to the Index page under the InfoForm View Folder.

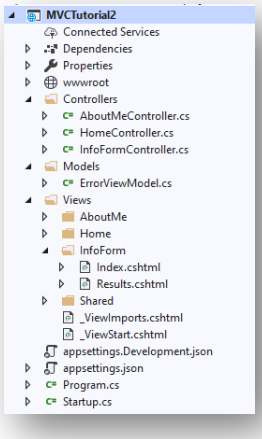
I'll create the Controller following the same steps as I did before.

Controllers -> InfoFormController

Then I'll create the View folder with its two view pages.

Views -> InfoForm -> Index.cshtml, Results.cshtml

The end result is this to the right:



I'll add a link in the shared layout page to the InfoForm Index page. I've added it to _Layout.cshtml.

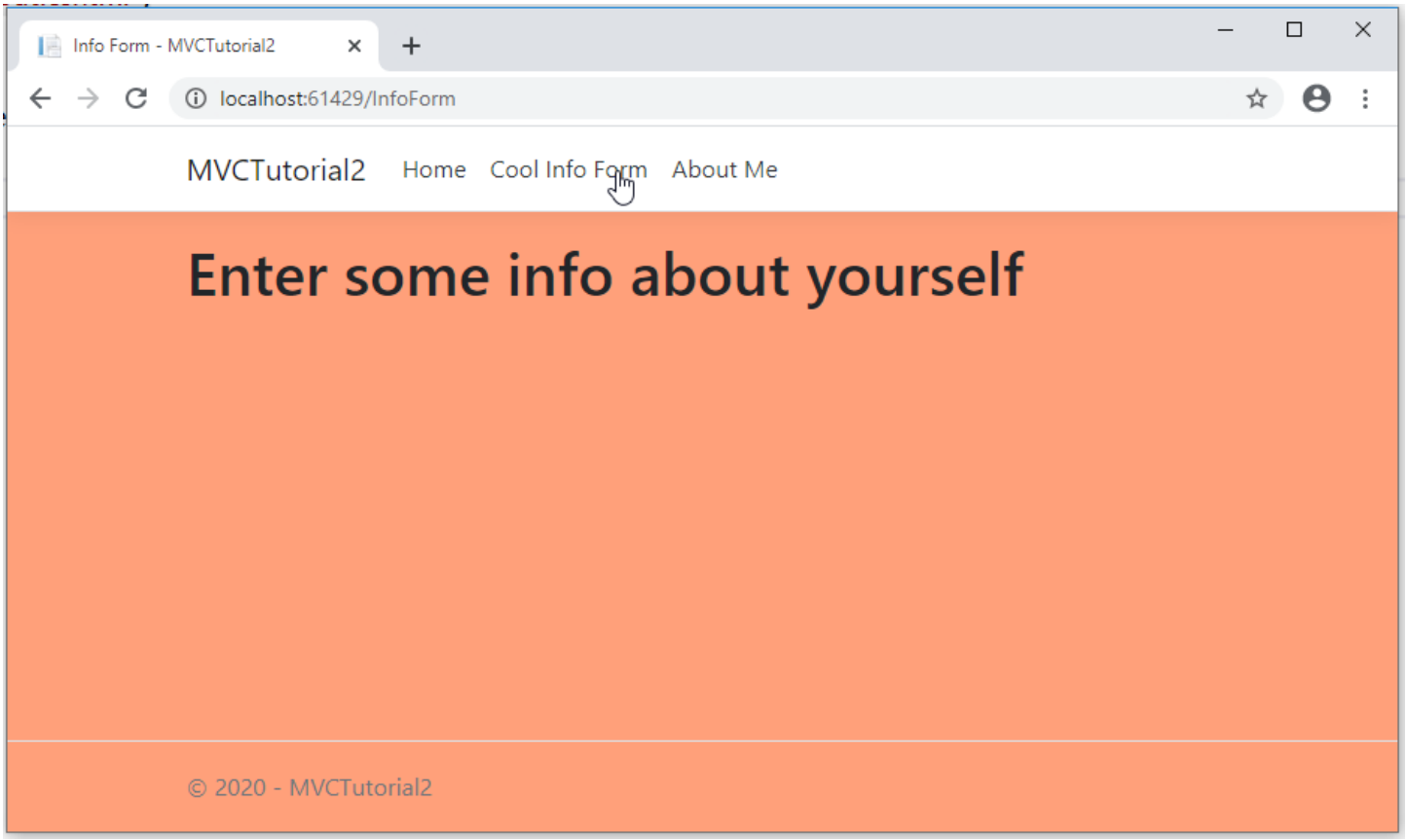
```
21              <li class="nav-item">
22                <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
23              </li>
24              <li class="nav-item">
25                <a class="nav-link text-dark" asp-area="" asp-controller="InfoForm" asp-action="Index">Cool Info Form</a>
26              </li>
27              <li class="nav-item">
28                <a class="nav-link text-dark" asp-area="" asp-controller="AboutMe" asp-action="Index">About Me</a>
29              </li>
```

I've added this code to the Index.cshtml under the InfoForm View folder.

```
1
2   @{
3       ViewData["Title"] = "Info Form";
4       Layout = "~/Views/Shared/_Layout.cshtml";
5   }
6
7   <h1>Enter some info about yourself</h1>
8
```

So next I'll run the code again and click on that Info Form link and see what happens.

Looks good:



I'm going to build out the form and I won't go through these steps because I went through these steps in the previous tutorial. It's going to have a form that includes a first and last name text box, gender radio buttons, birth month select (dropdown) along with a submit and clear button. All controls will use HTML5 validation on submittal. That means the tag required. When the form is submitted it will go back to the controller and send the information to the new view results.

The code for the form is on the next page. After I'm done with the code for the form I'm going to wire up the controller so that initially it routes to the index page with the form, but when the form is submitted it routes to the results page to display the results.

One thing I will point out before I show the code is that the form tag on the index page for InfoForm has a property of asp-controller="InfoForm". This tells it to route to the controller with that name. It also has a asp-action="ShowResults". This tells the controller which method to send the data to in the controller. I've also set the method as get in this week's example instead of post so you can see in the URL the values being passed. You almost never use get as the method because it does show those values in the URL so if it was a secure application then you wouldn't want that.

Here is the code for the form in Index.cshtml under the InfoForm View folder.

```
@{
    ViewData["Title"] = "Info Form";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h1>Enter some info about yourself</h1>
<form asp-controller="InfoForm" asp-action="ShowResults" method="get">
    <p>
        First Name:
        <input id="tbFirstName" name="first" type="text" required />
    </p>
    <p>
        Last Name:
        <input id="tbLirstName" name="last" type="text" required />
    </p>
    <p>
        Gender:<br />
        <input id="rbFemale" name="gender" type="radio" value="female" required />
        <label for="rbFemale">Female</label>
        <input id="rbMale" name="gender" type="radio" value="male" required />
        <label for="rbMale">Male</label>
    </p>
    <p>
        Birthday Month:<br />
        <select id="ddBirthMonth" name="month" required>
            <option value="">Please select a month...</option>
            <option value="January">January</option>
            <option value="February">February</option>
            <option value="March">March</option>
            <option value="April">April</option>
            <option value="May">May</option>
            <option value="June">June</option>
            <option value="July">July</option>
            <option value="August">August</option>
            <option value="September">September</option>
            <option value="October">October</option>
            <option value="November">November</option>
            <option value="December">December</option>
        </select>
    </p>
    <br />
    <input id="btnSubmit" type="submit" value="submit" />
    <input type="reset" value="reset" />
</form>
```

Here is the code in the controller to match the asp-action of ShowResults

```
16      [HttpGet]
        0 references
17      public IActionResult ShowResults(String first, String last, String gender, String month)
18      {
19          ViewData["first"] = first;
20          ViewData["last"] = last;
21          ViewData["gender"] = gender;
22          ViewData["month"] = month;
23          return View("Results");
24      }
```

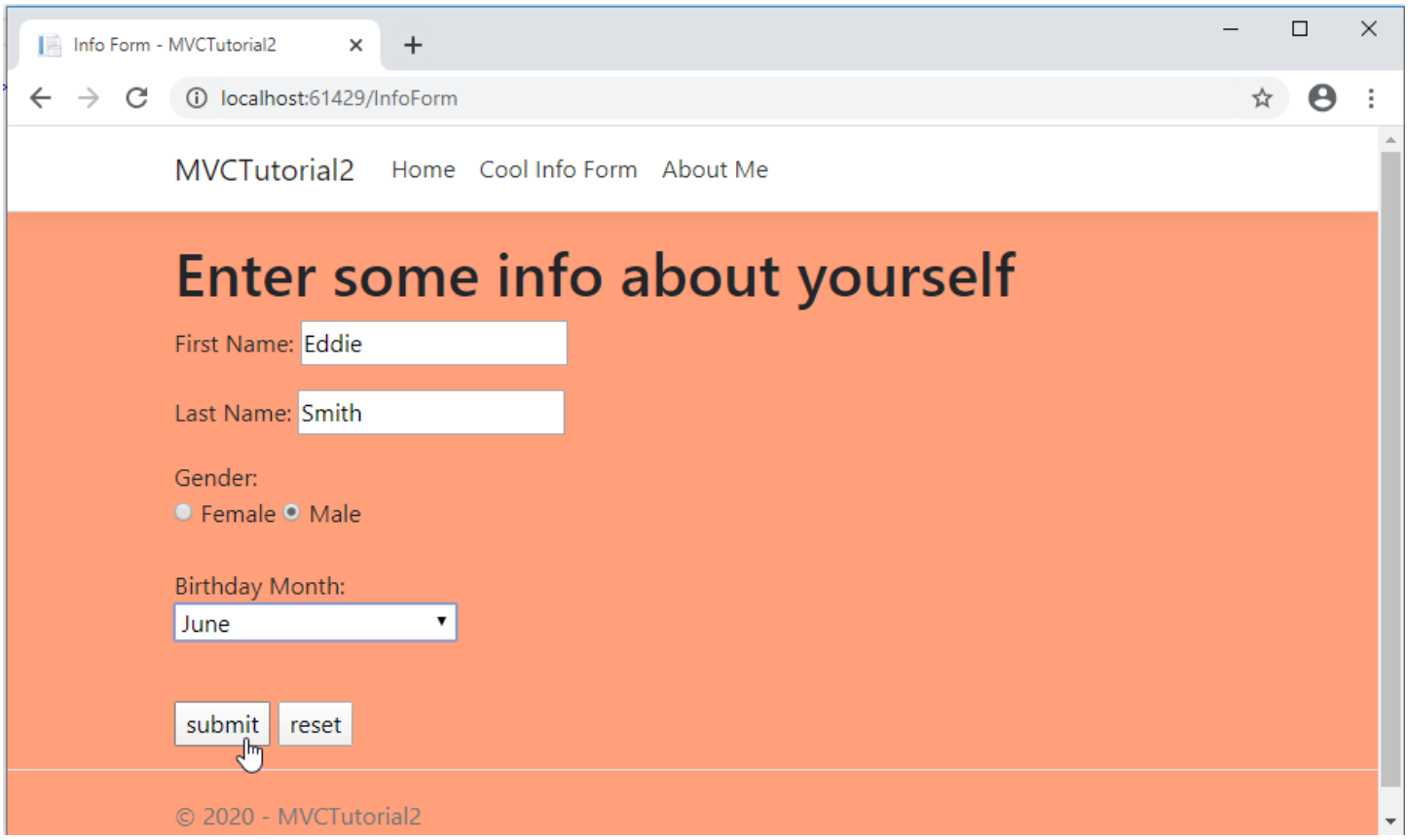
We're calling the ShowResults method on submittal. We pass in all form fields with a name attribute. These include first, last, gender, and month. In the method we assign them to the ViewData object with the names as the key. The data becomes the value for each data element.

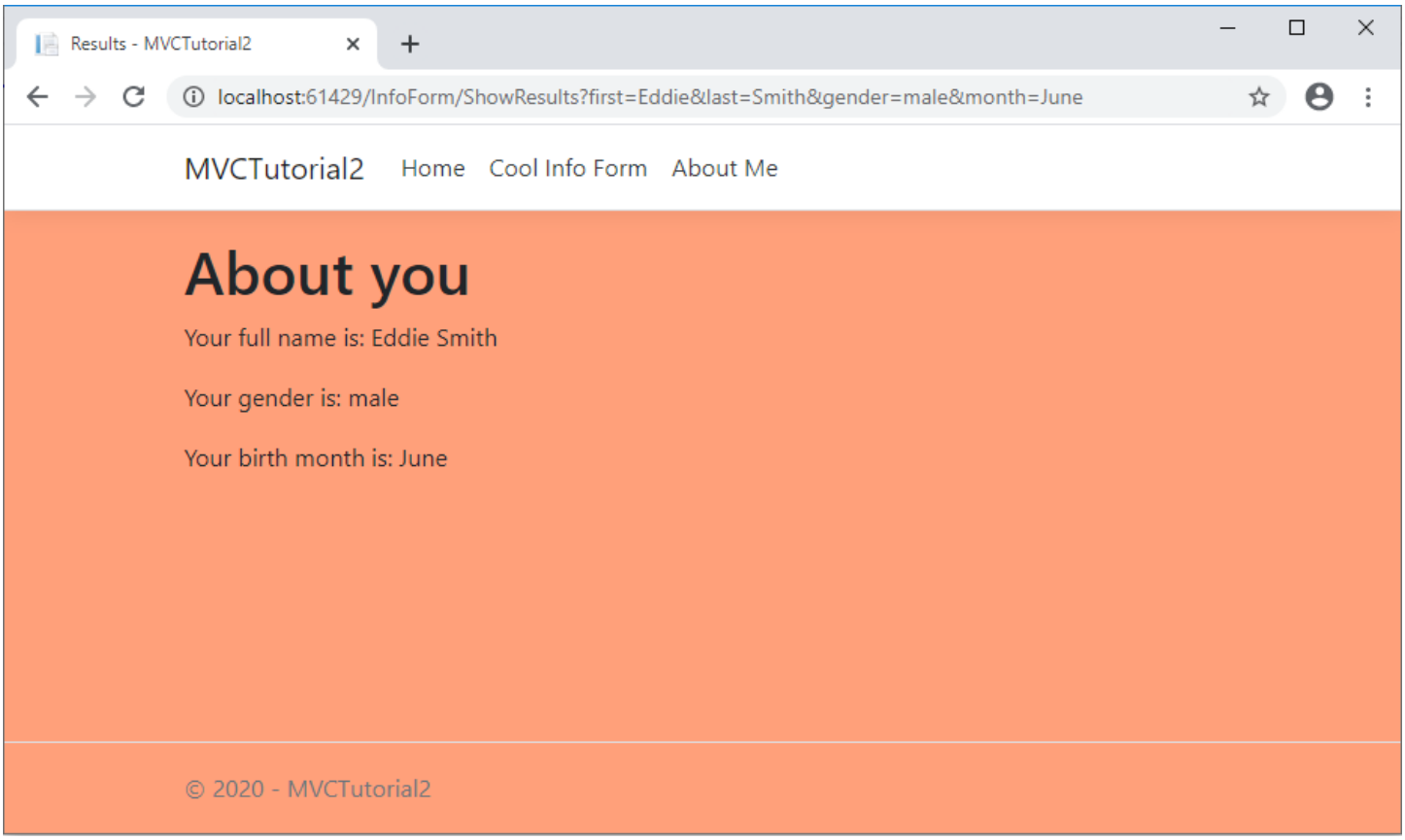
We return the View but to the Results.cshtml view page. This code on line 23 will reroute the response to Results.

In results we will display the data to the screen using ViewBag which is the Object equivalent of the ViewData Key->Value associative array object.

```
1
2   @{
3       ViewData["Title"] = "Results";
4       Layout = "~/Views/Shared/_Layout.cshtml";
5   }
6
7   <h1>About you</h1>
8
9   <p>Your full name is: @ViewBag.first @ViewBag.last</p>
10  <p>Your gender is: @ViewBag.gender</p>
11  <p>Your birth month is: @ViewBag.month</p>
```

The result is the form and results screen on the following page:


Info Form - MVCTutorial2
localhost:61429/InfoForm
MVCTutorial2
Home
Cool Info Form
About Me
Enter some info about yourself
First Name: Eddie
Last Name: Smith
Gender:
Female
Male
Birthday Month:
June
submit
reset
© 2020 - MVCTutorial2


Results - MVCTutorial2
localhost:61429/InfoForm/ShowResults?first=Eddie&last=Smith&gender=male&month=June
MVCTutorial2
Home
Cool Info Form
About Me
About you
Your full name is: Eddie Smith
Your gender is: male
Your birth month is: June
© 2020 - MVCTutorial2

There is one last thing I'd like to add. The startup (initial) page in our app is Home->Index. You can set this in Startup.cs. In our example in Startup.cs line 49 says this:

```
45      app.UseEndpoints(endpoints =>
46      {
47          endpoints.MapControllerRoute(
48              name: "default",
49              pattern: "{controller=Home}/{action=Index}/{id?}");
50      });
```

To change the starting page just change this to whatever controller and view you'd like to. I'm going to keep it Home->Index but just know that this is where you change it.

Attached to this tutorial in BlackBoard should be the code in a zip file.

If you have any questions then email me at s.okeefe@hvcc.edu