

Bootstrap 4

Tmpst uses Bootstrap 4 as the main frontend framework for the application. Tmpst uses Bootstrap to create the consistent Navbar throughout the application as well as using the grid system to align the containers of the pages correctly for both desktop and mobile users.

Bootstrap 4 is a framework for front-end languages such as JavaScript, CSS as well as HTML. Bootstrap is the most popular framework for these languages and with the new release of Bootstrap 4 is better than ever at providing a framework for mobile-first application

In this tutorial:

Pre-requisites:

- Visual Studio 2017 (Community, Professional or Enterprise)
- .NET Core v2.2
- .html or .cshtml files you want to add bootstrap to
- ASP.NET Core Web Application
- Bootstrap 4 Source Files
(<https://getbootstrap.com/docs/4.0/getting-started/download/>)
- Popper.js (Local or cdn. Ddn version available at:
<https://getbootstrap.com/docs/4.0/getting-started/download/>)
- JQuery v3.3.1 (<https://jquery.com/>)

Aims:

- Learn how to create a Bootstrap 4 Navbar
- Learn how to create a page using the Bootstrap 4 grid system

Note: In this tutorial, the code used as an example is similar to that which is on the Bootstrap website. There will be links to other tutorials if this one does not suit your specific needs

Navbar Tutorial

In this tutorial, we will create a sample bootstrap navbar using basic ASP links to controller methods

Steps:

1. Remove any pre-existing navbar
2. Add Bootstrap 4 source files to _Layout (or other view used as a layout page)
3. Add Navbar code to application

Step 1: Remove any pre-existing navbar

If you have just created a new web application for this tutorial, or are using a pre-existing app with limited content, the chances are that you will already have a Bootstrap navbar setup. However, this navbar will have been defaulted to Bootstrap version 3 which is no longer the most up-to-date version. A default version 3 navbar will look something like what is displayed below:

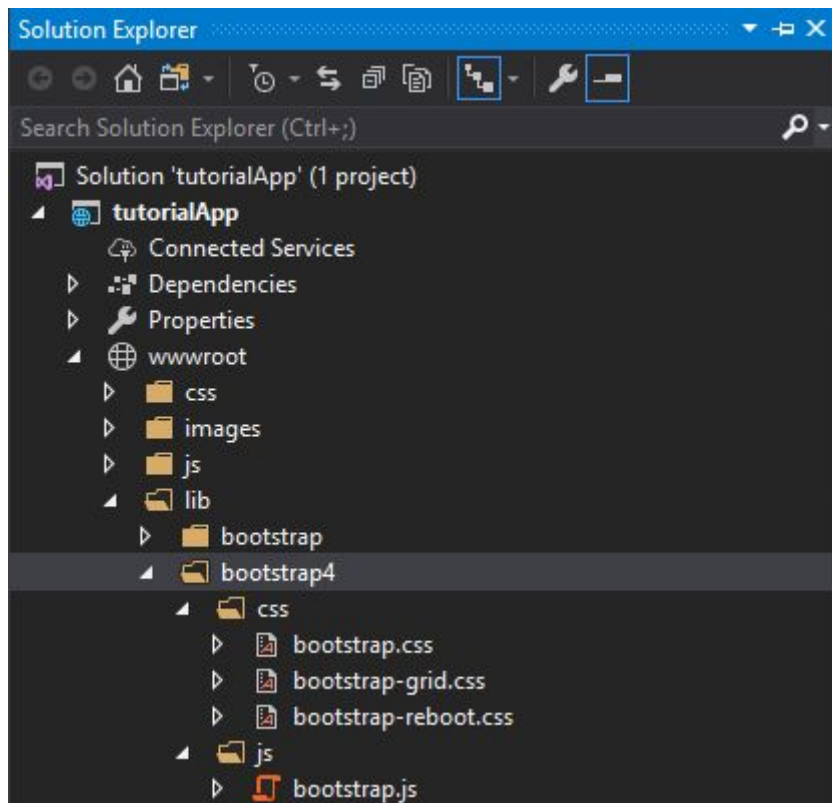
```
<nav class="navbar navbar-inverse navbar-fixed-top">
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a asp-area="" asp-controller="Home" asp-action="Index" class="navbar-brand">tutorialApp</a>
    </div>
    <div class="navbar-collapse collapse">
      <ul class="nav navbar-nav">
        <li><a asp-area="" asp-controller="Home" asp-action="Index">Home</a></li>
        <li><a asp-area="" asp-controller="Home" asp-action="About">About</a></li>
        <li><a asp-area="" asp-controller="Home" asp-action="Contact">Contact</a></li>
      </ul>
    </div>
  </div>
</nav>
```

You can go ahead and remove this code as we will be replacing it. If you want to be cautious you can comment it out rather than outright deleting it, but, we won't be using it again in this tutorial

Step 2: Add Bootstrap 4 source files to _Layout (or other view used as a layout page)

Now that the old navbar has been deleted, we need to set up the application to use Bootstrap 4. The Bootstrap used by the old navbar had to be referenced somewhere by the application. Chances are that if you open the root directory of your application and navigate to the lib folder, you should find a pre-existing bootstrap folder that contains all the JavaScript and CSS required for Bootstrap version 3. We no longer need this version of Bootstrap, however, it would be a good idea to keep the folder there until we have successfully setup Bootstrap 4.

Create a new folder in the lib folder called bootstrap4 and add the css and js folders located in bootstrap-x-x-x/dist/ to our new folder. Your new solution explorer should look like this:



Now that the folder has been added, we will need to add a reference to Bootstrap 4 into the same file that is referencing the original version 3. If you navigate to your `_Layout` where our version 3 navbar was originally you should see the following:

Head:

```
<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://ajax.aspnetcdn.com/ajax/bootstrap/3.3.7/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" />
  <link rel="stylesheet" href="~/css/site.min.css" asp-append-version="true" />
</environment>
```

The code above will load the locally stored version 3 bootstrap css file when the application is development. When the application is not in Development (Staging or Production), the application will load a minified version of the version 3 css through cdn. If, for whatever reason, the application cannot obtain the css file through cdn it will load the locally stored minified version instead.

Body:

```
<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://ajax.aspnetcdn.com/ajax/jquery/jquery-2.2.0.min.js"
    asp-fallback-src="~/lib/jquery/dist/jquery.min.js"
    asp-fallback-test="window.jQuery"
    crossorigin="anonymous"
    integrity="sha384-K+ctZQ+LL8q6tP7I94W+qzQsfRV2a+AfHIi9k8z8l9ggpc8X+Ytst4yBo/hH+8Fk">
  </script>
  <script src="https://ajax.aspnetcdn.com/ajax/bootstrap/3.3.7/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-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNlIcPD7Txa">
  </script>
  <script src="~/js/site.min.js" asp-append-version="true"></script>
</environment>
```

As with the references in the head, the application will load a local version of the bootstrap.js file when in development and the minified version through cdn when not in development

Reference Version 4:

As mentioned in the pre-requisites, we require the Bootstrap 4 files, JQuery and popper.js. All of these to be referenced **in the correct order** to ensure that Bootstrap 4 is loaded correctly. For this example I will load the unminified local files for development use and the minified file for production use and keep the JavaScript defined in the body and the css in the head. As stated earlier, the references have to be done in a certain order or it will not work:

1. JQuery (minified or unminified)
2. Popper.js (minified or unminified)
3. Bootstrap.js (minified or unminified)

As you may be able to tell, the order only applies to the JavaScript files. There is not order required for the css file as it is the only file of its type that is required. Your _Layout should like similar to the following:

Head:

```
<environment include="Development">
  <link href="~/lib/bootstrap4/css/bootstrap.css" rel="stylesheet" />
</environment>

<environment exclude="Development">
  <link href="~/lib/bootstrap4/css/bootstrap.min.css" rel="stylesheet" />
</environment>
```

The code above shows how to reference our local version of the Bootstrap 4 .css file both in development and outwith. For tutorial purposes, the reference to site.css has been removed

Body:

```
<environment include="Development">
  <script src="~/lib/jquery/dist/jquery.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>
  <script src="~/lib/bootstrap4/js/bootstrap.js"></script>
</environment>
<environment exclude="Development">
  <script src="~/lib/jquery/dist/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>
  <script src="~/lib/bootstrap4/js/bootstrap.min.js"></script>
</environment>
```

The code above shows how to reference our local version of the Bootstrap 4 .js file along with popper.js and JQuery both in development and outwith. For tutorial purposes, the reference to site.js has been removed. As mentioned earlier, these references must be in order for this to work

Make sure that you have removed the references to the version 3 Bootstrap so that there is not conflicting scripts. If you run the app now you will not notice much difference as there is still no navbar. You should check your developer console to ensure no errors have occurred during referencing.

Step 3: Add Navbar code to application

Now that the necessary files have been referenced we can go ahead and add a navbar in. In this example we will use the sample navbar referenced on Bootstrap page and then make some aesthetic changes.

Steps:

1. Add example navbar from Bootstrap website
2. Change style of Navbar (e.g. light to dark)

First of all, we need the example navbar from Bootstrap's website. Copy the first example navbar as is from <https://getbootstrap.com/docs/4.0/components/navbar/> and add it to where the version 3 navbar was. Your _Layout should look like this:

```

<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
          Dropdown
        </a>
        <div class="dropdown-menu" aria-labelledby="navbarDropdown">
          <a class="dropdown-item" href="#">Action</a>
          <a class="dropdown-item" href="#">Another action</a>
          <div class="dropdown-divider"></div>
          <a class="dropdown-item" href="#">Something else here</a>
        </div>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#">Disabled</a>
      </li>
    </ul>
    <form class="form-inline my-2 my-lg-0">
      <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
      <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
    </form>
  </div>
</nav>

```

If you run your application now you will have a navbar the same as the example given on the Bootstrap site. However, what if we want a different theme? After all the default background for an ASP web application is white.

Bootstrap offers multiple themes that can be used to change the appearance of one of their navbars. The full list is available [here](https://getbootstrap.com/docs/4.0/components/navbar/#color-schemes)

<https://getbootstrap.com/docs/4.0/components/navbar/#color-schemes>

For this tutorial we will use the dark theme. This theme contrasts nicely with the default white background. To use the dark theme remove the navbar-light bg-light theme from the nav declaration and replace with navbar-dark and bg-dark

Light theme:

```

<nav class="navbar navbar-expand-lg navbar-light bg-light">

```

Dark theme:

```

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

```

Now you can run the application and see something to what is displayed below:



Grid Tutorial

In this tutorial, we will create a container that contains column to separate. This tutorial assumes that you have already referenced Bootstrap 4 in your _Layout page correctly.

Steps:

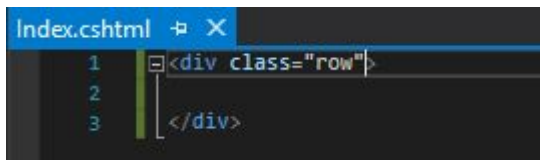
1. Create a row to contain the columns
2. Add the columns to the row

As mentioned earlier, Bootstrap 4 offers a grid system for making content suitable for both desktop and mobile. There are a few rules that need to be considered when using the grid system:

- Columns (class="col") **must be** direct children of rows
- Columns within a particular row cannot exceed 12

Step 1: Create Row

As stated in the rules of the grid system, if we want to use columns we first need to create a row to contain them. Select an view you want that uses the view that references Bootstrap 4 as its layout. For this tutorial I will be using the Index page. Add a div tag to your chosen page which has the class "row" as shown below:

A screenshot of a code editor window titled 'Index.cshtml'. It shows a code block starting with line 1: `<div class="row">`, line 2: (empty), and line 3: `</div>`. A tree view on the left shows the structure of the code block.

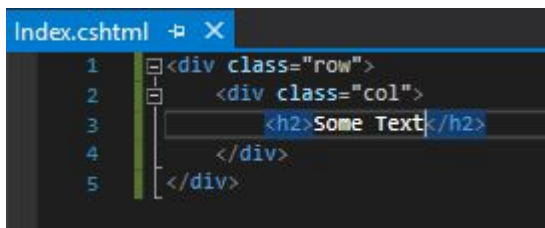
The code to the left will not do anything in particular just now. The class informs the DOM that the div is a Bootstrap row

Step 2: Add Columns to the Row

The content and the size of the columns is completely up to you. For the example I will create two columns which both take up 6 columns of the grid which adds up to exactly 12. The syntax for declaring columns is shown below:

```
<div class="col-{SIZE_OF_COLUMN}">
```

Alternatively, you can declare a column as just "col". This will auto adjust the column depending on the other contents of the row. For example:

A screenshot of a code editor window titled 'Index.cshtml'. It shows a code block starting with line 1: `<div class="row">`, line 2: `<div class="col">`, line 3: `<h2>Some Text</h2>`, line 4: `</div>`, and line 5: `</div>`. A tree view on the left shows the structure of the code block.

The image to the left shows a column with no number assignment. As there are no other columns contained within the parent row, the column will be interpreted as a col-12(full width) at runtime

```

Index.cshtml  [X]
1  <div class="row">
2    <div class="col">
3      <h2>Some Text</h2>
4    </div>
5    <div class="col">
6      <h2>Some More Text</h2>
7    </div>
8  </div>

```

The image to the left shows two columns with no number assignment. As there is two columns in the parent row, the columns will both be interpreted as col-6s. The number assigned to each column is calculated as:
 $12 / (\text{Number of columns})$

If you know the size you want the columns to be, it is always better to specify it. Using just col can be useful when dynamically updating the DOM with JQuery or when using a List if items.

In my column example I have added the following:

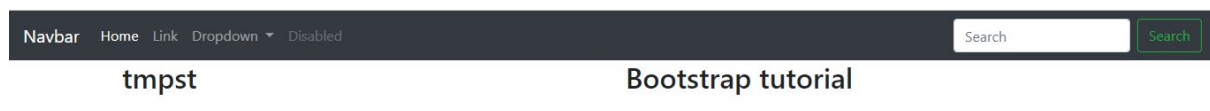
```

Index.cshtml  [X]
1  <div class="row">
2    <div class="col-6">
3      <h2>tmpst</h2>
4    </div>
5    <div class="col-6">
6      <h2>Bootstrap tutorial</h2>
7    </div>
8  </div>

```

The image on the left shows two explicitly defined columns which both contain a basic header 2 element

The result of this code is shown below:



Conclusion

That is the code required to implement Bootstrap 4's navbar and utilise the column system. The complexity of columns and rows are up to the developer

Common Issues:

Normally, implementing Bootstrap 4 does not cause too many problems, however, some of the most common issues are detailed below:

- Referencing script in the wrong order - As mentioned in the navbar tutorial, the JQuery, popper and Bootstrap scripts have to be referenced in order
- Not putting column within row - As mentioned in the grid tutorial, columns have to be direct children of rows or they will not work as intended
- Exceeding column limit - As mentioned in the grid tutorial, the columns of a row can only add up to a maximum of 12 before they are knocked down into a new row. If an element is not displayed properly it may be because you have added too many columns to a row

Outcomes

You should now be able to:

- Create a Bootstrap 4 Navbar
- Create a page using the Bootstrap 4 grid system

References

Although we hope this tutorial has been all the help you need, here are some useful links that may be of use:

Useful Resources:

- Bootstrap Navbar Docs - <https://getbootstrap.com/docs/4.0/components/navbar/>
- Bootstrap and Popper Download - <https://getbootstrap.com/docs/4.0/getting-started/download/>
- JQuery Download - <https://jquery.com/>
- Navbar Colour Schemes - <https://getbootstrap.com/docs/4.0/components/navbar/#color-schemes>
- Bootstrap 4 tutorial - <https://www.w3schools.com/bootstrap4/>