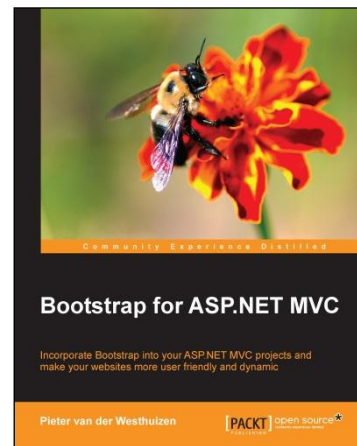


# Bootstrap for ASP.NET MVC

Pieter van der Westhuizen



## Chapter No.1 "Getting Started with ASP.NET MVC and Bootstrap"

## In this package, you will find:

The author's biography

A preview chapter from the book, Chapter no.1 "Getting Started with ASP.NET MVC and Bootstrap"

A synopsis of the book's content

Information on where to buy this book

## About the Author

**Pieter van der Westhuizen** is a freelance software and web developer specializing in ASP.NET MVC, web technologies, and MS Office development. He started his career in web development using classic ASP, Visual InterDev, HoTMetaL, and FrontPage.

Pieter has over 15 years of experience in the IT industry and is also one of the people fortunate enough to have his hobby become his full-time profession. He is also a technology evangelist for Add-in Express ([www.add-in-express.com](http://www.add-in-express.com)), which focuses on tools for Microsoft Office integration.

This is Pieter's first foray into book writing although he has been blogging since 2007 on his personal blog at [www.mythicalmanmoth.com](http://www.mythicalmanmoth.com) and on the Add-in Express blog since 2010. He lives with his wife and Harrier (the dog, not the bird or Jump Jet) in Pretoria, South Africa.

**For More Information:**

[www.packtpub.com/web-development/bootstrap-aspnet-mvc](http://www.packtpub.com/web-development/bootstrap-aspnet-mvc)

# Bootstrap for ASP.NET MVC

Twitter Bootstrap, simply known as Bootstrap, is the leading open source CSS/HTML and JavaScript framework on the Internet. Shortly after its launch, it became the most popular project on GitHub. It became so popular that Microsoft announced at their Build 2013 conference that all the web app project templates in Visual Studio 2013 will use Twitter Bootstrap by default.

One of the main reasons why Bootstrap is so prevalent is that it allows developers, many of whom are notoriously bad at user interface design, to build aesthetically pleasant-looking sites with a relatively small amount of effort. Bootstrap also offers a rich ecosystem of free and commercial templates, third-party components, tools, and an active and helpful community.

Using CSS frameworks and Bootstrap in particular with ASP.NET MVC is a natural fit. Bootstrap takes care of the typography, form layouts, and user interface components, and allows the developer to focus on what they are good at, that is, writing code. This aspect is particularly valuable for smaller development companies that do not necessarily have an in-house designer. Bootstrap Version 3 introduced a mobile-first approach, meaning all sites built with Bootstrap will be automatically responsive and optimized to be displayed on devices with smaller screens.

## What This Book Covers

*Bootstrap for ASP.NET MVC* walks you through the process of creating a fully functioning ASP.NET MVC website, using Bootstrap for its layout and user interface.

*Chapter 1, Getting Started with ASP.NET MVC and Bootstrap*, focuses on getting started with Bootstrap, from where to get the files, how to include them in your project, and takes a closer look at the default ASP.NET MVC project template. We'll also look at the benefits of bundling and minification of CSS and JavaScript.

*Chapter 2, Using Bootstrap CSS and HTML Elements*, examines the various Bootstrap CSS and HTML elements, how to include them in your ASP.NET MVC project, and how to configure and use their various options.

*Chapter 3, Using Bootstrap Components*, will be building on what we've learned in Chapter 2, Using Bootstrap CSS and HTML Elements. This chapter scrutinizes the different components such as navigation, alerts, progress bars, button groups, and badges.

*Chapter 4, Using Bootstrap JavaScript Plugins*, illustrates the use of Bootstrap's JavaScript plugins. We will be experimenting with modal dialogs, contextual dropdowns, tooltips, buttons, and UI components such as accordion and carousel.

**For More Information:**

[www.packtpub.com/web-development/bootstrap-aspnet-mvc](http://www.packtpub.com/web-development/bootstrap-aspnet-mvc)

*Chapter 5, Creating ASP.NET MVC Bootstrap Helpers*, guides you through the process of reducing the amount of HTML needed to generate Bootstrap elements by creating ASP.NET MVC helper methods and classes.

*Chapter 6, Creating T4 Templates to Scaffold Bootstrap Views*, moves on to the more advanced topic of creating T4 templates in order to generate Bootstrap-themed scaffolded views.

*Chapter 7, Converting a Bootstrap HTML Template into a Usable ASP.NET MVC Project*, shows how we can convert an open source HTML template and make it ready to be used with ASP.NET MVC.

*Chapter 8, Using the jQuery DataTables Plugin with Bootstrap*, demonstrates how to use the powerful jQuery DataTables plugin with Bootstrap and ASP.NET in order to show tabular data.

*Chapter 9, Making Things Easier with the TwitterBootstrapMVC Library*, examines the TwitterBootstrapMVC library, which contains a host of prebuilt HTML helpers to make the inclusion of Bootstrap components in ASP.NET MVC easier.

*Appendix, Bootstrap Resources*, provides a list of free Bootstrap resources, themes, and tools.

**For More Information:**

[www.packtpub.com/web-development/bootstrap-aspnet-mvc](http://www.packtpub.com/web-development/bootstrap-aspnet-mvc)

# 1

## Getting Started with ASP.NET MVC and Bootstrap

As developers, we can find it difficult to create great-looking user interfaces from scratch when using HTML and CSS. This is especially hard when developers have years of developing Windows Forms applications experience. Microsoft introduced Web Forms to abstract the complexities of building websites away for Windows Forms developers and to ease the switch from Windows Forms to the Web, this in turn made it very hard for Web Forms developers to switch to ASP.NET MVC and even harder for Windows Forms developers.

Twitter Bootstrap is a set of stylized components, plugins, and a layout grid that takes care of the heavy lifting. Microsoft included Bootstrap in all ASP.NET MVC project templates since 2013. In the sample project, we'll start by creating a new ASP.NET MVC project either by using the standard Visual Studio MVC project template or by starting with an empty MVC project and adding the necessary files as we need them.

In this chapter, we will cover the following topics:

- The files included in the Bootstrap distribution
- How to create an ASP.NET MVC site using the standard Visual Studio project template and Bootstrap
- How to create an empty ASP.NET MVC site and add the Bootstrap files manually
- How to create a `Layout` file that references the Bootstrap files
- Adding Bootstrap files using NuGet
- Improving site performance with bundling and minification

**For More Information:**

[www.packtpub.com/web-development/bootstrap-aspnet-mvc](http://www.packtpub.com/web-development/bootstrap-aspnet-mvc)

## The Bootstrap distribution

Before we can get started with Bootstrap, we first need to download its source files. At the time of writing this book, Bootstrap was at Version 3.1.1. You can download the latest version from <http://getbootstrap.com>.

The zip archive contains the following three folders:

- `css`
- `fonts`
- `js`

### Bootstrap style sheets (the `css` folder)

Do not be alarmed with the amount of files inside the `css` folder. This folder contains four `.css` files and two `.map` files. We would only need to include the `bootstrap.css` file in our project for the Bootstrap styles to be applied to our pages. The `bootstrap.min.css` file is simply a minified version of the aforementioned file.

The `.map` files can be ignored for the project we'll be creating. These files are used as a type of debug symbol (similar to the `.pdb` files in Visual Studio), which allow developers to live edit their preprocessor source files—something which is beyond the scope of this book.

### Bootstrap fonts (the `fonts` folder)

Bootstrap uses **Font Awesome** to display various icons and glyphs in Bootstrap sites. Font Awesome was designed specifically for Bootstrap and the `fonts` folder contains the following four different formats of the font files:

- Embedded OpenType (`glyphicons-halflings-regular.eot`)
- Scalable Vector Graphics (`glyphicons-halflings-regular.svg`)
- TrueType font (`glyphicons-halflings-regular.ttf`)
- Web Open Font Format (`glyphicons-halflings-regular.woff`)

It is a good idea to include all these files in your web project as this will enable your site to display the fonts correctly in different browsers.



The EOT font format is required for Internet Explorer 9 and newer. TTF is the traditional old font format and WOFF is a compressed form of TTF fonts. If you only need to support Internet Explorer 8 and later, iOS 4 and higher, as well as Android, you will only need to include the WOFF font.

## Bootstrap JavaScript files (the js folder)

The `js` folder contains two files. All the Bootstrap plugins are contained in the `bootstrap.js` file. The `bootstrap.min.js` file is simply a minified version of the aforementioned file. Before including the file in your project, make sure that you have a reference to the jQuery library because all Bootstrap plugins require jQuery.

The default project template automatically adds the jQuery library to your project and creates a bundle for it. The jQuery bundle will be included in your pages if you have the following line of code inside your view:

```
@Scripts.Render("~/bundles/jquery")
```



### Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at <http://www.packtpub.com>. If you purchased this book elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you.

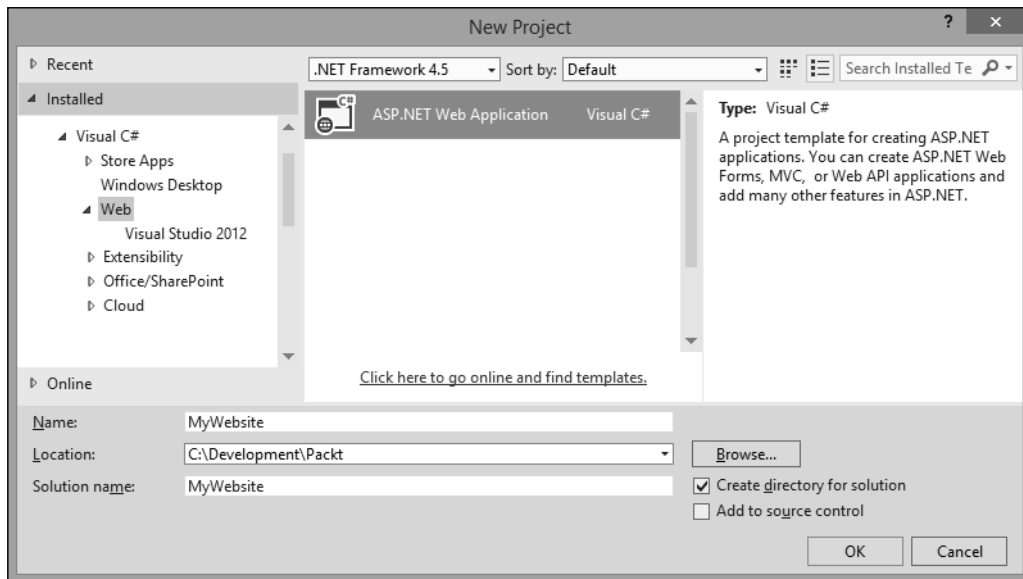
## The Bootstrap folder structure

The unzipped folder structure for Bootstrap will look something like the following screenshot:

```
bootstrap/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.min.css
│   ├── bootstrap-theme.css
│   └── bootstrap-theme.min.css
├── js/
│   ├── bootstrap.js
│   └── bootstrap.min.js
└── fonts/
    ├── glyphs-halflings-regular.eot
    ├── glyphs-halflings-regular.svg
    ├── glyphs-halflings-regular.ttf
    └── glyphs-halflings-regular.woff
```

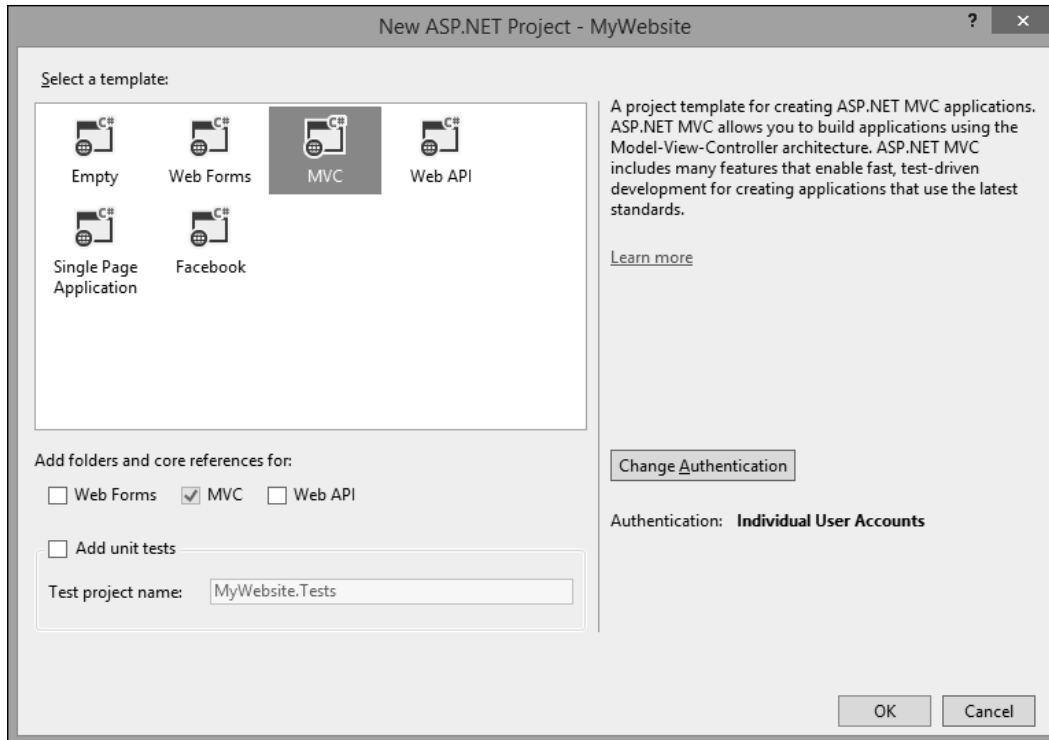
## Using Bootstrap with a site created with the standard Visual Studio project template

From Visual Studio 2013, when creating an ASP.NET project, you only have one project template to choose from, that is, the ASP.NET Web Application project template, as shown in the following screenshot:





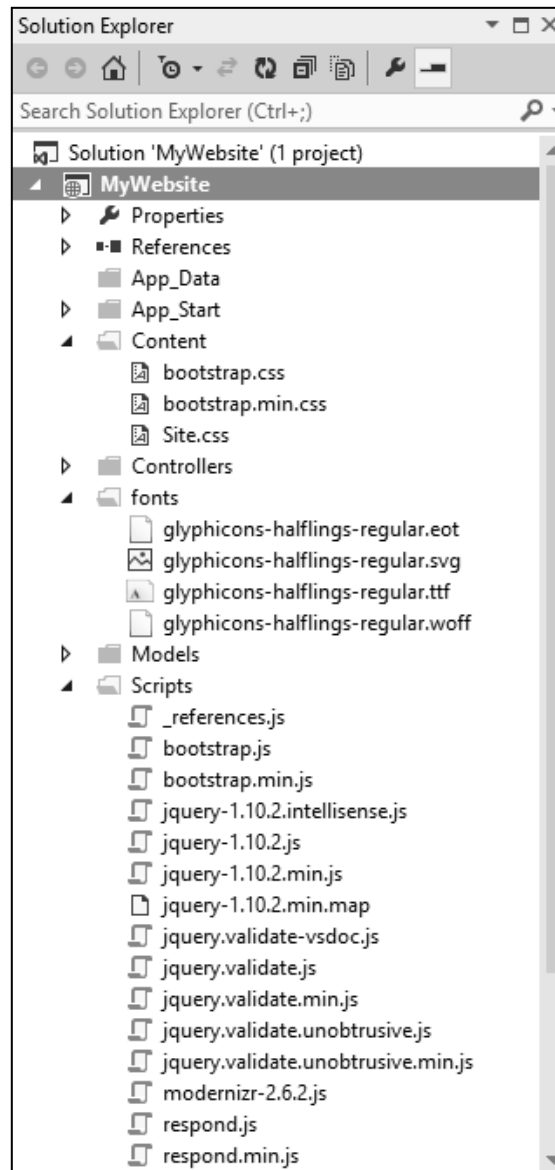
In the **New ASP.NET Project** dialog, you have a choice to select the type of ASP.NET web application you would like to create. To create an ASP.NET MVC web app that uses Bootstrap for its styling and layout, select the **MVC** template. You'll notice that the **MVC** checkbox is automatically selected, as shown in the following screenshot:



Click on the **OK** button to finish the creation of the MVC project in Visual Studio. You'll notice that the project template automatically adds a number of NuGet packages to your project, including the Bootstrap NuGet package.

## Examining the default MVC project layout

The default project template adds all the necessary Bootstrap files we discussed earlier, although it does not use the same folder naming convention as the default Bootstrap distribution. The default project layout will look similar to the following screenshot:



## The Content folder

The Content folder contains both the `bootstrap.css` and `bootstrap.min.css` files as well as a style sheet called `Site.css`. This file is used to apply any additional styling on top of the default styles provided by Bootstrap, and it is also used to specify the styles to use for the jQuery validation plugin required by ASP.NET MVC for form validation. For example, the following CSS highlights any input element with a reddish color and draws a border around the element if the validation for that field failed:

```
.field-validation-error {
    color: #b94a48;
}

.field-validation-valid {
    display: none;
}

input.input-validation-error {
    border: 1px solid #b94a48;
}

input[type="checkbox"].input-validation-error {
    border: 0 none;
}

.validation-summary-errors {
    color: #b94a48;
}

.validation-summary-valid {
    display: none;
}
```

## The fonts folder

The fonts folder contains the Glyphicon font in all the necessary formats.

## The Scripts folder

The `Scripts` folder contains a number of scripts. Most notably for this book, it contains the `bootstrap.js` and `bootstrap.min.js` JavaScript files. The default ASP.NET MVC project template also adds both minified and normal files for the following JavaScript libraries and plugins:

- jQuery
- jQuery validation plugin
- jQuery and jQuery validation support library for unobtrusive validation
- Modernizr
- Respond JS

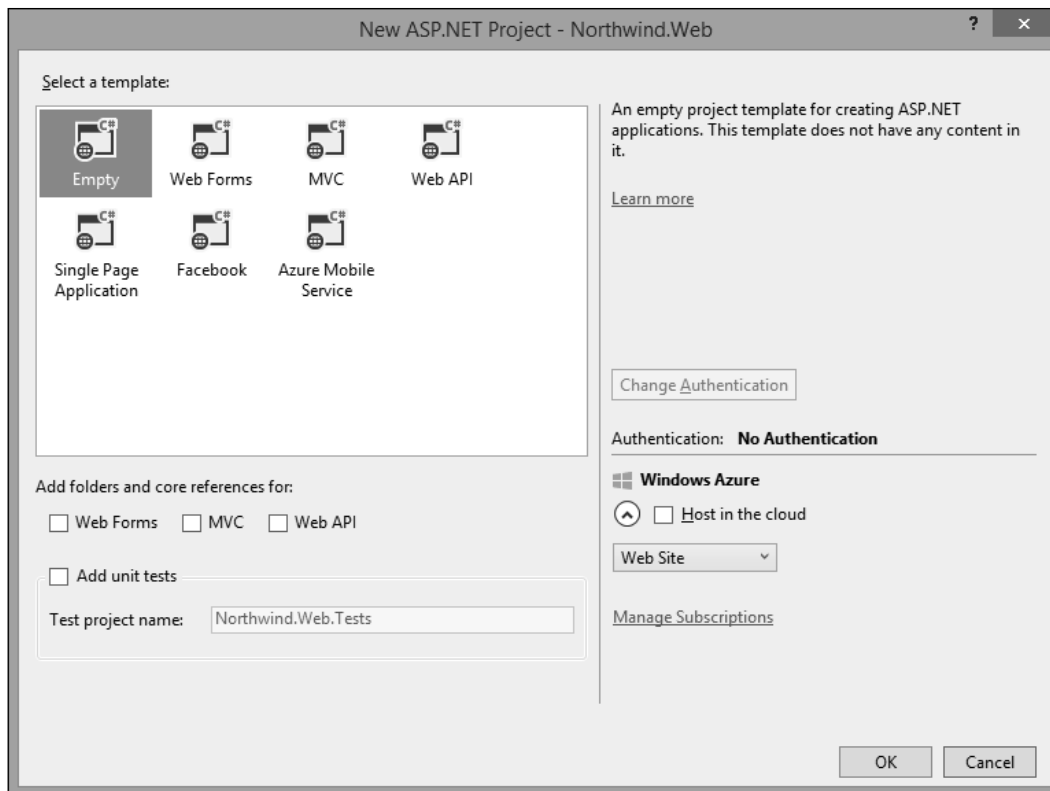
Visual Studio enables IntelliSense for jQuery, Bootstrap, and Modernizr as well as responds by adding the `_reference.js` file to the `Scripts` folder. This is a very useful feature when working with JavaScript and well worth using when working with the Bootstrap components.

Most of these libraries and files are beyond the scope of this book, but we will touch on some of them as we progress.

## Creating an empty ASP.NET MVC site and adding Bootstrap manually

The default project layout is a good start for any ASP.NET MVC project, but for the sample project we'll be building throughout this book, we'll create an empty ASP.NET MVC site and add the necessary files manually. This is done by performing the following steps:

1. Start by creating a new ASP.NET web application project in Visual Studio and name the project `Northwind.Web`.
2. This time, select the **Empty** template in the **New ASP.NET Project** dialog and make sure the **MVC** checkbox is selected, as shown in the following screenshot:



3. An empty project layout will be created for you and you'll notice that we do not have the `Content`, `Fonts`, or `Scripts` folder – we'll add them ourselves!

## Adding the Bootstrap style sheets

To add the Bootstrap style sheet files to your project, complete the following steps:

1. Create a new folder by right-clicking on the new project's name inside Visual Studio's **Solution Explorer** and navigating to **Add | New Folder**, name the new folder `css`.
2. Next, right-click on the newly created `css` folder and navigate to **Add | Existing Item...** from the context menu.
3. Browse to the folder in which you've extracted the Bootstrap distribution files and select the `bootstrap.css` file that you can locate in the `css` folder.

## Adding the Bootstrap fonts

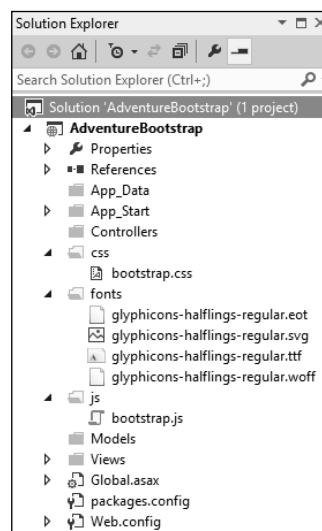
Add the required Bootstrap fonts by performing the following steps:

1. As with the style sheets, create a new folder called `fonts`.
2. Next, browse to the location to where you've extracted the Bootstrap download and add all the files from the `fonts` folder to your `fonts` folder in Visual Studio.
3. There should be four files in total, each named `glyphicons-halflings-regular` but with the following different file extensions:
  - `.eot`
  - `.svg`
  - `.ttf`
  - `.woff`

## Adding the Bootstrap JavaScript files

The final Bootstrap file we'll need is `bootstrap.js`. To add it, perform the following steps:

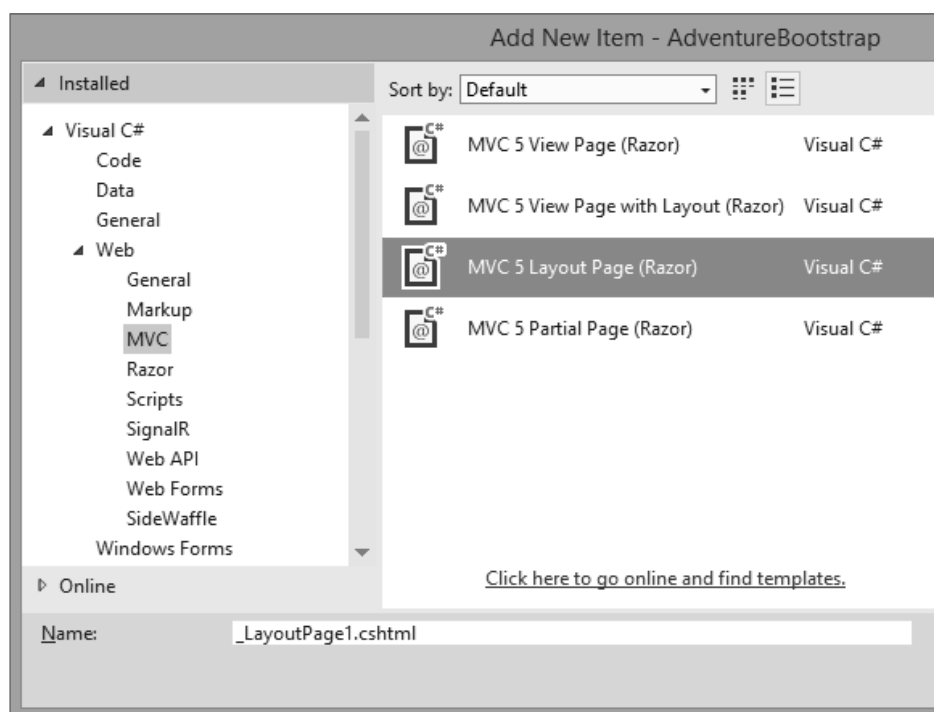
1. Before adding the `bootstrap.js` file to your Visual Studio project, create a new folder called `js`.
2. Add the `bootstrap.js` file to this folder.
3. Once completed, the project layout should look similar to the following screenshot in the Visual Studio's **Solution Explorer**:



## Creating the site Layout file

To maintain a persistent look across our site's pages, we'll use a `Layout` file. This layout file will use the basic Bootstrap HTML template at first and we'll build onto it as we progress throughout the book. To create this file, complete the following steps:

1. Create a new `Layout` file by right-clicking on the `Views` folder in the Visual Studio's **Solution Explorer** and navigate to **Add | New Folder**. Name the new folder `Shared`.
2. Next, right-click on the `Shared` folder and navigate to **Add | New Item....** Select the **MVC 5 Layout Page (Razor)** item template, name the new item `_Layout.cshtml` and click on **Add**, as shown in the following screenshot:



3. After the new file is added to your project, open it and replace its contents with the following HTML markup:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
```

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width,
initial-scale=1">
<title>@ViewBag.Title</title>
<!-- Bootstrap -->
<link href="@Url.Content("~/css/bootstrap.css")"
rel="stylesheet">
<!-- [if lt IE 9]>
  <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/
html5shiv.js"></script>
  <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/
respond.min.js"></script>
<![endif]-->
</head>
<body>
  @RenderBody()
  <script src="https://ajax.googleapis.com/ajax/libs/
jquery/1.11.0/jquery.min.js"></script>
  <script src="@Url.Content("~/js/bootstrap.js")"></script>
</body>
</html>
```

In the preceding markup, we set the viewport's width property to the device's width and the initial-scale value to 1. This will cause our site to adapt to the screen size of the device the user is viewing it from.

Next, we reference the Bootstrap style sheet by using the `Url.Content` helper method. This helper method converts a virtual or relative path to an absolute path, making sure that when the web page is opened, the style sheet will be loaded correctly.

We then check if the browser accessing the site is Internet Explorer 9 or earlier; if it is, we include the `HTML5Shiv` workaround that enables styling of HTML5 elements in Internet Explorer Version 9 and earlier. We also include the version of the `Respond JS` library suitable for versions of IE9 and earlier.

Just before the closing the `<body>` tag of the file, we include the jQuery library and the bootstrap JavaScript library.



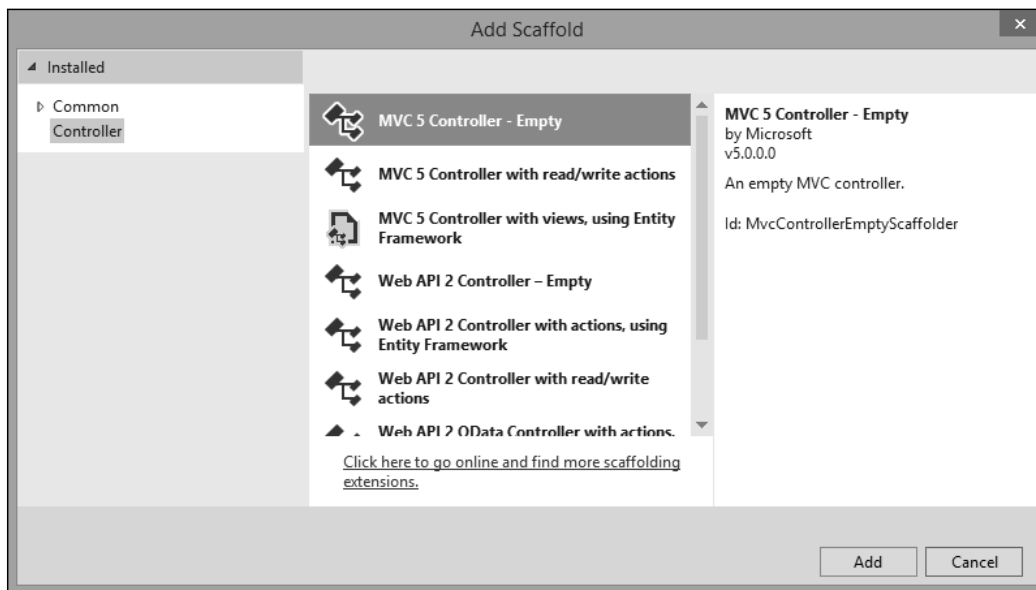
Note that the `HTML5Shiv` workaround, `Respond JS`, and `jQuery` files are loaded from a **Content Delivery Network (CDN)**. This is a good approach to use when referencing to most of the widely used JavaScript libraries. This should allow your site to load faster if the user has already visited a site, which uses the same library from the same CDN, because the library will be cached in their browser.



## Creating a home controller with a Bootstrap-themed view

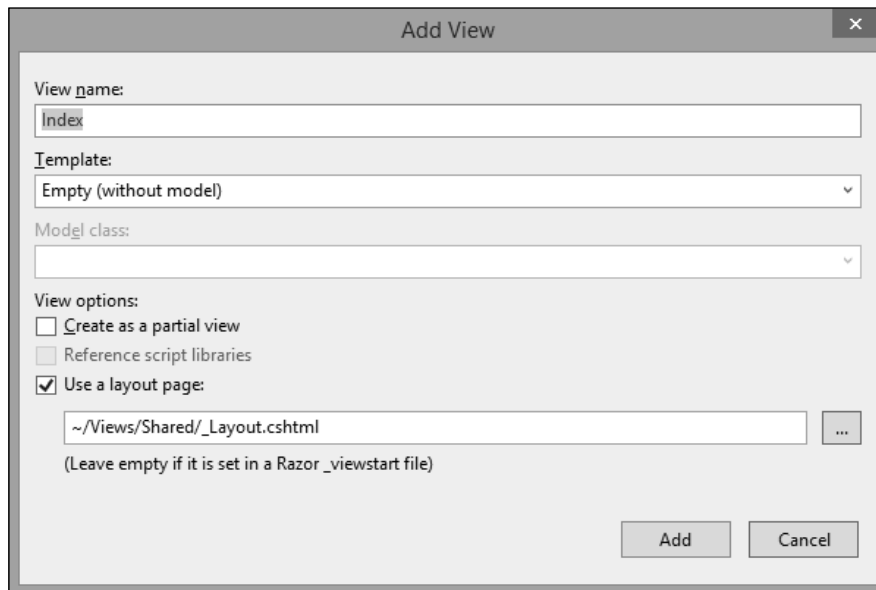
At the moment, the site we've created will return an error message that states that the requested resource cannot be found, when we run the project. We first need to add a new home controller and associate a view with its default action in order to avoid any errors. To add a new controller, perform the following steps:

1. Right-click on the **Controller** folder in the **Solution Explorer** section and navigate to **Add | Controller...**
2. In the **Add Scaffold** dialog, select the **MVC 5 Controller - Empty** item, as shown in the following screenshot:



3. When prompted, in the **Add Controller** dialog, enter `HomeController` as the new controller name and click on **Add**.
4. After the controller has been added, right-click inside the empty **Index** method and select **Add View**. The method is considered empty if it has only the one `return View()` statement in it.

5. In the **Add View** dialog, leave all the fields at their default values and select the layout page we've added earlier. Click on **Add** to continue, as shown in the following screenshot:



6. An empty view will be generated and you'll notice the `ViewBag.Title` property as well as the layout page to use for this view have been added at the top of the view. Consider the following example:

```
@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

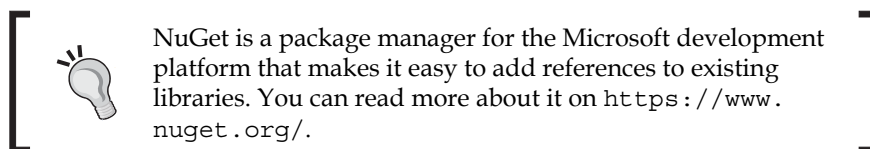
There will also be a single `<h2>` element in the view that you can leave as is. If you run your project, you should see the view displayed with the default Bootstrap styling applied to the `<h2>` element.

## Adding Bootstrap files using NuGet

So far we've created two ASP.NET MVC projects that use the Bootstrap frontend framework. The first included the Bootstrap assets by default because we created it with the standard ASP.NET MVC Visual Studio project template. The second, we created an empty ASP.NET MVC project and added the Bootstrap files manually.

NuGet is a package manager for the .NET framework and can be used to automatically add files and references to your Visual Studio projects. A Bootstrap package exists on the NuGet gallery site, which enables you to automatically add the Bootstrap assets to your project.

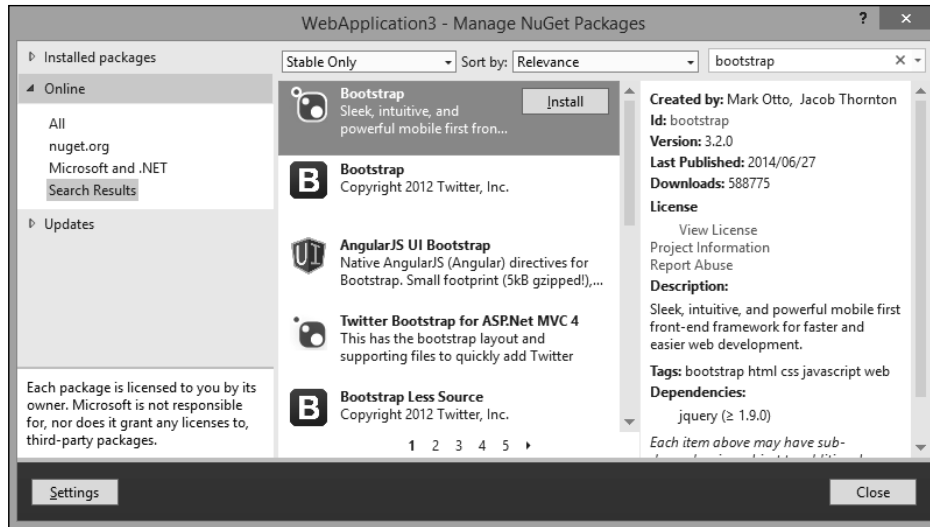
Bear in mind that though the Bootstrap NuGet package assumes that you want your Bootstrap files in the `Content` and `Scripts` folders and will create the folders and files as such, it will also automatically check whether you have the jQuery library referenced and if not, add it by design.



## Adding the Bootstrap NuGet package using the dialog

One option for adding the Bootstrap NuGet package to your project is to use the **Manage NuGet Packages** dialog box. To access the **Manage NuGet Packages** dialog and add the Bootstrap NuGet package, perform the following steps:

1. Right-click on the **References** node in the **Solution Explorer** section and select **Manage NuGet Packages**. Click on the **Online** tab and type **bootstrap** in the search box and press **Enter**, as shown in the following screenshot:

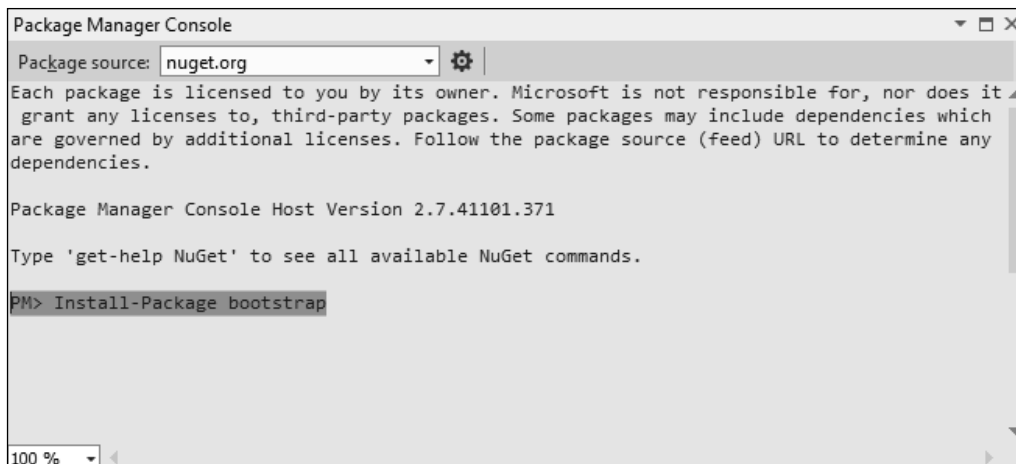


2. Click on the **Install** button to add the Bootstrap files to your project.

## Adding the Bootstrap NuGet package using the Package Manager Console

The second method of adding NuGet packages to your Visual Studio project is via the **Package Manager Console** and completing the following steps:

1. Inside Visual Studio, from the **Tools** menu, navigate to **Library Package Manager | Package Manager Console**. This will open the **Package Manager Console** window.
2. To install the Bootstrap NuGet packages, type `Install-Package bootstrap`, as shown in the following screenshot:



```
Package Manager Console
Package source: nuget.org
Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it
grant any licenses to, third-party packages. Some packages may include dependencies which
are governed by additional licenses. Follow the package source (feed) URL to determine any
dependencies.

Package Manager Console Host Version 2.7.41101.371

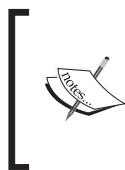
Type 'get-help NuGet' to see all available NuGet commands.

PM> Install-Package bootstrap
```

3. This will create the `Content`, `fonts`, and `Scripts` folders and add the necessary Bootstrap files to each.

## Improving your site performance with bundling and minification

Bundling and minification is a feature in ASP.NET that allows you to increase the speed at which your site loads. This is accomplished by limiting the number of requests to CSS and JavaScript files your site needs to make by combining these types of files into one large file and removing all unnecessary characters, such as comments, white spaces, and new line characters from the files.



Most modern browsers have a limit of six concurrent connections per hostname. This means that if you reference more than six CSS or JavaScript files on a page, the browser will only download six files at a time and queue the rest. Limiting the number of CSS and JavaScript files is always a good idea.

## Adding bundling to your Bootstrap project

Because we created an empty ASP.NET MVC site, the necessary reference for adding bundling was not automatically added to our project. Luckily, this can easily be added by using the NuGet **Package Manager Console** as follows:

1. Open the **Package Manager Console** window and enter the following command:  
**install-package Microsoft.AspNet.Web.Optimization**
2. This will install the `Microsoft.AspNet.Web.Optimization` NuGet package as well as all the packages it has dependencies on. These dependencies are as follows:
  - `Microsoft.Web.Infrastructure`
  - `WebGrease`
  - `Antlr`
  - `Newtonsoft.Json`
3. After the NuGet packages have been installed successfully, create a new static class called `BundleConfig` inside the `App_Start` folder.
4. Inside this class, we'll create a new static method called `RegisterBundles`, which accepts a parameter called `bundles`, whose type is a `BundleCollection` object. The code for the class is as follows:

```
public class BundleConfig
{
    public static void RegisterBundles(BundleCollection bundles)
    {
        bundles.Add(new ScriptBundle("~/bootstrap/js").Include(
            "~/js/bootstrap.js",
            "~/js/site.js"));

        bundles.Add(new StyleBundle("~/bootstrap/css").Include(
            "~/css/bootstrap.css",
            "~/css/site.css"));
    }
}
```

Bundles come in the following two types:

- `ScriptBundle`
- `StyleBundle`

The `StyleBundle` object is used to add style sheets to a bundle and the `ScriptBundle` object is used to add JavaScript files. The `Add` method of the `BundleCollection` object accepts both types of objects. The `StyleBundle` and `ScriptBundle` objects accept a string parameter, which specifies the virtual path of the files, and you should therefore, use the `Include` method to specify the path to the files you would like to include in the bundle.



Never include files with `.min` in their names, for example, `bootstrap.min.css` or `bootstrap.min.js` in bundles. The compiler will ignore these files as they are already minified.

## Including bundles in your ASP.NET layout

To include the bundles we created earlier in our `Layout` file, perform the following steps:

1. Open the `_Layout.cshtml` file in the `Shared` folder and change its markup to reflect the following (changes are highlighted):

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width,
  initial-scale=1">
  <title>@ViewBag.Title</title>
  @Styles.Render("~/bootstrap/css")

  <!-- [if lt IE 9]>
    <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/
    html5shiv.js"></script>
    <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/
    respond.min.js"></script>
  <![endif]-->
</head>
<body>
  @RenderBody()
  <script src="https://ajax.googleapis.com/ajax/libs/
  jquery/1.11.0/jquery.min.js"></script>
  @Scripts.Render("~/bootstrap/js")
</body>
```

2. If the Visual Studio HTML editor indicates that it cannot find the `Scripts` or `Styles` objects (that is indicated by the words being highlighted), this means a reference is missing from the `Web.config` file inside the `Views` folder.
3. To fix this, add a reference to `System.Web.Optimization` to the list of namespaces in this `Web.config` file. Consider the following example:

```
<namespaces>
  <add namespace="System.Web.Mvc" />
  <add namespace="System.Web.Mvc.Ajax" />
  <add namespace="System.Web.Mvc.Html" />
  <add namespace="System.Web.Routing" />
  <add namespace="AdventureBootstrap" />
  <add namespace="System.Web.Optimization" />
</namespaces>
```

## Testing bundling and minification

In order to enable bundling and minification, open the `Web.config` file inside the root of your project and change the `debug` attribute of the `compilation` element to `true` as follows:

```
<system.web>
  <compilation debug="true" targetFramework="4.5" />
  <httpRuntime targetFramework="4.5" />
</system.web>
```

The same can be achieved by setting the `EnableOptimizations` property of the `BundleTable` object to `true`. This statement can either be added to the `Global.asax` file's `Application_Start` method or to the `RegisterBundles` method of the `BundleConfig` class as follows:

```
BundleTable.EnableOptimizations = true;
```

After you've added references to the bundles inside your `Layout` file and set either the `debug` attribute or the `EnableOptimizations` property to `true`, build and run your project. Once the site is open in a browser, view its source. You should see a markup similar to the following:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
```



```

<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width,
initial-scale=1">
<title>Index</title>
<link href="/bootstrap/css?v=Tmmo-oSKW9MFwr7qyt2LfyMD1tap2GokH7
z1W2bhfgY1" rel="stylesheet"/>
<!-- [if lt IE 9] >
  <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/
html5shiv.js"></script>
  <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/
respond.min.js"></script>
<![endif]-->
</head>
<body>
<h2>Index</h2>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.0/
jquery.min.js"></script>
<script src="/bootstrap/js?v=raqa-So7giLQpXYq5LQiW8D-
yNoxOAJewB8VXtgFHfE1"></script>
</body>
</html>

```

Note that the highlighted lines contain the relative paths we've specified in our bundles and when clicking on, for example, the `/bootstrap/js` link, the browser will open a minified version of the Bootstrap JavaScript file. You can see it is minified because most white spaces and line breaks have been removed. The code will also be a combination of the Bootstrap JavaScript file as well as any other JavaScript files, which we might have added to the bundle.

## Summary

In this chapter, you've learned what is inside the Bootstrap download and how to include these files in your own ASP.NET MVC projects. We've also covered the various techniques of how to include these files and how to increase the performance of your site using bundling and minification.

In the next chapter, we'll dive into the inner working of Bootstrap's CSS and HTML elements and how to use them to design the layout of your site.

## Where to buy this book

You can buy Bootstrap for ASP.NET MVC from the Packt Publishing website:

<https://www.packtpub.com/web-development/bootstrap-aspnet-mvc>.

Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our [shipping policy](#).

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.



[www.PacktPub.com](http://www.PacktPub.com)

**For More Information:**

[www.packtpub.com/web-development/bootstrap-aspnet-mvc](http://www.packtpub.com/web-development/bootstrap-aspnet-mvc)