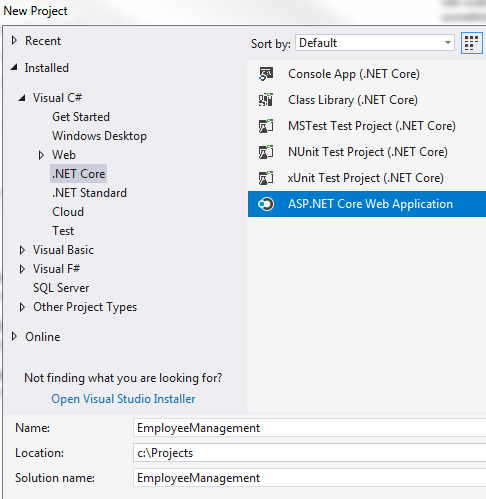
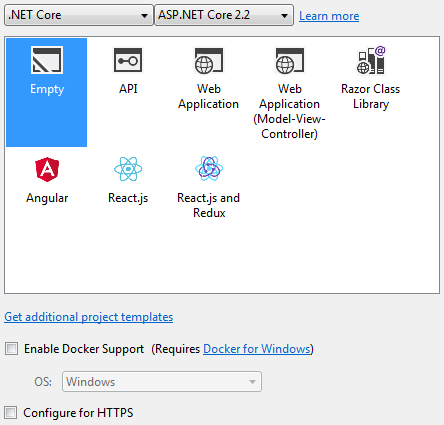
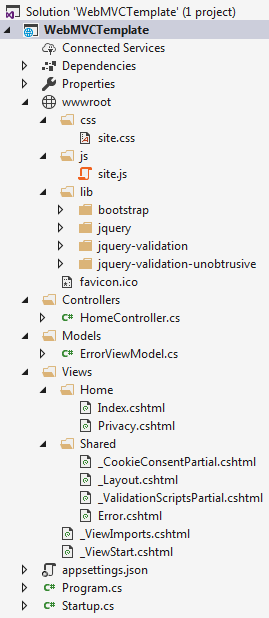
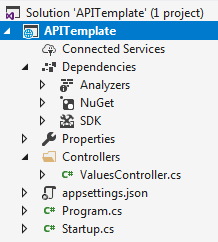
**Creating asp.net core web application**

**we will discuss**

* Creating a new ASP.NET Core Project in Visual Studio
* The different project templates that are available and what they do

**Creating a new ASP.NET Core Project in Visual Studio**   
  
**Step 1 :** To create a new asp.net core project in Visual Studio 2017   
  
**Step 2 :** In Visual Studio click on **File - New - Project.**   
  
  
**Step 3 :** In the **"New Project"**dialog, expand **"Installed"**. Then expand **"Visual C#" and**select **.NET Core**  
  
   
  
**Step 4 :**In the middle pane, you will find all the installed project templates. For example, you will find templates for creating .NET Console Application, Class Library, Unit Testing Projects (with MSTest, NUnit or xUnit Frameworks), ASP.NET Core Web Application.   
  
**Step 5 :**We want to create "ASP.NET Core Web Application". So, select **"ASP.NET Core Web Application"**   
  
**Step 6 :**In the **"Name"** textbox, type a name for your project. I am going to name it **EmployeeManagement**. This asp.net core web application that we are going to build as part of this project, allows us to Create, Read, Update and Delete employees.   
  
**Step 7 :**Also, specify a location where you want this project to be created. I am going to place my project in **Projects**folder in **C:\** drive.   
  
**Step 8 :**Finally click **OK**.   
  
**Step 9 :** On the next screen, select **ASP.NET Core 2.2**(The latest .NET Core SDK available as of this recording)   
  
   
  
**Step 10 :** Uncheck **"Configure for HTTPS"** checkbox   
  
This screen shows the different project templates available for creating ASP.NET Core application.   
  
**Empty :**The "Empty" template as the name implies does not contain any content. This is the template that we will use and set up everything manually from scratch so we clearly understand how the different pieces fit together.   
  
**Web Application (Model-View-Controller) :**This template includes everything we need to create an **ASP.NET Core MVC Web Application.**  
  
The MVC Web Application template creates **Models**, **Views**and **Controllers**folders and adds web specific things like **CSS**, **JavaScript files**, **Layout files**and other resources needed for a web site.   
  
The screenshot below shows a project that I have created using the **MVC Web Application template**. Notice we have Models, Views and Controllers folders. In the Views folder we have sample view files and layout file that represent the user interface of a web application. We also have JavaScript and CSS files that are typically required by a web application.   
  
   
  
**API :**This template includes everything we need to create an **ASP.NET Core RESTful HTTP service**  
  
All the web site specific things like JavaScript files, CSS files, view files, layout files are not required for an API, as it does do not have an user interface. The data that a Web API exposes is usually consumed by other applications.    
  
So an API template, just creates the **"Controllers"**folder. It does not create **Models**and **Views**folders as they are not required for an API. The screenshot below shows a project that I have created using the **API template**. Notice we only have the **Controllers**folder. We do not have all those JavaScript, CSS and Layout files that are not required for a RESTful API.   
  
   
  
**Web Application :**This template uses the new **Razor Pages framework**for building web applications. With Razor pages, coding page-focused scenarios is easier and more productive. We typically use this approach when we do not want the full complexity of ASP.NET MVC. We can think of it as a slimmer version of the MVC framework. We will discuss Razor Pages in detail in our upcoming videos.   
  
**Razor Class Library (RCL) :** As the name implies we use this template to create a **reusable Razor Class Library project**. This project contains all the reusable user interface pieces like **data models, page models, controllers, pages, Razor view**and **View components**. This Razor Class Library (RCL) project can then be reused in multiple applications. Any application that uses the RCL can override the views and pages it contains.   
  
**Angular, React.js, React.js and Redux :**These three templates allow us to create asp.net core web application in combination either with Angular, React or React and Redux.