MONASH University

FIT5032 - Internet Applications Development

Week 01 - HouseKeeping & Introduction to Visual Studio 2017

Last updated: 6th August 2018
Author: Jian Liew

Housekeeping

Before we begin, it is highly recommended for you to use your own personal computer for this subject. If you are planning to use the Monash computers, it is highly recommended to save all your work properly. Towards, the end of the semester, there will be a portfolio submission where you need to showcase all the work you have done so far. To prevent difficulties during this portfolio submission, it is suggested that you keep all your work neat and organised based on a **week by week structure.**

Recommended Structure

- **FIT5032-S2-2018**
  - **W1**
  - **W2**
  - **W3**
  - **….**

If you are using the Monash Lab computers, it is highly recommended keeping all directory names short. Please remember that for the Monash Lab computers, you only have full permission at certain locations.

Setting Up Your Own Development Environment

It is also important to understand the importance of your own development environment. It is important that you set up your development environment correctly. In this subject, there will be not only weekly demonstrations, there will also be weekly interviews regarding your progress in this subject. It is ultimately your responsibility to keep everything in a working condition.

Please refer to the document titled - Setting Up Your Development Environment available on Moodle for more details and general question & answers related to this matter.

Tutorial Structure

The tutorials in this unit are designed to be of a **self-paced and self-taught structure.** So, the aim of your tutor is to aid you in your learning experience. He or she will not go through all the materials that are covered in the labs and will **not do the exercise one by one as a class**. If help is needed, you can either post on the Moodle forums or you can email your tutor asking for clarification. This is slightly different in comparison to other units, where your tutor will lead the discussions. **Please take note of this, you will only be provided help if you make it known that you need help.**

Please note, that we will be using **Microsoft Visual Studio Professional 2017. Version 15.6.7.** All screenshots in this guide will be based on that version. If you are using a different version like the Community edition, it should be somewhat similar. However, if you are using a different major version like Visual Studio 2013, it is highly recommended upgrading it.

Before you proceed to do this tutorial, it is highly recommended to set up your development environment first and read the lecture slide on the background materials. If you do not, you may find some parts of this tutorial confusing.

Objectives

Estimated Time To Complete - 2 hours
Upon the completion of this tutorial, you will gain a basic understanding of

- What Visual Studio is
- The difference between a Project and a Solution
- The various HTML elements in the starter project
- Understand the difference between a cshtml file and a html file.
- The Benefits of Using a layout file
- A general idea of what is going to happen in the coming weeks

DoubtFire Submission

- T1.1 A screenshot of your computer running the default website.
- T1.2 An explanation of how the MVC works in your own words. (No more than a single PDF page with a diagrammatic representation)
- T1.3 [Updated 09/08/2018] Documentation detailing how an IDE like Visual Studio aids in the developer during their work.

General Reminder

The main objective of this lab is to give an introduction to not only what the MVC architecture is but also an introduction the various elements in a Visual Studio project.

Microsoft Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, web sites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both **native code and managed code.** Often times, developers would use the term manage and unmanaged code.

We will be using Microsoft Visual Studio for this subject.

This document aims to give a brief introduction to Microsoft Visual Studio.

Please remember to use the correct version, if you do not, there is a high chance some screen shots may differ significantly. If so, consult your tutor for help.

If you are not sure, please ask your tutor regarding this matter. Do keep in mind, there might be slight differences between the Community, Professional and Enterprise editions. However, for what we will be doing the differences does not matter. However, it would be a good idea to use the same version so that all the screenshots provided will be the same.

According to popular opinion, Visual Studio is often times considered one of the best IDE in the market.

Before we start, there is a need to understand the difference between a project and a solution in Visual Studio.

Project and Solution

Project

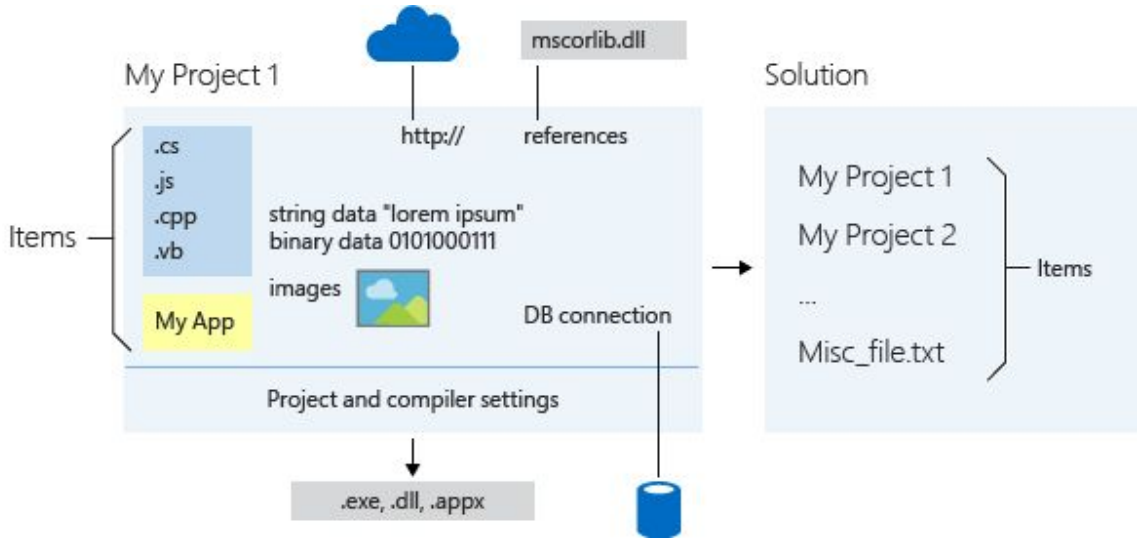
When you create an app, website, plug-in, etc. in Visual Studio, you start with a project. In a logical sense, **a project contains all the source code files, icons, images, data files, etc. that are compiled into an executable, library, or website.** A project also contains compiler settings and other configuration files that might be needed by various services or components that your program communicates with.

A project is defined in an XML file with an extension such as .vbproj, .csproj, or .vcxproj. This file contains a virtual folder hierarchy, and paths to all the items in the project. It also contains the build settings.

Solution

A project is contained within a solution. A solution contains one or more related projects, along with build information, Visual Studio window settings, and any miscellaneous files that aren't associated with a particular project. A solution is described by a text file (extension .sln) with its own unique format; it is not intended to be edited by hand.

A solution has an associated .suo file that stores settings, preferences, and configuration information for each user that has worked on the project.



Project and Solution

In short, when you start, you create a project, when you hit run, it builds into a solution. A solution may consist of one or more projects. It is quite common for a solution to consist of multiple projects for larger scale applications.

Features of an Enterprise Level IDE

Visual Studio is an example of an **Enterprise Level Integrated Development Environment (IDE)**. It has a lot of features that are very useful for developers. Previously, in other units, you might have used NetBeans or IntelliJ. Those are also examples of Enterprise Level IDE. BlueJ is not an enterprise level IDE, it however it is a good teaching tool to introduce object oriented programming to new learners.

Useful Hotkeys

It is a good idea to practice using these shortcuts as over time, it will increase your productivity significantly. Please note that some of these shortcut keys may not work due the language settings on your computer. It is advised that you understand the use of these shortcut keys and how they can speed up development process. Here are some of them which are commonly used. (If you are interested ask your tutor more about this)

Keyboard Shortcut	Feature
CTRL + K, CTRL + C	Comment Section
CTRL + K, CTRL + U	Uncomment Section
CTRL + SPACE	Toggle Intellisense (Code Completion Feature)
CTRL + -	Navigate Forward
CTRL + SHIFT + -	Navigate Backward
CTRL + K, CTRL + D	Reformat (Coding Standards)
CTRL + ,	Go to All
F5	Run Solution

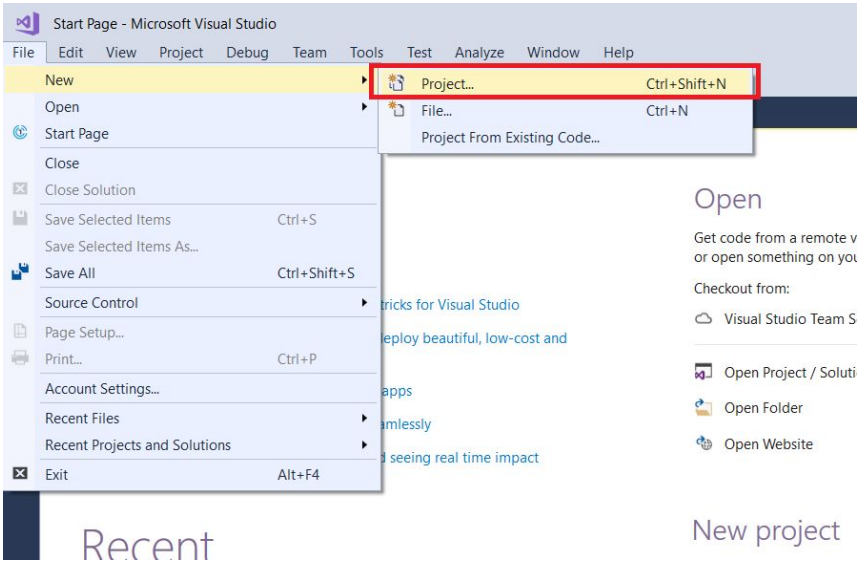
IntelliSense

IntelliSense is a **code-completion aid** that includes a number of features: List Members, Parameter Info, Quick Info, and Complete Word. These features help you to learn more about the code you're using, keep track of the parameters you're typing, and add calls to properties and methods with only a few keystrokes.

This is a very important feature if you plan to be a developer. Please note that it defaults to CTRL + SPACE, if you are using this hotkey this switch language, this keyboard shortcut may not work for you. It is recommended to change it to something else. Most programmers tend to use this shortcut key a lot.

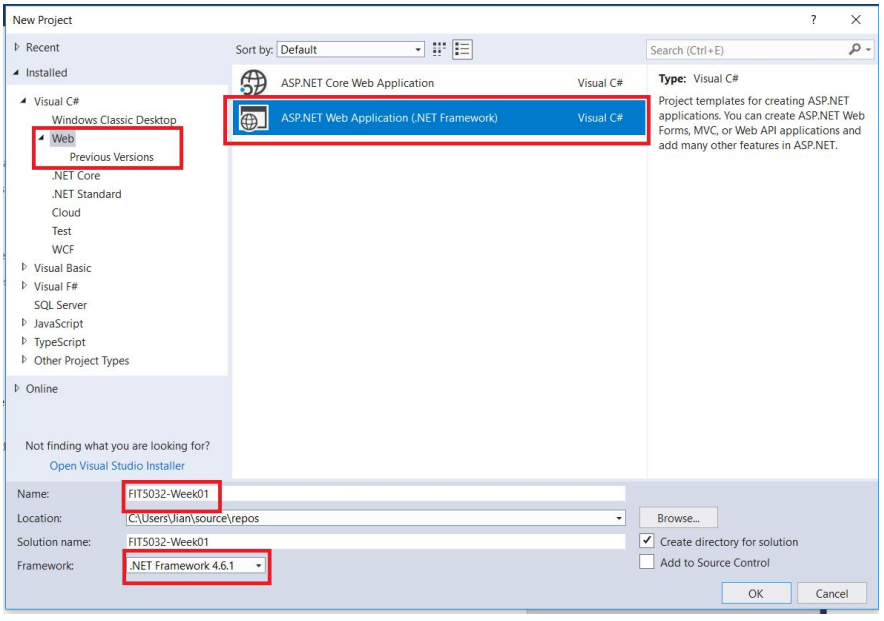
Getting Started with a Visual Studio .NET MVC Project

Step 1 - Starting a new Project



In order to create a new Project on MS Visual Studio, you will need to go to

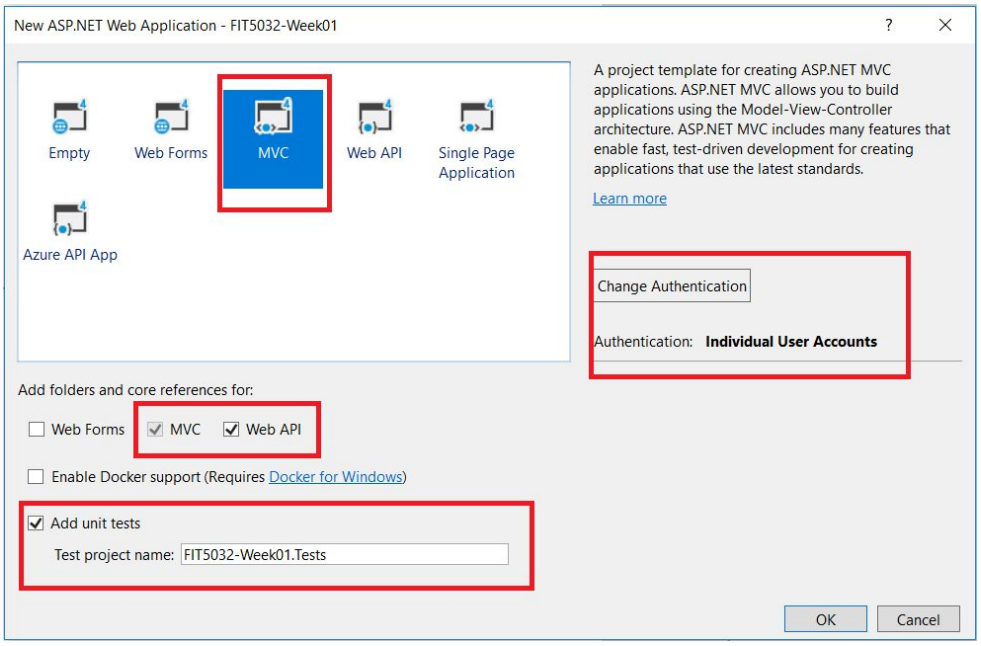
Step 2 - File → New → Project (Ctrl + Shift + N)



We will be using **FIT5032_Week01** as the name. It is highly recommended not to have spaces.

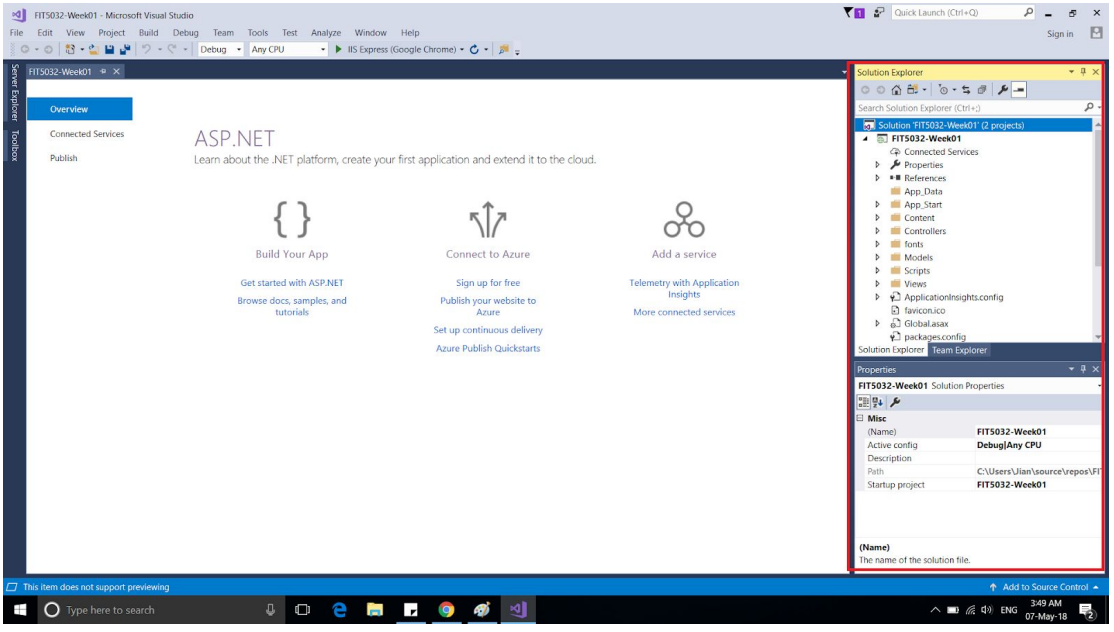
If you do not have the latest version installed select the most recent version. It is up to you if you want to use dashes (-) or (_) underscores. In the screenshot, I used an a dash, as long as it is consistent, everything should be working as intended. It considered to be better to use _ in this situation. (Using the - is often referred to as kebab-casing)

If you are using the Monash computers, please save everything in a place you have full permission to. For example, it would be either your Documents or your Desktop directory. If you do not know where this is, your tutor will show you where this is. (Locating the Desktop directory can be difficult in the labs due to mappings done by eSolutions)

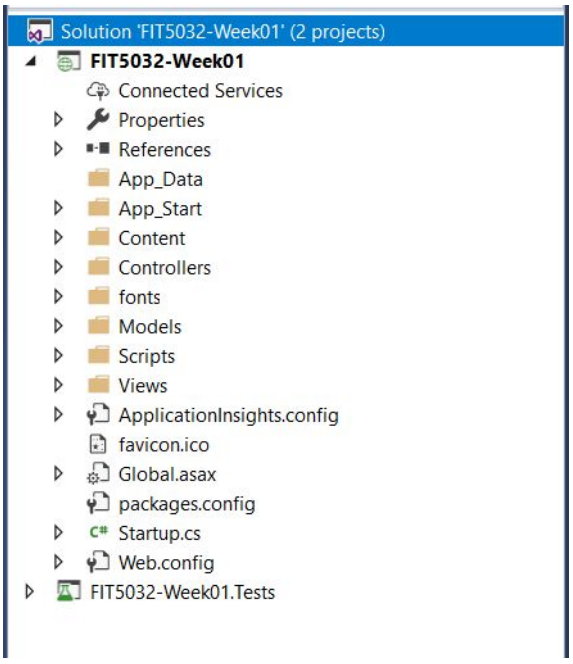


Here we will create the project with the following features.

1. MVC project (Week 1)
2. Authentication Type - Individual User Accounts (More details on Week 7)
3. MVC enabled (Whole semester)
4. Web API enabled (More details on Week 10)
5. Unit Test enabled (More details on Week 11)



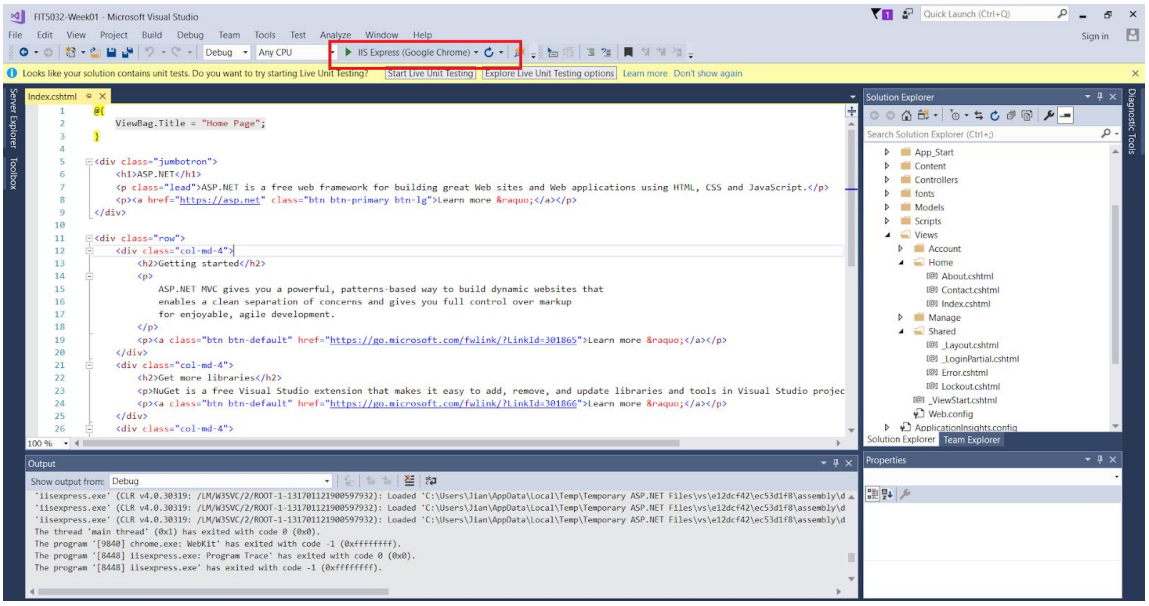
Upon the creation of the project, you will notice that the project is now populated on the right-hand side in the window tab called the solution explorer.



Solution Explorer

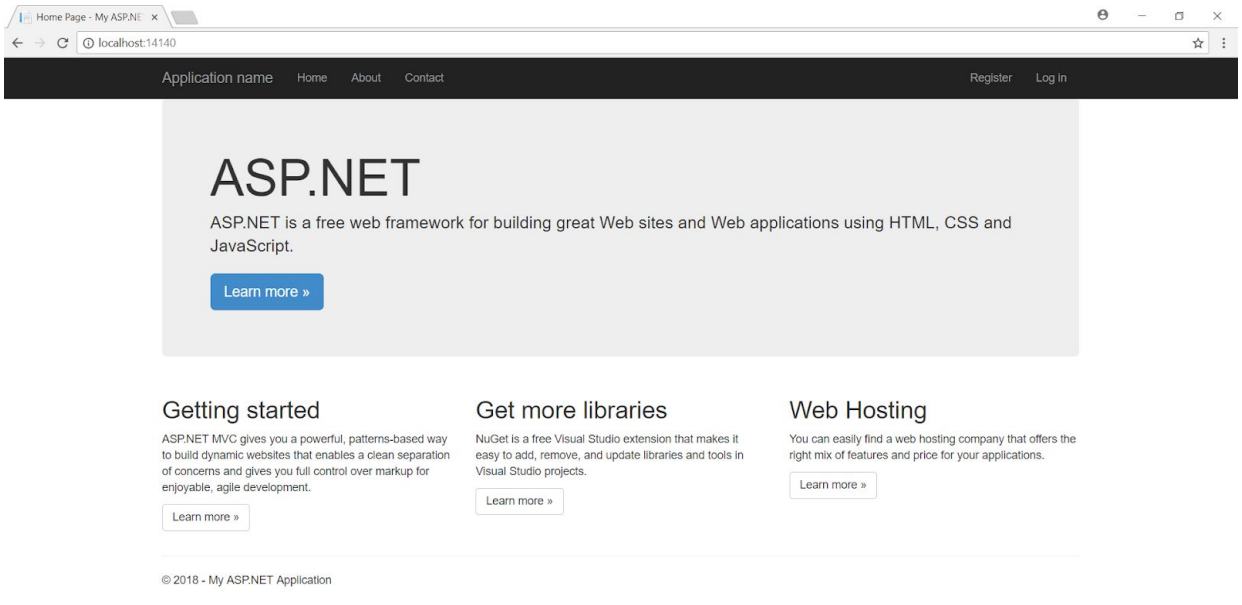
Before we move further, it is important to understand what the various directories and files are.

Directory or File	Purpose
App_Data	A storage point for file based data store. (Database file)
App_Start	Folder containing core configuration files
Content	Stores all files and resources to the solution
Controllers	The controllers of the MVC.
font	The font here is part of the Bootstrap requirements.
Models	The Models of the MVC. (More details Week 4)
Scripts	This contains the needed JavaScripts. (More details Week 5)
Views	The Views of the MVC
ApplicationInsights.config	This is to see the performance of the application (Week 11)
favicon.ico	This is the tiny icon that appears at the top of the tab (On your website).
packages.config	This is used in some project types to maintain the list of packages referenced by the project.
Web.config	XML containing configuration data. (This is a very important file)



How to Run the Solution

You can now click on the run button. Here, I have selected to use **Google Chrome** as the browser of my choice. It is highly recommended to use Google Chrome as it is currently the most widely used browser. If the computer you are using does not have **Google Chrome** installed, you can choose to use Internet Explorer or Edge. If you want to, you can also opt to use FireFox or the Developer Edition of FireFox. After the site has loaded, you can click around and see how it works.



Solution running on Google Chrome

Models, Views and Controllers

Before we go on further, we will examine how the project is laid out. We will first look at the View.

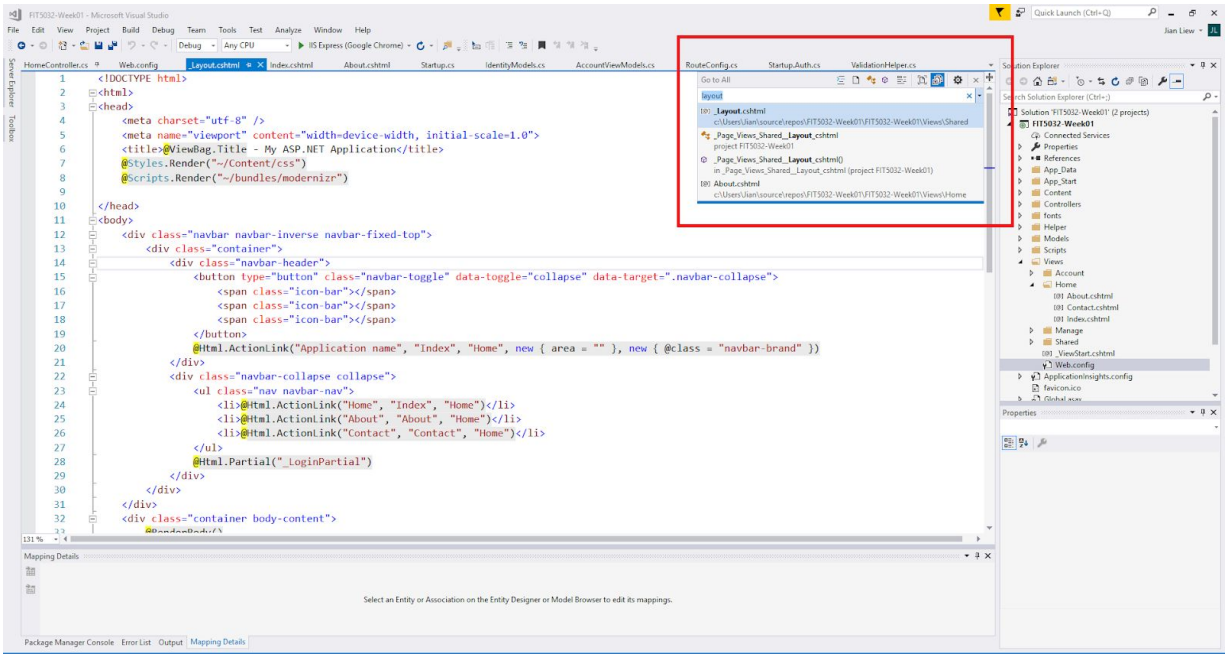
Views

The Views are normally just HTML files.

Most web applications have a common layout that provides the user a consistent experience as the user navigates from page to page. Layouts help to reduce the amount of duplicate codes in the view, helping in the **Don't Repeat Yourself (DRY) principle**. In our project, this file can be found under the **Shared directory, with the filename _Layout.cshtml.** You can think of this as the concept of Master pages, meaning that for all pages there are things that are always there for example the header (navigation bar) and footer (contents at the end of the page). Thus, it would be pointless for this portion of the code to repeat itself multiple times.

There is a difference between a .cshtml vs a html. The biggest difference is that a .cshtml would be able to utilise the Razor engine and have C# codes embedded within it. (Notice the @ in front)

Take a look at _Layout.cshtml for example. If you are unable to find this file, it is time to try out one of the hotkeys that is available on Visual Studio as CTRL + , (Control plus comma) and type in "layout".

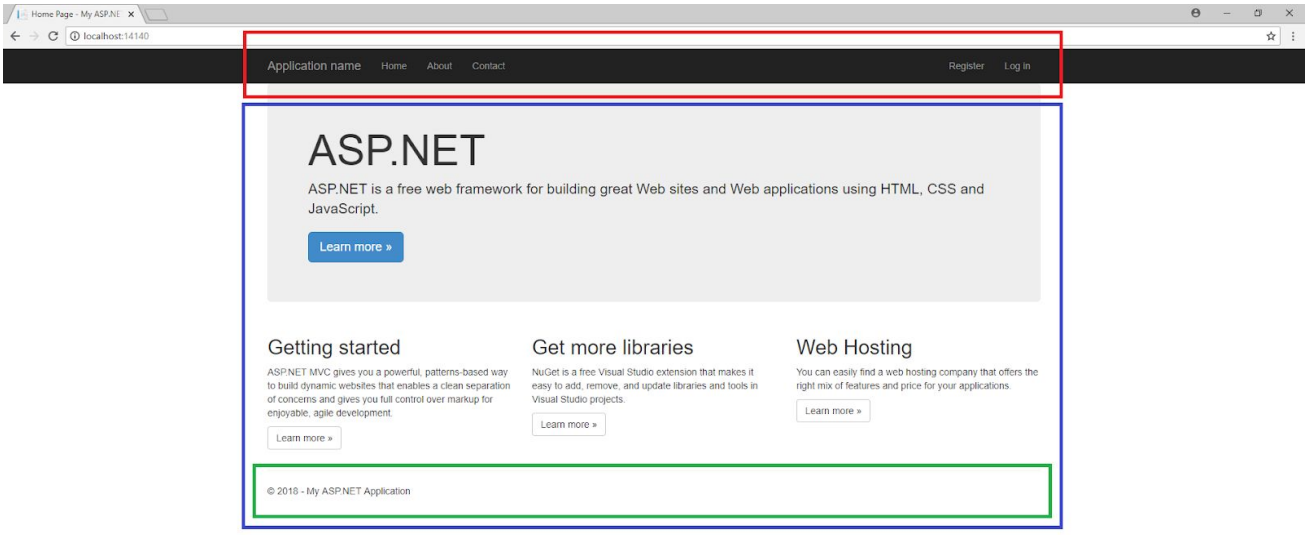


```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <div class="navbar 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="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("Application name", "Index", "Home", new { area = ""
}, new { @class = "navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("About", "About", "Home")</li>
                    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                </ul>
                @Html.Partial("_LoginPartial")
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
        </footer>
    </div>

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>
```

Shared Layout Explained

The shared layout can be broken down into 3 major items. They are

1. Head (This section is not visible to the user browser)
2. Navigation Bar (THe navigation bar)
3. Content Body (You will notice that the footer in this example is inside the content body)



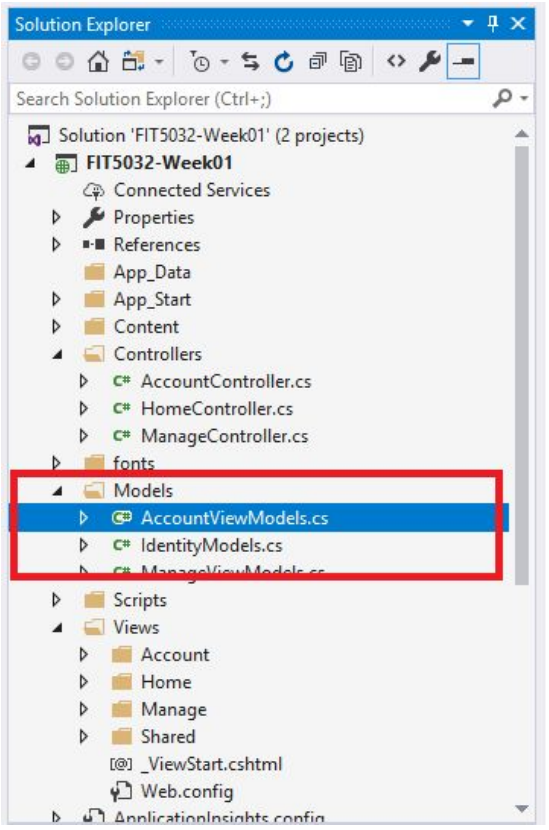
Based on that, it can be seen that the red box represents the navigation bar while, the blue box is actually body-contents and inside of the body-contents is the footer.

Notice that by default, the design leaves the footer inside of the body contents.

You will also realise there will be inclusion of HTML classes. For example, "`div class=navbar ...`" is actually using the Bootstrap framework. This will be further explained in the coming weeks. Essentially, it will create the markup needed for the CSS. If the Bootstrap framework is not present, the navigation bar will not be present at its current form.

There will be a more detailed explanation regarding the views next week.

Models

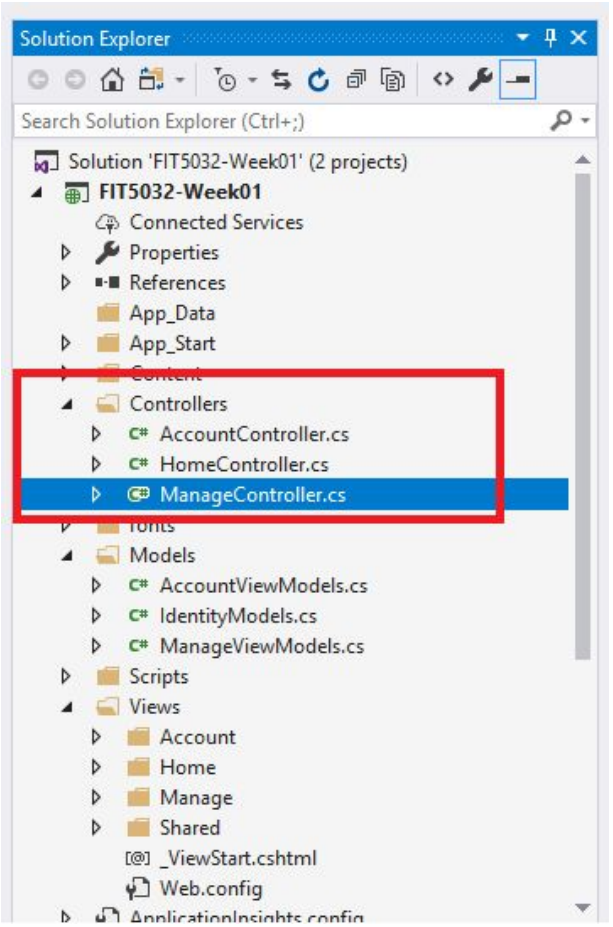


By default, if you created the project with the Authentication method Individual User Accounts, these models will be created automatically for you.

```
namespace FIT5032_Week01.Models
{
    public class ExternalLoginConfirmationViewModel
    {
        [Required]
        [Display(Name = "Email")]
        public string Email { get; set; }
    }

    ...
}
```

Models will play a variety of purpose in our application. We will discuss more in detail regarding this in subsequent weeks throughout the semester. The reason, for this is because we not only need an understanding of not only another programming language but also the purpose of different models. The example here is that of a ViewModel.

Controllers



Now, we shall take a look at the controller classes. By default there will be 3 controller classes created using the mentioned way of creating the project.

For now, now everything might seem very confusing but it would be clear in subsequent weeks. For example, in the HomeController, it can be seen that there will be Views return from every action. The syntax of C# will be explained in subsequent weeks as well.

```
namespace FIT5032_Week01.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult About()
```

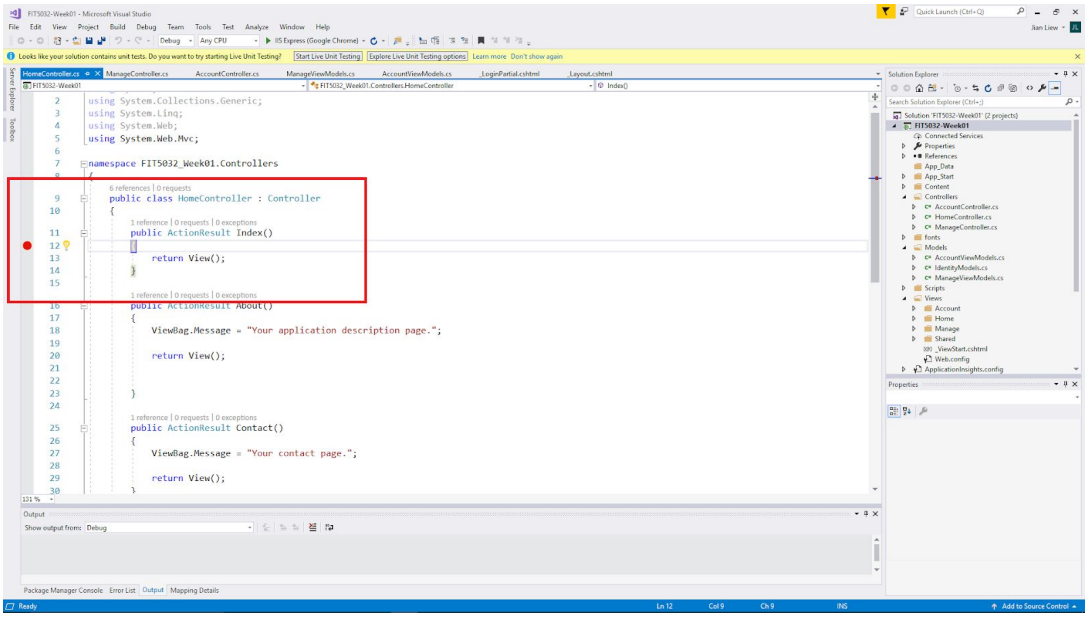
```
        {
            ViewBag.Message = "Your application description page.";

            return View();
        }

        public ActionResult Contact()
        {
            ViewBag.Message = "Your contact page.";

            return View();
        }
    }
}
```

Here, you can play around with the Message returned in the **ViewBag** and see how it passes the messages from the control classes to the view classes.

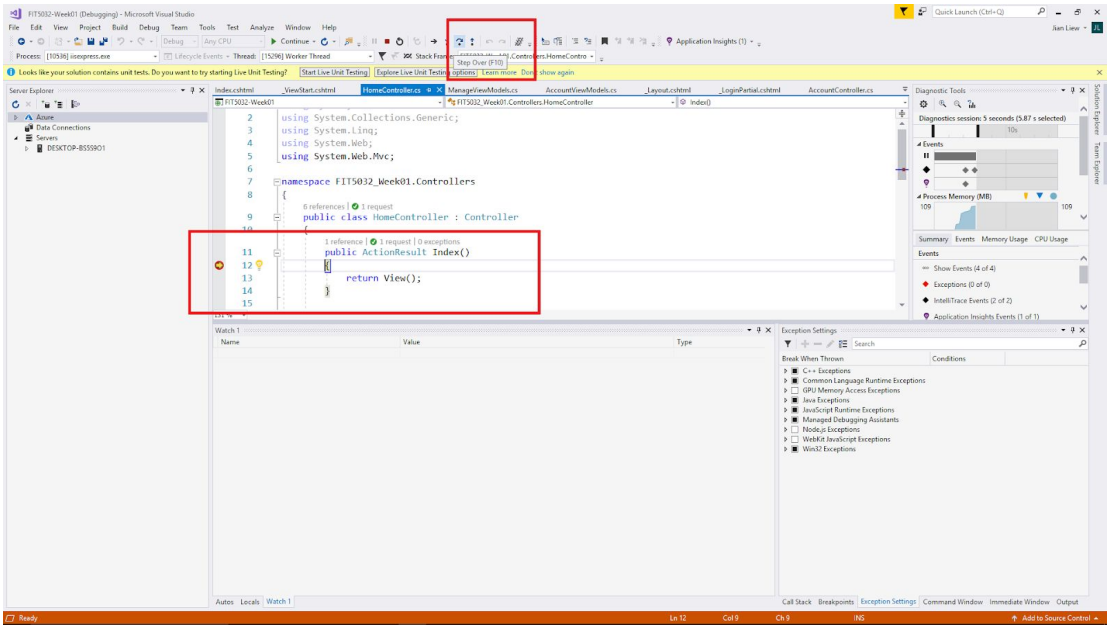
MVC In Action

If you want to see the MVC in action, you can follow the following steps.

Place a breakpoint at Line 12 on HomeController.cs, then hit run. Using breakpoints is a very important debugging skill. (The breakpoint is indicated by the red circle at the left hand side of the line numbers). You should start running your application. (By default it runs in debug mode)



After that, you can hit the "Step Over" button and see how the flow works. Notice that how the first request before you hit the "Index" page will be determined by the Control classes.



DoubtFire Submission

- T1.1 A screenshot of your computer running the created project. (You do not need to make any modifications to the project).
- T1.2 Documentation detailing how the MVC works, when you click on a button. For example, if you click on the Contact button on the main index of the website, what happens?
- T1.3 Documentation detailing how an IDE like Visual Studio aids in the developer during their work.

Summary

Upon the completion of this lab, you will gain an understanding of

- What is Visual Studio and how it is helpful for development
- A general understanding of how the labs will be structured and what is expected
- The amount of work that is expected
- A basic understanding of what the MVC is.

Document History & Change Log

Date	Summary
10th July 2018	Removed materials regarding code snippet and automatic code generation.
13th July 2018	Moved code snippet to Week 5 Explain regarding usage of dashes and underscores
6th August 2018	Fixed typo. Changed from "represent" to "present"
12th October 2018	Added various minor fixes to prevent confusion. ("Such as starting the project")