MONASH University

FIT5032 - Internet Applications Development

Week 05B - Working with Maps (MapBox JS API)

Last updated: 3rd November 2018
Author: Jian Liew

Introduction

There is no need to complete this tutorial. It functions as a supplementary material to showcase how to use a simple JavaScript API (MapBox) to show locations based on latitude and longitude on the map.

Working with maps is a very common use case. The objective of this supplementary material to give a quick introduction on how to use a simple JavaScript map API (Mapbox) in this case. The reason we are not using Google Maps API is because they have changed their payment module slightly and it might cause some difficulties for you if we plan to use it. There are other mapping APIs out there, however the concept to use them should be similar to this.

Objectives

Estimated Time To Complete - 2 hours

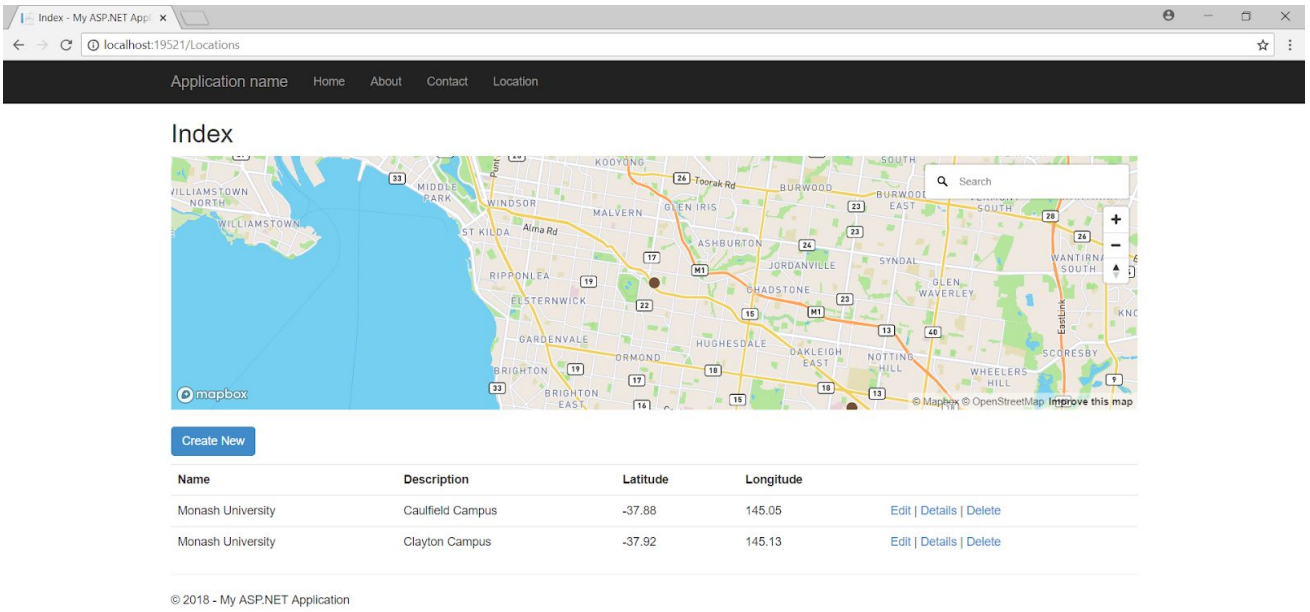
Upon the completion of this tutorial, you will gain a basic understanding of

- How to use MapBox to display points in the Map based on values stored in your database.

DoubtFire Submission

- None

Upon the completion of this tutorial. The end result of your web application would look like the following.



This is a really basic example, on how you can use the Mapbox API. Notice that there are two points on the map (indicated by the brown circle, Monash Caulfield & Monash Clayton)

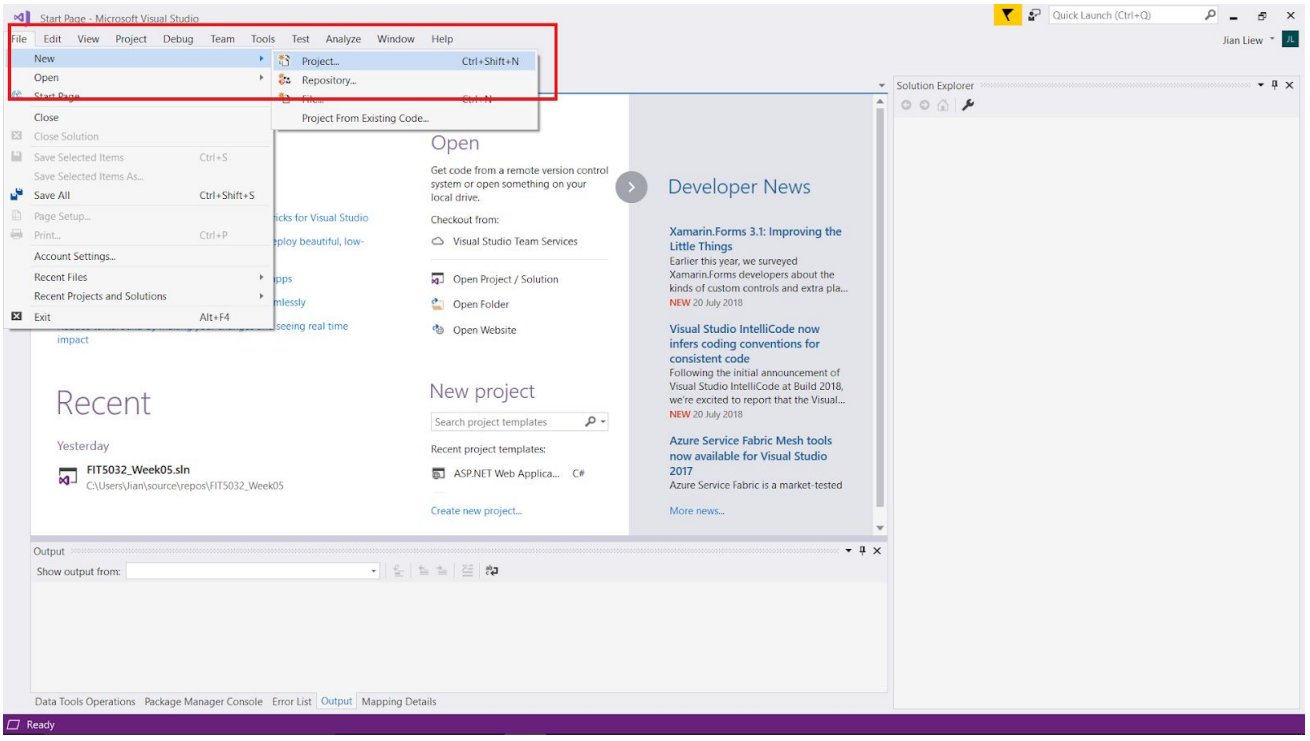
Certain features that are present in this tutorial are

1. It uses a database first approach to create a single table with latitude and longitude.
2. It uses the scaffolding feature to create both the controllers and views.
3. It uses the Mapbox API to introduce map controls. This includes the "Search" feature and another mapcontrol for Navigation Control (This is the zoom in and out button present on the map)

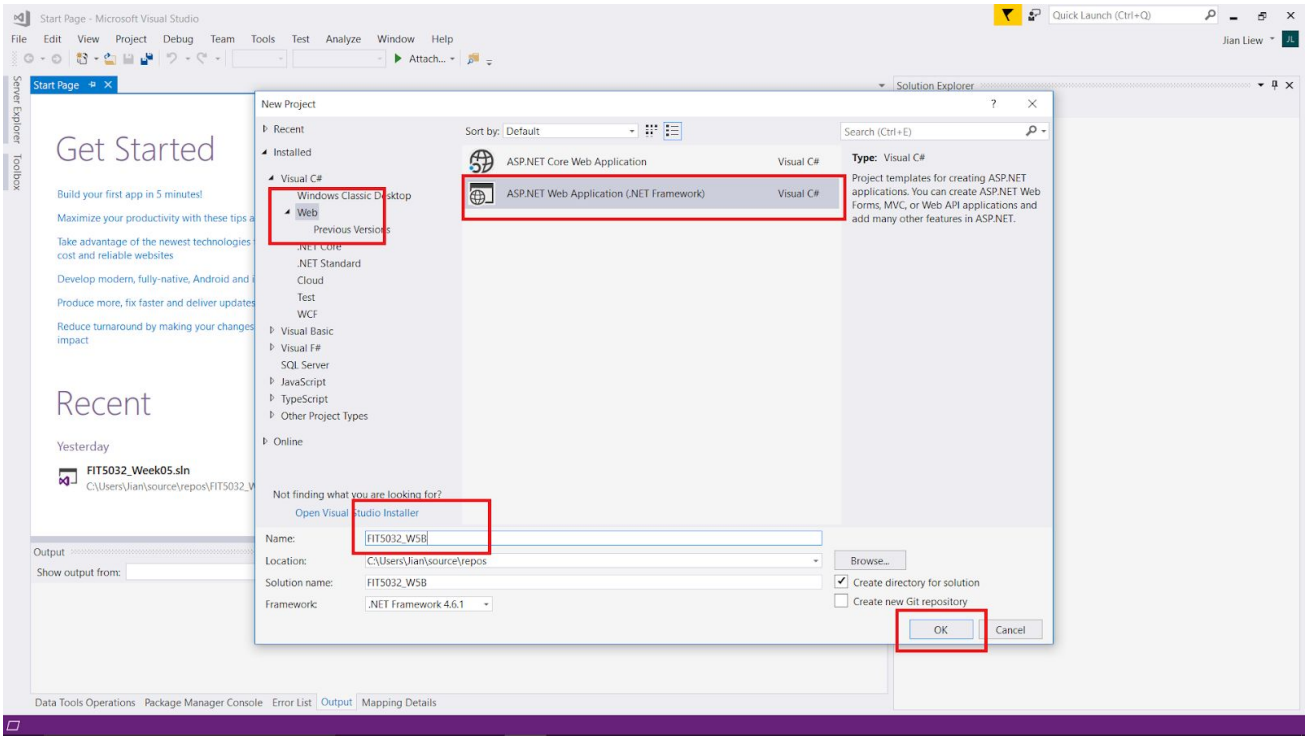
Please remember that this uses the LocalDb as it is the development environment. If you plan to deploy this, the strategy of creating the database will differ slightly.

One of the objectives of this tutorial is to demonstrate that you can manipulate the points and add more points to the database. Normally, it would be pointless to store only a single point or two points to the database.

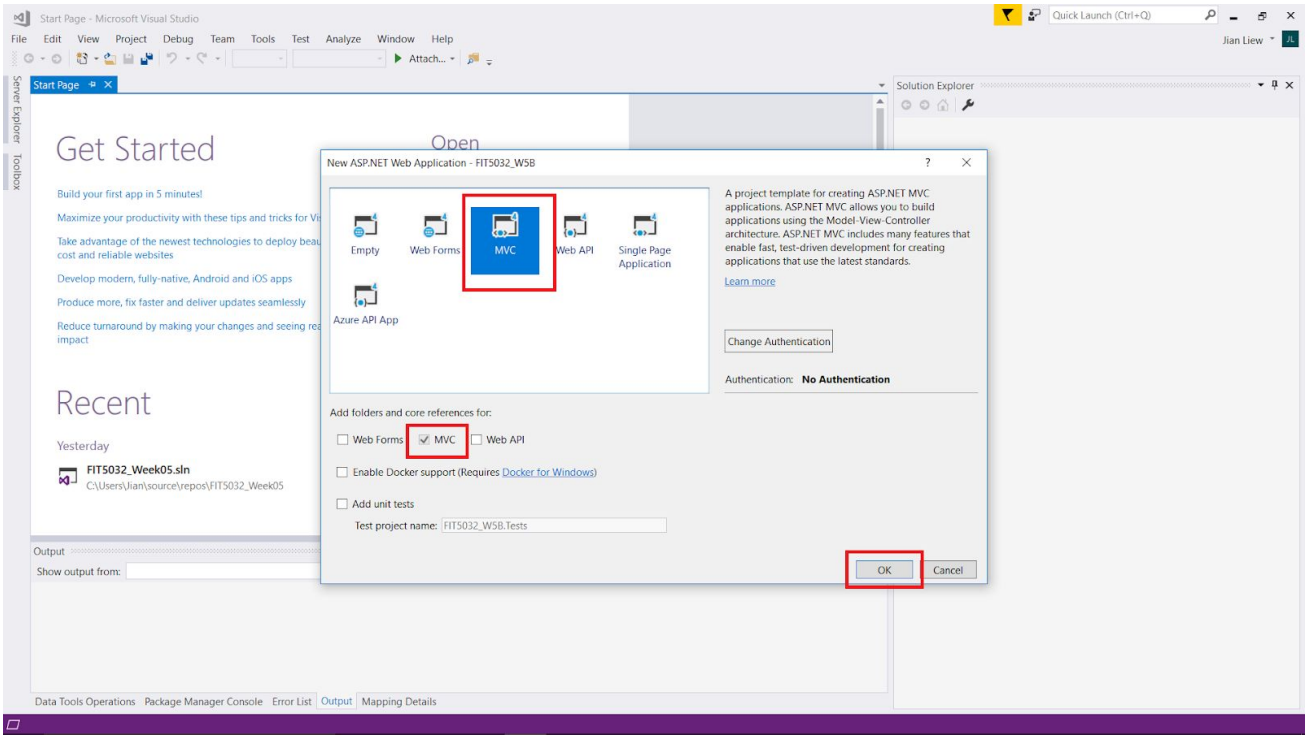
Step 1


Start Page - Microsoft Visual Studio
File Edit View Project Debug Team Tools Test Analyze Window Help
New
Open
Close
Close Solution
Save Selected Items
Save Selected Items As...
Save All
Page Setup...
Print...
Account Settings...
Recent Files
Recent Projects and Solutions
Exit
Project... Ctrl+Shift+N
Repository...
Project From Existing Code...
Open
Get code from a remote version control system or open something on your local drive.
Checkout from:
Visual Studio Team Services
Open Project / Solution
Open Folder
Open Website
Developer News
Xamarin.Forms 3.1: Improving the Little Things
Visual Studio IntelliCode now infers coding conventions for consistent code
Azure Service Fabric Mesh tools now available for Visual Studio 2017
Recent
Yesterday
FIT5032_Week05.sln
New project
Search project templates
Recent project templates:
ASP.NET Web Applica... C#
Create new project...
More news...
Solution Explorer
Output
Show output from:
Data Tools Operations Package Manager Console Error List Output Mapping Details
Ready

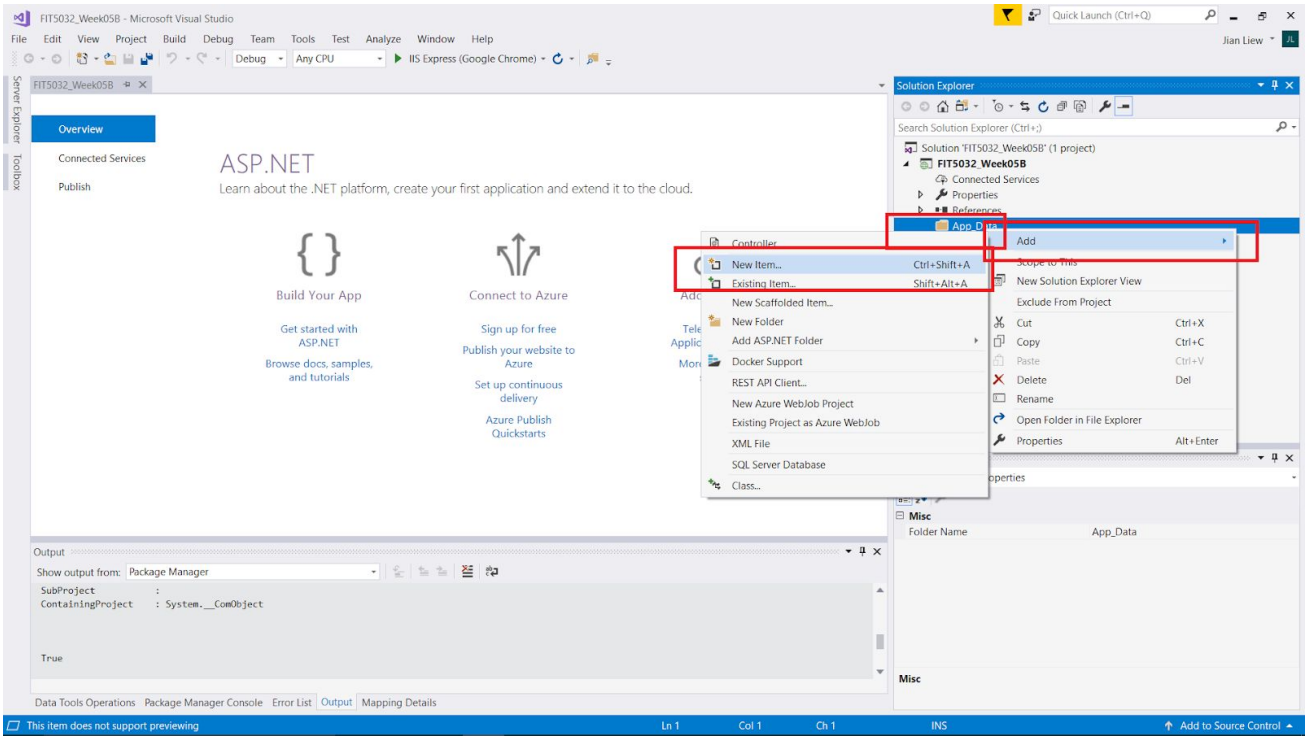
Step 2


Start Page - Microsoft Visual Studio
New Project
Recent
Installed
Visual C#
Windows Classic Desktop
Web
Previous Versions
.NET Core
.NET Standard
Cloud
Test
WCF
Visual Basic
Visual F#
SQL Server
JavaScript
TypeScript
Other Project Types
Online
Sort by: Default
ASP.NET Core Web Application Visual C#
ASP.NET Web Application (.NET Framework) Visual C#
Type: Visual C#
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
Not finding what you are looking for?
Open Visual Studio Installer
Name: FIT5032_W5B
Location: C:\Users\Jian\source\repos
Solution name: FIT5032_W5B
Framework: .NET Framework 4.6.1
Browse...
Create directory for solution
Create new Git repository
OK
Cancel
Get Started
Recent
Yesterday
FIT5032_Week05.sln
Solution Explorer
Output
Data Tools Operations Package Manager Console Error List Output Mapping Details

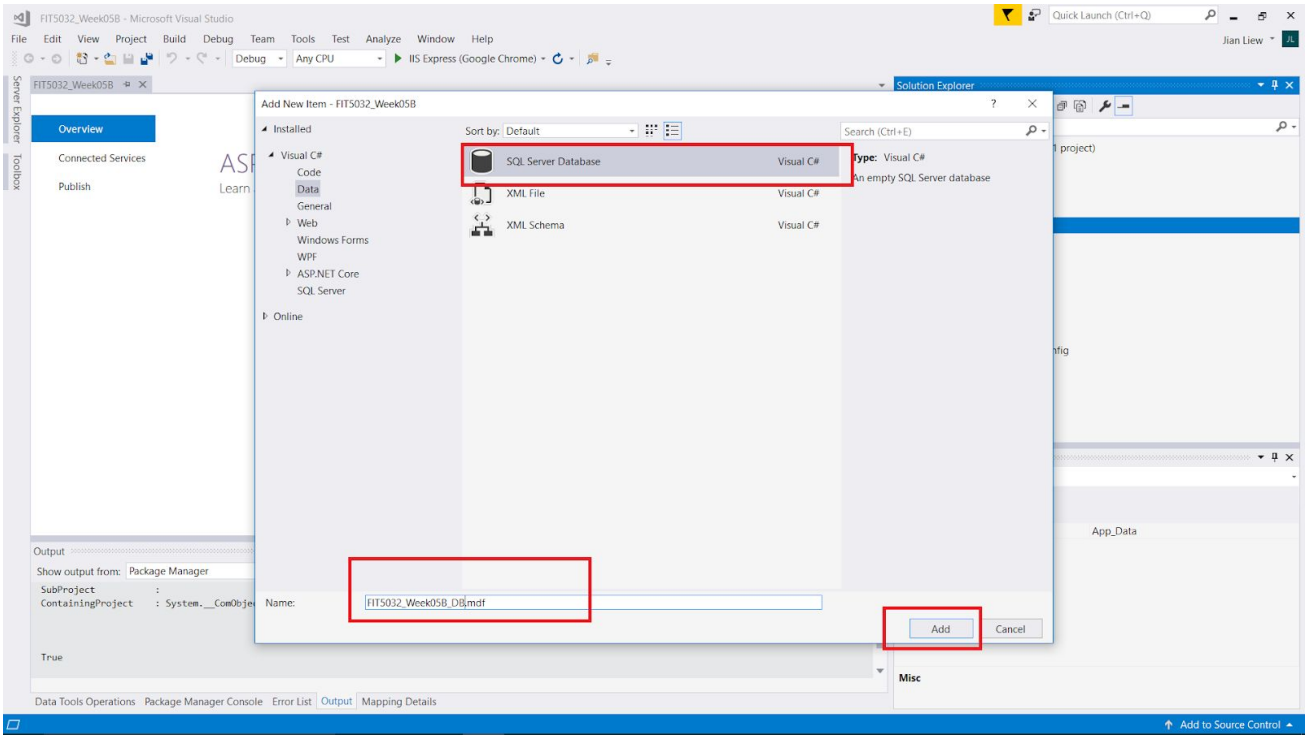
Step 3


Start Page - Microsoft Visual Studio
File Edit View Project Debug Team Tools Test Analyze Window Help
Start Page
Get Started
Open
Recent
Yesterday
FIT5032_Week05.sln
C:\Users\Jian\source\repos\FIT5032_Week05
New ASP.NET Web Application - FIT5032_W5B
Empty
Web Forms
MVC
Web API
Single Page Application
Azure API App
A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.
Learn more
Change Authentication
Authentication: No Authentication
Add folders and core references for:
Web Forms
MVC
Web API
Enable Docker support (Requires Docker for Windows)
Add unit tests
Test project name: FIT5032_W5B.Tests
OK
Cancel
Solution Explorer
Output
Show output from:
Data Tools Operations
Package Manager Console
Error List
Output
Mapping Details

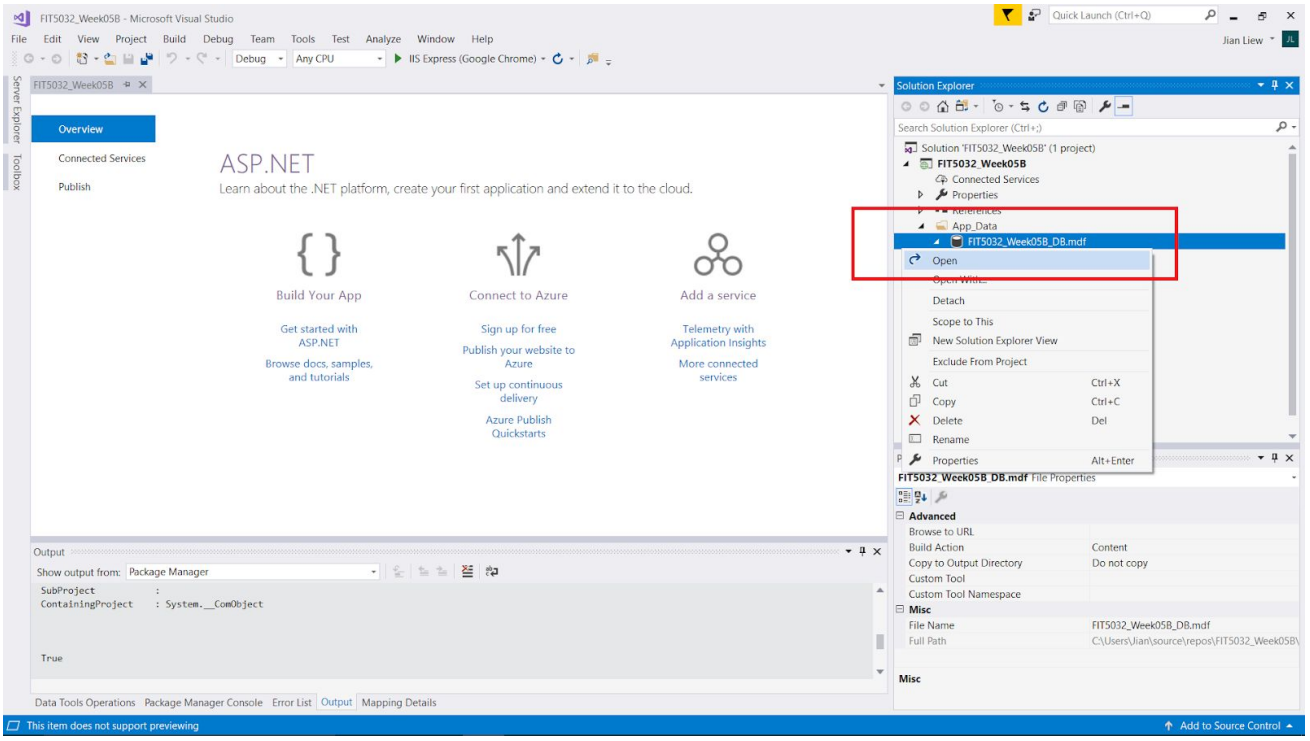
Step 4


FIT5032_Week05B - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug
Any CPU
IIS Express (Google Chrome)
Overview
Connected Services
Publish
ASP.NET
Learn about the .NET platform, create your first application and extend it to the cloud.
Build Your App
Get started with ASP.NET
Browse docs, samples, and tutorials
Connect to Azure
Sign up for free
Publish your website to Azure
Set up continuous delivery
Azure Publish Quickstarts
Solution Explorer
Solution 'FIT5032_Week05B' (1 project)
FIT5032_Week05B
Connected Services
Properties
References
App_Data
Add
New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A
New Scaffolded Item...
New Folder
Add ASP.NET Folder
Docker Support
REST API Client...
New Azure WebJob Project
Existing Project as Azure WebJob
XML File
SQL Server Database
Class...
New Solution Explorer View
Exclude From Project
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Rename
Open Folder in File Explorer
Properties Alt+Enter
Misc
Folder Name App_Data
Output
Show output from: Package Manager
SubProject :
ContainingProject : System.__ComObject
True
Data Tools Operations
Package Manager Console
Error List
Output
Mapping Details
This item does not support previewing
Ln 1 Col 1 Ch 1 INS
Add to Source Control

Step 5

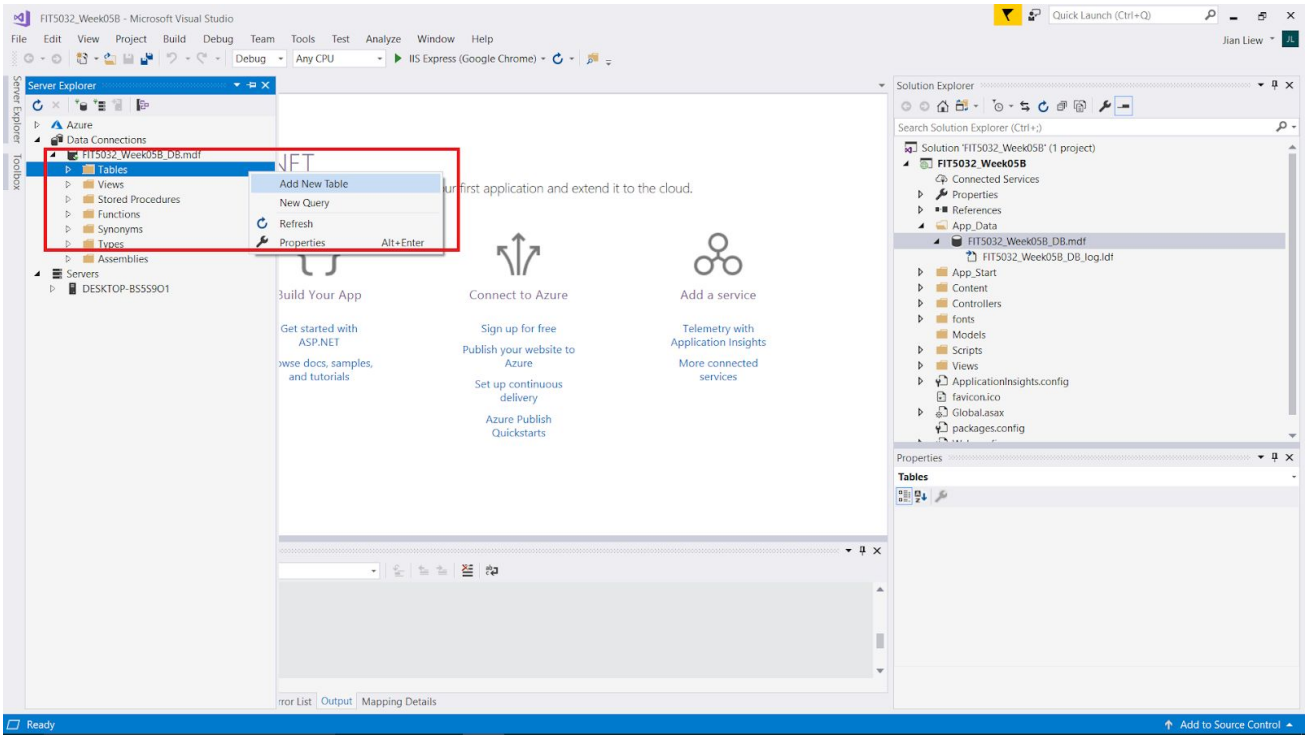

FIT5032_Week05B - Microsoft Visual Studio
Add New Item - FIT5032_Week05B
Installed
Visual C#
Code
Data
General
Web
Windows Forms
WPF
ASP.NET Core
SQL Server
Online
Sort by: Default
SQL Server Database
XML File
XML Schema
Type: Visual C#
An empty SQL Server database
Name: FIT5032_Week05B_DB.mdf
Add
Cancel
Solution Explorer
Overview
Connected Services
Publish
App_Data
Output
Show output from: Package Manager
Misc
Add to Source Control

Step 6


FIT5032_Week05B - Microsoft Visual Studio
Overview
Connected Services
Publish
ASP.NET
Learn about the .NET platform, create your first application and extend it to the cloud.
Build Your App
Get started with ASP.NET
Browse docs, samples, and tutorials
Connect to Azure
Sign up for free
Publish your website to Azure
Set up continuous delivery
Azure Publish Quickstarts
Add a service
Telemetry with Application Insights
More connected services
Solution Explorer
Solution 'FIT5032_Week05B' (1 project)
FIT5032_Week05B
Connected Services
Properties
References
App_Data
FIT5032_Week05B_DB.mdf
Open
Detach
Scope to This
New Solution Explorer View
Exclude From Project
Cut Ctrl+X
Copy Ctrl+C
Delete Del
Rename
Properties Alt+Enter
FIT5032_Week05B_DB.mdf File Properties
Advanced
Build Action Content
Copy to Output Directory Do not copy
Misc
File Name FIT5032_Week05B_DB.mdf
Output
Show output from: Package Manager
This item does not support previewing
Add to Source Control

Step 7

What we will do not is to create a really simple database that will store the coordinates of a point or location. Of course, you can store an address as well if you want to but an address would need to be geocoded in order for its position to be shown on the map.



Use the following SQL Query

```
CREATE TABLE [dbo].[Location]
(
	[Id] INT NOT NULL PRIMARY KEY IDENTITY(1,1),
	[Name] VARCHAR(MAX) NOT NULL,
	[Description] VARCHAR(MAX) NOT NULL,
	[Latitude] NUMERIC(10, 8) NOT NULL,
	[Longitude] NUMERIC(11, 8) NOT NULL,
	CONSTRAINT CK_Latitude CHECK (Latitude >= -90 AND Latitude <= 90),
	CONSTRAINT CK_Longtitude CHECK (Longitude >= -180 AND Longitude <= 180)
)
```

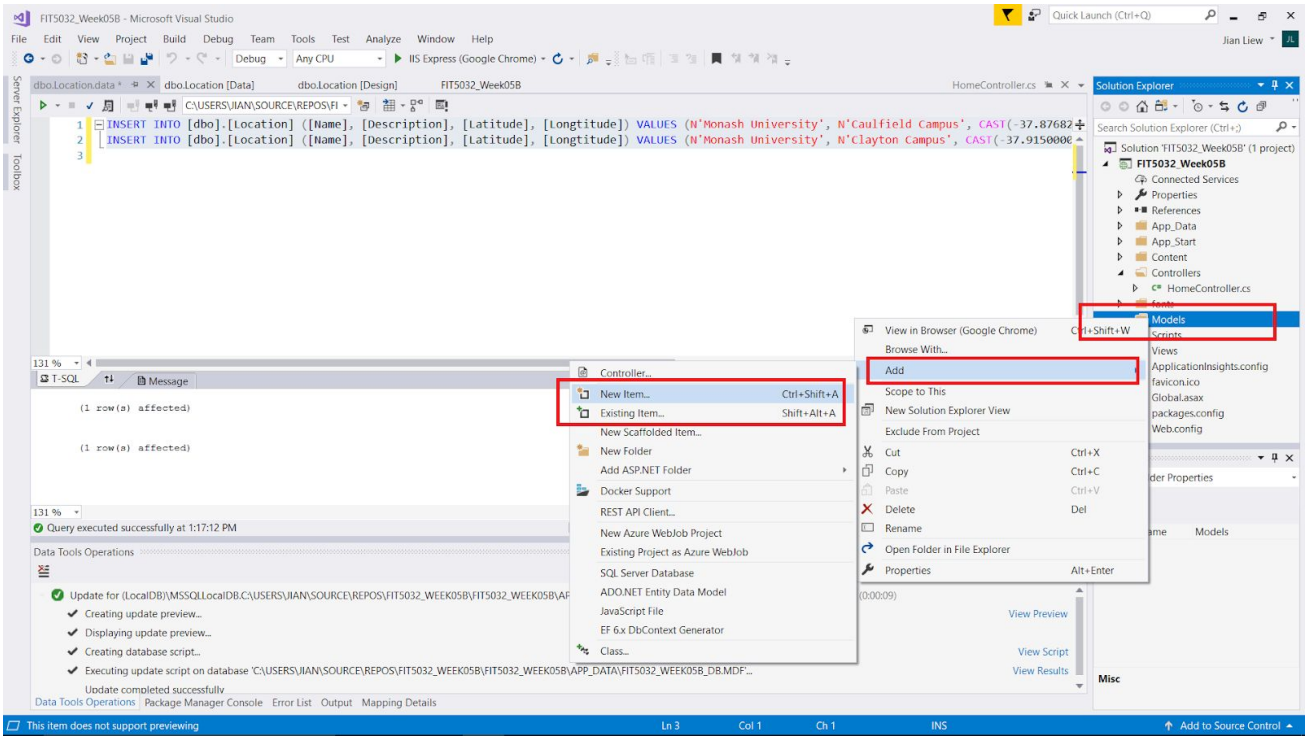
Step 8

Then use the following insert statements to populate the database.

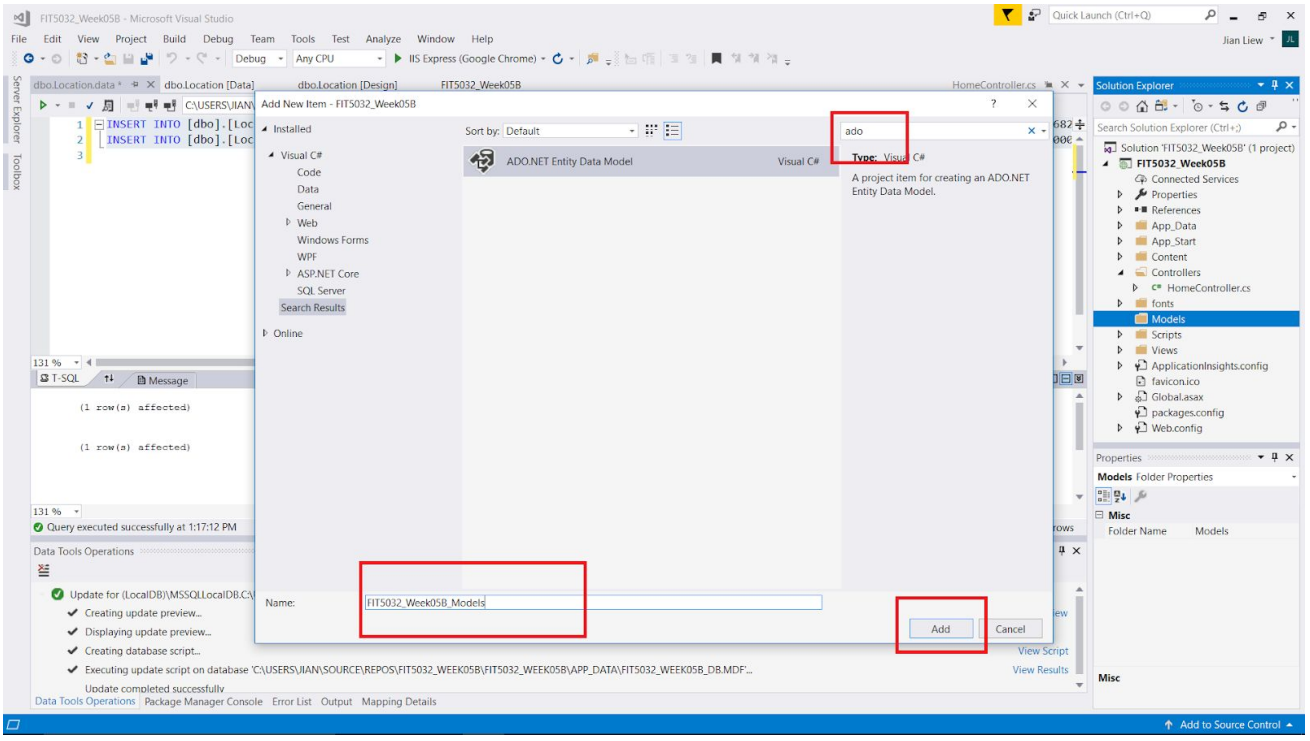
```
INSERT INTO [dbo].[Location] ([Name], [Description], [Latitude], [Longitude])
VALUES (N'Monash University', N'Caulfield Campus', CAST(-37.87682300 AS
Decimal(10, 8)), CAST(145.04583700 AS Decimal(11, 8)))
INSERT INTO [dbo].[Location] ([Name], [Description], [Latitude], [Longitude])
VALUES (N'Monash University', N'Clayton Campus', CAST(-37.91500000 AS
Decimal(10, 8)), CAST(145.13000000 AS Decimal(11, 8)))
```

Step 9

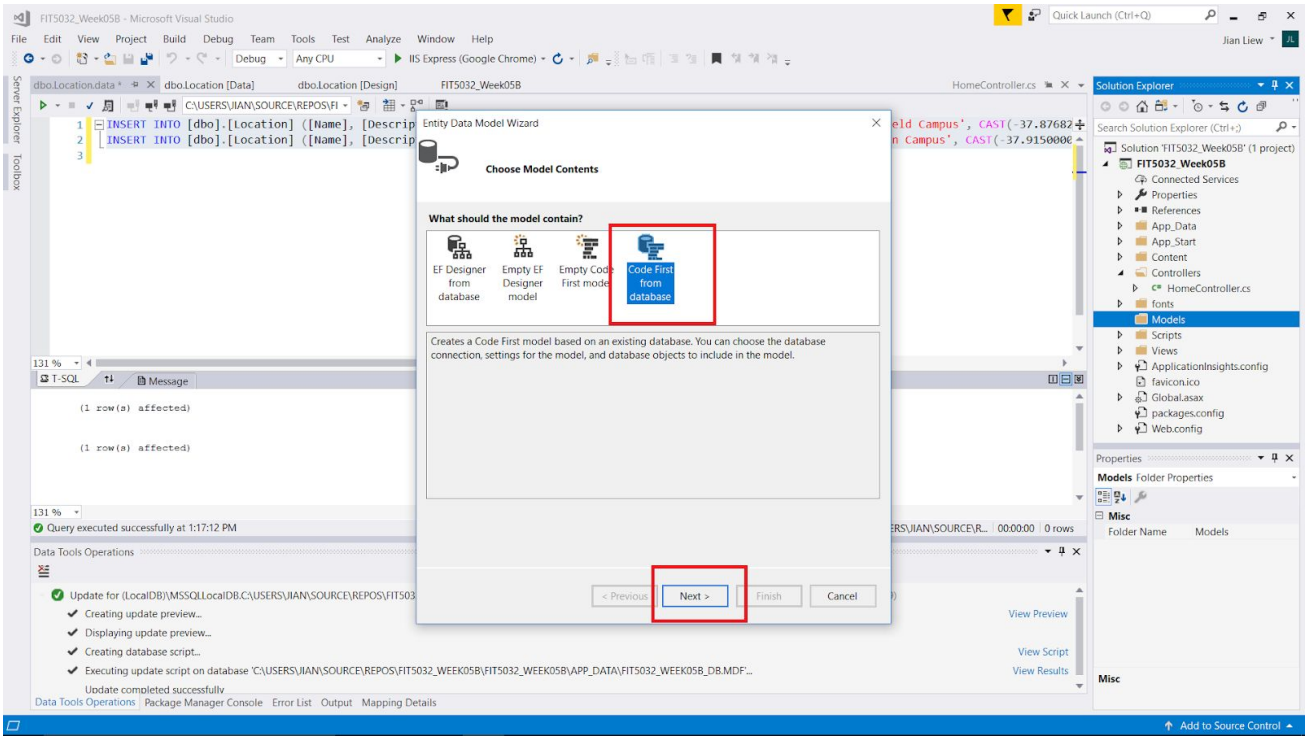
We will now create the model.



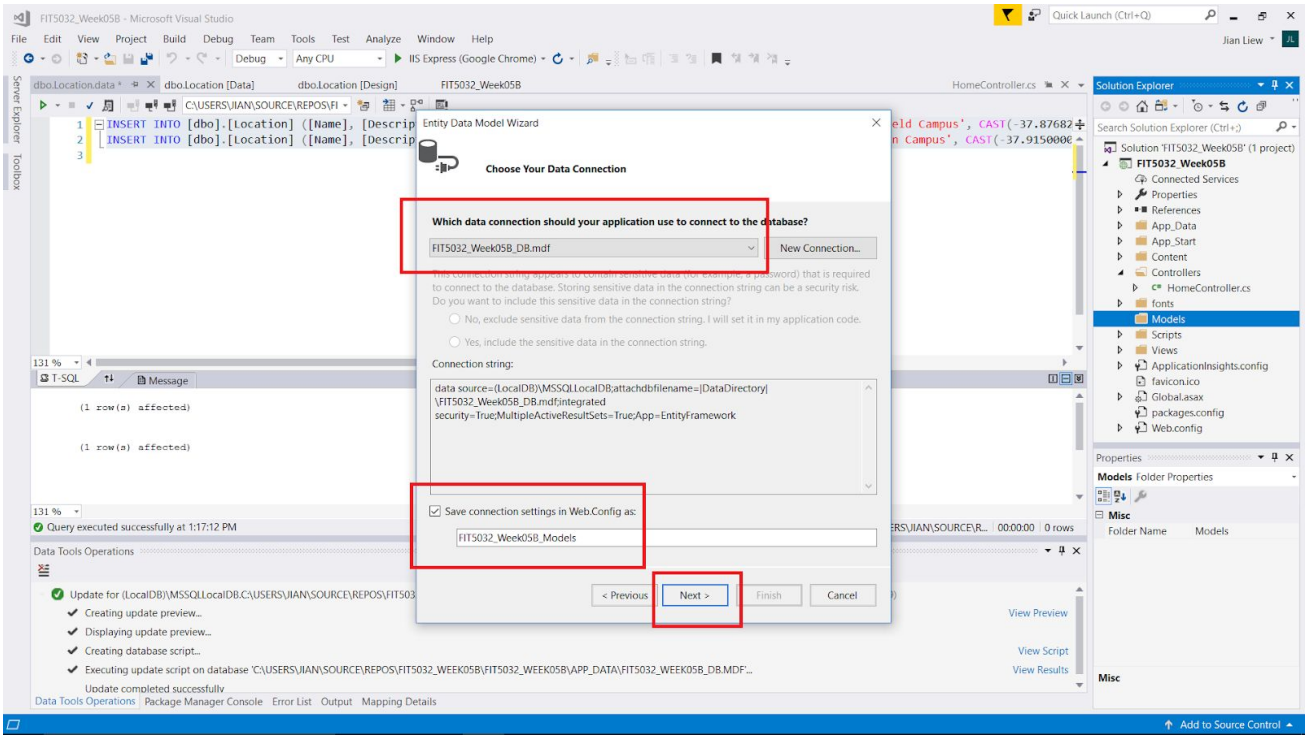
Step 10



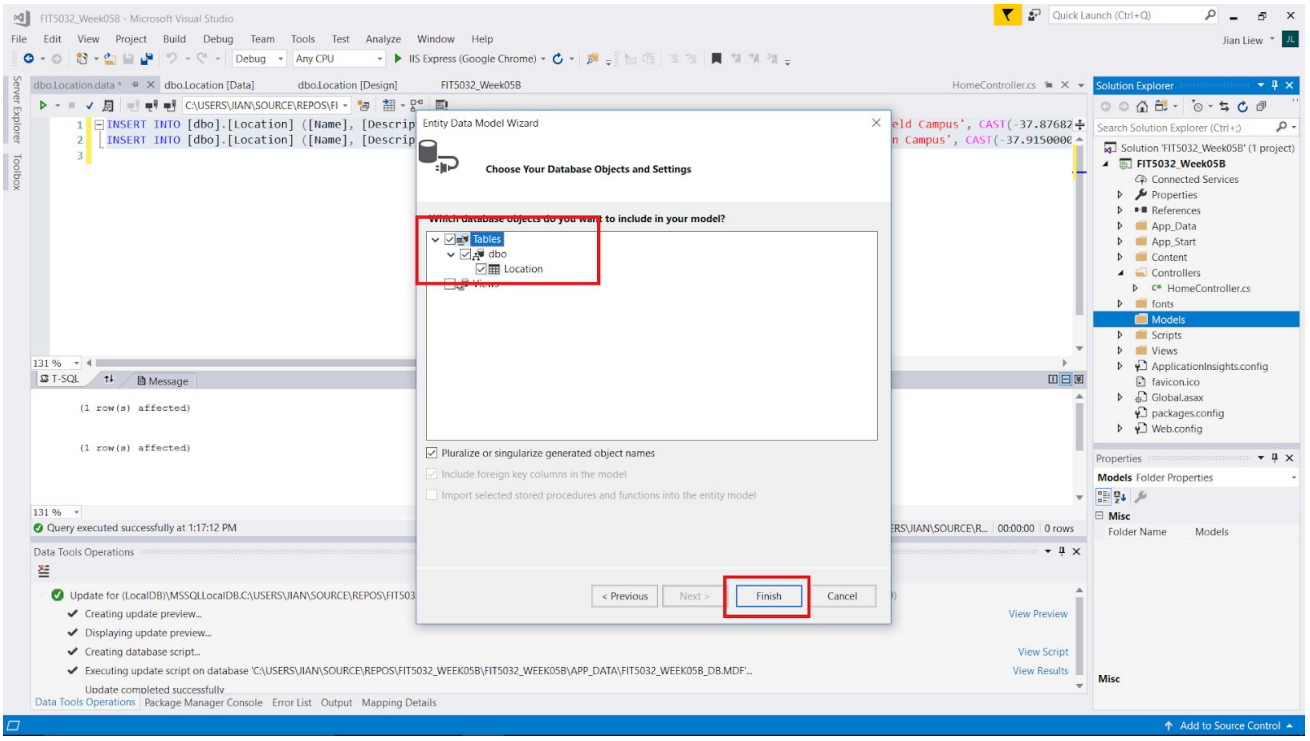
Step 11



Step 12

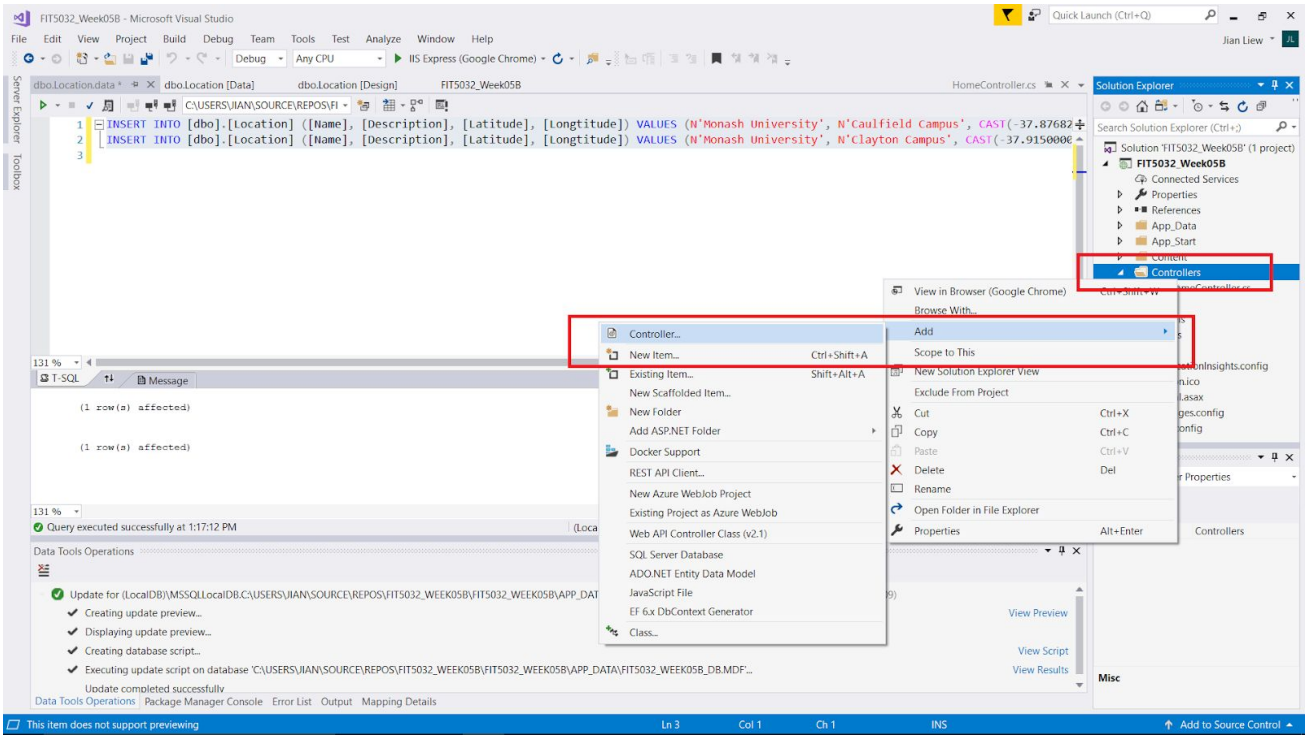


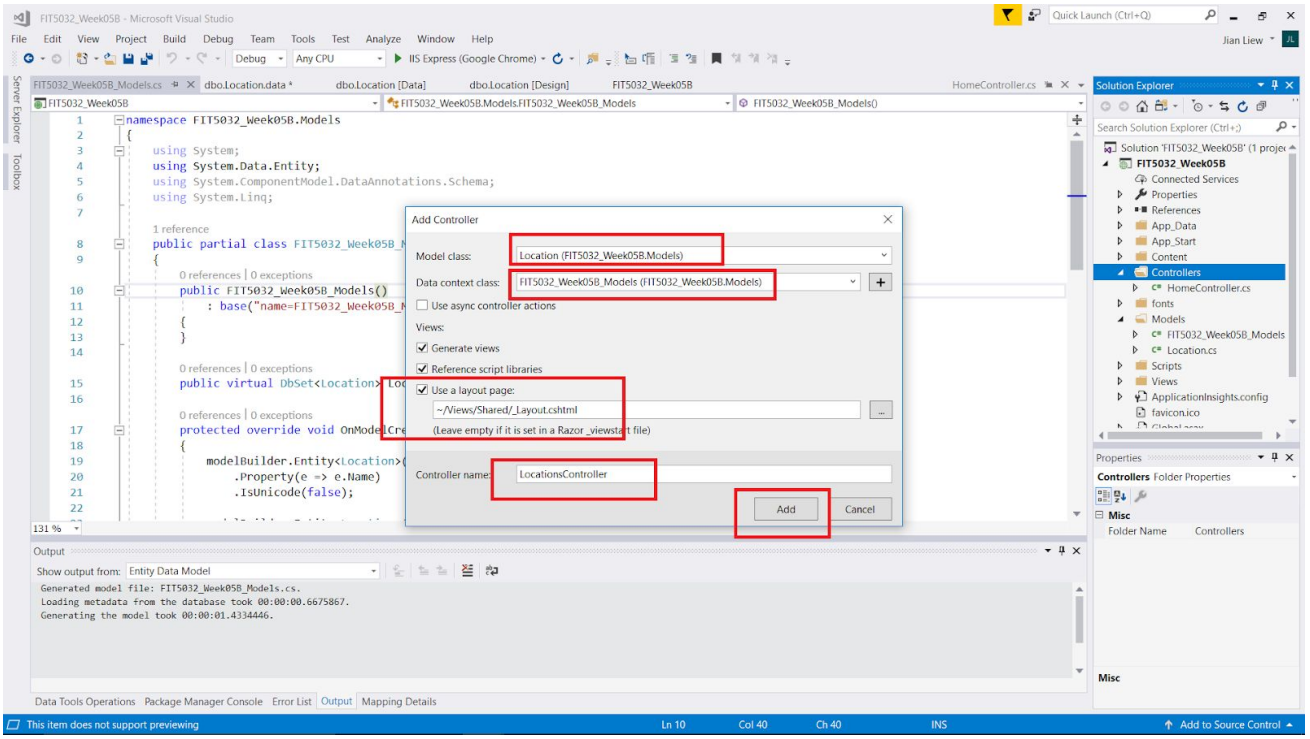
Step 13



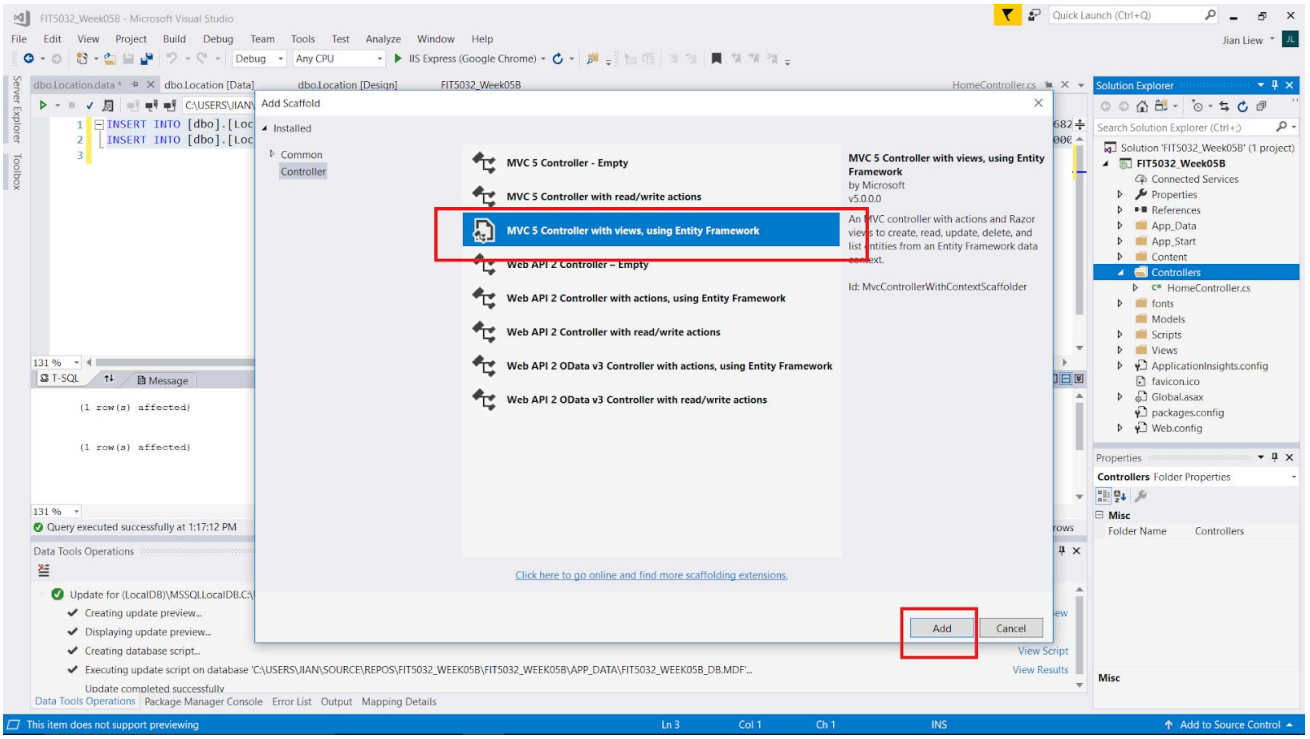
Step 14

Now after the models have been generated, we will proceed to generate the controllers and views using the inbuilt scaffolding features. **(Please rebuild your project first)**



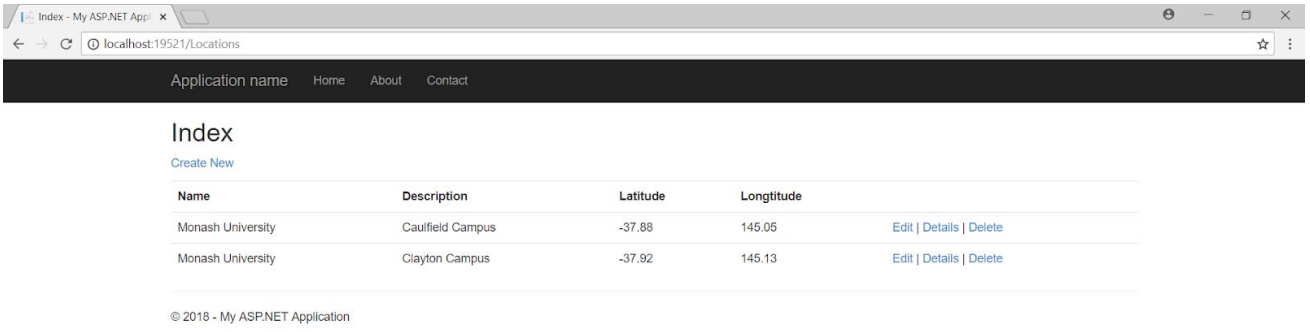


Step 9



Step 10

When you run the solution now, you should see the basic features. We will aim to integrate the index page with a map.



Step 11

Please obtain an API key from https://www.mapbox.com/

At the point this tutorial was written, MapBox was not available on the NuGet package manager so you will have to "install" it manually. (So, you will need to load the JS in the way stated here.)

Your access token is the entire string. (I have hidden my API in the screenshot above)

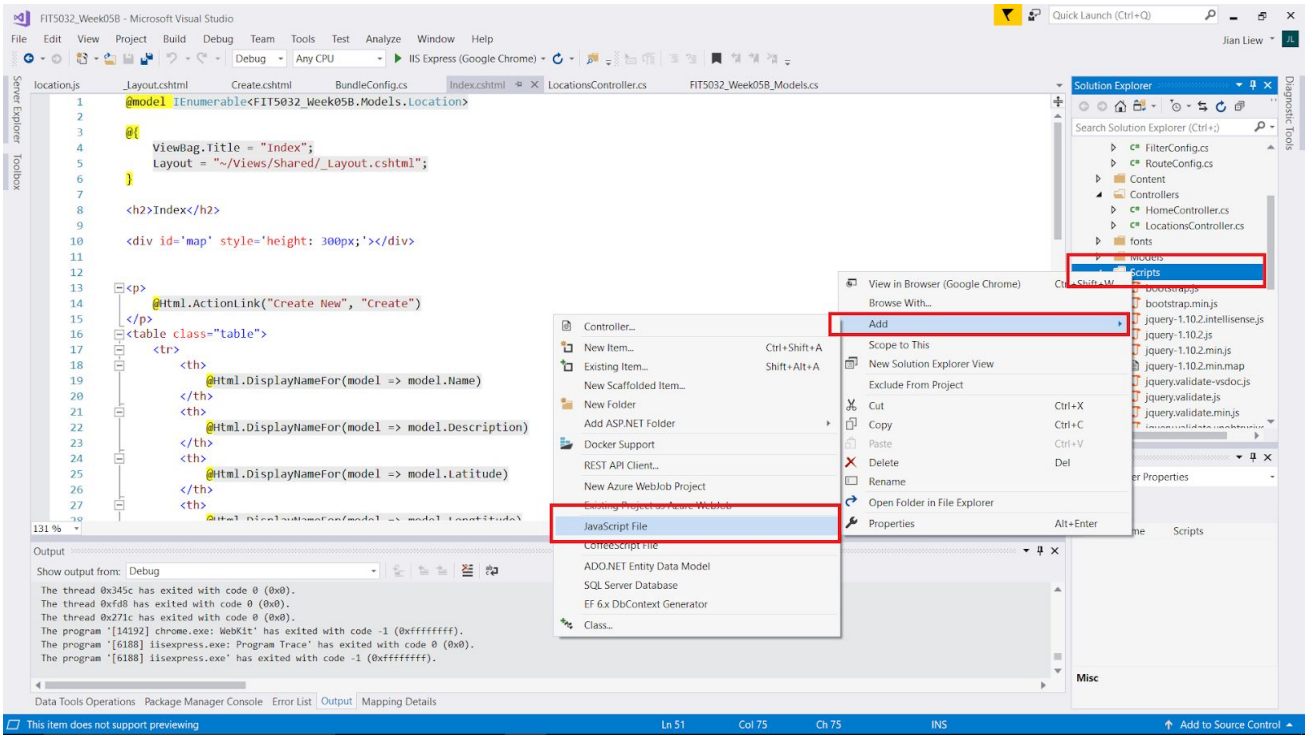
After, that you will need to make the following changes to the codes.

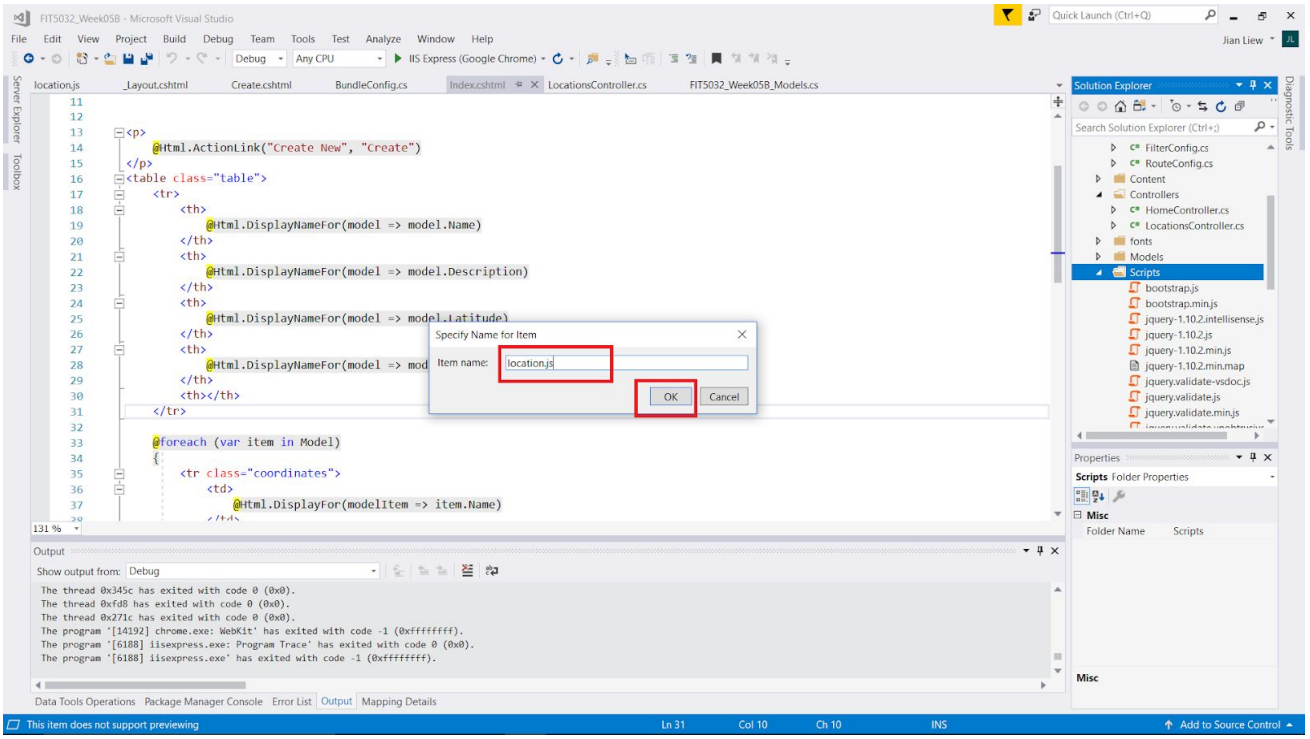
Please change the _Layout.cshtml to the following.

```
<!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")

    @* This is for the mapbox map. *@
    <script src='https://api.mapbox.com/mapbox-gl-js/v0.47.0/mapbox-gl.js'></script>
    <link href='https://api.mapbox.com/mapbox-gl-js/v0.47.0/mapbox-gl.css' rel='stylesheet' />

    @*This is used for the Geocoding Mapbox API*@
    <script
src='https://api.mapbox.com/mapbox-gl-js/plugins/mapbox-gl-geocoder/v2.2.0/mapbox-gl-geocoder.min.js'></scri
pt>
    <link rel='stylesheet'
href='https://api.mapbox.com/mapbox-gl-js/plugins/mapbox-gl-geocoder/v2.2.0/mapbox-gl-geocoder.css'
type='text/css' />
</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>
                    <li>@Html.ActionLink("Location", "Index", "Locations")</li>
                </ul>
            </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>
```

Step 12

Here you will need to add a JavaScript file which I have written.





Source codes of location.js

```
/**
 * This is a simple JavaScript demonstration of how to call MapBox API to load the maps.
 * I have set the default configuration to enable the geocoder and the navigation control.
 * https://www.mapbox.com/mapbox-gl-js/example/popup-on-click/
 *
 * @author Jian Liew <jian.liew@monash.edu>
 */
const TOKEN = "YOUR API KEY";
var locations = [];
// The first step is obtain all the latitude and longitude from the HTML
// The below is a simple jQuery selector
$(".coordinates").each(function () {
    var longitude = $(".longitude", this).text().trim();
    var latitude = $(".latitude", this).text().trim();
    var description = $(".description", this).text().trim();
    // Create a point data structure to hold the values.
    var point = {
        "latitude": latitude,
        "longitude": longitude,
        "description": description
    };

    // Push them all into an array.
    locations.push(point);
});

var data = [];
for (i = 0; i < locations.length; i++) {
    var feature = {
        "type": "Feature",
        "properties": {
            "description": locations[i].description,
            "icon": "circle-15"
        },
        "geometry": {
            "type": "Point",
            "coordinates": [locations[i].longitude, locations[i].latitude]
        }
    };
    data.push(feature)
}

mapboxgl.accessToken = TOKEN;
var map = new mapboxgl.Map({
    container: 'map',
    style: 'mapbox://styles/mapbox/streets-v10',
    zoom: 11,
    center: [locations[0].longitude, locations[0].latitude]
});

map.on('load', function () {
    // Add a layer showing the places.
    map.addLayer({
        "id": "places",
        "type": "symbol",
        "source": {
            "type": "geojson",
```

```
            "data": {
                "type": "FeatureCollection",
                "features": data
            }
        },
        "layout": {
            "icon-image": "{icon}",
            "icon-allow-overlap": true
        }


    });

    map.addControl(new MapboxGeocoder({
       accessToken: mapboxgl.accessToken
    }));;

    map.addControl(new mapboxgl.NavigationControl());
    // When a click event occurs on a feature in the places layer, open a popup at the
    // location of the feature, with description HTML from its properties.
    map.on('click', 'places', function (e) {
        var coordinates = e.features[0].geometry.coordinates.slice();
        var description = e.features[0].properties.description;

        // Ensure that if the map is zoomed out such that multiple
        // copies of the feature are visible, the popup appears
        // over the copy being pointed to.
        while (Math.abs(e.lngLat.lng - coordinates[0]) > 180) {
            coordinates[0] += e.lngLat.lng > coordinates[0] ? 360 : -360;
        }
        new mapboxgl.Popup()
            .setLngLat(coordinates)
            .setHTML(description)
            .addTo(map);
    });

    // Change the cursor to a pointer when the mouse is over the places layer.
    map.on('mouseenter', 'places', function () {
        map.getCanvas().style.cursor = 'pointer';
    });

    // Change it back to a pointer when it leaves.
    map.on('mouseleave', 'places', function () {
        map.getCanvas().style.cursor = '';
    });
});
```

You will also need to make changes to the Index.cshtml of Locations

```
@model IEnumerable<FIT5032_Week05B.Models.Location>

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

<h2>Index</h2>
<div id="map" style="height:300px"></div>
<br />
<p>
    @Html.ActionLink("Create New", "Create","Locations", new {@class="btn btn-primary" })
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Name)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Description)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Latitude)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Longitude)
        </th>
        <th></th>
    </tr>
    @foreach (var item in Model)
    {
        <tr class="coordinates">
            <td>
                @Html.DisplayFor(modelItem => item.Name)
            </td>
            <td class="description">
                @Html.DisplayFor(modelItem => item.Description)
            </td>
            <td class="latitude">
                @Html.DisplayFor(modelItem => item.Latitude)
            </td>
            <td class="longitude">
                @Html.DisplayFor(modelItem => item.Longitude)
            </td>
            <td>
                @Html.ActionLink("Edit", "Edit", new { id = item.Id }) |
                @Html.ActionLink("Details", "Details", new { id = item.Id }) |
                @Html.ActionLink("Delete", "Delete", new { id = item.Id })
            </td>
        </tr>
    }
</table>
@section Scripts {
    @Scripts.Render("~/bundles/mapbox")
}
```

Remember to add the needed scripts into the bundle.

```
using System.Web;
using System.Web.Optimization;

namespace FIT5032_Week05B
{
    public class BundleConfig
    {
        // For more information on bundling, visit
https://go.microsoft.com/fwlink/?LinkId=301862
        public static void RegisterBundles(BundleCollection bundles)
        {
            bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
                        "~/Scripts/jquery-{version}.js"));

            bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
                        "~/Scripts/jquery.validate*"));

            // Use the development version of Modernizr to develop with and learn from. Then,
when you're
            // ready for production, use the build tool at https://modernizr.com to pick only
the tests you need.
            bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
                        "~/Scripts/modernizr-*"));

            bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(
                      "~/Scripts/bootstrap.js",
                      "~/Scripts/respond.js"));

            // I added the location.js to the bundle called mapbox.
            bundles.Add(new ScriptBundle("~/bundles/mapbox").Include(
                      "~/Scripts/location.js"));

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

You can learn how to do this by reading the codes. (Reverse Engineering)

The idea here is that,

1. You must first load the main JavaScript API file. (This is done at the head section of the shared layout).
2. After that, you will have to introduce HTML markup at the Index.cshtml page. (Notice that I have introduced additional classes at certain places. The idea of it is these are the tags that my JavaScript will use to get the values from. These values will then be used to pass data to load the maps.
3. Next, you will need to include your own JavaScript in the bundle. You can sometimes skip this step but this is one of the better ways to do it. (Notice that, here I created a bundle called mapbox and added the location.js as part of it)
4. Soon after, I will load my own Javascript file at the Index page of the Location. (At the end of the Index.cshtml inside Location folder, there I have added a @section which will load the correct script referring to the name of the bundle).
5. Then, you will write your own location.js which is need to load the correct data into the maps.

One issue that you might encounter is the accuracy of the location. Please at the following above the latitude and longitude fields.

```
[DisplayFormat(DataFormatString = "{0:###.########}")]
```

The end result of it should be similar to the screen shot at the earlier page of this document.

Summary

Upon completion of this supplementary tutorial, students will gain a general understanding on how to pass data to the JavaScript side based on HTML markups. Even though so, this is not a good way of doing it as, latitude and longitude often gets updated and it is not often stored into a database.

In fact, there are better ways to do this, this tutorial demos a simple example of achieving such a use case. The better way to do this is to introduce a web API to retrieve this information. It will also provider cleaner codes as well.

Additional information

If you wish to change the icon, you will need to look at this
https://www.mapbox.com/maki-icons/

Challenge Exercise

Instead of asking the user to enter the coordinates a "geocoding" method will be used instead. You will need to look up how to geocode an address into the latitude and longitude. (Mapify can be used)

Change Log

Date	Summary
12 August 2018	Removed Gist links and used codes in the document instead.