MONASH University

FIT5032 - Internet Applications Development

Week 05C - Using DataTables (DataTables)

Last updated: 23rd July 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 (DataTables) to show data. DataTables can be found https://datatables.net/ There are other alternatives like jqGrid but this tutorial demonstrates how to use DataTables at a basic level.

Working with tables to display information is a very common use case. The objective of this supplementary material to give a quick introduction on how to use a simple JavaScript API (DataTables) in this case to achieve such a use case.

Objectives

Estimated Time To Complete - 1 hour

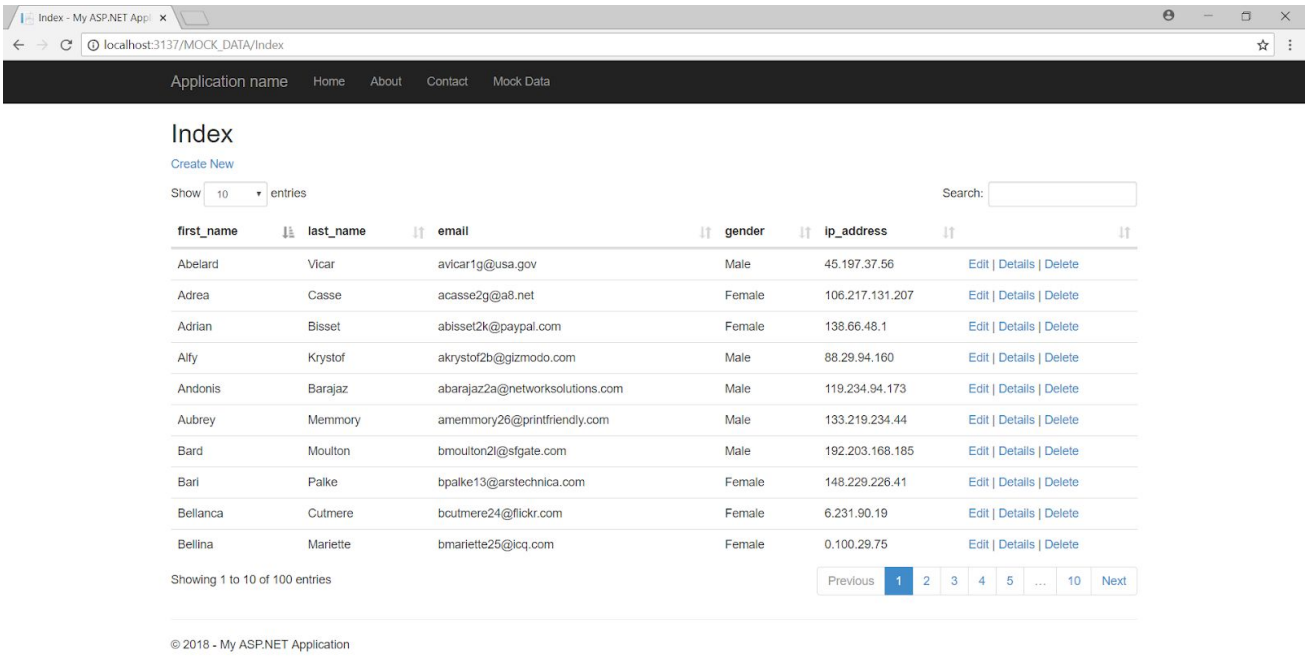
Upon the completion of this tutorial, you will gain a basic understanding of

- How to generate simple mock data using Mockaroo
- How to use DataTables to show data with zero configuration.

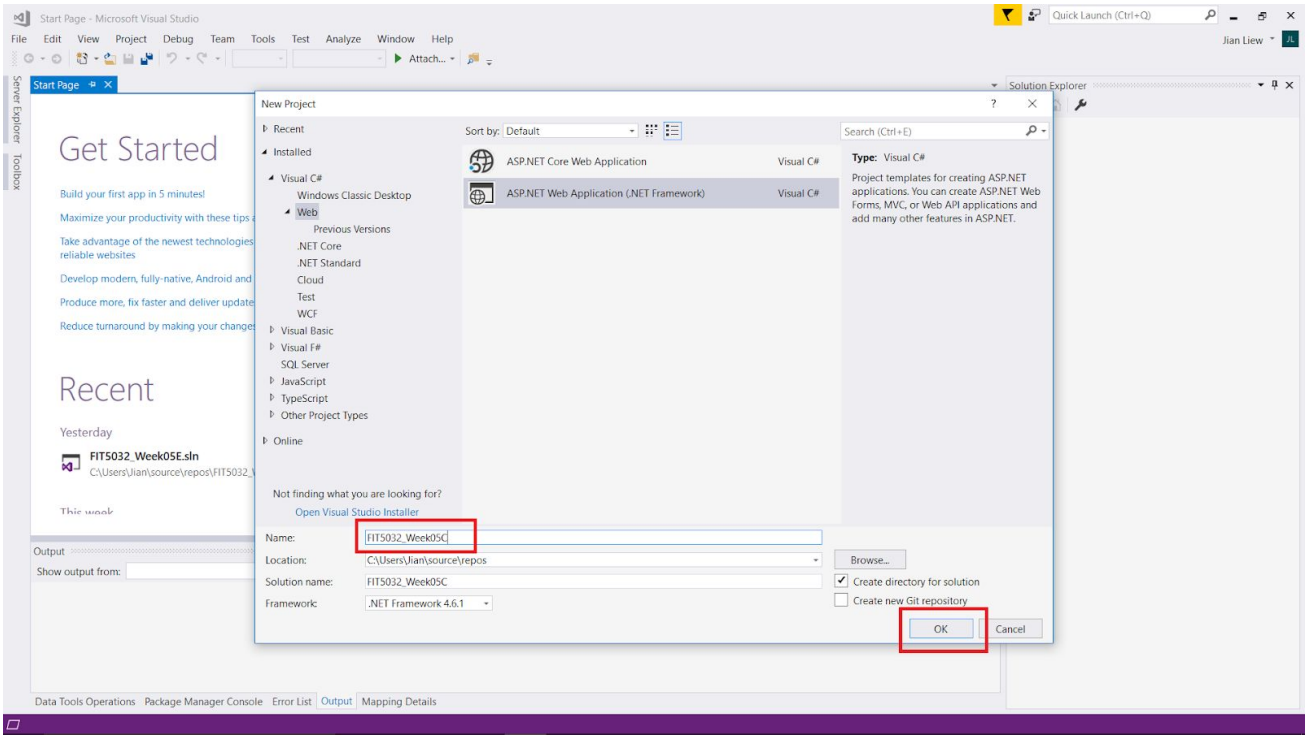
DoubtFire Submission

- **None**

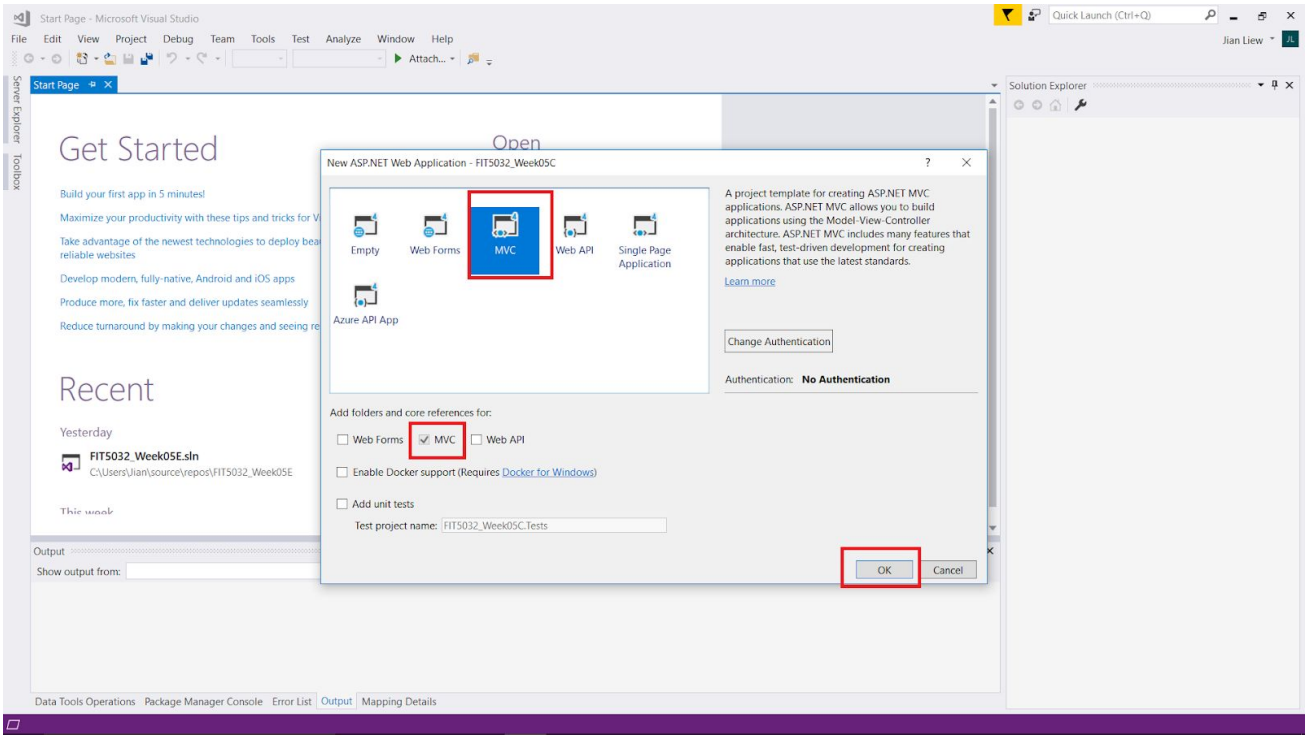
Upon the completion of this tutorial. The end result of your web application would look like the following.



Step 1

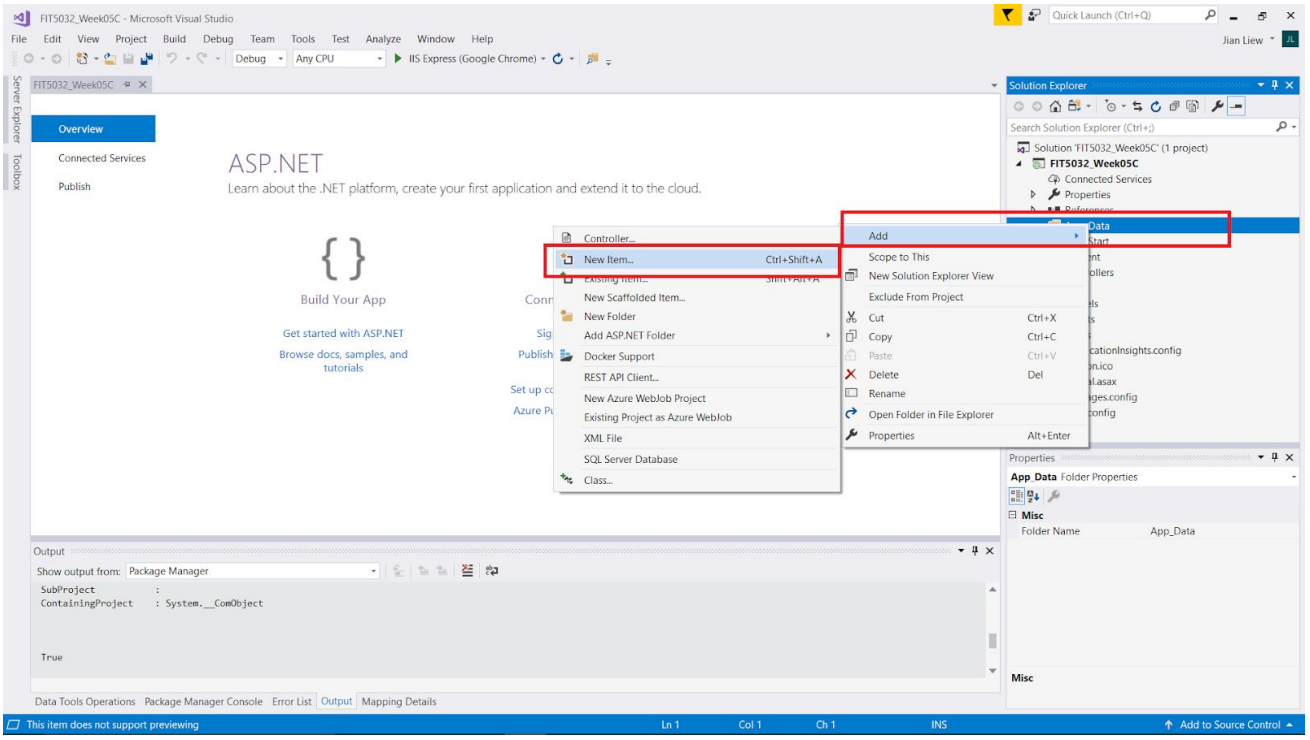

Start Page - Microsoft Visual Studio
File Edit View Project Debug Team Tools Test Analyze Window Help
Quick Launch (Ctrl+Q)
Jian Liew
Start Page
Get Started
Build your first app in 5 minutes!
Recent
Yesterday
FIT5032_Week05E.sln
New Project
Recent
Installed
Visual C#
Windows Classic Desktop
Web
Previous 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
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_Week05C
Location: C:\Users\Jian\source\repos
Solution name: FIT5032_Week05C
Framework: .NET Framework 4.6.1
Browse...
Create directory for solution
Create new Git repository
OK
Cancel
Solution Explorer
Output
Show output from:
Data Tools Operations
Package Manager Console
Error List
Output
Mapping Details

Step 2

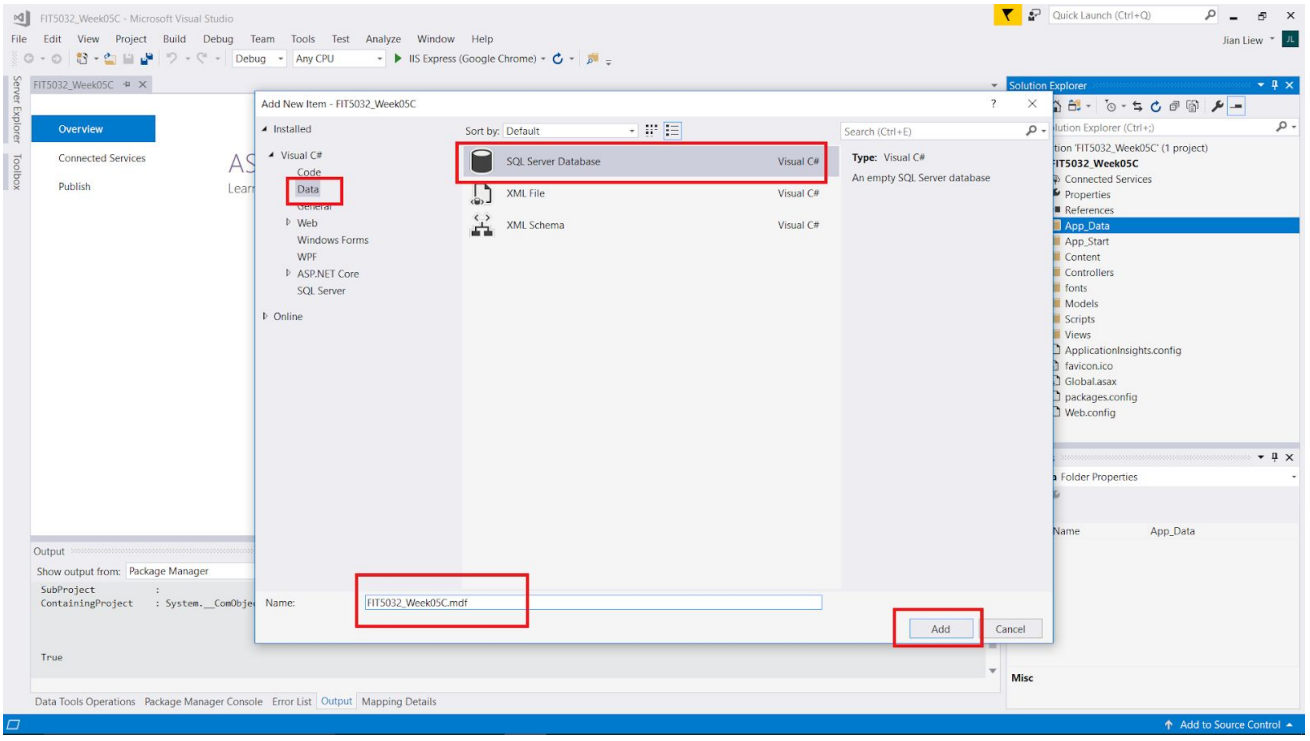

Start Page - Microsoft Visual Studio
File Edit View Project Debug Team Tools Test Analyze Window Help
Quick Launch (Ctrl+Q)
Jian Liew
Start Page
Get Started
Open
Build your first app in 5 minutes!
Recent
Yesterday
FIT5032_Week05E.sln
New ASP.NET Web Application - FIT5032_Week05C
Empty
Web Forms
MVC
Web 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_Week05C.Tests
OK
Cancel
Solution Explorer
Output
Show output from:
Data Tools Operations
Package Manager Console
Error List
Output
Mapping Details

Step 3

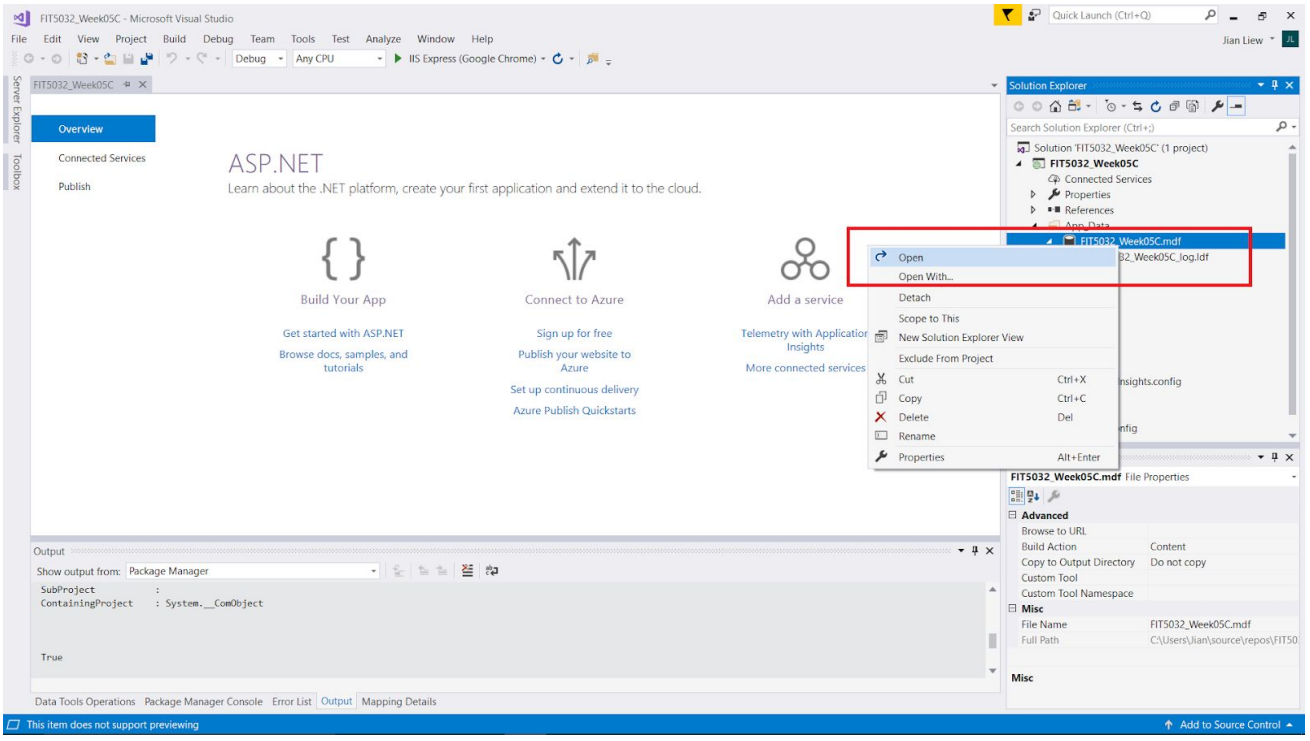
I will now use a Database First approach to generate some mock data.



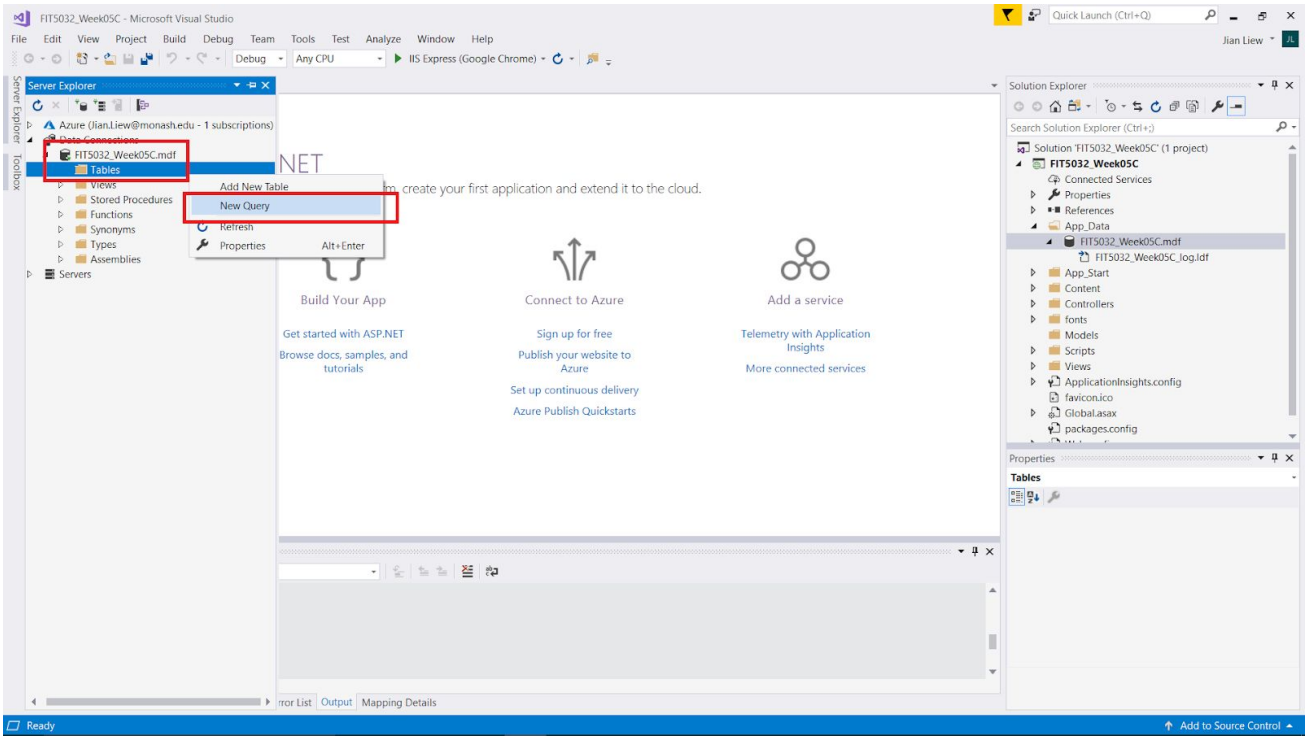
Step 4



Step 5


FIT5032_Week05C - 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
Open
Open With...
Detach
Scope to This
New Solution Explorer View
Exclude From Project
Cut Ctrl+X
Copy Ctrl+C
Delete Del
Rename
Properties Alt+Enter
Solution Explorer
Solution 'FIT5032_Week05C' (1 project)
FIT5032_Week05C
Connected Services
Properties
References
App_Data
FIT5032_Week05C.mdf
FIT5032_Week05C_log.ldf
FIT5032_Week05C.mdf File Properties
Advanced
Build Action Content
Copy to Output Directory Do not copy
Misc
File Name FIT5032_Week05C.mdf
Output
Package Manager
This item does not support previewing
Add to Source Control

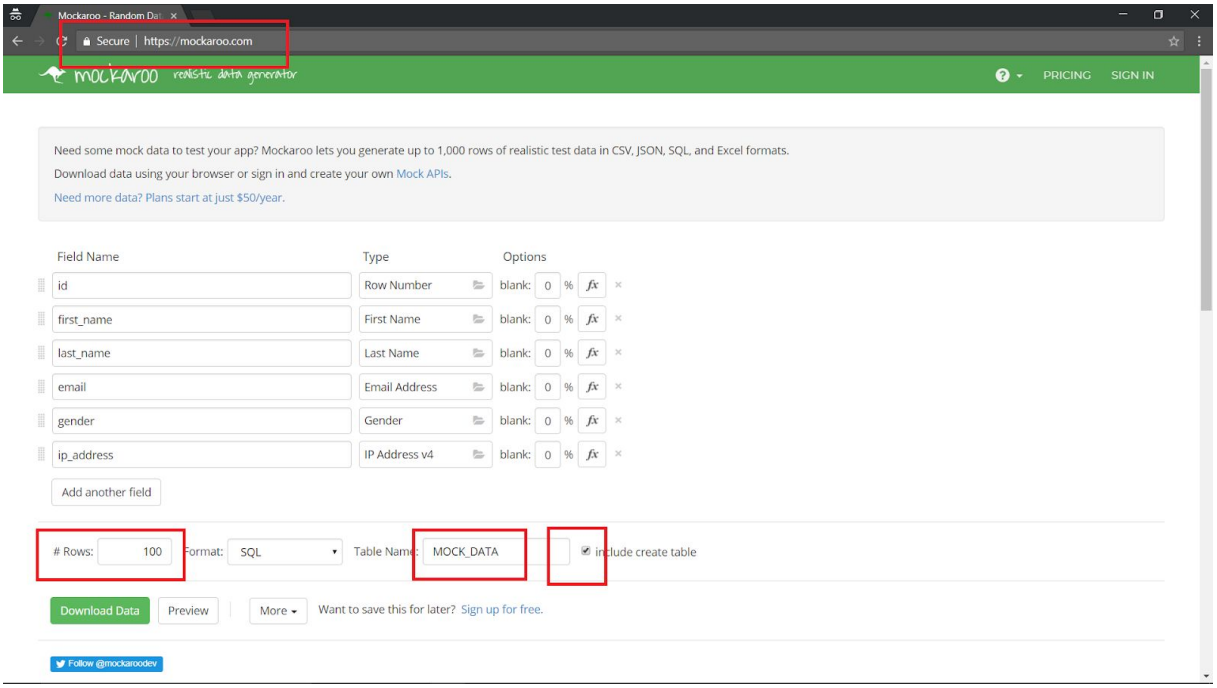
Step 6

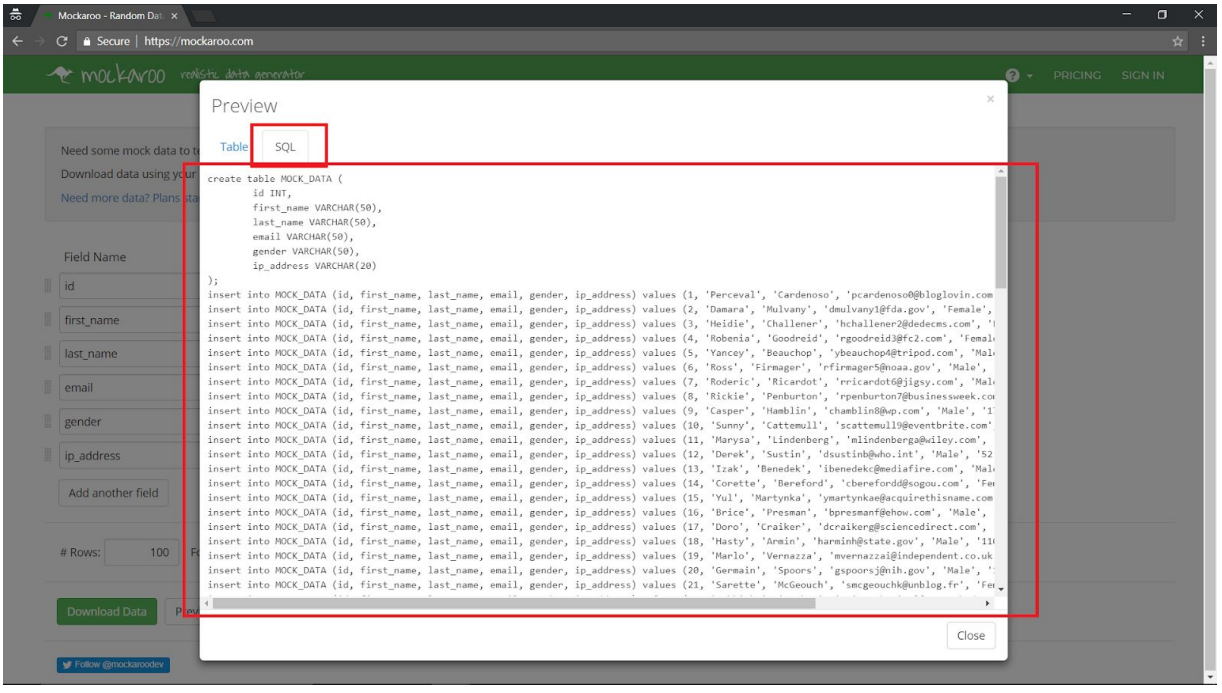

FIT5032_Week05C - Microsoft Visual Studio
Server Explorer
Azure (Jian.Liew@monash.edu - 1 subscriptions)
Data Connections
FIT5032_Week05C.mdf
Tables
Views
Stored Procedures
Functions
Synonyms
Types
Assemblies
Servers
Add New Table
New Query
Refresh
Properties Alt+Enter
Build Your App
Get started with 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
App_Start
Content
Controllers
fonts
Models
Scripts
Views
ApplicationInsights.config
favicon.ico
Global.asax
packages.config
Properties
Tables
Output
Mapping Details
Ready
Add to Source Control

Step 7

I will now generate some mock data from https://mockaroo.com/

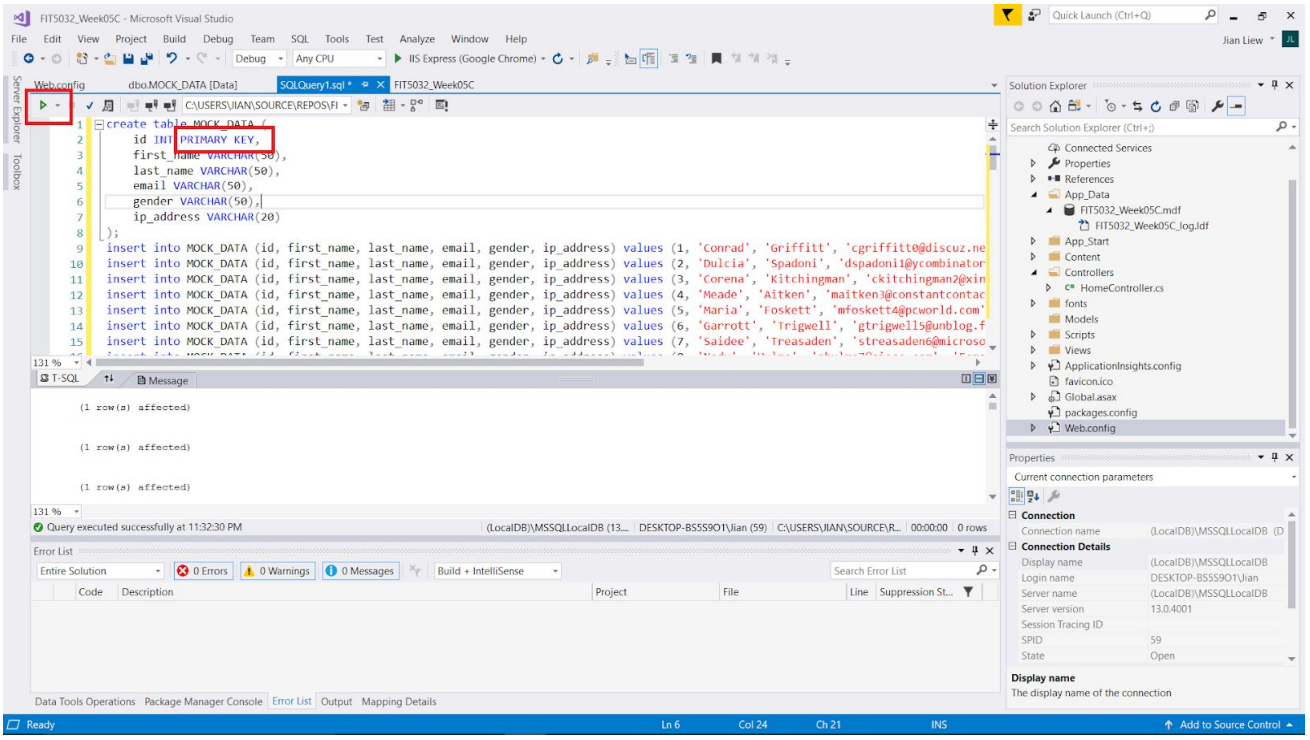
I will just generate 100 rows with the table name mock data and include the create table statements. Then, I will look at the preview and copy and paste the SQL. You can use Title Casing for the field names if you want but the purpose of this lab is to demonstrate how to use DataTables with mock data.



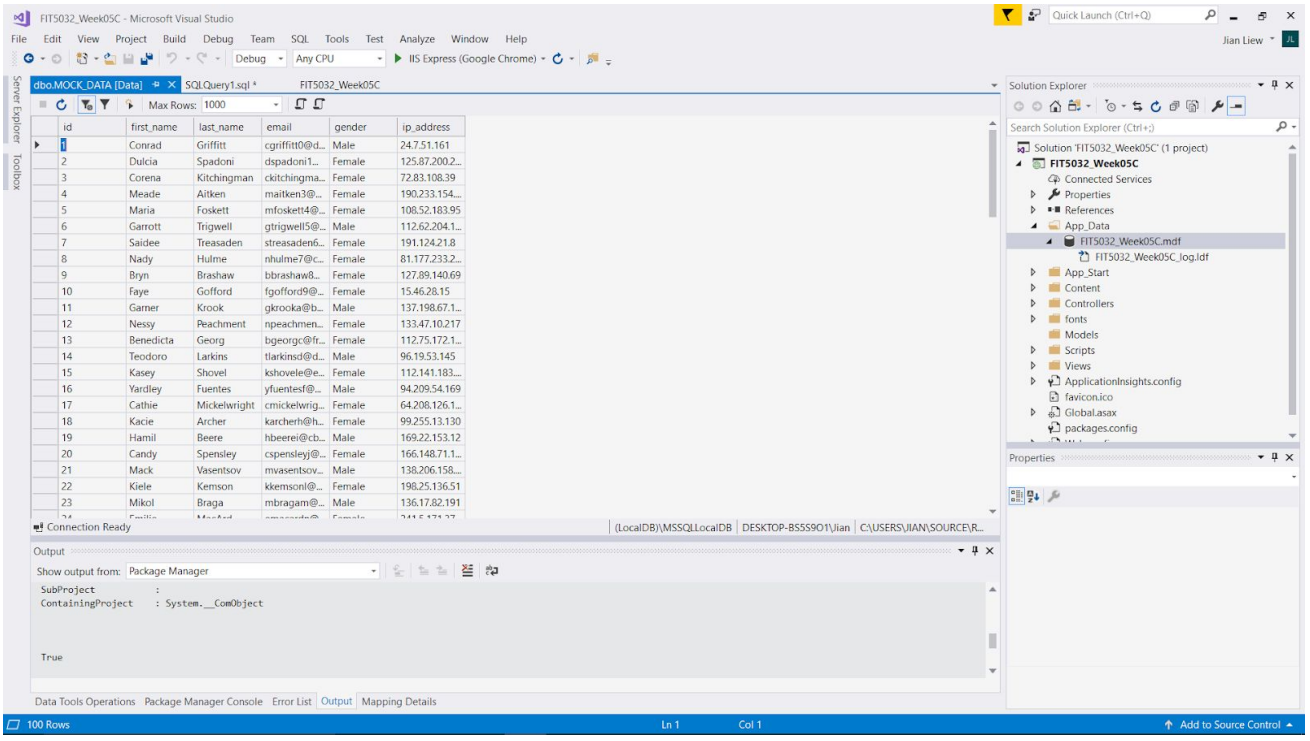


Step 8

Copy and paste the generated SQL into the query window and **introduce a Primary Key** and then run. It will automatically create the tables and run the insert statements.

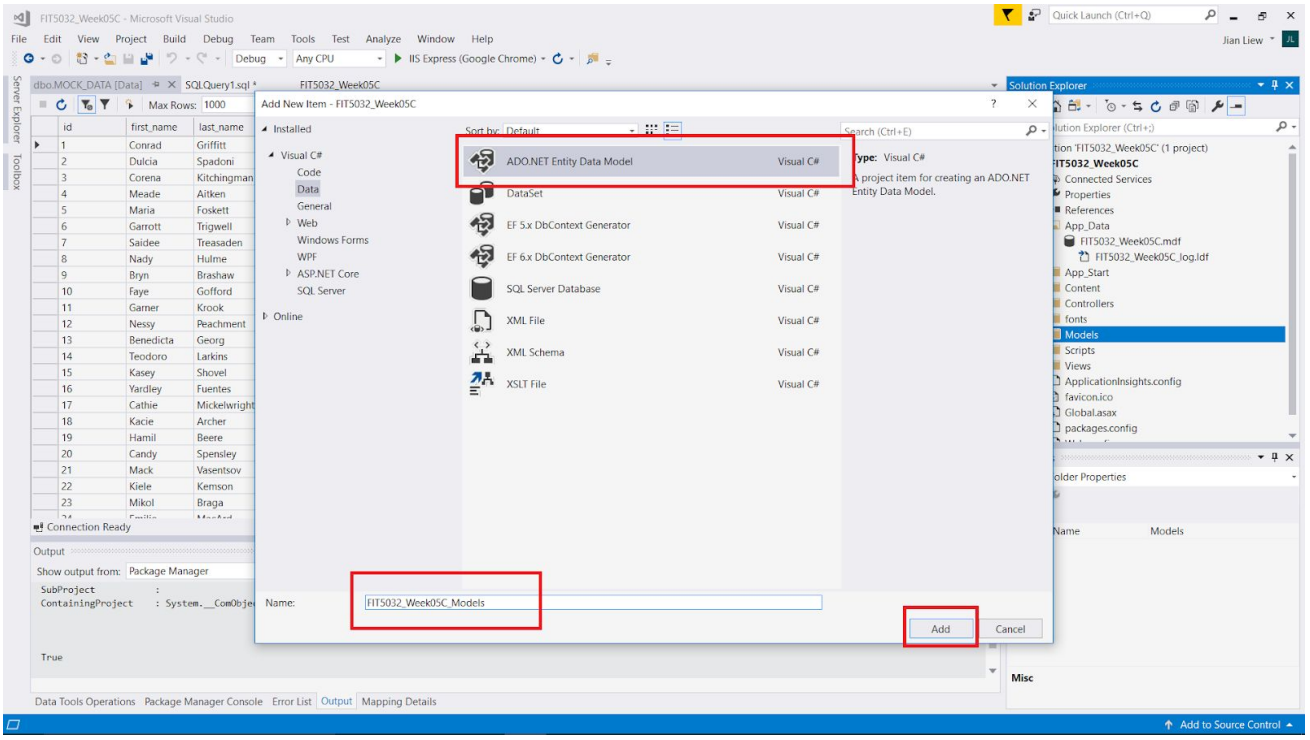


Your Mock_Data table should now have been created with 100 rows.

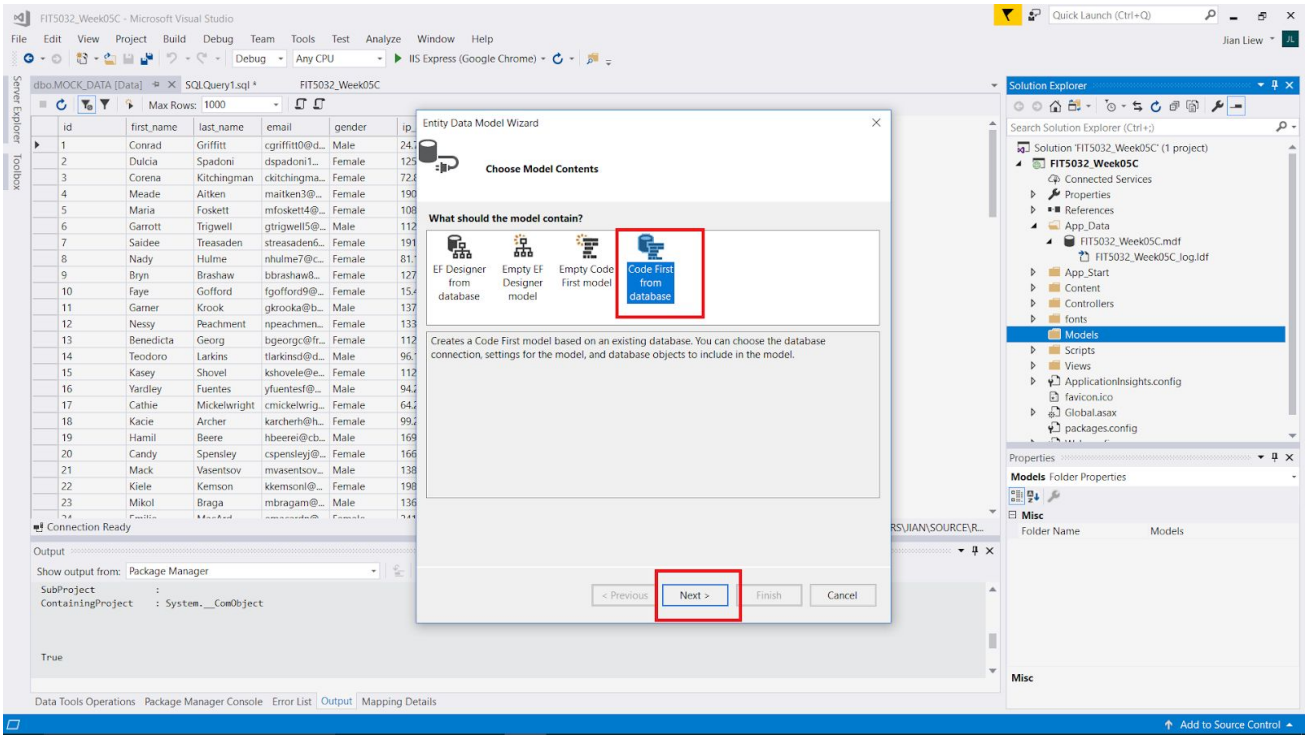


Step 9

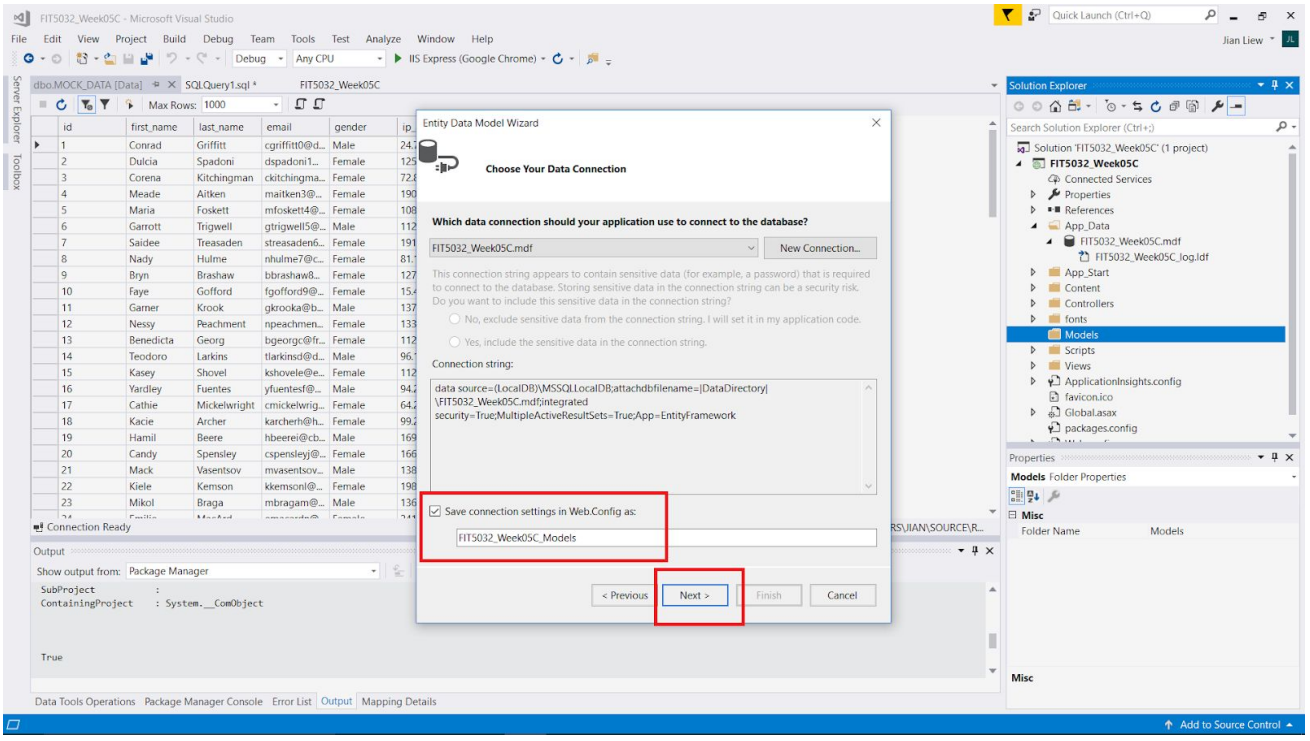
I will now need to generate the Models, Controllers and Views. Similar to what you have done on Week 4. (Scaffolding Controllers)



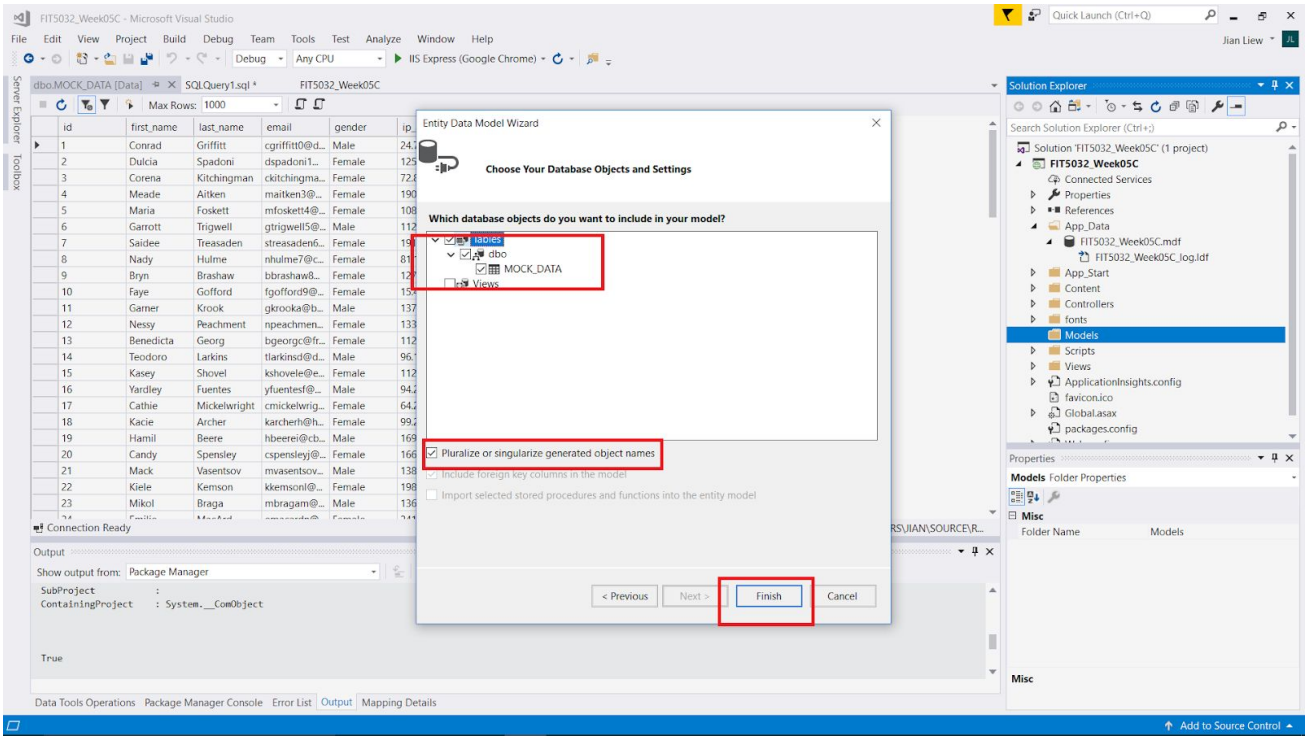
Step 10



Step 11

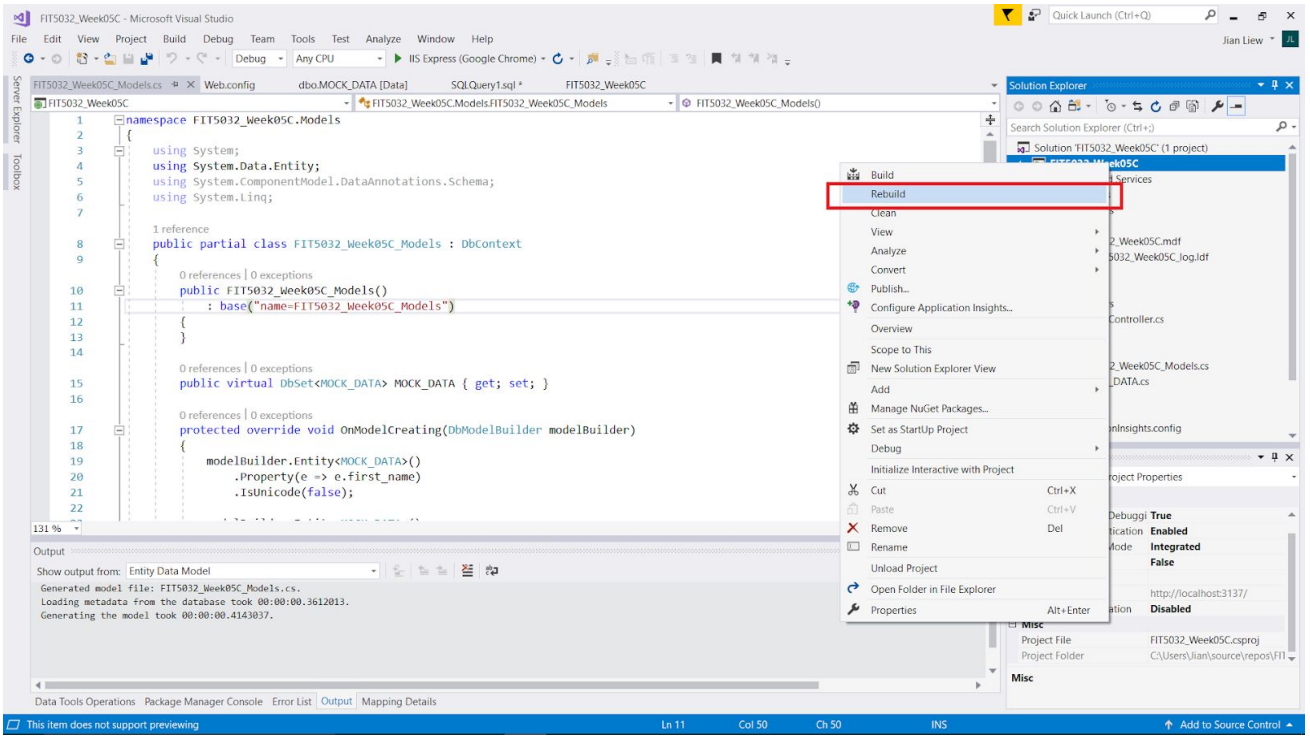


Step 12

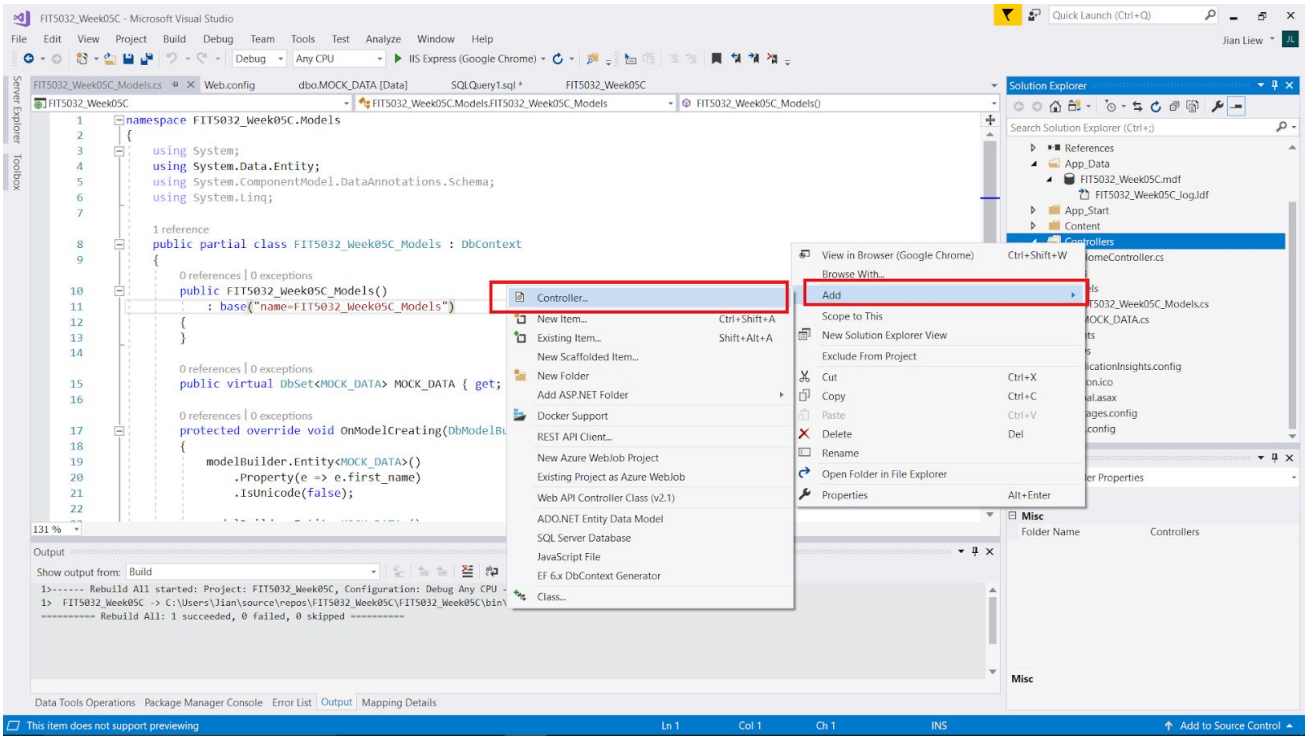


Click "Yes to All" if you see any warnings.

Step 13

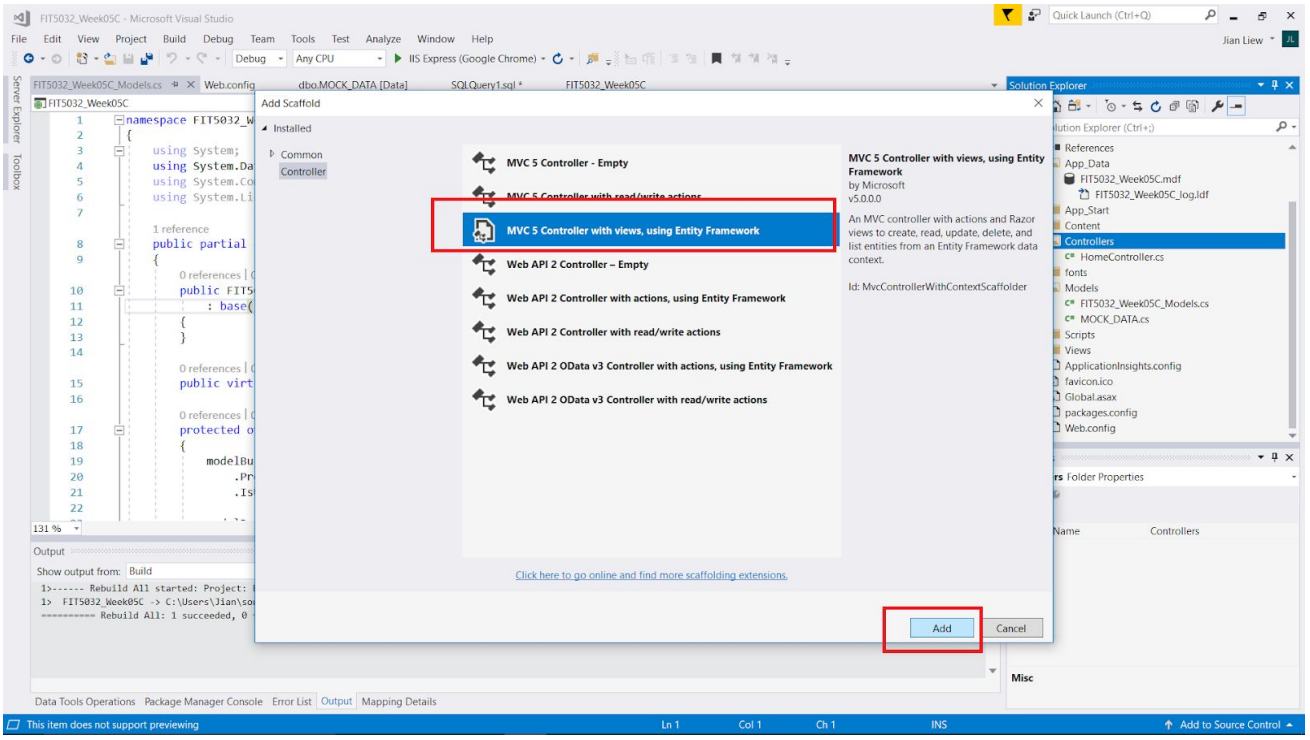


Step 14

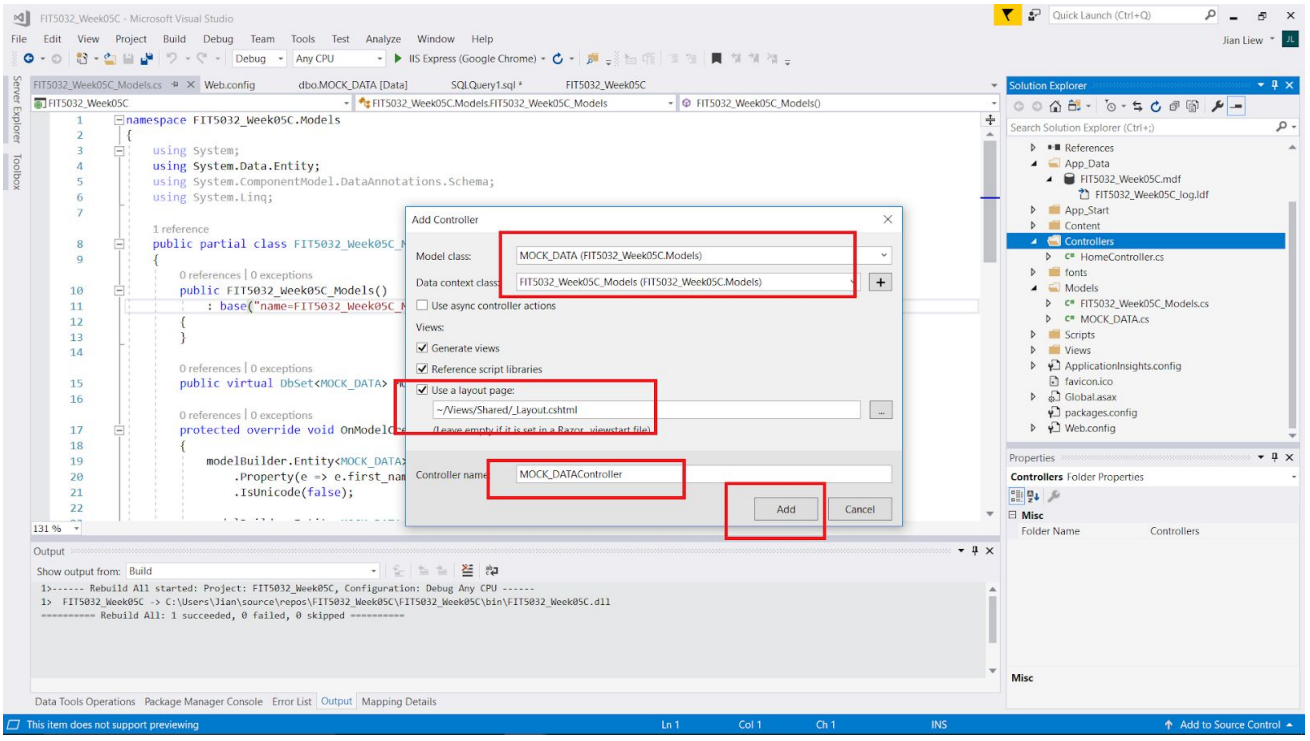


Step 15

Remember that I want both the Controllers and Views.

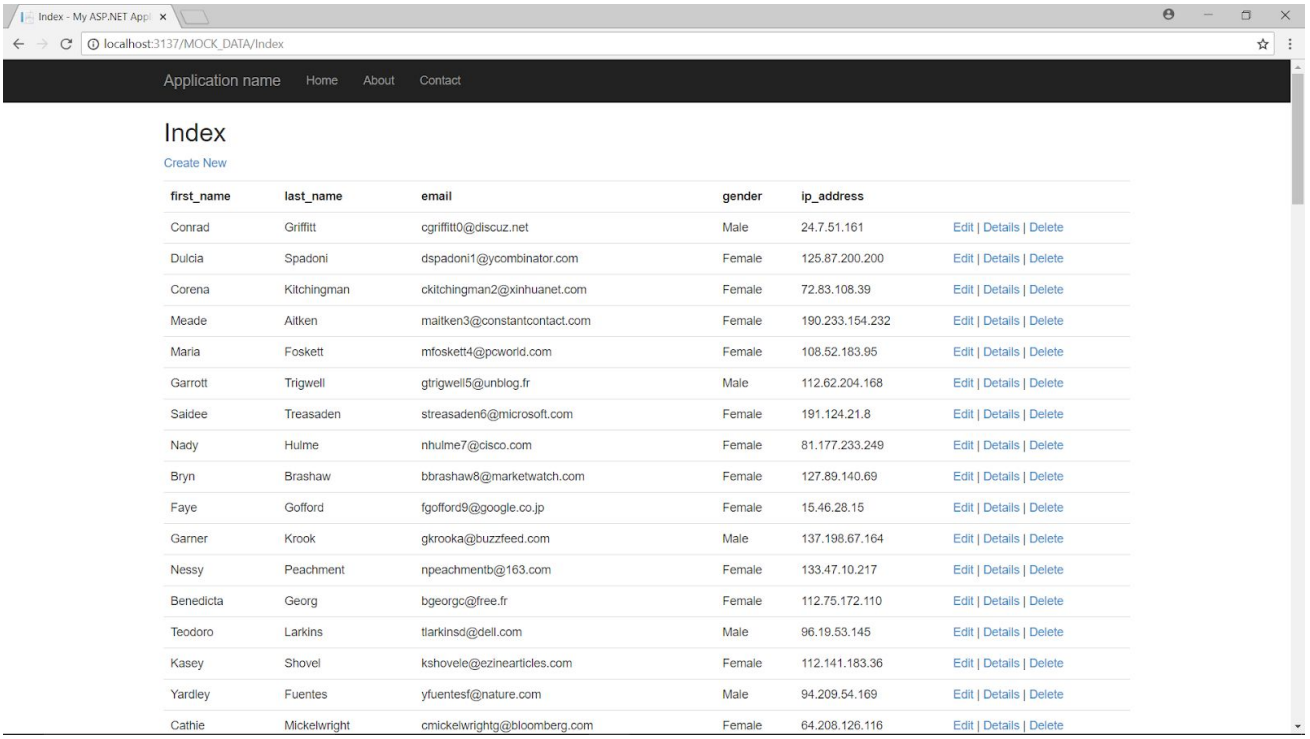


Step 16



Step 17

Now, that we have populated our application with some basic information. We can run and take a look at it. Remember that you can introduce you own hyperlink if you want at the shared layout.

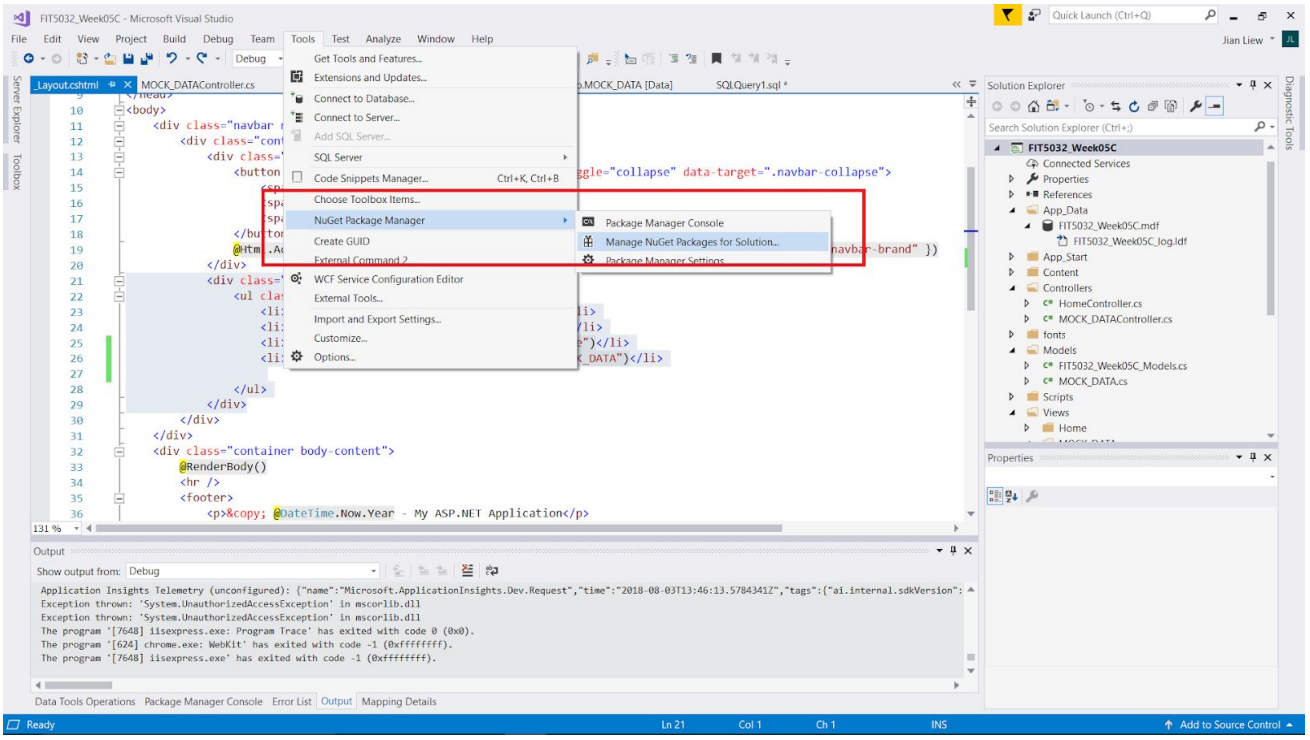


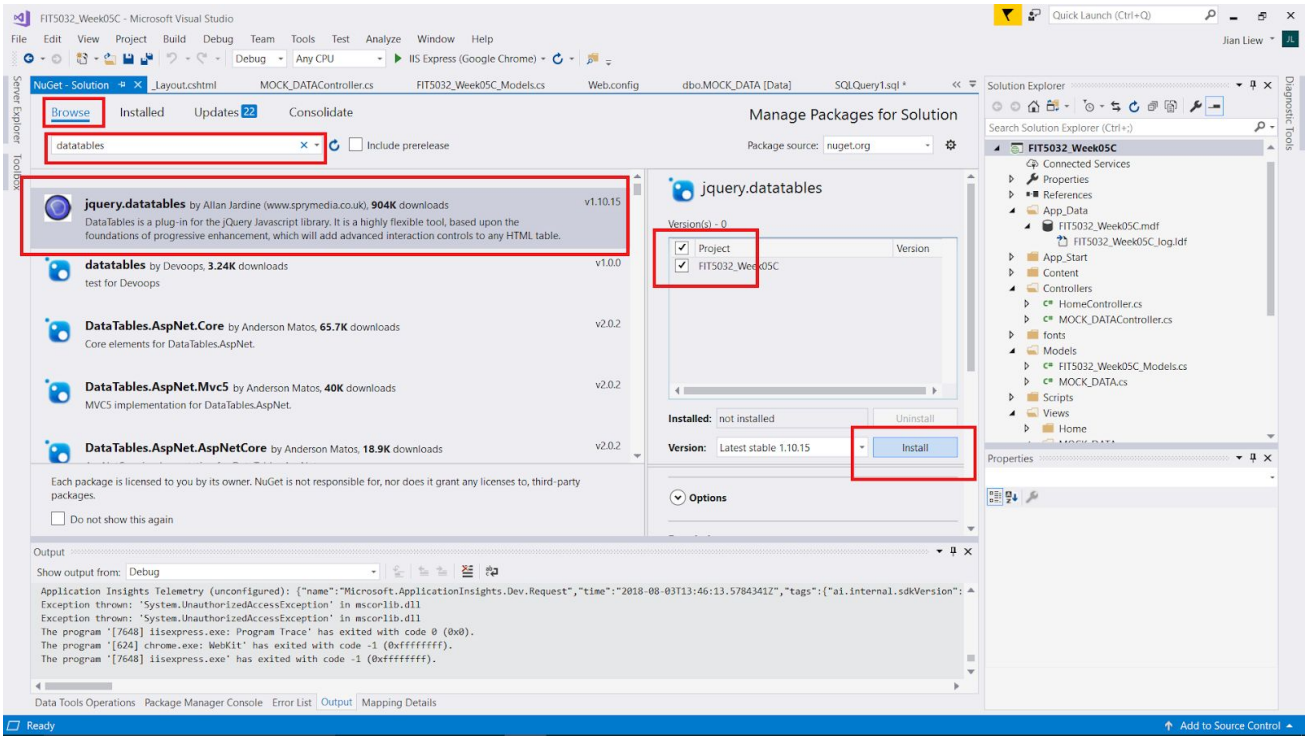
For example, you can introduce a Html ActionLink

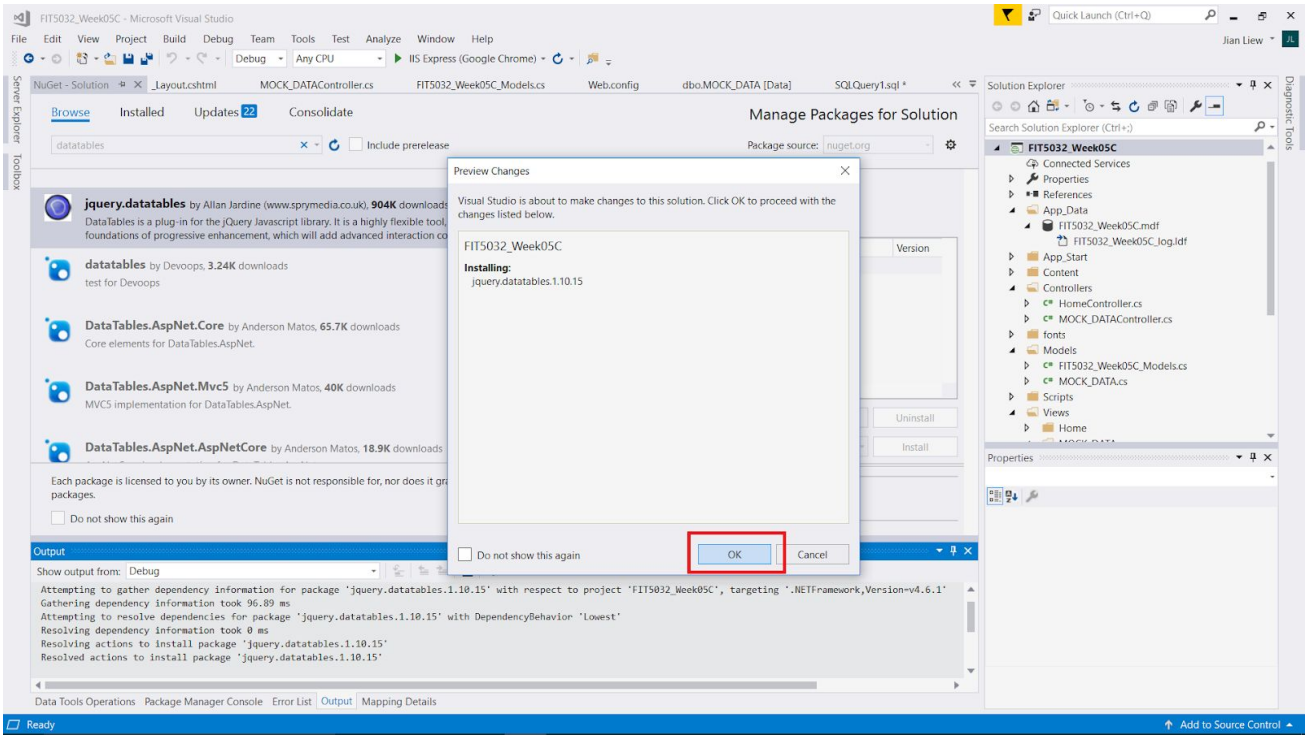
```
<div class="navbar-collapse collapse">
    <ul class="nav navbar-nav">
        <li>@Html.ActionLink("Home", "Index", "Home")</li>
        <li>@Html.ActionLink("About", "About", "Home")</li>
        <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
        <li>@Html.ActionLink("Mock Data", "Index", "MOCK_DATA")</li>
    </ul>
</div>
```

Step 18

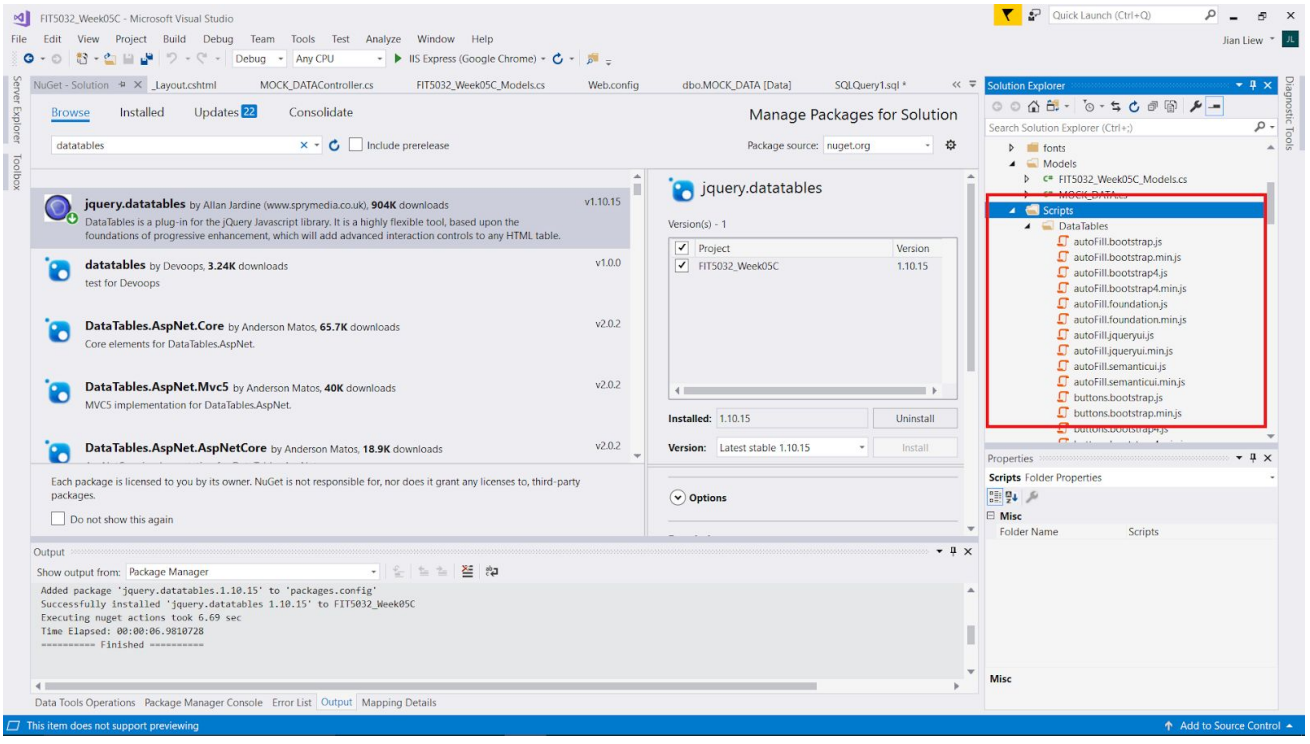
Now, we will begin to add "DataTables" into our application. I will use the GUI to do so.







Say "Yes to All" if you see any warnings.



You should now see that there is a DataTables folder under the Scripts folder.

Step 19

Here, I am going to use a shortcut to load the css and scripts. You should normally bundle them but it can be done this way too.

Copy the follow codes and replace the Index.cshtml of the MOCK_DATA

```
@model IEnumerable<FIT5032_Week05C.Models.MOCK_DATA>

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

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.first_name)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.last_name)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.email)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.gender)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.ip_address)
            </th>
            <th></th>

        </tr>
    </thead>
    <tbody>

        @foreach (var item in Model)
        {
```

```
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.first_name)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.last_name)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.email)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.gender)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.ip_address)
                </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>
        }

    </tbody>
</table>
@section Scripts {

    @Scripts.Render("~/Scripts/DataTables/jquery.dataTables.js")
    @Scripts.Render("~/Scripts/DataTables/dataTables.bootstrap.js")

    <script>
        $(document).ready(function () {
            $('.table').DataTable();
        });
    </script>

}
<link href="~/Content/DataTables/css/dataTables.bootstrap.min.css"
rel="stylesheet" />
```

Explanation

Here, I took a few shortcuts, I introduced a section called scripts and loaded the needed datatables JavaScript file. Normally this should be in a bundle. I also loaded the css file towards the end. You can also load these files via CDN if you really want.

In order for datatables to work, here is the general concept. You can read their documentation more if you want.

- The JavaScript file and CSS needs to be loaded.
- The table needs to have a thead and tbody element.
- The DataTable function needs to be called on the correct class.

Things that you can improve upon

- First, you can introduce an Attribute at the Model to make the display nicer.
- I used a super simplistic database design. You can improve upon this.
- You can also introduce more configuration at the DataTables as I am currently using it a zero configuration.
- **You can figure out a better way to do the pagination. Currently the way the pagination is done, is by loading all the data at once. This is actually a really bad practice. But since there is only 100 rows at the moment, it is acceptable. However, it would be better if pagination is actually done correctly via server side loading. If you really want to see why, you can generate more mock data and see why it is so. 10000 rows for example and see how fast it loads.**
- You can introduce validations to the models too.
- You can also introduce bootstrap elements within the tables itself like introducing button classes at the create, edit and delete links.

Conclusion

Upon the completion of this supplementary material, you would gain a basic understanding of

- How to use the NuGet package manager to obtain packages.
- How to generate mock data with Mockaroo.
- How to load JavaScript files (the easy way). There is a disadvantage doing it this way though. What do you think it is?
- Why does loading 10,000 rows or more this way is considered to be bad practice.