MONASH University

FIT5032 - Internet Applications Development

Week 04 - Introduction To Entity Framework

Last updated: 6th May 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.**

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.**

Objectives

Estimated Time To Complete : ~ 1 hours
Upon the completion of this tutorial, you will gain a basic understanding of

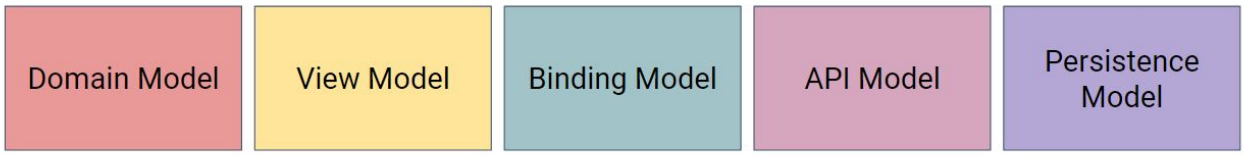
- The different approaches to using the Entity Framework
- Model First Approach
- Database First Approach
- Code First Approach

DoubtFire Submission

- T4.1 A document detailing your prefered approach when using the Entity Framework.
- T4.2 [Updated 09/08/2017] A document showing one of your week 4 database applications running. (in a PDF document)

What is a Model?

Before we venture into the Entity Framework (EF), there is a need to understand the different models. Remember that there are several types of model.



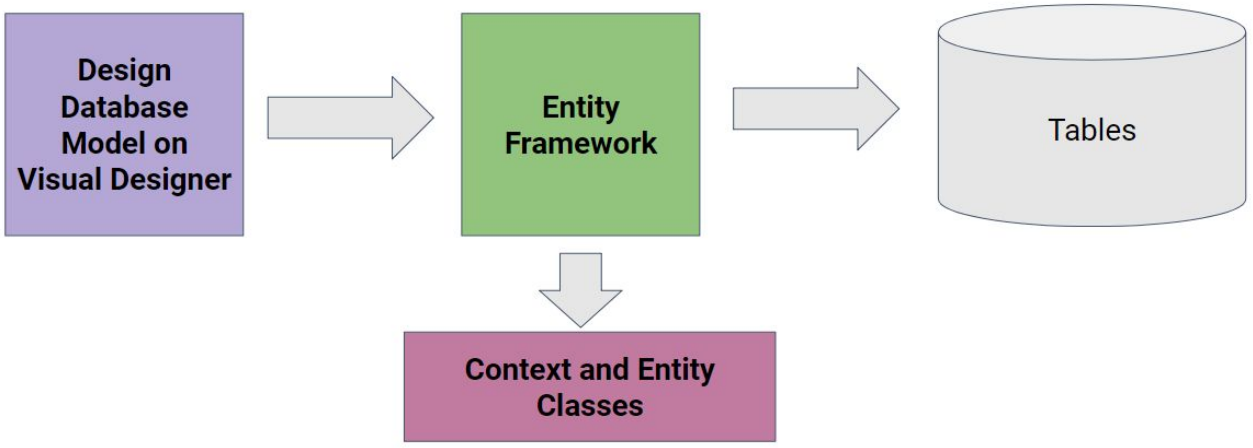
If you have created the project based on the suggested way, you would have seen what the ViewModel is. Normally it is good practice to have each form at the front end to have a backing ViewModel. However, it is also possible not to have a ViewModel. You will learn more regarding ViewModels on Week 6. In a sense, a ViewModel is a class designed to store and manage UI related data.

What we will try to achieve in this lab is a simple Persistence Model. (In a way what we try to persist information into the database)

The API model will be shown later in the semester as this is the creation of what is know as a WebAPI.

For the labs, we will be using what is known as the LocalDB. This, is normally only used during the development environment, if you application is deployed, there is a need for a real database connection.

The Model First Approach

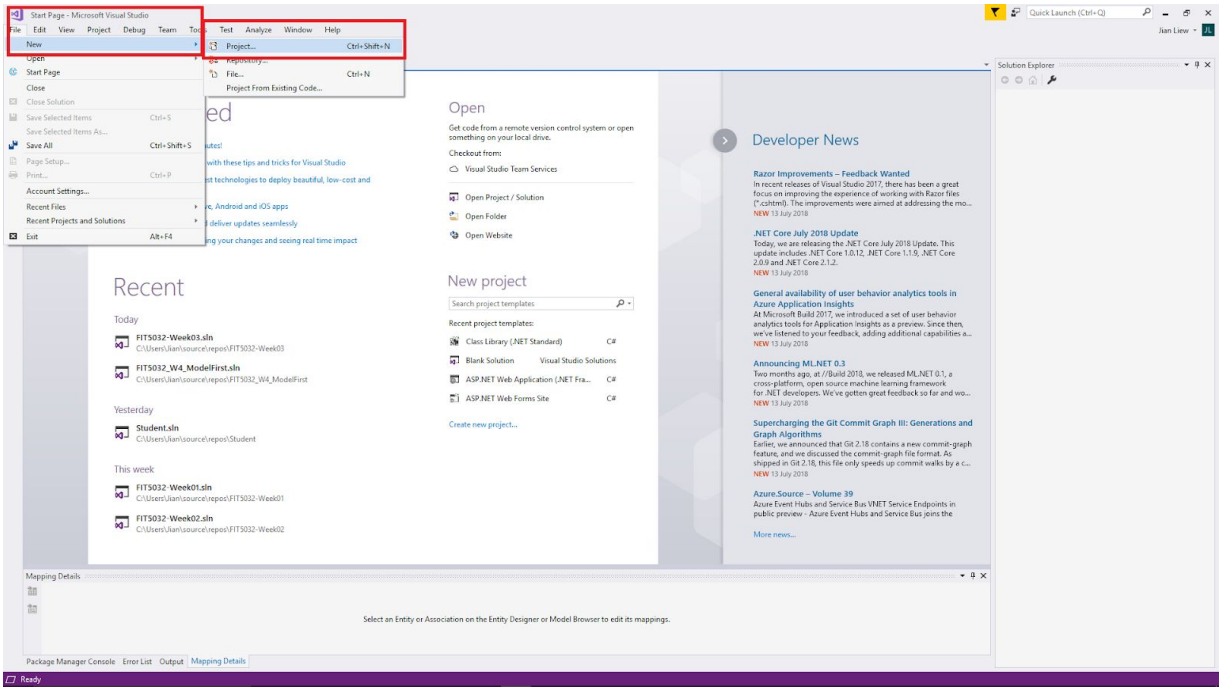


The Model First Approach when using Entity Framework

In the model first approach, the **visual designer** will be used to design the model layer. The database will then be engineered based on the design on the visual designer using the Entity Framework. This will automatically generate all the needed entity classes and set up the correct context. It will also create the needed tables in the database. (Please take note that the Model First approach is often times confused as Database First, but this is not the case based on this tutorial)

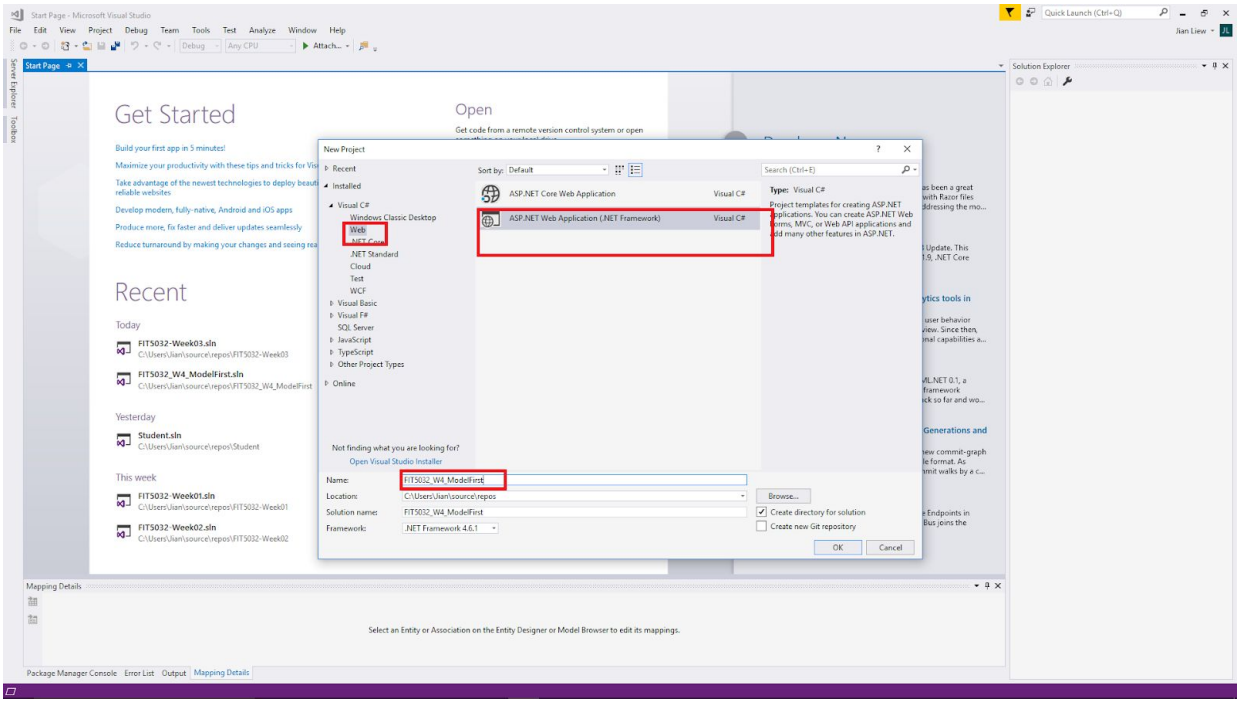
Please remember that these documents are structured in way it is possible to zoom into the images. When you are following the steps, use the correct casing. (Uppercase and lower cases, follow the document precisely)

Step 1



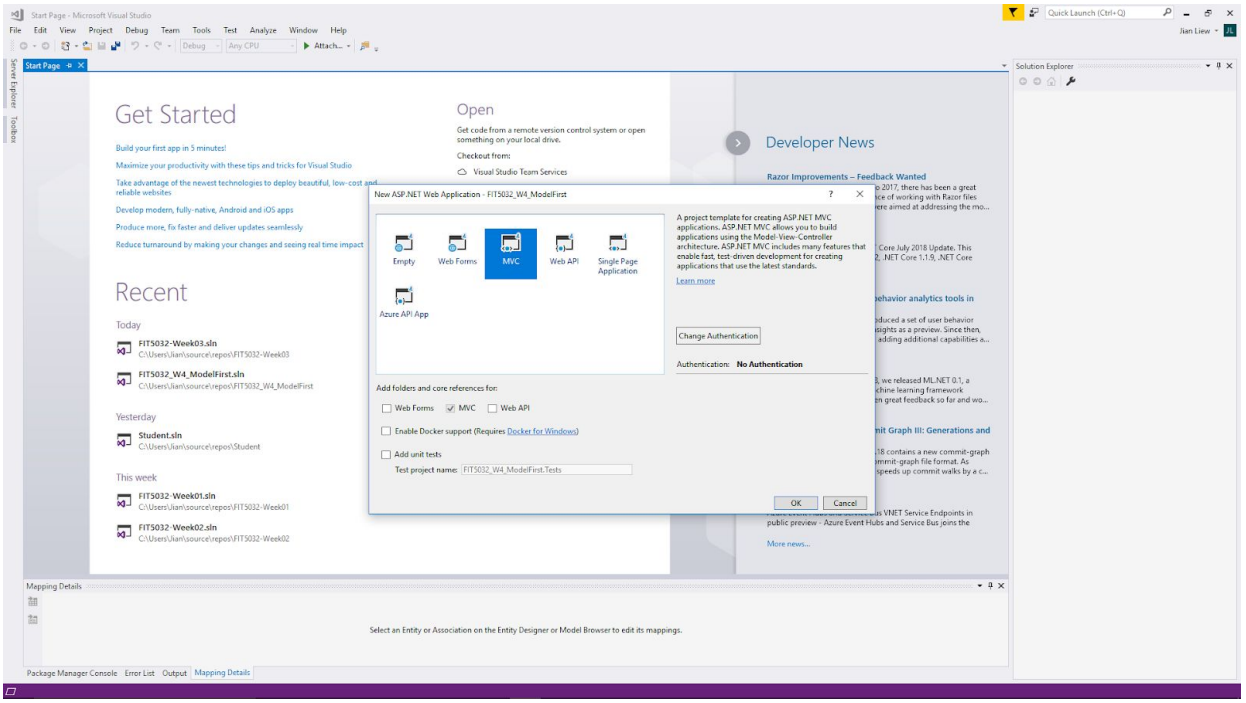
Create a new Project

Step 2

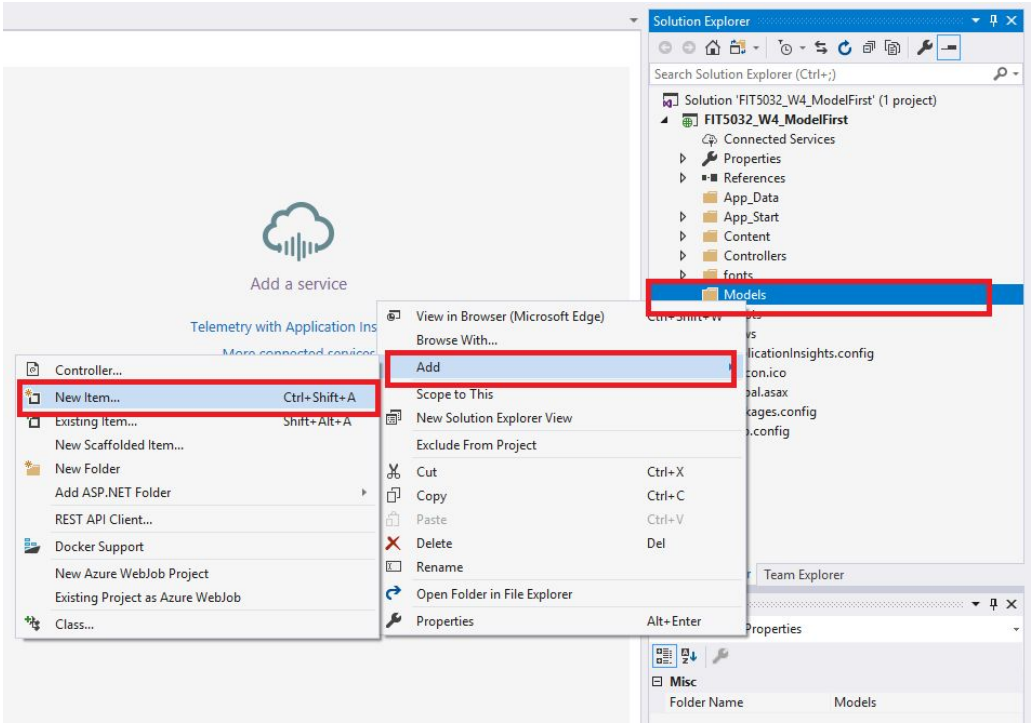


Name: **FIT5032_W4_ModelFirst**

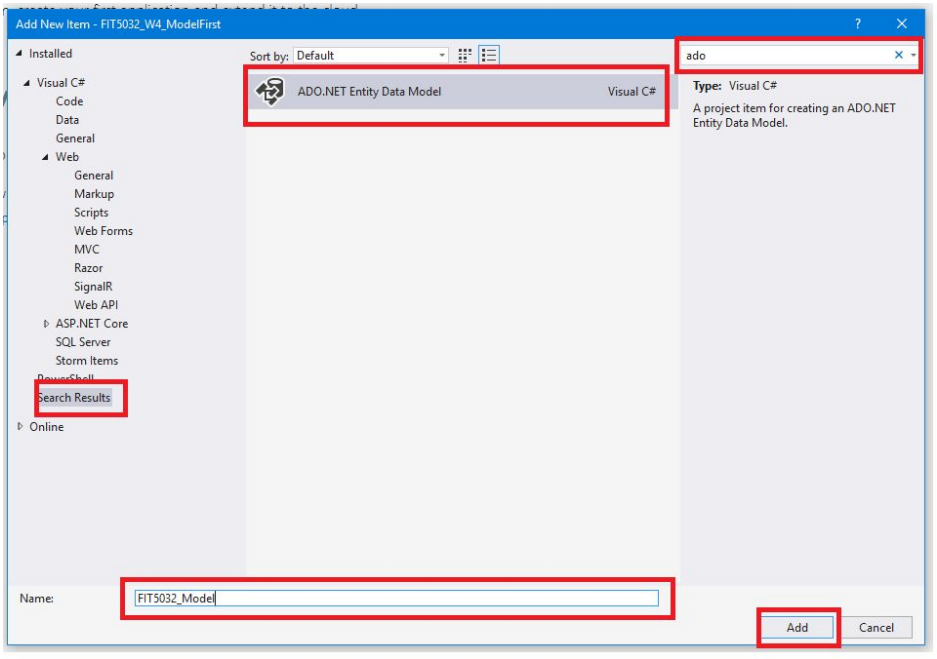
Step 3



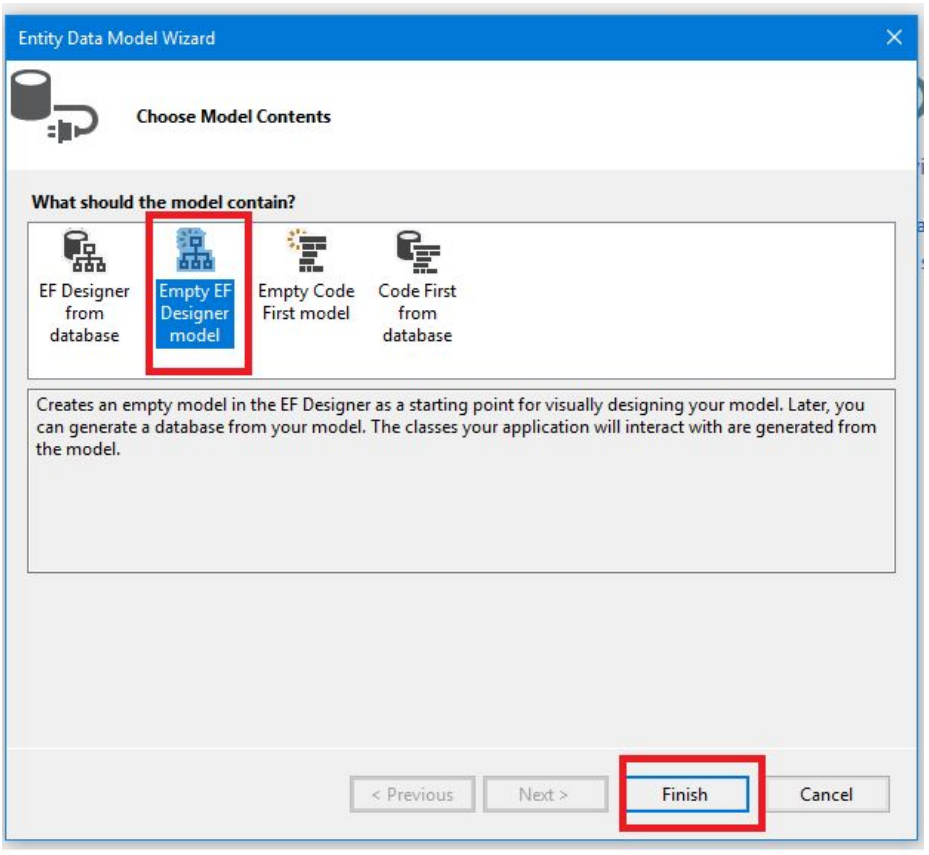
Step 4


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'FIT5032_W4_ModelFirst' (1 project)
FIT5032_W4_ModelFirst
Connected Services
Properties
References
App_Data
App_Start
Content
Controllers
fonts
Models
Add a service
Telemetry with Application Ins
Controller...
New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A
New Scaffolded Item...
New Folder
Add ASP.NET Folder
REST API Client...
Docker Support
New Azure WebJob Project
Existing Project as Azure WebJob
Class...
View in Browser (Microsoft Edge)
Browse With...
Add
Scope to This
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
Team Explorer
Misc
Folder Name
Models

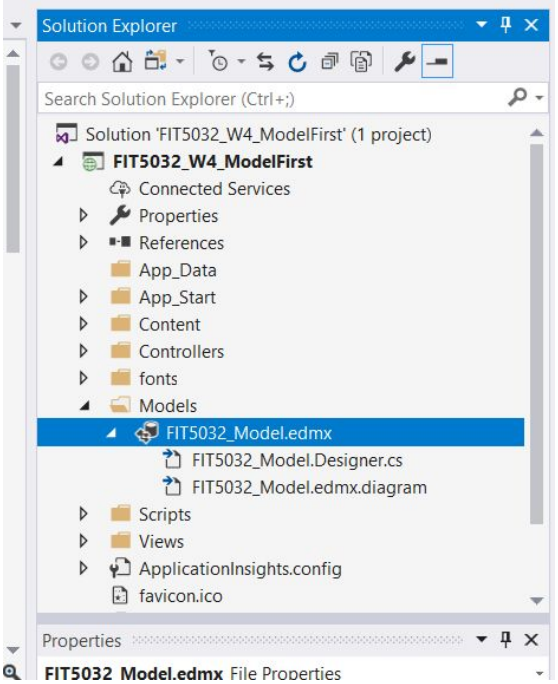
Step 5


Add New Item - FIT5032_W4_ModelFirst
Installed
Visual C#
Code
Data
General
Web
General
Markup
Scripts
Web Forms
MVC
Razor
SignalR
Web API
ASP.NET Core
SQL Server
Storm Items
Search Results
Online
Sort by: Default
ADO.NET Entity Data Model Visual C#
ado
Type: Visual C#
A project item for creating an ADO.NET Entity Data Model.
Name: FIT5032_Model
Add
Cancel

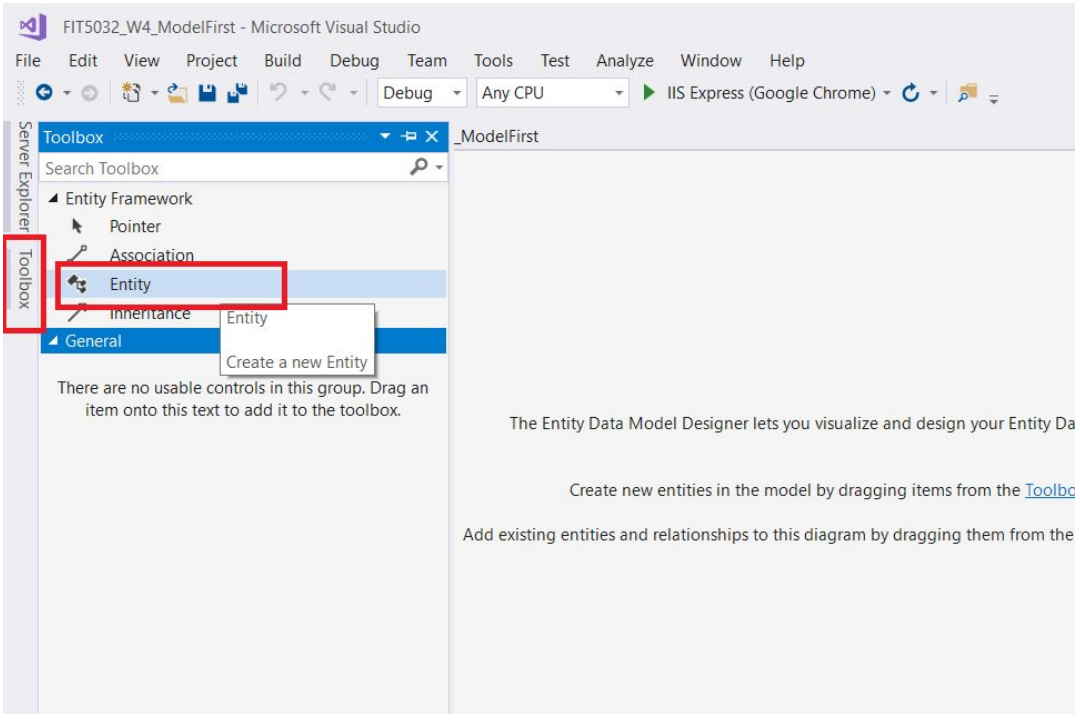
Step 6


Entity Data Model Wizard
Choose Model Contents
What should the model contain?
EF Designer from database
Empty EF Designer model
Empty Code First model
Code First from database
Creates an empty model in the EF Designer as a starting point for visually designing your model. Later, you can generate a database from your model. The classes your application will interact with are generated from the model.
< Previous
Next >
Finish
Cancel

Step 6

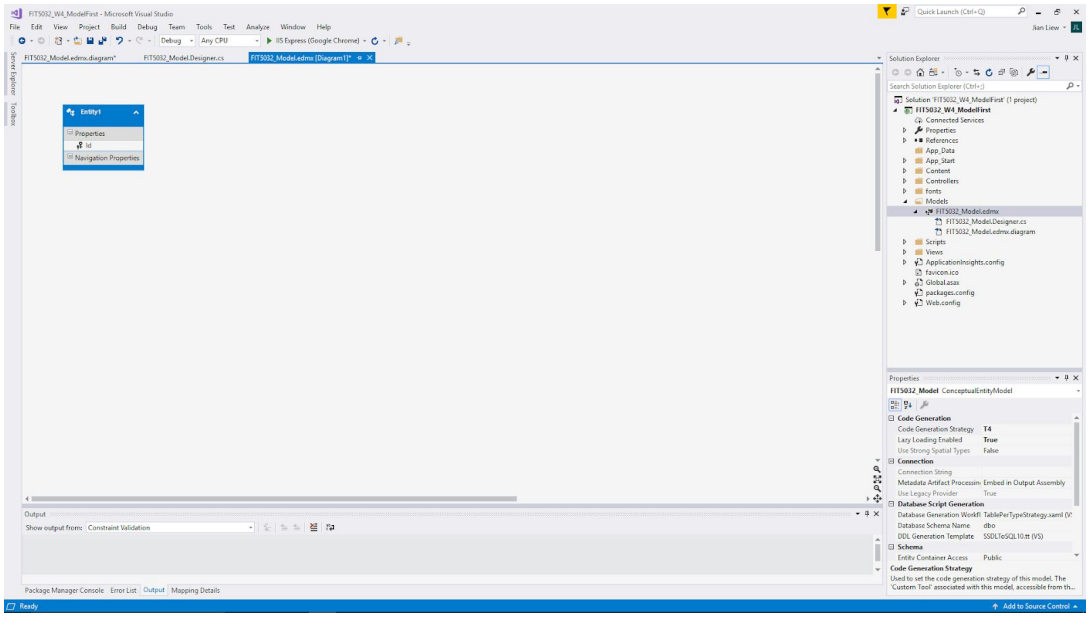

Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'FIT5032_W4_ModelFirst' (1 project)
FIT5032_W4_ModelFirst
Connected Services
Properties
References
App_Data
App_Start
Content
Controllers
fonts
Models
FIT5032_Model.edmx
FIT5032_Model.Designer.cs
FIT5032_Model.edmx.diagram
Scripts
Views
ApplicationInsights.config
favicon.ico
Properties
FIT5032_Model.edmx File Properties

Step 7



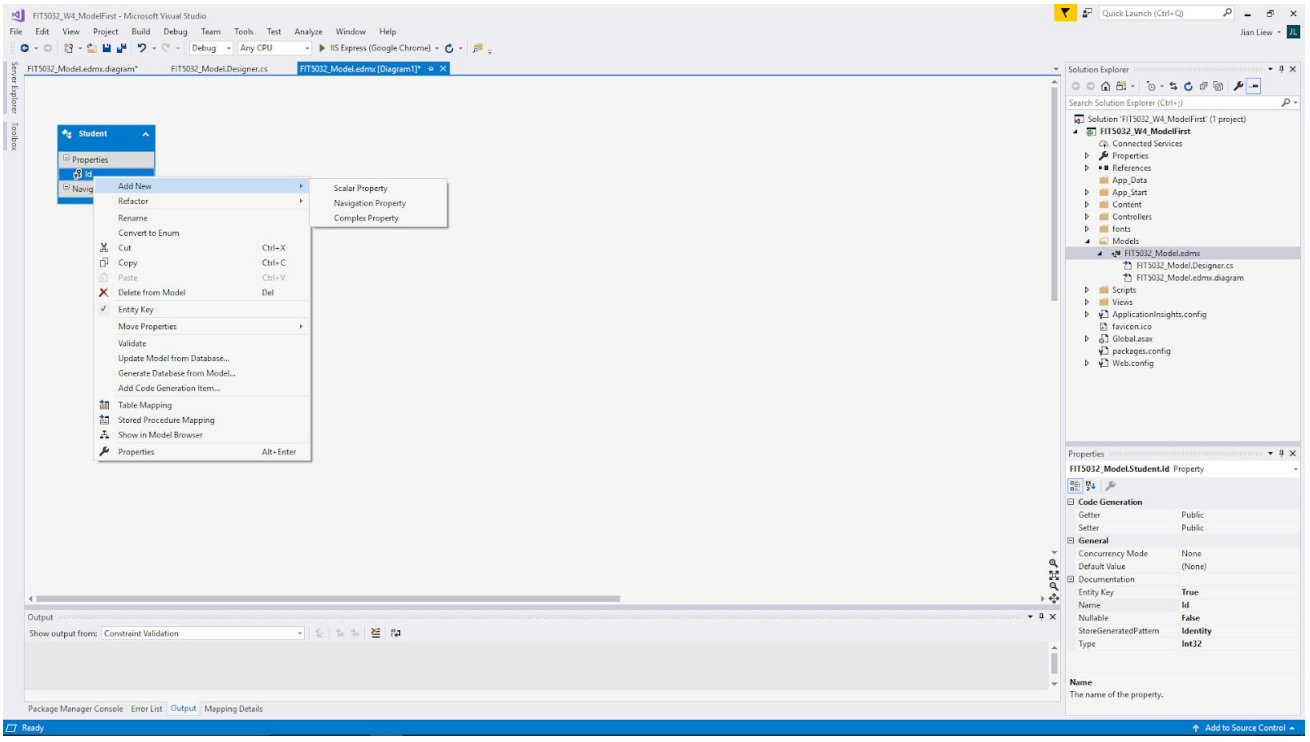
Step 8

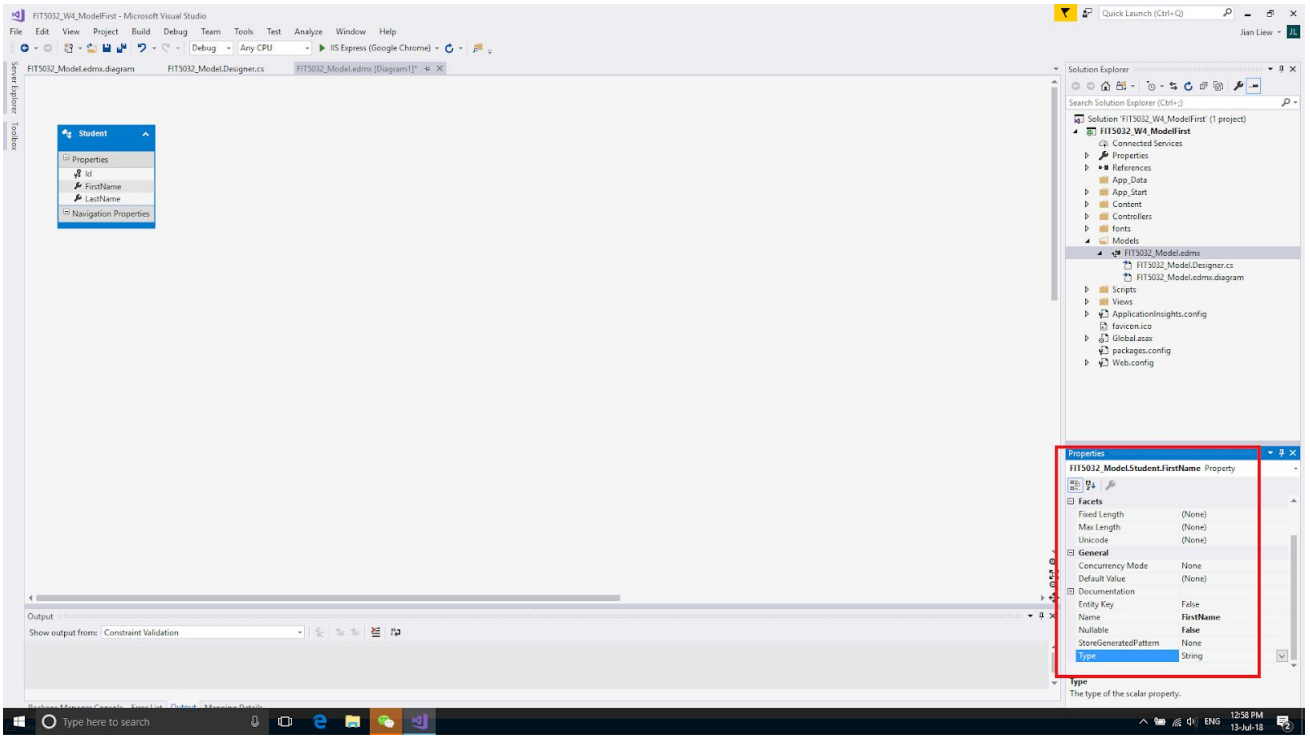
You can double click on the "Entity" option and an Entity will be created for you.



Step 9

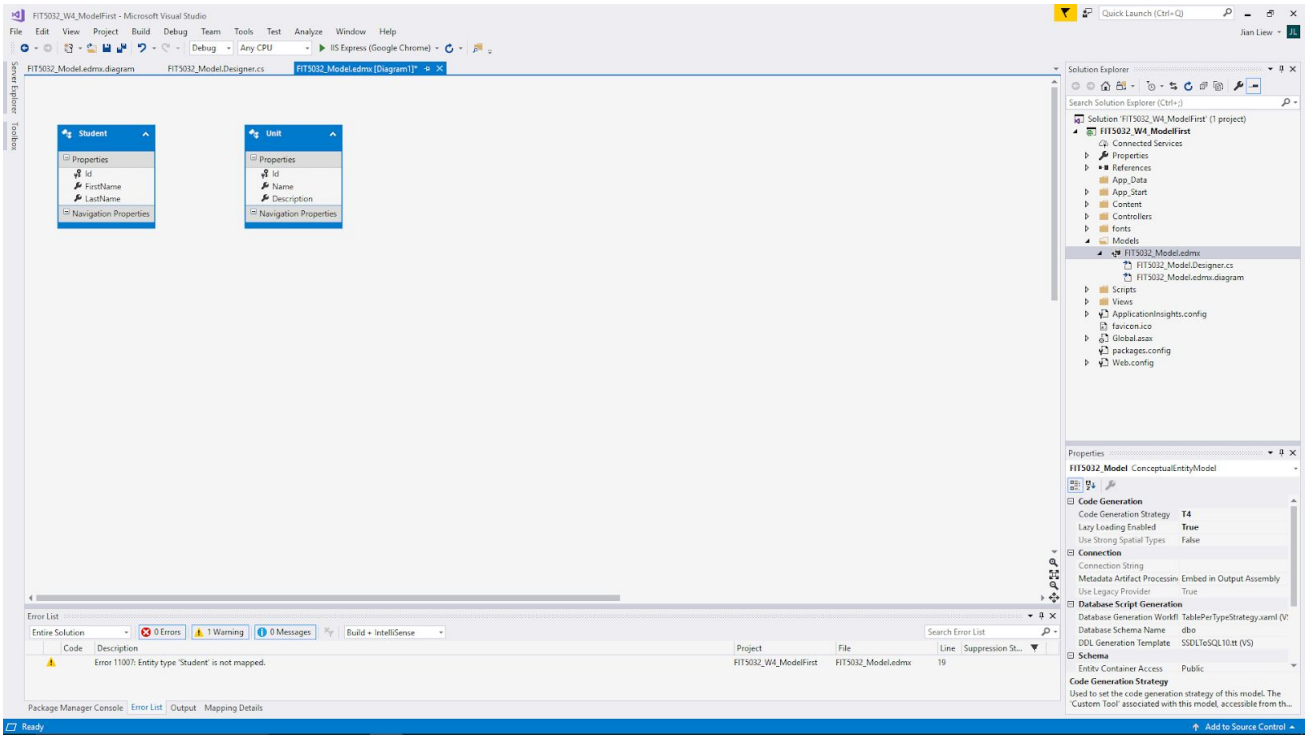
You can also add a new "Scalar" property. Here, I would rename my entity to be Student and introduce two scalar property (FirstName and LastName).



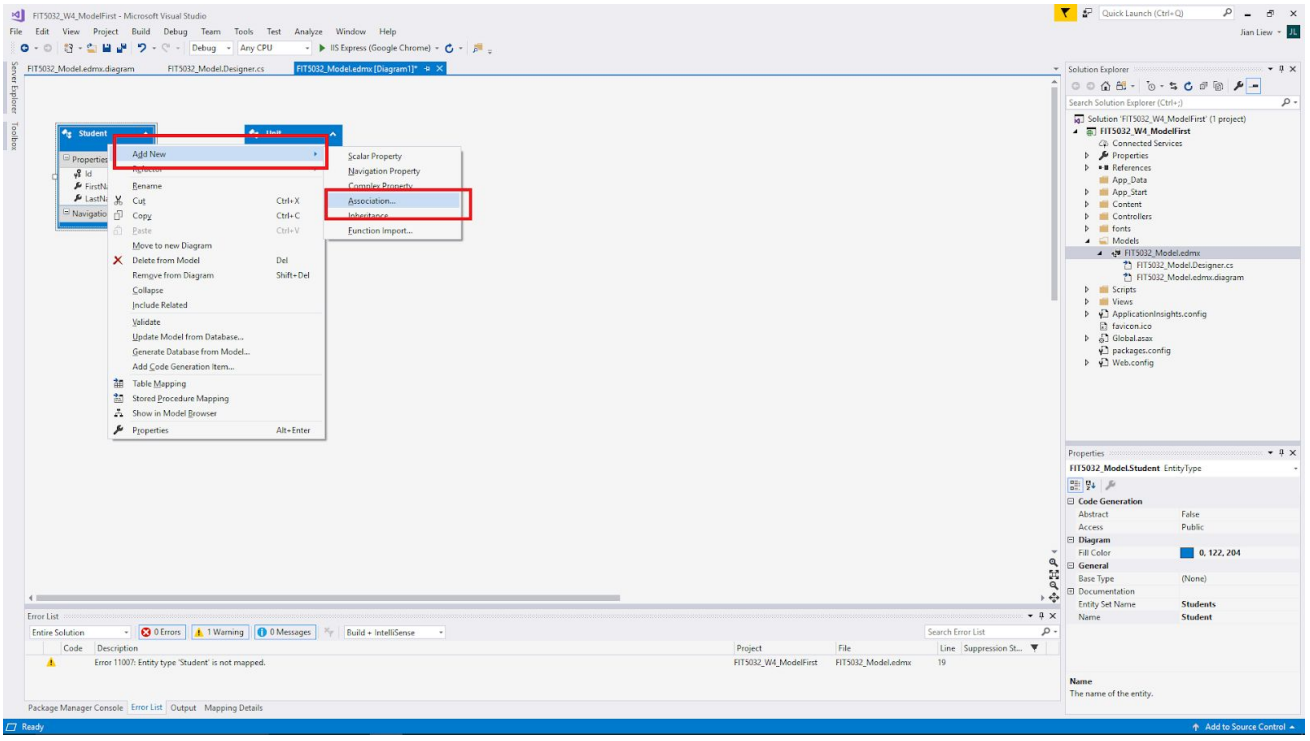


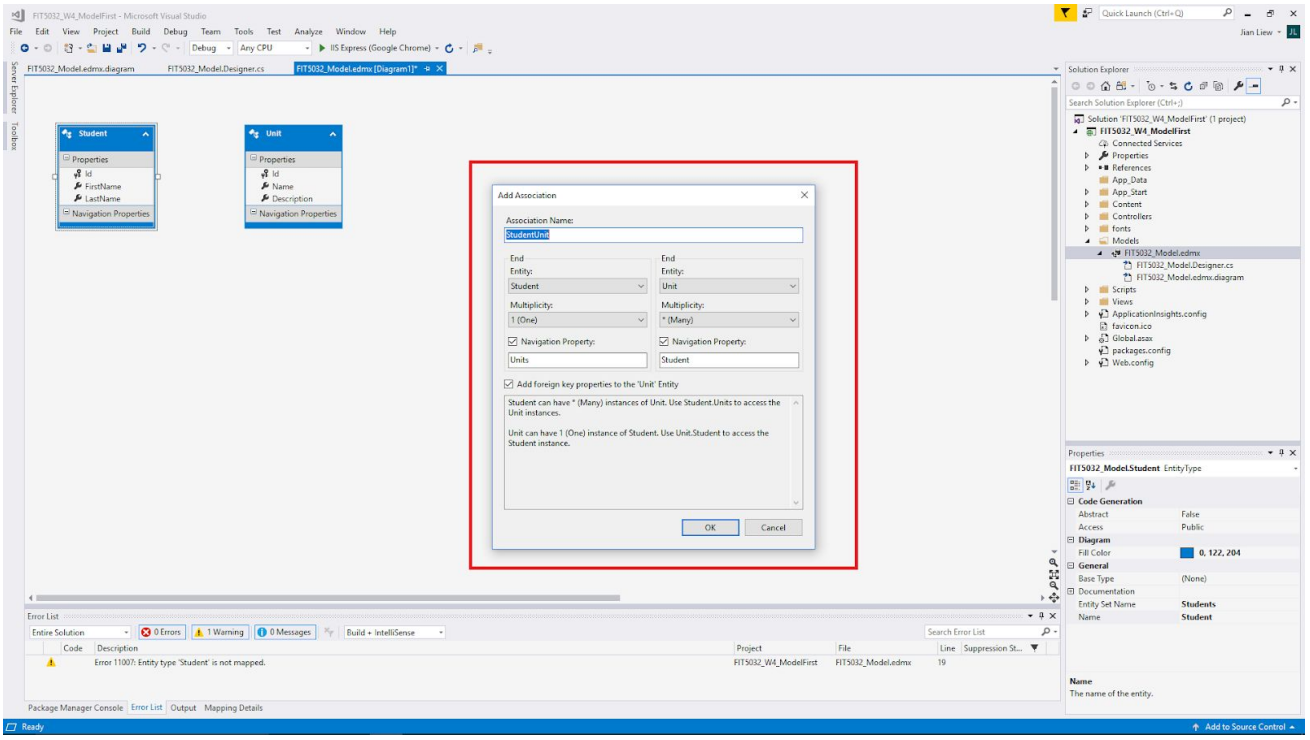
Step 10

You can of course create an association with this entity. But before that create another entity called unit. A unit has a Name as well as a Description.

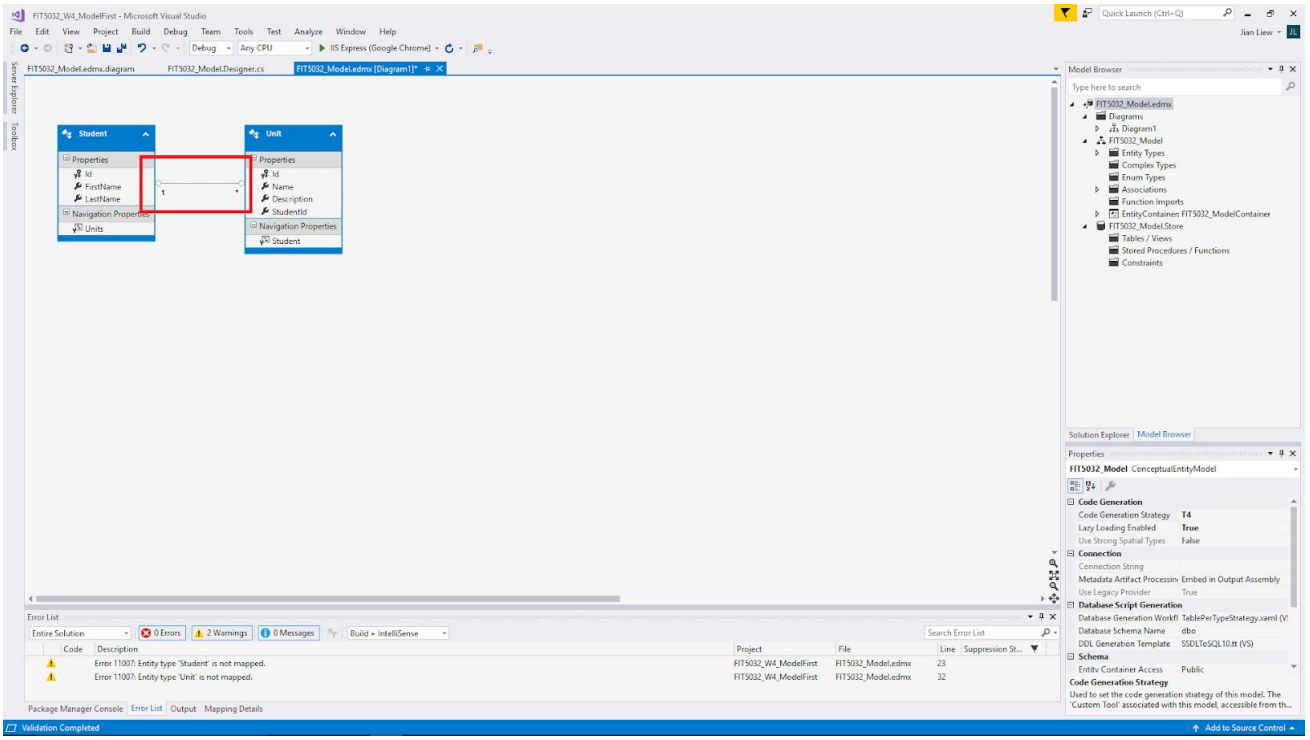


After that, create an association between them. Right click at an empty spot and…





You can change the configuration of this associate as if wish. Please remember that this is a class diagram. So, modelling skills play a very important role here.

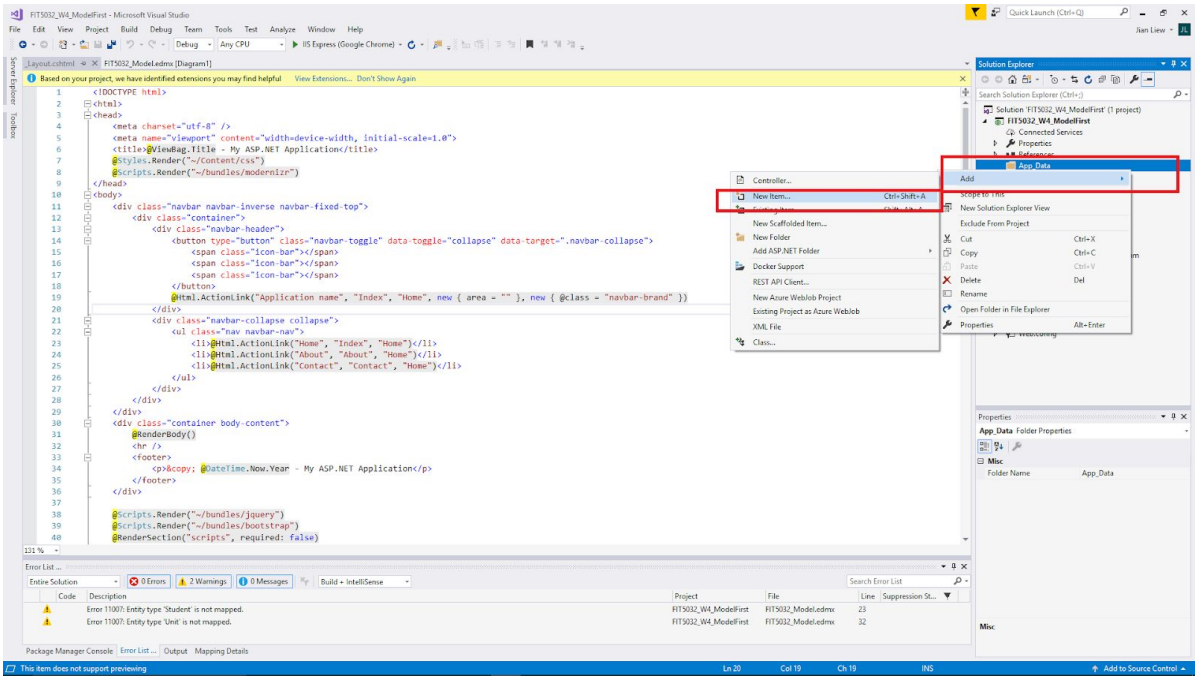


You will now notice that there is an association formed between these two entities.

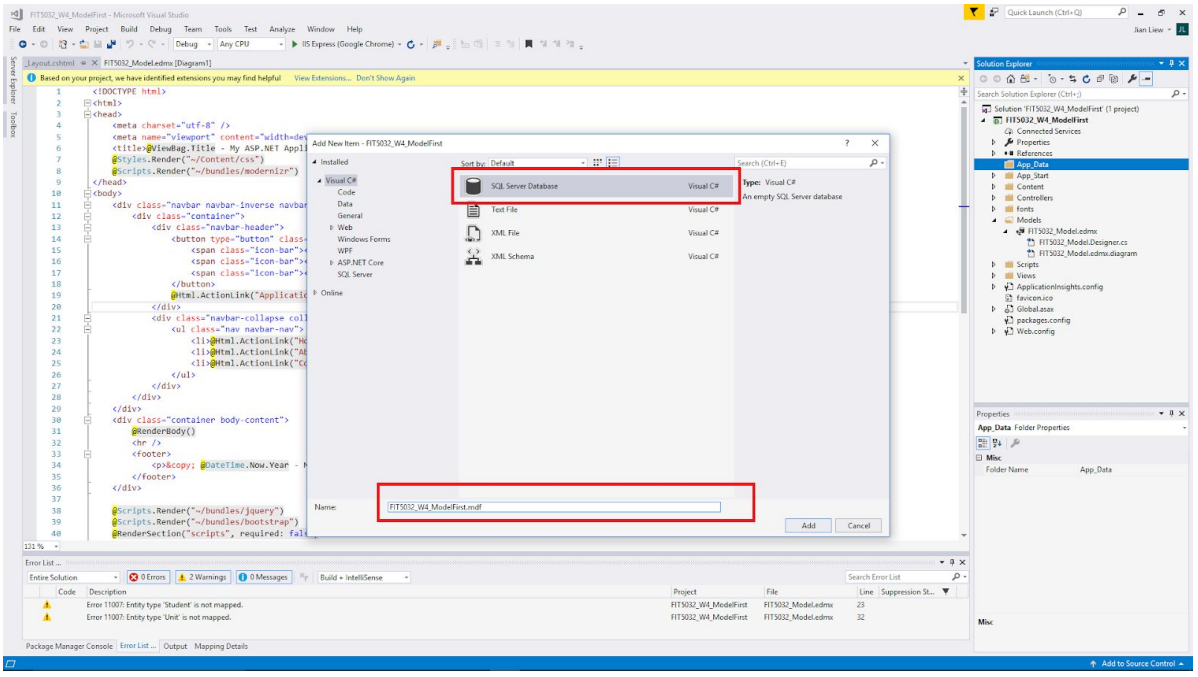
Step 10

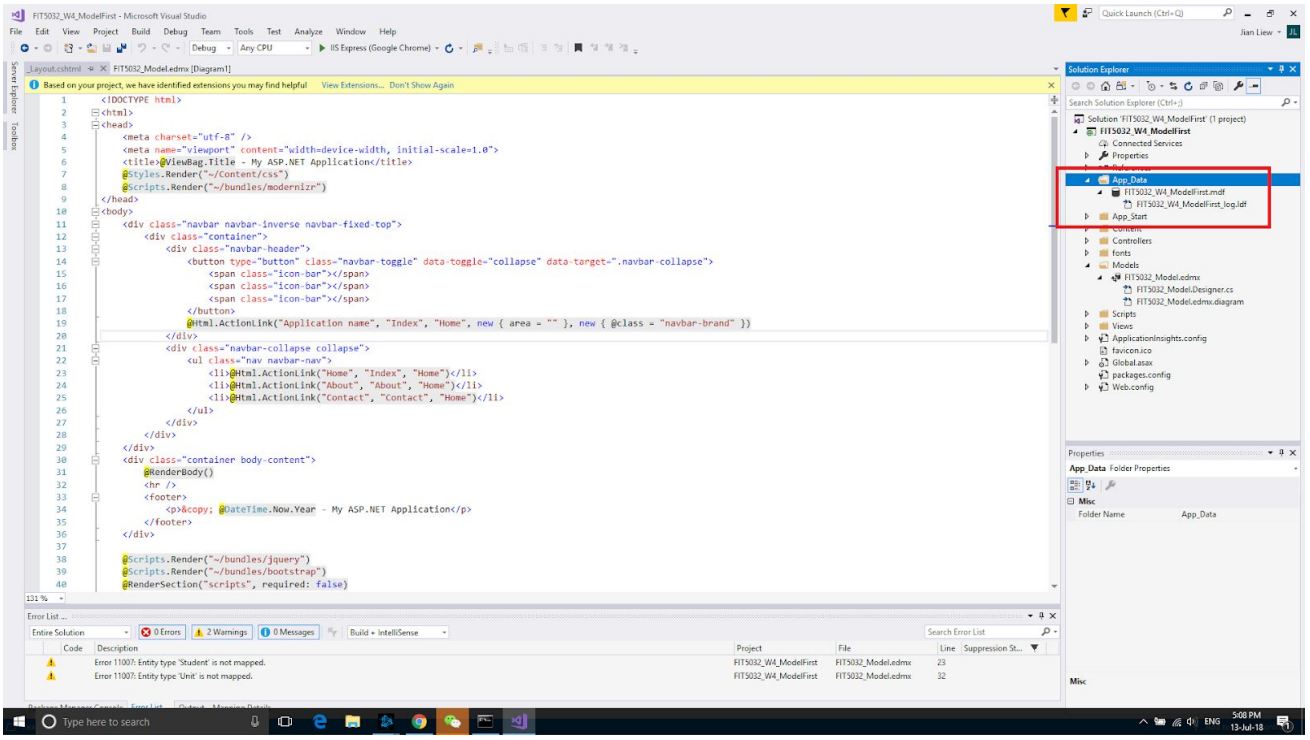
Now after we have created the entities what we want is to generate the database from the model.

Normally we would connect to an existing database somewhere but in Visual Studio we can use a .mdf file. This is an embedded database file. We have only so far created the models, but we want the EF to create the tables for us based on the models we did.

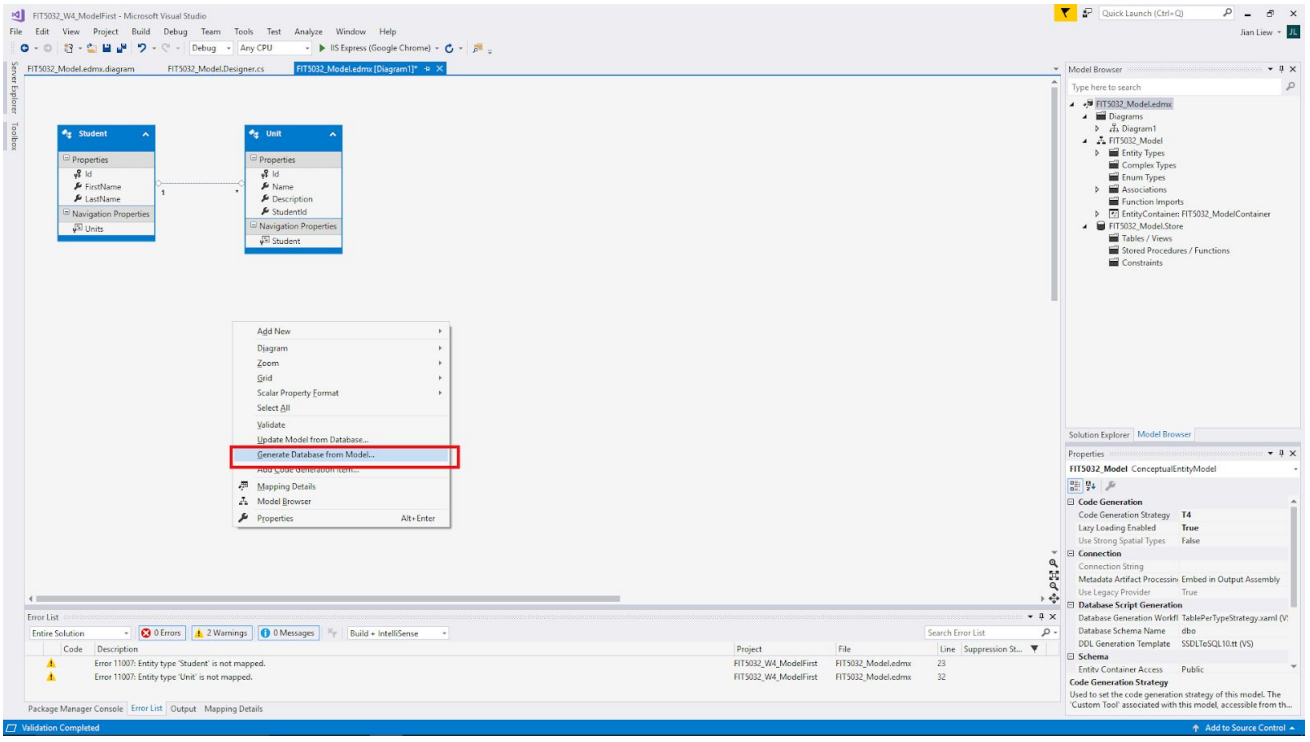


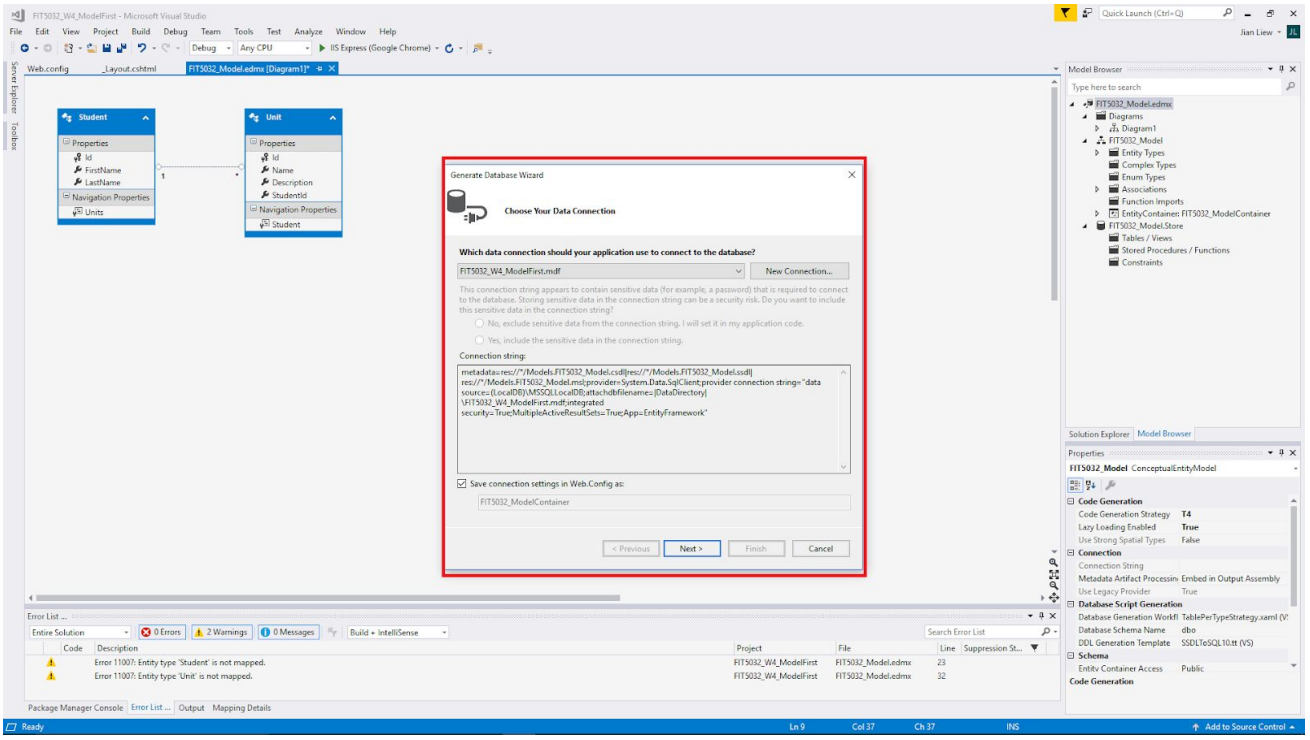
Right Click, the App_Data folder and click Add → New Item

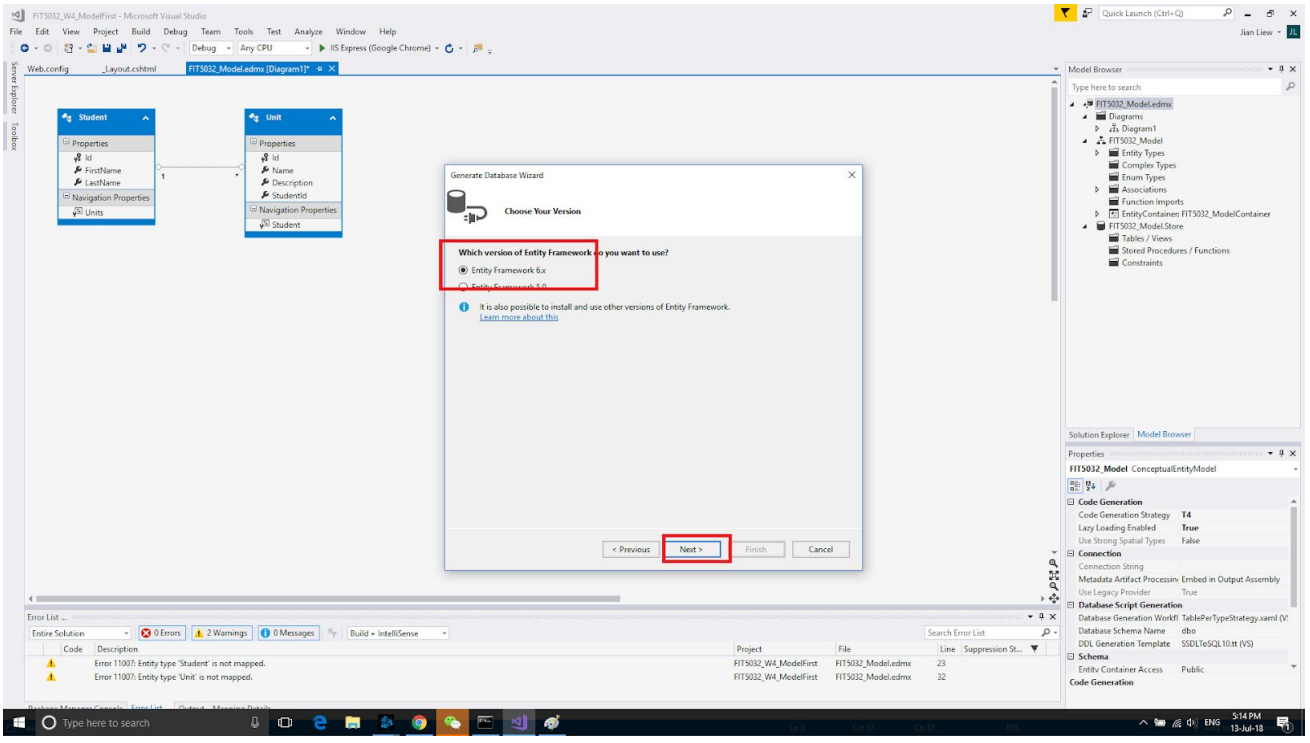


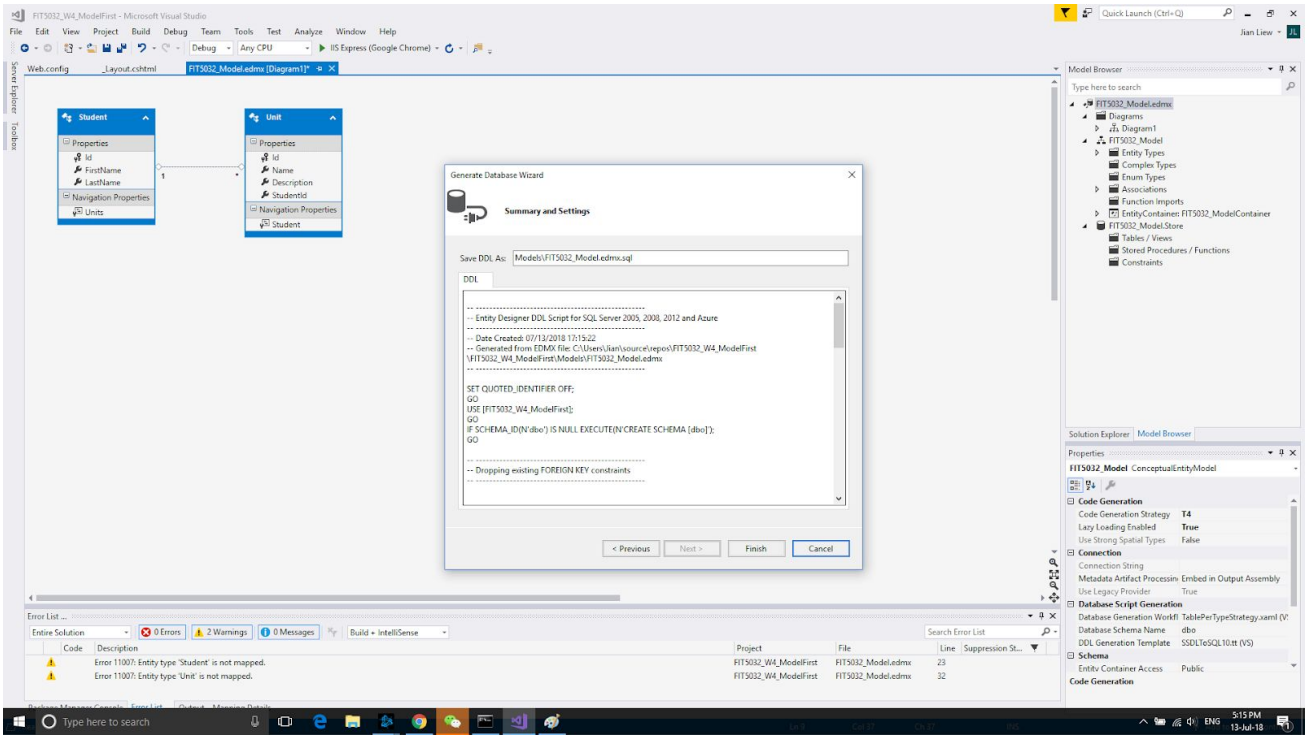


Notice that the .mdf file will be created.



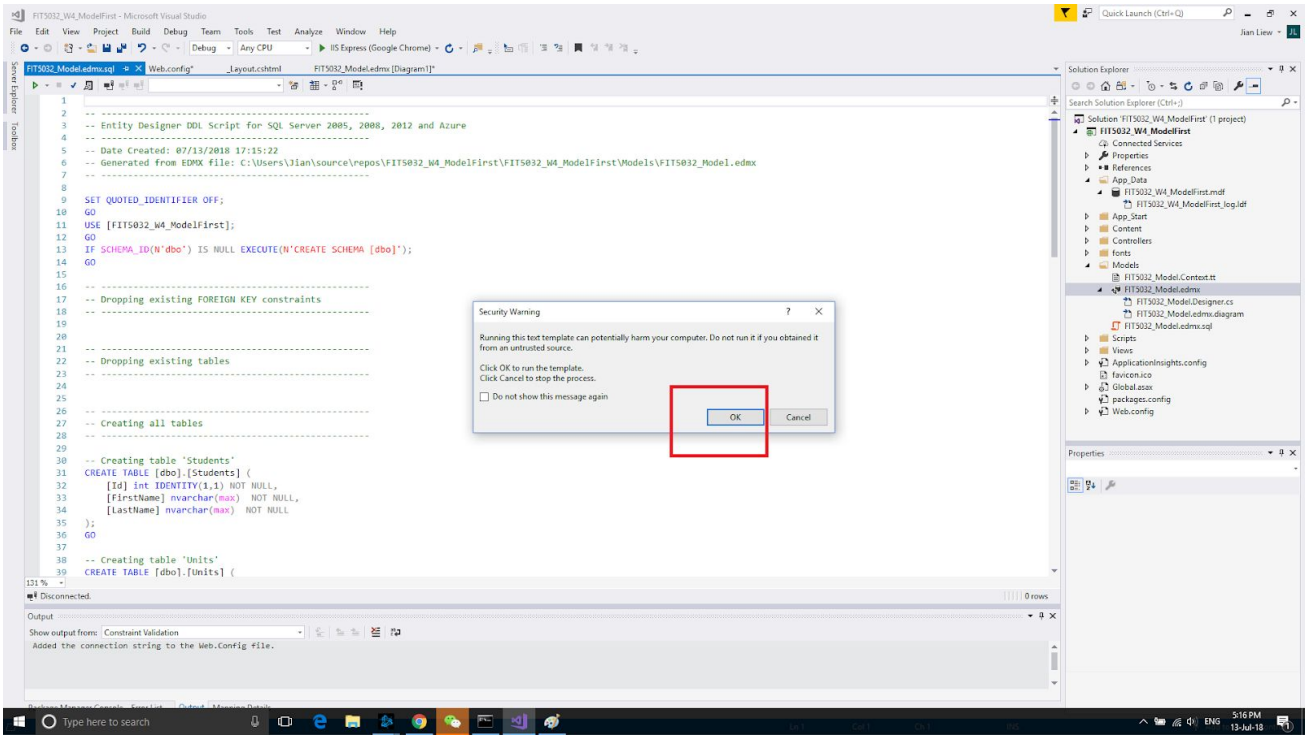

Generate Database Wizard
Choose Your Data Connection
Which data connection should your application use to connect to the database?
FIT5032_W4_ModelFirst.mdf
New Connection...
Connection string:
Save connection settings in Web.Config as:
FIT5032_ModelContainer
< Previous
Next >
Finish
Cancel


Generate Database Wizard
Choose Your Version
Which version of Entity Framework do you want to use?
Entity Framework 6.x
It is also possible to install and use other versions of Entity Framework.
Learn more about this
< Previous
Next >
Finish
Cancel

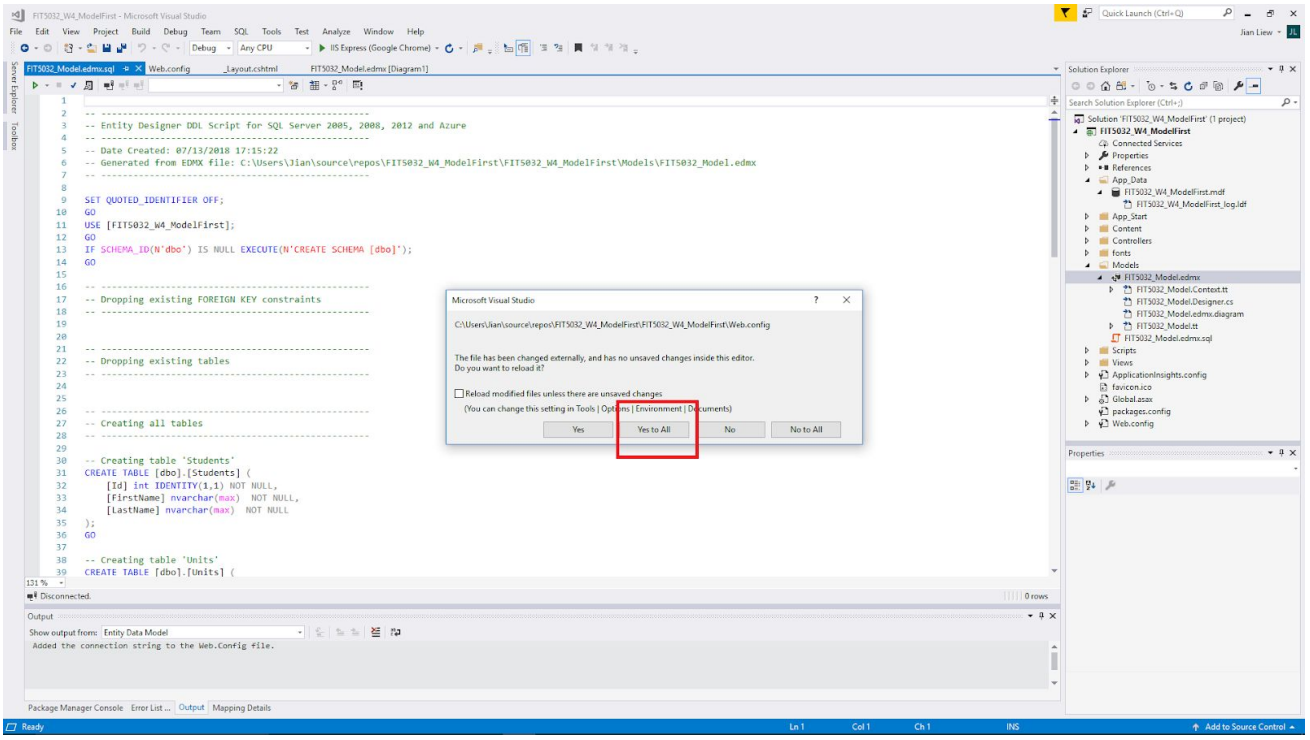


Visual Studio will now automatically create the **data definition language (DDL)** for the create table queries. This is the SQL query to create the needed tables.

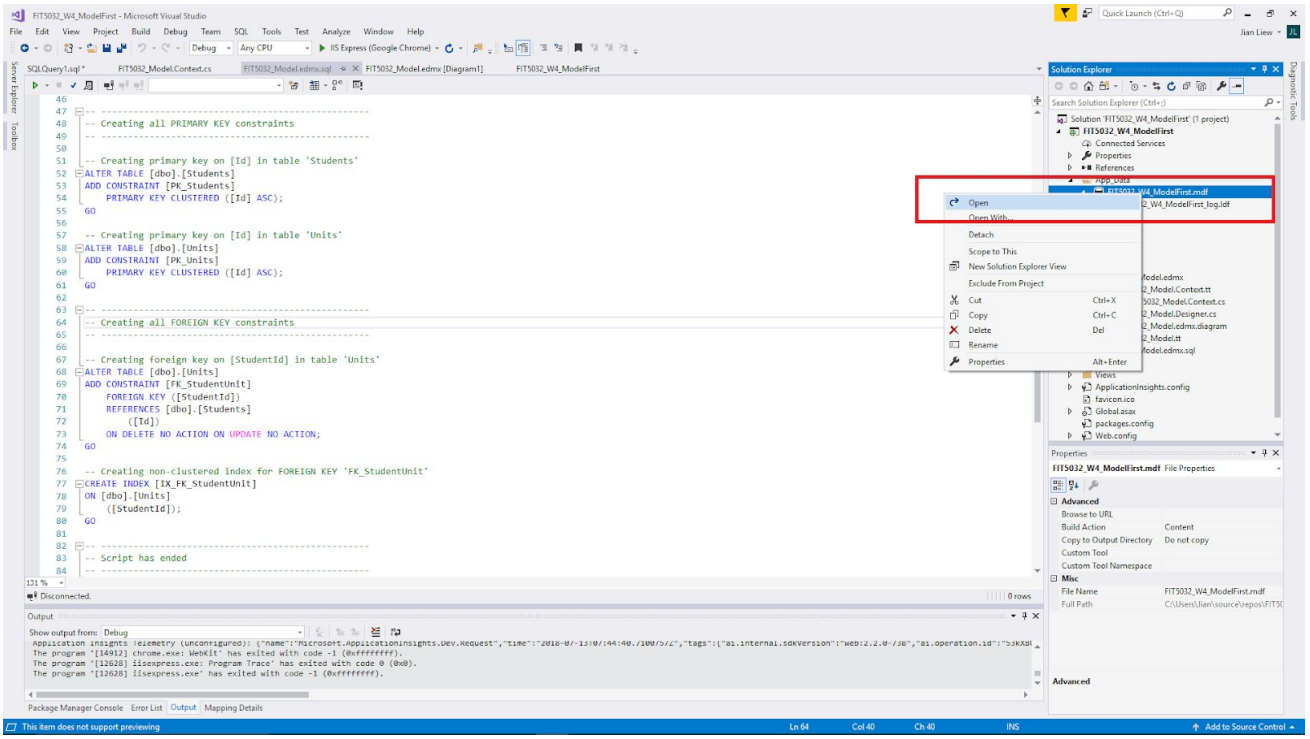
Click Finish and click OK to any "Warnings"

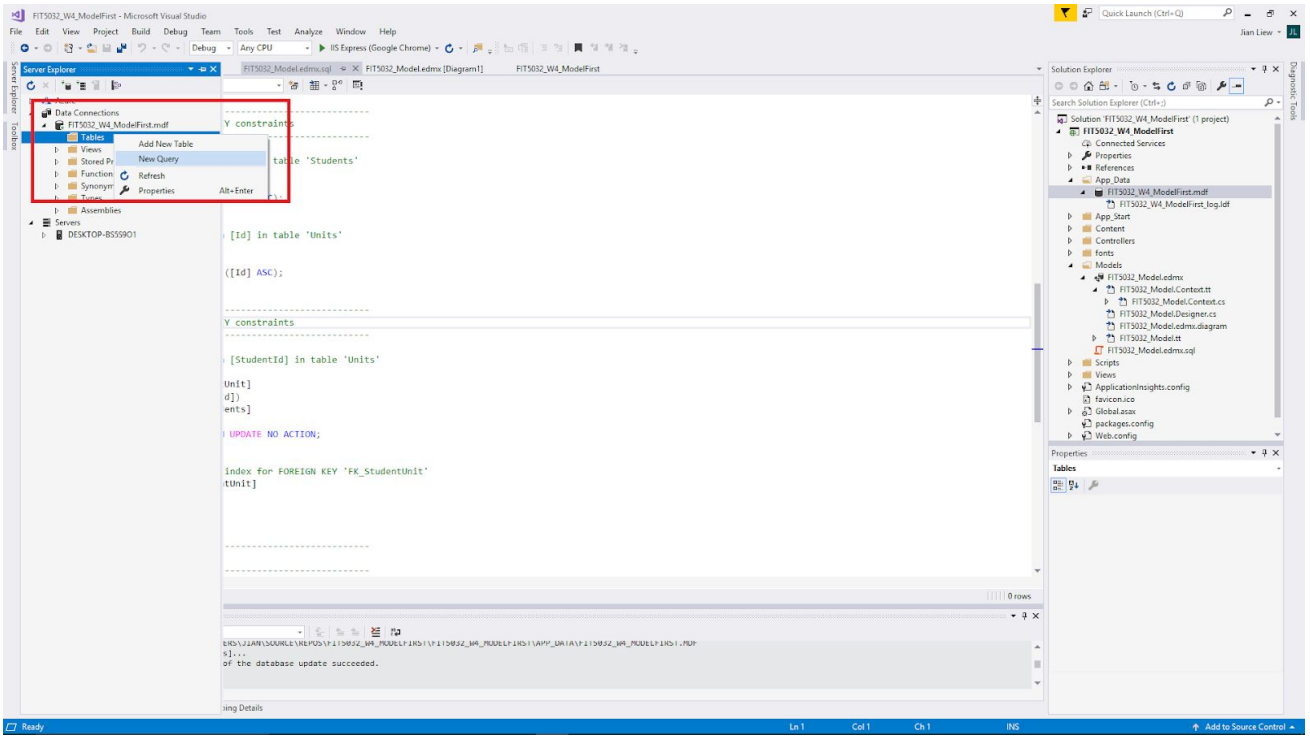


If you see this please also say "Yes to all"



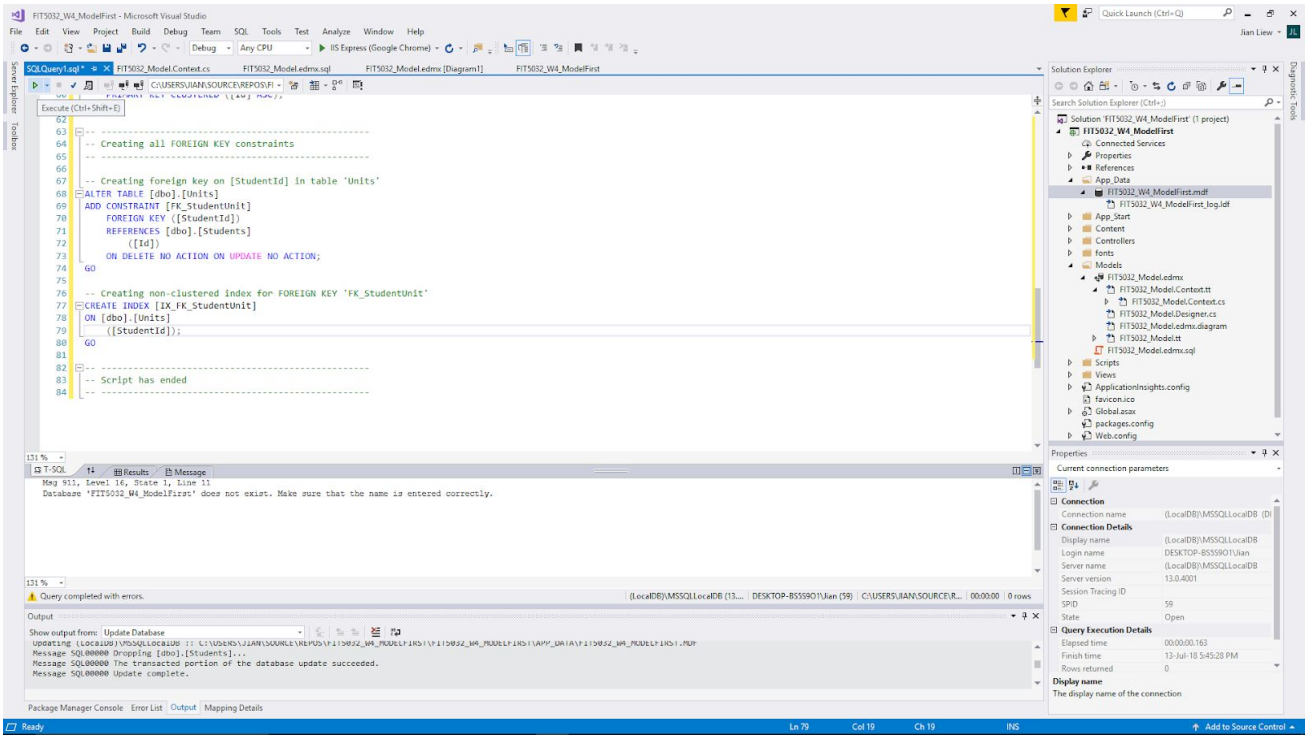
You will notice that this will automatically create the DDL needed to create the database tables. What will you need to do now, is to execute this SQL query in the database. Right click the .mdf file and select "Open"



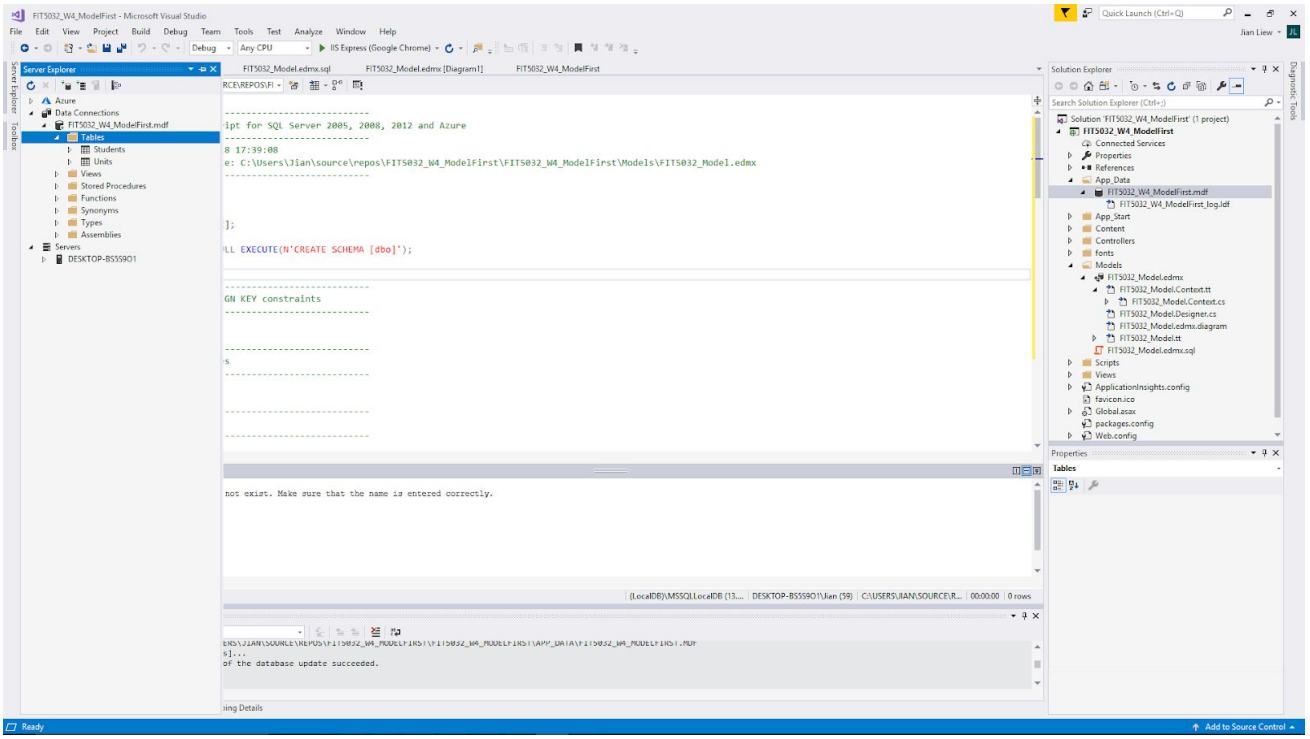


You can now select a "New Query" at the "Server Explorer" window.

You will now need to **copy and paste the contents of the .sql file generated at the query execution window**. You might see an error, in this case if you see the error below, this is expected.



Then you just need to the execute.



You should now see the tables created once you have "Refreshed" the .mdf file.

Upon the completion of this step, you have successfully completed the Model First approach.

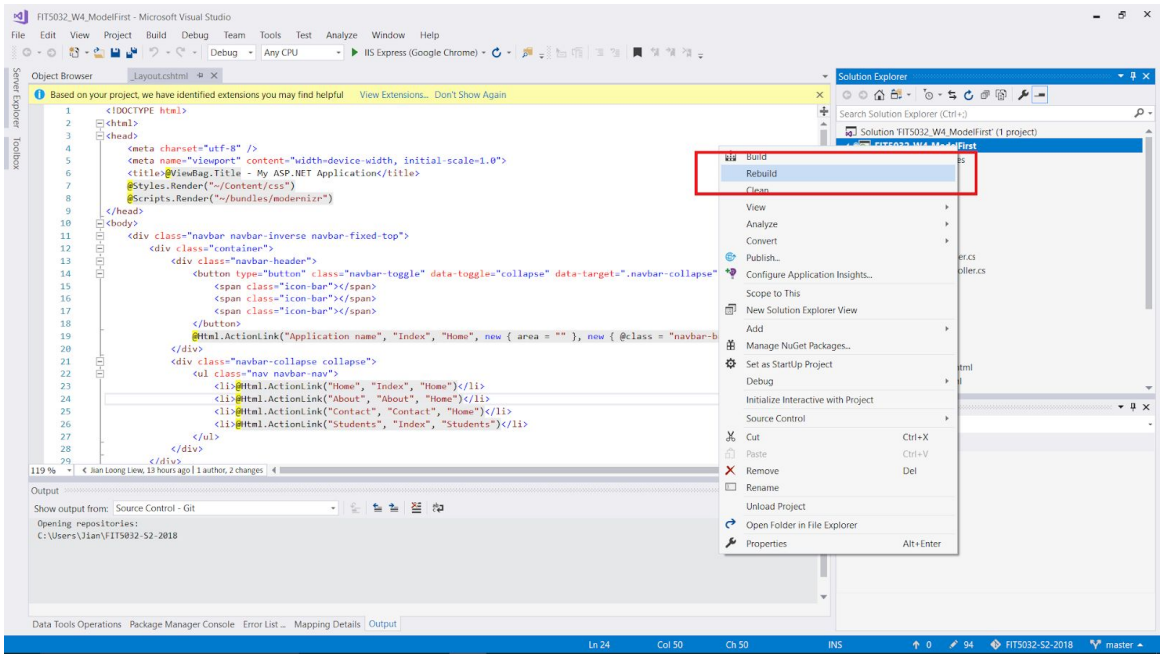
The general concept when using the Model first approach is, the Visual Designer would be used to draw a diagram (a diagram very similar to a class diagram). All the attributes and associations will be mapped into this diagram.

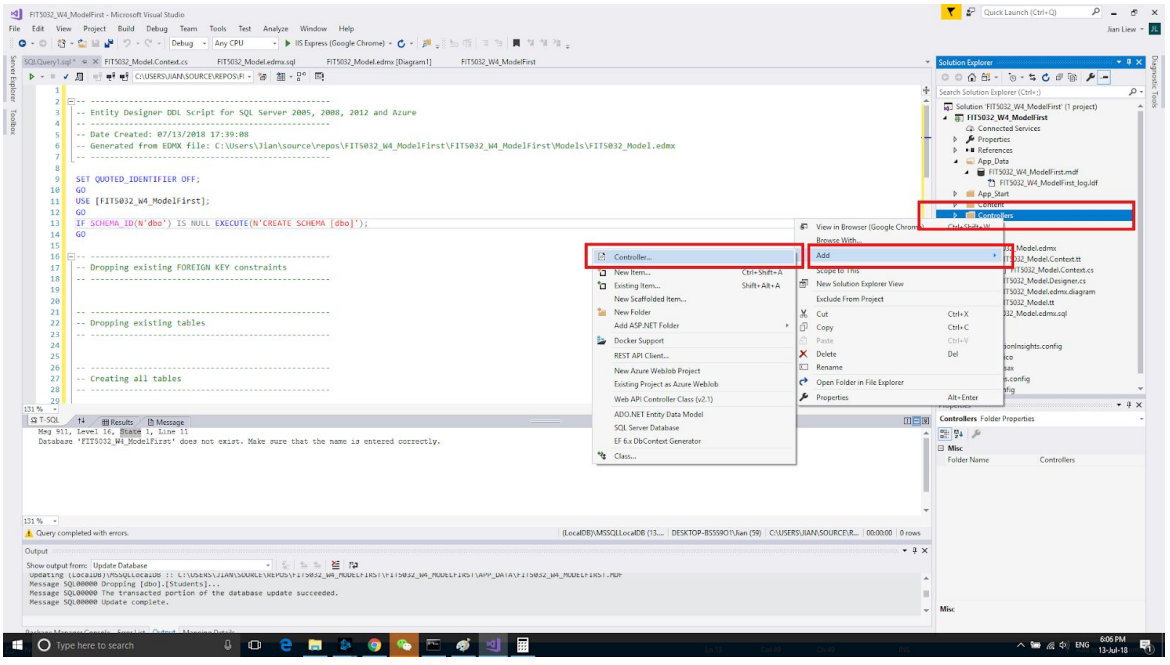
After the diagram has been drawn, the model can be used to generate the classes and the context. **Remember that the DbContext is very important for the application.** It will also create the needed DDL queries.

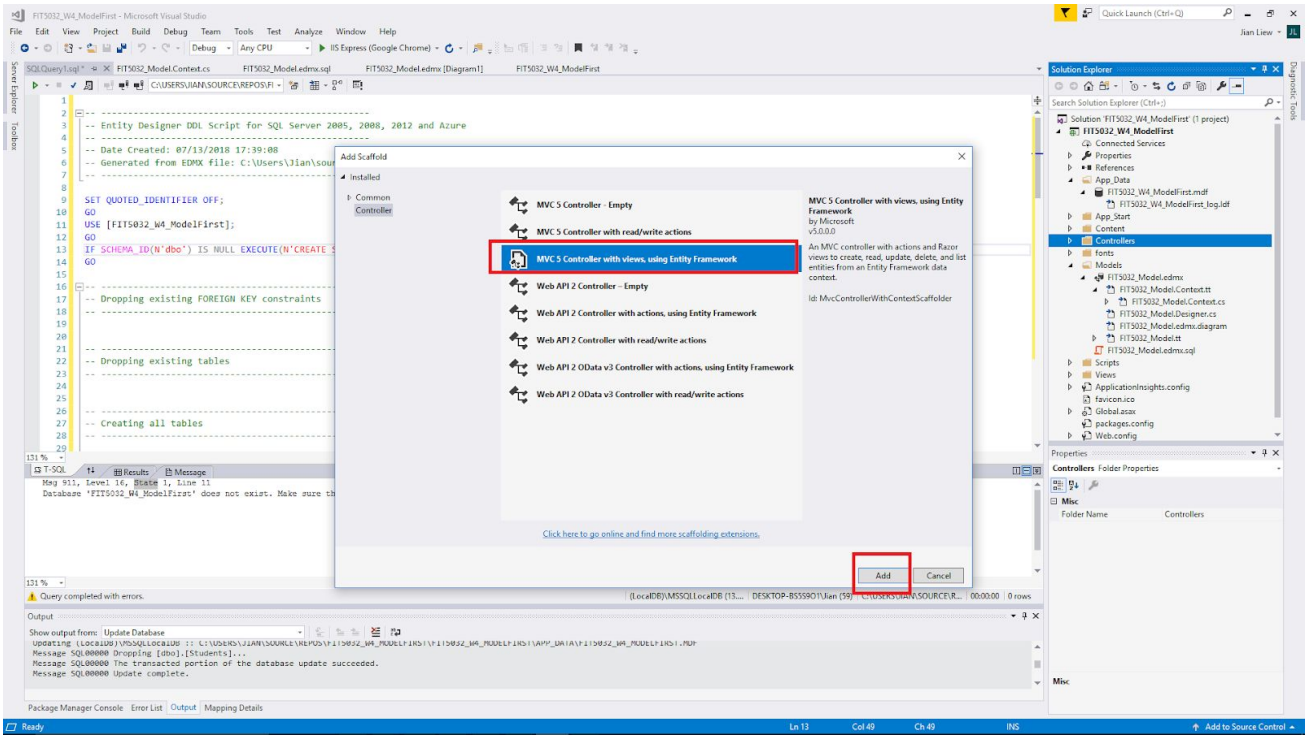
Also, remember that a database needs to be created to store the tables. The generated DDL will then be used to create the tables automatically. The interesting thing about this approach is that you will notice that you did not write a single line of SQL at all.

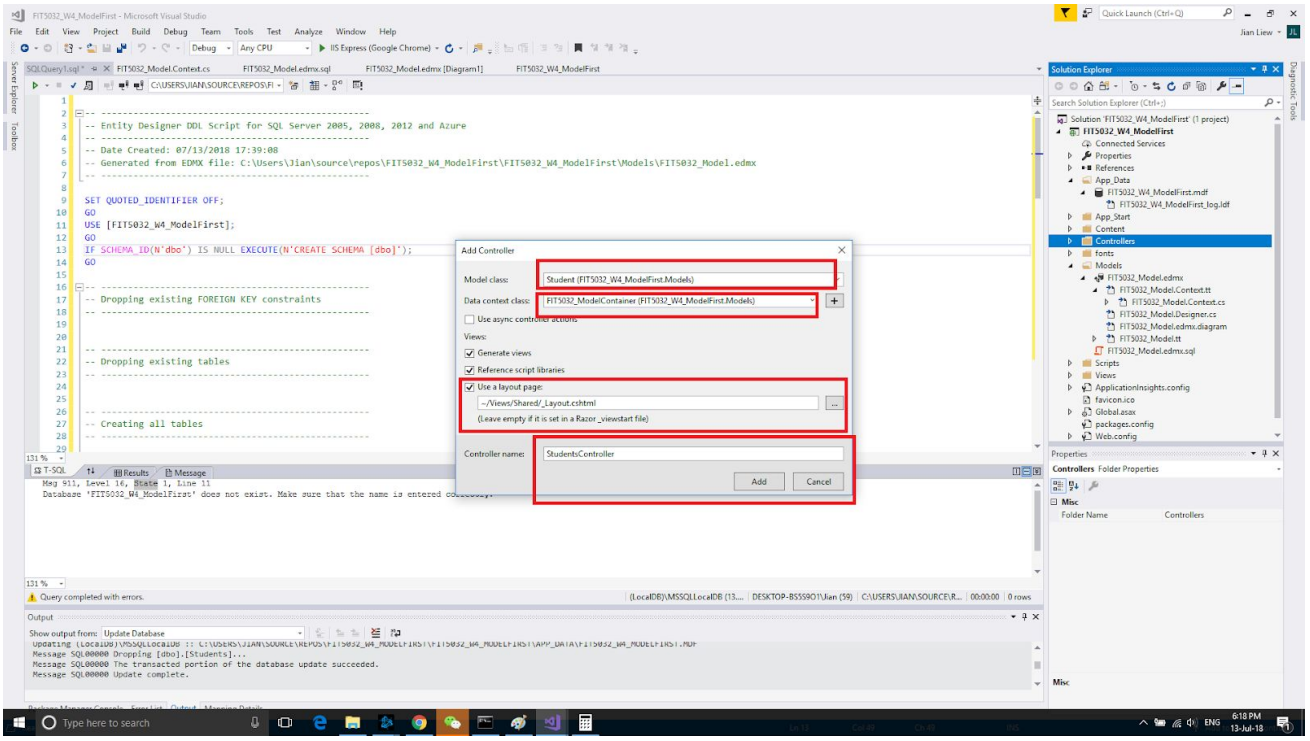
Scaffolding Controllers

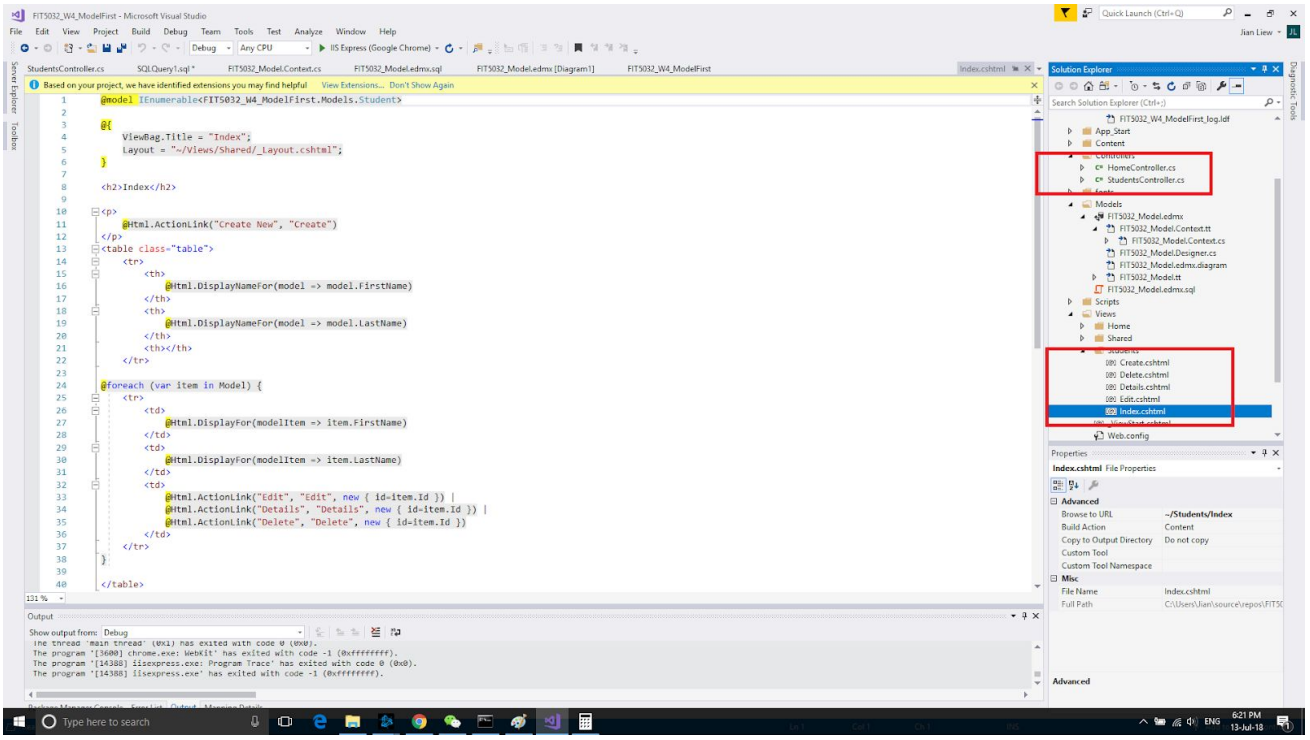
One of more interesting features in Visual Studio is that you can automatically generate certain parts of your codes to short cut development. At the end of the day, developers like to take shortcuts. **However, this feature is not perfect and should be tested out. Before you do this step, you will need to "Rebuild" your project. This is done by right clicking the project and selecting Rebuild.**



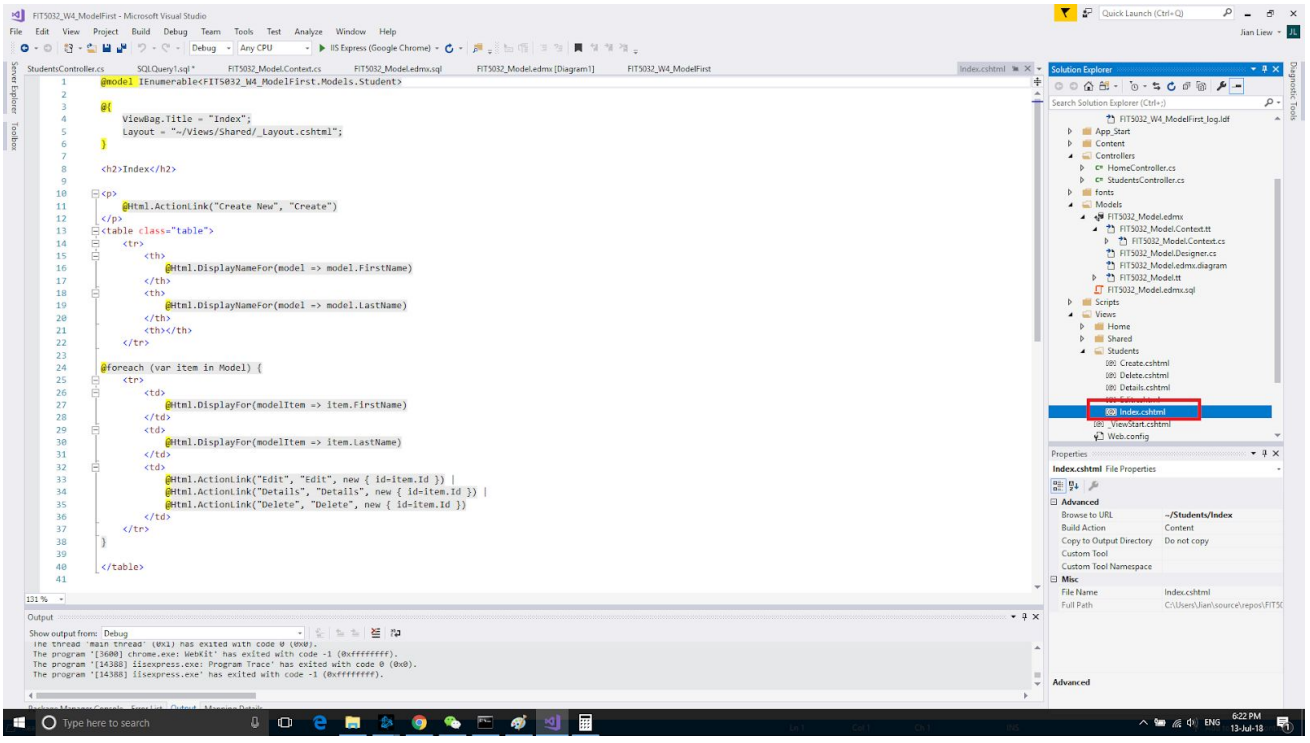



Add Scaffold
Installed
Common
Controller
MVC 5 Controller - Empty
MVC 5 Controller with read/write actions
MVC 5 Controller with views, using Entity Framework
Web API 2 Controller - Empty
Web API 2 Controller with actions, using Entity Framework
Web API 2 Controller with read/write actions
Web API 2 OData v3 Controller with actions, using Entity Framework
Web API 2 OData v3 Controller with read/write actions
MVC 5 Controller with views, using Entity Framework
by Microsoft
An MVC controller with actions and Razor views to create, read, update, delete, and list entities from an Entity Framework data context.
Id: MvcControllerWithContextScaffolder
Click here to go online and find more scaffolding extensions.
Add
Cancel


Add Controller
Model class:
Student (FIT5032_W4_ModelFirst.Models)
Data context class:
FIT5032_ModelContainer (FIT5032_W4_ModelFirst.Models)
Use async controller actions
Views:
Generate views
Reference script libraries
Use a layout page:
~/Views/Shared/_Layout.cshtml
(Leave empty if it is set in a Razor _viewstart file)
Controller name:
StudentsController
Add
Cancel

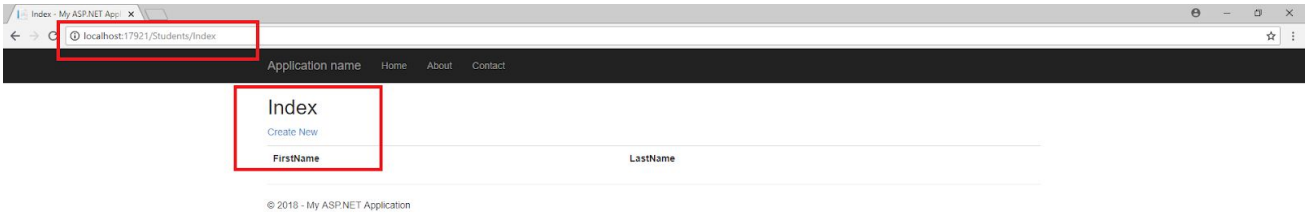


By doing this, Visual Studio is able to automatically generate stub codes for the Student Controller and as well as the Student View.



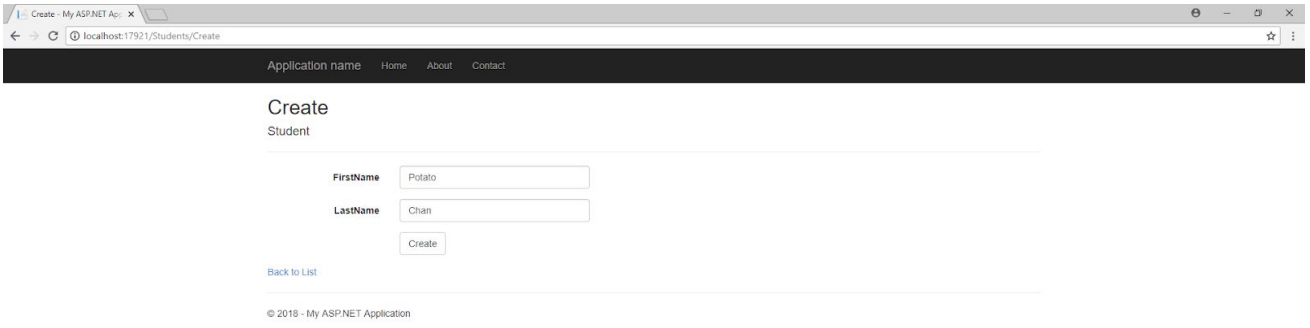
Double Click on the Index.cshtml inside of the Student View.

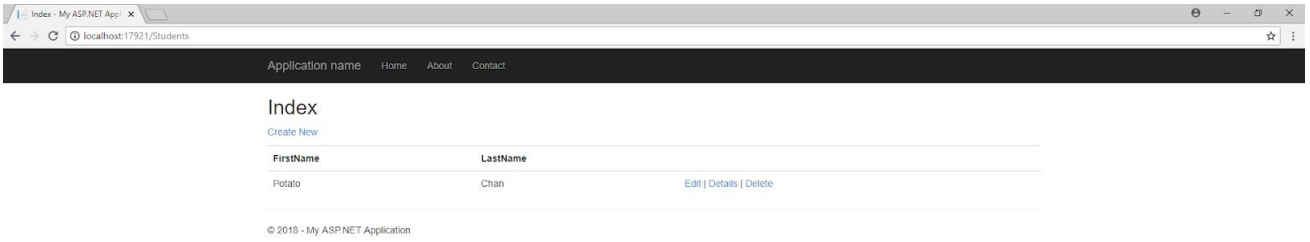
After that you can hit Run and you will notice that some pages have been created for us.



I will then go ahead and try it out.

I will create a student whose First Name is Potato and Last Name is Chan and hit the create button.





Notice that the student has now been created. In summary, what I can do is, I can automatically create the Controller and Views with minimal coding at best. You can add a link to this page.

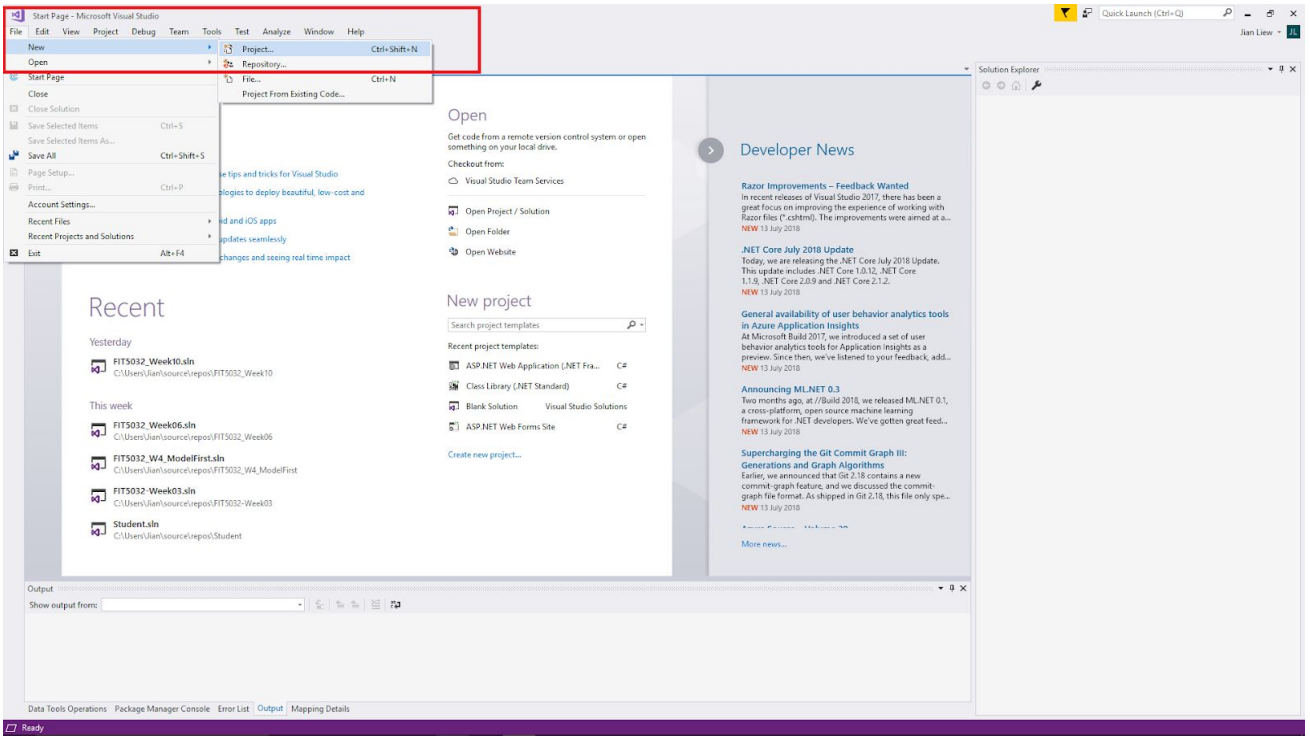
The Database First Approach

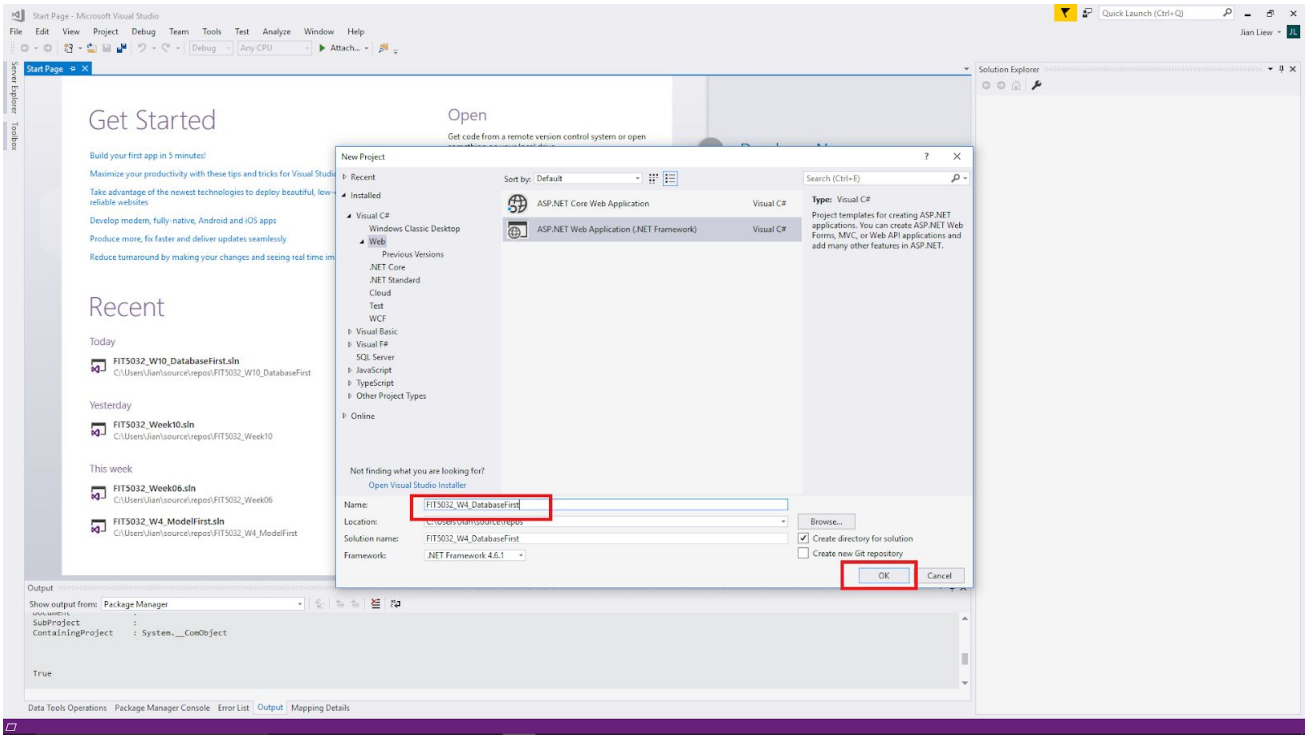
The database first approach involves writing Data Definition Language DDL first. So, you will need to use your knowledge of SQL. This is a common approach if there is an existing database design. Some developers that have an in depth knowledge in SQL prefer this way. This is one of the more common approaches as it is perceived to be easier.

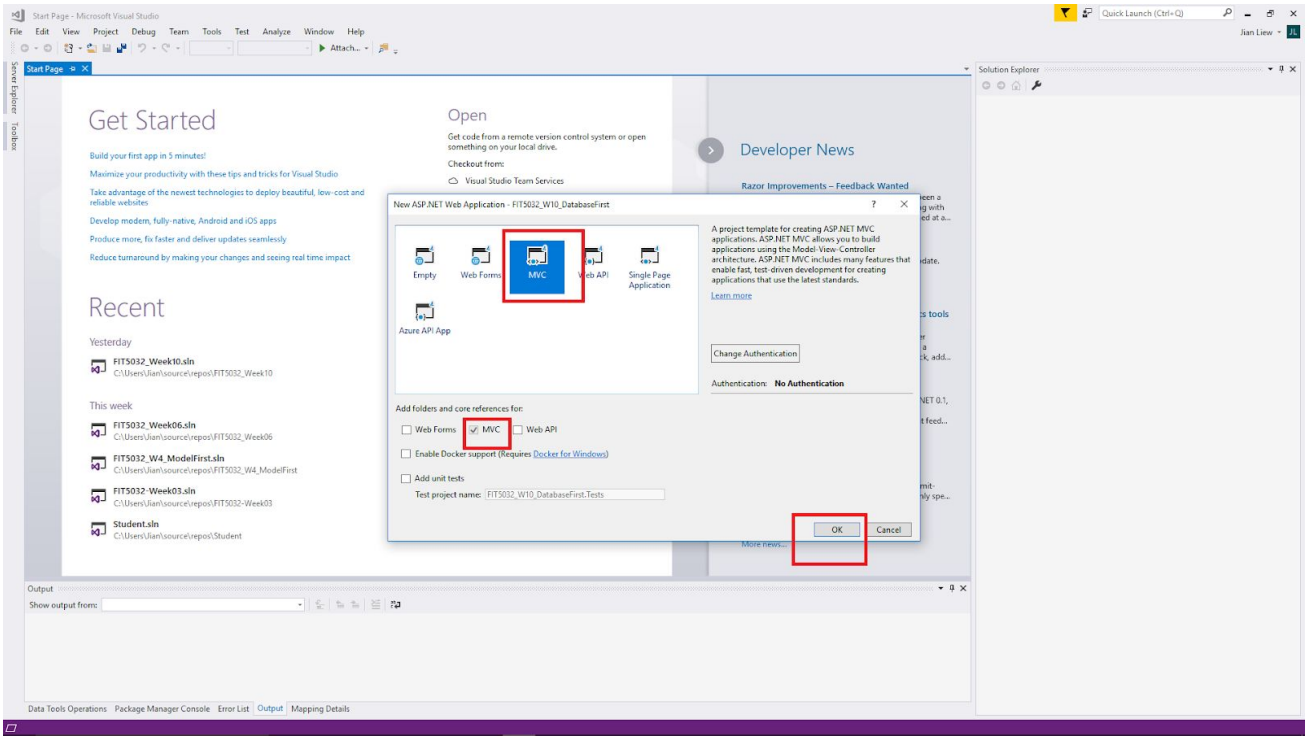
Please remember that every database syntax is slightly different. Even though SQL follows an ISO standard there are syntax which are database specific.

Step 1

I will create a new project to demonstrate the database first approach.

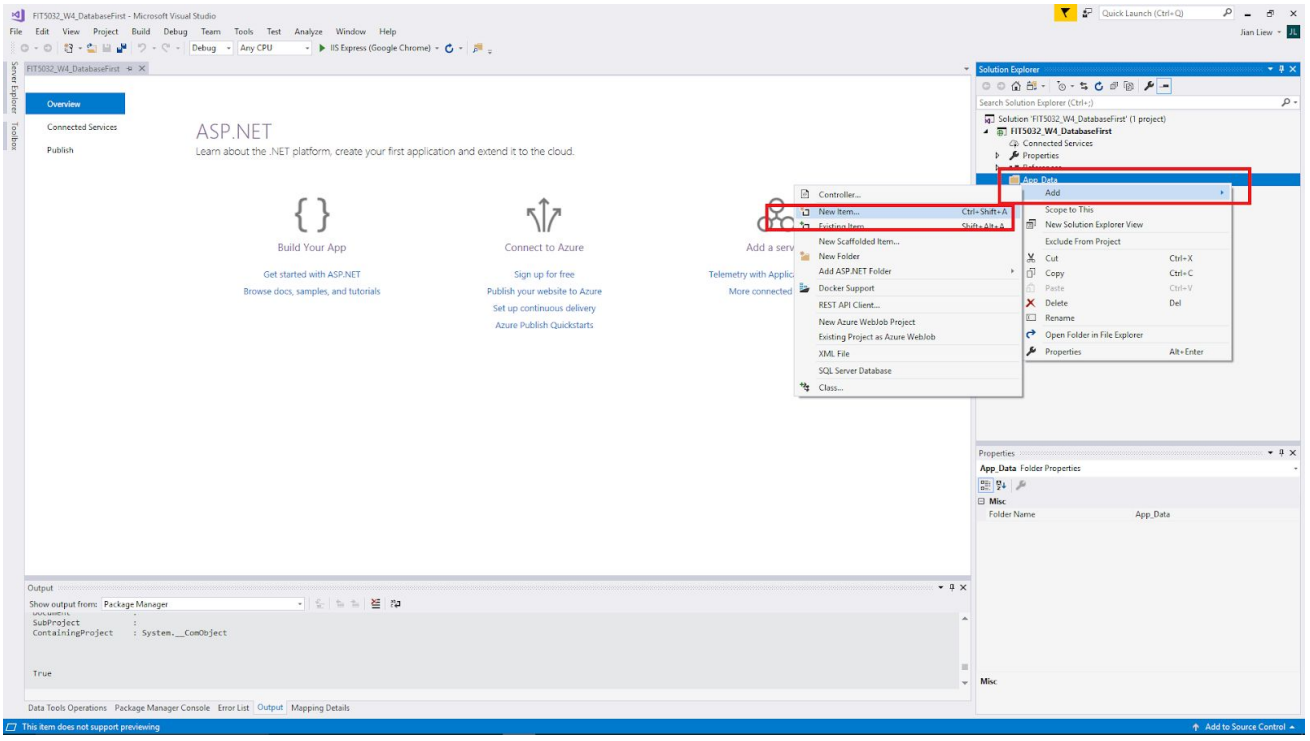


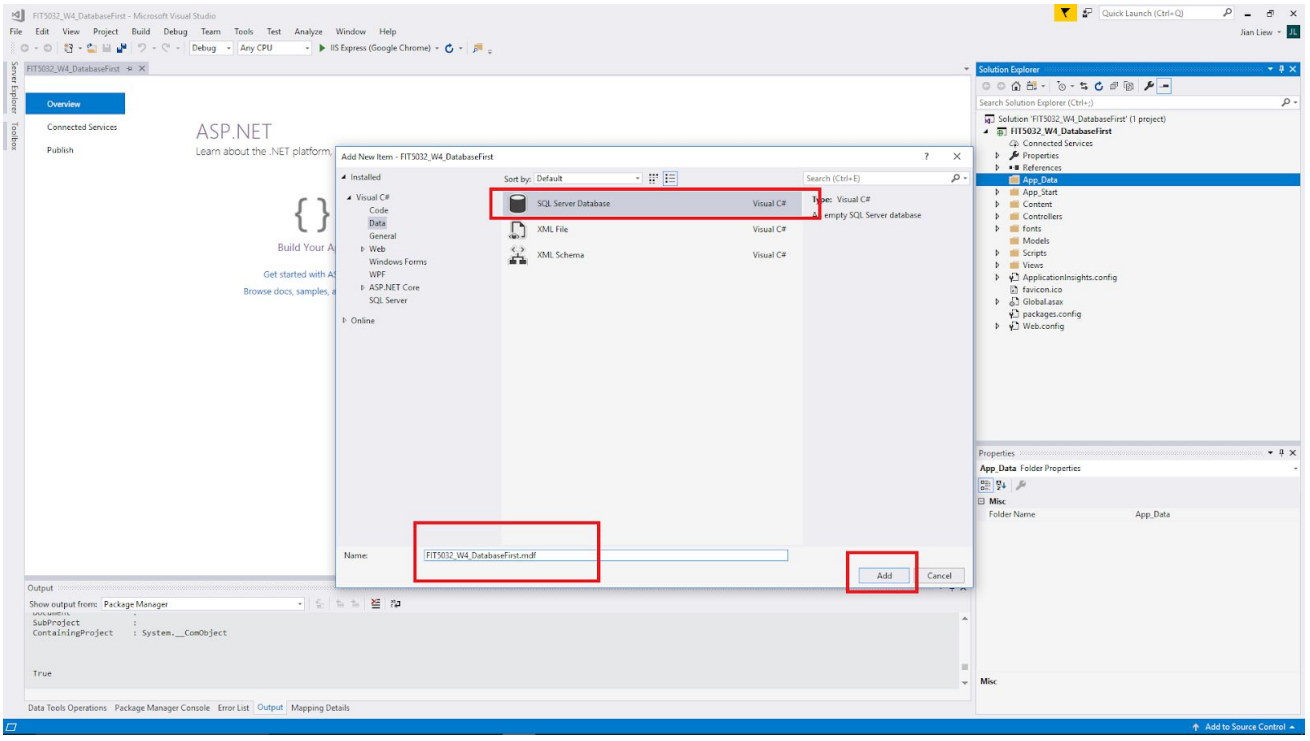

Start Page - Microsoft Visual Studio
File Edit View Project Debug Team Tools Test Analyze Window Help
Start Page
Get Started
Open
Recent
Today
FIT5032_W10_DatabaseFirst.sln
Yesterday
FIT5032_Week10.sln
This week
FIT5032_Week06.sln
FIT5032_W4_ModelFirst.sln
New Project
Recent
Installed
Visual C#
Windows Classic Desktop
Web
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)
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_W4_DatabaseFirst
Location:
Solution name: FIT5032_W4_DatabaseFirst
Framework: .NET Framework 4.6.1
Browse...
Create directory for solution
Create new Git repository
OK
Cancel
Output
Show output from: Package Manager
SubProject :
ContainingProject : System.__ComObject
True
Data Tools Operations
Package Manager Console
Error List
Output
Mapping Details
Solution Explorer


Start Page - Microsoft Visual Studio
File Edit View Project Debug Team Tools Test Analyze Window Help
Start Page
Get Started
Open
Get code from a remote version control system or open something on your local drive.
Checkout from:
Visual Studio Team Services
Developer News
Razor Improvements – Feedback Wanted
Recent
Yesterday
FIT5032_Week10.sln
This week
FIT5032_Week06.sln
FIT5032_W4_ModelFirst.sln
FIT5032-Week03.sln
Student.sln
New ASP.NET Web Application - FIT5032_W10_DatabaseFirst
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_W10_DatabaseFirst.Tests
OK
Cancel
Output
Show output from:
Data Tools Operations
Package Manager Console
Error List
Output
Mapping Details
Solution Explorer

Step 2

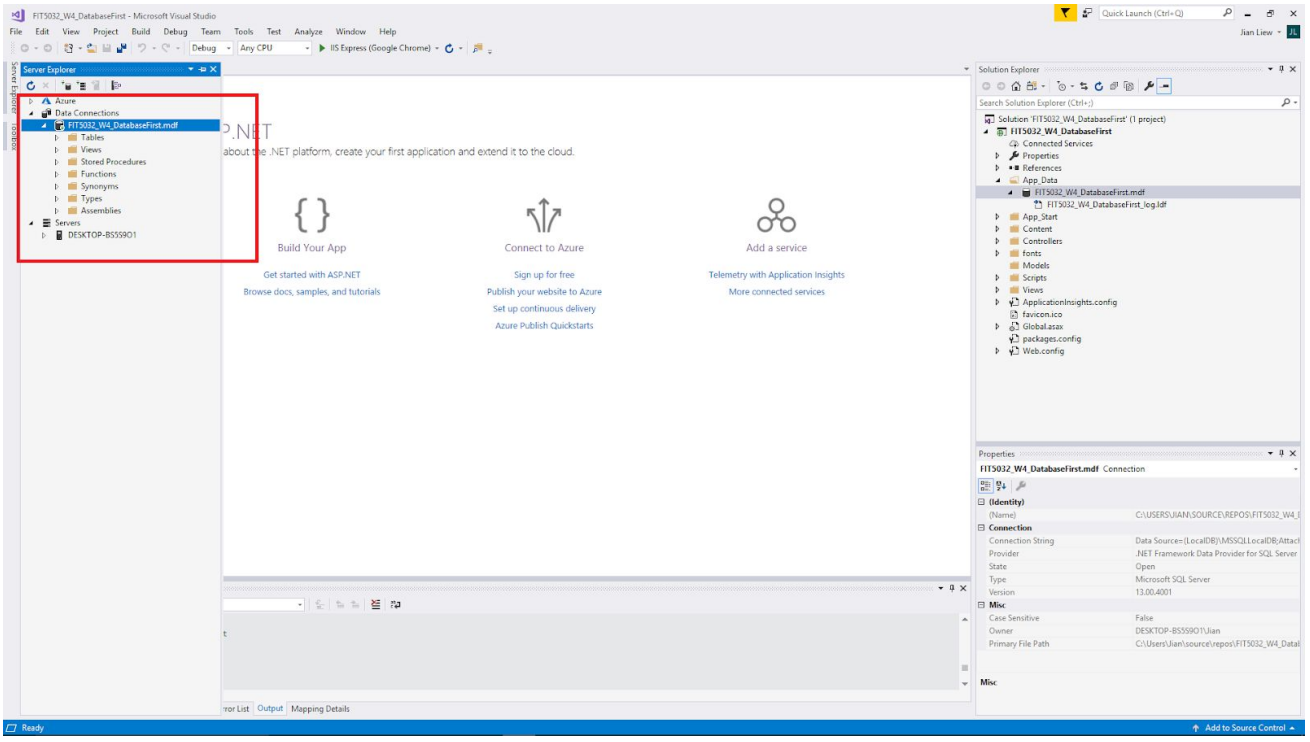
Right click App_Data and Select "New Item"



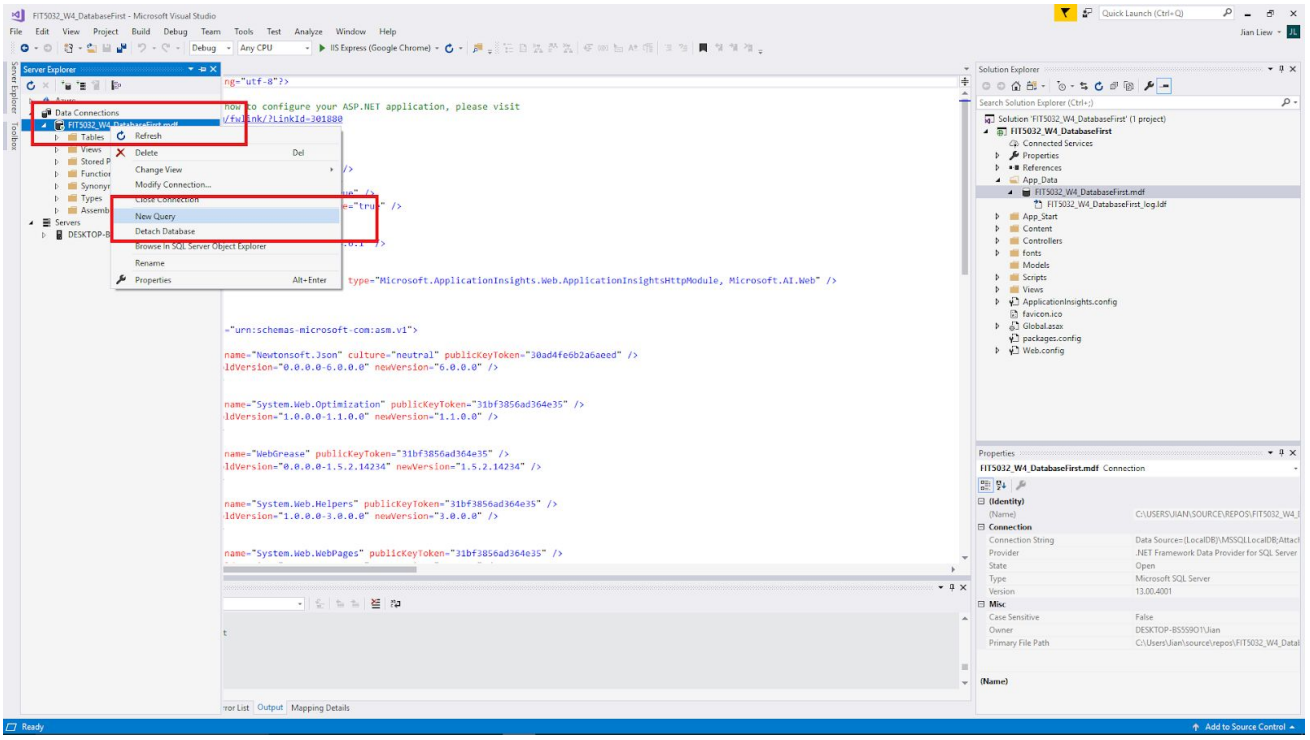


Step 3

You should see that the database first is created under the App_Data folder. Double click on it to open it.



You can then run SQL queries on this database.



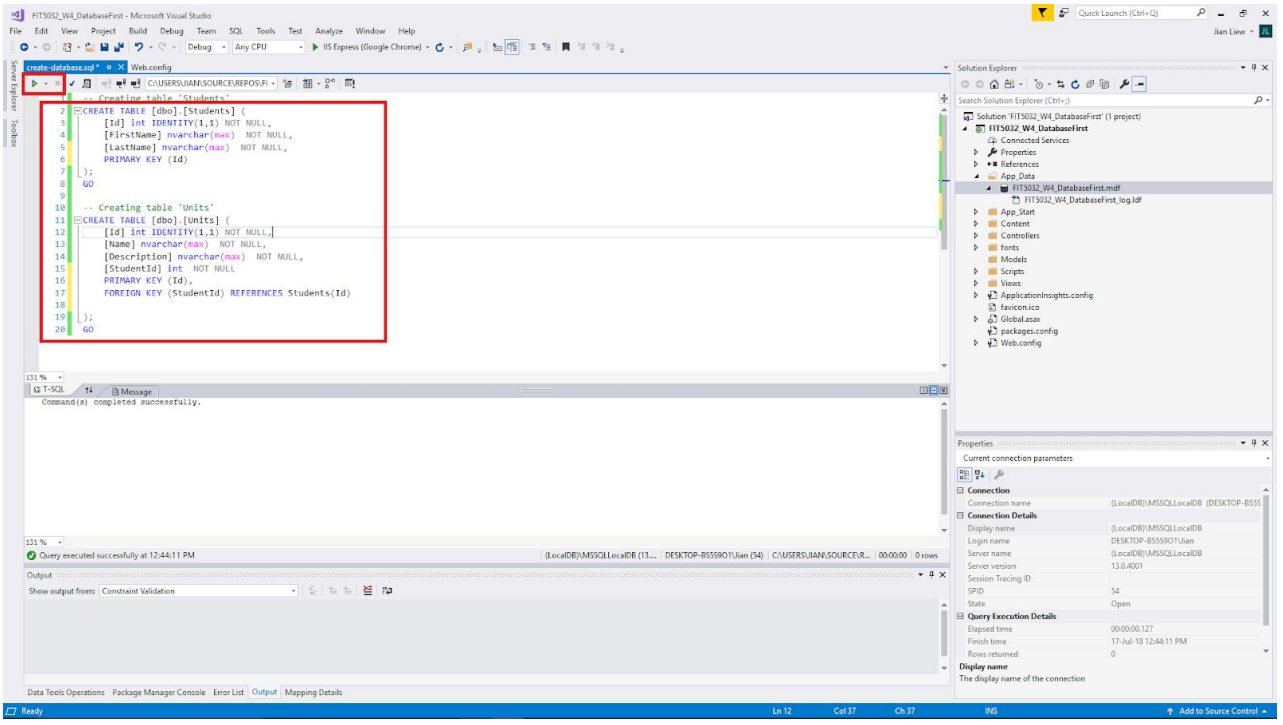
Step 4

The SQL needed to generate the tables are provided here. **Please note that if you using Edge or IE you might be able to copy and paste from this document. Please use Chrome instead. If you are using Adobe Reader there might white space introduced that the columns please remove them.**

```
-- Creating table 'Students'
CREATE TABLE [dbo].[Students] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [FirstName] nvarchar(max)  NOT NULL,
    [LastName] nvarchar(max)  NOT NULL,
      PRIMARY KEY (Id)
);
GO

-- Creating table 'Units'
CREATE TABLE [dbo].[Units] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(max)  NOT NULL,
    [Description] nvarchar(max)  NOT NULL,
    [StudentId] int  NOT NULL
      PRIMARY KEY (Id),
      FOREIGN KEY (StudentId) REFERENCES Students(Id)

);
GO
```

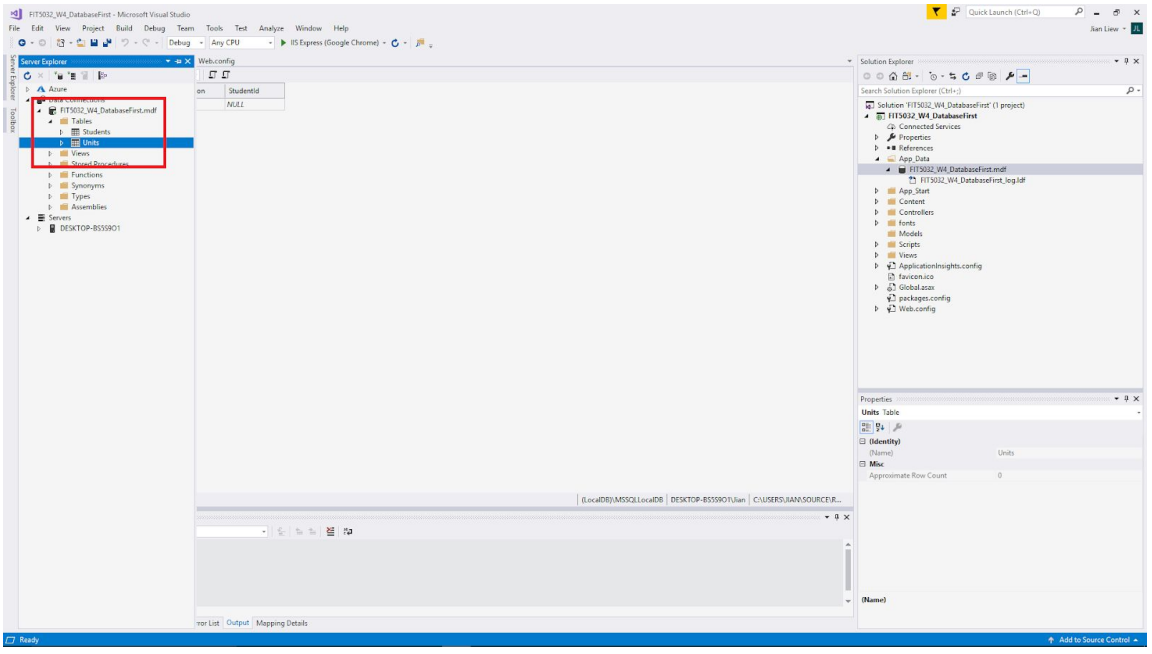


How do you think the nvarchar data type differs in comparison to the varchar? Also what do you think is the difference between a Primary Key and an Identity?

Instead of directly using SQL, you can also create the table manually when your right click the database and select "Add New Table". This will give you the GUI view of the table creation.

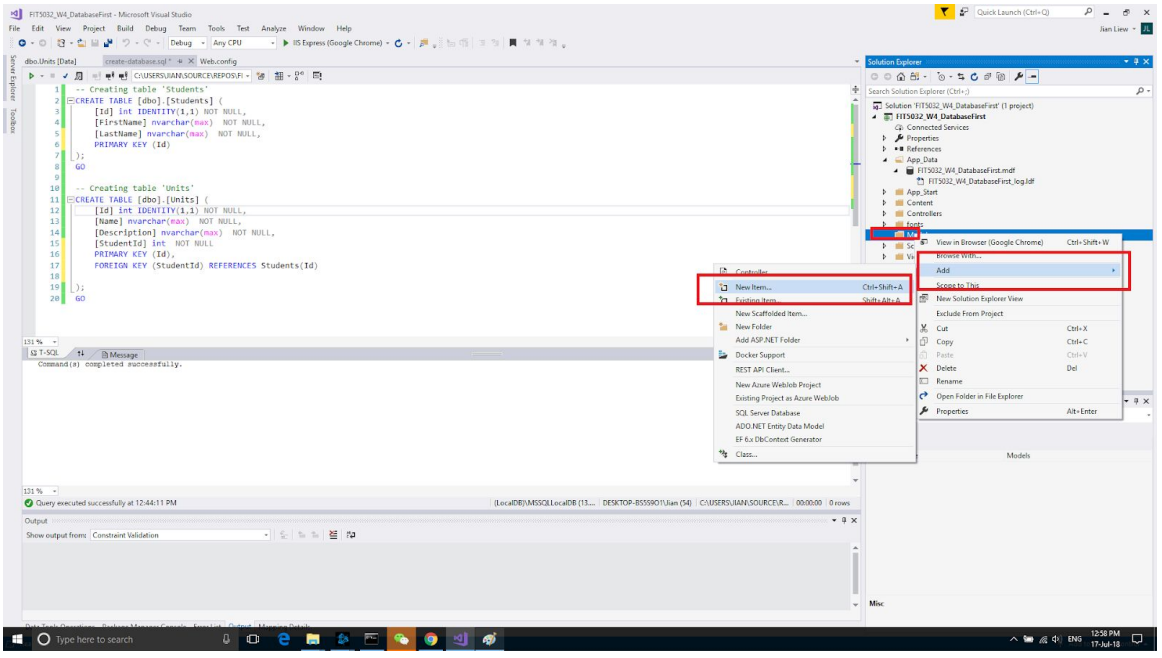
Step 5

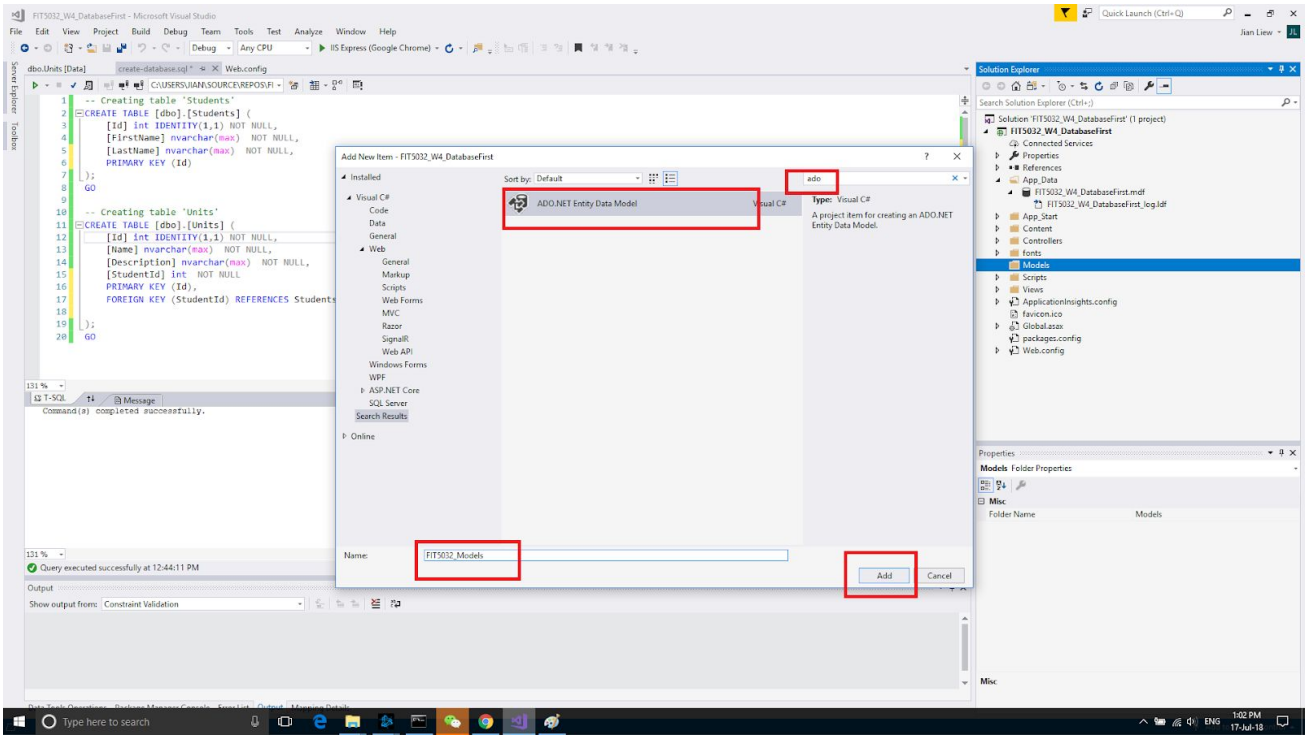
You can now see that the tables are created.

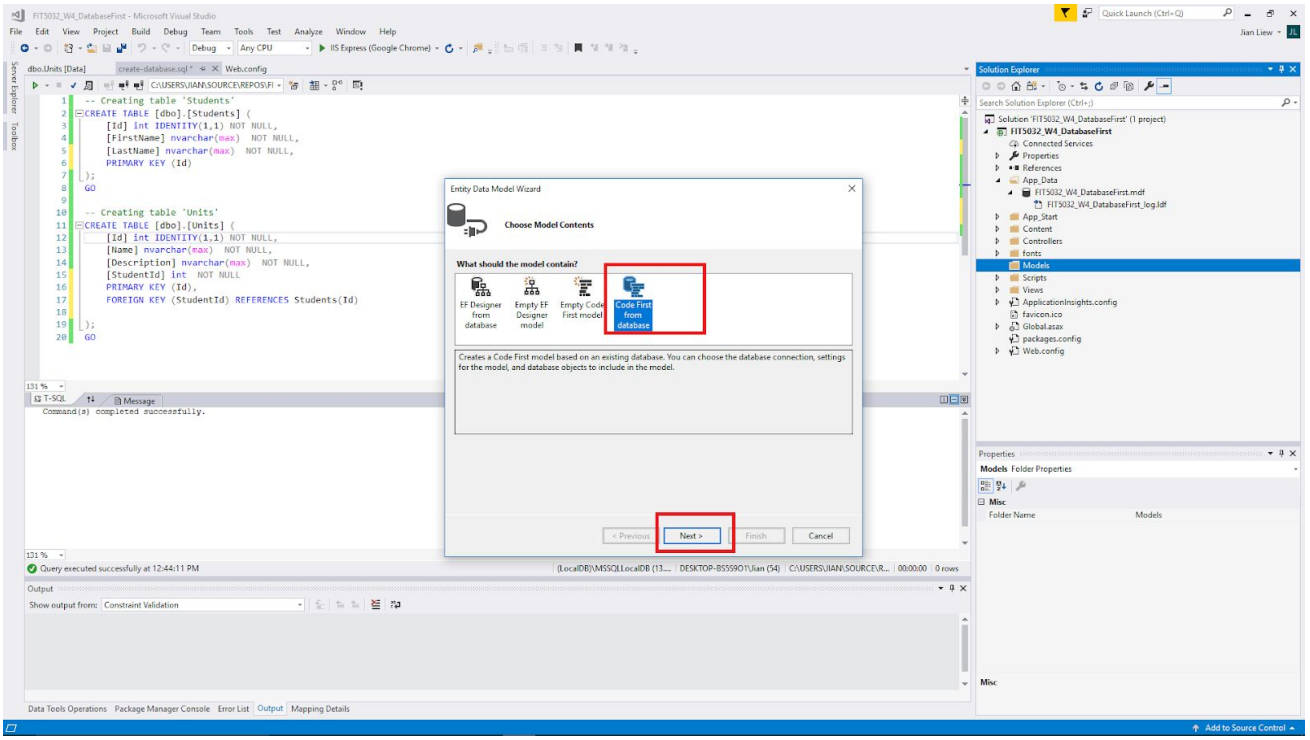


Step 6

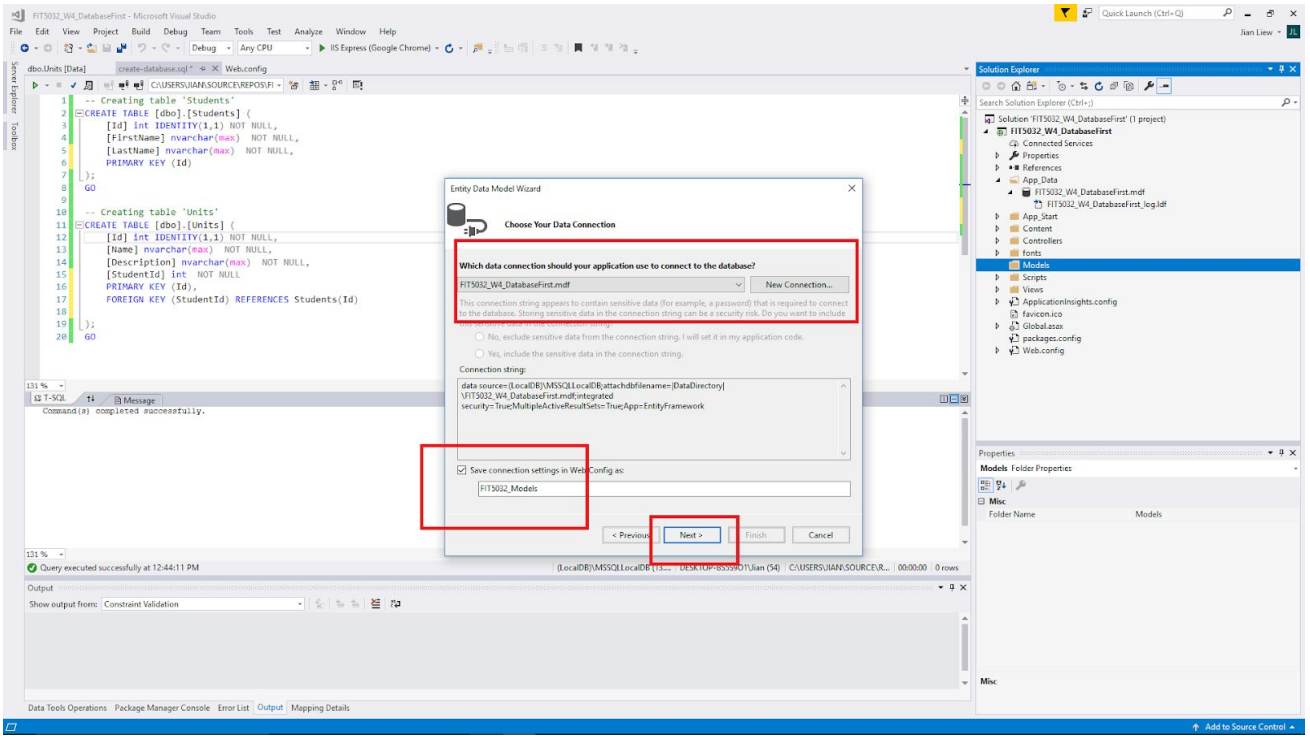
We will now generate the models from the database which has been created. Right click model and add then add a "New Item"

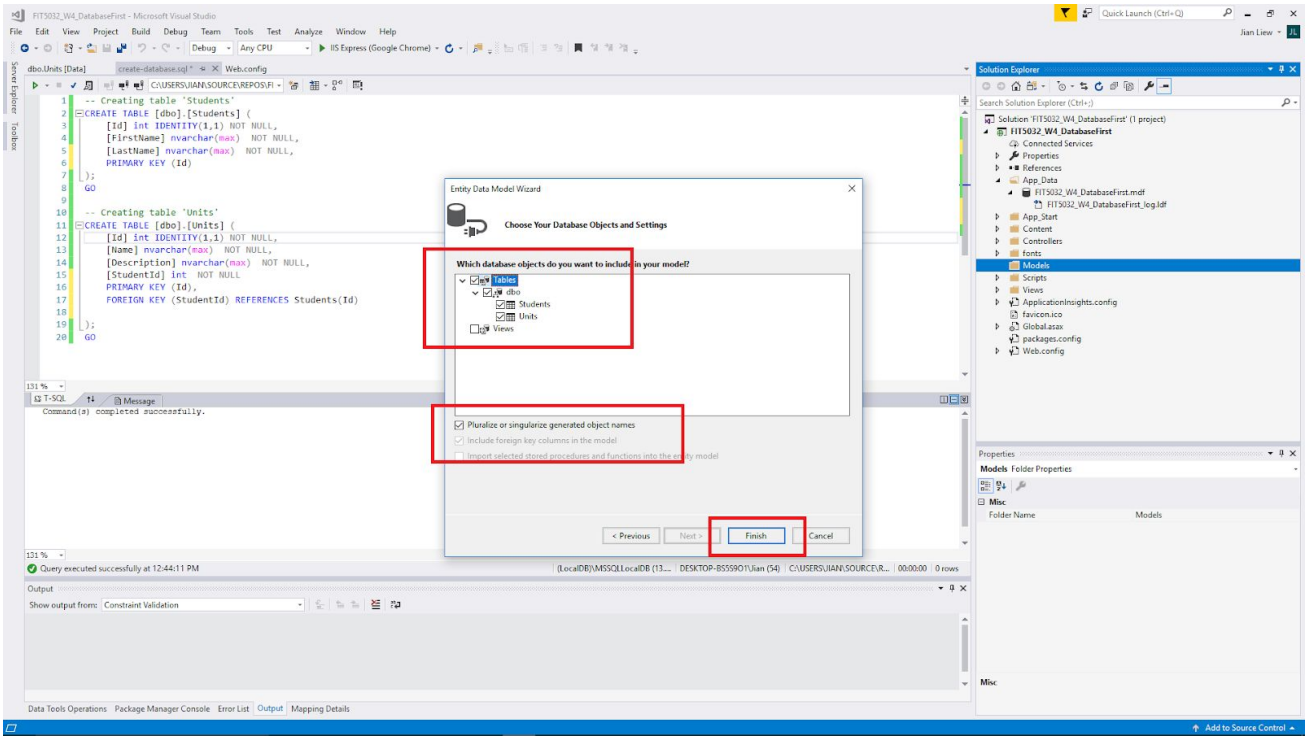


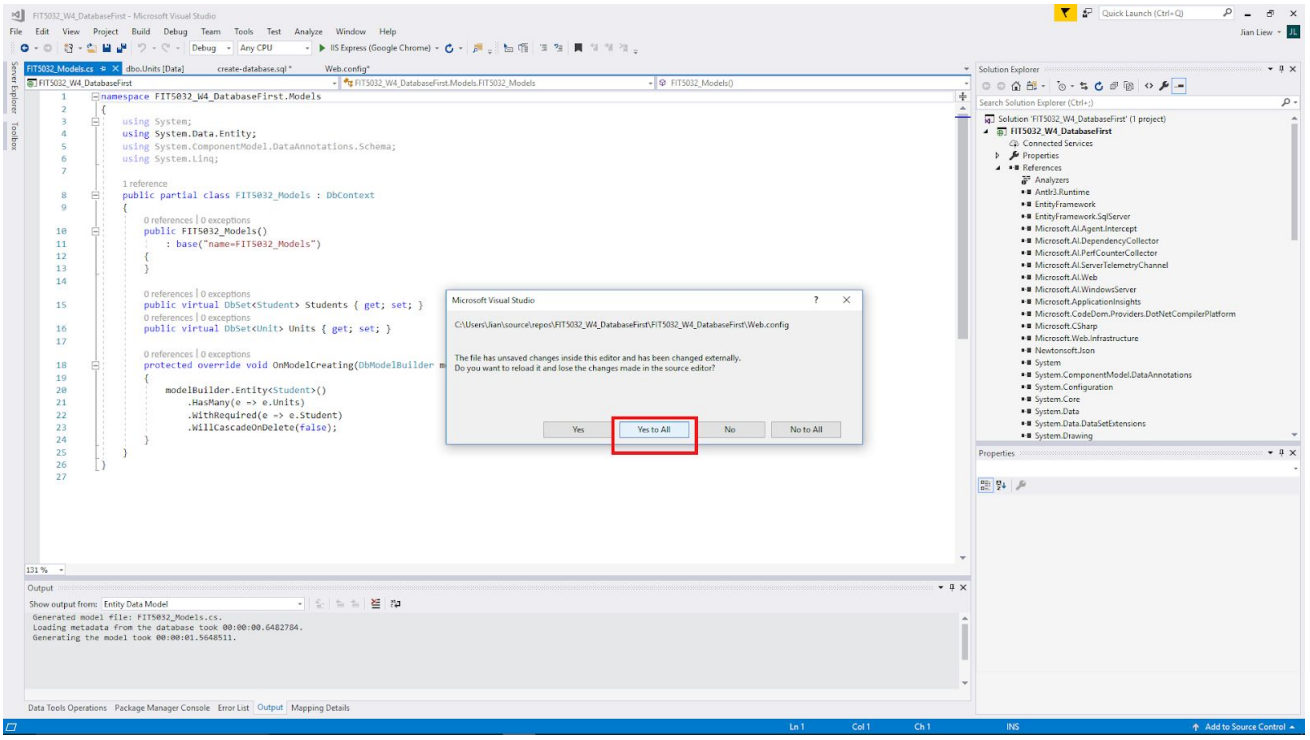


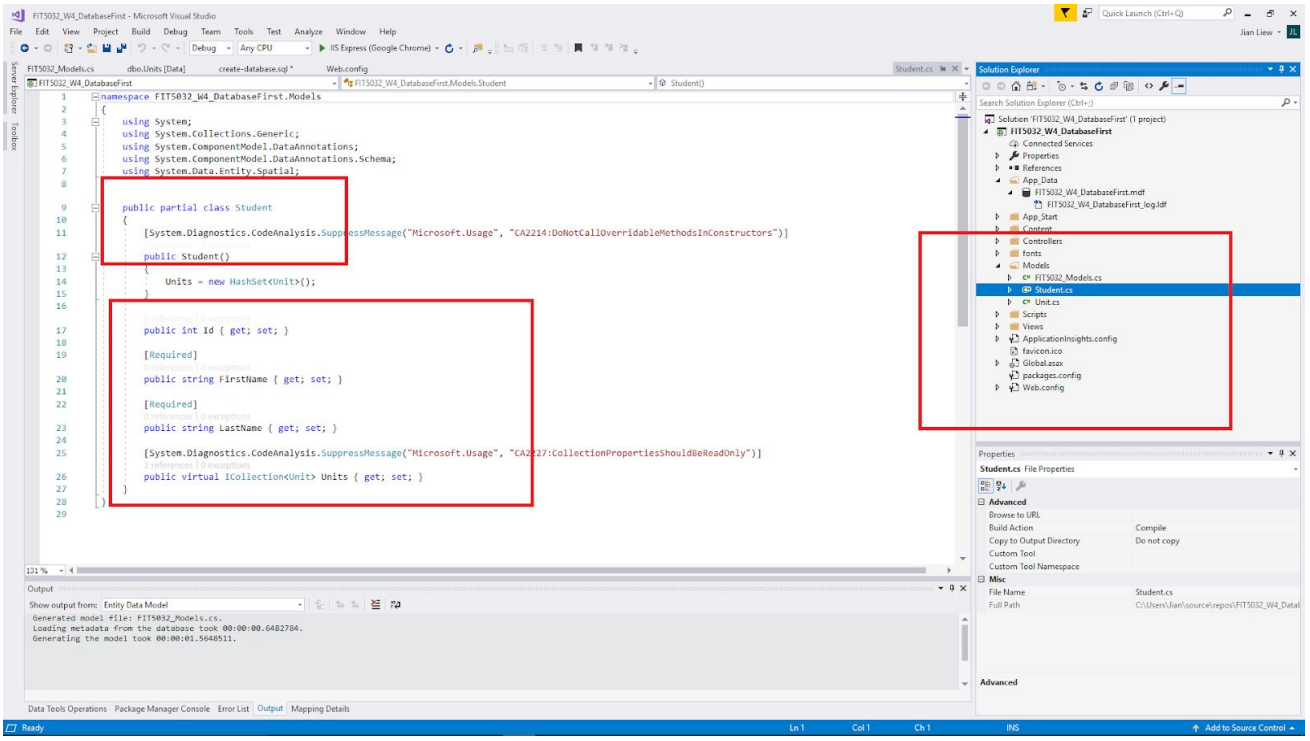


This option means that, it will create a code first approach from the database itself. This approach is generally much cleaner.


Entity Data Model Wizard
Choose Your Data Connection
Which data connection should your application use to connect to the database?
FIT5032_W4_DatabaseFirst.mdf
New Connection...
Connection string:
Save connection settings in Web.Config as:
FIT5032_Models
< Previous
Next >
Finish
Cancel


Entity Data Model Wizard
Choose Your Database Objects and Settings
Which database objects do you want to include in your model?
Tables
dbo
Students
Units
Views
Pluralize or singularize generated object names
Include foreign key columns in the model
< Previous
Next >
Finish
Cancel





After you have completed this task, you can **repeat the step where you scaffold the controllers to generate the views.** Essentially both ways will accomplish the same results.

The Code First Approach

The Code First approach involves the writing of the classes first. Your task will be explore how this is done if you have completed the two earlier approaches. The decision to do this is left up to you. It is requirement to understand how the Code First works.

DoubtFire Submission

The DoubtFire submission for this week is

- A document detailing your preferred approach and a justification why you would select this approach. Please note that this is an opinion based question and there is no correct answer for this question. This document should not be longer than 1 page in length.
- [Updated 09/08/2017] A document showing one of your week 4 database applications running. (in a PDF document)

Challenge Exercise

1. Normally in real life, models are more complicated than this. For example, in this tutorials only two tables or entities are used. However, real systems are more complicated than this. Let's assume that, there is a many to many relationship between Student and Unit. How would you model this? Is there a difference between using the model first or database first approach if this situation happens?
2. In the labs, I took a very simplistic approach when designing the database. Do you think you can improve upon it?

Change Log

Date	Summary
13 August 2018	• Added more specific instructions on where IE or Edge does not work well during the copy and paste of the SQL. • Added more instructions on when to "Rebuild" the project after the completion of models.
15 August 2018	• Added mention that you can use the "Add New Table" instead of using SQL statements.