**INF440 Lab Exercises - Creating a Data-First MVC Project using ASP.NET MVC**

This lab will introduce you to the basics of building an **ASP.NET MVC 5** web app using Visual Studio 2013 with the **Data-First** approach. Using ASP.NET MVC, Entity Framework, and Scaffolding, you will create a web application that provides an interface to an existing database.

This lab shows you how to automatically generate the code that enables users to display, edit, create, and delete data that resides in a database table. The generated code will allow you to perform common operations on the data residing in the database table.

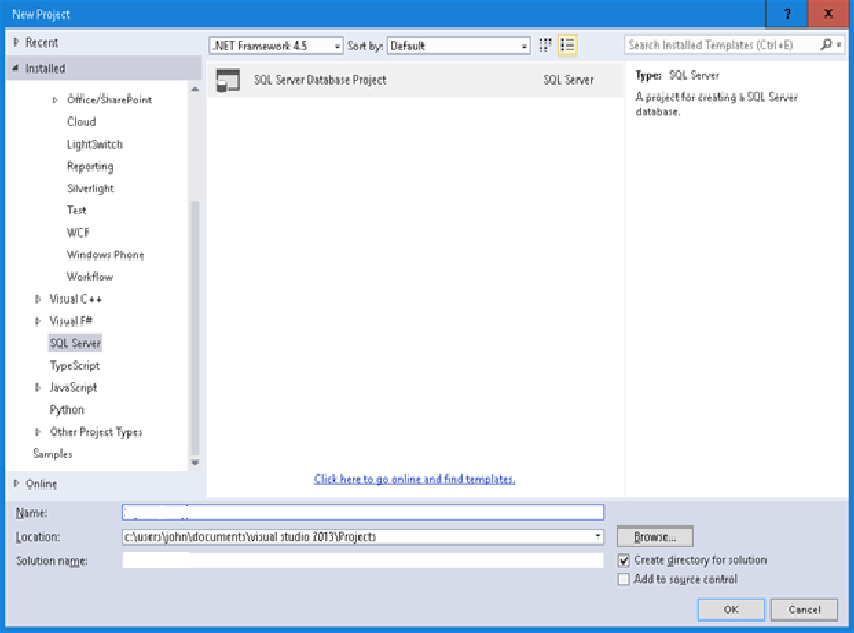
First, a small database will be created and used by the application.

**1. Getting Started – Setting up the Database**

Start **Visual Studio 2013**.

From the **Start** page, click **New Project**, then select **SQL Server** in the LHS window pane, then select **.NET Framework 4.5**, and finally select **SQL Server Database Project** in the main window Give it the name **INF440UniversityData**.

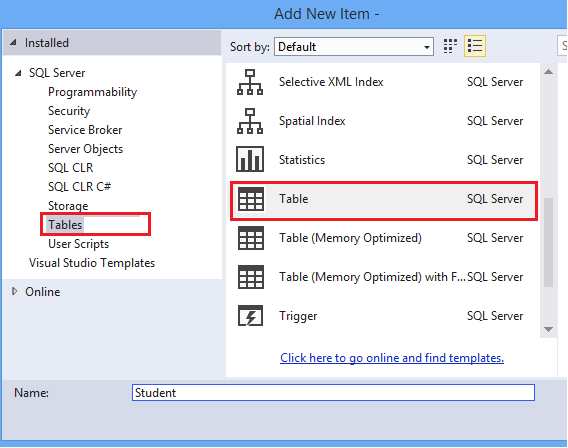
**Before** clicking **OK**, check the pathname in the **Location** box - if necessary, change “administrator” to your userid. But do **not** change the **Location** path.



Click **OK**. A new Visual Studio Project will be created for you. This may take several seconds.

You will now create the tables needed for this lab exercise by adding SQL scripts that define the tables.

Right-click your **Project** (**INF440UniversityData**) in the Solution Explorer window, and select **Add > New Item**. In the LHS pane of the pop-up window that appears, under **SQL Server**, select **Tables and Views**. And in the main window, select **Table**. Make the name of the table **Student.** Then click **Add.**



A new window will be displayed.

In the bottom window pane, replace the T-SQL command with the following code to create the table. (See image below)

CREATE TABLE [dbo].[Student] (

[StudentID] INT IDENTITY (1, 1) NOT NULL,

[LastName] NVARCHAR (50) NULL,

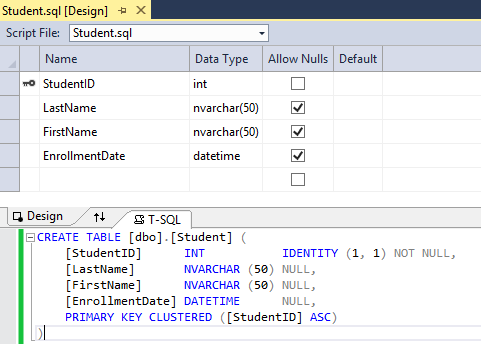
[FirstName] NVARCHAR (50) NULL,

[EnrollmentDate] DATETIME NULL,

PRIMARY KEY CLUSTERED ([StudentID] ASC)

)

Notice that the design window (top window pane) automatically synchronizes with the code. You can work with either the code or designer.



The actual creation of the table will be done in a moment.

Repeat the procedure to add another table. This time name it **Course** and insert the following T-SQL command.

CREATE TABLE [dbo].[Course] (

[CourseID] INT IDENTITY (1, 1) NOT NULL,

[Title] NVARCHAR (50) NULL,

[Credits] INT NULL,

PRIMARY KEY CLUSTERED ([CourseID] ASC)

)

And, repeat one more time to create a table named **Enrollment** and add the following SQL.

CREATE TABLE [dbo].[Enrollment] (

[EnrollmentID] INT IDENTITY (1, 1) NOT NULL,

[Grade] DECIMAL(3, 2) NULL,

[CourseID] INT NOT NULL,

[StudentID] INT NOT NULL,

PRIMARY KEY CLUSTERED ([EnrollmentID] ASC),

CONSTRAINT [FK\_dbo.Enrollment\_dbo.Course\_CourseID] FOREIGN KEY ([CourseID])

REFERENCES [dbo].[Course] ([CourseID]) ON DELETE CASCADE,

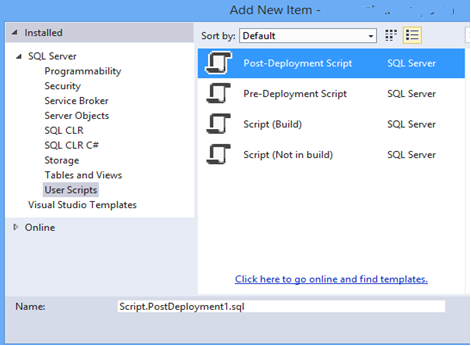
CONSTRAINT [FK\_dbo.Enrollment\_dbo.Student\_StudentID] FOREIGN KEY ([StudentID])

REFERENCES [dbo].[Student] ([StudentID]) ON DELETE CASCADE

)

We will populate the database tables with data automatically through a script that is run after the database is deployed. To do this, we need to create a **Post-Deployment Script** to the Project.

Repeat the above process of adding a new item to the Project, but, this time, the in LHS pane, under **SQL Server**, select **User Scripts**, and **Post-Deployment Script** in the main window. You can use the default name for this script. Click **Add**. (See image below)



Copy and paste the following T-SQL code to the post-deployment script – main window.. This script simply adds data to the database when no matching record is found. It does not overwrite or delete any data already entered in the database.

MERGE INTO Course AS Target

USING (VALUES

(1, 'Economics', 3),

(2, 'Literature', 3),

(3, 'Computing', 3)

)

AS Source (CourseID, Title, Credits)

ON Target.CourseID = Source.CourseID

WHEN NOT MATCHED BY TARGET THEN

INSERT (Title, Credits)

VALUES (Title, Credits);

MERGE INTO Student AS Target

USING (VALUES

(1, 'Ivanov', 'Ivan', '2012-01-24'),

(2, 'Marinova', 'Maria', '2013-01-26'),

(3, 'Petrov', 'Petar', '2012-09-20')

)

AS Source (StudentID, LastName, FirstName, EnrollmentDate)

ON Target.StudentID = Source.StudentID

WHEN NOT MATCHED BY TARGET THEN

INSERT (LastName, FirstName, EnrollmentDate)

VALUES (LastName, FirstName, EnrollmentDate);

MERGE INTO Enrollment AS Target

USING (VALUES

(1, 3.00, 1, 1),

(2, 2.80, 1, 2),

(3, 4.00, 2, 3),

(4, 3.40, 2, 1),

(5, 3.20, 3, 1),

(6, 4.00, 3, 2)

)

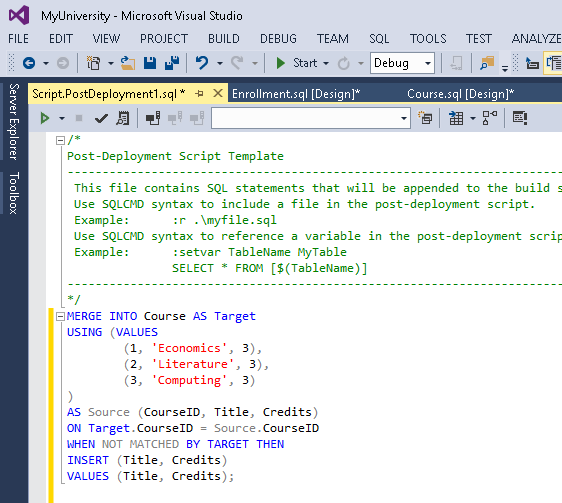
AS Source (EnrollmentID, Grade, CourseID, StudentID)

ON Target.EnrollmentID = Source.EnrollmentID

WHEN NOT MATCHED BY TARGET THEN

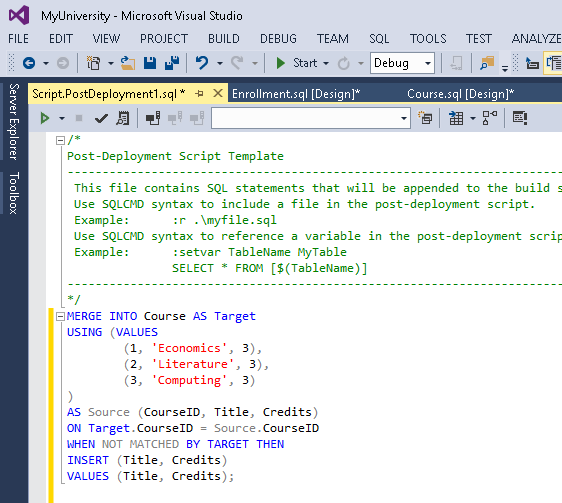
INSERT (Grade, CourseID, StudentID)

VALUES (Grade, CourseID, StudentID);

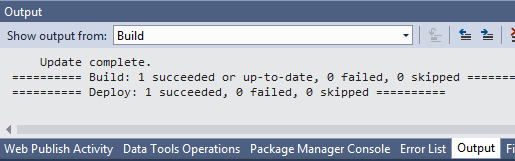


You now have created 4 SQL script files but no actual tables. You are ready to deploy your database project to **localdb** – the internal database of Visual Studio.

To build and deploy your database project, click the **Start** button – see image below.



Check the **Output** box to verify that the build and deployment succeeded. You should see the message below after a few seconds.

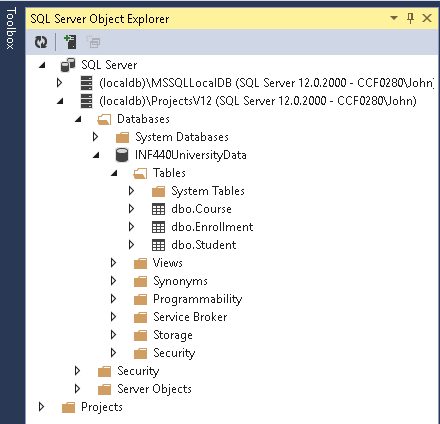


To see that the new database has been filled with data, open **SQL Server Object Explorer** and look for the name of the project in the correct local database server (in this case **(localdb)\Projects**) To open **SQL Server Object Explorer**, click **View** on the menu bar and select **SQL Server Object Explorer.**

Open **SQL Server > (localdb)\Projects … > Databases > INF440UniversityData > Tables**

To see that the tables are populated with data, right-click a table, and select **View Data**. An editable view of the table data is displayed. (See image below)

The database is now set up and populated with data. In the next section, you will create a web application for the database.



## Select File > Save All from the main menu.

## 2. Create a new ASP.NET Web Application

Right-click on the **Solution** **‘MyUniversity’** in the **Solution Explorer** window, and select **Add > New Project**.

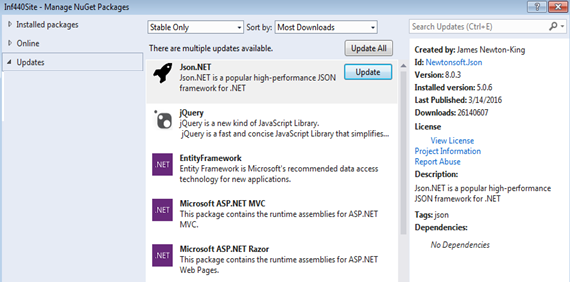
In the next window that appears, select **Visual C#** and **.NET Framework 4.5**. Then select the **ASP.NET Web Application** template. Name the Project **INF440Site**. Click **OK**.

In the **New ASP.NET Project** window that appears next, select the **MVC** template. Click **OK**.

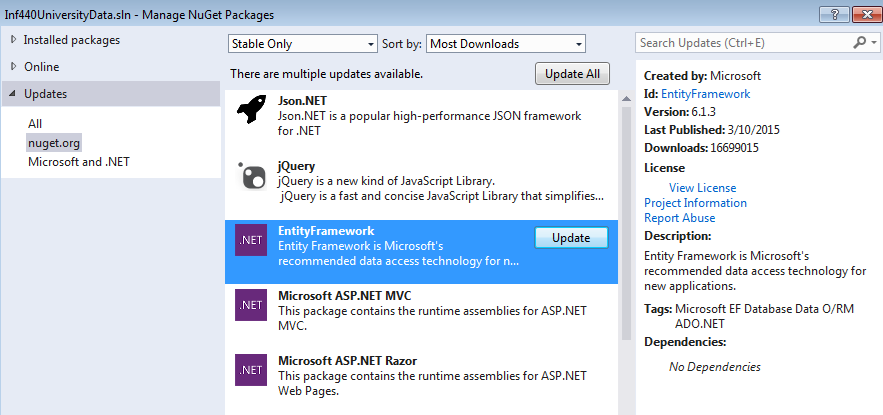
A new project is created with the default files and folders. Close the **Congratulations** page. (You may also close **SQL Server Object Explorer**, and the **Student, Course, Enrollment**, and **Post-Deployment Script** SQL files.)

For this lab, you need to use **Entity Framework, version 6.1.3 or higher**. Before going any further, double-check the version of **Entity Framework** in your Project through the **Manage NuGet Packages** window.

To do this, right click on the **Project INF440Site** and select **Manage NuGet Package for Solution**. Select **Installed Packages** in the LHS pane. Then select the **Entity Framework** package from the list of options. The version number is listed in the RHS window pane. If necessary, update your version of Entity Framework.



To do this, click **Updates** in the LHS window pane. In the middle window pane, select **Entity Framework**, and click its **Update** button. Accept the Microsoft licence terms.



Do the same for **Microsoft ASP.NET MVC**. The version number should be **5.2.3 or higher**

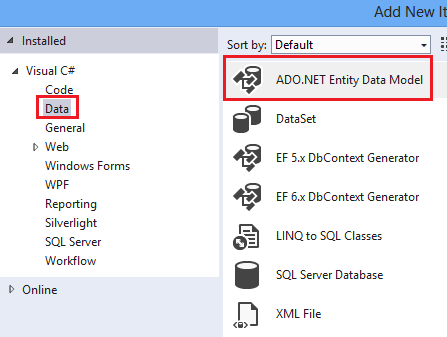
## 3. Create the MVC Models

You will now create **MVC** Models from the database tables using **Entity Framework**. These Models are classes that you will use to work with the database data. Each Model mirrors a table in the database and contains object properties, which are accessed via get() and set() methods, that correspond to the columns in the table.

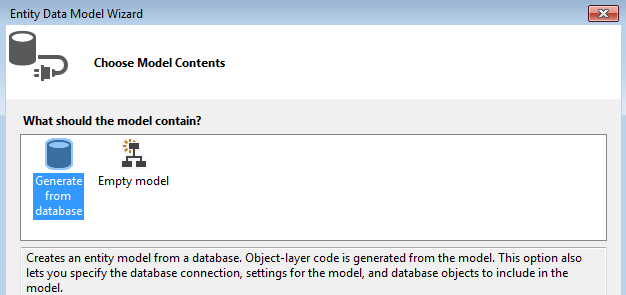
First, we need to connect the database to the project.

Right-click the **Models** folder, and select **Add >** **New Item**.

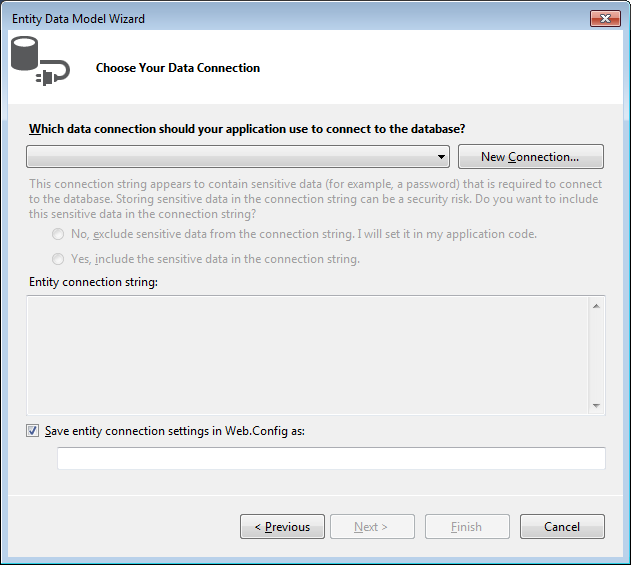
In the new window, select **Data** in the LHS pane and **ADO.NET Entity Data Model** from the options in the center pane. Name the new Model file **INF440Model.edmx** Click **Add**.



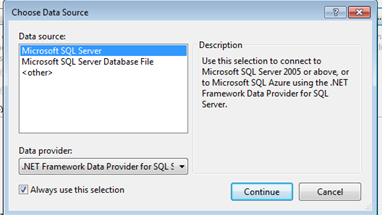
In the **Entity Data Model Wizard** window that appears, select the **Generate from Database** option, and click **Next**.



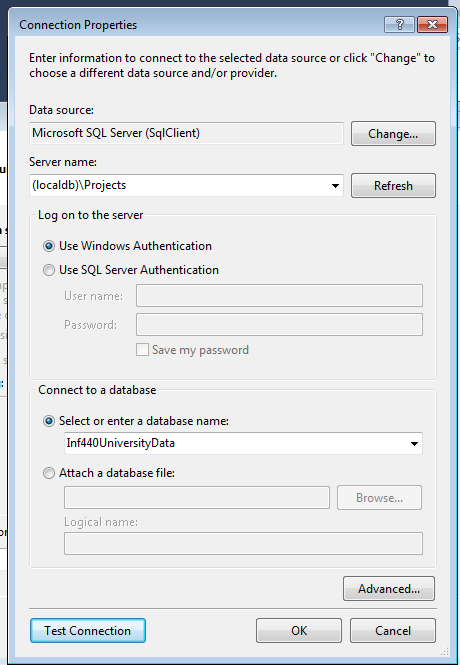
A **Choose Data Connection** window is displayed. Click the **New Connection** button



A **Choose Data Source** window may appear. If so, select the **Microsoft SQL Server** option. The Data Provider should be **.NET Framework Data Provider for SQL Server**. Click **Continue**.

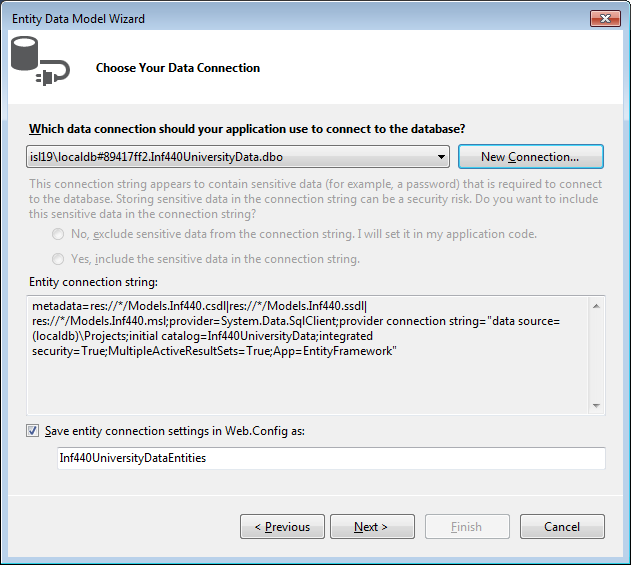


Otherwise, a **Connection Properties** window is displayed., Here, in the **Server Name** box, type  **(localdb)\Projects**. For the **Select a Database Name** box, use the pulldown menu to select **INF440UniversityData**.



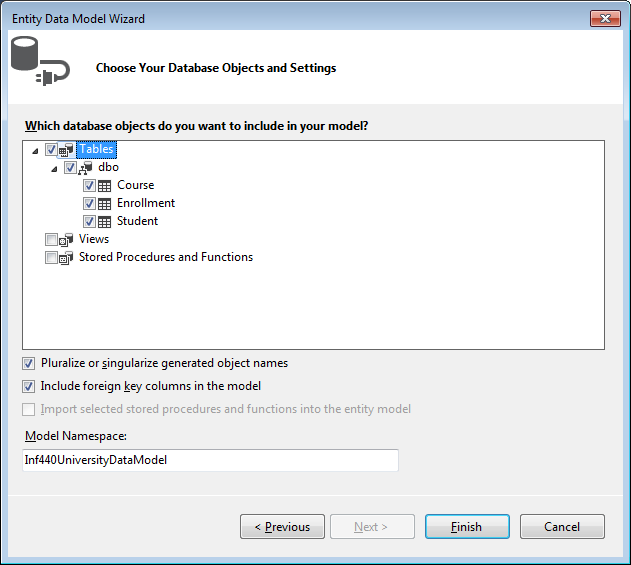
Click **OK**

The correct connection properties are now displayed. You can use the default name for connection in the Web.Config file. Just accept all the defaults.



Click **Next**.

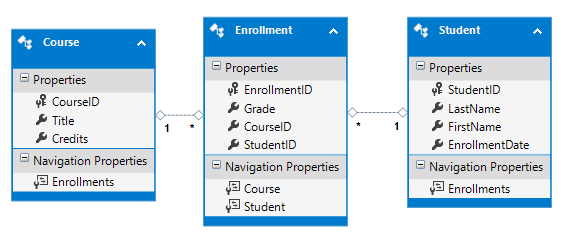
Select **Tables** to generate models for all three tables, **and click the checkboxes next to each table.**



Click **Finish**.

If you receive security warnings, select **OK** to continue running the template.

The models are generated from the database tables, and a diagram is displayed that shows the properties and relationships between the tables.



The **Models** folder now includes many new files related to the models that were generated from the database.

The **Inf440Model.Context.cs** file contains a class that derives from the **Entity Framework** **DbContext** class, and provides a property for each model class that corresponds to a database table. The **Course.cs**, **Enrollment.cs**, and **Student.cs** files contain the model classes that represent the databases tables. You will use both the context class and the model classes when working with **ASP.NET Scaffolding**.

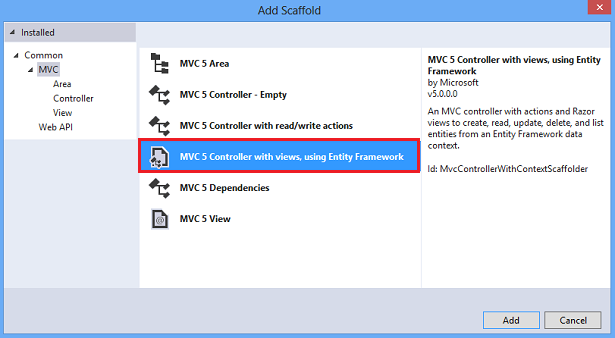
Before proceeding with this exercise, **Build the Solution**. In the next section, you will generate code based on the data models, but that section will not work if the **Solution** has not been built.

## 4. Generate Controllers and Views with Scaffolding

You are now ready to generate code that will provide standard data operations for the model classes. You add the code by adding a so-called **scaffold** item. There are many options for the type of scaffolding you can add. For this lab, the scaffold will include a **Controller** and **Views** that correspond to the **Student** and **Enrollment models** you created in the previous section.

To maintain consistency in your project, you will add the new **Controller** to the existing **Controllers** folder. Right-click the **Controllers** folder, and select **Add** **>** **New Scaffolded Item**.

Select the **MVC** option in the LHS pane. Then select the **MVC 5 Controller with views, using Entity Framework** option in the main pane.. This option will generate the controller and views for updating, deleting, creating and displaying the data in your model.

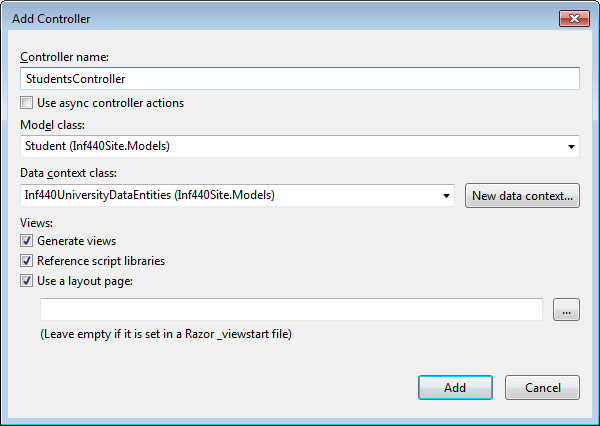


Click **Add**.

If you receive an error, it may be because you did not build the project in the previous section. If so, try building the project, and then add the scaffold item again.

In the **Add Controller** window, change the **Controller name** to **StudentsController**,

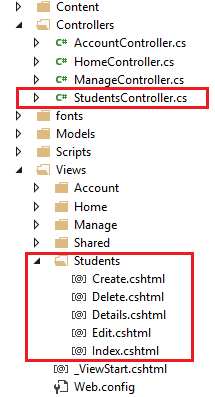
For the **Model class**, use the pull-down menu to select **Student (Inf440SiteModels)**. For the Data Context class, use the pull-down menu to select **INF440UniversityDataEntities (INF440SiteModels)**. (See image below) Click **Add.**

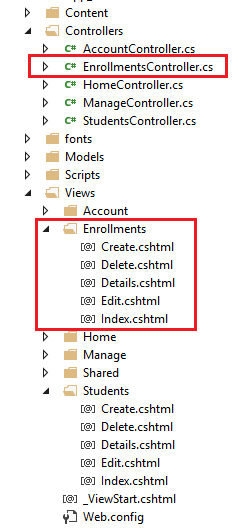


Perform the same steps again, but In the **Add Controller** window, change the **Controller name** to **EnrollmentsController**,

For the **Model class**, use the pull-down menu to select **Enrollment (INF440SiteModels)**. For the Data Context class, use the pull-down menu to select **INF440UniversityDataEntities (INF440SiteModels)**. Click **Add.**

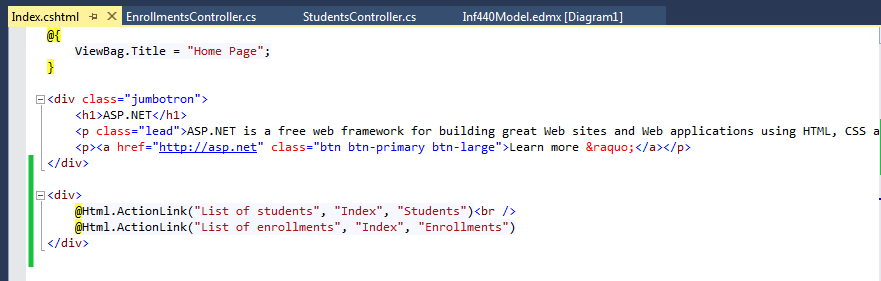
When finished, you should have **StudentsContoller.cs** and **EnrollmentsController.cs** files, and folders under **Views** named **Students** and **Enrollments** with the **Create, Delete, Details, Edit and Index views** as shown below.





**5. Add links to new views**

To make it easier for you to navigate to your new views, you can add a couple of hyperlinks to the **Index View** for Students and Enrollments. Open the file at **Views/Home/Index.cshtml**, which is the home page for your site. Add the following Razor code below the “**jumbotron**” div.



**<div>**

**@Html.ActionLink("List of students", "Index", "Students")<br />**

**@Html.ActionLink("List of enrollments", "Index", "Enrollments")**

**</div>**

For the **ActionLink** method of the first statement, the first parameter is the text to display in the link. The second parameter is the **Action** method, and the third parameter is the name of the controller. For example, the first link points to the **Index() Action** method in **StudentsController**. The actual hyperlink is constructed from these values. The first link ultimately takes users to the **Index.cshtml** file within the **Views/Students** folder.

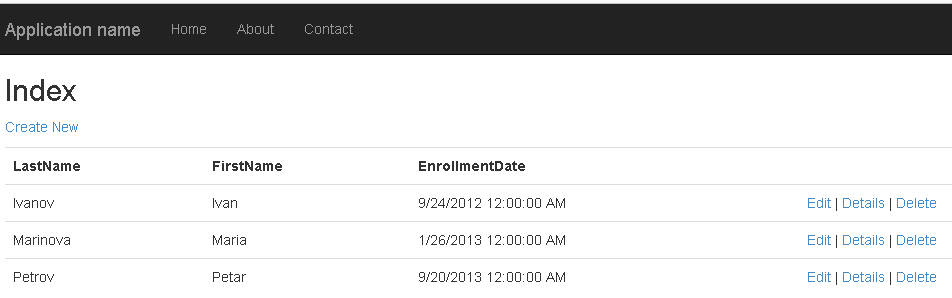
The second statement does the same for Enrollments.

**6. Display student views**

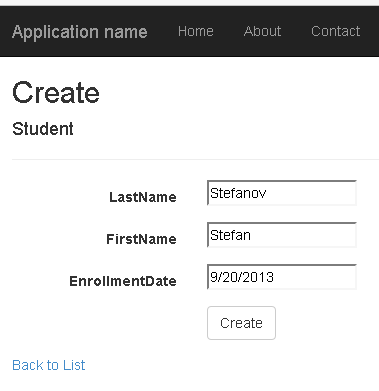
You can verify that the code added to your project correctly displays a list of the students, and enables users to edit, create, or delete the student records in the database.

Right-click the **Views/Home/Index.cshtml** file, and select **View in Browser**. You should see a web page with two links, List of students and List of enrollments. On this page, click the link for the list of students.

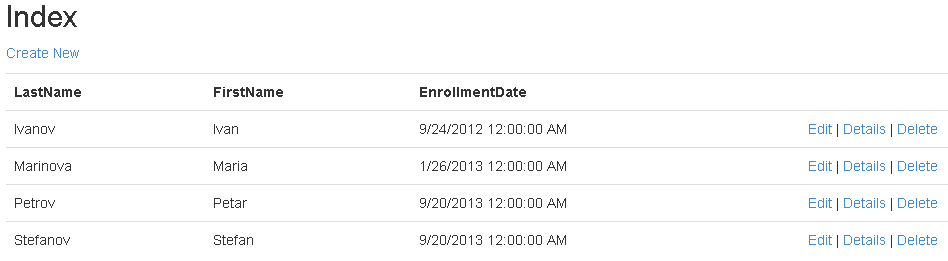
On the page which next displayed, notice the list of the students and links to modify this data.



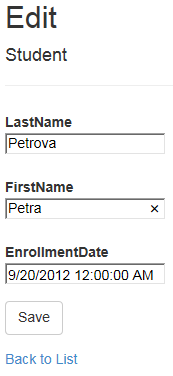
Click the **Create New** link and provide some values for a new student.



Click **Create**, and notice the new student is added to your list.

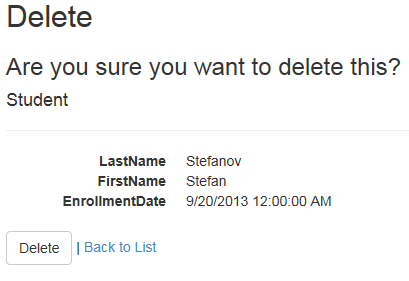


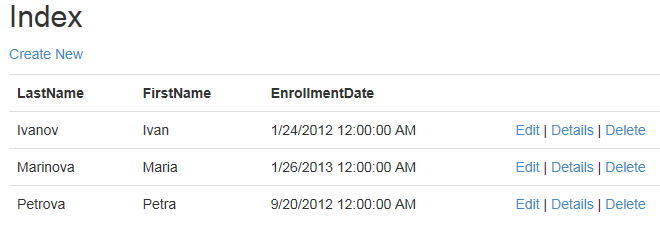
Select the **Edit** link, and change some of the values for a student.



Click **Save**, and notice the student record has been changed.

Finally, select the **Delete** link and confirm that you want to delete the record by clicking the **Delete** button.





**Conclusion**

If you have got this far, you have successfully completed this lab exercise, and created an MVC Database-First MVC architecture-based application. Congratulations!

Without writing any code, you have added views that perform common operations on the data in the **Student** table.