Complete C# Tutorial

Insert Update Delete In ASP.NET MVC 5 Without Entity Framework

In this article you will learn:

1. ☑ *ASP.NET MVC 5 with ADO.Net Connectivity*
2. ☑ *CRUD – CREATE, READ, UPDATE and DELETE Operation*

In this article you are going to learn how to **Insert, Update and Delete Records in ASP.NET MVC 5 without using heavy entity framework**. This article uses pure ADO.Net commands to work with SQL database. You will learn MVC 5 application with ado.net connectivity with complete pictorial guide and step by step examples. Once you complete this article, I promise, you will be able to write your own customize code in MVC 5. If you are beginners or trying to learn MVC, then don't skip this tutorial, because this article contains complete programming example with screen shot. As I understand the need of beginners that they require complete programming example instead of part of the code.

LET'S START ASP.NET MVC 5 WITH CRUD OPERATION

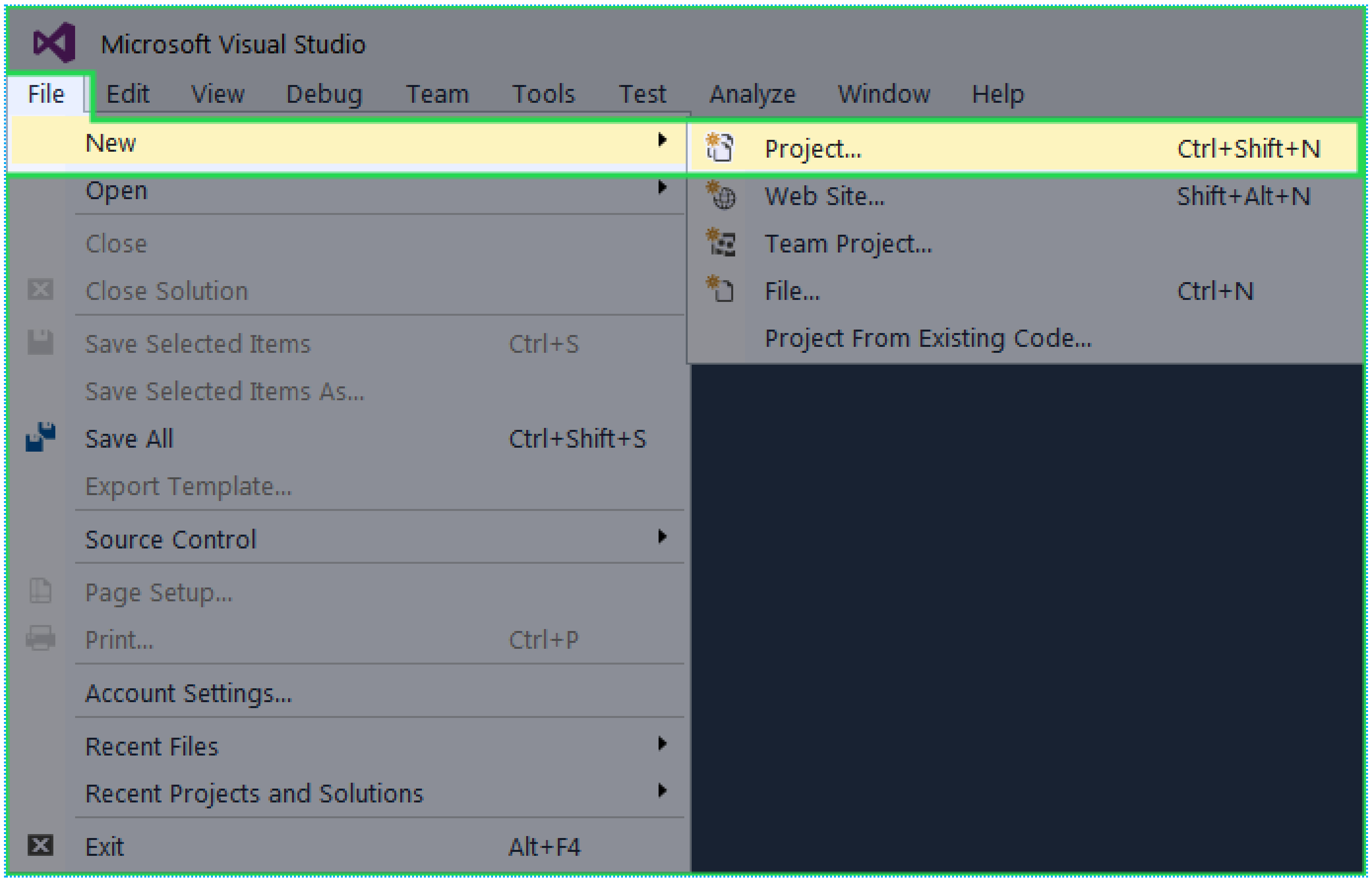
In this example, I am going to create a simple student registration form in MVC5 using CRUD (Create, Read, Update and Delete) Operation. In this tutorial I haven't use heavy entity framework and gives you way to write transparent ADO.NET Connectivity.

I have divided the entire article in following parts.
Task:

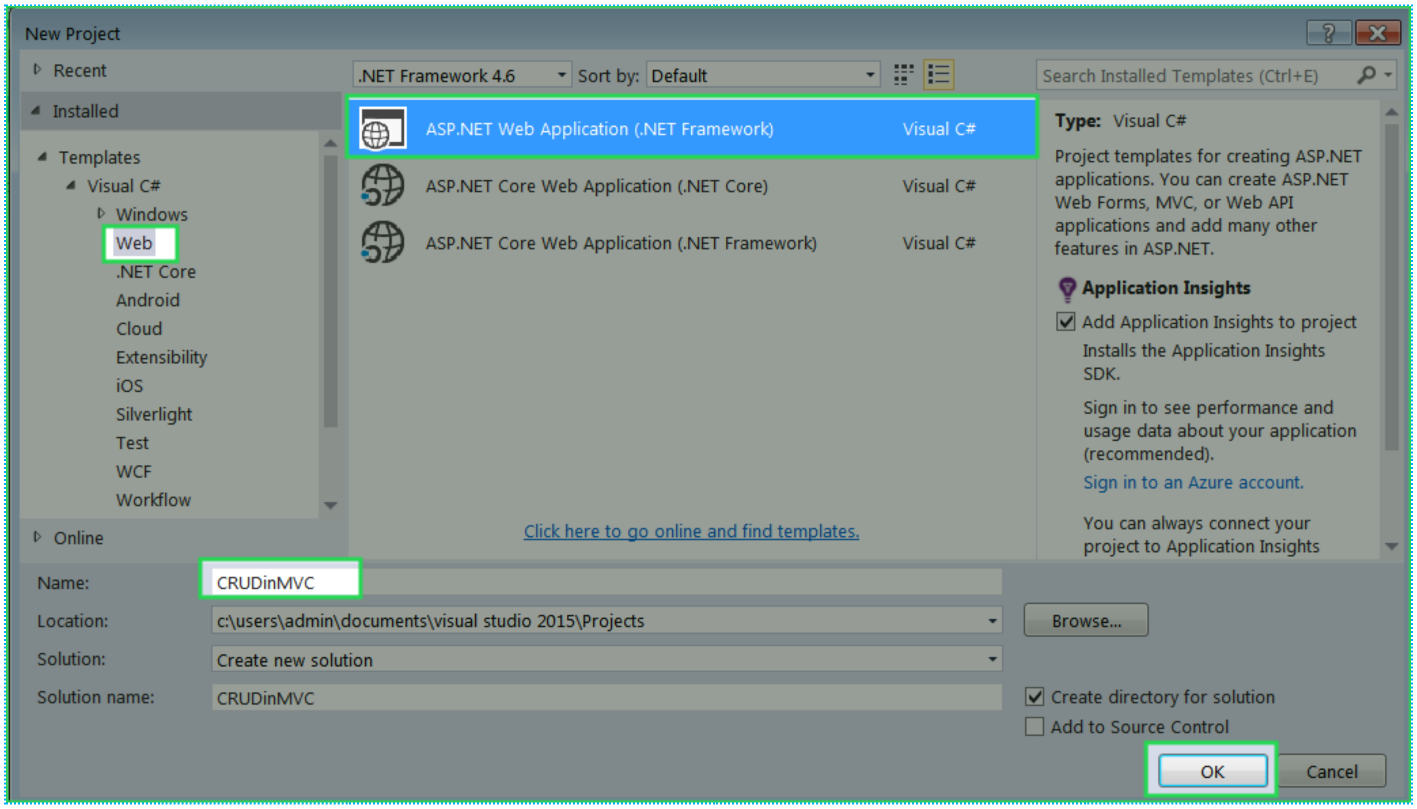
1. Creating New ASP.NET MVC5 Project
2. Creating Model Classes.
3. Creating Controller
4. Creating Database, Table and Store Procedure
5. Adding Connection String to Web.Config File.
6. Creating a class file for handling all the databases operations.
7. Adding Action Method in Controller.
8. Creating Partial View from Action Method
9. Configure RouteConfig.cs file
10. Run your Project

1. Create New ASP.NET MVC 5 Project

1. Open **Visual Studio 2015**. Click on **File ➔ New ➔ Project**.



2. Select **Web** in the template section and select **ASP.NET Web Application**. Give name **CRUDinMVC** and click **OK**.


New Project
Recent
Installed
Templates
Visual C#
Windows
Web
.NET Core
Android
Cloud
Extensibility
iOS
Silverlight
Test
WCF
Workflow
Online
.NET Framework 4.6
Sort by: Default
Search Installed Templates (Ctrl+E)
ASP.NET Web Application (.NET Framework) Visual C#
ASP.NET Core Web Application (.NET Core) Visual C#
ASP.NET Core Web Application (.NET Framework) Visual C#
Click here to go online and find templates.
Application Insights
Sign in to an Azure account.
Browse...
OK
Cancel

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.

☑ Add Application Insights to project

Installs the Application Insights SDK.

Sign in to see performance and usage data about your application (recommended).

You can always connect your project to Application Insights

Name: CRUDinMVC

Location: c:\users\admin\documents\visual studio 2015\Projects

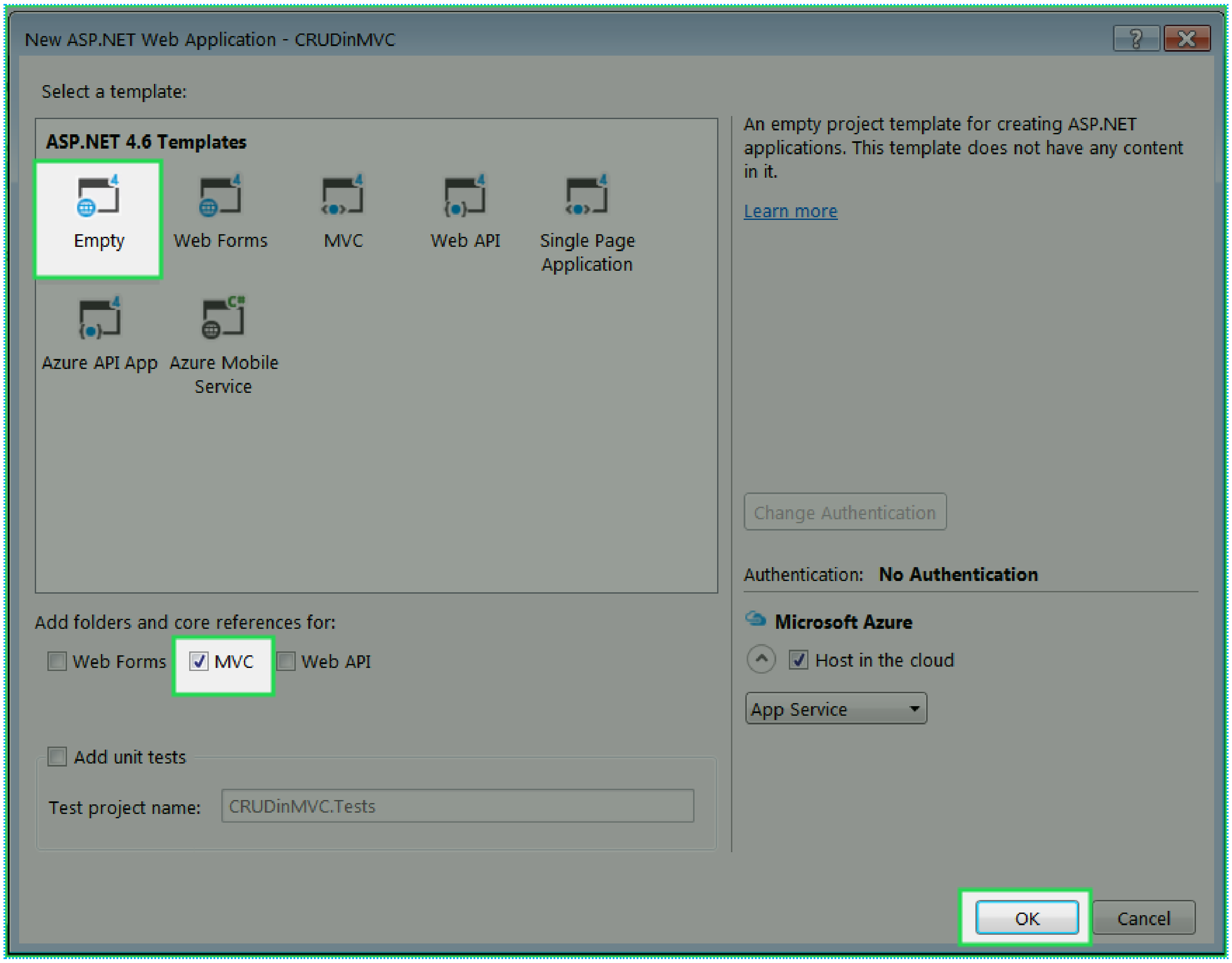
Solution: Create new solution

Solution name: CRUDinMVC

☑ Create directory for solution

☐ Add to Source Control

3. A **template** windows will open. Select **Empty** template and **check MVC checkbox** as described in the given picture.


New ASP.NET Web Application - CRUDinMVC

Select a template:

ASP.NET 4.6 Templates

Empty, Web Forms, MVC, Web API, Single Page Application, Azure API App, Azure Mobile Service

An empty project template for creating ASP.NET applications. This template does not have any content in it.

Learn more

Change Authentication

Authentication: **No Authentication**

Add folders and core references for:

☐ Web Forms ☑ MVC ☐ Web API

☐ Add unit tests

Test project name: CRUDinMVC.Tests

Microsoft Azure

☑ Host in the cloud

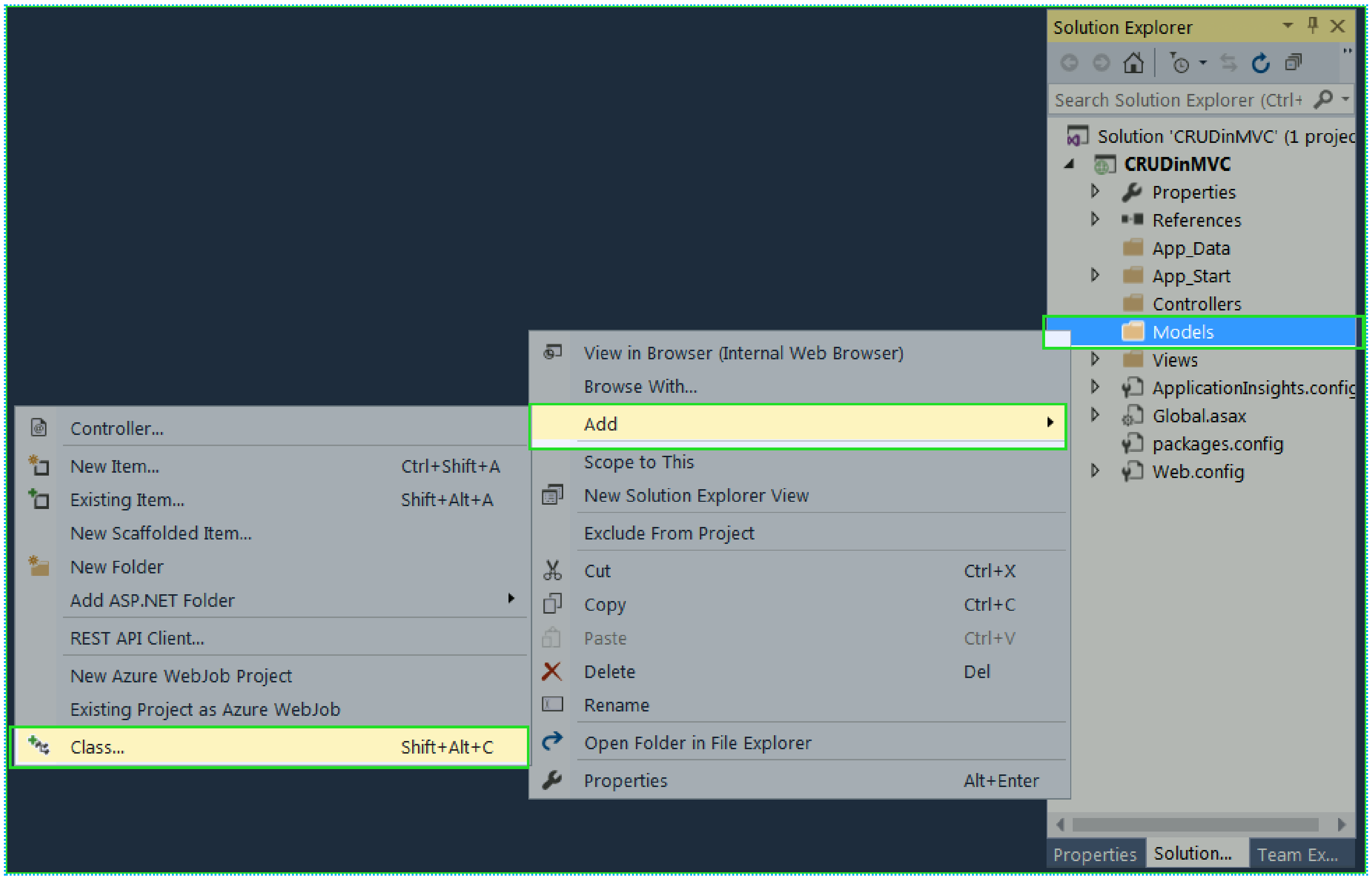
App Service

OK Cancel

4. Your project will open for develop. If you see *AZURE Hosting window* before opening the project, gently **cancel** it.

2. Create Model Class

1. Right click on **Models** Folder ➔ **Add** ➔ **Class**. Select **class** in middle pane and create new class `StudentModel.cs`.

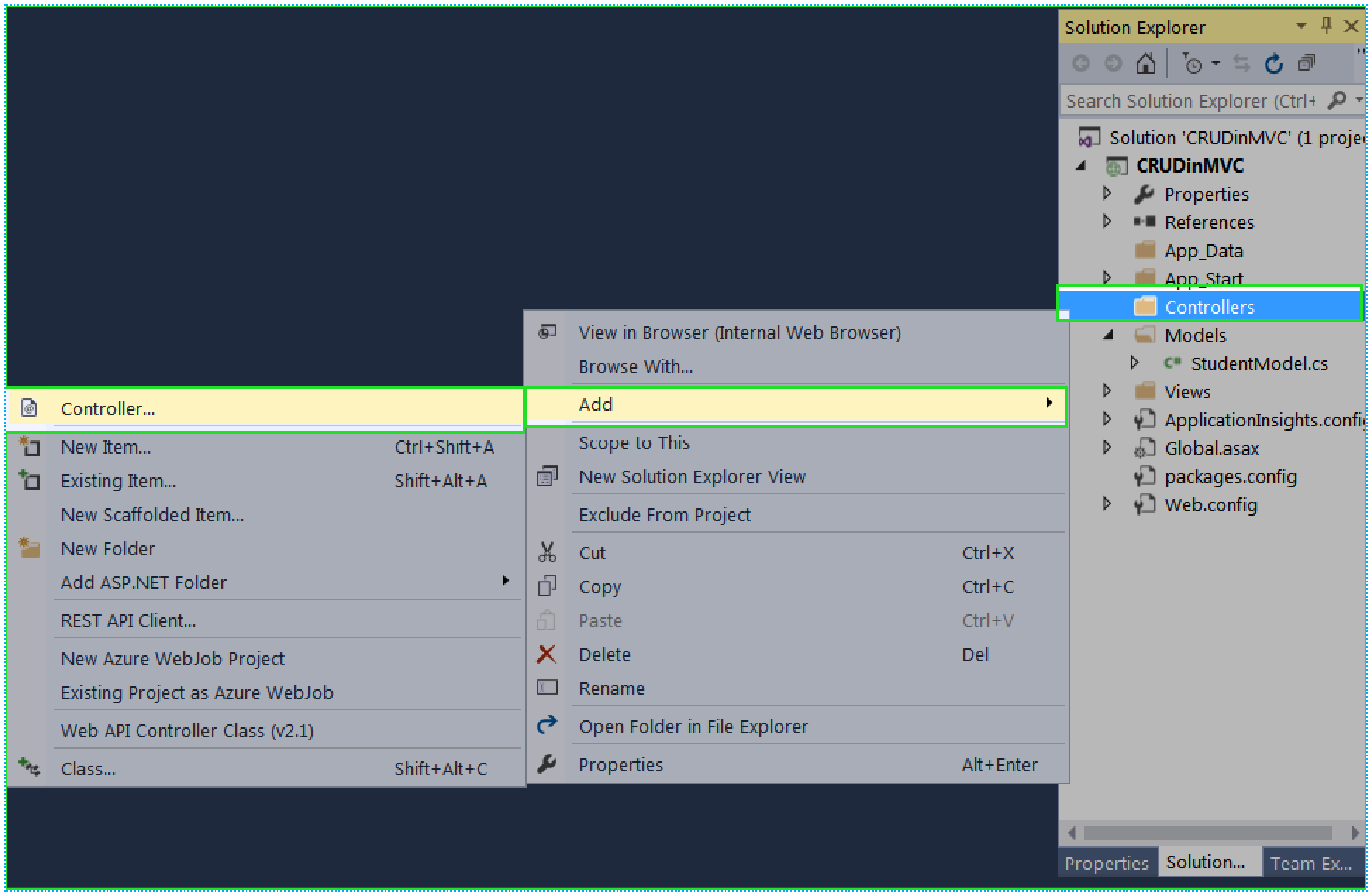


2. Your `StudentModel.cs` class should be look like this code snippets.

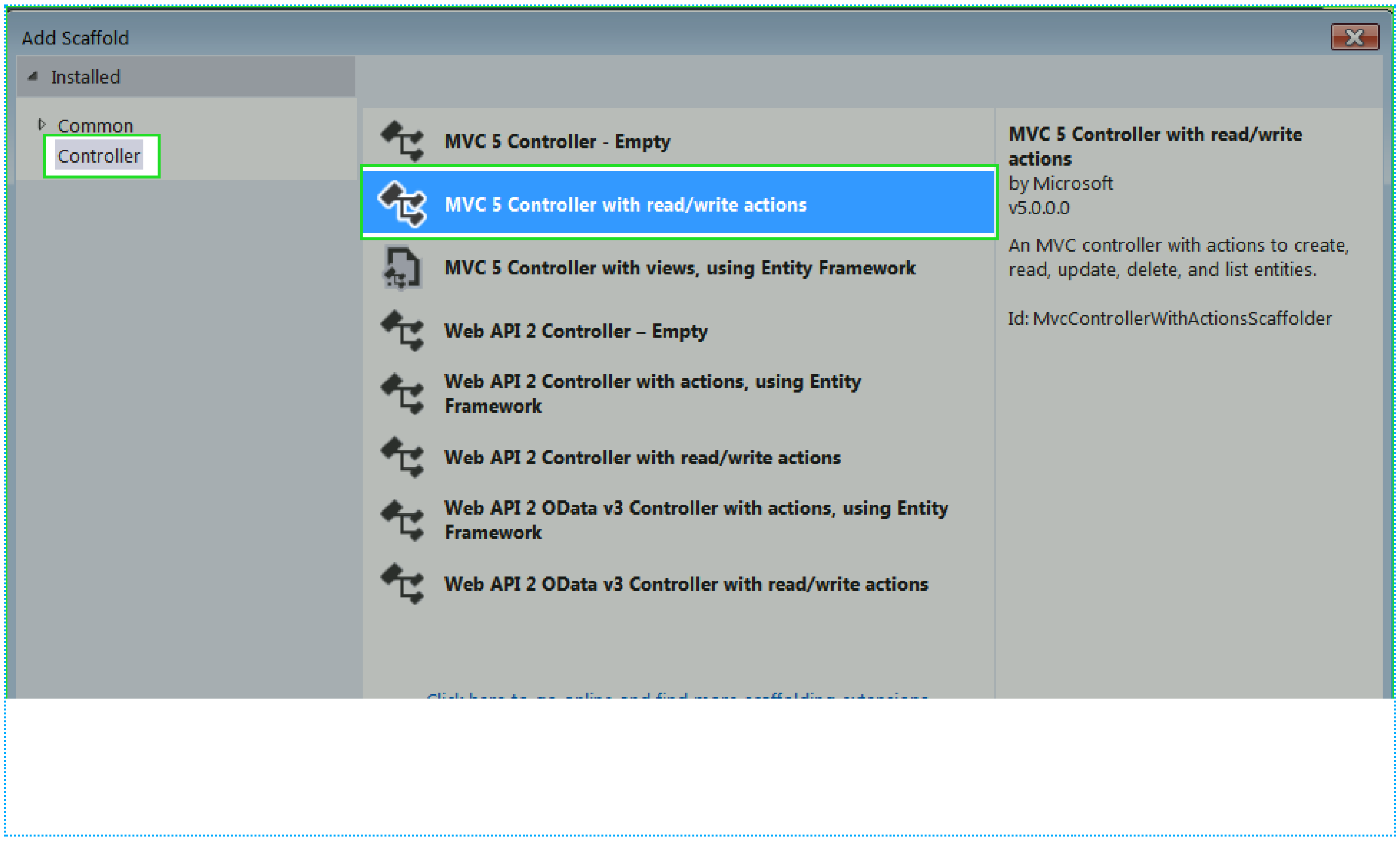
```
using System.ComponentModel.DataAnnotations;

namespace CRUDinMVC.Models
{
    public class StudentModel
    {
        [Display(Name = "Id")]
        public int Id { get; set; }

        [Required(ErrorMessage = "First name is required.")]
        public string Name { get; set; }

        [Required(ErrorMessage = "City is required.")]
        public string City { get; set; }

        [Required(ErrorMessage = "Address is required.")]
        public string Address { get; set; }
    }
}
```

3. Create Controller

1. **Right** Click on **Controllers** Folder ➔ **Add** ➔ **Controller**.



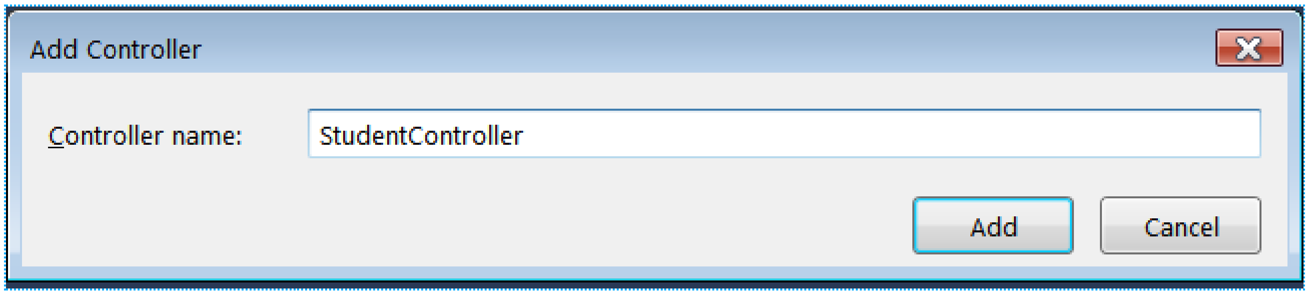
2. **Scaffold** window will open. Here, choose **MVC 5 Controller with read/write actions** and click **Add** button.



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 read/write actions
by Microsoft
v5.0.0.0
An MVC controller with actions to create, read, update, delete, and list entities.
Id: MvcControllerWithActionsScaffolder

3. Give name **StudentController** and click **Add**.



Controller name: StudentController

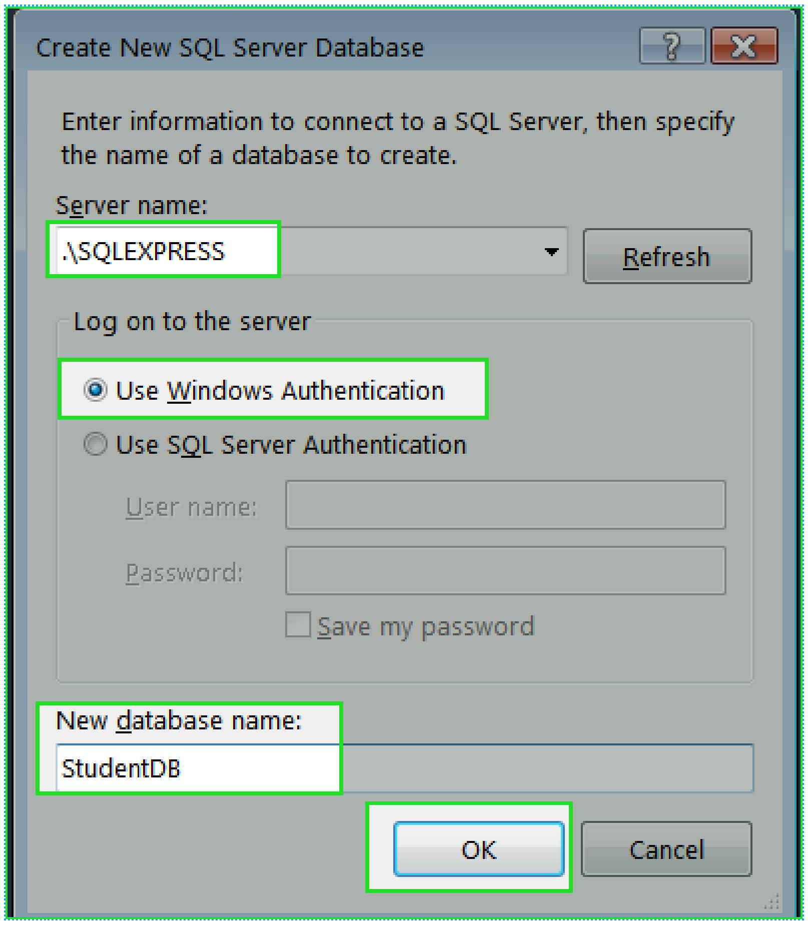
4. Create Database, Table and Store Procedure

CREATE OR SELECT DATABASE

1. Open **Server Explorer**. Go to **View ➔ Server Explorer**.

2. Right Click on **Data Connections ➔ Create New SQL Server Database**.

3. Create `StudentDB` database as the picture below.



Create New SQL Server Database

Enter information to connect to a SQL Server, then specify the name of a database to create.

Server name: .\SQLEXPRESS

Log on to the server

[✓ Use Windows Authentication]

Use SQL Server Authentication

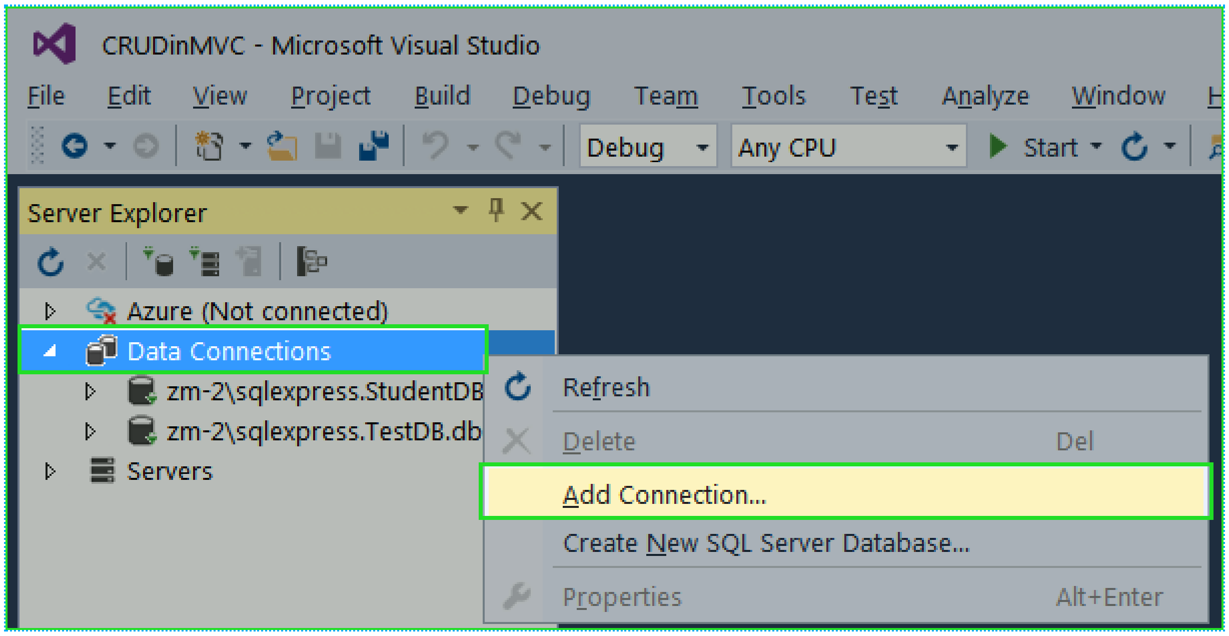
User name:

Password:

Save my password

New database name: StudentDB

4. If you have already a database then you can make connection with existing database. **Right Click** on **Data Connections → Add Connection**



CRUDinMVC - Microsoft Visual Studio

Server Explorer

Azure (Not connected)

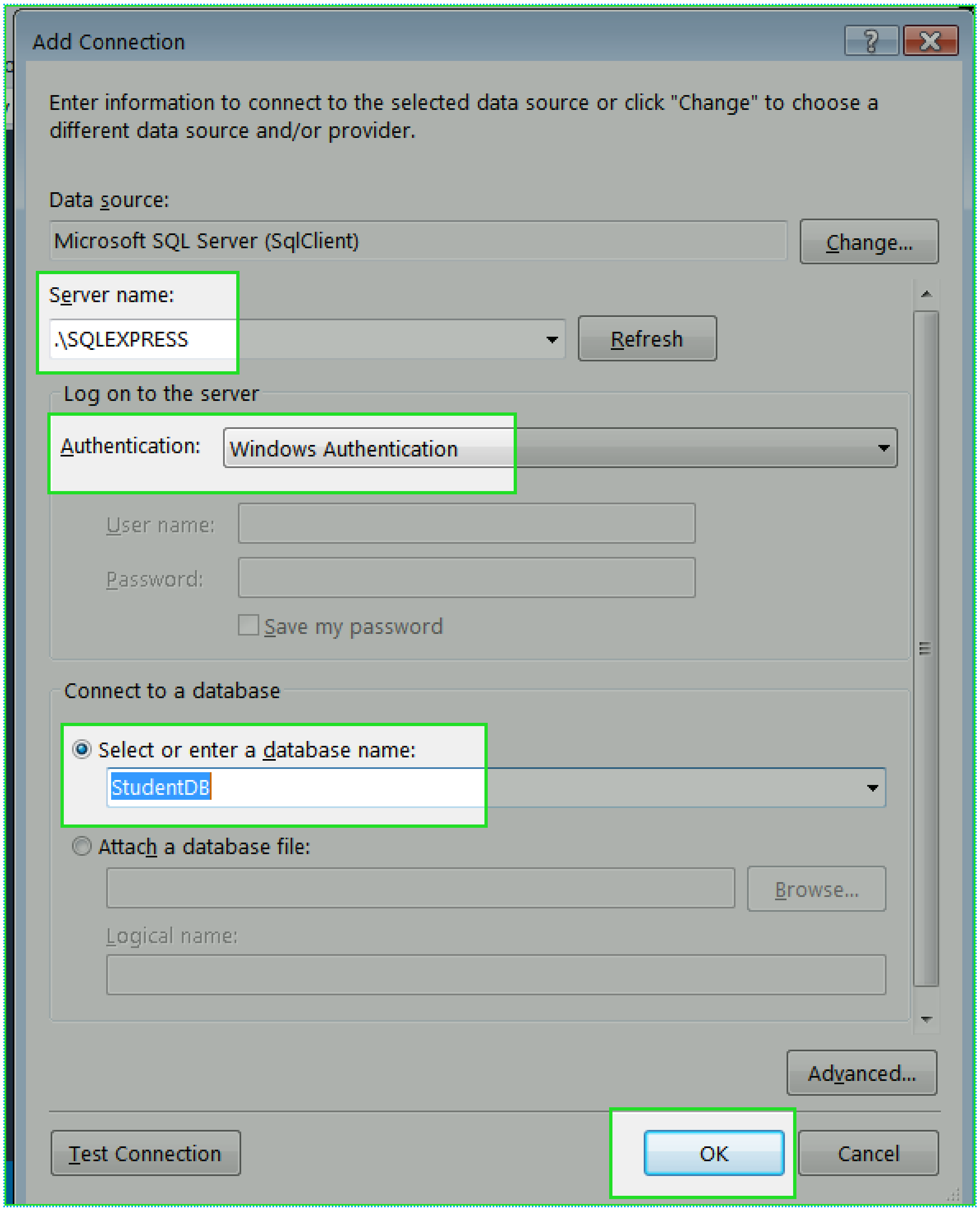
Data Connections

zm-2\sqlexpress.StudentDB

zm-2\sqlexpress.TestDB.db

Servers

5. Select database as picture below


Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:
Microsoft SQL Server (SqlClient)

Server name:
.\SQLEXPRESS

Log on to the server

Authentication: Windows Authentication

User name:

Password:

Save my password

Connect to a database

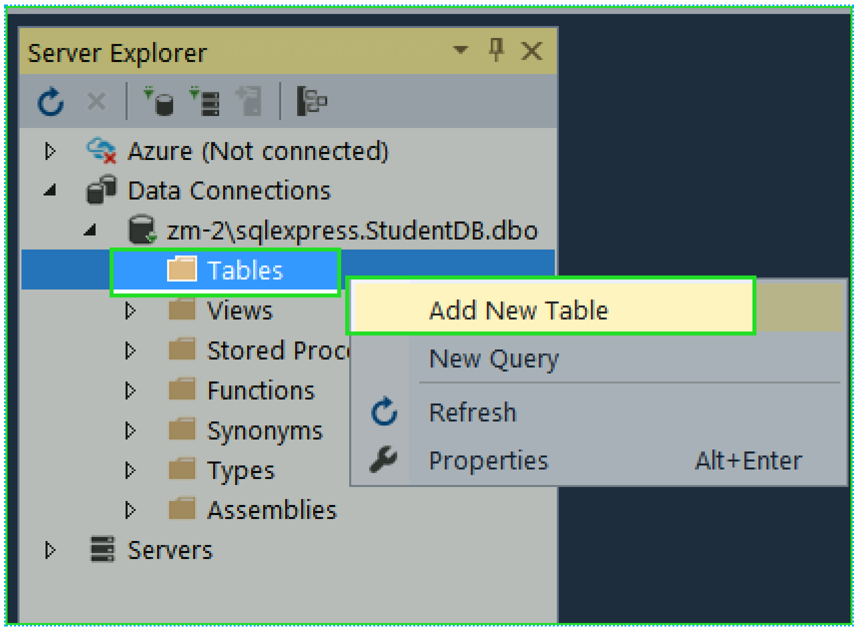
Select or enter a database name:
StudentDB

Attach a database file:

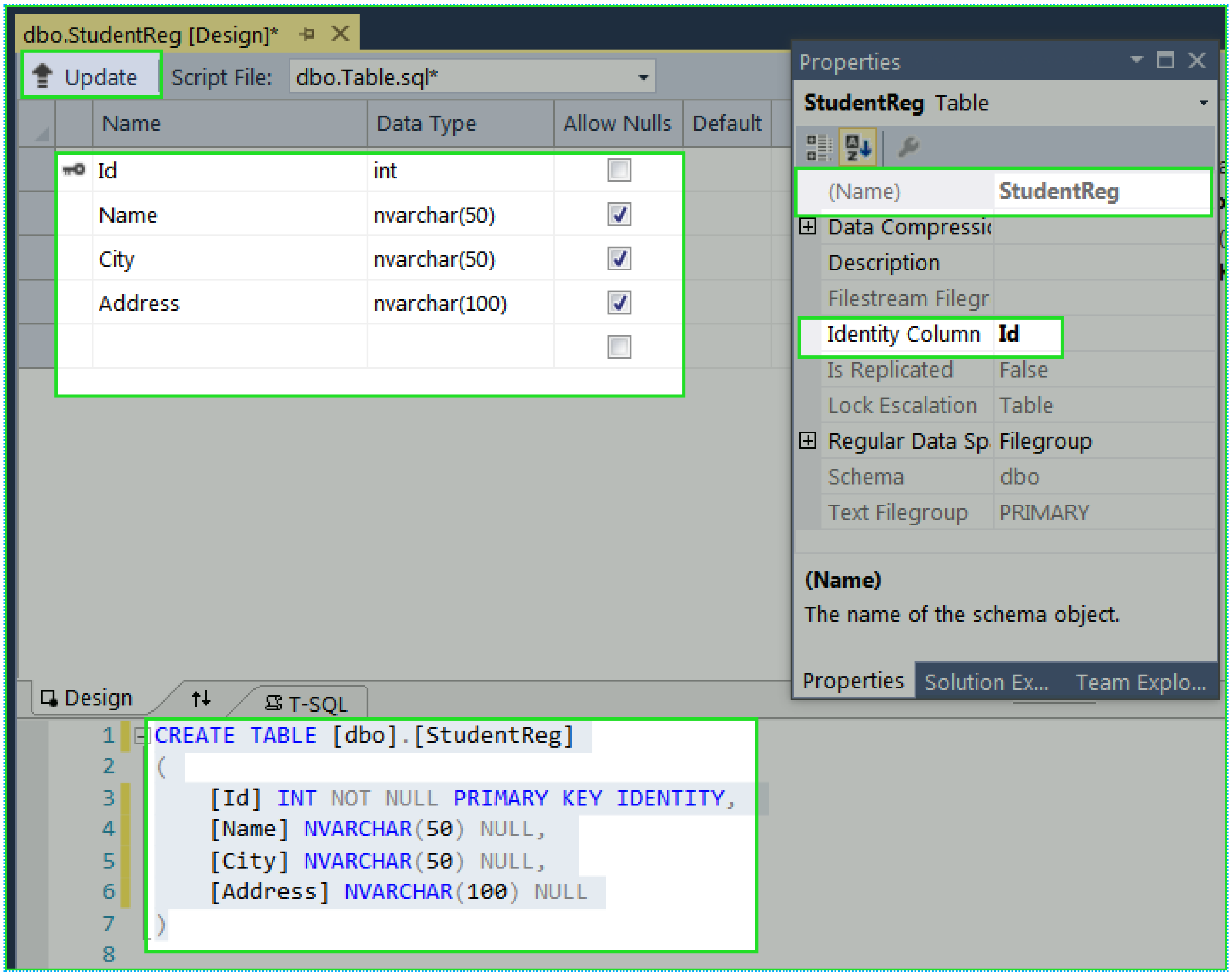
Logical name:

CREATE `StudentReg` TABLE

1. Now create a Table `StudentReg` in database according to models. Expand `StudentDB` database in **Server Explorer** and **Right Click** on **Table** and then select **Add New Table**.



2. Make Table exactly the same as described in the picture. Go to **Table Properties** and must set **Id** in **Identity column** and **StudentReg** in **Name** Column.


dbo.StudentReg [Design]*

Update Script File: dbo.Table.sql*

Name	Data Type	Allow Nulls	Default
Id	int	☐	
Name	nvarchar(50)	✓	
City	nvarchar(50)	✓	
Address	nvarchar(100)	✓	
		☐	

Properties

StudentReg Table

(Name)	StudentReg
Data Compressi	
Description	
Filestream Filegr	
Identity Column	Id
Is Replicated	False
Lock Escalation	Table
Regular Data Sp	Filegroup
Schema	dbo
Text Filegroup	PRIMARY

(Name)
The name of the schema object.

```sql
CREATE TABLE [dbo].[StudentReg]
(
    [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [Name] NVARCHAR(50) NULL,
    [City] NVARCHAR(50) NULL,
    [Address] NVARCHAR(100) NULL
)
```

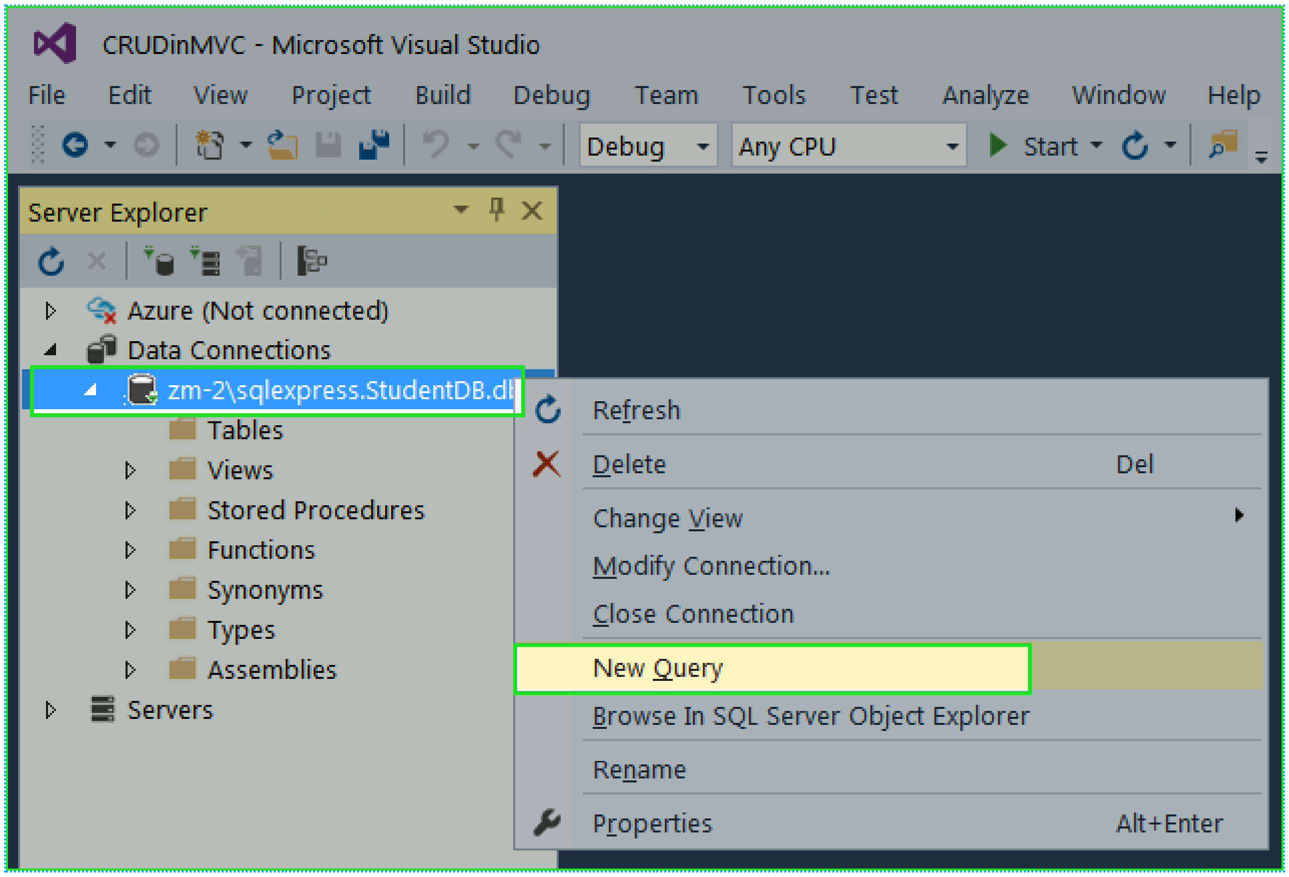
3. Table Scripts

```sql
CREATE TABLE [dbo].[StudentReg]
(
 [Id] INT NOT NULL PRIMARY KEY IDENTITY,
    [Name] NVARCHAR(50) NULL,
    [City] NVARCHAR(50) NULL,
    [Address] NVARCHAR(100) NULL
)
```

4. After finishing table designing click on **Update** button to save table.

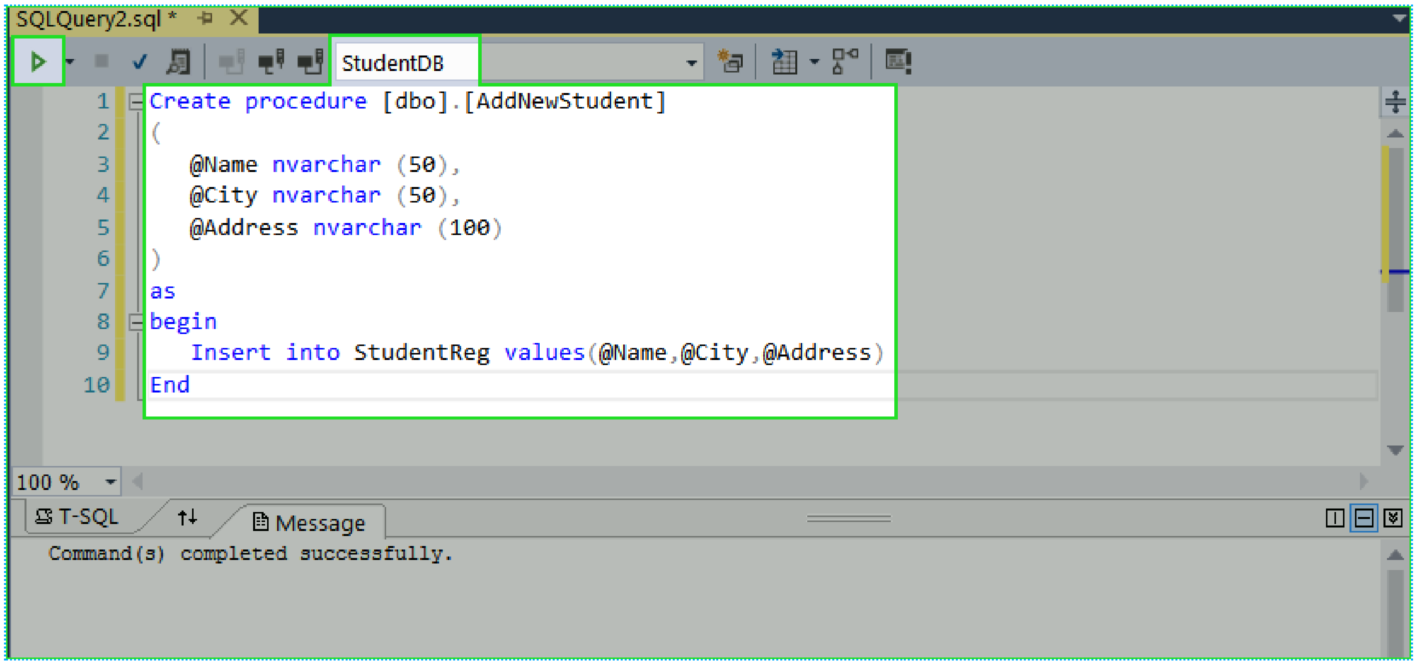
> **Create Store Procedure for `CRUD` – `Create`, `Read`, `Update` and `Delete` Operation.**

Open **New Query** Window. **Right** Click on database name in **Server Explorer** and select **New Query**. Now, Paste following code snippets one by one and execute code by pressing `Ctrl + Shift + E` or clicking on **Execute** ▶ icon on the top left corner of query window.



1. Insert Records – Store Procedure

```
Create procedure [dbo].[AddNewStudent]
(
   @Name nvarchar (50),
   @City nvarchar (50),
   @Address nvarchar (100)
)
as
begin
   Insert into StudentReg values(@Name,@City,@Address)
End
```



```sql
Create procedure [dbo].[AddNewStudent]
(
   @Name nvarchar (50),
   @City nvarchar (50),
   @Address nvarchar (100)
)
as
begin
   Insert into StudentReg values(@Name,@City,@Address)
End
```

```
Command(s) completed successfully.
```

2. View Added Records – Store Procedure

```sql
Create Procedure [dbo].[GetStudentDetails]
as
begin
   select * from StudentReg
End
```

3. Update Records – Store Procedure

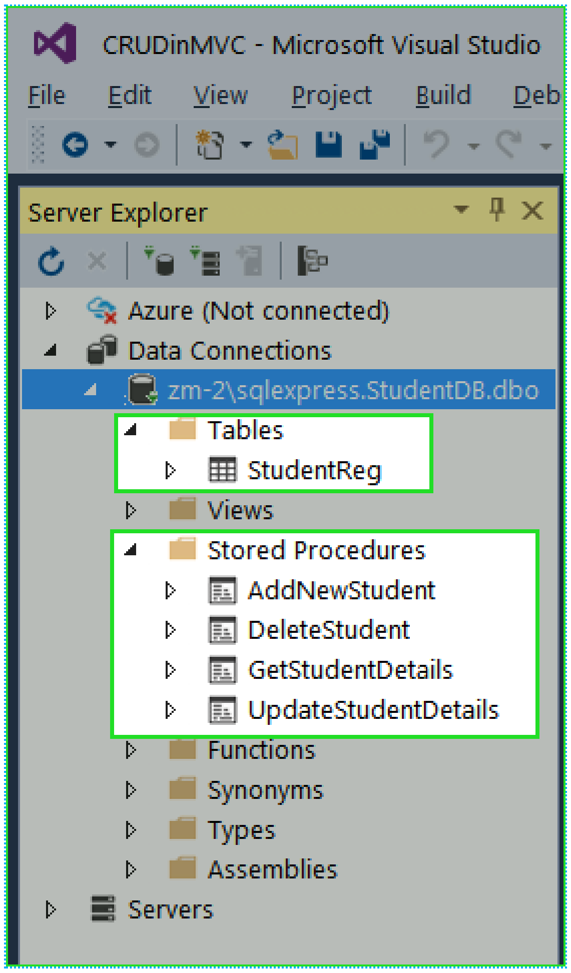
```sql
Create procedure [dbo].[UpdateStudentDetails]
(
   @StdId int,
   @Name nvarchar (50),
   @City nvarchar (50),
   @Address nvarchar (100)
)
as
begin
   Update StudentReg
   set Name=@Name,
   City=@City,
   Address=@Address
   where Id=@StdId
End
```

4. Delete Records – Store Procedure

```sql
Create procedure [dbo].[DeleteStudent]
(
   @StdId int
)
as
begin
```

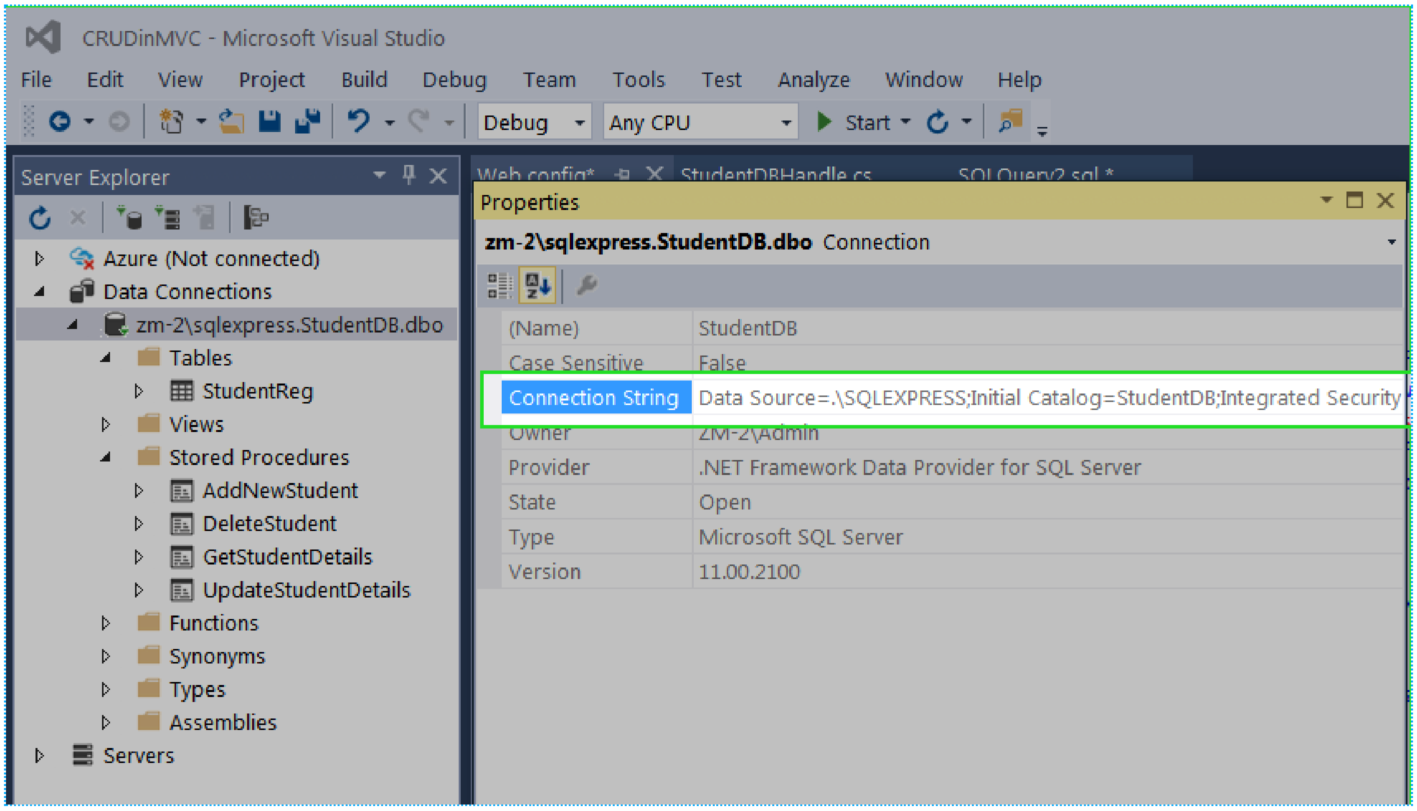
```
    Delete from StudentReg where Id=@StdId
End
```

After Creating all **Store Procedure** your **Server Explorer** will look like:



5. Add Connection String in Web.config file.

1. Add Connection String in **Web.config** file. You can find connection string in the database properties. **Right** click on **Database Name** and Select **Properties**. Copy Connection String and keep it.



CRUDinMVC - Microsoft Visual Studio

Server Explorer

```
Azure (Not connected)
Data Connections
  zm-2\sqlexpress.StudentDB.dbo
    Tables
      StudentReg
    Views
    Stored Procedures
      AddNewStudent
      DeleteStudent
      GetStudentDetails
      UpdateStudentDetails
    Functions
    Synonyms
    Types
    Assemblies
Servers
```

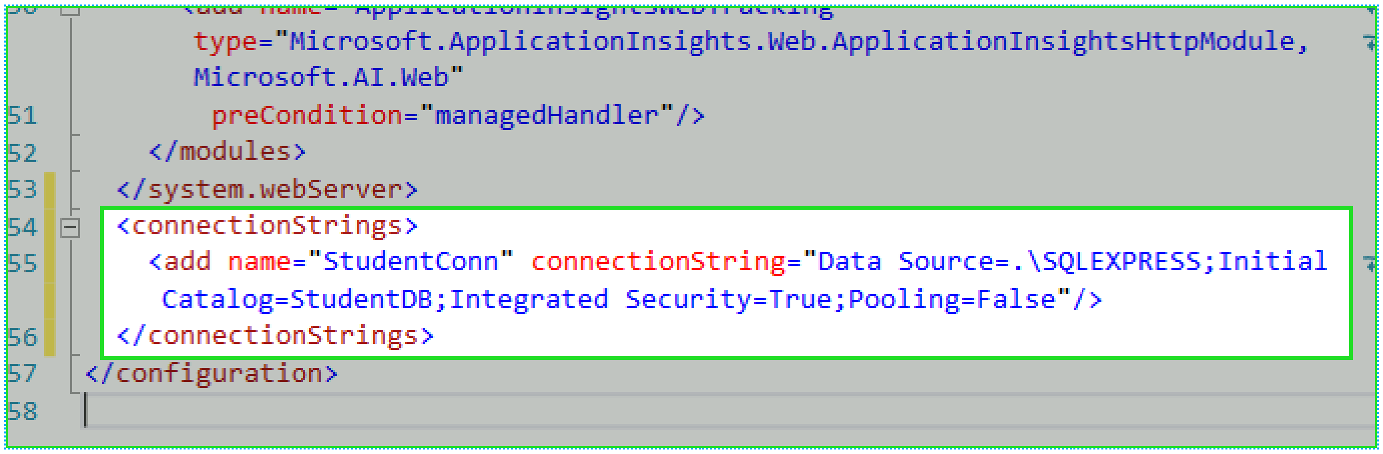
Properties

zm-2\sqlexpress.StudentDB.dbo Connection

(Name)	StudentDB
Case Sensitive	False
Connection String	Data Source=.\SQLEXPRESS;Initial Catalog=StudentDB;Integrated Security
Owner	ZM-2\Admin
Provider	.NET Framework Data Provider for SQL Server
State	Open
Type	Microsoft SQL Server
Version	11.00.2100

2. Open Root **Web.config** file and paste following code just before `</Configuratin>`.

```
<connectionStrings>
  <add name="StudentConn" connectionString="Data Source=.\SQLEXPRESS;Initial Catalog=Studen
</connectionStrings>
```



```
        type="Microsoft.ApplicationInsights.Web.ApplicationInsightsHttpModule,
        Microsoft.AI.Web"
         preCondition="managedHandler"/>
    </modules>
  </system.webServer>
  <connectionStrings>
    <add name="StudentConn" connectionString="Data Source=.\SQLEXPRESS;Initial
     Catalog=StudentDB;Integrated Security=True;Pooling=False"/>
  </connectionStrings>
</configuration>
```

6. Create StudentDBHandle.cs class for handling all the database operations.

Paste Following code in **StudentDBHandle.cs** class.

1. Right click on **Models** folder ➔ **Add** ➔ **Class**. Create new class `StudentDBHandle.cs`

2. Now paste following code in this file.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

namespace CRUDinMVC.Models
{
    public class StudentDBHandle
    {
        private SqlConnection con;
        private void connection()
        {
            string constring = ConfigurationManager.ConnectionStrings["studentconn"].ToStri
            con = new SqlConnection(constring);
        }

        // **************** ADD NEW STUDENT *********************
        public bool AddStudent(StudentModel smodel)
        {
            connection();
            SqlCommand cmd = new SqlCommand("AddNewStudent", con);
            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.AddWithValue("@Name", smodel.Name);
            cmd.Parameters.AddWithValue("@City", smodel.City);
            cmd.Parameters.AddWithValue("@Address", smodel.Address);

            con.Open();
            int i = cmd.ExecuteNonQuery();
            con.Close();

            if (i >= 1)
                return true;
            else
                return false;
        }

        // ********** VIEW STUDENT DETAILS ********************
        public List<StudentModel> GetStudent()
        {
            connection();
            List<StudentModel> studentlist = new List<StudentModel>();

            SqlCommand cmd = new SqlCommand("GetStudentDetails", con);
            cmd.CommandType = CommandType.StoredProcedure;
            SqlDataAdapter sd = new SqlDataAdapter(cmd);
            DataTable dt = new DataTable();

            con.Open();
            sd.Fill(dt);
            con.Close();

            foreach(DataRow dr in dt.Rows)
            {
                studentlist.Add(
                    new StudentModel
                    {
                        Id = Convert.ToInt32(dr["Id"]),
                        Name = Convert.ToString(dr["Name"]),
                        City = Convert.ToString(dr["City"]),
                        Address = Convert.ToString(dr["Address"])
                    });
            }
            return studentlist;
        }

        // ***************** UPDATE STUDENT DETAILS *********************
        public bool UpdateDetails(StudentModel smodel)
        {
            connection();
            SqlCommand cmd = new SqlCommand("UpdateStudentDetails", con);
            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.AddWithValue("@StdId", smodel.Id);
            cmd.Parameters.AddWithValue("@Name", smodel.Name);
            cmd.Parameters.AddWithValue("@City", smodel.City);
            cmd.Parameters.AddWithValue("@Address", smodel.Address);

            con.Open();
            int i = cmd.ExecuteNonQuery();
            con.Close();

            if (i >= 1)
                return true;
            else
                return false;
        }

        // ********************** DELETE STUDENT DETAILS *******************
        public bool DeleteStudent(int id)
        {
            connection();
            SqlCommand cmd = new SqlCommand("DeleteStudent", con);
            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.AddWithValue("@StdId", id);

            con.Open();
            int i = cmd.ExecuteNonQuery();
            con.Close();

            if (i >= 1)
                return true;
            else
                return false;
        }
    }
}
```

7. Add Action Method in StudentController

Open **StudentController** and add the following action methods. Here, I am going to add four action methods for following purpose.

a. `Index()` – Showing All Student Details

b. `Create()` – Adding New Student

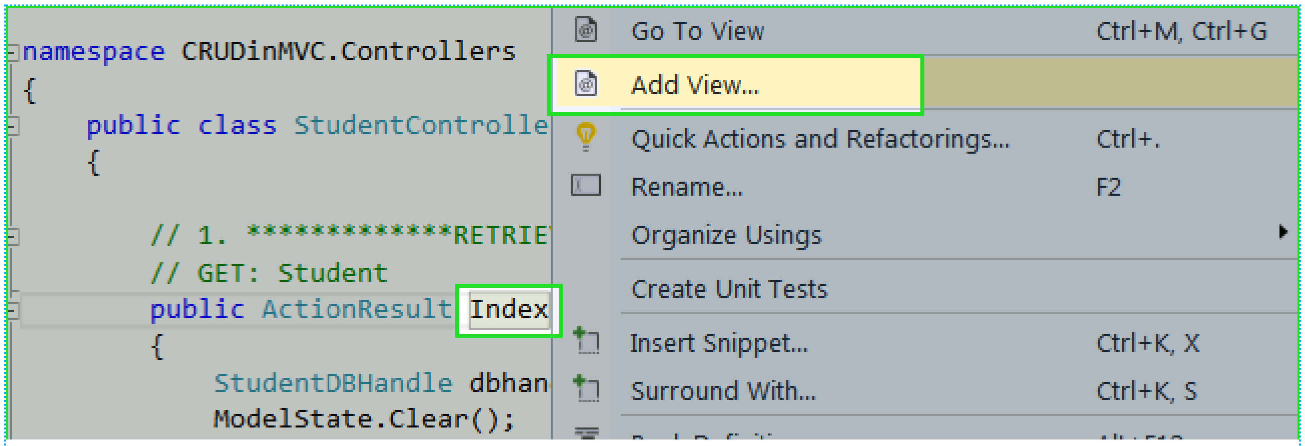
c. `Edit ()` – Edit or Update Student Details

d. `Delete ()` – Delete Student Details

```
using System.Web.Mvc;
using CRUDinMVC.Models;

namespace CRUDinMVC.Controllers
{
    public class StudentController : Controller
    {

        // 1. *************RETRIEVE ALL STUDENT DETAILS ******************
        // GET: Student
        public ActionResult Index()
        {
            StudentDBHandle dbhandle = new StudentDBHandle();
            ModelState.Clear();
            return View(dbhandle.GetStudent());
        }

        // 2. *************ADD NEW STUDENT ******************
        // GET: Student/Create
        public ActionResult Create()
        {
            return View();
        }

        // POST: Student/Create
        [HttpPost]
        public ActionResult Create(StudentModel smodel)
        {
            try
            {
                if (ModelState.IsValid)
                {
                    StudentDBHandle sdb = new StudentDBHandle();
                    if (sdb.AddStudent(smodel))
                    {
                        ViewBag.Message = "Student Details Added Successfully";
                        ModelState.Clear();
                    }
                }
                return View();
            }
            catch
            {
                return View();
            }
        }

        // 3. ************* EDIT STUDENT DETAILS ******************
        // GET: Student/Edit/5
        public ActionResult Edit(int id)
        {
            StudentDBHandle sdb = new StudentDBHandle();
            return View(sdb.GetStudent().Find(smodel => smodel.Id == id));
        }

        // POST: Student/Edit/5
        [HttpPost]
        public ActionResult Edit(int id, StudentModel smodel)
        {
            try
            {
                StudentDBHandle sdb = new StudentDBHandle();
                sdb.UpdateDetails(smodel);
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        // 4. ************* DELETE STUDENT DETAILS ******************
        // GET: Student/Delete/5
        public ActionResult Delete(int id)
        {
            try
            {
                StudentDBHandle sdb = new StudentDBHandle();
                if (sdb.DeleteStudent(id))
                {
                    ViewBag.AlertMsg = "Student Deleted Successfully";
                }
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }
    }
}
```

8. Create Partial View from Action Method

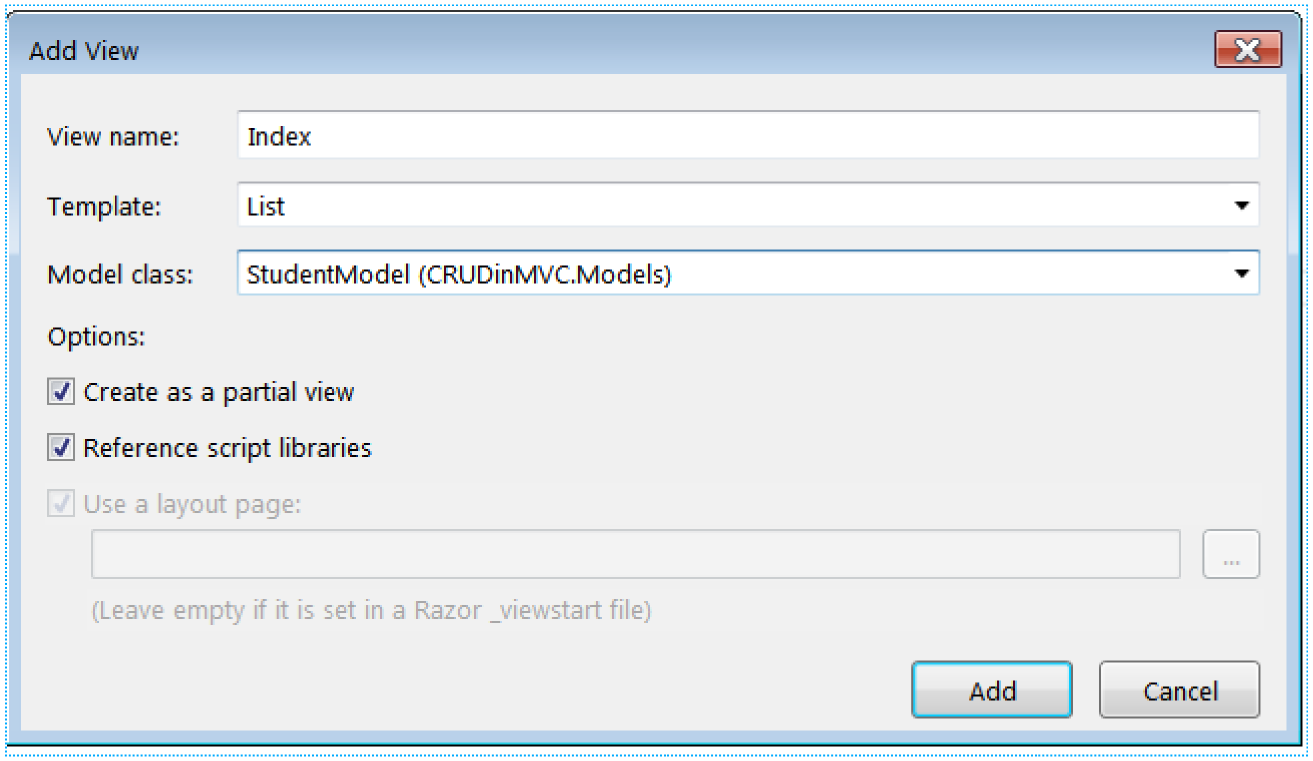
1. Right Click on **Index()** Action Method and Select **Add View**.


Go To View Ctrl+M, Ctrl+G
Add View...
Quick Actions and Refactorings... Ctrl+.
Rename... F2
Organize Usings
Create Unit Tests
Insert Snippet... Ctrl+K, X
Surround With... Ctrl+K, S

```
namespace CRUDinMVC.Controllers
{
    public class StudentControlle
    {

        // 1. *************RETRIE
        // GET: Student
        public ActionResult Index
        {
            StudentDBHandle dbhan
            ModelState.Clear();
```

2. Fill following details in Add View Dialog box.

- a. **View Name:** Index
- b. **Template:** List
- c. **Model Class:** StudentModel (CRUDinMVC.Models)
- d. **Check** both CheckBoxes
- e. Click **Add** button to **Add View**.



Add View

View name: Index

Template: List

Model class: StudentModel (CRUDinMVC.Models)

Options:

- [✓] Create as a partial view
- [✓] Reference script libraries
- [✓] Use a layout page:

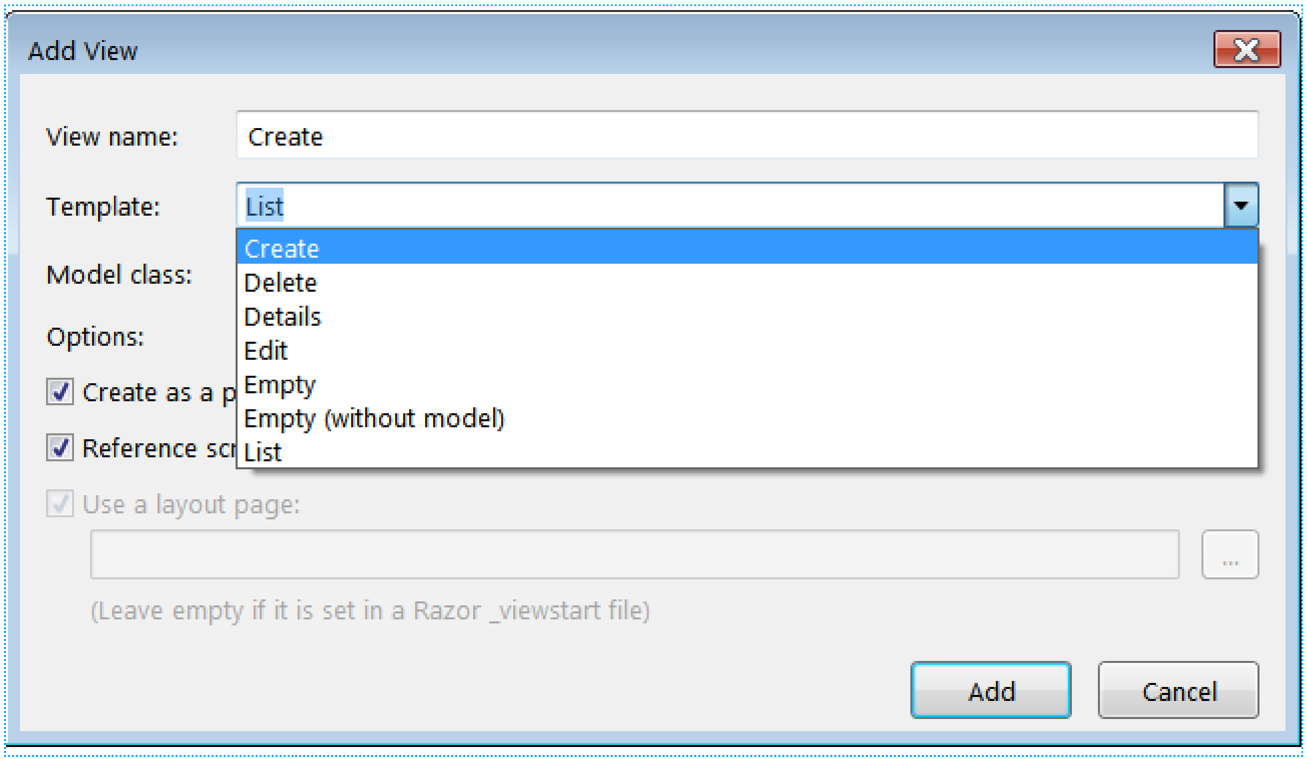
(Leave empty if it is set in a Razor _viewstart file)

3. View: Index()

Make sure to change the following highlighted line as example.

```
@model IEnumerable<CRUDinMVC.Models.StudentModel>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Name)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.City)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Address)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Name)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.City)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Address)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.Id })
            @Html.ActionLink("Delete", "Delete", new { id=item.Id }, new { onclick = "return
        </td>
    </tr>
}

</table>
```

4. Follow same procedure for `Create()` and `Edit()` Method. Choose **Create** Template for `Create()` Method and **Edit** Template for `Edit()` Method.



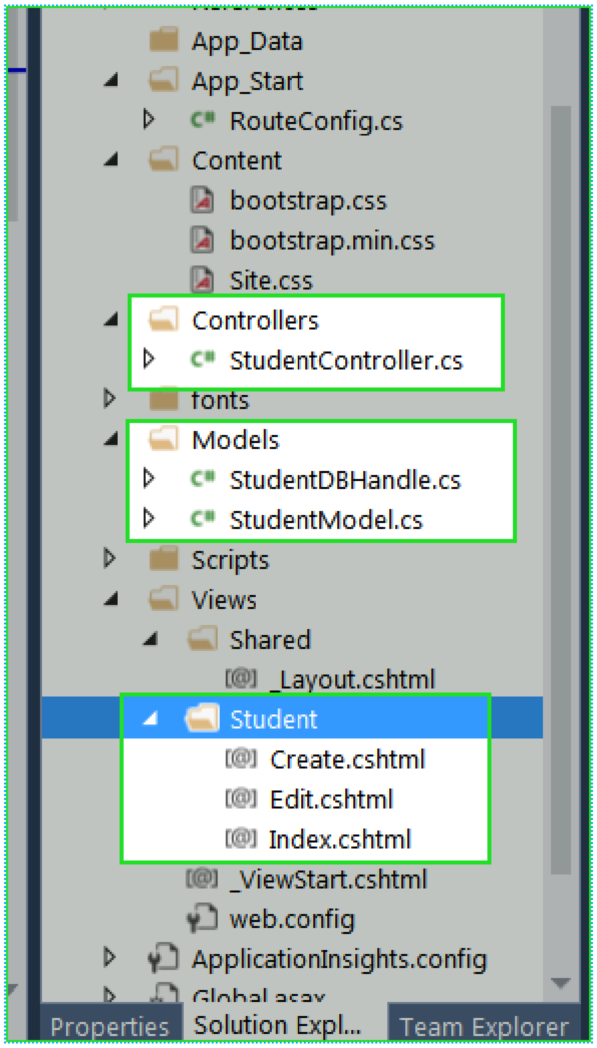
View: Create()

```
@model CRUDinMVC.Models.StudentModel


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>StudentModel</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label
            <div class="col-md-10">
                @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "fc
                @Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-dange
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.City, htmlAttributes: new { @class = "control-label
            <div class="col-md-10">
                @Html.EditorFor(model => model.City, new { htmlAttributes = new { @class = "fc
                @Html.ValidationMessageFor(model => model.City, "", new { @class = "text-dange
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Address, htmlAttributes: new { @class = "control-lab
            <div class="col-md-10">
                @Html.EditorFor(model => model.Address, new { htmlAttributes = new { @class =
                @Html.ValidationMessageFor(model => model.Address, "", new { @class = "text-da
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
        <h3>@ViewBag.Message</h3>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

<script src="~/Scripts/jquery-1.10.2.min.js"></script>
<script src="~/Scripts/jquery.validate.min.js"></script>
<script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>
```

View: Edit()

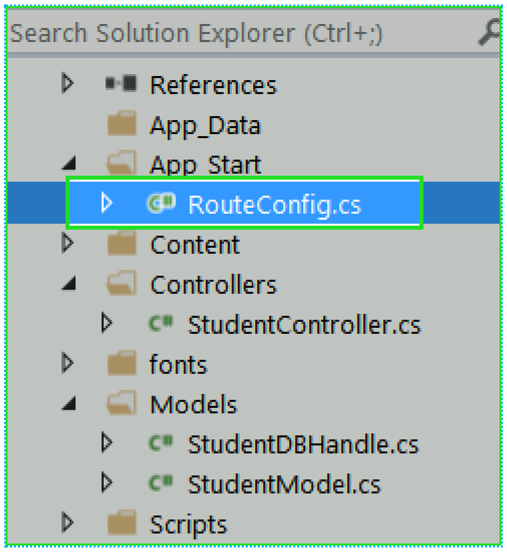
```
@model CRUDinMVC.Models.StudentModel


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>StudentModel</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.Id)

        <div class="form-group">
            @Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label
            <div class="col-md-10">
                @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "fo
                @Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-dange
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.City, htmlAttributes: new { @class = "control-label
            <div class="col-md-10">
                @Html.EditorFor(model => model.City, new { htmlAttributes = new { @class = "fo
                @Html.ValidationMessageFor(model => model.City, "", new { @class = "text-dange
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Address, htmlAttributes: new { @class = "control-lab
            <div class="col-md-10">
                @Html.EditorFor(model => model.Address, new { htmlAttributes = new { @class =
                @Html.ValidationMessageFor(model => model.Address, "", new { @class = "text-da
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

<script src="~/Scripts/jquery-1.10.2.min.js"></script>
<script src="~/Scripts/jquery.validate.min.js"></script>
<script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>
```

5. Your **Solution Explorer** should look like this.

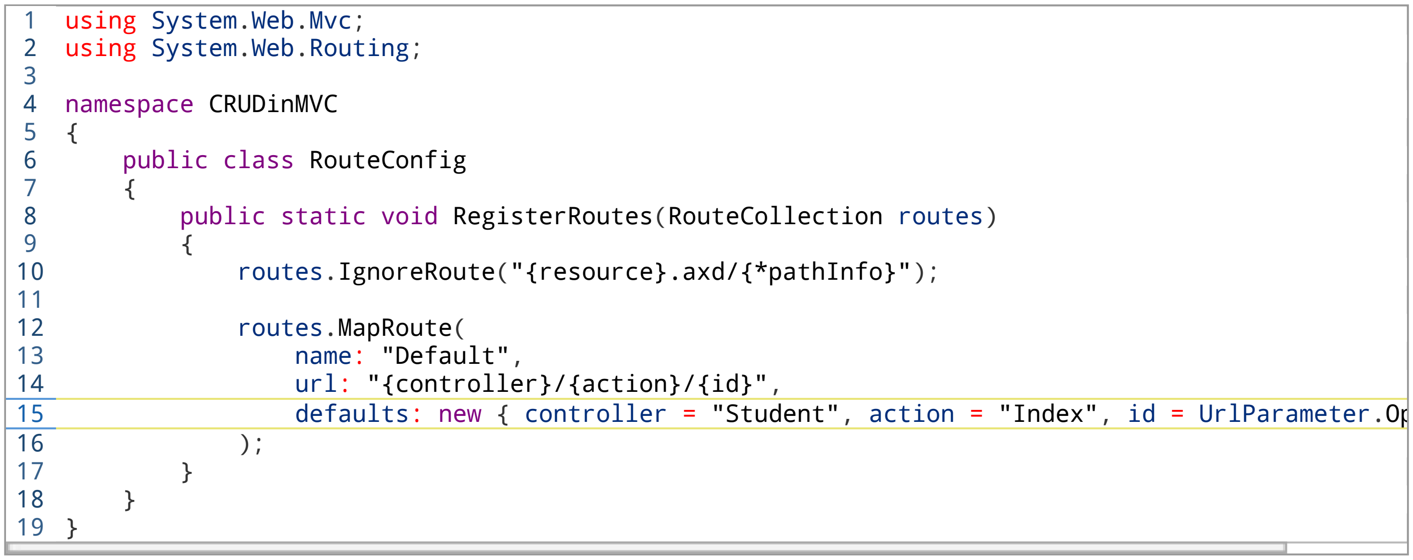


9. Configure RouteConfig.cs

1. **Open** `RouteConfig.cs` from **Solution Explorer**.



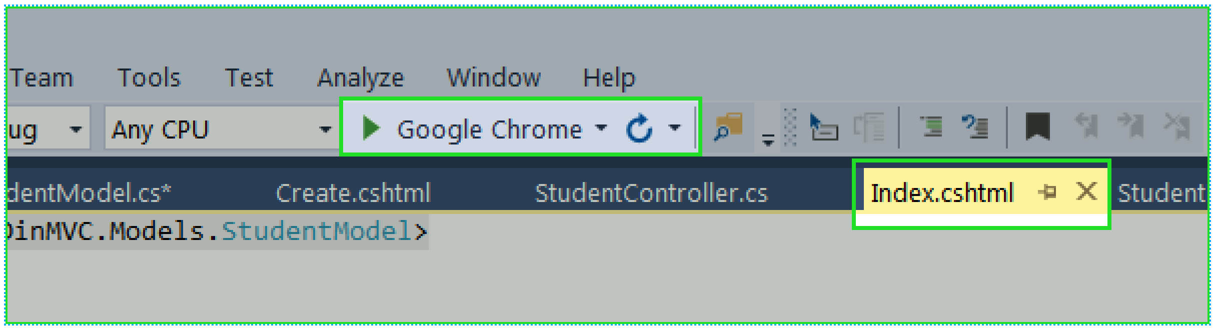
2. Change the highlighted code in your `RouteConfig.cs` file.



```
using System.Web.Mvc;
using System.Web.Routing;

namespace CRUDinMVC
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Student", action = "Index", id = UrlParameter.Op
            );
        }
    }
}
```

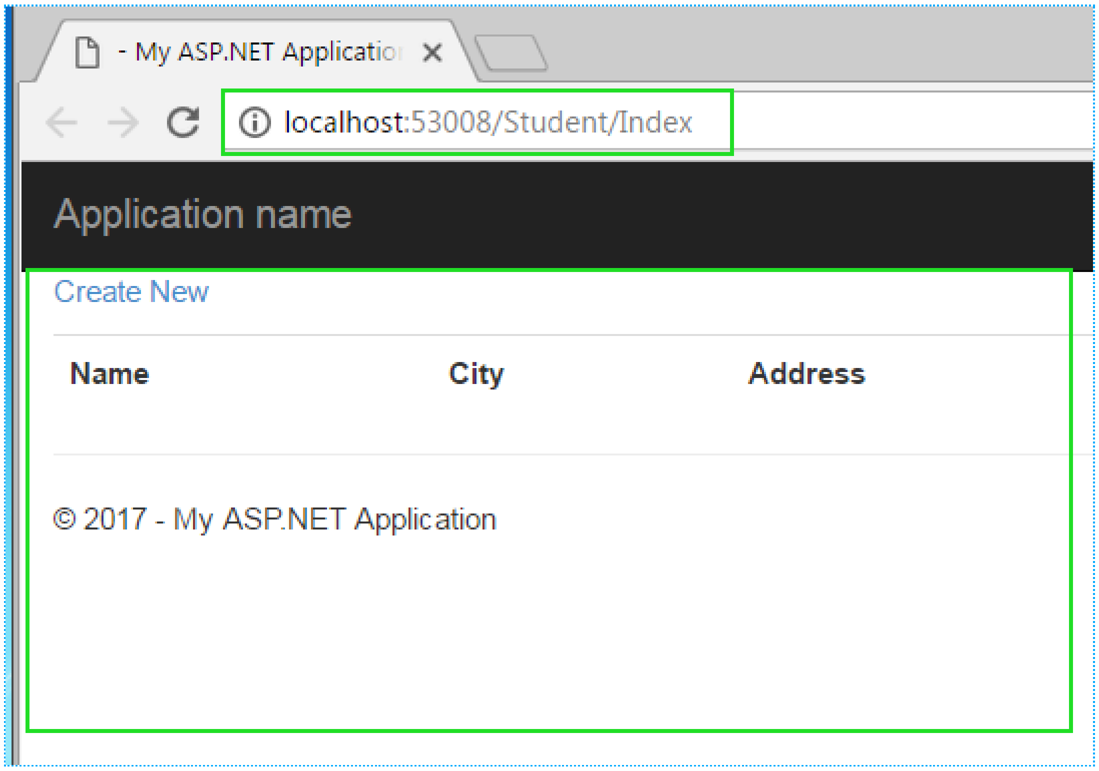
10. Run Your Program

1. Open **Index.cshtml** View Page then press F5 to run your program. Selecting **Index.cshtml** will start your project with this page.



```
)inMVC.Models.StudentModel>
```

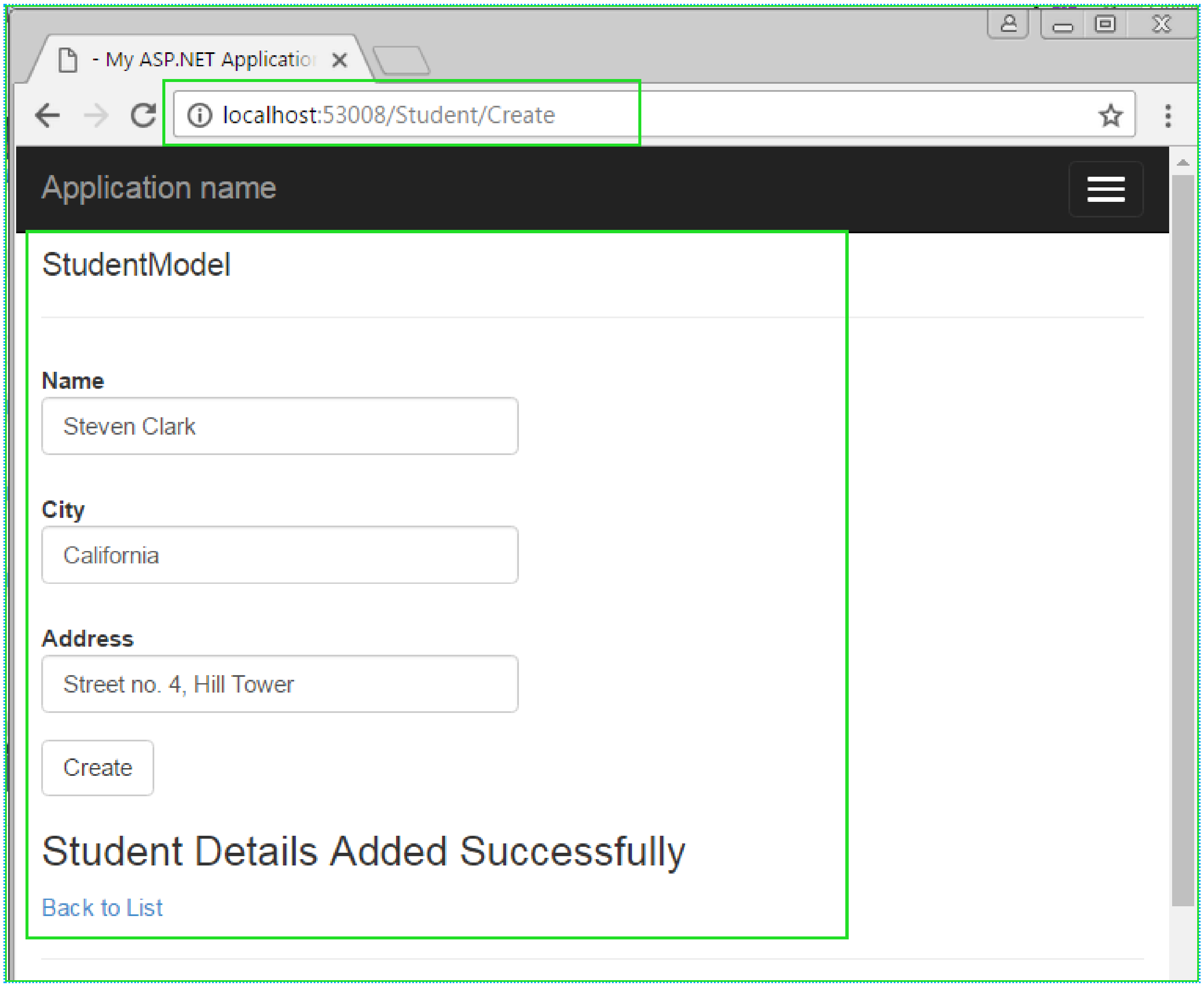
2. Index Page



Name	City	Address

© 2017 - My ASP.NET Application

3. Create Page.


- My ASP.NET Applicatio

localhost:53008/Student/Create

Application name

StudentModel

Name

Steven Clark

City

California

Address

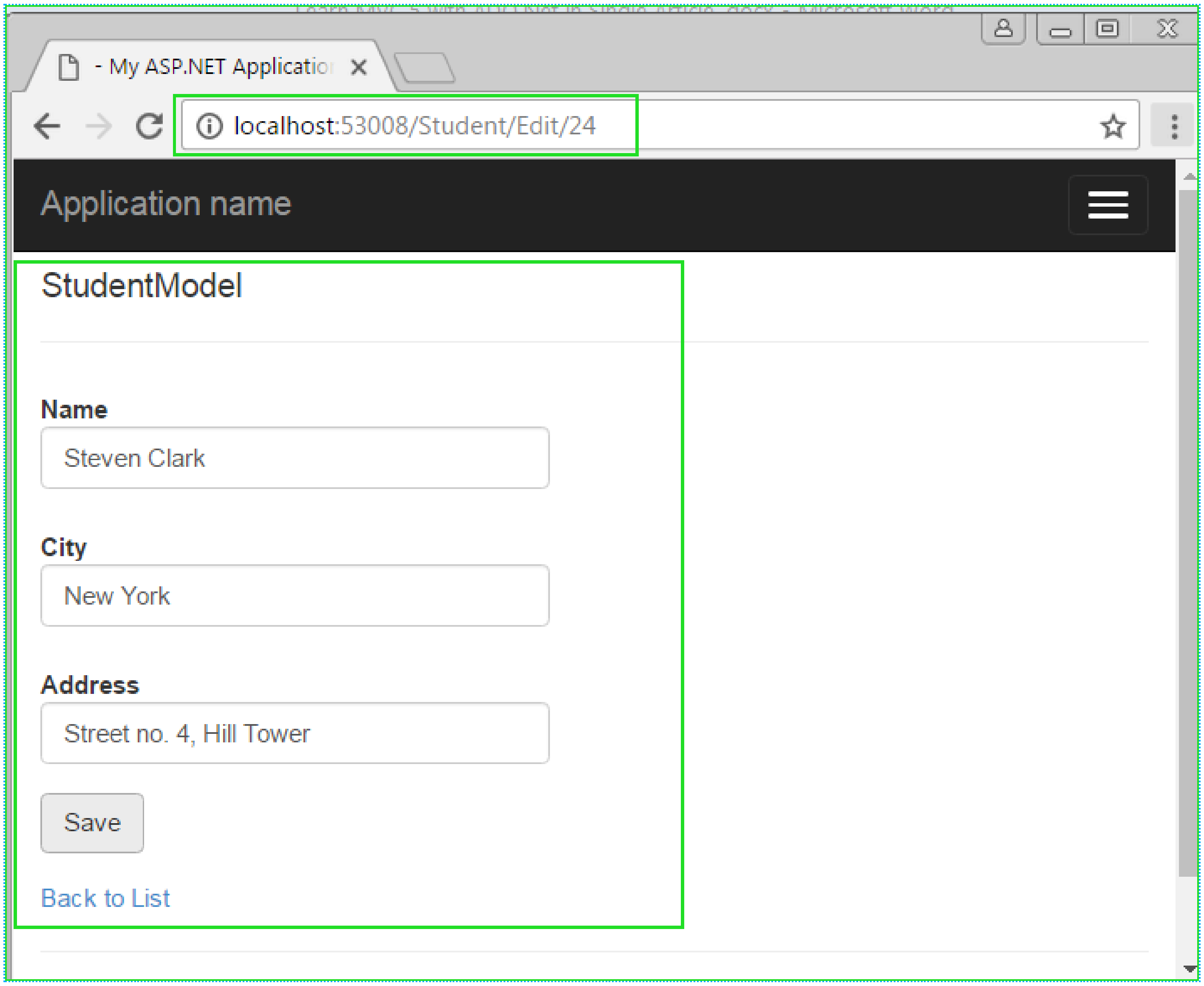
Street no. 4, Hill Tower

Create

Student Details Added Successfully

Back to List

4. Edit Page



- My ASP.NET Applicatio

localhost:53008/Student/Edit/24

Application name

StudentModel

Name

Steven Clark

City

New York

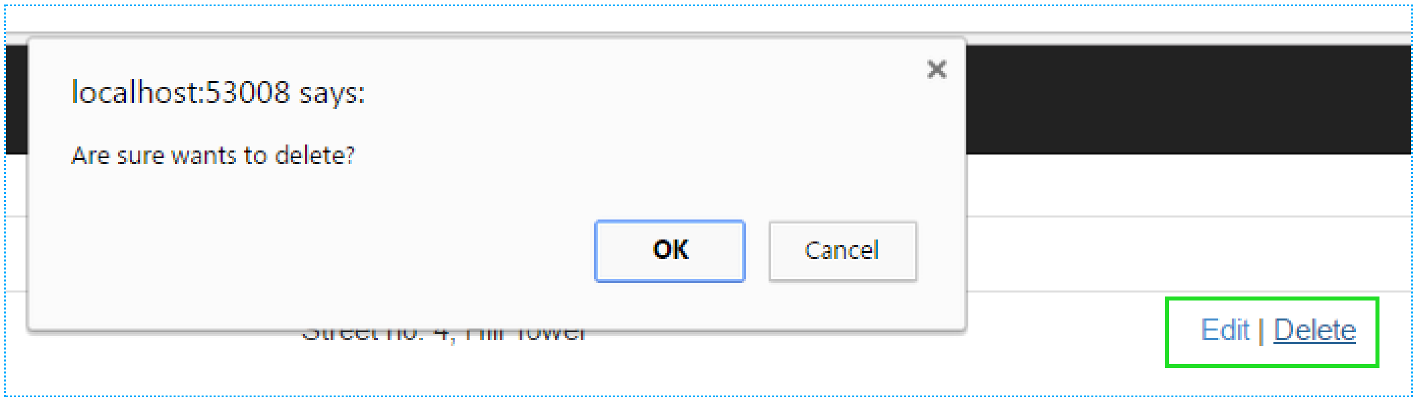
Address

Street no. 4, Hill Tower

Save

Back to List

5. Delete Page



localhost:53008 says:

Are sure wants to delete?

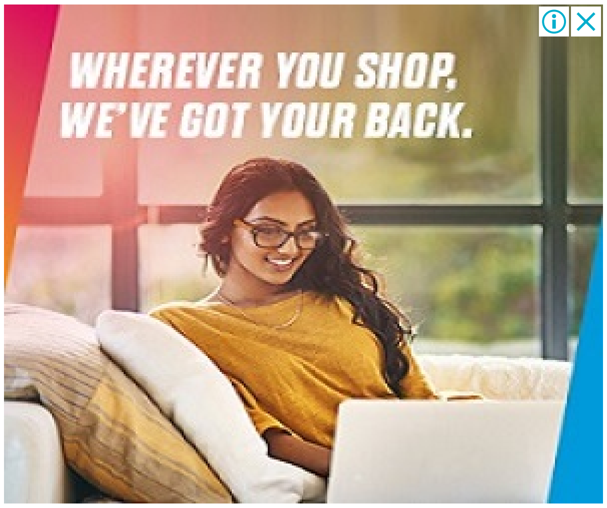
OK Cancel

Street no. 4, Hill Tower

Edit | Delete

SUMMARY:

In this tutorial, I have tried to explain full and complete **create, update and delete program in MVC 5 without entity framework**. I have inserted great number of screenshot so that you can easily finish this project on your system. Believe me, once after completing this chapter your understanding of MVC, will be increased at great level.



11 THOUGHTS ON "INSERT UPDATE DELETE IN ASP.NET MVC 5 WITHOUT ENTITY FRAMEWORK"

Sundar

MAY 8, 2017 AT 6:49 AM

Delete operation is not performed. Please help me

Nithin

MAY 10, 2017 AT 6:44 PM

Thanks a lot…
This is the best article and I understood a lot about mvc, stored procedure seeing this.
Please create more articles like this which are helpful for beginners like me.

Atiol Raphael

MAY 10, 2017 AT 10:36 PM

Awesome tutorial, I followed it up to the end, it helped me understand what goes on behind the scenes.
Am also currently developing a webapp with oracle database, Can I follow the same procedure for when storing data in an oracle DB or it differs?
Thanks in advance.

Naveen Kumar Rohilla

MAY 13, 2017 AT 12:44 PM

Thanks for this post. After long time i saw this type of article.

Hashan Meegahawatte

JUNE 15, 2017 AT 6:19 AM

good one for beginners

Hans

JUNE 16, 2017 AT 12:58 PM

Is there a way to have two different table in one view?

SARAVANAN A

JULY 4, 2017 AT 10:45 AM

That is very nice to understand step by step.thank you…..

SARAVANAN A

JULY 4, 2017 AT 11:44 AM

Thank you very much for posting this MVC simple project. Before i'am not knowing what is mvc and idea but now i'am start with my project.

Chetan

JULY 6, 2017 AT 12:01 PM

Nice Tutorial, I followed all the Steps. Its working 100% fine. but the view Layout (Bootstrap) is not reflecting in my Application. need Help.

Esa

AUGUST 8, 2017 AT 8:49 AM

It is easy to understand MVC. We are expect more examples like this. thank you

natuca

AUGUST 28, 2017 AT 8:17 AM

Very Nice Article. Please Keep Up z Good Work