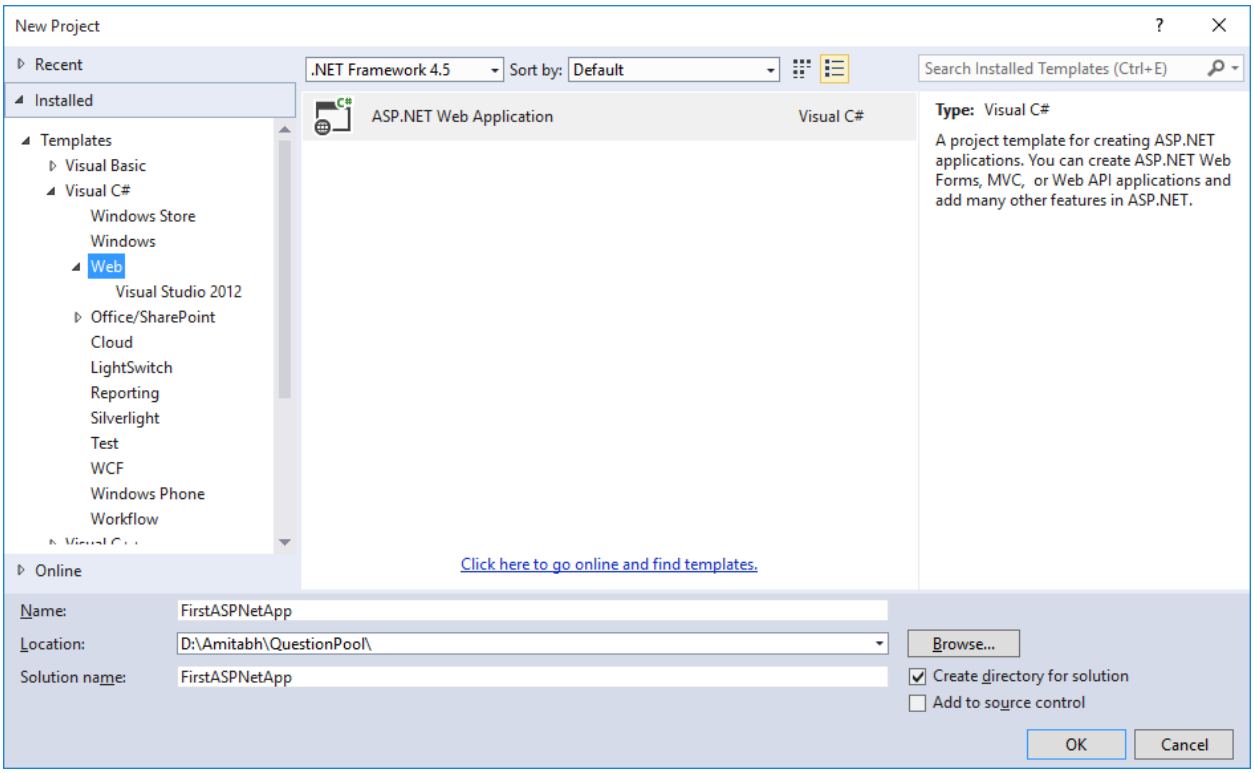
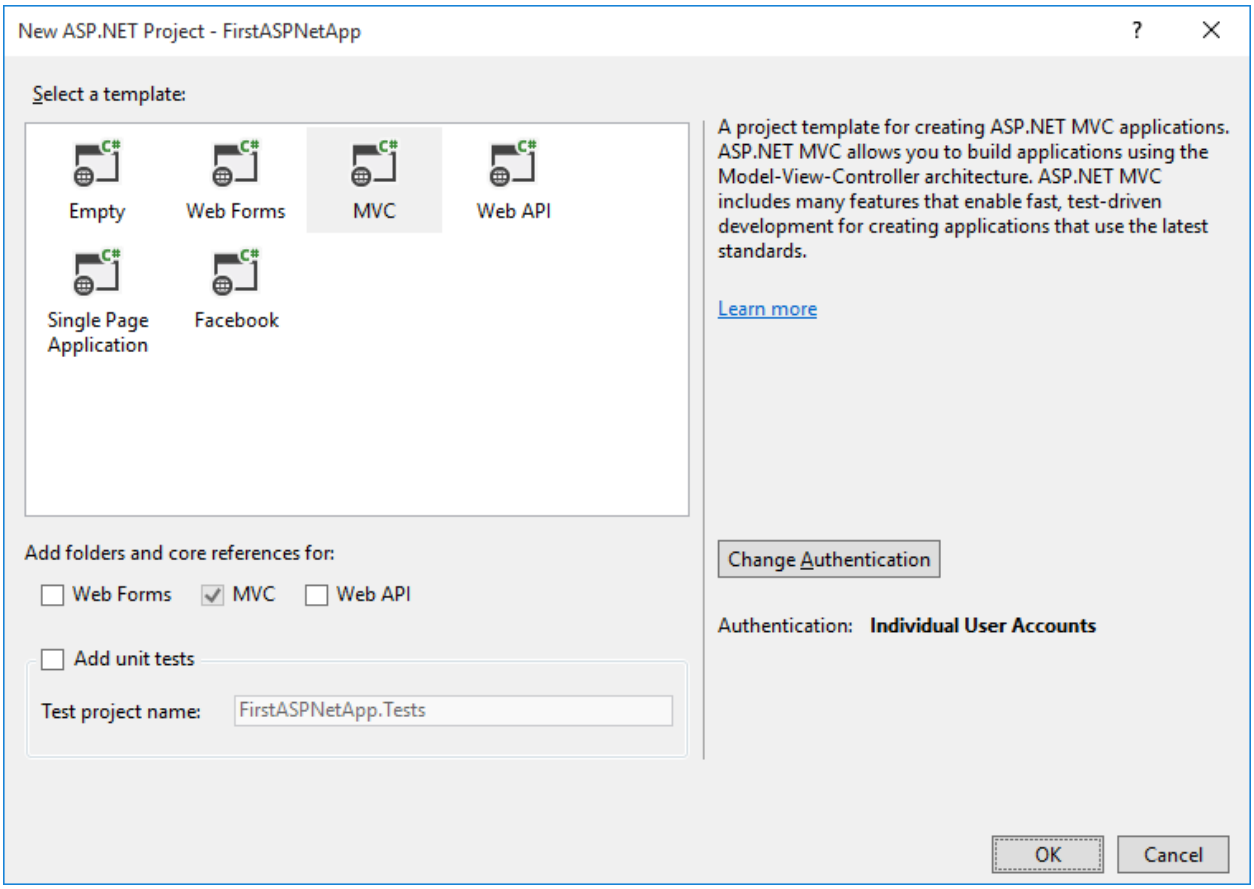
Exercise 1: Create a New ASP.Net MVC Project.

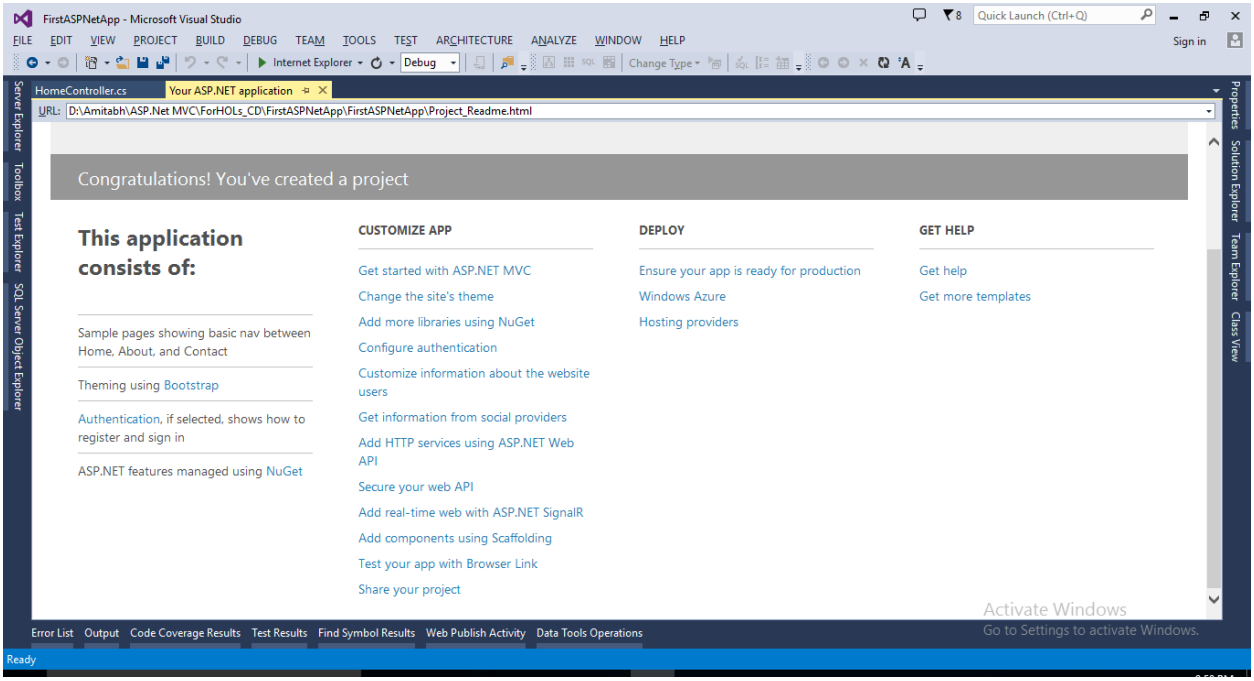
Open Visual Studio 2013. Go to **File** menu → **New** →**Project**, then select Visual C# from the left, then **Web** and then select **ASP.Net Web Application**. Name your project "FirstASPNetApp" and then click **OK**.



In the new **ASP.Net Project** dialog, click **MVC** and then click **OK**.



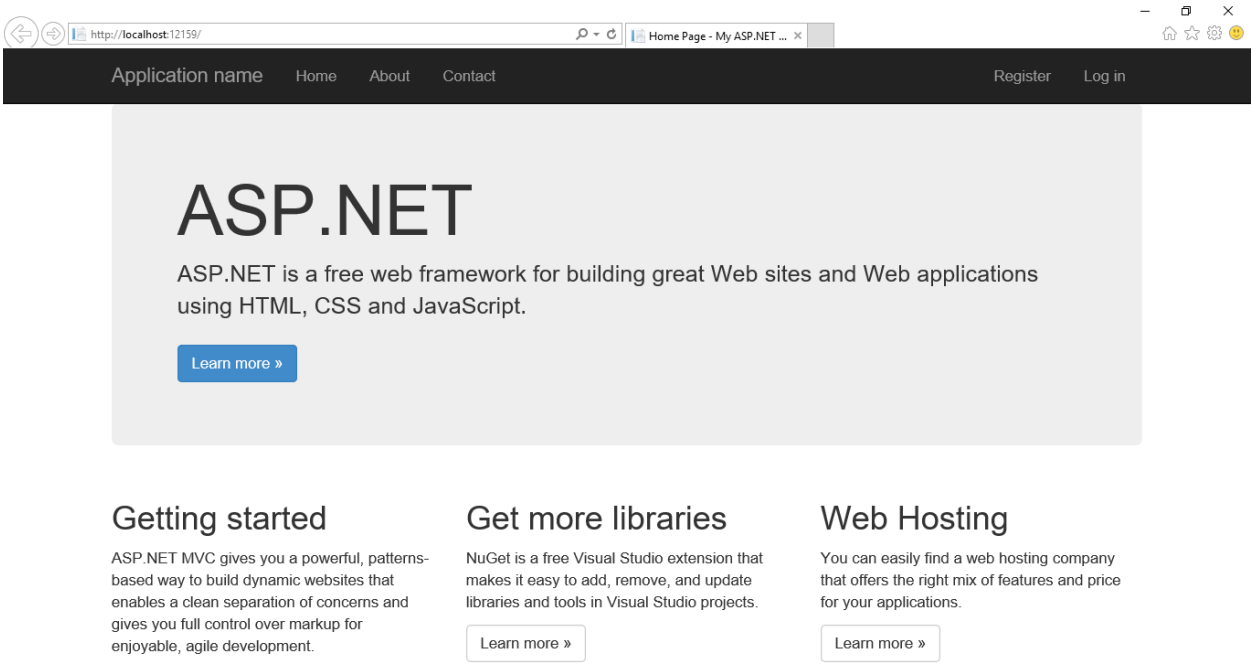
Visual Studio uses a default template for ASP.Net MVC project you just created. So you have a working application without doing anything.



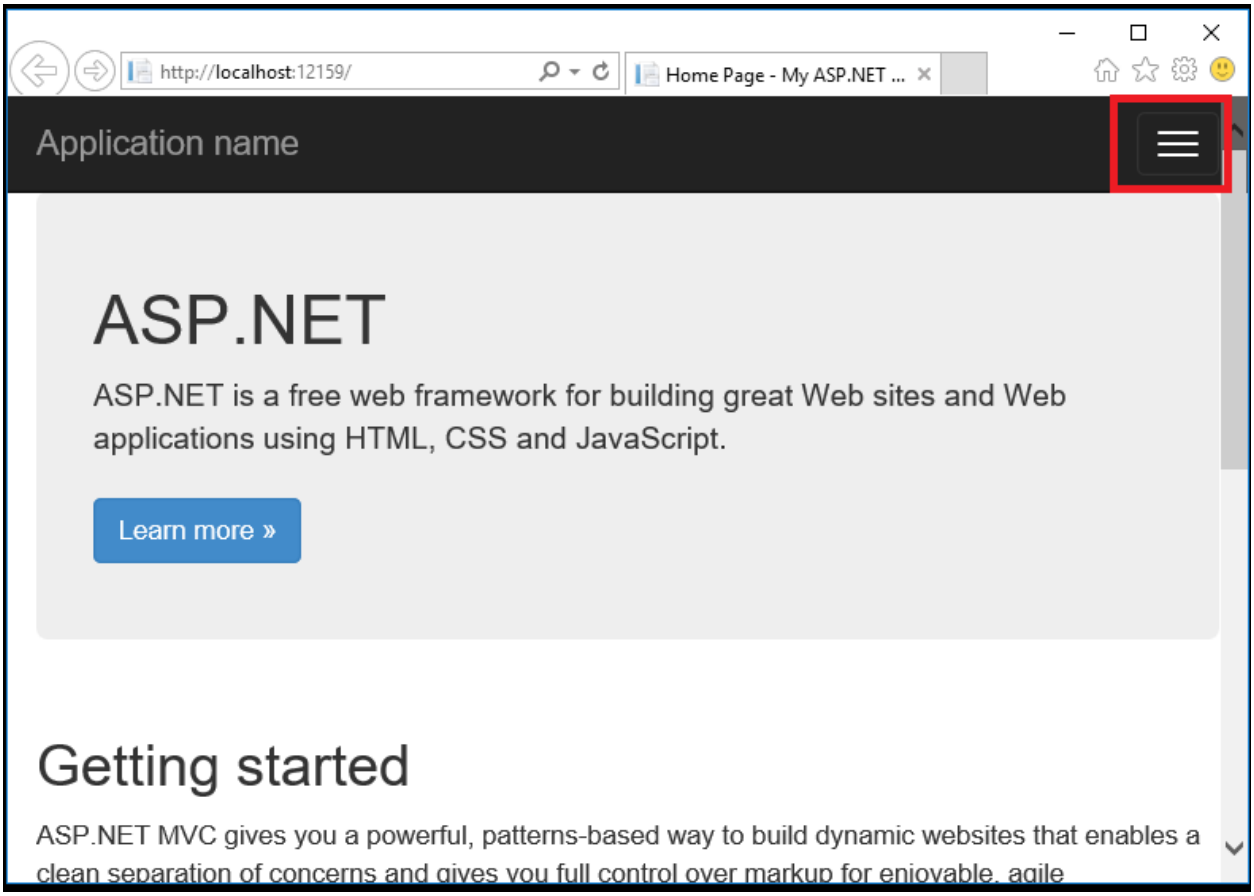
Click F5 to run the application. F5 causes Visual Studio to start IIS Express and run you web app. Visual Studio then launches a browser and opens applications home page.

The default template gives you Home, Contact and About pages. Depending upon the size of your browser window the links for these pages will be displayed.

Following figure displays Home page in full screen.



Following figure displays Home page resized down.



Exercise 2 - Part I: Creating Database and Tables

Before adding **ADO.Net Entity Data Model** in the project, you need to create database named *EmployeeMgmt* in your **MS SQL Server** with three **Tables Employee, Designation and Department**.

Note: Server name and Authentication will change according to the configuration of your system.

Following are the scripts for creating the tables:

Designation Table:

```
CREATE TABLE [dbo].[Designation](
     [ID] [int] NOT NULL,
     [Name] [varchar](50) NOT NULL,
 CONSTRAINT [PK_Designation] PRIMARY KEY CLUSTERED
(
     [ID] ASC
)
)
```

Department Table:

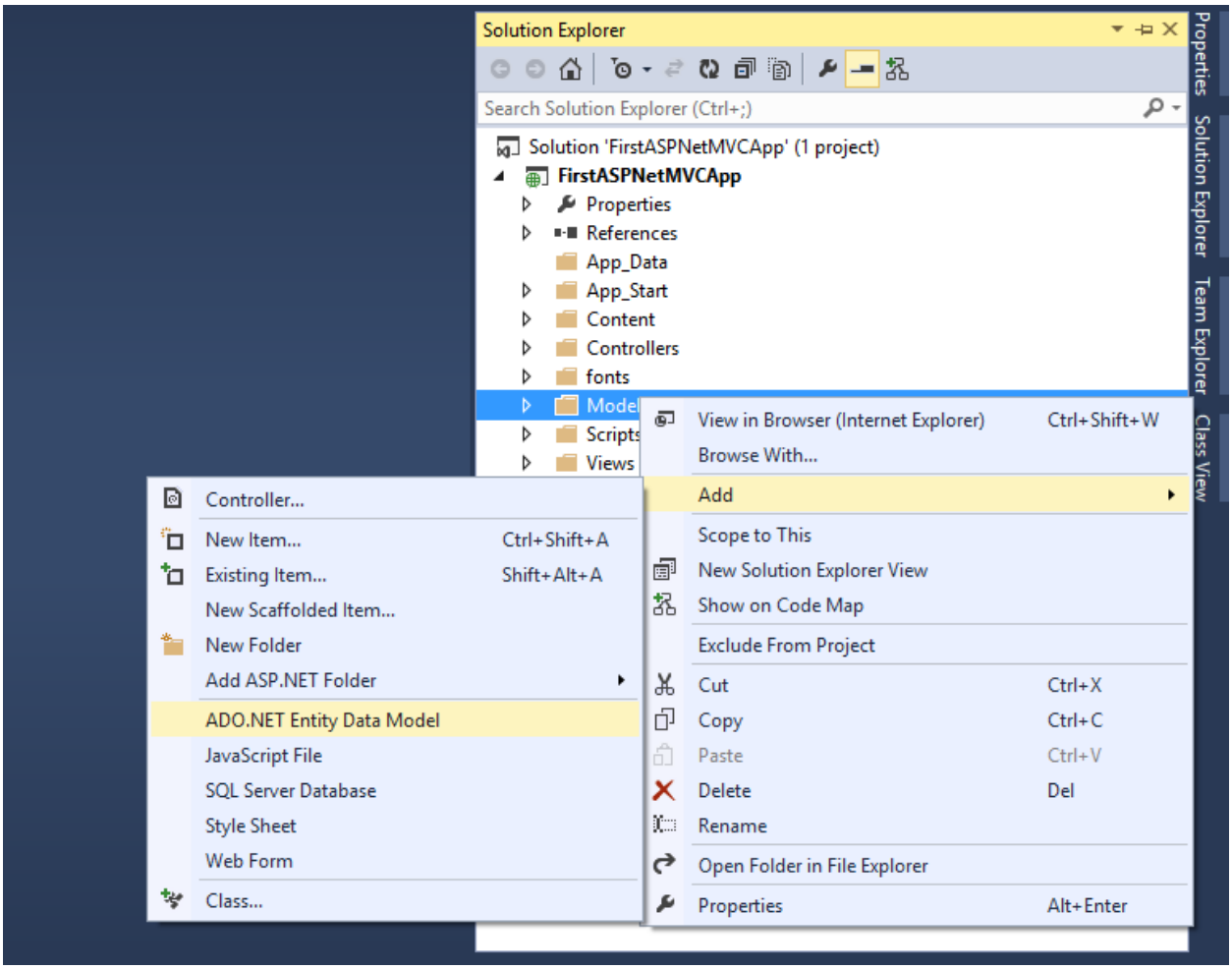
```
CREATE TABLE [dbo].[Department](
     [ID] [int] NOT NULL,
     [Name] [varchar](50) NOT NULL,
     [Head] [varchar](50) NOT NULL,
     [Location] [varchar](50) NOT NULL,
 CONSTRAINT [PK_Department] PRIMARY KEY CLUSTERED
(
     [ID] ASC
)
)
```

Employee Table:

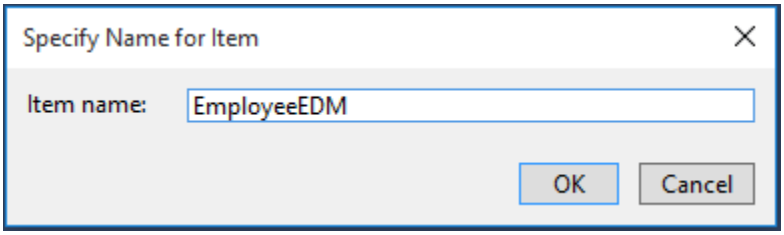
```
CREATE TABLE [dbo].[Employee](
     [ID] [int] NOT NULL,
     [Name] [varchar](50) NOT NULL,
     [Designation] [int] NOT NULL,
     [Department] [int] NOT NULL,
 CONSTRAINT [PK_Employee] PRIMARY KEY CLUSTERED
(
     [ID] ASC
)
)
GO

ALTER TABLE [dbo].[Employee]  WITH CHECK ADD  CONSTRAINT
[FK_Employee_Department] FOREIGN KEY([Department])
REFERENCES [dbo].[Department] ([ID])
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[Employee] CHECK CONSTRAINT
[FK_Employee_Department]
GO

ALTER TABLE [dbo].[Employee]  WITH CHECK ADD  CONSTRAINT
[FK_Employee_Designation] FOREIGN KEY([Designation])
REFERENCES [dbo].[Designation] ([ID])
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[Employee] CHECK CONSTRAINT
[FK_Employee_Designation]
GO
```

Exercise 2 – Part II: Adding Model (Using Entity Framework Database First Approach)

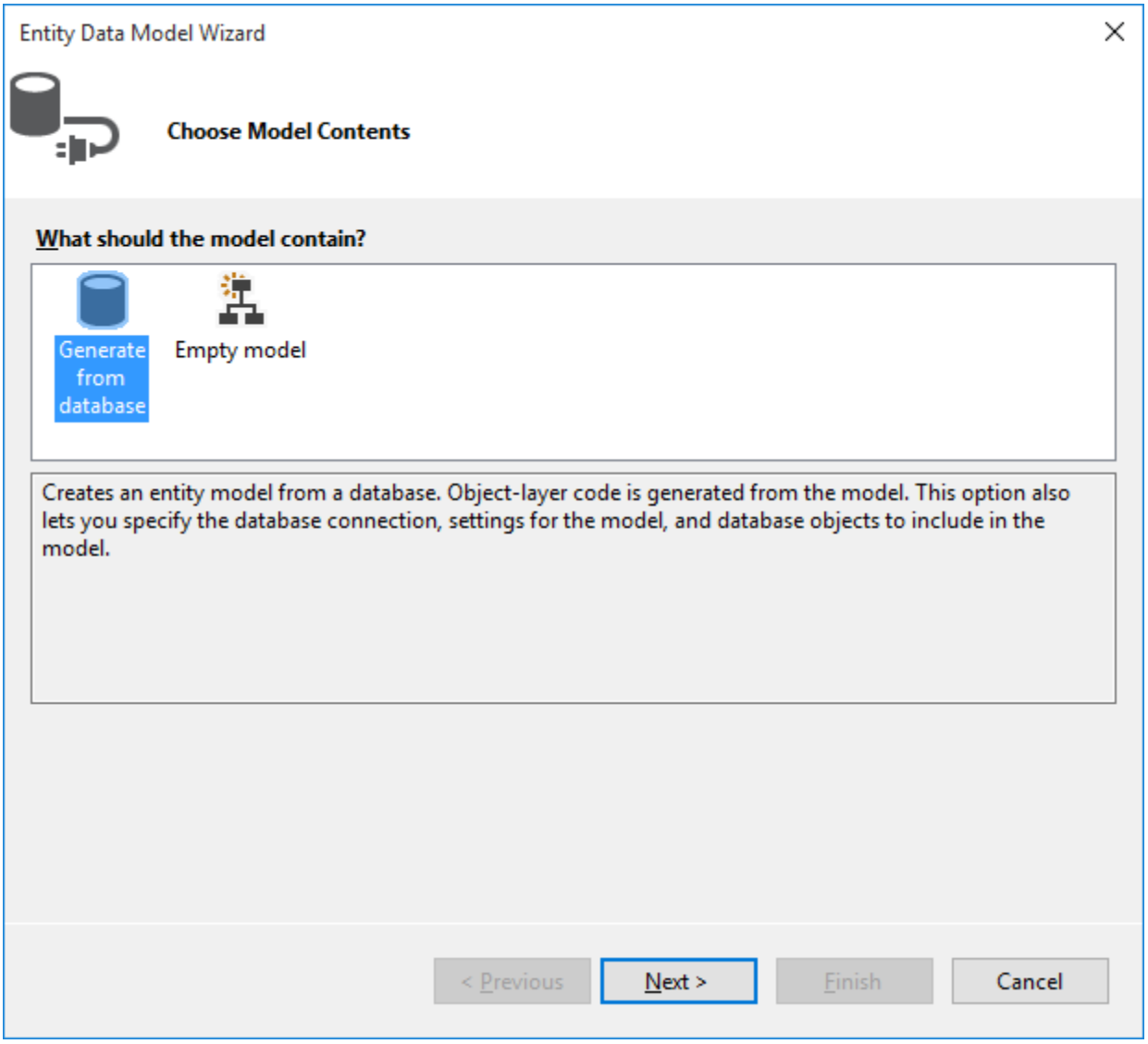
In **Solution Explorer**, right-click the *Model* folder and then click **Add**, then **ADO.Net Entity Data Model**.



Specify the name as below and click on OK button:



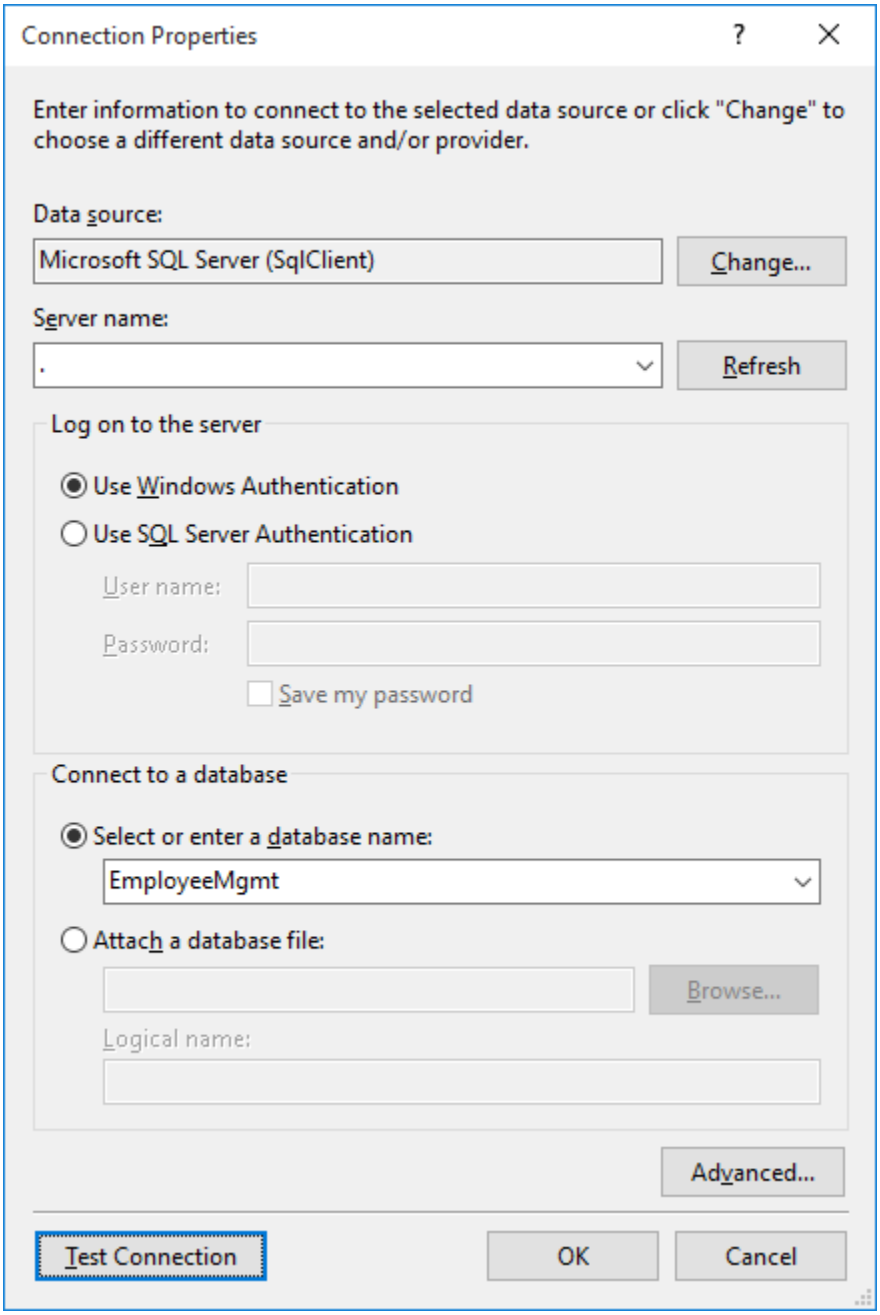
From the **Entity Data Model Wizard-Choose Model Contents** dialog, select *Generate from Database* option.



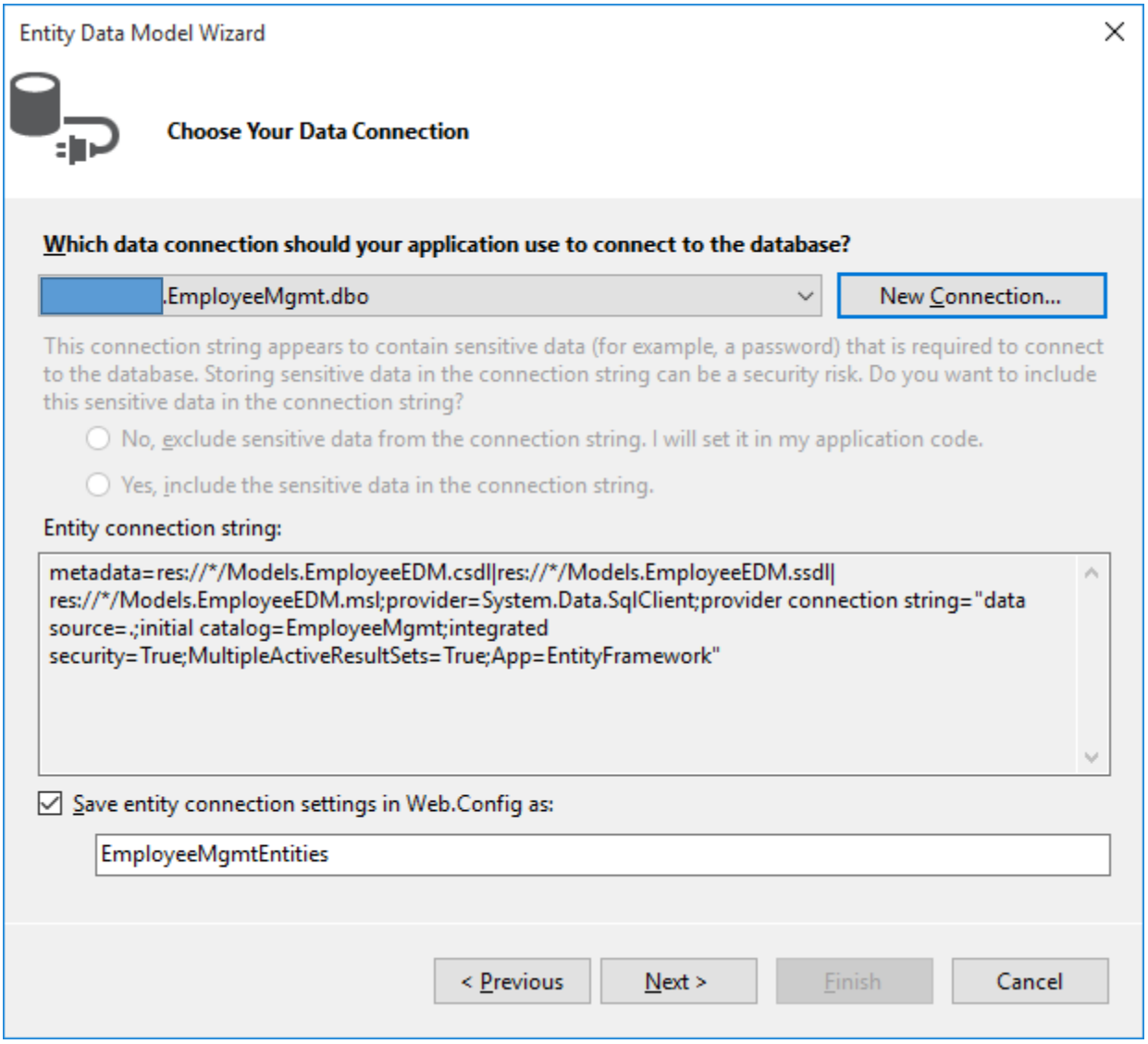
Click on Next button.

From the Entity Data **Model Wizard-Choose Your Data Connection** dialog, click on *New Connection*. Specify the properties as displayed below from the **Connection Properties** dialog.

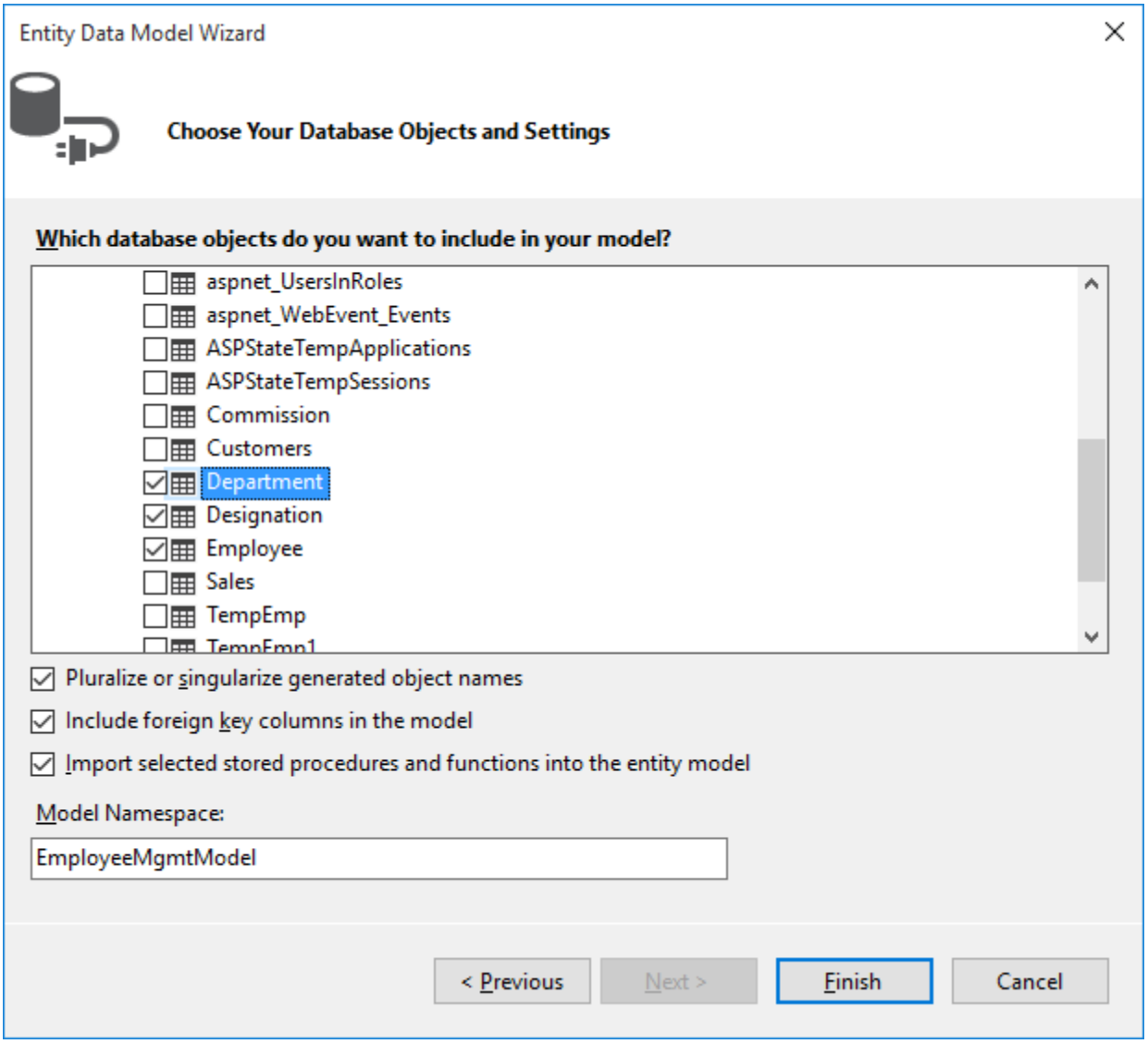
Note: Server name and Authentication will change according to the configuration of your system.



Click on **OK**. From the **Choose Your Data Connection** dialog Select *Save Entity connection settings in Web.config* as: option and then click **Next**.



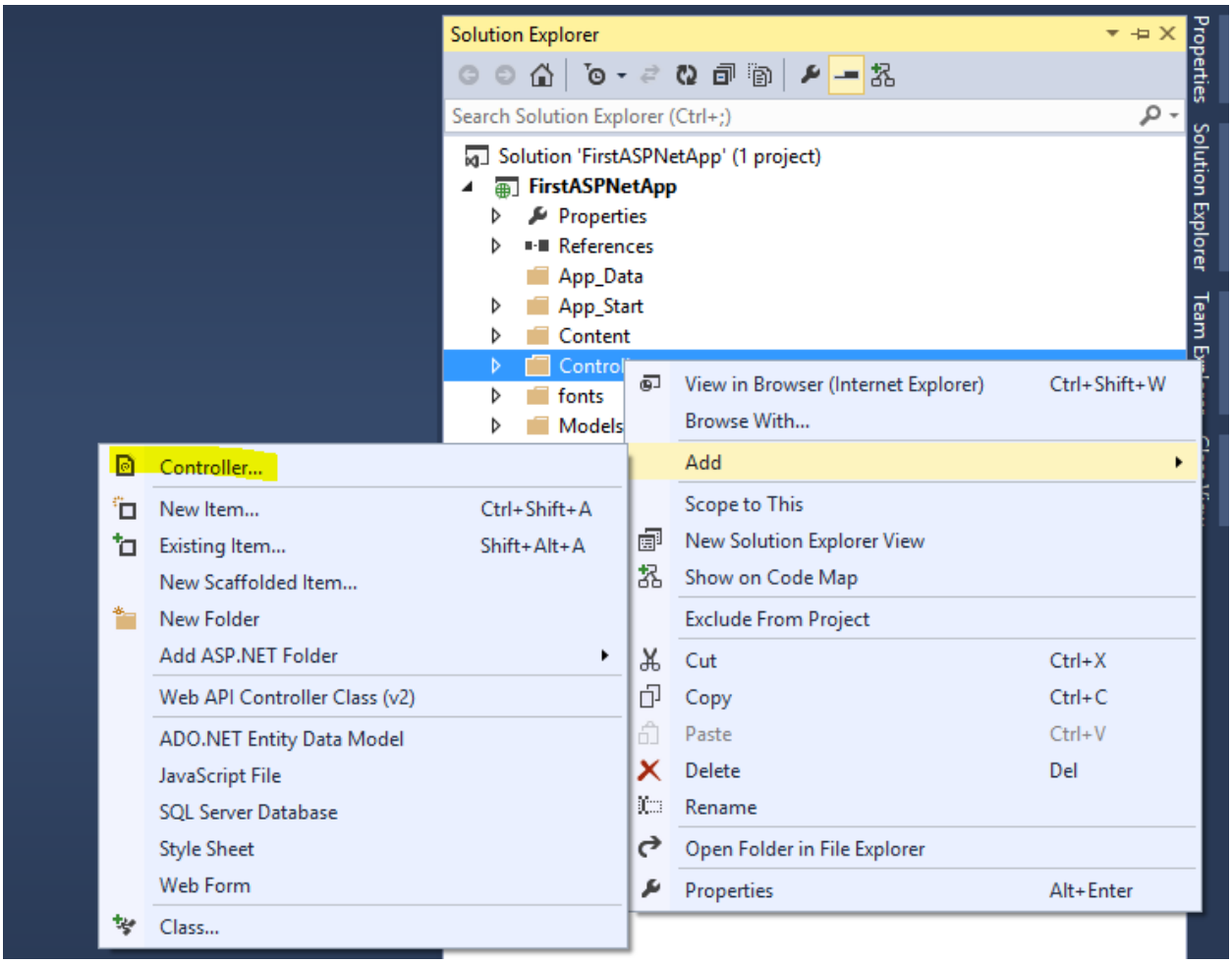
From the Choose you database Objects and Settings screen, select three tables Employee, Designation and Department. And click on Finish.



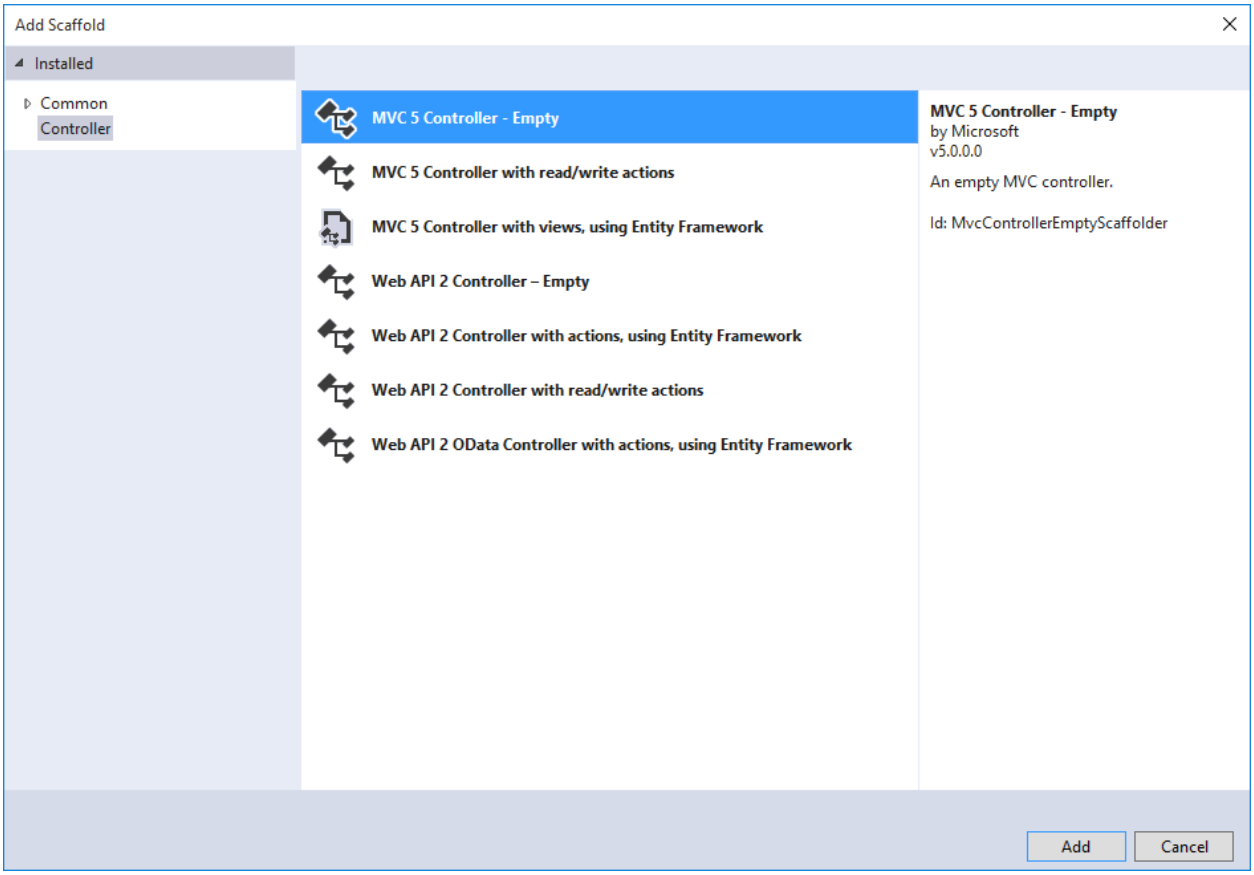
Finally click on Finish and build your application.

Exercise 3: Adding a Controller

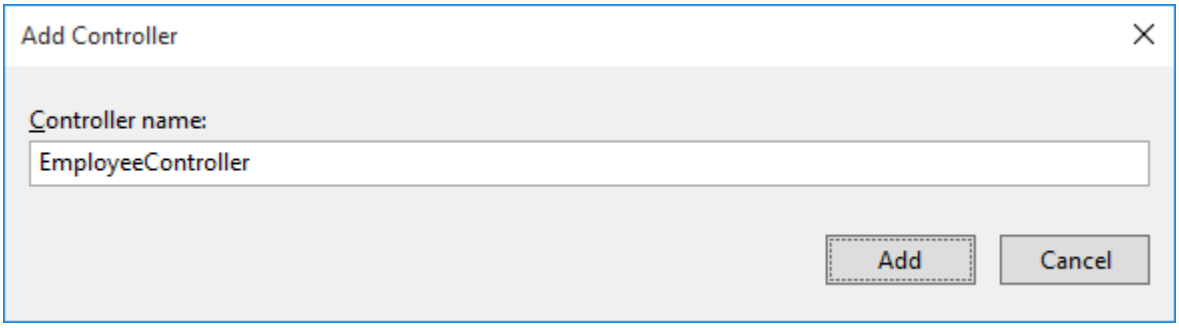
In **Solution Explorer**, right-click the *Controllers* folder and then click **Add**, then **Controller**.



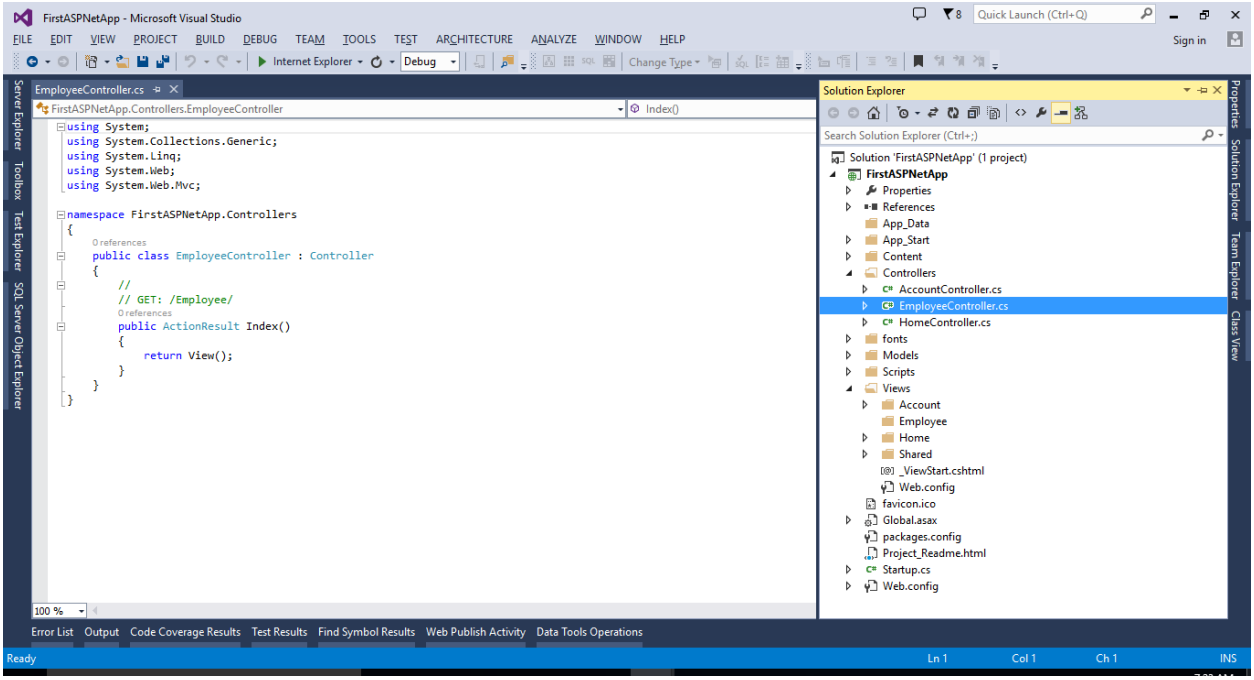
In the **Add Scaffold** dialog box, click **MVC 5 Controller - Empty**, and then click **Add**.



Name your new controller "EmployeeController" and click **Add**.



Notice in the **Solution Explorer** a new file has been created named "*EmployeeController.cs*" under Controllers folder and a new folder "*Views\Employee*".



Go to EmployeeController.cs file and make the following changes:

Add using statement at the top of the file:

```
using FirstASPNetMVCApp.Models;
```

Design the EmployeeController class as below:

```
public class EmployeeController : Controller
    {
        private EmployeeMgmtEntities1 db = new EmployeeMgmtEntities1();

        // GET: /Employee/
        public ActionResult Index()
        {
            var employees = db.Employees.Include(e => e.Department1).Include(e =>
e.Designation1);
            return View(employees.ToList());
        }

        // GET: /Employee/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Employee employee = db.Employees.Find(id);
            if (employee == null)
            {
                return HttpNotFound();
            }
            return View(employee);
        }
```

```
        // GET: /Employee/Create
        public ActionResult Create()
        {
            ViewBag.Department = new SelectList(db.Departments, "ID", "Name");
            ViewBag.Designation = new SelectList(db.Designations, "ID", "Name");
            return View();
        }

        // POST: /Employee/Create
        [HttpPost]
        public ActionResult Create([Bind(Include="ID,Name,Designation,Department")]
Employee employee)
        {
            if (ModelState.IsValid)
            {
                db.Employees.Add(employee);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.Department = new SelectList(db.Departments, "ID", "Name",
employee.Department);
            ViewBag.Designation = new SelectList(db.Designations, "ID", "Name",
employee.Designation);
            return View(employee);
        }

        // GET: /Employee/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Employee employee = db.Employees.Find(id);
            if (employee == null)
            {
                return HttpNotFound();
            }
            ViewBag.Department = new SelectList(db.Departments, "ID", "Name",
employee.Department);
            ViewBag.Designation = new SelectList(db.Designations, "ID", "Name",
employee.Designation);
            return View(employee);
        }

        // POST: /Employee/Edit/5
        [HttpPost]
        public ActionResult Edit([Bind(Include="ID,Name,Designation,Department")]
Employee employee)
        {
            if (ModelState.IsValid)
            {
                db.Entry(employee).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
```

```
            ViewBag.Department = new SelectList(db.Departments, "ID", "Name",
employee.Department);
            ViewBag.Designation = new SelectList(db.Designations, "ID", "Name",
employee.Designation);
            return View(employee);
        }

        // GET: /Employee/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Employee employee = db.Employees.Find(id);
            if (employee == null)
            {
                return HttpNotFound();
            }
            return View(employee);
        }

        // POST: /Employee/Delete/5
        [HttpPost, ActionName("Delete")]
        public ActionResult DeleteConfirmed(int id)
        {
            Employee employee = db.Employees.Find(id);
            db.Employees.Remove(employee);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
```

Exercise 4: Creating View

Under the View/Employee folder create four .cshtml files named Index.cshtml, Edit.cshtml, Details.cshtml, Create.cshtml and Delete.cshtml and add code in these files as below:

Index.cshtml

```
@model IEnumerable<FirstASPNetMVCApp.Models.Employee>

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>
```

```
<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Name)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Department1.Name)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Designation1.Name)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Name)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Department1.Name)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Designation1.Name)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.ID }) |
            @Html.ActionLink("Details", "Details", new { id=item.ID }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.ID })
        </td>
    </tr>
}

</table>
```

Edit.cshtml

```
@model FirstASPNetMVCApp.Models.Employee

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>


@using (Html.BeginForm())
{

    <div class="form-horizontal">
        <h4>Employee</h4>
        <hr />
        @Html.HiddenFor(model => model.ID)
```

```
        <div class="form-group">
            @Html.LabelFor(model => model.Name, new { @class = "control-label col-md-2"
})
            <div class="col-md-10">
                @Html.EditorFor(model => model.Name)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Designation, "Designation", new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("Designation", String.Empty)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Department, "Department", new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("Department", String.Empty)
            </div>
        </div>

        <div>
            <div>
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

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

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Details.cshtml

```
@model FirstASPNetMVCApp.Models.Employee

@{
    ViewBag.Title = "Details";
}

<h2>Details</h2>

<div>
    <h4>Employee</h4>
       <hr />
    <dl class="dl-horizontal">
        <dt>
```

```
            @Html.DisplayNameFor(model => model.Name)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Name)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Department1.Name)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Department1.Name)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Designation1.Name)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Designation1.Name)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.ID }) |
    @Html.ActionLink("Back to List", "Index")
</p>
```

Create.cshtml

```
@model FirstASPNetMVCApp.Models.Employee

@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>


@using (Html.BeginForm())
{

    <div class="form-horizontal">
        <h4>Employee</h4>
        <hr />

        <div class="form-group">
            @Html.LabelFor(model => model.ID, new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ID)
            </div>
        </div>

        <div class="form-group">
```

```
            @Html.LabelFor(model => model.Name, new { @class = "control-label col-md-2"
})
            <div class="col-md-10">
                @Html.EditorFor(model => model.Name)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Designation, "Designation", new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("Designation", String.Empty)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Department, "Department", new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("Department", String.Empty)
            </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>
    </div>
}

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

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Delete.cshtml

```
@model FirstASPNetMVCApp.Models.Employee

@{
    ViewBag.Title = "Delete";
}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Employee</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Name)
```

```
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Name)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Department1.Name)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Department1.Name)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Designation1.Name)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Designation1.Name)
        </dd>

    </dl>

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

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-default" /> |
            @Html.ActionLink("Back to List", "Index")
        </div>
    }
</div>
```

Once the views are created go to App_Start folder open RouteConfig.cs file and make the following modification in RegisterRoutes function of RouteConfig class.

```
public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

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

Press F5 and run your application to check the functionality.

Note: We are using _Layout.cshtml in this example and standard CSS classes which are available by default.