



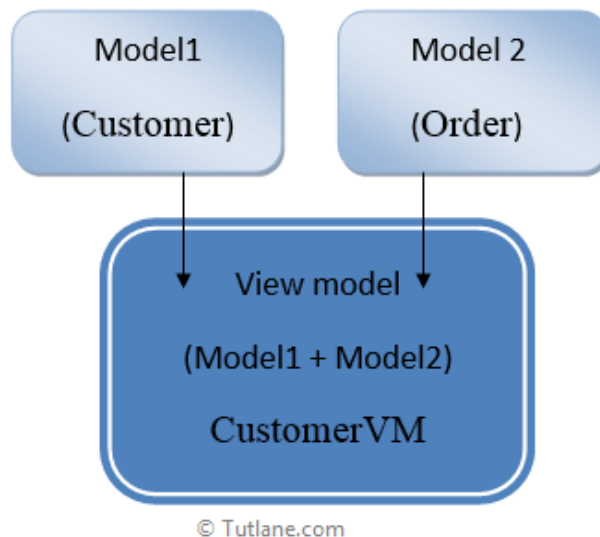
How to Use ViewModel in Asp.Net MVC with Example

Here we will learn what is viewmodel in asp.net mvc and how to use viewmodel in asp.net mvc applications with example.

ViewModel in Asp.Net MVC

The viewmodel in asp.net mvc represent only the data we want to display on view whether it is used for displaying or for taking input from view. If we want to display more than one model on view in asp.net mvc then we need to create a new viewmodel. Following image shows visual representation of view model in asp.net mvc.

Here we will learn asp.net mvc viewmodel with simple example if we have a **Customer** entity and **Order** entity. In view to display both entities (Customer entity + Order entity) data then we need to create viewmodel (CustomerVM) and we will select the properties whichever we want to display from both entities (Customer entity + Order entity) and create a viewmodel.



Now start with creating a viewmodel in asp.net mvc before that let's have look on table which we are going to use in our application.

Database Part

In this database part we will create required tables in database with following scripts.

Customer Table

following is the script to create **customer** table in your database

```
CREATE TABLE [dbo].[Customer](
[CustomerID] [int] IDENTITY(1,1) NOT NULL Primary key,
[Name] [varchar](100) NULL,
[Address] [varchar](300) NULL,
[Mobileno] [varchar](15) NULL,
[Birthdate] [datetime] NULL,
[EmailID] [varchar](300) NULL,)
```

Once we run above script it will create **Customer** table like as shown below

SAI-PC.OrderDB - dbo.Customer			
	Column Name	Data Type	Allow Nulls
🔑	CustomerID	int	<input type="checkbox"/>
	Name	varchar(100)	<input checked="" type="checkbox"/>
	Address	varchar(300)	<input checked="" type="checkbox"/>
	Mobileno	varchar(15)	<input checked="" type="checkbox"/>
	Birthdate	datetime	<input checked="" type="checkbox"/>
	EmailID	varchar(300)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

© Tutlane.com

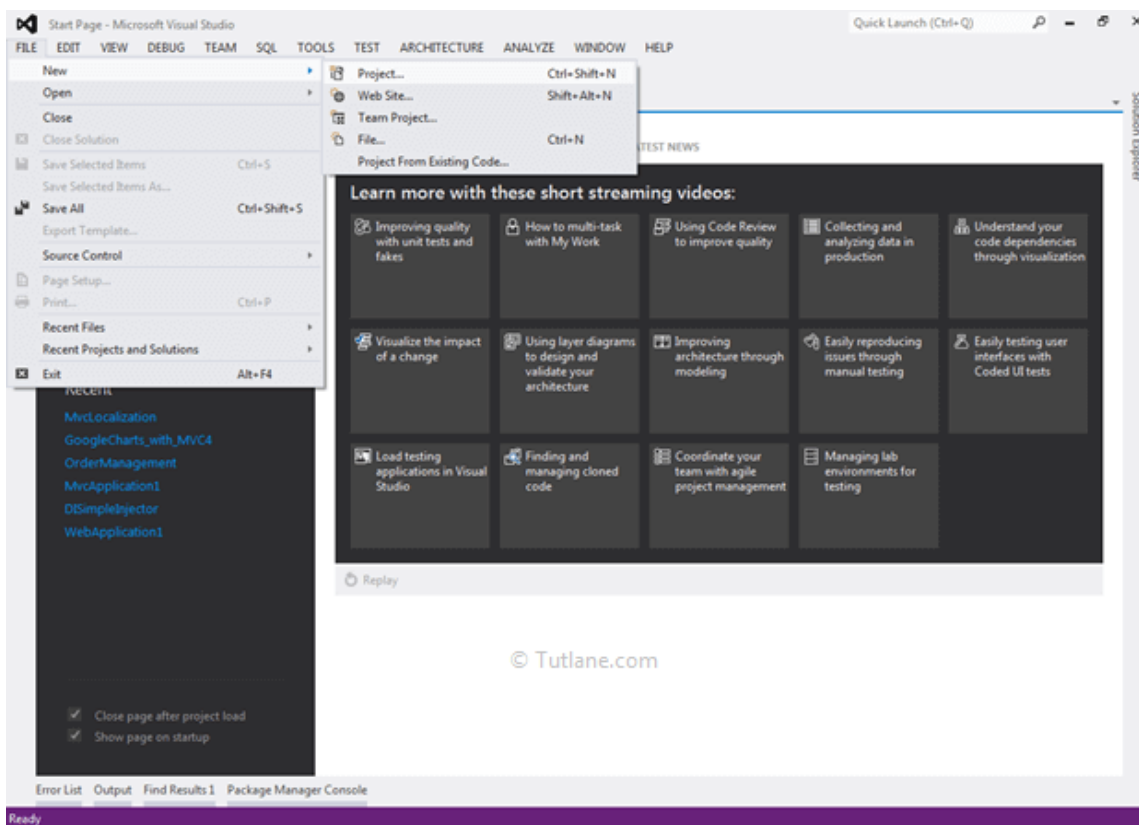
Order table

Following is the script to create order table in your database

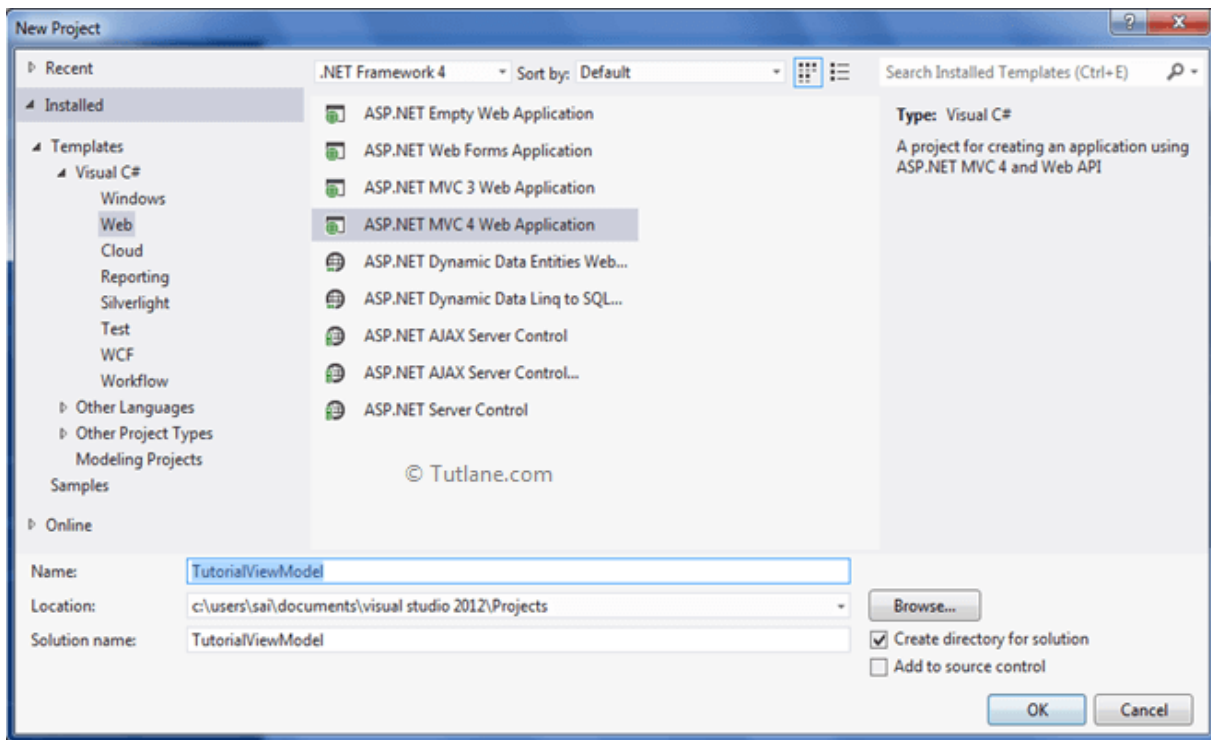
```
CREATE TABLE [dbo].[Order](
[OrderID] [int] IDENTITY(1,1) NOT NULL primary key,
[CustomerID] [int] NULL,
[OrderDate] [datetime] NULL,
[OrderPrice] [decimal](18, 0) NULL,)
```

Creating Asp.Net MVC Application

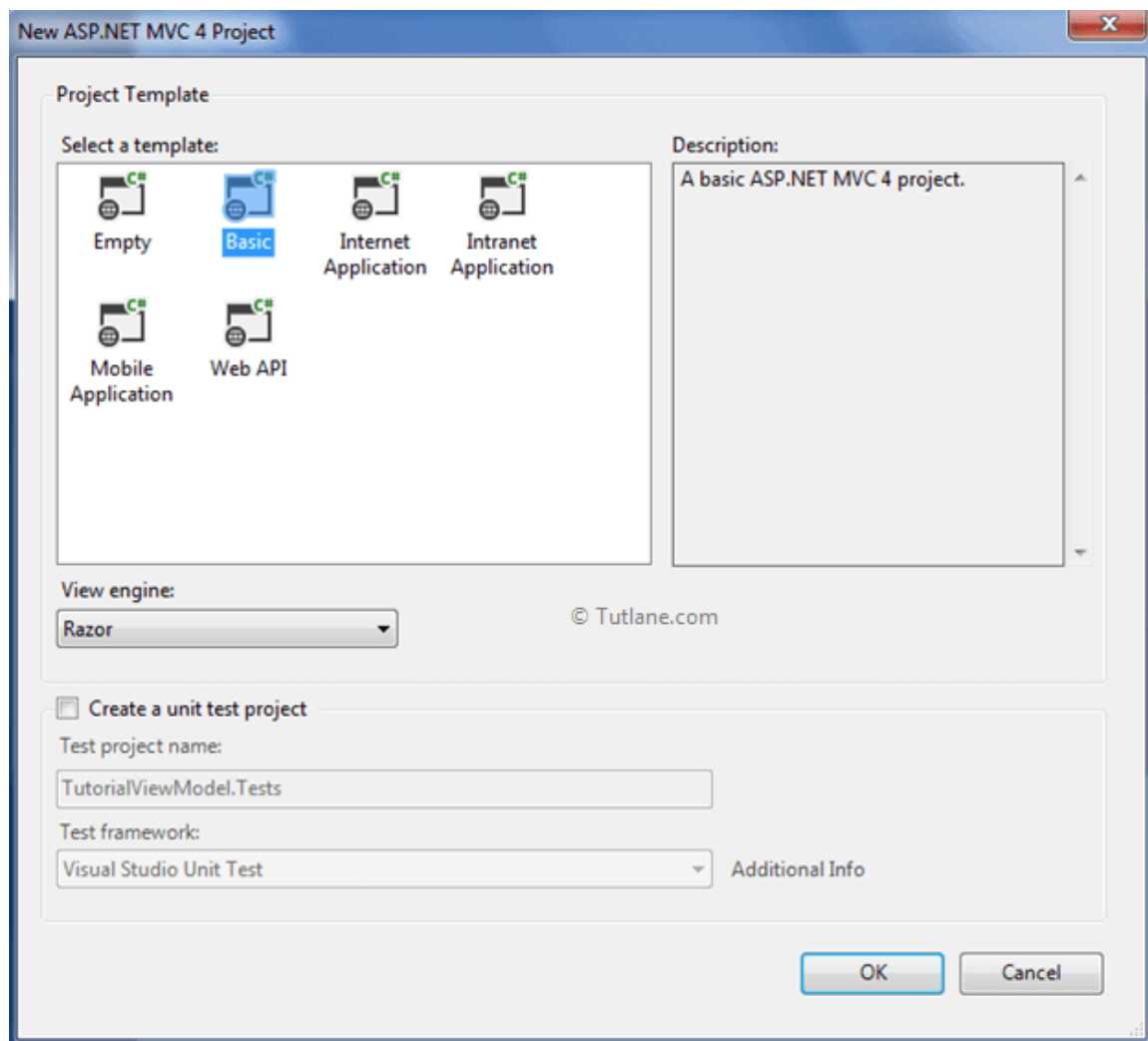
Let's start with creating new asp.net mvc 4 application for that Open visual studio studio àGo to File → Select New → Select Project



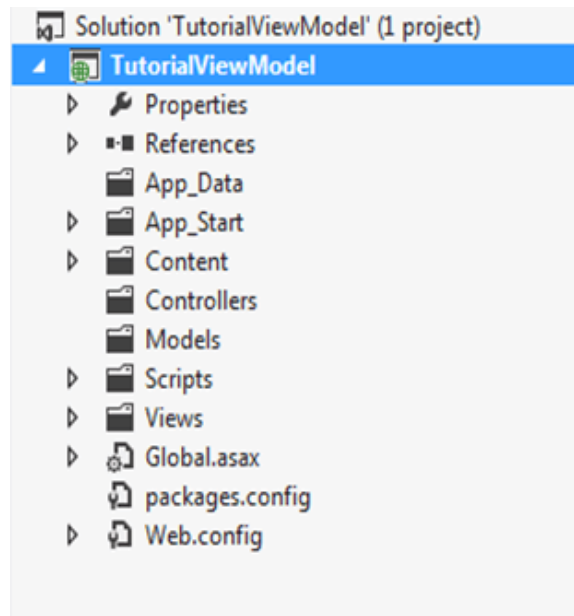
After that you will see new dialog will pop up for selecting your Template and Project type. From Templates select Visual C# → inside that select Web and then project type select ASP.NET MVC 4 Web Application and here we are giving name as **"TutorialViewModel"** then finally click on ok button



After naming it just click on OK now new dialog will pop up for selecting template in that Select Basic template and click ok like as shown below

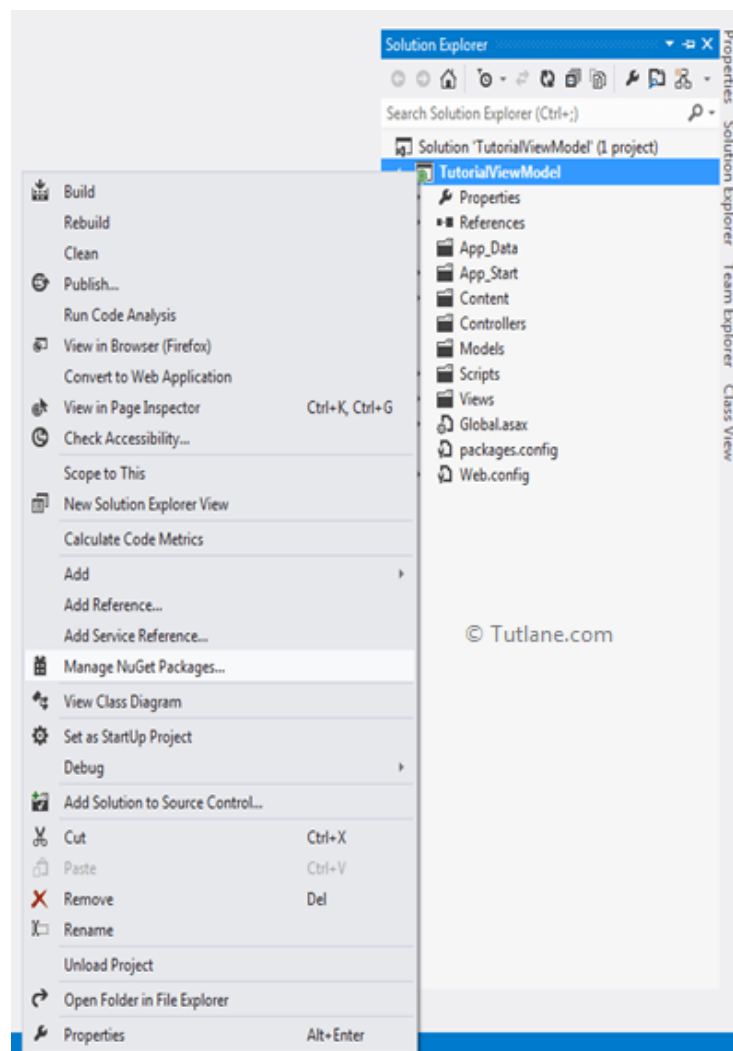


After creating the application here is the complete folder view.

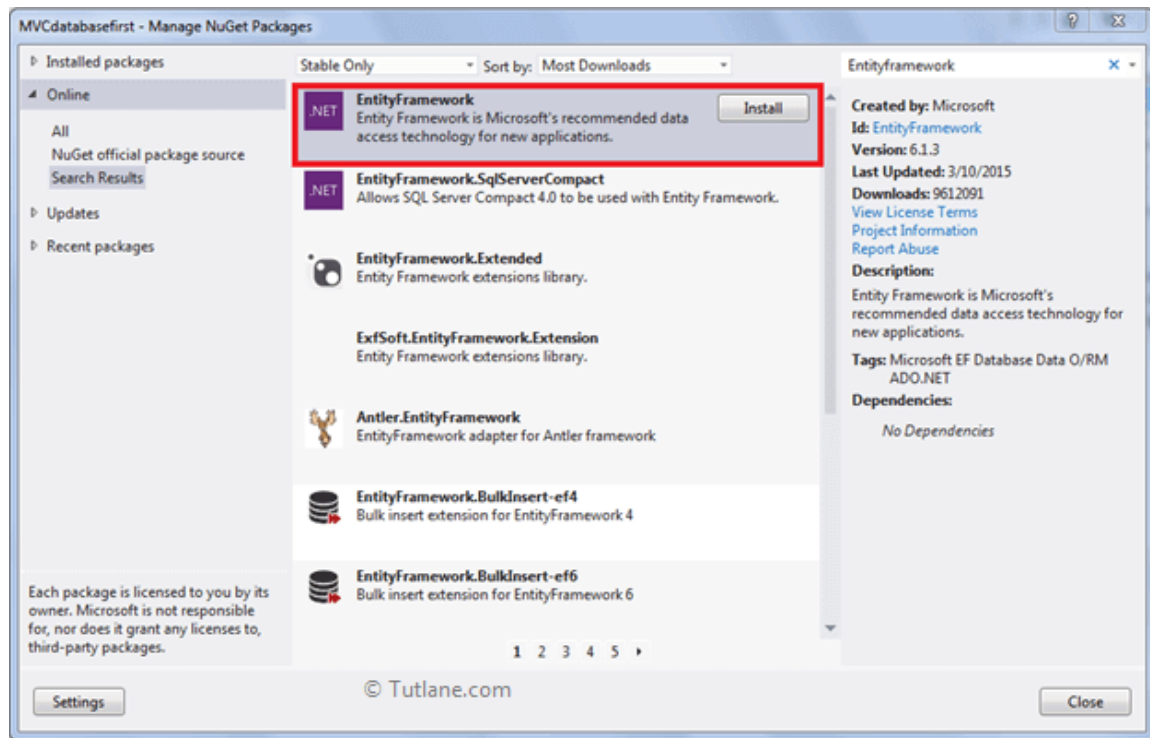


Installing Entity Framework in Application

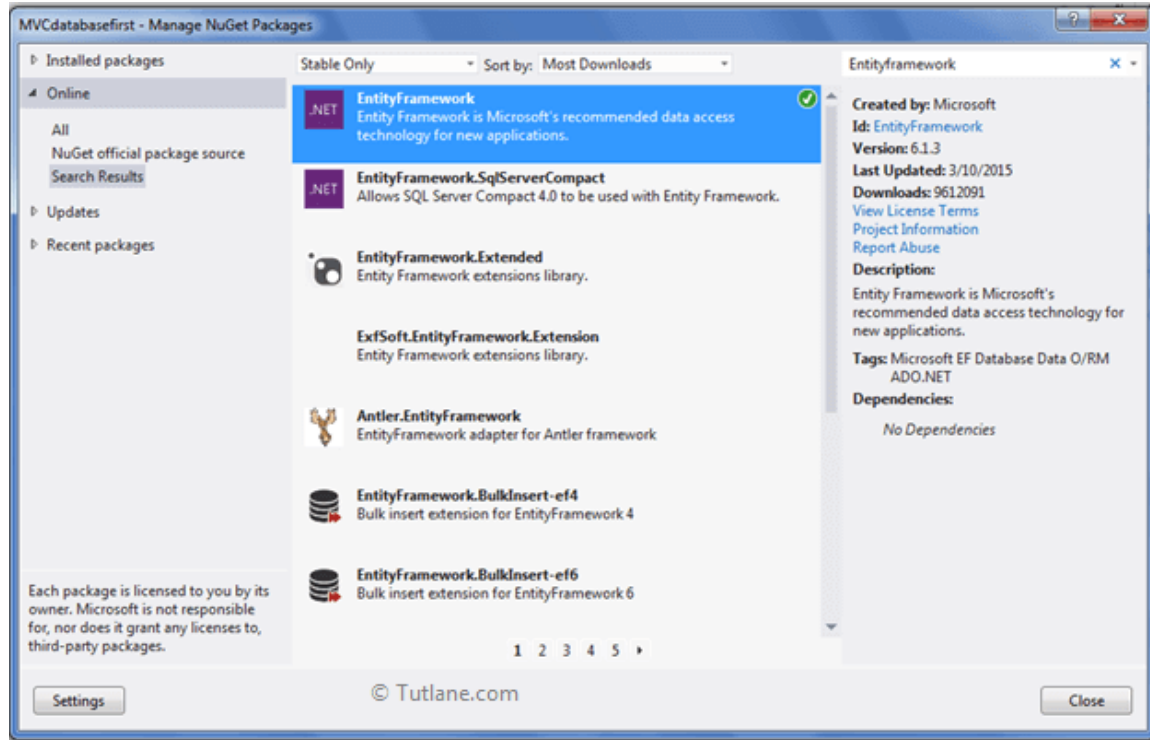
For adding Entity framework just right click on your application and from the list select "**Manage NuGet Packages**"



After select a new dialog will popup of "Manage NuGet Packages" inside search box enter "EntityFramework". After getting search value select EntityFramework click on install button.



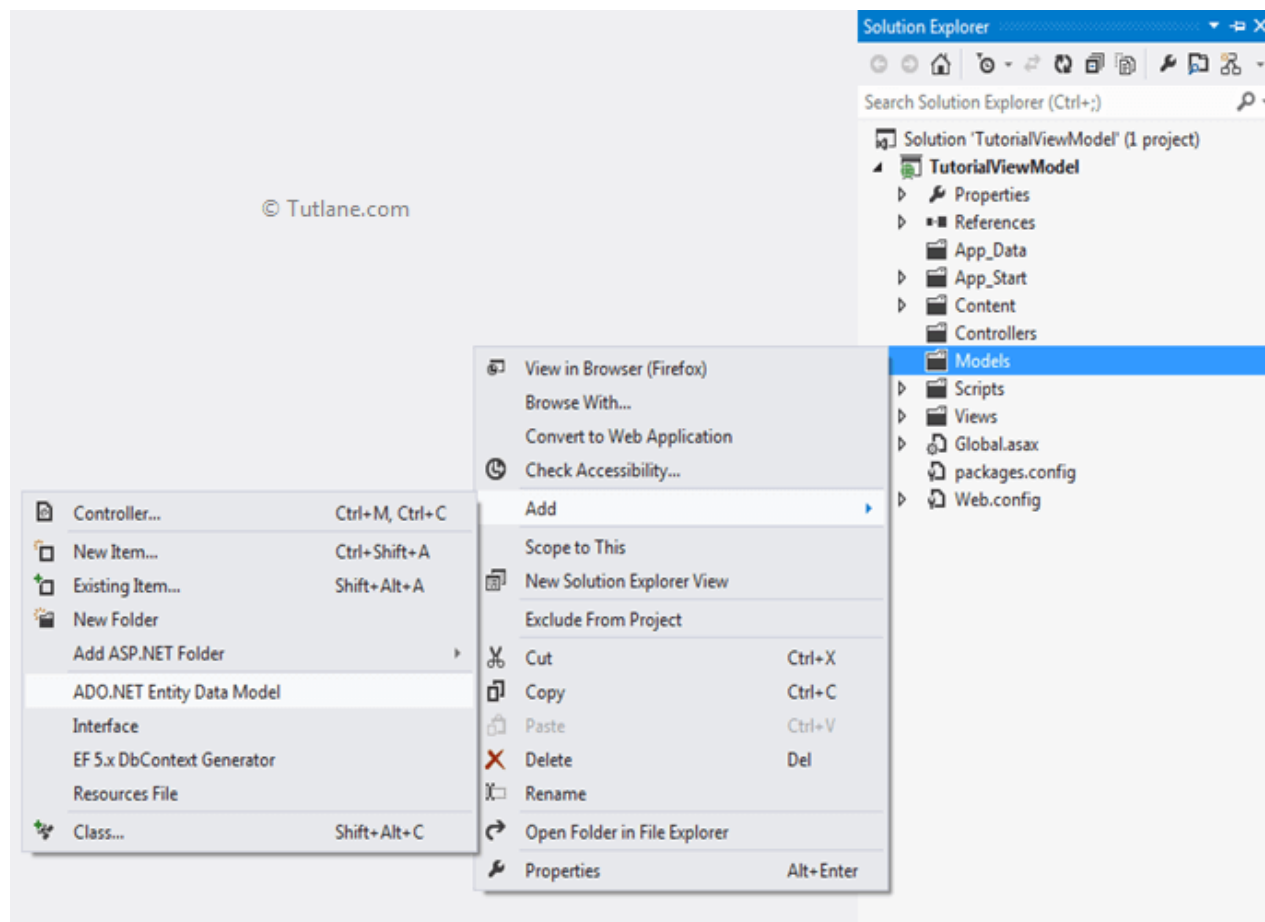
After adding it will show an ok sign in green color



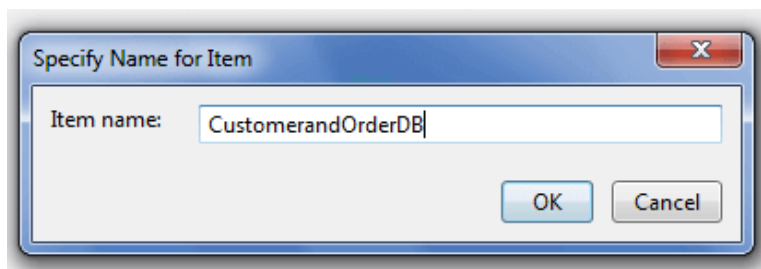
After adding Entity framework now we are going to add **ADO.NET Entity Data Model**

Adding ADO.NET Entity Data Model

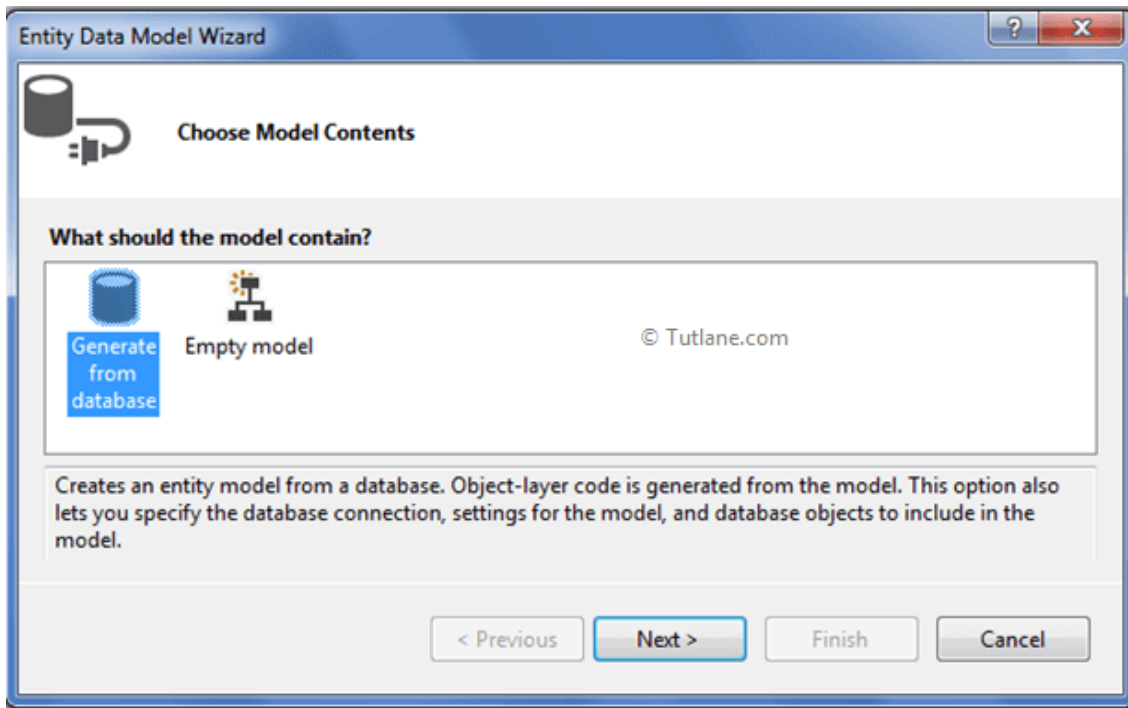
For adding ADO.NET Entity Data Model just right click on **Model** folder and select Add inside that Select **ADO.NET Entity Data Model** like as shown below



After clicking on ADO.NET Entity Data Model a New Dialog will popup for entering Item name inside that enter any name but it must be unique and click on OK button.

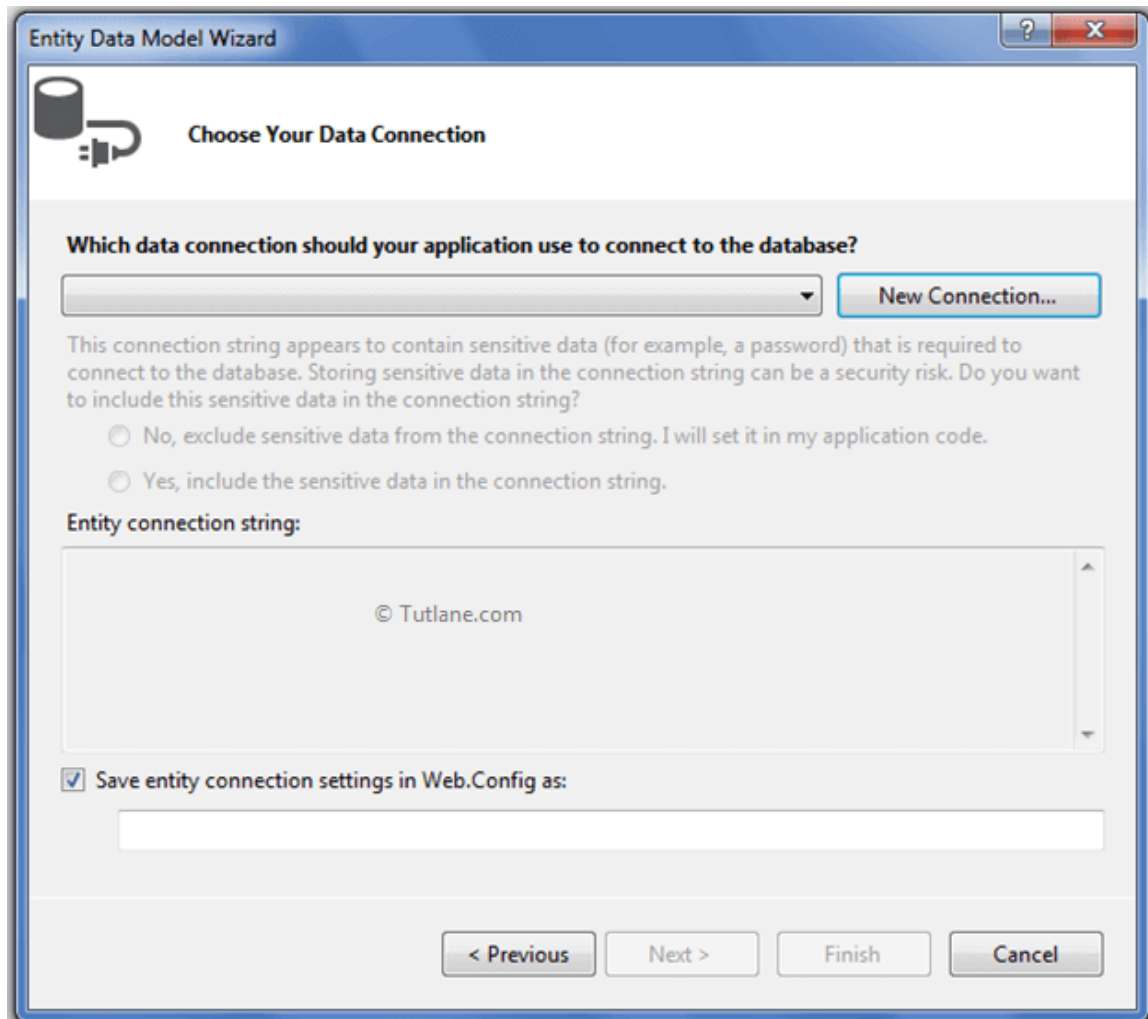


After that a new Wizard will popup where we are going to configure **Entity Data Model**. In this we are going to use Database first

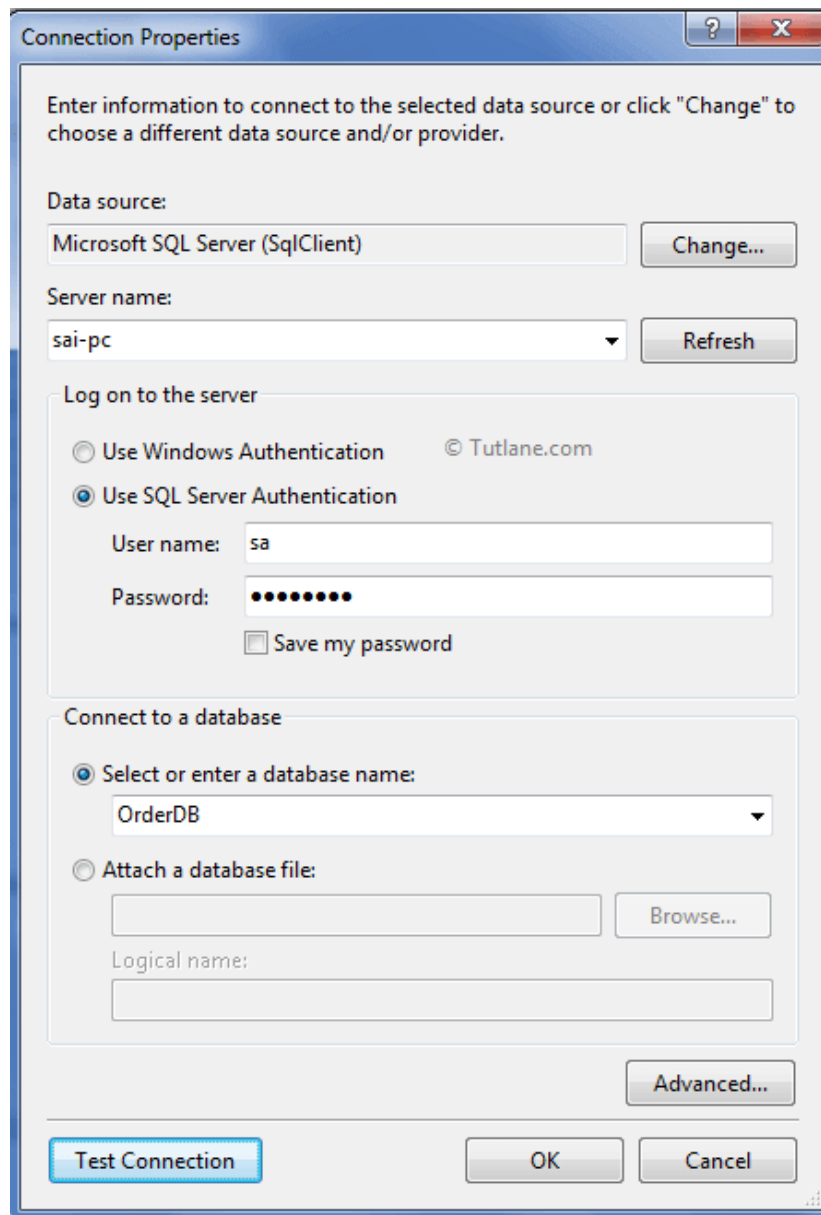


From that select **Generate from database** and click on Next button. After clicking on Next button a New Wizard will pop up for Choosing Data Connection.

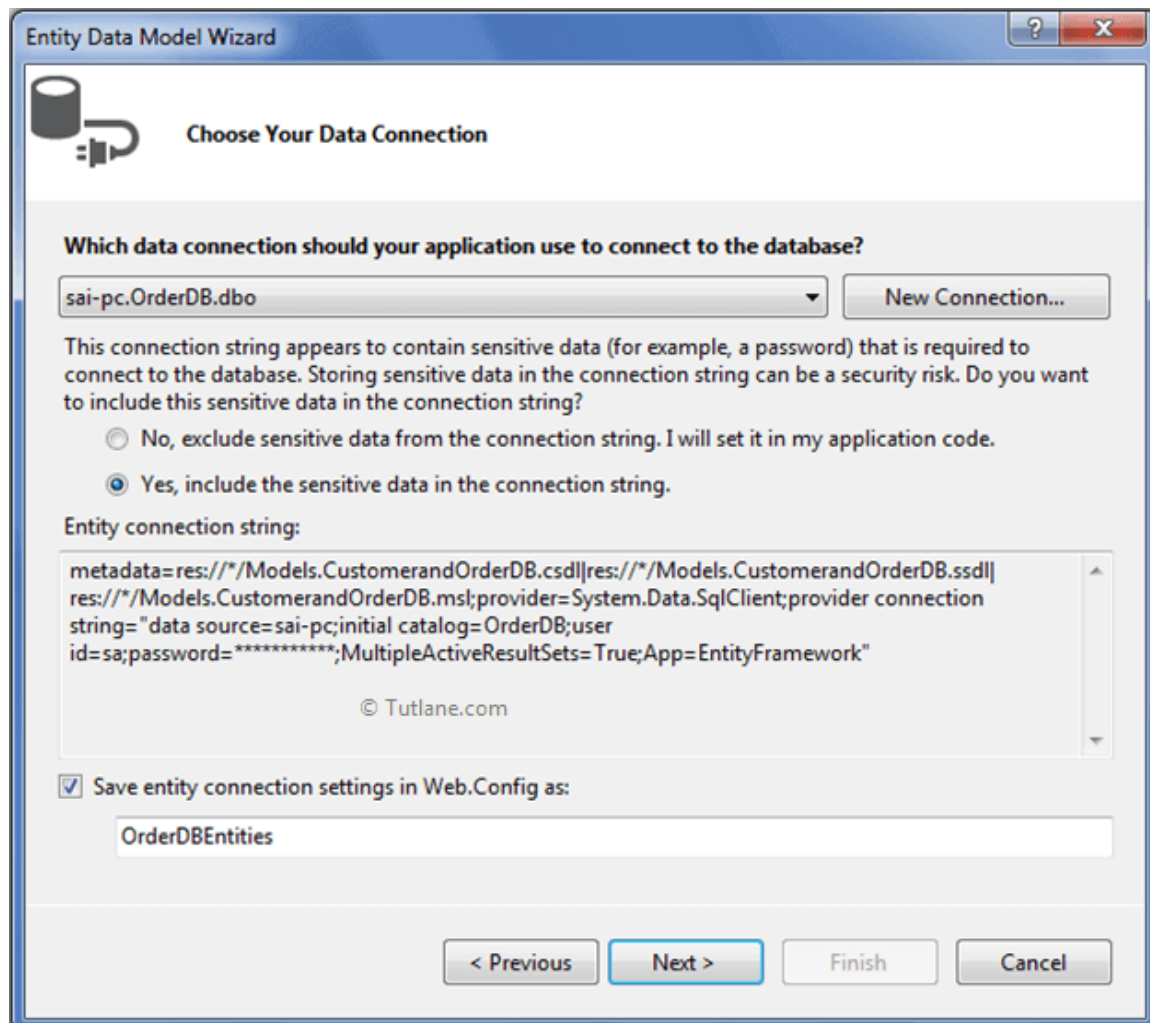
Choosing Data Connection



Now click on New Connection a new Dialog will popup. Here we need to configure it. In Server name you need to add your Sql Server Name and select either **Using Windows Authentication** or **Using Sql Server Authentication** to connect SQL Server. Here we selected **Using Sql Server Authentication** and entered User name and Password of Sql server. Last we are going to select Database Name "**OrderDB**" once we done click on OK button as shown below



After adding database connection our Entity Data Model Wizard will look like below snapshot



The image shows the 'Entity Data Model Wizard' window, specifically the 'Choose Your Data Connection' step. The window has a title bar with a question mark and a close button. Below the title bar is a header area with a database icon and the text 'Choose Your Data Connection'. The main content area asks 'Which data connection should your application use to connect to the database?'. There is a dropdown menu showing 'sai-pc.OrderDB.dbo' and a 'New Connection...' button. Below this, a warning message states: 'This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?'. There are two radio buttons: 'No, exclude sensitive data from the connection string. I will set it in my application code.' and 'Yes, include the sensitive data in the connection string.' The 'Yes' option is selected. Below the radio buttons is a text box labeled 'Entity connection string:' containing the following text: 'metadata=res://*/Models.CustomerandOrderDB.csdl|res://*/Models.CustomerandOrderDB.ssdl|res://*/Models.CustomerandOrderDB.msl;provider=System.Data.SqlClient;provider connection string="data source=sai-pc;initial catalog=OrderDB;user id=sa;password=*****;MultipleActiveResultSets=True;App=EntityFramework"'. Below the text box is a copyright notice '© Tutlane.com'. At the bottom, there is a checkbox labeled 'Save entity connection settings in Web.Config as:' which is checked. Below the checkbox is a text box containing 'OrderDBEntities'. At the very bottom are four buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'.

Entity Data Model Wizard

Choose Your Data Connection

Which data connection should your application use to connect to the database?

sai-pc.OrderDB.dbo New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string. I will set it in my application code.

☒ Yes, include the sensitive data in the connection string.

Entity connection string:

metadata=res://*/Models.CustomerandOrderDB.csdl|res://*/Models.CustomerandOrderDB.ssdl|res://*/Models.CustomerandOrderDB.msl;provider=System.Data.SqlClient;provider connection string="data source=sai-pc;initial catalog=OrderDB;user id=sa;password=*****;MultipleActiveResultSets=True;App=EntityFramework"

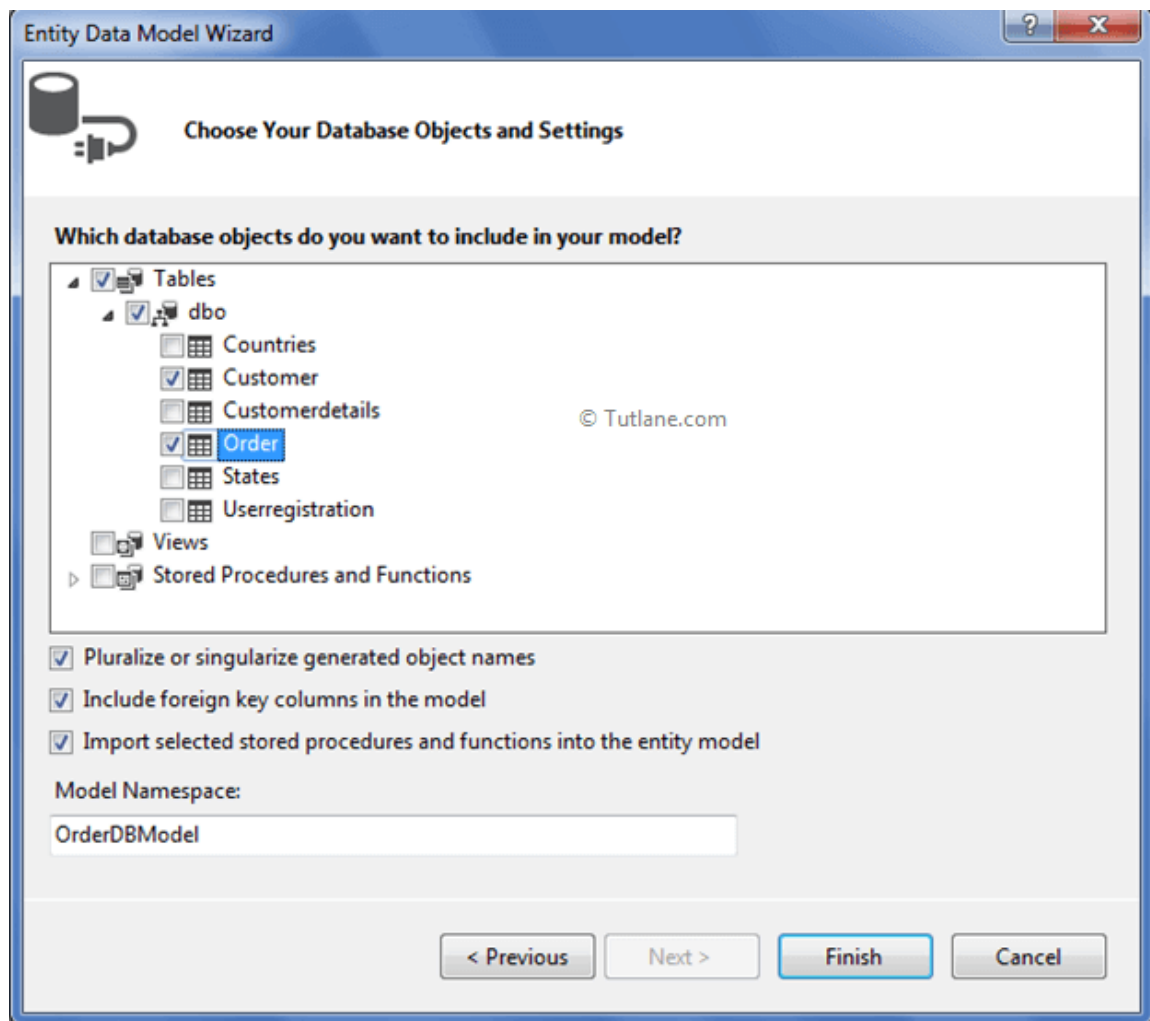
© Tutlane.com

☒ Save entity connection settings in Web.Config as:

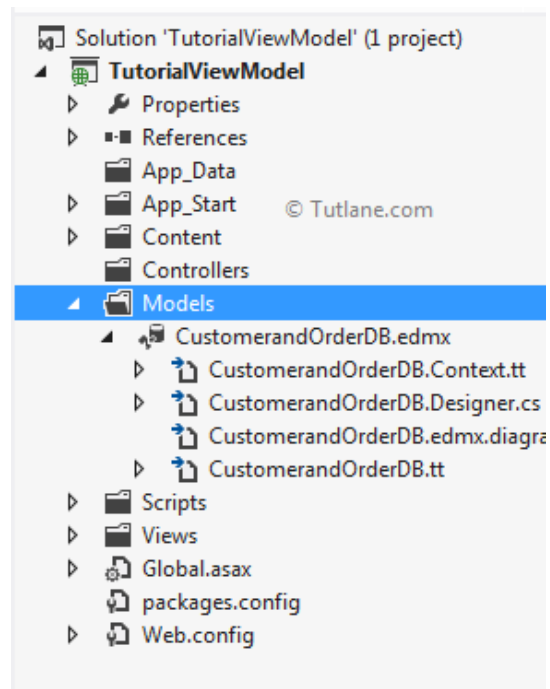
OrderDBEntities

< Previous Next > Finish Cancel

Now click on Next button. A new wizard will popup for selecting database object and in this we will see all the table which we have created in database.

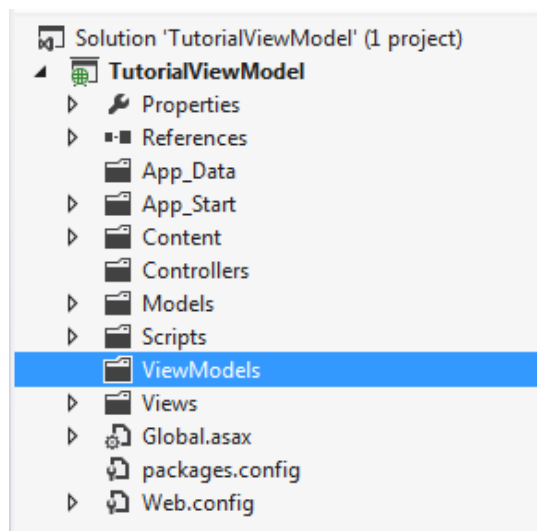


And last click on Finish button. Here is snapshot after adding ADO.NET Entity Data Model.



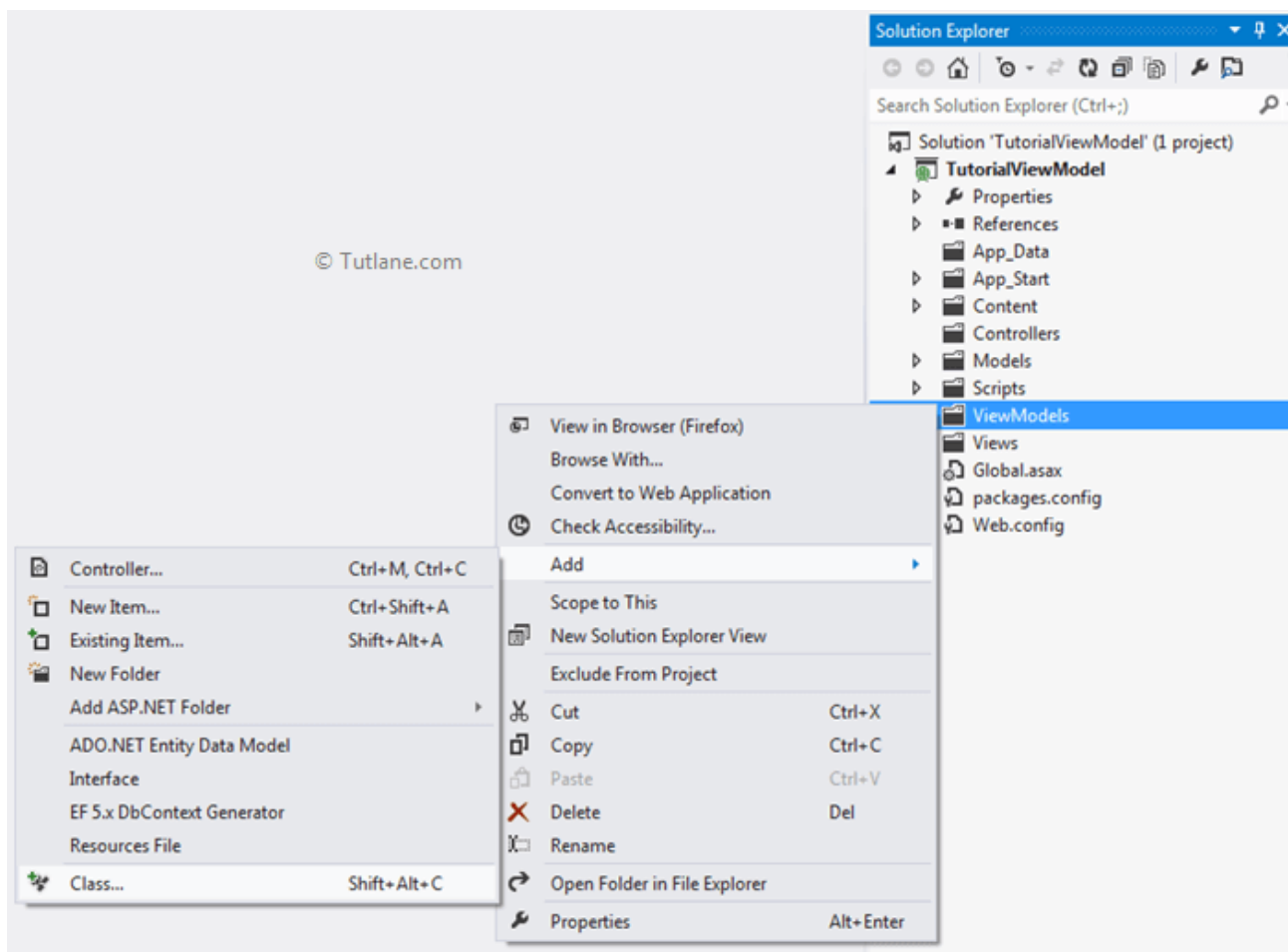
Adding ViewModels Folder

Now let's add ViewModels Folder to project for keeping all ViewModel in same folder.

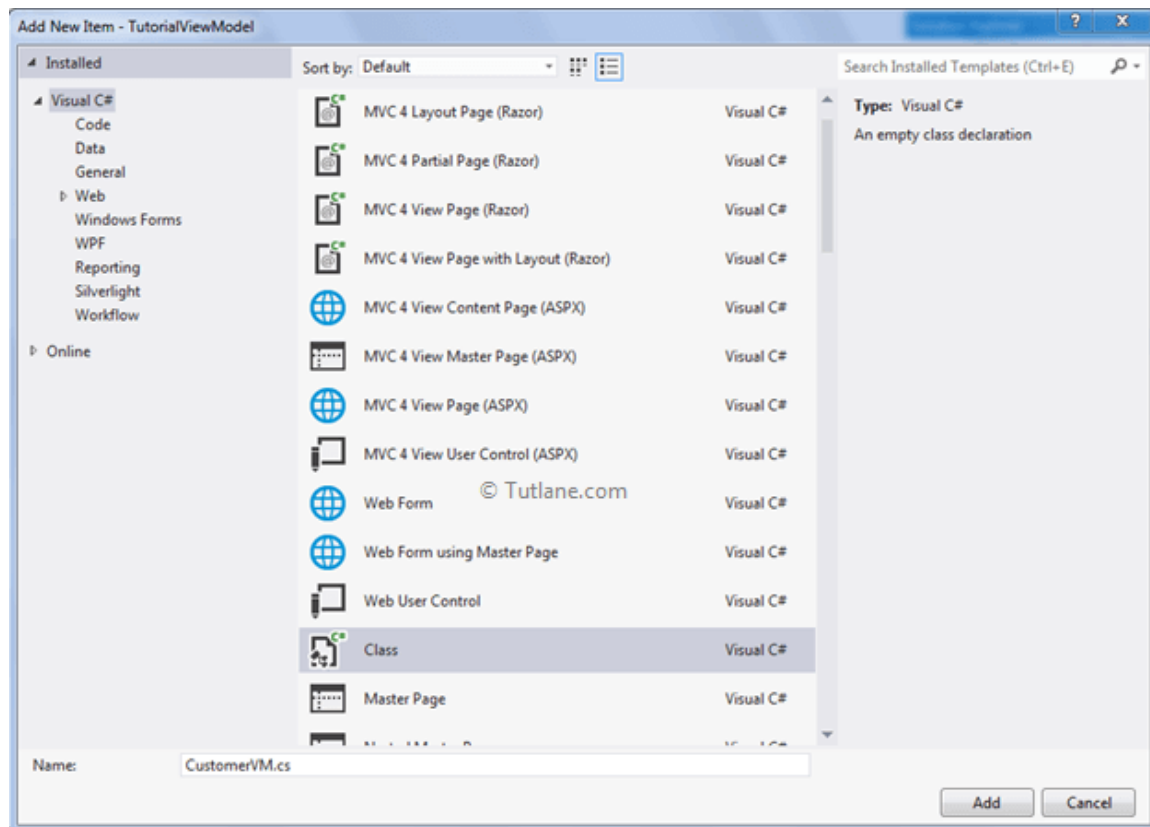


Adding CustomerVM in ViewModels folder

Inside this folder let's add ViewModel with name CustomerVM.cs for that just right Click on ViewModels folder then select Add and last select Class like as shown below



Once we select class then Add New Item dialog will popup in that select class and give name as **CustomerVM.cs**.



After adding **CustomerVM** View Model that would contain code like as shown below

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace TutorialViewModel.ViewModels
{
    public class CustomerVM
    {
    }
}
```

Now let's check out Domain Models Entities which we have add

Customer Entity

```
namespace TutorialViewModel.Models
{
    using System;
    using System.Collections.Generic;

    public partial class Customer
    {
        public int CustomerID { get; set; }
    }
}
```

```
public string Name { get; set; }
public string Address { get; set; }
public string Mobileno { get; set; }
public Nullable<System.DateTime> Birthdate { get; set; }
public string EmailID { get; set; }
}
}
```

Order Entity

```
namespace TutorialViewModel.Models
{
    using System;
    using System.Collections.Generic;

    public partial class Order
    {
        public int OrderID { get; set; }
        public Nullable<int> CustomerID { get; set; }
        public Nullable<System.DateTime> OrderDate { get; set; }
        public Nullable<decimal> OrderPrice { get; set; }
    }
}
```

ViewModel (CustomerVM.cs)

After checking Entities now let's add property from both Models to CustomerVM class. In this we took Name, Address, Mobileno from Customer model and Orderdate and OrderPrice from Order Model to display on View.

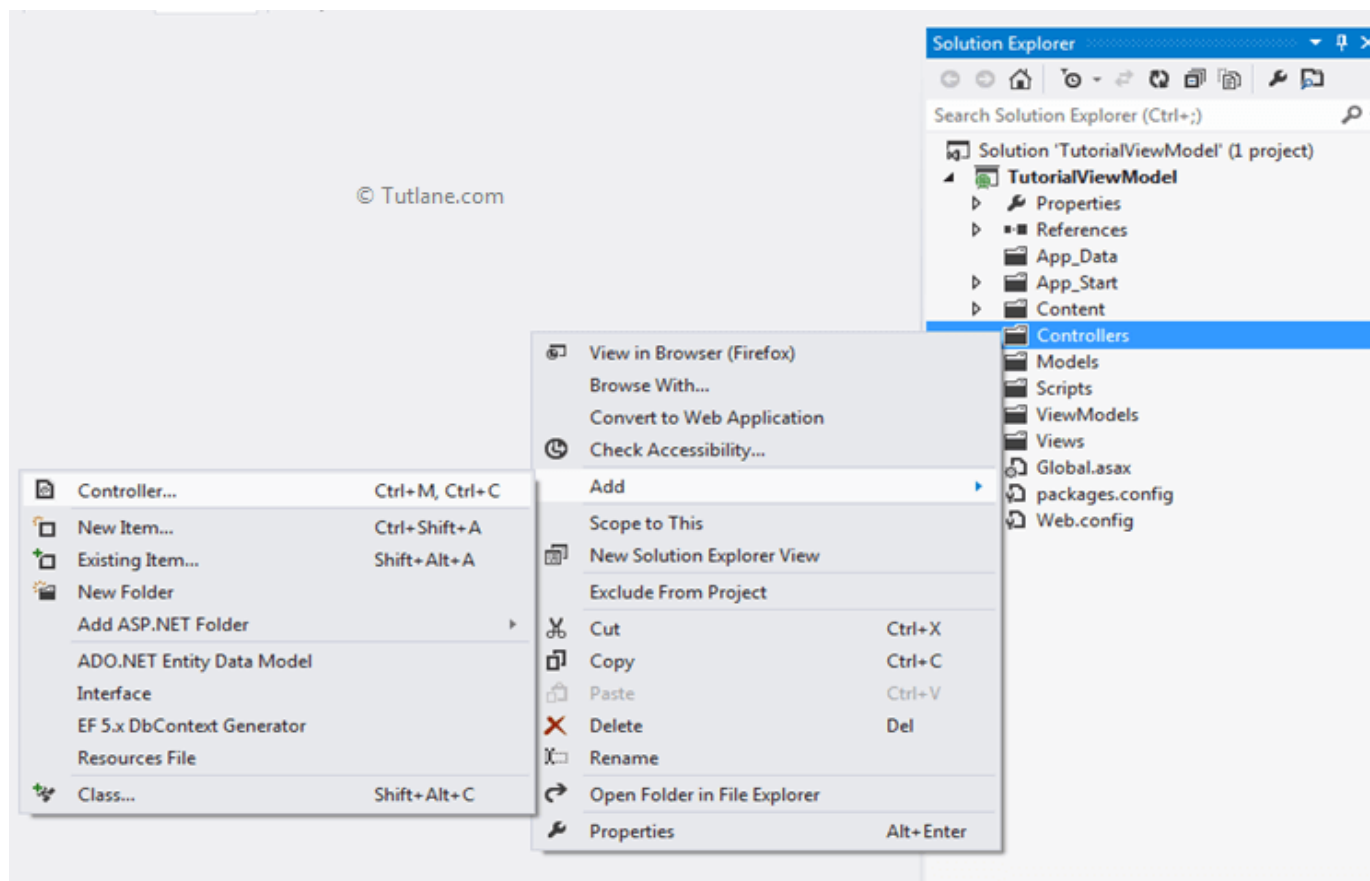
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace TutorialViewModel.ViewModels
{
    public class CustomerVM
    {
        public string Name { get; set; } // Customer
        public string Address { get; set; } // Customer
        public string Mobileno { get; set; } // Customer
        public Nullable<System.DateTime> OrderDate { get; set; } // Order
        public Nullable<decimal> OrderPrice { get; set; } // Order
    }
}
```

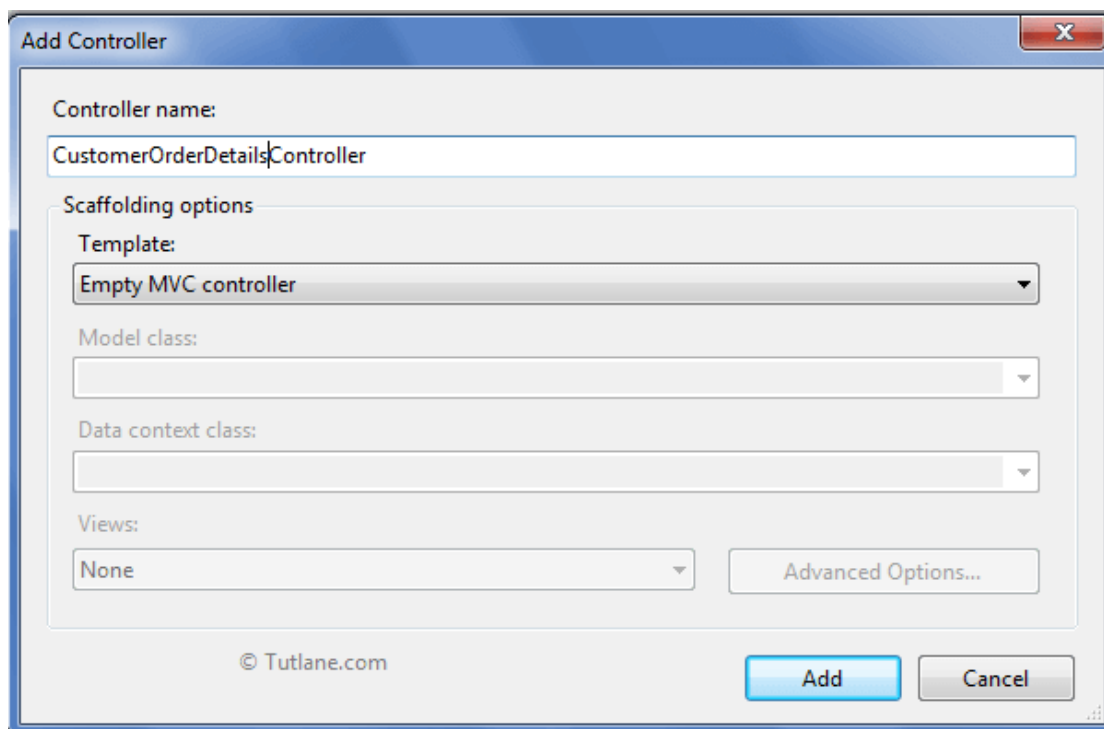
After creating ViewModel (CustomerVM) now let's add controller with Name **CustomerOrderDetails**.

Adding Controller in Application

To add controller right click on Controller Folder inside that select Add and then select Controller.



After selecting controller a new Add Controller Dialog will popup in that just add Name CustomerOrderDetailsController and in template select Empty MVC Controller and click on Add button.



After adding controller "**CustomerOrderDetailsController**" that will contain code like as shown below

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace TutorialViewModel.Controllers
{
    public class CustomerOrderDetailsController : Controller
    {

        [AcceptVerbs(HttpVerbs.Get)]
        public ActionResult Index()
        {
            return View();
        }

    }
}
```

Controller Action Methods (Index)

Now inside this Controller let's write code for getting values from database and populating ViewModel (CustomerVM).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using TutorialViewModel.Models;
using TutorialViewModel.ViewModels;
namespace TutorialViewModel.Controllers
{
    public class CustomerOrderDetailsController : Controller
    {
        [AcceptVerbs(HttpVerbs.Get)]
        public ActionResult Index()
        {
            OrderDBEntities orderdb = new OrderDBEntities(); //dbconnect class
            List<CustomerVM> CustomerVMlist = new List<CustomerVM>(); // to hold list of Customer and order
            ls
            var customerlist = (from Cust in orderdb.Customers
            join Ord in orderdb.Orders on Cust.CustomerID equals Ord.CustomerID
            selectnew { Cust.Name, Cust.Mobileno, Cust.Address, Ord.OrderDate, Ord.OrderPrice}).ToList();
            //query getting data from database from joining two tables and storing data in customerlist
            t
            foreach (var item in customerlist)
```

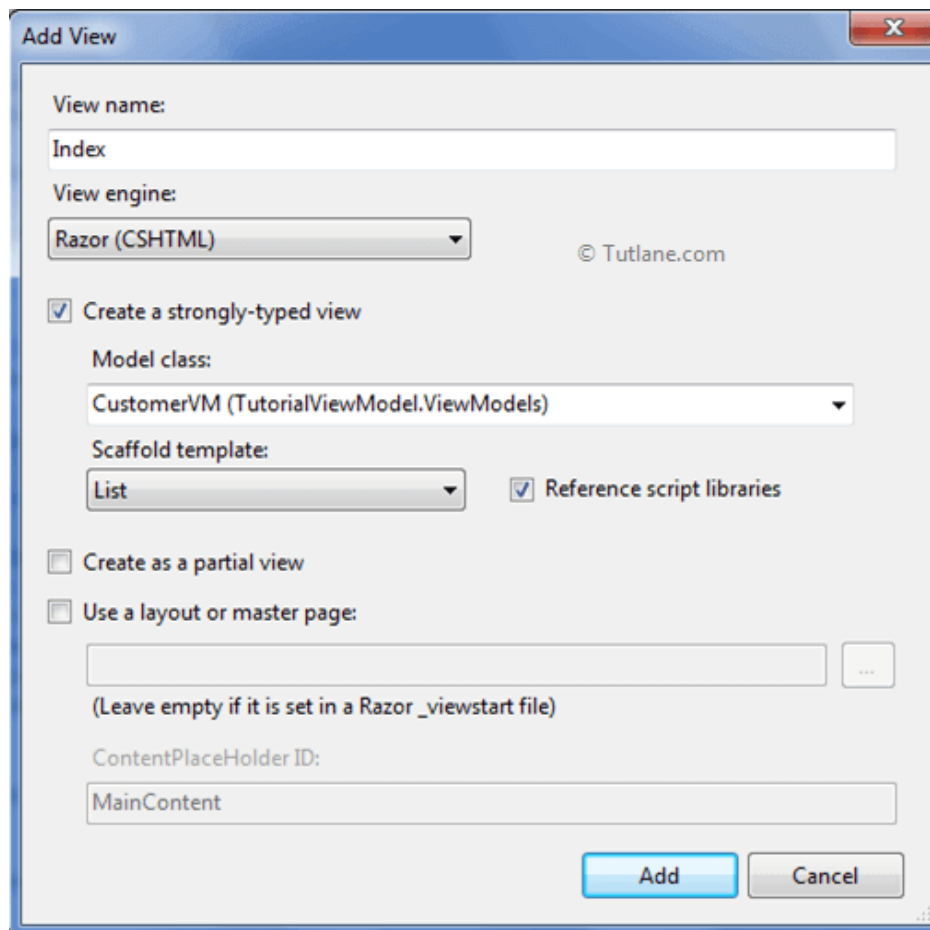


```
{  
CustomerVM objcvm = new CustomerVM(); // ViewModel  
objcvm.Name = item.Name;  
objcvm.Mobileno = item.Mobileno;  
objcvm.Address = item.Address;  
objcvm.OrderDate = item.OrderDate;  
objcvm.OrderPrice = item.OrderPrice;  
CustomerVMlist.Add(objcvm);  
}  
//Using foreach loop fill data from custmerlist to List<CustomerVM>.  
return View(CustomerVMlist); //List of CustomerVM (ViewModel)  
}  
}  
}
```

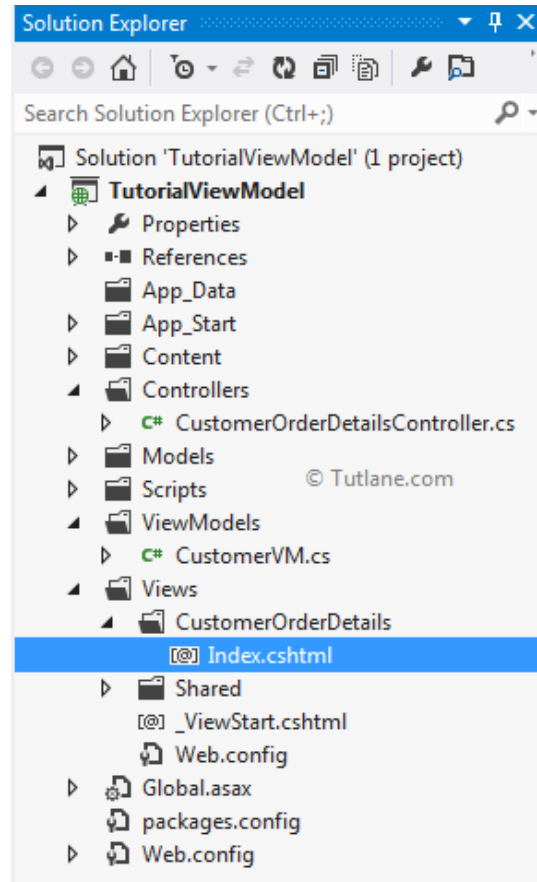
Now inside this controller I have wrote a Linq query for getting data from database with join both tables with (CustomerID) and stored it in (customerlist) and then applying foreach loop for pushing that data into (List<CustomerVM> CustomerVMlist) and lastly sending it to View.

Adding View in Application

For Adding View just right click inside Index ActionResult Method and Select "Add View" to create the view template for our Index form. Here in below snapshot we selected View engine as **Razor** and we are going to create a strongly type view for that we selected Model class **CustomerVM** and we want to create a List form for that we selected List in Scaffold template finally click on Add button.



After Adding View (Index view) the complete folder view of Project



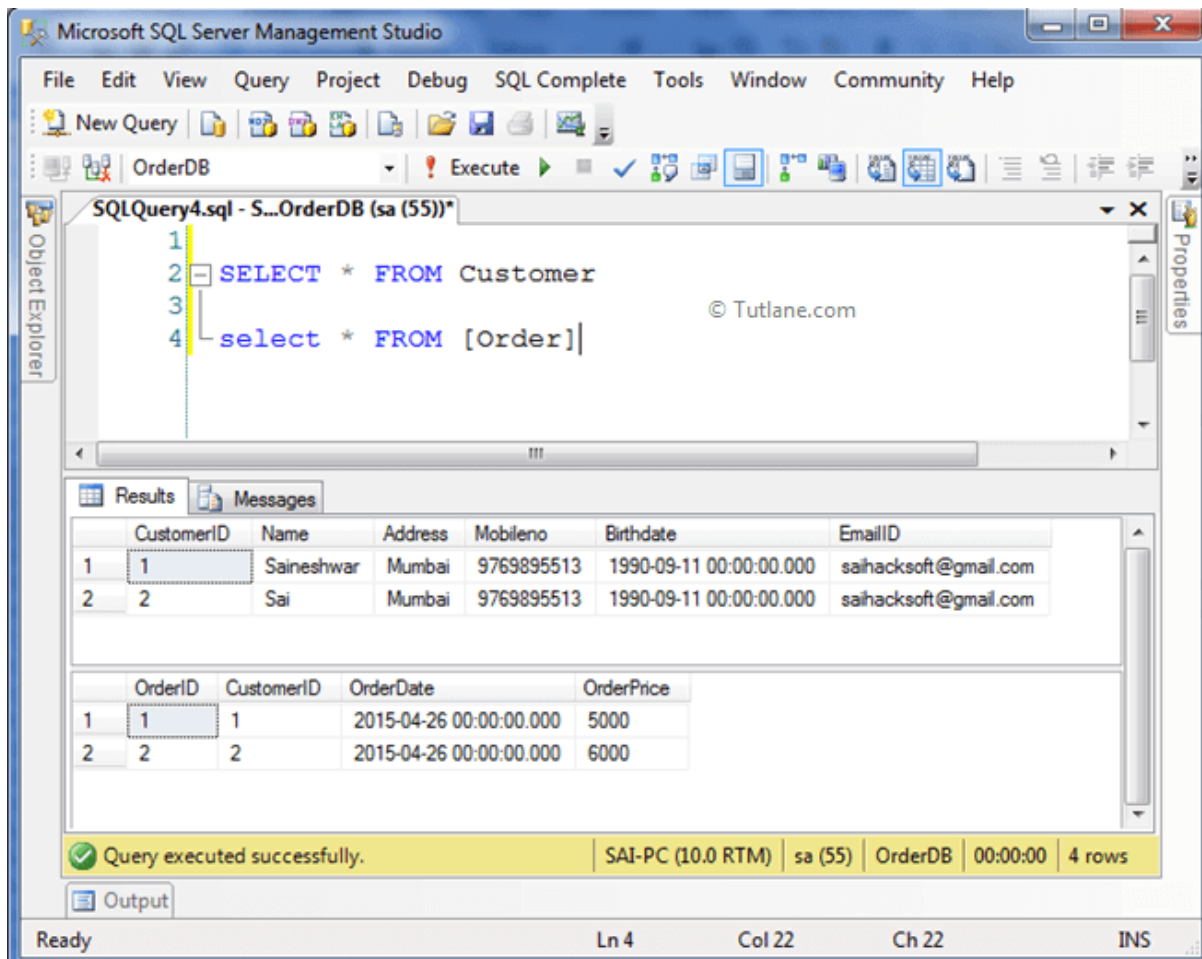
Index.cshtml View

After adding view below is a complete code of Index.cshtml which is generated.

```
@model IEnumerable<TutorialViewModel.ViewModels.CustomerVM>
@{
    Layout = null;
}
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width"/>
<title>Index</title>
</head>
<body>
<table cellpadding="5px">
<tr>
<th>
@Html.DisplayNameFor(model => model.Name)
</th>
<th>
@Html.DisplayNameFor(model => model.Address)
</th>
</tr>
</table>
```

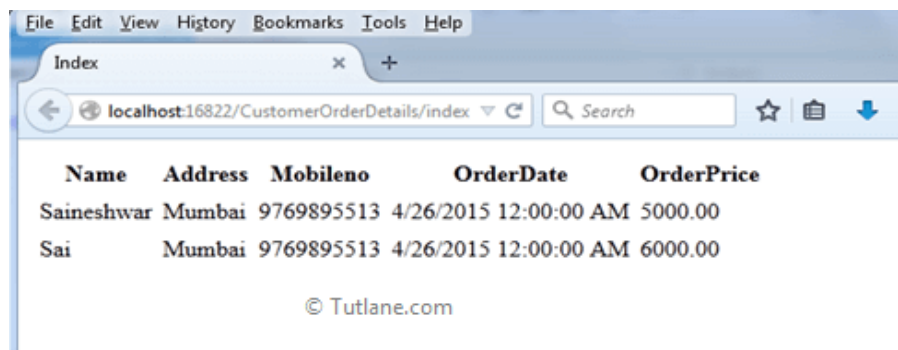
```
<th>
@Html.DisplayNameFor(model => model.Mobileno)
</th>
<th>
@Html.DisplayNameFor(model => model.OrderDate)
</th>
<th>
@Html.DisplayNameFor(model => model.OrderPrice)
</th>
</tr>
@foreach (var item in Model) {
<tr>
<td>
@Html.DisplayFor(modelItem => item.Name)
</td>
<td>
@Html.DisplayFor(modelItem => item.Address)
</td>
<td>
@Html.DisplayFor(modelItem => item.Mobileno)
</td>
<td>
@Html.DisplayFor(modelItem => item.OrderDate)
</td>
<td>
@Html.DisplayFor(modelItem => item.OrderPrice)
</td>
</tr>
}
</table>
</body>
</html>
```

In above generated View we have data from both models Customer and Order and then we have stored data in CustomerVM (ViewModel) and finally displayed on view. Our both tables **Customer** and **Order** contains data like as shown below



Now just run application and access page by entering URL like <http://localhost:####/CustomerOrderDetails/index>

Our output of viewmodel in asp.net mvc application will be like as shown below



Generally View Models in asp.net mvc are easy to use if we clear with information like where we want to display data or get input data from various domain models then always use ViewModels.

This is how we can create viewmodel in asp.net mvc and use it in mvc applications based on our requirements.



CONTACT US

📍 **Address:** No.1-93, Pochamma Colony, Manikonda, Hyderabad, Telangana - 500089

✉ **Email:** support@tutlane.com (mailto:support@tutlane.com)