

# Membangun Aplikasi Web Menggunakan Entity Framework dan MVC 5: Bagian 1

## Pendahuluan

Teknologi yang terus berkembang dan sebagai pengembang kita perlu untuk mengatasi dengan apa yang terbaru atau setidaknya populer saat ini. Sebagai pemula Anda mungkin menemukan kesulitan dengan teknologi terbaru karena akan memberikan Anda lebih banyak kebingungan apa yang membuat teknologi untuk menggunakan dan di mana untuk memulai. Kita tahu bahwa ada banyak sumber daya di luar sana yang dapat Anda gunakan sebagai referensi untuk belajar tetapi Anda masih merasa sulit untuk menghubungkan titik-titik dalam gambar. Kadang-kadang Anda mungkin berpikir kehilangan minat untuk belajar dan menyerah. Jika Anda bingung dan tidak tahu bagaimana untuk memulai membangun aplikasi web asp mvc dari awal, maka modul ini adalah untuk Anda.

## Latar Belakang

Membuat Aplikasi Web menggunakan Entity Framework dan MVC 5 ditargetkan pemula yang ingin terjun di ASP.NET MVC 5 dengan contoh praktis. Saya telah menulis seri modul ini sedemikian rupa sehingga mudah untuk mengikuti dan memahami dengan menyediakan proses langkah-demi-langkah untuk menciptakan aplikasi web sederhana dari awal dan menggunakan itu untuk IIS Web Server. Saat Anda mengerjakan sampai waktu Anda selesai mengikuti pembelajaran ini, Anda akan belajar bagaimana membuat database menggunakan SQL Server, mempelajari konsep ASP.NET MVC dan apa itu semua tentang, belajar Entity Framework, belajar dasar jQuery dan AJAX, belajar untuk membuat halaman seperti Pendaftaran, Login, Profil dan halaman Admin di mana pengguna dapat memodifikasi, menambah dan menghapus informasi. Anda juga akan belajar cara instal dan menyebarkan aplikasi Anda di IIS Web Server.

## Prasyarat

Sebelum Anda melangkah lebih jauh pastikan bahwa Anda memiliki pengetahuan dasar tentang teknologi berikut:

- SQL Server
- Visual Studio
- ASP.NET MVC and how stuff works in MVC
- Entity Framework
- C#
- LINQ basic
- Basics on HTML, CSS and JavaScript/jQuery ASP.NET MVC Overview

## **Environment dan Development Tools**

Berikut ini adalah alat dan environment setting yang saya gunakan dalam membangun aplikasi web.

- Windows 8.1 keatas
- IIS8
- Visual Studio 2015
- SQL Express 2014
- SQL Server Management Studio (Express)

## **Mari Mulai!**

Saya akan mencoba untuk menjaga demo ini sesederhana mungkin sehingga pemula dapat dengan mudah mengikuti. Dengan "sederhana" Maksudnya membatasi berbicara tentang teori dan konsep, melainkan melompat langsung ke dalam lumpur dan mendapatkan tangan Anda kotor dengan contoh kode.

## **Apa yang akan dipelajari:**

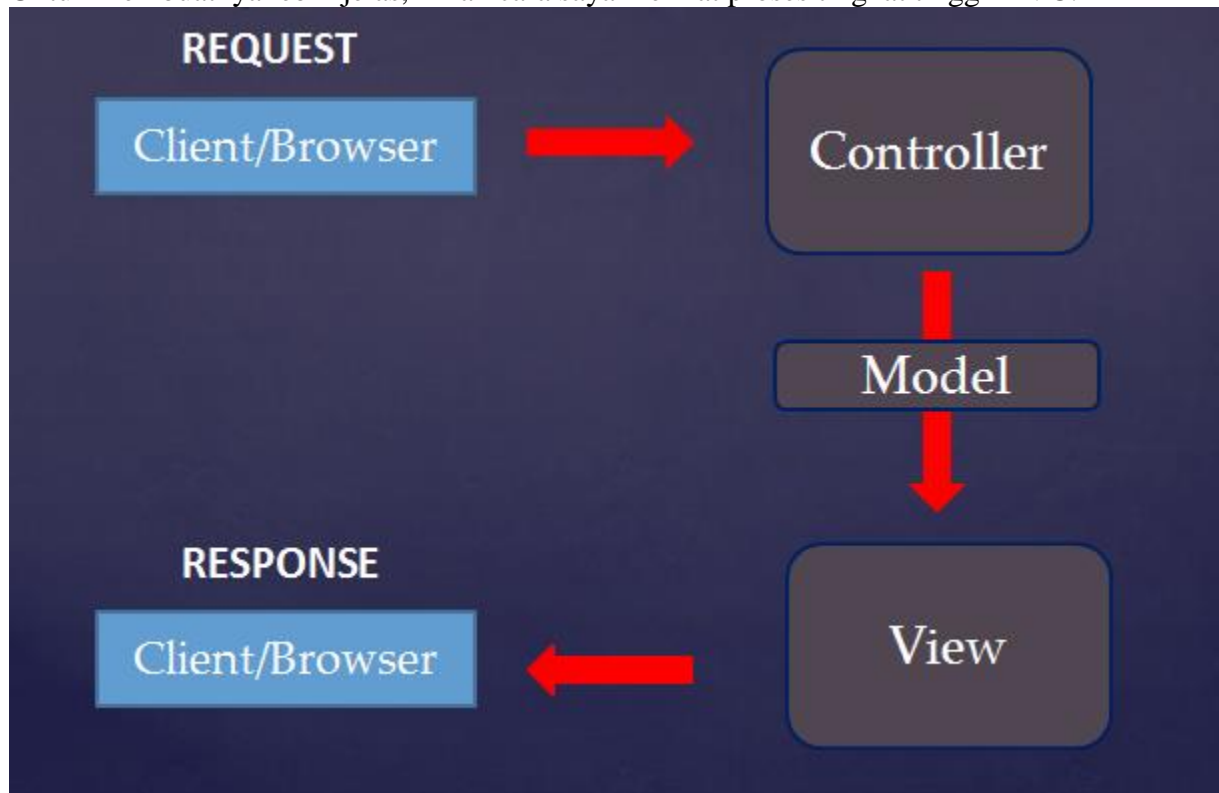
- Membuat database dari MS SQL Server
- Ikhtisar singkat ASP.NET MVC
- Implementasi Entity Framework Database
- Membuat halaman sederhana Signup

Sebelum kita mulai membangun aplikasi MVC mari kita bicara tentang sedikit tentang MVC karena sangat penting untuk mengetahui bagaimana kerangka kerja MVC.

## **Apa MVC?**

ASP.NET MVC memberikan Anda cara yang powerfull dan berpola, untuk membangun website dinamis, yang memungkinkan pemisahan bersih perhatian dan yang memberikan Anda kontrol penuh atas perubahan untuk pengembangan menyenangkan dan lincah.

Untuk membuatnya lebih jelas, inilah cara saya melihat proses tingkat tinggi MVC:



Tidak seperti ASP.NET WebForms di mana permintaan akan langsung ke file halaman (ASPX), di MVC ketika permintaan pengguna halaman, pertama kali akan berbicara dengan Controller, proses data saat diperlukan dan kembali melihat Model lalu View untuk dilihat pengguna.

#### Models

Model adalah bagian dari aplikasi yang menerapkan logika untuk data aplikasi. Seringkali, Model dijadikan model pernyataan(kode model) untuk pengambilan dan penyimpanan dalam database.

#### Controllers

Controllers adalah komponen yang menangani interaksi dengan pengguna, bekerja dengan model, dan memilih view untuk ditampilkan pada browser.

#### Views

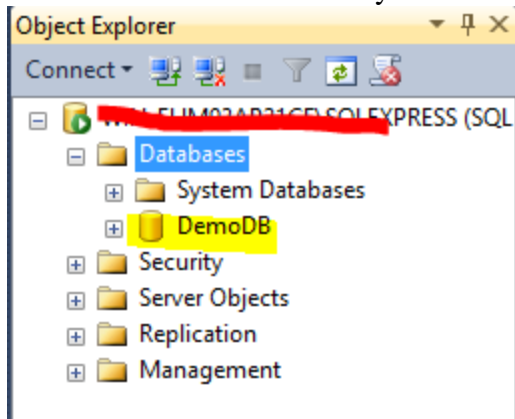
Views adalah komponen yang menampilkan aplikasi User Interface (UI), biasanya UI dibuat dari data model.

### LANGKAH 1: Membuat Database

Buka SQL Server Express Management Studio dan buat database berdasarkan petunjuk berikut ini:

- Klik kanan pada folder Databases
- Pilih New Database
- Masukkan nama database lalu klik OK. Note pada contoh ini digunakan nama "DemoDB" untuk database-nya.

Database DemoDB seharusnya terbentuk seperti gambar dibawah:



Note untuk membuat table dapat menggunakan kode seperti dibawah ini, atau dapat menggunakan GUI.

Buka New Query window atau tekan CTRL + N untuk menjalankan query window lalu run kode berikut:

LOOKUPRole table

```
USE [DemoDB]
GO

CREATE TABLE [dbo].[LOOKUPRole](
    [LOOKUPRoleID] [int] IDENTITY(1,1) NOT NULL,
    [RoleName] [varchar](100) DEFAULT '',
    [RoleDescription] [varchar](500) DEFAULT '',
    [RowCreatedSYSUserID] [int] NOT NULL,
    [RowCreatedDateTime] [datetime] DEFAULT GETDATE(),
    [RowModifiedSYSUserID] [int] NOT NULL,
    [RowModifiedDateTime] [datetime] DEFAULT GETDATE(),
    PRIMARY KEY (LOOKUPRoleID)
```

```
)  
GO
```

Menambahkan contoh data untuk table LOOKUPRole

```
INSERT INTO LOOKUPRole  
(RoleName,RoleDescription,RowCreatedSYSUserID,RowModifiedSYSUserID)  
VALUES ('Admin','Can Edit, Update, Delete',1,1)  
  
INSERT INTO LOOKUPRole  
(RoleName,RoleDescription,RowCreatedSYSUserID,RowModifiedSYSUserID)  
VALUES ('Member','Read only',1,1)
```

SYSUser table

```
USE [DemoDB]  
GO  
  
CREATE TABLE [dbo].[SYSUser](  
    [SYSUserID] [int] IDENTITY(1,1) NOT NULL,  
    [LoginName] [varchar](50) NOT NULL,  
    [PasswordEncryptedText] [varchar](200) NOT NULL,  
    [RowCreatedSYSUserID] [int] NOT NULL,  
    [RowCreatedDateTime] [datetime] DEFAULT GETDATE(),  
    [RowModifiedSYSUserID] [int] NOT NULL,  
    [RowModifiedDateTime] [datetime] DEFAULT GETDATE(),  
    PRIMARY KEY (SYSUserID)  
)  
  
GO
```

SYSUserProfile table

```
USE [DemoDB]  
GO
```

```
CREATE TABLE [dbo].[SYSUserProfile](
```

```

    [SYSUserProfileID] [int] IDENTITY(1,1) NOT NULL,
    [SYSUserID] [int] NOT NULL,
    [FirstName] [varchar](50) NOT NULL,
    [LastName] [varchar](50) NOT NULL,
    [Gender] [char](1) NOT NULL,
    [RowCreatedSYSUserID] [int] NOT NULL,
    [RowCreatedDateTime] [datetime] DEFAULT GETDATE(),
    [RowModifiedSYSUserID] [int] NOT NULL,
    [RowModifiedDateTime] [datetime] DEFAULT GETDATE(),
    PRIMARY KEY (SYSUserProfileID)
)
GO

ALTER TABLE [dbo].[SYSUserProfile] WITH CHECK ADD FOREIGN KEY([SYSUserID])
REFERENCES [dbo].[SYSUser] ([SYSUserID])
GO

```

#### SYSUserRole table

```

USE [DemoDB]
GO

CREATE TABLE [dbo].[SYSUserRole](
    [SYSUserRoleID] [int] IDENTITY(1,1) NOT NULL,
    [SYSUserID] [int] NOT NULL,
    [LOOKUPRoleID] [int] NOT NULL,
    [IsActive] [bit] DEFAULT (1),
    [RowCreatedSYSUserID] [int] NOT NULL,
    [RowCreatedDateTime] [datetime] DEFAULT GETDATE(),
    [RowModifiedSYSUserID] [int] NOT NULL,
    [RowModifiedDateTime] [datetime] DEFAULT GETDATE(),
    PRIMARY KEY (SYSUserRoleID)
)
GO

ALTER TABLE [dbo].[SYSUserRole] WITH CHECK ADD FOREIGN KEY([LOOKUPRoleID])

```

```
REFERENCES [dbo].[LOOKUPRole] ([LOOKUPRoleID])
```

```
GO
```

```
ALTER TABLE [dbo].[SYSUserRole] WITH CHECK ADD FOREIGN KEY([SYSUserID])
```

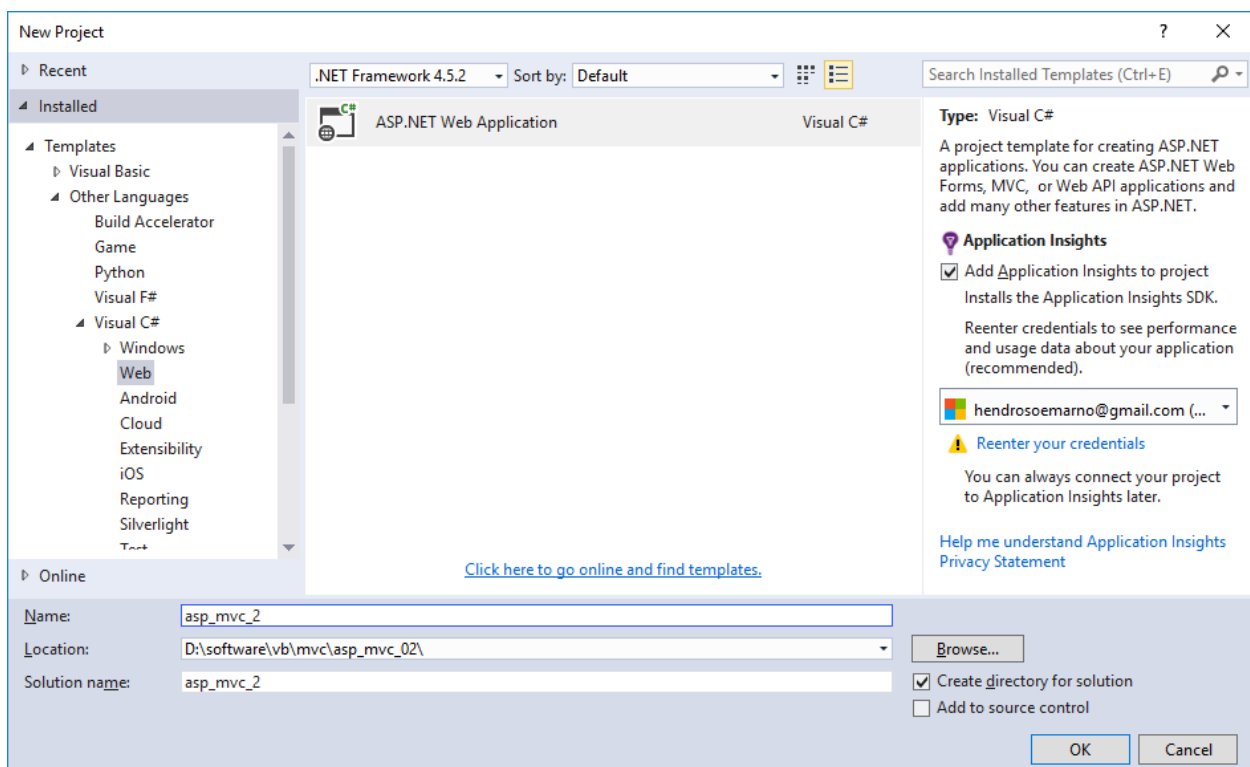
```
REFERENCES [dbo].[SYSUser] ([SYSUserID])
```

```
GO
```

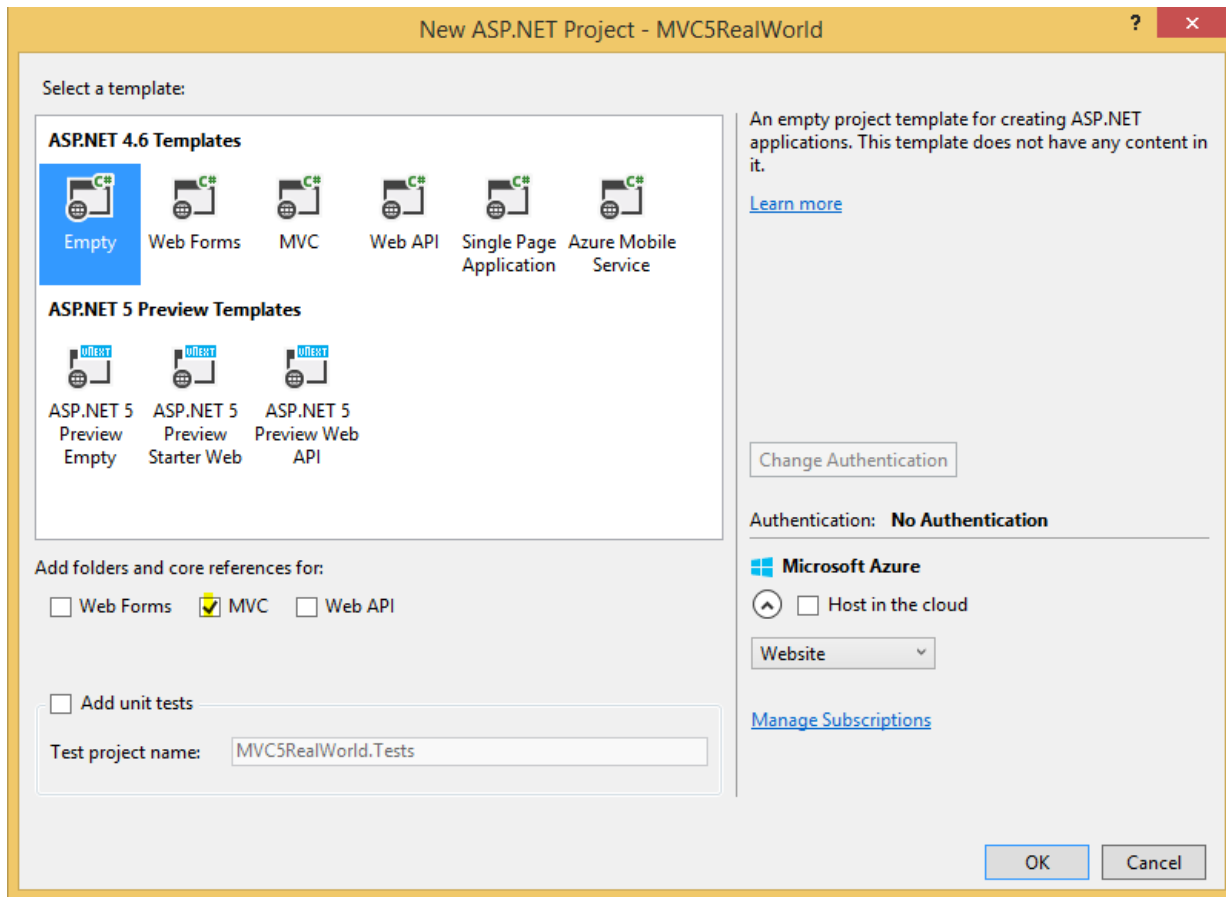
Kita baru saja membuat 4 tabel database. LANGKAH selanjutnya membuat aplikasi web.

## LANGKAH 2: Menambah project baru ASP.NET MVC 5

Jalankan Visual Studio 2015 dan pilih File > New > Project. Under New Project dialog select Templates > Visual C# > ASP.NET Web Application. Lihat gambar dibawah ini, untuk lebih jelasnya:



Beri nama pada project dan klik OK. Dalam kasus ini, nama project-nya “asp\_mvc\_2”. Setelah itu akan tampil “New ASP.NET Project” dialog seperti dibawah ini:



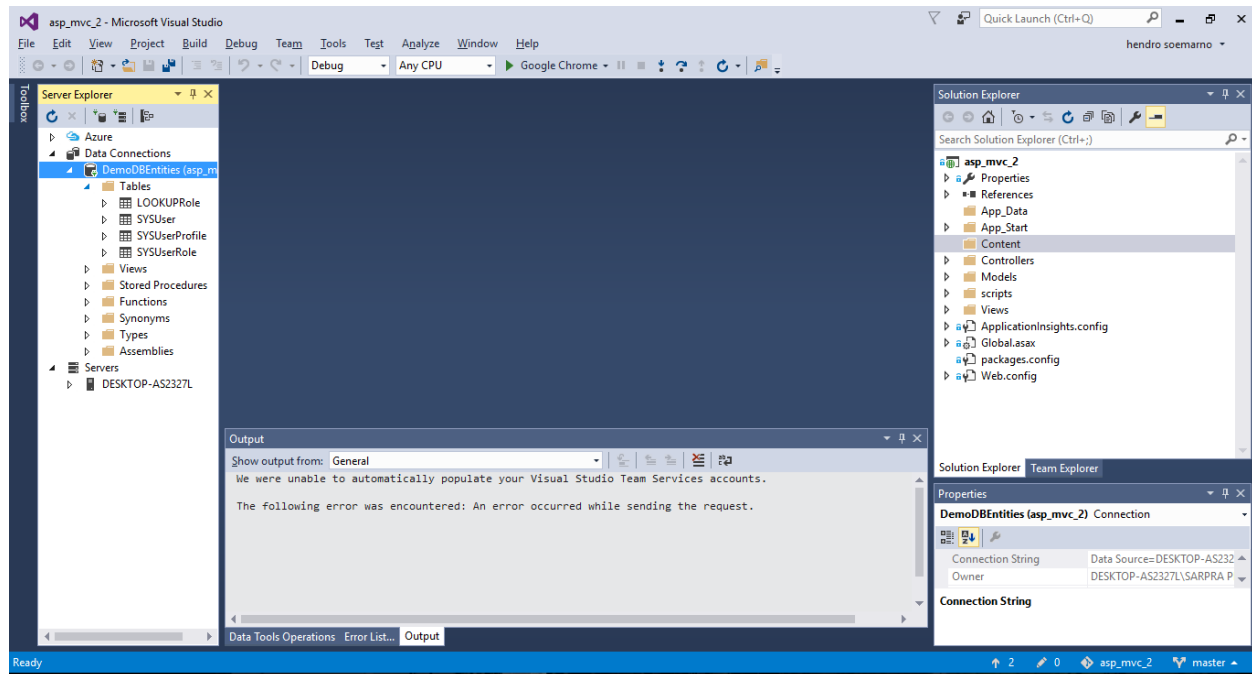
Project dialog ASP.NET yang baru untuk ASP.NET 4.6 templates memungkinkan untuk memilih tipe dari project yang akan dibuat, konfigurasi dari kombinasi teknologi seperti WebForms, MVC or Web API, mengkonfigurasi unit test project, mengkonfigurasi pilihan authentication dan juga menawarkan pilihan baru untuk hosting website di Azure cloud. Menambahkan itu juga akan mendapatkan templates ASP.NET 5.

*Note: Disini akan menggunakan aplikasi MVC 5. Sehingga detail dari setiap konfigurasi seperti unit testing, authentication, hosting in cloud, etc. tidak dibahas.*

Pilih "Empty" di ASP.NET 4.6 templates dan cek opsi MVC dan core reference seperti gambar diatas. Click OK dan tunggu Visual Studio membuat file dan template yang diperlukan untuk menjalankan MVC.

Terlihat seperti gambar:





### LANGKAH 3: Setting Data Access

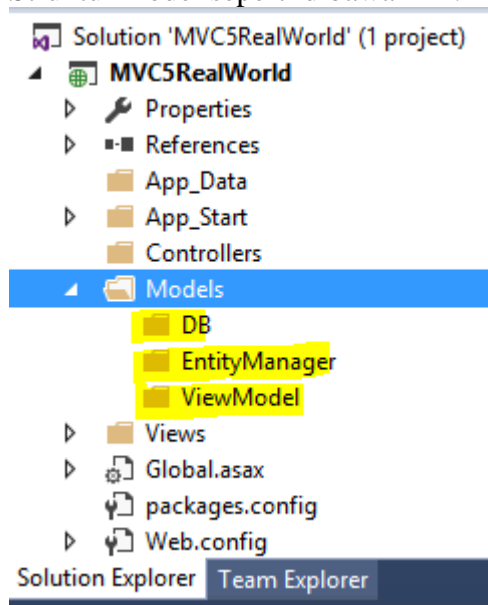
Modul ini menggunakan Entity Framework 6 (EF) untuk berhubungan dengan database.

Model hanyalah class. Class yang mengimplementasikan logika untuk data aplikasi. Sering, objek model adalah menyimpan dan membuka database.

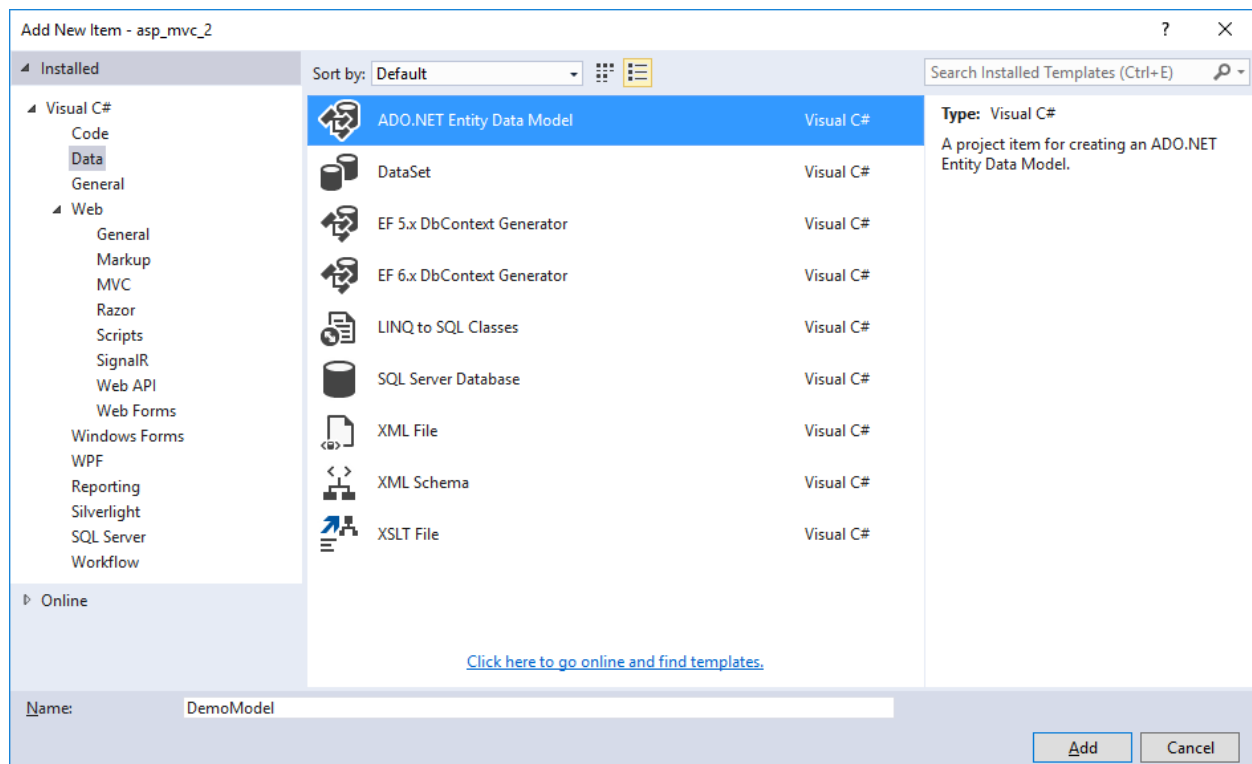
Atur struktur folder Model dengan menambahkan sub-folder berikut di dalam folder Models:

- DB
- EntityManager
- ViewModel

Struktur model seperti dibawah ini:



Folder "DB" dimana kita menyimpan entity models (.EDMX file). Untuk menambah entity, klik kanan folder DB dan pilih Add > New Item > Data > ADO.NET Entity Data Model. Seperti gambar:



Nama dari model bisa apa saja, dalam contoh ini menggunakan "DemoModel". Klik Add untuk melanjutkan dan pada langkah selanjutnya pilih "EF Designer from Database". Click Next untuk memproses. LANGKAH selanjutnya click tombol "New Connection" dan pilih "Microsoft SQL Server (SqlClient)" sebagai data source lalu click Next. Seperti pada gambar:

Connection Properties

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) Change...

Server name:  
WIN-EHM93AP21CF\SQLEXPRESS Refresh

Log on to the server

☒ Use Windows Authentication  
☐ Use SQL Server Authentication

User name: WIN-EHM93AP21CF\ProudMonkey  
Password: .....  
☐ Save my password

Connect to a database

☒ Select or enter a database name:  
DemoDB  
☐ Attach a database file:  
Browse...

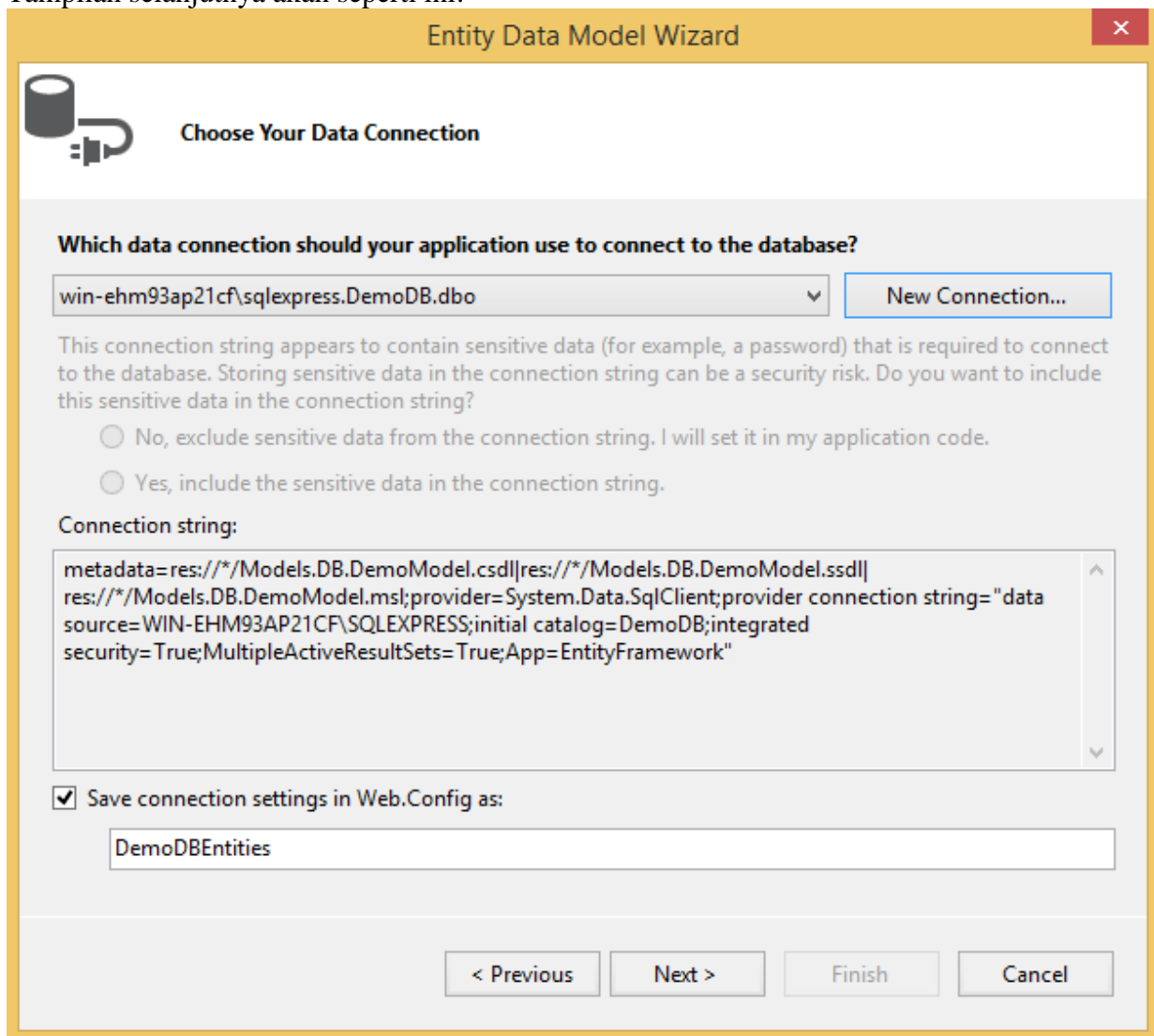
Logical name:

Advanced...

Test Connection OK Cancel

Nama SQL server dan pilih database yang telah dibuat pada LANGKAH 1. Jika sudah memiliki database sebelumnya, maka database itu dapat digunakan. Dan menggunakan windows authentication untuk login ke SQL Server. Ketika selesai klik "Test Connection" untuk memverifikasi koneksi dengan database. Jika sukses lanjutkan dengan klik OK.

Tampilan selanjutnya akan seperti ini:

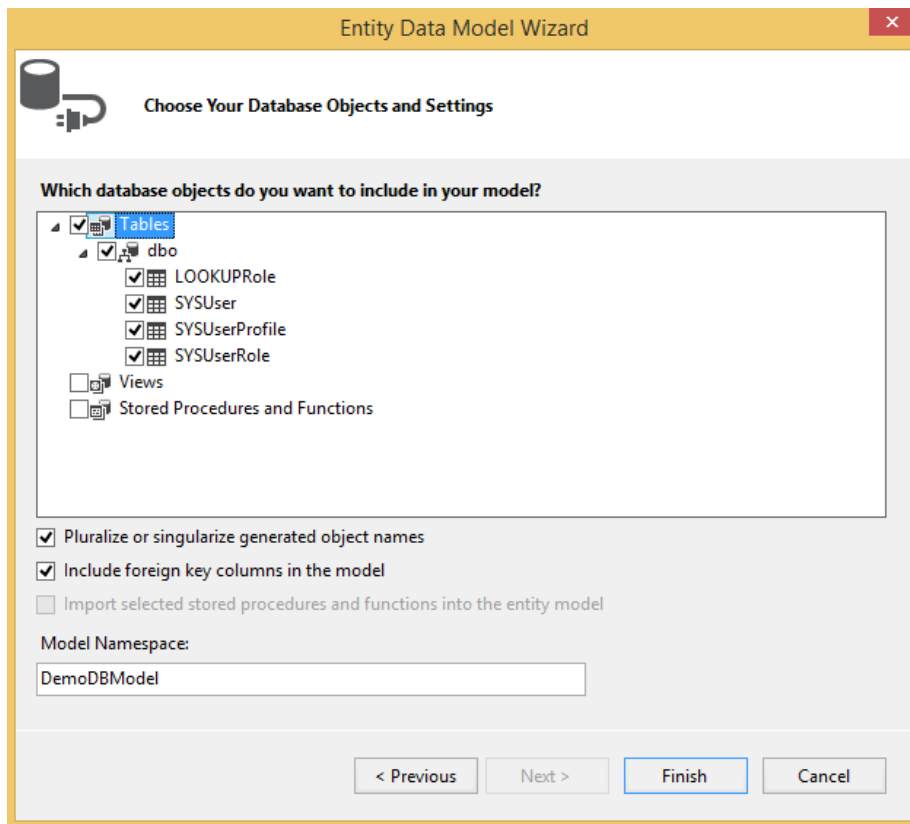


The image shows the 'Entity Data Model Wizard' window, specifically the 'Choose Your Data Connection' step. The window has a yellow title bar with the text 'Entity Data Model Wizard' and a close button. Below the title bar is a header area with a database icon and the text 'Choose Your Data Connection'. The main content area is light gray and contains the following elements:

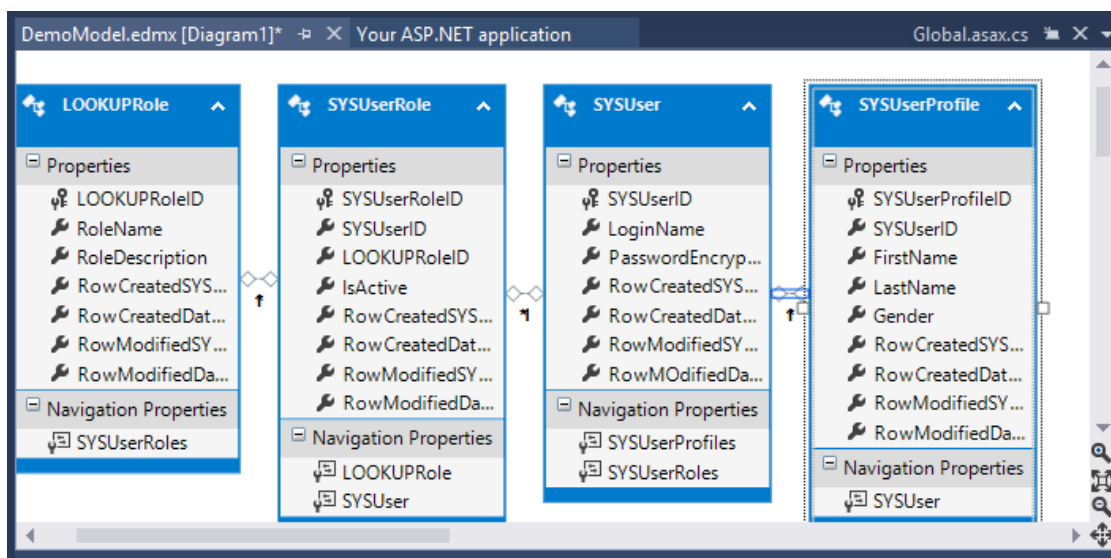
- A question: 'Which data connection should your application use to connect to the database?'
- A dropdown menu showing 'win-ehm93ap21cf\sqlexpress.DemoDB.dbo' with a downward arrow.
- A button labeled 'New Connection...'.
- A warning text: 'This connection string appears to contain sensitive data (for 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?'
- Two radio buttons:
  - ☐ 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.
- A label 'Connection string:' followed by a text box containing the connection string:

```
metadata=res://*/Models.DB.DemoModel.csdl|res://*/Models.DB.DemoModel.ssdl|
res://*/Models.DB.DemoModel.msl;provider=System.Data.SqlClient;provider connection string="data
source=WIN-EHM93AP21CF\SQLSERVER;initial catalog=DemoDB;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
```
- A checkbox labeled 'Save connection settings in Web.Config as:' which is checked.
- A text box containing the name 'DemoDBEntities'.
- Four buttons at the bottom: '< Previous', 'Next >', 'Finish', and 'Cancel'.

Catatan, bahwa connection string otomatis terbentuk. Click next dan pilih Entity Framework 6.x, dan kemudian akan tampil tampilan sebagai berikut:



Pilih table yang akan digunakan di aplikasi. Klik tombol Finish akan membuat entity model, seperti pada gambar:



Yang terjadi adalah EF membentuk otomatis objek kebutuhan dan hanya dibutuhkan penambahan query. EDMX atau entity data model akan melayani gateway utama untuk mengubah objek database and membuat perubahan.

## LANGKAH 4: Membuat Halaman Signup

### Menambah ViewModels

Entity Framework akan menghasilkan objek model dan mengatur akses data dengan aplikasi. Hasilnya, class LOOKUPRole, SYSUserRole, SYSUser dan SYSUserProfile otomatis dibuat oleh EF dan fitur semua kolom dari table database sebagai properti class.

Tidak diinginkan menggunakan semua class langsung ke View sehingga hanya dibuat sebagian class yang property nya dibutuhkan di View. Tambahkan "UserModel" class dengan klik kanan pada folder "ViewModel" lalu pilih Add > Class. File "UserModel.cs" adalah tempat semua keperluan user yang berhubungan dengan model views. Untuk halaman Signup kita akan menambahkan "UserSignUpView" class property sebagai berikut:

```
using System.ComponentModel.DataAnnotations;

namespace Asp_mvc_2.Models.ViewModel
{
    public class UserSignUpView
    {
        [Key]
        public int SYSUserID { get; set; }
        public int LOOKUPRoleID { get; set; }
        public string RoleName { get; set; }
        [Required(ErrorMessage = "**")]
        [Display(Name = "Login ID")]
        public string LoginName { get; set; }
        [Required(ErrorMessage = "**")]
        [Display(Name = "Password")]
        public string Password { get; set; }
        [Required(ErrorMessage = "**")]
        [Display(Name = "First Name")]
        public string FirstName { get; set; }
        [Required(ErrorMessage = "**")]
        [Display(Name = "Last Name")]
    }
}
```

```

        public string LastName { get; set; }
        public string Gender { get; set; }
    }
}

```

Catatan, ditambahkan atribut Required dan DisplayName untuk setiap property dalam UserSignUpView class. Attribute ini dinamakan Data Annotations. Data annotations adalah attribute class yang berjalan dibawah *System.ComponentModel.DataAnnotations* yang dapat digunakan untuk mendekorasi class atau properti untuk keperluan validation.

Digunakannya teknik validasi untuk memisahkan konsentrasi ketika menggunakan MVC dan data annotations dalam model, lalu kode validasi lebih sederhana, maintain, dan test.

Menambah UserManager Class

LANGKAH selanjutnya adalah membuat "UserManger" class yang menangani CRUD.

Klik kanan pada folder "EntityManager" dan tambah class baru dengan memilih Add > Class nama class adalah "UserManager". Ini adalah kode "UserManager.cs":

```

using System;
using System.Linq;
using Asp_mvc_2.Models.DB;
using Asp_mvc_2.Models.ViewModel;

namespace Asp_mvc_2.Models.EntityManager
{
    public class UserManager
    {
        public void AddUserAccount(UserSignUpView user) {

            using (DemoDBEntities db = new DemoDBEntities()) {

                SYSUser SU = new SYSUser();
                SU.LoginName = user.LoginName;
                SU.PasswordEncryptedText = user.Password;
                SU.RowCreatedSYSUserID = user.SYSUserID > 0 ? user.SYSUserID : 1;
            }
        }
    }
}

```

```

SU.RowModifiedSYSUserID = user.SYSUserID > 0 ? user.SYSUserID : 1; ;
SU.RowCreatedDateTime = DateTime.Now;
SU.RowModifiedDateTime = DateTime.Now;

db.SYSUsers.Add(SU);
db.SaveChanges();

SYSUserProfile SUP = new SYSUserProfile();
SUP.SYSUserID = SU.SYSUserID;
SUP.FirstName = user.FirstName;
SUP.LastName = user.LastName;
SUP.Gender = user.Gender;
SUP.RowCreatedSYSUserID = user.SYSUserID > 0 ? user.SYSUserID : 1;
SUP.RowModifiedSYSUserID = user.SYSUserID > 0 ? user.SYSUserID : 1;
SUP.RowCreatedDateTime = DateTime.Now;
SUP.RowModifiedDateTime = DateTime.Now;

db.SYSUserProfiles.Add(SUP);
db.SaveChanges();

if (user.LOOKUPRoleID > 0) {
    SYSUserRole SUR = new SYSUserRole();
    SUR.LOOKUPRoleID = user.LOOKUPRoleID;
    SUR.SYSUserID = user.SYSUserID;
    SUR.IsActive = true;
    SUR.RowCreatedSYSUserID = user.SYSUserID > 0 ? user.SYSUserID :
1;

    SUR.RowModifiedSYSUserID = user.SYSUserID > 0 ? user.SYSUserID :
1;

    SUR.RowCreatedDateTime = DateTime.Now;
    SUR.RowModifiedDateTime = DateTime.Now;

    db.SYSUserRoles.Add(SUR);
    db.SaveChanges();
}

```



```

    }
}

public bool IsLoginNameExist(string loginName) {
    using (DemoDBEntities db = new DemoDBEntities()) {
        return db.SYSUsers.Where(o => o.LoginName.Equals(loginName)).Any();
    }
}
}
}
}

```

**AddUserAccount()** adalah method yang meng inserts data ke database menggunakan Entity Framework. **IsLoginNameExist()** adalah method yang returns nya boolean. Method ini berguna untuk mengecek database untuk data yang ada menggunakan LINQ syntax.

#### Menambah Controllers

Klik kanan pada folder "Controllers" dan pilih Add > Controller > MVC 5 Controller - Empty and dan klik Add. Pada dialog berikutnya namakan controller sebagai "AccountController" dan click Add untuk generate class.

Ini adalah kode dari AccountController:

```

using System.Web.Mvc;
using System.Web.Security;
using Asp_mvc_2.Models.ViewModel;
using Asp_mvc_2.Models.EntityManager;

namespace Asp_mvc_2.Controllers
{
    public class AccountController : Controller
    {
        public ActionResult SignUp() {
            return View();
        }

        [HttpPost]
    }
}

```

```

public ActionResult SignUp(UserSignUpView USV) {
    if (ModelState.IsValid) {
        UserManager UM = new UserManager();
        if (!UM.IsLoginNameExist(USV.LoginName)) {
            UM.AddUserAccount(USV);
            FormsAuthentication.SetAuthCookie(USV.FirstName, false);
            return RedirectToAction("Welcome", "Home");
        }
        else
            ModelState.AddModelError("", "Login Name already taken.");
    }
    return View();
}
}
}

```

**AccountController** memiliki 2 method utama. Pertama "SignUp" yang memanggil "SignUp.cshtml" View. Kedua "SignUp" yang didekorasi dengan "[HttpPost]" attribute. Attribute ini mengkhususkan untuk meng overload "SignUp" method, yang dapat berubah melalui POST request saja.

Method kedua bertanggung jawab untuk meng insert entry baru ke database dan secara otomatis mengautentikasi user menggunakan **FormsAuthentication.SetAuthCookie()** method. Setelah autentikasi, lalu redirect user ke Welcome.cshtml page.

Tambahkan Controller lagi dan namakan sebagai "HomeController". Controller ini akan menjadi controller pada default page. Buat "Index" dan "Welcome" Views untuk controller ini untuk langkah selanjutnya. Ini adalah kode "HomeController" class:

```

using System.Web.Mvc;

namespace Asp_mvc_2.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index() {

```

```

        return View();
    }

    [Authorize]
    public ActionResult Welcome() {
        return View();
    }
}

```

**HomeController** class terdiri dari dua ActionResult method seperti **Index** and **Welcome**. "Index" method melayani halaman default redirect dan "Welcome" method akan menjadi halaman dimana users setelah mereka sukses terregistrasi. "[Authorize]" attribute berguna agar hanya available untuk logged-in users.

Untuk mengkonfigurasi halaman default pilih App\_Start > RouteConfig. Akan terlihat kode seperti dibawah ini:

```

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

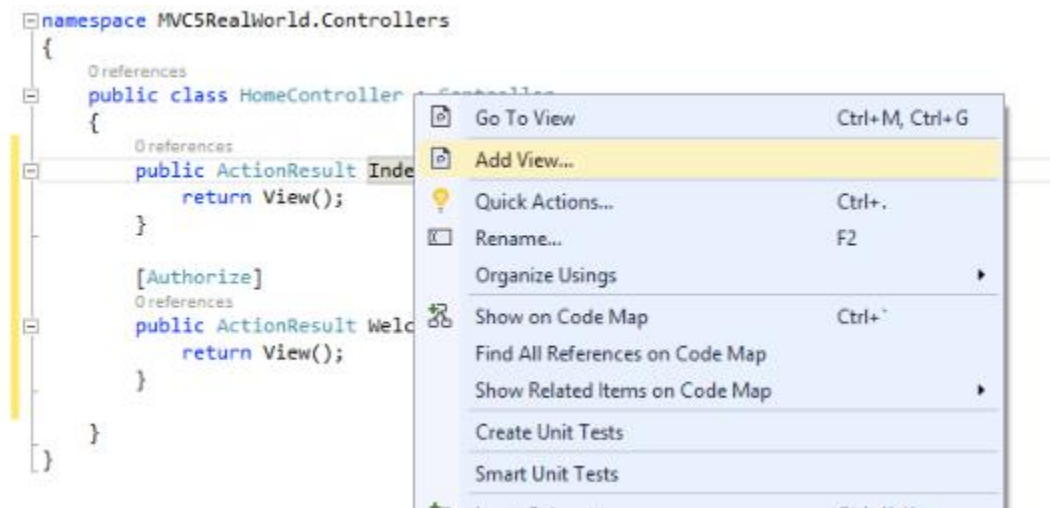
    routes.MapRoute(
        name: "Default",
        url: "{controller}/{action}/{id}",
        defaults: new { controller = "Home", action = "Index", id =
        UrlParameter.Optional }
    );
}

```

#### Menambah Views

Ada dua cara membuat Views. Membuat folder Views folder secara manual dan menambahkan koneksinya sendiri file .CSHTML atau dengan klik kanan pada Controller, seperti gambar

dibawah ini:



Setelah klik "Add View" akan tampil dialog berikut: (jika belum ada `_Layout.cshtml` tambahkan secara manual)

A screenshot of the 'Add View' dialog box in Visual Studio. The dialog has a yellow title bar and a close button. It contains several input fields and checkboxes. The 'View name' field is set to 'Index'. The 'Template' dropdown is set to 'Empty (without model)'. The 'Model class' and 'Data context class' fields are empty. Under the 'Options' section, the 'Create as a partial view' and 'Reference script libraries' checkboxes are unchecked, while the 'Use a layout page' checkbox is checked. Below this, there is a text field containing '~/\_Views/Shared/\_Layout.cshtml' and a button with three dots. A note below the text field says '(Leave empty if it is set in a Razor \_viewstart file)'. At the bottom right, there are 'Add' and 'Cancel' buttons.

Lalu klik Add. Lalu modifikasi halaman Index seperti kode dibawah ini:

```
@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

```
}

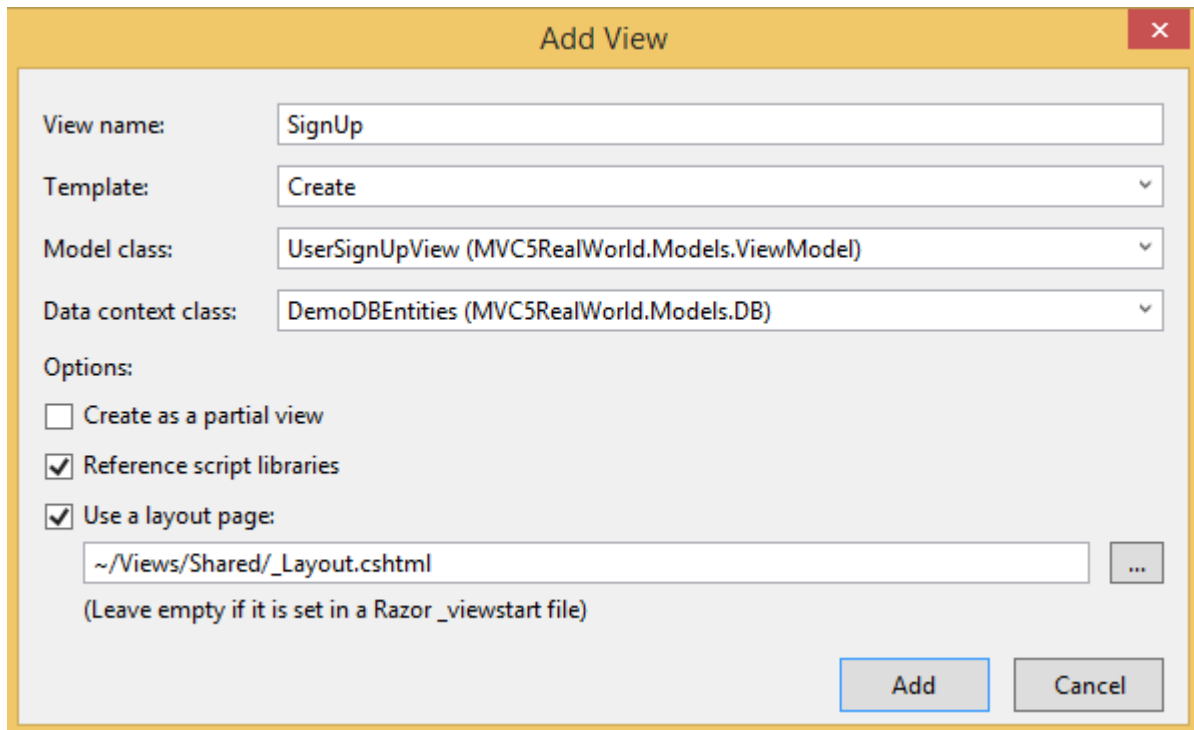
<h2>Index</h2>
<br/>
No Account yet? @Html.ActionLink("Signup Now!", "SignUp", "Account")
```

**ActionLink** memungkinkan kita menuju halaman SignUp yang berada pada Account controller. Lalu tambahkan View pada Welcome action dengan cara yang sama ketika menambah halaman Index. Ini adalah kode halaman Welcome:

```
@{
    ViewBag.Title = "Welcome";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Hi <b>@Context.User.Identity.Name</b>! Welcome to my first MVC 5 Web App!</h2>
```

Kembali pada AccountController dan tambah View untuk halaman SignUp. Dalam dialog Add View pilih “Create”, pilih UserSignUpView sebagai model dan pilih DemoDBEntities sebagai data context seperti gambar berikut:



View name: SignUp

Template: Create

Model class: UserSignUpView (MVC5RealWorld.Models.ViewModel)

Data context class: DemoDBEntities (MVC5RealWorld.Models.DB)

Options:

- ☐ Create as a partial view
- ☒ Reference script libraries
- ☒ Use a layout page:

~/Views/Shared/\_Layout.cshtml

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

Add Cancel

Klik add.

Gantilah HTML halaman SignUp seperti dibawah ini:

```
@model Asp_mvc_2.Models.ViewModel.UserSignUpView

@{
    ViewBag.Title = "SignUp";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>SignUp</h2>

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

    <div class="form-horizontal">
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.LoginName, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.LoginName, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.LoginName, "", new { @class = "text-danger" })
            </div>
        </div>

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

```

        @Html.ValidationMessageFor(model => model.Password, "", new { @class
= "text-danger" })
    </div>
</div>

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

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

<div class="form-group">
    @Html.LabelFor(model => model.Gender, htmlAttributes: new { @class =
"control-label col-md-2" })
    <div class="col-md-10">
        @Html.DropDownListFor(model => model.Gender, new List<SelectListItem>
{
            new SelectListItem { Text="Male", Value="M" },
            new SelectListItem { Text="Female", Value="F" }
        }, new { @class = "form-control" })
    </div>
</div>

```

```

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

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

```

Dengan meng includ **@model** statement pada atas file template view, kita dapat secara spesifik memilih tipe model yang view diinginkan. Dalam kasus ini menggunakan **Asp\_mvc\_2.Models.ViewModel.UserSignUpView**.

Visual Studio secara otomatis menyusun folders:



## LANGKAH 5: Web.Config

Yang terakhir adalah menambahkan konfigurasi berikut dalam system.web untuk meng enable forms autentikasi:

```

<system.web>
    <authentication mode="Forms">
        <forms loginUrl="~/Account/Login" defaultUrl="~/Home/Welcome">
        </forms>
    </authentication>
</system.web>

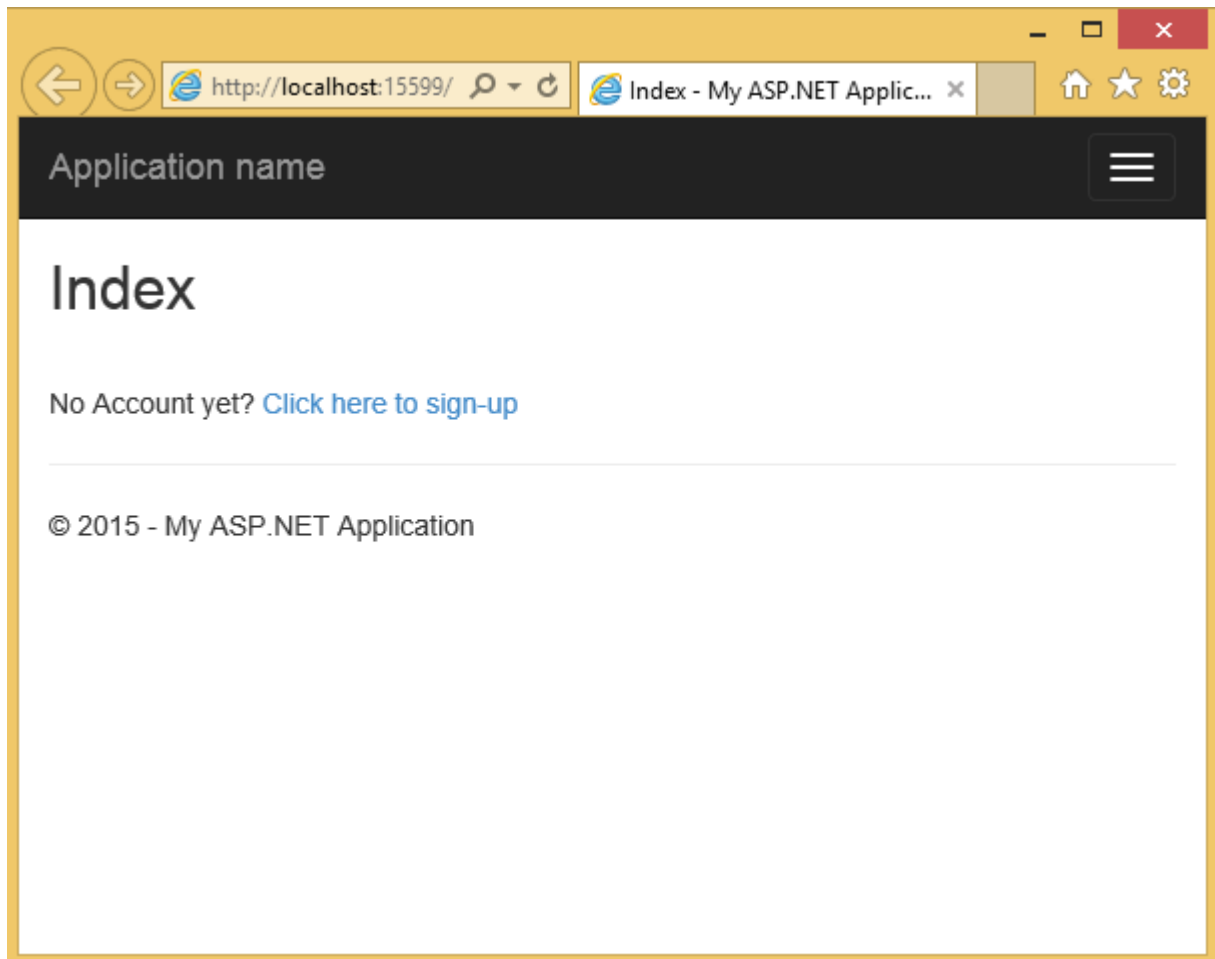
```

## LANGKAH 6: Jalankan Aplikasi

Gambar berikut adalah tampilah pada browser (jika belum sama tambahkan folder content beserta isi didalamnya):



Ketika Keluar pertama kali



Upon page validation

Application name

# SignUp

**Login ID**

**Password**

**First Name**

**Last Name**

**Gender**

Male


[Back to Main](#)


© 2015 - My ASP.NET Application


Supplying the required fields


←

→

 http://localhost:15599/







SignUp - My ASP.NET Appli... x

Application name

# SignUp

**Login ID**

**Password**

**First Name**

**Last Name**

**Gender**

Male

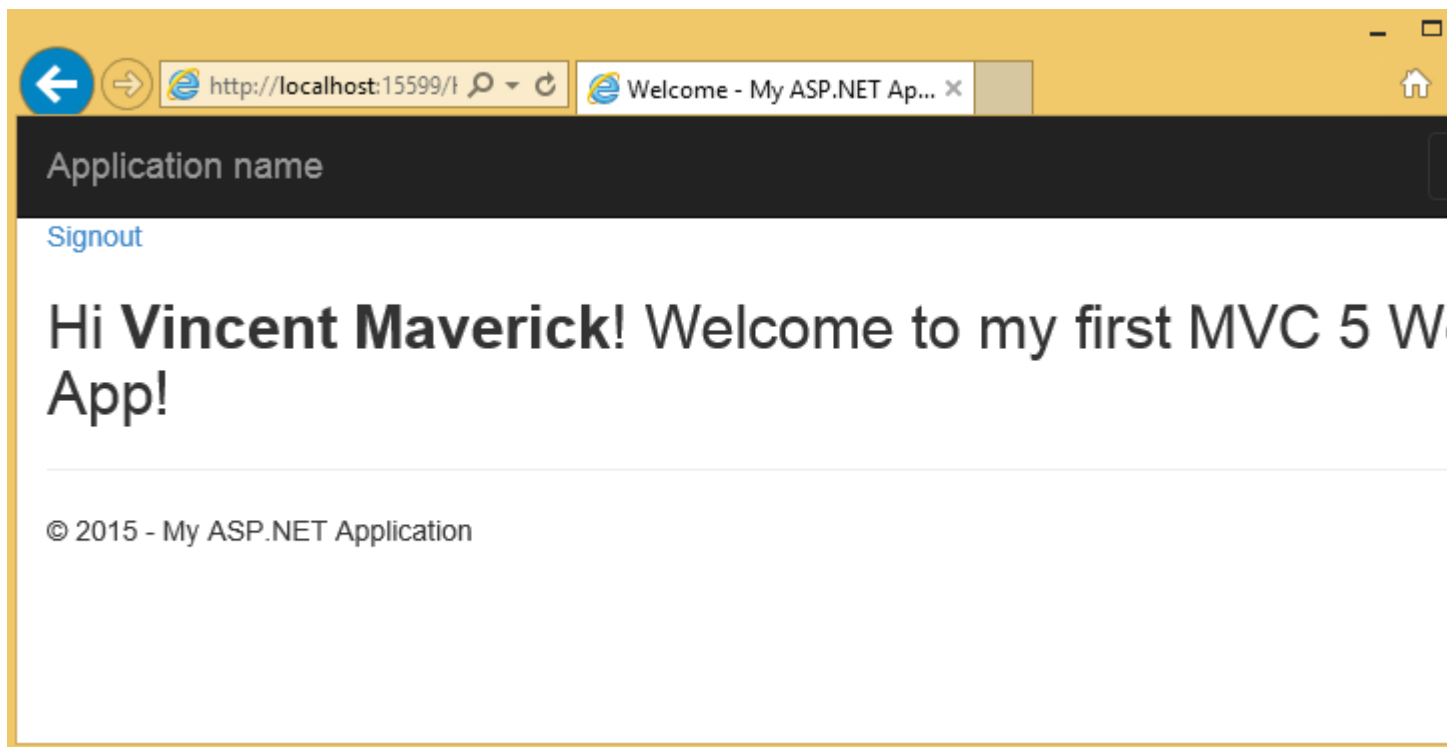
▼

Register

[Back to Main](#)

© 2015 - My ASP.NET Application

After successful registration



That's it! I hope you will find this article useful.