Intermediate Web

Fa 12018

MAY 11

COMPANY NAME

Authored by: Albert Michael Ludick

Logo Name

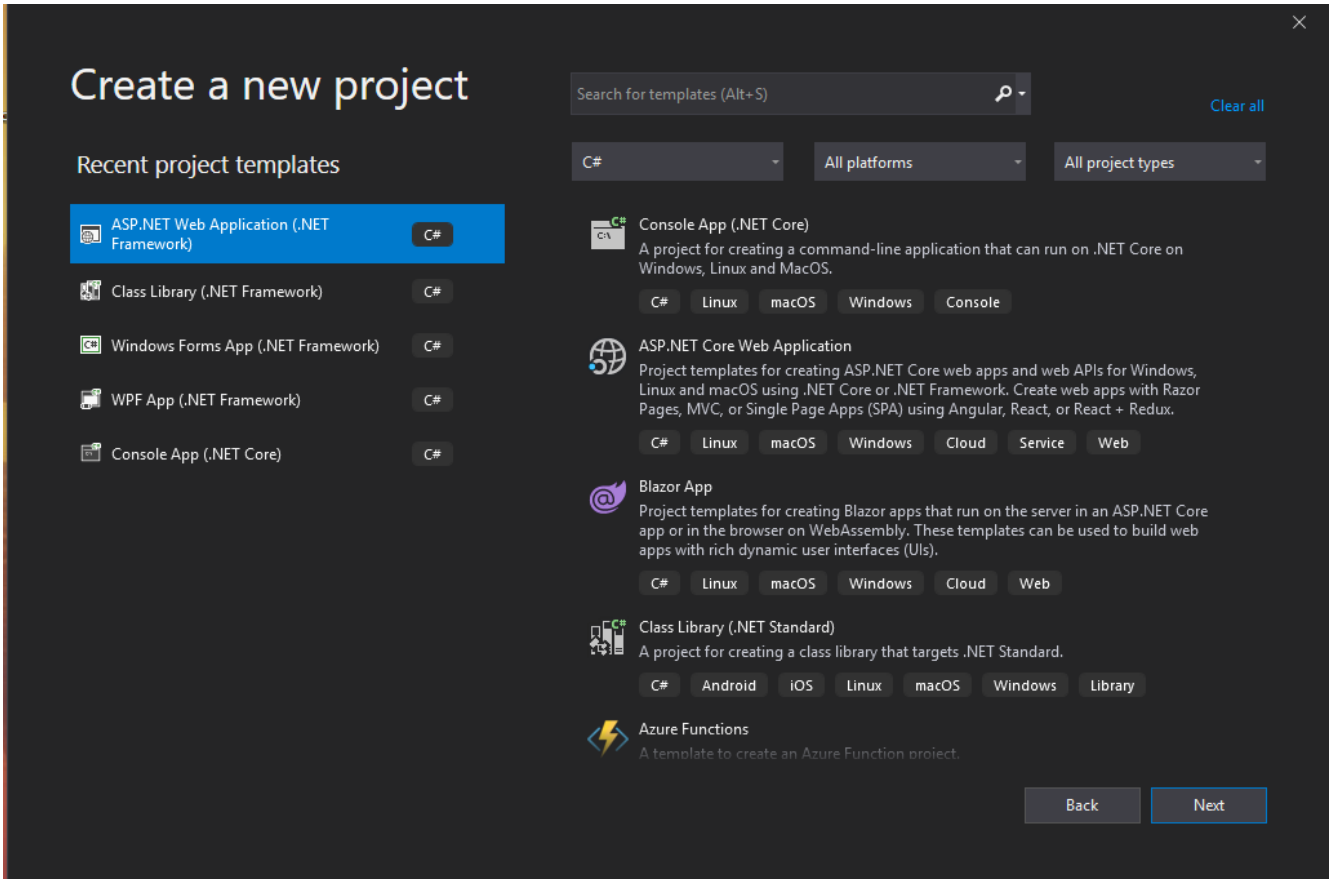
Contents

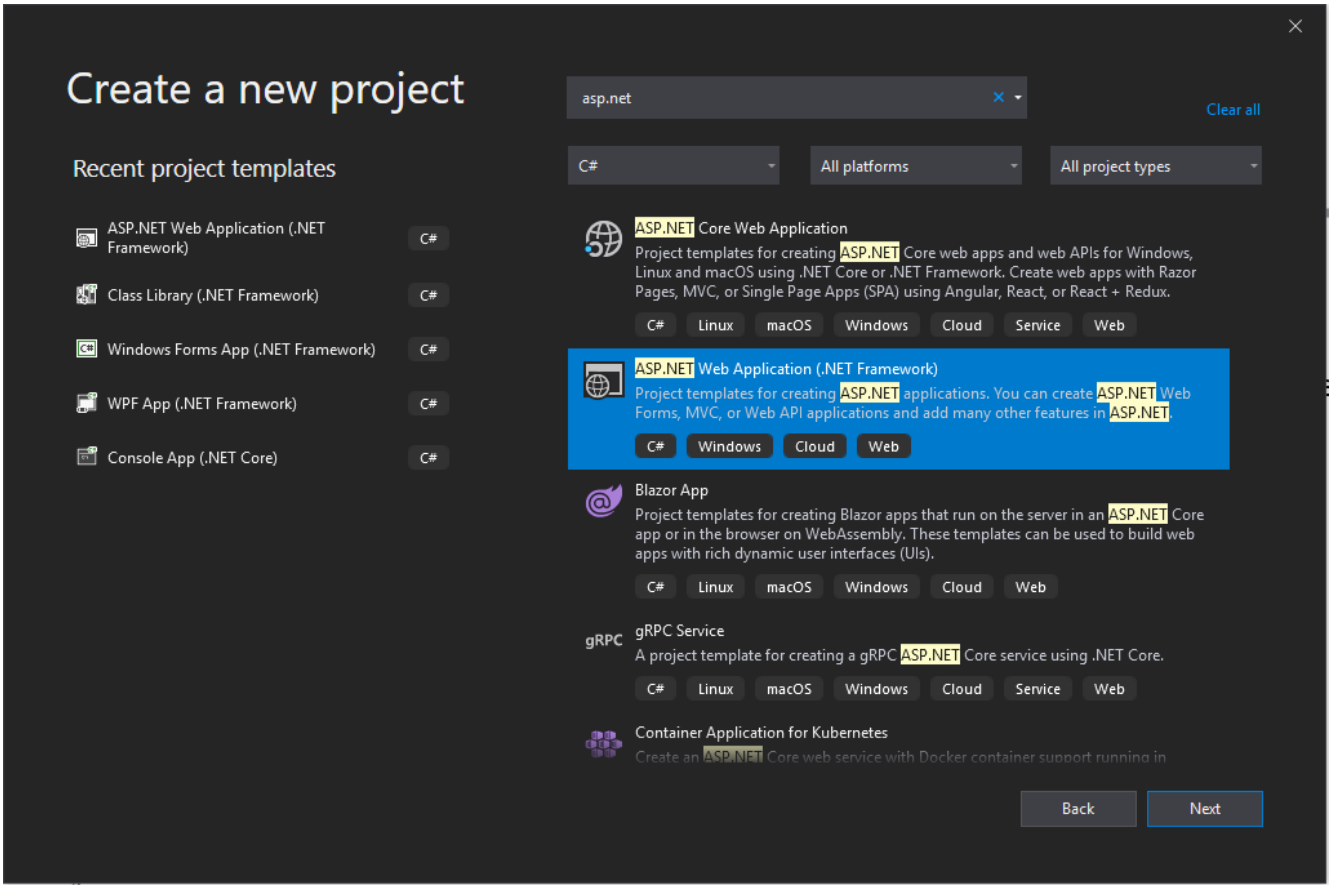
Introduction

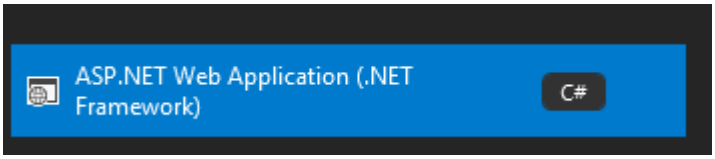
We were ask to make a web site that has a administrator role.Then we needed to add a place where he could create user and manage user roles. And then he could make event catagories and then he can also make his own events. Other user should be able only to make their own events but add event catagories or add user roles or edit user

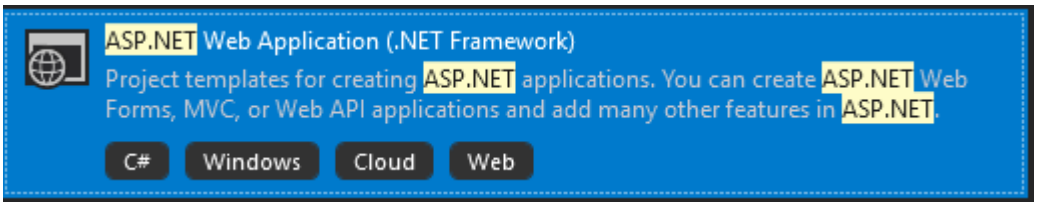
And all user should be able to view the map.the map can contain the user event and show where on the map the event is

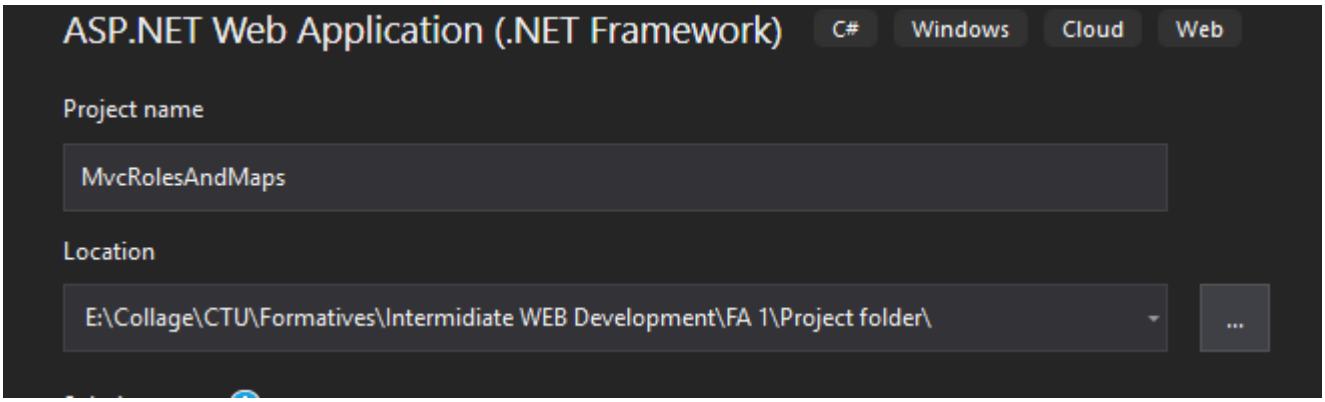
Content:

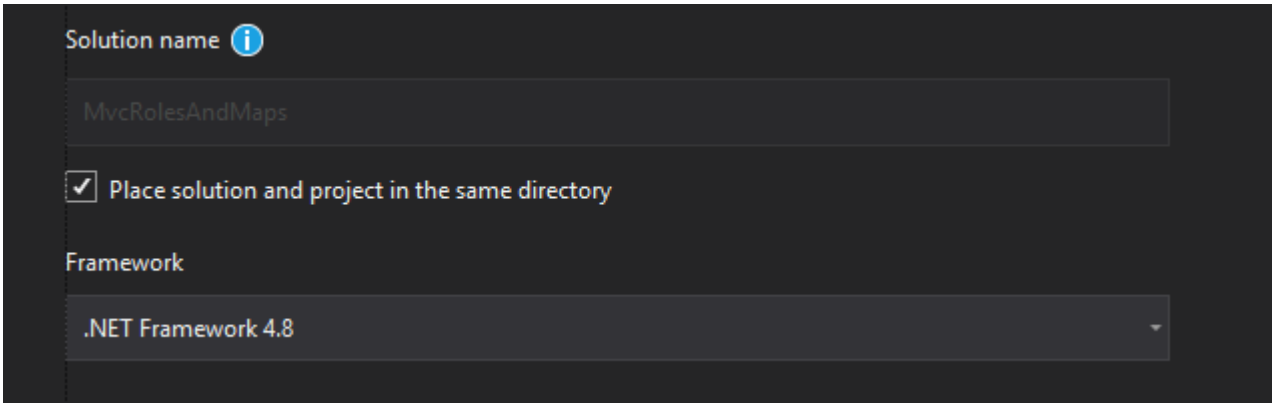

Create a new project
Search for templates (Alt+S)
Clear all
Recent project templates
C#
All platforms
All project types
ASP.NET Web Application (.NET Framework)
C#
Class Library (.NET Framework)
C#
Windows Forms App (.NET Framework)
C#
WPF App (.NET Framework)
C#
Console App (.NET Core)
C#
Console App (.NET Core)
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
C# Linux macOS Windows Console
ASP.NET Core Web Application
Project templates for creating ASP.NET Core web apps and web APIs for Windows, Linux and macOS using .NET Core or .NET Framework. Create web apps with Razor Pages, MVC, or Single Page Apps (SPA) using Angular, React, or React + Redux.
C# Linux macOS Windows Cloud Service Web
Blazor App
Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly. These templates can be used to build web apps with rich dynamic user interfaces (UIs).
C# Linux macOS Windows Cloud Web
Class Library (.NET Standard)
A project for creating a class library that targets .NET Standard.
C# Android iOS Linux macOS Windows Library
Azure Functions
A template to create an Azure Function project.
Back
Next

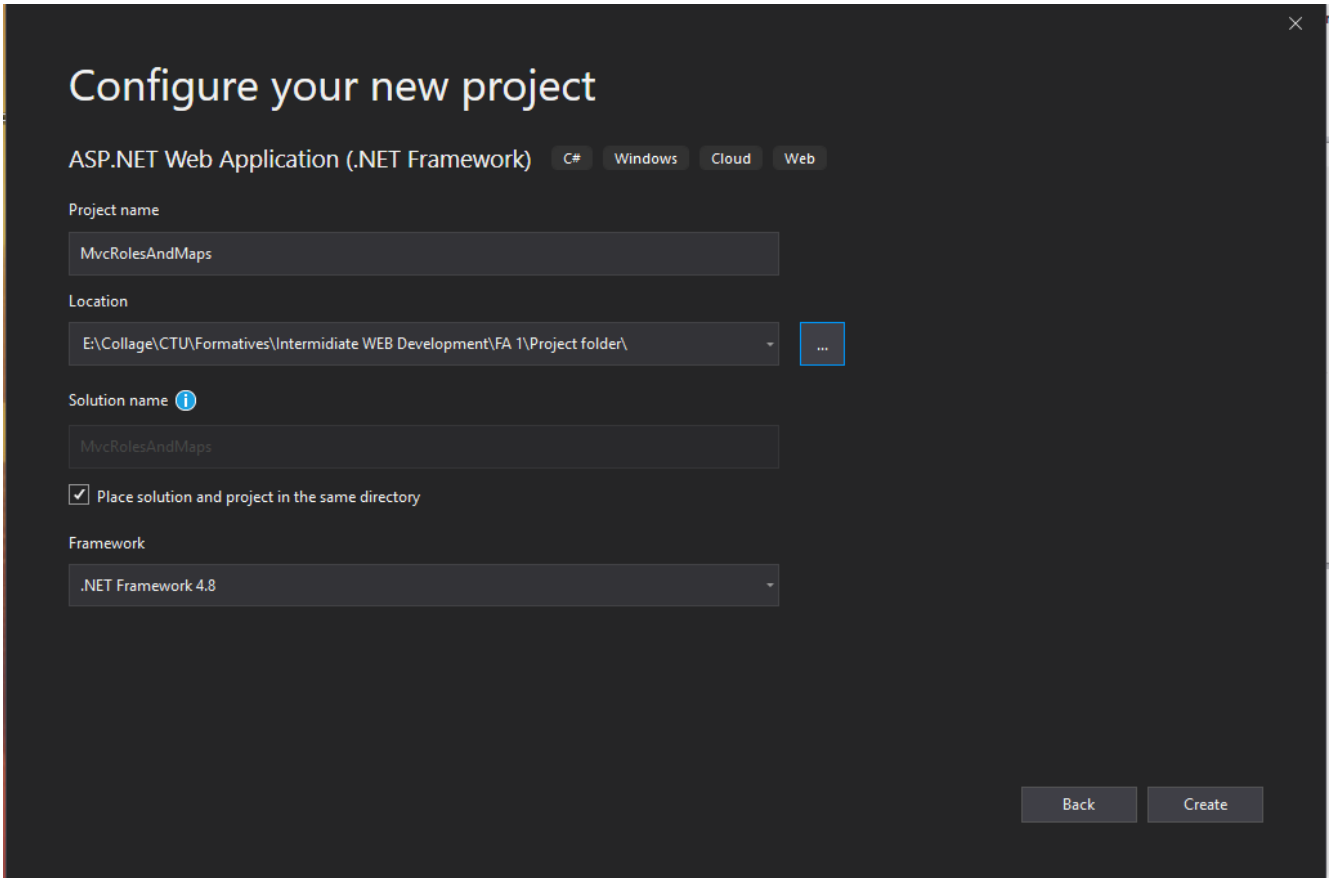

Create a new project
asp.net
Clear all
Recent project templates
C#
All platforms
All project types
ASP.NET Web Application (.NET Framework)
C#
Class Library (.NET Framework)
C#
Windows Forms App (.NET Framework)
C#
WPF App (.NET Framework)
C#
Console App (.NET Core)
C#
ASP.NET Core Web Application
Project templates for creating ASP.NET Core web apps and web APIs for Windows, Linux and macOS using .NET Core or .NET Framework. Create web apps with Razor Pages, MVC, or Single Page Apps (SPA) using Angular, React, or React + Redux.
C# Linux macOS Windows Cloud Service Web
ASP.NET Web Application (.NET Framework)
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.
C# Windows Cloud Web
Blazor App
Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly. These templates can be used to build web apps with rich dynamic user interfaces (UIs).
C# Linux macOS Windows Cloud Web
gRPC
gRPC Service
A project template for creating a gRPC ASP.NET Core service using .NET Core.
C# Linux macOS Windows Cloud Service Web
Container Application for Kubernetes
Back
Next

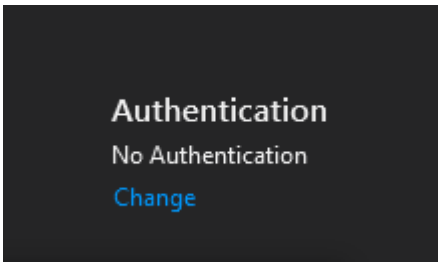

ASP.NET Web Application (.NET Framework)
C#

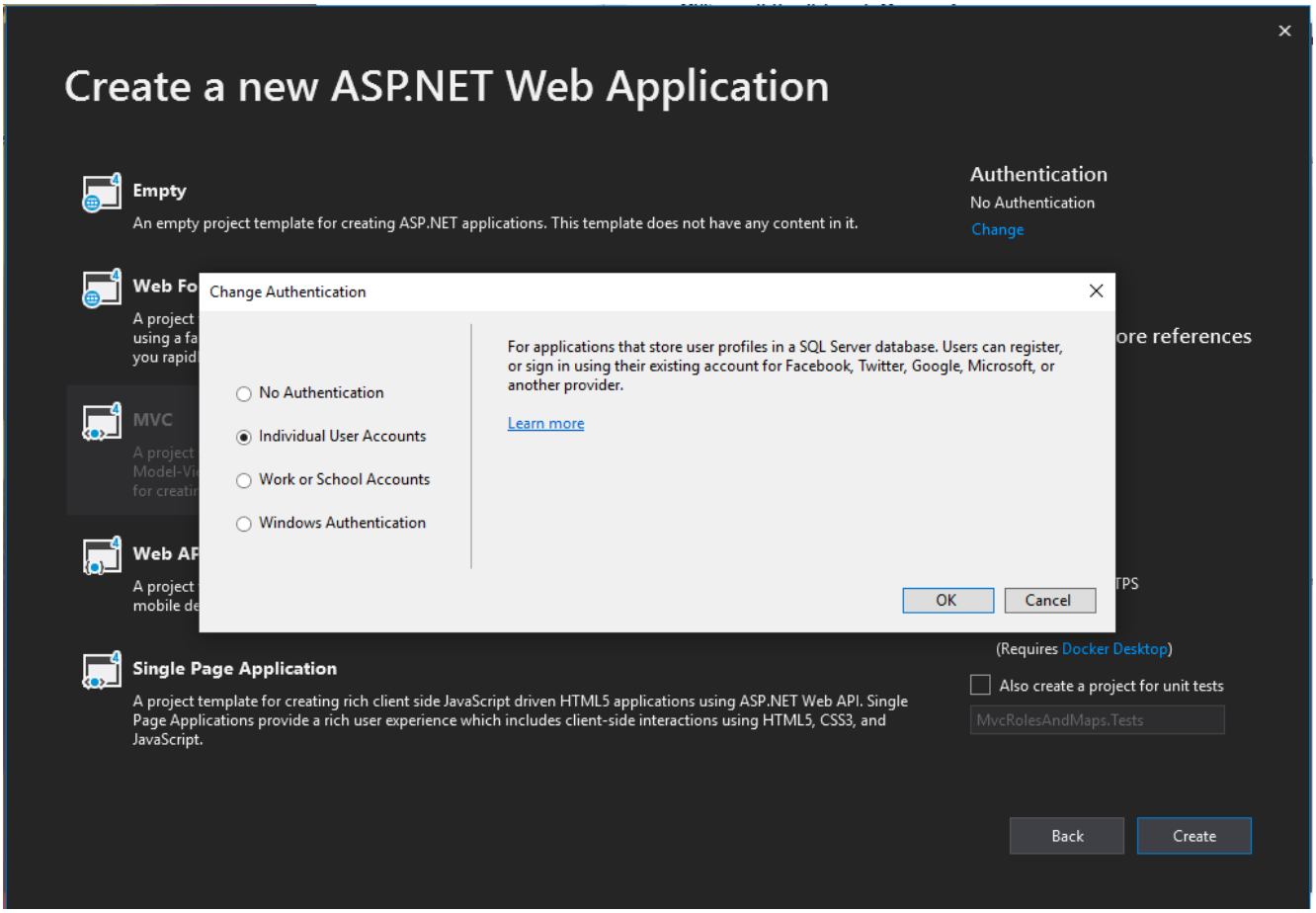

ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
C# Windows Cloud Web

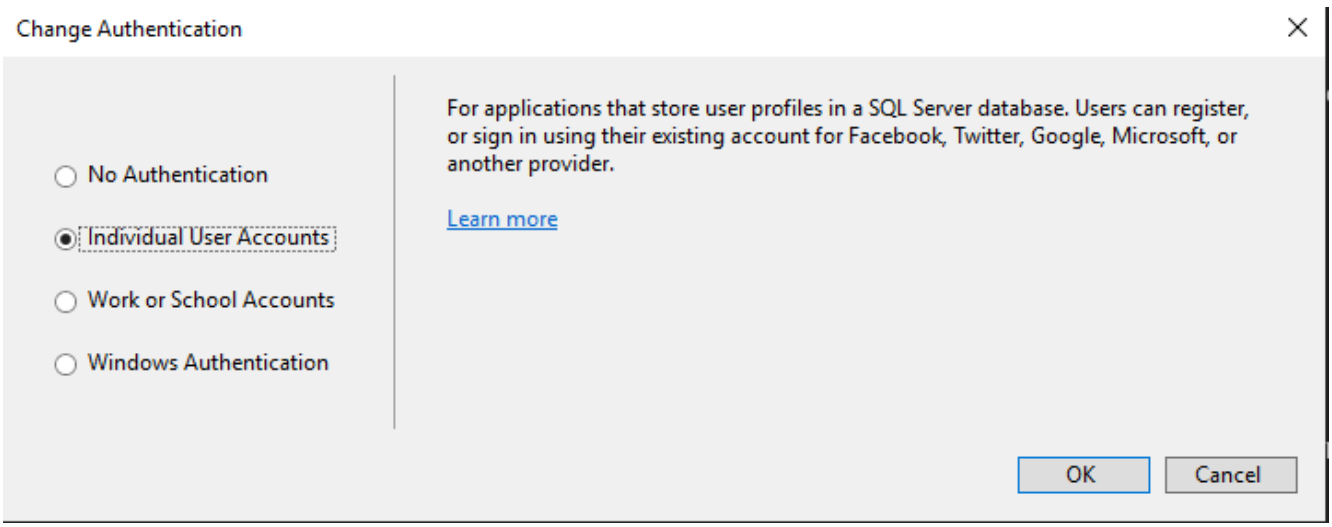

ASP.NET Web Application (.NET Framework)
C# Windows Cloud Web
Project name
MvcRolesAndMaps
Location
E:\Collage\CTU\Formatives\Intermidiate WEB Development\FA 1\Project folder\
...

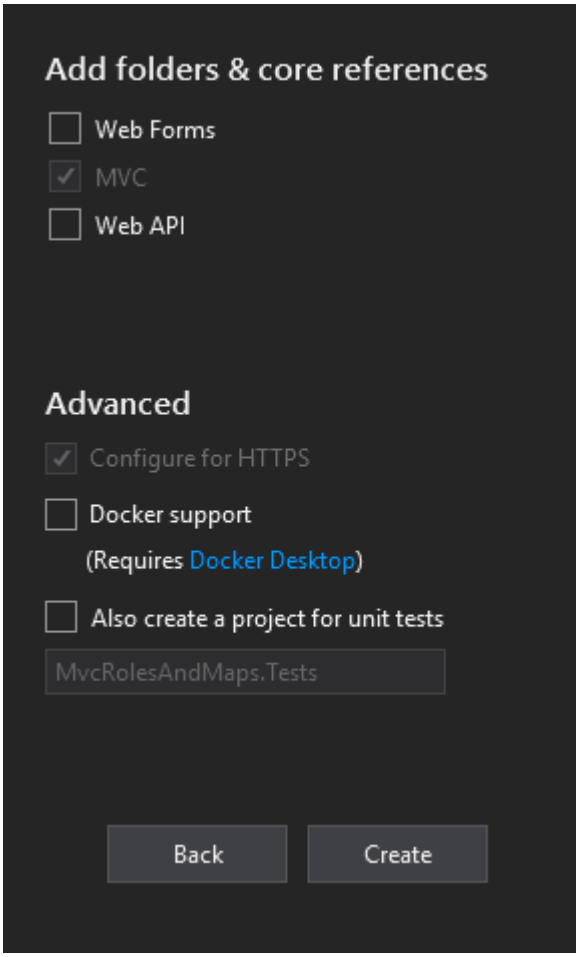

Solution name
MvcRolesAndMaps
Place solution and project in the same directory
Framework
.NET Framework 4.8

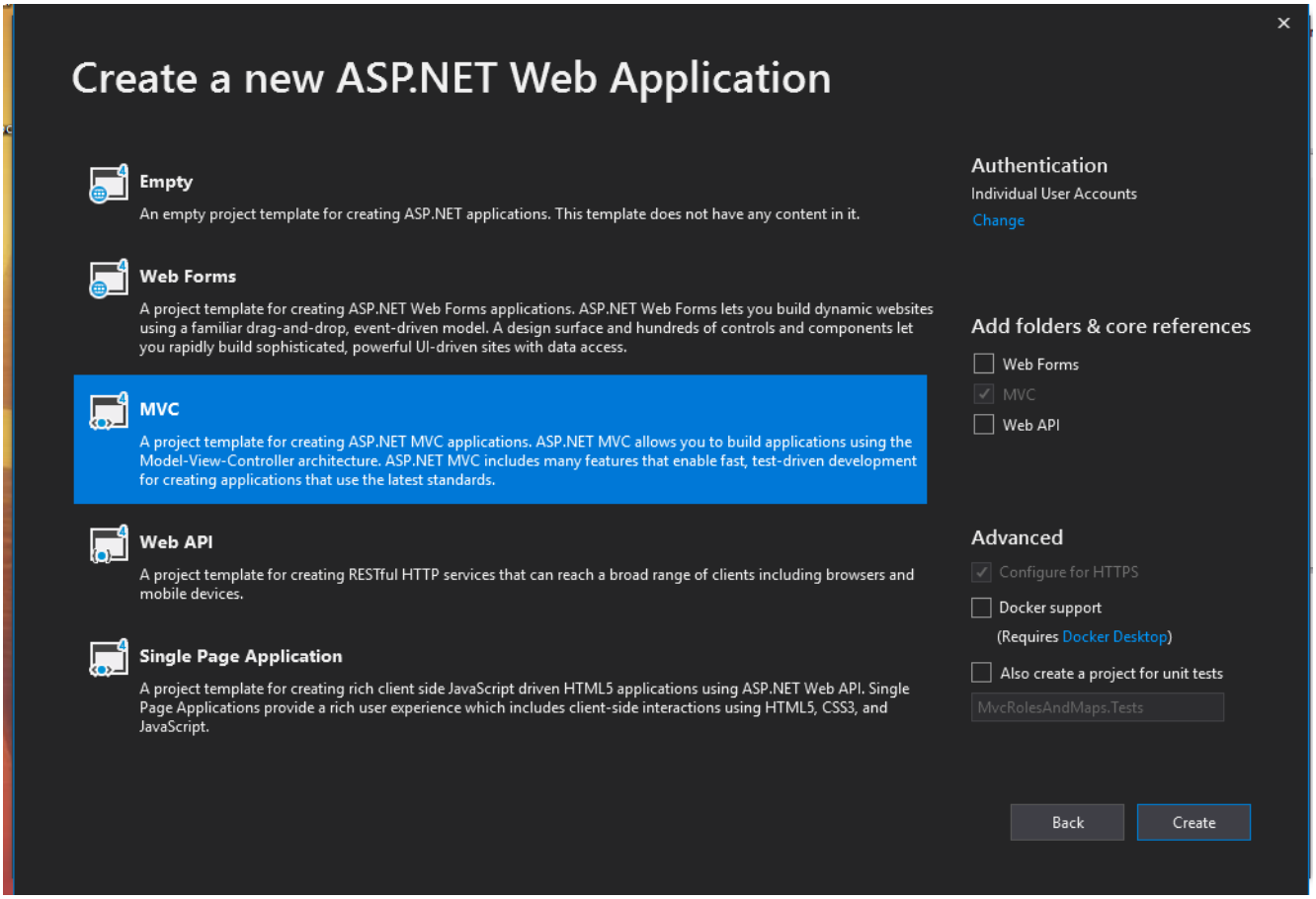

Configure your new project
ASP.NET Web Application (.NET Framework)
C#
Windows
Cloud
Web
Project name
MvcRolesAndMaps
Location
E:\Collage\CTU\Formatives\Intermidiate WEB Development\FA 1\Project folder\
...
Solution name
MvcRolesAndMaps
Place solution and project in the same directory
Framework
.NET Framework 4.8
Back
Create

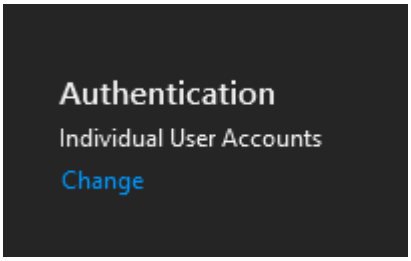

Authentication
No Authentication
Change

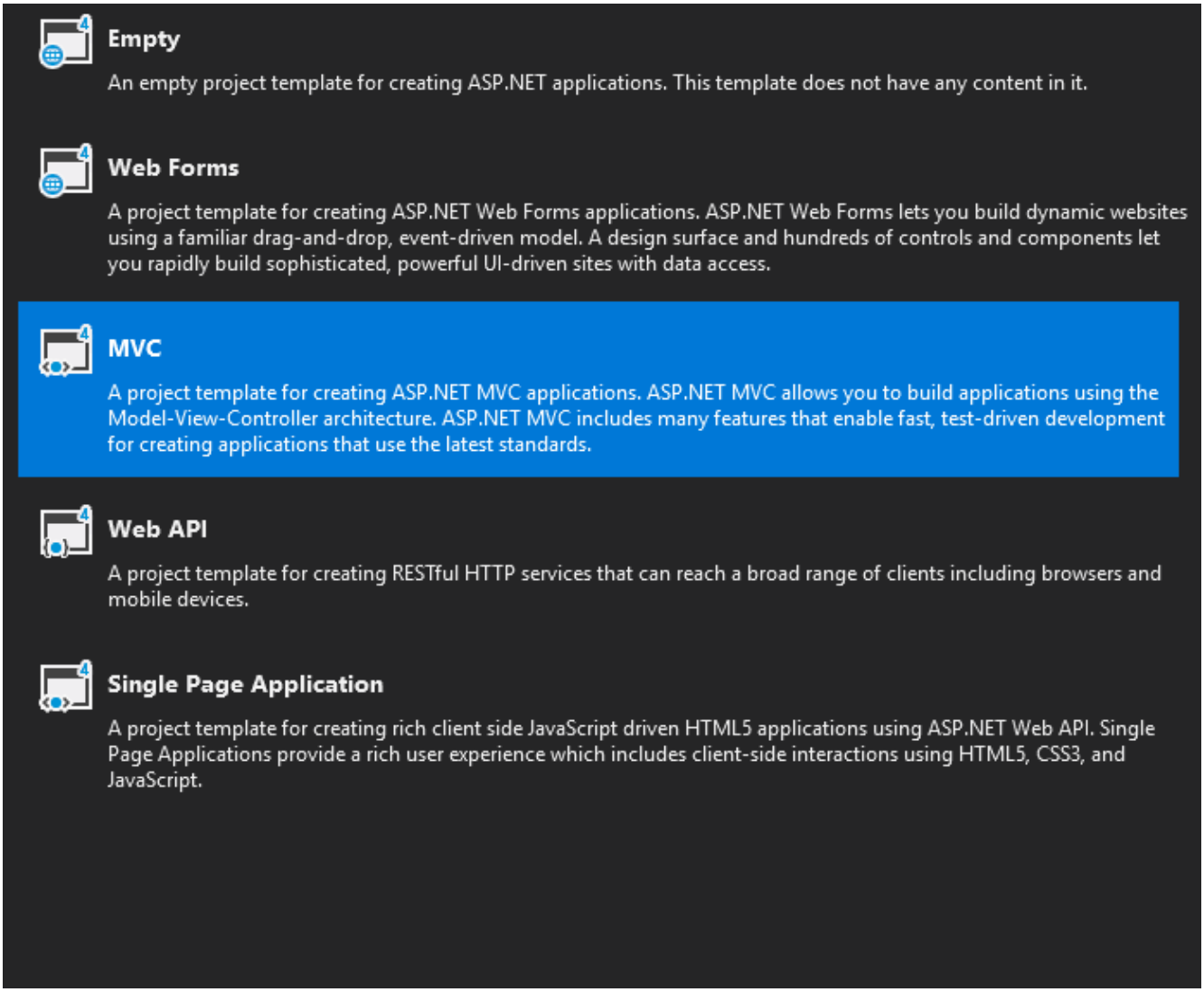

Create a new ASP.NET Web Application
Empty
An empty project template for creating ASP.NET applications. This template does not have any content in it.
Authentication
No Authentication
Change
Change Authentication
No Authentication
Individual User Accounts
Work or School Accounts
Windows Authentication
For applications that store user profiles in a SQL Server database. Users can register, or sign in using their existing account for Facebook, Twitter, Google, Microsoft, or another provider.
Learn more
OK
Cancel
MVC
Single Page Application
A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using HTML5, CSS3, and JavaScript.
(Requires Docker Desktop)
Also create a project for unit tests
MvcRolesAndMaps.Tests
Back
Create

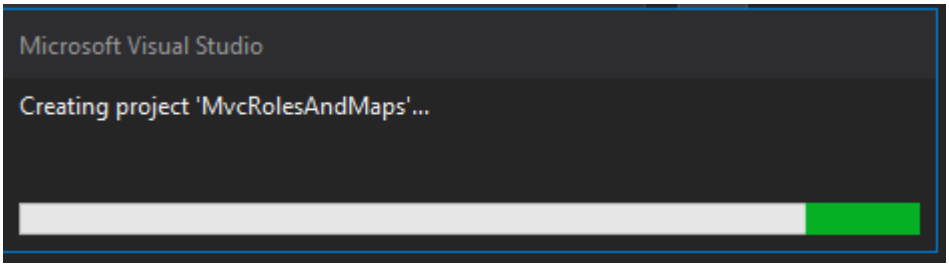

Change Authentication
No Authentication
Individual User Accounts
Work or School Accounts
Windows Authentication
For applications that store user profiles in a SQL Server database. Users can register, or sign in using their existing account for Facebook, Twitter, Google, Microsoft, or another provider.
Learn more
OK
Cancel

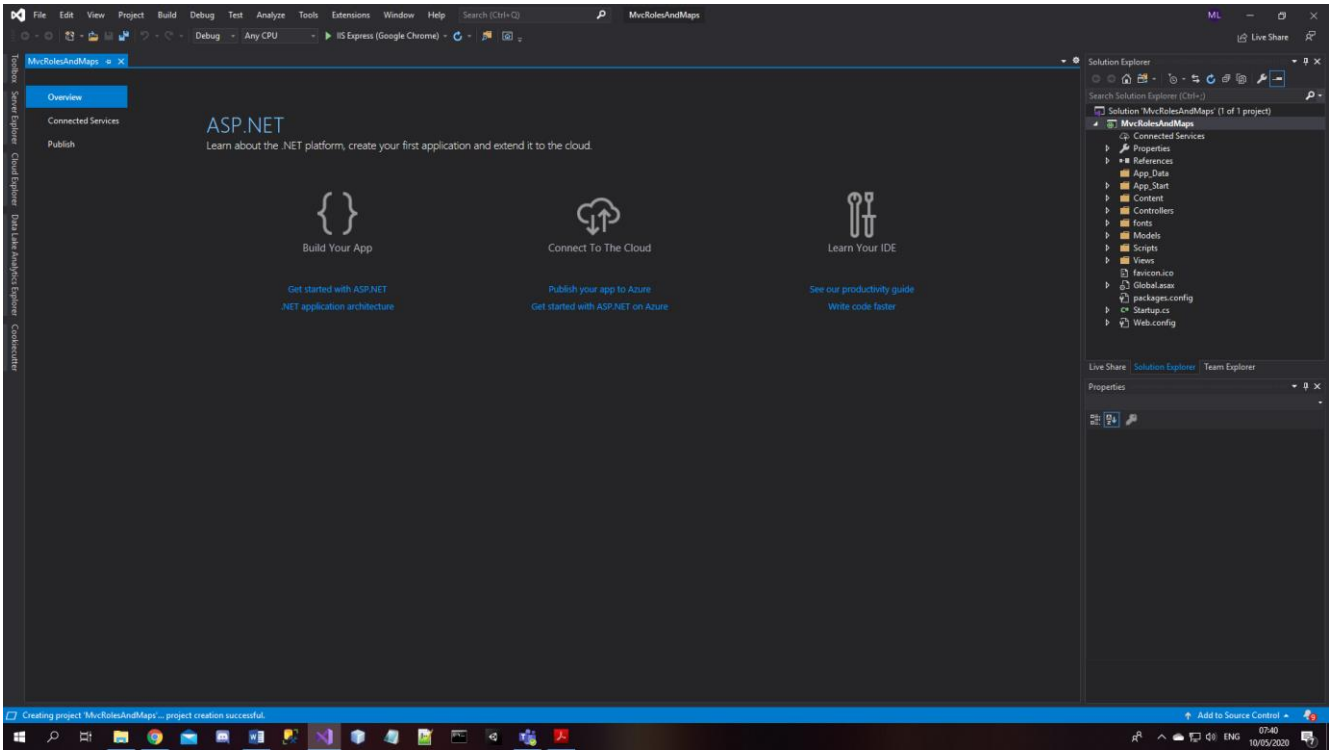

Add folders & core references
Web Forms
MVC
Web API
Advanced
Configure for HTTPS
Docker support
(Requires Docker Desktop)
Also create a project for unit tests
MvcRolesAndMaps.Tests
Back
Create

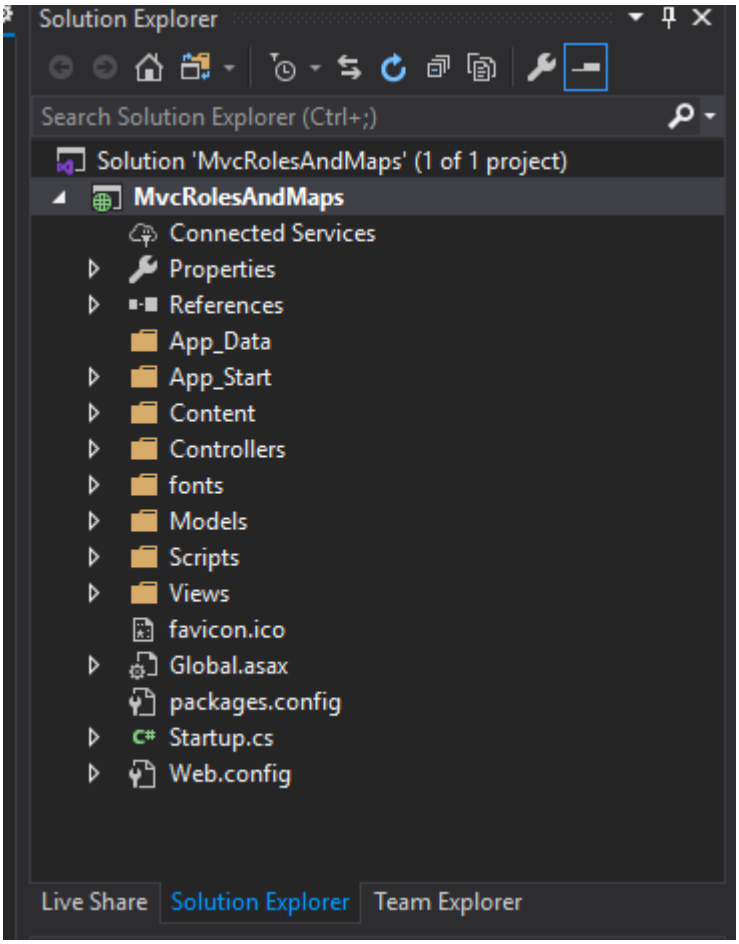

Create a new ASP.NET Web Application
Empty
An empty project template for creating ASP.NET applications. This template does not have any content in it.
Web Forms
A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.
MVC
A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.
Web API
A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.
Single Page Application
A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using HTML5, CSS3, and JavaScript.
Authentication
Individual User Accounts
Change
Add folders & core references
Web Forms
MVC
Web API
Advanced
Configure for HTTPS
Docker support
(Requires Docker Desktop)
Also create a project for unit tests
MvcRolesAndMaps.Tests
Back
Create


Authentication
Individual User Accounts
Change

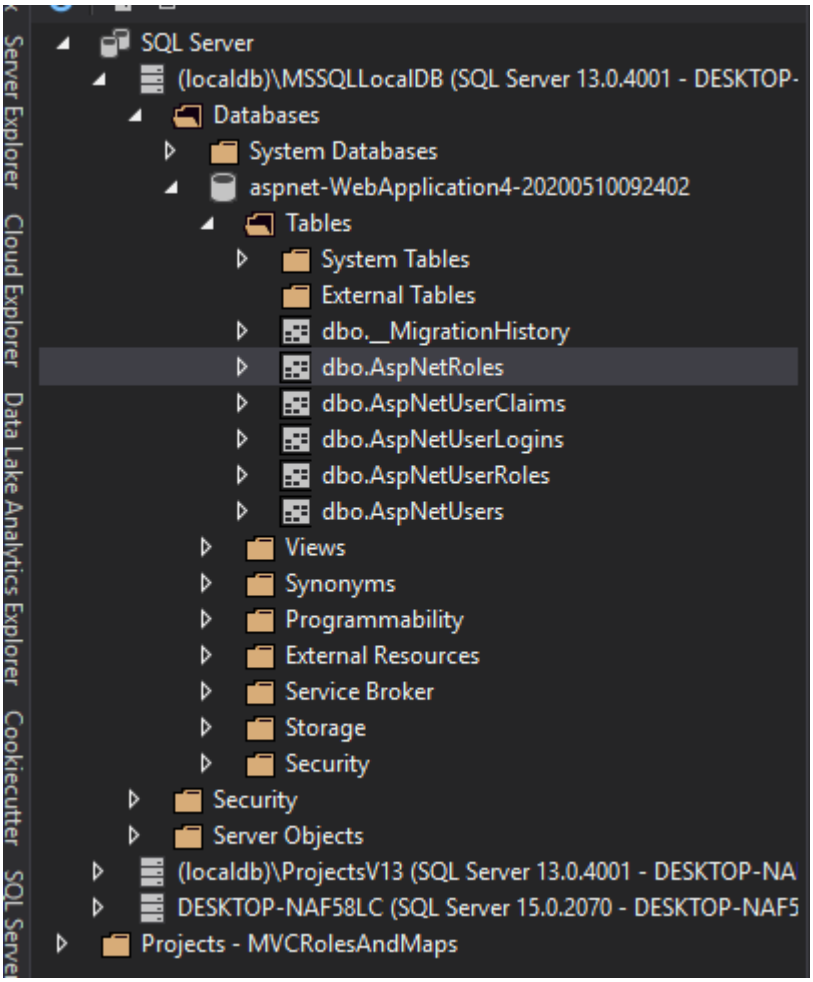

Empty
An empty project template for creating ASP.NET applications. This template does not have any content in it.
Web Forms
A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.
MVC
A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.
Web API
A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.
Single Page Application
A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using HTML5, CSS3, and JavaScript.

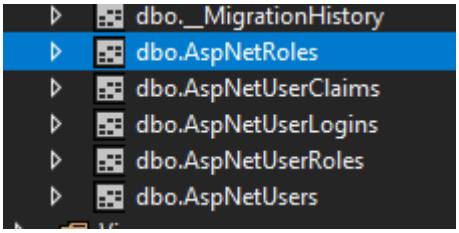

Microsoft Visual Studio
Creating project 'MvcRolesAndMaps'...

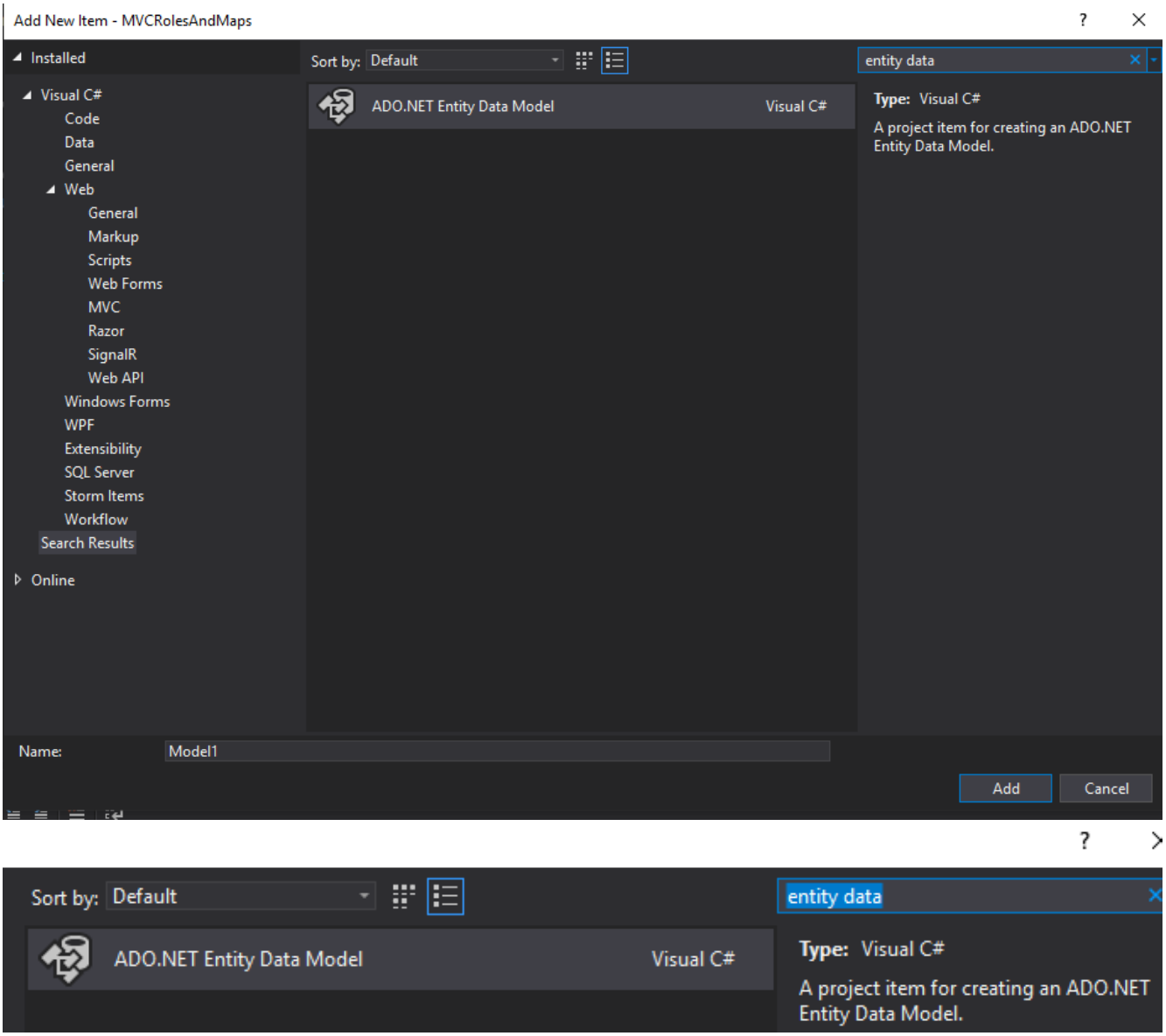

ASP.NET
Learn about the .NET platform, create your first application and extend it to the cloud.
Build Your App
Connect To The Cloud
Learn Your IDE
Solution Explorer
MvcRolesAndMaps


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MvcRolesAndMaps' (1 of 1 project)
MvcRolesAndMaps
Connected Services
Properties
References
App_Data
App_Start
Content
Controllers
fonts
Models
Scripts
Views
favicon.ico
Global.asax
packages.config
Startup.cs
Web.config
Live Share
Solution Explorer
Team Explorer

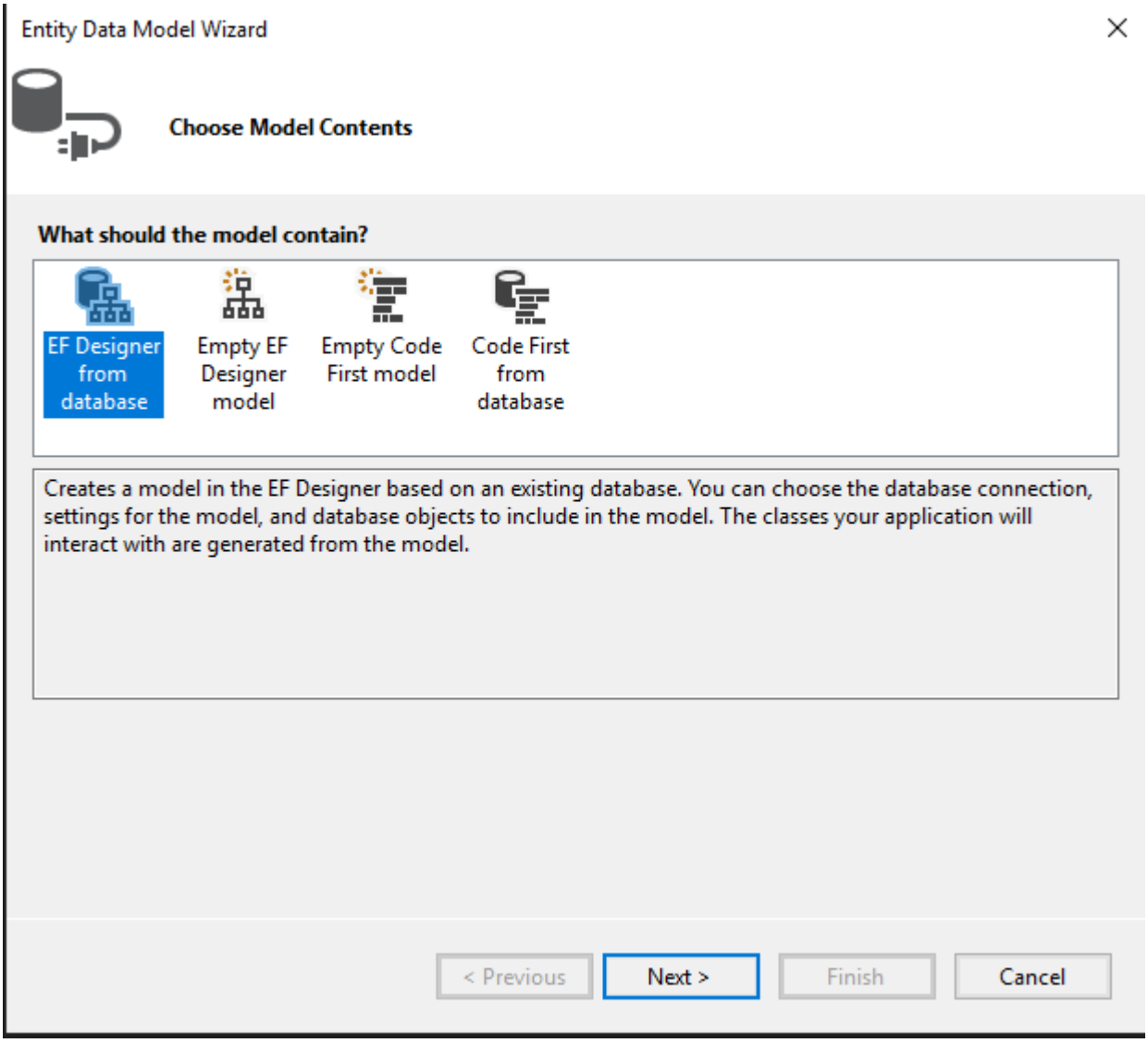
Step 2

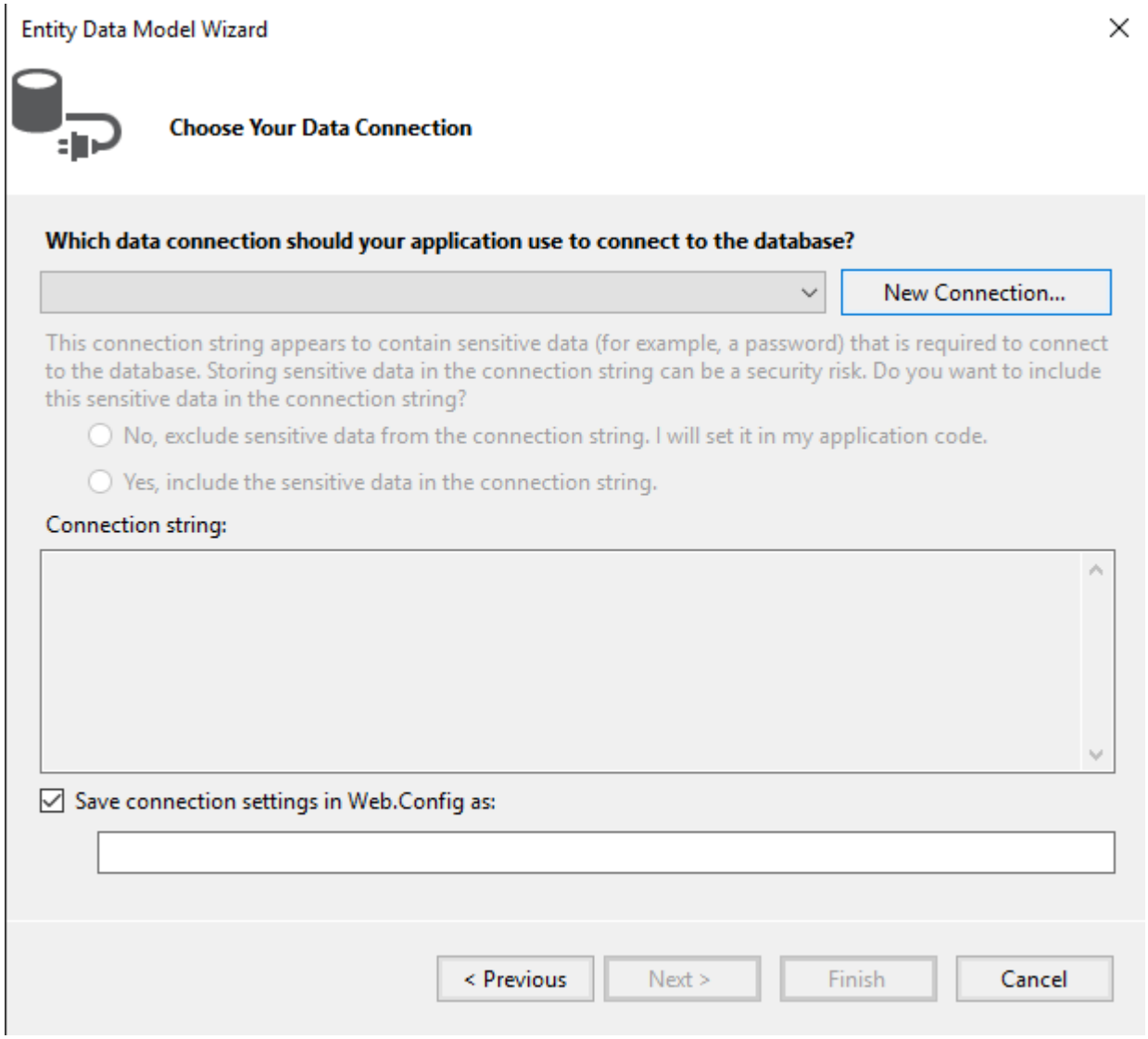

Server Explorer
Cloud Explorer
Data Lake Analytics Explorer
Cookiecutter
SQL Serve
SQL Server
(localdb)\MSSQLLocalDB (SQL Server 13.0.4001 - DESKTOP-
Databases
System Databases
aspnet-WebApplication4-20200510092402
Tables
System Tables
External Tables
dbo.__MigrationHistory
dbo.AspNetRoles
dbo.AspNetUserClaims
dbo.AspNetUserLogins
dbo.AspNetUserRoles
dbo.AspNetUsers
Views
Synonyms
Programmability
External Resources
Service Broker
Storage
Security
Security
Server Objects
(localdb)\ProjectsV13 (SQL Server 13.0.4001 - DESKTOP-NA
DESKTOP-NAF58LC (SQL Server 15.0.2070 - DESKTOP-NAF5
Projects - MVCRolesAndMaps

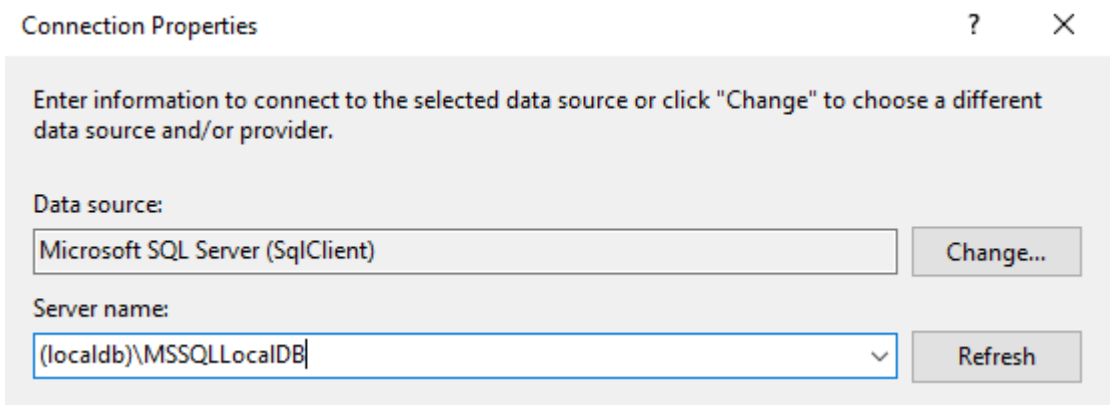

dbo.__MigrationHistory
dbo.AspNetRoles
dbo.AspNetUserClaims
dbo.AspNetUserLogins
dbo.AspNetUserRoles
dbo.AspNetUsers

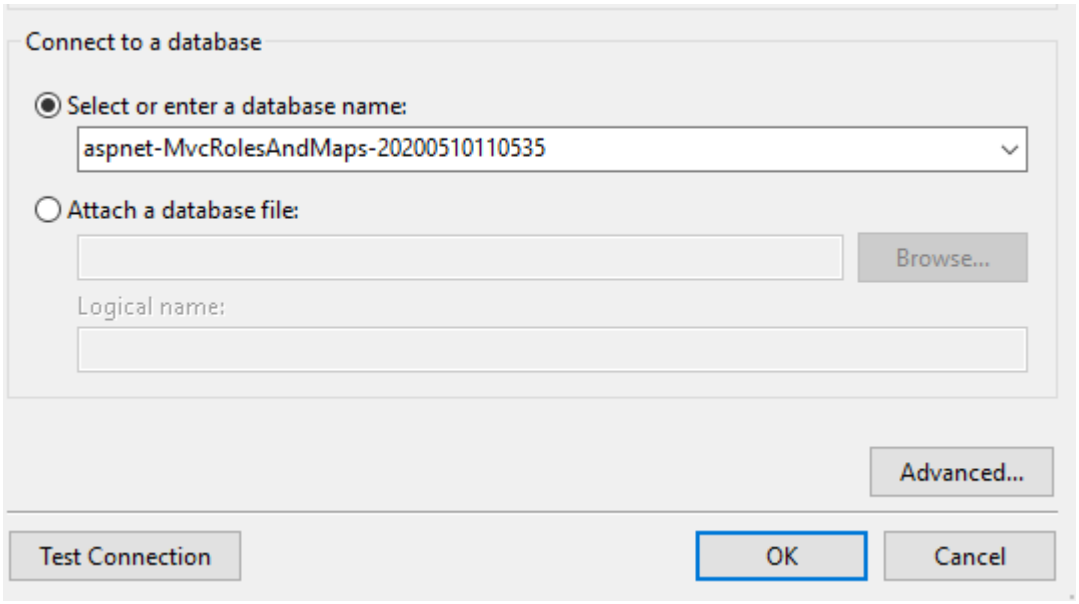

Add New Item - MVCRolesAndMaps
Installed
Visual C#
Code
Data
General
Web
General
Markup
Scripts
Web Forms
MVC
Razor
SignalR
Web API
Windows Forms
WPF
Extensibility
SQL Server
Storm Items
Workflow
Search Results
Online
Sort by: Default
ADO.NET Entity Data Model
Visual C#
entity data
Type: Visual C#
A project item for creating an ADO.NET Entity Data Model.
Name:
Model1
Add
Cancel

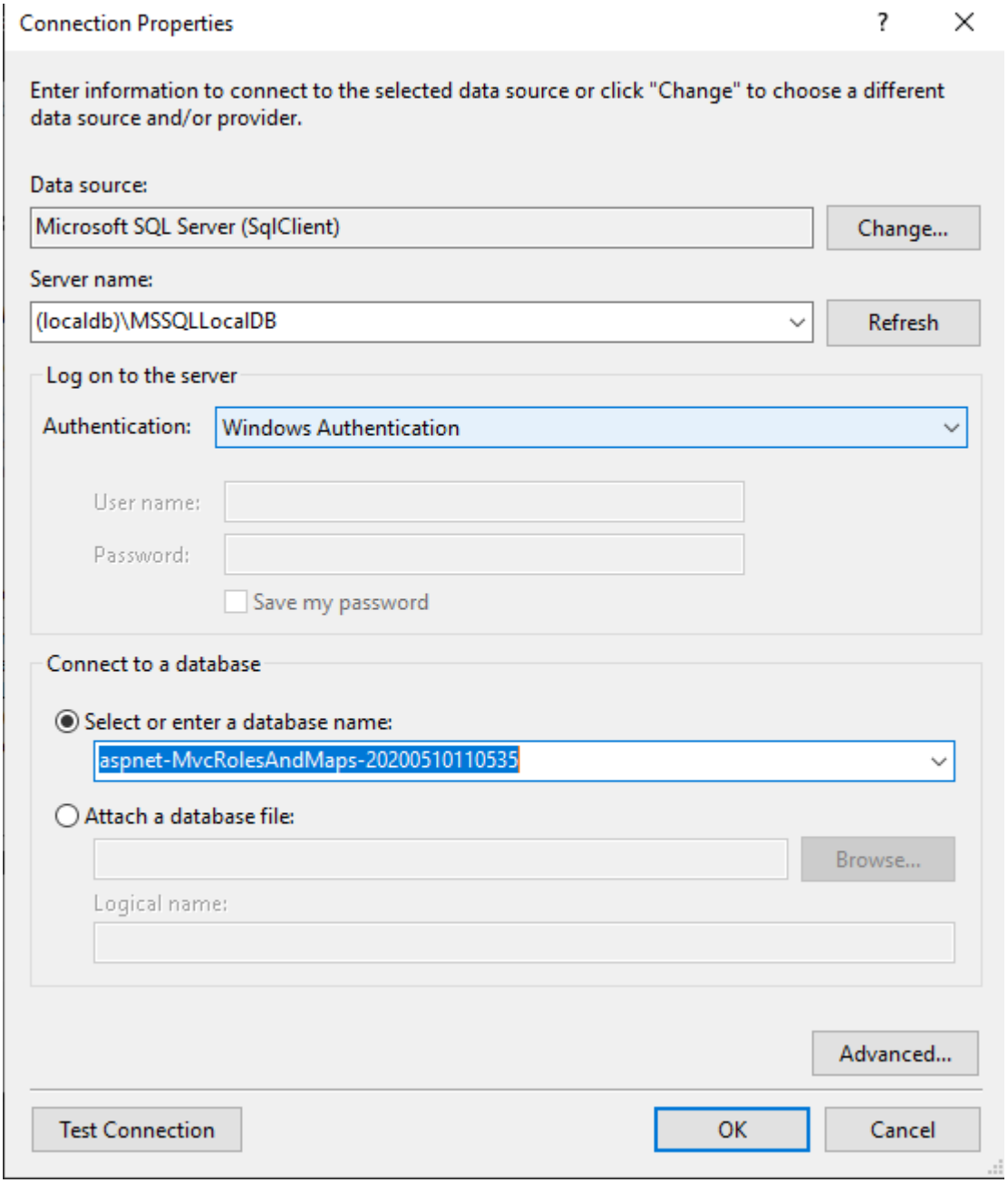
Sort by: Default
ADO.NET Entity Data Model
Visual C#
entity data
Type: Visual C#
A project item for creating an ADO.NET Entity Data Model.

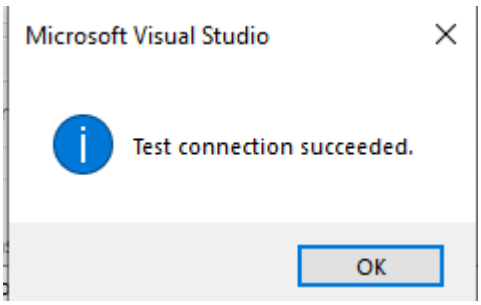

Entity Data Model Wizard
Choose Model Contents
What should the model contain?
EF Designer from database
Empty EF Designer model
Empty Code First model
Code First from database
Creates a model in the EF Designer based on an existing database. You can choose the database connection, settings for the model, and database objects to include in the model. The classes your application will interact with are generated from the model.
< Previous
Next >
Finish
Cancel

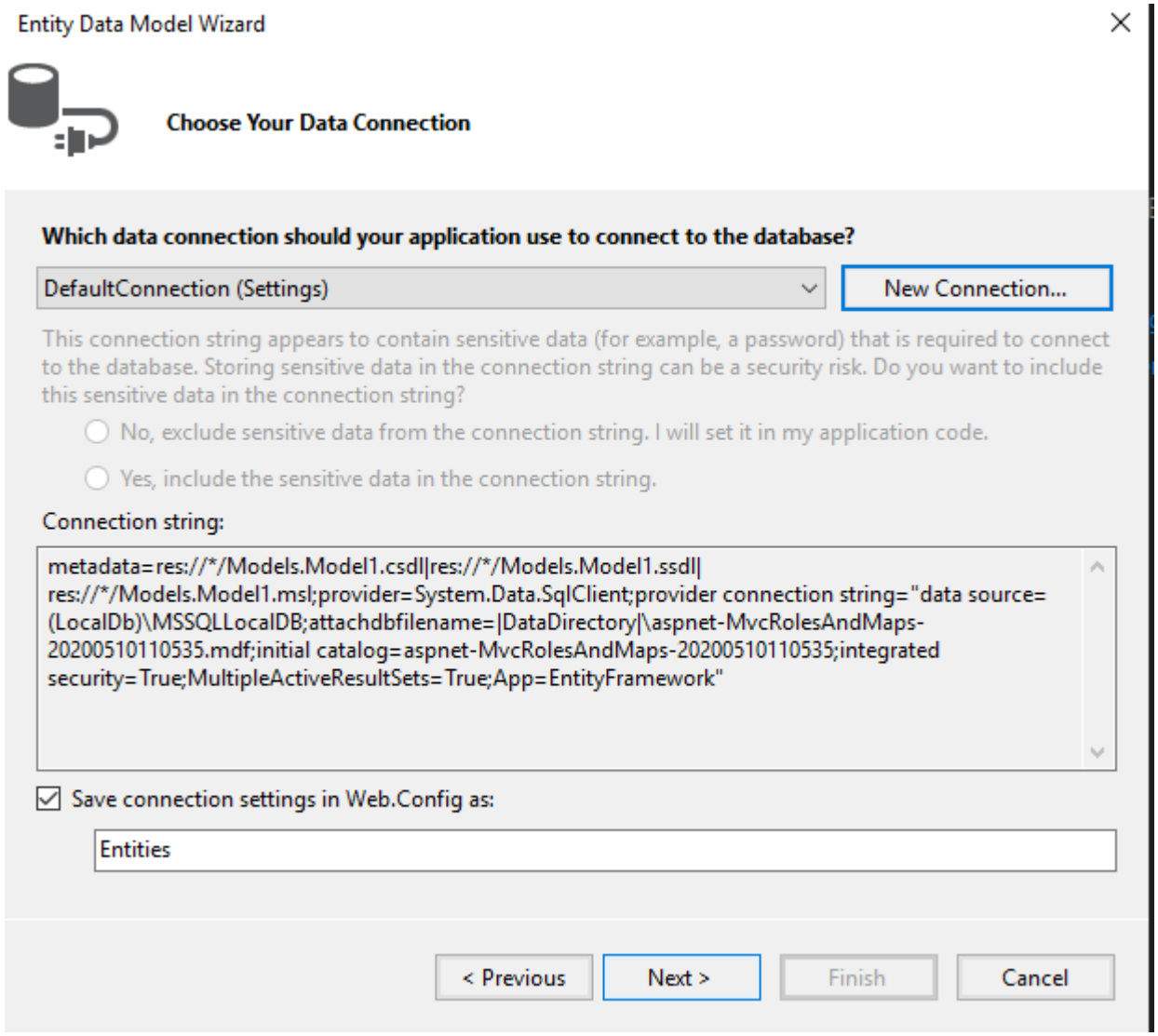

Entity Data Model Wizard
Choose Your Data Connection
Which data connection should your application use to connect to the database?
New Connection...
This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in 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.
Connection string:
Save connection settings in Web.Config as:
< Previous
Next >
Finish
Cancel

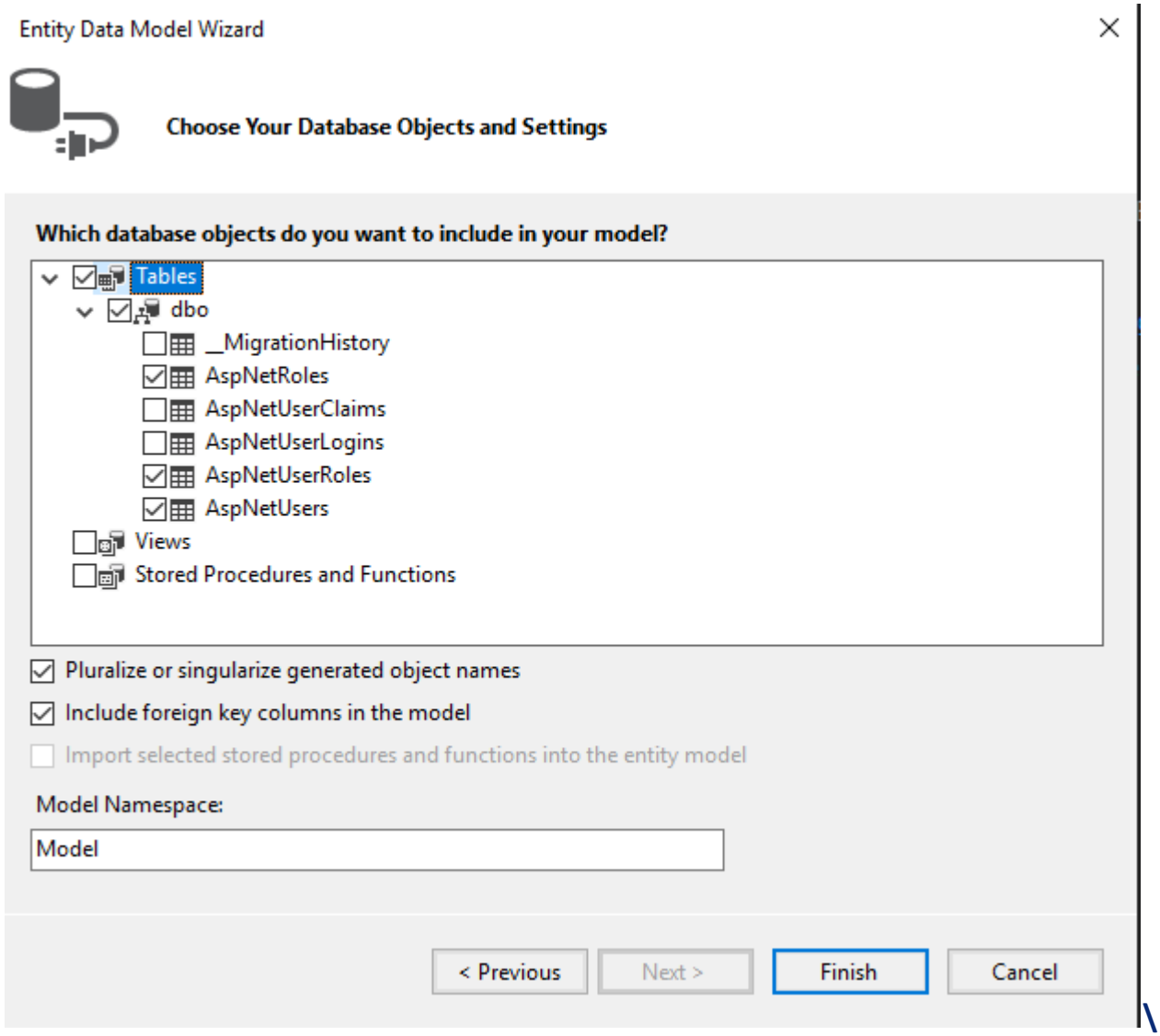

Connection Properties
Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.
Data source:
Microsoft SQL Server (SqlClient)
Change...
Server name:
(localdb)\MSSQLLocalDB
Refresh

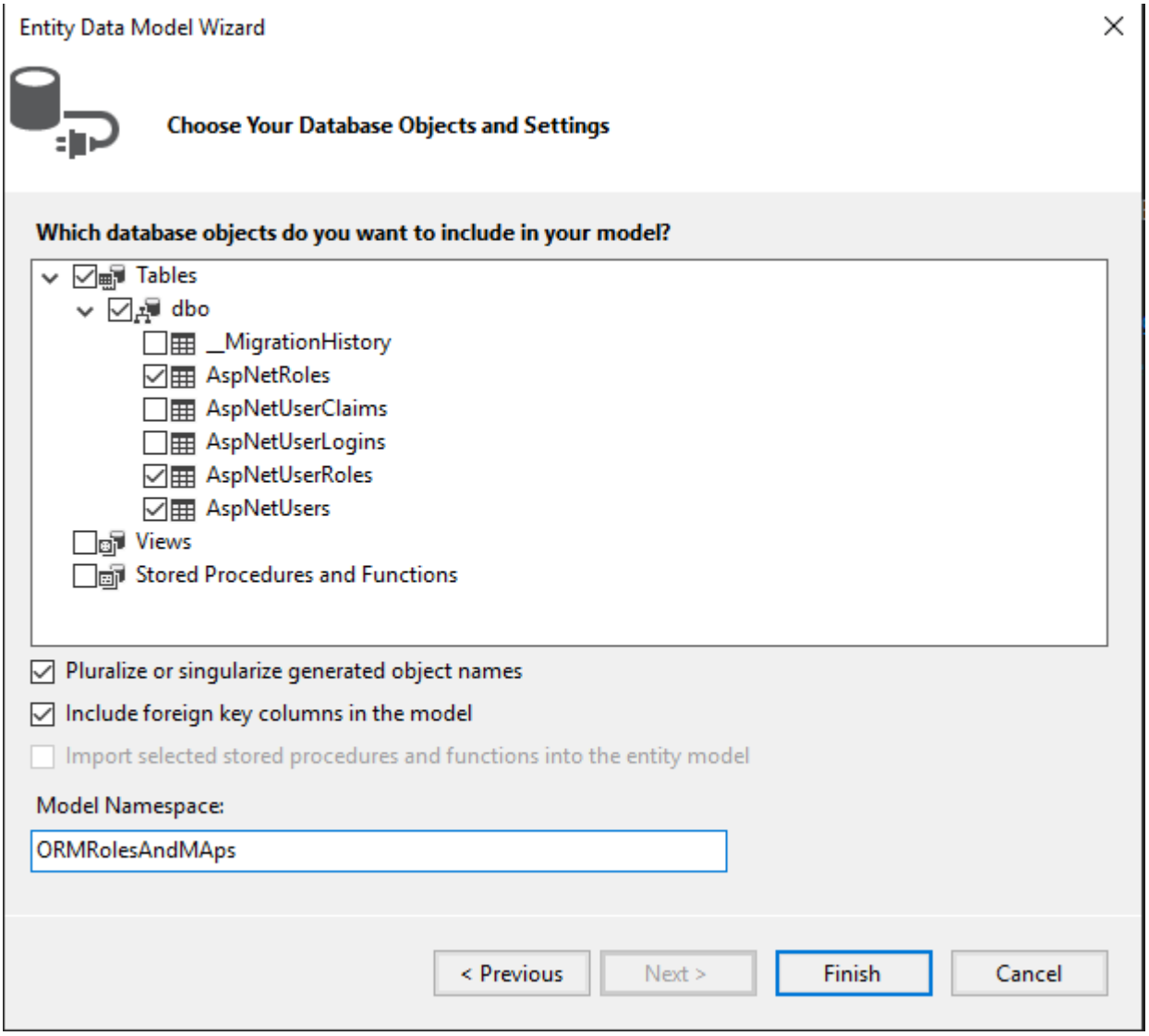

Connect to a database
Select or enter a database name:
aspnet-MvcRolesAndMaps-20200510110535
Attach a database file:
Browse...
Logical name:
Advanced...
Test Connection
OK
Cancel

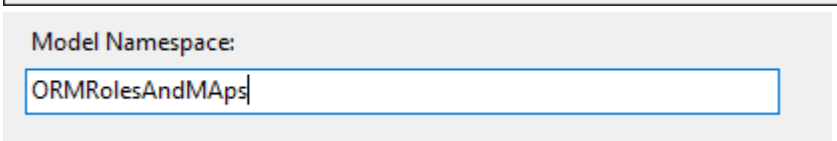

Connection Properties
Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.
Data source:
Microsoft SQL Server (SqlClient)
Change...
Server name:
(localdb)\MSSQLLocalDB
Refresh
Log on to the server
Authentication:
Windows Authentication
User name:
Password:
Save my password
Connect to a database
Select or enter a database name:
aspnet-MvcRolesAndMaps-20200510110535
Attach a database file:
Browse...
Logical name:
Advanced...
Test Connection
OK
Cancel

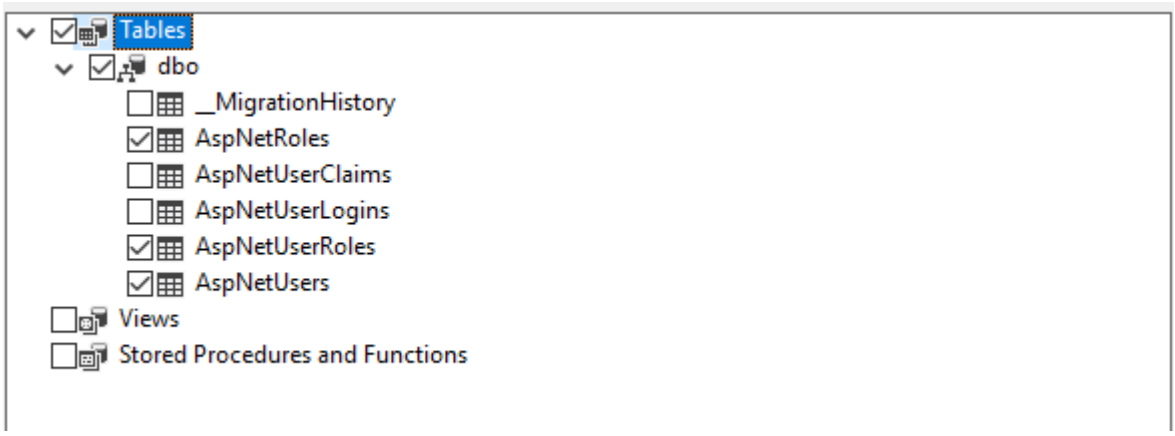

Microsoft Visual Studio
Test connection succeeded.
OK

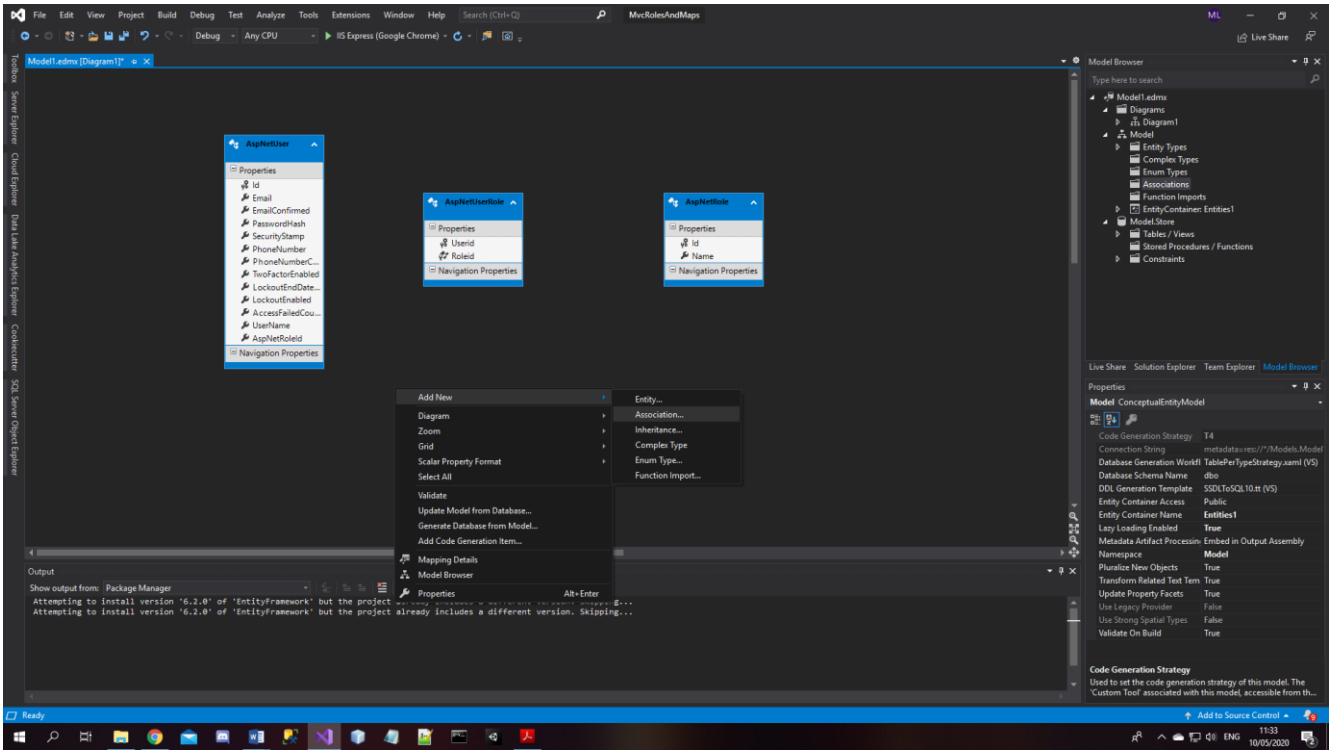

Entity Data Model Wizard
Choose Your Data Connection
Which data connection should your application use to connect to the database?
DefaultConnection (Settings)
New Connection...
This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. 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.
Connection string:
metadata=res://*/Models.Model1.csdl|res://*/Models.Model1.ssdl|
res://*/Models.Model1.msl;provider=System.Data.SqlClient;provider connection string="data source=
(LocalDb)\MSSQLLocalDB;attachdbfilename=|DataDirectory|\aspnet-MvcRolesAndMaps-
20200510110535.mdf;initial catalog=aspnet-MvcRolesAndMaps-20200510110535;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
Save connection settings in Web.Config as:
Entities
< Previous
Next >
Finish
Cancel

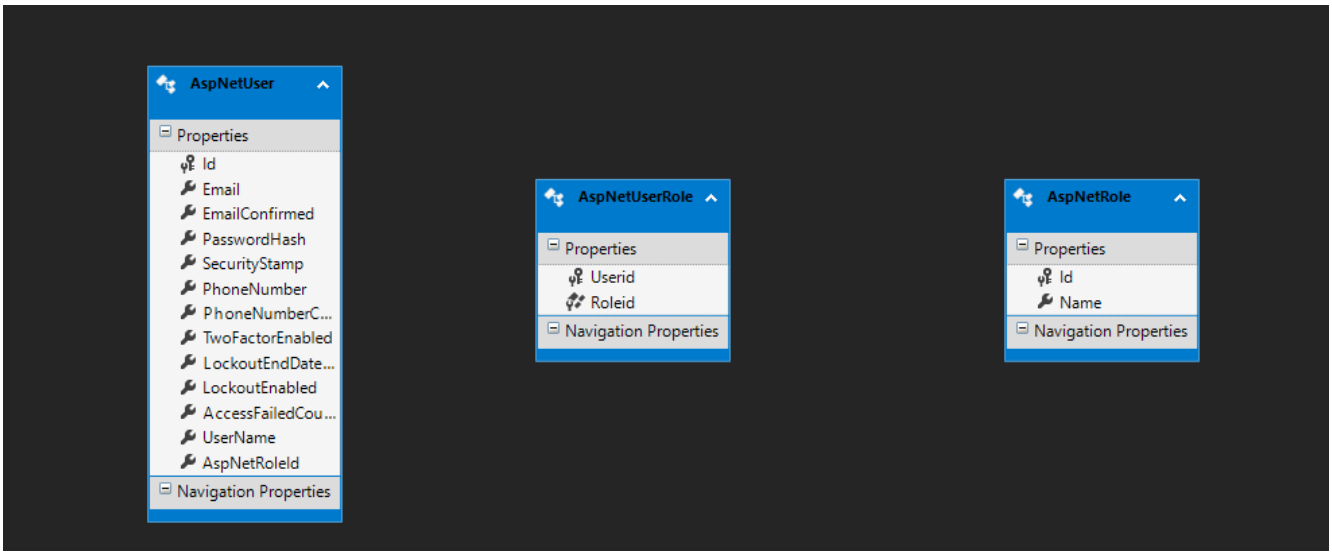

Entity Data Model Wizard
Choose Your Database Objects and Settings
Which database objects do you want to include in your model?
Tables
dbo
__MigrationHistory
AspNetRoles
AspNetUserClaims
AspNetUserLogins
AspNetUserRoles
AspNetUsers
Views
Stored Procedures and Functions
Pluralize or singularize generated object names
Include foreign key columns in the model
Import selected stored procedures and functions into the entity model
Model Namespace:
Model
< Previous
Next >
Finish
Cancel

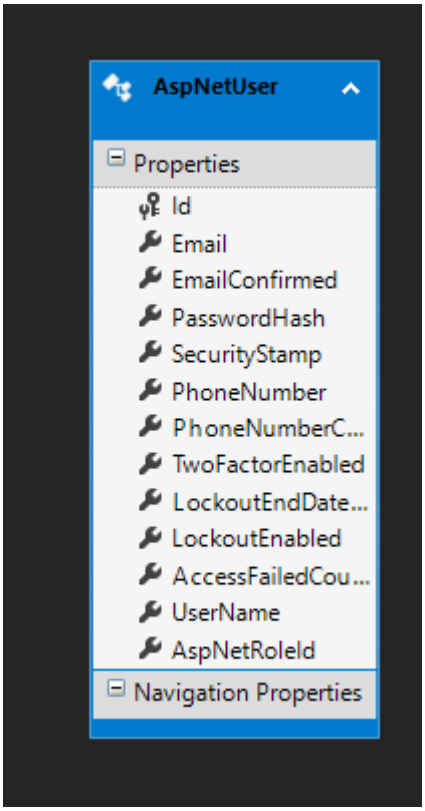

Entity Data Model Wizard
Choose Your Database Objects and Settings
Which database objects do you want to include in your model?
Tables
dbo
__MigrationHistory
AspNetRoles
AspNetUserClaims
AspNetUserLogins
AspNetUserRoles
AspNetUsers
Views
Stored Procedures and Functions
Pluralize or singularize generated object names
Include foreign key columns in the model
Import selected stored procedures and functions into the entity model
Model Namespace:
ORMRolesAndMAps
< Previous
Next >
Finish
Cancel

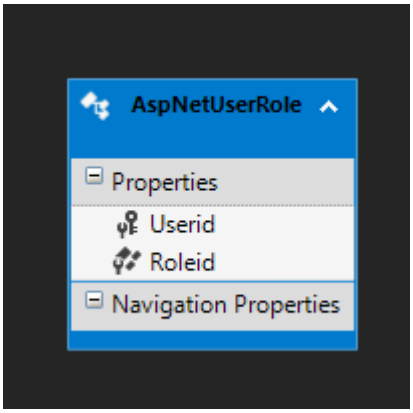

Model Namespace:
ORMRolesAndMAps

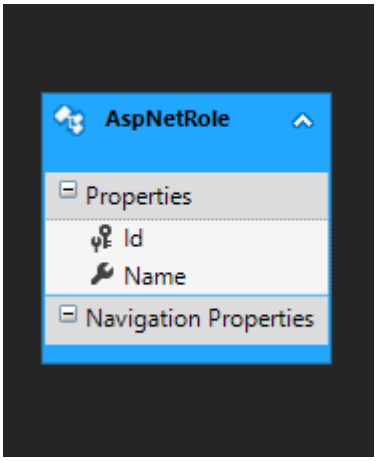

Tables
dbo
__MigrationHistory
AspNetRoles
AspNetUserClaims
AspNetUserLogins
AspNetUserRoles
AspNetUsers
Views
Stored Procedures and Functions

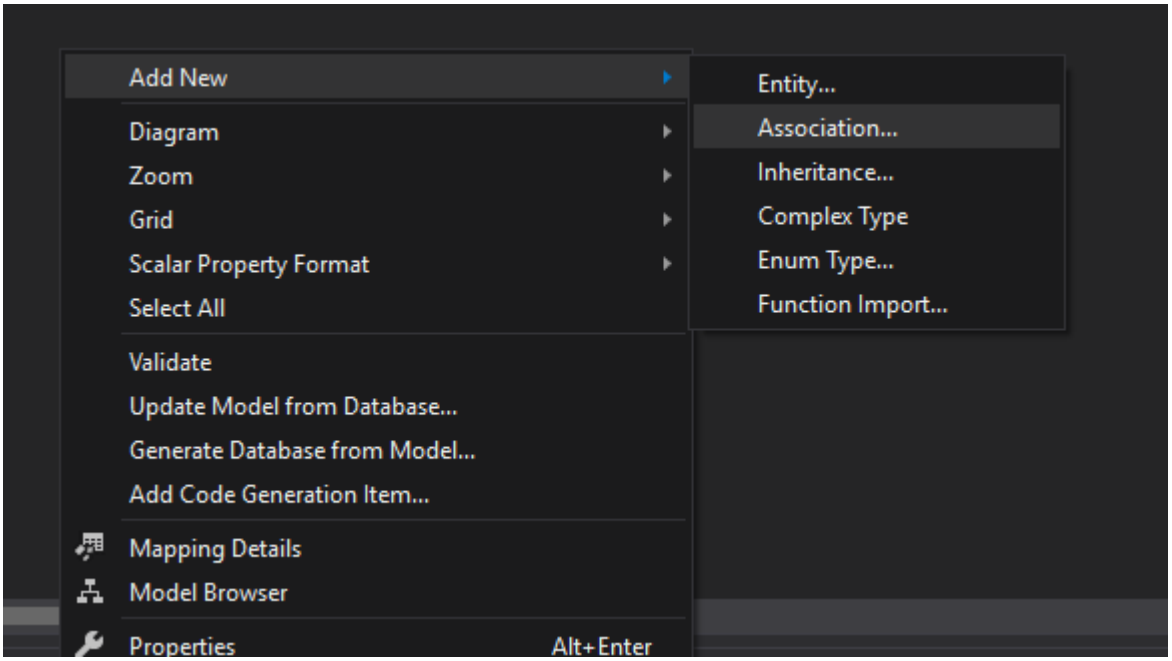

AspNetUser
Properties
Id
Email
EmailConfirmed
PasswordHash
SecurityStamp
PhoneNumber
PhoneNumberC...
TwoFactorEnabled
LockoutEndDate...
LockoutEnabled
AccessFailedCou...
UserName
AspNetRoleId
Navigation Properties
AspNetUserRole
Properties
UserId
RoleId
Navigation Properties
AspNetRole
Properties
Id
Name
Navigation Properties
Add New
Diagram
Zoom
Grid
Scalar Property Format
Select All
Validate
Update Model from Database...
Generate Database from Model...
Add Code Generation Item...
Mapping Details
Model Browser
Properties
Entity...
Association...
Inheritance...
Complex Type
Enum Type...
Function Import...

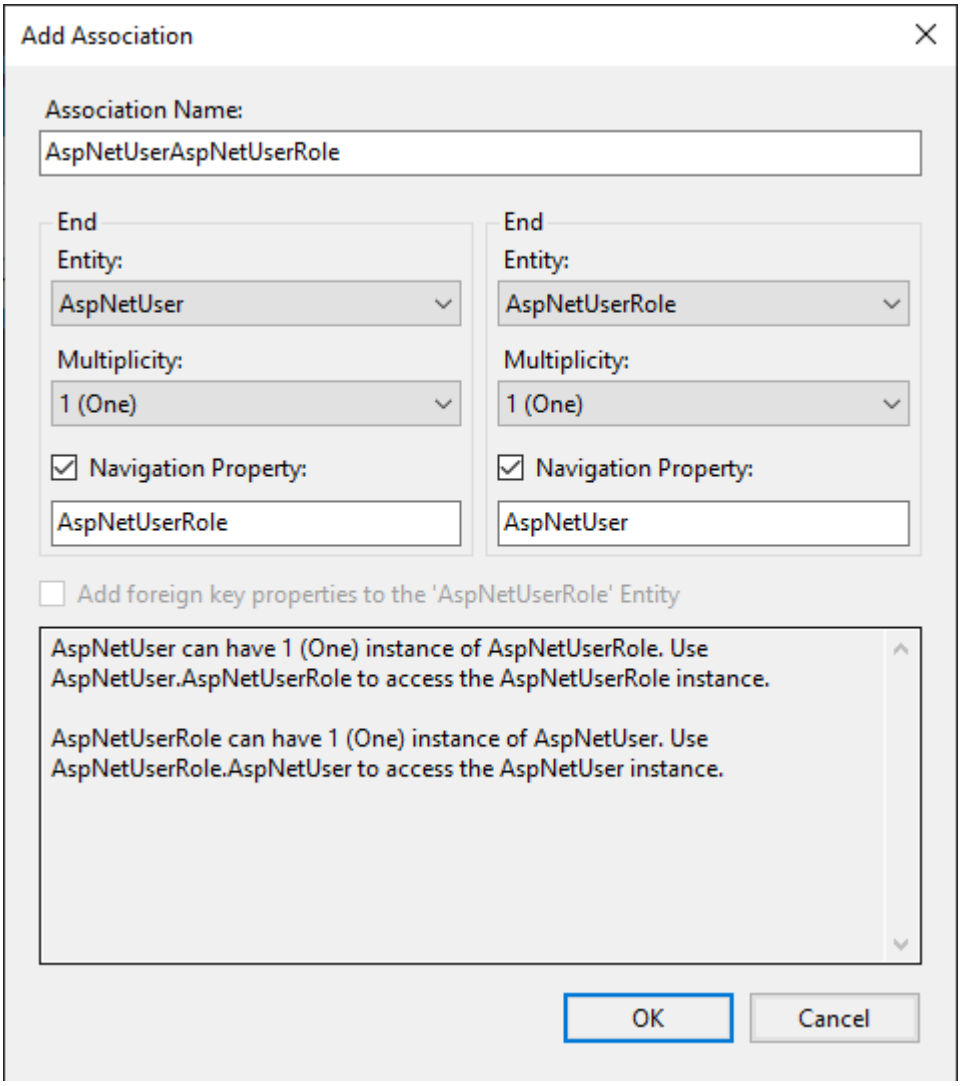

AspNetUser
Properties
Id
Email
EmailConfirmed
PasswordHash
SecurityStamp
PhoneNumber
PhoneNumberC...
TwoFactorEnabled
LockoutEndDate...
LockoutEnabled
AccessFailedCou...
UserName
AspNetRoleId
Navigation Properties
AspNetUserRole
Properties
UserId
RoleId
Navigation Properties
AspNetRole
Properties
Id
Name
Navigation Properties

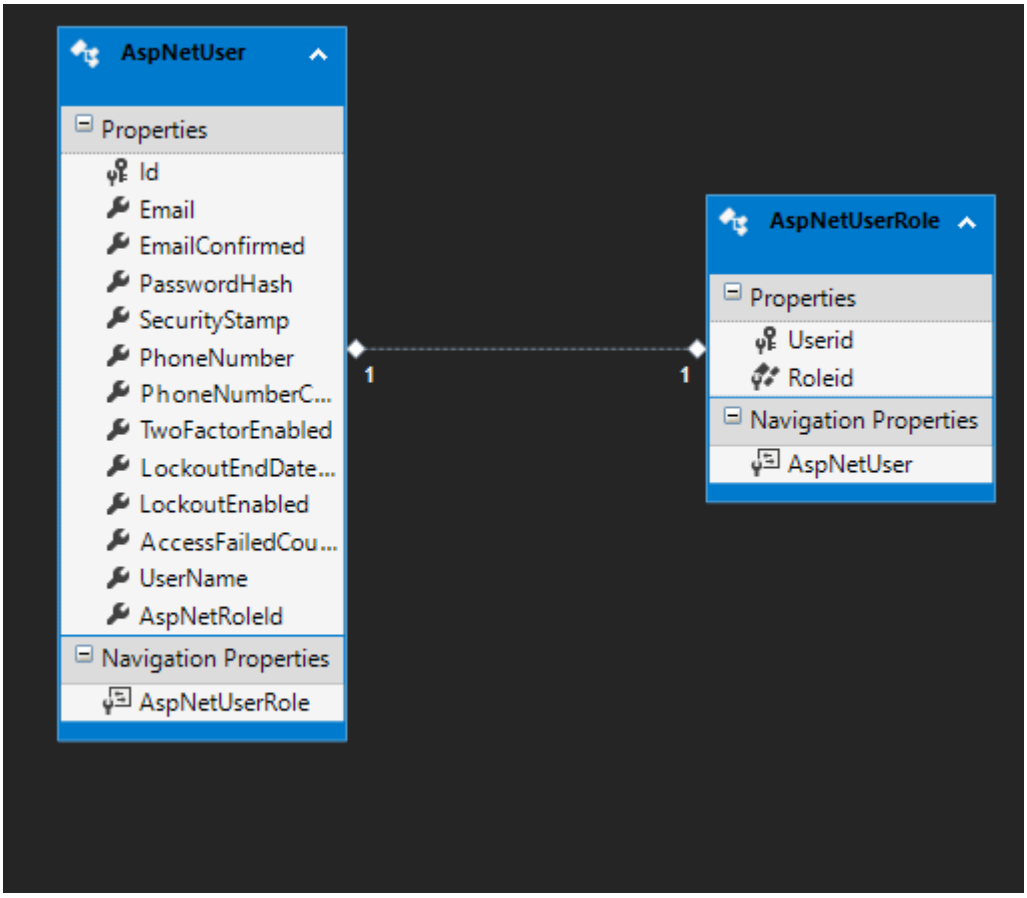

AspNetUser
Properties
Id
Email
EmailConfirmed
PasswordHash
SecurityStamp
PhoneNumber
PhoneNumberC...
TwoFactorEnabled
LockoutEndDate...
LockoutEnabled
AccessFailedCou...
UserName
AspNetRoleId
Navigation Properties

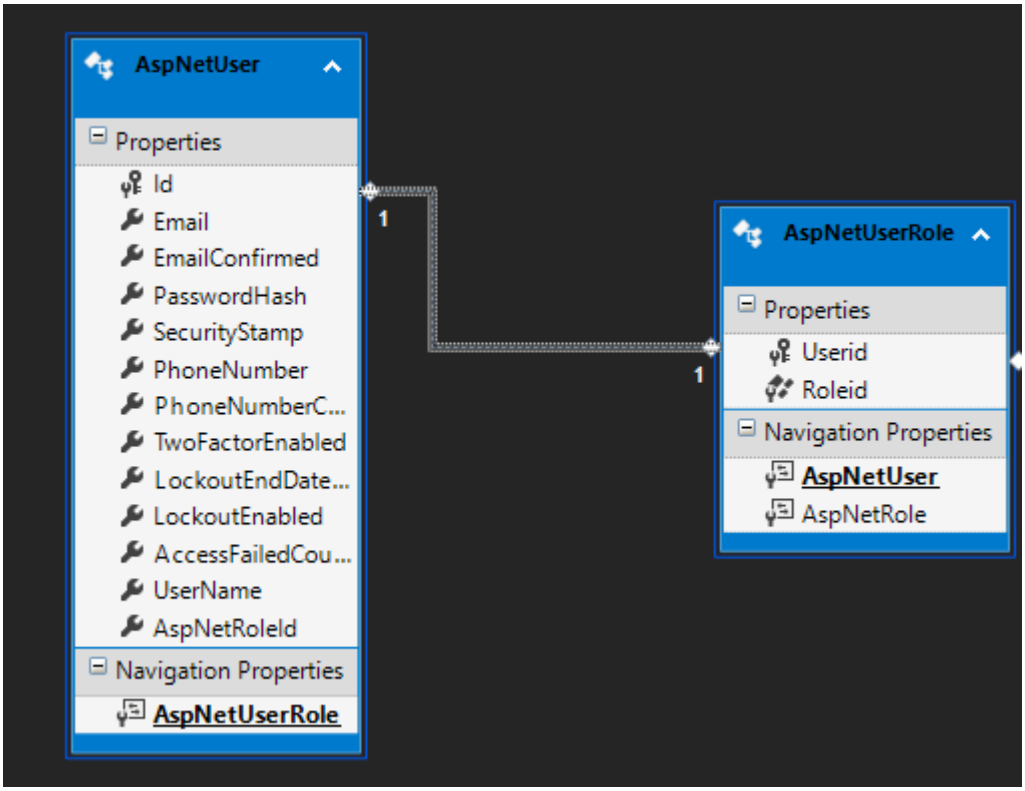

AspNetUserRole
Properties
Userid
Roleid
Navigation Properties

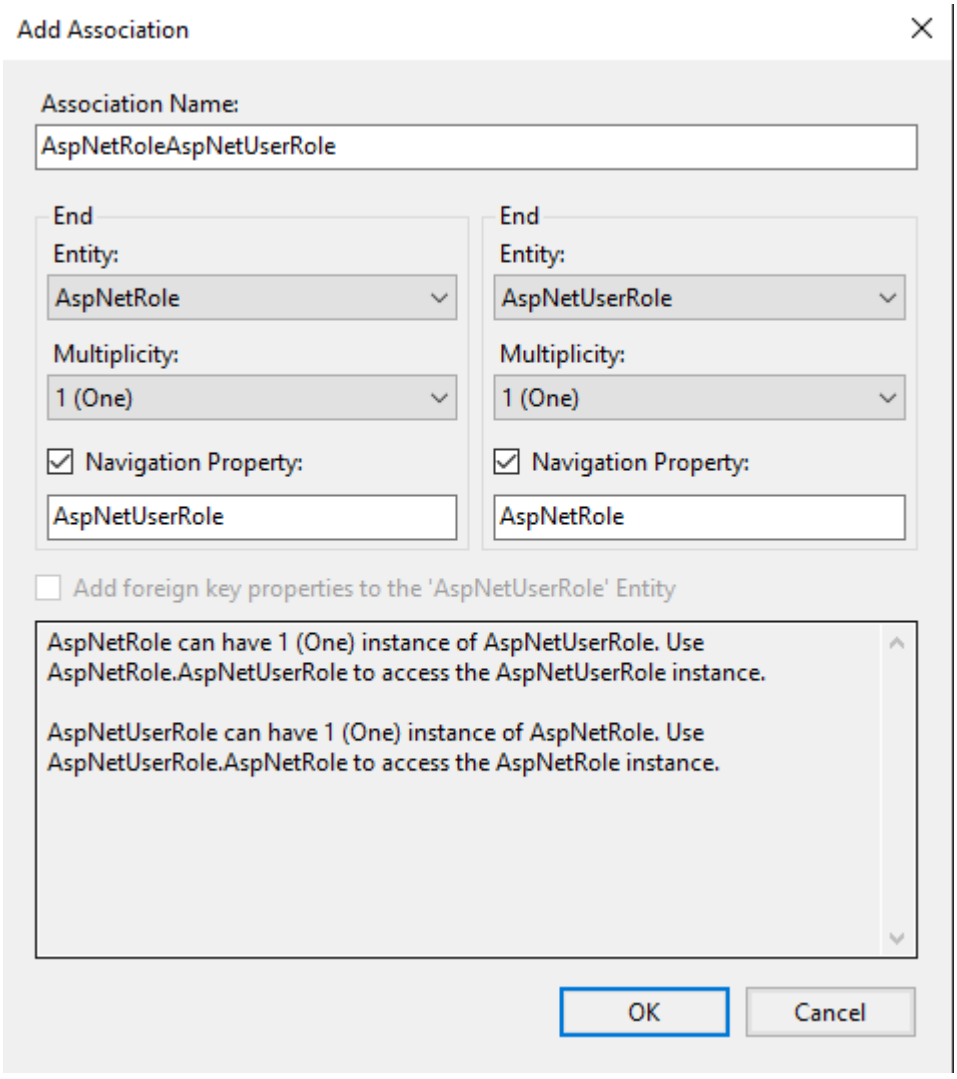

AspNetRole
Properties
Id
Name
Navigation Properties

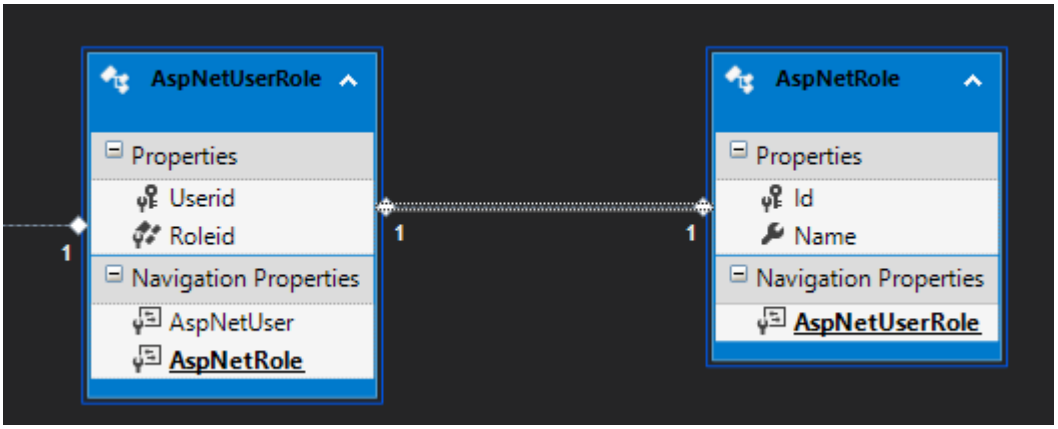

Add New
Diagram
Zoom
Grid
Scalar Property Format
Select All
Validate
Update Model from Database...
Generate Database from Model...
Add Code Generation Item...
Mapping Details
Model Browser
Properties
Alt+Enter
Entity...
Association...
Inheritance...
Complex Type
Enum Type...
Function Import...

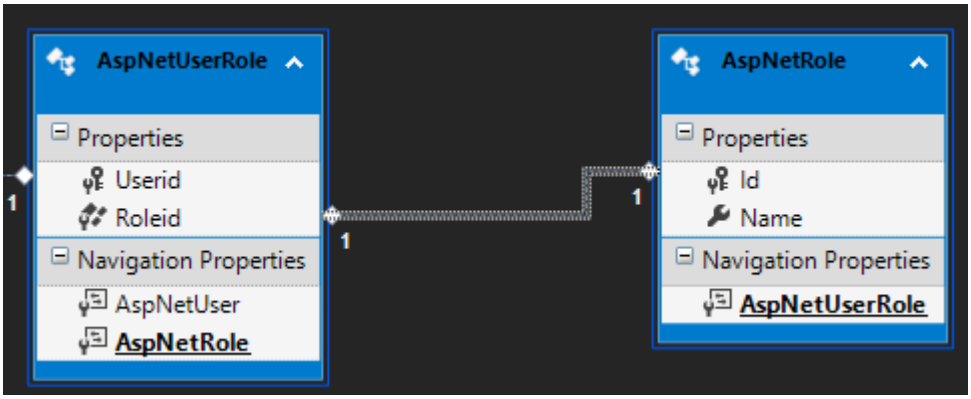

Add Association
Association Name:
AspNetUserAspNetUserRole
End
Entity:
AspNetUser
Multiplicity:
1 (One)
Navigation Property:
AspNetUserRole
End
Entity:
AspNetUserRole
Multiplicity:
1 (One)
Navigation Property:
AspNetUser
Add foreign key properties to the 'AspNetUserRole' Entity
AspNetUser can have 1 (One) instance of AspNetUserRole. Use AspNetUser.AspNetUserRole to access the AspNetUserRole instance.
AspNetUserRole can have 1 (One) instance of AspNetUser. Use AspNetUserRole.AspNetUser to access the AspNetUser instance.
OK
Cancel

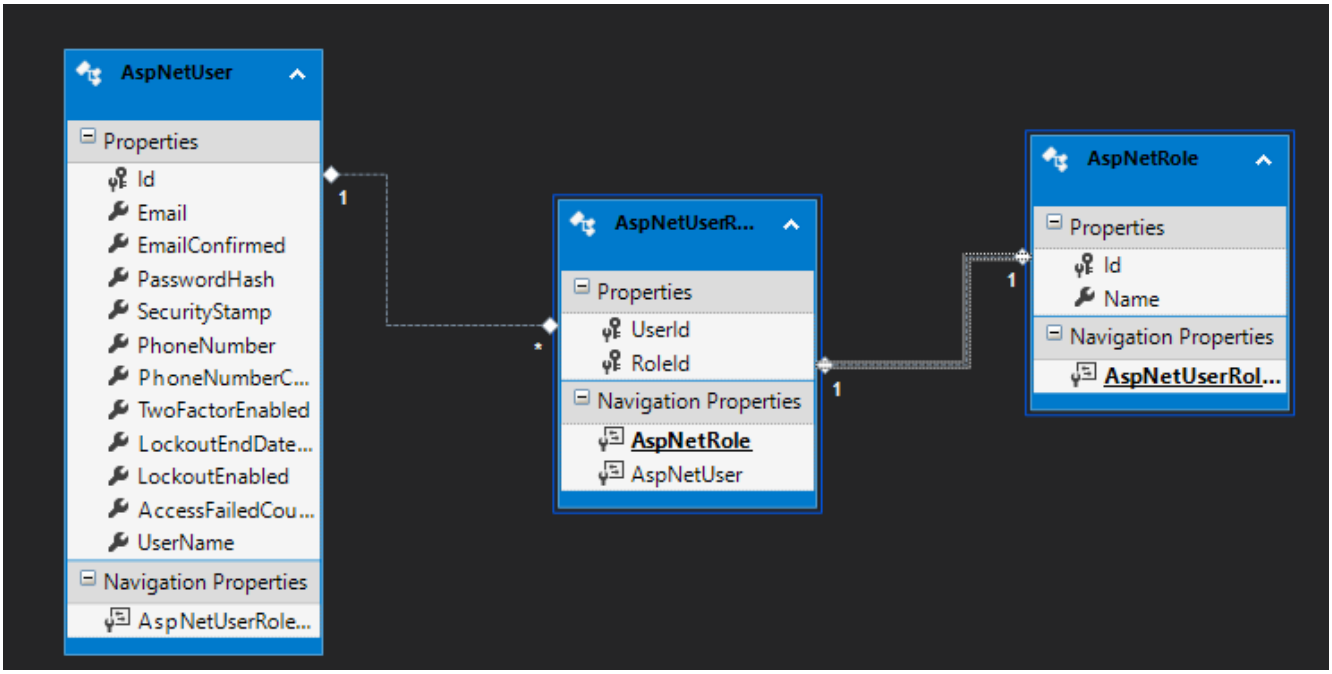

AspNetUser
Properties
Id
Email
EmailConfirmed
PasswordHash
SecurityStamp
PhoneNumber
PhoneNumberC...
TwoFactorEnabled
LockoutEndDate...
LockoutEnabled
AccessFailedCou...
UserName
AspNetRoleId
Navigation Properties
AspNetUserRole
1
1
AspNetUserRole
Properties
Userid
Roleid
Navigation Properties
AspNetUser

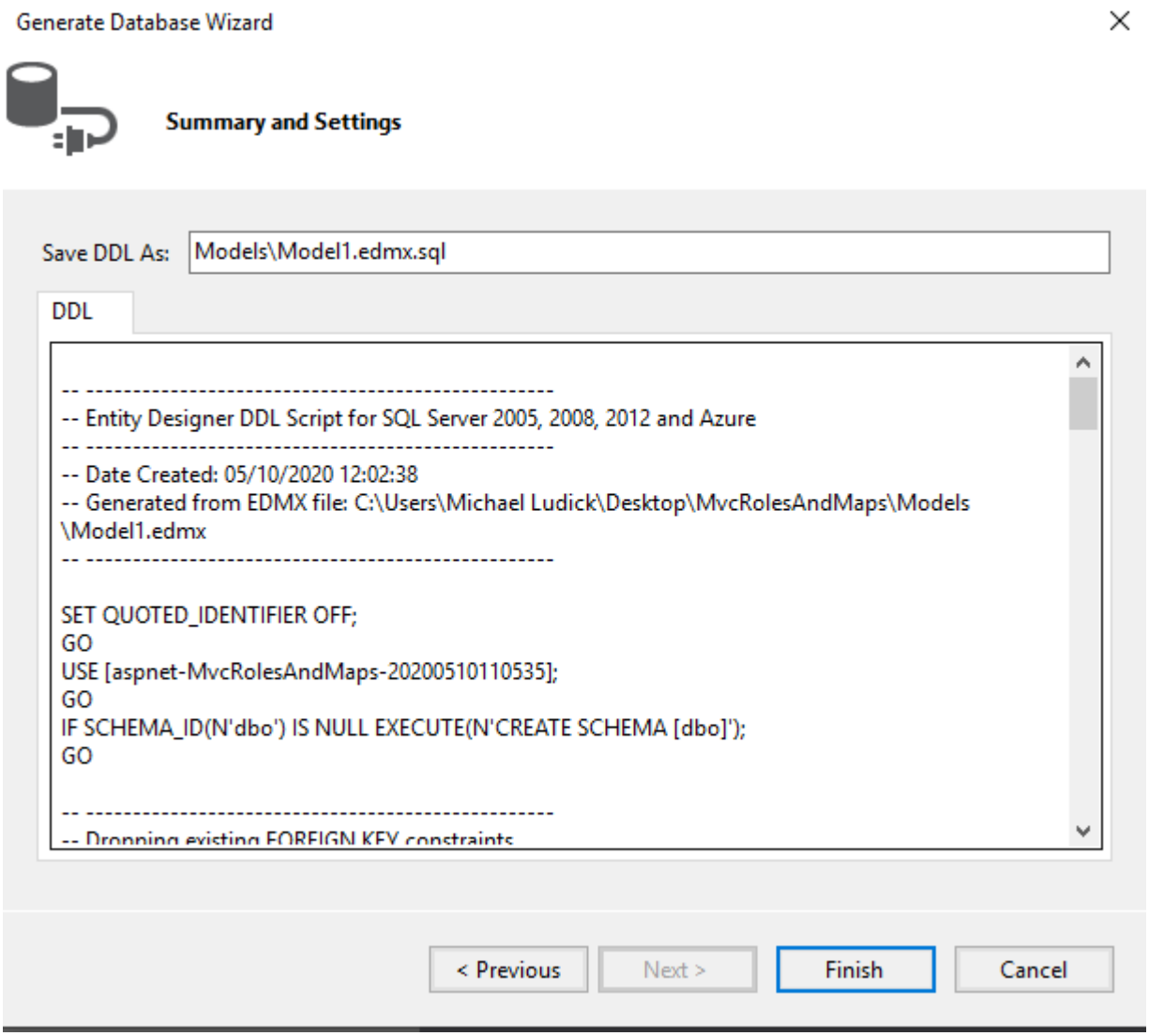

AspNetUser
Properties
Id
Email
EmailConfirmed
PasswordHash
SecurityStamp
PhoneNumber
PhoneNumberC...
TwoFactorEnabled
LockoutEndDate...
LockoutEnabled
AccessFailedCou...
UserName
AspNetRoleId
Navigation Properties
AspNetUserRole
1
1
AspNetUserRole
Properties
Userid
Roleid
Navigation Properties
AspNetUser
AspNetRole


Add Association
Association Name:
AspNetRoleAspNetUserRole
End
Entity:
AspNetRole
Multiplicity:
1 (One)
Navigation Property:
AspNetUserRole
End
Entity:
AspNetUserRole
Multiplicity:
1 (One)
Navigation Property:
AspNetRole
Add foreign key properties to the 'AspNetUserRole' Entity
AspNetRole can have 1 (One) instance of AspNetUserRole. Use AspNetRole.AspNetUserRole to access the AspNetUserRole instance.
AspNetUserRole can have 1 (One) instance of AspNetRole. Use AspNetUserRole.AspNetRole to access the AspNetRole instance.
OK
Cancel


AspNetUserRole
Properties
Userid
Roleid
Navigation Properties
AspNetUser
AspNetRole
1
1
1
AspNetRole
Properties
Id
Name
Navigation Properties
AspNetUserRole


AspNetUserRole
Properties
Userid
Roleid
Navigation Properties
AspNetUser
AspNetRole
1
1
AspNetRole
Properties
Id
Name
Navigation Properties
AspNetUserRole
1


AspNetUser
Properties
Id
Email
EmailConfirmed
PasswordHash
SecurityStamp
PhoneNumber
PhoneNumberC...
TwoFactorEnabled
LockoutEndDate...
LockoutEnabled
AccessFailedCou...
UserName
Navigation Properties
AspNetUserRole...
1
AspNetUserR...
Properties
UserId
RoleId
Navigation Properties
AspNetRole
AspNetUser
*
1
AspNetRole
Properties
Id
Name
Navigation Properties
AspNetUserRol...
1


Generate Database Wizard
Summary and Settings
Save DDL As:
Models\Model1.edmx.sql
DDL
-- ----------
-- Entity Designer DDL Script for SQL Server 2005, 2008, 2012 and Azure
-- ----------
-- Date Created: 05/10/2020 12:02:38
-- Generated from EDMX file: C:\Users\Michael Ludick\Desktop\MvcRolesAndMaps\Models
\Model1.edmx
-- ----------
SET QUOTED_IDENTIFIER OFF;
GO
USE [aspnet-MvcRolesAndMaps-20200510110535];
GO
IF SCHEMA_ID(N'dbo') IS NULL EXECUTE(N'CREATE SCHEMA [dbo]');
GO
-- ----------
-- Dropping existing FOREIGN KEY constraints
< Previous
Next >
Finish
Cancel

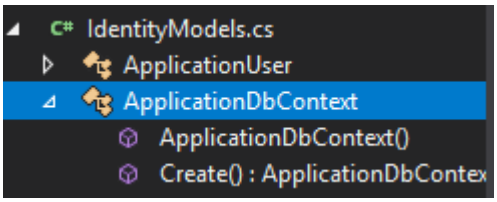
Step3

Sit I only made it like so for viewing purposes

```
<add name="DefaultConnection"
     connectionString="Data Source=(LocalDb)\MSSQLLocalDB;AttachDbFilename=
     |DataDirectory|\aspnet-MvcRolesAndMaps-20200510110535.mdf;
     Initial Catalog=aspnet-MvcRolesAndMaps-20200510110535;
     Integrated Security=True" providerName="System.Data.SqlClient" />
```

```
 <add name="Entities"
      connectionString="metadata=res://*/Models.Model3.csdl|res:
      //*/Models.Model3.ssdl|res://*/Models.Model3.msl;
      provider=System.Data.SqlClient;provider connection string=&quot;
   data source=(LocalDb)
   \MSSQLLocalDB;attachdbfilename=|DataDirectory|
   \aspnet-MvcRolesAndMaps-20200510110535.mdf;
   initial catalog=aspnet-MvcRolesAndMaps-20200510110535;
   integrated security=True;multipleactiveresultsets=True;
   application name=EntityFramework&quot;" providerName="System.Data.EntityClient" />
/connectionStrings>
```

Step 4




```
1   using System.Data.Entity;
2   using System.Security.Claims;
3   using System.Threading.Tasks;
4   using Microsoft.AspNet.Identity;
5   using Microsoft.AspNet.Identity.EntityFramework;
6
7   namespace MvcRolesAndMaps.Models
8   {
```

```
namespace MvcRolesAndMaps.Models
{
    // You can add profile data for the user by adding more properties to your ApplicationUser class, please visit https:
    14 references
    public class ApplicationUser : IdentityUser
    {
        2 references
        public async Task<ClaimsIdentity> GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)
        {
            // Note the authenticationType must match the one defined in CookieAuthenticationOptions.AuthenticationType
            var userIdentity = await manager.CreateIdentityAsync(this, DefaultAuthenticationTypes.ApplicationCookie);
            // Add custom user claims here
            return userIdentity;
        }
```

```
    5 references
    public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {
        1 reference
        public ApplicationDbContext()
            : base("DefaultConnection", throwIfV1Schema: false)
        {
        }

        1 reference
        public static ApplicationDbContext Create()
        {
            return new ApplicationDbContext();
        }
    }
}
```

Step 5

```
using Microsoft.Owin;
using Owin;
using MvcRolesAndMaps.Models;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;
using System.Threading.Tasks;
```

```
[assembly: OwinStartupAttribute(typeof(MvcRolesAndMaps.Startup))]
namespace MvcRolesAndMaps
{
    2 references
    public partial class Startup
```

```
public partial class Startup
{
    private readonly string Rolename = "Admin";
    0 references
```

```
public void Configuration(IAppBuilder app)
{
    CreateDefaultRoleAndUser();
    ConfigureAuth(app);
}
```

```
0 references
public async Task CreateDefaultRoleAndUser()
{
    ApplicationDbContext context = new ApplicationDbContext();

    var rolemanager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(context));
    var usermanger = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(context));
```

```
if (!rolemanager.RoleExists(Rolename))
{
    // first we create the Admin role
    var role = new Microsoft.AspNet.Identity.EntityFramework.IdentityRole();
    role.Name = Rolename;
    rolemanager.Create(role);
```

Can be simplified

```
if (!rolemanager.RoleExists(Rolename))
{
    // first we create the Admin role
    var role = new IdentityRole();
    role.Name = Rolename;
    rolemanager.Create(role);
```

```
// Here we create a user and grant him/her the Admin role.
// The default behavior when creating users is a bit weird.
// When users register, only their emails and passwords are required,
// but these emails are stored in the database as both emails and usernames.
// During login, the emails are again requested, but then compared to the
// username in the DB. This behavior can be changed, but for now we will
// just go with the flow.
```

```
string myEmail = "me@gmail.com";
string myPassword = "Ctu@2020";
var user = new ApplicationUser { UserName = myEmail, Email = myEmail };
var result = await usermanger.CreateAsync(user, myPassword);
if (result.Succeeded)
{
    var result1 = usermanger.AddToRole(user.Id, Rolename);
}
```

Step 6

AspNetRoles

	Id	Name
▷	bb8-cf70b40576da	Admin
◇	*NULL*	*NULL*

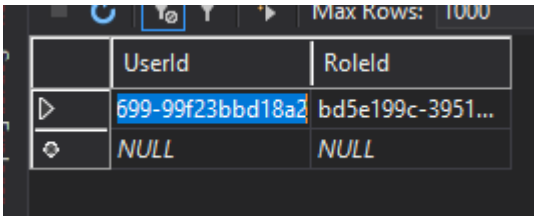
AspNetUser

	Id	Email	EmailConfirmed	PasswordHash	SecurityStamp
▷	699-99f23bbd18a2	me@gmail.com	False	AB00v5cDhicb/...	0421b37b-a483...
◇	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

SecurityStamp	PhoneNumber	PhoneNumber...	TwoFactorEna...
0421b37b-a483...	*NULL*	False	False
NULL	*NULL*	*NULL*	*NULL*

TwoFactorEna...	LockoutEndDa...	LockoutEnabled	AccessFailedC...	UserName
False	*NULL*	False	0	me@gmail.com
NULL	*NULL*	*NULL*	*NULL*	*NULL*

AspNetUSerRoles



Log in

Application name

Log in.

Use a local account to log in.

Email

me@gmail.com

Password

Remember me?

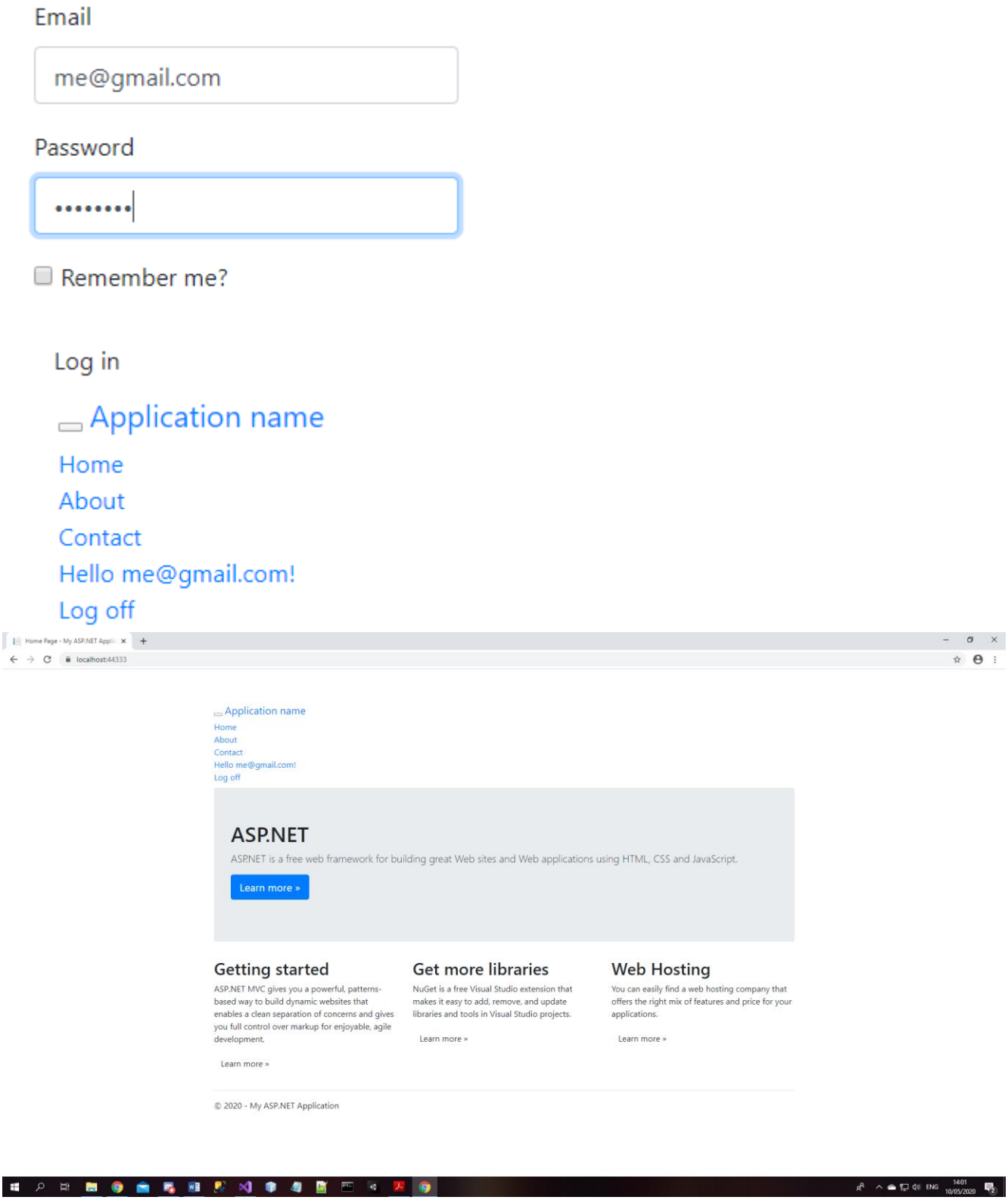
Log in

Register as a new user

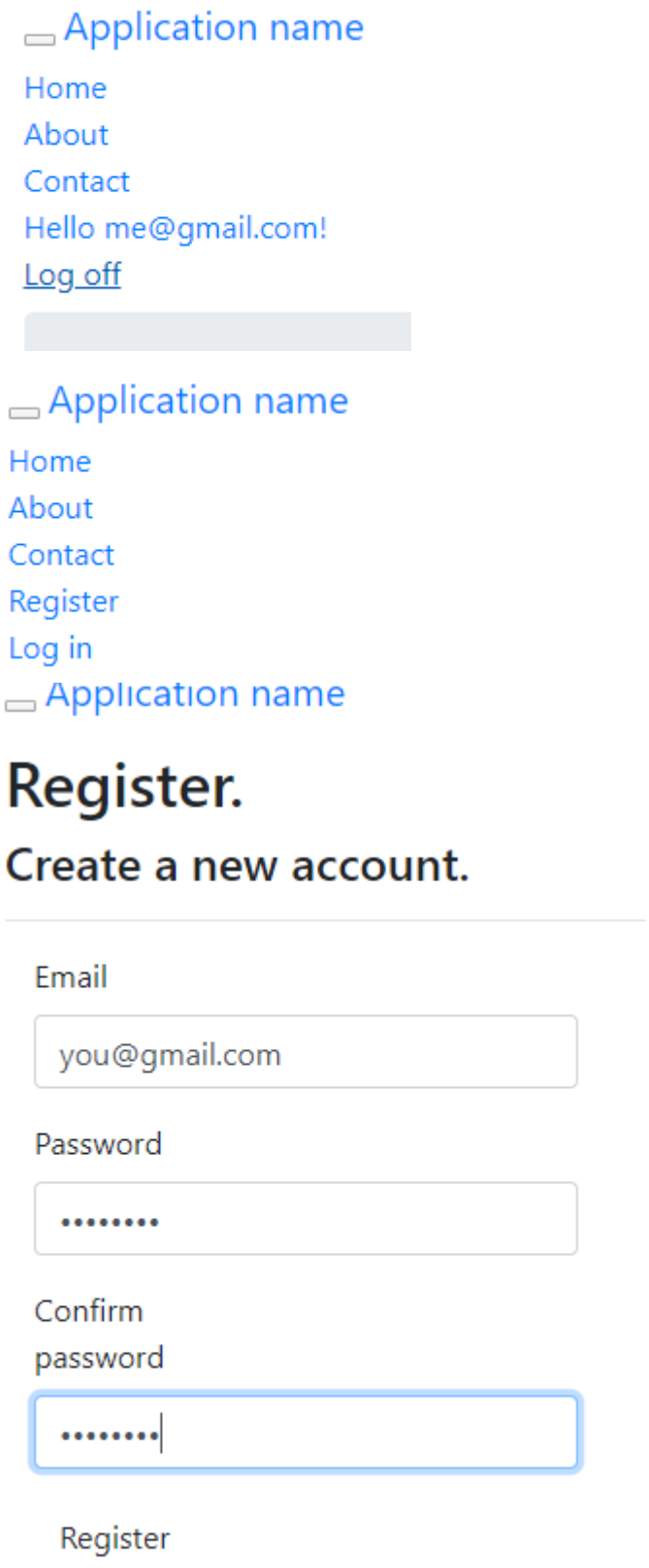
Use another service to log in.

There are no external authentication services configured. See this article for details on setting up this ASP.NET application to support logging in via external services.

© 2020 - My ASP.NET Application


Email
me@gmail.com
Password
Remember me?
Log in
Application name
Home
About
Contact
Hello me@gmail.com!
Log off
Home Page - My ASP.NET Applic
localhost:44333
Application name
Home
About
Contact
Hello me@gmail.com!
Log off
ASP.NET
ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.
Learn more »
Getting started
ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and gives you full control over markup for enjoyable, agile development.
Learn more »
Get more libraries
NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects.
Learn more »
Web Hosting
You can easily find a web hosting company that offers the right mix of features and price for your applications.
Learn more »
© 2020 - My ASP.NET Application
ENG
14:01
10/05/2020

Step 7


Application name
Home
About
Contact
Hello me@gmail.com!
Log off
Application name
Home
About
Contact
Register
Log in
Application name
Register.
Create a new account.
Email
you@gmail.com
Password
Confirm
password
Register

Application name

Home

About

Contact

Hello you@gmail.com!

Log off

AspNetUsers

	Id	Email	EmailConfirmed	PasswordHash	SecurityStamp
▷	699-99f23bbd18a2	me@gmail.com	False	AB00v5cDhicb/...	0421b37b-a483...
	e83969bd-1cf8-...	you@gmail.com	False	AHwYMKZ1z1P...	80ca5a62-8ede-...
◇	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

SecurityStamp	PhoneNumber	PhoneNumber...	TwoFactorEna...
0421b37b-a483...	*NULL*	False	False
80ca5a62-8ede-...	*NULL*	False	False
NULL	*NULL*	*NULL*	*NULL*

TwoFactorEna...	LockoutEndDa...	LockoutEnabled	AccessFailedC...	UserName
False	*NULL*	False	0	me@gmail.com
False	*NULL*	True	0	you@gmail.com
NULL	*NULL*	*NULL*	*NULL*	*NULL*

AspNetRoles

Refresh (Shift+Alt+R)

	me
▷	bb8-cf70b40576da	Admin
◇	*NULL*	*NULL*

AspNetUsersRoles

Max Rows: 1000

	UserId	RoleId
▷	699-99f23bbd18a2	bd5e199c-3951...
◇	*NULL*	*NULL*

Step 8

Controller

```
using System.Linq;
using System.Web.Mvc;
using Microsoft.AspNet.Identity.EntityFramework;
using MvcRolesAndMaps.Models;

namespace MvcRolesAndMaps.Controllers
```

```
namespace MvcRolesAndMaps.Controllers
{
    [Authorize(Roles ="Admin")]
    0 references
    public class RolesController : Controller
```

```
ApplicationDbContext db = new ApplicationDbContext();
0 references
public ActionResult Index()
{
    var Roles = db.Roles.ToList();
    return View(Roles);
}
// GET: Create
```

```
// GET: Create
0 references
public ActionResult Create()
{
    var Role = new IdentityRole();
    return View(Role);
}
// POST: Create
[HttpPost]
```

```
// POST: Create
[HttpPost]
0 references
public ActionResult Create(IdentityRole Role)
{
    db.Roles.Add(Role);
    db.SaveChanges();
    return RedirectToAction("Index");
}
```

Index

```
@model IEnumerable<Microsoft.AspNet.Identity.EntityFramework.IdentityRole>
@{
    ViewBag.Title = "Roles";
}

<h2>Roles</h2>
<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
```

```
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Name)
        </th>
        <th></th>
    </tr>
    @foreach(var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Name)
            </td>
        </tr>
    }
</table>
```

Create

```
@model Microsoft.AspNet.Identity.EntityFramework.IdentityRole
@{
    ViewBag.Title = "Create";
}
```

```
<h2>Create</h2>

<h2>Add New user Role</h2>

@using (Html.BeginForm())
{
    <p>Enter User Role Name:</p>

    @Html.EditorFor(m=>m.Name)
    <input type="submit" value="create Role" />
}
```

Step 9

USerRilesController

cRolesAndMaps

```
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;
using System.Threading.Tasks;
using System.Net;
using MvcRolesAndMaps.Models;
```

```
namespace MvcRolesAndMaps.Controllers
{
    [Authorize(Roles = "Admin")]
    0 references
    public class UserRolesController : Controller
    {
        ApplicationDbContext db = new ApplicationDbContext();
        // GET: UserRoles
```

```
    0 references
    public ActionResult Index()
    {
        // Get the users from the identity table
        var User = db.Users.ToList();
        return View();
    }
```

```
    }
    // GET:
    0 references
    public ActionResult EdituserRoles(string userId)
    {
        if (userId == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
```

```
0 references
public ActionResult EdituserRoles(string userId)
{
    if (userId == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    // Get a list of all the available roles - to display in the dropdown
    var roles = db.Roles.ToList();
    // We want to highlight the user's existing roles (if any) in the dropdown
    // Get all the names of the user's roles using the received user's id
```

```
// We want to highlight the user's existing roles (if any) in the dropdown
// Get all the names of the user's roles using the received user's id
var UserManger = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(db));
List<string> userRolesNameList = UserManger.GetRoles(userId).ToList();
// Use the names to get the complete role objects (including their IDs) of the users roles
var RoleManger = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(db));
List<IdentityRole> userRolesList = RoleManger.Roles.Where(r =>userRolesNameList.Contains(r.Name)).ToList();
// Create an array containing the user's existing roles' IDs (if any)
string[] userRolesIdsArray = new string[userRolesList.Count];
int index = 0;
```

```
// Get all the names of the user's roles using the received us
var UserManger = new UserManager<ApplicationUser>
    (new UserStore<ApplicationUser>(db));
List<string> userRolesNameList =
    UserManger.GetRoles(userId).ToList();
// Use the names to get the complete role objects (including t
var RoleManger = new RoleManager<IdentityRole>
    (new RoleStore<IdentityRole>(db));
List<IdentityRole> userRolesList = RoleManger.Roles.Where
    (r =>userRolesNameList.Contains(r.Name)).ToList();
// Create an array containing the user's existing roles' IDs (
string[] userRolesIdsArray = new string[userRolesList.Count];
int index = 0;
```

```
foreach(IdentityRole  role in userRolesList)
{
    userRolesIdsArray[index] = role.Id;
    index++;
}
// Add the roles to the viewbag, since it is treated as "additional data"
// (it is not the user's data normally associated with a controller action
// like this)
ViewBag.Roles = new MultiSelectList(roles, "Id", "Name", userRolesIdsArray);

return View(User);
}
```

```
// POST:
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<ActionResult> EdituserRoles(FormCollection collection)
{
    var UserManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(db));
    var RoleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(db));
    // Because the form contains data from two different models (user
    // and roles), we use a FormCollection to extract the data manually.
    // Alternatively, we could have created a special viewmodel just for
    // this purpose - which would also enable easier validation.
```

```
0 references
public async Task<ActionResult> EdituserRoles(FormCollection collection)
{
    var UserManager = new UserManager<ApplicationUser>
        (new UserStore<ApplicationUser>(db));
    var RoleManager = new RoleManager<IdentityRole>
        (new RoleStore<IdentityRole>(db));
    // Because the form contains data from two different models (user
    // and roles), we use a FormCollection to extract the data manually.
    // Alternatively, we could have created a special viewmodel just for
    // this purpose - which would also enable easier validation.
```

```
    // Extract the user's id
    string UserId = collection["id"];
    if(UserId == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    // Get the user
    Microsoft.AspNet.Identity.EntityFramework.IdentityUser user = db.Users.Find(UserId);
    if(user == null)
    {
        return HttpNotFound();
    }
```

```
// Extract the Role IDs from the collection (in a single
// text string - separated by commas)
string RoleIdString = collection["RoleId"];
if(RoleIdString != null)
{
    // Remove all of the user's existing Roles
    var userRoles = await UserManager.GetRolesAsync(UserId);
    await UserManager.RemoveFromRolesAsync(UserId, userRoles.ToArray());
    // Add the user's new Roles
    // Extract the roles from the text string into an array (by commas)
    string[] RolesIdArray = RoleIdString.Split(',');
    foreach(string roleId in RolesIdArray)
```

```
foreach(string roleId in RolesIdArray)
{
    // Retrieve the role, because you need its Name below
    IdentityRole role = RoleManager.FindById(roleId);
    // Add the user to the role
    UserManager.AddToRole(UserId, role.Name);
}
```

```
/* Repeat the processes used in the "GET" EditUserRoles action
  to display the user's new Roles */
// Get a list of all the available roles - to display in the dropdown
var roles = db.Roles.ToList();
// We want to highlight the user's existing roles (if any) in the dropdown
// Get all the names of the user's roles using the received user's id
List<string> UserRolesNamesList = UserManager.GetRoles(UserId).ToList();
// Use the names to get the complete role objects (including their IDs) of the users roles
List<IdentityRole> userRolesList = RoleManager.Roles.Where(r => UserRolesNamesList.Contains(r.Name)).ToList();
// Create an array containing the user's existing roles' IDs (if any)
string[] userRolesIdArray = new string[userRolesList.Count];
int index = 0;
```

```
/* Repeat the processes used in the "GET" EditUserR
  to display the user's new Roles */
// Get a list of all the available roles - to displ
var roles = db.Roles.ToList();
// We want to highlight the user's existing roles (
// Get all the names of the user's roles using the
List<string> UserRolesNamesList =
    UserManager.GetRoles(UserId).ToList();
// Use the names to get the complete role objects (
List<IdentityRole> userRolesList =
    RoleManager.Roles.Where(r =>
    UserRolesNamesList.Contains(r.Name)).ToList();
// Create an array containing the user's existing r
string[] userRolesIdArray =
    new string[userRolesList.Count];
int index = 0;
```

```
foreach(IdentityRole Role in userRolesList)
{
    userRolesIdArray[index] = Role.Id;
    index++;
}

ViewBag.Roles = new MultiSelectList(roles, "Id", "Name", userRolesIdArray);

return View(user);
```

Index

```
@model IEnumerable<Microsoft.AspNet.Identity.EntityFramework.IdentityUser>
@{
    ViewBag.Title = "Users";
}

<h2>Users</h2>
<table class="table">
```

```
<h2>Users</h2>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(Model => Model.UserName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Email)
        </th>
        <th></th>
    </tr>
    @foreach(var item in Model)
```

```
@foreach(var item in Model)
{
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.UserName)
        </td>
        <td>
            @Html.DisplayFor(modelItem=>item.Email)
        </td>
        <td>
            @Html.ActionLink("Edit Roles", "EditRoles", new { userId =item.Id})
        </td>
    </tr>
}
table>
```

```
        </t
    }
</table>
```

EditUSerRoles

```
@model Microsoft.AspNet.Identity.EntityFramework.IdentityUser
@{
    ViewBag.Title = "EditUserRoles";
}

<h2>EditUserRoles</h2>
```

```
<h2>EditUserRoles</h2>
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    <div class="form-horizontal">
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(Model=>Model.Id)
```

```
<div class="form-group">
    @Html.LabelFor(Model=>Model.UserName,htmlAttributes:new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @*
            Since we are not editing the username as well, we want to disable it.
            Replace the usual @Html.EditorFor() with @Html.TextBoxFor() and disable it
        *@
        @Html.TextBoxFor(Model => Model.UserName, new { disabled="disabled"})
    </div>
```

```
<div class="form-group">
    @Html.LabelFor(Model=>Model.UserName,
    htmlAttributes:new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @*
            Since we are not editing the username as well, we
            Replace the usual @Html.EditorFor() with @Html.Te
        *@
        @Html.TextBoxFor(Model =>
    Model.UserName, new { disabled="disabled"})
```

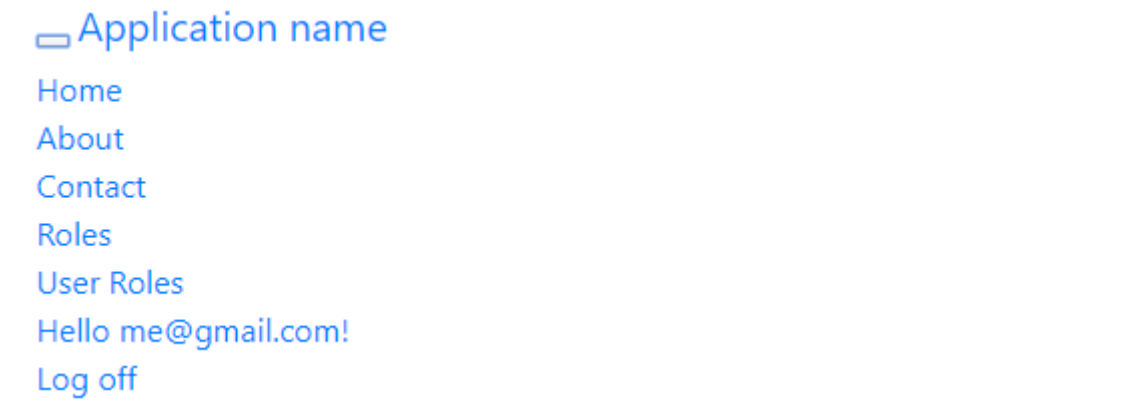
```
    </div>
</div>
<br/>
<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        @Html.Label("Roles (Press ctrl to select multiple rows)")
    </div>
    <div class=" col-md-offset-2 col-md-10">
        @Html.DropDownList("Roleid", (MultiSelectList)ViewBag.Roles, new { multiple ="multiple",size=8})
    </div>
</div>
<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Save" class="btn btn-default"/>
        </div>
```

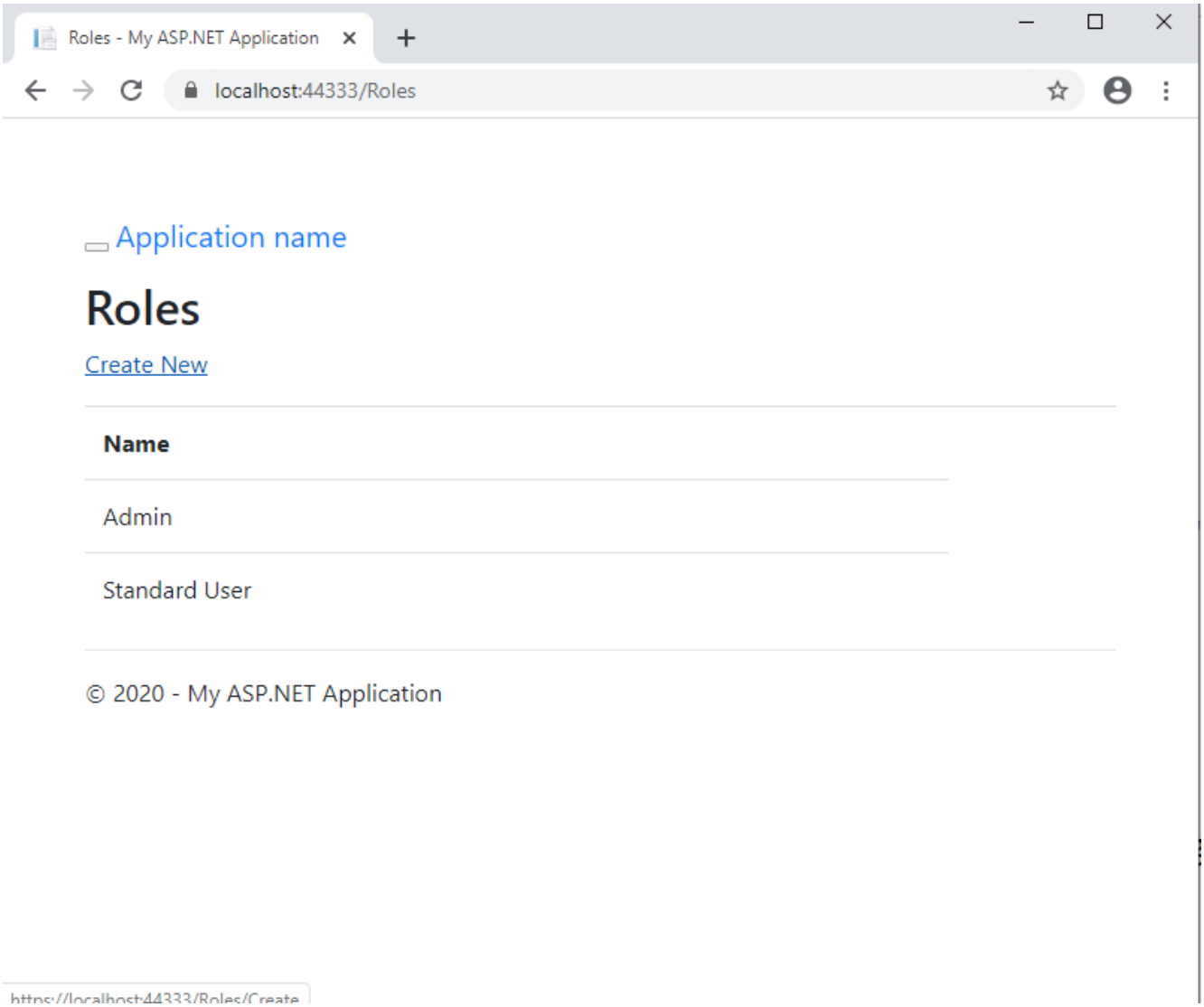
```
    </div>
</div>
<br/>
<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        @Html.Label("Roles " +
        "(Press ctrl to select multiple rows)")
    </div>
    <div class=" col-md-offset-2 col-md-10">
        @Html.DropDownList("Roleid",
        (MultiSelectList)ViewBag.Roles, new { multiple ="multiple",size=8})
    </div>
</div>
<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Save"
                   class="btn btn-default"/>
```

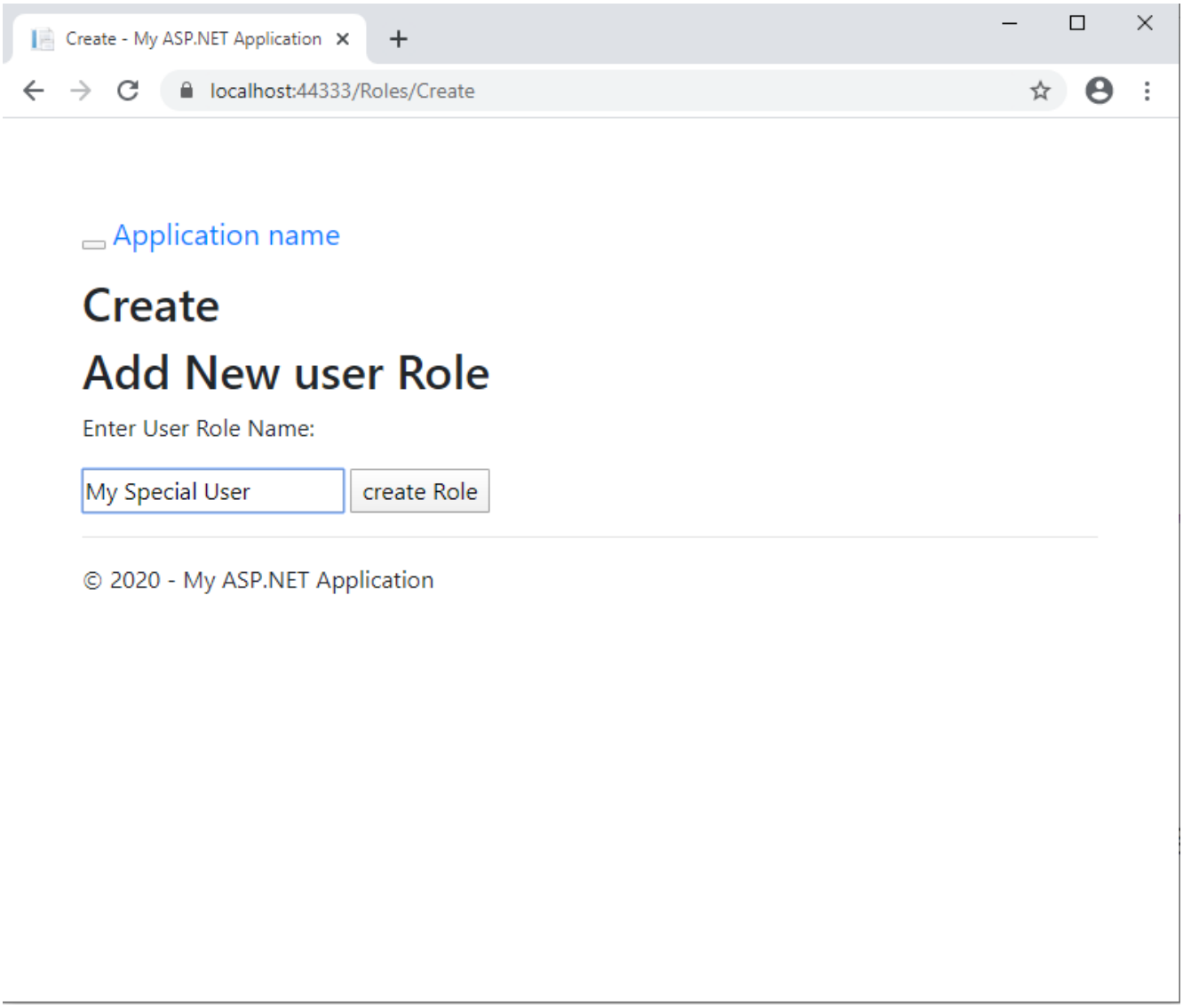
```
    </div>
</div>
<br/>
<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        @Html.Label("Roles " +
        "(Press ctrl to select multiple rows)")
    </div>
    <div class=" col-md-offset-2 col-md-10">
        @Html.DropDownList("Roleid",
        (MultiSelectList)ViewBag.Roles,
        new { multiple ="multiple",size=8})
    </div>
</div>
<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Save"
                   class="btn btn-default"/>
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save"
                       class="btn btn-default"/>
            </div>
        </div>
    </div>
</div>
```

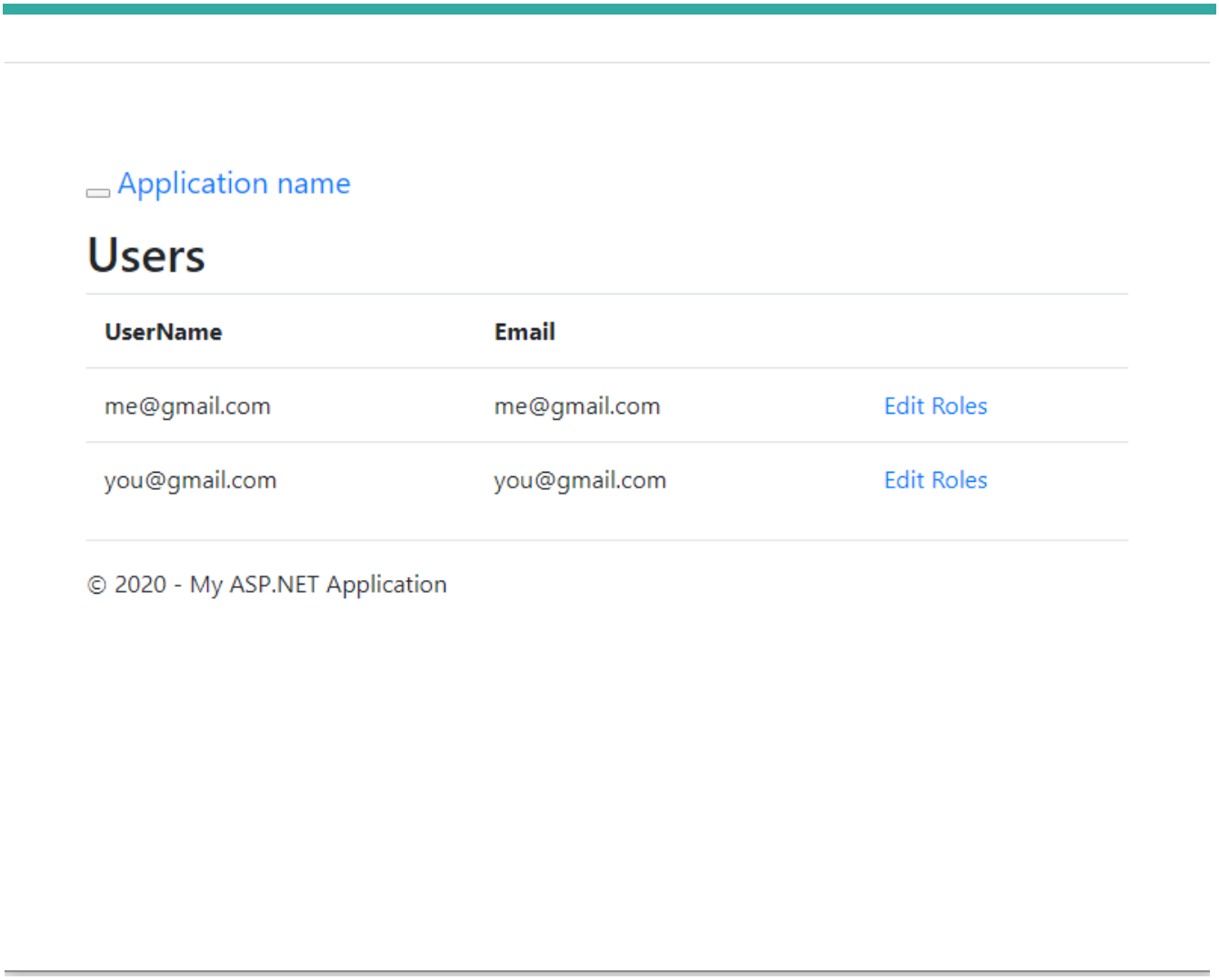
Step 10

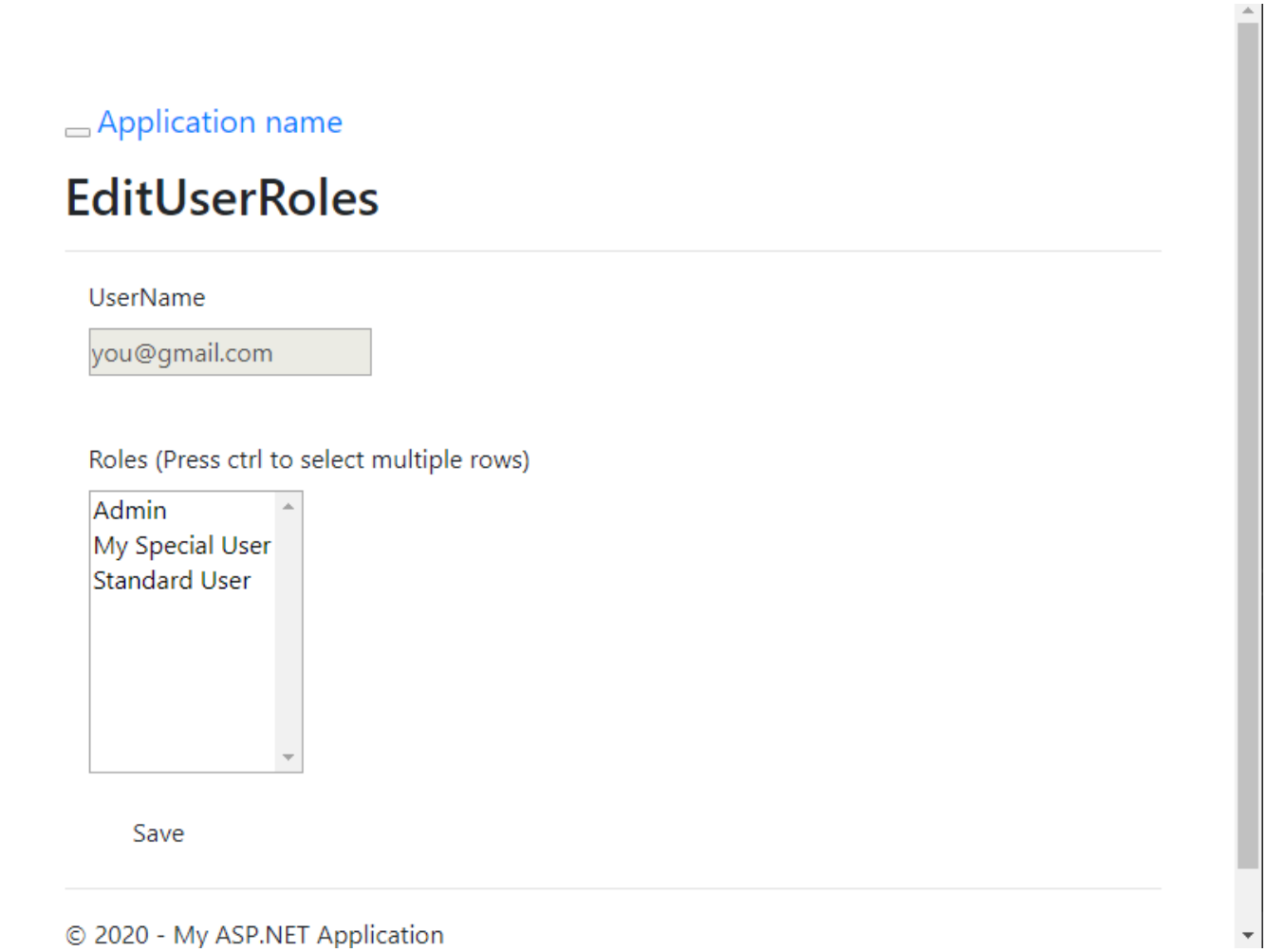
```
<div class="navbar-collapse collapse">
    <ul class="nav navbar-nav">
        <li>@Html.ActionLink("Home", "Index", "Home")</li>
        <li>@Html.ActionLink("About", "About", "Home")</li>
        <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
        @if (User.IsInRole("Admin"))
        {
            <li>@Html.ActionLink("Roles","Index","Roles")</li>
            <li>@Html.ActionLink("User Roles","Index","UserRoles")</li>
        }
    </ul>
```

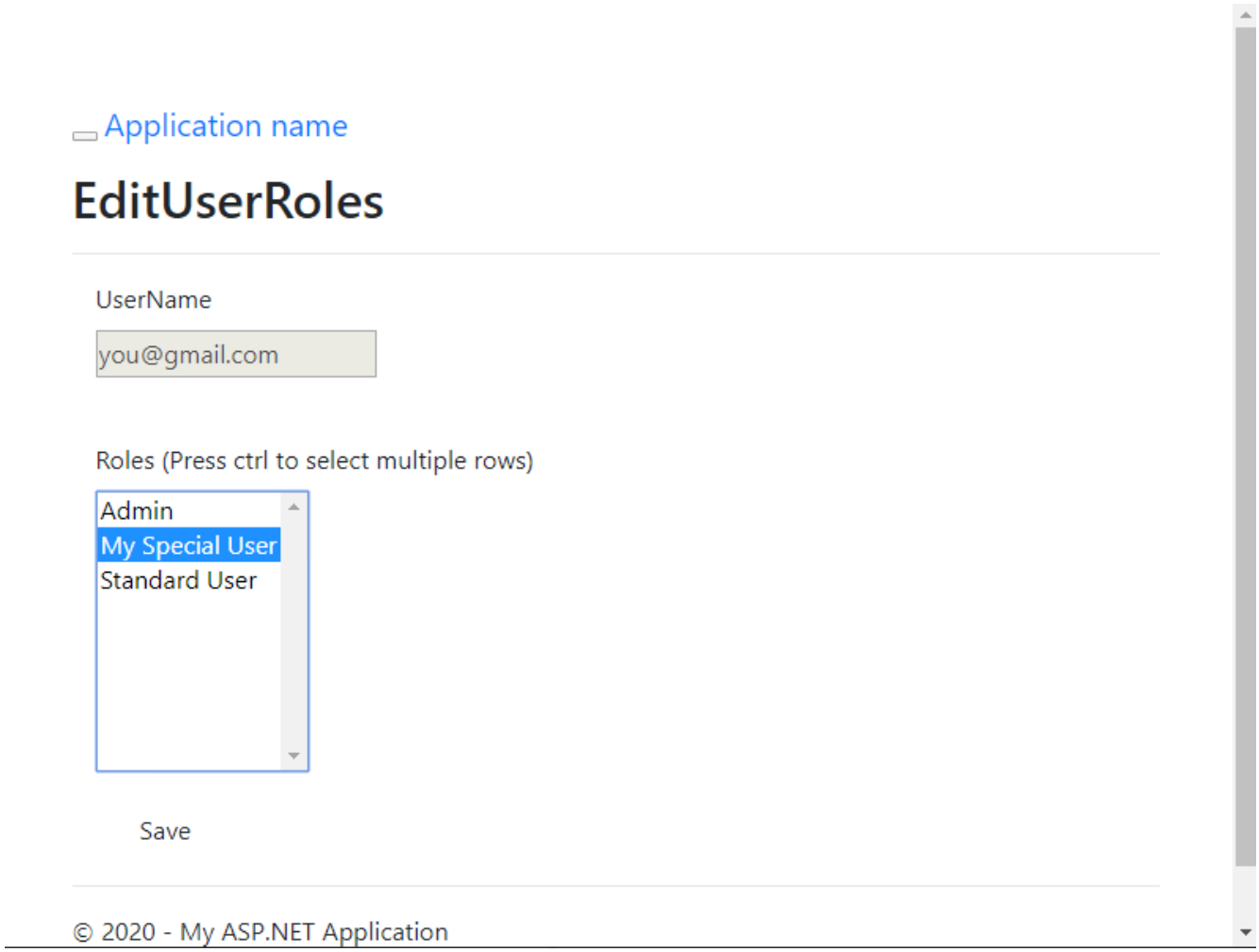


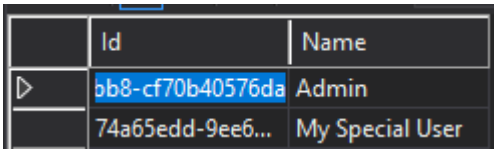

Roles - My ASP.NET Application
localhost:44333/Roles
Application name
Roles
Create New
Name
Admin
Standard User
© 2020 - My ASP.NET Application

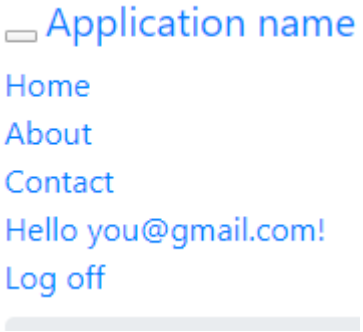

Create - My ASP.NET Application
localhost:44333/Roles/Create
Application name
Create
Add New user Role
Enter User Role Name:
My Special User
create Role
© 2020 - My ASP.NET Application

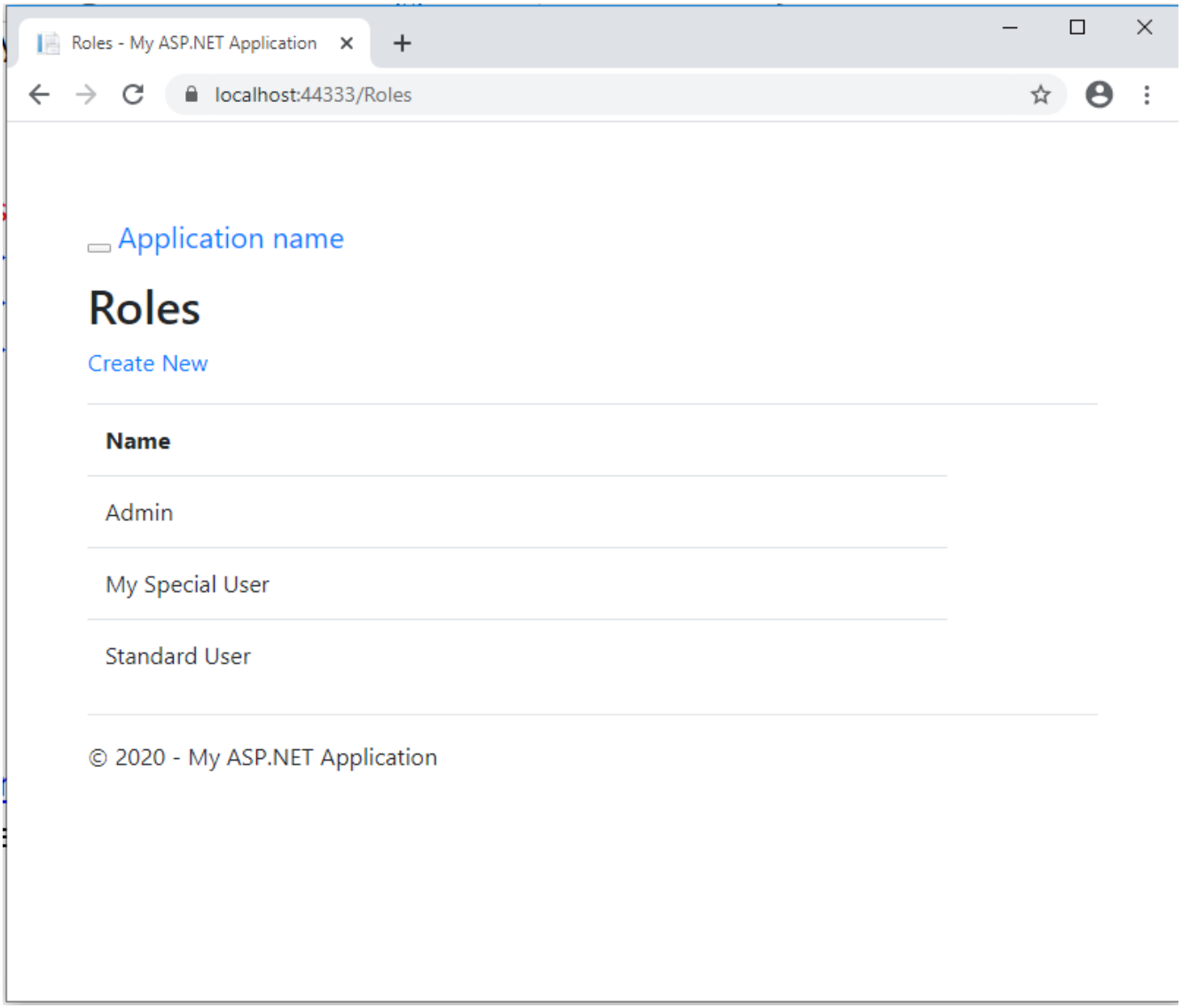

Application name
Users
UserName
Email
me@gmail.com
me@gmail.com
Edit Roles
you@gmail.com
you@gmail.com
Edit Roles
© 2020 - My ASP.NET Application


Application name
EditUserRoles
UserName
you@gmail.com
Roles (Press ctrl to select multiple rows)
Admin
My Special User
Standard User
Save
© 2020 - My ASP.NET Application

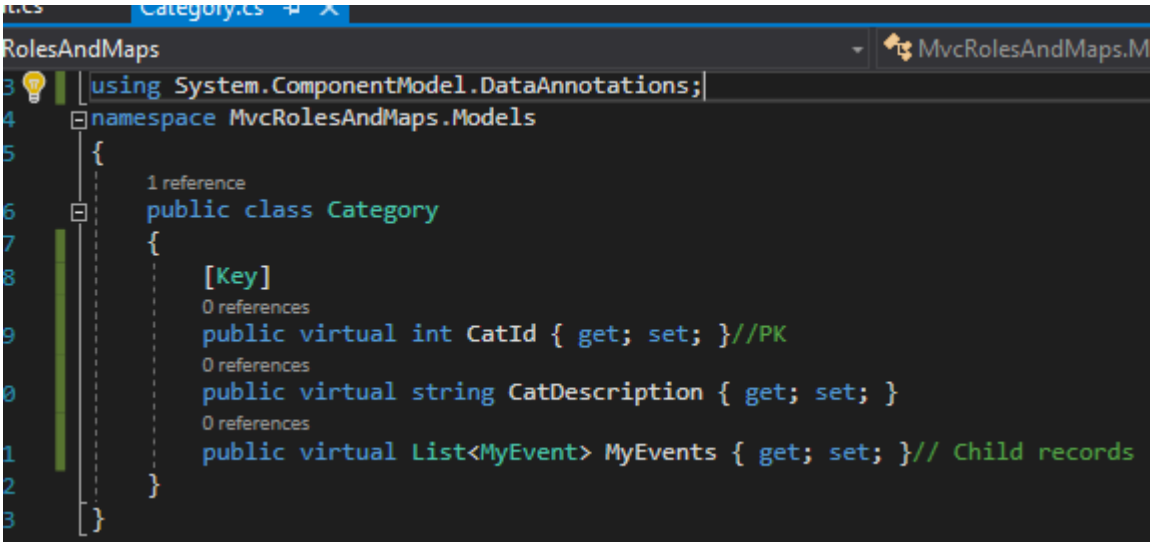

Application name
EditUserRoles
UserName
you@gmail.com
Roles (Press ctrl to select multiple rows)
Admin
My Special User
Standard User
Save
© 2020 - My ASP.NET Application


Id
Name
bb8-cf70b40576da
Admin
74a65edd-9ee6...
My Special User


Application name
Home
About
Contact
Hello you@gmail.com!
Log off

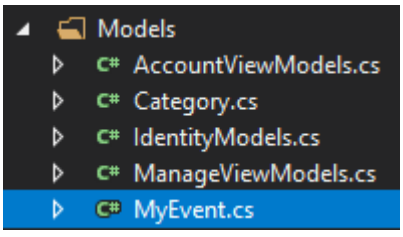

Roles - My ASP.NET Application
localhost:44333/Roles
Application name
Roles
Create New
Name
Admin
My Special User
Standard User
© 2020 - My ASP.NET Application

Step 11

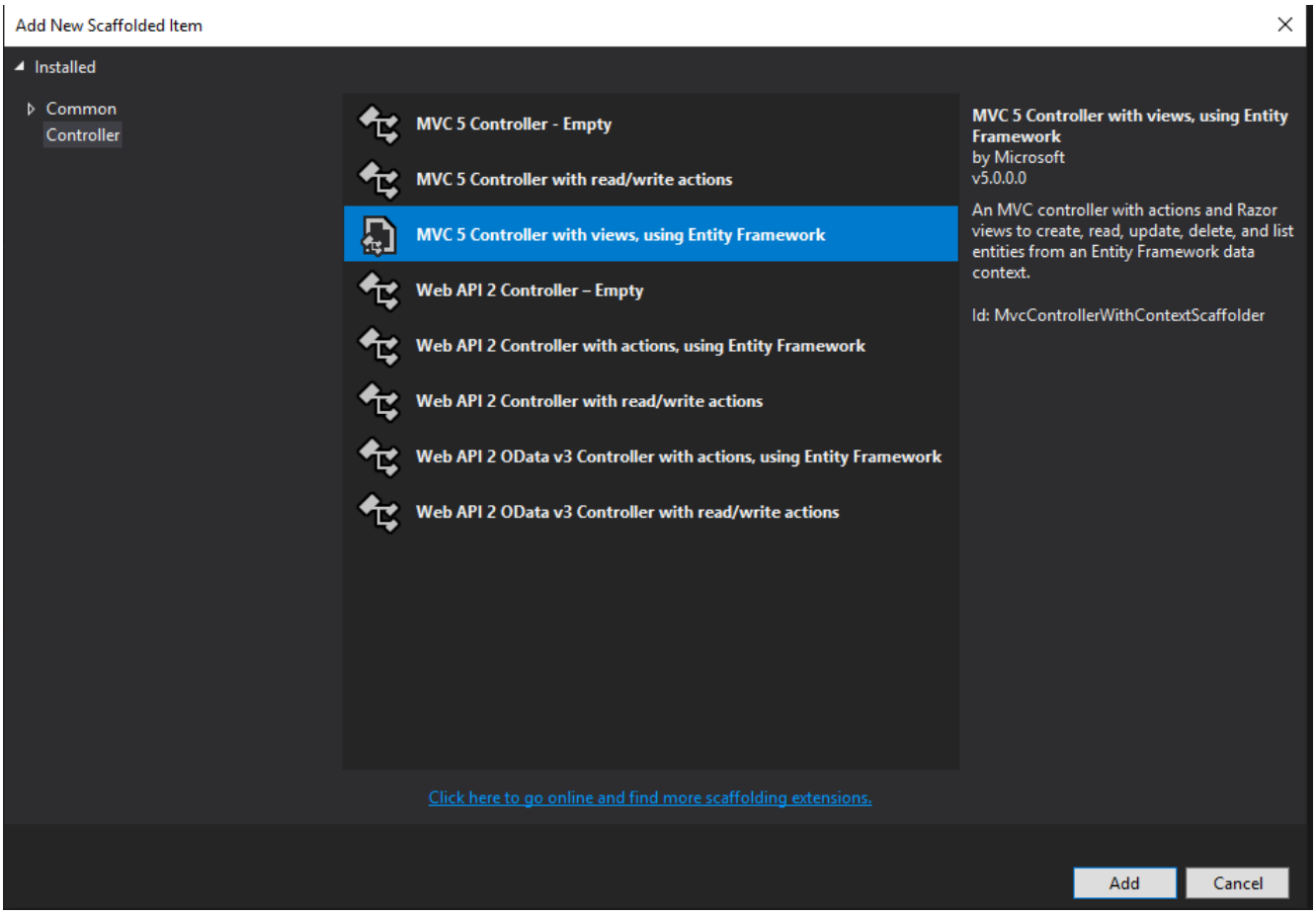

RolesAndMaps
MvcRolesAndMaps.M
using System.ComponentModel.DataAnnotations;
namespace MvcRolesAndMaps.Models
{
1 reference
public class Category
{
[Key]
0 references
public virtual int CatId { get; set; }//PK
0 references
public virtual string CatDescription { get; set; }
0 references
public virtual List<MyEvent> MyEvents { get; set; }// Child records
}
}

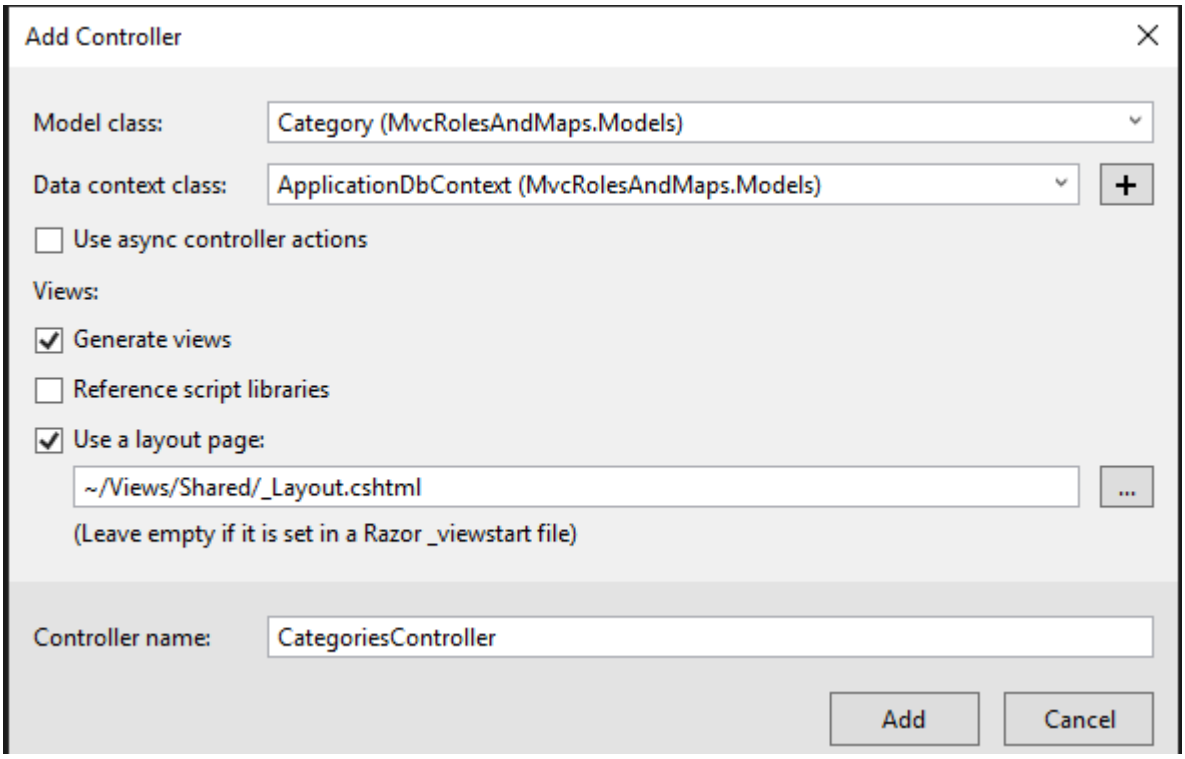
```
using System;
using System.ComponentModel.DataAnnotations;
namespace MvcRolesAndMaps.Models
{
    1 reference
    public class MyEvent
    {
        [Key]
        0 references
        public virtual int EventId { get; set; } // PK
        0 references
        public virtual string UserId { get; set; } // User that created the record
        0 references
        public virtual string EventName { get; set; }
        0 references
        public virtual float EventLat { get; set; } // latitude
        0 references
        public virtual float EventLong { get; set; } // longitude
        0 references
        public virtual string EventURL { get; set; }
        0 references
        public virtual DateTime EventDate { get; set; }
        0 references
        public virtual int CatId { get; set; } // FK to parent (Category)
        0 references
        public virtual Category EventCategory { get; set; } //Parent record
    }
}
```

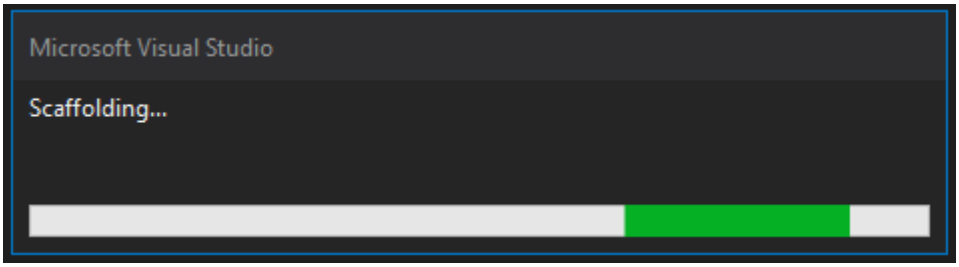
```
[Key]
0 references
public virtual int EventId { get; set; } // PK
0 references
public virtual string UserId { get; set; } // User that created the record
0 references
public virtual string EventName { get; set; }
0 references
public virtual float EventLat { get; set; } // latitude
0 references
public virtual float EventLong { get; set; } // longitude
0 references
public virtual string EventURL { get; set; }
0 references
public virtual DateTime EventDate { get; set; }
0 references
public virtual int CatId { get; set; } // FK to parent (Category)
0 references
public virtual Category EventCategory { get; set; } //Parent record
```

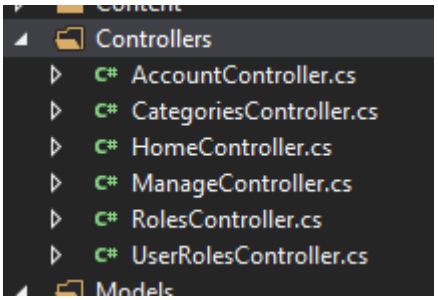


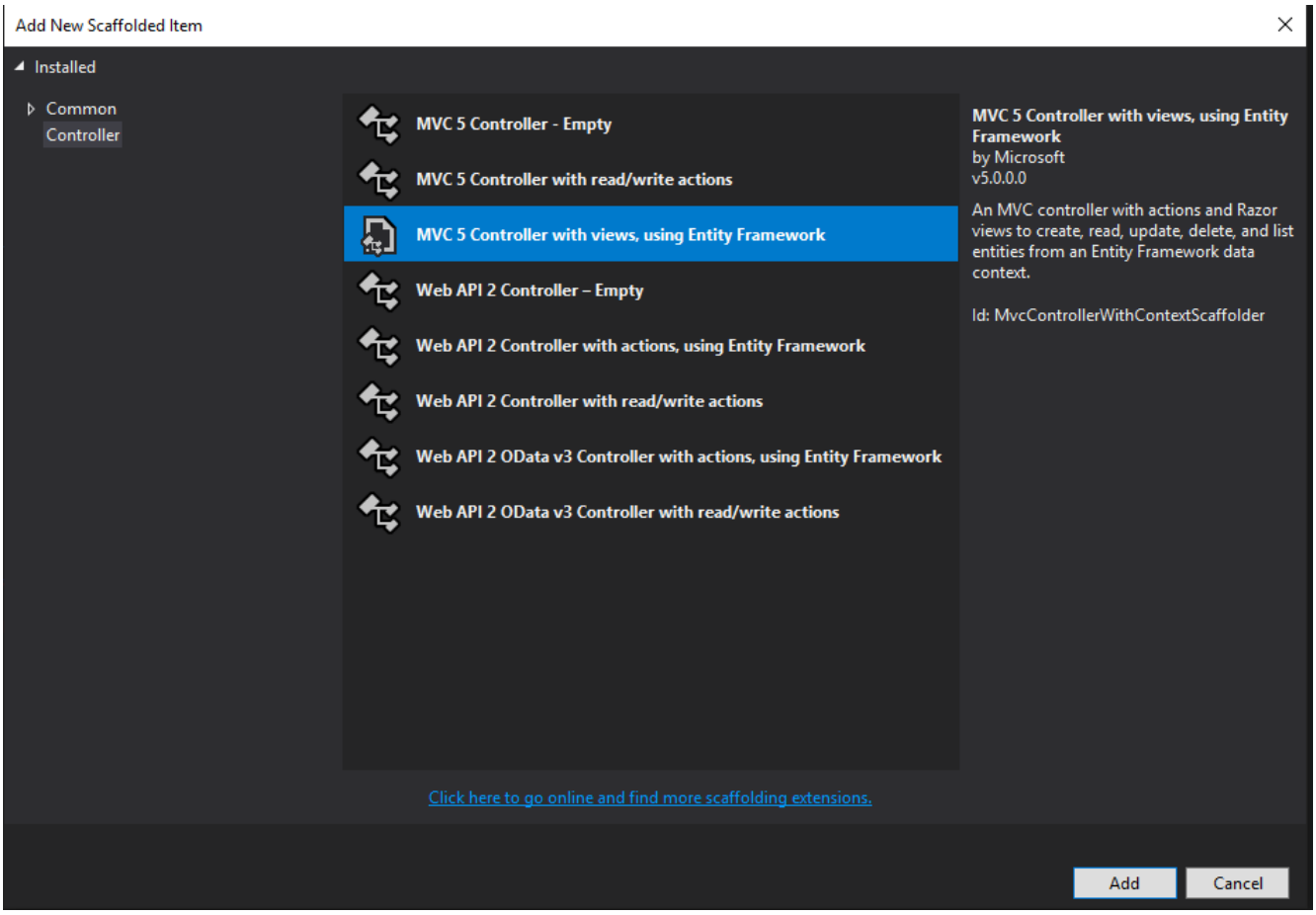
Step 12

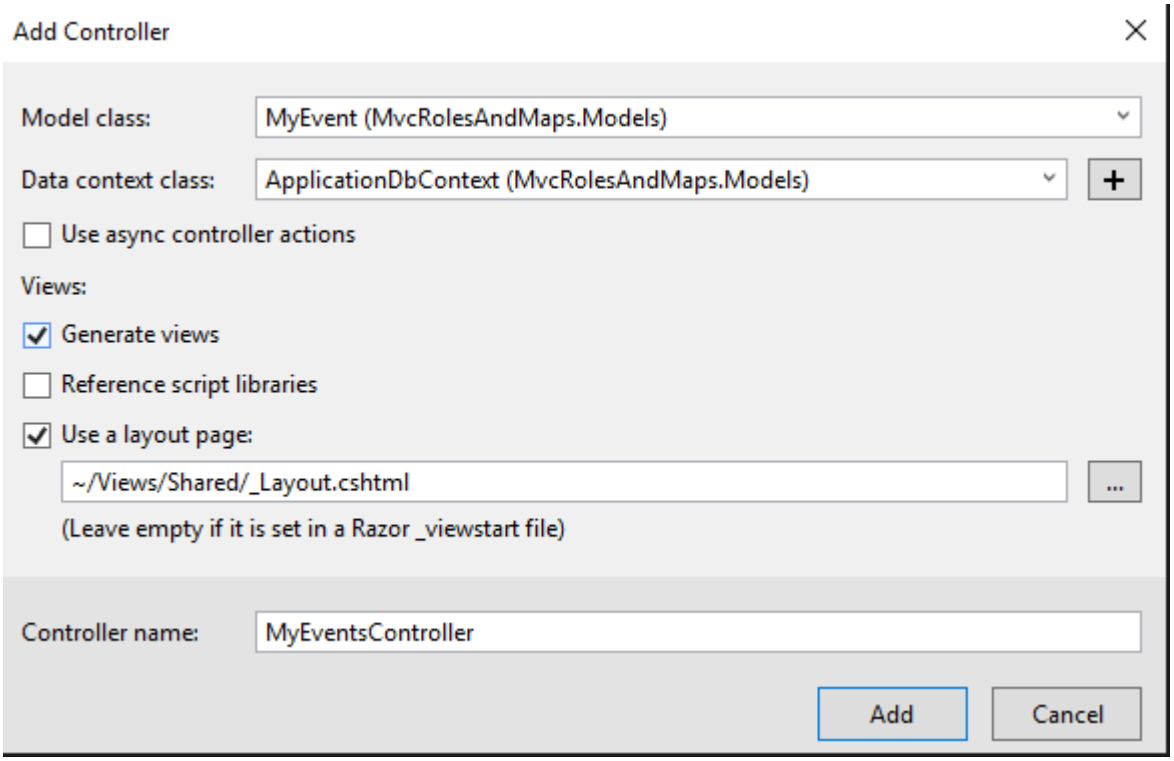

Add New Scaffolded Item
Installed
Common
Controller
MVC 5 Controller - Empty
MVC 5 Controller with read/write actions
MVC 5 Controller with views, using Entity Framework
Web API 2 Controller – Empty
Web API 2 Controller with actions, using Entity Framework
Web API 2 Controller with read/write actions
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 views, using Entity Framework
by Microsoft
v5.0.0.0
An MVC controller with actions and Razor views to create, read, update, delete, and list entities from an Entity Framework data context.
Id: MvcControllerWithContextScaffolder
Click here to go online and find more scaffolding extensions.
Add
Cancel

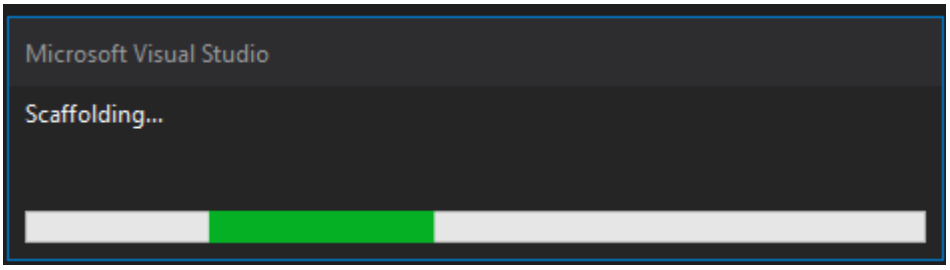

Add Controller
Model class:
Category (MvcRolesAndMaps.Models)
Data context class:
ApplicationDbContext (MvcRolesAndMaps.Models)
Use async controller actions
Views:
Generate views
Reference script libraries
Use a layout page:
~/Views/Shared/_Layout.cshtml
(Leave empty if it is set in a Razor _viewstart file)
Controller name:
CategoriesController
Add
Cancel

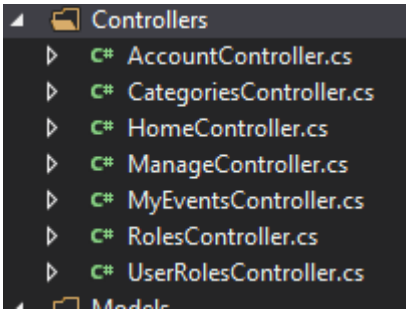

Microsoft Visual Studio
Scaffolding...


Controllers
AccountController.cs
CategoriesController.cs
HomeController.cs
ManageController.cs
RolesController.cs
UserRolesController.cs
Models


Add New Scaffolded Item
Installed
Common
Controller
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 views, using Entity Framework
by Microsoft
v5.0.0.0
An MVC controller with actions and Razor views to create, read, update, delete, and list entities from an Entity Framework data context.
Id: MvcControllerWithContextScaffolder
Click here to go online and find more scaffolding extensions.
Add
Cancel

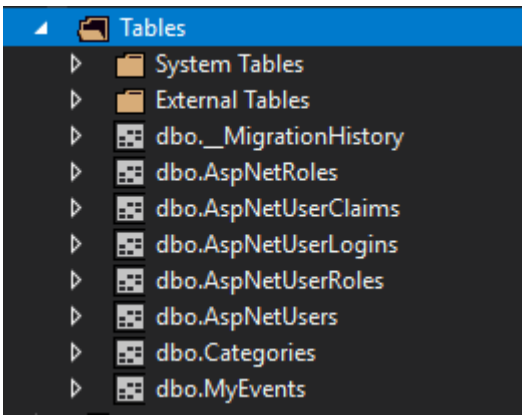


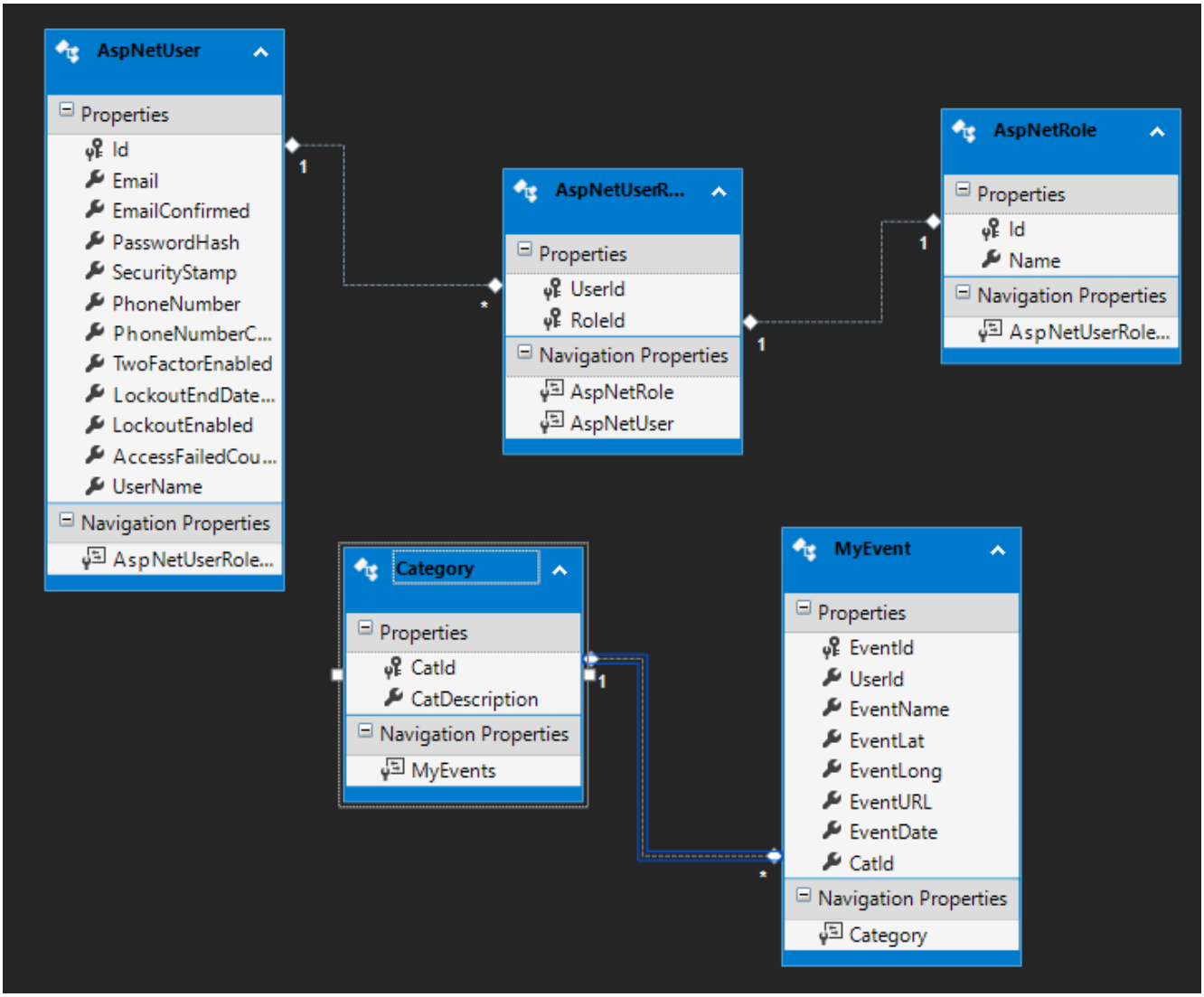





```
    11 references
    public System.Data.Entity.DbSet<MvcRolesAndMaps.Models.Category> Categories { get; set; }

    7 references
    public System.Data.Entity.DbSet<MvcRolesAndMaps.Models.MyEvent> MyEvents { get; set; }
}
```

```
6 references
public ApplicationDbContext()
    : base("DefaultConnection", throwIfV1Schema: false)
{
}

1 reference
public static ApplicationDbContext Create()
{
    return new ApplicationDbContext();
}

11 references
public System.Data.Entity.DbSet<MvcRolesAndMaps.Models.Category> Categories { get; set; }

7 references
public System.Data.Entity.DbSet<MvcRolesAndMaps.Models.MyEvent> MyEvents { get; set; }
```





Step 13

```
// More details see https://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public ActionResult Edit([Bind(Include = "EventId,EventName,EventLat,EventLong,EventURL,EventDate,CatId")] MyEvent myEvent)
{
    if (ModelState.IsValid)
```

```
= "EventId,EventName,EventLat,EventLong,EventURL,EventDate,CatId")] MyE
```

```
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public ActionResult Create([Bind(Include
    = "EventId,EventName,EventLat,EventLong," +
    "EventURL,EventDate,CatId")] MyEvent myEvent)
{
    if (ModelState.IsValid)
    {
        myEvent.UserId = User.Identity.GetUserId();
        // Manually add current user's Id
        db.MyEvents.Add(myEvent);
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    ViewBag.CatId = new SelectList(db.Categories, "CatId"
        , "CatDescription", myEvent.CatId);
    return View(myEvent);
}
```

```
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public ActionResult Create([Bind(Include = "EventId,EventName,EventLat,EventLong,EventURL,EventDate,CatId")] MyEvent myEvent)
{
    if (ModelState.IsValid)
    {
        myEvent.UserId = User.Identity.GetUserId(); // Manually add current user's Id
        db.MyEvents.Add(myEvent);
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    ViewBag.CatId = new SelectList(db.Categories, "CatId", "CatDescription", myEvent.CatId);
    return View(myEvent);
}
```

```
public ActionResult Details(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    MyEvent myEvent = db.MyEvents.Find(id);
    if (myEvent == null)
    {
        return HttpNotFound();
    }
    if (myEvent.UserId != User.Identity.GetUserId())
    {
        return HttpNotFound();
    }
    return View(myEvent);
}
```

```
0 references
public ActionResult Edit(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    MyEvent myEvent = db.MyEvents.Find(id);
    if (myEvent == null)
    {
        return HttpNotFound();
    }
    if (myEvent.UserId != User.Identity.GetUserId())
    {
        return HttpNotFound();
    }
    ViewBag.CatId = new SelectList(db.Categories, "CatId", "CatDescription", myEvent.CatId);
    return View(myEvent);
}
```

```
0 references
public ActionResult Edit(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult
            (HttpStatusCode.BadRequest);
    }
    MyEvent myEvent =
        db.MyEvents.Find(id);
    if (myEvent == null)
    {
        return HttpNotFound();
    }
    if (myEvent.UserId !=
        User.Identity.GetUserId())
    {
        return HttpNotFound();
    }
    ViewBag.CatId = new SelectList(db.Categories,
        "CatId",
        "CatDescription",
        myEvent.CatId);
    return View(myEvent);
}
```

```
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public ActionResult Edit([Bind(Include =
    "EventId,EventName,EventLat," +
    "EventLong,EventURL,EventDate,CatId")]
MyEvent myEvent)

    if (ModelState.IsValid)
    {
        myEvent.UserId
            = User.Identity.GetUserId();
        // Manually add current user's Id
        db.Entry(myEvent).State
            = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    ViewBag.CatId = new SelectList(db.Categories, "CatId"
        , "CatDescription", myEvent.CatId);
    return View(myEvent);
}
```

```
// more details see https://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public ActionResult Edit([Bind(Include = "EventId,EventName,EventLat,EventLong,EventURL,EventDate,CatId")] MyEvent myEvent)

    if (ModelState.IsValid)
    {
        myEvent.UserId = User.Identity.GetUserId(); // Manually add current user's Id
        db.Entry(myEvent).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    ViewBag.CatId = new SelectList(db.Categories, "CatId", "CatDescription", myEvent.CatId);
    return View(myEvent);
}
```

```
// GET: MyEvents/Delete/5
0 references
public ActionResult Delete(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult
            (HttpStatusCode.BadRequest);
    }
    MyEvent myEvent
        = db.MyEvents.Find(id);
    if (myEvent == null)
    {
        return HttpNotFound();
    }
    if (myEvent.UserId
        != User.Identity.GetUserId())
    {
        return HttpNotFound();
    }
    return View(myEvent);
}
```

```
0 references
public ActionResult Delete(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    MyEvent myEvent = db.MyEvents.Find(id);
    if (myEvent == null)
    {
        return HttpNotFound();
    }
    if (myEvent.UserId != User.Identity.GetUserId())
    {
        return HttpNotFound();
    }
    return View(myEvent);
}
```

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

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.EventCategory.CatDescription)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.EventName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.EventLat)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.EventLong)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.EventURL)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.EventDate)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
```

```
<div class="form-horizontal">
    <h4>MyEvent1</h4>
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })
    @Html.HiddenFor(model => model.EventId)
    <div class="form-group">
        @Html.LabelFor(model => model.EventName, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.EventName, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.EventName, "", new { @class = "text-danger" })
        </div>
    </div>

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

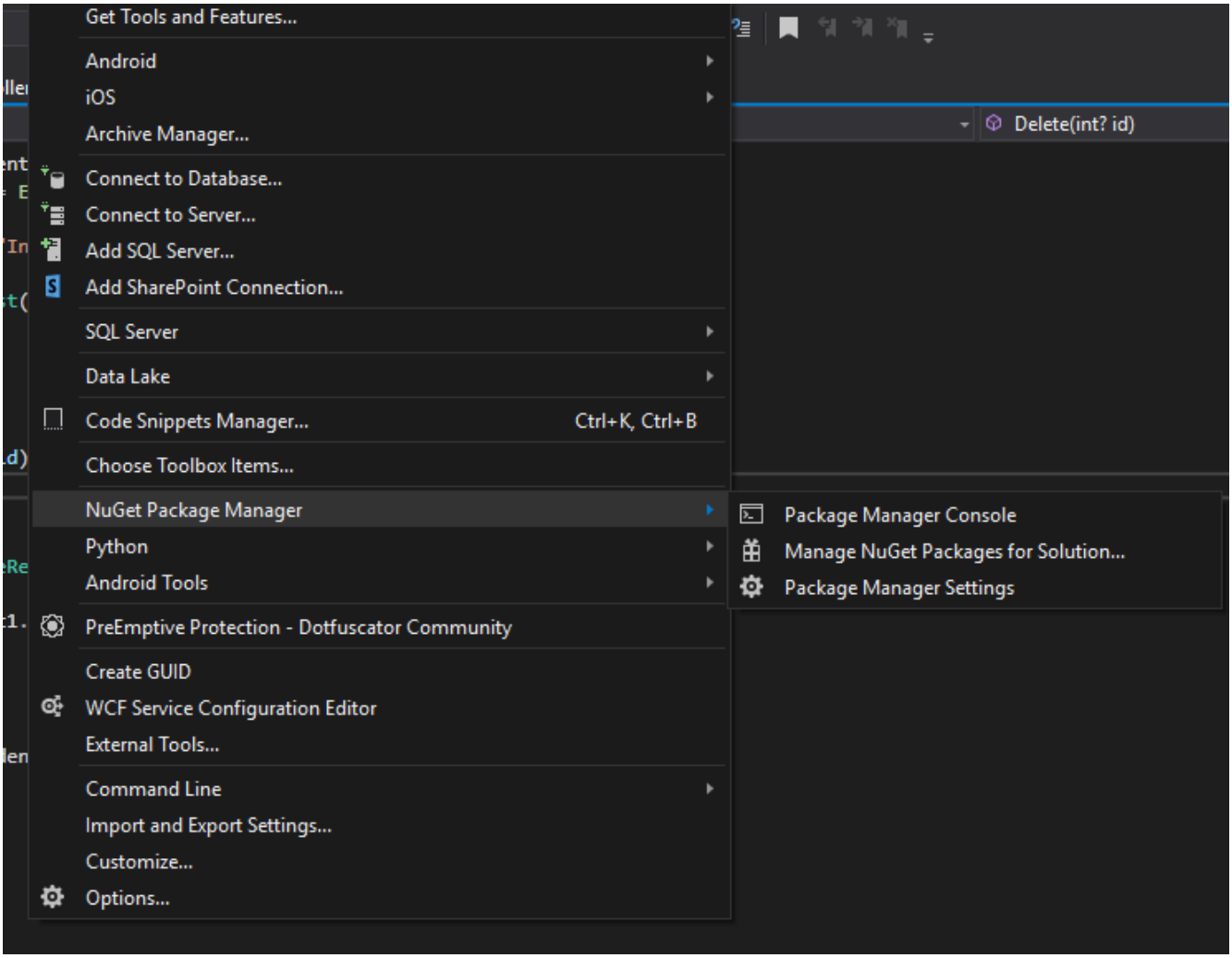
```
    @Html.ValidationSummary(true, "",
    new { @class = "text-danger" })
    @Html.HiddenFor(model => model.EventId)
    <div class="form-group">
        @Html.LabelFor(model => model.EventName,
       htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
  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

```
<dl class="dl-horizontal">
    <dt>
        @Html.DisplayNameFor(model => model.EventCategory.CatDescription)
    </dt>

    <dd>
        @Html.DisplayFor(model => model.EventCategory.CatDescription)
    </dd>
    <dt>
        @Html.DisplayNameFor(model => model.EventName)
    </dt>

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

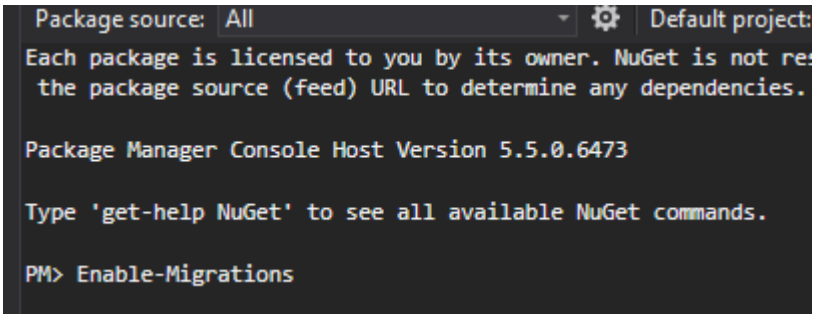
```
<dl class="dl-horizontal">
    <dt>
        @Html.DisplayNameFor(model => model.EventCategory.CatDescription)
    </dt>

    <dd>
        @Html.DisplayFor(model => model.EventCategory.CatDescription)
    </dd>
    <dt>
        @Html.DisplayNameFor(model => model.EventName)
    </dt>

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

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

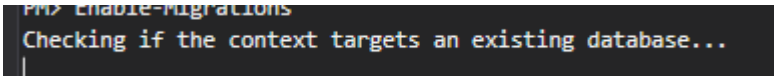
```
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })

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

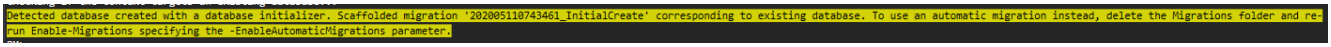
As you can see I deleted everything from the views that has to do with userid

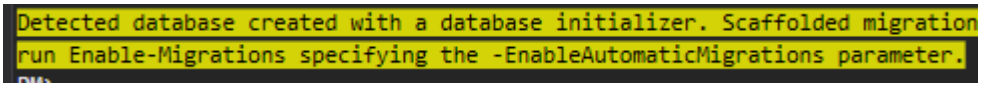


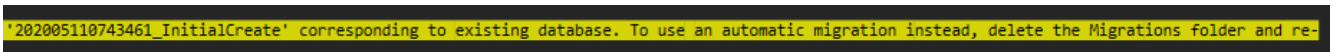
Step 14

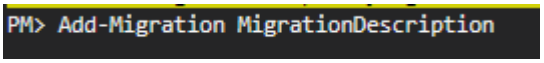

Package source: All
Default project:
Each package is licensed to you by its owner. NuGet is not re
the package source (feed) URL to determine any dependencies.
Package Manager Console Host Version 5.5.0.6473
Type 'get-help NuGet' to see all available NuGet commands.
PM> Enable-Migrations

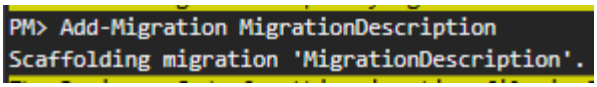

Checking if the context targets an existing database...

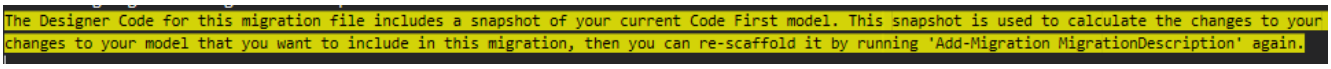

Detected database created with a database initializer. Scaffolded migration '202005110743461_InitialCreate' corresponding to existing database. To use an automatic migration instead, delete the Migrations folder and re-run Enable-Migrations specifying the -EnableAutomaticMigrations parameter.

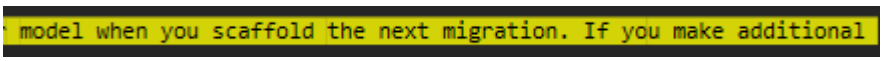

Detected database created with a database initializer. Scaffolded migration
run Enable-Migrations specifying the -EnableAutomaticMigrations parameter.

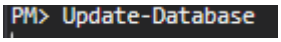

'202005110743461_InitialCreate' corresponding to existing database. To use an automatic migration instead, delete the Migrations folder and re-

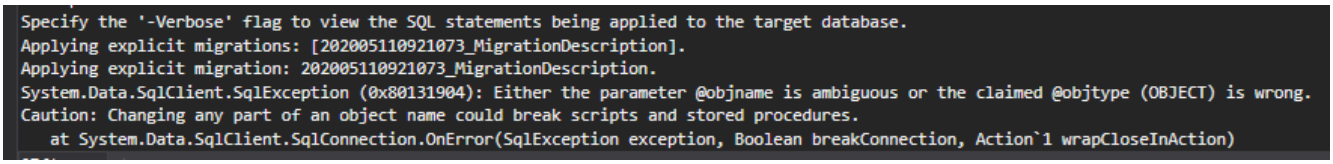

PM> Add-Migration MigrationDescription

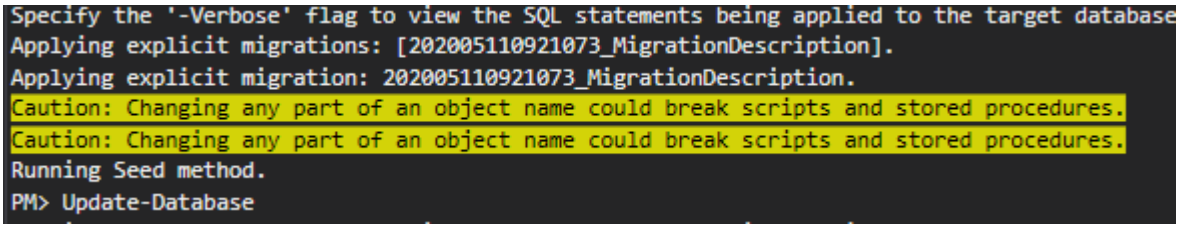

PM> Add-Migration MigrationDescription
Scaffolding migration 'MigrationDescription'.

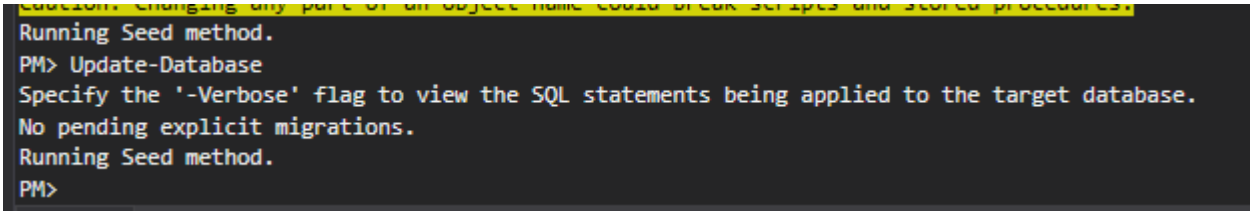

The Designer Code for this migration file includes a snapshot of your current Code First model. This snapshot is used to calculate the changes to your
changes to your model that you want to include in this migration, then you can re-scaffold it by running 'Add-Migration MigrationDescription' again.

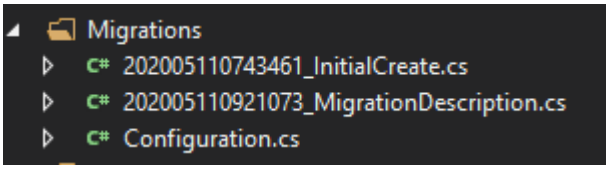

model when you scaffold the next migration. If you make additional


PM> Update-Database


Specify the '-Verbose' flag to view the SQL statements being applied to the target database.
Applying explicit migrations: [202005110921073_MigrationDescription].
Applying explicit migration: 202005110921073_MigrationDescription.
System.Data.SqlClient.SqlException (0x80131904): Either the parameter @objname is ambiguous or the claimed @objtype (OBJECT) is wrong.
Caution: Changing any part of an object name could break scripts and stored procedures.
at System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction)


Specify the '-Verbose' flag to view the SQL statements being applied to the target database
Applying explicit migrations: [202005110921073_MigrationDescription].
Applying explicit migration: 202005110921073_MigrationDescription.
Caution: Changing any part of an object name could break scripts and stored procedures.
Caution: Changing any part of an object name could break scripts and stored procedures.
Running Seed method.
PM> Update-Database


Running Seed method.
PM> Update-Database
Specify the '-Verbose' flag to view the SQL statements being applied to the target database.
No pending explicit migrations.
Running Seed method.
PM>

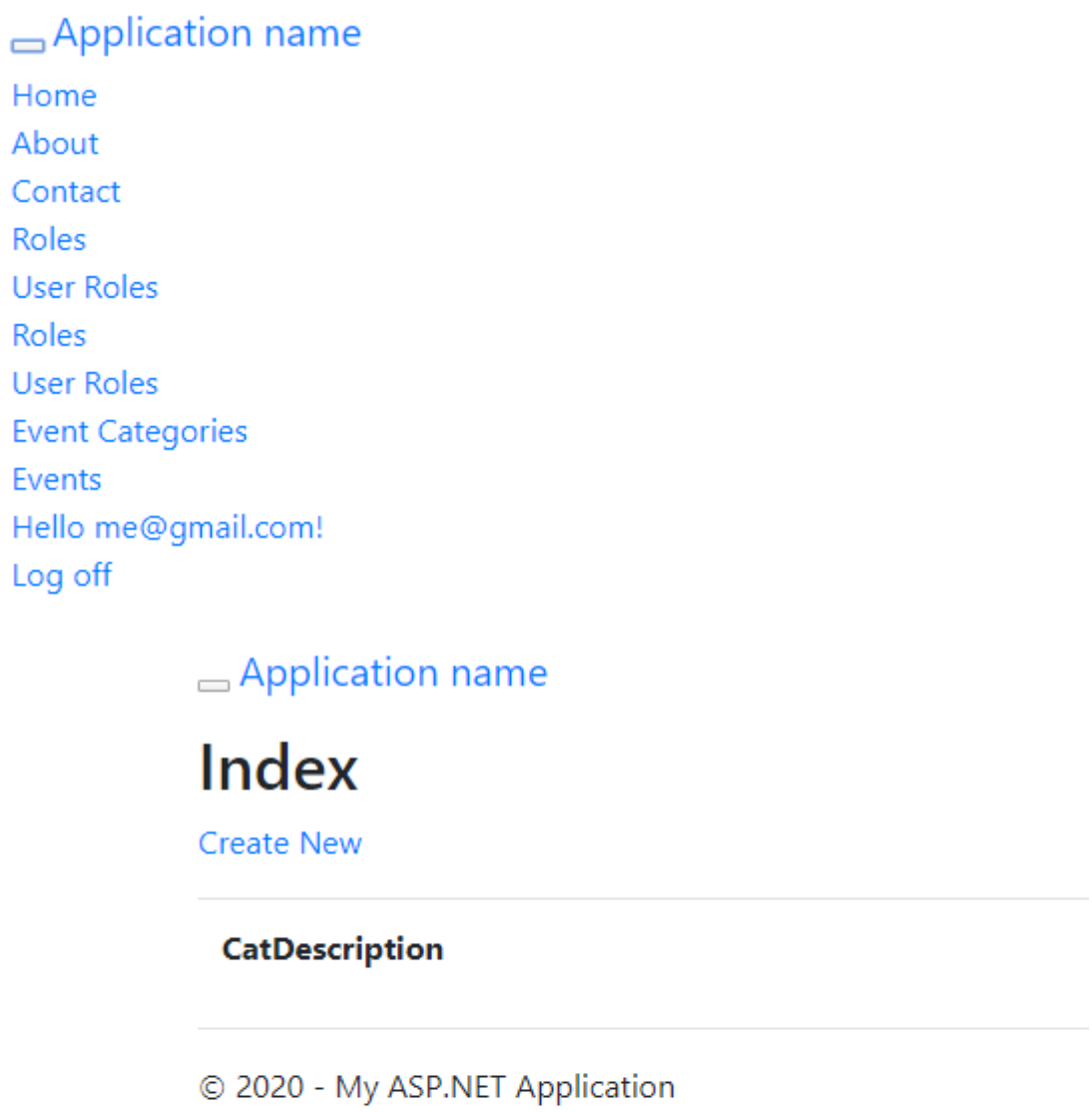

Migrations
202005110743461_InitialCreate.cs
202005110921073_MigrationDescription.cs
Configuration.cs

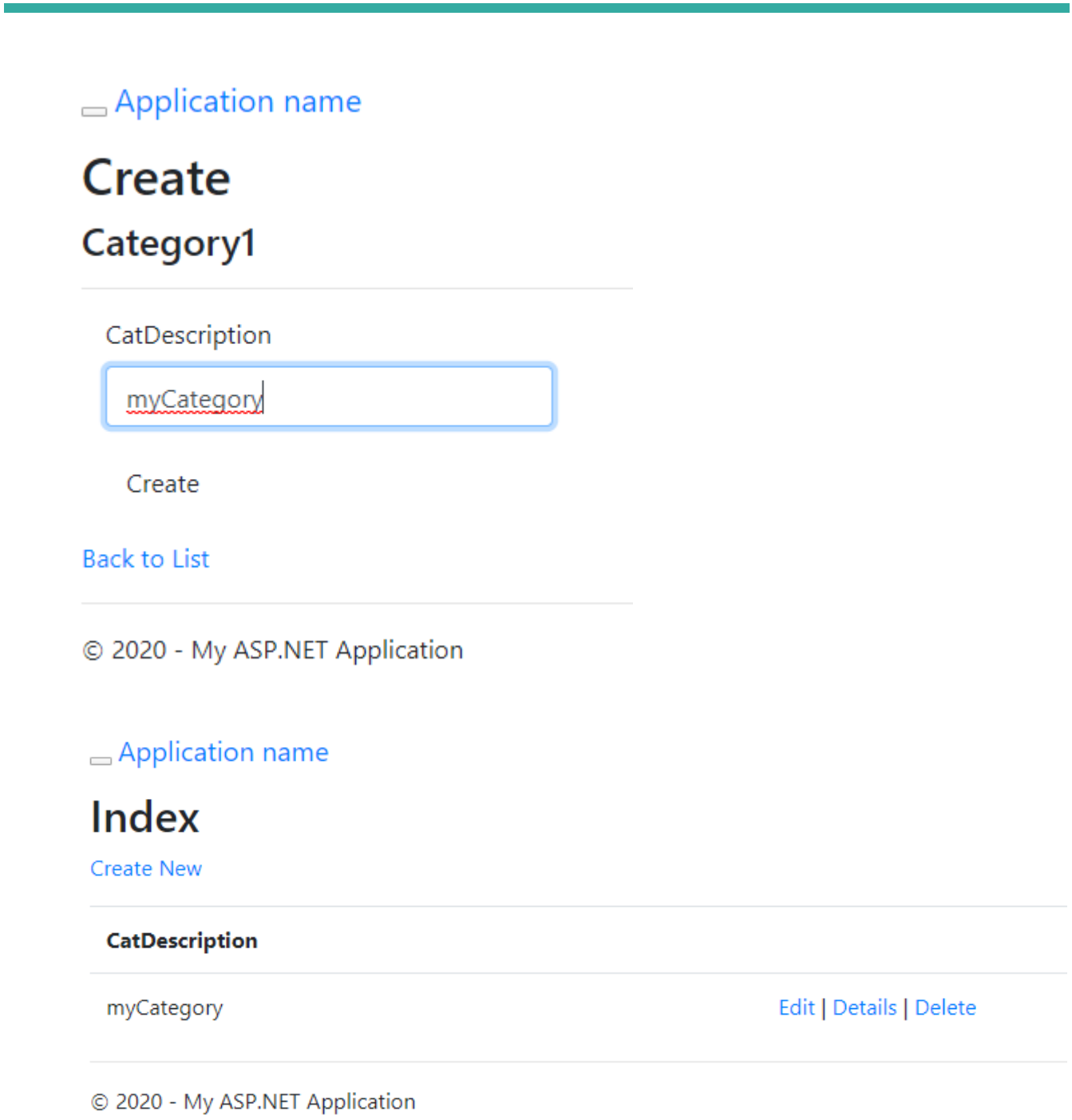
Step 15

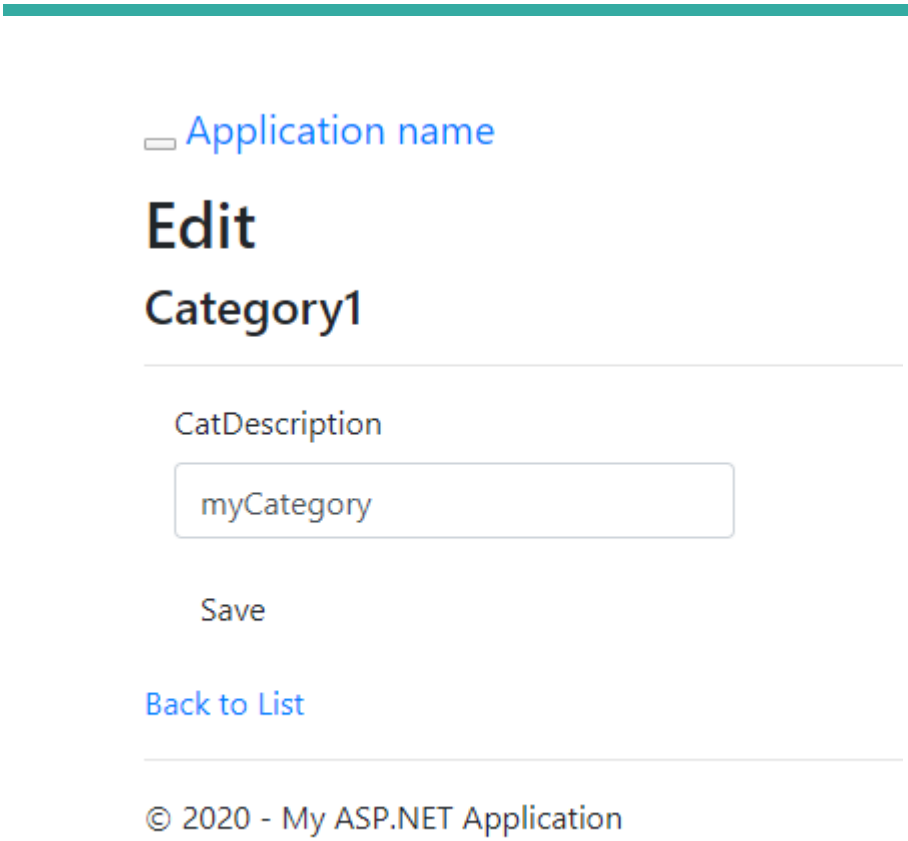
```
0 references
public class MyEvent1Controller : Controller
{
    private ApplicationDbContext db = new ApplicationDbContext();
```

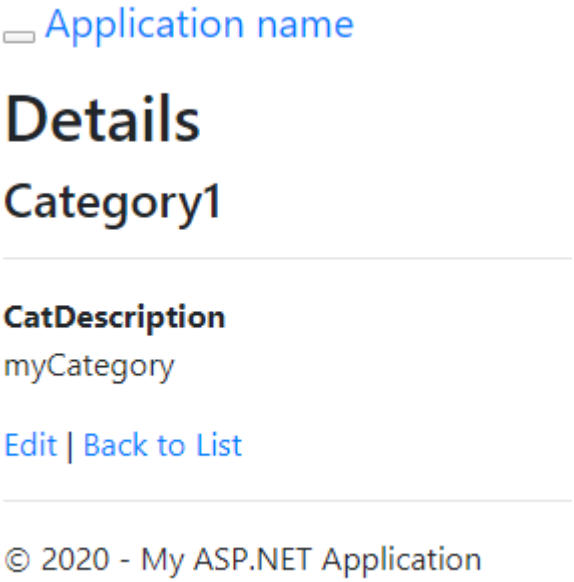
```
[Authorize(Roles = "Admin")]
0 references
public class Category1Controller : Controller
{
    private ApplicationDbContext db = new ApplicationDbContext();
```

Step 16


Application name
Home
About
Contact
Roles
User Roles
Roles
User Roles
Event Categories
Events
Hello me@gmail.com!
Log off
Application name
Index
Create New
CatDescription
© 2020 - My ASP.NET Application


Application name
Create
Category1
CatDescription
myCategory
Create
Back to List
© 2020 - My ASP.NET Application
Application name
Index
Create New
CatDescription
myCategory
Edit | Details | Delete
© 2020 - My ASP.NET Application


Application name
Edit
Category1
CatDescription
myCategory
Save
Back to List
© 2020 - My ASP.NET Application


Application name
Details
Category1
CatDescription
myCategory
Edit | Back to List
© 2020 - My ASP.NET Application

Application name

Delete

Are you sure you want to delete this?

Category1

CatDescription
myCategory

Delete | Back to List

© 2020 - My ASP.NET Application

Application name

Index

Create New

CatDescription

© 2020 - My ASP.NET Application

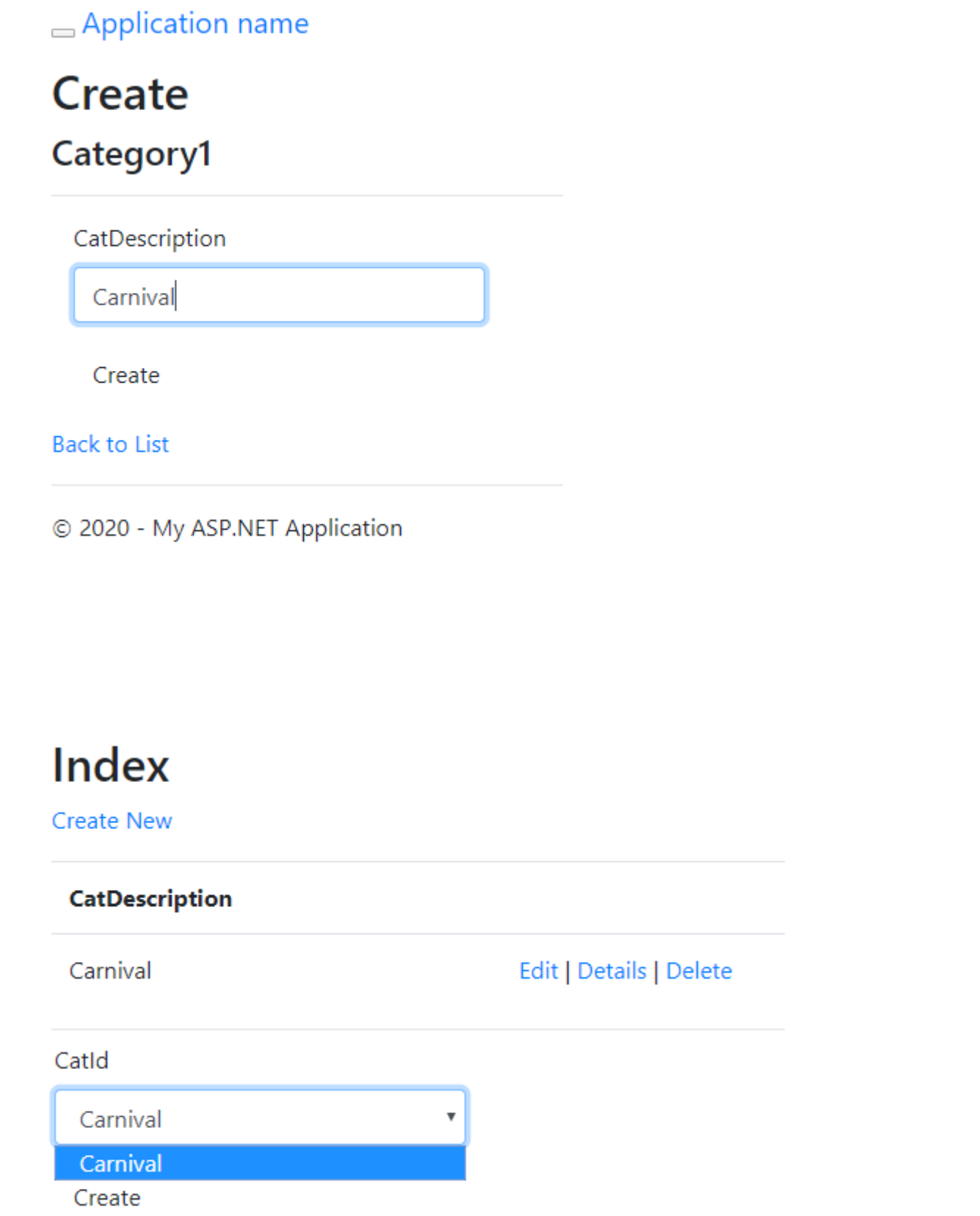
Application name

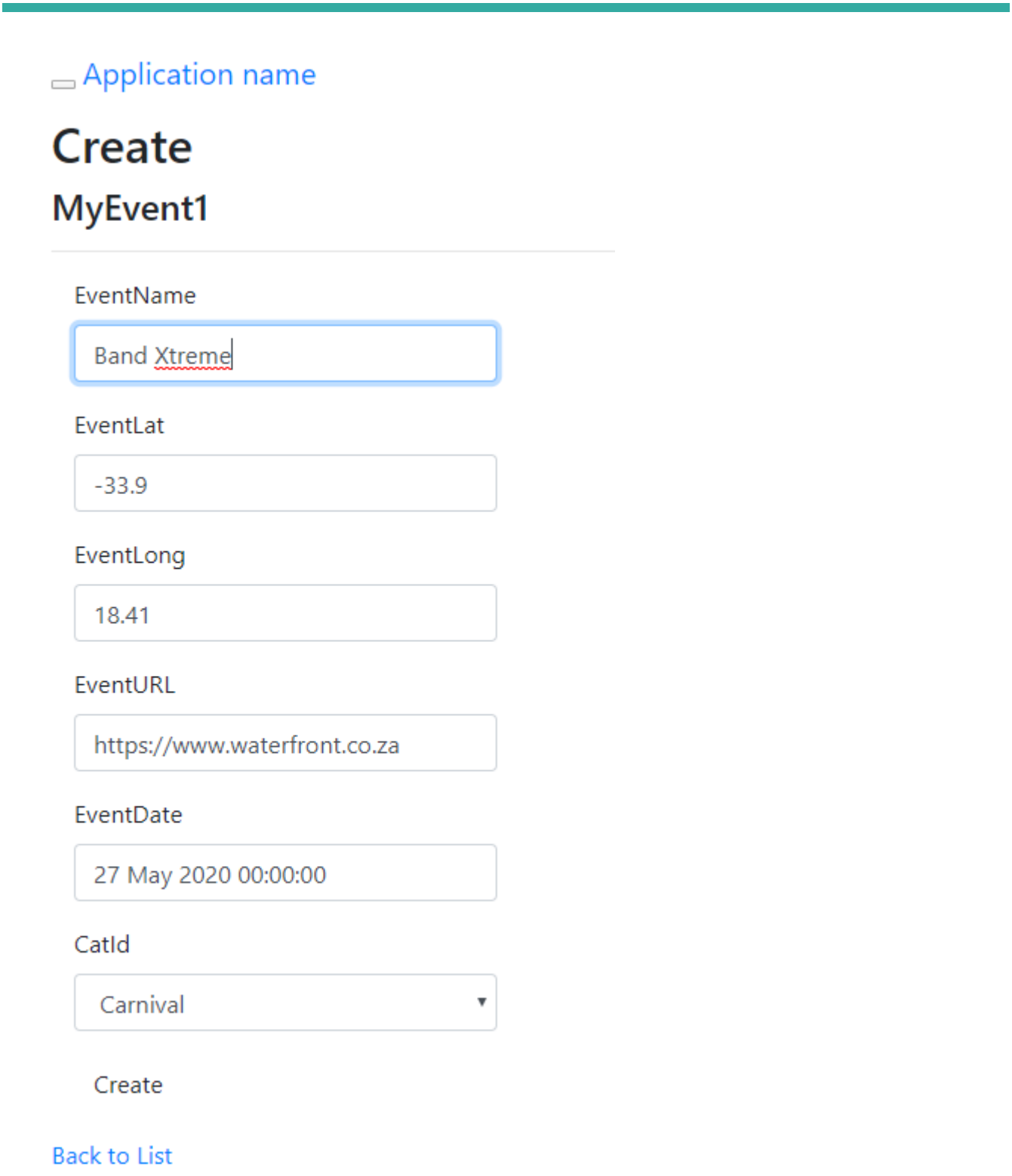
Index

Create New

CatDescription	EventName	EventLat	EventLong	EventURL	EventDate

© 2020 - My ASP.NET Application


Application name
Create
Category1
CatDescription
Carnival
Create
Back to List
© 2020 - My ASP.NET Application
Index
Create New
CatDescription
Carnival
Edit | Details | Delete
CatId
Carnival
Carnival
Create


Application name
Create
MyEvent1
EventName
Band Xtreme
EventLat
-33.9
EventLong
18.41
EventURL
https://www.waterfront.co.za
EventDate
27 May 2020 00:00:00
CatId
Carnival
Create
Back to List

Application name

Index

Create New

CatDescription	EventName	EventLat	EventLong	EventURL	EventDate	
Carnival	Band Xtreme	-33.9	18.41	https://www.waterfront.co.za	27/05/2020 00:00:00	Edit | Details | Delete

CatDescription	EventName	EventLat
Carnival	Band Xtreme	-33.9

EventLat	EventLong	EventURL
-33.9	18.41	https://www.waterfront.co.za

EventDate	
27/05/2020 00:00:00	Edit | Details | Delete

Application name

Details

MyEvent1

CatDescription
Carnival

EventName
Band Xtreme

EventLat
-33.9

EventLong
18.41

EventURL
https://www.waterfront.co.za

EventDate
27/05/2020 00:00:00

Edit | Back to List

Edit

MyEvent1

EventName

Band Xtreme

EventLat

-33.9

EventLong

18.41

EventURL

https://www.waterfront.co.za

EventDate

27/05/2020 00:00:00

CatId

Carnival

Save

Back to List

© 2020 - My ASP.NET Application

Application name

Delete

Are you sure you want to delete this?

MyEvent1

CatDescription
Carnival

EventName
Band Xtreme

EventLat
-33.9

EventLong
18.41

EventURL
https://www.waterfront.co.za

EventDate
27/05/2020 00:00:00

Delete | Back to List

Application name

Home
About
Contact
Events
Hello you@gmail.com!
Log off

Application name

Home
About
Contact
Events
Hello you@gmail.com!
Log off

Index

Create New

CatDescription	EventName	EventLat	EventLong	EventURL	EventDate

© 2020 - My ASP.NET Application

Application name

Home
About
Contact
Register
Log in

Step 17

```
using System.Linq;
using System.Web.Mvc;
namespace MvcRolesAndMaps.Controllers
{
    0 references
    public class MapController : Controller
    {
        private Models.ApplicationDbContext db = new Models.ApplicationDbContext();
```

```
private Models.ApplicationDbContext db = new Models.ApplicationDbContext();
// GET: Map
0 references
public ActionResult Index()
{
    return View();
}
// Get data for the map pins
0 references
public ActionResult GetJsonEvents()
```

```
0 references
public ActionResult Map()
{
    return View();
}
// Get data for the map pins
0 references
public ActionResult GetJsonEvents()
{
    // You need to specify which fields to include
    // otherwise EF includes everything - including
    // the category (which reads MyEvents, creating
    // a circular reference error).
```

```
// You need to specify which fields to include
// otherwise EF includes everything - including
// the category (which reads MyEvents, creating
// a circular reference error).
var events = db.MyEvent1.
Select(i => new {
    i.EventId,
    i.EventName,
    i.EventLat,
    i.EventLong
}).ToList();
return Json(events, JsonRequestBehavior.AllowGet);
```

```
}
// Get the clicked event's URL
0 references
public ActionResult GetEventUrl(int eventId)
{
    Models.MyEvent1 myEvent = db.MyEvent1.Find(eventId);
    if (myEvent == null)
    {
        return HttpNotFound();
    }
    var url = myEvent.EventURL;
    return Json(url, JsonRequestBehavior.AllowGet);
}
```

```
/* This code is based on this page:
* https://blogs.bing.com/maps/2015/02/05/visualizing-point-based-business-intelligence-data-on-bing-maps
*
* Next: Make the pins clickable.
*/
var map;
```

```
var map;
window.onload = function () {
    // Get the map from Microsoft
    var BingKey = "Asm_m-kvIRfg9W0zVBv09iv9L60HdgqQQSweVPdLT2qSocPSJHQ6ypcujtnsHJAE";
    map = new Microsoft.Maps.Map(document.getElementById("myMap"), {
        credentials: BingKey,
        mapTypeId: Microsoft.Maps.MapTypeId.road
    });
    // Add the events to the map. (The "GetJsonEvents" argument, informs Ajax what URL
    // the URL for your MapController/GetJsonEvents action).
    GetEventLocations("GetJsonEvents");
};
```

```
function GetEventLocations(MyUrl) {
    map.entities.clear();
    // Use Ajax to get the events from the server
    $.ajax({
        url: MyUrl, // action in MapController
        error: searchFailed, // function to display error message
        success: function (data) {
            // Loop through the received data
            $.each(data, function (objectNumber, eventObject) {
                // Extract data from the object and add it to a new pin
                var location = new Microsoft.Maps.Location(eventObject.EventLat, eventObject.EventLong);
                var pin = new Microsoft.Maps.Pushpin(location, {
                    text: eventObject.EventId + " " + eventObject.EventName
                });
                // Add a click event handler to the pushpin.
                // If the pin is clicked, pushpinClicked() will be called.
                Microsoft.Maps.Events.addHandler(pin, 'click', pushpinClicked);
                // Stick the pin to the map
                map.entities.push(pin);
            });
```

```
function searchFailed() {
    alert("Something went wrong with the AJAX");
}
function pushpinClicked(e) {
    var pinName = e.primitive.entity.iconText;
    // We need to extract MyEvent's Id so that we can pass it
    // to the server.
    // When we created the pin in GetEventLocations(), we constructed
    // the pin's text as: eventObject.EventId + " " + eventObject.EventName
    // We could thus use the first section of the text (before the space)
    // to extract the EventId.
    // Find the first space in the name (after the eventId)
    var space = pinName.indexOf(" ");
    // Extract the eventId - from start to first space
    var eventId = pinName.substring(0, space);
    // Use Ajax to get the event's url from the server
    $.ajax({
        url: "GetEventUrl", // action in MapController
        error: searchFailed,
        data: { eventId: eventId },
        success: function (data) {
            // redirect the browser to the event's page - similar
            // behavior as clicking on a link.
            window.open(
                data,
                '_blank' // <- This is what makes it open in a new window/tab.
            );
```

Step 18

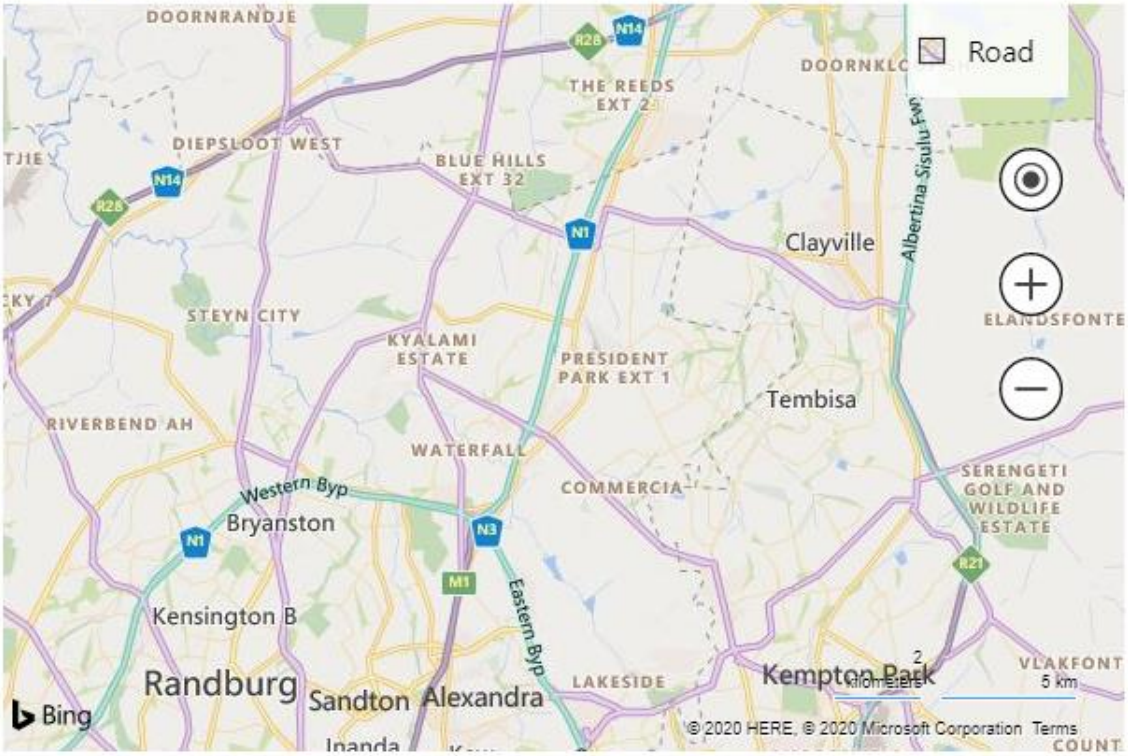
```
    }
    <li>@Html.ActionLink("Map", "Map", "Map")</li>
</ul>
@Html.Partial("_LoginPartial")
```

```
<div class="navbar-collapse collapse">
    <ul class="nav navbar-nav">
        <li>@Html.ActionLink("Home", "Index", "Home")</li>
        <li>@Html.ActionLink("About", "About", "Home")</li>
        <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
        @if (User.IsInRole("Admin"))
        {
            <li>@Html.ActionLink("Roles", "Index", "Roles")</li>
            <li>@Html.ActionLink("User Roles", "Index", "UserRoles")</li>
        }
        <!-- Test for specific user Role -->
        @if (User.IsInRole("Admin"))
        {
            <li>@Html.ActionLink("Roles", "Index", "Roles")</li>
            <li>@Html.ActionLink("User Roles", "Index", "UserRoles")</li>
            <li>@Html.ActionLink("Event Categories", "Index", "Category1")</li>
        }
        <!-- Test if user logged in - regardless of his/hr Roles -->
        @if ((System.Web.HttpContext.Current.User != null) &&
        (System.Web.HttpContext.Current.User.Identity.IsAuthenticated))
        {
            <li>@Html.ActionLink("Events", "Index", "MyEvent1")</li>
        }
        <li>@Html.ActionLink("Map", "Map", "Map")</li>
    </ul>
    @Html.Partial("_LoginPartial")
```

Step 19

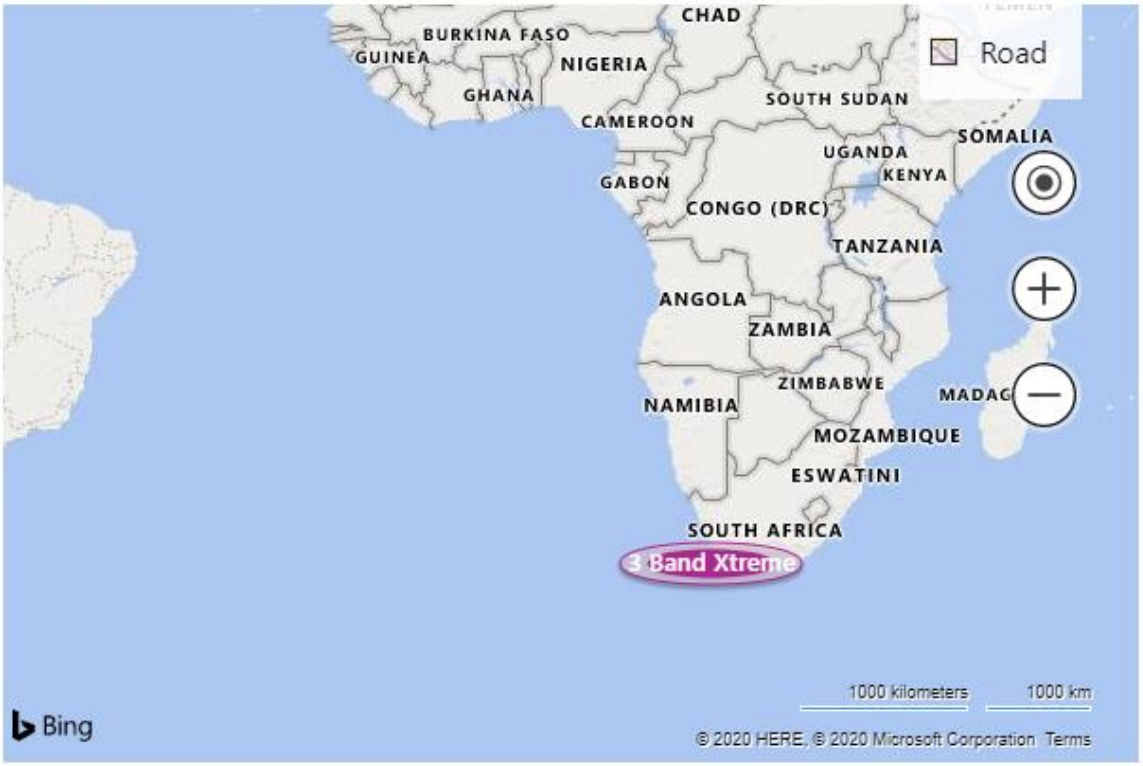
Application name

Map

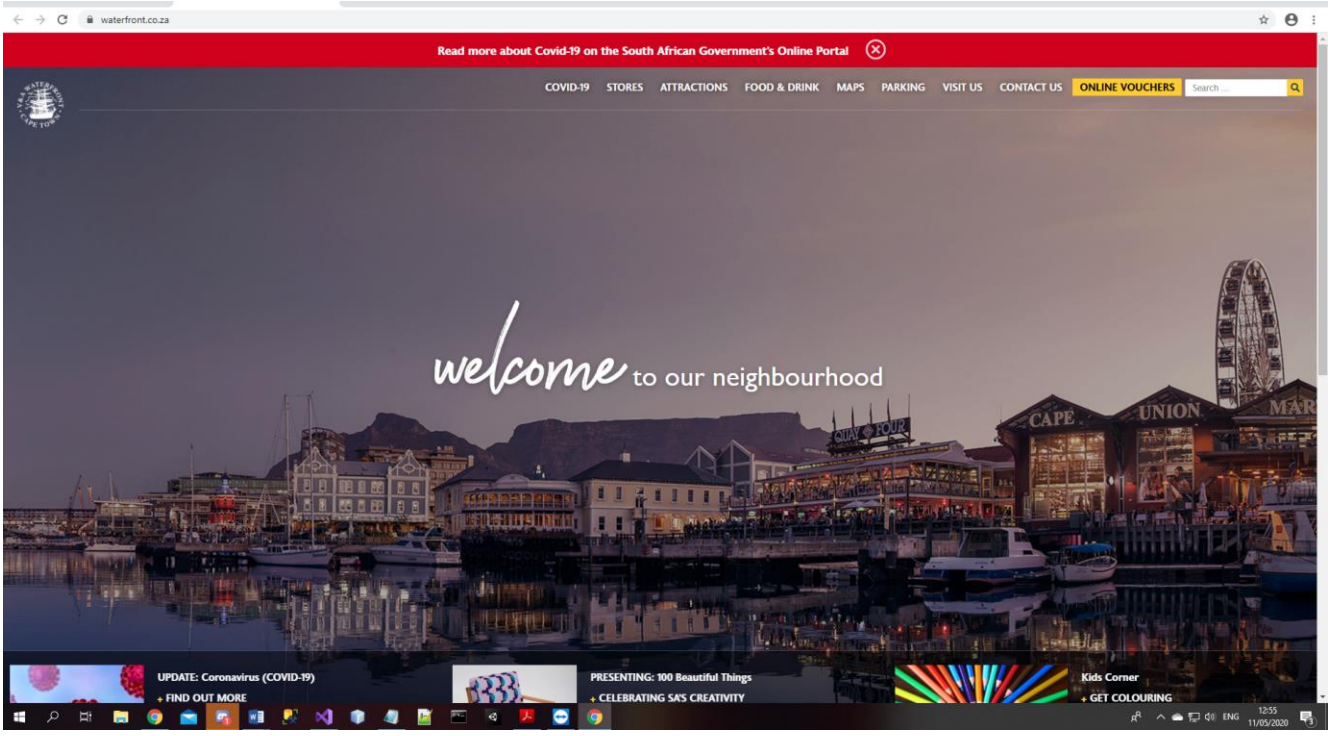


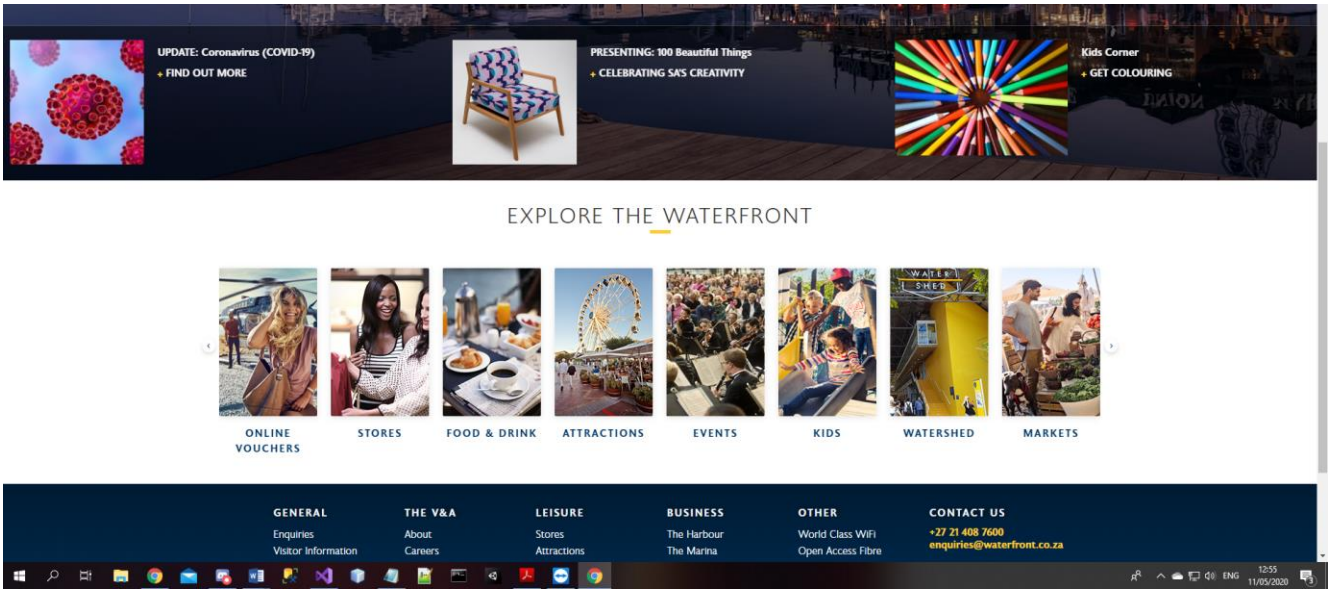
Application name

Map



When you clink on the pin





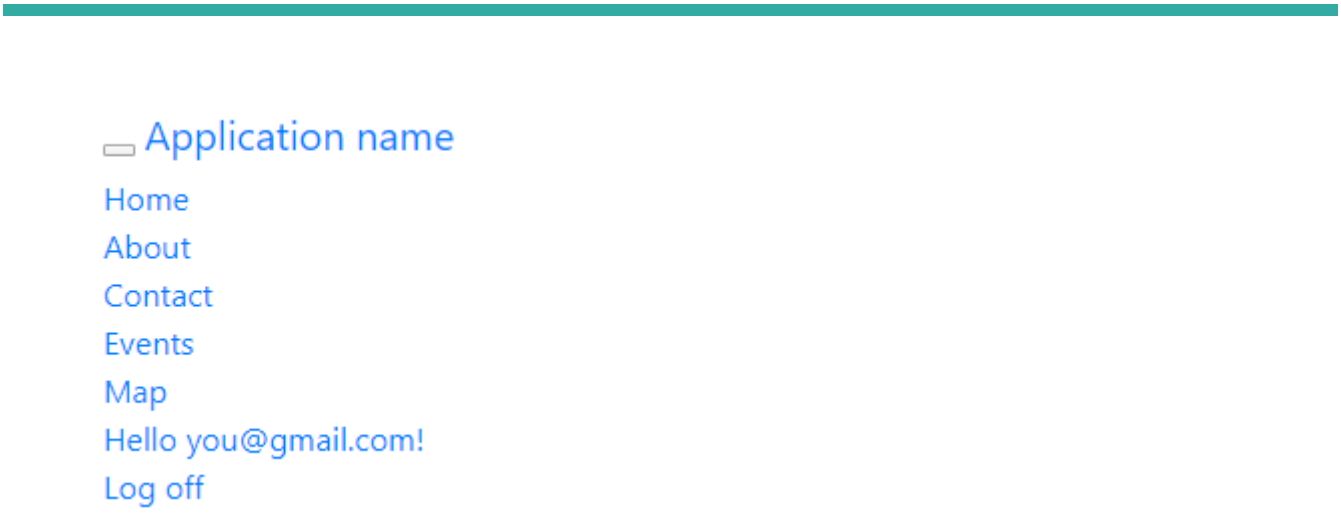
This is where the band xtreme pin took me

Application name

Home
About
Contact
Map
Register
Log in

Application name

Home
About
Contact
Roles
User Roles
Roles
User Roles
Event Categories
Events
Map
Hello me@gmail.com!
Log off


Application name
Home
About
Contact
Events
Map
Hello you@gmail.com!
Log off

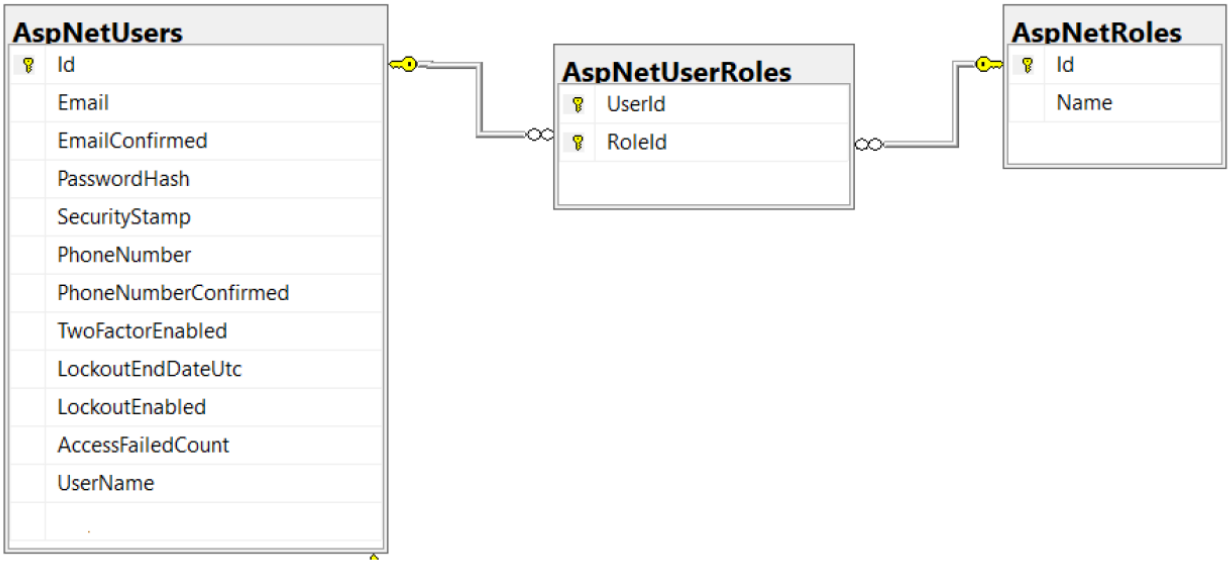
Conclusions

I completed all requirements in the documents

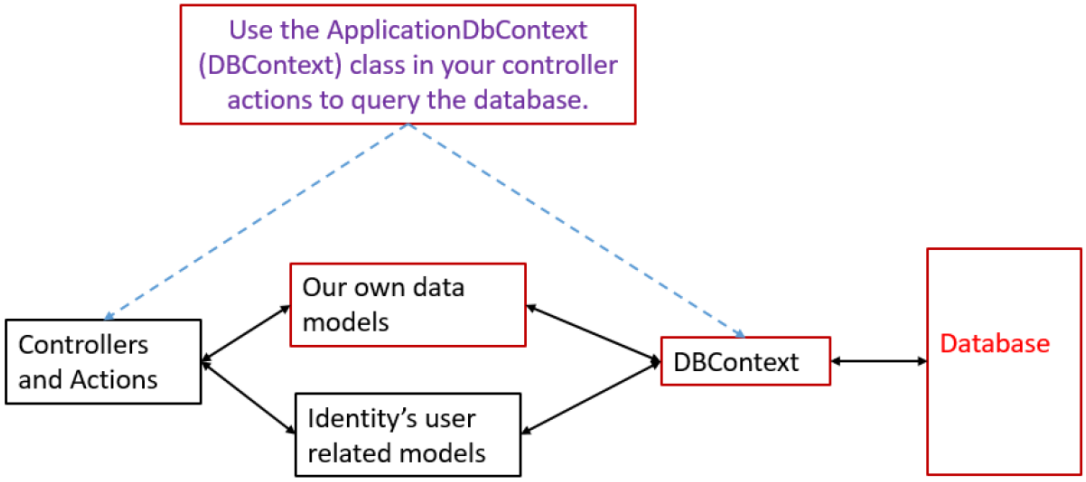
And everything worked and is still working and with mvc made my life as

A developer easier developing this site because of how the design pattern works

Appendix




```
<connectionStrings>
  <add name="DefaultConnection"
     connectionString="
           Data Source=(LocalDb)\MSSQLLocalDB;
           AttachDbFilename=|DataDirectory|\aspnet-MvcRolesAndMaps-20200501125720.mdf;
           Initial Catalog=aspnet-MvcRolesAndMaps-20200501125720;
           Integrated Security=True"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```



```
using MvcRolesAndMaps.Models;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;
using System.Threading.Tasks;
```

```
        public void Configuration(IAppBuilder app)
        {
            ConfigureAuth(app);
            CreateDefaultRoleAndUser();
        }

        // In this method we will create the default Admin roles and user
        private async Task CreateDefaultRoleAndUser()
        {
            ApplicationDbContext context = new ApplicationDbContext();

            var roleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(context));
            var UserManager = new UserManager<ApplicationUser>(new
UserStore<ApplicationUser>(context));

            if (!roleManager.RoleExists("Admin"))
            {
                // first we create the Admin role
                var role = new Microsoft.AspNet.Identity.EntityFramework.IdentityRole();
```

```
            if (!roleManager.RoleExists("Admin"))
            {
                // first we create the Admin role
                var role = new Microsoft.AspNet.Identity.EntityFramework.IdentityRole();
                role.Name = "Admin";
                roleManager.Create(role);

                // Here we create a user and grant him/her the Admin role.
                // The default behavior when creating users is a bit weird.
                // When users register, only their emails and passwords are required,
                // but these emails are stored in the database as both emails and usernames.
                // During login, the emails are again requested, but then compared to the
                // username in the DB. This behavior can be changed, but for now we will
                // just go with the flow.
                string myEmail = "me@gmail.com";
                string myPassword = "Ctu@2020"; // password must have a capital letter
                var user = new ApplicationUser { UserName = myEmail, Email = myEmail };
                var result = await UserManager.CreateAsync(user, myPassword);
                if (result.Succeeded)
                {
                    var result1 = UserManager.AddToRole(user.Id, "Admin");
                }
            }
        }
```

```
[Authorize(Roles = "Admin")]
public class RolesController : Controller
{
    ApplicationDbContext db = new ApplicationDbContext();

    // GET: Roles
    public ActionResult Index()
    {
        var Roles = db.Roles.ToList();
        return View(Roles);
    }

    // GET: Create
    public ActionResult Create()
    {
        var Role = new IdentityRole();
        return View(Role);
    }

    // POST: Create
    [HttpPost]
    public ActionResult Create(IdentityRole Role)
    {
        db.Roles.Add(Role);
        db.SaveChanges();
        return RedirectToAction("Index");
```

```
@model IEnumerable<Microsoft.AspNet.Identity.EntityFramework.IdentityRole>
@{
    ViewBag.Title = "Roles";
}

<h2>Roles</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>

<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Name)
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Name)
            </td>
        </tr>
    }
</table>
```

```
@model Microsoft.AspNet.Identity.EntityFramework.IdentityRole
@{
    ViewBag.Title = "Add New Role";
}
<h2>Add New User Role:</h2>

@using (Html.BeginForm())
{
    <p> Enter User Role Name:</p>

    @Html.EditorFor(m => m.Name)
    <input type="submit" value="Create Role" />
}
```

[Authorize(Roles = "Admin")] attribute above the controller.

```
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;
using System.Threading.Tasks;
using System.Net;

…
```

```
[Authorize(Roles = "Admin")]
public class UserRolesController : Controller
{
    ApplicationDbContext db = new ApplicationDbContext();

    public ActionResult Index()
    {
        // Get the users from the identity table
        var Users = db.Users.ToList();
        return View(Users);
    }

    // GET:
    public ActionResult EditUserRoles(string userId)
    {
        if (userId == null)
        {
```

```
        // GET:
        public ActionResult EditUserRoles(string userId)
        {
            if (userId == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }

            // Get the user
            Microsoft.AspNet.Identity.EntityFramework.IdentityUser user = db.Users.Find(userId);
            if (user == null)
            {
                return HttpNotFound();
            }

            // Get a list of all the available roles - to display in the dropdown
            var roles = db.Roles.ToList();

            // We want to highlight the user's existing roles (if any) in the dropdown
            // Get all the names of the user's roles using the received user's id
            var UserManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(db));
            List<string> userRolesNamesList = UserManager.GetRoles(userId).ToList();

            // Use the names to get the complete role objects (including their IDs) of the users roles
            var RoleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(db));
            List<IdentityRole> userRolesList = RoleManager.Roles.Where(r =>
serRolesNamesList.Contains(r.Name)).ToList();

            // Create an array containing the user's existing roles' IDs (if any)
            string[] userRolesIDsArray = new string[userRolesList.Count];

               ViewBag.Roles = new MultiSelectList(roles, "Id", "Name", userRolesIDsArray);

               return View(user);
           }

           // POST:
           [HttpPost]
           [ValidateAntiForgeryToken]
           public async Task<ActionResult> EditUserRoles(FormCollection collection)
           {
               var UserManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(db));
               var RoleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(db));

               // Because the form contains data from two different models (user
               // and roles), we use a FormCollection to extract the data manually.
               // Alternatively, we could have created a special viewmodel just for
               // this purpose - which would also enable easier validation.

               // Extract the user's id
               string userId = collection["id"];
               if (userId == null)
               {
                   return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
               }

               // Get the user
               Microsoft.AspNet.Identity.EntityFramework.IdentityUser user = db.Users.Find(userId);
               if (user == null)
```

```
// Get the user
Microsoft.AspNet.Identity.EntityFramework.IdentityUser user = db.Users.Find(userId);
if (user == null)
{
    return HttpNotFound();
}

// Extract the Role IDs from the collection (in a single
// text string - separated by commas)
string RoleIdString = collection["RoleId"];
if (RoleIdString != null)
{
    // Remove all of the user's existing Roles
    var userRoles = await UserManager.GetRolesAsync(userId);
    await UserManager.RemoveFromRolesAsync(userId, userRoles.ToArray());

    // Add the user's new Roles
    // Extract the roles from the text string into an array (by commas)
    string[] RoleIdArray = RoleIdString.Split(',');
    foreach (string roleId in RoleIdArray)
    {
        // Retrieve the role, because you need its Name below
        IdentityRole role = RoleManager.FindById(roleId);
        // Add the user to the role
        UserManager.AddToRole(userId, role.Name);
    }
}

/* Repeat the processes used in the "GET" EditUserRoles action
   to display the user's new Roles */
// Get a list of all the available roles - to display in the dropdown
```

```
            }

            /* Repeat the processes used in the "GET" EditUserRoles action
               to display the user's new Roles */
            // Get a list of all the available roles - to display in the dropdown
            var roles = db.Roles.ToList();

            // We want to highlight the user's existing roles (if any) in the dropdown
            // Get all the names of the user's roles using the received user's id
            List<string> userRolesNamesList = UserManager.GetRoles(userId).ToList();

            // Use the names to get the complete role objects (including their IDs) of the users roles
            List<IdentityRole> userRolesList = RoleManager.Roles.Where(r =>
userRolesNamesList.Contains(r.Name)).ToList();

            // Create an array containing the user's existing roles' IDs (if any)
            string[] userRolesIDsArray = new string[userRolesList.Count];
            int index = 0;
```

```
            foreach (IdentityRole role in userRolesList)
            {
                userRolesIDsArray[index] = role.Id; // This is the role id - not the user id we used earlier
                index++;
            }

            // Add the roles to the viewbag, since it is treated as "additional data"
            // (it is not the user's data normally associated with a controller action
            // like this)
            ViewBag.Roles = new MultiSelectList(roles, "Id", "Name", userRolesIDsArray);

            return View(user);
        }
    }
```

```
@model IEnumerable<Microsoft.AspNet.Identity.EntityFramework.IdentityUser>
@{
    ViewBag.Title = "Users";
}

<h2>Users</h2>

<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.UserName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Email)
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.UserName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Email)
            </td>
```

```
    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.UserName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Email)
            </td>
            <td>
                @Html.ActionLink("Edit Roles", "EditUserRoles", new { userId = item.Id })
            </td>
        </tr>
    }

</table>
```

```
@model Microsoft.AspNet.Identity.EntityFramework.IdentityUser
@{
    ViewBag.Title = "Users";
}

<h2>Edit User Roles</h2>

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

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

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

            <div class="col-md-10">
                @* Since we are not editing the username as well, we want to disable it.
            Replace the usual @Html.EditorFor() with @Html.TextBoxFor() and disable it *@
                @Html.TextBoxFor(model => model.UserName, new { disabled = "disabled" })
            </div>
        </div>
```

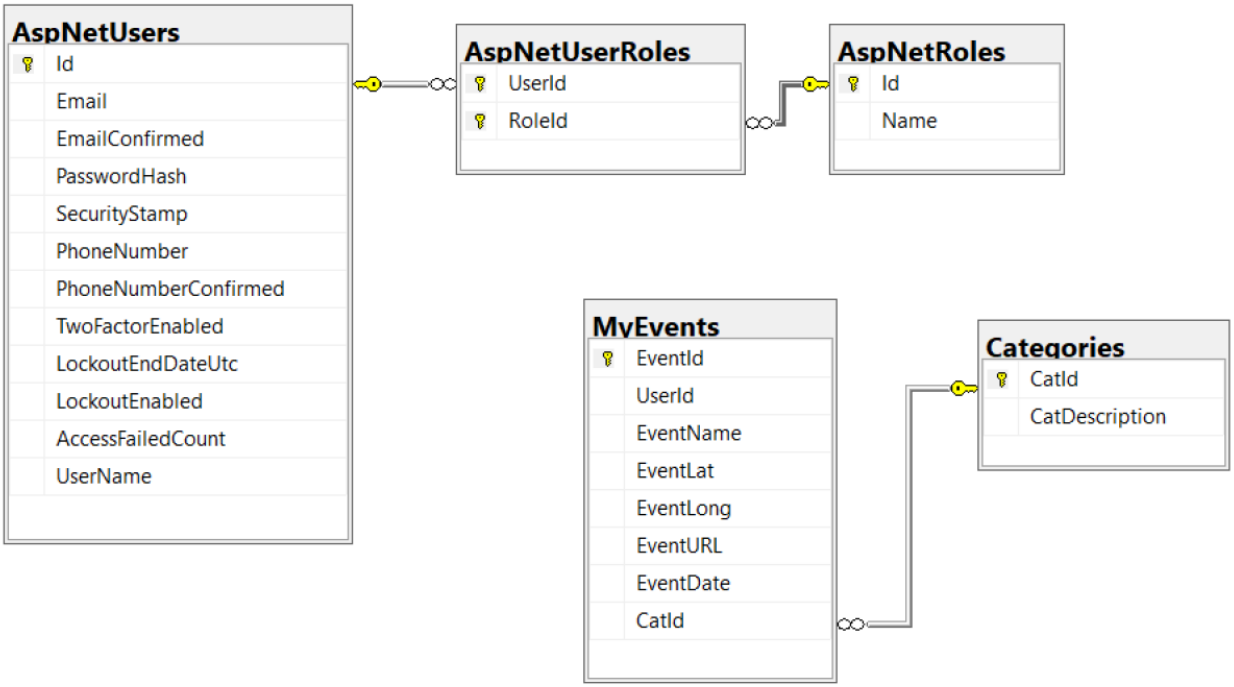
```
            Replace the usual @Html.EditorFor() with @Html.TextBoxFor() and disable it *@
                @Html.TextBoxFor(model => model.UserName, new { disabled = "disabled" })
            </div>
        </div>

        <br/>

        @*Display the Roles*@
        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                @Html.Label("Roles (Press Ctrl to select multiple rows)")
            </div>

            <div class="col-md-offset-2 col-md-10">
                @Html.DropDownList("RoleId", (MultiSelectList)ViewBag.Roles, new { multiple =
"multiple", size = 8})
            </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>
}
```

```
  <ul class="nav navbar-nav">
       <li>@Html.ActionLink("Home", "Index", "Home")</li>
       <li>@Html.ActionLink("About", "About", "Home")</li>
       <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
       @if (User.IsInRole("Admin"))
       {
           <li>@Html.ActionLink("Roles", "Index", "Roles")</li>
           <li>@Html.ActionLink("User Roles", "Index", "UserRoles")</li>
       }
   </ul>
```

\

```
  public class Category
  {
      [Key]
      public virtual int CatId { get; set; } // PK
      public virtual string CatDescription { get; set; }
      public virtual List<MyEvent> MyEvents { get; set; } // Child records
  }
```

```
public class MyEvent //Event is a reserved word - so call the class MyEvent
{
    [Key]
    public virtual int EventId { get; set; } // PK
    public virtual string UserId { get; set; } // User that created the record
    public virtual string EventName { get; set; }
    public virtual float EventLat { get; set; } // latitude
    public virtual float EventLong { get; set; } // longitude
    public virtual string EventURL { get; set; }
    public virtual DateTime EventDate { get; set; }
    public virtual int CatId { get; set; } // FK to parent (Category)
    public virtual Category EventCategory { get; set; } // Parent record
}
```



```
using Microsoft.AspNet.Identity;
```

Then manually add the UserId inside the Create action:

```
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = "EventId,EventName,EventLat,EventLong,
        EventURL,EventDate,CatId")] MyEvent myEvent)
        {
            if (ModelState.IsValid)
            {
                myEvent.UserId = User.Identity.GetUserId(); // Manually add current user's Id

                db.MyEvents.Add(myEvent);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.CatId = new SelectList(db.Categories, "CatId", "CatDescription", myEvent.CatId);
            return View(myEvent);
        }
```

```
public ActionResult Index()
 {
     string userId = User.Identity.GetUserId();
     var myEvents = db.MyEvents.Include(m => m.EventCategory)
                               .Where(m => m.UserId == userId);
     return View(myEvents.ToList());
 }
```

```
    public ActionResult Details(int? id)
     {
         if (id == null)
         {
             return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
         }
         MyEvent myEvent = db.MyEvents.Find(id);
         if (myEvent == null)
         {
             return HttpNotFound();
         }
         if(myEvent.UserId != User.Identity.GetUserId())
         {
             return HttpNotFound();
         }
         return View(myEvent);
     }


     // GET: MyEvents/Edit/5
     public ActionResult Edit(int? id)
     {
```

```
// GET: MyEvents/Edit/5
public ActionResult Edit(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    MyEvent myEvent = db.MyEvents.Find(id);
    if (myEvent == null)
    {
        return HttpNotFound();
    }
    if (myEvent.UserId != User.Identity.GetUserId())
    {
        return HttpNotFound();
    }
    ViewBag.CatId = new SelectList(db.Categories, "CatId", "CatDescription", myEvent.CatId);
    return View(myEvent);
}
```

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include = "EventId,UserId,EventName,EventLat,EventLong,
                    EventURL,EventDate,CatId")] MyEvent myEvent)
{
…
}
```

```
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Edit([Bind(Include = "EventId,EventName,EventLat,EventLong,
                      EventURL,EventDate,CatId")] MyEvent myEvent)
    {
        if (ModelState.IsValid)
        {
            myEvent.UserId = User.Identity.GetUserId(); // Manually add current user's Id

            db.Entry(myEvent).State = EntityState.Modified;
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        ViewBag.CatId = new SelectList(db.Categories, "CatId", "CatDescription", myEvent.CatId);
        return View(myEvent);
    }
```

```
public ActionResult Delete(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    MyEvent myEvent = db.MyEvents.Find(id);
    if (myEvent == null)
    {
        return HttpNotFound();
    }
    if (myEvent.UserId != User.Identity.GetUserId())
    {
        return HttpNotFound();
    }
    return View(myEvent);
}
```

Index.cshtml

```
        <th>
            @Html.DisplayNameFor(model => model.UserId)
        </th>
```

and

```
       <td>
            @Html.DisplayFor(modelItem => item.UserId)
        </td>
```

Create.cshtml

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

Edit.cshtml

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

Details.cshtml

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

Details.cshtml

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

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

Delete.cshtml

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

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

```
<ul class="nav navbar-nav">
    <li>@Html.ActionLink("Home", "Index", "Home")</li>
    <li>@Html.ActionLink("About", "About", "Home")</li>
    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>

    <!-- Test for specific user Role -->
    @if (User.IsInRole("Admin"))
    {
        <li>@Html.ActionLink("Roles", "Index", "Roles")</li>
        <li>@Html.ActionLink("User Roles", "Index", "UserRoles")</li>
        <li>@Html.ActionLink("Event Categories", "Index", "Categories")</li>
    }

    <!-- Test if user logged in - regardless of his/hr Roles -->
    @if ((System.Web.HttpContext.Current.User != null) &&
         (System.Web.HttpContext.Current.User.Identity.IsAuthenticated))
    {
        <li>@Html.ActionLink("Events", "Index", "MyEvents")</li>
    }

</ul>
```

```
public class MapController : Controller
    {
        private ApplicationDbContext db = new ApplicationDbContext();

        // Display the Map view
        public ActionResult Map()
        {
            return View();
        }

        // Get data for the map pins
        public ActionResult GetJsonEvents()
        {
            // You need to specify which fields to include
            // otherwise EF includes everything - including
            // the category (which reads MyEvents, creating
            // a circular reference error).
            var events = db.MyEvents.
                Select(i => new {
                    i.EventId,
                    i.EventName,
                    i.EventLat,
                    i.EventLong
                }).ToList();
            return Json(events, JsonRequestBehavior.AllowGet);
        }

        // Get the clicked event's URL
        public ActionResult GetEventUrl(int eventId)
        {
            MyEvent myEvent = db.MyEvents.Find(eventId);
            if (myEvent == null)
            {
                return HttpNotFound();
            }
            var url = myEvent.EventURL;
            return Json(url, JsonRequestBehavior.AllowGet);
        }
    }
```

- Map() simply returns the Map view we will create next.
- GetJsonEvents() retrieves all MyEvent records and return them in JSON format.
- GetEventUrl() retrieves a specified MyEvent and then returns it URL in JSON format.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Custom BI Pushpins</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

    <script type="text/javascript"
       src="https://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>
    <script type="text/javascript" src="~/Scripts/CustomMapJS.js"></script>
</head>
<body>
    <div id='myMap' style="position:relative;width:600px;height:400px;"></div><br />
</body>
</html>
```

```
var map;

window.onload = function () {
    // Get the map from Microsoft
    var BingKey = "Asm_m-kvIRfg9W0zVBv09iv9L60HdgqQQSweVPdLT2qSocPSJHQ6ypcujtnsHJAE";
    map = new Microsoft.Maps.Map(document.getElementById("myMap"), {
        credentials: BingKey,
        mapTypeId: Microsoft.Maps.MapTypeId.road
    });

    // Add the events to the map. (The "GetJsonEvents" argument, informs Ajax what URL to use (which is
    // the URL for your MapController/GetJsonEvents action).
    GetEventLocations("GetJsonEvents");
};

function GetEventLocations(MyUrl) {
    map.entities.clear();
    // Use Ajax to get the events from the server
    $.ajax({
        url: MyUrl, // action in MapController
        error: searchFailed, // function to display error message
        success: function (data) {
            // Loop through the received data
            $.each(data, function (objectNumber, eventObject) {
                // Extract data from the object and add it to a new pin
                var location = new Microsoft.Maps.Location(eventObject.EventLat, eventObject.EventLong);
```

```
            // Extract data from the object and add it to a new pin
            var location = new Microsoft.Maps.Location(eventObject.EventLat, eventObject.EventLong);
            var pin = new Microsoft.Maps.Pushpin(location, {
                text: eventObject.EventId + " " + eventObject.EventName
            });
            // Add a click event handler to the pushpin.
            // If the pin is clicked, pushpinClicked() will be called.
            Microsoft.Maps.Events.addHandler(pin, 'click', pushpinClicked);
            // Stick the pin to the map
```

```
            map.entities.push(pin);
        });
    }
);

    function searchFailed() {
        alert("Something went wrong with the AJAX");
    }

    function pushpinClicked(e) {
        var pinName = e.primitive.entity.iconText;

        // We need to extract MyEvent's Id so that we can pass it
        // to the server.
        // When we created the pin in GetEventLocations(), we constructed
        // the pin's text as: eventObject.EventId + " " + eventObject.EventName
        // We could thus use the first section of the text (before the space)
        // to extract the EventId.
        // Find the first space in the name (after the eventId)
        var space = pinName.indexOf(" ");

        // Extract the eventId - from start to first space
        var eventId = pinName.substring(0, space);

        // Use Ajax to get the event's url from the server
        $.ajax({
            url: "GetEventUrl", // action in MapController
            error: searchFailed,
            data: { eventId: eventId },
            success: function (data) {
                // redirect the browser to the event's page - similar
                // behavior as clicking on a link.
                window.open(
                    data,
```

```
<ul class="nav navbar-nav">
    <li>@Html.ActionLink("Home", "Index", "Home")</li>
    <li>@Html.ActionLink("About", "About", "Home")</li>
    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>

    <!-- Test for specific user Role -->
    @if (User.IsInRole("Admin"))
    {
        <li>@Html.ActionLink("Roles", "Index", "Roles")</li>
        <li>@Html.ActionLink("User Roles", "Index", "UserRoles")</li>
        <li>@Html.ActionLink("Event Categories", "Index", "Categories")</li>
    }

    <!-- Test if user logged in - regardless of his/hr Roles -->
    @if ((System.Web.HttpContext.Current.User != null) &&
         (System.Web.HttpContext.Current.User.Identity.IsAuthenticated))
    {
        <li>@Html.ActionLink("Events", "Index", "MyEvents")</li>
    }

    <li>@Html.ActionLink("Map", "Map", "Map")</li>
```

- Enable-Migrations
- Add-Migration *MigrationDescription*
- Update-Database

References:

PDf Document:

MVC User Roles and Map with Clickable Pins_FA1