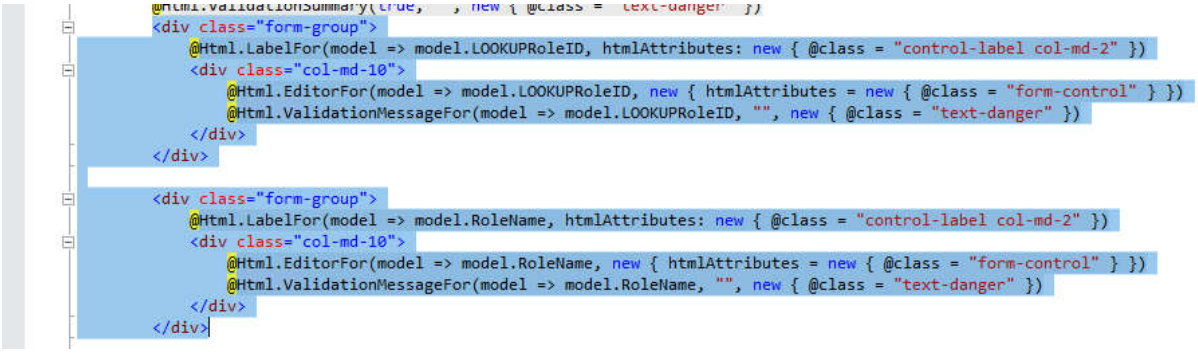
# *Övning MS SQL och MVC del 2*

*Denna övning, som är uppdelad i två delar, kommer att visa hur du kan arbeta med MS SQL och MVC. Applikationen är ett inloggningssystem som innehåller de flesta saker som vi har gått igenom under MVC-avsnittet.*

*Detta är den avslutande delen.*

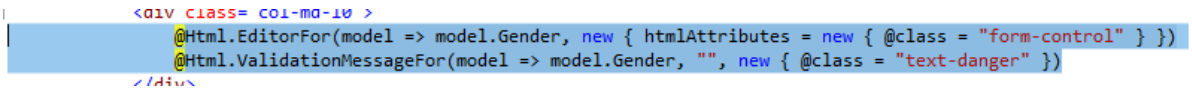
## Arbetsuppgift 1: Modifiera view

Steg 1: Dubbelklicka på view SignUp.cshtml.



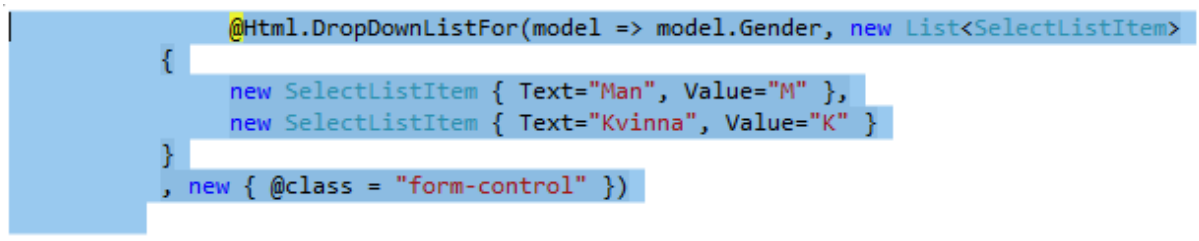
```
@Html.ValidationSummary(true, "", new { @class = "text-danger" })
<div class="form-group">
    @Html.LabelFor(model => model.LOOKUPRoleID, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.LOOKUPRoleID, new { htmlAttributes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

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

Steg 2: Ta bort blåmarkerad programkod, enligt bild ovan.



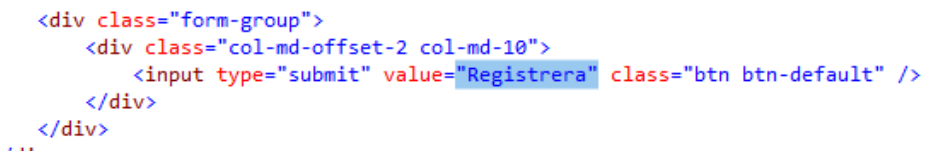
```
<div class="col-md-10">
    @Html.EditorFor(model => model.Gender, new { htmlAttributes = new { @class = "form-control" } })
    @Html.ValidationMessageFor(model => model.Gender, "", new { @class = "text-danger" })
</div>
```

Steg 3: Lokalisera programkod för Gender. Ersätt den med programkod enligt nedan:

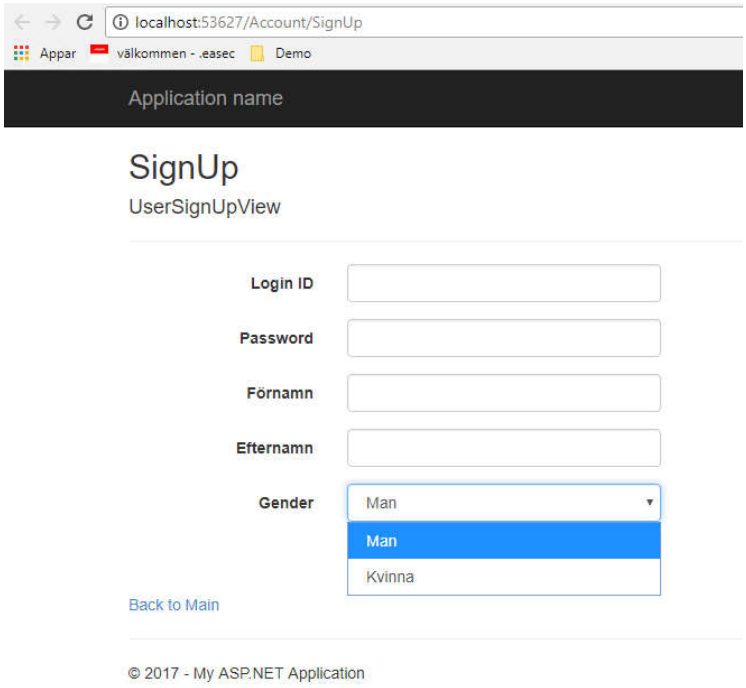


```
@Html.DropDownListFor(model => model.Gender, new List<SelectListItem>
{
    new SelectListItem { Text="Man", Value="M" },
    new SelectListItem { Text="Kvinna", Value="K" }
}
, new { @class = "form-control" })
```

Steg 4: Ändra text i knappen, enligt bild nedan:



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




Dina förändringar ger resultatet på bilden ovan.

Arbetsuppgift 2: Slå på Forms Authentication och skapa view för inloggning och utloggning

Steg 1: Vi börjar med att slå på Forms Authentication. Dubbelklicka på web.config, lägg till elementen <authentication> och <forms> under <system.web> elementet. Skall se ut så här:

```
<system.web>
  <authentication mode = "Forms">
    <forms loginUrl="~/Account/Login" defaultUrl="~/Home/Welcome"></forms>
  </authentication>
  <compilation debug="true" targetFramework="4.5.2" />
  <httpRuntime targetFramework="4.5.2" />
</system.web>
```

Om användaren har loggat på, skall hen skickas vidare till view Welcome.

Steg 2: Nästa steg är att skapa klass för inloggningssidan i View Model, lägg till följande programkod under klassen UserModel:

```
public class UserLoginView
{
    [Key]
    public int SYSUserID { get; set; }
    [Required(ErrorMessage = "*")]
    [Display(Name = "Login ID")]
    public string LoginName { get; set; }
    [Required(ErrorMessage = "*")]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public string Password { get; set; }
}
```

Steg 3: Dubbelklicka på UserManager.cs, lägg till följande metod:

```
public string GetUserPassword(string loginName)
{
    using (loginDBEntities db = new loginDBEntities())
    {
        var user = db.SYSUser.Where(o => o.LoginName.ToLower().Equals(loginName));
        if (user.Any())
            return user.FirstOrDefault().PasswordEncryptedText;
        else
            return string.Empty;
    }
}
```

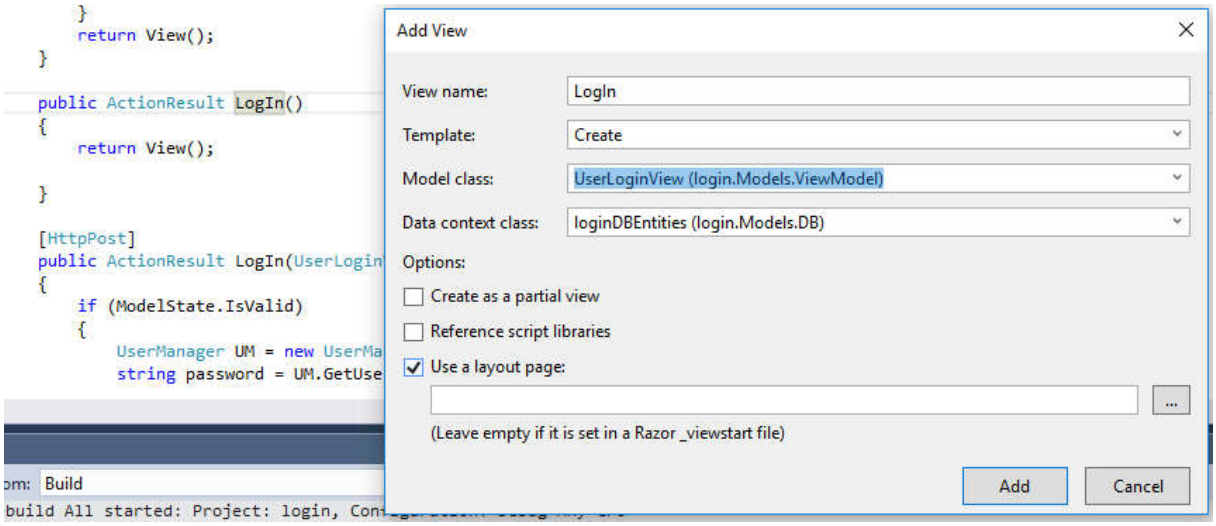
Steg 4: Dubbelklicka på AccountController.cs, lägg till följande metod:

```
public ActionResult LogIn()
{
    return View();

}
```

Steg 5: Klicka på Enter två gånger, skriv därefter in följande programkod:

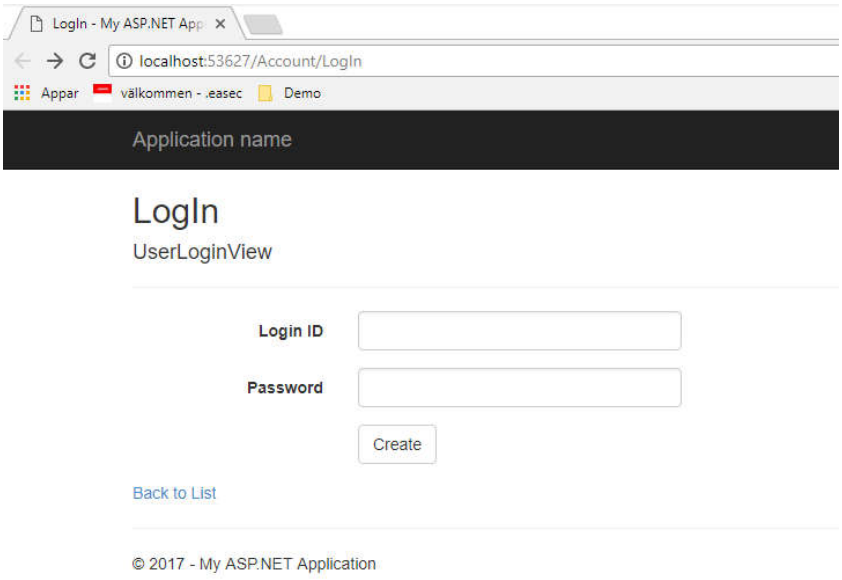
```
[HttpPost]
public ActionResult LogIn(UserLoginView ULV, string returnUrl)
{
    if (ModelState.IsValid)
    {
        UserManager UM = new UserManager();
        string password = UM.GetUserPassword(ULV.LoginName);

        if (string.IsNullOrEmpty(password))
            ModelState.AddModelError("", "Användarelogin eller lösenord felaktigt.");
        else
        {
            if (ULV.Password.Equals(password))
            {
                FormsAuthentication.SetAuthCookie(ULV.LoginName, false);
                return RedirectToAction("Welcome", "Home");
            }
            else
            {
                ModelState.AddModelError("", "Lösenord felaktigt");
            }

        }

    }
    // Har vi kommit hit, har något fallerat,
    return View(ULV);
}
```

Steg 6: Innan view för inloggning skapas, kontrollera att programkod är felfri, klicka på Build – Build Solution.




Steg 7: Skapa view för inloggning, genom att högerklicka på LogIn, välj Create i rutan för Template. Välj UserLoginView (login.Models.ViewModel) för Model class och loginDBEntities (login.Models.DB) för Data context class.

Klicka på Add.




Inloggningen ser ut som ovan, dags att modifiera denna lite.

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

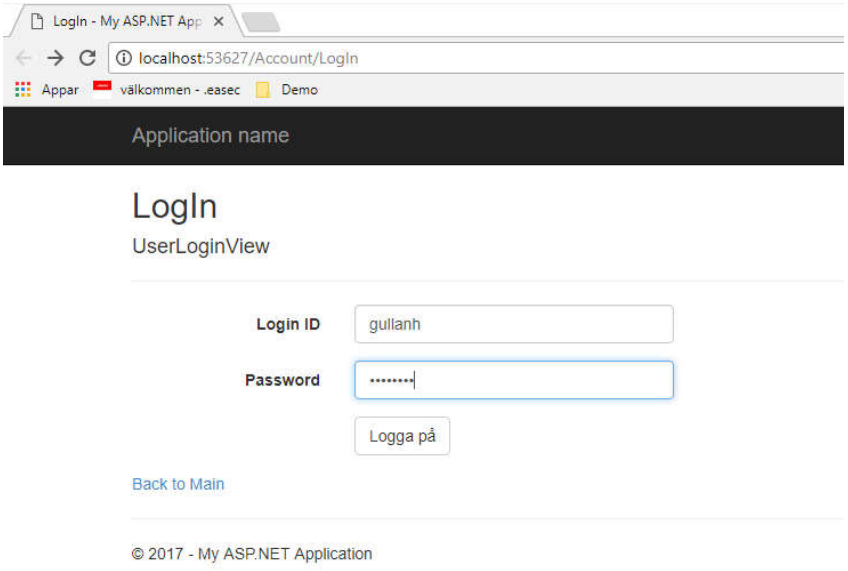
Steg 8: Ändra Create till Logga på, som bilden.

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

Steg 9: Ändra länk, enligt ovan.

Steg 10: Provkör din applikation genom att klicka på Debug – Start Debugging.

Logga på:




Användare hamnar på view som vi skapade i del 1.


Stäng ner webbläsare, avsluta Debugging.

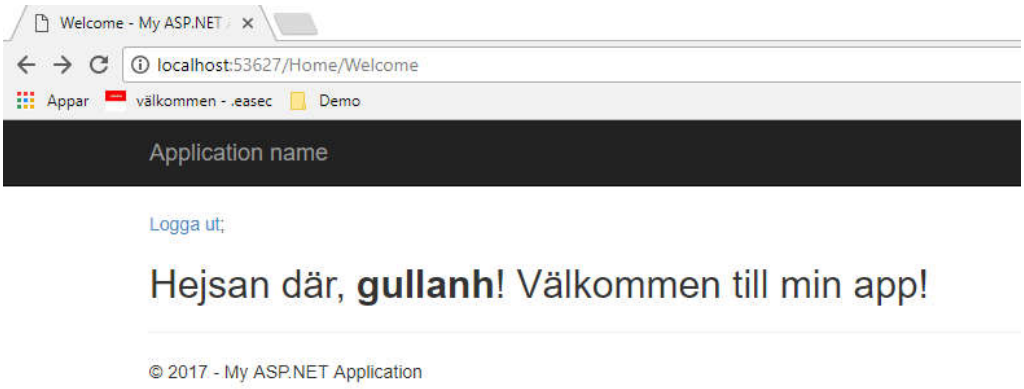
Steg 11: Nästa steg är att implementera funktion för utloggning. Lägg till följande metod i AccountController.cs:

```
[Authorize]
public ActionResult SignOut()
{
    FormsAuthentication.SignOut();
    return RedirectToAction("Index", "Home");
}
```

Steg 12: Modifiera Welcome.cs, enligt bild nedan:

```
<br />
@Html.ActionLink("Logga ut", "Signout", "Account");

<h2>Hejsan där, <b>@Context.User.Identity.Name</b>! Välkommen
```

Kommer att ge som resultat:



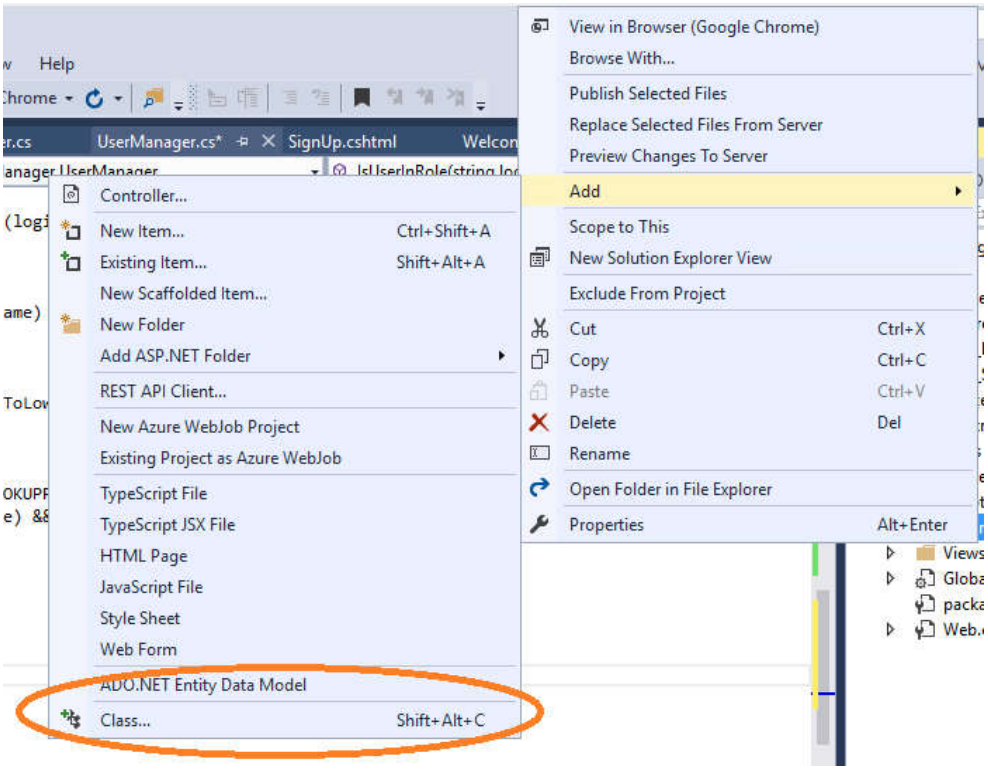

Arbetsuppgift 3: Implementera rollbaserad auktorisering

Steg 1: Dubbelklicka på UserManager.cs. Lägg till programkod enligt nedan:

```
public bool IsUserInRole(string loginName, string roleName)
{
    using (loginDBEntities db = new loginDBEntities())
    {
        SYSUser SU = db.SYSUser.Where(o => o.LoginName.ToLower().Equals(loginName))?.FirstOrDefault();
        if (SU != null)
        {
            var roles = from q in db.SYSUserRole
                        join r in db.LOOKUPRole on q.LOOKUPRoleID equals r.LOOKUPRoleID
                        where r.RoleName.Equals(roleName) && q.SYSUserID.Equals(SU.SYSUserID)
                        select r.RoleName;
            if (roles != null)
            {
                return roles.Any();
            }
        }
        return false;
    }
}
```

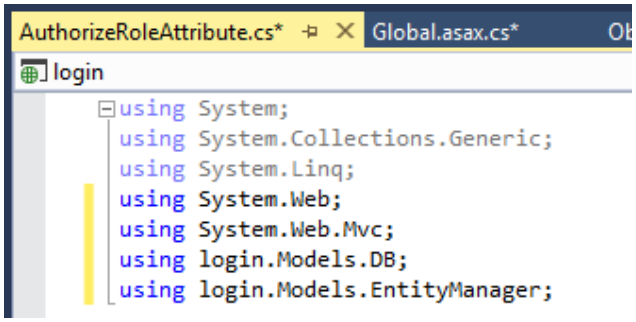
Skapa Custom Authorization Attribute Filter

Steg 2: Skapa ny katalog, genom att högerklicka på Login, välj sedan Add – New Folder.




Namnge katalogen till Security. Högerklicka på den nya katalog, klicka på Add – Class. Namnge klassen till: AuthorizeRoleAttribute.cs.

Steg 3: Modifiera programkod, så att den ser ut så här:



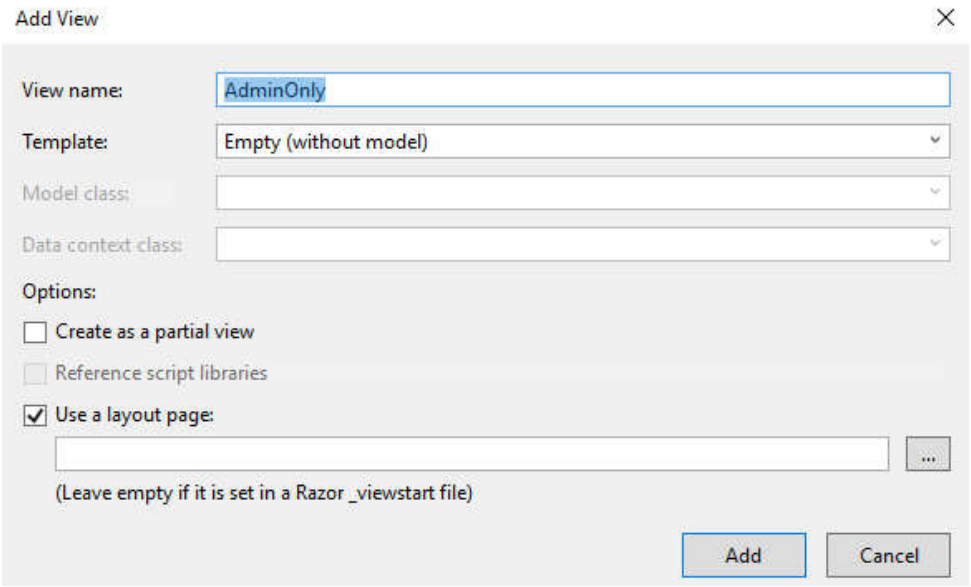

Steg 4: Lägg till följande programkod:

```
namespace login.Security
{
    public class AuthorizeRoleAttribute : AuthorizeAttribute
    {
        private readonly string[] userAssignedRoles;
        public AuthorizeRoleAttribute(params string[] roles)
        {
            this.userAssignedRoles = roles;
        }
        protected override bool AuthorizeCore(HttpContextBase httpContext)
        {
            bool authorize = false;
            using (loginDBEntities db = new loginDBEntities())
            {
                UserManager UM = new UserManager();
                foreach (var roles in userAssignedRoles)
                {
                    authorize = UM.IsUserInRole(httpContext.User.Identity.Name, roles);
                    if (authorize)
                        return authorize;
                }
            }
            return authorize;
        }
        protected override void HandleUnauthorizedRequest(AuthorizationContext filterContext)
        {
            filterContext.Result = new RedirectResult("~/Home/UnAuthorized");
        }
    }
}
```

Steg 5: Skifta till HomeController.cs, lägg till följande programkod:

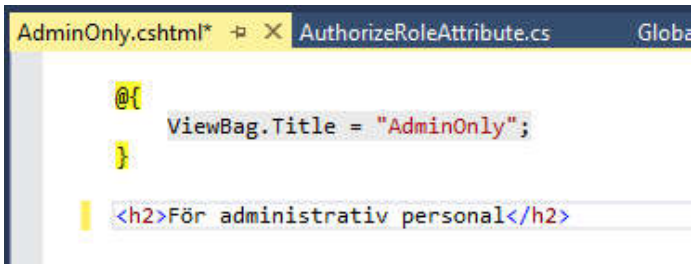
```
[AuthorizeRole("Admin")]
public ActionResult AdminOnly()
{
    return View();
}
public ActionResult UnAuthorized()
{
    return View();
}
```

Steg 6: Markera AdminOnly(), högerklicka och välj Add View.

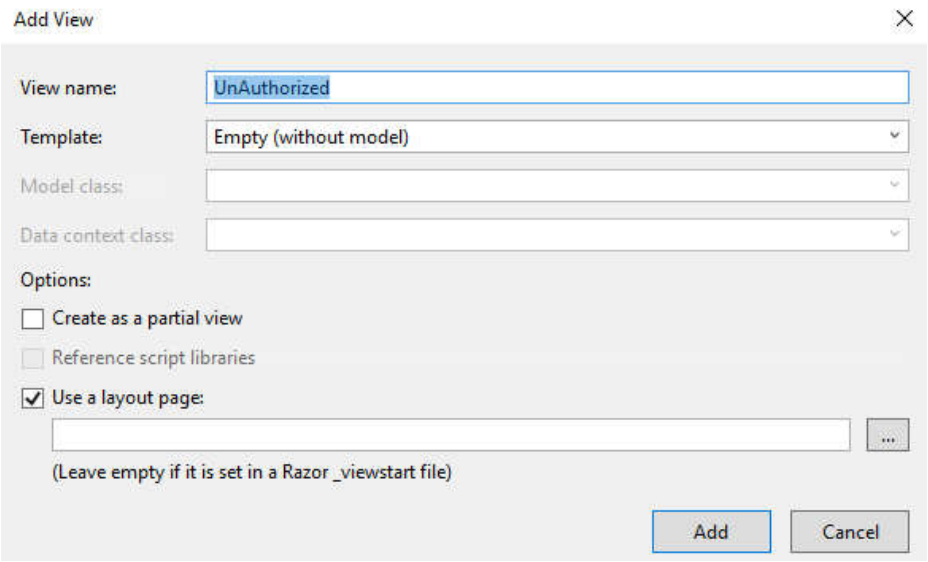



Klicka på Add.

Steg 7: Justera programkoden i view, enligt nedan:

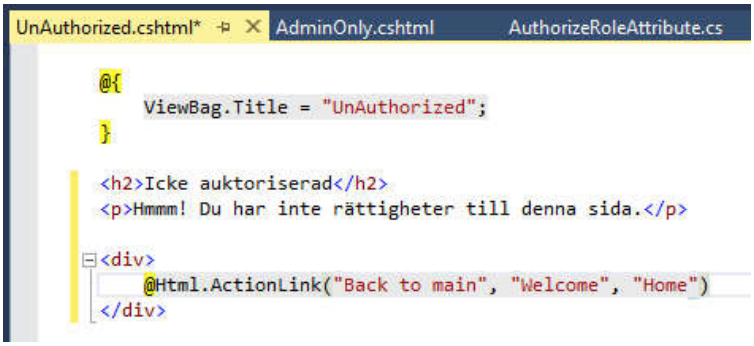



Steg 8: Dubbelklicka först på HomeController.cs, därefter markerar du UnAuthorized(), högerklicka och välj Add View.




Klicka på Add.

Steg 9: Justera programkoden i view, enligt nedan:

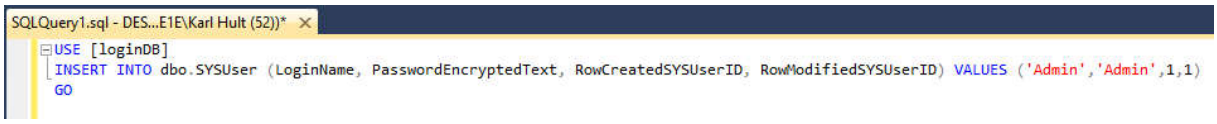



Arbetsuppgift 4: Lägg till lite testdata i databasen

Steg 1: Öppna SQL Management Studio, klicka på New Query, skriv in följande:

```
USE [loginDB]

INSERT INTO dbo.SYSUser (LoginName,
PasswordEncryptedText, RowCreatedSYSUserID,
RowModifiedSYSUserID) VALUES ('Admin','Admin',1,1)

GO
```

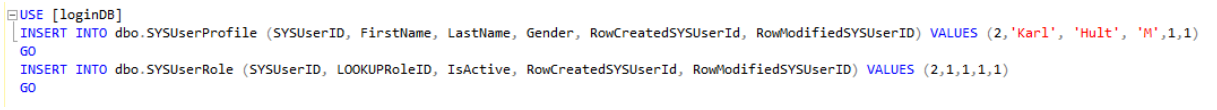
Markera din query och klicka sedan på Execute.




Steg 2: Fortsätt med denna query:

```
USE [loginDB]

INSERT INTO dbo.SYSUserProfile (SYSUserID, FirstName,
LastName, Gender, RowCreatedSYSUserId,
RowModifiedSYSUserID) VALUES (2,'Karl', 'Hult',
'M',1,1)

GO

INSERT INTO dbo.SYSUserRole (SYSUserID, LOOKUPRoleID,
IsActive, RowCreatedSYSUserId, RowModifiedSYSUserID)
VALUES (2,1,1,1,1)
```

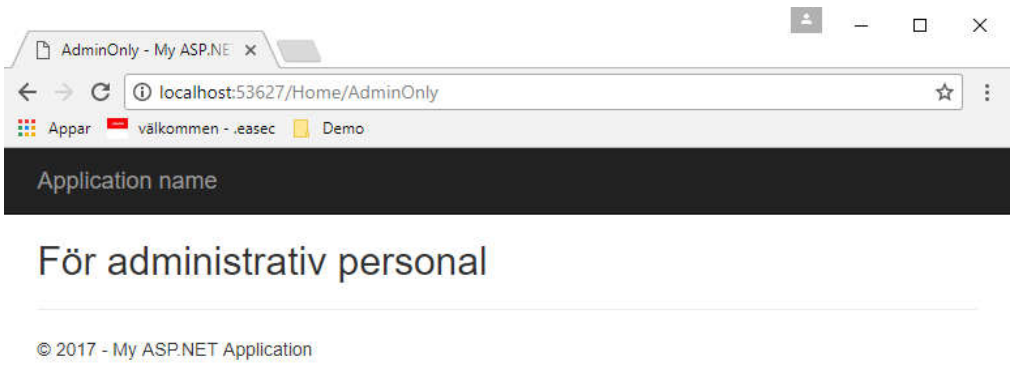
Markera din query och klicka sedan på Execute.




Arbetsuppgift 5: Testa funktionalitet

Steg 1: Klicka på Debug – Start Debugging.

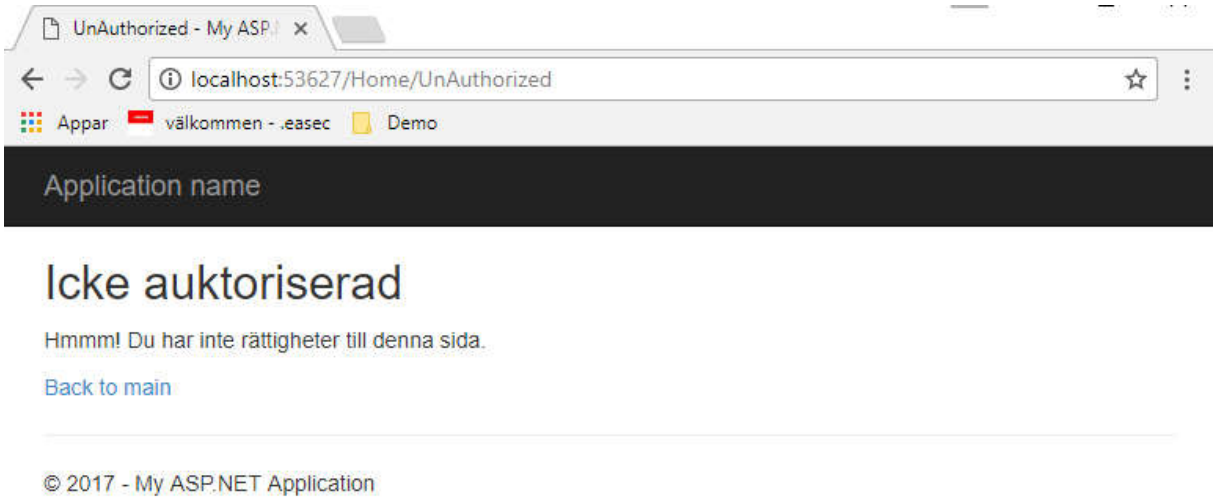
Steg 2: Logga på som Admin med lösenordet Admin. Under förutsättning att du körde SQL query tidigare.



Steg 3: Lägg till /AdminOnly i URL, sidan för auktoriserade användare skall visas.

Steg 4: Logga på användare som inte har rollen Admin.

Steg 5: Lägg till /AdminOnly i URL, sidan för icke auktoriserade användare skall visas.




Arbetsuppgift 6: Lägga till funktioner för att hämta och visa information

Lägg till klass i UserModel.cs

Steg 1:  I Solution Explorer, dubbelklicka på UserModel.cs.

Steg 2: Lägg till följande programkod:

```
public class UserProfileView
{
    [Key]
    public int SYSUserID { get; set; }
    public int LOOKUPRoleID { get; set; }
    public string RoleName { get; set; }
    public bool? IsRoleActive { get; set; }
    [Required(ErrorMessage = "*")]
    [Display(Name = "Login ID")]
    public string LoginName { get; set; }
    [Required(ErrorMessage = "*")]
    [Display(Name = "Password")]
    public string Password { get; set; }
    [Required(ErrorMessage = "*")]
    [Display(Name = "First Name")]
    public string FirstName { get; set; }
    [Required(ErrorMessage = "*")]
    [Display(Name = "Last Name")]
    public string LastName { get; set; }
    [Display(Name = "Kön")]
    public string Gender { get; set; }
}
```

Steg 3: Lägg till följande programkod, i UserModel.cs:

```
public class LOOKUPAvaiableRole
{
    [Key]
    public int LOOKUPRoleID { get; set; }
    public string RoleName { get; set; }
    public string RoleDescription { get; set; }
}
```

Steg 4: Lägg till följande programkod, i UserModel.cs:

```
public class Gender
{
    public string Text { get; set; }
    public string Value { get; set; }
}

public class UserRoles
{
    public int? SelectedRoleID { get; set; }
    public IEnumerable<LOOKUPAvaiableRole> UserRoleList { get; set; }
}

public class UserGender
{
    public string SelectedGender { get; set; }
    public IEnumerable<Gender> Gender { get; set; }
}

public class UserDataView
{
    public IEnumerable<UserProfileView> UserProfile { get; set; }
    public UserRoles UserRoles { get; set; }
    public UserGender UserGender { get; set; }
}
```

Steg 5: Nu kan det vara lämpligt att klicka på Build, för att kontrollera att programkod är felfri, klicka på Build – Build Solution.

Steg 6: I Solution Explorer, dubbelklicka på UserManager.cs.

Steg 7: Verifiera att: `using System.Collections.Generic;`, finns.

Steg 8: Lägg till följande programkod:

```
public List<LOOKUPAvaiableRole> GetAllRoles()
{
    using (loginDBEntities db = new loginDBEntities())
    {
        var roles = db.LOOKUPRole.Select(o => new LOOKUPAvaiableRole
        {
            LOOKUPRoleID = o.LOOKUPRoleID,
            RoleName = o.RoleName,
            RoleDescription = o.RoleDescription
        }).ToList();

        return roles;
    }
}
```

Steg 9: Lägg till följande programkod:

```
public int GetUserID(string loginName)
{
    using (loginDBEntities db = new loginDBEntities())
    {
        var user = db.SYSUser.Where(o => o.LoginName.Equals(loginName));
        if (user.Any())
            return user.FirstOrDefault().SYSUserID;
    }
    return 0;
}
```

Steg 10: Lägg till följande programkod:

```
public UserDataView GetUserDataView(string loginName)
{
    UserDataView UDV = new UserDataView();
    List<UserProfileView> profiles = GetAllUserProfiles();
    List<LOOKUPAvaiableRole> roles = GetAllRoles();

    int? userAssignedRoleID = 0, userID = 0;
    string userGender = string.Empty;

    userID = GetUserID(loginName);
    using (loginDBEntities db = new loginDBEntities())
    {
        userAssignedRoleID = db.SYSUserRole.Where(o => o.SYSUserID == userID)?.FirstOrDefault().LOOKUPRoleID;
        userGender = db.SYSUserProfile.Where(o => o.SYSUserID == userID)?.FirstOrDefault().Gender;
    }

    List<Gender> genders = new List<Gender>();
    genders.Add(new Gender { Text = "Man", Value = "M" });
    genders.Add(new Gender { Text = "Kvinna", Value = "K" });

    UDV.UserProfile = profiles;
    UDV.UserRoles = new UserRoles { SelectedRoleID = userAssignedRoleID, UserRoleList = roles };
    UDV.UserGender = new UserGender { SelectedGender = userGender, Gender = genders };
    return UDV;
}
```

```
public List<UserProfileView> GetAllUserProfiles()
{
    List<UserProfileView> profiles = new List<UserProfileView>();
    using (loginDBEntities db = new loginDBEntities())
    {
        UserProfileView UPV;
        var users = db.SYSUser.ToList();

        foreach(SYSUser u in db.SYSUser)
        {
            UPV = new UserProfileView();
            UPV.SYSUserID = u.SYSUserID;
            UPV.LoginName = u.LoginName;
            UPV.Password = u.PasswordEncryptedText;

            var SUP = db.SYSUserProfile.Find(u.SYSUserID);
            if(SUP != null)
            {
                UPV.FirstName = SUP.FirstName;
                UPV.LastName = SUP.LastName;
                UPV.Gender = SUP.Gender;
            }

            var SUR = db.SYSUserRole.Where(o => o.SYSUserID.Equals(u.SYSUserID));
            if (SUR.Any())
            {
                var userRole = SUR.FirstOrDefault();
                UPV.LOOKUPRoleID = userRole.LOOKUPRoleID;
                UPV.RoleName = userRole.LOOKUPRole.RoleName;
                UPV.IsRoleActive = userRole.IsActive;
            }
            profiles.Add(UPV);
        }
    }
    return profiles;
}
```

Arbetsuppgift 7: Lägg till metod

Steg 1: I Solution Explorer, dubbelklicka på HomeController.cs.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using login.Security;
using login.Models.ViewModel;
using login.Models.EntityManager;
```

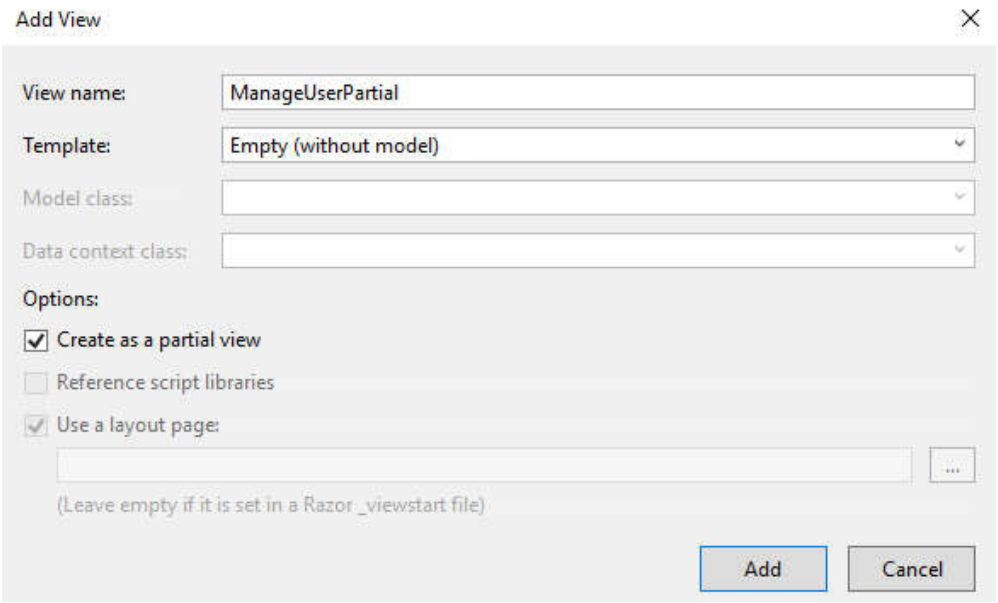
Steg 2: Verifiera att dessa finns, enligt bild ovan. Annars lägg till de som saknas.

Steg 3: Skriv in programkod enligt bild:

```
[AuthorizeRole("Admin")]
public ActionResult ManageUserPartial()
{
    if (User.Identity.IsAuthenticated)
    {
        string loginName = User.Identity.Name;
        UserManager UM = new UserManager();
        UserDataView UDV = UM.GetUserDataView(loginName);
        return PartialView(UDV);
    }
    return View();
}
```

Arbetsuppgift 8: Lägga till Partial View

Steg 1: Skapa Partial View, genom att högerklicka på metoden ManageUserPartial, välj sedan Add View.




Steg 2: Välj Empty (without model), klicka i boxrutan til vänster om Create as a partial view. Klicka på Add.

ManageUserPartial.cshtml* | AuthorizeRoleAttribute.cs | HomeController.cs | UserModel.cs | AdminOnly.

```
@model login.Models.ViewModel.UserDataView

<div>
    <h1>Lista över användare</h1>
    <span class="alert-success">@ViewBag.Message</span>
    <table class="table table-striped table-condensed tabel-hover">
        <<thead>
        <tr>
            <th>ID</th>
            <th>Login id</th>
            <th>Password</th>
            <th>Förnamn</th>
            <th>Efternamn</th>
            <th>Gender</th>

            <th colspan="2">Roll</th>
            <th></th>
            <th></th>
        </tr></thead>
        <tbody>
            @foreach (var i in Model.UserProfile)
            {
                <tr>
                    <td> @Html.DisplayFor(m => i.SYSUserID)</td>
                    <td> @Html.DisplayFor(m => i.LoginName)</td>
                    <td> @Html.DisplayFor(m => i.Password)</td>
                    <td> @Html.DisplayFor(m => i.FirstName)</td>
                    <td> @Html.DisplayFor(m => i.LastName)</td>
                    <td> @Html.DisplayFor(m => i.Gender)</td>
                    <td> @Html.DisplayFor(m => i.RoleName)</td>
                    <td> @Html.DisplayFor(m => i.LOOKUPRoleID)</td>
                    <td><a href="javascript:void(0)" class="lnkEdit">Edit</a></td>
                    <td><a href="javascript:void(0)" class="lnkDelete">Delete</a></td>
                </tr>
            }
        </tbody>
    </table>
</div>
```

Steg 3: Lägg till programkod enligt ovan.

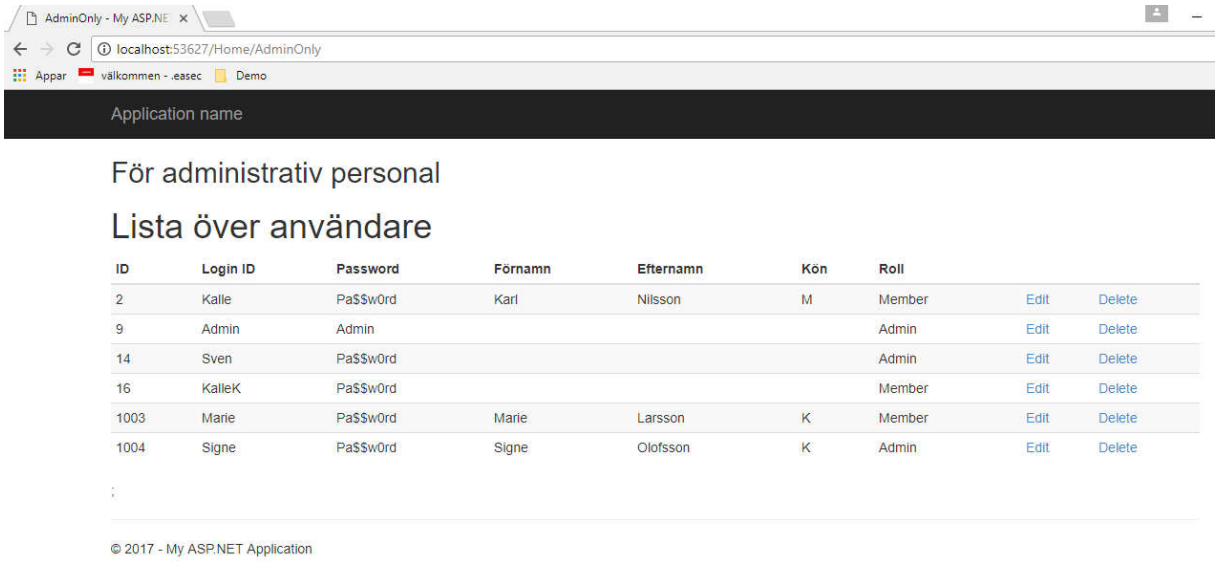
Steg 4: I Solution Explorer, dubbelklicka på AdminOnly.cshtml, lägg till följande:

```
<div id="divUserListContainer">
    @Html.Action("ManageUserPartial", "Home");
</div>
```

Steg 5: Klicka på Debug – Start Debugging.

Steg 6: Logga på som Admin, med lösenordet Admin.

Steg 7: Skriv in AdminOnly efter /Home/, klicka på Enter.



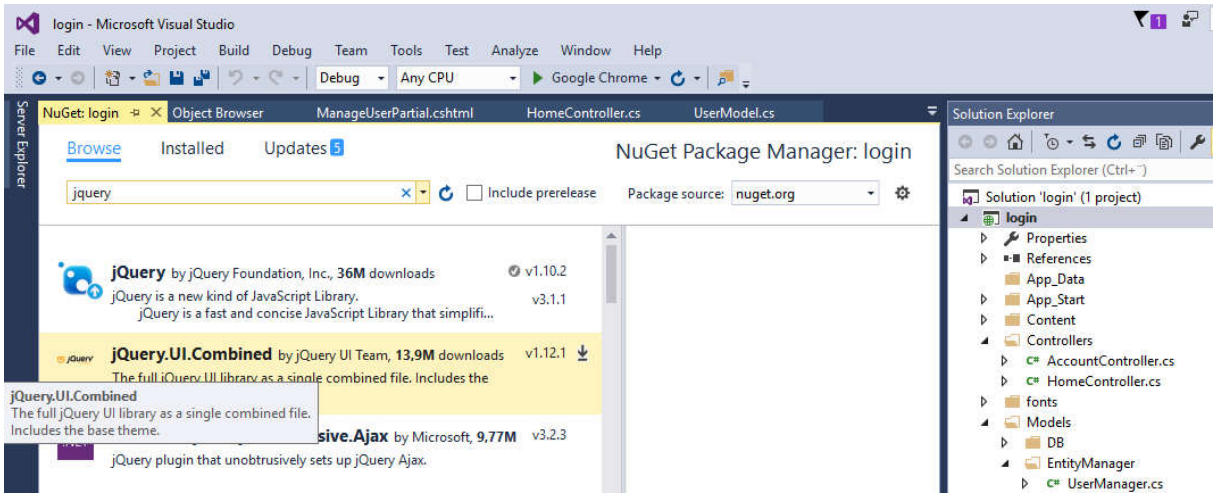

Resultatet kan se lite annorlunda ut beroende på hur många användare du har lagt till.

Arbetsuppgift 9: Lägga till möjlighet att editera och ta bort

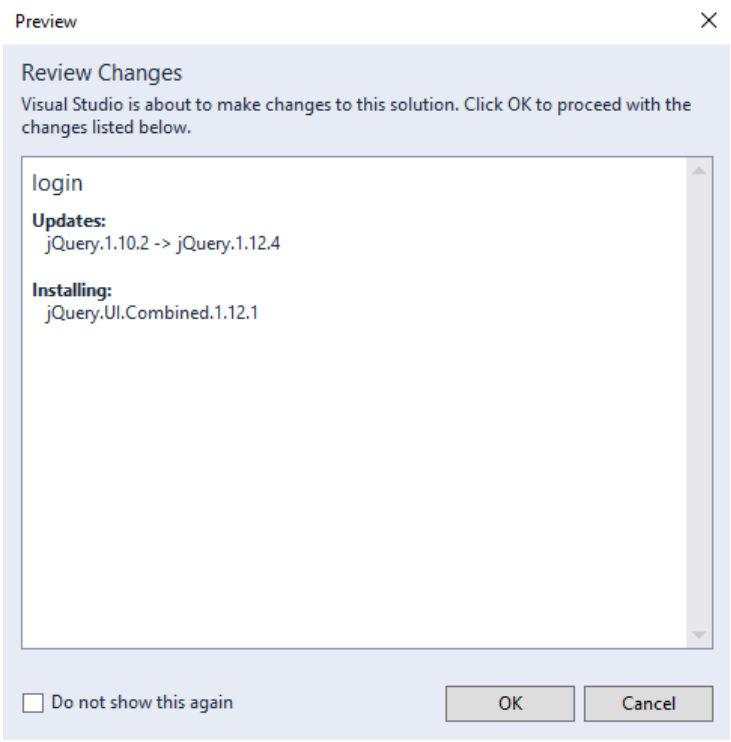
För dessa funktioner skall vi arbeta med jQuery.

Steg 1: Högerklicka på ditt projekt, klicka sedan på Manage Nuget Package. Klicka på Browse, skriv in jquery i sökrutan.

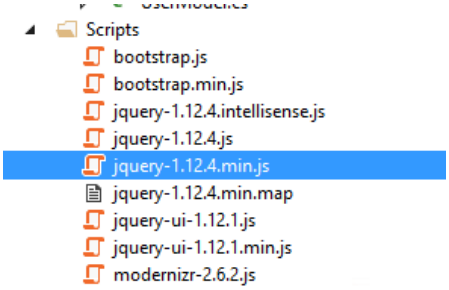
Steg 2: Välj jQuery.UI.Combined, se bild nedan:




Steg 3: Klicka på Install.




Steg 4: Klicka på OK, i dialogfönstret Review Changes.




Steg 5: Expandera katalog med namnet Scripts, verifiera att det finns ett antal skript här. Notera version på skripten.

Steg 6: Dubbelklicka på _Layout.cshtml under katalogen Shared.

Steg 7: Nederst på sidan, ta bort referens till jQuery.

```
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>
    <link href="~/Content/Site.css" rel="stylesheet" type="text/css" />
    <link href="~/Content/bootstrap.min.css" rel="stylesheet" type="text/css" />
    <script src="~/Scripts/modernizr-2.6.2.js"></script>
    <script src="~/Scripts/jquery-1.12.4.min.js"></script>
    <script src="~/Scripts/jquery-ui-1.12.1.min.js"></script>
    <link href="~/Content/themes/base/all.css" rel="stylesheet" type="text/css" />


</head>
```

Steg 8: I sektionen <HEAD>, lägg till referens för ~/Scripts/jquery-1.12.4.min.js, ~/Scripts/jquery-ui-1.12.1.min.js och /Content/themes/base/all.css, lägg denna sist av referenserna. Se bild ovan.

Steg 9: I Solution Explorer, dubbelklicka på klassen UserManager.cs, lägg till följande programkod:

```
public void UpdateUserAccount(UserProfileView user)
{
    using (loginDBEntities db = new loginDBEntities())
    {
        using (var dbContextTransaction = db.Database.BeginTransaction())
        {
            try
            {
                SYSUser SU = db.SYSUser.Find(user.SYSUserID);
                SU.LoginName = user.LoginName;
                SU.PasswordEncryptedText = user.Password;
                SU.RowCreatedSYSUserID = user.SYSUserID;
                SU.RowModifiedSYSUserID = user.SYSUserID;
                SU.RowCreatedDateTime = DateTime.Now;
                SU.RowModifiedDateTime = DateTime.Now;

                db.SaveChanges();
            }
```

Steg 10: Fortsätt med följande programkod:

```
var userProfile = db.SYSUserProfile.Where(o => o.SYSUserID == user.SYSUserID);
if (userProfile.Any())
{
    SYSUserProfile SUP = userProfile.FirstOrDefault();
    SUP.SYSUserID = SU.SYSUserID;
    SUP.FirstName = user.FirstName;
    SUP.LastName = user.LastName;
    SUP.Gender = user.Gender;
    SUP.RowCreatedSYSUserID = user.SYSUserID;
    SUP.RowModifiedSYSUserID = user.SYSUserID;
    SUP.RowCreatedDateTime = DateTime.Now;
    SUP.RowModifiedDateTime = DateTime.Now;

    db.SaveChanges();
}
```

Steg 11: Fortsätt med följande programkod:

```
if (user.LOOKUPRoleID > 0)
{
    var userRole = db.SYSUserRole.Where(o => o.SYSUserID == user.SYSUserID);
    SYSUserRole SUR = null;
    if (userRole.Any())
    {
        SUR = userRole.FirstOrDefault();
        SUR.LOOKUPRoleID = user.LOOKUPRoleID;
        SUR.SYSUserID = user.SYSUserID;
        SUR.IsActive = true;
        SUR.RowCreatedSYSUserID = user.SYSUserID;
        SUR.RowModifiedSYSUserID = user.SYSUserID;
        SUR.RowCreatedDateTime = DateTime.Now;
        SUR.RowModifiedDataTime = DateTime.Now;
    }

    else
    {
        SUR = new SYSUserRole();
        SUR.LOOKUPRoleID = user.LOOKUPRoleID;
        SUR.SYSUserID = user.SYSUserID;
        SUR.IsActive = true;
        SUR.RowCreatedSYSUserID = user.SYSUserID;
        SUR.RowModifiedSYSUserID = user.SYSUserID;
        SUR.RowCreatedDateTime = DateTime.Now;
        SUR.RowModifiedDataTime = DateTime.Now;
        db.SYSUserRole.Add(SUR);
    }
    db.SaveChanges();
}
dbContextTransaction.Commit();
```

Steg 12: Sista biten:

```
            }
            catch
            {
                dbContextTransaction.Rollback();
            }
        }
```

Här är hela klassen:

```
public void UpdateUserAccount(UserProfileView user)
        {
            using (loginDBEntities db = new loginDBEntities())
            {
                using (var dbContextTransaction =
db.Database.BeginTransaction())
                {
                    try
                    {
                        SYSUser SU = db.SYSUser.Find(user.SYSUserID);
                        SU.LoginName = user.LoginName;
                        SU.PasswordEncryptedText = user.Password;
                        SU.RowCreatedSYSUserID = user.SYSUserID;
                        SU.RowModifiedSYSUserID = user.SYSUserID;
                        SU.RowCreatedDateTime = DateTime.Now;
```

```
                        SU.RowModifiedDateTime = DateTime.Now;

                        db.SaveChanges();

                        var userProfile = db.SYSUserProfile.Where(o =>
o.SYSUserID == user.SYSUserID);
                        if (userProfile.Any())
                        {
                            SYSUserProfile SUP =
userProfile.FirstOrDefault();
                            SUP.SYSUserID = SU.SYSUserID;
                            SUP.FirstName = user.FirstName;
                            SUP.LastName = user.LastName;
                            SUP.Gender = user.Gender;
                            SUP.RowCreatedSYSUserID = user.SYSUserID;
                            SUP.RowModifiedSYSUserID = user.SYSUserID;
                            SUP.RowCreatedDateTime = DateTime.Now;
                            SUP.RowModifiedDateTime = DateTime.Now;

                            db.SaveChanges();
                        }

                        if (user.LOOKUPRoleID > 0)
                        {
                            var userRole = db.SYSUserRole.Where(o =>
o.SYSUserID == user.SYSUserID);
                            SYSUserRole SUR = null;
                            if (userRole.Any())
                            {
                                SUR = userRole.FirstOrDefault();
                                SUR.LOOKUPRoleID = user.LOOKUPRoleID;
                                SUR.SYSUserID = user.SYSUserID;
                                SUR.IsActive = true;
                                SUR.RowCreatedSYSUserID = user.SYSUserID;
                                SUR.RowModifiedSYSUserID = user.SYSUserID;
                                SUR.RowCreatedDateTime = DateTime.Now;
                                SUR.RowModifiedDataTime = DateTime.Now;
                            }

                            else
                            {
                                SUR = new SYSUserRole();
                                SUR.LOOKUPRoleID = user.LOOKUPRoleID;
                                SUR.SYSUserID = user.SYSUserID;
                                SUR.IsActive = true;
                                SUR.RowCreatedSYSUserID = user.SYSUserID;
                                SUR.RowModifiedSYSUserID = user.SYSUserID;
                                SUR.RowCreatedDateTime = DateTime.Now;
                                SUR.RowModifiedDataTime = DateTime.Now;
                                db.SYSUserRole.Add(SUR);
                            }
                            db.SaveChanges();
                        }
                        dbContextTransaction.Commit();
                    }
                    catch
                    {
                        dbContextTransaction.Rollback();
                    }
                }
```

```
        }
    }
```

Steg 13: Klicka på Build – Rebuild Solution, för att kontrollera att programkod är felfri.

Arbetsuppgift 10: Lägga till metod för att uppdatera användare

Steg 1: I Solution Explorer, dubbelklicka på HomeController.cs.

Steg 2: Lägg till följande programkod:

```
[AuthorizeRole("Admin")]
public ActionResult UpdateUserData(int userID, string loginName, string password, string firstname,
    string lastname, string gender, int roleID = 0)
{
    UserProfileView UPV = new UserProfileView();
    UPV.SYSUserID = userID;
    UPV.LoginName = loginName;
    UPV.Password = password;
    UPV.FirstName = firstname;
    UPV.LastName = lastname;
    UPV.Gender = gender;

    if (roleID > 0)
        UPV.LOOKUPRoleID = roleID;
    UserManager UM = new UserManager();
    UM.UpdateUserAccount(UPV);

    return Json(new { success = true });
}
```

Arbetsuppgift 11: Modifiera ManageUserPartial View

Steg 1: I SolutionExplorer under Views, dubbelklicka på ManageUserPartial.cshtml.

Steg 2: Skriv in programkod enligt nedan:


```
<div id="divEdit" style="display:none">
    <input type="hidden" id="hidID" />
    <table>
        <tr>
            <td>Login Name</td>
            <td><input type="text" id="txtLoginName" class="form-control" /></td>
        </tr>
        <tr>
            <td>Password</td>
            <td><input type="text" id="txtPassword" class="form-control" /></td>
        </tr>
        <tr>
            <td>Förnamn</td>
            <td><input type="text" id="txtFirstName" class="form-control" /></td>
        </tr>
        <tr>
            <td>Efternamn</td>
            <td><input type="text" id="txtLastName" class="form-control" /></td>
        </tr>
```

Steg 3: Fortsätt med nästa del programkod, tar vid efter avslutande </tr>:

```
        <tr>
            <td>Kön</td>
            <td>@Html.DropDownListFor(o => o.UserGender.SelectedGender,
            new SelectList(Model.UserGender.Gender, "Value", "Text"),
            "",
            new { id = "ddlGender", @class="Form-Control" })
            </td>
        </tr>
        <tr>
            <td>Roll</td>
            <td>@Html.DropDownListFor(o => o.UserRoles.SelectedRoleID,
 new SelectList(Model.UserRoles.UserRoleList, "LOOKUPRoleID", "RoleName"),
 "",
new { id = "ddlRoles", @class="form-control" })
</td>
        </tr>

    </table>
</div>
```

Arbetsuppgift 12: Lägga till JavaScript till view ManageUserPartial.cshtml

```
<script type="text/javascript">

    $(function () {


        var initDialog = function (type) {

            var title = type;
```

```
            $("#divEdit").dialog({
                autoOpen: false,
                modal: true,
                title: type + ' användare',
                width: 360,
                dialogClass: 'mod-no-close',
                buttons: {
                    Spara: function () {
                        var id = $("#hidID").val();
                        var role = $("#ddlRoles").val();
                        var loginName =
$("#txtLoginName").val();
                        var loginPass =
$("#txtPassword").val();
                        var fName = $("#txtFirstName").val();
                        var lName = $("#txtLastName").val();
                        var gender = $("#ddlGender").val();

                        UpdateUser(id, loginName, loginPass,
fName, lName, gender, role);
                        $(this).dialog("destroy");
                    },
                    Avbryt: function () {
$(this).dialog("destroy"); }
                }
            });
        }
```

```
        function UpdateUser(id, logName, logPass, fName,
lName, gender, role) {
            $.ajax({
                type: "POST",
                url: "@(Url.Action("UpdateUserData","Home"))",
                data: { userID: id, loginName: logName,
password: logPass, firstName: fName, lastName: lName, gender:
gender, roleID: role },
                success: function (data) {

$("#divUserListContainer").load("@(Url.Action("ManageUserParti
al","Home", new { status ="update" }))");
                },
                error: function (error) {
                    //to do:
                }
            });
        }


        $("a.lnkEdit").on("click", function () {
            initDialog("Editera");
            $(".alert-success").empty();
            var row = $(this).closest('tr');


$("#hidID").val(row.find("td:eq(0)").html().trim());

$("#txtLoginName").val(row.find("td:eq(1)").html().trim())

$("#txtPassword").val(row.find("td:eq(2)").html().trim())
```

```
$("#txtFirstName").val(row.find("td:eq(3)").html().trim())

$("#txtLastName").val(row.find("td:eq(4)").html().trim())

$("#ddlGender").val(row.find("td:eq(5)").html().trim())

            $("#ddlRoles").val(row.find("td:eq(7) >
input").val().trim());


            $("#divEdit").dialog("open");

            return false;

        });
</script>
```

Arbetsuppgift 14: Modifiera ManageUserPartial Action

Steg 1: Dubbelklicka på HomeController.cs.

```
[AuthorizeRole("Admin")]
public ActionResult ManageUserPartial(string status = "")
{
    if (User.Identity.IsAuthenticated)
    {
        string loginName = User.Identity.Name;
        UserManager UM = new UserManager();
        UserDataView UDV = UM.GetUserDataView(loginName);

        string message = string.Empty;
        if (status.Equals("update"))
            message = "Lyckad uppdatering!";
        else if (status.Equals("delete"))
            message = "Lyckad borttagning!";

        ViewBag.Message = message;

        return PartialView(UDV);
    }
    return RedirectToAction("Index", "Home");
}
```

Steg 2: Modifiera denna enligt programkod ovan.

Arbetsuppgift 15: Modifiera View

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

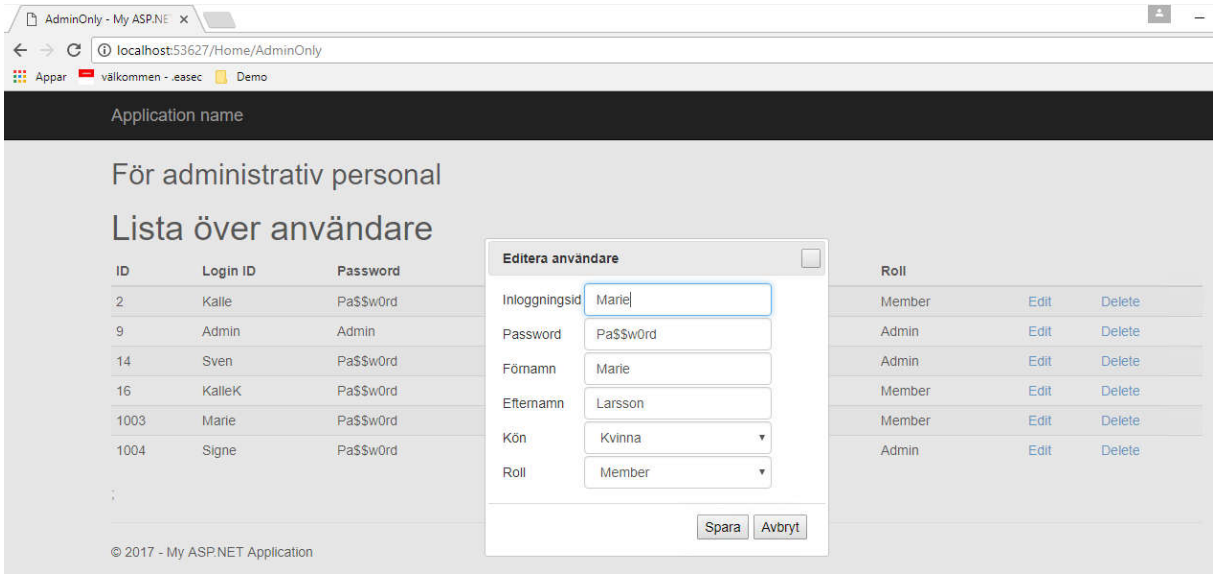
Steg 1: Dubbelklicka på AdminOnly.cshtml, lägg till Layout = ”~/Views/shared/_Layout.cshtml”;, enligt bild ovan.

Steg 2: Gör samma sak med Index.cshtml, Welcome.cshtml och SignUp.cshtml.

Arbetsuppgift 16: Provkör applikationen

Steg 1: Klicka på Debug – Start Debugging.

Steg 2: Via /Account/LogIn, logga på som Admin med lösenordet Admin.




Steg 3: Skriv in AdminOnly efter /Home. Klicka på Edit för någon användare. Ändra någon eller några parametrar, klicka på Spara.


Observera att informationen ändras. Stäng ner webbläsare, avsluta debugging.

Arbetsuppgift 17: Implementera borttagning

Steg 1: Dubbelklicka på UserMananger.cs.

Steg 2: Lägg till följande programkod:

```
public void DeleteUser(int userID)
{
    using (loginDBEntities db = new loginDBEntities())
    {
        using (var dbContextTransaction = db.Database.BeginTransaction())
        {
            try
            {

                var SUR = db.SYSUserRole.Where(o => o.SYSUserID == userID);
                if (SUR.Any())
                {
                    db.SYSUserRole.Remove(SUR.FirstOrDefault());
                    db.SaveChanges();
                }

                var SUP = db.SYSUserProfile.Where(o => o.SYSUserID == userID);
                if (SUP.Any())
                {
                    db.SYSUserProfile.Remove(SUP.FirstOrDefault());
                    db.SaveChanges();
                }

                var SU = db.SYSUser.Where(o => o.SYSUserID == userID);
                if (SU.Any())
                {
                    db.SYSUser.Remove(SU.FirstOrDefault());
                    db.SaveChanges();
                }

                dbContextTransaction.Commit();
            }
            catch
            {
                dbContextTransaction.Rollback();
            }
        }
    }
}
```

Steg 3: Dubbelklicka på HomeController.cs.

```
[AuthorizeRole("Admin")]
public ActionResult DeleteUser(int userID)
{
    UserManager UM = new UserManager();
    UM.DeleteUser(userID);
    return Json(new { success = true });
}
```

Steg 4: Lägg till programkod enligt ovan.

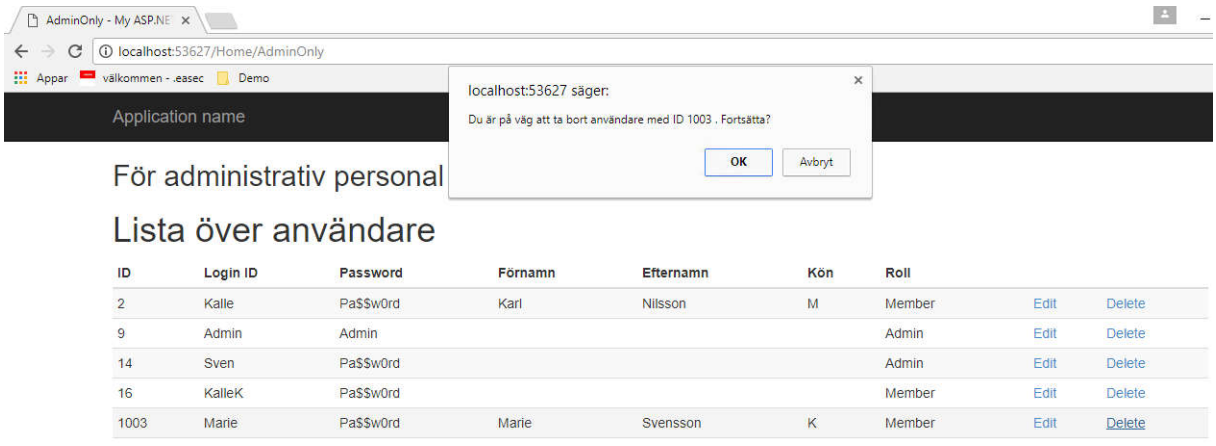
Steg 5: Dubbelklicka på ManageUserPartial.cshtml. Lägg till följande JavaScript innan avslutande </script>:

```
        function DeleteUser(id) {
            $.ajax({
                type: "POST",
                url: "@(Url.Action("DeleteUser","Home"))",
                data: { userID: id },
                success: function (data) {

$("#divUserListContainer").load("@(Url.Action("ManageUserPartial","Home", new { status
="delete" }))");
                },
                error: function (error) { }
            });
        }

        $("a.lnkDelete").on("click", function () {
            var row = $(this).closest('tr');
            var id = row.find("td:eq(0)").html().trim();
            var answer = confirm("Du är på väg att ta bort användare med ID " + id + "
. Fortsätta?");
            if (answer)
                DeleteUser(id);
            return false;
        });
    });
```

Steg 6: Klicka på Debug – Start Debugging.

Steg 7: Via /Account/LogIn, logga på som Admin med lösenordet Admin.

Steg 8: Skriv in AdminOnly efter /Home. Klicka på Delete för någon användare.




Steg 9: Klicka på OK och användare kommer att tas bort.

Steg 10: Avsluta webbläsaren, avsluta debugging.

Arbetsuppgift 18: Lägga till möjlighet för användare att editera sin profil (extra)

Steg 1: I Solution Explorer, dubbelklicka på UserManager.cs.

Steg 2: Lägg till programkod nedan:

```
public UserProfileView GetUserProfile(int userID)
{
    UserProfileView UPV = new UserProfileView();
    using (loginDBEntities db = new loginDBEntities())
    {
        var user = db.SYSUser.Find(userID);
        if (user != null)
        {
            UPV.SYSUserID = user.SYSUserID;
            UPV.LoginName = user.LoginName;
            UPV.Password = user.PasswordEncryptedText;

            var SUP = db.SYSUserProfile.Find(userID);
            if (SUP != null)
            {
                UPV.FirstName = SUP.FirstName;
                UPV.LastName = SUP.LastName;
                UPV.Gender = SUP.Gender;
            }

            var SUR = db.SYSUserRole.Find(userID);
            if (SUR != null)
            {
                UPV.LOOKUPRoleID = SUR.LOOKUPRoleID;
                UPV.RoleName = SUR.LOOKUPRole.RoleName;
                UPV.IsRoleActive = SUR.IsActive;
            }
        }
    }
    return UPV;
}
```

Steg 3: I Solution Explorer, dubbelklicka på HomeController.cs.

Steg 4: Lägg till följande programkod:

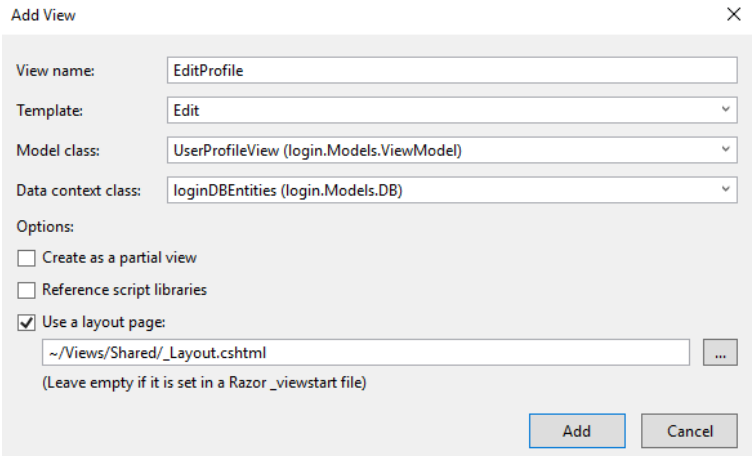
```
[Authorize]
public ActionResult EditProfile()
{
    string loginName = User.Identity.Name;
    UserManager UM = new UserManager();
    UserProfileView UPV = UM.GetUserProfile(UM.GetUserID(loginName));
    return View(UPV);
}


[HttpPost]
[Authorize]
public ActionResult EditProfile(UserProfileView profile)
{
    if (ModelState.IsValid)
    {
        UserManager UM = new UserManager();
        UM.UpdateUserAccount(profile);

        ViewBag.Status = "Lyckad uppdatering!";
    }
    return View(profile);
}
```

Programkoden är komponerad av två metoder, första EditProfile() kommer att kallas på när användare vill modifiera sin profil. Den hämtar information om profilen genom att kalla på metoden GetUserProfile() och skickar med parameter SYSUserID. Den andra metoden är av typer overload, som kommer att kallas på under POST request. Först kontrolleras att alla fält går igenom

valideringen och inte tomma. Sedan anropas metoden UpdateUserAccount() och skickar vidare model UserProfileView till metoden. UpdateUserAccount() är metoden som exekverar själva lagringen.

Bägge metoderna är dekorerade med [Authorize], för att försäkra att användare är inloggade.

Steg 5: Generera View, genom att högerklicka på metoden EditProfile(), välj Add View.




Ändra Template till Edit, ange Model class till UserProfileView (login.Models.ViewModel), klicka bort eventuell markering för Create as a partial view. Referera till din layoutfil.

Klicka på Add.

```
<h2>Redigera din profil</h2>
```

Steg 6: Börja med att ändra rubrik för sidan, enligt ovan.

Steg 7: Ta bort onödiga fält, LOOKUPRoleID and IsRoleActive.

Steg 8: Modifiera att roll inte kan förändras.

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

Steg 9: Ta bort blåmarkerad programkod, enligt bild ovan.

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

Steg 10: Lokalisera programkod för Gender. Ersätt den med programkod enligt nedan:

```
@Html.DropDownListFor(model => model.Gender, new List<SelectListItem> {
    new SelectListItem { Text="Man", Value="M" },
    new SelectListItem { Text="Kvinna", Value="K" }
}, new { @class = "form-control" })
```

Steg 11: Ändra text i knappen, enligt bild nedan:

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

Steg 12: Ändra @Html.ActionLink, enligt nedan.

```
<div>
    @Html.ActionLink("Back", "Welcome")
</div>
```

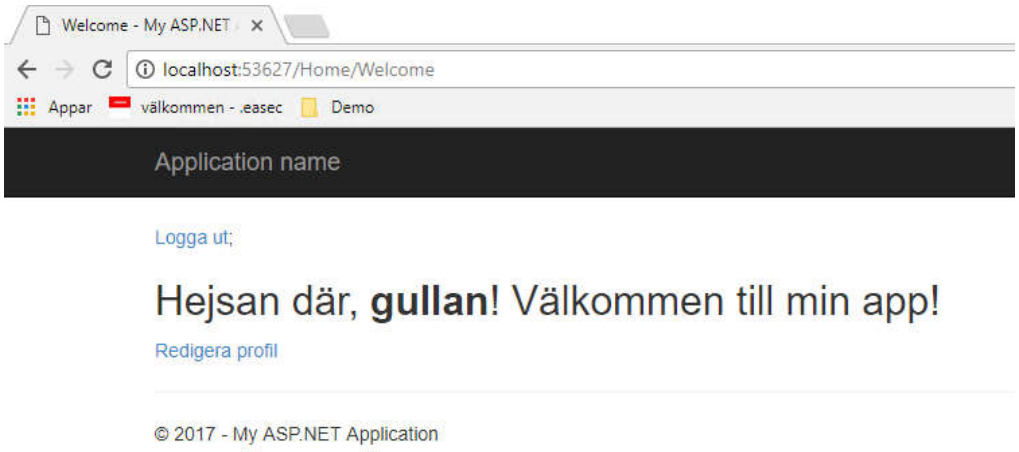
Steg 13: I Solution Explorer, dubbelklicka på Welcome.cshtml.

Steg 14: Underst, lägg till programkod enligt nedan:

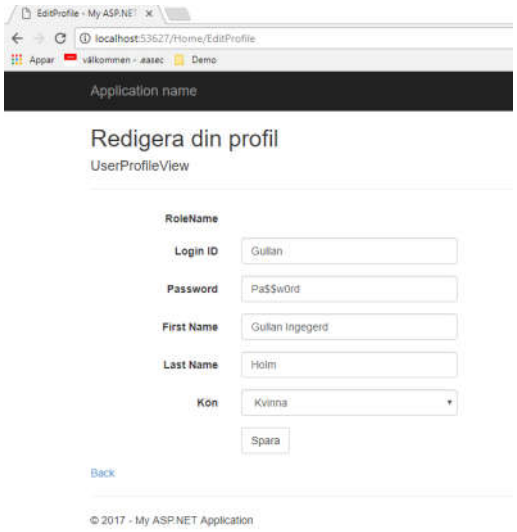
```
@Html.ActionLink("Redigera profil", "EditProfile", "Home")
```

Steg 15: Klicka på Debug – Start Debugging.

Steg 16: Logga på som ”vanlig” användare.



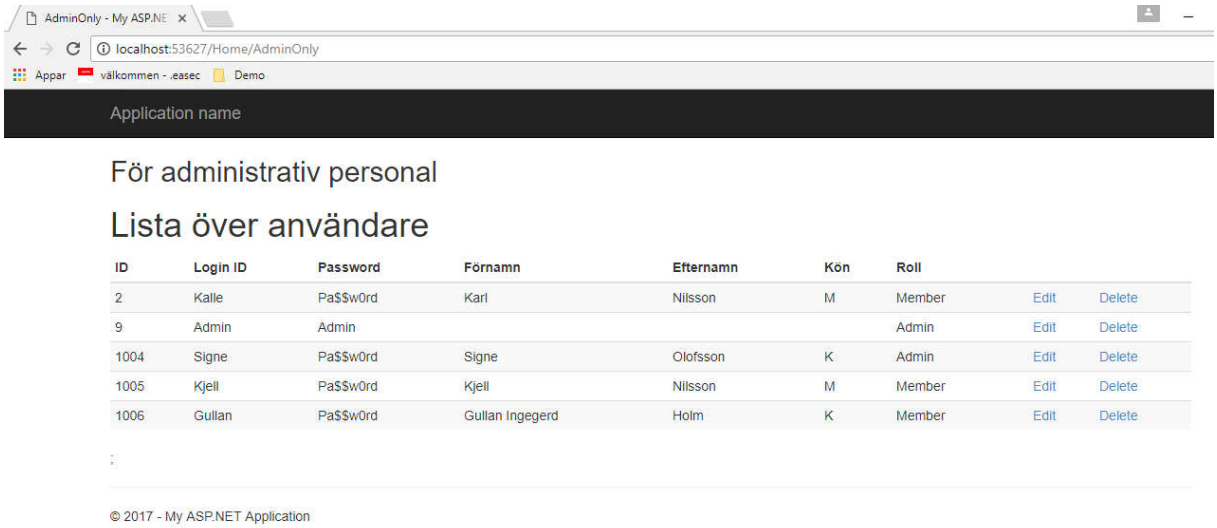

Steg 17: Klicka på länken Redigera profil.




Steg 18: Gör någon förändring, klicka på Spara. Klicka sedan på Back – Logga ut.

Steg 19: Logga på som Admin, med lösenordet Admin.

Steg 20: Skriv in AdminOnly efter /Home. Verifiera att dina förändringar visas.




Steg 21: Spara din lösning.