Badsha Rahim

How to make Custom Login, Register, and Logout in MVC 4 using Razor and EF

时间2013-06-07 13:45:34 [C-Sharpcorner Latest Articles](http://www.tuicool.com/sites/QFb2ym) [相似文章](http://www.tuicool.com/articles/dup?id=yuquYz) (0) 原文  [http://www.c-sharpcorner.com/UploadFile/raj1979/how-to-make-custom-login-register-and-logout-in-mvc-4-usin/](http://www.tuicool.com/articles/goto?id=yuquYz) [添加到推刊](http://www.tuicool.com/articles/yuquYz#add-article-to-kan)

收藏到推刊 [创建推刊](http://www.tuicool.com/articles/yuquYz#new-kan)

收 藏 取消

已收藏到推刊！

请填写推刊名   
描述不能大于100个字符!   
权限设置： 公开 仅自己可见

创建 取消

In this article you will learn how to make login, register, and logout screens with real time functionality using razor and entity framework data model.

**What is Entity Framework**

Entity Framework (EF) is an object-relational mapper that enables .NET developers to work with relational data using domain-specific objects. It eliminates the need for most of the data-access code that developers usually need to write. For more see:  [***http://msdn.microsoft.com/en-us/data/ef.aspx***](http://msdn.microsoft.com/en-us/data/ef.aspx)

**What is MVC?**

The Model-View-Controller (MVC) pattern separates the modeling of the domain, the presentation, and the actions based on user input into three separate classes [Burbeck92]:

**Model** - The model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller).

**View** - The view manages the display of information.

**Controller** - The controller interprets the mouse and keyboard inputs from the user, informing the model and/or the view to change as appropriate.

**Getting Started**

1. Create a new project; first open Visual Studio 2012
2. Then go to "File" => "New" => "Project..."
3. Select Web in installed templates
4. Select ASP.NET MVC 4 Web Application
5. Enter the Name and choose the location
6. Click OK

Now add a new ADO.NET Entity data Model.

Image 1.

To know how to configure ADO.NET Entity Data Model please read this article-

[***http://www.c-sharpcorner.com/uploadfile/raj1979/building-Asp-Net-mvc-web-applications-using-ado-net-entity-data-model/***](http://www.c-sharpcorner.com/uploadfile/raj1979/building-Asp-Net-mvc-web-applications-using-ado-net-entity-data-model/)

So let's move ahead without wasting time, here this is my data model scenario.

Image 2.

This is my model class which is generated when we configures data model, I just made slight changes.

//------------------------------------------------------------------------------

// <auto-generated>

//    This code was generated from a template.

//

//    Manual changes to this file may cause unexpected behavior in your application.

//    Manual changes to this file will be overwritten if the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace LoginInMVC4WithEF.Models

{

    using System;

    using System.Collections.Generic;

    using System.ComponentModel.DataAnnotations;

    public partial class Registration

    {

        public int UserId { get ; set ; }

        [ Required ]

        [EmailAddress]

        [ StringLength (150)]

        [ Display (Name = "Email Address: " )]

        public string Email { get ; set ; }

        [ Required ]

        [ DataType ( DataType .Password)]

        [ StringLength (150, MinimumLength = 6)]

        [ Display (Name = "Password: " )]

        public string Password { get ; set ; }

        public string PasswordSalt { get ; set ; }

        [ Required ]

        [ Display (Name = "First Name: " )]

        public string FirstName { get ; set ; }

        [ Required ]

        [ Display (Name = "Last Name: " )]

        public string LastName { get ; set ; }

        public string UserType { get ; set ; }

        public System. DateTime CreatedDate { get ; set ; }

        public bool IsActive { get ; set ; }

        public string IPAddress { get ; set ; }

   }

    
This is my context class again this is also generated by data model.

public partial class UserEntities2 : DbContext

{

    public UserEntities2()

        : base ( "name=UserEntities2" )

    {

    }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)

    {

        throw new UnintentionalCodeFirstException();

    }

    public DbSet< Registration > Registrations { get ; set ; }

}     
Now time to add a controller.

Image 3.   
    
Add namespaces on controller class

using System.Web.Security;

using LoginInMVC4WithEF.Models;

Now add following functions and methods.

//

        // GET: /User/

  public ActionResult Index()

        {

            return View();

        }

[HttpGet]

public ActionResult LogIn()

        {

            return View();

        }

[HttpPost]

public ActionResult LogIn(Models.Registration userr)

        {

            //if (ModelState.IsValid)

            //{

                if (IsValid(userr.Email, userr.Password))

                {

                    FormsAuthentication.SetAuthCookie(userr.Email, false );

                    return RedirectToAction( "Index" , "Home" );

                }

                else

                {

                    ModelState.AddModelError( "" , "Login details are wrong." );

                }

            //}

            //else

            //{

            //    ModelState.AddModelError("", "Error, Check data");

            //}

            return View(userr);

        }

[HttpGet]

public ActionResult Register()

        {

            return View();

        }

[HttpPost]

  public ActionResult Register(Models.Registration user)

        {

            try

            {

                if (ModelState.IsValid)

                {

                    using (var db = new LoginInMVC4WithEF.Models.UserEntities2())

                    {

                        var crypto = new SimpleCrypto.PBKDF2();

                        var encrypPass = crypto.Compute(user.Password);

                        var newUser = db.Registrations.Create();

                        newUser.Email = user.Email;

                        newUser.Password = encrypPass;

                        newUser.PasswordSalt = crypto.Salt;

                        newUser.FirstName = user.FirstName;

                        newUser.LastName = user.LastName;

                        newUser.UserType = "User" ;

                        newUser.CreatedDate = DateTime.Now;

                        newUser.IsActive = true ;

                        newUser.IPAddress = "642 White Hague Avenue" ;

                        db.Registrations.Add(newUser);

                        db.SaveChanges();

                        return RedirectToAction( "Index" , "Home" );

                    }

                }

                else

                {

                    ModelState.AddModelError( "" , "Data is not correct" );

                }

            }

            catch (DbEntityValidationException e)

            {

                foreach (var eve in e.EntityValidationErrors)

                {

                    Console.WriteLine( "Entity of type \"{0}\" in state \"{1}\" has the following validation errors:" ,

                        eve.Entry.Entity.GetType().Name, eve.Entry.State);

                    foreach (var ve in eve.ValidationErrors)

                    {

                        Console.WriteLine( "- Property: \"{0}\", Error: \"{1}\"" ,

                            ve.PropertyName, ve.ErrorMessage);

                    }

                }

                throw ;

            }

            return View();

        }

public ActionResult LogOut()

        {

            FormsAuthentication.SignOut();

            return RedirectToAction( "Index" , "Home" );

        }

  private bool IsValid( string email, string password)

        {

            var crypto = new SimpleCrypto.PBKDF2();

            bool IsValid = false ;

            using (var db = new LoginInMVC4WithEF.Models.UserEntities2())

            {

                var user = db.Registrations.FirstOrDefault(u => u.Email == email);

                if (user != null )

                {

                    if (user.Password == crypto.Compute(password, user.PasswordSalt))

                    {

                        IsValid = true ;

                    }

                }

            }

            return IsValid;

        }

  There I have functions and methods for index page and login, logout, register and isvalid, now let's make some change in \_Layout.vshtml.add fowling div.

< div style =" width : auto ; background-color : aqua ">

        @ if (Request.IsAuthenticated)

        {

            < strong > @ Html.Encode(User.Identity.Name) </ strong >

            @ Html.ActionLink( "Log Out" , "Logout" , "User" )

        }

        else

        {

            @ Html.ActionLink( "Register " , "Register" , "User" )

            < span > | </ span >

            @ Html.ActionLink( "Log In" , "Login" , "User" )

        }

    </ div >     
Now let's add views for presentation.   
    
Best way to add view right clicks on controller's method name and right click and Add View and select view engine type and select strongly-typed view and use layout master page and click add.

Image 4.   
    
LogIn.cshtml   
 

@model LoginInMVC4WithEF.Models. Registration

@{

    ViewBag.Title = "LogIn" ;

    Layout = "~/Views/Shared/\_Layout.cshtml" ;

}

< h2 > LogIn </ h2 >

@ using (Html.BeginForm())

{

    @ Html.ValidationSummary( true , "Login Failed, check details" );

    < div >

        < fieldset >

            < legend > Login Form </ legend >

            < div class ="editor-label"> @ Html.LabelFor(u=> u.Email) </ div >

            < div class ="editor-field"> @ Html.TextBoxFor(u=> u.Email)

                @ Html.ValidationMessageFor(u=> u.Email)

            </ div >

            < div class ="editor-label"> @ Html.LabelFor(u=> u.Password) </ div >

            < div   class ="editor-field"> @ Html.PasswordFor(u=> u.Password)

                @ Html.ValidationMessageFor(u=> u.Password)

            </ div >

            < input type ="submit" value ="Log In" />

        </ fieldset >

    </ div >

}

Register.cshtml   
 

@model LoginInMVC4WithEF.Models. Registration

@{

    ViewBag.Title = "Register" ;

    Layout = "~/Views/Shared/\_Layout.cshtml" ;

}

< h2 > Register </ h2 >

@ using (Html.BeginForm())

{

    @ Html.ValidationSummary( true , "Registeration Failed, fields." );

    < div >

        < fieldset >

            < legend > Register Form </ legend >

            < div class ="editor-label"> @ Html.LabelFor(u => u.Email) </ div >

            < div class ="editor-field"> @ Html.TextBoxFor(u => u.Email)

                @ Html.ValidationMessageFor(u => u.Email)

            </ div >

            < div class ="editor-label"> @ Html.LabelFor(u => u.Password) </ div >

            < div class ="editor-field"> @ Html.PasswordFor(u => u.Password)

                @ Html.ValidationMessageFor(u => u.Password)

            </ div >

            < div class ="editor-label"> @ Html.LabelFor(u => u.FirstName) </ div >

            < div class ="editor-field"> @ Html.TextBoxFor(u => u.FirstName)

                @ Html.ValidationMessageFor(u => u.FirstName)

            </ div >

             < div class ="editor-label"> @ Html.LabelFor(u => u.LastName) </ div >

            < div class ="editor-field"> @ Html.TextBoxFor(u => u.LastName)

                @ Html.ValidationMessageFor(u => u.LastName)

            </ div >

            < input type ="submit" value ="Register" />

        </ fieldset >

    </ div >

}     
LogOut.cshtml   
 

@{

    ViewBag.Title = "LogOut" ;

    Layout = "~/Views/Shared/\_Layout.cshtml" ;

}

< h2 > LogOut </ h2 >

public ActionResult LogOut()

{

    FormsAuthentication.SignOut();

     return RedirectToAction( "Index" , "Home" );

}

    
Now hit f5 to run the application and click on register button and don't put anything in T ext B oxes and click register button.

Image 5.   
    
As you can see in model class all fields are required so these message are displaying.

Image 6.   
    
Now if I put password value less than 6 characters or more than 150 characters then this message will display.

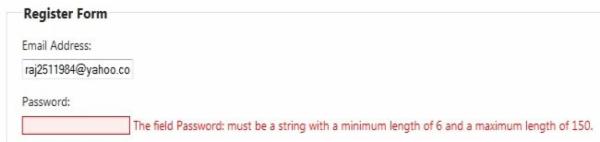


Image 7.   
    
Now insert all value in proper way then you can see it will register and data should insert in database, now you can login.

Image 8.

Image 9.   
    
**Conclusion**

In this article we have learn the basic concept of login using entity framework with razor.