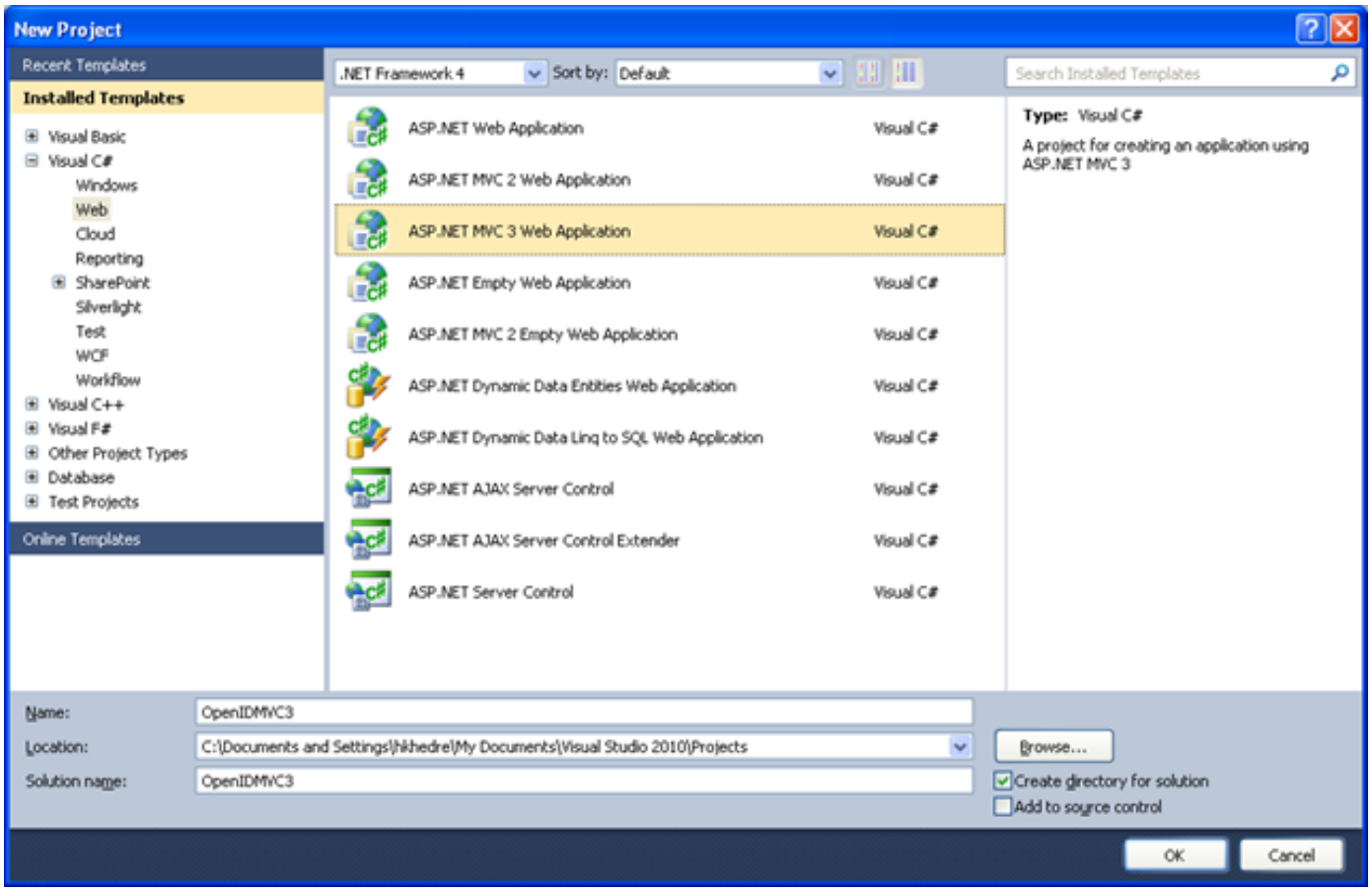
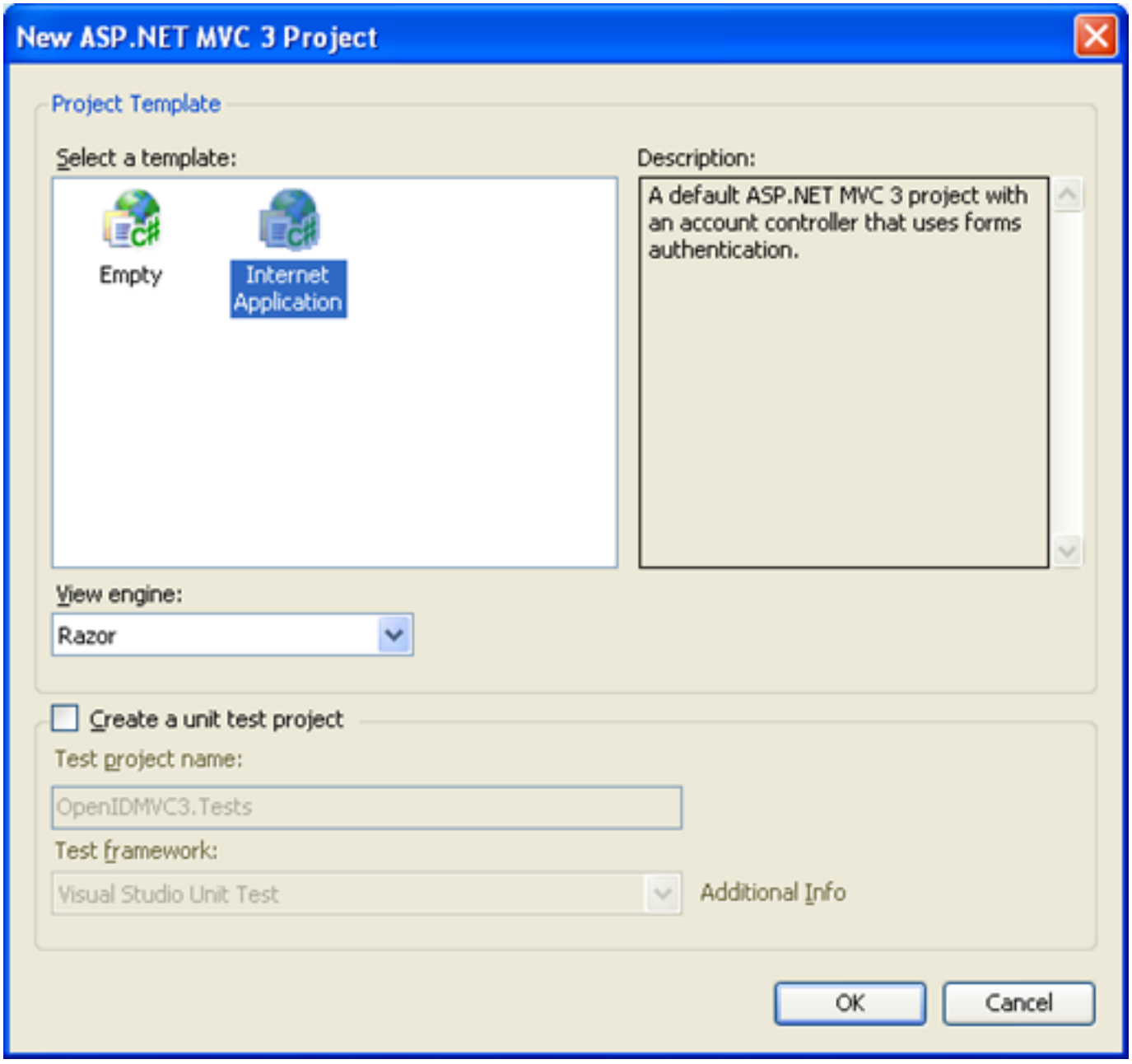
عنوان: **استفاده از OpenID در وب سایت جهت احراز هویت کاربران**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۰۴/۰۸ ۲۱:۲
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, C#, MVC, Security, OpenID

قبلا شرح مختصری در زمینه OpenID در اینجا گفته شد.
حال می‌خواهیم این امکان را در پروژه خود بکار ببریم، جهت این کار باید ابتدا یک پروژه ایجاد کرده و از کتابخانه‌های سورس باز موجود استفاده کرد.
1- ابتدا در ویژوال استودیو یا هر نرم افزار دیگر یک پروژه MVC ایجاد نمایید.



2- نوع Internet Application و برای View Engine سایت Razor را انتخاب نمایید.



3- کتابخانه DotNetOpenId سورس باز را می‌توانید مستقیما از این آدرس دانلود نموده یا از طریق Package Manager Console و با نوشتن Install-Package DotNetOpenAuth به صورت آنلاین این کتابخانه را نصب نمایید.

4- مدل‌های برنامه را مانند زیر ایجاد نمایید

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Globalization;
using System.Security.Cryptography;
using System.Text;
using System.Web.Mvc;
using System.Web.Security;

namespace OpenIDExample.Models
{
    #region Models

    public class ChangePasswordModel
    {
        [Required]
```

```
        [DataType(DataType.Password)]
        [Display(Name = "Current password")]
        public string OldPassword { get; set; }

        [Required]
        [ValidatePasswordLength]
        [DataType(DataType.Password)]
        [Display(Name = "New password")]
        public string NewPassword { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm new password")]
        [Compare("NewPassword", ErrorMessage = "The new password and confirmation password do not
match.")]
        public string ConfirmPassword { get; set; }
    }

    public class LogOnModel
    {
        [Display(Name = "OpenID")]
        public string OpenID { get; set; }

        [Required]
        [Display(Name = "User name")]
        public string UserName { get; set; }

        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [Display(Name = "Remember me?")]
        public bool RememberMe { get; set; }
    }

    public class RegisterModel
    {
        [Display(Name = "OpenID")]
        public string OpenID { get; set; }

        [Required]
        [Display(Name = "User name")]
        public string UserName { get; set; }

        [Required]
        [DataType(DataType.EmailAddress)]
        [Display(Name = "Email address")]
        public string Email { get; set; }

        [Required]
        [ValidatePasswordLength]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm password")]
        [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
        public string ConfirmPassword { get; set; }
    }

    #endregion Models

    #region Services

    // The FormsAuthentication type is sealed and contains static members, so it is difficult to
    // unit test code that calls its members. The interface and helper class below demonstrate
    // how to create an abstract wrapper around such a type in order to make the AccountController
    // code unit testable.

    public interface IMembershipService
    {
        int MinPasswordLength { get; }

        bool ValidateUser(string userName, string password);

        MembershipCreateStatus CreateUser(string userName, string password, string email, string
OpenID);

        bool ChangePassword(string userName, string oldPassword, string newPassword);
```

```
        MembershipUser GetUser(string OpenID);
    }

    public class AccountMembershipService : IMembershipService
    {
        private readonly MembershipProvider _provider;

        public AccountMembershipService()
            : this(null)
        {
        }

        public AccountMembershipService(MembershipProvider provider)
        {
            _provider = provider ?? Membership.Provider;
        }

        public int MinPasswordLength
        {
            get
            {
                return _provider.MinRequiredPasswordLength;
            }
        }

        public bool ValidateUser(string userName, string password)
        {
            if (String.IsNullOrEmpty(userName)) throw new ArgumentException("Value cannot be null or
empty.", "userName");
            if (String.IsNullOrEmpty(password)) throw new ArgumentException("Value cannot be null or
empty.", "password");

            return _provider.ValidateUser(userName, password);
        }

        public Guid StringToGUID(string value)
        {
            // Create a new instance of the MD5CryptoServiceProvider object.
            MD5 md5Hasher = MD5.Create();
            // Convert the input string to a byte array and compute the hash.
            byte[] data = md5Hasher.ComputeHash(Encoding.Default.GetBytes(value));
            return new Guid(data);
        }

        public MembershipCreateStatus CreateUser(string userName, string password, string email, string
OpenID)
        {
            if (String.IsNullOrEmpty(userName)) throw new ArgumentException("Value cannot be null or
empty.", "userName");
            if (String.IsNullOrEmpty(password)) throw new ArgumentException("Value cannot be null or
empty.", "password");
            if (String.IsNullOrEmpty(email)) throw new ArgumentException("Value cannot be null or
empty.", "email");

            MembershipCreateStatus status;
            _provider.CreateUser(userName, password, email, null, null, true, StringToGUID(OpenID), out
status);
            return status;
        }

        public MembershipUser GetUser(string OpenID)
        {
            return _provider.GetUser(StringToGUID(OpenID), true);
        }

        public bool ChangePassword(string userName, string oldPassword, string newPassword)
        {
            if (String.IsNullOrEmpty(userName)) throw new ArgumentException("Value cannot be null or
empty.", "userName");
            if (String.IsNullOrEmpty(oldPassword)) throw new ArgumentException("Value cannot be null or
empty.", "oldPassword");
            if (String.IsNullOrEmpty(newPassword)) throw new ArgumentException("Value cannot be null or
empty.", "newPassword");

            // The underlying ChangePassword() will throw an exception rather
            // than return false in certain failure scenarios.
            try
            {
                MembershipUser currentUser = _provider.GetUser(userName, true /* userIsOnline */);
                return currentUser.ChangePassword(oldPassword, newPassword);
            }
```

```
            catch (ArgumentException)
            {
                return false;
            }
            catch (MembershipPasswordException)
            {
                return false;
            }
        }

        public MembershipCreateStatus CreateUser(string userName, string password, string email)
        {
            throw new NotImplementedException();
        }
    }

    public interface IFormsAuthenticationService
    {
        void SignIn(string userName, bool createPersistentCookie);

        void SignOut();
    }

    public class FormsAuthenticationService : IFormsAuthenticationService
    {
        public void SignIn(string userName, bool createPersistentCookie)
        {
            if (String.IsNullOrEmpty(userName)) throw new ArgumentException("Value cannot be null or
empty.", "userName");

            FormsAuthentication.SetAuthCookie(userName, createPersistentCookie);
        }

        public void SignOut()
        {
            FormsAuthentication.SignOut();
        }
    }

    #endregion Services

    #region Validation

    public static class AccountValidation
    {
        public static string ErrorCodeToString(MembershipCreateStatus createStatus)
        {
            // See http://go.microsoft.com/fwlink/?LinkID=177550 for
            // a full list of status codes.
            switch (createStatus)
            {
                case MembershipCreateStatus.DuplicateUserName:
                    return "Username already exists. Please enter a different user name.";

                case MembershipCreateStatus.DuplicateEmail:
                    return "A username for that e-mail address already exists. Please enter a different
e-mail address.";

                case MembershipCreateStatus.InvalidPassword:
                    return "The password provided is invalid. Please enter a valid password value.";

                case MembershipCreateStatus.InvalidEmail:
                    return "The e-mail address provided is invalid. Please check the value and try
again.";

                case MembershipCreateStatus.InvalidAnswer:
                    return "The password retrieval answer provided is invalid. Please check the value
and try again.";

                case MembershipCreateStatus.InvalidQuestion:
                    return "The password retrieval question provided is invalid. Please check the value
and try again.";

                case MembershipCreateStatus.InvalidUserName:
                    return "The user name provided is invalid. Please check the value and try again.";

                case MembershipCreateStatus.ProviderError:
                    return "The authentication provider returned an error. Please verify your entry and
try again. If the problem persists, please contact your system administrator.";

                case MembershipCreateStatus.UserRejected:
```

```
                    return "The user creation request has been canceled. Please verify your entry and
try again. If the problem persists, please contact your system administrator.";

                default:
                    return "An unknown error occurred. Please verify your entry and try again. If the
problem persists, please contact your system administrator.";
            }
        }
    }

    [AttributeUsage(AttributeTargets.Field | AttributeTargets.Property, AllowMultiple = false,
Inherited = true)]
    public sealed class ValidatePasswordLengthAttribute : ValidationAttribute, IClientValidatable
    {
        private const string _defaultErrorMessage = "'{0}' must be at least {1} characters long.";
        private readonly int _minCharacters = Membership.Provider.MinRequiredPasswordLength;

        public ValidatePasswordLengthAttribute()
            : base(_defaultErrorMessage)
        {
        }

        public override string FormatErrorMessage(string name)
        {
            return String.Format(CultureInfo.CurrentCulture, ErrorMessageString,
                name, _minCharacters);
        }

        public override bool IsValid(object value)
        {
            string valueAsString = value as string;
            return (valueAsString != null && valueAsString.Length >= _minCharacters);
        }

        public IEnumerable<ModelClientValidationRule> GetClientValidationRules(ModelMetadata metadata,
ControllerContext context)
        {
            return new[]{
                new
ModelClientValidationStringLengthRule(FormatErrorMessage(metadata.GetDisplayName()), _minCharacters,
int.MaxValue)
            };
        }
    }

    #endregion Validation
}
```

-5 در پروژه مربوطه یک Controller به نام AccountController ایجاد نمایید. و کدهای زیر را برای آنها وارد نمایید.

```
using System.Web.Mvc;
using System.Web.Routing;
using System.Web.Security;
using DotNetOpenAuth.Messaging;
using DotNetOpenAuth.OpenId;
using DotNetOpenAuth.OpenId.RelyingParty;
using OpenIDExample.Models;

namespace OpenIDExample.Controllers
{
    public class AccountController : Controller
    {
        private static OpenIdRelyingParty openid = new OpenIdRelyingParty();

        public IFormsAuthenticationService FormsService { get; set; }

        public IMembershipService MembershipService { get; set; }

        protected override void Initialize(RequestContext requestContext)
        {
            if (FormsService == null) { FormsService = new FormsAuthenticationService(); }
            if (MembershipService == null) { MembershipService = new AccountMembershipService(); }

            base.Initialize(requestContext);
        }

        // **************************************
        // URL: /Account/LogOn
```

```
        // **************************************

        public ActionResult LogOn()
        {
            return View();
        }

        [HttpPost]
        public ActionResult LogOn(LogOnModel model, string returnUrl)
        {
            if (ModelState.IsValid)
            {
                if (MembershipService.ValidateUser(model.UserName, model.Password))
                {
                    FormsService.SignIn(model.UserName, model.RememberMe);
                    if (Url.IsLocalUrl(returnUrl))
                    {
                        return Redirect(returnUrl);
                    }
                    else
                    {
                        return RedirectToAction("Index", "Home");
                    }
                }
                else
                {
                    ModelState.AddModelError("", "The user name or password provided is incorrect.");
                }
            }

            // If we got this far, something failed, redisplay form
            return View(model);
        }

        // **************************************
        // URL: /Account/LogOff
        // **************************************

        public ActionResult LogOff()
        {
            FormsService.SignOut();

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

        // **************************************
        // URL: /Account/Register
        // **************************************

        public ActionResult Register(string OpenID)
        {
            ViewBag.PasswordLength = MembershipService.MinPasswordLength;
            ViewBag.OpenID = OpenID;
            return View();
        }

        [HttpPost]
        public ActionResult Register(RegisterModel model)
        {
            if (ModelState.IsValid)
            {
                // Attempt to register the user
                MembershipCreateStatus createStatus = MembershipService.CreateUser(model.UserName,
model.Password, model.Email, model.OpenID);

                if (createStatus == MembershipCreateStatus.Success)
                {
                    FormsService.SignIn(model.UserName, false /* createPersistentCookie */);
                    return RedirectToAction("Index", "Home");
                }
                else
                {
                    ModelState.AddModelError("", AccountValidation.ErrorCodeToString(createStatus));
                }
            }

            // If we got this far, something failed, redisplay form
            ViewBag.PasswordLength = MembershipService.MinPasswordLength;
            return View(model);
        }
```

```
        // ************************************
        // URL: /Account/ChangePassword
        // ************************************

        [Authorize]
        public ActionResult ChangePassword()
        {
            ViewBag.PasswordLength = MembershipService.MinPasswordLength;
            return View();
        }

        [Authorize]
        [HttpPost]
        public ActionResult ChangePassword(ChangePasswordModel model)
        {
            if (ModelState.IsValid)
            {
                if (MembershipService.ChangePassword(User.Identity.Name, model.OldPassword,
model.NewPassword))
                {
                    return RedirectToAction("ChangePasswordSuccess");
                }
                else
                {
                    ModelState.AddModelError("", "The current password is incorrect or the new password
is invalid.");
                }
            }

            // If we got this far, something failed, redisplay form
            ViewBag.PasswordLength = MembershipService.MinPasswordLength;
            return View(model);
        }

        // ************************************
        // URL: /Account/ChangePasswordSuccess
        // ************************************

        public ActionResult ChangePasswordSuccess()
        {
            return View();
        }

        [ValidateInput(false)]
        public ActionResult Authenticate(string returnUrl)
        {
            var response = openid.GetResponse();
            if (response == null)
            {
                //Let us submit the request to OpenID provider
                Identifier id;
                if (Identifier.TryParse(Request.Form["openid_identifier"], out id))
                {
                    try
                    {
                        var request = openid.CreateRequest(Request.Form["openid_identifier"]);
                        return request.RedirectingResponse.AsActionResult();
                    }
                    catch (ProtocolException ex)
                    {
                        ViewBag.Message = ex.Message;
                        return View("LogOn");
                    }
                }

                ViewBag.Message = "Invalid identifier";
                return View("LogOn");
            }

            //Let us check the response
            switch (response.Status)
            {
                case AuthenticationStatus.Authenticated:
                    LogOnModel lm = new LogOnModel();
                    lm.OpenID = response.ClaimedIdentifier;
                    //check if user exist
                    MembershipUser user = MembershipService.GetUser(lm.OpenID);
                    if (user != null)
                    {
                        lm.UserName = user.UserName;
                        FormsService.SignIn(user.UserName, false);
```

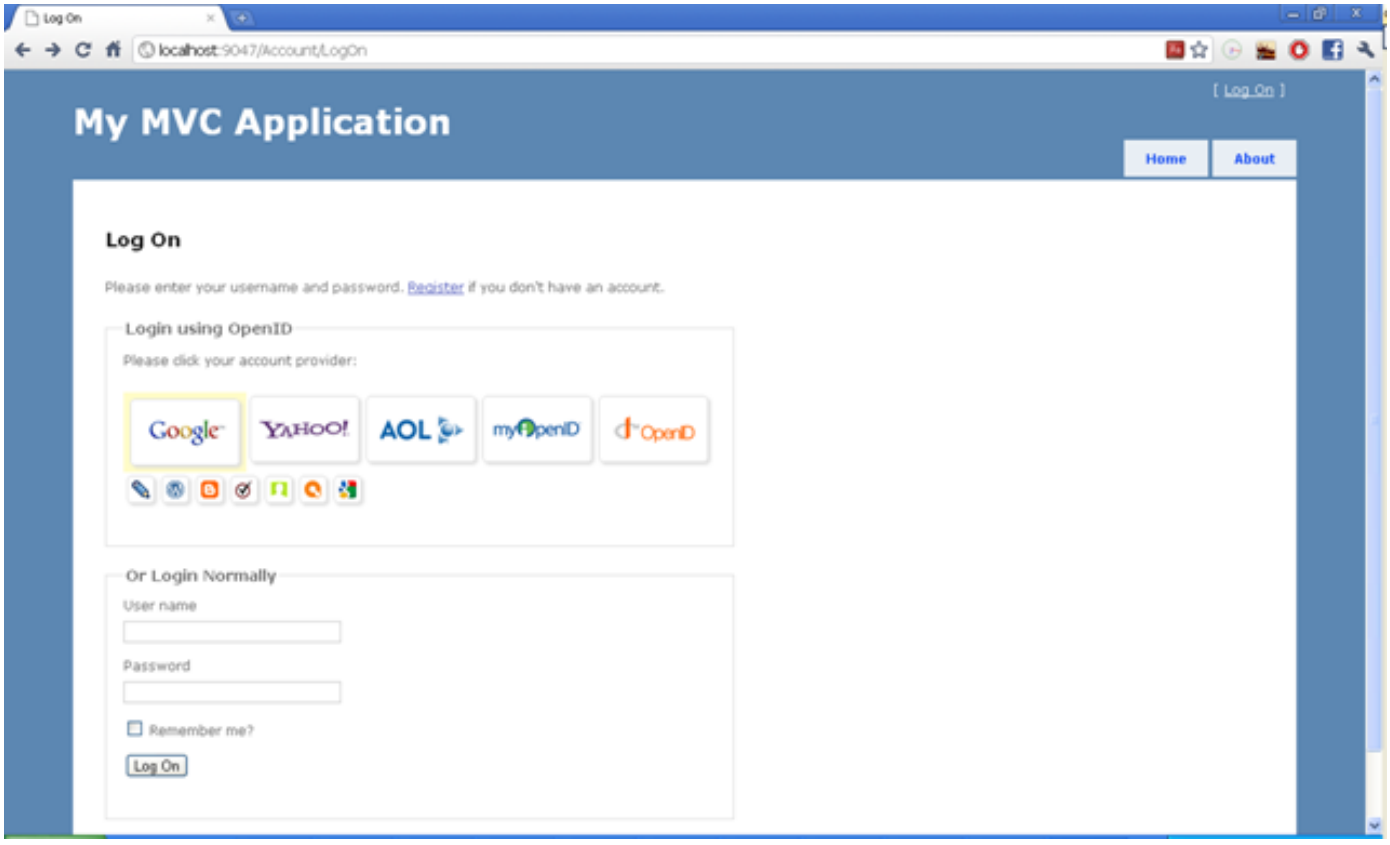
```
                    }

                    return View("LogOn", lm);

                case AuthenticationStatus.Canceled:
                    ViewBag.Message = "Canceled at provider";
                    return View("LogOn");
                case AuthenticationStatus.Failed:
                    ViewBag.Message = response.Exception.Message;
                    return View("LogOn");
            }

            return new EmptyResult();
        }
    }
}
```

6- سپس برای Action به نام LogOn یک View می‌سازیم، برای Authenticate نیازی به ایجاد View ندارد چون قرار است درخواست کاربر را به آدرس دیگری Redirect کند. سپس کدهای زیر را برای View ایجاد شده وارد می‌کنیم.

```
@model OpenIDExample.Models.LogOnModel
@{
    ViewBag.Title = "Log On";
}
<h2>
    Log On</h2>
<p>
    Please enter your username and password. @Html.ActionLink("Register", "Register")
    if you don't have an account.
</p>
<script src="@Url.Content("~/Scripts/jquery.validate.min.js")" type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")"
type="text/javascript"></script>
<form action="Authenticate?ReturnUrl=@HttpUtility.UrlEncode(Request.QueryString["ReturnUrl"])"
method="post" id="openid_form">
<input type="hidden" name="action" value="verify" />
<div>
    <fieldset>
        <legend>Login using OpenID</legend>
        <div class="openid_choice">
            <p>
                Please click your account provider:</p>
            <div id="openid_btns">
            </div>
        </div>
        <div id="openid_input_area">
            @Html.TextBox("openid_identifier")
            <input type="submit" value="Log On" />
        </div>
        <noscript>
            <p>
                OpenID is service that allows you to log-on to many different websites using a single
                indentity. Find out <a href="http://openid.net/what/">more about OpenID</a> and
                <a href="http://openid.net/get/">how to get an OpenID enabled account</a>.</p>
        </noscript>
        <div>
            @if (Model != null)
            {
                if (String.IsNullOrEmpty(Model.UserName))
                {
                <div class="editor-label">
                    @Html.LabelFor(model => model.OpenID)
                </div>
                <div class="editor-field">
                    @Html.DisplayFor(model => model.OpenID)
                </div>
                <p class="button">
                    @Html.ActionLink("New User ,Register", "Register", new { OpenID = Model.OpenID })
                </p>
                }
                else
                {
                    //user exist
                <p class="buttonGreen">
                    <a href="@Url.Action("Index", "Home")">Welcome , @Model.UserName, Continue..." </a>
                </p>
```

```
                }
            }
        </div>
    </fieldset>
</div>
</form>
@Html.ValidationSummary(true, "Login was unsuccessful. Please correct the errors and try again.")
@using (Html.BeginForm())
{
    <div>
        <fieldset>
            <legend>Or Login Normally</legend>
            <div class="editor-label">
                @Html.LabelFor(m => m.UserName)
            </div>
            <div class="editor-field">
                @Html.TextBoxFor(m => m.UserName)
                @Html.ValidationMessageFor(m => m.UserName)
            </div>
            <div class="editor-label">
                @Html.LabelFor(m => m.Password)
            </div>
            <div class="editor-field">
                @Html.PasswordFor(m => m.Password)
                @Html.ValidationMessageFor(m => m.Password)
            </div>
            <div class="editor-label">
                @Html.CheckBoxFor(m => m.RememberMe)
                @Html.LabelFor(m => m.RememberMe)
            </div>
            <p>
                <input type="submit" value="Log On" />
            </p>
        </fieldset>
    </div>
}
```

پس از اجرای پروژه صفحه ای شبیه به پایین مشاهده کرده و سرویس دهنده OpenID خاص خود را می‌توانید انتخاب نمایید.



7- برای فعال سازی عملیات احراز هویت توسط FormsAuthentication در سایت باید تنظیمات زیر را در فایل web.config انجام دهید.

```
<authentication mode="Forms">
      <forms loginUrl="~/Account/LogOn" timeout="2880" />
</authentication>
```

خوب تا اینجا کار تمام است و کاربر در صورتی که در سایت OpenID نام کاربری داشته باشد می‌تواند در سایت شما Login کند.
جهت مطالعات بیشتر ودانلود نمونه کدهای آماده می‌توانید به لینک‌های (^ و ^ و ^ و ^ و ^ و ^ و ^) مراجعه کنید.
کد کامل پروژه را می‌توانید از اینجا دانلود نمایید.

منبع

نظرات خوانندگان

نویسنده: ahmadalli
تاریخ: ۱۳۹۱/۰۴/۰۸ ۲۳:۱۲

خوب این کدهای شما برای نسخه‌های قدیمی MVC هست. من نصفش رو درست متوجه نشدم. اگر میشه برای نسخه ۴ یا ۳ هم مثال بزارید.

نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۱/۰۴/۰۹ ۱:۲۵

اگه میشه کد برنامه را ضمیمه کنید...

نویسنده: صابر فتح اللهی
تاریخ: ۱۳۹۱/۰۴/۰۹ ۲:۴

کدی که ابتدا گذاشته بودم ظاهرا خیلی قدیمی بود اون با کدهای جدید تعویض کردم و لینک دانلود کد هم برای دوستان قرار دادم

نویسنده: صابر فتح اللهی
تاریخ: ۱۳۹۱/۰۴/۰۹ ۲:۲۱

اصلاح شد

نویسنده: saleh
تاریخ: ۱۳۹۱/۰۴/۱۷ ۰:۳۳

فرضاً اگه شخصی که با OID لوگین کرده بخواد در فروم سایت من فعالیت کنه چه جوری به اطلاعات پستها و اعمالش پی ببرم؟ چون این شخص در سایت ثبت نام نکرده و طبیعتاً اکانتی در سایت نداره!

در قسمت اول مقاله به این موضوعات اشاره نکردید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۷ ۱۱:۴۴

بحث فوروم نوشته شده شما با طراحی یک سیستم جدید (که در اینجا مد نظر است) متفاوت است. سورس فوق را مطالعه کنید تا با نحوه یکپارچگی آن با سیستم membership دات نت آشنا شوید.

نویسنده: احمدعلی شفیعی
تاریخ: ۱۳۹۱/۰۴/۱۸ ۱۲:۳۸

ممنونم

نویسنده: کامران
تاریخ: ۱۳۹۲/۰۲/۱۲ ۲۳:۲۱

سلام دوست عزیز.
قربان برای ASP.NET Webforms هم میتونید مقاله ای رو آماده کنید؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۱۳ ۰:۴

یک پروژه دیگر هم در سایت در مورد پروتکل OAuth هست.

نویسنده: کامران
تاریخ: ۱۳۹۲/۰۲/۱۳ ۰:۹

ممنون از جوابتون، دوست عزیز این هم سمپل و کدهاش با MVC هست، مقالاتی برای Webforms پیدا کردم به زبان لاتین اما درست توضیح داده نشده متاسفانه

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۲/۱۳ ۰:۲۵

سلام
دوست گلم یه مثال هست گذاشتم توی این آدرس

Google OpenID Authentication در ASP.NET با استفاده از DotNetOpenAuth