عنوان: **OpenID چیست؟**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۰۴/۰۲ ۱۹:۲۳
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, C#, others, OpenID

چند وقتی میشه که دنبال روش‌های OpenID هستم که ببینم چطوری کار می‌کنند، خودم هم تازه شروع کردم خوب قبل از هر چیزی اول ببینیم مفهوم OpenID چی هست؟ و کم کم جلو میریم و مثال هایی معرفی می‌کنیم.

OpenID به شما اجازه می‌دهد با استفاده از اکانت (نام کاربری) که در یک سایت دارید بتوانید به سایت‌های متفاوتی وارد شوید (لاگین کنید) بدون این که نیاز به ثبت نام دوباره در آن سایت‌ها داشته باشید.

نمونه بارز آن می‌توان به سایت هایی مانند StackOverflow اشاره کرد.

OpenID به شما اجازه می‌دهد شما در یک نام کاربری که برای خود ایجاد کرده اید اطلاعاتی را که دارید با دیگر وبسایت‌ها به اشتراک بگذارید.
با OpenID کلمه عبور شما فقط توسط سرویس دهنده گرفته میشود و سرویس دهنده هویت شما را برای بازید از سایت دیگر تایید می‌کند.از طرف دیگر سرویس دهنده شما تا شما اجازه ندهید هیچ وب سایتی کلمه عبور شما را به هیچ وب سایتی نمی‌بیند. بنابراین نیازی نیست در مورد کلمه عبور خود نگرانی به خود راه دهید.

OpenID به سرعت در حال گسترش بروی وب استف در حال حاضر بیش از یک میلیارد نام کاربری (اکانت) وجود دارد و بیش از 50000 سایت OpenID را پذیرفته و با آن کار می‌کنند. چندین سازمان بزرگ موضوع OpenID را پذیرفته اند، سازمان هایی مانند Google, FaceBook,Yahoo!,Microsoft,AOL,MySpace,Sears, Universal Music Group, France Telecom, Novell, Sun, Telecom Italia
و بسیاری از سازمان‌های دیگر.

در کل می‌توان گفت ما در یک سایت خاص مانند یاهو ، گوگل، و... یک نام کاربری خواهیم داشت سپس برای ارتباط سایت مقصد (مثلا همین سایت) با نام کاربری ما در گوگل به این صورت عمل می‌شود که ابتدا از طریق این سایت ما به سایت گوگل هدایت می‌شویم در آنجا از ما یک تاییدیه جهت استفاده از سرویس OpenID از کاربر میگیرد. در صورت تایید کاربر سایت گوگل از این لحظه جهت احراز هویت کاربر برای ورود به سایت مقصد استفاده می‌کند .
زمانی که کاربر می‌خواهد به این سایت لاگین کند سایت نام کاربری و کلمه عبور او را (در صورتی که قبلا به گوگل لاگین کرده باشد نیازی نیست و وارد سایت می‌شود) به گوگل می‌فرستد و پس از تایید هویت در صورت صحیح بودن اجازه می‌دهد کاربر به آسانی وارد سایت مقصد شود.
تصویر زیر نمایانگر این روش می‌باشد.

در پست‌های آینده مثال‌ها و کامپوننت‌های سورس بازی جهت این کار به شما معرفی خواهیم کرد. جهت مطالعات بیشتر می‌توانید به این لینک‌ها مراجعه کنین (^ و ^ و ^).

نظرات خوانندگان

نویسنده: ashi mashi
تاریخ: ۱۳۹۱/۰۴/۱۸ ۱۱:۳۵

سلام،
این چیزی که شما میگید ، آیا مشابه single sign on هستش ؟
همون کار رو انجام میده ؟
آیا با Active directory قابل اتصال هست یا نه ؟

با تشکر....

نویسنده: صابر فتح اللهی
تاریخ: ۱۳۹۱/۰۴/۱۸ ۱۱:۳۸

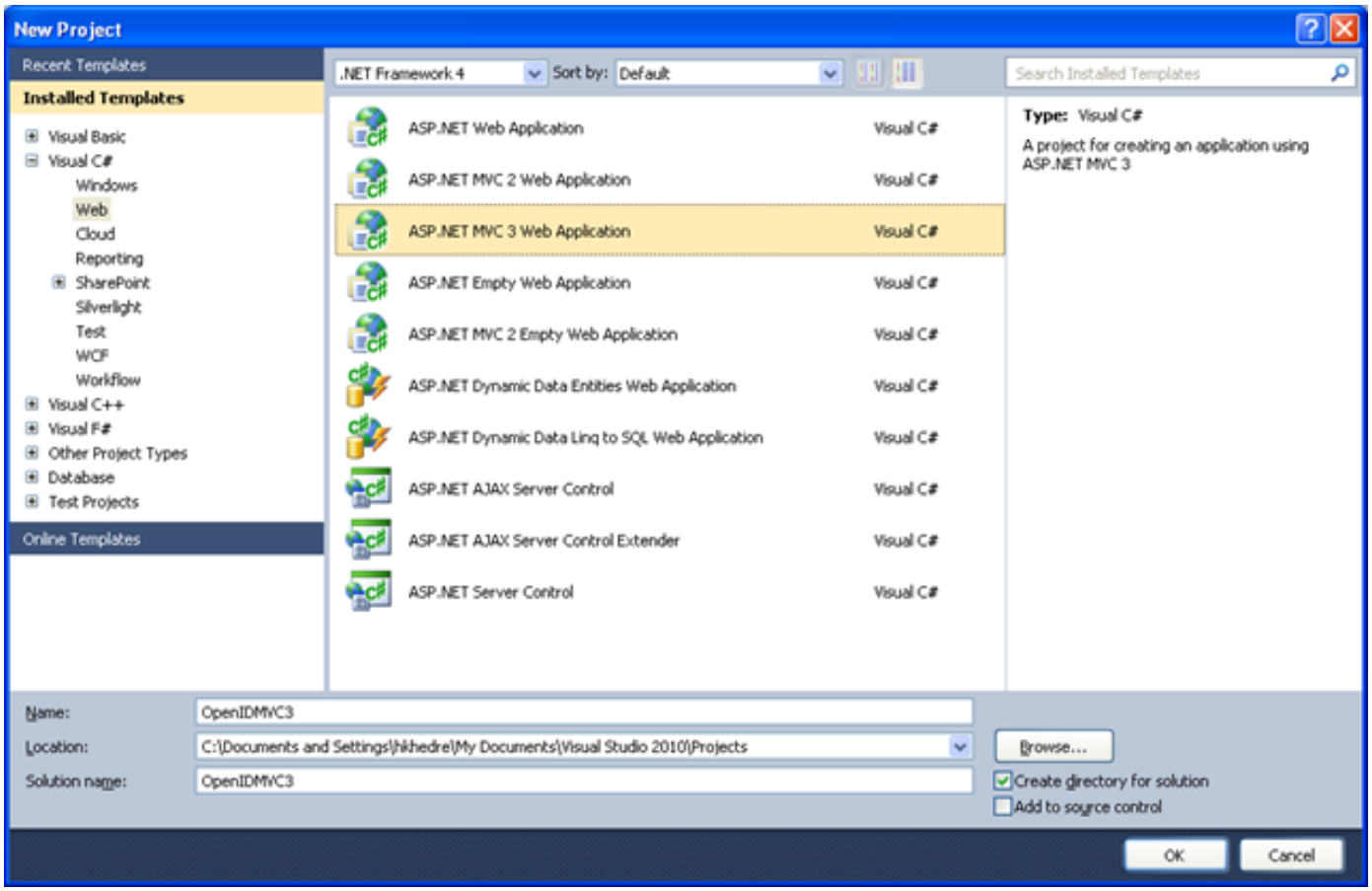
فکر نمی‌کنم بشه
چون Active Directory با Domain تنظیم میشه
اما این حالت احراز هویت روی فرم تنظیم میشه
برای SSO مطلب توی همین سایت هست در واقع توی اون روش ما با یک اکانت که توی یک سایت داریم از سایت‌های دیگه روی همون سرویس استفاده میکنیم و میشه گفت حالت اختصاصی داره
اما توی این حالت شما با هر سایتی که این قابلیت داشته باشه میتونی کار کنی

نویسنده: Hamid NCH
تاریخ: ۱۳۹۲/۰۹/۱۴ ۱۲:۳۱

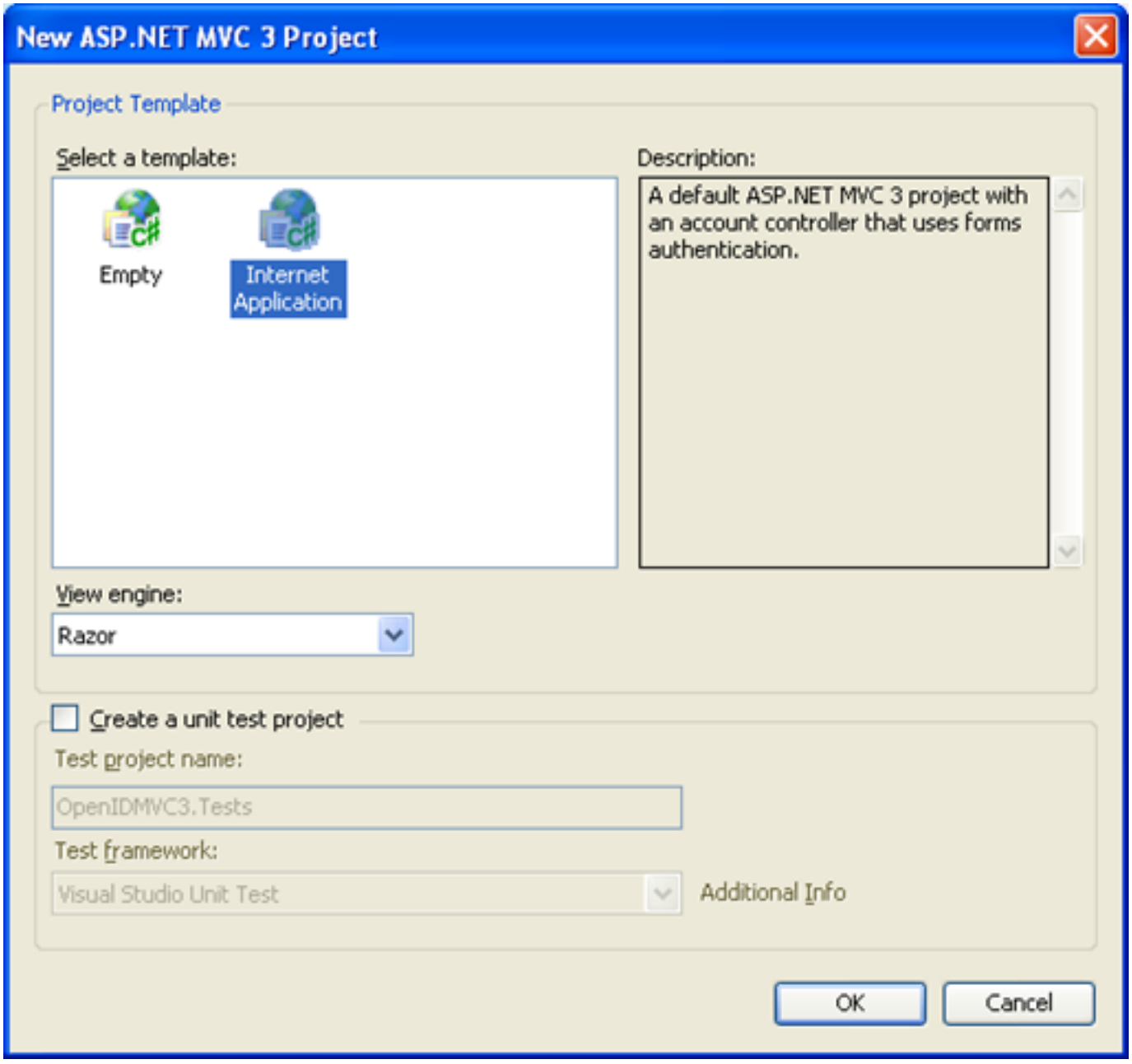
نقطه شروع یادگیری من در برنامه نویسی وب، پروژه رایگان شما جناب آقای استاد صابر فتح الهی بود.
بعد از اون هم که با سایت آقای دلشاد آشنا شدم و اینک سایت فوق العاده پربار جاری به همت تمام عزیزان.
و باز هم الان دارم از شما آقای فتح الهی چیز یاد میگیرم.

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);
```

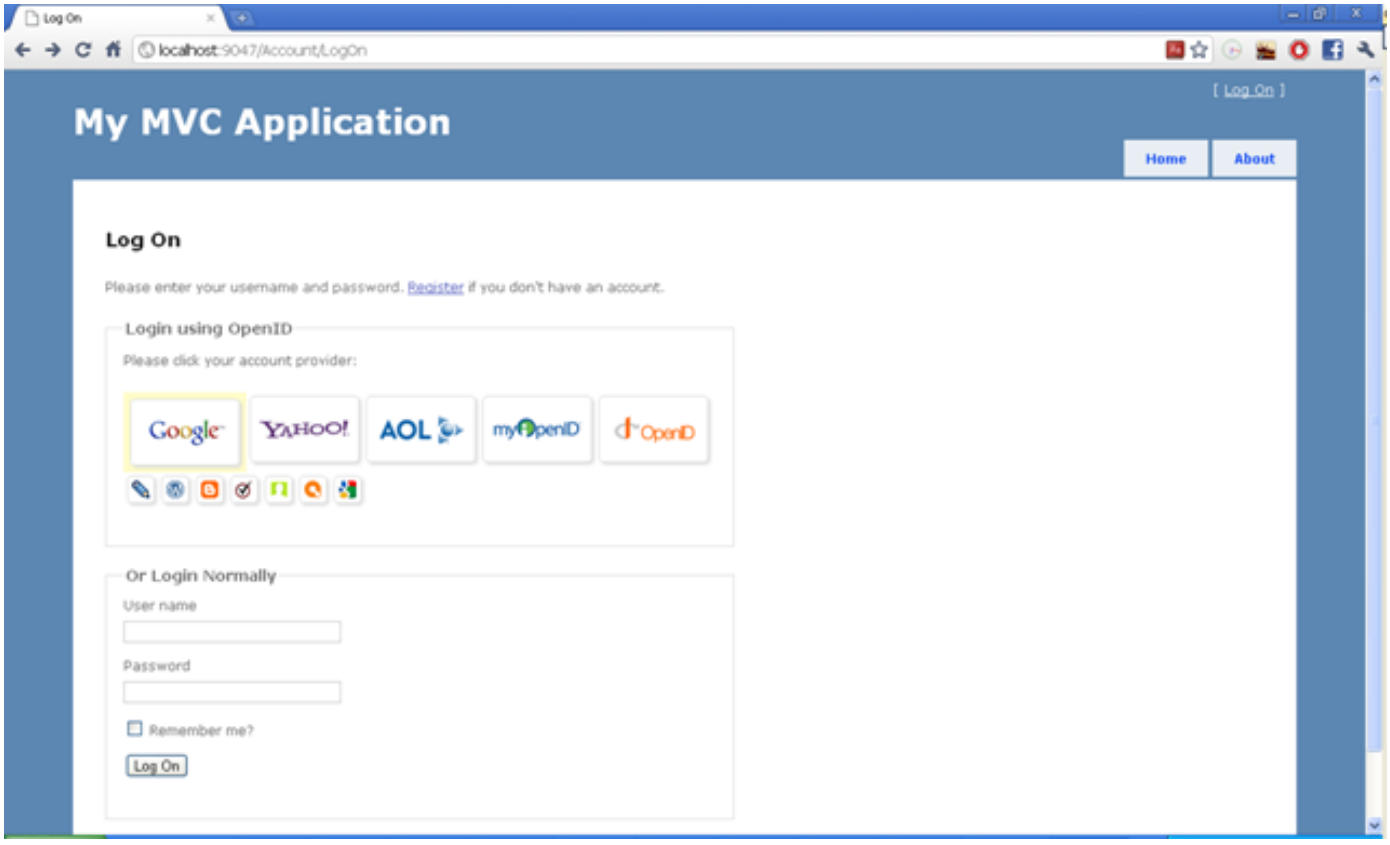
```
                    }

                    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

عنوان: **Google OpenID Authentication در ASP.NET با استفاده از DotNetOpenAuth**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۰۵/۰۴ ۵:۴۷
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, C#, Security, OpenID, DotNetOpenAuth

در پست‌های قبلی OpenID چیست؟
استفاده از OpenID در وب سایت جهت احراز هویت کاربران با مفاهیم OpenID تاحدودی آشنا شدیم
در این پست می‌خواهیم با یک مثال ساده نشان دهیم که سایت ما چگونه با استفاده از OpenID Authentication می تواند از اکانت گوگل استفاده کرده و کاربر در وب سایت ما شناسایی شود.
برای اینکار ابتدا
1- کتابخانه DotNetOpenAuth را از طریق NuGet به لیست رفرنس‌های پروژه وب خود اضافه نمایید
2- یک صفحه بنام Login.aspx یا هر نام دلخواهی را به پروژه خود اضافه نمایید. در نهایت کد Html صفحه شما به ید به صورت زیر باشد.

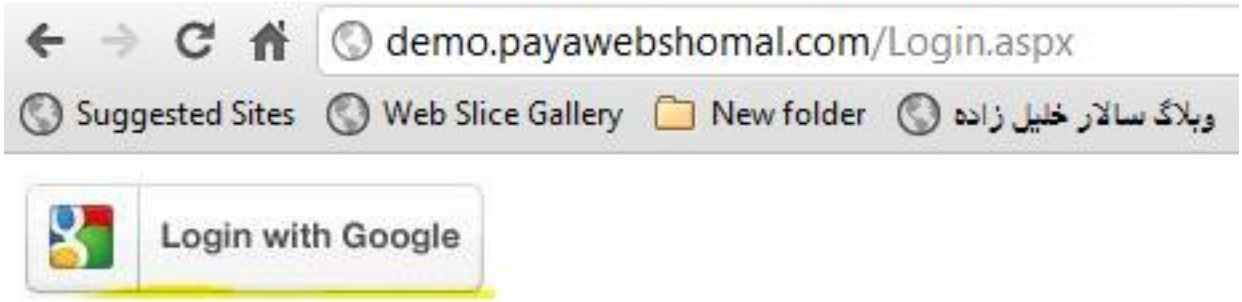
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Login.aspx.cs" Inherits="OAuthWebTest.Login"
%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div id="loginform">
        <div id="NotLoggedIn" runat="server">
            <asp:ImageButton ID="ButtonLoginWithGoogle" runat="server" ImageUrl="Google.JPG"
                OnCommand="ButtonLoginWithGoogle_Click"
CommandArgument="https://www.google.com/accounts/o8/id" />
            <p />
            <asp:Label runat="server" ID="LabelMessage" />
        </div>
    </div>
    </form>
</body>
</html>
```

در کد Html بالا به خصوصیت CommandArgument از کنترل ImageButton دقت نمایید که دارای مقدار "https://www.google.com/accounts/o8/id" می باشد. در واقع این آدرس باید برای اکانت گوگل جهت احراز هویت توسط OpenID استفاده شود. (آدرس API گوگل برای استفاده از این سرویس)
3- در قسمت کد نویسی صفحه کدهای زیر را وارد نمایید.

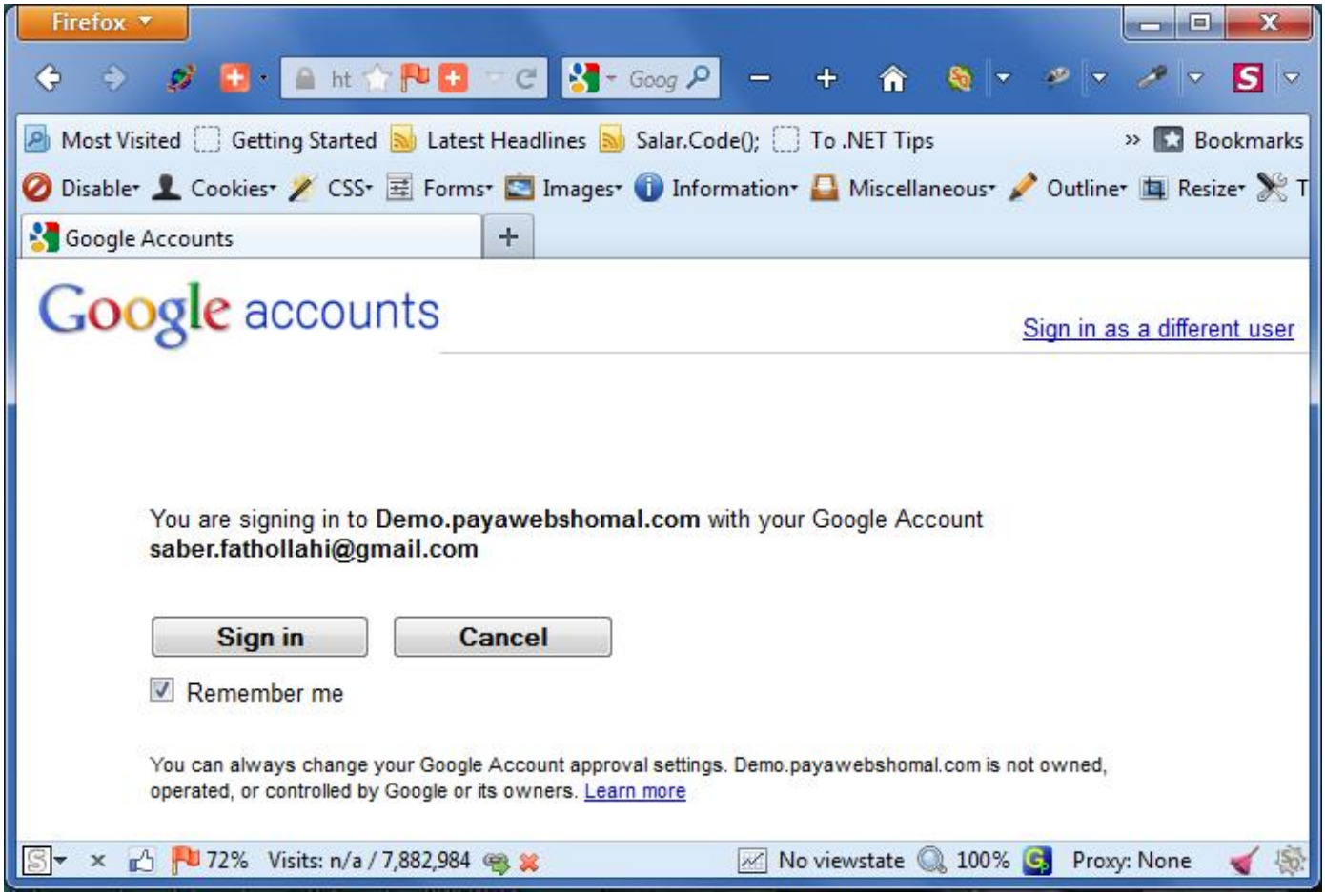
```
using System;
using System.Web.UI;
using System.Web.UI.WebControls;
using DotNetOpenAuth.OpenId.RelyingParty;

namespace OAuthWebTest
{
    public partial class Login : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            OpenIdRelyingParty openIdRelyingParty = new OpenIdRelyingParty();
            var response = openIdRelyingParty.GetResponse();
            if (response == null) return;
            switch (response.Status)
            {
                case AuthenticationStatus.Authenticated:
                    NotLoggedIn.Visible = false;
                    Session["GoogleIdentifier"] = response.ClaimedIdentifier.ToString();
                    Response.Redirect("Default.aspx");
                    break;
                case AuthenticationStatus.Canceled:
                    LabelMessage.Text = "Cancelled.";
                    break;
```

```
                case AuthenticationStatus.Failed:
                    LabelMessage.Text = "Login Failed.";
                    break;
            }
        }

        protected void ButtonLoginWithGoogle_Click(object src, CommandEventArgs e)
        {
            string discoveryUri = e.CommandArgument.ToString();
            OpenIdRelyingParty openId = new OpenIdRelyingParty();
            var returnToUrl = new UriBuilder(Request.Url) { Query = "" };
            var request = openId.CreateRequest(discoveryUri, returnToUrl.Uri, returnToUrl.Uri);
            request.RedirectToProvider();
        }
    }
}
```

همانگونه که مشاهده می‌کنید در رویداد کلیک دکمه لوگین ButtonLoginWithGoogle_Click ابتدا آدرس یکتا گوگل (مقدار CommandArgument اختصاص داده شده به کنترل ImageButton) به همراه آدرس صفحه جاری وب سایت توسط متد CreateRequest از شی openId ترکیب شده و یک درخواست (Request) ساخته شده و در نهایت برای سرویس دهنده گوگل ارسال می‌شود. با اجرای پروژه با تصویر زیر روبرو می‌شوید بروی کلید لوگین کلیک نمایید.



در واقع با کلیک روی دکمه مورد نظر تکه کدی که در بالا شرح داده شد اجرا شده و ما را به صفحه ای شبیه تصویر پایین هدایت می‌کند



در صورتی که کلید Sign In انتخاب شود شما به سایت (همین برنامه) اجازه داده اید که از اطلاعات حساب کاربری شما استفاده کند. پس از کلیک به برنامه اصلی (طبق آدرس بازگشت تعیین شده در رویداد ButtonLoginWithGoogle_Click) در رویداد PageLoad صفحه لاگین اطلاعات بازگشتی از سایت سرویس دهنده (در اینجا گوگل) چک می‌شود و با توجه با پاسخ مورد نظر عملیاتی انجام می‌شود، مثلا در صورتی که شما تایید کرده باشید اطلاعات شناسایی شما توسط گوگل در کلیدی به نام GoogleIdentifier در سشن ذهیره شده و شما به صفحه اصلی سایت هدایت می‌شوید.

دموی پروژه در این آدرس قرار داده شده است.

سورس پروژه از این آدرس قابل دسترسی است.

توجه: در این آدرس شرح داده شده که چگونه دسترسی سایت‌های دیگر به اکانت گوگل خود را قطع کنید

در پست‌های آینده اتصال به تویتر و ف ی س بوک و سایت‌های دیگر توضیح داده خواهد شد.

نظرات خوانندگان

نویسنده: مجتبی چنانی
تاریخ: ۱۳۹۱/۰۵/۰۴ ۹:۰

با سلام و تشکر از پست خوبتون

در زمانی که ما از سیستم‌های ورود و ثبت کاربران شرکت‌های دیگر استفاده می‌کنیم آیا می‌توانیم لاگ گرفته یا اینکه برای خودمان یک صفحه داشته باشیم تا ورود و خروج‌های اکانت‌های درون وبسایتمان را بررسی کنیم یا اینکه خودمان باید این بخش را کدنویسی کنیم؟

یک سئوال دیگر این است که زمانی که از openid های شرکت‌های دیگه استفاده می‌کنیم فقط احراز هویت را از این سرویس‌ها دریافت می‌کنیم یا اینکه در همه صفحات و دیگر کارهای کاربر نظارت به صورت خودکار انجام می‌شود یا اینکه باز هم باید کدنویسی کنیم؟

نویسنده: صابر فتح اللهی
تاریخ: ۱۳۹۱/۰۵/۰۴ ۱۴:۹

در صورتی که بخواهید لاگ بندازید باید خود کد نویسی کرده و در دیتابیس ذخیره کنید
بله می‌توان در هر صفحه با استفاده از کلیدی که در سشن کاربر ذخیره کرده اید (با توجه به اینکه سشن کاربران جداست) می‌توانید وجود کاربر را چک کنید

نویسنده: سینا علیزاده
تاریخ: ۱۳۹۲/۰۹/۰۳ ۱۳:۲۹

آقا عالی بود ممنون
میشه برای facebook و توییتر هم بزارید ، واقعا بهش نیاز دارم ممنون میشم
کارتون عالیه موفق باشید

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۹/۰۳ ۱۳:۳۵

نگاهی هم به پروژه DotNetAuth داشته باشید.

نویسنده: دادخواه
تاریخ: ۱۳۹۲/۱۰/۱۶ ۱۱:۳۱

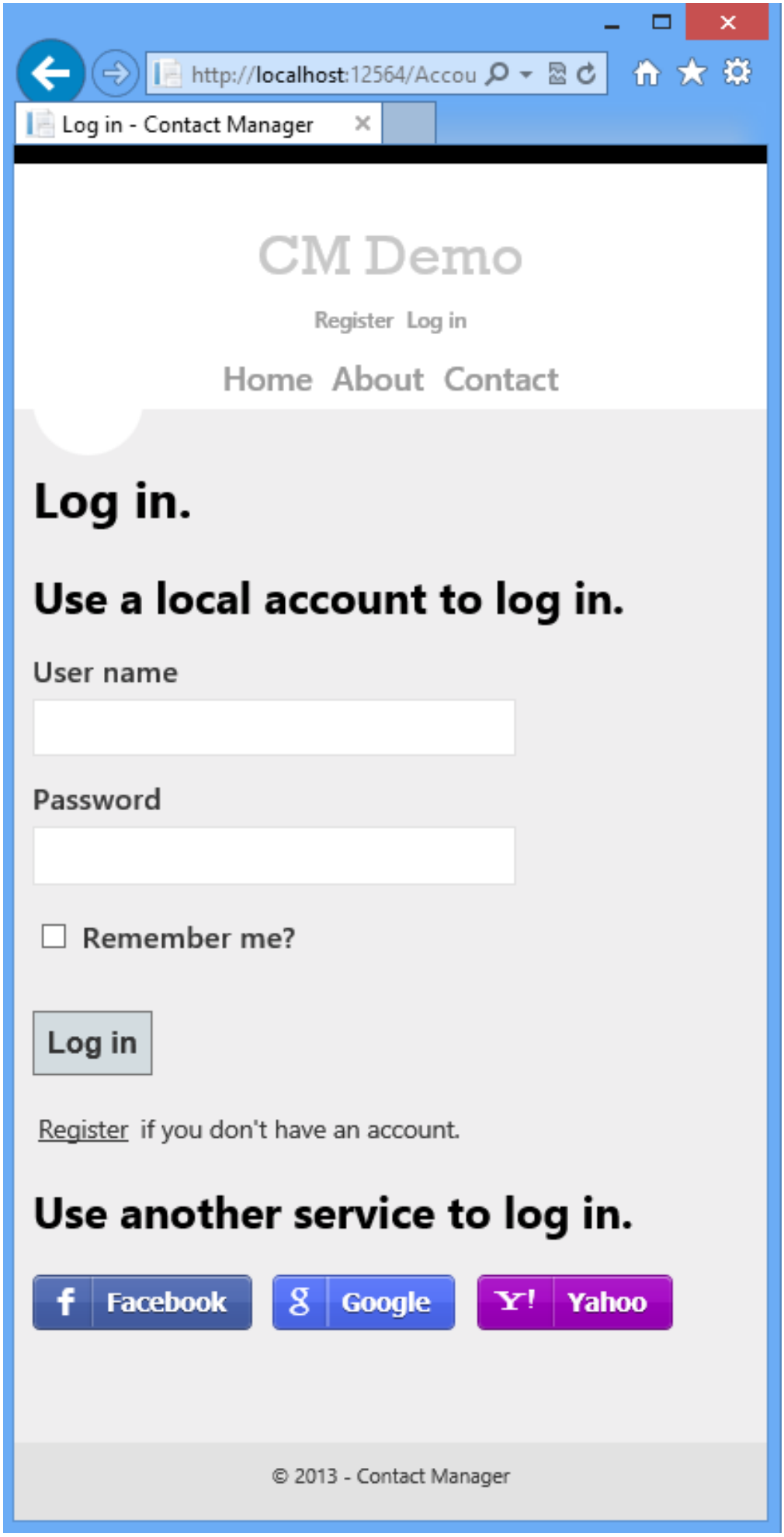
سلام.

این سرویس چطوری برای اکانت‌های یاهو کار می‌کنه؟

عنوان: **توزیع یک اپلیکیشن ASP.NET MVC 5 روی Windows Azure**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۹ ۱۰:۵۰
آدرس: www.dotnettips.info
گروه‌ها: Tips, OpenID, ASP.NET MVC 5, ASP.NET Identity, Windows Azure, OAuth, MVC, Security

این مقاله به شما نشان می‌دهد چگونه یک اپلیکیشن وب ASP.NET MVC 5 بسازید که کاربران را قادر می‌سازد با اطلاعات Facebook یا Google احراز هویت شده و به سایت وارد شوند. همچنین این اپلیکیشن را روی Windows Azure توزیع (Deploy) خواهید کرد.
می توانید بصورت رایگان یک حساب کاربری Windows Azure بسازید. اگر هم Visual Studio 2013 را ندارید، بسته SDK بصورت خودکار Visual Studio 2013 for Web را نصب می‌کند. پس از آن می‌توانید به توسعه رایگان اپلیکیشن‌های Azure بپردازید، اگر می‌خواهید از Visual Studio 2012 استفاده کنید به این مقاله مراجعه کنید. این مقاله نسبت به لینک مذکور بسیار ساده‌تر است.
این مقاله فرض را بر این می‌گذارد که شما هیچ تجربه ای در کار با Windows Azure ندارید. در انتهای این مقاله شما یک اپلیکیشن مبتنی بر داده (data-driven) و امن خواهید داشت که در فضای رایانش ابری اجرا می‌شود.
چیزی که شما یاد می‌گیرید:
چطور یک اپلیکیشن وب ASP.NET MVC 5 بسازید و آن را روی یک وب سایت Windows Azure منتشر کنید.
چگونه از OpenID ، OAuth و سیستم عضویت ASP.NET برای ایمن سازی اپلیکیشن خود استفاده کنید.
چگونه از API جدید سیستم عضویت برای مدیریت اعضا و نقش‌ها استفاده کنید.
چگونه از یک دیتابیس SQL برای ذخیره داده‌ها در Windows Azure استفاده کنید.

شما یک اپلیکیشن مدیریت تماس (Contact Manager) ساده خواهید نوشت که بر پایه ASP.NET MVC 5 بوده و از Entity Framework برای دسترسی داده استفاده می‌کند. تصویر زیر صفحه ورود نهایی اپلیکیشن را نشان می‌دهد.


http://localhost:12564/Accou
Log in - Contact Manager
CM Demo
Register Log in
Home About Contact
Log in.
Use a local account to log in.
User name
Password
Remember me?
Log in
Register if you don't have an account.
Use another service to log in.
Facebook
Google
Yahoo
© 2013 - Contact Manager

توجه: برای تمام کردن این مقاله به یک حساب کاربری Windows Azure نیاز دارید، که بصورت رایگان می‌توانید آن را بسازید. برای اطلاعات بیشتر به Windows Azure Free Trial مراجعه کنید.

در این مقاله:

برپایی محیط توسعه (development environment)

برپایی محیط Windows Azure

ایجاد یک اپلیکیشن ASP.NET MVC 5

توزیع اپلیکیشن روی Windows Azure

افزودن یک دیتابیس به اپلیکیشن

افزودن یک OAuth Provider

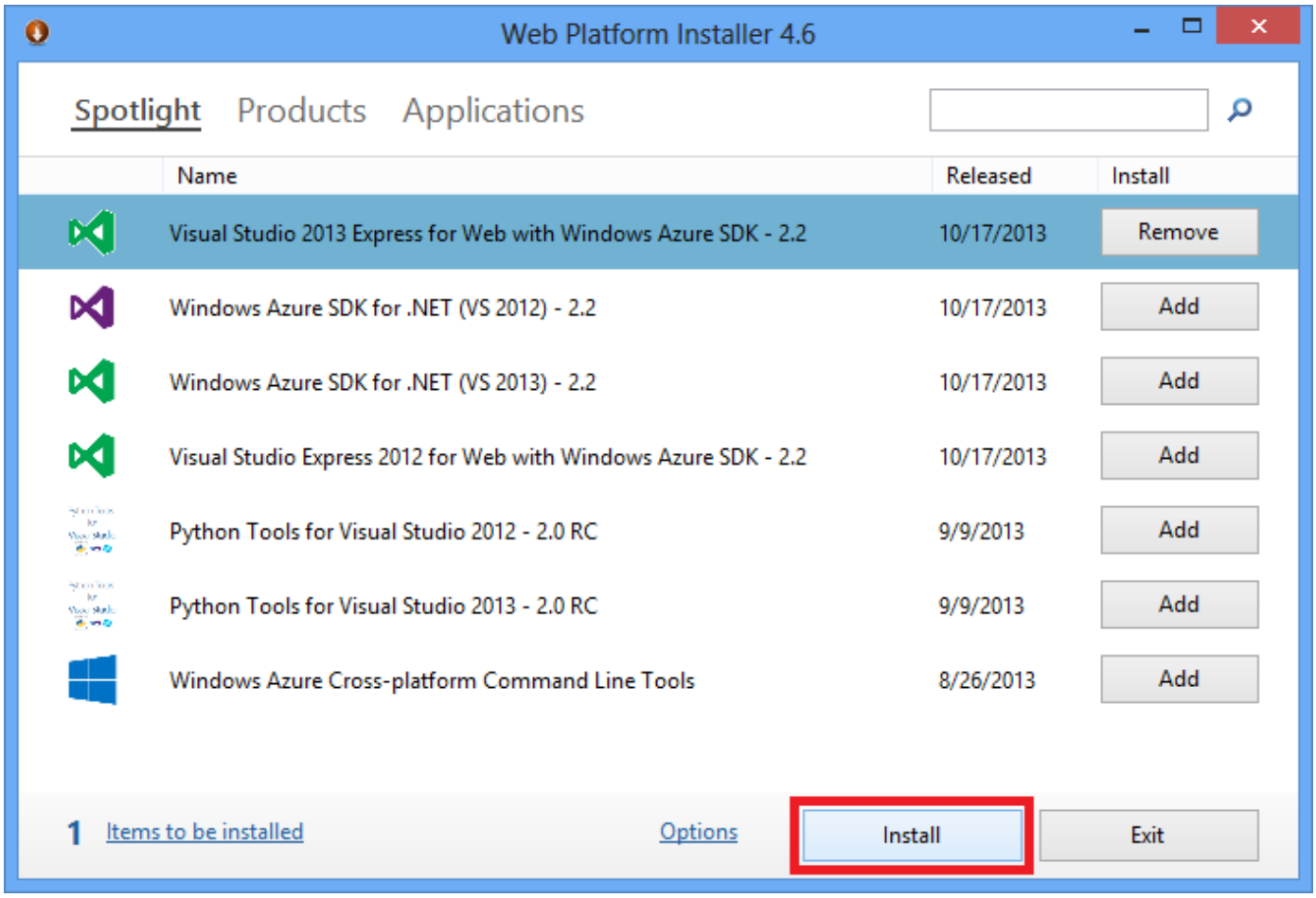
استفاده از Membership API

توزیع اپلیکیشن روی Windows Azure

قدم‌های بعدی

برپایی محیط توسعه

برای شروع Windows Azure SDK for .NET را نصب کنید. برای اطلاعات بیشتر به Windows Azure SDK for Visual Studio 2013 مراجعه کنید. بسته به اینکه کدام یک از وابستگی‌ها را روی سیستم خود دارید، پروسه نصب می‌تواند از چند دقیقه تا نزدیک دو ساعت طول بکشد. توسط Web Platform می‌توانید تمام نیازمندی‌های خود را نصب کنید.



هنگامی که این مرحله با موفقیت به اتمام رسید، تمام ابزار لازم برای شروع به کار را در اختیار دارید.

برپایی محیط Windows Azure

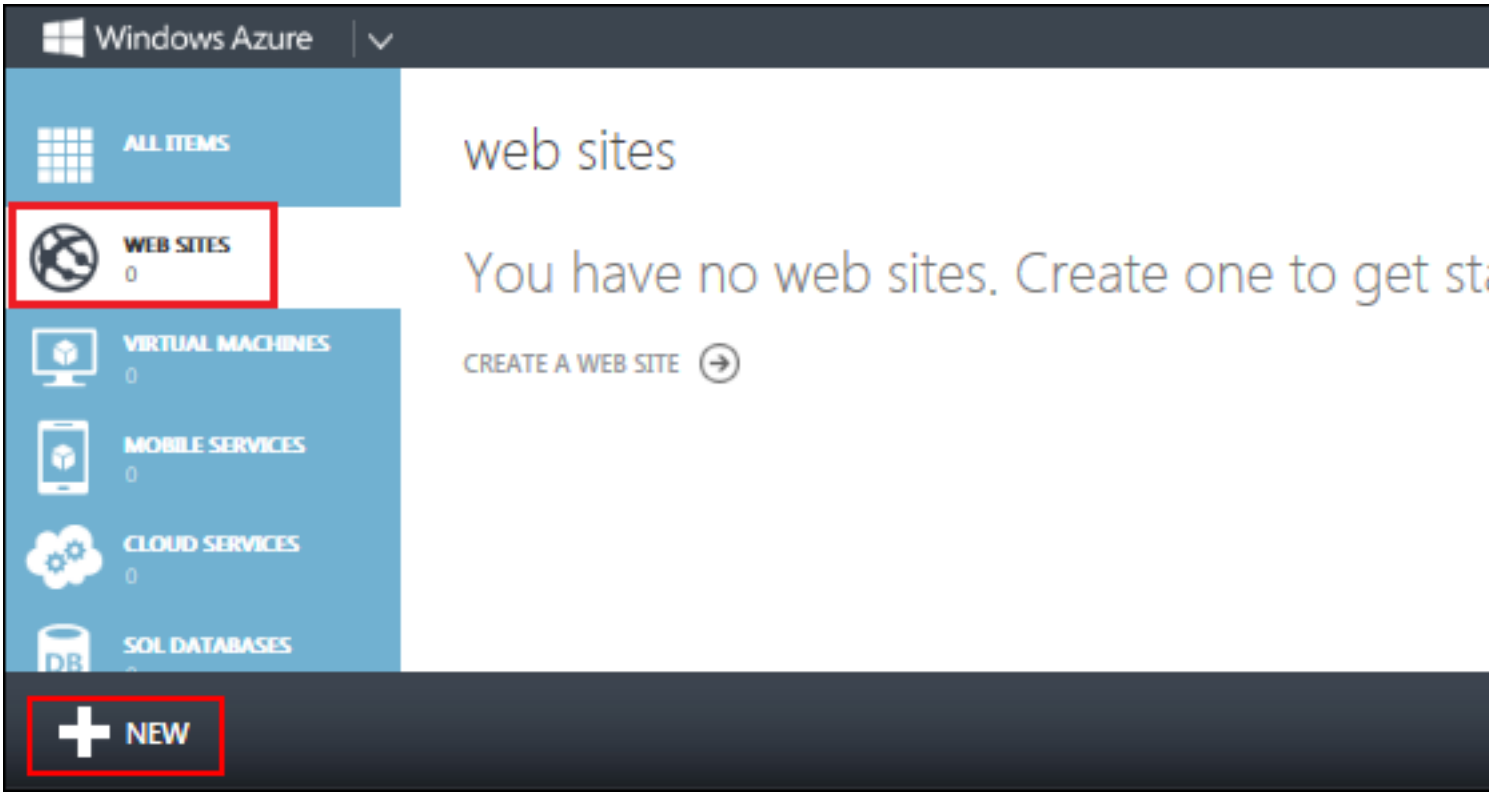
در قدم بعدی باید یک وب سایت Windows Azure و یک دیتابیس بسازیم.

ایجاد یک وب سایت و دیتابیس در Windows Azure

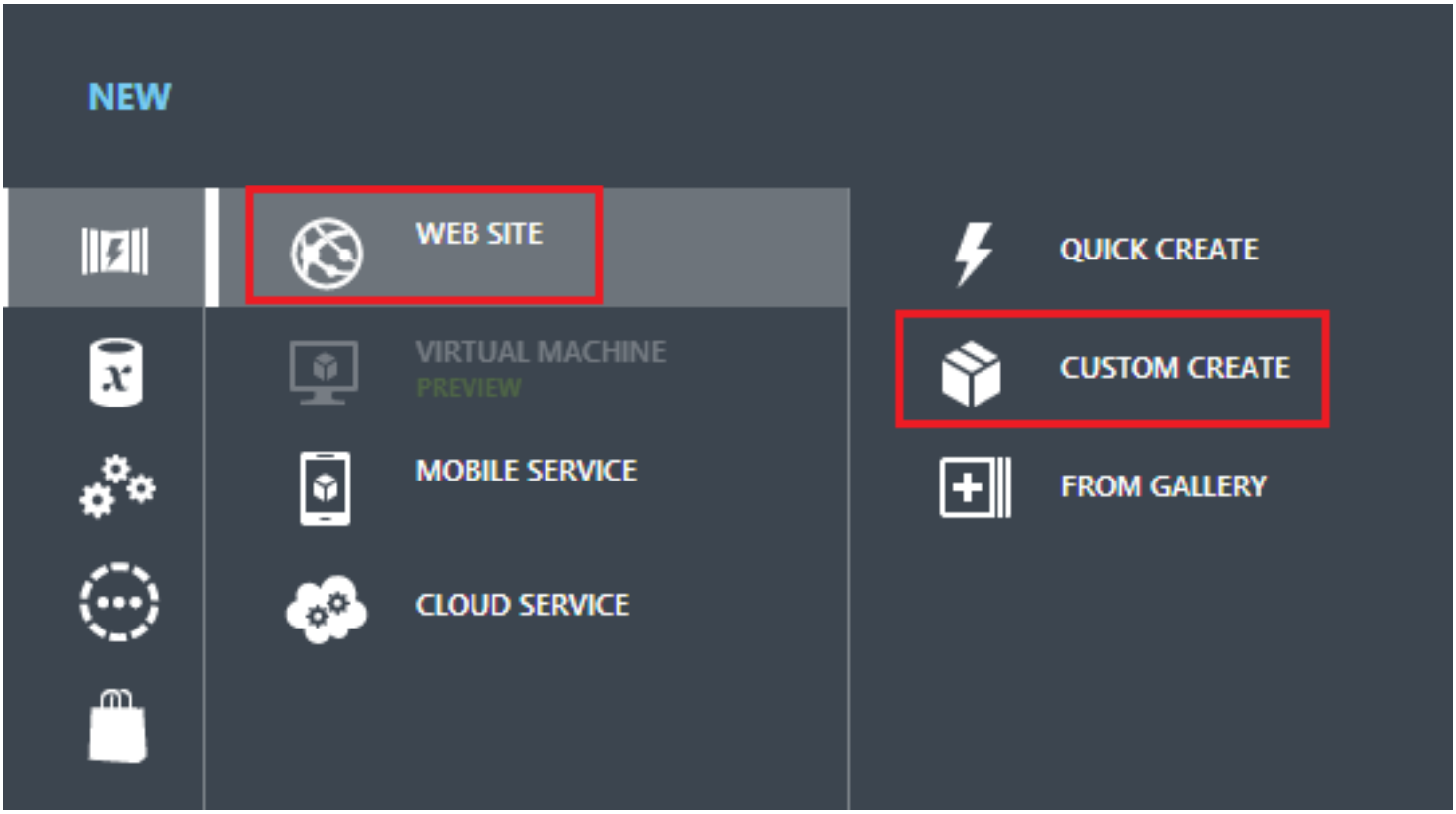
وب سایت Windows Azure شما در یک محیط اشتراکی (shared) میزبانی می‌شود، و این بدین معنا است که وب سایت‌های شما روی ماشین‌های مجازی (virtual machines) اجرا می‌شوند که با مشتریان دیگر Windows Azure به اشتراک گذاشته شده اند. یک محیط میزبانی اشتراکی گزینه ای کم هزینه برای شروع کار با رایانش‌های ابری است. اگر در آینده ترافیک وب سایت شما رشد چشم گیری داشته باشد، می‌توانید اپلیکیشن خود را طوری توسعه دهید که به نیازهای جدید پاسخگو باشد و آن را روی یک ماشین مجازی اختصاصی (dedicated VMs) میزبانی کنید. اگر معماری پیچیده‌تری نیاز دارید، می‌توانید به یک سرویس Windows Azure Cloud مهاجرت کنید. سرویس‌های ابری روی ماشین‌های مجازی اختصاصی اجرا می‌شوند که شما می‌توانید تنظیمات آنها را بر اساس نیازهای خود پیکربندی کنید.

Windows Azure SQL Database یک سرویس دیتابیس رابطه ای (relational) و مبتنی بر Cloud است که بر اساس تکنولوژی‌های SQL Server ساخته شده. ابزار و اپلیکیشن هایی که با SQL Server کار می‌کنند با SQL Database نیز می‌توانند کار کنند.

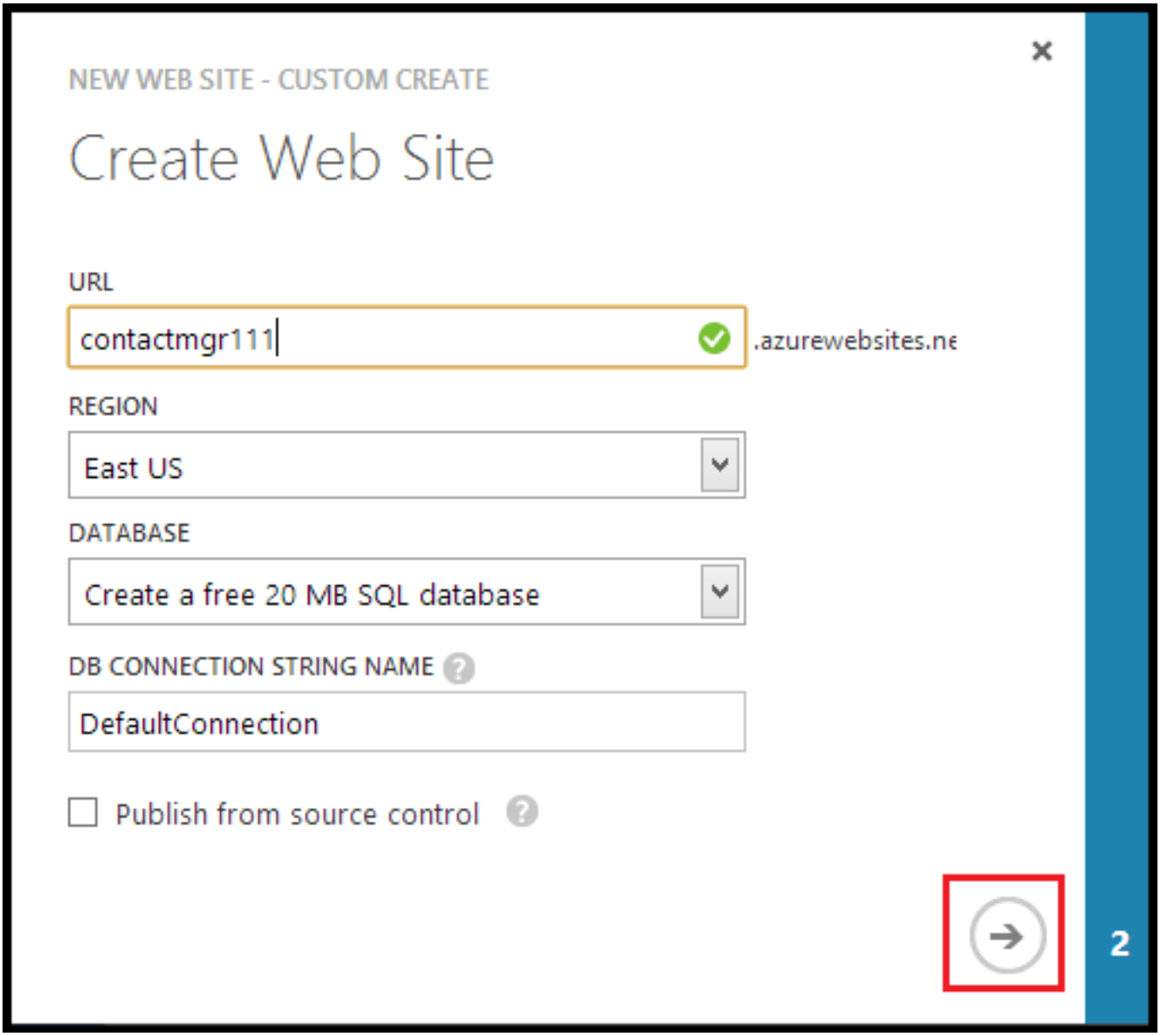
در پرتال مدیریتی Windows Azure روی **Web Sites** در قسمت چپ صفحه کلیک کنید، و گزینه **New** را برگزینید.



روی **Web Site** و سپس **Custom Create** کلیک کنید.



در مرحله **Create Web Site** در قسمت **URL** یک رشته وارد کنید که آدرسی منحصر بفرد برای اپلیکیشن شما خواهد بود. آدرس کامل وب سایت شما، ترکیبی از مقدار این فیلد و مقدار روبروی آن است.



در لیست **Database** گزینه **Create a free 20 MB SQL Database** را انتخاب کنید.

در لیست **Region** همان مقداری را انتخاب کنید که برای وب سایت تان انتخاب کرده اید. تنظیمات این قسمت مشخص می‌کند که ماشین مجازی (VM) شما در کدام مرکز داده (data center) خواهد بود.

در قسمت **DB Connection String Name** مقدار پیش فرض *DefaultConnection* را بپذیرید.

دکمه فلش پایین صفحه را کلیک کنید تا به مرحله بعد، یعنی مرحله **Specify Database Settings** بروید.

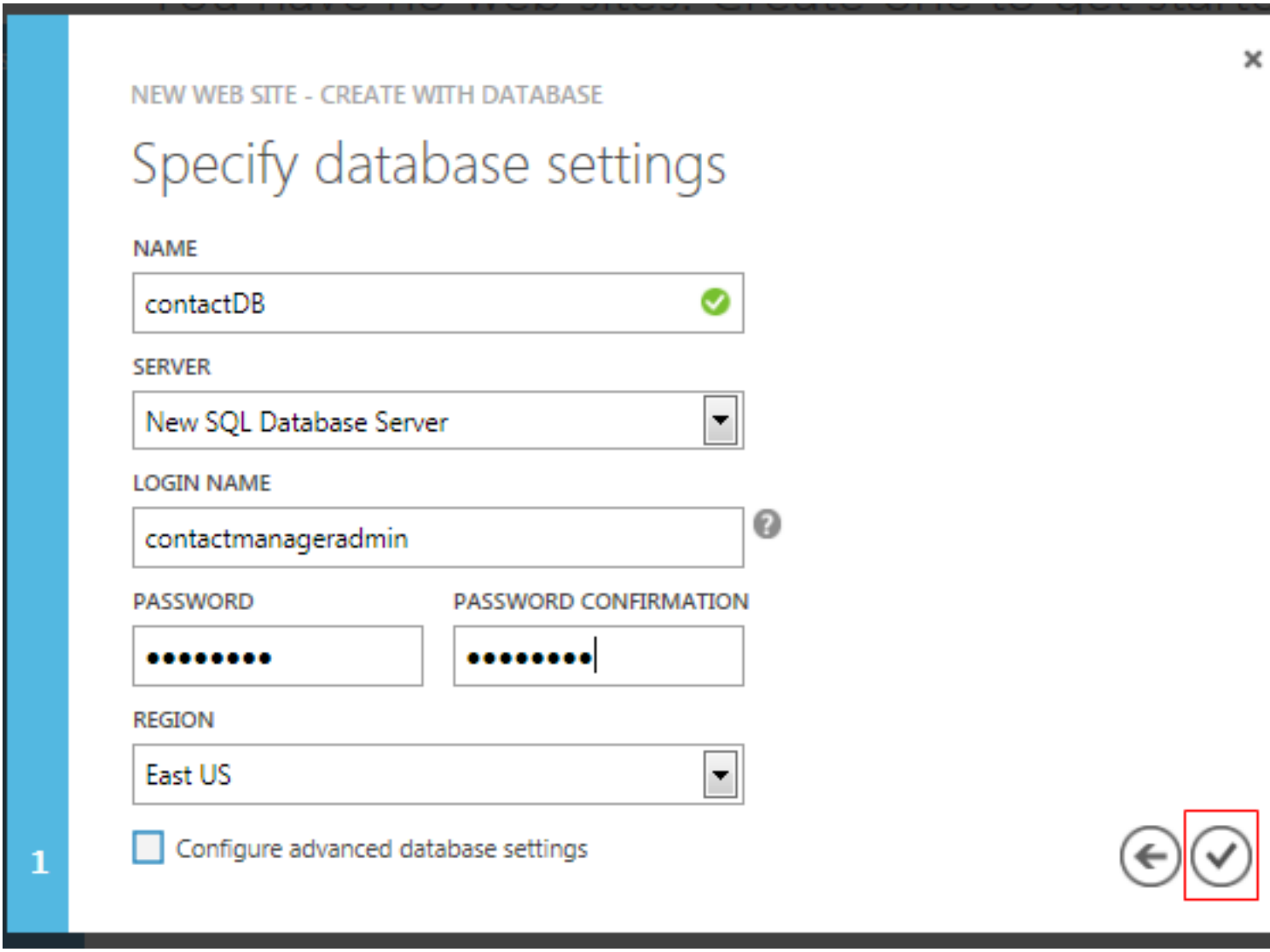
در قسمت **Name** مقدار *ContactDB* را وارد کنید (تصویر زیر).

در قسمت **Server** گزینه **New SQL Database Server** را انتخاب کنید. اگر قبلا دیتابیس ساخته اید می‌توانید آن را از کنترل dropdown انتخاب کنید.

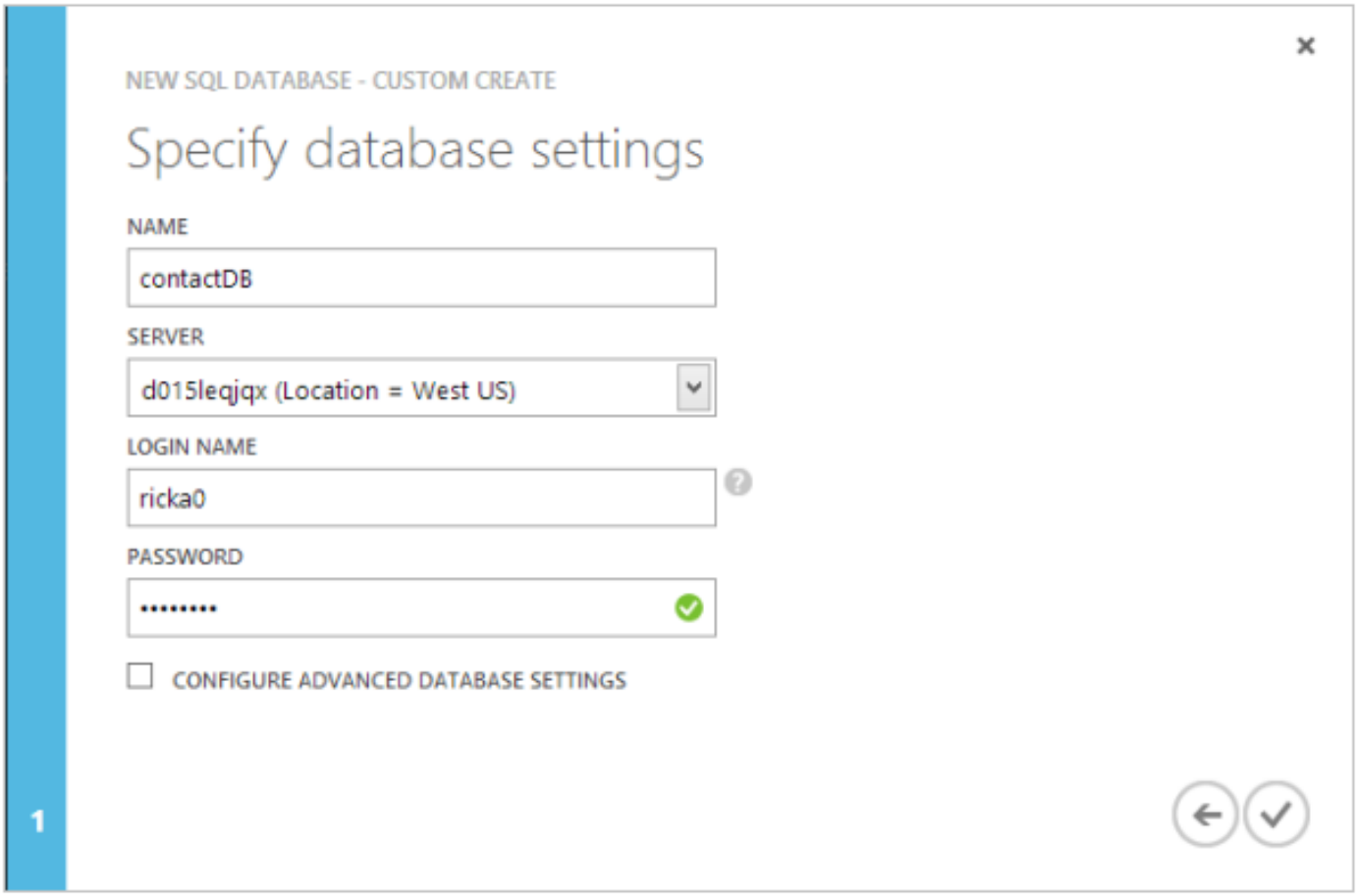
مقدار قسمت **Region** را به همان مقداری که برای ایجاد وب سایت تان تنظیم کرده اید تغییر دهید.

یک **Login Name** و **Password** مدیر (administrator) وارد کنید. اگر گزینه **New SQL Database server** را انتخاب کرده اید، چنین کاربری وجود ندارد و در واقع اطلاعات یک حساب کاربری جدید را وارد می‌کنید تا بعدا هنگام دسترسی به دیتابیس از آن استفاده کنید. اگر دیتابیس دیگری را از لیست انتخاب کرده باشید، اطلاعات یک حساب کاربری موجود از شما دریافت خواهد شد. در مثال این مقاله ما گزینه **Advanced** را رها می‌کنیم. همچنین در نظر داشته باشید که برای دیتابیس‌های رایگان تنها از یک Collation می‌توانید استفاده کنید.

دکمه تایید پایین صفحه را کلیک کنید تا مراحل تمام شود.



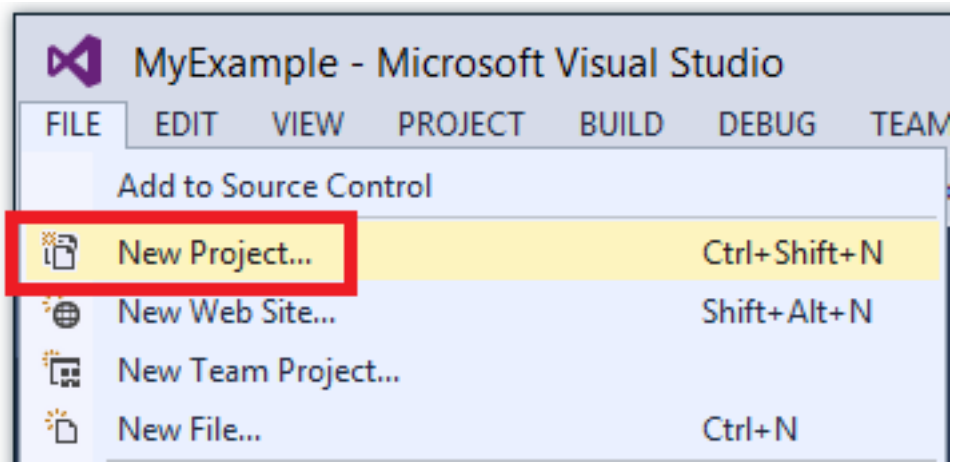
تصویر زیر استفاده از یک SQL Server و حساب کاربری موجود (existing) را نشان می‌دهد.



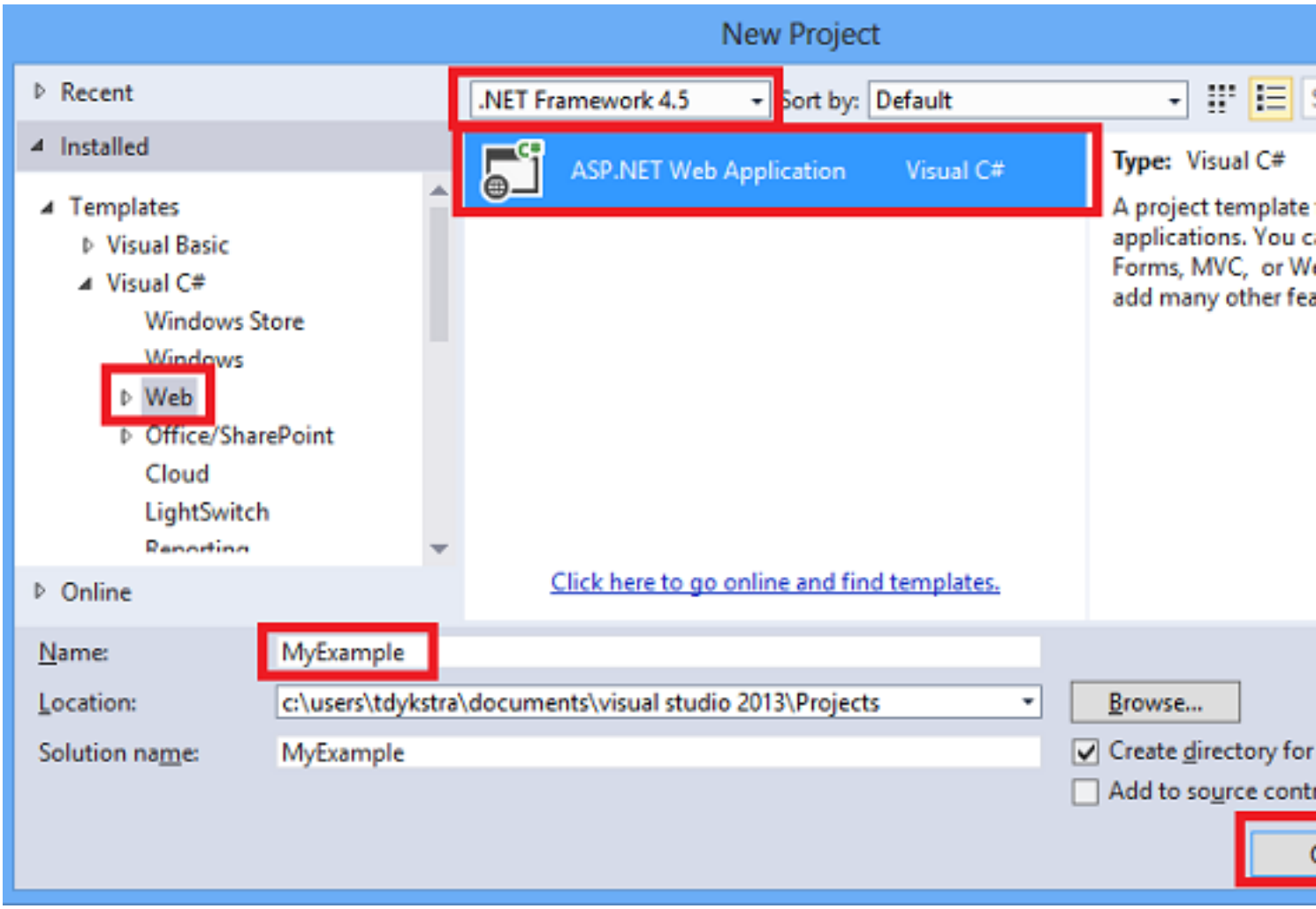
پرتال مدیریتی پس از اتمام مراحل، به صفحه وب سایت‌ها باز می‌گردد. ستون **Status** نشان می‌دهد که سایت شما در حال ساخته شدن است. پس از مدتی (معمولا کمتر از یک دقیقه) این ستون نشان می‌دهد که سایت شما با موفقیت ایجاد شده. در منوی پیمایش سمت چپ، تعداد سایت هایی که در اکانت خود دارید در کنار آیکون **Web Sites** نمایش داده شده است، تعداد دیتابیس‌ها نیز در کنار آیکون **SQL Databases** نمایش داده می‌شود.

یک اپلیکیشن ASP.NET MVC 5 بسازید

شما یک وب سایت Windows Azure ساختید، اما هنوز هیچ محتوایی در آن وجود ندارد. قدم بعدی ایجاد یک اپلیکیشن وب در ویژوال استودیو و انتشار آن است. ابتدا یک پروژه جدید بسازید.

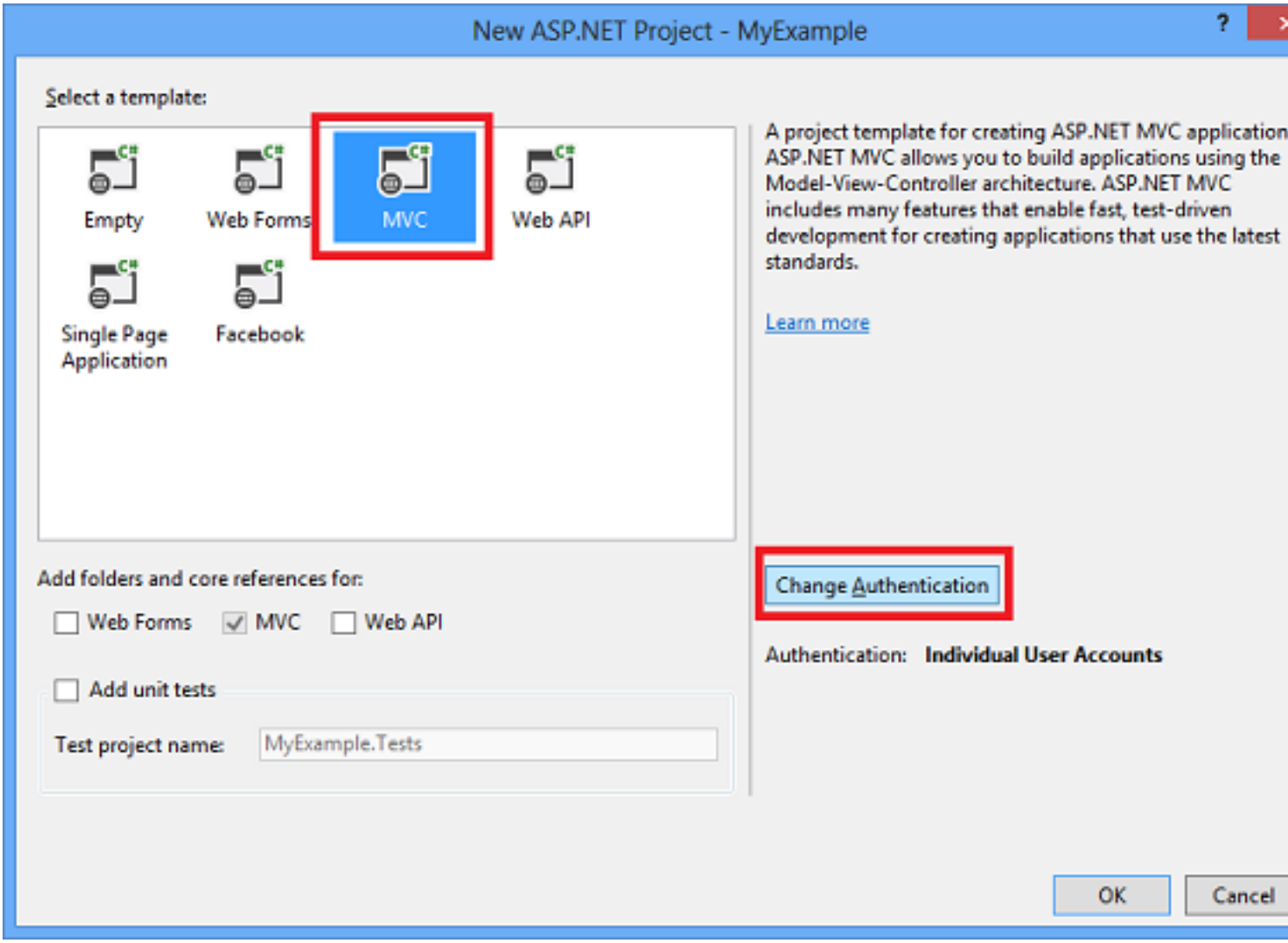


نوع پروژه را **ASP.NET Web Application** انتخاب کنید.



نکته: در تصویر بالا نام پروژه "MyExample" است اما حتما نام پروژه خود را به "ContactManager" تغییر دهید. قطعه کدهایی که در ادامه مقاله خواهید دید نام پروژه را ContactManager فرض می‌کنند.

در دیالوگ جدید ASP.NET نوع اپلیکیشن را **MVC** انتخاب کنید و دکمه **Change Authentication** را کلیک کنید.



گزینه پیش فرض **Individual User Accounts** را بپذیرید. برای اطلاعات بیشتر درباره متدهای دیگر احراز هویت به این لینک مراجعه کنید. دکمه‌های OK را کلیک کنید تا تمام مراحل تمام شوند.

تنظیم تیتر و پاورقی سایت

فایل *_Layout.cshtml* را باز کنید. دو نمونه از متن "My ASP.NET MVC Application" را با عبارت "Contact Manager" جایگزین کنید.

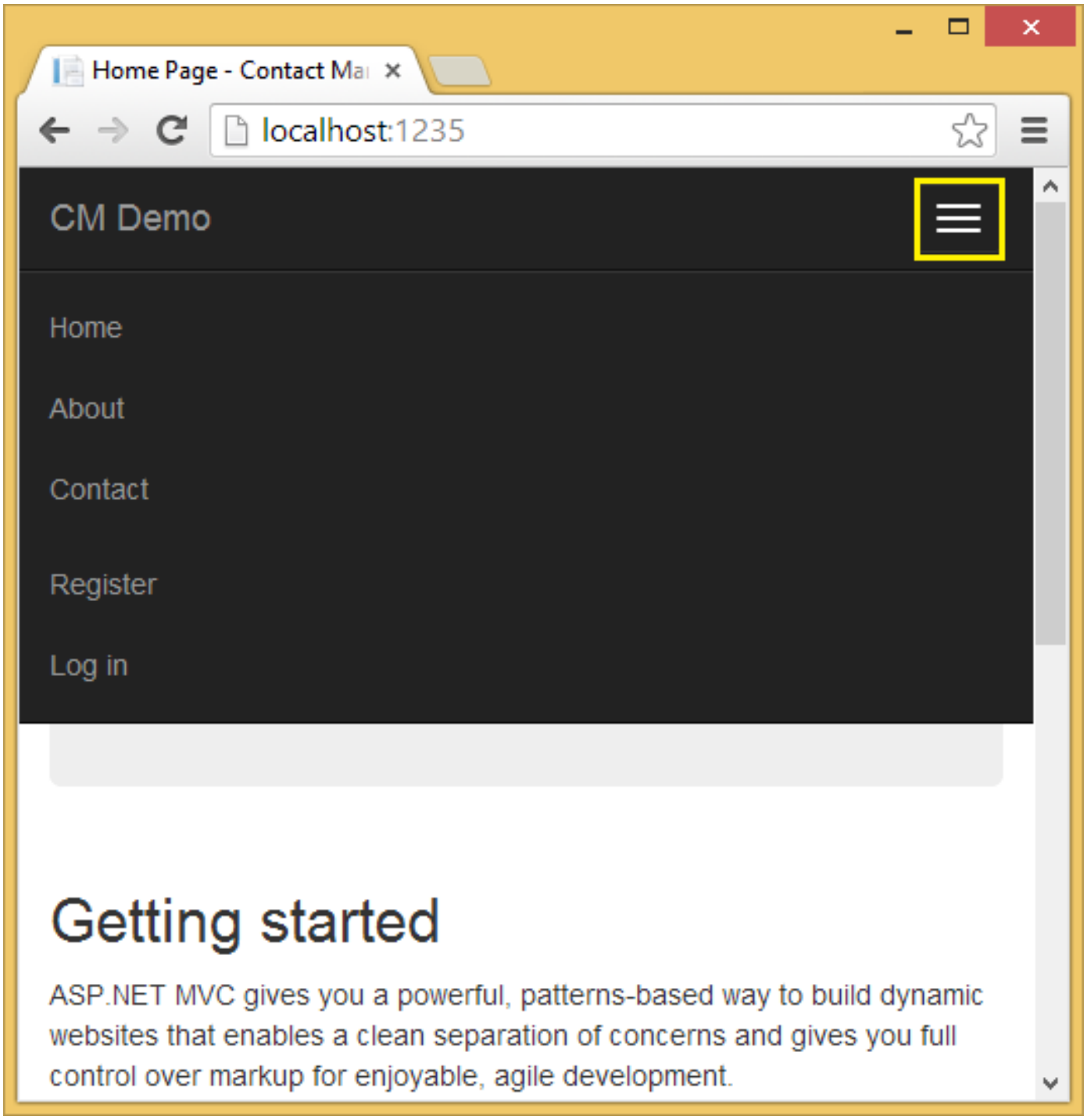
عبارت "Application name" را هم با "CM Demo" جایگزین کنید.

اولین Action Link را ویرایش کنید و مقدار *Home* را با *Cm* جایگزین کنید تا از *CmController* استفاده کند.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - Contact Manager</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")

</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="colla
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("CM Demo", "Index", "Cm", null, new { @class
            </div>
            <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>
                </ul>
                @Html.Partial("_LoginPartial")
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - Contact Manager</p>
        </footer>
    </div>

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>
```

اپلیکیشن را بصورت محلی اجرا کنید

اپلیکیشن را با Ctrl + F5 اجرا کنید. صفحه اصلی باید در مرورگر پیش فرض باز شود.

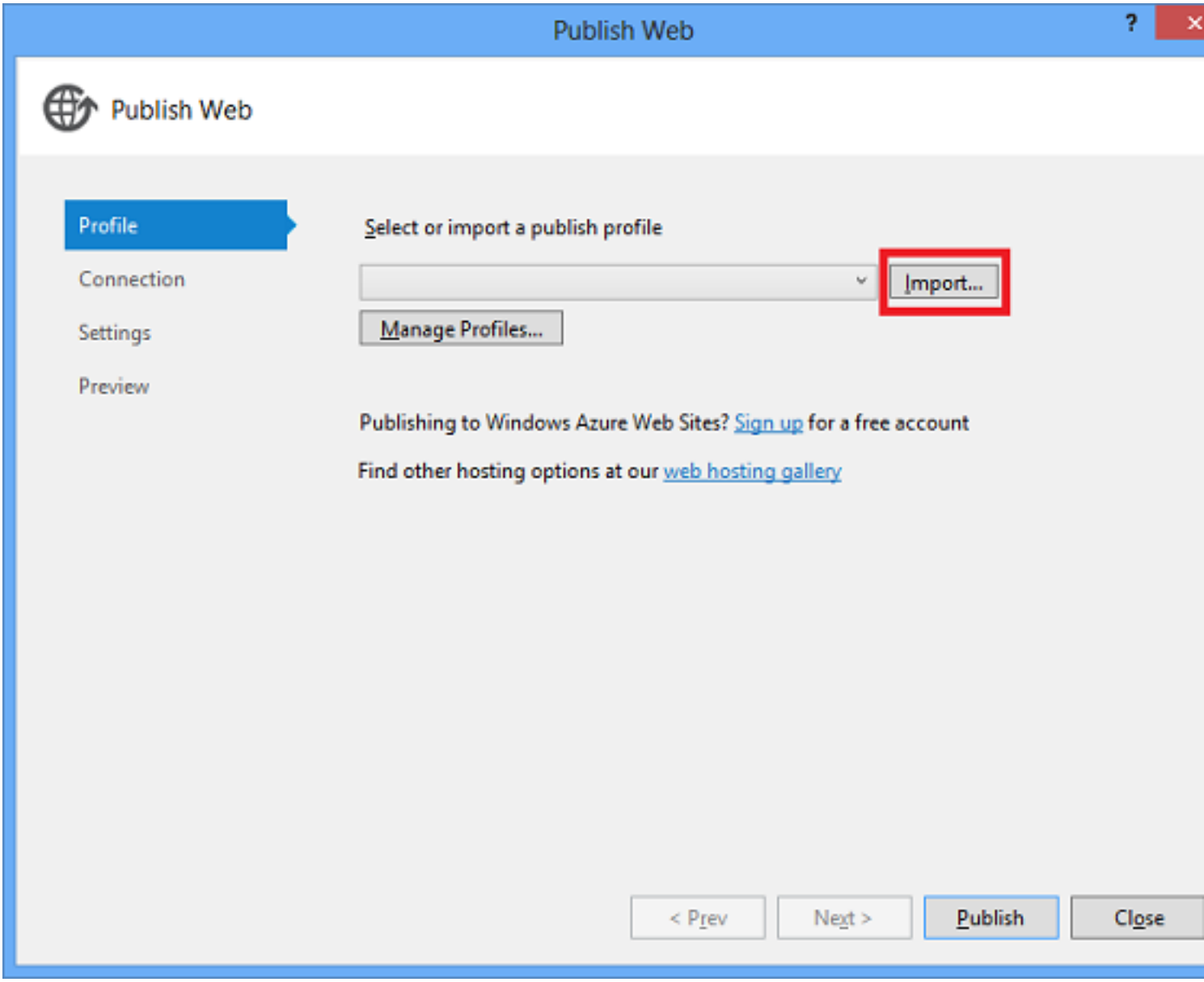


اپلیکیشن شما فعلا آماده است و می‌توانید آن را روی Windows Azure توزیع کنید. بعدا دیتابیس و دسترسی داده نیز اضافه خواهد شد.

اپلیکیشن را روی Windows Azure منتشر کنید

در ویژوال استودیو روی نام پروژه کلیک راست کنید و گزینه **Publish** را انتخاب کنید. ویزارد **Publish Web** باز می‌شود.

در قسمت **Profile** روی **Import** کلیک کنید.



حال دیالوگ **Import Publish Profile** نمایش داده می‌شود.

یکی از متدهای زیر را استفاده کنید تا ویژوال استودیو بتواند به اکانت Windows Azure شما متصل شود.
روی **Sign In** کلیک کنید تا با وارد کردن اطلاعات حساب کاربری وارد Windows Azure شوید.

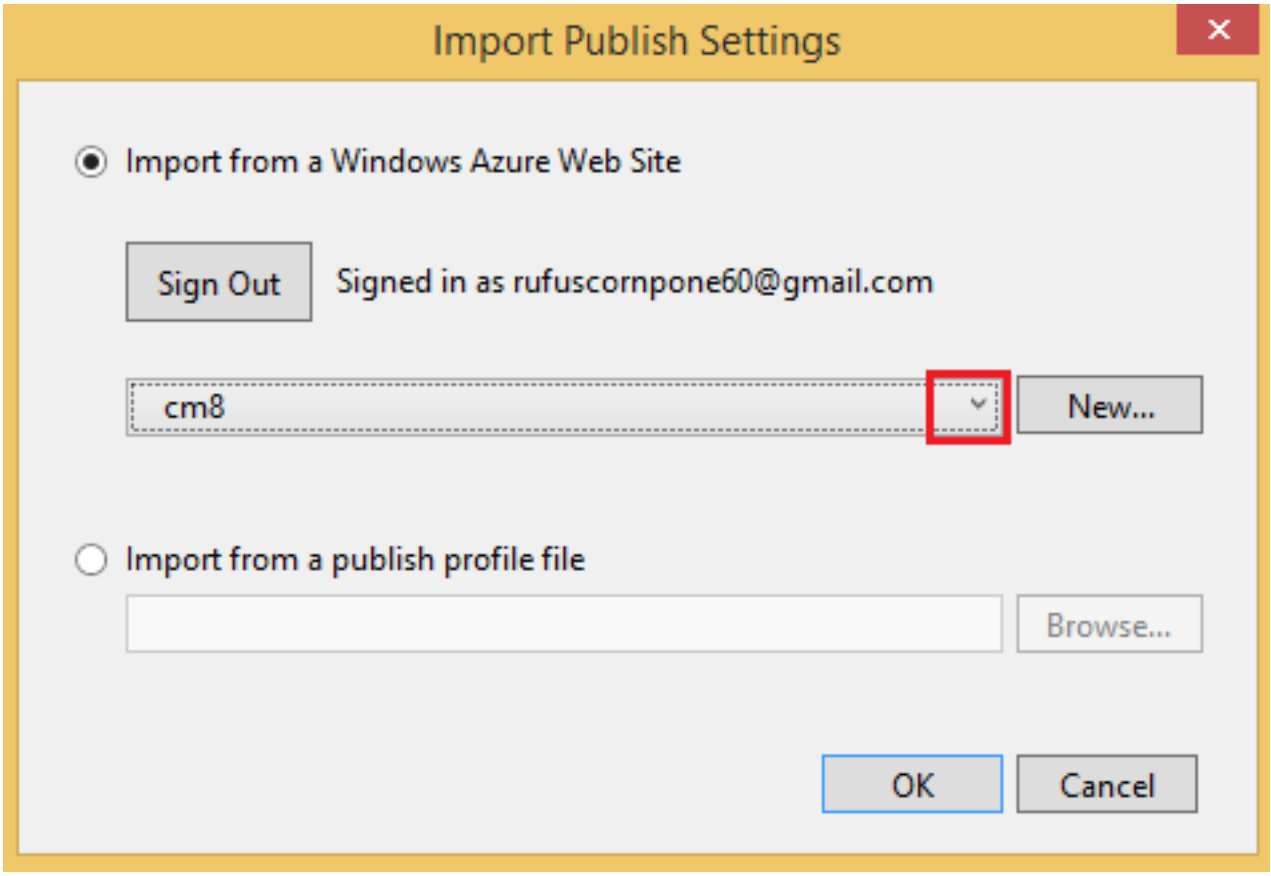
این روش ساده‌تر و سریع‌تر است، اما اگر از آن استفاده کنید دیگر قادر به مشاهده Windows Azure SQL Database یا Mobile Services در پنجره **Server Explorer** نخواهید بود.
روی **Manage subscriptions** کلیک کنید تا یک management certificate نصب کنید، که دسترسی به حساب کاربری شما را ممکن می‌سازد.

در دیالوگ باکس **Manage Windows Azure Subscriptions** به قسمت **Certificates** بروید. سپس **Import** را کلیک کنید. مراحل را دنبال کنید تا یک فایل subscription را بصورت دانلود دریافت کنید (فایل‌های *publishsettings*.) که اطلاعات اکانت Windows Azure شما را دارد.

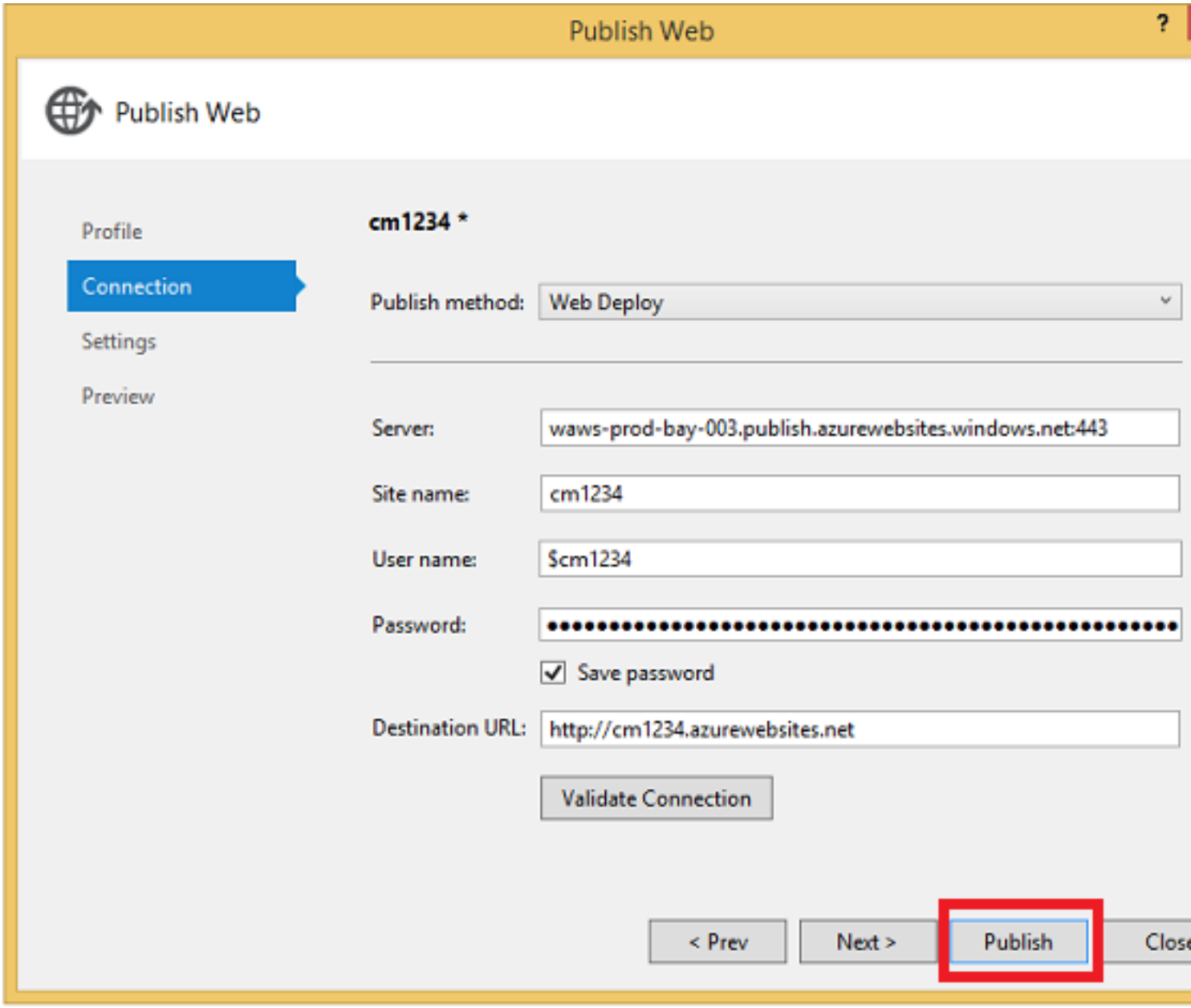
نکته امنیتی: این فایل تنظیمات را بیرون از پوشه‌های سورس کد خود دانلود کنید، مثلا پوشه Downloads. پس از اتمام عملیات Import هم این فایل را حذف کنید. کاربر مخربی که به این فایل دسترسی پیدا کند قادر خواهد بود تا سرویس‌های Windows Azure شما را کاملا کنترل کند.

برای اطلاعات بیشتر به How to Connect to Windows Azure from Visual Studio مراجعه کنید.

در دیالوگ باکس **Import Publish Profile** وب سایت خود را از لیست انتخاب کنید و OK را کلیک کنید.



در دیالوگ باکس **Publish Web** روی **Publish** کلیک کنید.



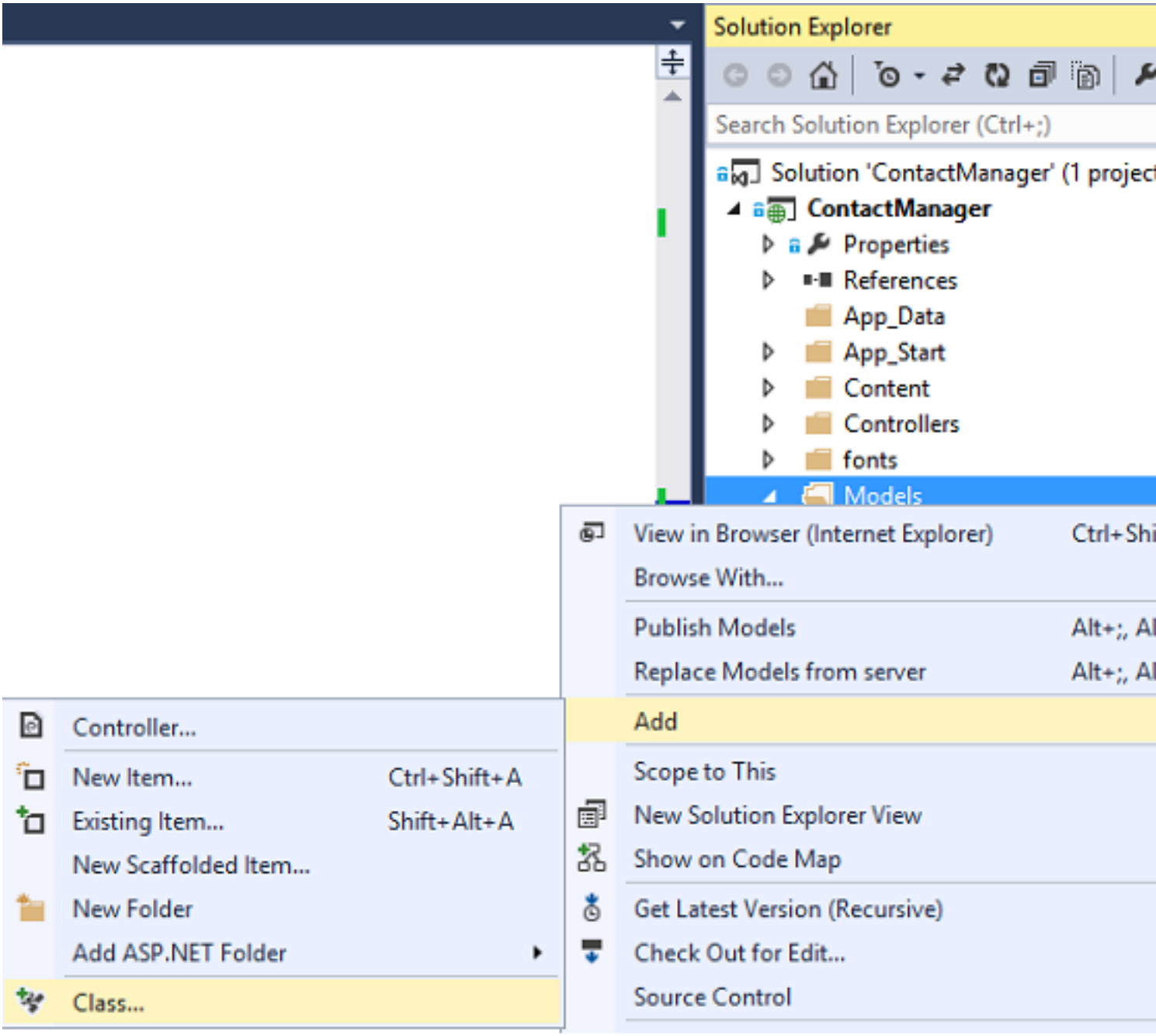
اپلیکیشن شما حالا در فضای ابری اجرا می‌شود. دفعه بعد که اپلیکیشن را منتشر کنید تنها فایل‌های تغییر کرده (یا جدید) آپلود خواهند شد.

یک دیتابیس به اپلیکیشن اضافه کنید

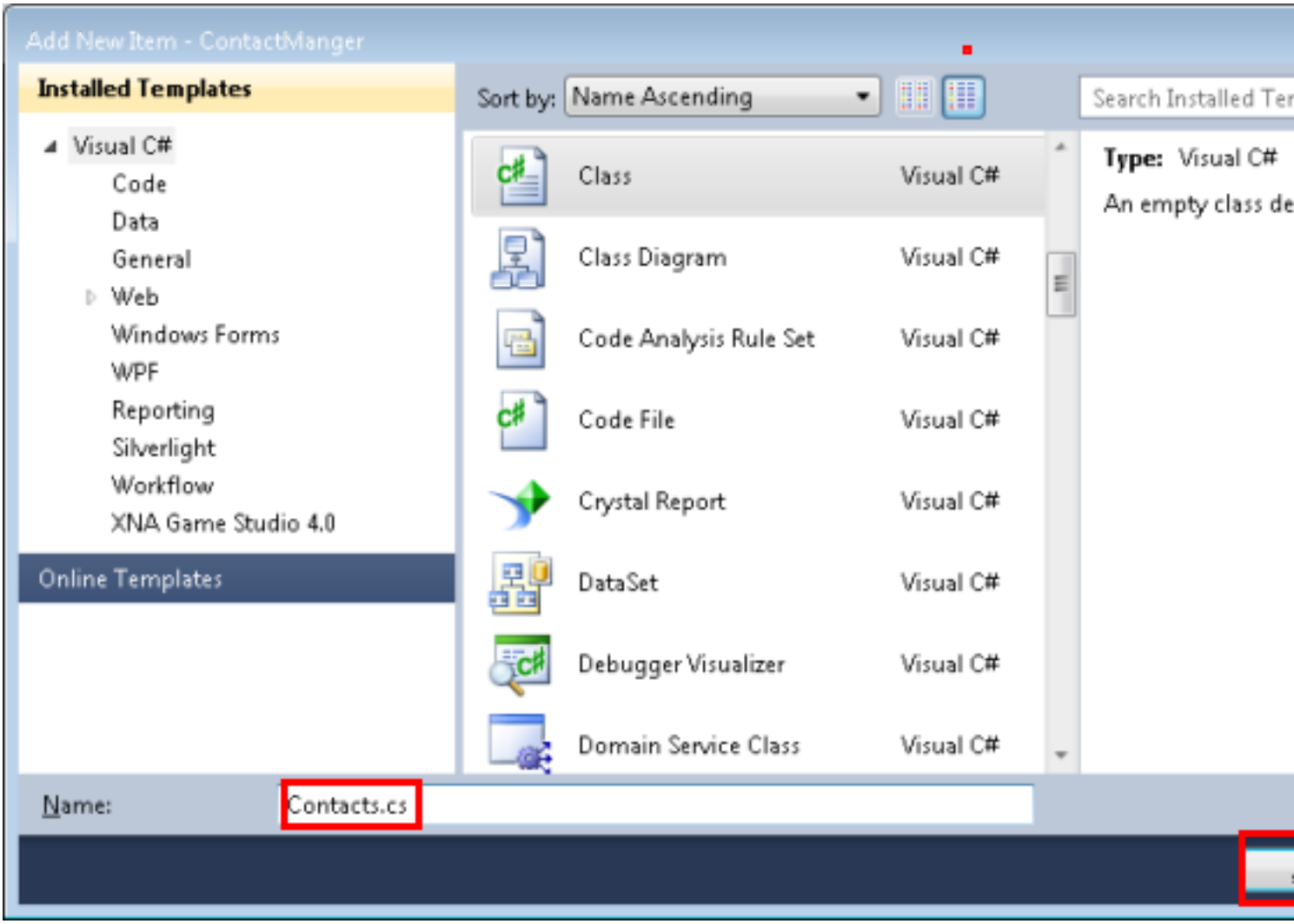
در مرحله بعد یک دیتابیس خواهیم ساخت تا اپلیکیشن ما بتواند اطلاعات را نمایش دهد و ویرایش کند. برای ایجاد دیتابیس و دسترسی به داده‌ها از Entity Framework استفاده خواهیم کرد.

کلاس‌های مدل Contacts را اضافه کنید

در پوشه Models پروژه یک کلاس جدید ایجاد کنید.



نام کلاس را به *Contact.cs* تغییر دهید و دکمه Add را کلیک کنید.



کد فایل Contact.cs را با قطعه کد زیر مطابقت دهید.

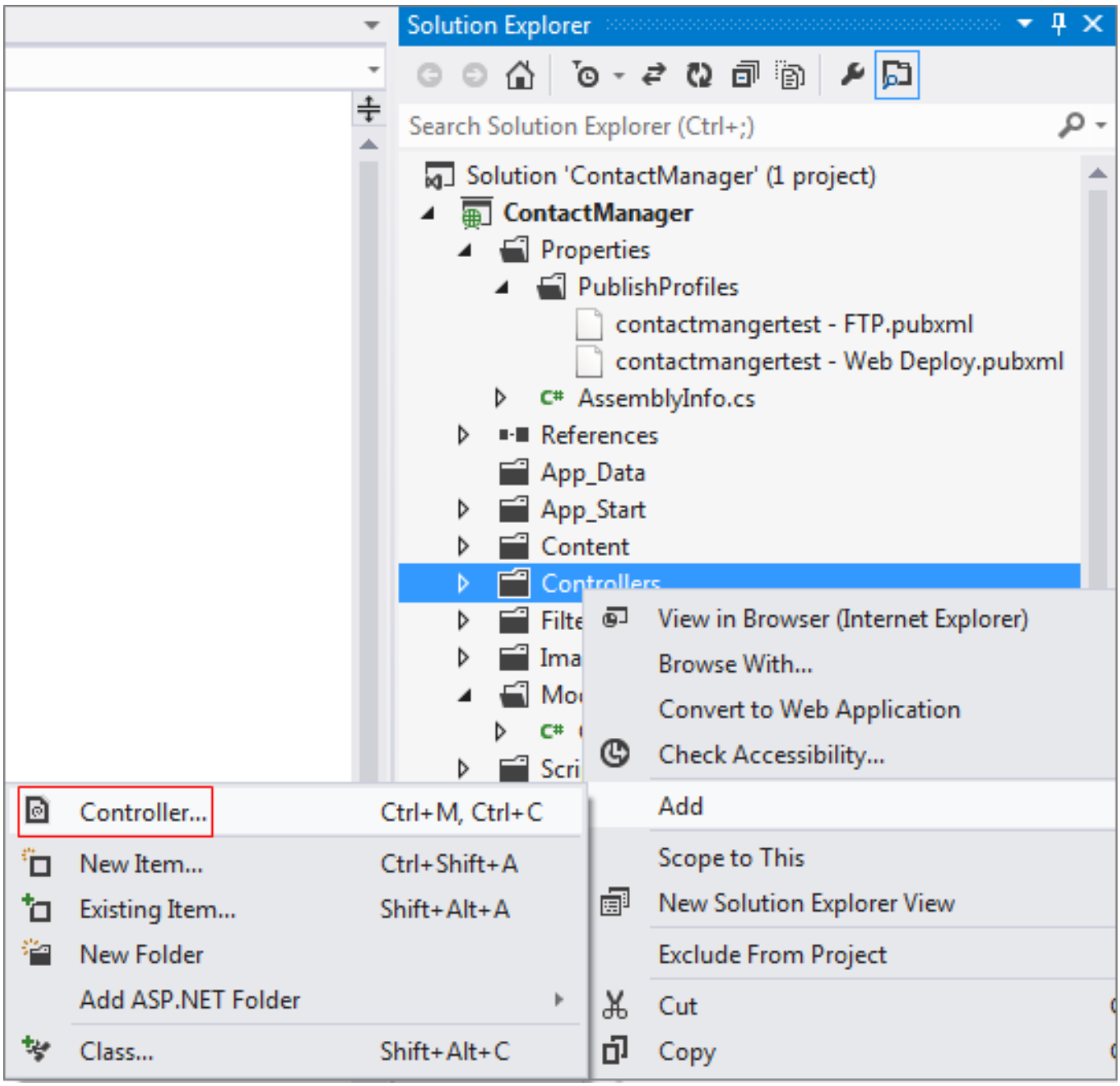
```
using System.ComponentModel.DataAnnotations;
using System.Globalization;
namespace ContactManager.Models
{
    public class Contact
    {
        public int ContactId { get; set; }
        public string Name { get; set; }
        public string Address { get; set; }
        public string City { get; set; }
        public string State { get; set; }
        public string Zip { get; set; }
        [DataType(DataType.EmailAddress)]
        public string Email { get; set; }
    }
}
```

این کلاس موجودیت Contact را در دیتابیس معرفی می‌کند. داده هایی که می‌خواهیم برای هر رکورد ذخیره کنیم تعریف شده اند، بعلاوه یک فیلد Primary Key که دیتابیس به آن نیاز دارد.

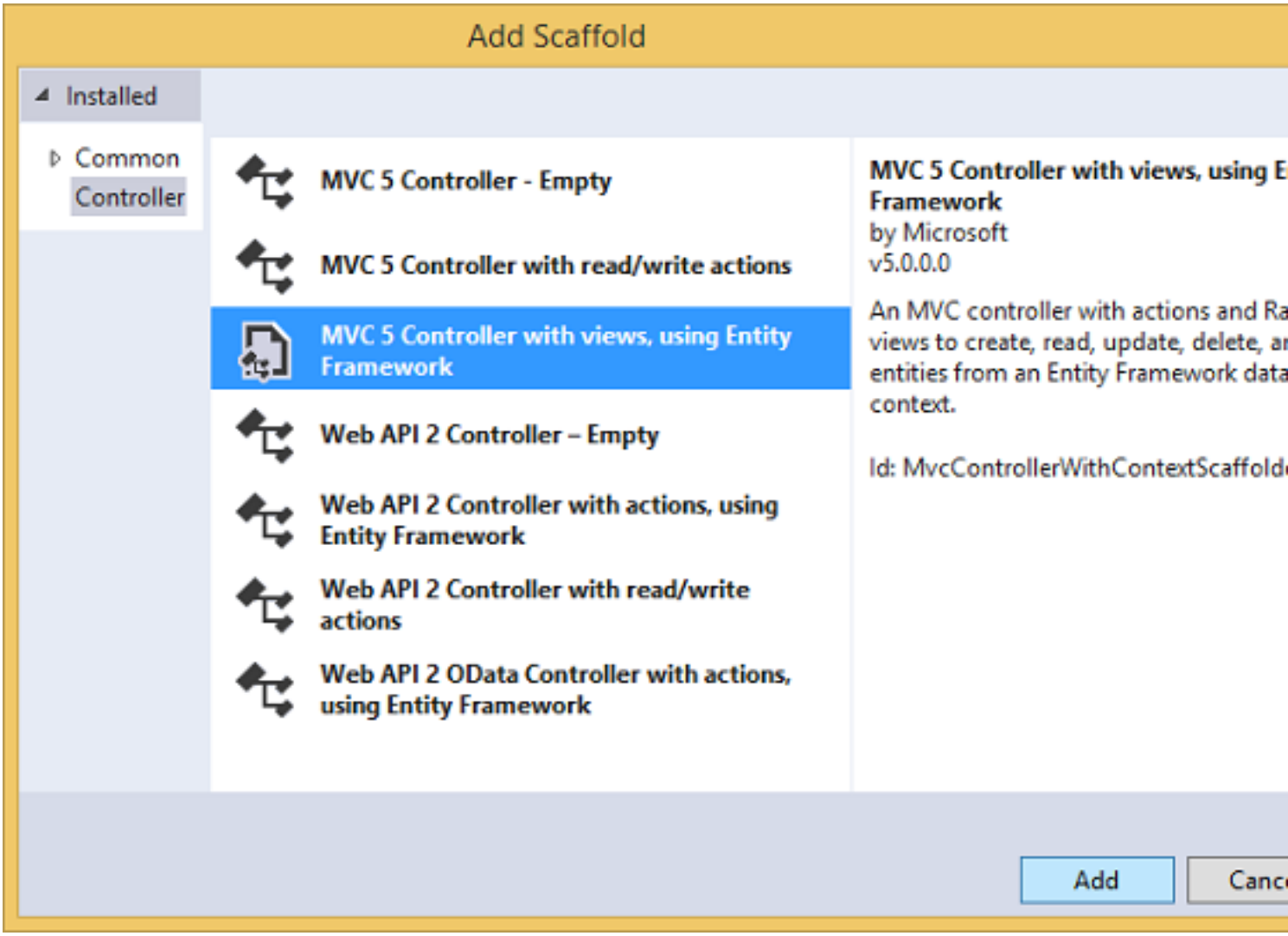
یک کنترلر و نما برای داده‌ها اضافه کنید

ابتدا پروژه را Build کنید (Ctrl + Shift+ B). این کار را باید پیش از استفاده از مکانیزم Scaffolding انجام دهید.

یک کنترلر جدید به پوشه Controllers اضافه کنید.



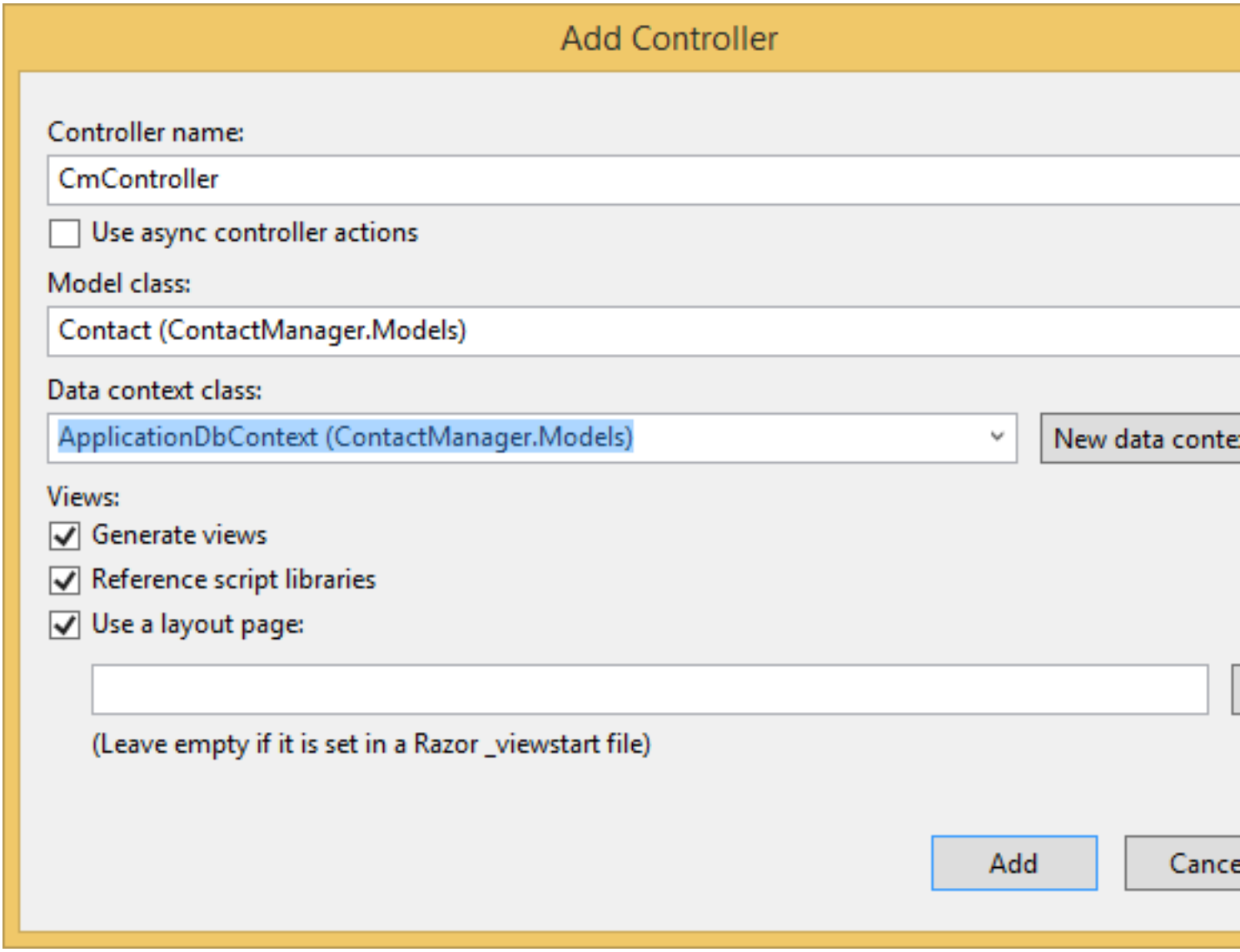
در دیالوگ باکس **Add Scaffold** گزینه **MVC 5 Controller with views, using EF** را انتخاب کنید.



در دیالوگ **Add Controller** نام "CmController" را برای کنترلر وارد کنید. (تصویر زیر.)

در لیست **Model** گزینه **Contact (ContactManager.Models)** را انتخاب کنید.

در قسمت **Data context class** گزینه **ApplicationDbContext (ContactManager.Models)** را انتخاب کنید. این **ApplicationDbContext** هم برای اطلاعات سیستم عضویت و هم برای داده‌های Contacts استفاده خواهد شد.

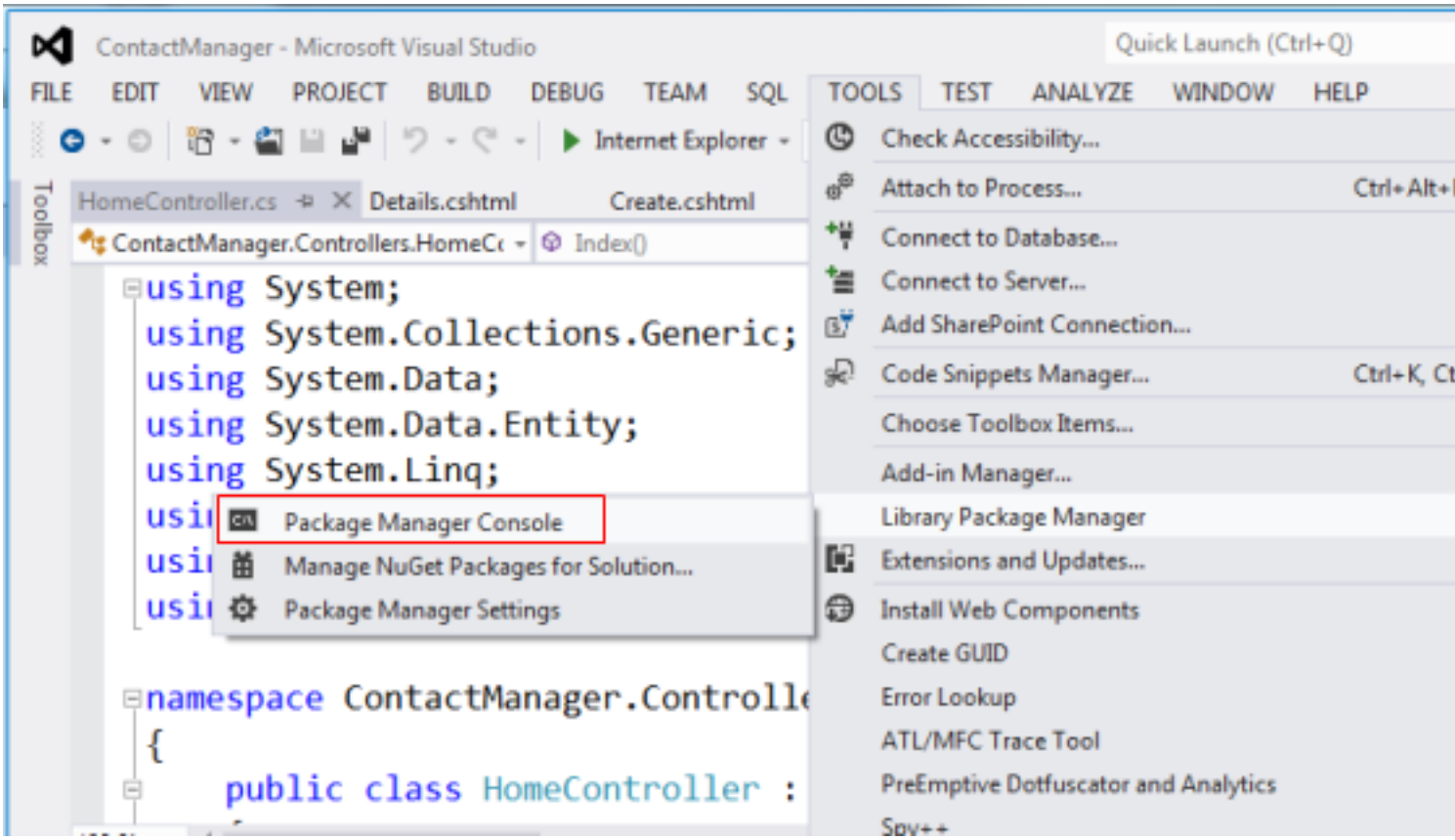


روی Add کلیک کنید. ویژوال استودیو بصورت خودکار با استفاده از Scaffolding متدها و Viewهای لازم برای عملیات CRUD را فراهم می‌کند، که همگی از مدل Contact استفاده می‌کنند.

فعالسازی مهاجرت ها، ایجاد دیتابیس، افزودن داده نمونه و یک راه انداز

مرحله بعدی فعال کردن قابلیت Code First Migrations است تا دیتابیس را بر اساس الگویی که تعریف کرده اید بسازد.

از منوی **Tools** گزینه **Library Package Manager** و سپس **Package Manager Console** را انتخاب کنید.



در پنجره باز شده فرمان زیر را وارد کنید.

```
enable-migrations
```

فرمان **enable-migrations** یک پوشه با نام *Migrations* می سازد و فایلی با نام *Configuration.cs* را به آن اضافه می‌کند. با استفاده از این کلاس می‌توانید داده‌های اولیه دیتابیس را وارد کنید و مهاجرت‌ها را نیز پیکربندی کنید.

در پنجره **Package Manager Console** فرمان زیر را وارد کنید.

```
add-migration Initial
```

فرمان **add-migration initial** فایلی با نام **<data_stamp> initial** ساخته و آن را در پوشه *Migrations* ذخیره می‌کند. در این مرحله دیتابیس شما ایجاد می‌شود. در این فرمان، مقدار **initial** اختیاری است و صرفا برای نامگذاری فایل مهاجرت استفاده شده. فایل‌های جدید را می‌توانید در **Solution Explorer** مشاهده کنید.

در کلاس **Initial** متد **Up** جدول Contacts را می‌سازد. و متد **Down** (هنگامی که می‌خواهید به وضعیت قبلی بازگردید) آن را drop می‌کند.

حال فایل *Migrations/Configuration.cs* را باز کنید. فضای نام زیر را اضافه کنید.

```
using ContactManager.Models;
```

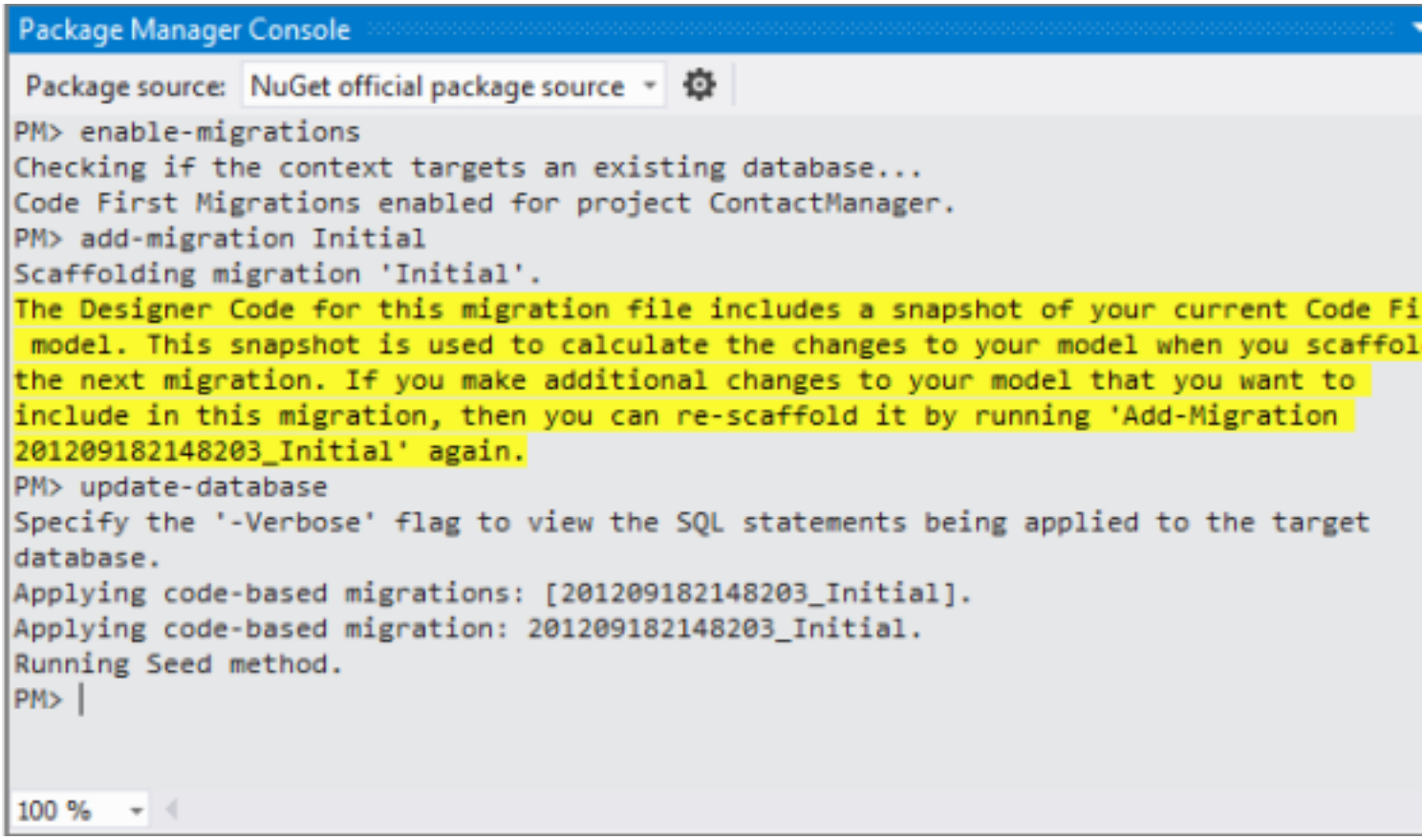
حال متد *Seed* را با قطعه کد زیر جایگزین کنید.

```
protected override void Seed(ContactManager.Models.ApplicationDbContext context)
{
    context.Contacts.AddOrUpdate(p => p.Name,
       new Contact
       {
           Name = "Debra Garcia",
           Address = "1234 Main St",
           City = "Redmond",
           State = "WA",
           Zip = "10999",
           Email = "debra@example.com",
       },
        new Contact
        {
            Name = "Thorsten Weinrich",
            Address = "5678 1st Ave W",
            City = "Redmond",
            State = "WA",
            Zip = "10999",
            Email = "thorsten@example.com",
        },
        new Contact
        {
            Name = "Yuhong Li",
            Address = "9012 State st",
            City = "Redmond",
            State = "WA",
            Zip = "10999",
            Email = "yuhong@example.com",
        },
        new Contact
        {
            Name = "Jon Orton",
            Address = "3456 Maple St",
            City = "Redmond",
            State = "WA",
            Zip = "10999",
            Email = "jon@example.com",
        },
        new Contact
        {
            Name = "Diliana Alexieva-Bosseva",
            Address = "7890 2nd Ave E",
            City = "Redmond",
            State = "WA",
            Zip = "10999",
            Email = "diliana@example.com",
        }
        );
}
```

این متد دیتابیس را Seed می‌کند، یعنی داده‌های پیش فرض و اولیه دیتابیس را تعریف می‌کند. برای اطلاعات بیشتر به Seeding and Debugging Entity Framework (EF) DBs مراجعه کنید.

در پنجره **Package Manager Console** فرمان زیر را وارد کنید.

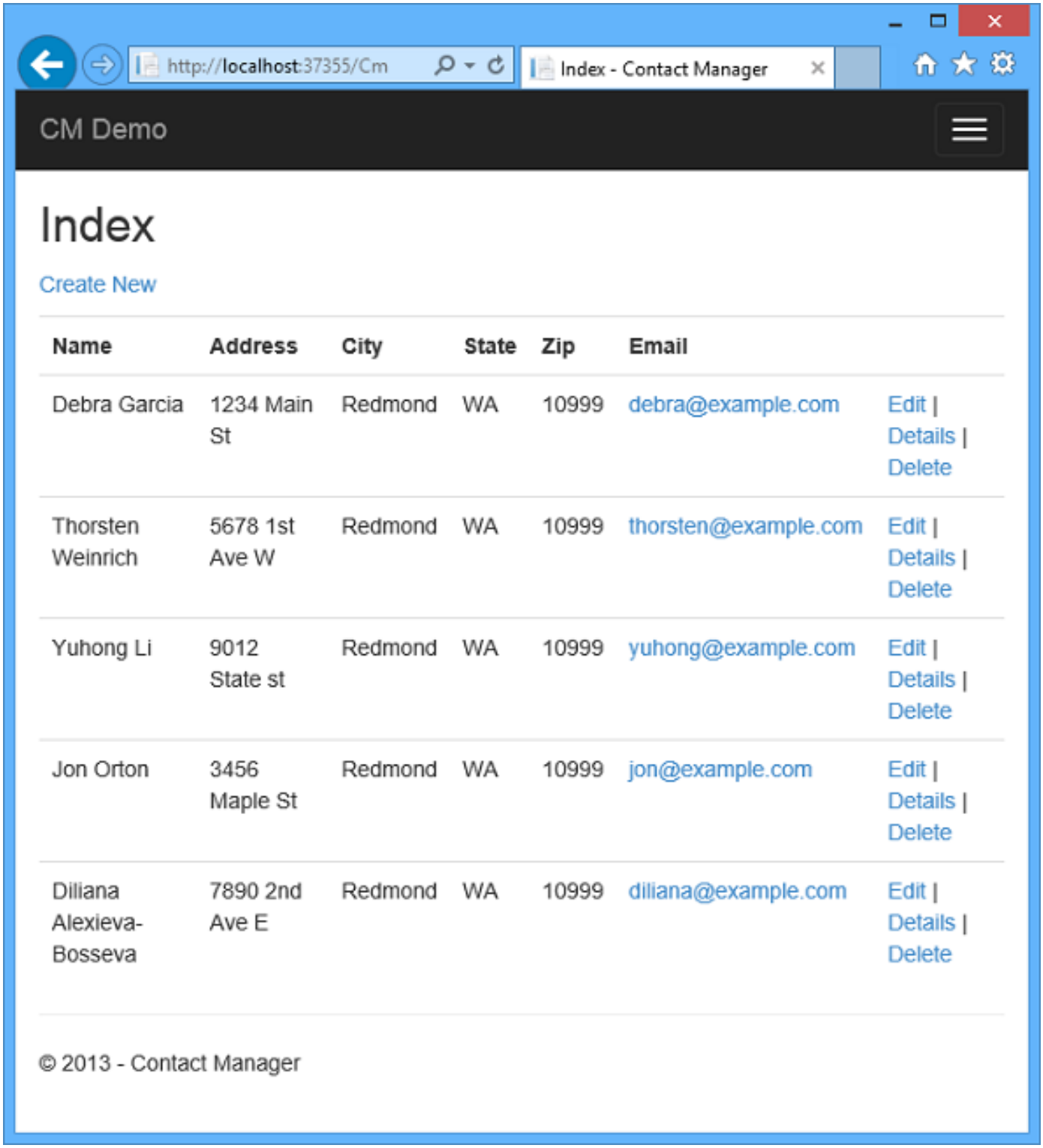
```
update-database
```



فرمان **update-database** مهاجرت نخست را اجرا می‌کند، که دیتابیس را می‌سازد. بصورت پیش فرض این یک دیتابیس SQL Server Express LocalDB است.

حال پروژه را با CTRL + F5 اجرا کنید.

همانطور که مشاهده می‌کنید، اپلیکیشن داده‌های اولیه (Seed) را نمایش می‌دهد، و لینک هایی هم برای ویرایش، حذف و مشاهده جزئیات رکوردها فراهم می‌کند. می‌توانید داده‌ها را مشاهده کنید، رکورد جدید ثبت کنید و یا داده‌های قبلی را ویرایش و حذف کنید.



یک تامین کننده OAuth2 و OpenID اضافه کنید OAuth یک پروتکل باز است که امکان authorization امن توسط یک متد استاندارد را فراهم می‌کند. این پروتکل می‌تواند در اپلیکیشن‌های وب، موبایل و دسکتاپ استفاده شود. قالب پروژه ASP.NET MVC internet از OAuth و OpenID استفاده می‌کند تا فیسبوک، توییتر، گوگل و حساب‌های کاربری مایکروسافت را بعنوان تامین کنندگان خارجی تعریف کند. به سادگی می‌توانید قطعه کدی را ویرایش کنید و از تامین کننده احراز هویت مورد نظرتان استفاده کنید.
مراحلی که برای اضافه کردن این تامین کنندگان باید دنبال کنید، بسیار مشابه همین مراحلی است که در این مقاله دنبال خواهید کرد. برای اطلاعات بیشتر درباره نحوه استفاده از فیسبوک بعنوان یک تامین کننده احراز هویت به Create an ASP.NET MVC 5 App with Facebook and Google OAuth2 and OpenID Sign-on مراجعه کنید.

علاوه بر احراز هویت، اپلیکیشن ما از نقش‌ها (roles) نیز استفاده خواهد کرد تا از authorization پشتیبانی کند. تنها کاربرانی که به نقش *canEdit* تعلق داشته باشند قادر به ویرایش اطلاعات خواهند بود (یعنی ایجاد، ویرایش و حذف رکورد ها).

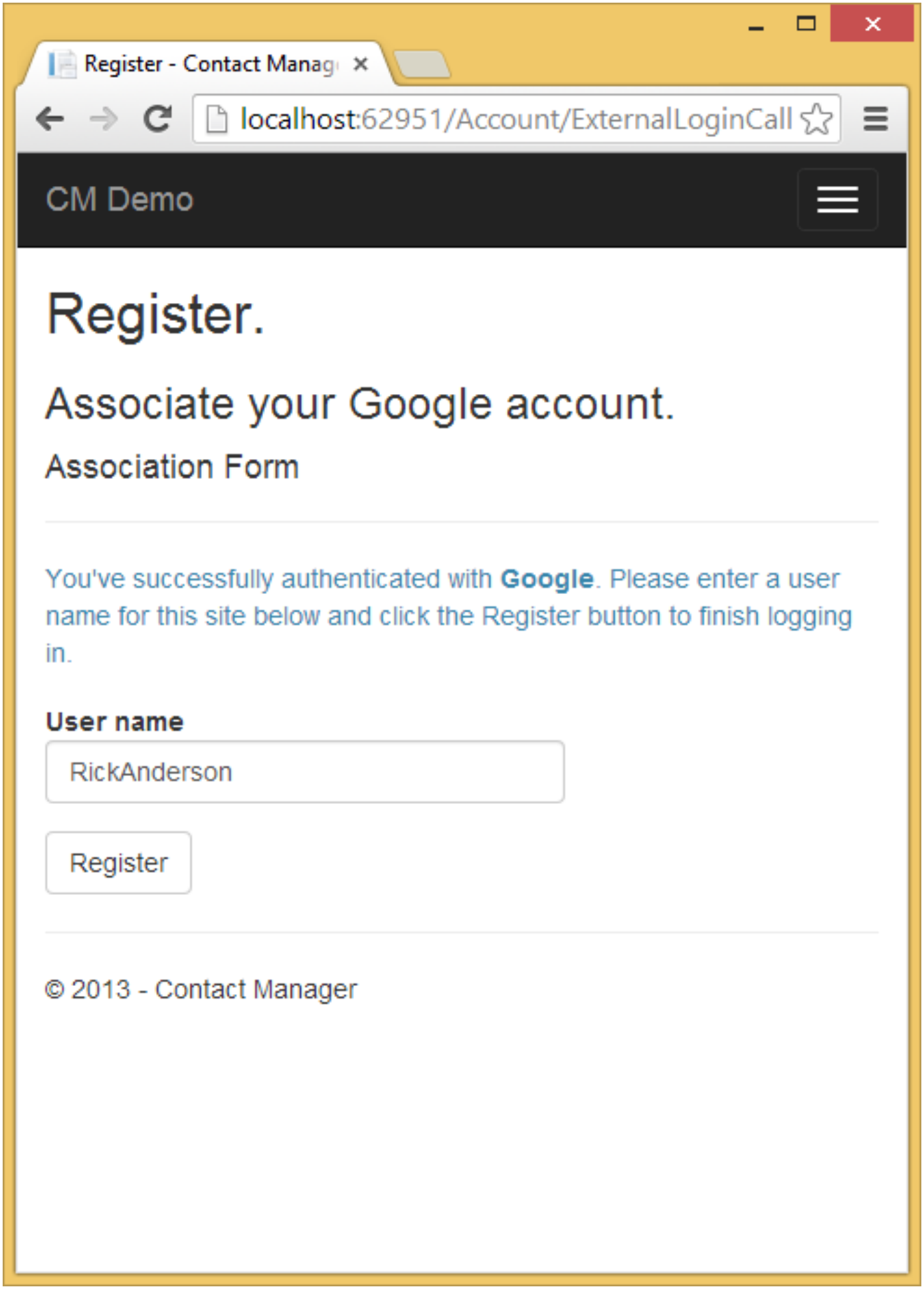
فایل *App_Start/Startup.Auth.cs* را باز کنید. توضیحات متد *app.UseGoogleAuthentication* را حذف کنید.

حال اپلیکیشن را اجرا کنید و روی لینک **Log In** کلیک کنید.

زیر قسمت **User another service to log in** روی دکمه **Google** کلیک کنید. اطلاعات کاربری خود را وارد کنید. سپس **Accept** را کلیک کنید تا به اپلیکیشن خود دسترسی کافی بدهید (برای آدرس ایمیل و اطلاعات پایه).

حال باید به صفحه ثبت نام (Register) هدایت شوید. در این مرحله می‌توانید در صورت لزوم نام کاربری خود را تغییر دهید.

نهایتا روی **Register** کلیک کنید.


Register - Contact Manag
localhost:62951/Account/ExternalLoginCall
CM Demo
Register.
Associate your Google account.
Association Form
You've successfully authenticated with Google. Please enter a user name for this site below and click the Register button to finish logging in.
User name
RickAnderson
Register
© 2013 - Contact Manager

استفاده از Membership API

در این قسمت شما یک کاربر محلی و نقش *canEdit* را به دیتابیس عضویت اضافه می‌کنید. تنها کاربرانی که به این نقش تعلق دارند قادر به ویرایش داده‌ها خواهند بود. یکی از بهترین تمرین‌ها (best practice) نام گذاری نقش‌ها بر اساس عملیاتی است که می‌توانند اجرا کنند. بنابراین مثلا *canEdit* نسبت به نقشی با نام *admin* ترجیح داده می‌شود. هنگامی که اپلیکیشن شما رشد می‌کند و بزرگتر می‌شود، شما می‌توانید نقش‌های جدیدی مانند *canDeleteMembers* اضافه کنید، بجای آنکه از نام‌های گنگی مانند *superAdmin* استفاده کنید.

فایل *Migrations/Configuration.cs* را باز کنید و عبارات زیر را به آن اضافه کنید.

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

متد **AddUserAndRole** را به این کلاس اضافه کنید.

```
bool AddUserAndRole(ContactManager.Models.ApplicationDbContext context)
 {
    IdentityResult ir;
    var rm = new RoleManager<IdentityRole>
        (new RoleStore<IdentityRole>(context));
    ir = rm.Create(new IdentityRole("canEdit"));
    var um = new UserManager<ApplicationUser>(
        new UserStore<ApplicationUser>(context));
    var user = new ApplicationUser()
    {
       UserName = "user1",
    };
    ir = um.Create(user, "Passw0rd1");
    if (ir.Succeeded == false)
       return ir.Succeeded;
    ir = um.AddToRole(user.Id, "canEdit");
    return ir.Succeeded;
 }
```

حالا از متد **Seed** این متد جدید را فراخوانی کنید.

```
protected override void Seed(ContactManager.Models.ApplicationDbContext context)
{
    AddUserAndRole(context);
    context.Contacts.AddOrUpdate(p => p.Name,
        // Code removed for brevity
}
```

این کدها نقش جدیدی با نام *canEdit* و کاربری با نام *user1* می سازد. سپس این کاربر به نقش مذکور اضافه می‌شود.

کدی موقتی برای تخصیص نقش *canEdit* به کاربران جدید Social Provider ها

در این قسمت شما متد **ExternalLoginConfirmation** در کنترلر Account را ویرایش خواهید کرد. با این تغییرات، کاربران جدیدی که توسط OAuth یا OpenID ثبت نام می‌کنند به نقش *canEdit* اضافه می‌شوند. تا زمانی که ابزاری برای افزودن و مدیریت نقش‌ها بسازیم، از این کد موقتی استفاده خواهیم کرد. تیم مایکروسافت امیدوار است ابزاری مانند WSAT برای مدیریت کاربران و نقش‌ها در آینده عرضه کند. بعدا در این مقاله با اضافه کردن کاربران به نقش‌ها بصورت دستی از طریق **Server Explorer** نیز آشنا خواهید شد.

فایل **Controllers/AccountController.cs** را باز کنید و متد **ExternalLoginConfirmation** را پیدا کنید.

درست قبل از فراخوانی **SignInAsync** متد **AddToRoleAsync** را فراخوانی کنید.

```
await UserManager.AddToRoleAsync(user.Id, "CanEdit");
```

کد بالا کاربر ایجاد شده جدید را به نقش canEdit اضافه می‌کند، که به آنها دسترسی به متدهای ویرایش داده را می‌دهد. تصویری

از تغییرات کد در زیر آمده است.

```
//
// POST: /Account/ExternalLoginConfirmation
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> ExternalLoginConfirmation(ExternalLoginConfirmati
{
    if (User.Identity.IsAuthenticated)
    {
        return RedirectToAction("Manage");
    }

    if (ModelState.IsValid)
    {
        // Get the information about the user from the external login provider
        var info = await AuthenticationManager.GetExternalLoginInfoAsync();
        if (info == null)
        {
            return View("ExternalLoginFailure");
        }
        var user = new ApplicationUser() { UserName = model.UserName };
        var result = await UserManager.CreateAsync(user);
        if (result.Succeeded)
        {
            result = await UserManager.AddLoginAsync(user.Id, info.Login);
            if (result.Succeeded)
            {
                await UserManager.AddToRoleAsync(user.Id, "CanEdit");
                await SignInAsync(user, isPersistent: false);
                return RedirectToLocal(returnUrl);
            }
        }
        AddErrors(result);
    }

    ViewBag.ReturnUrl = returnUrl;
    return View(model);
}
```

در ادامه مقاله اپلیکیشن خود را روی Windows Azure منتشر خواهید کرد و با استفاده از Google و تامین کنندگان دیگر وارد سایت می‌شوید. هر فردی که به آدرس سایت شما دسترسی داشته باشد، و یک حساب کاربری Google هم در اختیار داشته باشد

می‌تواند در سایت شما ثبت نام کند و سپس دیتابیس را ویرایش کند. برای جلوگیری از دسترسی دیگران، می‌توانید وب سایت خود را متوقف (stop) کنید.

در پنجره **Package Manager Console** فرمان زیر را وارد کنید.

```
Update-Database
```

فرمان را اجرا کنید تا متد **Seed** را فراخوانی کند. حال **AddUserAndRole** شما نیز اجرا می‌شود. تا این مرحله نقش *canEdit* ساخته شده و کاربر جدیدی با نام *user1* ایجاد و به آن افزوده شده است.

محافظت از اپلیکیشن توسط SSL و خاصیت Authorize

در این قسمت شما با استفاده از خاصیت Authorize دسترسی به اکشن متدها را محدود می‌کنید. کاربران ناشناس (Anonymous) تنها قادر به مشاهده متد **Index** در کنترلر **home** خواهند بود. کاربرانی که ثبت نام کرده اند به متدهای **Index** و **Details** در کنترلر **Cm** و صفحات About و Contact نیز دسترسی خواهند داشت. همچنین دسترسی به متدهایی که داده‌ها را تغییر می‌دهند تنها برای کاربرانی وجود دارد که در نقش *canEdit* هستند.

خاصیت Authorize و RequireHttps را به اپلیکیشن اضافه کنید. یک راه دیگر افزودن این خاصیت‌ها به تمام کنترلرها است، اما تجارب امنیتی توصیه می‌کند که این خاصیت‌ها روی کل اپلیکیشن اعمال شوند. با افزودن این خاصیت‌ها بصورت global تمام کنترلرها و اکشن متدهایی که می‌سازید بصورت خودکار محافظت خواهند شد، و دیگر لازم نیست بیاد داشته باشید کدام کنترلرها و متدها را باید ایمن کنید.

برای اطلاعات بیشتر به Securing your ASP.NET MVC App and the new AllowAnonymous Attribute مراجعه کنید.

فایل *App_Start/FilterConfig.cs* را باز کنید و متد *RegisterGlobalFilters* را با کد زیر مطابقت دهید.

```
public static void
RegisterGlobalFilters(GlobalFilterCollection filters)
{
    filters.Add(new HandleErrorAttribute());
    filters.Add(new System.Web.Mvc.AuthorizeAttribute());
    filters.Add(new RequireHttpsAttribute());
}
```

خاصیت Authorize در کد بالا از دسترسی کاربران ناشناس به تمام متدهای اپلیکیشن جلوگیری می‌کند. شما برای اعطای دسترسی به متدهایی خاص از خاصیت AllowAnonymous استفاده خواهید کرد. در آخر خاصیت RequireHTTPS باعث می‌شود تا تمام دسترسی‌ها به اپلیکیشن وب شما از طریق HTTPS صورت گیرد.

حالا خاصیت AllowAnonymous را به متد **Index** در کنترلر Home اضافه کنید. از این خاصیت برای اعطای دسترسی به تمامی کاربران سایت استفاده کنید. قسمتی از کد کنترلر Home را در زیر می‌بینید.

```
namespace ContactManager.Controllers
 {
    public class HomeController : Controller
    {
       [AllowAnonymous]
       public ActionResult Index()
       {
          return View();
       }
```

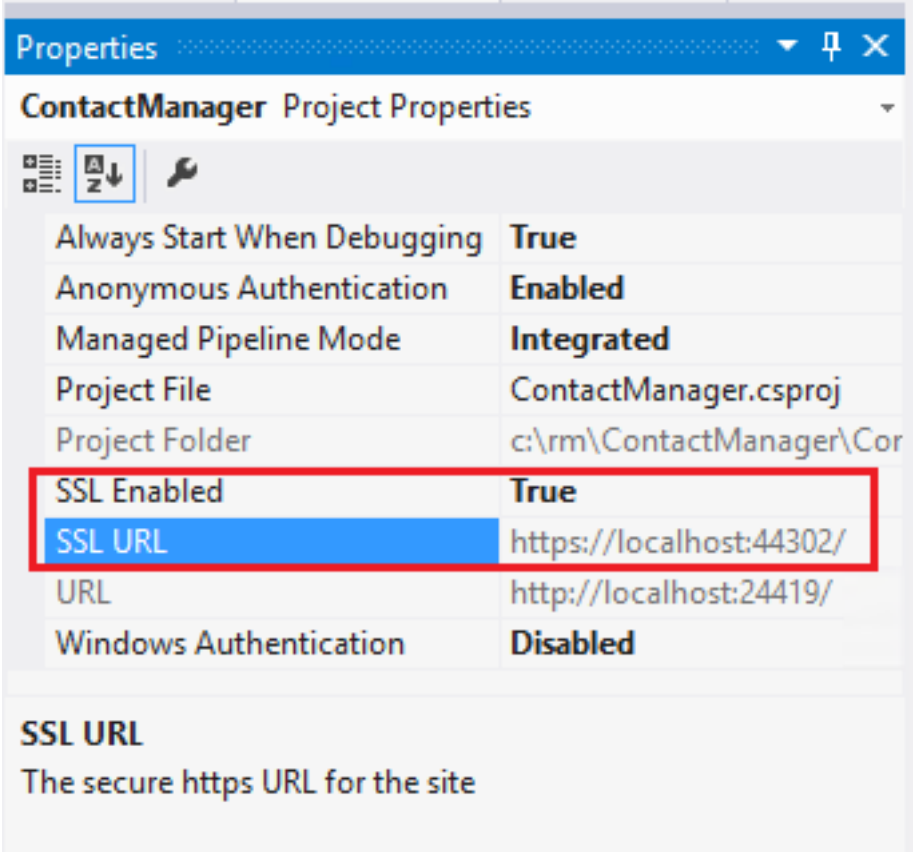
یک جستجوی عمومی برای عبارت *AllowAnonymous* انجام دهید. همانطور که مشاهده می‌کنید این خاصیت توسط متدهای ورود و ثبت نام در کنترلر Account نیز استفاده شده است.

در کنترلر *CmController* خاصیت *[Authorize(Roles="canEdit")]* را به تمام متدهایی که با داده سر و کار دارند اضافه کنید، به غیر از متدهای Index و Details. قسمتی از کد کامل شده در زیر آمده است.

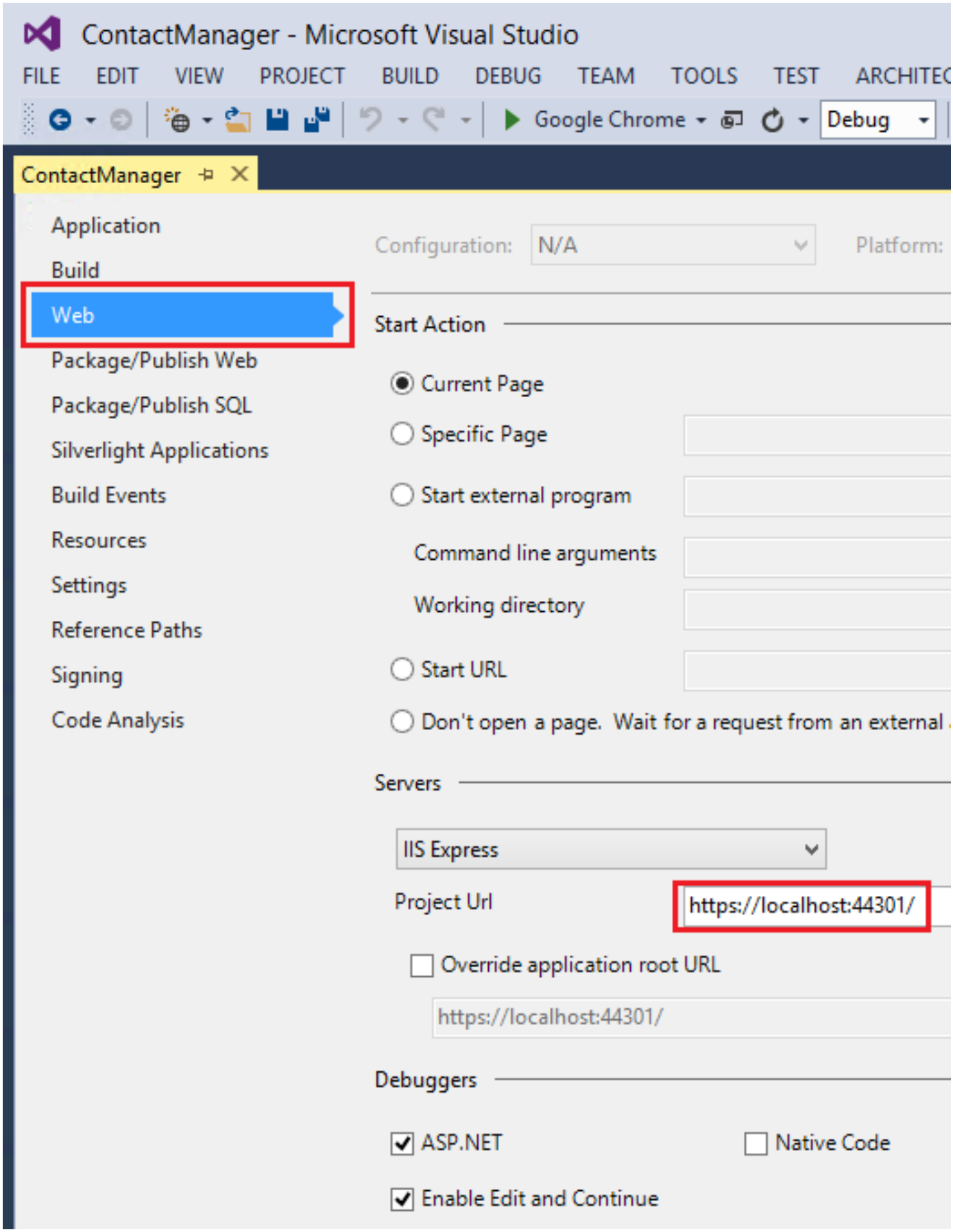
```
public class CmController : Controller
{
    private ContactManagerContext db = new ContactManagerContext();

    // GET: /Cm/Create

    [Authorize(Roles = "canEdit")]
    public ActionResult Create()
    {
        return View();
    }

    // POST: /Cm/Create
    [HttpPost]
    [ValidateAntiForgeryToken]
    [Authorize(Roles = "canEdit")]
    public ActionResult Create([Bind(Include = "ContactId,Name,Address,City,
    {
        if (ModelState.IsValid)
        {
            db.Contacts.Add(contact);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        return View(contact);
    }

    // GET: /Cm/Edit/5
    [Authorize(Roles = "canEdit")]
    public ActionResult Edit(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        Contact contact = db.Contacts.Find(id);
        if (contact == null)
        {
            return HttpNotFound();
        }
        return View(contact);
    }
```

فعال سازی SSL برای پروژه

در Solution Explorer پروژه خود را انتخاب کنید. سپس کلید F4 را فشار دهید تا دیالوگ خواص (Properties) باز شود. حال مقدار خاصیت **SSL Enabled** را به true تنظیم کنید. آدرس **SSL URL** را کپی کنید. این آدرس چیزی شبیه به https://localhost:44300/ خواهد بود.

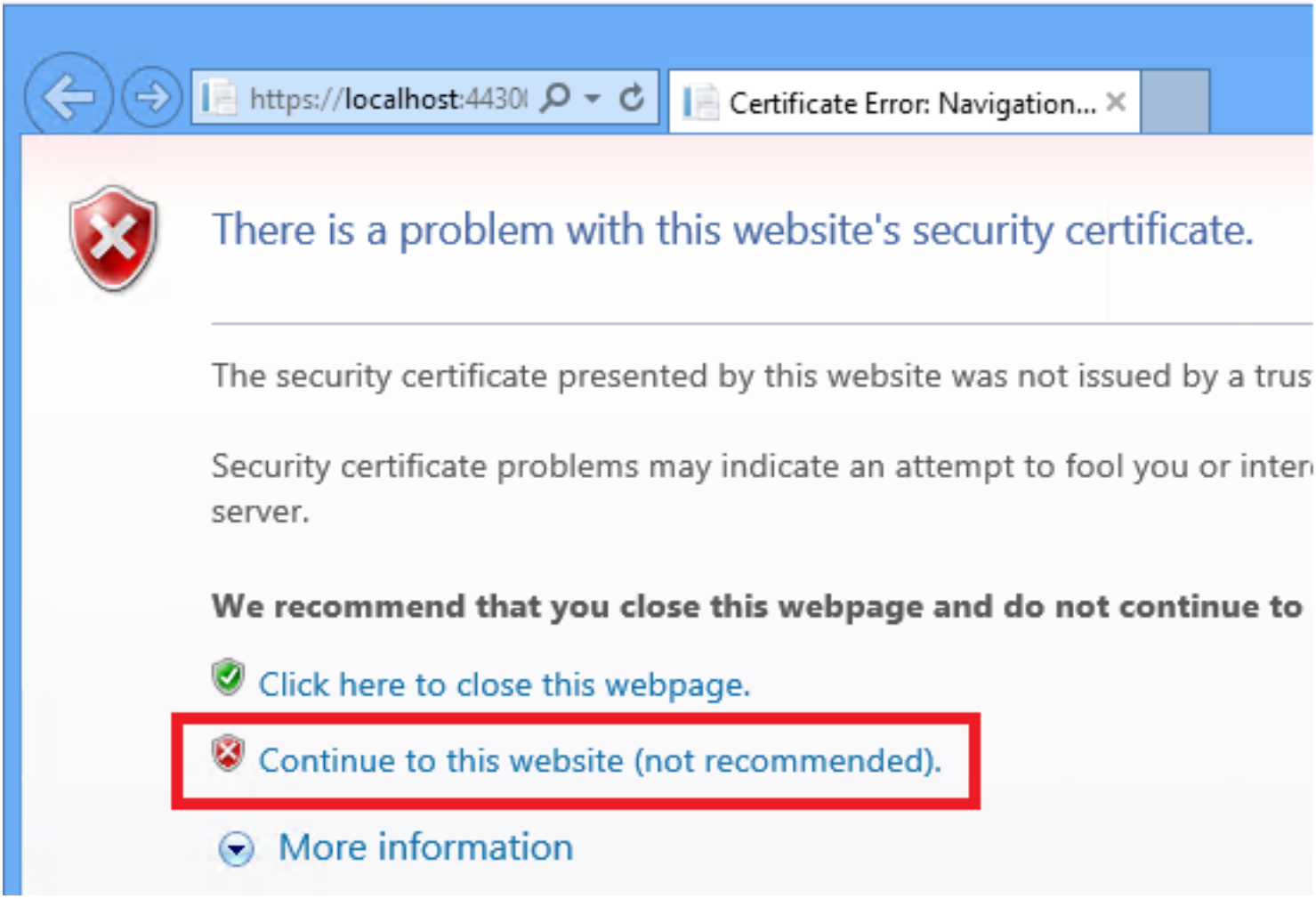


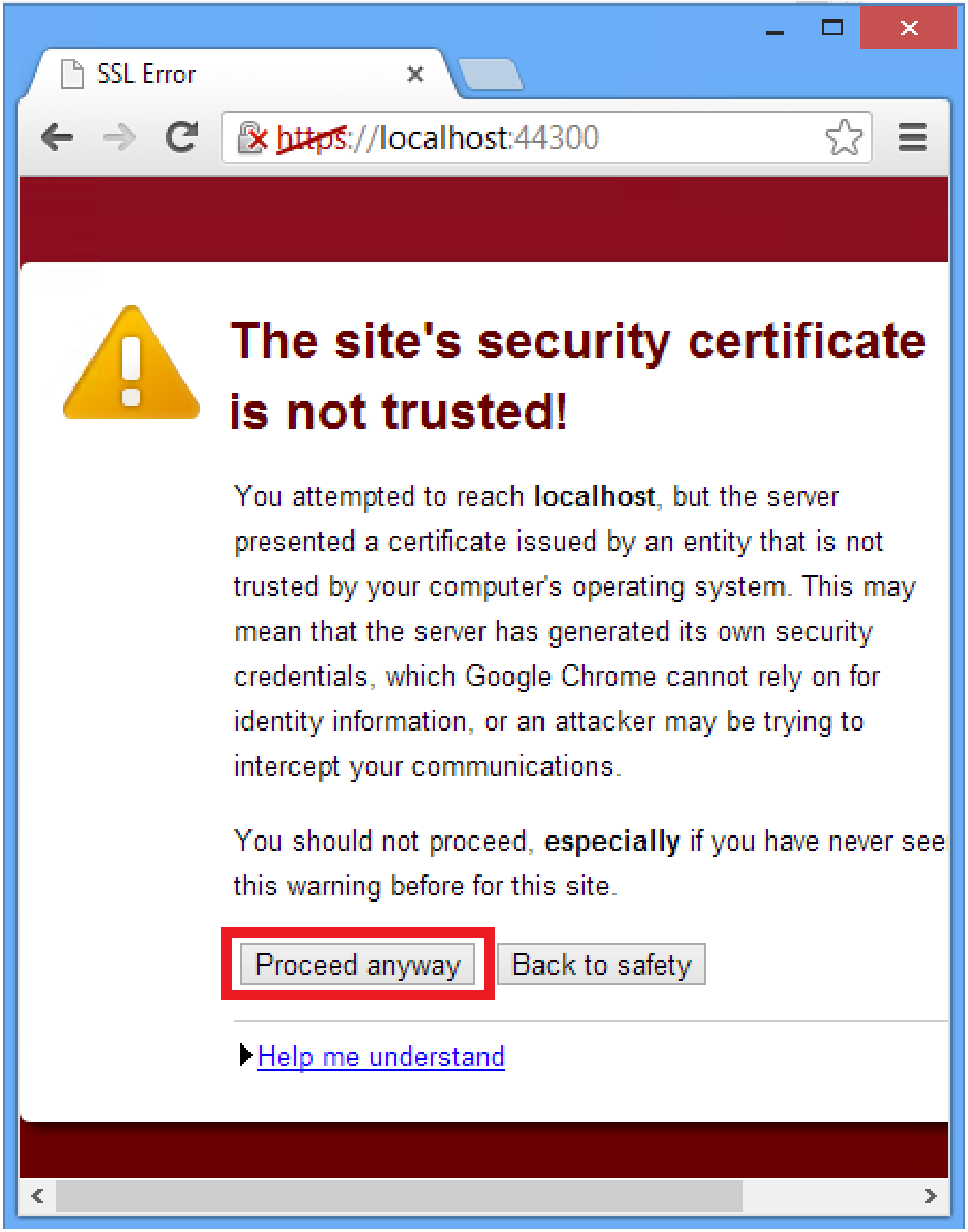
روی نام پروژه کلیک راست کنید و **Properties** را انتخاب کنید. در قسمت چپ گزینه **Web** را انتخاب کنید. حالا مقدار **Project Url** را به آدرسی که کپی کرده اید تغییر دهید. نهایتا تغییرات را ذخیره کنید و پنجره را ببندید.



حال پروژه را اجرا کنید. مرورگر شما باید یک پیام خطای اعتبارسنجی به شما بدهد. دلیلش این است که اپلیکیشن شما از یک

Valid Certificate استفاده نمی‌کند. هنگامی که پروژه را روی Windows Azure منتشر کنید دیگر این پیغام را نخواهید دید. چرا که سرورهای مایکروسافت همگی لایسنس‌های معتبری دارند. برای اپلیکیشن ما می‌توانید روی **Continue to this website** را انتخاب کنید.





حال مرورگر پیش فرض شما باید صفحه **Index** از کنترلر home را به شما نمایش دهد.

اگر از یک نشست قبلی هنوز در سایت هستید (logged-in) روی لینک **Log out** کلیک کنید و از سایت خارج شوید.

روی لینک‌های **About** و **Contact** کلیک کنید. باید به صفحه ورود به سایت هدایت شوید چرا که کاربران ناشناس اجازه دسترسی به این صفحات را ندارند.

روی لینک **Register** کلیک کنید و یک کاربر محلی با نام *Joe* بسازید. حال مطمئن شوید که این کاربر به صفحات Home, About و Contact دسترسی دارد.

روی لینک *CM Demo* کلیک کنید و مطمئن شوید که داده‌ها را مشاهده می‌کنید.

حال روی یکی از لینک‌های ویرایش (Edit) کلیک کنید. این درخواست باید شما را به صفحه ورود به سایت هدایت کند، چرا که کاربران محلی جدید به نقش *canEdit* تعلق ندارند.

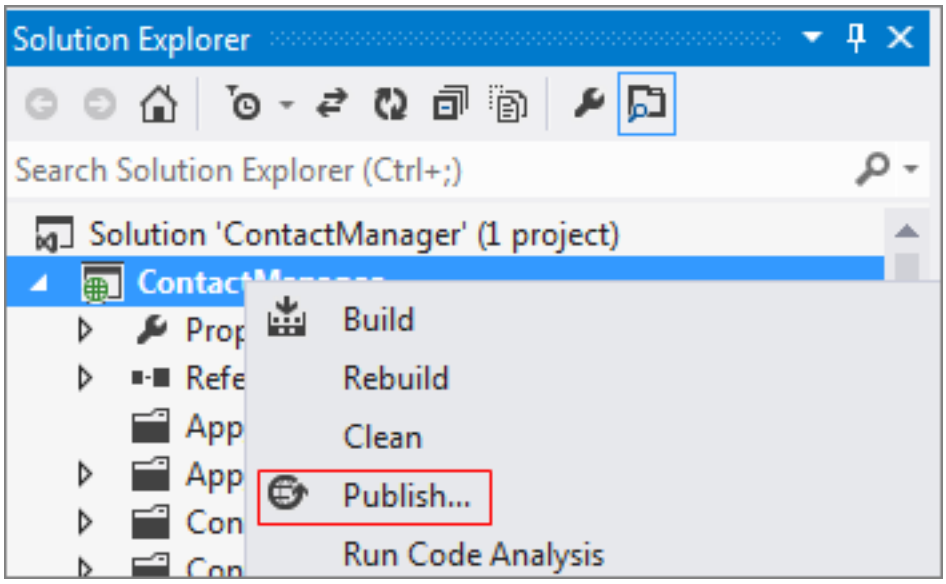
با کاربر *user1* که قبلا ساختید وارد سایت شوید. حال به صفحه ویرایشی که قبلا درخواست کرده بودید هدایت می‌شوید.

اگر نتوانستید با این کاربر به سایت وارد شوید، کلمه عبور را از سورس کد کپی کنید و مجددا امتحان کنید. اگر همچنان نتوانستید به سایت وارد شوید، جدول **AspNetUsers** را بررسی کنید تا مطمئن شوید کاربر *user1* ساخته شده است. این مراحل را در ادامه مقاله خواهید دید.

در آخر اطمینان حاصل کنید که می‌توانید داده‌ها را تغییر دهید.

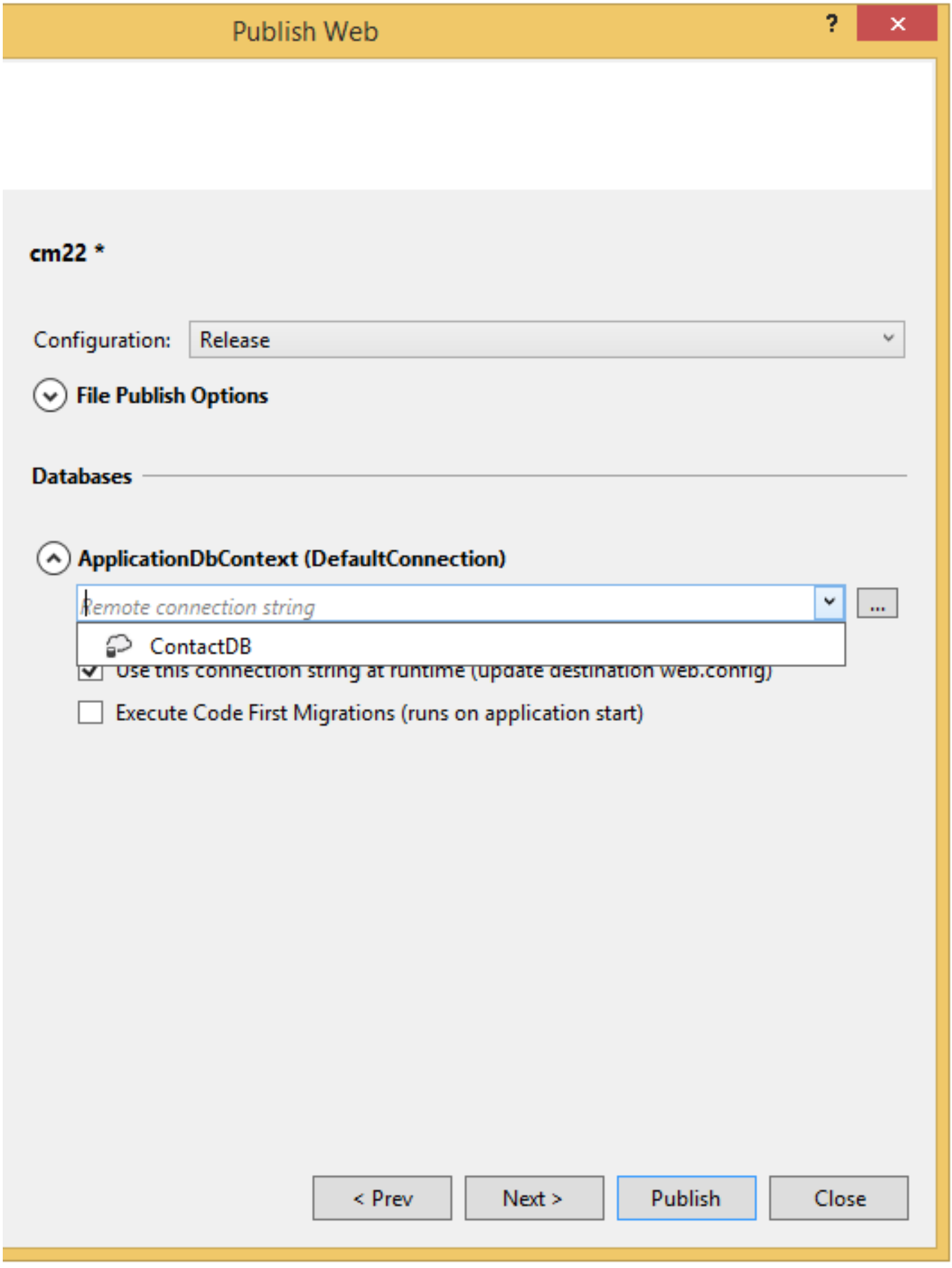
اپلیکیشن را روی Windows Azure منتشر کنید

ابتدا پروژه را Build کنید. سپس روی نام پروژه کلیک راست کرده و گزینه **Publish** را انتخاب کنید.

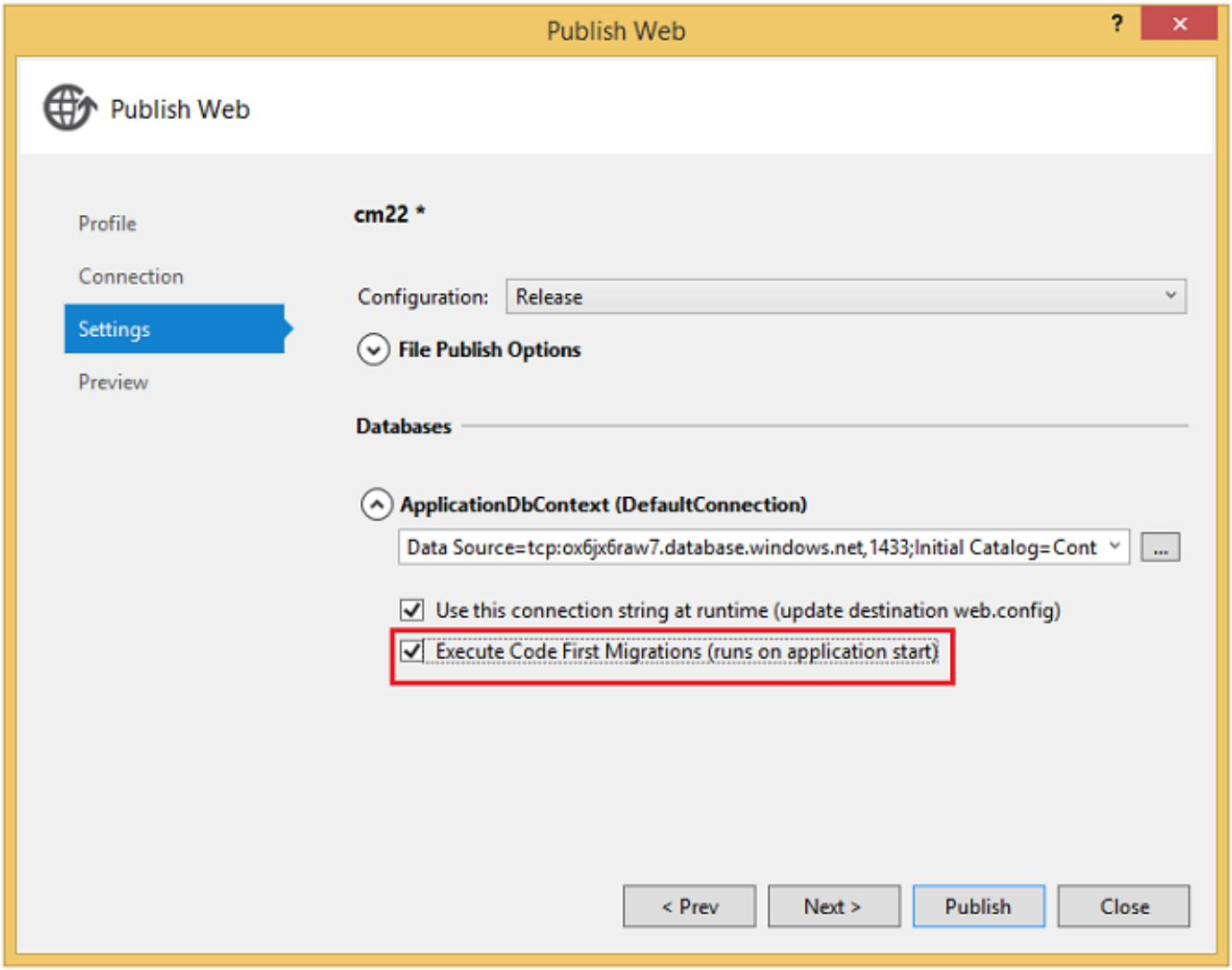


در دیالوگ باز شده روی قسمت **Settings** کلیک کنید. روی **File Publish Options** کلیک کنید تا بتوانید **Remote connection** را برای **ApplicationDbContext** و دیتابیس ContactDB انتخاب کنید.

اگر ویژوال استودیو را پس از ساخت Publish profile بسته و دوباره باز کرده اید، ممکن است رشته اتصال را در لیست موجود نبینید. در چنین صورتی، بجای ویرایش پروفایل انتشار، یک پروفایل جدید بسازید. درست مانند مراحلی که پیشتر دنبال کردید.


Publish Web
cm22 *
Configuration:
Release
File Publish Options
Databases
ApplicationDbContext (DefaultConnection)
Remote connection string
...
ContactDB
Use this connection string at runtime (update destination web.config)
Execute Code First Migrations (runs on application start)
< Prev
Next >
Publish
Close

زیر قسمت **ContactManagerContext** گزینه **Execute Code First Migrations** را انتخاب کنید.

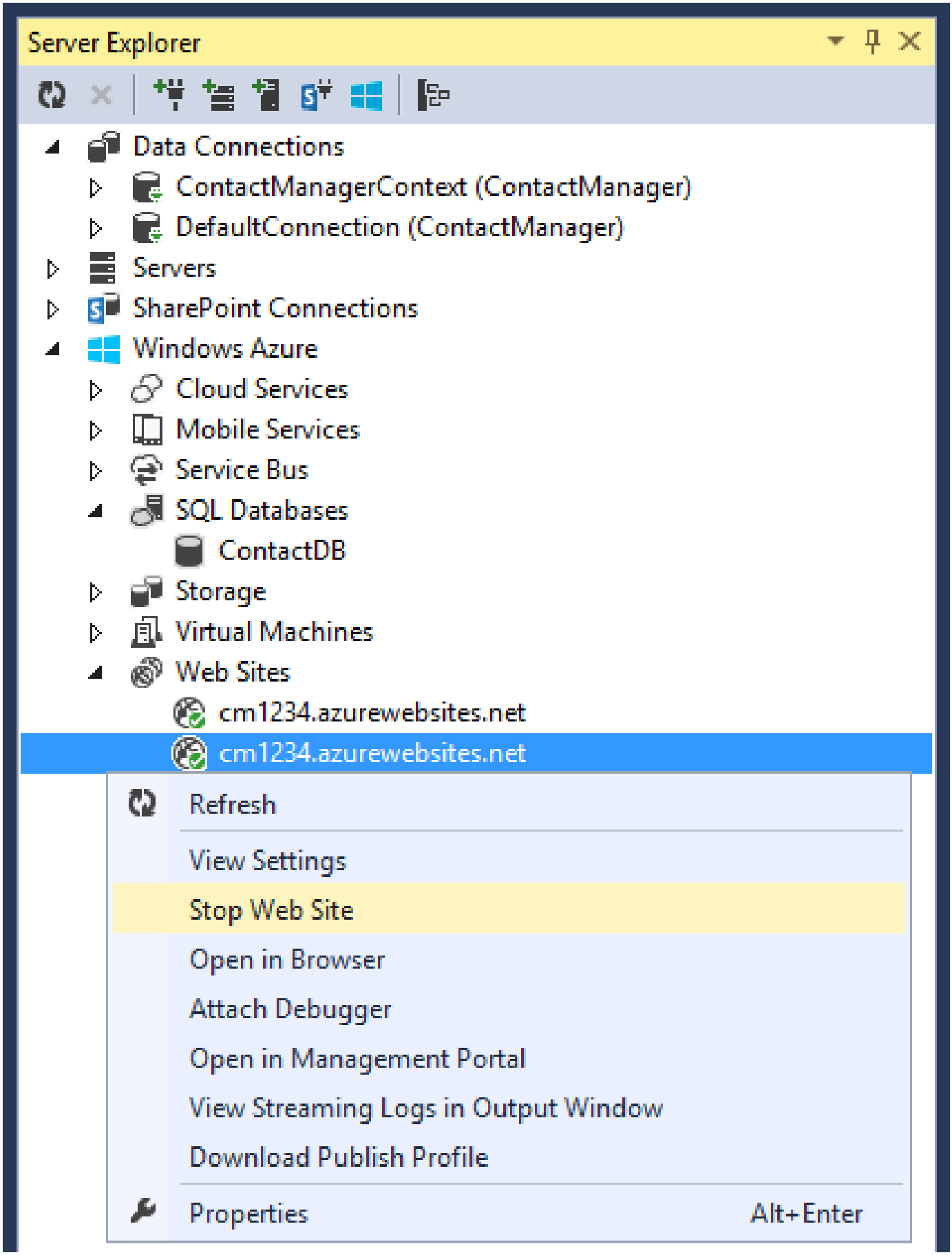


حال **Publish** را کلیک کنید تا اپلیکیشن شما منتشر شود. با کاربر *user1* وارد سایت شوید و بررسی کنید که می‌توانید داده‌ها را ویرایش کنید یا خیر.

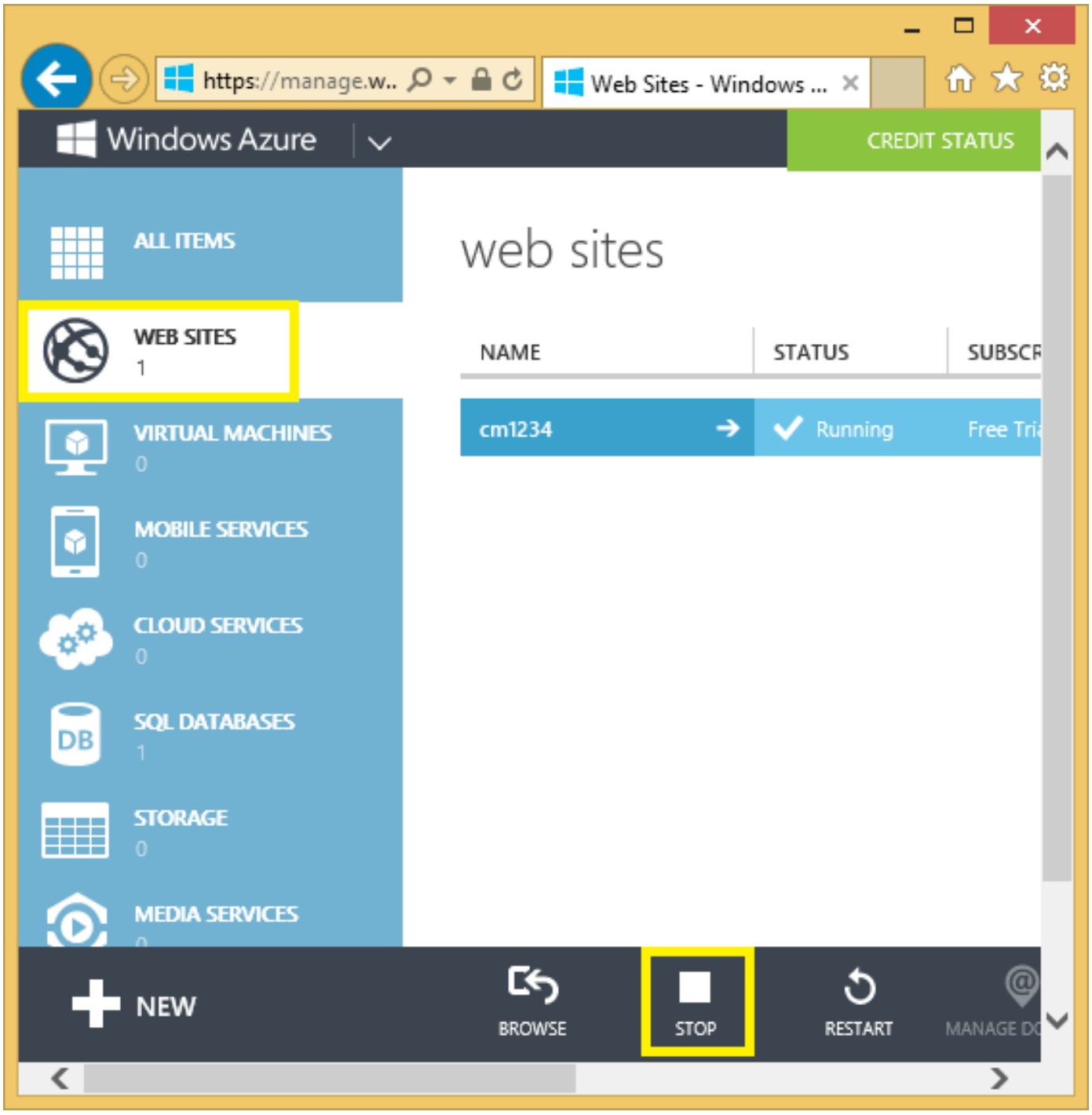
حال از سایت خارج شوید و توسط یک اکانت Google یا Facebook وارد سایت شوید، که در این صورت نقش canEdit نیز به شما تعلق می‌گیرد.

برای جلوگیری از دسترسی دیگران، وب سایت را متوقف کنید

در **Server Explorer** به قسمت **Web Sites** بروید. حال روی هر نمونه از وب سایت‌ها کلیک راست کنید و گزینه **Stop Web Site** را انتخاب کنید.


Server Explorer
Data Connections
ContactManagerContext (ContactManager)
DefaultConnection (ContactManager)
Servers
SharePoint Connections
Windows Azure
Cloud Services
Mobile Services
Service Bus
SQL Databases
ContactDB
Storage
Virtual Machines
Web Sites
cm1234.azurewebsites.net
cm1234.azurewebsites.net
Refresh
View Settings
Stop Web Site
Open in Browser
Attach Debugger
Open in Management Portal
View Streaming Logs in Output Window
Download Publish Profile
Properties
Alt+Enter

یک راه دیگر متوقف کردن وب سایت از طریق پرتال مدیریت Windows Azure است.

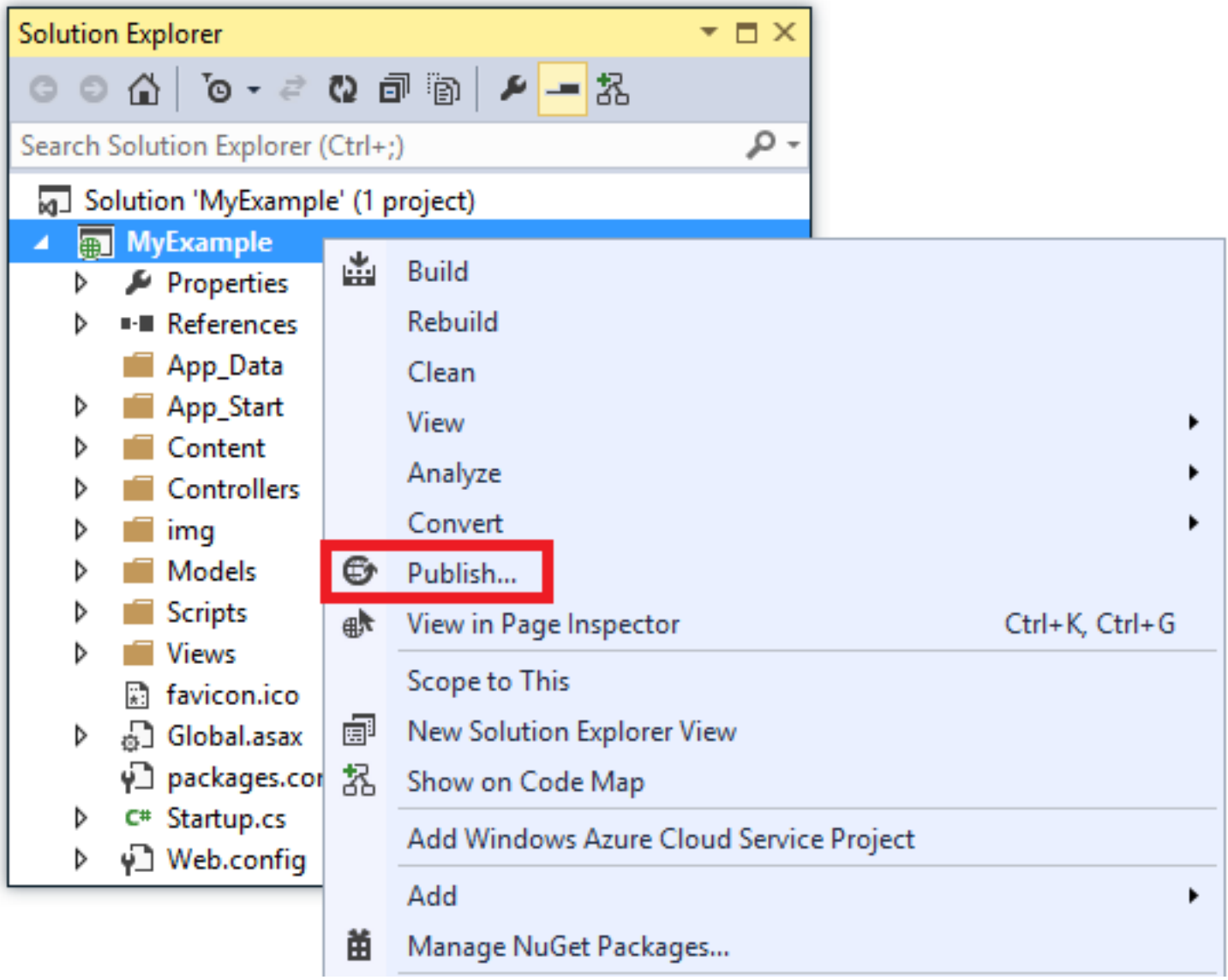


فراخوانی AddToRoleAsync را حذف و اپلیکیشن را منتشر و تست کنید

کنترلر Account را باز کنید و کد زیر را از متد **ExternalLoginConfirmation** حذف کنید.

```
await UserManager.AddToRoleAsync(user.Id, "CanEdit");
```

پروژه را ذخیره و Build کنید. حال روی نام پروژه کلیک راست کرده و **Publish** را انتخاب کنید.



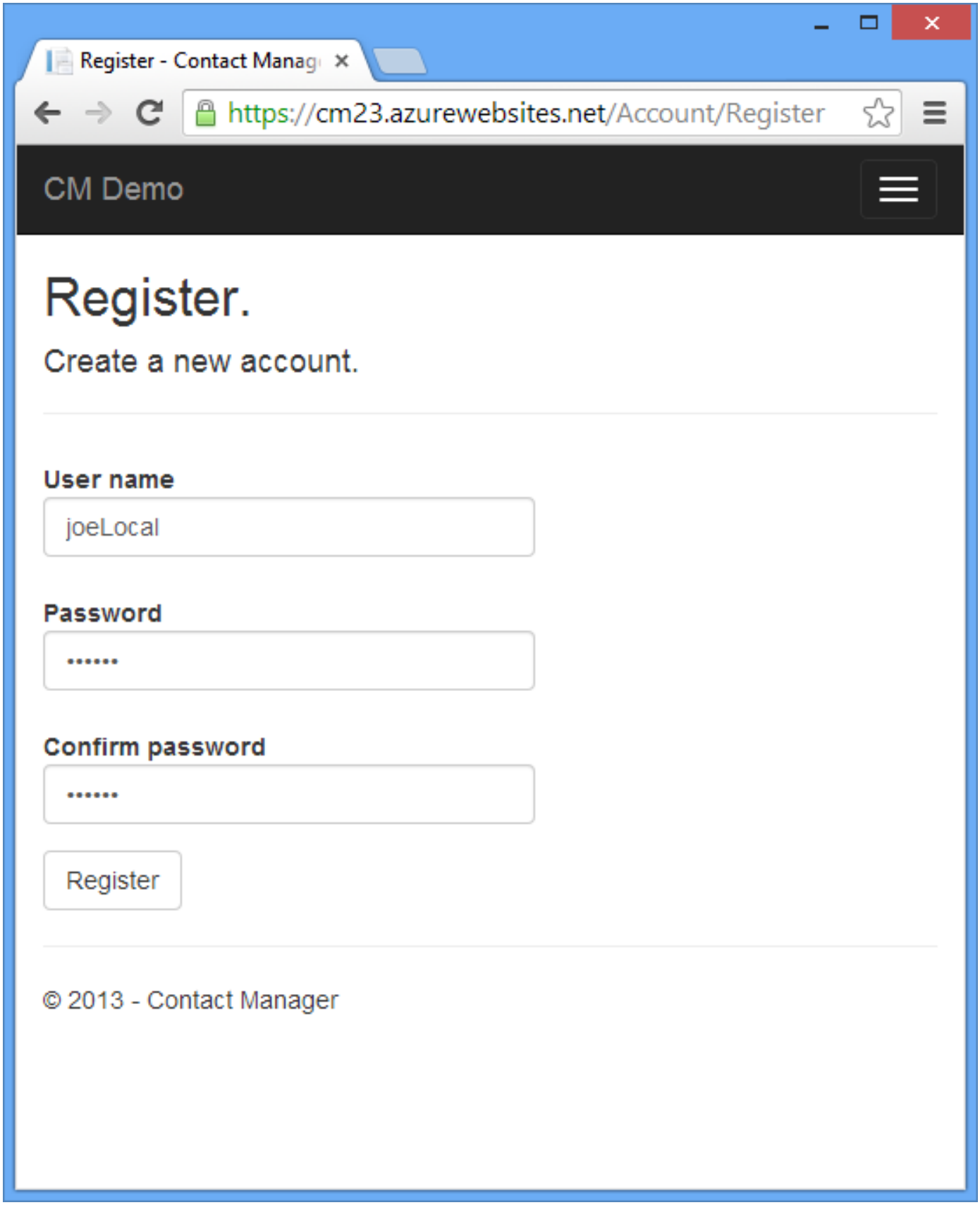
دکمه **Start Preview** را فشار دهید. در این مرحله تنها فایل هایی که نیاز به بروز رسانی دارند آپلود خواهند شد.

وب سایت را راه اندازی کنید. ساده‌ترین راه از طریق پرتال مدیریت Windows Azure است. توجه داشته باشید که تا هنگامی که وب سایت شما متوقف شده، نمی‌توانید اپلیکیشن خود را منتشر کنید.

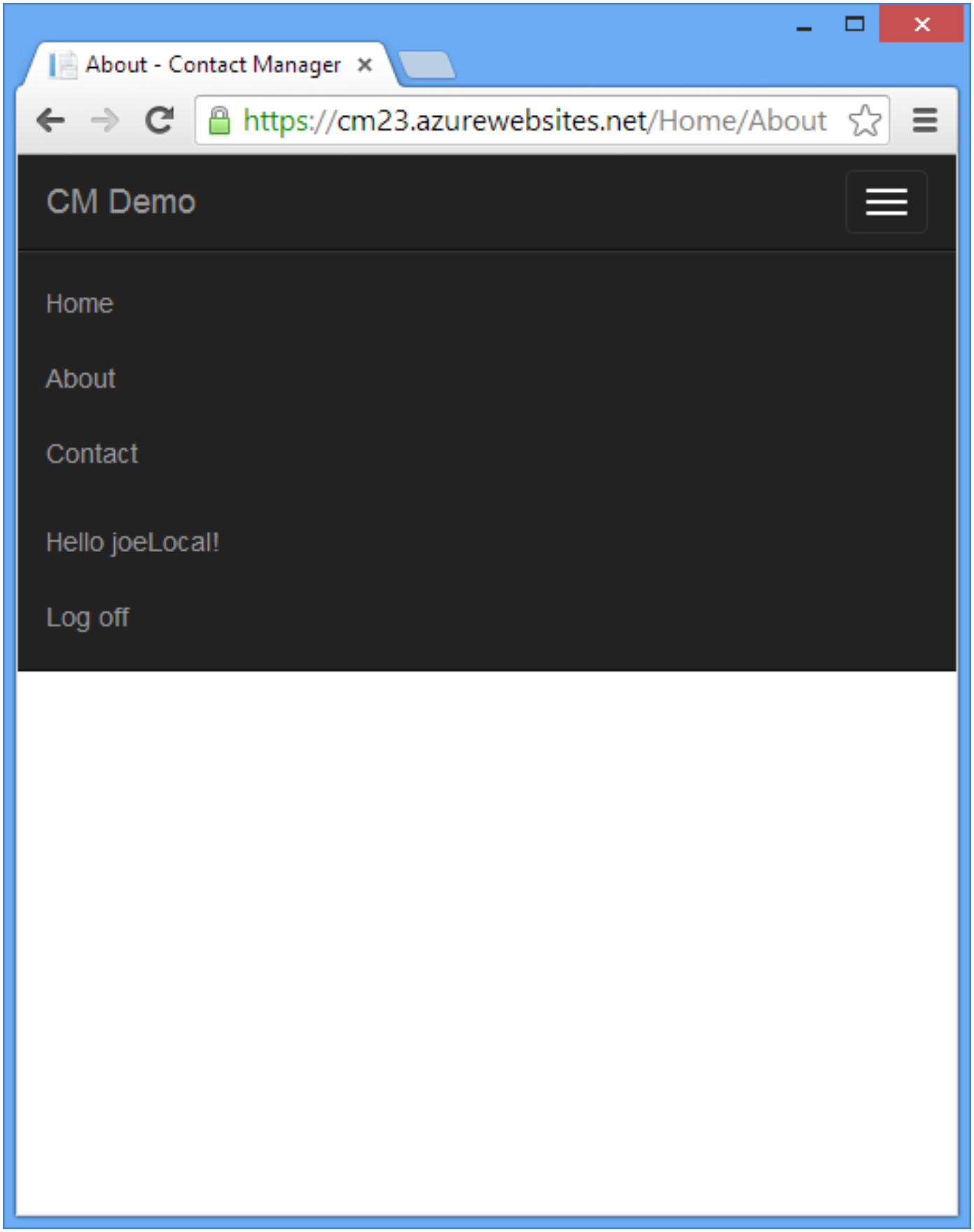
حال به ویژوال استودیو بازگردید و اپلیکیشن را منتشر کنید. اپلیکیشن Windows Azure شما باید در مرورگر پیش فرض تان باز شود. حال شما در حال مشاهده صفحه اصلی سایت بعنوان یک کاربر ناشناس هستید.

روی لینک **About** کلیک کنید، که شما را به صفحه ورود هدایت می‌کند.

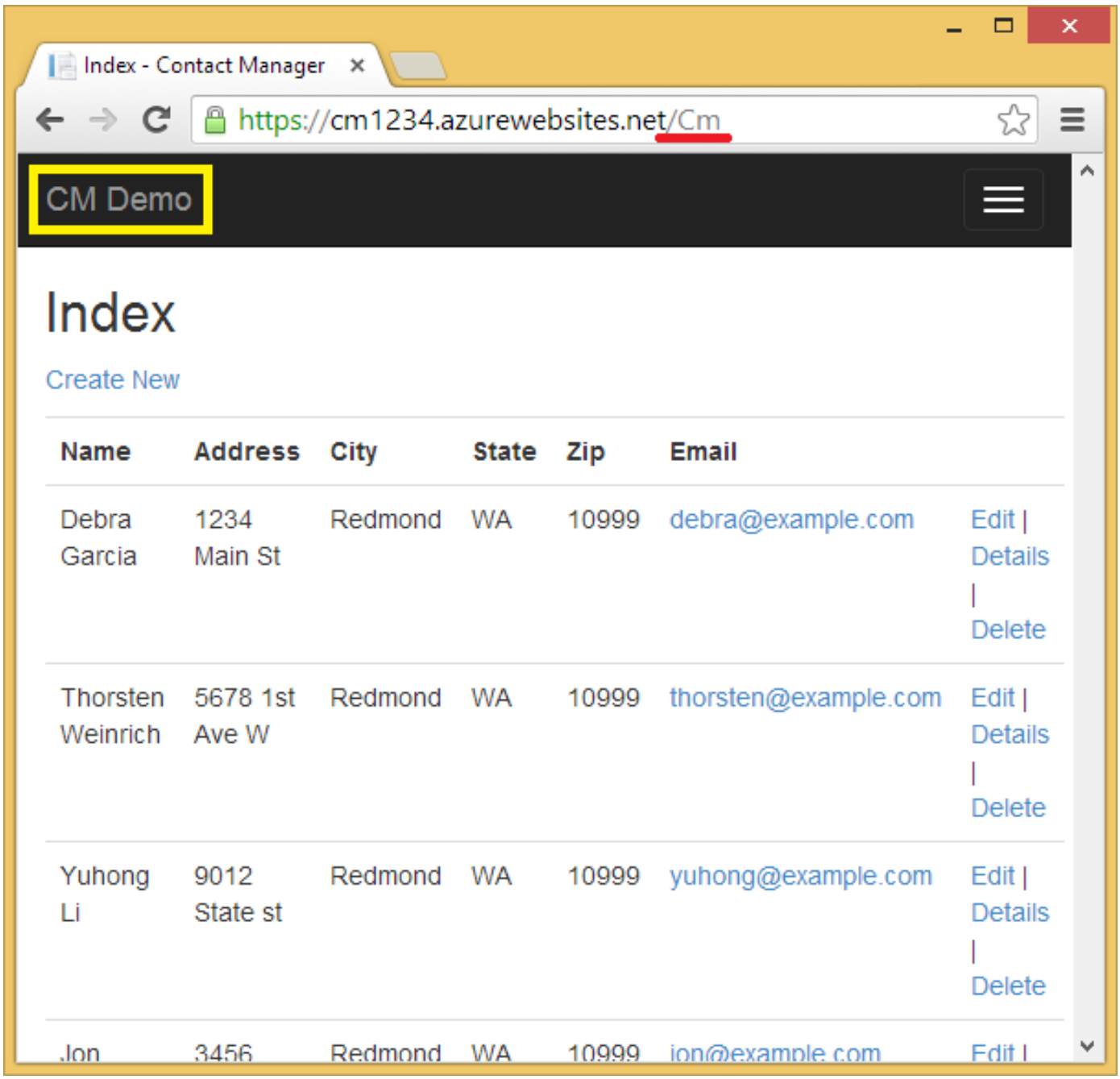
روی لینک **Register** در صفحه ورود کلیک کنید و یک حساب کاربری محلی بسازید. از این حساب کاربری برای این استفاده می‌کنیم که ببینیم شما به صفحات فقط خواندنی (read-only) و نه صفحاتی که داده‌ها را تغییر می‌دهند دسترسی دارید یا خیر. بعدا در ادامه مقاله، دسترسی حساب‌های کاربری محلی (local) را حذف می‌کنیم.



مطمئن شوید که به صفحات About و Contact دسترسی دارید.



لینک **CM Demo** را کلیک کنید تا به کنترلر *CmController* هدایت شوید.



روی یکی از لینک‌های Edit کلیک کنید. این کار شما را به صفحه ورود به سایت هدایت می‌کند. در زیر قسمت **User another service to log in** یکی از گزینه‌های Google یا Facebook را انتخاب کنید و توسط حساب کاربری ای که قبلا ساختید وارد شوید.

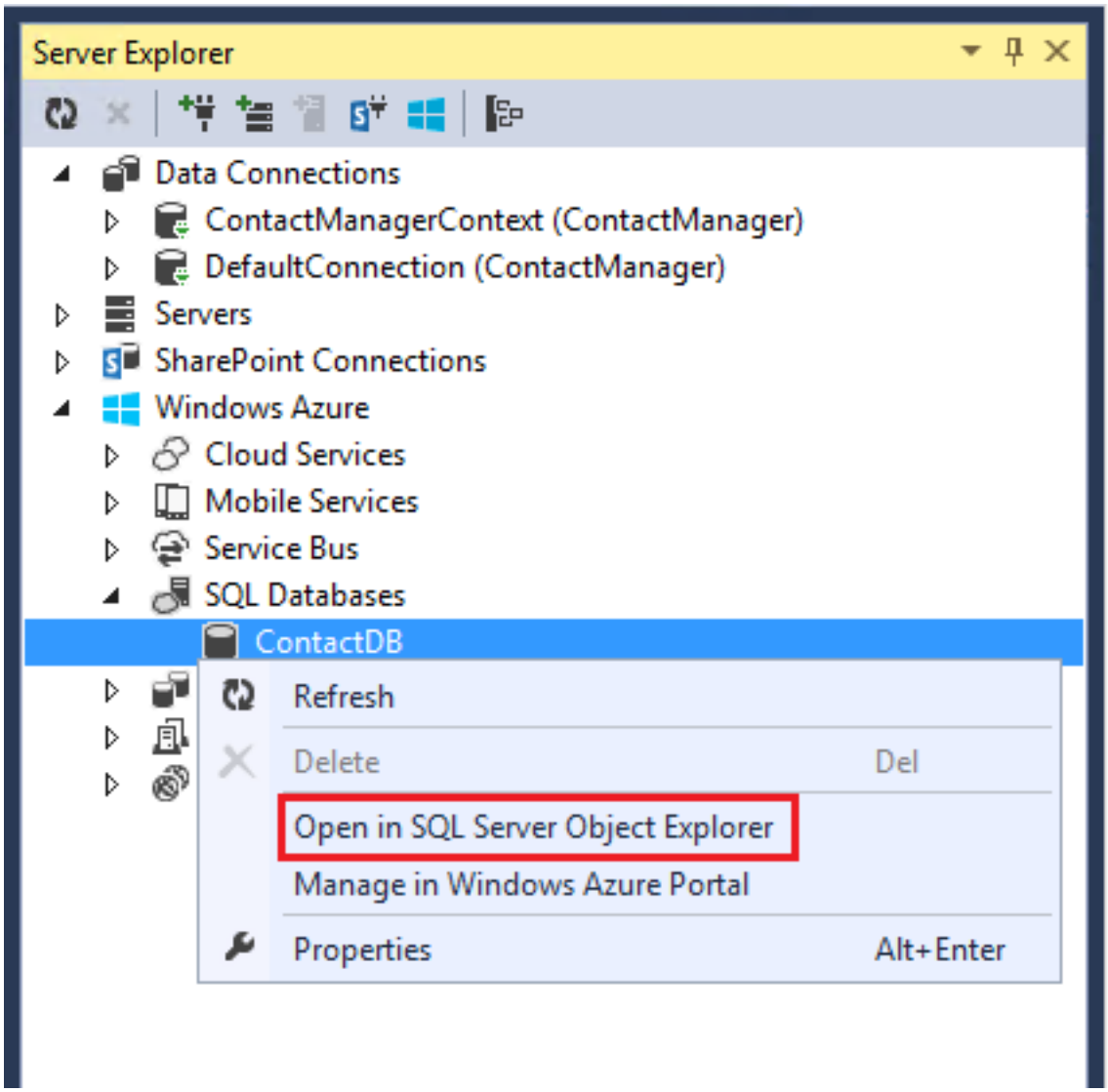
حال بررسی کنید که امکان ویرایش اطلاعات را دارید یا خیر.

نکته: شما نمی‌توانید در این اپلیکیشن از اکانت گوگل خود خارج شده، و با همان مرورگر با اکانت گوگل دیگری وارد اپلیکیشن شوید. اگر دارید از یک مرورگر استفاده می‌کنید، باید به سایت گوگل رفته و از آنجا خارج شوید. برای وارد شدن به اپلیکیشن توسط یک اکانت دیگر می‌توانید از یک مرورگر دیگر استفاده کنید.

دیتابیس SQL Azure را بررسی کنید

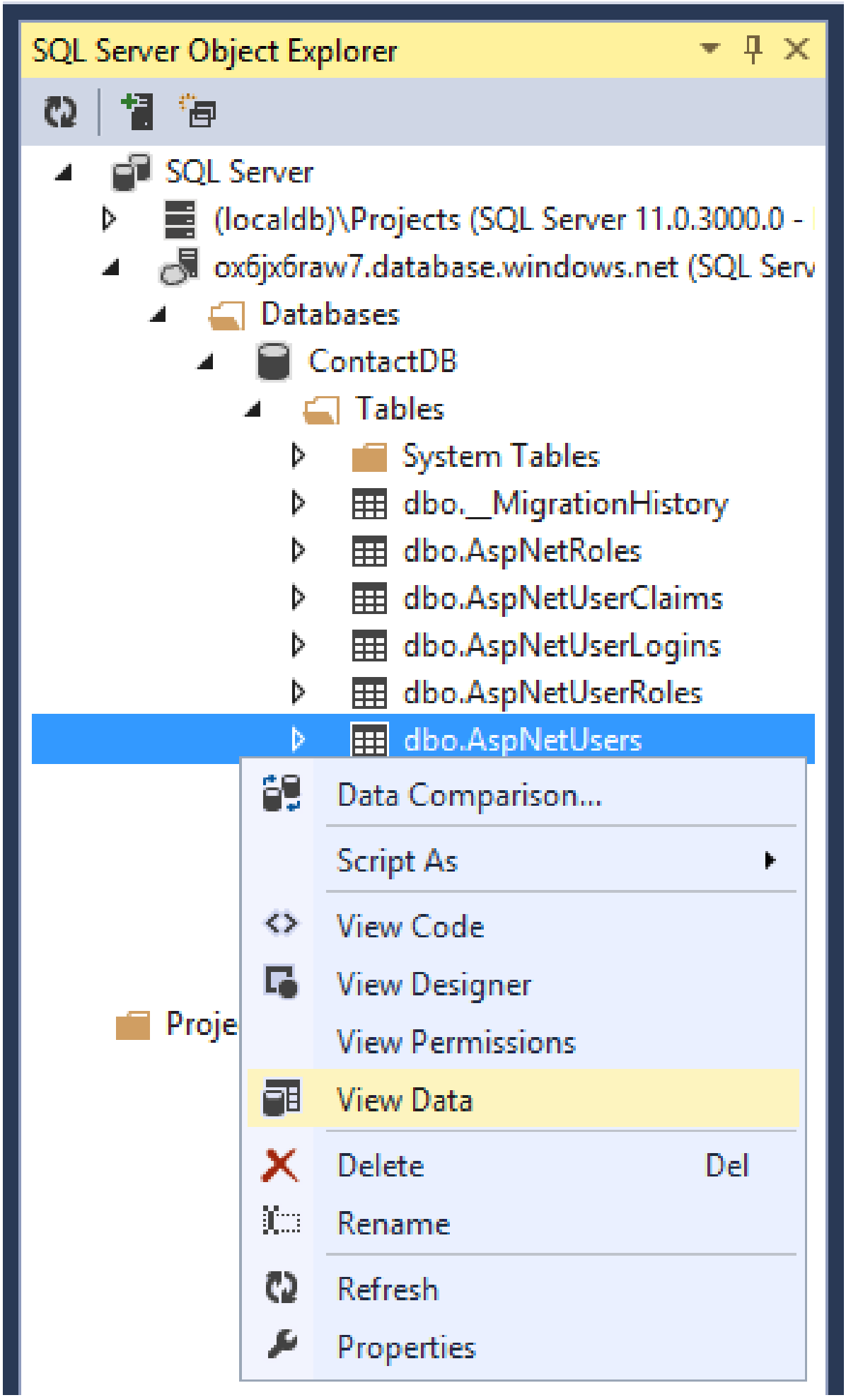
در **Server Explorer** دیتابیس **ContactDB** را پیدا کنید. روی آن کلیک راست کرده و **Open in SQL Server Object Explorer** را

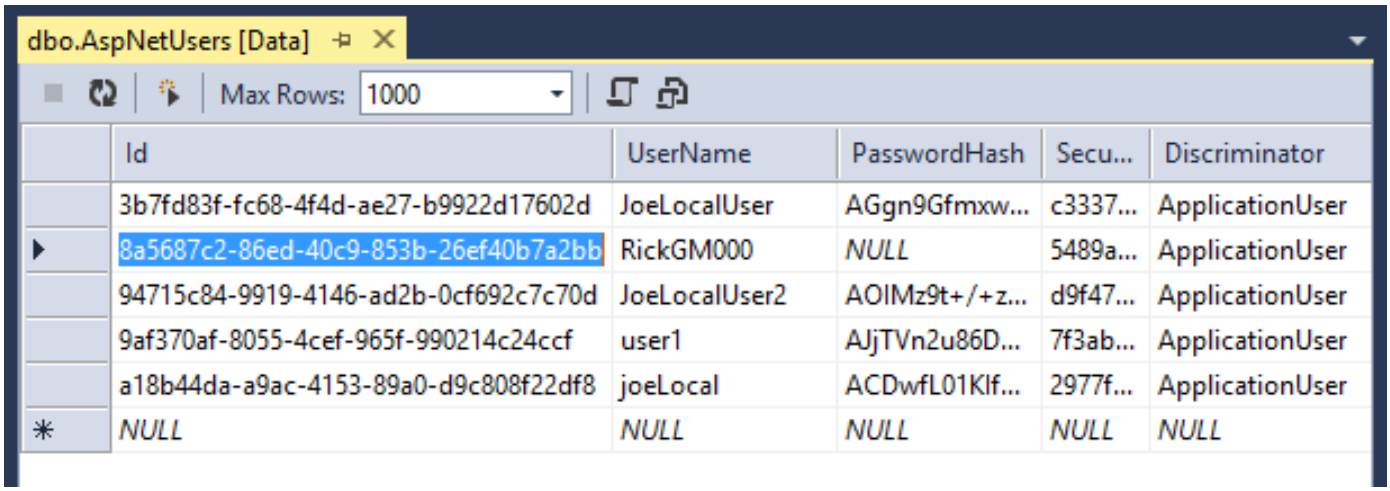
انتخاب کنید.



توجه: اگر نمی‌توانید گره **SQL Databases** را باز کنید و یا **ContactDB** را در ویژوال استودیو نمی‌بینید، باید مراحلی را طی کنید تا یک پورت یا یکسری پورت را به فایروال خود اضافه کنید. دقت داشته باشید که در صورت اضافه کردن Port Rangeها ممکن است چند دقیقه زمان نیاز باشد تا بتوانید به دیتابیس دسترسی پیدا کنید.

روی جدول **AspNetUsers** کلیک راست کرده و **View Data** را انتخاب کنید.

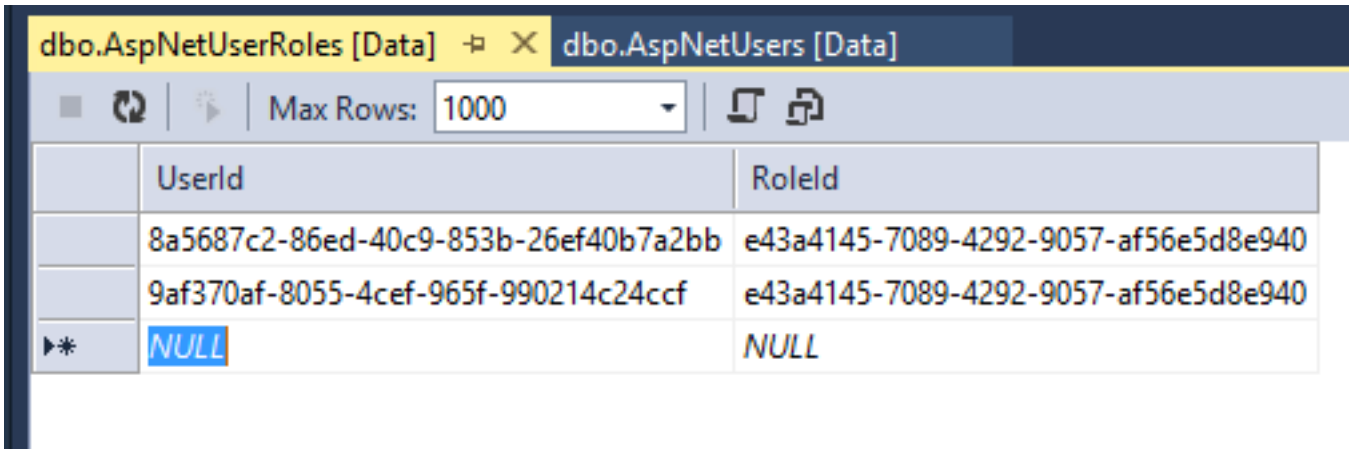

SQL Server Object Explorer
SQL Server
(localdb)\Projects (SQL Server 11.0.3000.0 -
ox6jx6raw7.database.windows.net (SQL Serv
Databases
ContactDB
Tables
System Tables
dbo.__MigrationHistory
dbo.AspNetRoles
dbo.AspNetUserClaims
dbo.AspNetUserLogins
dbo.AspNetUserRoles
dbo.AspNetUsers
Data Comparison...
Script As
View Code
View Designer
View Permissions
View Data
Delete
Del
Rename
Refresh
Properties


dbo.AspNetUsers [Data]

Max Rows: 1000

Id	UserName	PasswordHash	Secu...	Discriminator
3b7fd83f-fc68-4f4d-ae27-b9922d17602d	JoeLocalUser	AGgn9Gfmxw...	c3337...	ApplicationUser
8a5687c2-86ed-40c9-853b-26ef40b7a2bb	RickGM000	*NULL*	5489a...	ApplicationUser
94715c84-9919-4146-ad2b-0cf692c7c70d	JoeLocalUser2	AOIMz9t+/+z...	d9f47...	ApplicationUser
9af370af-8055-4cef-965f-990214c24ccf	user1	AJjTVn2u86D...	7f3ab...	ApplicationUser
a18b44da-a9ac-4153-89a0-d9c808f22df8	joeLocal	ACDwfL01Klf...	2977f...	ApplicationUser
NULL	*NULL*	*NULL*	*NULL*	*NULL*

حالا روی **AspNetUserRoles** کلیک راست کنید و **View Data** را انتخاب کنید.


dbo.AspNetUserRoles [Data] | dbo.AspNetUsers [Data]

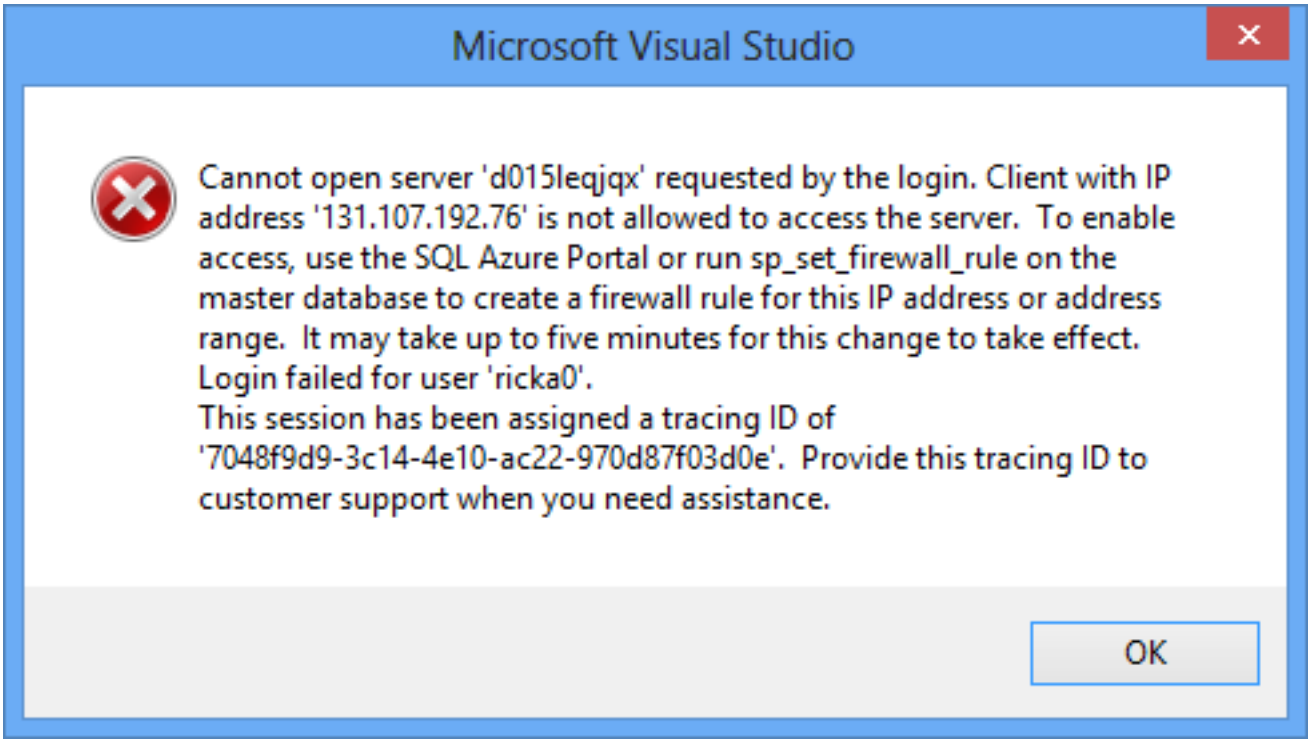
Max Rows: 1000

UserId	RoleId
8a5687c2-86ed-40c9-853b-26ef40b7a2bb	e43a4145-7089-4292-9057-af56e5d8e940
9af370af-8055-4cef-965f-990214c24ccf	e43a4145-7089-4292-9057-af56e5d8e940
NULL	*NULL*

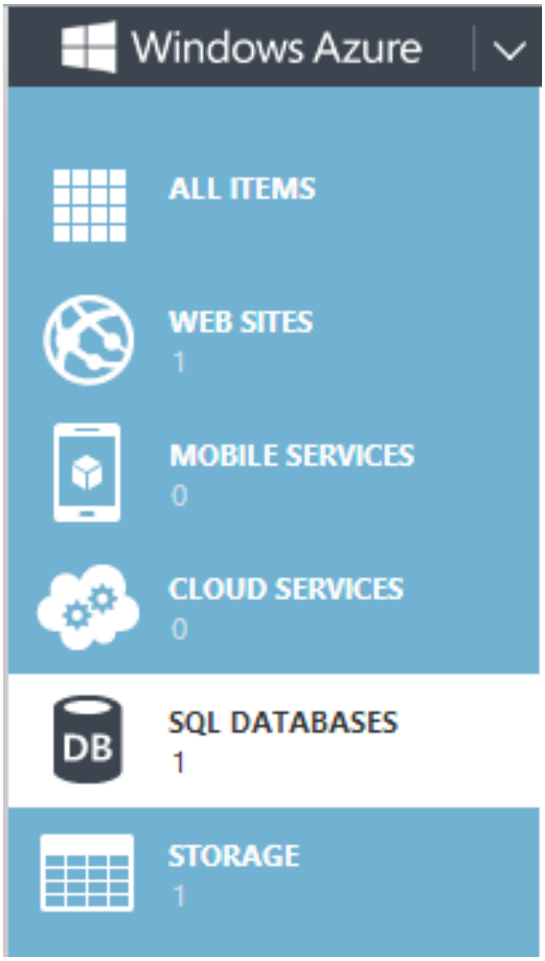
اگر شناسه کاربران (User ID) را بررسی کنید، مشاهده می‌کنید که تنها دو کاربر *user1* و اکانت گوگل شما به نقش *canEdit* تعلق دارند.

Cannot open server login error

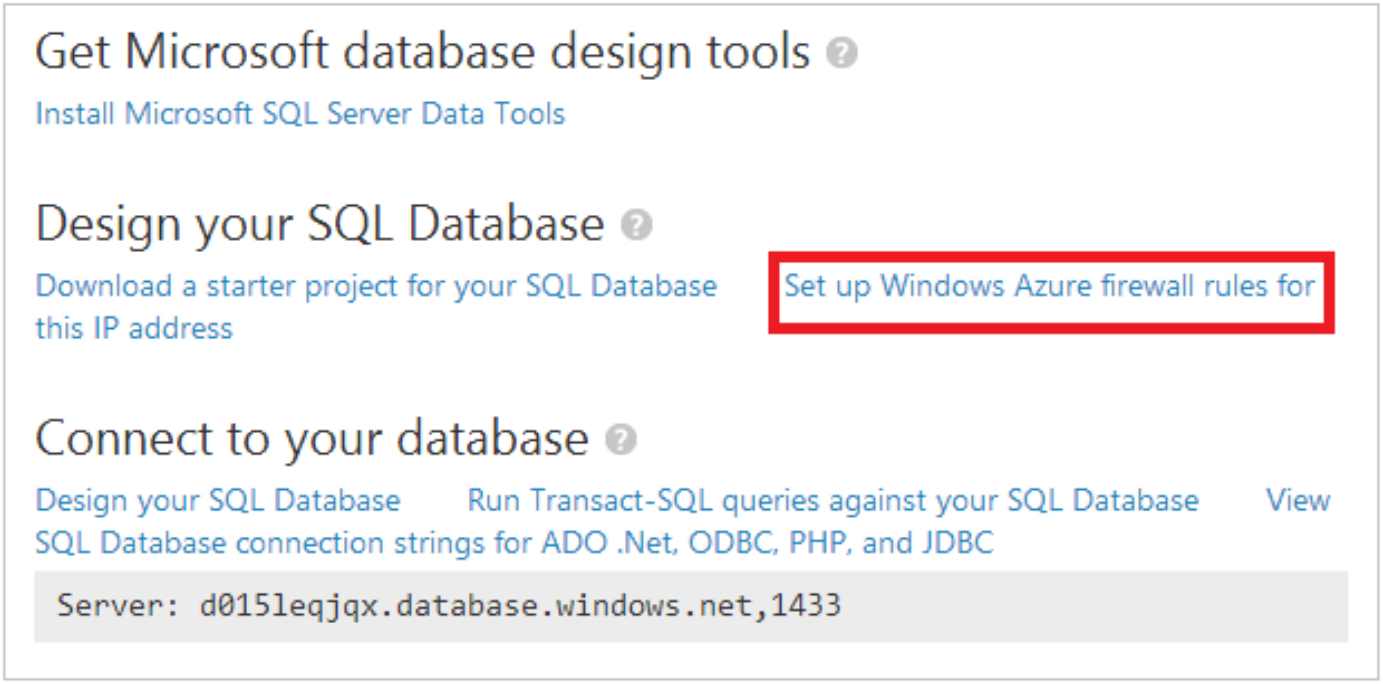
اگر خطایی مبنی بر "Cannot open server" دریافت می‌کنید، مراحل زیر را دنبال کنید.



شما باید آدرس IP خود را به لیست آدرس‌های مجاز (Allowed IPs) اضافه کنید. در پرتال مدیریتی Windows Azure در قسمت چپ صفحه، گزینه **SQL Databases** را انتخاب کنید.



دیتابیس مورد نظر را انتخاب کنید. حالا روی لینک **Set up Windows Azure firewall rules for this IP address** کلیک کنید.



هنگامی که با پیغام "The current IP address xxx.xxx.xxx.xxx is not included in existing firewall rules. Do you want to update the firewall rules?" مواجه شدید **Yes** را کلیک کنید. افزودن یک آدرس IP بدین روش معمولا کافی نیست و در فایروال‌های سازمانی و بزرگ باید Range بیشتری را تعریف کنید.

مرحله بعد اضافه کردن محدوده آدرس‌های مجاز است.

مجددا در پرتال مدیریتی Windows Azure روی **SQL Databases** کلیک کنید. سروری که دیتابیس شما را میزبانی می‌کند انتخاب کنید.

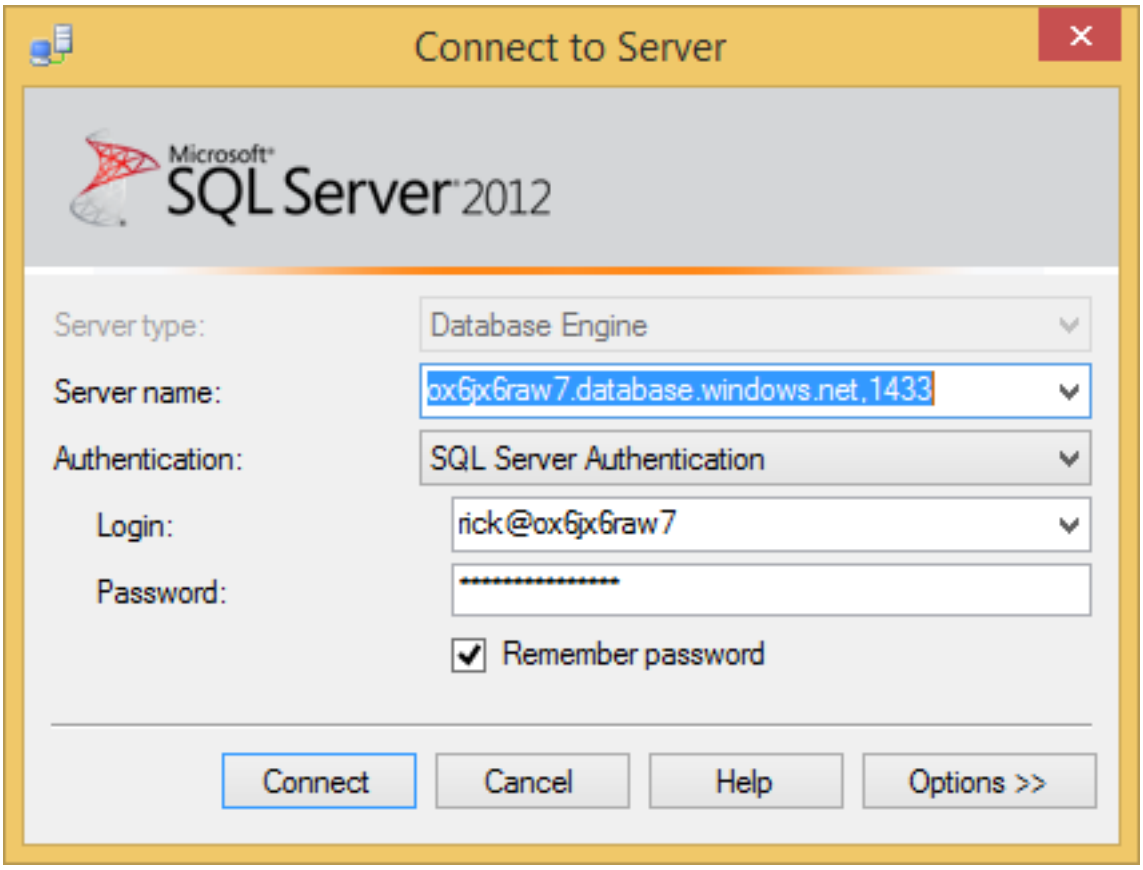
در بالای صفحه لینک **Configure** را کلیک کنید. حالا نام rule جدید، آدرس شروع و پایان را وارد کنید.

در پایین صفحه **Save** را کلیک کنید.

در آخر می‌توانید توسط SSOX به دیتابیس خود متصل شوید. از منوی **View** گزینه **SQL Server Object Explorer** را انتخاب کنید. روی **SQL Server** کلیک راست کرده و **Add SQL Server** را انتخاب کنید.

در دیالوگ **Connect to Server** متد احراز هویت را به **SQL Server Authentication** تغییر دهید. این کار نام سرور و اطلاعات ورود پرتال Windows Azure را به شما می‌دهد.

در مرورگر خود به پرتال مدیریتی بروید و **SQL Databases** را انتخاب کنید. دیتابیس **ContactDB** را انتخاب کرده و روی **View SQL Database connection strings** کلیک کنید. در صفحه **Connection Strings** مقادیر **Server** و **User ID** را کپی کنید. حالا مقادیر را در دیالوگ مذکور در ویژوال استودیو بچسبانید. مقدار فیلد **User ID** در قسمت **Login** وارد می‌شود. در آخر هم کلمه عبوری که هنگام ساختن دیتابیس تنظیم کردید را وارد کنید.

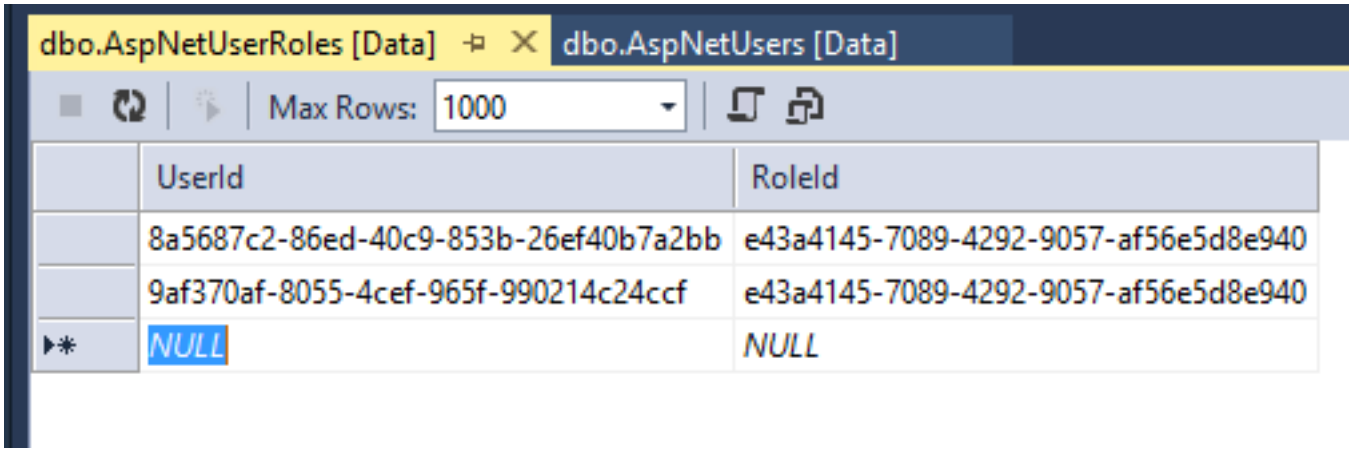


حالا می‌توانید با مراحلی که پیشتر توضیح داده شد به دیتابیس **Contact DB** مراجعه کنید.

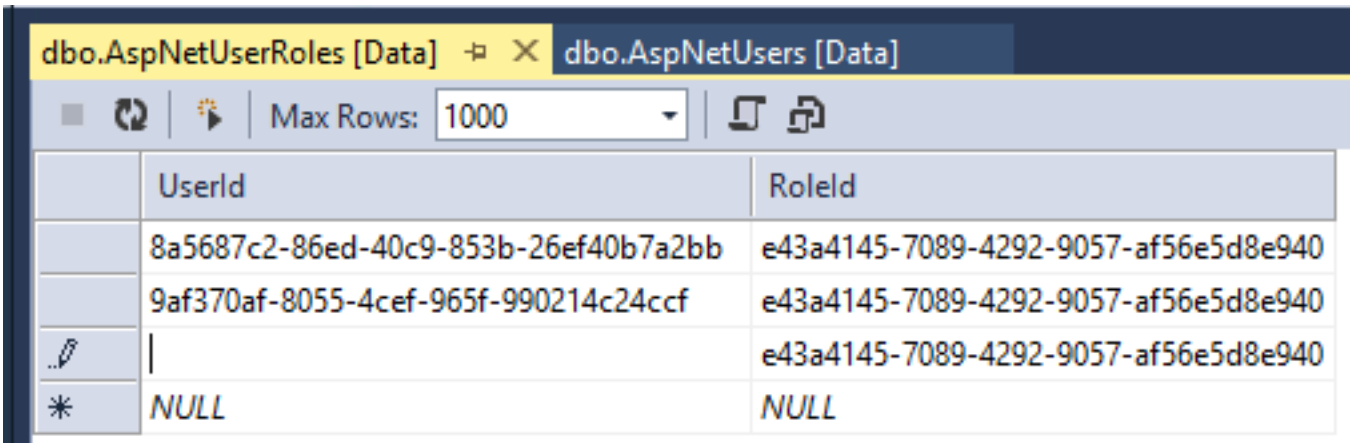
افزودن کاربران به نقش canEdit با ویرایش جداول دیتابیس

پیشتر در این مقاله، برای اضافه کردن کاربران به نقش canEdit از یک قطعه کد استفاده کردیم. یک راه دیگر تغییر جداول دیتابیس بصورت مستقیم است. مراحلی که در زیر آمده اند اضافه کردن کاربران به یک نقش را نشان می‌دهند.

در **SQL Server Object Explorer** روی جدول **AspNetUserRoles** کلیک راست کنید و **View Data** را انتخاب کنید.



حالا *RoleId* را کپی کنید و در ردیف جدید بچسبانید.



شناسه کاربر مورد نظر را از جدول **AspNetUsers** پیدا کنید و مقدار آن را در ردیف جدید کپی کنید. همین! کاربر جدید شما به نقش canEdit اضافه شد.

نکاتی درباره ثبت نام محلی (Local Registration)

ثبت نام فعلی ما از بازنشانی کلمه‌های عبور (password reset) پشتیبانی نمی‌کند. همچنین اطمینان حاصل نمی‌شود که کاربران سایت انسان هستند (مثلا با استفاده از یک CAPTCHA). پس از آنکه کاربران توسط تامین کنندگان خارجی (مانند گوگل) احراز هویت شدند، می‌توانند در سایت ثبت نام کنند. اگر می‌خواهید ثبت نام محلی را برای اپلیکیشن خود غیرفعال کنید این مراحل را دنبال کنید:

در کنترلر Account متدهای *Register* را ویرایش کنید و خاصیت **AllowAnonymous** را از آنها حذف کنید (هر دو متد GET و POST).

این کار ثبت نام کاربران ناشناس و بدافزارها (bots) را غیر ممکن می‌کند.

در پوشه *Views/Shared* فایل *_LoginPartial.cshtml* را باز کنید و لینک Register را از آن حذف کنید.

در فایل *Views/Account/Login.cshtml* نیز لینک Register را حذف کنید.

اپلیکیشن را دوباره منتشر کنید.

قدم‌های بعدی

برای اطلاعات بیشتر درباره نحوه استفاده از Facebook بعنوان یک تامین کننده احراز هویت، و اضافه کردن اطلاعات پروفایل به قسمت ثبت نام کاربران به لینک زیر مراجعه کنید. Create an ASP.NET MVC 5 App with Facebook and Google OAuth2 and

OpenID Sign-on

برای یادگیری بیشتر درباره ASP.NET MVC 5 هم به سری مقالات Getting Started with ASP.NET MVC 5 می توانید مراجعه کنید. همچنین سری مقالات Getting Started with EF and MVC مطالب خوبی درباره مفاهیم پیشرفته EF ارائه می‌کند.

نظرات خوانندگان

نویسنده: مهمان
تاریخ: ۱۳۹۲/۱۰/۱۹ ۱۴:۴

دوست عزیز

با صبر و حوصله و دقت فراوان یک مقاله خوب را منتشر کردید. ممنون(رای من 5)

عنوان: **دریافت اطلاعات بیشتر از Social Provider ها در VS 2013**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۹ ۱۷:۳۵
آدرس: www.dotnettips.info
گروه‌ها: OpenID, ASP.NET MVC 5, ASP.NET Identity, OAuth, MVC, Security

هنگامی که یک پروژه جدید ASP.NET را در VS 2013 می‌سازید و متد احراز هویت آن را Individual User Accounts انتخاب می‌کنید، قالب پروژه، امکانات لازم را برای استفاده از تامین کنندگان ثالث، فراهم می‌کند، مثلا مایکروسافت، گوگل، توییتر و فیسبوک. هنگامی که توسط یکی از این تامین کننده‌ها کاربری را احراز هویت کردید، می‌توانید اطلاعات بیشتری درخواست کنید. مثلا عکس پروفایل کاربر یا لیست دوستان او. سپس اگر کاربر به اپلیکیشن شما سطح دسترسی کافی داده باشد می‌توانید این اطلاعات را دریافت کنید و تجربه کاربری قوی‌تر و بهتری ارائه کنید.
در این پست خواهید دید که چطور می‌شود از تامین کننده Facebook اطلاعات بیشتری درخواست کرد. پیش فرض این پست بر این است که شما با احراز هویت فیسبوک و سیستم کلی تامین کننده‌ها آشنایی دارید. برای اطلاعات بیشتر درباره راه اندازی احراز هویت فیسبوک به این لینک مراجعه کنید.
برای دریافت اطلاعات بیشتر از فیسبوک مراحل زیر را دنبال کنید.
یک اپلیکیشن جدید ASP.NET MVC با تنظیمات Individual User Accounts بسازید.
احراز هویت فیسبوک را توسط کلید هایی که از Facebook دریافت کرده اید فعال کنید. برای اطلاعات بیشتر در این باره می‌توانید به این لینک مراجعه کنید.
برای درخواست اطلاعات بیشتر از فیسبوک، فایل Startup.Auth.cs را مطابق لیست زیر ویرایش کنید.

```
List<string> scope = newList<string>() { "email", "user_about_me", "user_hometown", "friends_about_me",
"friends_photos" };
 var x = newFacebookAuthenticationOptions();
 x.Scope.Add("email");
 x.Scope.Add("friends_about_me");
 x.Scope.Add("friends_photos");
 x.AppId = "636919159681109";
 x.AppSecret = "f3c16511fe95e854cf5885c10f83f26f";
 x.Provider = newFacebookAuthenticationProvider()
{
    OnAuthenticated = async context =>
    {
         //Get the access token from FB and store it in the database and
        //use FacebookC# SDK to get more information about the user
        context.Identity.AddClaim(
        new System.Security.Claims.Claim("FacebookAccessToken",
                                         context.AccessToken));
    }
};
 x.SignInAsAuthenticationType = DefaultAuthenticationTypes.ExternalCookie;
 app.UseFacebookAuthentication(x);
```

در خط 1 مشخص می‌کنیم که چه scope هایی از داده را می‌خواهیم درخواست کنیم.
از خط 10 تا 17 رویداد OnAuthenticated را مدیریت می‌کنیم که از طرف Facebook OWIN authentication اجرا می‌شود. این متد هر بار که کاربری با فیسبوک خودش را احراز هویت می‌کند فراخوانی می‌شود. پس از آنکه کاربر احراز هویت شد و به اپلیکیشن سطح دسترسی لازم را اعطا کرد، تمام داده‌ها در FacebookContext ذخیره می‌شوند.
خط 14 شناسه FacebookAccessToken را ذخیره می‌کند. ما این آبجکت را از فیسبوک دریافت کرده و از آن برای دریافت لیست دوستان کاربر استفاده می‌کنیم.
نکته: در این مثال تمام داده‌ها بصورت Claims ذخیره می‌شوند، اما اگر بخواهید می‌توانید از ASP.NET Identity برای ذخیره آنها در دیتابیس استفاده کنید.
در قدم بعدی لیست دوستان کاربر را از فیسبوک درخواست می‌کنیم. ابتدا فایل Views/Shared/_LoginPartial.cshtml را باز کنید و لینک زیر را به آن بیافزایید.

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

هنگامی که کاربری وارد سایت می‌شود و این لینک را کلیک می‌کند، ما لیست دوستان او را از فیسبوک درخواست می‌کنیم و بهمراه عکس‌های پروفایل شان آنها را لیست می‌کنیم.
تمام Claimها را از **UserIdentity** بگیرید و آنها را در دیتابیس ذخیره کنید. در این قطعه کد ما تمام Claim هایی که توسط OWIN دریافت کرده ایم را می‌خوانیم، و شناسه FacebookAccessToken را در دیتابیس عضویت ASP.NET Identity ذخیره می‌کنیم.

```
//
        // GET: /Account/LinkLoginCallback
        publicasyncTask<ActionResult> LinkLoginCallback()
        {
            var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync(XsrfKey,
User.Identity.GetUserId());
            if (loginInfo == null)
            {
                return RedirectToAction("Manage", new { Message = ManageMessageId.Error });
            }
            var result = await UserManager.AddLoginAsync(User.Identity.GetUserId(), loginInfo.Login);
            if (result.Succeeded)
            {
                var currentUser = await UserManager.FindByIdAsync(User.Identity.GetUserId());
                //Add the Facebook Claim
                await StoreFacebookAuthToken(currentUser);
                return RedirectToAction("Manage");
            }
            return RedirectToAction("Manage", new { Message = ManageMessageId.Error });
        }
```

خط 14-15 شناسه FacebookAccessToken را در دیتابیس ذخیره می‌کند.
StoreFacebookAuthToken تمام اختیارات (claim)های کاربر را از UserIdentity می‌گیرد و Access Token را در قالب یک User Claim در دیتابیس ذخیره می‌کند. اکشن LinkLoginCallback هنگامی فراخوانی می‌شود که کاربر وارد سایت شده و یک تامین کننده دیگر را می‌خواهد تنظیم کند.
اکشن ExternalLoginConfirmation هنگام اولین ورود شما توسط تامین کنندگان اجتماعی مانند فیسبوک فراخوانی می‌شود.
در خط 26 پس از آنکه کاربر ایجاد شد ما یک FacebookAccessToken را بعنوان یک Claim برای کاربر ذخیره می‌کنیم.

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> ExternalLoginConfirmation(ExternalLoginConfirmationViewModel model,
string returnUrl)
        {
            if (User.Identity.IsAuthenticated)
            {
                return RedirectToAction("Manage");
            }

            if (ModelState.IsValid)
            {
                // Get the information about the user from the external login provider
                var info = await AuthenticationManager.GetExternalLoginInfoAsync();
                if (info == null)
                {
                    return View("ExternalLoginFailure");
                }
                var user = newApplicationUser() { UserName = model.Email };
                var result = await UserManager.CreateAsync(user);
                if (result.Succeeded)
                {
                    result = await UserManager.AddLoginAsync(user.Id, info.Login);
                    if (result.Succeeded)
                    {
                        await StoreFacebookAuthToken(user);
                        await SignInAsync(user, isPersistent: false);
                        return RedirectToLocal(returnUrl);
                    }
                }
                AddErrors(result);
            }

            ViewBag.ReturnUrl = returnUrl;
```

```
            return View(model);
        }
```

اکشن **ExternalLoginCallback** هنگامی فراخوانی می‌شود که شما برای اولین بار یک کاربر را به یک تامین کننده اجتماعی اختصاص می‌دهید. در خط 17 شناسه دسترسی فیسبوک را بصورت یک claim برای کاربر ذخیره می‌کنیم.

```
//
        // GET: /Account/ExternalLoginCallback
        [AllowAnonymous]
        publicasyncTask<ActionResult> ExternalLoginCallback(string returnUrl)
        {
            var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync();
            if (loginInfo == null)
            {
                return RedirectToAction("Login");
            }

            // Sign in the user with this external login provider if the user already has a login
            var user = await UserManager.FindAsync(loginInfo.Login);
            if (user != null)
            {
                //Save the FacebookToken in the database if not already there
                await StoreFacebookAuthToken(user);
                await SignInAsync(user, isPersistent: false);
                return RedirectToLocal(returnUrl);
            }
            else
            {
                // If the user does not have an account, then prompt the user to create an account
                ViewBag.ReturnUrl = returnUrl;
                ViewBag.LoginProvider = loginInfo.Login.LoginProvider;
                return View("ExternalLoginConfirmation", newExternalLoginConfirmationViewModel { Email
= loginInfo.Email });
            }
        }
```

در آخر شناسه FacebookAccessToken را در دیتابیس ASP.NET Identity ذخیره کنید.

```
privateasyncTask StoreFacebookAuthToken(ApplicationUser user)
        {
            var claimsIdentity = await
AuthenticationManager.GetExternalIdentityAsync(DefaultAuthenticationTypes.ExternalCookie);
            if (claimsIdentity != null)
            {
                // Retrieve the existing claims for the user and add the FacebookAccessTokenClaim
                var currentClaims = await UserManager.GetClaimsAsync(user.Id);
                var facebookAccessToken = claimsIdentity.FindAll("FacebookAccessToken").First();
                if (currentClaims.Count() <=0 )
                {
                    await UserManager.AddClaimAsync(user.Id, facebookAccessToken);
                }
```

پکیج Facebook C#SDK را نصب کنید. http://nuget.org/packages/Facebook

فایل AccountViewModel.cs را باز کنید و کد زیر را اضافه کنید.

```
public class FacebookViewModel
     {
         [Required]
         [Display(Name = "Friend's name")]
         public string Name { get; set; }

        public string ImageURL { get; set; }
    }
```

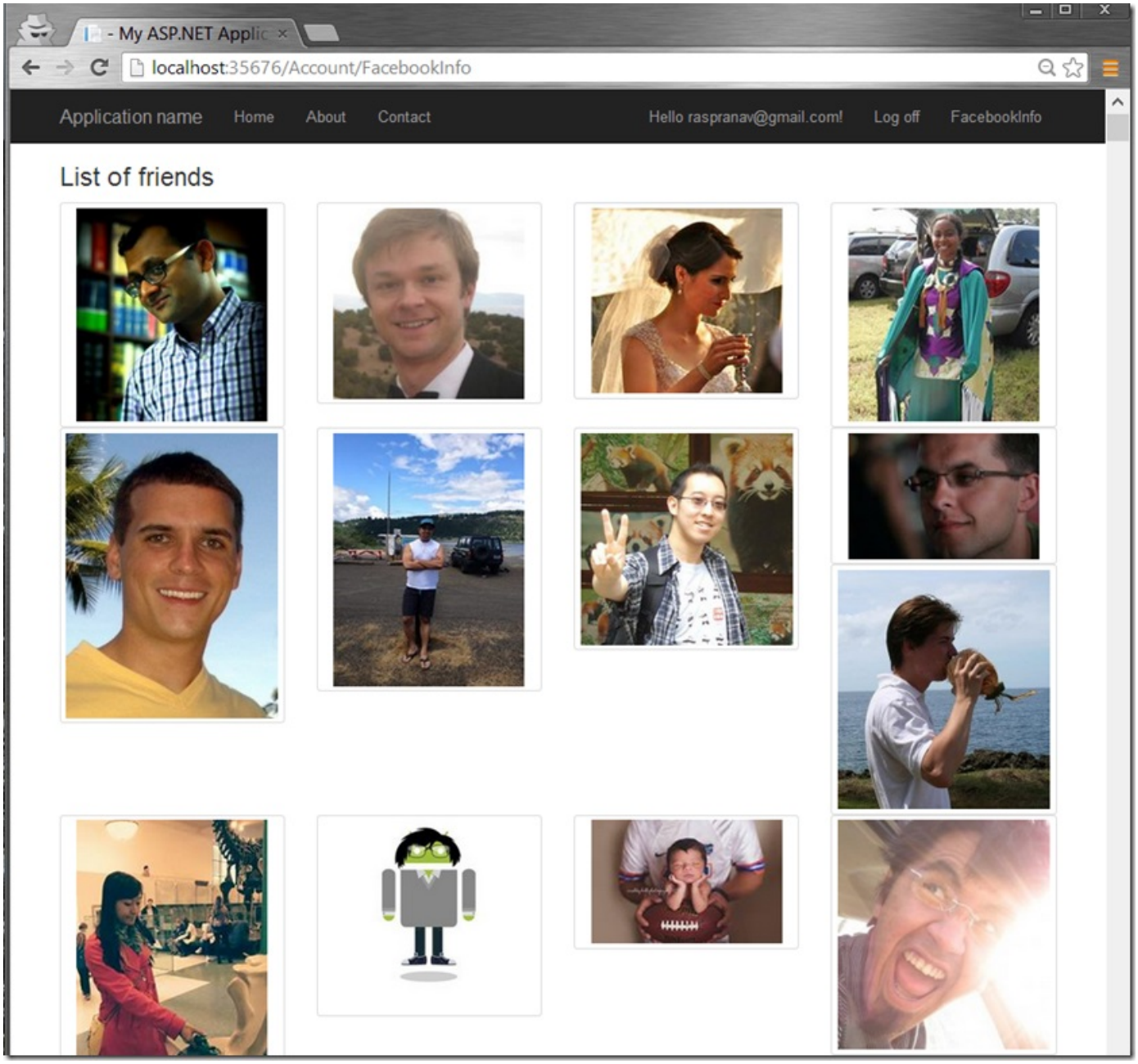
کد زیر را به کنترلر Account اضافه کنید تا عکس‌های دوستان تان را دریافت کنید.

```
//GET: Account/FacebookInfo
[Authorize]
publicasyncTask<ActionResult> FacebookInfo()
{
    var claimsforUser = await UserManager.GetClaimsAsync(User.Identity.GetUserId());
    var access_token = claimsforUser.FirstOrDefault(x => x.Type == "FacebookAccessToken").Value;
    var fb = newFacebookClient(access_token);
    dynamic myInfo = fb.Get("/me/friends");
    var friendsList = newList<FacebookViewModel>();
    foreach (dynamic friend in myInfo.data)
    {
        friendsList.Add(newFacebookViewModel()
           {
               Name = friend.name,
               ImageURL = @"https://graph.facebook.com/" + friend.id + "/picture?type=large"
           });
    }

    return View(friendsList);
}
```

در پوشه Views/Account یک نمای جدید با نام FacebookInfo.cshtml بسازید و کد Markup آن را مطابق لیست زیر تغییر دهید.

```
@model IList<WebApplication96.Models.FacebookViewModel>
 @if (Model.Count > 0)
 {
     <h3>List of friends</h3>
     <div class="row">
             @foreach (var friend in Model)
             {
               <div class="col-md-3">
                <a href="#" class="thumbnail">
                  <img src=@friend.ImageURL alt=@friend.Name />
                 </a>
               </div>
              }
     </div>
 }
```

در این مرحله، شما می‌توانید لیست دوستان خود را بهمراه عکس‌های پروفایل شان دریافت کنید.
پروژه را اجرا کنید و توسط Facebook وارد سایت شوید. باید به سایت فیسبوک هدایت شوید تا احراز هویت کنید و دسترسی لازم را به اپلیکیشن اعطا کنید. پس از آن مجددا به سایت خودتان باید هدایت شوید.
حال هنگامی که روی لینک FacebookInfo کلیک می‌کنید باید صفحه ای مشابه تصویر زیر ببینید.



این یک مثال ساده از کار کردن با تامین کنندگان اجتماعی بود. همانطور که مشاهده می‌کنید، براحتی می‌توانید داده‌های بیشتری برای کاربر جاری درخواست کنید و تجربه کاربری و امکانات بسیار بهتری را در اپلیکیشن خود فراهم کنید.

نظرات خوانندگان

نویسنده: حامد شیربندی
تاریخ: ۱۳۹۳/۱۱/۲۹ ۱۹:۳

با سلام و تشکر
توی دریافت اطلاعات لیست دوستان از فیسبوک باید دقت داشت که طبق مستندات فیسبوک برای Graph API در اینجا ما فقط به اطلاعات دوستانی دسترسی خواهیم داشت که از Facebook Login استفاده کرده باشند.
البته این نکته رو هم باید اضافه کنم که برای دسترسی به هر اطلاعاتی از اکانت فیسبوک کاربر، باید مجوزش رو هم ارسال کنیم
بنابراین برای دریافت اطلاعات مربوط به لیست دوستان باید دستور زیر رو هم به کلاس startUp اضافه کنیم

```
x.Scope.Add("user_friends");
```

برای اطلاع از نام کل مجوزها و حتی تست اونها برای دریافت اطلاعات اکانت از Graph API Explorer استفاده کنید.