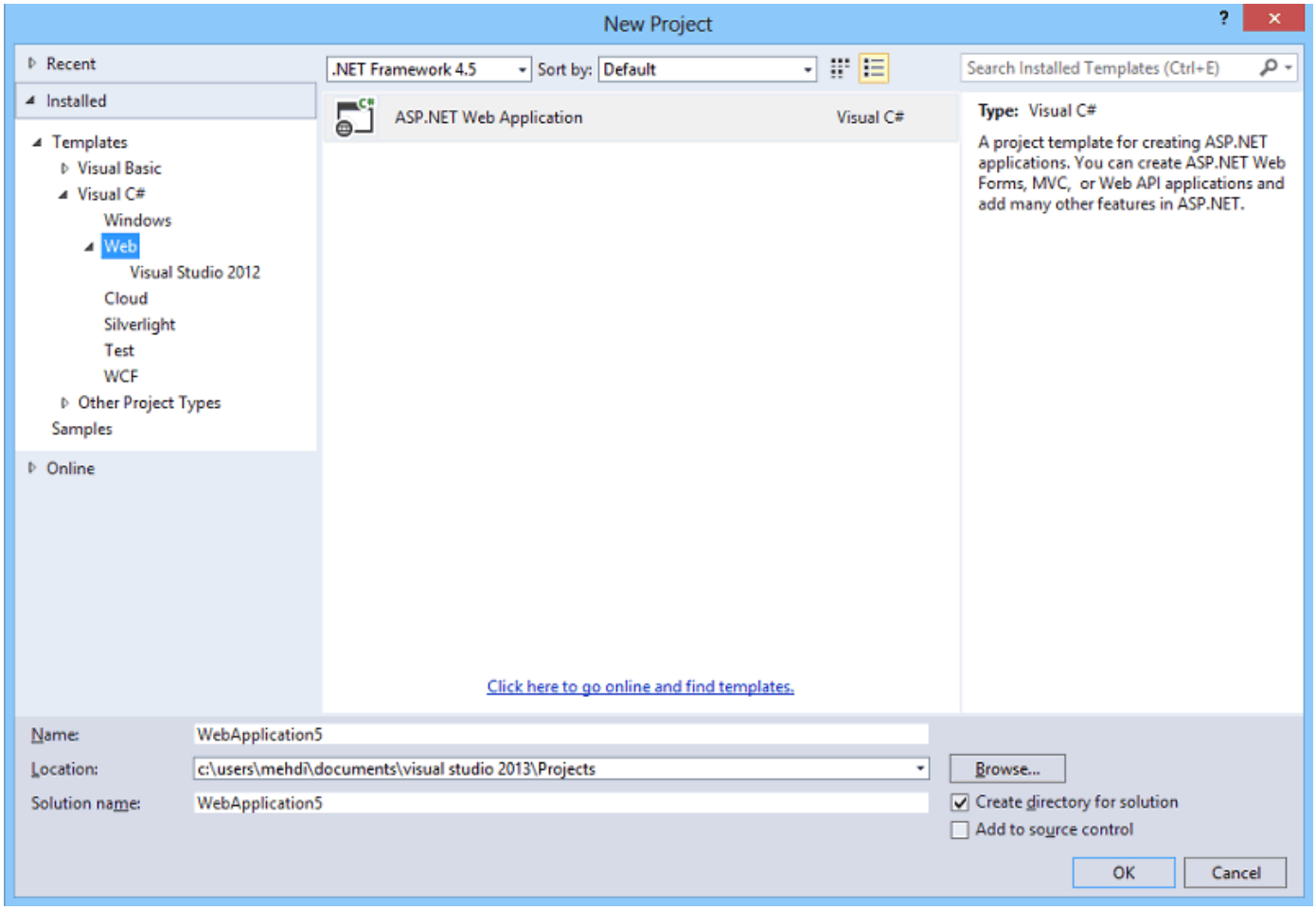
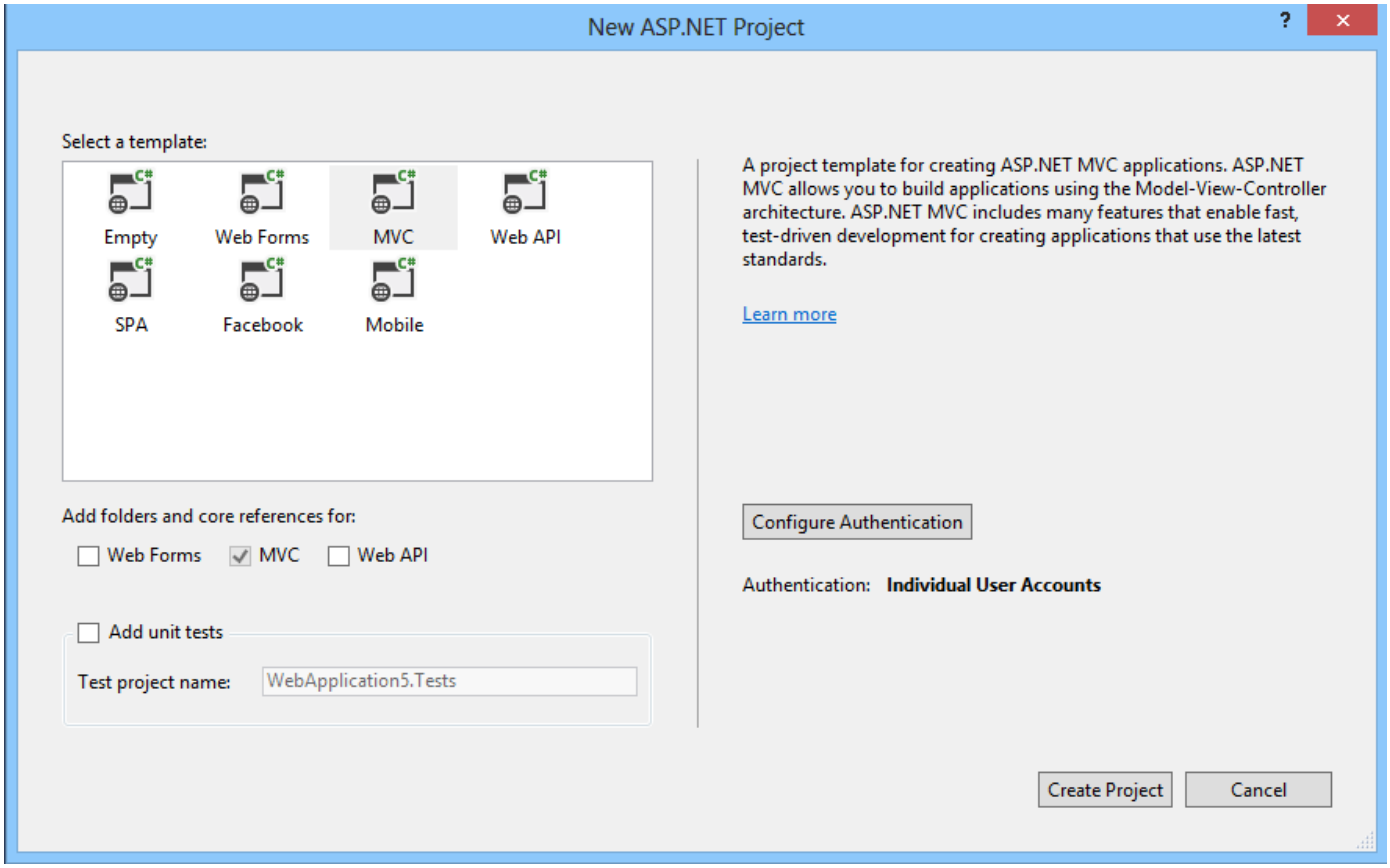
عنوان: **بررسی تغییرات ASP.NET MVC 5 beta1**
نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۴/۱۰ ۲۳:۱۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.NET MVC 5, MVC

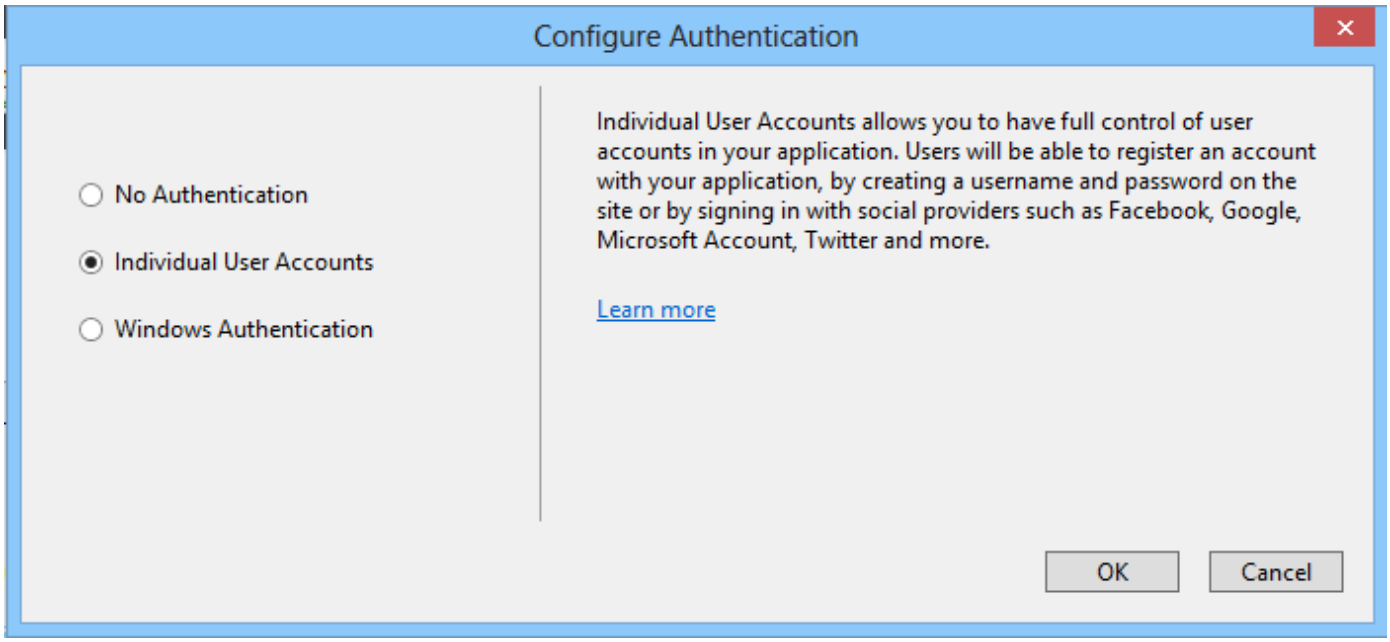
همانطور که می‌دانید، مایکروسافت در کنفرانس Build 2013 که چند روز پیش برگزار شد، Visual Studio 2013 Preview را به همراه ASP.NET MVC 5 beta1 و Entity Framework 6 beta 1 و تعدادی محصول دیگر، معرفی کرد.
در طی این مقاله قصد دارم تجربیات کار خودم با نسخه‌ی پیش نمایش MVC 5 را به اشتراک بزارم و نه صرفا بررسی یک change-log ساده.
برای کار با MVC 5 شما ابتدا باید یکی از نسخه‌های Visual Studio 2013 را نصب کنید. من در مقاله از Visual Studio Express 2013 Preview For Web استفاده می‌کنم.
ابتدا New Project را زده تا یک پروژه جدید را آغاز کنیم. از قسمت Templates، بخش Web را که انتخاب کنید، اولین تغییر را مشاهده خواهید کرد.بله! دیگر خبر از چند ASP.NET نیست.حداقل در دسته بندی تبدیل به یک **ASP.NET واحد** شده اند.



با انتخاب ok باز نیز با قالب جدیدی به شکل زیر برای انتخاب پروژه مواجه می‌شوید.



اینجا همه چیز تکراری است به غیر از گزینه `Configure Authentication`.

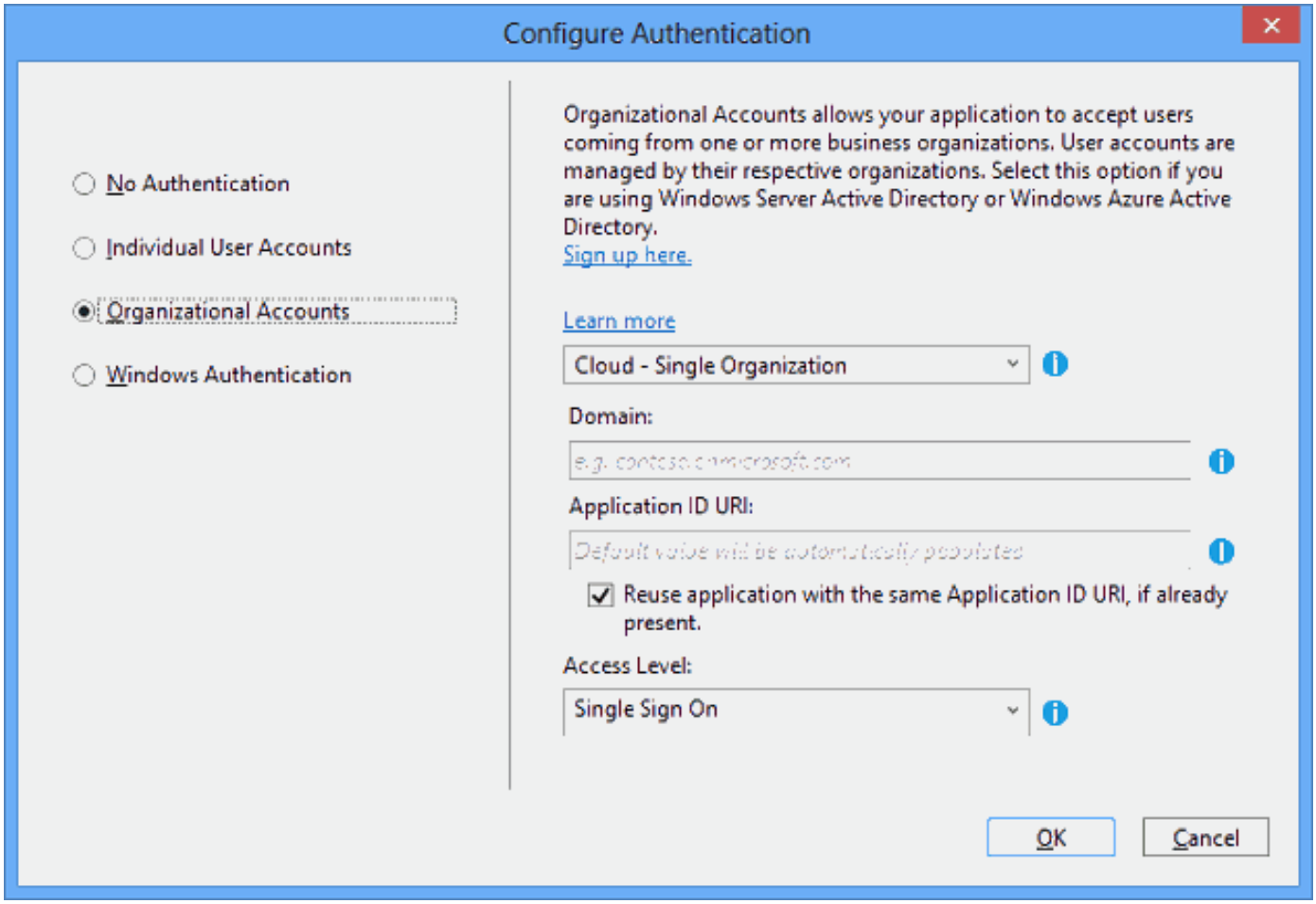


همه‌ی گزینه‌ها تکراری اند به غیر از گزینه `Individual User Accounts`. البته این همان `FormsAuthentication` قبلی است. نکته قابل توجه، یکپارچی آن با سرویس‌های اجتماعی و شبکه‌های سرویس دهنده است. البته در نسخه‌ی قبلی نیز این سیستم وجود

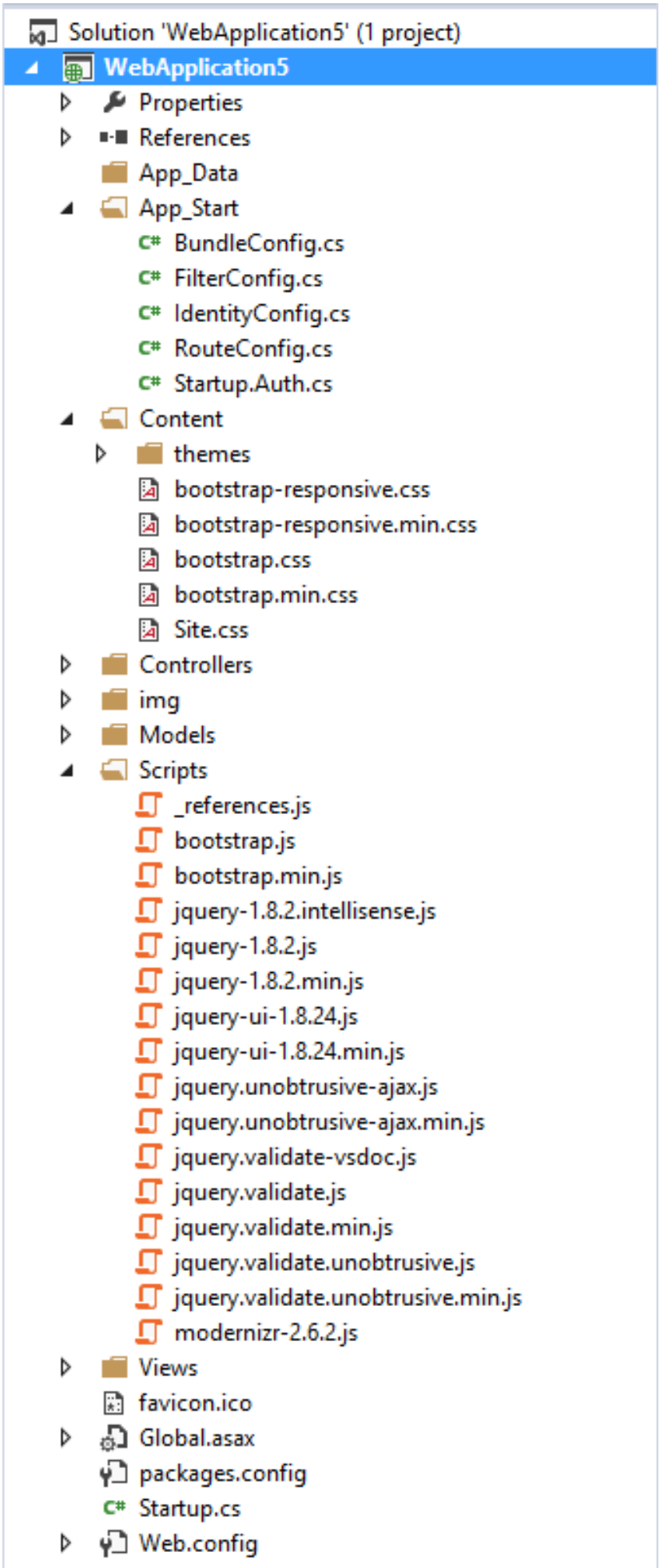
داشت، ولی این دفعه با ASP.NET Identity یک پارچه است که در ادامه بیشتر آن را خواهید دید.

البته گویا حالت دیگری به نام **Organizational Accounts** نیز وجود دارد که گویا برای فعال سازی، باید یک بسته‌ی به روز رسانی دریافت می‌کردم، که من نکردم.(اینترنت حجمی و شبانه دانلود کردن...)

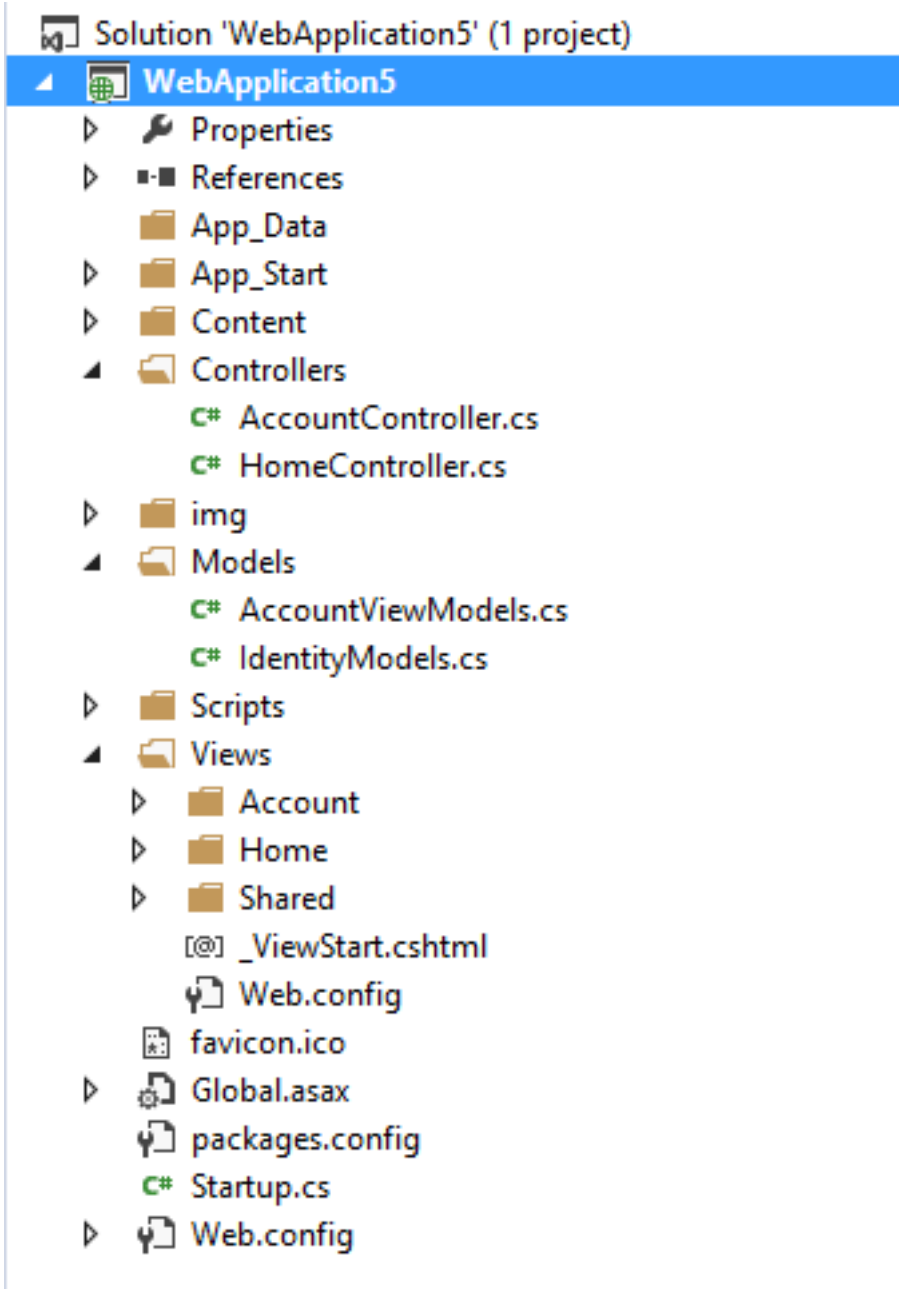
این حالت که در شکل زیر مشخص است، امکان یکپارچگی احراز هویت با Active Directory در windows server و azure را دارد.



پس از ایجاد پروژه یک نگاهی به Solution Explorer می‌اندازیم.

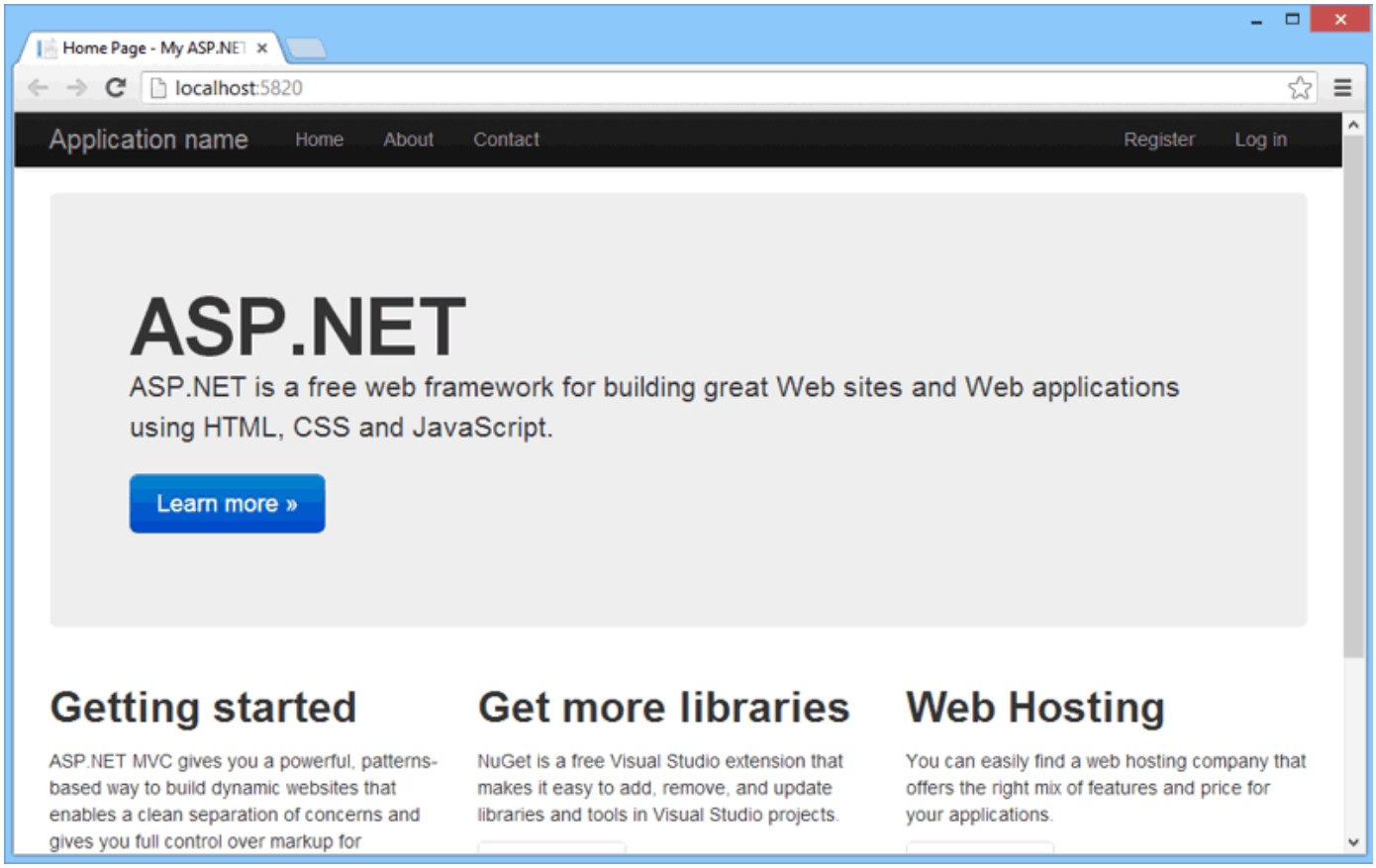

Solution 'WebApplication5' (1 project)
WebApplication5
Properties
References
App_Data
App_Start
BundleConfig.cs
FilterConfig.cs
IdentityConfig.cs
RouteConfig.cs
Startup.Auth.cs
Content
themes
bootstrap-responsive.css
bootstrap-responsive.min.css
bootstrap.css
bootstrap.min.css
Site.css
Controllers
img
Models
Scripts
_references.js
bootstrap.js
bootstrap.min.js
jquery-1.8.2.intellisense.js
jquery-1.8.2.js
jquery-1.8.2.min.js
jquery-ui-1.8.24.js
jquery-ui-1.8.24.min.js
jquery.unobtrusive-ajax.js
jquery.unobtrusive-ajax.min.js
jquery.validate-vsdoc.js
jquery.validate.js
jquery.validate.min.js
jquery.validate.unobtrusive.js
jquery.validate.unobtrusive.min.js
modernizr-2.6.2.js
Views
favicon.ico
Global.asax
packages.config
Startup.cs
Web.config

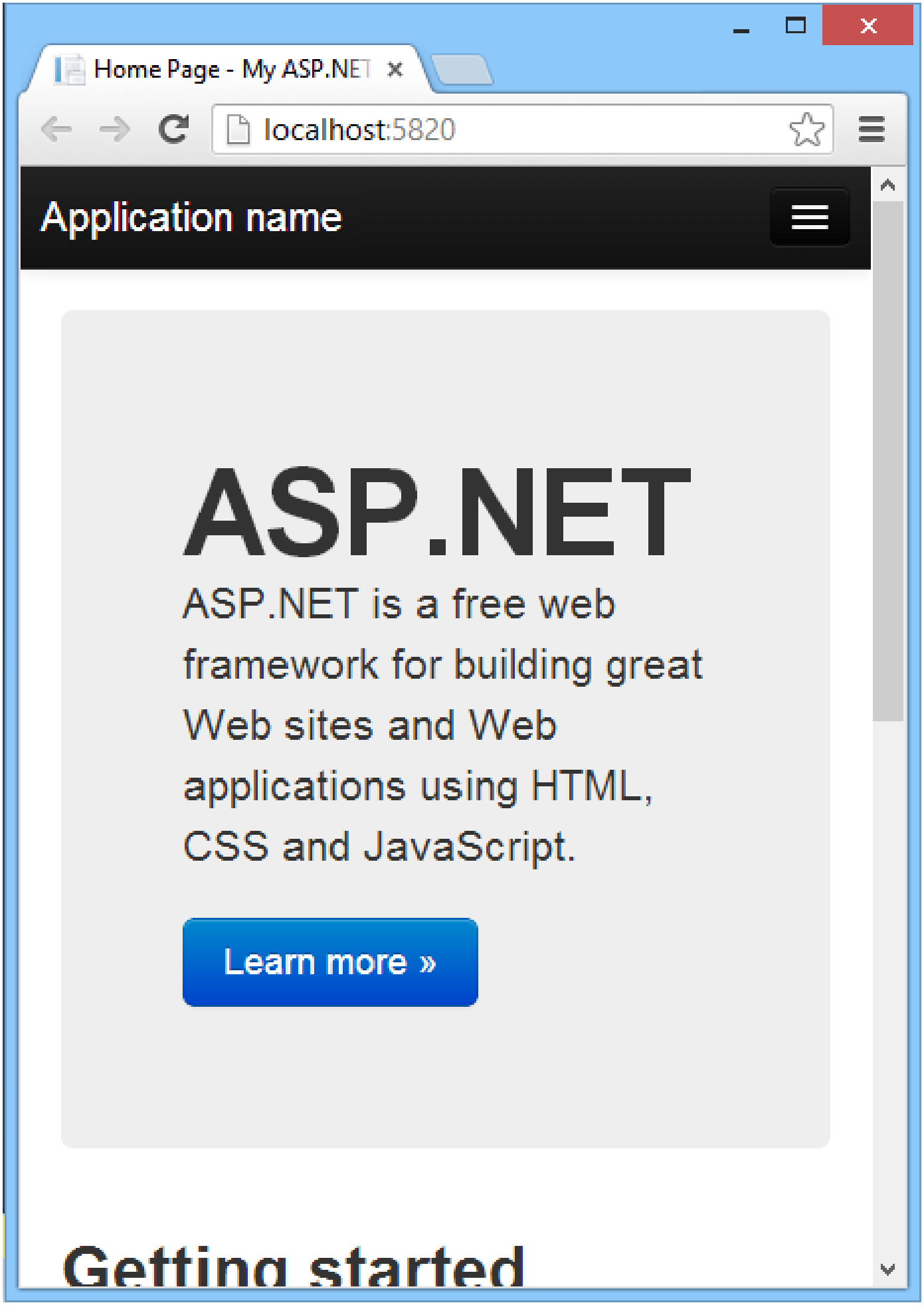
همان طور که می‌بینید ساختار اصلی با نسخه‌های قبلی هیچ تفاوتی نکرده و تنها کتاب خانه ای که اینجا خودنمایی می‌کند و به چشم آشنا نیست، **twitter bootstrap** است!

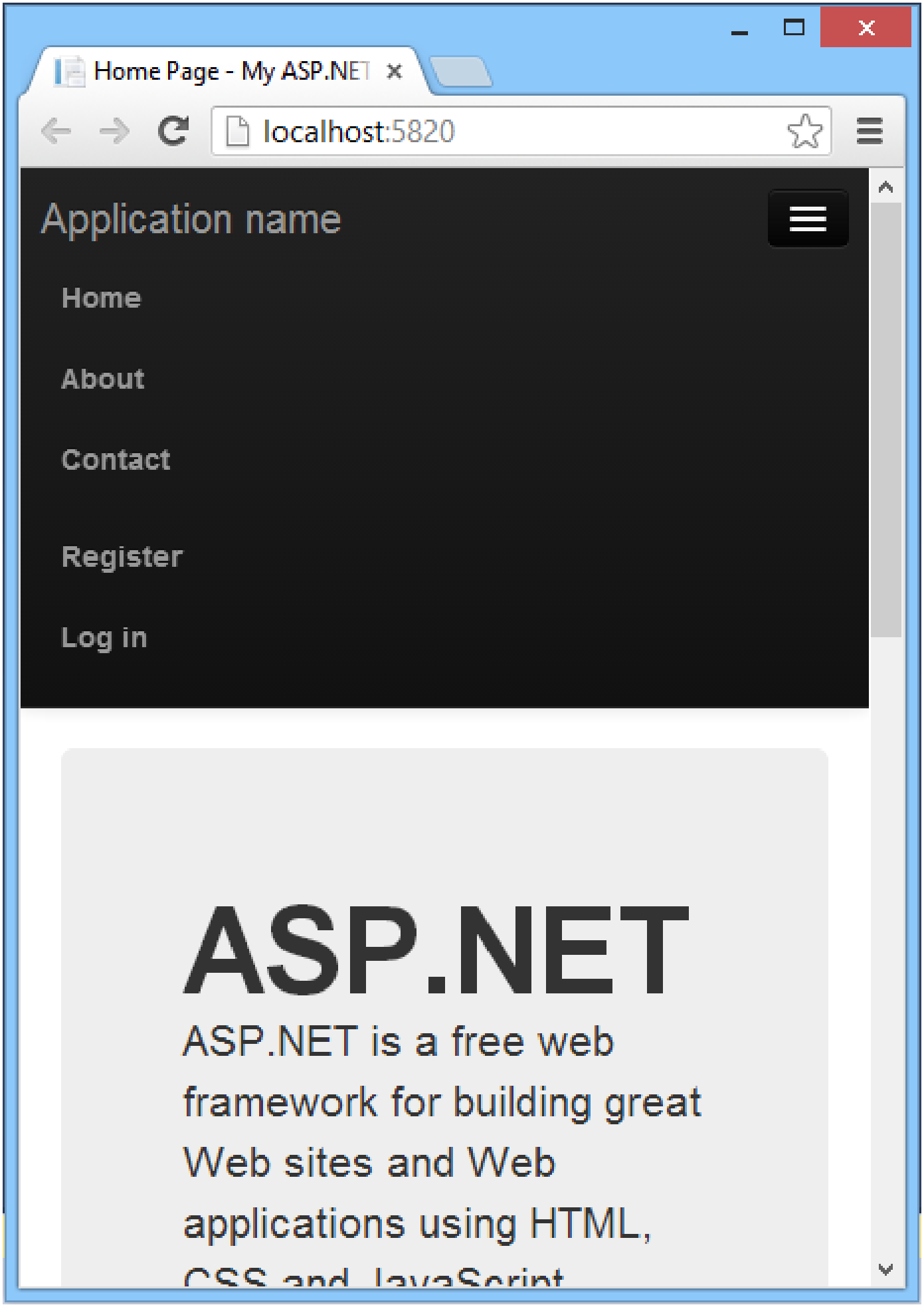


با توجه به پوشه‌ی مدل این را متوجه می‌شویم که مایکروسافت هم به **لزوم ViewModel** اعتقاد پیدا کرده است.

با اجرا کردن پروژه **bootstrap** و **responsive** بودن آن، خودنمایی می‌کنند.


Home Page - My ASP.NET
localhost:5820
Application name
Home
About
Contact
Register
Log in
ASP.NET
ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.
Learn more »
Getting started
ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and gives you full control over markup for
Get more libraries
NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects.
Web Hosting
You can easily find a web hosting company that offers the right mix of features and price for your applications.


Home Page - My ASP.NET
localhost:5820
Application name
ASP.NET
ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.
Learn more »
Getting started

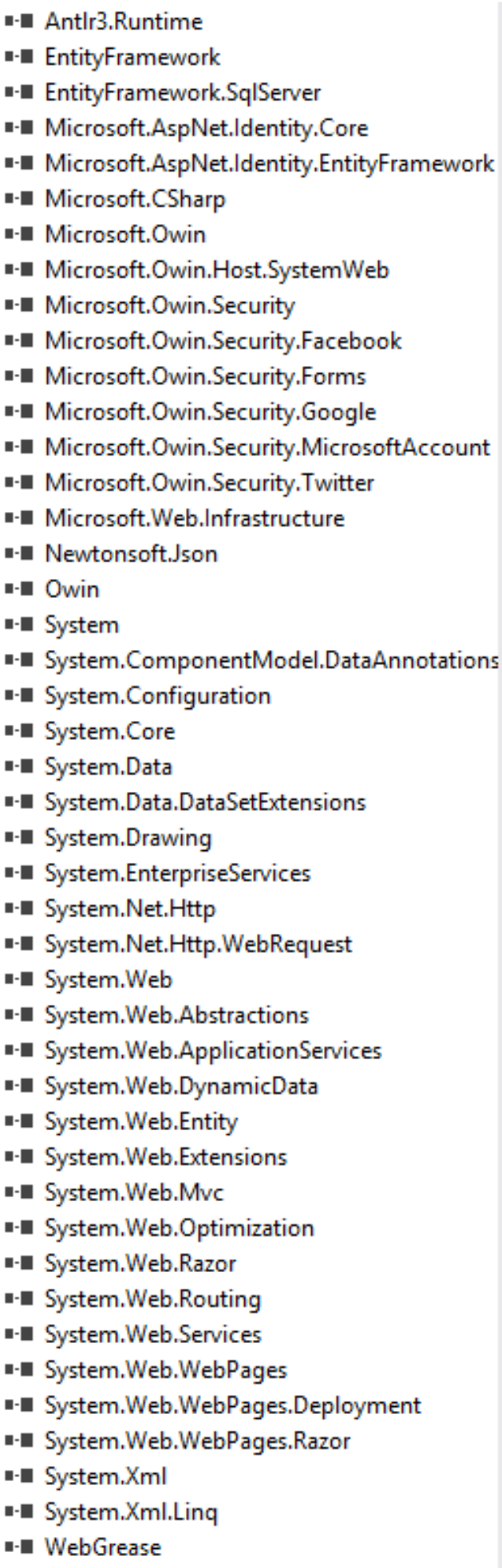

Home Page - My ASP.NET
localhost:5820
Application name
Home
About
Contact
Register
Log in
ASP.NET
ASP.NET is a free web framework for building great Web sites and Web applications using HTML,

اگر نگاهی به کنترلر Account بیندازیم، با موارد جالبی روبرو می‌شویم.

```
//
// POST: /Account/Register
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        try
        {
            // Create a profile, password, and link the local login before signing in the user
            User user = new User(model.UserName);
            if (await Users.Create(user) &&
                await Secrets.Create(new UserSecret(model.UserName, model.Password)) &&
                await Logins.Add(new UserLogin(user.Id, IdentityConfig.LocalLoginProvider, model.UserName)))
            {
                await SignIn(user.Id, isPersistent: false);
                return RedirectToAction("Index", "Home");
            }
            else
            {
                ModelState.AddModelError(String.Empty, "Failed to create login for: " + model.UserName);
            }
        }
        catch (DbEntityValidationException e)
        {
            ModelState.AddModelError("", e.EntityValidationErrors.First().ValidationErrors.First().ErrorMessage);
        }
    }

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

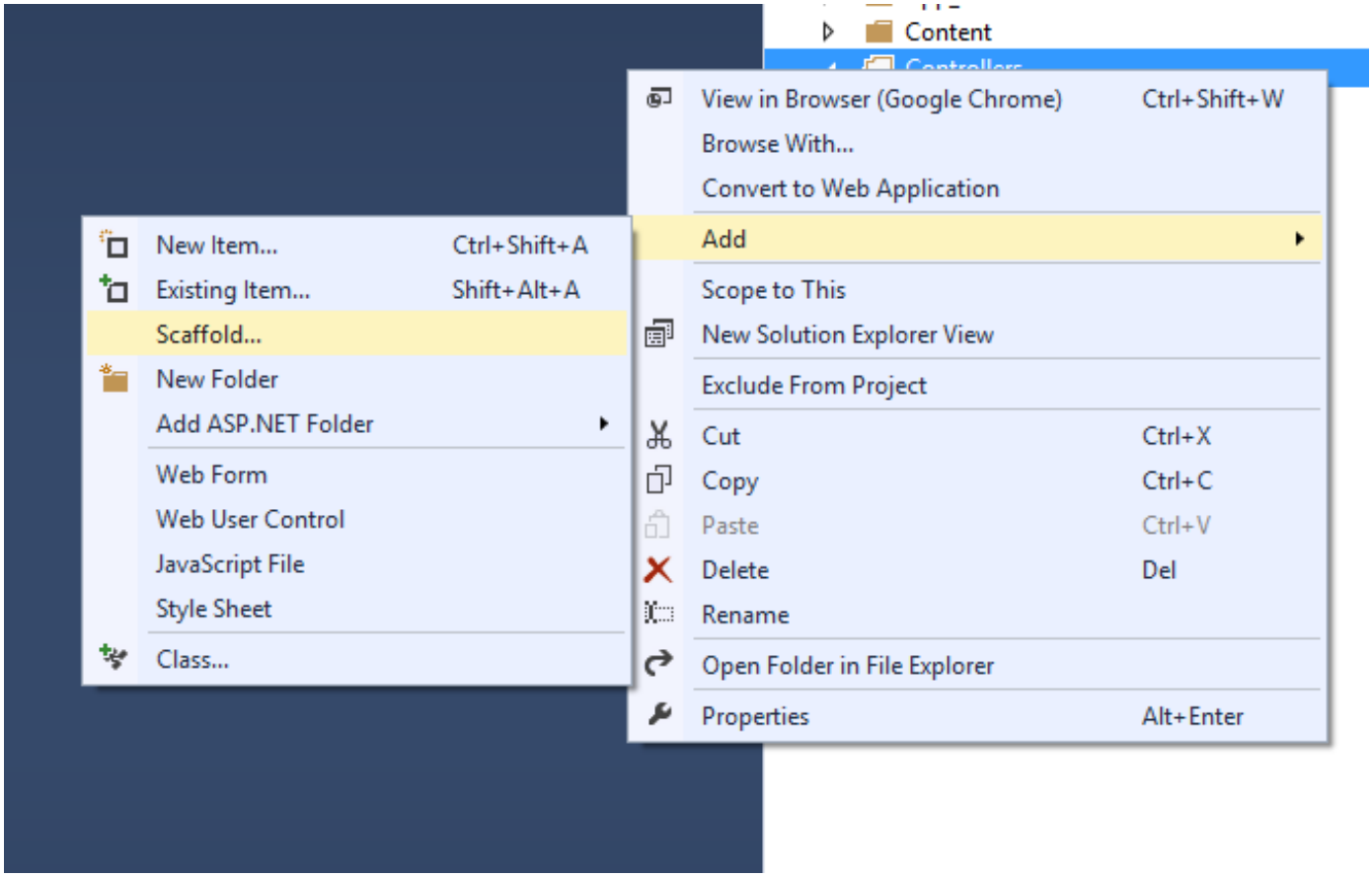
به لطف **سیستم Identity جدید** ، Entity Framework 6 و .NET 4.5 ، می‌بینیم که تا حد امکان، عملیات به صورت **آسنکرون(نامتقارن)** انجام شده اند که برای برنامه‌های scalable بسیار مفید و ضروری به نظر می‌رسد.

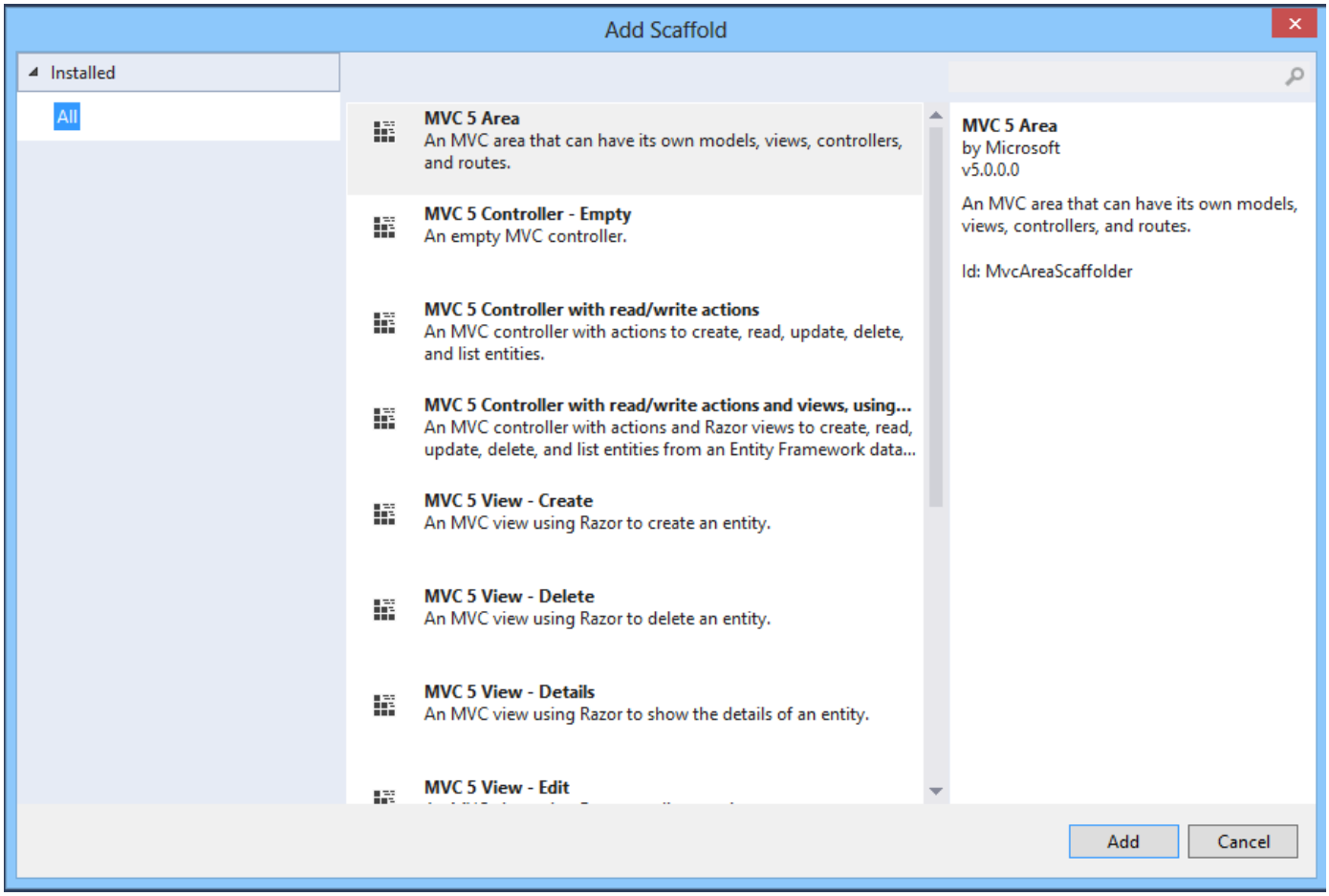
اگر نگاهی به referenceهای پروژه هم بیندازیم، حضور بسیاری از کتاب خانه‌های نام آشنا را به صورت پیش فرض، شاهد هستیم.


Antlr3.Runtime
EntityFramework
EntityFramework.SqlServer
Microsoft.AspNet.Identity.Core
Microsoft.AspNet.Identity.EntityFramework
Microsoft.CSharp
Microsoft.Owin
Microsoft.Owin.Host.SystemWeb
Microsoft.Owin.Security
Microsoft.Owin.Security.Facebook
Microsoft.Owin.Security.Forms
Microsoft.Owin.Security.Google
Microsoft.Owin.Security.MicrosoftAccount
Microsoft.Owin.Security.Twitter
Microsoft.Web.Infrastructure
Newtonsoft.Json
Owin
System
System.ComponentModel.DataAnnotations
System.Configuration
System.Core
System.Data
System.Data.DataSetExtensions
System.Drawing
System.EnterpriseServices
System.Net.Http
System.Net.Http.WebRequest
System.Web
System.Web.Abstractions
System.Web.ApplicationServices
System.Web.DynamicData
System.Web.Entity
System.Web.Extensions
System.Web.Mvc
System.Web.Optimization
System.Web.Razor
System.Web.Routing
System.Web.Services
System.Web.WebPages
System.Web.WebPages.Deployment
System.Web.WebPages.Razor
System.Xml
System.Xml.Linq
WebGrease

Entity Framework نسخه‌ی 6 beta1 به صورت پیش فرض در پروژه وجود دارد. **خوشبختانه دیگر خبری هم از System.Data.Entity نیست.** همچنین حضور پررنگ **Owin** و **ASP.NET Identity** را متوجه خواهید شد.

خب قبلا اگر قصد افزودن کنترلر جدیدی به بروژه داشتید، به راحتی در هر جای پروژه گزینه ای به نام AddController را می‌زدید. اما اینجا سناریو کمی متفاوت است.

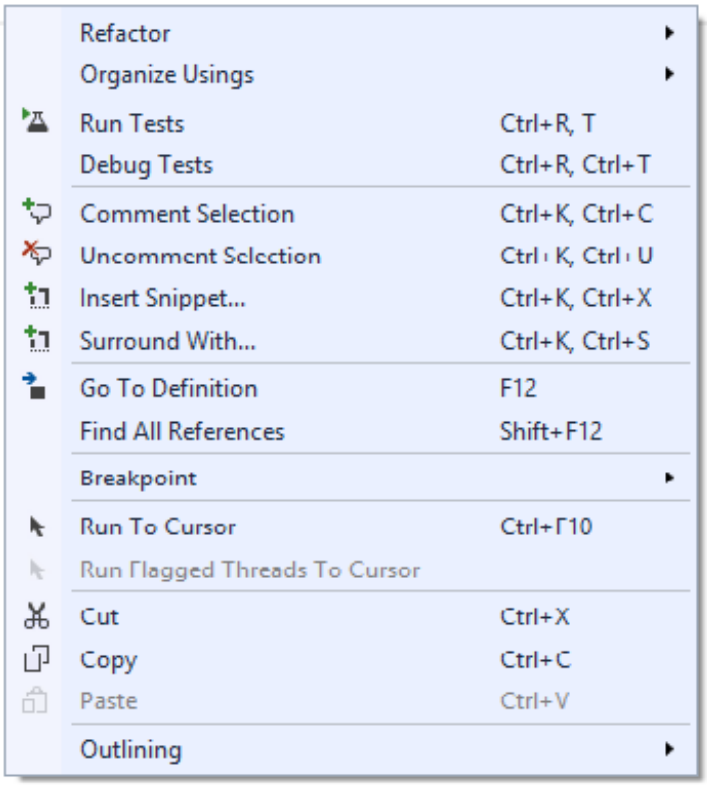




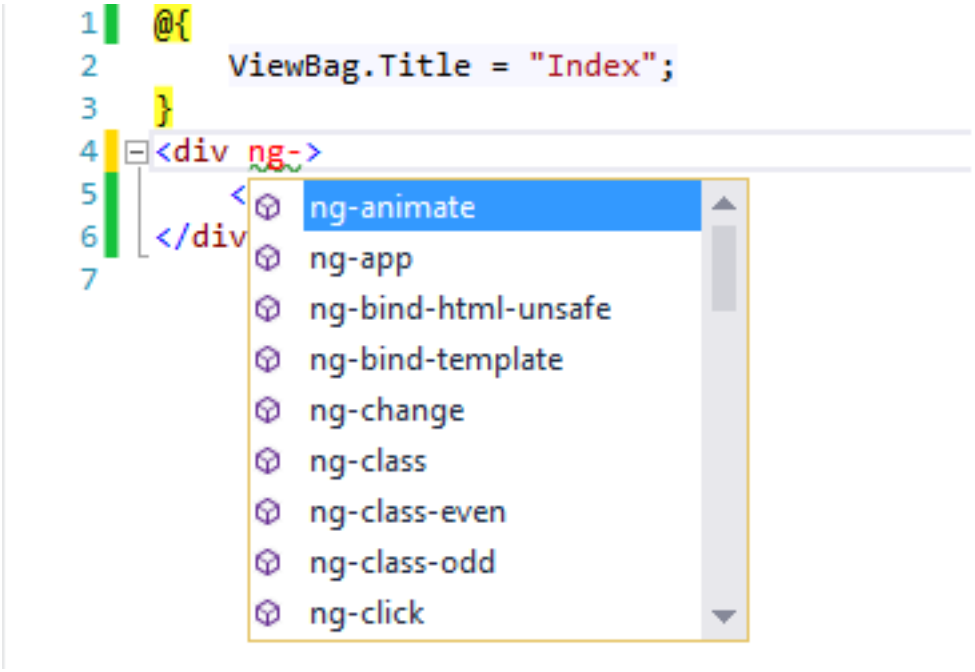
همه چیز گویا با **Scaffolding** یکپارچه شده. به گفته‌ی تیم ASP.NET، Scaffold کاملا از نو نوشته شده، در بررسی‌های اولیه من، کدهای تولیدی چندان تفاوتی با نسخه‌ی قبل نداشت.احتمالا تغییرات در جای دیگری است.

حال اگر بخواهیم برای Controller ایجاد شده یه View ایجاد کنیم باید طبق روال سابق از شرتکات ctrl+v, ctrl+m استفاده کنیم. اما...

```
namespace WebApplication5.Controllers
{
    public class HelloWorldController : Controller
    {
        //
        // GET: /HelloWorld/
        public ActionResult Index()
        {
            return View();
        }
    }
}
```



بله، لااقل در این IDE در اقدامی ناجوانمردانه! این گزینه حذف شده است. گویا باید وارد فولدر Views شده و به صورت دستی فولدر جدیدی ایجاد و از گزینه‌ی Scaffold برای افزودن View جدید اقدام کرد.



همین طور که می‌بینید، Intellisense ویژوال استادیو به صورت توکار، از **AngularJs** پشتیبانی می‌کند.

نتیجه گیری:

گویا مایکروسافت نیز به این نتیجه رسیده که ASP.NET MVC در نسخه‌ی سوم خود، کاملا پخته و به بلوغ رسیده است و پس از آن باید فقط آن را بهینه کرده و تغییرات اساسی در آن انجام ندهد .تیم ASP.NET تنها حواسش منعطف به همگام شدن با تکنولوژی‌های روز Web است و این را با پشتیبانی پیش فرض از bootstrap و Angularjs شاهد هستیم. گویا خط مشی تیم توسعه دهنده نیز این گونه است. از جمله تغییرات خوب بحث Identity هست که کاملا به EF Code First یک پارچه هست و دیگر مشکلات کار با سرویس Memberships وجود ندارد. در کل شما یک سری اینترفیس پیاده سازی می‌کنید و بقیه مسائل توسط این کتابخانه‌ی نو ظهور مدیریت می‌شود و فراموش نکنیم که این کتابخانه با OAuth یکپارچه است.(دریافت اطلاعات بیشتر)

طبق change-log رسمی، تغییر آنچنانی در MVC رخ نداده است. فقط یک سری آپدیت و بهینه سازی و همگام سازی با تکنولوژی‌های جدید.

نظر شما در مورد این تغییرات چیست؟

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۱۱ ۰:۲۷

- نکات مهم Bootstrap رو ما در سایت جاری بررسی کردیم و الزاما برای استفاده از آن نیازی به MVC5 نیست. همین الان در MVC4 هم می‌تونید ازش استفاده کنید. ولی درکل هر وقت مایکروسافت دست روی چیزی می‌گذارد، مزیتش تهیه حداقل 20 جلد کتاب جدید در مورد CSS و Bootstrap و طراحی است که در نهایت برای دنیای وب، از لحاظ بالا رفتن کیفیت کارهای انجام شده، بسیار مفید خواهد بود.
- در کل این به روز رسانی برای مدیریت و دریافت تغییرات انجام شده اخیر بسیار مناسب خواهد بود (تمام اجزای MVC مانند اسکریپت‌های اعتبارسنجی سازگار با نسخه جدید jQuery، فشرده سازهای CSS و JS، قسمت‌های مرتبط با SignalR و Web API همین Owin ایی که نامبردید، مرتبا به روز می‌شوند). حداقل دیگر نیازی به دریافت چند گیگ به روز رسانی VS 2012 نیست و به یکباره می‌شود تمام آن‌ها را در VS 2013 داشت.
- همچنین با توجه به سورس باز بودن MVC، دنبال کردن History سورس کنترل آن‌ها در جهت مشاهده تغییرات انجام شده ضروری است. یعنی صرفا نباید در منوها یا صفحه دیالوگ‌های جدید به دنبال تغییرات بود. اگر تغییرات سورس کنترل را بررسی کنید مواردی مانند MVC Attribute Routing، رفع تعدادی از باگ‌های Razor parser و تغییرات گسترده‌ای در Web API انجام شده (بیشتر موارد مرتبط به Web API است).

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۲۳ ۹:۲۵

دمویی از تغییرات انجام شده در MVC5 و Web API 2

Jon Galloway: Bleeding edge ASP.NET: See what is new and next for MVC, Web API, SignalR and more

دریافت از اینجا: (^)

نویسنده: ابوالفضل
تاریخ: ۱۳۹۲/۰۹/۱۰ ۱۶:۵

با سلام
با توجه به تغییرات سیستم امنیتی mvc در نگارش 4 که از وب ماتریکس استفاده می‌کرد و در دات نت که بحث owin و غیره مطرح هست، یه مشکلی که وجود داره ، ساخت یه سری کلاس زیربنایی هست که اصطلاحا به فریم ورک تعبیر میشه. اگر بخوایم مثلا برای قسمت زیربنایی نام کاربری رو داشته باشیم، چه روشی رو پیشنهاد می‌کنید؟
مثلا در mvc 4 من از وب ماتریکس WebMatrix.WebData.WebSecurity.CurrentUserName استفاده میکردم، ولی الان با mvc 5 نال میشه و مقدار نداره.

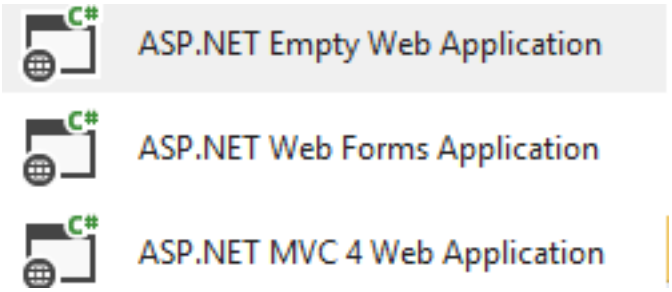
نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۹/۱۰ ۱۸:۳

روش web matrix آنچنان مقبولیتی پیدا نکرده (^). بهتره از روش پروژه سورس باز IRIS استفاده کنید که مبتنی است بر Forms Authentication و مباحث Roles را هم داره.

نویسنده: منصور جعفری
تاریخ: ۱۳۹۲/۱۲/۲۹ ۱۲:۷

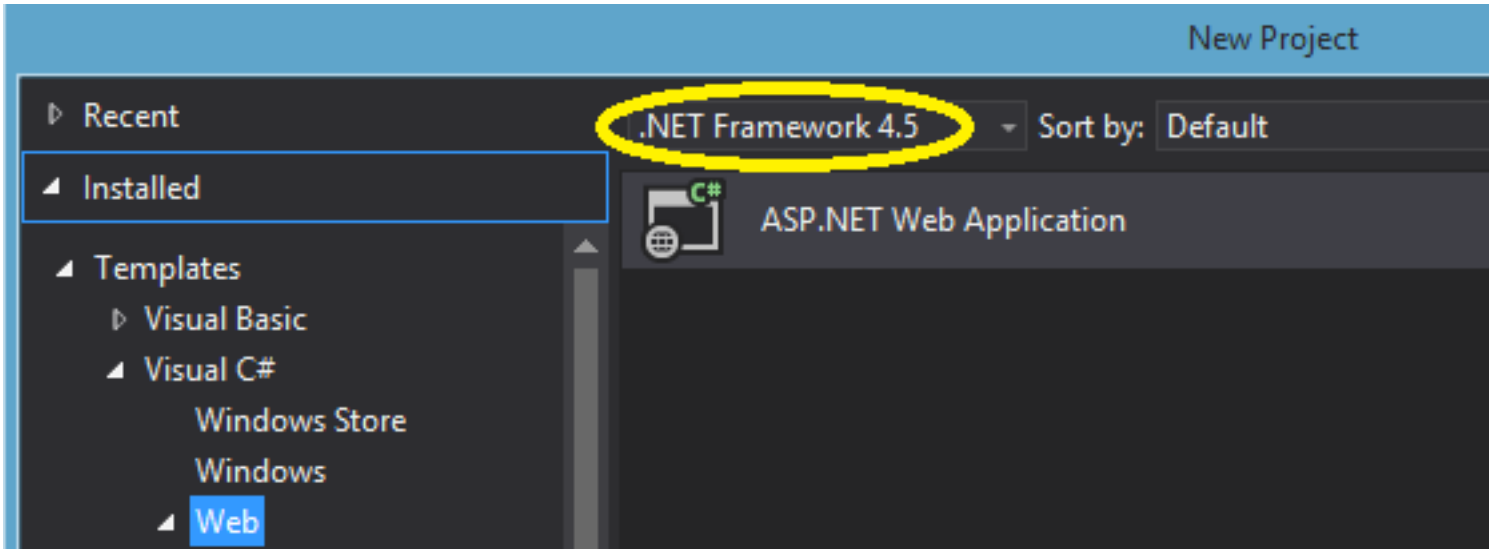
سلام

چطور میشه مثل MVC4 یک پروژه خالی از MVC 5 به وجود آورد و یا به‌طور کل آیا خود ویژوال استادیو MVC 5 رو بصورت پیش‌فرض داخل خودش نداره ؟
یعنی به این‌شکل که MVC 4 وجود داره MVC5 به ویژوال استادیو اضافه بشه؟

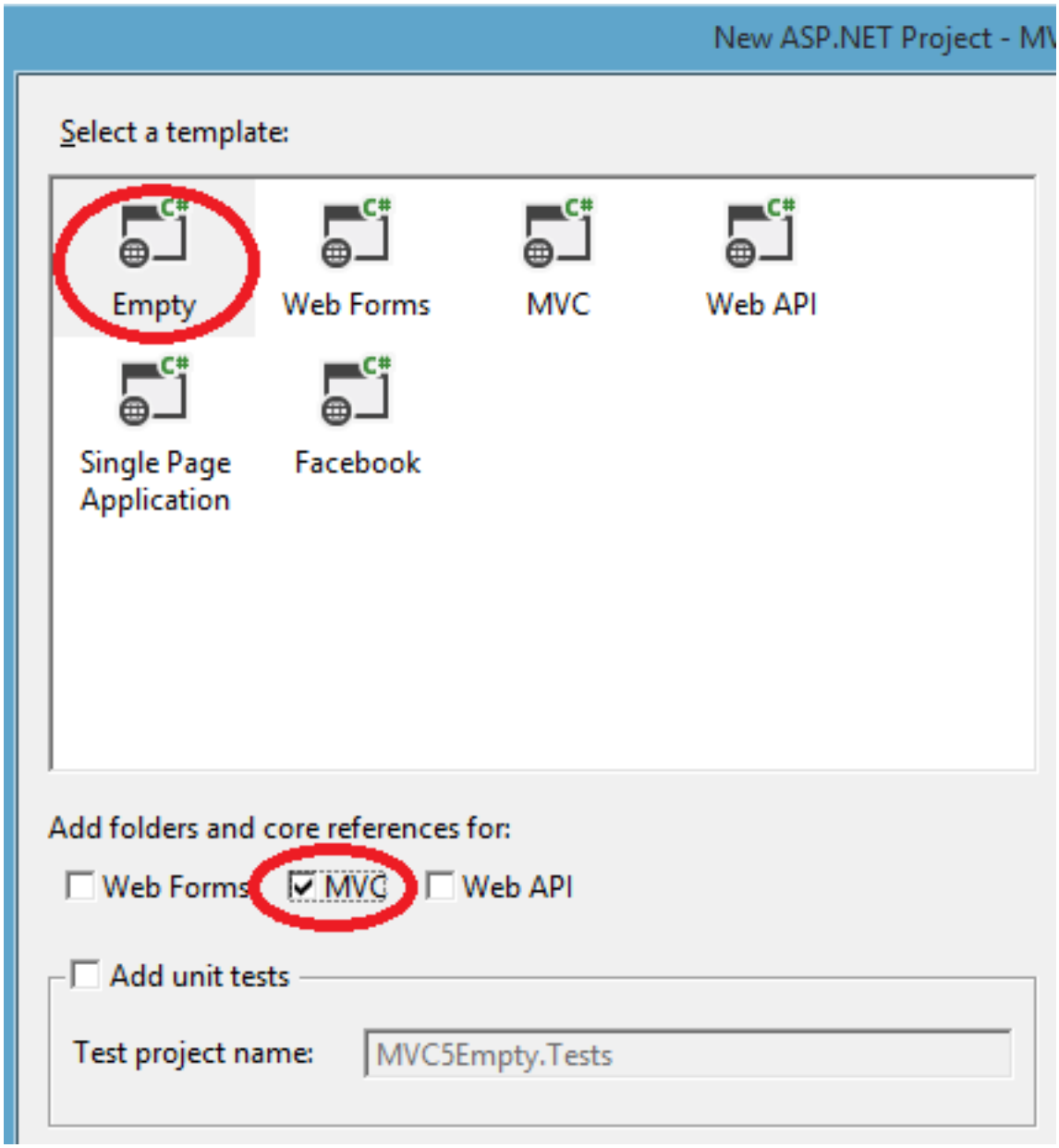


نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۲۹ ۱۲:۵۱

در VS 2013 به همراه آخرین به روز رسانی‌ها:
- انتخاب دات نت 4.5 یا 4.5.1 به معنای کار با MVC 5.x است:



سپس در صفحه‌ی ظاهر شده، امکان انتخاب گزینه‌ی خالی نیز هست:


New ASP.NET Project - MV
Select a template:
Empty
Web Forms
MVC
Web API
Single Page Application
Facebook
Add folders and core references for:
Web Forms
MVC
Web API
Add unit tests
Test project name:
MVC5Empty.Tests

عنوان: **قابلیت Attribute Routing در ASP.NET MVC 5**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۲/۰۹/۱۷ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.NET MVC 5, routing, Attibute Routing, MVC

در ASP.NET MVC 5 یک قابلیت جدید با نام Attribute Routing افزوده شده است که به ما این اجازه را می‌دهد تا Routeهای سفارشی برای کنترلرها و اکشن متدهایمان با اضافه کردن یک Attribute با نام Route تعریف کنیم.
همچنین می‌توانیم ویژگی RoutePrefix نیز برای کنترلرهایمان تعریف کنیم تا همه‌ی اکشن متدها نیز از آن پیروی کنند. این ویژگی را با ذکر یک مثال معرفی میکنیم :

ابتدا لازم است این ویژگی را در کلاس RouteConfig فعال کنیم :

```
public static void RegisterRoutes(RouteCollection routes)
{
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

    routes.MapMvcAttributeRoutes();

    // ...
}
```

قدم بعدی تنها افزودن Attributeهای ذکر شده به کنترلر و اکشن متدهایمان می‌باشد، به طور مثال ما در اینجا یک کنترلر با نام ProductController ایجاد کرده ایم و کنترلر را با ویژگی RoutePrefix مزین کرده ایم که در این حالت به ASP.NET MVC می‌گویم که تمام اکشن متدهای داخل این کنترلر با products شروع شوند :

```
[RoutePrefix("products")]
public class ProductsController : Controller
{
    public ProductsController()
    {
    }

    [Route]
    public ActionResult Index()
    {
        return View();
    }
}
```

همانطور که در کد فوق ملاحظه می‌کنید اکشن متد Index را با افزودن ویژگی Route که آدرس products/~ را تطبیق می‌دهد تعیین کرده ایم.

نحوه تعیین Optional URI Parameter :
کافی است علامت سوال را به آخر پارامتر اضافه کنیم :

```
[Route("{id?}")]
public ActionResult Index(int id)
{
   return View();
}
```

نحوه تعیین Default Route :

```
[RoutePrefix("products")]
[Route("{action=index}")]
public class ProductsController : Controller
{
    public ProductsController()
    {
    }
    public ActionResult Index()
    {
```

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

نحوه تعیین Constraint برای Routeها :

```
[Route("{id:int}")]
public ActionResult Delete(int id)
{
   return View();
}
```

در مثال فوق گفته ایم که Id باید از نوع عدد صحیح باشد در غیر اینصورت آن را تطبیق نمی‌دهد.
همچنین می‌توانید از عبارات Regex نیز استفاده کنید به طور مثال در کد زیر پارامتر title باید یک متن و یا عبارت فارسی باشد در غیر اینصورت تطبیقی صورت نمی‌گیرد:

```
[Route("{title:regex(\u0600-\u06FF)}")]
public ActionResult Search(string title)
{
   return View();
}
```

در لینکی که در بالا معرفی شده لیست کامل Constraintها را می‌توانید مشاهده نمائید،

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۹/۱۷ ۱۰:۱

با تشکر از شما. آیا در این حالت ذکر مسیریابی پیش فرض الزامی است؟ یعنی باید بالای تمام کنترلرها مثال default route شما را لحاظ کرد؟

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۲/۰۹/۱۷ ۱۰:۱۵

سلام،
اگر حالت فوق را فعال نکنید پیش فرض مسیر یابی همانند مسیریابی‌های قبلی در MVC 4 خواهد بود.

نویسنده: رضا گرمارودی
تاریخ: ۱۳۹۲/۰۹/۱۷ ۱۲:۱۵

ضمن تشکر از مطلب خوبتون از این قابلیت چگونه برای آدرس دهی اتوماتیک استفاده کرد؟

برای مثال در بالا اکشن Index یک پارامتر Id دارد . چطوری می‌توان با ذکر products و Id مستقیما به اکشن Index فوروارد بشه یعنی : http://localhost/products/10

این سوال و بدون قابلیت مذکور و در Mvc4 چطوری صحیح‌تر است انجام شود؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۹/۱۷ ۱۲:۳۵

این آدرس‌هی خودکار هست. نیاز به کار اضافه‌تری نداره؛ چون action پیش فرض مسیر رو مقدار دهی کرده (البته ذکر این attribute routing هم ضروری نیست؛ اگر مثل قبل یک default route پیش فرض تعریف شده باشه):

```
[Route("{action=index}")]
```

برای اطلاعات بیشتر این مطلب رو مطالعه کنید: asp-net-mvc-4

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۹/۱۷ ۱۳:۱

مطابق مقاله‌ای که ابتدای بحث لینک داده شده، امکان ترکیب هر دو حالت attribute routing و convention-based routing با هم وجود دارد و اگر در حالت قدیمی یعنی convention-based routing، شما مثل قبل یک default route تعریف کرده باشید، دیگر نیازی به ذکر و تکرار default route هم نام یک کنترلر در حالت attribute routing نخواهید داشت و فقط جایی که واقعا نیاز است باید از آن استفاده کرد. attribute routing کار رو برای تعریف قیدها یا constraints خیلی ساده و طبیعی کرده.

نویسنده: رشیدیان
تاریخ: ۱۳۹۳/۰۶/۱۲ ۱۲:۲۵

سلام و ممنون بابت انتشار این موضوع مفید.
یک سئوال داشتم: برای ایجاد زیر دامنه چه راهکاری هست؟
مثلا: forum.site.com

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۶/۱۲ ۱۲:۳۴

ساب دومین ارتباطی به مسیریابی نداره چون مسیریابی مرتبط هست به اجزایی از URL که بعد از دومین و پورت آن مشخص می‌شوند. همینقدر که ساب دومین در IIS تعریف شده (به همراه تنظیمات DNS آن؛ مثلا تنظیم فایل hosts)، سیستم مسیریابی خودش رو با اون انطباق می‌ده.

نویسنده: ندا
تاریخ: ۱۳۹۳/۰۶/۱۳ ۹:۲۱

سلام؛ با تشکر از توضیحاتتون. من نمی‌خوام که نام کنترلر و اکشن‌ها در url سایت نمایش داده شه.
من مثلا در اکانت کنترلر از [RoutePrefix("manageuser")] استفاده کردم و بالای اکشن موردنظرم از [Route] استفاده کردم ولی تنها تاثیرش این بود که با url پیش فرض قبلی به صفحه دسترسی نداشتم ولی به جای نام کنترلر manageuser هم وارد می‌کردم فایده نداشت. در ضمن در routes.MapMvcAttributeRoutes ، RegisterRoutes(); هم اضافه کردم ممنون میشم راهنماییم کنید

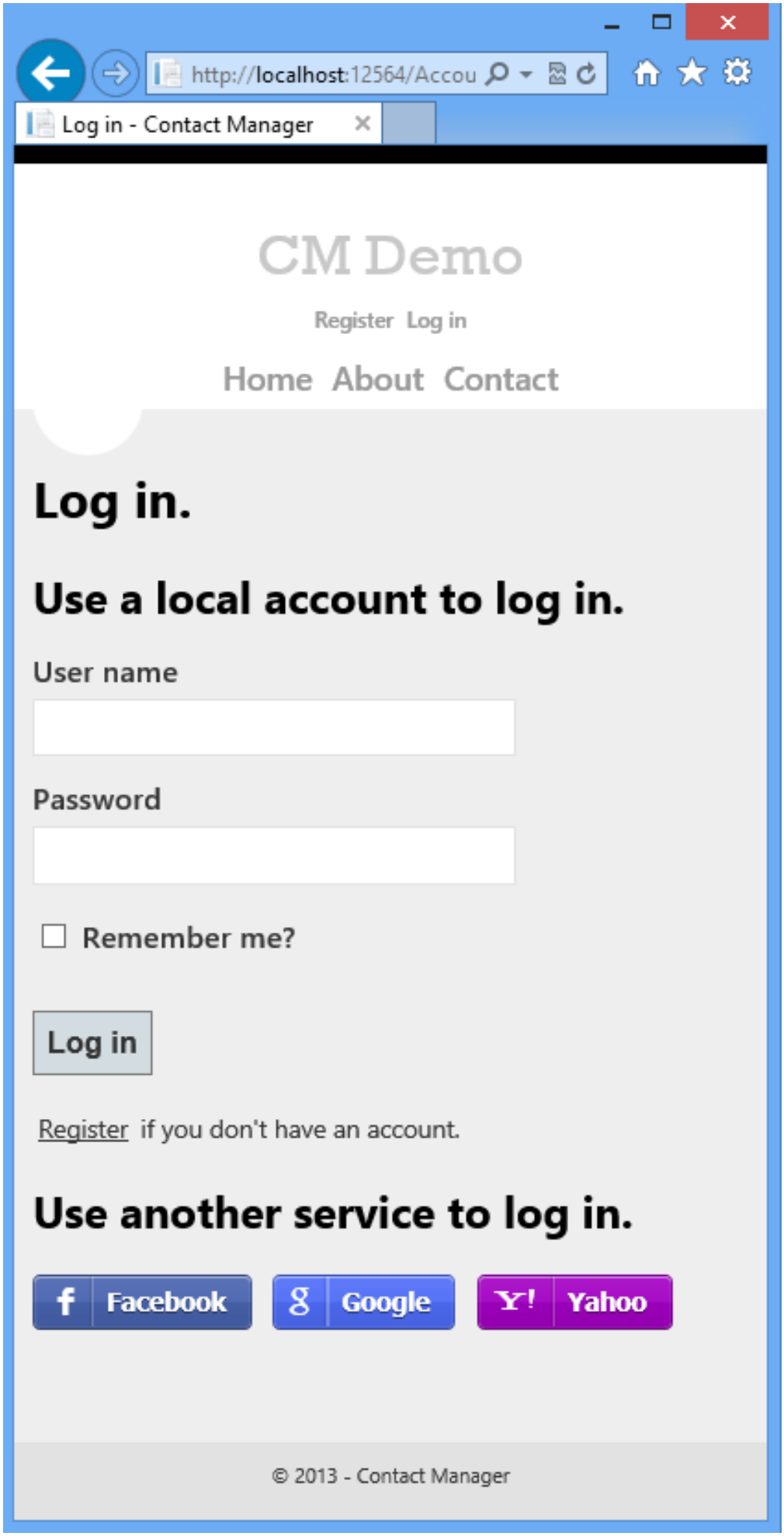
نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۶/۱۳ ۱۱:۳۱

دیباگش کنید تا مشخص شه کدوم مسیریابی اعمال شده: چطور مسیریابی‌های ASP.NET MVC را دیباگ کنیم؟

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

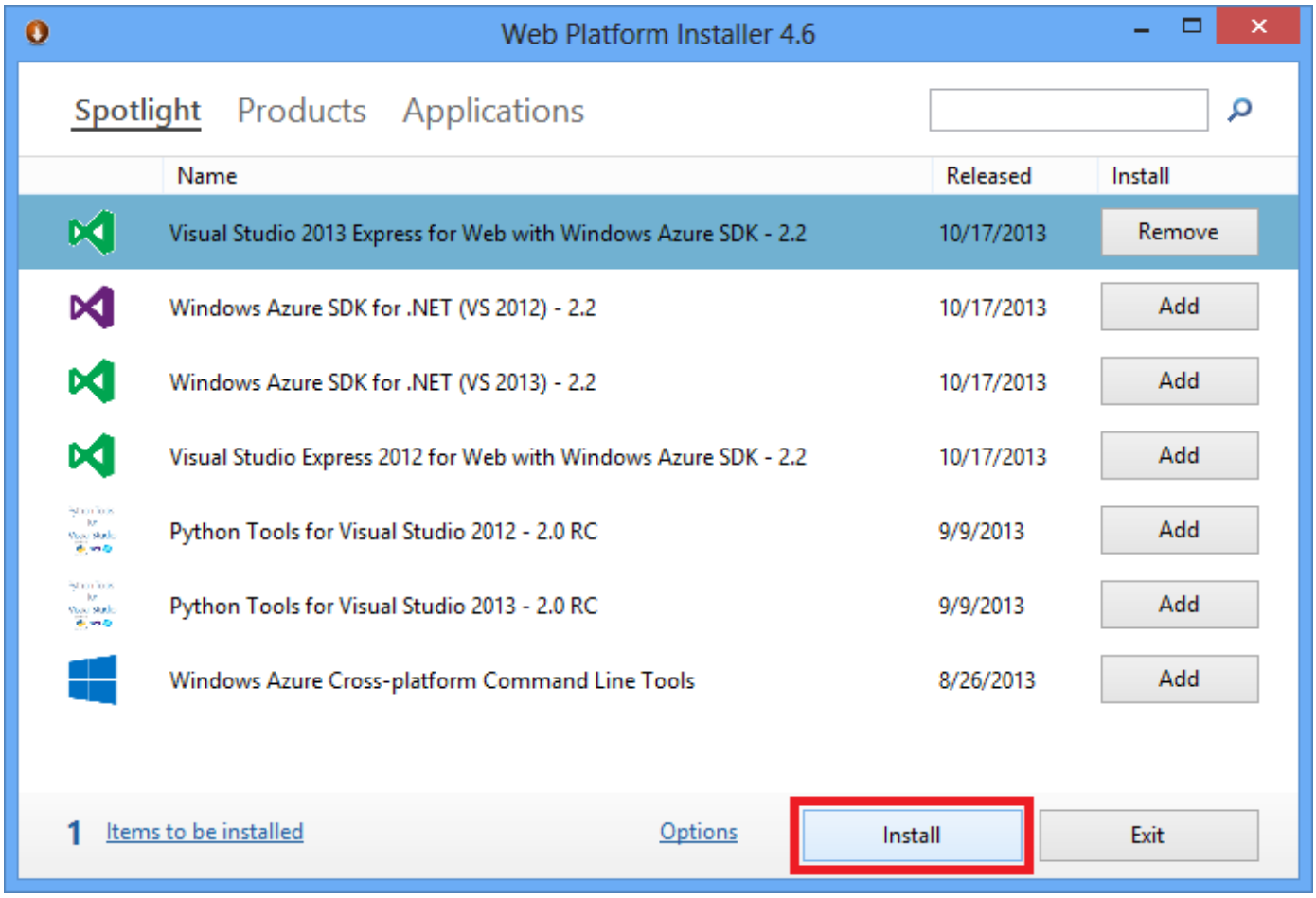
استفاده از Membership API

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

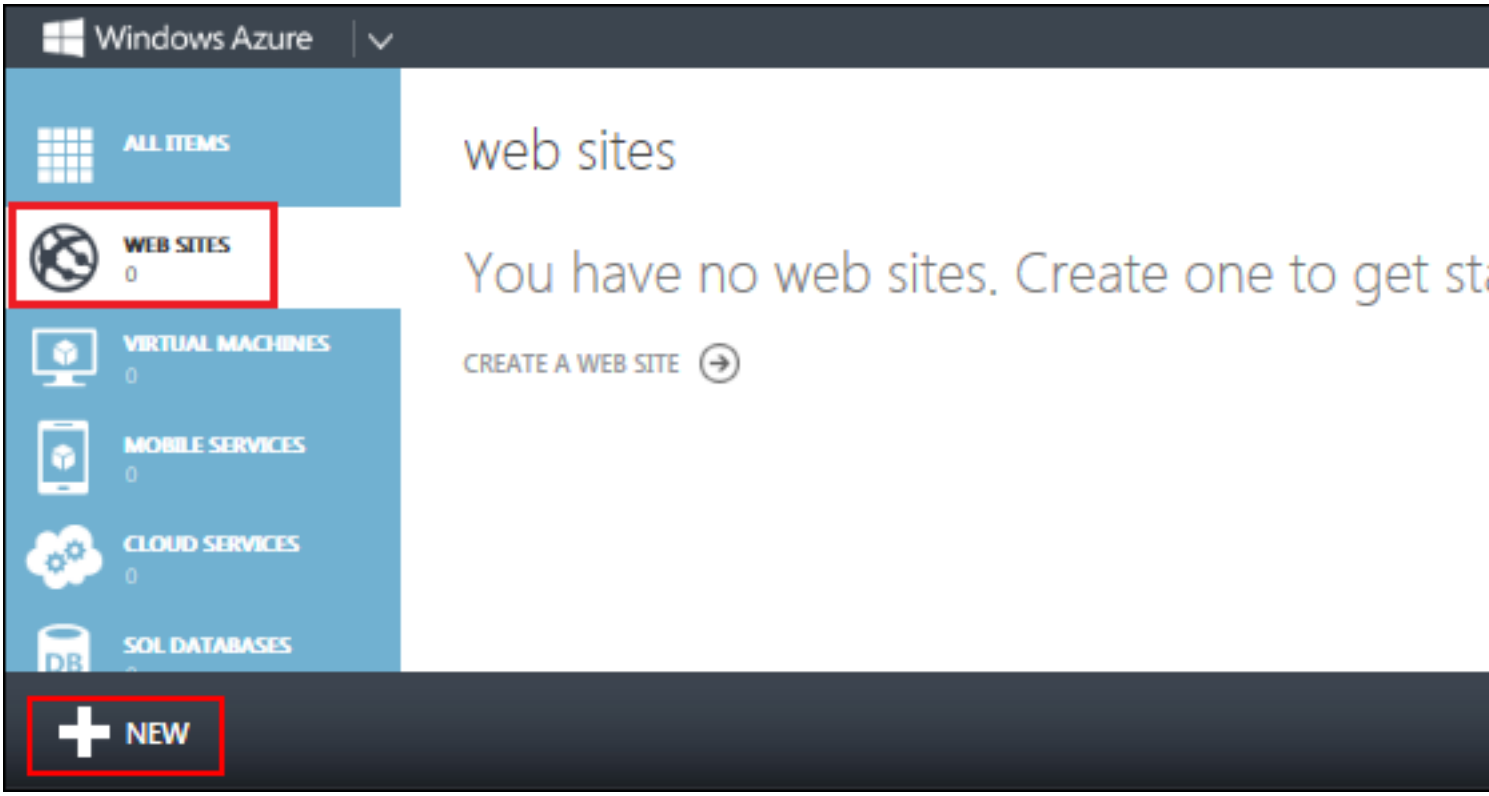
در قدم بعدی باید یک وب سایت Windows Azure و یک دیتابیس بسازیم.

ایجاد یک وب سایت و دیتابیس در Windows Azure

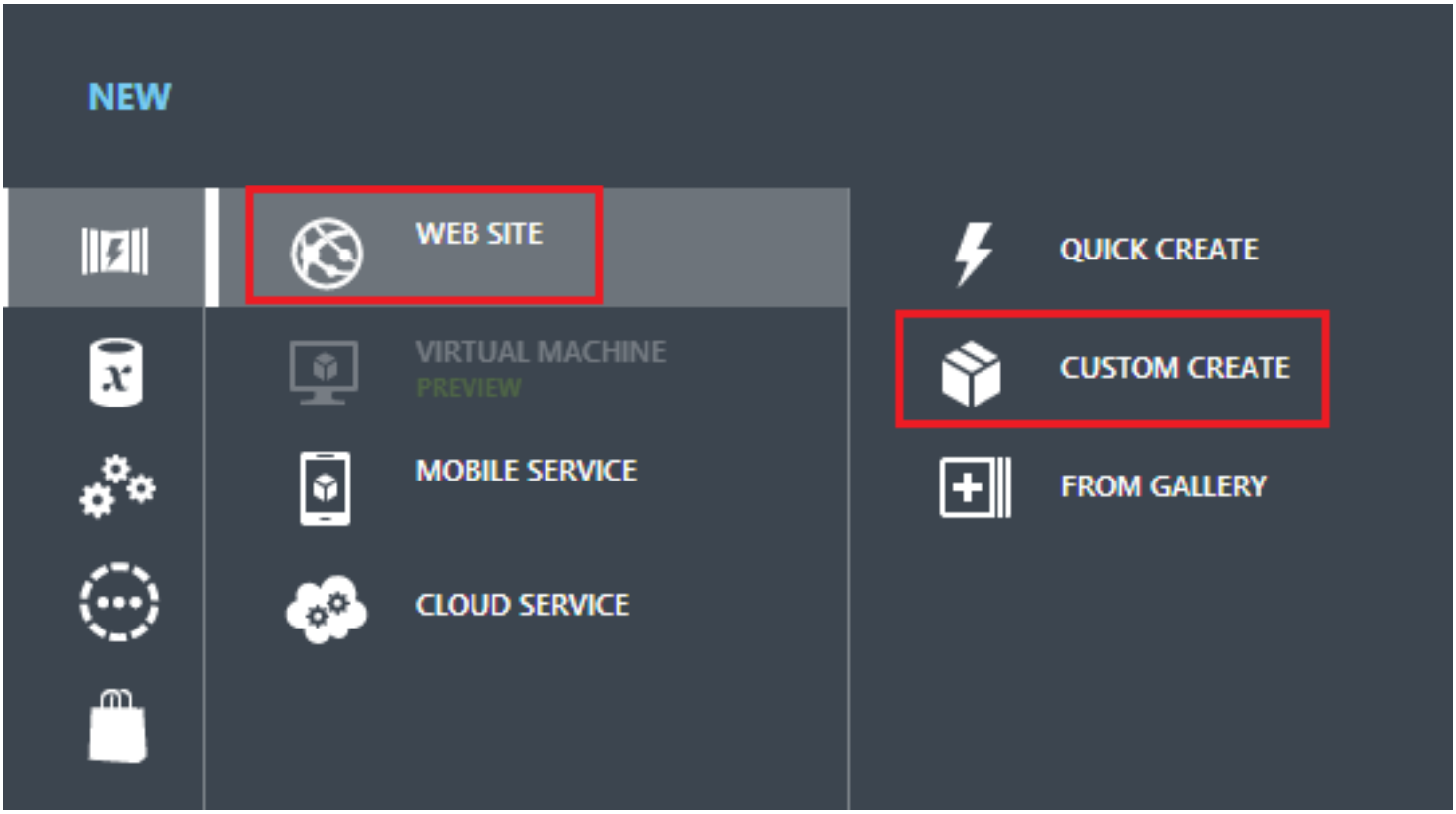
وب سایت Windows Azure شما در یک محیط اشتراکی (shared) میزبانی می‌شود، و این بدین معنا است که وب سایت‌های شما روی ماشین‌های مجازی (virtual machines) اجرا می‌شوند که با مشتریان دیگر Windows Azure به اشتراک گذاشته شده اند. یک محیط میزبانی اشتراکی گزینه ای کم هزینه برای شروع کار با رایانش‌های ابری است. اگر 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.

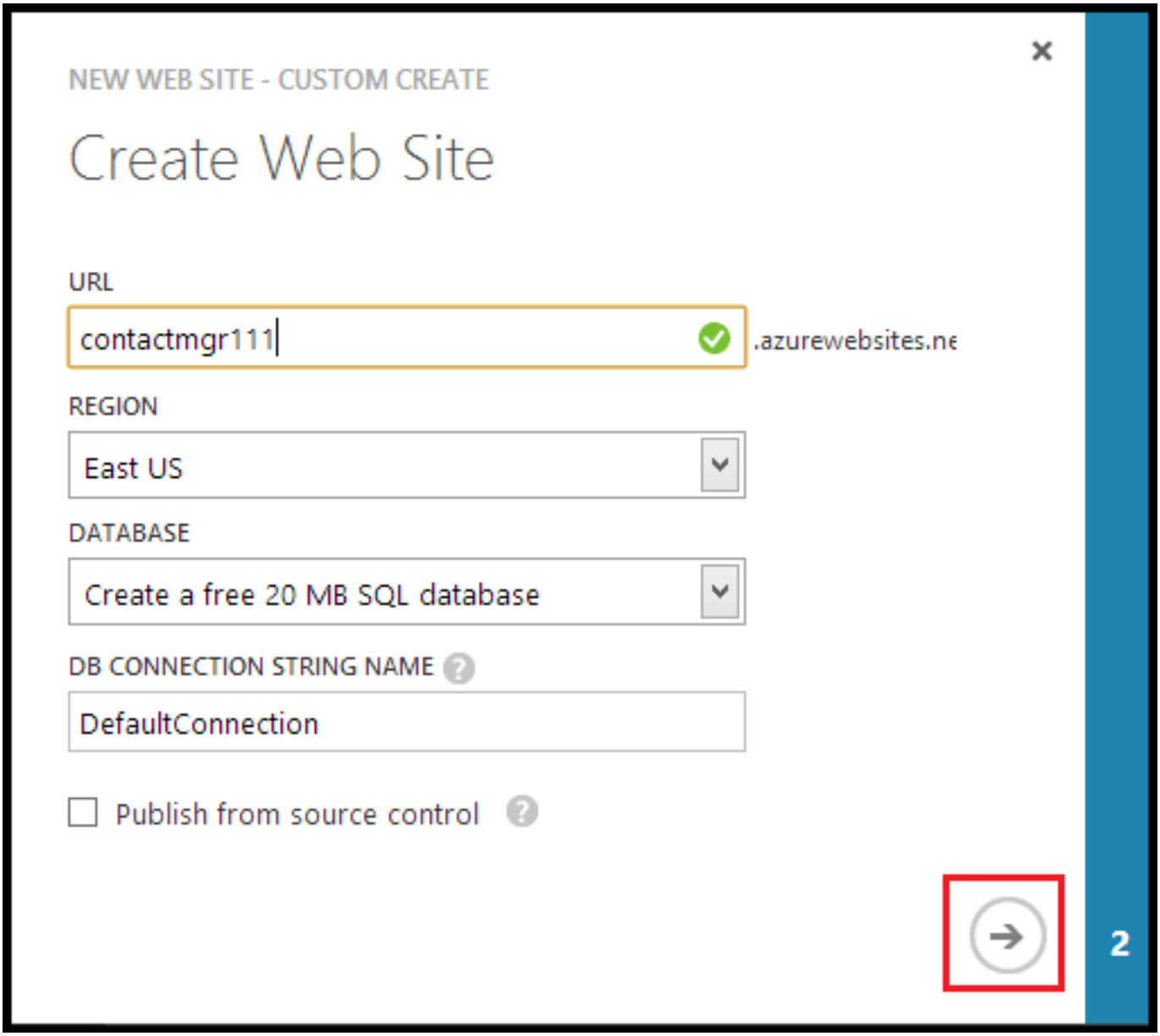
در پرتال مدیریتی Windows Azure روی **Web Sites** در قسمت چپ صفحه کلیک کنید، و گزینه **New** را برگزینید.



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.

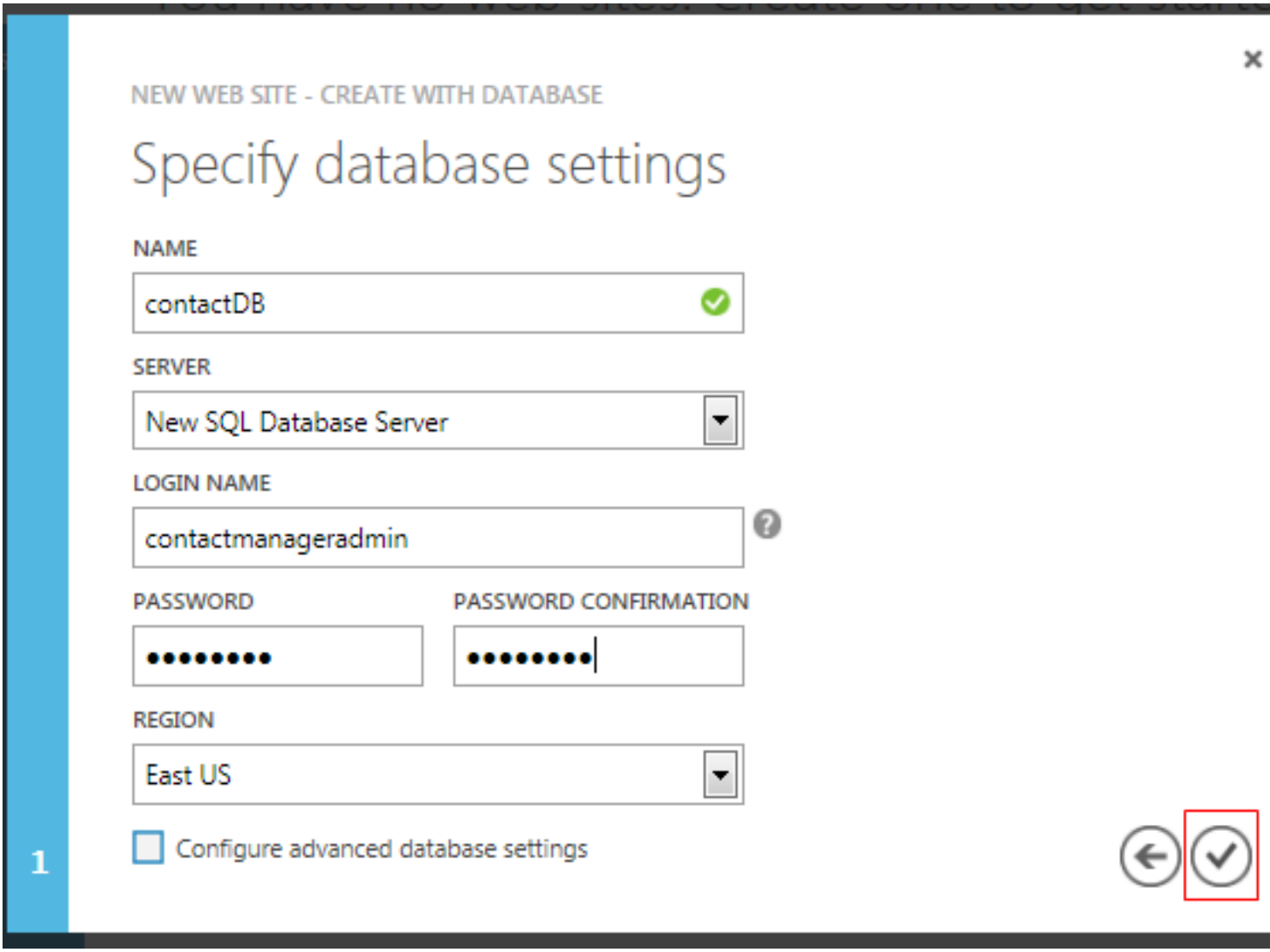
در قسمت **Name** مقدار *ContactDB* را وارد کنید (تصویر زیر).

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

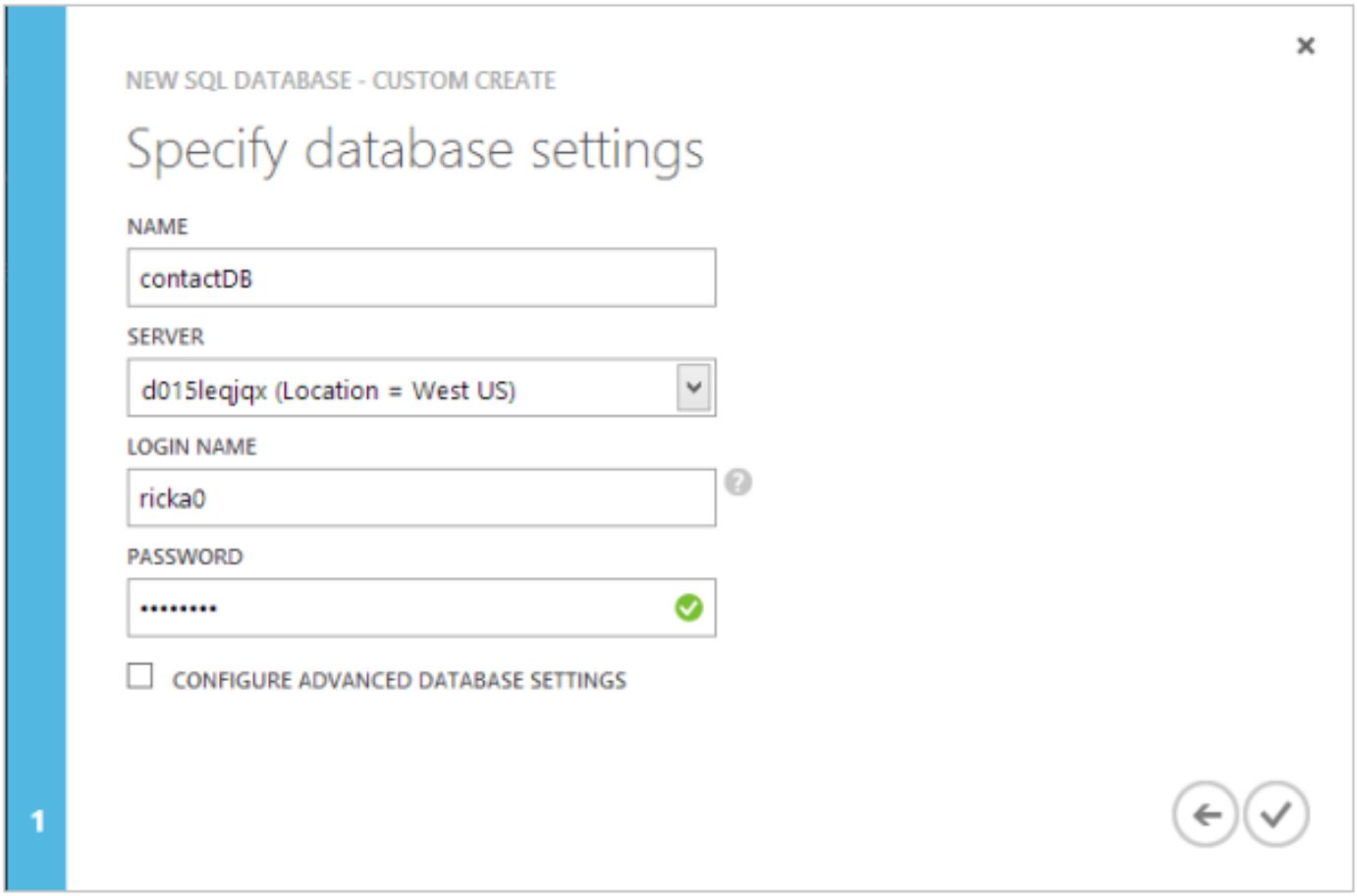
مقدار قسمت **Region** را به همان مقداری که برای ایجاد وب سایت تان تنظیم کرده اید تغییر دهید.

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.



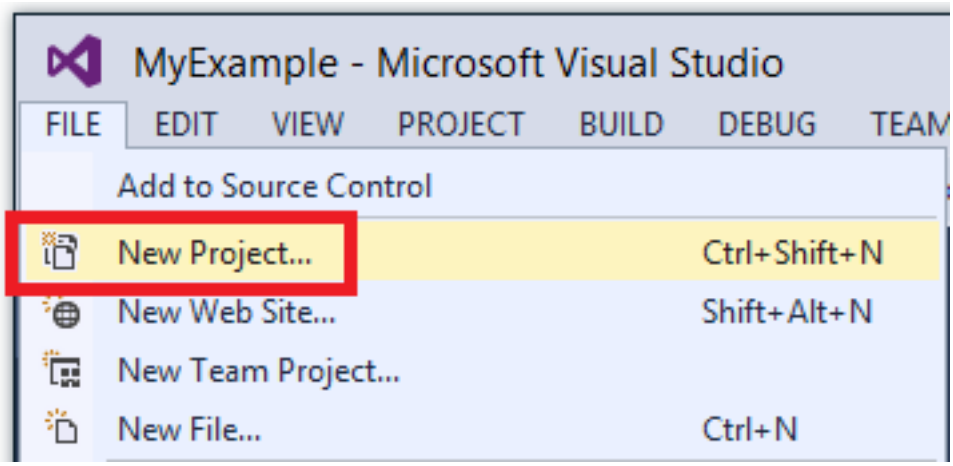
تصویر زیر استفاده از یک SQL Server و حساب کاربری موجود (existing) را نشان می‌دهد.



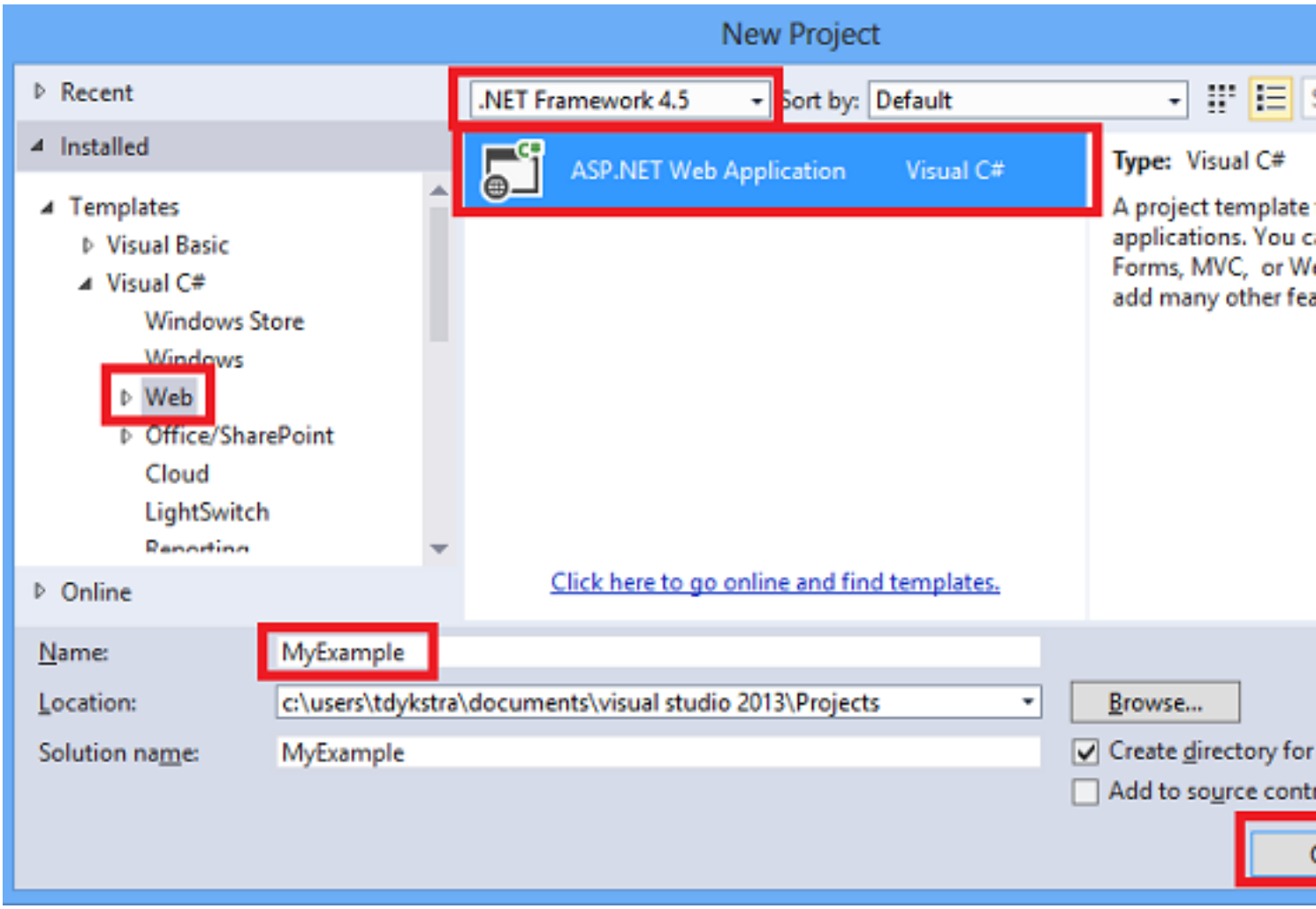
پرتال مدیریتی پس از اتمام مراحل، به صفحه وب سایت‌ها باز می‌گردد. ستون **Status** نشان می‌دهد که سایت شما در حال ساخته شدن است. پس از مدتی (معمولا کمتر از یک دقیقه) این ستون نشان می‌دهد که سایت شما با موفقیت ایجاد شده. در منوی پیمایش سمت چپ، تعداد سایت هایی که در اکانت خود دارید در کنار آیکون **Web Sites**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

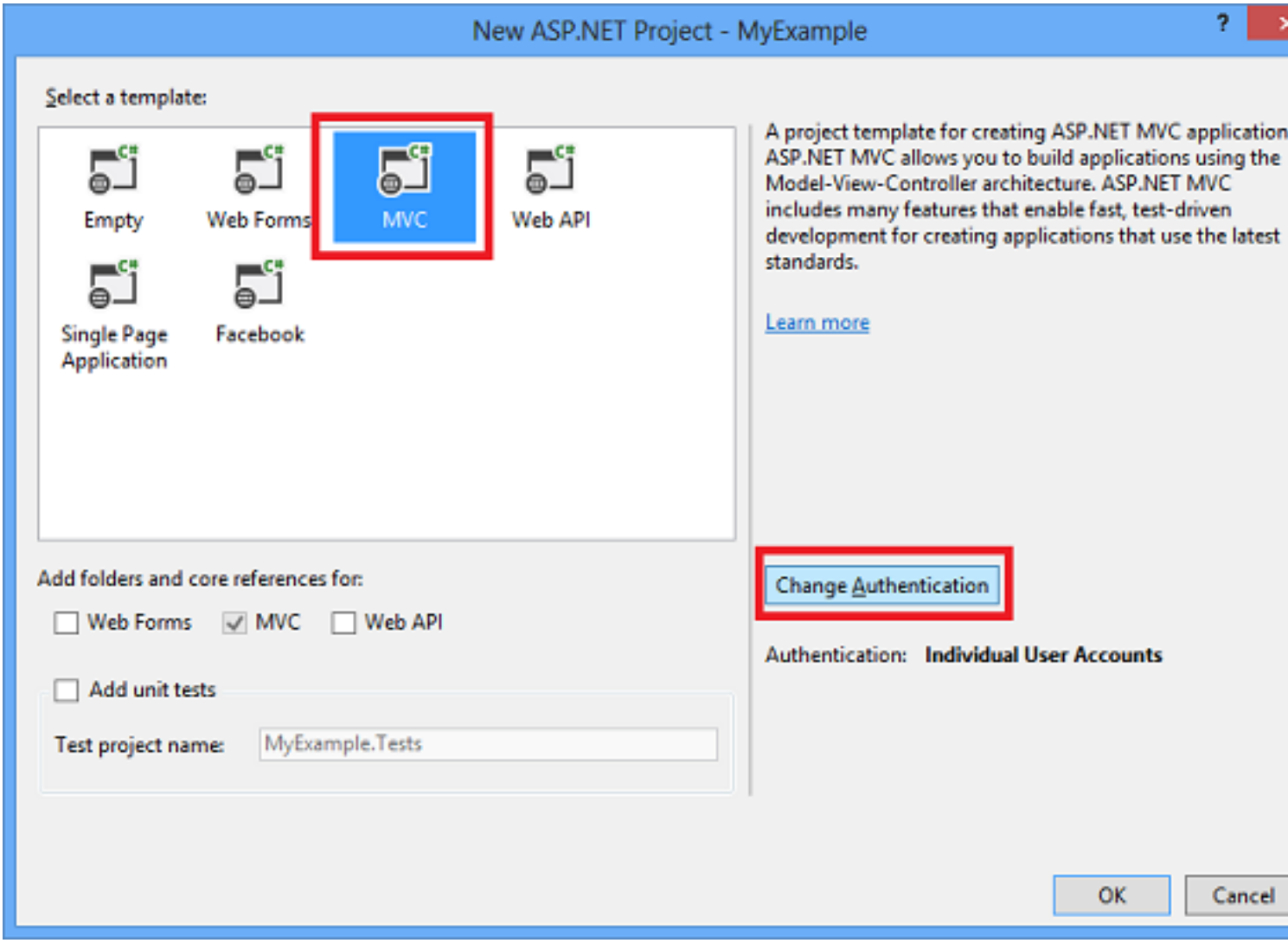


نوع پروژه را **ASP.NET Web Application** انتخاب کنید.



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.

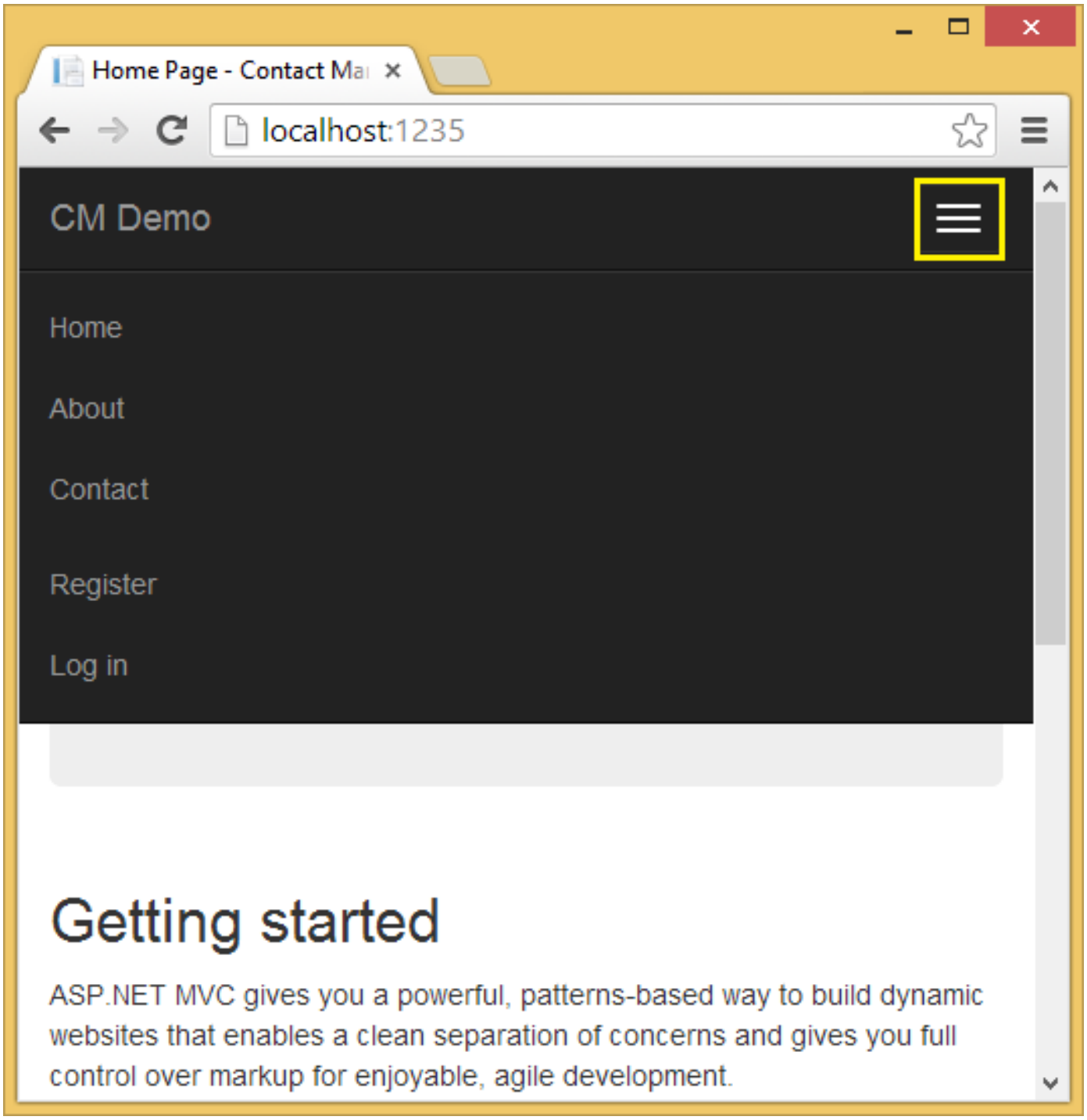
عبارت "Application name" را هم با "CM Demo" جایگزین کنید.

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.

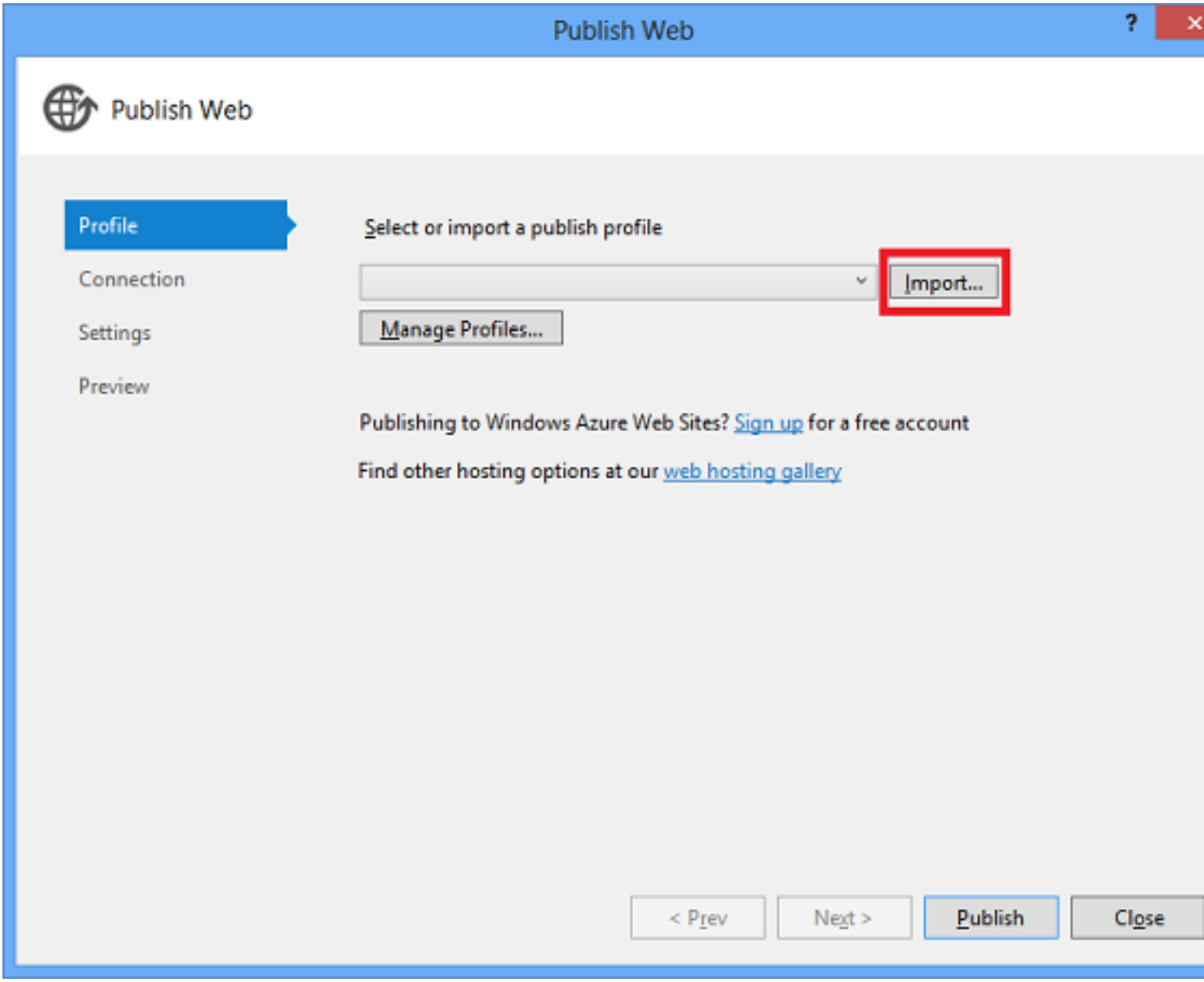


اپلیکیشن شما فعلا آماده است و می‌توانید آن را روی Windows Azure توزیع کنید. بعدا دیتابیس و دسترسی داده نیز اضافه خواهد شد.

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.

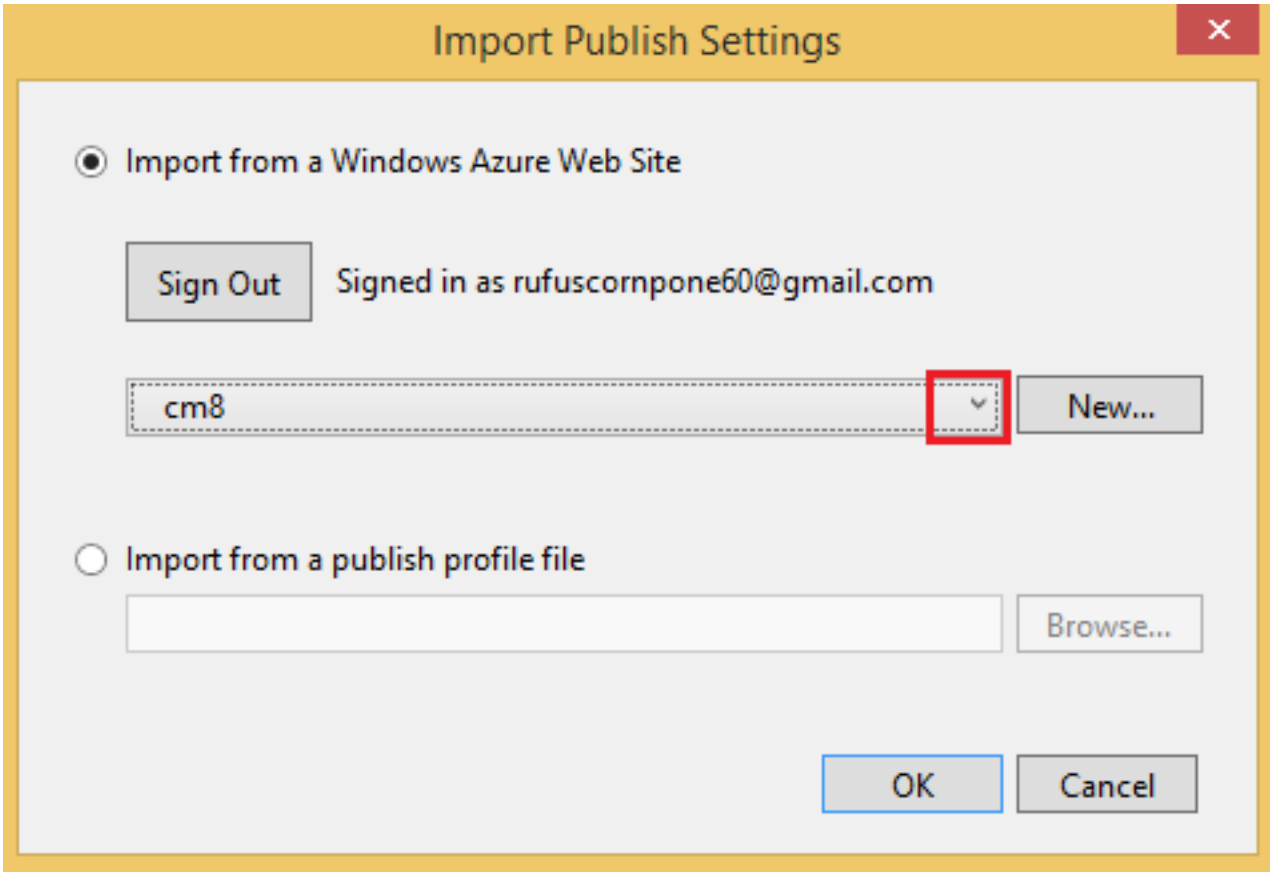
این روش ساده‌تر و سریع‌تر است، اما اگر از آن استفاده کنید دیگر قادر به مشاهده Windows Azure SQL Database یا Mobile Services در پنجره **Server Explorer** نخواهید بود.
روی **Manage subscriptions** کلیک کنید تا یک management certificate نصب کنید، که دسترسی به حساب کاربری شما را ممکن می‌سازد.

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.

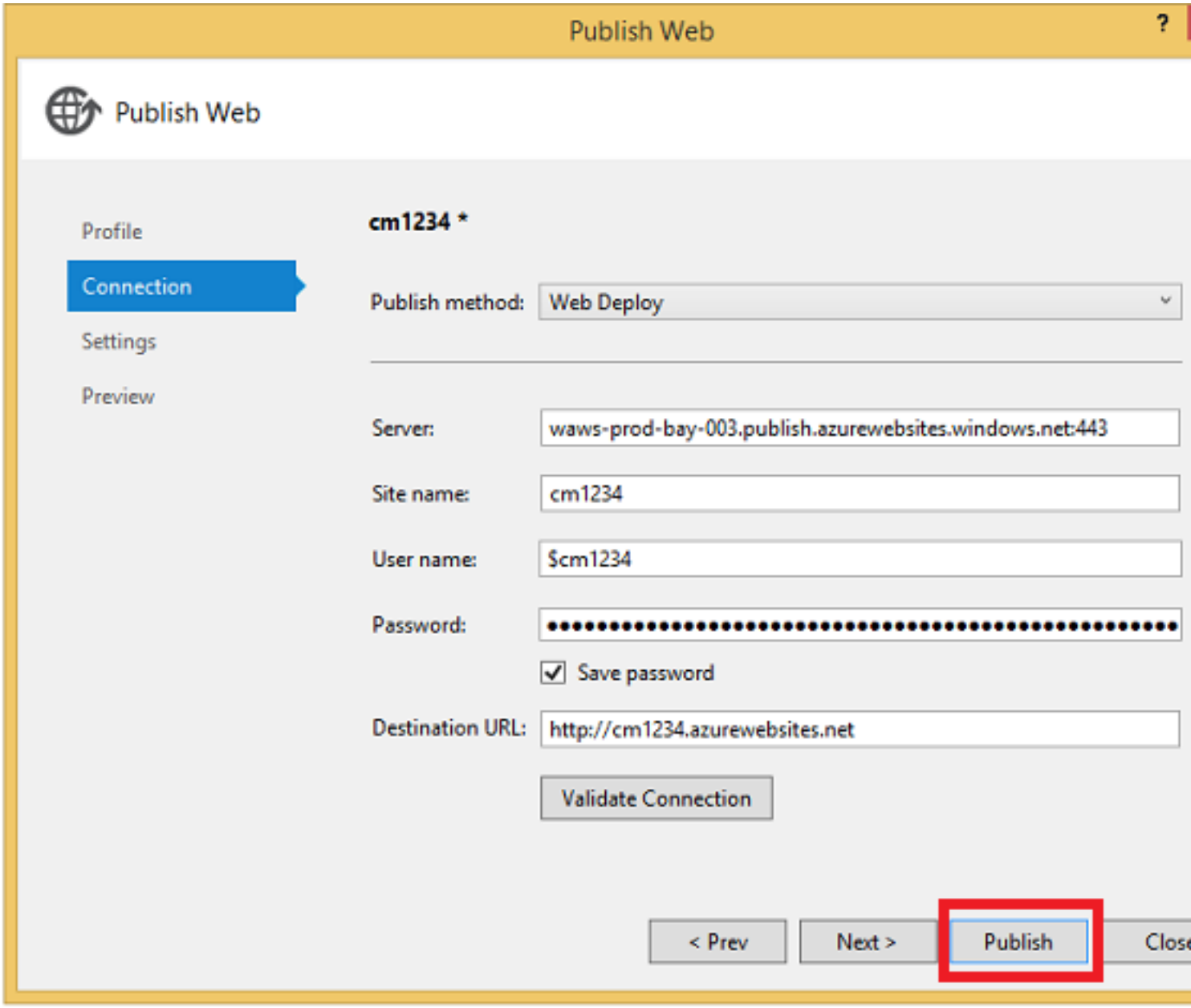
نکته امنیتی: این فایل تنظیمات را بیرون از پوشه‌های سورس کد خود دانلود کنید، مثلا پوشه Downloads. پس از اتمام عملیات Import هم این فایل را حذف کنید. کاربر مخربی که به این فایل دسترسی پیدا کند قادر خواهد بود تا سرویس‌های Windows Azure شما را کاملا کنترل کند.

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.



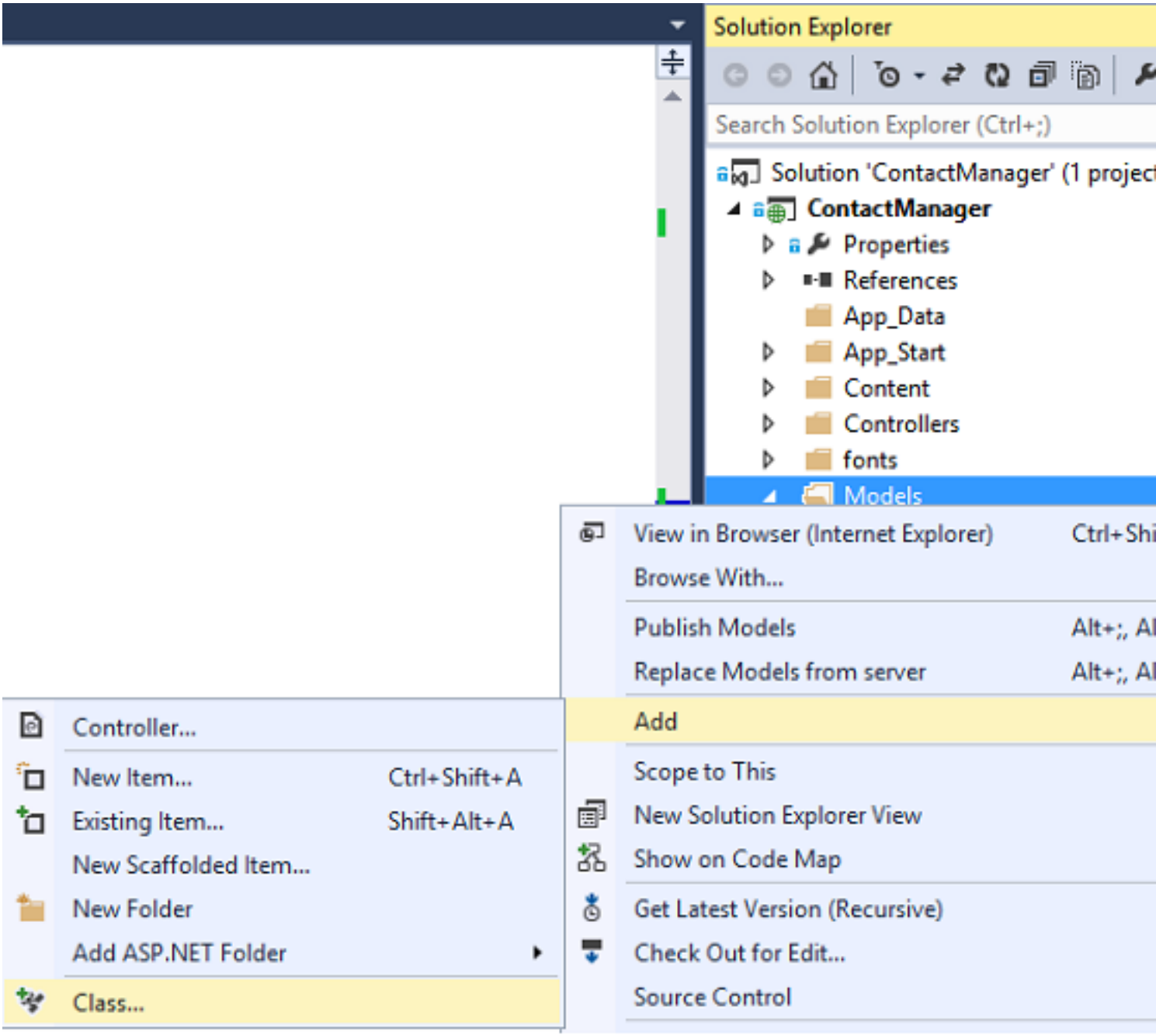
اپلیکیشن شما حالا در فضای ابری اجرا می‌شود. دفعه بعد که اپلیکیشن را منتشر کنید تنها فایل‌های تغییر کرده (یا جدید) آپلود خواهند شد.

یک دیتابیس به اپلیکیشن اضافه کنید

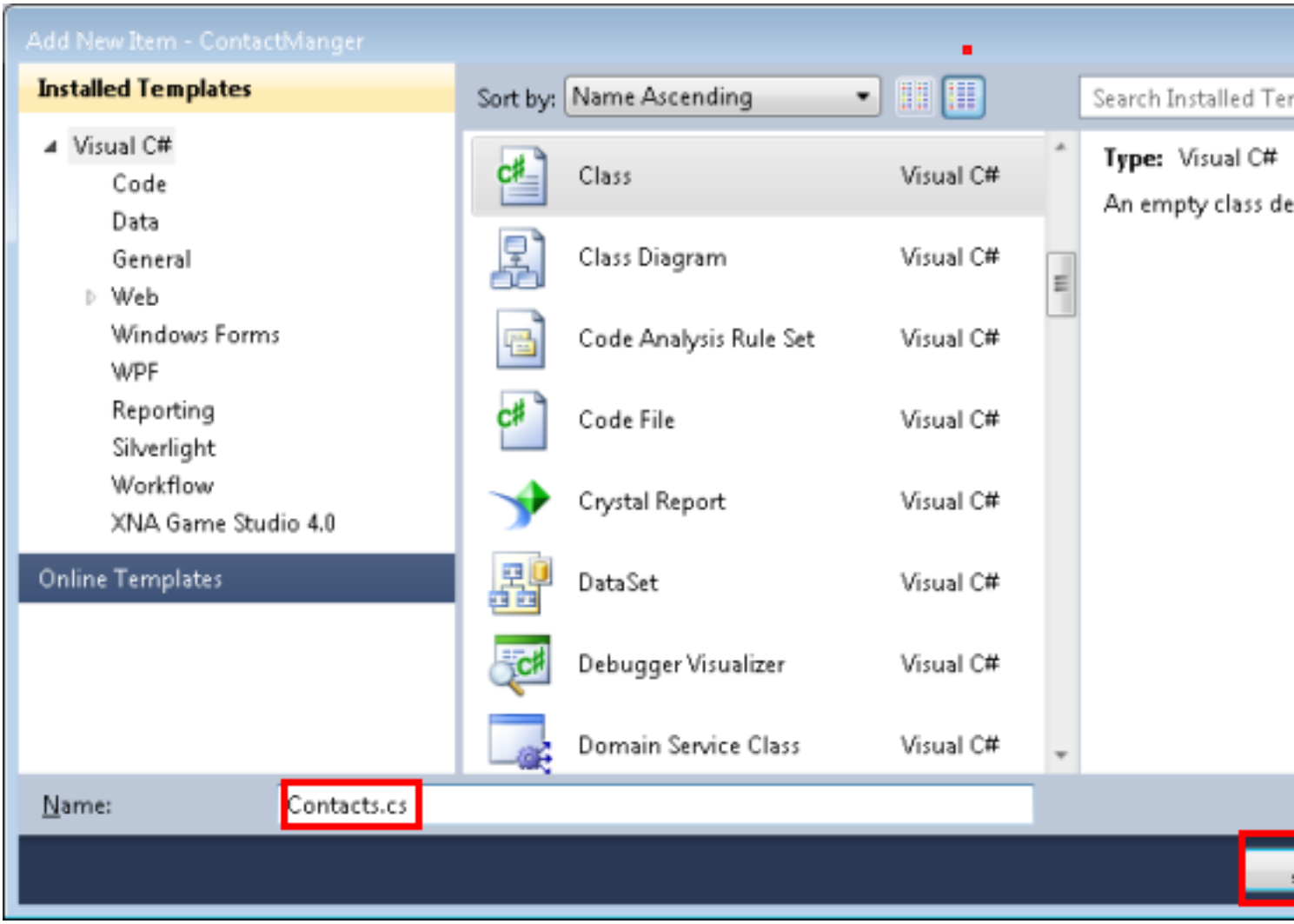
در مرحله بعد یک دیتابیس خواهیم ساخت تا اپلیکیشن ما بتواند اطلاعات را نمایش دهد و ویرایش کند. برای ایجاد دیتابیس و دسترسی به داده‌ها از Entity Framework استفاده خواهیم کرد.

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.

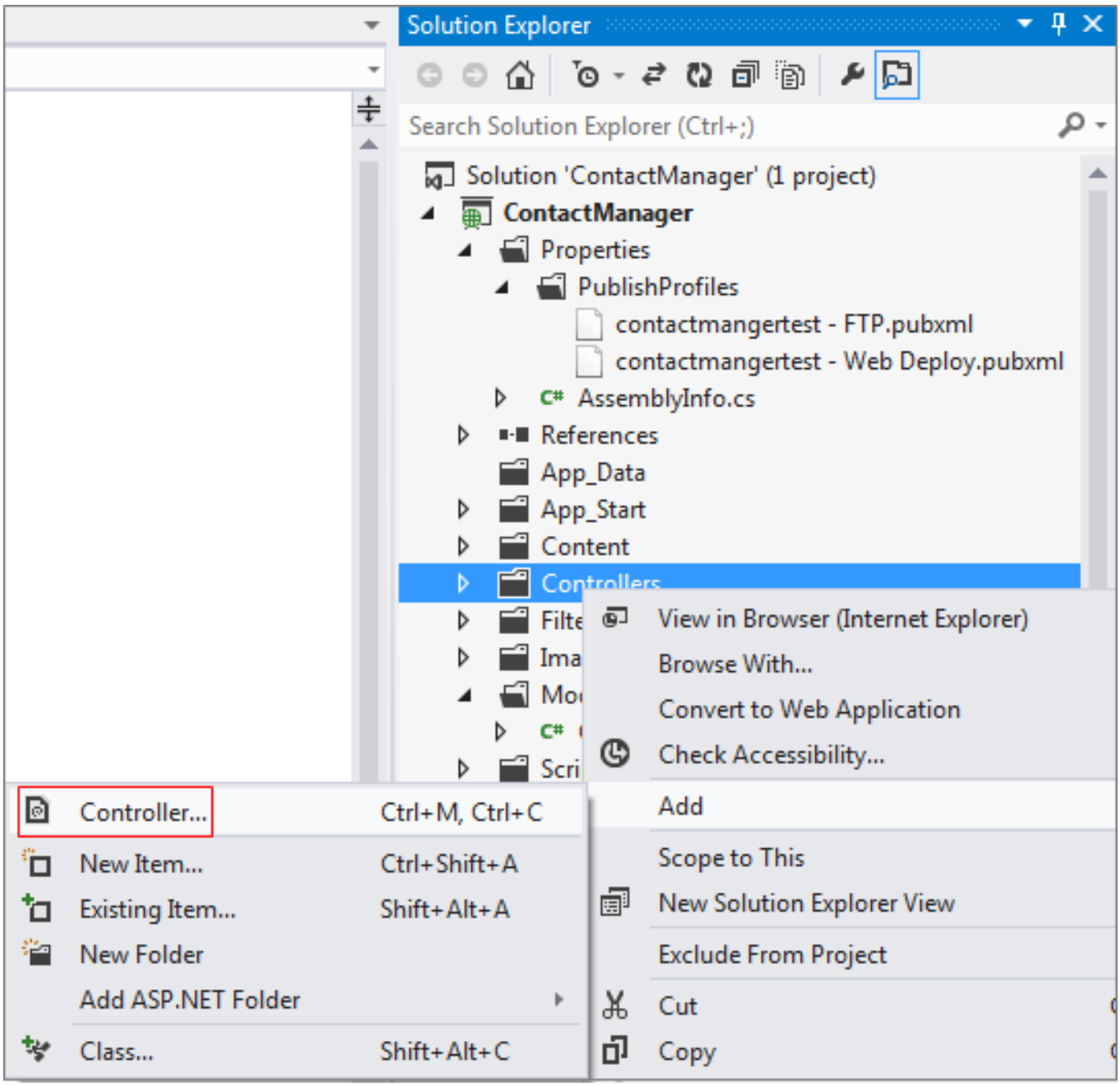
```
using System.ComponentModel.DataAnnotations;
using System.Globalization;
namespace ContactManager.Models
{
    public class Contact
    {
        public int ContactId { get; set; }
        public string Name { get; set; }
        public string Address { get; set; }
        public string City { get; 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.

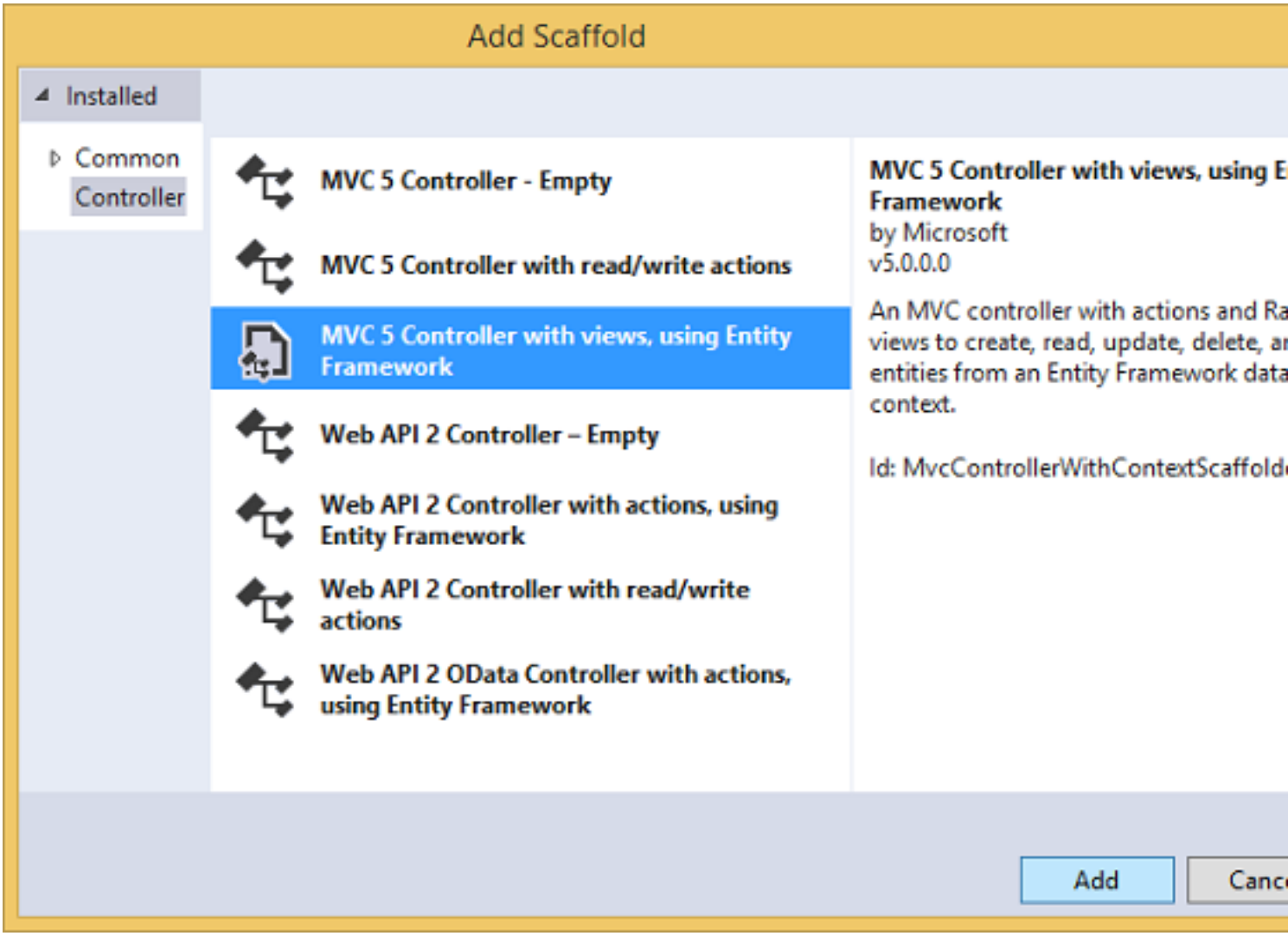
یک کنترلر و نما برای داده‌ها اضافه کنید

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.



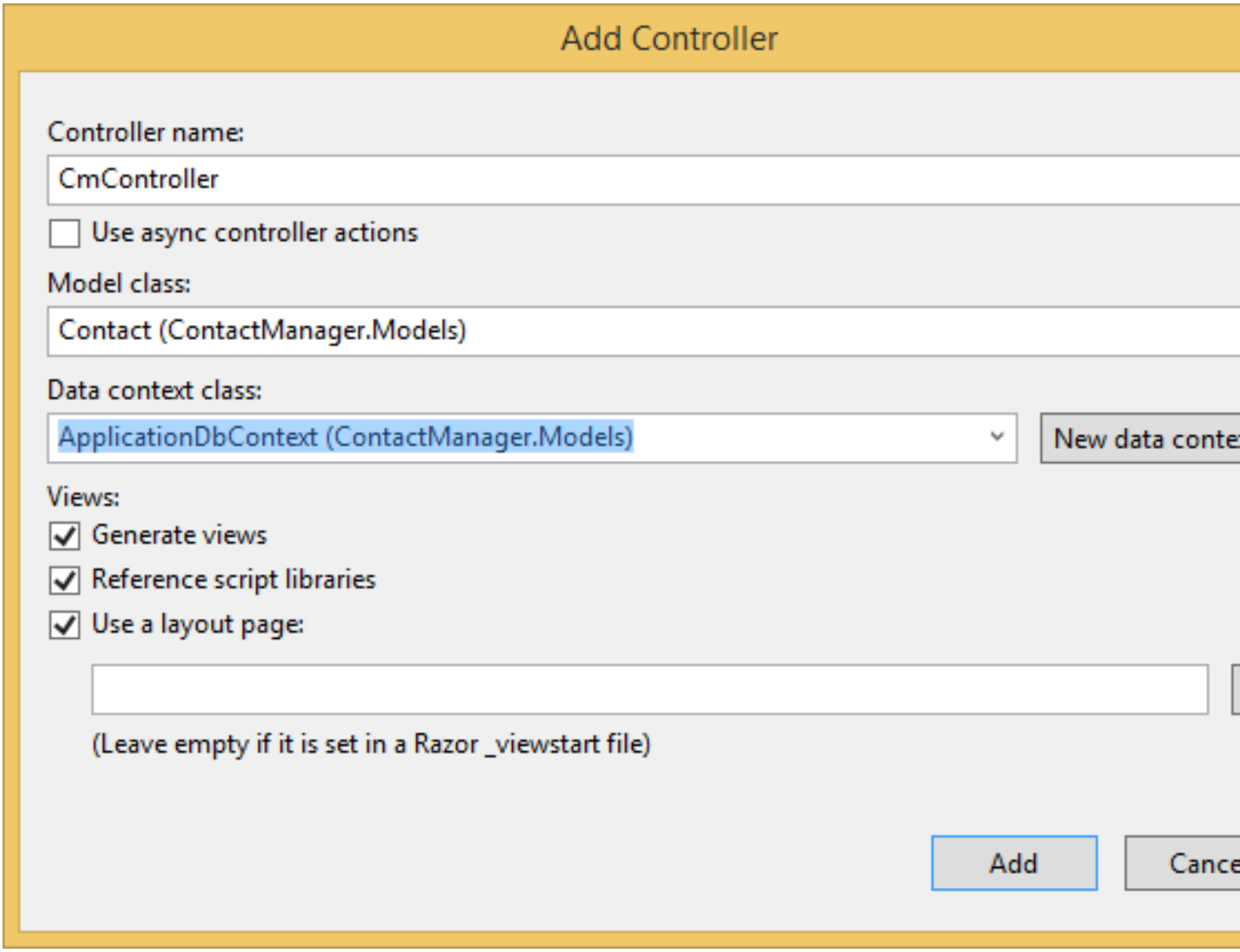
در دیالوگ باکس **Add Scaffold** گزینه **MVC 5 Controller with views, using EF** را انتخاب کنید.



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.

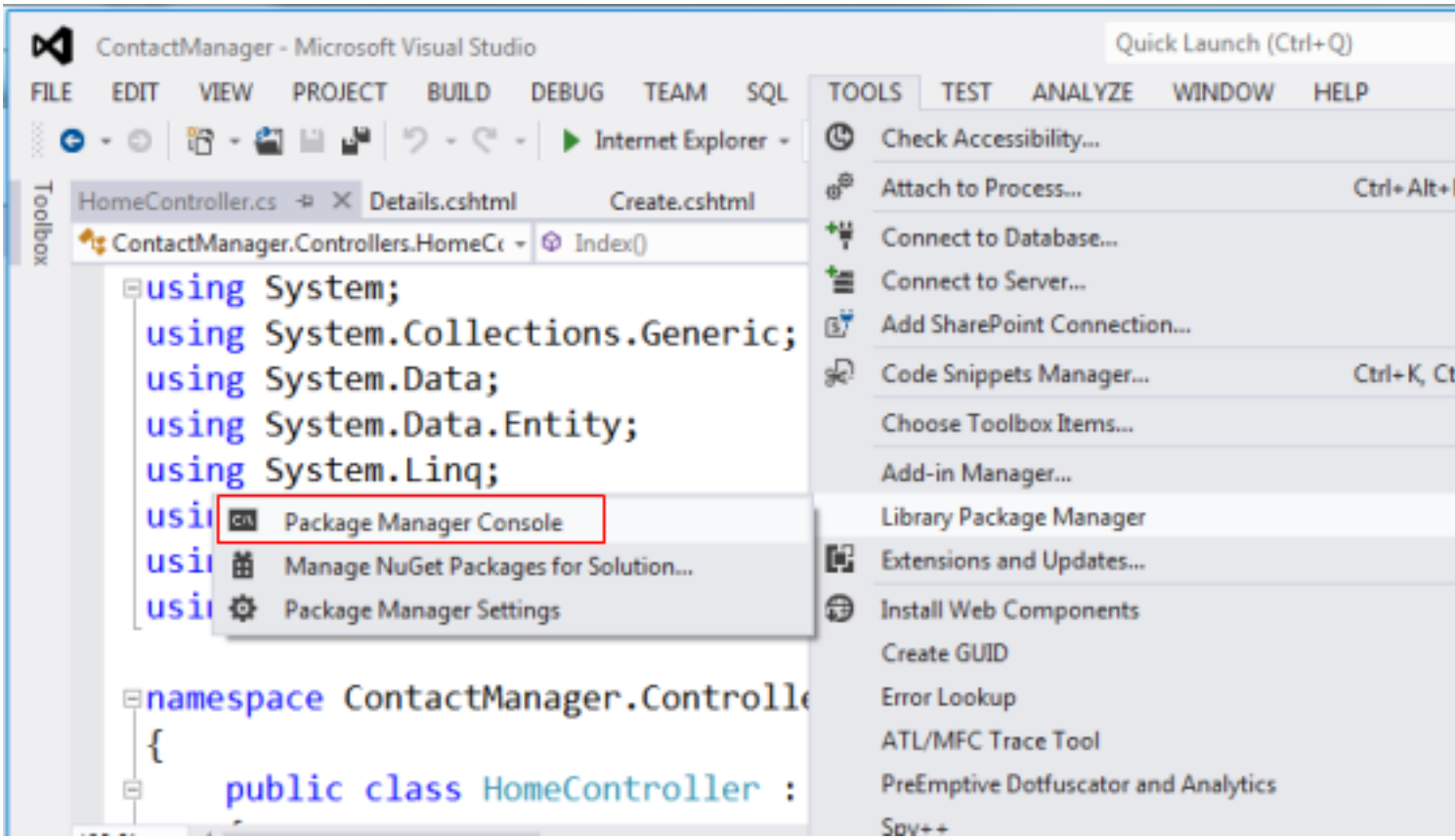


روی Add کلیک کنید. ویژوال استودیو بصورت خودکار با استفاده از Scaffolding متدها و Viewهای لازم برای عملیات CRUD را فراهم می‌کند، که همگی از مدل Contact استفاده می‌کنند.

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;
```

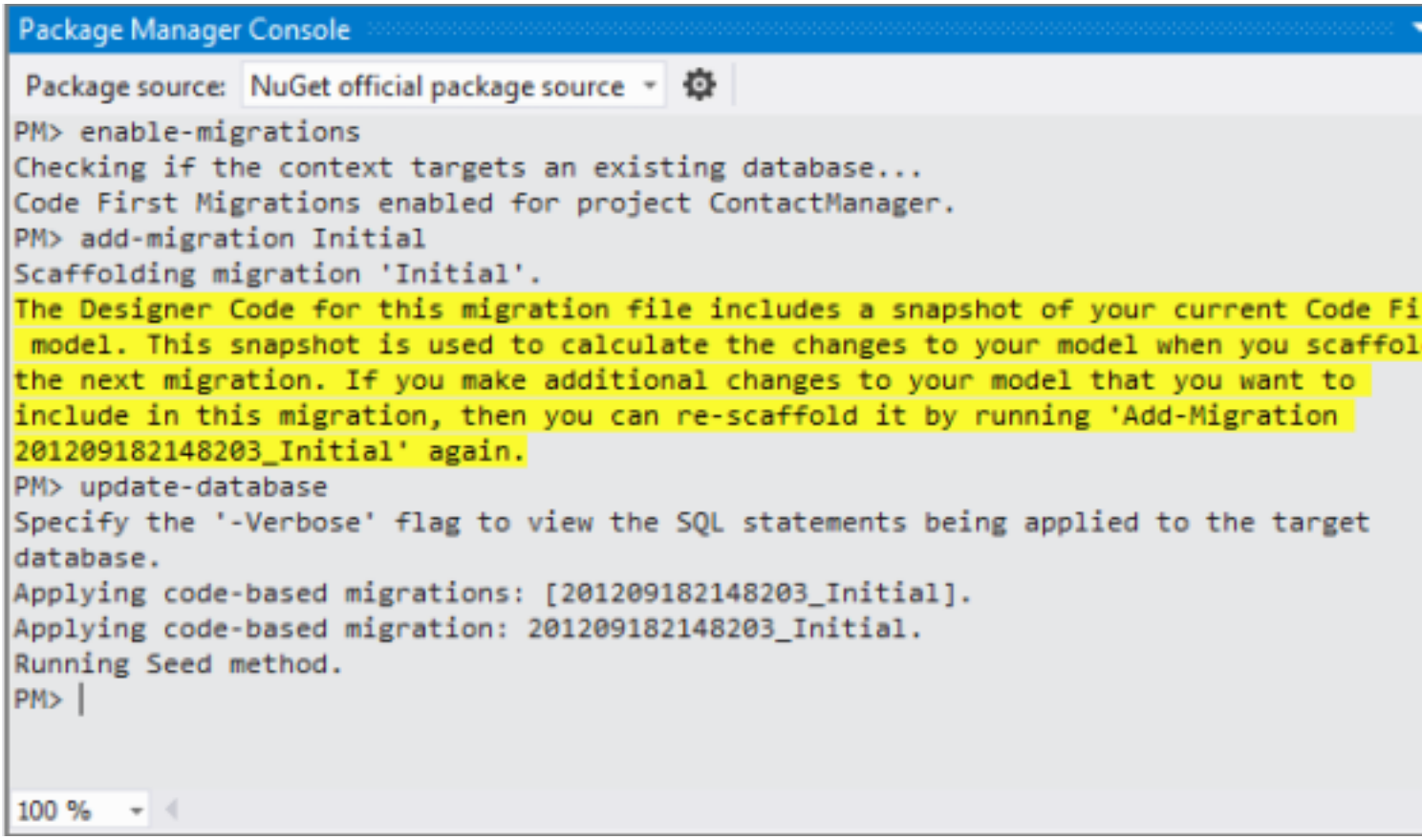
حال متد *Seed* را با قطعه کد زیر جایگزین کنید.

```
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.

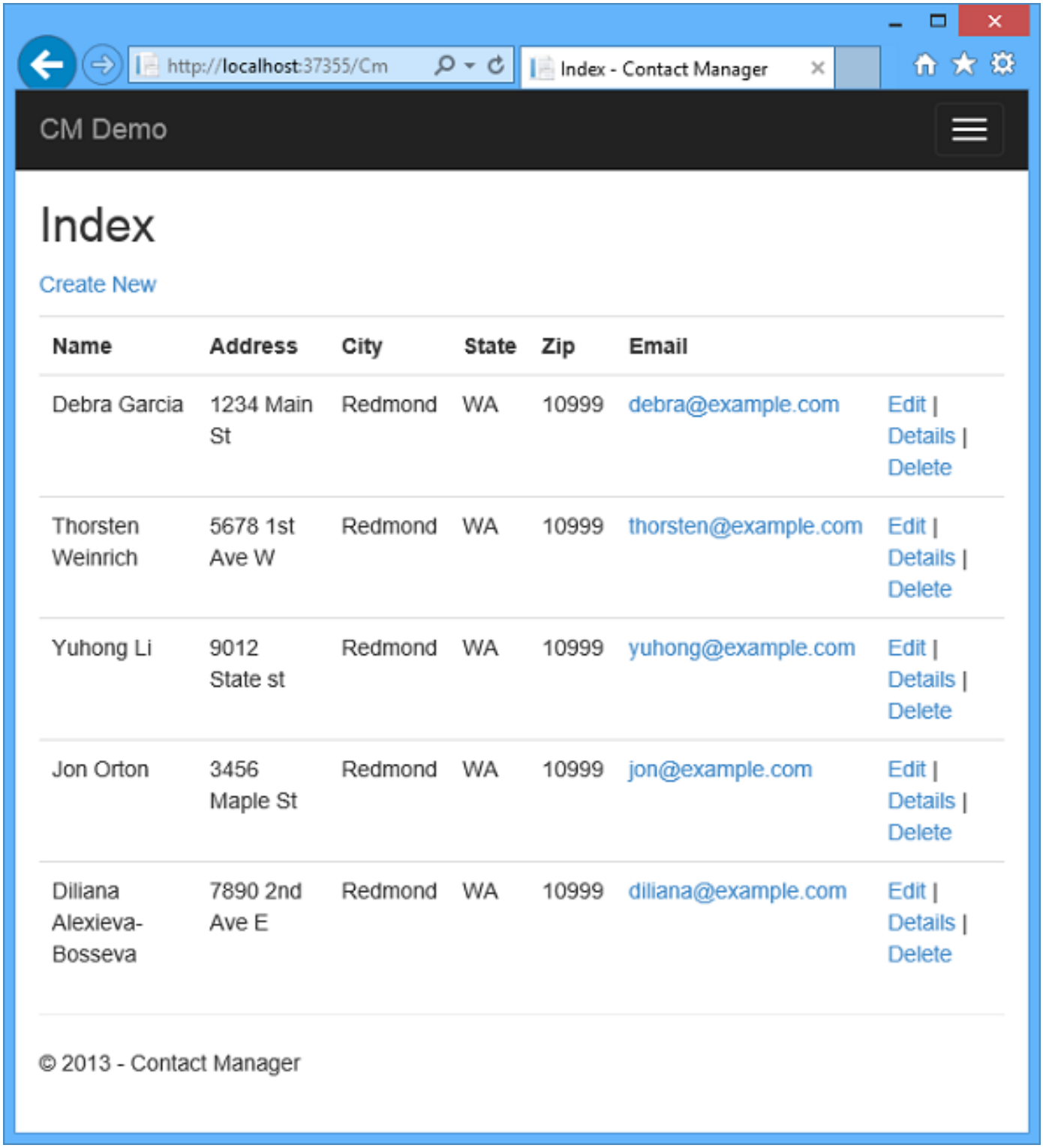
```
update-database
```



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).

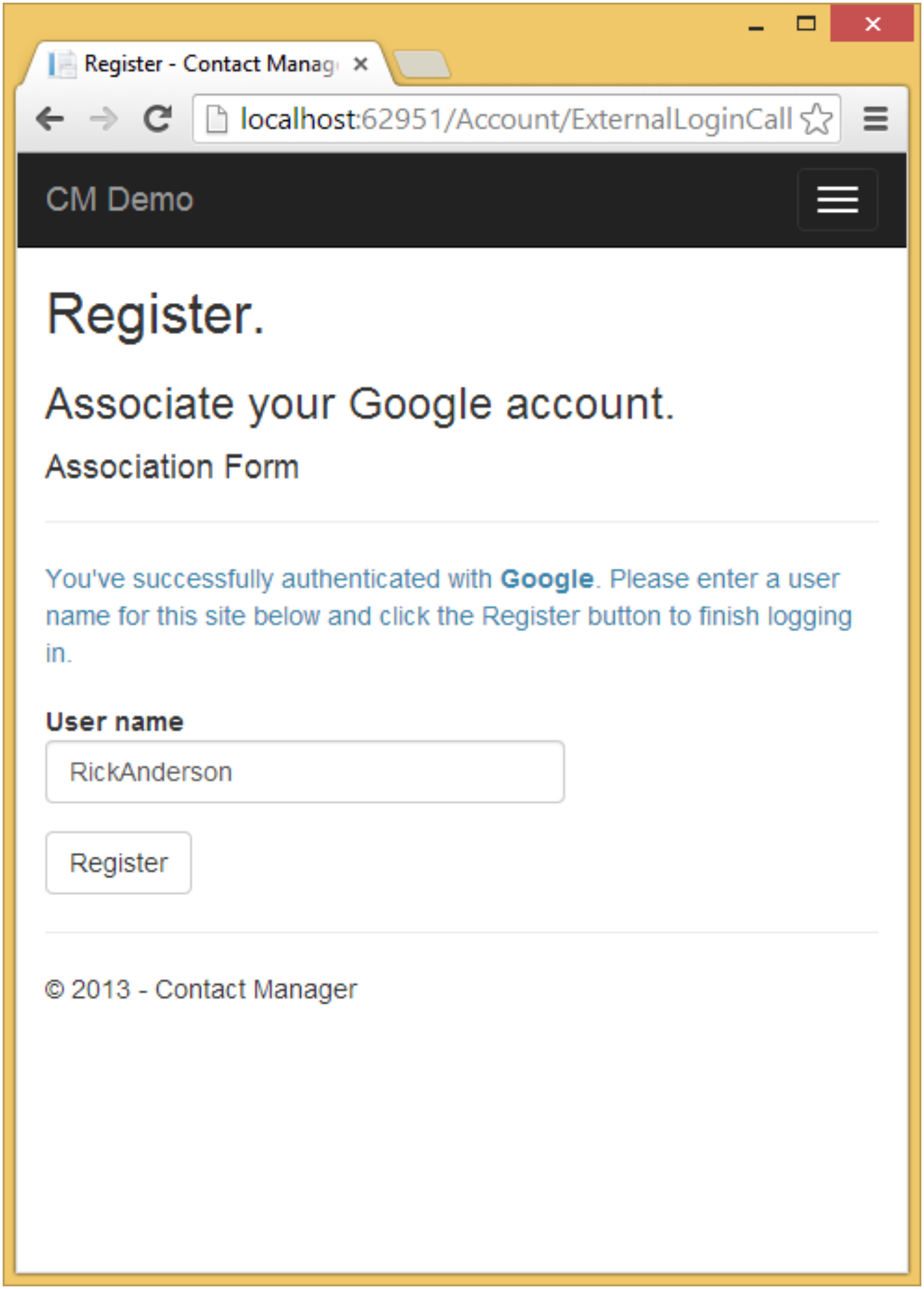
فایل *App_Start/Startup.Auth.cs* را باز کنید. توضیحات متد *app.UseGoogleAuthentication* را حذف کنید.

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();
       }
```

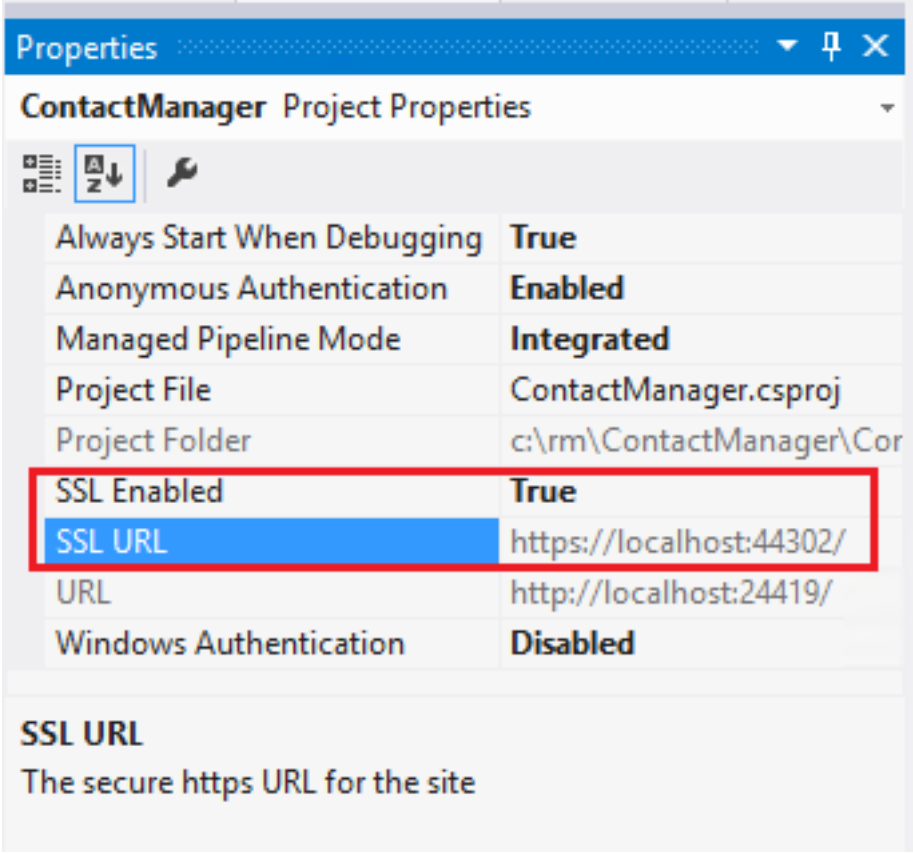
یک جستجوی عمومی برای عبارت *AllowAnonymous* انجام دهید. همانطور که مشاهده می‌کنید این خاصیت توسط متدهای ورود و ثبت نام در کنترلر Account نیز استفاده شده است.

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.

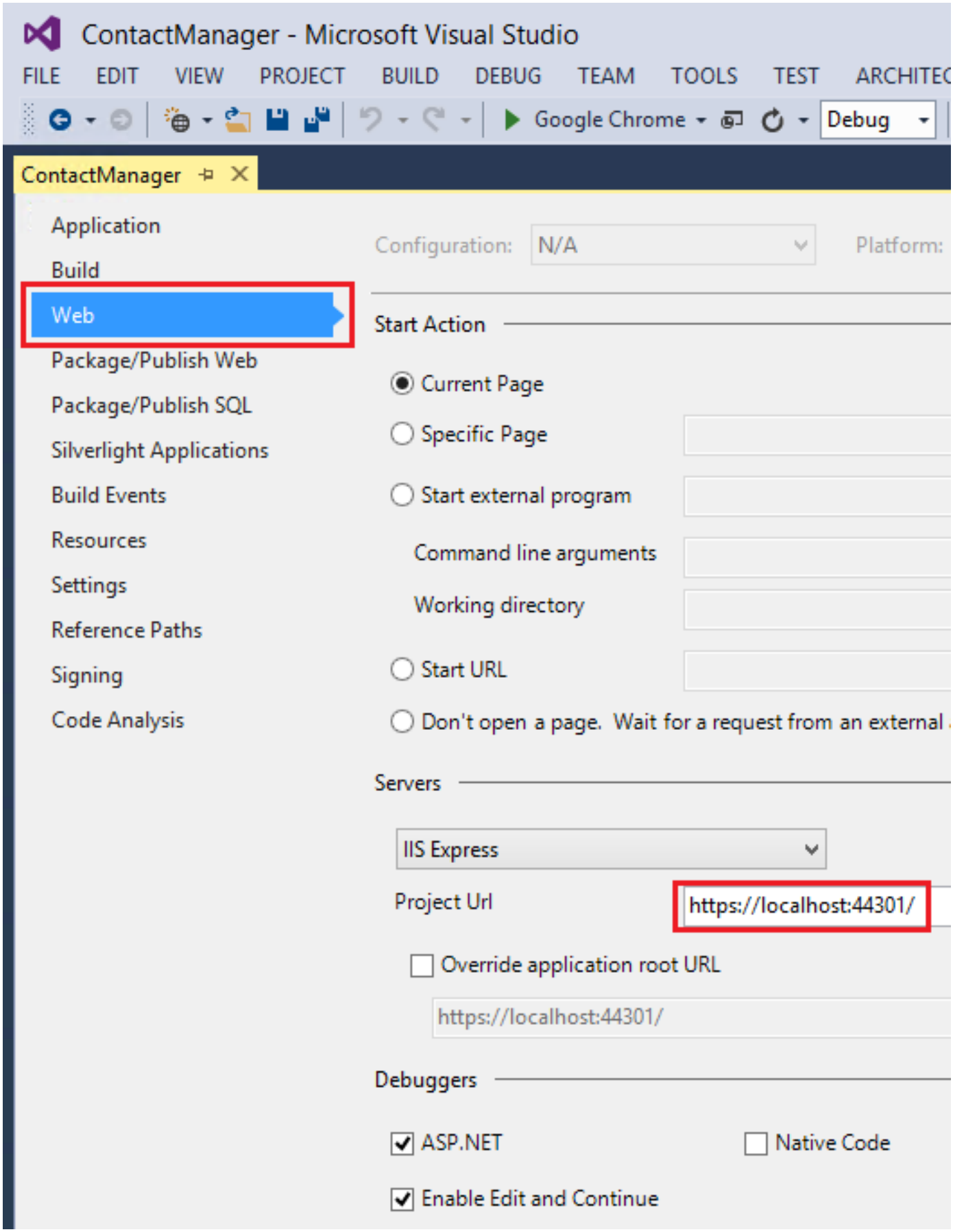
```
public class CmController : Controller
{
    private ContactManagerContext db = new ContactManagerContext();

    // 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.

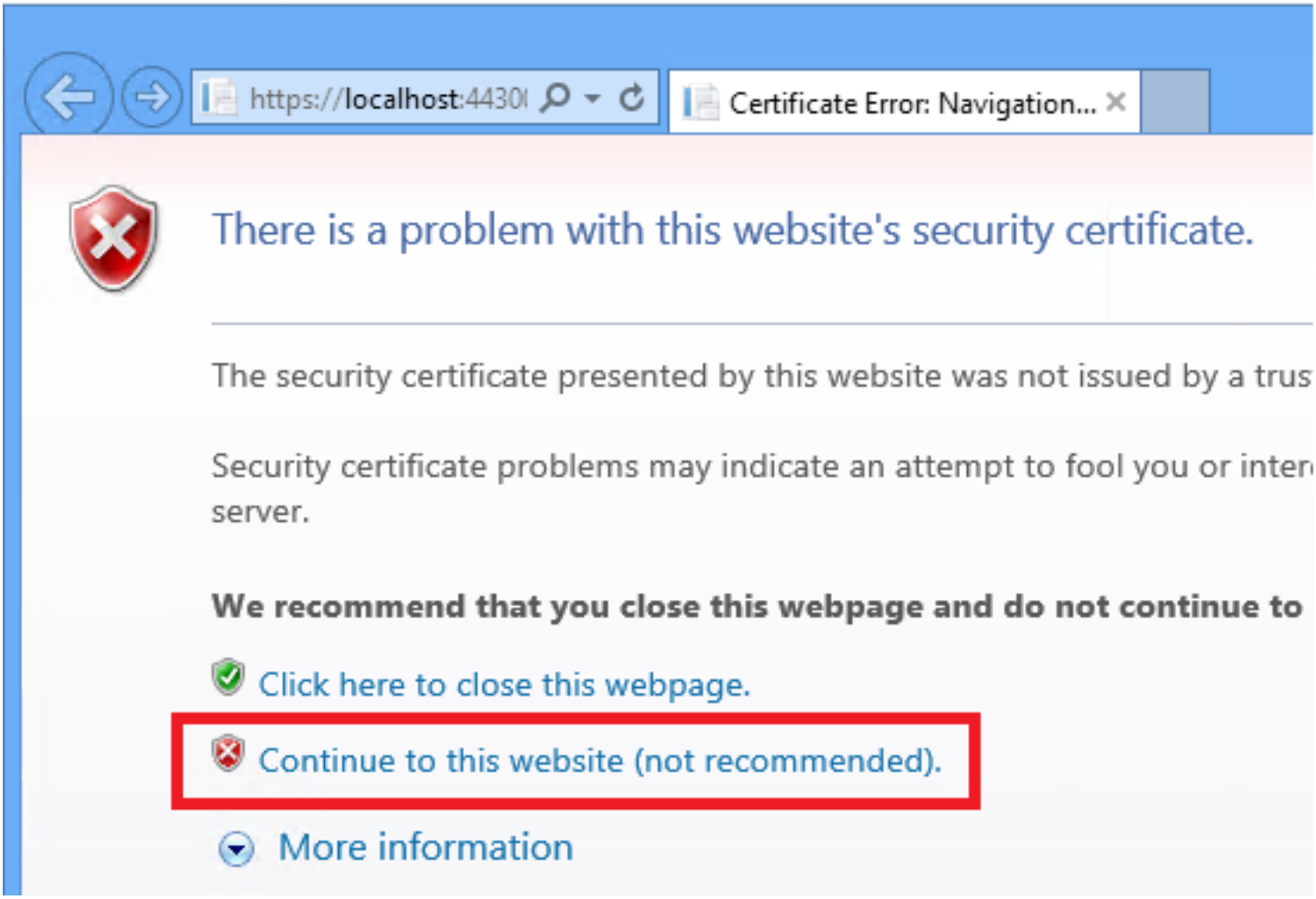


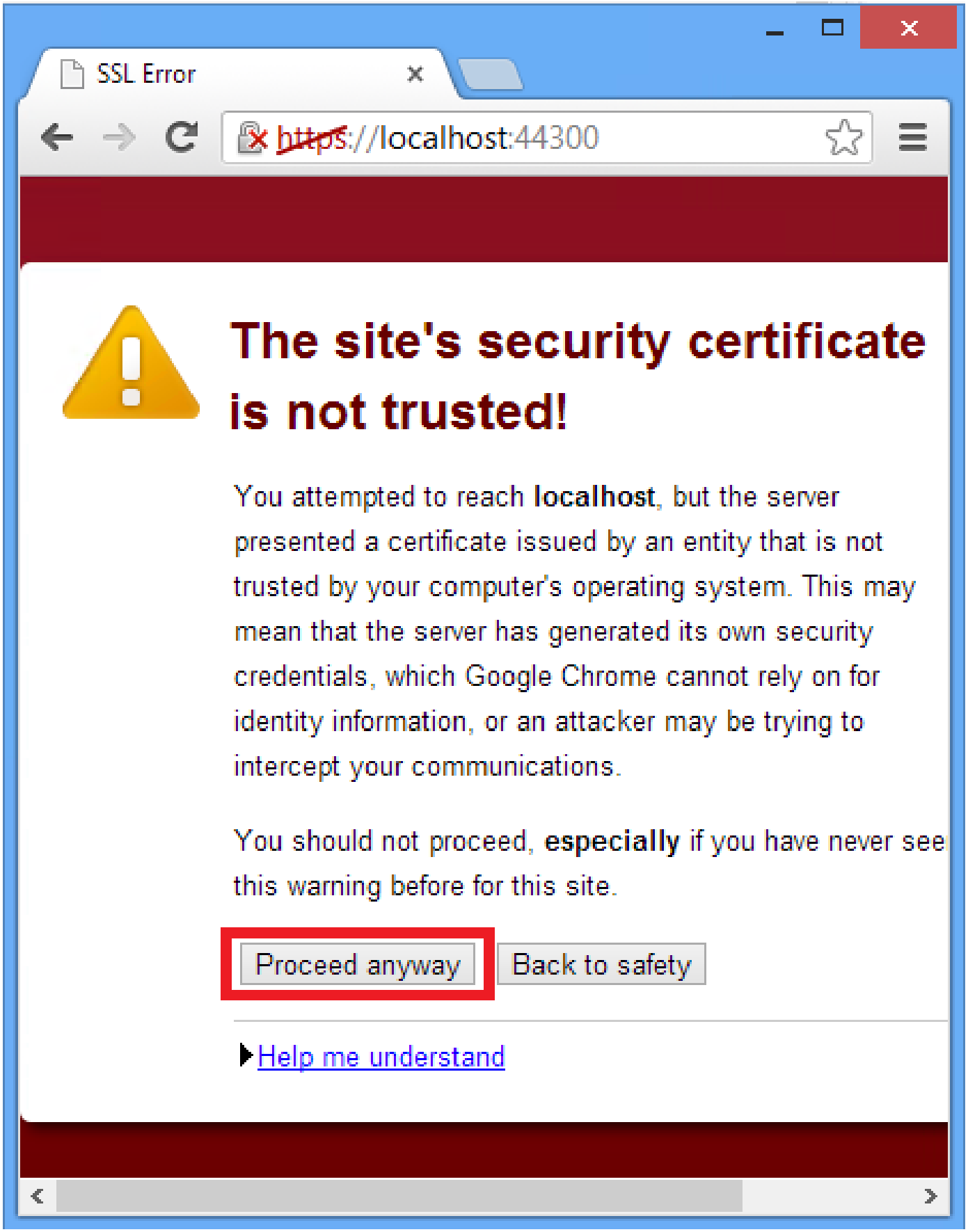
روی نام پروژه کلیک راست کنید و **Properties** را انتخاب کنید. در قسمت چپ گزینه **Web** را انتخاب کنید. حالا مقدار **Project Url** را به آدرسی که کپی کرده اید تغییر دهید. نهایتا تغییرات را ذخیره کنید و پنجره را ببندید.



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.

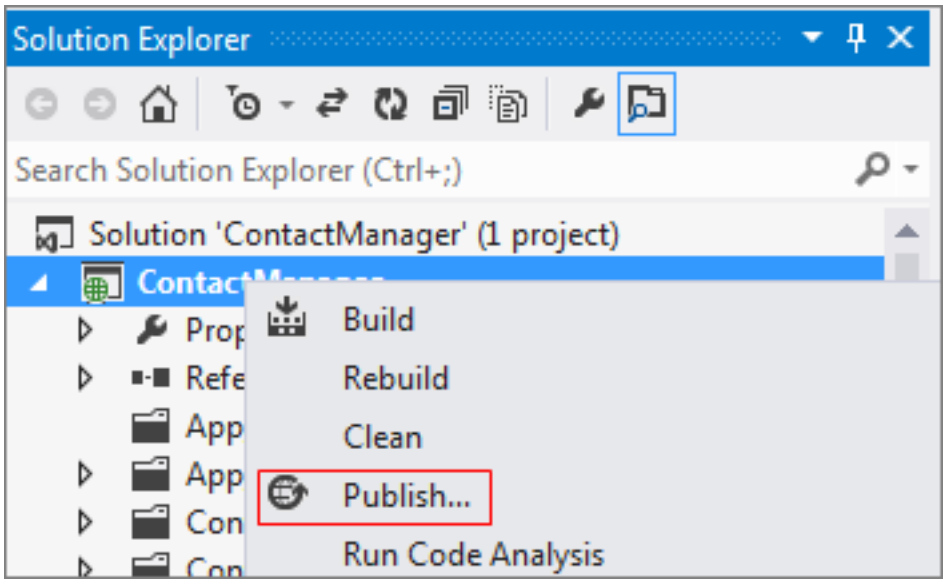
با کاربر *user1* که قبلا ساختید وارد سایت شوید. حال به صفحه ویرایشی که قبلا درخواست کرده بودید هدایت می‌شوید.

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.

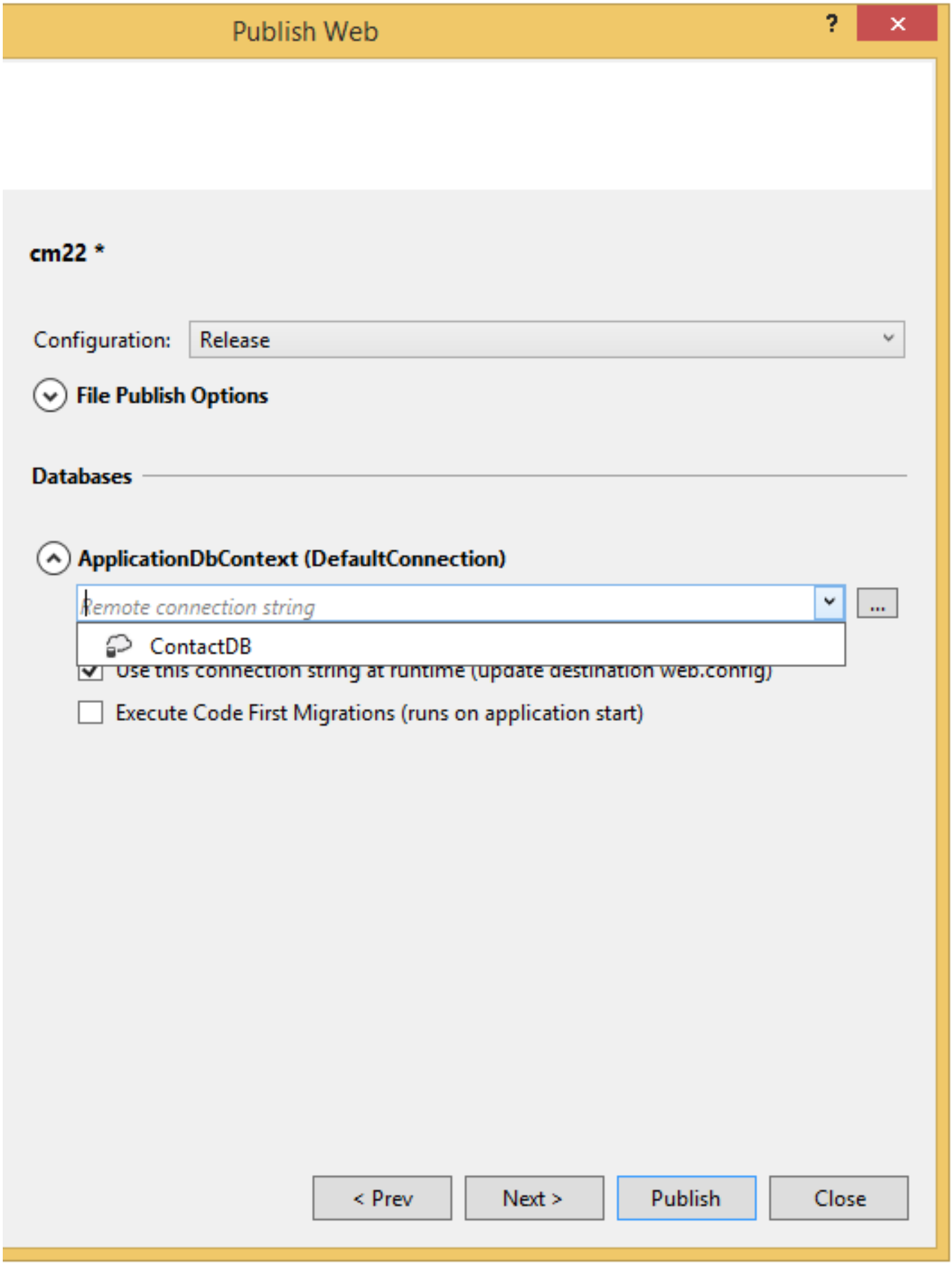
اپلیکیشن را روی Windows Azure منتشر کنید

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

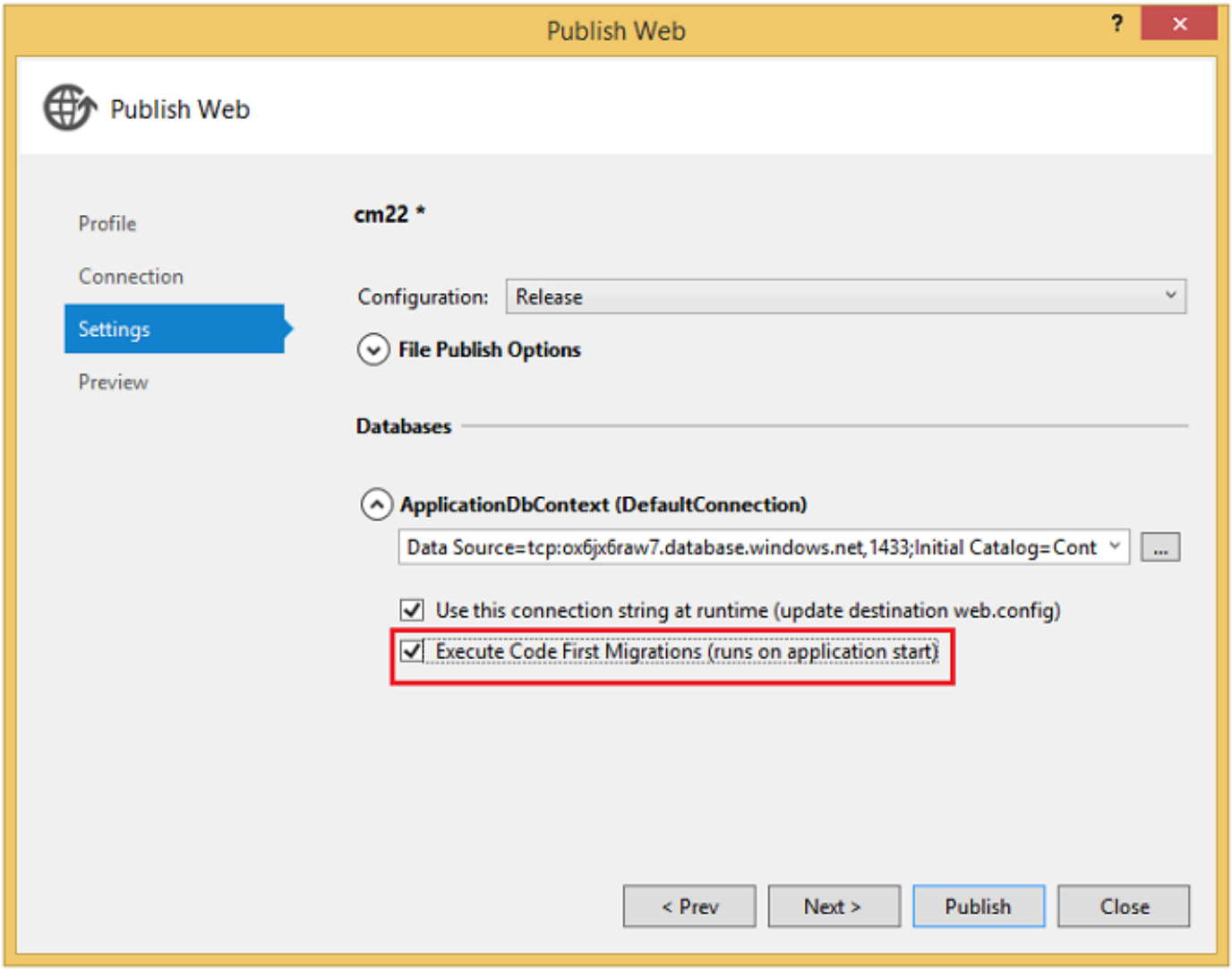


در دیالوگ باز شده روی قسمت **Settings** کلیک کنید. روی **File Publish Options** کلیک کنید تا بتوانید **Remote connection string** را برای **ApplicationDbContext** و دیتابیس ContactDB انتخاب کنید.

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.

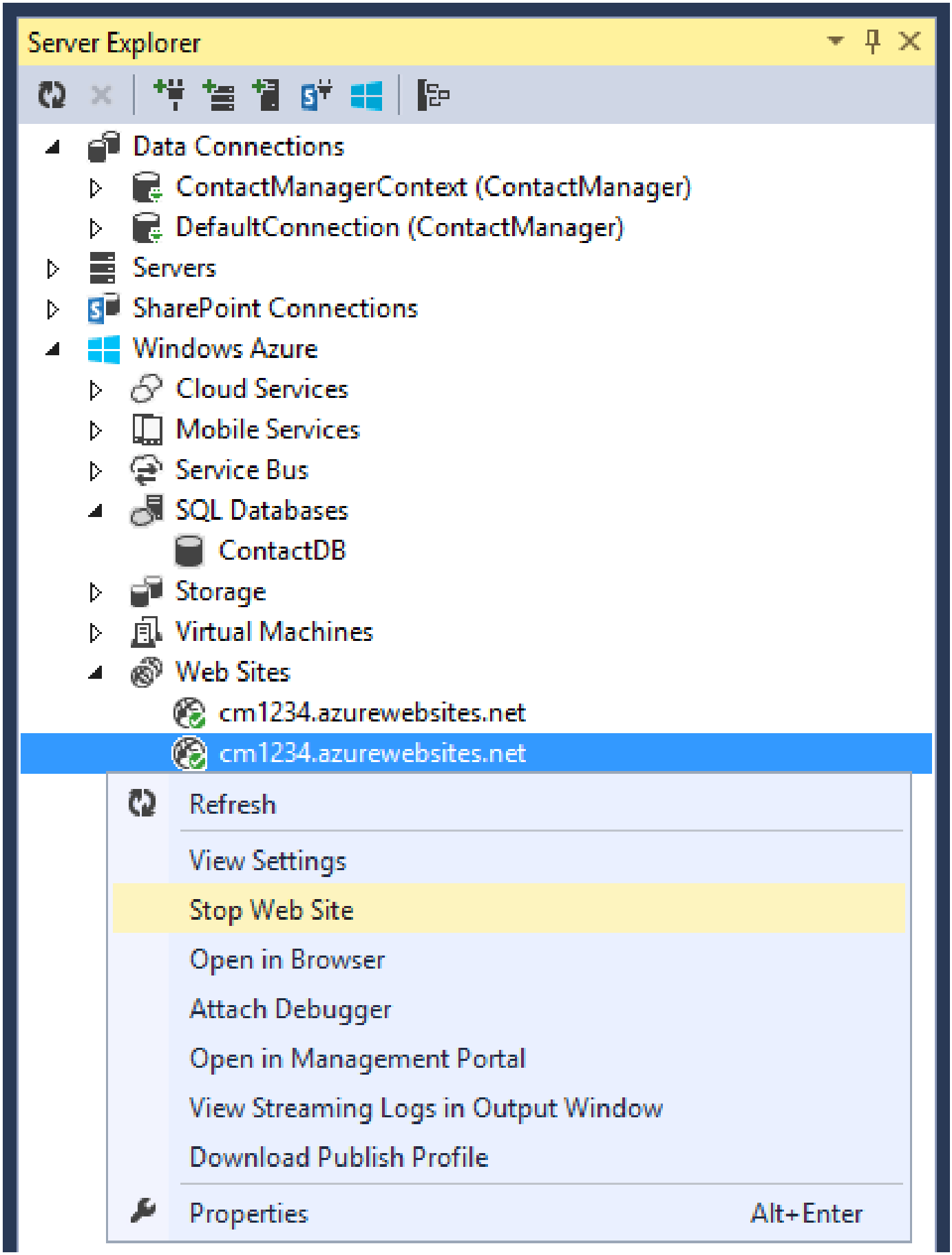


حال **Publish** را کلیک کنید تا اپلیکیشن شما منتشر شود. با کاربر *user1* وارد سایت شوید و بررسی کنید که می‌توانید داده‌ها را ویرایش کنید یا خیر.

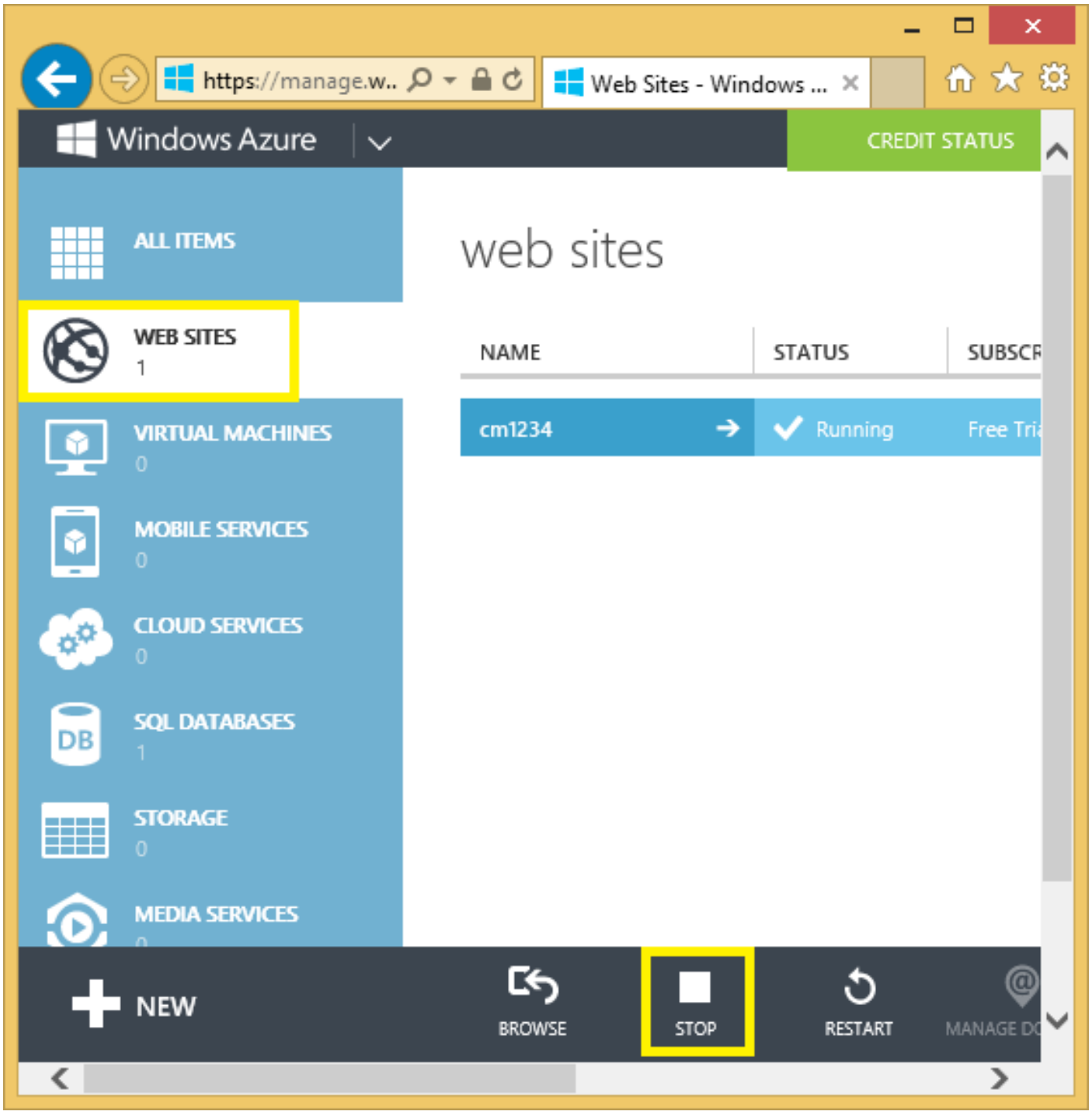
حال از سایت خارج شوید و توسط یک اکانت Google یا Facebook وارد سایت شوید، که در این صورت نقش canEdit نیز به شما تعلق می‌گیرد.

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.

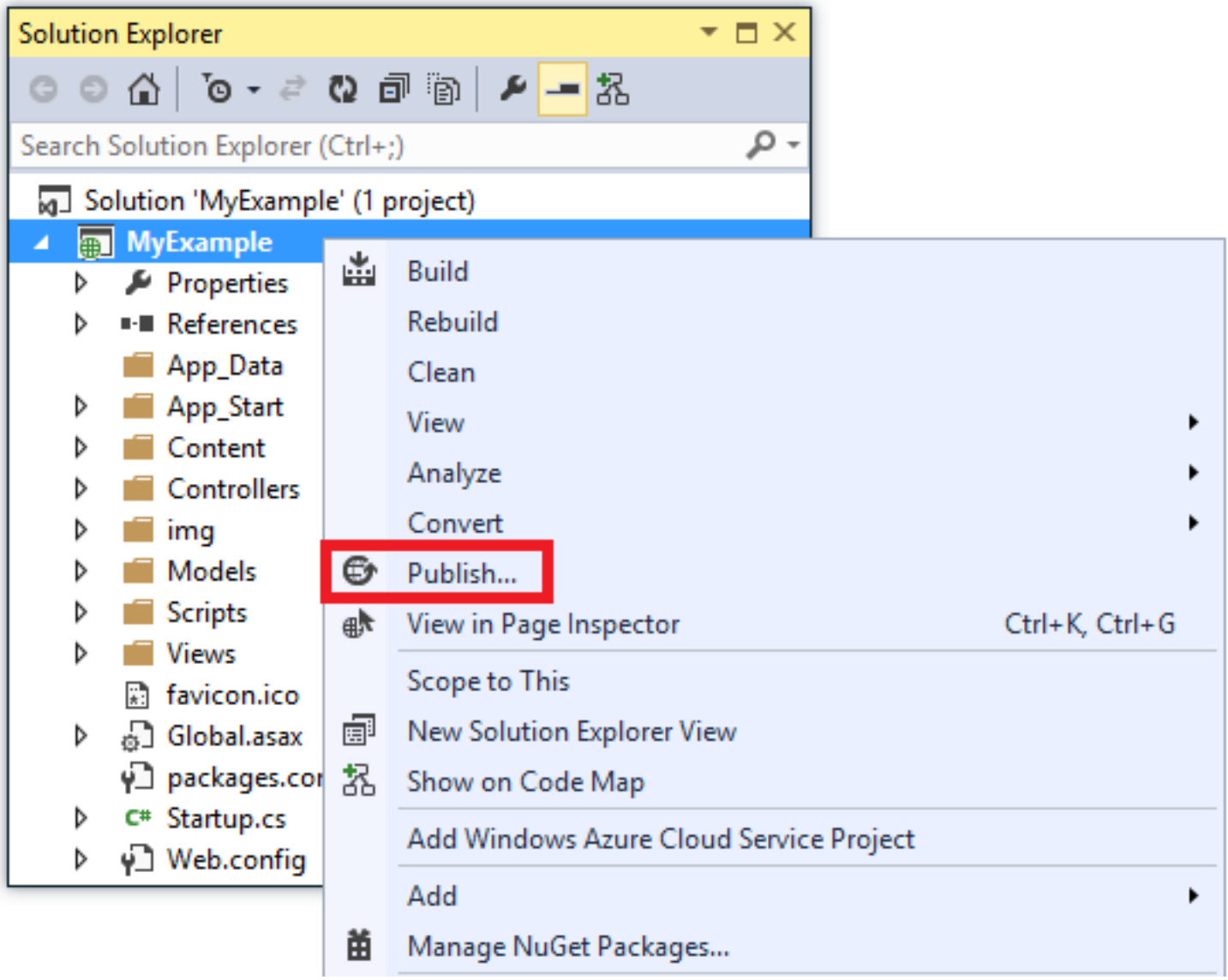


فراخوانی AddToRoleAsync را حذف و اپلیکیشن را منتشر و تست کنید

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.



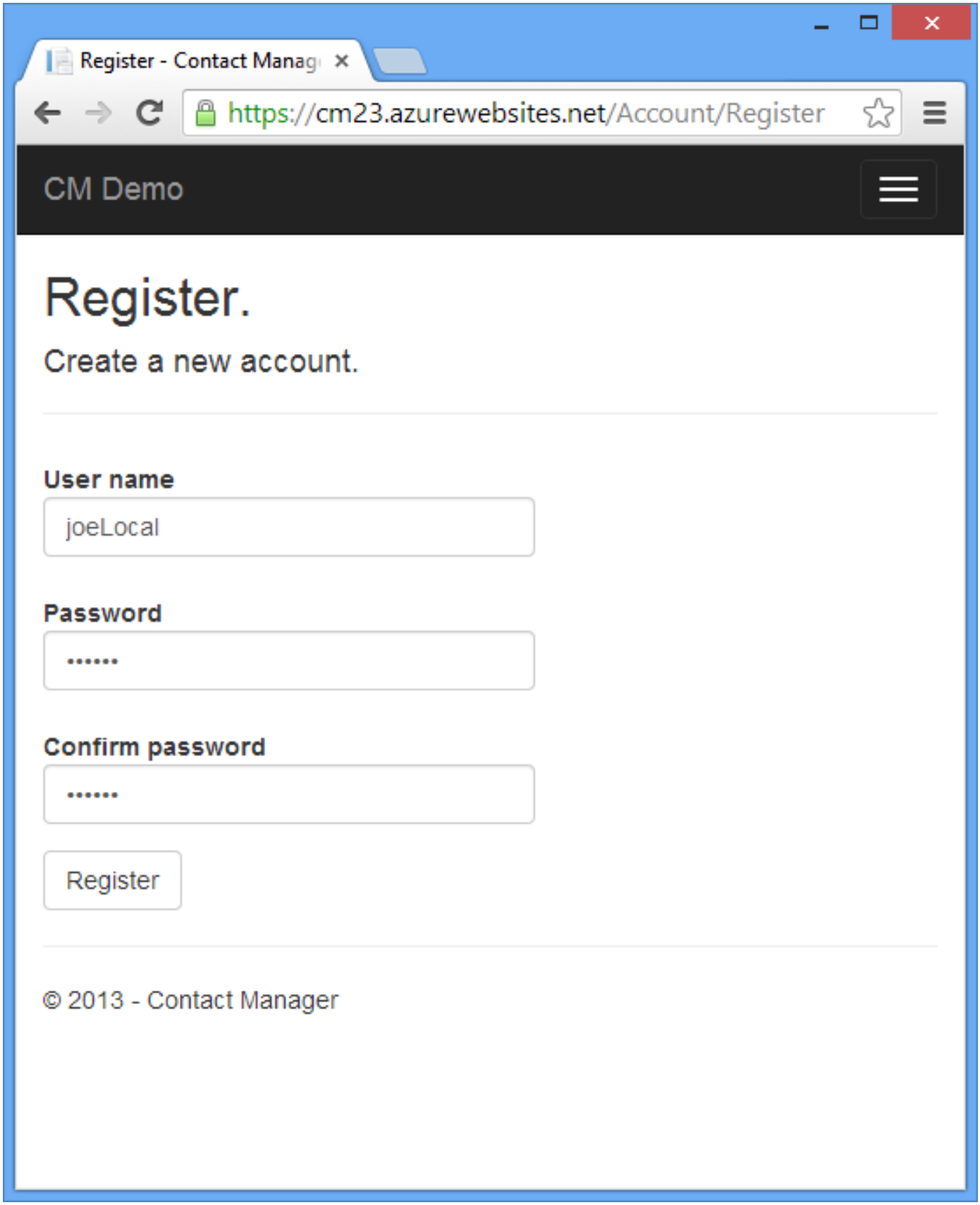
دکمه **Start Preview** را فشار دهید. در این مرحله تنها فایل هایی که نیاز به بروز رسانی دارند آپلود خواهند شد.

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.

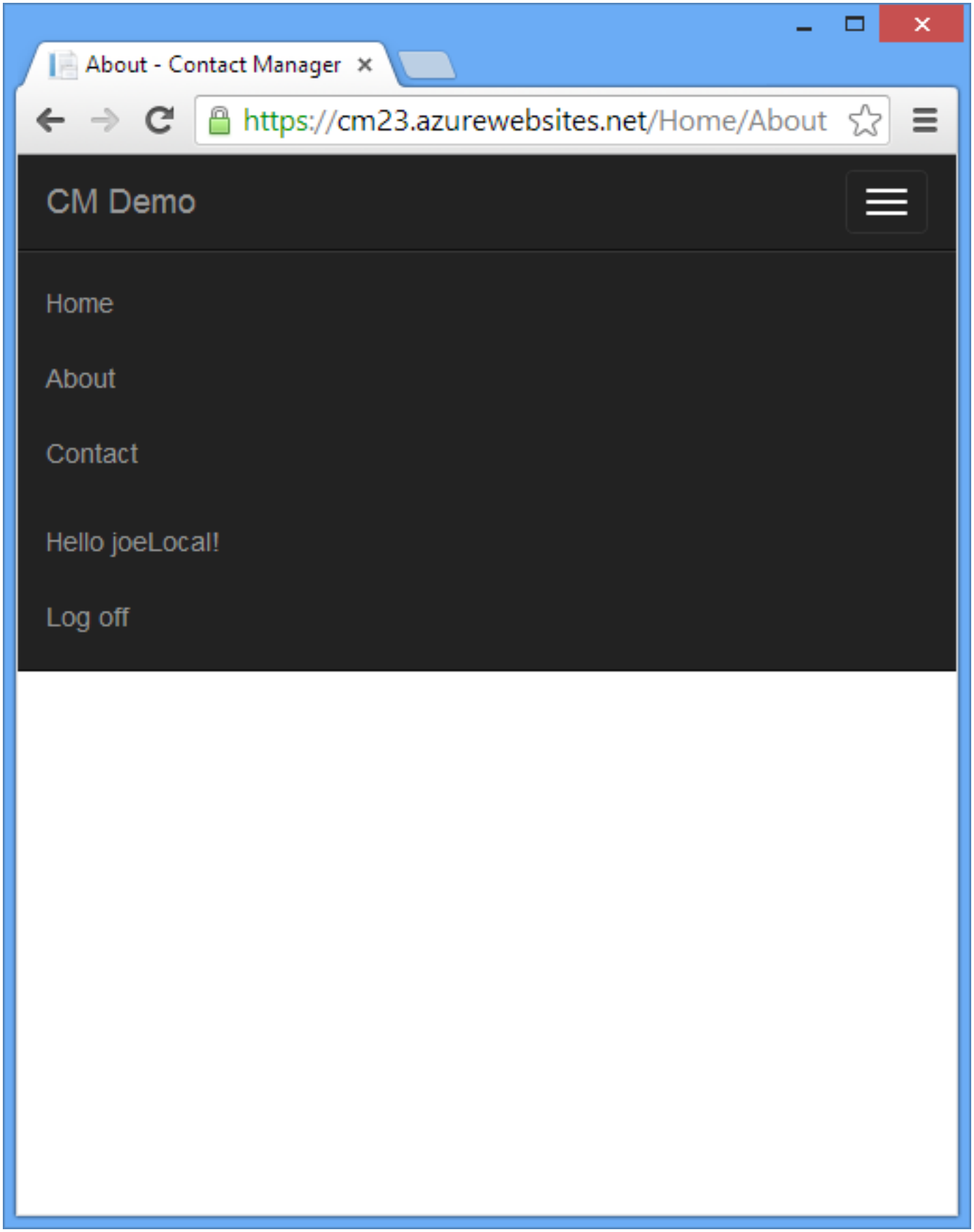
حال به ویژوال استودیو بازگردید و اپلیکیشن را منتشر کنید. اپلیکیشن Windows Azure شما باید در مرورگر پیش فرض تان باز شود. حال شما در حال مشاهده صفحه اصلی سایت بعنوان یک کاربر ناشناس هستید.

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

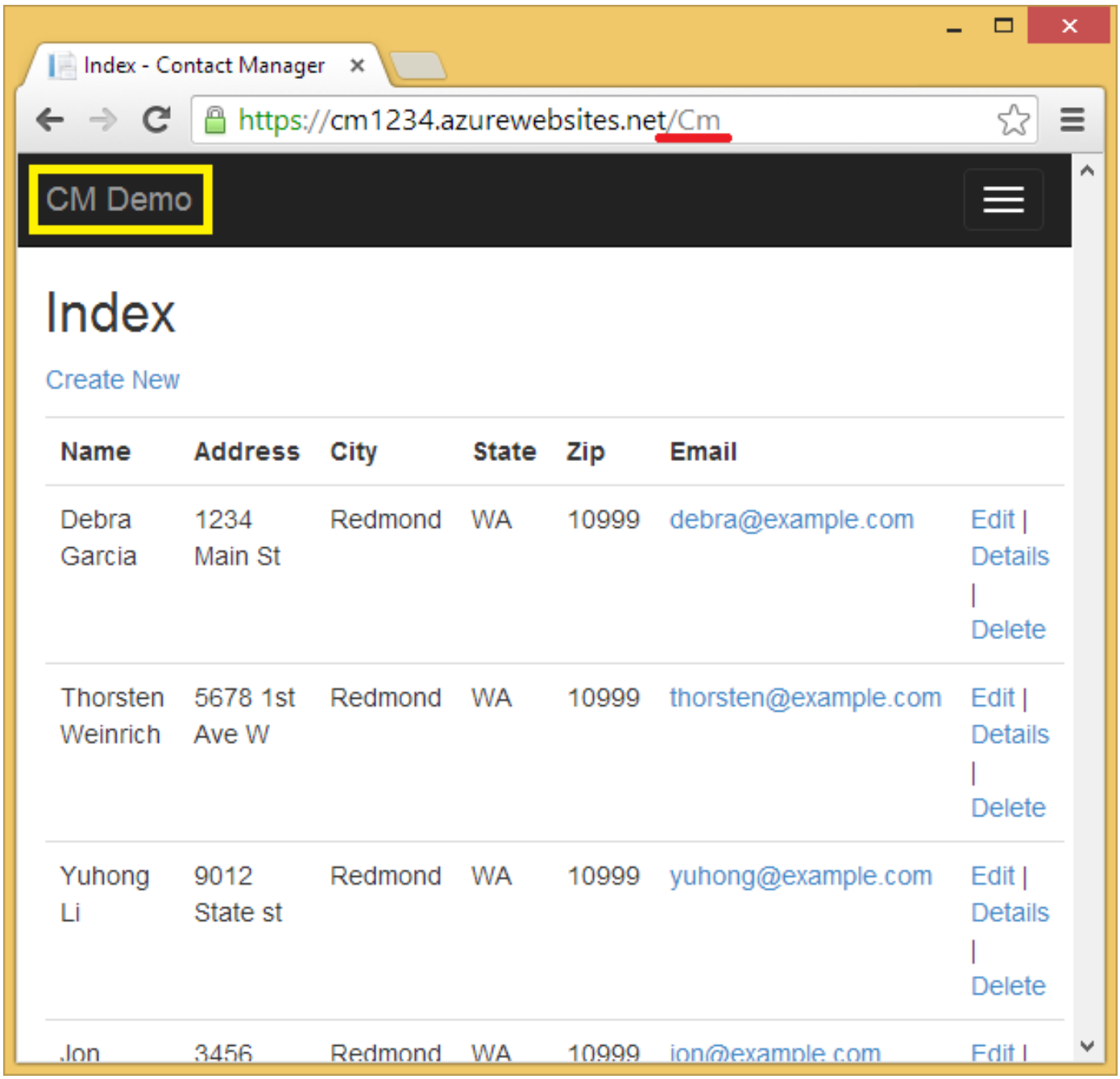
روی لینک **Register** در صفحه ورود کلیک کنید و یک حساب کاربری محلی بسازید. از این حساب کاربری برای این استفاده می‌کنیم که ببینیم شما به صفحات فقط خواندنی (read-only) و نه صفحاتی که داده‌ها را تغییر می‌دهند دسترسی دارید یا خیر. بعدا در ادامه مقاله، دسترسی حساب‌های کاربری محلی (local) را حذف می‌کنیم.



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.

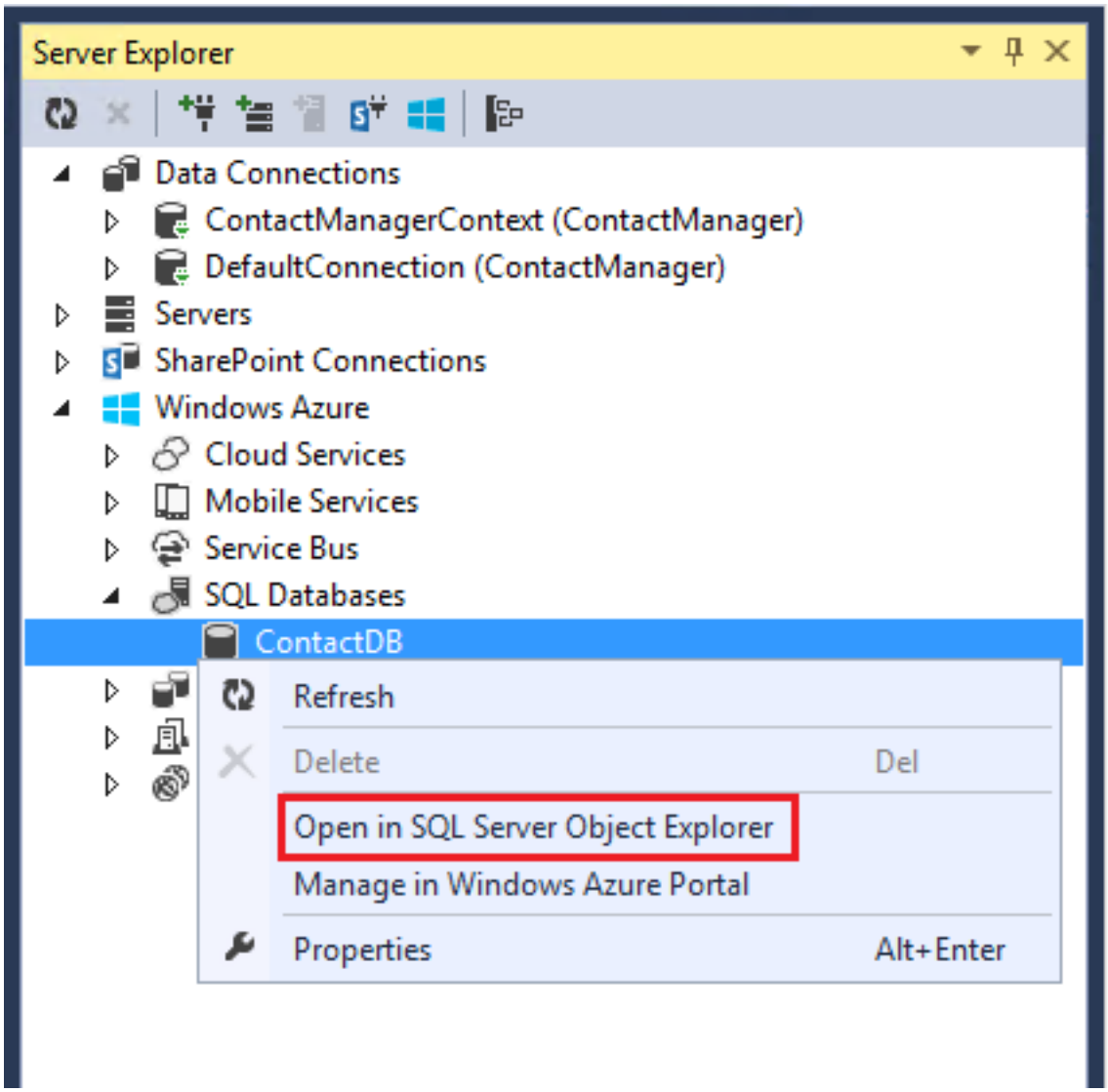
حال بررسی کنید که امکان ویرایش اطلاعات را دارید یا خیر.

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

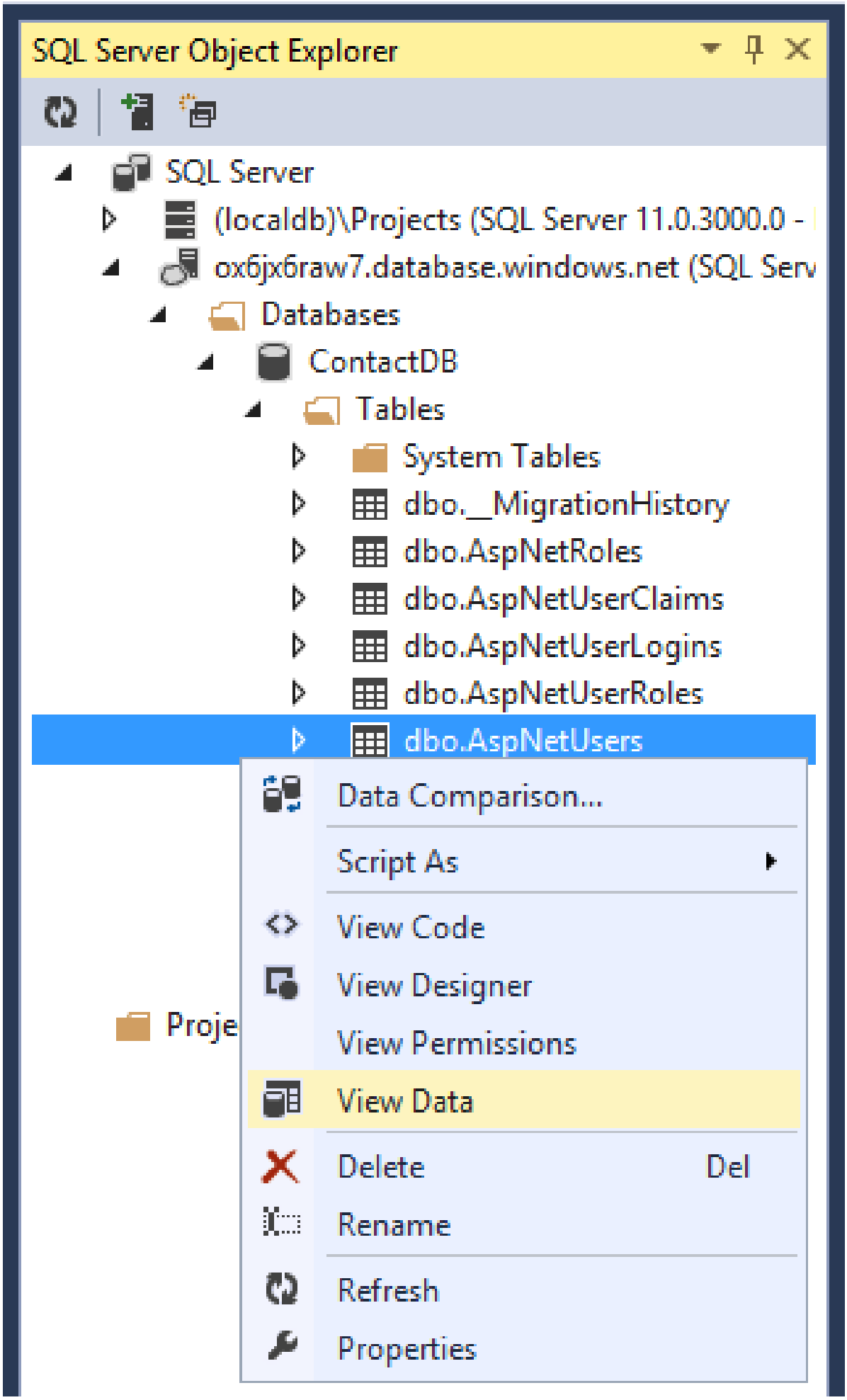
در **Server Explorer** دیتابیس **ContactDB** را پیدا کنید. روی آن کلیک راست کرده و **Open in SQL Server Object Explorer** را

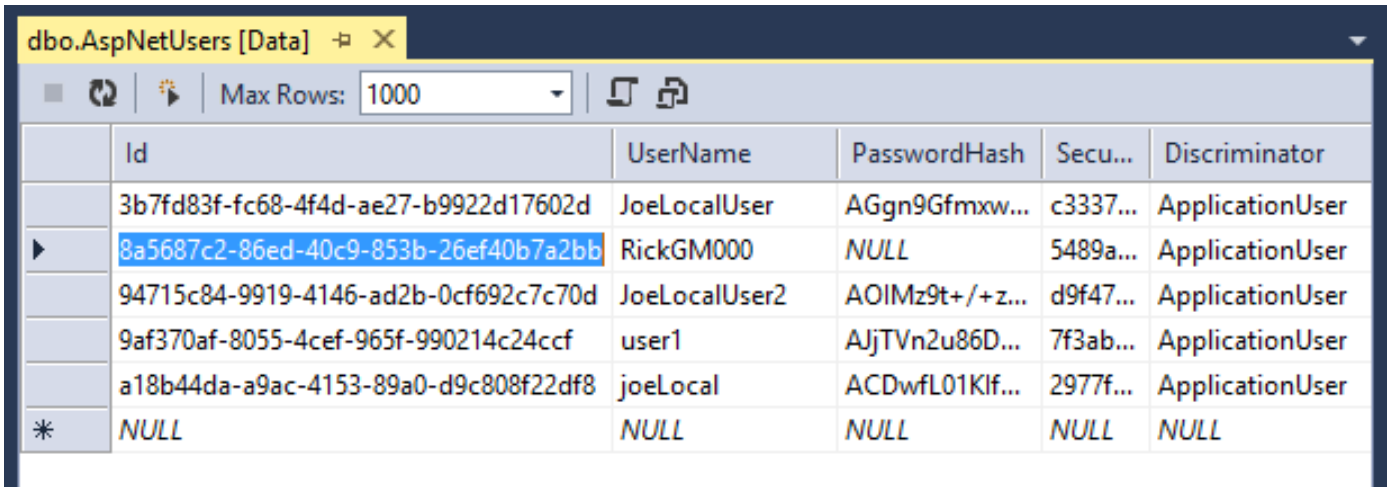
انتخاب کنید.



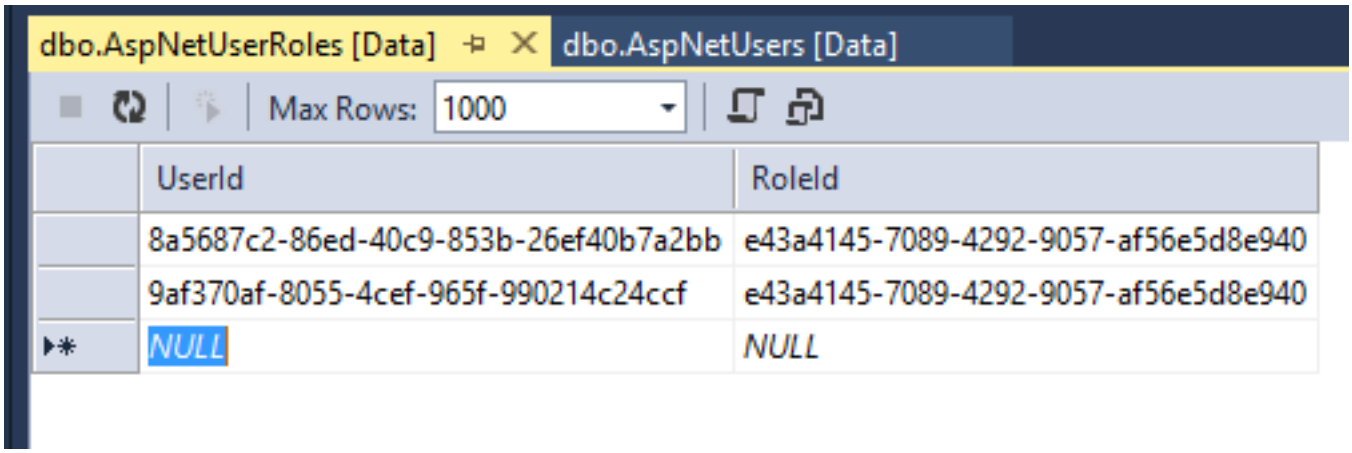
توجه: اگر نمی‌توانید گره **SQL Databases** را باز کنید و یا **ContactDB** را در ویژوال استودیو نمی‌بینید، باید مراحلی را طی کنید تا یک پورت یا یکسری پورت را به فایروال خود اضافه کنید. دقت داشته باشید که در صورت اضافه کردن Port Rangeها ممکن است چند دقیقه زمان نیاز باشد تا بتوانید به دیتابیس دسترسی پیدا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



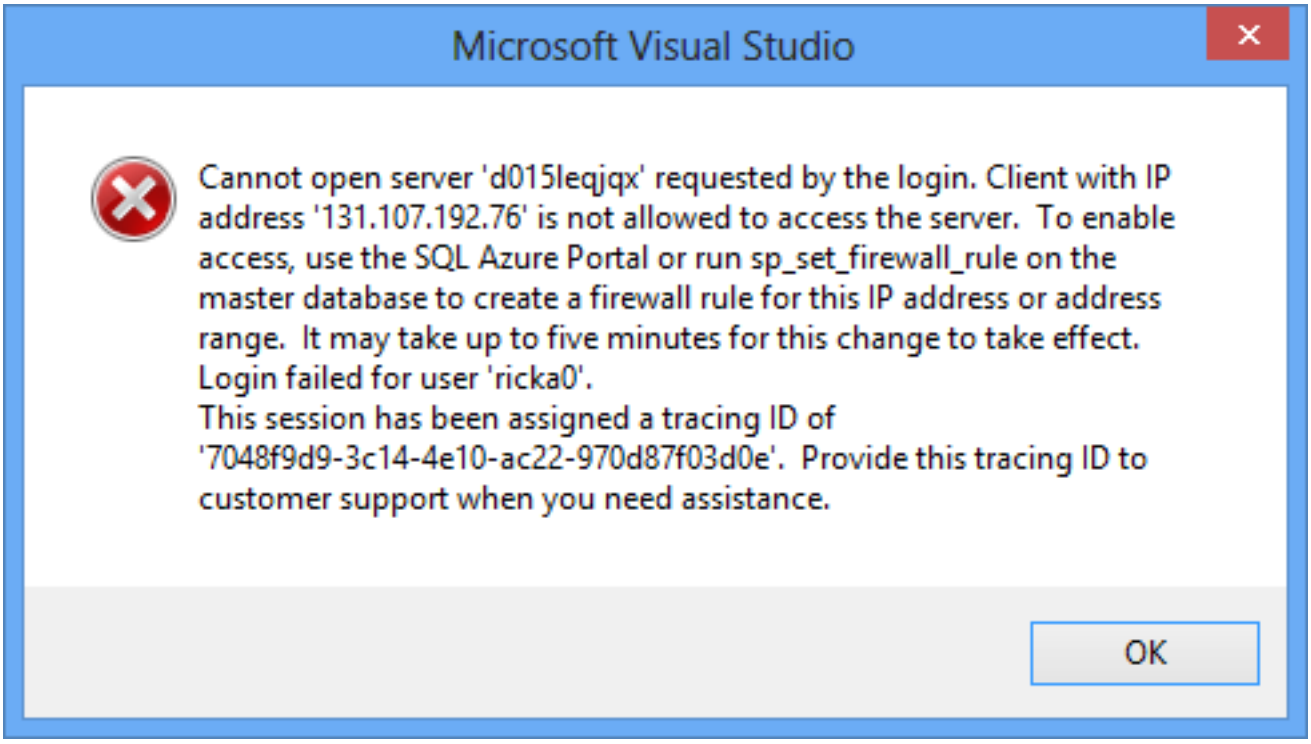
حالا روی **AspNetUserRoles** کلیک راست کنید و **View Data** را انتخاب کنید.



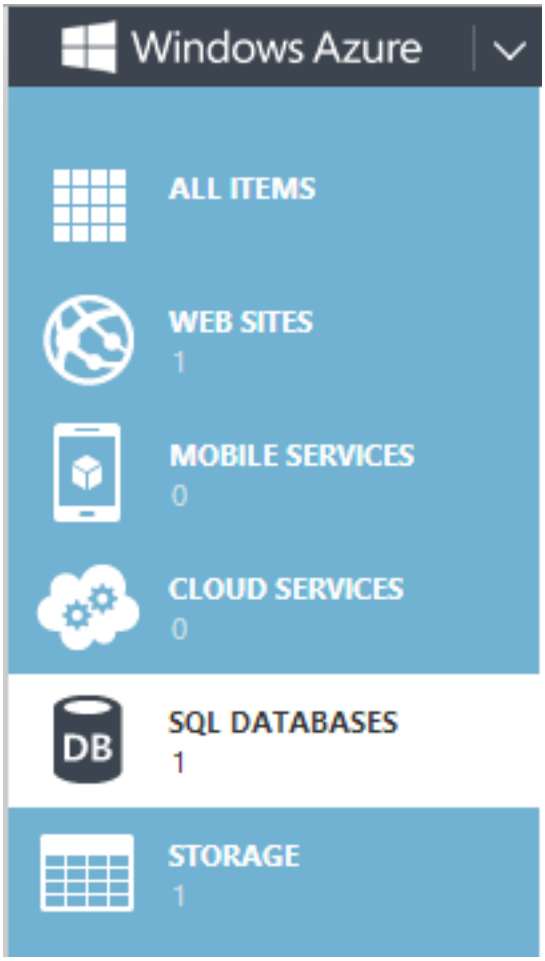
اگر شناسه کاربران (User ID) را بررسی کنید، مشاهده می‌کنید که تنها دو کاربر *user1* و اکانت گوگل شما به نقش *canEdit* تعلق دارند.

Cannot open server login error

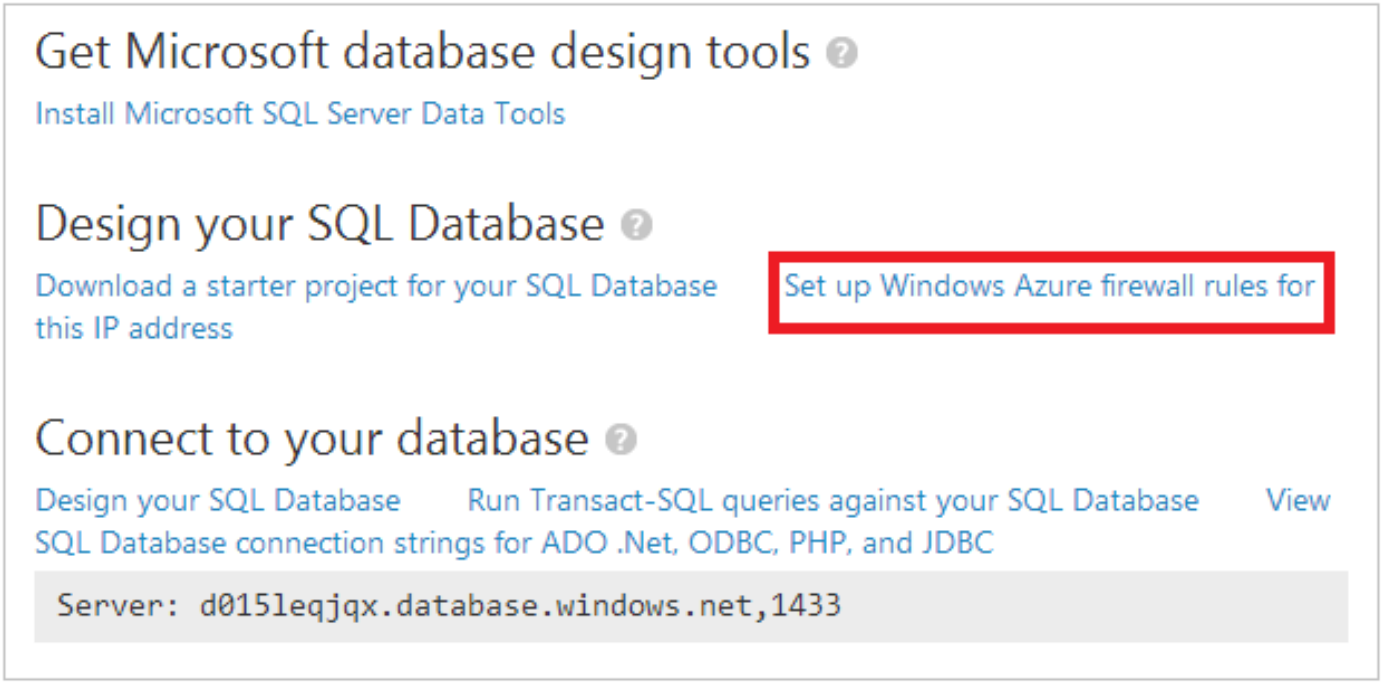
اگر خطایی مبنی بر "Cannot open server" دریافت می‌کنید، مراحل زیر را دنبال کنید.



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.

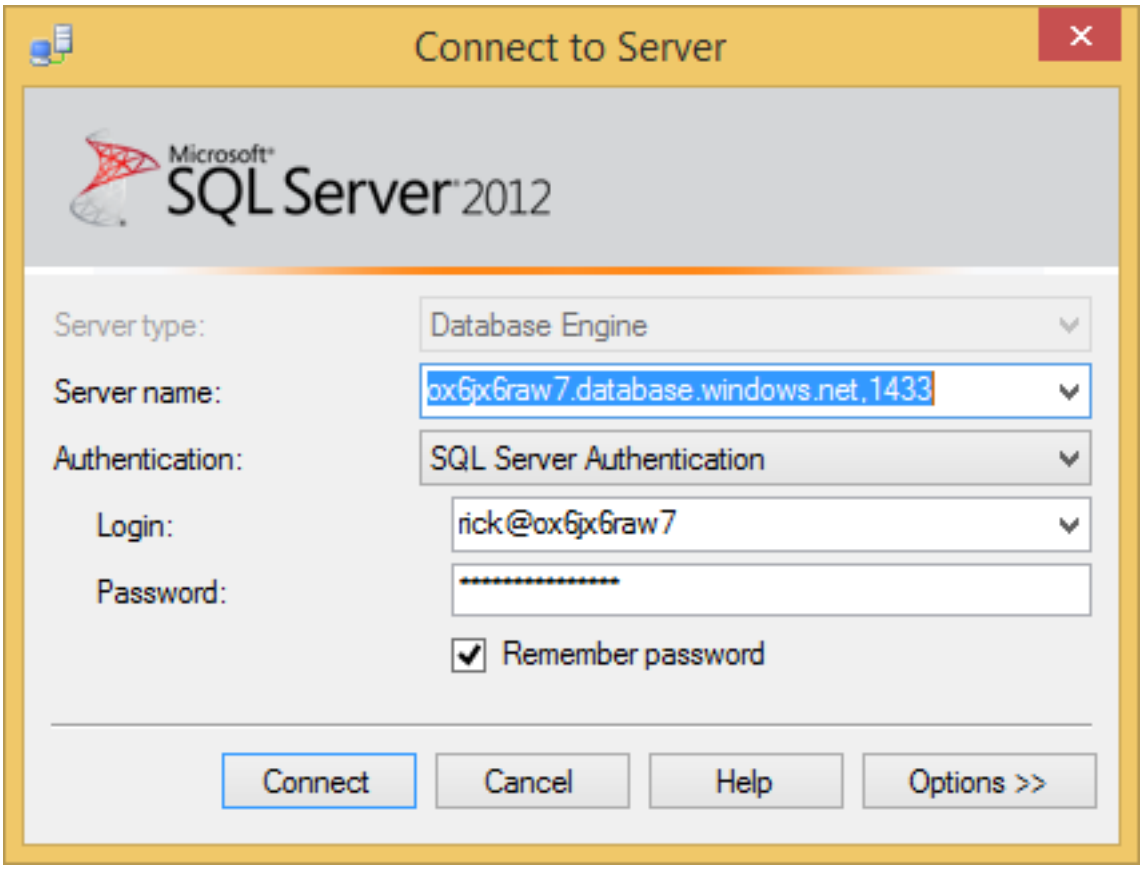
در بالای صفحه لینک **Configure** را کلیک کنید. حالا نام rule جدید، آدرس شروع و پایان را وارد کنید.

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.

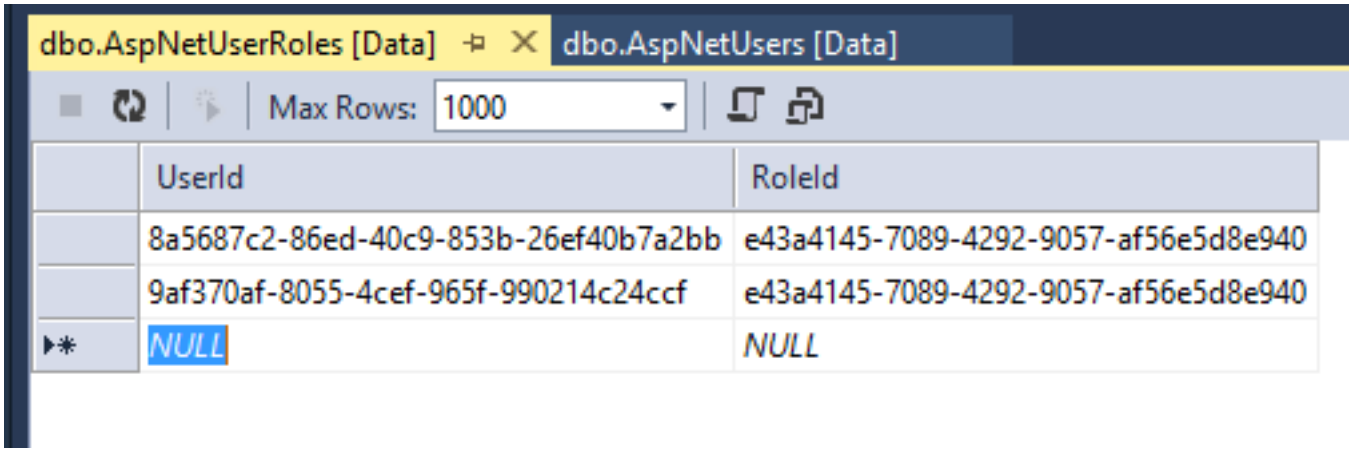


حالا می‌توانید با مراحلی که پیشتر توضیح داده شد به دیتابیس **Contact DB** مراجعه کنید.

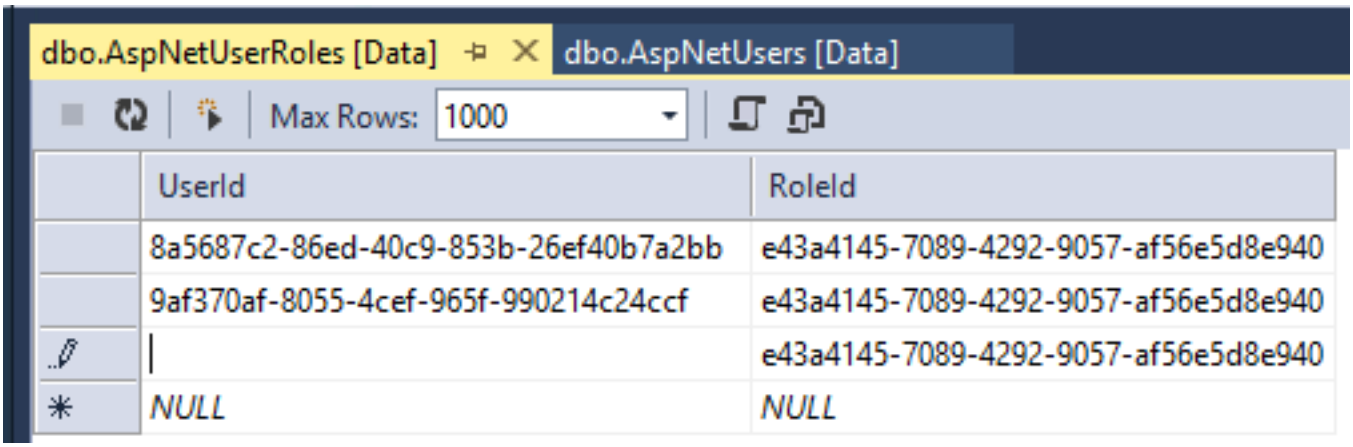
افزودن کاربران به نقش canEdit با ویرایش جداول دیتابیس

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: **ساخت یک اپلیکیشن ساده ToDo با ASP.NET Identity**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۹ ۱۳:۳۵
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, Tips, ASP.NET MVC 5, ASP.NET Identity, MVC

یک سناریوی فرضی را در نظر بگیرید. اگر بخواهیم IdentityDbContext و دیگر DbContextهای اپلیکیشن را ادغام کنیم چه باید کرد؟ مثلا یک سیستم وبلاگ که برخی کاربران می‌توانند پست جدید ثبت کنند، برخی تنها می‌توانند کامنت بگذارند و تمامی کاربران هم اختیارات مشخص دیگری دارند. در چنین سیستمی شناسه کاربران (User ID) در بسیاری از مدل‌ها (موجودیت‌ها و مدل‌های اپلیکیشن) وجود خواهد داشت تا مشخص شود هر رکورد به کدام کاربر متعلق است. در این مقاله چنین سناریو هایی را بررسی می‌کنیم و best practiceهای مربوطه را مرور می‌کنیم.
در این پست یک اپلیکیشن ساده ToDo خواهیم ساخت که امکان تخصیص to-doها به کاربران را نیز فراهم می‌کند. در این مثال خواهیم دید که چگونه می‌توان مدل‌های مختص به سیستم عضویت (IdentityDbContext) را با مدل‌های دیگر اپلیکیشن مخلوط و استفاده کنیم.

تعریف نیازمندی‌های اپلیکیشن

تنها کاربران احراز هویت شده قادر خواهند بود تا لیست ToDoهای خود را ببینند، آیتم‌های جدید ثبت کنند یا داده‌های قبلی را ویرایش و حذف کنند.
کاربران نباید آیتم‌های ایجاد شده توسط دیگر کاربران را ببینند.
تنها کاربرانی که به نقش Admin تعلق دارند باید بتوانند تمام ToDoهای ایجاد شده را ببینند.

پس بگذارید ببینیم چگونه می‌شود اپلیکیشنی با ASP.NET Identity ساخت که پاسخگوی این نیازمندی‌ها باشد.
ابتدا یک پروژه ASP.NET MVC جدید با مدل احراز هویت Individual User Accounts بسازید. در این اپلیکیشن کاربران قادر خواهند بود تا بصورت محلی در وب سایت ثبت نام کنند و یا با تامین کنندگان دیگری مانند گوگل و فیسبوک وارد سایت شوند. برای ساده نگاه داشتن این پست ما از حساب‌های کاربری محلی استفاده می‌کنیم.
در مرحله بعد ASP.NET Identity را راه اندازی کنید تا بتوانیم نقش مدیر و یک کاربر جدید بسازیم. می‌توانید با اجرای اپلیکیشن راه اندازی اولیه را انجام دهید. از آنجا که سیستم ASP.NET Identity توسط Entity Framework مدیریت می‌شود می‌توانید از تنظیمات پیکربندی Code First برای راه اندازی دیتابیس خود استفاده کنید.
در قدم بعدی راه انداز دیتابیس را در Global.asax تعریف کنید.

```
Database.SetInitializer<MyDbContext>(new MyDbInitializer());
```

ایجاد نقش مدیر و کاربر جدیدی که به این نقش تعلق دارد

اگر به قطعه کد زیر دقت کنید، می‌بینید که در خط شماره 5 متغیری از نوع UserManager ساخته ایم که امکان اجرای عملیات گوناگونی روی کاربران را فراهم می‌کند. مانند ایجاد، ویرایش، حذف و اعتبارسنجی کاربران. این کلاس که متعلق به سیستم ASP.NET Identity است همتای SQLMembershipProvider در ASP.NET 2.0 است.
در خط 6 یک RoleManager می‌سازیم که امکان کار با نقش‌ها را فراهم می‌کند. این کلاس همتای SQLRoleMembershipProvider در ASP.NET 2.0 است.
در این مثال نام کلاس کاربران (موجودیت کاربر در IdentityDbContext) برابر با "MyUser" است، اما نام پیش فرض در قالب‌های پروژه VS 2013 برابر با "ApplicationUser" می‌باشد.

```
public class MyDbInitializer : DropCreateDatabaseAlways<MyDbContext>
    {
        protected override void Seed(MyDbContext context)
        {
            var UserManager = new UserManager<MyUser>(new

                                                UserStore<MyUser>(context));

            var RoleManager = new RoleManager<IdentityRole>(new
                                            RoleStore<IdentityRole>(context));
```

```
            string name = "Admin";
            string password = "123456";


            //Create Role Admin if it does not exist
            if (!RoleManager.RoleExists(name))
            {
                var roleresult = RoleManager.Create(new IdentityRole(name));
            }

            //Create User=Admin with password=123456
            var user = new MyUser();
            user.UserName = name;
            var adminresult = UserManager.Create(user, password);

            //Add User Admin to Role Admin
            if (adminresult.Succeeded)
            {
                var result = UserManager.AddToRole(user.Id, name);
            }
            base.Seed(context);
        }
    }
```

حال فایلی با نام Models/AppModels.cs بسازید و مدل EF Code First اپلیکیشن را تعریف کنید. از آنجا که از EF استفاده می‌کنیم، روابط کلیدها بین کاربران و ToDoها بصورت خودکار برقرار می‌شود.

```
public class MyUser : IdentityUser
    {
        public string HomeTown { get; set; }
        public virtual ICollection<ToDo>
                              ToDoes { get; set; }
    }

    public class ToDo
    {
        public int Id { get; set; }
        public string Description { get; set; }
        public bool IsDone { get; set; }
        public virtual MyUser User { get; set; }
    }
```

در قدم بعدی با استفاده از مکانیزم Scaffolding کنترلر جدیدی بهمراه تمام Viewها و متدهای لازم برای عملیات CRUD بسازید.
برای اطلاعات بیشتر درباره نحوه استفاده از مکانیزم Scaffolding به این لینک مراجعه کنید.
لطفا دقت کنید که از DbContext فعلی استفاده کنید. این کار مدیریت داده‌های Identity و اپلیکیشن شما را یکپارچه‌تر می‌کند.
DbContext شما باید چیزی شبیه به کد زیر باشد.

```
public class MyDbContext : IdentityDbContext<MyUser>
    {
        public MyDbContext()
            : base("DefaultConnection")
        {
         }

         protected override void OnModelCreating(DbModelBuilder modelBuilder)
         {
        public System.Data.Entity.DbSet<AspnetIdentitySample.Models.ToDo>
                    ToDoes { get; set; }
    }
```

تنها کاربران احراز هویت شده باید قادر به اجرای عملیات CRUD باشند

برای این مورد از خاصیت Authorize استفاده خواهیم کرد که در MVC 4 هم وجود داشت. برای اطلاعات بیشتر لطفا به این لینک مراجعه کنید.

```
[Authorize]
public class ToDoController : Controller
```

کنترلر ایجاد شده را ویرایش کنید تا کاربران را به ToDoها اختصاص دهد. در این مثال تنها اکشن متدهای Create و List را بررسی خواهیم کرد. با دنبال کردن همین روش می‌توانید متدهای Edit و Delete را هم بسادگی تکمیل کنید.
یک متد constructor جدید بنویسید که آبجکتی از نوع UserManager می‌پذیرد. با استفاده از این کلاس می‌توانید کاربران را در ASP.NET Identity مدیریت کنید.

```
private MyDbContext db;
          private UserManager<MyUser> manager;
          public ToDoController()
          {
              db = new MyDbContext();
              manager = new UserManager<MyUser>(new UserStore<MyUser>(db));
          }
```

اکشن متد Create را بروز رسانی کنید

هنگامی که یک ToDo جدید ایجاد می‌کنید، کاربر جاری را در ASP.NET Identity پیدا می‌کنیم و او را به ToDoها اختصاص می‌دهیم.

```
public async Task<ActionResult> Create
          ([Bind(Include="Id,Description,IsDone")] ToDo todo)
          {
              var currentUser = await manager.FindByIdAsync
                                                  (User.Identity.GetUserId());
              if (ModelState.IsValid)
              {
                  todo.User = currentUser;
                  db.ToDoes.Add(todo);
                  await db.SaveChangesAsync();
                  return RedirectToAction("Index");
              }

              return View(todo);
          }
```

اکشن متد List را بروز رسانی کنید

در این متد تنها ToDoهای کاربر جاری را باید بگیریم.

```
public ActionResult Index()
          {
              var currentUser = manager.FindById(User.Identity.GetUserId());

               return View(db.ToDoes.ToList().Where(
                                  todo => todo.User.Id == currentUser.Id));
          }
```

تنها مدیران سایت باید بتوانند تمام ToDoها را ببینند

بدین منظور ما یک اکشن متد جدید به کنترل مربوطه اضافه می‌کنیم که تمام ToDoها را لیست می‌کند. اما دسترسی به این متد را تنها برای کاربرانی که در نقش مدیر وجود دارند میسر می‌کنیم.

```
[Authorize(Roles="Admin")]
          public async Task<ActionResult> All()
          {
```

```
            return View(await db.ToDoes.ToListAsync());
        }
```

نمایش جزئیات کاربران از جدول ToDo ها

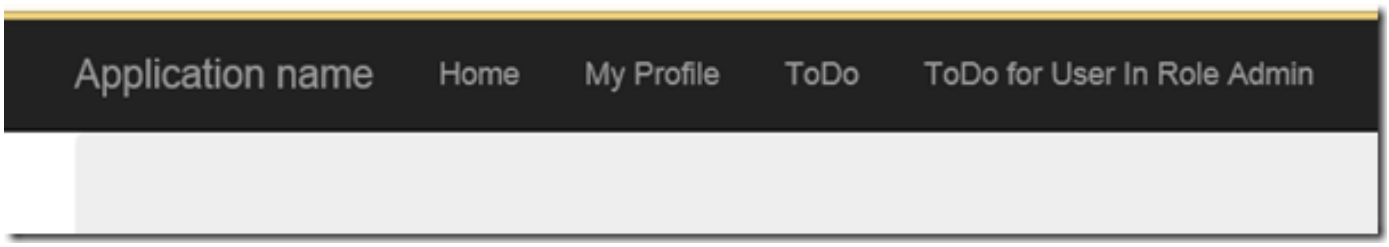
از آنجا که ما کاربران را به ToDo هایشان مرتبط می‌کنیم، دسترسی به داده‌های کاربر ساده است. مثلا در متدی که مدیر سایت تمام آیتم‌ها را لیست می‌کند می‌توانیم به اطلاعات پروفایل تک تک کاربران دسترسی داشته باشیم و آنها را در نمای خود بگنجانیم. در این مثال تنها یک فیلد بنام **HomeTown** اضافه شده است، که آن را در کنار اطلاعات ToDo نمایش می‌دهیم.

```
@model IEnumerable<AspnetIdentitySample.Models.ToDo>

  @{
    ViewBag.Title = "Index";
  }

  <h2>List of ToDoes for all Users</h2>
  <p>
      Notice that we can see the User info (UserName) and profile info such as HomeTown for the user as
well.
      This was possible because we associated the User object with a ToDo object and hence
      we can get this rich behavior.
  12:  </p>

  <table class="table">
      <tr>
          <th>
              @Html.DisplayNameFor(model => model.Description)
          </th>
          <th>
              @Html.DisplayNameFor(model => model.IsDone)
          </th>
          <th>@Html.DisplayNameFor(model => model.User.UserName)</th>
          <th>@Html.DisplayNameFor(model => model.User.HomeTown)</th>
      </tr>
  25:
  26:       @foreach (var item in Model)
  27:       {
  28:           <tr>
  29:               <td>
  30:                   @Html.DisplayFor(modelItem => item.Description)
  31:               </td>
  32:               <td>
                   @Html.DisplayFor(modelItem => item.IsDone)
              </td>
              <td>
                  @Html.DisplayFor(modelItem => item.User.UserName)
              </td>
              <td>
                  @Html.DisplayFor(modelItem => item.User.HomeTown)
              </td>
          </tr>
      }

  </table>
```

صفحه Layout را بروز رسانی کنید تا به ToDoها لینک شود

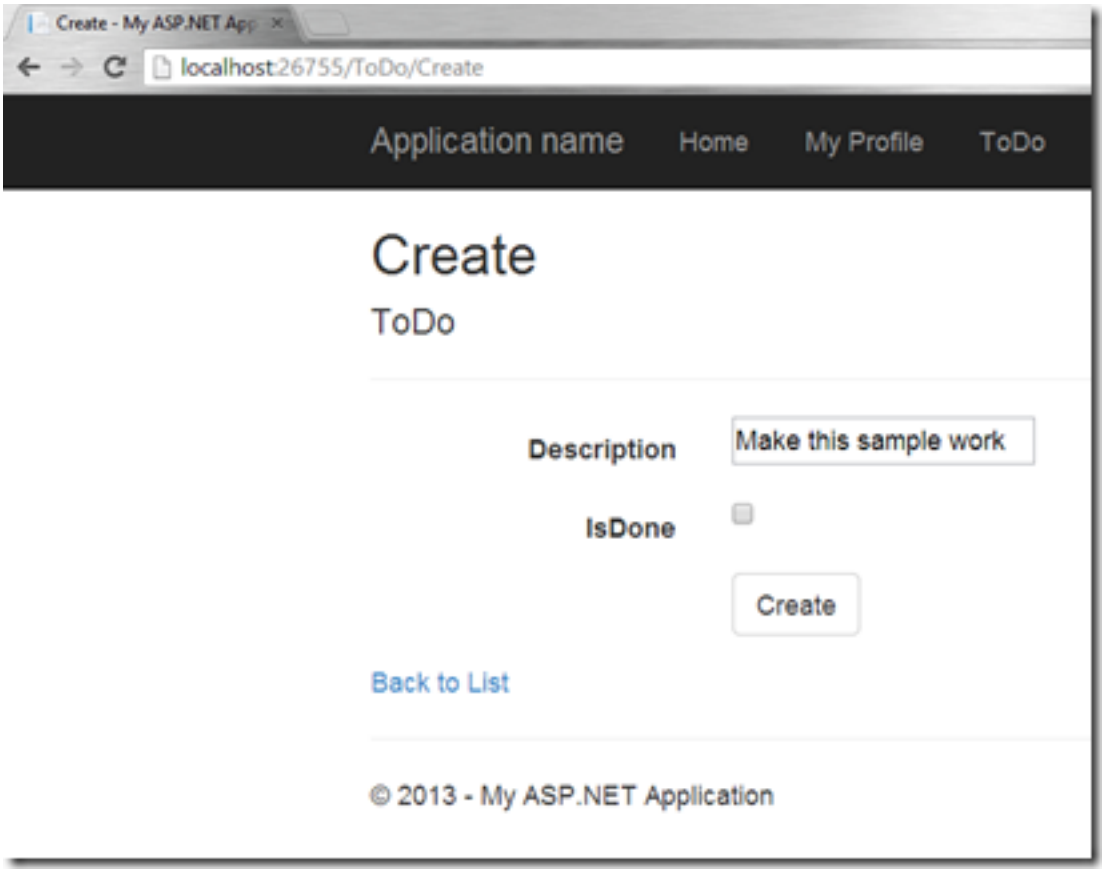
```
<li>@Html.ActionLink("ToDo", "Index", "ToDo")</li>
 <li>@Html.ActionLink("ToDo for User In Role Admin", "All", "ToDo")</li>
```

حال اپلیکیشن را اجرا کنید. همانطور که مشاهده می‌کنید دو لینک جدید به منوی سایت اضافه شده اند.

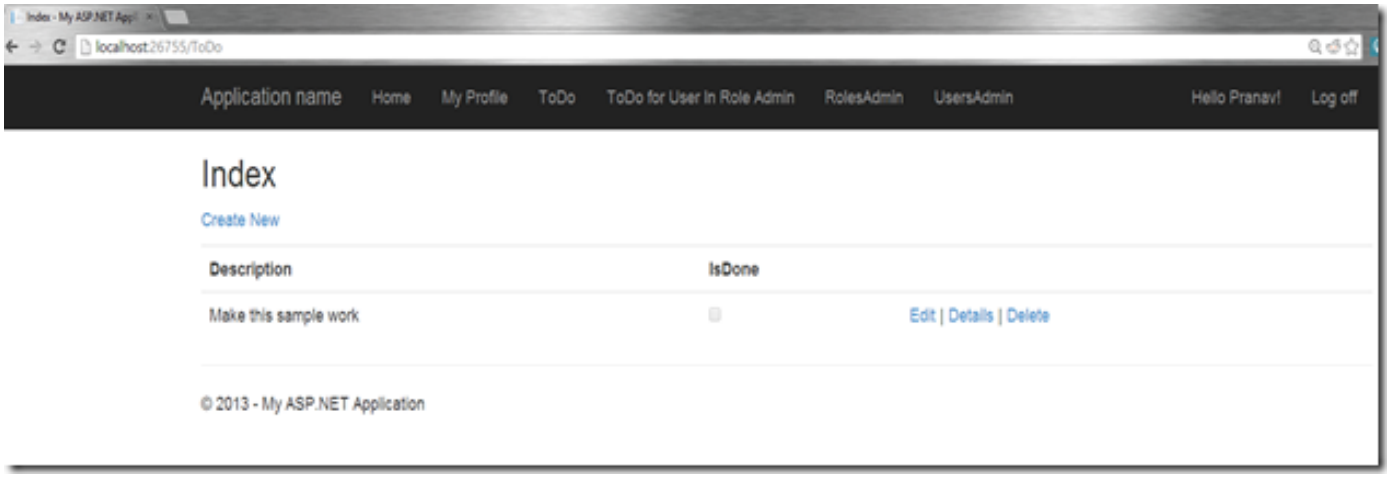


ساخت یک ToDo بعنوان کاربر عادی

روی لینک ToDo کلیک کنید، باید به صفحه ورود هدایت شوید چرا که دسترسی تنها برای کاربران احراز هویت شده تعریف وجود دارد. می‌توانید یک حساب کاربری محلی ساخته، با آن وارد سایت شوید و یک ToDo بسازید.



پس از ساختن یک ToDo می‌توانید لیست رکوردهای خود را مشاهده کنید. دقت داشته باشید که رکوردهایی که کاربران دیگر ثبت کرده اند برای شما نمایش داده نخواهند شد.

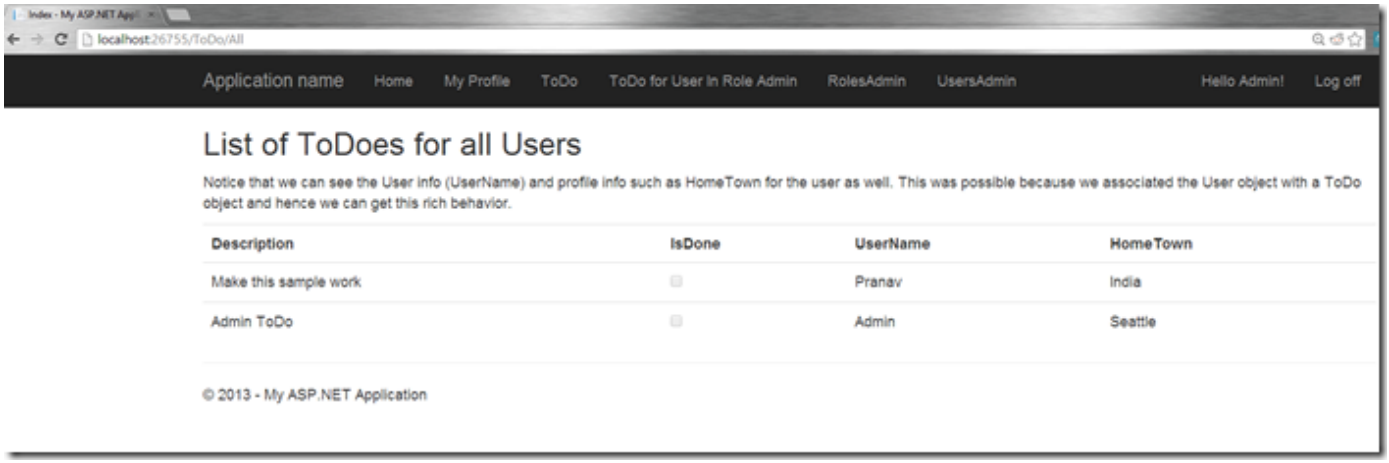


مشاهده تمام ToDoها بعنوان مدیر سایت

روی لینک **ToDoes for User in Role Admin** کلیک کنید. در این مرحله باید مجددا به صفحه ورود هدایت شوید چرا که شما در نقش مدیر نیستید و دسترسی کافی برای مشاهده صفحه مورد نظر را ندارید. از سایت خارج شوید و توسط حساب کاربری مدیری که هنگام راه اندازی اولیه دیتابیس ساخته اید وارد سایت شوید.

```
User = Admin
Password = 123456
```

پس از ورود به سایت بعنوان یک مدیر، می‌توانید ToDoهای ثبت شده توسط تمام کاربران را مشاهده کنید.



نظرات خوانندگان

نویسنده: اس ام
تاریخ: ۱۳۹۳/۰۱/۰۱ ۲۰:۲۴

میشه این پروژه رو برا دانلود بزارید؟ ممنون.

نویسنده: میثم سلیمانی
تاریخ: ۱۳۹۳/۰۴/۲۸ ۱۷:۴۰

```
var currentUser = await manager.FindByIdAsync(User.Identity.GetUserId());
```

این GetUserId() چرا وجود نداره؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۲۸ ۱۷:۴۹

در فضای نام Microsoft.AspNet.Identity تعریف شده.

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; }
    }
```

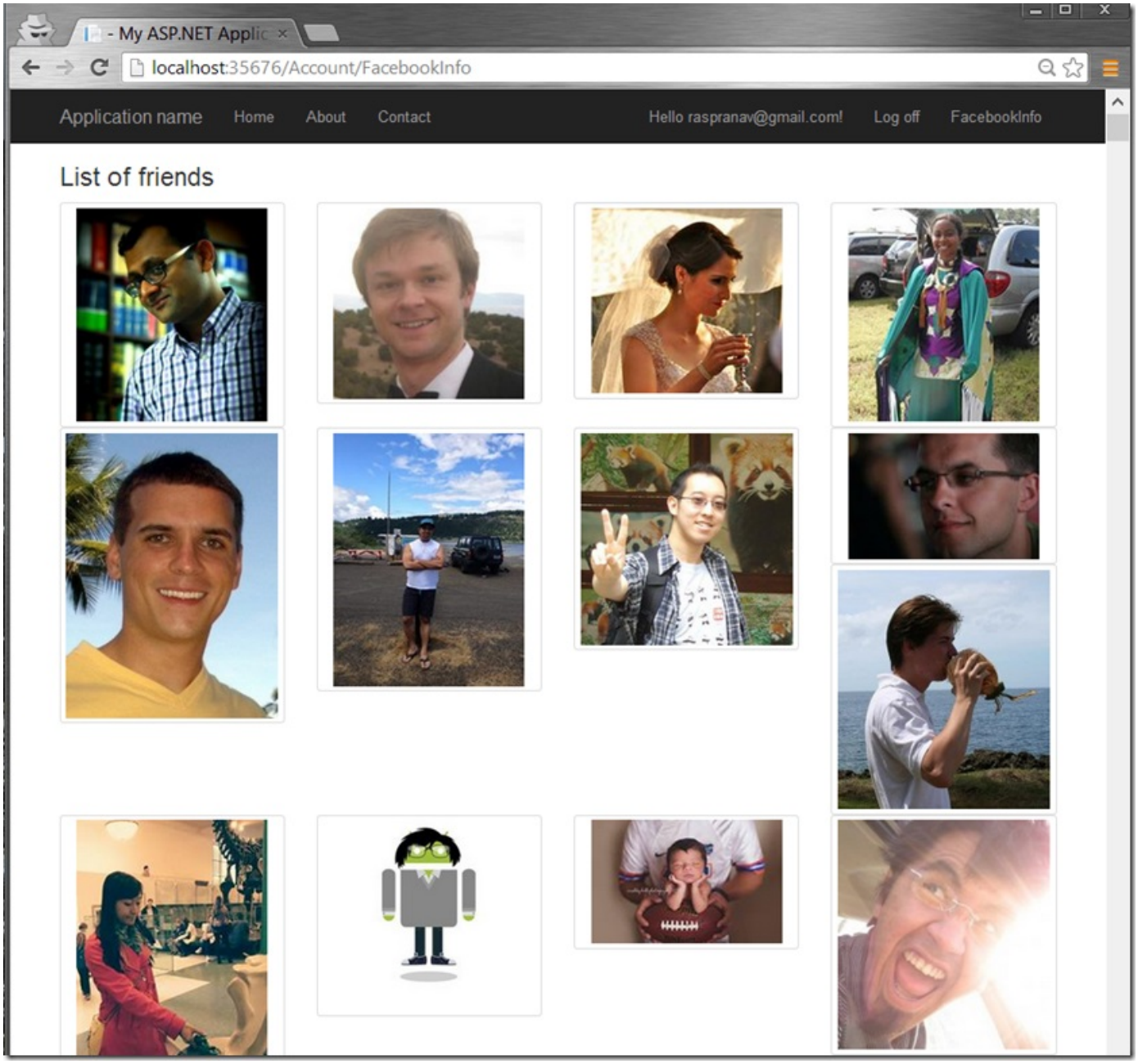
کد زیر را به کنترلر Account اضافه کنید تا عکس‌های دوستان تان را دریافت کنید.

```
//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.

عنوان: **سفارشی کردن ASP.NET Identity در MVC 5**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۹ ۱۹:۵۰
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, MemberShip, ASP.NET MVC 5, ASP.NET Identity, MVC, Security

یکی از نیازهای رایج توسعه دهندگان هنگام استفاده از سیستم عضویت ASP.NET سفارشی کردن الگوی داده‌ها است. مثلا ممکن است بخواهید یک پروفایل سفارشی برای کاربران در نظر بگیرید، که شامل اطلاعات شخصی، آدرس و تلفن تماس و غیره می‌شود. یا ممکن است بخواهید به خود فرم ثبت نام فیلدهای جدیدی اضافه کنید و آنها را در رکورد هر کاربر ذخیره کنید.
یکی از مزایای ASP.NET Identity این است که بر پایه EF Code First نوشته شده است. بنابراین سفارشی سازی الگوی دیتابیس و اطلاعات کاربران ساده است.
یک اپلیکیشن جدید ASP.NET MVC بسازید و نوع احراز هویت را Individual User Accounts انتخاب کنید. پس از آنکه پروژه جدید ایجاد شد فایل IdentityModels.cs را در پوشه Models باز کنید. کلاسی با نام *ApplicationUser* مشاهده می‌کنید که همتای *UserProfile* در فریم ورک SimpleMembership است. این کلاس خالی است و از کلاس *IdentityUser* ارث بری می‌کند و شامل خواص زیر است.

```
public class IdentityUser : IUser
    {
        public IdentityUser();
        public IdentityUser(string userName);

        public virtual ICollection<identityuserclaim> Claims { get; }
        public virtual string Id { get; set; }
        public virtual ICollection<identityuserlogin> Logins { get; }
        public virtual string PasswordHash { get; set; }
        public virtual ICollection<identityuserrole> Roles { get; }
        public virtual string SecurityStamp { get; set; }
        public virtual string UserName { get; set; }
    }
```

اگر دقت کنید خواهید دید که فیلد Id بر خلاف SimpleMembership یک عدد صحیح یا int نیست، بلکه بصورت یک رشته ذخیره می‌شود. پیاده سازی پیش فرض ASP.NET Identity مقدار این فیلد را با یک GUID پر می‌کند. در این پست تنها یک فیلد آدرس ایمیل به کلاس کاربر اضافه می‌کنیم. با استفاده از همین فیلد در پست‌های آتی خواهیم دید چگونه می‌توان ایمیل‌های تایید ثبت نام برای کاربران ارسال کرد. کلاس ApplicationUser بدین شکل خواهد بود.

```
public class ApplicationUser : IdentityUser
{
    public string Email { get; set; }
}
```

حال برای آنکه کاربر بتواند هنگام ثبت نام آدرس ایمیل خود را هم وارد کند، باید مدل فرم ثبت نام را بروز رسانی کنیم.

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

    [Required]
    [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength =
6)]
    [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; }

    [Required]
    [Display(Name = "Email address")]
```

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

}
```

سپس فایل View را هم بروز رسانی می‌کنیم تا یک برچسب و تکست باکس برای آدرس ایمیل نمایش دهد.

```
<div class="form-group">
     @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label" })
     <div class="col-md-10">
         @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
     </div>
 </div>
```

برای تست این تغییرات، صفحه About را طوری تغییر می‌دهید تا آدرس ایمیل کاربر جاری را نمایش دهد. این قسمت همچنین نمونه ای از نحوه دسترسی به اطلاعات کاربران است.

```
public ActionResult About()
{
    ViewBag.Message = "Your application description page.";
    UserManager<ApplicationUser> UserManager = new UserManager<ApplicationUser>(new
UserStore<ApplicationUser>(new ApplicationDbContext()));
    var user = UserManager.FindById(User.Identity.GetUserId());
    if (user != null)
        ViewBag.Email = user.Email;
    else
        ViewBag.Email = "User not found.";

    return View();
}
```

همین! تمام کاری که لازم بود انجام دهید همین بود. از آنجا که سیستم ASP.NET Identity توسط Entity Framework مدیریت می‌شود، روی الگوی دیتابیس سیستم عضویت کنترل کامل دارید. بنابراین به سادگی می‌توانید با استفاده از قابلیت Code First مدل‌های خود را سفارشی کنید.

در پست‌های آتی این مطلب را ادامه خواهیم داد تا ببینیم چگونه می‌توان ایمیل‌های تاییدیه برای کاربران ارسال کرد.

نظرات خوانندگان

نویسنده: رجایی
تاریخ: ۱۳۹۲/۱۱/۰۸ ۱۹:۲۰

سلام؛ از زحماتتون بسیار تشکر می‌کنم. من وب سایت را به روش چند لایه‌ی مرسوم می‌سازم:

data layer - domain models - website - service layer

با توجه به آن چگونه می‌توان از asp.net identity استفاده کرد؟ زیرا مثلا نیاز است از کلید جدول users در مدل‌های دیگه استفاده شود. آیا این کلید را باید در لایه دومین استفاده کرد؟

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۱/۰۸ ۲۳:۴۵

موجودیت‌های مربوط به ASP.NET Identity رو در لایه مدل‌ها قرار بدین و از یک DbContext استفاده کنید. یعنی DbSetهای مدل برنامه و Identity رو در یک کانتکست تعریف کنید.

نویسنده: رجایی
تاریخ: ۱۳۹۲/۱۱/۱۰ ۱۲:۴

سلام...ممنون از پاسختون.

با توجه به راهنمایی شما در قسمت context در لایه data layer بدین صورت درج کردم

```
public class Context : DbContext
    {
        public DbSet<IdentityUserClaim> AspNetUserClaims { set; get; }
        public DbSet<IdentityRole> AspNetRoles { set; get; }
        public DbSet<IdentityUserLogin> AspNetUserLogins { set; get; }
        public DbSet<IdentityUserRole> AspNetUserRoles { set; get; }
        public DbSet<IdentityUser> AspNetUsers { set; get; }
        //-------------------------------------
        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);
            modelBuilder.Entity<IdentityRole>().HasKey<string>(r => r.Id).ToTable("AspNetRoles");
            modelBuilder.Entity<IdentityUser>().ToTable("AspNetUsers");
            modelBuilder.Entity<IdentityUserLogin>().HasKey(l => new { l.UserId, l.LoginProvider,
l.ProviderKey }).ToTable("AspNetUserLogins");
            modelBuilder.Entity<IdentityUserRole>().HasKey(r => new { r.RoleId, r.UserId
}).ToTable("AspNetUserRoles");
            modelBuilder.Entity<IdentityUserClaim>().ToTable("AspNetUserClaims");
        }
    }
```

ولی وقتی سایت اجرا می‌شود ایراد زیر نمایش داده می‌شود

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۱/۱۰ ۱۲:۱۴

خطا رو فراموش کردید ارسال کنید.

نویسنده: مهرداد راهی
تاریخ: ۱۳۹۲/۱۱/۱۱ ۰:۵۵

سلام من MVC کار نمیکنم توی ASP.Net وب‌فرمز استفاده میکنم از این امکان. فایل IdentityModels.cs رو نداره توی پروژه .
مشکل کجاس؟
-enable migrations هم زدم خطا داد

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۱/۱۱ ۱:۵۳

از VS 2013 استفاده می‌کنی ؟

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۱/۱۱ ۲:۵۱

لازم نیست تمام این آبجکت‌ها رو به context نگاشت کنید. قالب پروژه‌های VS 2013 بصورت خودکار در پوشه Models کلاسی بنام IdentityModels میسازه. این کلاس شامل کلاسی بنام ApplicationDbContext میشه که تعریفی مانند لیست زیر داره:

```
public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
{
    public ApplicationDbContext() : base("DefaultConnection") { }
}
```

این کلاس رو کلا حذف کنید، چون قراره از یک DbContext برای تمام موجودیت‌ها استفاده کنید.
کلاس ApplicationUser که معرف موجودیت کاربران هست رو در لایه دامنه‌ها تعریف کنید و دقت کنید که باید از IdentityUser ارث بری کنه، حال با نام پیش فرض یا با نام دلخواه. سپس باید کلاسی بسازید که از UserManager<T> مشتق میشه. با استفاده از این کلاس می‌تونید به موجودیت‌های کاربران دسترسی داشته باشید. بعنوان مثال:

```
public class AppUserManager : UserManager<AppUser>{
    public AppUserManager() : base(new UserStore<AppUser>(new ShirazBilitDbContext())) { }
}
```

همونطور که می‌بینید کلاس موجودیت کاربر در اینجا AppUser نام داره، پس هنگام استفاده از UserManager نوع داده رو بهش نگاشت می‌کنیم. کد کلاس AppUser هم مطابق لیست زیر خواهد بود.

```
public class AppUser : IdentityUser
{
    public string Email { get; set; }
    public string ConfirmationToken { get; set; }
    public bool IsConfirmed { get; set; }
}
```

همونطور که مشخصه کلاس کاربران سفارشی سازی شده و سه فیلد به جدول کاربران اضافه کردیم. فیلدهای بیشتر یا موجودیت پروفایل کاربران هم باید به همین کلاس افزوده بشن. اگر پست‌ها رو بیاد بیارید گفته شد که ASP.NET Identity با مدل EF Code-First کار میکنه.

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۱/۱۱ ۲:۵۷

از VS 2013 استفاده کنید و .NET 4.5.
اگر این فایل برای شما ایجاد نمیشه پس در قالب پروژه‌های Web Forms وجود نداره. ارتباطی با مهاجرت‌ها هم نداره، کلاس موجودیت کاربر رو خودتون می‌تونید ایجاد کنید. اگر به نظرات بالا مراجعه کنید گفته شد که کلاس کاربران باید از

IdentityModels ارث بری کنه.

نویسنده: رجایی
تاریخ: ۱۳۹۲/۱۱/۱۲ ۹:۲۱

عذر خواهی می‌کنم فراموش کردم. ایراد بدین صورت است:

System.InvalidOperationException: The model backing the 'Context' context has changed since the database was created. Consider using Code First Migrations to update the database (http://go.microsoft.com/fwlink/?LinkId=238269).

مشخص است که می‌گوید context تغییر می‌کند.ولی من از migration استفاده می‌کنم و codefirst ولی باز هم این ایراد رو در اتصال به دیتابیس نشان می‌دهد. من از add-migration هم استفاده می‌کنم تا تغییرات موجودیت‌ها رو کامل به من نشان دهد که چیزی را عنوان نمی‌کند.

نویسنده: افتاب
تاریخ: ۱۳۹۲/۱۲/۲۷ ۲۳:۳۸

سلام و متشکر ،
دنبال آن میگشتم که چه طوری میشه دو تا صفحه لوگین در پروژه داشت که ورودی کاربران از مدیران جدا باشد

نویسنده: افتاب
تاریخ: ۱۳۹۲/۱۲/۲۷ ۲۳:۴۴

میشه در مورد «این کلاس رو کلا حذف کنید، چون قراره از یک DbContext برای تمام موجودیت‌ها استفاده کنید....» بیشتر توضیح بفرمایید؟

نویسنده: سعید رضایی
تاریخ: ۱۳۹۳/۰۲/۰۱ ۱۶:۴۱

با عرض سلام.تو Vs2012+mvc4 نمیشه از identity استفاده کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۰۱ ۱۷:۲۶

خیر. با دات نت 4.5 کامپایل شده.

نویسنده: میثم سلیمانی
تاریخ: ۱۳۹۳/۰۵/۲۶ ۱۱:۷

با سلام و احترام
من دستور زیر رو تو asp.net mvc Identity sample 2 تو اکشن Login اضافه کردم

```
UserManager<ApplicationUser> UserManager = new UserManager<ApplicationUser>(new
UserStore<ApplicationUser>(new ApplicationDbContext()));
    var user = UserManager.FindById(User.Identity.GetUserId());
```

اما user و null میده !
اکشن login

```
public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
        {
            if (!ModelState.IsValid)
            {
                return View(model);
            }

            // This doen't count login failures towards lockout only two factor authentication
            // To enable password failures to trigger lockout, change to shouldLockout: true
            var result = await SignInManager.PasswordSignInAsync(model.Email, model.Password,
model.RememberMe, shouldLockout: false);
            switch (result)
            {

                case SignInStatus.Success:
                    {
                        UserManager<ApplicationUser> UserManager = new UserManager<ApplicationUser>(new
UserStore<ApplicationUser>(new ApplicationDbContext()));
                        var user = UserManager.FindById(User.Identity.GetUserId());
                        return RedirectToLocal(returnUrl);
                    }
                case SignInStatus.LockedOut:
                    return View("Lockout");
                case SignInStatus.RequiresVerification:
                    return RedirectToAction("SendCode", new { ReturnUrl = returnUrl });
                case SignInStatus.Failure:
                default:
                    ModelState.AddModelError("", "Invalid login attempt.");
                    return View(model);
            }
        }
```

عنوان: **افزودن تصدیق ایمیل به ASP.NET Identity در MVC 5**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۹ ۲۱:۴۵
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, Tips, ASP.NET MVC 5, ASP.NET Identity, MVC, Security

در پست قبلی نحوه سفارشی کردن پروفایل کاربران در ASP.NET Identity را مرور کردیم. اگر بیاد داشته باشید یک فیلد آدرس ایمیل به کلاس کاربر اضافه کردیم. در این پست از این فیلد استفاده میکنیم تا در پروسه ثبت نام ایمیل‌ها را تصدیق کنیم. بدین منظور پس از ثبت نام کاربران یک ایمیل فعالسازی برای آنها ارسال می‌کنیم که حاوی یک لینک است. کاربران با کلیک کردن روی این لینک پروسه ثبت نام خود را تایید می‌کنند و می‌توانند به سایت وارد شوند. پیش از تایید پروسه ثبت نام، کاربران قادر به ورود نیستند.

در ابتدا باید اطلاعات کلاس کاربر را تغییر دهید تا دو فیلد جدید را در بر گیرد. یک فیلد شناسه تایید (confirmation token) را ذخیره می‌کند، و دیگری فیلدی منطقی است که مشخص می‌کند پروسه ثبت نام تایید شده است یا خیر. پس کلاس *ApplicationUser* حالا باید بدین شکل باشد.

```
public class ApplicationUser : IdentityUser
{
    public string Email { get; set; }
    public string ConfirmationToken { get; set; }
    public bool IsConfirmed { get; set; }
}
```

اگر پیش از این کلاس ApplicationUser را تغییر داده اید، باید مهاجرت‌ها را فعال کنید و دیتابیس را بروز رسانی کنید. حالا می‌توانیم از این اطلاعات جدید در پروسه ثبت نام استفاده کنیم و برای کاربران ایمیل‌های تاییدیه را بفرستیم.

```
private string CreateConfirmationToken()
{
    return ShortGuid.NewGuid();
}

private void SendEmailConfirmation(string to, string username, string confirmationToken)
{
    dynamic email = new Email("RegEmail");
    email.To = to;
    email.UserName = username;
    email.ConfirmationToken = confirmationToken;
    email.Send();
}

//
// POST: /Account/Register
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        string confirmationToken = CreateConfirmationToken();
        var user = new ApplicationUser()
        {
            UserName = model.UserName,
            Email = model.Email,
            ConfirmationToken = confirmationToken,
                IsConfirmed = false };
        var result = await UserManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
            SendEmailConfirmation(model.Email, model.UserName, confirmationToken);
            return RedirectToAction("RegisterStepTwo", "Account");
        }
        else
        {
            AddErrors(result);
        }
    }

    // If we got this far, something failed, redisplay form
```

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

برای تولید شناسه‌های تایید (tokens) از کلاسی بنام *ShortGuid* استفاده شده است. این کلاس یک مقدار GUID را encode می‌کند که در نتیجه آن مقدار خروجی کوتاه‌تر بوده و برای استفاده در URLها ایمن است. کد این کلاس را از این وبلاگ گرفته ام. پس از ایجاد حساب کاربری باید شناسه تولید شده را به آن اضافه کنیم و مقدار فیلد IsConfirmed را به false تنظیم کنیم. برای تولید ایمیل‌ها من از Postal استفاده می‌کنم. Postal برای ساختن ایمیل‌های دینامیک شما از موتور Razor استفاده می‌کند. می‌توانید ایمیل‌های ساده (plain text) یا HTML بسازید، عکس و فایل در آن درج و ضمیمه کنید و امکانات بسیار خوب دیگر. اکشن متد RegisterStepTwo تنها کاربر را به یک View هدایت می‌کند که پیامی به او نشان داده می‌شود.

بعد از اینکه کاربر ایمیل را دریافت کرد و روی لینک تایید کلیک کرد به اکشن متد *RegisterConfirmation* باز می‌گردیم.

```
private bool ConfirmAccount(string confirmationToken)
{
    ApplicationDbContext context = new ApplicationDbContext();
    ApplicationUser user =  context.Users.SingleOrDefault(u => u.ConfirmationToken ==
confirmationToken);
    if (user != null)
    {
        user.IsConfirmed = true;
        DbSet<ApplicationUser> dbSet = context.Set<ApplicationUser>();
        dbSet.Attach(user);
        context.Entry(user).State = EntityState.Modified;
        context.SaveChanges();

        return true;
    }
    return false;
}

[AllowAnonymous]
public ActionResult RegisterConfirmation(string Id)
{
    if (ConfirmAccount(Id))
    {
        return RedirectToAction("ConfirmationSuccess");
    }
    return RedirectToAction("ConfirmationFailure");
}
```

متد *ConfirmAccount* سعی می‌کند کاربری را در دیتابیس پیدا کند که شناسه تاییدش با مقدار دریافت شده از URL برابر است. اگر این کاربر پیدا شود، مقدار خاصیت IsConfirmed را به true تغییر می‌دهیم و همین مقدار را به تابع باز می‌گردانیم. در غیر اینصورت false بر می‌گردانیم. اگر کاربر تایید شده است، می‌تواند به سایت وارد شود. برای اینکه مطمئن شویم کاربران پیش از تایید ایمیل شان نمی‌توانند وارد سایت شوند، باید اکشن متد *Login* را کمی تغییر دهیم.

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
{
    if (ModelState.IsValid)
    {
        var user = await UserManager.FindAsync(model.UserName, model.Password);
        if (user != null && user.IsConfirmed)
        {
            await SignInAsync(user, model.RememberMe);
            return RedirectToLocal(returnUrl);
        }
        else
        {
            ModelState.AddModelError("", "Invalid username or password.");
        }
    }

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

تنها کاری که می‌کنیم این است که به دنبال کاربری می‌گردیم که فیلد IsConfirmed آن true باشد. اگر مقدار این فیلد false باشد کاربر را به سایت وارد نمی‌کنیم و پیغام خطایی نمایش می‌دهیم.
همین. این تمام چیزی بود که برای اضافه کردن تصدیق ایمیل به اپلیکیشن خود نیاز دارید. از آنجا که سیستم ASP.NET Identity با Entity Framework مدیریت می‌شود و با مدل Code First ساخته شده، سفارشی کردن اطلاعات کاربران و سیستم عضویت ساده‌تر از همیشه است.

توضیحاتی درباره کار با Postal

اگر به متد SendEmailConfirmation دقت کنید خواهید دید که آبجکتی از نوع Email می‌سازیم (که در اسمبلی‌های Postal وجود دارد) و از آن برای ارسال ایمیل استفاده می‌کنیم. عبارت "RegEmail" نام نمایی است که باید برای ساخت ایمیل استفاده شود. این متغیر از نوع dynamic است، مانند خاصیت ViewBag. بدین معنا که می‌توانید مقادیر مورد نظر خود را بصورت خواص دینامیک روی این آبجکت تعریف کنید. از آنجا که Postal از موتور Razor استفاده می‌کند، بعدا در View ایمیل خود می‌توانید به این مقادیر دسترسی داشته باشید.
در پوشه Views پوشه جدیدی بنام Emails بسازید. سپس یک فایل جدید با نام RegEmail.cshtml در آن ایجاد کنید. کد این فایل را با لیست زیر جایگزین کنید.

```
To: @ViewBag.To
From: YOURNAME@gmail.com
Subject: Confirm your registration

Hello @ViewBag.UserName,
Please confirm your registration by following the link bellow.

@Html.ActionLink(Url.Action("RegisterConfirmation", "Account", new { id = @ViewBag.ConfirmationToken
}), "RegisterConfirmation", "Account", new { id = @ViewBag.ConfirmationToken }, null)
```

این فایل، قالب ایمیل‌های شما خواهد بود. ایمیل‌ها در حال حاظر بصورت plain text ارسال می‌شوند. برای اطلاعات بیشتر درباره ایمیل‌های HTML و امکانات پیشرفته‌تر به سایت پروژه Postal مراجعه کنید.
همانطور که مشاهده می‌کنید در این نما همان خاصیت‌های دینامیک تعریف شده را فراخوانی می‌کنیم تا مقادیر لازم را بدست آوریم.

To. ViewBag آدرس ایمیل گیرنده را نشان می‌دهد.
UserName. ViewBag نام کاربر جاری را نمایش می‌دهد.
ConfirmationToken. ViewBag شناسه تولید شده برای تایید کاربر است.

در این قالب لینکی به متد RegisterConfirmation در کنترلر Account وجود دارد که شناسه تایید را نیز با پارامتری بنام id انتقال می‌دهد.
یک فایل _ViewStart.cshtml هم در این پوشه بسازید و کد آن را با لیست زیر جایگزین کنید.

```
@{ Layout = null; /* Overrides the Layout set for regular page views. */ }
```

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۱۹ ۲۲:۱۴

با تشکر از شما. لطفا View ایمیل ارسالی را (متن حاوی لینک) که توسط کتابخانه Postal پردازش می‌شود، نیز ارسال نمائید. چون الان به نظر متد SendEmailConfirmation مشخص نیست چه متنی را ارسال می‌کند و چطور آن متن را دریافت می‌کند.

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۹ ۲۳:۷

با تشکر از شما، پست بروز رسانی شد.

نویسنده: کاربر
تاریخ: ۱۳۹۲/۱۱/۰۵ ۱۶:۵۲

با تشکر لطفا تنظیمات smtp مربوطه رو هم قرار بدید
سایت سازنده تنظیمات رو در وب کانفیگ قرار داده بود، برای جیمیل من تنظیمات زیر رو مینویسم ولی خطای timed out میده

```
<system.net>
<mailSettings>
<smtp >
  <network host="smtp.gmail.com" userName="My Gmail Email"
  password="my Password" enableSsl="true" port="465" defaultCredentials="true"/>
</smtp>
</mailSettings>
  </system.net>
```

لطفا راهنمایی بفرمایید با تشکر

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۱/۰۵ ۱۹:۵۲

```
<system.net>
  <mailSettings>
    <smtp deliveryMethod="Network" from="armin.zia@gmail.com">
      <network host="smtp.gmail.com" port="587" defaultCredentials="false" enableSsl="true"
userName="YOUR-EMAIL" password="YOUR-PASSWORD" />
    </smtp>
  </mailSettings>
</system.net>
```

نویسنده: داود
تاریخ: ۱۳۹۲/۱۲/۰۵ ۰:۲۷

اگر بخوایم کاربر در فرم لاگین هم با username و هم با email که تأیید شده وارد بشه باید چه کار کنیم؟

نویسنده: Mr.J
تاریخ: ۱۳۹۳/۰۲/۰۱ ۱۵:۱۳

سلام
من دقیقا طبق دستورات بالا کدهام رو نوشتم اما این خطارو میگیره...

```
The SMTP host was not specified.
```

و در قسمت web.config اصلی سایت هم این کدهارو اضافه کردم

```
<system.net>
<mailSettings>
<smtp from="my_gmail">
  <network host="smtp.gmail.com" port="587" defaultCredentials="false" enableSsl="true"
userName="my_gmail" password="mypassword" />
</smtp>
</mailSettings>
  </system.net>
```

مشکل از کجاست.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۰۱ ۱۷:۱۲

نباید مشکلی باشد. مگر اینکه محل قرارگیری تنظیمات system.net شما توسط برنامه قابل یافت شدن نباشد. مثلا آن‌را داخل system.web قرار داده باشید یا مکان دیگری. system.net یک مدخل مجزا و مستقل است. همچنین اگر سایت چندین وب کانفیگ دارد (مانند برنامه‌های ASP.NET MVC)، وب کانفیگ موجود در ریشه سایت باید تنظیم شود و نه مورد موجود در پوشه‌ی Views برنامه.

عنوان: Attribute Routing در ASP.NET MVC 5
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۲۵ ۲:۴۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, ASP.Net MVC, ASP.NET MVC 5

Routing مکانیزم مسیریابی ASP.NET MVC است، که یک URI را به یک اکشن متد نگاشت می‌کند. MVC 5 نوع جدیدی از مسیر یابی را پشتیبانی میکند که Attribute Routing یا مسیریابی نشانه ای نام دارد. همانطور که از نامش پیداست، مسیریابی نشانه ای از Attributeها برای این امر استفاده میکند. این روش به شما کنترل بیشتری روی URIهای اپلیکیشن تان می‌دهد.
مدل قبلی مسیریابی (conventional-routing) هنوز کاملا پشتیبانی می‌شود. در واقع می‌توانید هر دو تکنیک را بعنوان مکمل یکدیگر در یک پروژه استفاده کنید.
در این پست قابلیت‌ها و گزینه‌های اساسی مسیریابی نشانه ای را بررسی میکنیم.
چرا مسیریابی نشانه ای؟
فعال سازی مسیریابی نشانه ای
پارامترهای اختیاری URI و مقادیر پیش فرض
پیشوند مسیر ها
مسیر پیش فرض
محدودیت‌های مسیر ها محدودیت‌های سفارشی

نام مسیر ها
ناحیه‌ها (Areas)

چرا مسیریابی نشانه ای

برای مثال یک وب سایت تجارت آنلاین بهینه شده اجتماعی، می‌تواند مسیرهایی مانند لیست زیر داشته باشد:

{productId:int}/{productTitle}

نگاشت می‌شود به: ProductsController.Show(int id)

{username}

نگاشت می‌شود به: ProfilesController.Show(string username)

{username}/catalogs/{catalogId:int}/{catalogTitle}

نگاشت می‌شود به: CatalogsController.Show(string username, int catalogId)

در نسخه قبلی ASP.NET MVC، قوانین مسیریابی در فایل RouteConfig.cs تعریف می‌شدند، و اشاره به اکشن‌های کنترلرها به نحو زیر انجام می‌شد:

```
routes.MapRoute(
    name: "ProductPage",
    url: "{productId}/{productTitle}",
    defaults: new { controller = "Products", action = "Show" },
    constraints: new { productId = "\\d+" }
);
```

هنگامی که قوانین مسیریابی در کنار اکشن متدها تعریف می‌شوند، یعنی در یک فایل سورس و نه در یک کلاس پیکربندی خارجی، درک و فهم نگاشت URIها به اکشن‌ها واضح‌تر و راحت می‌شود. تعریف مسیر قبلی، می‌تواند توسط یک attribute ساده بدین صورت نگاشت شود:

```
[Route("{productId:int}/{productTitle}")]
public ActionResult Show(int productId) { ... }
```

فعال سازی Attribute Routing

برای فعال سازی مسیریابی نشانه ای، متد MapMvcAttributeRoutes را هنگام پیکربندی فراخوانی کنید.

```
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapMvcAttributeRoutes();
    }
}
```

همچنین می‌توانید مدل قبلی مسیریابی را با تکنیک جدید تلفیق کنید.

```
public static void RegisterRoutes(RouteCollection routes)
{
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

    routes.MapMvcAttributeRoutes();

    routes.MapRoute(
        name: "Default",
        url: "{controller}/{action}/{id}",
        defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
    );
}
```

پارامترهای اختیاری URI و مقادیر پیش فرض

می توانید با اضافه کردن یک علامت سوال به پارامترهای مسیریابی، آنها را optional یا اختیاری کنید. برای تعیین مقدار پیش فرض هم از فرمت parameter=value استفاده می‌کنید.

```
public class BooksController : Controller
{
    // eg: /books
    // eg: /books/1430210079
    [Route("books/{isbn?}")]
    public ActionResult View(string isbn)
    {
        if (!String.IsNullOrEmpty(isbn))
        {
            return View("OneBook", GetBook(isbn));
        }
        return View("AllBooks", GetBooks());
    }

    // eg: /books/lang
    // eg: /books/lang/en
    // eg: /books/lang/he
    [Route("books/lang/{lang=en}")]
    public ActionResult ViewByLanguage(string lang)
    {
        return View("OneBook", GetBooksByLanguage(lang));
    }
}
```

در این مثال، هر دو مسیر books/ و books/1430210079/ به اکشن متد "View" نگاشت می‌شوند، مسیر اول تمام کتاب‌ها را لیست میکند، و مسیر دوم جزئیات کتابی مشخص را لیست می‌کند. هر دو مسیر books/lang/ و books/lang/en/ به یک شکل نگاشت می‌شوند، چرا که مقدار پیش فرض این پارامتر en تعریف شده.

پیشوند مسیرها (Route Prefixes)

برخی اوقات، تمام مسیرها در یک کنترلر با یک پیشوند شروع می‌شوند. بعنوان مثال:

```
public class ReviewsController : Controller
{
    // eg: /reviews
```

```
    [Route("reviews")]
    public ActionResult Index() { ... }
    // eg: /reviews/5
    [Route("reviews/{reviewId}")]
    public ActionResult Show(int reviewId) { ... }
    // eg: /reviews/5/edit
    [Route("reviews/{reviewId}/edit")]
    public ActionResult Edit(int reviewId) { ... }
}
```

همچنین می‌توانید با استفاده از خاصیت [RoutePrefix] یک پیشوند عمومی برای کل کنترلر تعریف کنید:

```
[RoutePrefix("reviews")]
public class ReviewsController : Controller
{
    // eg.: /reviews
    [Route]
    public ActionResult Index() { ... }
    // eg.: /reviews/5
    [Route("{reviewId}")]
    public ActionResult Show(int reviewId) { ... }
    // eg.: /reviews/5/edit
    [Route("{reviewId}/edit")]
    public ActionResult Edit(int reviewId) { ... }
}
```

در صورت لزوم، می‌توانید برای بازنویسی (override) پیشوند مسیرها از کاراکتر ~ استفاده کنید:

```
[RoutePrefix("reviews")]
public class ReviewsController : Controller
{
    // eg.: /spotlight-review
    [Route("~/spotlight-review")]
    public ActionResult ShowSpotlight() { ... }

    ...
}
```

مسیر پیش فرض

می توانید خاصیت [Route] را روی کنترلر اعمال کنید، تا اکشن متد را بعنوان یک پارامتر بگیرید. این مسیر سپس روی تمام اکشن متدهای این کنترلر اعمال می‌شود، مگر آنکه یک [Route] بخصوص روی اکشن‌ها تعریف شده باشد.

```
[RoutePrefix("promotions")]
[Route("{action=index}")]
public class ReviewsController : Controller
{
    // eg.: /promotions
    public ActionResult Index() { ... }

    // eg.: /promotions/archive
    public ActionResult Archive() { ... }

    // eg.: /promotions/new
    public ActionResult New() { ... }

    // eg.: /promotions/edit/5
    [Route("edit/{promoId:int}")]
    public ActionResult Edit(int promoId) { ... }
}
```

محدودیت‌های مسیر ها

با استفاده از Route Constraints می‌توانید نحوه جفت شدن پارامترها در قالب مسیریابی را محدود و کنترل کنید. فرمت کلی {parameter:constraint} است. بعنوان مثال:

```
// eg: /users/5
[Route("users/{id:int}"]
public ActionResult GetUserById(int id) { ... }

// eg: users/ken
[Route("users/{name}"]
public ActionResult GetUserByName(string name) { ... }
```

در اینجا، مسیر اول تنها در صورتی انتخاب می‌شود که قسمت id در URI یک مقدار integer باشد. در غیر اینصورت مسیر دوم انتخاب خواهد شد.

جدول زیر constraintها یا محدودیت هایی که پشتیبانی می‌شوند را لیست می‌کند.

مثال	توضیحات	محدودیت
{x:alpha}	کاراکترهای الفبای لاتین را تطبیق (match) می‌دهد (a-z, A-Z).	alpha
{x:bool}	یک مقدار منطقی را تطبیق می‌دهد.	bool
{x:datetime}	یک مقدار **DateTime** را تطبیق می‌دهد.	datetime
{x:decimal}	یک مقدار پولی را تطبیق می‌دهد.	decimal
{x:double}	یک مقدار اعشاری 64 بیتی را تطبیق می‌دهد.	double
{x:float}	یک مقدار اعشاری 32 بیتی را تطبیق می‌دهد.	float
{x:guid}	یک مقدار GUID را تطبیق می‌دهد.	guid
{x:int}	یک مقدار 32 بیتی integer را تطبیق می‌دهد.	int
{x:length(6)} {x:length(1,20)}	رشته ای با طول تعیین شده را تطبیق می‌دهد.	length
{x:long}	یک مقدار 64 بیتی integer را تطبیق می‌دهد.	long
{(x:max(10}	یک مقدار integer با حداکثر مجاز را تطبیق می‌دهد.	max
{(x:maxlength(10}	رشته ای با حداکثر طول تعیین شده را تطبیق می‌دهد.	maxlength
{(x:min(10}	مقداری integer با حداقل مقدار تعیین شده را تطبیق می‌دهد.	min
{(x:minlength(10}	رشته ای با حداقل طول تعیین شده را تطبیق می‌دهد.	minlength
{x:range(10,50)}	مقداری integer در بازه تعریف شده را تطبیق می‌دهد.	range
{($\{x:regex(^\d{3}-\d{3}-\d{4}	یک عبارت با قاعده را تطبیق می‌دهد.	regex

توجه کنید که بعضی از constraint ها، مانند "min" آرگومان‌ها را در پرانتز دریافت می‌کنند.
می توانید محدودیت‌های متعددی روی یک پارامتر تعریف کنید، که باید با دونقطه جدا شوند. بعنوان مثال:

```
// eg: /users/5
// but not /users/10000000000 because it is larger than int.MaxValue,
// and not /users/0 because of the min(1) constraint.
[Route("users/{id:int:min(1)}")]
public ActionResult GetUserById(int id) { ... }
```

مشخص کردن اختیاری بودن پارامتر ها، باید در آخر لیست constraints تعریف شود:

```
// eg: /greetings/bye
// and /greetings because of the Optional modifier,
// but not /greetings/see-you-tomorrow because of the maxlength(3) constraint.
[Route("greetings/{message:maxlength(3)?}")]
public ActionResult Greet(string message) { ... }
```

محدودیت‌های سفارشی

با پیاده سازی قرارداد IRouteConstraint می‌توانید محدودیت‌های سفارشی بسازید. بعنوان مثال، constraint زیر یک پارامتر را به لیستی از مقادیر قابل قبول محدود می‌کند:

```
public class ValuesConstraint : IRouteConstraint
{
    private readonly string[] validOptions;
    public ValuesConstraint(string options)
    {
        validOptions = options.Split('|');
    }

    public bool Match(HttpContextBase httpContext, Route route, string parameterName,
RouteValueDictionary values, RouteDirection routeDirection)
    {
        object value;
        if (values.TryGetValue(parameterName, out value) && value != null)
        {
            return validOptions.Contains(value.ToString(), StringComparer.OrdinalIgnoreCase);
        }
        return false;
    }
}
```

قطعه کد زیر نحوه رجیستر کردن این constraint را نشان می‌دهد:

```
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        var constraintsResolver = new DefaultInlineConstraintResolver();

        constraintsResolver.ConstraintMap.Add("values", typeof(ValuesConstraint));

        routes.MapMvcAttributeRoutes(constraintsResolver);
    }
}
```

حالا می‌توانید این محدودیت سفارشی را روی مسیرها اعمال کنید:

```
public class TemperatureController : Controller
{
    // eg: temp/celsius and /temp/fahrenheit but not /temp/kelvin
    [Route("temp/{scale:values(celsius|fahrenheit)}")]
    public ActionResult Show(string scale)
    {
```

```
        return Content("scale is " + scale);
    }
}
```

نام مسیر ها

می توانید به مسیرها یک نام اختصاص دهید، با این کار تولید URIها هم راحت‌تر می‌شوند. بعنوان مثال برای مسیر زیر:

```
[Route("menu", Name = "mainmenu")]
public ActionResult MainMenu() { ... }
```

می‌توانید لینکی با استفاده از Url.RouteUrl تولید کنید:

```
<a href="@Url.RouteUrl("mainmenu")">Main menu</a>
```

ناحیه‌ها (Areas)

برای مشخص کردن ناحیه ای که کنترلر به آن تعلق دارد می‌توانید از خاصیت [RouteArea] استفاده کنید. هنگام استفاده از این خاصیت، می‌توانید با خیال راحت کلاس AreaRegistration را از ناحیه مورد نظر حذف کنید.

```
[RouteArea("Admin")]
[RoutePrefix("menu")]
[Route("{action}")]
public class MenuController : Controller
{
    // eg: /admin/menu/login
    public ActionResult Login() { ... }

    // eg: /admin/menu/show-options
    [Route("show-options")]
    public ActionResult Options() { ... }

    // eg: /stats
    [Route("~/stats")]
    public ActionResult Stats() { ... }
}
```

با این کنترلر، فراخوانی تولید لینک زیر، رشته " */Admin/menu/show-options* " را بدست میدهد:

```
Url.Action("Options", "Menu", new { Area = "Admin" })
```

به منظور تعریف یک پیشوند سفارشی برای یک ناحیه، که با نام خود ناحیه مورد نظر متفاوت است می‌توانید از پارامتر AreaPrefix استفاده کنید. بعنوان مثال:

```
[RouteArea("BackOffice", AreaPrefix = "back-office")]
```

اگر از ناحیه‌ها هم بصورت مسیریابی نشانه ای، و هم بصورت متداول (که با کلاس‌های AreaRegistration پیکربندی می‌شوند) استفاده می‌کنید باید مطمئن شوید که رجیستر کردن نواحی اپلیکیشن پس از مسیریابی نشانه ای پیکربندی می‌شود. به هر حال رجیستر کردن ناحیه‌ها پیش از تنظیم مسیرها بصورت متداول باید صورت گیرد. دلیل آن هم مشخص است، برای اینکه درخواست‌های ورودی بدرستی با مسیرهای تعریف شده تطبیق داده شوند، باید ابتدا attribute routes، سپس area registration و در آخر default route رجیستر شوند. بعنوان مثال:

```
public static void RegisterRoutes(RouteCollection routes)
{
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
```

```
    routes.MapMvcAttributeRoutes();

    AreaRegistration.RegisterAllAreas();

    routes.MapRoute(
        name: "Default",
        url: "{controller}/{action}/{id}",
        defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
    );
}
```

نظرات خوانندگان

نویسنده: منصور جعفری
تاریخ: ۱۳۹۲/۱۲/۲۹ ۰:۲۷

سلام
الان من در قسمت route.config به این صورت کدهام تعریف شده

```
public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
            routes.MapMvcAttributeRoutes();
            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "List", id = UrlParameter.Optional }
            );
        }
```

و اومدم توی کنترل Home و اکشن موبوط به اون که دارای یک پارامتر دریافتی برای پیج هست این کد رو تعریف کردم:

```
[Route("Page/{page?}")]
        public ActionResult List(int page=1)
        {
```

اما وقتی برنامه رو اجرا میکنم خطای 404 رو میده. ممنون میشم راهنمایی کنید مشکل از کجاست.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۲۹ ۱:۱۱

برای ریشه سایت ویژگی زیر را هم باید اضافه کنید:

```
[Route("~/")]
```

نویسنده: منصور جعفری
تاریخ: ۱۳۹۲/۱۲/۲۹ ۱۲:۳۳

ممنونم که پاسخ دادید یعنی باید به این شکل بنویسم

```
[Route("~/Page/{page?}")]
        public ActionResult List(int page=1)
        {
```

نیازی به تعریف اکشن یا RoutePrifix نیست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۲۹ ۱۲:۴۵

قابلیت ترکیب دارند:

```
[Route("~/")]
[Route("Page/{page?}")]
public ActionResult List(int page=1)
{
```

نویسنده: یاشار راشدی
تاریخ: ۱۳۹۳/۰۳/۲۹ ۱۳:۲۲

در صورتی که از Areaها استفاده کنید باید بالای هر کنترلر داخل Area حتما Prefix مربوط به Area را اضافه کنید وگرنه Exception دریافت می‌کنید.

```
[RouteArea("Admin")]
[RoutePrefix("menu")]
[Route("{action}")]
public class MenuController : Controller
{
    // eg: /admin/menu/login
    public ActionResult Login() { ... }

    // eg: /admin/menu/show-options
    [Route("show-options")]
    public ActionResult Options() { ... }

    // eg: /stats
    [Route("~/stats")]
    public ActionResult Stats() { ... }
```

توضیحات بیشتر

http://blogs.msdn.com/b/webdev/archive/2013/10/17/attribute-routing-in-asp-net-mvc-5.aspx#route-areas