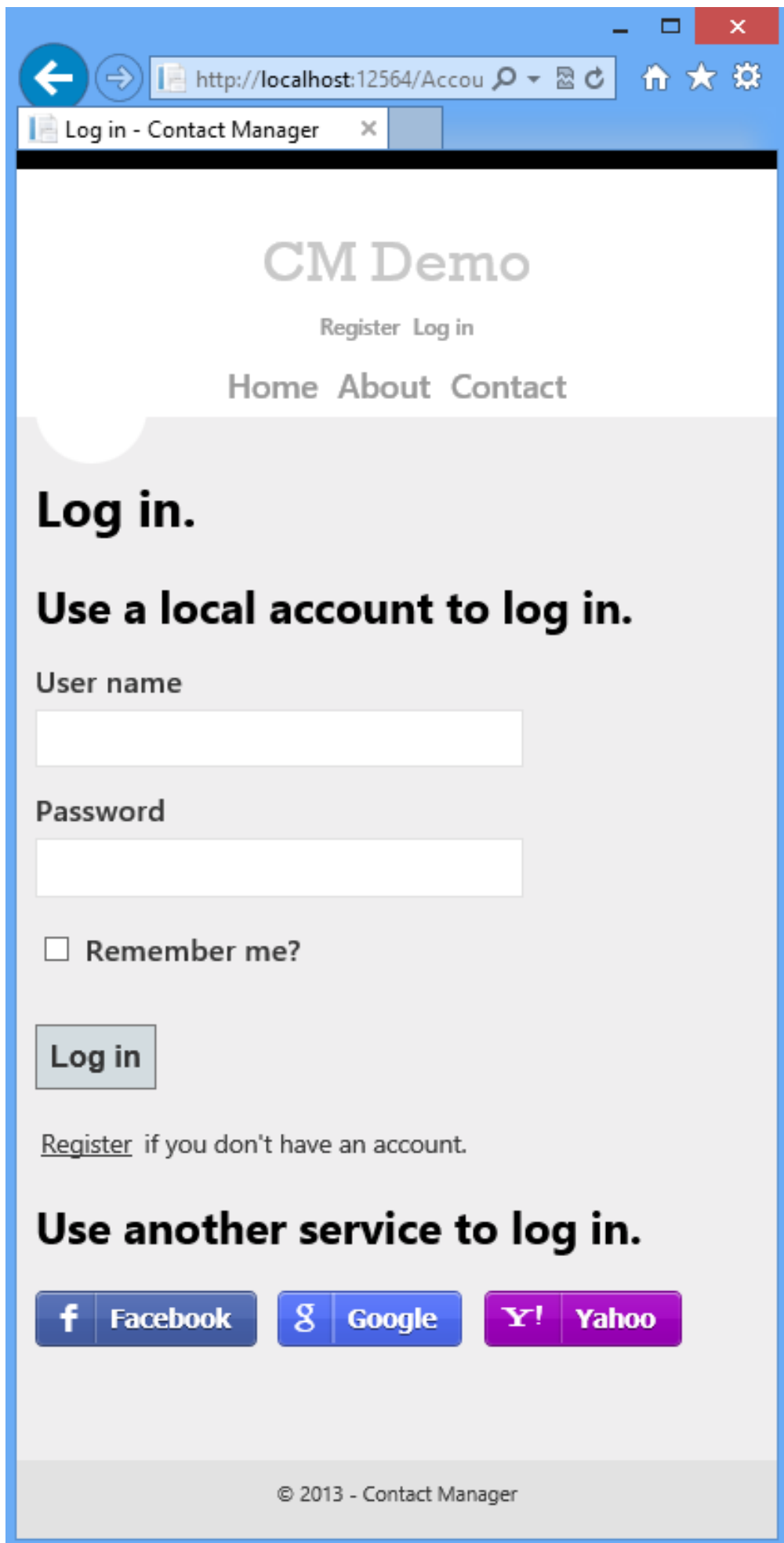


این مقاله به شما نشان می‌دهد چگونه یک اپلیکیشن وب ASP.NET MVC 5 بسازید که کاربران را قادر می‌سازد با اطلاعات Facebook یا Google احراز هویت شده و به سایت وارد شوند. همچنین این اپلیکیشن را روی Windows Azure توزیع (Deploy) خواهید کرد. می‌توانید بصورت رایگان یک حساب کاربری Windows Azure بسازید. اگر هم Visual 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 تجربه‌ای 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.



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.

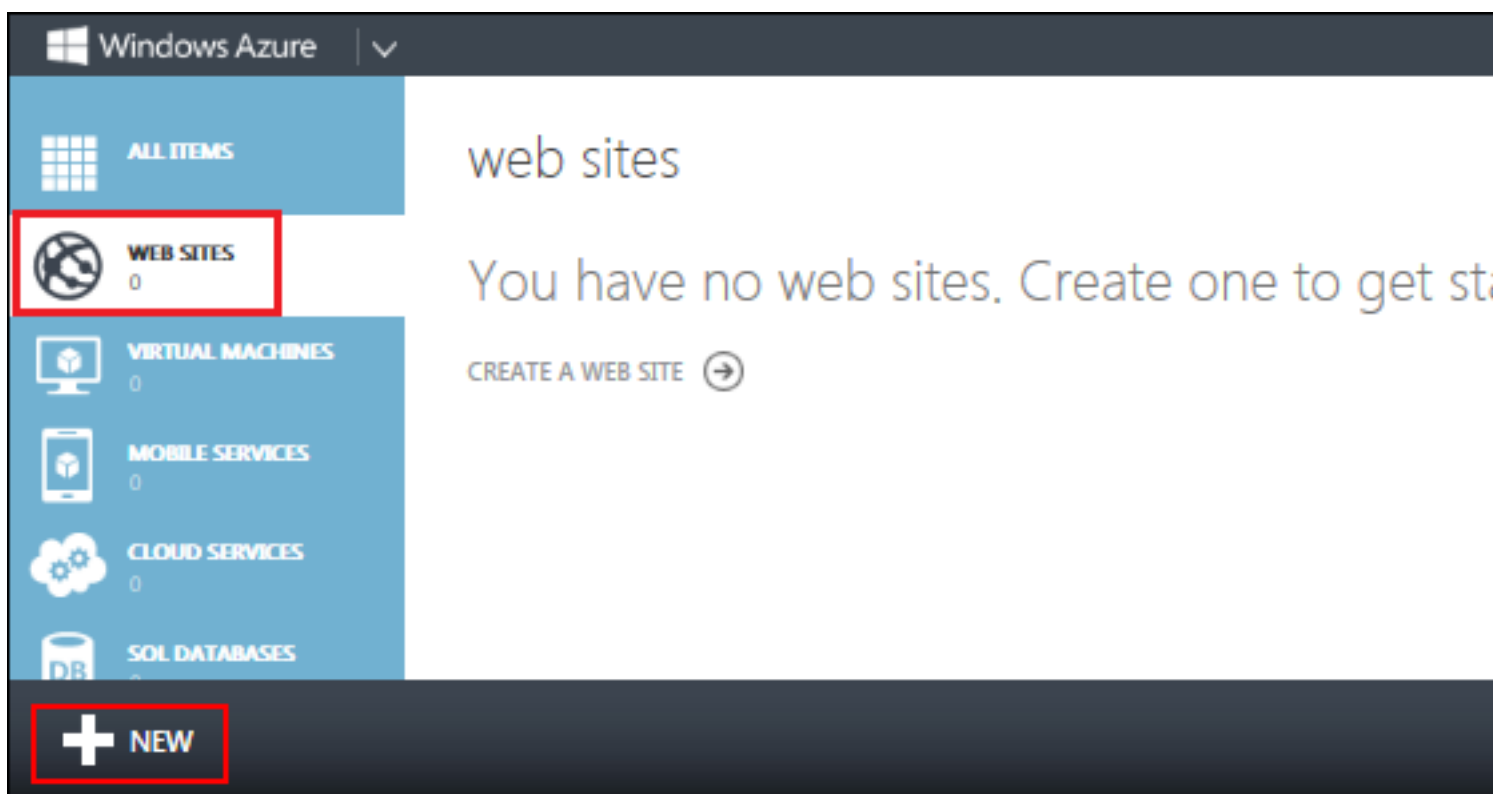
### برپایی محیط Windows Azure

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

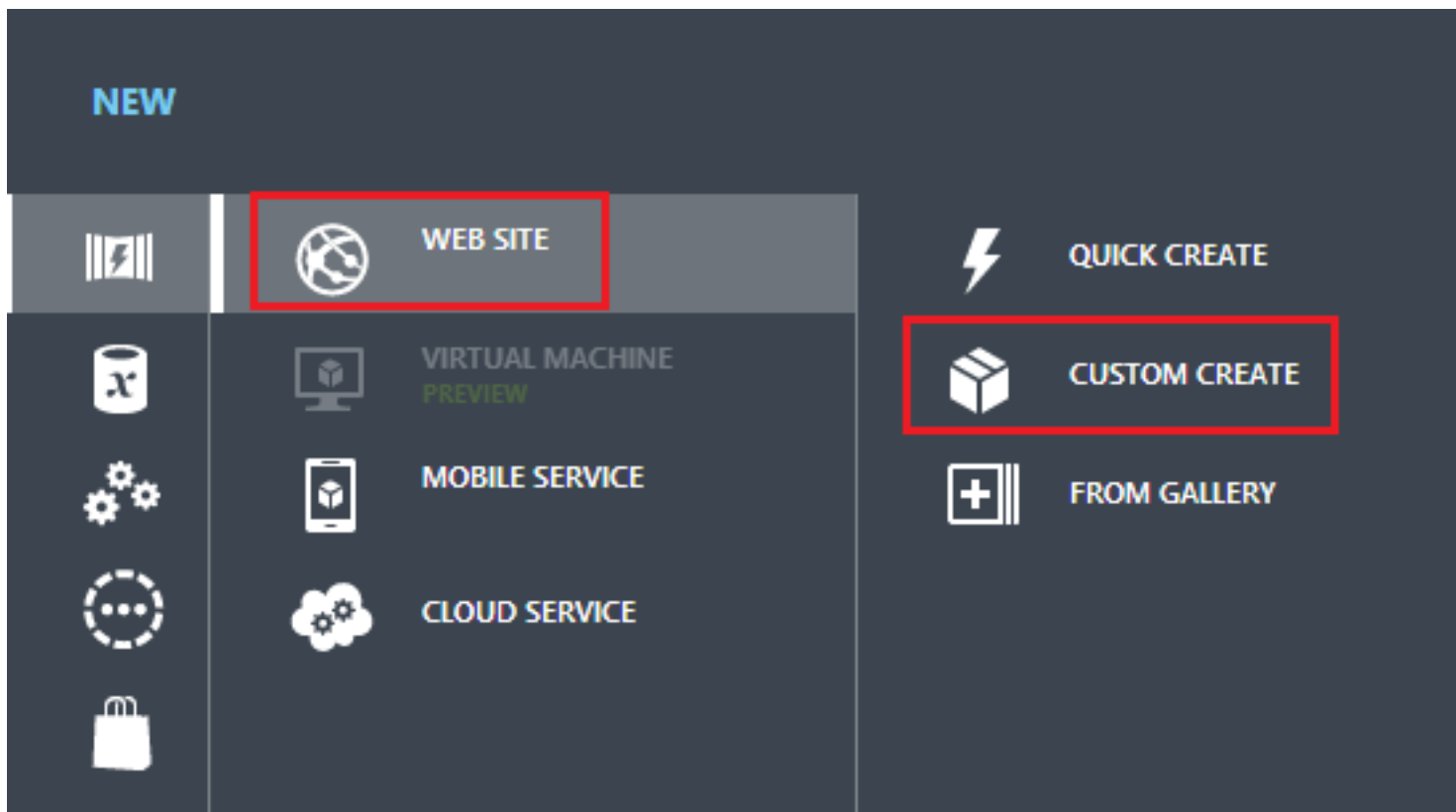
ایجاد یک وب سایت و دیتابیس در Windows Azure

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.

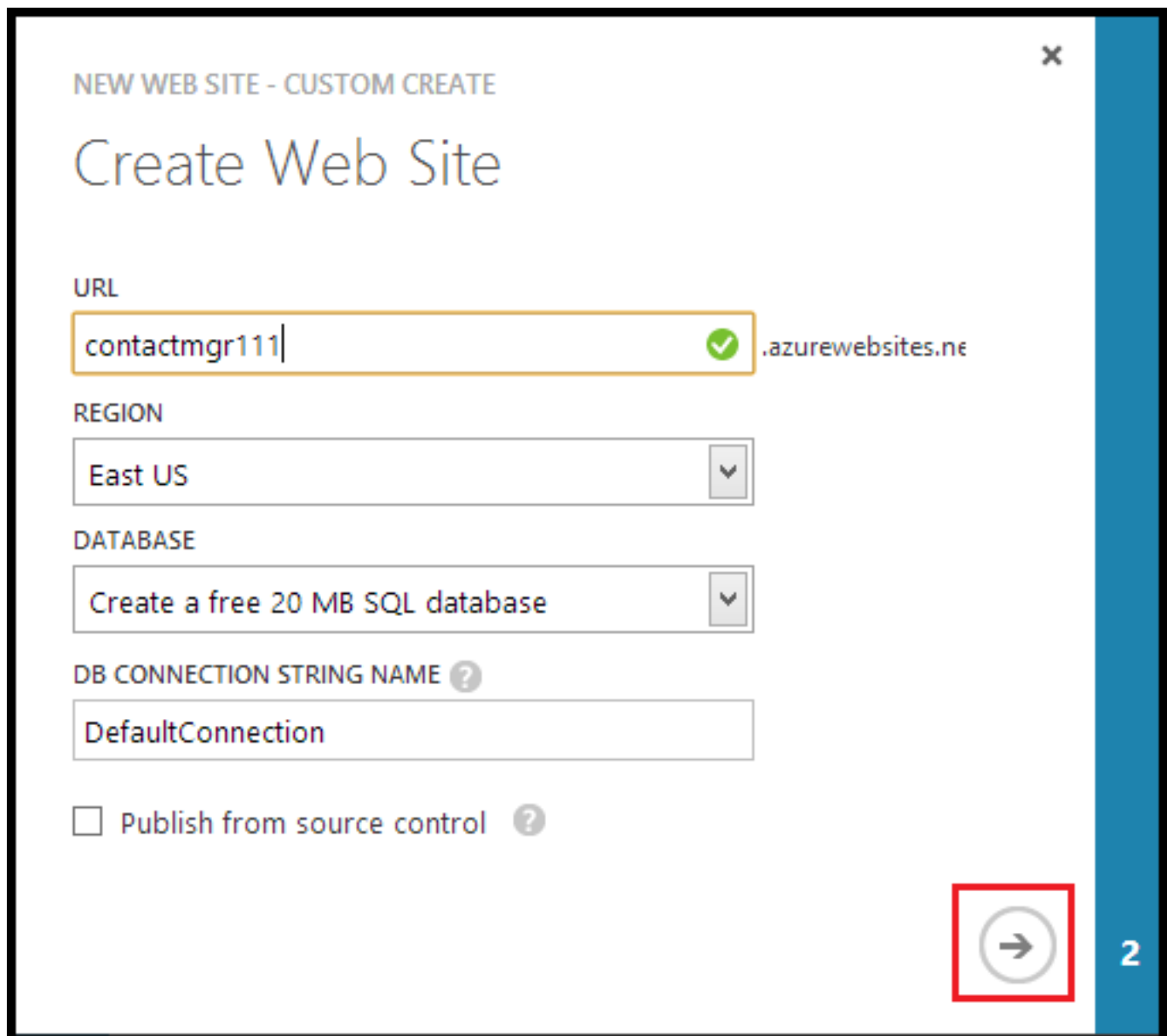
Windows Azure SQL Database یک سرویس دیتابیس رابطه ای (relational) و مبتنی بر Cloud است که بر اساس تکنولوژی‌های SQL Server ساخته شده. ابزار و اپلیکیشن‌هایی که با SQL Server کار می‌کنند با SQL Database نیز می‌توانند کار کنند. در [پرتال مدیریتی Windows Azure](#) روی **Web Sites** در قسمت چپ صفحه کلیک کنید، 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 قبلاً 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 بعداً 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.

NEW WEB SITE - CREATE WITH DATABASE

## Specify database settings

NAME  
contactDB ✓

SERVER  
New SQL Database Server ▼

LOGIN NAME  
contactmanageradmin ?

PASSWORD  
●●●●●●●●

PASSWORD CONFIRMATION  
●●●●●●●●

REGION  
East US ▼

☐ Configure advanced database settings

1

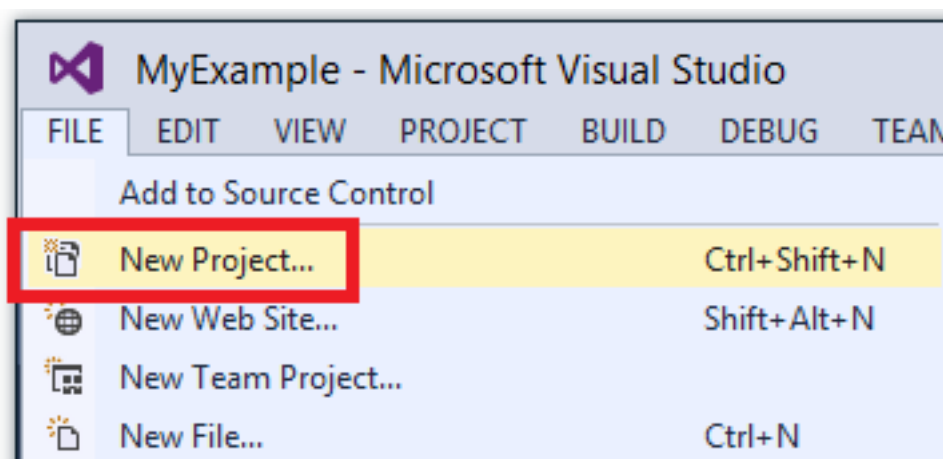
← ✓

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 (معمولاً کمتر از یک دقیقه) این ستون نشان می‌دهد که سایت شما با موفقیت ایجاد شده. در منوی پیمایش سمت چپ، تعداد سایت‌هایی 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.

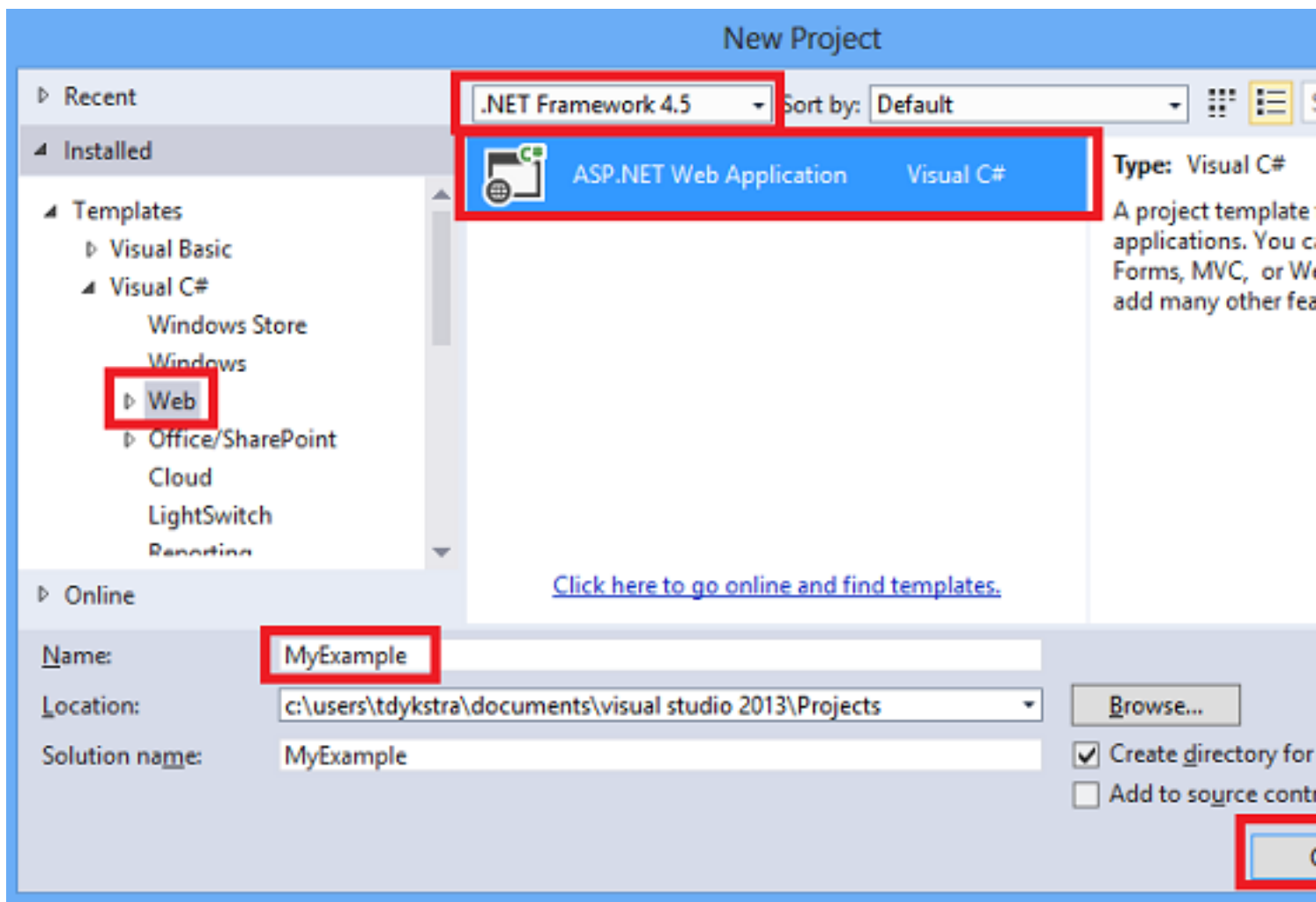
#### یک اپلیکیشن ASP.NET MVC 5 بسازید

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.



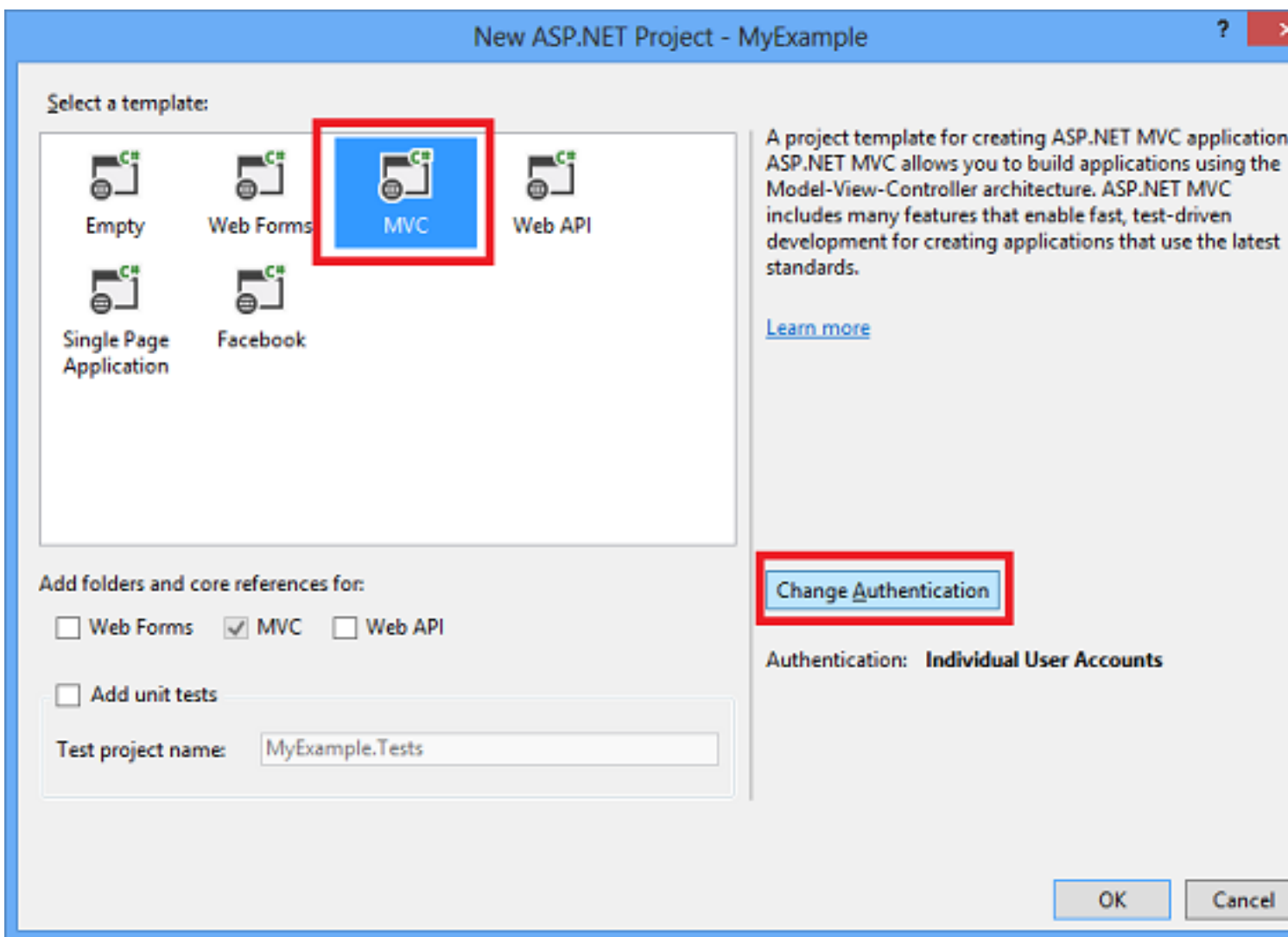


نوع پروژه را **ASP.NET Web Application** انتخاب کنید.



**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 تیترو 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.

```

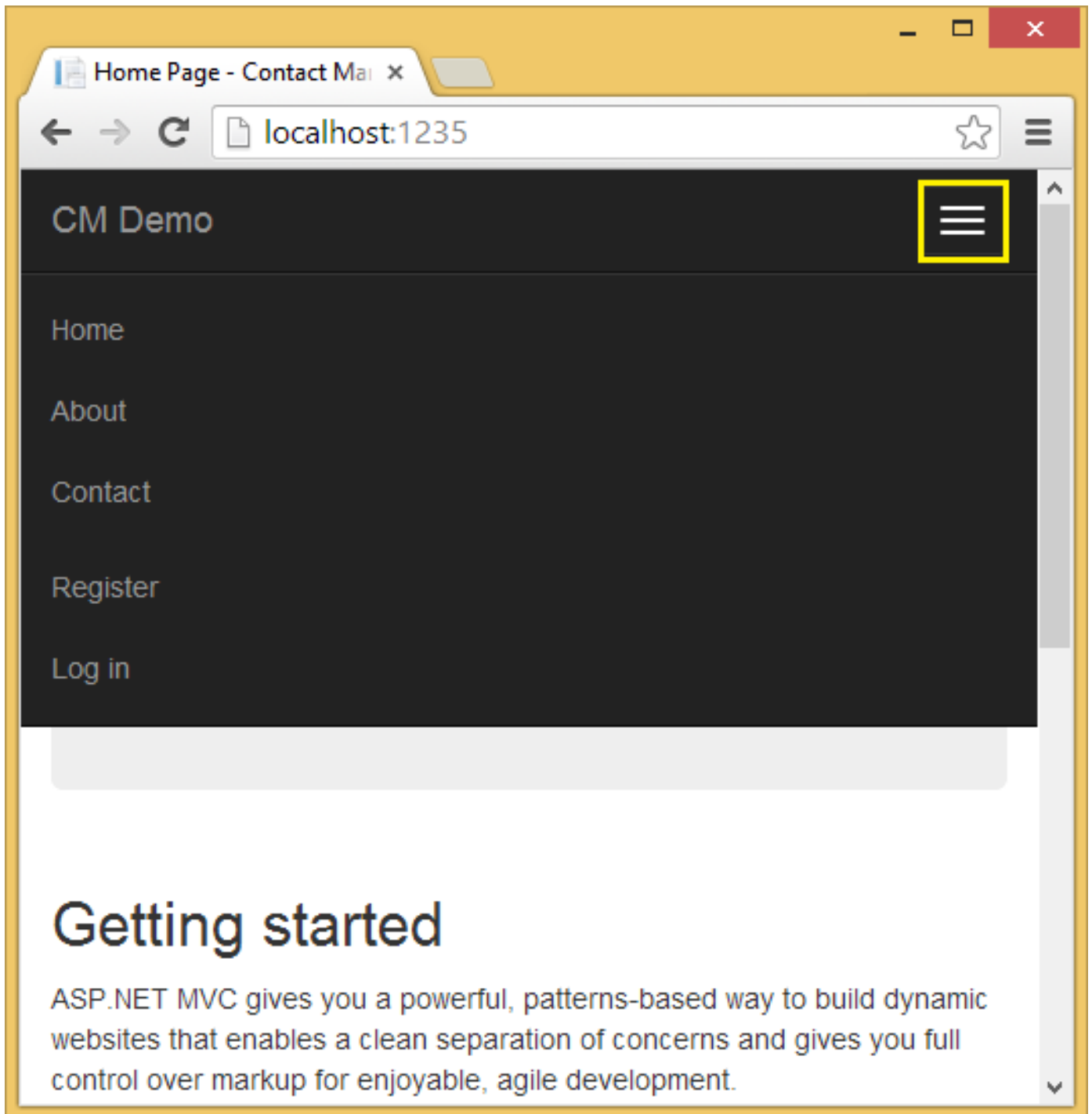
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - Contact Manager</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")

</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>

```

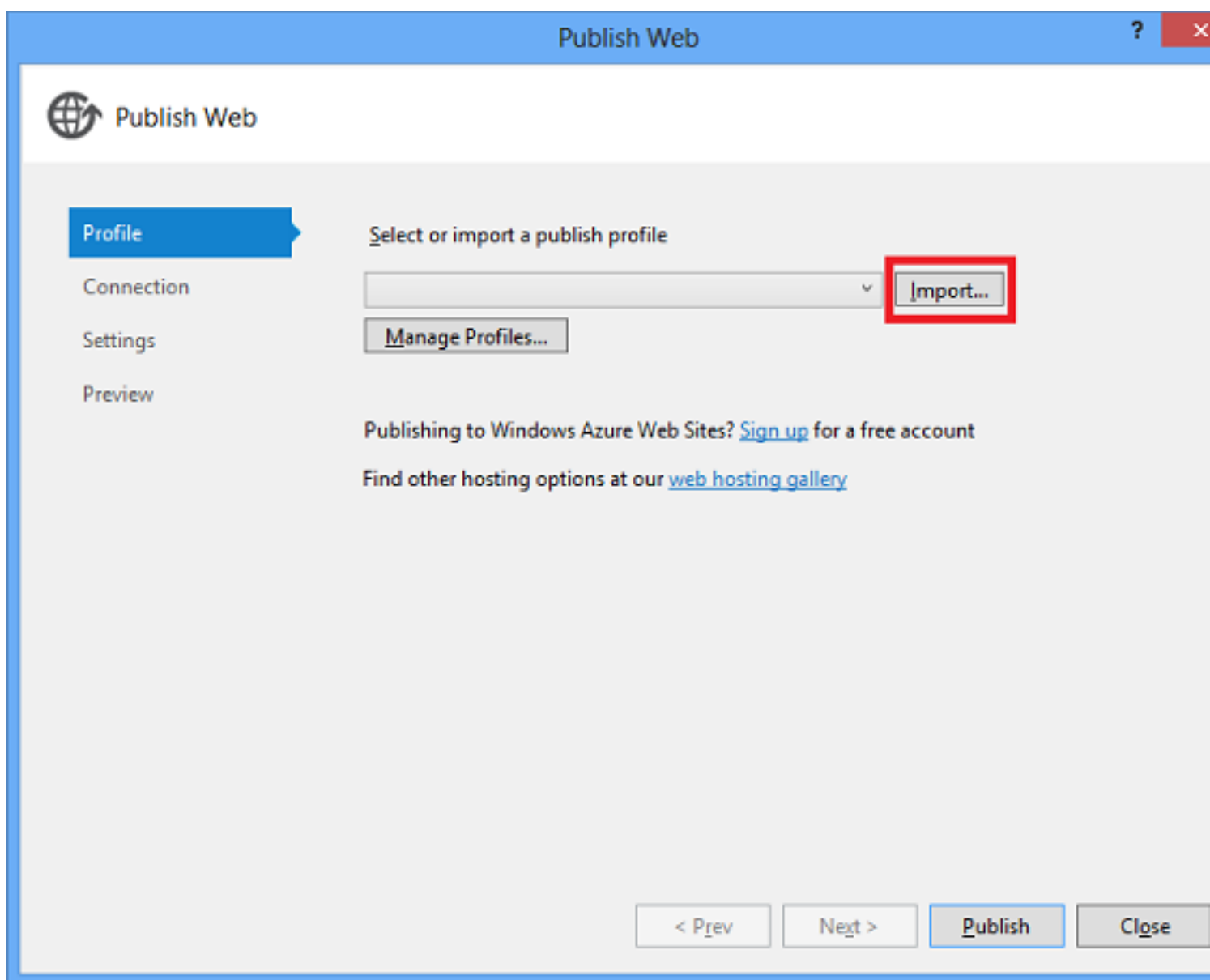
اپلیکیشن را بصورت محلی اجرا کنید  
اپلیکیشن را با **Ctrl + F5** اجرا کنید. صفحه اصلی باید در مرورگر پیش فرض باز شود.



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.

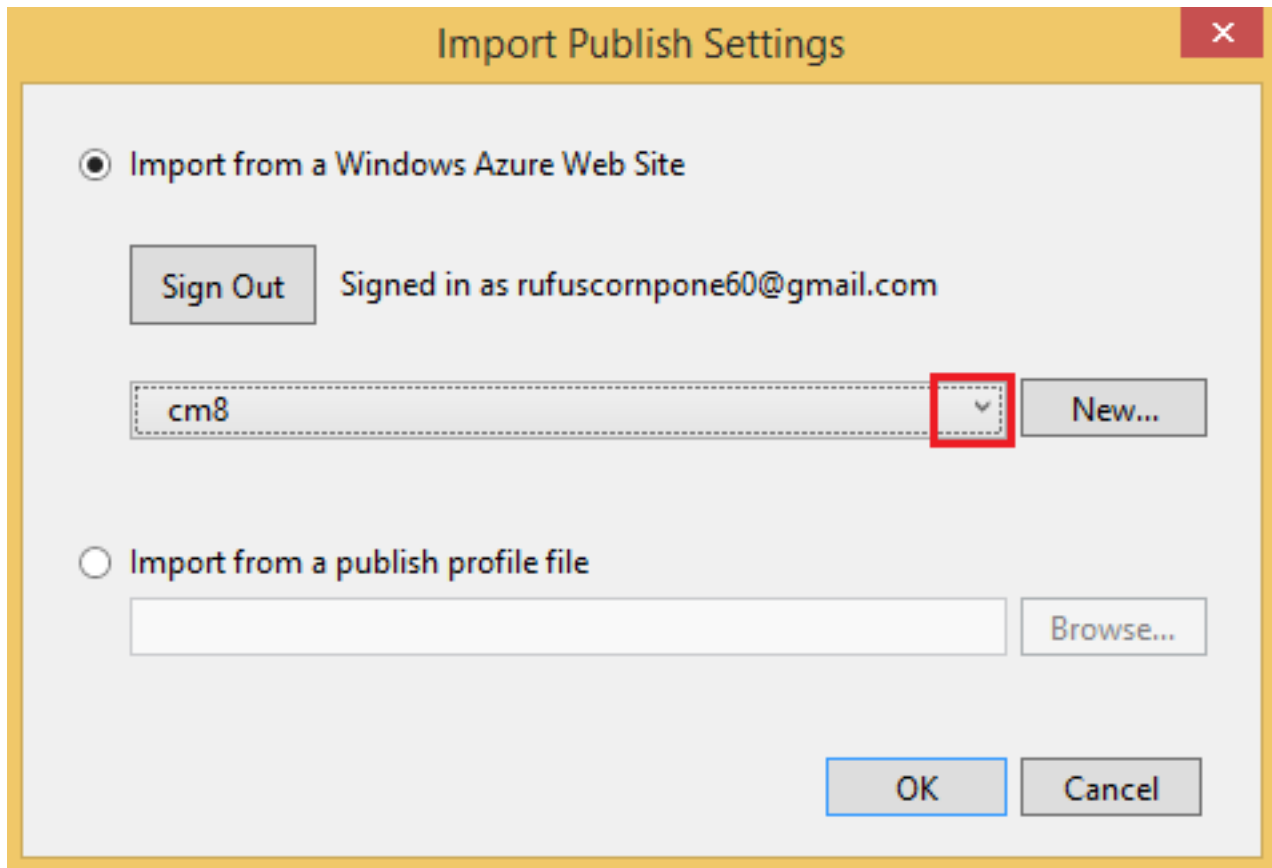
یکی از متدهای زیر را استفاده کنید تا ویژوال استودیو بتواند به اکانت Windows Azure شما متصل شود. روی **Sign In** کلیک کنید تا با وارد کردن اطلاعات حساب کاربری وارد Windows Azure شوید.

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، مثلاً 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 کاملاً 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.



The image shows a dialog box titled "Import Publish Settings" with a yellow header bar and a red close button in the top right corner. The dialog has two main sections. The first section, "Import from a Windows Azure Web Site", is selected with a radio button. It contains a "Sign Out" button, the text "Signed in as rufuscornpone60@gmail.com", a text box with "cm8" and a dropdown arrow (highlighted with a red rectangle), and a "New..." button. The second section, "Import from a publish profile file", is unselected. It contains an empty text box and a "Browse..." button. At the bottom right are "OK" and "Cancel" buttons.

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

**Publish Web**

Profile: **cm1234 \***

Connection (selected)

Settings

Preview

Publish method: Web Deploy

Server: waws-prod-bay-003.publish.azurewebsites.windows.net:443

Site name: cm1234

User name: \$cm1234

Password: [Masked]

☒ Save password

Destination URL: http://cm1234.azurewebsites.net

Validate Connection

< Prev   Next >   **Publish**   Close

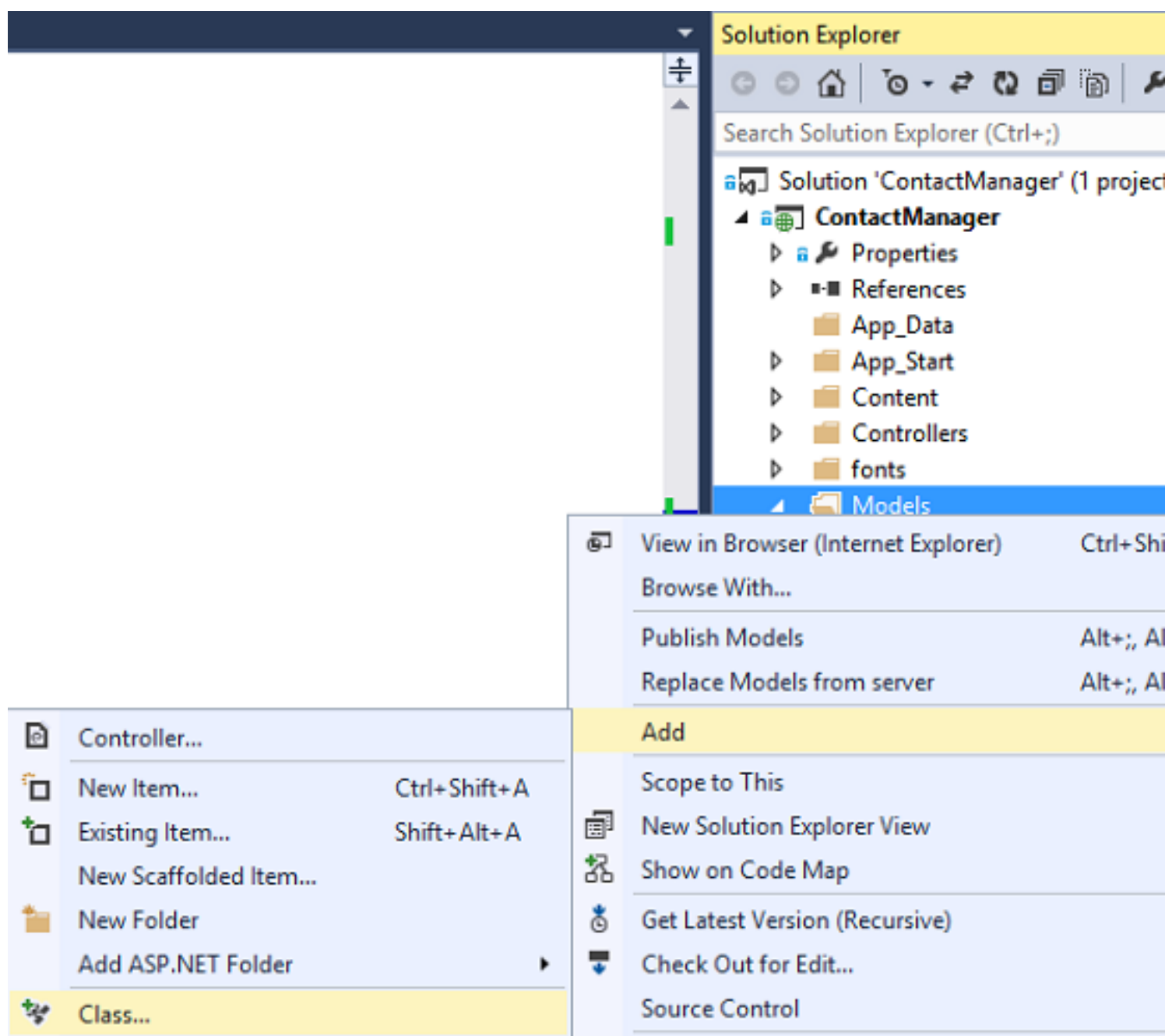
اپلیکیشن شما حالا در فضای ابری اجرا می‌شود. دفعه بعد که اپلیکیشن را منتشر کنید تنها فایل‌های تغییر کرده (یا جدید) آپلود خواهند شد.

#### 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.

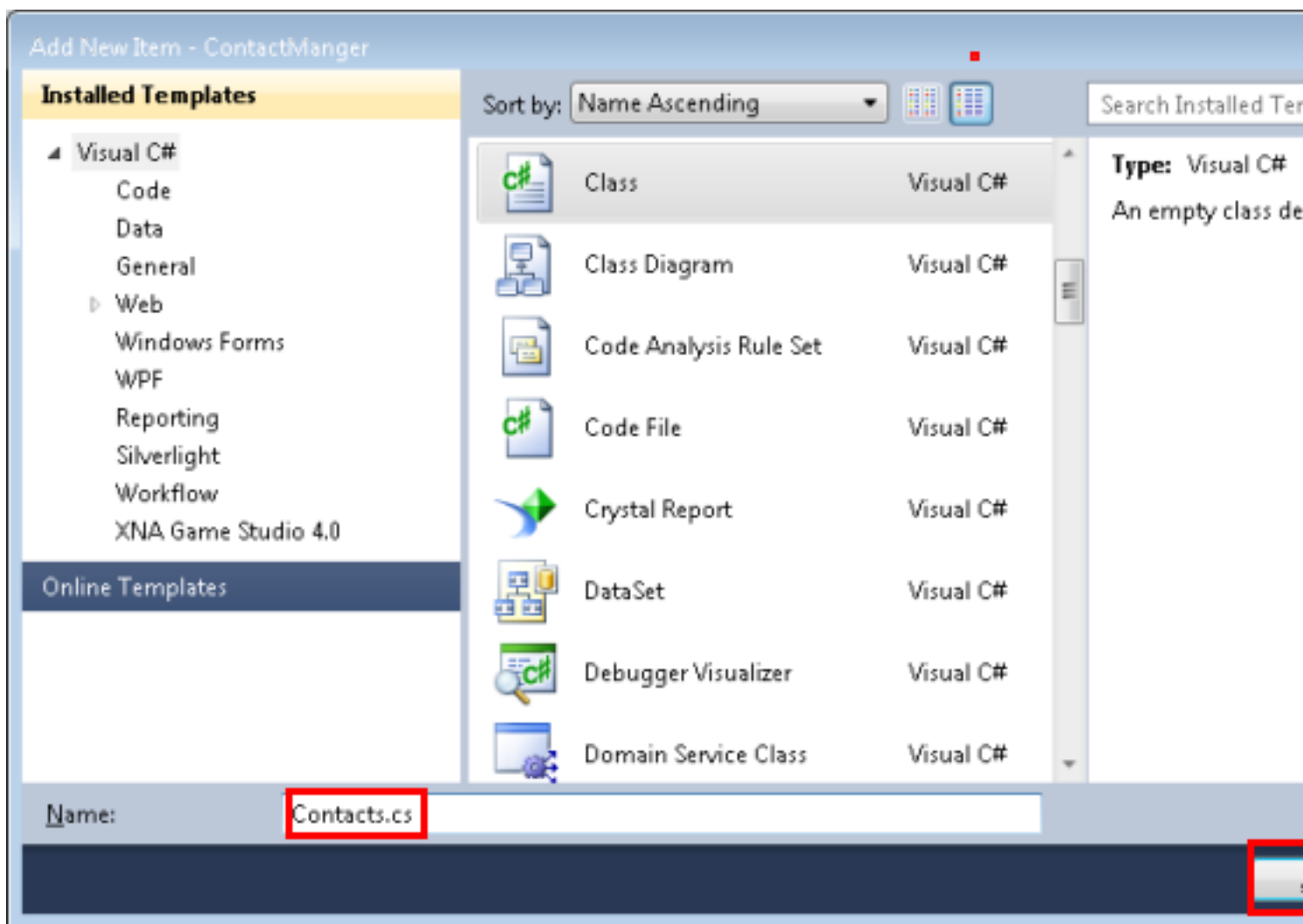
#### کلاس‌های مدل Contacts را اضافه کنید

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; }
    }
}
```

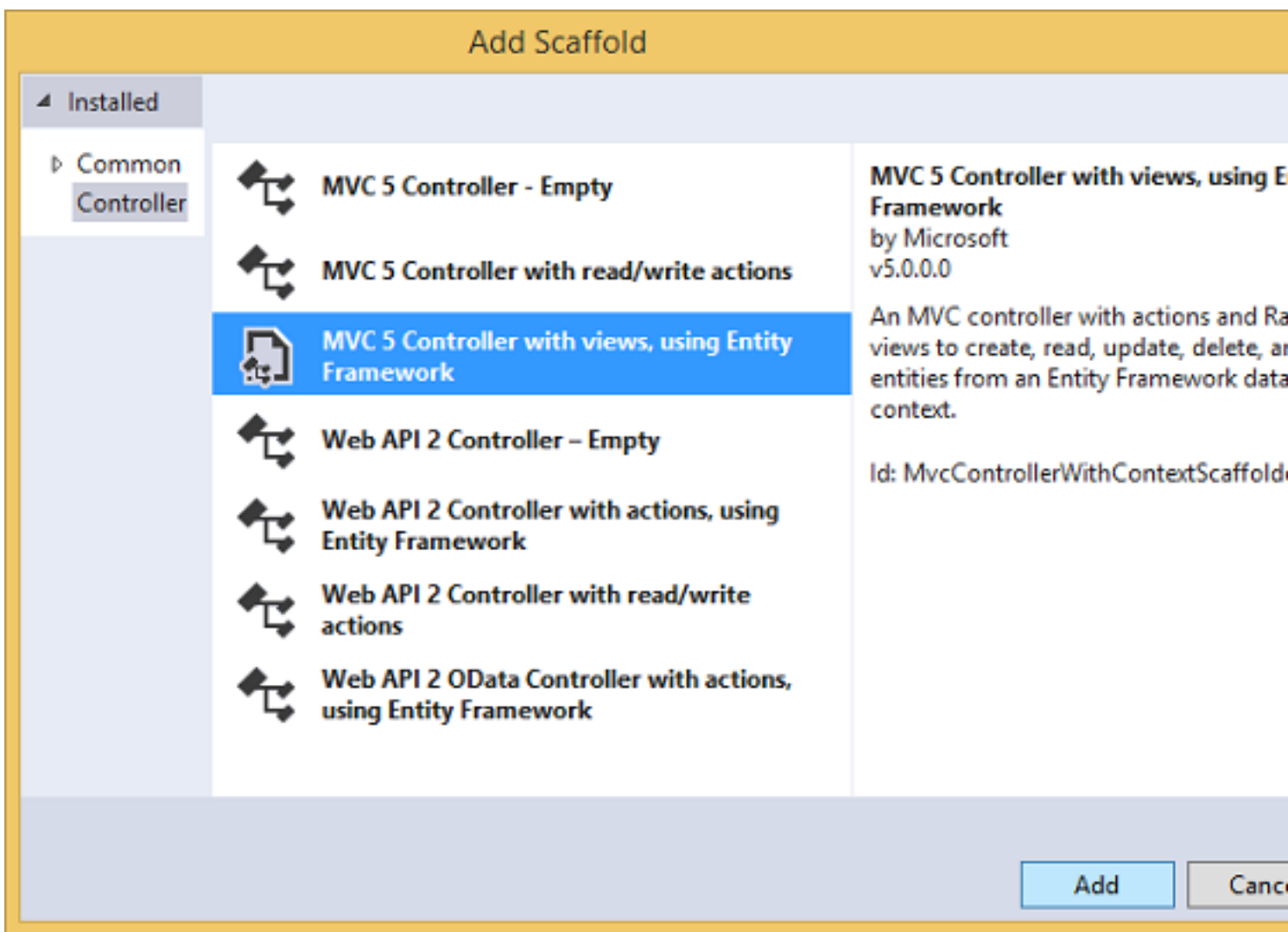
این کلاس موجودیت Contact را در دیتابیس معرفی می‌کند. داده‌هایی که می‌خواهیم برای هر رکورد ذخیره کنیم تعریف شده‌اند، علاوه بر یک فیلد Primary Key که دیتابیس به آن نیاز دارد.

**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.

### Add Controller

Controller name:

☐ Use async controller actions

Model class:

Data context class:

New data context class

Views:

☒ Generate views

☒ Reference script libraries

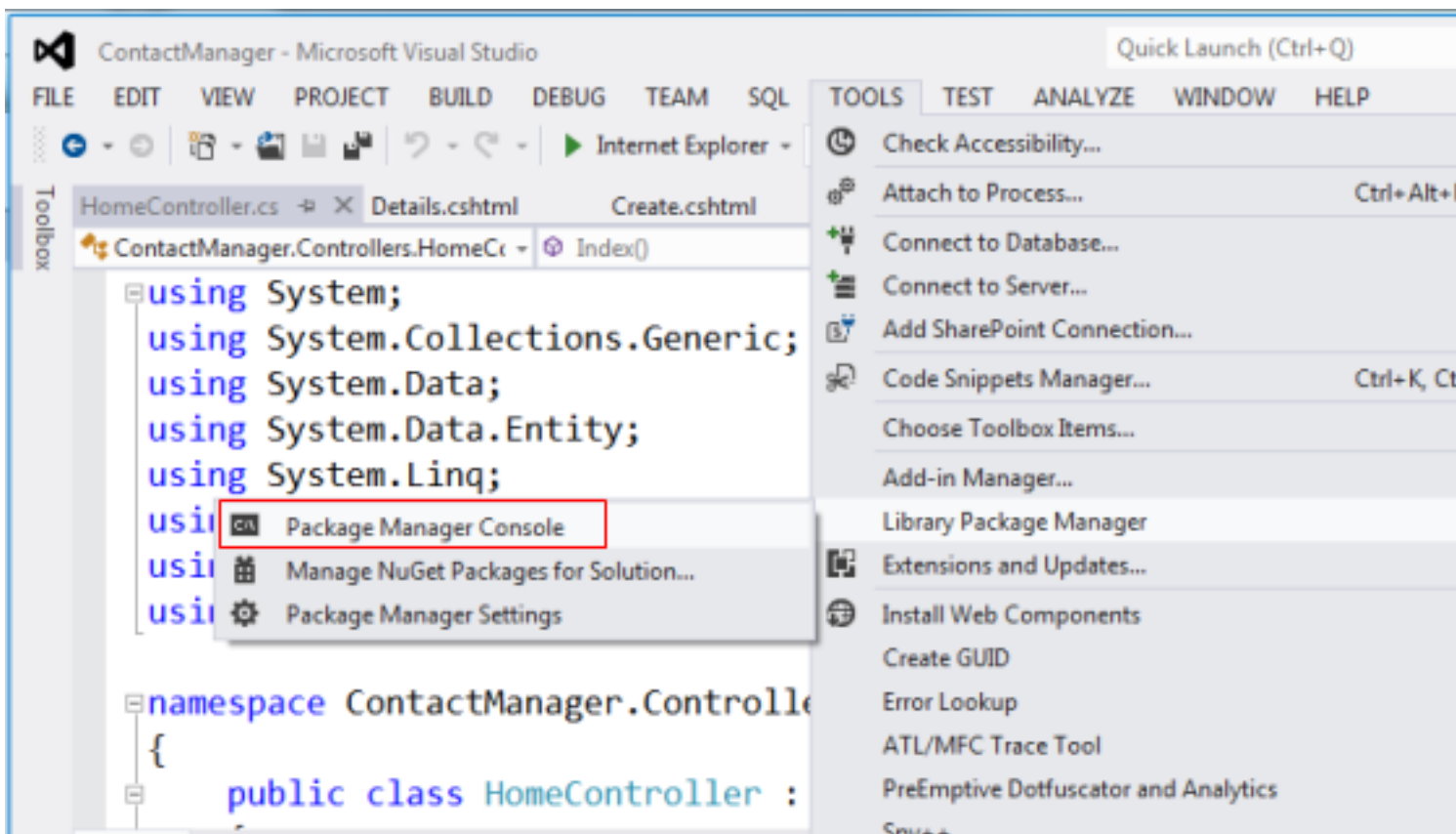
☒ Use a layout page:

(Leave empty if it is set in a Razor \_viewstart file)

AddCancel

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 مهاجرت‌ها، 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* می‌سازد 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 پی‌گیری کنید.

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

```
add-migration Initial
```

فرمان **add-migration initial** فایلی با نام **initial <data\_stamp>** ساخته و آن را در پوشه *Migrations* ذخیره می‌کند. در این مرحله دیتابیس شما ایجاد می‌شود. در این فرمان، مقدار **initial** اختیاری است و صرفاً 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
```

```

Package Manager Console
Package source: NuGet official package source
PM> enable-migrations
Checking if the context targets an existing database...
Code First Migrations enabled for project ContactManager.
PM> add-migration Initial
Scaffolding migration 'Initial'.
The Designer Code for this migration file includes a snapshot of your current Code Fi
model. This snapshot is used to calculate the changes to your model when you scaffold
the next migration. If you make additional changes to your model that you want to
include in this migration, then you can re-scaffold it by running 'Add-Migration
201209182148203_Initial' again.
PM> update-database
Specify the '-Verbose' flag to view the SQL statements being applied to the target
database.
Applying code-based migrations: [201209182148203_Initial].
Applying code-based migration: 201209182148203_Initial.
Running Seed method.
PM> |
100 %

```

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.

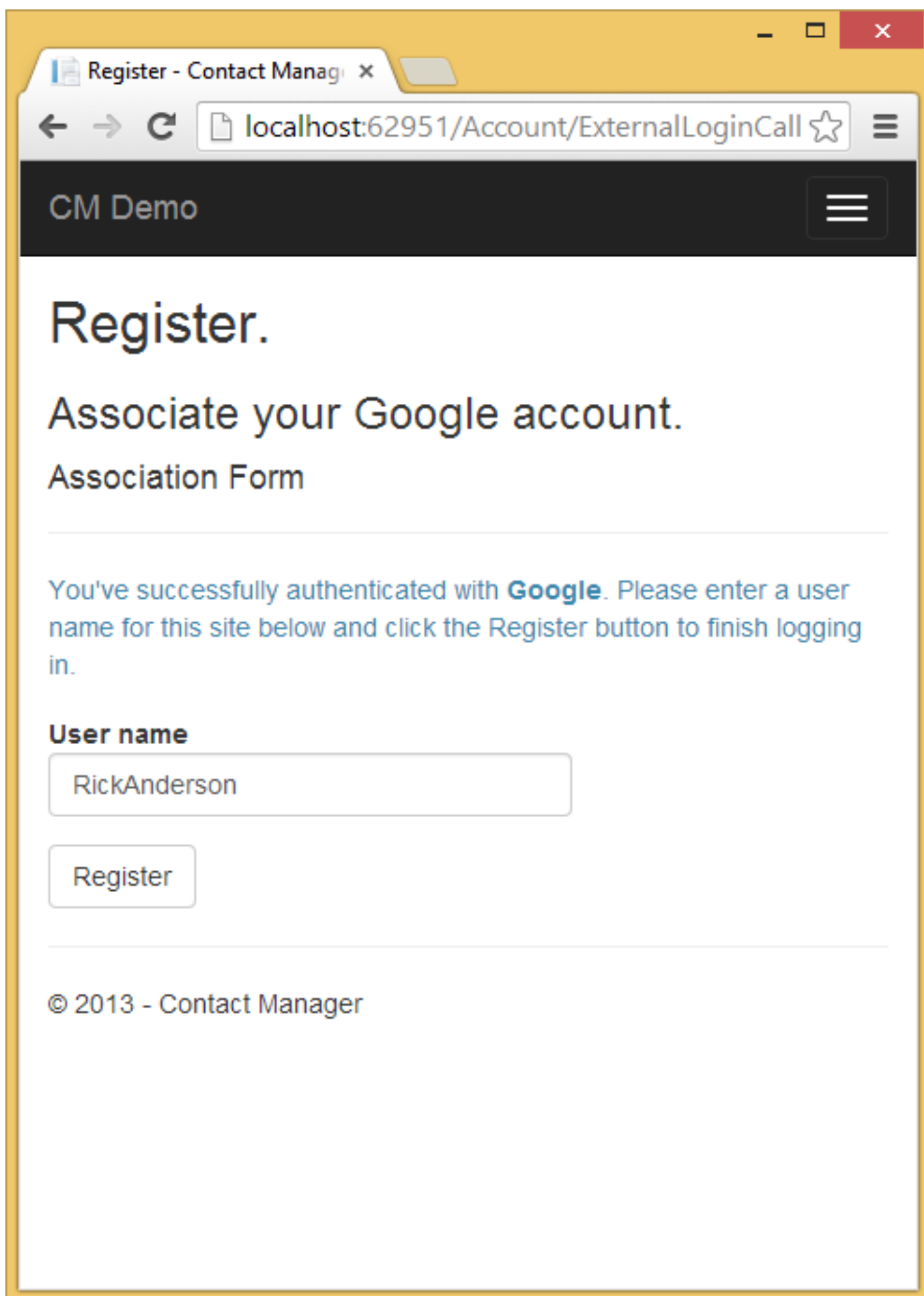
همانطور که مشاهده می‌کنید، اپلیکیشن داده‌های اولیه (Seed) را نمایش می‌دهد، و لینک‌هایی هم برای ویرایش، حذف و مشاهده جزئیات رکوردها فراهم می‌کند. می‌توانید داده‌ها را مشاهده کنید، رکورد جدید ثبت کنید و یا داده‌های قبلی را ویرایش و حذف کنید.



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 از internet 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. برای اضافه کردن این تامین کنندگان باید دنبال کنید، بسیار مشابه همین مراحل 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. نهایتاً روی **Register** کلیک کنید.



The screenshot shows a web browser window with a single tab titled "Register - Contact Manag". The address bar displays "localhost:62951/Account/ExternalLoginCall". The page has a dark header with "CM Demo" and a hamburger menu icon. The main content area features a large heading "Register." followed by the sub-heading "Associate your Google account." and "Association Form". A message states: "You've successfully authenticated with **Google**. Please enter a user name for this site below and click the Register button to finish logging in." Below this is a text input field containing "RickAnderson" and a "Register" button. The footer shows "© 2013 - Contact Manager".

Register - Contact Manag x

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**

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 مثلاً *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* می‌سازد. 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. یا 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. بعداً 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");
        }

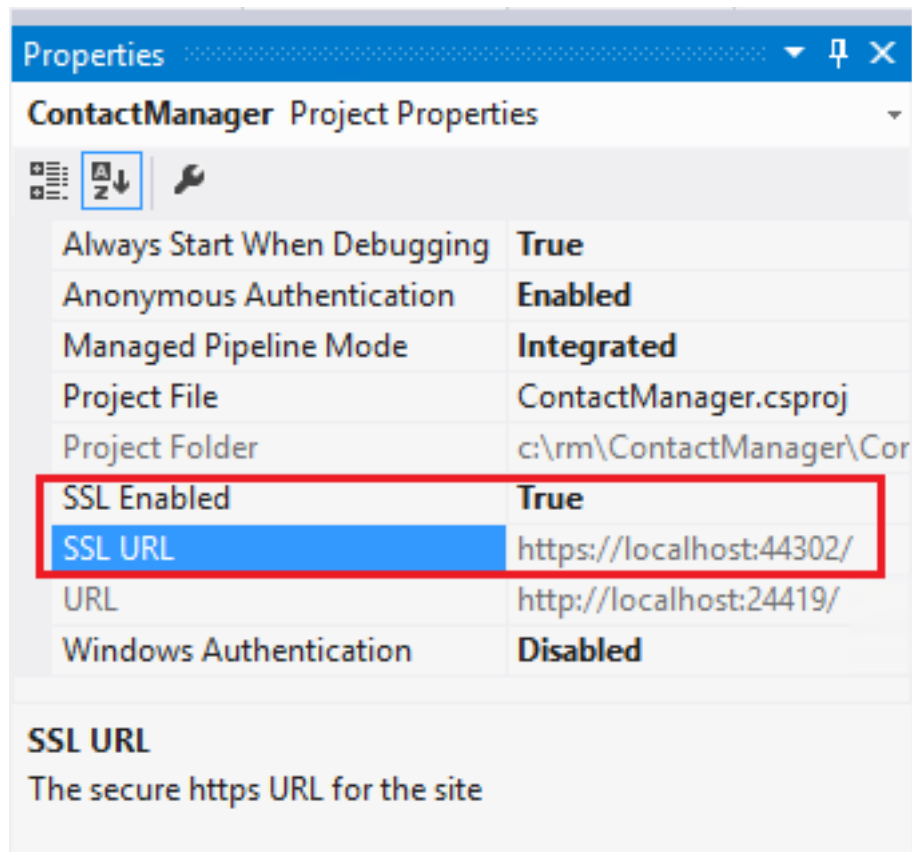
        return View(contact);
    }

    // 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);
    }
}

```

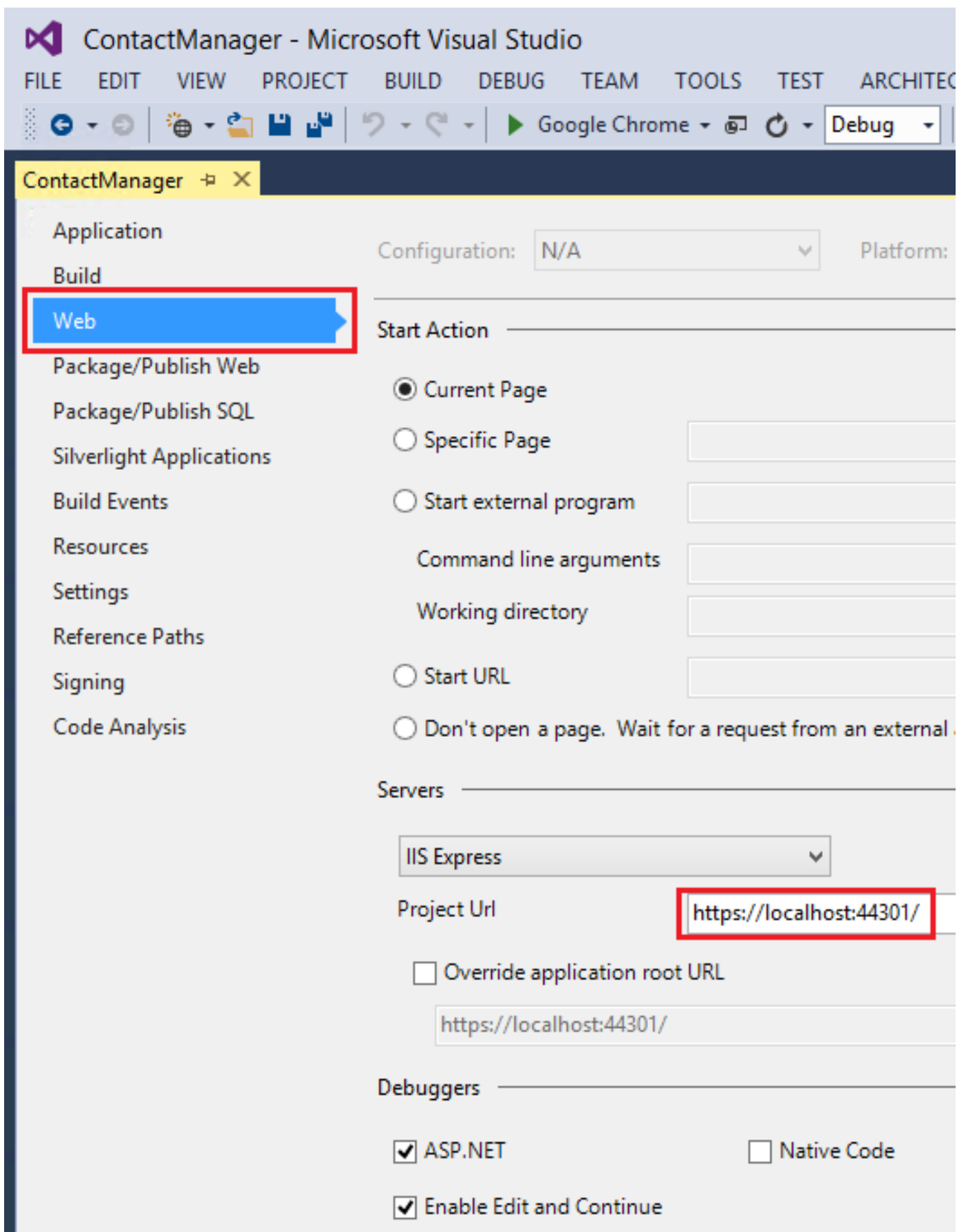
## فعال سازی SSL برای پروژه

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.



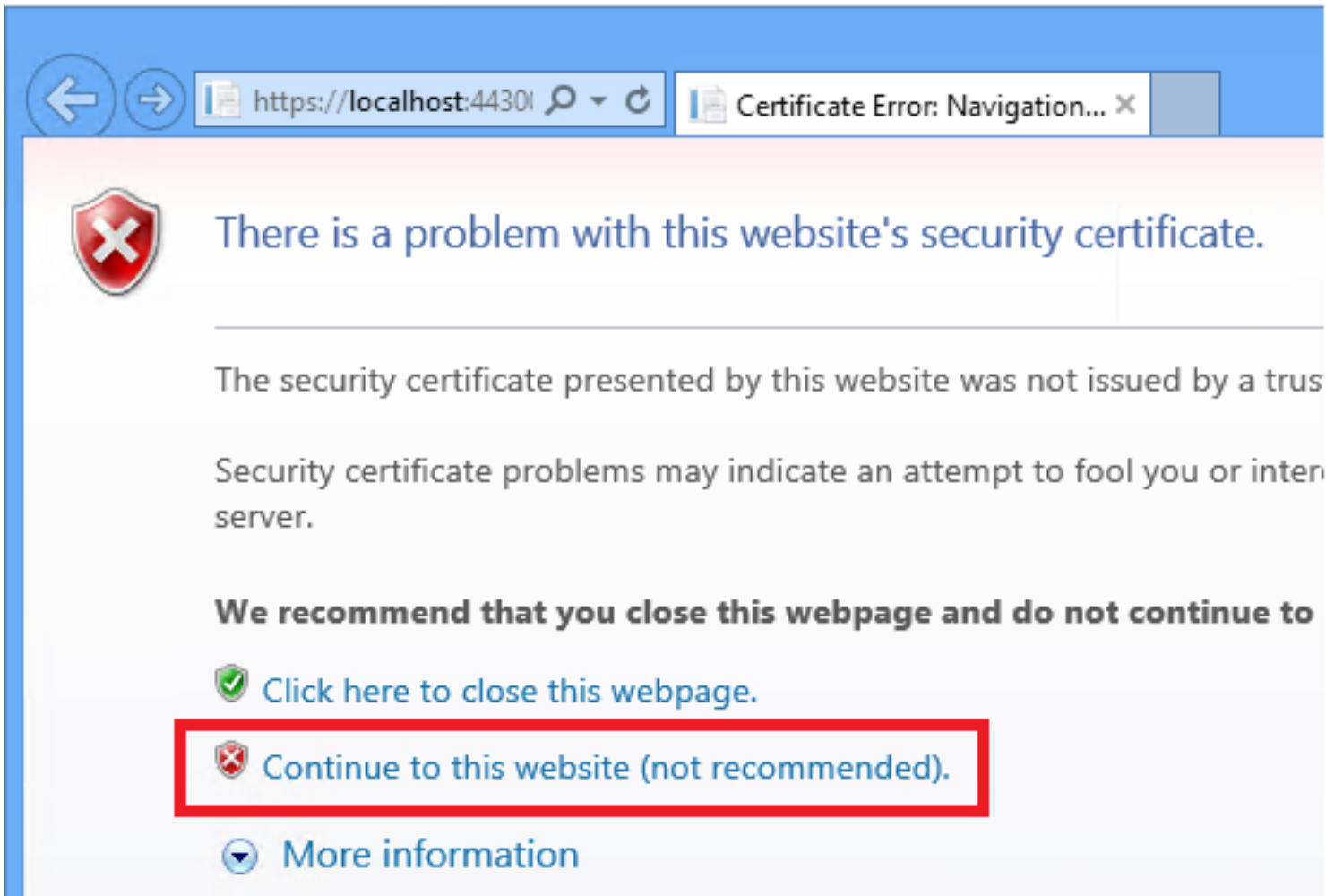
روی نام پروژه کلیک راست کنید و **Properties** را انتخاب کنید. در قسمت چپ گزینه **Web** را انتخاب کنید. حالا مقدار **Project Url** را به آدرسی که کپی کرده اید تغییر دهید. نهایتاً تغییرات را ذخیره کنید و پنجره را ببندید.

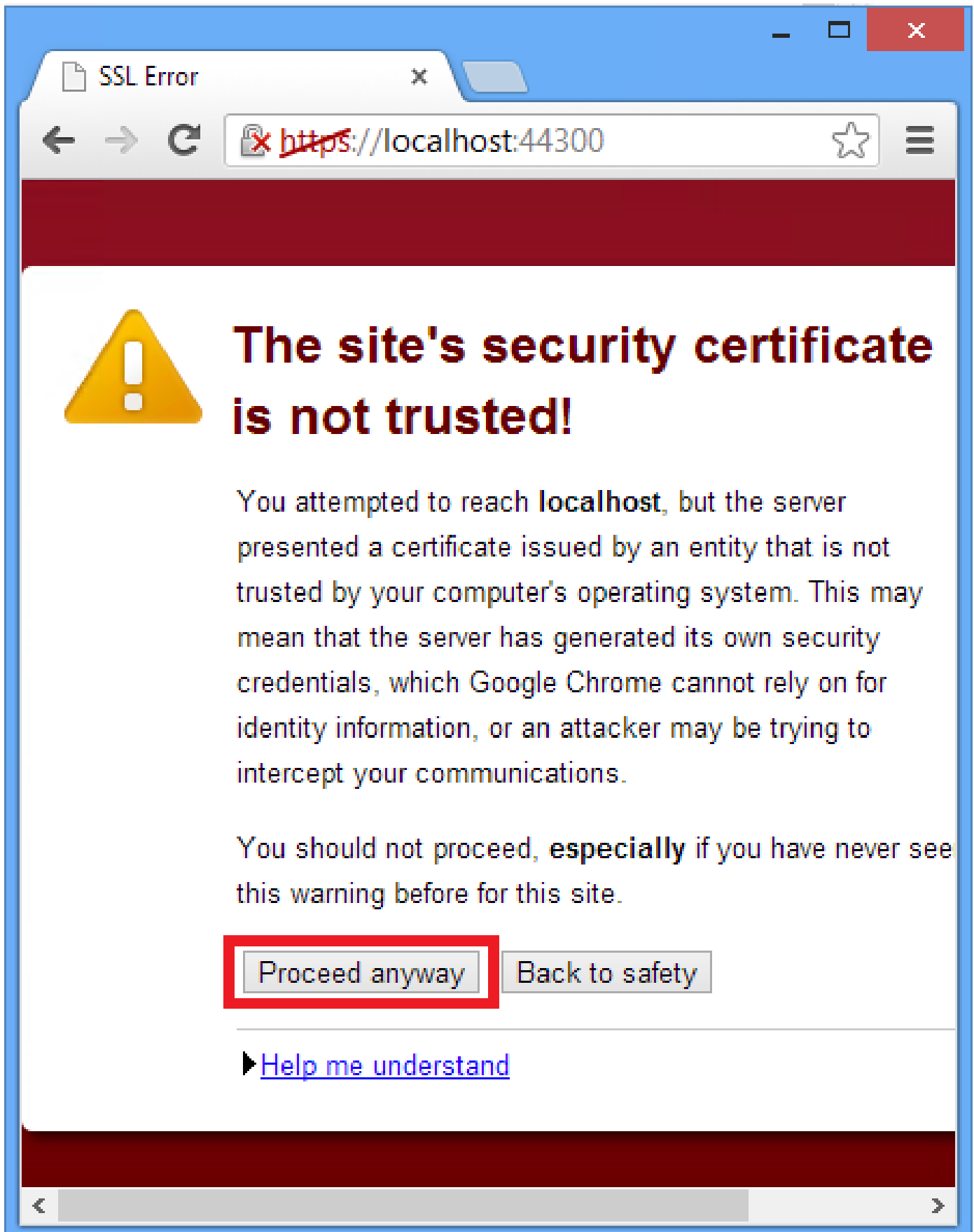




حال پروژه را اجرا کنید. مرورگر شما باید یک پیام خطای اعتبارسنجی به شما بدهد. دلیلش این است که اپلیکیشن شما از یک

**Valid Certificate** استفاده نمی‌کند. هنگامی که پروژه را روی Windows Azure منتشر کنید دیگر این پیام 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.

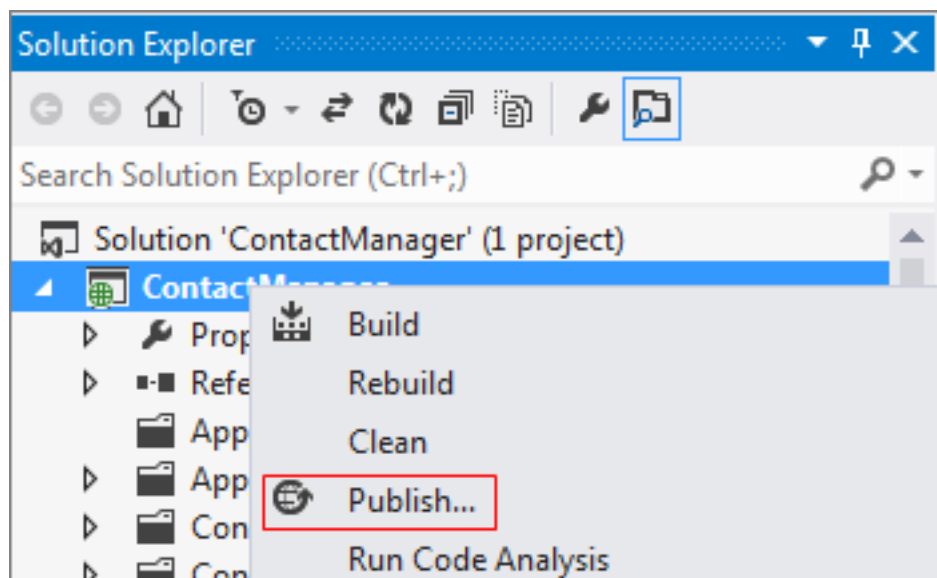
با کاربر *user1* که قبلاً ساختید وارد سایت شوید. حال به صفحه ویرایشی که قبلاً درخواست کرده بودید هدایت می‌شوید.

اگر نتوانستید با این کاربر به سایت وارد شوید، کلمه عبور را از سورس کد کپی کنید و مجدداً 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 مراحل 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)

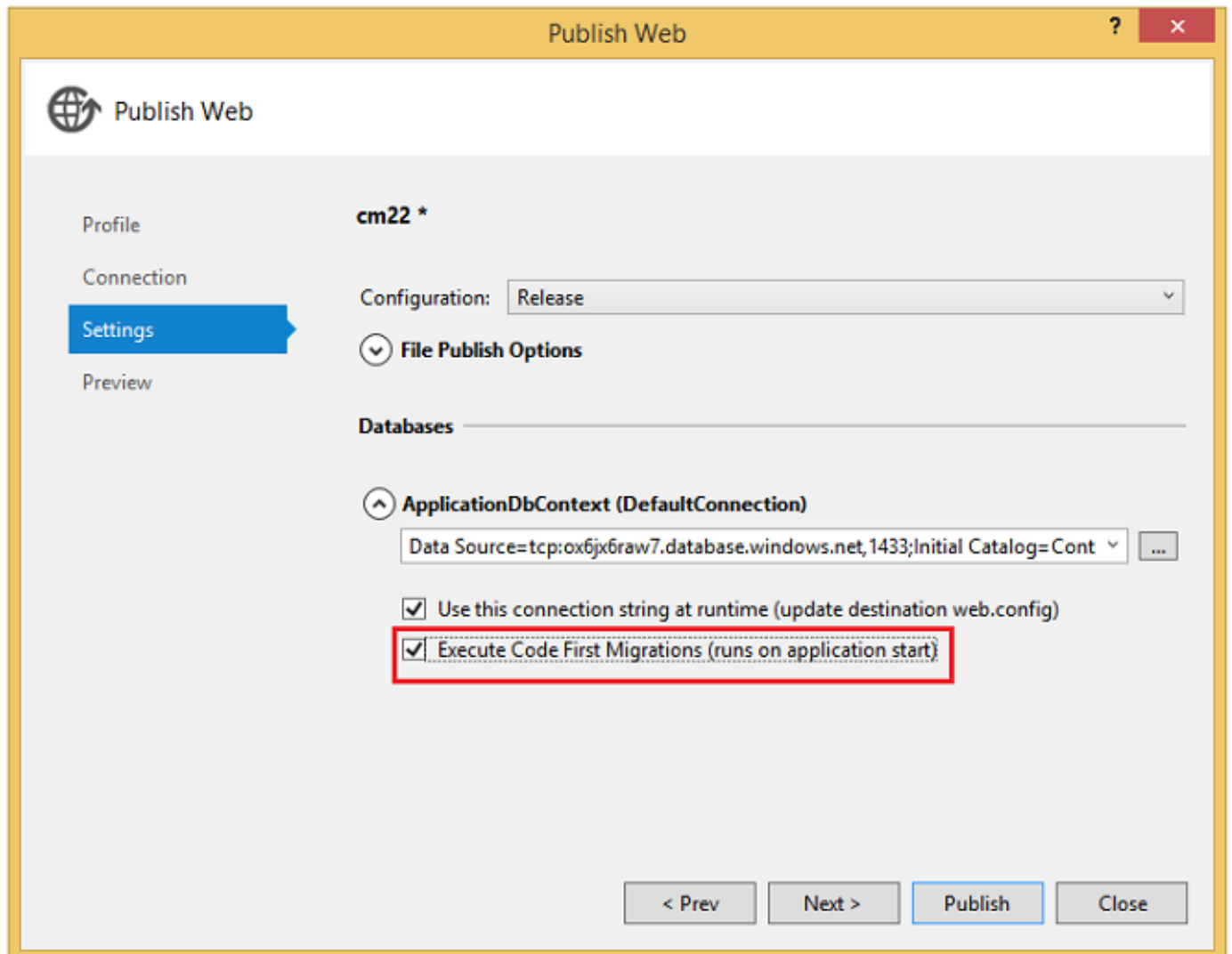
< Prev

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.

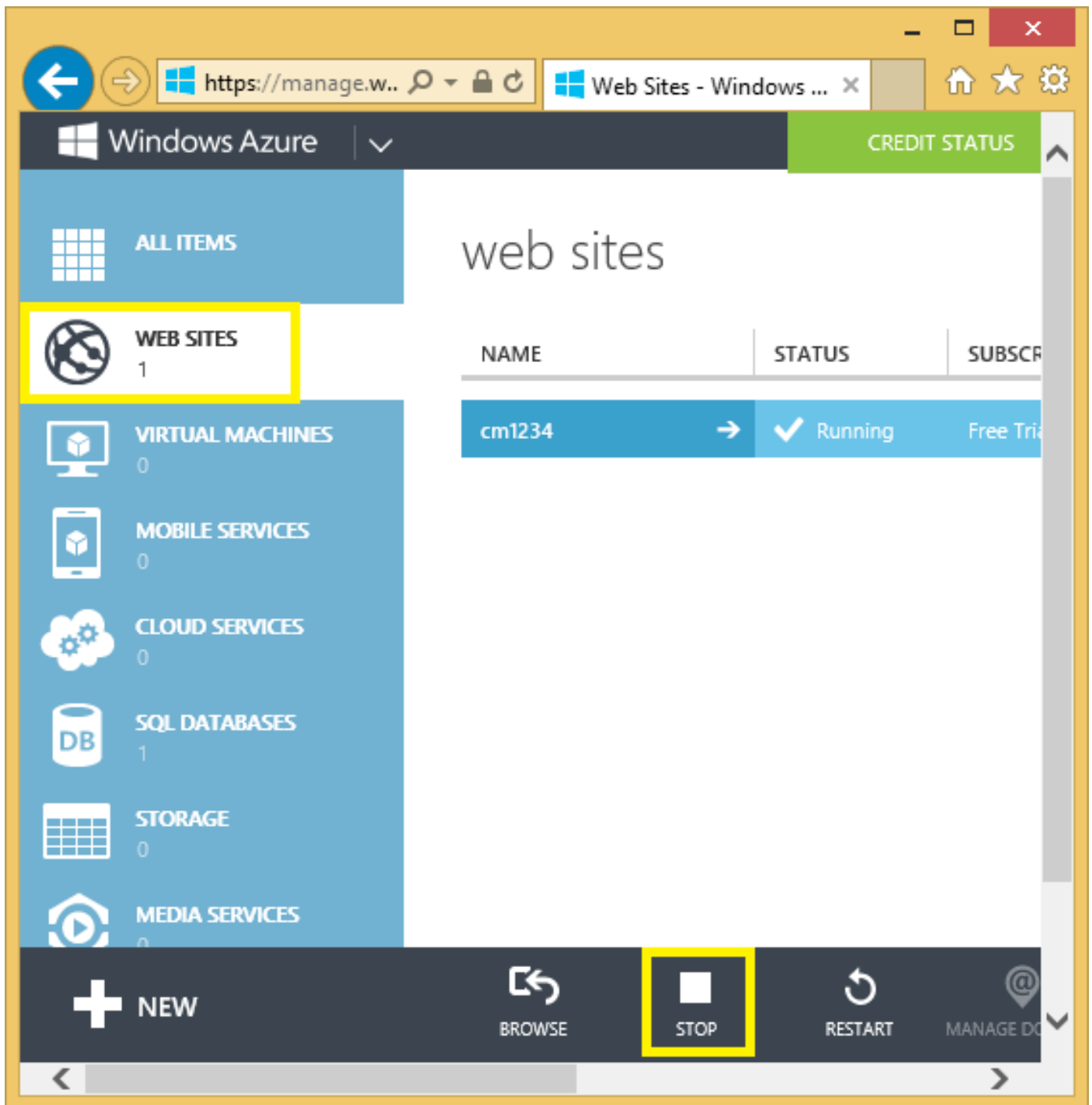
حال از سایت خارج شوید و توسط یک اکانت Google یا Facebook وارد سایت شوید، که در این صورت نقش canEdit نیز به شما تعلق می‌گیرد.

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.



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

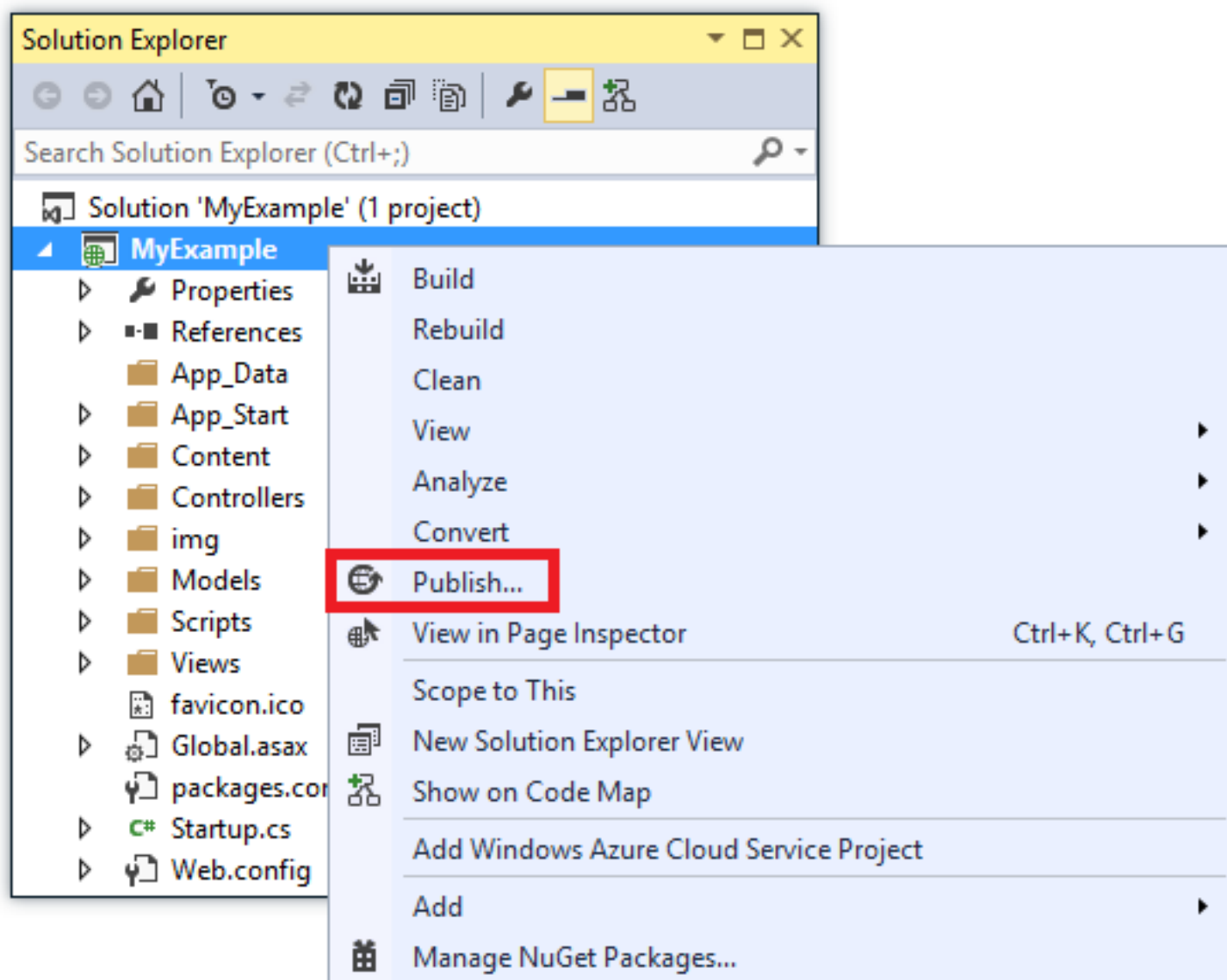


فراخوانی `AddToRoleAsync` را حذف و اپلیکیشن را منتشر و تست کنید  
کنترلر `Account` را باز کنید و کد زیر را از متد `ExternalLoginConfirmation` حذف کنید.

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

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





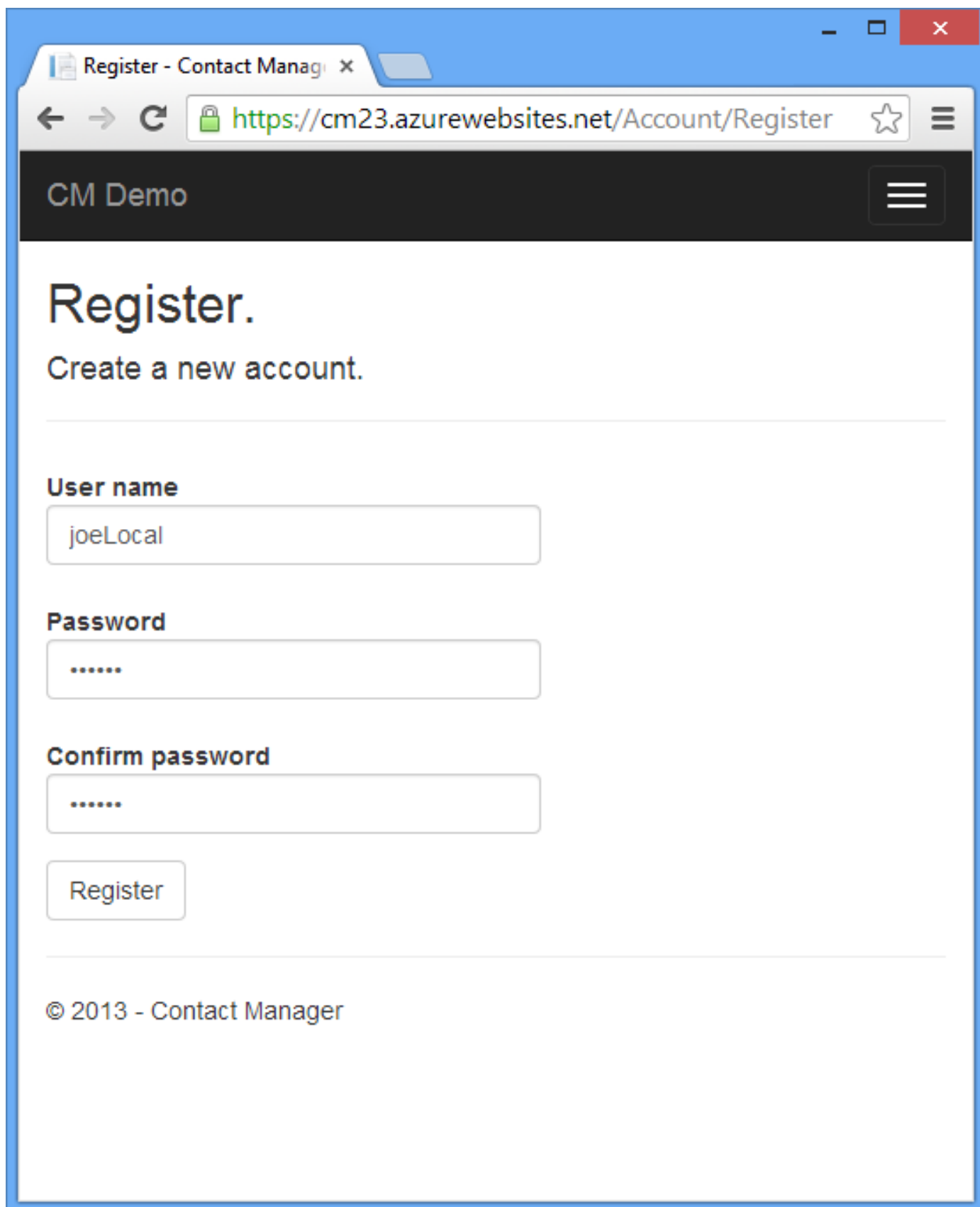
دکمه **Start Preview** را فشار دهید. در این مرحله تنها فایل هایی که نیاز به بروز رسانی دارند آپلود خواهند شد.

وب سایت را راه اندازی کنید. ساده ترین راه از طریق پرتال مدیریت Windows Azure است. توجه داشته باشید که تا هنگامی که وب سایت شما متوقف شده، نمی توانید 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.

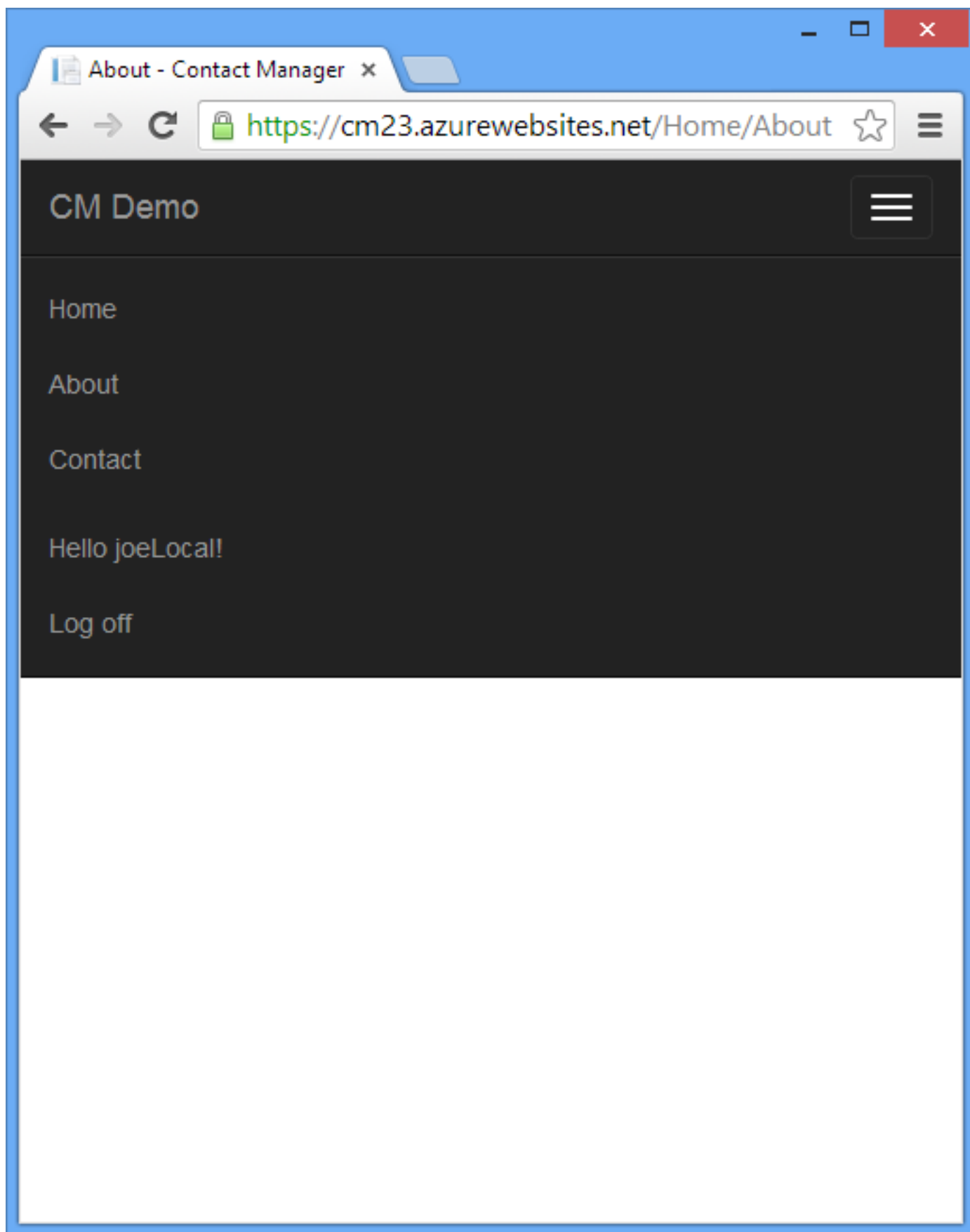
روی لینک **About** کلیک کنید، که شما را به صفحه ورود هدایت می کند.

روی لینک **Register** در صفحه ورود کلیک کنید و یک حساب کاربری محلی بسازید. از این حساب کاربری برای این استفاده می کنیم که ببینیم شما به صفحات فقط خواندنی (read-only) و نه صفحاتی که داده ها را تغییر می دهند دسترسی دارید یا خیر. بعداً در ادامه مقاله، دسترسی حساب های کاربری محلی (local) را حذف می کنیم.



The screenshot shows a web browser window with a single tab titled "Register - Contact Manag". The address bar displays the URL "https://cm23.azurewebsites.net/Account/Register". The page has a dark header bar with the text "CM Demo" and a hamburger menu icon. The main content area is titled "Register." with the subtitle "Create a new account." Below this, there are three input fields: "User name" containing "joeLocal", "Password" with masked characters "\*\*\*\*\*", and "Confirm password" also with masked characters "\*\*\*\*\*". A "Register" button is positioned below the confirm password field. At the bottom of the page, the footer text reads "© 2013 - Contact Manager".

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 قبلاً ساختید وارد شوید.

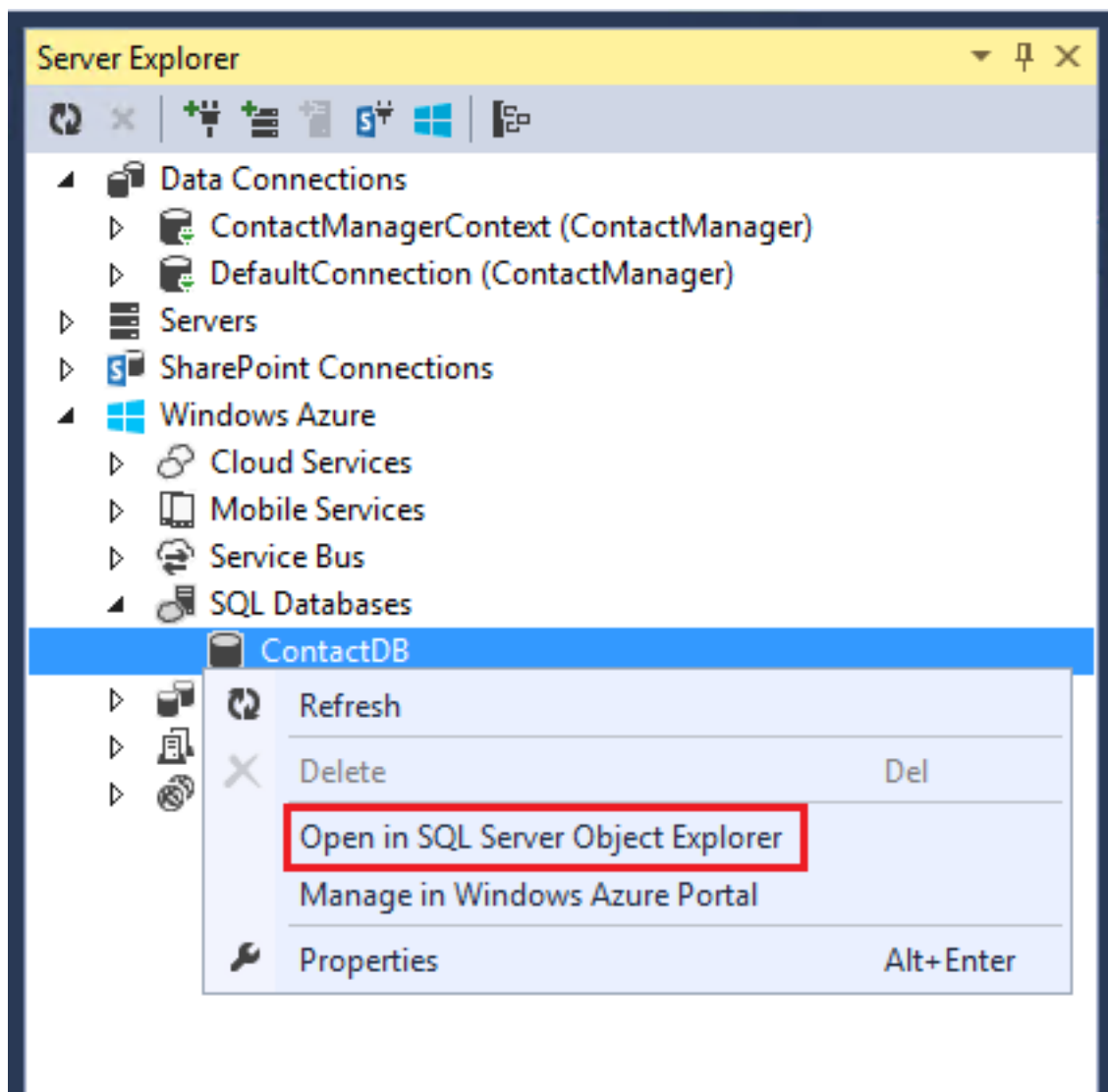
حال بررسی کنید که امکان ویرایش اطلاعات را دارید یا خیر.

**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**

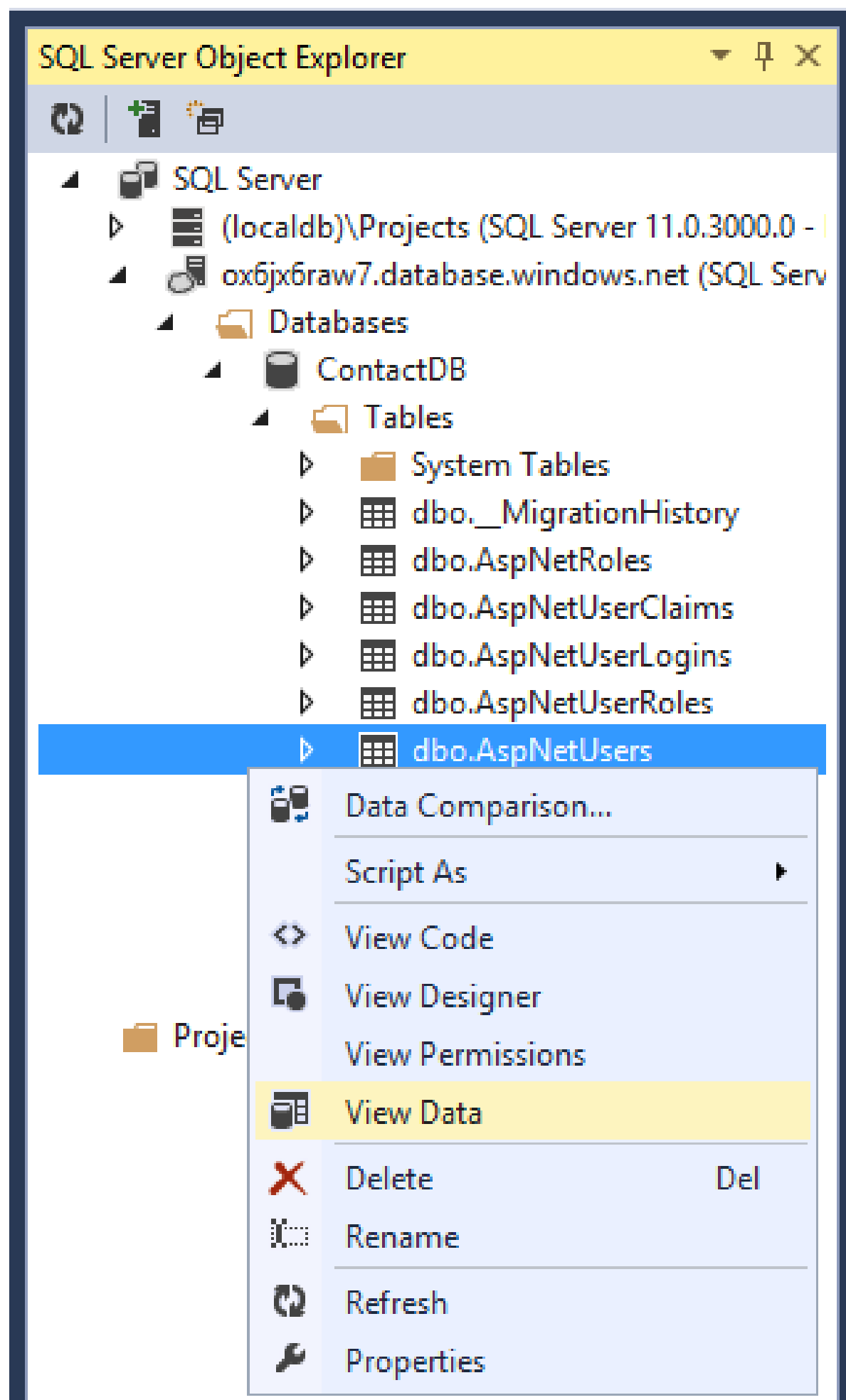
در **Server Explorer** دیتابیس **ContactDB** را پیدا کنید. روی آن کلیک راست کرده و **Open in SQL Server Object Explorer** را

انتخاب کنید.



**توجه:** اگر نمی‌توانید گره **SQL Databases** را باز کنید و یا **ContactDB** را در ویژوال استودیو نمی‌بینید، باید مراحل را طی کنید تا یک پورت یا یکسری پورت را به فایروال خود اضافه کنید. دقت داشته باشید که در صورت اضافه کردن Port Range ها 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.



dbo.AspNetUsers [Data]					
Max Rows: 1000					
	Id	UserName	PasswordHash	Secu...	Discriminator
	3b7fd83f-fc68-4f4d-ae27-b9922d17602d	JoeLocalUser	AGgn9Gfmwx...	c3337...	ApplicationUser
▶	8a5687c2-86ed-40c9-853b-26ef40b7a2bb	RickGM000	NULL	5489a...	ApplicationUser
	94715c84-9919-4146-ad2b-0cf692c7c70d	JoeLocalUser2	AOIMz9t+/+z...	d9f47...	ApplicationUser
	9af370af-8055-4cef-965f-990214c24ccf	user1	AJjTVn2u86D...	7f3ab...	ApplicationUser
	a18b44da-a9ac-4153-89a0-d9c808f22df8	joeLocal	ACDwfl01Klf...	2977f...	ApplicationUser
*	NULL	NULL	NULL	NULL	NULL

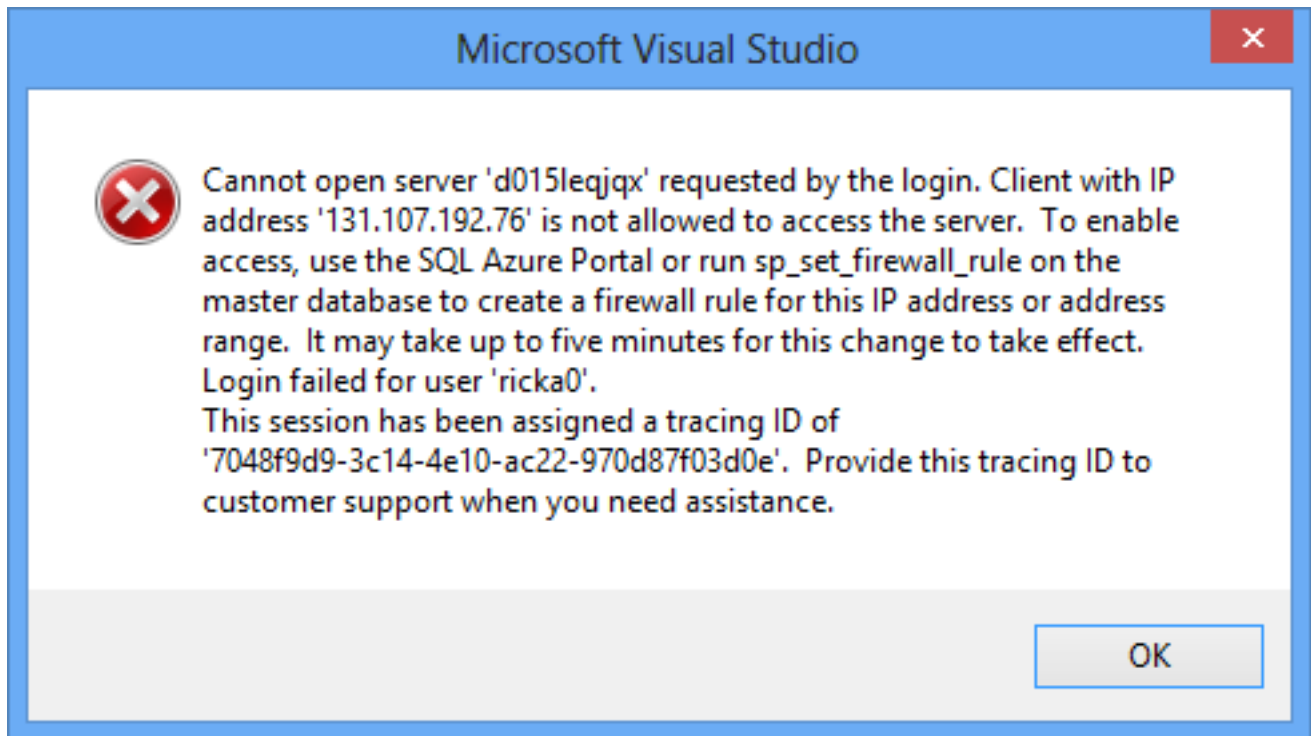
حالا روی **AspNetUserRoles** کلیک راست کنید و **View Data** را انتخاب کنید.

dbo.AspNetUserRoles [Data]		dbo.AspNetUsers [Data]	
Max Rows: 1000			
	Userid	Roleid	
	8a5687c2-86ed-40c9-853b-26ef40b7a2bb	e43a4145-7089-4292-9057-af56e5d8e940	
	9af370af-8055-4cef-965f-990214c24ccf	e43a4145-7089-4292-9057-af56e5d8e940	
▶*	NULL	NULL	

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

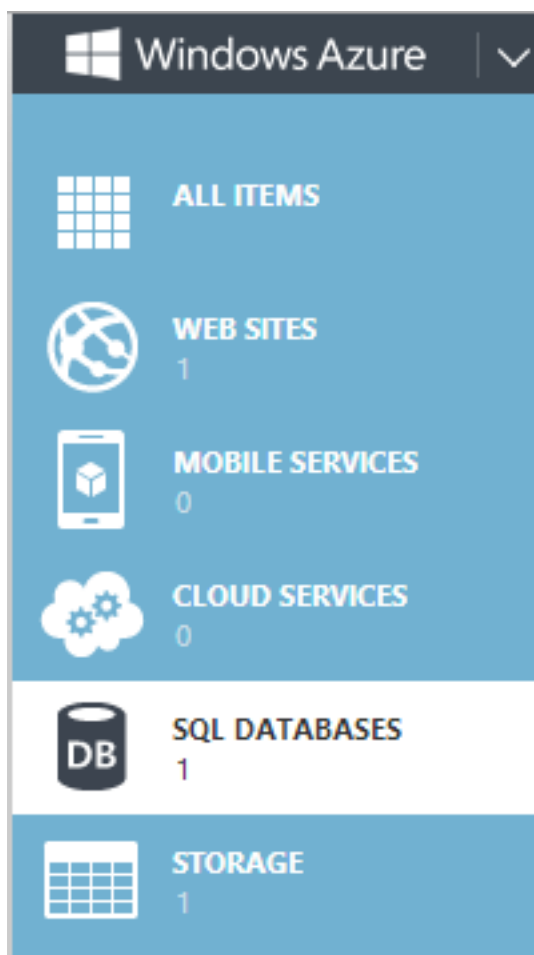
#### Cannot open server login error

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



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





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

## Get Microsoft database design tools ?

[Install Microsoft SQL Server Data Tools](#)

## Design your SQL Database ?

[Download a starter project for your SQL Database this IP address](#)

[Set up Windows Azure firewall rules for this IP address](#)

## Connect to your database ?

[Design your SQL Database](#)[Run Transact-SQL queries against your SQL Database](#)[View SQL Database connection strings for ADO .Net, ODBC, PHP, and JDBC](#)

Server: d0151eqjqx.database.windows.net,1433

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

مرحله بعد اضافه کردن محدوده آدرس‌های مجاز است.

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

SERVER	EDITION	MAX SIZE	
d015leqjqx	Web	1 GB	

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

# d015leqjqx

DASHBOARD
DATABASES
CONFIGURE
HISTORY

## allowed ip addresses

CURRENT CLIENT IP ADDRESS

131.107.174.240

ADD TO THE ALLOWED IP ADDRESSES.

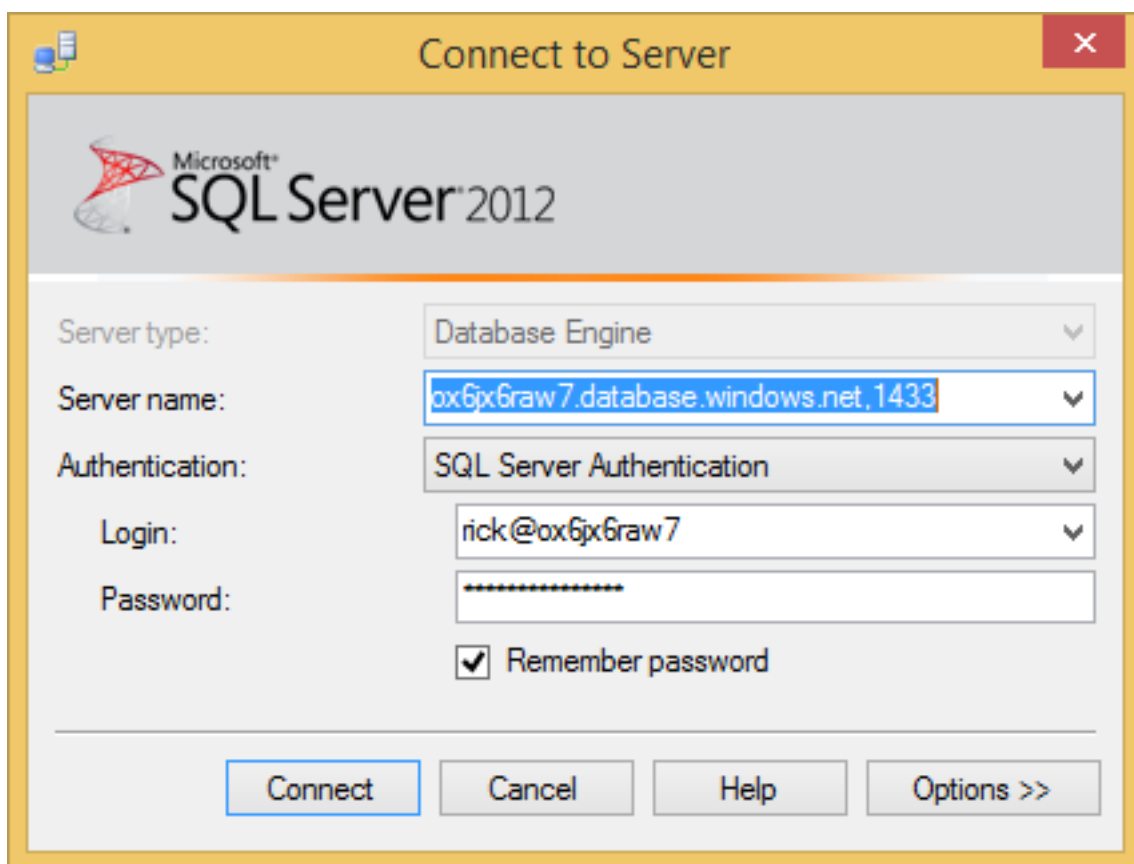
131.107.147.240	131.107.147.240	131.107.147.240
pc1	131.107.000.000	131.107.255.255
RULE NAME	START IP ADDRESS	END IP ADDRESS

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 مراحل 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. مراحل 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.

dbo.AspNetUserRoles [Data]		dbo.AspNetUsers [Data]	
		Max Rows:	1000
UserId	RoleId		
8a5687c2-86ed-40c9-853b-26ef40b7a2bb	e43a4145-7089-4292-9057-af56e5d8e940		
9af370af-8055-4cef-965f-990214c24ccf	e43a4145-7089-4292-9057-af56e5d8e940		
▶*	NULL		

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

dbo.AspNetUserRoles [Data]		dbo.AspNetUsers [Data]	
		Max Rows:	1000
UserId	RoleId		
8a5687c2-86ed-40c9-853b-26ef40b7a2bb	e43a4145-7089-4292-9057-af56e5d8e940		
9af370af-8055-4cef-965f-990214c24ccf	e43a4145-7089-4292-9057-af56e5d8e940		
✎			
*	NULL		

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](#)

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 می کند.

## نظرات خوانندگان

نویسنده: مهمان  
تاریخ: ۱۴:۴ ۱۳۹۲/۱۰/۱۹

دوست عزیز

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

در این مقاله جایگزینی پایاده سازی پیش فرض [ASP.NET Identity](#) را بررسی می‌کنیم. در ادامه خواهید خواند:

جزئیات نحوه پایاده سازی یک Storage Provider برای ASP.NET Identity

تشریح اینترفیس هایی که باید پایاده سازی شوند، و نحوه استفاده از آنها در ASP.NET Identity

ایجاد یک دیتابیس MySQL روی Windows Azure

نحوه استفاده از یک ابزار کلاینت (MySQL Workbench) برای مدیریت دیتابیس مذکور

نحوه جایگزینی پایاده سازی سفارشی با نسخه پیش فرض در یک اپلیکیشن ASP.NET MVC

در انتهای این مقاله یک اپلیکیشن ASP.NET MVC خواهیم داشت که از ASP.NET Identity و تامین کننده سفارشی جدید استفاده می‌کند. دیتابیس اپلیکیشن MySQL خواهد بود و روی Windows Azure میزبانی می‌شود. سورس کد کامل این مثال را هم می‌توانید از [این لینک](#) دریافت کنید.

### پایاده سازی یک Storage Provider سفارشی برای ASP.NET Identity

ASP.NET Identity سیستم توسعه پذیری است که می‌توانید بخش‌های مختلف آن را جایگزین کنید. در این سیستم بناهای سطح بالایی مانند Managers و Stores وجود دارند.

Managers کلاس‌های سطح بالایی هستند که توسعه دهندگان از آنها برای اجرای عملیات مختلف روی ASP.NET Identity استفاده می‌کنند. مدیریت کننده‌های موجود عبارتند از UserManager و RoleManager. کلاس UserManager برای اجرای عملیات مختلف روی کاربران استفاده می‌شود، مثلاً ایجاد کاربر جدید یا حذف آنها. کلاس RoleManager هم برای اجرای عملیات مختلف روی نقش‌ها استفاده می‌شود.

Stores کلاس‌های سطح پایین‌تری هستند که جزئیات پایاده سازی را در بر می‌گیرند، مثلاً اینکه موجودیت‌های کاربران و نقش‌ها چگونه باید ذخیره و بازیابی شوند. این کلاس‌ها با مکانیزم ذخیره و بازیابی تلفیق شده اند. مثلاً

Microsoft.AspNet.Identity.EntityFramework کلاسی با نام UserStore دارد که برای ذخیره و بازیابی Userها و داده‌های مربوطه توسط EntityFramework استفاده می‌شود.

Managers از Stores تفکیک شده اند و هیچ وابستگی ای به یکدیگر ندارند. این تفکیک بدین منظور انجام شده که بتوانید مکانیزم ذخیره و بازیابی را جایگزین کنید، بدون اینکه اپلیکیشن شما از کار بیافتد یا نیاز به توسعه بیشتر داشته باشد. کلاس‌های Manager می‌توانند با هر Store ای ارتباط برقرار کنند. از آنجا که شما از APIهای سطح بالای UserManager برای انجام عملیات CRUD روی کاربران استفاده می‌کنید، اگر UserStore را با پایاده سازی دیگری جایگزین کنید، مثلاً AzureTable Storage یا MySql، نیازی به بازنویسی اپلیکیشن نیست.

در مثال جاری پایاده سازی پیش فرض Entity Framework را با یک تامین کننده MySQL جایگزین می‌کنیم.

پایاده سازی کلاس‌های Storage

برای پایاده سازی تامین کننده‌های سفارشی، باید کلاس هایی را پایاده سازی کنید که همتای آنها در

Microsoft.AspNet.Identity.EntityFramework وجود دارند:

UserStore<TUser>

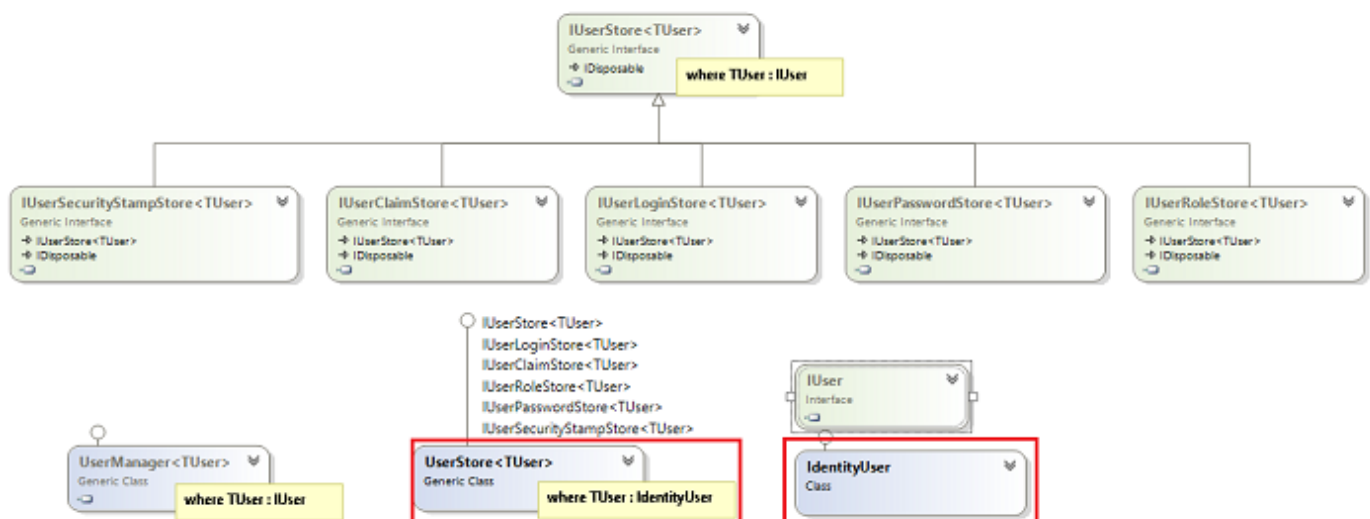
IdentityUser

RoleStore<TRole>

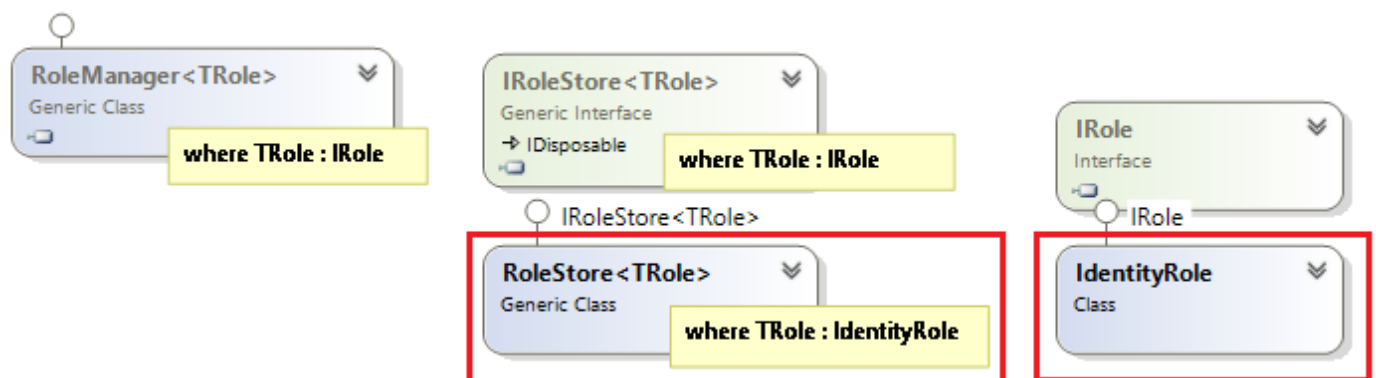
IdentityRole

پایاده سازی پیش فرض Entity Framework را در تصاویر زیر مشاهده می‌کنید.

Users



## Roles



در مخزن پیش فرض ASP.NET Identity EntityFramework کلاس‌های بیشتری برای موجودیت‌ها مشاهده می‌کنید.

IdentityUserClaim

IdentityUserLogin

IdentityUserRole

همانطور که از نام این کلاس‌ها مشخص است، اختیارات، نقش‌ها و اطلاعات ورود کاربران توسط این کلاس‌ها معرفی می‌شوند. در مثال جاری این کلاس‌ها را پیاده سازی نخواهیم کرد، چرا که بارگذاری اینگونه رکوردها از دیتابیس به حافظه برای انجام عملیات پایه (مانند افزودن و حذف اختیارات کاربران) سنگین است. در عوض کلاس‌های backend store اینگونه عملیات را بصورت مستقیم روی دیتابیس اجرا خواهند کرد. بعنوان نمونه متد `UserStore.GetClaimsAsync()` را در نظر بگیرید. این متد به نوبه خود متد `userClaimTable.FindByUserId(user.Id)` را فراخوانی می‌کند که یک کوئری روی جدول مربوطه اجرا می‌کند و لیستی از اختیارات کاربر را بر می‌گرداند.

```
public Task<IList<Claim>> GetClaimsAsync(IdentityUser user)
{
    ClaimsIdentity identity = userClaimsTable.FindByUserId(user.Id);
    return Task.FromResult<IList<Claim>>(identity.Claims.ToList());
}
```



```
}
```

برای پیاده سازی یک تامین کننده سفارشی MySQL مراحل زیر را دنبال کنید.  
1. کلاس کاربر را ایجاد کنید، که اینترفیس **IUser** را پیاده سازی می کند.

```
public class IdentityUser : IUser
{
    public IdentityUser(){...}
    public IdentityUser(string userName) (){...}
    public string Id { get; set; }
    public string Username { get; set; }
    public string PasswordHash { get; set; }
    public string SecurityStamp { get; set; }
}
```

2. کلاس User Store را ایجاد کنید، که اینترفیس های **IUserStore** , **IUserClaimStore** , **IUserLoginStore** , **IUserRoleStore** و **IUserPasswordStore** را پیاده سازی می کند. توجه کنید که تنها اینترفیس **IUserStore** را باید پیاده سازی کنید، مگر آنکه بخواهید از امکاناتی که دیگر اینترفیس ها ارائه می کنند هم استفاده کنید.

```
public class UserStore : IUserStore<IdentityUser>,
                        IUserClaimStore<IdentityUser>,
                        IUserLoginStore<IdentityUser>,
                        IUserRoleStore<IdentityUser>,
                        IUserPasswordStore<IdentityUser>
{
    public UserStore(){...}
    public Task CreateAsync(IdentityUser user){...}
    public Task<IdentityUser> FindByIdAsync(string userId){...}
    ...
}
```

3. کلاس Role را ایجاد کنید که اینترفیس **IRole** را پیاده سازی می کند.

```
public class IdentityRole : IRole
{
    public IdentityRole(){...}
    public IdentityRole(string roleName) (){...}
    public string Id { get; set; }
    public string Name { get; set; }
}
```

4. کلاس Role Store را ایجاد کنید که اینترفیس **IRoleStore** را پیاده سازی می کند. توجه داشته باشید که پیاده سازی این مخزن اختیاری است و در صورتی لازم است که بخواهید از نقش ها در سیستم خود استفاده کنید.

```
public class RoleStore : IRoleStore<IdentityRole>
{
    public RoleStore(){...}
    public Task CreateAsync(IdentityRole role){...}
    public Task<IdentityRole> FindByIdAsync(string roleId){...}
    ....
}
```

کلاس های بیشتری هم وجود دارند که مختص پیاده سازی مثال جاری هستند.

**MySQLDatabase:** این کلاس اتصال دیتابیس MySQL و کوئری‌ها را کپسوله می‌کند. کلاس‌های UserStore و RoleStore توسط نمونه ای از این کلاس و هله سازی می‌شوند.

**RoleTable:** این کلاس جدول Roles و عملیات CRUD مربوط به آن را کپسوله می‌کند.

**UserClaimsTable:** این کلاس جدول UserClaims و عملیات CRUD مربوط به آن را کپسوله می‌کند.

**UserLoginsTable:** این کلاس جدول UserLogins و عملیات CRUD مربوط به آن را کپسوله می‌کند.

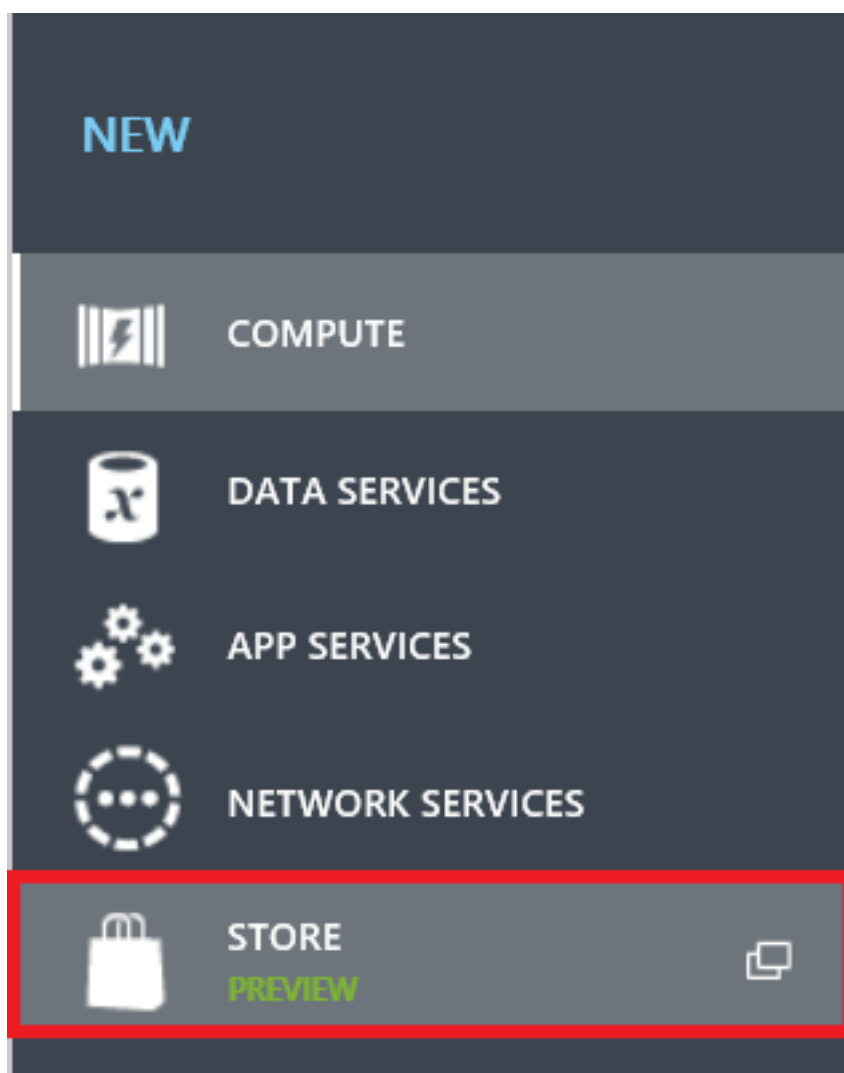
**UserRolesTable:** این کلاس جدول UserRoles و عملیات CRUD مربوطه به آن را کپسوله می‌کند.

**UserTable:** این کلاس جدول Users و عملیات CRUD مربوط به آن را کپسوله می‌کند.

ایجاد یک دیتابیس MySQL روی Windows Azure

1. به [پورتال مدیریتی Windows Azure](#) وارد شوید.

2. در پایین صفحه روی **+NEW** کلیک کنید و گزینه **STORE** را انتخاب نمایید.



در ویزارد **Choose Add-on** به سمت پایین اسکرول کنید و گزینه **ClearDB MySQL Database** را انتخاب کنید. سپس به مرحله بعد بروید.

# Choose an Add-on

ALL

APP SERVICES

DATA



ClearDB MySQL Database



ClearPointe Azure Management



cloudinary




D&B Business Insight




Embarke Email Analytics



Engine Yard Platform as a Service






## ClearDB MySQL Database

SuccessBricks, Inc.

ClearDB is a powerful, fault-tolerant database-as-a-service in the cloud for your MySQL powered applications.

PUBLISHED DATE

10/10/2012



2

3

4. راهکار **Free** بصورت پیش فرض انتخاب شده، همین گزینه را انتخاب کنید و نام دیتابیس را به **IdentityMySQLDatabase** تغییر دهید. نزدیک ترین ناحیه (region) به خود را انتخاب کنید و به مرحله بعد بروید.

PURCHASE FROM STORE

Personalize Add-on

PLANS (4)

☒ Free

Great for getting started and developing your apps.  
Includes 20 MB of storage and up to 4 connections.

0 USD/month

☐ Venus

Excellent for light test and staging apps that need a reliable MySQL database. Includes support for up to 1 GB of storage and up to 15 connections.

9.99 USD/month

PROMOTION CODE

?

NAME

✓

REGION

▼

5. روی علامت checkmark کلیک کنید تا دیتابیس شما ایجاد شود. پس از آنکه دیتابیس شما ساخته شد می توانید از قسمت **ADD-**

The screenshot shows the Azure portal interface. On the left sidebar, the 'ADD-ONS' menu item is highlighted with a red box. The main content area displays the 'add-ons' section with a 'PREVIEW' label. A table lists the add-ons, with 'IdentityMySQLDatabase' highlighted by a red box. The table has columns 'NAME' and 'TYPE'. The 'IdentityMySQLDatabase' add-on is of type 'App Service'. At the bottom right, the 'CONNECTION INFO' button is highlighted with a red box.

NAME	TYPE
IdentityMySQLDatabase	App Service

6. همانطور که در تصویر بالا می بینید، می توانید اطلاعات اتصال دیتابیس (connection info) را از پایین صفحه دریافت کنید.


7. اطلاعات اتصال را با کلیک کردن روی دکمه مجاور کپی کنید تا بعداً در اپلیکیشن MVC خود از آن استفاده کنیم.

×

## Connection info


**CONNECTIONSTRING**  


Database=IdentityMySQLDatabase;Data Source=us-cdb-azure-west-b.cleardb.com;User Id=root@us-cdb-azure-west-b.cleardb.com;Password=us-cdb-azure-west-b.cleardb.com;Persist Security Info=True;Server=us-cdb-azure-west-b.cleardb.com;SslMode=REQUIRED;TrustServerCertificate=True



**CONNECTIONURL**  

mysql://bc554a1b496531:a7277b9f@us-cdb-azure-west-b.cleardb.com:3306/IdentityMySQLDatabase





### ایجاد جداول ASP.NET Identity در یک دیتابیس MySQL

ابتدا ابزار MySQL Workbench را نصب کنید.

1. ابزار مذکور را [از اینجا](#) دانلود کنید.

2. هنگام نصب، گزینه **Setup Type: Custom** را انتخاب کنید.

3. در قسمت انتخاب قابلیت ها، گزینه های **Applications** و **MySQLWorkbench** را انتخاب کنید و مراحل نصب را به اتمام برسانید.

4. اپلیکیشن را اجرا کرده و روی **MySQLConnection** کلیک کنید تا رشته اتصال جدیدی تعریف کنید. رشته اتصالی که در مراحل

قبل از Azure MySQL Database کپی کردید را اینجا استفاده کنید. بعنوان مثال:

**Connection Name**

: AzureDB;

**Host Name**

: us-cdb-azure-west-b.cleardb.com;

Username

: <username>;

Password

: <password>;

Default Schema

: IdentityMySQLDatabase

5. پس از برقراری ارتباط با دیتابیس، یک برگ **Query** جدید باز کنید. فرامین زیر را برای ایجاد جداول مورد نیاز کپی کنید.

```
CREATE TABLE `IdentityMySQLDatabase`.`users` (
  `Id` VARCHAR(45) NOT NULL,
  `UserName` VARCHAR(45) NULL,
  `PasswordHash` VARCHAR(100) NULL,
  `SecurityStamp` VARCHAR(45) NULL,
  PRIMARY KEY (`id`));

CREATE TABLE `IdentityMySQLDatabase`.`roles` (
  `Id` VARCHAR(45) NOT NULL,
  `Name` VARCHAR(45) NULL,
  PRIMARY KEY (`Id`));

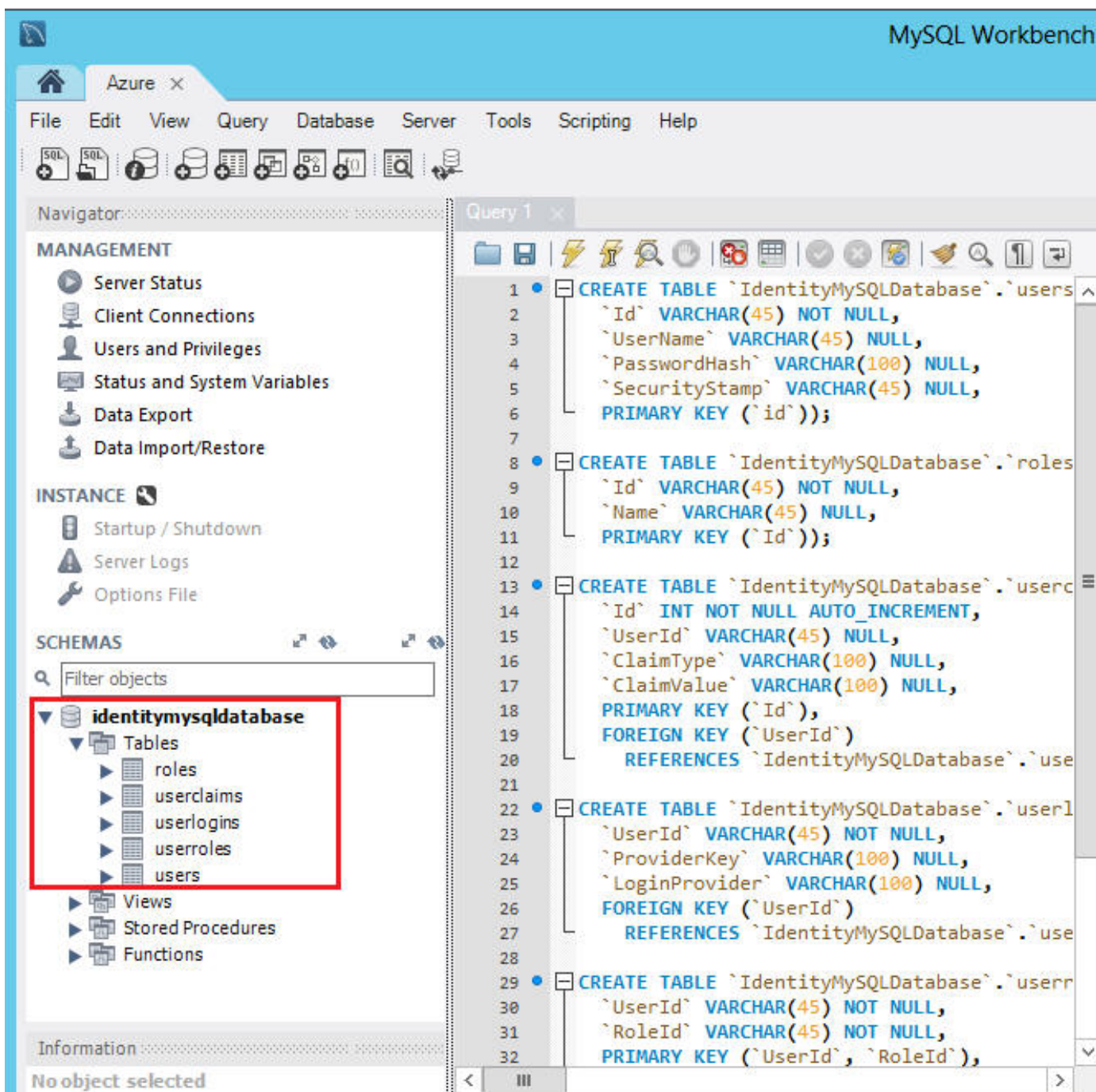
CREATE TABLE `IdentityMySQLDatabase`.`userclaims` (
  `Id` INT NOT NULL AUTO_INCREMENT,
  `UserId` VARCHAR(45) NULL,
  `ClaimType` VARCHAR(100) NULL,
  `ClaimValue` VARCHAR(100) NULL,
  PRIMARY KEY (`Id`),
  FOREIGN KEY (`UserId`)
    REFERENCES `IdentityMySQLDatabase`.`users` (`Id`) on delete cascade);

CREATE TABLE `IdentityMySQLDatabase`.`userlogins` (
  `UserId` VARCHAR(45) NOT NULL,
  `ProviderKey` VARCHAR(100) NULL,
  `LoginProvider` VARCHAR(100) NULL,
  FOREIGN KEY (`UserId`)
    REFERENCES `IdentityMySQLDatabase`.`users` (`Id`) on delete cascade);

CREATE TABLE `IdentityMySQLDatabase`.`userroles` (
  `UserId` VARCHAR(45) NOT NULL,
  `RoleId` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`UserId`, `RoleId`),
  FOREIGN KEY (`UserId`)
    REFERENCES `IdentityMySQLDatabase`.`users` (`Id`)
    on delete cascade
    on update cascade,
  FOREIGN KEY (`RoleId`)
    REFERENCES `IdentityMySQLDatabase`.`roles` (`Id`)
    on delete cascade
    on update cascade);
```

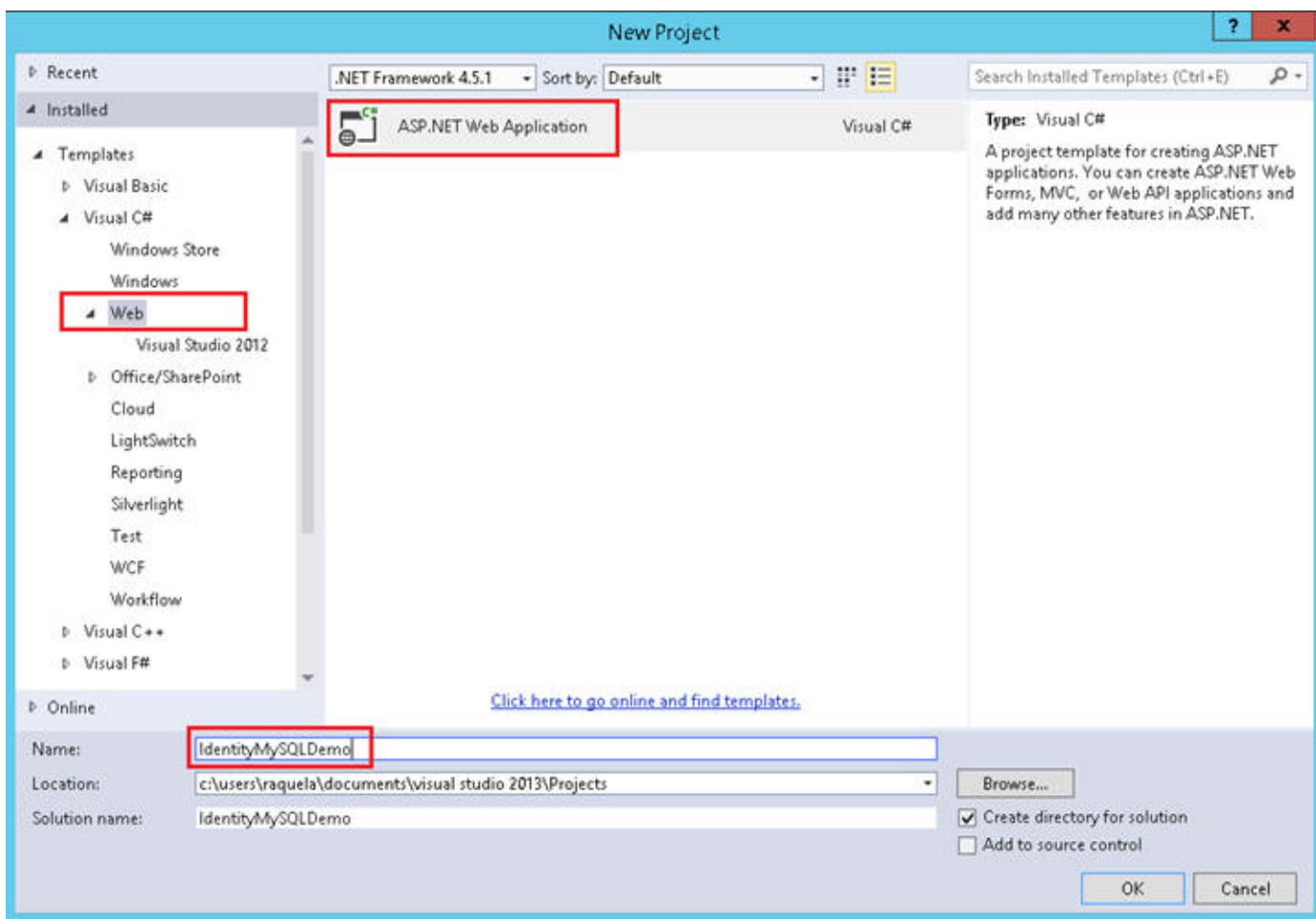
6. حالا تمام جداول لازم برای ASP.NET Identity را در اختیار دارید، دیتابیس ما MySQL است و روی Windows Azure میزبانی شده.



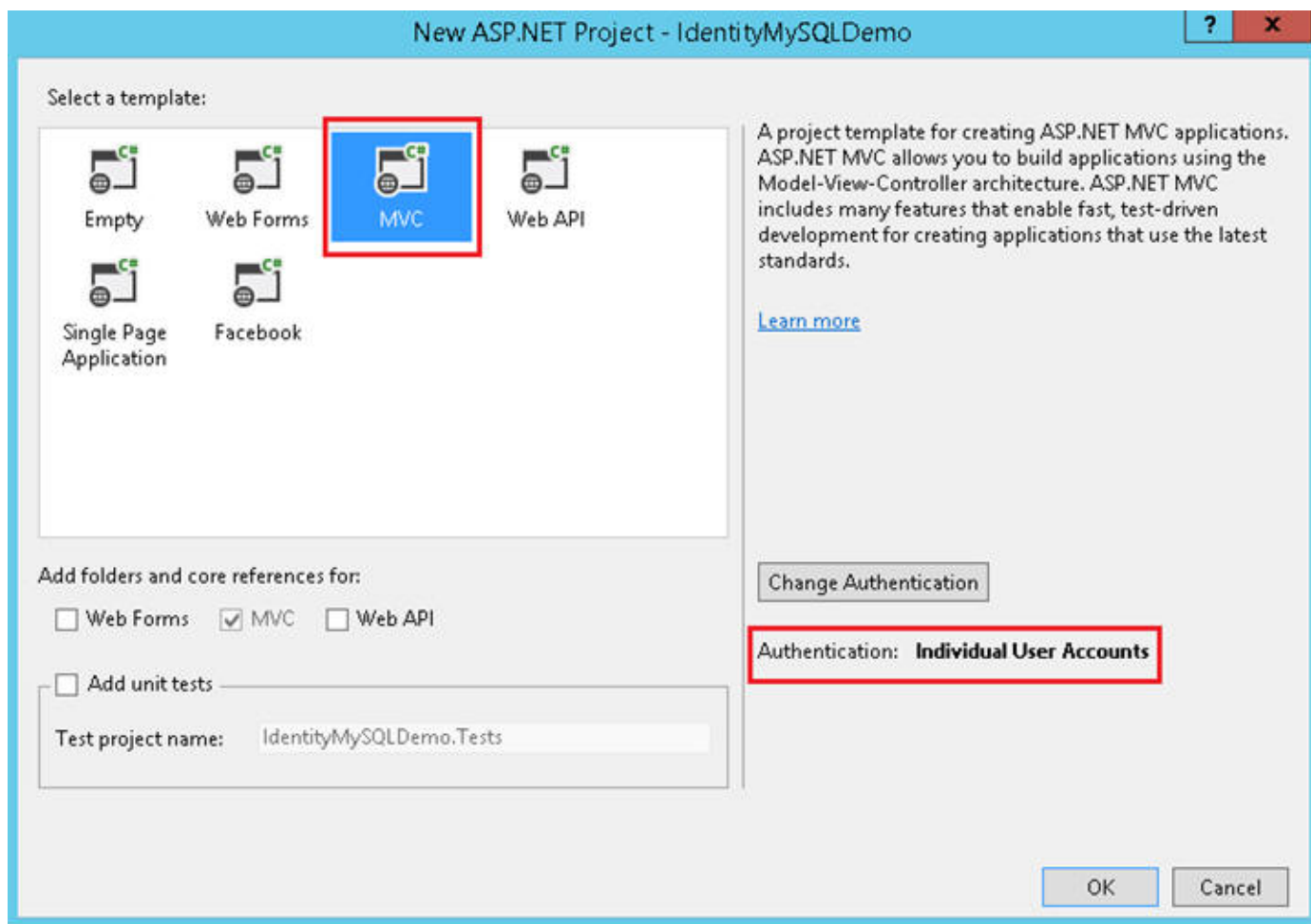


ایجاد یک اپلیکیشن ASP.NET MVC و پیگر بندی آن برای استفاده از MySQL Provider

1. به مخزن <https://github.com/raquelsa/AspNet.Identity.MySQL> بروید.
2. در گوشه سمت راست پایین صفحه روی دکمه Download Zip کلیک کنید تا کل پروژه را دریافت کنید.
3. محتوای فایل دریافتی را در یک پوشه محلی استخراج کنید.
4. پروژه AspNet.Identity.MySQL را باز کرده و آن را کامپایل (build) کنید.
5. روی نام پروژه کلیک راست کنید و گزینه Add, New Project را انتخاب نمایید. پروژه جدیدی از نوع ASP.NET Web Application بسازید و نام آن را به IdentityMySQLDemo تغییر دهید.



6. در پنجره New ASP.NET Project قالب MVC را انتخاب کنید و تنظیمات پیش فرض را بپذیرید.



7. در پنجره Solution Explorer روی پروژه IdentityMySQLDemo کلیک راست کرده و **Manage NuGet Packages** را انتخاب کنید. در قسمت جستجوی دیالوگ باز شده عبارت "Identity.EntityFramework" را وارد کنید. در لیست نتایج این پکیج را انتخاب کرده و آن را حذف (Uninstall) کنید. پیغامی مبنی بر حذف وابستگی‌ها باید دریافت کنید که مربوط به پکیج EntityFramework است، گزینه Yes را انتخاب کنید. از آنجا که کاری با پیاده سازی فرض نخواهیم داشت، این پکیج‌ها را حذف می‌کنیم.

8. روی پروژه IdentityMySQLDemo کلیک راست کرده و **Add, Reference, Solution, Projects** را انتخاب کنید. در دیالوگ باز شده پروژه **AspNet.Identity.MySQL** را انتخاب کرده و OK کنید.

9. در پروژه IdentityMySQLDemo پوشه Models را پیدا کرده و کلاس **IdentityModels.cs** را حذف کنید.

10. در پروژه IdentityMySQLDemo تمام ارجاعات "using Microsoft.AspNet.Identity.EntityFramework;" را با "using;" جایگزین کنید.

11. در پروژه IdentityMySQLDemo تمام ارجاعات به کلاس "ApplicationUser" را با "IdentityUser" جایگزین کنید.

12. کنترلر Account را باز کنید و متد سازنده آنرا مطابق لیست زیر تغییر دهید.

```
public AccountController() : this(new UserManager<IdentityUser>(new UserStore(new MySQLDatabase()))){
}
```

13. فایل web.config را باز کنید و رشته اتصال DefaultConnection را مطابق لیست زیر تغییر دهید.

```
<add name="DefaultConnection" connectionString="Database=IdentityMySQLDatabase;Data Source=<DataSource>;User Id=<UserID>;Password=<Password>" providerName="MySql.Data.MySqlClient" />
```

مقادیر <UserId>, <DataSource> و <Password> را با اطلاعات دیتابیس خود جایگزین کنید.

#### اجرای اپلیکیشن و اتصال به دیتابیس MySQL

1. روی پروژه IdentityMySQLDemo کلیک راست کرده و **Set as Startup Project** را انتخاب کنید.
2. اپلیکیشن را با Ctrl + F5 کامپایل و اجرا کنید.
3. در بالای صفحه روی **Register** کلیک کنید.
4. حساب کاربری جدیدی بسازید.

[Application name](#) [Home](#) [About](#) [Contact](#)

## Register.

Create a new account.

User name

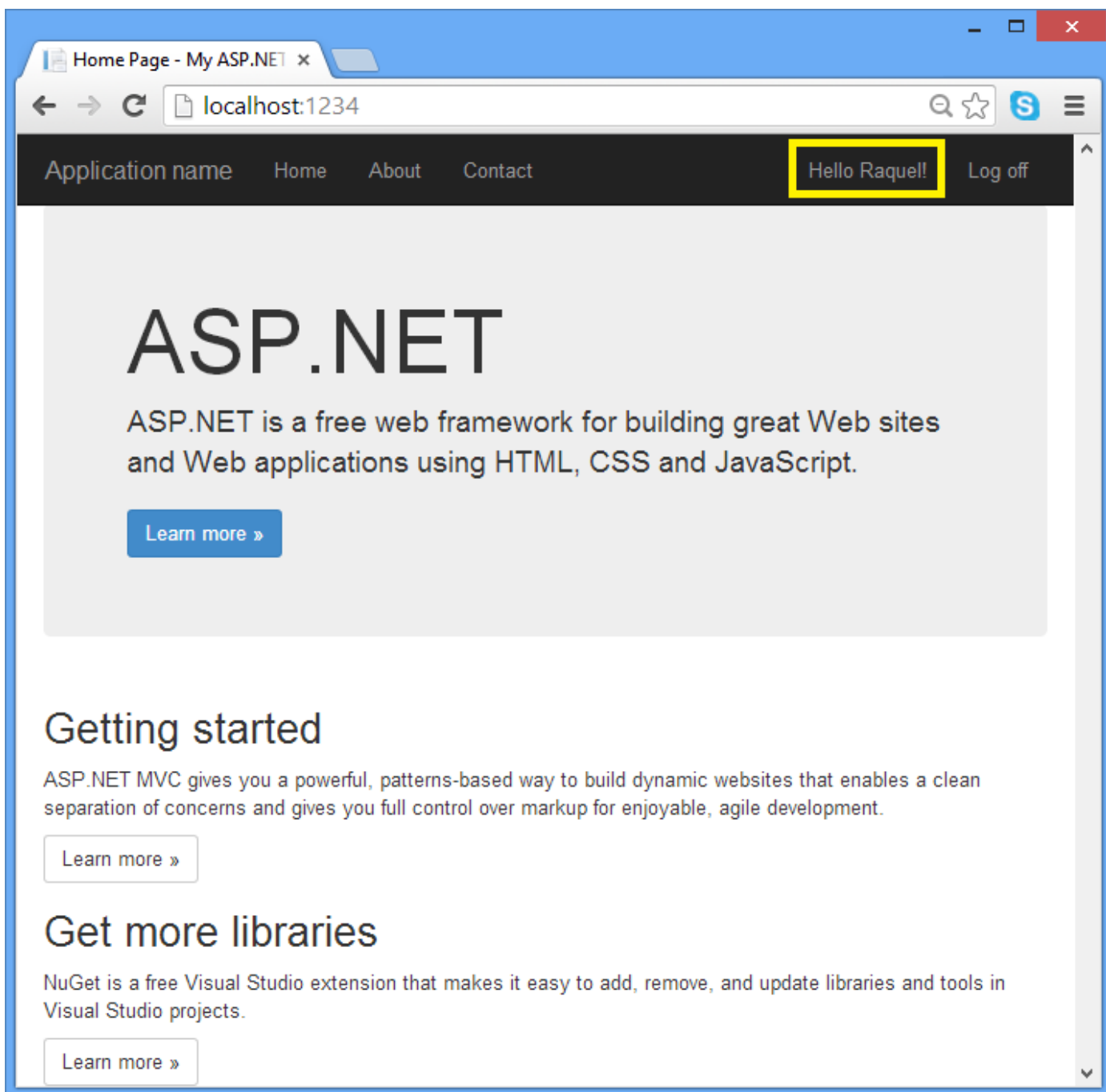
Password

Confirm password

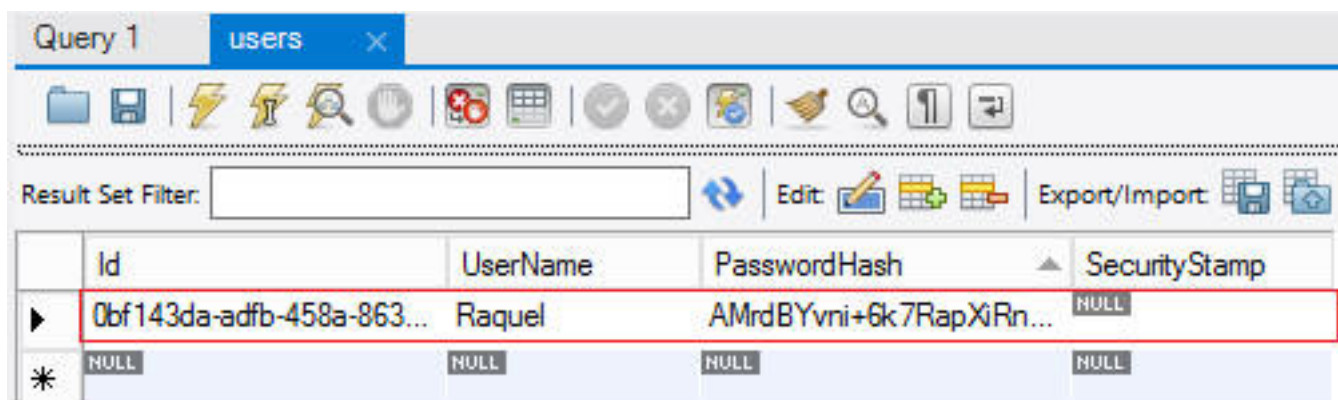
Register

© 2013 - My ASP.NET Application

5. در این مرحله کاربر جدید باید ایجاد شده و وارد سایت شود.



6. به ابزار MySQL Workbench بروید و محتوای جداول **IdentityMySQLDatabase** را بررسی کنید. جدول **users** را باز کنید و اطلاعات کاربر جدید را بررسی نمایید.



	Id	UserName	PasswordHash	SecurityStamp
►	0bf143da-adfb-458a-863...	Raquel	AMrdBYvni+6k7RapXiRn...	NULL
*	NULL	NULL	NULL	NULL

برای ساده نگاه داشتن این مقاله از بررسی تمام کدهای لازم خودداری شده، اما اگر مراحل را دنبال کنید و سورس کد نمونه را دریافت و بررسی کنید خواهید دید که پیاده سازی تامین کنندگان سفارشی برای ASP.NET Identity کار نسبتاً ساده ای است.