عنوان: **مهاجرت داده عضویت و پروفایل از Universal Providers به ASP.NET Identity**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۲۲ ۶:۵۵
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, MemberShip, Identity, ASP.NET Identity, Universal Providers

در این مقاله مهاجرت داده‌های سیستم عضویت، نقش‌ها و پروفایل‌های کاربران که توسط Universal Providers ساخته شده اند به مدل ASP.NET Identity را بررسی می‌کنیم. رویکردی که در این مقاله استفاده شده و قدم‌های لازمی که توضیح داده شده اند، برای اپلیکیشنی که با SQL Membership کار می‌کند هم می‌توانند کارساز باشند.
با انتشار Visual Studio 2013، تیم ASP.NET سیستم جدیدی با نام ASP.NET Identity معرفی کردند. می‌توانید در این لینک بیشتر درباره این انتشار بخوانید.
در ادامه مقاله قبلی تحت عنوان مهاجرت از SQL Membership به ASP.NET Identity ، در این پست به مهاجرت داده‌های یک اپلیکیشن که از مدل Providers برای مدیریت اطلاعات کاربران، نقش‌ها و پروفایل‌ها استفاده می‌کند به مدل جدید ASP.NET Identity می‌پردازیم. تمرکز این مقاله اساسا روی مهاجرت داده‌های پروفایل کاربران خواهد بود، تا بتوان به سرعت از آنها در اپلیکیشن استفاده کرد. مهاجرت داده‌های عضویت و نقش ها، شبیه پروسه مهاجرت SQL Membership است. رویکردی که در ادامه برای مهاجرت داده پروفایل‌ها دنبال شده است، می‌تواند برای اپلیکیشنی با SQL Membership نیز استفاده شود.
بعنوان یک مثال، با اپلیکیشن وبی شروع می‌کنیم که توسط Visual Studio 2012 ساخته شده و از مدل Providers استفاده می‌کند. پس از آن یک سری کد برای مدیریت پروفایل ها، ثبت نام کاربران، افزودن اطلاعات پروفایل به کاربران و مهاجرت الگوی دیتابیس می‌نویسیم و نهایتا اپلیکیشن را بروز رسانی می‌کنیم تا برای استفاده از سیستم Identity برای مدیریت کاربران و نقش‌ها آماده باشد. و بعنوان یک تست، کاربرانی که قبلا توسط Universal Providers ساخته شده اند باید بتوانند به سایت وارد شوند، و کاربران جدید هم باید قادر به ثبت نام در سایت باشند.
سورس کد کامل این مثال را می‌توانید از این لینک دریافت کنید.

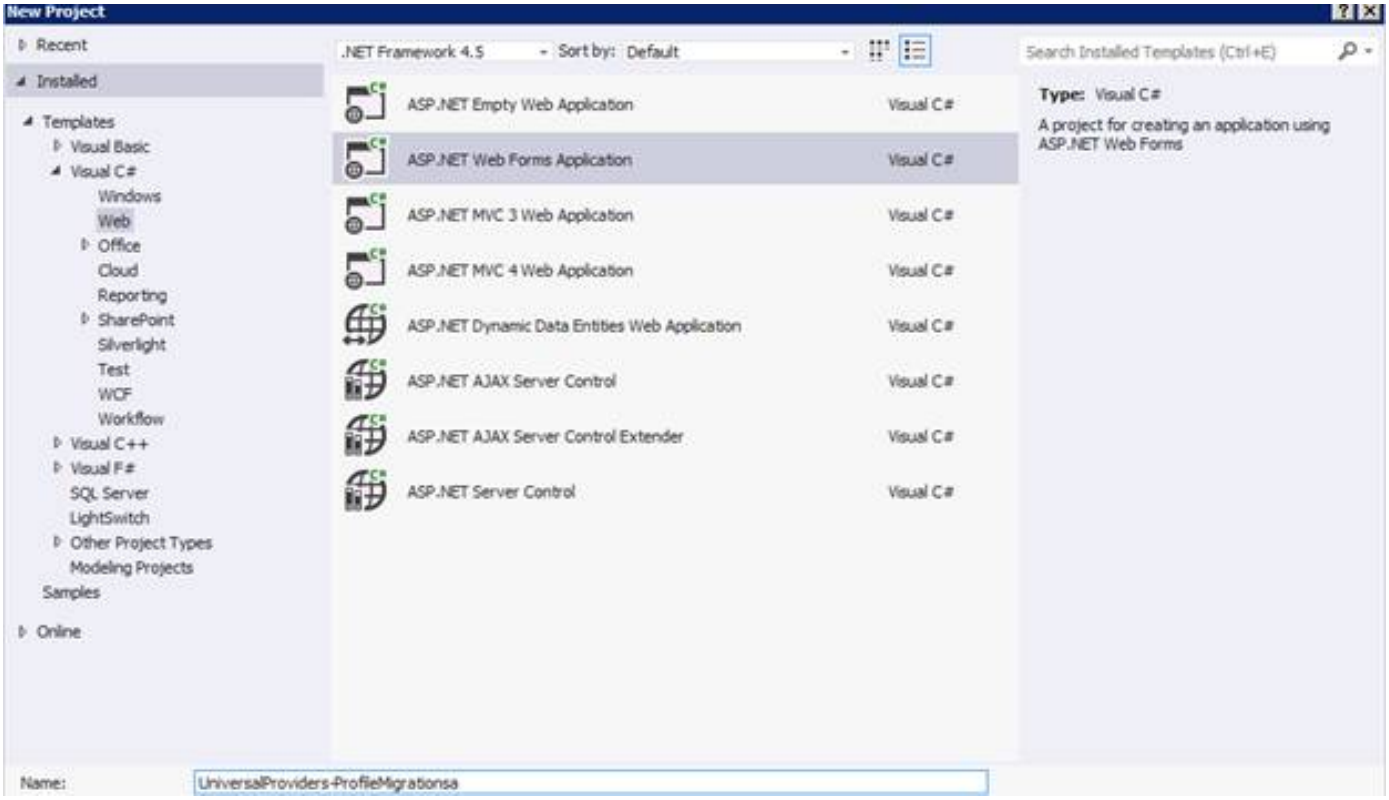
خلاصه مهاجرت داده پروفایل ها

قبل از آنکه با مهاجرت‌ها شروع کنیم، بگذارید تا نگاهی به تجربه مان از ذخیره اطلاعات پروفایل‌ها در مدل Providers بیاندازیم.
اطلاعات پروفایل کاربران یک اپلیکیشن به طرق مختلفی می‌تواند ذخیره شود. یکی از رایج‌ترین این راه ها، استفاده از تامین کننده‌های پیش فرضی است که بهمراه Universal Providers منتشر شدند. بدین منظور انجام مراحل زیر لازم است
کلاس جدیدی بسازید که دارای خواصی برای ذخیره اطلاعات پروفایل است.
کلاس جدیدی بسازید که از 'ProfileBase' ارث بری می‌کند و متدهای لازم برای دریافت پروفایل کاربران را پیاده سازی می‌کند.
استفاده از تامین کننده‌های پیش فرض را، در فایل web.config فعال کنید. و کلاسی که در مرحله 2 ساختید را بعنوان کلاس پیش فرض برای خواندن اطلاعات پروفایل معرفی کنید.

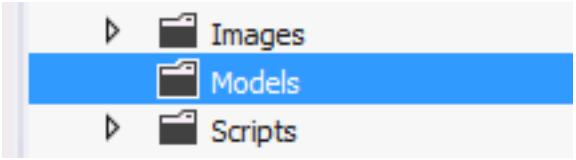
اطلاعات پروفایل‌ها بصورت xml serialized و binary در جدول 'Profiles' ذخیره می‌شوند.

پس از آنکه به سیستم ASP.NET Identity مهاجرت کردیم، اطلاعات پروفایل deserialized شده و در قالب خواص کلاس User ذخیره می‌شوند. هر خاصیت، بعدا می‌تواند به یک ستون در دیتابیس متصل شود. مزیت بدست آمده این است که مستقیما از کلاس User به اطلاعات پروفایل دسترسی داریم. ناگفته نماند که دیگر داده‌ها serialize/deserialize هم نمی‌شوند.

شروع به کار در Visual Studio 2012 پروژه جدیدی از نوع ASP.NET 4.5 Web Forms application بسازید. مثال جاری از یک قالب Web Forms استفاده می‌کند، اما می‌توانید از یک قالب MVC هم استفاده کنید.



پوشه جدیدی با نام 'Models' بسازید تا اطلاعات پروفایل را در آن قرار دهیم.



بعنوان یک مثال، بگذارید تا تاریخ تولد کاربر، شهر سکونت، قد و وزن او را در پروفایلش ذخیره کنیم. قد و وزن بصورت یک کلاس سفارشی (custom class) بنام 'PersonalStats' ذخیره می‌شوند. برای ذخیره و بازیابی پروفایل ها، به کلاسی احتیاج داریم که 'ProfileBase' را ارث بری می‌کند. پس کلاس جدیدی با نام 'AppProfile' بسازید.

```
public class ProfileInfo
{
    public ProfileInfo()
    {
        UserStats = new PersonalStats();
    }
    public DateTime? DateOfBirth { get; set; }
    public PersonalStats UserStats { get; set; }
    public string City { get; set; }
}

public class PersonalStats
{
    public int? Weight { get; set; }
    public int? Height { get; set; }
}

public class AppProfile : ProfileBase
{
    public ProfileInfo ProfileInfo
```

```
    {
        get { return (ProfileInfo)GetPropertyValue("ProfileInfo"); }
    }
    public static AppProfile GetProfile()
    {
        return (AppProfile)HttpContext.Current.Profile;
    }
    public static AppProfile GetProfile(string userName)
    {
        return (AppProfile)Create(userName);
    }
}
```

پروفایل را در فایل web.config خود فعال کنید. نام کلاسی را که در مرحله قبل ساختید، بعنوان کلاس پیش فرض برای ذخیره و بازیابی پروفایل‌ها معرفی کنید.

```
<profile defaultProvider="DefaultProfileProvider" enabled="true"
    inherits="UniversalProviders_ProfileMigrations.Models.AppProfile">
  <providers>
    .....
  </providers>
</profile>
```

برای دریافت اطلاعات پروفایل از کاربر، فرم وب جدیدی در پوشه Account بسازید و آنرا 'AddProfileData.aspx' نامگذاری کنید.

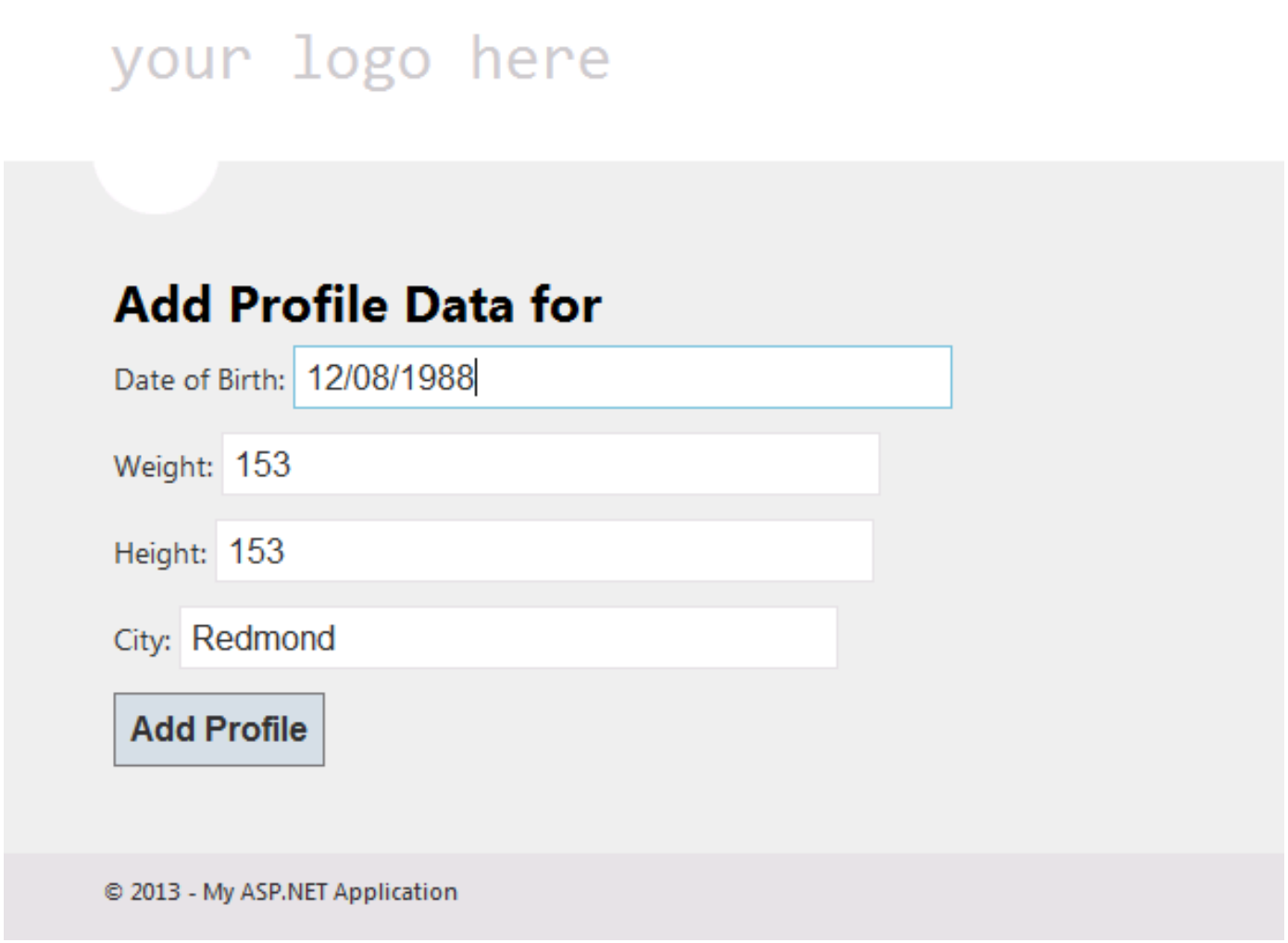
```
<h2> Add Profile Data for <%# User.Identity.Name %></h2>
<asp:Label Text="" ID="Result" runat="server" />
<div>
    Date of Birth:
    <asp:TextBox runat="server" ID="DateOfBirth"/>
</div>
<div>
    Weight:
    <asp:TextBox runat="server" ID="Weight"/>
</div>
<div>
    Height:
    <asp:TextBox runat="server" ID="Height"/>
</div>
<div>
    City:
    <asp:TextBox runat="server" ID="City"/>
</div>
<div>
    <asp:Button Text="Add Profile" ID="Add" OnClick="Add_Click" runat="server" />
</div>
```

کد زیر را هم به فایل code-behind اضافه کنید.

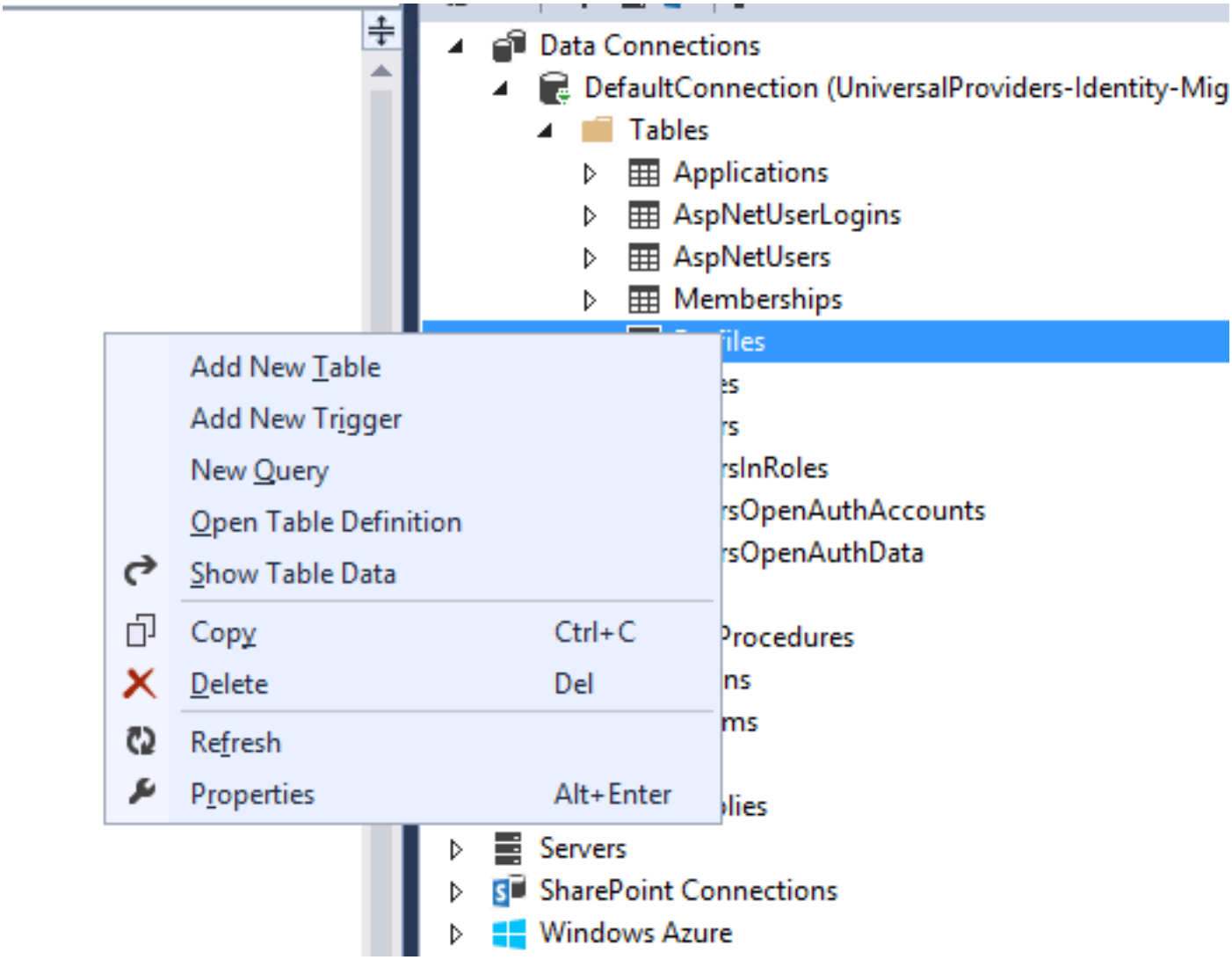
```
protected void Add_Click(object sender, EventArgs e)
{
    AppProfile profile = AppProfile.GetProfile(User.Identity.Name);
    profile.ProfileInfo.DateOfBirth = DateTime.Parse(DateOfBirth.Text);
    profile.ProfileInfo.UserStats.Weight = Int32.Parse(Weight.Text);
    profile.ProfileInfo.UserStats.Height = Int32.Parse(Height.Text);
    profile.ProfileInfo.City = City.Text;
    profile.Save();
}
```

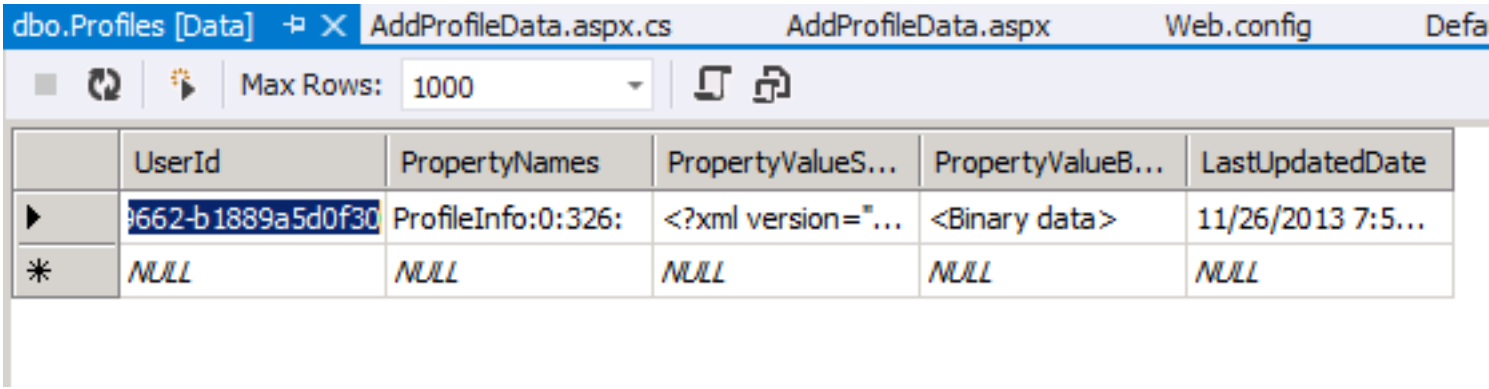
دقت کنید که فضای نامی که کلاس AppProfile در آن قرار دارد را وارد کرده باشید.

اپلیکیشن را اجرا کنید و کاربر جدیدی با نام ' **olduser** ' بسازید. به صفحه جدید 'AddProfileData' بروید و اطلاعات پروفایل کاربر را وارد کنید.



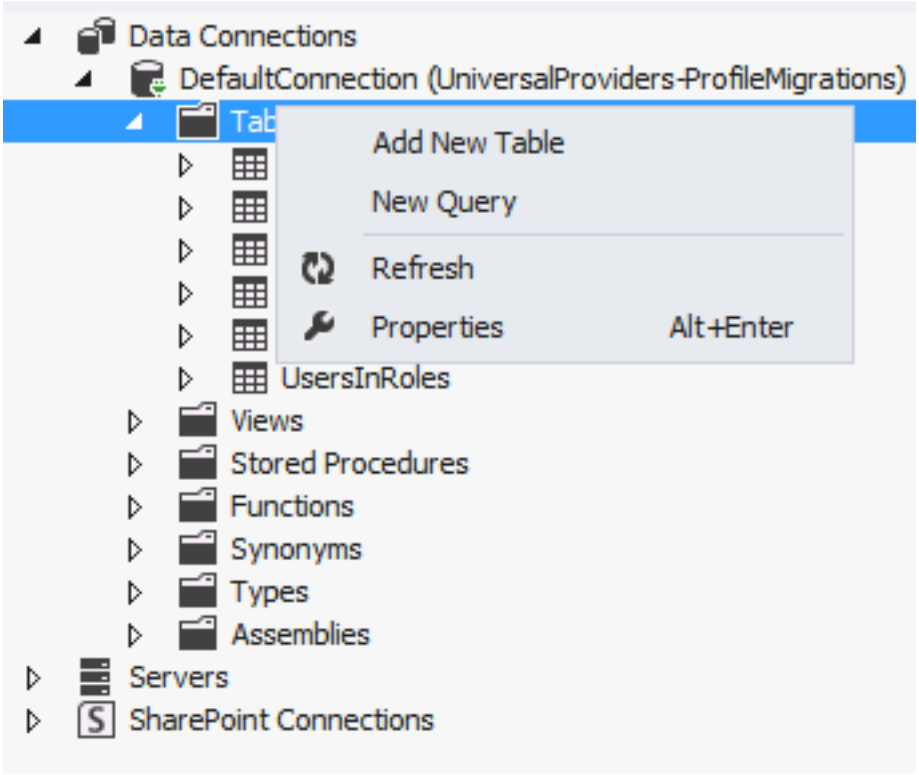
با استفاده از پنجره Server Explorer می‌توانید تایید کنید که اطلاعات پروفایل با فرمت xml در جدول 'Profiles' ذخیره می‌شوند.



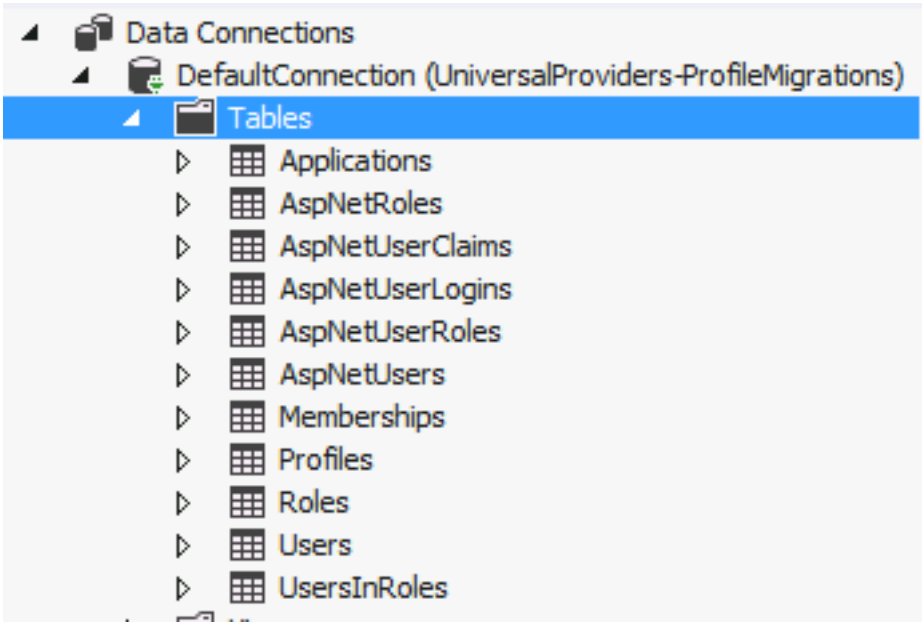


مهاجرت الگوی دیتابیس

برای اینکه دیتابیس فعلی بتواند با سیستم ASP.NET Identity کار کند، باید الگوی ASP.NET Identity را بروز رسانی کنیم تا فیلدهای جدیدی که اضافه کردیم را هم در نظر بگیرد. این کار می‌تواند توسط اسکریپت‌های SQL انجام شود، باید جداول جدیدی بسازیم و اطلاعات موجود را به آنها انتقال دهیم. در پنجره 'Server Explorer' گره 'DefaultConnection' را باز کنید تا جداول لیست شوند. روی Tables کلیک راست کنید و 'New Query' را انتخاب کنید.



اسکریپت مورد نیاز را از آدرس https://raw.github.com/suhasj/UniversalProviders-Identity-Migrations/master/Migration.txt دریافت کرده و آن را اجرا کنید. اگر اتصال خود به دیتابیس را تازه کنید خواهید دید که جداول جدیدی اضافه شده اند. می‌توانید داده‌های این جداول را بررسی کنید تا ببینید چگونه اطلاعات منتقل شده اند.



مهاجرت اپلیکیشن برای استفاده از ASP.NET Identity

پکیج‌های مورد نیاز برای ASP.NET Identity را نصب کنید:

Microsoft.AspNet.Identity.EntityFramework

Microsoft.AspNet.Identity.Owin

Microsoft.Owin.Host.SystemWeb

Microsoft.Owin.Security.Facebook

Microsoft.Owin.Security.Google

Microsoft.Owin.Security.MicrosoftAccount

Microsoft.Owin.Security.Twitter

اطلاعات بیشتری درباره مدیریت پکیج‌های NuGet از اینجا قابل دسترسی هستند.
برای اینکه بتوانیم از الگوی جاری دیتابیس استفاده کنیم، ابتدا باید مدل‌های لازم ASP.NET Identity را تعریف کنیم تا موجودیت‌های دیتابیس را Map کنیم. طبق قرارداد سیستم Identity کلاس‌های مدل یا باید اینترفیس‌های تعریف شده در Identity.Core dll را پیاده سازی کنند، یا می‌توانند پیاده سازی‌های پیش فرضی را که در Microsoft.AspNet.Identity.EntityFramework وجود دارند گسترش دهند. ما برای نقش ها، اطلاعات ورود کاربران و claimها از پیاده سازی‌های پیش فرض استفاده خواهیم کرد. نیاز به استفاده از یک کلاس سفارشی User داریم. پوشه جدیدی در پروژه با نام 'IdentityModels' بسازید. کلاسی با نام 'User' در این پوشه بسازید و کد آن را با لیست زیر تطابق دهید.

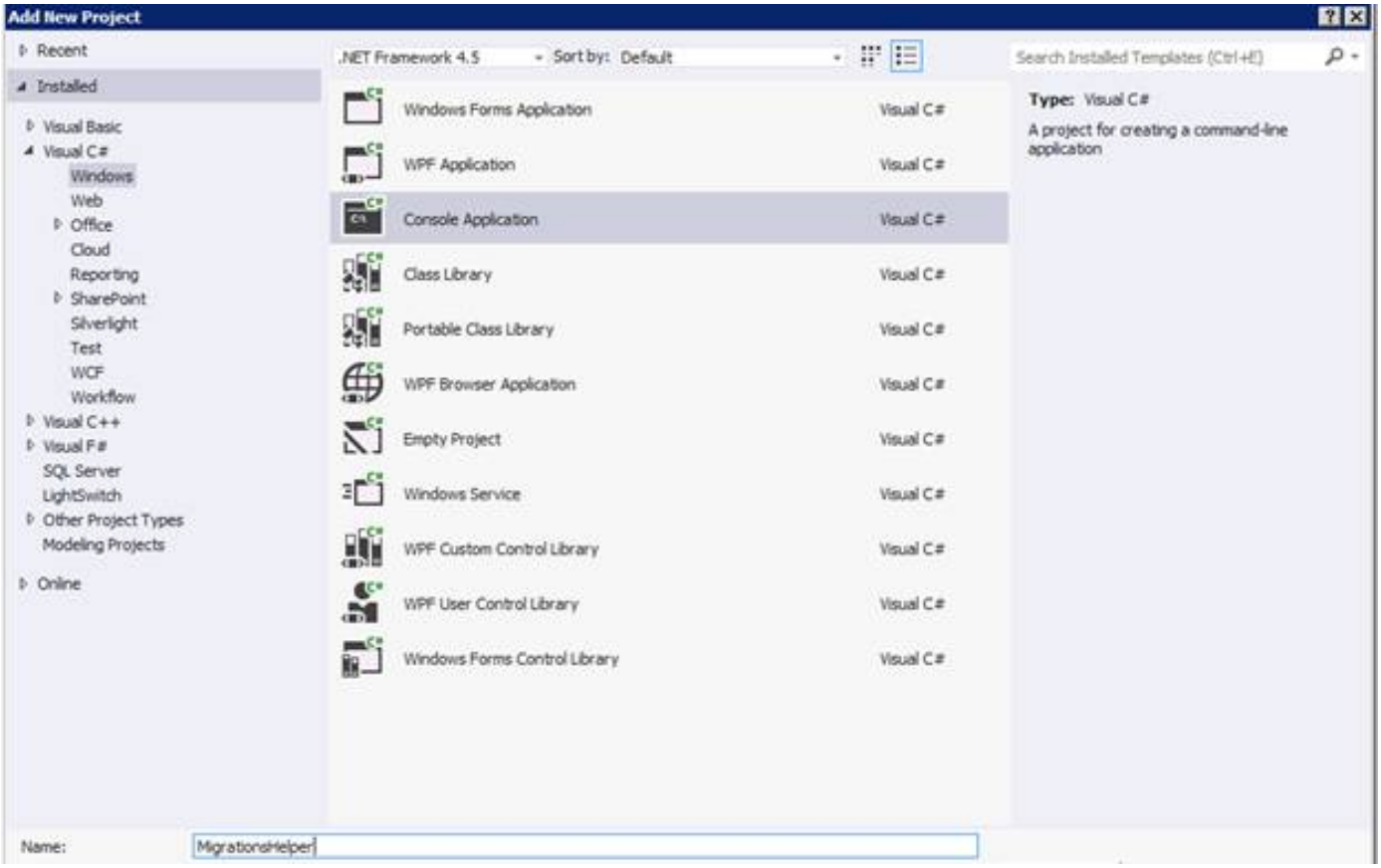
```
using Microsoft.AspNet.Identity.EntityFramework;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using UniversalProviders_ProfileMigrations.Models;

namespace UniversalProviders_Identity_Migrations
{
    public class User : IdentityUser
    {
        public User()
        {
            CreateDate = DateTime.UtcNow;
            IsApproved = false;
            LastLoginDate = DateTime.UtcNow;
            LastActivityDate = DateTime.UtcNow;
            LastPasswordChangedDate = DateTime.UtcNow;
            Profile = new ProfileInfo();
        }

        public System.Guid ApplicationId { get; set; }
        public bool IsAnonymous { get; set; }
        public System.DateTime? LastActivityDate { get; set; }
        public string Email { get; set; }
        public string PasswordQuestion { get; set; }
        public string PasswordAnswer { get; set; }
        public bool IsApproved { get; set; }
        public bool IsLockedOut { get; set; }
        public System.DateTime? CreateDate { get; set; }
        public System.DateTime? LastLoginDate { get; set; }
        public System.DateTime? LastPasswordChangedDate { get; set; }
        public System.DateTime? LastLockoutDate { get; set; }
        public int FailedPasswordAttemptCount { get; set; }
        public System.DateTime? FailedPasswordAttemptWindowStart { get; set; }
        public int FailedPasswordAnswerAttemptCount { get; set; }
        public System.DateTime? FailedPasswordAnswerAttemptWindowStart { get; set; }
        public string Comment { get; set; }
        public ProfileInfo Profile { get; set; }
    }
}
```

دقت کنید که 'ProfileInfo' حالا بعنوان یک خاصیت روی کلاس User تعریف شده است. بنابراین می‌توانیم مستقیما از کلاس کاربر با اطلاعات پروفایل کار کنیم.
محتویات پوشه‌های **IdentityModels** و **IdentityAccount** را از آدرس https://github.com/suhasj/UniversalProviders-Identity-Migrations/tree/master/UniversalProviders-Identity-Migrations دریافت و کپی کنید. این فایل‌ها مابقی مدل ها، و صفحاتی برای مدیریت کاربران و نقش‌ها در سیستم جدید ASP.NET Identity هستند.

انتقال داده پروفایل‌ها به جداول جدید

همانطور که گفته شد ابتدا باید داده‌های پروفایل را deserialize کرده و از فرمت xml خارج کنیم، سپس آنها را در ستون‌های جدول AspNetUsers ذخیره کنیم. ستون‌های جدید در مرحله قبل به دیتابیس اضافه شدند، پس تنها کاری که باقی مانده پر کردن این ستون‌ها با داده‌های ضروری است. بدین منظور ما از یک اپلیکیشن کنسول استفاده می‌کنیم که تنها یک بار اجرا خواهد شد، و ستون‌های جدید را با داده‌های لازم پر می‌کند.

در solution جاری یک پروژه اپلیکیشن کنسول بسازید.



آخرین نسخه پکیج Entity Framework را نصب کنید. همچنین یک رفرنس به اپلیکیشن وب پروژه بدهید (کلیک راست روی پروژه و گزینه 'Add Reference').

کد زیر را در کلاس Program.cs وارد کنید. این قطعه کد پروفایل تک تک کاربران را می‌خواند و در قالب 'ProfileInfo' آنها را serialize می‌کند و در دیتابیس ذخیره می‌کند.

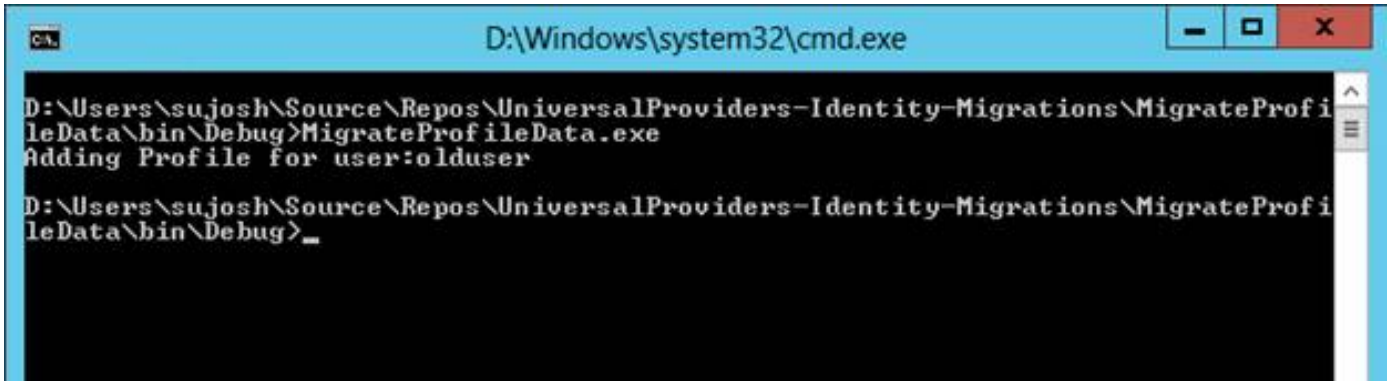
```
public class Program
{
    var dbContext = new ApplicationDbContext();
    foreach (var profile in dbContext.Profiles)
    {
        var stringId = profile.UserId.ToString();
        var user = dbContext.Users.Where(x => x.Id == stringId).FirstOrDefault();
        Console.WriteLine("Adding Profile for user:" + user.UserName);
        var serializer = new XmlSerializer(typeof(ProfileInfo));
        var stringReader = new StringReader(profile.PropertyValueStrings);
        var profileData = serializer.Deserialize(stringReader) as ProfileInfo;
        if (profileData == null)
        {
            Console.WriteLine("Profile data deserialization error for user:" + user.UserName);
        }
        else
```

```
        {
            user.Profile = profileData;
        }
    }
    dbContext.SaveChanges();
}
```

برخی از مدل‌های استفاده شده در پوشه 'IdentityModels' تعریف شده اند که در پروژه اپلیکیشن وبمان قرار دارند، بنابراین افزودن فضاهای نام مورد نیاز فراموش نشود.

کد بالا روی دیتابیسی که در پوشه App_Data وجود دارد کار می‌کند، این دیتابیس در مراحل قبلی در اپلیکیشن وب پروژه ایجاد شد. برای اینکه این دیتابیس را رفرنس کنیم باید رشته اتصال فایل app.config اپلیکیشن کنسول را بروز رسانی کنید. از همان رشته اتصال web.config در اپلیکیشن وب پروژه استفاده کنید. همچنین آدرس فیزیکی کامل را در خاصیت 'AttachDbFilename' وارد کنید.

یک Command Prompt باز کنید و به پوشه bin اپلیکیشن کنسول بالا بروید. فایل اجرایی را اجرا کنید و نتیجه را مانند تصویر زیر بررسی کنید.



در پنجره Server Explorer جدول 'AspNetUsers' را باز کنید. حال ستون‌های این جدول باید خواص کلاس مدل را منعکس کنند.

کارایی سیستم را تایید کنید

با استفاده از صفحات جدیدی که برای کار با ASP.NET Identity پیاده سازی شده اند سیستم را تست کنید. با کاربران قدیمی که در دیتابیس قبلی وجود دارند وارد شوید. کاربران باید با همان اطلاعات پیشین بتوانند وارد سیستم شوند. مابقی قابلیت‌ها را هم بررسی کنید. مثلا افزودن OAuth، ثبت کاربر جدید، تغییر کلمه عبور، افزودن نقش ها، تخصیص کاربران به نقش‌ها و غیره.
داده‌های پروفایل کاربران قدیمی و جدید همگی باید در جدول کاربران ذخیره شده و بازیابی شوند. جدول قبلی دیگر نباید رفرنس شود.