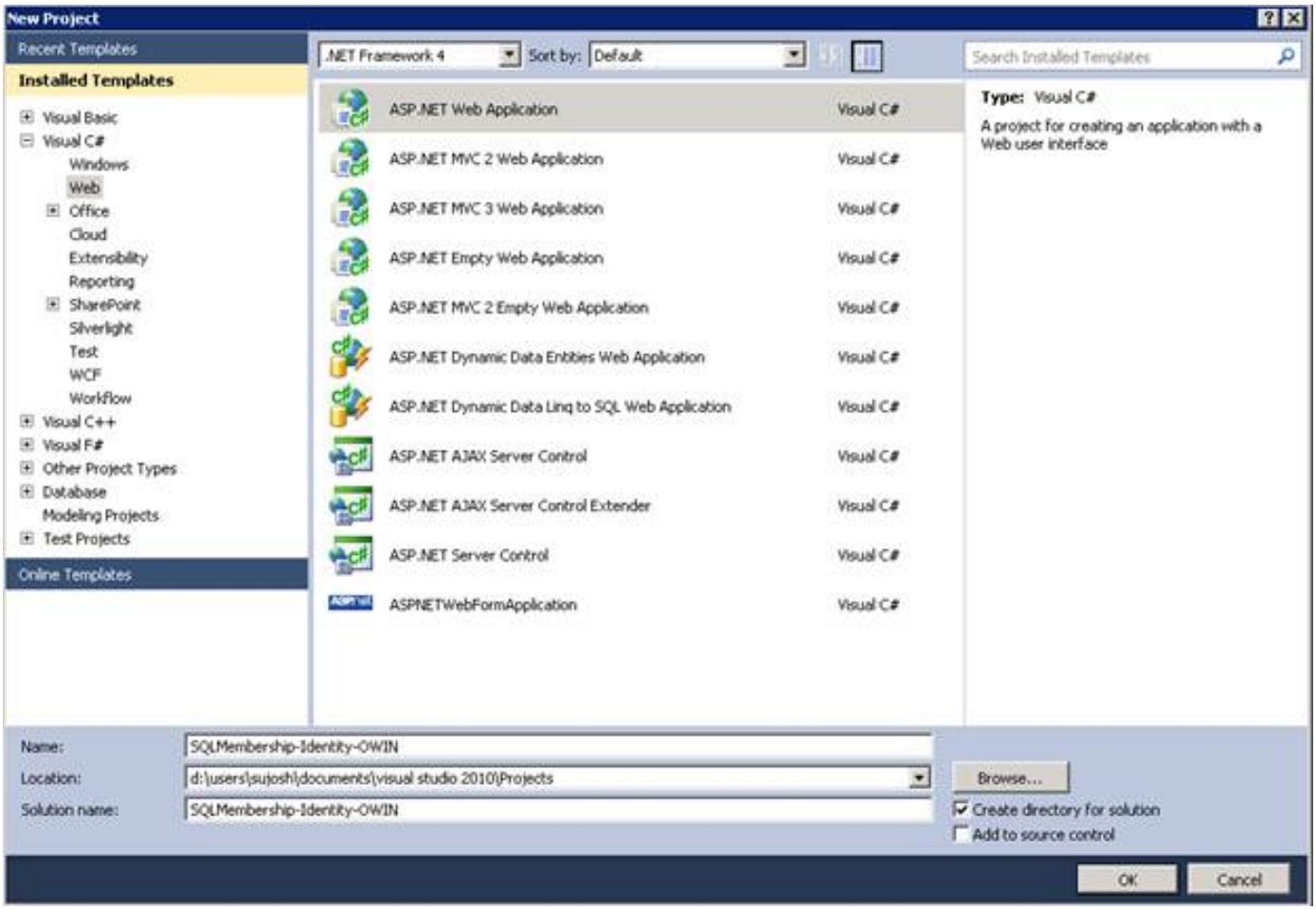
عنوان: **مهاجرت از ASP.NET Identity به SQL Membership**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۲۱ ۱۸:۳۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, Security, MemberShip, ASP.NET Identity

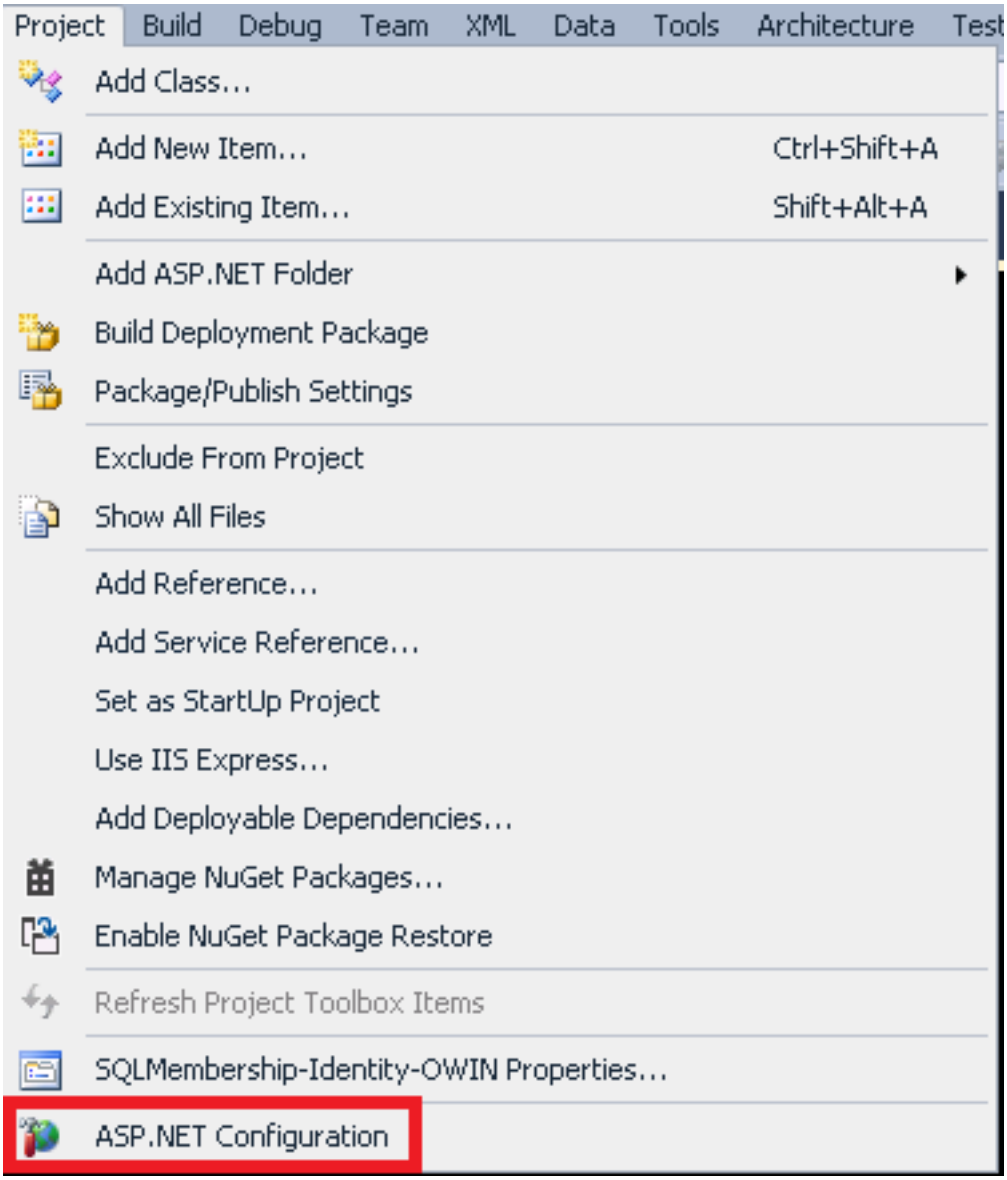
در این مقاله مهاجرت یک اپلیکیشن وب که توسط SQL Membership ساخته شده است را به سیستم جدید ASP.NET Identity بررسی می‌کنیم. برای این مقاله از یک قالب اپلیکیشن وب (Web Forms) که توسط Visual Studio 2010 ساخته شده است برای ساختن کاربران و نقش‌ها استفاده می‌کنیم. سپس با استفاده از یک SQL Script دیتابیس موجود را به دیتابیسی که ASP.NET Identity نیاز دارد تبدیل می‌کنیم. در قدم بعدی پکیج‌های مورد نیاز را به پروژه اضافه می‌کنیم و صفحات جدیدی برای مدیریت حساب‌های کاربری خواهیم ساخت. بعنوان یک تست، کاربران قدیمی که توسط SQL Membership ساخته شده بودند باید قادر باشند به سایت وارد شوند. همچنین کاربران جدید باید بتوانند بدون هیچ مشکلی در سیستم ثبت نام کنند. سورس کد کامل این مقاله را می‌توانید از این لینک دریافت کنید.

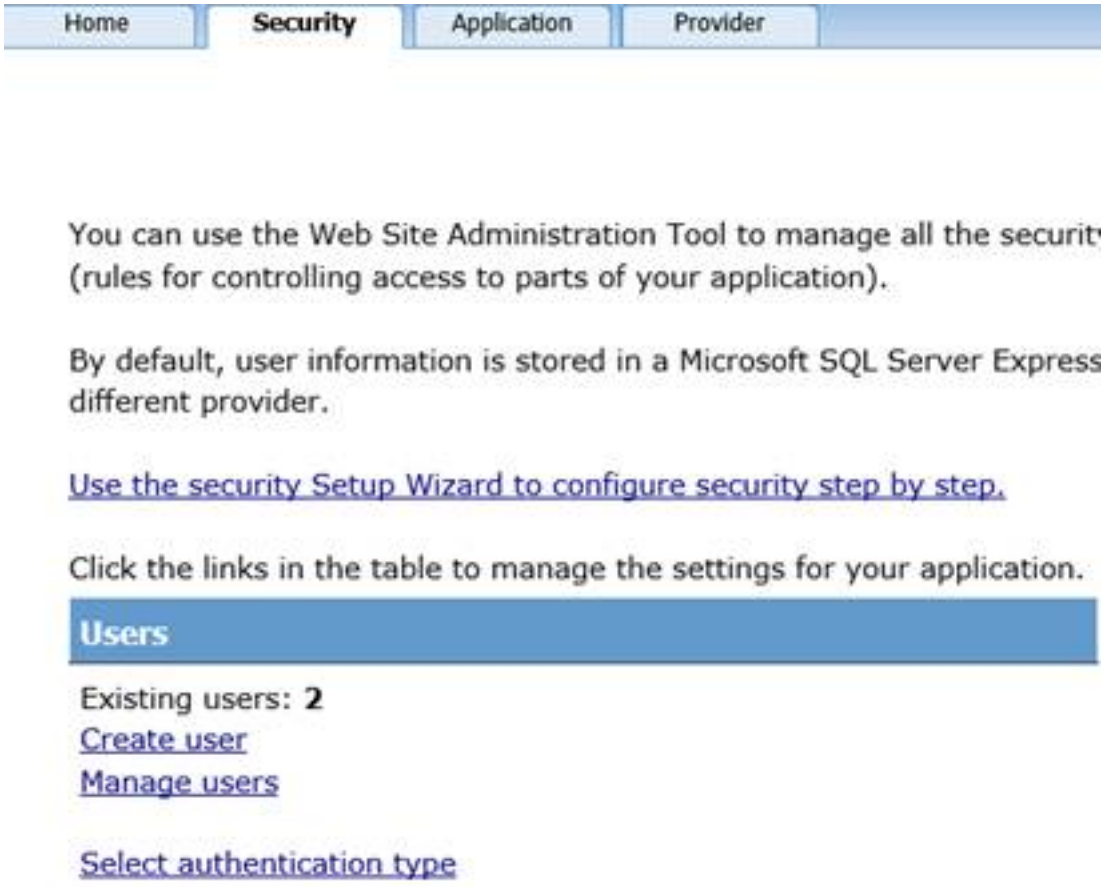
یک اپلیکیشن با SQL Membership بسازید

برای شروع به اپلیکیشنی نیاز داریم که از SQL Membership استفاده می‌کند و دارای داده هایی از کاربران و نقش‌ها است. برای این مقاله، بگذارید پروژه جدیدی توسط VS 2010 بسازیم.

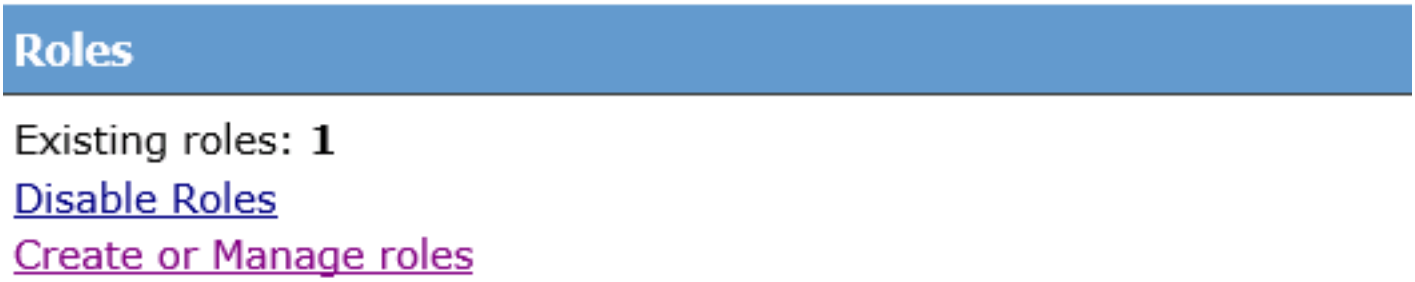


حال با استفاده از ابزار ASP.NET Configuration دو کاربر جدید بسازید: **oldAdminUser** و **oldUser** .

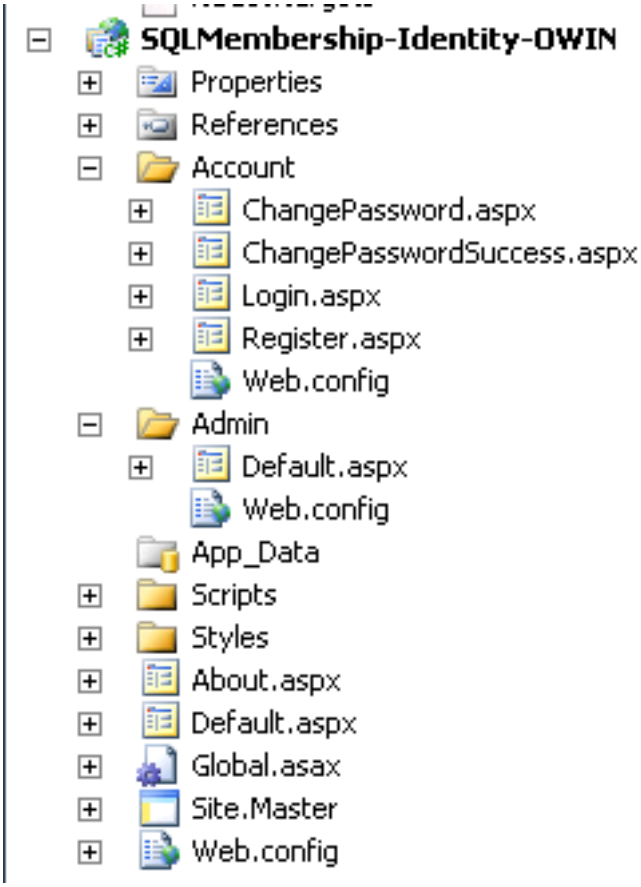

Project
Build
Debug
Team
XML
Data
Tools
Architecture
Test
Add Class...
Add New Item... Ctrl+Shift+A
Add Existing Item... Shift+Alt+A
Add ASP.NET Folder
Build Deployment Package
Package/Publish Settings
Exclude From Project
Show All Files
Add Reference...
Add Service Reference...
Set as StartUp Project
Use IIS Express...
Add Deployable Dependencies...
Manage NuGet Packages...
Enable NuGet Package Restore
Refresh Project Toolbox Items
SQLMembership-Identity-OWIN Properties...
ASP.NET Configuration



نقش جدیدی با نام Admin بسازید و کاربر oldAdminUser را به آن اضافه کنید.

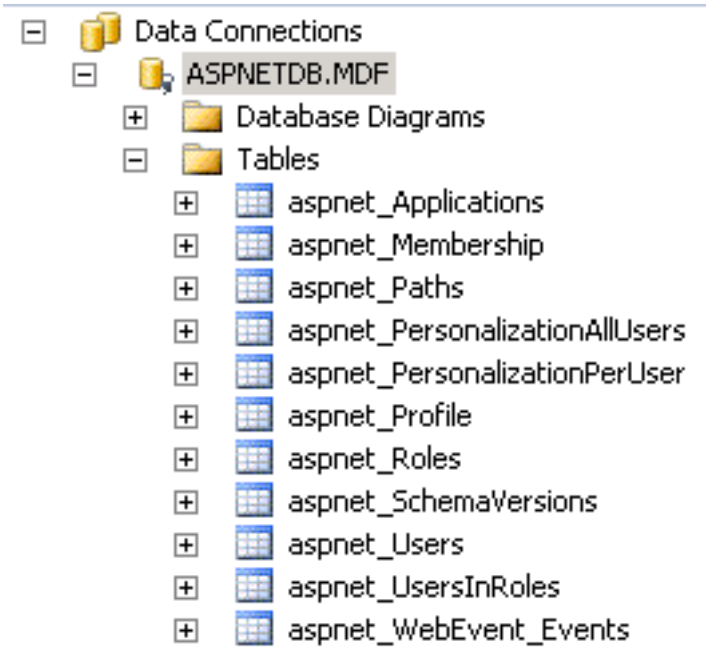


بخش جدیدی با نام Admin در سایت خود بسازید و فرمی بنام Default.aspx به آن اضافه کنید. همچنین فایل web.config این قسمت را طوری پیکربندی کنید تا تنها کاربرانی که در نقش Admin هستند به آن دسترسی داشته باشند. برای اطلاعات بیشتر به این لینک مراجعه کنید.



پنجره Server Explorer را باز کنید و جداول ساخته شده توسط SQL Membership را بررسی کنید. اطلاعات اصلی کاربران که برای ورود به سایت استفاده می‌شوند، در جداول **aspnet_Users** و **aspnet_Membership** ذخیره می‌شوند. داده‌های مربوط به نقش‌ها نیز در جدول **aspnet_Roles** ذخیره خواهند شد. رابطه بین کاربران و نقش‌ها نیز در جدول **aspnet_UsersInRoles** ذخیره می‌شود، یعنی اینکه هر کاربری به چه نقش هایی تعلق دارد.

برای مدیریت اساسی سیستم عضویت، مهاجرت جداول ذکر شده به سیستم جدید ASP.NET Identity کفایت می‌کند.



مهاجرت به Visual Studio 2013

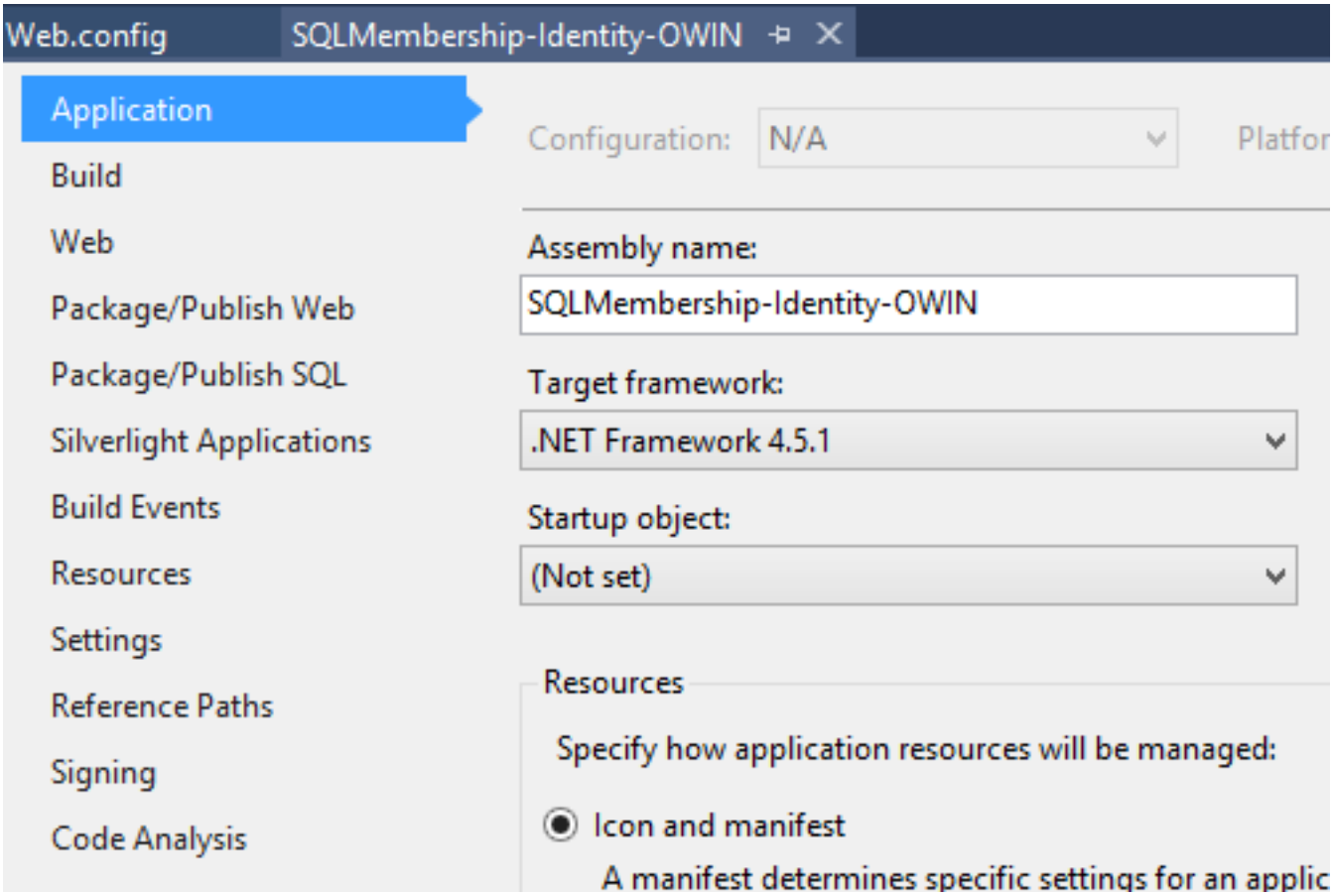
برای شروع ابتدا Visual Studio Express 2013 for Web یا Visual Studio 2013 را نصب کنید.

حال پروژه ایجاد شده را در نسخه جدید ویژوال استودیو باز کنید. اگر نسخه ای از SQL Server Express را روی سیستم خود نصب نکرده باشید، هنگام باز کردن پروژه پیغامی به شما نشان داده می‌شود. دلیل آن وجود رشته اتصالی است که از SQL Server Express استفاده می‌کند. برای رفع این مساله می‌توانید SQL Express را نصب کنید، و یا رشته اتصال را طوری تغییر دهید که از LocalDB استفاده کند.

فایل web.config را باز کرده و رشته اتصال را مانند تصویر زیر ویرایش کنید.

```
<configuration>
  <connectionStrings>
    <add name="ApplicationServices"
         connectionString="data source=(LocalDb)\v11.0;Integrated Security=SSPI;AttachDBFilename=|DataDirectory|\aspnetdb.mdf"
         providerName="System.Data.SqlClient" />
  </connectionStrings>
```

پنجره Server Explorer را باز کنید و مطمئن شوید که الگوی جداول و داده‌ها قابل رویت هستند.
سیستم ASP.NET Identity با نسخه 4.5 دات نت فریم ورک و بالاتر سازگار است. پس نسخه فریم ورک پروژه را به آخرین نسخه (4.5.1) تغییر دهید.



پروژه را Build کنید تا مطمئن شوید هیچ خطایی وجود ندارد.

نصب پکیج‌های NuGet

در پنجره Solution Explorer روی نام پروژه خود کلیک راست کرده، و گزینه Manage NuGet Packages را انتخاب کنید. در قسمت جستجوی دیالوگ باز شده، عبارت "Microsoft.AspNet.Identity.EntityFramework" را وارد کنید. این پکیج را در لیست نتایج انتخاب کرده و آن را نصب کنید. نصب این بسته، نیازمندهای موجود را بصورت خودکار دانلود و نصب می‌کند:
EntityFramework و **ASP.NET Identity Core** . حال پکیج‌های زیر را هم نصب کنید (اگر نمی‌خواهید OAuth را فعال کنید، 4 پکیج آخر را نادیده بگیرید).

Microsoft.AspNet.Identity.Owin

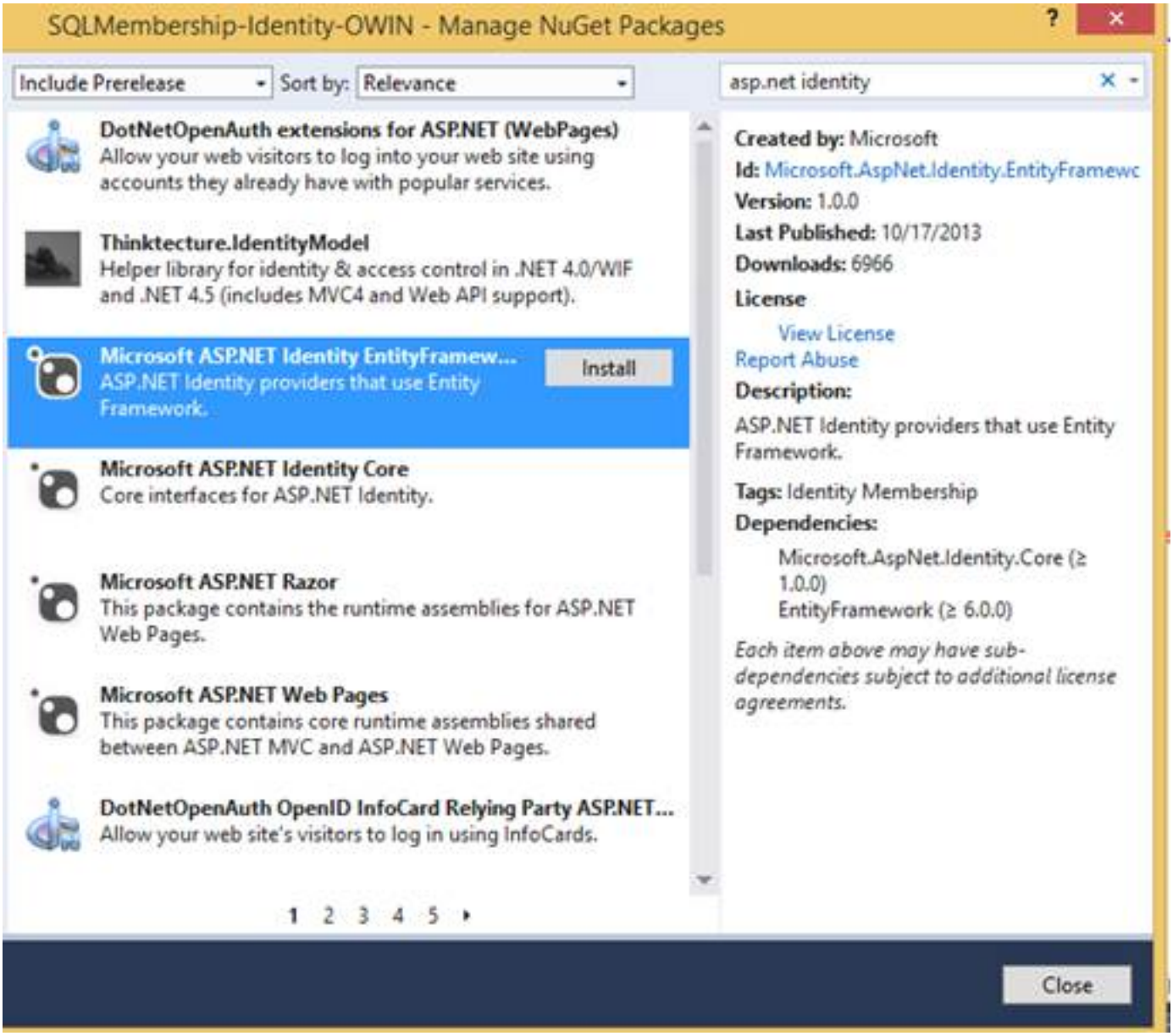
Microsoft.Owin.Host.SystemWeb

Microsoft.Owin.Security.Facebook

Microsoft.Owin.Security.Google

Microsoft.Owin.Security.MicrosoftAccount

Microsoft.Owin.Security.Twitter

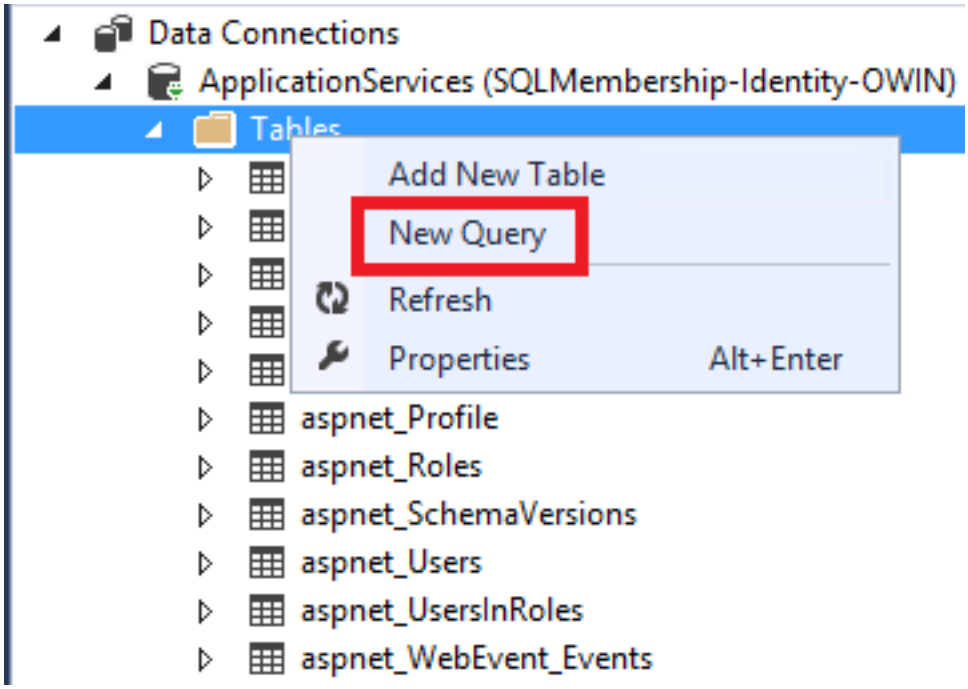


مهاجرت دیتابیس فعلی به سیستم ASP.NET Identity

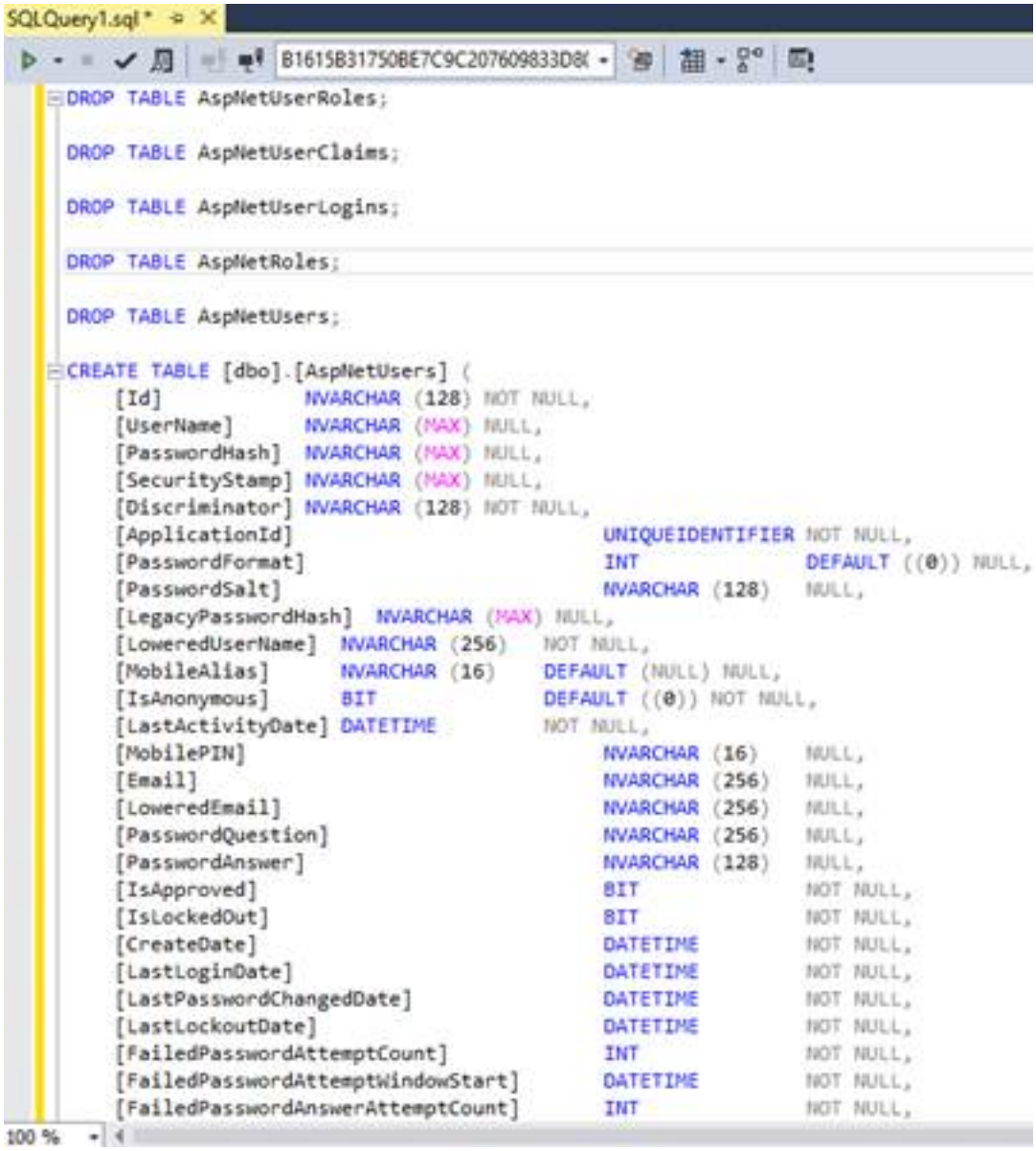
قدم بعدی مهاجرت دیتابیس فعلی به الگویی است، که سیستم ASP.NET Identity به آن نیاز دارد. بدین منظور ما یک اسکریپت SQL را اجرا می‌کنیم تا جداول جدیدی بسازد و اطلاعات کاربران را به آنها انتقال دهد. فایل این اسکریپت را می‌توانید از لینک https://github.com/suhasj/SQLMembership-Identity-OWIN دریافت کنید.

این اسکریپت مختص این مقاله است. اگر الگوی استفاده شده برای جداول سیستم عضویت شما ویرایش/سفارشی-سازی شده باید این اسکریپت را هم بر اساس این تغییرات بروز رسانی کنید.

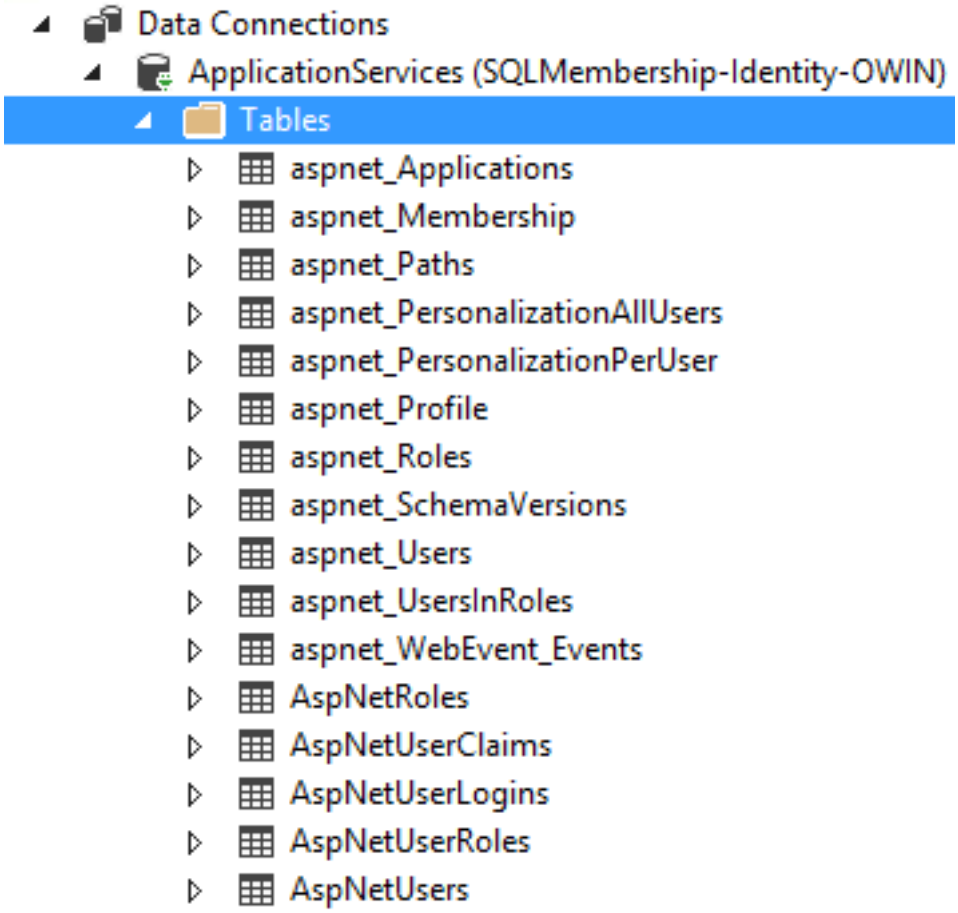
پنجره Server Explorer را باز کنید. گره اتصال ApplicationServices را باز کنید تا جداول را مشاهده کنید. روی گره Tables کلیک راست کرده و گزینه New Query را انتخاب کنید.



در پنجره کوئری باز شده، تمام محتویات فایل *Migrations.sql* را کپی کنید. سپس اسکریپت را با کلیک کردن دکمه Execute اجرا کنید.



ممکن است با اخطاری مواجه شوید مبنی بر آنکه امکان حذف (drop) بعضی از جداول وجود نداشت. دلیلش آن است که چهار عبارت اولیه در این اسکریپت، تمام جداول مربوط به Identity را در صورت وجود حذف می‌کنند. از آنجا که با اجرای اولیه این اسکریپت چنین جداولی وجود ندارند، می‌توانیم این خطاها را نادیده بگیریم. حال پنجره Server Explorer را تازه (refresh) کنید و خواهید دید که پنج جدول جدید ساخته شده اند.



لیست زیر نحوه Map کردن اطلاعات از جداول SQL Membership به سیستم Identity را نشان می‌دهد.

aspnet_Roles --> **AspNetRoles**

aspnet_Users, aspnet_Membership --> **AspNetUsers**

aspnet_UsersInRoles --> **AspNetUserRoles**

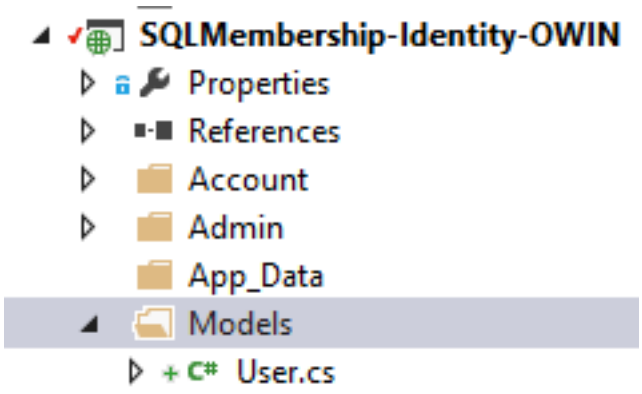
جداول AspNetUserClaims و AspNetUserLogins خالی هستند. فیلد تفکیک کننده (Discriminator) در جدول AspNetUsers باید مطابق نام کلاس مدل باشد، که در مرحله بعدی تعریف خواهد شد. همچنین ستون PasswordHash به فرم 'encrypted password|password salt|password format' می‌باشد. این شما را قادر می‌سازد تا از رمزنگاری برای ذخیره و بازیابی کلمه‌های عبور استفاده کنید. این مورد نیز در ادامه مقاله بررسی شده است.

ساختن مدل‌ها و صفحات عضویت

بصورت پیش فرض سیستم ASP.NET Identity برای دریافت و ذخیره اطلاعات در دیتابیس عضویت از Entity Framework استفاده می‌کند. برای آنکه بتوانیم با جداول موجود کار کنیم، می‌بایست ابتدا مدل هایی که الگوی دیتابیس را نمایندگی می‌کنند ایجاد کنیم. برای این کار مدل‌های ما یا باید اینترفیس‌های موجود در Identity.Core را پیاده سازی کنند، یا می‌توانند پیاده سازی‌های پیش فرض را توسعه دهند. پیاده سازی‌های پیش فرض در Microsoft.AspNet.Identity.EntityFramework وجود دارند.

در نمونه ما، جداول AspNetLogins, AspNetUserClaims, AspNetRoles و AspNetUserRole ستون هایی دارند که شباهت زیادی به پیاده سازی‌های پیش فرض سیستم Identity دارند. در نتیجه می‌توانیم از کلاس‌های موجود، برای Map کردن الگوی جدید استفاده کنیم. جدول AspNetUsers ستون‌های جدیدی نیز دارد. می‌توانیم کلاس جدیدی بسازیم که از **IdentityUser** ارث بری کند و آن را گسترش دهیم تا این فیلدهای جدید را پوشش دهد.

پوشه ای با نام Models بسازید (در صورتی که وجود ندارد) و کلاسی با نام User به آن اضافه کنید.



کلاس User باید کلاس IdentityUser را که در اسمبلی Microsoft.AspNet.Identity.EntityFramework وجود دارد گسترش دهد. خاصیت هایی را تعریف کنید که نماینده الگوی جدول AspNetUser هستند. خواص ID, Username, PasswordHash و SecurityStamp در کلاس IdentityUser تعریف شده اند، بنابراین این خواص را در لیست زیر نمی‌بینید.

```
public class User : IdentityUser
    {
        public User()
        {
            CreateDate = DateTime.Now;
            IsApproved = false;
            LastLoginDate = DateTime.Now;
            LastActivityDate = DateTime.Now;
            LastPasswordChangedDate = DateTime.Now;
            LastLockoutDate = DateTime.Parse("1/1/1754");
            FailedPasswordAnswerAttemptWindowStart = DateTime.Parse("1/1/1754");
            FailedPasswordAttemptWindowStart = DateTime.Parse("1/1/1754");
        }

        public System.Guid ApplicationId { get; set; }
        public string MobileAlias { get; set; }
        public bool IsAnonymous { get; set; }
        public System.DateTime LastActivityDate { get; set; }
        public string MobilePIN { get; set; }
        public string Email { get; set; }
        public string LoweredEmail { get; set; }
        public string LoweredUserName { 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; }
    }
```

حال برای دسترسی به دیتابیس مورد نظر، نیاز به یک DbContext داریم. اسمبلی Microsoft.AspNet.Identity.EntityFramework کلاسی با نام **IdentityDbContext** دارد که پیاده سازی پیش فرض برای دسترسی به دیتابیس ASP.NET Identity است. نکته قابل توجه این است که IdentityDbContext آبجکتی از نوع **TUser** را می‌پذیرد. TUser می‌تواند هر کلاسی باشد که از IdentityUser ارث بری کرده و آن را گسترش می‌دهد.

در پوشه Models کلاس جدیدی با نام **ApplicationDbContext** بسازید که از IdentityDbContext ارث بری کرده و از کلاس

User استفاده می‌کند.

```
public class ApplicationDbContext : IdentityDbContext<User>
{

}
```

مدیریت کاربران در ASP.NET Identity توسط کلاسی با نام UserManager انجام می‌شود که در اسمبلی Microsoft.AspNet.Identity.EntityFramework قرار دارد. چیزی که ما در این مرحله نیاز داریم، کلاسی است که از UserManager ارث بری می‌کند و آن را طوری توسعه می‌دهد که از کلاس User استفاده کند.

در پوشه Models کلاس جدیدی با نام UserManager بسازید.

```
public class UserManager : UserManager<User>
{

}
```

کلمه عبور کاربران بصورت رمز نگاری شده در دیتابیس ذخیره می‌شوند. الگوریتم رمز نگاری SQL Membership با سیستم ASP.NET Identity تفاوت دارد. هنگامی که کاربران قدیمی به سایت وارد می‌شوند، کلمه عبورشان را توسط الگوریتم‌های قدیمی SQL Membership رمزگشایی می‌کنیم، اما کاربران جدید از الگوریتم‌های ASP.NET Identity استفاده خواهند کرد.

کلاس UserManager خاصیتی با نام **PasswordHasher** دارد. این خاصیت نمونه ای از یک کلاس را ذخیره می‌کند، که اینترفیس IPasswordHasher را پیاده سازی کرده است. این کلاس هنگام تراکنش‌های احراز هویت کاربران استفاده می‌شود تا کلمه‌های عبور را رمزنگاری/رمزگشایی شوند. در کلاس UserManager کلاس جدیدی بنام SQLPasswordHasher بسازید. کد کامل را در لیست زیر مشاهده می‌کنید.

```
public class SQLPasswordHasher : PasswordHasher
{
        public override string HashPassword(string password)
        {
            return base.HashPassword(password);
        }

        public override PasswordVerificationResult VerifyHashedPassword(string hashedPassword, string
providedPassword)
        {
            string[] passwordProperties = hashedPassword.Split('|');
            if (passwordProperties.Length != 3)
            {
                return base.VerifyHashedPassword(hashedPassword, providedPassword);
            }
            else
            {
                string passwordHash = passwordProperties[0];
                int passwordformat = 1;
                string salt = passwordProperties[2];
                if (String.Equals(EncryptPassword(providedPassword, passwordformat, salt),
passwordHash, StringComparison.CurrentCultureIgnoreCase))
                {
                    return PasswordVerificationResult.SuccessRehashNeeded;
                }
                else
                {
                    return PasswordVerificationResult.Failed;
                }
            }
        }

        //This is copied from the existing SQL providers and is provided only for back-compat.
        private string EncryptPassword(string pass, int passwordFormat, string salt)
        {
            if (passwordFormat == 0) // MembershipPasswordFormat.Clear
                return pass;

            byte[] bIn = Encoding.Unicode.GetBytes(pass);
```

```
            byte[] bSalt = Convert.FromBase64String(salt);
            byte[] bRet = null;

            if (passwordFormat == 1)
            { // MembershipPasswordFormat.Hashed
                HashAlgorithm hm = HashAlgorithm.Create("SHA1");
                if (hm is KeyedHashAlgorithm)
                {
                    KeyedHashAlgorithm kha = (KeyedHashAlgorithm)hm;
                    if (kha.Key.Length == bSalt.Length)
                    {
                        kha.Key = bSalt;
                    }
                    else if (kha.Key.Length < bSalt.Length)
                    {
                        byte[] bKey = new byte[kha.Key.Length];
                        Buffer.BlockCopy(bSalt, 0, bKey, 0, bKey.Length);
                        kha.Key = bKey;
                    }
                    else
                    {
                        byte[] bKey = new byte[kha.Key.Length];
                        for (int iter = 0; iter < bKey.Length; )
                        {
                            int len = Math.Min(bSalt.Length, bKey.Length - iter);
                            Buffer.BlockCopy(bSalt, 0, bKey, iter, len);
                            iter += len;
                        }
                        kha.Key = bKey;
                    }
                    bRet = kha.ComputeHash(bIn);
                }
                else
                {
                    byte[] bAll = new byte[bSalt.Length + bIn.Length];
                    Buffer.BlockCopy(bSalt, 0, bAll, 0, bSalt.Length);
                    Buffer.BlockCopy(bIn, 0, bAll, bSalt.Length, bIn.Length);
                    bRet = hm.ComputeHash(bAll);
                }
            }

            return Convert.ToBase64String(bRet);
    }
}
```

دقت کنید تا فضاهای نام System.Text و System.Security.Cryptography را وارد کرده باشید.

متد EncodePassword کلمه عبور را بر اساس پیاده سازی پیش فرض SQL Membership رمزنگاری می‌کند. این الگوریتم از System.Web گرفته می‌شود. اگر اپلیکیشن قدیمی شما از الگوریتم خاصی استفاده می‌کرده است، همینجا باید آن را منعکس کنید. دو متد دیگر نیز بنام‌های *HashPassword* و *VerifyHashedPassword* نیاز داریم. این متدها از EncodePassword برای رمزنگاری کلمه‌های عبور و تایید آنها در دیتابیس استفاده می‌کنند.

سیستم SQL Membership برای رمزنگاری (Hash) کلمه‌های عبور هنگام ثبت نام و تغییر آنها توسط کاربران، از PasswordHash, PasswordSalt و PasswordFormat استفاده می‌کرد. در روند مهاجرت، این سه فیلد در ستون PasswordHash جدول AspNetUsers ذخیره شده و با کاراکتر '|' جدا شده اند. هنگام ورود کاربری به سایت، اگر کله عبور شامل این فیلدها باشد از الگوریتم SQL Membership برای بررسی آن استفاده می‌کنیم. در غیر اینصورت از پیاده سازی پیش فرض ASP.NET Identity استفاده خواهد شد. با این روش، کاربران قدیمی لازم نیست کلمه‌های عبور خود را صرفا بدلیل مهاجرت اپلیکیشن ما تغییر دهند.

کلاس UserManager را مانند قطعه کد زیر بروز رسانی کنید.

```
public UserManager()
            : base(new UserStore<User>(new ApplicationDbContext()))
        {
            this.PasswordHasher = new SQLPasswordHasher();
 }
```

ایجاد صفحات جدید مدیریت کاربران

قدم بعدی ایجاد صفحاتی است که به کاربران اجازه ثبت نام و ورود را می‌دهند. صفحات قدیمی SQL Membership از کنترل هایی استفاده می‌کنند که با ASP.NET Identity سازگار نیستند. برای ساختن این صفحات جدید به این مقاله مراجعه کنید. از آنجا که در این مقاله پروژه جدید را ساخته ایم و پکیج‌های لازم را هم نصب کرده ایم، می‌توانید مستقیما به قسمت Adding Web Forms for registering users to your application بروید.

چند تغییر که باید اعمال شوند:

فایل‌های Register.aspx.cs و Login.aspx.cs از کلاس UserManager استفاده می‌کنند. این ارجاعات را با کلاس UserManager جدیدی که در پوشه Models ساختید جایگزین کنید.

همچنین ارجاعات استفاده از کلاس IdentityUser را به کلاس User که در پوشه Models ساختید تغییر دهید.

لازم است توسعه دهنده مقدار ApplicationId را برای کاربران جدید طوری تنظیم کند که با شناسه اپلیکیشن جاری تطابق داشته باشد. برای این کار می‌توانید پیش از ساختن حساب‌های کاربری جدید در فایل Register.aspx.cs ابتدا شناسه اپلیکیشن را بدست آورید و اطلاعات کاربر را بدرستی تنظیم کنید.

مثال: در فایل Register.aspx.cs متد جدیدی تعریف کنید که جدول aspnet_Applications را بررسی میکند و شناسه اپلیکیشن را بر اساس نام اپلیکیشن بدست می‌آورد.

```
private Guid GetApplicationID()
{
    using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ApplicationServices"].ConnectionString))
            {
                string queryString = "SELECT ApplicationId from aspnet_Applications WHERE
ApplicationName = '/'"; //Set application name as in database

                SqlCommand command = new SqlCommand(queryString, connection);
                command.Connection.Open();

                var reader = command.ExecuteReader();
                while (reader.Read())
                {
                    return reader.GetGuid(0);
                }

            return Guid.NewGuid();
        }
}
```

حال می‌توانید این مقدار را برای آبجکت کاربر تنظیم کنید.

```
var currentApplicationId = GetApplicationID();

User user = new User() { UserName = Username.Text,
ApplicationId=currentApplicationId, …};
```

در این مرحله می‌توانید با استفاده از اطلاعات پیشین وارد سایت شوید، یا کاربران جدیدی ثبت کنید. همچنین اطمینان حاصل کنید که کاربران پیشین در نقش‌های مورد نظر وجود دارند.

مهاجرت به ASP.NET Identity مزایا و قابلیت‌های جدیدی را به شما ارائه می‌کند. مثلا کاربران می‌توانند با استفاده از تامین کنندگان ثالثی مثل Facebook, Google, Microsoft, Twitter و غیره به سایت وارد شوند. اگر به سورس کد این مقاله مراجعه کنید خواهید دید که امکانات OAuth نیز فعال شده اند.

در این مقاله انتقال داده‌های پروفایل کاربران بررسی نشد. اما با استفاده از نکات ذکر شده می‌توانید پروفایل کاربران را هم بسادگی منتقل کنید. کافی است مدل‌های لازم را در پروژه خود تعریف کرده و با استفاده از اسکریپت‌های SQL داده‌ها را انتقال دهید.

نظرات خوانندگان

نویسنده: سمیرا قادری
تاریخ: ۱۳۹۲/۱۰/۲۳ ۱۳:۳۳

سلام ممنون از مطالب خوبتون .
زمانیکه در اپلکیشن خودم asp.net Configuration و تب Security را انتخاب می‌نمایم با پیغام زیر مواجه می‌شوم Unable to connect to SQL Server database.
دستور زیر را هم اجرا کردم
C:\Windows\Microsoft.NET\Framework\v4.0.30319\aspnet_regsql.exe -S . -A all -d tt -U sa_tt -P asd123
در صورتیکه از login و password آن مطمئن هستم ولی مشکل حل نشد
در صورتیکه مشکل از permission هست چگونه باید آن را حل کنم
webconfig برنامه به شرح زیر است
Data Source=.;Initial Catalog=tt;User ID=sa_tt;Password=asd123
با تشکر

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۲۳ ۱۴:۴۵

تنظیمات وب کانفیگ خودتون رو با نمونه MSDN برای تعریف membership مطابقت بدید. جزئیات قسمت‌های مختلف آن

نویسنده: حمید
تاریخ: ۱۳۹۲/۱۰/۲۳ ۲۱:۱۰

خیلی ممنون آموزش‌های مفیدتون!
فرض کنید ما یک سیستم Authentication مثل Membership داریم با 2 تا نقش. حالا می‌خواهیم نقش A به فرم F1 و بعضی از متدهاش دسترسی داشته و در فرم F2، نقش B به آن دسترسی داشته باشه. خوب تا اینجا فکر کنم پیاده سازیش راحت باشه. ولی مساله از جایی پیچیده میشه که ما بخواهیم در زمان اجرا این تنظیمات رو عوض کنیم مثلا: در فرم F1، نقش A دسترسیش به متدها و حتی به کل فرم تغییر کنه و یا اجازه دسترسی به فرم B بهش داده بشه و این شرایط رو برای چندین نقش و یا در سطح کاربر داشت توصیه چیه؟ از چه تکنولوژی و یا راهکاری میشه به این مقصود رسید.
خیلی ممنون

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۲۳ ۲۱:۳۷

پایه‌اش همین مسایل هست. فقط قسمتی که کار Authorization انجام میشه رو می‌توان سفارشی کرد. مثلا:

Dynamic Controller/Action Authorization in ASP.NET MVC
MVC Dynamic Authorization
ASP.NET MVC Custom Authorize Attribute with Roles Parser

نویسنده: کامران سادین
تاریخ: ۱۳۹۲/۱۱/۰۸ ۲:۱۳

ممنون از مطلب خوبتون.
من جداول خودم رو برای احراز هویت دارم آیا میشه همین جداول رو با ASP.NET Identity کار کنم؟
اگر نمیشه، این بانک ASP.NET Identity که توی SQL Server نمیسازه بانک Database.mdf رو میسازه، میشه توی SQL بسازیم از

همون ابتدای پروژه؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۱/۰۸ ۹:۲۰

سیستم کارش EF Code first هست. این سیستم کدهاش گره خورده به بانک اطلاعاتی خاصی نیست. الان در این مثال رشته اتصالی به یک localdb اشاره می‌کنه. شما می‌تونید کلا این رشته و نحوه‌ی تعریف اون رو برای کار با SQL Server یا SQL CE یا هر بانک اطلاعاتی دیگری که پروایدر code first داره، تغییر بدید و استفاده کنید. (و اگر با ef code first آشنایی ندارید، کم کم در آینده نمی‌تونید با کتابخانه‌های کمکی و جانبی دات نت کار کنید)

نویسنده: کامران سادین
تاریخ: ۱۳۹۲/۱۱/۰۸ ۱۰:۲۷

مقاله ای نیز راجع به تغییر فیلدهاش هم بنویسید بد نیست دوستان علاقه دارند. با تشکر.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۱/۰۸ ۱۰:۵۸

« سفارشی کردن ASP.NET Identity در MVC 5 » بیشتر در اینجا

نویسنده: مهدی
تاریخ: ۱۳۹۳/۰۱/۱۹ ۱۲:۳۴

دوست عزیز برای حل این مشکل شما باید پوشه سورس برنامه رو تغییر بدین. این مشکل به دلیل وجود خط فاصله تو آدرس (دایرکتوری) محل قرار گیری پروژه هست. اگه محل پروژه رو در جایی قرار دهید که در آدرس آن از فاصله استفاده نشده باشد به خوبی کار خواهد کرد.