عنوان: **معرفی ASP.NET Identity**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۴ ۹:۴۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, Entity framework, MVC, Security, ASP.NET Identity

سیستم ASP.NET Membership بهمراه ASP.NET 2.0 در سال 2005 معرفی شد، و از آن زمان تا بحال تغییرات زیادی در چگونگی مدیریت احزار هویت و اختیارات کاربران توسط اپلیکیشن‌های وب بوجود آمده است. ASP.NET Identity نگاهی تازه است به آنچه که سیستم Membership هنگام تولید اپلیکیشن‌های مدرن برای وب، موبایل و تبلت باید باشد.

پیش زمینه: سیستم عضویت در ASP.NET

ASP.NET Membership

ASP.NET Membership طراحی شده بود تا نیازهای سیستم عضویت وب سایت‌ها را تامین کند، نیازهایی که در سال 2005 رایج بود و شامل مواردی مانند مدل احراز هویت فرم، و یک پایگاه داده SQL Server برای ذخیره اطلاعات کاربران و پروفایل هایشان می‌شد. امروزه گزینه‌های بسیار بیشتری برای ذخیره داده‌های وب اپلیکیشن‌ها وجود دارد، و اکثر توسعه دهندگان می‌خواهند از اطلاعات شبکه‌های اجتماعی نیز برای احراز هویت و تعیین سطوح دسترسی کاربرانشان استفاده کنند. محدودیت‌های طراحی سیستم ASP.NET Membership گذر از این تحول را دشوار می‌کند:

الگوی پایگاه داده آن برای SQL Server طراحی شده است، و قادر به تغییرش هم نیستید. می‌توانید اطلاعات پروفایل را اضافه کنید، اما تمام داده‌ها در یک جدول دیگر ذخیره می‌شوند، که دسترسی به آنها نیز مشکل‌تر است، تنها راه دسترسی Profile Provider API خواهد بود.

سیستم تامین کننده (Provider System) امکان تغییر منبع داده‌ها را به شما می‌دهد، مثلا می‌توانید از بانک‌های اطلاعاتی MySQL یا Oracle استفاده کنید. اما تمام سیستم بر اساس پیش فرض هایی طراحی شده است که تنها برای بانک‌های اطلاعاتی relational درست هستند. می‌توانید تامین کننده (Provider) ای بنویسید که داده‌های سیستم عضویت را در منبعی به غیر از دیتابیس‌های relational ذخیره می‌کند؛ مثلا Windows Azure Storage Tables. اما در این صورت باید مقادیر زیادی کد بنویسید. مقادیر زیادی هم System.NotImplementedException باید بنویسید، برای متد هایی که به دیتابیس‌های NoSQL مربوط نیستند.

از آنجایی که سیستم ورود/خروج سایت بر اساس مدل Forms Authentication کار می‌کند، سیستم عضویت نمی‌تواند از OWIN استفاده کند. OWIN شامل کامپوننت هایی برای احراز هویت است که شامل سرویس‌های خارجی هم می‌شود (مانند Microsoft Accounts, Facebook, Google, Twitter). همچنین امکان ورود به سیستم توسط حساب‌های کاربری سازمانی (Organizational Accounts) نیز وجود دارد مانند Active Directory و Windows Azure Active Directory. این کتابخانه از JWT، OAuth 2.0 و CORS نیز پشتیبانی می‌کند.

ASP.NET Simple Membership

ASP.NET simple membership به عنوان یک سیستم عضویت، برای فریم ورک Web Pages توسعه داده شد. این سیستم با WebMatrix و Visual Studio 2010 SP1 انتشار یافت. هدف از توسعه این سیستم، آسان کردن پروسه افزودن سیستم عضویت به یک اپلیکیشن Web Pages بود. این سیستم پروسه کلی کار را آسان‌تر کرد، اما هنوز مشکلات ASP.NET Membership را نیز داشت.

محدودیت هایی نیز وجود دارند:

ذخیره داده‌های سیستم عضویت در بانک‌های اطلاعاتی non-relational مشکل است.

نمی توانید از آن در کنار OWIN استفاده کنید.

با فراهم کننده‌های موجود ASP.NET Membership بخوبی کار نمی‌کند. توسعه پذیر هم نیست.

ASP.NET Universal Providers

ASP.NET Universal Providers برای ذخیره سازی اطلاعات سیستم عضویت در Windows Azure SQL Database توسعه پیدا کردند. با SQL Server Compact هم بخوبی کار می‌کنند. این تامین کننده‌ها بر اساس Entity Framework Code First ساخته شده بودند و بدین معنا بود که داده‌های سیستم عضویت را می‌توان در هر منبع داده ای که توسط EF پشتیبانی می‌شود ذخیره کرد. با انتشار این تامین کننده‌ها الگوی دیتابیس سیستم عضویت نیز بسیار سبک‌تر و بهتر شد. اما این سیستم بر پایه زیر ساخت ASP.NET

Membership نوشته شده است، بنابراین محدودیت‌های پیشین مانند محدودیت‌های SqlMembershipProvider هنوز وجود دارند. به بیان دیگر، این سیستم‌ها همچنان برای بانک‌های اطلاعاتی relational طراحی شده اند، پس سفارشی سازی اطلاعات کاربران و پروفایل‌ها هنوز مشکل است. در آخر آنکه این تامین کننده‌ها هنوز از مدل احراز هویت فرم استفاده می‌کنند.

ASP.NET Identity همانطور که داستان سیستم عضویت ASP.NET طی سالیان تغییر و رشد کرده است، تیم ASP.NET نیز آموخته‌های زیادی از بازخوردهای مشتریان شان بدست آورده اند. این پیش فرض که کاربران شما توسط یک نام کاربری و کلمه عبور که در اپلیکیشن خودتان هم ثبت شده است به سایت وارد خواهند شد، دیگر معتبر نیست. دنیای وب اجتماعی شده است. کاربران از طریق وب سایت‌ها و شبکه‌های اجتماعی متعددی با یکدیگر در تماس هستند، خیلی از اوقت بصورت زنده! شبکه هایی مانند Facebook و Twitter. همانطور که توسعه نرم افزارهای تحت وب رشد کرده است، الگوها و مدل‌های پیاده سازی نیز تغییر و رشد کرده اند. امکان Unit Testing روی کد اپلیکیشن‌ها، یکی از مهم‌ترین دلواپسی‌های توسعه دهندگان شده است. در سال 2008 تیم ASP.NET فریم ورک جدیدی را بر اساس الگوی Model-View-Controller (MVC) اضافه کردند. هدف آن کمک به توسعه دهندگان، برای تولید برنامه‌های ASP.NET با قابلیت Unit Testing بهتر بود. توسعه دهندگانی که می‌خواستند کد اپلیکیشن‌های خود را Unit Test کنند، همین امکان را برای سیستم عضویت نیز می‌خواستند.
با در نظر گرفتن تغییراتی که در توسعه اپلیکیشن‌های وب بوجود آمده ASP.NET Identity با اهداف زیر متولد شد:
یک سیستم هویت واحد (One ASP.NET Identity system)
سیستم ASP.NET Identity می‌تواند در تمام فریم ورک‌های مشتق از ASP.NET استفاده شود. مانند ASP.NET MVC, Web Forms, Web Pages, Web API و SignalR

از این سیستم می‌توانید در تولید اپلیکیشن‌های وب، موبایل، استور (Store) و یا اپلیکیشن‌های ترکیبی استفاده کنید.

سادگی تزریق داده‌های پروفایل درباره کاربران
روی الگوی دیتابیس برای اطلاعات کاربران و پروفایل‌ها کنترل کامل دارید. مثلا می‌توانید به سادگی یک فیلد، برای تاریخ تولد در نظر بگیرید که کاربران هنگام ثبت نام در سایت باید آن را وارد کنند.

کنترل ذخیره سازی/واکشی اطلاعات
بصورت پیش فرض ASP.NET Identity تمام اطلاعات کاربران را در یک دیتابیس ذخیره می‌کند. تمام مکانیزم‌های دسترسی به داده‌ها توسط EF Code First کار می‌کنند.
از آنجا که روی الگوی دیتابیس، کنترل کامل دارید، تغییر نام جداول و یا نوع داده فیلدهای کلیدی و غیره ساده است.
استفاده از مکانیزم‌های دیگر برای مدیریت داده‌های آن ساده است، مانند SharePoint, Windows Azure Storage Table و دیتابیس‌های NoSQL.

تست پذیری
ASP.NET Identity تست پذیری اپلیکیشن وب شما را بیشتر می‌کند. می‌توانید برای تمام قسمت هایی که از ASP.NET Identity استفاده می‌کنند تست بنویسید.

تامین کننده نقش (Role Provider)
تامین کننده ای وجود دارد که به شما امکان محدود کردن سطوح دسترسی بر اساس نقوش را می‌دهد. بسادگی می‌توانید نقش‌های جدید مانند "Admin" بسازید و بخش‌های مختلف اپلیکیشن خود را محدود کنید.

Claims Based
ASP.NET Identity از امکان احراز هویت بر اساس Claims نیز پشتیبانی می‌کند. در این مدل، هویت کاربر بر اساس دسته ای از اختیارات او شناسایی می‌شود. با استفاده از این روش توسعه دهندگان برای تعریف هویت کاربران، آزادی عمل بیشتری نسبت به مدل Roles دارند. مدل نقش‌ها تنها یک مقدار منطقی (bool) است؛ یا عضو یک نقش هستید یا خیر، در حالیکه با استفاده از روش Claims می‌توانید اطلاعات بسیار ریز و دقیقی از هویت کاربر در دست داشته باشید.

تامین کنندگان اجتماعی

به راحتی می‌توانید از تامین کنندگان دیگری مانند Microsoft, Facebook, Twitter, Google و غیره استفاده کنید و اطلاعات مربوط به کاربران را در اپلیکیشن خود ذخیره کنید.

Windows Azure Active Directory

برای اطلاعات بیشتر به این لینک مراجعه کنید.

یکپارچگی با OWIN

ASP.NET Identity بر اساس OWIN توسعه پیدا کرده است، بنابراین از هر میزبانی که از OWIN پشتیبانی می‌کند می‌توانید استفاده کنید. همچنین هیچ وابستگی ای به System.Web وجود ندارد. ASP.NET Identity یک فریم ورک کامل و مستقل برای OWIN است و می‌تواند در هر اپلیکیشنی که روی OWIN میزبانی شده استفاده شود.

ASP.NET Identity از OWIN برای ورود/خروج کاربران در سایت استفاده می‌کند. این بدین معنا است که بجای استفاده از Forms Authentication برای تولید یک کوکی، از OWIN CookieAuthentication استفاده می‌شود.

پکیج NuGet

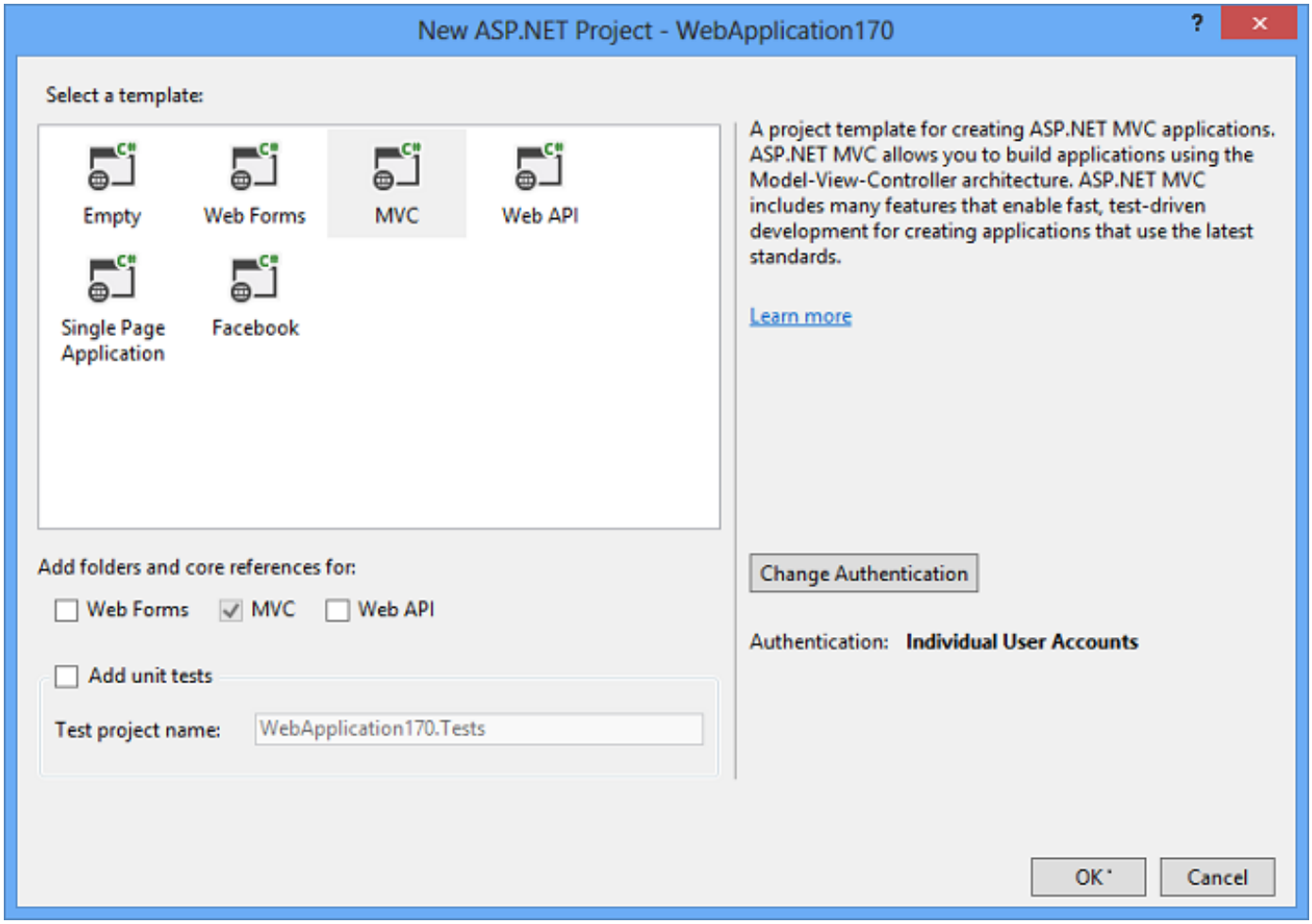
ASP.NET Identity در قالب یک بسته NuGet توزیع می‌شود. این بسته در قالب پروژه‌های ASP.NET MVC, Web Forms و Web API که با Visual Studio 2013 منتشر شدند گنجانده شده است.

توزیع این فریم ورک در قالب یک بسته NuGet این امکان را به تیم ASP.NET می‌دهد تا امکانات جدیدی توسعه دهند، باگ‌ها را برطرف کنند و نتیجه را بصورت چابک به توسعه دهندگان عرضه کنند.

شروع کار با ASP.NET Identity

ASP.NET Identity در قالب پروژه‌های ASP.NET MVC, Web Forms, Web API و SPA که بهمراه Visual Studio 2013 منتشر شده اند استفاده می‌شود. در ادامه به اختصار خواهیم دید که چگونه ASP.NET Identity کار می‌کند.

یک پروژه جدید ASP.NET MVC با تنظیمات Individual User Accounts بسازید.

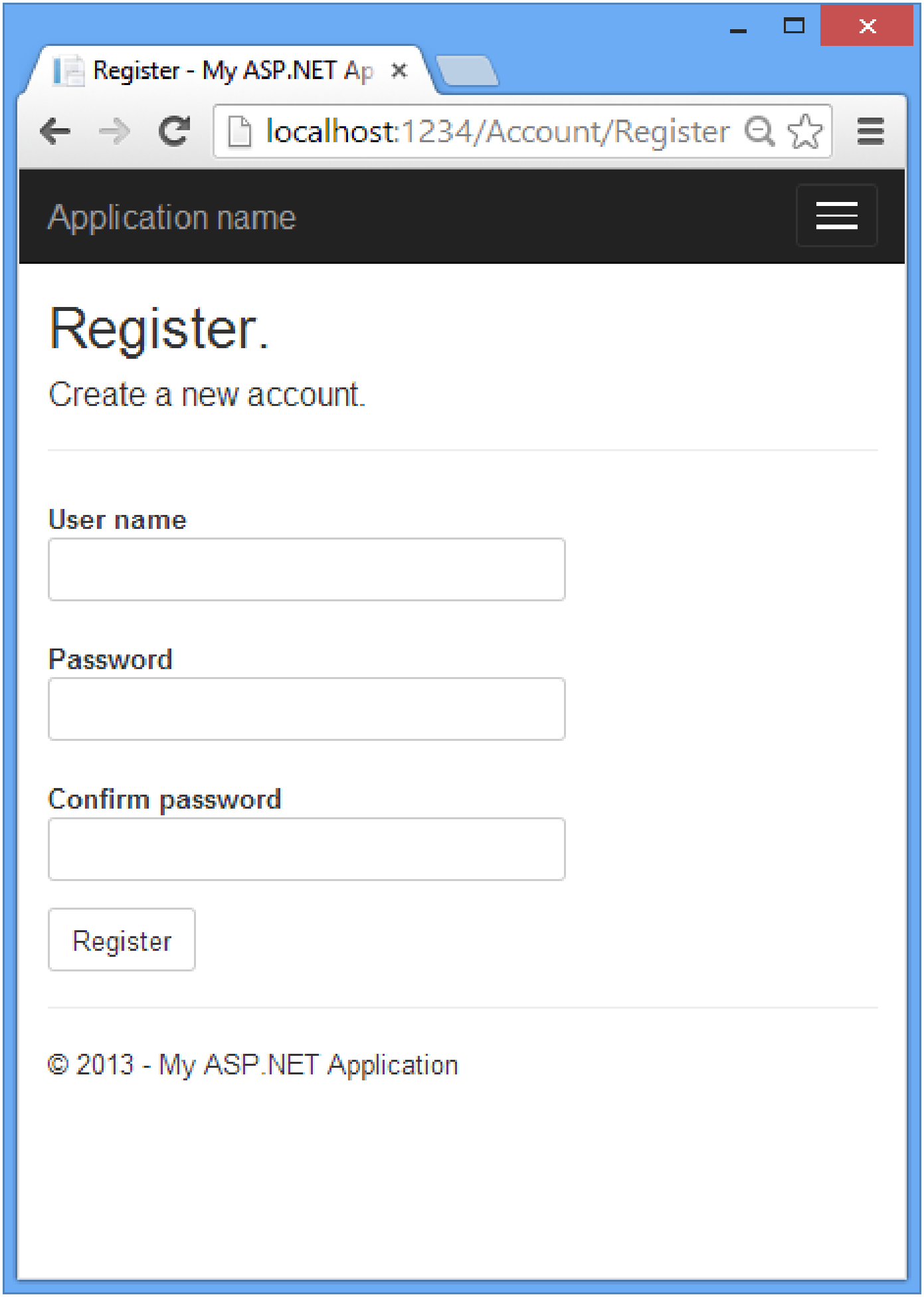


پروژه ایجاد شده شامل سه بسته می‌شود که مربوط به ASP.NET Identity هستند:

Microsoft.AspNet.Identity.EntityFramework این بسته شامل پیاده سازی ASP.NET Identity با Entity Framework می‌شود، که تمام داده‌های مربوطه را در یک دیتابیس SQL Server ذخیره می‌کند.
Microsoft.AspNet.Identity.Core این بسته محتوی تمام interfaceهای ASP.NET Identity است. با استفاده از این بسته می‌توانید پیاده سازی دیگری از ASP.NET Identity بسازید که منبع داده متفاوتی را هدف قرار می‌دهد. مثلا Windows Azure Storage Table و دیتابیس‌های NoSQL.

Microsoft.AspNet.Identity.OWIN این بسته امکان استفاده از احراز هویت OWIN را در اپلیکیشن‌های ASP.NET فراهم می‌کند. هنگام تولید کوکی‌ها از OWIN Cookie Authentication استفاده خواهد شد.

اپلیکیشن را اجرا کرده و روی لینک Register کلیک کنید تا یک حساب کاربری جدید ایجاد کنید.


Register - My ASP.NET Ap
localhost:1234/Account/Register
Application name
Register.
Create a new account.
User name
Password
Confirm password
Register
© 2013 - My ASP.NET Application

هنگامیکه بر روی دکمه‌ی Register کلیک شود، کنترلر Account، اکشن متد Register را فراخوانی می‌کند تا حساب کاربری جدیدی با استفاده از ASP.NET Identity API ساخته شود.

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = new ApplicationUser() { UserName = model.UserName };
        var result = await UserManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
            await SignInAsync(user, isPersistent: false);
            return RedirectToAction("Index", "Home");
        }
        else
        {
            AddErrors(result);
        }
    }

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

اگر حساب کاربری با موفقیت ایجاد شود، کاربر توسط فراخوانی متد SignInAsync به سایت وارد می‌شود.

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = new ApplicationUser() { UserName = model.UserName };
        var result = await UserManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
            await SignInAsync(user, isPersistent: false);
            return RedirectToAction("Index", "Home");
        }
        else
        {
            AddErrors(result);
        }
    }

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

```
private async Task SignInAsync(ApplicationUser user, bool isPersistent)
{
    AuthenticationManager.SignOut(DefaultAuthenticationTypes.ExternalCookie);

    var identity = await UserManager.CreateIdentityAsync(
       user, DefaultAuthenticationTypes.ApplicationCookie);

    AuthenticationManager.SignIn(
       new AuthenticationProperties() {
          IsPersistent = isPersistent
       }, identity);
}
```

از آنجا که ASP.NET Identity و OWIN Cookie Authentication هر دو Claims-based هستند، فریم ورک، انتظار آبجکتی از نوع ClaimsIdentity را خواهد داشت. این آبجکت تمامی اطلاعات لازم برای تشخیص هویت کاربر را در بر دارد. مثلا اینکه کاربر

مورد نظر به چه نقش هایی تعلق دارد؟ و اطلاعاتی از این قبیل. در این مرحله می‌توانید Claimهای بیشتری را به کاربر بیافزایید.

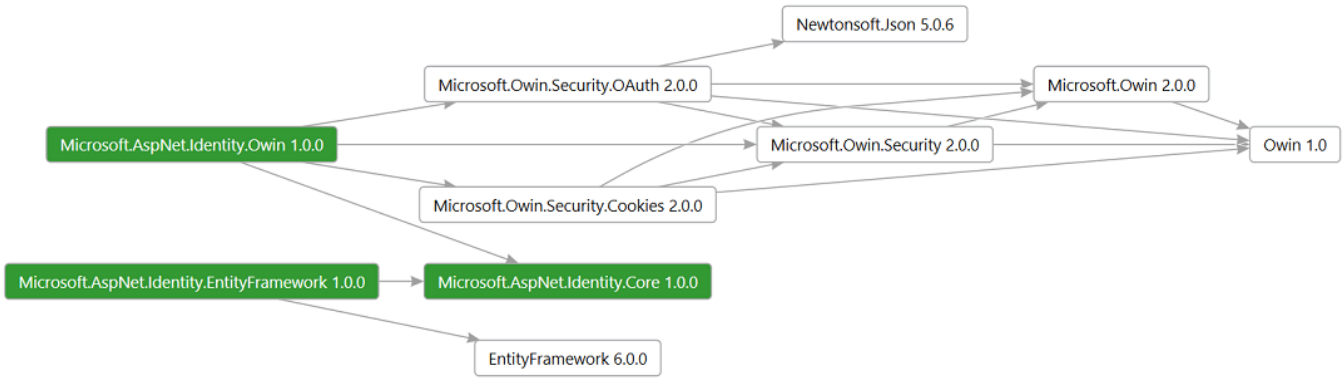
کلیک کردن روی لینک Log off در سایت، اکشن متد LogOff در کنترلر Account را اجرا می‌کند.

```
// POST: /Account/LogOff
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult LogOff()
{
    AuthenticationManager.SignOut();
    return RedirectToAction("Index", "Home");
}
```

همانطور که مشاهده می‌کنید برای ورود/خروج کاربران از AuthenticationManager استفاده می‌شود که متعلق به OWIN است. متد SignOut همتای متد FormsAuthentication.SignOut است.

کامپوننت‌های ASP.NET Identity

تصویر زیر اجزای تشکیل دهنده ASP.NET Identity را نمایش می‌دهد. بسته هایی که با رنگ سبز نشان داده شده اند سیستم کلی ASP.NET Identity را می‌سازند. مابقی بسته‌ها وابستگی هایی هستند که برای استفاده از ASP.NET Identity در اپلیکیشن‌های ASP.NET لازم اند.



دو پکیج دیگر نیز وجود دارند که به آنها اشاره نشد:
Microsoft.Security.Owin.Cookies این بسته امکان استفاده از مدل احراز هویت مبتنی بر کوکی (Cookie-based Authentication) را فراهم می‌کند. مدلی مانند سیستم ASP.NET Forms Authentication.
EntityFramework که نیازی به معرفی ندارد.

مهاجرت از Membership به ASP.NET Identity

تیم ASP.NET و مایکروسافت هنوز راهنمایی رسمی، برای این مقوله ارائه نکرده اند. گرچه پست‌های وبلاگ‌ها و منابع مختلفی وجود دارند که از جنبه‌های مختلفی به این مقوله پرداخته اند. امیدواریم تا در آینده نزدیک مایکروسافت راهنمایی‌های لازم را منتشر کند، ممکن است ابزار و افزونه هایی نیز توسعه پیدا کنند. اما در یک نگاه کلی می‌توان گفت مهاجرت بین این دو فریم ورک زیاد ساده نیست. تفاوت‌های فنی و ساختاری زیادی وجود دارند، مثلا الگوی دیتابیس‌ها برای ذخیره اطلاعات کاربران، مبتنی بودن بر فریم

ورک OWIN و غیره. اگر قصد اجرای پروژه جدیدی را دارید پیشنهاد می‌کنم از فریم ورک جدید مایکروسافت ASP.NET Identity استفاده کنید.

قدم‌های بعدی Create an ASP.NET MVC 5 App with Facebook and Google OAuth2 and OpenID Sign-on

در این مقاله خواهید دید چگونه اطلاعات پروفایل را اضافه کنید و چطور از ASP.NET Identity برای احراز هویت کاربران توسط Facebook و Google استفاده کنید. Deploy a Secure ASP.NET MVC app with Membership, OAuth, and SQL Database to a Windows Azure Web Site

Individual User Accounts

Organizational Accounts

Customizing profile information in ASP.NET Identity in VS 2013 templates

Get more information from Social providers used in the VS 2013 project templates

https://github.com/rustd/AspnetIdentitySample

پروژه نمونه ASP.NET Identity می‌تواند مفید باشد. در این پروژه نحوه کارکردن با کاربران و نقش‌ها و همچنین نیازهای مدیریتی رایج نمایش داده شده.

نظرات خوانندگان

نویسنده: ناظم
تاریخ: ۱۳۹۲/۱۰/۱۴ ۱۱:۴۶

سلام دوست عزیز
به خاطر این مطلب بسیار خوبتون سپاسگذارم.
آیا کتاب یا راهنمای جامعی برای کسی مثل من که اصلا تجربه کار با هیچ کدام از این سیستم‌های مدیریت کاربر رو نداره سراغ دارین؟ لینکهای بالا همشون موردی هستن

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۱۴ ۱۲:۳۰

مباحث ابتدایی Forms Authentication مربوط است به ASP.NET 1.x؛ دوره‌ای در این مورد . همچنین در MVC هم قابل استفاده است (^ و ^). مباحث membership هم مربوط است به ASP.NET 2.x. کتابی در این مورد .

نویسنده: مهران
تاریخ: ۱۳۹۲/۱۰/۱۴ ۲۱:۳۹

سلام
با تشکر از مطالب و ارائه خوبتون؛
خواستم بدونم چطور میشه، کاربر را بر اساس عملیات کنترلر (Actions) تعیین دسترسی کرد؟

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۴ ۲۱:۴۴

تا بحال با کتاب یا دوره جامعی درباره ASP.NET Identity مواجه نشدم، اگر منبع مناسبی پیدا کنم به اشتراک میذارم. در چند پست آتی بیشتر درباره این فریم ورک صحبت خواهم کرد و مثال هایی عملی نیز در نظر خواهم گرفت

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۴ ۲۳:۲۷

سوالتون رو دقیقا متوجه نشدم. از خاصیت Authorize میتونید استفاده کنید، که قابل اعمال بر روی تک تک اکشن متدها و یا کل کنترلر است. خاصیت AllowAnonymous برای دسترسی عمومی استفاده می‌شود. برای اطلاعات بیشتر درباره نحوه استفاده از ASP.NET Identity و ساختار کلی OWIN لطفا به لینک‌های ضمیمه شده مراجعه کنید.

```
[Authorize]
public controller account
{
    public ActionResult Index()  { }
    public ActionResult Manage() { }

    [AllowAnonymous]
    public ActionResult Info() { }
}

[Authorize(Roles="Admin")]
public controller admin
{
    public ActionResult Index() { }
}
```

نویسنده: مهران
تاریخ: ۱۳۹۲/۱۰/۱۵ ۰:۱

فکر میکنم با مطالعه بیشتر Claims Based که در مطلب اشاره کردید، به راه حل مورد نظر برسم.

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۵ ۲:۱۶

درسته، درباره Claims-based authorization هم یک یا دو پست می‌نویسم.

نویسنده: وحید
تاریخ: ۱۳۹۲/۱۰/۲۰ ۲:۱۸

با تشکر از مطلب مفیدتون سوالی داشتم
یک جا گفتید " مثلا الگوی دیتابیس‌ها برای ذخیره اطلاعات کاربران " منظورتون از دیتابیس Table هست؟
چرا از کد زیر task برای خروجی استفاده کردید

```
async Task<ActionResult> Register
```

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۲۱ ۶:۴۹

منظور دیتابیس سیستم عضویت است، همانطور که گفته شد این دیتابیس توسط EF ساخته می‌شود، بنابراین جداول، فیلدها و دیگر موارد را میتوانید سفارشی کنید.

همانطور که از امضای این متد مشخص است، عملیات بصورت Async پردازش می‌شوند. برای اطلاعات بیشتر به این لینک نمونه مراجعه کنید.

نویسنده: Programmer
تاریخ: ۱۳۹۲/۱۰/۲۳ ۱۷:۱۸

با عرض سلام و تشکر بابت پست‌های مفید که یقیناً خیلی براش زحمت می‌کشید.

اگر ممکنه کمی در مورد ساختار Identity و کلاس‌ها و اینترفیس‌ها و نحوه کار با اونها بصورتی که بتونیم خودمون هم Custom Implement اش رو انجام بدیم توضیح بدید.

اینکه اینترفیس هایی چون IUser و IUserStore و IUserPasswordStore و IUserSecurityStampStore و در طرف دیگه UserManager و UserStore و IdentityUser چی هستند و با چه هدفی تعریف شدند و قراره چیکار کنند و در آخر برای اینکه مطابق آموزش‌هایی که در سایت هست در مورد MVC و تعریف لایه‌های مختلف و سرویس‌های مورد نیاز، چطور باید از Identity استفاده کرد که هماهنگ با اون مطالب باشه؟

ممنون

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۲۳ ۲۰:۴

در پست‌های آتی پیاده سازی یک تامین کننده (Provider) برای MySQL رو بررسی می‌کنم. برای اطلاعات بیشتر به Implementing a Custom MySQL ASP.NET Identity Storage Provider مراجعه کنید.

هنوز مستندات کامل و رسمی برای این فریم ورک عرضه نشده اما مطالب مفید زیادی در اینترنت وجود دارند. چند لینک در زیر لیست شده:

ASP.NET Core Identity

The good, the bad and the ugly of ASP.NET Identity

درباره تطابق با آموزش‌های سایت: دقیقا متوجه نشدم منظورتون کدوم الگوها است، اما چند نکته تکمیلی:
بصورت پیش فرض وقتی ASP.NET Identity به پروژه اضافه میشه کلاسی بنام ApplicationDbContext ایجاد میشه که بعنوان DbContext پیش فرض برای دیتابیس عضویت استفاده خواهد شد. اگر موجودیت جدیدی برای پروفایل کاربران بسازید باید بعنوان یک DbSet<T> به این کلاس افزوده بشه. اگر نیازی به تفکیک دیتابیس سیستم عضویت و دیتابیس اپلیکیشن نیست، بهتره از یک DbContext استفاده کنید. لایه بندی مدل ها، سرویس‌ها و کد دسترسی به داده‌ها هم ساده است چرا که کل سیستم توسط EF Code First مدیریت میشه. بنابراین استفاده از الگوهای رایج مانند تزریق وابستگی‌ها و غیره مشابه دیگر سناریوهای EF Code First است.

نویسنده: کامران سادین
تاریخ: ۱۳۹۲/۱۰/۲۹ ۱۷:۱

با سلام.
بنده می‌خواستم بدونم آیا در .NET ورژن 4 هم میتوانیم استفاده کنیم؟
با ویژوال استادیو 2012

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۲۹ ۱۸:۴۰

با نصب پکیج‌های مربوط به ASP.NET Identity و غیرفعال کردن Forms Auth می‌تونید همچین کاری بکنید اما توصیه نمیشه.
سیستم Identity اکثر عملیاتش رو بصورت Async انجام میده که نیاز به NET 4.5. داره. دلایل دیگه ای هم وجود داره که اگر یک جستجوی ساده در اینترنت بکنید مطالب خوبی در این باره پیدا می‌کنید، مثلا لینک زیر:

http://stackoverflow.com/questions/19237285/using-asp-net-identity-in-mvc-4

نویسنده: سوران
تاریخ: ۱۳۹۲/۱۲/۰۱ ۱۲:۳۷

با تشکر از مطالب آموزنده شما ،
من در ارتباط با ارث بری از کلاس IdentityUser یک سوال داشتم. با توجه به نمونه کدهای تمپلیت vs2013 یک کلاس ApplicationUser از کلاس IdentityUser ارث بری می‌کنه و DbContext هم مربوط به این کلاس میشه، یعنی به صورت زیر

```
ApplicationDbContext : IdentityDbContext<ApplicationUser>
```

حال سوال من اینه که اگه چند کلاس داشته باشیم که بخوایم از IdentityUser ارث بری داشته باشند، چطور میشه اونها را در یک DbContext استفاده کرد؟
اگر مقاله یا مثالی در این مورد معرفی کنید ممنون میشم .
با تشکر

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۲/۰۳ ۵:۲۴

کلاس کاربر:

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

    public virtual UserProfile Profile { get; set; }
}
```

کلاس پروفایل کاربر:

```
public class UserProfile
{
    public int Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }

    public DateTime? Birthday { get; set; }

    public byte[] Avatar { get; set; }
}
```

کلاس کانتکست دیتابیس:

```
public class SampleDbContext : IdentityDbContext
{
    public SampleDbContext() : base("DefaultConnection") { }

    static SampleDbContext()
    {
        Database.SetInitializer(new DropCreateDatabaseIfModelChanges<SampleDbContext>());
    }

    public DbSet<UserProfile> UserProfiles { get; set; }
    public DbSet<Customer> Customers { get; set; }
    public DbSet<Product> Products { get; set; }
    ...
}
```

این کلاس‌ها می‌تونن تو لایه دیگری مثل Domain Models تعریف بشن.

نویسنده: رضا گرمارودی
تاریخ: ۱۳۹۲/۱۲/۰۳ ۹:۲۱

سلام؛ Identity در کار با SQLCE مشکل داره! هنگام چک کردن نام کاربری و کلمه عبور پیغامی مبنی بر استفاده از تابع Lowercase میدهد که گویا در SqlCe به شیوه دیگری باید صدا زده شود !

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۰۳ ۹:۳۶

این پروژه سورس باز هست. مشکلات آن‌را برای رفع در اینجا گزارش کنید. نحوه‌ی گزارش مشکل هم باید کمی فنی باشد. حداقل جزئیات exception و stack trace آن باید گزارش شوند.

نویسنده: احمد
تاریخ: ۱۳۹۳/۰۲/۲۳ ۱۰:۲۸

سلام
آیا در پروژه‌های windows application که از wcf استفاده میکنند هم میتوانیم از این سیستم استفاده کنیم.؟

نویسنده: داریوش حمیدی
تاریخ: ۱۳۹۳/۰۷/۱۸ ۲۰:۴

سلام ; در کلاس IdentityUser این خصوصیت دو به چی اشاره دارند ؟

1-SecurityStamp

این مقدار Random تغیر می‌کنید وقتی که اطلاعات کاربری تغیر پیدا می‌کند مثل تغیر رمز عبور ..

2-TwoFactorEnabled

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۲۹ ۱۳:۴۶

« اعمال تزریق وابستگی‌ها به مثال رسمی ASP.NET Identity »