عنوان: **انتشار پیش نمایش ASP.NET Identity 2.0.0-alpha1**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۲۳ ۱۶:۳۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MemberShip, Visual Studio 2013, ASP.NET Identity

مایکروسافت در تاریخ 20 دسامبر 2013 پیش نمایش نسخه جدید ASP.NET Identity را معرفی کرد. تمرکز اصلی در این انتشار، رفع مشکلات نسخه 1.0 بود. امکانات جدیدی هم مانند Account Confirmation و Password Reset اضافه شده اند.

دانلود این انتشار

ASP.NET Identity را می‌توانید در قالب یک پکیج NuGet دریافت کنید. در پنجره Manage NuGet Packages می‌توانید پکیج‌های Preview را لیست کرده و گزینه مورد نظر را

نصب کنید. برای نصب پکیج‌های pre-release توسط Package Manager Console از فرامین زیر استفاده کنید.

```
Install-Package Microsoft.AspNet.Identity.EntityFramework -Version 2.0.0-alpha1 -Pre
Install-Package Microsoft.AspNet.Identity.Core -Version 2.0.0-alpha1 -Pre
Install-Package Microsoft.AspNet.Identity.OWIN -Version 2.0.0-alpha1 -Pre
```

دقت کنید که حتما از گزینه "Include Prerelease" استفاده می‌کنید. برای اطلاعات بیشتر درباره نصب پکیج‌های Pre-release لطفا به این لینک و یا این لینک مراجعه کنید.

در ادامه لیست امکانات جدید و مشکلات رفع شده را می‌خوانید.

Account Confirmation

سیستم ASP.NET Identity حالا از Account Confirmation پشتیبانی می‌کند. این یک سناریوی بسیار رایج است. در اکثر وب سایت‌های امروزی پس از ثبت نام، حتما باید ایمیل خود را تایید کنید. پیش از تایید ثبت نام قادر به انجام هیچ کاری در وب سایت نخواهید بود، یعنی نمی‌توانید Login کنید. این روش مفید است، چرا که از ایجاد حساب‌های کاربری نامعتبر (bogus) جلوگیری می‌کند. همچنین این روش برای برقراری ارتباط با کاربران هم بسیار کارآمد است. از آدرس‌های ایمیل کاربران می‌توانید در وب سایت‌های فروم، شبکه‌های اجتماعی، تجارت آنلاین و بانکداری برای اطلاع رسانی و دیگر موارد استفاده کنید.

نکته: برای ارسال ایمیل باید تنظیمات SMTP را پیکربندی کنید. مثلا می‌توانید از سرویس‌های ایمیل محبوبی مانند SendGrid استفاده کنید، که با Windows Azure براحتی یکپارچه می‌شود و از طرف توسعه دهنده اپلیکیشن هم نیاز به پیکربندی ندارد.

در مثال زیر نیاز دارید تا یک سرویس ایمیل برای ارسال ایمیل‌ها پیکربندی کنید. همچنین کاربران پیش از تایید ایمیل شان قادر به بازنشانی کلمه عبور نیستند.

Password Reset

این هم یک سناریوی رایج و استاندارد است. کاربران در صورتی که کلمه عبورشان را فراموش کنند، می‌توانند از این قابلیت برای بازنشانی آن استفاده کنند. کلمه عبور جدیدی بصورت خودکار تولید شده و برای آنها ارسال می‌شود. کاربران با استفاده از این رمز عبور جدید می‌توانند وارد سایت شوند و سپس آن را تغییر دهند.

Security Token Provider

هنگامی که کاربران کلمه عبورشان را تغییر می‌دهند، یا اطلاعات امنیتی خود را بروز رسانی می‌کنند (مثلا حذف کردن لاگین‌های خارجی مثل فیسبوک، گوگل و غیره) باید شناسه امنیتی (security token) کاربر را بازتولید کنیم و مقدار قبلی را Invalidate یا بی اعتبار سازیم. این کار بمنظور حصول اطمینان از بی اعتبار بودن تمام شناسه‌های قبلی است که توسط کلمه عبور پیشین تولید شده بودند. این قابلیت، یک لایه امنیتی بیشتر برای اپلیکیشن شما فراهم می‌کند. چرا که وقتی کاربری کلمه عبورش را تغییر بدهد از همه جا logged-out می‌شود. یعنی از تمام مرورگرهایی که برای استفاده از اپلیکیشن استفاده کرده خارج خواهد شد.

برای پیکربندی تنظیمات این قابلیت می‌توانید از فایل Startup.Auth.cs استفاده کنید. می‌توانید مشخص کنید که میان افزار OWIN cookie هر چند وقت یکبار باید شناسه امنیتی کاربران را بررسی کند. به لیست زیر دقت کنید.

```
// Enable the application to use a cookie to store information for the signed in user
// and to use a cookie to temporarily store information about a user logging in with a third party
login provider
// Configure the sign in cookie
app.UseCookieAuthentication(newCookieAuthenticationOptions {
    AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
    LoginPath = newPathString("/Account/Login"),
    Provider = newCookieAuthenticationProvider {
        OnValidateIdentity = SecurityStampValidator.OnValidateIdentity<ApplicationUserManager,
ApplicationUser>(
```

```
            validateInterval: TimeSpan.FromSeconds(5),
            regenerateIdentity: (manager, user) => user.GenerateUserIdentityAsync(manager))
    }
});
```

امکان سفارشی کردن کلیدهای اصلی Users و Roles

در نسخه 1.0 نوع فیلدهای کلید اصلی در جداول Users و Roles از نوع رشته (string) بود. این بدین معنا است که وقتی از Entity Framework و Sql Server برای ذخیره داده‌های ASP.NET Identity استفاده می‌کنیم داده‌های این فیلدها بعنوان nvarchar ذخیره می‌شوند. درباره این پیاده سازی پیش فرض در فروم هایی مانند سایت StackOverflow بسیار بحث شده است. و در آخر با در نظر گرفتن تمام بازخورد ها، تصمیم گرفته شد یک نقطه توسعه پذیری (extensibility) اضافه شود که توسط آن بتوان نوع فیلدهای اصلی را مشخص کرد. مثلا شاید بخواهید کلیدهای اصلی جداول Users و Roles از نوع int باشند. این نقطه توسعه پذیری مخصوصا هنگام مهاجرت داده‌های قبلی بسیار مفید است، مثلا ممکن است دیتابیس قبلی فیلدهای UserId را با فرمت GUID ذخیره کرده باشد.

اگر نوع فیلدهای کلید اصلی را تغییر دهید، باید کلاس‌های مورد نیاز برای Claims و Logins را هم اضافه کنید تا کلید اصلی معتبری دریافت کنند. قطعه کد زیر نمونه ای از نحوه استفاده این قابلیت برای تعریف کلیدهای int را نشان می‌دهد.

```
de Snippet
publicclassApplicationUser : IdentityUser<int, CustomUserLogin, CustomUserRole, CustomUserClaim>
{
}
publicclassCustomRole : IdentityRole<int, CustomUserRole>
{
    public CustomRole() { }
    public CustomRole(string name) { Name = name; }
}
publicclassCustomUserRole : IdentityUserRole<int> { }
publicclassCustomUserClaim : IdentityUserClaim<int> { }
publicclassCustomUserLogin : IdentityUserLogin<int> { }

publicclassApplicationDbContext : IdentityDbContext<ApplicationUser, CustomRole, int, CustomUserLogin,
CustomUserRole, CustomUserClaim>
{
}
```

پشتیبانی از IQueryable روی Users و Roles

کلاس‌های UserStore و RoleStore حالا از IQueryable پشتیبانی می‌کنند، بنابراین می‌توانید براحتی لیست کاربران و نقش‌ها را کوئری کنید.

بعنوان مثال قطعه کد زیر دریافت لیست کاربران را نشان می‌دهد. از همین روش برای دریافت لیست نقش‌ها از RoleManager می‌توانید استفاده کنید.

```
//
// GET: /Users/
public async Task<ActionResult> Index()
{
    return View(await UserManager.Users.ToListAsync());
}
```

پشتیبانی از عملیات Delete از طریق UserManager

در نسخه 1.0 اگر قصد حذف یک کاربر را داشتید، نمی‌توانستید این کار را از طریق UserManager انجام دهید. اما حالا می‌توانید مانند قطعه کد زیر عمل کنید.

```
var user = await UserManager.FindByIdAsync(id);

if (user == null)
{
    return HttpNotFound();
```

```
}
var result = await UserManager.DeleteAsync(user);
```

میان افزار UserManagerFactory

شما می‌توانید با استفاده از یک پیاده سازی Factory، وهله ای از UserManager را از OWIN context دریافت کنید. این الگو مشابه چیزی است که برای گرفتن AuthenticationManager در OWIN context استفاده می‌کنیم. این الگو همچنین روش توصیه شده برای گرفتن یک نمونه از UserManager به ازای هر درخواست در اپلیکیشن است.

قطعه کد زیر نحوه پیکربندی این میان افزار در فایل StartupAuth.cs را نشان می‌دهد.

```
// Configure the UserManager
app.UseUserManagerFactory(newUserManagerOptions<ApplicationUserManager>()
{
    DataProtectionProvider = app.GetDataProtectionProvider(),
    Provider = newUserManagerProvider<ApplicationUserManager>()
    {
        OnCreate = ApplicationUserManager.Create
    }
});
```

و برای گرفتن یک وهله از UserManager:

```
HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();
```

میان افزار DbContextFactory

سیستم ASP.NET Identity از Entity Framework برای ذخیره داده هایش در Sql Server استفاده می‌کند. بدین منظور، ASP.NET Identity کلاس ApplicationDbContext را رفرنس می‌کند. میان افزار DbContextFactory به ازای هر درخواست در اپلیکیشن یک وهله از ApplicationDbContext را به شما تحویل می‌دهد.

می توانید پیکربندی لازم را در StartupAuth.cs انجام دهید.

```
app.UseDbContextFactory(ApplicationDbContext.Create);
```

Samples

امکانات جدید را می‌توانید در پروژه https://aspnet.codeplex.com پیدا کنید. لطفا به پوشه Identity در سورس کد مراجعه کنید. برای اطلاعاتی درباره نحوه اجرای پروژه هم فایل readme را بخوانید.

برای مستندات ASP.NET Identity 1.0 هم به http://www.asp.net/identity سر بزنید. هنوز مستنداتی برای نسخه 2.0 منتشر نشده، اما بزودی با انتشار نسخه نهایی مستندات و مثال‌های جدیدی به سایت اضافه خواهند شد.

Known Issues

در کنار قابلیت‌های جدیدی مانند Account Confirmation و Password Reset، دو خاصیت جدید به کلاس IdentityUser اضافه شده اند: 'Email' و 'IsConfirmed'. این تغییرات الگوی دیتابیسی که توسط ASP.NET Identity 1.0 ساخته شده است را تغییر می‌دهد. بروز رسانی پکیج‌ها از نسخه 1.0 به 2.0 باعث می‌شود که اپلیکیشن شما دیگر قادر به دسترسی به دیتابیس عضویت نباشد، چرا که مدل دیتابیس تغییر کرده. برای بروز رسانی الگوی دیتابیس می‌توانید از Code First Migrations استفاده کنید.

نکته: نسخه جدید به EntityFramework 6.1.0-alpha1 وابستگی دارد، که در همین تاریخ (20 دسامبر 2013) پیش نمایش شد. http://blogs.msdn.com/b/adonet/archive/2013/12/20/ef-6-1-alpha-1-available.aspx

EntityFramework 6.1.0-alpha1 بروز رسانی هایی دارد که سناریوی مهاجرت در ASP.NET Identity را تسهیل می‌کند، به همین دلیل از نسخه جدید EF استفاده شده. تیم ASP.NET هنوز باگ‌های زیادی را باید رفع کند و قابلیت‌های جدیدی را هم باید پیاده سازی کند. بنابراین پیش از نسخه نهایی RTM شاهد پیش نمایش‌های دیگری هم خواهیم بود که در ماه‌های آتی منتشر می‌شوند.

برای اطلاعات بیشتر درباره آینده ASP.NET Identity به لینک زیر سری بزنید.

https://aspnetidentity.codeplex.com/wikipage?title=Roadmap&version=1

نظرات خوانندگان

نویسنده: ابوالفضل رجب پور
تاریخ: ۱۳۹۲/۱۰/۳۰ ۲۰:۴۴

سلام

پیاده سازی Single Sign on در این سیستم کجا کار قرار داره؟ در واقع چطور میشه پیاده سازی ش کرد؟
در سیستم membership قبلی، اگر کلید اپلیکیشن رو در وب کانفیگ برنامه هاتون که دامین هاشون مشترک بود (در واقع ساب دامین ها)، یکسان وارد میکردی، برنامه‌ها بصورت SSO کار می‌کرد و احتیاجی به هیچ کاری نداشت. حالا در سیستم جدید همون روش جواب میده؟ برای برنامه‌های با دامین‌های متفاوت چطور؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۳۰ ۲۳:۷

CookieDomain رو باید تنظیم کنید.