عنوان: **Identity 2.0 : تایید حساب های کاربری و احراز هویت دو مرحله ای**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۳/۰۲/۱۰ ۱:۳۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security, ASP.NET Identity

در پست قبلی نگاهی اجمالی به انتشار نسخه جدید Identity Framework داشتیم. نسخه جدید تغییرات چشمگیری را در فریم ورک بوجود آورده و قابلیت‌های جدیدی نیز عرضه شده‌اند. دو مورد از این قابلیت‌ها که پیشتر بسیار درخواست شده بود، تایید حساب‌های کاربری (Account Validation) و احراز هویت دو مرحله ای (Two-Factor Authorization) بود. در این پست راه اندازی این دو قابلیت را بررسی می‌کنیم.

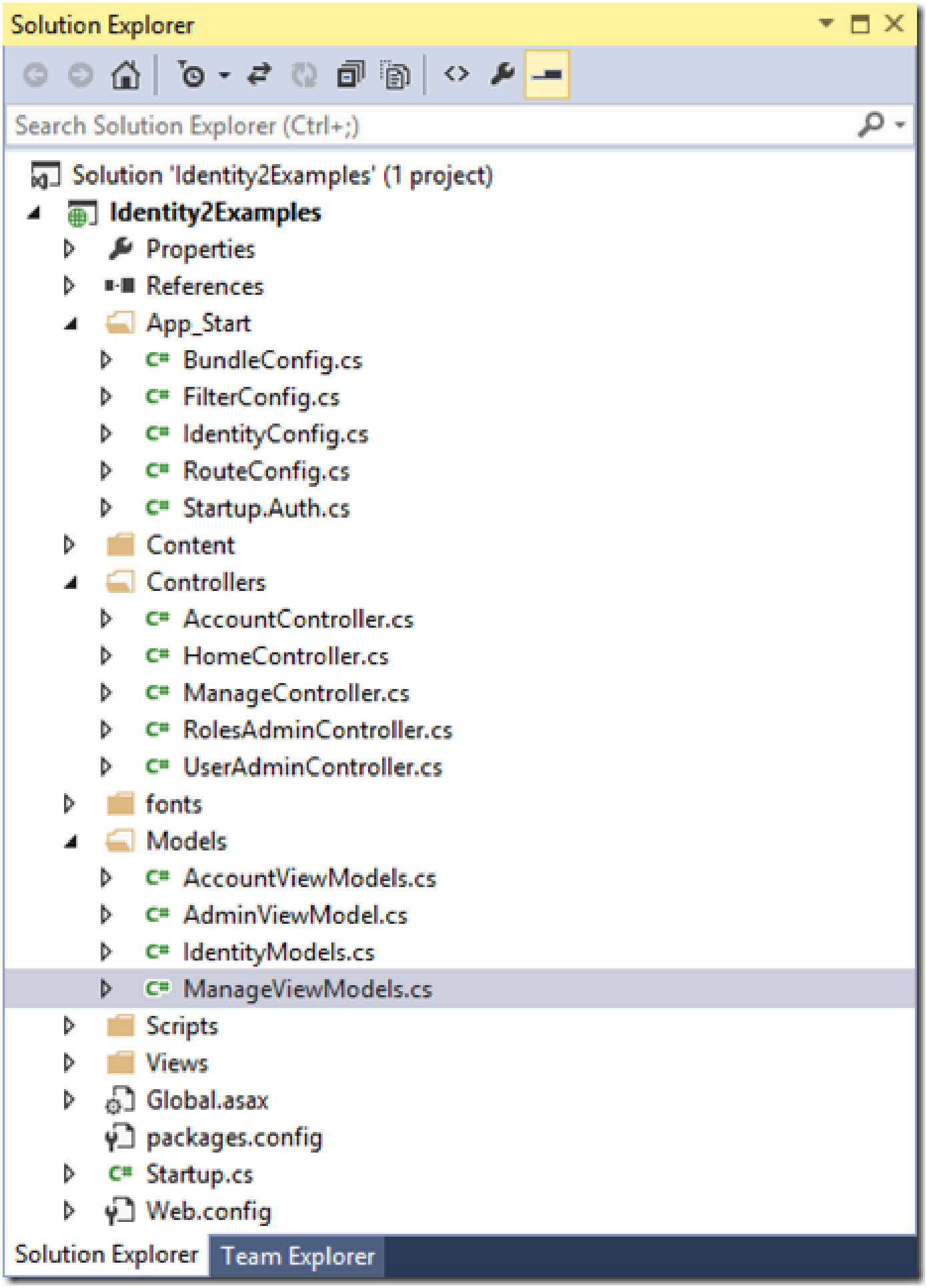
تیم ASP.NET Identity پروژه نمونه ای را فراهم کرده است که می‌تواند بعنوان نقطه شروعی برای اپلیکیشن‌های MVC استفاده شود. پیکربندی‌های لازم در این پروژه انجام شده‌اند و برای استفاده از فریم ورک جدید آماده است.

شروع به کار : پروژه نمونه را توسط NuGet ایجاد کنید

برای شروع یک پروژه ASP.NET خالی ایجاد کنید (در دیالوگ قالب‌ها گزینه Empty را انتخاب کنید). سپس کنسول Package Manager را باز کرده و دستور زیر را اجرا کنید.

```
PM> Install-Package Microsoft.AspNet.Identity.Samples -Pre
```

پس از اینکه NuGet کارش را به اتمام رساند باید پروژه ای با ساختار متداول پروژه‌های ASP.NET MVC داشته باشید. به تصویر زیر دقت کنید.


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Identity2Examples' (1 project)
Identity2Examples
Properties
References
App_Start
BundleConfig.cs
FilterConfig.cs
IdentityConfig.cs
RouteConfig.cs
Startup.Auth.cs
Content
Controllers
AccountController.cs
HomeController.cs
ManageController.cs
RolesAdminController.cs
UserAdminController.cs
fonts
Models
AccountViewModels.cs
AdminViewModel.cs
IdentityModels.cs
ManageViewModels.cs
Scripts
Views
Global.asax
packages.config
Startup.cs
Web.config
Solution Explorer
Team Explorer

همانطور که می‌بینید ساختار پروژه بسیار مشابه پروژه‌های معمول MVC است، اما آیتم‌های جدیدی نیز وجود دارند. فعلا تمرکز اصلی ما روی فایل *IdentityConfig.cs* است که در پوشه *App_Start* قرار دارد.

اگر فایل مذکور را باز کنید و کمی اسکرول کنید تعاریف دو کلاس سرویس را مشاهده می‌کنید: EmailService و SmsService.

```
public class EmailService : IIdentityMessageService
{
    public Task SendAsync(IdentityMessage message)
    {
        // Plug in your email service here to send an email.
        return Task.FromResult(0);
    }
}

public class SmsService : IIdentityMessageService
{
    public Task SendAsync(IdentityMessage message)
    {
        // Plug in your sms service here to send a text message.
        return Task.FromResult(0);
    }
}
```

اگر دقت کنید هر دو کلاس قرارداد IIdentityMessageService را پیاده سازی می‌کنند. می‌توانید از این قرارداد برای پیاده سازی سرویس‌های اطلاع رسانی ایمیلی، پیامکی و غیره استفاده کنید. در ادامه خواهیم دید چگونه این دو سرویس را بسط دهیم.

یک حساب کاربری مدیریتی پیش فرض ایجاد کنید

پیش از آنکه بیشتر جلو رویم نیاز به یک حساب کاربری در نقش مدیریتی داریم تا با اجرای اولیه اپلیکیشن در دسترس باشد. کلاسی بنام ApplicationDbInitializer در همین فایل وجود دارد که هنگام اجرای اولیه و یا تشخیص تغییرات در مدل دیتابیس، اطلاعاتی را Seed می‌کند.

```
public class ApplicationDbInitializer
    : DropCreateDatabaseIfModelChanges<ApplicationDbContext>
{
    protected override void Seed(ApplicationDbContext context) {
        InitializeIdentityForEF(context);
        base.Seed(context);
    }

    //Create User=Admin@Admin.com with password=Admin@123456 in the Admin role
    public static void InitializeIdentityForEF(ApplicationDbContext db)
    {
        var userManager =
           HttpContext.Current.GetOwinContext().GetUserManager<ApplicationUserManager>();

        var roleManager =
            HttpContext.Current.GetOwinContext().Get<ApplicationRoleManager>();

        const string name = "admin@admin.com";
        const string password = "Admin@123456";
        const string roleName = "Admin";

        //Create Role Admin if it does not exist
        var role = roleManager.FindByName(roleName);

        if (role == null) {
            role = new IdentityRole(roleName);
            var roleresult = roleManager.Create(role);
        }

        var user = userManager.FindByName(name);

        if (user == null) {
            user = new ApplicationUser { UserName = name, Email = name };
            var result = userManager.Create(user, password);
```

```
            result = userManager.SetLockoutEnabled(user.Id, false);
        }

        // Add user admin to Role Admin if not already added
        var rolesForUser = userManager.GetRoles(user.Id);

        if (!rolesForUser.Contains(role.Name)) {
            var result = userManager.AddToRole(user.Id, role.Name);
        }
    }
}
```

همانطور که می‌بینید این قطعه کد ابتدا نقشی بنام Admin می‌سازد. سپس حساب کاربری ای، با اطلاعاتی پیش فرض ایجاد شده و بدین نقش منتسب می‌گردد. اطلاعات کاربر را به دلخواه تغییر دهید و ترجیحا از یک آدرس ایمیل زنده برای آن استفاده کنید.

تایید حساب‌های کاربری : چگونه کار می‌کند

بدون شک با تایید حساب‌های کاربری توسط ایمیل آشنا هستید. حساب کاربری ای ایجاد می‌کنید و ایمیلی به آدرس شما ارسال می‌شود که حاوی لینک فعالسازی است. با کلیک کردن این لینک حساب کاربری شما تایید شده و می‌توانید به سایت وارد شوید.

اگر به کنترلر AccountController در این پروژه نمونه مراجعه کنید متد Register را مانند لیست زیر می‌یابید.

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = new ApplicationUser { UserName = model.Email, Email = model.Email };
        var result = await UserManager.CreateAsync(user, model.Password);

        if (result.Succeeded)
        {
            var code = await UserManager.GenerateEmailConfirmationTokenAsync(user.Id);

            var callbackUrl = Url.Action(
                "ConfirmEmail",
                "Account",
                new { userId = user.Id, code = code },
                protocol: Request.Url.Scheme);

            await UserManager.SendEmailAsync(
                user.Id,
                "Confirm your account",
                "Please confirm your account by clicking this link: <a href=\""
                + callbackUrl + "\">link</a>");

            ViewBag.Link = callbackUrl;

            return View("DisplayEmail");
        }
        AddErrors(result);
    }

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

اگر به قطعه کد بالا دقت کنید فراخوانی متد UserManager.SendEmailAsync را می‌یابید که آرگومانهایی را به آن پاس می‌دهیم. در کنترلر جاری، آبجکت UserManager یک خاصیت (Property) است که وهله ای از نوع ApplicationUserManager را باز می‌گرداند. اگر به فایل *IdentityConfig.cs* مراجعه کنید تعاریف این کلاس را خواهید یافت. در این کلاس، متد استاتیک ()Create وهله ای از ApplicationUserManager را می‌سازد که در همین مرحله سرویس‌های پیام رسانی پیکربندی می‌شوند.

```
public static ApplicationUserManager Create(
    IdentityFactoryOptions<ApplicationUserManager> options,
    IOwinContext context)
```

```
{
    var manager = new ApplicationUserManager(
        new UserStore<ApplicationUser>(
            context.Get<ApplicationDbContext>()));

    // Configure validation logic for usernames
    manager.UserValidator = new UserValidator<ApplicationUser>(manager)
    {
        AllowOnlyAlphanumericUserNames = false,
        RequireUniqueEmail = true
    };

    // Configure validation logic for passwords
    manager.PasswordValidator = new PasswordValidator
    {
        RequiredLength = 6,
        RequireNonLetterOrDigit = true,
        RequireDigit = true,
        RequireLowercase = true,
        RequireUppercase = true,
    };

    // Configure user lockout defaults
    manager.UserLockoutEnabledByDefault = true;
    manager.DefaultAccountLockoutTimeSpan = TimeSpan.FromMinutes(5);
    manager.MaxFailedAccessAttemptsBeforeLockout = 5;

    // Register two factor authentication providers. This application
    // uses Phone and Emails as a step of receiving a code for verifying the user
    // You can write your own provider and plug in here.
    manager.RegisterTwoFactorProvider(
        "PhoneCode",
        new PhoneNumberTokenProvider<ApplicationUser>
    {
        MessageFormat = "Your security code is: {0}"
    });

    manager.RegisterTwoFactorProvider(
        "EmailCode",
        new EmailTokenProvider<ApplicationUser>
    {
        Subject = "SecurityCode",
        BodyFormat = "Your security code is {0}"
    });

    manager.EmailService = new EmailService();
    manager.SmsService = new SmsService();

    var dataProtectionProvider = options.DataProtectionProvider;

    if (dataProtectionProvider != null)
    {
        manager.UserTokenProvider =
            new DataProtectorTokenProvider<ApplicationUser>(
                dataProtectionProvider.Create("ASP.NET Identity"));
    }

    return manager;
}
```

در قطعه کد بالا کلاس‌های EmailService و SmsService روی وهله ApplicationUserManager تنظیم می‌شوند.

```
manager.EmailService = new EmailService();
manager.SmsService = new SmsService();
```

درست در بالای این کدها می‌بینید که چگونه تامین کنندگان احراز هویت دو مرحله ای (مبتنی بر ایمیل و پیامک) رجیستر می‌شوند.

```
// Register two factor authentication providers. This application
// uses Phone and Emails as a step of receiving a code for verifying the user
// You can write your own provider and plug in here.
manager.RegisterTwoFactorProvider(
    "PhoneCode",
```

```
    new PhoneNumberTokenProvider<ApplicationUser>
{
    MessageFormat = "Your security code is: {0}"
});

manager.RegisterTwoFactorProvider(
    "EmailCode",
    new EmailTokenProvider<ApplicationUser>
    {
        Subject = "SecurityCode",
        BodyFormat = "Your security code is {0}"
    });
```

تایید حساب‌های کاربری توسط ایمیل و احراز هویت دو مرحله ای توسط ایمیل و/یا پیامک نیاز به پیاده سازی هایی معتبر از قراردارد IIdentityMessageService دارند.

پیاده سازی سرویس ایمیل توسط ایمیل خودتان

پیاده سازی سرویس ایمیل نسبتا کار ساده ای است. برای ارسال ایمیل‌ها می‌توانید از اکانت ایمیل خود و یا سرویس هایی مانند SendGrid استفاده کنید. بعنوان مثال اگر بخواهیم سرویس ایمیل را طوری پیکربندی کنیم که از یک حساب کاربری Outlook استفاده کند، مانند زیر عمل خواهیم کرد.

```
public class EmailService : IIdentityMessageService
{
    public Task SendAsync(IdentityMessage message)
    {
        // Credentials:
        var credentialUserName = "yourAccount@outlook.com";
        var sentFrom = "yourAccount@outlook.com";
        var pwd = "yourApssword";

        // Configure the client:
        System.Net.Mail.SmtpClient client =
            new System.Net.Mail.SmtpClient("smtp-mail.outlook.com");

        client.Port = 587;
        client.DeliveryMethod = System.Net.Mail.SmtpDeliveryMethod.Network;
        client.UseDefaultCredentials = false;

        // Creatte the credentials:
        System.Net.NetworkCredential credentials =
            new System.Net.NetworkCredential(credentialUserName, pwd);

        client.EnableSsl = true;
        client.Credentials = credentials;

        // Create the message:
        var mail =
            new System.Net.Mail.MailMessage(sentFrom, message.Destination);

        mail.Subject = message.Subject;
        mail.Body = message.Body;

        // Send:
        return client.SendMailAsync(mail);
    }
}
```

تنظیمات SMTP میزبان شما ممکن است متفاوت باشد اما مطمئنا می‌توانید مستندات لازم را پیدا کنید. اما در کل رویکرد مشابهی خواهید داشت.

پیاده سازی سرویس ایمیل با استفاده از SendGrid

سرویس‌های ایمیل متعددی وجود دارند اما یکی از گزینه‌های محبوب در جامعه دات نت SendGrid است. این سرویس API قدرتمندی برای زبان‌های برنامه نویسی مختلف فراهم کرده است. همچنین یک Web API مبتنی بر HTTP نیز در دسترس است. قابلیت دیگر اینکه این سرویس مستقیما با Windows Azure یکپارچه می‌شود.

می توانید در سایت SendGrid یک حساب کاربری رایگان بعنوان توسعه دهنده بسازید. پس از آن پیکربندی سرویس ایمیل با مرحله قبل تفاوت چندانی نخواهد داشت. پس از ایجاد حساب کاربری توسط تیم پشتیبانی SendGrid با شما تماس گرفته خواهد شد تا از صحت اطلاعات شما اطمینان حاصل شود. برای اینکار چند گزینه در اختیار دارید که بهترین آنها ایجاد یک اکانت ایمیل در دامنه وب سایتتان است. مثلا اگر هنگام ثبت نام آدرس وب سایت خود را www.yourwebsite.com وارد کرده باشید، باید ایمیلی مانند info@yourwebsite.com ایجاد کنید و توسط ایمیل فعالسازی آن را تایید کند تا تیم پشتیبانی مطمئن شود صاحب امتیاز این دامنه خودتان هستید.

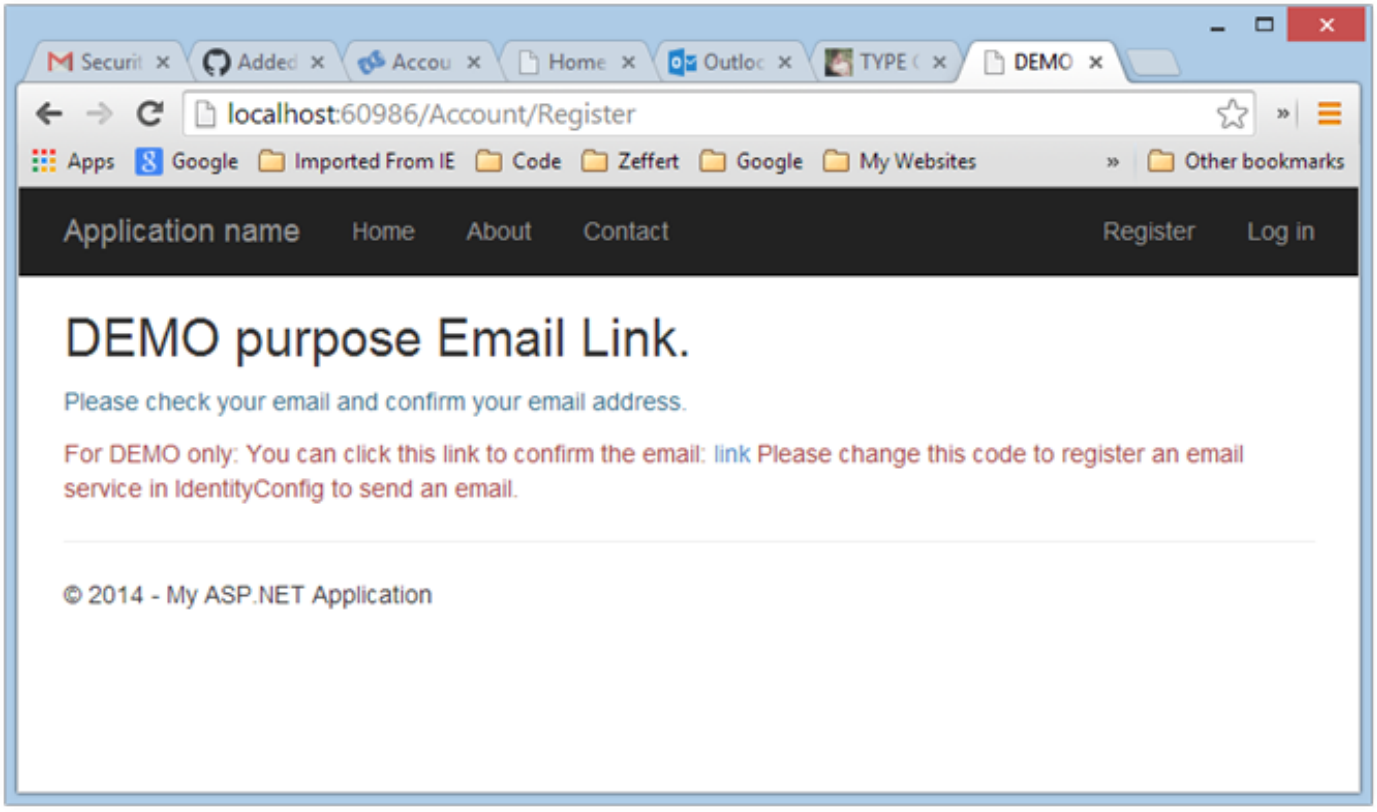
تنها چیزی که در قطعه کد بالا باید تغییر کند اطلاعات حساب کاربری و تنظیمات SMTP است. توجه داشته باشید که نام کاربری و آدرس فرستنده در اینجا متفاوت هستند. در واقع می‌توانید از هر آدرسی بعنوان آدرس فرستنده استفاده کنید.

```
public class EmailService : IIdentityMessageService
{
    public Task SendAsync(IdentityMessage message)
    {
        // Credentials:
        var sendGridUserName = "yourSendGridUserName";
        var sentFrom = "whateverEmailAdressYouWant";
        var sendGridPassword = "YourSendGridPassword";

        // Configure the client:
        var client =
            new System.Net.Mail.SmtpClient("smtp.sendgrid.net", Convert.ToInt32(587));

        client.Port = 587;
        client.DeliveryMethod = System.Net.Mail.SmtpDeliveryMethod.Network;
        client.UseDefaultCredentials = false;

        // Creatte the credentials:
        System.Net.NetworkCredential credentials =
            new System.Net.NetworkCredential(credentialUserName, pwd);

        client.EnableSsl = true;
        client.Credentials = credentials;

        // Create the message:
        var mail =
            new System.Net.Mail.MailMessage(sentFrom, message.Destination);

        mail.Subject = message.Subject;
        mail.Body = message.Body;

        // Send:
        return client.SendMailAsync(mail);
    }
}
```

حال می‌توانیم سرویس ایمیل را تست کنیم.

آزمایش تایید حساب‌های کاربری توسط سرویس ایمیل

ابتدا اپلیکیشن را اجرا کنید و سعی کنید یک حساب کاربری جدید ثبت کنید. دقت کنید که از آدرس ایمیلی زنده که به آن دسترسی دارید استفاده کنید. اگر همه چیز بدرستی کار کند باید به صفحه ای مانند تصویر زیر هدایت شوید.

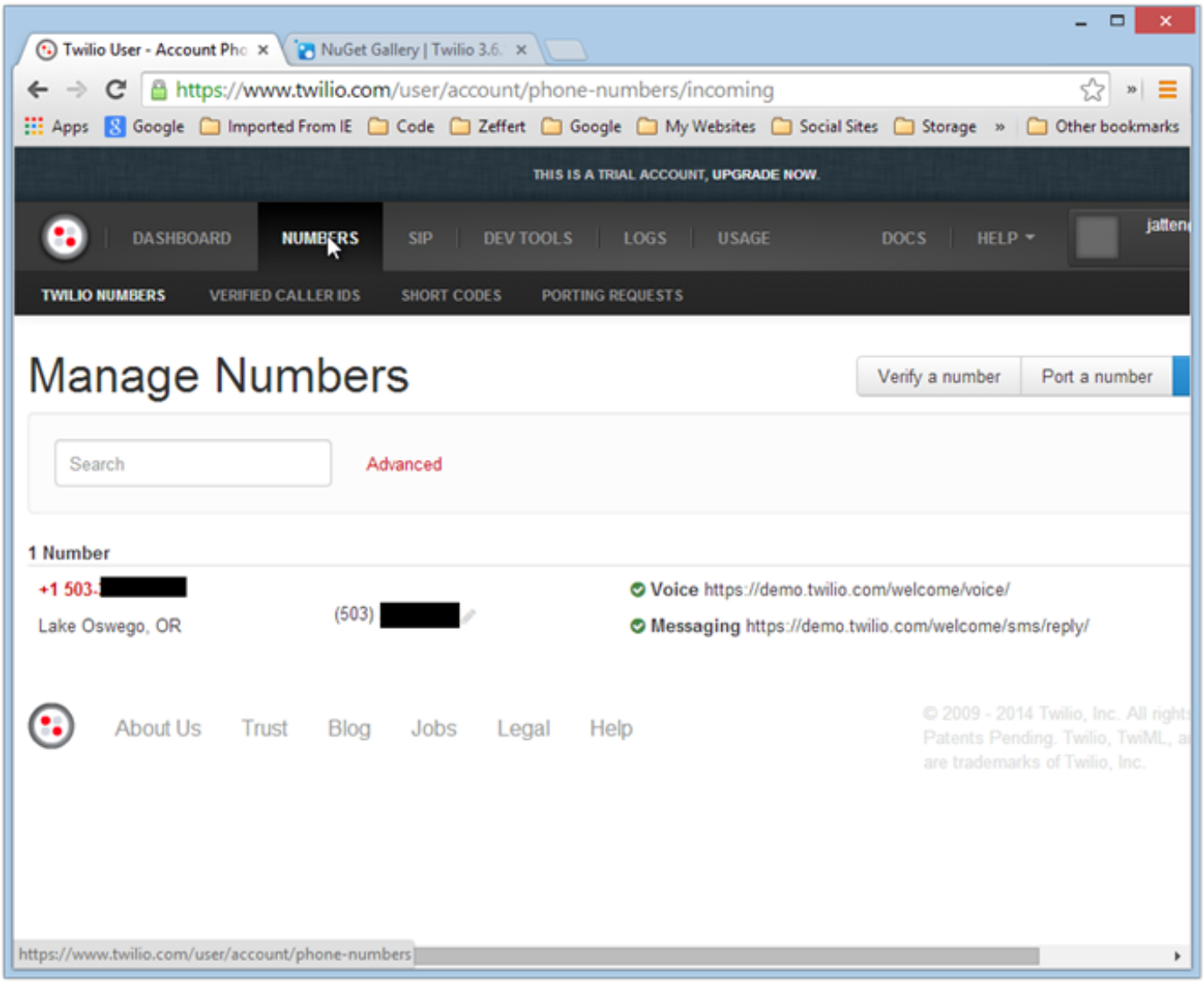


همانطور که مشاهده می‌کنید پاراگرافی در این صفحه وجود دارد که شامل لینک فعالسازی است. این لینک صرفا جهت تسهیل کار توسعه دهندگان درج می‌شود و هنگام توزیع اپلیکیشن باید آن را حذف کنید. در ادامه به این قسمت باز می‌گردیم. در این مرحله ایمیلی حاوی لینک فعالسازی باید برای شما ارسال شده باشد.

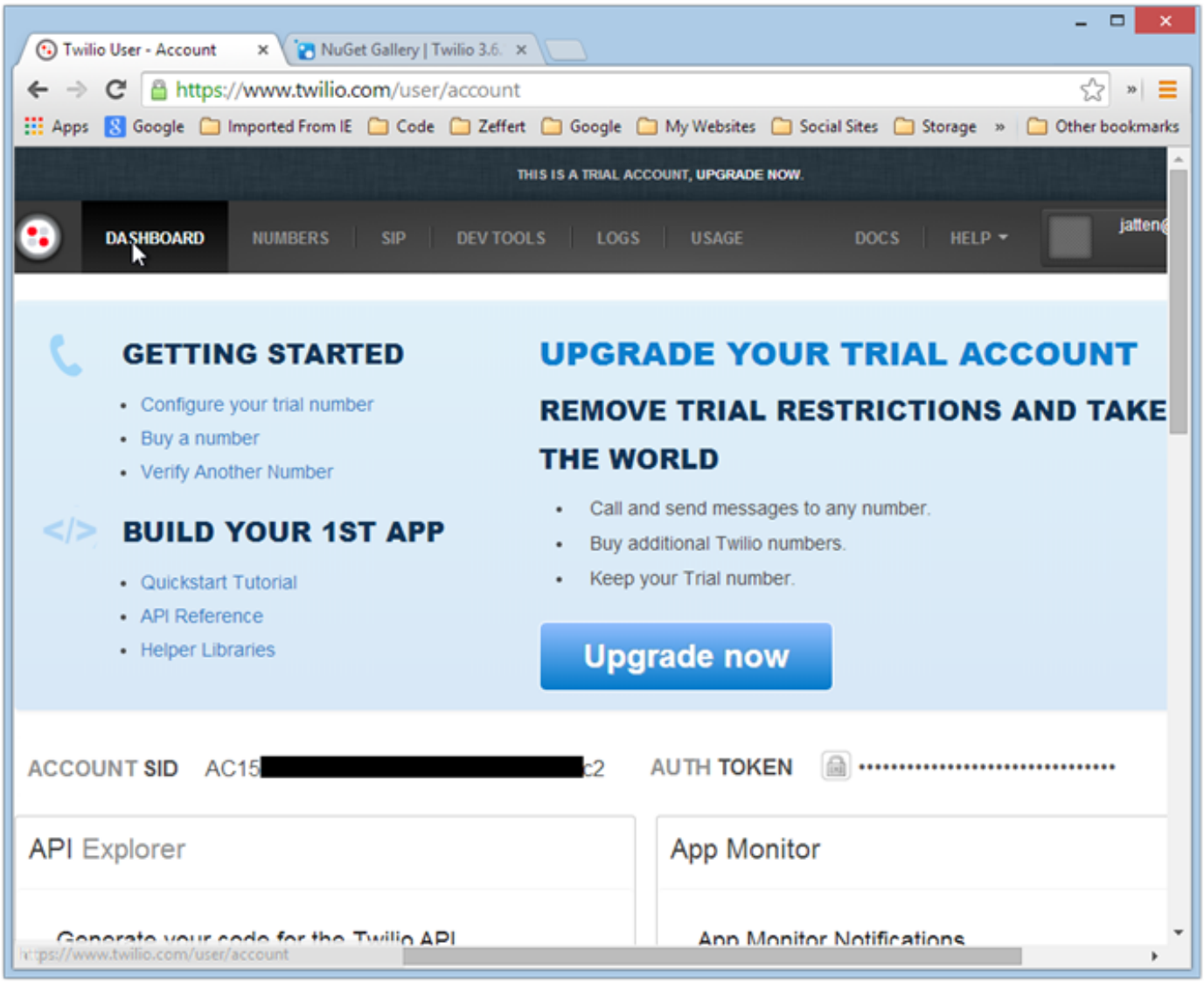
پیاده سازی سرویس SMS

برای استفاده از احراز هویت دو مرحله ای پیامکی نیاز به یک فراهم کننده SMS دارید، مانند Twilio . مانند SendGrid این سرویس نیز در جامعه دات نت بسیار محبوب است و یک C# API قدرتمند ارائه می‌کند. می‌توانید حساب کاربری رایگانی بسازید و شروع به کار کنید.

پس از ایجاد حساب کاربری یک شماره SMS، یک شناسه SID و یک شناسه Auth Token به شما داده می‌شود. شماره پیامکی خود را می‌توانید پس از ورود به سایت و پیمایش به صفحه Numbers مشاهده کنید.



شناسه‌های SID و Auth Token نیز در صفحه Dashboard قابل مشاهده هستند.



اگر دقت کنید کنار شناسه Auth Token یک آیکون قفل وجود دارد که با کلیک کردن روی آن شناسه مورد نظر نمایان می‌شود.

حال می‌توانید از سرویس Twilio در اپلیکیشن خود استفاده کنید. ابتدا بسته NuGet مورد نیاز را نصب کنید.

```
PM> Install-Package Twilio
```

پس از آن فضای نام Twilio را به بالای فایل *IdentityConfig.cs* اضافه کنید و سرویس پیامک را پیاده سازی کنید.

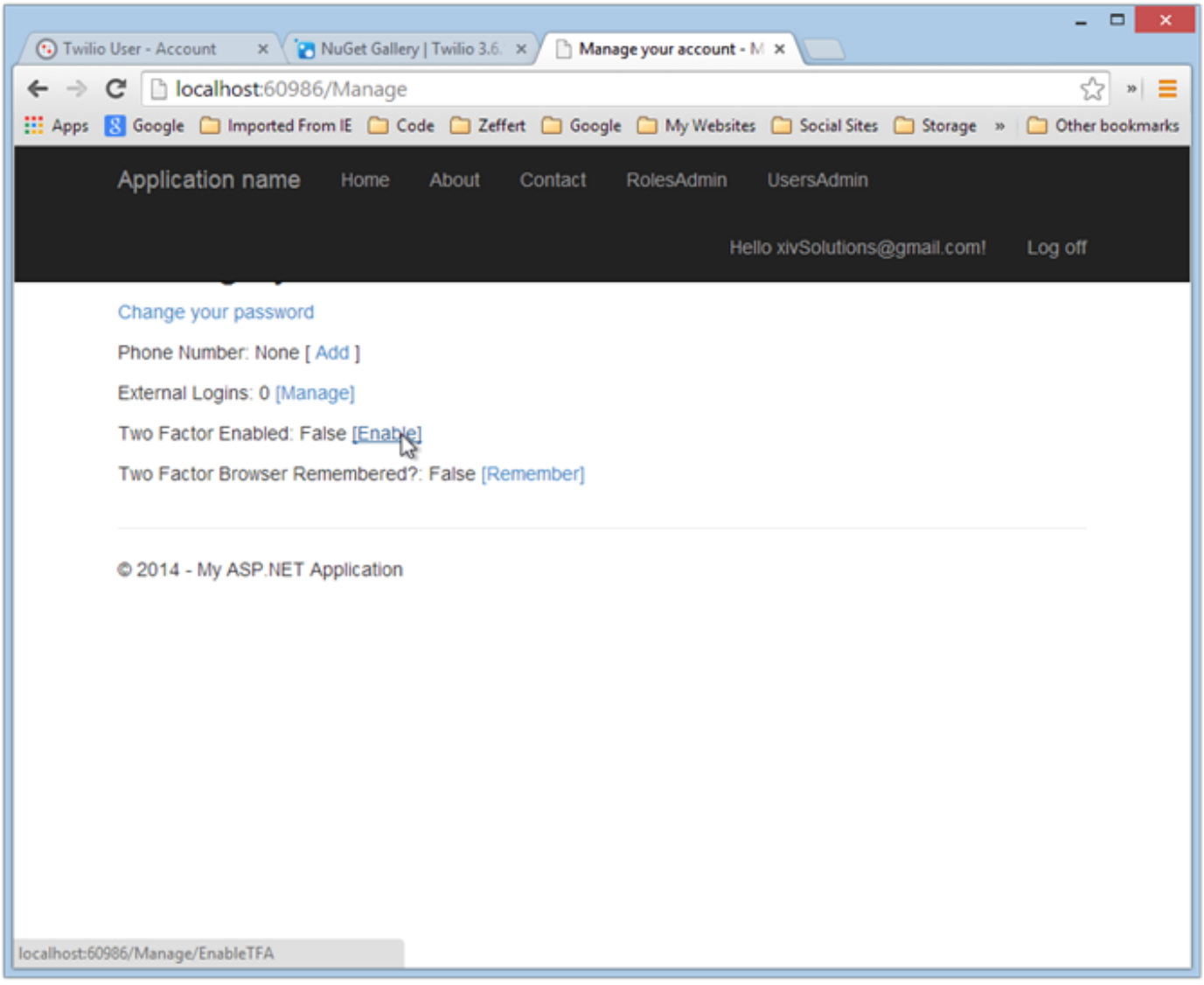
```
public class SmsService : IIdentityMessageService
{
    public Task SendAsync(IdentityMessage message)
    {
        string AccountSid = "YourTwilioAccountSID";
        string AuthToken = "YourTwilioAuthToken";
        string twilioPhoneNumber = "YourTwilioPhoneNumber";

        var twilio = new TwilioRestClient(AccountSid, AuthToken);

        twilio.SendSmsMessage(twilioPhoneNumber, message.Destination, message.Body);

        // Twilio does not return an async Task, so we need this:
        return Task.FromResult(0);
```

```
    }
}
```

حال که سرویس‌های ایمیل و پیامک را در اختیار داریم می‌توانیم احراز هویت دو مرحله ای را تست کنیم.

آزمایش احراز هویت دو مرحله ای

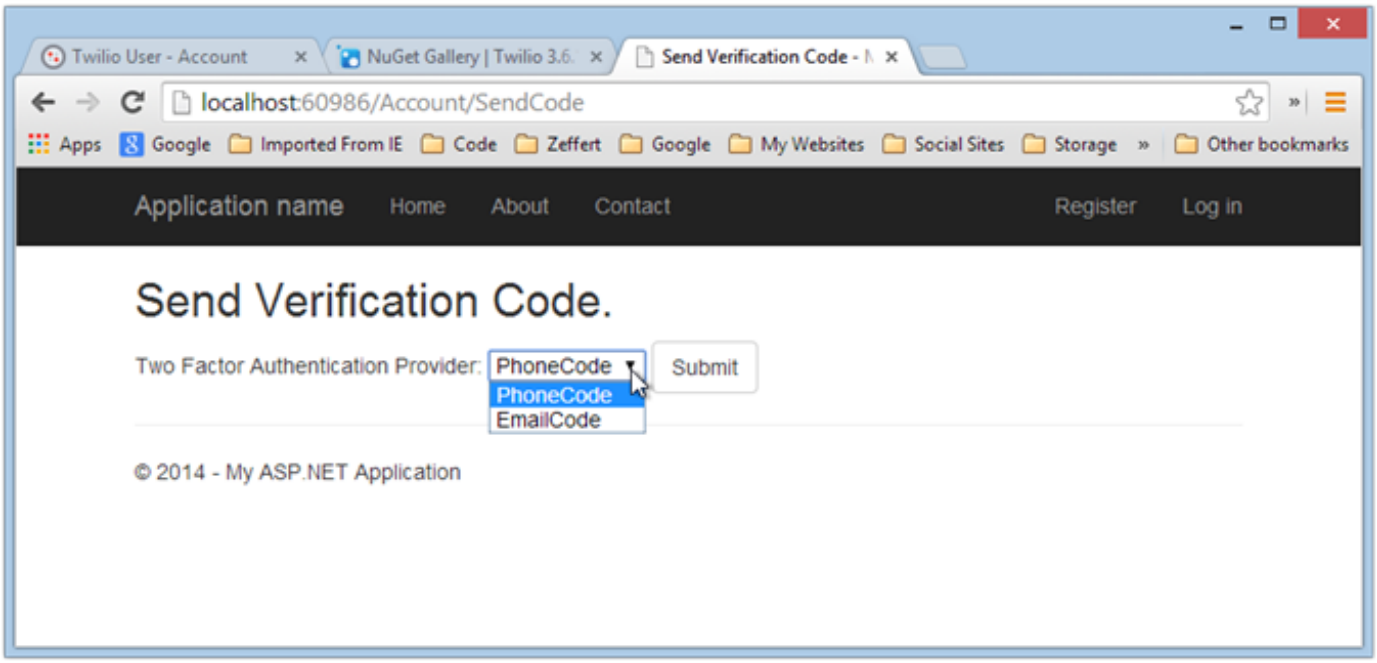
پروژه نمونه جاری طوری پیکربندی شده است که احراز هویت دو مرحله ای اختیاری است و در صورت لزوم می‌تواند برای هر کاربر بصورت جداگانه فعال شود. ابتدا توسط حساب کاربری مدیر، یا حساب کاربری ای که در قسمت تست تایید حساب کاربری ایجاد کرده اید وارد سایت شوید. سپس در سمت راست بالای صفحه روی نام کاربری خود کلیک کنید. باید صفحه ای مانند تصویر زیر را مشاهده کنید.



در این قسمت باید احراز هویت دو مرحله ای را فعال کنید و شماره تلفن خود را ثبت نمایید. پس از آن یک پیام SMS برای شما ارسال خواهد شد که توسط آن می‌توانید پروسه را تایید کنید. اگر همه چیز بدرستی کار کند این مراحل چند ثانیه بیشتر نباید زمان بگیرد، اما اگر مثلا بیش از 30 ثانیه زمان برد احتمالا اشکالی در کار است.

حال که احراز هویت دو مرحله ای فعال شده از سایت خارج شوید و مجددا سعی کنید به سایت وارد شوید. در این مرحله یک

انتخاب به شما داده می‌شود. می‌توانید کد احراز هویت دو مرحله ای خود را توسط ایمیل یا پیامک دریافت کنید.



پس از اینکه گزینه خود را انتخاب کردید، کد احراز هویت دو مرحله ای برای شما ارسال می‌شود که توسط آن می‌توانید پروسه ورود به سایت را تکمیل کنید.

حذف میانبرهای آزمایشی

همانطور که گفته شد پروژه نمونه شامل میانبرهایی برای تسهیل کار توسعه دهندگان است. در واقع اصلا نیازی به پیاده سازی سرویس‌های ایمیل و پیامک ندارید و می‌توانید با استفاده از این میانبرها حساب‌های کاربری را تایید کنید و کدهای احراز هویت دو مرحله ای را نیز مشاهده کنید. اما قطعا این میانبرها پیش از توزیع اپلیکیشن باید حذف شوند.

بدین منظور باید نماها و کدهای مربوطه را ویرایش کنیم تا اینگونه اطلاعات به کلاینت ارسال نشوند. اگر کنترلر AccountController را باز کنید و به متد Register() بروید با کد زیر مواجه خواهید شد.

```
if (result.Succeeded)
{
    var code = await UserManager.GenerateEmailConfirmationTokenAsync(user.Id);
    var callbackUrl =
        Url.Action("ConfirmEmail", "Account",
            new { userId = user.Id, code = code }, protocol: Request.Url.Scheme);

    await UserManager.SendEmailAsync(user.Id, "Confirm your account",
        "Please confirm your account by clicking this link: <a href=\"" + callbackUrl + "\">link</a>");

    // This should not be deployed in production:
    ViewBag.Link = callbackUrl;

    return View("DisplayEmail");
}

AddErrors(result);
```

همانطور که می‌بینید پیش از بازگشت از این متد، متغیر callbackUrl به ViewBag اضافه می‌شود. این خط را Comment کنید یا به کلی حذف نمایید.

نمایی که این متد باز می‌گرداند یعنی *DisplayEmail.cshtml* نیز باید ویرایش شود.

```
@{
    ViewBag.Title = "DEMO purpose Email Link";
}

<h2>@ViewBag.Title.</h2>

<p class="text-info">
    Please check your email and confirm your email address.
</p>

<p class="text-danger">
    For DEMO only: You can click this link to confirm the email: <a href="@ViewBag.Link">link</a>
    Please change this code to register an email service in IdentityConfig to send an email.
</p>
```

متد دیگری که در این کنترلر باید ویرایش شود ()VerifyCode است که کد احراز هویت دو مرحله ای را به صفحه مربوطه پاس می‌دهد.

```
[AllowAnonymous]
public async Task<ActionResult> VerifyCode(string provider, string returnUrl)
{
    // Require that the user has already logged in via username/password or external login
    if (!await SignInHelper.HasBeenVerified())
    {
        return View("Error");
    }

    var user =
        await UserManager.FindByIdAsync(await SignInHelper.GetVerifiedUserIdAsync());

    if (user != null)
    {
        ViewBag.Status =
            "For DEMO purposes the current "
            + provider
            + " code is: "
            + await UserManager.GenerateTwoFactorTokenAsync(user.Id, provider);
    }

    return View(new VerifyCodeViewModel { Provider = provider, ReturnUrl = returnUrl });
}
```

همانطور که می‌بینید متغیری بنام Status به ViewBag اضافه می‌شود که باید حذف شود.

نمای این متد یعنی *VerifyCode.cshtml* نیز باید ویرایش شود.

```
@model IdentitySample.Models.VerifyCodeViewModel

@{
    ViewBag.Title = "Enter Verification Code";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("VerifyCode", "Account", new { ReturnUrl = Model.ReturnUrl }, FormMethod.Post,
new { @class = "form-horizontal", role = "form" })) {
    @Html.AntiForgeryToken()
    @Html.ValidationSummary("", new { @class = "text-danger" })
    @Html.Hidden("provider", @Model.Provider)
    <h4>@ViewBag.Status</h4>
    <hr />

    <div class="form-group">
        @Html.LabelFor(m => m.Code, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
```

```
            @Html.TextBoxFor(m => m.Code, new { @class = "form-control" })
        </div>
    </div>

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <div class="checkbox">
                @Html.CheckBoxFor(m => m.RememberBrowser)
                @Html.LabelFor(m => m.RememberBrowser)
            </div>
        </div>
    </div>

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Submit" />
        </div>
    </div>
}
```

در این فایل کافی است ViewBag.Status را حذف کنید.

از تنظیمات ایمیل و SMS محافظت کنید

در مثال جاری اطلاعاتی مانند نام کاربری و کلمه عبور، شناسه‌های SID و Auth Token همگی در کد برنامه نوشته شده اند. بهتر است چنین مقادیری را بیرون از کد اپلیکیشن نگاه دارید، مخصوصا هنگامی که پروژه را به سرویس کنترل ارسال می‌کند (مثلا مخازن عمومی مثل GitHub). بدین منظور می‌توانید یکی از پست‌های اخیر را مطالعه کنید.

نظرات خوانندگان

نویسنده: دات نت کدینگ
تاریخ: ۱۳۹۳/۰۲/۱۱ ۱۳:۱

سلام
مقاله مفیدی بود برای من، ظاهرا نمیتوان نام جداول ایجادی توسط identity را تغییر داد، من از متد OnModelCreating() هم کمک گرفتم ولی بی نتیجه بود. به شکل زیر

```
modelBuilder.Entity<IdentityUser>().ToTable("MyUserRoles");
```

ممنون میشوم راهنمایی نمایید.

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۲/۱۱ ۱۳:۲۶

از ef-migrations برای اعمال تغییرات انجام شده استفاده کردید؟ بدون اینکار تغییری اعمال نمیشه.

نویسنده: دات نت کدینگ
تاریخ: ۱۳۹۳/۰۲/۱۴ ۰:۳۳

منظورتون اجرا کردن migration باشه، بله،
اینکار رو انجام دادین؟ شدنیه؟

نویسنده: علی
تاریخ: ۱۳۹۳/۰۳/۰۵ ۱۴:۱۹

توی این سری کدها، اون قسمت فعال سازی توسط ایمیل هنگام ثبت نام فعال هستش. اما مسئله ای که وجود داره کاربری که ثبت نام میکنه به صورت خودکار فعال هستش. حتی اگه روی اون لینکی که به ایمیل فرستاده میشه کلیک نکنه. راحت میتونه لاگین کنه. باید کجا این مسئله چک شود و چگونه؟ ممنون

نویسنده: Sam
تاریخ: ۱۳۹۳/۰۵/۰۴ ۲۳:۳۲

من از مدل لایه ای که آقای نصیری در مباحث ان تی تی کد فرست گفتن استفاد می‌کنم ، می‌خواستم بدونم که چطور میشه کلاس‌ها user identity رو به sql server به وسیله code first و migraiton انتقال داد البته با نام‌های دلخواه جداول در sql...
ممنون میشم اگر توضیح بدید،

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۰۵ ۱۰:۲

در Asp.Net Identity نباید یک DbContext جداگانه ایجاد کنید. از همان ApplicationDbContext آن جهت اضافه کردن سایر مدل‌های برنامه استفاده کنید؛ به همراه سفارشی سازی و توسعه آن. مابقی مسایل و نکات آن مانند سایر مباحث متداول EF Code first است و تفاوتی نمی‌کند.

نویسنده: سام میرزاقراچه
تاریخ: ۱۳۹۳/۰۶/۰۴ ۲۲:۳

با سلام مجدد
چگونه می‌توان در Role Manager یک Property جدید اضافه کرد و یا همان بخش Custom کردن Role در Identity 2 .....

نویسنده: ربال
تاریخ: ۱۳۹۳/۰۸/۱۸ ۰:۵۰

با سلام.آیا استفاد ه از Twilio محدودیت دارد؟ یعنی اگر تو پروژه استفاده کنیم بعدا ب مشکل بر نخواهیم خورد (از جانب رایگان بودنش)
ممنون .