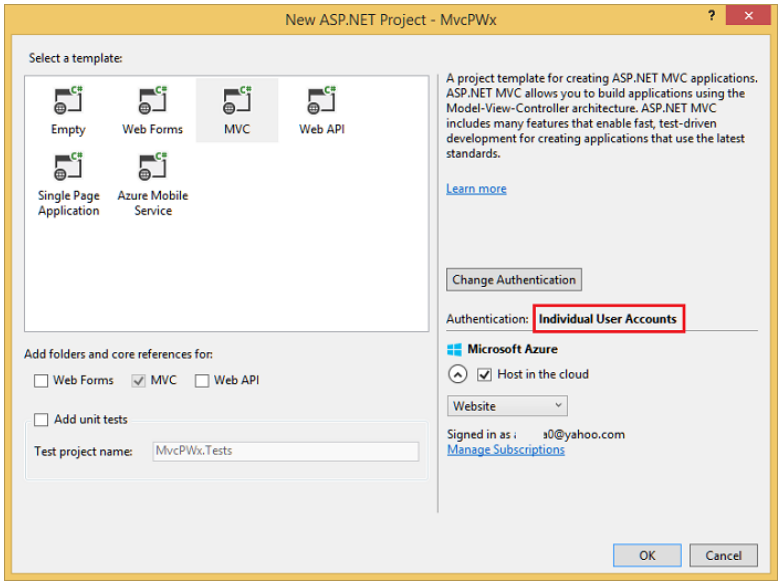
آموزش MVC – پیاده سازی یک اپلیکیشن تحت وب امن مبتنی بر ASP.NET MVC با قابلیت (ثبت ورود) login، (تایید ایمیل) email confirmation و (تنظیم مجدد گذرواژه) password reset (C#)

این آموزش نحوی ساخت یک اپلیکیشن تحت وب با ایمنی بالا مبتنی بر ASP.NET MVC5 به همراه قابلیت هایی نظیر تایید ایمیل، تنظیم مجدد گذرواژه و ثبت ورود به وسیله ی سیستم مدیریت کاربران در برنامه های ASP.NET (ASP.NET Identity membership) را به شما آموزش می دهد.

ساخت یک اپلیکیشن تحت وب ASP.NET مبتنی بر معماری MVC

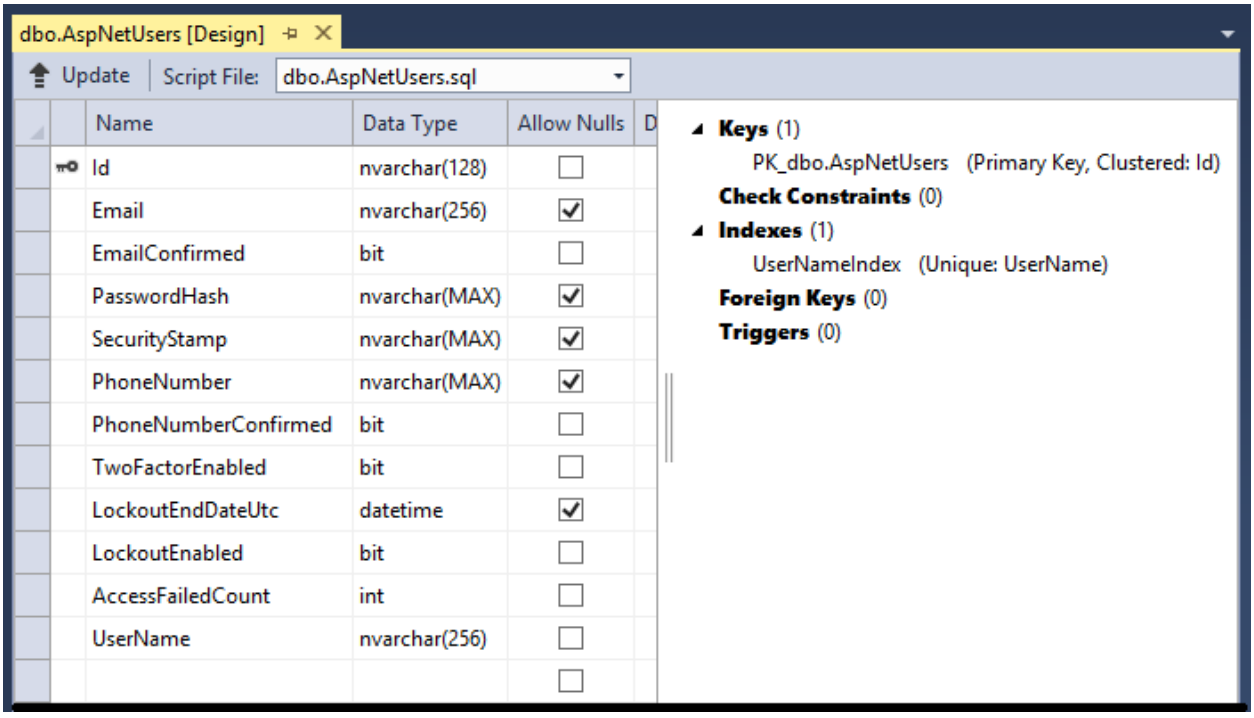
فرایند را با نصب و راه اندازی محیط برنامه نویسی Visual Studio Express 2013 for Web یا Visual Studio 2013 آغاز نمایید. لازم است آپدیت سوم Visual Studio 2013 یا جدیدتر را نصب نمایید.

1. یک پروژه ی اپلیکیشن تحت وب جدید ASP.NET ایجاد نموده و از قالب های آماده ی توسعه ی اپلیکیشن، MVC را انتخاب نمایید. لازم به ذکر است که Web forms نیز از توابع کتابخانه ای ASP.NET Identity پشتیبانی می کند، بنابراین می توانید همین مراحل را برای پروژه های web forms نیز طی نمایید.

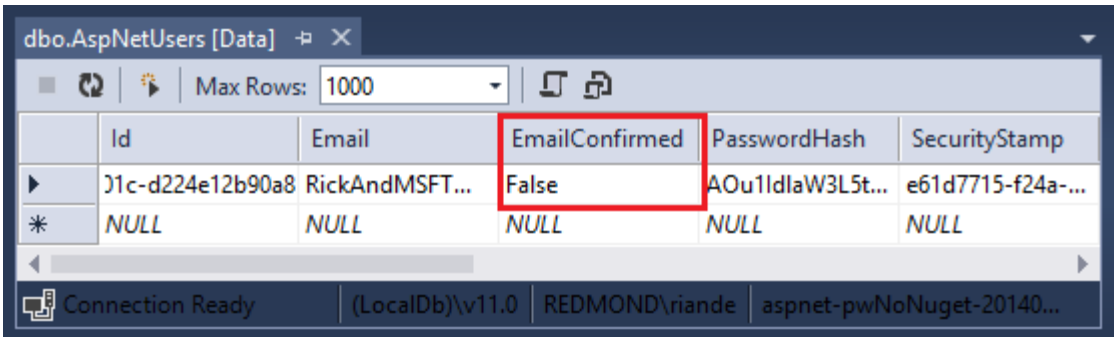


2. قابلیت احرازهویت پیش فرض را بر روی Individual User Accounts تنظیم نمایید. در صورت تمایل، می توانید اپلیکیشن را در بستر Azure میزبانی کنید. برای این منظور کافی است چک باکس مربوطه را تیک دار نمایید. در پروژه ی حاضر، اپلیکیشن را در بستر Azure نیز نصب و مستقر خواهیم کرد.
3. پروژه را طوری تنظیم کنید که از SSL استفاده کند.
4. اپلیکیشن را اجرا کرده، بر روی لینک Register کلیک نموده و یک کاربر جدید ثبت نام/معرفی نمایید. در این برهه تنها اعتبارسنجی که از طریق ایمیل صورت می گیرد به واسطه ی دستور [EmailAddress] می باشد.
5. در کادر Server Explorer، به آدرس Data Connections\DefaultConnection\Tables\AspNetUsers مراجعه نموده و پس از راست کلیک، گزینه ی Open table definition را انتخاب کنید.

تصویر زیر اسکیما (schema) AspNetUsers را نمایش می دهد:



6. بر روی جدول AspNetUsers راست کلیک کرده و Show Table Data را انتخاب نمایید.



7. در این برهه، ایمیل هنوز تایید (confirm) نشده است و ستون EmailConfirmed داری مقدار False می باشد.
8. بر روی سطر مربوطه کلیک کرده و delete را جهت حذف آن انتخاب نمایید. این ایمیل را در گام بعدی پیاده سازی پروژه بار دیگر اضافه نموده و یک ایمیل جهت تایید انجام عملیات به کاربر ارسال خواهید کرد.

پیاده سازی قابلیت تایید درستی ایمیل (Email confirmation)

توصیه می شود همیشه ایمیل یک کاربر که اخیرا ثبت نام کرده را بررسی کرده و مطمئن شوید این کاربر هویت شخص دیگری را جعل نمی کنند (به عبارت دیگر با آدرس ایمیل شخص دیگری در سایت شما ثبت نام نمی کنند). فرض کنید یک تالار گفتگو (forum) در اپلیکیشن خود تعبیه کرده و قصد دارید مانع از ثبت نام کاربری با ایمیل "bob@example.com" تحت عنوان joe@contoso.com شوید. بدون قابلیت بررسی و تایید ایمیل، ممکن است کاربر joe@contoso.com ایمیل های نامربوط از اپلیکیشن شما دریافت کند. برای مثال، فرض کنید کاربری به نام Bob به صورت تصادفی و بدون اینکه متوجه شود با ایمیل "bib@example.com" ثبت نام کرده است. این کاربر نمی تواند از قابلیت بازیابی گذرواژه (password recover) استفاده کند چرا که اپلیکیشن آدرس دقیق و صحیح ایمیل او را ندارد. لازم به ذکر است که قابلیت بررسی و تایید ایمیل، تنها میزان کمی مصونیت در برابر ربات ها فراهم کرده و از اپلیکیشن دربرابر spammer های مصمم محافظت نخواهد کرد. این ربات ها نام های جایگزین و مستعار متعددی برای آدرس های ایمیل خود دارند که با آن ها می توانند در سایت ثبت نام کنند.

بهتر است به کاربرانی که هنوز درستی اطلاعات آن ها توسط ایمیل، SMS یا هر مکانیزم دیگر تایید نشده، امکان ارسال داده به وب سایت خود را ندهید. در زیر به پیاده سازی قابلیت تایید ایمیل پرداخته و کد برنامه را طوری

ویرایش می کنیم که مانع از ورود کاربران جدید با اطلاعات تایید نشده (که درستی اطلاعات ثبت نام و ایمیل آن ها بررسی و کاملا تصدیق نشده)، می شود.

نصب و تنظیم SendGrid

اگرچه مبحث پیشرو پیاده سازی اطلاع رسانی email را تنها از طریق SendGrid به شما آموزش می دهد، شما می توانید ایمیل را با استفاده از SMTP یا سایر مکانیزم های تعریف شده برای این منظور ارسال کنید.

1. در PMC (پنجره ی کنسول مدیریت پکیج ها)، دستور زیر را درج نمایید:

```
Install-Package SendGrid
```

2. می توانید با مراجعه به صفحه ی ثبت نام Azure SendGrid (https://go.microsoft.com/fwlink/?linkid=271033&clcid=0x409) به صورت رایگان یک حساب کاربری SendGrid ایجاد نمایید. کد زیر را جهت انجام تنظیمات لازم سرویس SendGrid اضافه نمایید:

```
public class EmailService : IIdentityMessageService
{
public async Task SendAsync(IdentityMessage message)
  {
    await configSendGridasync(message);
  }
  // Use NuGet to install SendGrid (Basic C# client lib)
  private async Task configSendGridasync(IdentityMessage message)
  {
    var myMessage = new SendGridMessage();
    myMessage.AddTo(message.Destination);
    myMessage.From = new System.Net.Mail.MailAddress(
                "Joe@contoso.com", "Joe S.");
    myMessage.Subject = message.Subject;
    myMessage.Text = message.Body;
    myMessage.Html = message.Body;
    var credentials = new NetworkCredential(
           ConfigurationManager.AppSettings["mailAccount"],
           ConfigurationManager.AppSettings["mailPassword"]
           );
    // Create a Web transport for sending email.
    var transportWeb = new Web(credentials);
    // Send the email.
    if (transportWeb != null)
```

```
    {
      await transportWeb.DeliverAsync(myMessage);
    }
    else
    {
      Trace.TraceError("Failed to create Web transport.");
      await Task.FromResult(0);
    }
  }
}
```

لازم است دستورات using زیر را نیز به پروژه اضافه نمایید:

```
using SendGrid;
using System.Net;
using System.Configuration;
using System.Diagnostics;
```

جهت ساده نگه داشتن مثال پیشرو، تنظیمات اپلیکیشن را در فایل web.config ذخیره می کنیم:

```
</connectionStrings>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0" />
    <!-- Markup removed for clarity. -->
    <add key="mailAccount" value="xyz" />
    <add key="mailPassword" value="password" />
  </appSettings>
 <system.web>
```

نکته: هیچگاه اطلاعات حساس و محرمانه را در کد برنامه (source code) ذخیره نکنید. حساب کاربری و اطلاعات احراز هویت (credentials) در بخشی به نام appSetting نگهداری می شوند. در بستر Azure، شما می توانید این مقادیر را با رعایت کامل امنیت در تب Configure، داخل پورتال Azure ذخیره نمایید.

فعال سازی قابلیت تایید ایمیل (email confirmation) در کلاس کنترلر Account

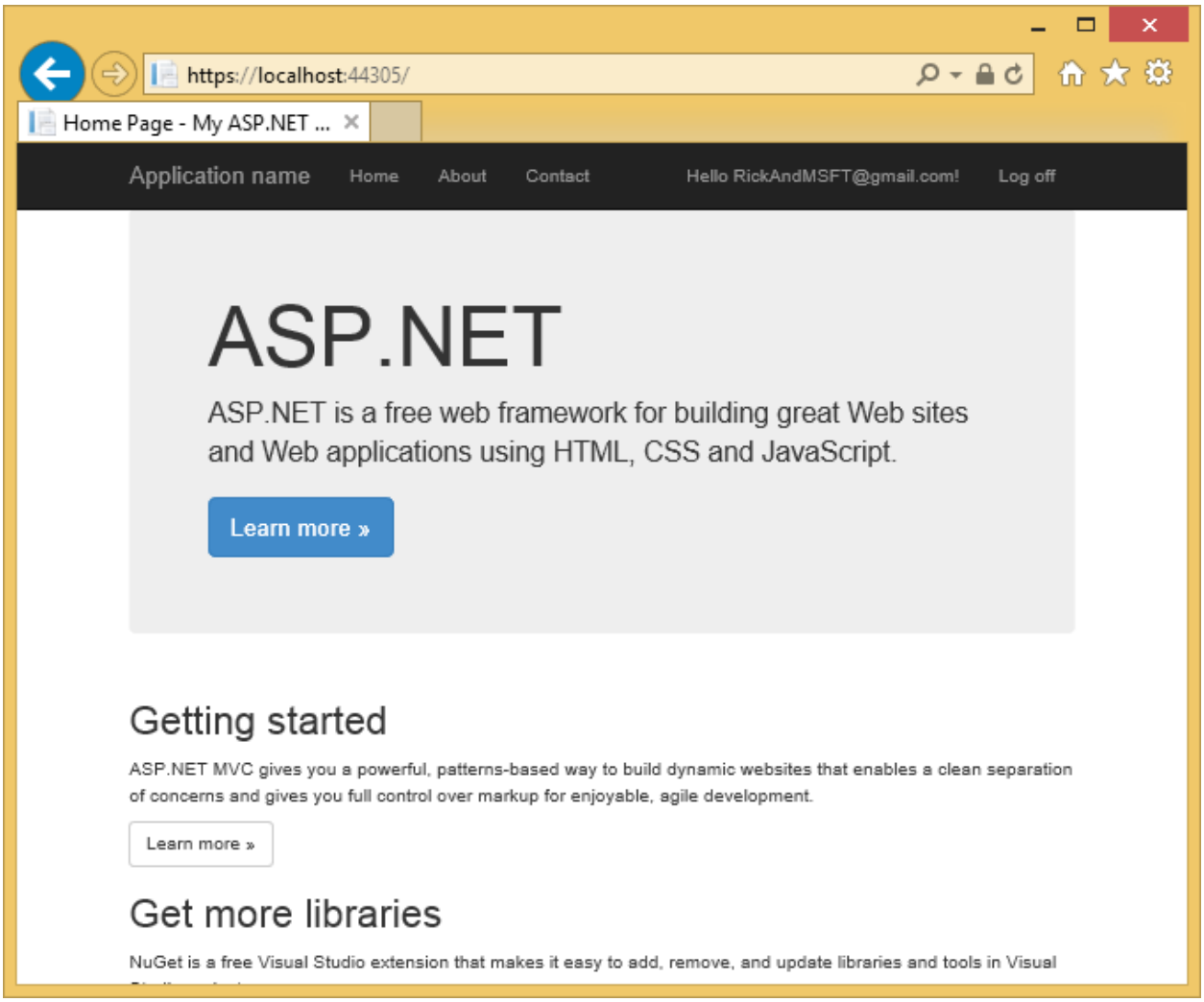
```
//
// POST: /Account/Register
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Register(RegisterViewModel model)
{
  if (ModelState.IsValid)
  {
```

```
    var user = new ApplicationUser { UserName = model.Email, Email = model.Email };
    var result = await UserManager.CreateAsync(user, model.Password);
    if (result.Succeeded)
    {
      await SignInManager.SignInAsync(user, isPersistent:false, rememberBrowser:false);
    string 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 <a href=\""
        + callbackUrl + "\">here</a>");
      return RedirectToAction("Index", "Home");
    }
    AddErrors(result);
  }
  // If we got this far, something failed, redisplay form
  return View(model);
}
```

فایل Views\Account\ConfirmEmail.cshtml را بررسی کرده و مطمئن شوید که گرامر و سینتکس razor آن صحیح و استاندارد می باشد. (ممکن است کاراکتر @ در اولین خط از محتوای فایل درج نشده باشد.)

```
@{
  ViewBag.Title = "Confirm Email";
}
<h2>@ViewBag.Title.</h2>
<div>
  <p>
    Thank you for confirming your email. Please @Html.ActionLink("Click here to Log in", "Login", "Account",
routeValues: null, htmlAttributes: new { id = "loginLink" })
  </p>
</div>
```

اپلیکیشن را اجرا نموده و بر روی لینک Register کلیک کنید. پس از تکمیل و ارسال فرم ثبت نام (registration)، شما وارد سایت (login) می شوید.



حساب کاربری ایمیل خود را چک کرده و بر روی لینک مربوطه جهت تایید ایمیل مورد نظر کلیک نمایید.

الزامی کردن تایید ایمیل (email confirmation) قبل از ثبت ورود کاربر (login)

در حال حاضر زمانی که یک کاربر فرم ثبت نام را تکمیل می کند، بلافاصله به داخل سایت راه پیدا (log in) می کند. بهتر است که ایمیل کاربر را قبل از اینکه وارد سایت شود، حتما بررسی و تایید نمایید. در زیر، کد را طوری ویرایش می کنید که کاربران جدید ملزوم به داشتن ایمیل معتبر و تایید شده برای ورود به سایت شوند (احراز هویت شده و login شوند). متد HttpPost Register را با اعمال تغییرات هایلایت شده ی زیر بروز رسانی نمایید:

```
//
// POST: /Account/Register
```

```
[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)
    {
    // Comment the following line to prevent log in until the user is confirmed.
      // await SignInManager.SignInAsync(user, isPersistent:false, rememberBrowser:false);
      string 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 <a href=\"" + callbackUrl + "\">here</a>");
    // Uncomment to debug locally
    // TempData["ViewBagLink"] = callbackUrl;
    ViewBag.Message = "Check your email and confirm your account, you must be confirmed "
              + "before you can log in.";
    return View("Info");
      //return RedirectToAction("Index", "Home");
    }
    AddErrors(result);
  }
  // If we got this far, something failed, redisplay form
  return View(model);
}
```

با قرار دادن متد SignInAsync در بدنه ی کامنت، کاربر دیگر پس از ثبت نام و registration، به داخل سایت راه پیدا نمی کند. خط TempData["ViewBagLink"] = callbackUrl; را می توانید جهت اشکال زدایی اپلیکیشن و تست قابلیت ثبت نام (registration) بدون ارسال ایمیل بکار ببرید. دستور ViewBag.Message برای نمایش دستورالعمل های مربوط به بررسی و تایید ایمیل بکار می رود.

فایل Views\Shared\Info.cshtml را ایجاد کرده و کد razor markup (کد C# داخل HTML) زیر را اضافه نمایید:

```
@{
  ViewBag.Title = "Info";
}
<h2>@ViewBag.Title.</h2>
<h3>@ViewBag.Message</h3>
```

دستور Authorize (attribute) را به متد Contact از کلاس Home اضافه نمایید. می توانید با کلیک بر روی لینک Contact، بررسی کرده و مطمئن شوید که کاربران ناشناخته دسترسی به داخل سایت ندارند و کاربران احراز هویت شده می توانند به سایت لاگین کنند.

```
[Authorize]
public ActionResult Contact()
{
  ViewBag.Message = "Your contact page.";
  return View();
}
```

شما بایستی بدنه ی متد HttpPost Login را بروز رسانی کنید:

```
//
// POST: /Account/Login
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
{
   if (!ModelState.IsValid)
   {
      return View(model);
   }
// Require the user to have a confirmed email before they can log on.
var user = await UserManager.FindByNameAsync(model.Email);
if (user != null)
{
  if (!await UserManager.IsEmailConfirmedAsync(user.Id))
  {
     ViewBag.errorMessage = "You must have a confirmed email to log on.";
     return View("Error");
  }
   }
   // This doesn't count login failures towards account lockout
   // To enable password failures to trigger account lockout, change to shouldLockout: true
   var result = await SignInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe,
shouldLockout: false);
   switch (result)
   {
      case SignInStatus.Success:
         return RedirectToLocal(returnUrl);
      case SignInStatus.LockedOut:
         return View("Lockout");
      case SignInStatus.RequiresVerification:
         return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, RememberMe = model.RememberMe });
      case SignInStatus.Failure:
      default:
```

```
            ModelState.AddModelError("", "Invalid login attempt.");
            return View(model);
        }
    }
```

فایل (view) Views\Shared\Error.cshtml را بروزرسانی کنید تا پیغام خطا برای کاربر نمایش داده شود:

```
@model System.Web.Mvc.HandleErrorInfo
@{
    ViewBag.Title = "Error";
}
<h1 class="text-danger">Error.</h1>
@{
  if (String.IsNullOrEmpty(ViewBag.errorMessage))
  {
    <h2 class="text-danger">An error occurred while processing your request.</h2>
  }
  else
  {
    <h2 class="text-danger">@ViewBag.errorMessage</h2>
  }
}
```

تمامی حساب های کاربری (account ها) را که دارای نام جایگزین ایمیل (email alias) بوده و شما می خواهید آن را تست کنید، از جدول AspNetUsers حذف نمایید. اپلیکیشن را اجرا کرده و مطمئن شوید که بدون بررسی و تایید آدرس ایمیل خود نمی توانید وارد سایت شوید.

پس از بررسی و تایید آدرس ایمیل، بر روی لینک Contact کلیک کنید.

تنظیم مجدد رمزعبور (Password recovery/reset)

کاراکترهای کامنت گذاری را از متد HttpPost ForgotPassword در کلاس account حذف نمایید:

```
//
// POST: /Account/ForgotPassword
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> ForgotPassword(ForgotPasswordViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = await UserManager.FindByNameAsync(model.Email);
        if (user == null || !(await UserManager.IsEmailConfirmedAsync(user.Id)))
        {
```

```
        // Don't reveal that the user does not exist or is not confirmed
        return View("ForgotPasswordConfirmation");
      }
    string code = await UserManager.GeneratePasswordResetTokenAsync(user.Id);
    var callbackUrl = Url.Action("ResetPassword", "Account", new { userId = user.Id, code = code }, protocol:
Request.Url.Scheme);
    await UserManager.SendEmailAsync(user.Id, "Reset Password", "Please reset your password by clicking <a href=\"" +
callbackUrl + "\">here</a>");
      return RedirectToAction("ForgotPasswordConfirmation", "Account");
    }
    // If we got this far, something failed, redisplay form
    return View(model);
  }
```

همچنین کاراکترهای کامنت گذاری را از لینک (ActionLink) ForgotPassword در فایل (razor view) Views\Account\Login.cshtml حذف نمایید:

```
@using MvcPWy.Models
@model LoginViewModel
@{
  ViewBag.Title = "Log in";
}
<h2>@ViewBag.Title.</h2>
<div class="row">
  <div class="col-md-8">
    <section id="loginForm">
      @using (Html.BeginForm("Login", "Account", new { ReturnUrl = ViewBag.ReturnUrl }, FormMethod.Post, new {
@class = "form-horizontal", role = "form" }))
      {
        @Html.AntiForgeryToken()
        <h4>Use a local account to log in.</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
          @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label" })
          <div class="col-md-10">
            @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
            @Html.ValidationMessageFor(m => m.Email, "", new { @class = "text-danger" })
          </div>
        </div>
        <div class="form-group">
          @Html.LabelFor(m => m.Password, new { @class = "col-md-2 control-label" })
          <div class="col-md-10">
            @Html.PasswordFor(m => m.Password, new { @class = "form-control" })
            @Html.ValidationMessageFor(m => m.Password, "", new { @class = "text-danger" })
          </div>
        </div>
        <div class="form-group">
          <div class="col-md-offset-2 col-md-10">
```

```
            <div class="checkbox">
              @Html.CheckBoxFor(m => m.RememberMe)
              @Html.LabelFor(m => m.RememberMe)
            </div>
          </div>
        </div>
        <div class="form-group">
          <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Log in" class="btn btn-default" />
          </div>
        </div>
        <p>
          @Html.ActionLink("Register as a new user", "Register")
        </p>
      @* Enable this once you have account confirmation enabled for password reset functionality *@
      <p>
        @Html.ActionLink("Forgot your password?", "ForgotPassword")
        </p>
      }
    </section>
  </div>
  <div class="col-md-4">
    <section id="socialLoginForm">
      @Html.Partial("_ExternalLoginsListPartial", new ExternalLoginListViewModel { ReturnUrl = ViewBag.ReturnUrl })
    </section>
  </div>
</div>
@section Scripts {
  @Scripts.Render("~/bundles/jqueryval")
}
```

اکنون صفحه ی ورود به سایت، یک لینک جهت تنظیم مجدد رمزعبور (reset password) برای کاربر نمایش می دهد.

ارسال مجدد لینک تایید ایمیل

زمانی که کاربری یک حساب کاربری محلی جدید ایجاد می کند، اپلیکیشن یک لینک تایید (confirmation link) به آن ها از طریق ایمیل می فرستد. کاربر مجاب است که قبل از ورود به سایت، بر روی این اینک کلیک کند. چنانچه کاربر به صورت تصادفی ایمیل تایید را پاک کند یا ایمیل هیچگاه به دست کاربر نرسد، اپلیکیشن بایستی لینک تایید را دوباره به او ارسال نماید. تغییراتی که در کد زیر مشاهده می کنید، نحوه ی پیاده سازی این قابلیت را نمایش می دهد.

متد کمکی (helper) زیر را به پایین فایل Controllers\AccountController.cs اضافه نمایید:

```
private async Task<string> SendEmailConfirmationTokenAsync(string userID, string subject)
{
  string code = await UserManager.GenerateEmailConfirmationTokenAsync(userID);
  var callbackUrl = Url.Action("ConfirmEmail", "Account",
    new { userId = userID, code = code }, protocol: Request.Url.Scheme);
  await UserManager.SendEmailAsync(userID, subject,
    "Please confirm your account by clicking <a href=\"" + callbackUrl + "\">here</a>");
  return callbackUrl;
}
```

متد Register را جهت استفاده از متد کمکی (helper method) جدید به صورت زیر بروز رسانی نمایید:

```
//
// POST: /Account/Register
[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)
    {
      // Comment the following line to prevent log in until the user is confirmed.
      // await SignInManager.SignInAsync(user, isPersistent:false, rememberBrowser:false);
      string callbackUrl = await SendEmailConfirmationTokenAsync(user.Id, "Confirm your account");
      ViewBag.Message = "Check your email and confirm your account, you must be confirmed "
                + "before you can log in.";
      return View("Info");
      //return RedirectToAction("Index", "Home");
    }
    AddErrors(result);
  }
  // If we got this far, something failed, redisplay form
  return View(model);
}
```

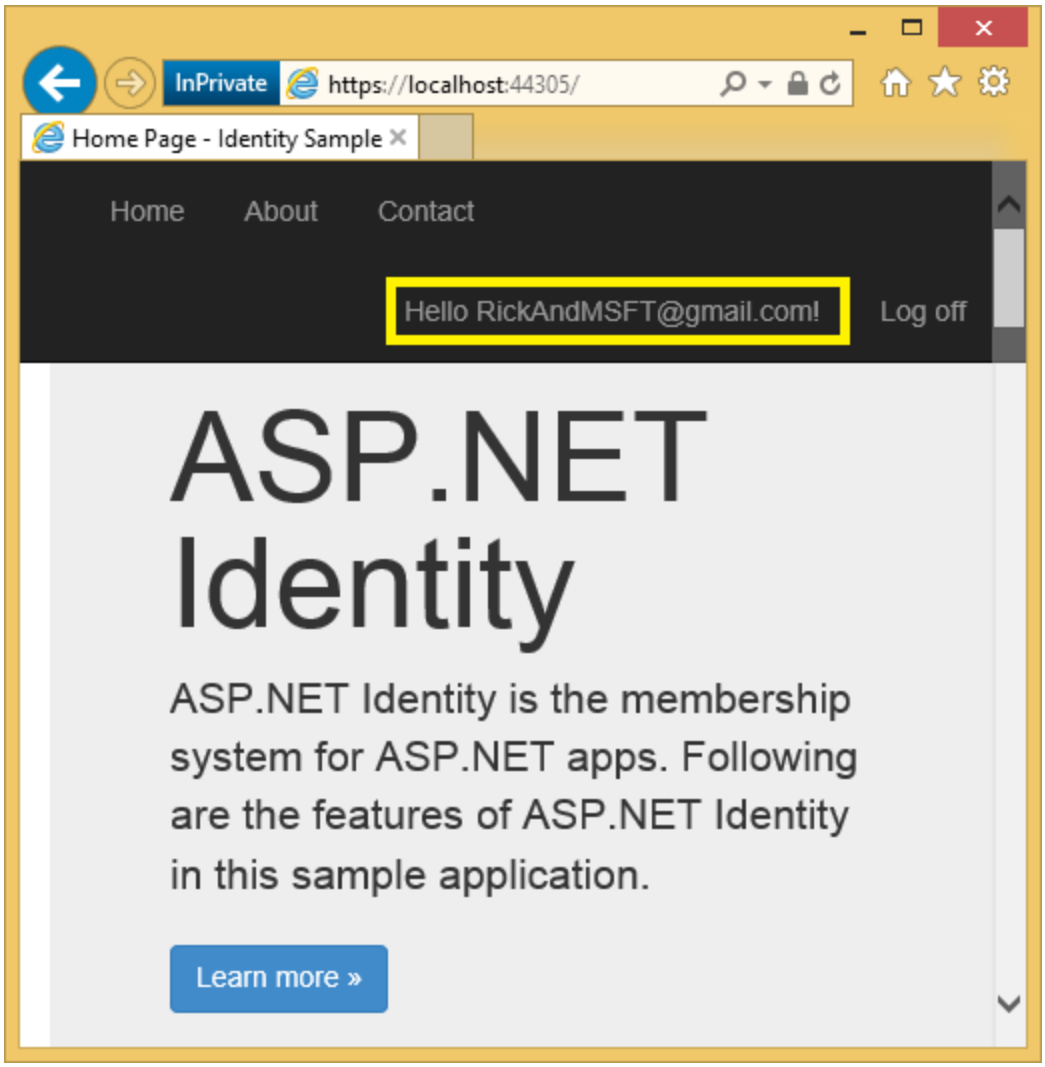
متد Login را طوری ویرایش و بروز رسانی نمایید که اگر حساب کاربری تایید نشد، رمزعبور را مجددا ارسال کند:

```
//
// POST: /Account/Login
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
{
  if (!ModelState.IsValid)
```

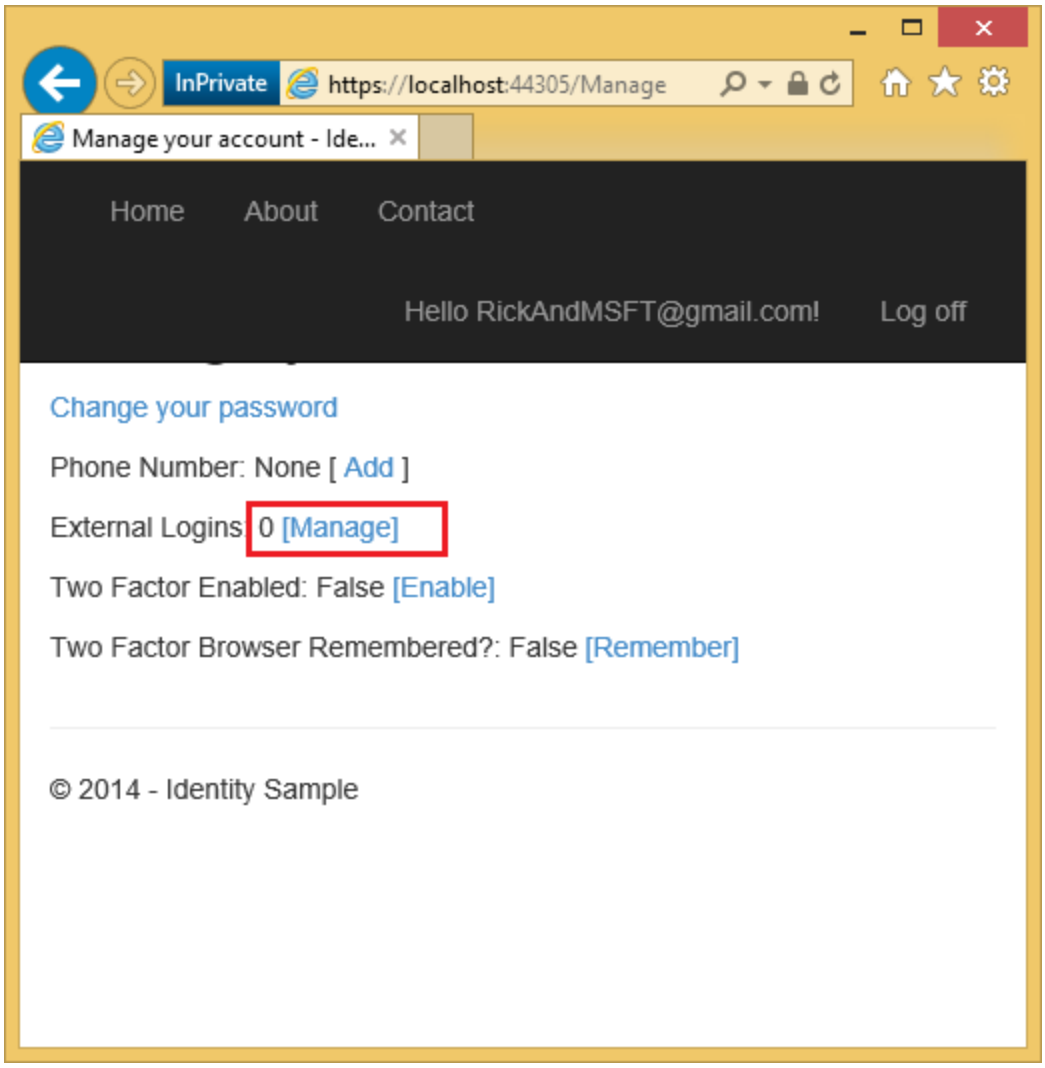
```
 {
   return View(model);
 }
 // Require the user to have a confirmed email before they can log on.
// var user = await UserManager.FindByNameAsync(model.Email);
 var user =  UserManager.Find(model.Email, model.Password);
 if (user != null)
 {
   if (!await UserManager.IsEmailConfirmedAsync(user.Id))
   {
     string callbackUrl = await SendEmailConfirmationTokenAsync(user.Id, "Confirm your account-Resend");
      // Uncomment to debug locally
      // ViewBag.Link = callbackUrl;
     ViewBag.errorMessage = "You must have a confirmed email to log on. "
                 + "The confirmation token has been resent to your email account.";
     return View("Error");
   }
 }
 // This doesn't count login failures towards account lockout
 // To enable password failures to trigger account lockout, change to shouldLockout: true
 var result = await SignInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe,
shouldLockout: false);
 switch (result)
 {
   case SignInStatus.Success:
     return RedirectToLocal(returnUrl);
   case SignInStatus.LockedOut:
     return View("Lockout");
   case SignInStatus.RequiresVerification:
     return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, RememberMe = model.RememberMe });
   case SignInStatus.Failure:
   default:
     ModelState.AddModelError("", "Invalid login attempt.");
     return View(model);
 }
}
```

ترکیب حساب های کاربری شبکه های اجتماعی و حساب های کاربری (محلی) ورود به سایت

می توانید حساب های کاربری محلی و اجتماعی را با کلیک بر روی لینک ایمیل، با هم ادغام نمایید. همان طور که در توالی زیر مشاهده می کنید، آدرس RickAndMSFT@gmail.com ابتدا به عنوان یک حساب ثبت ورود محلی تعریف شده است. اما شما می توانید حساب کاربری را اول به عنوان ثبت ورود به شبکه ی اجتماعی (social login) ایجاد کرده، سپس یک حساب کاربری ثبت ورود به سایت محلی را به آن اضافه نمایید.

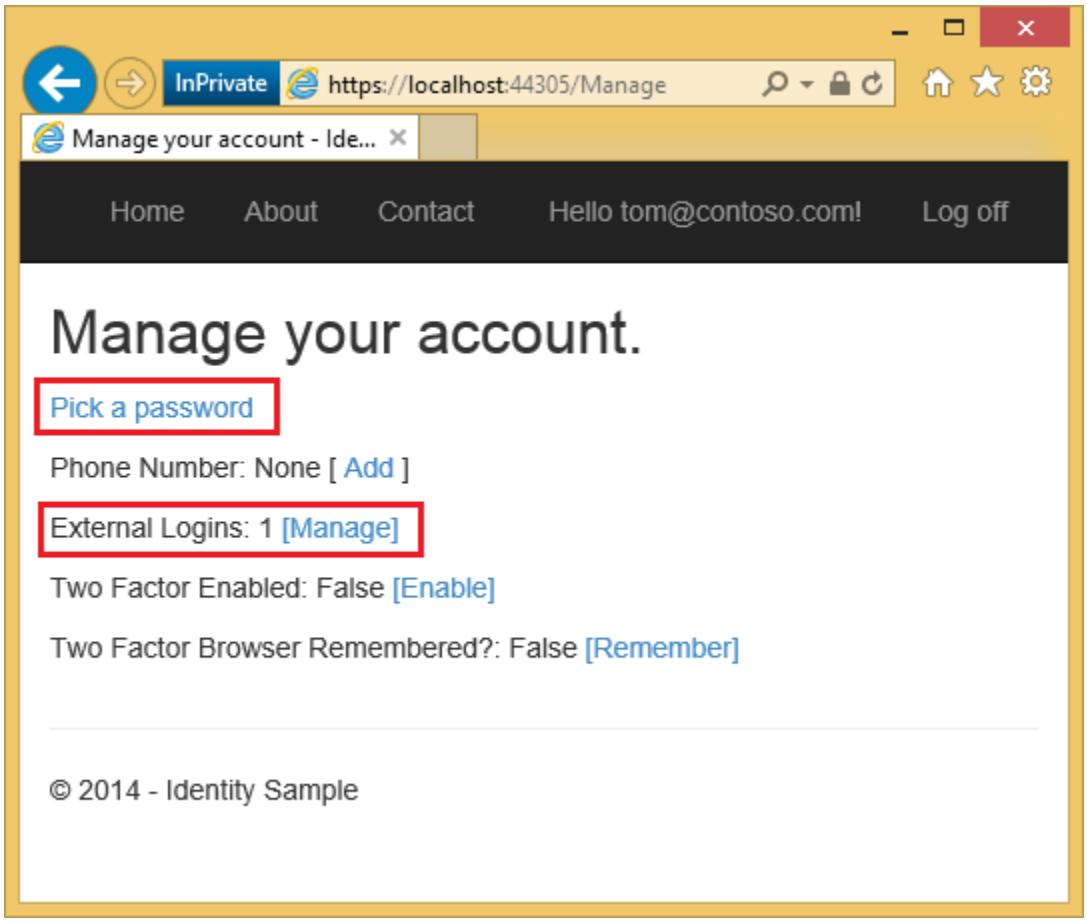


بر روی لینک Manage کلیک نمایید. همان طور که می بینید هیچ حساب کاربری ورود به شبکه ی اجتماعی (social login) مرتبط با این حساب کاربری وجود ندارد (مقابل External Login مقدار 0 قرار داده شده است).

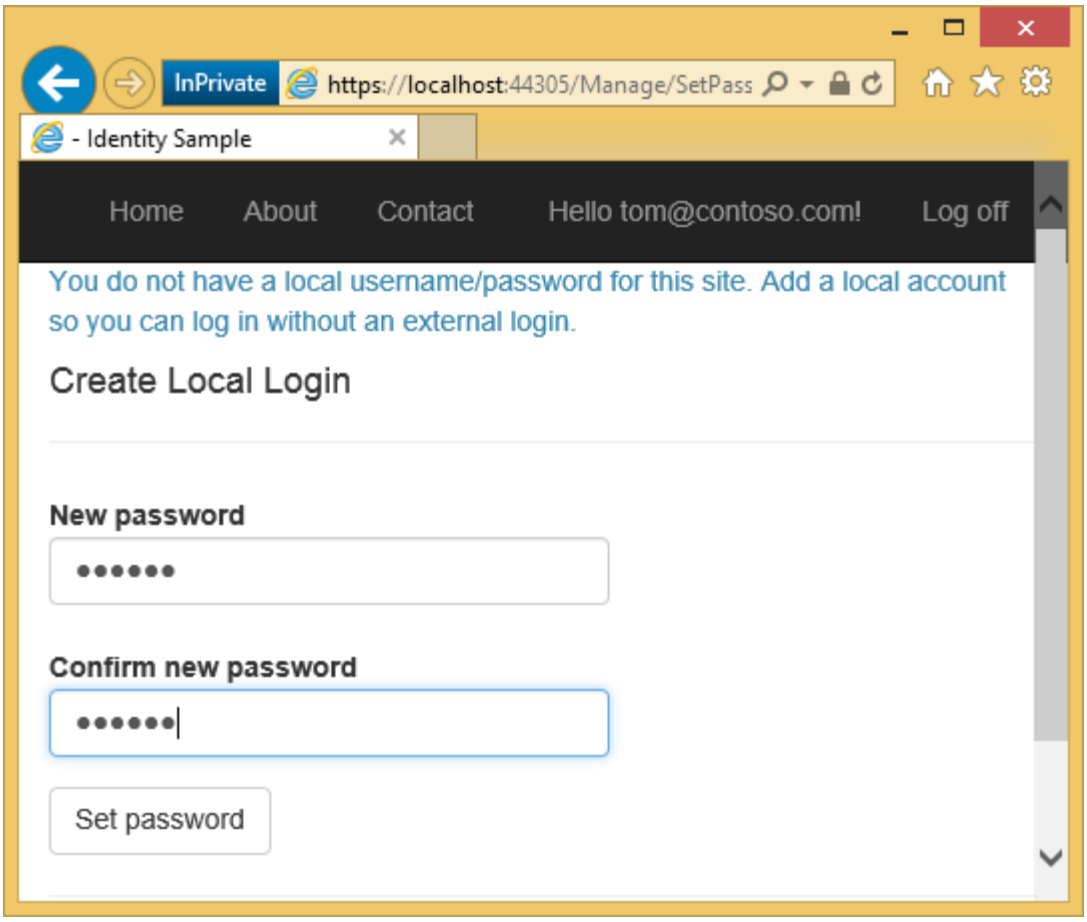


بر روی لینک مرتبط با سرویس ورود به سایت (login service) دیگری کلیک نموده و درخواست های ورودی اپلیکیشن را بپذیرد (accept نمایید). اکنون که دو حساب کاربری محلی و شبکه ی اجتماعی با هم ترکیب شده اند، شما قادر خواهید بود با هر یک از دو حساب کاربری که می خواهید وارد سایت شوید. بد نیست از کاربران خود بخواهید علاوه بر حساب های ورود به شبکه های اجتماعی (social account) خود، یک حساب کاربری محلی نیز ایجاد کنند تا در صورت خراب یا در دسترس نبودن سرویس احرازهویت شبکه ی اجتماعی مربوطه (social log in authentication service) یا عدم دسترسی به حساب کاربری حساب شبکه ی اجتماعی خود، بتوانند به سایت لاگین کنند.

در تصویر زیر، Tom یک حساب کاربری ورود به شبکه های اجتماعی دارد. همان طور که می بینید مقابل آیتم External Login مقدار 1 قرار داده شده است.



با کلیک بر روی لینک Pick a password می توانید یک حساب کاربری (محلی) ورود به سایت/local logon که با حساب کاربری مزبور (social account) مرتبط است، اضافه نمایید.



اشکال زدایی اپلیکیشن (Debug)

چنانچه ایمیلی که دربردارنده ی لینک مربوطه باشد را دریافت نمی کنید، در آن صورت لازم است اقدامات زیر را انجام دهید:

- پوشه ی spam یا junk را بررسی کنید.
- به حساب کاربری SendGrid خود وارد شده (لاگین کرده) و بر روی لینک Email Activity کلیک نمایید.