عنوان: **ساخت یک اپلیکیشن ساده ToDo با ASP.NET Identity**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۹ ۱۳:۳۵
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, Tips, ASP.NET MVC 5, ASP.NET Identity, MVC

یک سناریوی فرضی را در نظر بگیرید. اگر بخواهیم IdentityDbContext و دیگر DbContextهای اپلیکیشن را ادغام کنیم چه باید کرد؟ مثلا یک سیستم وبلاگ که برخی کاربران می‌توانند پست جدید ثبت کنند، برخی تنها می‌توانند کامنت بگذارند و تمامی کاربران هم اختیارات مشخص دیگری دارند. در چنین سیستمی شناسه کاربران (User ID) در بسیاری از مدل‌ها (موجودیت‌ها و مدل‌های اپلیکیشن) وجود خواهد داشت تا مشخص شود هر رکورد به کدام کاربر متعلق است. در این مقاله چنین سناریو هایی را بررسی می‌کنیم و best practiceهای مربوطه را مرور می‌کنیم.
در این پست یک اپلیکیشن ساده ToDo خواهیم ساخت که امکان تخصیص to-doها به کاربران را نیز فراهم می‌کند. در این مثال خواهیم دید که چگونه می‌توان مدل‌های مختص به سیستم عضویت (IdentityDbContext) را با مدل‌های دیگر اپلیکیشن مخلوط و استفاده کنیم.

تعریف نیازمندی‌های اپلیکیشن

تنها کاربران احراز هویت شده قادر خواهند بود تا لیست ToDoهای خود را ببینند، آیتم‌های جدید ثبت کنند یا داده‌های قبلی را ویرایش و حذف کنند.
کاربران نباید آیتم‌های ایجاد شده توسط دیگر کاربران را ببینند.
تنها کاربرانی که به نقش Admin تعلق دارند باید بتوانند تمام ToDoهای ایجاد شده را ببینند.

پس بگذارید ببینیم چگونه می‌شود اپلیکیشنی با ASP.NET Identity ساخت که پاسخگوی این نیازمندی‌ها باشد.
ابتدا یک پروژه ASP.NET MVC جدید با مدل احراز هویت Individual User Accounts بسازید. در این اپلیکیشن کاربران قادر خواهند بود تا بصورت محلی در وب سایت ثبت نام کنند و یا با تامین کنندگان دیگری مانند گوگل و فیسبوک وارد سایت شوند. برای ساده نگاه داشتن این پست ما از حساب‌های کاربری محلی استفاده می‌کنیم.
در مرحله بعد ASP.NET Identity را راه اندازی کنید تا بتوانیم نقش مدیر و یک کاربر جدید بسازیم. می‌توانید با اجرای اپلیکیشن راه اندازی اولیه را انجام دهید. از آنجا که سیستم ASP.NET Identity توسط Entity Framework مدیریت می‌شود می‌توانید از تنظیمات پیکربندی Code First برای راه اندازی دیتابیس خود استفاده کنید.
در قدم بعدی راه انداز دیتابیس را در Global.asax تعریف کنید.

```
Database.SetInitializer<MyDbContext>(new MyDbInitializer());
```

ایجاد نقش مدیر و کاربر جدیدی که به این نقش تعلق دارد

اگر به قطعه کد زیر دقت کنید، می‌بینید که در خط شماره 5 متغیری از نوع UserManager ساخته ایم که امکان اجرای عملیات گوناگونی روی کاربران را فراهم می‌کند. مانند ایجاد، ویرایش، حذف و اعتبارسنجی کاربران. این کلاس که متعلق به سیستم ASP.NET Identity است همتای SQLMembershipProvider در ASP.NET 2.0 است.
در خط 6 یک RoleManager می‌سازیم که امکان کار با نقش‌ها را فراهم می‌کند. این کلاس همتای SQLRoleMembershipProvider در ASP.NET 2.0 است.
در این مثال نام کلاس کاربران (موجودیت کاربر در IdentityDbContext) برابر با "MyUser" است، اما نام پیش فرض در قالب‌های پروژه VS 2013 برابر با "ApplicationUser" می‌باشد.

```
public class MyDbInitializer : DropCreateDatabaseAlways<MyDbContext>
    {
        protected override void Seed(MyDbContext context)
        {
            var UserManager = new UserManager<MyUser>(new

                                                UserStore<MyUser>(context));

            var RoleManager = new RoleManager<IdentityRole>(new
                                            RoleStore<IdentityRole>(context));
```

```
            string name = "Admin";
            string password = "123456";


            //Create Role Admin if it does not exist
            if (!RoleManager.RoleExists(name))
            {
                var roleresult = RoleManager.Create(new IdentityRole(name));
            }

            //Create User=Admin with password=123456
            var user = new MyUser();
            user.UserName = name;
            var adminresult = UserManager.Create(user, password);

            //Add User Admin to Role Admin
            if (adminresult.Succeeded)
            {
                var result = UserManager.AddToRole(user.Id, name);
            }
            base.Seed(context);
        }
    }
```

حال فایلی با نام Models/AppModels.cs بسازید و مدل EF Code First اپلیکیشن را تعریف کنید. از آنجا که از EF استفاده می‌کنیم، روابط کلیدها بین کاربران و ToDoها بصورت خودکار برقرار می‌شود.

```
public class MyUser : IdentityUser
    {
        public string HomeTown { get; set; }
        public virtual ICollection<ToDo>
                              ToDoes { get; set; }
    }

    public class ToDo
    {
        public int Id { get; set; }
        public string Description { get; set; }
        public bool IsDone { get; set; }
        public virtual MyUser User { get; set; }
    }
```

در قدم بعدی با استفاده از مکانیزم Scaffolding کنترلر جدیدی بهمراه تمام Viewها و متدهای لازم برای عملیات CRUD بسازید.
برای اطلاعات بیشتر درباره نحوه استفاده از مکانیزم Scaffolding به این لینک مراجعه کنید.
لطفا دقت کنید که از DbContext فعلی استفاده کنید. این کار مدیریت داده‌های Identity و اپلیکیشن شما را یکپارچه‌تر می‌کند.
DbContext شما باید چیزی شبیه به کد زیر باشد.

```
public class MyDbContext : IdentityDbContext<MyUser>
    {
        public MyDbContext()
            : base("DefaultConnection")
        {
         }

         protected override void OnModelCreating(DbModelBuilder modelBuilder)
         {
        public System.Data.Entity.DbSet<AspnetIdentitySample.Models.ToDo>
                    ToDoes { get; set; }
    }
```

تنها کاربران احراز هویت شده باید قادر به اجرای عملیات CRUD باشند

برای این مورد از خاصیت Authorize استفاده خواهیم کرد که در MVC 4 هم وجود داشت. برای اطلاعات بیشتر لطفا به این لینک مراجعه کنید.

```
[Authorize]
public class ToDoController : Controller
```

کنترلر ایجاد شده را ویرایش کنید تا کاربران را به ToDoها اختصاص دهد. در این مثال تنها اکشن متدهای Create و List را بررسی خواهیم کرد. با دنبال کردن همین روش می‌توانید متدهای Edit و Delete را هم بسادگی تکمیل کنید.
یک متد constructor جدید بنویسید که آبجکتی از نوع UserManager می‌پذیرد. با استفاده از این کلاس می‌توانید کاربران را در ASP.NET Identity مدیریت کنید.

```
private MyDbContext db;
          private UserManager<MyUser> manager;
          public ToDoController()
          {
              db = new MyDbContext();
              manager = new UserManager<MyUser>(new UserStore<MyUser>(db));
          }
```

اکشن متد Create را بروز رسانی کنید

هنگامی که یک ToDo جدید ایجاد می‌کنید، کاربر جاری را در ASP.NET Identity پیدا می‌کنیم و او را به ToDoها اختصاص می‌دهیم.

```
public async Task<ActionResult> Create
          ([Bind(Include="Id,Description,IsDone")] ToDo todo)
          {
              var currentUser = await manager.FindByIdAsync
                                                  (User.Identity.GetUserId());
              if (ModelState.IsValid)
              {
                  todo.User = currentUser;
                  db.ToDoes.Add(todo);
                  await db.SaveChangesAsync();
                  return RedirectToAction("Index");
              }

              return View(todo);
          }
```

اکشن متد List را بروز رسانی کنید

در این متد تنها ToDoهای کاربر جاری را باید بگیریم.

```
public ActionResult Index()
          {
              var currentUser = manager.FindById(User.Identity.GetUserId());

               return View(db.ToDoes.ToList().Where(
                                  todo => todo.User.Id == currentUser.Id));
          }
```

تنها مدیران سایت باید بتوانند تمام ToDoها را ببینند

بدین منظور ما یک اکشن متد جدید به کنترل مربوطه اضافه می‌کنیم که تمام ToDoها را لیست می‌کند. اما دسترسی به این متد را تنها برای کاربرانی که در نقش مدیر وجود دارند میسر می‌کنیم.

```
[Authorize(Roles="Admin")]
          public async Task<ActionResult> All()
          {
```

```
            return View(await db.ToDoes.ToListAsync());
        }
```

نمایش جزئیات کاربران از جدول ToDo ها

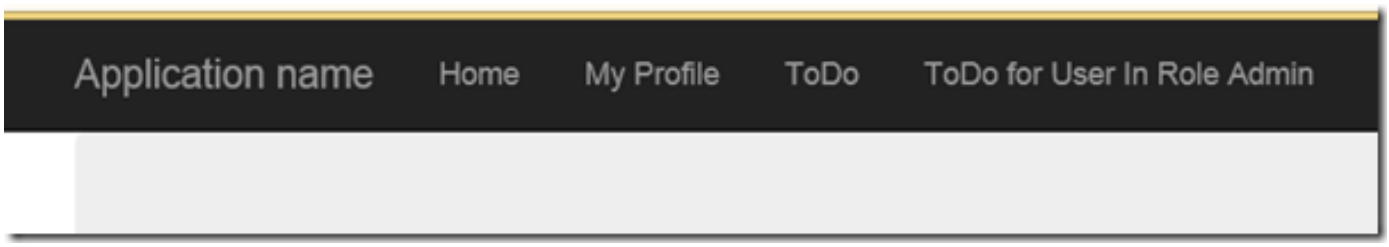
از آنجا که ما کاربران را به ToDo هایشان مرتبط می‌کنیم، دسترسی به داده‌های کاربر ساده است. مثلا در متدی که مدیر سایت تمام آیتم‌ها را لیست می‌کند می‌توانیم به اطلاعات پروفایل تک تک کاربران دسترسی داشته باشیم و آنها را در نمای خود بگنجانیم. در این مثال تنها یک فیلد بنام **HomeTown** اضافه شده است، که آن را در کنار اطلاعات ToDo نمایش می‌دهیم.

```
@model IEnumerable<AspnetIdentitySample.Models.ToDo>

  @{
    ViewBag.Title = "Index";
  }

  <h2>List of ToDoes for all Users</h2>
  <p>
      Notice that we can see the User info (UserName) and profile info such as HomeTown for the user as
well.
      This was possible because we associated the User object with a ToDo object and hence
      we can get this rich behavior.
  12:  </p>

  <table class="table">
      <tr>
          <th>
              @Html.DisplayNameFor(model => model.Description)
          </th>
          <th>
              @Html.DisplayNameFor(model => model.IsDone)
          </th>
          <th>@Html.DisplayNameFor(model => model.User.UserName)</th>
          <th>@Html.DisplayNameFor(model => model.User.HomeTown)</th>
      </tr>
  25:
  26:       @foreach (var item in Model)
  27:       {
  28:           <tr>
  29:               <td>
  30:                   @Html.DisplayFor(modelItem => item.Description)
  31:               </td>
  32:               <td>
                   @Html.DisplayFor(modelItem => item.IsDone)
              </td>
              <td>
                  @Html.DisplayFor(modelItem => item.User.UserName)
              </td>
              <td>
                  @Html.DisplayFor(modelItem => item.User.HomeTown)
              </td>
          </tr>
      }

  </table>
```

صفحه Layout را بروز رسانی کنید تا به ToDoها لینک شود

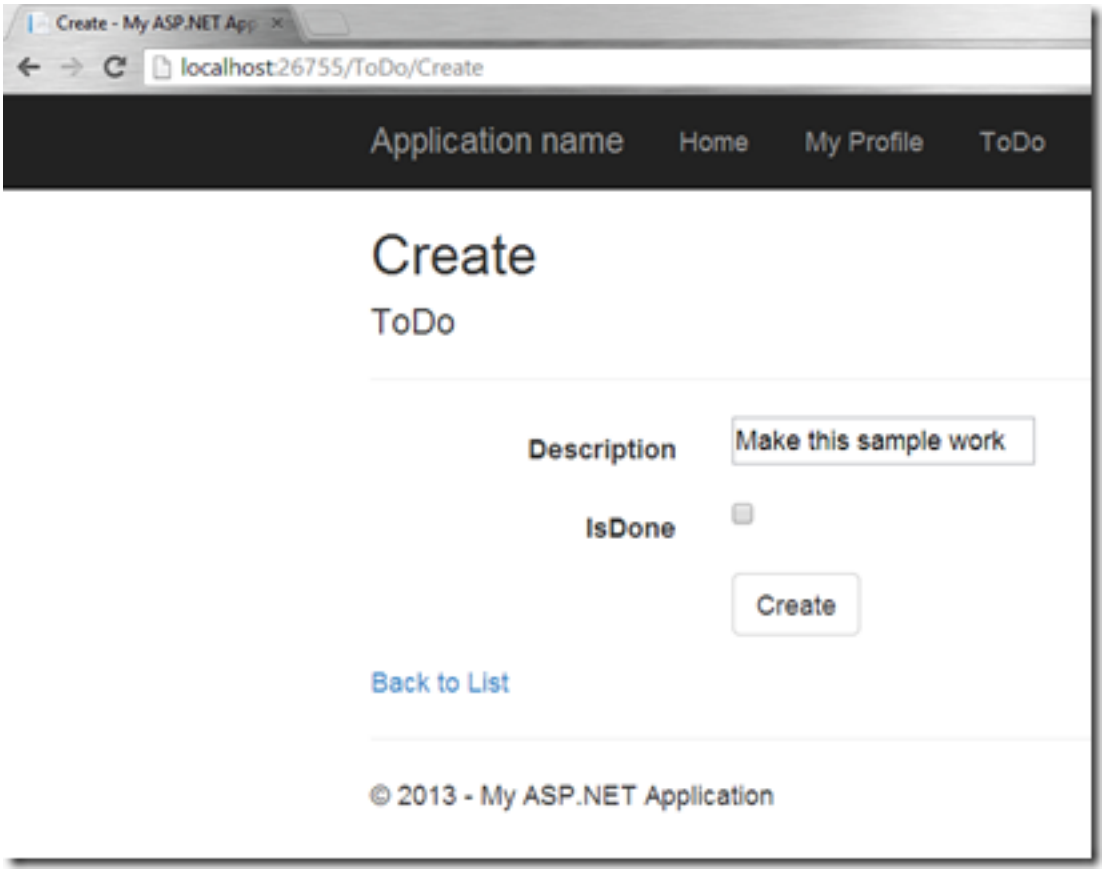
```
<li>@Html.ActionLink("ToDo", "Index", "ToDo")</li>
 <li>@Html.ActionLink("ToDo for User In Role Admin", "All", "ToDo")</li>
```

حال اپلیکیشن را اجرا کنید. همانطور که مشاهده می‌کنید دو لینک جدید به منوی سایت اضافه شده اند.

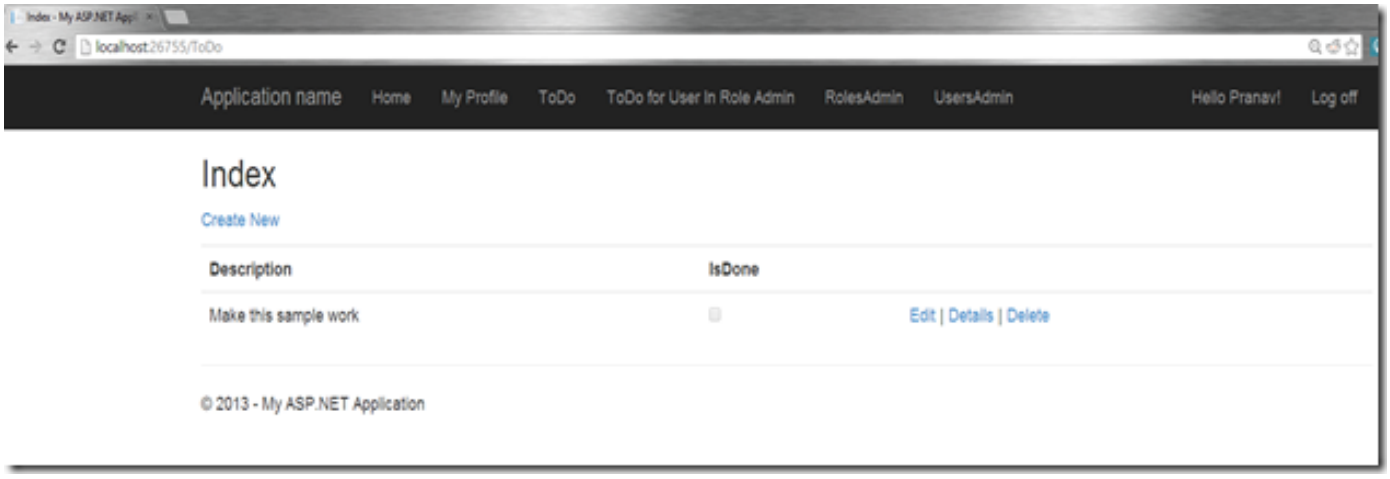


ساخت یک ToDo بعنوان کاربر عادی

روی لینک ToDo کلیک کنید، باید به صفحه ورود هدایت شوید چرا که دسترسی تنها برای کاربران احراز هویت شده تعریف وجود دارد. می‌توانید یک حساب کاربری محلی ساخته، با آن وارد سایت شوید و یک ToDo بسازید.



پس از ساختن یک ToDo می‌توانید لیست رکوردهای خود را مشاهده کنید. دقت داشته باشید که رکوردهایی که کاربران دیگر ثبت کرده اند برای شما نمایش داده نخواهند شد.

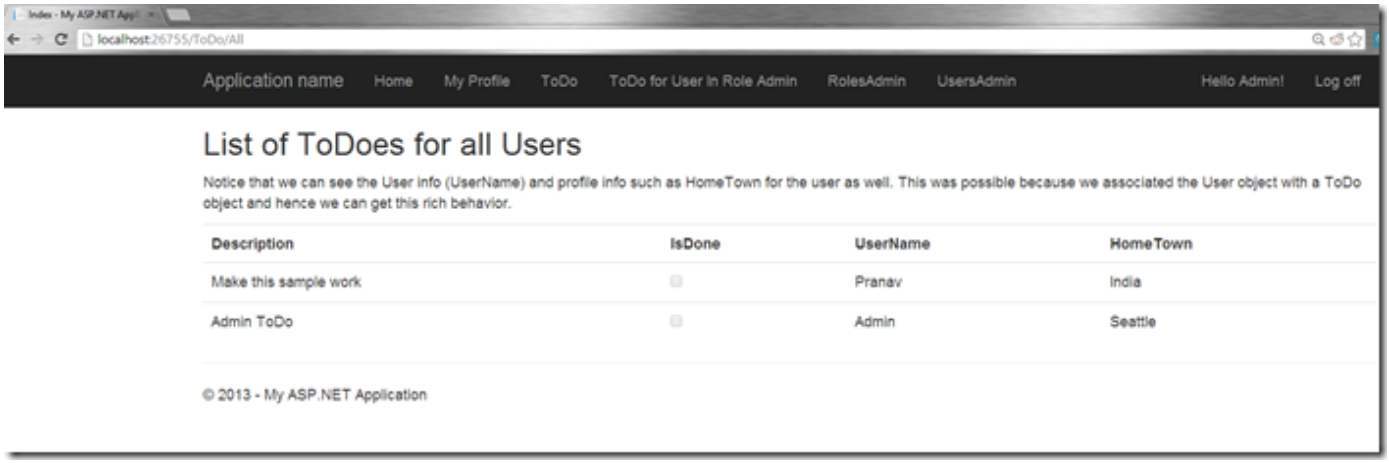


مشاهده تمام ToDoها بعنوان مدیر سایت

روی لینک **ToDoes for User in Role Admin** کلیک کنید. در این مرحله باید مجددا به صفحه ورود هدایت شوید چرا که شما در نقش مدیر نیستید و دسترسی کافی برای مشاهده صفحه مورد نظر را ندارید. از سایت خارج شوید و توسط حساب کاربری مدیری که هنگام راه اندازی اولیه دیتابیس ساخته اید وارد سایت شوید.

```
User = Admin
Password = 123456
```

پس از ورود به سایت بعنوان یک مدیر، می‌توانید ToDoهای ثبت شده توسط تمام کاربران را مشاهده کنید.



نظرات خوانندگان

نویسنده: اس ام
تاریخ: ۱۳۹۳/۰۱/۰۱ ۲۰:۲۴

میشه این پروژه رو برا دانلود بزارید؟ ممنون.

نویسنده: میثم سلیمانی
تاریخ: ۱۳۹۳/۰۴/۲۸ ۱۷:۴۰

```
var currentUser = await manager.FindByIdAsync(User.Identity.GetUserId());
```

این GetUserId() چرا وجود نداره؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۲۸ ۱۷:۴۹

در فضای نام Microsoft.AspNet.Identity تعریف شده.