عنوان: **سفارشی کردن ASP.NET Identity در MVC 5**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۹ ۱۹:۵۰
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, MemberShip, ASP.NET MVC 5, ASP.NET Identity, MVC, Security

یکی از نیازهای رایج توسعه دهندگان هنگام استفاده از سیستم عضویت ASP.NET سفارشی کردن الگوی داده‌ها است. مثلا ممکن است بخواهید یک پروفایل سفارشی برای کاربران در نظر بگیرید، که شامل اطلاعات شخصی، آدرس و تلفن تماس و غیره می‌شود. یا ممکن است بخواهید به خود فرم ثبت نام فیلدهای جدیدی اضافه کنید و آنها را در رکورد هر کاربر ذخیره کنید.
یکی از مزایای ASP.NET Identity این است که بر پایه EF Code First نوشته شده است. بنابراین سفارشی سازی الگوی دیتابیس و اطلاعات کاربران ساده است.
یک اپلیکیشن جدید ASP.NET MVC بسازید و نوع احراز هویت را Individual User Accounts انتخاب کنید. پس از آنکه پروژه جدید ایجاد شد فایل IdentityModels.cs را در پوشه Models باز کنید. کلاسی با نام *ApplicationUser* مشاهده می‌کنید که همتای *UserProfile* در فریم ورک SimpleMembership است. این کلاس خالی است و از کلاس *IdentityUser* ارث بری می‌کند و شامل خواص زیر است.

```
public class IdentityUser : IUser
    {
        public IdentityUser();
        public IdentityUser(string userName);

        public virtual ICollection<identityuserclaim> Claims { get; }
        public virtual string Id { get; set; }
        public virtual ICollection<identityuserlogin> Logins { get; }
        public virtual string PasswordHash { get; set; }
        public virtual ICollection<identityuserrole> Roles { get; }
        public virtual string SecurityStamp { get; set; }
        public virtual string UserName { get; set; }
    }
```

اگر دقت کنید خواهید دید که فیلد Id بر خلاف SimpleMembership یک عدد صحیح یا int نیست، بلکه بصورت یک رشته ذخیره می‌شود. پیاده سازی پیش فرض ASP.NET Identity مقدار این فیلد را با یک GUID پر می‌کند. در این پست تنها یک فیلد آدرس ایمیل به کلاس کاربر اضافه می‌کنیم. با استفاده از همین فیلد در پست‌های آتی خواهیم دید چگونه می‌توان ایمیل‌های تایید ثبت نام برای کاربران ارسال کرد. کلاس ApplicationUser بدین شکل خواهد بود.

```
public class ApplicationUser : IdentityUser
{
    public string Email { get; set; }
}
```

حال برای آنکه کاربر بتواند هنگام ثبت نام آدرس ایمیل خود را هم وارد کند، باید مدل فرم ثبت نام را بروز رسانی کنیم.

```
public class RegisterViewModel
{
    [Required]
    [Display(Name = "User name")]
    public string UserName { get; set; }

    [Required]
    [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength =
6)]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public string Password { get; set; }

    [DataType(DataType.Password)]
    [Display(Name = "Confirm password")]
    [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
    public string ConfirmPassword { get; set; }

    [Required]
    [Display(Name = "Email address")]
```

```
    public string Email { get; set; }

}
```

سپس فایل View را هم بروز رسانی می‌کنیم تا یک برچسب و تکست باکس برای آدرس ایمیل نمایش دهد.

```
<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" })
     </div>
 </div>
```

برای تست این تغییرات، صفحه About را طوری تغییر می‌دهید تا آدرس ایمیل کاربر جاری را نمایش دهد. این قسمت همچنین نمونه ای از نحوه دسترسی به اطلاعات کاربران است.

```
public ActionResult About()
{
    ViewBag.Message = "Your application description page.";
    UserManager<ApplicationUser> UserManager = new UserManager<ApplicationUser>(new
UserStore<ApplicationUser>(new ApplicationDbContext()));
    var user = UserManager.FindById(User.Identity.GetUserId());
    if (user != null)
        ViewBag.Email = user.Email;
    else
        ViewBag.Email = "User not found.";

    return View();
}
```

همین! تمام کاری که لازم بود انجام دهید همین بود. از آنجا که سیستم ASP.NET Identity توسط Entity Framework مدیریت می‌شود، روی الگوی دیتابیس سیستم عضویت کنترل کامل دارید. بنابراین به سادگی می‌توانید با استفاده از قابلیت Code First مدل‌های خود را سفارشی کنید.

در پست‌های آتی این مطلب را ادامه خواهیم داد تا ببینیم چگونه می‌توان ایمیل‌های تاییدیه برای کاربران ارسال کرد.

نظرات خوانندگان

نویسنده: رجایی
تاریخ: ۱۳۹۲/۱۱/۰۸ ۱۹:۲۰

سلام؛ از زحماتتون بسیار تشکر می‌کنم. من وب سایت را به روش چند لایه‌ی مرسوم می‌سازم:

data layer - domain models - website - service layer

با توجه به آن چگونه می‌توان از asp.net identity استفاده کرد؟ زیرا مثلا نیاز است از کلید جدول users در مدل‌های دیگه استفاده شود. آیا این کلید را باید در لایه دومین استفاده کرد؟

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۱/۰۸ ۲۳:۴۵

موجودیت‌های مربوط به ASP.NET Identity رو در لایه مدل‌ها قرار بدین و از یک DbContext استفاده کنید. یعنی DbSetهای مدل برنامه و Identity رو در یک کانتکست تعریف کنید.

نویسنده: رجایی
تاریخ: ۱۳۹۲/۱۱/۱۰ ۱۲:۴

سلام...ممنون از پاسختون.

با توجه به راهنمایی شما در قسمت context در لایه data layer بدین صورت درج کردم

```
public class Context : DbContext
    {
        public DbSet<IdentityUserClaim> AspNetUserClaims { set; get; }
        public DbSet<IdentityRole> AspNetRoles { set; get; }
        public DbSet<IdentityUserLogin> AspNetUserLogins { set; get; }
        public DbSet<IdentityUserRole> AspNetUserRoles { set; get; }
        public DbSet<IdentityUser> AspNetUsers { set; get; }
        //-------------------------------------
        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);
            modelBuilder.Entity<IdentityRole>().HasKey<string>(r => r.Id).ToTable("AspNetRoles");
            modelBuilder.Entity<IdentityUser>().ToTable("AspNetUsers");
            modelBuilder.Entity<IdentityUserLogin>().HasKey(l => new { l.UserId, l.LoginProvider,
l.ProviderKey }).ToTable("AspNetUserLogins");
            modelBuilder.Entity<IdentityUserRole>().HasKey(r => new { r.RoleId, r.UserId
}).ToTable("AspNetUserRoles");
            modelBuilder.Entity<IdentityUserClaim>().ToTable("AspNetUserClaims");
        }
    }
```

ولی وقتی سایت اجرا می‌شود ایراد زیر نمایش داده می‌شود

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۱/۱۰ ۱۲:۱۴

خطا رو فراموش کردید ارسال کنید.

نویسنده: مهرداد راهی
تاریخ: ۱۳۹۲/۱۱/۱۱ ۰:۵۵

سلام من MVC کار نمیکنم توی ASP.Net وب‌فرمز استفاده میکنم از این امکان. فایل IdentityModels.cs رو نداره توی پروژه . مشکل کجاس؟
-enable migrations هم زدم خطا داد

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۱/۱۱ ۱:۵۳

از VS 2013 استفاده می‌کنی ؟

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۱/۱۱ ۲:۵۱

لازم نیست تمام این آبجکت‌ها رو به context نگاشت کنید. قالب پروژه‌های VS 2013 بصورت خودکار در پوشه Models کلاسی بنام IdentityModels میسازه. این کلاس شامل کلاسی بنام ApplicationDbContext میشه که تعریفی مانند لیست زیر داره:

```
public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
{
    public ApplicationDbContext() : base("DefaultConnection") { }
}
```

این کلاس رو کلا حذف کنید، چون قراره از یک DbContext برای تمام موجودیت‌ها استفاده کنید.
کلاس ApplicationUser که معرف موجودیت کاربران هست رو در لایه دامنه‌ها تعریف کنید و دقت کنید که باید از IdentityUser ارث بری کنه، حال با نام پیش فرض یا با نام دلخواه. سپس باید کلاسی بسازید که از UserManager<T> مشتق میشه. با استفاده از این کلاس می‌تونید به موجودیت‌های کاربران دسترسی داشته باشید. بعنوان مثال:

```
public class AppUserManager : UserManager<AppUser>{
    public AppUserManager() : base(new UserStore<AppUser>(new ShirazBilitDbContext())) { }
}
```

همونطور که می‌بینید کلاس موجودیت کاربر در اینجا AppUser نام داره، پس هنگام استفاده از UserManager نوع داده رو بهش نگاشت می‌کنیم. کد کلاس AppUser هم مطابق لیست زیر خواهد بود.

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

همونطور که مشخصه کلاس کاربران سفارشی سازی شده و سه فیلد به جدول کاربران اضافه کردیم. فیلدهای بیشتر یا موجودیت پروفایل کاربران هم باید به همین کلاس افزوده بشن. اگر پست‌ها رو بیاد بیارید گفته شد که ASP.NET Identity با مدل EF Code-First کار میکنه.

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۱/۱۱ ۲:۵۷

از VS 2013 استفاده کنید و .NET 4.5.
اگر این فایل برای شما ایجاد نمیشه پس در قالب پروژه‌های Web Forms وجود نداره. ارتباطی با مهاجرت‌ها هم نداره، کلاس موجودیت کاربر رو خودتون می‌تونید ایجاد کنید. اگر به نظرات بالا مراجعه کنید گفته شد که کلاس کاربران باید از

IdentityModels ارث بری کنه.

نویسنده: رجایی
تاریخ: ۱۳۹۲/۱۱/۱۲ ۹:۲۱

عذر خواهی می‌کنم فراموش کردم. ایراد بدین صورت است:

System.InvalidOperationException: The model backing the 'Context' context has changed since the database was created. Consider using Code First Migrations to update the database (http://go.microsoft.com/fwlink/?LinkId=238269).

مشخص است که می‌گوید context تغییر می‌کند.ولی من از migration استفاده می‌کنم و codefirst ولی باز هم این ایراد رو در اتصال به دیتابیس نشان می‌دهد. من از add-migration هم استفاده می‌کنم تا تغییرات موجودیت‌ها رو کامل به من نشان دهد که چیزی را عنوان نمی‌کند.

نویسنده: افتاب
تاریخ: ۱۳۹۲/۱۲/۲۷ ۲۳:۳۸

سلام و متشکر ،
دنبال آن میگشتم که چه طوری میشه دو تا صفحه لوگین در پروژه داشت که ورودی کاربران از مدیران جدا باشد

نویسنده: افتاب
تاریخ: ۱۳۹۲/۱۲/۲۷ ۲۳:۴۴

میشه در مورد «این کلاس رو کلا حذف کنید، چون قراره از یک DbContext برای تمام موجودیت‌ها استفاده کنید....» بیشتر توضیح بفرمایید؟

نویسنده: سعید رضایی
تاریخ: ۱۳۹۳/۰۲/۰۱ ۱۶:۴۱

با عرض سلام.تو Vs2012+mvc4 نمیشه از identity استفاده کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۰۱ ۱۷:۲۶

خیر. با دات نت 4.5 کامپایل شده.

نویسنده: میثم سلیمانی
تاریخ: ۱۳۹۳/۰۵/۲۶ ۱۱:۷

با سلام و احترام
من دستور زیر رو تو asp.net mvc Identity sample 2 تو اکشن Login اضافه کردم

```
UserManager<ApplicationUser> UserManager = new UserManager<ApplicationUser>(new
UserStore<ApplicationUser>(new ApplicationDbContext()));
    var user = UserManager.FindById(User.Identity.GetUserId());
```

اما user و null میده !
اکشن login

```
public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
        {
            if (!ModelState.IsValid)
            {
                return View(model);
            }

            // This doen't count login failures towards lockout only two factor authentication
            // To enable password failures to trigger lockout, change to shouldLockout: true
            var result = await SignInManager.PasswordSignInAsync(model.Email, model.Password,
model.RememberMe, shouldLockout: false);
            switch (result)
            {

                case SignInStatus.Success:
                    {
                        UserManager<ApplicationUser> UserManager = new UserManager<ApplicationUser>(new
UserStore<ApplicationUser>(new ApplicationDbContext()));
                        var user = UserManager.FindById(User.Identity.GetUserId());
                        return RedirectToLocal(returnUrl);
                    }
                case SignInStatus.LockedOut:
                    return View("Lockout");
                case SignInStatus.RequiresVerification:
                    return RedirectToAction("SendCode", new { ReturnUrl = returnUrl });
                case SignInStatus.Failure:
                default:
                    ModelState.AddModelError("", "Invalid login attempt.");
                    return View(model);
            }
        }
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۲۹ ۱۳:۴۴

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

نویسنده: ایمان صالحی
تاریخ: ۱۳۹۳/۱۰/۱۷ ۱۵:۵۰

من DataSetهای مربوط به پایگاه داده خودم رو در ApplicationDbContext مینویسم و پایگاه داده اجرا میشه و مشکلی پیش نمیاد..کار هم میکنه..اما سوالم اینه که کارم از نظر استاندارد بودن مشکلی داره یا نه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۰/۱۷ ۱۵:۵۶

مشکلی ندارد. ApplicationDbContext از IdentityDbContext مشتق می‌شود که آن هم از DbContext مشتق شده:

```
public class IdentityDbContext<TUser, TRole, TKey, TUserLogin, TUserRole, TUserClaim> :
DbContext where TUser : IdentityUser<TKey, TUserLogin, TUserRole, TUserClaim>
where TRole : IdentityRole<TKey, TUserRole>
where TUserLogin : IdentityUserLogin<TKey>
where TUserRole : IdentityUserRole<TKey>
where TUserClaim : IdentityUserClaim<TKey>
```

یعنی نیازی به چند DbContext سفارشی در برنامه ندارید. همان ApplicationDbContext ، در حقیقت Context اصلی کاری برنامه است.

نویسنده: سانی
تاریخ: ۱۳۹۳/۱۱/۲۵ ۱۲:۴۶

سلام

این سوال رو توی asp.net , barnamenevis هم پرسیدم. من برای Identity از sample خود MVC استفاده کردم؛ به این صورت که با استفاده از nuget آن را نصب کرم. حال توی یک لیست بازشو میخوام فقط کاربرانی را نشان دهم که یک نقش را دارند.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۱/۲۵ ۱۴:۵۱

مثال مطلب اعمال تزریق وابستگی‌ها به مثال رسمی ASP.NET Identity ، جهت تکمیل کلاس ApplicationRoleManager آن بهبود داده شد (^). برای نمونه :

```
public IList<ApplicationUser> GetApplicationUsersInRole(string roleName)
{
    var selectedUserIds = from role in this.Roles
                          where role.Name == roleName
                          from user in role.Users
                          select user.UserId;
    return _users.Where(applicationUser => selectedUserIds.Contains(applicationUser.Id)).ToList();
}
```

نویسنده: مهدی عطار
تاریخ: ۱۳۹۴/۰۲/۲۶ ۲۰:۱

با سلام وتشکر از مطالب خوبتون
یه سوال برام پیش اومده راجع به identity اگه قرار باشه به عنوان مثال به اکشن ویرایش محصول دسترسی مدیر بدیم و بعد بخواهیم آن را از داخل سایت تغییر دهیم نه در بخش کد نویسی چکار باید بکنیم. ممنون میشم اگه مطلب یا سایت و مقاله و راهنمایی نمایید. با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۲۷ ۱۲:۲۹

فیلتر Authorize را سفارشی سازی کنید .

نویسنده: مهدی عطار
تاریخ: ۱۳۹۴/۰۲/۲۷ ۲۲:۱۵

چطوری تو صفحه admin امکان این رو قرار بدیم که admin بتونه authorize واسه یه عملیات تعریف کنه تغییر بده حذف کنه و رولی رو به این authorize اضافه یا حذف کنه. با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۲۷ ۲۳:۳۰

سفارشی سازی فیلتر Authorize از ارث بری از AuthorizeAttribute و سپس override کردن متد

```
public override void OnAuthorization(AuthorizationContext filterContext)
```

آن شروع می‌شود. در اینجا به اطلاعاتی مانند

```
filterContext.ActionDescriptor.ControllerDescriptor.ControllerName
filterContext.ActionDescriptor.ActionName
```

و خیلی موارد دیگر (آدرس صفحه filterContext.HttpContext.Request.Url تا کاربر filterContext.HttpContext.User و غیره) دسترسی خواهید داشت.
سپس باید طراحی جدیدی را بر اساس ControllerName و ActionName پیاده سازی کنید (یک جدول جدید طراحی کنید) تا این

اکشن متدها یا کنترلرها امکان انتساب چندین Role متغیر را داشته باشند.
حالا زمانیکه این فیلتر Authorize سفارشی سازی شده بجای فیلتر Autorize اصلی استفاده می‌شود، نام اکشن متد و کنترلر جاری را از filterContext دریافت می‌کنید. سپس این دو مورد به همراه اطلاعات User جاری، پارامترهایی خواهند شد جهت کوئری گرفتن از بانک اطلاعاتی و جدولی که از آن صحبت شد.
در اینجا هر زمانیکه نیاز بود دسترسی کاربری را قطع کنید فقط کافی است نتیجه‌ی این فیلتر سفارشی را به نحو ذیل بازگردانید:

```
filterContext.Result = new HttpStatusCodeResult(403);
```

بنابراین در قسمت ادمین، یک صفحه‌ی جدید برای ثبت نام کنترلرها و اکشن متدها به همراه نقش‌های پویای آن‌ها خواهید داشت. سپس در این فیلتر Authorize سفارشی، دقیقا مشخص است که اکنون در کدام کنترلر و اکشن متد قرار داریم. بر این اساس (و سایر پارامترهایی که می‌توان از filterContext استخراج کرد) یک کوئری گرفته می‌شود و نقش‌های پویای فیلتر Authorize دریافت می‌شوند. نقش‌های کاربر جاری هم که مشخص هستند. این‌ها را با هم مقایسه می‌کنید و خروجی 403 را درصورت عدم تطابق، تنظیم خواهید کرد.
ضمنا در صفحه‌ی طراحی انتساب نقش‌های متغیر به اکشن متدها یا کنترلرها، امکان یافتن پویای لیست آن‌ها نیز وجود دارد.