

در پروژه خود می‌توانیم StructureMap را به گونه‌ای تنظیم کنیم که کار تزریق لایه‌های انتزاعی ASP.NET را نیز انجام دهد؛ مثلاً CurrentHttpContext و یا داده‌های مربوط به مسیریابی و... به عنوان مثال در برنامه شما ممکن است کدهای زیر چندین و چند بار تکرار شده باشند:

```
var userId= User.Identity.GetUserId();
var user = _context.Users.Find(userId);

var user = int.Parse(User.Identity.GetUserId());
```

کدهای فوق به این معنی است که پروژه‌ی شما به صورت کامل به سیستم ASP.NET Identity گره خورده است. خوب، این حالت زمانی پیچیده‌تر خواهد شد که در آینده بخواهید به یک سیستم Identity جدیدتر مهاجرت کنید. در ادامه نحوه‌ی تزریق وابستگی‌های رایج ASP.NET را بررسی خواهیم کرد. ابتدا یک کلاس رجستری را به صورت زیر ایجاد خواهیم کرد:

```
public class CommonASPNETRegistry : StructureMap.Configuration.DSL.Registry
{
    public CommonASPNETRegistry()
    {
        For<IIIdentity>().Use(() => HttpContext.Current.User.Identity);
        // Other dependencies
    }
}
```

در کد فوق همانطور که مشخص است، یک کلاس رجستری ایجاد کرده‌ایم (Registry در واقع یکی از مفاهیم مربوط به استراکچر مپ می‌باشد که امکان ماژولار کردن تنظیمات را درون کلاس‌هایی مجزا، در اختیارمان قرار می‌دهد). درون سازنده‌ی این کلاس گفته‌ایم: زمانیکه درخواستی برای اینترفیس IIIdentity داده شد، یک وهله از HttpContext.Current.User.Identity را در اختیار درخواست کننده قرار بده.

لازم به ذکر است می‌توانستیم از وابستگی‌های عنوان شده نیز بدون تزریق کردن آنها درون کنترلرها نیز استفاده کنیم. اما رجستر کردن آنها این امکان را در اختیارمان قرار می‌دهد تا در هر جایی از برنامه‌مان بتوانیم به آنها دسترسی پیدا کنیم. در ادامه خواهیم دید که دسترسی آسان به آنها می‌تواند خیلی مفید واقع شود؛ همچنین امکان تست کردن نیز آسانتر خواهد شد. قدم بعدی افزودن Registry ایجاد شده به تنظیمات IoC Container مان است:

```
public static class SmObjectFactory
{
    private static readonly Lazy<Container> _containerBuilder =
        new Lazy<Container>(defaultContainer, LazyThreadSafetyMode.ExecutionAndPublication);

    public static IContainer Container
    {
        get { return _containerBuilder.Value; }
    }

    private static Container defaultContainer()
    {
        return new Container(ioc =>
        {
            // Other settings
            ioc.AddRegistry(new CommonASPNETRegistry());
        });
    }
}
```

اکنون به سادگی می‌توانیم از وابستگی‌های عنوان شده در برنامه‌مان استفاده کنیم. برای استفاده‌ی از آن، مثال اول را در نظر

بگیرید "یافتن کاربر فعلی". همانطور که عنوان شد، استفاده از کدهایی شبیه به حالت زیر جهت یافتن کاربر جاری در برنامه ممکن است چندین بار تکرار شده باشد:

```
var user = int.Parse(User.Identity.GetUserId());
```

خوب، برای حل این مشکل اینترفیس زیر را اضافه می‌کنیم:

```
public interface ICurrentUser
{
    ApplicationUser User { get; }
}
```

پیاده‌سازی آن نیز به این صورت خواهد بود:

```
public class CurrentUser : ICurrentUser
{
    private readonly IIdentity _identity;
    private readonly IApplicationUserManager _userManager;
    private ApplicationUser _user;
    public CurrentUser(IIdentity identity, IApplicationUserManager userManager)
    {
        _identity = identity;
        _userManager = userManager;
    }
    public ApplicationUser User
    {
        get { return _user ?? (_user = _userManager.FindById(int.Parse(_identity.GetUserId()))); }
    }
}
```

درون کلاس فوق به اینترفیس `IIdentity` جهت ارائه آی‌دی کاربر جاری و اینترفیس `IApplicationUserManager` جهت یافتن اطلاعات کاربر نیاز خواهیم داشت. همانطور که مشاهده می‌کنید فیلد `_user` در صورتیکه از قبل موجود باشد، برگردانده خواهد شد؛ در غیر اینصورت آن را از کانتکست مربوطه واکنشی خواهد کرد. اکنون با استفاده از روش فوق نه تنها درون کنترلرهایمان بلکه در هر جایی از برنامه‌مان می‌توانیم به کاربر جاری دسترسی داشته باشیم. همچنین در آینده نیز به راحتی می‌توانیم از سیستم `ASP.NET Identity` به هر سیستم دیگری سوئیچ کنیم. برای استفاده از اینترفیس فوق نیز به این صورت عمل خواهیم کرد:

```
public class HomeController : BaseController
{
    private readonly ICurrentUser _currentUser;
    public HomeController(ICurrentUser user)
    {
        _user = user;
    }
    public ActionResult Index()
    {
        // user
        var user = _currentUser.User;
        // user id
        var userId = _currentUser.User.Id;
    }
}
```

## نظرات خوانندگان

نویسنده: وحید نصیری  
تاریخ: ۱۳:۱۸ ۱۳۹۴/۰۵/۲۲

با تشکر. مثال [AspNetIdentityDependencyInjectionSample](#) جهت تزریق IIdentity به روز شد. با این [تغییرات](#).