عنوان: **تبدیلگر تاریخ شمسی برای AutoMapper**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۰۷ ۱۰:۲۲
آدرس: www.dotnettips.info
گروه‌ها: MVC, Entity framework, AutoMapper

فرض کنید مدل معادل با جدول بانک اطلاعاتی ما چنین ساختاری را دارد:

```
public class User
{
    public int Id { set; get; }
    public string Name { set; get; }
    public DateTime RegistrationDate { set; get; }
}
```

و ViewModel ایی که قرار است به کاربر نمایش داده شود این ساختار را دارد:

```
public class UserViewModel
{
    public int Id { set; get; }
    public string Name { set; get; }
    public string RegistrationDate { set; get; }
}
```

در اینجا می‌خواهیم حین تبدیل User به UserViewModel، تاریخ میلادی به صورت خودکار، تبدیل به یک رشته‌ی شمسی شود.
برای مدیریت یک چنین سناریوهایی توسط AutoMapper، امکان نوشتن تبدیلگرهای سفارشی نیز پیش بینی شده‌است.

### تبدیلگر سفارشی تاریخ میلادی به شمسی مخصوص AutoMapper

در ذیل یک تبدیلگر سفارشی مخصوص AutoMapper را با پیاده سازی اینترفیس ITypeConverter آن ملاحظه می‌کنید:

```
public class DateTimeToPersianDateTimeConverter : ITypeConverter<DateTime, string>
{
    private readonly string _separator;
    private readonly bool _includeHourMinute;

    public DateTimeToPersianDateTimeConverter(string separator = "/", bool includeHourMinute = true)
    {
        _separator = separator;
        _includeHourMinute = includeHourMinute;
    }

    public string Convert(ResolutionContext context)
    {
        var objDateTime = context.SourceValue;
        return objDateTime == null ? string.Empty : toShamsiDateTime((DateTime)context.SourceValue);
    }

    private string toShamsiDateTime(DateTime info)
    {
        var year = info.Year;
        var month = info.Month;
        var day = info.Day;
        var persianCalendar = new PersianCalendar();
        var pYear = persianCalendar.GetYear(new DateTime(year, month, day, new GregorianCalendar()));
        var pMonth = persianCalendar.GetMonth(new DateTime(year, month, day, new GregorianCalendar()));
        var pDay = persianCalendar.GetDayOfMonth(new DateTime(year, month, day, new
GregorianCalendar()));
        return _includeHourMinute ?
            string.Format("{0}{1}{2}{1}{3} {4}:{5}", pYear, _separator, pMonth.ToString("00",
CultureInfo.InvariantCulture), pDay.ToString("00", CultureInfo.InvariantCulture),
info.Hour.ToString("00"), info.Minute.ToString("00"))
            : string.Format("{0}{1}{2}{1}{3}", pYear, _separator, pMonth.ToString("00",
CultureInfo.InvariantCulture), pDay.ToString("00", CultureInfo.InvariantCulture));
    }
}
```

ITypeConverter دو پارامتر جنریک را قبول می‌کند. پارامتر اول نوع ورودی و پارامتر دوم، نوع خروجی مورد انتظار است. در اینجا باید خروجی متد Convert را بر اساس آرگومان دوم ITypeConverter مشخص کرد. توسط ResolutionContext می‌توان به برای مثال context.SourceValue که معادل DateTime دریافتی است، دسترسی یافت. سپس این DateTime را بر اساس متد toShamsiDateTime تبدیل کرده و بازگشت می‌دهیم.

**ثبت و معرفی تبدیلگرهای سفارشی AutoMapper**

پس از تعریف یک تبدیلگر سفارشی AutoMapper، اکنون نیاز است آن‌را به AutoMapper معرفی کنیم:

```
public class TestProfile1 : Profile
{
    protected override void Configure()
    {
        // این تنظیم سراسری هست و به تمام خواص زمانی اعمال می‌شود
        this.CreateMap<DateTime, string>().ConvertUsing(new DateTimeToPersianDateTimeConverter());
        this.CreateMap<User, UserViewModel>();
     }

    public override string ProfileName
    {
        get { return this.GetType().Name; }
    }
}
```

جهت مدیریت بهتر نگاشت‌های AutoMapper ابتدا یک کلاس Profile را آغاز خواهیم کرد و سپس توسط متدهای CreateMap، کار معرفی نگاشت‌ها را آغاز می‌کنیم.
همانطور که مشاهده می‌کنید در اینجا دو نگاشت تعریف شده‌اند. یکی برای تبدیل User به UserViewModel و دیگری، معرفی نحوه‌ی نگاشت DateTime به string، توسط تبدیلگر سفارشی DateTimeToPersianDateTimeConverter است که به کمک متد الحاقی ConvertUsing صورت گرفته‌است.
باید دقت داشت که تنظیمات تبدیلگرهای سفارشی سراسری هستند و در کل برنامه و به تمام پروفایل‌ها اعمال می‌شوند.

**بررسی خروجی تبدیلگر سفارشی تاریخ**

اکنون کار استفاده از تنظیمات AutoMapper با ثبت پروفایل تعریف شده آغاز می‌شود:

```
Mapper.Initialize(cfg => // In Application_Start()
{
     cfg.AddProfile<TestProfile1>();
});
```

سپس نحوه‌ی استفاده از متد Mapper.Map همانند قبل خواهد بود:

```
var dbUser1 = new User
{
    Id = 1,
    Name = "Test",
    RegistrationDate = DateTime.Now.AddDays(-10)
};

var uiUser = new UserViewModel();

Mapper.Map(source: dbUser1, destination: uiUser);
```

در اینجا در حین کار تبدیل و نگاشت dbUser به uiUser، زمانیکه AutoMapper به هر خاصیت DateTime ایی می‌رسد، مقدار آن‌را با توجه به تبدیلگر سفارشی تاریخی که به آن معرفی کردیم، تبدیل به معادل رشته‌ای شمسی می‌کند.

**نوشتن تبدیلگرهای غیر سراسری**

همانطور که عنوان شد، معرفی تبدیلگرها به AutoMapper سراسری است و در کل برنامه اعمال می‌شود. اگر نیاز است فقط برای یک مدل خاص و یک خاصیت خاص آن تبدیلگر نوشته شود، باید نگاشت مورد نظر را به صورت ذیل تعریف کرد:

```
this.CreateMap<User, UserViewModel>()
             .ForMember(userViewModel => userViewModel.RegistrationDate,
                        opt => opt.ResolveUsing(src =>
                        {
                             var dt = src.RegistrationDate;
                             return dt.ToShortDateString();
                        }));
```

اینبار در همان کلاس پروفایل ابتدای بحث، نگاشت User به ViewModel آن با کمک متد ForMember، سفارشی سازی شده‌است. در اینجا عنوان شده‌است که اگر به خاصیت ویژه‌ی RegistrationDate رسیدی، مقدار آن‌را با توجه به فرمولی که مشخص شده، محاسبه کرده و بازگشت بده. این تنظیم خصوصی است و به کل برنامه اعمال نمی‌شود.

**خصوصی سازی تبدیلگرها با تدارک موتورهای نگاشت اختصاصی**

اگر می‌خواهید تنظیمات TestProfile1 به کل برنامه اعمال نشود، نیاز است یک MappingEngine جدید و مجزای از MappingEngine سراسری AutoMapper را ایجاد کرد:

```
var configurationStore = new ConfigurationStore(new TypeMapFactory(), MapperRegistry.Mappers);
configurationStore.AddProfile<TestProfile1>();
var mapper = new MappingEngine(configurationStore);
mapper.Map(source: dbUser1, destination: uiUser);
```

به صورت پیش فرض و در پشت صحنه، متد Mapper.Map از یک MappingEngine سراسری استفاده می‌کند. اما می‌توان در یک برنامه چندین MappingEngine مجزا داشت که نمونه‌ای از آن‌را در اینجا مشاهده می‌کنید.

**کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:**

AM_Sample02.zip