عنوان: **نگاشت دیتای XML به کمک AutoMapper**
نویسنده: محمد صاحب
تاریخ: ۱۳۹۱/۰۷/۱۴ ۱۳:۲۱
آدرس: www.dotnettips.info
برچسب‌ها: AutoMapper, LINQ to XML

صورت مساله که مشخصه قراره دیتای رو از منبع داده‌ی Xml به model مورد نظرمون نگاشت کنیم چیزی شبیه کاری که متد GetEntries انجام میده و تو این پست معرفی شده...

AutoMapper به صورت داخلی و با استفاده از قراردادها نمیتونه xml رو به object تبدیل کنه ولی این کار به کمک LINQ to XML قابل انجامه.

مثالی که برای این پست انتخاب شده سوژه‌ی داغ روزهای اخیره ؟!
مدل زیر رو در نظر داشته باشید

```
public class PreciousMetal
    {
        public string Name { get; set; }
        public float Price { get; set; }
        public DateTime UpdateTime { get; set; }
    }
```

قراره از یک وب سرویس اطلاعات مربوط به فلزات گرانبها رو دریافت و به مدل PreciousMetal نگاشت کنیم.ساختار اطلاعات دریافتی ما به شکل زیره

```
<pricelist currency="usd">
  <price timestamp="1349347920" per="ozt" commodity="gold">1788.70</price>
  <price timestamp="1349347860" per="ozt" commodity="palladium">665.50</price>
  <price timestamp="1349347920" per="ozt" commodity="platinum">1701.25</price>
  <price timestamp="1349347920" per="ozt" commodity="silver">34.91</price>
</pricelist>
```

برای نگاشت‌های معمولی کار سختی نداریم و از MapFrom استفاده میکنیم مثلا برای قیمت

```
Mapper.CreateMap<XElement, PreciousMetal>().ForMember(des => des.Price,
                                                       op =>
                                                       op.MapFrom(src => src.Value));
```

ولی برای زمان دریافت قیمت با توجه به متفاوت بودن زمان دریافتی مثلا در اینجا Unix time از Custom value resolvers استفاده میکنیم

```
public class UnixTimestampResolver : ValueResolver<XElement, DateTime>
    {
        protected override DateTime ResolveCore(XElement source)
        {
            var origin = new DateTime(1970, 1, 1, 0, 0, 0, 0);
            return origin.AddSeconds(Convert.ToDouble(source.Attribute("timestamp").Value));
        }
    }
```

همچنیا میخواهیم از معادل فارسی نام فلزات گرانبها استفاده کنیم

```
public class EnglishPMetalToFarsiResolver : ValueResolver<XElement, string>
    {
        readonly Dictionary<string, string> _pMetaldic = new Dictionary<string, string>
                                                         {
                                                             {"gold", "طلا"},
                                                             {"palladium", "پالادیوم"},
                                                             {"platinum", "پلاتین "},
                                                             {"silver", "نقره"}
```

```
                                                    };
        protected override string ResolveCore(XElement source)
        {
            string pMetalFarsi;
            return _pMetaldic.TryGetValue(source.Attribute("commodity").Value, out pMetalFarsi) ?
pMetalFarsi : string.Empty;
        }
    }
```

نکته: از سری قبلی آشنایی با AutoMapper همیشه بین انتخاب Custom Value Formatters و Custom value resolvers مشکل داشتم مثلا همین قسمت بنظر خودم Custom Value Formatters مناسبتر میاد بعد کمی وقت گذاشتن مشخص شد گویا یه جورایی Custom Value Formatters اضافه س و اشتباه تو طراحی بوده.

و اما نحوه استفاده

```
static void Main(string[] args)
        {
            //تعریف نگاشت‌ها Mapper.CreateMap<XElement, PreciousMetal>().ForMember(des => des.Name,
                                                                op =>
op.ResolveUsing<EnglishPMetalToFarsiResolver>())
                .ForMember(des => des.Price,
                           op =>
                           op.MapFrom(src => src.Value))

                .ForMember(des => des.UpdateTime, op => op.ResolveUsing<UnixTimestampResolver>());

            Mapper.AssertConfigurationIsValid();

            //دریافت قیمت‌ها از منبع داده
            var doc = XDocument.Load("http://www.xmlcharts.com/cache/precious-metals.xml");
            var priceData = doc.Descendants("pricelist").Take(1).Elements("price");

            //فراخوانی نگاشت
            var preciousMetals = Mapper.Map<IEnumerable<XElement>, IList<PreciousMetal>>(priceData);


            foreach (var preciousMetal in preciousMetals)
            {
                Console.WriteLine(preciousMetal.Name + " " + preciousMetal.Price + " " +
preciousMetal.UpdateTime.ToShortDateString());
            }

            Console.ReadLine();

        }
```