عنوان: به روز رسانی خواص راهبری و مجموعه‌های Entity Framework توسط AutoMapper
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۰۹ ۱۷:۲
آدرس: www.dotnettips.info
گروه‌ها: MVC, Entity framework, AutoMapper

فرض کنید مدل‌های بانک اطلاعاتی ما چنین ساختاری را دارند:

```
public abstract class BaseEntity
{
    public int Id { set; get; }
}

public class User : BaseEntity
{
        public string Name { set; get; }

        public virtual ICollection<Advertisement> Advertisements { get; set; }
}

public class Advertisement : BaseEntity
{
    public string Title { get; set; }
    public string Description { get; set; }

    [ForeignKey("UserId")]
    public virtual User User { get; set; }
    public int UserId { get; set; }
}
```

و همچنین مدل‌های رابط کاربری یا ViewModelهای برنامه نیز به صورت ذیل تعریف شده‌اند:

```
public class AdvertisementViewModel
{
    public int Id { get; set; }
    public string Title { get; set; }
    public int UserId { get; set; }
}

public class UserViewModel
{
    public int Id { set; get; }
    public string Name { set; get; }
    public List<AdvertisementViewModel> Advertisements { get; set; }
}
```

به روز رسانی خواص راهبری Entity framework توسط AutoMapper

در کلاس‌های فوق، یک کاربر، تعدادی تبلیغات را می‌تواند ثبت کند. در این حالت اگر بخواهیم خاصیت User کلاس Advertisement را توسط AutoMapper به روز کنیم، با رعایت دو نکته، اینکار به سادگی انجام خواهد شد:

الف) همانطور که در کلاس Advertisement جهت تعریف کلید خارجی مشخص است، UserId نیز علاوه بر User ذکر شده‌است. این مورد کار نگاشت UserId اطلاعات دریافتی از کاربر را ساده کرده و در این حالت نیازی به یافتن اصل User این UserId از بانک اطلاعاتی نخواهد بود.

ب) چون در اطلاعات دریافتی از کاربر تنها Id او را داریم و نه کل شیء مرتبط را، بنابراین باید به AutoMapper اعلام کنیم تا از این خاصیت صرفنظر کند که اینکار توسط متد Ignore به نحو ذیل قابل انجام است:

```
this.CreateMap<AdvertisementViewModel, Advertisement>()
      .ForMember(advertisement => advertisement.Description, opt => opt.Ignore())
      .ForMember(advertisement => advertisement.User, opt => opt.Ignore());
```

به روز رسانی مجموعه‌های Entity Framework توسط AutoMapper

فرض کنید چنین اطلاعاتی از کاربر و رابط کاربری برنامه دریافت شده است:

```
var uiUser1 = new UserViewModel
{
    Id = 1,
    Name = "user 1",
    Advertisements = new List<AdvertisementViewModel>
    {
        new AdvertisementViewModel
        {
            Id = 1,
            Title = "Adv 1",
            UserId = 1
        },
        new AdvertisementViewModel
        {
            Id = 2,
            Title = "Adv 2",
            UserId = 1
        }
    }
};
```

اکنون می‌خواهیم معادل این رکورد را از بانک اطلاعاتی یافته و سپس اطلاعات آن‌را بر اساس اطلاعات UI به روز کنیم. شاید در نگاه اول چنین روشی پیشنهاد شود:

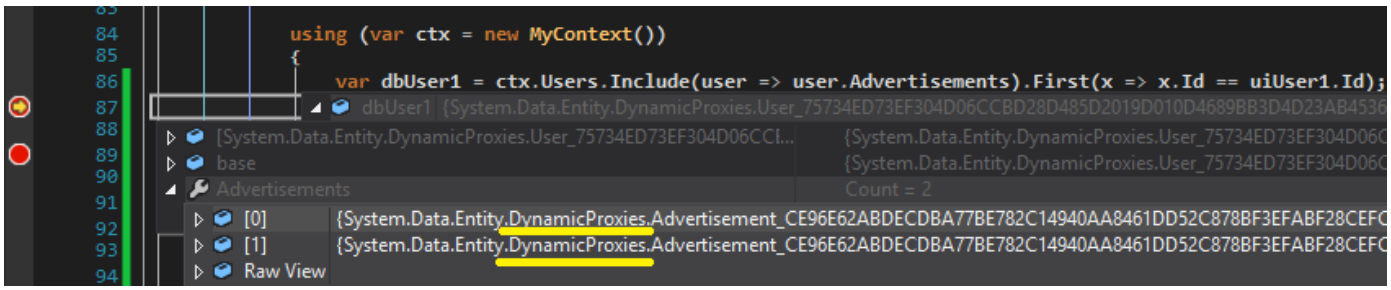
```
var dbUser1 = ctx.Users.Include(user => user.Advertisements).First(x => x.Id == uiUser1.Id);
Mapper.Map(source: uiUser1, destination: dbUser1);
```

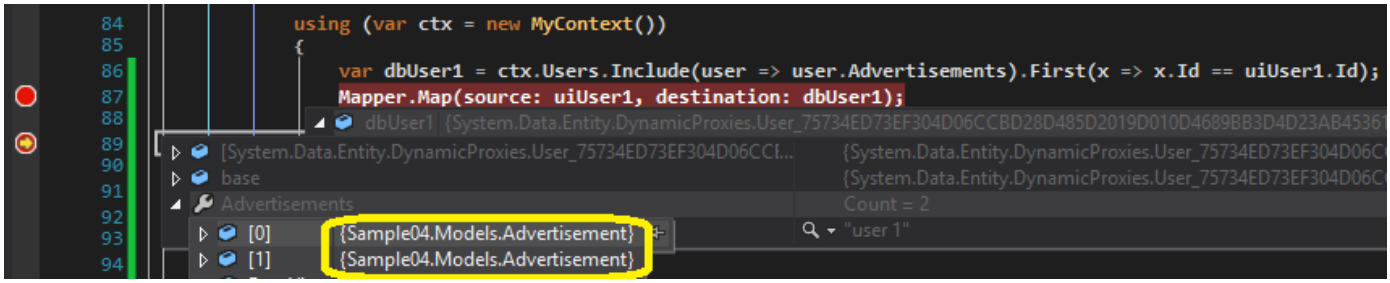
ابتدا کاربری را که Id آن مساوی uiUser1.Id است، یافته و سپس به AutoMapper اعلام می‌کنیم تا تمام اطلاعات آن‌را به صورت یکجا به روز کند. این نگاشت را نیز برای آن تعریف خواهیم کرد:

```
this.CreateMap<UserViewModel, User>()
```

در یک چنین حالتی، ابتدا شیء user 1 از بانک اطلاعاتی دریافت شده (و با توجه به وجود Include، تمام تبلیغات او نیز دریافت می‌شوند)، سپس ... دو رکورد دریافتی از کاربر، کاملا جایگزین اطلاعات موجود می‌شوند. این جایگزینی سبب تخریب پروکسی‌های EF می‌گردند. برای مثال اگر پیشتر تبلیغی با Id=1 در بانک اطلاعاتی وجود داشته، اکنون با نمونه‌ی جدیدی جایگزین می‌شود که سیستم Tracking و ردیابی EF اطلاعاتی در مورد آن ندارد. به همین جهت اگر در این حالت ctx.SaveChanges فراخوانی شود، عملیات ثبت و یا به روز رسانی با شکست مواجه خواهد شد.

علت را در این دو تصویر بهتر می‌توان مشاهده کرد:





تصویر اول که مستقیما از بانک اطلاعاتی حاصل شده‌است، دارای پروکسی‌های EF است. اما در تصویر دوم، جایگزین شدن این پروکسی‌ها را مشاهده می‌کنید که سبب خواهد شد این اشیاء دیگر تحت نظارت EF نباشند.

راه حل:

در این مورد خاص باید به AutoMapper اعلام کنیم تا کاری با لیست تبلیغات کاربر دریافت شده‌ی از بانک اطلاعاتی نداشته باشد و آن‌را راسا جایگزین نکند:

```
this.CreateMap<UserViewModel, User>().ForMember(user => user.Advertisements, opt => opt.Ignore());
```

در اینجا متد Ignore را بر روی لیست تبلیغات کاربر بانک اطلاعاتی فراخوانی کرده‌ایم، تا اطلاعات آن پس از اولین نگاشت انجام شده‌ی توسط AutoMapper دست نخورده باقی بماند.
سپس کار ثبت یا به روز رسانی را به صورت نیمه خودکار مدیریت می‌کنیم:

```
using (var ctx = new MyContext())
{
    var dbUser1 = ctx.Users.Include(user => user.Advertisements).First(x => x.Id == uiUser1.Id);
    Mapper.Map(source: uiUser1, destination: dbUser1);

    foreach (var uiUserAdvertisement in uiUser1.Advertisements)
    {
        var dbUserAdvertisement = dbUser1.Advertisements.FirstOrDefault(ad => ad.Id ==
uiUserAdvertisement.Id);
        if (dbUserAdvertisement == null)
        {
            // Add new record
            var advertisement = Mapper.Map<AdvertisementViewModel, Advertisement>(uiUserAdvertisement);
            dbUser1.Advertisements.Add(advertisement);
        }
        else
        {
            // Update the existing record
            Mapper.Map(uiUserAdvertisement, dbUserAdvertisement);
        }
    }

    ctx.SaveChanges();
}
```

- در اینجا ابتدا db user معادل اطلاعات ui user از بانک اطلاعاتی، به همراه لیست تبلیغات او دریافت می‌شود و اطلاعات ابتدایی او نگاشت خواهند شد.
- سپس بر روی اطلاعات تبلیغات دریافتی از کاربر، یک حلقه را تشکیل خواهیم داد. در اینجا هربار بررسی می‌کنیم که آیا معادل این تبلیغ هم اکنون به شیء db user متصل است یا خیر؟ اگر متصل نبود، یعنی یک رکورد جدید است و باید Add شود. اگر متصل بود صرفا باید به روز رسانی صورت گیرد.
- برای حالت ایجاد شیء جدید بانک اطلاعاتی، بر اساس uiUserAdvertisement دریافتی، می‌توان از متد Mapper.Map استفاده کرد؛ خروجی این متد، یک شیء جدید تبلیغ است.
- برای حالت به روز رسانی اطلاعات db user موجود، بر اساس اطلاعات ارسالی کاربر نیز می‌توان از متد Mapper.Map کمک

گرفت.

نکته‌ی مهم

چون در اینجا از متد Include استفاده شده‌است، فراخوانی‌های FirstOrDefault داخل حلقه، سبب رفت و برگشت اضافه‌تری به بانک اطلاعاتی نخواهند شد.

کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:

AM_Sample04.zip