عنوان: انتقال خواص محاسباتی Entity Framework به ViewModelها توسط AutoMapper
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۲۰ ۱۷:۳
آدرس: www.dotnettips.info
گروه‌ها: MVC, Entity framework, AutoMapper

در طی دو مطلب (^ و ^) با نحوه‌ی قرار دادن خواص محاسباتی، درون کلاس‌های مدل‌های بانک اطلاعاتی مورد استفاده‌ی توسط Entity Framework آشنا شدیم. در اینجا قصد داریم این خواص محاسباتی را از کلاس‌های اصلی مدل‌های بانک اطلاعاتی خود خارج و به ViewModelها منتقل کنیم؛ چون اساسا هدف از این نوع خواص ویژه، ارائه اطلاعات نمایشی است به کاربر و نه ذخیره سازی آن‌ها در بانک اطلاعاتی.

مدل‌ها و تنظیمات برنامه

مدل‌ها و تنظیمات مورد استفاده‌ی در مثال جاری، با مدل‌های مطلب « لغو Lazy Loading در حین کار با AutoMapper و Entity Framework » یکی است. فقط ViewModel مورد استفاده اینبار یک‌چنین ساختاری را دارد:

```
public class UserViewModel
{
    public int Id { set; get; }
    public string CustomName { set; get; }
    public int PostsCount { set; get; }
}
```

در اینجا می‌خواهیم در حین نگاشت اطلاعات جدول کاربران بانک اطلاعاتی به UserViewModel :
- خاصیت CustomName از جمع نام و سن شخص تشکیل شود.
- خاصیت PostsCount بیانگر جمع مطالب ارسالی آن شخص باشد.

نگاشت‌های AutoMapper می‌توانند حاوی توابع تجمعی نیز باشند

برای حل مساله‌ی فوق تنها کافی است نگاشت ذیل را تهیه کنیم:

```
public class TestProfile : Profile
{
    protected override void Configure()
    {
        this.CreateMap<User, UserViewModel>()
            .ForMember(dest => dest.CustomName,
                       opt => opt.MapFrom(src => src.Name + "[" + src.Age + "]"))
             .ForMember(dest => dest.PostsCount,
                        opt => opt.MapFrom(src => src.BlogPosts.Count()));
    }

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

در این نگاشت عنوان شده‌است که اطلاعات CustomName را مطابق فرمول خاص جمع نام شخص و سن او تهیه کن. همچنین مقدار PostsCount، باید از جمع تعداد مطالب ارسالی او تشکیل شود.

کوئری نهایی استفاده کننده از تنظیمات نگاشت تهیه شده

در ادامه متدهای Project To را جهت استفاده‌ی از تنظیمات نگاشت فوق بکار می‌گیریم:

```
using (var context = new MyContext())
{
    var user1 = context.Users
```

```
                        .Project()
                        .To<UserViewModel>()
                        .FirstOrDefault();

    if (user1 != null)
    {
        Console.Write(user1.CustomName);
        Console.Write(user1.PostsCount);
    }
}
```

این کوئری یک چنین خروجی SQL ایی را به همراه دارد:

```
SELECT
                    [Limit1].[Id] AS [Id],
                    [Limit1].[C1] AS [C1],
                    [Limit1].[C2] AS [C2]
                    FROM ( SELECT TOP (1)
                        [Project1].[Id] AS [Id],
                        CASE WHEN ([Project1].[Name] IS NULL) THEN N'' ELSE [Project1].[Name] END
                     + N'[' +  CAST( [Project1].[Age] AS nvarchar(max)) + N']' AS [C1],
                        [Project1].[C1] AS [C2]
                        FROM ( SELECT
                            [Extent1].[Id] AS [Id],
                            [Extent1].[Name] AS [Name],
                            [Extent1].[Age] AS [Age],
                            (SELECT
                                COUNT(1) AS [A1]
                                FROM [dbo].[BlogPosts] AS [Extent2]
                                WHERE [Extent1].[Id] = [Extent2].[UserId]) AS [C1]
                            FROM [dbo].[Users] AS [Extent1]
                        )  AS [Project1]
                    )  AS [Limit1]
```

همانطور که مشاهده می‌کنید، تنظیمات نگاشت تهیه شده (نحوه‌ی تهیه‌ی نام و جمع تعداد مطالب شخص) به SQL ترجمه شده‌اند.

کدهای کامل این مطلب را از اینجا می‌توانید دریافت کنید.