عنوان: **استفاده از خاصیت Local در Entity Framework**
نویسنده: شهروز جعفری
تاریخ: ۱۳۹۱/۰۵/۰۱ ۱۸:۳۷
آدرس: www.dotnettips.info
برچسب‌ها: Entity framework, DbContext, LINQ to Objects

زمانی که از LINQ To Entity استفاده می‌کنیم، با هر بار اجرای یک کوئری، این کوئری به سمت دیتابیس ارسال شده و اطلاعات مورد نظر را بازیابی می‌کند. حال اگر ما موجودیت جدیدی را به Context جاری اضافه کرده ولی آن را ذخیره نکرده باشیم، به علت عدم وجود موجودیت در دیتابیس (در حافظه وجود دارد) کوئری ارسالی ما این موجودیت جدید را شامل نمی‌شود. البته شایان ذکر است زمانیکه از متد Find استفاده می‌کنیم، به صورت پیش فرض ابتدا داخل حافظه کاوش شده و در صورت عدم وجود اطلاعات، کوئری را در دیتابیس اجرا می‌کند.
نکته قابل توجه این است که بعنوان مثال ما نیاز داریم یک لیست از موجودیت‌ها را به اشکال زیر داشته باشیم.
به صورت لیست
به صورت لیست sort شده براساس Name
فیلتر بر روی لیستی از فیلدهای موجود
این لیست باید شامل تمامی داده‌های موجود (چه در رم و چه در دیتابیس) باشد.

نکته: توسط متد ToList میتوان به لیستی از موجودیت‌های مورد نظر دست یافت ولی امکان استفاده از تمامی داده‌های موجود (چه در رم و چه در دیتابیس) میسر نمی‌باشد.
کلاس DbSet خاصیتی به نام Local دارد که امکان استفاده از تمامی داده‌های موجود را به ما می‌دهد و شامل هر داده‌ای که از دیتابیس Load شده، هر داده‌ای که اضافه شده، هر داده‌ای که پاک شده (Delete Flag) ولی هنوز ذخیره نشده می‌شود. بنابراین در هنگام استفاده باید توجه داشت به علت اینکه هیچ نوع کوئری به دیتابیس ارسال نشده، قطعه کد زیر دارای مقدار Destinations in memory: 0 خواهد بود

```
private static void GetLocalDestinationCount()
    {
        using (var context = new BreakAwayContext())
        {
            var count = context.Destinations.Local.Count;
            Console.WriteLine("Destinations in memory: {0}", count);
        }
    }
```

استفاده از متد Load

برای مثال می‌توانیم از Foreach استفاده کنیم و تمام اطلاعات مورد نظر را بدست آوریم

```
private static void GetLocalDestinationCount()
    {
        using (var context = new BreakAwayContext())
        {
            foreach (var destination in context.Destinations)
            {
                Console.WriteLine(destination.Name);
            }
            var count = context.Destinations.Local.Count;
            Console.WriteLine("Destinations in memory: {0}", count);
        }
    }
```

کد بالا یک حلقه بر روی موجودیت‌های Destinations است که نیازی به توضیح خاصی ندارد.
حال با استفاده از متد Load قادر به جمع آوری اطلاعات دیتابیس به داخل رم نیز خواهیم بود و کد بالا تمیزتر خواهد شد.

```
private static void GetLocalDestinationCountWithLoad()
    {
        using (var context = new BreakAwayContext())
        {
            context.Destinations.Load();
            var count = context.Destinations.Local.Count;
            Console.WriteLine("Destinations in memory: {0}", count);
        }
    }
```

متد Load یک extension method روی IQueryable<T> است که در فضای نام System.Data.Entity موجود است. پس امکان اجرای یک LINQ query و سپس Load کردن آن را در حافظه را خواهیم داشت.
به کد زیر توجه کنید:

```
private static void LoadAustralianDestinations()
    {
        using (var context = new BreakAwayContext())
        {
            var query = from d in context.Destinations
                        where d.Country == "Australia"
                        select d;
            query.Load();
            var count = context.Destinations.Local.Count;
            Console.WriteLine("Aussie destinations in memory: {0}", count);
        }
    }
```

همچنین ما قادر به استفاده از LINQ query روی داده‌های Local که این متد در حافظه جمع آوری کرده، نیز خواهیم بود.

به کد زیر توجه کنید.

```
private static void LocalLinqQueries()
    {
        using (var context = new BreakAwayContext())
        {
            context.Destinations.Load();
            var sortedDestinations = from d in context.Destinations.Local
                                     orderby d.Name
                                     select d;
            Console.WriteLine("All Destinations:");
            foreach (var destination in sortedDestinations)
            {
                Console.WriteLine(destination.Name);

            }
        }
        var aussieDestinations = from d in context.Destinations.Local
                                 where d.Country == "Australia"
                                 select d;
        Console.WriteLine();
        Console.WriteLine("Australian Destinations:");
        foreach (var destination in aussieDestinations)
        {
            Console.WriteLine(destination.Name);
        }
    }
```

ابتدا کلیه داده‌ها Load می‌شود سپس به وسیله Foreach نام‌ها استخراج می‌شوند. سپس ار همان داده‌ها جهت اعمال فیلتر، استفاده مشود.

تفاوت بین Linq providerهای مختلف:
عموماً دیتابیسها حساس به حروف کوچک و بزرگ نیستند؛ به عنوان مثال اگر Great و great در دیتابیس وجود داشته باشند اگر به دیتابیس کوئری اسال شود و درخواست great داشته باشد هر دو را شامل میشود. حال اگر از Local استفاده شود به جهت اینکه در واقع از Linq to Object استفاده می‌کند فقط great را شامل خواهد شد.
تفاوت دیگر این است که LINQ to Object از متد Last پشتیبانی می‌کند ولی LINQ to Entities خیر.
در پست بعدی قصد دارم در مورد ObservableCollection توضیحاتی کلی بدهم.