عنوان: Func یا Expression Func در EF
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۲۶ ۸:۰
آدرس: www.dotnettips.info
برچسب‌ها: Entity framework, LINQ

با بررسی کدهای مختلف Entity framework گاهی از اوقات در امضای توابع کمکی نوشته شده، Func<> مشاهده می‌شود و در بعضی از موارد Expression<Func<>> و ... به نظر استفاده کنندگان دقیقا نمی‌دانند که تفاوت این دو در چیست و کدامیک را باید/یا بهتر است بکار برد.

ابتدا مثال کامل ذیل را در نظر بگیرید:

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Data.Entity.Migrations;
using System.Linq;
using System.Linq.Expressions;

namespace Sample
{
    public abstract class BaseEntity
    {
        public int Id { set; get; }
    }

    public class Receipt : BaseEntity
    {
        public int TotalPrice { set; get; }
    }

    public class MyContext : DbContext
    {
        public DbSet<Receipt> Receipts { get; set; }
    }

    public class Configuration : DbMigrationsConfiguration<MyContext>
    {
        public Configuration()
        {
            AutomaticMigrationsEnabled = true;
            AutomaticMigrationDataLossAllowed = true;
        }

        protected override void Seed(MyContext context)
        {
            if (!context.Receipts.Any())
            {
                for (int i = 0; i < 20; i++)
                {
                    context.Receipts.Add(new Receipt { TotalPrice = i });
                }
            }
            base.Seed(context);
        }
    }

    public static class EFUtils
    {
        public static IList<T> LoadEntities<T>(this DbContext ctx, Expression<Func<T, bool>> predicate)
where T : class
        {
            return ctx.Set<T>().Where(predicate).ToList();
        }

        public static IList<T> LoadData<T>(this DbContext ctx, Func<T, bool> predicate) where T : class
        {
            return ctx.Set<T>().Where(predicate).ToList();
        }
    }

    public static class Test
    {
        public static void RunTests()
        {
```

```
            startDB();

            using (var context = new MyContext())
            {
                var list1 = context.LoadEntities<Receipt>(x => x.TotalPrice == 10);
                var list2 = context.LoadData<Receipt>(x => x.TotalPrice == 20);
            }
        }

        private static void startDB()
        {
            Database.SetInitializer(new MigrateDatabaseToLatestVersion<MyContext, Configuration>());
            // Forces initialization of database on model changes.
            using (var context = new MyContext())
            {
                context.Database.Initialize(force: true);
            }
        }
    }
}
```

در این مثال ابتدا کلاس Receipt تعریف شده و سپس توسط کلاس MyContext در معرض دید EF قرار گرفته است. در ادامه توسط کلاس Configuration نحوه آغاز بانک اطلاعاتی مشخص گردیده است؛ به همراه ثبت تعدادی رکورد نمونه.
نکته اصلی مورد بحث، کلاس کمکی EFUtils است که در آن دو متد الحاقی LoadEntities و LoadData تعریف شده‌اند. در متد LoadEntities، امضای متد شامل Expression Func است و در متد LoadData فقط Func ذکر شده است.
در ادامه اگر برنامه را توسط فراخوانی متد RunTests اجرا کنیم، به نظر شما خروجی SQL حاصل از list1 و list2 چیست؟
احتمالا شاید عنوان کنید که هر دو یک خروجی SQL دارند (با توجه به اینکه بدنه متدهای LoadEntities و LoadData دقیقا/یا به نظر یکی هستند) اما یکی از پارامتر 10 استفاده می‌کند و دیگری از پارامتر 20. تفاوت دیگری ندارند.
اما ... اینطور نیست!
خروجی SQL متد LoadEntities در متد RunTests به صورت زیر است:

```
SELECT
[Extent1].[Id] AS [Id],
[Extent1].[TotalPrice] AS [TotalPrice]
FROM [dbo].[Receipts] AS [Extent1]
WHERE 10 = [Extent1].[TotalPrice]
```

و ... خروجی متد LoadData به نحو زیر:

```
SELECT
[Extent1].[Id] AS [Id],
[Extent1].[TotalPrice] AS [TotalPrice]
FROM [dbo].[Receipts] AS [Extent1]
```

بله. در لیست دوم هیچ فیلتری انجام نشده (در حالت استفاده از Func خالی) و کل اطلاعات موجود در جدول Receipts، بازگشت داده شده است.
چرا؟
Func اشاره‌گری است به یک متد و Expression Func بیانگر ساختار درختی عبارت lambda نوشته شده است. این ساختار درختی صرفا بیان می‌کند که عبارت lambda منتسب، چه کاری را قرار است یا می‌تواند انجام دهد؛ بجای انجام واقعی آن.

```
public static IQueryable<TSource> Where<TSource>(this IQueryable<TSource> source,
Expression<Func<TSource, bool>> predicate);
public static IEnumerable<TSource> Where<TSource>(this IEnumerable<TSource> source, Func<TSource, bool>
predicate);
```

اگر از Expression Func استفاده شود، از متد Where ایی استفاده خواهد شد که خروجی IQueryable دارد. اگر از Func استفاده شود، از overload دیگری که خروجی و ورودی IEnumerable دارد به صورت خودکار استفاده می‌گردد.
بنابراین هرچند بدنه دو متد LoadEntities و LoadData به ظاهر یکی هستند، اما بر اساس نوع ورودی Where ایی که دریافت می‌کنند، اگر Expression Func باشد، EF فرصت آنالیز و ترجمه عبارت ورودی را خواهد یافت اما اگر Func باشد، ابتدا باید کل

اطلاعات را به صورت یک لیست IEnumerable دریافت و سپس سمت کلاینت، خروجی نهایی را فیلتر کند.
اگر برنامه را اجرا کنید نهایتا هر دو لیست یک و دو، بر اساس شرط عنوان شده عمل خواهند کرد و فیلتر خواهند شد. اما در حالت اول این فیلتر شدن سمت بانک اطلاعاتی است و در حالت دوم کل اطلاعات بارگذاری شده و سپس سمت کاربر فیلتر می‌شود (که کارآیی پایینی دارد).

نتیجه گیری

به امضای متد Where ایی که در حال استفاده است دقت کنید. همینطور در مورد Count ، Sum و یا موارد مشابه دیگری که predicate قبول می‌کنند.

نظرات خوانندگان

نویسنده: حسین مرادی نیا
تاریخ: ۱۳۹۲/۰۴/۲۶ ۲۰:۱

اوایل که از Entity Framework استفاده میکردم دچار همین مشکل شدم. در یک برنامه تمام متدهای دارای شرط لایه سرویسم رو با استفاده از Func پیاده سازی کرده بودم و بعد از مدتی متوجه شدم که برای دریافت یک رکورد از جدول هم برنامه خیلی کند عمل میکنه. کد زیر رو نگاه کنید:

```
public virtual TEntity Find(Func<TEntity, bool> predicate)
{
    return _tEntities.Where(predicate).FirstOrDefault();
}
```

در این حالت همه رکوردها از جدول مورد نظر واکشی میشه و بعد فقط یکی از آنها (اولین رکورد) در سمت کلاینت جدا و بازگشت داده میشه.

نویسنده: Saleh
تاریخ: ۱۳۹۲/۰۵/۱۰ ۱۰:۴۲

با تشکر از شما
جهت اطلاع دوستان:
کتاب Functional Programming In CS نوشته Oliver Sturm به طور کامل مبحث Genericها را پوشش داده و موضوعات Func و Expressionها را مفصل تشریح کرده است.

نویسنده: Saleh
تاریخ: ۱۳۹۲/۰۵/۱۰ ۱۲:۰

استفاده از Predicate<> چه تفاوتی با استفاده شما از Func دارد؟

حتی آقای نصیری هم به همین صورت استفاده کرده اند.

```
public virtual TEntity Find(Expression<Func<TEntity, bool>> predicate)
```

```
public virtual TEntity Find(Expression<Predicate<TEntity>> predicate)
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۵/۱۰ ۱۲:۳۸

استفاده نشده. فرق است بین یک پارامتر با *نام* predicate و یک delegate به نام Predicate . ضمنا Func هم یک Delegate است .