عنوان: **نوشتن پرس و جو در Entity Framework با استفاده از LINQ To Entity قسمت دوم**
نویسنده: محسن جمشیدی
تاریخ: ۱۳۹۲/۰۴/۱۳ ۱۰:۵
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, LINQ to Entity

در قسمت قبل با نحوه اجرای پرس و جو آشنا شدید و همچنین به بررسی متدهای Find و Single و First و تفاوت‌های آنها پرداختیم. در این قسمت با خصوصیت Local و متد Load آشنا خواهیم شد. همانطور که در قسمت قبل دیدید، مقادیر اولیه‌ای برای Database و جداولمان مشخص کردیم. برای جدول Customer این داده‌ها را داشتیم:

ID	Name	Family
یک مقدار Guid	Vahid	Nasiri
یک مقدار Guid	Mohsen	Akbari
یک مقدار Guid	Mohsen	Jamshidi

ID توسط Database تولید می‌شوند به همین دلیل از ذکر مقداری مشخص خودداری شده است.
به کد زیر دقت کنید:

```
private static void Query7()
{
    using (var context = new StoreDbContext())
    {
        // Add
        context.Customers.Add(new Customer { Name = "Ali", Family = "Jamshidi" });

        // change
        var customer1 = context.Customers.Single(c => c.Family == "Jamshidi");
        customer1.Name = "Mohammad";

        // Remove
        var customer2 = context.Customers.Single(c => c.Family == "Akbari");
        context.Customers.Remove(customer2);

        var customers = context.Customers.Where(c => c.Name != "Vahid");

        foreach (var cust in customers)
        {
            Console.WriteLine("Customer Name: {0}, Customer Family: {1}", cust.Name, cust.Family)
        }
    }
}
```

همانطور که مشاهده می‌کنید عمل اضافه، تغییر و حذف روی Customer انجام شده ولی هنوز هیچ تغییری در Database ذخیره نشده است. آخرین پرس و جو چه نتیجه ای را دربر خواهد داشت؟

بله، فقط تغییر یک موجودیت در نظر گرفته شده است ولی اضافه و حذف نه!
نتیجه مهمی که حاصل می‌شود این است که در پرس و جوهایی که روی Database اجرا می‌شوند سه مورد را باید در نظر داشت:

داده هایی که اخیرا به DbContext اضافه شده‌اند ولی هنوز در Database ذخیره نشده‌اند، درنظر گرفته نخواهند شد.

داده هایی که در DbContext حذف شده‌اند ولی در Database هستند، در نتیجه پرس و جو خواهند بود.

داده هایی که قبلا از database توسط پرس و جوی دیگری گرفته شده و تغییر کرده‌اند، آن تغییرات در نتیجه پرس و جو موثر

خواهند بود.

پس پرس و جوهای LINQ ابتدا روی database انجام می‌شوند و idهای بازگشت داده شده با idهای موجود در DbContext مطابقت داده می‌شوند یا در DbContext وجود دارند که در این صورت آن موجودیت بازگشت داده می‌شود یا وجود ندارند که در این صورت موجودیتی که از Database خوانده شده، بازگشت داده می‌شوند.
برای درک بیشتر کد زیر را در نظر بگیرید:

```
private static void Query7_1()
{
    using (var context = new StoreDbContext())
    {
        // Add
        context.Customers.Add(new Customer { Name = "Ali", Family = "Jamshidi" });

        // change
        var customer1 = context.Customers.Single(c => c.Family == "Jamshidi");
        customer1.Name = "Vahid";

        // Remove
        var customer2 = context.Customers.Single(c => c.Family == "Akbari");
        context.Customers.Remove(customer2);

        var customers = context.Customers.Where(c => c.Name != "Vahid");

        foreach (var cust in customers)
        {
            Console.WriteLine("Customer Name: {0}, Customer Family: {1}", cust.Name, cust.Family)
        }
    }
}
```

این کد همان کد قبلی است اما نام customer1 در DbContext (که Mohsen بوده در Database) به Vahid تغییر کرده و پرس و جو روی نام هایی زده شده است که Vahid نباشند خروجی به صورت زیر خواهد بود:

Customer Name: Vahid, Customer Family: Jamshidi

Customer Name: Mohsen, Customer Family: Akbari Vahid در خروجی آمده در صورتیکه در شرط صدق نمی‌کند چراکه پرس و جو روی Database زده شده، جاییکه نام این مشتری Mohsen بوده اما موجودیتی بازگشت داده شده که دارای همان Id هست اما در DbContext دستخوش تغییر شده است.

Local: همانطور که قبلا اشاره شد خصوصیتی از DbSet می‌باشد که شامل تمام داده هایی هست که:

اخیرا از database پرس و جو شده است (می‌تواند تغییر کرده یا نکرده باشد)

اخیرا به Context اضافه شده است (توسط متد Add)

دقت شود که Local شامل داده‌هایی که از database خوانده شده و از Context، حذف (Remove) شده‌اند، نمی‌باشد.
نوع این خصوصیت ObservableCollection می‌باشد که می‌توان از آن برای Binding در پروژه‌های ویندوزی استفاده کرد.
به کد زیر دقت کنید:

```
private static void Query8()
{
    using (var context = new StoreDbContext())
    {
        // Add
        context.Customers.Add(new Customer { Name = "Ali", Family = "Jamshidi" });

        // change
        var customer1 = context.Customers.Single(c => c.Family == "Jamshidi");
```

```
        customer1.Name = "Mohammad";

        // Remove
        var customer2 = context.Customers.Single(c => c.Family == "Akbari");
        context.Customers.Remove(customer2);

        var customers = context.Customers.Local;

        foreach (var cust in customers)
        {
            Console.WriteLine("Customer Name: {0}, Customer Family: {1}", cust.Name, cust.Family);
        }
    }
}
```

کد بالا شبیه به کد قبلی می‌باشد با این تفاوت که در انتها foreach روی Local زده شده است. خروجی به صورت زیر خواهد بود:

همانطور که ملاحظه می‌کنید Local شامل Ali Jamshidi که اخیرا اضافه شده (ولی در Database ذخیره نشده) و Mohammad Jamshidi که از Database خوانده شده و تغییر کرده، می‌باشد اما شامل Mohsen Akbari که از Database خوانده شده اما در Context حذف شده است، نمی‌باشد.
می‌توان روی Local نیز پرس و جوی اجرا کرد. در این صورت از پروایدر LINQ To Object استفاده خواهد شد و درنتیجه دست بازتر هست و تمام امکانات این پروایدر می‌توان استفاده کرد.

Load: یکی دیگر از مواردی که باعث اجرای پرس و جو می‌شود متد Load می‌باشد که یک Extension Method می‌باشد. این متد در حقیقت یک پیمایش روی پرس و جو انجام می‌دهد و باعث بارگذاری داده‌ها در Context می‌شود. مانند استفاده از ToList البته بدون ساختن List که سربار ایجاد می‌کند.

```
private static void Query9()
{
    using (var context = new StoreDbContext())
    {
        var customers = context.Customers.Where(c => c.Name == "Mohsen");
        customers.Load();

        foreach (var cust in context.Customers.Local)
        {
            Console.WriteLine("Customer Name: {0}, Customer Family: {1}", cust.Name, cust.Family);
        }
    }
    // Output:
    // Customer Name: Mohsen, Customer Family: Akbari
    // Customer Name: Mohsen, Customer Family: Jamshidi
}
```

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۴/۱۳ ۱۰:۴۸

این دوجمله رو

«- داده هایی که اخیرا به DbContext اضافه شده‌اند ولی هنوز در Database ذخیره نشده‌اند، درنظر گرفته نخواهند شد.
- داده هایی که در DbContext حذف شده‌اند ولی در Database هستند، در نتیجه پرس و جو خواهند بود.»

میشه خلاصه‌اش کرد به «تا زمانیکه SaveChanges فراخوانی نشه، از اطلاعات تغییر کرده نمیشه کوئری گرفت (کوئری‌ها همیشه روی دیتابیس انجام می‌شن)؛ اما خاصیت Local این تغییرات محلی رو داره یا اینکه در change tracker میشه موارد EntityState.Added | EntityState.Modified | EntityState.Unchanged رو هم کوئری گرفت».

نویسنده: محسن جمشیدی
تاریخ: ۱۳۹۲/۰۴/۱۳ ۱۴:۷

دقیقا!
جهت تاکید بیشتر روی "اجرا شدن پرس و جو در Database نه DbContext" متد Query7_1 به متن اضافه شد

نویسنده: مجید_فاضلی نسب
تاریخ: ۱۳۹۲/۰۸/۲۶ ۲۳:۱۴

DbContext دقیقا چیه ؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۸/۲۷ ۰:۵۱

قسمت‌های 11 و 12 سری EF رو مطالعه کنید.

نویسنده: پژمان
تاریخ: ۱۳۹۲/۱۱/۲۰ ۱۶:۵۳

با تشکر از مقاله آموزندتون، خواستم بدونم که مثلا در چه سناریویی بهتر است از متد .Localاستفاده کرد(یا به عبارتی این متد کی به درد کار ما میخورد)، با توجه به اینکه این متد اطلاعاتی که به اصطلاح In-Memory هستند را برای ما میآورد؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۱/۲۰ ۱۷:۷

بیشتر برای استفاده در WPF و برنامه‌های دسکتاپ است .

نویسنده: پژمان
تاریخ: ۱۳۹۲/۱۱/۲۰ ۱۷:۱۰

خیلی ممنون، در مورد Load هم تو نت گشتم چیزی دستگیرم نشد، اگه در این مورد هم مقاله ای باز سراغ دارید ممنون میشم لینکشو بدید ممنوون

نویسنده: محسن خان

تاریخ: ۱۳۹۲/۱۱/۲۰ ۱۷:۲۵

در همان مقاله‌ای که لینک دادم، اواسط آن به Load هم پرداخته. کاربرد عملی آن در پروژه طراحی فریم ورک WPF با EF در سایت هست.

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۱۱/۲۰ ۱۸:۱۲

یکی از کاربرداش در حذف اشیاء مرتبط