عنوان: یافتن Contextهای Dispose نشده در Entity framework
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۰۶ ۱۸:۴۰
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, Profiler

این دو متد را در نظر بگیرید:

```
private static void disposedContext()
        {
            using (var context = new MyContext())
            {
                Debug.WriteLine("Posts count: " + context.BlogPosts.Count());
            }
        }

        private static void nonDisposedContext()
        {
            var context = new MyContext();
            Debug.WriteLine("Posts count: " + context.BlogPosts.Count());
        }
```

در اولی با استفاده از using، شیء context به صورت خودکار dispose خواهد شد؛ اما در دومی از using استفاده نشده‌است.

سؤال: در یک برنامه‌ی بزرگ چطور می‌توان لیست Contextهای Dispose نشده را یافت؟

در EF 6 با تعریف یک IDbConnectionInterceptor سفارشی می‌توان به متدهای باز، بسته و dispose شدن یک Connection دسترسی یافت. اگر Context ایی dispose نشده باشد، اتصال آن نیز dispose نخواهد شد.

```
using System.Data;
using System.Data.Common;
using System.Data.Entity.Infrastructure.Interception;

namespace EFNonDisposedContext.Core
{
    public class DatabaseInterceptor : IDbConnectionInterceptor
    {
        public void Closed(DbConnection connection, DbConnectionInterceptionContext
interceptionContext)
        {
            Connections.AddOrUpdate(connection, ConnectionStatus.Closed);
        }

        public void Disposed(DbConnection connection, DbConnectionInterceptionContext
interceptionContext)
        {
            Connections.AddOrUpdate(connection, ConnectionStatus.Disposed);
        }

        public void Opened(DbConnection connection, DbConnectionInterceptionContext
interceptionContext)
        {
            Connections.AddOrUpdate(connection, ConnectionStatus.Opened);
        }

        // the rest of the IDbConnectionInterceptor methods ...

    }
}
```

همانطور که ملاحظه می‌کنید، با پیاده سازی IDbConnectionInterceptor، به سه متد Closed، Opened و Disposed یک DbConnection می‌توان دسترسی یافت.

مشکل مهم! در زمان فراخوانی متد Disposed، دقیقا کدام DbConnection باز شده، رها شده‌است؟
پاسخ به این سؤال را در مطلب « ایجاد خواص الحاقی » می‌توانید مطالعه کنید. با استفاده از یک ConditionalWeakTable به هر کدام از اشیاء DbConnection یک Id را انتساب خواهیم داد و پس از آن به سادگی می‌توان وضعیت این Id را ردگیری کرد.
برای این منظور، لیستی از ConnectionInfo را تشکیل خواهیم داد:

```
public enum ConnectionStatus
    {
        None,
        Opened,
        Closed,
        Disposed
    }

    public class ConnectionInfo
    {
        public string ConnectionId { set; get; }
        public string StackTrace { set; get; }
        public ConnectionStatus Status { set; get; }

        public override string ToString()
        {
            return string.Format("{0}:{1} [{2}]",ConnectionId, Status, StackTrace);
        }
    }
```

در اینجا ConnectionId را به کمک ConditionalWeakTable محاسبه می‌کنیم.
StackTrace توسط نکته‌ی مطلب « کدام سلسله متدها، متد جاری را فراخوانی کرده‌اند؟ » تهیه می‌شود.
Status نیز وضعیت جاری اتصال است که بر اساس متدهای فراخوانی شده در پیاده سازی IDbConnectionInterceptor مشخص می‌گردد.

در پایان کار برنامه فقط باید یک گزارش تهیه کنیم از لیست ConnectionInfoهایی که Status آن‌ها مساوی Disposed نیست. این موارد با توجه به مشخص بودن Stack trace هر کدام، دقیقا محل متدی را که در آن context مورد استفاده dispose نشده‌است، مشخص می‌کنند.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید

EFNonDisposedContext.zip

نظرات خوانندگان

نویسنده: علیرضا م
تاریخ: ۱۳۹۳/۰۷/۰۹ ۱۰:۳۹

سلام
اگر امکان دارد ارتباط این مطلب رو با Unit of work که در قسمت 12 آموزش Code First بیان نمودید ، توضیح دهید.
اگر درست فهمیده باشم بیان شد الگوی واحد کار برای جلوگیری وهله سازی در هر متود، به کار گرفته میشود در صورتی که هدف مقاله فعلی پیدا کردن وهله‌های dispose نشده درون متدهای برنامه است.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۰۹ ۱۱:۲۰

- همه شاید از الگوی واحد کار استفاده نکنند.
- کسانی هم که از الگوی واحد کار استفاده می‌کنند شاید بد نباشد بررسی کنند که در پایان کار Context و Connection زنده‌ای هنوز وجود دارد یا خیر.
- همه جا امکان استفاده از الگوی واحد کاری که از یک Context در طول یک درخواست استفاده می‌کند، نیست. خصوصا در مکان‌هایی که وهله سازی آن‌ها را نمی‌توان تحت کنترل خودکار IoC Containerها در آورد؛ مثلا در یک Role Provider که راسا توسط ASP.NET وهله سازی می‌شود و یا یک وظیفه‌ی فعال پس زمینه.
- گزارشی که در انتهای کار روش فوق تهیه می‌شود، مستقل است از نحوه‌ی بکارگیری و مدیریت وهله‌های Context. همچنین مستقل است از Code-first یا Db first و غیره. قابلیت interceptor آن، بحثی است عمومی.
- «هدف مقاله فعلی پیدا کردن وهله‌های dispose نشده درون متدهای برنامه است»
نهایتا از هر روشی که استفاده کنید، در متدی مشخص، وهله سازی می‌شود و شاید در جایی Dispose و یا خیر. در اینجا می‌شود از این نوع مکان‌ها گزارش گرفت.