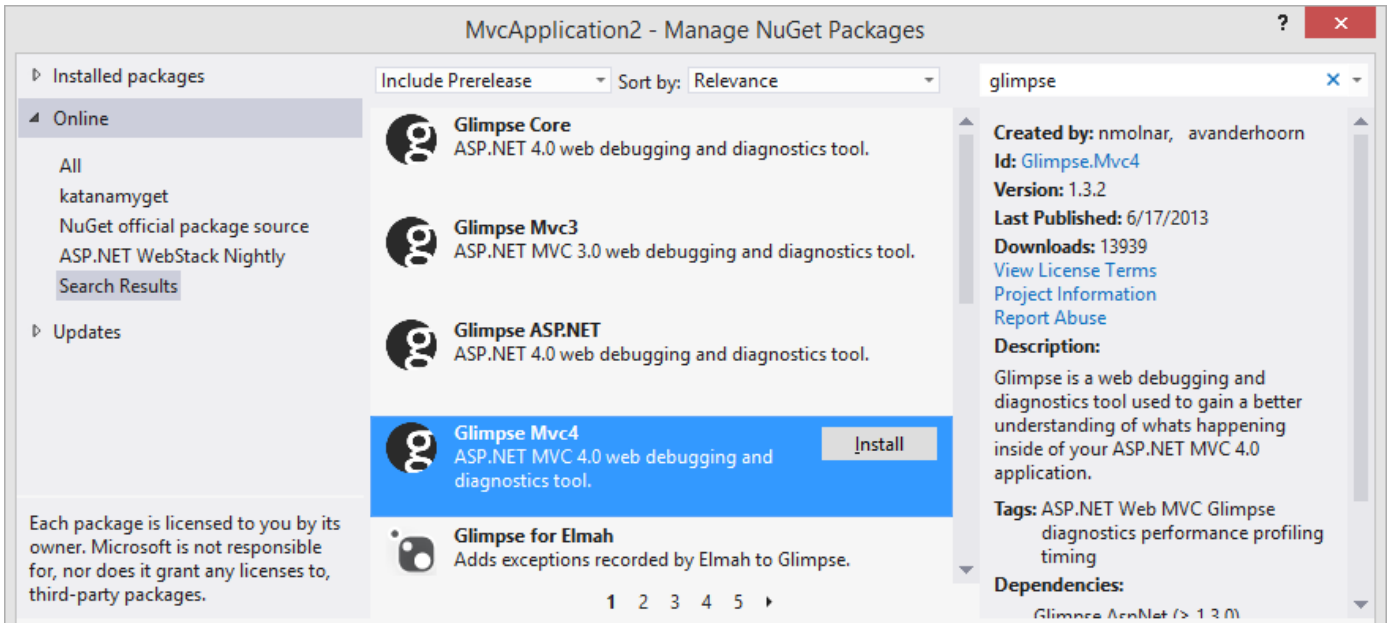
عنوان: **آشنایی و بررسی ابزار Glimpse**
نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۲/۰۴/۳۱ ۰:۱۵
آدرس: www.dotnettips.info
گروه‌ها: Debugging, Visual Studio, Glimpse, Profiler

در مطلب MiniProfiler ابزار مانیتور کارآیی وب سایت‌ها را بررسی کردیم. اما ابزار Glimpse هم جزو ابزارهای حرفه‌ای است که در مطلبی آقای هانسلمن در سایت خود به آن پرداخته بودند. اما دیدم جای یک مطلب فارسی در این رابطه خالی است.

Glimpse چیست؟

glimpse یک ابزار حرفه‌ای برای نمایش زمان اجرای کدها، پیکربندی سرور، درخواست‌های وب، اشکال زدایی و بررسی کارآیی وب سایت‌های MVC و Web Forms می‌باشد. البته بدون آنکه در کدهای پروژه شما تغییری ایجاد نماید.

ابتدا در پنجره Nuget عبارت glimpse را جستجو و آن را نصب نمایید:



کتابخانه‌های زیادی برای این ابزار آماده شده‌اند:
کتابخانه Glimpse Core
که هسته اصلی ابزار است، حتما باید نصب شود.
کتابخانه Glimpse ASP.NET
برای بررسی وب سایت‌های نوشته شده با ASP.NET Web Forms استفاده می‌شود. البته بری Mvc هم لازم است.
کتابخانه Glimpse Mvc2, Glimpse Mvc3، Glimpse Mvc4
برای بررسی وب سایت‌های نوشته شده با ASP.NET Mvc
کتابخانه Glimpse Ado
برای بررسی و نمایش زمان کوئری بر روی پایگاه داده
کتاخانه Glimpse EF4.3، Glimpse EF5، Glimpse EF6
برای زمانیکه از نگارش‌های مختلف Entity Framework استفاده می‌نماییم

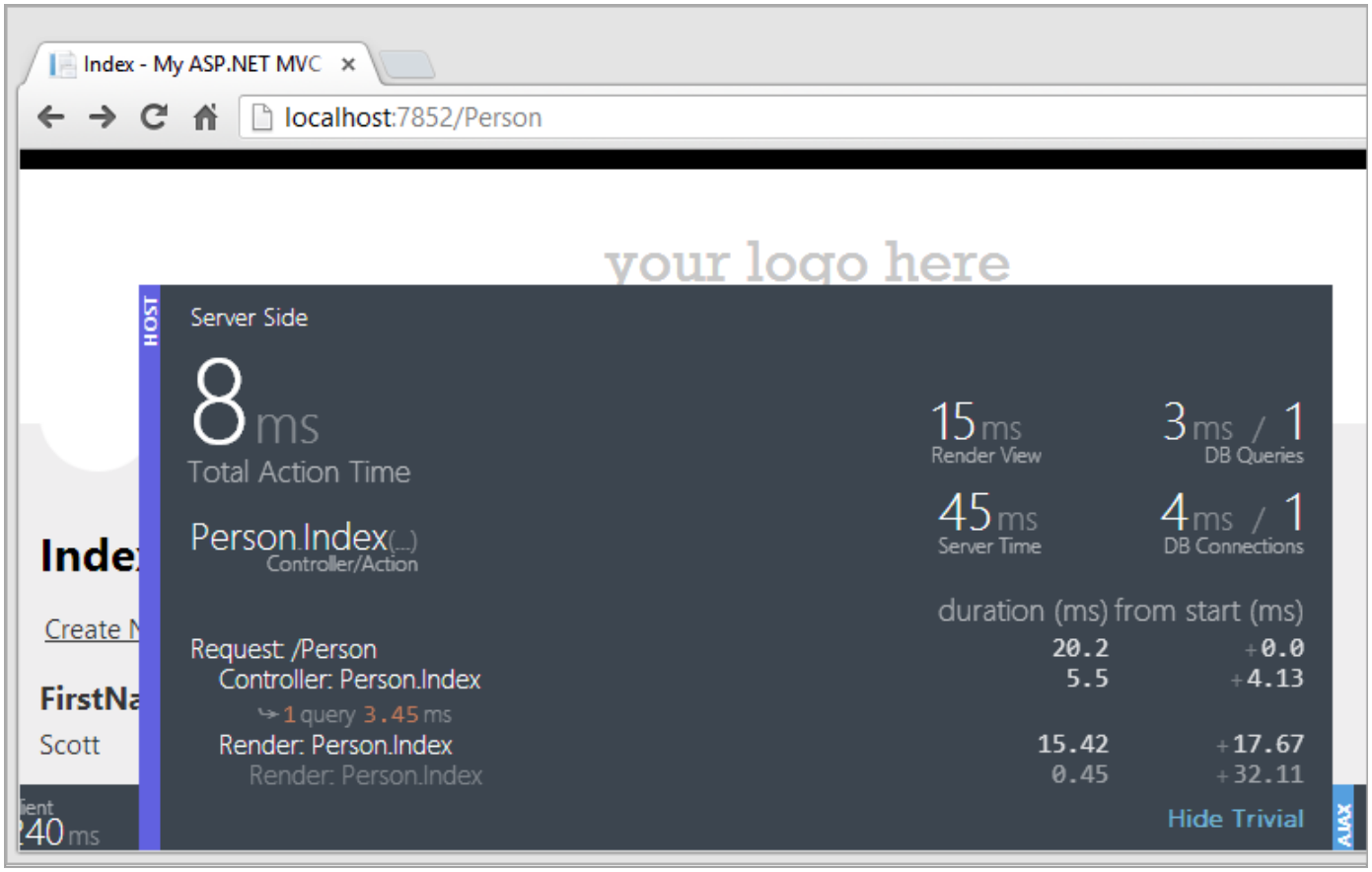
پس از نصب کتابخانه‌های مورد نیاز، پروژه را rebuild و سپس اجرا نمایید. برای فعال کردن glimpse آدرس http://{your-site}/Glimpse.axd را اجرا کنید تا صفحه تنظیمات آن فعال شوند و سپس بر روی گزینه Turn Glimpse on، کلیک کنید. همچنین با گزینه Turn Glimpse off می‌توانید آن را غیر فعال نمایید.

علاوه بر این، تنظیمات استاندارد این ابزار قابل تغییر است.
به صفحه اصلی سایت برگشته و صفحه را بروز رسانی کنید. ابزار glimpse در پایین مرورگر نمایش داده می‌شود.

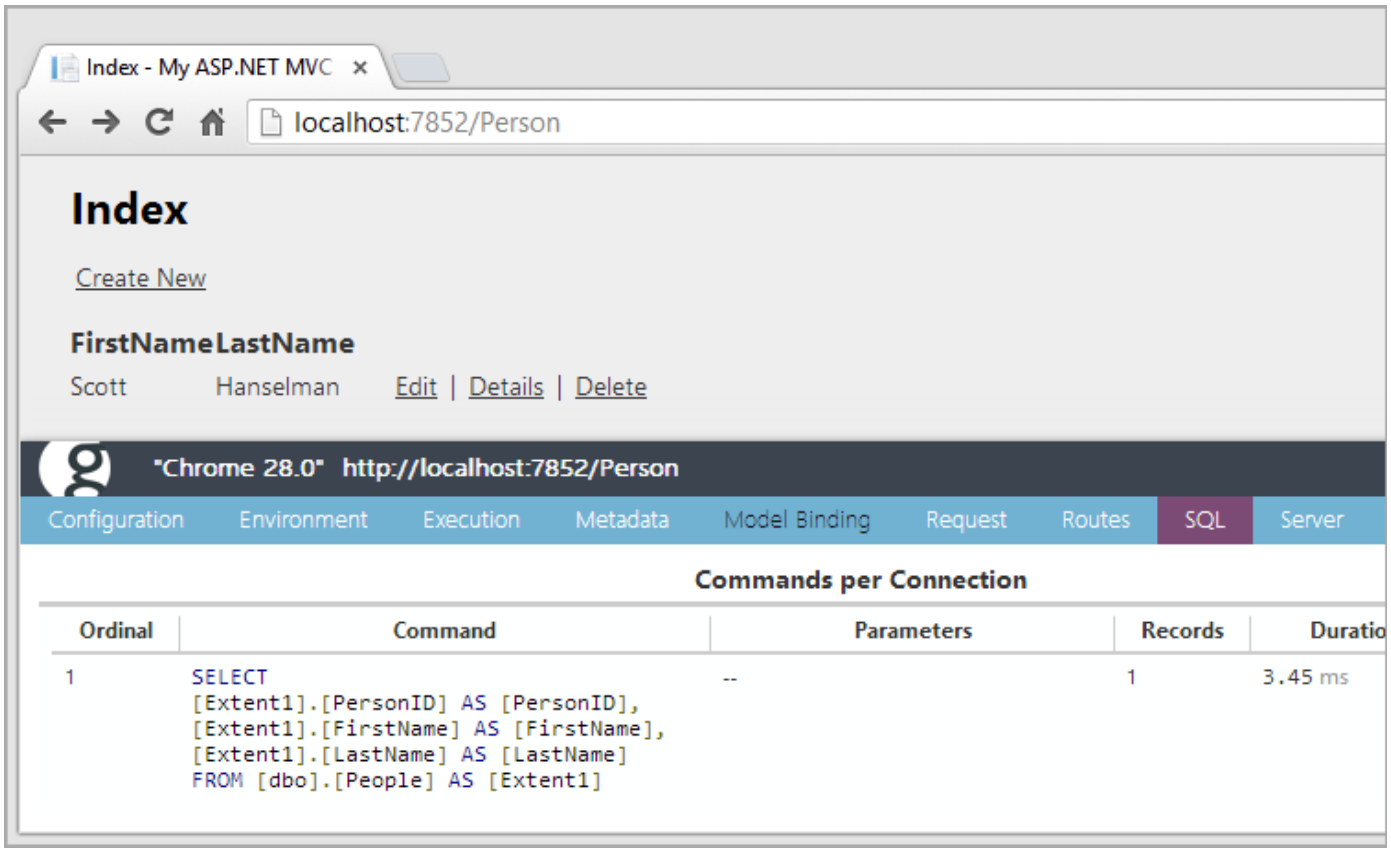
این ابزار شامل سه قسمت است:

HTTP
اطلاعات Request و زمان پاسخ و اطلاعات سرور نمایش داده می‌شود
HOST
اطلاعات صفحه اجرا شده، زمان پاسخ و تعداد کوئری‌های اجرا شده و زمان آن نمایش داده می‌شوند
AJAX
اطلاعات درخواست‌های اجکسی این صفحه و تعداد آن نمایش داده می‌شوند

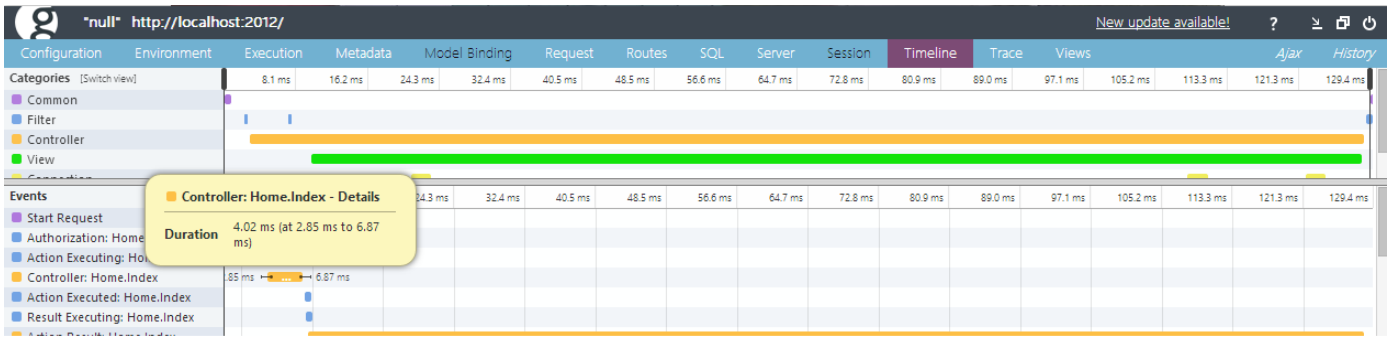
بر روی هر یک از این قسمت‌ها با حرکت ماوس، جزئیات آن قسمت نمایش داده می‌شود.



اگر بر روی آیکون g ابزار کلیک کنید، همچون developer tools مرورگرها باز شده و دارای زبانه‌های متعددی می‌باشد. مثلا اگر پلاگین ado و ef5 نصب باشند، در زبانه SQL می‌توانید کوئری‌های اجرا شده و زمان مصرف شده آن‌ها را مشاهده نمایید



زبانه دیگر Timeline است که زمان انقیاد اشیاء و رویدادها را بصورت گرافیکی نمایش می‌دهد.



در مطلب بعدی به جزئیات بیشتری از این ابزار می‌پردازم.

نظرات خوانندگان

نویسنده: ali
تاریخ: ۱۳۹۲/۰۵/۰۷ ۱۲:۵۶

ممنون بابت این مطلب مفید.

هنگام آپلود سایت اگر نخوایم این ابزار کار کند باید قبلش uninstall کنیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۵/۰۸ ۱۰:۵۰

نه الزاما. میشه در وب کانفیگ غیرفعالش کرد .

نویسنده: رضا گرمارودی
تاریخ: ۱۳۹۲/۱۲/۰۳ ۱۲:۹

glimpse و Miniprofiler هر دو با Ef6 مشکل دارند . گرچه در سایت‌های برنامه‌های فوق عنوان شده که Ef6 را پوشش میدهند اما هر کدوم به نحوی باگی دارند. از اونجایی که در Ef6 با Rdbms اسکیوال CE کار می‌کنم و همانند Sql server پروفایلری نداره که دستورات ارسالی را بشه دید شما در Ef6 به غیر از دو پروفایل ذکر شده از چه پروفایلری استفاده می‌کنید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۰۳ ۱۳:۸

روش « نمایش خروجی SQL کدهای Entity framework 6 در کنسول دیباگ ویژوال استودیو » نیاز به ابزار اضافی ندارد.

نویسنده: هیمن صادقی
تاریخ: ۱۳۹۳/۰۴/۲۸ ۱۴:۵۵

سپاس از مطب شما
زمانی که با entity framework 6 استفاده می‌کنیم
خطا زیر رو میده راه حل برایش مشکل وجود دارد

```
No Entity Framework provider found for the ADO.NET provider with invariant name 'System.Data.Odbc'. Make sure the provider is registered in the 'entityFramework' section of the application config file. See http://go.microsoft.com/fwlink/?LinkId=260882 for more information.
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۲۸ ۱۵:۴

برای EF 6 بسته جداگانه دارد:

```
PM> Install-Package Glimpse.EF6
```

نویسنده: هیمن صادقی
تاریخ: ۱۳۹۳/۰۴/۲۸ ۱۵:۱۲

با نصب این بسته بازم خطا رخ میده
در سایت‌های مختلف جستجو کردم پاسخ مناسب پیدا نکردم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۲۸ ۱۷:۱۴

- محل گزارش خطاهای این پروژه
+ در EF 6 فایل کانفیگ برنامه حتما باید ویرایش شود و تعریف پروایدر را داشته باشد (^ و ^)؛ مثلا:

```
<entityFramework>
    <providers>
      <provider invariantName="System.Data.Odbc"
type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
      <provider invariantName="System.Data.SqlClient"
type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
    </providers>
  </entityFramework>
```

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۳/۱۰ ۲۰:۴۰

با تشکر از مطالبی که ارئه کردید
شما زحمت دو مطلب در این زمینه رو کشیدید
خواستم بدونم که شما تجربه عملی کار با این دو ابزار را دارید
به نظر شما کدام ابزار برای انتخاب بهتر هست؟
ابزاری که در این مقاله معرفی کردید یا miniprofiler

نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۴/۰۳/۱۱ ۱۳:۴۳

ابزار Glimpse خیلی حرفه ای تر است حتی امکان استفاده MiniProfiler بصورت پلاگین در آن نیز وجود دارد

```
Install-Package Glimpse.MiniProfiler
```

عنوان: **تنظیمات امنیتی Glimpse**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۲/۰۸/۱۴ ۰:۴۵
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, MVC, Glimpse, Profiler

در مورد glimpse پیشتر مطالبی در سایت منتشر شده است :

آشنایی و بررسی ابزار Glimpse

بعد از آپلود سایت ما می‌توانیم دسترسی به تنظیمات خاص glimpse را تنها به کاربران عضو محدود کنیم:

```
<location path="Glimpse.axd" >
    <system.web>
        <authorization>
            <allow users="Administrator" />
            <deny users="*" />
        </authorization>
    </system.web>
</location>
```

یا می‌توانیم آنرا غیرفعال کنیم :

```
<glimpse defaultRuntimePolicy="Off" xdt:Transform="SetAttributes">
</glimpse>
```

همچنین می‌توانیم با پیاده سازی اینترفیس IRuntimePolicy سیاست‌های مختلف نمایش تب‌های glimpse را تعیین کنیم :

```
using Glimpse.AspNet.Extensions;
using Glimpse.Core.Extensibility;

namespace Test
{
    public class GlimpseSecurityPolicy:IRuntimePolicy
    {
        public RuntimePolicy Execute(IRuntimePolicyContext policyContext)
        {
            // You can perform a check like the one below to control Glimpse's permissions within your
application.
// More information about RuntimePolicies can be found at http://getglimpse.com/Help/Custom-Runtime-
Policy
var httpContext = policyContext.GetHttpContext();
            if (!httpContext.User.IsInRole("Administrator "))
            {
                return RuntimePolicy.Off;
            }

            return RuntimePolicy.On;
        }

        public RuntimeEvent ExecuteOn
        {
            get { return RuntimeEvent.EndRequest; }
        }
    }
}
```

زمانیکه glimpse را از طریق Nuget نصب می‌کنید کلاس فوق به صورت اتوماتیک به پروژه اضافه می‌شود با این تفاوت که به صورت کامنت شده است تنها کاری شما باید انجام بدید کدهای فوق را از حالت کامنت خارج کنید و Role مربوطه را جایگزین کنید.

نکته : کلاس فوق نیاز به رجیستر شدن ندارد و تشخیص آن توسط Glimpse به صورت خودکار انجام می‌شود.

عنوان: **کدام سلسله متدها، متد جاری را فراخوانی کرده‌اند؟**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۰۶ ۹:۴۰
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, Reflection, Profiler, StackTrace

یکی از نیازهای نوشتن یک برنامه‌ی پروفایلر، نمایش اطلاعات متدهایی است که سبب لاگ شدن اطلاعاتی شده‌اند. برای مثال در طراحی interceptorهای EF 6 به یک چنین متدهایی می‌رسیم:

```
public void ScalarExecuted(DbCommand command,
                                    DbCommandInterceptionContext<object> interceptionContext)
        {
        }
```

سؤال: در زمان اجرای ScalarExecuted دقیقا در کجا قرار داریم؟ چه متدی در برنامه، در کدام کلاس، سبب رسیدن به این نقطه شده‌است؟
تمام این اطلاعات را در زمان اجرا توسط کلاس StackTrace می‌توان بدست آورد:

```
public static string GetCallingMethodInfo()
        {
            var stackTrace = new StackTrace(true);
            var frameCount = stackTrace.FrameCount;

            var info = new StringBuilder();
            var prefix = "-- ";
            for (var i = frameCount - 1; i >= 0; i--)
            {
                var frame = stackTrace.GetFrame(i);
                var methodInfo = getStackFrameInfo(frame);
                if (string.IsNullOrWhiteSpace(methodInfo))
                    continue;

                info.AppendLine(prefix + methodInfo);
                prefix = "-" + prefix;
            }

            return info.ToString();
        }
```

ایجاد یک نمونه جدید از کلاس StackTrace با پارامتر true به این معنا است که می‌خواهیم اطلاعات فایل‌های متناظر را نیز در صورت وجود دریافت کنیم.
خاصیت stackTrace.FrameCount مشخص می‌کند که در زمان فراخوانی متد GetCallingMethodInfo که اکنون برای مثال درون متد ScalarExecuted قرار گرفته‌است، از چند سطح بالاتر این فراخوانی صورت گرفته‌است. سپس با استفاده از متد stackTrace.GetFrame می‌توان به اطلاعات هر سطح دسترسی یافت.
در هر StackFrame دریافتی، با فراخوانی stackFrame.GetMethod می‌توان نام متد فراخوان را بدست آورد. متد stackFrame.GetFileLineNumber دقیقا شماره سطری را که فراخوانی از آن صورت گرفته، بازگشت می‌دهد و stackFrame.GetFileName نیز نام فایل مرتبط را مشخص می‌کند.

یک نکته:
شرط عمل کردن متدهای stackFrame.GetFileName و stackFrame.GetFileLineNumber در زمان اجرا، وجود فایل PDB اسمبلی در حال بررسی است. بدون آن اطلاعات محل قرارگیری فایل سورس مرتبط و شماره سطر فراخوان، قابل دریافت نخواهند بود.

اکنون بر اساس این اطلاعات، متد getStackFrameInfo چنین پیاده سازی را خواهد داشت:

```
private static string getStackFrameInfo(StackFrame stackFrame)
        {
            if (stackFrame == null)
                return string.Empty;

            var method = stackFrame.GetMethod();
```

```
            if (method == null)
                return string.Empty;

            if (isFromCurrentAsm(method) || isMicrosoftType(method))
            {
                return string.Empty;
            }

            var methodSignature = method.ToString();
            var lineNumber = stackFrame.GetFileLineNumber();
            var filePath = stackFrame.GetFileName();

            var fileLine = string.Empty;
            if (!string.IsNullOrEmpty(filePath))
            {
                var fileName = Path.GetFileName(filePath);
                fileLine = string.Format("[File={0}, Line={1}]", fileName, lineNumber);
            }

            var methodSignatureFull = string.Format("{0} {1}", methodSignature, fileLine);
            return methodSignatureFull;
        }
```

و خروجی آن برای مثال چنین شکلی را خواهد داشت:

```
Void Main(System.String[]) [File=Program.cs, Line=28]
```

که وجود file و line آن تنها به دلیل وجود فایل PDB اسمبلی مورد بررسی است.

در اینجا خروجی نهایی متد GetCallingMethodInfo به شکل زیر است که در آن چند سطح فراخوانی را می‌توان مشاهده کرد:

```
-- Void Main(System.String[]) [File=Program.cs, Line=28]
--- Void disposedContext() [File=Program.cs, Line=76]
---- Void Opened(System.Data.Common.DbConnection,
System.Data.Entity.Infrastructure.Interception.DbConnectionInterceptionContext)
[File=DatabaseInterceptor.cs,Line=157]
```

جهت تعدیل خروجی متد GetCallingMethodInfo، عموما نیاز است مثلا از کلاس یا اسمبلی جاری صرفنظر کرد یا اسمبلی‌های مایکروسافت نیز در این بین شاید اهمیتی نداشته باشند و بیشتر هدف بررسی سورس‌های موجود است تا فراخوانی‌های داخلی یک اسمبلی ثالث:

```
private static bool isFromCurrentAsm(MethodBase method)
        {
            return method.ReflectedType == typeof(CallingMethod);
        }

        private static bool isMicrosoftType(MethodBase method)
        {
            if (method.ReflectedType == null)
                return false;

            return method.ReflectedType.FullName.StartsWith("System.") ||
                   method.ReflectedType.FullName.StartsWith("Microsoft.");
        }
```

کد کامل CallingMethod.cs را از اینجا می‌توانید دریافت کنید:

CallingMethod.cs

نظرات خوانندگان

نویسنده: علیرضا
تاریخ: ۱۳۹۳/۰۷/۱۰ ۲۳:۳۸

چه موقعی GetMethod میتواند Null برگرداند؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۱۱ ۰:۳۷

زمانیکه کامپایلر مباحث inlining متدها را جهت بهینه سازی اعمال کند.

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۴/۱۰ ۸:۵۳

بیان این نکته خالی از لطف نیست که در دات نت ۴.۵ به بعد یک سری attribute هم برای راحتی کار ارائه شده است. یک نمونه آن callermemberinfo است که در این مقاله یکی از استفاده‌های کاربردی آن را می‌بینید

نویسنده: محسن خان
تاریخ: ۱۳۹۴/۰۴/۱۰ ۹:۴۲

callermemberinfo فقط یک سطح را بازگشت می‌ده و همچنین امضای متدها را هم باید تغییر داد. زمانیکه قرار هست پروفایلری را تهیه کنید، این پروفایلر نباید سبب تغییر کدهای اصلی برنامه شود.

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 Opened، Closed 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.