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.