InsightProfiler: Überblick über Abläufe

Tiefe Einblicke

Der Name ist Programm: InsightProfiler verschafft Überblick über den internen Ablauf und die Laufzeiten von .NET-Software – auch bei Applikationen mit mehreren Threads. Auf der Heft-CD finden Sie eine voll funktionfähige Version.

Und dann kommt der Fehler: Er tritt natürlich in einer fremden Komponente auf, im noch schwer durchschaubaren Klassengewirr des Projektes, zu dem Sie eben erst dazugestoßen sind. Oder schlicht in genau dem Programmcode, den nur der Kollege kennt, der seit gestern für drei Wochen im Urlaub ist. Nun heißt es: trotz Zeitdruck einen kühlen Kopf bewahren und sich einen Überblick über die Abläufe verschaffen.

Welche Klassen sind beteiligt und welche Aufgabe erfüllen Sie? Wie spielen die beteiligten Objekte zusammen? Welche Methode ruft welche andere Methode auf? Und letztlich, was geht bei alldem schief?

In so einem Fall kann der InsightProfiler helfen, schneller zu verstehen, was in einem fremden Programmabschnitt wirklich vor sich geht. Exklusiv für die Leser der dotnetpro steht auf der Heft-CD eine kostenlose Version des InsightProfilers in der „dotnetpro Edition“ bereit und wartet darauf in Ihre Werkzeugkiste zu wandern.

Ein Feature in wildfremden Programmcode einbauen

Stellen Sie sich folgende Szene vor. Ihre Firma entwickelt den *ILSpy*. Dieses Tool decompiliert .NET-Assemblies zurück in C#-Quellcode. Außerhalb dieser Szene ist *ILSpy* Open Source. Innerhalb der Szene ist der Hauptentwickler gerade in den Urlaub entfleucht und lässt Sie mit einem Chef zurück, der heute etwas durch den Wind ist. Auftritt Chef.

„Wenn wir nicht bis Mittag einen rechtlichen Warnhinweis in jede dekompilierte Methode einbauen, verklagt uns die Konkurrenz und wir können zusperren!“

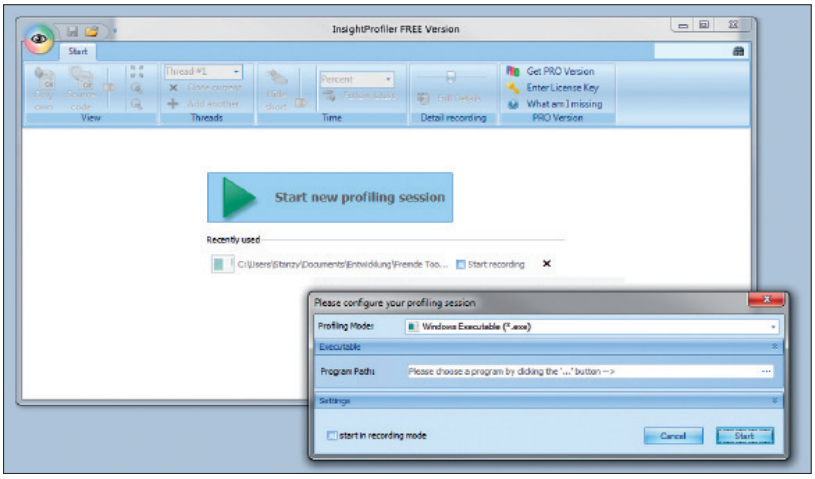
In dieser Situation haben Sie zwei Möglichkeiten: Entweder Sie starten den Browser und suchen in einer der Jobbörsen nach einer neuen Beschäftigung oder Sie stellen sich der Aufgabe und beginnen mit der Suche nach der Antwort auf die Frage, an welcher Stelle diese Funktionalität in *ILSpy* wohl einzubauen wäre.

Die erste Option vergessen Sie ganz schnell wieder, denn Sie haben Tools, die Ihnen bei der Suche helfen. Sie beruhigen also erst mal Ihren Chef: „Kein Problem, gibt‘s sonst noch etwas, was ich bis Mittag erledigen soll?“ und doppelklicken auf die *InsightProfiler.exe*, die sie direkt von der CD der dotnetpro aus starten.

Schritt 1: Profiler starten

Auf dem Startbildschirm **(Abbildung 1)** der Anwendung klicken Sie einen grünen Play-Pfeil mit der Beschriftung „Start new profiling session“ und wählen im folgenden Dialog *ILSpy.exe* aus. Den Haken bei *start in recording mode* können Sie getrost weglassen, Sie wollen ja nur das aufzeichnen was wirklich für Ihren Fall interessant ist. Anschließend klicken Sie auf den Start-Button und beobachten, wie sich das Fenster des *ILSpy* öffnet.

Auf der linken Seite des Fensters finden Sie Assemblies, Klassen und Methoden, die bei einem Klick auf der rechten Seite ihren dekompi-



[Abb. 1] Im Startbildschirm wählen Sie die zu analysierende Exe-Datei aus.

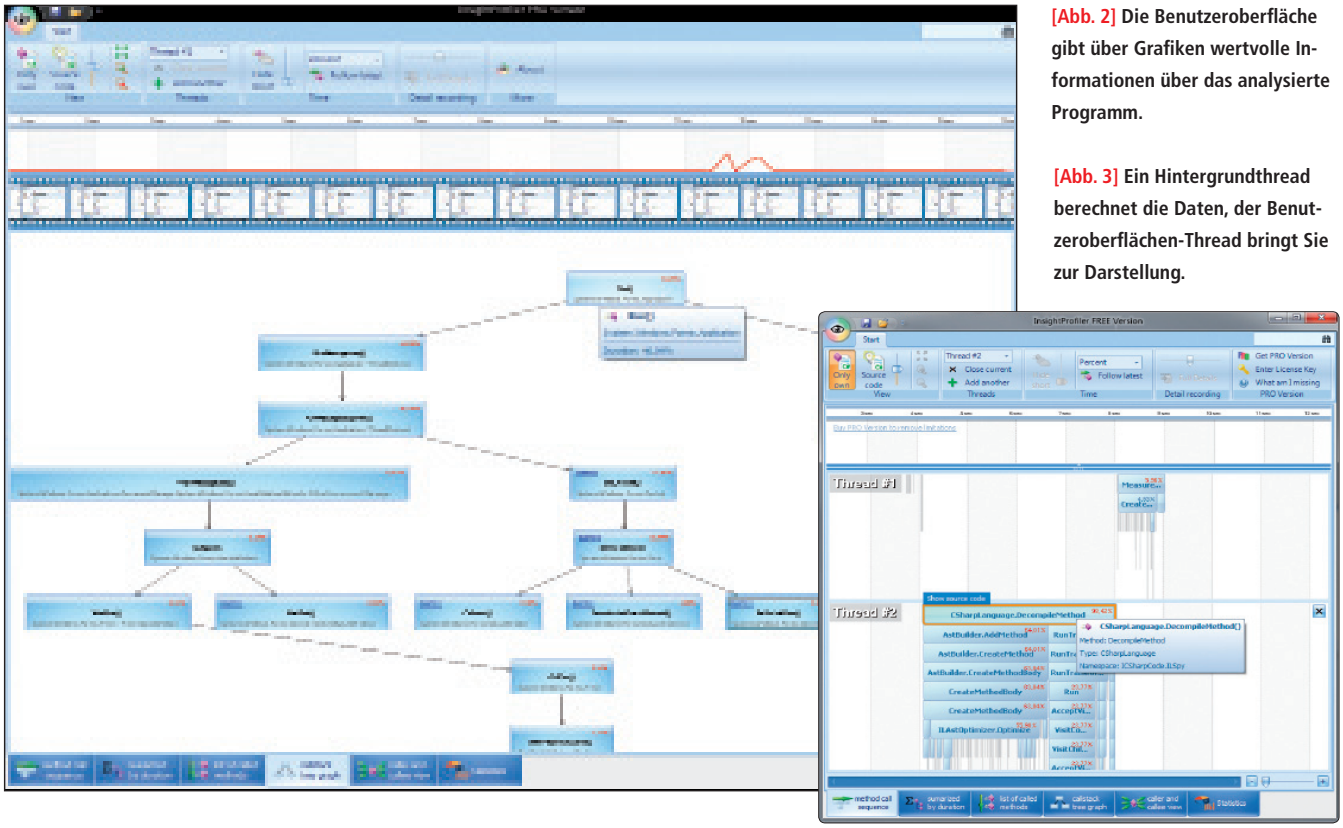
Auf einen Blick

Florian Standhartinger hielt schon mit sechs Jahren seinen ersten Commodore C64 in den Händen. Er entdeckte bereits damals mit BASIC seine Leidenschaft fürs Programmieren. Zur zeit arbeitet er bei msg systems ag am Aufbau des C#-Bereiches mit. Sie erreichen ihn über florian.st@ndhartinger.de oder www.productivity-boost.com.

Inhalt

- Überblick über fremden Code erlangen
- Auch Programme mit mehreren Threads lassen sich durchforschen
- Grafische Anzeige mit Zoom in und Zoom out

dnpCode
A1207Insight



[Abb. 2] Die Benutzeroberfläche gibt über Grafiken wertvolle Informationen über das analysierte Programm.

[Abb. 3] Ein Hintergrundthread berechnet die Daten, der Benutzeroberflächen-Thread bringt Sie zur Darstellung.

lierten Inhalt preisgeben. Sie öffnen also das Pluszeichen einer Klasse und halten einen Moment inne, bevor Sie das Dekompilat einer Methode anfordern.

Schritt 2: Einen Teil des Programmablaufes aufzeichnen

Ihr Ziel ist es, der Textausgabe der zurückübersetzten Methode einen Warnhinweis hinzuzufügen. Um zu verstehen in welchen Klassen diese Textausgabe überhaupt erzeugt wird, lassen Sie den InsightProfiler aufzeichnen, was im *ILSpy* stattfindet, wenn dieser eine Methode anzeigt um sich dann anzusehen, wo Sie den Eingriff vornehmen können. Also klicken Sie auf den großen roten Knopf mit der Aufschrift *Start recording method calls* um dem Profiler mitzuteilen, dass es ab hier interessant wird. Anschließend wählen Sie in *ILSpy* eine beliebige Methode aus und warten, bis das Ergebnis auf der rechten Seite angezeigt wird.

Ist der Methodenbody sichtbar, klicken Sie auf den blauen Stop-Button im Profiler. Das schließt die Aufzeichnung ab.

Wie Sie während der Aufzeichnung gesehen haben, hat der Profiler mitgeschnitten was im Programm abgelaufen ist und zeigt diese Informationen in Form eines Diagrammes an. Mit Mausrad und Drag-and-Drop-Gesten können Sie nun das Aufgezeichnete inspizieren (Abbildung 2).

Schritt 3: Multithreading entwirren

Eine der verwirrendsten Technologien, mit denen sich Entwickler in den letzten Jahren herumschlagen mussten, dürfte das Multithreading sein. Aufgrund der größer werdende Zahl von Prozessorkernen pro PC, wird diese Problematik nicht so bald verschwinden.

Mit herkömmlichem Debugging einen Ablauf zu verstehen, in dem verschiedene Threads zusammenspielen, ist eine ganz besondere Herausforderung.

Und auch das Beispiel hält diese Überraschung bereit. Nachdem Ihnen die angezeigten Aufrufe des Benutzeroberflächen-Threads zu wenig nach dem Dekompilieren einer Methode aussehen, beschließen Sie, mal in den anderen Threads des Prozesses zu stöbern. Sie klicken hierzu einfach ein paar Mal auf den Button *Add another* im Ribbon-Menü, bis Sie auf einen Thread stoßen, dessen Aktivität genau dort aufhört, wo der Benutzeroberflächen-Thread seine Arbeit beginnt. Sehr verdächtig – und tatsächlich finden Sie hier was Sie suchen (Abbildung 3).

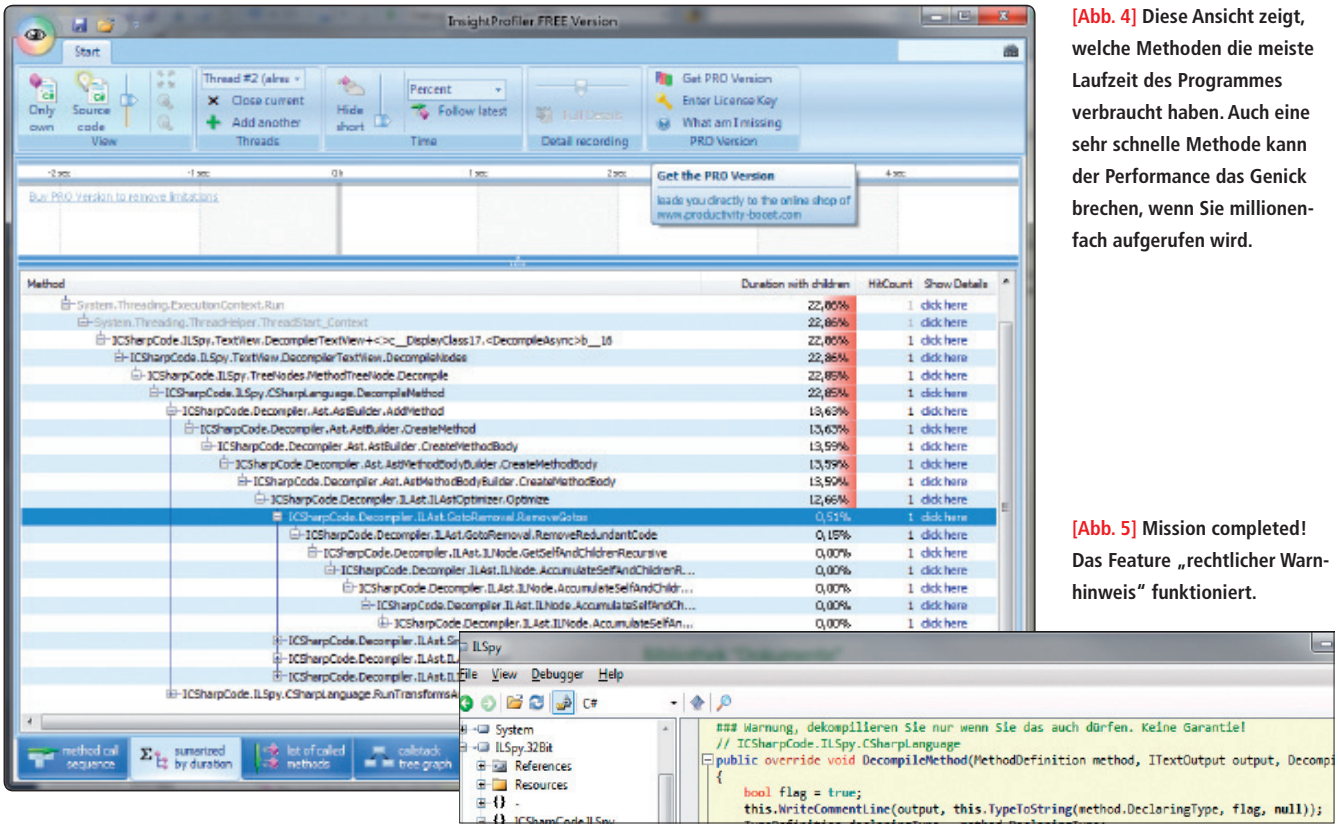
Eine Methode *DecompileMethod* einer Klasse *CSharpLanguage* nimmt hier 98 Prozent der vergangenen Zeit ein. Das ist die Stelle wo Sie eingreifen werden, um Ihren Hinweistext unterzubringen.

Um diese Ansicht des InsightProfilers nutzen zu können, ist es wichtig, genau zu verstehen, was sie anzeigt. Die horizontale Achse ist die Zeitachse. Daran, dass Funktion *DecompileMethod* so breit ist, sehen Sie, dass Sie viel Zeit gebraucht hat. Die vertikale Achse stellt dar, welche Methoden sich jeweils gegenseitig aufgerufen haben. Im Screenshot unseres Beispieles etwa sieht man, dass die Methode *AddMethod* einer Klasse *AstBuilder* (mit etwa 64 Prozent Dauer) und eine Methode *RunTransformationsAndGenerateCode* von *DecompileMethod* aufgerufen wurden.

Schritt 4: Cruisen durch den Programmablauf

Jetzt wo das Ziel so nah ist, bleibt sogar noch etwas Zeit, um den Spieltrieb zu befriedigen. Zoomen Sie mit dem Mausrad hinein und hinaus, vergleichen Sie Abhängigkeiten zwischen den verschiedenen Threads oder klicken Sie sich einfach durch die Methoden, für die der Profiler den Quellcode anzeigt, oder per integriertem ILSpy dekompiliert.

Selbst wer seit Jahren an einem Programm entwickelt, dürfte hier ob der intutiven Darstellung das eine oder andere Aha-Erlebnis haben, was die Zeitverhältnisse oder gar die ablaufenden Methodenaufrufe innerhalb eines Programmes



[Abb. 4] Diese Ansicht zeigt, welche Methoden die meiste Laufzeit des Programmes verbraucht haben. Auch eine sehr schnelle Methode kann der Performance das Genick brechen, wenn Sie millionenfach aufgerufen wird.

[Abb. 5] Mission completed! Das Feature „rechtlicher Warnhinweis" funktioniert.

angeht. Aber Moment, wieso heißt das Ding eigentlich Profiler?

Schritt 5: Bonuspunkte durch Performance Profiling

Ganz davon abgesehen, dass InsightProfiler eine völlig neue Art bietet, Einblicke in ein abgelaufenes Programm zu erlangen, ist er nebenbei ein vollständiger Performance Profiler. Und dazu noch einer der ganz wenigen, die es ermöglichen, einen markierten Zeitabschnitt aus der Gesamtmenge der abgelaufenen Methoden herauszurechnen um nur das anzuzeigen, was den Anwender tatsächlich interessiert (Abbildung 4).

Markieren Sie also in der Zeitleiste direkt unterhalb des Menüs mit der Maus den Zeitbereich, in dem Sie die Dekompilierung der Methode aufgezeichnet haben. Wenn Sie Start und Ende des Recordings gut gewählt haben, kann dieser Schritt auch entfallen. Anschließend wechseln Sie mit dem am unteren Rand befindlichen Karteireiter *summarized by duration* in die klassische Profileransicht, die man in der einen oder anderen Art wohl in den meisten Performance Profilern findet.

Hier wird die Dauer, die das Programm in dem immer gleichen Callstack verbraucht hat, aufsummiert angezeigt und vom aufwändigsten zum schnellsten von oben nach unten sortiert dargestellt. Das, was im Programm (beziehungsweise im gerade gewählten Thread) am längsten gedauert hat, sehen Sie also ganz oben.

Als Dreingabe zur unverhofft schnellen Einarbeitung in den ILSpy wollen Sie Ihrem Chef noch eine besondere Überraschung präsentieren und prägen sich das Ergebnis ein. Noch direkter lassen sich manche Performance-Probleme übrigens in der Ansicht *list of called methods* erkennen. Diese Tabelle etwa nach *Own duration* sortiert entblößt schnell die Methode *AccumulateSelfAndChildrenRecursive* als diejenige, in der die meiste Zeit verbraucht wird. In der Spalte *Show Details* lässt sich ein Kontextmenü öffnen, das einen über *Who called that Method* direkt in eine graphische Visualisierung (callstack tree graph) des verbrauchten Zeitaufwandes bringt.

Schritt 6: Minimalinvasive Chirurgie

Jetzt, da Sie wissen, wo Sie eingreifen müssen, ist die Operation einfach. Im Profiler haben Sie bereits erkannt, dass die Methode *DecompileMethod* die richtige Stelle für Ihren Eingriff ist. Dort gibt es einen Methodenparameter der offensichtlich Textausgaben für das angezeigte Dekompilat entgegennimmt. Sie fügen also Ihren rechtlichen Warnhinweis direkt zu Beginn der Methode hinzu und starten den modifizierten ILSpy (Abbildung 5).

Schritt 7: Loorbeeren ernten

Gerade da hören Sie gehetzte Schritte auf dem Flur. Ein tief roter Kopf erscheint in der Tür: „Ich bin extra noch mal umgekehrt um Ihnen das zu sagen: Diesmal dürfen wir den Termin nicht verschieben, keine Chance! Wenn wir bis Mittag nicht..."

Sie fallen Ihrem Chef ins Wort: „Ach was, diese Kleinigkeit! Ist doch längst fertig. Und wussten Sie, dass das langsamste am Dekompilieren einer Methode das Entfernen von Gotos ist?"

Und während die Gesichtsfarbe des mit geöffnetem Mund vor Ihnen stehenden Chefs langsam wieder normale Töne annimmt, nehmen Sie ihre Jacke und verabschieden sich: „Ich geh dann heute früher in die Mittagspause, ich hab ja im Moment eh nichts zu tun."

Fazit

InsightProfiler sollte in keiner .NET-Werkzeugkiste fehlen. Mit seiner Hilfe erlangen Sie schnell Überblick über die Abläufe in einem Stück .NET-Software. Das Tool gibt es in einer kostenlosen Variante, der dotnetpro-Version auf der Heft-CD und einer Professional Version für 99 Euro. [tib]