NETBEANS PROFILER

TOMASZ ŁUKASZUK

STRESZCZENIE: Dokument zawiera podstawowe informacje dotyczące programu NetBeans Profiler. Stanowi uproszczoną instrukcję jego używania. Dotyczy NetBeans Profiler w wersji 6.0 i profilowania kodu języka java.

1. WPROWADZENIE

NetBeans Profiler jest wydajnym narzędziem diagnostycznym udostępniającym informacje o zachowaniu się aplikacji w czasie wykonania. Umożliwia deweloperowi precyzyjną kontrolę nad tym, które części aplikacji są diagnozowane czym skraca czas diagnozy, a poprzez uszczegółowienie wyników ułatwia ich interpretację. Diagnozowana aplikacja może być uruchamiana zarówno lokalnie, jak i na systemie zdalnym. Profiler śledzi stany wątków, wykorzystanie procesora i zużycie pamięci systemowej. Poprzez ścisłe wpisanie się w kolejność działań wykonywanych w NetBeans IDE, Profiler ułatwia wyśledzenie problemów z wydajnością i nieefektywnym użyciem pamięci.

2. MOŻLIWOŚCI NETBEANS PROFILER

(1) Monitorowanie aplikacji w czasie wykonania
Aplikacja jest uruchamiana bez instrumentacji. Dostarczane są wysokiego poziomu informacje o kilku ważnych parametrach maszyny JVM (aktywność wątków, alokacja pamięci). Ten tryb może być używany do wstępnej, ogólnej analizy.
(2) Analiza wydajności aplikacji
Dostarczane są szczegółowe dane dotyczące wydajności aplikacji, między innymi czasy wykonania poszczególnych metod i ilości ich wywołań.
(3) Analiza zużycia pamięci
Dostarczane są szczegółowe informacje dotyczące alokacji obiektów, stanu i działania garbage collection.

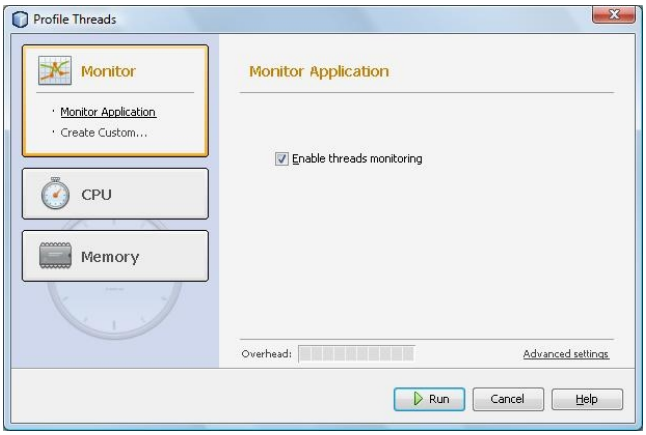
3. ANALZA APLIKACJI ZA POMOCĄ NETBEANS PROFILER

(1) Pierwsze uruchomienie Profiler'a
Uruchamiając Profiler po raz pierwszy, środowisko NetBeans IDE musi

dokonać kilku operacji inicjalizujących, aby móc zbierać dane i prezentować wiarygodne wyniki pozwalające na analizę aplikacji.
Przed pierwszym użyciem Profiler'a należy dokonać jego kalibracji. Kalibrację należy wykonać tylko raz, chyba że parametry maszyny, na której pracuje narzędzie ulegną zmianom, wówczas należy dokonać ponownej kalibracji. Kalibracja dostępna jest w menu:
Profiler > Advanced Commands > Run Profiler Calibration

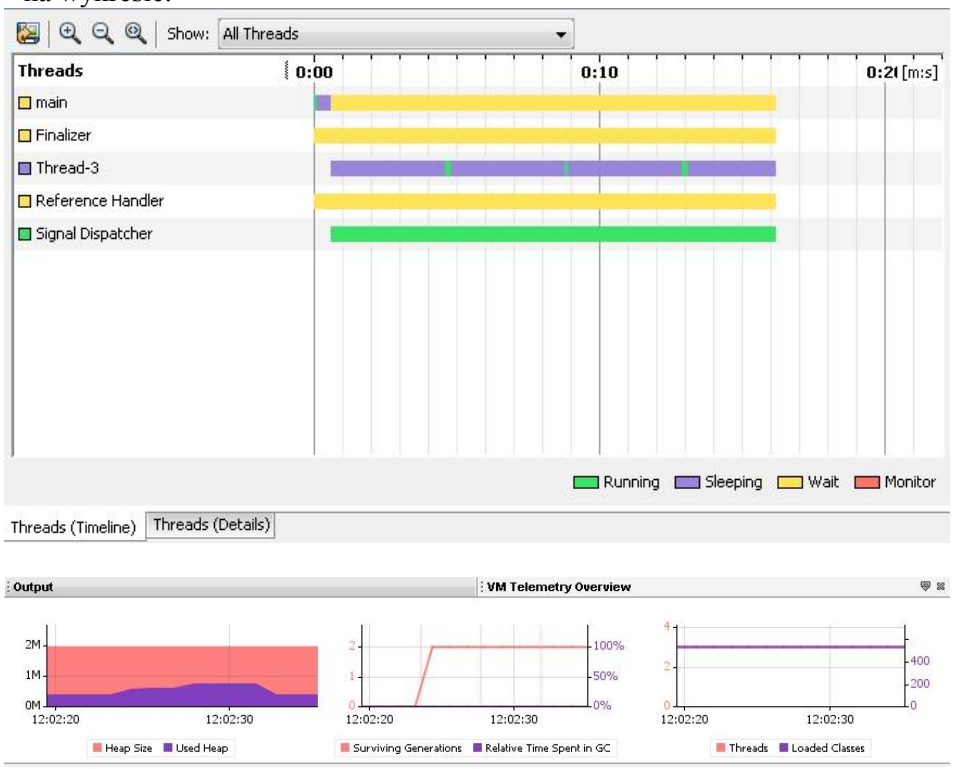
Pierwszy raz uruchamiając Profiler dla określonego projektu wyświetlone zostanie okienko dialogowe z informacją, że środowisko musi dokonać integracji projektu z Profiler'em. Integracja polega na modyfikacji skryptu uruchomieniowego build.xml projektu poprzez dodanie do niego wpisów o koniecznych bibliotekach używanych przez Profiler.

(2) Monitorowanie aplikacji
- upewnij się, że aplikacja, którą chcesz monitorować jest ustawiona jako główny projekt w środowisku IDE
- wybierz z menu polecenie Profile > Profile Main Project
- wybierz Monitor w oknie dialogowym wyboru zadania profilowania
- jeżeli chcesz monitorować pracę wątków występujących w aplikacji zaznacz opcję Enable threads monitoring
- uruchom analizę klikając przycisk Run



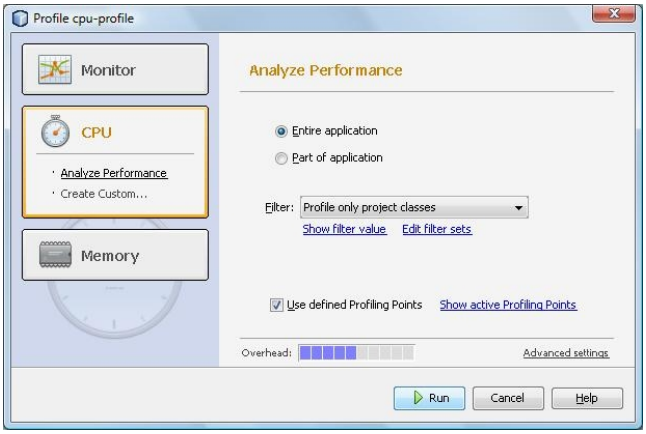
Aplikacja zostanie uruchomiona i w trakcie jej działania będzie prezentowany aktualny stan wątków (o ile wybrano opcję Enable threads monitoring) oraz w oknie Telemetry Overview podstawowe informacje o stanie aplikacji (ilość wykorzystywanej pamięci, czas spędzany w garbage collection, ilość używanych klas, wątków). Po najechaniu kursorem myszy na wybrany wykres zostaną pokazane bardziej szczegółowe statystyki o elemencie prezentowanym

na wykresie.

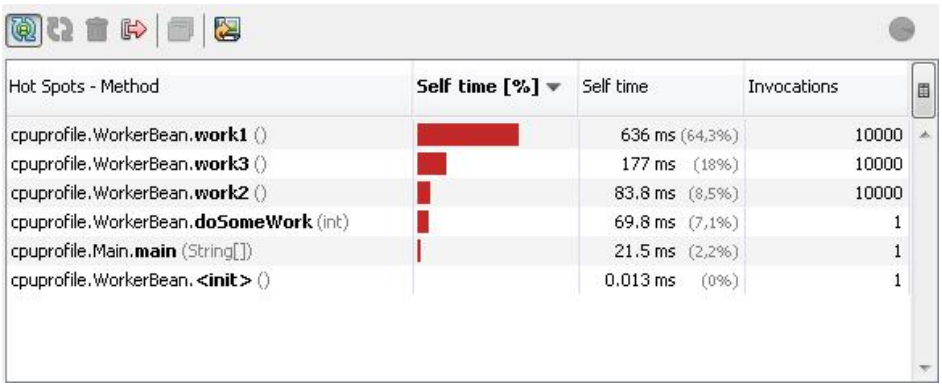


(3) Analiza wydajności CPU

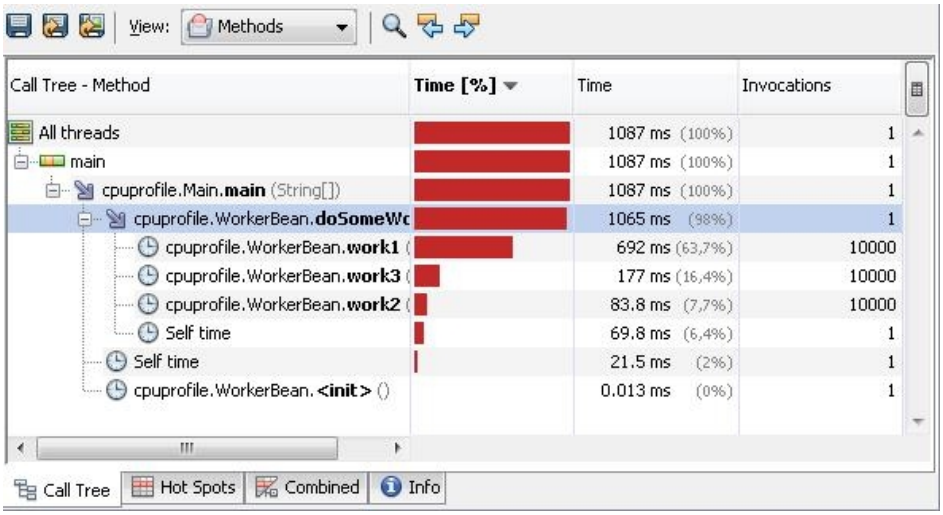
- upewnij się, że aplikacja, którą chcesz monitorować jest ustawiona jako główny projekt w środowisku IDE
- wybierz z menu polecenie Profile > Profile Main Project
- wybierz CPU w oknie dialogowym wyboru zadania profilowania
- ustaw opcje analizy: czy analizowana ma być cała aplikacja (Entire Application) czy tylko wybrana jej część (Part of Application, należy wskazać tzw. root methods, czyli metody, klasy lub pakiety które mają być analizowane, wyniki będą zbierane, kiedy wątek aplikacji przeniesie się do root method), filtrowanie analizowanych klas (Profile all classes, Profile only project classes, Quick filter, ...)
- aby obserwować jak zmienia się wykorzystanie czasu procesora przez poszczególne funkcje w czasie wykonania aplikacji otwórz okno Live Results (Window > Profiling > Live Results)
- uruchom analizę klikając przycisk Run



Aplikacja zostanie uruchomiona i będzie analizowana w czasie wykonywania. Na bieżąco w oknie Live Results będzie prezentowane wykorzystanie czasu procesora przez poszczególne metody i ilości ich wywołań. Po zakończeniu działania aplikacji całościowe wyniki wydajności zostaną zaprezentowane w oddzielnym oknie (zdjęcie stanu aplikacji) w formie grafu wywołań (Call Tree) oraz listy funkcji z informacjami o ich czasie wykonania i liczbie wywołań (Hot Spots).

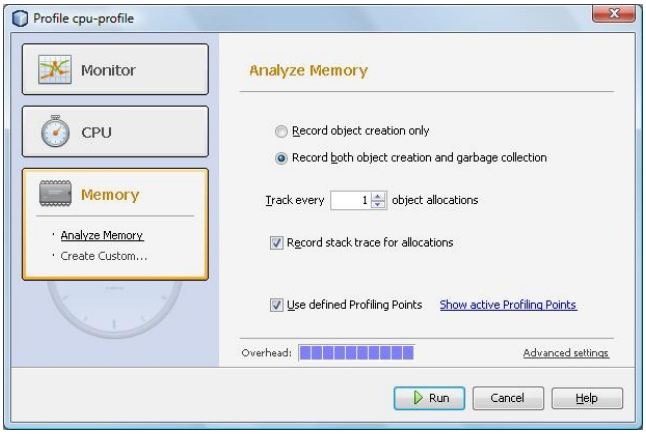


Hot Spots - Method	Self time [%]	Self time	Invocations
cpuprofile.WorkerBean.**work1** ()		636 ms (64,3%)	10000
cpuprofile.WorkerBean.**work3** ()		177 ms (18%)	10000
cpuprofile.WorkerBean.**work2** ()		83,8 ms (8,5%)	10000
cpuprofile.WorkerBean.**doSomeWork** (int)		69,8 ms (7,1%)	1
cpuprofile.Main.**main** (String[])		21,5 ms (2,2%)	1
cpuprofile.WorkerBean.**<init>** ()		0,013 ms (0%)	1

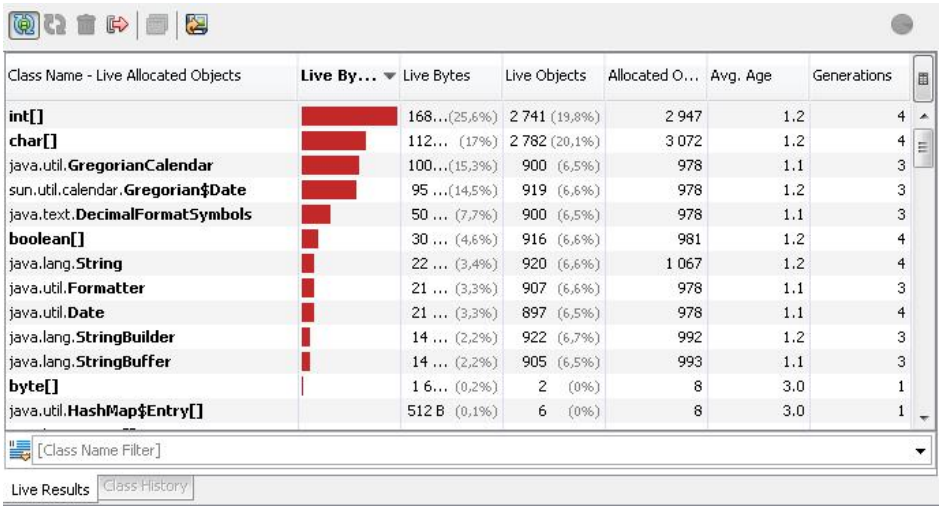


(4) Analiza zużycia pamięci

- upewnij się, że aplikacja, którą chcesz monitorować jest ustawiona jako główny projekt w środowisku IDE
- wybierz z menu polecenie Profile > Profile Main Project
- wybierz Memory w oknie dialogowym wyboru zadania profilowania
- ustaw opcje analizy: czy zbierane mają być dane o alokacji obiektów (Record object creation only), czy rozszerzone dane o życiu obiektów, jak liczba istniejących obiektów poszczególnych typów, ich rozmiar i średni czas życia (Record both object creation and garbage collection), pierwsza opcja jest podzbiorem drugiej; czy mają być także zbierane dane o stanie stosu wywołań funkcji (Record stack traces for allocation, pomocne przy robieniu zdjęcia stanu pamięci); co który obiekt ma być analizowany (Track every N object allocations)
- aby obserwować jak zmienia się wykorzystanie pamięci w czasie wykonania aplikacji otwórz okno Live Results (Window > Profiling > Live Results)
- uruchom analizę klikając przycisk Run



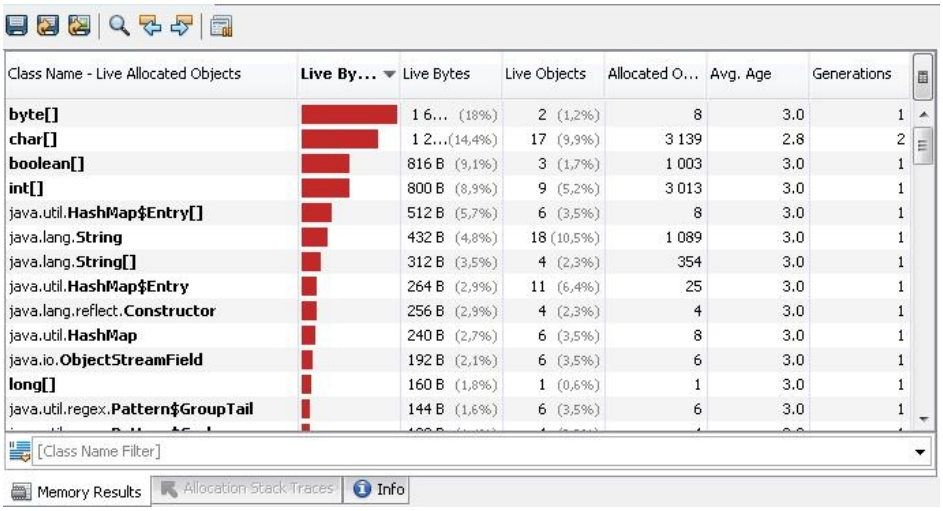
Po uruchomieniu aplikacji na bieżąco w oknie Live Results prezentowany jest stan pamięci z wyszczególnieniem poszczególnych zaalokowanych obiektów. Natomiast po zakończeniu aplikacji w osobnym oknie prezentowany jest stan pamięci (zdjęcie stanu aplikacji), kiedy aplikacja już nie działa.



Class Name - Live Allocated Objects	Live By...	Live Bytes	Live Objects	Allocated O...	Avg. Age	Generations
int[]		168... (25,6%)	2 741 (19,8%)	2 947	1.2	4
char[]		112... (17%)	2 782 (20,1%)	3 072	1.2	4
java.util.**GregorianCalendar**		100... (15,3%)	900 (6,5%)	978	1.1	3
sun.util.calendar.**Gregorian$Date**		95 ... (14,5%)	919 (6,6%)	978	1.2	3
java.text.**DecimalFormatSymbols**		50 ... (7,7%)	900 (6,5%)	978	1.1	3
boolean[]		30 ... (4,6%)	916 (6,6%)	981	1.2	4
java.lang.**String**		22 ... (3,4%)	920 (6,6%)	1 067	1.2	4
java.util.**Formatter**		21 ... (3,3%)	907 (6,6%)	978	1.1	3
java.util.**Date**		21 ... (3,3%)	897 (6,5%)	978	1.1	4
java.lang.**StringBuilder**		14 ... (2,2%)	922 (6,7%)	992	1.2	3
java.lang.**StringBuffer**		14 ... (2,2%)	905 (6,5%)	993	1.1	3
byte[]		1 6... (0,2%)	2 (0%)	8	3.0	1
java.util.**HashMap$Entry[]**		512 B (0,1%)	6 (0%)	8	3.0	1

[Class Name Filter]

Live Results | Class History



(5) Zdjęcia stanu aplikacji

W każdym momencie analizy aplikacji możliwe jest zrobienie zdjęcia jej stanu. Zaletami tego mechanizmu są:

- możliwość analizowania aplikacji na podstawie zdjęcia, kiedy sesja profilowania się zakończy
- zdjęcie zawiera bardziej szczegółowe informacje niż dane otrzymywane „na żywo”
- zdjęcia mogą być pomiędzy sobą porównywane, co pozwala na obserwowanie rezultatów zmian usprawniających działanie aplikacji (np. poprawę wydajności).

BIBLIOGRAFIA

(1) http://www.netbeans.org/kb/60/java/profiler-intro.html
(2) http://pl.wikipedia.org/wiki/NetBeans

Politechnika Białostocka
Katedra Oprogramowania
ul. Wiejska 45A
15-351 Białystok

17 marca 2008