

# Java Profiler своими руками

Владимир Поляков  
IHC.RU, Zaycev.net





# Profiling

Динамический анализ потребляемых ресурсов

- Время выполнения
- Частота вызовов
- Занимаемая память



# Profiling

- Инструментирующий (`currentTimeMillis`)
- Сэмплирующий (Монте-Карло)



# Теория

Do It Yourself Java Profiling (Роман Елизаров)

<https://habrahabr.ru/post/143468/>



# Safepoint bias

Точка, в которой JIT-скомпилированный код может остановиться

- Stop-the-world GC
- Hot swap
- ...
- Стектрейсы!

<https://github.com/jvm-profiling-tools/async-profiler>



# Память

Тоже можно измерить

<https://github.com/google/allocation-instrumenter>



## Практика

Что делать если нужно просто узнать на чем приложение тормозит и почему?

**46.5%**  
**VisualVM**

**25.7%**  
**JProfiler**

**20.6%**  
Custom in-house tools

**17.1%**  
Java  
Mission  
Control

**15.0%**  
YourKit

**13.9%**  
None

**8.4%**  
Don't know

**6.2%**  
Other

**9.1%**  
NetBeans profiler

**3.5%** JProbe

**3.3%** XRebel





## Практика

Что делать если нужно просто узнать на чем приложение тормозит и почему?

- Вставить профайлер в приложение и отправить в продакшен



## Зачем в production?

- Реальные данные
- Внешние сервисы
- Инфраструктура (железо, сеть, ...)



## Плюсы

- Специализированное решение

Знает про вашу архитектуру, особенности кода и значимые данные

- Позволяет сразу нырнуть в конкретный кейс

Дополнительно можно себе помочь, наполняя трейс данными из кода

- Тонкая настройка того, что хочется видеть



## Минусы

- Есть накладные расходы
- Нужно встраивать в приложение
- Профайлер сам себя не напишет

Итак,



## Шаг 1

Вспомогательные классы для записи трейсов

- Record
- Call
- Profiler



## Шаг 2

Начало и окончание трейса

- ServletFilterConfiguration (или web.xml)
- ProfilingServletFilter (выше spring security)
- ThreadNameServletFilter (ниже spring security)



## Шаг 3

Запись вызовов (Нужно инструментировать!)

- ProfilingClassFileTransformer
- Agent
- Манифест
- -javaagent:agent/target/agent-0.0.1-SNAPSHOT-final.jar





## Шаг 4

Вывод того что получилось

- RecordUtil
- System.out.println(), Telegram, БД, ...
- Визуализируем! ([vis/index.html](http://vis/index.html))



## Шаг 5

БД!

- <https://github.com/ttddyy/datasource-proxy>
- AirDbConfig и MainDbConfig
- RecordUtil - QueryTop



## Шаг 6

Тюнинг фильтров. Что интересно, а что нет?

- `TransformerFilters`



## Шаг 7

Переносим инструментирование на этап  
компиляции

- pom.xml
- instrumenter



## Код

<https://github.com/drxaos-edu/instrumenting-profiler>

**Вопросы?**

