JDK Flight Recorder в 20 25 JDK

Алексей Рагозин
alexey.ragozin@gmail.com

Москва, Октябрь 2025

Год 2025

JFR технология с историей

Новые фичи с каждой версией OpenJDK

Что мы имеем сегодня?

JDK 25

JEP 509 / JEP 518 – улучшения семплирования

JEP 520 – трассировка методов

Новые аннотации для прикладных событий

Что такое Java Flight Recorder?

Профайлеры делятся на

- Хорошие
- Бесплатные
- Java Flight Recorder

Эпоха JRockit

JDK Flight Recorder появился в JRockit JVM

Идея “черного ящика”

- Собираем сигналы в процессе работы
- Сохраняем файл с записью
  (бинарный формат, метаданные)
- Можем восстановить картину инцидента по записи

2008 – BEA Systems приобретена Oracle

2010 – Последняя версия JRockit - R28

Эпоха Java 8

В Java 7, JDK Flight Recorder появляется в HotSpot JVM

В Java 8 дистрибутив от Oracle включает

- JDK Flight Recorder
- Mission Control
- Visual VM

JFR – проприетарная технология

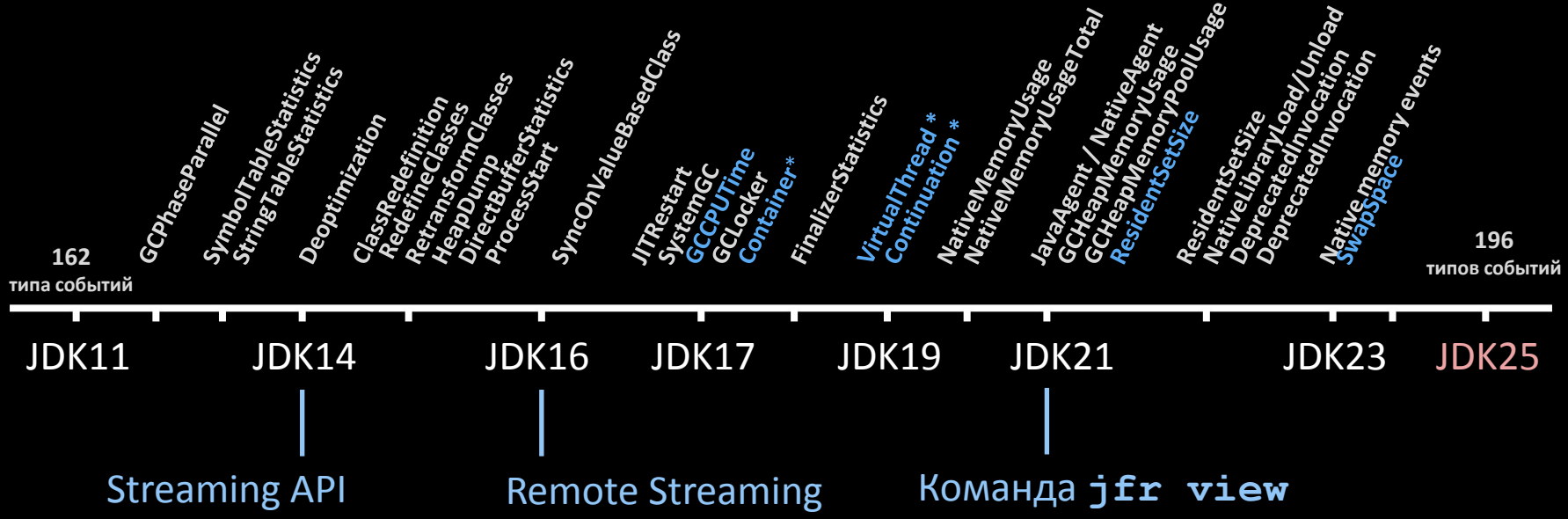
Печально известный флаг
`-XX:+UnlockCommercialFeatures`

Эпоха Open JDK

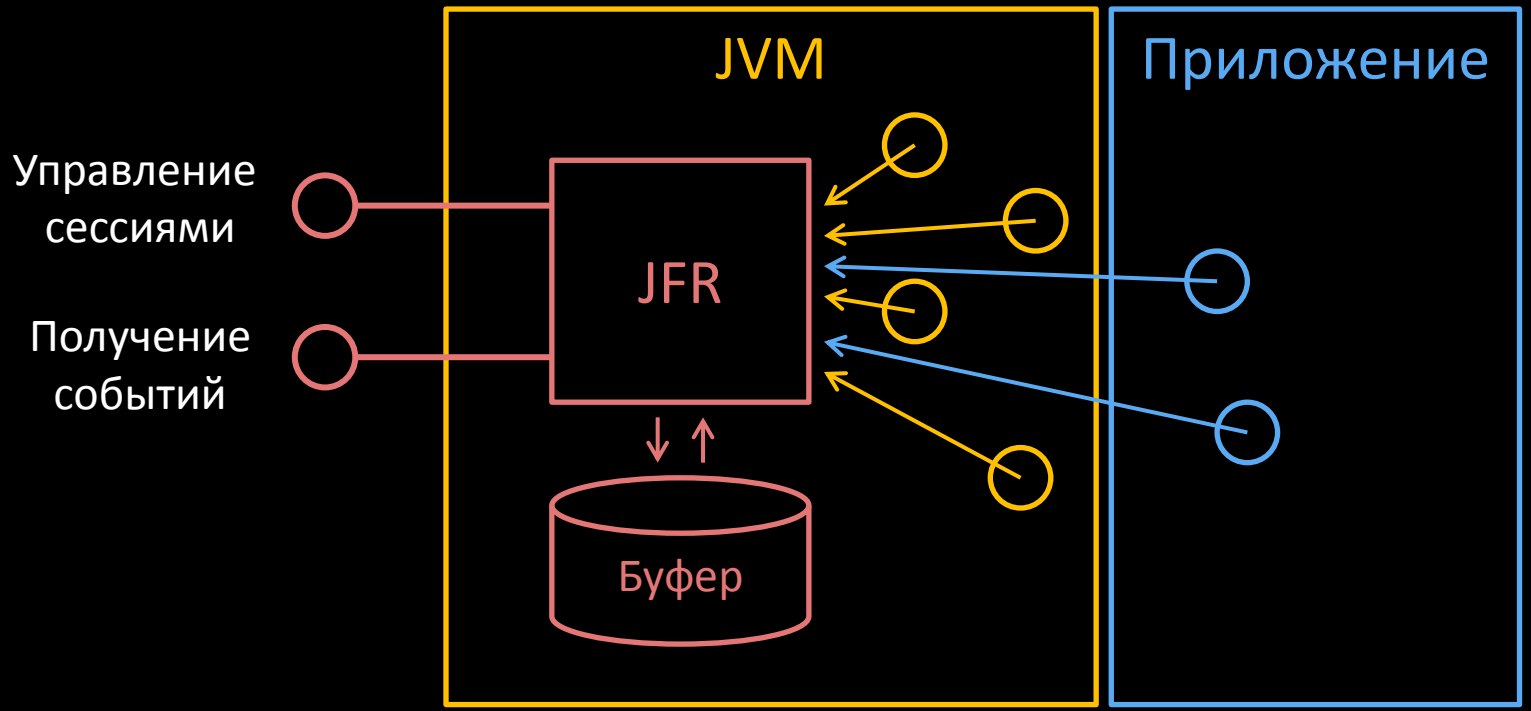
С появлением OpenJDK, JDK Flight Recorder перерождается как open source продукт

В Open JDK 11

- Новый JDK Flight Recorder
- Стандартная лицензия
- Mission Control – отдельный продукт
- API для прикладных событий и интеграции
- Инструменты командной строки


162
типа событий
GCPhaseParallel
SymbolTableStatistics
StringTableStatistics
Deoptimization
ClassRedefinition
RedefineClasses
RetransformClasses
HeapDump
DirectBufferStatistics
ProcessStart
SyncOnValueBasedClass
JITRestart
SystemGC
GCCPUTime
GCLocker
Container*
FinalizerStatistics
VirtualThread *
Continuation *
NativeMemoryUsage
NativeMemoryUsageTotal
JavaAgent / NativeAgent
GCHeapMemoryUsage
GCHeapMemoryPoolUsage
ResidentSetSize
ResidentSetSize
NativeLibraryLoad/Unload
DeprecatedInvocation
DeprecatedInvocation
Native memory events
SwapSpace
196
типов событий
JDK11
JDK14
JDK16
JDK17
JDK19
JDK21
JDK23
JDK25
Streaming API
Remote Streaming
Команда jfr view

Архитектура Java Flight Recorder



События JFR

Простая структура данных

- Строгая схема для каждого типа событий
- Атрибуты
- Опциональный стек-трейс

Удобно представлять в виде JSON

События JVM

Конфигурация и окружение

Переменные окружения / настройки JVM / конфигурация железа / …

“Кишочки” JVM

Компилятор / safepoint / сборка мусора / …

Профилирование

Сэмплирование / IO / блокировки / exceptions / …

Здесь могут быть ваши события!

Исчерпывающий каталог по всем версиям

https://sap.github.io/SapMachine/jfrevents

Доступные API

- Управление JFR

  Как запустить JFR?

  на старте JVM

  - `java -XX:StartFlightRecording:filename=/path/file.jfr`

  для локального процесса

  - `jcmd <pid> JFR.start duration=60s filename=myrecording.jfr`

  удалённо

  - Mission Control / JMX / …

  или через API

Доступные API

- Управление JFR
- Стриминг событий

  Позволяет получать события с малой задержкой для обработки

Virtual thread – свежая фича JVM

До JDK24 была актуальна
проблема с synchronized блоками

Мы хотим видеть в мониторинге есть ли у нас такая проблема?

Обработка событий

```
private final Timer meterRegistry.timer("jfr.thread.pinning");

void handle(RecordedEvent event) {
   var thread=event.getThread() != null ? event.getThread().getJavaName() : "<unknown>";
   var duration=event.getDuration();
   var startTime=LocalDateTime.ofInstant(event.getStartTime(),ZoneId.systemDefault());
   var stackTrace=formatStackTrace(event.getStackTrace(),STACK_TRACE_MAX_DEPTH);

   log.warn("Thread '{}' pinned for: {}ms at {}, stacktrace: \n{}",
      thread,
      duration.toMillis(),
      startTime,
      stackTrace
   );

   timer.record(duration);
}
```

Создание и запуск сессии

```
@Component
class JfrEventLifecycle implements SmartLifecycle {
...
   @Override
   public synchronized void start() {
      recordingStream = new RecordingStream();

      recordingStream.enable("jdk.VirtualThreadPinned").withStackTrace();
      recordingStream.onEvent(
            "jdk.VirtualThreadPinned", virtualThreadPinnedEventHandler::handle);

      recordingStream.setMaxAge(Duration.ofSeconds(10));

      recordingStream.startAsync();
      running = true;
   }
}
...
```

jfr-thread-pinning-spring-boot - *https://github.com/mikemybytes/jfr-thread-pinning-spring-boot*

Доступные API

- Управление JFR
- Стриминг событий
- Публикация событий

Позволяет добавлять новые события и реализовывать точки сбора телеметрии

Публикация событий

Создать свой тип событий это просто!

```java
package com.example.restservice;

import jdk.jfr.Event;

public class RestCallEvent extends Event {
    public String path;
    public String key;
    public long dataSize;
}
```

```java
@GetMapping("/set")
public String set(
        @RequestParam(value = "key") String key,
        @RequestParam(value = "val") String val) {
    RestCallEvent event = new RestCallEvent();
    event.begin();
    event.key = key;
    try {
        store.put(key, val);
        event.dataSize = val.length();
        return val;
    } finally {
        event.end();
        event.commit();
    }
}
```

Доступные API

- Управление JFR
- Стриминг событий
- Публикация событий
- Чтение JFR дампов

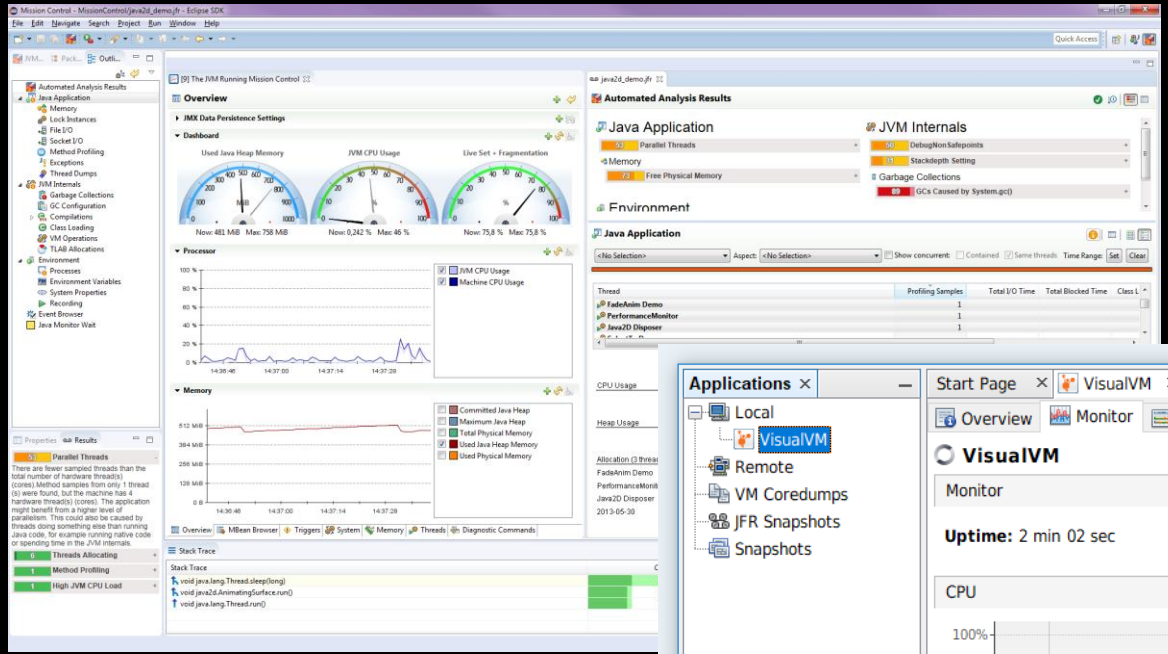
Позволяет читать события из файлов в формате JFR

Доступные API

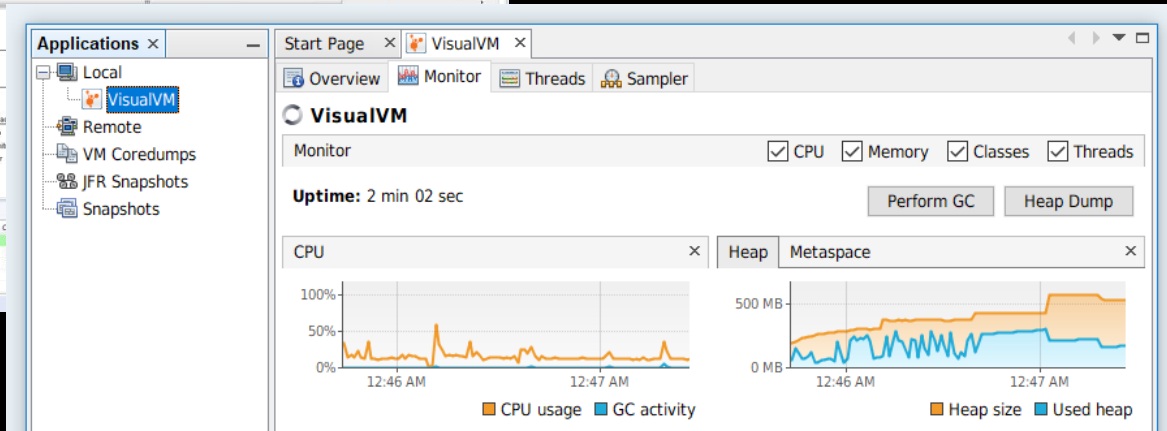
- Управление JFR
- Стриминг событий
- Публикация событий
- Чтение JFR дампов

Инструменты для работы с JFR

Mission control



Visual VM



Работа из командной строки

```
jcmd <pid> JFR.start duration=60s filename=myrecording.jfr
```

```
15405:
Started recording 1. The result will be written to:

/path/to/myapp/myrecording.jfr
```

```
jfr print --json --events ThreadCPULoad myrecording.jfr
```

```
{
  "recording": {
    "events": [{
      "type": "jdk.ThreadCPULoad",
      "values": {
        "startTime": "2025-10-01T05:42:25.358108188+03:00",
        "eventThread": {
          "osName": "C2 CompilerThread1",
          "osThreadId": 15733,
          "javaName": "C2 CompilerThread1",
          "javaThreadId": 19,
          "group": {
            "parent": null,
            "name": "system"
          },
          "virtual": false
...
```

JQ Кон-фу

```
jfr print --json --events ThreadCPULoad
    myrecording.jfr | jq '
[.recording.events[]
  | select(.type == "jdk.ThreadCPULoad")
  | {
      javaName: .values.eventThread.javaName,
      user: .values.user,
      system: .values.system} ]
  | group_by(.javaName)
  | map({
    javaName: .[0].javaName,
    user: (map(.user) | add),
    system: (map(.system) | add)
  }) | sort_by(-.user)'
```


```
[
  {
    "javaName": "C2 CompilerThread1",
    "user": 1.4746960910000002,
    "system": 0.0628239843
  },
  {
    "javaName": "C2 CompilerThread0",
    "user": 0.27601727600000003,
    "system": 0.00119408049
  },
  {
    "javaName": "C1 CompilerThread0",
    "user": 0.11426830399999999,
    "system": 0.00334142412
  },
  {
    "javaName": "DestroyJavaVM",
    "user": 0.029225044,
    "system": 0.0012193053
  },
  {
    "javaName": "http-nio-8080-exec-8",
    "user": 0.0218696855,
    "system": 0.0008536230300000001
  },
...
```

или ...

```
jfr view thread-cpu-load myrecording.jfr
```

Thread CPU Load

Thread	System	User
C2 CompilerThread1	0.36%	5.29%
DestroyJavaVM	0.12%	2.92%
C2 CompilerThread0	0.02%	2.51%
C1 CompilerThread0	0.02%	0.40%
http-nio-8080-exec-10	0.00%	0.22%
http-nio-8080-exec-14	0.00%	0.22%
http-nio-8080-exec-3	0.00%	0.20%
http-nio-8080-exec-8	0.00%	0.20%
http-nio-8080-exec-1	0.00%	0.17%
http-nio-8080-exec-4	0.00%	0.17%
http-nio-8080-exec-6	0.01%	0.17%
http-nio-8080-exec-2	0.01%	0.15%
http-nio-8080-exec-7	0.00%	0.15%
http-nio-8080-exec-11	0.00%	0.15%
http-nio-8080-exec-12	0.02%	0.15%
http-nio-8080-exec-13	0.00%	0.15%
Attach Listener	0.01%	0.13%
http-nio-8080-exec-9	0.00%	0.12%
http-nio-8080-exec-5	0.00%	0.10%
JFR Periodic Tasks	0.02%	0.07%
...		

можно даже без файла

```
jcmd $PID JFR.view thread-cpu-load
```

Thread CPU Load

Thread	System	User
C2 CompilerThread1	0.36%	5.29%
DestroyJavaVM	0.12%	2.92%
C2 CompilerThread0	0.02%	2.51%
C1 CompilerThread0	0.02%	0.40%
http-nio-8080-exec-10	0.00%	0.22%
http-nio-8080-exec-14	0.00%	0.22%
http-nio-8080-exec-3	0.00%	0.20%
http-nio-8080-exec-8	0.00%	0.20%
http-nio-8080-exec-1	0.00%	0.17%
http-nio-8080-exec-4	0.00%	0.17%
http-nio-8080-exec-6	0.01%	0.17%
http-nio-8080-exec-2	0.01%	0.15%
http-nio-8080-exec-7	0.00%	0.15%
http-nio-8080-exec-11	0.00%	0.15%
http-nio-8080-exec-12	0.02%	0.15%
http-nio-8080-exec-13	0.00%	0.15%
Attach Listener	0.01%	0.13%
http-nio-8080-exec-9	0.00%	0.12%
http-nio-8080-exec-5	0.00%	0.10%
JFR Periodic Tasks	0.02%	0.07%
...		

```
jfr view --help
```

```
Java virtual machine views:
 class-modifications       gc-concurrent-phases longest-compilations
 compiler-configuration    gc-configuration     native-memory-committed
 compiler-phases           gc-cpu-time          native-memory-reserved
 compiler-statistics       gc-pause-phases      safepoints
 deoptimizations-by-reason gc-pauses            tlabs
 deoptimizations-by-site   gc-references        vm-operations
 gc                        heap-configuration

Environment views:
 active-recordings        cpu-information       jvm-flags
 active-settings          cpu-load              native-libraries
 container-configuration  cpu-load-samples      network-utilization
 container-cpu-throttling cpu-tsc               recording
 container-cpu-usage      environment-variables system-information
 container-io-usage       events-by-count       system-processes
 container-memory-usage   events-by-name        system-properties

Application views:
 allocation-by-class   exception-count       native-methods
 allocation-by-site    file-reads-by-path    object-statistics
 allocation-by-thread  file-writes-by-path   pinned-threads
 class-loaders         finalizers            socket-reads-by-host
 contention-by-address hot-methods           socket-writes-by-host
 contention-by-class   latencies-by-type     thread-allocation
 contention-by-site    longest-class-loading thread-count
 contention-by-thread  memory-leaks-by-class thread-cpu-load
 exception-by-message  memory-leaks-by-site  thread-start
```

2025

JDK 25

Open JDK 25

Улучшение сэмплирования

- JEP 509 – CPU Time Profiling
- JEP 518 – JFR Cooperative Sampling

Трассировка методов

Улучшение для прикладных событий

- @Contextual
- @Throttle

Проблемы сэмплирования JFR

Два события

- `jdk.ExecutionSample`
- `jdk.NativeMethodSample`

Код JVM слепая зона

Потеря сэмплов

`-XX:+DebugNonSafepoints` – проблема всех сэмплеров

JEP 509/518

Новое событие

`jdk.CPUTimeSample#enabled=true`

Использует perf счётчики (только Linux)

Сэмплируются только потоки на CPU

Не теряются сэмплы

Эксперементальный статус

Трассировка методов

Реализована через инструментации

Два режима

- MethodTiming – статистика вызовов
- MethodTrace – событие на каждый вызов

Не требует агентов!

```
jcmd $PID JFR.start method-trace=org/…/SimpleJpaRepository

jcmd $PID JFR.view method-calls
```

Method Calls

Traced Method	Caller	Invocations
org.springframework./…/.SimpleJpaRepository.applyComment...	org.springframework./…/.SimpleJpaRepository.getHints()	2,067
org.springframework./…/.SimpleJpaRepository.getQueryHints()	org.springframework./…/.SimpleJpaRepository.getHints()	2,067
org.springframework./…/.SimpleJpaRepository.getHints()	org.springframework./…/.SimpleJpaRepository.findById(...)	2,067
org.springframework./…/.SimpleJpaRepository.getDomainCla...	org.springframework./…/.SimpleJpaRepository.findById(...)	2,067
org.springframework./…/.SimpleJpaRepository.findById(...)	java.lang.invoke.Lam/…/000052108000.invokeVirtual(...)	2,067
org.springframework./…/.SimpleJpaRepository.save(Object)	java.lang.invoke.Lam/…/000052108000.invokeVirtual(...)	345

Timespan: 20:09:04 - 20:19:04

```
jfr print --events jdk.MethodTrace recording.jfr
```

```
jdk.MethodTrace {
  startTime = 18:46:50.941 (2025-09-16)
  duration = 3.53 ms
  method = org.springframework.data.jpa.repository.support.SimpleJpaRepository.save(Object)
  eventThread = "http-nio-8080-exec-7" (javaThreadId = 55)
  stackTrace = [
    jdk.internal.reflect.DirectMethodHandleAccessor.invoke(Object, Object[]) line: 104
    java.lang.reflect.Method.invoke(Object, Object[]) line: 565
    org.springframework.aop.support.AopUtils.invokeJoinpointUsingReflection(Object, Method, Object[]) line: 359
    org.springframework.data.repository.core.support.RepositoryMethodInvoker$RepositoryFragmentMethodInvoker...
    org.springframework.data.repository.core.support.RepositoryMethodInvoker.doInvoke(Class, RepositoryInvoc...
    ...
  ]
}

jdk.MethodTrace {
  startTime = 18:46:50.941 (2025-09-16)
  duration = 3.59 ms
  method = org.springframework.data.jpa.repository.support.SimpleJpaRepository.save(Object)
  eventThread = "http-nio-8080-exec-13" (javaThreadId = 66)
  stackTrace = [
    jdk.internal.reflect.DirectMethodHandleAccessor.invoke(Object, Object[]) line: 104
    java.lang.reflect.Method.invoke(Object, Object[]) line: 565
    org.springframework.aop.support.AopUtils.invokeJoinpointUsingReflection(Object, Method, Object[]) line: 359
    org.springframework.data.repository.core.support.RepositoryMethodInvoker$RepositoryFragmentMethodInvoker...
```

@Contextual

Класс события

```
@Label("Trace")
@Name("com.example.Trace")
class TraceEvent extends Event {
  @Label("ID")
  @Contextual
  String id;

  @Label("Name")
  @Contextual
  String name;
}
```

```
jdk.JavaMonitorEnter {
  Context: Trace.id = "00-0af7651916cd43dd8448eb211c80319c-00f067aa0ba902b7-01"
  Context: Trace.name = "POST /checkout/place-order"
  startTime = 17:51:29.038 (2025-02-07)
  duration = 50.56 ms
  monitorClass = java.util.ArrayDeque (classLoader = bootstrap)
  previousOwner = "Order Thread" (javaThreadId = 56209, virtual = true)
  address = 0x60000232ECB0
  eventThread = "Order Thread" (javaThreadId = 52613, virtual = true)
  stackTrace = [
   java.util.zip.ZipFile$CleanableResource.getInflater() line: 685
   java.util.zip.ZipFile$ZipFileInflaterStream.<init>() line: 388
   java.util.zip.ZipFile.getInputStream(ZipEntry) line: 355
   java.util.jar.JarFile.getInputStream(ZipEntry) line: 833
   ...
 ]
}
```

@Contextual

Корреляция на уровне парсинга

Не экспортируется в JSON через `jfr print --json` ☹

На практике работает странно

Пока ощущения смешанные

Заключение

Доступность – часть OpenJDK, доступен везде

Работа из консоли – `jcmd` / `jfr`

Mission Control – JFR фронтэнд с богатым функционалом

Экосистема

- API / интеграция IDE и другими профайлерами

Трассировка – закрыт последний гештальт

- Без агентов – работает из коробки

Ссылки

Каталог событий JFR для разных версий

https://sap.github.io/SapMachine/jfrevents

Памятка по запуску JFR из командной строки

https://habr.com/en/companies/krista/articles/532632/

What’s new in JFR in JDK25

https://egahlin.github.io/2025/05/31/whats-new-in-jdk-25.html

Работа с JDK Flight Recorder из командной строки

Вебинар - 13 ноября 2025

Регистрация

https://aragozin.timepad.ru/event/3627763/

Спасибо!

Email: alexey.ragozin@gmail.com
Блог: blog.ragozin.info
Github: https://github.com/aragozin
Митапы: https://aragozin.timepad.ru