Сверхзвуковая Java: знакомимся с Quarkus

Владимир Маслов

Консист Бизнес Групп

Сегодня в программе

- Как устроен Quarkus
- Стандарты MicroProfile
- GraalVM и нативные образы
- Как ускорить Spring Boot

Кто я

- Пишу бекенд на Java и Spring + Spring Boot
- Вхожу в ТОП-10 авторов Хабрахабра
  - *@HotWaterMusic*
- Три года делал игры
- Работал над Единой Мобильной Платформой в качестве бекенд-разработчика и младшего архитектора
- Снова Пишу бекенд на Java и Spring + Spring Boot

Есть решение!

Но есть пара проблем…

- Долго стартует
- Большой размер приложения
- Уверены, что разбираетесь во всей «магии» «под капотом»?

И что нам теперь делать?

Встречайте Quarkus

QUARKUS

Что ты такое?

- Создан RedHat
- Анонсирован в марте 2019
- Каждая новая версия приносит много новых фич
- Некоторые уже пишут на нем коммерческие приложения, и их уже устраивает
- P.S. Thorntail – всё.

Что ты такое?

- Создан для cloud-native, serverless и FaaS (AWS Lambda)
- Быстрый старт, малый футпринт в памяти
- DI времени компиляции
- Смотрит в сторону AOT

Создание проекта

```
mvn io.quarkus:quarkus-maven-plugin:0.23.2:create \
  -DprojectGroupId=org.acme \
  -DprojectArtifactId=getting-started \
  -DclassName="org.acme.quickstart.HelloResource" \
  -Dpath="/hello"
```

… или просто идем на code.quarkus.io

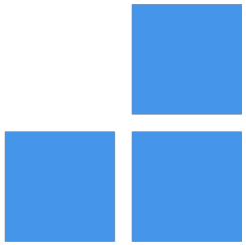
Пример контроллера Spring vs. Quarkus

Скорость запуска приложения

Возьмем обычный REST API...

Quarkus + GraalVM

15 MB



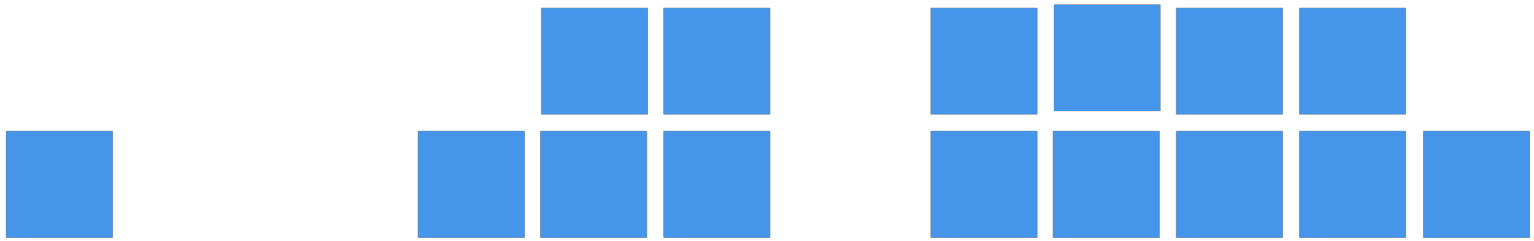
Quarkus + OpenJDK

75 MB

Spring + Boot + OpenJDK

150 MB

…и просто добавим CRUD



Quarkus + GraalVM

35 MB

Quarkus + OpenJDK

135 MB

Spring + Boot + OpenJDK

220 MB

Императивный? Реактивный? Без разницы!

```
@Inject
SpeakService say;

@GET
@Produces(MediaType.TEXT_PLAIN)
public String hi() {
        return say.hi();
}
```

```
@Inject
@Stream(”kafka”)
Publisher<String> reactiveSay;

@GET
@Produces(MediaType.SERVER_SENT_EVENTS)
public Publisher<String> stream() {
        return reactiveSay;
}
```

Можно смешивать оба подхода в одном приложении

Хочу, чтобы всё было как на Spring

Но как же я буду без Spring?

- Поддерживаются все ваши любимые аннотации Spring Dependency Injection
- Это не значит, что у вас стартует Spring Application Context
- И классы вроде BeanPostProcessor - тоже не заработают
- Но сможете писать Java Config
- Зато поддерживается основная часть Spring Data JPA и Spring Web
- Все это реализовано с помощью расширений

MicroProfile

Набор стандартов (спецификаций)
Enterprise Java для микросервисов
Быстро развивается (это вам не JavaEE)
Cloud Native
Реализация: библиотека SmallRye

Что еще?

- Vert.x
- Hibernate
- RESTEasy
- Netty
- Kubernetes/OpenShift
- Apache Camel
- Apache Kafka
- Метрики и Health Checks

HTTP-слой

- Нужны REST эндпоинты? В Quarkus есть RESTEasy для JAX-RS эндпоинтов
- Есть фильтры и интерцепторы
- Есть еще Undertow и Vert.x HTTP endpoint-ы – у них свои механизмы фильтрации
- Под капотом HTTP-слоя используется Vert.x

Как быть

Советы для Бута

Quarkus Arc

- CDI
- Основан на CDI 2.0. который работает поверх javax.inject
- Нет, это не Weld (который Red Hat делал 10 лет, и который слишком proxy-heavy)

Quarkus Arc

- Спецификация CDI сознательно реализована не целиком
- Есть ограничения
- Подробности: https://quarkus.io/guides/cdi-reference.html

Приватные члены классов

- Чтобы добраться до private членов класса, приходится использовать рефлексию
- Substrate VM это не нравится
- Поэтому не рекомендуется использовать их в своих бинах
- Просто делайте их package-private

```
@ApplicationScoped
public class CounterBean {
        @Inject
        CounterService counterService;

        void onMessage(@Observes Event msg) {}
}
```

Arc: старт приложения

- Загружаем метаданные
- Инициализируем структуры
- Всё, нет необходимости анализировать классы приложения снова
- Работает и в GraalVM, и в HotSpot

GraalVM

- Умеет собирать «нативный образ» - быстро стартующий машинный код без JVM.
- В нативной сборке сильно ограничена рефлексия.

GraalVM и рефлексия

- В нативной сборке сильно ограничена рефлексия.
- https://github.com/oracle/graal/blob/master/substratevm/REFLECTION.md

GraalVM в подробностях

Олег Чирухин — GraalVM Всемогущий
https://www.youtube.com/watch?v=lH4H0LEAo9g

Группа в Телеграм

Как быть

Как быть

Полезные ссылки

https://quarkus.io/

https://lordofthejars.github.io/quarkus-cheat-sheet/

https://developers.redhat.com/blog/2019/08/26/10-quarkus-videos-to-get-you-up-to-speed-with-supersonic-subatomic-java/

Заключение

Спасибо за внимание!

Вопросы?

https://t.me/hotwatercode

@vladimir_maslov

vlmr.maslov@gmail.com