Spring: your next Java micro-framework

О чем вообще речь?

[...] Typically, a microframework facilitates receiving an HTTP request, routing the HTTP request to the appropriate controller, dispatching the controller, and returning an HTTP response. Microframeworks are often specifically designed for building the APIs for another service or application.

- ❏ Легко разрабатывать
- ❏ Быстрые
- ❏ Простые
- ❏ Готовы к облаку

На Spring Boot легко разрабатывать

Что в этом помогает:

- Автоконфигурации
- Все остальное

Но другие микрофреймворки умеют Livereload!

- Enter DevTools
- Не просто Livereload, а Livereload везде
- Даже на продакшене!
- Но там лучше не надо

На Spring Boot легко разрабатывать

Spring Boot Dev Tools умеет многое

- Настроить дефолты - не кешировать шаблоны, DEBUG логирование
- Загрузить properties из ~/.config/spring-boot-devtools/
- Автоматический перезапуск при изменении Classpath
- Удаленный автоматический перезапуск
- LiveReload протокол

Микрофреймворки

- ✓ Легко разрабатывать
- ❑ Быстрые
- ❑ Простые
- ❑ Готовы к облаку

Spring Boot быстрый

Spring Boot 2.3+ можно запустить за 1 секунду

Benchmark	(sample)	Mode	Cnt	Score	Error	Units	Beans	Classes
MainBenchmark	actr	avgt	10	1.316	± 0.060	s/op	186	5666
MainBenchmark	jdbc	avgt	10	1.237	± 0.050	s/op	147	5625
MainBenchmark	demo	avgt	10	1.056	± 0.040	s/op	111	5266
MainBenchmark	slim	avgt	10	1.003	± 0.011	s/op	105	5208
MainBenchmark	thin	avgt	10	0.855	± 0.028	s/op	60	4892
MainBenchmark	lite	avgt	10	0.694	± 0.015	s/op	30	4580
MainBenchmark	func	avgt	10	0.652	± 0.017	s/op	25	4378

Dave Syer - How fast is Spring?

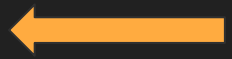
https://spring.io/blog/2018/12/12/how-fast-is-spring

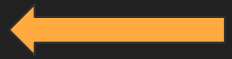
Spring Boot быстрый

```yaml
spring:
  security:
    oauth2:
      resourceserver:
        jwt:
          issuer-uri: https://id-srv.my-company.com/keys
      client:
        registration:
          id-srv:
            client-id:
            client-secret:
            grant-type: authorization-code
            provider: id-srv
        provider:
          id-srv:
            issuer-uri: https://id-srv.my-company.com/oauth/token
  jpa:
    hibernate:
      ddl-auto: validate

  cloud:
    config:
      uri: https://config-srv.my-company.com

eureka:
    client:
      register-with-eureka: true
      fetch-registry: true
```





Spring Boot быстрый

Spring Boot 2.3+ можно запустить за 1 секунду

...но можно и не запустить. Тогда можно попробовать:

- Ленивую загрузку бинов
- Ленивую инициализацию репозиториев
- Распаковать executable JARs и указать main class
- Реактивный стек
- Использовать `spring-context-indexer`
- Функциональные бины и контроллеры
- Жить без Actuator, если можете

Spring Boot быстрый

Ленивая загрузка бинов:

- Была доступна всегда с `@Lazy`
- Начиная с Spring Boot 2.3 можно включить глобально

```
spring.main.lazy-initialization=true
```

- Сохранят время запуска, но жертвует временем первого запроса

Spring Boot быстрый

Low hanging fruits для быстрого запуска

- Ленивая загрузка бинов

```
spring.main.lazy-initialization=true
```

- Ленивую инициализацию Spring Data JPA репозиториев

```
spring.data.jpa.repositories.bootstrap-mode=lazy
```

- Использовать `spring-context-indexer`

Spring Boot быстрый

Если Spring уже больше не может, пора разгонять JVM

- CPU это чаще всего bottleneck
- Разные JVMs (например J9)
- Использовать CDS
- Компилировать в нативный файл с GraalVM

Spring Boot быстрый

Что делать если все равно медленно? Профайлить!

Но перед этим можно посмотреть на /actuator/startup или на JFR + JMC


```
{
  "startupStep": {
    "name": "spring.beans.instantiate",
    "id": 95,
    "parentId": 5,
    "tags": [
      {
        "key": "beanName",
        "value": "slowpoke"
      }
    ]
  },
  "startTime": "2020-11-16T20:27:48.369306532Z",
  "endTime": "2020-11-16T20:27:49.373253743Z",
  "duration": "PT1.003947211S"
},
```

Микрофреймворки

- ✓ Легко разрабатывать
- ✓ Быстрые
- ❑ Простые
- ❑ Готовы к облаку

Spring Boot может быть простым

Но при этом перестанет быть таким легким в работе.

- ✓ Component scanning → Functional beans
- ✓ Route mapping → Functional routing
- ✓ Generated queries → `@Query` и не JPA единым

"Мне не нужен DI, я могу все сам описать в main"

Spring Boot может быть простым

Functional Beans Registration

- Вначале был XML для конфигурации контекста
- Ему на ~~смену~~ помощь пришли аннотации
- Теперь время функций

Spring Context = XML + Annotations + Functional Beans

Spring Boot может быть простым

Functional Routing

- Сначала появился в WebFlux - WebFlux.fn
- Теперь поддерживается и в WebMvc.fn
- Совсем прекрасен с Kotlin DSL

Routing = Annotations + Functional Routing

Микрофреймворки

✓ Легко разрабатывать

✓ Быстрые

✓ Простые *

❑ Готовы к облаку

* Но можно и пойти во все тяжкие с AOP и рефлексией

Spring Boot готов к облаку

What's in the box?

- Cloud Native движение началось со Spring Boot
- Spring Cloud
  - Circuit Breakers
  - Configuration
  - Service registry
  - Load balancing
  - API Gateways
- И даже Serverless
  - Spring Cloud Function

Spring Boot готов к облаку

Облачные платформы уже поддерживаются

- Kubernetes
  - Liveness probe
  - Readiness probe
  - Volume Mounted Config Directory Trees
  - Graceful shutdown
- Cloud Foundry
  - Bindings
  - Buildpacks
- AWS
- Alibaba
- И другие

Spring Boot готов к облаку

И даже не нужно писать Dockerfile-ы!

```
$ ./gradlew bootBuildImage
```

```
$ ./mvnw spring-boot:build-image
```

Cloud Native Buildpacks

https://buildpacks.io

Buildpack знает как собирать приложения за вас. И собирать правильно.

Код → Контейнер

- Модульность
- Слои
- Повторяемые билды

Paketo Buildpacks

https://paketo.io

Java Buildpack - все для Java

- Bellsoft Liberica
- Поддерживается Maven / Gradle / Leiningen / SBT
- Сертификаты

Java Native Image Buildpack

- GraalVM native images!

Spring Boot Buildpack

- Layered JAR ==> Docker layers
- Поддержка binding-ов

Spring Boot

Если он ходит как микрофреймворк и крякает как микрофреймворк…

- ✓ Быстрый
- ✓ Легко разрабатывать
- ✓ Простой
- ✓ Готов к облаку

Спасибо!

@alek_sys
github.com/alek-sys