

Перевод Spring Boot микросервисов с Java 8 на 11

Что может пойти (не) так?

Владимир Плизгá
ЦФТ

Привет!

- Владимир Плизгá
<https://github.com/Toparvion>
- ЦФТ (Центр Финансовых Технологий)
Один из **крупнейших** разработчиков ПО в России
- Backend-разработчик (Java)
7+ лет в деле
- TechLead в команде Интернет-банка



Интернет-банк для предоплаченных карт

- 20+ федеральных партнеров



- В топ-10 рейтинга Markswebb Mobile Banking Rank 2018
<http://markswebb.ru/e-finance/mobile-banking-rank-2018/>

Проект состоит из
1 монолита и
≈25 микросервисов
на *Spring Boot*



Ой, что будет...

1. Погружение
2. Особенности перехода
 - Сборка проекта
 - Обновление Spring
 - Deployment
3. Новшества платформы
 - Single-File Programs
 - Class Data Sharing
 - JShell
4. Всплытие

Чего точно не будет:



Обзора **всех**
новых **фич** Java
с 9 по 11 версии



Инструкции
по распилу
на **модули**



Агитации
переходить
немедленно

1. Погружение

2. Особенности перехода

- Сборка проекта
- Обновление Spring
- Deployment

3. Новшества платформы

- Single-File Programs
- Class Data Sharing
- JShell

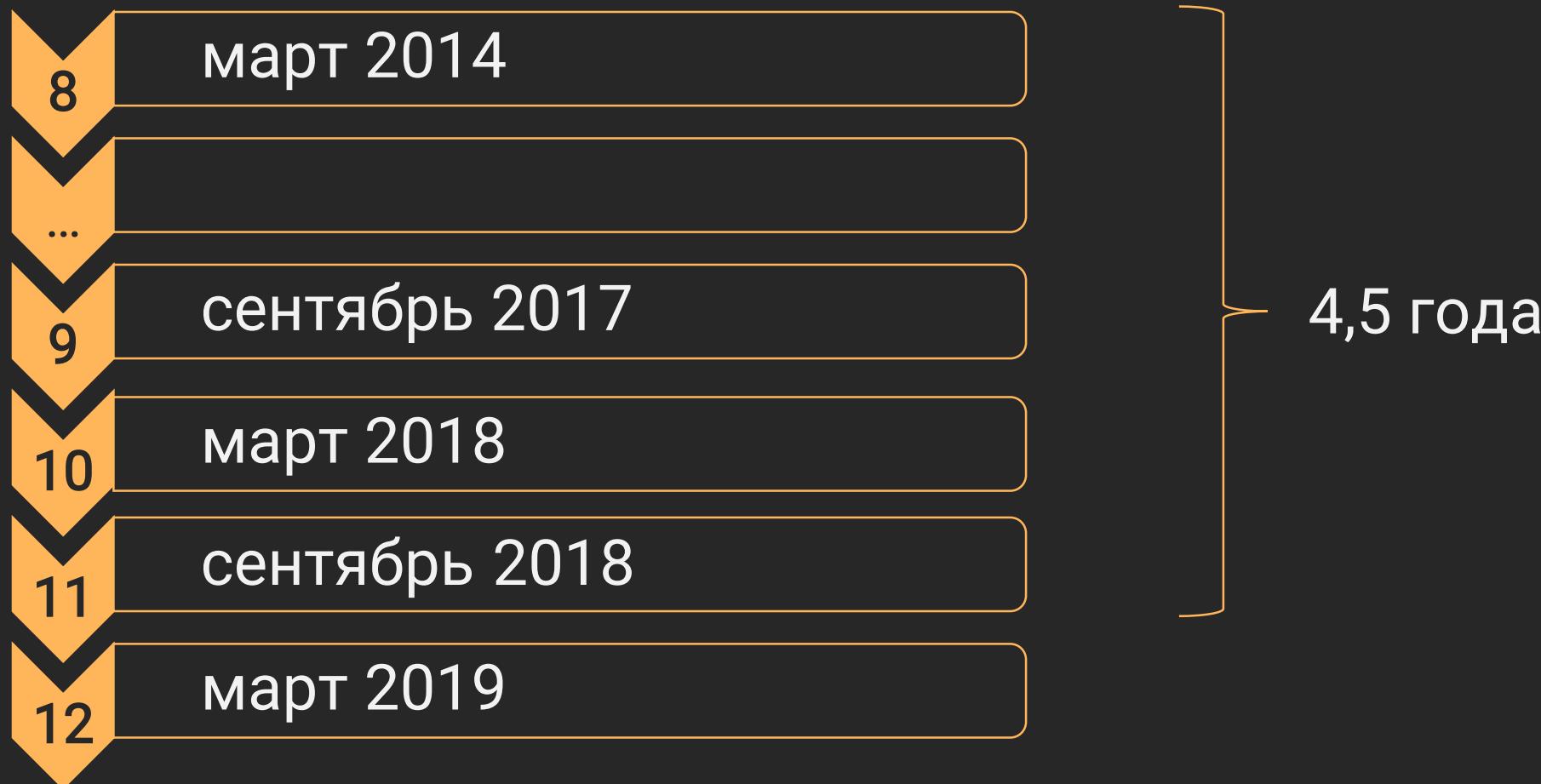
4. Всплытие



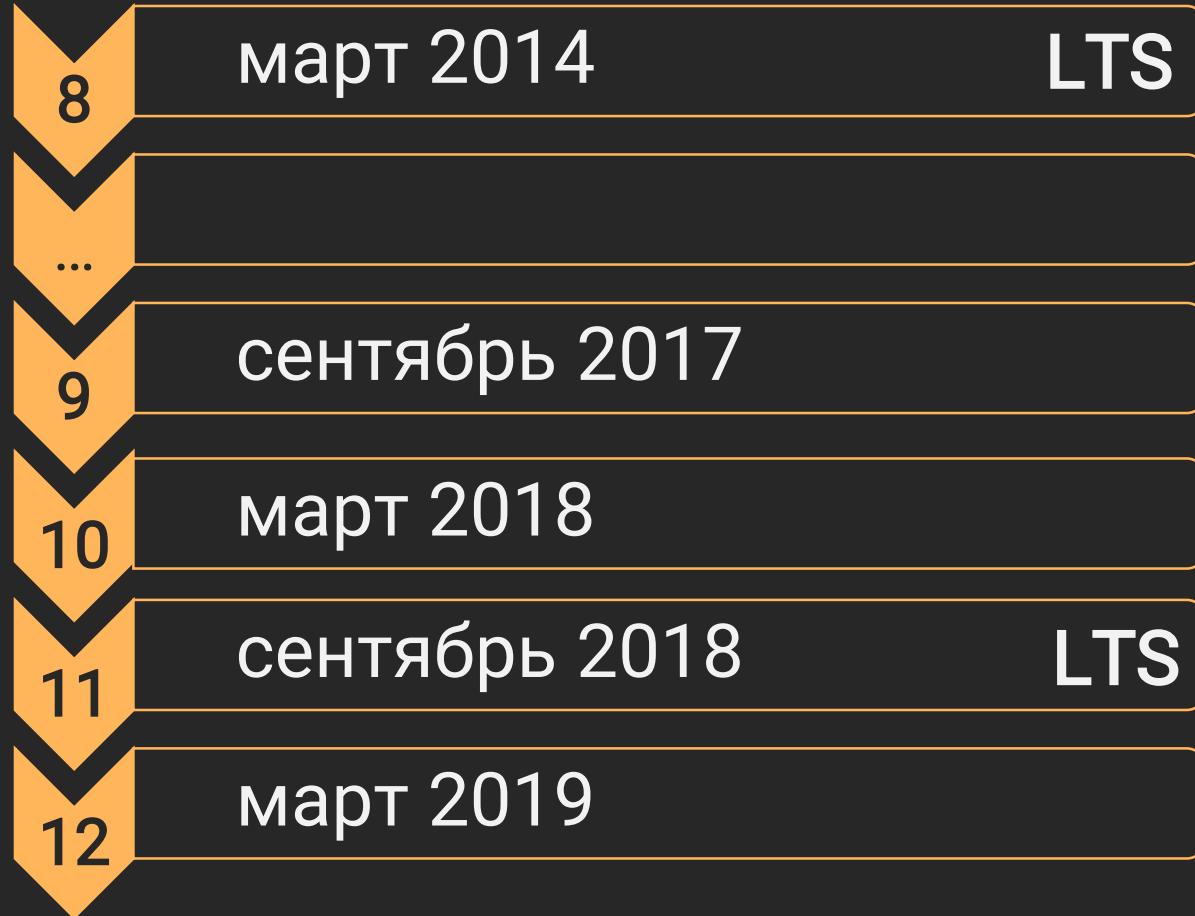
Погружение

Почему и зачем?

Цикл релизов Java

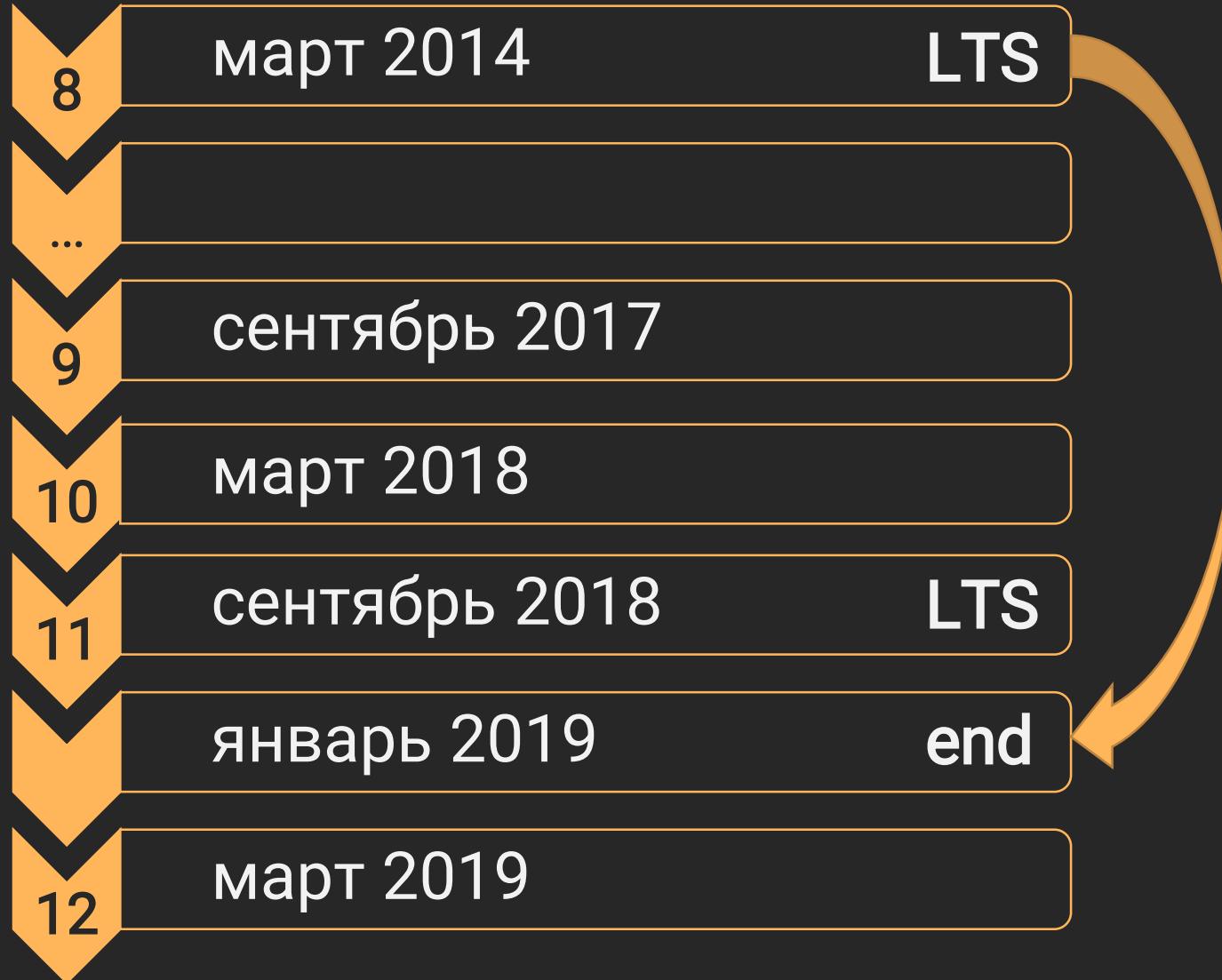


Среди релизов есть особенные



* LTS – Long-Term
Support (Oracle)

В январе'19 версия 8 резко постарела



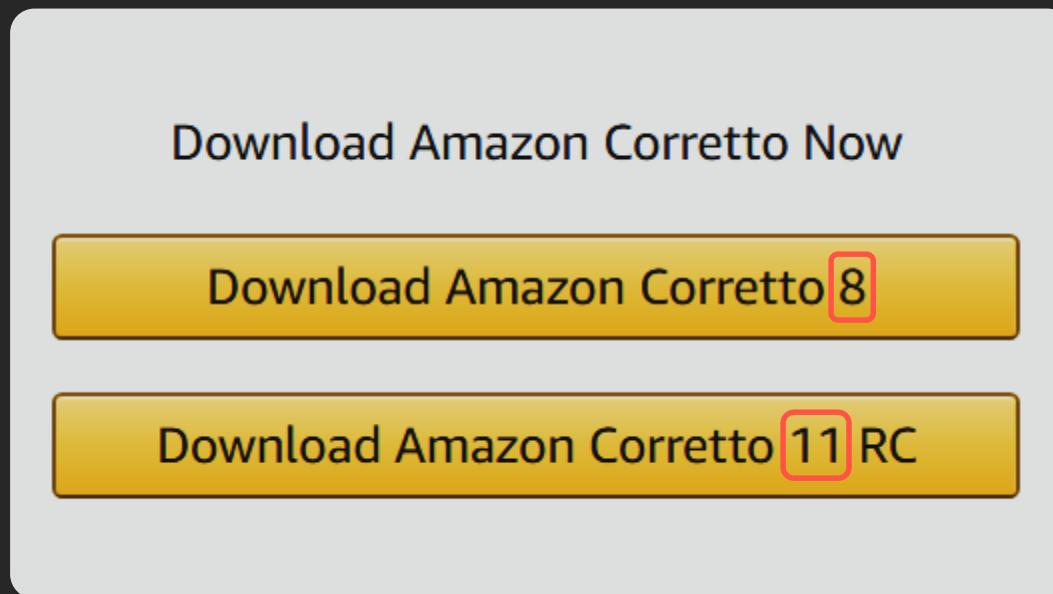
Public Updates от Oracle
прекращены 17.01.19

Условия поддержки Java 8 у других вендоров могут отличаться



Ориентация на LTS-релизы – распространенная практика

Пример с Amazon Corretto



<https://aws.amazon.com/en/corretto>

Пример с Liberica JDK



LTS versions
(long-term support)

Download **JDK 8**



Current release

Download **JDK 11**

<https://bell-sw.com/>

Пример со Spring (и его Boot)

“

Spring Framework 5.1 requires JDK 8 ...
and specifically supports JDK 11
(as the next long-term support release)

<https://github.com/spring-projects/spring-framework/wiki/Upgrading-to-Spring-Framework-5.x>

И зачем это все?



1. Погружение

2. Особенности перехода

- Сборка проекта
- Обновление Spring
- Deployment

3. Новшества платформы

- Single-File Programs
- Class Data Sharing
- JShell

4. Всплытие



Сборка проекта на Gradle

Грабли на самом раннем этапе

Для сборки на Java 11 нужен Gradle 5

The screenshot shows a GitHub issue page for the repository `gradle / gradle`. The issue is titled `Java 11 support` and has the ID `#5120`. It is marked as `Closed`. A yellow arrow points from the issue title to the `Milestone` section, which shows a green progress bar labeled `5.0 RC1`. The URL of the issue is <https://github.com/gradle/gradle/issues/5120>.

gradle / gradle

Code Issues Pull requests Wiki Insights

Java 11 support #5120

Closed

b3indpirate opened this issue on 20 Apr 2018 · 36 comments

Milestone

5.0 RC1

<https://github.com/gradle/gradle/issues/5120>

Но есть один нюанс...

```
testCompile - Dependencies for source set 'test'  
+--- project :shared  
...  
+--- ru.ftc.pc.testing:daotestkit-starter:1.0-SNAPSHOT  
|   +--- ru.ftc.pc.testing:daotestkit:0.5-SNAPSHOT  
...  
...
```

\$ gradle dependencies



snapshot



Found 2 dependencies resolved in 1 project across 9 configurations

Gradle 5.0 (build scan)

:dependencies

testCompile ✓ - 31.123s

ru.ftc.pc.testing:daotestkit:0.5-SNAPSHOT

ru.ftc.pc.testing:daotestkit-starter:1.0-SNAPSHOT

testRuntime ✓ - 24.515s

ru.ftc.pc.testing:daotestkit:0.5-SNAPSHOT

ru.ftc.pc.testing:daotestkit-starter:1.0-SNAPSHOT

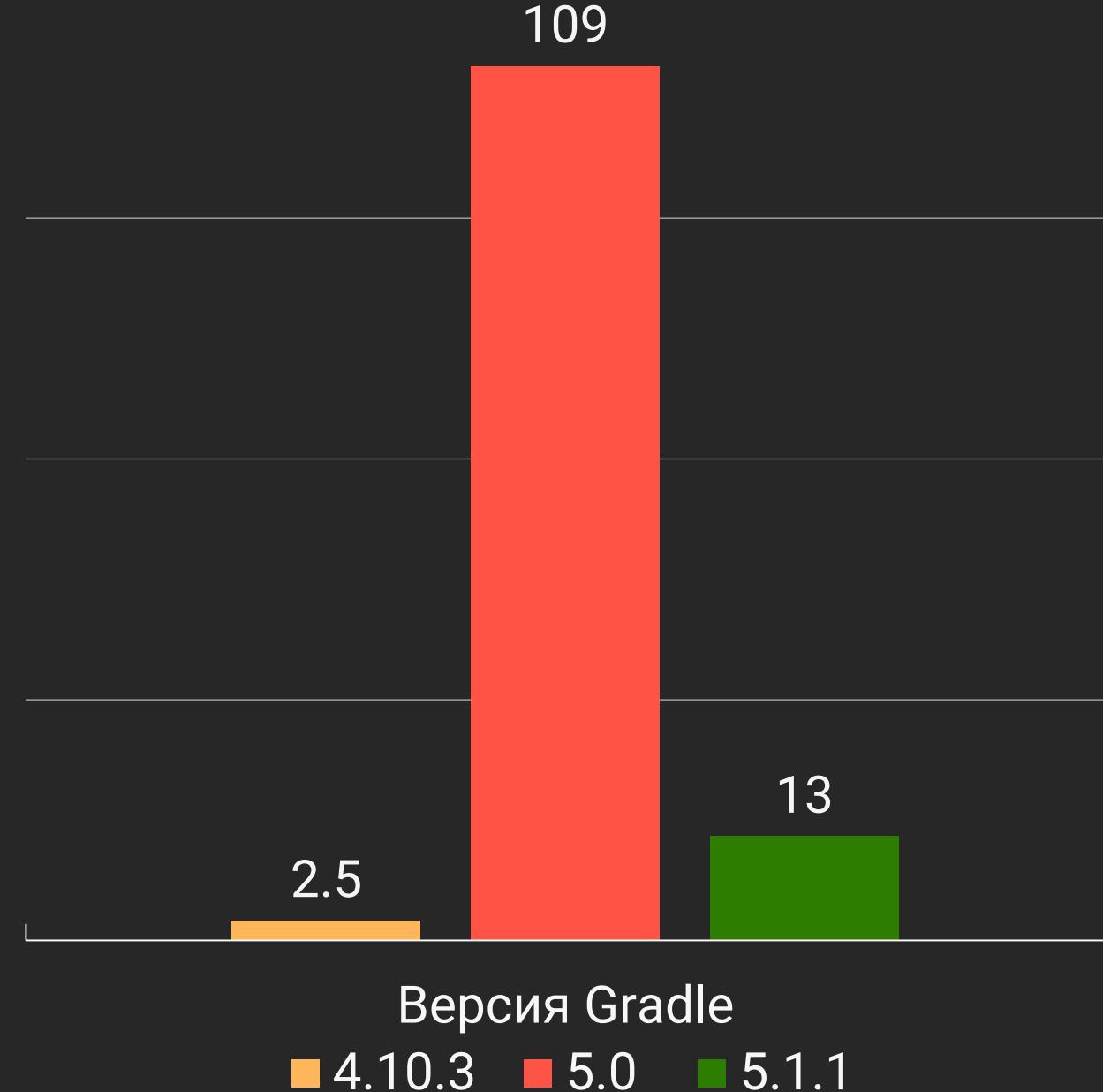
detachedConfiguration175 ✓ - 12.269s

ru.ftc.pc.testing:daotestkit:0.5-SNAPSHOT

ru.ftc.pc.testing:daotestkit-starter:1.0-SNAPSHOT

Время '\$ gradle dependencies' (сек)

Некоторые сборки
на Gradle 5.0
замедлились в 10+ раз



Gradle 5.1 позволяет связывать
репозитории с зависимостями

Repository to dependency matching

```
repositories {  
    maven {  
        url "https://repo.mycompany.com"  
        content {  
            includeGroupByRegex "com\\.mycompany.*"  
        }  
    }  
}
```

build.gradle

Подробнее:

 gradle repository to

Maven repository filtering

```
repositories {  
    maven {  
        url "http://repo.mycompany.com/releases"  
        mavenContent {  
            releasesOnly()  
        }  
    }  
    maven {  
        url "http://repo.mycompany.com/snapshots"  
        mavenContent {  
            snapshotsOnly()  
        }  
    }  
}
```

Поддержка BOM (как было)

```
plugins {  
    id 'org.springframework.boot' version '2.1.3.RELEASE'  
}  
apply plugin: 'io.spring.dependency-management'  
dependencies {  
    implementation 'org.springframework.boot'  
        + ':spring-boot-starter'  
        #+ ':2.1.3.RELEASE'  
}
```

build.gradle

Поддержка BOM (как стало)

```
dependencies {  
    implementation platform(  
        'org.springframework.boot'  
        + ':spring-boot-dependencies'  
        + ':2.1.3.RELEASE')  
  
    implementation 'org.springframework.boot'  
        + ':spring-boot-starter'  
        #+ ':2.1.3.RELEASE'
```

build.gradle

Gradle + Lombok

Подключение библиотеки после 5.x

Как было до 5.0

```
dependencies {  
    ...  
    compileOnly('org.projectlombok:lombok')  
    ...  
}
```

build.gradle

Как стало после 5.0

```
$ gradle compileJava
...
[...].java:160: error: cannot find symbol
...
100 errors
...
> Task :compileJava FAILED
```

Как должно стать после 5.0

```
dependencies {  
    ...  
    annotationProcessor('org.projectlombok:lombok')*  
    compileOnly('org.projectlombok:lombok')  
    ...  
}
```

build.gradle

* Для тестов – testAnnotationProcessor

* Можно через плагин: <https://plugins.gradle.org/plugin/io.freefair.lombok>

Совместимость с новым импортом BOM

```
implementation platform("org.springframework.boot"
    + ":spring-boot-dependencies:2.1.3.RELEASE")
annotationProcessor("org.projectlombok:lombok")
```

build.gradle

```
> Task :dependencyInsight
org.projectlombok:lombok: FAILED
Failures:
- Could not find org.projectlombok:lombok: .
org.projectlombok:lombok FAILED
\--- annotationProcessor
```

dependencyInsight --dependency lombok

И как быть?

- Способ 1 – добавить директиву:

```
implementation annotationProcessor("org.springframework.boot"  
+ ":spring-boot-dependencies:2.1.3.RELEASE")
```

build.gradle

- Способ 2 – плагин `io.spring.dependency-management`
- Способ 3 – плагин `io.freefair.lombok`

Разделение зависимостей

Compile & Runtime Scopes

Во времена Gradle 4 жил-был импорт...

```
// ...  
  
import org.apache.commons.collections.CollectionUtils;  
  
// ...
```

SomeClass.java

... а потом случился Gradle 5

```
$ gradle compileJava  
...  
SomeClass.java:24: error: package  
org.apache.commons.collections does not exist  
import org.apache.commons.collections.CollectionUtils;
```

\$ gradle compileJava

Зависимость оказалась транзитивной

```
$ gradle dependencyInsight --dependency commons-collections \
                           --configuration compile
> Task dependencyInsight

commons-collections
\--- ribbon-httpclient
    \--- spring-cloud-starter-netflix-ribbon
        \--- spring-cloud-starter-netflix-eureka-client
            \--- project :shared
                \--- compile
```

```
$ gradle dependencyInsight
```

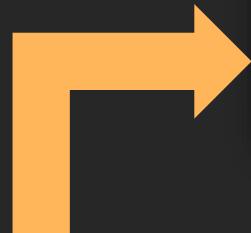
Новшество версии Gradle 5.0

“

... the compilation classpath **only** includes
compile-scoped dependencies

https://docs.gradle.org/5.0/userguide/upgrading_version_4.html

Источник runtime зависимости



commons-collections

\--- ribbon-httpclient

 \--- spring-cloud-starter-netflix-ribbon

 \--- spring-cloud-starter-netflix-eureka-client

 \--- project :shared

 \--- **compile**

[Home](#) » [com.netflix.ribbon](#) » [ribbon-httpclient](#) » 2.3.0

Ribbon HttpClient » 2.3.0

ribbon-httpclient

Runtime Dependencies (12)

Category/License

Collections

Apache 2.0

Group / Artifact



commons-collections » commons-collections

<https://mvnrepository.com/artifact/com.netflix.ribbon/ribbon-httpclient/2.3.0>

И как быть?

- Менять транзитивные зависимости на **явные**
- Анализировать зависимости (Gradle):
 - `dependencies --configuration compile`
 - `dependencyInsight --configuration compile --dependency myDep`
 - Build Scan <https://scans.gradle.com/>

Минутка справедливости: Maven

Для работы с Java 11 нужен Maven 3.5.0, а также:

- compiler plugin: 3.8.0
- surefire & failsafe: 2.22.0

Полезные материалы:

- <https://blog.codefx.org/java/java-11-migration-guide/>
- <https://winterbe.com/posts/2018/08/29/migrate-maven-projects-to-java-11-jigsaw/>

Для тех, кто любит поподробнее

The image consists of two main parts. On the left is a presentation slide with a dark blue background showing a crowd of people. The title 'Joker<?> 2018' is at the top in white, stylized text. Below it, 'Robert Scholte' is listed in large white text, followed by 'Sourcegrounds' in smaller white text. On the right is a photograph of a man with short dark hair, wearing a grey t-shirt and a lanyard, speaking into a microphone. He is positioned in front of a blurred background with text like 'er', 'so', and 'me'. The overall composition is a collage of these two images.

Apache Maven
supports ALL Java

 JUG.ru

<https://2018.jokerconf.com/2018/talks/ciamjmk7uwqwyseuy6iao>

1. Погружение

2. Особенности перехода

- Сборка проекта
- Обновление Spring
- Deployment

3. Новшества платформы

- Single-File Programs
- Class Data Sharing
- JShell

4. Всплытие



Причём тут Spring?

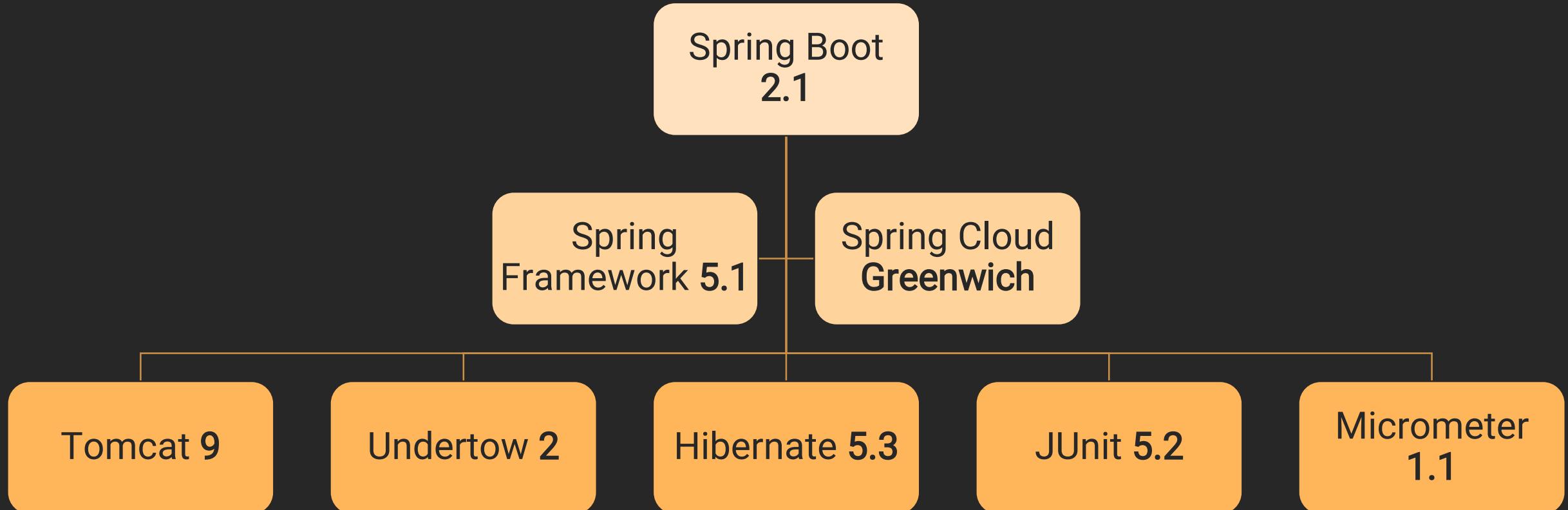
“

Java 11 is supported as of Spring Boot 2.1.0.M2*

<https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-with-Java-9-and-above>

* Но можно пробовать и на меньших версиях

Версии транзитивных зависимостей



Запрет переопределения бинов

Новшество версии 2.1

“

Bean overriding
has been **disabled** by default
to prevent a bean being
accidentally overridden

<https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-2.1-Release-Notes>

Такие изменения
бывают не только
случайными



Пример 1: библиотечный код

```
@Configuration
public class FeignClientsConfiguration {

    @Bean
    @ConditionalOnMissingBean(FeignLoggerFactory.class)
    public FeignLoggerFactory feignLoggerFactory() {
        return new DefaultFeignLoggerFactory(logger);
    }
}
```

org.springframework.cloud.openfeign.FeignClientsConfiguration

Пример 1: прикладной код

```
@Configuration  
public class DefaultFeignClientConfiguration {  
  
    @Bean  
    public FeignLoggerFactory feignLoggerFactory() {  
        return new MyFeignLoggerFactory();  
    }  
}
```

com.example.DefaultFeignClientConfiguration

Пример 1: последствия

```
BeanDefinitionOverrideException: ...
Cannot register bean definition [...]
    for bean 'feignLoggerFactory':
There is already [...] bound.
```

application log

Пример 2: прикладной код

```
@Configuration  
public class MyConfiguration {  
  
    @Bean  
    RetryListener retryListener() {  
        return new RetryListenerSupport() {  
            // implementation details  
        };  
    }  
}
```

MyConfiguration.java

Пример 2: тестовый код

```
@SpringBootTest
class MyTest {
    // (test methods)

    @Configuration
    static class TestConfiguration {
        @Bean
        RetryListener retryListener() {
            return new TestRetryListener();
        }
    }
}
```

MyTest.java

Пример 2: последствия

```
BeanDefinitionOverrideException: ...
Cannot register bean definition [...]
    for bean 'retryListener':
There is already [...] bound.
```

application log

Решение (вариант 1): глобально

```
spring.main.allow-bean-definition-overriding=true
```

application.properties

- + Быстро, просто, понятно
- + Обратно совместимо
- + Масштабно
- Не безопасно
- Не гибко

Решение (вариант 2): локально

```
@SpringBootTest
class MyTest {
    // (test methods)

    @Configuration
    static class TestConfiguration {
        @Bean
        @Primary
        RetryListener customRetryListener() {
            return new TestRetryListener();
        }
    }
}
```

MyTest.java

Пояснения к решению 2

- Префикс `custom` (или иной) – для различия имен
 - Аннотация `@Primary` – для перекрытия исходного бина
-
- | | |
|-----------------|--------------------|
| + Безопасно | - Не масштабируемо |
| + Контролируемо | - [Трудоемко] |
| + Гибко | |

Оптимизация производительности

Есть хорошая новость

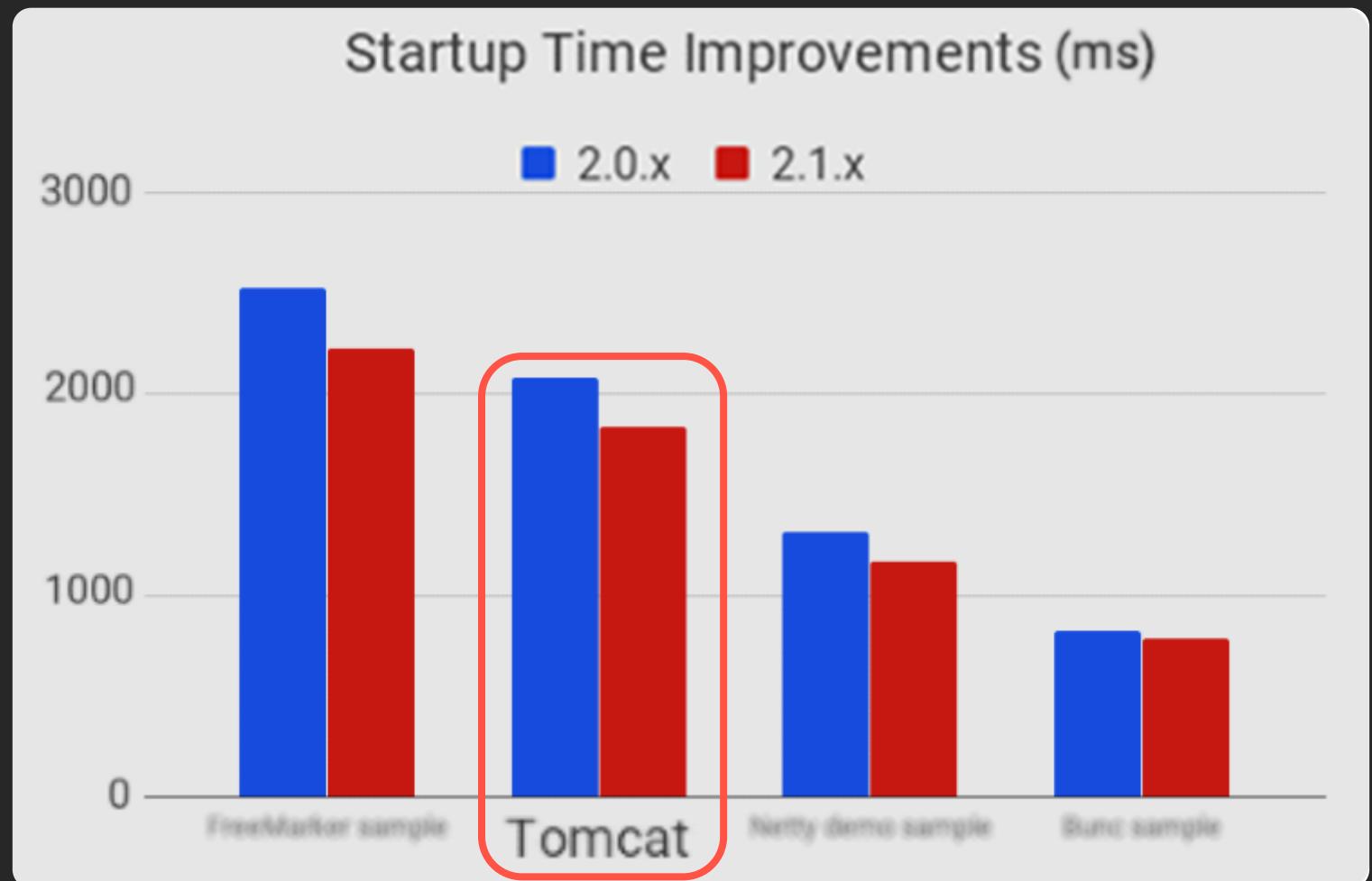
“

... Spring Boot 2.1 and Spring 5.1
have some **quite nice** optimizations
for startup time and heap usage

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

По данным испытаний от разработчиков

Время запуска
на Tomcat
сократилось
на $\approx 14\%$



<https://github.com/dsyer/spring-boot-allocations>

По нашему опыту

Время старта микросервисов, сек



Как еще можно ускорить запуск?

Источники вдохновения от Dave Syer (@david_syer):

- <https://spring.io/blog/2018/12/12/how-fast-is-spring>
- <https://www.youtube.com/watch?v=97UTDmonq7w>
- <https://github.com/dsyer/spring-boot-allocations>

Кардинальный метод ускорения

The slide is a promotional graphic for JPoint 2019. It features a dark blue background with a light blue gradient overlay. In the top left corner is the JPoint 2019 logo, which includes a stylized coffee cup icon and the text "JPoint 2019". Below the logo, the speaker's name "Никита Липский" and his affiliation "Excelsior" are displayed. A large, semi-transparent white box contains a quote in Russian: "Нас Spring Boot, а мы крепчаем: невыносимая легкость АОТ-компиляции Spring-приложений". To the right of the text box is a portrait photograph of Nikita Lypskiy, a man with short brown hair and blue eyes, wearing a black t-shirt. The overall design is modern and professional.

<https://jpoint.ru/talks/1emn9byzaklzozk6ec56dp>

Опробованные способы (1/2)

- Fix the location of the Spring Boot config file(s)
Удобно (кое-где), но не очень эффективно
- Unpack the fat jar and run with an explicit classpath
Перспективно
- Run the JVM with –noverify
Сомнительно, см. <https://www.youtube.com/watch?v=-OocG7tFIOQ>

Опробованные способы (2/2)

- Use Class Data Sharing (CDS)
Не просто, но полезно (см. далее)
- «Let's make SpringBoot app start faster»
<https://dev.to/bufferings/lets-make-springboot-app-start-faster-k9m>
- Use Spring Boot 2.1 and Spring 5.1
Бесплатно, без регистрации, без СМС*

*)

“

... several introspection algorithms
have been streamlined ...
potentially causing side effects

<https://github.com/spring-projects/spring-framework/wiki/Upgrading-to-Spring-Framework-5.x>

Lombok

Причем здесь Lombok?

“

Spring Boot 2.1 has upgraded to Lombok 1.18.x
[which] will no longer generate
a private, no-args constructor by default

<https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-2.1-Release-Notes>

Причина изменения

<https://github.com/rzwitserloot/lombok/issues/1708>

The screenshot shows a GitHub issue page for the repository `rzwitserloot/lombok`. The page title is "extraPrivate=true from 1.16.22 breaks Jackson creator detection" and the issue number is [#1708](#). The status is "Closed" by `bcalmac` on 30 May 2018, with 21 comments. A comment from `rspilker` on 5 Jun 2018 discusses addressing feedback about projects breaking due to the update.

extraPrivate=true from 1.16.22 breaks Jackson creator detection [#1708](#)

Closed bcalmac opened this issue on 30 May 2018 · 21 comments

A specific comment from `rspilker` is highlighted, stating: "We got a lot of feedback that that version broke a lot of projects, for instance [#1563](#). So we looked for an alternative."

Collaborator + 😊 ...

We got a lot of feedback that that version broke a lot of projects, for instance [#1563](#). So we looked for an alternative.

Пример из реальной жизни

```
@Getter  
@ToString  
@RequiredArgsConstructor  
class SendInfoCommand {  
    private final String first;  
    private final String second;  
}
```

SendInfoCommand.java

```
@RabbitListener(queues = QUEUE_NAME)  
public void receive(@Payload SendInfoCommand sendInfoCommand) {  
    // listener implementation  
}
```

CommandListener.java

И это работает!
(по крайней мере,
до 1.16.20)



Запуск на Lombok 1.16.20

Caused by:

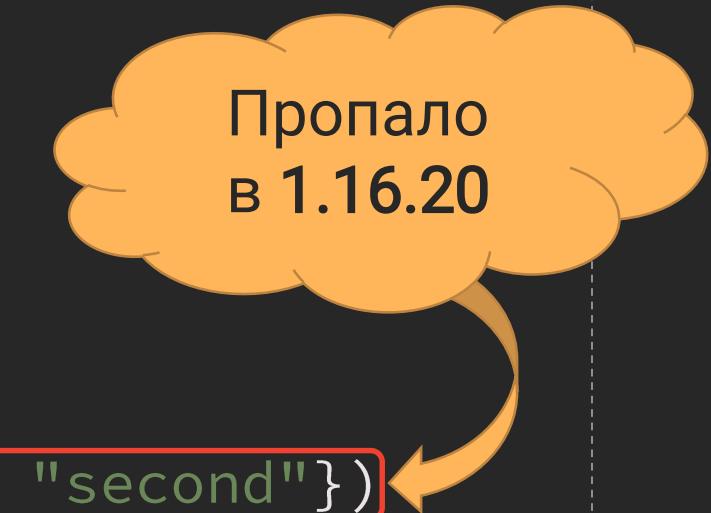
com.fasterxml.jackson.databind.exc.InvalidDefinitionException:
Cannot construct instance of `SendInfoCommand`
(no Creators, like default construct, exist):

...

application log

DeLombok

```
@Getter  
@ToString  
@RequiredArgsConstructor  
class SendInfoCommand {  
    private final String first;  
    private final String second;  
  
    @java.beans.ConstructorProperties({"first", "second"})  
    public SendInfoCommand(String first, String second) {  
        this.first = first;  
        this.second = second;  
    }  
}
```



SendInfoCommand.java

Lombok Changelog (v1.16.20)

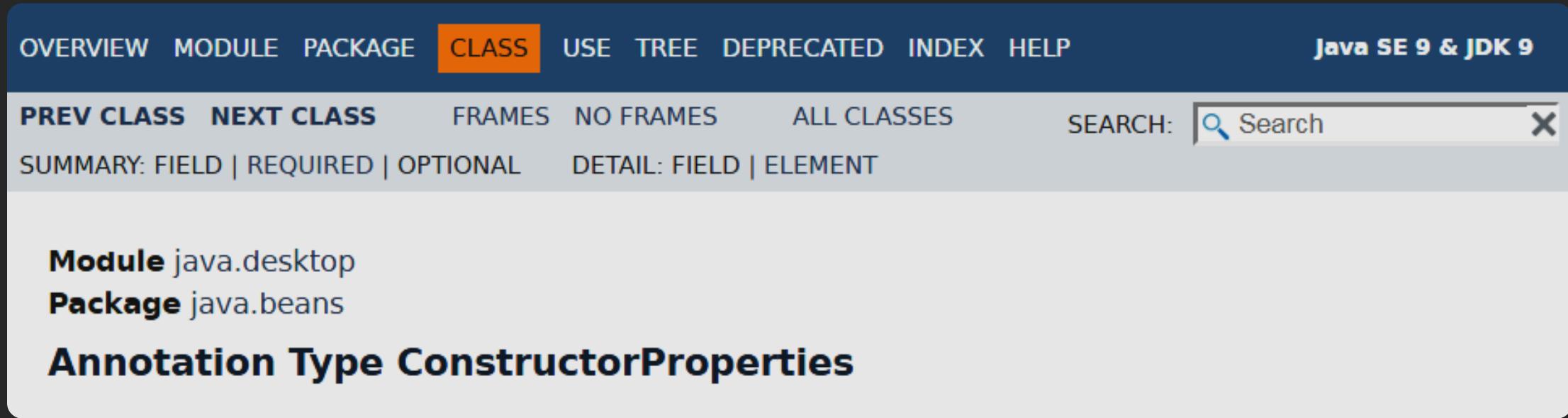
“

BREAKING CHANGE: by default
lombok no longer ... generates
@ConstructorProperties...

Oracle ... broke this annotation
with the release of JDK9

<https://projectlombok.org/changelog>

Причем тут JDK9?



The screenshot shows a screenshot of the Java SE 9 & JDK 9 API documentation. The top navigation bar includes links for OVERVIEW, MODULE, PACKAGE, CLASS (which is highlighted in orange), USE, TREE, DEPRECATED, INDEX, and HELP. To the right of the navigation bar is the text "Java SE 9 & JDK 9". Below the navigation bar are links for PREV CLASS and NEXT CLASS, and buttons for FRAMES, NO FRAMES, and ALL CLASSES. A search bar with the placeholder "Search" and a clear button "X" is also present. Underneath the search bar are links for SUMMARY: FIELD | REQUIRED | OPTIONAL and DETAIL: FIELD | ELEMENT. The main content area displays the following navigation:

- Module** java.desktop
- Package** java.beans
- Annotation Type ConstructorProperties**

<https://docs.oracle.com/javase/9/docs/api/java/beans/ConstructorProperties.html>

Причем тут JDK9?

The screenshot shows a JavaDoc interface. At the top, there's a navigation bar with tabs: ОБЩЕЕ, ИНСТРУМЕНТЫ, ПОДСКАЗКИ, and a highlighted tab labeled CLASS. To the right of the tabs are links for USE, TABLE, EXPANDED, INDEX, and HELP. On the far right of the bar, the text "Java SE 9 & JDK 9" is displayed in a red-bordered box. Below the bar, there are several search and filter buttons, including "CLASS", "METHOD", "FIELD", "ENUM", "INTERFACE", "TYPE", and "INTERNAL". A search input field with the placeholder "Search..." is also present. In the main content area, a red box highlights the text "Module java.desktop". Below it, the section title "Annotation Type ConstructorProperties" is shown in bold black font. A blue link at the bottom provides the URL: <https://docs.oracle.com/javase/9/docs/api/java/beans/ConstructorProperties.html>.

Зачем добавили extraPrivate=true?

“

The reason we added this feature was that since java9 **@ConstructorProperties** moved to the module **java.desktop**, which is not added by default.

<https://github.com/rzwitserloot/lombok/issues/1708>

Варианты решения

1. Вернуть генерацию аннотации вручную:

```
lombok.anyConstructor.addConstructorProperties=true
```

lombok.config

- Требует зависимости от `java.desktop` на Jigsaw

2. Использовать аннотацию `@Setter`

- Делает класс изменяемым (mutable)
- Может не помочь на 1.18, потому что...

Lombok Changelog (v1.18.0)

“

BREAKING CHANGE: configuration key
`lombok.noArgsConstructor.extraPrivate`
is now `false` by default

<https://projectlombok.org/changelog>

Варианты решения II

1. Вернуть генерацию конструктора вручную:

```
lombok.noArgsConstructor.extraPrivate=true
```

`lombok.config`

2. Использовать обходные пути:

➤ <https://github.com/rzwitserloot/lombok/issues/1708>

Краткая хроника событий

1.16.20: убрали `@ConstructorProperties`

1.16.22: добавили `extraPrivate=true`

1.18.0: переключили `extraPrivate=false`

FAIL

Не все то золото,
что Lombok



Что можно сделать?

- Завести в проекте файл `lombok.config`
- Договориться с командой о порядке применения lombok

Пример файла lombok.config

```
# Stops further searching for lombok.config
config.stopBubbling = true

# Stable features
lombok.val.flagUsage = error
# ...

# Experimental features
lombok.extensionMethod.flagUsage = error
# ...

# Our discussed settings
lombok.equalsAndHashCode.doNotUseGetters = true
lombok.anyConstructor.addConstructorProperties = true
```

Что можно сделать?

- Завести в проекте файл `lombok.config`
- Договориться с командой о порядке применения lombok
- Не полагаться на версию Lombok из Spring Boot ВОМ
- Не забывать о `delombok`
-  ;)

Spring Cloud Greenwich



Проекты Spring Cloud Netflix
переведены в **режим поддержки**

Что значит «Maintenance Mode»?



Без новых фич,
только правка
багов и угроз



С 23.01.19
минимум
на 1 год



Не касается
Eureka и модулей
concurrency-limits

- <https://spring.io/blog/2018/12/12/spring-cloud-greenwich-rc1-available-now>

К счастью, есть альтернативы

Текущий вариант	Альтернатива
Hystrix	Resilience4j
Hystrix Dashboard / Turbine	Micrometer + Monitoring System
Ribbon	Spring Cloud Loadbalancer
Zuul 1	Spring Cloud Gateway
Archaius 1	Spring Boot external config + Spring Cloud Config

Spring Cloud Greenwich
зависит от OpenFeign 10.1.0

Изменение в API OpenFeign: ответы

```
Response response =  
    Response.builder()  
        // diff padding  
        .status(STATUS)  
        .reason(REASON)  
        .headers(emptyMap())  
    .build();
```

v9.x

```
Response response =  
    Response.builder()  
        .request(REQUEST)  
        .status(STATUS)  
        .reason(REASON)  
        .headers(Map.of())  
    .build();
```

v10.x

Изменение в API OpenFeign: запросы

```
Request request =  
    Request.create(  
        "GET",  
        "http://somehost.com",  
        Collections.emptyMap(),  
        new byte[] {},  
        Charset.defaultCharset());
```

v9.x

```
Request request =  
    Request.create(  
        Request.HttpMethod.GET,  
        "http://somehost.com",  
        Collections.emptyMap(),  
        Request.Body.empty());
```

v10.x

В Greenwich появилась поддержка @QueryMap

```
@FeignClient("authClient")
public class AuthTemplate {
    @GetMapping(path = "/auth")
    String authEndpoint(@SpringQueryMap Credentials creds);
}
```

POJO с полями
login и password

AuthTemplate.java

```
GET/auth?login={login}&password={password}
```

получившийся URI

Подробнее:

 springcloud feign querymap

OpenFeign поддерживает Java HTTP Client

- Обеспечивает работу по **HTTP/2**
- Требует запуска на **Java 11+**

```
GitHub github = Feign.builder()  
    .client(new Http2Client())  
    .target(GitHub.class, "https://api.github.com");
```

<https://github.com/OpenFeign/feign#java-11-http2>

[Standard] HTTP Client (JEP-321)

```
HttpClient client = HttpClient.newBuilder()
    .version(HTTP_1_1)
    .followRedirects(Redirect.NORMAL)
    .connectTimeout(Duration.ofSeconds(20))
    .proxy(ProxySelector.of(new InetSocketAddress(/* */)))
    .authenticator(Authenticator.getDefault())
    .build();

HttpResponse<String> response = client.send(request,
    BodyHandlers.ofString());

System.out.println(response.statusCode());
System.out.println(response.body());
```

java.net.http.HttpClient javadoc

А если хочется узнать больше

A promotional slide for a Java conference. The background is a gradient from dark blue to light blue. In the top left, there's a white coffee cup icon next to the text "JPoint 2019". Below that, "Chris Hegarty" is written in large white font, with "Oracle" in smaller white font underneath. In the bottom left, there's a white rectangular box containing the text "Java 11 reactive HTTP Client". To the right of this text is a portrait photo of a man with short brown hair and glasses, wearing a dark t-shirt. He is looking slightly to his left with a neutral expression. The overall design is clean and professional.

<https://jpoint.ru/talks/35watcftbrvf8a6xrqeykj>

1. Погружение

2. Особенности перехода

- Сборка проекта
- Обновление Spring
- Deployment

3. Новшества платформы

- Single-File Programs
- Class Data Sharing
- JShell

4. Всплытие



Базовый образ JDK

Для запуска в Docker

В чем вопрос?

- При Java 8 был популярен образ

openjdk:8-jre-alpine*

- Для Java 11 такого образа нет

- ✓ Проверенный
- ✓ Обновляемый
- ✓ Легкий (80 МБ)

* Alpine – компактный дистрибутив Linux

В Java 11 больше нет JRE.
Но можно собрать из модулей.

И как это сделать?

The slide features a dark blue background with a red, abstract, vine-like pattern. At the top left, the text "Joker<?> 2018" is displayed in white. Below it, the name "Юрий Артамонов" is written in large, bold, white capital letters, followed by "CUBA.platform" in a smaller white font. The main content area is a white rectangle containing the text "Jlink и Custom Runtime Image – мастерская Франкенштейна" in black. To the right of this text is a portrait of a smiling man with short brown hair, wearing a white t-shirt with "FREE" and a crown graphic, and a blue zip-up hoodie. The bottom right corner of the slide contains the JUG.ru logo.

Jlink и Custom Runtime Image –
мастерская Франкенштейна



<https://2018.jokerconf.com/2018/talks/4sanwhahpe8kgagmceqsyk>

Официальный образ
openjdk:11-jre-slim
весит ≈270 МБ



Некоторые способы остаться на Alpine

- Использовать AdoptOpenJDK
<https://github.com/AdoptOpenJDK/openjdk-docker> ≈240 MB
- Перейти на Azul Zulu OpenJDK
или на Bellsoft Liberica OpenJDK ≈200 MB
- Собрать самому
<https://stackoverflow.com/a/53669152/3507435> ≈75 MB
- Дождаться реализации проекта Portola
<https://openjdk.java.net/projects/portola/> ? MB

Образы AdoptOpenJDK

- Пример тега
 - adoptopenjdk/openjdk11:jdk-11.0.2.9-alpine-slim
- Включают опцию `-XX:+UseContainerSupport`
- Поставляются с HotSpot и OpenJ9

“

The Alpine Linux and the slim images
are not yet TCK certified

<https://github.com/AdoptOpenJDK/openjdk-docker>

Подробнее о других вариантах



The slide features a blue gradient background with white geometric shapes. At the top left is the JPoint 2019 logo, which includes a stylized coffee cup icon. Below the logo, the speaker's name, "Дмитрий Чуйко", is displayed in large white text, with "BellSoft" in smaller text underneath. A subtitle in a white box reads "Не клади все яйца в один контейнер". On the right side of the slide is a portrait photo of Dmitry Chyiko, a man with short brown hair wearing a dark suit and a microphone headset.

JPoint 2019

Дмитрий Чуйко
BellSoft

Не клади все яйца
в один контейнер



<https://jpoint.ru/talks/35hks9g6etavlgijj8nfvh>

Оптимизация логирования

Контекст задачи

- Spring Boot по умолчанию использует Logback
- Logback отправляет логи в ELK* асинхронно
- За отправку отвечает Logback Logstash Appender
- Он поддерживает несколько стратегий ожидания записей

* ELK – Elastic[Search]
& Logstash & Kibana

Обычно подходит стратегия sleeping

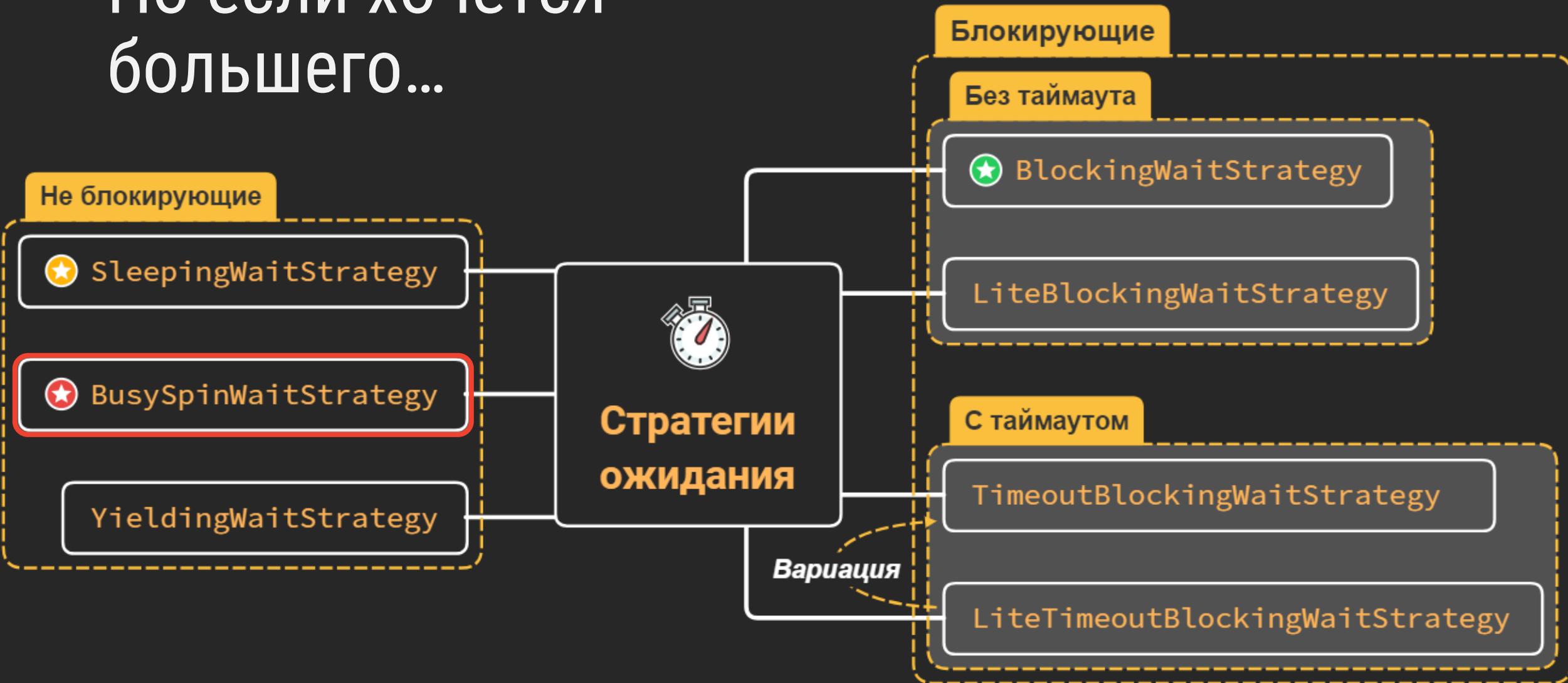
“

... it attempts to be conservative
with CPU usage ...

A common use case is
for **asynchronous logging**.

<https://github.com/LMAX-Exchange/disruptor/wiki/Getting-Started>

Но если хочется большего...



В Java 9 добавлен метод
Thread#onSpinWait

Пример применения

```
class EventHandler {  
    volatile boolean eventNotificationNotReceived;  
  
    void waitForEventAndHandleIt() {  
        while (eventNotificationNotReceived) {  
            java.lang.Thread.onSpinWait();  
        }  
        readAndProcessEvent();  
    }  
    // other methods
```

Thread#onSpinWait javadoc

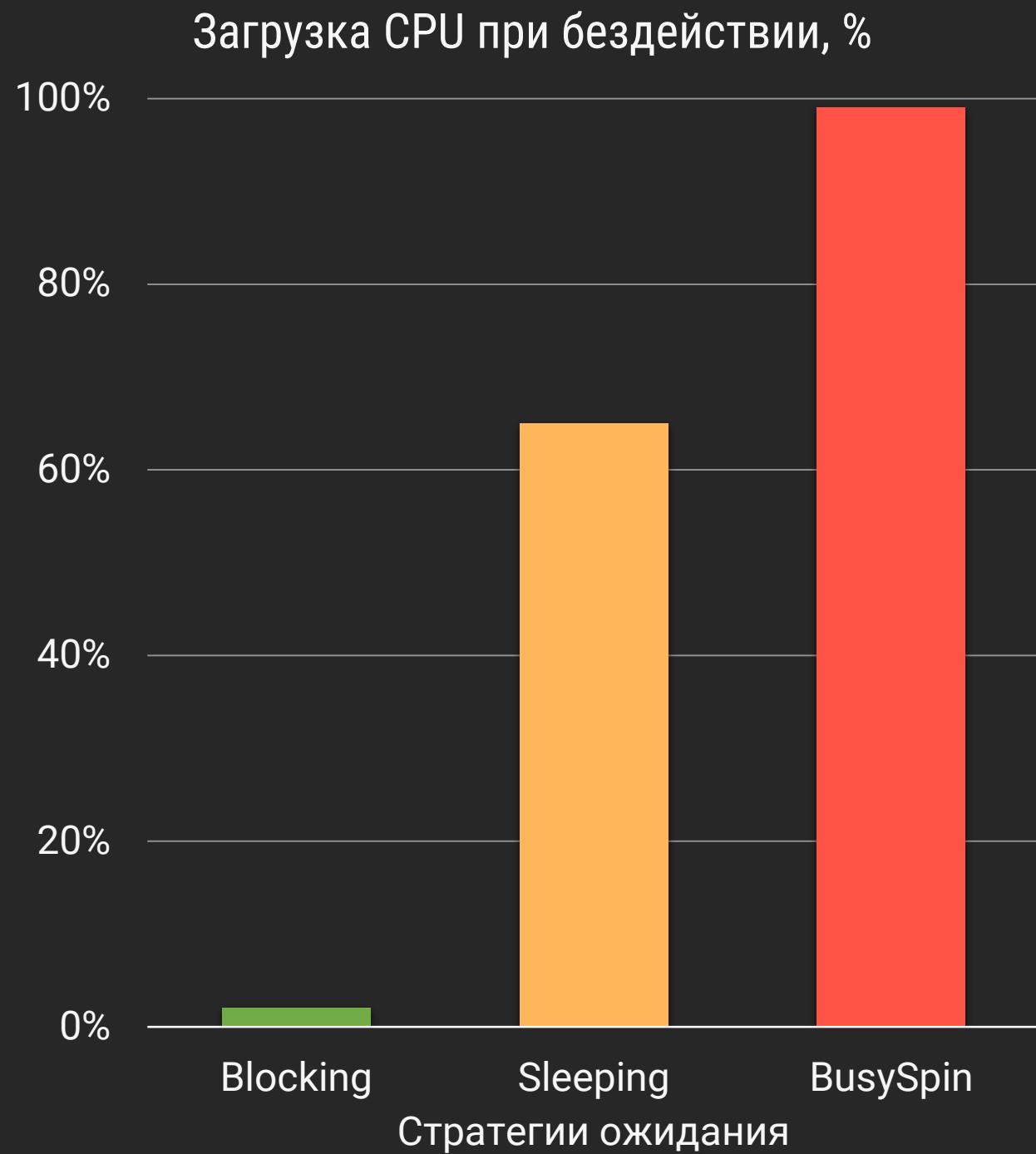
- Реализован в HotSpot для x86 (инструкцией PAUSE)

Подключение стратегии к Logback

```
<appender name="LOGSTASH" class="...LogstashTcpSocketAppender">
    <keepAliveDuration>5 minutes</keepAliveDuration>
    <reconnectionDelay>10 second</reconnectionDelay>
    <waitStrategyType>busySpin</waitStrategyType>
    <destination>elk.example.com</destination>
    <encoder class="...LogstashEncoder"/>
</appender>
```

logback.xml

Не все стратегии
одинаково экономны
с CPU



Когда пора менять стратегию?

WARN Dropped 54 events (and counting...) due to ring buffer
at max capacity 8192

WARN Dropped 785 total events due to ring buffer
at max capacity 8192

application log

Размер буфера
по умолчанию

- <https://github.com/logstash/logstash-logback-encoder/tree/logstash-logback-encoder-5.2#async-appenders>

1. Погружение

2. Особенности перехода

- Сборка проекта
- Обновление Spring
- Deployment

3. Новшества платформы

- Single-File Programs
- Class Data Sharing
- JShell

4. Всплытие



Single-File Source-Code Programs

~~Java Script~~
A Script in Java

Новый режим запуска программ в Java 11

“

As of JDK 10, the java launcher
operates in three modes ...

Here we add a **new, fourth mode**:
launching a class declared in a **source file**

<https://openjdk.java.net/jeps/330>

Single-File Source-Code Java Programs

Java 11 умеет вот так:

```
$ java SomeClassSource.java
```

- Под капотом используется javac
 - Класс компилируется в память (не на диск)
 - Custom class loader → application class loader *
- * Классы библиотек не видят стартовый класс

Shebang Files

А еще Java 11 умеет вот так:

```
$ ./SomeClassSource
```

```
#!/usr/bin/java --source 11
```

```
class Hello {  
    public static void main(String[] args) {  
        System.out.println("Hello, " + args[0]);  
    }  
}
```

SomeClassSource

Особенности Shebang Files

- Файл должен быть исполняемым (`chmod +x`)
- Номера строк в ошибках остаются корректными
- Имя скрипта не обязано совпадать с именем класса



– Вот же круто!..
А зачем?

Helm – пакетный менеджер для Kubernetes

- Может расширяться при помощи плагинов
- Плагин – любой исполняемый файл
- Традиционно плагины писались на Go, Shell, Python, Groovy, ...
- Теперь можно писать еще и на Java



Преимущества плагина на Java



≈100%
переносимость



Не нужен отдельный
runtime



Проще поддержка
(меньше языков)

Пример подключения Java-плагина к Helm

```
name: "generator-plugin"
version: "0.1.0"
#...
#command: "$HELM_PLUGIN_DIR/shell-plugin.sh"
command: "java $HELM_PLUGIN_DIR/GeneratorPlugin.java"
```

plugin.yaml

Файл `GeneratorPlugin.java` – обычный Java класс с методом `main()` и без shebang line.

Поправки на реальность

- Нет поддержки shebang line в IDE
<https://youtrack.jetbrains.com/issue/IDEA-205455>
- Нельзя указать кодировку скрипта
- Запуску подлежит только один файл

1. Погружение

2. Особенности перехода

- Сборка проекта
- Обновление Spring
- Deployment

3. Новшества платформы

- Single-File Programs
- Class Data Sharing
- JShell

4. Всплытие



Суть [App]CDS в одном слайде

- Разные экземпляры JVM загружают одни и те же классы JDK
 - На это тратится время CPU и пространство RAM
 - Идея 1: загрузить эти классы в архив и расшарить его между экземплярами JVM
 - Идея 2: сделать то же самое для классов приложения и сторонних библиотек
- 

Актуальная* инструкция по применению CDS

1. Составить** список разделяемых классов
2. Упаковать все*** классы списка в архив
3. Запустить**** приложение с использованием архива

* но это не точно

** но это не обязательно

*** но можно и не все

**** но может не получиться

Условия эксперимента

- **Задача:** разместить максимум классов в **shared-архиве**
- **Объект:** **микросервис** на Spring Boot & Cloud (Zuul)
- **Режим:** только **запуск** (локально, JDK 11)
- **Инструменты:** VisualVM, time, rmap

AppCDS + Spring Boot: в чем сложность?

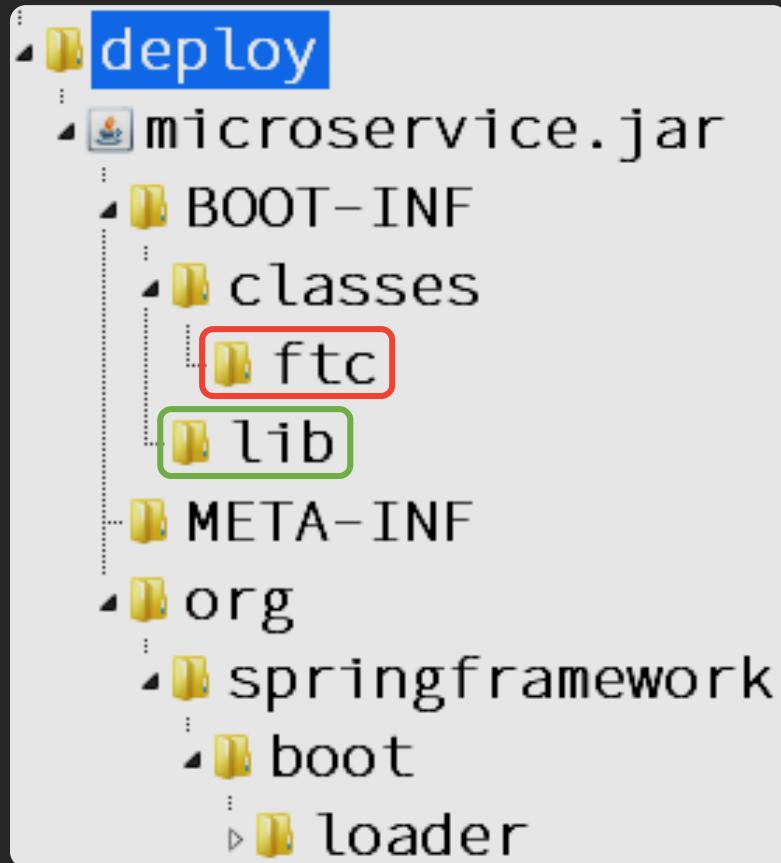
“

CDS/AppCDS supports archiving
classes from JAR* files only

<https://docs.oracle.com/en/java/javase/11/tools/java.html>

* Но не “über” JAR!

AppCDS: повод похудеть для “fat JAR”



В Ж У Х !



- * В файле META-INF/MANIFEST.MF:
 - Main-Class – прикладной класс
 - Class-Path – содержимое lib

Создание дампа классов для AppCDS

```
$ java -Xshare:off -XX:DumpLoadedClassList=classes.list \  
-jar microservice.jar
```

```
$ java -Xshare:dump -XX:SharedClassListFile=classes.list \  
-XX:SharedArchiveFile=classes.jsa \  
-jar microservice.jar
```

```
$ java -Xshare:on -XX:SharedArchiveFile=classes.jsa \  
-Xlog:class+load=info:file=class-load.log  
-jar microservice.jar
```

shell

Анализируем class-load.log

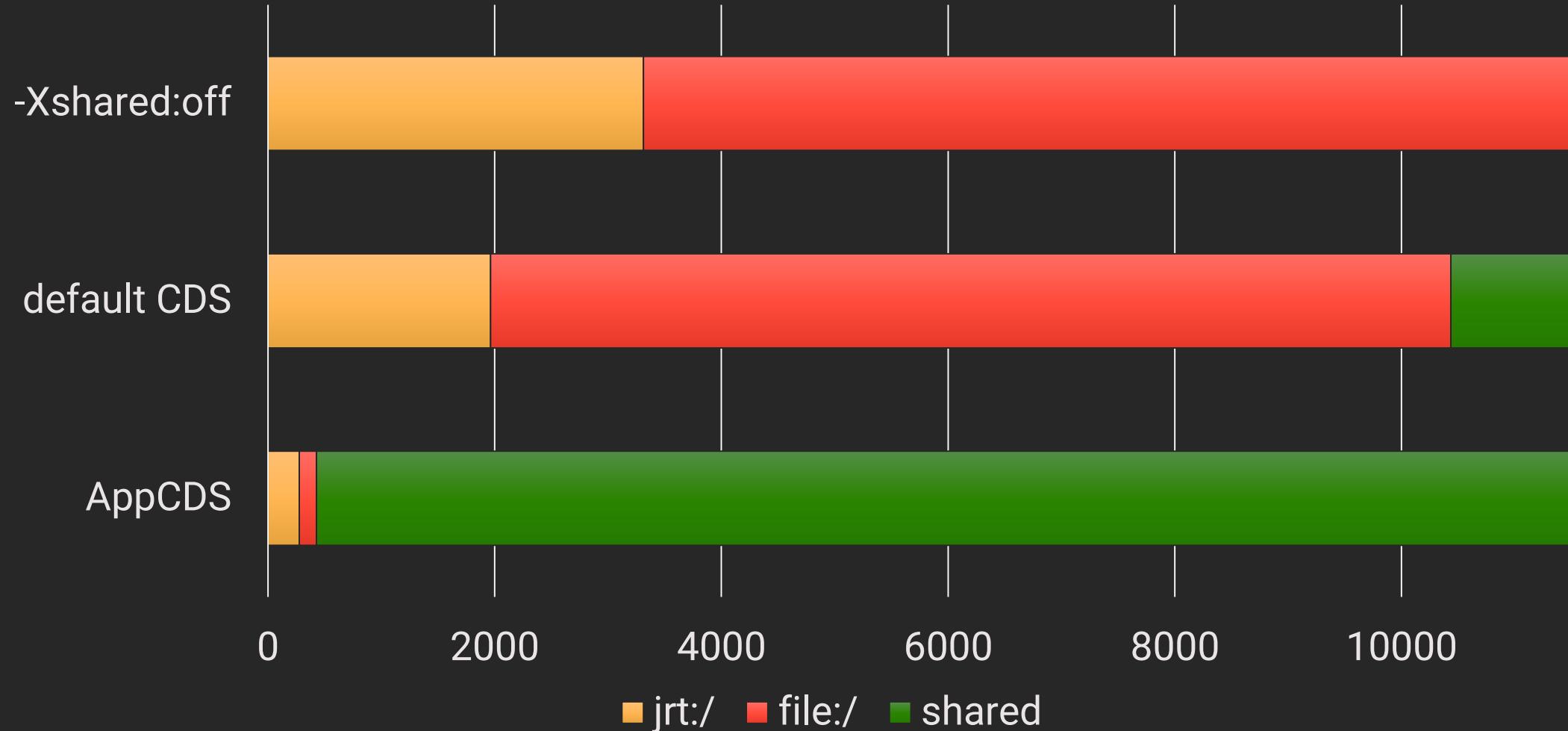
```
[info] [class,load] java.util.Map source: shared objects file  
# ...  
[info] [class,load] java.io.Console source: jrt:/java.base  
# ...  
[info] [class,load] ftc...AppClass source: file:/...  
# ...
```

class-load.log

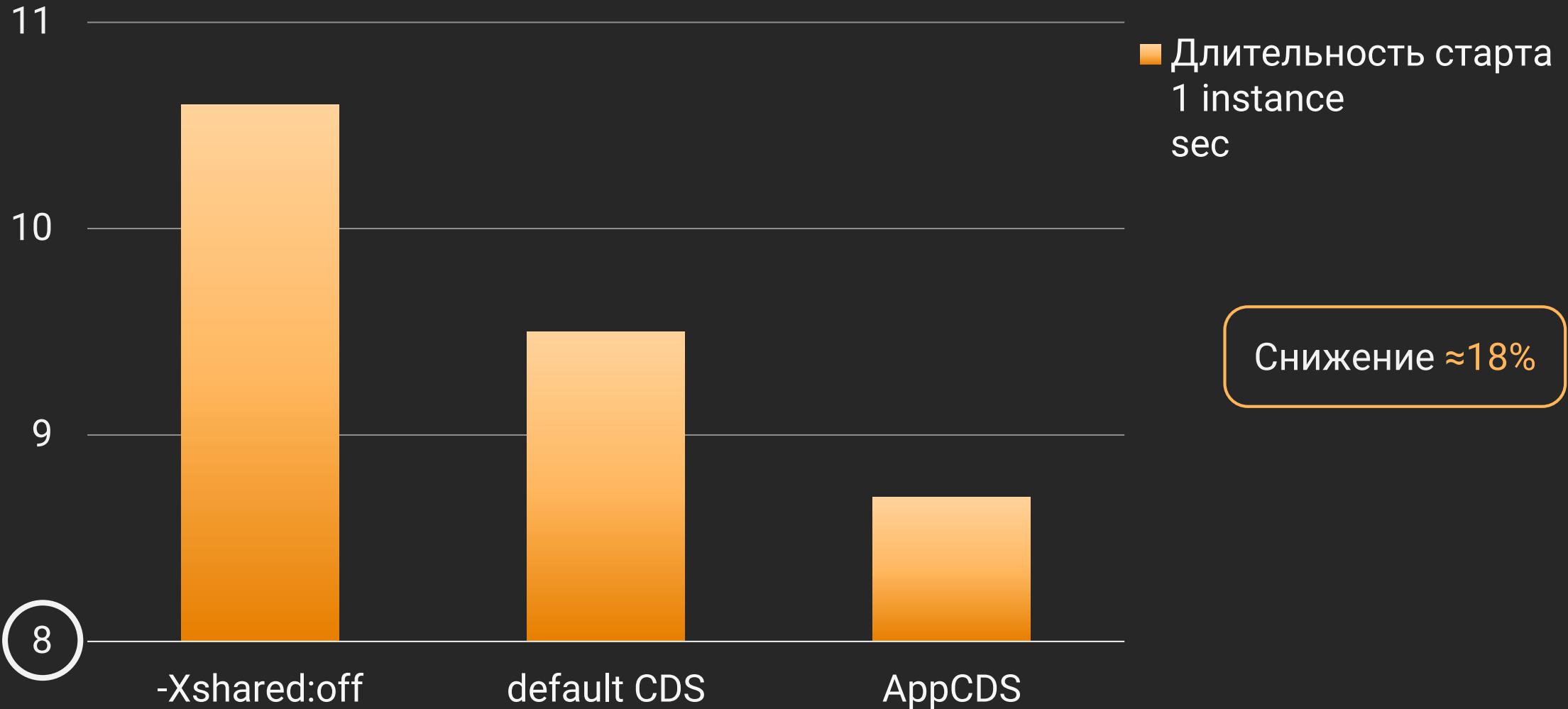
```
$ grep -o 'source: jrt:/' class-load.log | wc -l  
277
```

shell

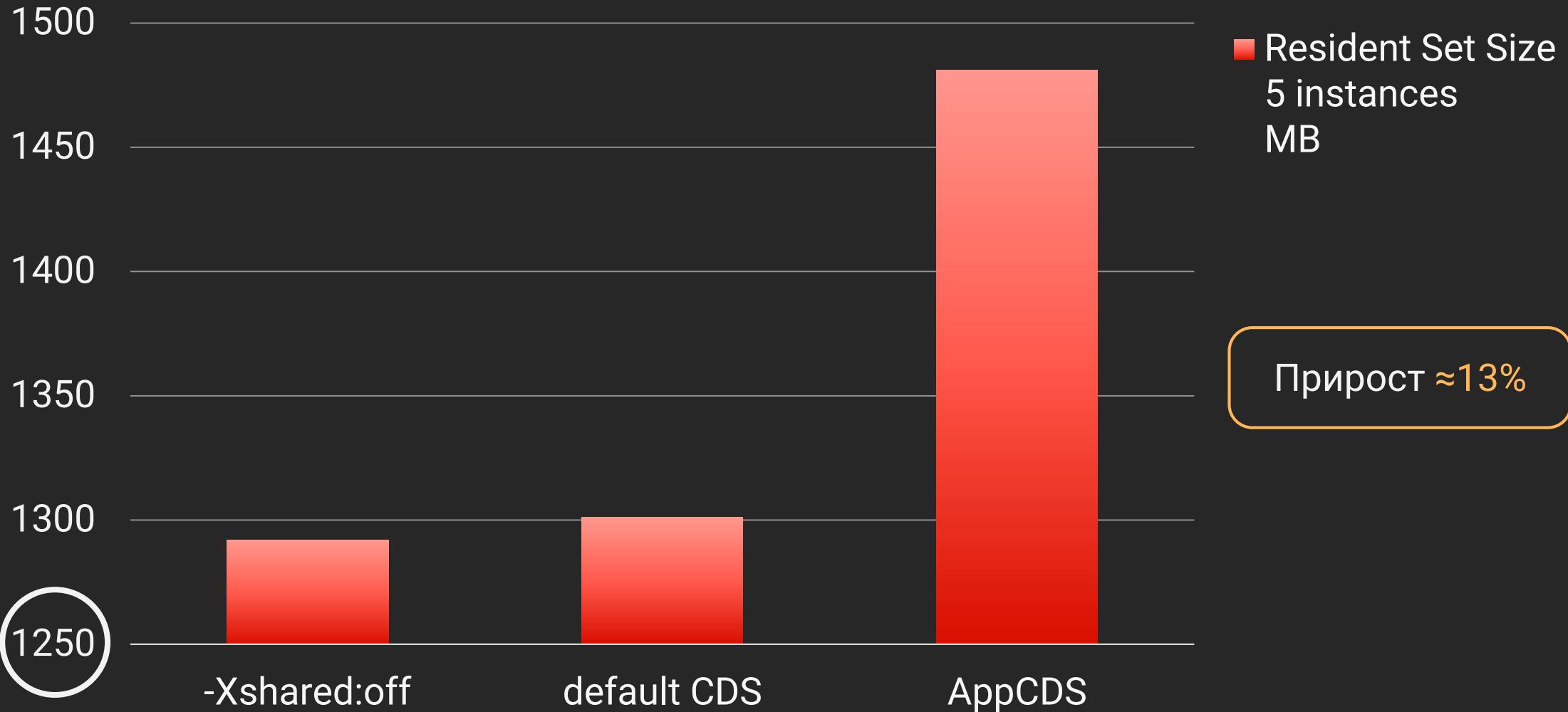
Распределение источников классов



Время запуска микросервиса



Потребление памяти (RSS)



Теперь, благодаря AppCDS,
Java сможет отжирать
больше памяти
за **меньшее** время!

Или нет?..



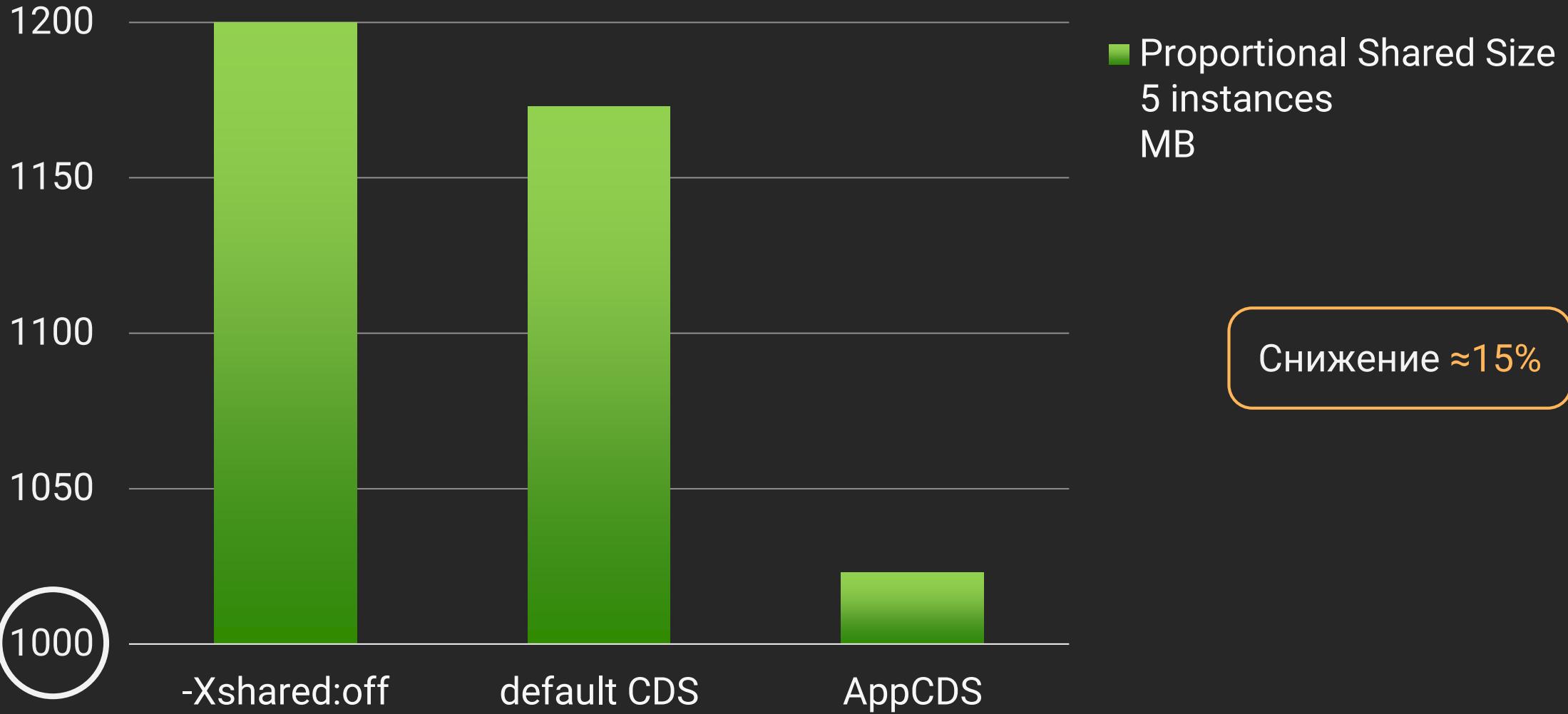
Поправка на реальность

“

... the memory footprint of a process ...
might appear to **increase**,
because **more pages are mapped**
to the process's address space

<https://docs.oracle.com/en/java/javase/11/vm/class-data-sharing.html>

Потребление памяти (PSS)



Профит в том, что
та же память достанется
другим JVM процессам

[App]CDS: текущий расклад дел

- Требует много ручных действий
- + Улучшает и время старта, и потребление памяти
- + Легко применим для основных классов JDK
- + Активно развивается и обрастает практиками

А если хочется деталей...

The slide features a purple gradient background with a white wolf logo and the text "jbreak; 2018". Below this, it says "Volker Simonis" and "SAP". A white speech bubble contains the text "Class data sharing in the HotSpot VM". To the right is a photo of Volker Simonis, a smiling man with short dark hair wearing a red t-shirt with a gear logo and "FOSDEM '14" text. The slide has a blue border at the bottom.

<https://2018.jbreak.ru/talks/76so01njvm2a0eqgagysm>

1. Погружение

2. Особенности перехода

- Сборка проекта
- Обновление Spring
- Deployment

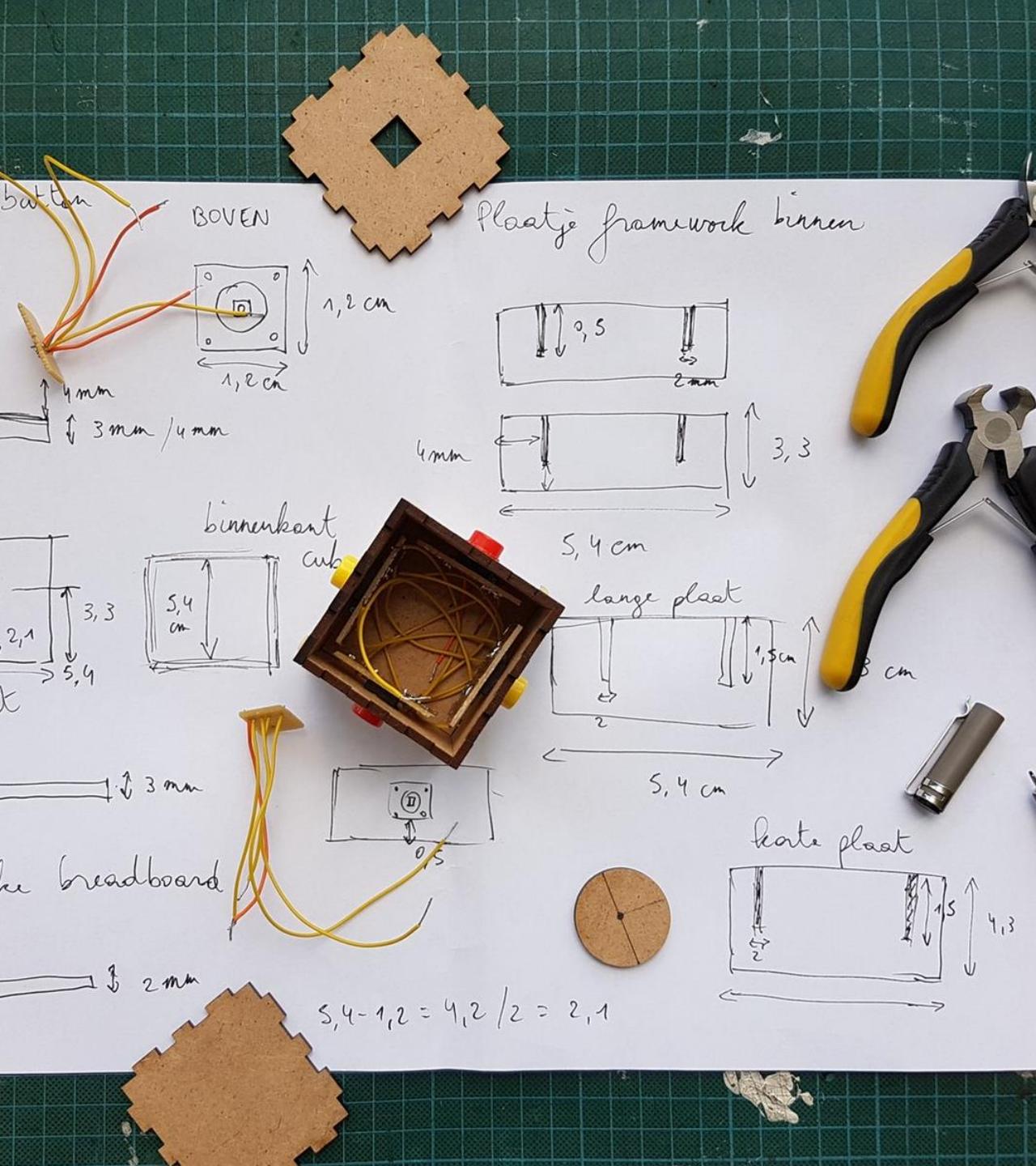
3. Новшества платформы

- Single-File Programs
- Class Data Sharing
- JShell

4. Всплытие



Мы часто пишем
наброски кода
для проверки гипотез
и создания прототипов



Прототипы можно писать в:

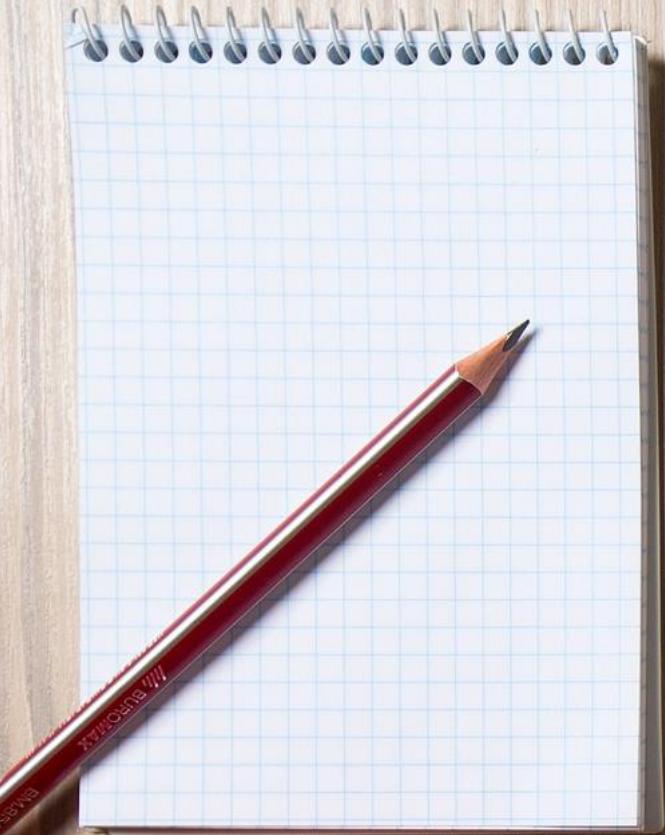
- JUnit тестах
- IDEA Scratches
- ~~методах main() в прикладных классах~~
(не надо так делать)
- *(свой вариант)*
- **JShell** (*не путать с “JS hell”*)

JEP-222, Java 9



А зачем!?

Write once,
run test
everywhere!



Пример применения

Дано:

- Библиотека Spring Cloud Commons
- Метод `InetUtils#findFirstNonLoopbackAddress`
- Возвращает «внешний» сетевой адрес машины

Найти:

- Что вернет метод при вызове изнутри Docker-контейнера?

```
public InetAddress findFirstNonLoopbackAddress() {  
    InetAddress result = null;  
    try {  
        int lowest = Integer.MAX_VALUE;  
        for (Enumeration<NetworkInterface> nics =  
            NetworkInterface.getNetworkInterfaces();  
            nics.hasMoreElements(); ) {  
            NetworkInterface ifc = nics.nextElement();  
            if (ifc.isUp()) {  
                log.trace("Testing interface: " + ifc.getDisplayName());  
                if (ifc.getIndex() < lowest || result == null) {  
                    lowest = ifc.getIndex();  
                } else if (result != null) {  
                    continue;  
                }  
                if (!ignoreInterface(ifc.getDisplayName())) {  
                    for (Enumeration<InetAddress> addrs = ifc  
                        .getInetAddresses(); addrs.hasMoreElements(); ) {  
                        InetAddress address = addrs.nextElement();  
                        if (address instanceof Inet4Address  
                            && !address.isLoopbackAddress()  
                            && isPreferredAddress(address)) {  
                            log.trace("Found non-loopback interface: "  
                                + ifc.getDisplayName());  
                            result = address;  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

Предсказуемо,
не правда ли?

Поправка на реальность: Spring Boot

- JShell нужны классы приложения и библиотек
- Классы запакованы в über JAR/WAR
- JShell **не умеет** работать с такими архивами

Вариант решения

1. Распаковать über JAR/WAR
2. Составить значение опции `--class-path` для JShell
3. Запустить JShell с этой опцией
4. Упахаться, повторяя это каждый раз
Автоматизировать шаги 1-3



`jshellw` – обертка над JShell для запуска с classpath'ом из Spring Boot Jar/War
<https://github.com/toparvion/springboot-jshell-adapter>

Запуск JShell через обертку

```
Created temp directory /tmp/...
Extracted 191 files from the archive to /tmp/...
Starting JShell with '/usr/bin/jshell --class-path ...
|   Welcome to JShell -- Version 11.0.1
|   For an introduction type: /help intro

jshell> ./jshellw app.jar
```



jshellw – обертка над JShell для запуска с classpath'ом из Spring Boot Jar/War

<https://github.com/toparvion/springboot-jshell-adapter>

Проверка корректности classpath

```
jshell> /env  
| --class-path /tmp/.../BOOT-INF/classes:  
|           /tmp/.../BOOT-INF/lib/lib1.jar:  
|           /tmp/.../BOOT-INF/lib/lib2.jar:...  
  
./jshellw app.jar
```



jshellw – обертка над JShell для запуска с classpath'ом из Spring Boot Jar/War

<https://github.com/toparvion/springboot-jshell-adapter>

Запуск прототипа в JShell

```
jshell> import ...InetUtils  
jshell> import ...InetUtilsProperties  
jshell> var utils = new InetUtils(new InetUtilsProperties())  
jshell> var myAddress = utils.findFirstNonLoopbackAddress()  
myAddress ==> /172.29.6.175  
|   created variable myAddress : InetAddress
```

./jshellw app.jar



jshellw – обертка над JShell для запуска с classpath'ом из Spring Boot Jar/War

<https://github.com/toparvion/springboot-jshell-adapter>

Другие применения JShell в контейнерах



Отладка
операций
с файлами



Работа со
свойствами
окружения



Тестирование
специфичных
библиотек (API)

1. Погружение

2. Особенности перехода

- Сборка проекта
- Обновление Spring
- Deployment

3. Новшества платформы

- Single-File Programs
- Class Data Sharing
- JShell

4. Всплытие



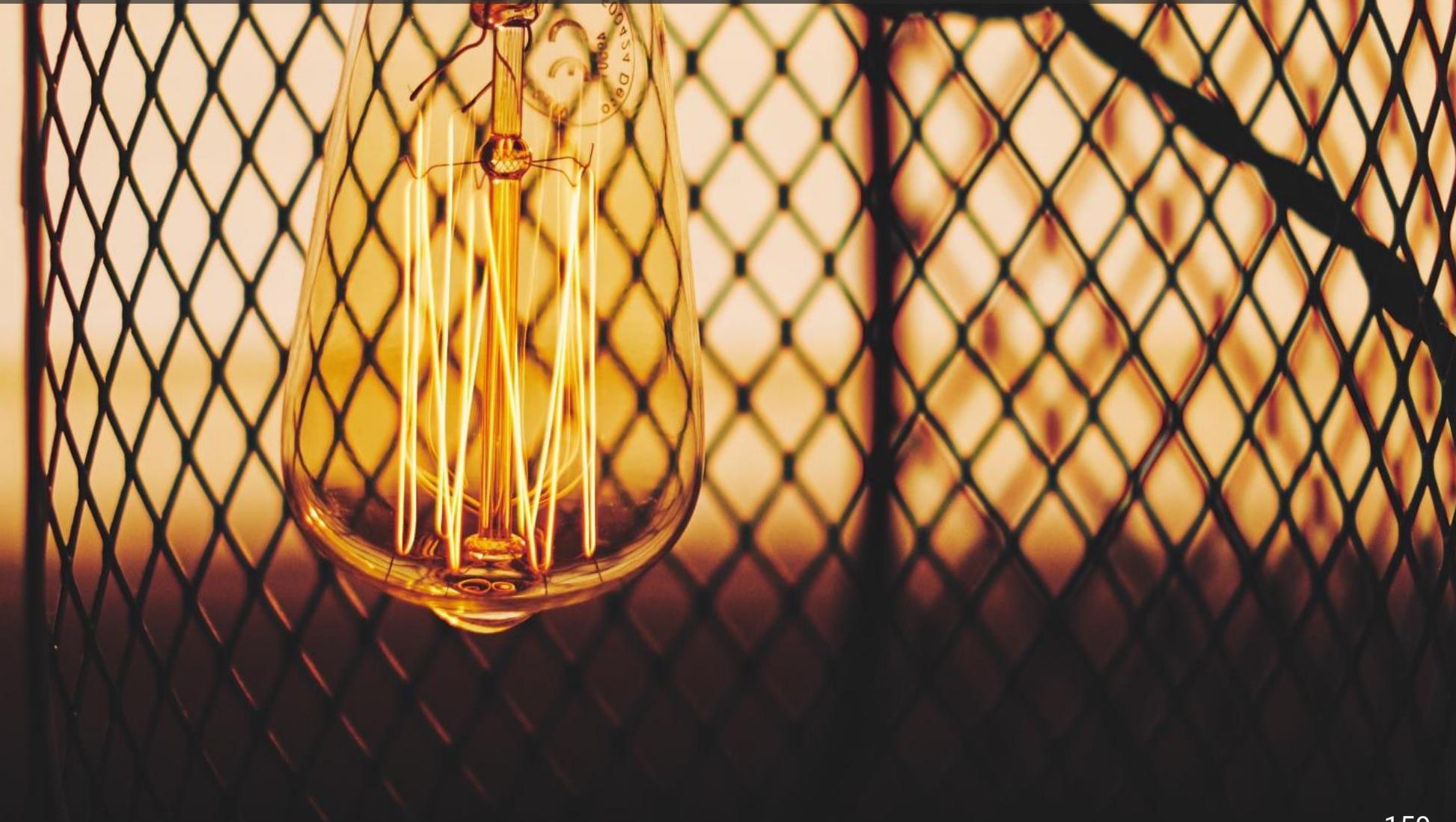
Обновлять Java –
как делать ремонт:
не обязательно,
сложно, но **полезно**



Версии инструментов с поддержкой Java 11

- Сборка:
 - Maven 3.5
 - Gradle 5.1
- Spring:
 - Boot 2.1
 - Framework 5.1
 - Cloud Greenwich
 - Lombok 1.18

Java 8 (пока) никуда не уходит 😊



Когда ждать следующей мажорной версии?

“

There are no “major releases” per se any more ...
Instead there is a steady stream
of “feature releases.”

<https://blogs.oracle.com/java-platform-group/update-and-faq-on-the-java-se-release-cadence>

Режимы обновления на любой вкус

- «На каждый чих»
По мере появления фиксов
- При выходе feature releases
2 раза в год (март, сентябрь)
- При выходе LTS releases
1 раз в 3 года (2014, 2018, 2021)

Диетологи советуют
питаться часто,
но по чуть-чуть





Владимир Плизгá

ЦФТ

 @toparvion

 Топарвон

Q & A