Обратная совместимость

инструменты и подходы для контроля над Dependency Hell

Спикеры

Александр Лампель

Сбер

@lampelay

Евгений Калинин

Сбер

@david_berg

Работаем в Сбере

несколько лет над проектом

Сбербанк Онлайн

Agenda

Dependency hell + Версионирование + Совместимость

Специфика в Сбере

Версии

Совместимость

Инструменты

Примеры

Выводы

Мы расскажем об инструментах и подходах, которые помогут обезопасить себя при работе с N количеством внешних зависимостей.

Важно, что зависимости часто изменяются и при этом имеют еще и транзитивные зависимости на другие библиотеки

Ограничения

Вселенная Java

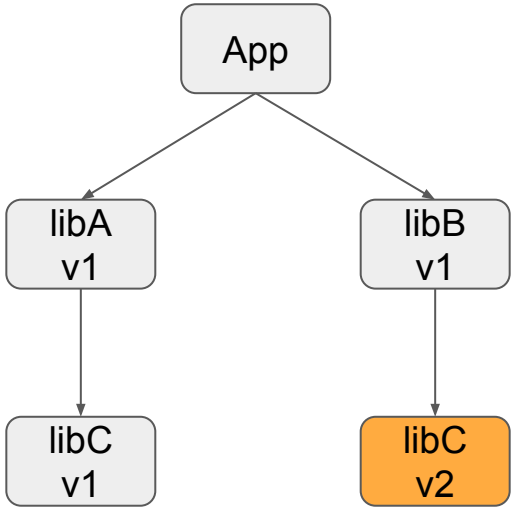
Но не только JVM-based языки (Scala, Groovy, Kotlin & etc.)

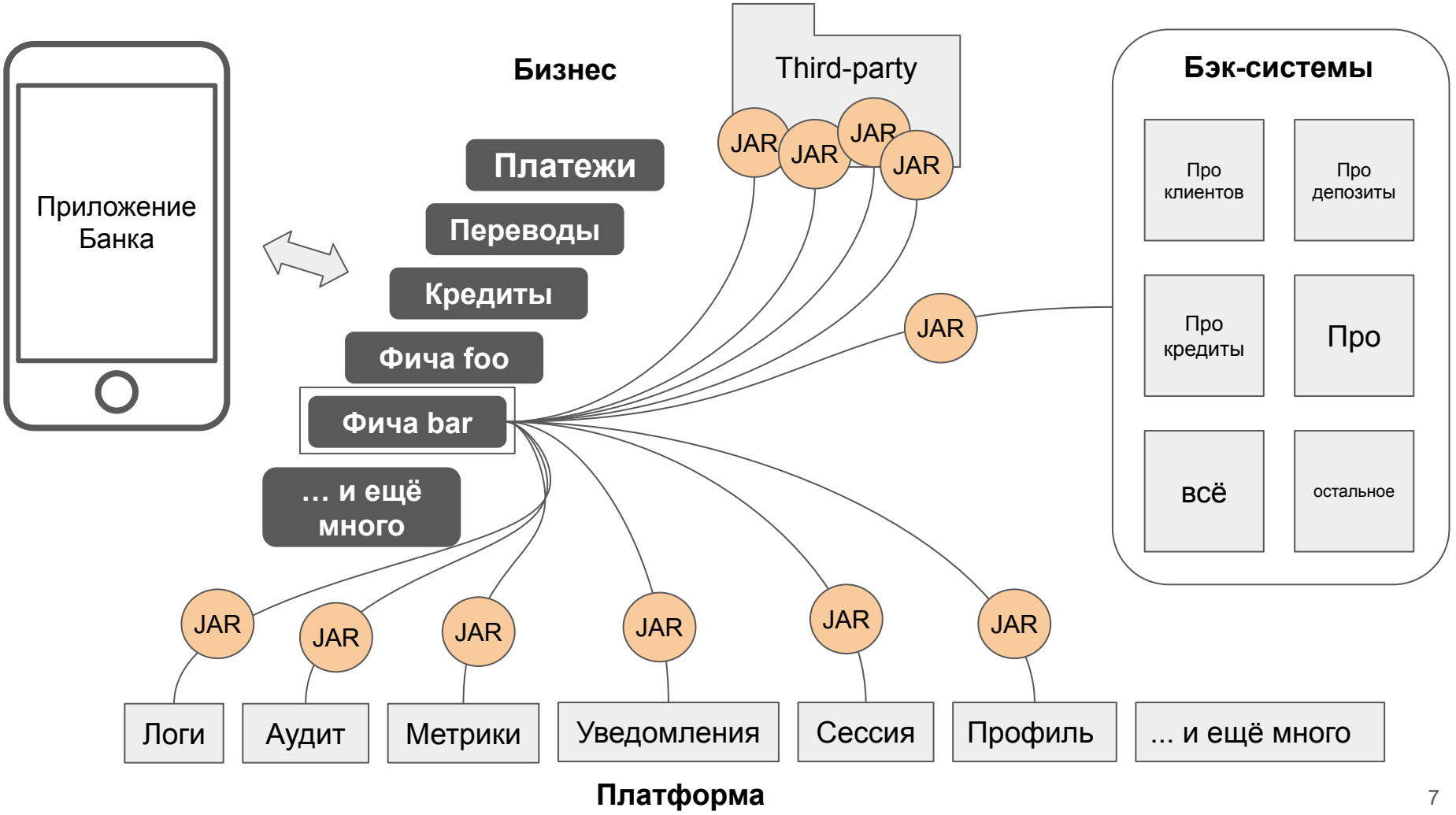
Предполагается, что вы знакомы с понятиями

- ➔ jar файл
- ➔ API
- ➔ Maven + POM

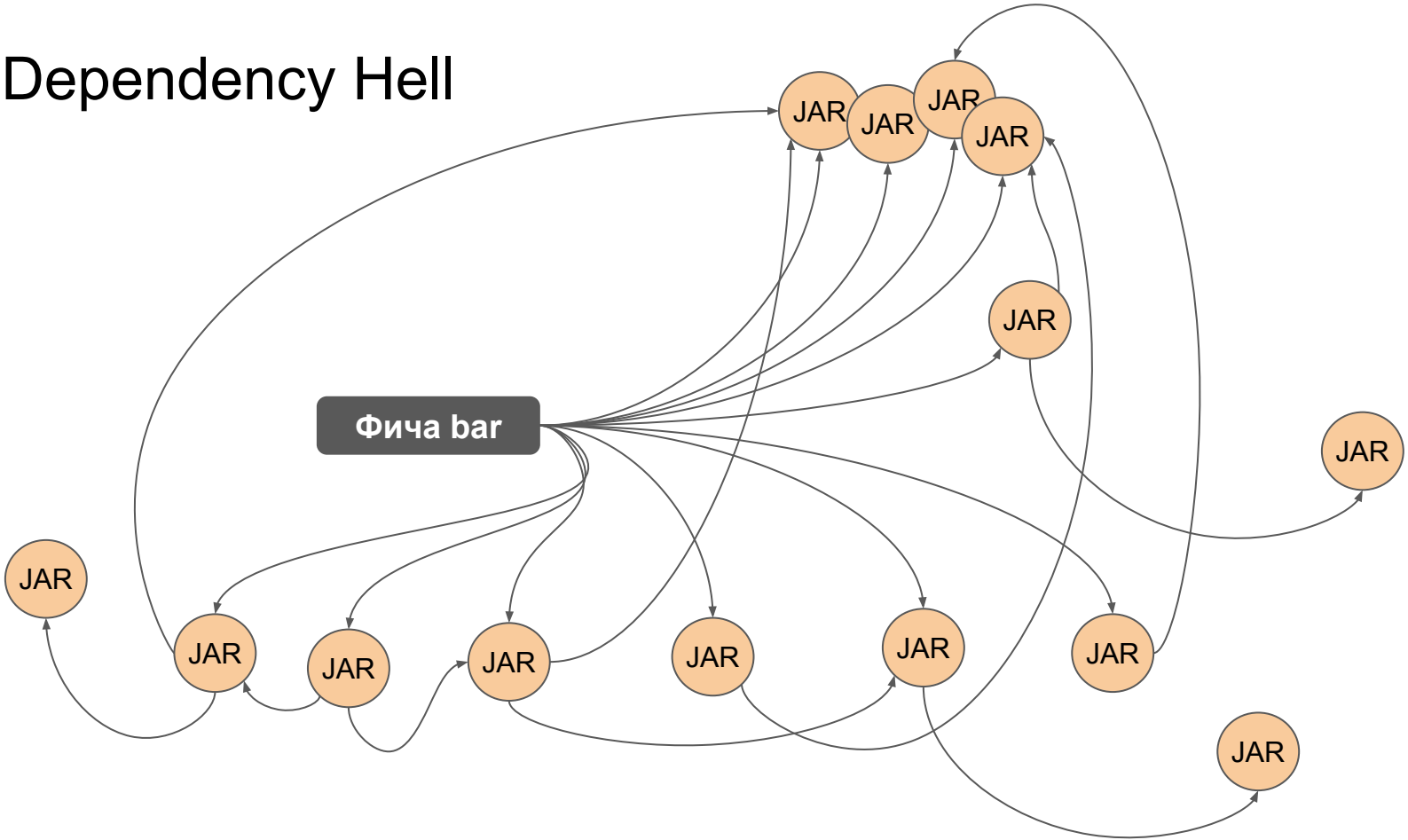
Проблема dependency hell

- ➔ Любое приложение это import N внешних библиотек (зависимостей)
- ➔ Приложение любого масштаба
- ➔ Dependency hell <=> jar hell – в контексте доклада это эквивалентно




Приложение Банка
Бизнес
Платежи
Переводы
Кредиты
Фича foo
Фича bar
… и ещё много
Third-party
JAR
JAR
JAR
JAR
JAR
Бэк-системы
Про клиентов
Про депозиты
Про кредиты
Про
всё
остальное
JAR
JAR
JAR
JAR
JAR
JAR
Логи
Аудит
Метрики
Уведомления
Сессия
Профиль
... и ещё много
Платформа

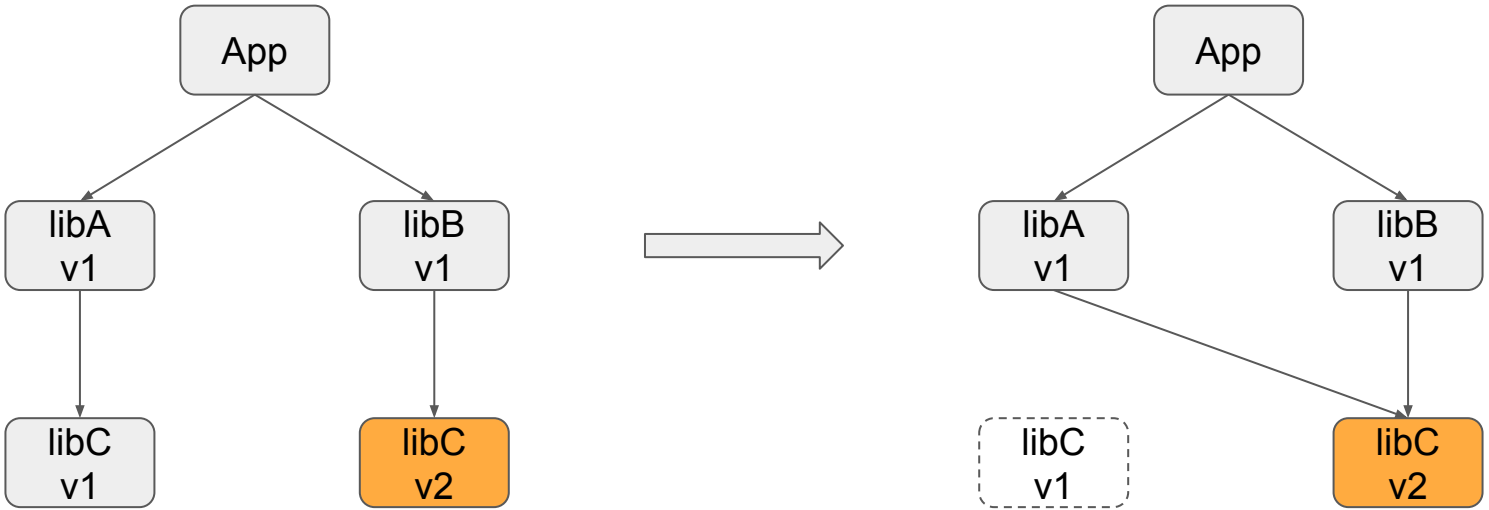
Dependency Hell



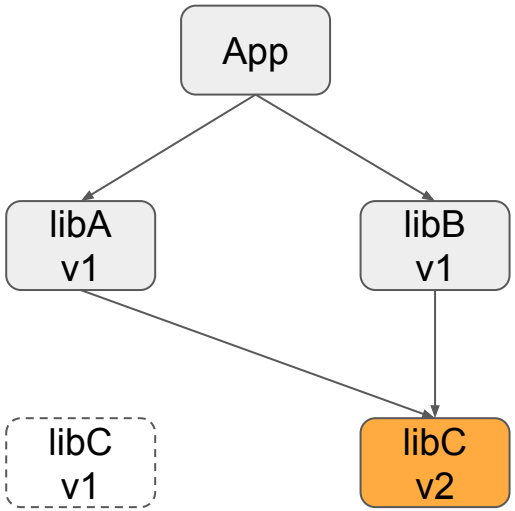
Dependency Hell

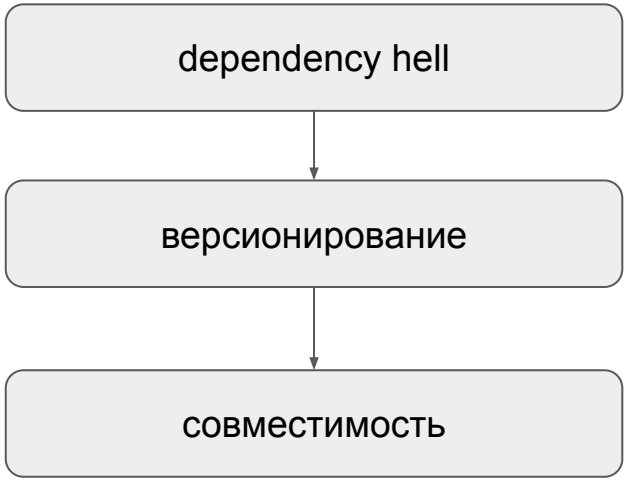
- ➔ Библиотеки зависят друг от друга
- ➔ Все библиотеки тянут транзитивные зависимости
- ➔ Версии зависимостей библиотек не согласованы
- ➔ Библиотеки обновляются в разное время

Dependency Hell и версионирование



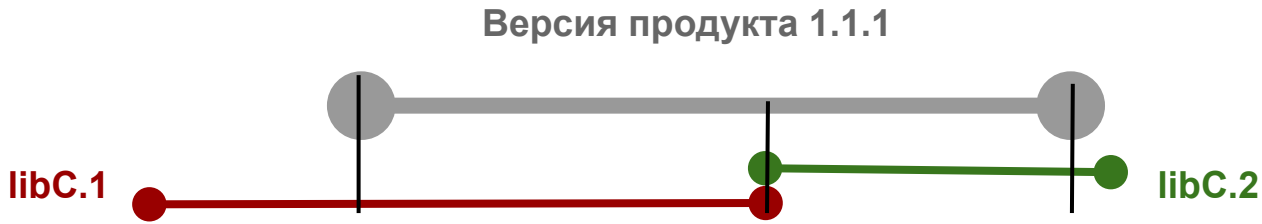
Dependency Hell и версионирование





Совместимость и Dependency Hell

Как, когда и чем контролируя совместимость зависимостей (библиотек) управлять dependency hell

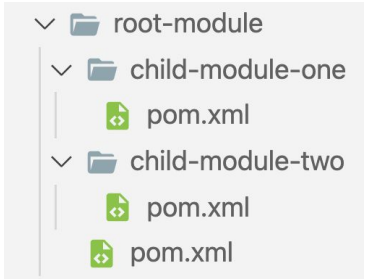


Такой подход это следствие ограничений со стороны

- продукт живет долго
- должны поддерживать его версии для Java 1.7, 1.8, 11
- команд много и все движутся со своей скоростью

Версионирование

Dependency Management



Родительский модуль

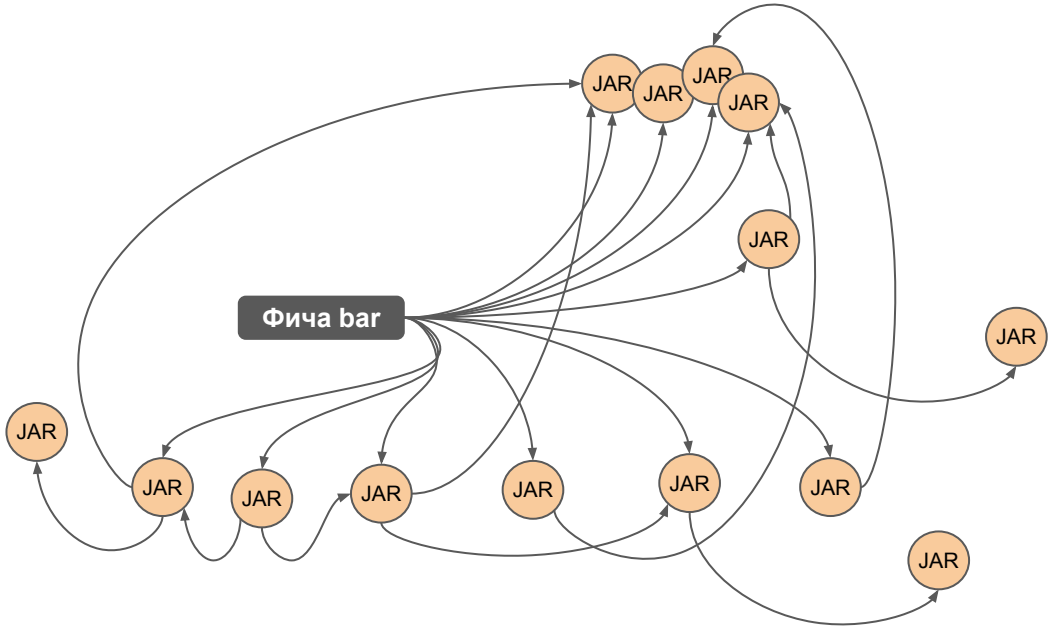
```
<groupId>ru.sbrf</groupId>
<artifactId>root-module</artifactId>
<version>1.0</version>

<dependencyManagement>
  <dependencies>

    <dependency>
      <groupId>org.lib.group</groupId>
      <artifactId>library-one</artifactId>
      <version>1.2.1</version>
    </dependency>

    <dependency>
      <groupId>org.another.lib.group</groupId>
      <artifactId>library-two</artifactId>
      <version>2.6.9</version>
    </dependency>

  </dependencies>
</dependencyManagement>
```

Дочерний модуль

```
<parent>
  <groupId>ru.sbrf</groupId>
  <artifactId>root-module</artifactId>
  <version>1.0</version>
</parent>

<artifactId>child-module</artifactId>
<version>1.0</version>

<dependencies>

  <dependency>
    <groupId>org.lib.group</groupId>
    <artifactId>library-one</artifactId>
  </dependency>

  <dependency>
    <groupId>org.another.lib.group</groupId>
    <artifactId>library-two</artifactId>
  </dependency>

</dependencies>
```

BOM

Bill of materials – ведомость материалов



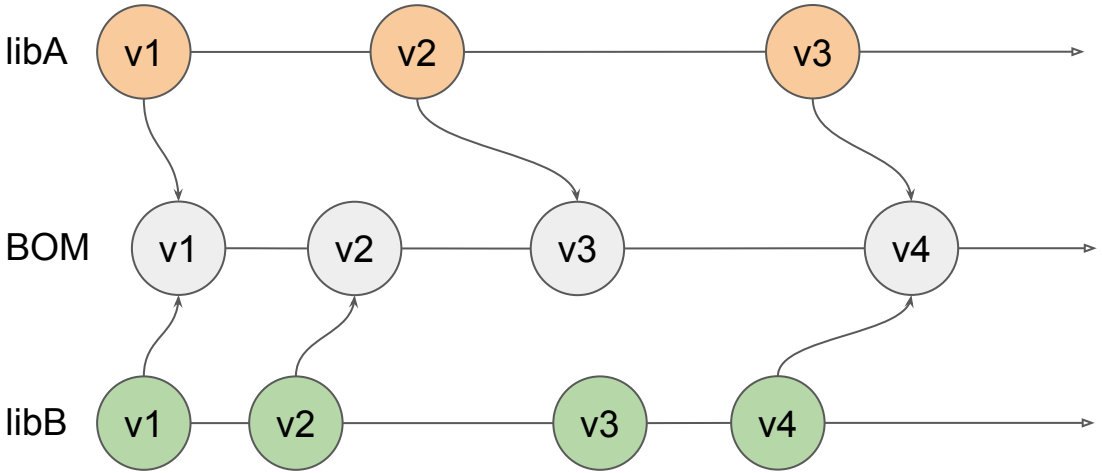

```
<groupId>ru.sbrf</groupId>
<artifactId>my-project-bom</artifactId>
<version>1.0</version>


<dependencyManagement>
  <dependencies>

    <dependency>
      <groupId>org.lib.group</groupId>
      <artifactId>library-one</artifactId>
      <version>1.2.1</version>
    </dependency>

    <dependency>
      <groupId>org.another.lib.group</groupId>
      <artifactId>library-two</artifactId>
      <version>2.6.9</version>
    </dependency>

    <dependency>
      <groupId>org.another.lib.group</groupId>
      <artifactId>library-three</artifactId>
      <version>4.0</version>
    </dependency>

    <dependency>
      <groupId>org.another.lib.group</groupId>
      <artifactId>library-four</artifactId>
      <version>9.9.9</version>
    </dependency>

  </dependencies>
</dependencyManagement>
```

BOM

```xml
<groupId>ru.sbrf</groupId>
<artifactId>root-module</artifactId>
<version>2.0</version>

<dependencyManagement>
  <dependencies>

    <dependency>
      <groupId>ru.sbrf</groupId>
      <artifactId>my-project-bom</artifactId>
      <version>1.0</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>

  </dependencies>
</dependencyManagement>
```

```xml
<groupId>ru.sbrf</groupId>
<artifactId>root-module</artifactId>
<version>2.0</version>

<dependencyManagement>
  <dependencies>

    <dependency>
      <groupId>org.lib.group</groupId>
      <artifactId>library-one</artifactId>
      <version>1.2.1</version>
    </dependency>

    <dependency>
      <groupId>org.another.lib.group</groupId>
      <artifactId>library-two</artifactId>
      <version>2.6.9</version>
    </dependency>

    <dependency>
      <groupId>org.another.lib.group</groupId>
      <artifactId>library-three</artifactId>
      <version>4.0</version>
    </dependency>

    <dependency>
      <groupId>org.another.lib.group</groupId>
      <artifactId>library-four</artifactId>
      <version>9.9.9</version>
    </dependency>

  </dependencies>
</dependencyManagement>
```

Изменения в BOM



git

artifact repository

Совместимость

Совместимость

Бывает:

- ➔ **прямая** - работает, если откатиться на предыдущую версию
- ➔ **обратная** - работает, если обновиться на новую версию
- ➔ **полная** - работает вне зависимости от подключённой версии

Обратная совместимость

Сохранение работоспособности при замене старой версии компонента на новую.

➔ **binary** – на уровне скомпилированного кода

Работает, если не компилировать и подменить jar в папке lib.

➔ **sources** – на уровне исходного кода

Код компилируется с новой версией библиотеки.

Обратная совместимость

Пример **сохранения** обратной совместимости

```
/** v1.0 */
class MyLittleSerivce {
    public List<String> getSomeRandomWords() {
        {…
    }
}
```

```
/** v1.1 */
class MyLittleSerivce {
    public List<String> getSomeRandomWords() {
        {…
    }

    // добавили новый метод
    public List<Integer> getSomeNumbers() {
        {…
    }
}
```

Обратная совместимость

Примеры **нарушения** обратной совместимости

```java
/** v1.0 */
interface MyLittleSerivce {

    public List<String> findSomeWords();
}
```

```java
/** v1.1 */
interface MyLittleSerivce {
    // изменилась сигнатура метода — добавили аргумент
    public List<String> findSomeWords(String searchPattern);
}
```

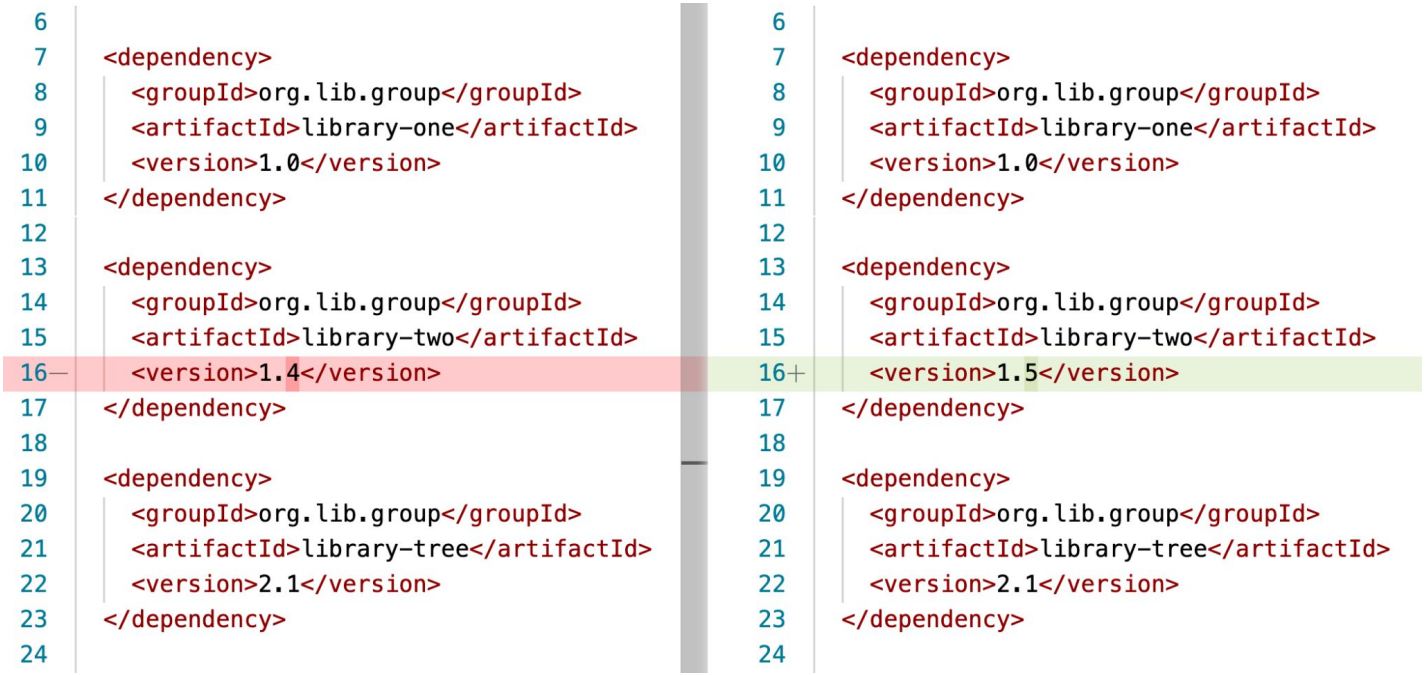
```java
/** v1.1 */
interface MyLittleSerivce {
    public List<String> getSomeRandomWords();

    public List<Integer> getSomeNumbers();
}
```

```java
/** v1.2 */
interface MyLittleSerivce {
    // метод getSomeRandomWords исчез

    public List<Integer> getSomeNumbers();
}
```

Проверка совместимости

Триггер на

Pull request



```
6
7   <dependency>
8     <groupId>org.lib.group</groupId>
9     <artifactId>library-one</artifactId>
10    <version>1.0</version>
11  </dependency>
12
13  <dependency>
14    <groupId>org.lib.group</groupId>
15    <artifactId>library-two</artifactId>
16-   <version>1.4</version>
17  </dependency>
18
19  <dependency>
20    <groupId>org.lib.group</groupId>
21    <artifactId>library-tree</artifactId>
22    <version>2.1</version>
23  </dependency>
24
```

```
6
7   <dependency>
8     <groupId>org.lib.group</groupId>
9     <artifactId>library-one</artifactId>
10    <version>1.0</version>
11  </dependency>
12
13  <dependency>
14    <groupId>org.lib.group</groupId>
15    <artifactId>library-two</artifactId>
16+   <version>1.5</version>
17  </dependency>
18
19  <dependency>
20    <groupId>org.lib.group</groupId>
21    <artifactId>library-tree</artifactId>
22    <version>2.1</version>
23  </dependency>
24
```

Проверка совместимости

Тестовый класс для запуска инструмента проверки обратной совместимости

Для получения разницы между старой и новой версией зависимости по **git diff**

BOM-файл

Обратная совместимость Java API

Наши ожидания от инструмента

➔ сравнить две jar-ки

➔ отчёт по результатам сравнения

Опционально

➔ метаданные результатов для автоматизации проверки

➔ вызов из Java кода

Обратная совместимость Java API

Инструменты

- ➔ japi-compliance-checker
- ➔ revapi
- ➔ japicmp
- ➔ sigtest

japi-compliance-checker

https://github.com/lvc/japi-compliance-checker

Написан на Perl

Рисует красивый отчёт

Код возврата 0 – совместимость сохранена

Нет своего Java API

japi-compliance-checker

Вызов из командной строки

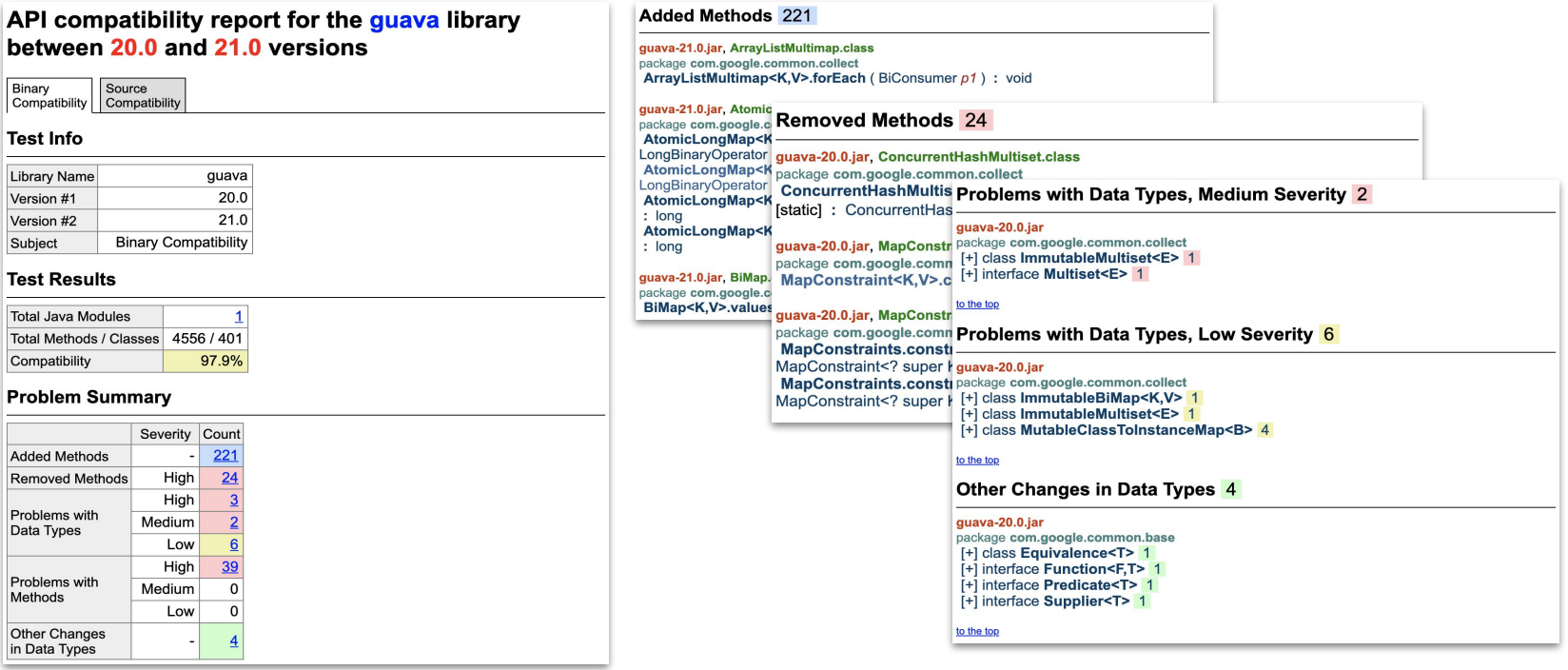
```
~ $   perl  japi-compliance-checker.pl  guava-20.0.jar  guava-21.0.jar
~ $   open  ./compat_reports/guava/20.0_to_21.0/compat_report.html
```

Вызов из Java кода

```java
public boolean checkCompatibility(String pathToOldVersion, String pathToNewVersion)
        throws IOException, InterruptedException {

    int exitCode = new ProcessBuilder()
        .command("perl", "japi-compliance-checker.pl", pathToOldVersion, pathToNewVersion)
        .start()
        .waitFor();

    return exitCode == 0;
}
```

japi-compliance-checker

Отчёт



API compatibility report for the guava library between 20.0 and 21.0 versions

Binary Compatibility | Source Compatibility

Test Info

Library Name	guava
Version #1	20.0
Version #2	21.0
Subject	Binary Compatibility

Test Results

Total Java Modules	1
Total Methods / Classes	4556 / 401
Compatibility	97.9%

Problem Summary

	Severity	Count
Added Methods	-	221
Removed Methods	High	24
Problems with Data Types	High	3
	Medium	2
	Low	6
Problems with Methods	High	39
	Medium	0
	Low	0
Other Changes in Data Types	-	4

Added Methods 221

guava-21.0.jar, ArrayListMultimap.class
package com.google.common.collect
ArrayListMultimap<K,V>.forEach (BiConsumer *p1*) : void

guava-21.0.jar, Atomic
package com.google.c
AtomicLongMap<K
LongBinaryOperator
AtomicLongMap<K
LongBinaryOperator
AtomicLongMap<K
: long
AtomicLongMap<K
: long

guava-21.0.jar, BiMap.
package com.google.c
BiMap<K,V>.values

Removed Methods 24

guava-20.0.jar, ConcurrentHashMultiset.class
package com.google.common.collect
ConcurrentHashMultis
[static] : ConcurrentHas

guava-20.0.jar, MapConstr
package com.google.comm
MapConstraint<K,V>.c

guava-20.0.jar, MapConstr
package com.google.comm
MapConstraints.consti
MapConstraint<? super K
MapConstraints.consti
MapConstraint<? super K

Problems with Data Types, Medium Severity 2

guava-20.0.jar
package com.google.common.collect
[+] class ImmutableMultiset<E> 1
[+] interface Multiset<E> 1

to the top

Problems with Data Types, Low Severity 6

guava-20.0.jar
package com.google.common.collect
[+] class ImmutableBiMap<K,V> 1
[+] class ImmutableMultiset<E> 1
[+] class MutableClassToInstanceMap<B> 4

to the top

Other Changes in Data Types 4

guava-20.0.jar
package com.google.common.base
[+] class Equivalence<T> 1
[+] interface Function<F,T> 1
[+] interface Predicate<T> 1
[+] interface Supplier<T> 1

to the top

RevApi

https://revapi.org

Написан на Java

Есть Java API

Есть плагин для Maven

Можно указать свой шаблон для отчёта

RevApi

Вызов из Java кода

```
public boolean checkCompatibility(Archive oldJar, Archive newJar) {

    Revapi revapi = Revapi.builder().withAllExtensionsFromThreadContextClassLoader().build();

    AnalysisContext analysisContext = AnalysisContext.builder()
        .withOldAPI(API.of(oldJar).build())
        .withNewAPI(API.of(newJar).build())
        .withConfigurationFromJSON("{ /* json with configuration. See docs... */ }").build();

    AnalysisResult result = revapi.analyze(analysisContext);

    return result.isSuccess();
}
```

RevApi

Отчёт

RevAPI report

Previous artifact:

- com.google.guava : guava : 20.0

☐ Show supplementary

Next artifact:

- com.google.guava : guava : 21.0

☐ Show supplementary

Differences

☐ Show non breaking

Old:

—

New:
com.google.guava : guava : 21.0

```
method boolean com.google.common.base.Equivalence<T>::test(T, T)
```

- A final method has been added to an inheritable class. (`java.method.finalMethodAddedToNonFinalClass`)

  Binary : **potentially breaking** | *Source* : **potentially breaking**

Old:
com.google.guava : guava : 20.0

```
class com.google.common.base.Objects.ToStringHelper
```

New:

—

- Class was removed. (`java.class.removed`)

  Binary : **breaking** | *Source* : **breaking**

Old:
com.google.guava : guava : 20.0

```
method <T> T com.google.common.base.Objects::firstNonNull(T, T)
```

New:

—

- Method was removed. (`java.method.removed`)

  Binary : **breaking** | *Source* : **breaking**

japicmp

https://siom79.github.io/japicmp

Похож на RevApi

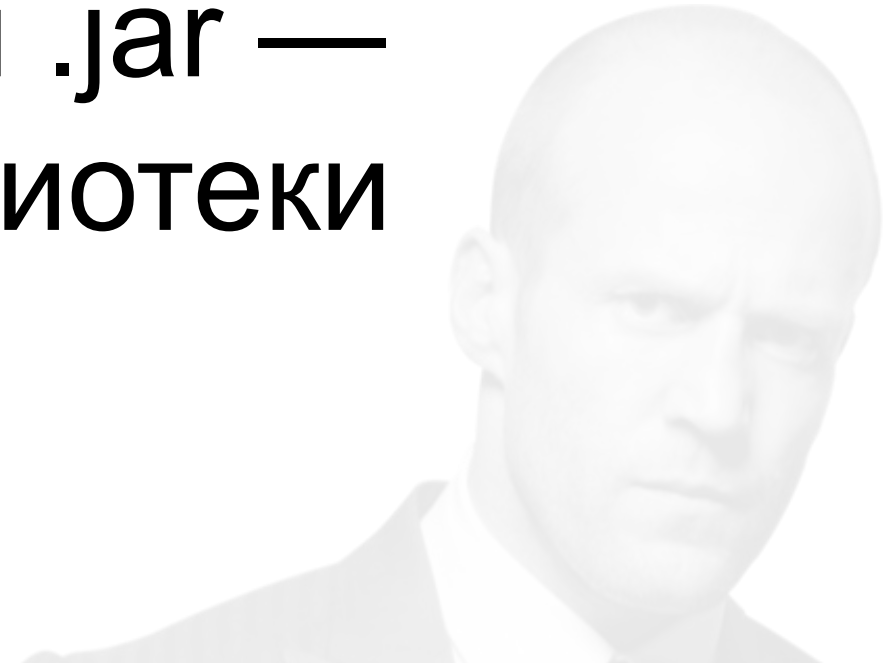
SigTest

https://wiki.openjdk.java.net/display/CodeTools/SigTest

Разрабатывается Oracle

Не только инструмент сравнения

Не всё внутри .jar — Java API библиотеки



Разделение API и реализации

➔ Классы и интерфейсы в определённых пакетах внутри **.jar**

API

```java
package my.superlib.api;

public interface SuperService {

    void doIt();

}
```

IMPL

```java
package my.superlib.impl;

import my.superlib.api.SuperService;

import org.springframework.stereotype.Service;

@Service
public class SuperServiceImpl implements SuperService {
    public void doIt() {
        /** реализация */
    }

    public void doSomeInternalLogic() {
        /** то, что не должно использоваться напрямую */
    }
}
```

Разделение API и реализации

➔ Отдельная **.jar** с Java API и отдельная с реализацией

API

```
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>1.7.32</version>
</dependency>
```

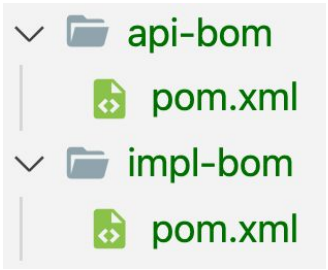
IMPL

```
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-simple</artifactId>
  <version>1.7.32</version>
</dependency>
```

```
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>log4j-over-slf4j</artifactId>
  <version>1.7.32</version>
</dependency>
```

Разделение API и реализации

Два разных BOM артефакта



– проверяем

– игнорируем

Пометка API зависимостей в BOM

```
<dependency>
  <!-- API -->
  <groupId>org.lib.group</groupId>
  <artifactId>library-one-api</artifactId>
  <version>1.0</version>
</dependency>
<dependency>
  <groupId>org.lib.group</groupId>
  <artifactId>library-one-impl</artifactId>
  <version>1.0</version>
</dependency>

<dependency>
  <!-- API -->
  <groupId>org.lib.group</groupId>
  <artifactId>library-two-api</artifactId>
  <version>1.5</version>
</dependency>
<dependency>
  <groupId>org.lib.group</groupId>
  <artifactId>library-tree</artifactId>
  <version>2.1</version>
</dependency>
```

Обратная совместимость конфигурации

Параметры Spring-Boot стартеров

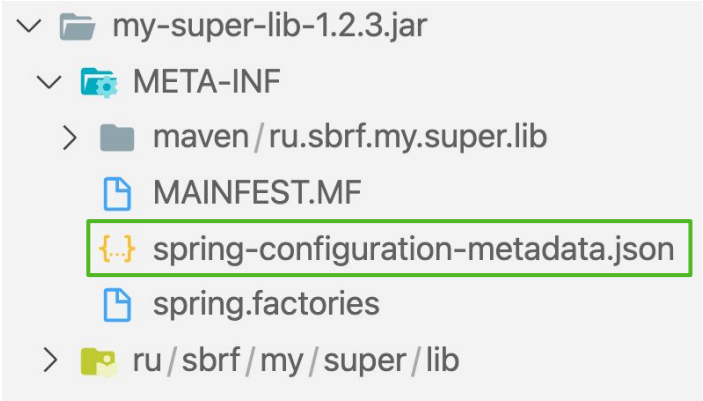
application.properties

```
my-super-lib.config.parameter-one=123
my-super-lib.config.parameter-two=false
my-super-lib.config.parameter-three=Foo. Bar!
```

application.yaml

```yaml
my-super-lib:
  config:
    parameter-one: 123
    parameter-two: false
    parameter-three: Foo. Bar!
```

Обратная совместимость конфигурации



Генерируется при подключении

spring-boot-configuration-processor

```
{
  "groups": [
    {
      "name": "ru.sbrf.my.super.lib",
      "type": "ru.sbrf.my.super.lib.boot.MySuperLibConfigrationProperties",
      "sourceType": "ru.sbrf.my.super.lib.boot.MySuperLibConfigrationProperties"
    }
  ],
  "properties": [
    {
      "name": "my-super-lib.config.parameter-one",
      "type": "java.lang.Integer",
      "description": "Some number.",
      "sourceType": "ru.sbrf.my.super.lib.boot.MySuperLibConfigrationProperties",
      "defaultValue": 123
    },
    {
      "name": "my-super-lib.config.parameter-two",
      "type": "java.lang.Boolean",
      "description": "Some flag.",
      "sourceType": "ru.sbrf.my.super.lib.boot.MySuperLibConfigrationProperties",
      "defaultValue": false
    },
    {
      "name": "my-super-lib.config.parameter-three",
      "type": "java.lang.String",
      "description": "Some text.",
      "sourceType": "ru.sbrf.my.super.lib.boot.MySuperLibConfigrationProperties",
      "defaultValue": "Foo. Bar!"
    }
  ],
  "hints": []
}
```

Обратная совместимость конфигурации

Правила проверки обратной совместимости:

➔ Добавление нового параметра без значения по-умолчанию

➔ Изменение типа данных параметра

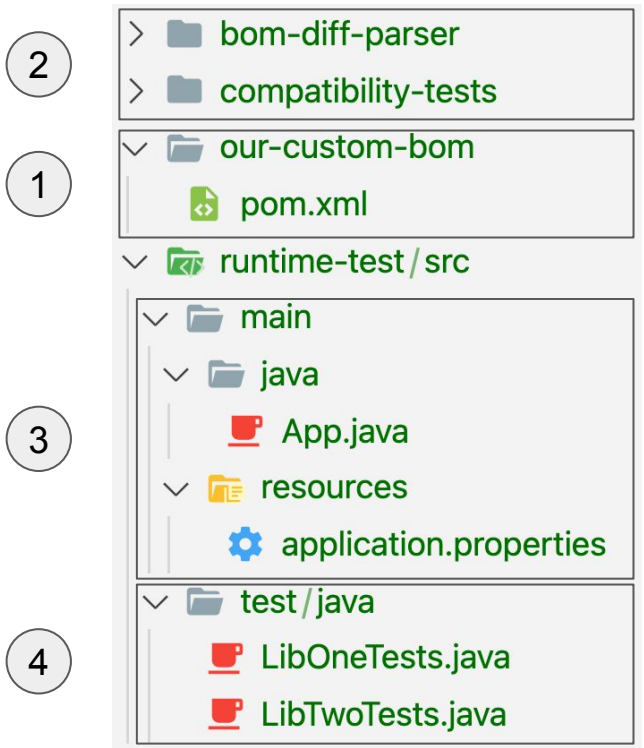
Дополнительные правила (не критично, но стоит обратить внимание):

➔ удаление параметра

➔ изменение значения по-умолчанию

Обратная совместимость конфигурации

Тесты зависимостей runtime



- ➔ Тесты самых важных сценариев использования
- ➔ Проверка работы в едином контексте

Выводы

- ➔ используем версионирование
- ➔ делаем валидный срез зависимостей через BOM
- ➔ храним предыдущее состояние API (например, в git)
- ➔ проверяем обратную совместимость Java API и конфигурации
- ➔ разделяем артефакты на api + impl
- ➔ тесты на контракты сторонних зависимостей

Пример другого подхода к dependency hell

https://youtu.be/aw6YJLJG5hw

Обратная совместимость

https://youtu.be/EgOZSr-Uc3w

Тестирование конфигурации

https://www.youtube.com/watch?v=Tk_nmV-mWOA

https://www.youtube.com/watch?v=SLZNVSb5vfY

Благодарности

Хотели бы поблагодарить отдельно наших коллег, кто активно участвовал

Александр Березкин

Мария Пастухова

Евгений Семенов

Александр Намаконов

Максим Немец

Игорь Абрамов

Михаил Подгорняк