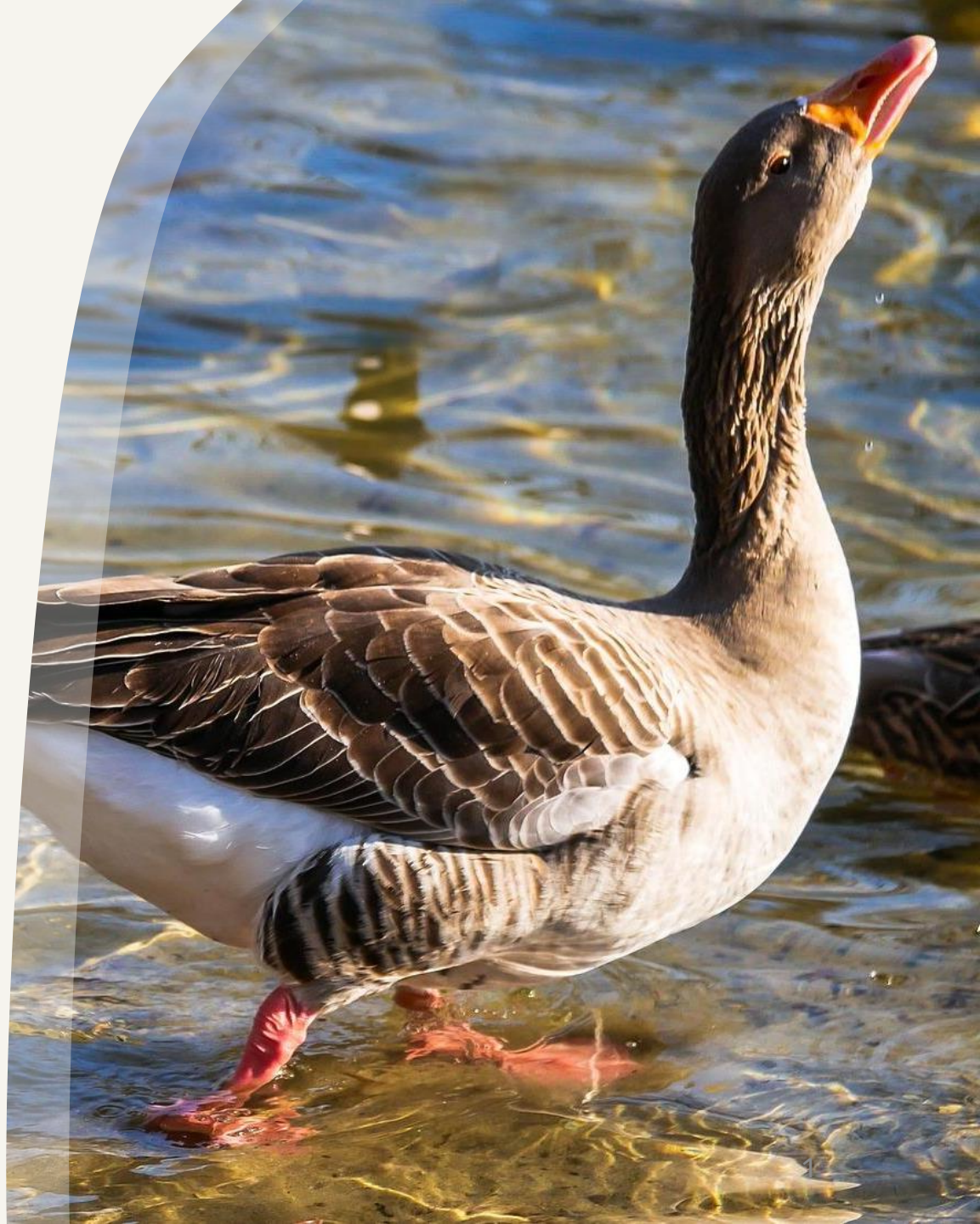


Инъекция тестовых поведений

Как выйти сухим из воды?

Владимир Плизга
ЦФТ, PrePaid



ПИЦЦЕРИЯ

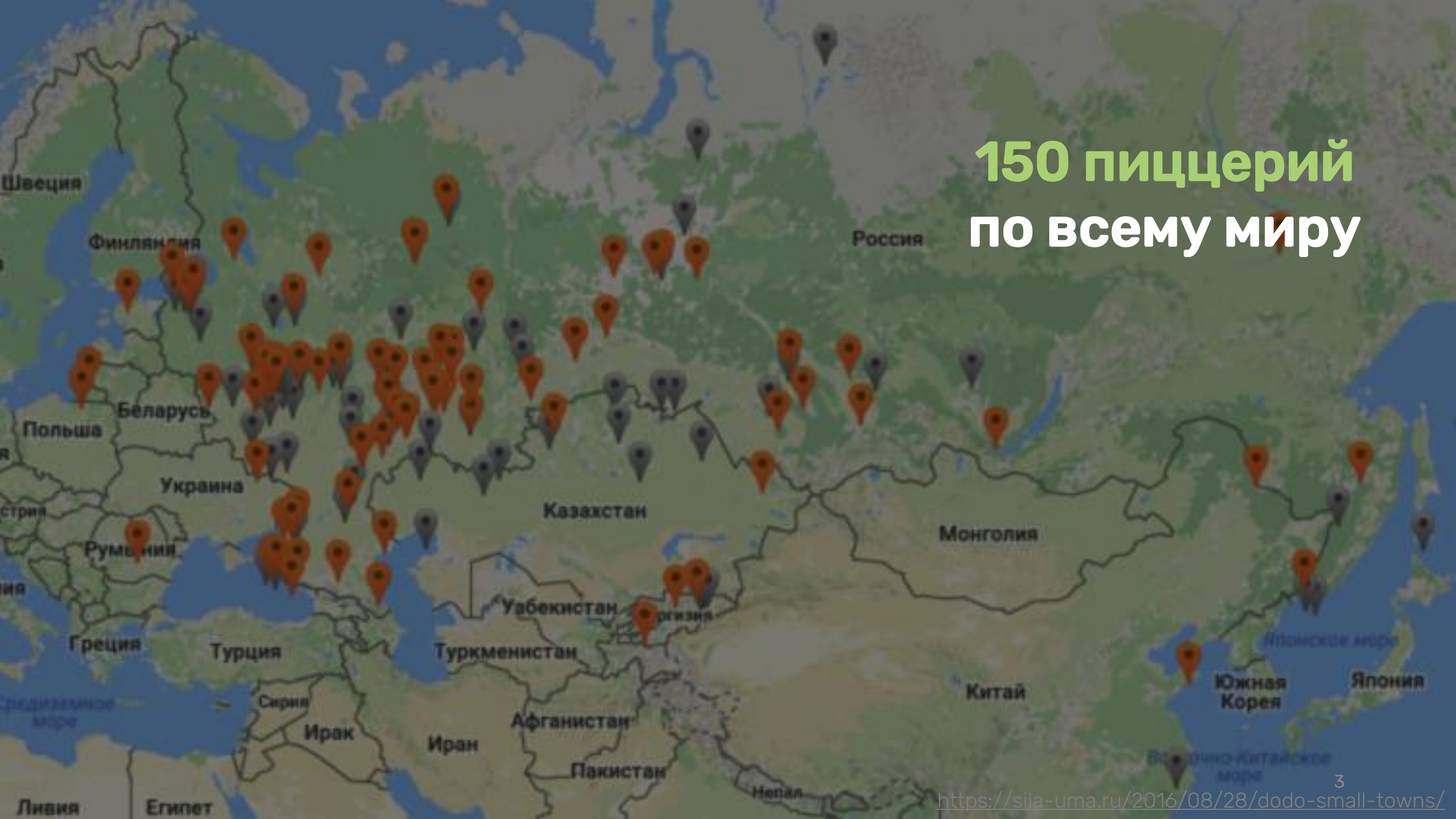



ДОДО ПИЦЦА

ДОСТАВКА

23 декабря 2016 г.
Неделя до Нового года...

150 пиццерий по всему миру





25 декабря
открытие пиццерии
в Санкт-Петербурге



Офтоп

Alina Tolmacheva

24 дек 2016

👁 30 746

«Додо Пицца» по ошибке перечислила клиентам 10 млн рублей за уже оплаченные заказы

Генеральный директор «Додо пицца» Федор Овчинников [рассказал](#) на своих страницах в соцсетях, что из-за технической ошибки российские клиенты сети получили 10 млн рублей за совершенные заказы, которые они уже оплачивали ранее. Пока Овчинников не знает, как решить ситуацию.

“

*... оказалось, что фоновая задача смотрит ...
на **реальное** подключение к Яндекс.Кассе.*

*... при этом фоновая задача смотрела
на версию **тестовой** базы.*

Хабр



Как в «Додо Пицца» потеряли 8 миллионов за один час из-за технической ошибки, а потом вернули

<https://habr.com/ru/company/yamoney/blog/325762/>



Андрей Арефьев

Руководитель направления
электронной коммерции
«Додо Пицца»

*Особенно обидно было осознавать,
что мы вернули деньги, которых
не получали – это были
тестовые заказы.*

<https://vc.ru/flood/21255-dodopizza-money-back>



*Что тесту хорошо,
то production'у – смерть!*





Редкое поведение



Mocks & stubs



Логирование



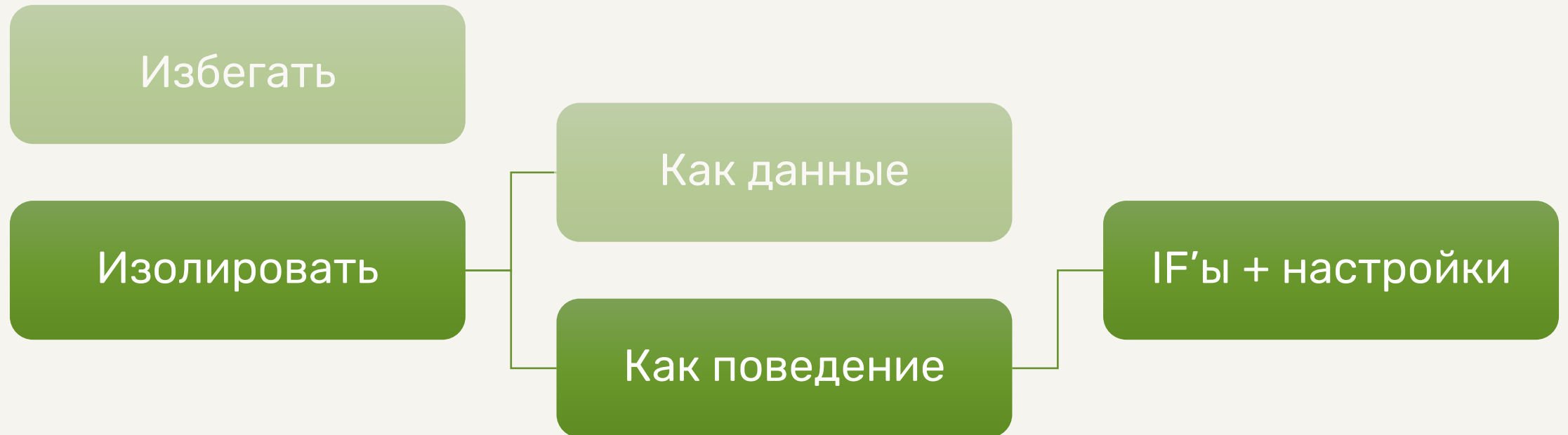
Антибезопасность



(добавь своё)

**Not for
production!**

И как быть?



IF'ы + настройки: Play! Framework

```
// Mode
try {
    mode = Mode.valueOf(configuration.getProperty("application.mode", "DEV").toUpperCase())
```

```
class DefaultMailSystemFactory extends AbstractMailSystemFactory {

    private static final MailSystem LEGACY MOCK MAIL SYSTEM = new LegacyMockMailSystem();
    private static final MailSystem PRODUCTION_MAIL_SYSTEM = new ProductionMailSystem();

    @Override
    public MailSystem currentMailSystem() {
        if (Play.useDefaultMockMailSystem()) {
            return LEGACY MOCK MAIL SYSTEM;
        } else {
            return PRODUCTION_MAIL_SYSTEM;
        }
    }
}
```

<https://github.com/playframework/play1/blob/master/framework/src/play/Play.java>

IF'ы + настройки: HotSpot JVM

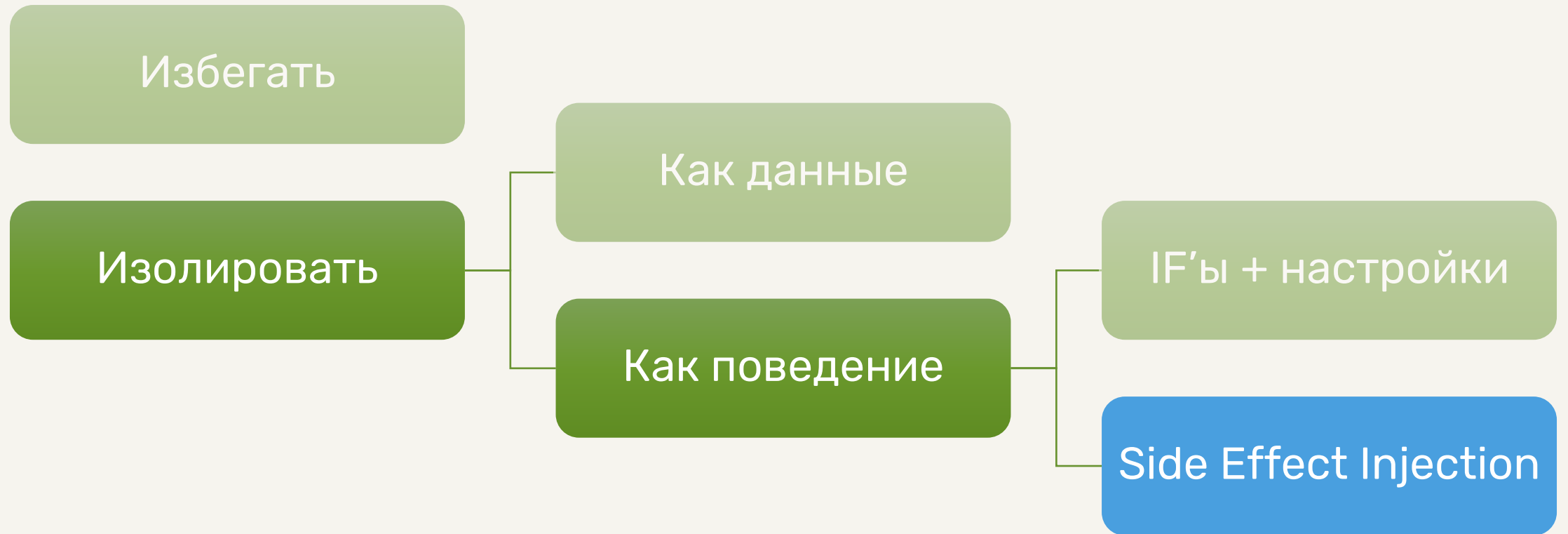
```
DEBUG_ONLY(if (ArchiveRelocationMode == 1 && use_requested_addr) {  
    // This is for simulating mmap failures at the requested address. In debug builds, we do it  
    // here (after all archives have possibly been mapped), so we can thoroughly test the code for  
    // failure handling (releasing all allocated resource, etc).  
    log_info(cds)("ArchiveRelocationMode == 1: always map archive(s) at an alternative address");  
    if (static_result == MAP_ARCHIVE_SUCCESS) {  
        static_result = MAP_ARCHIVE_MMAP_FAILURE;  
    }  
    if (dynamic_result == MAP_ARCHIVE_SUCCESS) {  
        dynamic_result = MAP_ARCHIVE_MMAP_FAILURE;  
    }  
});
```

<http://hg.openjdk.java.net/jdk/jdk14/file/6c954123ee8d/src/hotspot/share/memory/metaspaceshared.cpp>

IF'ы + настройки: что не так?

- Применимы **не везде** (например, в библиотеках)
- Нужно предусматривать **заранее**
- Легко **накосячить** в production

И как быть?



Ключевая идея Side Effect Injection




Писать исходный код как обычно, а тестовое поведение **внедрять извне** и только тогда, когда **это нужно**.



Java Virtual Machine Tool Interface


JVM TI позволяет трансформировать байт-код классов во время их загрузки.

- <https://docs.oracle.com/en/java/javase/16/docs/specs/jvmti.html>
- <https://habr.com/ru/company/odnoklassniki/blog/458812/>



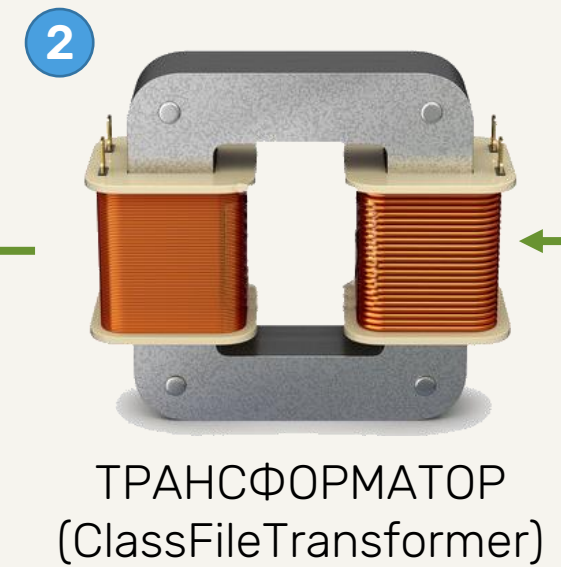
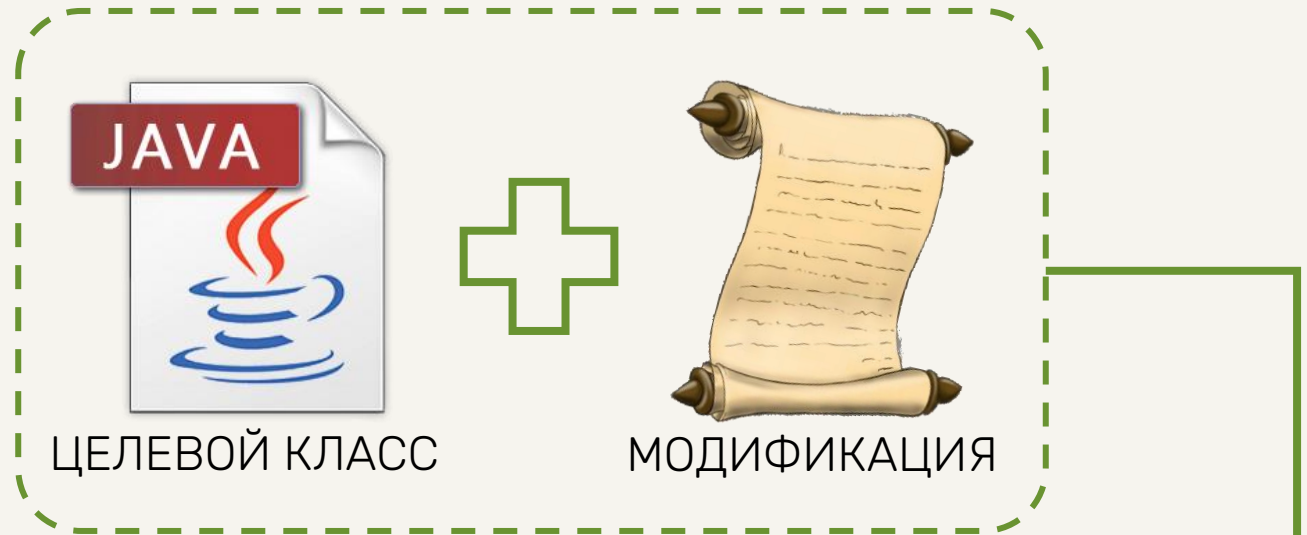
Андрей Паньгин
Одноклассники

JVM TI:
как сделать «плагин»
для виртуальной машины



Side Effect Injection

На пальцах



1 Java agent

MyJavaAgent.java

```
package com.example;
public class MyJavaAgent {
    public static void premain(String agentArgs, Instrumentation inst) {
        // ...
    }
}
```

myagent.jar!/META-INF/MANIFEST.MF

```
Manifest-Version: 1.0
Premain-Class: com.example.MyJavaAgent
```

```
$ java -javaagent:myagent.jar=arg1,arg2,...
```

2 Трансформатор

ClassFileTransformer.java

```
package java.lang.instrument;

public interface ClassFileTransformer {
    byte[]
        transform(   ClassLoader      loader,
                    String            className,
                    Class<?>         classBeingRedefined,
                    ProtectionDomain protectionDomain,
                    byte[]           classfileBuffer)
        throws   IllegalClassFormatException;
}
```



Принцип один.
Реализаций много.



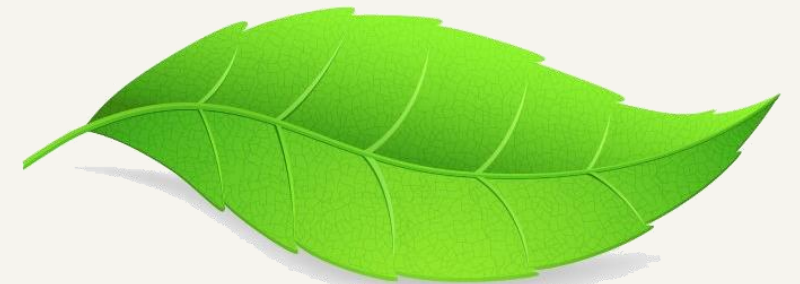


AspectJ



Byteman

Принцип один.
Реализаций много.



jMint

Eclipse AspectJ



Что?

Аспектно-ориентированное **расширение** Java

С версии 5 можно писать и на «чистой» Java

Когда?

Создан в **2001 г.** компанией PARC

Кто?

Живёт под зонтиком **Eclipse Foundation**

“

AspectJ enables
clean modularization
of crosscutting concerns,
such as error checking and
handling, synchronization,
context-sensitive behavior,
performance optimizations,
monitoring and logging,
debugging support,
and multi-object protocols

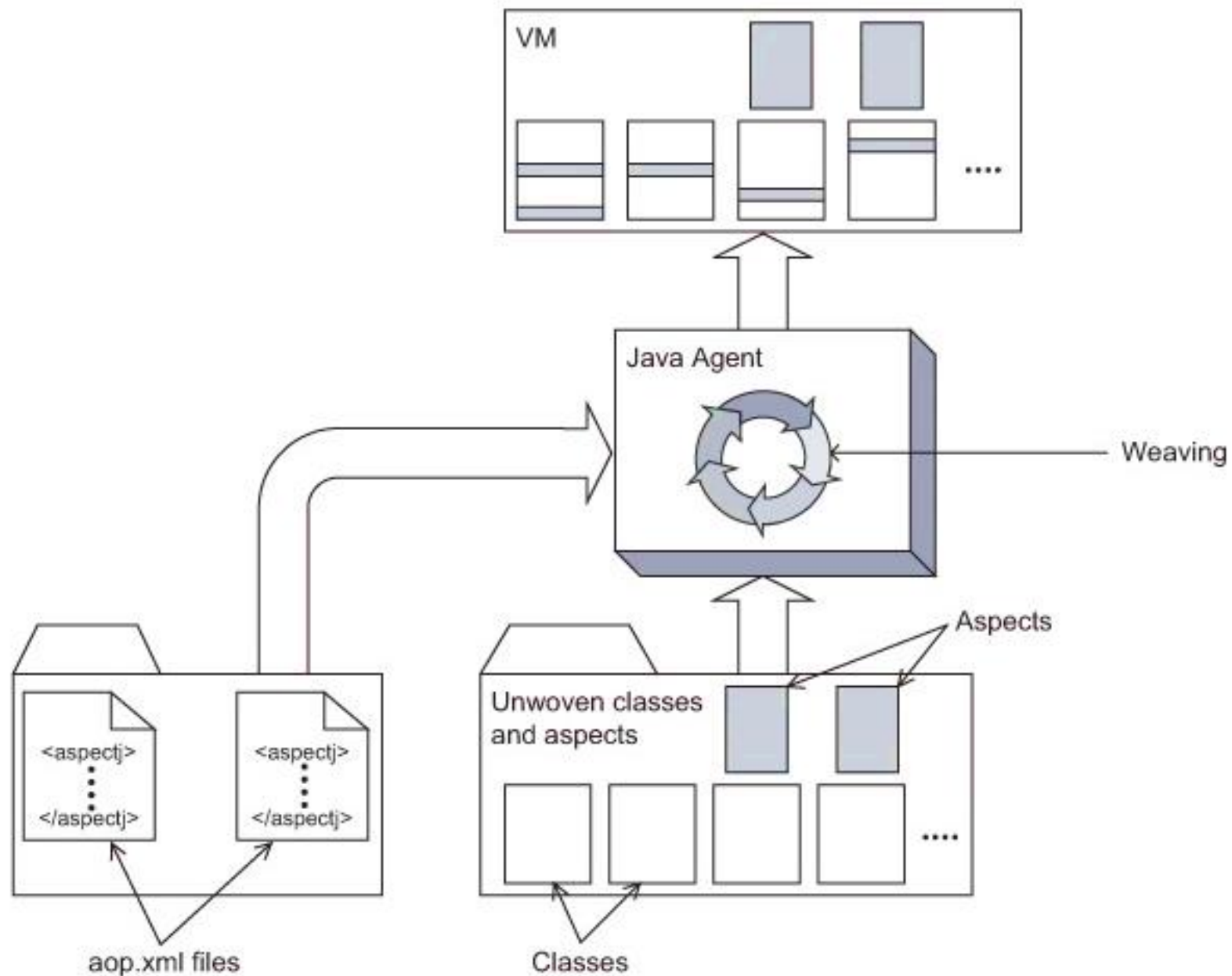
<https://www.eclipse.org/aspectj/>



Устройство AspectJ Weaver

По мнению авторов

Источник: книга **AspectJ in Action** (2nd Edition) глава **8.3.1**



Пора бы и код показать...





AspectJ: заметки

- 👍 Гибкий язык модификаций
- 👍 Отличная поддержка в IDE
- 👎 Нужно хранить вместе с целевым кодом
- 👎 Нужно тянуть `aspectjrt.jar` к себе в classpath



JBoss ByteMan



Что?

Инструмент для трассировки, мониторинга, **отладки** и **тестирования** приложений на Java

Когда?

Разрабатывается с **2009 г.** как один из проектов JBoss

Кто?

Спонсируется **Red Hat**

“

It injects Java code into your application ... without the need for you to recompile, repackage or even redeploy your application.

<https://byteman.jboss.org/>



Пора бы и код показать...

ByteMan: заметки



- 👍 Легко подключается к приложению
- 👍 Умеет править классы в `java.lang` и т.п.
- 👎 Нет поддержки в IDE
- 👎 Самобытный язык модификаций



jMint



Что?

Инструмент для внедрения **отладочных** и **тестовых** поведений в Java приложения

Когда?

Создан в **2016 г.** как личный R&D проект

Кто?

Активно применяется в **ЦФТ**
(Дирекция prepaid карт)

“

Модификации должны описываться так же, как если бы мы делали их прямо в исходном коде.

[придумал только что](#)

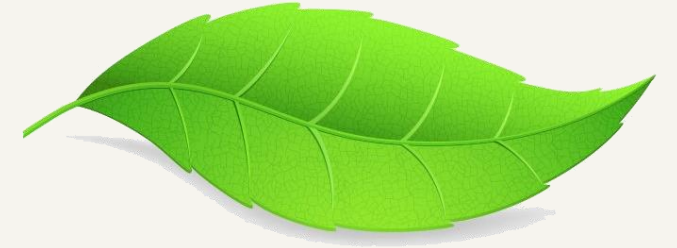


Дроплет – имя модификации в jMint

- Версия целевого класса, содержащая **только модифицирующий код**
 - Код компилируется библиотекой Javassist (\approx **JDK 1.5**)
- **Игнорирует** модификаторы доступа, поля, наследование, аннотации и любые непомеченные методы
- Может иметь в имени суффикс **DropLet** или **_DropLet**
- Может создаваться **2-мя способами**:
 - **с нуля**: ничего лишнего, но надо писать руками
 - **из копии**: «ломать не строить», но остается много шума

А вот как это работает...

jMint: заметки



- 👍 Синтаксис похож на целевой класс
- 👍 Модификации можно хранить где угодно
- 👎 Ущербный язык модификаций
- 👎 Неочевидность отличий от целевого класса



- Status
- Changes
- Workspace
- Build with Parameters
- Configure
- Delete Project
- Rebuild Last
- Favorite
- Move
- Job Config History
- Rename

Project upc2sand

Подключить дроплеты:	
<input checked="" type="checkbox"/> core/AuthenticateMP.java	
<input type="checkbox"/> core/BookerImpl.java	
<input type="checkbox"/> core/CDASClientImpl.java	
<input type="checkbox"/> core/CardUpc.java	
<input checked="" type="checkbox"/> core/CardUpc_isFirstLogin_Tur	
<input type="checkbox"/> core/CommonAbstractAmqpP	
<input type="checkbox"/> core/Consumer.java	
<input type="checkbox"/> core/ControlRateDecrease.java	
<input type="checkbox"/> core/ControlRateIncrease.java	
<input type="checkbox"/> core/CoreContextImpl.java	
<input type="checkbox"/> core/CoreContextImpl_RT.java	Дроплет предназначен для имитации работы через мобильное приложение. Это необходимо, например, при тестировании
<input type="checkbox"/> core/Crypter.java	Отключает шифрование паролей
<input type="checkbox"/> core/EquitiesProvider.java	Исключает тип баланса BNS_DEBT для мобильных платформ.
<input checked="" type="checkbox"/> core/EurekaInstanceConfigProvider.java	Принудительно включает регистрацию комплекса в Eureka по IP-адресу вместо hostName.
<input type="checkbox"/> core/FRAMOSClientImpl.java	Подменяет статус проверки платежа на фрод после получения ответа от FRAMOS
<input type="checkbox"/> core/LimitsManagerImpl.java	Подменяет поход в Limon на заведомо успешную проверку лимитов с фейковым ID проверки.
<input type="checkbox"/> core/LimitsSyncService.java	Подменяет номер счета для загрузки лимитов на заведомо корректный
<input type="checkbox"/> core/LoyaltyApiClientImpl.java	При загрузке бонусных балансов с использованием REST-API подменяет ean на зарегистрированный в Лояльности.
<input type="checkbox"/> core/OperationEnricher.java	Логирует сообщения об элементах Истории, игнорируемых при загрузке в Elastic
<input type="checkbox"/> core/ParameterType.java	При переводах ЗК обеспечивает передачу в ДП фиксированного курса валюты
<input type="checkbox"/> core/PayPassManagerImpl.java	@cutpoint INSTEAD
<input type="checkbox"/> core/PcvrClientImpl.java	Подменяет ответ PCVR
<input type="checkbox"/> core/QpayAdapterImpl.java	Для 2960264879676 формирует ответ от ДП о превышении лимита и формирует ответ от ДП для входящего перевода ЗК ДП для oid=246886901
<input type="checkbox"/> core/QpayClientImpl.java	Возвращает ошибку об аресте карты пользователя при попытке заказать раурасс
<input type="checkbox"/> core/QpayClientImplLimitExceeded.java	Возвращает ошибку о превышении максимальной суммы перевода при попытке "Получить наличные" с валютного кошелька (больше или равно 4400 \$/€)
<input type="checkbox"/> core/QpayClientImpl_Block_Arrested.java	Возвращает ошибку об аресте карты по ФЗ-167 пользователя при попытке входа
<input type="checkbox"/> core/QpayClientImpl_TransStatus.java	Подменяет статусы указанных по oID адресных денежных переводов
<input type="checkbox"/> core/QpayClientImpl_TransStatus_RT.java	Подменяет статусы указанных по oID адресных денежных переводов
<input type="checkbox"/> core/QpayModule.java	Подключает имитатор Денежных переводов (QPAY).
<input type="checkbox"/> core/QpayNameFormProviderDecrease.java	При переводах ЗК уменьшает курс до 40, обеспечивает видимость изменения курса при заполнении формы
<input type="checkbox"/> core/QpayNameFormProviderIncrease.java	При переводах ЗК увеличивает курс до 70, обеспечивает видимость изменения курса при заполнении формы
<input type="checkbox"/> core/QpayNameServiceBuilder.java	Добавляет на форму ДП по ФИО статическую надпись, чтобы эта форма явно визуально отличалась от остальных.
<input type="checkbox"/> core/QpayPhoneFormProvider.java	Для эмуляции ошибок по лимитам от ДП в DP3(QpayFault 4516, 4517, 4518, 4519, 4520 для переводов отправленных на указанные в дроплете ean-ы)

Droplets

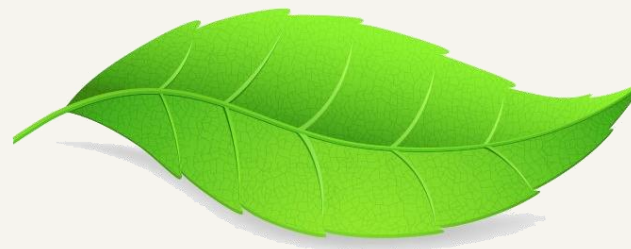
Подключить дроплеты:

- core/AuthenticateMP.java
- core/BookerImpl.java
- core/CDASClientImpl.java
- core/CardUpc.java

Build History trend ^

#112	Wed Jul 22 12:41:45 NOVT 2020
#111	Wed Jul 22 11:12:54 NOVT 2020
#110	Tue Jul 21 09:46:39 NOVT 2020
#109	

jMint: кишочки



<https://toparvion.pro/talk/2018/jbreak/>

A promotional banner for the 'jbreak 2018' event. On the left, there is a white wolf head logo next to the text 'jbreak 2018'. Below this, the name 'Владимир Плизга' is written in white, with 'ЦФТ' underneath. A white speech bubble contains the text 'Side Effect Injection, или Добродетельные КОСТЫЛИ'. On the right, a photograph shows a man in a blue shirt and red lanyard standing in front of a red backdrop with 'MOBI FEST' and 'ЦФТ' logos.

 **jbreak** 2018

Владимир Плизга
ЦФТ

Side Effect Injection,
или Добродетельные
КОСТЫЛИ



twitch.tv/HyperRPG



Сравнение инструментов

Свойство \ инструмент	AspectJ	ByteMan	jMint
Модификация приватных методов			
Отдельное хранение модификаций			
Поддержка в IDEA			
Тот же язык, что у целевого класса			
Чистый classpath			
Внедрение в классы JVM			
Добавление новых полей и методов			
Подключение "на лету"			

Сравнение инструментов

Свойство \ инструмент	AspectJ	ByteMan	jMint
Модификация приватных методов	✓	✓	✓
Отдельное хранение модификаций	–	✓	✓
Поддержка в IDEA	✓	–	✓ ¹
Тот же язык, что у целевого класса	✓ ²	–	✓
Чистый classpath	–	✓	✓
Внедрение в классы JVM	✓ ³	✓	–
Добавление новых полей и методов	✓	–	–
Подключение “на лету”	–	✓	–

¹ Без учёта ограничений Javassist

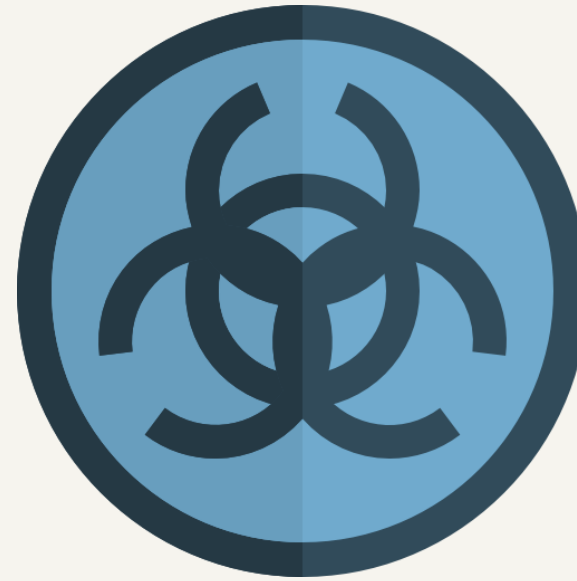
² Не считая «чистый» AspectJ и выражения в pointcut'ax

³ Кроме классов javaх.* (в них можно)

Щас будет график

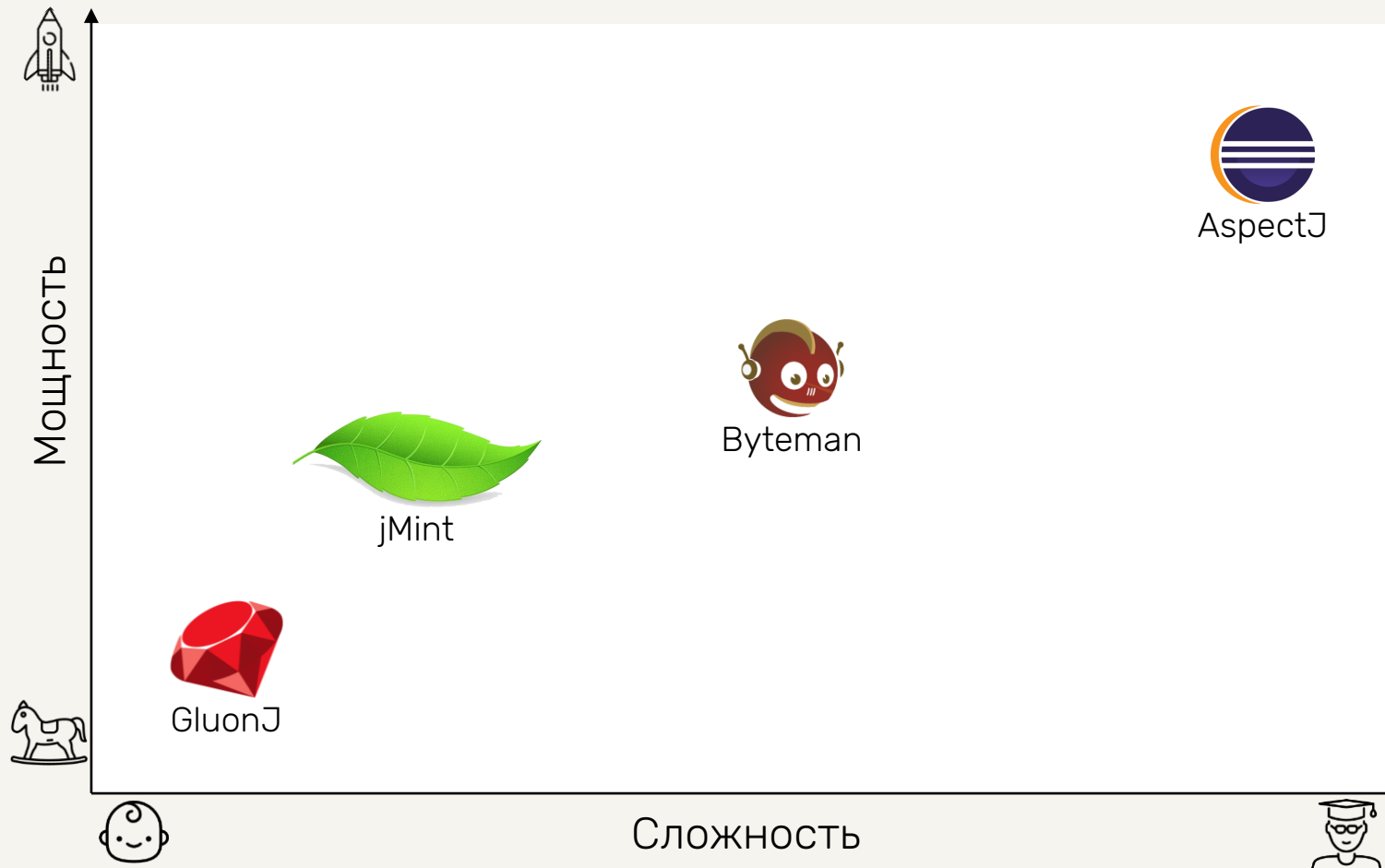


Вкусовщина



Субъективщина

Мощность **VS** сложность



Side Effect Injection: **напутствие**

- Держите «боевой» код и настройки **ЧИСТЫМИ**
- Делайте модификации **ПРОСТЫМИ**
 - *А если не получается, комбинируйте подходы*
- Храните модификации от «боевого» кода **ОТДЕЛЬНО**
 - *И будьте готовы к их устареванию 🤪*
- Применяйте SEI **ПО НАЗНАЧЕНИЮ**
 - *Оценивая альтернативы*



*Всё, что может пойти не так,
обязательно пойдёт не так.*

Закон Мерфи





AspectJ

www.eclipse.org/aspectj



Byteman

byteman.jboss.org



jMint

github.com/toparvion/jmint

Владимир Плизга
ЦФТ, PrePaid

 @toparvion

 Toparvion

 <https://toparvion.pro/>



Q&A

Источники материалов

- Photo by [William Felker](#) on [Unsplash](#)
- Photo by [Karolina Grabowska](#) from [Pexels](#)
- Photo by [Marc Schulte](#) on [Unsplash](#)
- Photo by [Brett Sayles](#) from [Pexels](#)
- <https://www.flaticon.com/authors/freepik>
- <https://www.flaticon.com/packs/archeology-40>