

# Проверка покрытия тестами схем Camunda



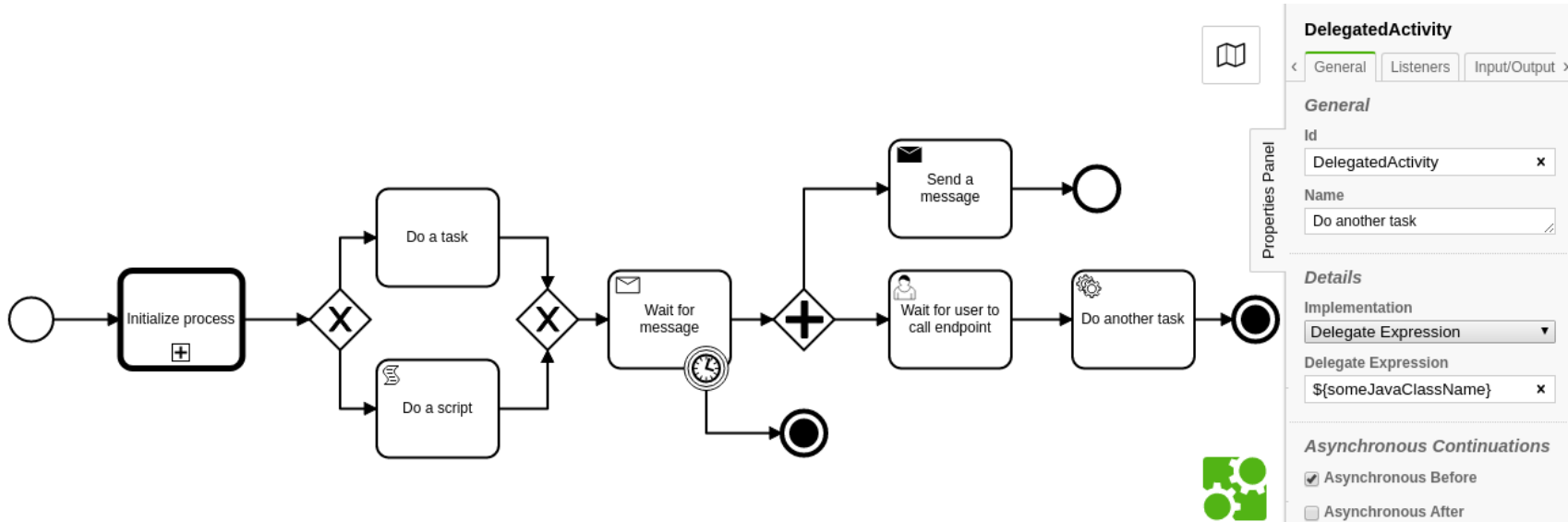
# TLDR: о чем

- Что было/что хотелось;
- Миграция на JUnit5;
- История страданий и пулл-реквесты;
- Что получилось в итоге.



# TLDR: Camunda

- BPMN-движок — оркестрация, делегаты...
- Рисуют аналитики, но это не Platformeco:)



# Что было

- Стрёмная инициализация

```
8 class CamundaCoverageSpringJUnit4ClassRunner(clazz: Class<*>) : SpringJUnit4ClassRunner(clazz) {
9     override fun withBeforeClasses(s: Statement): Statement {
10         testClass.getAnnotatedFields(ClassRule::class.java).forEach { it: FrameworkField!
11             when (it.type) {
12                 ProcessEngineRule::class.java -> {
13                     it.field.set(
14                         it,
15                         this.testContextManager.testContext.applicationContext.getBean(ProcessEngineRule::class.java)
16                     )
17                 }
18             }
19         }
20         return super.withBeforeClasses(s)
21     }
22 }
```

# Что было

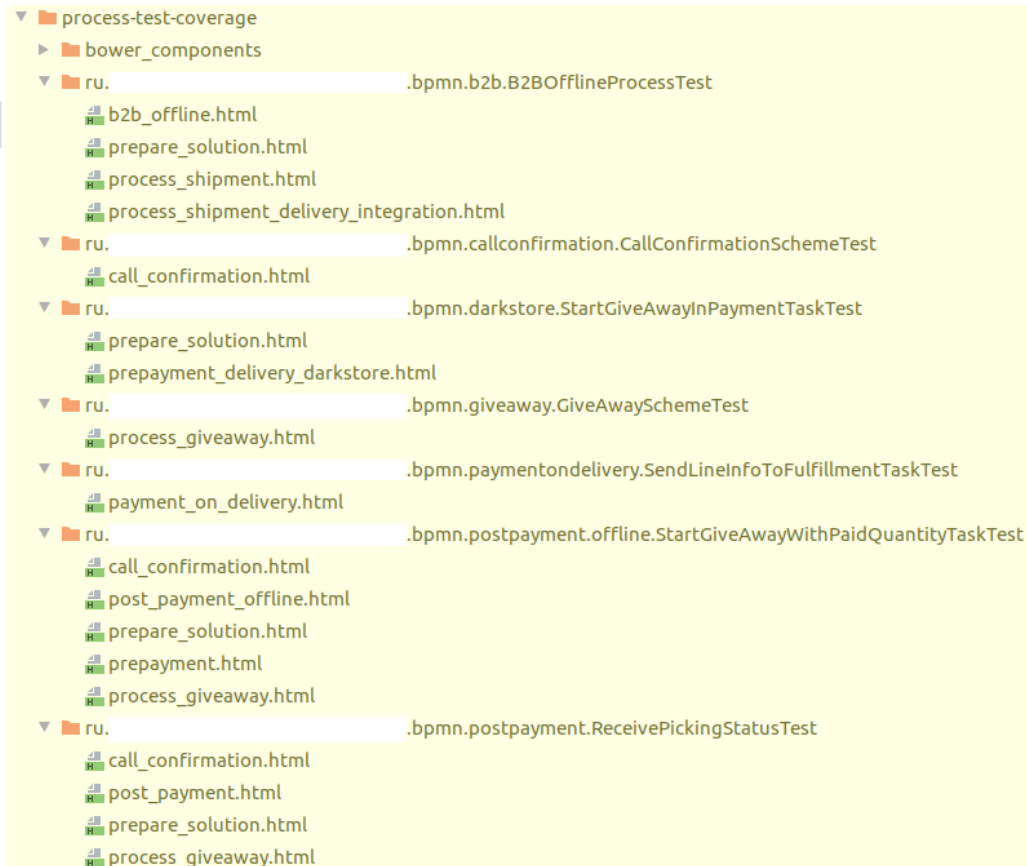
- Стрёмная инициализация;
- Сюиты: дублирование тестов

```
@RunWith(Suite::class)
@SuiteClasses(value = [
    ValidateRefundRequestTaskTest::class,
    RefundAmountAfterOrderCompletedTaskTest::class,
    EditSolutionFromRefundRequestTaskTest::class
])
class AllRefundBpmnCoverageTests
```



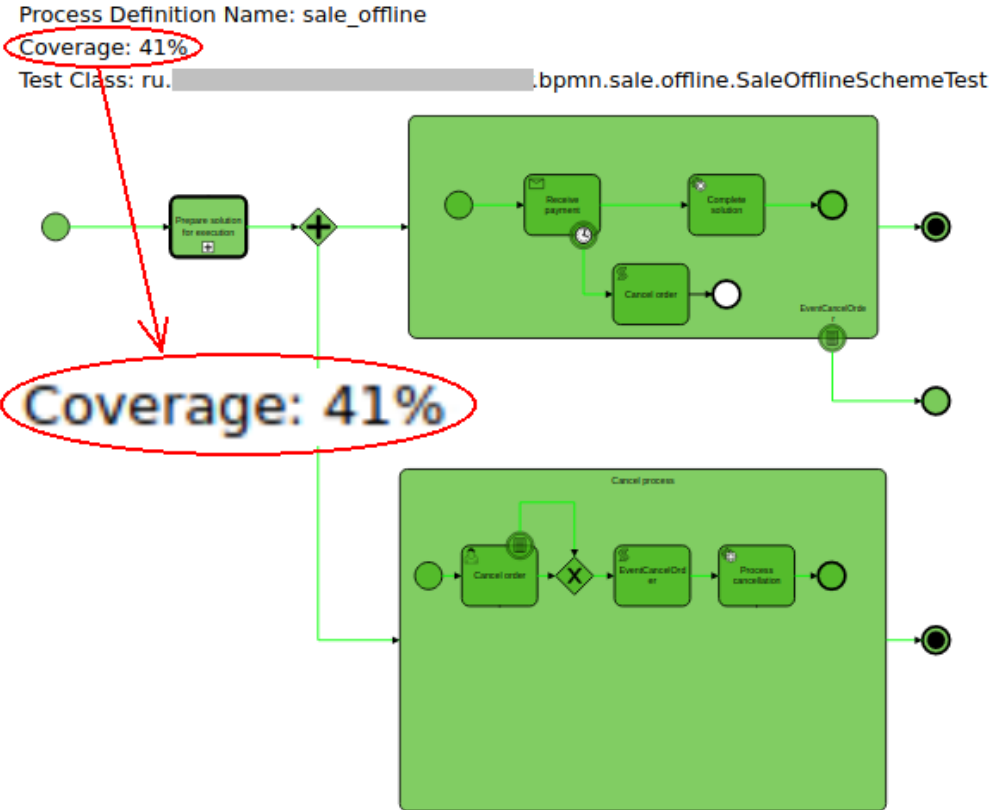
# Что было

- Стрёмная иници
- Сюиты: дублиро
- Склад HTML



# Что было

- Стрёмная инициализация
- Сюиты: дублированы
- Склад HTML;
- Ошибочные данные



# Что было

- Стрёмная инициализация;
- Сюиты: дублирование тестов;
- Склад HTML;
- Ошибочные данные;
- Никто этим не пользовался :(





# Что хотелось

- Агрегированные данные по всем схемам;
- Корректные данные!
- Чтобы тесты валились, если покрытие маленькое.



# Идея

1. Пошарить как-то состояние между классами и получить общие проценты;  
(возможно, считать самостоятельно)
2. Запустить тест в конце, чтобы проверить;
3. ???
4. PROFIT



# JUnit5

- `Store` — для хранения состояния;
- `TestExecutionListener` — для запуска кода после всех тестов.

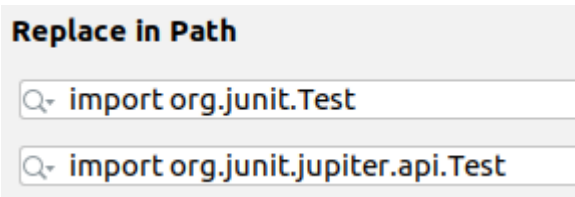


# Реализация



# Миграция на JUnit5

- Смена импортов — для 90% случаев



- @Before → @BeforeEach
- @BeforeClass → @BeforeAll



# Миграция на JUnit5

- Смена импортов — для 90% случаев;
- Rule → Extension

надо немного подумать



# Миграция на JUnit5

- Смена импортов — для 90% случаев;
- Rule → Extension;
- ловля исключений → `assertThrows`

```
val exception : BpmnError = assertThrows<BpmnError> {  
    | validatePaymentTask.execute(execution)  
}
```

```
assertThat(exception.message).contains("...")
```



# Миграция на JUnit5

- Смена импортов — для 90% случаев;
- Rule → Extension;
- ловля исключений → assertThrows;
- TLDR: изи. [Есть гайд для IntelliJ.](#)





# Нюансы тестов Camunda

- Гвоздями прибиты к JUnit4



# Нюансы тестов Camunda

- Гвоздями прибиты к JUnit4


**Пилишь ты такой тесты камунды на junit5...**  
**А тем временем у них в репе:**

**chore(engine): refactor junit3 test code to junit4**

- \* Migrate junit3 helper methods to ProcessEngineTestRule methods;
- \* Port test classes to JUnit4;
- \* Add missing dependencies after porting (e.g. JUnit Asserts);
- \* Refactor test code to Java 8 syntax where possible.
- \* Migrate authorization tests: Call authorization super.setUp after test setUp
- \* Make JdbcStatementTimeoutTest work with junit4
- \* Make CompetingHistoryCleanupAcquisitionTest work with junit4
- \* Migrate JPAVariableTest to junit4
- \* Refactor concurrency tests;

Related to [CAM-12149](#)

 master (#892)

 koevskinikola committed 5 days ago



# Нюансы тестов Camunda

- Гвоздями прибиты к JUnit4;
- JUnit5 — только в планах



yanavasileva commented 20 days ago

Member



The mentioned branch ([WIP branch test-junit5](#)) was a slack time project to explore the options, unfortunately, I didn't have time to continue the work on it.

As further information from the platform side, the following ticket has been created:

<https://jira.camunda.com/browse/CAM-11955>

We haven't started the work on it, please let me know in case you are interested in making a contribution.



# Нюансы тестов Camunda

- Гвоздями прибиты к JUnit4;
- JUnit5 — только в планах;
- Итого: адаптер JUnit4 → JUnit5 для ProcessEngineRule.



# Рефакторинг проверки покрытия

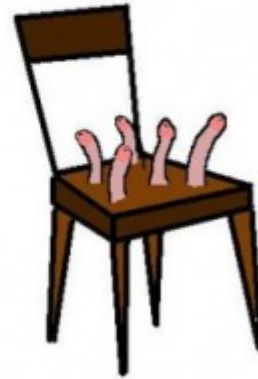


# Рефакторинг проверки покрытия

когда делаешь что-то нетривиальное  
с использованием библиотеки и  
нужная тебе вещь закопана в  
приватных методах



использовать  
"немного"  
рефлексии



скопипастить  
полбиблиотеки



# Рефакторинг проверки покрытия

Итого: PoC с MethodHandles, чтобы  
вызвать super родителя.

[ДАННЫЕ УДАЛЕНЫ]

Реализация с MethodHandle  
была настолько отвратительна,  
что была удалена из этой  
презентации, дабы не  
травмировать вашу психику



# Пулл-реквесты





# Пулл-реквесты

- Флаги для отключения лишних отчетов

<https://github.com/camunda/camunda-bpm-process-test-coverage/pull/56>



# Пулл-реквесты

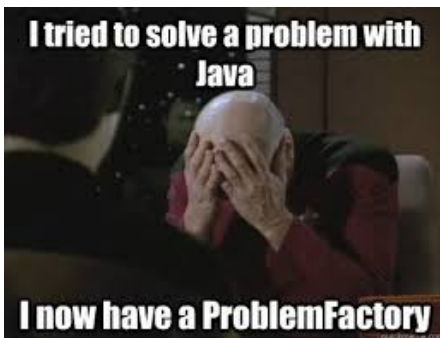
- Флаги для отключения лишних отчетов;
- Нормальные проценты в отчете
- <https://github.com/camunda/camunda-bpm-process-test-coverage/pull/55>



# Пулл-реквесты

- Флаги для отключения лишних отчетов;
- Нормальные проценты в отчете;
- Пошарить состояние

<https://github.com/camunda/camunda-bpm-process-test-coverage/pull/57>



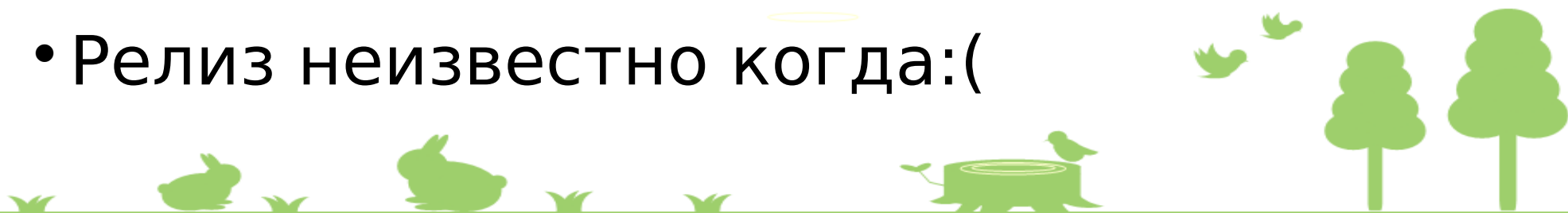
# Пулл-реквесты

- Флаги для отключения лишних отчетов;
- Нормальные проценты в отчете;
- Пошарить состояние;
- Все 3 приняли!



# Пулл-реквесты

- Флаги для отключения лишних отчетов;
- Нормальные проценты в отчете;
- Пошарить состояние;
- Все 3 приняли!
- Релиз неизвестно когда:(



# Рефакторинг проверки покрытия v2

- Listener глотает все исключения;
- Надо делать maven-таск;



# Рефакторинг проверки покрытия v2

- Listener глотает все исключения;
- Надо делать maven-таск, НО
  1. Это maven;



# Рефакторинг проверки покрытия v2

- Listener глотает все исключения;
- Надо делать maven-таск, НО
  1. Это maven;
  2. Надо думать о шаринге между запусками;





# Рефакторинг проверки покрытия v2

- Listener глотает все исключения;
- Надо делать maven-таск, НО
  1. Это maven;
  2. Надо думать о шагах запуска;
  3. Я ленивая жопа;



# Рефакторинг проверки покрытия v2

- Listener глотает все исключения;
- Надо делать maven-таск, НО
  1. Это maven;
  2. Надо думать о шаринге между за
  3. Я ленивая жопа;
  4. Поэтому запуск через Launcher.



# ИТОГО

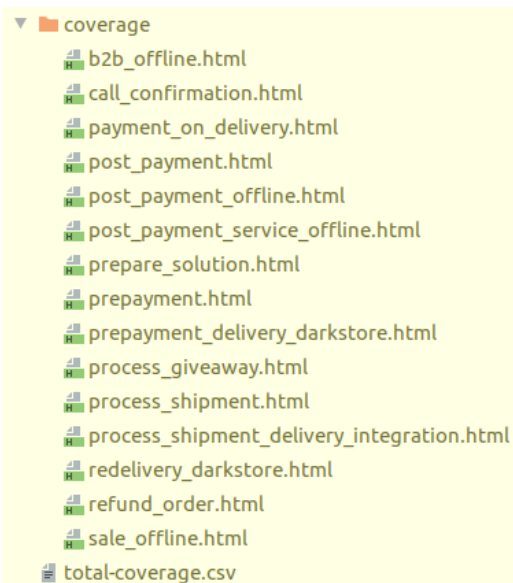
- Чуть менее стрёмная реализация

```
protected open fun createProcessEngineRule(processEngine: ProcessEngine) : ProcessEngineRule =  
    ProcessEngineRule(processEngine)  
  
override fun beforeAll(context: ExtensionContext) {  
    val springContext : ApplicationContext = getApplicationContext(context)  
    val processEngine : ProcessEngine = springContext.getBean(ProcessEngine::class.java)  
    processEngineRule = createProcessEngineRule(processEngine)  
    val description : Description = convertContextToClassDescription(context)  
    initProcessEngineRule(description, context)  
}
```



# ИТОГО

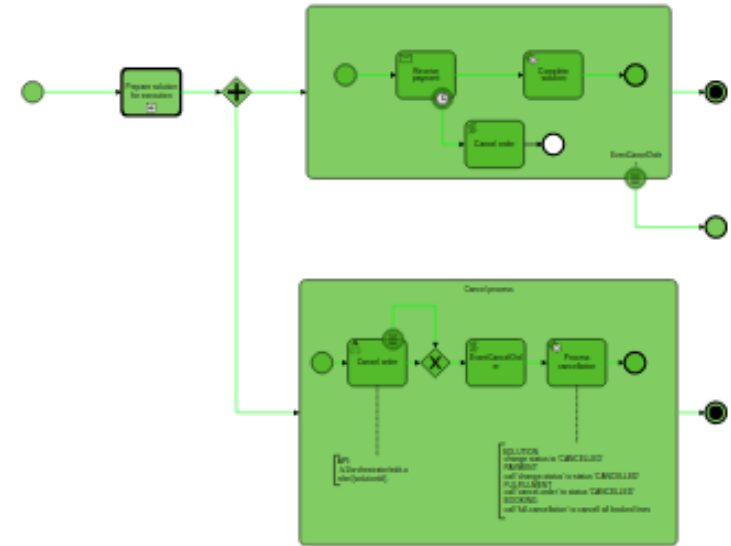
- Чуть менее стрёмная реализация;
- Все в одной папочке и итоговый отчет



# ИТОГО

- Чуть менее стрёмная реализация
- Все в одной папочке и итог
- Данные похожи на правду

Process Definition Name: sale\_offline  
Coverage: 95.1%



# ИТОГО

- Чуть менее стрёмная реализация;
- Все в одной папчке и итоговый отчет;
- Данные похожи на правду;
- Тесты валятся, когда все плохо

```
[ERROR] SchemesCoverageTest.assertMinimumCoverage [Coverage for process 'prepayment' is 56%, but expected minimum is 95%]
```

```
Expecting:
```

```
<0.5566502463054187>
```

```
to be greater than:
```

```
<0.95>
```

```
[INFO]
```

```
[ERROR] Tests run: 1017, Failures: 9, Errors: 0, Skipped: 0
```

```
[INFO]
```

```
[INFO] -----
```

```
[INFO] BUILD FAILURE
```



# ИТОГО

- Чуть менее стрёмная реализация;
- Все в одной папочке и итоговый отчет;
- Данные похожи на правду;
- Тесты валятся, когда все плохо;
- Несколько тикетов на улучшение покрытия.



# ИТОГО

- Чуть менее стрёмная реализация;
- Все в одной папочке и итоговый отчет;
- Данные похожи на правду;
- Тесты валятся, когда все плохо;
- Несколько тикетов на улучшение покрытия.

Вопросы — в  [v\\_o\\_chesnokov](#)

