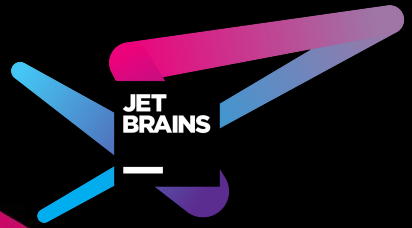


Типы автоматического тестирования в IntelliJ IDEA

Юрий Артамонов

Автор ты кто

Юрий Артамонов @jreznot

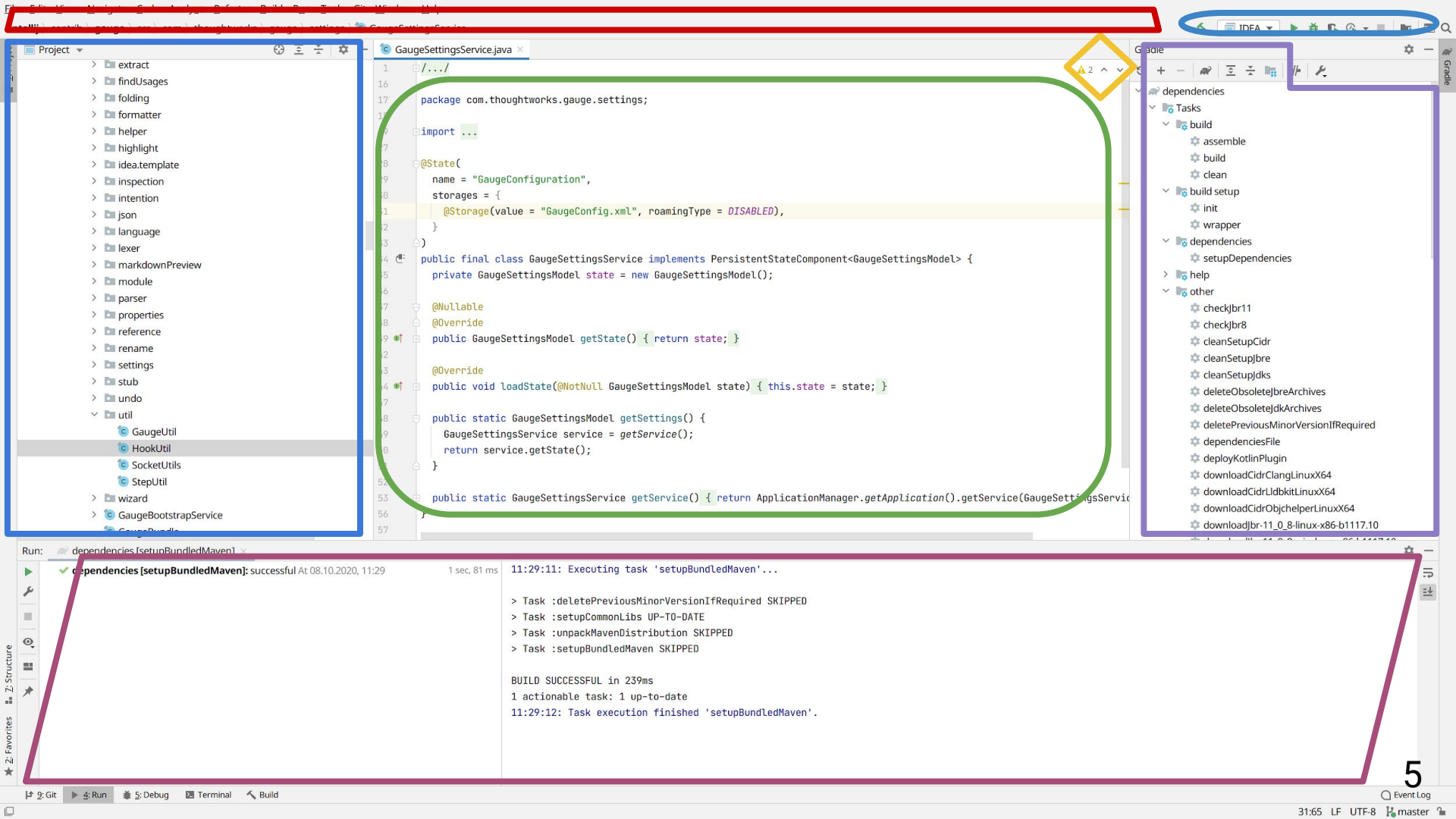
1. Разрабатывал фреймворки и библиотеки для Java > 10 лет
2. Придумываю новые возможности IDE для ваших любимых фреймворков в IntelliJ IDEA
3. Автор Selenium UI Testing плагина и мейнтейнер Gauge плагина для IntelliJ IDEA

План действий

1. Как устроены JetBrains IDE
2. Что тестировать в IDE
3. Unit & Functional тесты
4. Property-based тесты
5. UI тесты
6. Как работают команды IDE в JetBrains

Зачем вам слушать этот доклад?

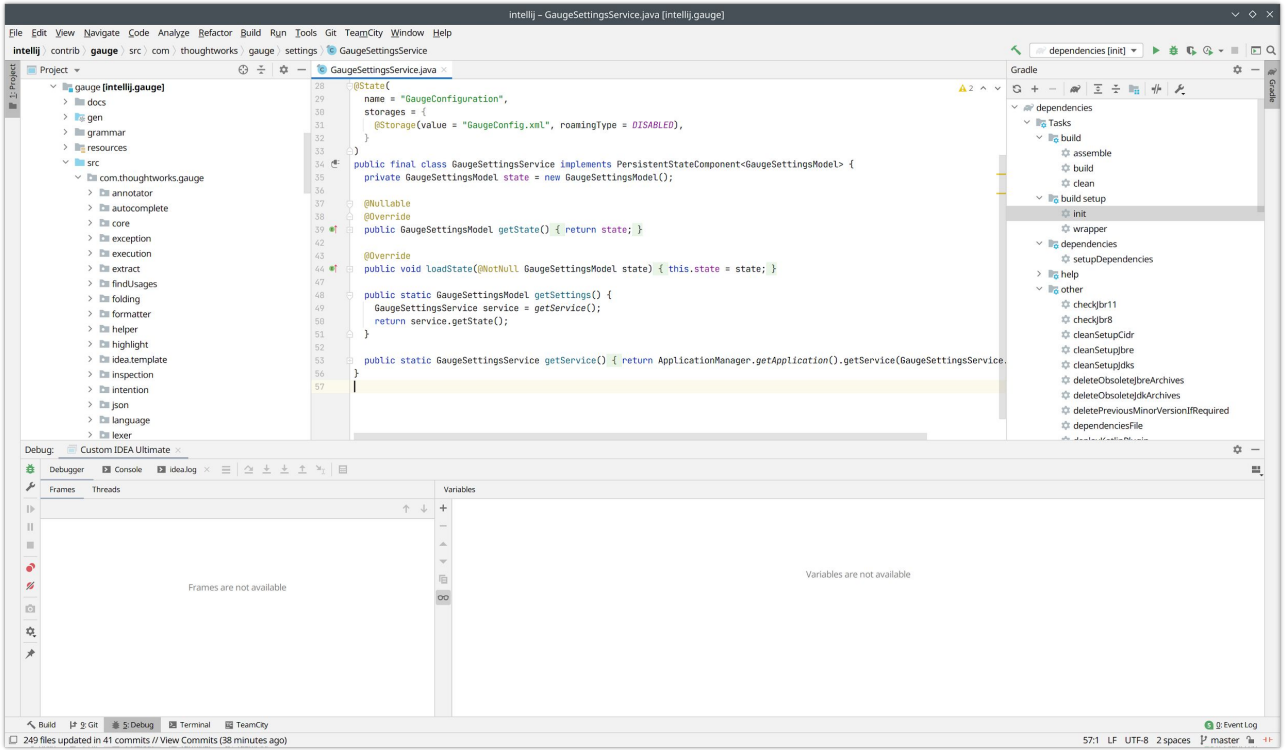
1. Разработать плагин для JetBrains IDE
2. Посмотреть, а как тестируют другие
3. Узнать что-то новое про устройство IDE


Project
extract
findUsages
folding
formatter
helper
highlight
idea.template
inspection
intention
json
language
lexer
markdownPreview
module
parser
properties
reference
rename
settings
stub
undo
util
GaugeUtil
HookUtil
SocketUtils
StepUtil
wizard
GaugeBootstrapService
GaugeSettingsService.java
/.../
package com.thoughtworks.gauge.settings;
import ...
@State(
name = "GaugeConfiguration",
storages = {
@Storage(value = "GaugeConfig.xml", roamingType = DISABLED),
}
)
public final class GaugeSettingsService implements PersistentStateComponent<GaugeSettingsModel> {
private GaugeSettingsModel state = new GaugeSettingsModel();
@Nullable
@Override
public GaugeSettingsModel getState() { return state; }
@Override
public void loadState(@NotNull GaugeSettingsModel state) { this.state = state; }
public static GaugeSettingsModel getSettings() {
GaugeSettingsService service = getService();
return service.getState();
}
public static GaugeSettingsService getService() { return ApplicationManager.getApplication().getService(GaugeSettingsServic
}
Gradle
dependencies
Tasks
build
assemble
build
clean
build setup
init
wrapper
dependencies
setupDependencies
help
other
checkJbr11
checkJbr8
cleanSetupCidr
cleanSetupJbre
cleanSetupJdks
deleteObsoleteJbreArchives
deleteObsoleteJdkArchives
deletePreviousMinorVersionIfRequired
dependenciesFile
deployKotlinPlugin
downloadCidrClangLinuxX64
downloadCidrLldbkitLinuxX64
downloadCidrObjchelperLinuxX64
downloadJbr-11_0_8-linux-x86-b1117.10
Run:
dependencies [setupBundledMaven]
dependencies [setupBundledMaven]: successful At 08.10.2020, 11:29
1 sec, 81 ms
11:29:11: Executing task 'setupBundledMaven'...
> Task :deletePreviousMinorVersionIfRequired SKIPPED
> Task :setupCommonLibs UP-TO-DATE
> Task :unpackMavenDistribution SKIPPED
> Task :setupBundledMaven SKIPPED
BUILD SUCCESSFUL in 239ms
1 actionable task: 1 up-to-date
11:29:12: Task execution finished 'setupBundledMaven'.
9: Git
4: Run
5: Debug
Terminal
Build
Event Log
31:65
LF
UTF-8
master

5

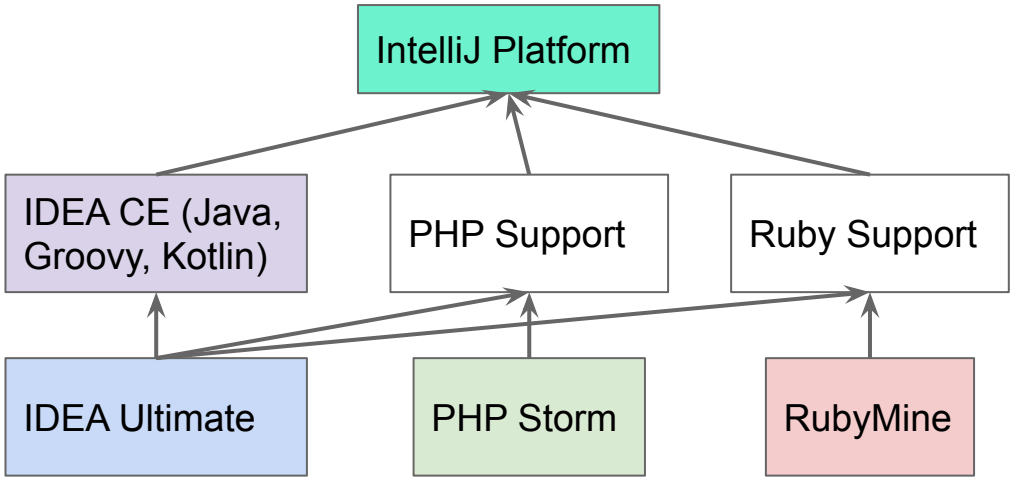
JetBrains IDE снаружи

- Menus
- Project View
- **Editor**
- Tool windows
- Settings
- Debugger
- VCS
- Status bar



JetBrains IDE изнутри

- Компоненты и сервисы
- Виртуальная файловая система (VFS)
- Поддержка языков (PSI)
- Редактор кода
- Инспекции
- Индексы
- Фоновые процессы
- UI библиотека
- Точки расширения



Типовые тестовые сценарии

1. Создание проекта
2. Редактирование кода
3. Рефакторинг
4. Взаимодействие с VCS
5. Сборка
6. Запуск
7. Отладка

Модульные тесты и специфика IDE

1. Можно иногда
2. В IDE всё зависит от всего.

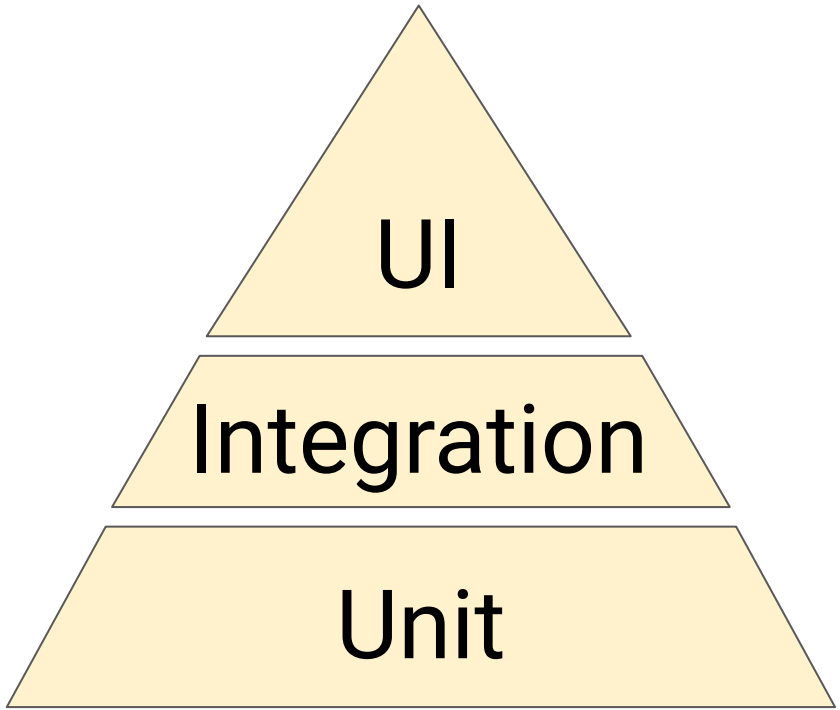
Пример: Selenium плагин зависит от Java, CSS, XPath, Docker, Maven и Gradle

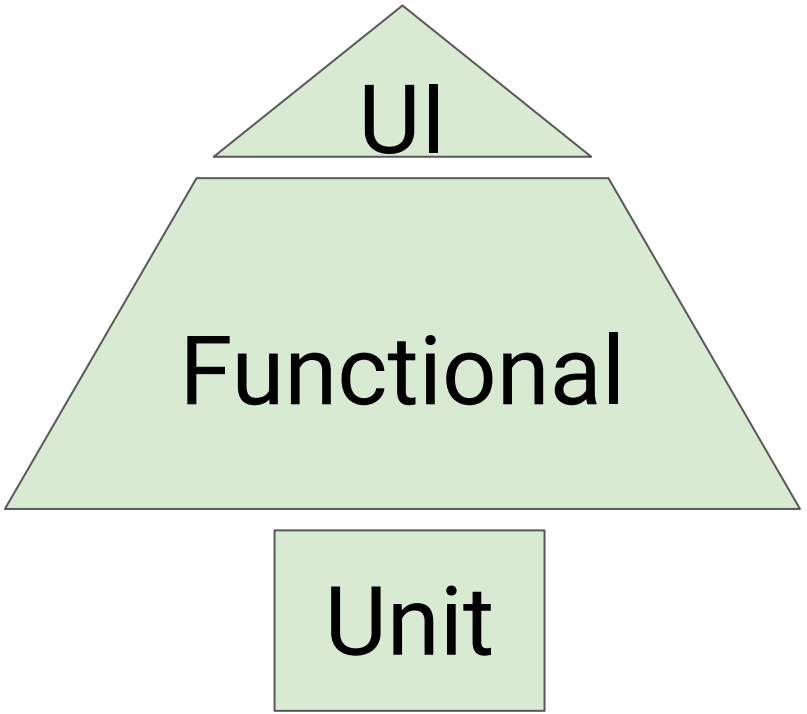
Функциональные тесты IDE

Какие тесты мы пишем чаще всего:

- Тесты запускаются в памяти без UI
- Тесты используют реальные реализации большинства подсистем IDE, за исключением UI компонентов
- Тесты проверяют один из сценариев использования целиком, а не отдельные функции и свойства реализации.

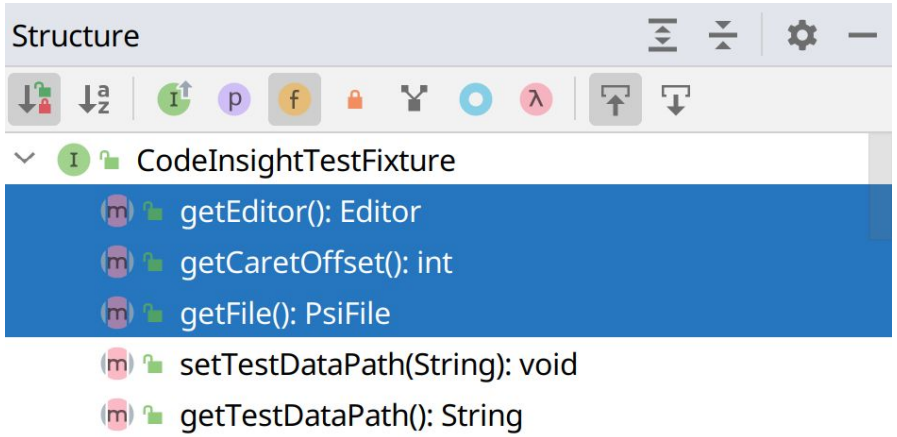
Пирамида или Ёлка





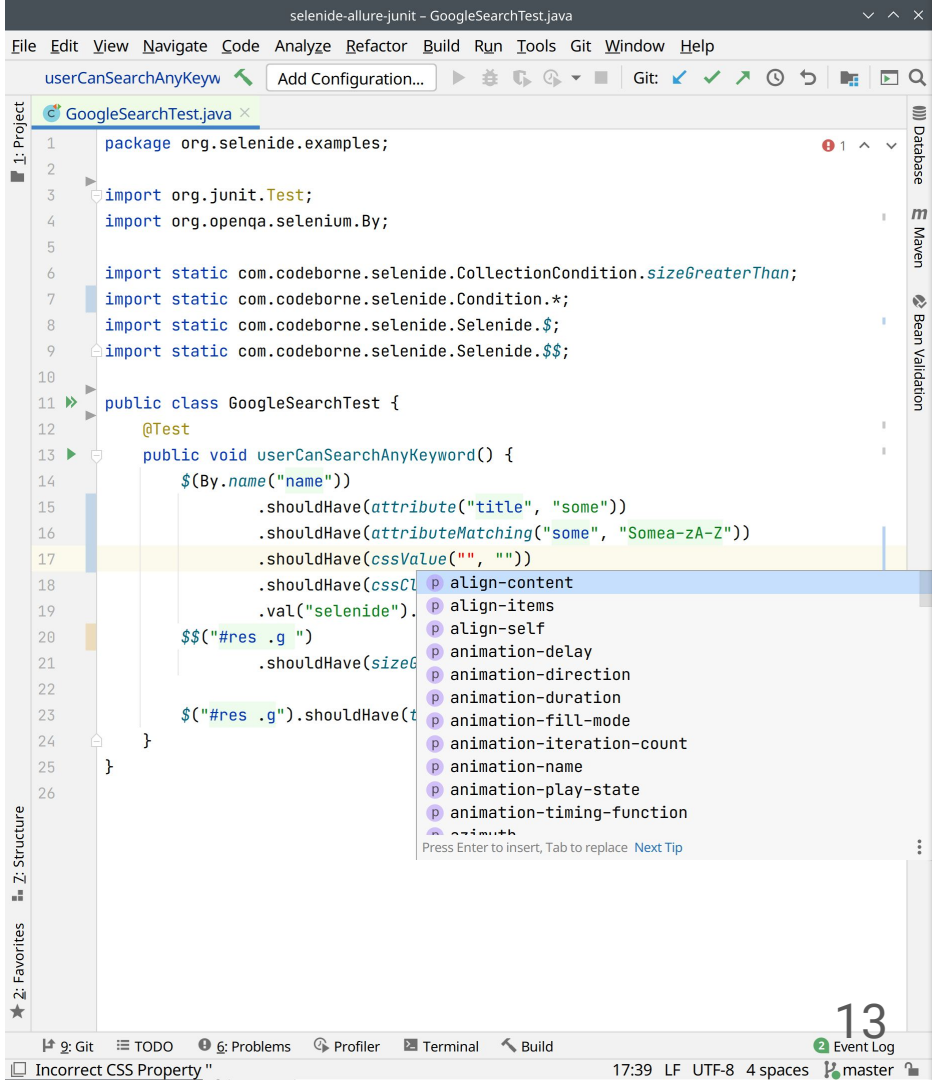
Fixture - модель IDE в памяти

- Тесты не тестируют UI,
  а работают с моделью IDE, как с интерфейсом.
- Большинство тестов принимают на вход файл/файлы, выполняют сценарий и сравнивают результат с эталоном.



На что писать функциональные тесты

1. Lexers / Parsers
2. Highlighting
3. Auto completion
4. Inspections
5. Intentions
6. References
7. Navigation / Find usages

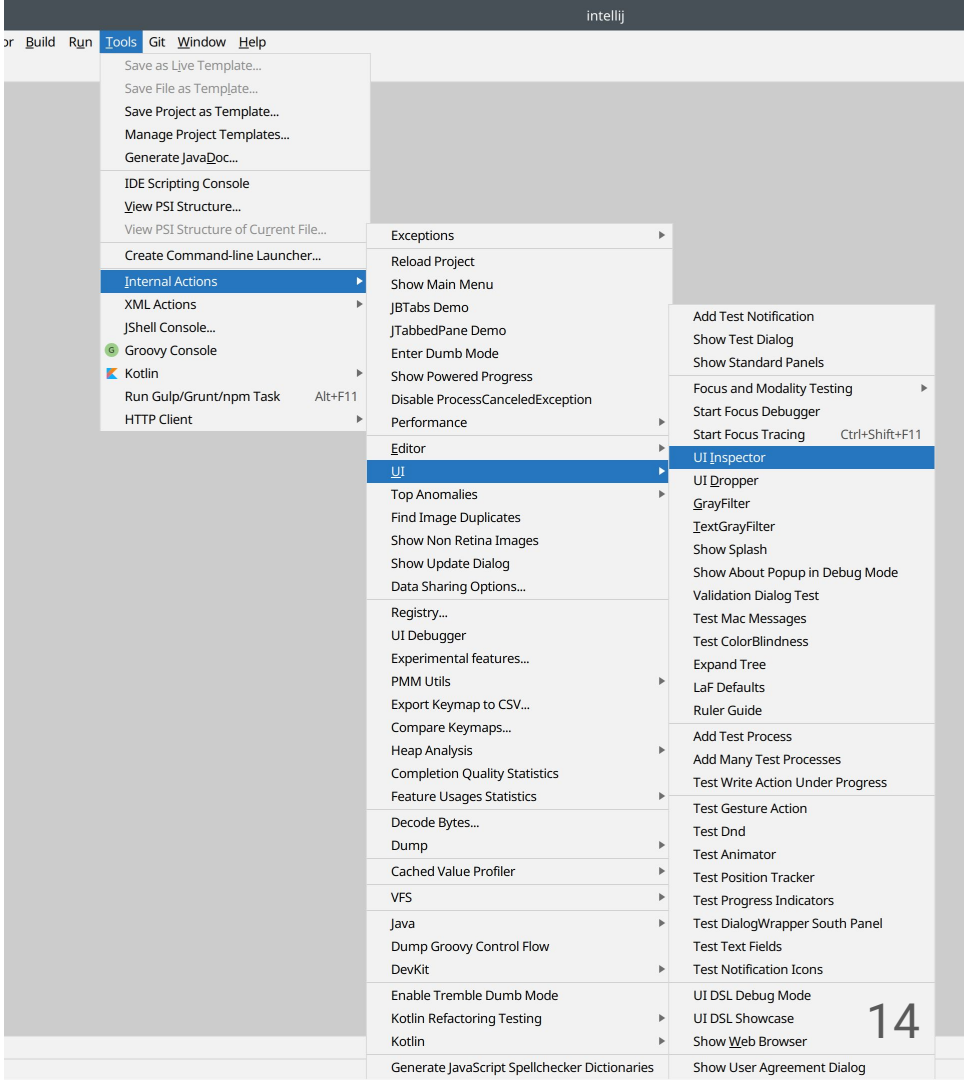


Internal Mode

Специальный режим работы IDE, в котором доступны опции для разработчиков.

- View PSI Structure of Current File
- Internal Actions
- UI Inspector
- Registry (только тссс…)

Включить в VM Options своей IDE:
-Didea.is.internal=true

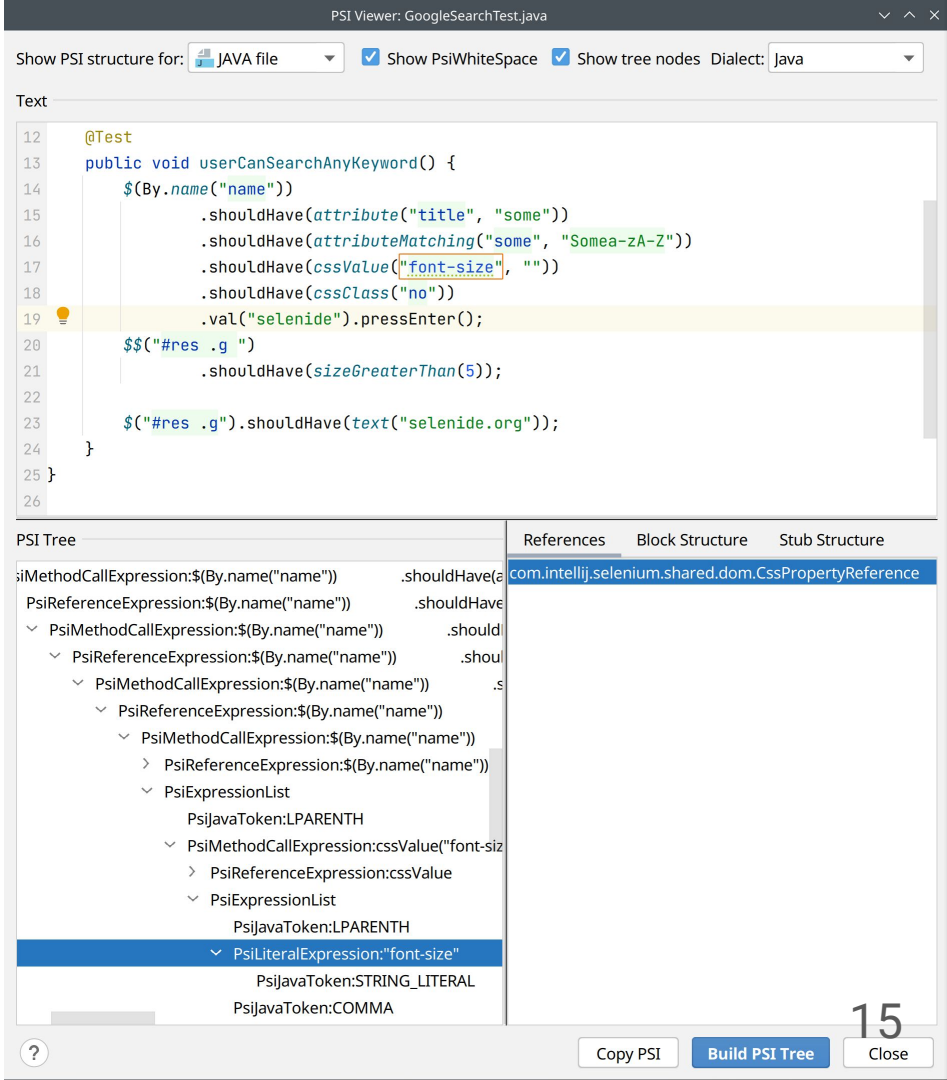


PSI Viewer

Tools - View PSI Structure of Current File

Позволяет изучить структуру PSI дерева в файле:

- Дерево PSI элементов
- Ссылки между элементами
- Встроенные фрагменты кода



Тестируем парсер

Базовый класс: `com.intellij.testFramework.ParsingTestCase`

1. Создаём тестовый метод:
   ```
   public void testSpecWithDataTable() { doTest(true); }
   ```

2. Добавляем исходный файл: `SpecWithDataTable.spec`

3. И файл с ожидаемой структурой кода: `SpecWithDataTable.txt`

Демо

Light and Heavy Tests

Мы разделяем интеграционные тесты на:

- Heavy Tests - создают новый проект на каждый тест
- Light Tests - переиспользуют проекты между запусками, когда возможно

В основном, мы рекомендуем писать Light Tests, чтобы сократить время исполнения.

Каким Mock-фреймворком пользуются в Intellij IDEA ?

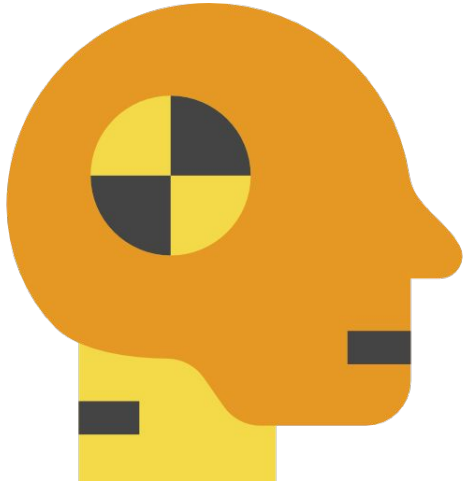
- JMockit
- Mockito
- EasyMock
- Custom

Никаким

Моки для тестирования плагинов

1. Моки сервисов IDE
2. Mock JDK
3. Классы из библиотек
4. Реальные зависимости

Демо



Тестирование зависимых плагинов

1. **Пример**: Spring плагин + Kotlin
   Два комплекта тестов с разными плагинами в ClassPath IDE.

2. **Пример**: Java тесты в IntelliJ CE и IntelliJ Ultimate

Property-based тесты

Алгоритм:

1. Подготовить / сгенерировать сценарий из случайных действий
2. На каждом шаге сценария выполнить проверку свойства
3. При падении теста подготовить сериализованное представление сценария для воспроизведения

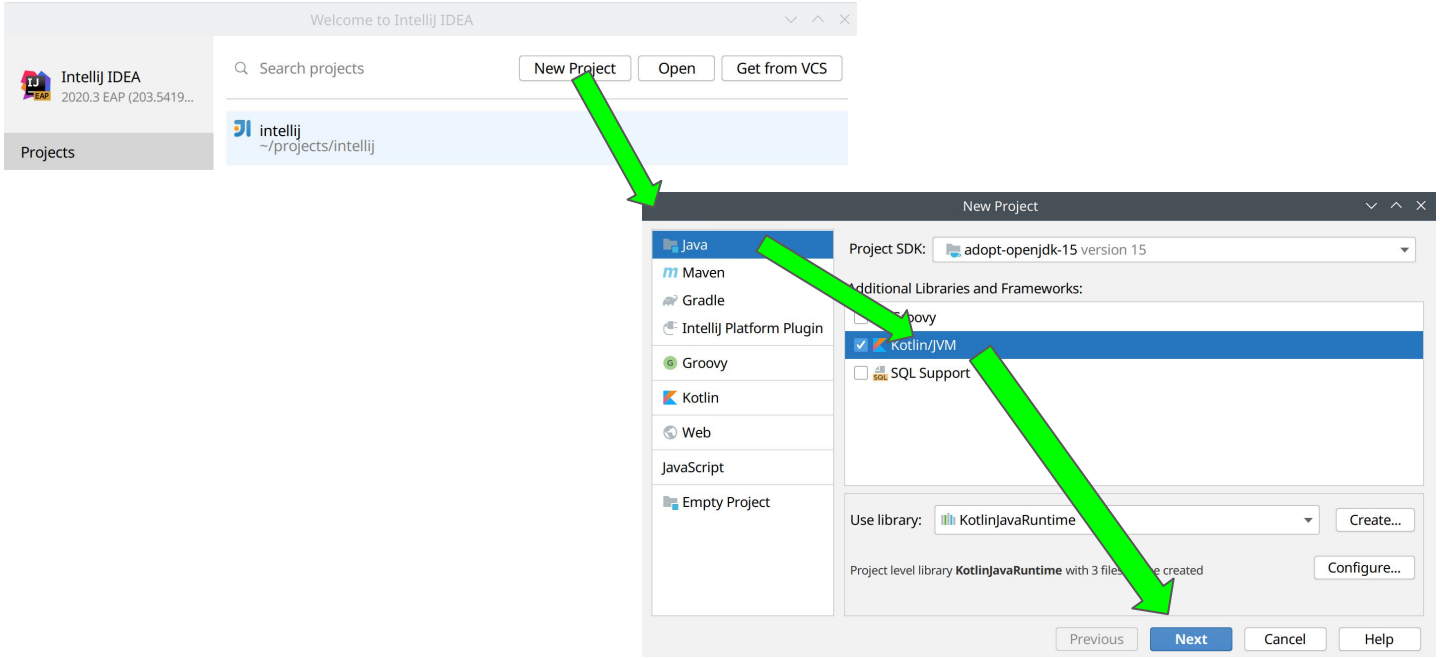
JetBrains/jetCheck

Примеры:

1. Вставить комментарии в случайные строки
2. Поменять тип всех методов на Object

E2E тестирование IDE

А как протестировать пользовательский сценарий целиком?



Selenium-like подход для тестирования UI

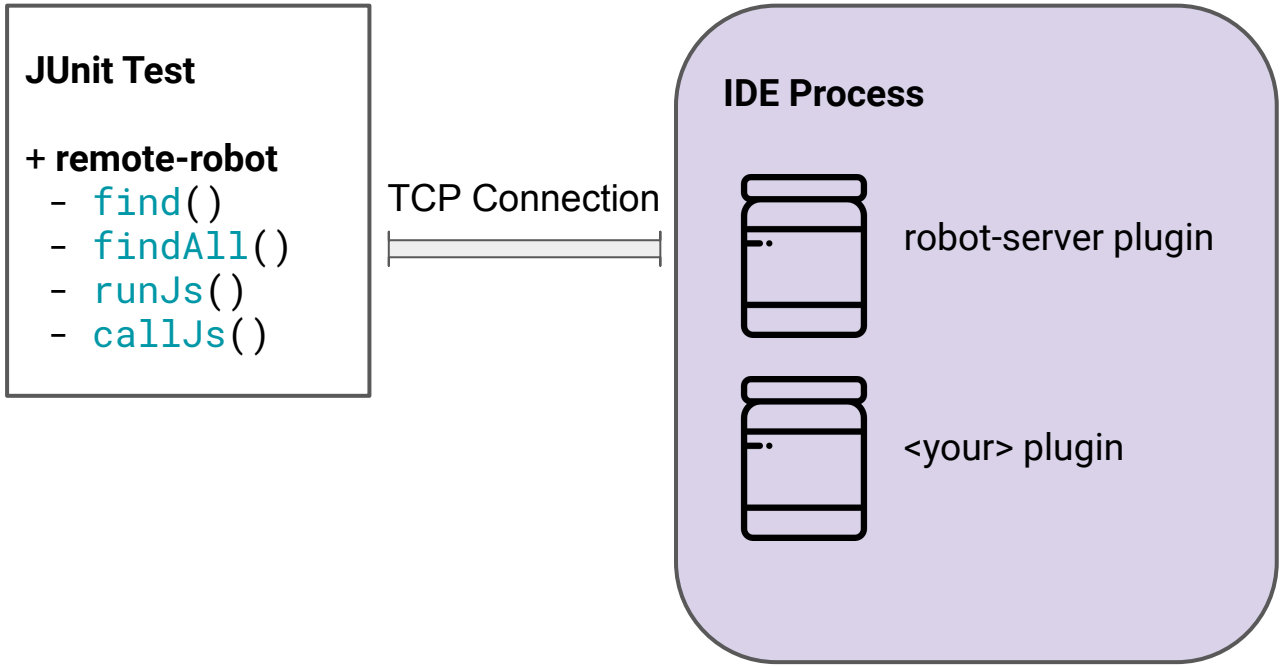
1. **Чем плохи готовые решения?**
   По сравнению с web, в desktop почти нет инструментов для автоматизации UI.

2. **Классический вариант тестирования Java UI:**
   `java.awt.Robot` в runtime приложения / FEST-Swing

JetBrains/intellij-ui-test-robot

Схема взаимодействия

RemoteRobot и IntelliJ IDEA plugin - наш WebDriver / W3C API.



Поиск элементов на экране

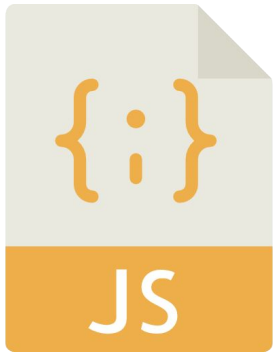
1. Поиск компонентов по XPath и атрибутам
2. Исполнение JavaScript

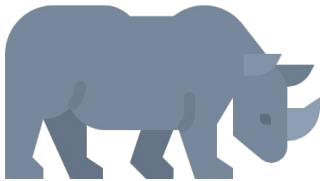
```
projectSteps.createUltimateProject { this: UltimateNewProjectDialogFixture
    setupSpringInitializr { this: SpringInitializrInitialViewFixture
        next.click()
        pause(1000)
        waitForIgnoringError { next.isEnabled() }
        step("Select java 8") {
            find<ComponentFixture>(byXpath("//div[translate(@accessiblename, 'JV', 'jv')='java version:' and @class='ComboBox']"))
                    .click()
            find<ComponentFixture>(byXpath("//div[@class='CustomComboPopup' and @name='ComboPopup.popup']"))
                    .findText("8").click()
        }
        next.click()
        dependencyDialog { this: DependenciesDialogFixture
            chooseWebStarter()
            assert(selectedDependenciesPanel).hasText("Spring Web")
        }
    }
```

Чем исполнить JavaScript на JVM

- Rhino (Mozilla) mozilla/rhino
- Nashorn Engine (JEP-174)

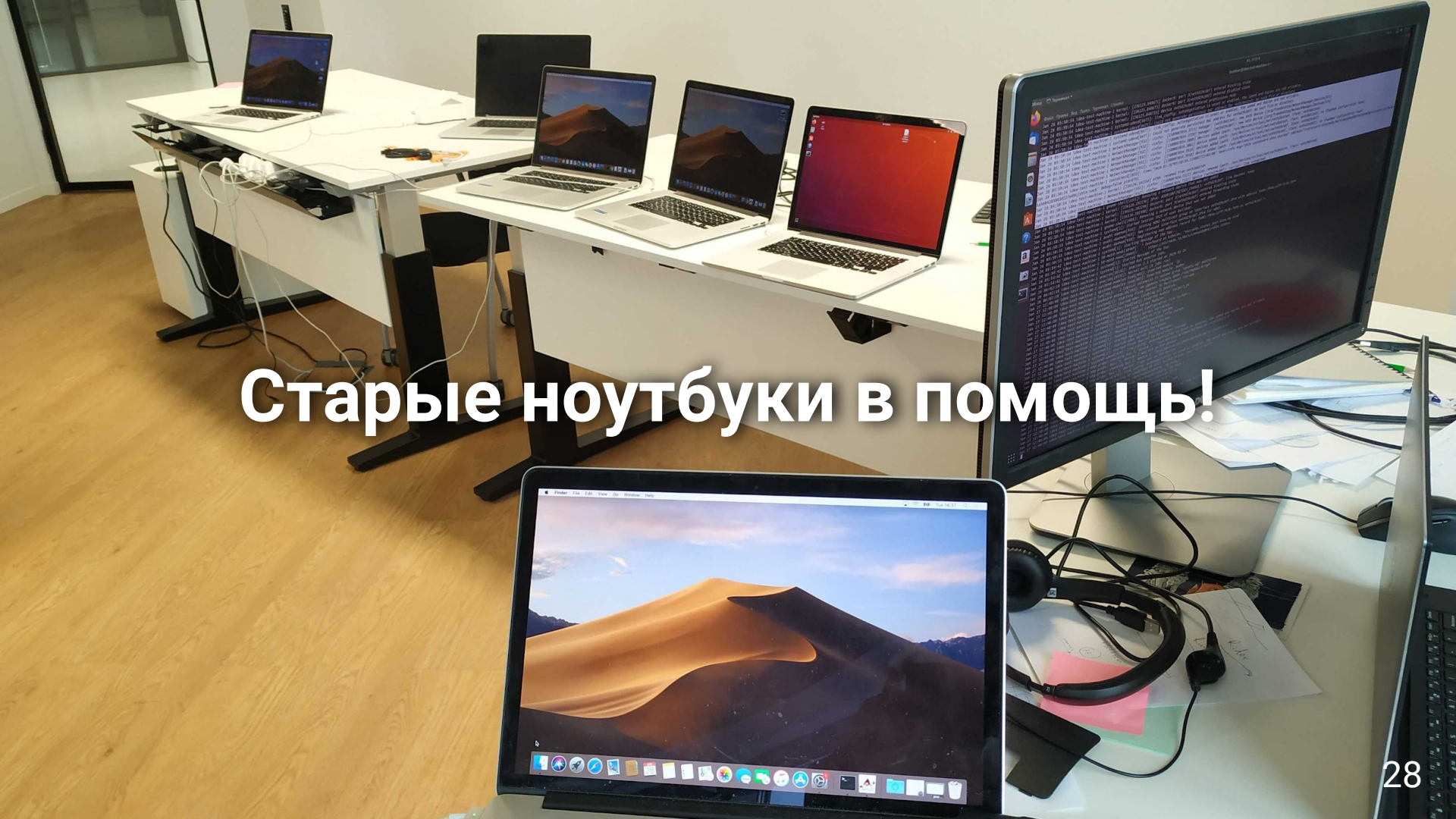
JDK 8 - JDK 15





Удалённое исполнение JavaScript

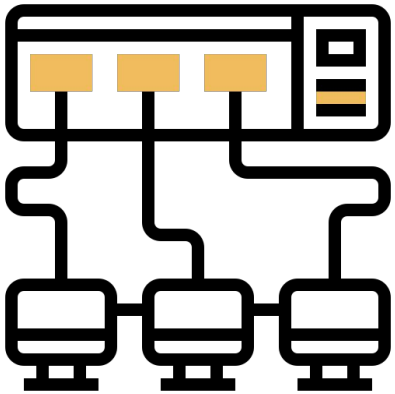
1. Отправка JavaScript по сети
2. Исполнение JavaScript ES6 при помощи Rhino Engine


Старые ноутбуки в помощь!

Парк машин

Robot-registry (наш Selenoid + Selenoid UI) управляет:

1. Реальные машины (старые ноутбуки в серверной)
2. Docker контейнеры с Linux + X11 + XFCE

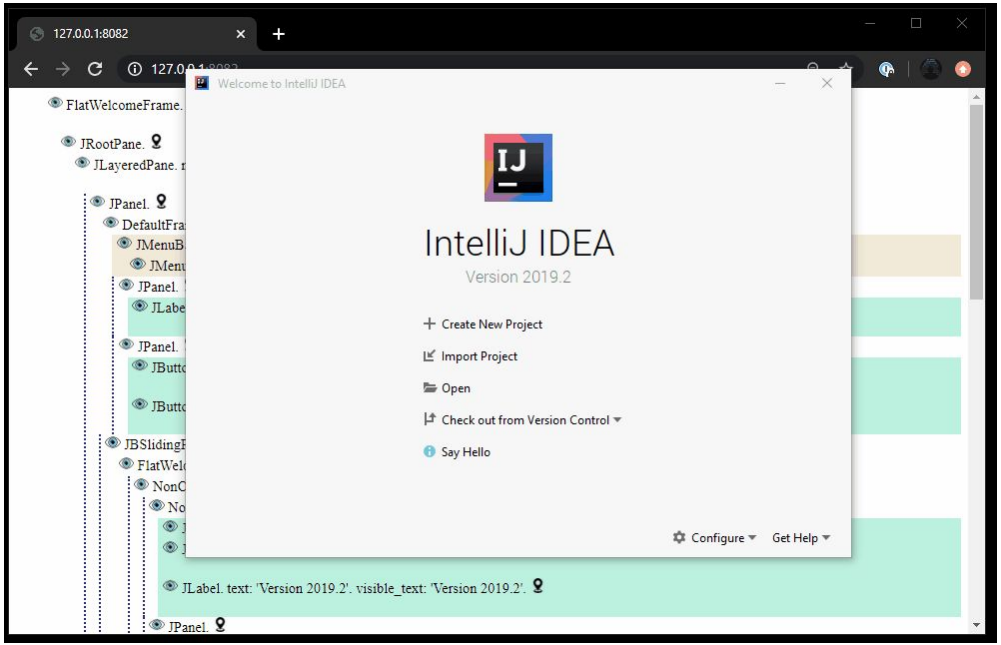


Возможности:

- Запуск/остановка версий IDE для тестирования
- Получение структуры UI
- Подключение к машинам по VNC
- Резервирование машин для ручной проверки тестов

Robot Registry

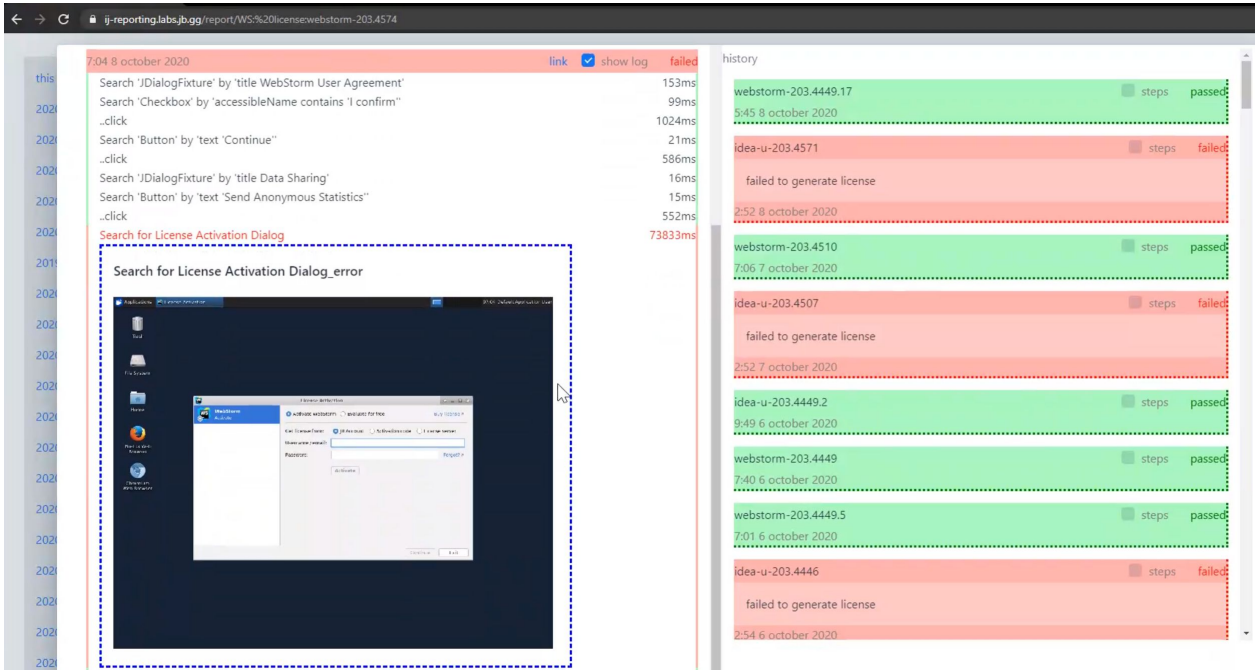
Иерархия компонентов Java Swing



ColorBlindnessPanel.
JCheckBox. Visible text: 'Adjust colors for red-green vision deficien
deuteranopia)'.
ComboBox.
DarculaComboBoxUI.
CellRendererPane.
SwingActionLink. text: 'How it works'. Visible text: 'How it works'.
JCheckBox. Visible text: 'Support screen readers (requires restart)'.
JBCheckBox. text: 'Use contrast scrollbars'. Visible text: 'Use contrast
JPanel.
JPanel.
JLabel. text: 'Theme:'. Visible text: 'Theme:'.
JComboBox.
DarculaComboBoxUI.
CellRendererPane.
UIResource. text: 'IntelliJ Light'.
JPanel.
FontComboBox.
DarculaComboBoxUI.
CellRendererPane.
FontInfoRenderer.
JLabel. text: 'Size:'. Visible text: 'Size:'.
ComboBox.
DarculaComboBoxUI.
CellRendererPane.
BorderlessTextField. Visible text: '13'.
JCheckBox. text: 'Use custom font:'. Visible text: 'Use custom font:'.

Разбор упавших тестов

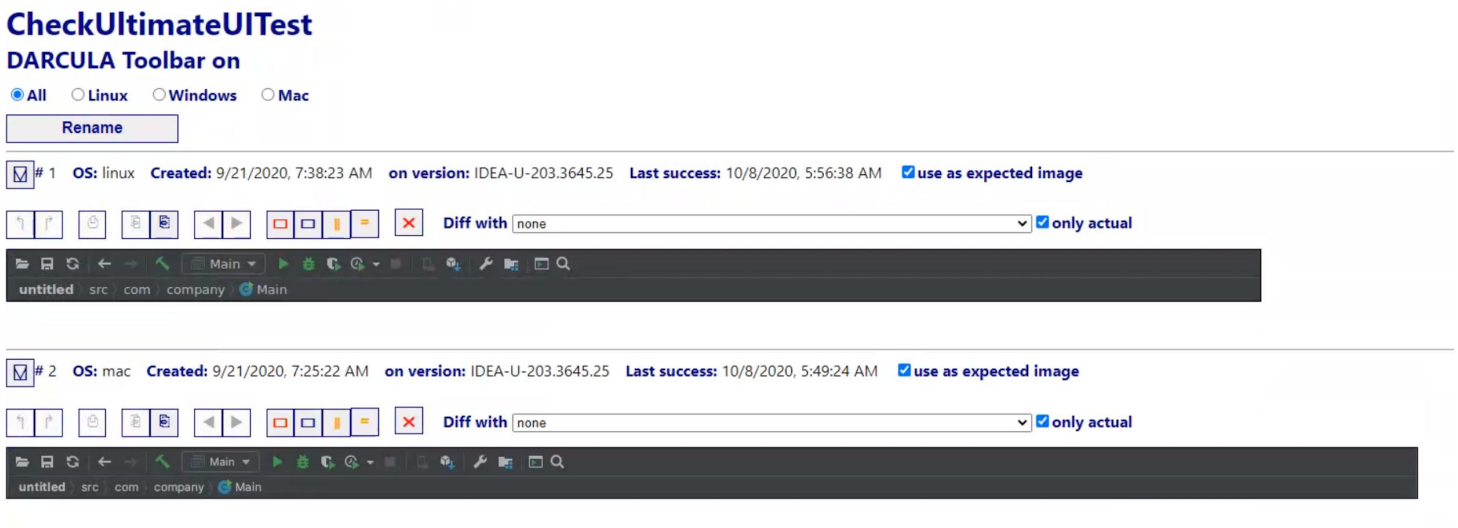
1. Сервис отчётов
2. TestRail



Визуальные проверки

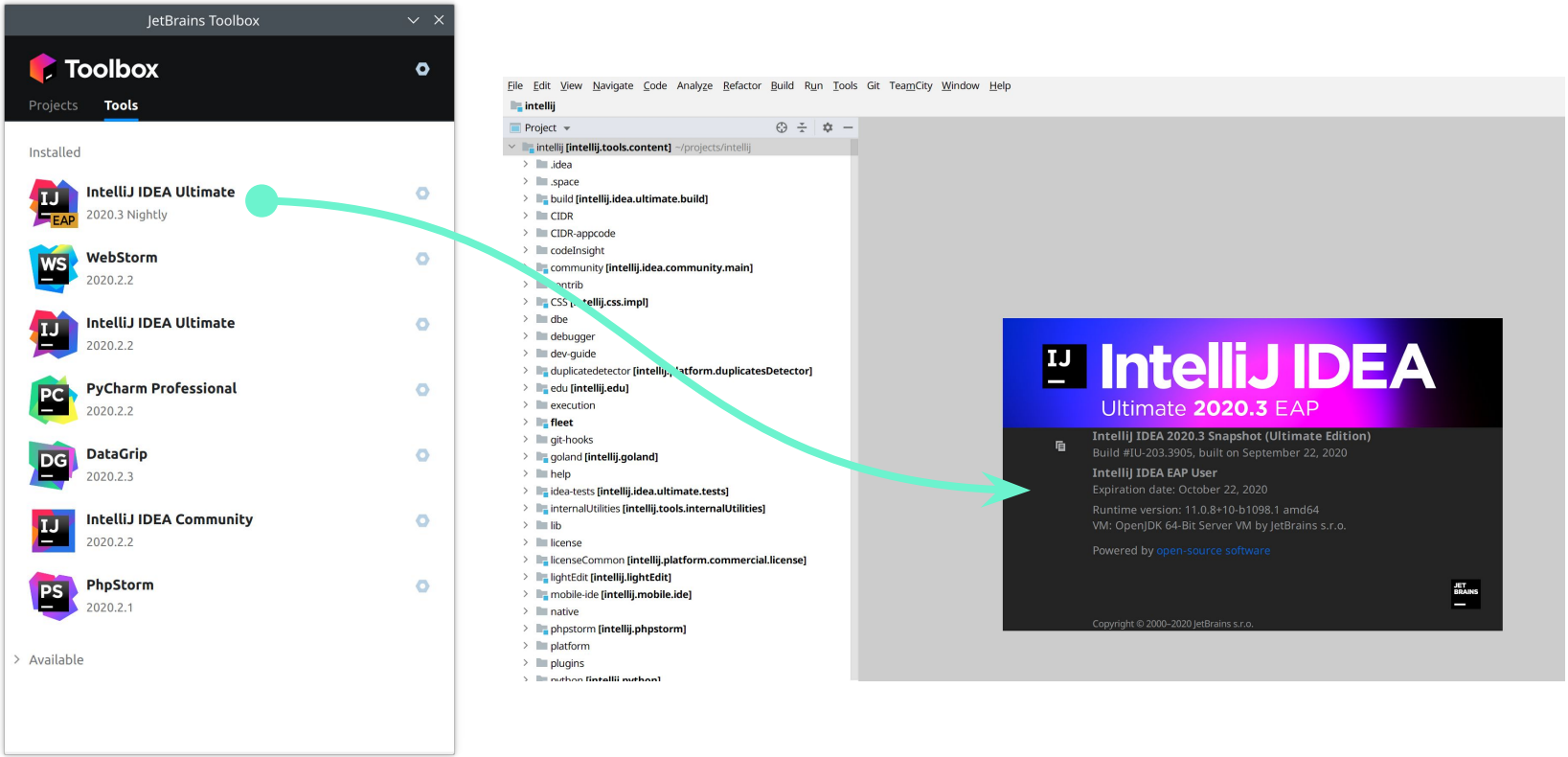
- Сервис актуальных скриншотов
- Сравнение скриншотов и OCR при помощи aShot:

pazone/ashot



Как работают команды IDE в JetBrains

Экстремальный догфудинг

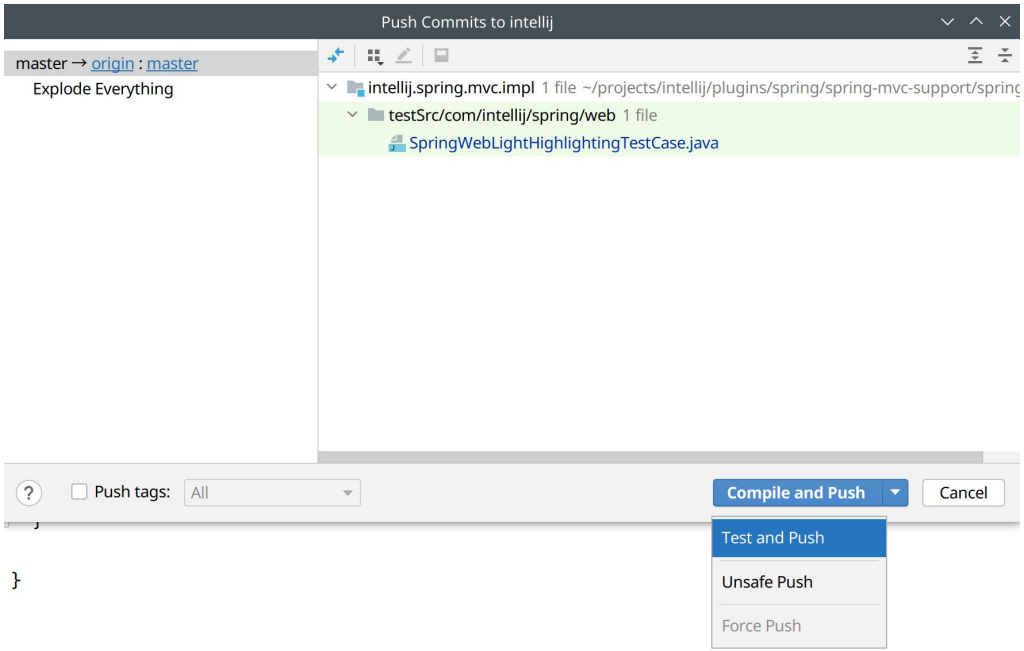


Много автоматизации на CI

- Inspections (статический анализ, SSR, i18n, ...)
- Zero-tolerance Inspections
- Тесты производительности на больших OSS проектах
- Автоматический cherry-pick для веток после ревью
- Автоматическое создание релизных веток
- ...

Safe Push

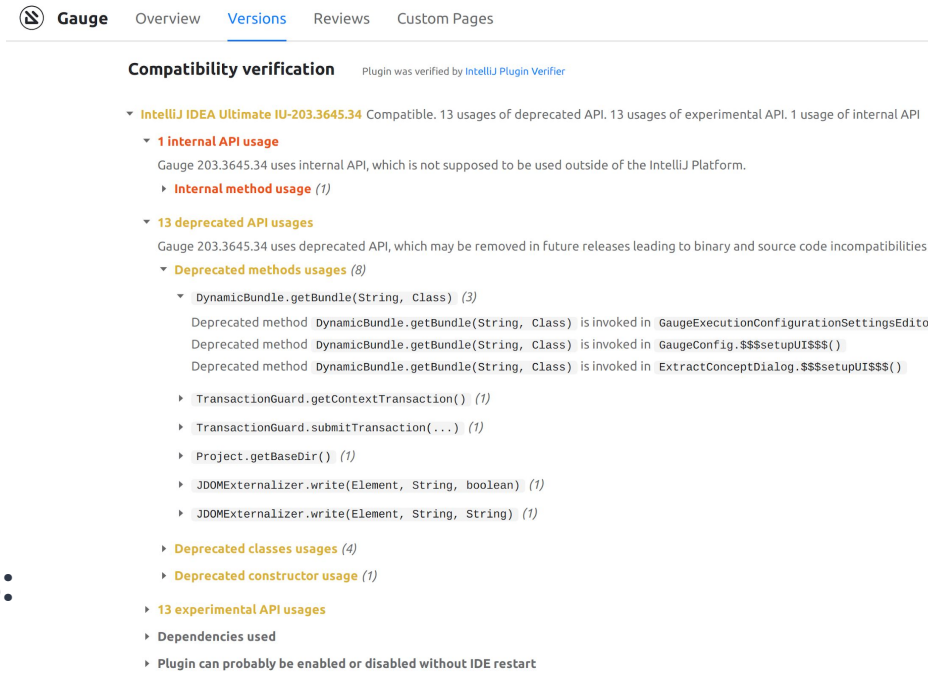
Как запушить всё в пятницу вечером и потом не сгореть от стыда?



Секретная технология!

Совместимость API и как это проверить

1. Бинарная совместимость с плагинами
2. Предупреждения об использование Internal и Deprecated API



JetBrains/intellij-plugin-verifier

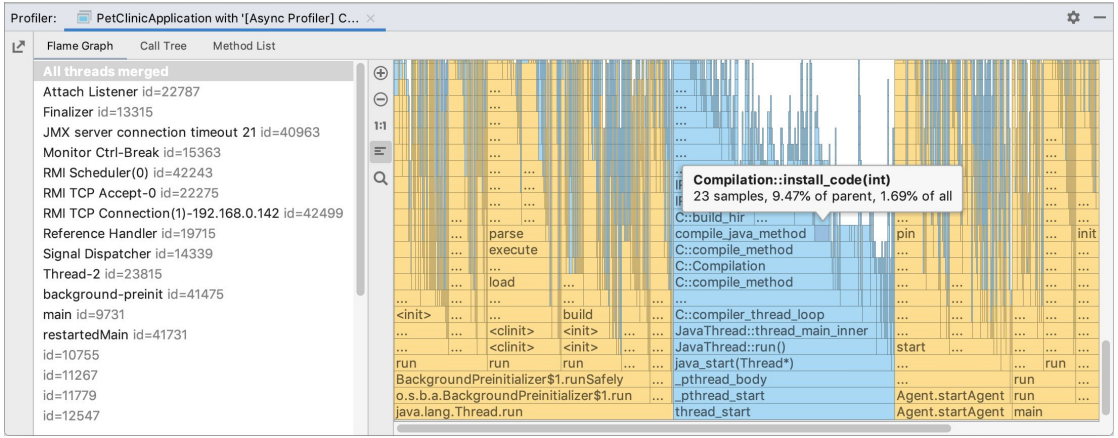
*** Для Java библиотек воспользуйтесь:**

siom79/japicmp

Инструментарий анализа производительности

Какими инструментами мы пользуемся для анализа производительности:

1. Debugger (CE)
2. Thread dump analysis (CE)
3. Eclipse MAT (OSS)
4. IDEA + Async Profiler (IU)
5. YourKit (Commercial)



Казуальный профайлер для Java и Kotlin

Spot Profiler:
https://plugins.jetbrains.com/plugin/13355-spot-profiler-for-java-and-kotlin

Куда копать дальше

- Исходный код IntelliJ IDEA CE: github.com/JetBrains/intellij-community
- Документация Plugin DevKit: https://www.jetbrains.org/intellij/sdk/docs/basics/testing_plugins/testing_plugins.html
- IntelliJ UI Test Robot: https://github.com/JetBrains/intellij-ui-test-robot

Вопросы ?

Twitter:
@Yuriy_Artamonov