

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 мейнтайнер Gauge плагина для IntelliJ IDEA

Joker<?>

#jokerconf



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

The screenshot shows the IntelliJ IDEA interface with several components highlighted:

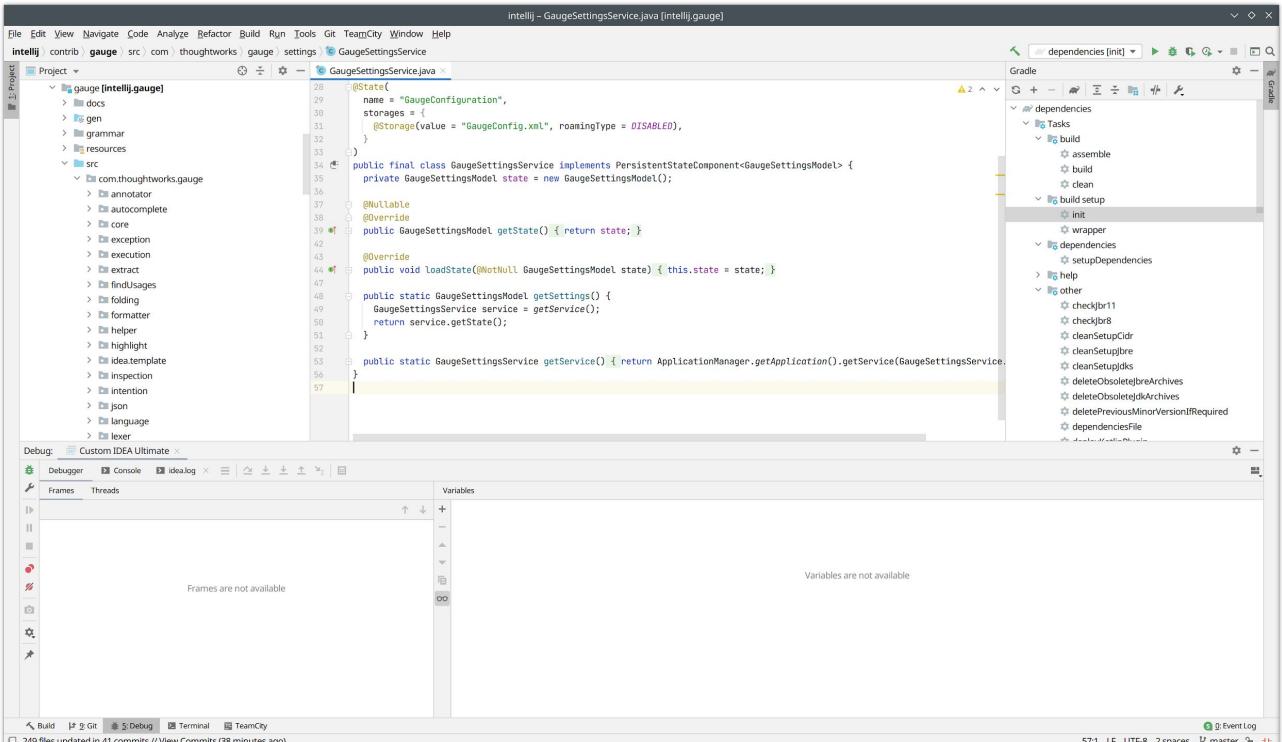
- Project Tool Window (Left):** Contains a tree view of project files under the "Project" tab. A blue box highlights the "util" folder, which contains "GaugeUtil", "HookUtil", "SocketUtils", and "StepUtil".
- Code Editor (Center):** Displays the file `GaugeSettingsService.java`. A green oval highlights the code area. A yellow box highlights the status bar at the top right of the editor.
- Gradle Tool Window (Right):** Shows the build script structure. A purple box highlights the "dependencies" section, which includes tasks like "assemble", "build", and "clean".
- Run Tool Window (Bottom Left):** Shows the execution results of the "dependencies [setupBundledMaven]" task. A red box highlights the "Run" tab. A pink box highlights the log output.
- Status Bar (Bottom Right):** Shows the current time (31:65), file format (LF), encoding (UTF-8), and branch (master). A blue box highlights the "Event Log" icon.

Code Editor Content (GaugeSettingsService.java):

```
1  ...
16 package com.thoughtworks.gauge.settings;
17
18 import ...
19
20 @State(
21     name = "GaugeConfiguration",
22     storages = {
23         @Storage(value = "GaugeConfig.xml", roamingType = DISABLED),
24     }
25 )
26 public final class GaugeSettingsService implements PersistentStateComponent<GaugeSettingsModel> {
27     private GaugeSettingsModel state = new GaugeSettingsModel();
28
29     @Nullable
30     @Override
31     public GaugeSettingsModel getState() { return state; }
32
33     @Override
34     public void loadState(@NotNull GaugeSettingsModel state) { this.state = state; }
35
36     public static GaugeSettingsModel getSettings() {
37         GaugeSettingsService service = getService();
38         return service.getState();
39     }
40
41     public static GaugeSettingsService getService() { return ApplicationManager.getApplication().getService(GaugeSettingsService.class); }
42 }
43
44 }
```

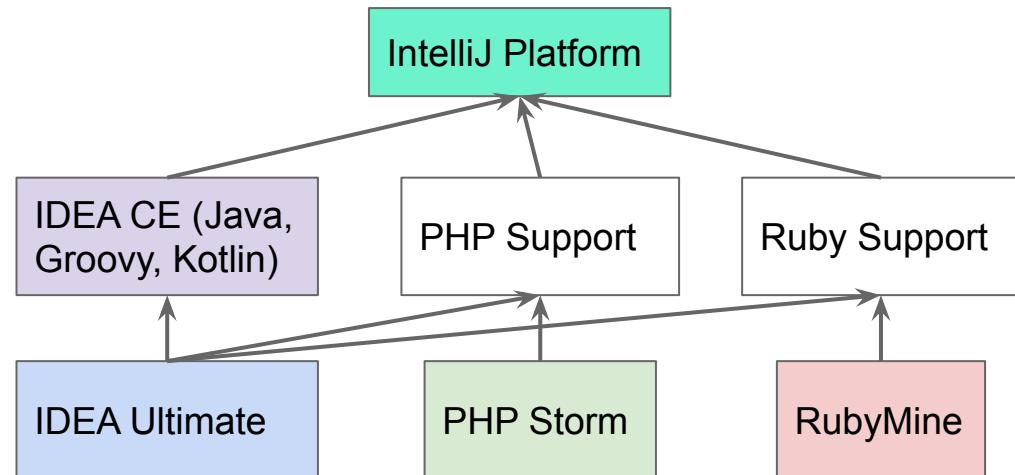
JetBrains IDE снаружи

- 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

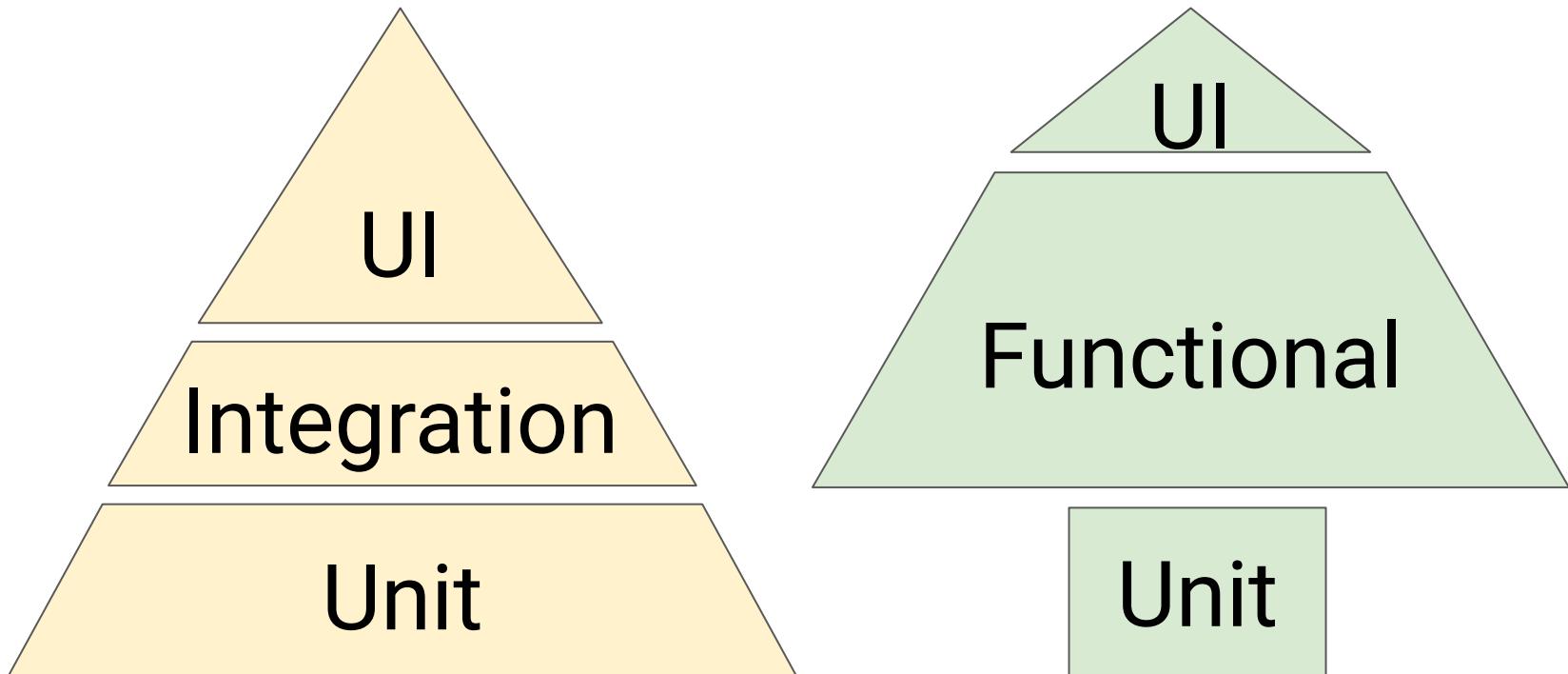


Функциональные тесты IDE

Какие тесты мы пишем чаще всего:

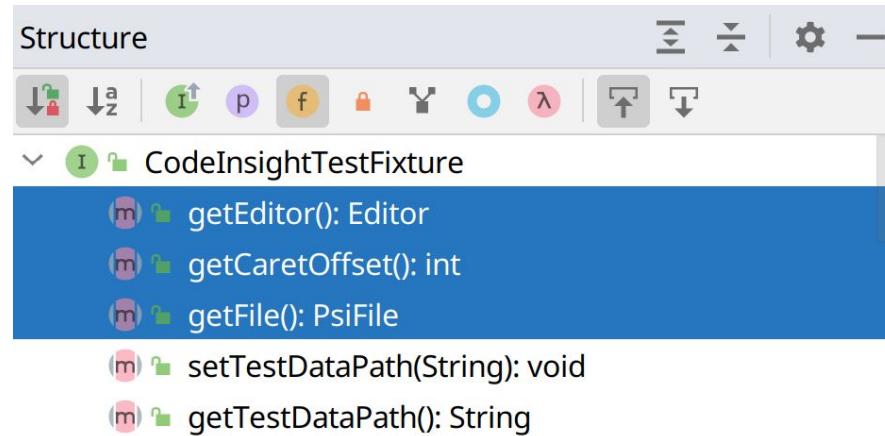
- Тесты запускаются в памяти без UI
- Тесты используют реальные реализации большинства подсистем IDE, за исключением UI компонентов
- Тесты проверяют один из сценариев использования целиком, а не отдельные функции и свойства реализации.

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

The screenshot shows an IDE interface with a Java file named `selenide-allure-junit - GoogleSearchTest.java`. The code is a functional test for Google search. A code completion dropdown is open at the bottom of the screen, listing various CSS properties starting with `align-`, such as `align-content`, `align-items`, `align-self`, etc. The IDE has tabs for Project, Favorites, and Structure. The bottom navigation bar includes Git, TODO, Problems, Profiler, Terminal, Build, and Event Log.

```
File Edit View Navigate Code Analyze Refactor Build Run Tools Git Window Help
userCanSearchAnyKeyw Add Configuration...
Project GoogleSearchTest.java
1 package org.openqa.selenium.examples;
2
3 import org.junit.Test;
4 import org.openqa.selenium.By;
5
6 import static com.codeborne.selenide.CollectionCondition.sizeGreaterThanOrEqual;
7 import static com.codeborne.selenide.Condition.*;
8 import static com.codeborne.selenide.Selenide.$;
9 import static com.codeborne.selenide.Selenide.$$;
10
11 public class GoogleSearchTest {
12     @Test
13     public void userCanSearchAnyKeyword() {
14         $(By.name("name"))
15             .shouldHave(attribute("title", "some"))
16             .shouldHave(attributeMatching("some", "Some-a-zA-Z"))
17             .shouldHave(cssValue("", ""))
18             .shouldHave(cssValue("align-content", "center"))
19             .val("selenide")
20             $$("#res .g")
21                 .shouldHave(sizeGreaterThanOrEqual(1))
22
23             $$("#res .g").shouldHave(text("selenide"))
24     }
25 }
```

13

Event Log

Incorrect CSS Property

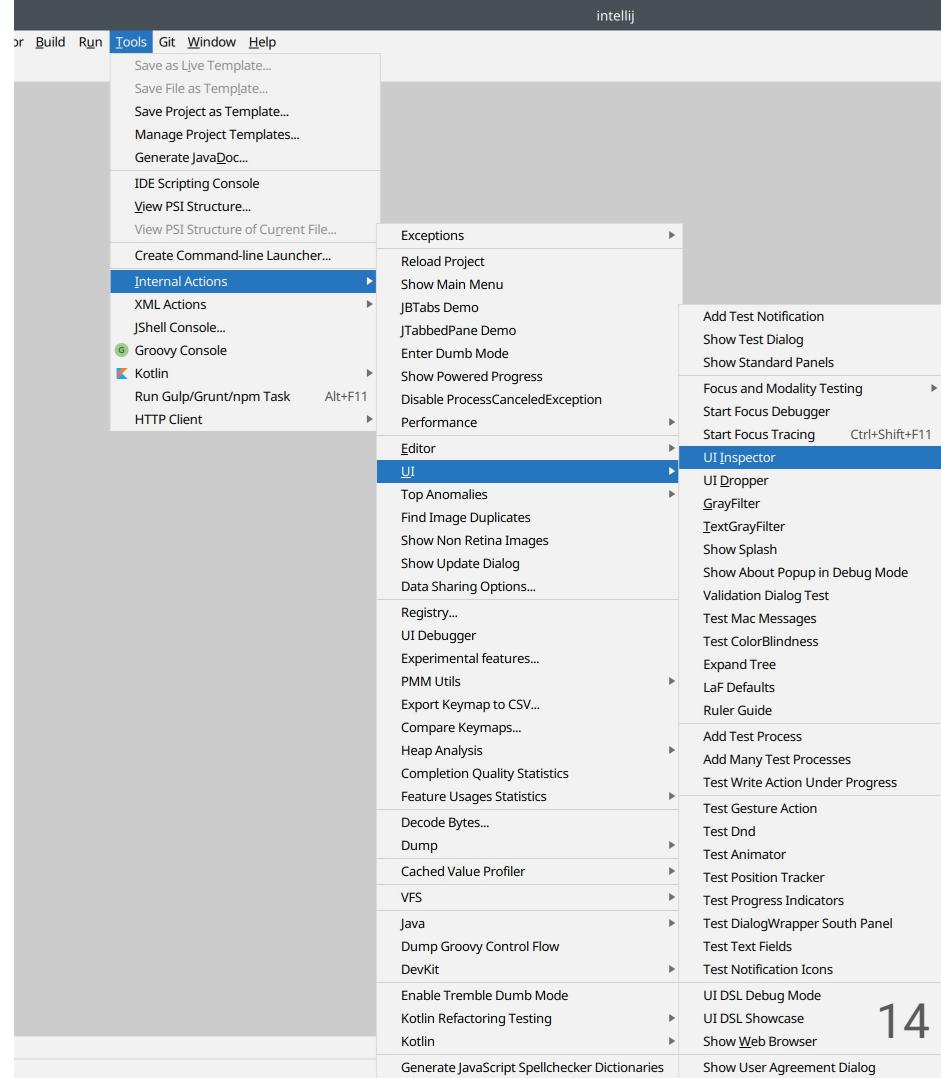
17:39 LF UTF-8 4 spaces master

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

PSI Viewer: GoogleSearchTest.java

Show PSI structure for: JAVA file Show PsiWhiteSpace Show tree nodes Dialect: Java

Text

```
12 @Test
13 public void userCanSearchAnyKeyword() {
14     $(By.name("name"))
15         .shouldHave(attribute("title", "some"))
16         .shouldHave(attributeMatching("some", "Somea-zA-Z"))
17         .shouldHave(cssValue("font-size", ""))
18         .shouldHave(cssClass("no"))
19         .val("selenide").pressEnter();
20     $$("#res .g")
21         .shouldHave(sizeGreater Than(5));
22
23     $("#res .g").shouldHave(text("selenide.org"));
24 }
25 }
```

PSI Tree

References	Block Structure	Stub Structure
com.intellij.selenium.shared.dom.CssPropertyReference		

```
iMethodCallExpression:$(By.name("name")) .shouldHave(a
PsiReferenceExpression:$(By.name("name")) .shouldHave(
    PsiMethodCallExpression:$(By.name("name")) .shouldHave(
        PsiReferenceExpression:$(By.name("name")) .shouldHave(
            PsiMethodCallExpression:$(By.name("name"))
                PsiReferenceExpression:$(By.name("name"))
                    PsiMethodCallExpression:cssValue("font-siz
                        PsiReferenceExpression:cssValue
                            PsiExpressionList
                                PsiJavaToken:LPARENTH
                                    PsiMethodCallExpression:cssValue("font-siz
                                        PsiReferenceExpression:cssValue
                                            PsiExpressionList
                                                PsiJavaToken:LPARENTH
                                                    PsiLiteralExpression:"font-size"
                                                        PsiJavaToken:STRING_LITERAL
                                                            PsiJavaToken:COMMA

```

?

Copy PSI Build PSI Tree Close

15

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



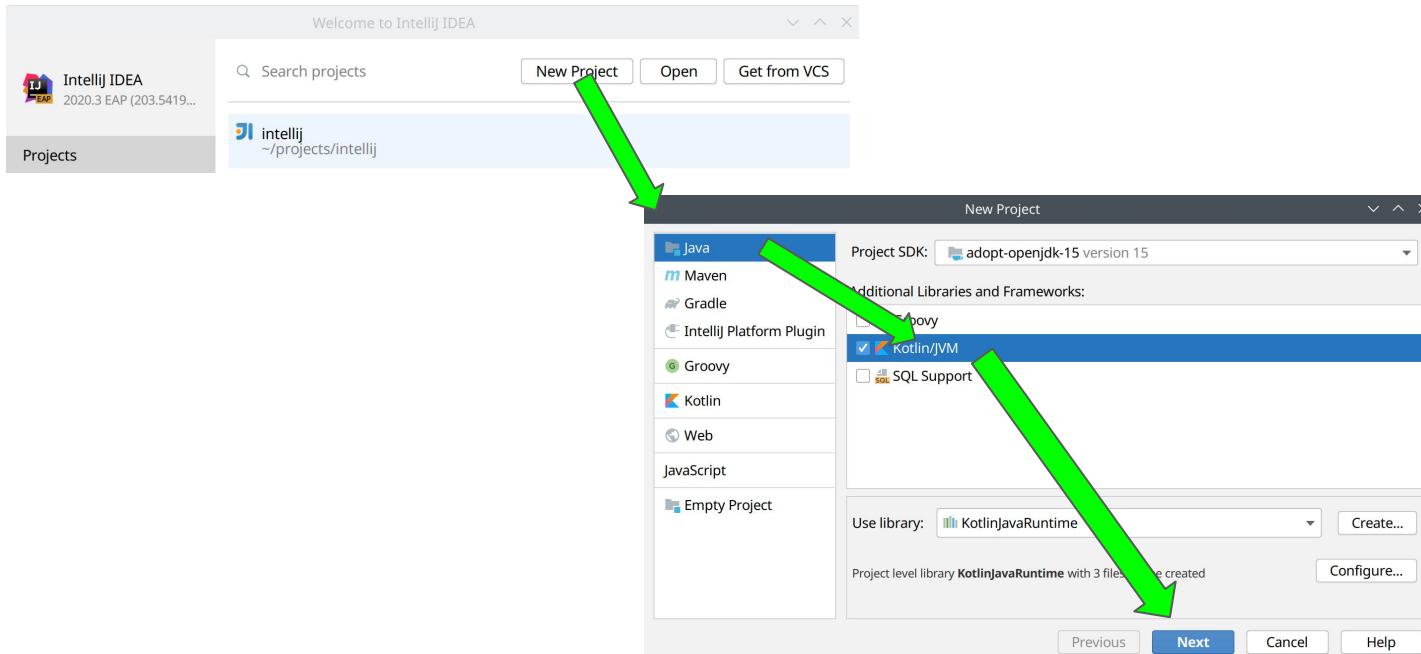
[JetBrains/jetCheck](https://github.com/JetBrains/jetCheck)

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:

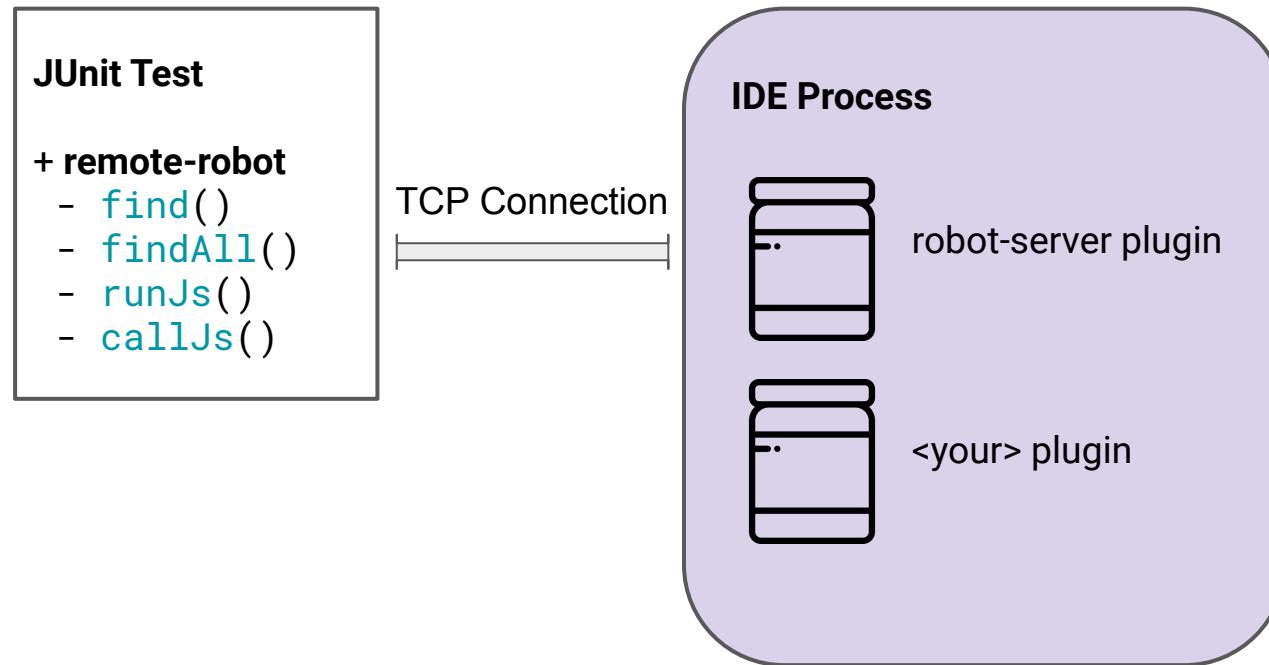
`java.awt.Robot` в runtime приложения / FEST-Swing



[JetBrains/intellij-ui-test-robot](https://github.com/JetBrains/intellij-ui-test-robot)

Схема взаимодействия

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



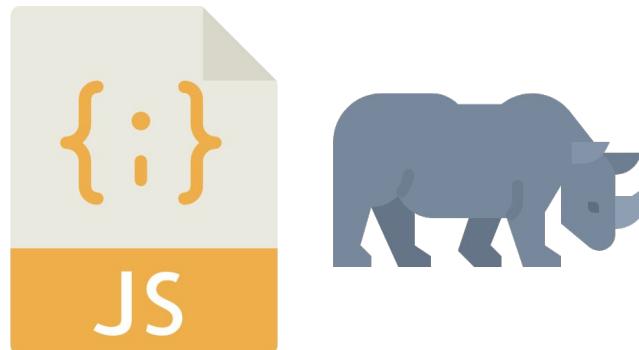
Поиск элементов на экране

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](https://github.com.mozilla/rhino)
- Nashorn Engine (JEP-174)  **JDK 8 - JDK 15**

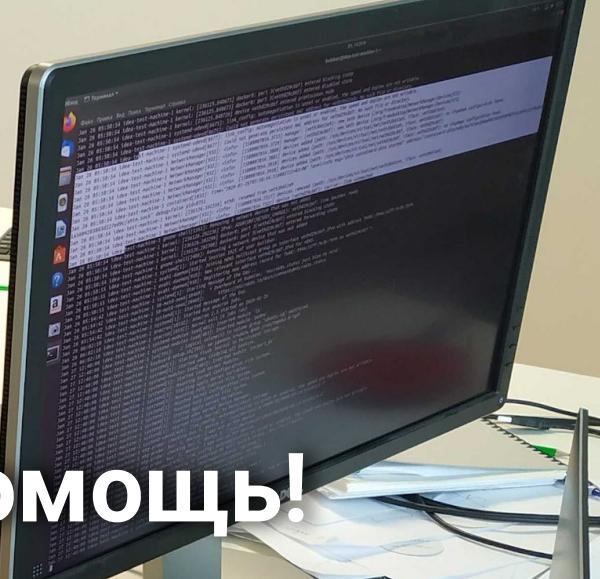


Удалённое исполнение JavaScript

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

```
fun click() = step("click at ${elementData.tag}") {
    scroll()
    container.runJs("""
        robot.click(component, new Point(${x + height / 2}, $centerY))
    """)
}
```

Старые ноутбуки в помощь!



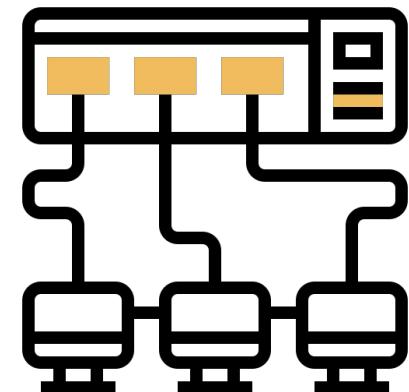
Парк машин

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

robot-registry

Find container

reserve	delete	more...	idea-img:latest http://172.31.150.195:18182	FavoritesTest	vnc noVnc
reserve	delete	more...	idea-img:latest http://172.31.144.150:18181	LicenseKeyActivationTest	vnc noVnc
reserve	more...		intellij-windows10-ui-1() http://172.31.129.156:8180	KarmaDebugTest. Windows	vnc noVnc
reserve	delete	more...	idea-img:latest http://172.31.128.151:18182	LicenseKeyActivationTest	vnc noVnc
reserve	delete	more...	idea-img:latest http://172.31.128.151:18181	FavoritesTest	vnc noVnc

Run ide: idea-u ▾ 202 ▾ 202.7660.17 ▾

launch

Kill ide

status

update 'robot-server'

sync test data

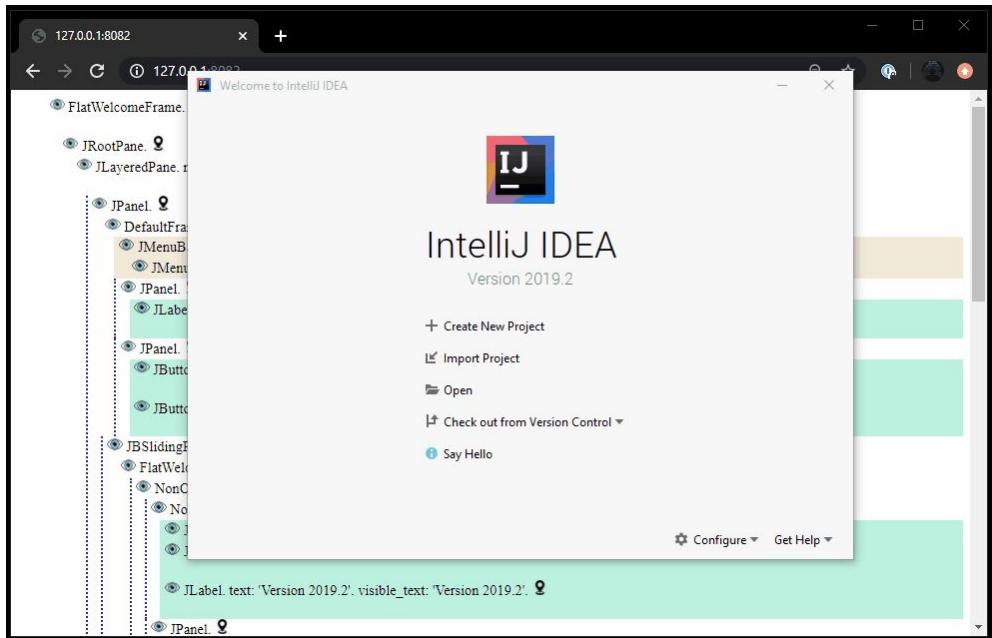
components hierarchy idea log

For linux: vncviewer 172.31.128.151:15901 &

reserve more...

ubuntu(swift)
http://172.30.164.42:8180

Иерархия компонентов Java Swing



ColorBlindnessPanel.

JCheckBox. Visible text: 'Adjust colors for red-green vision deficiency (deutanopia)'.

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 scrollbars'.

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

The screenshot shows a browser window displaying a test report from ji-reporting.labs.jb.gg/report/WS%20license/webstorm-203.4574. The report lists several test steps, some of which have failed. A specific step, "Search for License Activation Dialog_error", is highlighted with a red border. To the right of the report, a history of test runs is shown, with each run represented by a horizontal bar indicating the status of its steps.

Test Run	Status	Details
webstorm-203.4449.17	passed	steps passed
idea-u-203.4571	failed	failed to generate license
webstorm-203.4510	passed	steps passed
idea-u-203.4507	failed	failed to generate license
idea-u-203.4449.2	passed	steps passed
webstorm-203.4449	passed	steps passed
webstorm-203.4449.5	passed	steps passed
idea-u-203.4446	failed	failed to generate license

Визуальные проверки

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



[pazone/ashot](#)

CheckUltimateUITest

DARCULA Toolbar on

All Linux Windows Mac

[Rename](#)

1 OS: linux Created: 9/21/2020, 7:38:23 AM on version: IDEA-U-203.3645.25 Last success: 10/8/2020, 5:56:38 AM use as expected image

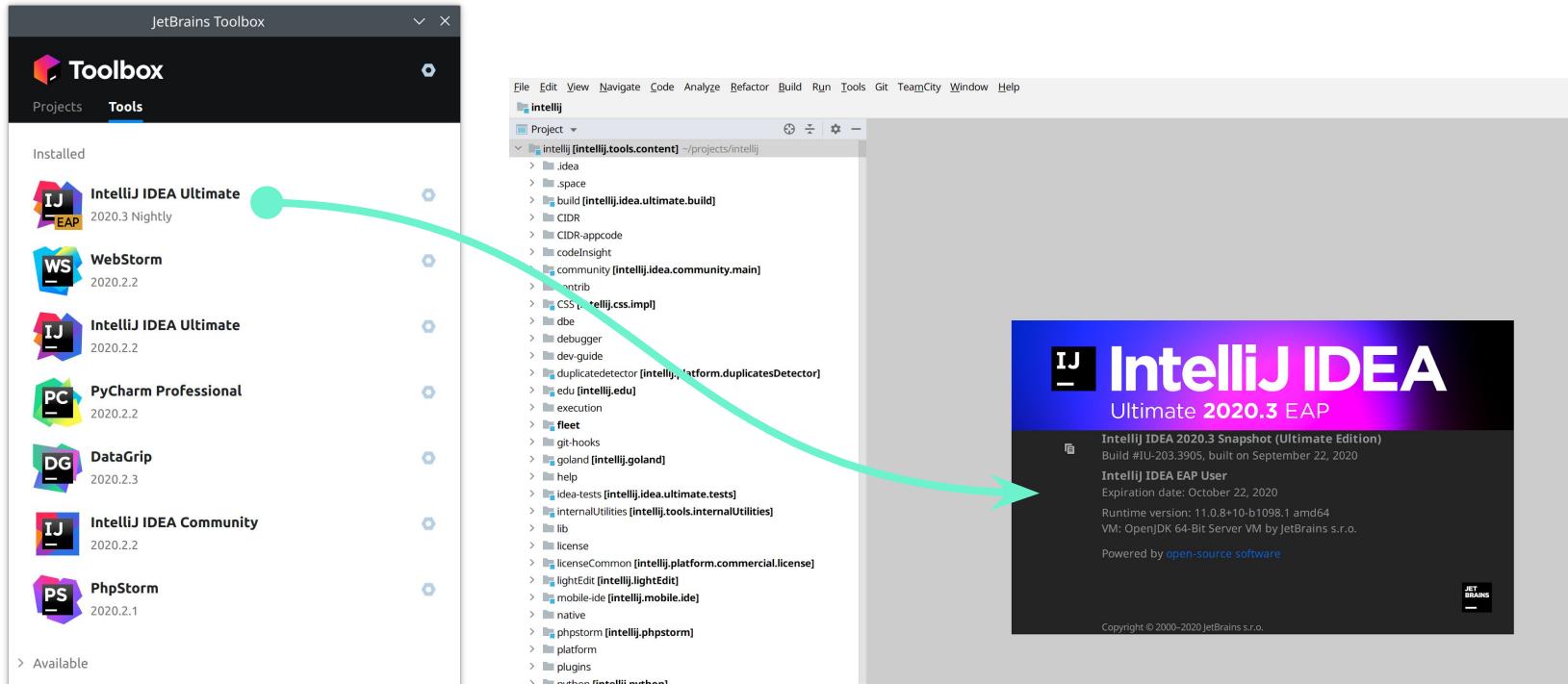
Diff with none only actual

2 OS: mac Created: 9/21/2020, 7:25:22 AM on version: IDEA-U-203.3645.25 Last success: 10/8/2020, 5:49:24 AM use as expected image

Diff with none only actual

Как работают команды IDE в JetBrains

Экстремальный догфудинг



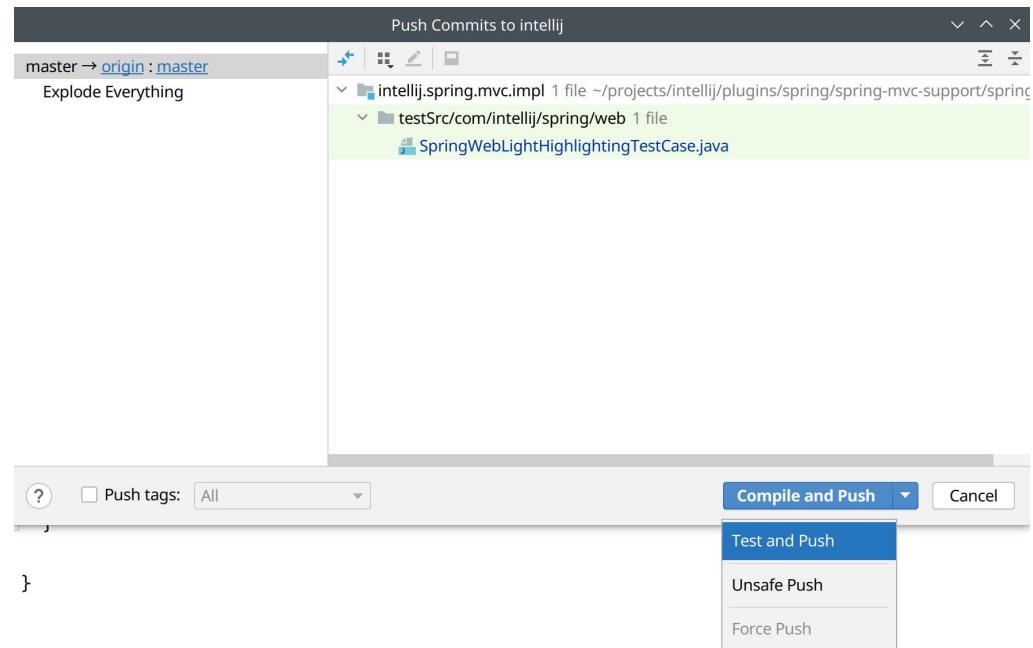
Много автоматизации на CI

- 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](https://github.com/JetBrains/intelliJ-plugin-verifier)

* Для Java библиотек воспользуйтесь:



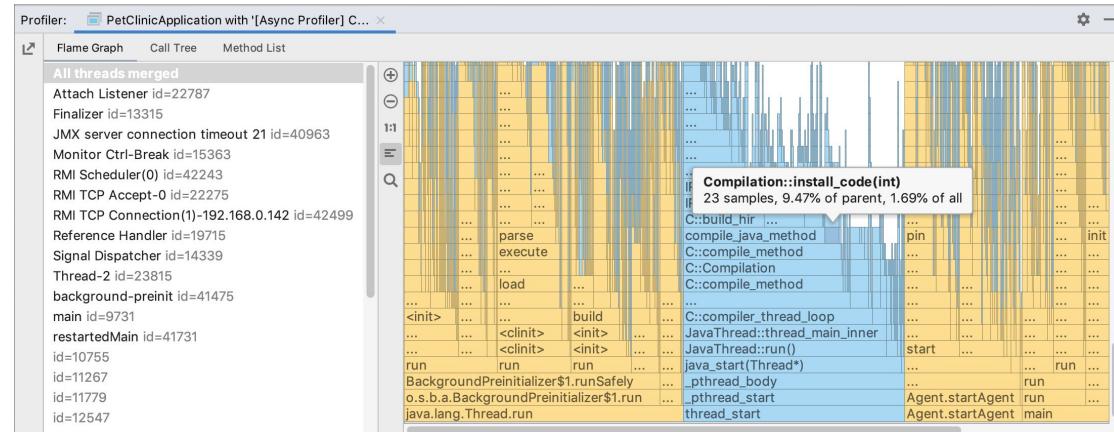
[siom79/japicmp](https://github.com/siom79/japicmp)

The screenshot shows the 'Compatibility verification' report for the IntelliJ IDEA Ultimate plugin. It indicates that the plugin is compatible with version IU-203.3645.34, with 13 usages of deprecated API, 1 usage of experimental API, and 1 usage of internal API. The report is verified by the IntelliJ Plugin Verifier. The interface includes tabs for Gauge, Overview, Versions (which is selected), Reviews, and Custom Pages. Below the tabs, there are sections for Internal API usage, Deprecated API usages, Experimental API usages, and Dependencies used, each with a list of specific code snippets and their locations.

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>

The screenshot shows a code editor with Java code annotated by the Spot Profiler plugin. The code is part of a class that implements the `executeWriteAction` method. The annotations show execution times for specific blocks of code:

- Line 45: `final Project project = editor.getProject();` - 177mks
- Line 54: `final MoverWrapper mover = getSuitableMover(editor, file);` - 787ms
- Line 57: `mover.move(editor, file);` - 552ms

The code itself is as follows:

```
44
45 @Override
46 public void executeWriteAction(Editor editor, Caret caret, DataContext dataContext) {
47     final Project project = editor.getProject();
48     assert project != null;
49     final PsiDocumentManager documentManager = PsiDocumentManager.getInstance(project);
50     final Document document = editor.getDocument();
51     documentManager.commitDocument(document);
52     PsiFile file = getRoot(documentManager.getPsiFile(document), editor);
53
54     if (file != null) {
55         final MoverWrapper mover = getSuitableMover(editor, file); 787ms
56         if (mover != null && mover.getInfo().toMove2 != null) {
57             LineRange range = mover.getInfo().toMove;
58             if ((range.startLine > 0 || isDown) && (range.endLine < document.getLineCount() || !isDown)) {
59                 mover.move(editor, file); 552ms
60             }
61         }
62     }
}
```

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