

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

Joker<?>

#jokerconf

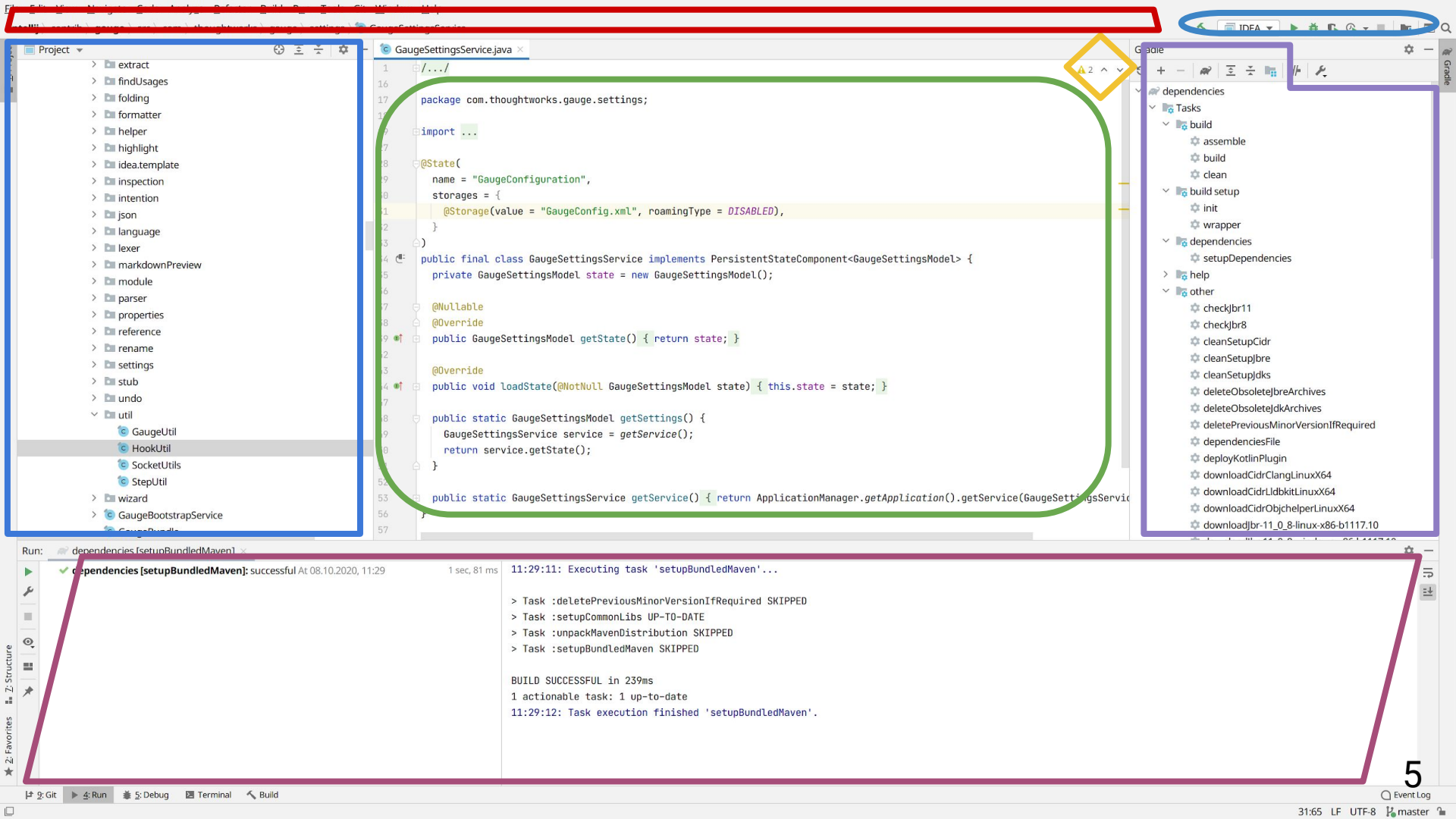


План действий

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

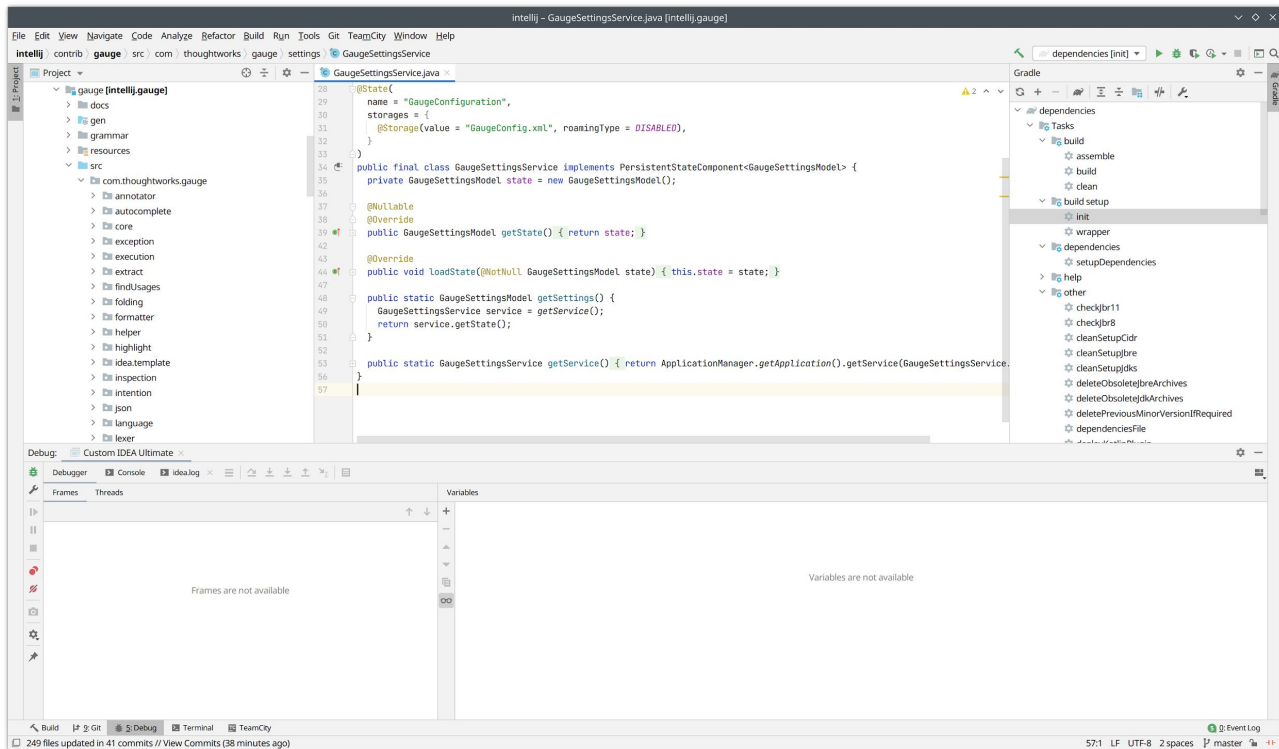
Зачем вам слушать этот доклад?

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



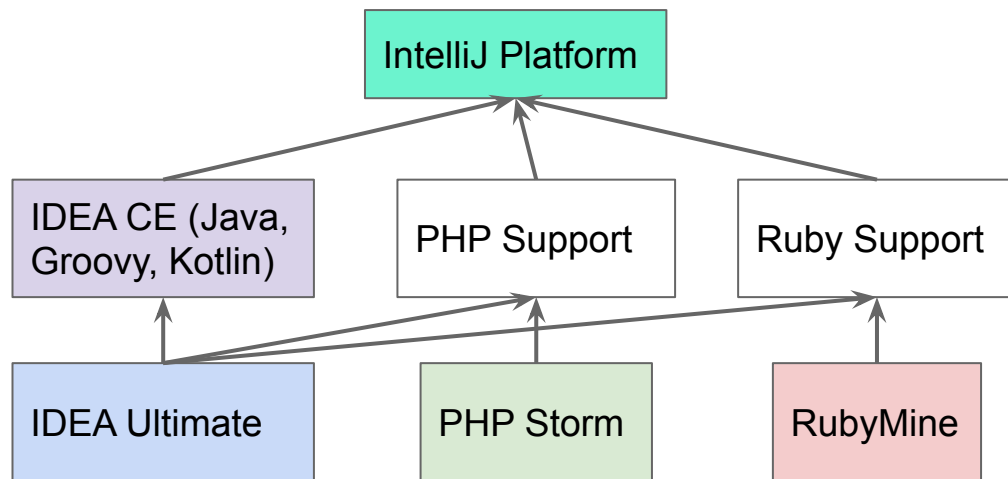
JetBrains IDE снаружи

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



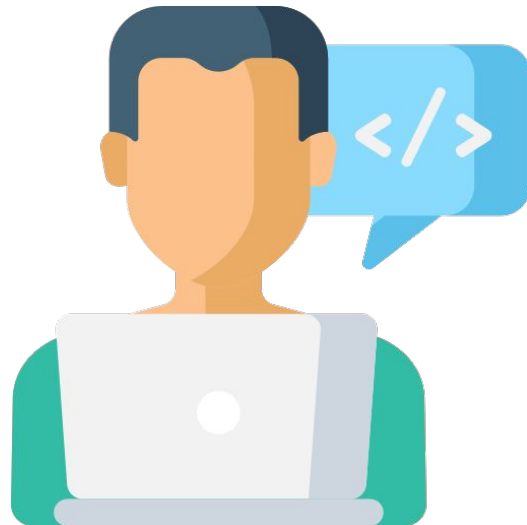
JetBrains IDE изнутри

- 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

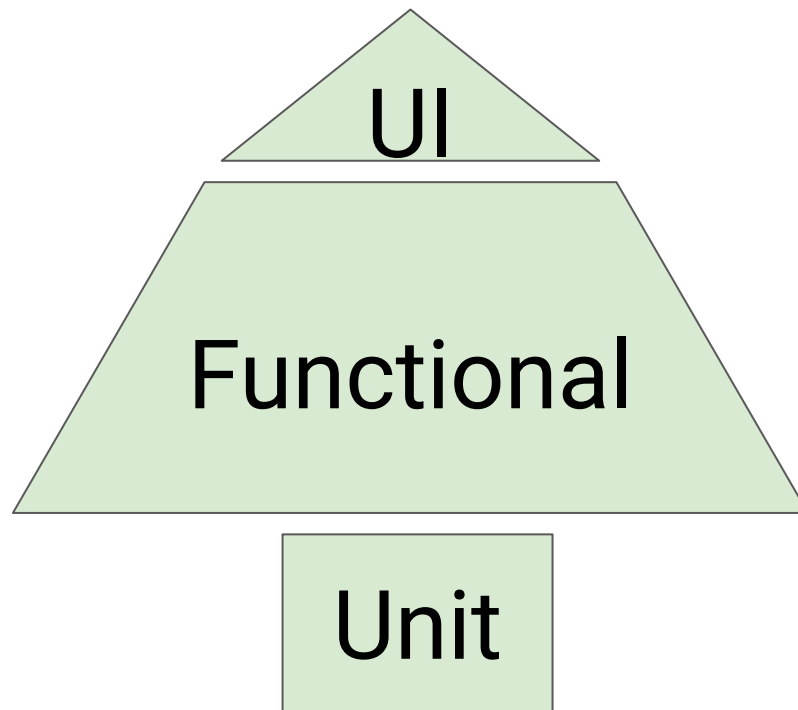
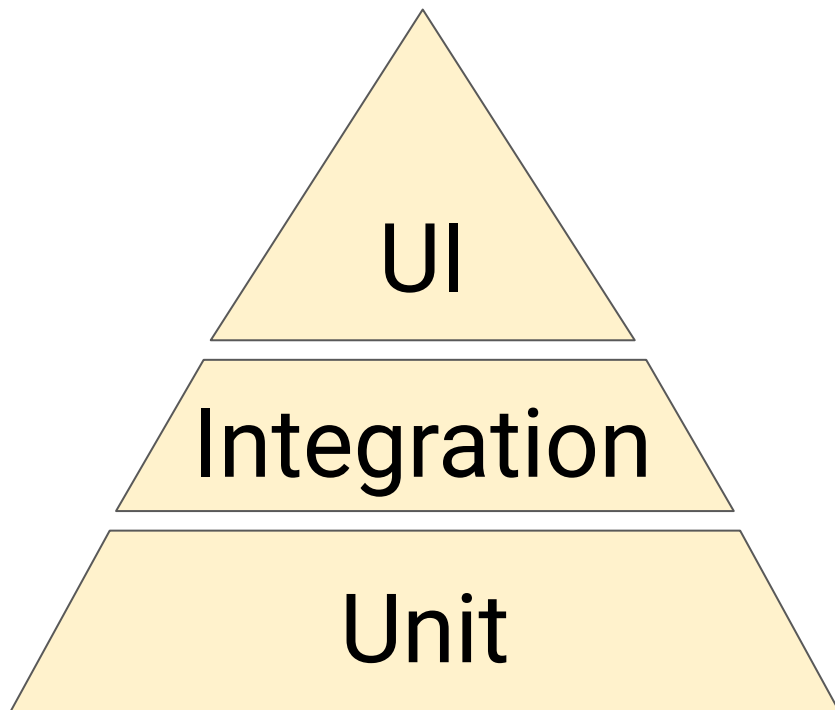


Функциональные тесты IDE

Какие тесты мы пишем чаще всего:

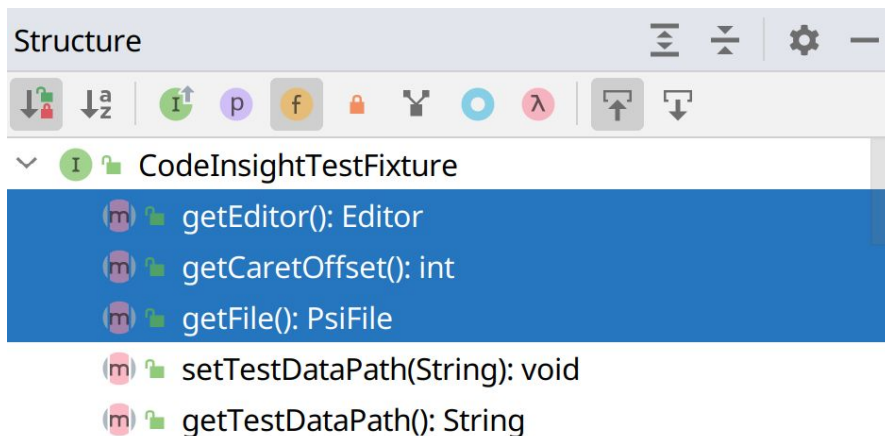
- Тесты запускаются в памяти без UI
- Тесты используют реальные реализации большинства подсистем IDE, за исключением UI компонентов
- Тесты проверяют один из сценариев использования целиком, а не отдельные функции и свойства реализации.

Пирамида или Ёлка



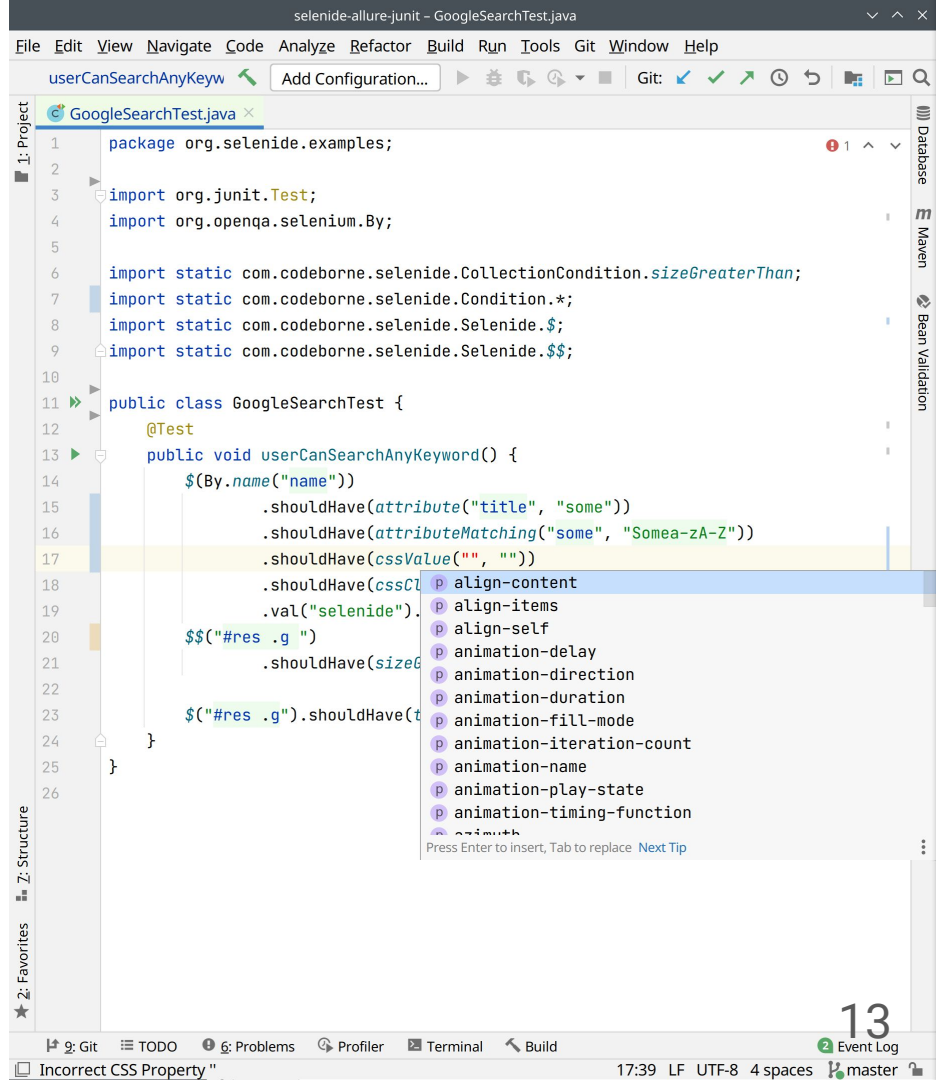
Fixture - модель IDE в памяти

- 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



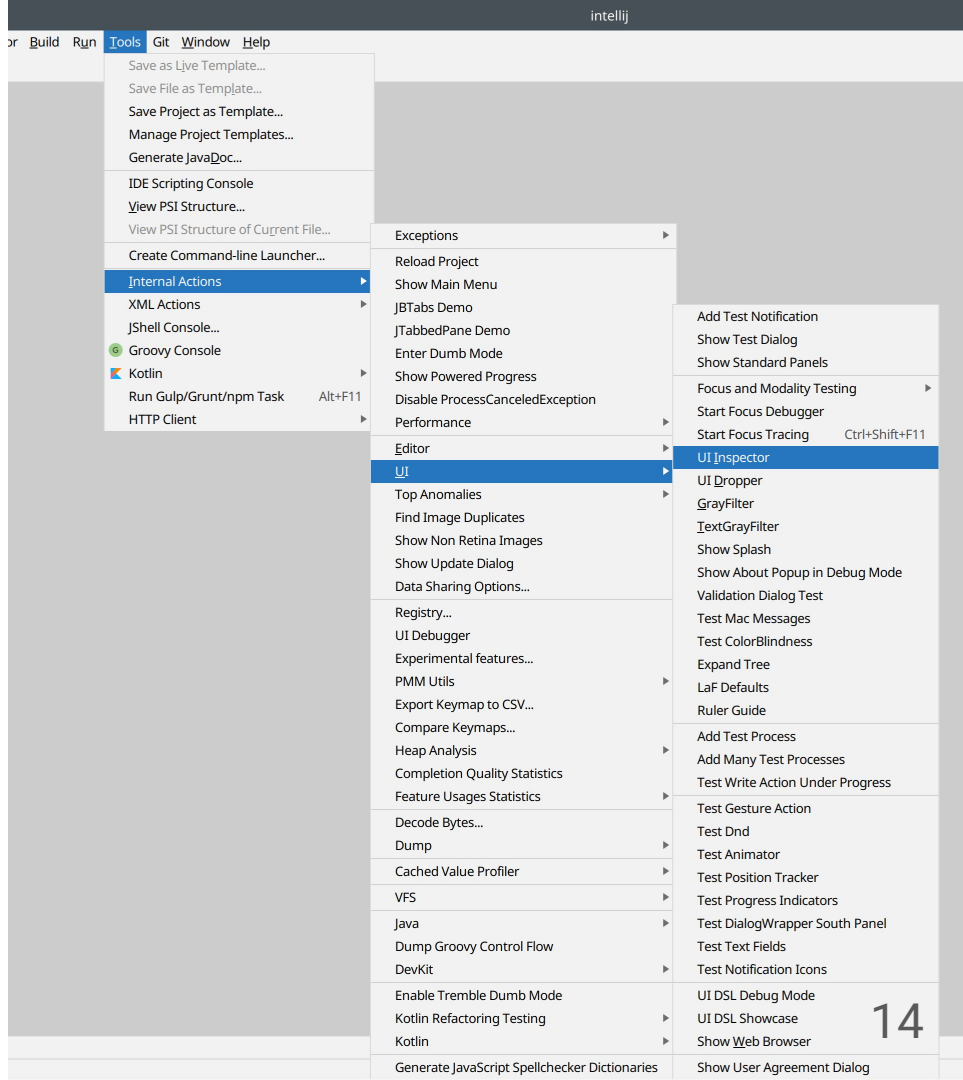
Internal Mode

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`



Tools - View PSI Structure of Current File

- 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 Моск-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



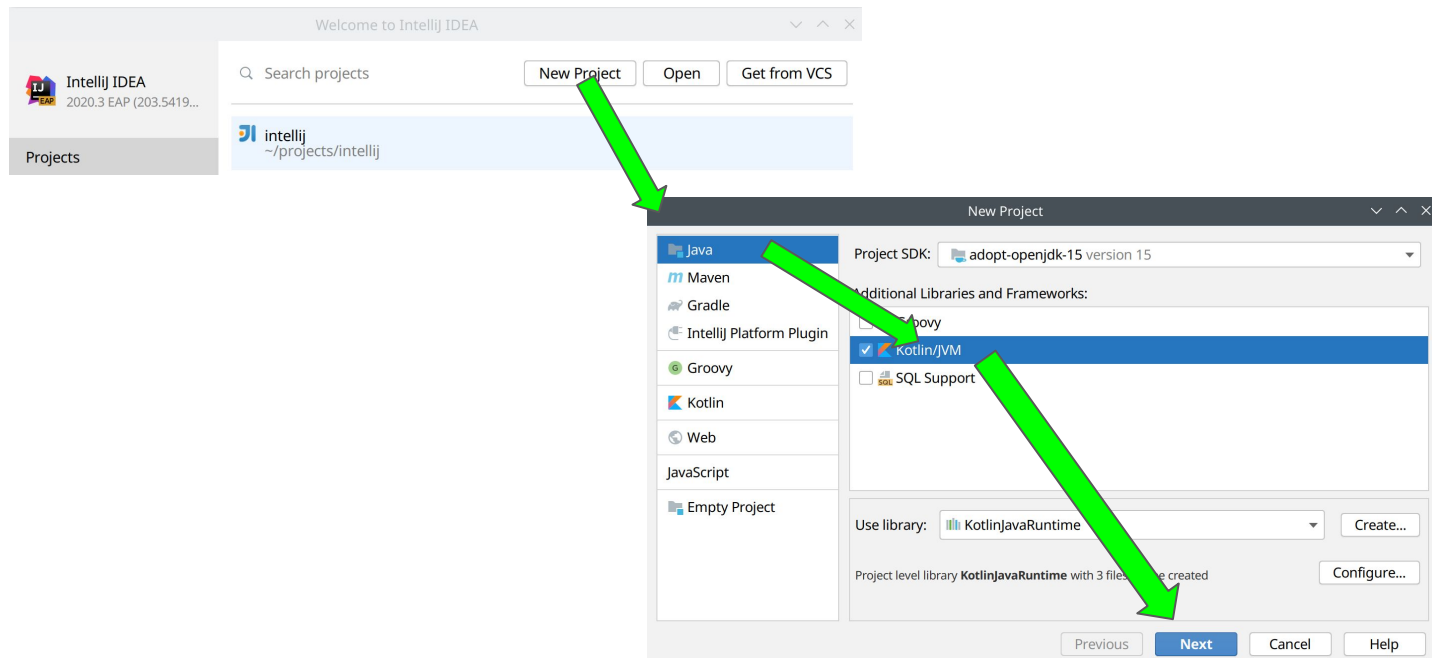
[JetBrains/jetCheck](https://github.com/JetBrains/jetCheck)

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:

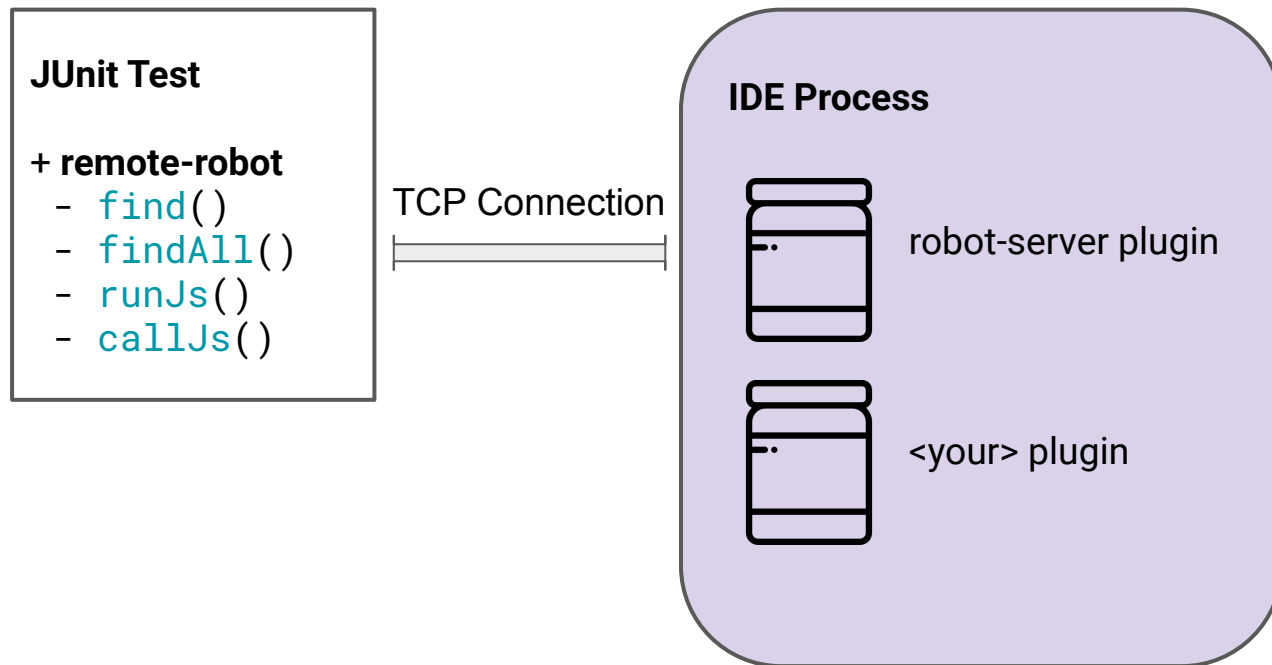
`java.awt.Robot` в runtime приложения / FEST-Swing



[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

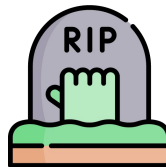
1. Поиск компонентов по XPath и атрибутам
2. Исполнение JavaScript

```
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")
        }
    }
}
```

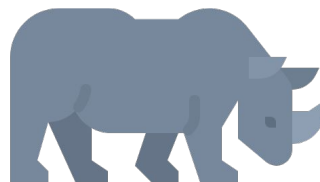
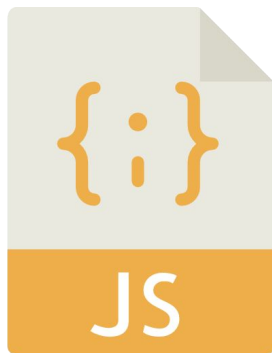
Чем исполнить JavaScript на JVM

- 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))
    """)
}
fun s
    this
    debugger
    delete
    do
    false
    for
    function
```

A photograph of a workspace with several old laptops lined up on a white desk. In the foreground, a laptop shows a macOS desktop with a sand dune wallpaper. To the right, a large monitor displays a terminal window with lines of code. The text "Старые ноутбуки в помощь!" is overlaid in the center.

Старые ноутбуки в помощь!

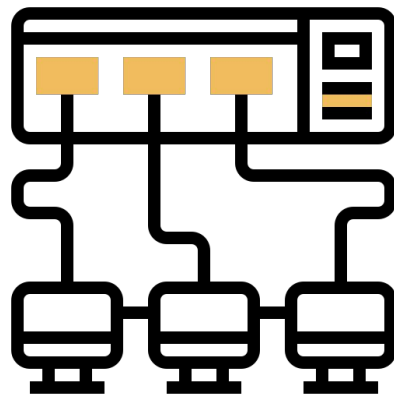
Парк машин

Robot-registry (наш Selenium + Selenium 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

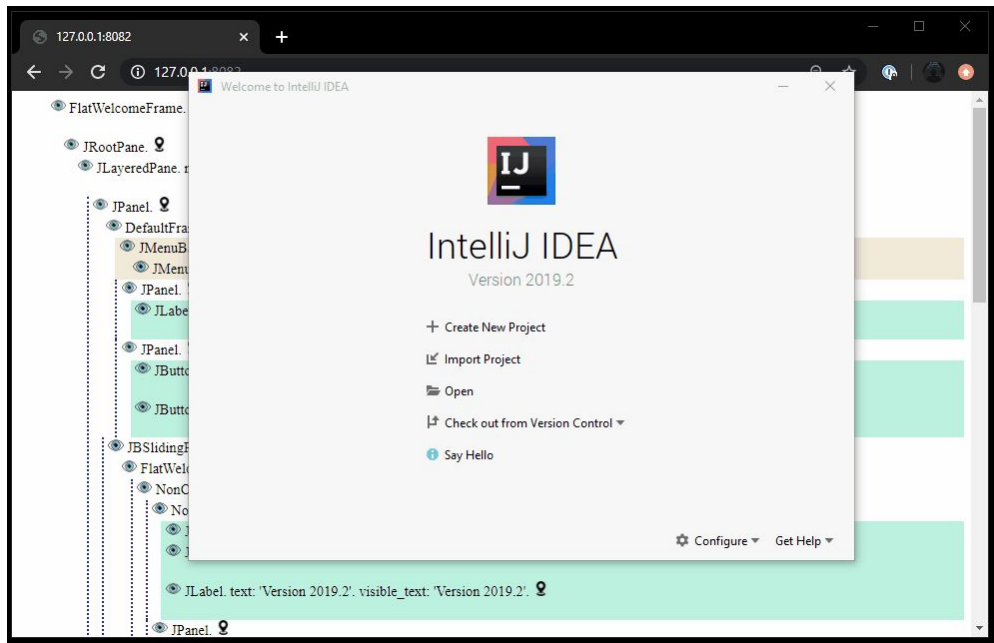
reserve delete more...	idea-img:latest http://172.31.150.195:18182	FavoritesTest	vnc noVnc
reserve delete more...	idea-img:latest http://172.31.144.150:18181	LicenseKeyActivationTest	vnc noVnc
reserve more...	intellij-windows10-ui-1() http://172.31.129.156:8180	KarmaDebugTest. Windows	vnc noVnc
reserve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

ComboBox.

DarculaComboBoxUI.

CellRendererPane.

SwingActionLink. text: 'How it works'. Visible text: 'How it works'.

JCheckBox. Visible text: 'Support screen readers (requires restart)'.

JCheckBox. 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

The screenshot displays a TestRail report for a failed test case. The browser address bar shows the URL: `ij-reporting.labs.jb.gg/report/WS%20license.webstorm-203.4574`.

Test Case Details:

- Test Case ID:** 704
- Test Case Name:** Search for License Activation Dialog
- Test Case Status:** failed
- Test Case Steps:**
 - Search 'JDialogFixture' by 'title WebStorm User Agreement' (153ms)
 - Search 'Checkbox' by 'accessibleName contains 'I confirm'' (99ms)
 - Click (1024ms)
 - Search 'Button' by 'text 'Continue'' (21ms)
 - Click (586ms)
 - Search 'JDialogFixture' by 'title Data Sharing' (16ms)
 - Search 'Button' by 'text 'Send Anonymous Statistics'' (15ms)
 - Click (552ms)

Test Case Screenshot:

The screenshot shows a Windows desktop environment with a dark theme. A 'License Activation' dialog box is open in the center, titled 'License Activation'. The dialog has a 'License Key' field and a 'Generate License' button. The background shows a file explorer window with a 'Test' folder selected.

History:

The history section shows a list of test runs for the same test case, with a total of 73833ms. The runs are categorized by status: passed (green) and failed (red).

Test Case ID	Test Case Name	Status	Steps	Time
webstorm-203.4449.17	Search for License Activation Dialog	passed	steps	5:45 8 october 2020
idea-u-203.4571	Search for License Activation Dialog	failed	steps	failed to generate license
webstorm-203.4510	Search for License Activation Dialog	passed	steps	7:06 7 october 2020
idea-u-203.4507	Search for License Activation Dialog	failed	steps	failed to generate license
idea-u-203.4449.2	Search for License Activation Dialog	passed	steps	9:49 6 october 2020
webstorm-203.4449	Search for License Activation Dialog	passed	steps	7:40 6 october 2020
webstorm-203.4449.5	Search for License Activation Dialog	passed	steps	7:01 6 october 2020
idea-u-203.4446	Search for License Activation Dialog	failed	steps	failed to generate license

Визуальные проверки

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



[pazone/ashot](https://github.com/pazone/ashot)

CheckUltimateUITest

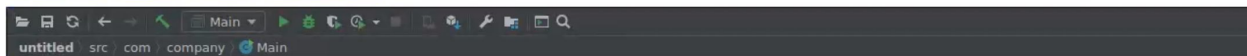
DARCULA Toolbar on

☒ 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 ☒ only actual



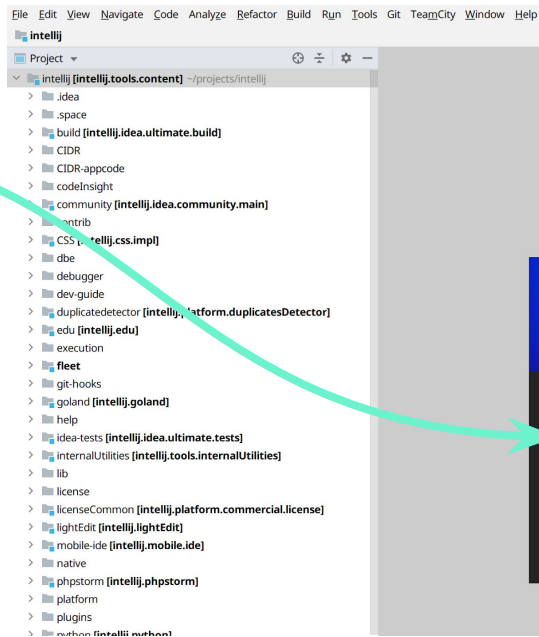
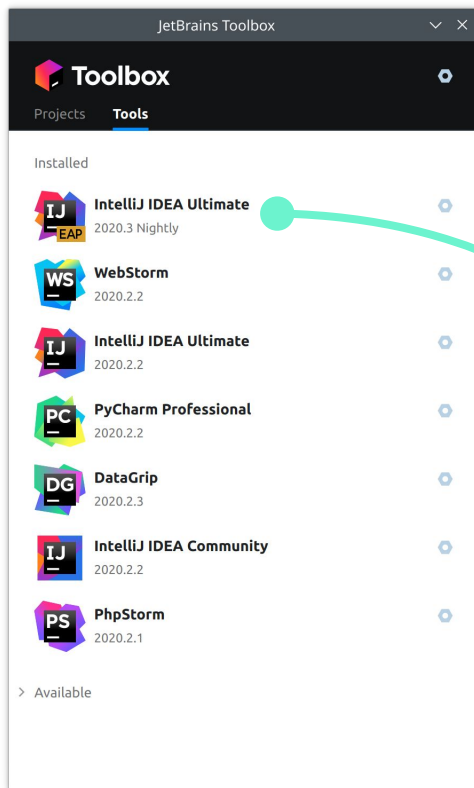
2 OS: mac Created: 9/21/2020, 7:25:22 AM on version: IDEA-U-203.3645.25 Last success: 10/8/2020, 5:49:24 AM ☒ use as expected image

Diff with ☒ only actual



Как работают команды IDE в JetBrains

Экстремальный догфудинг



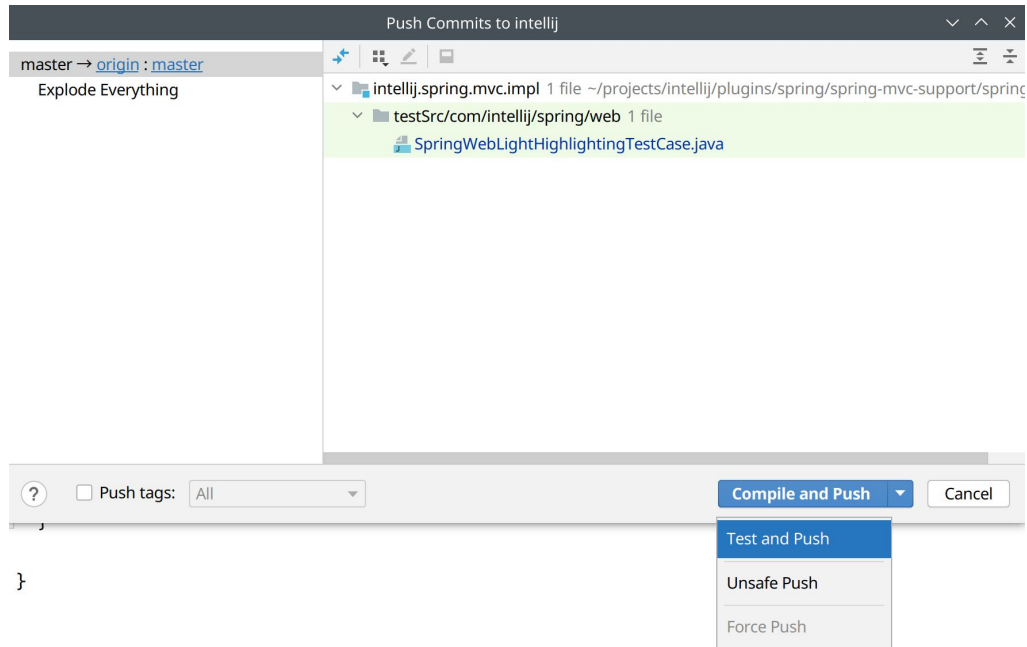
Много автоматизации на CI

- 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 запустить 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 использовании 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)

Gauge Overview Versions Reviews Custom Pages

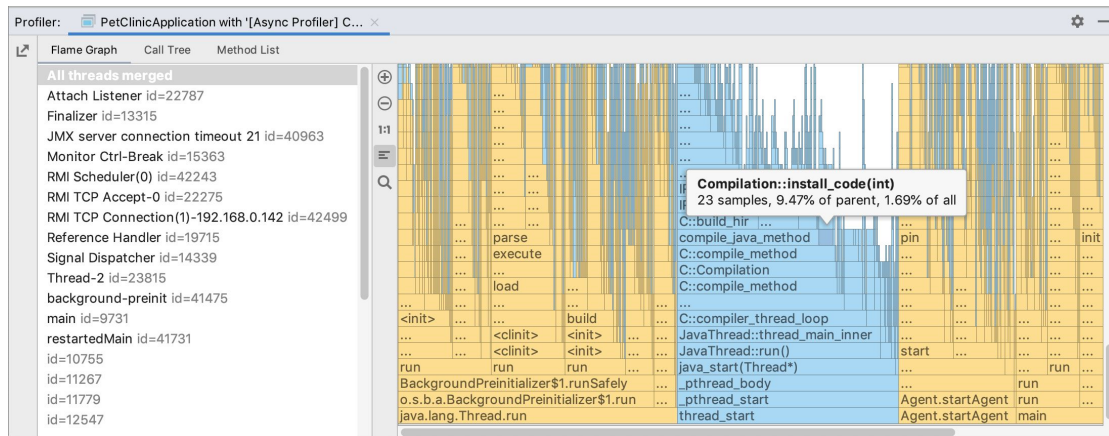
Compatibility verification Plugin was verified by IntelliJ Plugin Verifier

- ▼ **IntelliJ IDEA Ultimate IU-203.3645.34** Compatible. 13 usages of deprecated API. 13 usages of experimental API. 1 usage of internal API
 - ▼ **1 internal API usage**
Gauge 203.3645.34 uses internal API, which is not supposed to be used outside of the IntelliJ Platform.
 - ▶ **Internal method usage** (1)
 - ▼ **13 deprecated API usages**
Gauge 203.3645.34 uses deprecated API, which may be removed in future releases leading to binary and source code incompatibilities
 - ▼ **Deprecated methods usages** (8)
 - ▼ **DynamicBundle.getBundle(String, Class)** (3)
Deprecated method `DynamicBundle.getBundle(String, Class)` is invoked in `GaugeExecutionConfigurationSettingsEdito`
Deprecated method `DynamicBundle.getBundle(String, Class)` is invoked in `GaugeConfig.$$$setUpUI$$$()`
Deprecated method `DynamicBundle.getBundle(String, Class)` is invoked in `ExtractConceptDialog.$$$setUpUI$$$()`
 - ▶ `TransactionGuard.getContextTransaction()` (1)
 - ▶ `TransactionGuard.submitTransaction(...)` (1)
 - ▶ `Project.getBaseDir()` (1)
 - ▶ `JDOMEexternalizer.write(Element, String, boolean)` (1)
 - ▶ `JDOMEexternalizer.write(Element, String, String)` (1)
 - ▶ **Deprecated classes usages** (4)
 - ▶ **Deprecated constructor usage** (1)
 - ▶ **13 experimental API usages**
 - ▶ Dependencies used
 - ▶ Plugin can probably be enabled or disabled without IDE restart

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 displays a code editor with a Spot Profiler overlay. The code is a Java method `executeWriteAction` that takes `Editor editor`, `Caret caret`, and `DataContext dataContext` as parameters. The profiler highlights several lines of code with performance data:

- Line 48: `final PsiDocumentManager documentManager = PsiDocumentManager.getInstance(project);` is annotated with `177mks`.
- Line 54: `final MoverWrapper mover = getSuitableMover(editor, file);` is annotated with `787ms`.
- Line 58: `mover.move(editor, file);` is annotated with `552ms`.

The profiler also shows a vertical timeline on the left side of the code, indicating the execution flow and timing of the operations.

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](https://twitter.com/Yuriy_Artamonov)