




**Нужен рефакторинг?
Есть Idea!**

Heisenbug Moscow, 2018



**Меня зовут Артем
и я автоматизатор...**


Привет, Артем



Автоматизация тестирования

The image shows a variety of gift boxes and cards. On the left, there are stacks of boxes with floral patterns, some with the text 'Live LAUGH Love'. In the center, there are boxes with a blue and white floral design and a box with a bicycle pattern. On the right, there are tall stacks of solid-colored boxes in red, white, and green, and a display of many colorful cards.

Много разных проектов

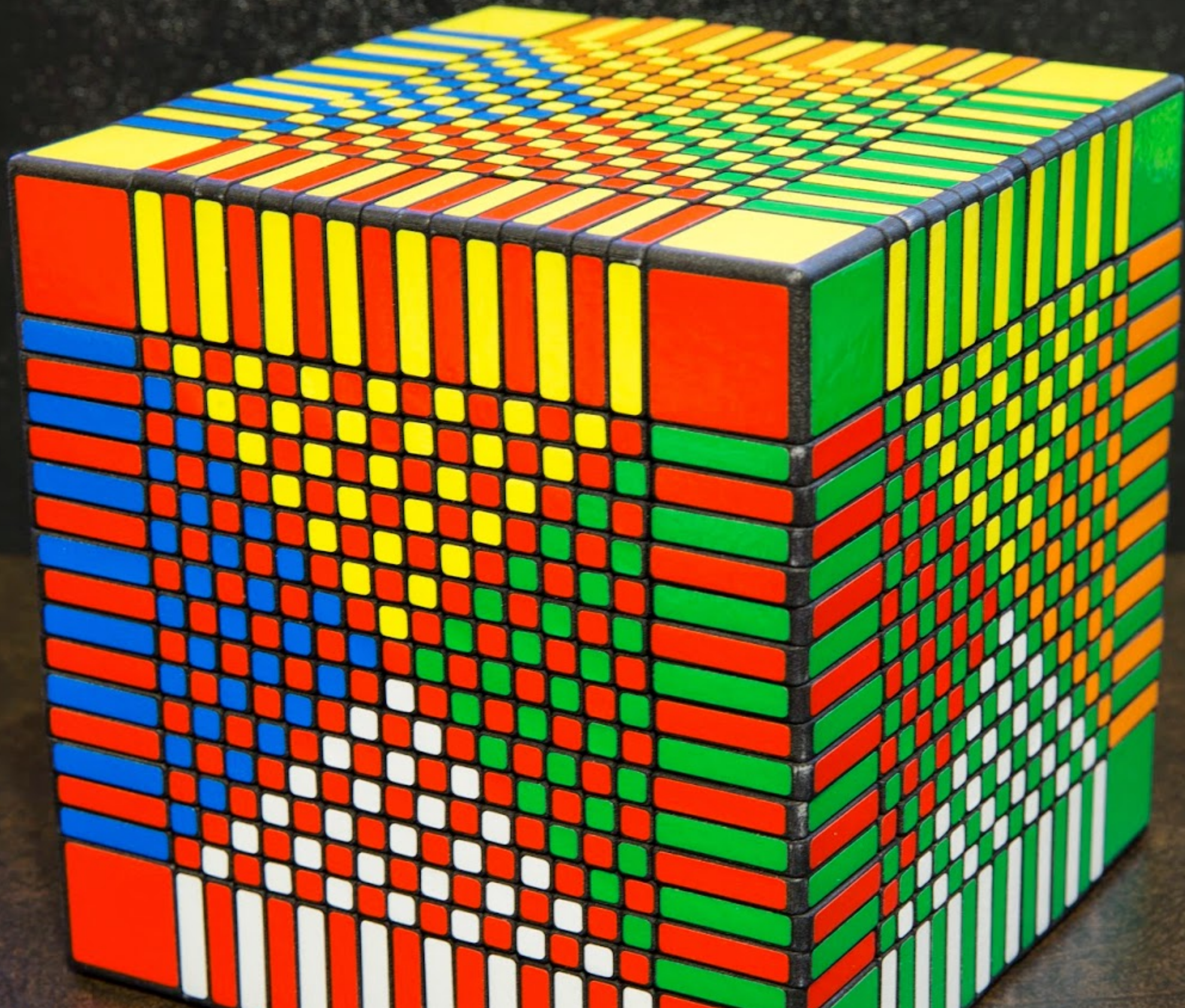


В каждом проекте сюрприз

BEVOСHИГЪ



СВОЖИПІЇ



ЖОНСІЉ

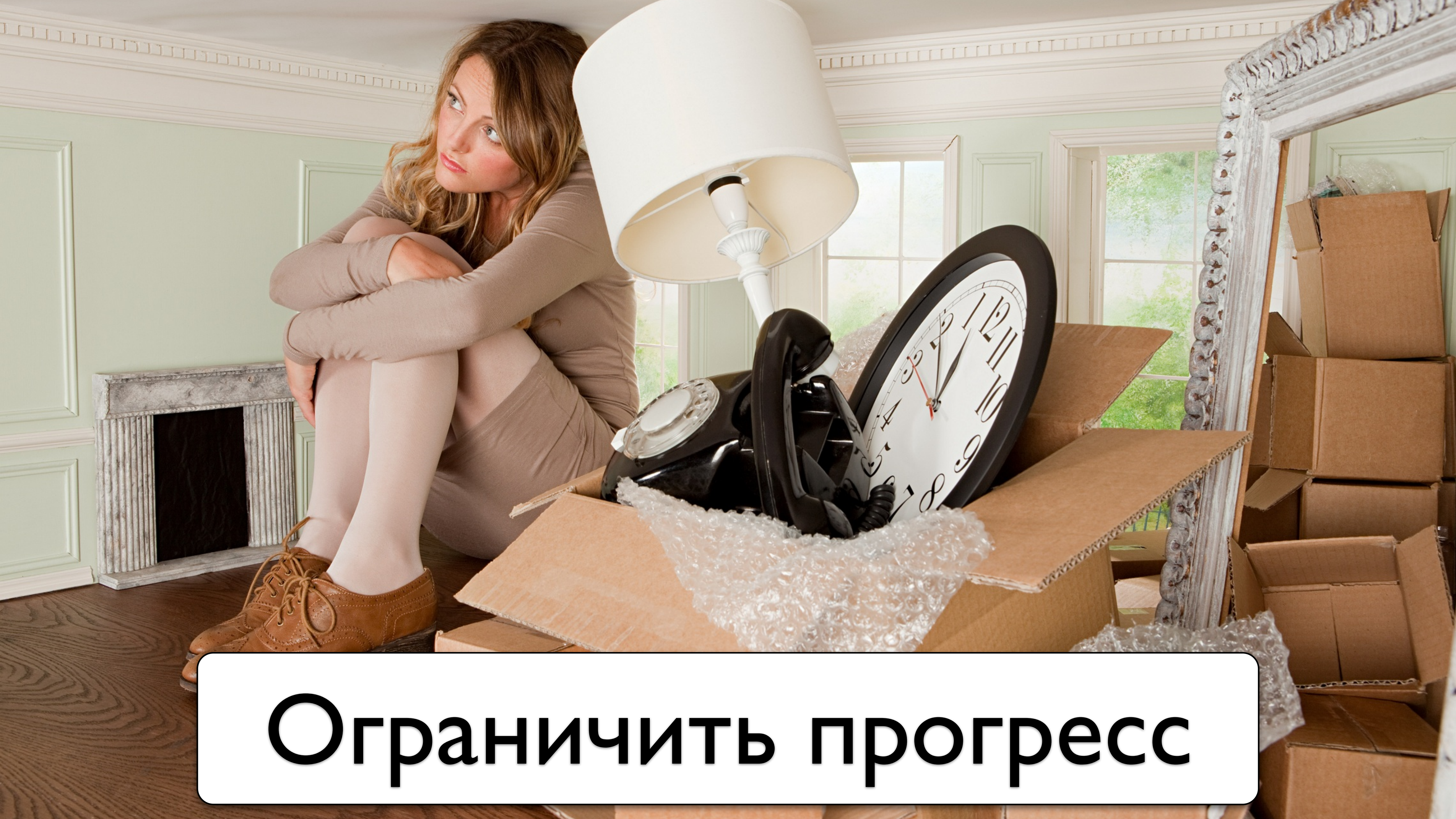


Расставить все по местам



Как обеспечить равномерность?





Ограничить прогресс

Быстро догонять




Добавить пару рук



Совершенствовать орудие труда





Я видел много инструментов



Возможности Idea Plugin



Тесты катать - не ракеты строить



Тесты катать - не ракеты строить

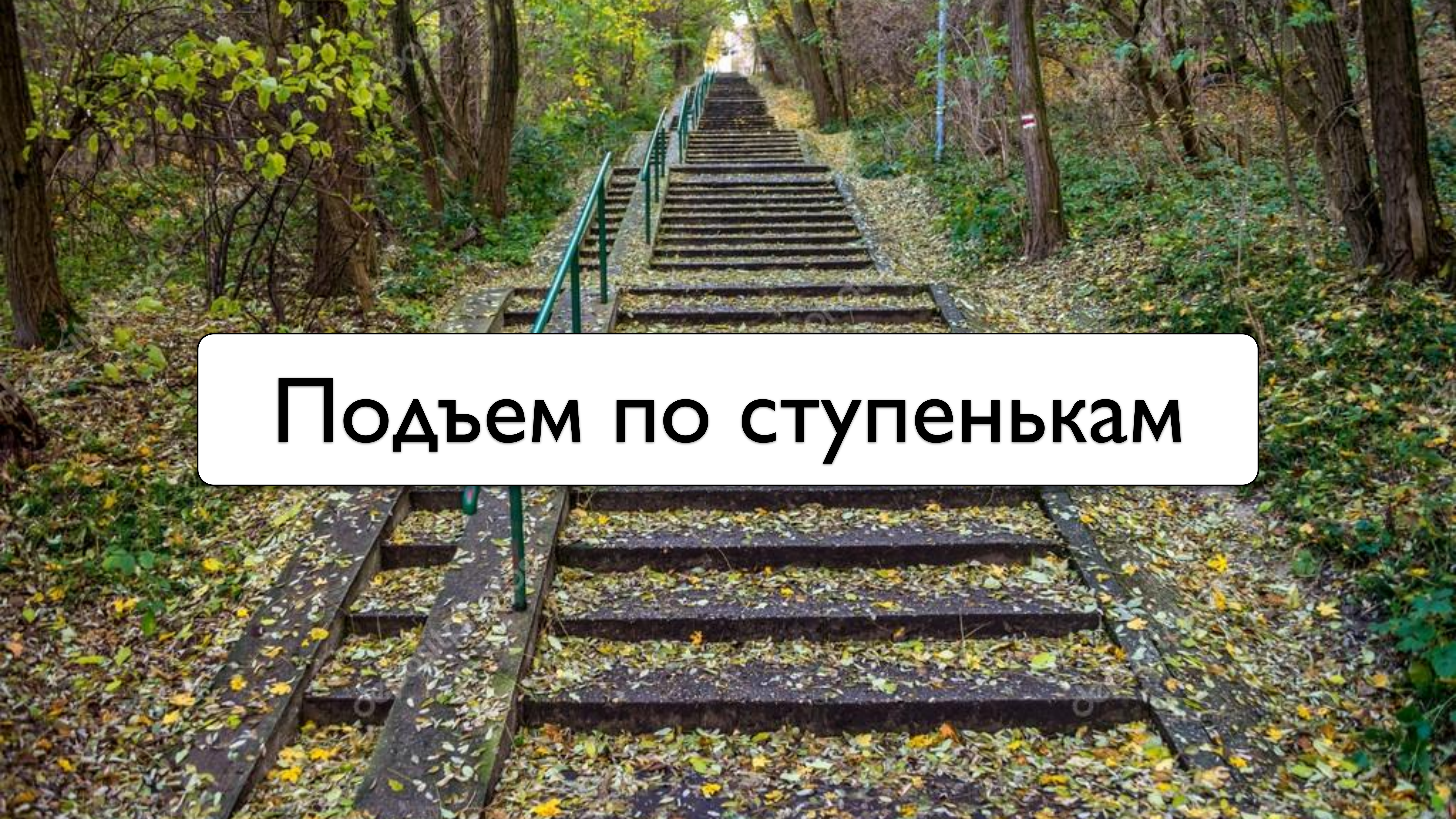
Атомарные

Похожие

Простые

Доктор, я перевожу
тесты на **jUnit5** силой мысли,
это нормально?





Подъём по ступенькам



Вперед к приключениям

Импорт в проект


Экспорт из проекта

Миграция проекта

Импорт в проект

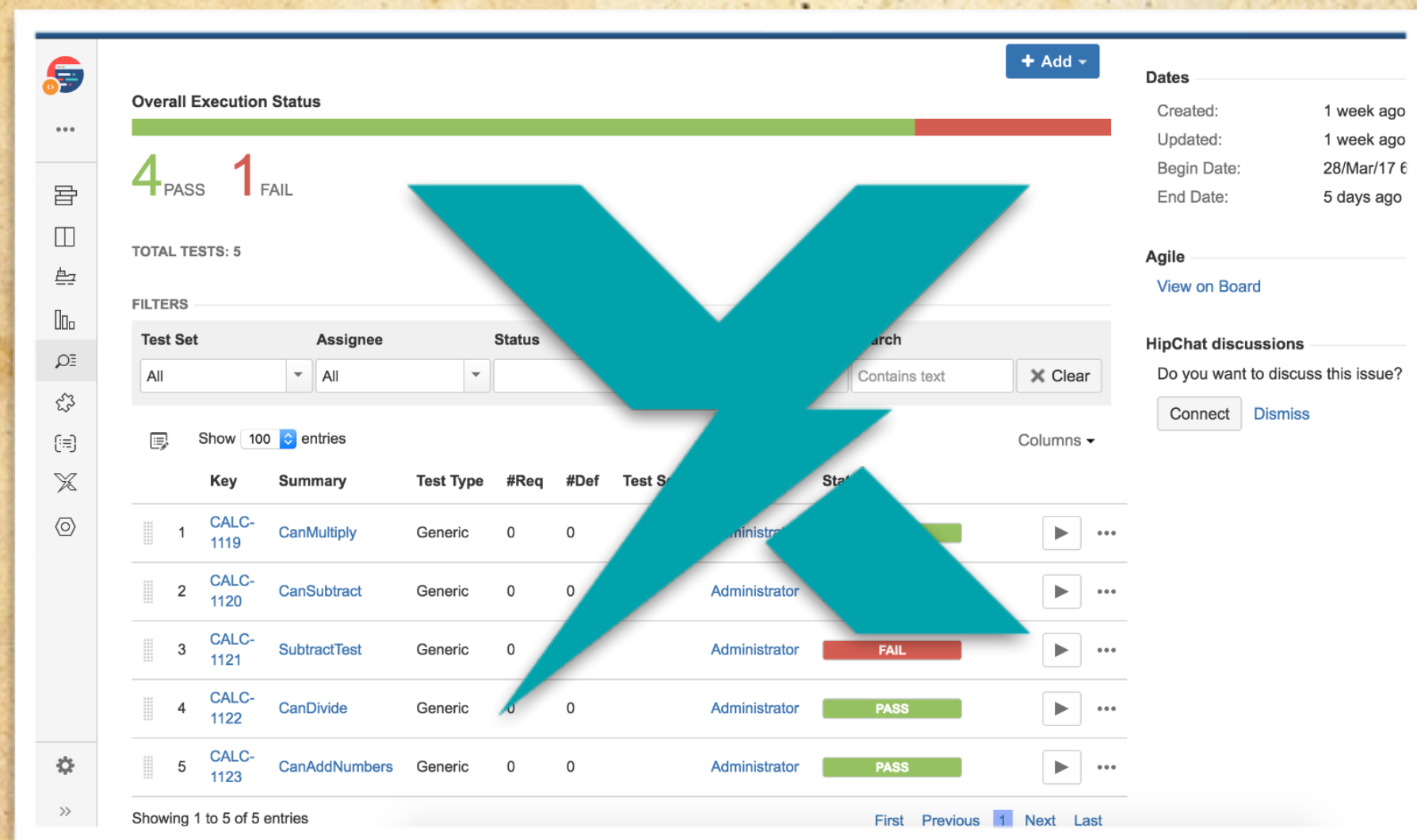
Экспорт из проекта

Миграция проекта



Какую проблему решаем?

Что автоматизировано?



Overall Execution Status

4 PASS 1 FAIL

TOTAL TESTS: 5

FILTERS

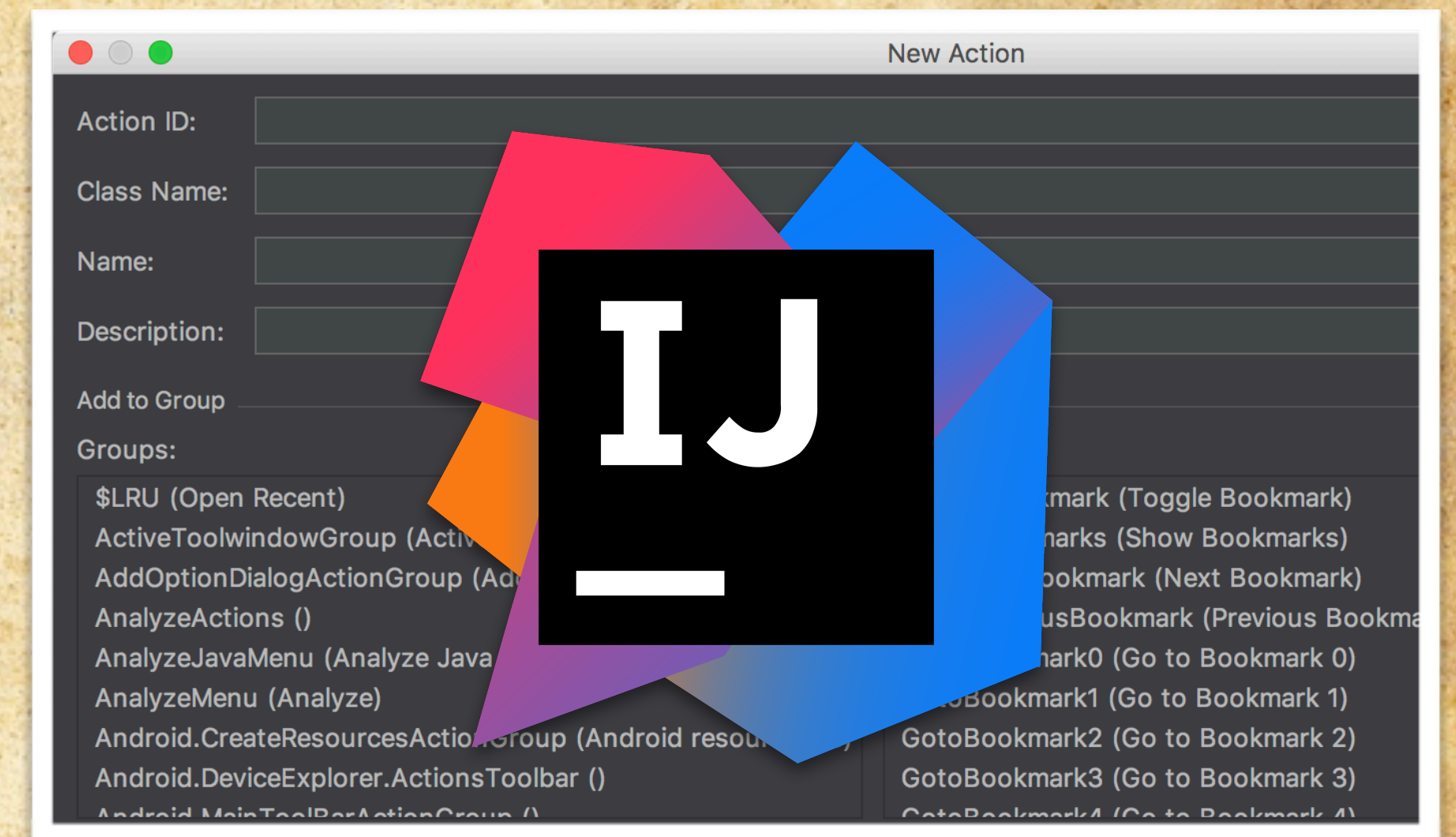
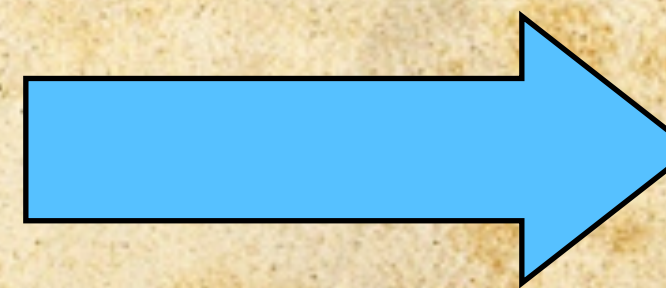
Test Set	Assignee	Status
All	All	

Search: Contains text [X] Clear

Show 100 entries

Key	Summary	Test Type	#Req	#Def	Test S	Stat
1	CALC-1119 CanMultiply	Generic	0	0	Administrator	PASS
2	CALC-1120 CanSubtract	Generic	0	0	Administrator	PASS
3	CALC-1121 SubtractTest	Generic	0	0	Administrator	FAIL
4	CALC-1122 CanDivide	Generic	0	0	Administrator	PASS
5	CALC-1123 CanAddNumbers	Generic	0	0	Administrator	PASS

Showing 1 to 5 of 5 entries



New Action

Action ID: []

Class Name: []

Name: []

Description: []

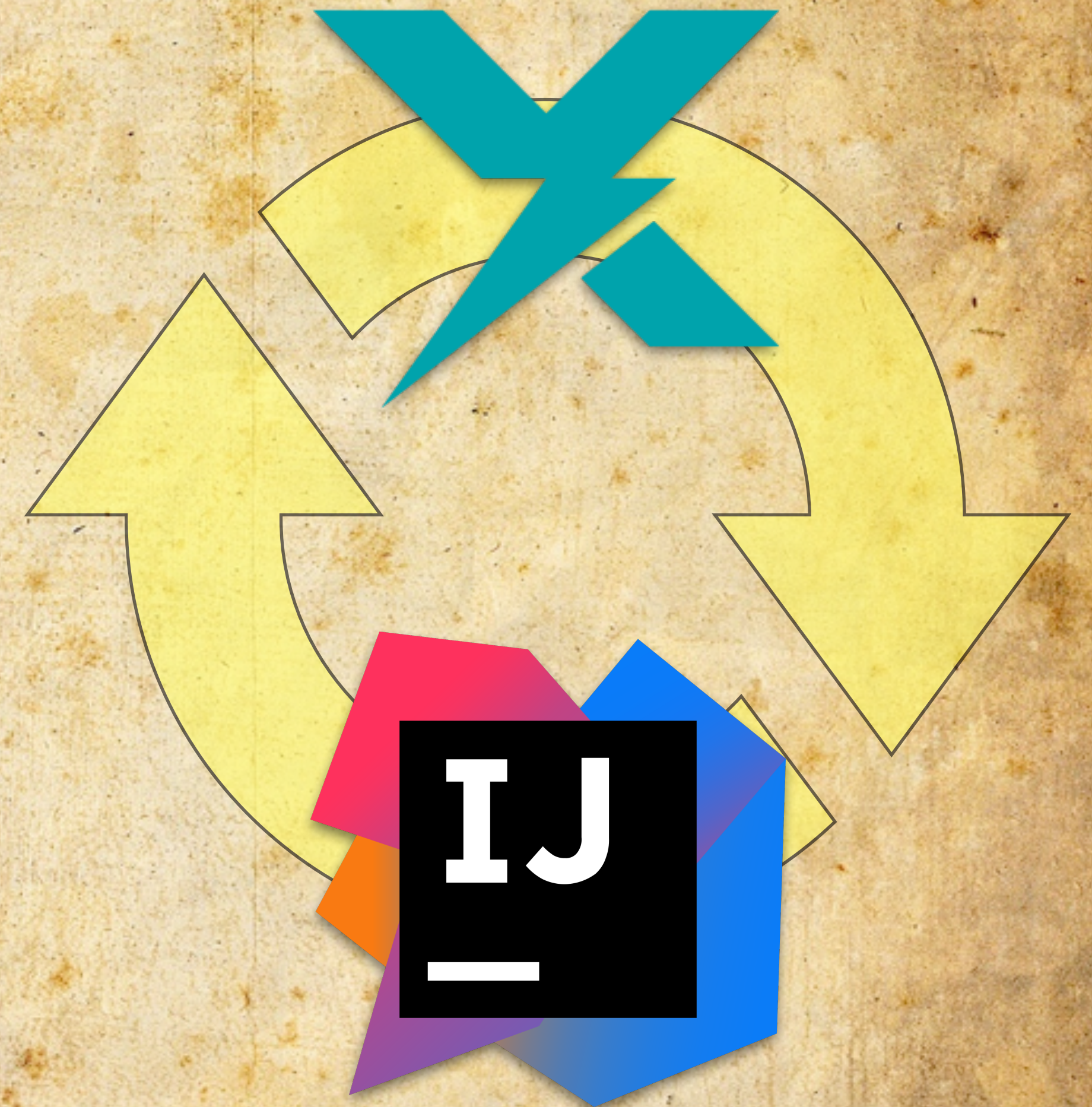
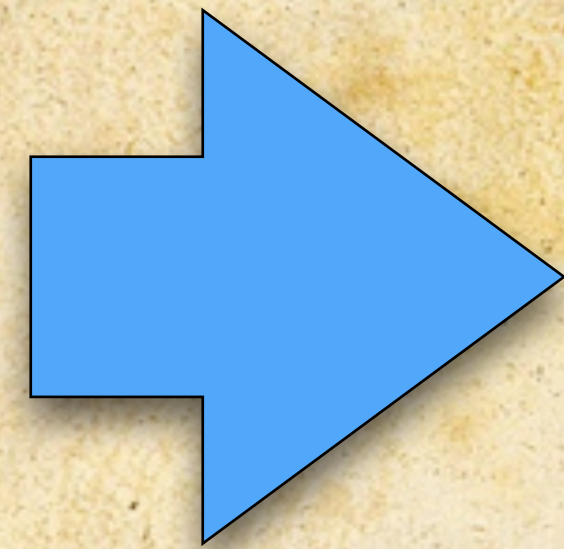
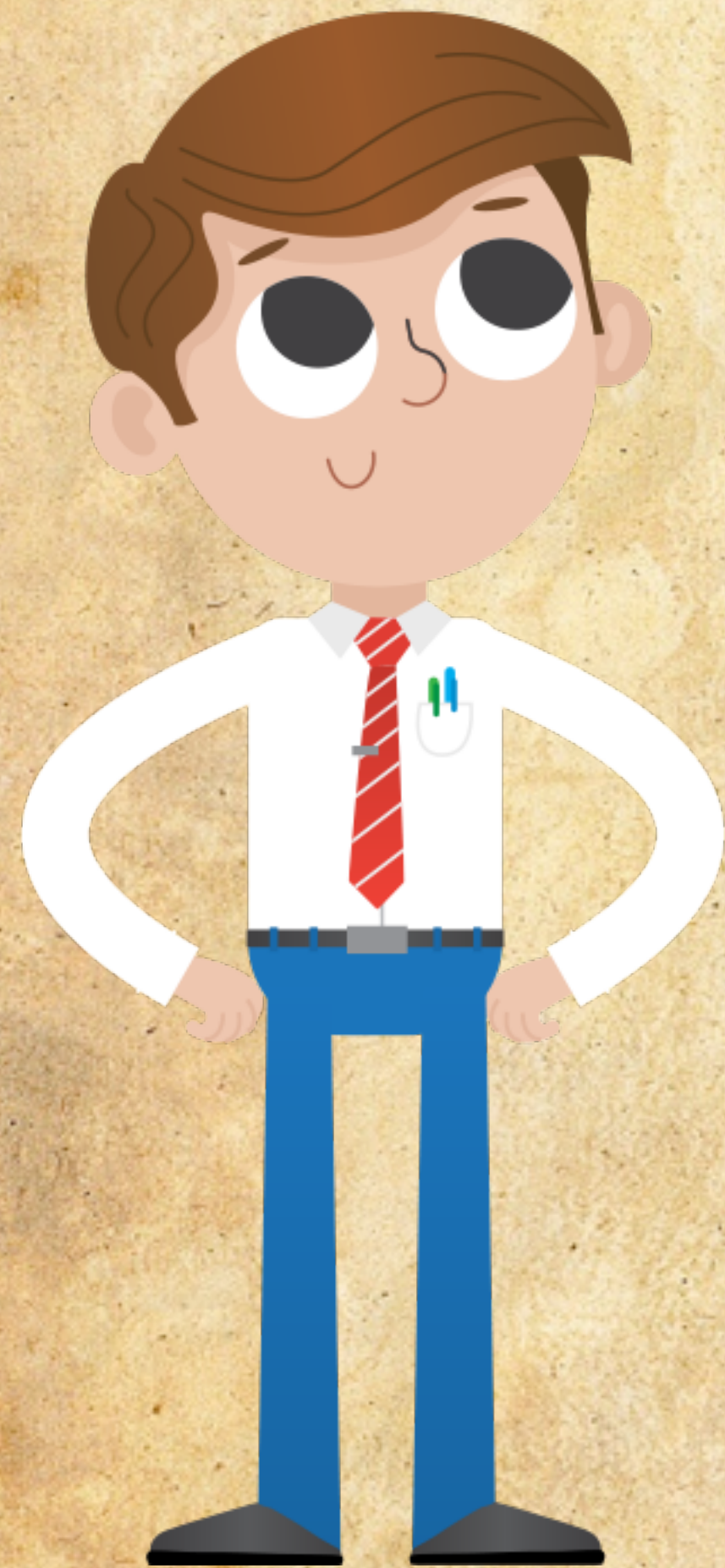
Add to Group

Groups:

- \$LRU (Open Recent)
- ActiveToolwindowGroup (ActiveToolwindowGroup)
- AddOptionDialogActionGroup (AddOptionDialogActionGroup)
- AnalyzeActions ()
- AnalyzeJavaMenu (Analyze Java)
- AnalyzeMenu (Analyze)
- Android.CreateResourcesActionGroup (Android resour)
- Android.DeviceExplorer.ActionsToolBar ()
- Android.MainToolBarActionGroup (Android.MainToolBarActionGroup)
- Bookmark (Toggle Bookmark)
- Bookmarks (Show Bookmarks)
- Bookmark (Next Bookmark)
- PreviousBookmark (Previous Bookmark)
- Bookmark0 (Go to Bookmark 0)
- Bookmark1 (Go to Bookmark 1)
- Bookmark2 (Go to Bookmark 2)
- Bookmark3 (Go to Bookmark 3)
- Bookmark4 (Go to Bookmark 4)



Задача на автоматизацию



Возьмем название теста

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @DisplayName("Добавление в избранное после создания заметки")  
    public void shouldAddToFavoritesAfterNodeTest() { ... }  
  
    @Test  
    @DisplayName("Удаление из избранного после удаления заметки")  
    public void shouldDeleteToFavoritesAfterNodeTest() { ... }  
  
}
```

Поиск по тексту в Jira

The screenshot shows the Jira web interface. At the top, there is a navigation bar with the Jira logo and menu items: Dashboards, Projects, Issues, Create, Search, and a help icon. Below the navigation bar, the search results page is displayed. The search bar contains the query "text ~ "Добавление в избранное после создания заметки"". The search results show a single issue from the project "Allure Examples / AE-5". The issue title is "Добавление в избранное после создания заметки". The issue type is "Task" and the status is "TO DO". The assignee is "Unassigned". The issue has several action buttons: Edit, Comment, Assign, More, Start Progress, Done, and Admin. The left sidebar shows the "Order by" dropdown menu with "AE-5" selected.


JIRA Dashboards ▾ Projects ▾ Issues ▾ **Create** Search 🔍 🔊 ? ▾

» Search

🟢 text ~ "Добавление в избранное после создания заметки" ? 🔍 [Basi](#)

Order by ▾

- AE-5
- Добавление в избранное после создан...

 **Allure Examples / AE-5**


Добавление в избранное после создания заметки

Details

Type: Task

Status: **TO DO** ([View Workflow](#))

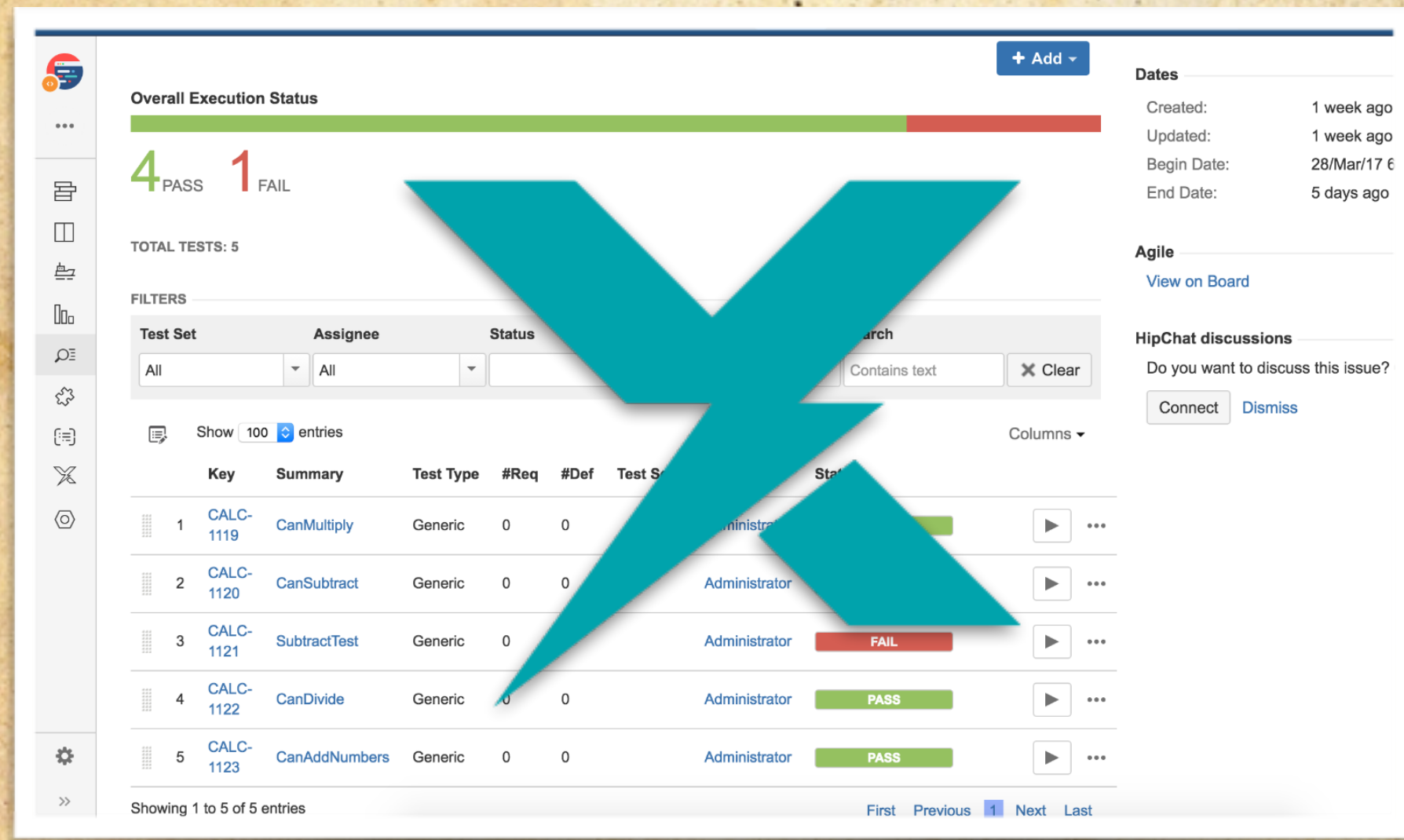
People

Assignee:  Unassigned

Нужно расставить ссылки

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @TmsLink("АЕ-5")  
    @DisplayName("Добавление в избранное после создания заметки")  
    public void shouldAddToFavoriteAfterNodeTest() { ... }  
  
    @Test  
    @TmsLink("АЕ-4")  
    @DisplayName("Удаление из избранного после удаления заметки")  
    public void shouldDeleteToFavoriteAfterNodeTest() { ... }  
  
}
```

Для всех тестов сложно



Overall Execution Status

4 PASS 1 FAIL

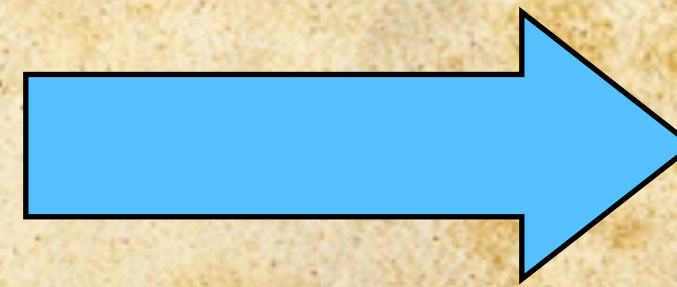
TOTAL TESTS: 5

FILTERS

Test Set	Assignee	Status
All	All	

Key	Summary	Test Type	#Req	#Def	Test S	Stat
1	CALC-1119 CanMultiply	Generic	0	0	Administrator	PASS
2	CALC-1120 CanSubtract	Generic	0	0	Administrator	PASS
3	CALC-1121 SubtractTest	Generic	0	0	Administrator	FAIL
4	CALC-1122 CanDivide	Generic	0	0	Administrator	PASS
5	CALC-1123 CanAddNumbers	Generic	0	0	Administrator	PASS

Showing 1 to 5 of 5 entries



New Action

Action ID:

Class Name:

Name:


Description:

Add to Group

Groups:

- \$LRU (Open Recent)
- ActiveToolwindowGroup (ActiveToolwindowGroup)
- AddOptionDialogActionGroup (AddOptionDialogActionGroup)
- AnalyzeActions ()
- AnalyzeJavaMenu (Analyze Java)
- AnalyzeMenu (Analyze)
- Android.CreateResourcesActionGroup (Android resour)
- Android.DeviceExplorer.ActionsToolBar ()
- Android.MainToolBarActionGroup ()
- Bookmark (Toggle Bookmark)
- Bookmarks (Show Bookmarks)
- Bookmark (Next Bookmark)
- PreviousBookmark (Previous Bookma
- Bookmark0 (Go to Bookmark 0)
- Bookmark1 (Go to Bookmark 1)
- Bookmark2 (Go to Bookmark 2)
- Bookmark3 (Go to Bookmark 3)
- Bookmark4 (Go to Bookmark 4)



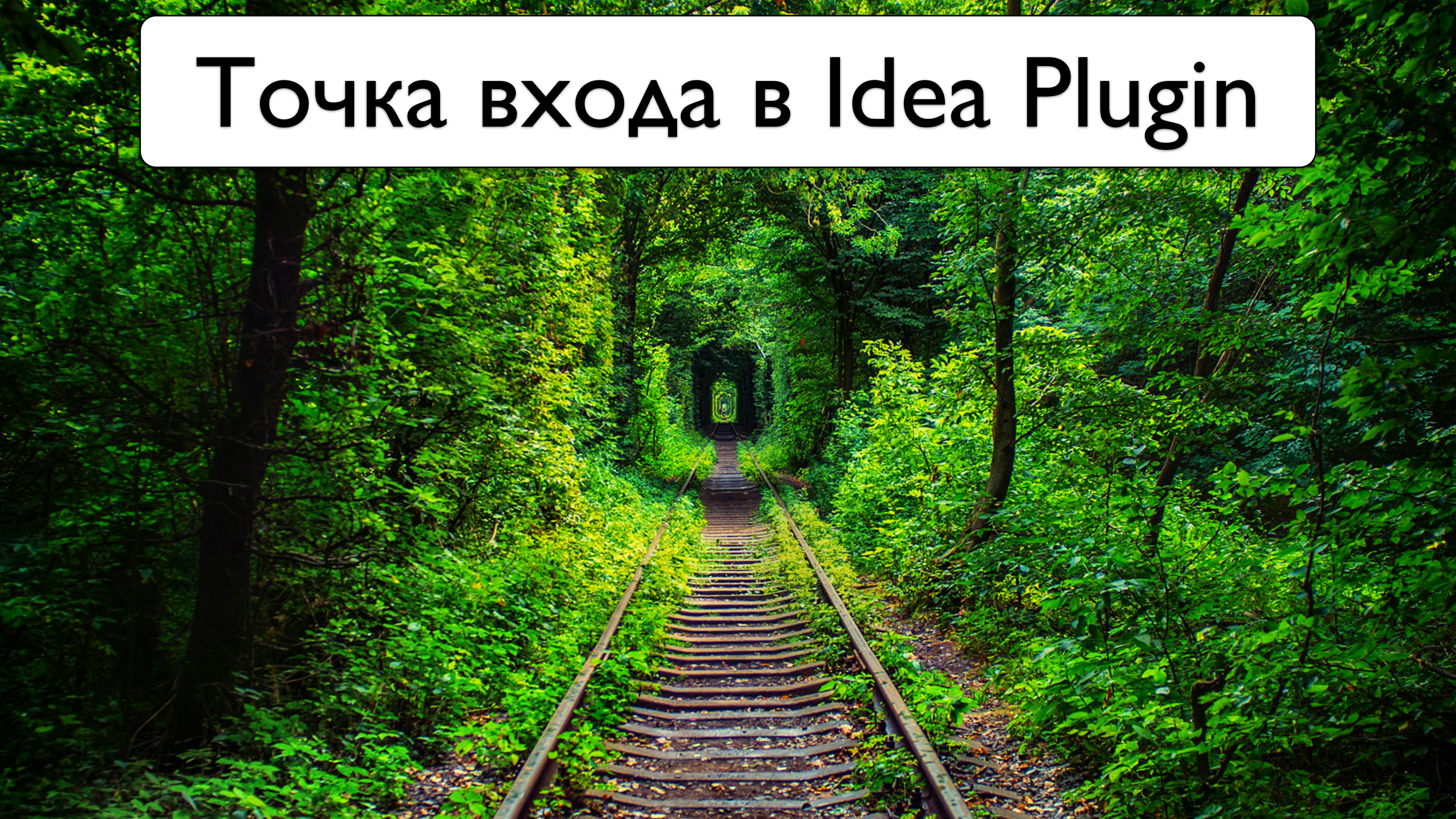
A 3D maze made of white rectangular blocks. A red path is highlighted, starting from the top left and winding through several loops. At the bottom right, a red arrow points out of the maze, indicating the exit. The text 'Способ решения проблемы' is overlaid on a white rounded rectangle in the center of the maze.

Способ решения проблемы

Idea Plugin



Точка входа в Idea Plugin



Точка ВХОДА в Idea Plugin

```
public class AnAction {  
  
    @Override  
    public void actionPerformed(AnActionEvent event) {  
        //MAGIC here  
    }  
  
}
```

Очень похоже на Reflection



PsiClass - работа с классом

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @TmsLink("АЕ-5")  
    @DisplayName("Добавление в избранное после создания заметки")  
    public void shouldAddToFavoritesAfterNodeTest() { ... }  
  
    @Test  
    @TmsLink("АЕ-4")  
    @DisplayName("Удаление из избранного после удаления заметки")  
    public void shouldDeleteToFavoritesAfterNodeTest() { ... }  
  
}
```

PsiMethod - работа с методами

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @TmsLink("АЕ-5")  
    @DisplayName("Добавление в избранное после создания заметки")  
    public void shouldAddToFavoritesAfterNodeTest() { ... }  
  
    @Test  
    @TmsLink("АЕ-4")  
    @DisplayName("Удаление из избранного после удаления заметки")  
    public void shouldDeleteToFavoritesAfterNodeTest() { ... }  
  
}
```

PsiAnnotation - аннотации

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @TmsLink("AE-5")  
    @DisplayName("Добавление в избранное после создания заметки")  
    public void shouldAddToFavoriteAfterNodeTest() { ... }  
  
    @Test  
    @TmsLink("AE-4")  
    @DisplayName("Удаление из избранного после удаления заметки")  
    public void shouldDeleteToFavoriteAfterNodeTest() { ... }  
  
}
```



Давайте запилим плагин!

Создаем JiraKeyImportAction

```
public class JiraKeyImportAction extends AnAction {  
  
    @Override  
    public void actionPerformed(AnActionEvent event) {  
        // Magic here  
    }  
  
}
```


AnActionEvent - ЭТО КОНТЕКСТ

```
public class JiraKeyImportAction extends AnAction {  
  
    @Override  
    public void actionPerformed(AnActionEvent event) {  
  
        PsiElement element = event.getData(PSI_ELEMENT)  
  
    }  
  
}
```

Работаем ТОЛЬКО С КЛАССАМИ

```
public class JiraKeyImportAction extends AnAction {  
  
    @Override  
    public void actionPerformed(AnActionEvent event) {  
        PsiElement element = event.getData(PSI_ELEMENT)  
  
        if (element instanceof PsiClass) {  
            addTmsLinkToClassMethods((PsiClass)element)  
        }  
  
    }  
  
}
```

Выбрали класс для апдейта

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @DisplayName("Добавление в избранное после создания заметки")  
    public void shouldAddToFavoritesAfterNodeTest() { ... }  
  
    @Test  
    @DisplayName("Удаление из избранного после удаления заметки")  
    public void shouldDeleteToFavoritesAfterNodeTest() { ... }  
  
    private void utilityMethod() { ... }  
  
}
```

Получаем все методы класса

```
public class JiraKeyImportAction extends AnAction {  
  
    void addTmsLinkToClassMethods(PsiClass testClass) {  
  
        Arrays.stream(testClass.getMethods())  
            .filter(m -> m.hasAnnotation("org..Test"))  
            .filter(m -> m.hasAnnotation("org..DisplayName"))  
            .forEach(this::addTmsLinkToMethod)  
  
    }  
}
```

Фильтруем тестовые методы

```
public class JiraKeyImportAction extends AnAction {  
    void addTmsLinkToClassMethods(PsiClass testClass) {  
        Arrays.stream(testClass.getMethods())  
            .filter(m -> m.hasAnnotation("org.Test"))  
            .filter(m -> m.hasAnnotation("org.DisplayName"))  
            .forEach(this::addTmsLinkToMethod)  
    }  
}
```

Выбрали методы для апдейта

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @DisplayName("Добавление в избранное после создания заметки")  
    public void shouldAddToFavoritesAfterNoteTest() { ... }  
  
    @Test  
    @DisplayName("Удаление из избранного после удаления заметки")  
    public void shouldDeleteToFavoritesAfterNoteTest() { ... }  
  
    private void utilityMethod() { ... }  
  
}
```

Берем аннотацию DisplayName

```
public class JiraKeyImportAction extends AnAction {  
  
    void addTmsLinkToMethod(PsiMethod method) {  
  
        PsiAnnotation nameAnnotation =  
            method.getAnnotation("org.DisplayName");  
        String text = name  
            .findDeclaredAttributeValue("value").getText()  
        String key = jiraClient.findByText(text);  
        addTmsLinkToMethod(testMethod, key);  
    }  
}
```

Нашли аннотацию с именем

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @DisplayName("Добавление в избранное после создания заметки")  
    public void shouldAddToFavoritesAfterNoteTest() { ... }  
  
    @Test  
    @DisplayName("Удаление из избранного после удаления заметки")  
    public void shouldDeleteToFavoritesAfterNoteTest() { ... }  
  
    private void utilityMethod() { ... }  
  
}
```


Берем текст DisplayName

```
public class JiraKeyImportAction extends AnAction {  
  
    void addTmsLinkToMethod(PsiMethod testMethod) {  
  
        PsiAnnotation name =  
            method.getAnnotation("org..DisplayName");  
        String text = name  
            .findDeclaredAttributeValue("value").getText()  
        String key = jiraClient.findByText(text);  
        addTmsLinkToMethod(testMethod, key);  
    }  
}
```

Тест для поиска в Jira

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @DisplayName("Добавление в избранное после создания заметки")  
    public void shouldAddToFavoritesAfterNodeTest() { ... }  
  
    @Test  
    @DisplayName("Удаление из избранного после удаления заметки")  
    public void shouldDeleteToFavoritesAfterNodeTest() { ... }  
  
    private void utilityMethod() { ... }  
  
}
```

Ищем Issue по тексту в Jira

```
public class JiraKeyImportAction extends AnAction {  
  
    void addTmsLinkToMethod(PsiMethod testMethod) {  
  
        PsiAnnotation name =  
            method.getAnnotation("org..DisplayName");  
        String text = name  
            .findDeclaredAttributeValue("value").getText();  
        String key = jiraClient.findByText(text);  
        addTmsLinkToMethod(testMethod, key);  
    }  
}
```

Посмотрим на Jira API

Search for issues using JQL (G... ✕ ▾)

Create screen tab

Update screen tab

Delete screen tab

Get all screen tab fields

Add screen tab field

Remove screen tab field

Move screen tab field

Move screen tab

▾ Search

Search for issues using JQL (GET)

Search for issues using JQL (POST)

Search for issues using JQL (GET)

GET /rest/api/3/search

Searches for issues using [JQL](#) [↗]. **Permissions** [↗] **required:** Permission to access Jira.

If the JQL query expression is too large to be encoded as a query parameter, use the [POST](#) version of this resource.

App scope required: `READ`

OAuth scopes required: `read:jira-work`

Request

QUERY PARAMETERS

`jql`

`string`

Поиск по тексту в Jira

The screenshot shows the Jira interface with a search bar containing the query "text ~ \"Добавление в избранное после создания заметки\"". The search results display a single issue from the "Allure Examples / AE-5" project. The issue title is "Добавление в избранное после создания заметки". The issue type is "Task" and the status is "TO DO". The assignee is "Unassigned". The interface includes navigation menus at the top, a search bar, and a list of actions for the issue.

JIRA Dashboards ▾ Projects ▾ Issues ▾ **Create** Search 🔍 🔊 ⓘ ▾

» Search Save as Share Export

✔ text ~ "Добавление в избранное после создания заметки" ⓘ 🔍 Basi

Order by ▾

- ✔ AE-5
Добавление в избранное после создан...

Allure Examples / AE-5

Добавление в избранное после создания заметки

Edit Comment Assign More ▾ Start Progress Done Admin ▾

Details

Type: ✔ Task

Status: **TO DO** (View Workflow)

People

Assignee: Unassigned

Добавляем аннотацию в код

```
public class JiraKeyImportAction extends AnAction {  
  
void addTmsLinkToMethod(PsiMethod testMethod) {  
  
    PsiAnnotation name =  
        method.getAnnotation("org..DisplayName");  
    String text = name  
        .findDeclaredAttributeValue("value").getText()  
    String key = jiraClient.findByText(text);  
    addTmsLinkToMethod(testMethod, key);  
}  
}
```

Формируем текст аннотации

```
public class JiraKeyImportAction extends AnAction {  
  
    void addTmsLinkToMethod(PsiMethod method, String key) {  
  
        String tmsLinksAnnotationText =  
            "@io..TmsLink(\"\" + key + "\")";  
        PsiAnnotation tmsLink =  
            method.addAnnotation(tmsLinkAnnotationText);  
        JavaCodeStyleManager.getInstance(project)  
            .shortenClassReferences(tmsLink);  
    }  
}
```

Добавляем аннотацию

```
public class JiraKeyImportAction extends AnAction {  
  
    void addTmsLinkToMethod(PsiMethod method, String key) {  
  
        String tmsLinksAnnotationText =  
            "@io..TmsLink(\"\" + key + "\")";  
        PsiAnnotation tmsLink =  
            method.addAnnotation(tmsLinkAnnotationText);  
        JavaCodeStyleManager.getInstance(project)  
            .shortenClassReferences(tmsLink);  
    }  
}
```


Добавили аннотацию

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @io.gameta.allure.TmsLink("АЕ-5")  
    @DisplayName("Добавление в избранное после создания заметки")  
    public void shouldAddToFavoritesAfterNodeTest() { ... }  
  
    @Test  
    @DisplayName("Удаление из избранного после удаления заметки")  
    public void shouldDeleteToFavoritesAfterNodeTest() { ... }  
  
}
```

Формируем текст аннотации

```
public class JiraKeyImportAction extends AnAction {  
  
    void addTmsLinkToMethod(PsiMethod method, String key) {  
  
        String tmsLinksAnnotationText =  
            "@io..TmsLink(\"\" + key + "\")";  
        PsiAnnotation tmsLink =  
            method.addAnnotation(tmsLinkAnnotationText);  
        JavaCodeStyleManager.getInstance(project)  
            .shortenClassReferences(tmsLink);  
    }  
}
```

Добавили импорт аннотации

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @TmsLink("AE-5")  
    @DisplayName("Добавление в избранное после создания заметки")  
    public void shouldAddToFavoritesAfterNodeTest() { ... }  
  
    @Test  
    @DisplayName("Удаление из избранного после удаления заметки")  
    public void shouldDeleteToFavoritesAfterNodeTest() { ... }  
  
}
```

A close-up photograph of Aragorn from 'The Lord of the Rings' with a speech bubble overlaid on the right side. He has long brown hair and a beard, and is wearing a dark green tunic. He is holding a small dark object in his right hand. The background is a blurred, warm-toned interior.

**Нельзя просто взять
и изменить код!**

Обертка над изменением кода

```
public class JiraKeyImportAction extends AnAction {  
  
    void addTmsLinkToMethod(PsiMethod method, String key) {  
  
        CommandProcessor.getInstance().executeCommand(() -> {  
            getApplication().runWriteAction(() -> {  
                //код с предыдущего слайда  
            })  
        })  
    })  
}  
}
```



Open issues [Switch filter ▾](#)

[View all issues and filters](#)

Order by Priority ▾

- AE-5
Добавление в избранное после создания за...
- AE-4
Удаление из избранного после удаления за...
- AE-6
Авторизация пользователя



Allure Examples / AE-4

2 of 3 ▲ ▼ ↗

Удаление из избранного после удаления заметки

Details

Type: Task

Status: **TO DO** ([View Workflow](#))

Priority: ↑ Medium

Resolution: Unresolved

Labels: [favorites](#)

People

Assignee:

Unassigned
[Assign to me](#)

Reporter:

admin

Votes:

0

Watchers:

1 [Stop watching this issue](#)

Description

Click to add description

Attachments

Drop files to attach, or [browse](#).

INSTALL



ENGINE



DRIVETRAIN



INDUCTION



SUSPENSION



BRAKES



NITRO



TIRES

Что еще можно прокачать?

215 KPH

20 SEC

2,85

HANDLING

POWER 334 HP

BRAKING 4,50

WEIGHT 3578 KG

NITRO 1,29

**Теги для
запуска тестов**



Labels для разметки тестов

The image shows a JIRA issue page for 'Добавление заметки' (Add note) in the 'Allure Examples / AE-5' project. The issue is currently in the 'Unresolved' state and has two labels: 'favorites' and 'regress'. A callout box highlights the 'Resolution: Unresolved' and 'Labels: favorites regress' fields. The 'Details' section shows the issue type as 'Task', status as 'TO DO', priority as 'Medium', and resolution as 'Unresolved'. The 'People' section shows the assignee as 'Unassigned' and the reporter as 'admin'.

JIRA Dashboards ▾ Projects ▾ Issues ▾ **Create** Search 🔍 🔊 ? ⚙️ 👤

Allure Examples / AE-5
Добавление заметки

Resolution: Unresolved
Labels: favorites regress

Details

Type: Task
Status: **TO DO** (View Workflow)
Priority: ↑ Medium
Resolution: Unresolved
Labels: favorites regress

People

Assignee: Unassigned
Assign to me

Reporter: admin

Votes:

Создаем JiraLabelsImportAction

```
public class JiraLabelsImportAction extends AnAction {  
  
    @Override  
    public void actionPerformed(AnActionEvent event) {  
  
        PsiElement element = event.getData(PSI_ELEMENT)  
        if (element instanceof PsiClass) {  
            addTagsToClassMethods((PsiClass)element)  
        }  
  
    }  
  
}
```

Фильтруем методы с ключом

```
public class JiraLabelsImportAction extends AnAction {  
    void addTagsToClassMethods(PsiClass testClass) {  
        Arrays.stream(testClass.getMethods())  
            .filter(m -> m.hasAnnotation("org.TmsLink"))  
            .forEach(this::addTmsLinkToMethod)  
    }  
}
```

Зачитываем ключ Issue в Jira

```
public class JiraLabelsImportAction extends AnAction {
    void addTagsToMethod(PsiMethod testMethod) {
        PsiAnnotation tmsLink =
            testMethod.getAnnotation("io..TmsLink");
        String key = getTextValue(tmsLink);
        List<String> labels = jiraClient.getLables(key);
        String tagsText = getTagsText(labels);
        PsiAnnotation tags = create(tagsText, testMethod);
        testMethod.getModifiersList()
            .addAfter(tags, tmsLink);
    }
}
```

Получили ключи тестов

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @TmsLink("АЕ-5")  
    @DisplayName("Добавление в избранное после создания заметки")  
    public void shouldAddToFavoritesAfterNodeTest() { ... }  
  
    @Test  
    @TmsLink("АЕ-4")  
    @DisplayName("Удаление из избранного после удаления заметки")  
    public void shouldDeleteToFavoritesAfterNodeTest() { ... }  
  
}
```

Забираем лейбочки из Jira

```
public class JiraLabelsImportAction extends AnAction {  
    void addTagsToMethod(PsiMethod testMethod) {  
        PsiAnnotation tmsLink =  
            testMethod.getAnnotation("io..TmsLink");  
        String key = getTextValue(tmsLink);  
        List<String> labels = jiraClient.getLables(key);  
        String tagsText = getTagsText(labels);  
        PsiAnnotation tags = create(tagsText, testMethod);  
        testMethod.getModifiersList()  
            .addAfter(tags, tmsLink);  
    }  
}
```

Labels для разметки тестов

JIRA Dashboards ▾ Projects ▾ Issues ▾ **Create** Search 🔍 🔊 ? ⚙️ 👤

Allure Examples / AE-5
Добавление заметки

Resolution: Unresolved
Labels: favorites regress

Edit Comment Add

Details

Type: Task
Status: **TO DO** (View Workflow)
Priority: ↑ Medium
Resolution: Unresolved
Labels: favorites regress

People

Assignee: Unassigned
Assign to me

Reporter: admin

Votes:

Description

Создаем строку с аннотацией

```
public class JiraLabelsImportAction extends AnAction {  
    void addTagsToMethod(PsiMethod testMethod) {  
        PsiAnnotation tmsLink =  
            testMethod.getAnnotation("io..TmsLink");  
        String key = getTextValue(tmsLink);  
        List<String> labels = jiraClient.getLables(key);  
        String tagsText = getTagsText(labels);  
        PsiAnnotation tags = create(tagsText, testMethod);  
        testMethod.getModifiersList()  
            .addAfter(tags, tmsLink);  
    }  
}
```

АННОТАЦИЯ В ВИДЕ СТРОКИ

```
public String getInnerTagsText(List<String> labels) {  
    return features.stream()  
        .map(f -> "@org..Tag(\"" + f + "\")")  
        .collect(Collectors.joining(","))  
}
```

```
public String getTagsText(List<String> labels) {  
    String inner = getInnerTagsText(labels);  
    return "@org..Tag({" + inner + "})";  
}
```

```
@org..Tags({@org..Tag("regress"), ...})
```

Создаем аннотацию из строки

```
public class JiraLabelsImportAction extends AnAction {  
    void addTagsToMethod(PsiMethod testMethod) {  
        PsiAnnotation tmsLink =  
            testMethod.getAnnotation("io..TmsLink");  
        String key = getTextValue(tmsLink);  
        List<String> labels = jiraClient.getLables(key);  
        String tagsText = getTagsText(labels);  
        PsiAnnotation tags = create(tagsText, testMethod);  
        testMethod.getModifiersList()  
            .addAfter(tags, tmsLink);  
    }  
}
```

@Tags после @TmsLink

```
public class JiraLabelsImportAction extends AnAction {  
    void addTagsToMethod(PsiMethod testMethod) {  
        PsiAnnotation tmsLink =  
            testMethod.getAnnotation("io..TmsLink");  
        String key = getTextValue(tmsLink);  
        List<String> labels = jiraClient.getLables(key);  
        String tagsText = getTagsText(labels);  
        PsiAnnotation tags = create(tagsText, testMethod);  
        testMethod.getModifiersList()  
            .addAfter(tags, tmsLink);  
    }  
}
```

Добавили теги в тест

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @TmsLink("AE-5")  
    @org..Tags({@org..Tag("regress")})  
    @DisplayName("Добавление в избранное после создания заметки")  
    public void shouldAddToFavoritesAfterNodeTest() { ... }  
  
}
```

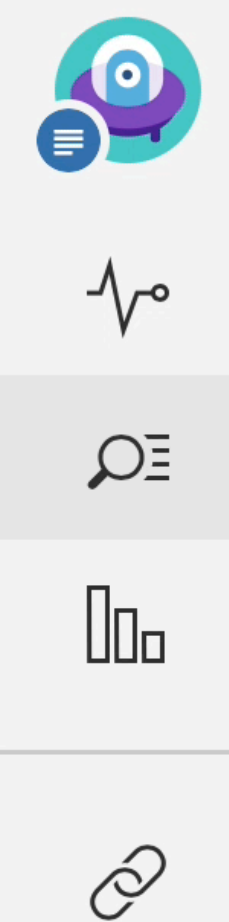
Оптимизируем импорты

```
public class JiraLabelsImportAction extends AnAction {  
    void addTagsToMethod(PsiMethod testMethod) {  
        addImport(testMethod.getContainingFile(), "org..Tag");  
        addImport(testMethod.getContainingFile(), "org..Tags");  
  
        // код из предыдущего слайда с write операцией  
  
        optimizeImports(testMethod.getContainingFile());  
    }  
}
```

Оптимизировали импорты

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @TmsLink("АЕ-5")  
    @Tags({@Tag("regress")})  
    @DisplayName("Добавление в избранное после создания заметки")  
    public void shouldAddToFavoritesAfterNoteTest() { ... }  
  
}
```





Open issues [Switch filter ▾](#)

[View all issues and filters](#)

Order by Priority ▾

AE-5

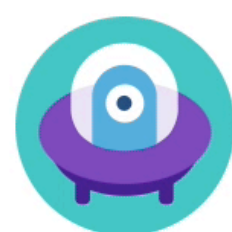
Добавление в избранное после создания за...

AE-4

[Удаление из избранного после удаления за...](#)

AE-6

Авторизация пользователя



Allure Examples / AE-4

2 of 3 ▲ ▼ ↗

Удаление из избранного после удаления заметки

Details

Type: Task

Status: **TO DO** ([View Workflow](#))

Priority: ↑ Medium

Resolution: Unresolved

Labels: [favorites](#)

People

Assignee:

Unassigned
[Assign to me](#)

Reporter:

admin

Votes:

0

Watchers:

1 [Stop watching this issue](#)

Description

Click to add description

Attachments

Drop files to attach, or [browse](#).

Какие еще есть идеи?


Импорт дескрипшена

Импорт Feature/Story

Импорт в проект

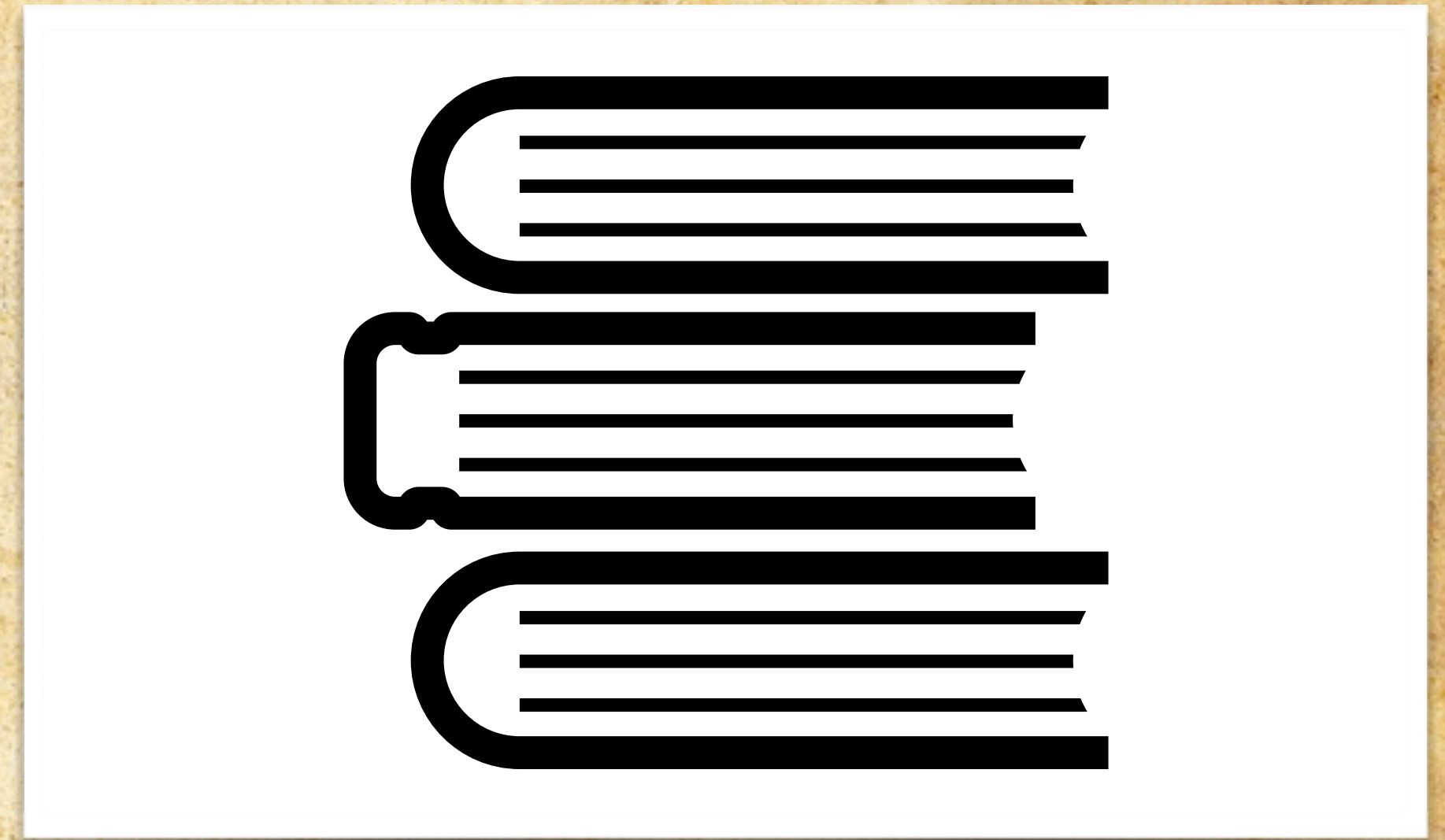
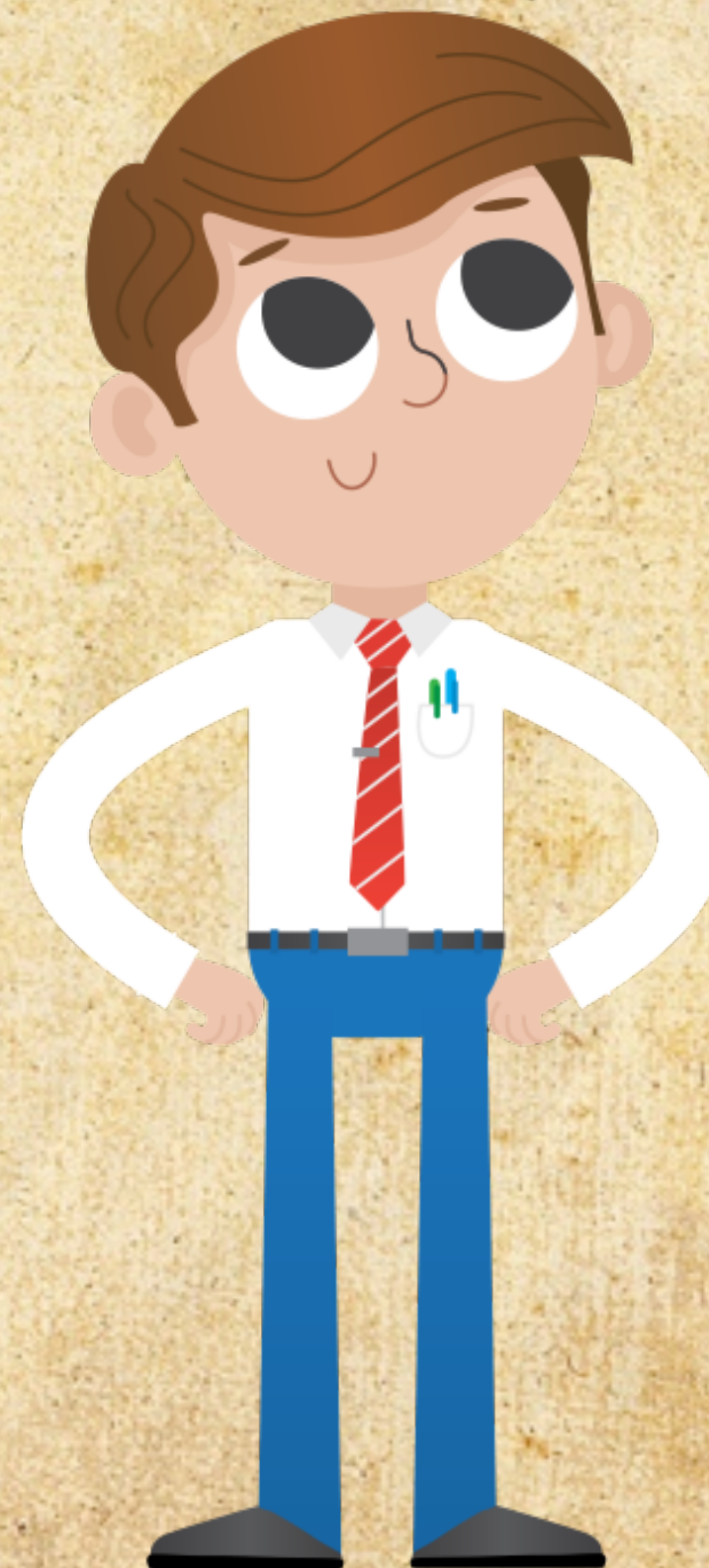
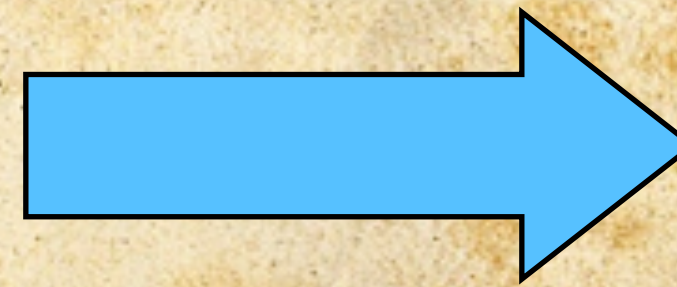
Экспорт из проекта

Миграция проекта

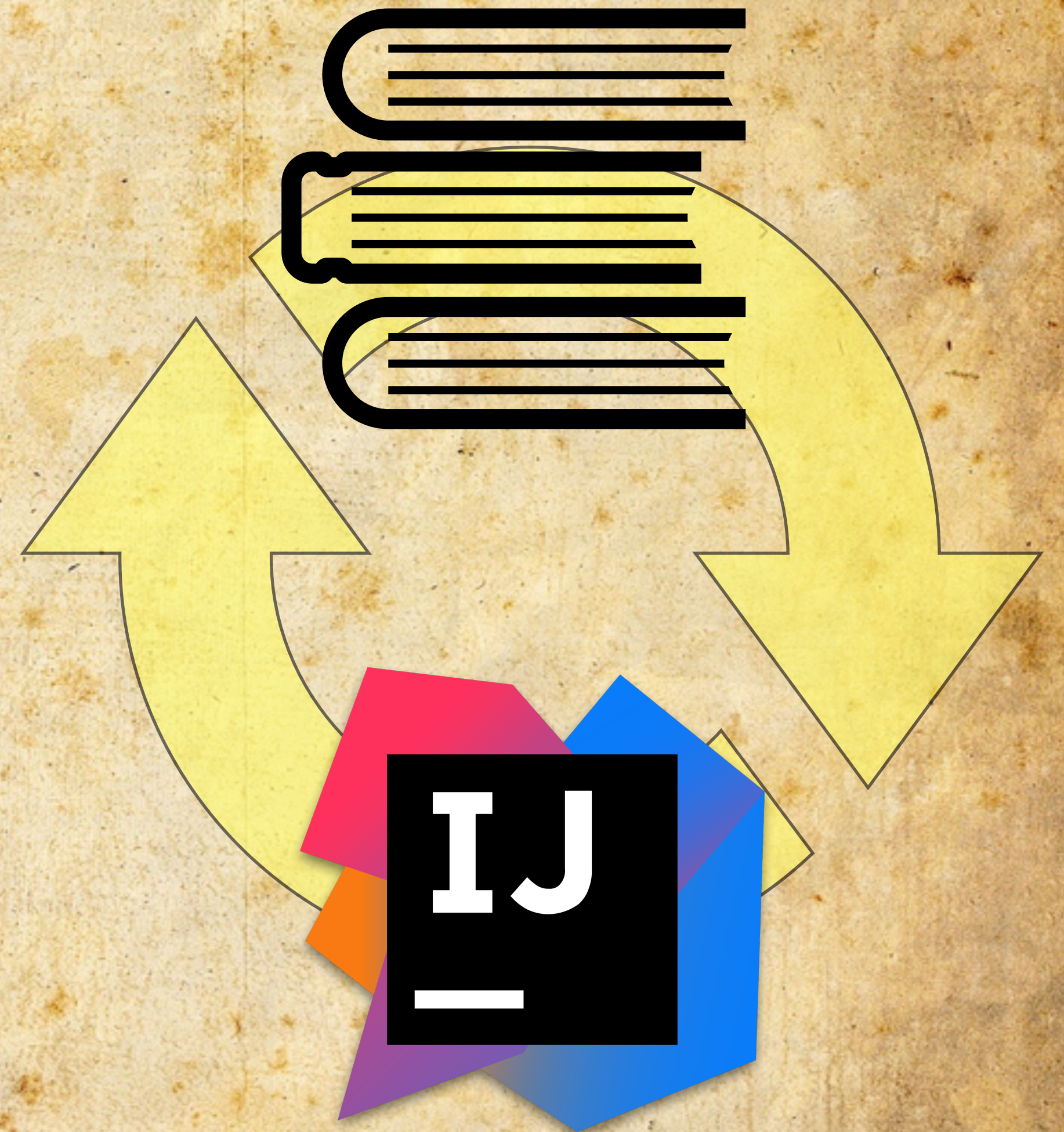
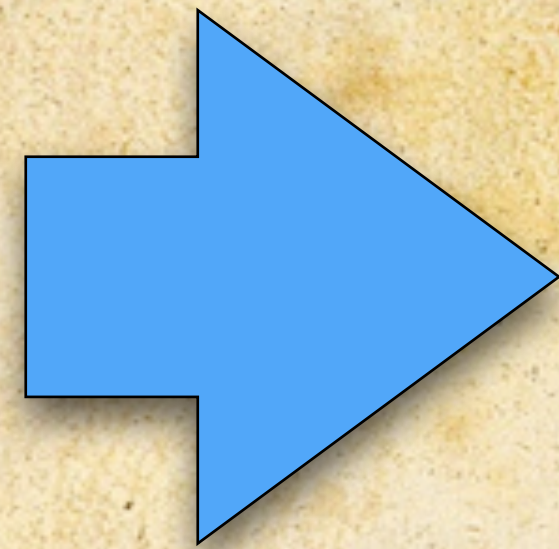
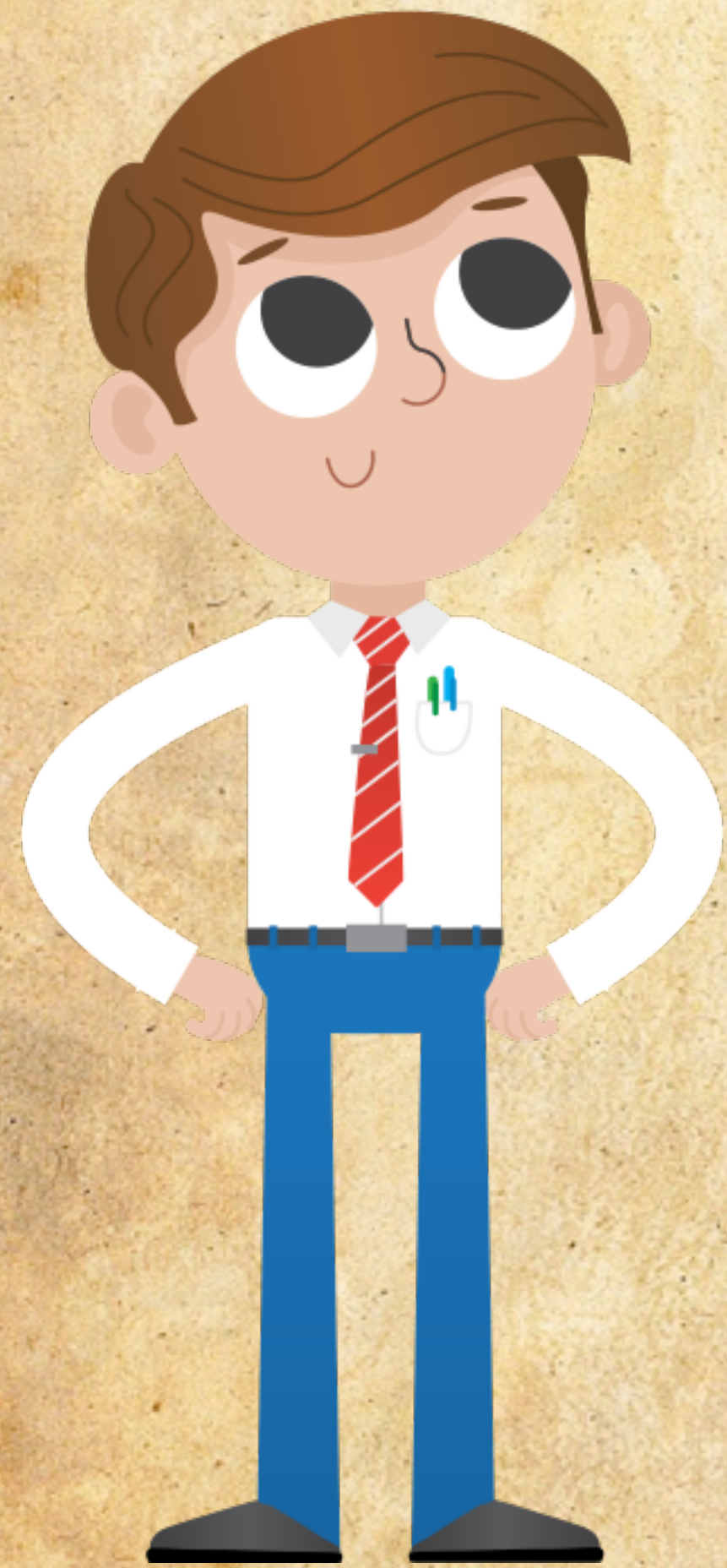



Какую проблему решаем?

А что в тестах проверяется?



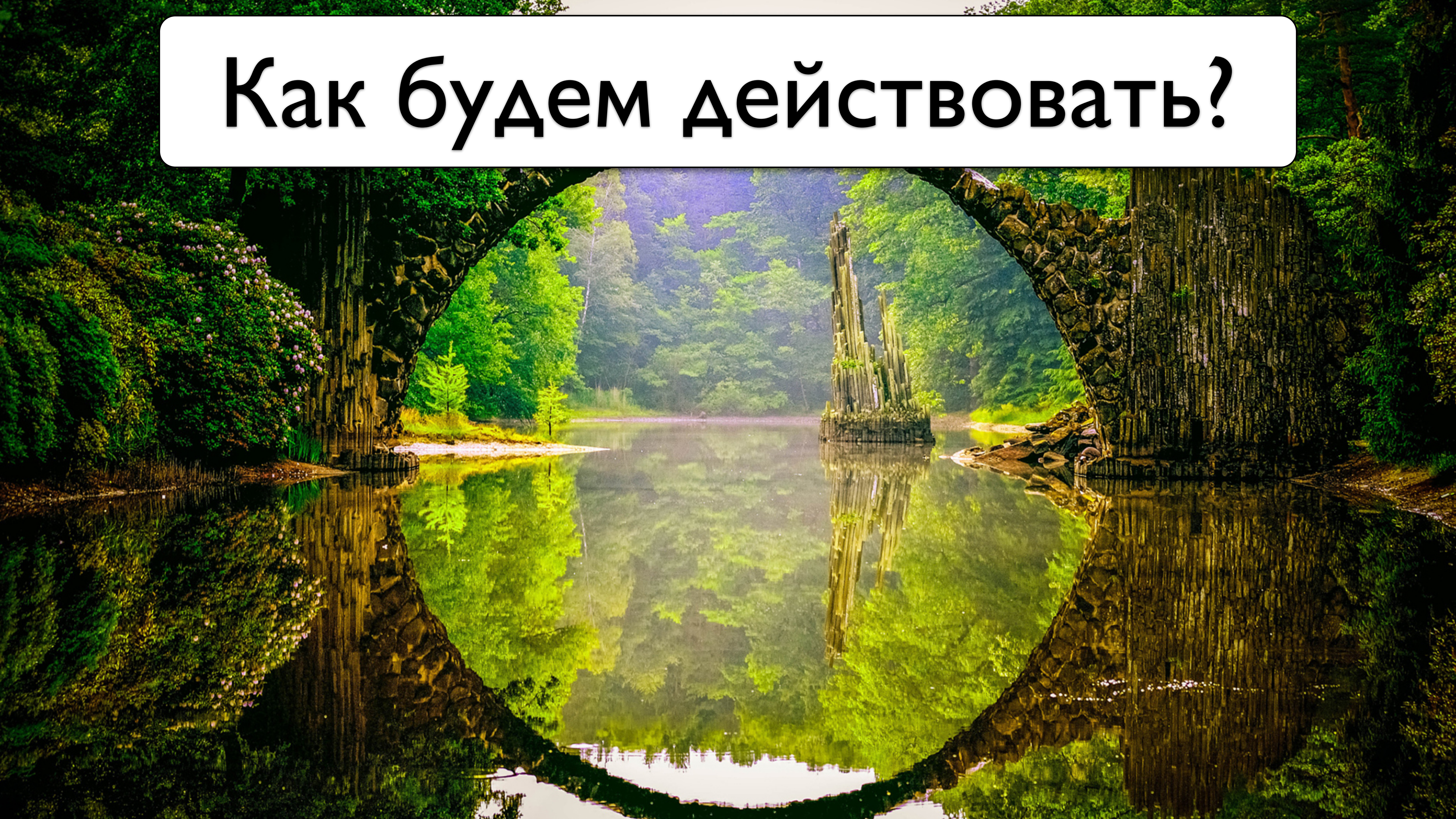
Задача на автоматизацию



A 3D maze with white walls and a red path leading from the top left towards the bottom right. A red arrow points out of the maze at the bottom right. A white rounded rectangle with a black border is overlaid in the center, containing the text.

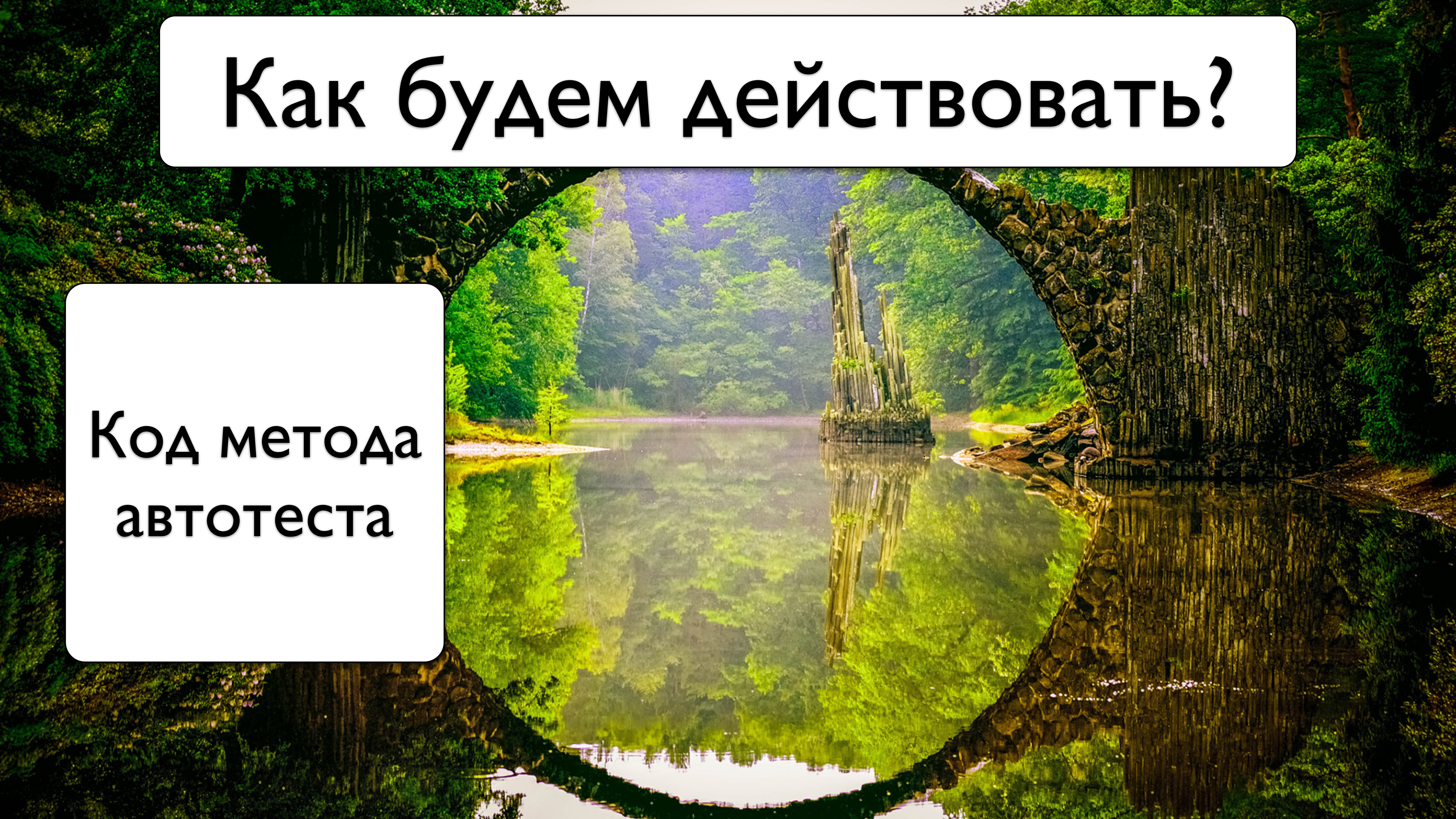
Способ решения проблемы

Как будем действовать?



Как будем действовать?

Код метода
автотеста



Мета информация

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @TmsLink("AE-5")  
    @Features({@Feature("Favorites")})  
    @Stories({@Story("Add favorites after note")})  
    @DisplayName("Добавление в избранное после создания заметки")  
    public void shouldAddToFavoritesAfterNodeTest() {  
        steps.openMainPage();  
        ...  
    }  
}
```

Сценарий теста

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @TmsLink("АЕ-5")  
    @Features({@Feature("Favorites")})  
    @Stories({@Story("Add favorites after note")})  
    @DisplayName("Добавление в избранное после создания заметки")  
    public void shouldAddToFavoritesAfterNodeTest() {  
        steps.openMainPage();  
        ...  
    }  
}
```

Как будем действовать?

Код метода
автотеста

Соберем
все
данные

Информация для экспорта

```
public class TestCase implements Serializable {  
  
    private String id;  
    private String name;  
    private List<String> features;  
    private List<String> stories;  
    private List<String> steps;  
  
    ...  
}
```

Как будем действовать?

Код метода
автотеста

Соберем
все
данные

Сохраним
в удобном
формате

HTML-шаблон на Freemarker

<#FREEMARKER>

[Home](#) | [Manual](#) | [Java API](#)



[WHAT IS APACHE FREEMARKER™?](#)

[DOWNLOAD / MAVEN](#)

[DOCUMENTATION](#)

- [Manual](#)
- [Java API](#)
- [Manual Chinese translation](#)

[TOOLING](#)

- [Editor and IDE plugins](#)
- [Online template tester](#)

What is Apache FreeMarker™?

Apache FreeMarker™ is a *template engine*: a Java library to generate text output (HTML web pages, e-mails, configuration files, source code, etc.) based on templates and changing data. Templates are written in the FreeMarker Template Language (FTL), which is a simple, specialized language (not a full-blown programming language like PHP). Usually, a general-purpose programming language (like Java) is used to prepare the data (issue database queries, do business calculations). Then, Apache FreeMarker displays that prepared data using templates. In the template you are focusing on how to present the data, and outside the template you are focusing on what data to



Давайте запилим плагин!

Создаем TestCaseExportAction

```
public class TestCaseExportAction extends AnAction {  
    @Override  
    public void actionPerformed(AnActionEvent event) {  
        PsiElement element = event.getData(PSI_ELEMENT);  
        List<TestCase> testcases =  
            Arrays.stream(((PsiClass) element).getMethods())  
                .filter(m -> m.hasAnnotation("org..Test"))  
                .map(this::getTestCaseFromMethod)  
                .collect(Collectors.toList());  
        writeTestCases(event.getProject(), testcases);  
    }  
}
```

Берем все методы тестов

```
public class TestCaseExportAction extends AnAction {
    @Override
    public void actionPerformed(AnActionEvent event) {
        PsiElement element = event.getData(PSI_ELEMENT);
        List<TestCase> testcases =
            Arrays.stream(((PsiClass) element).getMethods())
                .filter(m -> m.hasAnnotation("org.Test"))
                .map(this::getTestCaseFromMethod)
                .collect(Collectors.toList());
        writeTestCases(event.getProject(), testcases);
    }
}
```

Собираем информацию

```
public class TestCaseExportAction extends AnAction {
    @Override
    public void actionPerformed(AnActionEvent event) {
        PsiElement element = event.getData(PSI_ELEMENT);
        List<TestCase> testcases =
            Arrays.stream(((PsiClass) element).getMethods())
                .filter(m -> m.hasAnnotation("org..Test"))
                .map(this::getTestCaseFromMethod)
                .collect(Collectors.toList());
        writeTestCases(event.getProject(), testcases);
    }
}
```

Сохраняем результат

```
public class TestCaseExportAction extends AnAction {
    @Override
    public void actionPerformed(AnActionEvent event) {
        PsiElement element = event.getData(PSI_ELEMENT);
        List<TestCase> testcases =
            Arrays.stream(((PsiClass) element).getMethods())
                .filter(m -> m.hasAnnotation("org..Test"))
                .map(this::getTestCaseFromMethod)
                .collect(Collectors.toList());
        writeTestCases(event.getProject(), testcases);
    }
}
```

Как собираем данные метода?

```
public class TestCaseExportAction extends AnAction {  
  
    public TestCase getTestCaseFromMethod(PsiMethod method) {  
        TestCase testCase = new TestCase();  
        testCase.setId(getTmsLinkText(method));  
        testCase.setName(getDisplayNameText(method));  
        testCase.setFeature(getFeaturesText(method));  
        testCase.setStories(getStoriesText(method));  
        testCase.setSteps(getSteps(method));  
        return testCase;  
    }  
}
```

Уже делали в первом плагине

```
public class TestCaseExportAction extends AnAction {  
  
    public TestCase getTestCaseFromMethod(PsiMethod method) {  
        TestCase testCase = new TestCase();  
        testCase.setId(getTmsLinkText(method));  
        testCase.setName(getDisplayNameText(method));  
        testCase.setFeature(getFeaturesText(method));  
        testCase.setStories(getStoriesText(method));  
        testCase.setSteps(getSteps(method));  
        return testCase;  
    }  
}
```

Уже делали в первом плагине

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @TmsLink("АЕ-5")  
    @Features({@Feature("Favorites")})  
    @Stories({@Story("Add favorites after note")})  
    @DisplayName("Добавление в избранное после создания заметки")  
    public void shouldAddToFavoritesAfterNodeTest() {  
        steps.openMainPage();  
        ...  
    }  
}
```

То же самое, ТОЛЬКО МАССИВЫ

```
public class TestCaseExportAction extends AnAction {  
  
    public TestCase getTestCaseFromMethod(PsiMethod method) {  
        TestCase testCase = new TestCase();  
        testCase.setId(getTmsLinkText(method));  
        testCase.setName(getDisplayNameText(method));  
        testCase.setFeature(getFeaturesText(method));  
        testCase.setStories(getStoriesText(method));  
        testCase.setSteps(getSteps(method));  
        return testCase;  
    }  
}
```


То же самое, ТОЛЬКО МАССИВЫ

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @TmsLink("АЕ-5")  
    @Features({@Feature("Favorites")})  
    @Stories({@Story("Add favorites after note")})  
    @DisplayName("Добавление в избранное после создания заметки")  
    public void shouldAddToFavoritesAfterNodeTest() {  
        steps.openMainPage();  
        ...  
    }  
}
```

А ВОТ ЭТО НЕ ТРИВИАЛЬНО

```
public class TestCaseExportAction extends AnAction {  
  
    public TestCase getTestCaseFromMethod(PsiMethod method) {  
        TestCase testCase = new TestCase();  
        testCase.setId(getTmsLinkText(method));  
        testCase.setName(getDisplayNameText(method));  
        testCase.setFeature(getFeaturesText(method));  
        testCase.setStories(getStoriesText(method));  
        testCase.setSteps(getSteps(method));  
        return testCase;  
    }  
}
```

Сценарий теста

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    public void shouldAddToFavoritesAfterNoteTest() {  
        steps.openMainPage();  
        steps.openAutoCardPage("Volvo XC90");  
        steps.addNotesToAutoCard("В хорошем состоянии");  
        steps.openFavoritesPage("В хорошем состоянии");  
        steps.checkFavoritesListContains("Volvo XC90");  
    }  
}
```

Добываем список Шагов

```
public class ScenarioExportAction extends AnAction {  
    public List<String> getSteps(PsiMethod testMethod) {  
        Arrays.stream(method.getBody().getStatements())  
            .filter(PsiMethodCallExpression.class::isInstance)  
            .map(PsiMethodClassExpression.class::cast)  
            .map(PsiMethodClassExpression::resolveMethod())  
            .filter(m -> m.hasAnnotation("...Step"))  
            .map(m -> m.getAnnotation("...Step"))  
            .map(a -> a.getDeclaredAttribute("value"))  
            .map(PsiAnnotationMemberValue::getText)  
            .forEach(Collectors.toList())  
    }  
}
```

Не метод, а оператор вызова

```
public class AddFavoritesAfterNoteTest {  
    @Test  
    public void shouldAddToFavoritesFromNodeTest() {  
        steps.openMainPage();  
        steps.openAutoCardPage("Volvo XC90");  
        steps.addNotesToAutoCard("В хорошем состоянии");  
        steps.openFavoritesPage();  
        steps.checkFavoritesListContains("Volvo XC90");  
    }  
}
```

PsiMethodCallExpression

Добываем список методов

```
public class ScenarioExportAction extends AnAction {  
    public List<String> getSteps(PsiMethod testMethod) {  
        Arrays.stream(method.getBody().getStatements())  
            .filter(PsiMethodCallExpression.class::isInstance)  
            .map(PsiMethodClassExpression.class::cast)  
            .map(PsiMethodClassExpression::resolveMethod)  
            .filter(m -> m.hasAnnotation("...Step"))  
            .map(m -> m.getAnnotation("...Step"))  
            .map(a -> a.getDeclaredAttribute("value"))  
            .map(PsiAnnotationMemberValue::getText)  
            .forEach(Collectors.toList())  
    }  
}
```

Получили реальные методы

```
public class BasicSteps {
```

```
@Step("Открываем главную страницу")
```

```
public void openMainPage() { ... }
```

```
@Step("Открываем страницу машины марки {mark}")
```

```
public void openAutoCardPage(String mark) { ... }
```

```
@Step("Добавляем заметку {text} к машине")
```

```
public void addNotesToAutoCard(String text) { ... }
```

```
private void utilityMethod { ... }
```

```
}
```

конкретный PsiMethod

Добываем список шагов

```
public class ScenarioExportAction extends AnAction {
    public List<String> getSteps(PsiMethod testMethod) {
        Arrays.stream(method.getBody().getStatements())
            .filter(PsiMethodCallExpression.class::isInstance)
            .map(PsiMethodClassExpression.class::cast)
            .map(PsiMethodClassExpression::resolveMethod())
            .filter(m -> m.hasAnnotation("...Step"))
            .map(m -> m.getAnnotation("...Step"))
            .map(a -> a.getDeclaredAttribute("value"))
            .map(PsiAnnotationMemberValue::getText)
            .forEach(Collectors.toList())
    }
}
```


Получили текст аннотации

```
public class BasicSteps {  
  
    @Step( "Открываем главную страницу" )  
    public void openMainPage() { ... }  
  
    @Step( "Открываем страницу машины марки {mark}" )  
    public void openAutoCardPage(String mark) { ... }  
  
    @Step( "Добавляем заметку {text} к машине" )  
    public void addNotesToAutoCard(String text) { ... }  
  
    private void utilityMethod { ... }  
  
}
```

Добываем список Шагов

```
public class ScenarioExportAction extends AnAction {  
    public List<String> getSteps(PsiMethod testMethod) {  
        Arrays.stream(method.getBody().getStatements())  
            .filter(PsiMethodCallExpression.class::isInstance)  
            .map(PsiMethodClassExpression.class::cast)  
            .map(PsiMethodClassExpression::resolveMethod())  
            .filter(m -> m.hasAnnotation("...Step"))  
            .map(m -> m.getAnnotation("...Step"))  
            .map(a -> a.getDeclaredAttribute("value"))  
            .map(PsiAnnotationMemberValue::getText)  
            .forEach(Collectors.toList())  
    }  
}
```

Как сохранить результат?

```
public class TestCaseExportAction extends AnAction {  
  
    public void writeTestCases(Project project,  
                               List<TestCase> testCases) {  
        String content = FreemarketUtils  
            .processTemplate("testcases.ftl", testCases);  
        Path projectDir = Paths.get(project.getBasePath())  
        Path indexFile = projectDir.resolve("index.html");  
        Files.write(indexFile, content.getBytes());  
    }  
}
```

Шаблон "testcases.ftl"

```
<#list testcases as testcase>
  <h2>${testcase.id} - ${testcase.name}</h2>
  <div class="meta">
    <div class="features">${testcase.features}</div>
  </div>
  <ul class="list-group">
    <#list testcase.steps as step>
      <li class="list-group-item">${step}</li>
    </#list>
  </ul>
</#list>
```

Заполняем шаблон данными

```
public class TestCaseExportAction extends AnAction {  
  
    public void writeTestCases(Project project,  
                               List<TestCase> testCases) {  
        String content = FreemarketUtils  
            .processTemplate("testcases.ftl", testCases);  
        Path projectDir = Paths.get(project.getBasePath())  
        Path indexFile = projectDir.resolve("index.html");  
        Files.write(indexFile, content.getBytes());  
    }  
}
```

Сохраняем результат

```
public class TestCaseExportAction extends AnAction {  
  
    public void writeTestCases(Project project,  
                               List<TestCase> testCases) {  
        String content = FreemarketUtils  
            .processTemplate("testcases.ftl", testCases);  
        Path projectDir = Paths.get(project.getBasePath())  
        Path indexFile = projectDir.resolve("index.html");  
        Files.write(indexFile, content.getBytes());  
    }  
}
```



Project ~/Developer/eroshenkoam/

- testcase-export
 - .gradle
 - .idea
 - gradle
 - src
 - main
 - java
 - io.eroshenkoam.autotests
 - BasicSteps
 - test
 - java
 - io.eroshenkoam.autotests
 - AddFavoritesExportTest
 - build.gradle.kts
 - gradlew
 - gradlew.bat
 - index.html
 - settings.gradle.kts
 - testcase-export.iml

External Libraries

Scratches and Consoles

```
2
3 import io.qameta.allure.*;
4 import org.junit.jupiter.api.DisplayName;
5 import org.junit.jupiter.api.Test;
6
7 public class AddFavoritesExportTest {
8
9     private BasicSteps steps;
10
11     @Test
12     @TmsLink("AE-5")
13     @Features({@Feature("Favorites"), @Feature("Notes")})
14     @Stories({@Story("Add to favorites after adding note")})
15     @DisplayName("Добавление в избранное после создания заметки")
16     public void shouldAddToFavoritesAfterNodeTest() {
17         steps.openMainPage();
18         steps.openAutoCardPage( mark: "Volvo XC90");
19         steps.addNotesToAutoCard( text: "В хорошем состоянии");
20         steps.openFavoritesPage();
21         steps.checkFavoritesListContains( mark: "Volvo XC90");
22     }
23
24 }
25
```


Какие еще есть идеи?


Экспорт в другие системы

Поиск "плохого" кода

Импорт в проект

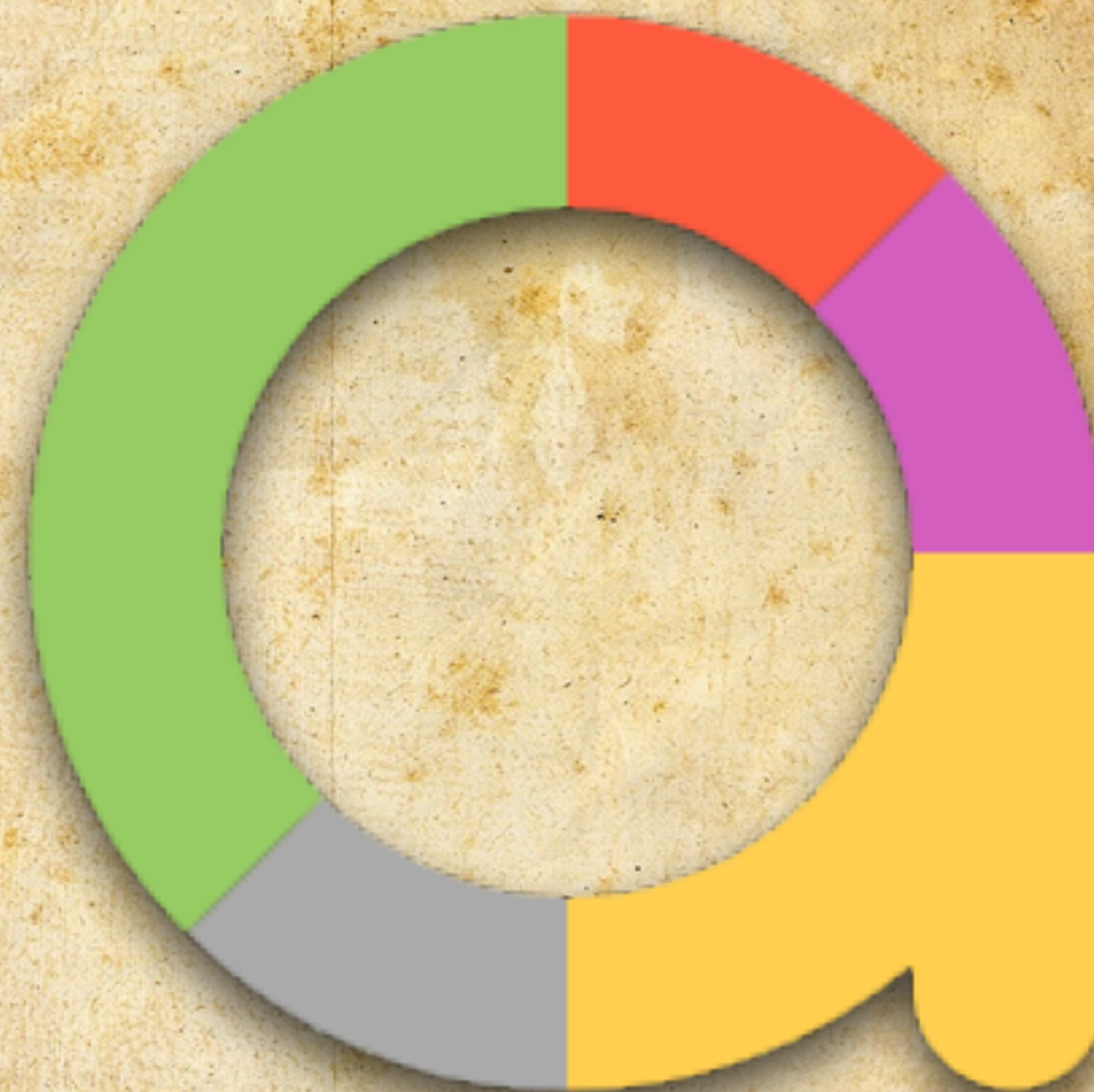
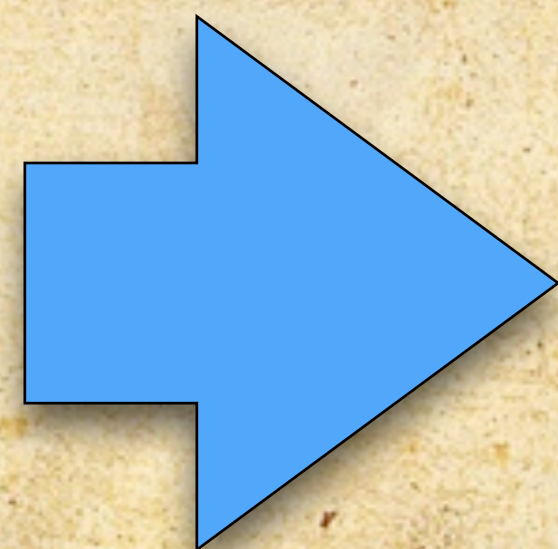
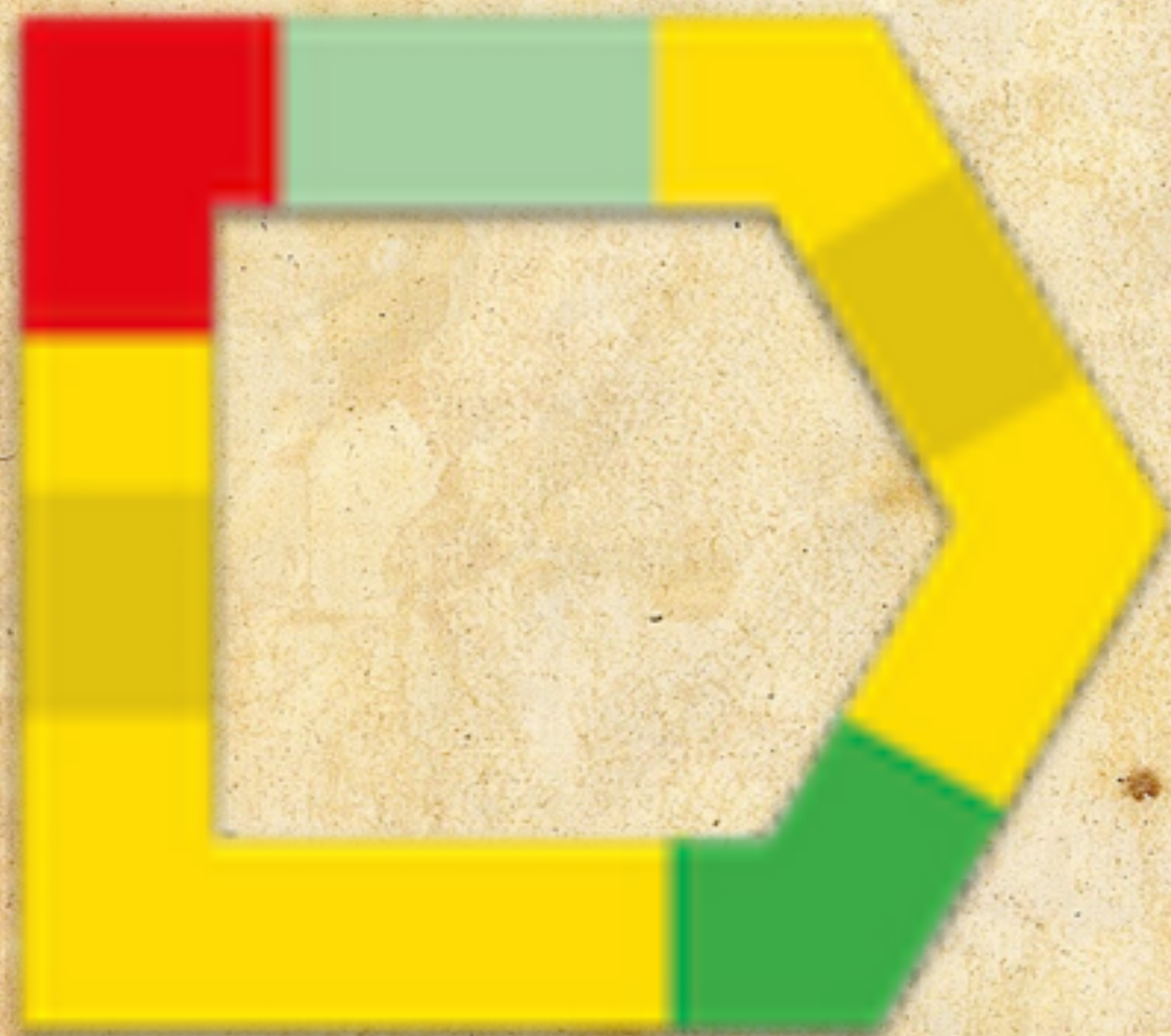
Экспорт из проекта

Миграция проекта



Какую проблему решаем?

Переезд на второй Allure



Аннотация Name

```
@Test
@Name("My simple test here")
public void simpleTest() {
    //test body
}
```

```
@Test
@DisplayName("My simple test here")
public void simpleTest() {
    //test body here
}
```

Аннотация Feature

```
@Test
@Feature( ["First", "Second" ])
public void simpleTest() { .. }
```

```
@Test
@Features( {
    @Feature( "First" ),
    @Feature( "Second" )
} )
public void simpleTest() { ... }
```

Аннотация Story

```
@Test
@Story( ["First", "Second" ])
public void simpleTest() { .. }
```

```
@Test
@Stories( {
    @Story( "First" ),
    @Story( "Second" )
} )
public void simpleTest() { ... }
```

Аннотация Issue

```
@Test
@Issue( ["AF-1", "AF-2"] )
public void simpleTest() { .. }
```

```
@Test
@Issues( {
    @Issue( "AF-1" ),
    @Issue( "AF-2" )
} )
public void simpleTest() { ... }
```


Аннотация TmsLink

```
@Test
@TestCaseId( [ "AE-4", "AE-5" ] )
public void simpleTest() { .. }
```

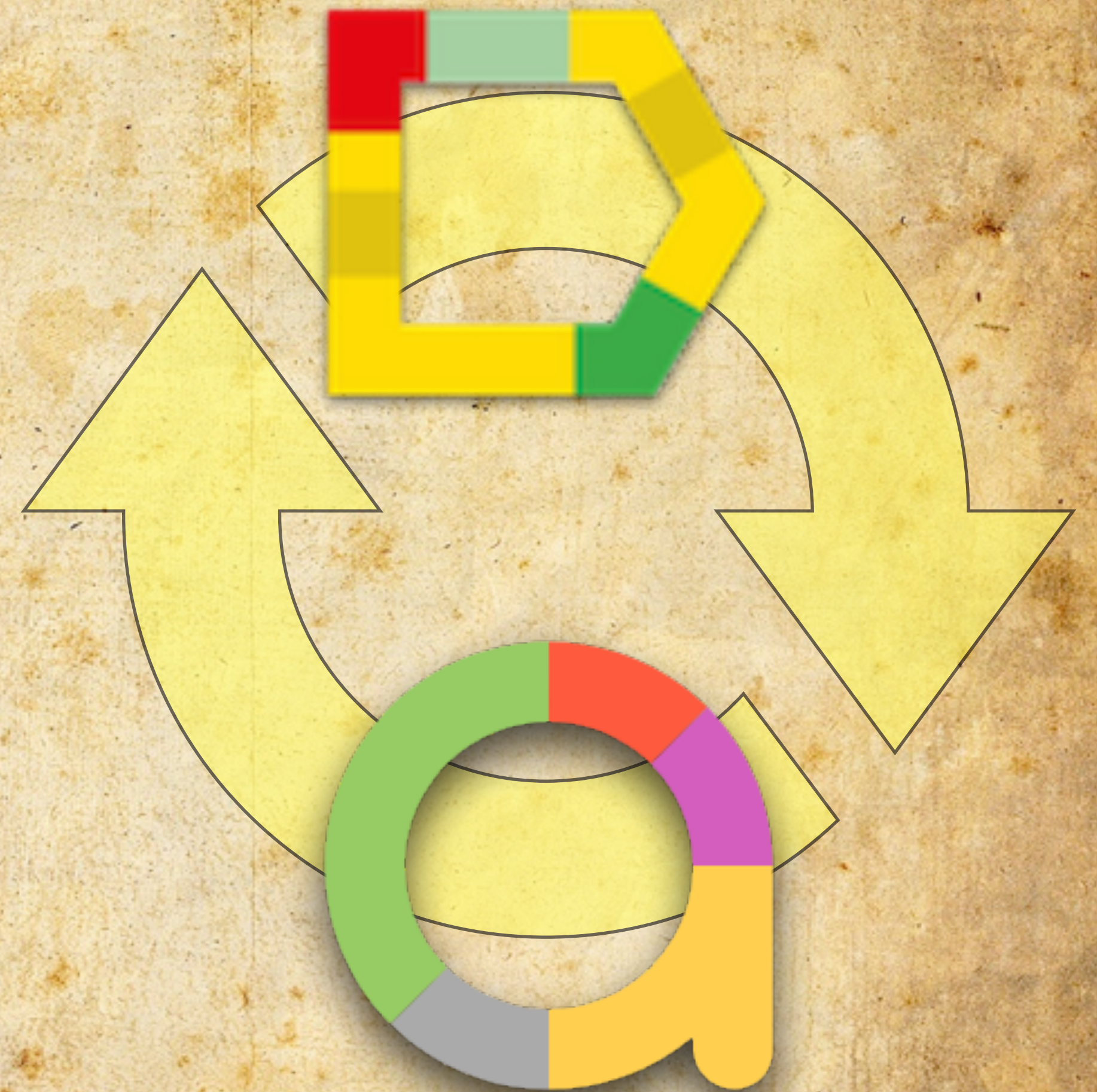
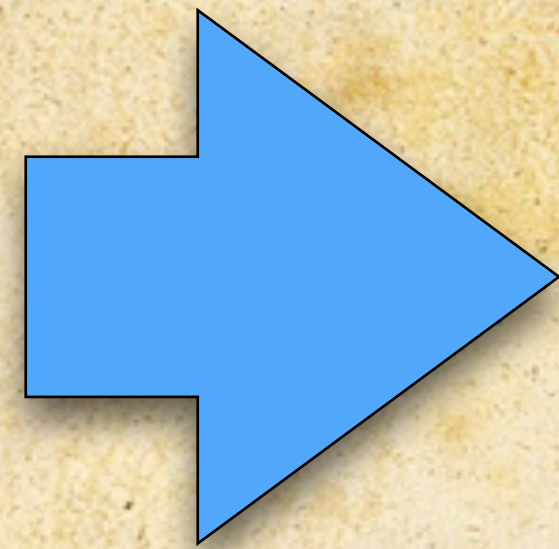
```
@Test
@TestLinks( {
    @TmsLink( "AE-4" ),
    @TmsLink( "AE-5" )
} )
public void simpleTest() { ... }
```


Ультимейт - аннотация Step

```
@Step("Login as user {0}")  
public void loginAsUser(String username) {  
    //step body here  
}
```

```
@Step("Login as user {username}")  
public void simpleTest(String username) {  
    //test body here  
}
```

Переезд на второй Allure



A 3D maze with white walls and a red path leading from the top left towards the bottom right. A red arrow points out of the maze at the bottom right. A white rounded rectangle with a black border is overlaid on the maze, containing the text.

Способ решения проблемы

Будем мигрировать по-одному

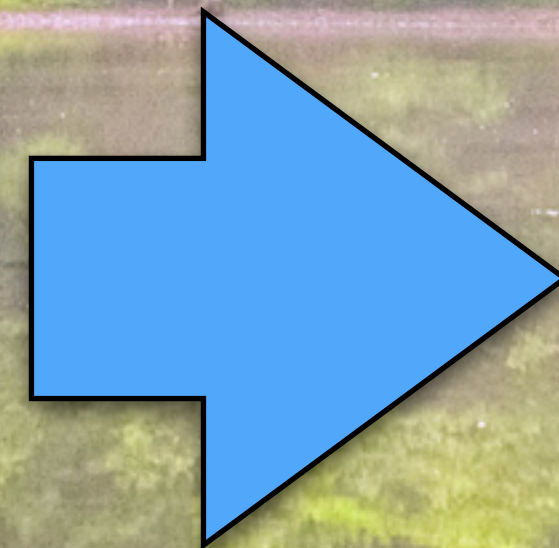
@Name

@Feature

@Story

....

@Step



@DisplayName

@Features

@Stories

....

@Step

Создаем AllureMigrationAction

```
public void actionPerformed(AnActionEvent event) {  
    PsiElement element = event.getData(PSI_ELEMENT);  
    Arrays.stream(((PsiClass) element).getMethods())  
        .filter(m -> m.hasAnnotation("org..Test"))  
        .forEach(this::migrateTestAnnotations)  
    }  
    Arrays.stream(((PsiClass) element).getMethods())  
        .filter(m -> m.hasAnnotation("org..Step"))  
        .forEach(this::migrateStepAnnotation)  
    }  
}
```

Мигрируем тестовые методы

```
public void actionPerformed(ActionEvent event) {  
    PsiElement element = event.getData(PSI_ELEMENT);  
    Arrays.stream(((PsiClass) element).getMethods())  
        .filter(m -> m.hasAnnotation("org..Test"))  
        .forEach(this::migrateTestAnnotations)  
    }  
    Arrays.stream(((PsiClass) element).getMethods())  
        .filter(m -> m.hasAnnotation("org..Step"))  
        .forEach(this::migrateStepAnnotation)  
    }  
}
```

Мигрируем по одной

```
public void migrateTestAnnotations(PsiMethod method) {  
  
    migrateDisplayName(method);  
    migrateDescription(method);  
    migrateFeatures(method);  
    migrateStories(method);  
    migrateTmsIds(method);  
    migrateIssues(method);  
    migrateLins(method);  
  
}
```


Мигрируем на @Features

```
public void migrateFeatures(PsiMethod method) {  
  
    PsiAnnotation oldFeature =  
        method.getAnnotation("ru..Feature");  
    List<String> features = getValues(oldFeature, "value");  
    String featuresText = getFeaturesText(features);  
    PsiAnnotation features = createAnnotation(featuresText);  
    method.getModifiersList()  
        .addAfter(features, oldFeature);  
    oldFeature.delete();  
}
```

Тест с аннотацией @Feature

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @Feature({"Favorites", "Notes"})  
    public void shouldAddToFavoritesAfterNodeTest() {  
        ...  
    }  
  
}
```

Ищем аннотацию @Feature

```
public void migrateFeatures(PsiMethod method) {  
  
    PsiAnnotation oldFeature =  
        method.getAnnotation("ru.Feature");  
    List<String> features = getValues(oldFeature, "value");  
    String featuresText = getFeaturesText(features);  
    PsiAnnotation features = createAnnotation(featuresText);  
    method.getModifiersList()  
        .addAfter(features, oldFeature);  
    oldFeature.delete();  
}
```

Взяли значение @Feature

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @Feature({ "Favorites", "Notes" })  
    public void shouldAddToFavoritesAfterNodeTest() {  
        ...  
    }  
  
}
```

Создаем новую @Features

```
public void migrateFeatures(PsiMethod method) {  
  
    PsiAnnotation oldFeature =  
        method.getAnnotation("ru..Feature");  
    List<String> features = getValues(oldFeature, "value");  
    String featuresText = getFeaturesText(features);  
    PsiAnnotation features = createAnnotation(featuresText);  
    method.getModifiersList()  
        .addAfter(features, oldFeature);  
    oldFeature.delete();  
}
```

АННОТАЦИЯ В ВИДЕ СТРОКИ

```
public String getInnerFeaturesText(List<String> features) {  
    return features.stream()  
        .map(f -> "@io..Feature(\"" + f + "\")")  
        .collect(Collectors.joining(","))  
}
```

```
public String getFeaturesText(List<String> features) {  
    String inner = getInnerFeaturesText(features);  
    return "@io..Features({" + inner + "})";  
}
```

```
@io..Features({@io..Feature("Favorites"), ...})
```

Создаем новую @Features

```
public void migrateFeatures(PsiMethod method) {  
  
    PsiAnnotation oldFeature =  
        method.getAnnotation("ru..Feature");  
    List<String> features = getValues(oldFeature, "value");  
    String featuresText = getFeaturesText(features);  
    PsiAnnotation features = createAnnotation(featuresText);  
    method.getModifiersList()  
        .addAfter(features, oldFeature);  
    oldFeature.delete();  
}
```

Добавляем ее к методу

```
public void migrateFeatures(PsiMethod method) {  
  
    PsiAnnotation oldFeature =  
        method.getAnnotation("ru..Feature");  
    List<String> features = getValues(oldFeature, "value");  
    String featuresText = getFeaturesText(features);  
    PsiAnnotation features = createAnnotation(featuresText);  
    method.getModifiersList()  
        .addAfter(features, oldFeature);  
    oldFeature.delete();  
}
```


Добавилась новая аннотация

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @Feature( {"Favorites", "Notes"} )  
    @Features( { @Feature( "Favorites" ), @Feature( "Notes" ) } )  
    public void shouldAddToFavoritesAfterNoteTest() {  
        ...  
    }  
  
}
```

Удаляем старую @Feature

```
public void migrateFeatures(PsiMethod method) {  
  
    PsiAnnotation oldFeature =  
        method.getAnnotation("ru..Feature");  
    List<String> features = getValues(oldFeature, "value");  
    String featuresText = getFeaturesText(features);  
    PsiAnnotation features = createAnnotation(featuresText);  
    method.getModifiersList()  
        .addAfter(features, oldFeature);  
    oldFeature.delete();  
}
```

Осталась ТОЛЬКО НОВАЯ

```
public class AddFavoritesAfterNoteTest {  
  
    @Test  
    @Features({@Feature("Favorites"), @Feature("Notes")})  
    public void shouldAddToFavoritesAfterNodeTest() {  
        ...  
    }  
  
}
```

Остальные по аналогии

@Name

@Feature

@Story

....

@Step

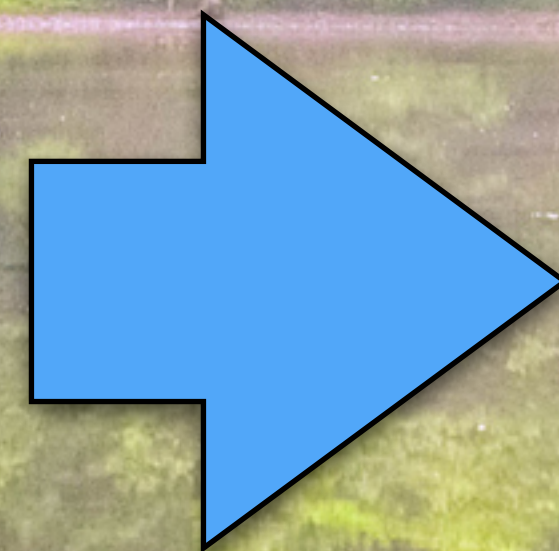
@DisplayName

@Features

@Stories

....

@Step



Мигрируем степы тестов

```
public void actionPerformed(ActionEvent event) {  
    PsiElement element = event.getData(PSI_ELEMENT);  
    Arrays.stream(((PsiClass) element).getMethods())  
        .filter(m -> m.hasAnnotation("org..Test"))  
        .forEach(this::migrateTestAnnotations)  
    }  
    Arrays.stream(((PsiClass) element).getMethods())  
        .filter(m -> m.hasAnnotation("org..Step"))  
        .forEach(this::migrateStepAnnotation)  
    }  
}
```

Мигрируем методы @Step

```
public void migrateStepAnnotation(PsiMethod method) {  
    PsiAnnotation oldStep =  
        method.getAnnotation("ru..Step");  
    String oldstepText = getValue(oldStep, "value");  
    String[] params = Arrays.stream(method.getParameters())  
        .map(JvmParameter::getName).toArray(String[]::new);  
    String newStepText = convert(oldStepText, params);  
    PsiAnnotation newStep = createStep(newStepText);  
    method.getModifiersList()  
        .addAfter(newStep, oldStep);  
    oldStep.delete();  
}
```

Метод с аннотацией @Step

```
public class BasicSteps {  
  
    @Step("Открываем страницу машины марки {0}")  
    public void openAutoCardPage(String mark) { ... }  
  
    @Step("Добавляем заметку {0} к машине")  
    public void addNotesToAutoCard(String text) { ... }  
  
}
```

Получаем аннотацию @Step

```
public void migrateStepAnnotation(PsiMethod method) {  
    PsiAnnotation oldStep =  
        method.getAnnotation("ru..Step");  
    String oldstepText = getValue(oldStep, "value");  
    String[] params = Arrays.stream(method.getParameters())  
        .map(JvmParameter::getName).toArray(String[]::new);  
    String newStepText = convert(oldStepText, params);  
    PsiAnnotation newStep = createStep(newStepText);  
    method.getModifiersList()  
        .addAfter(newStep, oldStep);  
    oldStep.delete();  
}
```


Получили текст аннотации

```
public class BasicSteps {  
  
    @Step( "Открываем страницу машины марки {0}" )  
    public void openAutoCardPage(String mark) { ... }  
  
    @Step( "Добавляем заметку {0} к машине" )  
    public void addNotesToAutoCard(String text) { ... }  
  
}
```

Получаем параметры метода

```
public void migrateStepAnnotation(PsiMethod method) {  
    PsiAnnotation oldStep =  
        method.getAnnotation("ru..Step");  
    String oldstepText = getValue(oldStep, "value");  
    String[] params = Arrays.stream(method.getParameters())  
        .map(JvmParameter::getName).toArray(String[]::new);  
    String newStepText = convert(oldStepText, params);  
    PsiAnnotation newStep = createStep(newStepText);  
    method.getModifiersList()  
        .addAfter(newStep, oldStep);  
    oldStep.delete();  
}
```

Получили параметры метода

```
public class BasicSteps {  
  
    @Step("Открываем страницу машины марки {0}")  
    public void openAutoCardPage(String mark) { ... }  
  
    @Step("Добавляем заметку {0} к машине")  
    public void addNotesToAutoCard(String text) { ... }  
  
}
```

Конвертируем текст степа

```
public void migrateStepAnnotation(PsiMethod method) {  
    PsiAnnotation oldStep =  
        method.getAnnotation("ru..Step");  
    String oldstepText = getValue(oldStep, "value");  
    String[] params = Arrays.stream(method.getParameters())  
        .map(JvmParameter::getName).toArray(String[]::new);  
    String newStepText = convert(oldStepText, params);  
    PsiAnnotation newStep = createStep(newStepText);  
    method.getModifiersList()  
        .addAfter(newStep, oldStep);  
    oldStep.delete();  
}
```

Добавляем НОВЫЙ @Step

```
public void migrateStepAnnotation(PsiMethod method) {  
    PsiAnnotation oldStep =  
        method.getAnnotation("ru..Step");  
    String oldstepText = getValue(oldStep, "value");  
    String[] params = Arrays.stream(method.getParameters())  
        .map(JvmParameter::getName).toArray(String[]::new);  
    String newStepText = convert(oldStepText, params);  
    PsiAnnotation newStep = createStep(newStepText);  
    method.getModifiersList()  
        .addAfter(newStep, oldStep);  
    oldStep.delete();  
}
```

Добавили НОВЫЙ @Step

```
public class BasicSteps {  
  
    @Step("Открываем страницу машины марки {0}")  
    @Step("Открываем страницу машины марки {mark}")  
    public void openAutoCardPage(String mark) { ... }  
  
    @Step("Добавляем заметку {0} к машине")  
    @Step("Добавляем заметку {text} к машине")  
    public void addNotesToAutoCard(String text) { ... }  
  
}
```

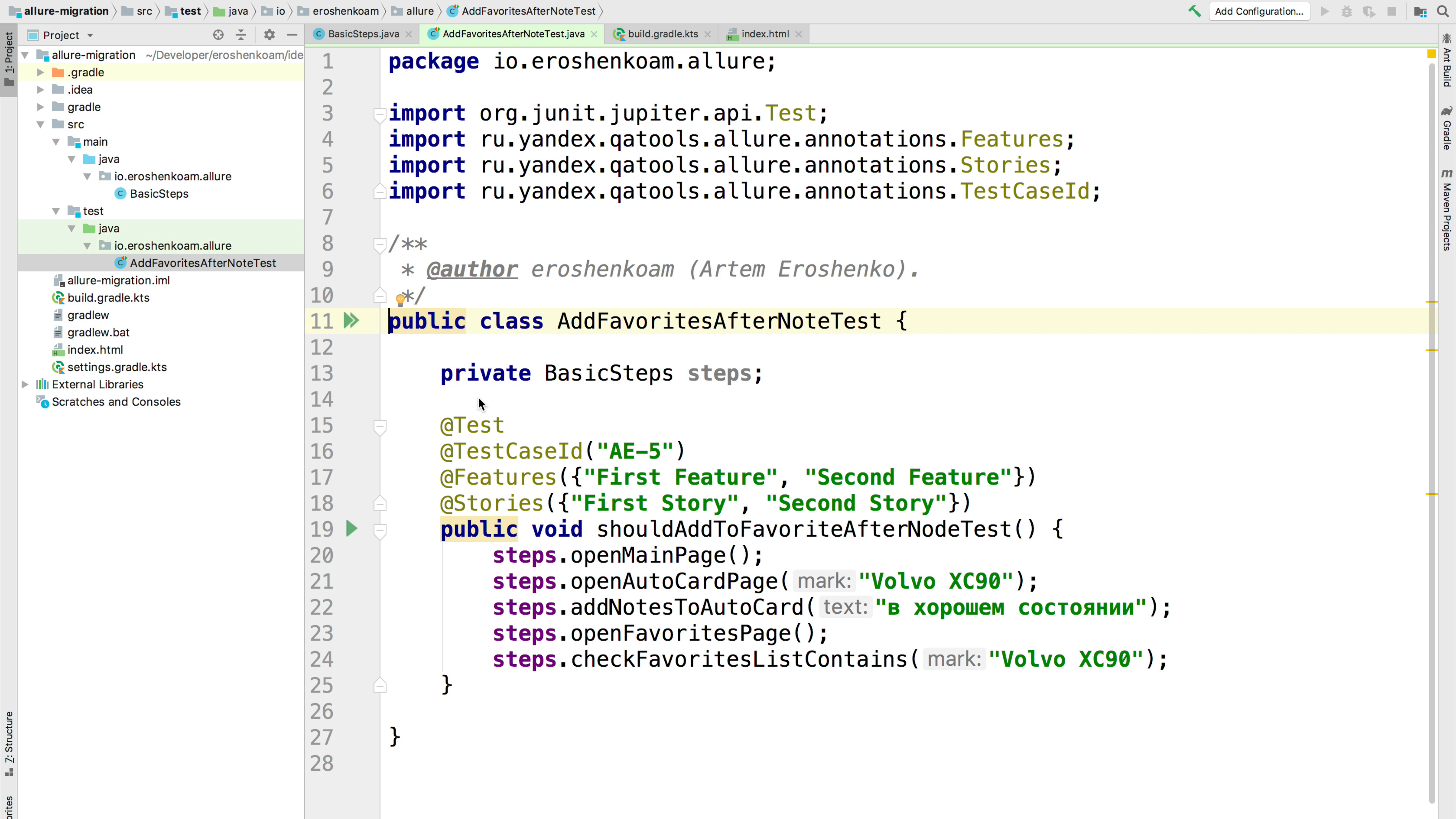
Удаляем старый @Step

```
public void migrateStepAnnotation(PsiMethod method) {  
    PsiAnnotation oldStep =  
        method.getAnnotation("ru..Step");  
    String oldstepText = getValue(oldStep, "value");  
    String[] params = Arrays.stream(method.getParameters())  
        .map(JvmParameter::getName).toArray(String[]::new);  
    String newStepText = convert(oldStepText, params);  
    PsiAnnotation newStep = createStep(newStepText);  
    method.getModifiersList()  
        .addAfter(newStep, oldStep);  
    oldStep.delete();  
}
```

Удалили старый @Step

```
public class BasicSteps {  
  
    @Step("Открываем страницу машины марки {mark}")  
    public void openAutoCardPage(String mark) { ... }  
  
    @Step("Добавляем заметку {text} к машине")  
    public void addNotesToAutoCard(String text) { ... }  
  
}
```



Project

- allure-migration ~/Developer/eroshenkoam/ide
 - .gradle
 - .idea
 - gradle
 - src
 - main
 - java
 - io.eroshenkoam.allure
 - BasicSteps
 - test
 - java
 - io.eroshenkoam.allure
 - AddFavoritesAfterNoteTest
 - allure-migration.iml
 - build.gradle.kts
 - gradlew
 - gradlew.bat
 - index.html
 - settings.gradle.kts

External Libraries

Scratches and Consoles

```
3 import ru.yandex.qatools.allure.annotations.Step;
4
5 /**
6  * @author eroshenkoam (Artem Eroshenko).
7  */
8 public class BasicSteps {
9
10     @Step("Открываем главную страницу")
11     public void openMainPage() {}
12
13
14
15     @Step("Открываем страницу машины марки {0}")
16     public void openAutoCardPage(String mark) {}
17
18
19
20     @Step("Добавляем заметку {0} к машине")
21     public void addNotesToAutoCard(String text) {}
22
23
24
25     @Step("Открываем страницу избранного")
26     public void openFavoritesPage() {}
27
28
29
30     @Step("Проверяем что список избранного соержит марку {0}")
31     public void checkFavoritesListContains(String mark) {}
32
33
34
35 }
36
```

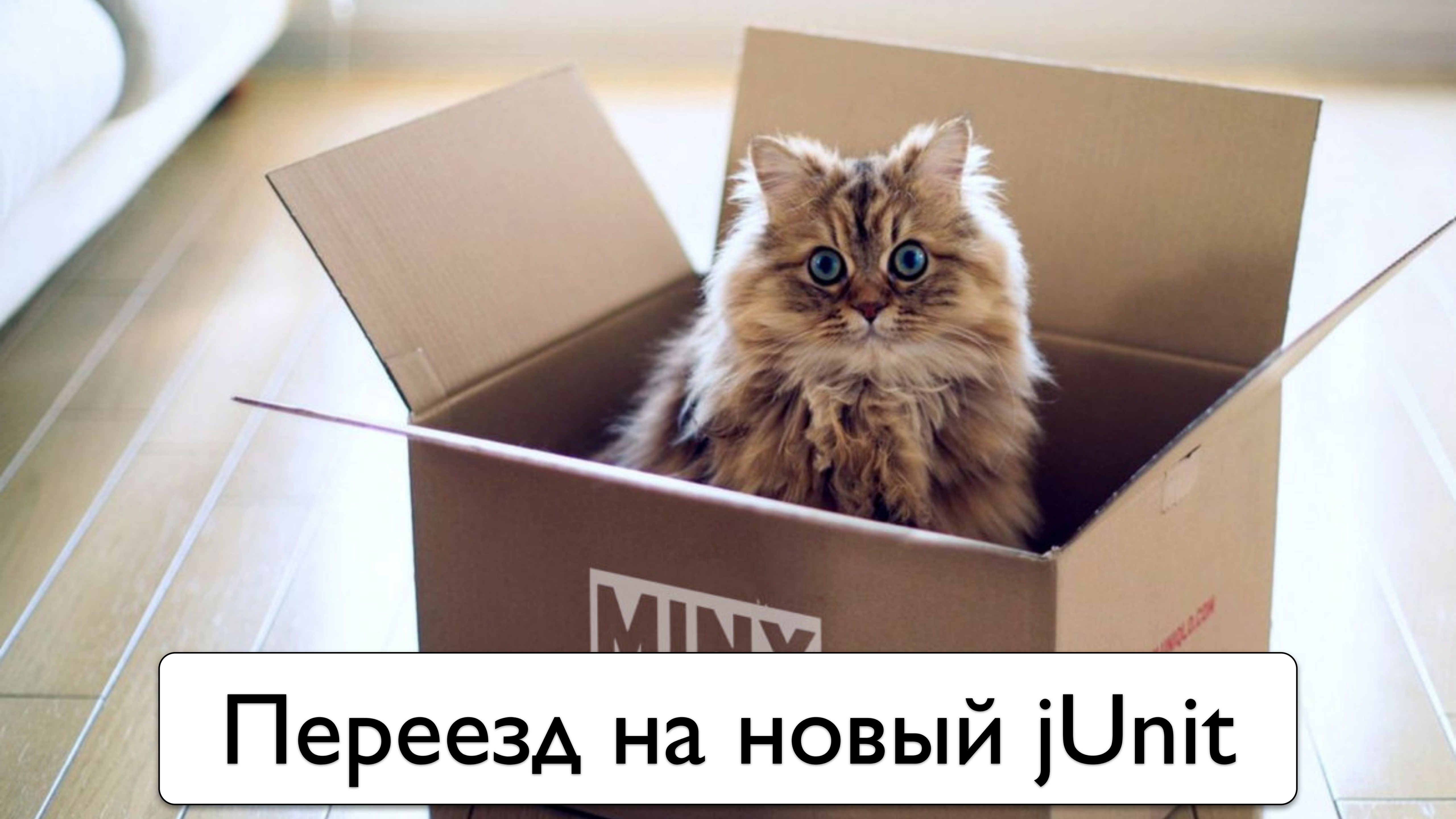
Ant Build

Gradle

Maven Projects

Какие еще есть идеи?



A fluffy brown and white cat with blue eyes is sitting inside an open cardboard box on a wooden floor. The box has the word 'MINI' printed on it. The cat is looking directly at the camera with a curious expression.

Переезд на новый jUnit

Guice - Dependency Injection



 ФИЛЬТРЫ



HEISENBUG
// 2017 Moscow

Артем Ерошенко
Независимый консультант

Простота, доверие,
контроль — три кита
автоматизации
веб-тестирования

1:00:36

Артем Ерошенко — Простота,
доверие, контроль — три
Heisenbug •

1,7 тыс. просмотров • 8 месяцев назад

Следующий Heisenbug уже скоро! 6-7 декабря,
Москва — Heisenbug 2018 Moscow. Инструменты

Как ВЫГЛЯДИТ ТЕСТ на junit4

```
@RunWith(GuiceRunner.class)
@GuiceModules(WebModule.class)
public class JUnit4Test {
    @Rule
    @Inject
    private RuleChain rules; //WebDriverRule, ScreenshotOnFail

    @Inject
    private BaseSteps steps;

    @Test
    public void shouldAddToFavoritesAfterNodeTest() { ... }
}
```

Первый проект руками

JUnit5 ->



Как выглядит тест на jUnit5

```
@ExtendWith(GuiceExtension.class)
@IncludeModules(WebModule.class)
@ExtendWith({
    WebDriverExtension.class,
    ScreenshotOnFail.class
})
public class JUnit4Test {
    @Inject
    private BaseSteps steps;

    @Test
    public void shouldAddToFavoritesAfterNodeTest() { ... }
}
```

Заменяли Runner на Extension

```
@ExtendWith(GuiceExtension.class) // @RunWith(GuiceRunner.class)
@IncludeModules(WebModule.class)
@ExtendWith({
    WebDriverExtension.class,
    ScreenshotOnFail.class
})
public class JUnit4Test {
    @Inject
    private BaseSteps steps;

    @Test
    public void shouldAddToFavoritesAfterNodeTest() { ... }
}
```

Другая аннотация для модулей

```
@ExtendWith(GuiceExtension.class)
@IncludeModules(WebModule.class) // @GuiceModules(WebModule.class)
@ExtendWith({
    WebDriverExtension.class,
    ScreenshotOnFail.class
})
public class JUnit4Test {
    @Inject
    private BaseSteps steps;

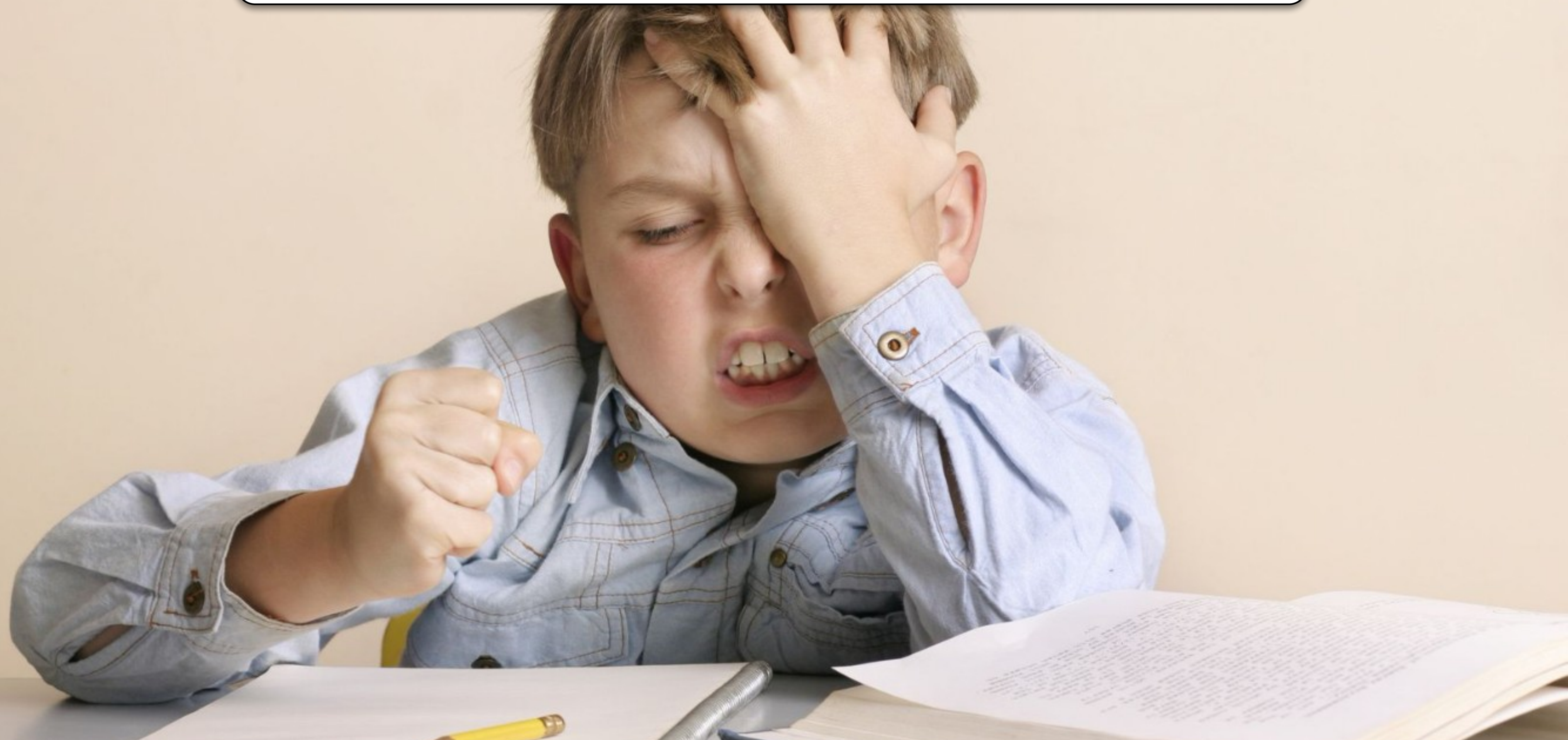
    @Test
    public void shouldAddToFavoritesAfterNodeTest() { ... }
}
```

Заменили Rule на Extension

```
@ExtendWith(GuiceExtension.class)
@IncludeModules(WebModule.class)
@ExtendWith({
    WebDriverExtension.class,
    ScreenshotOnFail.class
})
public class JUnit4Test {
    @Inject
    private BaseSteps steps;

    @Test
    public void shouldAddToFavoritesAfterNodeTest() { ... }
}
```

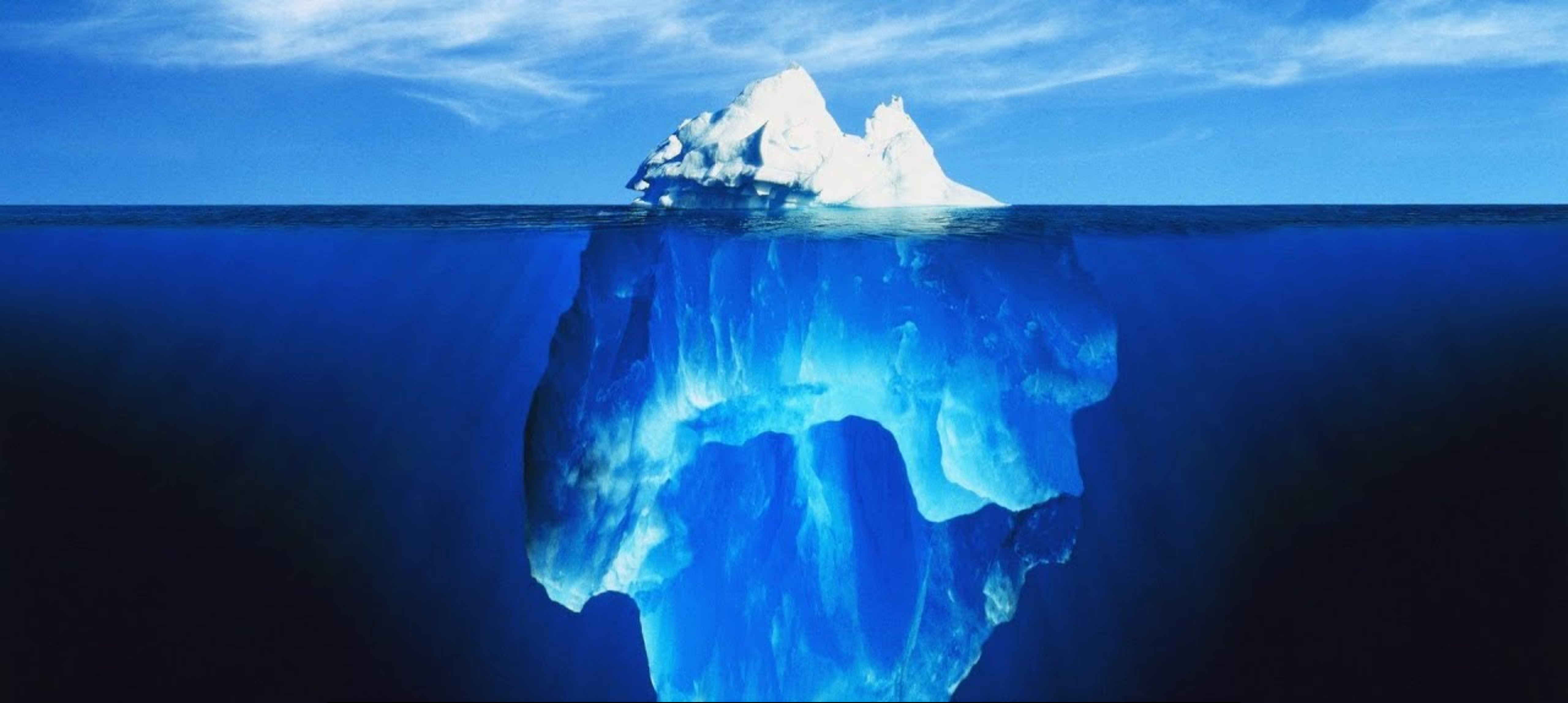
Домашнее задание



Импорт в проект

Экспорт из проекта


Миграция проекта



Вершина айсберга

Совершенствовать орудие труда



A dense, close-up background of numerous Poké Balls. The balls are arranged in a somewhat chaotic pattern, with some in the foreground and others receding into the background. Each ball is a vibrant red on the top half and a clean white on the bottom half, separated by a black horizontal band. A small, silver-colored circular detail is visible on the black band of each ball. The lighting creates soft highlights and shadows, giving the balls a three-dimensional appearance.

Собери свою коллекцию



Спасибо, Артем

**Вот так я и стал
автоматизатором**



Арте́м

Ероше́нко

@eroshenkoam

<https://github.com/eroshenkoam/idea-plugin-sample>

Вопросы?

