



Павел Стрельченко

Служебный роман плагин



Павел Стрельченко

Павел Стрельченко



Android-разработчик в HeadHunter

Ментор и лектор в Android Academy MSK

Увлекаюсь разработкой плагинов

О чем сегодня будем говорить?

- 1 Генерация boilerplate-кода
- 2 Автоматизация взаимодействия с JIRA
- 3 Добавление инспекций кода
- 4 Tips & tricks в разработке плагина

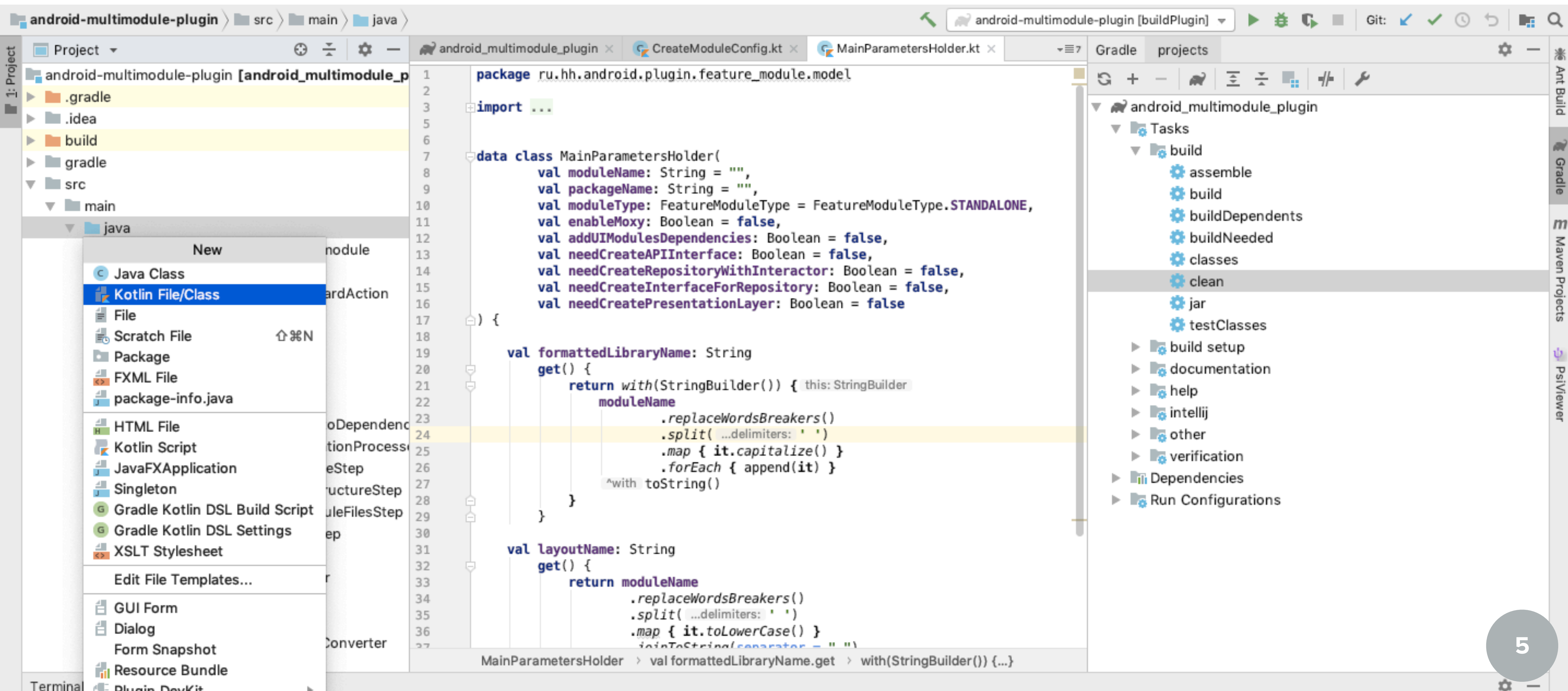


Ссылка на слайды

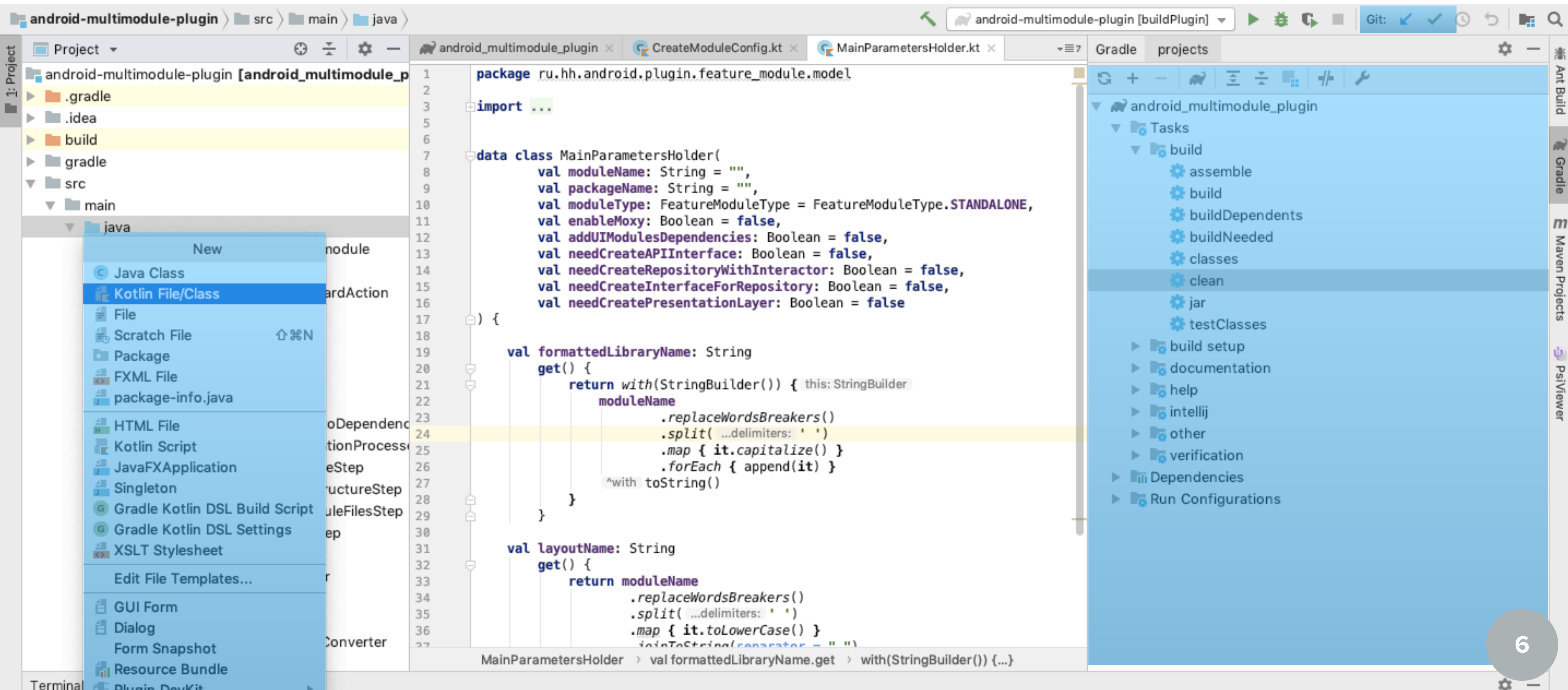


В предыдущей
серии

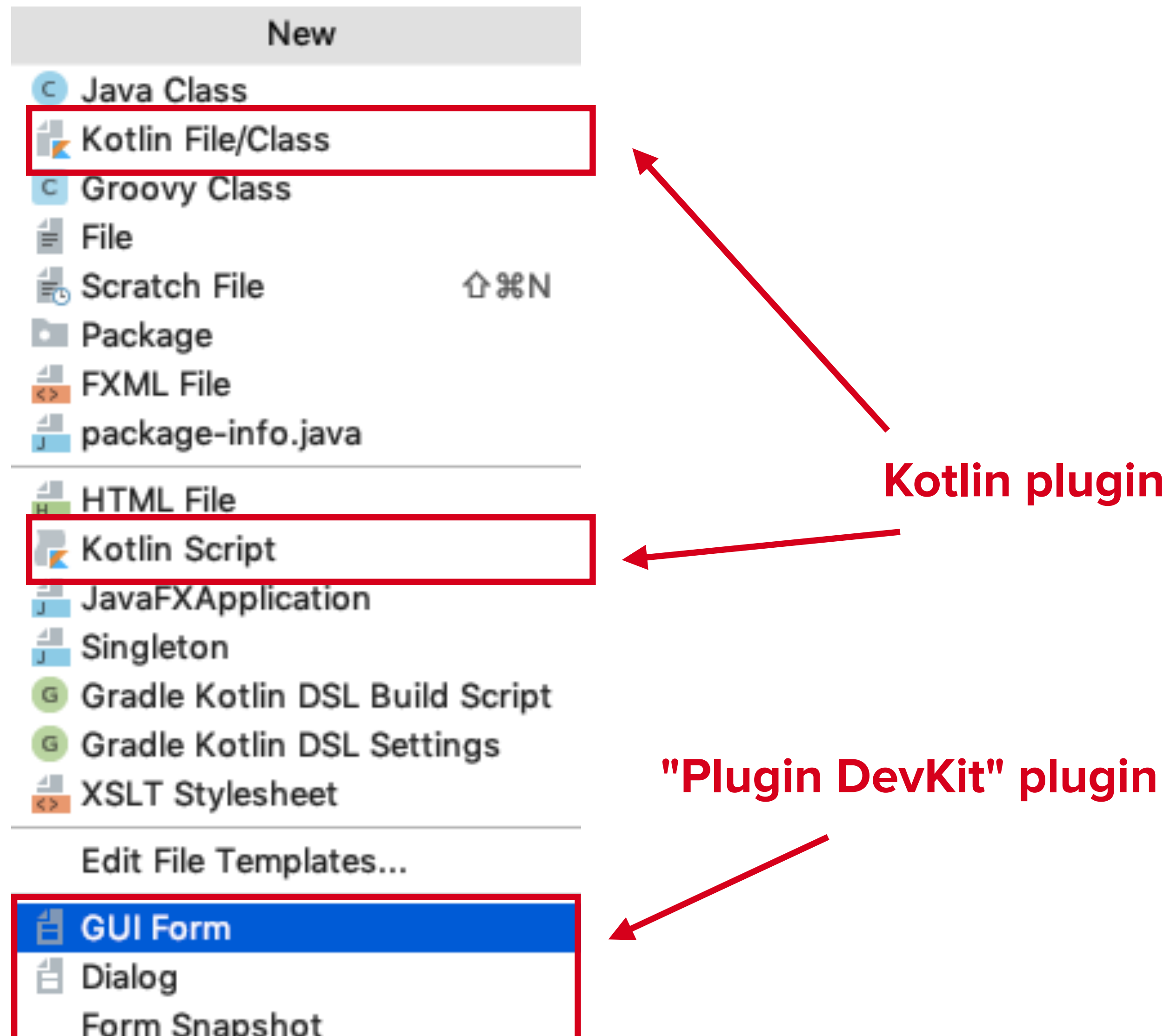
IntelliJ IDEA = ядро + плагины



IntelliJ IDEA = ядро + плагины



Плагины добавляют Action-ы



PSI = Program Structure Interface



- 1 PSI нужен для представления программы в IDEA
- 2 PSI-структурой пронизана вся IDEA
- 3 Для каждого языка программирования - свой PSI

gradle-intellij-plugin

- 1 **Позволяет использовать Gradle**
- 2 **Добавляет полезные gradle-таски**

2.1 **runIde**

2.2 **buildPlugin**

2.3 **verifyPlugin**

- 3 **Становится проще добавлять зависимости от других плагинов**





Генерация

boilerplate-кода



**Зачем это
моей команде?**

Несколько примеров

01

Добавление
аннотаций
`@SerializedName`

02

Генерация
EMPTY-объектов

03

Генерация
Page object-ов

@SerializedName-аннотации

01

Добавление
аннотаций
`@SerializedName`

02

Генерация
EMPTY-объектов

03

Генерация
Page object-ов

У нас много network-моделей

```
data class NotificationDataNetwork(  
    @SerializedName("text")  
    val text: String? = null,  
    @SerializedName("was_seen_by_user")  
    val wasSeenByUser: Boolean? = null,  
    @SerializedName("on_open_get_api_url")  
    val onOpenGetApiUrl: String? = null,  
    @SerializedName("delete_api_url")  
    val deleteApiUrl: String? = null,  
    @SerializedName("link_url")  
    val linkUrl: String? = null,  
    @SerializedName("link_mode")  
    val linkMode: String? = null  
)
```

```
data class ResumeViewedNetwork(  
    @SerializedName("created_at")  
    val createdAt: String? = null,  
    @SerializedName("employer")  
    val employer: FullEmployerNetwork? = null,  
    @SerializedName("viewed")  
    val viewed: Boolean? = null  
)
```

```
data class IdNameDictionaryNetwork(  
    @SerializedName("id")  
    val id: String? = null,  
    @SerializedName("name")  
    val name: String? = null  
)
```

Как жили без плагина?

```
1 package com.example.mylibrary
2
3
4 data class NotificationDataNetwork(
5     val wasSeenByUser: Boolean? = null,
6     val text: String? = null,
7     val onOpenGetApiUrl: String? = null,
8     val deleteApiUrl: String? = null,
9     val linkUrl: String? = null,
10    val linkMode: String? = null
11 )
```

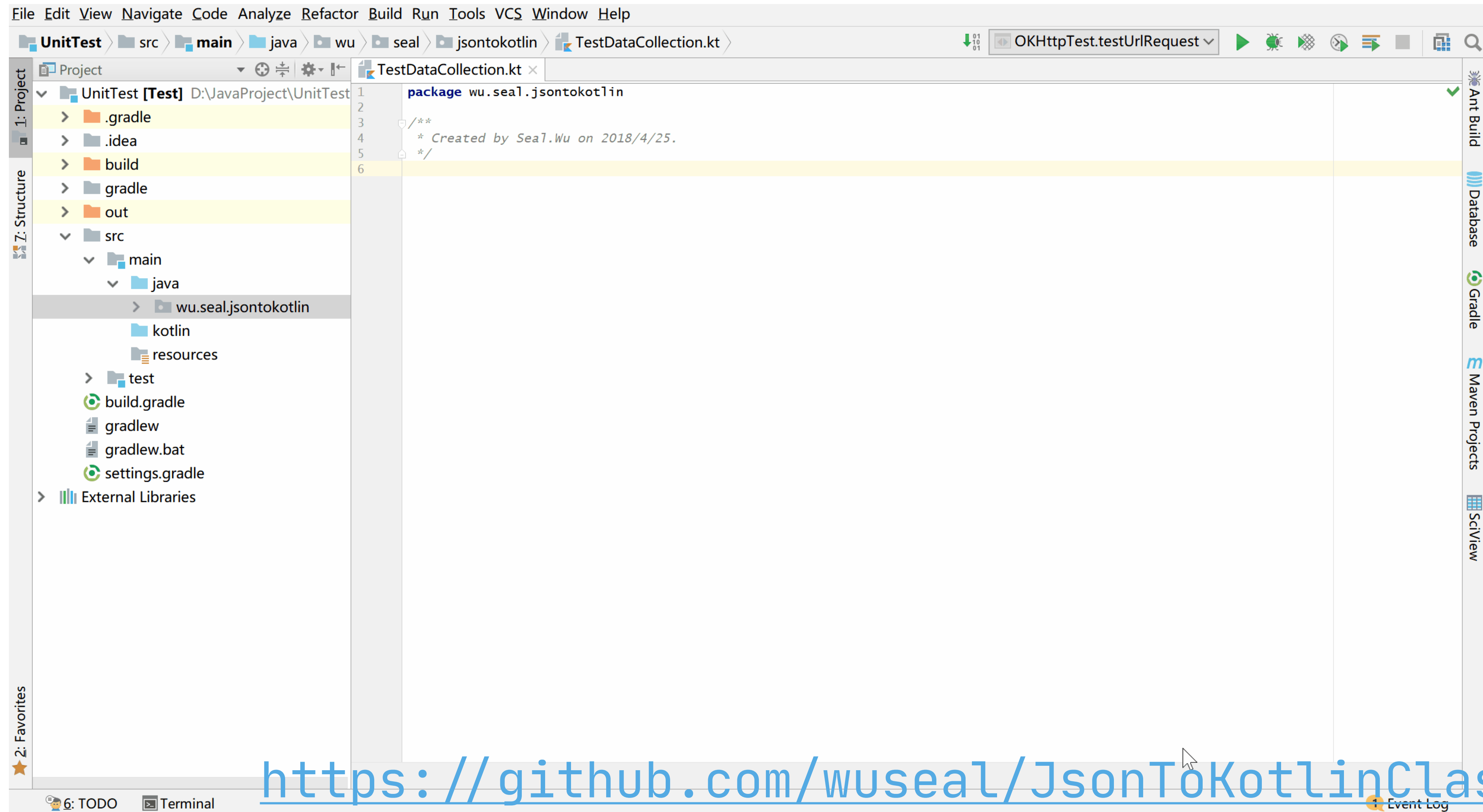


Как жили без плагина?

```
1 package com.example.mylibrary
2
3
4 data class NotificationDataNetwork(
5     val wasSeenByUser: Boolean? = null,
6     val text: String? = null,
7     val onOpenGetApiUrl: String? = null,
8     val deleteApiUrl: String? = null,
9     val linkUrl: String? = null,
10    val linkMode: String? = null
11 )
```

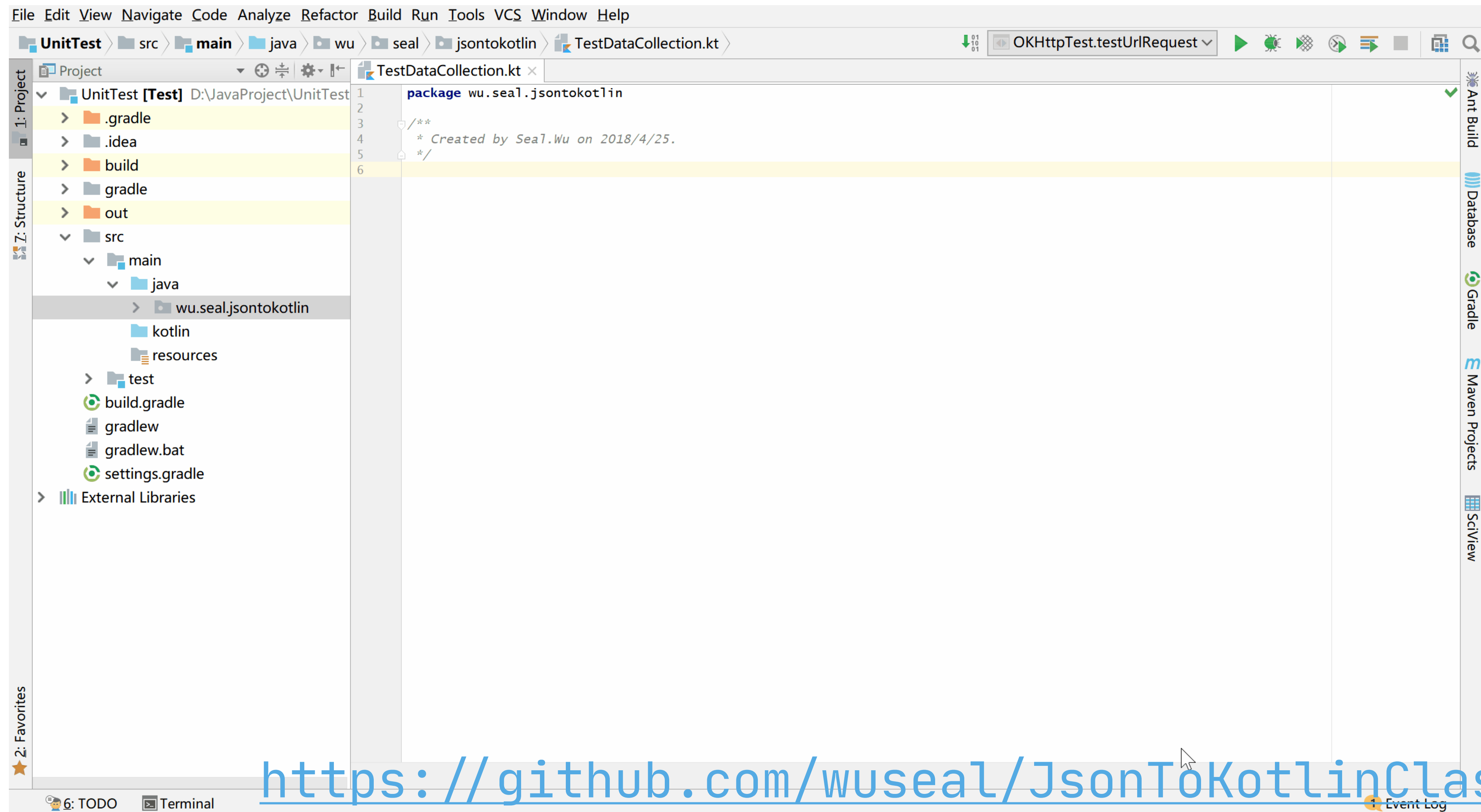


Как жили без плагина?



<https://github.com/wuseal/JsonToKotlinClass>

Как жили без плагина?



<https://github.com/wuseal/JsonToKotlinClass>

Что можно сделать плагином?

```
1 package com.example.mylibrary
2
3
4 data class NotificationDataNetwork(
5     val wasSeenByUser: Boolean? = null,
6     val text: String? = null,
7     val onOpenGetApiUrl: String? = null,
8     val deleteApiUrl: String? = null,
9     val linkUrl: String? = null,
10    val linkMode: String? = null
11 )
```



Что можно сделать плагином?

```
1 package com.example.mylibrary
2
3
4 data class NotificationDataNetwork(
5     val wasSeenByUser: Boolean? = null,
6     val text: String? = null,
7     val onOpenGetApiUrl: String? = null,
8     val deleteApiUrl: String? = null,
9     val linkUrl: String? = null,
10    val linkMode: String? = null
11 )
```



Генерация EMPTY-объектов

01

Добавление
аннотаций
`@SerializedName`

02

Генерация
EMPTY-объектов

03

Генерация
Page object-ов

У нас много domain-моделей

```
data class Photo(  
    val id: String,  
    val small: String,  
    val medium: String?  
) {  
  
    companion object {  
  
        val EMPTY = Photo(  
            id = String.EMPTY,  
            small = String.EMPTY,  
            medium = null  
        )  
  
    }  
  
}
```

```
data class SomeDataClass(  
    val id: Int,  
    val longValue: Long?,  
    val booleanValue: Boolean,  
    val stringValue: String,  
) {  
  
    companion object {  
  
        val EMPTY = SomeDataClass(  
            id = 0,  
            longValue = null,  
            booleanValue = false,  
            stringValue = String.EMPTY  
        )  
  
    }  
  
}
```

Как жили без плагина?

```
1 package com.example.mylibrary
2
3
4 data class Currency(
5     val rate: Double,
6     val code: String,
7     val abbr: String,
8     val inUse: Boolean,
9     val default: Boolean,
10    val name: String
11 )
```

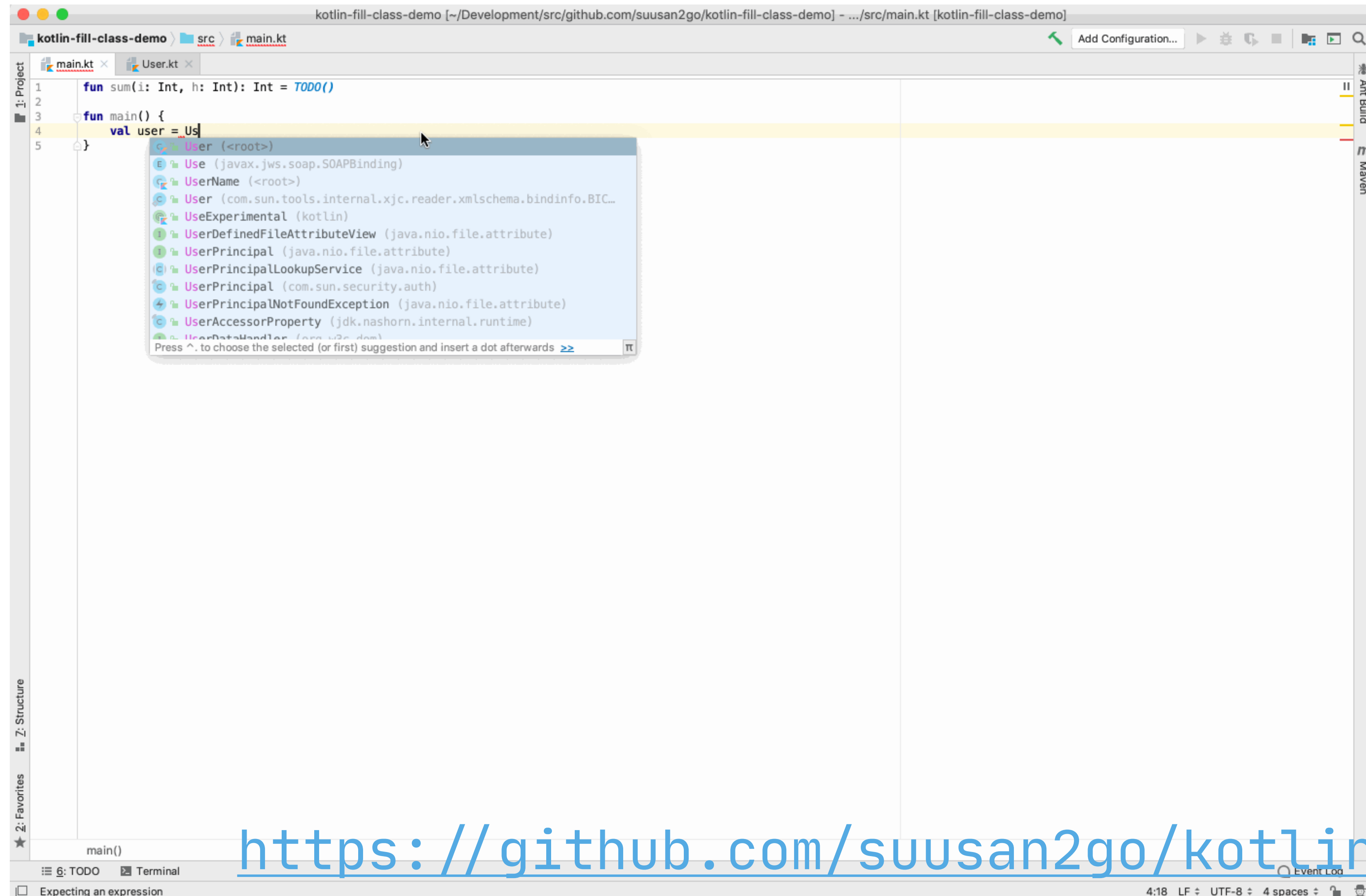


Как жили без плагина?

```
1      package com.example.mylibrary
2
3
4      data class Currency(
5          val rate: Double,
6          val code: String,
7          val abbr: String,
8          val inUse: Boolean,
9          val default: Boolean,
10         val name: String
11     )
```

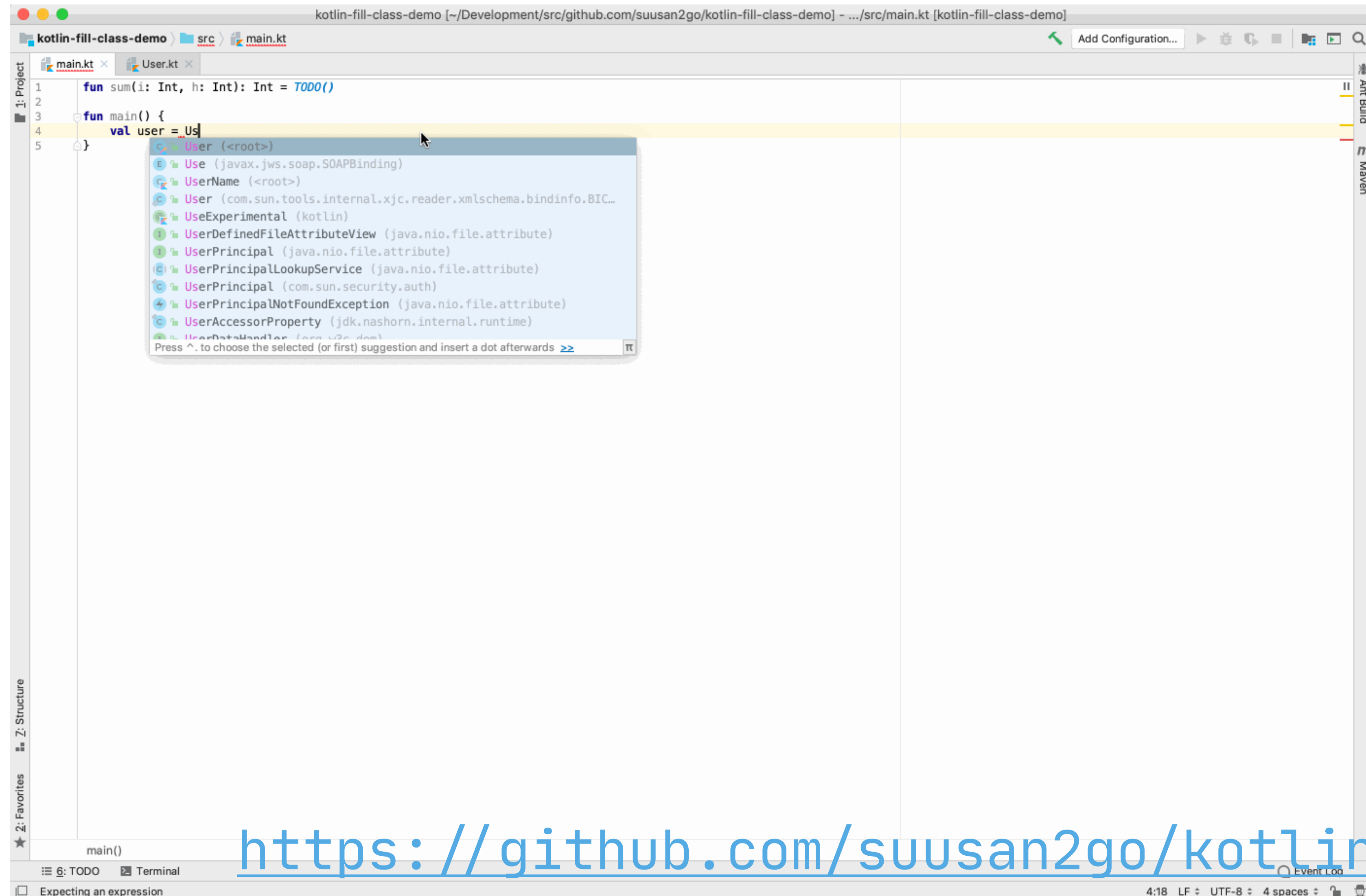


Как жили без плагина?



<https://github.com/suusan2go/kotlin-fill-class>

Как жили без плагина?



<https://github.com/suusan2go/kotlin-fill-class>

Что можно сделать плагином?

```
1 package com.example.mylibrary
2
3
4 data class Currency(
5     val rate: Double,
6     val code: String,
7     val abbr: String,
8     val inUse: Boolean,
9     val default: Boolean,
10    val name: String
11 )
```



Что можно сделать плагином?

```
1 package com.example.mylibrary
2
3
4 data class Currency(
5     val rate: Double,
6     val code: String,
7     val abbr: String,
8     val inUse: Boolean,
9     val default: Boolean,
10    val name: String
11 )
```



Генерация Page object-ов

01

Добавление
аннотаций
`@SerializedName`

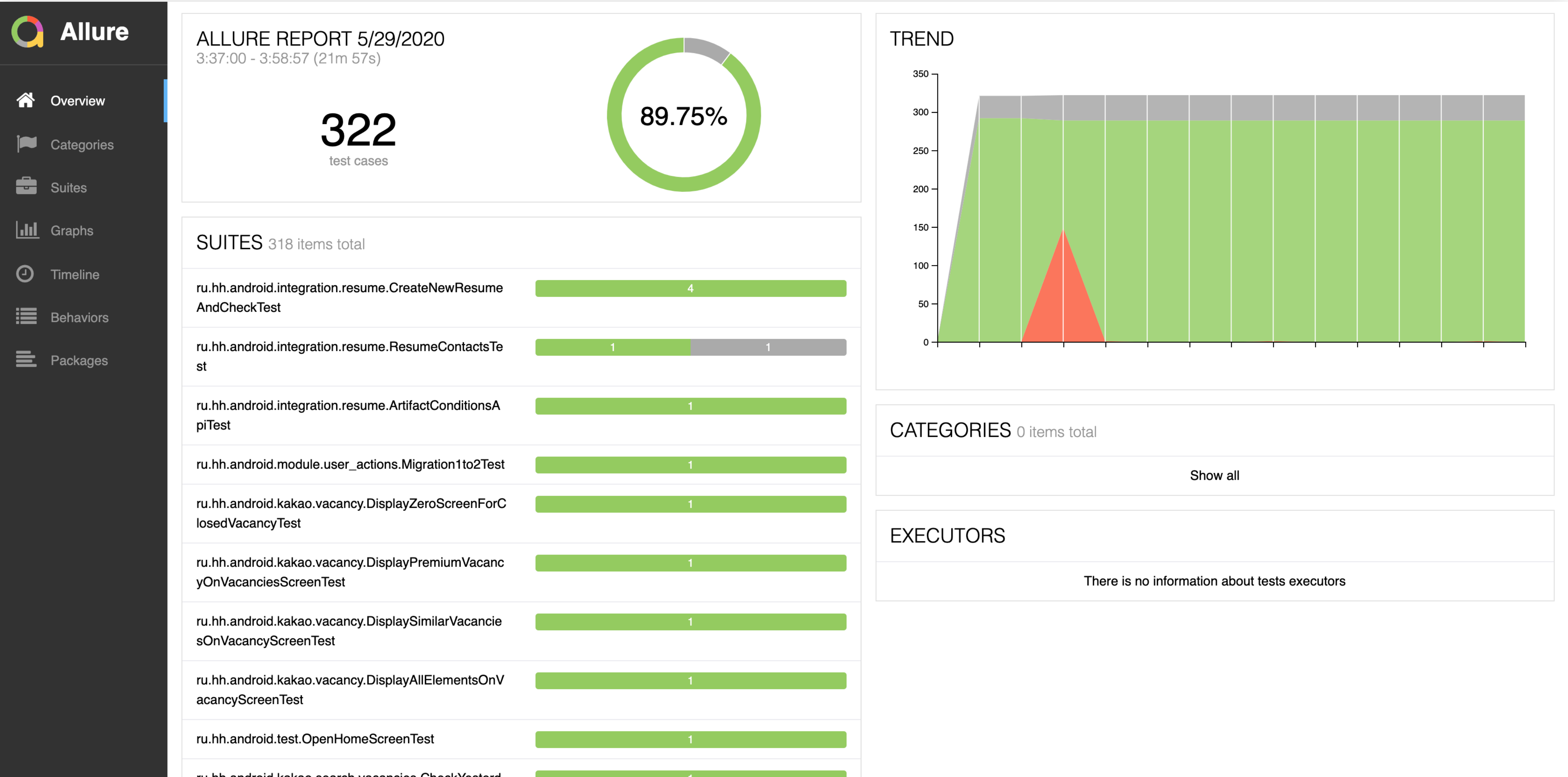
02

Генерация
EMPTY-объектов

03

Генерация
Page object-ов

У нас много UI-тестов



Для каждого нужны Page Object-ы

```
class FavouritesScreen : Screen<FavouritesScreen>() {

    val fragmentContainer = KView { withId(R.id.fragment_favorite_screen_container_container) }
    val favoriteButton = KButton(withId(R.id.toolbar_container_tab)) { withText("Вакансии") }
    val autosearchButton = KButton { withId(R.id.autosearch_tab_with_badge_title) }
    ...

    val actions = Action()
    val checks = Check()

    inner class Action : ScreenIntentions<Action>() {

        fun TestContext<TestCase>.longClickVacancy(position: Int) {
            step("Лонгклик по $position вакансии в избранном") {
                listFavouritesScreen {
                    waitForFirstChildVisible()
                    childAt<ItemFavorite>(position) {
                        longClick()
                    }
                }
            }
        }
    }
}
```

Как жили без плагина?

The screenshot shows the Android Studio IDE with the XML editor on the left and the preview window on the right. The XML code defines a dialog box with a title and a body text. The preview window shows the rendered dialog box with the text "Нет резюме для отклика" and two buttons: "Создать резюме" and "Удалить резюме".

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:layout_width="match_parent"
5     android:layout_height="wrap_content"
6     android:orientation="vertical"
7     android:paddingBottom="16dp">
8
9     <View
10         style="@style/BottomSheetIndicatorStyle"
11         android:layout_width="36dp"
12         android:layout_height="4dp"
13         android:layout_gravity="center_horizontal" />
14
15     <TextView
16         android:id="@+id/fragment_action_bottom_sheet_text_view_title"
17         style="@style/Title1.LightBlack"
18         android:layout_width="match_parent"
19         android:layout_height="wrap_content"
20         android:layout_marginStart="12dp"
21         android:layout_marginTop="20dp"
22         android:layout_marginEnd="12dp"
23         android:gravity="center_horizontal"
24         tools:text="Нет резюме для отклика" />
25
26     <TextView
27         android:id="@+id/fragment_action_bottom_sheet_text_view_body"
```

The preview window shows a dialog box with the following content:

Нет резюме для отклика

У вас нет ни одного резюме
Хотите создать и опубликовать
резюме прямо сейчас?

Создать резюме

Удалить резюме

Как жили без плагина?

The screenshot displays the Android Studio interface. On the left, the XML code for a layout is shown, with line numbers 1 through 27. The code defines a `LinearLayout` with a vertical orientation and a `paddingBottom` of 16dp. Inside the `LinearLayout`, there is a `View` with a style of `@style/BottomSheetIndicatorStyle`, a width of 36dp, a height of 4dp, and a gravity of `center_horizontal`. Below the `View` is a `TextView` with an ID of `@+id/fragment_action_bottom_sheet_text_view_title`, a style of `@style/Title1.LightBlack`, a width of `match_parent`, a height of `wrap_content`, a margin start of 12dp, a margin top of 20dp, a margin end of 12dp, a gravity of `center_horizontal`, and text that says "Нет резюме для отклика". Below the `TextView` is another `TextView` with an ID of `@+id/fragment_action_bottom_sheet_text_view_body`.

On the right, the preview window shows a mobile device screen with a dialog box. The dialog box has a title "Нет резюме для отклика" and a message "У вас нет ни одного резюме. Хотите создать и опубликовать резюме прямо сейчас?". There are two buttons: "Создать резюме" (Create resume) and "Удалить резюме" (Delete resume). The preview window also shows a "Component Tree" on the left and "Attributes" on the right.

Что можно сделать плагином?

The screenshot displays the Android Studio interface. On the left, the XML editor shows the following code:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   xmlns:tools="http://schemas.android.com/tools"
4   android:layout_width="match_parent"
5   android:layout_height="wrap_content"
6   android:orientation="vertical"
7   android:paddingBottom="16dp">
8
9   <View
10    style="@style/BottomSheetIndicatorStyle"
11    android:layout_width="36dp"
12    android:layout_height="4dp"
13    android:layout_gravity="center_horizontal" />
14
15   <TextView
16    android:id="@+id/fragment_action_bottom_sheet_text_view_title"
17    style="@style/Title1.LightBlack"
18    android:layout_width="match_parent"
19    android:layout_height="wrap_content"
20    android:layout_marginStart="12dp"
21    android:layout_marginTop="20dp"
22    android:layout_marginEnd="12dp"
23    android:gravity="center_horizontal"
24    tools:text="Нет резюме для отклика" />
25
26   <TextView
27    android:id="@+id/fragment_action_bottom_sheet_text_view_body"
```

On the right, the preview window shows a dialog box with the following text:

Нет резюме для отклика
У вас нет ни одного резюме
Хотите создать и опубликовать
резюме прямо сейчас?

Below the text are two buttons: "Создать резюме" (Create resume) and "Удалить резюме" (Delete resume).

The interface also includes a Palette on the left, a Component Tree on the bottom left, and an Attributes panel on the right.

Что можно сделать плагином?

The screenshot displays the Android Studio interface. On the left, the XML editor shows the following code:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   xmlns:tools="http://schemas.android.com/tools"
4   android:layout_width="match_parent"
5   android:layout_height="wrap_content"
6   android:orientation="vertical"
7   android:paddingBottom="16dp">
8
9   <View
10    style="@style/BottomSheetIndicatorStyle"
11    android:layout_width="36dp"
12    android:layout_height="4dp"
13    android:layout_gravity="center_horizontal" />
14
15   <TextView
16    android:id="@+id/fragment_action_bottom_sheet_text_view_title"
17    style="@style/Title1.LightBlack"
18    android:layout_width="match_parent"
19    android:layout_height="wrap_content"
20    android:layout_marginStart="12dp"
21    android:layout_marginTop="20dp"
22    android:layout_marginEnd="12dp"
23    android:gravity="center_horizontal"
24    tools:text="Нет резюме для отклика" />
25
26   <TextView
27    android:id="@+id/fragment_action_bottom_sheet_text_view_body"
```

On the right, the preview window shows a dialog box with the following text:

Нет резюме для отклика
У вас нет ни одного резюме
Хотите создать и опубликовать
резюме прямо сейчас?

Below the text are two buttons: "Создать резюме" (Create resume) and "Удалить резюме" (Delete resume).

The interface also includes a Palette on the left, a Component Tree on the bottom left, and an Attributes panel on the right.

Общий алгоритм

- 1 **Определить контекст, в котором будет работать наш Action**
- 2 **Получить текст будущего кусочка кода и преобразовать текст в PSI**
- 3 **Точно определить, в какое конкретно место хотим добавить код**

Общий алгоритм

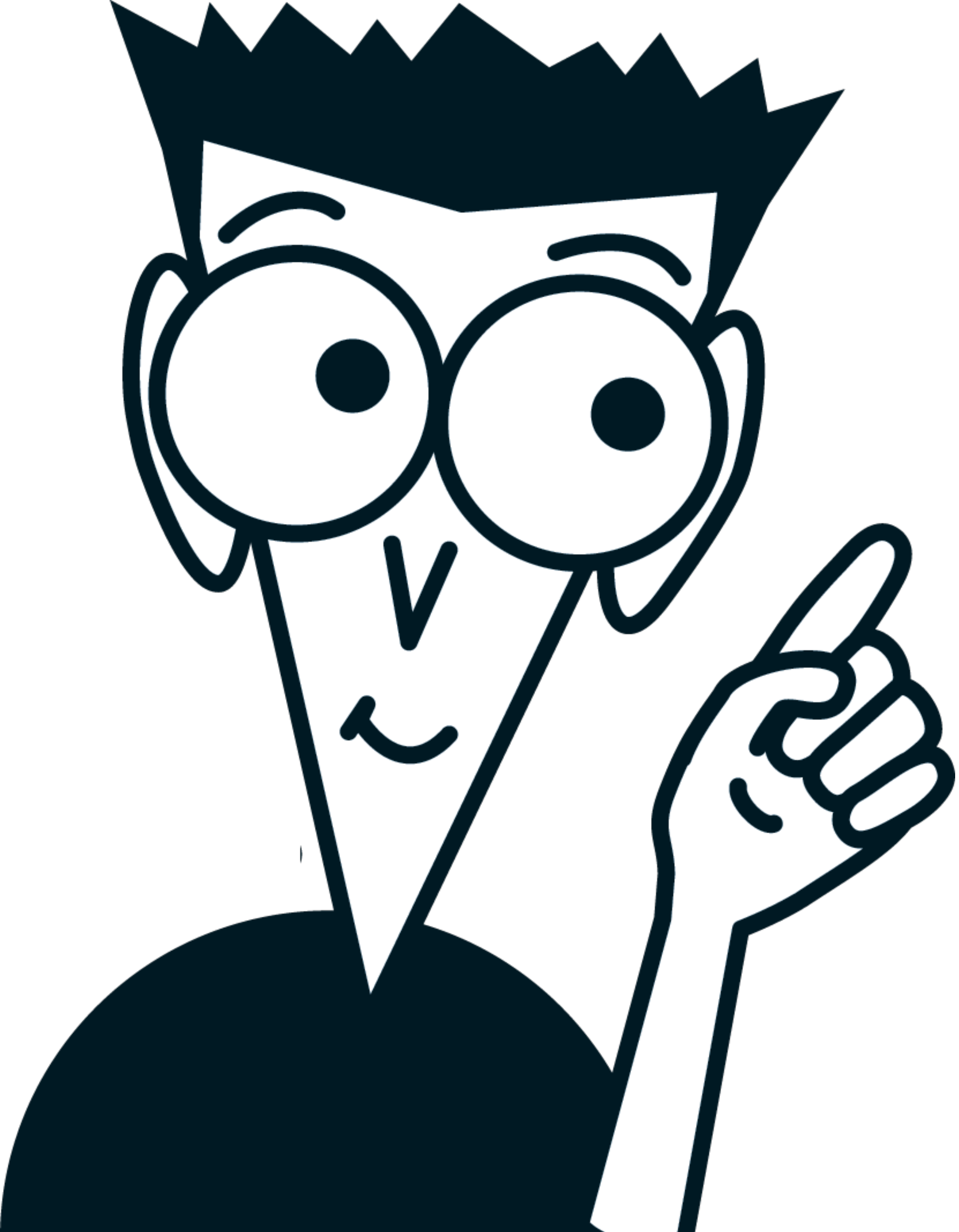
- 1** Определить контекст, в котором будет работать наш Action
- 2 Получить текст будущего кусочка кода и преобразовать текст в PSI
- 3 Точно определить, в какое конкретно место хотим добавить код

Общий алгоритм

- 1 Определить контекст, в котором будет работать наш Action
- 2 Получить текст будущего кусочка кода и преобразовать текст в PSI**
- 3 Точно определить, в какое конкретно место хотим добавить код

Общий алгоритм

- 1 Определить контекст, в котором будет работать наш Action
- 2 Получить текст будущего кусочка кода и преобразовать текст в PSI
- 3 Точно определить, в какое конкретно место хотим добавить код**



Определение контекста

Есть несколько способов...

- 1 Получить данные о контексте из `AnActionEvent`
- 2 Использовать `CodeInsightAction`

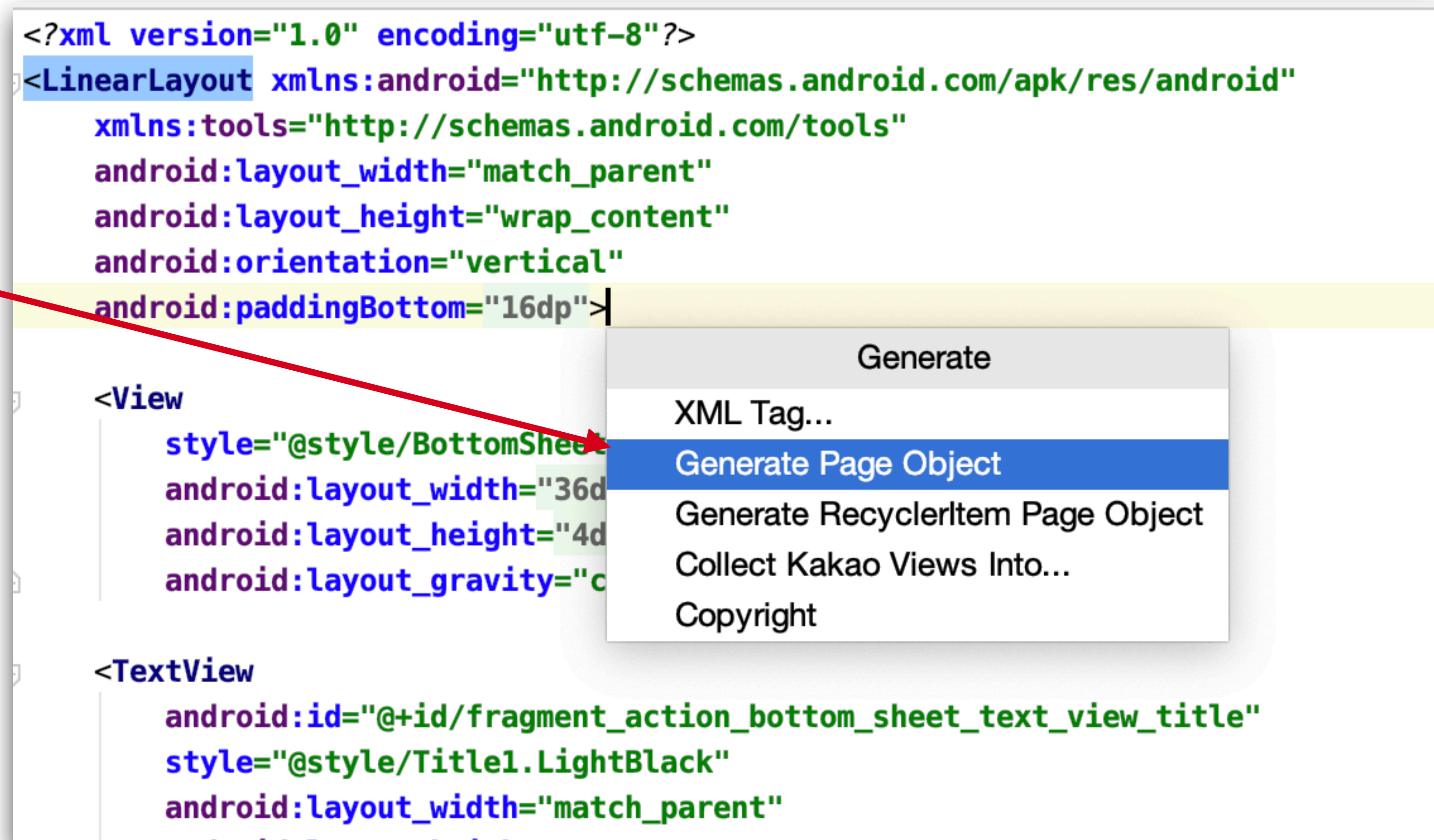
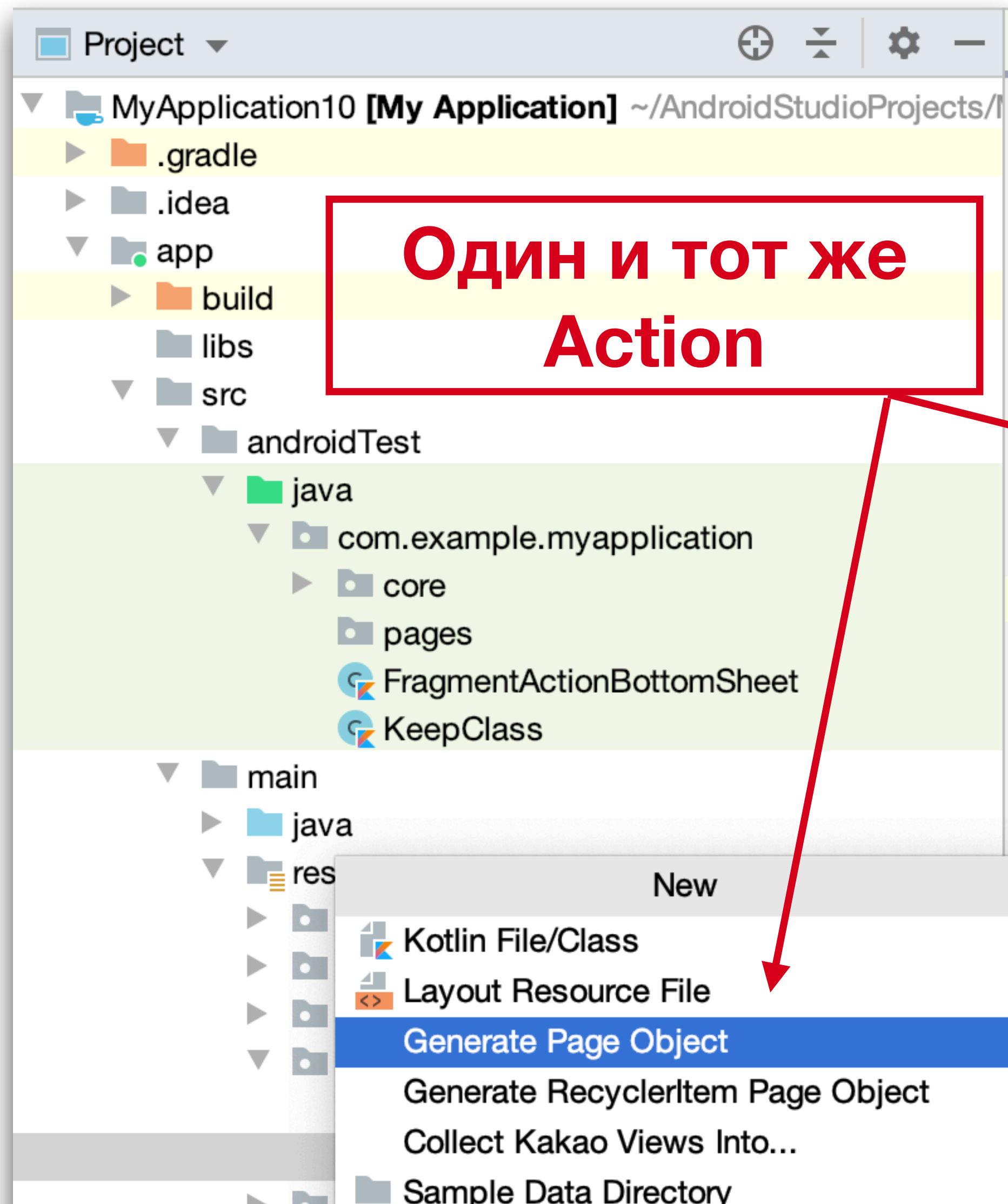
Что такое AnActionEvent

```
class MySimpleAction : AnAction() {  
  
    override fun update(e: AnActionEvent) {  
        super.update(e)  
        e.presentation.isEnabled = e.checkSomeCondition()  
    }  
  
    override fun actionPerformed(e: AnActionEvent) {  
        e.project?.let { project →  
            // TODO to some stuff  
        }  
    }  
}
```

Что такое AnActionEvent

```
class MySimpleAction : AnAction() {  
  
    override fun update(e: AnActionEvent) {  
        super.update(e)  
        e.presentation.isEnabled = e.checkSomeCondition()  
    }  
  
    override fun actionPerformed(e: AnActionEvent) {  
        e.project?.let { project →  
            // TODO to some stuff  
        }  
    }  
  
}
```

Когда нужен AnActionEvent



Получаем класс из редактора кода

```
data class NotificationDataNetwork(  
    val wasSeenByUser: Boolean? = null,  
    val text: String? = null,  
    val onOpenGetApiUrl: String,  
    val pp: Int,  
    val deleteApiUrl: String? = null,  
    val linkUrl: String? = null,  
    val linkMode: String? = null  
)
```

```
data class NotificationDataNetwork(  
    val wasSeenByUser: Boolean? = null,  
    val text: String? = null,  
    val onOpenGetApiUrl: String,  
    val pp: Int,  
    val deleteAp  
    val linkUrl:  
    val linkMode
```

Generate

- equals() and hashCode()
- toString()
- Generate EMPTY Object...**
- Generate @SerializedName Annotations...
- Override Methods... ^O
- Implement Methods... ^|
- Test...
- Copyright

Получаем класс из редактора кода

```
private val AnActionEvent.ktClassFromEditor: KtClass?
    get() {
        val currentPsiFile = getData(LangDataKeys.PSI_FILE)
        val editor = getData(PlatformDataKeys.EDITOR)

        if (currentPsiFile == null || editor == null) {
            return null
        }

        val offset = editor.caretModel.offset
        val psiElement = currentPsiFile.findElementAt(offset)

        return PsiTreeUtil.getParentOfType(psiElement, KtClass::class.java)
    }
```

Тянем данные из AnActionEvent

```
private val AnActionEvent.ktClassFromEditor: KtClass?
    get() {
        val currentPsiFile = getData(LangDataKeys.PSI_FILE)
        val editor = getData(PlatformDataKeys.EDITOR)

        if (currentPsiFile == null || editor == null) {
            return null
        }

        val offset = editor.caretModel.offset
        val psiElement = currentPsiFile.findElementAt(offset)

        return PsiTreeUtil.getParentOfType(psiElement, KtClass::class.java)
    }
```

Получили файл и редактор кода

```
private val AnActionEvent.ktClassFromEditor: KtClass?
    get() {
        val currentPsiFile = getData(LangDataKeys.PSI_FILE)
        val editor = getData(PlatformDataKeys.EDITOR)

        if (currentPsiFile == null || editor == null) {
            return null
        }

        val offset = editor.caretModel.offset
        val psiElement = currentPsiFile.findElementAt(offset)

        return PsiTreeUtil.getParentOfType(psiElement, KtClass::class.java)
    }
```

Получаем положение курсора

```
private val AnActionEvent.ktClassFromEditor: KtClass?
    get() {
        val currentPsiFile = getData(LangDataKeys.PSI_FILE)
        val editor = getData(PlatformDataKeys.EDITOR)

        if (currentPsiFile == null || editor == null) {
            return null
        }

        val offset = editor.caretModel.offset
        val psiElement = currentPsiFile.findElementAt(offset)

        return PsiTreeUtil.getParentOfType(psiElement, KtClass::class.java)
    }
```

Находим выделенный элемент

```
private val AnActionEvent.ktClassFromEditor: KtClass?
get() {
    val currentPsiFile = getData(LangDataKeys.PSI_FILE)
    val editor = getData(PlatformDataKeys.EDITOR)

    if (currentPsiFile == null || editor == null) {
        return null
    }

    val offset = editor.caretModel.offset
    val psiElement = currentPsiFile.findElementAt(offset)

    return PsiTreeUtil.getParentOfType(psiElement, KtClass::class.java)
}
```

Получаем класс по дереву PSI

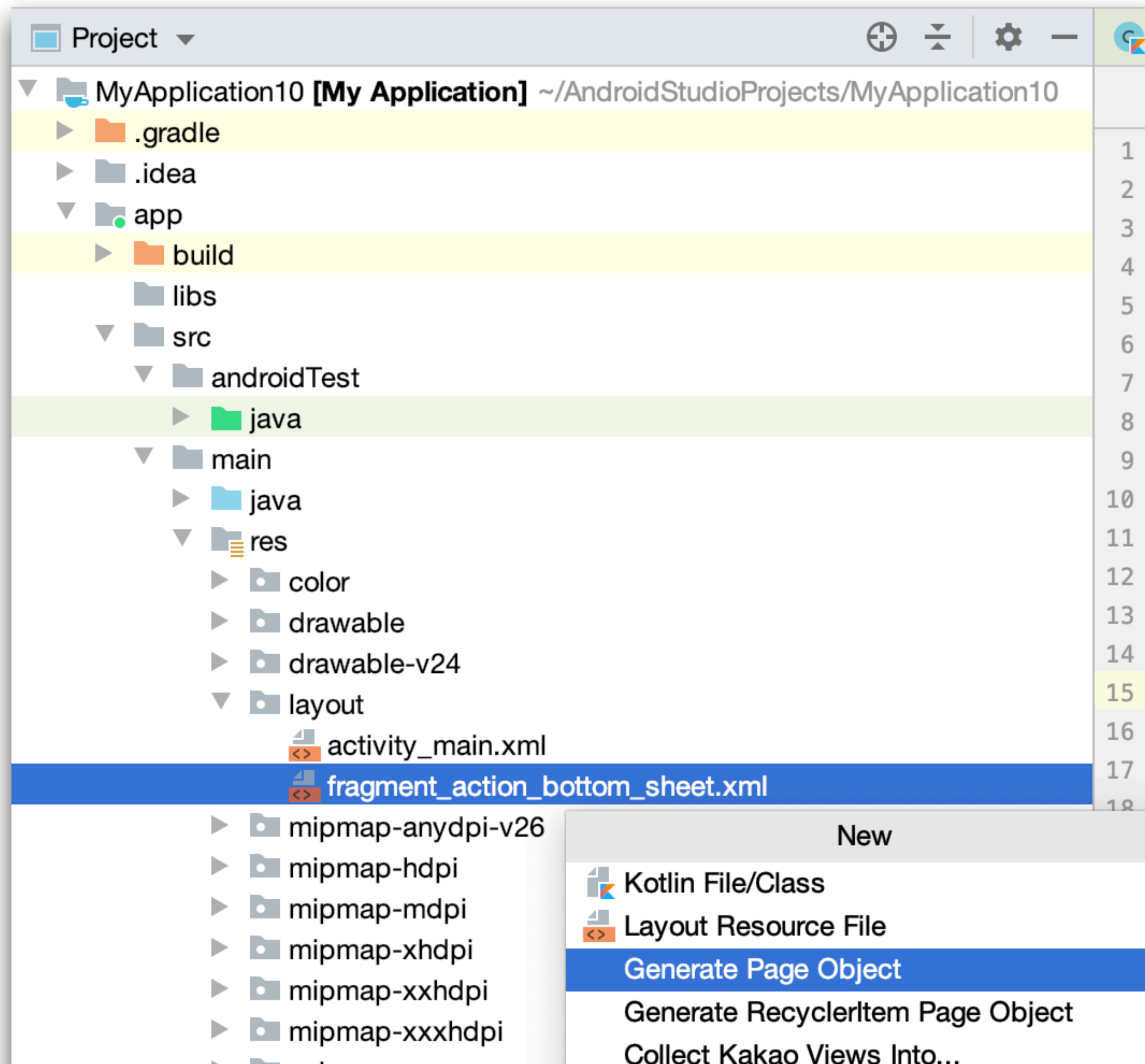
```
private val AnActionEvent.ktClassFromEditor: KtClass?
    get() {
        val currentPsiFile = getData(LangDataKeys.PSI_FILE)
        val editor = getData(PlatformDataKeys.EDITOR)

        if (currentPsiFile == null || editor == null) {
            return null
        }

        val offset = editor.caretModel.offset
        val psiElement = currentPsiFile.findElementAt(offset)

        return PsiTreeUtil.getParentOfType(psiElement, KtClass::class.java)
    }
```

Получаем контекст из Project View



Получаем XML-файл из Project View

```
private val AnActionEvent.selectedKtClass: XmlFile?  
    get() {  
        val selectedPsiElement = getData(PlatformDataKeys.PSI_ELEMENT)  
  
        return (selectedPsiElement as? XmlFile)  
            ?.takeIf { it.parent?.name == "layout" }  
    }
```

Достаем PsiElement из event-a

```
private val AnActionEvent.selectedKtClass: XmlFile?  
    get() {  
        val selectedPsiElement = getData(PlatformDataKeys.PSI_ELEMENT)  
  
        return (selectedPsiElement as? XmlFile)  
            ?.takeIf { it.parent?.name == "layout" }  
    }
```

Формируем контекст для res/layout

```
private val AnActionEvent.selectedKtClass: XmlFile?
    get() {
        val selectedPsiElement = getData(PlatformDataKeys.PSI_ELEMENT)

        return (selectedPsiElement as? XmlFile)
            ?.takeIf { it.parent?.name == "layout" }
    }
```

А можно как-нить
полегче?...

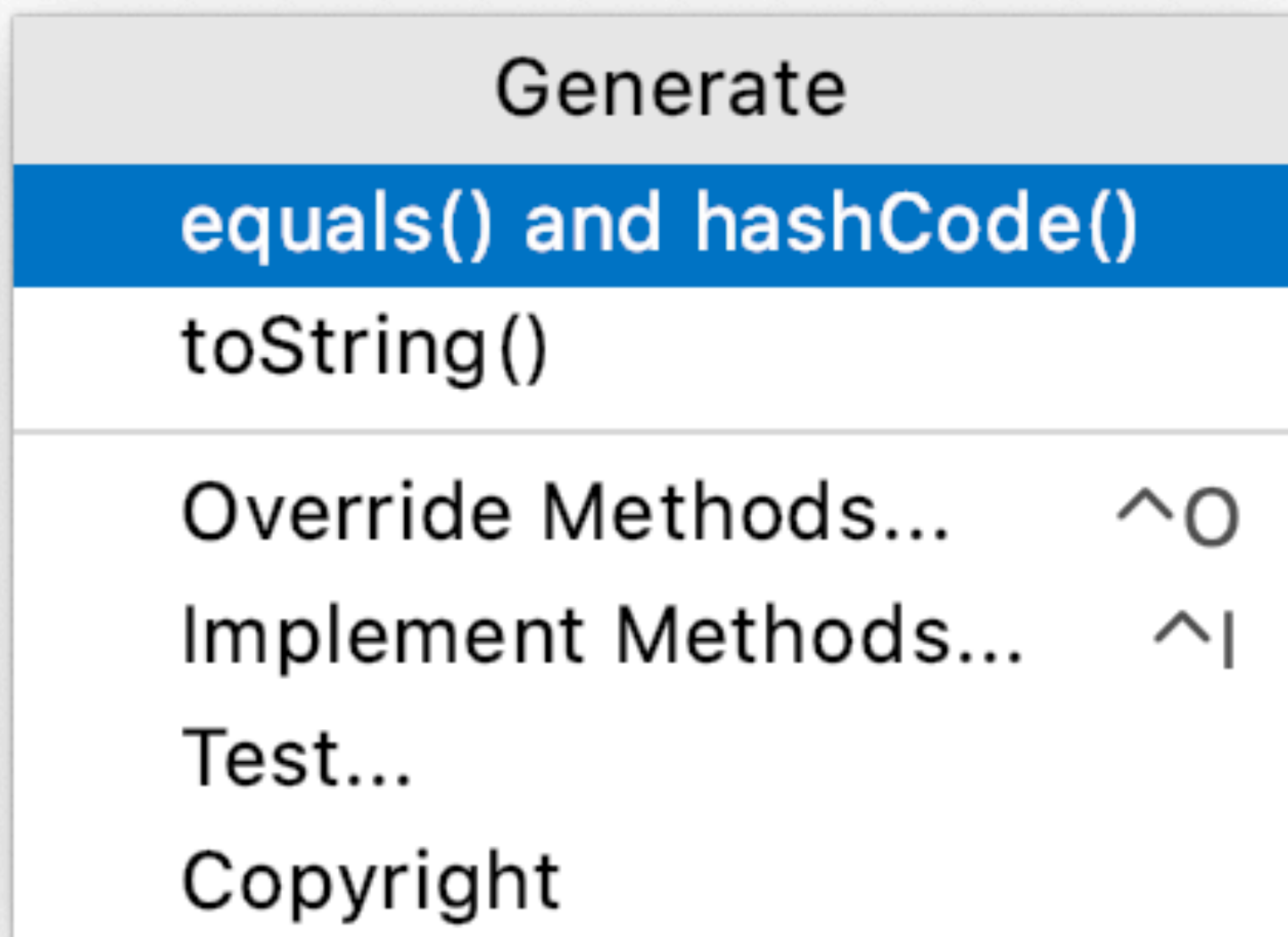
Конечно

Конечно

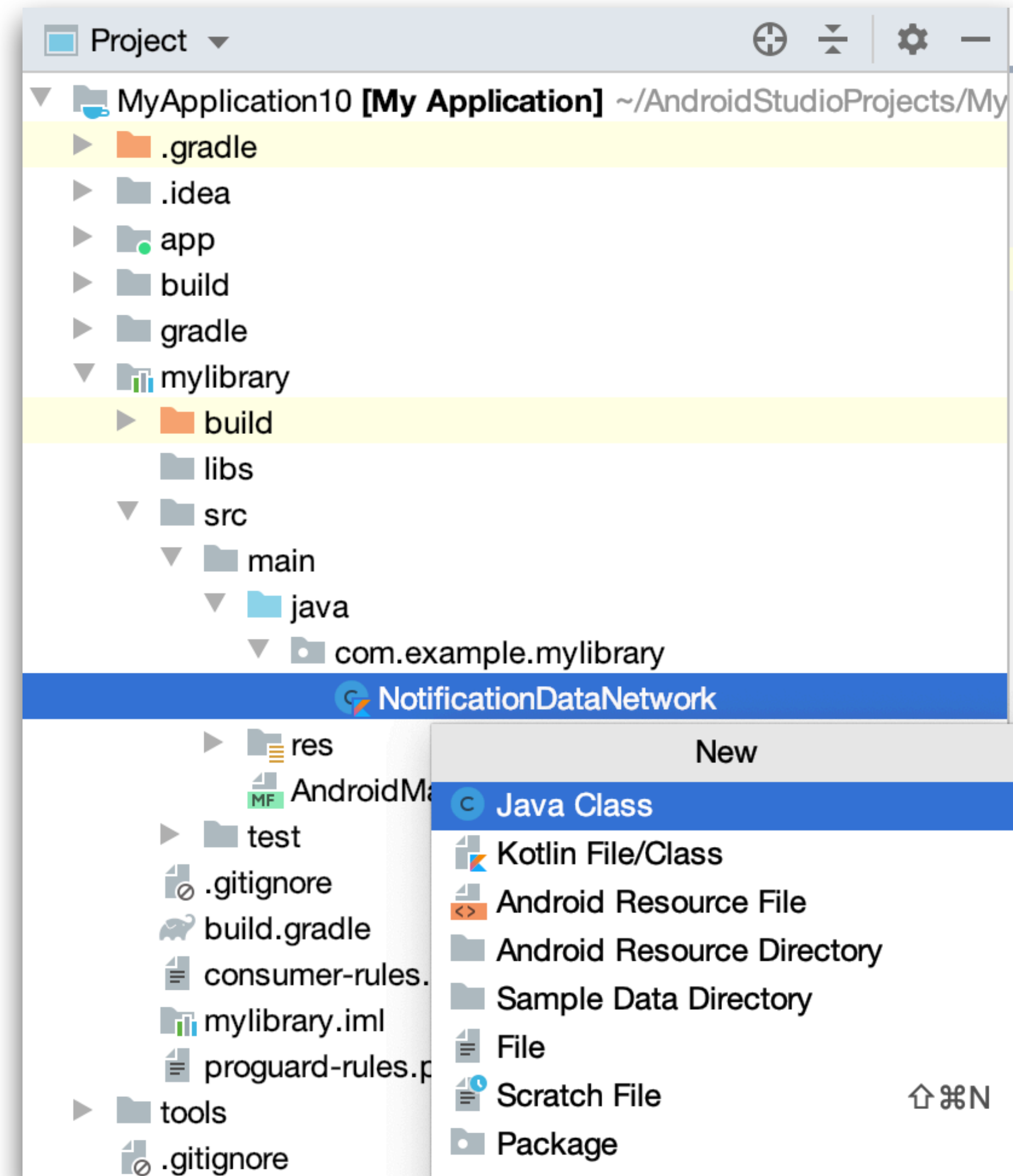
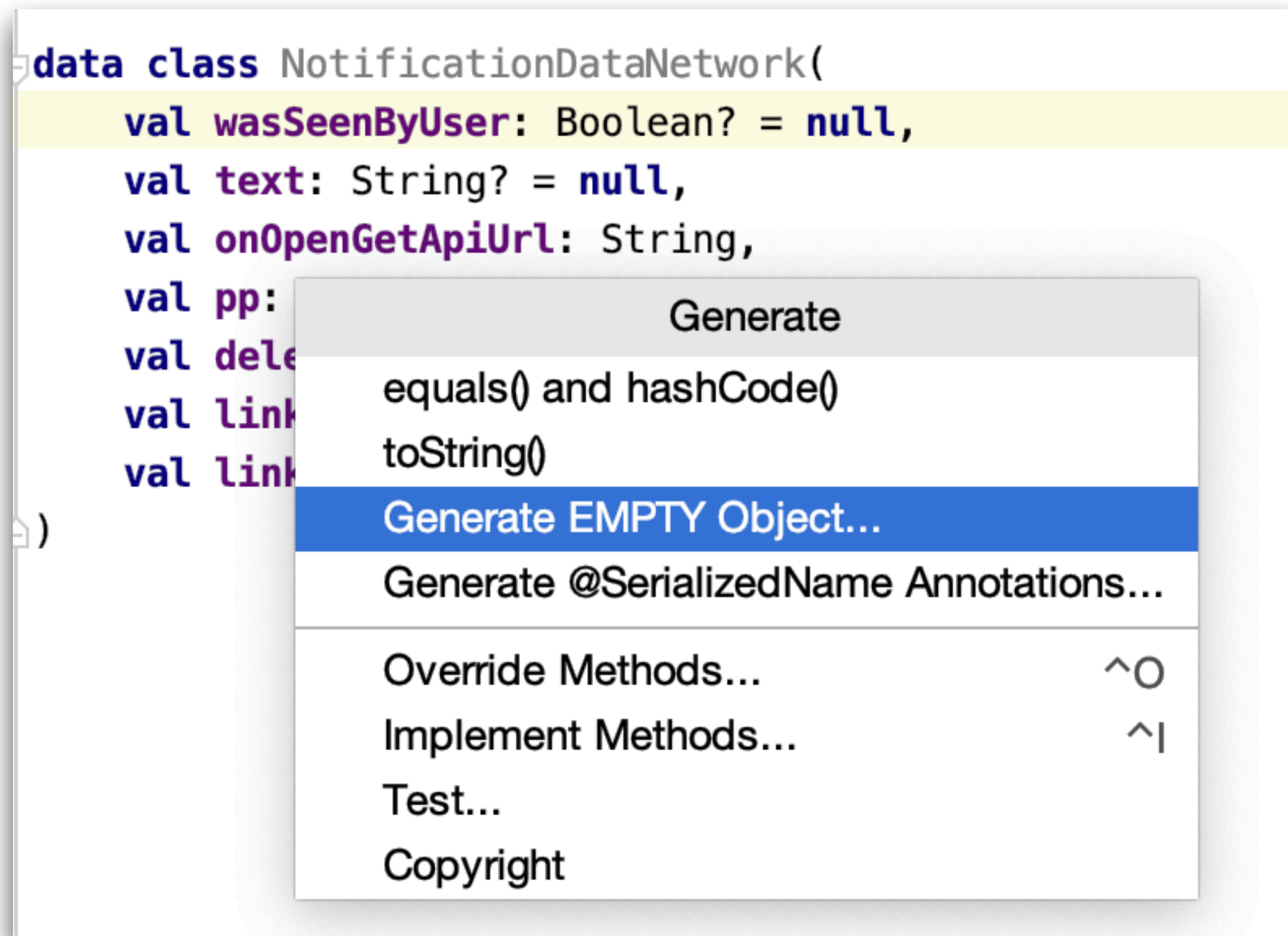
НО ЕСТЬ НЮАНСЫ

Что такое CodeInsightAction

```
data class FileTemplateData(  
    val templateFileName: String,  
    val outputFileName: String,  
    val outputFileType: FileType,  
    val outputFilePsiDirectory: PsiDirectory?  
)
```



Когда нужен CodeInsightAction



Формируем контекст для XML

```
class MyXmlAction : CodeInsightAction(), CodeInsightActionHandler {  
  
    override fun getHandler(): CodeInsightActionHandler = this  
    override fun startInWriteAction(): Boolean = false  
  
    override fun isValidForFile(  
        project: Project,  
        editor: Editor,  
        file: PsiFile  
    ): Boolean {  
        return file is XmlFile && file.parent?.name == "layout"  
    }  
  
    override fun invoke(project: Project, editor: Editor, file: PsiFile) { ... }  
  
}
```

Наследуем CodeInsightAction

```
class MyXmlAction : CodeInsightAction(), CodeInsightActionHandler {  
  
    override fun getHandler(): CodeInsightActionHandler = this  
    override fun startInWriteAction(): Boolean = false  
  
    override fun isValidForFile(  
        project: Project,  
        editor: Editor,  
        file: PsiFile  
    ): Boolean {  
        return file is XmlFile && file.parent?.name == "layout"  
    }  
  
    override fun invoke(project: Project, editor: Editor, file: PsiFile) { ... }  
  
}
```

Реализуем CodeInsightActionHandler

```
class MyXmlAction : CodeInsightAction(), CodeInsightActionHandler {  
  
    override fun getHandler(): CodeInsightActionHandler = this  
    override fun startInWriteAction(): Boolean = false  
  
    override fun isValidForFile(  
        project: Project,  
        editor: Editor,  
        file: PsiFile  
    ): Boolean {  
        return file is XmlFile && file.parent?.name == "layout"  
    }  
  
    override fun invoke(project: Project, editor: Editor, file: PsiFile) { ... }  
  
}
```

Обработка — не в режиме write

```
class MyXmlAction : CodeInsightAction(), CodeInsightActionHandler {  
  
    override fun getHandler(): CodeInsightActionHandler = this  
    override fun startInWriteAction(): Boolean = false  
  
    override fun isValidForFile(  
        project: Project,  
        editor: Editor,  
        file: PsiFile  
    ): Boolean {  
        return file is XmlFile && file.parent?.name == "layout"  
    }  
  
    override fun invoke(project: Project, editor: Editor, file: PsiFile) { ... }  
  
}
```

Определяем контекст res/layout

```
class MyXmlAction : CodeInsightAction(), CodeInsightActionHandler {  
  
    override fun getHandler(): CodeInsightActionHandler = this  
    override fun startInWriteAction(): Boolean = false  
  
    override fun isValidForFile(  
        project: Project,  
        editor: Editor,  
        file: PsiFile  
    ): Boolean {  
        return file is XmlFile && file.parent?.name == "layout"  
    }  
  
    override fun invoke(project: Project, editor: Editor, file: PsiFile) { ... }  
  
}
```

Для Kotlin-классов — всё проще

```
class GenerateEmptyObjectAction : KotlinGenerateActionBase() {  
  
    override fun isValidForClass(targetClass: KtClassOrObject): Boolean {  
        return targetClass is KtClass  
            && targetClass.isData()  
            && targetClass.companionObjects.firstOrNull()  
                ?.findPropertyByName("EMPTY") == null  
    }  
  
    override fun invoke(project: Project, editor: Editor, file: PsiFile) { ... }  
  
}
```

Наследуем KotlinGenerateActionBase

```
class GenerateEmptyObjectAction : KotlinGenerateActionBase() {  
  
    override fun isValidForClass(targetClass: KtClassOrObject): Boolean {  
        return targetClass is KtClass  
            && targetClass.isData()  
            && targetClass.companionObjects.firstOrNull()  
                ?.findPropertyByName("EMPTY") == null  
    }  
  
    override fun invoke(project: Project, editor: Editor, file: PsiFile) { ... }  
  
}
```

Контекст уже в Kotlin-классе!

```
class GenerateEmptyObjectAction : KotlinGenerateActionBase() {  
  
    override fun isValidForClass(targetClass: KtClassOrObject): Boolean {  
        return targetClass is KtClass  
            && targetClass.isData()  
            && targetClass.companionObjects.firstOrNull()  
                ?.findPropertyByName("EMPTY") == null  
    }  
  
    override fun invoke(project: Project, editor: Editor, file: PsiFile) { ... }  
  
}
```

Проверяем — data ли класс?

```
class GenerateEmptyObjectAction : KotlinGenerateActionBase() {  
  
    override fun isValidForClass(targetClass: KtClassOrObject): Boolean {  
        return targetClass is KtClass  
            && targetClass.isData()  
            && targetClass.companionObjects.firstOrNull()  
                ?.findPropertyByName("EMPTY") == null  
    }  
  
    override fun invoke(project: Project, editor: Editor, file: PsiFile) { ... }  
  
}
```

Есть ли уже EMPTY-объект?

```
class GenerateEmptyObjectAction : KotlinGenerateActionBase() {  
  
    override fun isValidForClass(targetClass: KtClassOrObject): Boolean {  
        return targetClass is KtClass  
            && targetClass.isData()  
            && targetClass.companionObjects.firstOrNull()  
                ?.findPropertyByName("EMPTY") == null  
    }  
  
    override fun invoke(project: Project, editor: Editor, file: PsiFile) { ... }  
  
}
```

Резюмируем



- 1 **AnActionEvent** содержит тонну данных о контексте
- 2 **CodeInsighthAction** упростит **Generate-action-ы**
- 3 Для Kotlin-а есть **KotlinGenerateActionBase**

Резюмируем

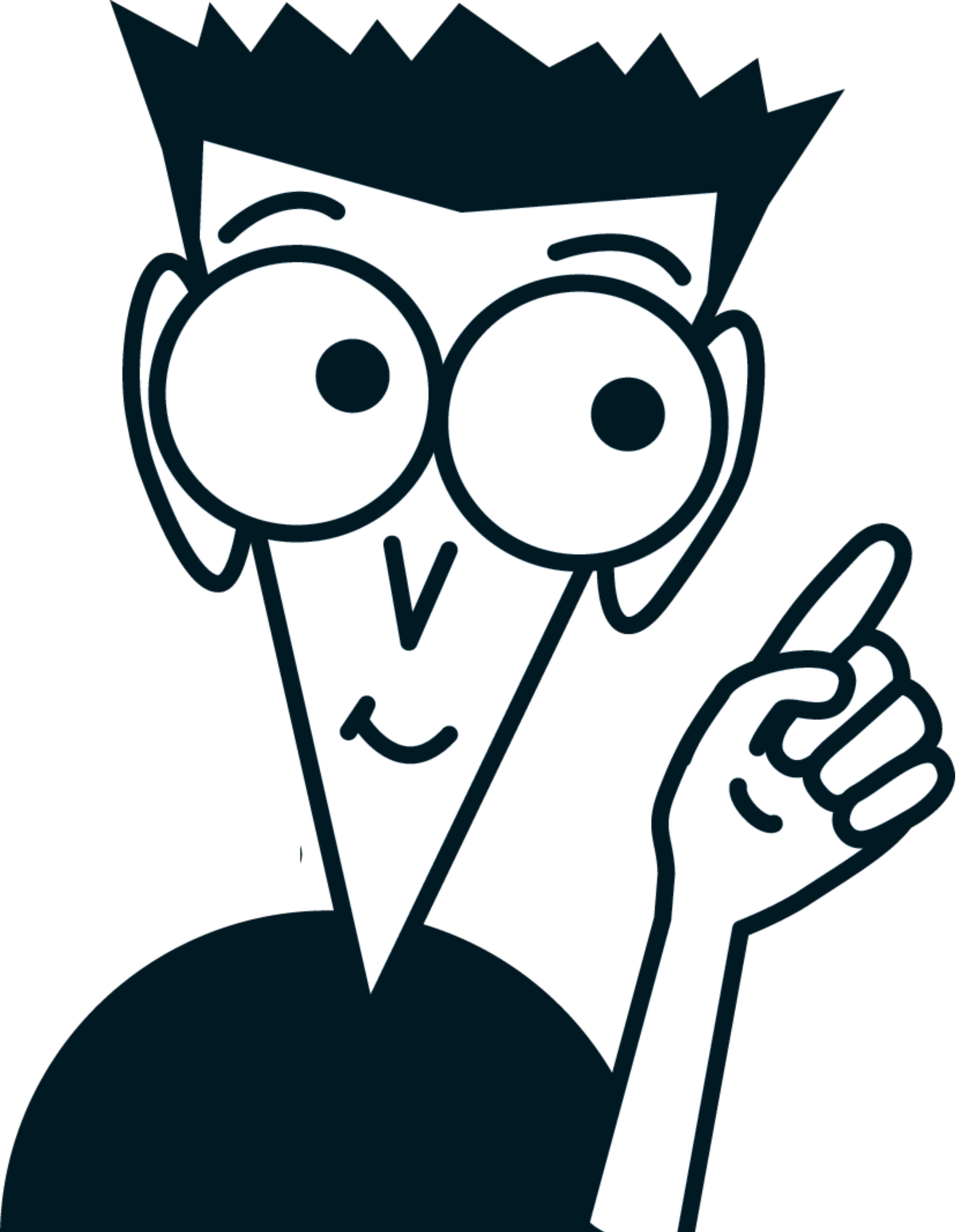


- 1 `AnActionEvent` содержит тонну данных о контексте
- 2 **`CodeInsightAction` упростит `Generate-action`-ы**
- 3 Для Kotlin-а есть `KotlinGenerateActionBase`

Резюмируем



- 1 `AnActionEvent` содержит тонну данных о контексте
- 2 `CodeInsightAction` упростит `Generate-action`-ы
- 3 Для Kotlin-а есть `KotlinGenerateActionBase`



Получение текста и создание PSI

Хорошая новость

Хорошая новость

Получайте текст как угодно

Навскидку несколько вариантов

- 1 Захардкодить нужный текст
- 2 Получить текст из шаблона FreeMarker
- 3 Сформировать текст на основе вашего контекста

Разберём самый интересный

- 1 Захардкодить нужный текст
- 2 Получить текст из шаблона FreeMarker

3 Сформировать текст на основе вашего контекста

Получаем текст EMPTY-объекта

```
private fun KtClass.toEmptyObjectDeclaration(): String {
    val emptyParametersText = primaryConstructorParameters.joinToString(
        prefix = "\n", postfix = "\n", separator = "\n"
    ) { parameter →
        "${parameter.name} = ${parameter.getEmptyValueText()}"
    }

    return """
    val EMPTY = ${this.name}($emptyParametersText)
    """
}
```

Обходим параметры конструктора

```
private fun KtClass.toEmptyObjectDeclaration(): String {  
    val emptyParametersText = primaryConstructorParameters.joinToString(  
        prefix = "\n", postfix = "\n", separator = "\n"  
    ) { parameter →  
        "${parameter.name} = ${parameter.getEmptyValueText()}"  
    }  
  
    return ""  
    val EMPTY = "${this.name}($emptyParametersText)"  
    ""  
}
```

Формируем текст из параметра

```
private fun KtClass.toEmptyObjectDeclaration(): String {  
    val emptyParametersText = primaryConstructorParameters.joinToString(  
        prefix = "\n", postfix = "\n", separator = "\n"  
    ) { parameter →  
        "${parameter.name} = ${parameter.getEmptyValueText()}"  
    }  
  
    return ""  
    val EMPTY = ${this.name}($emptyParametersText)  
    ""  
}
```

Получаем текст из параметров

```
private fun KtParameter.getEmptyValueText(): String {  
    val type = this.type()  
  
    return when {  
        type == null || type.isMarkedNullable → "null"  
        KotlinBuiltIns.isBoolean(type) → "false"  
        KotlinBuiltIns.isChar(type) → ""  
        KotlinBuiltIns.isString(type) → "String.EMPTY"  
        KotlinBuiltIns.isListOrNullableList(type) → "emptyList()"  
        ...  
        else → "$type.EMPTY"  
    }  
}
```

Текст EMPTY-объекта готов

```
private fun KtClass.toEmptyObjectDeclaration(): String {  
    val emptyParametersText = primaryConstructorParameters.joinToString(  
        prefix = "\n", postfix = "\n", separator = "\n"  
    ) { parameter →  
        "${parameter.name} = ${parameter.getEmptyValueText()}"  
    }  
  
    return ""  
    val EMPTY = ${this.name}($emptyParametersText)  
    ""  
}
```

Для создания PSI нужна фабрика



`JavaPsiFacade.getElementsFactory(project)`



`GroovyPsiElementFactory.getInstance(project)`

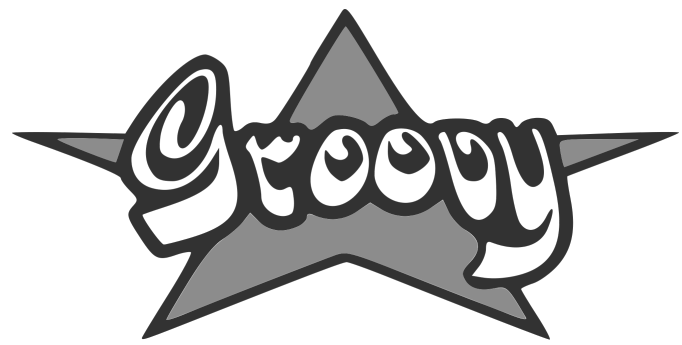


`KtPsiFactory(project)`

Нам нужна KtPsiFactory



`JavaPsiFacade.getElementsFactory(project)`



`GroovyPsiElementFactory.getInstance(project)`



`KtPsiFactory(project)`

KtPsiFactory создаст property

```
private fun createEmptyObjectProperty(ktClass: KtClass): KtProperty {  
    val emptyObjectText = ktClass.toEmptyObjectDeclaration()  
  
    val ktPsiFactory = KtPsiFactory(ktClass.project)  
    return ktPsiFactory.createProperty(emptyObjectText)  
}
```

Резюмируем



- 1 Текст кода можно получить любым способом
- 2 Для формирования PSI-элементов есть фабрики

Резюмируем

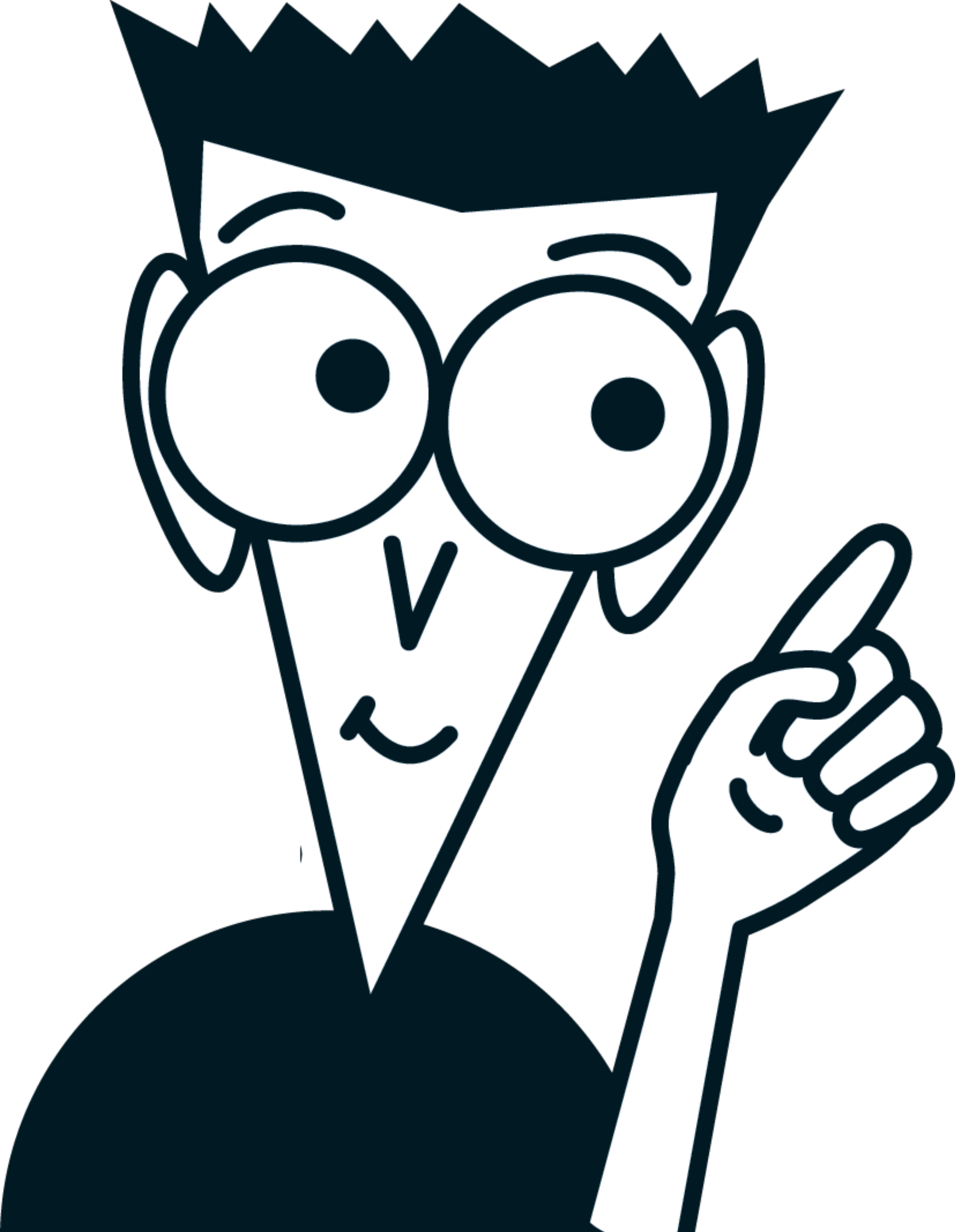


1

Текст кода можно получить любым способом

2

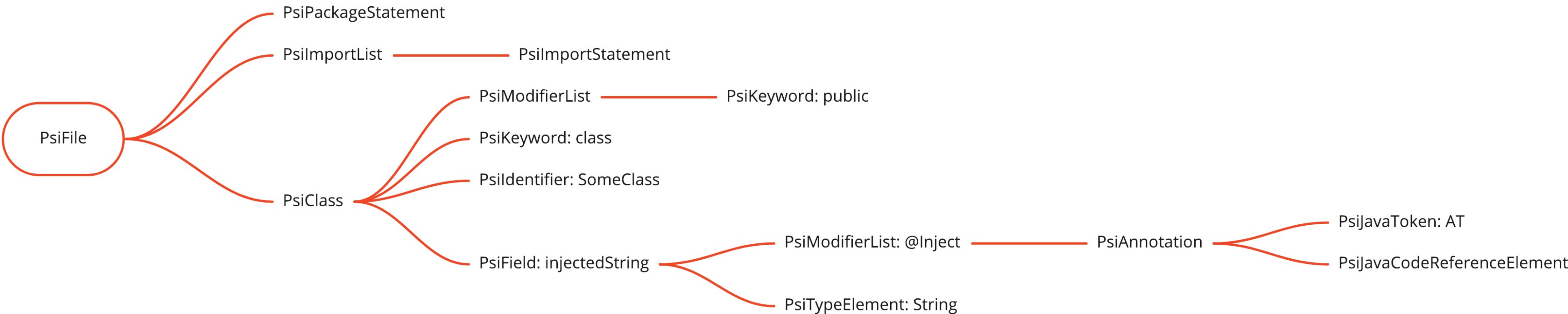
Для формирования PSI-элементов есть фабрики



**Добавляем код в
PSI-структуру**

Немного о PSI

Иерархия PsiFile



Интерфейс модификации PSI

```
public interface PsiElement {  
  
    ...  
  
    PsiElement add(@NotNull PsiElement element);  
  
    PsiElement addBefore(@NotNull PsiElement element, @Nullable PsiElement anchor);  
  
    PsiElement addAfter(@NotNull PsiElement element, @Nullable PsiElement anchor);  
  
}
```

Child PsiElement — В КОНЕЦ

```
public interface PsiElement {  
  
    ...  
  
    PsiElement add(@NotNull PsiElement element);  
  
    PsiElement addBefore(@NotNull PsiElement element, @Nullable PsiElement anchor);  
  
    PsiElement addAfter(@NotNull PsiElement element, @Nullable PsiElement anchor);  
  
}
```

Child PsiElement — перед anchor

```
public interface PsiElement {  
  
    ...  
  
    PsiElement add(@NotNull PsiElement element);  
  
    PsiElement addBefore(@NotNull PsiElement element, @Nullable PsiElement anchor);  
  
    PsiElement addAfter(@NotNull PsiElement element, @Nullable PsiElement anchor);  
  
}
```

Child PsiElement — после anchor

```
public interface PsiElement {  
  
    ...  
  
    PsiElement add(@NotNull PsiElement element);  
  
    PsiElement addBefore(@NotNull PsiElement element, @Nullable PsiElement anchor);  
  
    PsiElement addAfter(@NotNull PsiElement element, @Nullable PsiElement anchor);  
  
}
```

Важно! Если не учесть структуру PSI,
плагин будет работать не так, как вы хотите



Важно! Если не учесть структуру PSI,
плагин будет работать не так, как вы хотите



Пример с @SerializedName

```
1 package com.example.mylibrary
2
3
4 data class NotificationDataNetwork(
5     val wasSeenByUser: Boolean? = null,
6     val text: String? = null,
7     val onOpenGetApiUrl: String? = null,
8     val deleteApiUrl: String? = null,
9     val linkUrl: String? = null,
10    val linkMode: String? = null
11 )
```

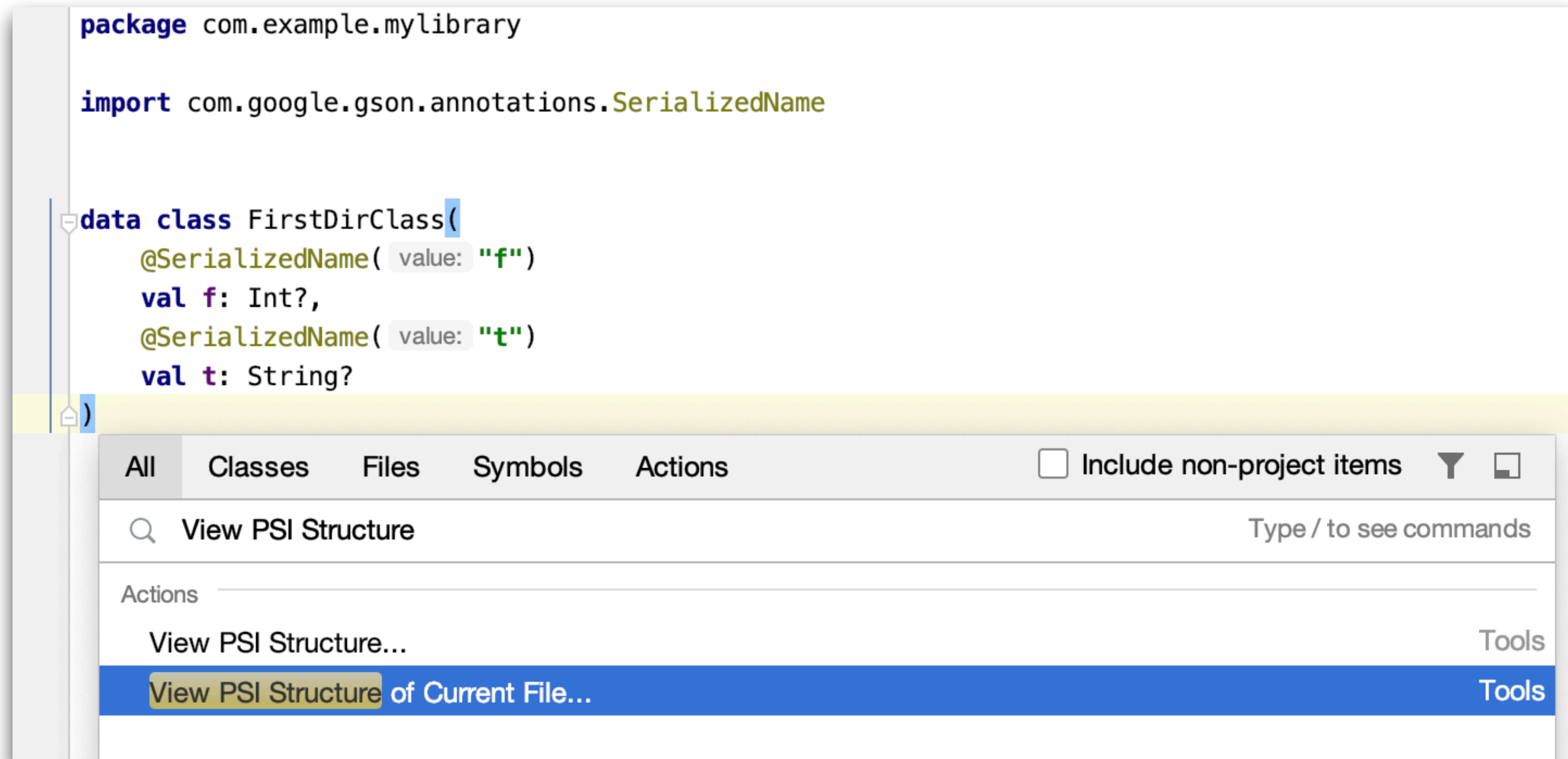


Пример с @SerializedName

```
1 package com.example.mylibrary
2
3
4 data class NotificationDataNetwork(
5     val wasSeenByUser: Boolean? = null,
6     val text: String? = null,
7     val onOpenGetApiUrl: String? = null,
8     val deleteApiUrl: String? = null,
9     val linkUrl: String? = null,
10    val linkMode: String? = null
11 )
```



Смотрим PSI структуру



Смотрим PSI структуру

The screenshot shows the 'PSI Viewer: FirstDirClass.kt' window. At the top, there are controls for 'Show PSI structure for:' (set to 'Kot...'), checkboxes for 'Show PsiWhiteSpace' and 'Show tree nodes' (both checked), and an 'Extension:' dropdown set to '.kt'. Below this is a 'Text' tab showing the source code of 'FirstDirClass.kt' with line numbers 1 through 11. The code defines a package, an import, and a data class. The data class definition is highlighted with a yellow background and an orange border. Below the code is a 'PSI Tree' tab with sub-tabs for 'References', 'Block Structure', and 'Stub Structure'. The 'PSI Tree' view shows a tree structure starting with 'PsiWhiteSpace', followed by a collapsed 'CLASS' node, which contains 'MODIFIER_LIST' and 'PsiWhiteSpace'.

PSI Viewer: FirstDirClass.kt

Show PSI structure for: Kot... ☒ Show PsiWhiteSpace ☒ Show tree nodes Extension: .kt

Text

```
1 package com.example.mylibrary
2
3 import com.google.gson.annotations.SerializedName
4
5
6 data class FirstDirClass(
7     @SerializedName(value: "f")
8     val f: Int?,
9     @SerializedName(value: "t")
10    val t: String?
11 )
```

PSI Tree

- ▶ PsiWhiteSpace
- ▼ CLASS
 - ▶ MODIFIER_LIST
 - PsiWhiteSpace

References Block Structure Stub Structure

Nothing to show

Смотрим PSI структуру

PSI Viewer: FirstDirClass.kt

Show PSI structure for: Kot... ☒ Show PsiWhiteSpace ☒ Show tree nodes Extension: .kt

Text

```
1 package com.example.mylibrary
2
3 import com.google.gson.annotations.SerializedName
4
5
6 data class FirstDirClass(
7     @SerializedName(value: "f")
8     val f: Int?,
9     @SerializedName(value: "t")
10    val t: String?
11 )
```

PSI Tree

References

- ▶ PsiWhiteSpace
- ▼ CLASS
 - ▶ MODIFIER_LIST
 - PsiWhiteSpace
 - PsiElement(class)
 - PsiWhiteSpace
 - PsiElement(IDENTIFIER)
 - ▼ PRIMARY_CONSTRUCTOR
 - ▼ VALUE_PARAMETER_LIST
 - PsiElement(LPAR)
 - PsiWhiteSpace

Copy PSI

PSI Viewer: FirstDirClass.kt

Show PSI structure for: Kot... ☒ Show PsiWhiteSpace ☒ Show tree nodes Extension: .kt

Text

```
1 package com.example.mylibrary
2
3 import com.google.gson.annotations.SerializedName
4
5
6 data class FirstDirClass(
7     @SerializedName(value: "f")
8     val f: Int?,
9     @SerializedName(value: "t")
10    val t: String?
11 )
```

PSI Tree

References

Block Structure

Stub Structure

- ▼ PRIMARY_CONSTRUCTOR
 - ▼ VALUE_PARAMETER_LIST
 - PsiElement(LPAR)
 - PsiWhiteSpace
 - ▼ VALUE_PARAMETER
 - ▼ MODIFIER_LIST
 - ▼ ANNOTATION_ENTRY
 - PsiElement(AT)
 - ▼ CONSTRUCTOR_CALLEE
 - ▼ TYPE_REFERENCE
 - ▼ USER_TYPE

Nothing to show

Copy PSI

Close

Build PSI Tree

Добавляем аннотации в KtClass

```
private fun addSerializedNameAnnotations(ktClass: KtClass) {  
    val ktPsiFactory = KtPsiFactory(ktClass.project)  
  
    ktClass.project.executeWriteCommand("SerializedNameService") {  
        ktClass.primaryConstructorParameters.forEach { parameter →  
            val text = "@${SERIALIZED_NAME_FQN}(\\"${parameter.name!!.toSnake()}\\")"  
            val annotationEntry = ktPsiFactory.createAnnotationEntry(text)  
            val breakLineElement = ktPsiFactory.createWhiteSpace("\\n")  
  
            ktClass.getValueParametersList()?.let { valueParameterList →  
                valueParameterList.addBefore(annotationEntry, parameter)  
                valueParameterList.addBefore(breakLineElement, parameter)  
            }  
        }  
    }  
}
```

ГОТОВИМСЯ К МОДИФИКАЦИИ КОДА

```
private fun addSerializedNameAnnotations(ktClass: KtClass) {
    val ktPsiFactory = KtPsiFactory(ktClass.project)

    ktClass.project.executeWriteCommand("SerializedNameService") {
        ktClass.primaryConstructorParameters.forEach { parameter →
            val text = "@${SERIALIZED_NAME_FQN}("${parameter.name!!.toSnake()}")"
            val annotationEntry = ktPsiFactory.createAnnotationEntry(text)
            val breakLineElement = ktPsiFactory.createWhiteSpace("\n")

            ktClass.getValueParametersList()?.let { valueParameterList →
                valueParameterList.addBefore(annotationEntry, parameter)
                valueParameterList.addBefore(breakLineElement, parameter)
            }
        }
    }
}
```

Проходим по каждому параметру

```
private fun addSerializedNameAnnotations(ktClass: KtClass) {  
    val ktPsiFactory = KtPsiFactory(ktClass.project)  
  
    ktClass.project.executeWriteCommand("SerializedNameService") {  
        ktClass.primaryConstructorParameters.forEach { parameter →  
            val text = "@${SERIALIZED_NAME_FQN}("${parameter.name!!.toSnake()}")"  
            val annotationEntry = ktPsiFactory.createAnnotationEntry(text)  
            val breakLineElement = ktPsiFactory.createWhiteSpace("\n")  
  
            ktClass.getValueParametersList()?.let { valueParameterList →  
                valueParameterList.addBefore(annotationEntry, parameter)  
                valueParameterList.addBefore(breakLineElement, parameter)  
            }  
        }  
    }  
}
```

Формируем текст и PSI

```
private fun addSerializedNameAnnotations(ktClass: KtClass) {
    val ktPsiFactory = KtPsiFactory(ktClass.project)

    ktClass.project.executeWriteCommand("SerializedNameService") {
        ktClass.primaryConstructorParameters.forEach { parameter →
            val text = "@${SERIALIZED_NAME_FQN}("${parameter.name!!.toSnake()}")"
            val annotationEntry = ktPsiFactory.createAnnotationEntry(text)
            val breakLineElement = ktPsiFactory.createWhiteSpace("\n")

            ktClass.getValueParametersList()?.let { valueParameterList →
                valueParameterList.addBefore(annotationEntry, parameter)
                valueParameterList.addBefore(breakLineElement, parameter)
            }
        }
    }
}
```

Модифицируем код

```
private fun addSerializedNameAnnotations(ktClass: KtClass) {  
    val ktPsiFactory = KtPsiFactory(ktClass.project)  
  
    ktClass.project.executeWriteCommand("SerializedNameService") {  
        ktClass.primaryConstructorParameters.forEach { parameter →  
            val text = "@${SERIALIZED_NAME_FQN}("${parameter.name!!.toSnake()}")"  
            val annotationEntry = ktPsiFactory.createAnnotationEntry(text)  
            val breakLineElement = ktPsiFactory.createWhiteSpace("\n")  
  
            ktClass.getValueParametersList()?.let { valueParameterList →  
                valueParameterList.addBefore(annotationEntry, parameter)  
                valueParameterList.addBefore(breakLineElement, parameter)  
            }  
        }  
    }  
}
```

Последние штрихи

```
private fun addSerializedNameAnnotations(ktClass: KtClass) {  
    val ktPsiFactory = KtPsiFactory(ktClass.project)  
  
    ktClass.project.executeWriteCommand("SerializedNameService") {  
        ktClass.primaryConstructorParameters.forEach { parameter →  
            val text = "@${SERIALIZED_NAME_FQN}(\\"${parameter.name!!.toSnake()}\\")"  
            ...  
        }  
  
        ktClass.containingKtFile.commitAndUnblockDocument()  
        ShortenReference.DEFAULT.process(ktClass)  
        CodeStyleManager.getInstance(ktClass.project).reformat(ktClass.containingKtFile)  
    }  
}
```

Добавляем импорт аннотации

```
private fun addSerializedNameAnnotations(ktClass: KtClass) {  
    val ktPsiFactory = KtPsiFactory(ktClass.project)  
  
    ktClass.project.executeWriteCommand("SerializedNameService") {  
        ktClass.primaryConstructorParameters.forEach { parameter →  
            val text = "@${SERIALIZED_NAME_FQN}(\\"${parameter.name!!.toSnake()}\\")"  
            ...  
        }  
  
        ktClass.containingKtFile.commitAndUnblockDocument()  
        ShortenReference.DEFAULT.process(ktClass)  
        CodeStyleManager.getInstance(ktClass.project).reformat(ktClass.containingKtFile)  
    }  
}
```

Правим стиль кода

```
private fun addSerializedNameAnnotations(ktClass: KtClass) {  
    val ktPsiFactory = KtPsiFactory(ktClass.project)  
  
    ktClass.project.executeWriteCommand("SerializedNameService") {  
        ktClass.primaryConstructorParameters.forEach { parameter →  
            val text = "@${SERIALIZED_NAME_FQN}("${parameter.name!!.toSnake()}")"  
            ...  
        }  
  
        ktClass.containingKtFile.commitAndUnblockDocument()  
        ShortenReference.DEFAULT.process(ktClass)  
        CodeStyleManager.getInstance(ktClass.project).reformat(ktClass.containingKtFile)  
    }  
}
```

Подведем итоги



- 1 Генерировать код - не очень сложно**
- 2 Сначала контекст, потом текст, затем PSI
- 3 CodeInsightAction упрощает работу с контекстом
- 4 Kotlin-классы модифицировать реально удобно

Подведем итоги



- 1 Генерировать код - не очень сложно
- 2 Сначала контекст, потом текст, затем PSI**
- 3 CodeInsightAction упрощает работу с контекстом
- 4 Kotlin-классы модифицировать реально удобно

Подведем итоги



- 1 Генерировать код - не очень сложно
- 2 Сначала контекст, потом текст, затем PSI
- 3 CodeInsightAction упрощает работу с контекстом**
- 4 Kotlin-классы модифицировать реально удобно

Подведем итоги



- 1 Генерировать код - не очень сложно
- 2 Сначала контекст, потом текст, затем PSI
- 3 CodeInsightAction упрощает работу с контекстом
- 4 Kotlin-классы модифицировать реально удобно**

Есть вопросы?

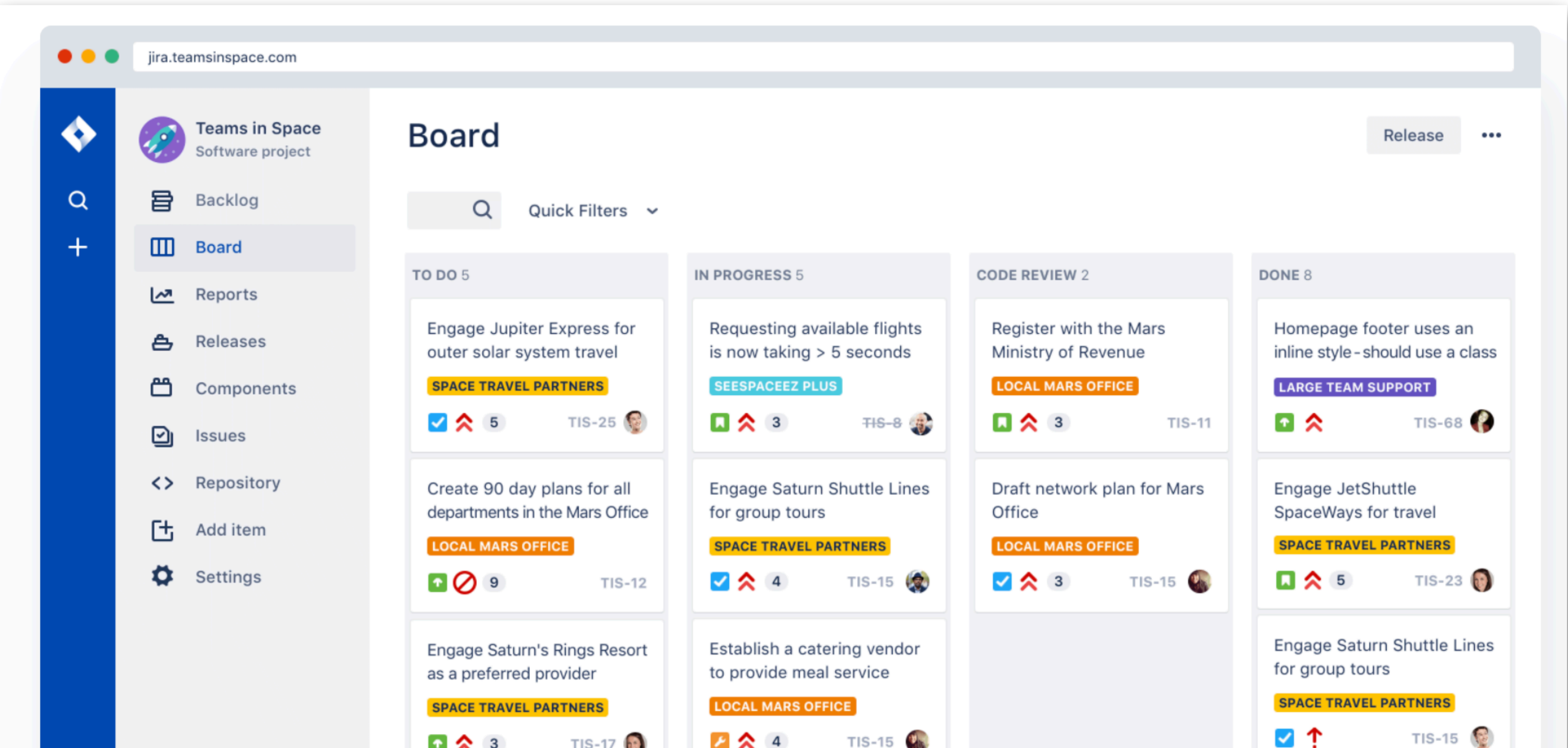


- 1 Генерировать код - не очень сложно
- 2 Сначала контекст, потом текст, затем PSI
- 3 CodeInsightAction упрощает работу с контекстом
- 4 Kotlin-классы модифицировать реально удобно

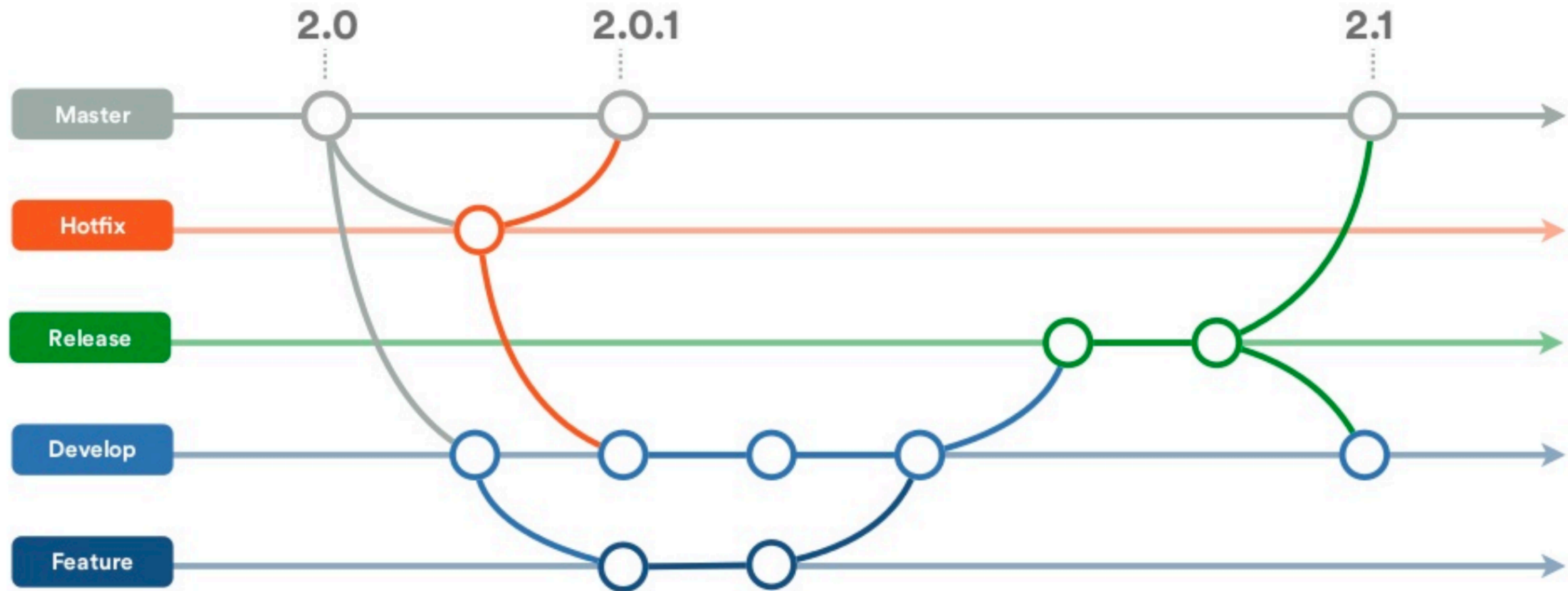


Автоматизация **работы с JIRA**

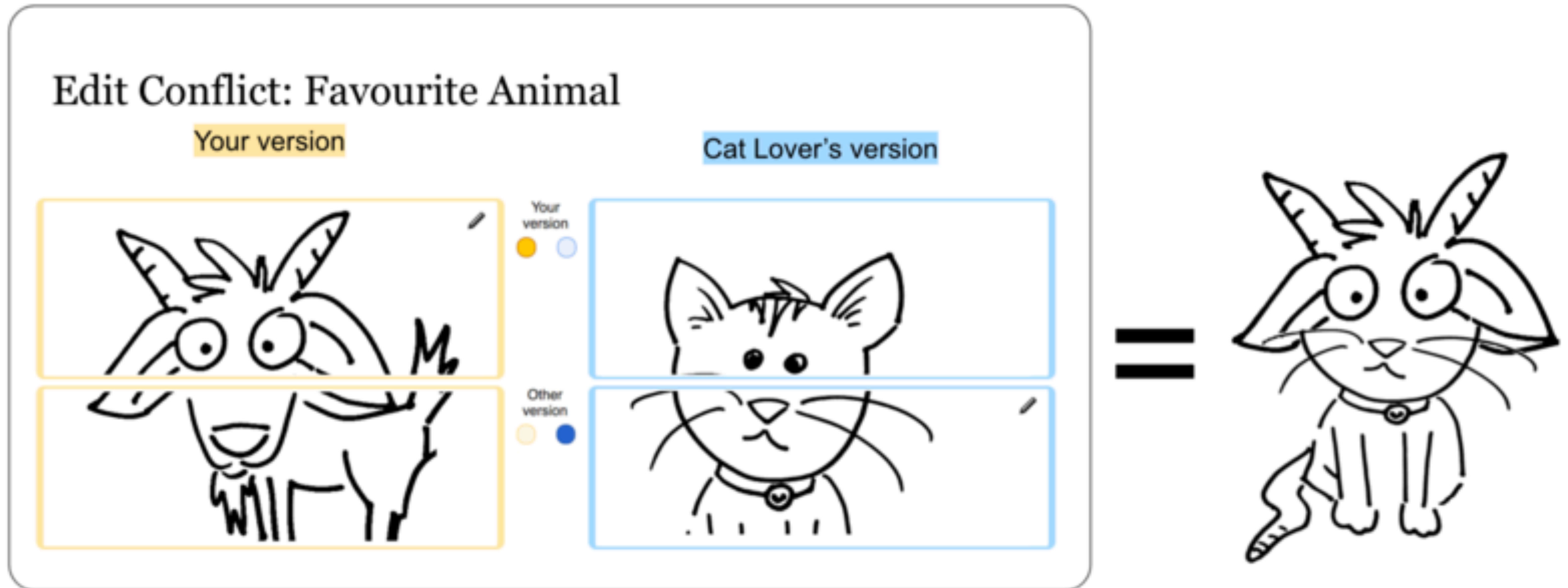
Мы ведем задачи в Jira...



... и работаем по GitFlow



... И вынуждены мерджить develop



... И заводите задачи в JIRA



R&D :: Мобильные приложения / MOB-13429

Merge develop to PORTFOLIO-9217

✎ Редактировать

💬 Комментарий

📌 Назначить

⋮ Еще...

🔄 Открыть запрос заново

▼ Детали запроса

Тип:

☒ Задача

Статус:

CLOSED (Посмотреть Workflow)

Приоритет:

🔼 Средний

Резолюция:

Won't Fix

Проявляется в версиях:

Нет

Исправить в версиях:

Нет

Метки:

ANDROID

ANDROID_APP

Development Team:

mobile-core

Story Points:

0,5

Rank (Obsolete):

9223372036854775807

Epic Link:

Android applicant

▼ Описание

Очередной merge develop-ветки в портфельную ветку [PORTFOLIO-9217](#)

110

... И ЗАВОДИТЬ ЗАДАЧИ В JIRA



R&D :: Мобильные приложения / MOB-13429

Merge develop to PORTFOLIO-9217

✎ Редактировать

💬 Комментарий

📌 Назначить

🔗 Epic

▼ Детали запроса

Тип:

☒ Задача

☐ Bug

Приоритет:

🔼 Средний

Проявляется в версиях:

Нет

Метки:

ANDROID

Development Team:

mobile-core

Story Points:

0,5

Rank (Obsolete):

9223372036854775807

Epic Link:

Android applicant

🔴 CLOSED (Посмотреть Workflow)

Won't Fix

Нет

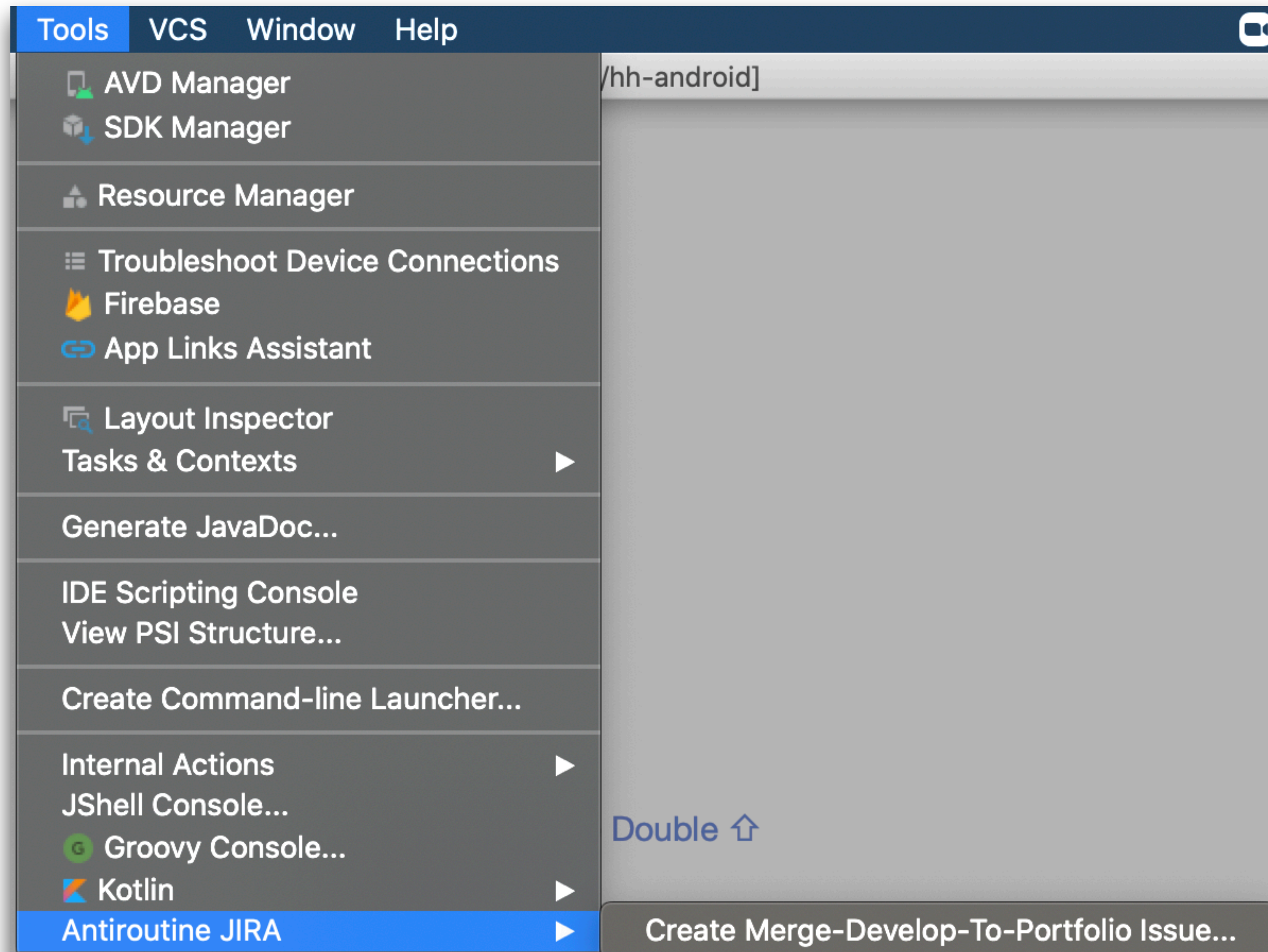
🔍 Исправить в версиях:

▼ Описание

Очередной merge develop-ветки в портфельную ветку [PORTFOLIO-9217](#)

Надоеело

Что можно сделать плагином?



Декомпозируем задачу

1

Ввод и хранение пользовательских credentials для Jira

2

Работа с git-ом — узнать текущий branch, создать новый

3

Работа с API Jira

Сокращаем рассказ ;)

1

Ввод и ~~хранение~~ пользовательских credentials для Jira

2

Работа с git-ом — узнать текущий branch, создать новый

3

~~Работа с API Jira~~

<https://developer.atlassian.com/server/jira/platform/java-apis/>

https://www.jetbrains.org/intellij/sdk/docs/basics/persisting_sensitive_data.html

Основные моменты реализации

1 Ввод пользовательских credentials для Jira

2 Взаимодействие с плагином git-a

Основные моменты реализации

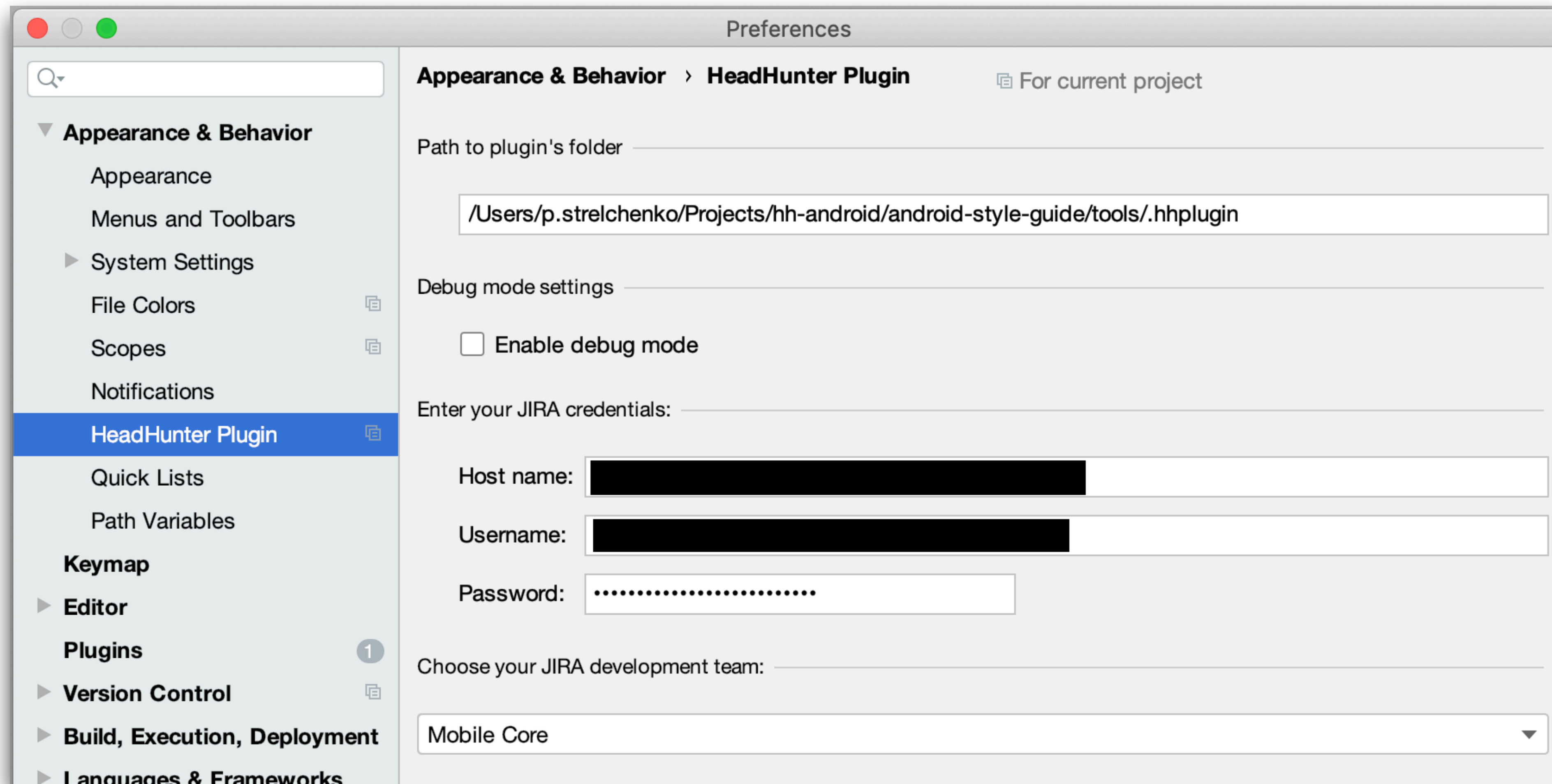
1 Ввод пользовательских credentials для Jira

2 Взаимодействие с плагином git-a

Credentials

Git plugin

Панель для ввода credentials



Credentials

Git plugin

Напишем кусочек UI для ввода

```
@Suppress("UnstableApiUsage")
fun createComponent(): JPanel {
    return panel {
        ...
        titledRow("Enter your JIRA credentials: ") {
            row("Username: ") {
                jiraUsernameTextField = JTextField()
                jiraUsernameTextField()
            }
            row("Password: ") {
                jiraPasswordTextField = JPasswordField()
                jiraPasswordTextField()
            }
        }
    }
}
```

Credentials

Git plugin

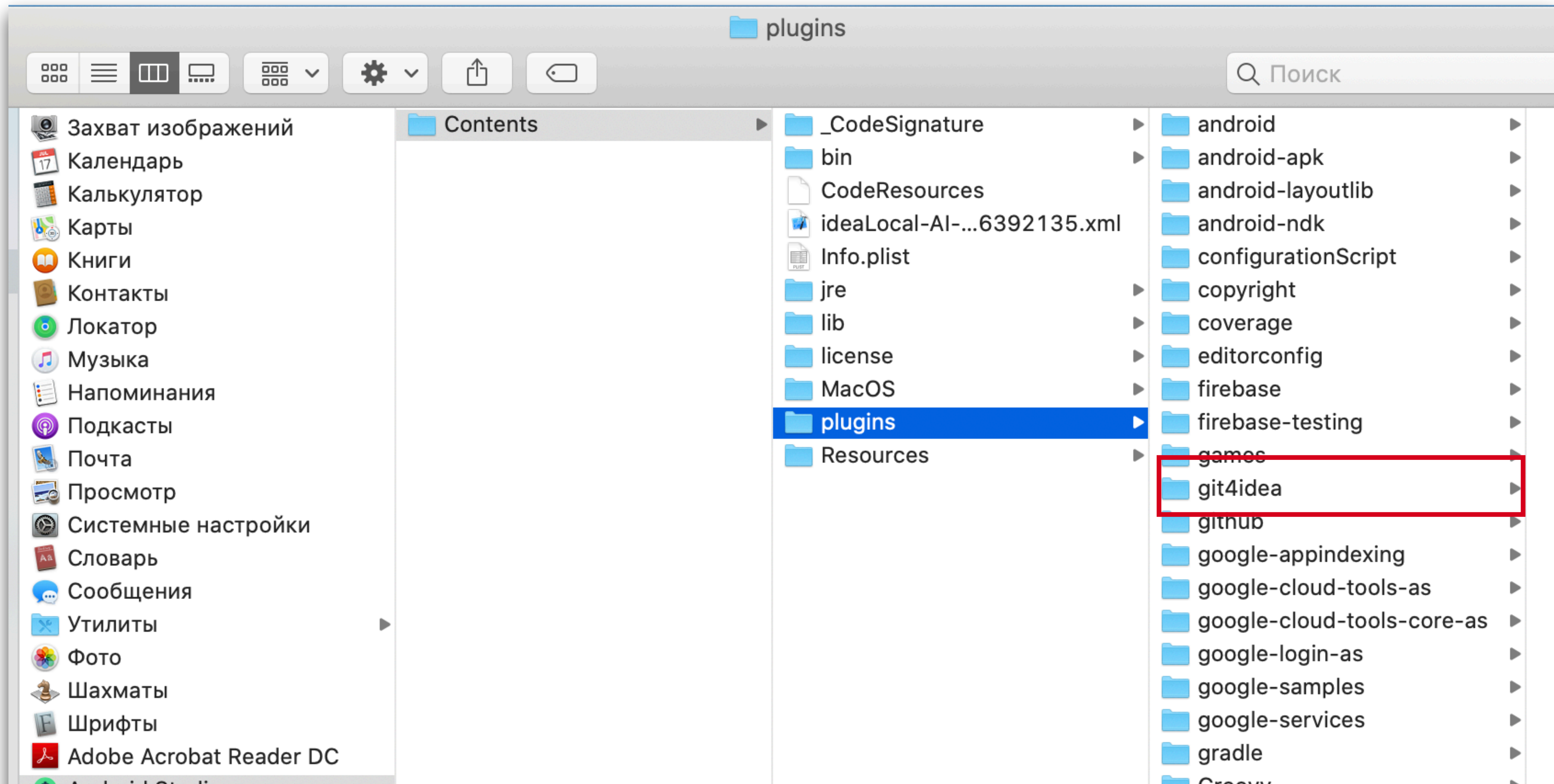
Подключаем плагин Git-a

```
intellij.type = 'IC'  
intellij.localPath = "/Applications/Android Studio.app"  
// Use this only for documentation  
//intellij.version = currentIdeVersion  
intellij.plugins = ["android", "Kotlin", "Groovy", "java", "git4idea"]
```

Подключаем плагин Git-a

```
intellij.type = 'IC'  
intellij.localPath = "/Applications/Android Studio.app"  
// Use this only for documentation  
//intellij.version = currentIdeVersion  
intellij.plugins = ["android", "Kotlin", "Groovy", "java", "git4idea"]
```

Откуда брать имена плагинов?..



Объявляем зависимость в .xml

```
<idea-plugin>  
  ...  
  <depends>Git4Idea</depends>  
  ...  
</idea-plugin>
```

Получаем название текущей ветки

```
@Service
class GitService(val project: Project) {

    ...

    fun getCurrentBranchName(): String {
        return GitRepositoryManager.getInstance(project).repositories
            .find { it.presentableUrl.endsWith("hh-android") }
            ?.currentBranch
            ?.name
    }
}
```

Создаем новую ветку

```
@Service
class GitService(val project: Project) {

    fun checkoutNewBranch(name: String) {
        GitRepositoryManager.getInstance(project).repositories
            .find { it.presentableUrl.endsWith("hh-android") }
            ?.let { myRepository →
                GitBrancher.getInstance(project).checkoutNewBranch(name)
            }
    }
}
```

Подведем итоги



1

Автоматизировать можно что угодно

2

Теперь разрабатывать UI в плагинах проще

3

Помните, что можно использовать чужие плагины

Подведем итоги



- 1 Автоматизировать можно что угодно
- 2 Теперь разрабатывать UI в плагинах проще**
- 3 Помните, что можно использовать чужие плагины

Подведем итоги



1

Автоматизировать можно что угодно

2

Теперь разрабатывать UI в плагинах проще

3

Помните, что можно использовать чужие плагины

Есть вопросы?



- 1 Автоматизировать можно что угодно
- 2 Теперь разрабатывать UI в плагинах проще
- 3 Помните, что можно использовать чужие плагины



Добавление инспекций кода



**Зачем это
моей команде?**

Например — для проверки @id/

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_lint_check_container_root"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
```

```
<ImageView
```

```
    android:id="@+id/some_wrong_id_declaration"
```

```
    android:layout_width="wrap_content"
```

Wrong id declaration! Should have prefix fragment_lint_check_image

Replace current id with `fragment\_lint\_check\_image` prefix ↵ ↶ ↷ More actions... ↵ ↶

```
<EditText
```

```
    android:id="@+id/sds"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"/>
```

```
<ProgressBar
```

```
    android:id="@+id/assaa"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"/>
```

Можно даже QuickFix показать

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@+id/fragment_lint_check_container_root"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical">
```

```
<ImageView
```

```
    android:id="@+id/some_wrong_id_declaration"
```

- 💡 Replace current id with `fragment\_lint\_check\_image` prefix
- 💡 Provide feedback on this warning
- ✕ Suppress: Add tools:ignore="WrongViewIdCodeStyleFormatting" attribute

```
<Edit
```

- ✎ Override Resource in Other Configuration... ▶
- ✎ Rearrange tag attributes ▶
- ✎ Remove attribute ▶
- ✎ Reset to default namespace ▶

```
<ProgressBar
```

```
    android:id="@+id/assaa"
```

Или для проверки putSerializable

```
class MyFragment : Fragment() {  
    companion object {  
        private const val ARGS_DATA HOLDER = "args_data_holder"  
  
        fun newInstance(dataHolder: TestDataHolder): MyFragment {  
            return MyFragment().apply { this: MyFragment  
                val bundle = Bundle()  
                bundle.putSerializable(ARGS_DATA HOLDER, dataHolder)  
            }  
        }  
    }  
}
```

Argument is not fully Serializable! Check it and its inner fields! [First not serializable class name: TestDataHolder.TestDescription.TestBoard]

[Provide feedback on this warning](#)

[More actions...](#)

Пока ты спал — нам дали tutorial

Code Inspections



The IntelliJ Platform provides tools designed for static code analysis called *code inspections*, which help the user maintain and clean up code without actually executing it. Custom code inspections can be implemented as IntelliJ Platform plugins. Examples of the plugin approach are the IntelliJ Platform SDK code samples [inspection_basics](#) and [comparing_references_inspection](#). In addition, the `comparing_references_inspection` code sample demonstrates implementing a unit test.

You can also create custom inspections through the IntelliJ IDEA user interface. See [Code Inspection](#) and [Creating Custom Inspections](#) for more information.

Creating an Inspection Plugin

The `comparing_references_inspection` code sample adds a new inspection to the **Java | Probable Bugs** group in the [Inspections list](#). The inspection reports when the `==` or `!=` operator is used between Java expressions of reference types.

It illustrates the components for a custom inspection plugin:

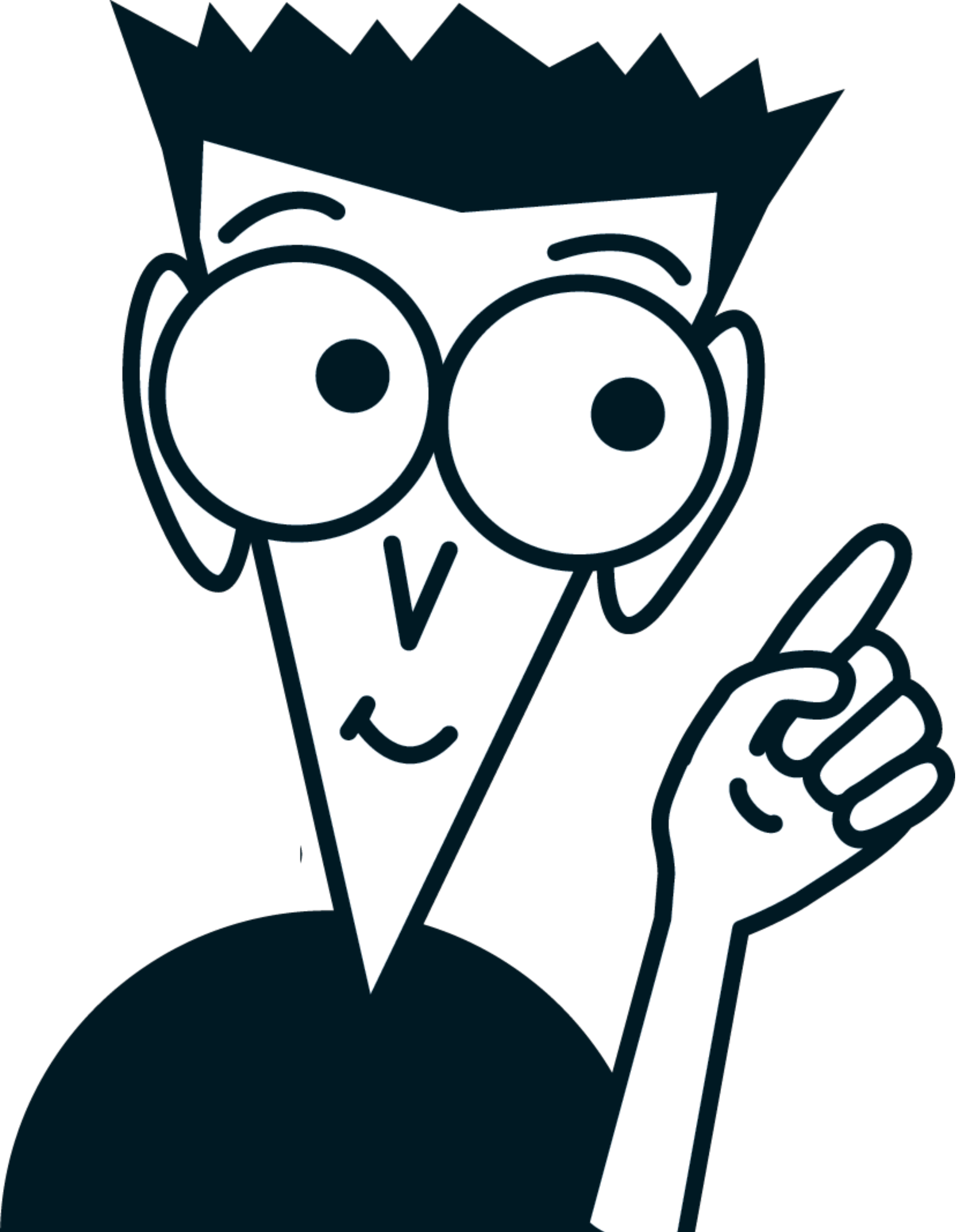
- Describing an [inspection](#) in the plugin configuration file.
- Implementing a [local inspection class](#) to inspect Java code in the IntelliJ Platform-based IDE editor.

https://www.jetbrains.org/intellij/sdk/docs/tutorials/code_inspections.html



**А почему
не detekt?**

**А почему
не Android Lint?**



**detekt — только
про Kotlin-код**



**А чем инспекции
лучше Android Lint?...**



Android Lint — это фреймворк над инспекциями

Android Lint — это фреймворк над инспекциями

Но есть нюансы

Нюансы?..

- 1 Нет документации
- 2 Очень нестабильное API
- 3 «Пляски с бубном» при создании из плагина

Нюансы?..

1

Нет документации

2

Очень нестабильное API

3

«Пляски с бубном» при создании из плагина

Нюансы?..

1

Нет документации

2

Очень нестабильное API

3

«Пляски с бубном» при создании из плагина

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_lint_check_container_root"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
```

```
<ImageView
```

```
    android:id="@+id/some_wrong_id_declaration"
```

Wrong id declaration! Should have prefix fragment_lint_check_image

Replace current id with `fragment\_lint\_check\_image` prefix   More actions...  

```
<EditText
```

```
    android:id="@+id/sds"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"/>
```

```
<ProgressBar
```

```
    android:id="@+id/assaa"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"/>
```

Для Lint Rule нужно всего лишь...

- 1 Описать логику проверки в Detector-e
- 2 Создать класс-инспекции и добавить его в LintIdeRegistry
- 3 Зарегистрировать инспекцию в plugin.xml

Для Lint Rule нужно всего лишь...

- 1 Описать логику проверки в Detector-е
- 2 Создать класс-инспекции и добавить его в LintIdeRegistry
- 3 Зарегистрировать инспекцию в plugin.xml

Описываем логику проверки

```
class WrongViewIdCodeStyleFormattingDetector : LayoutDetector() {  
  
    ...  
  
    override fun getApplicableAttributes(): Collection<String>? {  
        return listOf(SdkConstants.ATTR_ID) // проверка для android:id атрибута  
    }  
  
    override fun visitAttribute(context: XmlContext, attribute: Attr) {  
        ...  
        if (attribute.hasCorrectIdValue()).not()) {  
            context.report(...)   
        }  
    }  
}
```

Описываем логику проверки

```
class WrongViewIdCodeStyleFormattingDetector : LayoutDetector() {  
  
    ...  
  
    override fun getApplicableAttributes(): Collection<String>? {  
        return listOf(SdkConstants.ATTR_ID) // проверка для android:id атрибута  
    }  
  
    override fun visitAttribute(context: XmlContext, attribute: Attr) {  
        ...  
        if (attribute.hasCorrectIdValue()).not()) {  
            context.report(...)   
        }  
    }  
}
```

Создаём класс инспекции

```
class WrongViewIdCodeStyleFormattingInspection : AndroidLintInspectionBase(  
    displayName = "WrongViewIdCodeStyleFormatting",  
    issue = Issue.create(  
        id = "WrongViewIdCodeStyleFormatting",  
        briefDescription = "Rule for checking android:id in layout XML files",  
        explanation = "Full description of your issue for users",  
        category = Category.LINT,  
        enabledByDefault = true,  
        severity = Severity.WARNING,  
        priority = 5,  
        implementation = Implementation(  
            WrongViewIdCodeStyleFormattingDetector::class.java,  
            Scope.RESOURCE_FILE_SCOPE  
        )  
    )  
)  
)  
)
```

Не забудем про LintIdeRegistry

```
class WrongViewIdCodeStyleFormattingInspection : AndroidLintInspectionBase(...) {  
  
    init {  
        val registry = LintIdeIssueRegistry()  
  
        val registryIssue = registry.getIssue("WrongViewIdCodeStyleFormatting")  
        if (registryIssue == null) {  
            val issues = registry.issues as? MutableList<Issue>  
            issues?.add(issue)  
        }  
    }  
}
```

Регистрируем правило в plugin.xml

```
<idea-plugin>
  ...
  <extensions defaultExtensionNs="com.intellij">

    <globalInspection
      groupName="Inspections hh.ru "
      shortName="AndroidLintWrongViewIdCodeStyleFormatting"
      hasStaticDescription="true"
      displayName="Wrong View's id"
      bundle="messages.PluginBundle"
      enabledByDefault="true"
      level="WARNING"
      implementationClass="ru.hh.plugin.lint.WrongViewIdCodeStyleFormattingInspection"
    />

  </extensions>
</idea-plugin>
```

Добавляем префикс "AndroidLint"

```
<idea-plugin>
  ...
  <extensions defaultExtensionNs="com.intellij">

    <globalInspection
      groupName="Inspections hh.ru "
      shortName="AndroidLintWrongViewIdCodeStyleFormatting"
      hasStaticDescription="true"
      displayName="Wrong View's id"
      bundle="messages.PluginBundle"
      enabledByDefault="true"
      level="WARNING"
      implementationClass="ru.hh.plugin.lint.WrongViewIdCodeStyleFormattingInspection"
    />

  </extensions>
</idea-plugin>
```



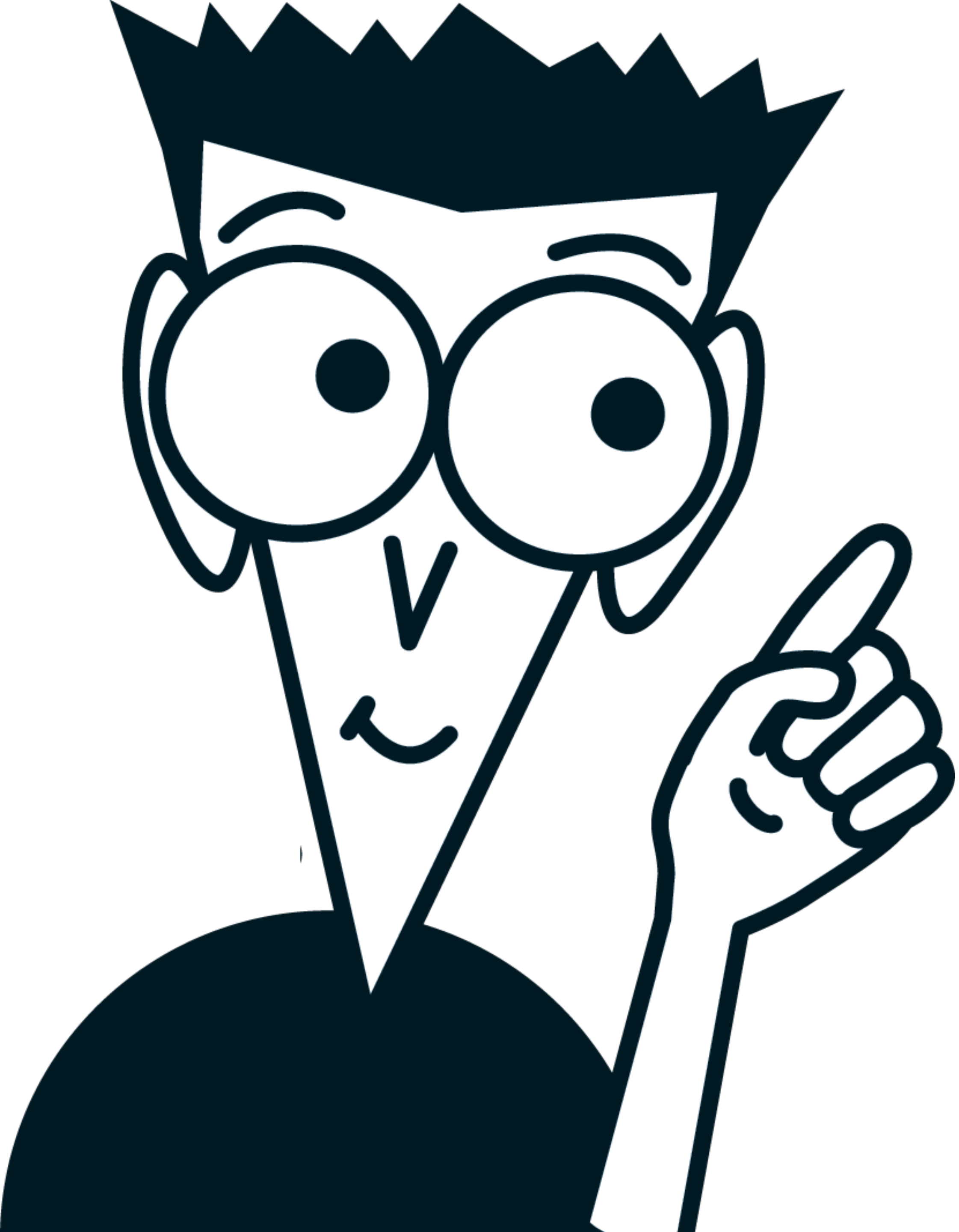
**А минусы у этого
решения есть?..**

Да.



СІ нічого не узнає про эти правила





**Разве что
TeamCity...**



Но это уже совсем
другая история

Подведем итоги



- 1 Инспекции - способ автоматизировать проверки**
- 2 Если зависите от android-плагина - юзайте Lint
- 3 Запуск инспекций на CI потребуют шаманства

Подведем итоги



1

Инспекции - способ автоматизировать проверки

2

Если зависите от android-плагина - юзайте Lint

3

Запуск инспекций на CI потребуют шаманства

Подведем итоги



- 1 Инспекции - способ автоматизировать проверки
- 2 Если зависите от android-плагина - юзайте Lint
- 3 Запуск инспекций на CI потребуют шаманства**

Есть вопросы?



- 1 Инспекции - способ автоматизировать проверки
- 2 Если зависите от android-плагины - юзайте Lint
- 3 Запуск инспекций на CI потребуют шаманства

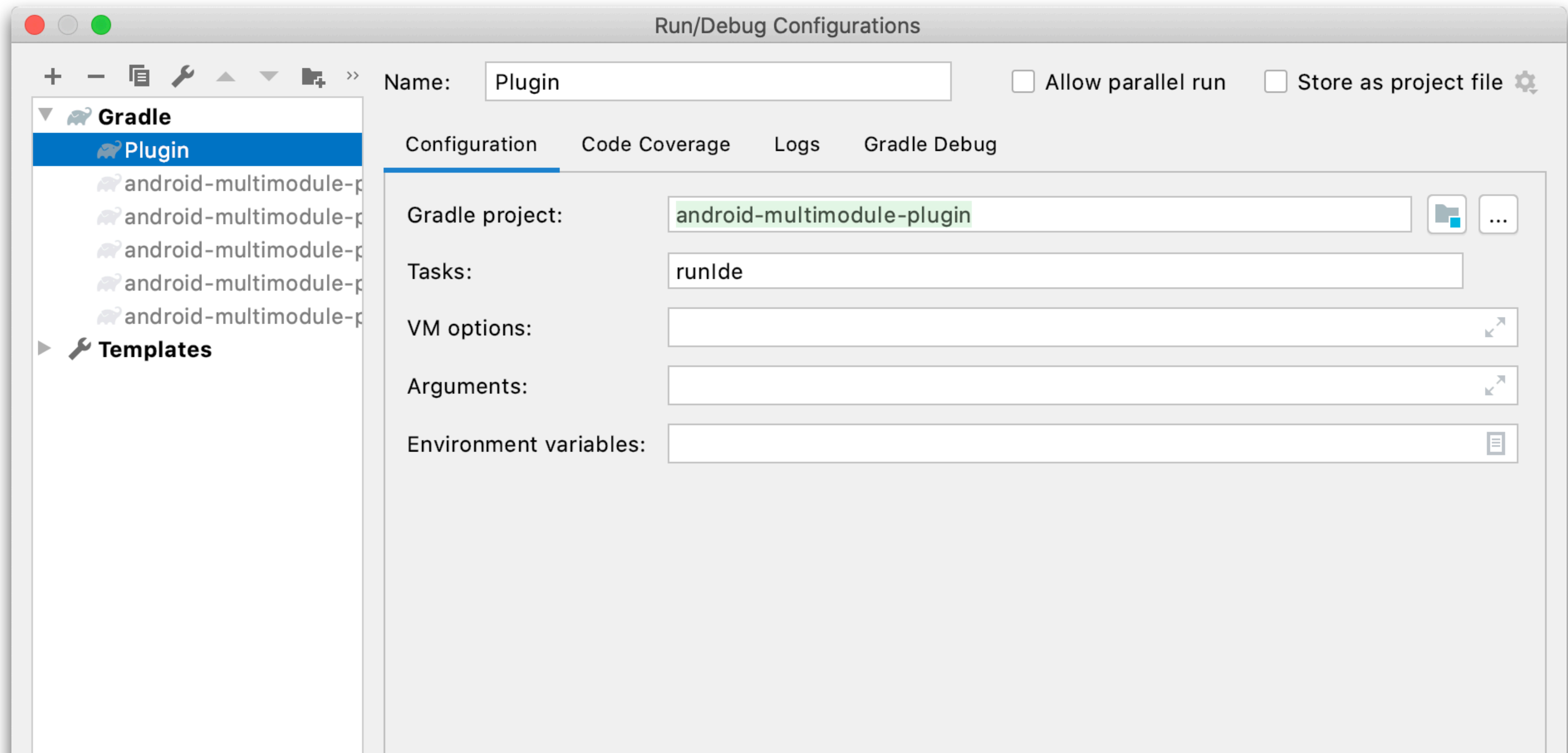


Советы и хитрости

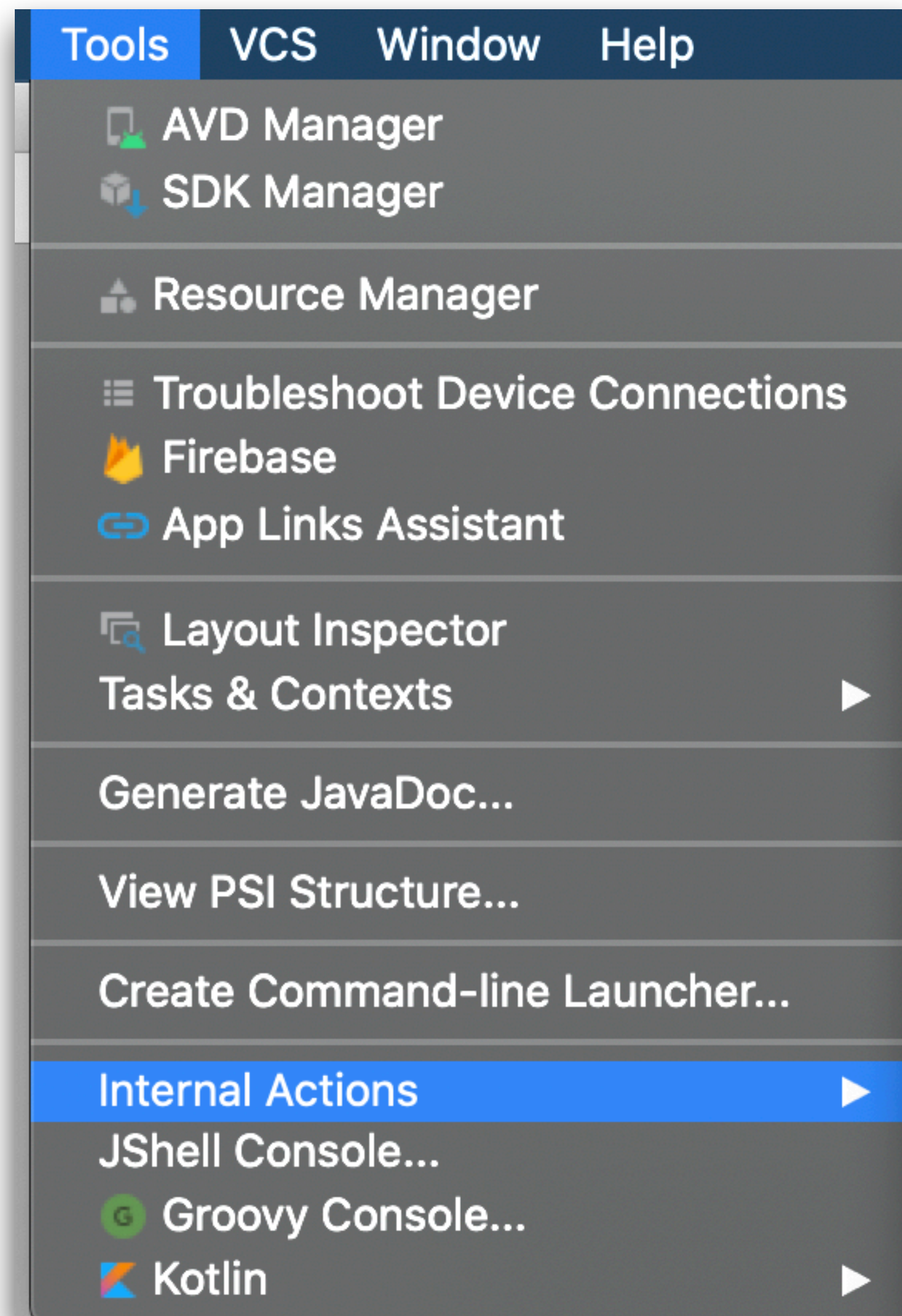
Используем

UI Inspector

Используем runIde для отладки



Находим меню Internal Actions

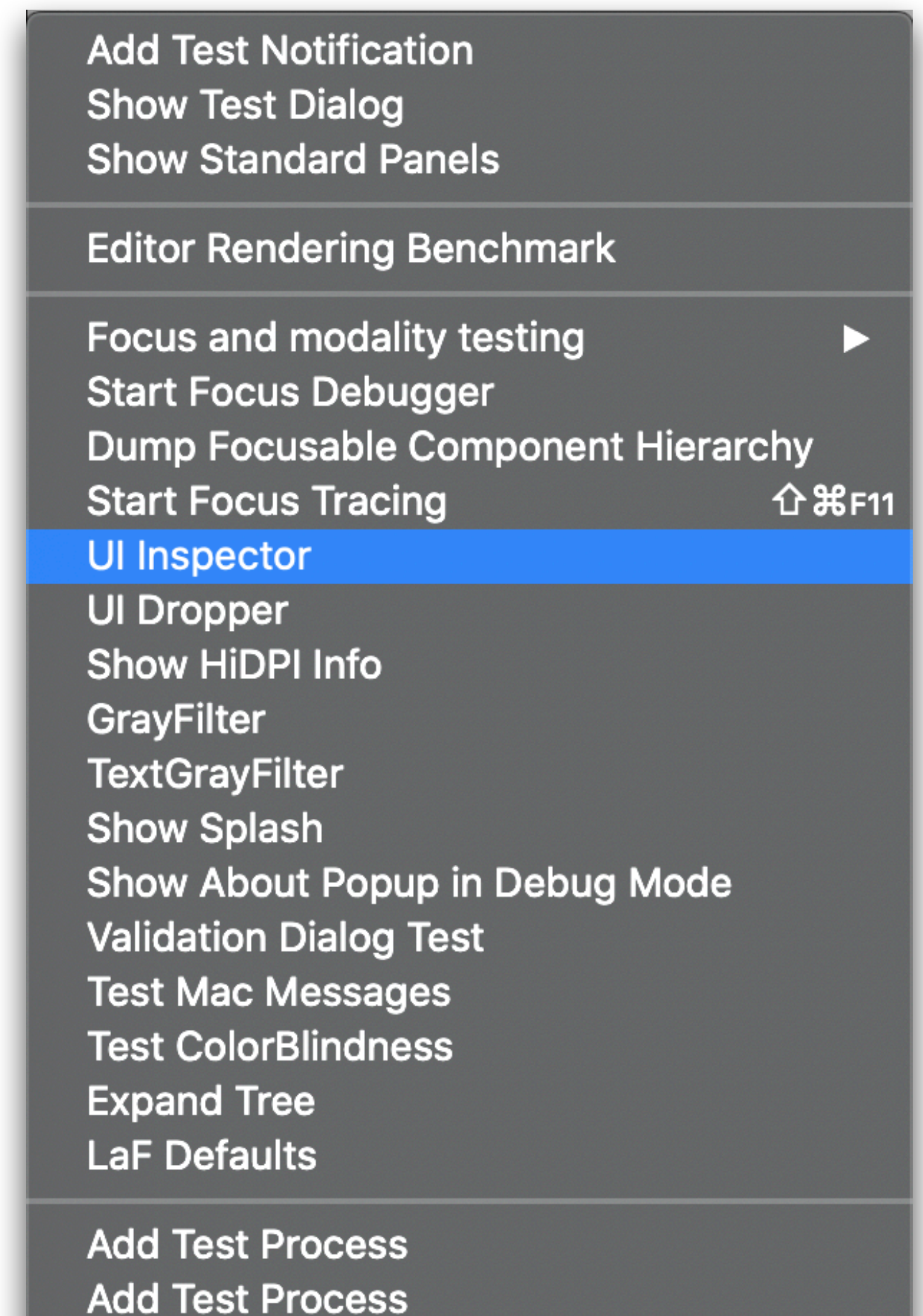
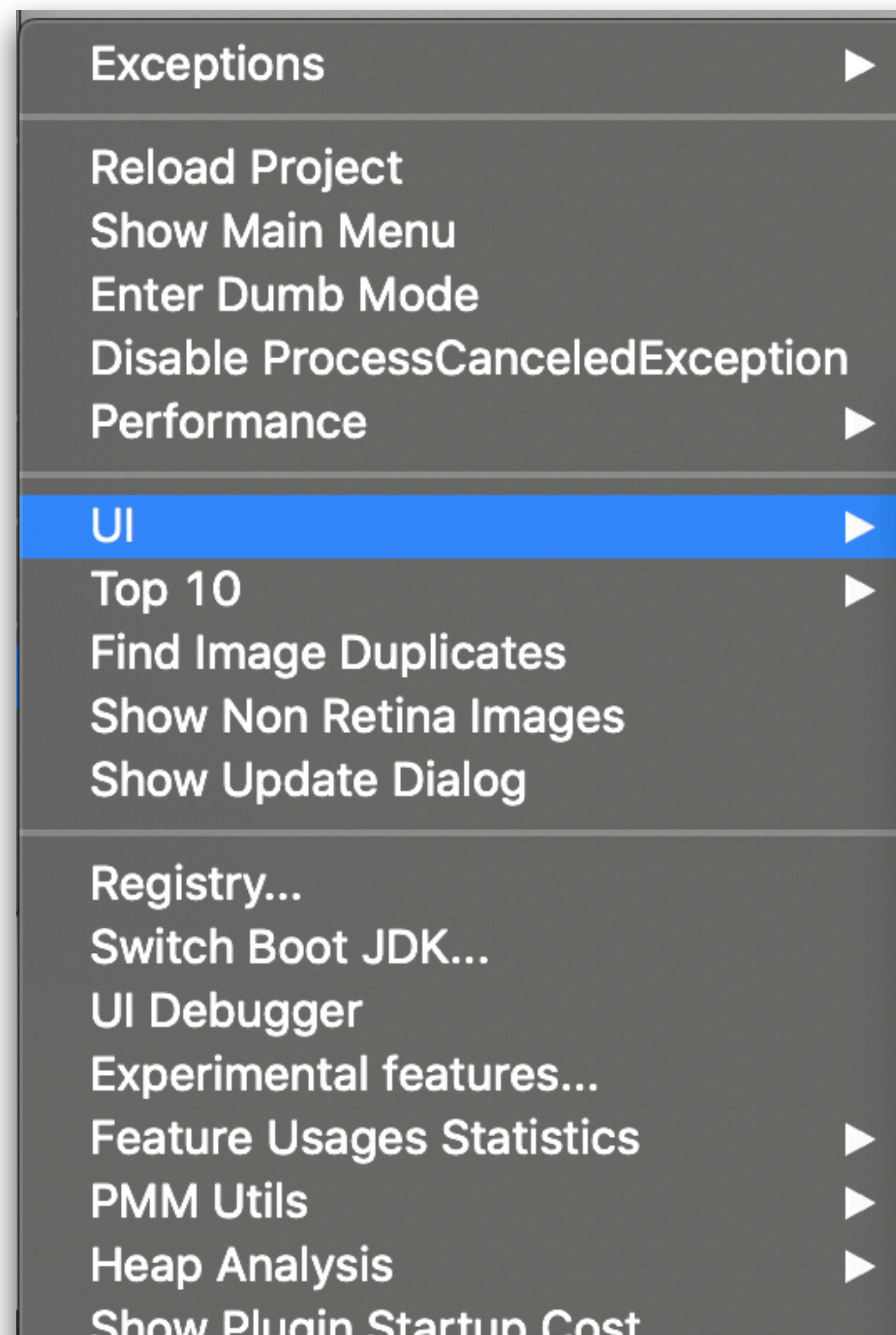
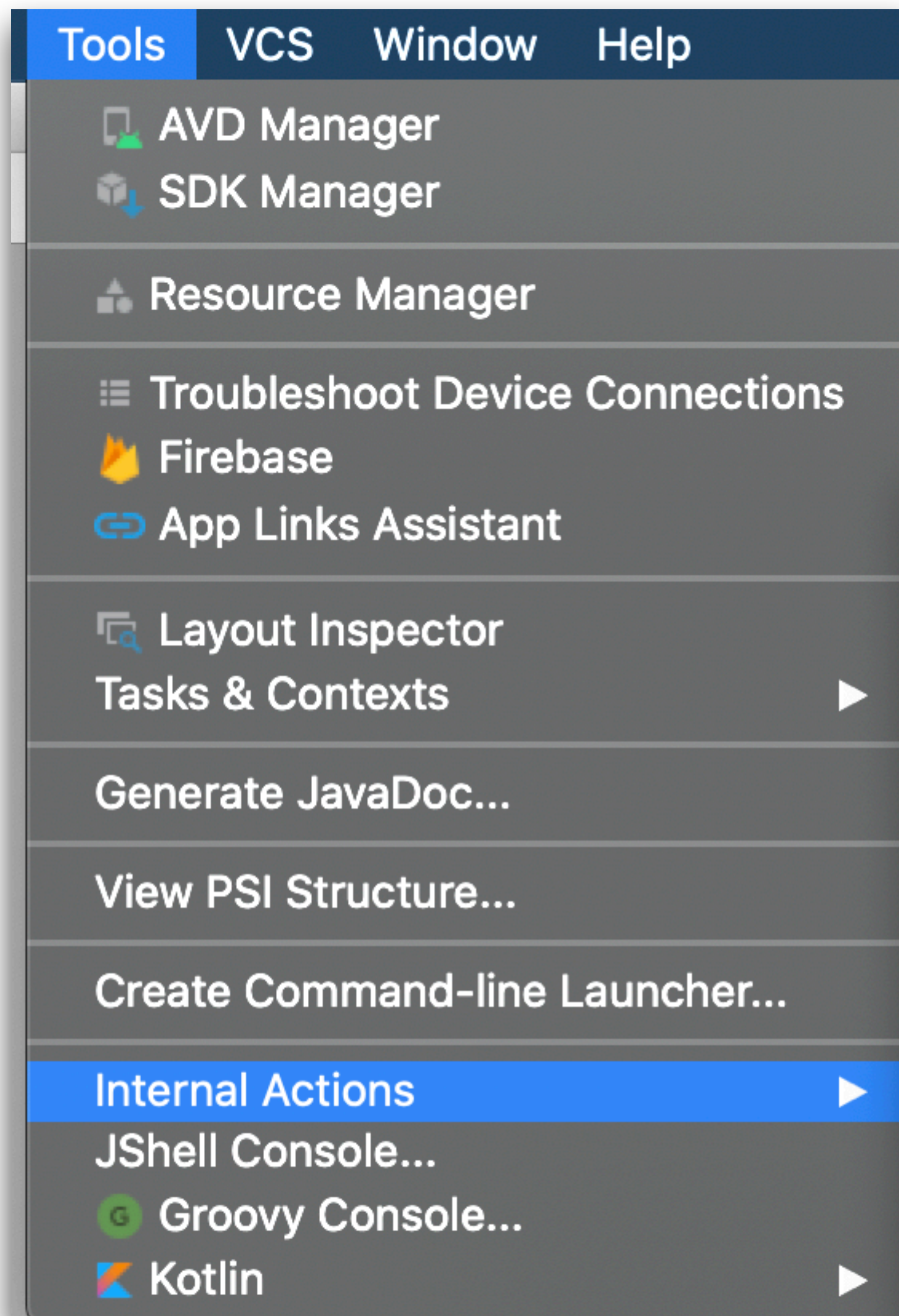


**Это меню можно включить
и в обычной Android Studio**

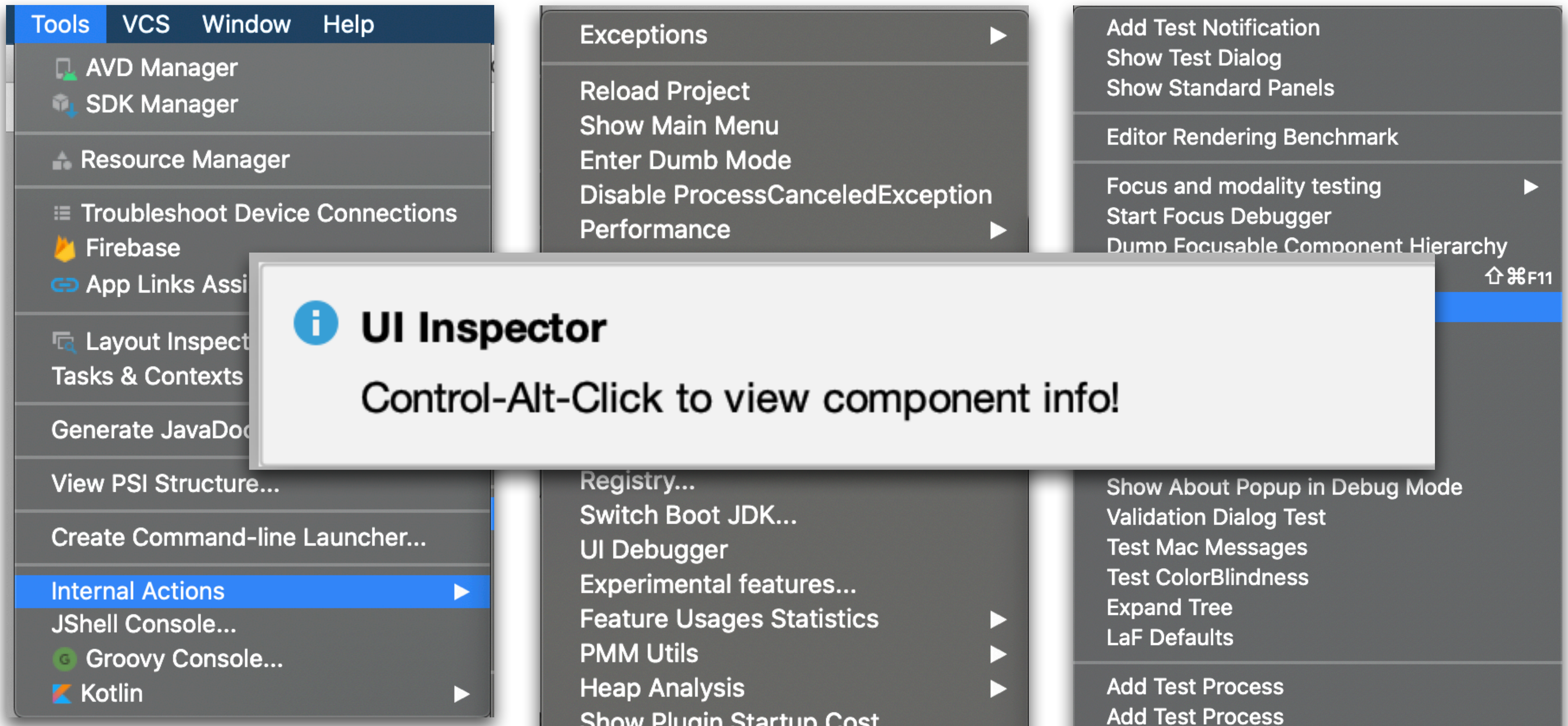
```
// Help > Custom IDEA properties
```

```
idea.is.internal=true
```

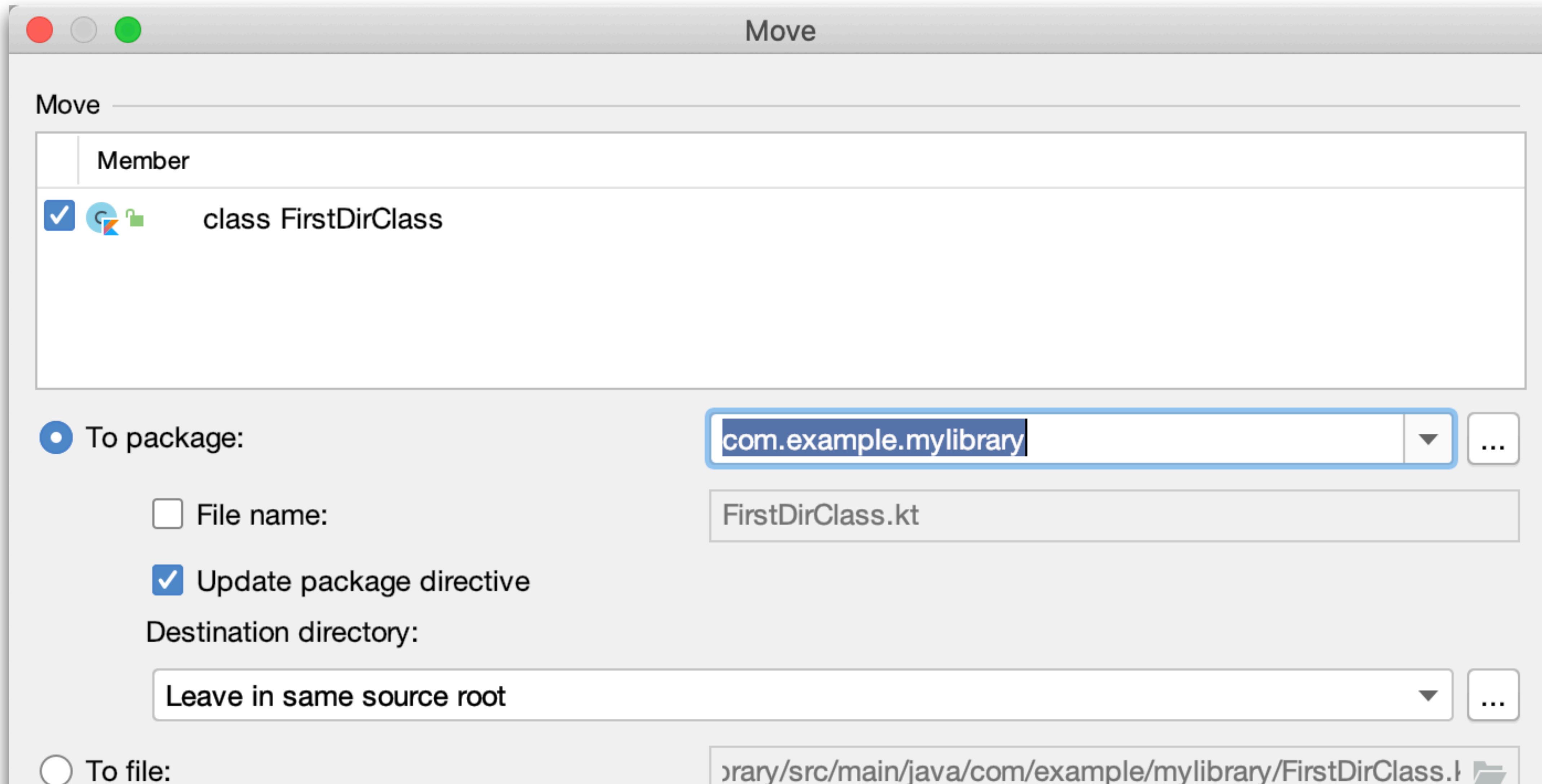
Находим UI Inspector



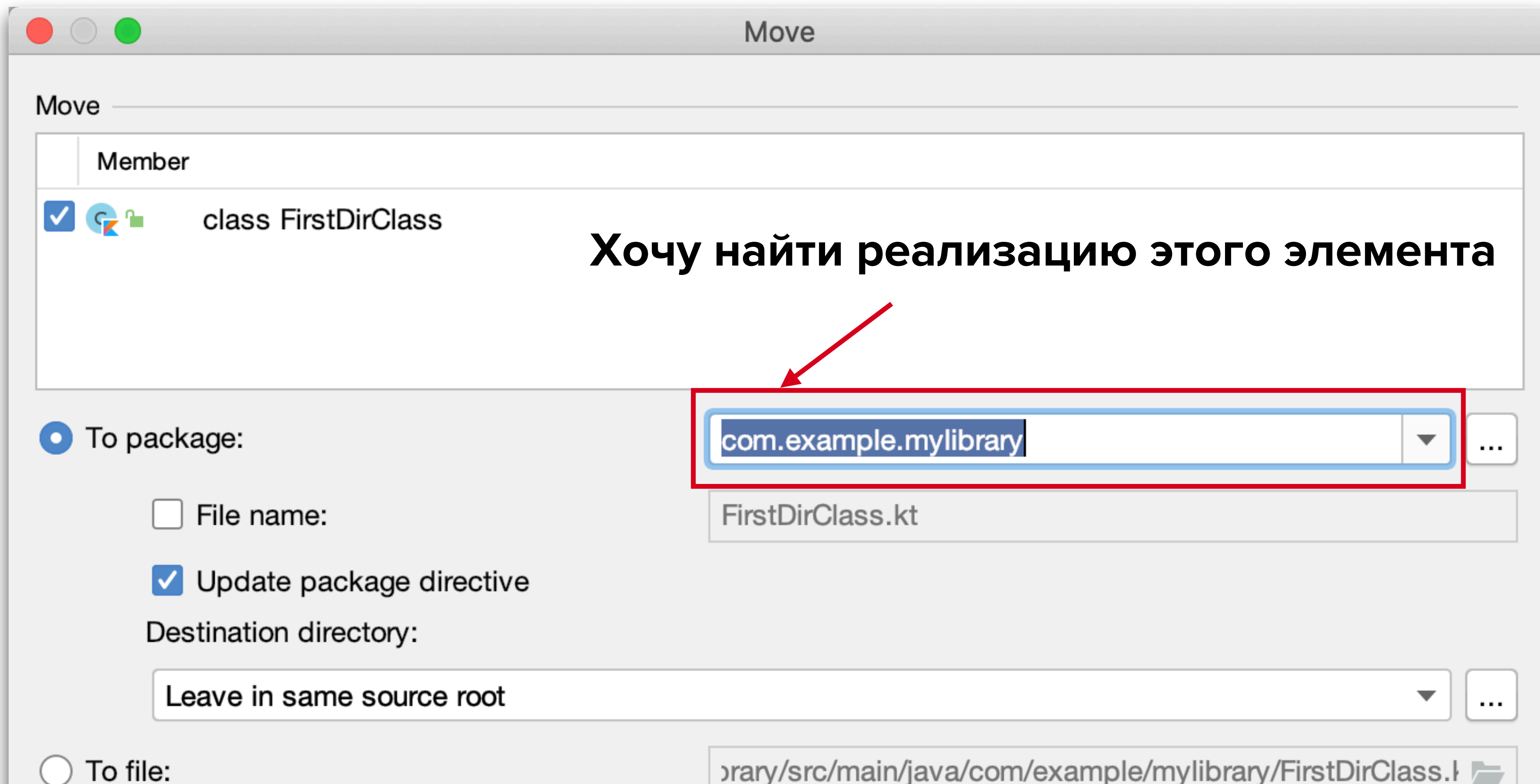
Находим UI Inspector



Пример использования



Пример использования



Иерархия UI

com.intellij.openapi.editor.impl.EditorComponentImpl

Highlight Refresh

▼ PackageNameReferenceEditorCombo: 373x30 @ 302:7, opaque, double-b

▼ EditorComboBox: 343x30 @ 0:0, opaque

BasicArrowButton: 26x30 @ 317:0

CellRendererPane: 0x0 @ 0:0

▼ ComboboxEditorTextField(myEditorField@EditorComboBox): 302x1

▼ JPanel: 302x17 @ 0:0, opaque, double-buffered

MyHeaderPanel: 302x0 @ 0:0, opaque, double-buffered

▼ JLayeredPane: 302x17 @ 0:0

▼ MyScrollPane: 302x17 @ 0:0, opaque

▼ JBViewport: 302x17 @ 0:0, opaque

EditorComponentImpl: 302x17 @ 0:0, opaque, d

MyScrollBar: 0x0 @ 0:0

OpaqueAwareScrollBar: 0x0 @ 0:0

Corner: 0x0 @ 302:0, opaque, double-buffered

Corner: 0x0 @ 0:0, opaque, double-buffered

Corner: 0x0 @ 302:17, opaque, double-buffered

Corner: 0x0 @ 0:17, opaque, double-buffered

▼ JBViewport: 0x17 @ 0:0, opaque

EditorGutterComponentImpl: 0x17 @ 0:0, opaque

FixedSizeButton: 30x30 @ 343:0, opaque

JRadioButton: 95x19 @ 0:12, opaque

▼ TextFieldWithBrowseButton: 373x30 @ 302:158, opaque, double-buffered

ExtendableTextField: 373x30 @ 0:0, opaque

Property	Value
class	com.intellij.openapi.editor.impl.EditorComponentImpl
hierarchy	javax.swing.text.JTextComponent -> javax.swing.JComponent
ui	com.intellij.openapi.editor.impl.EditorComponentImpl\$EditorAccessibility...
location	0:0
locationOnScreen	821:330
size	302x17
opaque	Yes
border	ResourceViewBorder UIResource (com.intellij.ui.components.JBViewport...
foreground	■ r:0 g:0 b:0 a:255 argb:0xFF000000 UIResource
background	□ r:255 g:255 b:255 a:255 argb:0xFFFFFFFF
font	HelveticaNeue (Helvetica Neue), 13px UIResource
minimumSize	1x1
maximumSize	32767x32767
preferredSize	141x17
text	com.example.mylibrary
editable	Yes
visibleRect	302x17 @ 0:0
layout	-
alignmentX	0.5

Иерархия UI

com.intellij.openapi.editor.impl.EditorComponentImpl

Highlight Refresh

▼ PackageNameReferenceEditorCombo: 373x30 @ 302:7, opaque, double-b

▼ EditorComboBox: 343x30 @ 0:0, opaque

BasicArrowButton: 26x30 @ 317:0

CellRendererPane: 0x0 @ 0:0

▼ ComboboxEditorTextField(myEditorField@EditorComboBox): 302x1

▼ JPanel: 302x17 @ 0:0, opaque, double-buffered

MyHeaderPanel: 302x0 @ 0:0, opaque, double-buffered

▼ JLayeredPane: 302x17 @ 0:0

▼ MyScrollPane: 302x17 @ 0:0, opaque

▼ JBViewport: 302x17 @ 0:0, opaque

EditorComponentImpl: 302x17 @ 0:0, opaque, d

MyScrollBar: 0x0 @ 0:0

OpaqueAwareScrollBar: 0x0 @ 0:0

Corner: 0x0 @ 302:0, opaque, double-buffered

Corner: 0x0 @ 0:0, opaque, double-buffered

Corner: 0x0 @ 302:17, opaque, double-buffered

Corner: 0x0 @ 0:17, opaque, double-buffered

▼ JBViewport: 0x17 @ 0:0, opaque

EditorGutterComponentImpl: 0x17 @ 0:0, opaque

FixedSizeButton: 30x30 @ 343:0, opaque

JRadioButton: 95x19 @ 0:12, opaque

▼ TextFieldWithBrowseButton: 373x30 @ 302:158, opaque, double-buffered

ExtendableTextField: 373x30 @ 0:0, opaque

Property	Value
class	com.intellij.openapi.editor.impl.EditorComponentImpl
hierarchy	javax.swing.text.JTextComponent -> javax.swing.JComponent
ui	com.intellij.openapi.editor.impl.EditorComponentImpl\$EditorAccessibility...
location	0:0
locationOnScreen	821:330
size	302x17
opaque	Yes
border	ResourceViewBorder UIResource (com.intellij.ui.components.JBViewport...
foreground	■ r:0 g:0 b:0 a:255 argb:0xFF000000 UIResource
background	□ r:255 g:255 b:255 a:255 argb:0xFFFFFFFF
font	HelveticaNeue (Helvetica Neue), 13px UIResource
minimumSize	1x1
maximumSize	32767x32767
preferredSize	141x17
text	com.example.mylibrary
editable	Yes
visibleRect	302x17 @ 0:0
layout	-
alignmentX	0.5

Резюме



UI Inspector поможет найти функционал, который вы уже где-то видели

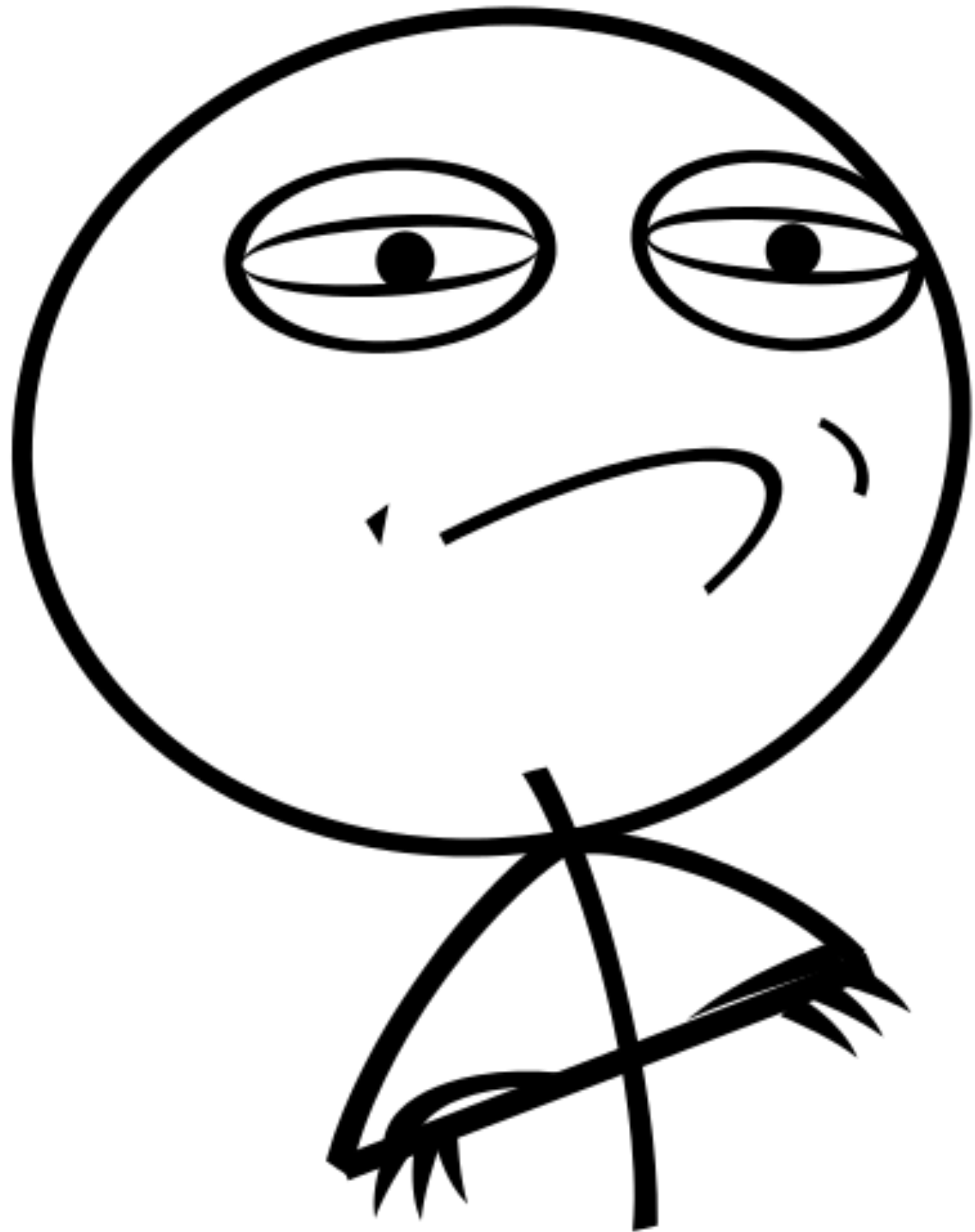
Исследуем

ГОТОВЫЕ плагины

A background image of Jack Sparrow from the Pirates of the Caribbean franchise, looking off to the side with a serious expression. He is wearing his signature dreadlocks and a dark coat.

**У нас нет
документации,
но есть
кое-что получше**

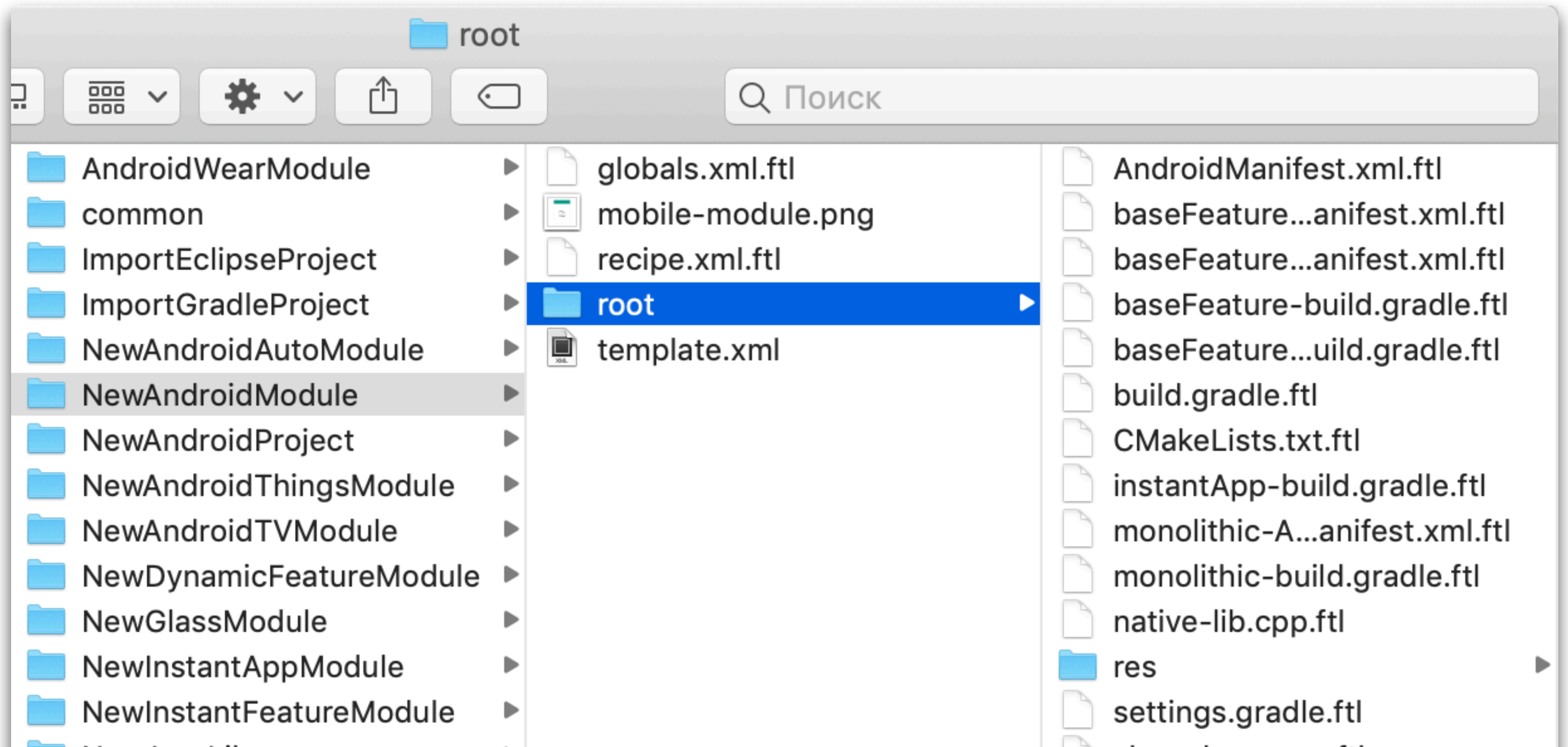
код плагинов



Найдем что-нибудь

полезное

Вспомним про FreeMarker



Можно рендерить свои шаблоны

```
private fun resolveTemplateInternal(
    event: AnActionEvent,
    templateDirName: String,
    dialogTitle: String,
    afterFilesCreationAction: (List<File>, Map<String, Any>) -> Unit
) {
    val (project, androidFacet, currentModuleRootDir, currentModulePackageName) = fetchInitil

    val templateHandle = getTemplateHandle(project, templateDirName)
    val namedModuleTemplate = NamedModuleTemplate(
        name = NAMED_MODULE_TEMPLATE_NAME,
        paths = AndroidModulePathsData(currentModuleRootDir, currentModulePackageName)
    )
    val renderTemplateModel = RenderTemplateModel(
        androidFacet,
        templateHandle,
        currentModulePackageName,
        namedModuleTemplate,
        COMMAND_NAME,
        ProjectSyncInvoker.DefaultProjectSyncInvoker(),
        shouldOpenFiles: true
    )
    val templateWizardStep = ConfigureTemplateParametersStep(
        renderTemplateModel
    )
}
```

Можно легко их отлаживать

```
private fun doDryRun(event: AnActionEvent, templateDirName: String) {
    val (project, androidFacet, currentModuleRootDir, currentModulePackageName) = fetchInitData(event)

    val templateParametersMap = mutableMapOf<String, Any>().also { paramsMap ->
        DryRunValuesInjector(project, androidFacet, currentModuleRootDir, currentModulePackageName)
            .injectValues(paramsMap)
    }

    val templateHandle = getTemplateHandle(project, templateDirName)

    val context = RenderingContext.Builder.newContext(templateHandle.template, project)
        .withCommandName(COMMAND_NAME)
        .withModule(androidFacet.module)
        .withParams(templateParametersMap)
        .build()

    val notificationsFactory = NotificationsFactory(project)
    if (templateHandle.template.render(context, dryRun: true)) {
        notificationsFactory.info(CODE_GENERATOR_MESSAGES_TITLE, "Template at `${templateDirName}` has no errors")
    } else {
        notificationsFactory.error(
```

В итоге чему научились

- 1 **Хранить FreeMarker шаблоны ближе к своей кодовой базе**
- 2 Быстрее отлаживать эти шаблоны без перезагрузки Android Studio

В итоге чему научились

1

Хранить FreeMarker шаблоны ближе к своей кодовой базе

2

Быстрее отлаживать эти шаблоны без перезагрузки Android Studio

В итоге чему научились

1

Хранить FreeMarker шаблоны ближе к своей кодовой базе

2

Быстрее отлаживать эти шаблоны без перезагрузки Android Studio



Поддерживаем

новую Android Studio

Каждая версия AS требует правок



Меняем package name-ы

```
// Android Studio 3.5.*
```

```
import org.jetbrains.kotlin.idea.refactoring.toPsiDirectory
```

```
fun Project.findPsiFileByName(filename: String): PsiFile? {  
    return this.guessProjectDir().toPsiDirectory()?.findFile(filename)  
}
```

```
// Android Studio 3.6.3
```

```
import org.jetbrains.kotlin.idea.core.util.toPsiDirectory
```

```
fun Project.findPsiFileByName(filename: String): PsiFile? {  
    return this.guessProjectDir().toPsiDirectory()?.findFile(filename)  
}
```

Страдаем из-за UnstableApiUsage

```
// Android Studio 3.5.*
```

```
titledRow("Choose your destiny:") {  
    row {  
        cell {  
            comboBox(  
                model = appModuleComboBoxModel,  
                growPolicy = GrowPolicy.SHORT_TEXT,  
                renderer = null  
            )  
        }  
    }  
}
```

Страдаем из-за UnstableApiUsage

```
// Android Studio 3.6.3
```

```
titledRow("Choose your destiny:") {  
    row {  
        cell {  
            val cellBuilder = comboBox(  
                model = CollectionComboBoxModel(appModules),  
                getter = { selectedAppModule.selected },  
                setter = { selectedAppModule = it },  
                growPolicy = GrowPolicy.SHORT_TEXT,  
                renderer = null  
            ).also { appModuleComboBox = it.component }  
            cellBuilder.component(CCFlags.growX)  
        }  
    }  
}
```

Полностью меняем реализацию

```
// Android Studio 3.5.*
```

```
val AndroidFacet.packageName: String  
    get() = AndroidManifestUtils.getPackageName(this)  
        ?: String.EMPTY
```

```
// Android Studio 3.6.3
```

```
val AndroidFacet.packageName: String  
    get() = cachedValueFromPrimaryManifest { this.packageName }.value  
        ?: String.EMPTY
```

Подведем итоги



- 1 Для новой AS придется дорабатывать плагин
- 2 Старайтесь меньше использовать UnstableApi

Подведем итоги



1

Для новой AS придется дорабатывать плагин

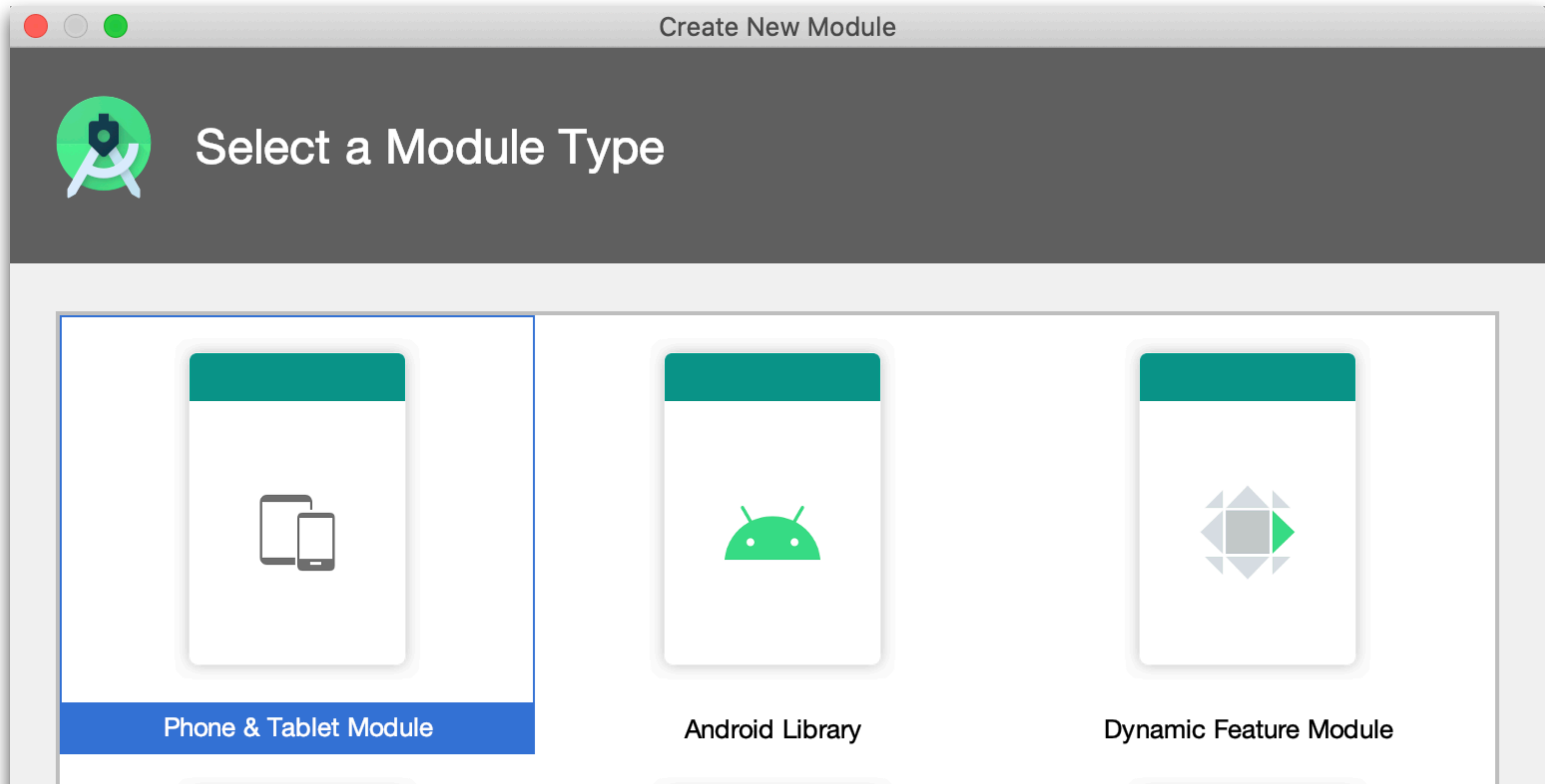
2

Старайтесь меньше использовать UnstableApi

Грустная история

Со счастливым концом

У каждого есть мечта (с)



И к ней возможно приблизиться

com.android.tools.idea.npw.ui.WizardGallery

Highlight

Refresh

JPANEL: 771x419 @ 0:108, opaque, double-buffered

JPANEL: 723x371 @ 24:24, opaque, double-buffered

JBScrollPane: 723x371 @ 0:0, opaque

JBScrollBar: 14x369 @ 708:1

JBScrollBar: 0x0 @ 0:0

Corner: 0x0 @ 722:1, opaque, double-buffered

Corner: 0x0 @ 1:1, opaque, double-buffered

Corner: 0x0 @ 722:370, opaque, double-buffered

Corner: 0x0 @ 1:370, opaque, double-buffered

JBViewport: 721x369 @ 1:1, opaque

WizardGallery: 721x1102 @ 0:0, opaque

Clicked Info

CellRendererPane: 0x0 @ 0:0

CellRendererPane: 0x0 @ 0:0

nel: 771x47 @ 0:527, opaque, double-buffered

ErrorText: 0x0 @ 0:0, opaque, double-buffered

JBScrollPane: 0x0 @ 0:0

JBViewport: 0x0 @ 0:0, opaque

JLabel(myLabel@ErrorText): 0x0 @ 0:0

JBScrollBar: 0x0 @ 0:0

JBScrollBar: 0x0 @ 0:0

Property	Value
class	com.android.tools.idea.npw.ui.WizardGallery
hierarchy	com.android.tools.adtui.ASGallery -> com.intellij.ui.component...
ui	com.intellij.ui.components.WideSelectionListUI@18b4300a
location	0:0
locationOnScreen	503:290
size	721x1102
opaque	Yes
border	ViewBorder (com.intellij.ui.components.JBViewport\$ViewBord...
foreground	■ r:0 g:0 b:0 a:255 argb:0xFF000000 UIResource
background	□ r:255 g:255 b:255 a:255 argb:0xFFFFFFFF
font	HelveticaNeue (Helvetica Neue), 13px UIResource
cellRenderer	ExpandedItemListAdapterWrapper[com.android.tools.adt...
minimumSize	716x1102
maximumSize	716x1102
preferredSize	716x1102
visibleRect	721x369 @ 0:0
layout	-
alignmentY	0.5

191

Находим класс нужного Action-a

@NotNull

```
public static ChooseModuleTypeStep createWithDefaultGallery(
    Project project,
    ProjectSyncInvoker projectSyncInvoker
) {
    ArrayList<ModuleGalleryEntry> moduleDescriptions = new ArrayList<>();
    for (ModuleDescriptionProvider provider :
ModuleDescriptionProvider.EP_NAME.getExtensions()) {
        moduleDescriptions.addAll(provider.getDescriptions(project));
    }
    return new ChooseModuleTypeStep(
        project, moduleDescriptions, projectSyncInvoker
    );
}
```

Находим точку расширения

```
@NotNull
public static ChooseModuleTypeStep createWithDefaultGallery(
    Project project,
    ProjectSyncInvoker projectSyncInvoker
) {
    ArrayList<ModuleGalleryEntry> moduleDescriptions = new ArrayList<>();
    for (ModuleDescriptionProvider provider :
ModuleDescriptionProvider.EP_NAME.getExtensions()) {
        moduleDescriptions.addAll(provider.getDescriptions(project));
    }
    return new ChooseModuleTypeStep(
        project, moduleDescriptions, projectSyncInvoker
    );
}
```

Extension Points & Extensions

```
/**
 * This interface provides extension point to customize the
 * list of new Modules that can be created.
 * Each provider can return a list of Modules it knows how to create,
 * and a UI Step to implement its creation.
 */
public interface ModuleDescriptionProvider {
    ExtensionPointName<ModuleDescriptionProvider> EP_NAME =
        ExtensionPointName.create("com.android.moduleDescriptionProvider");

    Collection<? extends ModuleGalleryEntry> getDescriptions(Project project);
}
```

Настраиваем gradle-intellij-plugin

```
// build.gradle
```

```
intellij.type = 'IC'
```

```
// From Android Studio 3.6.3 "About" screen
```

```
intellij.version = "192.7142.36"
```

```
intellij.plugins = ["android", "Kotlin", "Groovy", "java", "git4idea"]
```

```
runIde {
```

```
    ideDirectory "/Applications/Android Studio.app/Contents"
```

```
}
```

Настраиваем plugin.xml

```
<idea-plugin>
    ...

    <depends>org.jetbrains.android</depends>

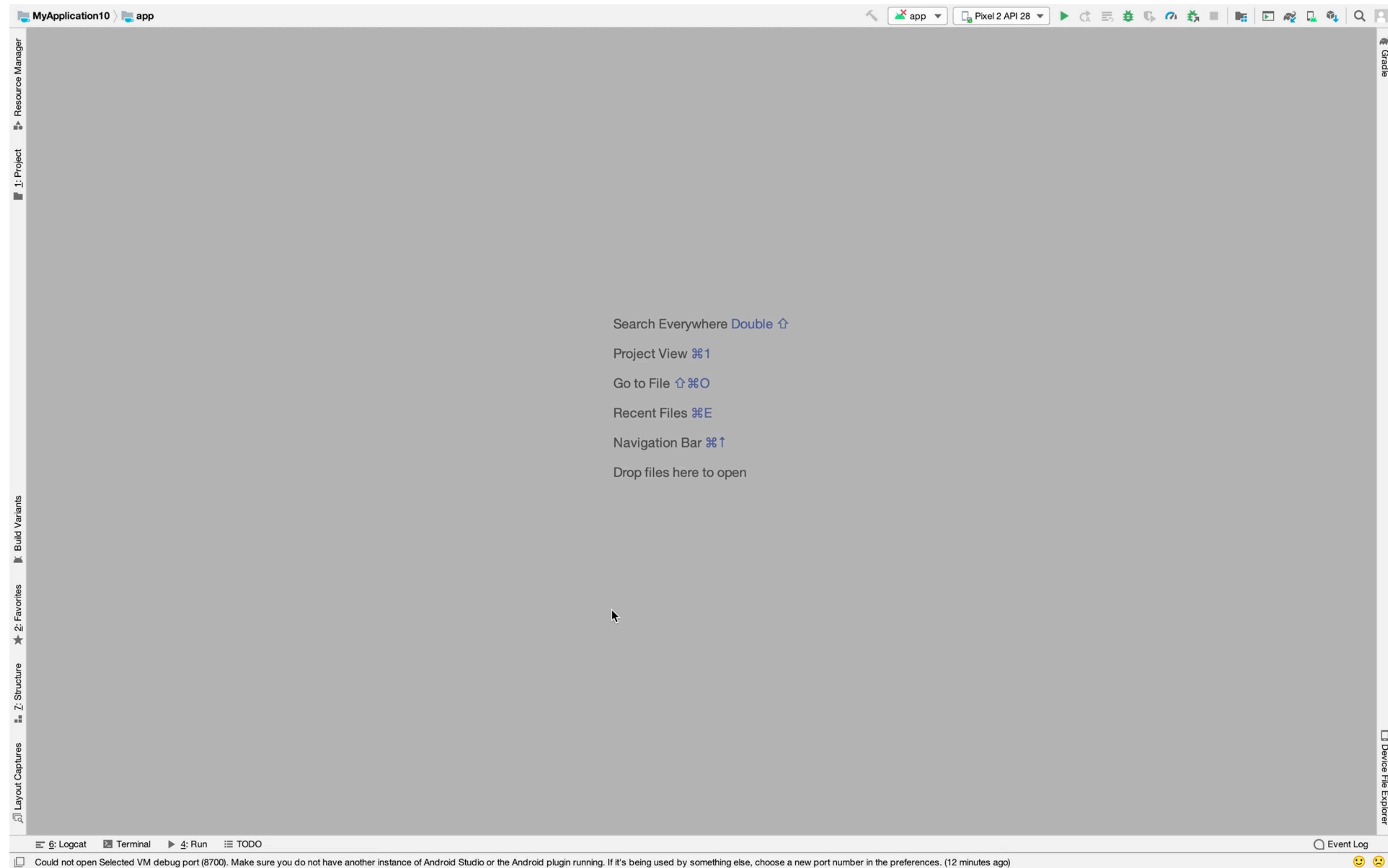
    <extensions defaultExtensionNs="com.android">
        <moduleDescriptionProvider
            implementation="ru.hh.android.plugin.MyModuleDescriptionProvider" />
    </extensions>

</idea-plugin>
```

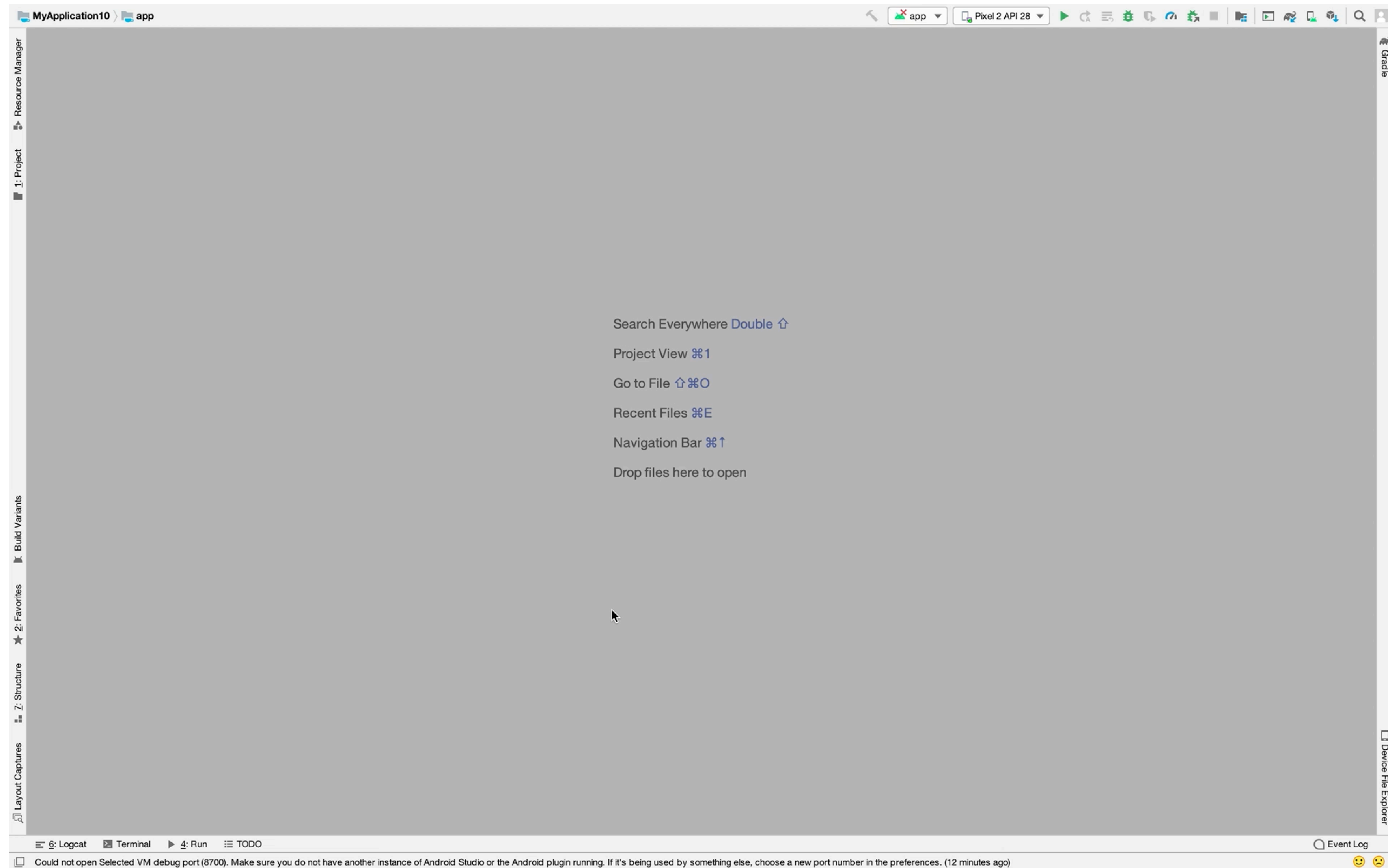
Реализуем нужный extension

```
class MyModuleDescriptionProvider : ModuleDescriptionProvider {  
    override fun getDescriptions(project: Project): MutableCollection<out  
ModuleGalleryEntry> {  
        return ImmutableList.of(MyModuleEntry())  
    }  
  
    private class MyModuleEntry : ModuleGalleryEntry {  
        override fun getIcon(): Image? = null  
        override fun getName(): String = "My module"  
        override fun getDescription(): String = "My module description"  
        override fun createStep(  
            model: NewModuleModel  
        ): SkippableWizardStep<*> = CreateMyModuleStep(model)  
    }  
}
```

Запускаем Android Studio и...

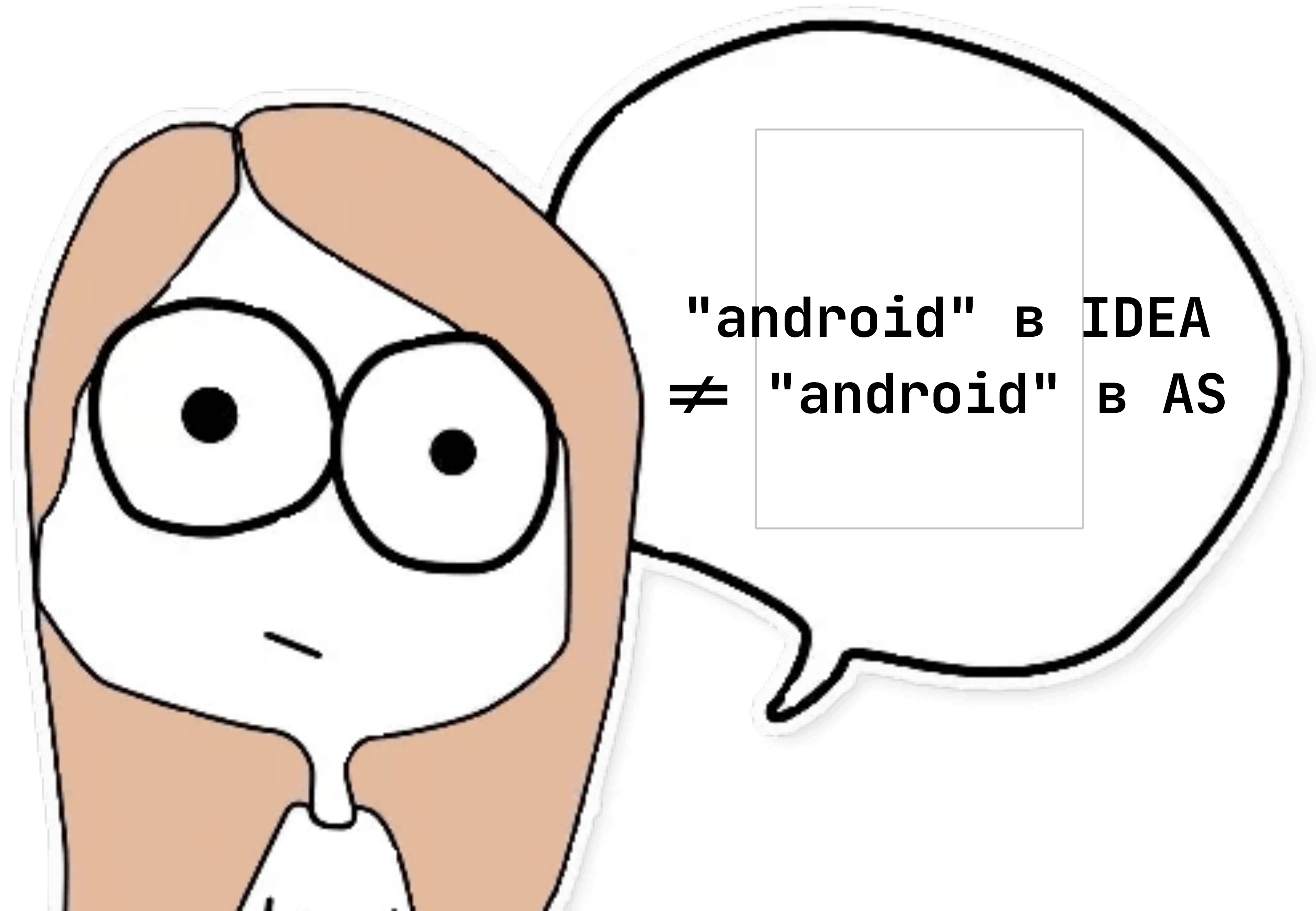


Запускаем Android Studio и...



WTF?!!!!

```
com.google.common.util.concurrent.ExecutionError: java.lang.AbstractMethodError: Receiver
class ru.hh.android.plugin.MyModuleDescriptionProvider$MyModuleEntry
does not define or inherit an implementation of the resolved method
'abstract javax.swing.Icon getIcon()' of interface
com.android.tools.idea.npw.module.ModuleGalleryEntry.
    at com.google.common.cache.LocalCache$Segment.get(LocalCache.java:2048)
    at com.google.common.cache.LocalCache.get(LocalCache.java:3951)
    at com.google.common.cache.LocalCache.getOrLoad(LocalCache.java:3973)
    at com.google.common.cache.LocalCache$LocalLoadingCache.get(LocalCache.java:4957)
    at com.android.tools.adtui.ASGallery.getCellIcon(ASGallery.java:357)
    at com.android.tools.adtui.ASGallery.access$300(ASGallery.java:72)
    at
com.android.tools.adtui.ASGallery$GalleryCellRenderer.createCellRendererComponent(ASGallery.java
:411)
    at
com.android.tools.adtui.ASGallery$GalleryCellRenderer.getListCellRendererComponent(ASGallery.jav
a:403)
    at
com.intellij.ui.ExpandedTextListCellRendererWrapper.getListCellRendererComponent(ExpandedTextLi
```



**"android" в IDEA
≠ "android" в AS**

Выход есть — `intellij.localPath`

```
// build.gradle
```

```
intellij.type = 'IC'
```

```
// From Android Studio 3.6.3 "About" screen
```

```
// intellij.version = "192.7142.36"
```

```
intellij.localPath = "/Applications/Android Studio.app/Contents"
```

```
intellij.plugins = ["android", "Kotlin", "Groovy", "java", "git4idea"]
```

```
runIde {
```

```
    ideDirectory "/Applications/Android Studio.app/Contents"
```

```
}
```

Выход есть — `intellij.localPath`

```
// build.gradle
```

```
intellij.type = 'IC'
```

```
// From Android Studio 3.6.3 "About" screen
```

```
// intellij.version = "192.7142.36"
```

```
intellij.localPath = "/Applications/Android Studio.app/Contents"
```

```
intellij.plugins = ["android", "Kotlin", "Groovy", "java", "git4idea"]
```

```
runIde {
```

```
    ideDirectory "/Applications/Android Studio.app/Contents"
```

```
}
```

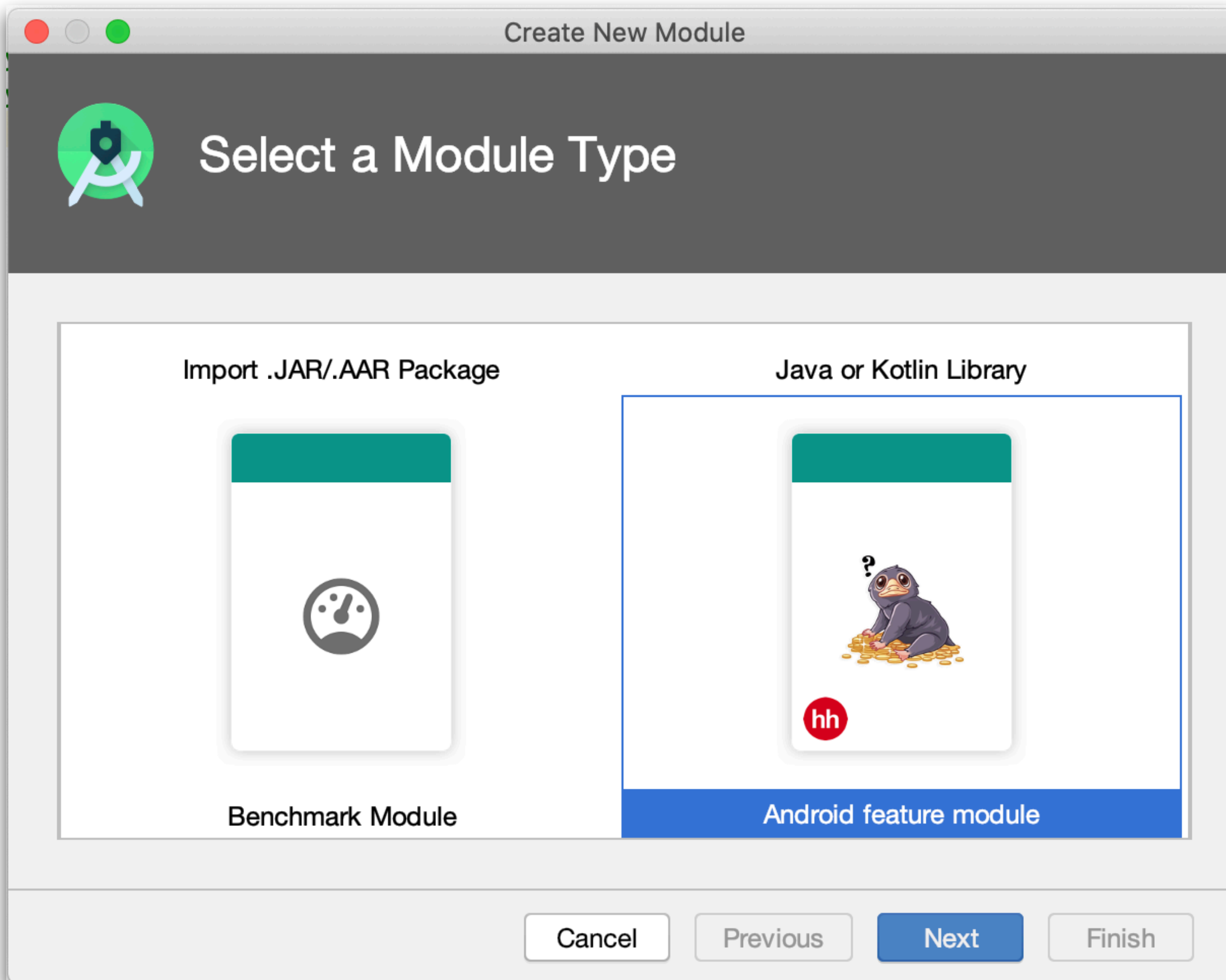
ПРАВИМ КОД

```
class MyModuleDescriptionProvider : ModuleDescriptionProvider {  
    override fun getDescriptions(project: Project): MutableCollection<out  
ModuleGalleryEntry> {  
        return ImmutableList.of(MyModuleEntry())  
    }  
  
    private class MyModuleEntry : ModuleGalleryEntry {  
        override val icon: Icon? = null  
        override val name: String = "My module"  
        override val description: String = "My module description"  
        override fun createStep(  
            model: NewModuleModel  
        ): SkippableWizardStep<*> = CreateMyModuleStep(model)  
    }  
}
```

API действительно изменилось

```
class MyModuleDescriptionProvider : ModuleDescriptionProvider {  
  
    override fun getDescriptions(project: Project): MutableCollection<out  
ModuleGalleryEntry> {  
        return ImmutableList.of(MyModuleEntry())  
    }  
  
    private class MyModuleEntry : ModuleGalleryEntry {  
        override val icon: Icon? = null  
        override val name: String = "My module"  
        override val description: String = "My module description"  
        override fun createStep(  
            model: NewModuleModel  
        ): SkippableWizardStep<*> = CreateMyModuleStep(model)  
    }  
}
```

Запускаем Android Studio и...



Но есть мааааленькие проблемы...



Сборка android-multimodule-plugin APP 10:53 PM

[hh-android](#) › [Modules Plugin](#) › [#33](#) failed. Changes by [Стрельченко Павел](#) Responsible

Users: Стрельченко Павел

Но есть мааааленькие проблемы...

```
/**
 * Represents a file in {@link VirtualFileSystem}. A particular file is represented by equal
 * {@code VirtualFile} instances for the entire lifetime of the IDE process, unless the file
 * is deleted, in which case {@link #isValid()} will return {@code false}.
 * <p/>
 * VirtualFile instances are created on request, so there can be several instances corresponding to the same file.
 * All of them are equal, have the same {@code hashCode} and use shared storage for all related data, including user data (s
 * <p/>
 * If an in-memory implementation of VirtualFile is required, {@link LightVirtualFile}
 * can be used.
 * <p/>
 * Please see <a href="http://www.jetbrains.org/intellij/sdk/docs/basics/virtual_file_system.html">Virtual File System</a>
 * for high-level overview.
 *
 * @see VirtualFileSystem
 * @see VirtualFileManager
 */
public abstract class VirtualFile extends UserDataHolderBase implements ModificationTracker {
    public static final VirtualFile[] EMPTY_ARRAY = new VirtualFile[0];

    /**
     * Used as a property name in the {@link VirtualFilePropertyEvent} fired when the name of a
     * {@link VirtualFile} changes.
     *
     * @see VirtualFileListener#propertyChanged
     */
}
```

Но есть мааааленькие проблемы...

```
package com.intellij.openapi.vfs;

import ...

public abstract class VirtualFile extends UserDataHolderBase implements ModificationTracker {
    public static final VirtualFile[] EMPTY_ARRAY = new VirtualFile[0];
    public static final String PROP_NAME = "name";
    public static final String PROP_ENCODING = "encoding";
    public static final String PROP_WRITABLE = "writable";
    public static final String PROP_HIDDEN = "HIDDEN";
    public static final String PROP_SYMLINK_TARGET = "symlink";
    private static final Logger LOG = Logger.getInstance("#com.intellij.openapi.vfs.VirtualFile");
    private static final Key<byte[]> BOM_KEY = Key.create("BOM");
    private static final Key<Charset> CHARSET_KEY = Key.create("CHARSET");
    private static final Key<String> DETECTED_LINE_SEPARATOR_KEY = Key.create("DETECTED_LINE_SEPARATOR_KEY");

    protected VirtualFile() {
    }

    @NotNull
    public abstract String getName();
}
```

Мы с ними

обязательно справимся

Подведем итоги



- 1 UI Inspector поможет найти существующие классы**
- 2 Раскопки в существующих плагилах помогают
- 3 Нужно допиливать плагин для новой Android Studio
- 4 Против несовместимости — `intellij.localPath`

Подведем итоги



- 1 UI Inspector поможет найти существующие классы
- 2 Раскопки в существующих плагинах помогают**
- 3 Нужно допиливать плагин для новой Android Studio
- 4 Против несовместимости — `intellij.localPath`

Подведем итоги



- 1 UI Inspector поможет найти существующие классы
- 2 Раскопки в существующих плагилах помогают
- 3 Нужно допиливать плагин для новой Android Studio**
- 4 Против несовместимости — `intellij.localPath`

Подведем итоги



- 1 UI Inspector поможет найти существующие классы
- 2 Раскопки в существующих плагилах помогают
- 3 Нужно допиливать плагин для новой Android Studio
- 4 **Против несовместимости — `intellij.localPath`**

Есть вопросы?



- 1 UI Inspector поможет найти существующие классы
- 2 Раскопки в существующих плагилах помогают
- 3 Нужно допиливать плагин для новой Android Studio
- 4 Против несовместимости — `intellij.localPath`



Что делать

дальше?

Плагины уничтожают рутину

1

Разработка плагина — хороший способ ускорить свою команду

Плагины уничтожают рутину

1

Разработка плагина — хороший способ ускорить свою команду

2

Автоматизировать можно все что угодно

Плагины уничтожают рутину

- 1 Разработка плагина — хороший способ ускорить свою команду
- 2 Автоматизировать можно все что угодно
- 3 Изучайте чужие плагины, там много всего интересного**

Плагины уничтожают рутину

- 1 Разработка плагина — хороший способ ускорить свою команду
- 2 Автоматизировать можно все что угодно
- 3 Изучайте чужие плагины, там много всего интересного
- 4 Проблемы с плагинами есть, но не страшные**



**Форкните наш
проект на Гитхабе**

Спросите меня о чем-нибудь :-)

- 1 Генерация boilerplate-кода
- 2 Автоматизация взаимодействия с JIRA
- 3 Добавление инспекций кода
- 4 Tips & tricks в разработке плагина



Ссылка на слайды