mobius

Павел Ставицкий

Автоматизация миграции Android приложений на Bazel

Обо мне

- 6 лет в Android разработке
- Grab (Singapore)
- Миграция Grab Android приложения на Bazel

Павел Ставицкий

О чем доклад?

- Поговорим о Bazel

О чем доклад?

- Поговорим о Bazel
- Создадим с нуля Gradle плагин для миграции Android приложения на Bazel

О чем доклад?

- Поговорим о Bazel
- Создадим с нуля Gradle плагин для миграции Android приложения на Bazel
- Рассмотрим type-safe Kotlin DSL для генерации кода Bazel скриптов

Поговорим о Bazel

Что такое Bazel?

- Система сборки проектов (от Google)

Что такое Bazel?

- Система сборки проектов (от Google)
- Полиглот - сборка проектов на разных языках и платформах

Что такое Bazel?

- Система сборки проектов (от Google)
- Полиглот - сборка проектов на разных языках и платформах
- **Поддержка больших моно-репозиториев**

Преимущества Bazel

- Общая система сборки для большинства проектов в организации (включая Android, iOS, backend и др.)

Преимущества Bazel

- Общая система сборки для большинства проектов в организации (включая Android, iOS, backend и др.)
- **Одна команда, которая все это поддерживает**

Преимущества Bazel

- Общая система сборки для большинства проектов в организации (включая Android, iOS, backend и др.)
- Одна команда, которая все это поддерживает
- **Сетевой кэш**

Подробнее про билд-системы

youtu.be/Wd91Y8sRs2k

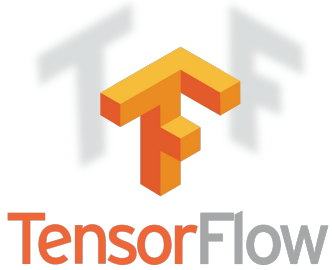
Артем Зиннатуллин — Android builds at Lyft

Преимущества Bazel

- Общая система сборки для большинства проектов в организации (включая Android, iOS, backend и др.)
- Одна команда, которая все это поддерживает
- Сетевой кэш
- **Кэш фазы анализа**

Кто использует Bazel?

Кто использует Bazel?



Когда мигрировать на Bazel?

- Проблемы со сборкой на Gradle (большой проект)

Когда мигрировать на Bazel?

- Проблемы со сборкой на Gradle (большой проект)
- **Правильная многомодульная архитектура**

Многомодульная архитектура

youtu.be/FMiFtsew2UY

Денис Неклюдов — Как не состариться во время сборки: Kapt и другие приключения

Многомодульная архитектура

youtu.be/FMiFtsew2UY

Денис Неклюдов — Как не состариться во время сборки: Kapt и другие приключения

youtu.be/pMEAD6jjbaI

Владимир Тагаков — Многомодульность и Dagger 2

Когда мигрировать на Bazel?

- Проблемы со сборкой на Gradle (большой проект)
- Правильная многомодульная архитектура
- **Убраны технологии, которые влияют на скорость сборки (data binding и пр. кодогенераторы)**

Как выглядит Bazel?

Пример Gradle скрипта

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
}

android {
    compileSdkVersion 30

    defaultConfig {
        applicationId "com.morfly.sample"
        minSdkVersion 21
        targetSdkVersion 30
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
}

dependencies {
    implementation "org.jetbrains.kotlin:kotlin-stdlib:1.5.10"
    implementation 'androidx.core:core-ktx:1.6.0'
    implementation 'androidx.appcompat:appcompat:1.3.1'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.0'
}
```

Пример Bazel скрипта

```
load("@io_bazel_rules_kotlin//kotlin:kotlin.bzl", "kt_android_library")
load("@rules_jvm_external//:defs.bzl", "artifact")

kt_android_library(
    name = "my_lib",
    srcs = glob([
        "src/main/java/**/*.kt",
        "src/main/kotlin/**/*.kt",
    ]),
    custom_package = "com.morfly.sample",
    manifest = "src/main/AndroidManifest.xml",
    resource_files = glob(["src/main/res/**"]),
    deps = [
        artifact("androidx.core:core-ktx"),
        artifact("androidx.appcompat:appcompat"),
        artifact("com.google.android.material:material"),
        artifact("androidx.constraintlayout:constraintlayout"),
    ],
)
```

Starlark язык

`docs.bazel.build/versions/main/skylark/language.html`

Starlark язык (официальная документация)

Компоненты Bazel скрипта

```
load("@io_bazel_rules_kotlin//kotlin:kotlin.bzl", "kt_android_library")
load("@rules_jvm_external//:defs.bzl", "artifact")

kt_android_library(
    name = "my_lib",
    srcs = glob([
        "src/main/java/**/*.kt",
        "src/main/kotlin/**/*.kt",
    ]),
    custom_package = "com.morfly.sample",
    manifest = "src/main/AndroidManifest.xml",
    resource_files = glob(["src/main/res/**"]),
    deps = [
        artifact("androidx.core:core-ktx"),
        artifact("androidx.appcompat:appcompat"),
        artifact("com.google.android.material:material"),
        artifact("androidx.constraintlayout:constraintlayout"),
    ],
)
```

Компоненты Bazel скрипта

```
load("@io_bazel_rules_kotlin//kotlin:kotlin.bzl", "kt_android_library")
load("@rules_jvm_external//:defs.bzl", "artifact")

kt_android_library(
    name = "my_lib",
    srcs = glob([
        "src/main/java/**/*.kt",
        "src/main/kotlin/**/*.kt",
    ]),
    custom_package = "com.morfly.sample",
    manifest = "src/main/AndroidManifest.xml",
    resource_files = glob(["src/main/res/**"]),
    deps = [
        artifact("androidx.core:core-ktx"),
        artifact("androidx.appcompat:appcompat"),
        artifact("com.google.android.material:material"),
        artifact("androidx.constraintlayout:constraintlayout"),
    ],
)
```

Компоненты Bazel скрипта

```
load("@io_bazel_rules_kotlin//kotlin:kotlin.bzl", "kt_android_library")
load("@rules_jvm_external//:defs.bzl", "artifact")

kt_android_library(
    name = "my_lib",
    srcs = glob([
        "src/main/java/**/*.kt",
        "src/main/kotlin/**/*.kt",
    ]),
    custom_package = "com.morfly.sample",
    manifest = "src/main/AndroidManifest.xml",
    resource_files = glob(["src/main/res/**"]),
    deps = [
        artifact("androidx.core:core-ktx"),
        artifact("androidx.appcompat:appcompat"),
        artifact("com.google.android.material:material"),
        artifact("androidx.constraintlayout:constraintlayout"),
    ],
)
```

Компоненты Bazel скрипта

```
load("@io_bazel_rules_kotlin//kotlin:kotlin.bzl", "kt_android_library")
load("@rules_jvm_external//:defs.bzl", "artifact")

kt_android_library(
    name = "my_lib",
    srcs = glob([
        "src/main/java/**/*.kt",
        "src/main/kotlin/**/*.kt",
    ]),
    custom_package = "com.morfly.sample",
    manifest = "src/main/AndroidManifest.xml",
    resource_files = glob(["src/main/res/**"]),
    deps = [
        artifact("androidx.core:core-ktx"),
        artifact("androidx.appcompat:appcompat"),
        artifact("com.google.android.material:material"),
        artifact("androidx.constraintlayout:constraintlayout"),
    ],
)
```

Компоненты Bazel скрипта

```
load("@io_bazel_rules_kotlin//kotlin:kotlin.bzl", "kt_android_library")
load("@rules_jvm_external//:defs.bzl", "artifact")

kt_android_library(
    name = "my_lib",
    srcs = glob([
        "src/main/java/**/*.kt",
        "src/main/kotlin/**/*.kt",
    ]),
    custom_package = "com.morfly.sample",
    manifest = "src/main/AndroidManifest.xml",
    resource_files = glob(["src/main/res/**"]),
    deps = [
        artifact("androidx.core:core-ktx"),
        artifact("androidx.appcompat:appcompat"),
        artifact("com.google.android.material:material"),
        artifact("androidx.constraintlayout:constraintlayout"),
    ],
)
```

Компоненты Bazel скрипта

```
load("@io_bazel_rules_kotlin//kotlin:kotlin.bzl", "kt_android_library")
load("@rules_jvm_external//:defs.bzl", "artifact")

kt_android_library(
    name = "my_lib",
    srcs = glob([
        "src/main/java/**/*.kt",
        "src/main/kotlin/**/*.kt",
    ]),
    custom_package = "com.morfly.sample",
    manifest = "src/main/AndroidManifest.xml",
    resource_files = glob(["src/main/res/**"]),
    deps = [
        artifact("androidx.core:core-ktx"),
        artifact("androidx.appcompat:appcompat"),
        artifact("com.google.android.material:material"),
        artifact("androidx.constraintlayout:constraintlayout"),
    ],
)
```

Компоненты Bazel скрипта

```
load("@io_bazel_rules_kotlin//kotlin:kotlin.bzl", "kt_android_library")
load("@rules_jvm_external//:defs.bzl", "artifact")

kt_android_library(
    name = "my_lib",
    srcs = glob([
        "src/main/java/**/*.kt",
        "src/main/kotlin/**/*.kt",
    ]),
    custom_package = "com.morfly.sample",
    manifest = "src/main/AndroidManifest.xml",
    resource_files = glob(["src/main/res/**"]),
    deps = [
        artifact("androidx.core:core-ktx"),
        artifact("androidx.appcompat:appcompat"),
        artifact("com.google.android.material:material"),
        artifact("androidx.constraintlayout:constraintlayout"),
    ],
)
```

Компоненты Bazel скрипта

```
load("@io_bazel_rules_kotlin//kotlin:kotlin.bzl", "kt_android_library")
load("@rules_jvm_external//:defs.bzl", "artifact")

kt_android_library(
    name = "my_lib",
    srcs = glob([
        "src/main/java/**/*.kt",
        "src/main/kotlin/**/*.kt",
    ]),
    custom_package = "com.morfly.sample",
    manifest = "src/main/AndroidManifest.xml",
    resource_files = glob(["src/main/res/**"]),
    deps = [
        artifact("androidx.core:core-ktx"),
        artifact("androidx.appcompat:appcompat"),
        artifact("com.google.android.material:material"),
        artifact("androidx.constraintlayout:constraintlayout"),
    ],
)
```

Компоненты Bazel скрипта

```
load("@io_bazel_rules_kotlin//kotlin:kotlin.bzl", "kt_android_library")
load("@rules_jvm_external//:defs.bzl", "artifact")

kt_android_library(
    name = "my_lib",
    srcs = glob([
        "src/main/java/**/*.kt",
        "src/main/kotlin/**/*.kt",
    ]),
    custom_package = "com.morfly.sample",
    manifest = "src/main/AndroidManifest.xml",
    resource_files = glob(["src/main/res/**"]),
    deps = [
        artifact("androidx.core:core-ktx"),
        artifact("androidx.appcompat:appcompat"),
        artifact("com.google.android.material:material"),
        artifact("androidx.constraintlayout:constraintlayout"),
    ],
)
```

Компоненты Bazel скрипта

```
load("@io_bazel_rules_kotlin//kotlin:kotlin.bzl", "kt_android_library")
load("@rules_jvm_external//:defs.bzl", "artifact")

kt_android_library(
    name = "my_lib",
    srcs = glob([
        "src/main/java/**/*.kt",
        "src/main/kotlin/**/*.kt",
    ]),
    custom_package = "com.morfly.sample",
    manifest = "src/main/AndroidManifest.xml",
    resource_files = glob(["src/main/res/**"]),
    deps = [
        artifact("androidx.core:core-ktx"),
        artifact("androidx.appcompat:appcompat"),
        artifact("com.google.android.material:material"),
        artifact("androidx.constraintlayout:constraintlayout"),
    ],
)
```

Подробнее про Bazel для Android

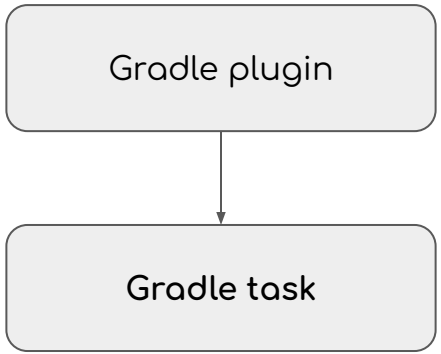
youtu.be/5xOOit9b7Yc

Степан Гончаров — Bazel для Android разработчиков

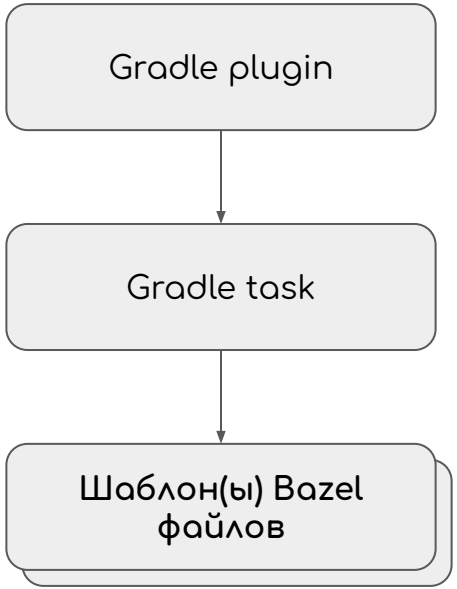
Пишем свой Gradle plugin для автоматизации миграции на Bazel

Создание нашего Gradle plugin

Создание нашего Gradle plugin



Создание нашего Gradle plugin



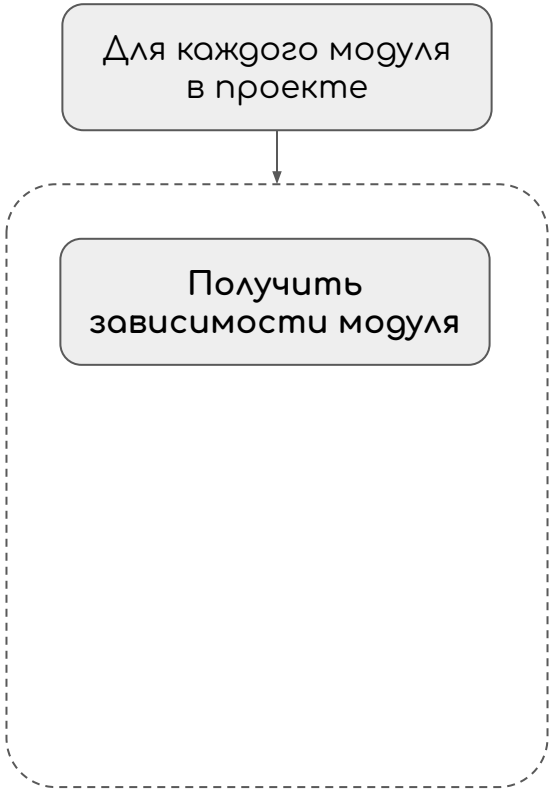
Gradle task

```
./gradlew migrateToBazel
```

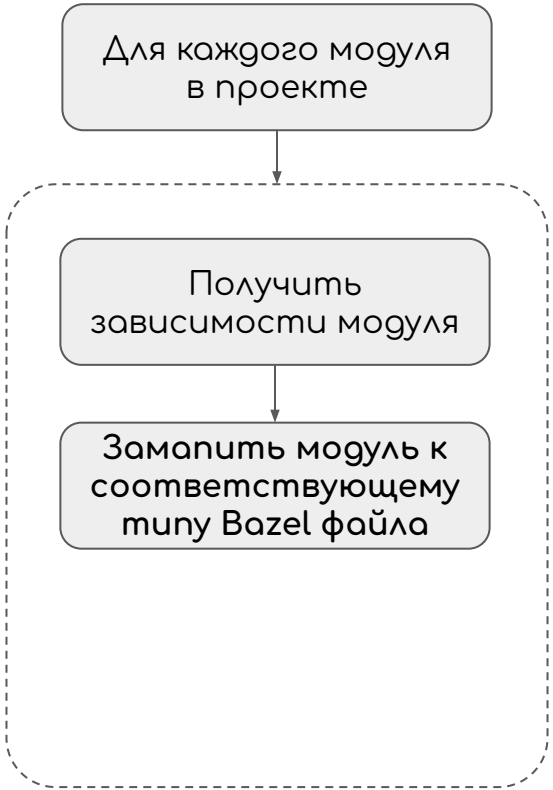
Что делает Gradle task?

Для каждого модуля в проекте

Что делает Gradle task?



Что делает Gradle task?



Типы Bazel модулей

- Kotlin Android модуль — `kt_android_library`

Типы Bazel модулей

- Kotlin Android модуль
- Pure Java Android модуль — `android_library`

Типы Bazel модулей

- Kotlin Android модуль
- Pure Java Android модуль
- **Kotlin library модуль** — `kt_jvm_library`

Типы Bazel модулей

- Kotlin Android модуль
- Pure Java Android модуль
- Kotlin library модуль
- **Pure Java library модуль** — `java_library`

...

Дополнительные Bazel файлы

только в одном экземпляре

- WORKSPACE файл

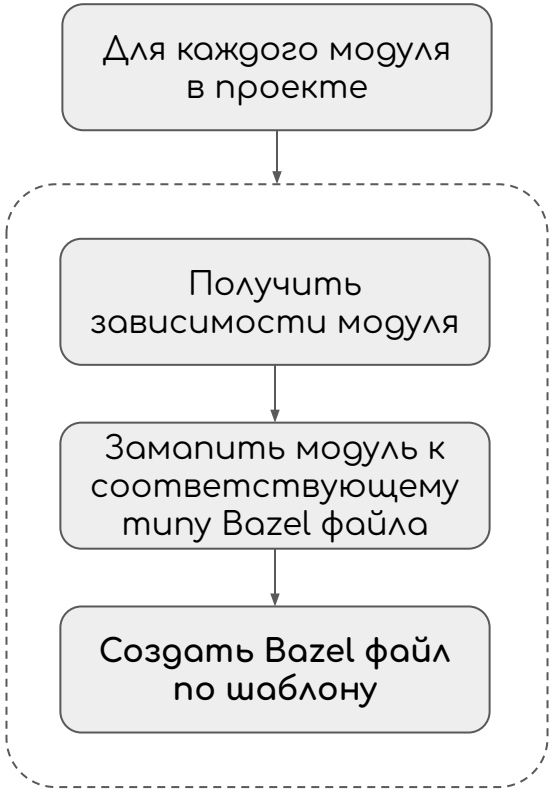
Дополнительные Bazel файлы

только в одном экземпляре

- WORKSPACE файл
- Root BUILD файл

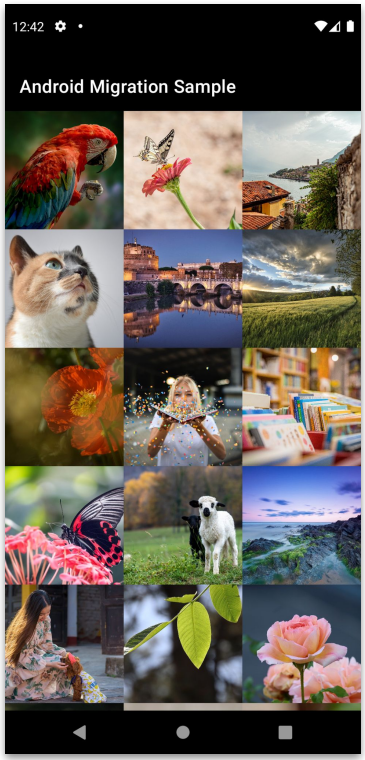
...

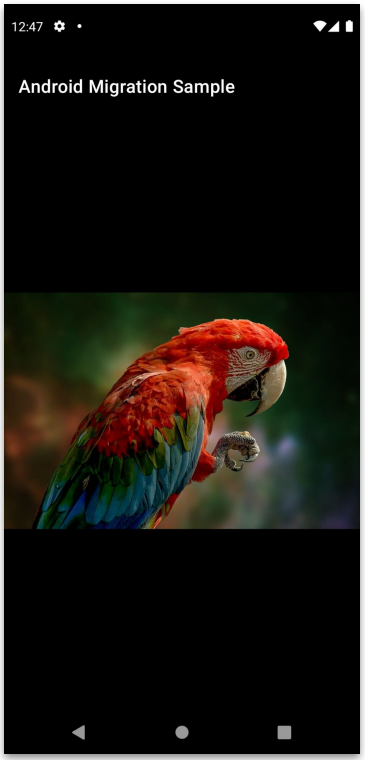
Что делает Gradle task?



Coding time...

Android приложение





Создаем Gradle plugin

```
import org.gradle.api.Plugin
import org.gradle.api.Project
import org.gradle.kotlin.dsl.register


class MigrationPlugin : Plugin<Project> {

    override fun apply(target: Project) {

        target.tasks.register<MigrateToBazelTask>("migrateToBazel")
    }
}
```

Создаем Gradle plugin

```
import org.gradle.api.Plugin
import org.gradle.api.Project
import org.gradle.kotlin.dsl.register


class MigrationPlugin : Plugin<Project> {

    override fun apply(target: Project) {

        target.tasks.register<MigrateToBazelTask>("migrateToBazel")
    }
}
```

Регистрируем Gradle task

```
import org.gradle.api.Plugin
import org.gradle.api.Project
import org.gradle.kotlin.dsl.register


class MigrationPlugin : Plugin<Project> {

    override fun apply(target: Project) {

        target.tasks.register<MigrateToBazelTask>("migrateToBazel")
    }
}
```

Регистрируем Gradle task

```
import org.gradle.api.Plugin
import org.gradle.api.Project
import org.gradle.kotlin.dsl.register


class MigrationPlugin : Plugin<Project> {

   override fun apply(target: Project) {

       target.tasks.register<MigrateToBazelTask>("migrateToBazel")
   }
}
```

Регистрируем Gradle task

```
import org.gradle.api.Plugin
import org.gradle.api.Project
import org.gradle.kotlin.dsl.register


class MigrationPlugin : Plugin<Project> {

   override fun apply(target: Project) {

       target.tasks.register<MigrateToBazelTask>("migrateToBazel")
   }
}
```

Регистрируем Gradle task

```
./gradlew migrateToBazel
```

Регистрируем Gradle task

```
// root build.gradle.kts file

apply<MigrationPlugin>()
```

Пишем Gradle task

```
import org.gradle.api.DefaultTask
import org.gradle.api.tasks.TaskAction


open class MigrateToBazelTask : DefaultTask() {

    @TaskAction
    fun migrateToBazel() {

        ...
    }
}
```

Demo time...

Type-safe DSL для генерации Bazel скриптов

Шаблон Bazel файла

```
fun android_library_template(
    name: String,
    packageName: String,
    artifactDeps: String,
    moduleDeps: String

) = """
load("@io_bazel_rules_kotlin//kotlin:kotlin.bzl", "kt_android_library")
load("@rules_jvm_external//:defs.bzl", "artifact")

kt_android_library(
    name = "$name",
    srcs = glob(["src/main/java/**/*.kt"]),
    resource_files = glob(["src/main/res/**"]),
    custom_package = "$packageName",
    manifest = "src/main/AndroidManifest.xml",
    visibility = ["//visibility:public"],
    deps = [
        "//:dagger",
$moduleDeps,
$artifactDeps
    ],
)
"""
```

Type-safe шаблон Bazel файла

```
fun android_library_build(
    name: String,
    packageName: String,
    artifactDeps: List<String>,
    moduleDeps: List<String>

) = BUILD.bazel {

    load("@io_bazel_rules_kotlin//kotlin:kotlin.bzl", "kt_android_library")
    load("@rules_jvm_external//:defs.bzl", "artifact")

    kt_android_library(
        name = name,
        srcs = glob("src/main/java/**/*.kt"),
        resource_files = glob("src/main/res/**"),
        custom_package = packageName,
        manifest = "src/main/AndroidManifest.xml",
        visibility = list["//visibility:public"],
        deps = moduleDeps + artifactDeps.map { artifact(it) }
    )
}
```

Как войти в контекст Starlark DSL?

```
BUILD.bazel {

    ...
}
```

Обозначение фрагментов кода далее

Kotlin (Starlark DSL)

Starlark (Bazel script)

Компоненты Starlark DSL

`load` statement

```
load("@io_bazel_rules_kotlin//kotlin:kotlin.bzl", "kt_android_library")
```

Компоненты Starlark DSL

`load` statement

```
BUILD.bazel {

   load("@io_bazel_rules_kotlin//kotlin:kotlin.bzl", "kt_android_library")

   ...
}
```

Компоненты Starlark DSL

Присвоение переменных

```
NAME = "app"
```

Компоненты Starlark DSL

Присвоение переменных

```
BUILD.bazel {

    val NAME by "app"

    ...



}
```

Компоненты Starlark DSL

Присвоение переменных

```
NAME = "app"

SRCS = ["MainActivity.java", "MainViewModel.java"]
```

Компоненты Starlark DSL

Присвоение переменных

```
BUILD.bazel {

   val NAME by "app"

   val SRCS by list["MainActivity.java", "MainViewModel.java"]

   ...

}
```

Компоненты Starlark DSL

Присвоение переменных

```
NAME = "app"

SRCS = ["MainActivity.java", "MainViewModel.java"]

MANIFEST_VALUES = {"minSdkVersion": "21"}
```

Компоненты Starlark DSL

Присвоение переменных

```
BUILD.bazel {
    val NAME by "app"

    val SRCS by list["MainActivity.java", "MainViewModel.java"]

    val MANIFEST_VALUES by dict { "minSdkVersion" to "21" }
    ...
}
```

Компоненты Starlark DSL

Конкатенация

```
SRCS = ["MainActivity.java"] + ["ViewModel.java"]
```

Компоненты Starlark DSL

Конкатенация

```
BUILD.bazel {

   val SRCS by list["MainActivity.java"] `+` list["MainViewModel.java"]

   ...
}
```

Компоненты Starlark DSL

Конкатенация

```
BUILD.bazel {

   val SRCS by list["MainActivity.java"] `+` list["MainViewModel.java"]

   ...
}
```

Компоненты Starlark DSL

Конкатенация

```
BUILD.bazel {

    val SRCS: List<String> by list["MainActivity.java"] `+` "MainViewModel.java"

    ...
}
```

Компоненты Starlark DSL

Конкатенация

```
BUILD.bazel {

    val SRCS: List<String> by list["MainActivity.java"] `+` "MainViewModel.java"

                                                        Compilation error:
                                                         Required: List
                                                         Found: String
    ...
}
```

Компоненты Starlark DSL

List comprehensions

```
[i + i for i in LIST]
```

Компоненты Starlark DSL

List comprehension синтаксис

```
[<expression> for <item> in <list>]
```

Компоненты Starlark DSL

List comprehensions

```
BUILD.bazel {

   "i" `in` LIST take { it `+` it }

   ...
}
```

Компоненты Starlark DSL

List comprehensions

```
BUILD.bazel {

   "i" `in` LIST take { it `+` it }
    <item>

   ...
}
```

Компоненты Starlark DSL

List comprehensions

```
BUILD.bazel {

   "i" `in` LIST take { it `+` it }
        <list>

   ...
}
```

Компоненты Starlark DSL

List comprehensions

```
BUILD.bazel {

   "i" `in` LIST take { it `+` it }
                      <expression>

   ...
}
```

Компоненты Starlark DSL

List comprehensions

```
[i + i for i in LIST]
```

Компоненты Starlark DSL

Build rules и функции

```
SRCS = ["MainActivity.java"]

android_binary(
    name = "app",
    srcs = SRCS + ["utils/StringUtils.java"]
)
```

Компоненты Starlark DSL

Build rules и функции

```
BUILD.bazel {

   val SRCS by list["MainActivity.java"]

   android_binary(
      name = "app",
      srcs = SRCS `+` list["utils/StringUtils.java"]
   )

   ...
}
```

Компоненты Starlark DSL

Build rules и функции

```
BUILD.bazel {

   val SRCS by list["MainActivity.java"]

   android_binary(
      name = "app",
      srcs = SRCS `+` glob("utils/**/*.java")
   )

   ...
}
```

Компоненты Starlark DSL

Build rules и функции

```
BUILD.bazel {

   val SRCS by list["MainActivity.java"]

   android_binary {
      name = "app"
      srcs = SRCS `+` glob("src/main/java/**/*.java")
      "manifest_values" `=` { "minSdkVersion" to "21" }
   }

   ...
}
```

Компоненты Starlark DSL

Форматирование строк

```
"androidx.compose.ui:ui:%s" % "1.0.0"

...
```

Компоненты Starlark DSL

Форматирование строк

```
"androidx.compose.ui:ui:%s" % "1.0.0"

"androidx.compose.ui:ui:{v}".format(v = "1.0.0")
```

Компоненты Starlark DSL

Форматирование строк

```
BUILD.bazel {

   "androidx.compose.ui:ui:%s" `%` "1.0.0"


   ...


}
```

Компоненты Starlark DSL

Форматирование строк

```
BUILD.bazel {

    "androidx.compose.ui:ui:%s" `%` "1.0.0"

    "androidx.compose.ui:ui:{v}".format { "v" `=` "1.0.0" }

    ...
}
```

Компоненты Starlark DSL

Объединяя все вместе...

```
VIEW_MODELS_WITH_RES_IMPORTS = ["MainViewModel.kt", ...]

[
    genrule(
        name = "modify_imports_in_" + file[0:-3],
        srcs = [file],
        outs = [file[0:-3] + "_synthetic.kt"],
        cmd = """
        cat $(SRCS) |
        sed 's/import com.morfly.R/import com.morfly.viewmodels.R/g' > $(OUTS)
        """,
    )
    for file in VIEW_MODELS_WITH_RES_IMPORTS
]

APP_FILES = [
    "modify_imports_in_" + file[0:-3]
    for file in VIEW_MODELS_WITH_RES_IMPORTS
]
```

Компоненты Starlark DSL

Объединяя все вместе...

```
VIEW_MODELS_WITH_RES_IMPORTS = ["MainViewModel.kt", ...]

[
    genrule(
        name = "modify_imports_in_" + file[0:-3],
        srcs = [file],
        outs = [file[0:-3] + "_synthetic.kt"],
        cmd = """
        cat $(SRCS) |
        sed 's/import com.morfly.R/import com.morfly.viewmodels.R/g' > $(OUTS)
        """,
    )
    for file in VIEW_MODELS_WITH_RES_IMPORTS
]

APP_FILES = [
    "modify_imports_in_" + file[0:-3]
    for file in VIEW_MODELS_WITH_RES_IMPORTS
]
```

Компоненты Starlark DSL

Объединяя все вместе...

```
VIEW_MODELS_WITH_RES_IMPORTS = ["MainViewModel.kt", ...]

[
    genrule(
        name = "modify_imports_in_" + file[0:-3],
        srcs = [file],
        outs = [file[0:-3] + "_synthetic.kt"],
        cmd = """
        cat $(SRCS) |
        sed 's/import com.morfly.R/import com.morfly.viewmodels.R/g' > $(OUTS)
        """,
    )
    for file in VIEW_MODELS_WITH_RES_IMPORTS
]

APP_FILES = [
    "modify_imports_in_" + file[0:-3]
    for file in VIEW_MODELS_WITH_RES_IMPORTS
]
```

Компоненты Starlark DSL

Объединяя все вместе...

```
VIEW_MODELS_WITH_RES_IMPORTS = ["MainViewModel.kt", ...]

[
    genrule(
        name = "modify_imports_in_" + file[0:-3],
        srcs = [file],
        outs = [file[0:-3] + "_synthetic.kt"],
        cmd = """
        cat $(SRCS) |
        sed 's/import com.morfly.R/import com.morfly.viewmodels.R/g' > $(OUTS)
        """,
    )
    for file in VIEW_MODELS_WITH_RES_IMPORTS
]

APP_FILES = [
    "modify_imports_in_" + file[0:-3]
    for file in VIEW_MODELS_WITH_RES_IMPORTS
]
```

Компоненты Starlark DSL

Объединяя все вместе...

```
VIEW_MODELS_WITH_RES_IMPORTS = ["MainViewModel.kt", ...]

[
    genrule(
        name = "modify_imports_in_" + file[0:-3],
        srcs = [file],
        outs = [file[0:-3] + "_synthetic.kt"],
        cmd = """
        cat $(SRCS) |
        sed 's/import com.morfly.R/import com.morfly.viewmodels.R/g' > $(OUTS)
        """,
    )
    for file in VIEW_MODELS_WITH_RES_IMPORTS
]

APP_FILES = [
    "modify_imports_in_" + file[0:-3]
    for file in VIEW_MODELS_WITH_RES_IMPORTS
]
```

Компоненты Starlark DSL

Объединяя все вместе...

```
"MainViewModel.kt"


    file[0:-3] → "MainViewModel"
```

Компоненты Starlark DSL

Объединяя все вместе...

```
VIEW_MODELS_WITH_RES_IMPORTS = ["MainViewModel.kt", ...]

[
    genrule(
        name = "modify_imports_in_" + file[0:-3],
        srcs = [file],
        outs = [file[0:-3] + "_synthetic.kt"],
        cmd = """
        cat $(SRCS) |
        sed 's/import com.morfly.R/import com.morfly.viewmodels.R/g' > $(OUTS)
        """,
    )
    for file in VIEW_MODELS_WITH_RES_IMPORTS
]

APP_FILES = [
    "modify_imports_in_" + file[0:-3]
    for file in VIEW_MODELS_WITH_RES_IMPORTS
]
```

Компоненты Starlark DSL

Объединяя все вместе...

```
VIEW_MODELS_WITH_RES_IMPORTS = ["MainViewModel.kt", ...]

[
    genrule(
        name = "modify_imports_in_" + file[0:-3],
        srcs = [file],
        outs = [file[0:-3] + "_synthetic.kt"],
        cmd = """
        cat $(SRCS) |
        sed 's/import com.morfly.R/import com.morfly.viewmodels.R/g' > $(OUTS)
        """,
    )
    for file in VIEW_MODELS_WITH_RES_IMPORTS
]

APP_FILES = [
    "modify_imports_in_" + file[0:-3]
    for file in VIEW_MODELS_WITH_RES_IMPORTS
]
```

Компоненты Starlark DSL

Объединяя все вместе...

```
VIEW_MODELS_WITH_RES_IMPORTS = ["MainViewModel.kt", ...]

[
    genrule(
        name = "modify_imports_in_" + file[0:-3],
        srcs = [file],
        outs = [file[0:-3] + "_synthetic.kt"],
        cmd = """
        cat $(SRCS) |
        sed 's/import com.morfly.R/import com.morfly.viewmodels.R/g' > $(OUTS)
        """,
    )
    for file in VIEW_MODELS_WITH_RES_IMPORTS
]

APP_FILES = [
    "modify_imports_in_" + file[0:-3]
    for file in VIEW_MODELS_WITH_RES_IMPORTS
]
```

Компоненты Starlark DSL

Объединяя все вместе...

```
fun android_library_build(
   packageName: String,
   ...

) = BUILD {

   val VIEW_MODELS_WITH_RES_IMPORTS by list["MainViewModel.kt", ...]

   "file" `in` VIEW_MODELS_WITH_RES_IMPORTS take { file →

      genrule(
         name = "modify_imports_in_" `+` file[0..-3],
         srcs = list[file],
         outs = list[file[0..-3] `+` "_synthetic.kt"],
         cmd = """
         cat $(SRCS) |
         sed 's/import $packageName.R/import $packageName.viewmodels.R/g' > $(OUTS)
         """.trimIndent()
      )
   }

   val APP_FILES by ("file" `in` VIEW_MODELS_WITH_RES_IMPORTS take {
      "modify_imports_in_" `+` it[0..-3]
   })
}
```

Компоненты Starlark DSL

Объединяя все вместе...

```
fun android_library_build(
   packageName: String,
   ...

) = BUILD {

   val VIEW_MODELS_WITH_RES_IMPORTS by list["MainViewModel.kt", ...]

   "file" `in` VIEW_MODELS_WITH_RES_IMPORTS take { file →

      genrule(
         name = "modify_imports_in_" `+` file[0..-3],
         srcs = list[file],
         outs = list[file[0..-3] `+` "_synthetic.kt"],
         cmd = """
         cat $(SRCS) |
         sed 's/import $packageName.R/import $packageName.viewmodels.R/g' > $(OUTS)
         """.trimIndent()
      )
   }

   val APP_FILES by ("file" `in` VIEW_MODELS_WITH_RES_IMPORTS take {
      "modify_imports_in_" `+` it[0..-3]
   })
}
```

Компоненты Starlark DSL

Объединяя все вместе...

```
fun android_library_build(
   packageName: String,
   ...

) = BUILD {

   val VIEW_MODELS_WITH_RES_IMPORTS by list["MainViewModel.kt", ...]

   "file" `in` VIEW_MODELS_WITH_RES_IMPORTS take { file →

      genrule(
         name = "modify_imports_in_" `+` file[0..-3],
         srcs = list[file],
         outs = list[file[0..-3] `+` "_synthetic.kt"],
         cmd = """
         cat $(SRCS) |
         sed 's/import $packageName.R/import $packageName.viewmodels.R/g' > $(OUTS)
         """.trimIndent()
      )
   }

   val APP_FILES by ("file" `in` VIEW_MODELS_WITH_RES_IMPORTS take {
      "modify_imports_in_" `+` it[0..-3]
   })
}
```

Итоги

Итоги

Компоненты для авто-миграции на Bazel

- Gradle Plugin

Итоги

Компоненты для авто-миграции на Bazel

- Gradle Plugin
- DSL для type-safe генерации Starlark кода

С чего начать?

github.com/morfly/
android-bazel-migration-sample

Пример проекта из доклада на GitHub

С чего начать?

github.com/morfly/
android-bazel-migration-sample

Пример проекта из доклада на GitHub

github.com/morfly/airin

Инструмент для авто-миграции на Bazel - Airin