

Как Gradle может сделать разработку Android приложений проще

Даниил
Попов

Android
Team Lead

Mail.ru
Group

Коротко обо мне

- В Android разработке с 2012 года
- Занимался мобильными играми
- Разрабатывал SDK и библиотеки
- Делаю мессенджеры в Mail.ru Group



Коротко о Gradle



- Появился в 2007 году
- Предоставляет DSL на Groovy
- Можно собирать не только Java проекты
- Имеет встроенные средства для профилирования и отладки

О чём будем говорить

- Где живут таски и плагины
- Какие задачи они могут решать
- Best Practices
- И кое-что еще

Где живут таски и плагины

Где живут таски и плагины

- Непосредственно в build.gradle
- В другом скрипте (apply from)
- В директории buildSrc
- В отдельном проекте

Непосредственно в build.gradle

Непосредственно в build.gradle

```
class InScriptTask extends DefaultTask {  
    @TaskAction  
    def sayHello() {  
        println 'Hello from script task'  
    }  
  
    tasks.register("helloTask", InScriptTask)
```

Непосредственно в build.gradle

```
class InScriptTask extends DefaultTask {  
    @TaskAction  
    def sayHello() {  
        println 'Hello from script task'  
    }  
}  
  
tasks.register("helloTask", InScriptTask)
```

Непосредственно в build.gradle

```
class InScriptPlugin implements Plugin<Project> {
    void apply(Project project) {
        project.task('helloInline') {
            doLast {
                println 'I live in build.gradle'
            }
        }
    }
}

apply plugin: InScriptPlugin
apply plugin: 'com.android.application'

android { ... }
```

Непосредственно в build.gradle

```
class InScriptPlugin implements Plugin<Project> {
    void apply(Project project) {
        project.task('helloInline') {
            doLast {
                println 'I live in build.gradle'
            }
        }
    }
}

apply plugin: InScriptPlugin
apply plugin: 'com.android.application'

android { ... }
```

Непосредственно в build.gradle

Плюсы

- Быстро

Минусы

- Громоздко
- Нет переиспользования
- Нет версионирования
- Нельзя протестировать

В другом скрипте

В другом скрипте

```
apply from: '../gradle/custom/ApplyFromPlugin.gradle'  
apply from: 'https://example.com/script.gradle'  
apply plugin: 'com.android.application'  
  
android { ... }
```

В другом скрипте

Плюсы

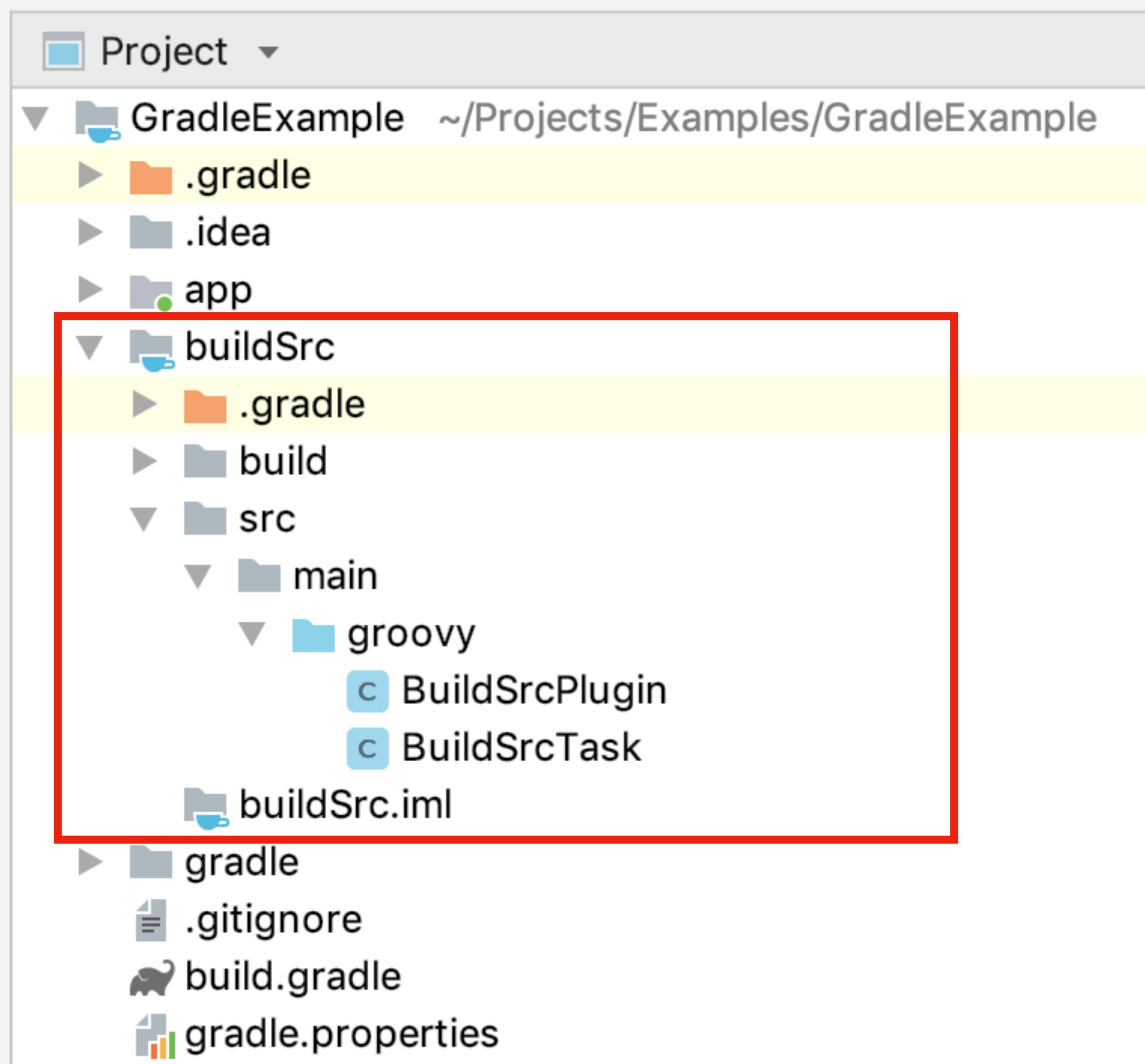
- Быстро
- Не громоздко
- Переиспользование

Минусы

- Нет версионирования
- Нельзя протестировать
- Нужно явно указывать путь

В директории buildSrc

В директории buildSrc



В директории buildSrc

```
class BuildSrcTask extends DefaultTask {  
    @TaskAction  
    def hello() {  
        println 'Hello from buildSrc'  
    }  
}
```

В директории buildSrc

```
class BuildSrcPlugin implements Plugin<Project> {
    @Override
    void apply(final Project project) {
        project.task('helloBuildSrc') {
            doLast {
                println('Hello from buildSrc plugin')
            }
        }
    }
}
```

В директории buildSrc

```
apply plugin: BuildSrcPlugin
apply plugin: 'com.android.application'

android { ... }

dependencies { ... }

tasks.register("helloTask", BuildSrcTask)
```

В директории buildSrc

Плюсы

- Быстро
- Не громоздко
- Переиспользование
- Можно протестировать

Минусы

- Нет версионирования

В отдельном проекте

В отдельном проекте

```
buildscript {  
    repositories {  
        maven {  
            url 'http://my-artifactory/artifactory/libs-release'  
        }  
        jcenter()  
        mavenLocal()  
        google()  
    }  
    dependencies { ... }  
}  
  
apply plugin: OtherProjectPlugin
```

В отдельном проекте

```
buildscript {  
    repositories {  
        maven {  
            url 'http://my-artifactory/artifactory/libs-release'  
        }  
        jcenter()  
        mavenLocal()  
        google()  
    }  
    dependencies { ... }  
}  
  
apply plugin: OtherProjectPlugin
```

В отдельном проекте

Плюсы

- Не громоздко
- Переиспользование
- Есть версионирование
- Можно протестировать

Минусы

- Правки в нескольких проектах

Что делают task'и

Возможности

- Генерация кода/ресурсов
- Проверки кода/ресурсов
- Кастомные проверки
- И т. д.

Генерация ресурсов

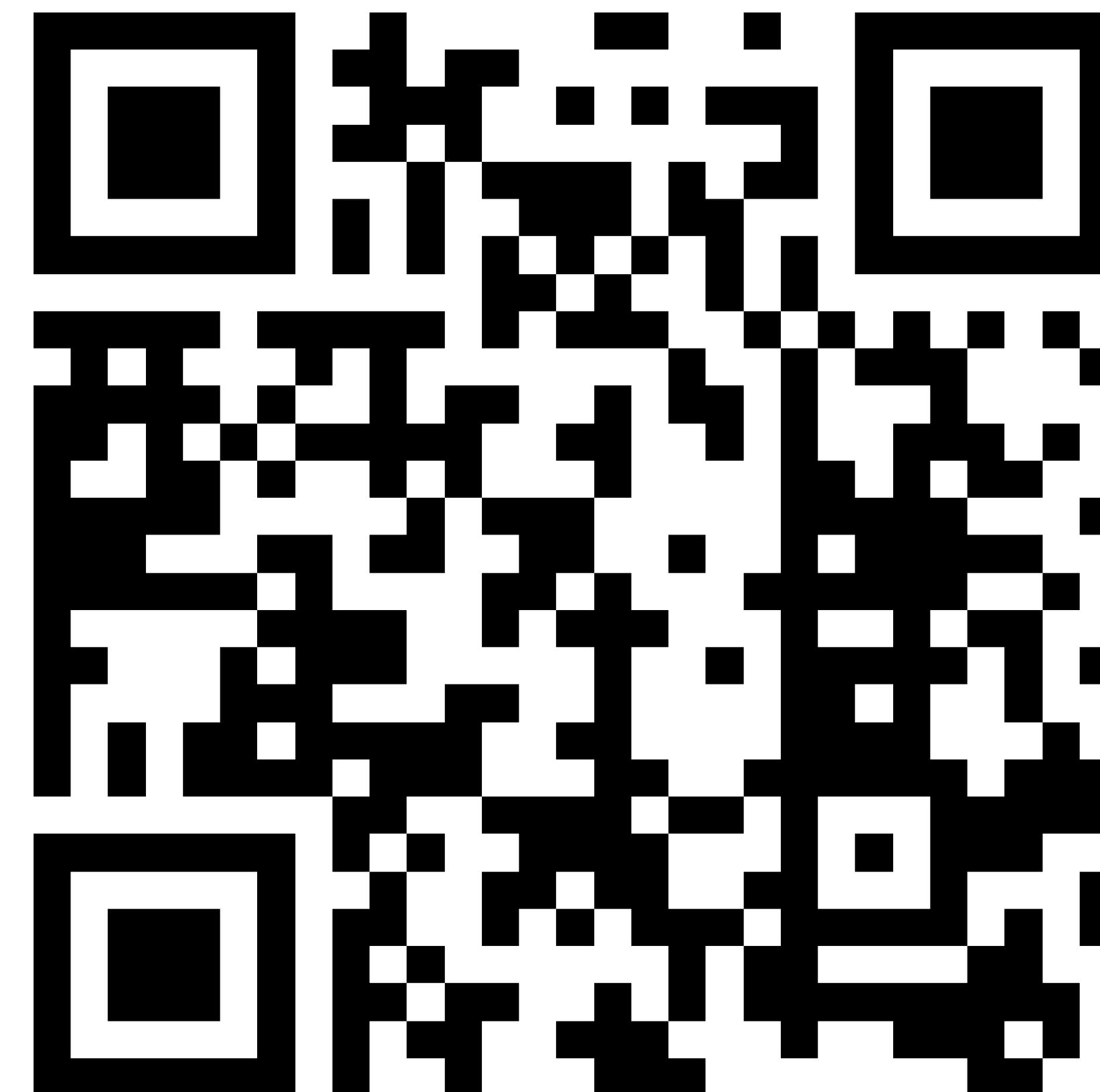
Генерация ресурсов

```
afterEvaluate {  
    android.applicationVariants.all { variant ->  
        variant.mergeResources.dependsOn tasks.generateResources  
    }  
}
```

Генерация ресурсов

```
android {  
    sourceSets {  
        main.res.srcDirs += ["${project.buildDir}/generated/translations",  
                            "${project.buildDir}/generated/themes"]  
    }  
}
```

Темизация приложения



Статические анализаторы

Статические анализаторы

```
import org.gradle.api.plugins.quality.Checkstyle

class MyCheckstyle extends Checkstyle {

    MyCheckstyle() {
        include '**/*.java'
        reports {
            xml.setDestination new File(project.buildDir,
                'reports/checkstyle/main.xml')
            html.setDestination new File(project.buildDir,
                'reports/checkstyle/main.html')
        }
        classpath = project.files()
        configFile = project.rootProject.file('checks/checkstyle.xml')
    }
}
```

Статические анализаторы

```
task checkstyle(type: MyCheckstyle) {  
    source 'src/main/java'  
}
```

```
task pmd(type: MyPmd) {  
    source 'src/main/java'  
}
```

Статические анализаторы

```
task checkstyle(type: MyCheckstyle) {  
    source 'src/main/java'  
}
```

```
task pmd(type: MyPmd) {  
    source 'src/main/java'  
}
```

Кастомные проверки

Кастомные проверки

- Проверка готовности кода к релизу
- Проверка зависимостей
- Проверка переводов
- Проверка дубликатов строк
- И т. д.

Готовность к релизу

Готовность к релизу

```
if (originalUrl == null) {  
    // Log more info about this case to find the problem  
    // TODO! https://jira.mail.ru/browse/IMA-15884  
    logInfoAboutNullableSnippetUrl(originalUrl);  
}
```

Готовность к релизу

```
task checkReleaseTodo {  
    doLast {  
        final def tag = 'TODO!'  
        def tagCount = 0  
        def fileCount = 0  
        rootDir.traverse(type: FileType.FILES, nameFilter: ~/.*/.(?:java|xml)/) {  
            final def count = it.getText('UTF-8').count(tag)  
            if (count > 0) {  
                tagCount += count  
                fileCount++  
            }  
        }  
        if (tagCount > 0) {  
            throw new GradleException("Found $tagCount $tag in $fileCount file(s)")  
        }  
    }  
}
```

Не даем собрать релиз

```
afterEvaluate {  
    tasks.compileStoreReleaseSources.dependsOn tasks.checkReleaseTodo  
    tasks.compileStoreReleaseSources.dependsOn tasks.checkStrings  
}
```

Что делают плагины

Возможности

1. Группировка других плагинов и их конфигурация
2. Объединение и конфигурация тасок
3. Настройка зависимостей проекта
4. И т. д.

Группировка плагинов

```
class AndroidAppPlugin implements Plugin<Project> {  
  
    @Override  
    void apply(final Project project) {  
        project.pluginManager.apply('com.android.application')  
        project.pluginManager.apply('kotlin-android')  
        project.pluginManager.apply('checkstyle')  
        project.pluginManager.apply('pmd')  
    }  
}
```

Объединение задач

```
class AndroidAppPlugin implements Plugin<Project> {

    @Override
    void apply(final Project project) {
        project.pluginManager.apply('checkstyle')

        project.checkstyle {
            toolVersion = '8.7'
        }

        def task = project.task('runCheckstyle', type: MyCheckstyle) {
            source 'src/main/java'
        }
        project.tasks.check.dependsOn task
    }
}
```

Объединение задач

```
class AndroidAppPlugin implements Plugin<Project> {

    @Override
    void apply(final Project project) {
        project.pluginManager.apply('checkstyle')

        project.checkstyle {
            toolVersion = '8.7'
        }

        def task = project.task('runCheckstyle', type: MyCheckstyle) {
            source 'src/main/java'
        }
        project.tasks.check.dependsOn task
    }
}
```

Объединение задач

```
class AndroidAppPlugin implements Plugin<Project> {

    @Override
    void apply(final Project project) {
        project.pluginManager.apply('checkstyle')

        project.checkstyle {
            toolVersion = '8.7'
        }

        def task = project.task('runCheckstyle', type: MyCheckstyle) {
            source 'src/main/java'
        }
        project.tasks.check.dependsOn task
    }
}
```

Объединение задач

```
class AndroidAppPlugin implements Plugin<Project> {

    @Override
    void apply(final Project project) {
        project.pluginManager.apply('checkstyle')

        project.checkstyle {
            toolVersion = '8.7'
        }

        def task = project.task('runCheckstyle', type: MyCheckstyle) {
            source 'src/main/java'
        }
        project.tasks.check.dependsOn task
    }
}
```

Настройка зависимостей

```
class AndroidAppPlugin implements Plugin<Project> {

    @Override
    void apply(final Project project) {
        project.pluginManager.apply('com.android.application')
        project.dependencies {
            implementation 'com.android.support:appcompat-v7:28.0.0'
            implementation 'com.squareup.okhttp3:okhttp:3.14.0'
            implementation 'com.google.code.gson:gson:2.8.5'
        }
    }
}
```

Best Practices

Использование сети

Плохоходить в сеть

- При конфигурировании
- При выполнении задач сборки



Incremental task

Incremental task

```
class MyIncrementalTask extends DefaultTask {  
    @InputDirectory  
    File sourceDirectory = project.file('src/main/java')  
  
    @TaskAction  
    void doWork(final IncrementalTaskInputs inputs) {  
        if (inputs.incremental) {  
            println 'Do incremental work'  
            inputs.outOfDate({ changed ->  
                println "File '${changed.file.absolutePath}' changed"  
            })  
            inputs.removed({ removed ->  
                println "File '${removed.file.absolutePath}' removed"  
            })  
        } else {  
            println 'Do all work'  
        }  
    }  
}
```

Incremental task

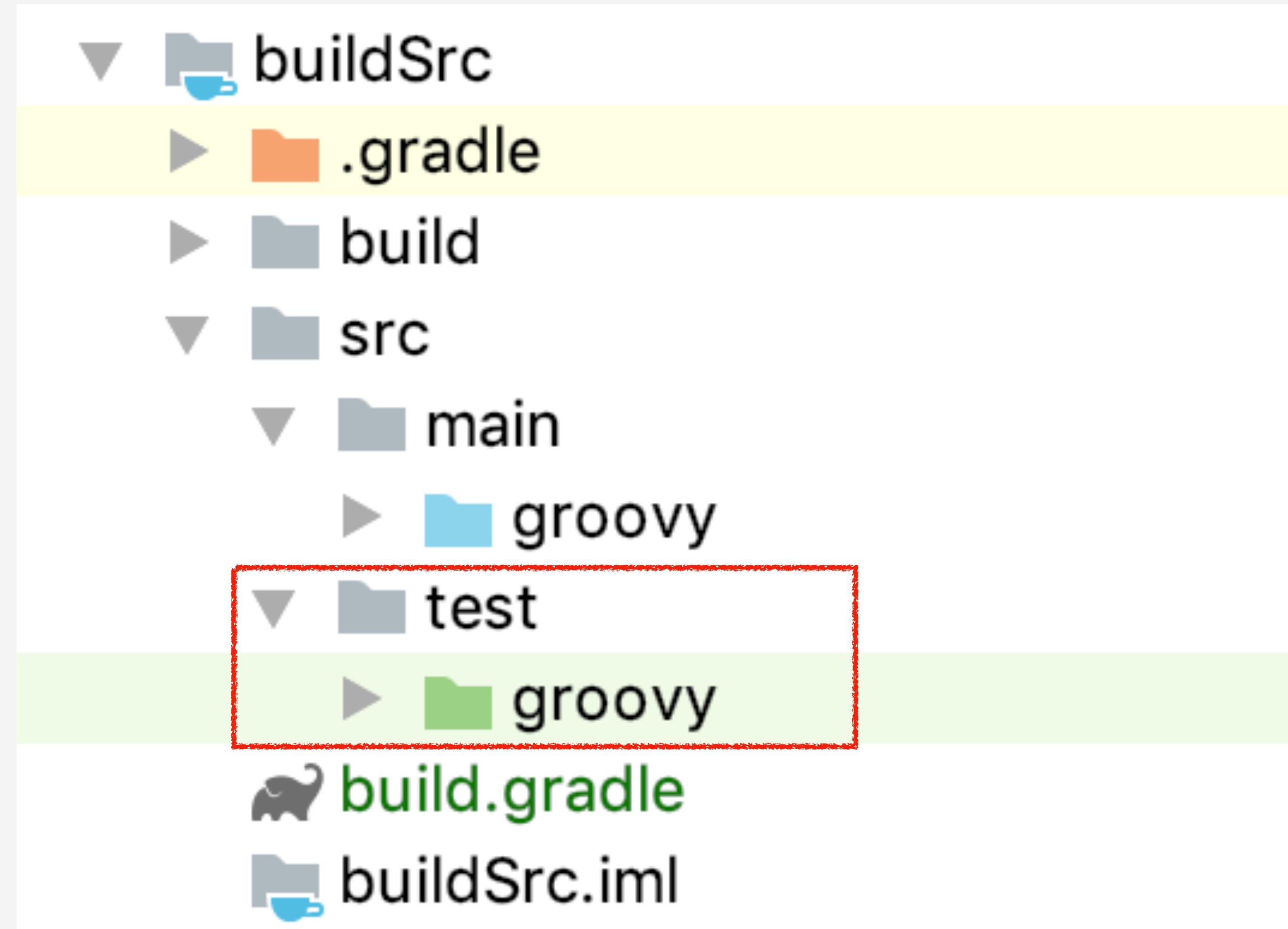
```
class MyIncrementalTask extends DefaultTask {  
    @InputDirectory  
    File sourceDirectory = project.file('src/main/java')  
  
    @TaskAction  
    void doWork(final IncrementalTaskInputs inputs) {  
        if (inputs.incremental) {  
            println 'Do incremental work'  
            inputs.outOfDate({ changed ->  
                println "File '${changed.file.absolutePath}' changed"  
            })  
            inputs.removed({ removed ->  
                println "File '${removed.file.absolutePath}' removed"  
            })  
        } else {  
            println 'Do all work'  
        }  
    }  
}
```

Incremental task

```
class MyIncrementalTask extends DefaultTask {  
    @InputDirectory  
    File sourceDirectory = project.file('src/main/java')  
  
    @TaskAction  
    void doWork(final IncrementalTaskInputs inputs) {  
        if (inputs.incremental) {  
            println 'Do incremental work'  
            inputs.outOfDate({ changed ->  
                println "File '${changed.file.getAbsolutePath}' changed"  
            })  
            inputs.removed({ removed ->  
                println "File '${removed.file.getAbsolutePath}' removed"  
            })  
        } else {  
            println 'Do all work'  
        }  
    }  
}
```

Тестирование плагинов

Тестирование плагинов



Тестирование плагинов

```
@Test
void testDependency() {
    Project project = ProjectBuilder.builder().build()
    project.pluginManager.apply(AndroidAppPlugin)
    assert project.pluginManager.findPlugin('com.android.application') != null
    assert project.tasks.checkCheckstyle instanceof MyCheckstyle
}
```

Тестирование плагинов

```
@Test
void testDependency() {
    Project project = ProjectBuilder.builder().build()
    project.pluginManager.apply(AndroidAppPlugin)
    assert project.pluginManager.findPlugin('com.android.application') != null
    assert project.tasks.checkCheckstyle instanceof MyCheckstyle
}
```

Тестирование плагинов

```
@Test
void testDependency() {
    Project project = ProjectBuilder.builder().build()
    project.pluginManager.apply(AndroidAppPlugin)
    assert project.pluginManager.findPlugin('com.android.application') != null
    assert project.tasks.checkCheckstyle instanceof MyCheckstyle
}
```

Конфигурирование плагинов

Конфигурирование плагинов

1. Описать extension
2. Зарегистрировать его
3. Использовать в build.gradle

Конфигурирование плагинов

```
class MyExtension {  
    boolean enableAnalyzers  
    String name  
}
```

Конфигурирование плагинов

```
class AndroidAppPlugin implements Plugin<Project> {

    @Override
    void apply(final Project project) {
        def extension = project.extensions.create('myApp', MyExtension)
        project.task('welcome') {
            doLast {
                println "Hello ${extension.name}"
            }
        }
    }
}
```

Конфигурирование плагинов

```
class AndroidAppPlugin implements Plugin<Project> {

    @Override
    void apply(final Project project) {
        def extension = project.extensions.create('myApp', MyExtension)
        project.task('welcome') {
            doLast {
                println "Hello ${extension.name}"
            }
        }
    }
}
```

Конфигурирование плагинов

```
apply plugin: AndroidAppPlugin

android { ... }

dependencies { ... }

myApp {
    enableAnalyzers = true
    name = 'John'
}
```

Конфигурирование плагинов

```
class AndroidAppPlugin implements Plugin<Project> {

    @Override
    void apply(final Project project) {
        def extension = project.extensions.create('myApp', MyExtension)
        project.afterEvaluate {
            if (extension.enableAnalyzers) {
                addCheckstyle(project)
            }
        }
    }
}
```

Дополнительные возможности

Версии зависимостей

Версии зависимостей

```
buildscript {  
    ext.versions = [  
        'compileSdk'      : 28,  
        'targetSdk'       : 28,  
        'minSdk'          : 19,  
        'junit'           : '4.12',  
        'gradlePlugin'   : '3.2.1',  
        'supportLib'     : '28.0.0',  
        'okhttp'          : '3.12.1'  
    ]  
  
    dependencies {  
        classpath "com.android.tools.build:gradle:${versions.gradlePlugin}"  
    }  
}
```

root build.gradle

Версии зависимостей

```
buildscript {  
    ext.versions = [  
        'compileSdk'      : 28,  
        'targetSdk'       : 28,  
        'minSdk'          : 19,  
        'junit'           : '4.12',  
        'gradlePlugin'   : '3.2.1',  
        'supportLib'     : '28.0.0',  
        'okhttp'          : '3.12.1'  
    ]  
  
    dependencies {  
        classpath "com.android.tools.build:gradle:${versions.gradlePlugin}"  
    }  
}
```

root build.gradle

Версии зависимостей

```
android {  
    compileSdkVersion versions.compileSdk  
  
    defaultConfig {  
        minSdkVersion versions.minSdk  
        targetSdkVersion versions.targetSdk  
    }  
  
    dependencies {  
        implementation "com.android.support:appcompat-v7:${versions.supportLib}"  
        implementation "com.squareup.okhttp3:okhttp:${versions.okhttp}"  
        testImplementation "junit:junit:${versions.junit}"  
    }  
}
```

project build.gradle

Версии зависимостей

```
android {  
    compileSdkVersion versions.compileSdk  
  
    defaultConfig {  
        minSdkVersion versions.minSdk  
        targetSdkVersion versions.targetSdk  
    }  
  
    dependencies {  
        implementation "com.android.support:appcompat-v7:${versions.supportLib}"  
        implementation "com.squareup.okhttp3:okhttp:${versions.okhttp}"  
        testImplementation "junit:junit:${versions.junit}"  
    }  
}
```

project build.gradle

Версия приложения

Версия приложения

```
android {  
    defaultConfig {  
        versionCode getAppVersionCode()  
        versionName getAppVersionName()  
    }  
}
```

Версия приложения

versions.properties:

major=2
minor=0
patch=15
code=123

Версия приложения

```
def getVersionProps() {  
    def version = new Properties()  
    file("version.properties").withInputStream { version.load(it) }  
    return version  
}  
  
int getAppVersionCode() {  
    return Integer.parseInt(getVersionProps().code)  
}  
  
String getAppVersionName() {  
    def v = getVersionProps()  
    def sdf = new SimpleDateFormat('dd-MM-yy HH:mm:ss')  
    return "${v.major}.${v.minor}.${v.patch}(${sdf.format(new Date())})"  
}
```

Версия приложения

```
def getVersionProps() {  
    def version = new Properties()  
    file("version.properties").withInputStream { version.load(it) }  
    return version  
}  
  
int getAppVersionCode() {  
    return Integer.parseInt(getVersionProps().code)  
}  
  
String getAppVersionName() {  
    def v = getVersionProps()  
    def sdf = new SimpleDateFormat('dd-MM-yy HH:mm:ss')  
    return "${v.major}.${v.minor}.${v.patch}(${sdf.format(new Date())})"  
}
```

Версия приложения

```
def getVersionProps() {  
    def version = new Properties()  
    file("version.properties").withInputStream { version.load(it) }  
    return version  
}  
  
int getAppVersionCode() {  
    return Integer.parseInt(getVersionProps().code)  
}  
  
String getAppVersionName() {  
    def v = getVersionProps()  
    def sdf = new SimpleDateFormat('dd-MM-yy HH:mm:ss')  
    return "${v.major}.${v.minor}.${v.patch}(${sdf.format(new Date())})"  
}
```

Версия приложения

UNINSTALL

FORCE STOP

App notifications

Permissions

No permissions requested

Storage

4.21 MB used in internal storage

Data usage

No data used

Battery

No battery use since last full charge

Open by default

No defaults set

version 2.0.15(24-03-19 23:05:05)

Внешние параметры

Внешние параметры

```
task printProps {  
    doLast {  
        println "Env property ${System.env.MY_PROP}"  
        println "Script property $myProp"  
    }  
}
```

Внешние параметры

```
task printProps {  
    doLast {  
        println "Env property ${System.env.MY_PROP}"  
        println "Script property $myProp"  
    }  
}
```

Внешние параметры



```
gradle.properties x

1 # Project-wide Gradle settings.
2 # IDE (e.g. Android Studio) users:
3 # Gradle settings configured through the IDE *will override*
4 # any settings specified in this file.
5 # For more details on how to configure your build environment visit
6 # http://www.gradle.org/docs/current/userguide/build_environment.html
7 # Specifies the JVM arguments used for the daemon process.
8 # The setting is particularly useful for tweaking memory settings.
9 org.gradle.jvmargs=-Xmx1536m
10 # When configured, Gradle will run in incubating parallel mode.
11 # This option should only be used with decoupled projects. More details, visit
12 # http://www.gradle.org/docs/current/userguide/multi_project_builds.html#sec:decoupled_projects
13 # org.gradle.parallel=true
14
15
16 myProp=World
```

Внешние параметры

```
if (allowPublishing()) {  
    publishing {  
        publications {  
            library(MavenPublication) { ... }  
        }  
        repositories {  
            maven {  
                credentials {  
                    username "$artifactoryUser"  
                    password "$artifactoryPassword"  
                }  
                url "$artifactoryUrl/$artifactoryRepo"  
            }  
        }  
    }  
}
```

Внешние параметры

```
if (allowPublishing()) {  
    publishing {  
        publications {  
            library(MavenPublication) { ... }  
        }  
        repositories {  
            maven {  
                credentials {  
                    username "$artifactoryUser"  
                    password "$artifactoryPassword"  
                }  
                url "$artifactoryUrl/$artifactoryRepo"  
            }  
        }  
    }  
}
```

Внешние параметры

```
private boolean allowPublishing() {  
    return System.env.GITLAB_RUNNER == null &&  
        project.hasProperty('artifactoryUser') &&  
        project.hasProperty('artifactoryPassword')  
}
```

Заключение



Gradle **HE** Build Script



ПРИМЕНЯЙТЕ!

@ mail.ru
group

Спасибо