Rambler&Co 85,47
Компания

20 октября 2016 в 09:53

Несколько Gradle фишек для вашего Android приложения

Разработка под Android*, Блог компании Rambler&Co

В одну из последних рассылок Android Weekly попала статья, в которой упомянули интересные особенности организации сборки проекта. По прочтения мне захотелось поделиться кое-чем из того, что использую я для настройки сборки Android проекта.

Избавляемся от дублирования кода в ваших build.gradle файлах

Казалось бы простая идея, однако такой подход используется довольно редко.

Предположим, у вас есть несколько модулей в приложении, в каждом из которых необходимо прописать buildToolsVersion. Часто эту задачу р путем вынесения конкретной версии в ext-переменную.
Помимо этого, можно оптимизировать код, задав это значение только в одном месте.

Напомню о возможности применения кода из другого gradle файла в вашем build.gradle:

```
apply plugin: 'com.android.application'
apply from: "$buildSystemDir/application.gradle"
```

А в файле application.gradle уже можно указать нужные версии:

```
android {
   compileSdkVersion 24
   buildToolsVersion "24.0.1"

   defaultConfig {
       minSdkVersion 15
       targetSdkVersion 24
       testInstrumentationRunner 'android.support.test.runner.AndroidJUnitRunner'
   }


   compileOptions {
       sourceCompatibility JavaVersion.VERSION_1_7
       targetCompatibility JavaVersion.VERSION_1_7
   }
}
```

Сюда вы можете вынести любые настройки всего проекта. Затем, в каждом конкретном модуле вы можете переопределить значения (при необходимости).

В отдельный gradle файл можно вынести не только конфигурации android — extension, но и тестовые зависимости, настройку jacoco, findbugs и т.д. и т.п.

Добавляем свойства в проект

Возможно вы уже заметили переменную “$buildSystemDir”. Она задана в build.gradle файле корневого проекта:

```
allprojects {
   it.extensions.add("buildSystemDir", "$rootProject.projectDir/buildSystem/")
}
```

После этого в каждом модуле можно использовать это свойство без всяких префиксов. Что, опять же, чуть-чуть сокращает код ;)

Подпись apk

Тема безопасного хранения ключей и паролей от сертификата — очень интересная, и я буду очень рад, если кто-то подскажет хорошее реше ci, безопасное), а пока я приведу свой вариант.

Информация о каждом сертификате хранится в *.properties файле. Он может как присутствовать на машине сборки проекта так и отсутствова

sign.gradle:

```
Properties signProp = new Properties()
try {
   signProp.load(rootProject.file('signForTest.properties').newDataInputStream())

   project.ext {
       forTest = new Object() {
           def storeFile = signProp.get("forTest.storeFile")
           def storePassword = signProp.get("forTest.storePassword")
           def keyAlias = signProp.get("forTest.keyAlias")
           def keyPassword = signProp.get("forTest.keyPassword")
       }
   }
} catch (IOException e) {
   project.ext {
       forTest = new Object() {
           def storeFile = "/"
           def storePassword = ""
           def keyAlias = ""
           def keyPassword = ""
       }
   }
}

android {
   signingConfigs {
       forTest {
           storeFile file(project.ext.forTest.storeFile)
           storePassword project.ext.forTest.storePassword
           keyAlias project.ext.forTest.keyAlias
           keyPassword project.ext.forTest.keyPassword
       }
   }
}
```

Вот и всё. Осталось применить этот файл в build.gradle нашего приложения и использовать сконфигурированный signingConfig в нашем buildT

buildSrc

Что же делать, когда в системе сборки нужен полноценный код, а свой плагин писать нет времени? Для этого можно использовать директори buildSrc.

В корневом каталоге вашего проекта (там где лежит gradlew) создаем новый модуль buildSrc. Этот модуль должен быть обычным java проект

(или groovy или чем-либо ещё), со следующим build.gradle файлом:

```
apply plugin: "groovy"

repositories {
   mavenCentral()
}

dependencies {
   compile localGroovy()
   compile gradleApi()
}

// START SNIPPET addToRootProject
rootProject.dependencies {
   runtime project(path)
}
// END SNIPPET addToRootProject
```

Теперь в этом проекте можно создать какой-либо класс Awesome.groovy, а в build.gradle Android модуля этот класс импортировать и использо

При запуске сборки вашего проекта в первую очередь будет выполнен build модуля buildSrc. После чего произойдет конфигурация остальны модулей.

Не могу сказать, что рекомендую использовать данное решение, т.к. это немного увеличивает время сборки проекта с нуля. Но в крайнем сл может пригодиться.

Flavor dimensions

Эта возможность для конфигурирования проекта хорошо описана в доках (// tools.android.com/tech-docs/new-build-system/user-guide#TOC-M flavor-variants
). Но, либо о ней не все знают, либо не понимают, в каких случаях ее можно использовать, поэтому я решил об этом рассказать.

Предположим, вам необходимо создать платное и бесплатное приложение. При этом ваше приложение выпускается для телевизоров, планше телефонов. Более того, ваше приложение публикуется в разных маркетах.

Добавим в свой build.gradle следующий код:

```
android {
   flavorDimensions "device", "paid", "market"

   productFlavors {
       tv {
           dimension 'device'
       }
       tablet {
           dimension 'device'
       }
       phone {
           dimension 'device'
       }

       free {
           dimension 'paid'
       }
       premium {
           dimension 'paid'
       }

       google {
           dimension 'market'
       }

       amazon {
           dimension 'market'
       }
   }
}
```

Объявив 7 flavor-ов в трех разных группах вы получили целых 12 различных вариантов приложения. Добавим сюда еще buildTypes, и мы пол
огромное количество apk-шек.

> phoneFreeGoogleDebug
> phoneFreeGoogleRelease
> phoneFreeAmazonDebug
> phoneFreeAmazonDebug
> phonePremiumGoogleDebug
> …. и т.д.

Для каждого объявленного вами flavor создаются свои sourceSet. К примеру, если вы выбрали флавор phoneFreeAmazonDebug будут использ
следующие sourceSets:

> src/phoneFreeAmazon
> src/phoneFree
> src/phoneAmazon
> src/freeAmazon
> src/phone
> src/free
> src/amazon

Таким образом, открываются широкие возможности для кастомизации сборок.

Указываем minSdkVersion для buildType

Для ускорения сборки приложения на этапе разработки часто рекомендуют создать отдельный flavor “develop” и указать для него minSdkVers
21. Однако, это не очень удобно, и часто хочется указать этот параметр в buildType debug. Изначально, плагин сборки не позволяет этого сд
однако, при необходимости, эту проблему можно решить следующим хайком.

Для нужного buildType необходимо добавить ext переменную:

```
…
buildTypes {
...
   debug {
...
       ext.minSdkVersion = 21
   }
}
```

Ниже необходимо добавить следующий код:

```
preBuild.doFirst {
   android.applicationVariants.all {
       if (it.buildType.hasProperty("minSdkVersion")) {
           int i = it.buildType.ext.minSdkVersion;
           it.mergedFlavor.setMinSdkVersion(new com.android.builder.core.DefaultApiVersion(i))
       }
   }
}
```

Теперь для всех ваших flavor в debug сборке minSdkVersion будет 21. Однако, тут есть жесткая завязка на внутренности плагина, поэтому пр
обновлении версии плагина что-то может поломаться. Поэтому, не могу рекомендовать использовать этот хак — выбор за вами.

Вместо заключения хочу отметить, что Gradle — очень мощный инструмент для сборки проекта. Если вы уделяете много внимания качеству к
вашего приложения, то не забывайте приводить в порядок и код в build.gradle файлах.

android, gradle

ПОХОЖИЕ ПУБЛИКАЦИИ

26 октября 2015 в 19:21
Конвейерное производство Android приложений
+12 16,4k 137 18

16 сентября 2015 в 12:30
Тестирование на Android: Robolectric + Jenkins + JaCoCo
+20 20k 187 5

1 сентября 2015 в 13:02
Открытый митап Rambler.Android
+11 3,8k 16 9

Комментарии (7)

petrovichtim 20 октября 2016 в 14:38

Спасибо, ничего нового не узнал.

slartus 20 октября 2016 в 15:17

именуем релизный апк:

```
applicationVariants.all { variant ->
        variant.outputs.each { output ->
            if (variant.buildType.name == "release") {
                def vCode = variant.versionCode
                if (vCode < 10)
                    vCode = "0" + vCode

                def date = new Date()
                def formattedDate = date.format('yyyyMMddHHmm')

                output.outputFile = new File(
                        output.outputFile.parent,
                        variant.productFlavors[0].name + "_c${vCode}_" + formattedDate + ".apk")
            }
        }
    }
```

mr-cpp 20 октября 2016 в 18:36 *(комментарий был изменён)*

Статья неплохая. Но мне кажется мало кто будет искать справку по gradle на хабре. Более перспективным для этого документация на stackoverflow. Вам не каж
P.S. у меня там больше рейтинг, я бы вам обязательно плюс поставил)

HotIceCream 20 октября 2016 в 20:50

Спасибо, не знал об этом новом разделе. К сожалению, я не настолько уверен в своем английском, что бы писать англоязычные статьи.

tolkkv 20 октября 2016 в 22:52 *(комментарий был изменён)*

```
allprojects {
   it.extensions.add("buildSystemDir", "$rootProject.projectDir/buildSystem/")
}
```

it.extentsions.add? То ли я ничего не понимаю то ли это делается куда проще — project.ext, как советуют в документации

```
ext {
   buildSystemDir = "$rootProject.projectDir/buildSystem/"
}

task getMyProp << { logger.quiet "i am prop  buildSystemDir: $buildSystemDir"  }
```

И это распространяется на подпроекты тоже…
Откуда вы взяли такую страшненькую конструкцию?

HotIceCream 20 октября 2016 в 23:16

В статье, ссылку на которую я дал в начале как раз и используют такой подход, как вы написали.

it.extensions в данном примере действительно избыточен, он полезен тогда, когда необходимо указать тип добавляемого расширения.

hyperax 27 октября 2016 в 22:37 *(комментарий был изменён)*

Применяю следующий способ подписи APK файлов

Создаем в папочку
project/app/scripts

Туда кладем дополнительные gradle-скрипты и, в частности, скрипт для подписи приложения:
release.gradle

```
android {
    signingConfigs {
        release {
            storeFile file("my_keystore.jks")
            storePassword "some_pass"
            keyAlias "alias_name"
            keyPassword "another_pass"
        }
    }
    buildTypes {
        release {
            signingConfig signingConfigs.release
        }
    }
}
```

Далее, подключаем скрипты из папочки scripts в build.gradle для приложения (/app/build.gradle)

```
fileTree('scripts').each { apply from: "${it}" }
```

Для безопасности добавляем соотв. скрипт и jks файл в git исключения:

```
# release sign info
*.jks
app/scripts/release.gradle
```

ИНТЕРЕСНЫЕ ПУБЛИКАЦИИ

Электроника для ролевых игр в 2016 году GT
1,9k 5 8

Тысячная избранная статья. Как устроено рецензирование в Википедии GT
2,5k 13 28

Библиотека компонентов как инструмент поддержания целостности вебсайта
1,5k 13 6

SmartPoket. Часть 3, заключительная: мой личный органайзер GT
2,9k 6 10

Разработчик «умных» кредиток Plastc собрал предзаказов на $9 млн и объявил о банкротстве GT
7,8k 11 17

Аккаунт
Войти
Регистрация

Разделы
Публикации
Хабы
Компании
Пользователи
Песочница

Информация
О сайте
Правила
Помощь
Соглашение
Помощь стартапам

Услуги
Реклама
Тарифы
Контент
Семинары

Приложения
Загрузите в App Store
ДОСТУПНО В Google

Служба поддержки
Мобильная версия