Чудинов У.Д. Методика применения Flavors при разработке приложений для Android // Международный журнал информационных технологий и энергоэффективности. – 2023. – Т. 8 № 1(27) с. 24–29

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.4

МЕТОДИКА ПРИМЕНЕНИЯ FLAVORS ПРИ РАЗРАБОТКЕ ПРИЛОЖЕНИЙ ДЛЯ ANDROID

Чудинов У.Д.
Челябинский государственный университет, Челябинск, Россия (454001, Челябинск, ул. Братьев Кашириных, 129), e-mail:zotreex@ya.ru

В статье рассматриваются варианты применения возможностей Gradle, в частности Flavors, для приложений под операционную систему Android. Приведены примеры для монолитного и многомодульного приложения, реализован Gradle плагин, для упрощения работы, а также настройки получения исходников каждой отдельной сборки.

Ключевые слова: android; flavors; разработка приложений; android приложение.

METHODOLOGY USING FLAVORS IN ANDROID APPLICATION

Chudinov E.D.
Chelyabinsk State University, Chelyabinsk, Russia (454001, Chelyabinsk, Bratiev Kashirinish St, 129), e-mail: zotreex@ya.ru

The article discusses the options for using Gradle features, in particular Flavors, for applications for the Android operating system. Examples are given for a monolithic and multi-module application, a Gradle plugin is implemented to simplify work, as well as settings for obtaining the sources of each individual assembly.

Keywords: android; flavors; mobile application development; android application

Существует ряд задач, для решения которых необходимо без дублирования кодовой базы реализовать разные варианты сборок приложения. Для этого существуют Flavors - это функция в Gradle, которая позволяет создавать различные версии одного и того же приложения с различными конфигурациями и настройками [1]. Технология очень востребована, но в официальной документации Google отсутствует описание конкретной последовательности действий в ряде ситуаций, например применение в многомодульных проектах, а также в скриптах использующих Kotlin DSL. Имеющиеся примеры ограничиваются простой ситуацией, из которой сложно извлечь последовательность действий в иных ситуациях. При знакомстве с данной технологией возникает необходимость доступного объяснения работы технологии, а также наличия примеров применения в соответствии с новым стандартом Gradle скриптов в виде Kotlin DSL.

Gradle – это система сборки для Java-проектов. Она позволяет описывать зависимости между модулями проекта и автоматически собирать их. Gradle использует сценарии на языке Groovy для описания сборки, но также поддерживает использование сценариев на языке

Kotlin. Большую популярность получил из-за своей гибкости и возможности интеграции с другими инструментами сборки кода.

Flavors отлично решают проблему использования разных SDK. Так, летом 2022 года, в Google Play [1] многие разработчики получили предупреждение о недопустимости использования Huawei SDK в своих приложениях. Чтобы избежать блокировки, можно используя Flavors сделать следующее:

1. Открыть build.gradle вашего проекта
2. Найти блок android
3. Добавить строку flavorsDimensions(“services”)
4. Добавить блок с описанием типов см. рисунок 1.
5. Нажать кнопку Sync project with Gradle
6. Создать папку для каждого Flavor в корневой src проекта, в ней можно будет хранить специфические файлы и ресурсы для каждого Flavor.
7. Настройте сборку нужного варианта, используя меню Build > Select Build Variant в Android Studio

Рисунок 1 – Добавление вариантов Google/Huawei. Пример использования Kotlin DSL.

Для того, чтобы сборщик знал, откуда необходимо брать изменённый код под определённую сборку, необходимо воспользоваться механизмом sourceSets. Обычно он его используют для реализации более чем одной папки res, для разделения файлов ресурсов по категориям/разделам. В случае с Flavors он исполняет ту же роль. Необходимо открыть тот же build.gradle и добавить описание источника кода в нужный flavor. Пример кода см. Рисунок 2.

Рисунок 2 – Добавление sourceSets в сборку Google.

При необходимости можно реализовать аналогичное решение в любом модуле вашего приложения. Для этого нужно повторить все шаги, только в gradle файле вашего модуля, найти его можно во вкладке Project > Gradle Scripts. См. Рисунок 3.

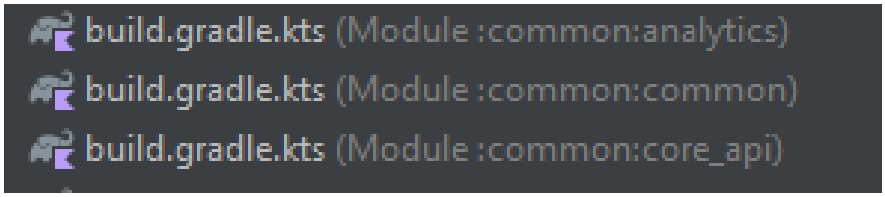

Рисунок 3 – Примеры Gradle скриптов на уровне модулей.

После проделанных манипуляций во вкладке Build > Select Build Variant получим следующий результат в Таблице 1.

Таблица 1 – Итоговые конфигурации сборки

Services / Build Type	Google	Huawei
release	googleRelease	huaweiRelease
debug	googleDebug	huaweiDebug

Переключив в пункте app нужный вариант, Gradle автоматически проведёт индексацию всех файлов проекта и обновит конфигурацию. Android Studio отметит отдельно файлы, которые изменены именно в этом варианте сборке см. Рисунок 4.

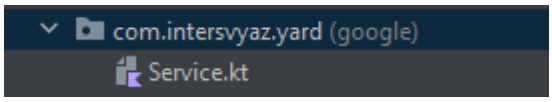

Рисунок 4 – Подсказка (google) в Android Studio.

При использовании Flavors в многомодульном приложении, может возникнуть ошибка сборки, связанная с тем, что модуль пытается создать вариант с таким же именем, как уже создан где-либо в проекте. Рассмотрим ситуацию, когда нам нужно применить функционал Flavors на множество модулей.

Чтобы не прописывать одну и ту же конфигурацию в каждом модуле, можно упростить задачу, реализовав Gradle Plugin [2]. Для этого нужно:

1. Создать Kotlin модуль buildSrc
2. Создать класс наследующий Plugin<Project>
3. Переопределить метод apply.
4. Вызвать project.android и определить значения для вариантов сборок
5. Важное отличие от добавления в монолитном приложении, на уровне модулей придётся указать missingDimensionStrategy, на уровне gradle app внести изменения как если бы модулей не было.
6. В каждом gradle файле модуля прописать plugins { `feature-module` }, Где feature-module название вашего класса до слова Plugin.

Пример готового плагина на Рисунке 5.

```
class FeatureModulePlugin : Plugin<Project> {
    chudinoff
    override fun apply(project: Project) {
        project.addPlugins()
        configureAndroidSettings(project)
    }

    chudinoff +1 *
    private fun configureAndroidSettings(project: Project) {
        project.android { this: BaseExtension
            flavorDimensions( ...dimensions: "services")
            productFlavors { this: NamedDomainObjectContainer<ProductFlavor>
                create( name: "google") { this: ProductFlavor
                    dimension = "services"
                    missingDimensionStrategy( dimension: "services")
                }
                create( name: "prod") { this: ProductFlavor
                    dimension = "services"
                    missingDimensionStrategy( dimension: "services")
                }
            }

            sourceSets { this: NamedDomainObjectContainer<AndroidSourceSet>
                getByName( name: "huawei") { this: AndroidSourceSet
                    java { this: AndroidSourceDirectorySet
                        srcDirs("src/huawei/java", "src/main/java")
                    }
                }
                getByName( name: "google") { this: AndroidSourceSet
                    java { this: AndroidSourceDirectorySet
                        srcDirs("src/google/java", "src/main/java")
                    }
                }
```

Рисунок 5 – Готовый Gradle плагин написанный на Kotlin.

Ещё одна проблема, которую умеет решать механизм Flavors, это потребность большего количества вариантов сборки. Например, помимо возможности разделять Google и Huawei варианты, есть потребность делить сборки для разных клиентов [2].

Для этого существует MultiDimensions Flavors. Чтобы его применить достаточно:

1. В flavorDimensions через запятую добавить новое “Измерение” – например buyer. В таком же порядке будут имена в сгенерированных командах assemble.
2. В productFlavors определить новые варианты с указанием dimension = “buyer”
3. Аналогично повторить изменения в sourceSets.

Но возможности Gradle позволяют упростить генерацию большого количества сборок, без необходимости прописывать каждую в productFlavors и sourceSets. Для этого можно создать дата-класс в синтаксисе Kotlin. Предположим класс Buyer содержащий поля

FlavorName и ApplicationId, первое необходимо для создания варианта, второе для изменения имени пакета приложения у данного варианта.

Далее необходимо создать переменную, например, buyerList и используя listOf<Buyer>() перечислить в круглых скобках все экземпляры покупателей. Теперь можно используя buyerList и forEach упростить создание вариантов, пример реализации productFlavors см. Рисунок 6, пример реализации sourceSets см Рисунок 7. Обратите внимание, в sourceSets изменился способ обращения к нужному варианту, обусловлено это различиями синтаксиса Groovy и Kotlin DSL варианта [3].

```
productFlavors { this: NamedDomainObjectContainer<ProductFlavor>
    buyerList.forEach {
        create( name: "${it.flavorname}") { this: ProductFlavor
            signingConfig = signingConfigs.getByName( name: "release")
            dimension = "buyer"
            setApplicationId(it.applicationId)
        }
    }
```

Рисунок 6 – Пример использования циклов при создании вариантов сборок.

```
sourceSets { this: NamedDomainObjectContainer<AndroidSourceSet>
    buyerList.forEach { buyer ->
        named(buyer.flavorname) { this: AndroidSourceSet
            kotlin.srcDirs("src/${buyer.flavorname}/kotlin")
            res.srcDirs("src/${buyer.flavorname}/kotlin/com/intersvyaz/yard/ui/$it/res")
        }
    }
```

Рисунок 7 – Пример использования циклов при указании исходного кода варианта сборок.

После проведённых операций выйдет трёхмерное пространство для выбора варианта сборки. Gradle сгенерирует все вариации assemble команд, будут доступы сборки вида assemble[buyer][service][type], где buyer любой из списка buyerList, service будет представлен Google или Huawei, а type – Release или Debug. Например assembleBuyer1HuaweiDebug, возьмёт ресурсы из src в порядке имён, т.е на основной main вариант, будут применены изменения из папки Buyer1, затем из Huawei и следом из Debug. Для крайне редких ситуаций вы можете разместить изменения в папке с именем Buyer1Huawei, HuaweiDebug или Buyer1HuaweiDebug, тогда они будут выбираться приоритетнее. Но в концепции переиспользования кодовой базы, делать это стоит только при большой необходимости.

Дополнительные примеры ситуаций, когда Flavors решают проблему создания приложений с различными настройками, конфигурациями и возможностями:

- Создание бесплатной и платной версии приложения. Используя описанный механизм, можно создать две версии приложения с различным функционалом. Например, в бесплатной версии может быть ограниченное количество функций и наличие рекламы, в то время как в платной версии расширенный функционал и отсутствие рекламы.

- Создание версии приложения для разных регионов с различными языками и валютами. Например, версия для России будет на русском, а указание цен в рублях, а для Турецкой версии то же самое приложение будет на турецком и цены в лирах.

Рассмотренные в статье реализации механизма Flavors необходимо применять при разработке современных мобильных приложений. Разработанная методика предоставляет последовательность действий для использования Flavors в ситуациях множественных измерений и при использовании Kotlin DSL. Предоставленная реализация плагина для Gradle значительно упрощает применение и поддержку работы Flavors в многомодульном приложении. Внесение изменений в структуру проекта может проходить затруднительно, но для этого есть все инструменты и подробная инструкция из этой статьи. Разработчику так же поможет отличная интеграция Android Studio с настройками Gradle, например подсказки о наличии дополнительных вариантов этого файла в другой конфигурации, а также возможность собрать код с нужными параметрами в одно нажатие. Данная технология позволяет решать достаточно много задач в Android разработке.

Список литературы

1. Документация Android. URL: https://developer.android.com/studio/build/ (дата обращения 20.11.2022).
2. Advanced Android Flavors Part 2 — Enter Flavor Dimensions. URL: https://proandroiddev.com/advanced-android-flavors-part-2-enter-flavor-dimensions-4ad7f486f6 (дата обращения 20.11.2022).
3. Kotlin DSL: Теория и Практика. URL: https://habr.com/ru/company/haulmont/blog/341402/ (дата обращения 20.11.2022).

References

1. Android documentation. URL: https://developer.android.com/studio/build/ (date appeals 20. 11.2022).
2. Advanced Android Flavors Part 2 — Enter Flavor Dimensions. URL: https://proandroiddev.com/advanced-android-flavors-part-2-enter-flavor-dimensions-4ad7f486f6 (retrieved 11/20/20/2022).
3. Kotlin DSL: Theory and Practice. URL: https://habr.com/en/company/haulmont/blog/341402/ (accessed 2022.11.20).