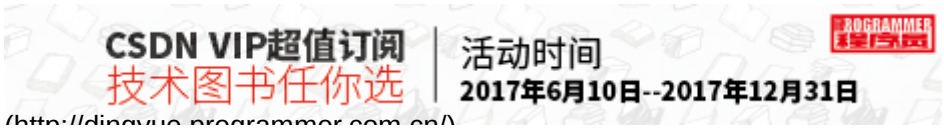
拥抱 Android Studio（三）：溯源，Groovy 与 Gradle 基础

Android (http://www.csdn.net/tag/Android/news) Gradle (http://www.csdn.net/tag/Gradle/news) Groovy (http://www.csdn.net/tag/Groovy/news) 开发经验 (http://www.csdn.net/tag/开发经验/news)



本文为“拥抱Android Studio”系列第三篇。作者何畅彬，现任BugTags技术总监，关注移动SDK研发、后端服务设计与实现，个人博客：http://kvh.io/ (http://kvh.io/)。

在将 ADT 与 Android Studio 进行对比鼓励开发者们积极拥抱，并列举了 Android Studio 与 Gradle 下一些深入实用的功能，分享自己以及在帮助用户集成过程中遇到的坑之后，作者追本溯源，对 Groovy 与 Gradle 进行了讲解，帮助开发者们更快上手。

关于学习方式续

回忆起大学那个白衣飘飘的年代，开始金工实习却发现基础学的不牢靠，越来越胆小，越来越糊涂。所幸得到一位高年级学姐指导，赶紧找当时的书或者笔记，快速把基础知识温习一遍，再结合实践中思考，终于豁然开朗。

相信看过前一篇 《Android Studio 与 Gradle 深入》 (http://www.csdn.net/article/2015-12-21/2826520)的同学，有一部分就会遇到我初识 Gradle 时的困惑：代码我也依稀看得懂，但就是不知道还能这样写，为什么这样写。

问题与解决方案

回想我在 Gradle 的学习过程中遇到的问题与及其解决方案，总结出下面三点：

- 原理不懂：学习 Groovy 与 Gradle 的基础原理
- Gradle 实践不懂：学会找示例，学习开源例子
- 方法和属性不懂：学会查文档

下面的我将以解决三个问题为线索，介绍 Groovy 和 Gradle。

学习 Groovy

Groovy 概述

Gradle 采用了 Groovy 语言作为主要的脚本语言。一个 build.gradle 文件，其实是一个 Groovy 类。

Groovy 是一个基于 JVM 的语言，代码最终编译成字节码（bytecode）在 JVM 上运行。它具有类似于 Java 的语法风格，但是语法又比 Java 要灵活和方便，同时具有动态语言（如 ruby 和 Python）的一些特性。

Groovy 的诸多特定，很适合用来定义 DSL (https://en.wikipedia.org/wiki/Domain-specific_language)（Domain Specific Language）。

> 简单的来讲 DSL 是一个面向特定小领域的语言，如常见的 HTML、CSS 都是 DSL，它通常是以配置的方式进行编程，与之相对的是通用语言（General Purpose Language），如 Java 等。

既然是一门语言，就肯定有自己的特性。我们要从下面几个步骤来介绍 Groovy：

- 环境安装
- 语言基础
- 语言特性及其本质

环境安装

Groovy 官方安装文档 (http://www.groovy-lang.org/install.html)提供多种方式进行安装，确保你不会在**跪在环境配置的路上 ^-^**

- Windows 下推荐 binary 包配置环境变量

下面只介绍 Mac 下使用 sdkman 的安装方式。

- 下载安装 sdkman，执行下面命令，按照提示安装即可

```
curl -s http://get.sdkman.io | bash
```

- 使环境变量生效

```
$ source "$HOME/.sdkman/bin/sdkman-init.sh"
```

- 安装 Groovy

```
$ sdk install groovy
```

- 查看当前版本，如果能否运行且输出对应版本，就是成功了

```
$ groovy -version
Groovy Version: 2.4.4 JVM: 1.8.0_25 Vendor: Oracle Corporation OS: Mac OS X
```


安装好环境之后，先来一个 hello, world！


```
println "hello, world!"
```

- 保存退出，执行

```
$ groovy test.groovy
hello, world!
```

Wow, So easy!

语言基础

下面将会用一些实际的例子，介绍一些最重要的点，

> 例子都已经传到 github 的 demo (https://github.com/kevinho/Embrace-Android-Studio-Demo/tree/master/s3-GroovyGradle) 项目中。
>
> 第一次使用 demo 项目的时候，需要等待自动下载几个远程包。
>
> 笔者一个 Java 程序员，可以你能够看到很多 Java 的习性还是留在代码中。

文件与类，变量与函数

Groovy 代码文件，支持不显式声明类：

ScriptClass.groovy

```
groovy/build/classes/main/io/kvh/as/groovy/ScriptClass.class
```

```
public class ScriptClass extends Script {
    public ScriptClass() {
        CallSite[] var1 = $getCallSiteArray();
    }

    public ScriptClass(Binding context) {
        CallSite[] var2 = $getCallSiteArray();
        CallSite[] var1 = $getCallSiteArray();
        return var1[1].callCurrent(this, "hello,world");// got it?
    }
}
```

Groovy 支持如下的方式来定义变量和函数：

VarAndMethod.groovy

```
def varAndMethod() {
    def a = 1//不显式声明变量类型
    a = "abc"//运行时改变类型

    println a//无需；结束一行代码
    a = 4//最后一行作为返回值
}
def ret = varAndMethod()//文件内运行方法

println ret//输出4
```

字符串

Groovy 支持单引号，双引号，三单引号声明一个字符串；

Quoted.groovy

```
println doubleQ
println tripleQ
}
```

Groovy 还支持以：

```
"""..."""
/.../
$/.../$
```

来声明字符串，详情参见参考文档。

List，Array 和 Map

Groovy 默认使用 java.util.ArrayList 来提供 List 服务，但提供了更加灵活易用的操作方式：

Collections.groovy

```
def playList() {
    def lst = ["a",2,true]//支持不同类型元素

    println(lst)
}

playList()
```

请输入标题
请输入链接地址
请输入推荐理由
请输入标签

```
    println(strArr)
}

playArray()
```

发布到 主题 发布 评论

使用 key:value 的方式定义 Map，注意 key 的正确使用方式：

```
def playMap() {
    def map = [a: "a", b: "b"]

    println(map)
    def key = "name"
    def map2 = [key: 'a']//未使用
```

import

Groovy 提供了更强大的 import

- 默认 import，这些类都是被默认 import 到代码中的，可以直接使用

```
import java.lang.*
import java.util.*
import java.io.*
import java.net.*
import groovy.lang.*
import groovy.util.*
import java.math.BigInteger
import java.math.BigDecimal
```

- import alias

引入一个类，通过 as 关键字赋予一个别名，有点 JavaScript 的意思么？

```
import java.lang.String as KString
println(new KString("aaa"))
```



```
{ [closureParameters -> ] statements }
```

可以省略方括号内的内容，也就是说，可以没有参数列表。

Closure.groovy

当闭包不声明参数列表，默认参数是 it；闭包被定义之后，是一个 Closure 对象，可以对其调用 call 方法使其执行。

```
def defaultIt() {
    3.times {
        println it //默认参数 it
    }
}

defaultIt()
```


面向对象特性

- 定义和实例化一个类

GroovyCLass.groovy

```
class People{
    String name
    int age
}

People p1 = new People();
People p2 = new People(name:"Luis",age: 29)//通过类似 map 的方式赋值参数
```

- 方法的默认参数

```
def foo(String p1, int p2 = 1) {
    println(p1)
    println(p2)
}
foo("hello")
```

```
class People{
    String name
    int age
}
```

当变量声明为 final 的时候，默认就没有 setter

- Trait

Groovy 提供了一个叫做 Trait 特性实现了多继承，还有很多强大的功能，读者可以自己探索。

```
trait Fly {
    void fly() {
        println("fly")
    }
}

trait Walk {

}

Duck duck = new Duck()
duck.fly()
duck.walk()
```

Groovy 基础小结

至此，我们已经熟悉了 Groovy 的基本语法和特性，相信你也能够使用 Groovy 写一些基础程序了。Groovy 还有很多深入的内容，请用到的时候，参考这本这个 pdf：《Programming Groovy 2》 (https://pragprog.com/book/vslg2/programming-groovy-2)。

下面开始介绍使用 Groovy 写 Gradle 程序，主要的内容来自 《Gradle Sser Guide》 (https://docs.gradle.org/current/userguide/userguide.html)。

学习 Gradle

Gradle 安装

- 下载 zip (https://gradle.org/gradle-download/) 安装包
- Mac 下使用 home brew

Gradle Wrapper 会把不同的版本 Gradle 安装在：

```
$USER_HOME/.gradle/wrapper/dists
```

Gradle Build 的生命周期

回忆一下《Android Studio 与 Gradle 深入》 (http://www.csdn.net/article/2015-12-21/2826520)中的 Android Studio 项目文件结构：

```
.
├── app                     //app module
│   ├── build.gradle        //app module 的 build.gradle
├── build.gradle            //项目 build.gradle，通常配置项目全局配置，如 repositories 和 dependencies
├── gradle.properties       //项目属性文件，通常可以放置一些常量
├── lib                     //lib module
│   ├── build.gradle        //lib module 的 build.gradle
└── settings.gradle         //项目总体设置，通常是配置项目中所有的 module
```

Gradle 构建三个阶段：

- 初始化：Gradle 支持单 module 构建和多 module 构建（Android Studio 创建的项目默认是多 module）。初始化阶段，Gradle 会为每一个 module 中的 build.gradle 文件创建一个 Project 实例。
- 配置：项目根目录的 build.gradle 会首先被执行

settings.gradle

```
include ':app', ':groovy'

println 'print in settings.gradle'
```

在 settings.gradle 文件中，添加一行打印语句，在控制台中，切换到当前项目根目录下执行：

```
./gradlew -p groovy build
```

可以在看出 settings.gradle 的代码每次都会率先执行。

Task

接下来，我们开始学习 Gradle 的核心 Task。

groovy/build.gradle

定义一个 Task：

```
task hello {
    doLast {
        println 'Hello,'
    }
}
```



```
task World << {
    println 'World!'
}
```

注意，如果定义成这样：

```
task hi {
    println 'description hi'
}
```

在进行初始化和配置的时候，下面语句就会运行。

```
println 'hi'
```

这种语法通常是用来定义 task 的描述信息。

Task 可设置 dependsOn 和 finalizedBy：

```
task hello {
    doLast {
        println 'Hello,'
    }
}
```


Plugin

Gradle 的核心代码，只提供了一个框架，具体的功能（如构建 Android 工程）是通过插件机制来实现的。

Gradle 提供了大量官方的插件，如 Maven、Groovy、Java、Publishing、Signing等，也有大量第三方的插件（Android），甚至每个人都可以自己实现一个插件(如 笔者开发的 Bugtags 插件，这个将在最后一篇讲述)。

这些 plugin 定义了一系列的 task、DSL 和约定，在build.gradle 文件使用这些 plugin：

```
apply plugin: java
```

当你写了一个独立的 file_uri.gradle 文件，你可以通过：

```
apply from: 'file_uri.gradle'
```

来引入你的 gradle 文件，这个文件甚至可以在某个服务器上。

Gradle 实践参考

学习了基础理论之后，如果你还是不知道如何开始写，那就先来实现一个自定义 apk 名称的功能吧！


```
android.applicationVariants.all { variant ->//获取 variant 参数，就是 productFlavor x buildType
    variant.outputs.each { output ->//获取输出文件
        ".a
    }
```

你问我怎么知道 android 下有个 applicationVariants？其实我也不知道的，也得找文档。

因为使用的是 Android 的插件，那就得在谷歌搜 “android gradle plugin dsl”，果然有个 Android Plugin DSL Reference (http://google.github.io/android-gradle-dsl/current/)。

点进去找找，里面有关于 build variant 的文档： applicationVariants (http://google.github.io/android-gradle-dsl/current/com.android.build.gradle.AppExtension.html#com.android.build.gradle.AppExtension:applicationVariants)，既然是一个 Set，那就可以调用 all 方法。

写代码调试，再配合文档，你就晓得该怎么写了。

如果你还是不知道如何入手，那我提供几个开源参考：

- gradle-bintray-plugin (https://github.com/bintray/gradle-bintray-plugin)：bintray 提供的开源插件

- gradle-node-plugin (https://github.com/srs/gradle-node-plugin)： 一个运行 NodeJS 脚本的插件
- linkedin-gradle-plugins (https://github.com/linkedin/gradle-plugins)： linkedin的 Gradle 插件集合

参考文档

相信参照开源项目动手写了几个小程序之后，你已经小有感觉了，那就记得把文档地址备齐了，用到的时候，查一下：

- Android Plugin DSL Reference (http://google.github.io/android-gradle-dsl/current/index.html)：使用 Android 插件必备

另外，也有大量很好的中文文档，比如这几篇：

- 邓凡平老师的 Gradle 介绍 (http://blog.csdn.net/innost/article/details/48228651)
- Gradle User Guide 中文版 (https://www.gitbook.com/book/dongchuan/gradle-user-guide-/details)

总结

笔者从 Gradle 入门到现在略懂，经历了大量懵懂的时光。最后狠下心去系统学习了 Groovy 和 Gradle 的基础之后，最终茅塞顿开。希望读者遇到类似的情况，一定要沉下心，多学多练。

在接下来的两篇，我将分别介绍将发布远程库和编写 Gradle 插件。

系列导读

本文是笔者《拥抱 Android Studio》系列第三篇，其他篇请点击：

- 拥抱 Android Studio 之一：从 ADT 到 Android Studio (http://www.csdn.net/article/2015-12-17/2826507)
- 拥抱 Android Studio 之二：Android Studio 与 Gradle 深入 (http://www.csdn.net/article/2015-12-21/2826520)

(http://geek.csdn.net/user/publishlist/tangxiaoyin)
唐门教主 (http://geek.csdn.net/user/publishlist/tangxiaoyin)
发布于 Android开发者 (http://geek.csdn.net/forum/65) 2016-01-05 10:17

ehomewetrust (http://my.csdn.net/ehomewetrust) 2016-02-06 10:16 来自 移动客户端
(http://my.csdn.net/ehomewetrust)
java升级版。代码少了一大半。

freefaces (http://geek.csdn.net/user/publishlist/freefacefly) 2016-01-06 20:32
(http://geek.csdn.net/user/publishlist/freefacefly)
忘了在文章中加提问 qq 群了，
有问题可以加：453503476

oolgloo (http://geek.csdn.net/user/publishlist/oolgloo) 2016-01-05 20:40
(http://geek.csdn.net/user/publishlist/oolgloo)

写了一个 gradle plugin，可以修改 aapt 生成的二进制文件，编译插件包，下：https://github.com/wequick/Small/tree/master/Android (https://github.com/wequick/Small/tree/master/Android)

0 freefaces (http://geek.csdn.net/user/publishlist/freefacefly) 2016-01-06 11:19
(http://geek.csdn.net/user/publishlist/freefacefly)
这个有点意思哈，不过我还是想问下，这个 plugin 主要是实现什么功能呢？

0 (http://geek.csdn.net/user/publishlist/qq_24918901)
gvm.net已经不存在了 更换了域名 需要修改命令 `curl -s http://get.sdkman.io | bash;`

0 freefaces (http://geek.csdn.net/user/publishlist/freefacefly) 2016-01-05 15:10
(http://geek.csdn.net/user/publishlist/freefacefly)
谢谢提醒！但是我没法修改这个文档。所以我在我的博客上修改了。

0 唐门教主 (http://geek.csdn.net/user/publishlist/tangxiaoyin) 2016-01-05 17:20
(http://geek.csdn.net/user/publishlist/tangxiaoyin)
回复@freefaces (http://geek.csdn.net/user/publishlist/freefacefly):
已修正，请查阅指正:)

0 shiter (http://geek.csdn.net/user/publishlist/wangyaninglm) 2016-01-05 13:44
(http://geek.csdn.net/user/publishlist/wangyaninglm)
谢谢分享，这个系列写的好

请输入标题
请输入链接地址

请输入推荐理由

请输入标签

发布到 主题 发布 评论