# Groovy语言和Gradle构建学习笔记

[Groovy语言和Gradle构建学习记录 - 1 -](#_Toc24552)

[一、 Groovy - 2 -](#_Toc12166)

[1.1、Groovy介绍 - 2 -](#_Toc86)

[1.2、Groovy使用 - 2 -](#_Toc17050)

[二、 Gradle - 3 -](#_Toc17058)

[2.1、Gradle介绍 - 3 -](#_Toc32579)

[2.2、Gradle构建结构 - 3 -](#_Toc8102)

[2.3 Gradle工作流程 - 4 -](#_Toc22103)

## Groovy

### 1.1、Groovy介绍

Groovy是一种基于JVM的敏捷开发语言，Groovy 代码能够与 Java 代码很好地结合，也能用于扩展现有代码。由于其运行在 JVM 上的特性，Groovy 可以使用其他 Java 语言编写的库。

### 1.2、Groovy使用

本次学习Groovy主要是为了结合Gradle在Android构建这个领域，所以用到的都一些比较简单基础的东西，但是Groovy本身是一门非常复杂的语言

Groovy使用基本知识：

1. Groovy注释标记和Java一样，可以使用 // 或者 /\*\*/
2. 结尾可以不适用 ;
3. 定义变量可以不指定类型， 使用关键字def
4. Java中的基本数据类型

在Java中的int , boolean , float等这些基本数据类型，在Groovy中对应的就是它们的包装类型，int 对应的就是Integer ，boolean对应Boolean

1. Groovy中的容器类

Groovy中的容器类有三种：

1. List 底层对应Java中的List接口，一般用ArrayList作为真正的实现类

定义：List变量由[ ]定义 ， def list = [5 , ‘test’ , true] //元素可以是任何对象

存取：可以直接通过索引取值，类似数组，list[i] , 并且不必担心数组越界问题，list会自动往该索引添加元素

1. Map 底层对应Java中的LinkedHashMap

定义：Map变量[ : ]定义 , def map = [‘str’ : obj1 , ‘str2’ : obj2]

存取：map.keyName , map[‘keyname’] , 添加新元素 map.keyName = “keyname”

1. Range 表示范围，是对List的扩展

定义：def range 1..6 //表示包含1,2,3,4,5,6这6个值

def rangeWithoutEnd = 1..<6 //表示不含最后一个元素

1. 闭包

闭包表示一段可执行的代码

a) 定义格式：

def xxx = {

Paramters -> code

}

或者

def xxx = {

无参数 , code

}

例子

def closure = {

String param1 , int param2 -> //->前面定义参数，后边定义代码

println “this is code”

}

b )闭包的调用

例 ： 闭包对象 = closure

Closure.call(参数) || Closure(参数)

4、Api文档： http://www.groovy-lang.org/api.html

## Gradle

### 2.1、Gradle介绍

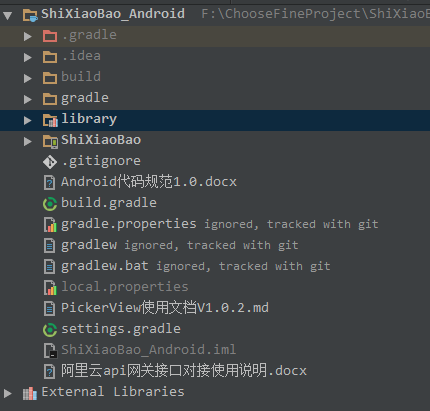
Gradle是一个基于Apache Ant和Apache Maven概念的项目自动化构建工具。它使用一种基于Groovy的特定领域语言来声明项目设置，抛弃了基于XML的各种繁琐配置。

面向Java应用为主。当前其支持的语言限于Java、Groovy和Scala，计划未来将支持更多的语言

### 2.2、Gradle构建结构

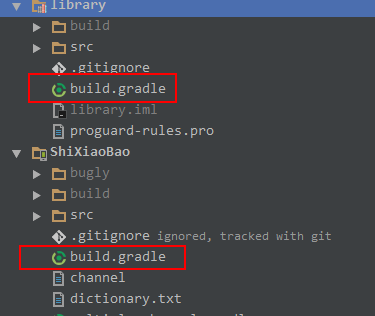
Gradle中，每一个待编译的工程都叫一个project，并且每一个project构建的时候有包含一系列的Task， Task的数量又是由编译脚本指定的插件决定的

以Android开发为例，我们的施小包项目为例，如下图

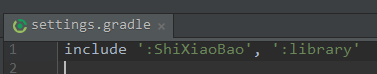


项目中包含一个Android Library工程(Library)和一个Android APP工程(ShiXiaoBao), ShiXiaoBao依赖于Library

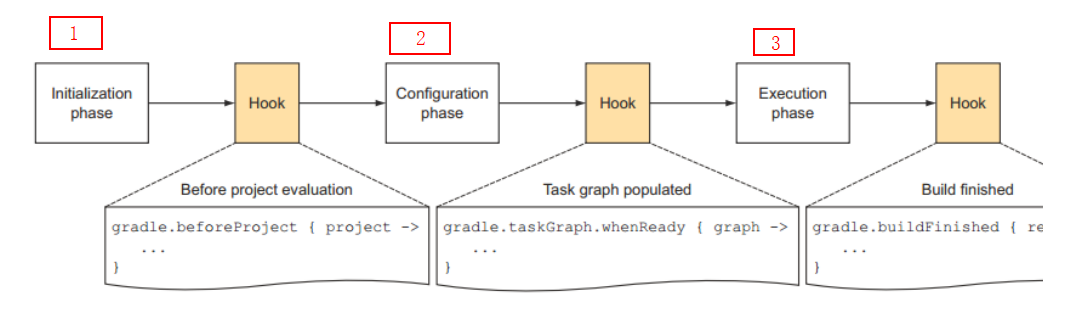
所以这种情况下，其实每一个Library和每一个App都是单独的project，Gradle要求每一个project根目录下要有一个build.gradle文件，该文件就是该project的编译脚本



如果要编译一个project，就得进入该project的目录，执行gradle命令，所以如果有多个project就需要执行多次命令，因此在Gradle中有个一叫做Multi-projects Build, 只需要在最外层项目下添加一个build.gradle文件和一个setting.gradle文件，setting.gradle中需要指明这个项目中包含了那些project



### 2.3 Gradle工作流程



上图就是gradle的工作流程

主要包含三个阶段

第一阶段就是初始化阶段，相当于执行setting.gradle

下一个阶段就是Configuration阶段，这个阶段的目标就是解析每个project的build.gradle,在这两个阶段中可以添加一些定制化的hook关于每个project中各个Task的依赖关系，Configuration会建立一个有向图来描述依赖关系在这里，我们还可以添加一个hook，放依赖关系建立之后执行一些操作

最后阶段就是执行任务，任务执行完成之后，还可以添加hook

### 2.4、具体使用

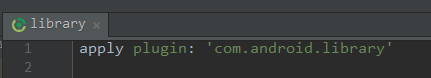
Gradle是基于Groovy，而Groovy又是基于Java的，所以最后执行的时候，还是会把gradle文件转换成Java对象，Gradle主要有三种对象

1. Gralde ： 执行gralde 命令时，会构造出一个Gradle对象
2. Project ：build.gradle会转换成Project对象
3. Settings ：setttings.gralde 会转换成Settings对象

Project对应具体的工程，不同工程需要加载不同的插件，而一个Project中的Task数量也往往是由插件决定的，

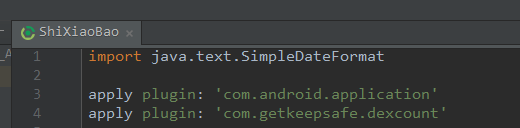
1、加载插件 ： 调用apply函数

例如：编译Library 加载此插件



编译Applicaiton，就加载第一个插件

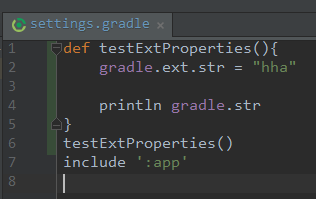
当然还可以加载其他插件，类似第二种



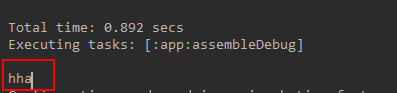
1. 设置属性

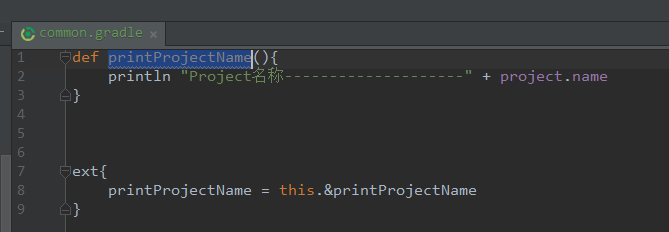
在我们的项目中一般都是由多个build.gradle文件存在的，所以这时候我们如何设置属性使得该属性能够跨脚本传播

Gradle提供了一种 extra property的方法，我们在第一次定义该属性的时候可以通过ext前缀标记，定义好之后后面的存取就不需要ext了，ext属性支持Project和Gradle对象



上图，就是设置gradle的ext属性，gradle默认就是Settings和Project的成员变量，所以可以直接引用,下图是输出结果，

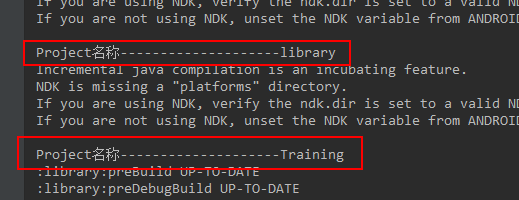


如果多个项目的build.gradle需要引用同一个函数，如果每个文件中都都写一个相同的方法就会十分麻烦并且代码冗余，这种情况下我们可以新建一个gralde文件，以类似插件的方式，通过project的apply方法注入, 这里在根目录下新建一个common.gradle文件

由于我们在app和Library中都需要用到这个函数所以我们可以通过apply函数在build,gradle文件中引用

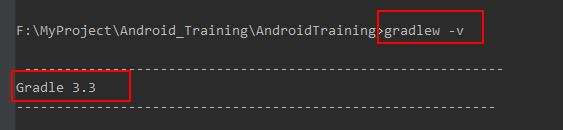


我们每个project的build.gradle都会引用这个common文件，所以上述common中的project相当于引用他的project,我们在每个build文件中只需要使用printProjectName就可以使用common文件中的方法了



1. 常用命令行

gradlew -v 查看gradle版本号



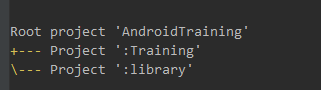
gradlew clean 清除自动生成的build文件夹

gradlew build 检查依赖并打包，该命令会把debug和release下的包都打出来

gradlew assembleDebug 编译并打包debug包

gradlew assembleRelease 编译并打包release包

gradlew projects 查看项目下包含了那些project



gradlew tasks 查看项目包含那些Task ,若要查看具体project下的信息，通过gradlew 目录:tasks , 会列出所有的task后面的是描述该task所做的事

