困惑的build.gradle

在由Eclipse转到**[Android](http://lib.csdn.net/base/android" \o "Android知识库" \t "_blank)** Studio之后，第一个让我感到困惑的是build.gradle这个文件了，实在话，当时我并不知道这么写是几个意思。

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. apply plugin: 'com.android.application'
3. android {
4. compileSdkVersion 21
5. buildToolsVersion "21.1.2"

8. defaultConfig {
9. applicationId "com.example.test"
10. minSdkVersion 17
11. targetSdkVersion 21
12. versionCode 1
13. versionName "1.0"
14. }
15. buildTypes {
16. release {
17. minifyEnabled false
18. proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
19. }
20. }
21. }

24. dependencies {
25. compile fileTree(dir: 'libs', include: ['\*.jar'])
26. compile 'com.android.support:appcompat-v7:21.0.3'
27. }

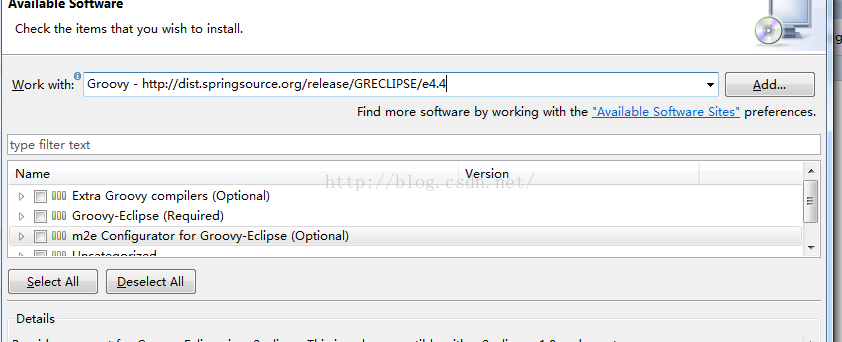
如上就是建立一个as工程生成的基本构建文件了，暂且不需要知道他底层怎么实现，至少要知道这么写是几个意思，为何要这么写，我们需要一步一步来揭穿它。

一：Gradle 是基于Groovy语言来构建的，第一点就得了解什么是Groovy语言，以及它的语法。

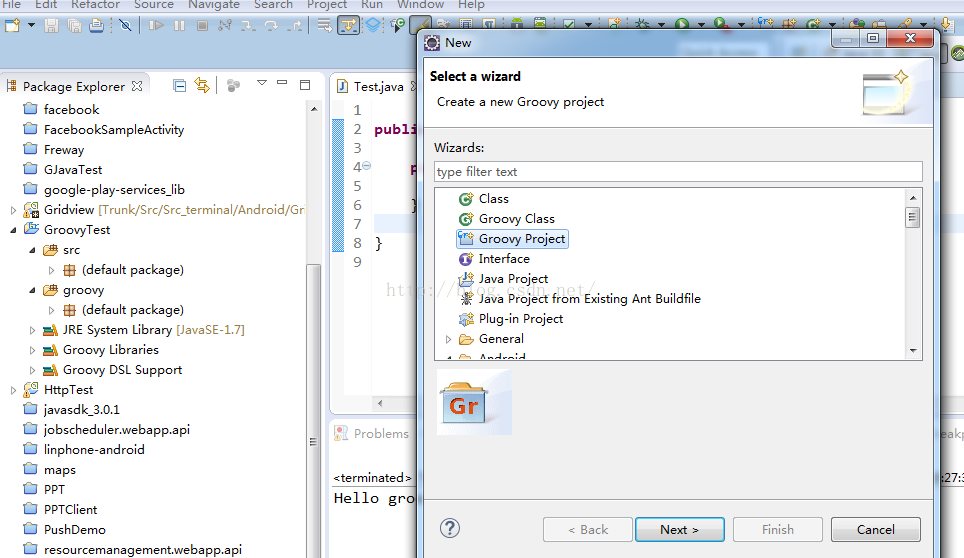
Groovy是一种动态语言，是从[**Java**](http://lib.csdn.net/base/javase)语言进化而来，可以说是Java的加强，它能运行于JVM上，具体的一些介绍可以自行谷歌了解，这里只是为了简单的看懂Groovy和基本使用。

开发环境：

Eclipse Groovy开发插件   http://dist.springsource.org/release/GRECLIPSE/e4.4   后边的e4.x要自己对照自己的eclipse版本，否则可能出现无法安装的问题，我的是4.4。



第一个Groovy程序：



右键，新建一个Groovy工程，添加一个groovy资源包（只是为了区分），然后在src下先新建一个JavaTest.java类，勾选main方法，关在方法里写下一行代码打印一个字符串 Hello groovy!

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. **<pre** name="code" class="html"**>**public class JavaTest {
2. public static void main(String[] args) {
3. System.out.println("Hello groovy!");
4. }
5. }

上面是一个很简单的Java类，运行之后将会在控制台输出Hello groovy!

然后同理，我们在groovy资源包下新建一个GroovyTest.groovy  选择新建－other-Grooovy Class

**[java]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. **class** GroovyTest {
2. **static** main(args) {
3. println "Hello groovy!"
4. }
5. }

右键使用Groovy运行后，也会在控制台上打印出 Hello groovy!

相比于JavaTest.java我们发现，GroovyTest.groovy。似乎和JavaTest.java长得很像，只不过是将JavaTest.java去掉了一些内容，如class的修饰符public ，打印的System.out ，参数args的类型String[]。下面我们对GroovyTest.groovy做一些改动：

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. class GroovyTest {
2. static main(args) {
3. System.out.println "Hello groovy!"
4. }
5. }

运行之后发现一样能输入Hello groovy!，  有意思，有意思，那我们再给args加上String[]的类型能，

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. class GroovyTest {
2. static main(String[] args) {
3. System.out.println.println "Hello groovy!"
4. }
5. }

发现一样能运行，并且同样输入Hello groovy! ，有趣了，是不是说我把java里面的代码都拷过来也能运行呢，这时变成这样：

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. public class GroovyTest {
2. public static void main(String[] args) {
3. System.out.println("Hello groovy!");
4. }
5. }

和你想的结果一样，他也准确无语的输出了Hello groovy!，哦，原来和我们看到的一些说明是正确的，groovy和java代码是一样的，它只不过是对java代码进行简化了。

不是说简化吗，假如我把GroovyTest类的代码简化成这样：

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. println "Hello groovy!"

运行后，发现居然可以输出来Hello groovy!

下面是一些说明：

Groovy类和java类一样，完全可以用标准java bean的语法定义一个Groovy类。但作为另一种语言，可以使用更Groovy的方式定义类，这样的好处是，可以少写一半以上的javabean代码。

（1）不需public修饰符

如前面所言，Groovy的默认访问修饰符就是public，如果Groovy类成员需要public修饰，则根本不用写它。

（2）不需要类型说明

同样前面也说过，Groovy也不关心变量和方法参数的具体类型。

（3）不需要getter/setter方法

在很多[ide](http://baike.baidu.com/subview/5775/5401660.htm)（如[eclipse](http://baike.baidu.com/view/23576.htm" \t "_blank)）早就可以为程序员自动产生getter/setter方法了，在Groovy中，不需要getter/setter方法--所有类成员（如果是默认的public）根本不用通过getter/setter方法引用它们（当然，如果一定要通过getter/setter方法访问成员属性，Groovy也提供了它们）。

（4）不需要构造函数

不再需要程序员声明任何构造函数，因为实际上只需要两个构造函数（1个不带参数的默认构造函数，1个只带一个map参数的[构造函数](http://baike.baidu.com/view/411124.htm" \t "_blank)--由于是map类型，通过这个参数可以构造对象时任意初始化它的成员变量）。

（5）不需要[return](http://baike.baidu.com/view/1350512.htm" \t "_blank)

Groovy中，方法不需要return来[返回值](http://baike.baidu.com/view/1289389.htm" \t "_blank)。

（6）不需要（）

Groovy中方法调用可以省略（）（构造函数除外）。

语法说明（有一些为个人见解，因为本人也没有深入研究，只是为了去理解AS的build.gradle，如有错误，请大家指正）：

因为groovy是动态语言，就跟js一样，Groovy对于类型的定义都用def这个关键字来声明，它在底层会自动判断当前的变量是一个什么类型：

我们在上面也看到GroovyTest.groovy文件中，只写了一个println "Hello groovy!" 也是可以正常运行的，所以以下我们都用最简化直接在文件中输入这些代码。

定义一个字符串

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. def aa="Hello"
2. println aa

假如把打印跟定义换个位置，将会报错

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. println aa
2. def aa="Hello"

由此看来,groovy也是由上到下来运行的，看起来像是把这些都放到main之中去运行。

//先定义再赋值

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. def str
2. str="r"
3. println str

定义一个整形

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. def num=10
2. println num

定义一个整形数组集合

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. def colum=[2,3,4,5]
2. println colum[0]

因为groovy已经重载了<<所以我们可以用<<来向集合加入一个值

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. def strArry=["jin","mu","shui"]
2. strArry.add("huo")
3. strArry **<<** "tu"
4. println strArry
5. //输出：[jin, mu, shui, huo, tu]

定义一个Map类型

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. def map=[key:"name",value:"mjk"]
2. println map["key"]
3. println map["value"]

输出：

name  
mjk

只要看到是  key:value都是map类型，因为key会自动转换成string，所以你可以写成def map=["key":"name","value":"mjk"]也可以写成def map=[key:"name",value:"mjk"]

如果想知道当前这个变量是一个什么类型可以用 .class来打印

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. def str="I am string"
2. println str.class

输出：class java.lang.String

循环的使用：

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. def arry=["a","b","c","d"]
2. for(a in arry){
3. println a
4. }

定义一个方法，接收两个参数，输出它们的和

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. def num=10
2. def method(a,b){
3. println a+b
4. }
5. method num,9

当然使用method(num,9)是一样的，只是去掉了括号,参数之间用逗号隔开

在定义一个map类型的时候，一定要加[]，如果不加会报错，而在方法调用时，可以不加[]，如果方法调用的map参数要加[]，则必须在外部加()

如：

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. //打印 map值
2. def printMap(map){
3. println map["key"]
4. }
5. printMap key:"name"

输出：name

//定义一个方法，打印接收到的值

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. def printStr(str){
2. println str
3. }
4. printStr "Hello groovy!"

定义闭包：

闭包是Groovy里面一个很重要的特性，这里要着重讲，

先看一个groovy对集合本身的实现了的一个包含闭包的方法each

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. def acoll = ["妈妈", "爸爸", "我"]
2. acoll.each{
3. println it
4. }

输出：

妈妈  
爸爸  
我

acoll是一个字符串的集合，这个集合里面有一个方法叫each，接收一个参数，这个参数是一个闭包，因为闭包也是一种类型，所以它也能像变量一样当参数传递给方法使用。我们这里简单的理解，闭包是一个用花括号{}括起来的代码块，这个代码块在被触发的时候调用。

上面each中的it是一个关键字，它保存了返回到这个闭包中的一个值。下面我们来用自己的方法实现这个each

//定义一个方法,这个方法接收一个集合参数跟一个闭包，当方法被调用时，遍历集合里面的值，并把值传回给闭包

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. def myEach(arry,block){
2. for(i in arry){
3. block(i)
4. }
5. }
6. def arr=["爸爸","妈妈","我"]//字符串集合
7. //调用myEach方法
8. myEach(arr,{
9. println it
10. });

输出：

爸爸  
妈妈  
我

原来流程是这样，当我们调用myEach方法时，要传入一个集合arr，和闭包。当方法,myEach在遍历集合的时候，每得到一个值 就会回调一下这个闭包并把值传给它，从这里看是不是觉得，这个闭包既像一个变量参数，又像一个方法呢。

定义一个闭包并调用它：

//因为闭包是一种类型，所以在定义的时候要用＝来给它赋值，这是跟方法有本质的区别

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. def block={
2. println "调用block"
3. }
4. block()
5. 输出：调用block
6. 定义一个闭包并传入一个参数
7. def block={
8. println "调用block"+it
9. }
10. block("－－并传入参数")

输出：调用block－－并传入参数

要是我们不想用it这个关键字，也是可以定义自己的关键字的，只是我们要使用->来告诉它，是用我们自己定义的变量来接收值，而不用it

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. def block={param -**>**
2. println "调用block"+param
3. }
4. block("－－并传入参数")
5. //两个参数
6. def block={param1,param2-**>**
7. println "调用block"+param1+param2
8. }
9. block("－－参数1","参数2")

下面我们定义一个block，这个block是用来打印接收到的值，目的是去遍历给定的集合

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. def block={data-**>**
2. println "value="+data
3. }
4. //定义一个方法,这个方法接收一个集合参数跟一个闭包，当方法被调用时，遍历集合里面的值，并把值传回给闭包
5. def myEach(arry,block){
6. for(i in arry){
7. block(i)
8. }
9. }
10. def arr=["爸爸","妈妈","我"]//字符串集合
11. //调用myEach方法
12. myEach(arr,block)

输出：

value=爸爸  
value=妈妈  
value=我

我们知道方法的括号是可以去掉的，如果上面调用方法时我们去掉括号就变成了：

//原来的是：myEach(arr,block)

myEach arr,block

而如果只是接收一个闭包的方法就可以写成：

myEach block

如果block是匿名的，那就成为：

myEach {

}

所以我们可以知道，集合的each方法是怎么实现遍历的，当我们的这个集合已经知道了里面的值，调用each时，只需要传入一个闭包参数，就可以得到每一个值。

一般来说，如果有多个参数，而最后一个为闭包的时候，调用方法的时候习惯写成：

myEach(param1,param2,param3){

}

实例：

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. //定义一个方法,这个方法接收一个集合参数跟一个闭包，当方法被调用时，遍历集合里面的值，并把值传回给闭包
2. def myEach(arry,block){
3. for(i in arry){
4. block(i)
5. }
6. }
7. def arr=["爸爸","妈妈","我"]//字符串集合
8. //调用myEach方法，并传入一个集合跟一个匿名的闭包
9. myEach (arr){data-**>**
10. println "value="+data
11. }

到此差不多把groovy里面的一些基本的语法说完了。下面讲一下gradle

二：gradle

Gradle是一个构建工具，感兴趣的可以去官网上了解，它的构建是基于一个一个的任务，task。我觉得把groovy弄懂了，对于gradle只是多去看一下开发指南就行，这里不再说明。

三：回到build.gradle

讲完了上面，这个时候可以把开篇的build.gradle贴到这里说了，直接在代码里面加解释：

**[html]** [view plain](http://blog.csdn.net/huanchengdao/article/details/51472628) [copy](http://blog.csdn.net/huanchengdao/article/details/51472628)

[在CODE上查看代码片](https://code.csdn.net/snippets/1692034)

1. //这一行，一看就知道调用了一个方法叫apply 传入一个map类型的参数，key是plugin ，plugin的值是"com.android.application"说明是一个android应用工程
2. //在gradle中有介绍把你自己的一组代码，构建做成一个plugin，如果把plugin的值改成"com.android.library"就说明这是一个库工程
3. apply plugin: 'com.android.application'
5. //android这个方法接收一个闭包，在闭包里，调用的compileSdkVersion,  buildToolsVersion等方法并给他们传值，并且还调用了defaultConfig这个方法来做一些配置，这个方法也是接收一个闭包，可以想象，通过闭包回调可以对值进行设置，当调用defaultConfig方法的时候，方法里面会先调用传入的这个闭包来初始化一些值，然后再进行别的其他操作。这也就是闭包的魅力所在。
6. android {
7. compileSdkVersion 21
8. buildToolsVersion "21.1.2"

11. defaultConfig {
12. applicationId "com.example.test"
13. minSdkVersion 17
14. targetSdkVersion 21
15. versionCode 1
16. versionName "1.0"
17. }
18. buildTypes {
19. release {
20. minifyEnabled false
21. proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
22. }
23. }
24. }

27. dependencies {
28. compile fileTree(dir: 'libs', include: ['\*.jar'])
29. compile 'com.android.support:appcompat-v7:21.0.3'
30. }

通过以上这些讲解，希望读者们能对build.gadle文件有一些认识，这样你也可以自己修改build.gadle文件来完成你自己想要的构建流程。

另外大家可以在这里找到关于Gradle一些内置的方法还有他们的功能，以及AS为android工程提供的一些方法：

http://google.github.io/android-gradle-dsl/current/

