

2.1 给project添加自定义属性(properties)



GuoYiheng 关注

2016.12.29 01:46:19 字数 815 阅读 5,734

除了默认属性,我们也可以给project^[1]添加自定义属性,Gradle通过执行project的build文件来完成对project的配置,我们可以在project对应的build.gradle文件中通过extra语句来定义一个变量作为自定义的属性^[2].比如,我们可以定义一个变量 `GsonVersion`,代表Gson的版本:

```
1 | ext {  
2 |     GsonVersion = '2.8.0'  
3 | }
```

当ext代码块里只有一句时,也可采用下面这种更快捷的写法:

```
1 | ext.GsonVersion = '2.8.0'
```

然后在 `dependencies` 代码块里调用它:

```
1 | dependencies {  
2 |     compile fileTree(include: ['*.jar'], dir: 'libs')  
3 |     androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {  
4 |         exclude group: 'com.android.support', module: 'support-annotations'  
5 |     })  
6 |     testCompile 'junit:junit:4.12'  
7 |  
8 |     //注意,groovy中如果句子中要引用变量需要使用双引号  
9 |     compile "com.google.code.gson:gson:$GsonVersion"  
10 | }
```

在上一篇 [1.7 配置仓库](#)中有提到登陆需要用户名和密码的maven仓库的问题,当时我们是直接把真实的用户名和密码写在了username和password后:

```
1 | repositories {  
2 |     maven {  
3 |         credentials {  
4 |             username '你的用户名'  
5 |             password '你的密码'  
6 |         }  
7 |         url 'maven仓库的URL地址'  
8 |     }  
9 | }
```

你当然也可以通过使用 `ext` 语句定义两个变量分别代表用户名和密码,但是,更多时候我们并不希望把我们真实的用户名和密码写在build.gradle文件里,比如当我们把代码分享到GitHub,当别人点开build.gradle就会知道我们的用户名和密码,这显然不是我们想要的.所以通常会把这些不想被别人看到的信息写在gradle.properties文件里,然后通过自定义的属性来引用他们.

首先,在项目根目录的gradle.properties文件里定义好user和password,并把真实的用户名和密码赋值给他们:

```
1 | USER_NAME = 'user_name'  
2 | PASS_WORD = 'password'
```

然后把build.gradle文件里的代码改为:

```
1 | repositories {  
2 |     maven {  
3 |         credentials {  
4 |             username USER_NAME
```

```

5 |         password PASS_WORD
6 |     }
7 |     url 'maven仓库的URL地址'
8 | }
9 | }

```

如果你看过[1.1 Android工程中的Gradle文件](#),那你可能就会想起来我在那里贴出的app模块的*build.gradle*里配置签名时也是采用的这种方法。

除了直接在*build.gradle*文件里定义属性的值,我们也可以在命令行使用**-P**完成赋值:

```
1 | >gradlew -PUSER_NAME=user_name -PPASS_WORD=password assemble
```

通过下面的代码,可以对整篇文章内容进行更直观的展示和更深入的了解:

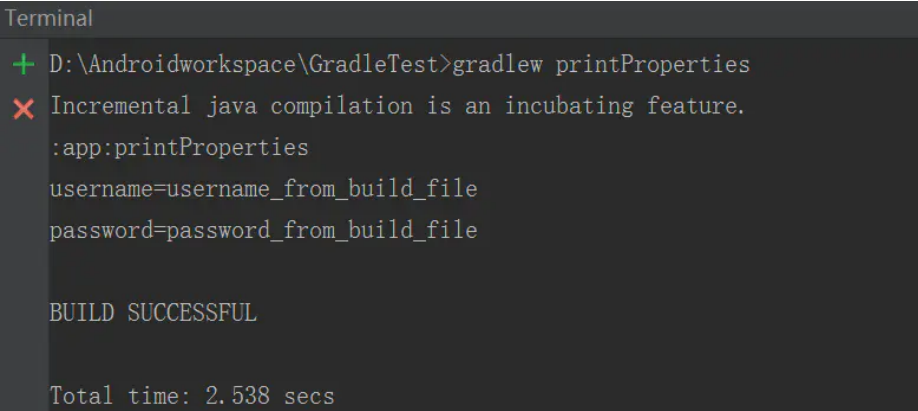
```

1 | ext {
2 |     if (!project.hasProperty('USER_NAME')) {
3 |         USER_NAME = 'username_from_build_file'
4 |     }
5 |     if (!project.hasProperty('PASS_WORD')) {
6 |         PASS_WORD = 'password_from_build_file'
7 |     }
8 | }
9 |
10 | task printProperties() {
11 |     doLast {
12 |         println "username=$USER_NAME"
13 |         println "password=$PASS_WORD"
14 |     }
15 | }

```

在上面的代码里我们首先分别判断是否存在 **USER_NAME** 和 **PASS_WORD** 属性,如果不存在,分别赋值为 **username_from_build_file** 和 **password_from_build_file**,然后定义了一个打印属性的值的任务 **printProperties** [\[3\]](#).

如果我们不在*gradle.properties*文件里定义变量 **USER_NAME** 和 **PASS_WORD** 并赋值,执行结果如下:



```

Terminal
+ D:\Androidworkspace\GradleTest>gradlew printProperties
✗ Incremental java compilation is an incubating feature.
:app:printProperties
username=username_from_build_file
password=password_from_build_file

BUILD SUCCESSFUL

Total time: 2.538 secs

```

G 1.png

如果我们在*gradle.properties*文件里定义变量 **USER_NAME** 和 **PASS_WORD** 并赋值:

```

1 | USER_NAME=username_from_gradle_properties
2 | PASS_WORD=password_from_gradle_properties

```

执行结果如下:

```
Terminal
+ D:\Androidworkspace\GradleTest>gradlew printProperties
✗ Incremental java compilation is an incubating feature.
:app:printProperties
username=username_from_gradle_properties
password=password_from_gradle_properties
```

G2.png

通过命令行对property赋值具有最高优先级,所以如果当我们在命令行对 **USER_NAME** 和 **PASS_WORD** 再次赋值,执行结果如下:

```
Terminal
+ D:\Androidworkspace\GradleTest>gradlew -PUSER_NAME=user_name -PPASS_WORD=password printProperties
✗ Incremental java compilation is an incubating feature.
:app:printProperties
username=user_name
password=password

BUILD SUCCESSFUL

Total time: 2.512 secs
```

G3.png

- 1. 这里的project包括但不限于整个Project,关于project参考[0.1 project和task的概念及构建过程](#)
- 2. 如果在定义变量时加上了关键字def或者其他类型的关键字,就表明该变量只是当前build文件的一个局部变量,我在这里没有加任何关键字,这个没有被定义类型的变量就会被看做project对象的一个属性(attribute),可以被该project的所有子project引用.
- 3. 关于自定义任务(task),我会在后面详细介绍,这里你只需要知道这个任务的作用即可,.

1人点赞 >

Gradle学习