

Gradle学习系列之六——使用Java Plugin

在[本系列的上篇文章](#)中，我们讲到了如何自定义Property，在本篇文章中，我们将讲到如何使用java Plugin。

请通过以下方式下载本系列文章的Github示例代码：

```
git clone https://github.com/davenkin/gradle-learning.git
```

Gradle最常用的Plugin便是java Plugin了。和其他Plugin一样，java Plugin并没有什么特别的地方，只是向Project中引入了多个Task和Property。当然，java Plugin也有比较与众不同的地方，其中之一便是它在项目中引入了构建生命周期的概念，就像Maven一样。但是，和Maven不同的是，Gradle的项目构建生命周期并不是Gradle的内建机制，而是由Plugin自己引入的。

(一) java Plugin引入的主要Task

执行“gradle build”，我们已经可以看到java Plugin所引入的主要Task：

```
compileJava
processResources
classes
jar
assemble
compileTestJava
processTestResources
testClasses
```

我翻译的书：



我读过的书：



昵称：无知者云
园龄：4年8个月
粉丝：88
关注：3

我的标签

社会(17)

诗歌(14)

java(13)


gradle(10)

j2ee(9)

```
:test
:check
:build

BUILD SUCCESSFUL

Total time: 4.813 secs
```

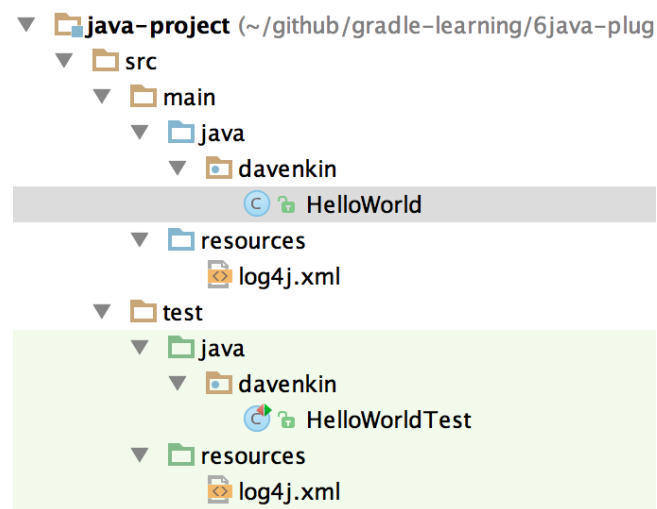


build也是java Plugin所引入的一个Task，它依赖于其他Task，其他Task又依赖于另外的Task，所以有了以上Task执行列表。以上Task执行列表基本上描述了java Plugin向项目中所引入的构建生命周期概念。

除了定义众多的Task外，java Plugin还向Project中加入了一些额外的Property。比如，sourceCompatibility用于指定在编译Java源文件时所使用的Java版本，archivesBaseName用于指定打包成Jar文件时的文件名称。

(二) Java项目的目录结构

在默认情况下，Gradle采用了与Maven相同的Java项目目录结构：



关于Maven标准目录结构，请参考[Maven官网](#)。当然，跟Maven一样，以上只是默认的目录结构，我

java ee(9)
事务(8)
transaction(5)
Ruby(3)
数据库(3)
更多

随笔档案(65)
2014年7月 (1)
2014年3月 (1)
2014年2月 (1)
2013年11月 (10)
2013年6月 (2)
2013年4月 (1)
2013年3月 (1)
2013年2月 (12)
2013年1月 (1)
2012年12月 (2)
2012年11月 (1)
2012年10月 (1)
2012年8月 (1)
2012年7月 (2)
2012年6月 (2)
2012年5月 (3)
2012年4月 (10)


们可以通过配置来修改这些目录结构。

(三) 配置已有source set


Gradle在采用了Maven目录结构的同时，还融入了自己的一些概念，即source set。对于上图中的目录结构，Gradle实际上为我们创建了2个source set，一个名为main，一个名为test。

请注意，这里的source set的名字main与上图目录结构中的main文件夹并无必然的联系，只是在默认情况下，Gradle为了source set概念到文件系统目录结构的映射方便，才采用了相同的名字。对于test，也是如此。我们完全可以在build.gradle文件中重新配置这些source set所对应的目录结构，同时，我们还可以创建新的source set。

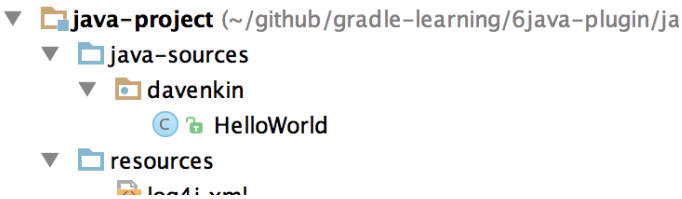
从本质上讲，Gradle的每个source set都包含有一个名字，并且包含有一个名为java的Property和一个名为resources的Property，他们分别用于表示该source set所包含的Java源文件集合和资源文件集合。在实际应用时，我们可以将他们设置成任何目录值。比如，我们可以重新设置main的目录结构：



```
sourceSets {
    main {
        java {
            srcDir 'java-sources'
        }
        resources {
            srcDir 'resources'
        }
    }
}
```

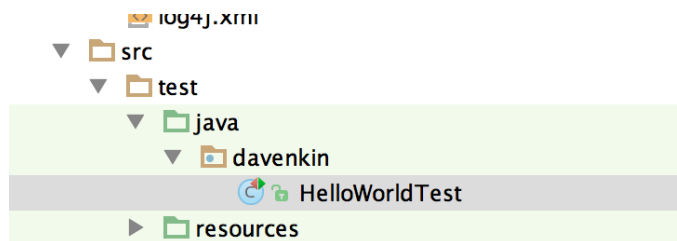


此时所对应的项目目录结构如下：



2012年4月 (10)
2012年3月 (4)
2012年2月 (7)
2011年12月 (1)
2011年10月 (1)

最新评论
1. Re:毕业两年返校随想
楼主在西电？
--wxyjuly
2. Re:Gradle学习系列之三——读懂Gradle语法
mark
--headchen
3. Re:Gradle学习系列之十一——自定义Plugin（本系列完）
DateAndTimePluginExtension 这个里面的timeFormat 是不是没有用，为什么样定义呢？
--灵感仰
4. Re:Gradle学习系列之一——Gradle快速入门
您好，图中这个是否应该是taskA？
--Mr.XuFeng
5. Re:Gradle学习系列之十一——自定义Plugin（本系列完）
博主你真是太棒了！循序渐进通俗易懂。我这两天看gradle看地稀里糊涂的，终于在你



我们重新设置了main的目录结构，而对于test，我们保留了Gradle默认的目录结构。

(四) 创建新的source set

要创建一个新的source set也是非常简单的，比如，我们可以创建一个名为api的source set来存放程序中的接口类：

```
sourceSets {  
    api  
}
```

当然，以上配置也可以与main放在一起。在默认情况下，该api所对应的Java源文件目录被Gradle设置为`${path-to-project}/src/api/java`，而资源文件目录则被设置成了`${path-to-project}/src/api/resources`。我们也可以像上面的main一样重新对api的目录结构进行配置。

Gradle会自动地为每一个新创建的source set创建相应的Task，创建规律为：对于名为mySourceSet的source set，Gradle将为其创建`compile<mySourceSet>Java`、`process<mySourceSet>Resources`和`<mySourceSet>Classes`这3个Task。对于这里api而言，Gradle会为其创建名为`compileApiJava`、`processApiResource`和`apiClasses` Task。我们可以在命令行中执行`"gradle apiClasses"`。

你可能会注意到，对于main而言，Gradle并没有相应的`compileMainJava`，原因在于：由于main是Gradle默认创建的source set，并且又是及其重要的source set，Gradle便省略掉了其中的“Main”，而是直接使用了`compileJava`作为main的编译Task。对于test来说，Gradle依然采用了`compileTestJava`。

通常的情况是，我们自己创建的名为api的source set会被其他source set所依赖，比如main中的类需要实现api中的某个接口等。此时我们需要做两件事情。第一，我们需要在编译main之前对api进行编译，即编译main中Java源文件的Task应该依赖于api中的Task：

这弄明白了。谢谢你！

--无意义

阅读排行榜

1. Gradle学习系列之一——Gradle快速入门(32401)
2. Gradle学习系列之二——创建Task的多种方法(10425)
3. Gradle学习系列之七——依赖管理(9412)
4. 重温大师经典：Martin Fowler的持续集成(9320)
5. Gradle学习系列之三——读懂Gradle语法(9242)

评论排行榜

1. 重温大师经典：Martin Fowler的持续集成(12)
2. 5个小时写一个扑克牌游戏——金钩钓鱼(10)
3. Gradle学习系列之二——创建Task的多种方法(6)
4. 用Spring MVC3+Ant+Jenkins+SVN+Tomcat做一个持续集成例子(5)
5. Gradle学习系列之九——自定义Task类型(4)

推荐排行榜

1. Java加载资源文件的两种方法(8)

```
classes.dependsOn apiClasses
```

第二，在编译main时，我们需要将api编译生成的class文件放在main的classpath下。此时，我们可以对main和test做以下配置：

```
sourceSets {  
    main {  
        compileClasspath = compileClasspath + files(api.output.classesDir)  
    }  
    test {  
        runtimeClasspath = runtimeClasspath + files(api.output.classesDir)  
    }  
}
```

之所以需要对test的runtimeClasspath进行设置，是因为在运行测试时我们也需要加载api中的类。

在[下一篇文章](#)中，我们将讲到如何管理依赖。

标签: [gradle](#)

好文要顶

关注我

收藏该文



无知者云

关注 - 3

粉丝 - 88

[+加关注](#)

1

推荐

0

反对

(请您对文章做出评价)

« 上一篇: [Gradle学习系列之五——自定义Property](#)

» 下一篇: [Gradle学习系列之七——依赖管理](#)

2. Gradle学习系列之一——Gradle快速入门(7)

3. 重温大师经典：Martin Fowler的持续集成(6)


4. Java事务之一——Java事务的基本问题(5)

5. 用Spring MVC3+Ant+Jenkins+SVN+Tomcat做一个持续集成例子(4)

#1楼 2014-08-14 15:48 未来之风 

真想知道楼主是怎么学习的gradle，才能对gradle DSL的各个细节理解的这么透彻的。是自己根据对groovy的相关经验推导过来的？还是根据gradle文档学习到的？还是有这方面的较好的英文文章来帮助理解？真想知道这“渔”的方法额。


支持(0) 反对(0)

#2楼[楼主] 2014-08-14 20:05 无知者云 

@ 未来之风

我是这么想的，首先gradle使用的是groovy，也就是它必须遵循groovy的语法，带着这个问题问题就迎刃而解了。另外，我们也不用在groovy上有多么精通。动态语言嘛，闭包很重要，对于DSL尤其如此，所以我就专门去研究了一下groovy的闭包特性，然后再连猜带猜般的自己试，结果就出来了。另外，gradle的官方英文文档，我肯定也是读过许多的，还有一本书挺好：<http://www.amazon.in/Gradle-Effective-Implementation-Hubert-Ikkink/dp/1849518106>。

支持(0) 反对(0)

#3楼 2014-08-16 18:58 未来之风 

@ 程序员Davenkin

恩，你这么一层层逻辑推导下来，看来就是得这么一步步的来。那本书从名字看起来相关性比较大。你这系列的文章都看完了，现在差不多对gradle的以project为基础的构成理念有所了解了，现在至少知道在那里下手了。谢谢。

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)



注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

【推荐】50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库

【推荐】融云即时通讯云 - 豆果美食、Faceu等亿级APP都在用



最新IT新闻:

- DuckDuckGo宣布与雅虎合作
 - 戴尔抛弃安卓系统，今后将集中精力生产二合一Windows设备
 - 趣味科普：树木也要睡觉吗？
 - 特斯拉正在测试8.0版固件：界面大改版+新Autopilot功能
 - 苹果联手NASA推出音乐短片庆祝“朱诺号”太空任务
- » 更多新闻...



最新知识库文章:

- 编程同写作，写代码只是在码字
 - 遇见程序员男友
 - 设计师的视觉设计五项修炼
 - 我听到过的最精彩的一个软件纠错故事
 - 如何避免软件工程中最昂贵错误的发生
- » 更多知识库文章...