Gradle学习系列之三——读懂Gradle语法

在本系列的上篇文章中，我们讲到了创建Task的多种方法，在本篇文章中，我们将学习如何读懂Gradle。

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是一种声明式的构建工具。在执行时，Gradle并不会一开始便顺序执行build.gradle文件中的内容，而是分为两个阶段，第一个阶段是配置阶段，然后才是实际的执行阶段。在配置阶段，Gradle将读取所有build.gradle文件的所有内容来配置Project和Task等，比如设置Project和Task的Property，处理Task之间的依赖关系等。

虽然很多时候我们只需要照着网上的例子写自己的DSL语句就行了，但是此时我们所知道的也就只有这么多了。如果我们能够了解Gradle DSL的内部工作机制，那么我们便可以达到

举一反三的效果。在前面的文章中我们讲到，Gradle的DSL只是Groovy语言的内部DSL，也即必须遵循Groovy的语法规则。现在，让我们先来看看以下非常简单的Task：

```
task showDescription1 << {
   description = 'this is task showDescription'
   println description
}
```

```
task showDescription2 << {
   println description
}
showDescription2.description = 'this is task showDescription'


task showDescription3 << {
   println description
}

showDescription3 {
   description = 'this is task showDescription'
}
```

以上3个Task完成的功能均相同，即先设置Task的description属性，在将其输出到命令行。但是，他们对description的设置方式是不同的。对于showDescription1，我们在定义一个Task的同时便设置description；对于showDescription2，其本身便是Project的一个Property；而对于showDescription3，我们是在一个和它同名的方法中设置description。

事实上，对于每一个Task，Gradle都会在Project中创建一个同名的Property，所以我们可以将该Task当作Property来访问，showDescription2便是这种情况。另外，Gradle还会创建一个同名的方法，该方法接受一个闭包，我们可以使用该方法来配置Task，showDescription3便是这种情况。

要读懂Gradle，我们首先需要了解Groovy语言中的两个概念，一个Groovy中的Bean概念，一个是Groovy闭包的delegate机制。

Groovy中的Bean和Java中的Bean有一个很大的不同，即Groovy为每一个字段都会自动生成getter和setter，并且我们可以通过像访问字段本身一样调用getter和setter，比如：

```
class GroovyBeanExample {
   private String name
}

def bean = new GroovyBeanExample()
```


我的标签

社会(17)

诗歌(14)

java(13)

gradle(10)

j2ee(9)

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)

```
bean.name = 'this is name'
println bean.name
```

我们看到，GroovyBeanExample只定义了一个私有的name属性，并没有getter和setter。但是在使用时，我们可以直接对name进行访问，无论时读还是写。事实上，我们并不是在直接访问name属性，当我们执行"bean.name = 'this is name'"时，我们实际调用的是"bean.setName('this is name')"，而在调用"println bean.name"时，我们实际调用的是"println bean.getName()"。这里的原因在于，Groovy动态地为name创建了getter和setter，采用像直接访问的方式的目的是为了增加代码的可读性，使它更加自然，而在内部，Groovy依然是在调用setter和getter方法。这样，我们便可以理解上面对showDescription2的description设置原理。

另外，Gradle大量地使用了Groovy闭包的delegate机制。简单来说，delegate机制可以使我们将一个闭包中的执行代码的作用对象设置成任意其他对象。比如：

```
class Child {
   private String name
}

class Parent {
   Child child = new Child();

   void configChild(Closure c) {
      c.delegate = child
      c.setResolveStrategy Closure.DELEGATE_FIRST
      c()
   }
}

def parent = new Parent()
parent.configChild {
name = "child name"
}

println parent.child.name
```

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)
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——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

在上面的例子中，当我们调用configChild()方法时，我们并没有指出name属性是属于Child的，但是它的确是在设置Child的name属性。事实上光从该方法的调用中，我们根本不知道name是属于哪个对象的，你可能会认为它是属于Parent的。真实情况是，在默认情况下，name的确被认为是属于Parent的，但是我们在configChild()方法的定义中做了手脚，使其不再访问Parent中的name（Parent也没有name属性），而是Child的name。在configChild()方法中，我们将该方法接受的闭包的delegate设置成了child，然后将该闭包的ResolveStrategy设置成了DELEGATE_FIRST。这样，在调用configChild()时，所跟闭包中代码被代理到了child上，即这些代码实际上是在child上执行的。此外，闭包的ResolveStrategy在默认情况下是OWNER_FIRST，即它会先查找闭包的owner（这里即parent），如果owner存在，则在owner上执行闭包中的代码。这里我们将其设置成了DELEGATE_FIRST，即该闭包会首先查找delegate（本例中即child），如果找到，该闭包便会在delegate上执行。对于上面的showDescription3，便是这种情况。当然，实际情况会稍微复杂一点，比如showDescription3()方法会在内部调用showDescription3的configure()方法，再在configure()方法中执行闭包中的代码。

你可能会发现，在使用Gradle时，我们并没有像上面的parent.configChild()一样指明方法调用的对象，而是在build.gradle文件中直接调用task()，apply()和configuration()方法等，这是因为在没有说明调用对象的情况下，Gradle会自动将调用对象设置成当前Project。比如调用apply()方法和调用project.apply()方法的效果是一样的。查查Gradle的Project文档，你会发现这些方法都是Project类的方法。

另外举个例子，对于configurations()方法（它的作用我们将在后面的文章中讲到），该方法实际上会将所跟闭包的delegate设置成ConfigurationContainer，然后在该ConfigurationContainer上执行闭包中的代码。再比如，dependencies()方法，该方法会将所跟闭包的delegate设置成DependencyHandler。

还有，Project还定义了configure(Object object,Closure configureClosure)方法，该方法是专门用来配置对象的（比如Task），它会将configureClosure的delegate设置成object，之后configureClosure中的执行代码其实是在object上执行的。和Groovy Bean一样，delegate机制的一个好处是可以增加所创建DSL的可读性。

在下一篇文章中，我们将讲到如何进行增量式构建。

标签: gradle

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(32392)

2. Gradle学习系列之二——创建Task的多种方法(10420)

3. Gradle学习系列之七——依赖管理(9404)

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(9238)

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

好文要顶 关注我 收藏该文

无知者云
关注 - 3
粉丝 - 88
+加关注

0 推荐 0 反对

(请您对文章做出评价)

« 上一篇：Gradle学习系列之二——创建Task的多种方法

» 下一篇：Gradle学习系列之四——增量式构建

posted @ 2013-11-12 09:19 无知者云 阅读(9239) 评论(2) 编辑 收藏

#1楼 2015-09-22 19:56 身带吴钩

mark

支持(0) 反对(0)

#2楼 2016-03-19 22:00 headchen

mark

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。

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)
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)

最新IT新闻:
· Linux Mint 18发布，基于Ubuntu的发行版
· 移植.NET Core计划，整合各平台变得更简单了！
· 太机智了：国外用户利用VR视频逃过盗版审查
· 微软研发出智能捕蚊器 可预防80多种疾病传播
· 宇宙竟真的存在多维空间？
» 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...