**Gradle整理**

|  |  |  |  |
| --- | --- | --- | --- |
| 时间 | 修改人 | **C**reate/**M**odify | 备注 |
| 2014-09-16 | 邓建新 | C | 创建文档 |

# 概述

## 定义

在Java构建工具的世界里，先有了Ant，然后有了Maven。Maven的CoC[1]、依赖管理以及项目构建规则重用性等特点，让Maven几乎成为Java构建工具的事实标准。然而，冗余的依赖管理配置、复杂并且难以扩展的构建生命周期，都成为使用Maven的困扰。

[Gradle](http://www.gradle.org/)作为新的构建工具，获得了[2010 Springy](http://www.springsource.org/node/2871)大奖，并入围了2011的[Jax](http://jaxconf.com/)最佳Java技术发明奖。它是基于Groovy语言的构建工具，既保持了Maven的优点，又通过使用Groovy定义的DSL[2]，克服了 Maven中使用XML繁冗以及不灵活等缺点。在Eugene Dvorkin撰写的文章《[最让人激动的5个Java项目》](http://eugenedvorkin.com/five-exciting-java-projects-you-should-know-about/)中，他是这样介绍Gradle的：

“工程自动化是软件项目成功的必要条件，而且它应该实现起来简单、易用、好玩。构建没有千篇一律的方法，所以Gradle没有死板的强加方法于我们，尽管你会认为查找和描述方法很重要，然而Gradle对于如何描述有着非常好的支持。我不认为工具能够拯救我们，但是Gradle能给你所需要的自由，你可以利用Gradle构建易描述的、可维护的、简洁的、高性能项目”。

Gradle中有两个基本的概念：project和task。每个Gradle的构建由一个project构成，它代表着需要被构建的组件或者构建的整个项目。每个project由一个或者多个task组成。task代表着Gradle构建过程中可执行的最小单元。例如当构建一个组件时，可能需要先编译、打包、然后再生成文档或者发布等，这其中的每个步骤都可以定义成一个task。

## 安装

下载Gradle包

配置Gradle环境： GRADLE\_HOME = ； path = path;%GRADLE\_HOME%\bin

建议安装 Groovy，并配置环境变量。

下载地址：

## 开发工具

使用intellIJ构建gradle项目。要求intelliJ13以上，我使用的是13.1.3, 在测试过程中使用12的版本未能创建gradle项目。

新建一个gradle项目：

1. 新建一个project，选择gradle，如图1-1

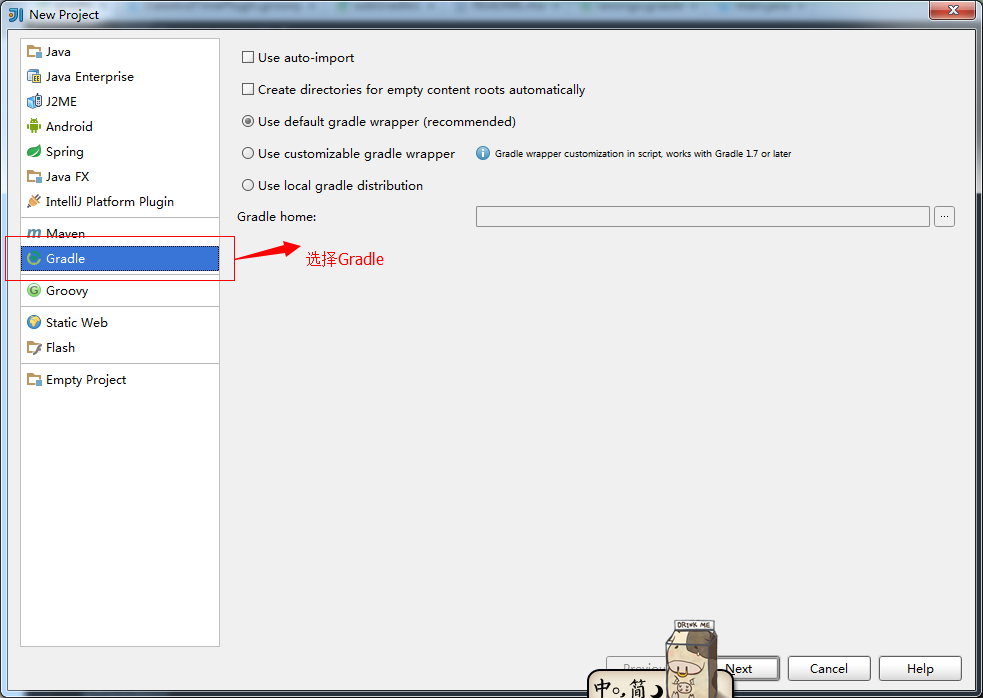


图 1-1

1. 新建成功，显示gradle项目文档结构，如图1-2

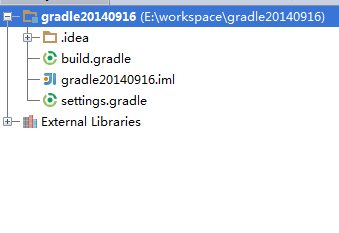


图1-2

gradle包含两个文件：build.gradle和settings.gradle。build.gradle类似ant的build文件，而settings.gradle为项目的配置文件。

用户可以使用-b xxx.gradle来指定build文件。

# 二、Task

## 构建并运行第一个task

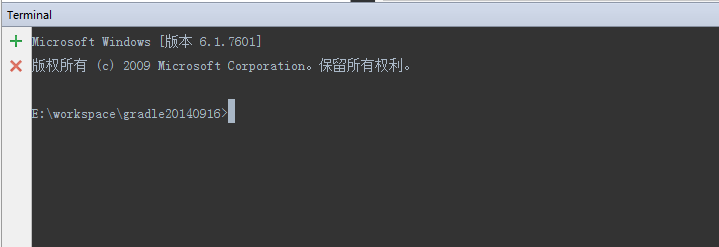
下面，让我们新建一个build.gradle文件（如果有，则直接编辑），然后输入如下内容:

  task hello {  
      doLast{  
           println "hello world"  
      }  
 }

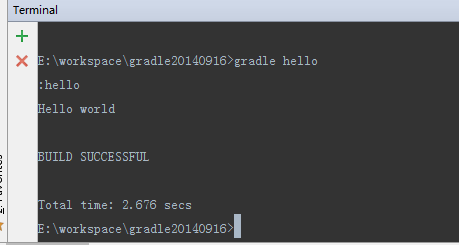
运行，介绍两种运行方式：

1. 命令行：

使用intelliJ带的控制台，或者windows控制台，切换至项目路径，如图2-1.



运行命令 gradle hello



1. IEDA，这里不做介绍，用户自己摸索。

## 定义属性

在Gradle中，我们可以定义以下三种属性并使用它们：

**1)System Properties**

System Properties 实际是指的JVM的system properties。我们知道，在运行java程序时，可以使用-D来设置Java的系统变量，在Gradle中，你也可以做同样的事情。比如

gradle xxx -DmySystemProp=xxxx

同时，在build.gradle中，应该这样使用这个变量：

task printSysProps << {  
  println System.properties['system']  
}

2)Project Properties

Project Properties是Gradle专门为Project定义的属性。它的最大优点在于当运行gradle的时候，我们可以使用-P来设置它的值。比如，

gradle xxx -PmyProjectProp=xxxxx

而在build.gradle中，可以这样使用这个变量：

task printProps << {  
    if (project.hasProperty('commandLineProjectProp')) {    
        println commandLineProjectProp  
    }    
}

同时，当我们执行gradle properties查看属性列表时，这个变量的名称以及值会显示在结果中。

***3)Ext(ra) Properties***

    另外，我们还可以为Project或者Task定义Ext属性，也称动态属性，我们必须使用关键字ext(对应ExtraPropertiesExtension的实例)去定义动态属性。从这点可以看出，Gradle已经为我们设计了很多不同的类，去做不同的事情，我们只需要遵循Convention，使用他们即可。如果忘记写ext关键字，gradle运行时则会提示:

"Dynamic properties are deprecated...."。这是因为以前版本的gradle定义动态属性时，不需要加ext关键字的。

对于Project和Task而言，动态属性定义的方式完全一样，只是作用域不一样。  
当定义完成后，我们就可以使用project.prop 或者 task.prop来访问这些动态属性了。下面让我们看一个例子：

ext.projectProperties="ext projectProperties-value"  
task printExtProps << {  
  ext.taskProperties="ext.task.properties-value"  
  if (project.hasProperty('projectProperties')){  
    println "ext.projectProperties values is " + projectProperties    
  }  
  if (printExtProps.hasProperty('taskProperties')){  
    println "task has defined ext.taskProperties value is " + taskProperties    
  }  
}

**注意：**，对于ext定义的动态属性，并不能通过外部的方式修改它的值，只能在build.gradle中去设置或者修改它的值。

同时，如果是为project定义的ext动态属性，也会显示在gradle properties的结果中。

## Task的依赖

通过task的dependsOn定义依赖关系，如

task hello << {  
    println 'Hello world!'  
}  
task intro(dependsOn: hello) << {  
    println "I'm Gradle"   
}

执行 "gradle intro"，结果将是:

Hello World  
I'm Gradle

## Copy Task

## Task包含的属性

|  |  |  |
| --- | --- | --- |
| Name | 类型 | 描述 |
| dependsOn | Array | 描述task的依赖关系，允许添加多个依赖。 |
| description | String | 描述信息 |
| doFirst | Action | Task首先运行的一段代码。 |
| doLast | Action | Task最后运行的一段代码，注册是可以使用<< 替代。 |

# 三、 Project

Gradle除了借鉴了ant或者maven的继承的方式定义子项目，也提供了一种更为方便的集中配置的方式，大大减少了构建带来的复杂度。

## 多项目构建

# 附录1 API

|  |  |  |
| --- | --- | --- |
| Name | 参数 | 描述 |
| file | String|Object | 新建一个File对象。 |
| files | String|Object|Array | 新建一个FileCollection对象。 |
| fileTree | Object|String | 新建一个FileTree对象 |
| ZipTree | Object|String | 新建一个FileTree对象。  使用该方法时，gradle会将zip文件解压到%GRADLE\_HOME%/build/tmp/expandedArchives/ |
| tarTree | Object|String | 新建一个FileTree对象 |
| getDestDir | None | 获取目标文件夹 |
|  |  |  |