**Gradle教程**

[Ant](http://www.yiibai.com/ant/)和[Maven](http://www.yiibai.com/maven/)共享在[Java](http://www.yiibai.com/java/)市场上相当大的成功。ANT是在2000年发布了第一个版本的工具，它是基于程序编程思想的发展。 后来，人们在 Apache-Ivy的帮助下，网络接受插件和依赖管理的能力有所提升。但主要缺点是使用XML作为一种格式来写构建脚本。XML是分层的，不利于程序的编程，而且当XML文件变大以后变得难以管理。

Maven在2004年推出的，它比ANT有一个很大的改进。它改变了结构并且继续使用XML编写生成规范。Maven的依赖约定和能够通过网络下载依赖关系。Maven的主要好处是它的生命周期。虽然接连的多个项目生命周期相同，这是以灵活性为代价的。 Maven也面临着依赖管理的一些问题。它不会在同一库版本之间处理好矛盾，复杂的定制构建脚本实际上Maven比ANT更难写。

最后，[Gradle](http://www.yiibai.com/gradle/)于2012年发布，带来了一些更高效的特点。

Gradle是一个基于Apache Ant和Apache Maven概念的项目自动化建构工具。它使用一种基于Groovy的特定领域语言(DSL)来声明项目设置，抛弃了基于XML的各种繁琐配置。 面向Java应用为主。当前其支持的语言限于Java、Groovy和Scala，计划未来将支持更多的语言。

## 下面是一些Gradle的功能

* 按约定声明构建和建设；
* 强大的支持多工程的构建；
* 强大的依赖管理（基于Apache Ivy），提供最大的便利去构建工程；
* 全力支持已有的 Maven 或者Ivy仓库基础建设；
* 支持传递性依赖管理，在不需要远程仓库和pom.xml和ivy配置文件的前提下；
* 基于groovy脚本构建，其build脚本使用groovy语言编写；
* 具有广泛的领域模型支持构建；
* 深度 API；
* 易迁移；
* 自由和开放源码，Gradle是一个开源项目，基于 [ASL](http://www.gradle.org/license?_ga=1.156736505.391095409.1474008833) 许可。

## 为什么使用Groovy?

完整的Gradle API是使用Groovy语言设计的。这是基于XML内部 DSL 的优点。Gradle是其核心的通用构建工具; 它的主要焦点是Java项目。在这些项目中，团队成员要熟悉Java，这是为了更好的构建透明，给所有团队成员的项目。

类似于 [Python](http://www.yiibai.com/python)，Groovy或[Ruby](http://www.yiibai.com/ruby/)语言是最好的构建框架。为什么Groovy被选中？这是因为它为使用Java的人提供了迄今为止最大的透明度。Groovy的基本语法与Java是一样的。

本站代码下载：[http://www.yiibai.com/siteinfo/download.html](http://www.yiibai.com/siteinfo/download.html?from=article)

# Gradle安装配置（Windows）

在本教程中，我们将学习 Gradle 的安装，对于一个初学者，有时安装开发环境也是一个比较麻烦的问题。如果按照 Gradle 官方网站的说明安装，则可能会遇到一些麻烦，有时还要在互联网上做一些搜索，查找为什么进入命令提示符输入 gradle -v 得不到任何东西。下面是完整的步骤来安装 Gradle ，为我们后续的 Gradle学习的开发环境作好准备！

## 前提条件

* Windows 8 (或更高版本，本教程是 Windows 10 64bit)
* Java JDK必须安装（本教程 JDK1.8）

执行以下命令来验证 Java 安装。在写本教程之前，我的系统中已经安装了JDK1.8。

C:/Users/Administrator>java -version

如果命令执行成功，应该会得到下面的输出。

C:/Users/Administrator>java -version

java version "1.8.0\_65"

Java(TM) SE Runtime Environment (build 1.8.0\_65-b17)

Java HotSpot(TM) 64-Bit Server VM (build 25.65-b01, mixed mode)

如果没有安装 JDK，可以自行安装，参考 - [http://www.yiibai.com/java/java\_environment\_setup.html](http://www.yiibai.com/java/java_environment_setup.html" \o "http://www.yiibai.com/java/java_environment_setup.html)

## 安装步骤说明

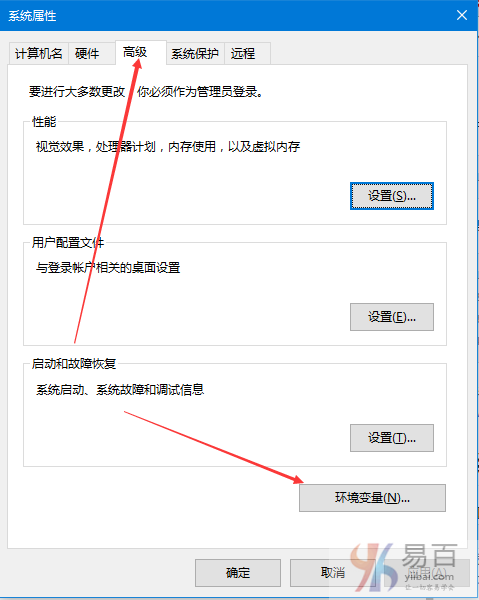
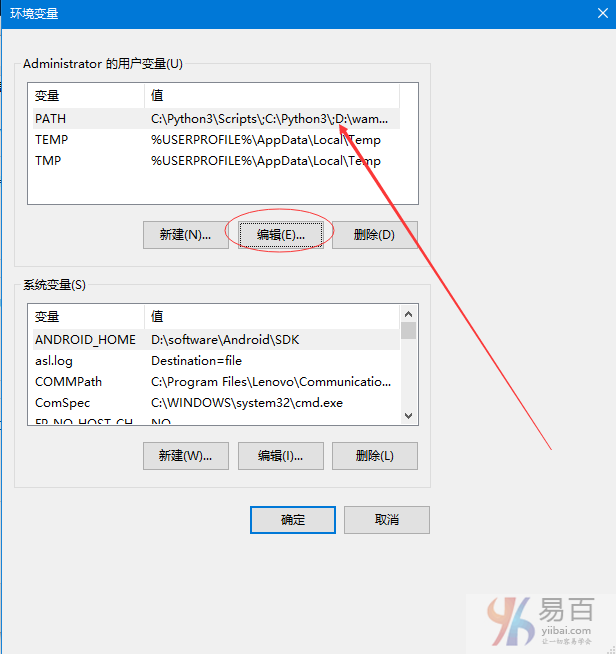
从网站上下载 Gradle： <http://www.gradle.org/downloads> ， 当前最新版本是 GRADLE 3.1 。

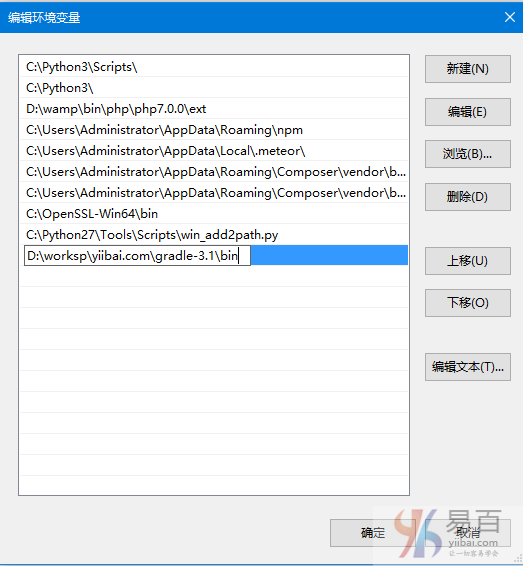
**下载gradle并解压缩**

下载后将它解压缩。 通常把这样的东西放入到我的开发目录中。如：D:workspyiibai.com 目录下。

**配置环境变量**

现在，在桌面上右键点击“电脑”（又名“我的电脑”或”此电脑”），然后选择“属性”，打开 “高级” 选项卡，如下图所示：

  
在上图中点击“环境变量”， 如下图所示 -  


在上图中点击“编辑”后新建一个环境变量项， 把解压的 D:workspgradle-3.1in 目录放入输入框中并点击 “确定”。如下图所示 -  


将 Gradle 和 Java JDK 的 bin 目录加入到环境变量之后，单击“确定”（有多次回退点击“确定”）完成。

**检测gradle是否安装配置成功**

现在，让我们来测试一下安装的情况。打开命令提示符，然后键入gradle -v，将看到类似下面的内容：

C:/Users/Administrator>gradle -v

------------------------------------------------------------

Gradle 3.1

------------------------------------------------------------

Build time: 2016-10-02 10:53:53 UTC

Revision: 13f38ba699afd86d7cdc4ed8fd7dd3960c0b1f97

Groovy: 2.4.7

Ant: Apache Ant(TM) version 1.9.6 compiled on June 29 2015

JVM: 1.8.0\_65 (Oracle Corporation 25.65-b01)

OS: Windows 10 10.0 amd64

Shell

## 相关教程

* [Gradle 15分钟入门教程](http://www.yiibai.com/gradle/gradle-tutorial-for-beginners.html" \o "Gradle 15分钟入门教程)

本站代码下载：[http://www.yiibai.com/siteinfo/download.html](http://www.yiibai.com/siteinfo/download.html?from=article" \t "_blank)

# Gradle构建脚本

Gradle构建脚本文件用来处理两件事情：一个是项目，另一个是任务。每个Gradle生成表示一个或多个项目。一个项目表示一个JAR库或Web应用程序，也可能表示由其他项目产生的JAR文件组装的ZIP。 简单地说，一个项目是由不同的任务组成。一个任务是指构建执行的一块工作。任务可能是编译一些类，创建一个JAR，产生的Javadoc或发布一些归档文件库。

注：Gradle使用Groovy语言编写脚本。

# 编写构建脚本

Gradle提供了一个域特定语言（DSL），用于描述构建。 它使用 Groovy 语言，使其更容易来形容和构建。Gradle 中的每一个构建脚本使用UTF-8进行编码保存，并命名为 build.gradle 。

### 创建build.gradle文件

看看下面的例子演示一个小脚本，它将打印 “Yiibai gradle qick start“。复制并保存以下脚本到文件：D:workspyiibai.comgradle-3.1studyscript/build.gradle。脚本中定义一个任务名称 helloword，这是用来打印”Yiibai gradle qick start“ 字符串。

在命令提示符下，进入存储 build.gradle 文件的目录并执行以下命令，得到结果如下所示 -

D:workspyiibai.comgradle-3.1studyscript>gradle -q helloword

Yiibai gradle qick start

Shell

可以通过指定一个快捷键（<<符号表示）到 doLast 语句来简化 helloword 任务。 如果将快捷方式添加到上述 helloword 任务，它看起来如下面脚本。

task helloword << {

println 'Yiibai gradle qick start'

}

使用 gradle -q helloword 命令执行上面的脚本，也会得到同样的输出结果。

## Gradle如何使用Groovy

Gradle API默认会到build.gradle中寻找指定的task来执行

Gradle 构建脚本使用 Groovy API。 作为一个入门，我们来看看下面的例子。以下示例演示将字符串转换为大写。  
复制并保存下面的代码到 build.gradle 文件。

task upper << {

String expString = 'YiiBai gradle tutorials'

println "Original: " + expString

println "Upper case: " + expString.toUpperCase()}

Groovy

使用 gradle –q upper 执行上面的代码，如果命令执行成功，会得到下面的输出。

D:workspyiibai.comgradle-3.1studyscript>gradle -q upper

Original: YiiBai gradle tutorials

Upper case: YIIBAI GRADLE TUTORIALS

Shell

下面的例子 4 次打印隐式参数 ($it) 的值。复制并保存下面的代码到 build.gradle 文件。

task count << {

4.times {

print "$it "

}

}

Groovy

使用 gradle –q count 执行上面的代码，如果命令执行成功，会得到下面的输出。

D:workspyiibai.comgradle-3.1studyscript>gradle -q count

0 1 2 3

Shell

## Groovy的JDK方法

Groovy增加了很多有用的方法到标准的Java类。 例如，它从Java API可迭代遍历实现Iterable接口的元素的each() 方法。

复制并保存下面的代码到 build.gradle 文件。

task groovyJDKMethod << {

String myName = "Yiibai";

myName.each() {

println "${it}"

};

}

Groovy

使用 gradle –q groovyJDKMethod 执行上面的代码，如果命令执行成功，会得到下面的输出。

D:workspyiibai.comgradle-3.1studyscript>gradle -q groovyJDKMethod

Y

i

i

b

a

i

Shell

本站代码下载：[http://www.yiibai.com/siteinfo/download.html](http://www.yiibai.com/siteinfo/download.html?from=article" \t "_blank)

# Gradle任务

Gradle构建脚本描述一个或多个项目。每个项目都由不同的任务组成。任务是构建执行的一项工作。任务可以是编译一些类，将类文件存储到单独的目标文件夹中，创建JAR，生成Javadoc或将一些归档发布到存储库。

## 定义任务

task是用于将任务定义到构建脚本中的关键字。看看下面的例子，它是一个叫作 hello 的任务，将打印一个字符串：hello world。将以下脚本复制并保存到 build.gradle 文件中。 此构建脚本定义一个名称为 “hello” 的任务，用于打印hello world字符串。

task hello {

doLast {

println 'hello world'

}

}

Shell

在存储 build.gradle 文件的目录位置执行以下命令，应该看到输出结果如下 -

D:/worksp/yiibai.com/gradle-3.1/study/script>gradle -q hello

hello world

Shell

可以通过为 doLast 语句指定快捷方式（表示符号 <<）来简化此 hello 任务。如果添加这个快捷方式到上面的 hello 任务中，参考如下脚本。

task hello << {

println 'hello world'

}

Shell

在存储 build.gradle 文件的目录位置执行以下命令，应该看到输出结果如下 -

D:/worksp/yiibai.com/gradle-3.1/study/script>gradle -q hello

hello world

Shell

## 任务依赖关系

您可能已经猜到，可以声明依赖于其他任务的任务。下面声明依赖于其他任务的任务，将以下代码复制并保存到build.gradle文件中。

task hello << {

println 'Hello world!'

}

task intro(dependsOn: hello) << {

println "I'm Gradle"

}

Shell

在存储 build.gradle 文件的目录位置执行以下命令，应该看到输出结果如下 -

D:/worksp/yiibai.com/gradle-3.1/study/script>gradle -q intro

Hello world!

I'm Gradle

Shell

~~要添加依赖关系，相应的任务不需要存在。懒依赖 - 其他任务不存在~~，依赖关系与定义task的顺序无关，将以下代码复制并保存到build.gradle文件中。

task taskX(dependsOn: 'taskY') << {

println 'taskX'

}

task taskY << {

println 'taskY'

}

Shell

在存储 build.gradle 文件的目录位置执行以下命令，应该看到输出结果如下 -

D:/worksp/yiibai.com/gradle-3.1/study/script>gradle -q taskX

taskY

taskX

Shell

## 定位任务

如果要查找在构建文件中定义的任务，则必须使用相应的项目标准属性。这意味着每个任务都可以作为项目的属性，使用任务名称作为属性名称。  
看看下面的代码，访问任务作为属性。将以下代码复制并保存到build.gradle文件中。

task hello

println hello.name

println project.hello.name

Shell

在存储 build.gradle 文件的目录位置执行以下命令，应该看到输出结果如下 -

D:/worksp/yiibai.com/gradle-3.1/study/script>gradle -q hello

hello

hello

Shell

您还可以通过任务集合使用所有属性。  
将以下代码复制并保存到build.gradle文件中。

task hello

println tasks.hello.name

println tasks['hello'].name

Shell

在存储 build.gradle 文件的目录位置执行以下命令，应该看到输出结果如下 -

D:/worksp/yiibai.com/gradle-3.1/study/script>gradle -q hello

hello

hello

Shell

## 向任务添加依赖关系

要将一个任务依赖于另一个任务，这意味着当一个任务完成时，另一个任务才会开始。 每个任务都使用任务名称进行区分。 任务名称集合由其任务集合引用。 要引用另一个项目中的任务，应该使用项目路径作为相应任务名称的前缀。  
以下示例将从任务taskX添加依赖项到任务taskY。

task taskX << {

println 'taskX'

}

task taskY(dependsOn: 'taskX') << {

println "taskY"

}

Shell

在存储 build.gradle 文件的目录位置执行以下命令，应该看到输出结果如下 -

D:/worksp/yiibai.com/gradle-3.1/study/script>gradle -q taskY

taskX

taskY

Shell

上面的例子是通过使用名字添加对任务的依赖。 还有另一种方法实现任务依赖性，即使用Task对象定义依赖性。  
现在采用上面任务的相同示例，但是使用任务对象而不是任务参考名称来实现依懒关系。  
将以下代码复制并保存到build.gradle文件中。

task taskY << {

println 'taskY'

}

task taskX << {

println 'taskX'

}

taskY.dependsOn taskX

Shell

在存储 build.gradle 文件的目录位置执行以下命令，应该看到输出结果如下 -

D:/worksp/yiibai.com/gradle-3.1/study/script>gradle -q taskY

taskX

taskY

Shell

还有另一种方法来添加任务依赖，它就是通过使用闭包。在这种情况下，任务通过闭包释放。如果您在构建脚本中使用闭包，那么应该返回任务对象的单个任务或集合。以下示例将任务中从taskX添加依赖项到项目中的所有任务，其名称以“lib”开头。  
将以下代码复制并保存到build.gradle文件中。

task taskX << {

println 'taskX'

}

taskX.dependsOn {

tasks.findAll {

task -> task.name.startsWith('lib')

}

}

task lib1 << {

println 'lib1'

}

task lib2 << {

println 'lib2'

}

task notALib << {

println 'notALib'

}

Shell

在存储 build.gradle 文件的目录位置执行以下命令，应该看到输出结果如下 -

D:/worksp/yiibai.com/gradle-3.1/study/script>gradle -q taskX

lib1

lib2

taskX

Shell

## 向任务添加描述

可以向任务添加描述。 执行Gradle任务时会显示此描述。 这可以通过使用description关键字。  
将以下代码复制并保存到build.gradle文件中。

task copy(type: Copy) {

description 'Copies the resource directory to the target directory.'

from 'resources'

into 'target'

include('\*\*/\*.txt', '\*\*/\*.xml', '\*\*/\*.properties')

println("description applied")

}

Shell

在存储 build.gradle 文件的目录位置执行以下命令，应该看到输出结果如下 -

D:/worksp/yiibai.com/gradle-3.1/study/script>gradle -q taskX

description applied

Shell

## 跳过任务

如果用于跳过任务的逻辑不能用谓词表示，则可以使用StopExecutionException。 如果操作抛出此异常，则会跳过此操作执行以及此任务的任何后续操作的执行。 构建继续执行下一个任务。  
将以下代码复制并保存到build.gradle文件中。

task compile << {

println 'We are doing the compile.'

}

compile.doFirst {

// Here you would put arbitrary conditions in real life.

// But this is used in an integration test so we want defined behavior.

if (true) { throw new StopExecutionException() }

}

task myTask(dependsOn: 'compile') << {

println 'I am not affected'

}

SQL

在存储 build.gradle 文件的目录位置执行以下命令，应该看到输出结果如下 -

D:/worksp/yiibai.com/gradle-3.1/study/script>gradle -q myTask

I am not affected

Shell

Gradle在处理任务时有不同的阶段。 首先，有一个配置阶段，其中直接在任务的闭包中指定的代码被执行。 针对每个可用任务执行配置块，而不仅针对稍后实际执行的那些任务。

## 参考 -

* <https://docs.gradle.org/current/userguide/more_about_tasks.html>
* <https://docs.gradle.org/current/userguide/tutorial_using_tasks.html>

本站代码下载：[http://www.yiibai.com/siteinfo/download.html](http://www.yiibai.com/siteinfo/download.html?from=article" \t "_blank)

# Gradle依赖管理

Gradle构建脚本定义了构建项目的过程; 每个项目包含一些依赖项和一些发表项。依赖项意味着支持构建项目的东西，例如来自其他项目的所需JAR文件以及类路径中的外部JAR（如JDBC JAR或Eh-cache JAR）。发布表示项目的结果，如测试类文件和构建文件，如war文件。

Gradle负责构建和发布结果。 发布基于定义的任务。 可能希望将文件复制到本地目录，或将其上传到远程[Maven](http://www.yiibai.com/maven/)或lvy存储库，或者可以在同一个多项目构建中使用另一个项目的文件。 发布的过程称为发布。

## 声明依赖关系

Gradle遵循一些特殊语法来定义依赖关系。 以下脚本定义了两个依赖项，一个是Hibernate core 3.6.7，第二个是Junit 4.0和更高版本。如下面的代码所示，可在build.gradle文件中使用此代码。

apply plugin: 'java'

repositories {

mavenCentral()

}

dependencies {

compile group: 'org.hibernate', name: 'hibernate-core', version: '3.6.7.Final'

testCompile group: 'junit', name: 'junit', version: '4.+'

}

SQL

## 依赖关系配置

依赖关系配置只是定义了一组依赖关系。 您可以使用此功能声明从Web下载外部依赖关系。这定义了以下不同的标准配置。

* 编译 − 编译项目的生产源所需的依赖关系。
* 运行时 - 运行时生产类所需的依赖关系。 默认情况下，还包括编译时依赖项。
* 测试编译 - 编译项目测试源所需的依赖项。 默认情况下，它包括编译的产生的类和编译时的依赖。
* 测试运行时 - 运行测试所需的依赖关系。 默认情况下，它包括运行时和测试编译依赖项。

## 外部依赖

外部依赖是一种依赖。 这是对当前构建之外的一些文件的依赖，并且存储在某种类型的存储库（可以使本地建的仓库）中，例如：[Maven](http://www.yiibai.com/maven/" \o "Maven) central，corporate Maven或lvy repository或本地文件系统中的目录。  
以下代码片段是定义外部依赖关系。 在build.gradle文件中使用如下代码。

dependencies {

compile group: 'org.hibernate', name: 'hibernate-core', version: '3.6.7.Final'

}

Shell

## 存储库

在添加外部依赖关系时， Gradle在存储库中查找它们。 存储库只是文件的集合，按分组，名称和版本来组织构造。 默认情况下，Gradle不定义任何存储库。 我们必须至少明确地定义一个存储库。 下面的代码片段定义了 maven 仓库。 在build.gradle文件中使用此代码。

repositories {

mavenCentral()

}

Shell

下面的代码是定义远程maven。 在build.gradle文件中可使用下面代码。

repositories {

maven {

url "http://repo.mycompany.com/maven2"

}

}

Shell

## 发布文件

依赖关系配置也用于发布文件。 这些已发布的文件称为工件。 通常，我们使用插件来定义工件。 但是需要告诉Gradle在哪里发布文件。可以通过将存储库附加到上传存档任务来实现此目的。 请查看以下用于发布Maven存储库的语法。 执行时，Gradle将根据项目需求生成并上传Pom.xml。 在build.gradle文件中使用此代码。

apply plugin: 'maven'

uploadArchives {

repositories {

mavenDeployer {

repository(url: "file://localhost/tmp/myRepo/")

}

}

}

SQL

本站代码下载：[http://www.yiibai.com/siteinfo/download.html](http://www.yiibai.com/siteinfo/download.html?from=article" \t "_blank)

# Gradle插件

插件只是一组任务，几乎所有的任务，如编译任务，设置域对象，设置源文件等都由插件处理。

## 插件类型

Gradle中有两种类型的插件：脚本插件和二进制插件。 脚本插件是一个额外的构建脚本，它提供了一种声明性方法来操作构建，通常在构建中使用。 二进制插件是实现插件接口并采用编程方法来操作构建的类。二进制插件可以驻留在插件JAR中的一个构建脚本和项目层次结构或外部。

## 应用插件

Project.apply（）API方法用于应用特定的插件。您可以多次使用相同的插件。有两种类型的插件：一种是脚本插件，第二种是二进制插件。

### 脚本插件

脚本插件可以从本地文件系统上的脚本或远程位置应用。文件系统位置相对于项目目录，而远程脚本位置指定HTTP URL。看看下面的代码片段。它将other.gradle插件用于构建脚本。 在build.gradle文件中使用此代码如下。

apply from: 'other.gradle'

Shell

### 二进制插件

每个插件由插件标识。一些核心插件是使用短名称来应用它，一些社区插件是使用插件ID的完全限定名称。有时它允许指定一个插件类。  
看看下面的代码片段。 它显示如何使用应用Java插件。 在build.gradle文件中使用此代码。

apply plugin: JavaPlugin

Shell

看看下面的代码，使用短名称应用核心插件。 在build.gradle文件中使用此代码如下所示。

plugins {

id 'java'

}

Shell

看看下面的代码，使用短名称应用社区插件。 在build.gradle文件中使用此代码如下所示。

plugins {

id "com.yiibai.bintray" version "0.1.0"

}

Shell

## 编写自定义插件

在创建自定义插件时，您需要编写一个插件的实现。 Gradle实例化插件并使用Plugin.apply（）方法调用插件实例。以下示例包含一个简单的hello插件，它将一个问候任务添加到项目中。看看下面的代码。 在build.gradle文件中使用此代码。

apply plugin: HelloPlugin

class HelloPlugin implements Plugin<Project> {

void apply(Project project) {

project.task('hello') << {

println "Hello from the HelloPlugin."

}

}

}

Shell

使用以下代码执行上述脚本。

D:/worksp/yiibai.com/gradle-3.1/study/script> gradle -q hello

Hello from the HelloPlugin

Shell

## 从构建获取输入

大多数插件需要从构建脚本中的配置获得支持。 Gradle项目有一个关联“ExtensionContainer”对象，它有助于跟踪传递给插件的所有设置和属性。

让我们在项目中添加一个简单的扩展对象。在这里，我们向项目中添加一个问候语扩展对象，它允许我们配置问候语。在build.gradle文件中使用此代码如下所示。

apply plugin: HelloPlugin

greeting.message = 'Hi,this from configure Gradle message.'

class HelloPlugin implements Plugin<Project> {

void apply(Project project) {

// Add the 'greeting' extension object

project.extensions.create("greeting", HelloPluginExtension)

// Add a task that uses the configuration

project.task('hello') << {

println project.greeting.message

}

}

}

class HelloPluginExtension {

def String message = 'Hello,this from HelloPluginExtension message.'

}

Shell

使用以下代码执行上述脚本，得到结果如下 -

D:/worksp/yiibai.com/gradle-3.1/study/script>gradle -q hello

Hi,this from configure Gradle message.

Shell

Gradle为每个扩展对象添加了一个配置闭包，因此可以将分组设置在一起。 看看下面的代码。 在build.gradle文件中使用此代码。

apply plugin: GreetingPlugin

greeting {

message = 'Hi'

greeter = 'Gradle'

}

class GreetingPlugin implements Plugin<Project> {

void apply(Project project) {

project.extensions.create("greeting", GreetingPluginExtension)

project.task('hello') << {

println "${project.greeting.message} from ${project.greeting.greeter}"

}

}

}

class GreetingPluginExtension {

String message

String greeter

}

SQL

使用以下代码执行上述脚本，得到结果如下 -

D:/worksp/yiibai.com/gradle-3.1/study/script>gradle -q hello

Hi from Gradle

Shell

## 标准Gradle插件

在Gradle分布中包含不同的插件。

### 语言插件

这些插件的添加，让JVM在编译和执行时对各种语言支持。

| **插件Id** | **自动应用** | **描述** |
| --- | --- | --- |
| java | java-base | 向项目添加Java编译，测试和绑定的功能。它作为许多其他Gradle插件的基础。 |
| groovy | java,groovy-base | 添加对构建Groovy项目的支持。 |
| scala | java,scala-base | 添加对构建Scala项目的支持。 |
| antlr | Java | 添加了使用Antlr生成解析器的支持。 |

### 孵化语言插件

这些插件添加对各种语言的支持。

| **插件Id** | **自动应用** | **描述** |
| --- | --- | --- |
| 汇编 | — | 向项目添加本地汇编语言功能。 |
| c | — | 向项目添加C语言的源代码编译功能。 |
| cpp | — | 向项目添加C++语言的源代码编译功能。 |
| objective-c | — | 向项目添加objective-c语言的源代码编译功能。 |
| objective-cpp | — | 向项目添加Objective-C++语言的源代码编译功能。 |
| windows-resources | — | 添加本机二进制文件包括Windows资源的支持。 |

本站代码下载：[http://www.yiibai.com/siteinfo/download.html](http://www.yiibai.com/siteinfo/download.html?from=article" \t "_blank)

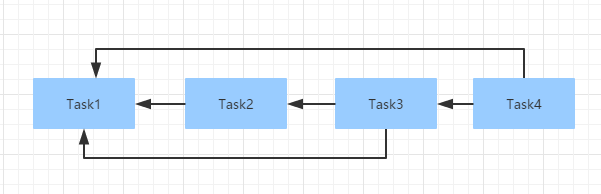
# Gradle运行构建

Gradle提供了一个命令行来执行构建脚本。 它可以一次执行多个任务。在这里将介绍如何使用不同的选项来执行多个任务。

## 执行多个任务

Gradle可以从单个构建文件-build.gradle执行多个任务。使用gradle命令处理构建文件。此命令将按列出的顺序编译每个任务，并使用不同的选项执行每个任务以及依赖关系。

**示例** - 假设有四个任务 - task1，task2，task3和task4。task3和task4取决于task1和task2。 看看下面的图表。



在上面的四个任务是相互依赖的，用一个箭头符号表示。 看看下面的代码。 将其复制并粘贴到build.gradle文件中。

task task1 << {

println 'compiling source #1'

}

task task2(dependsOn: task1) << {

println 'compiling unit tests #2'

}

task task3(dependsOn: [task1, task2]) << {

println 'running unit tests #3'

}

task task4(dependsOn: [task1, task3]) << {

println 'building the distribution #4'

}

Groovy

使用以下代码来编译和执行上述任务。如果命令执行成功，将获得以下输出 -

D:/worksp/yiibai.com/gradle-3.1/study/script>gradle task4 task1

:task1

compiling source #1

:task2

compiling unit tests #2

:task3

running unit tests #3

:task4

building the distribution #4

BUILD SUCCESSFUL

Total time: 1.265 secs

Groovy

排除任务

要执行中排除某个任务时，可以在gradle命令中使用-x选项，并指出要排除的任务的名称。  
使用以下命令用于从上面的脚本中排除 task1 这个任务。  
使用以下代码来编译和执行上述任务。如果命令执行成功，将获得以下输出 -

D:/worksp/yiibai.com/gradle-3.1/study/script>gradle task4 -x task1

:task2

compiling unit tests #2

:task3

running unit tests #3

:task4

building the distribution #4

BUILD SUCCESSFUL

Total time: 1.272 secs

Groovy

## 发生故障时继续构建

Gradle将在任何任务失败时立即中止执行。但是有时我们希望即使发生故障，也可以继续执行。 为此，要在gradle命令使用-continue选项。它分别处理每个任务及其依赖关系。 重要的是，它将捕获每个遇到的故障，并在构建的执行结束时生成报告。 假设一个任务失败，那么相关的后续依懒任务也不会被执行。

## 选择执行哪些构建

当运行gradle命令时，它在当前目录中查找构建文件。我们也可以使用-b选项选择指定的构建文件的路径,不指定时默认选择build.gradle作为构建文件。以下示例显示在subdir/ 子目录中创建一个新文件 newbuild.gradle，并创建一个名称为 hello 项目。创建的newbuild.gradle文件的代码内容如下 -

task hello << {

println "Use File：$buildFile.name in '$buildFile.parentFile.name'."

}

Groovy

使用以下代码来编译和执行上述任务。如果命令执行成功，将获得以下输出 -

D:/worksp/yiibai.com/gradle-3.1/study/script> gradle -q -b subdir/newbuild.gradle hello

Use File：newbuild.gradle in 'subdir'.

Groovy

## 获取构建信息

Gradle提供了几个内置任务来检索有关任务和项目的详细信息。这对理解构建的结构和依赖性以及调试一些问题很有用。可使用项目报告插件向项目中添加任务，来生成这些报告。

### 列出项目

可以使用gradle -q projects命令来列出所选项目及其子项目的项目层次结构。下面是一个列出构建文件中的所有项目的示例 -

D:/worksp/yiibai.com/gradle-3.1/study/script>gradle -q projects

------------------------------------------------------------

Root project

------------------------------------------------------------

Root project 'script'

No sub-projects

To see a list of the tasks of a project, run gradle <project-path>:tasks

For example, try running gradle :tasks

Groovy

报告显示每个项目的描述（如果有指定的话）。可以使用以下命令指定描述。将其粘贴到build.gradle文件中。  
再一次执行命令，得到以下结果 -

D:/worksp/yiibai.com/gradle-3.1/study/script>gradle -q projects

------------------------------------------------------------

Root project - The shared API for the application

------------------------------------------------------------

Root project 'script' - The shared API for the application

No sub-projects

To see a list of the tasks of a project, run gradle <project-path>:tasks

For example, try running gradle :tasks

Groovy

### 列出任务

使用以下命令列出属于多个项目的所有任务。如下所示 -

D:/worksp/yiibai.com/gradle-3.1/study/script>gradle -q tasks --all

------------------------------------------------------------

All tasks runnable from root project - The shared API for the application

------------------------------------------------------------

Build Setup tasks

-----------------

init - Initializes a new Gradle build. [incubating]

wrapper - Generates Gradle wrapper files. [incubating]

Help tasks

----------

buildEnvironment - Displays all buildscript dependencies declared in root project 'script'.

components - Displays the components produced by root project 'script'. [incubating]

dependencies - Displays all dependencies declared in root project 'script'.

dependencyInsight - Displays the insight into a specific dependency in root project 'script'.

help - Displays a help message.

model - Displays the configuration model of root project 'script'. [incubating]

projects - Displays the sub-projects of root project 'script'.

properties - Displays the properties of root project 'script'.

tasks - Displays the tasks runnable from root project 'script'.

Other tasks

-----------

task4

task1

task2

task3

Groovy

以下是其它一些命令及其说明的列表。

| **编号** | **命令** | **描述** |
| --- | --- | --- |
| 1 | gradle –q help –task | 提供有关指定任务或多个任务的使用信息（如路径，类型，描述，组）。 |
| 2 | gradle –q dependencies | 提供所选项目的依赖关系的列表。 |
| 3 | gradle -q api:dependencies —configuration | 提供有关配置的有限依赖项的列表。 |
| 4 | gradle –q buildEnvironment | 提供构建脚本依赖项的列表 |
| 5 | gradle –q dependencyInsight | 提供了一个洞察到一个特定的依赖 |
| 6 | gradle –q properties | 提供所选项目的属性列表 |

本站代码下载：[http://www.yiibai.com/siteinfo/download.html](http://www.yiibai.com/siteinfo/download.html?from=article" \t "_blank)

# Gradle构建JAVA项目

本章介绍如何使用Gradle构建文件来构建一个Java项目。 首先，我们必须向构建脚本中添加Java插件，因为它提供了编译Java源代码，运行单元测试，创建Javadoc和创建JAR文件的任务。 在build.gradle文件中使用以下代码行。

apply plugin: 'java' //这就代表添加了java插件

Groovy

怎么样，是不是很简单？

## Java默认的项目布局

每当添加一个插件到构建中，它会假设一个特定的Java项目设置（类似于Maven）。看看下面的目录结构。

* src/main/java 目录包含Java源代码；
* src/test/java 目录包含测试用的源代码；

如果遵循上面设置，以下构建文件足以编译，测试并捆绑Java项目。  
要启动构建，请在命令行上键入以下命令。

D:/worksp/gradle> gradle build

Groovy

SourceSets可用于指定不同的项目结构。例如，指定源代码存储在src文件夹中，而不是在src/main/java中。 看看下面的目录结构。

apply plugin: 'java'

sourceSets {

main {

java {

srcDir 'src'

}

}

test {

java {

srcDir 'test'

}

}

}

Groovy

## 初始化任务执行

Gradle还不支持多个项目模板。但它提供了一个init初始化任务来创建一个新的Gradle项目的结构。如果没有指定其他参数，任务将创建一个Gradle项目，其中包含gradle包装器文件，build.gradle和settings.gradle文件。

当使用java-library作为值并添加--type参数时，将创建一个java项目结构，build.gradle文件包含带有[Junit](http://www.yiibai.com/junit/" \o "Junit)的某个Java模板。 看看下面build.gradle文件的代码。

apply plugin: 'java'

repositories {

jcenter()

}

dependencies {

compile 'org.slf4j:slf4j-api:1.7.12'

testCompile 'junit:junit:4.12'

}

Groovy

在仓库(repositories)这部分中，它定义了要从哪里找到依赖。Jcenter是为了解决依赖问题。 依赖关系（dependencies）部分用于提供有关外部依赖关系的信息。

## 指定Java版本

通常，Java项目要有一个版本和一个目标JRE，在其上编译它。version和sourceCompatibility属性可以在build.gradle文件中设置。

version = 0.1.1

sourceCompatibility = 1.8

Groovy

如果这是一个可执行的Java应用程序，MANIFEST.MF文件必须要指定具有main方法的类。

apply plugin: 'java'

jar {

manifest {

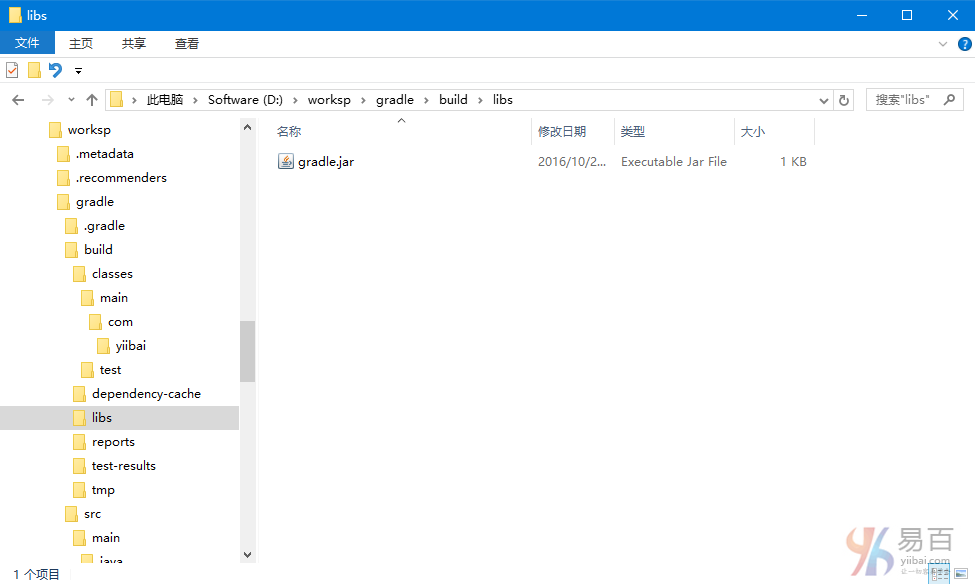
attributes 'Main-Class': 'com.yiibai.main.Application'

}

}

Groovy

让我们考虑一个例子。创建一个目录结构，如下面的屏幕截图所示。



将以下Java代码复制到App.java文件中，并将其存储到 D:/worksp/gradle/src/main/java/com/yiibai 目录中。

package com.yiibai;

/\*\*

\* Hello world!

\*

\*/

public class App {

public static void main( String[] args ){

System.out.println( "Hello World, Gradle Build Work! " );

}

}

Java

将以下Java代码复制到AppTset.java文件中，并存储到D:/worksp/gradle/src/test/java/com/yiibai目录中。

package com.yiibai;

/\*\*

\* Hello world!

\*

\*/

public class App{

public static void main( String[] args ){

System.out.println( "Hello World!" );

}

}

Java

将以下代码复制到build.gradle文件中，并放入 D:/worksp/gradle 目录。

apply plugin: 'java'

repositories {

jcenter()

}

dependencies {

compile 'org.slf4j:slf4j-api:1.7.12'

testCompile 'junit:junit:4.12'

}

jar {

manifest {

attributes 'Main-Class': 'com.yiibai.main.App'

}

}

SQL

要编译和执行上述脚本，请使用以下命令。

D:/worksp/gradle> gradle tasks

....

D:/worksp/gradle> gradle assemble

....

D:/worksp/gradle> gradle build

:compileJava UP-TO-DATE

:processResources UP-TO-DATE

:classes UP-TO-DATE

:jar UP-TO-DATE

:assemble UP-TO-DATE

:compileTestJava

:processTestResources UP-TO-DATE

:testClasses

:test

:check

:build

BUILD SUCCESSFUL

Total time: 1.638 secs

Groovy

查看相应目录中的所有类文件，并检查 D:/worksp/gradle/build/libs文件夹中，应该会有一个 gradle.jar文件。

本站代码下载：[http://www.yiibai.com/siteinfo/download.html](http://www.yiibai.com/siteinfo/download.html?from=article" \t "_blank)