# Gradle入门系列（2）：第一个Java项目

2015-07-05 JustinWu 安卓应用频道



这篇教程的主要内容是讲解如何用Gradle编译和打包一个简单的Java项目。

该Java项目只有一个需求：我们的构建脚本必须创建一个可执行的Jar文件，换句话说，我们必须能够使用命令java -jar jarfile.jar 来运行我们的程序。我们来看一下如何满足这个需求。

## 创建一个Java项目

我们可以使用Java插件（译注：关于Gradle插件的定义，请查看第一篇教程）来创建一个Java项目，为了做到这点，我们需要把下面这段语句加入到build.gradle文件中：

```
apply plugin: 'java'
```

就是这样，现在我们已经创建了一个Java项目。Java插件会在我们的构建中添加一些新的约定（如默认的项目结构），新的任务，和新的属性。
让我们来快速地看一下默认的项目结构。

## Java项目结构

默认的项目结构如下：

- src/main/java目录包含了项目的源代码。
- src/main/resources目录包含了项目的资源（如属性文件）。
- src/test/java目录包含了测试类。
- src/test/resources目录包含了测试资源。所有我们构建生成的文件都会在build目录下被创建，这个目录涵盖了以下的子目录，这些子目录我们会在这篇教程中提到，另外还有一些子目录我们会放在以后讲解。
- classes目录包含编译过的.class文件。
- libs目录包含构建生成的jar或war文件。

## 为构建加入一个主类（main class）

让我们创建一个简单的主类，在这个类中会打印一个“Hello world”然后System.out出来。这个HelloWorld类的源代码如下：

```
package net.petrikainulainen.gradle;

public class HelloWorld {

  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```

HelloWorld类存放在src/main/java/net/petrikainulainen/gradle目录

这很好，然而，我们还需要编译和打包我们的项目，不是吗？我们先看一下这个Java工程中的任务。

## Java工程中的任务

Java插件在我们的构建中加入了很多任务，我们这篇教程涉及到的任务如下：

- assemble任务会编译程序中的源代码，并打包生成Jar文件，这个任务不执行单元测试。
- build任务会执行一个完整的项目构建。
- clean任务会删除构建目录。
- compileJava任务会编译程序中的源代码。

我们还可以执行以下命令得到一个可运行任务及其描述的完整列表

```
gradle tasks
```

这是一个很好的方式，不需要阅读构建脚本，就能对你的项目进行大致的浏览，如果我们在项目根目录下运行这个命令，我们可以看到以下输出：

```
> gradle tasks
:tasks

------------------------------------------------------------
All tasks runnable from root project
```

```
------------------------------------------------------------

Build tasks
-----------
assemble - Assembles the outputs of this project.
build - Assembles and tests this project.
buildDependents - Assembles and tests this project and all projects that depend on it.
buildNeeded - Assembles and tests this project and all projects it depends on.
classes - Assembles classes 'main'.
clean - Deletes the build directory.
jar - Assembles a jar archive containing the main classes.
testClasses - Assembles classes 'test'.

Build Setup tasks
-----------------
init - Initializes a new Gradle build. [incubating]
wrapper - Generates Gradle wrapper files. [incubating]

Documentation tasks
-------------------
javadoc - Generates Javadoc API documentation for the main source code.

Help tasks
----------
dependencies - Displays all dependencies declared in root project 'first-java-project'.
dependencyInsight - Displays the insight into a specific dependency in root project 'first-
java-project'.
help - Displays a help message
projects - Displays the sub-projects of root project 'first-java-project'.
properties - Displays the properties of root project 'first-java-project'.
tasks - Displays the tasks runnable from root project 'first-java-project'.

Verification tasks
------------------
check - Runs all checks.
test - Runs the unit tests.

Rules
-----
```

> Pattern: build<ConfigurationName>: Assembles the artifacts of a configuration.
> Pattern: upload<ConfigurationName>: Assembles and uploads the artifacts belonging to a configuration.
> Pattern: clean<TaskName>: Cleans the output files of a task.
>
> To see all tasks and more detail, run with --all.
>
> BUILD SUCCESSFUL
>
> Total time: 2.792 secs

我们继续，下面要讲怎样打包我们的项目。

## 打包我们的项目

我们可以通过使用两个不同的任务来打包项目。
如果我们在命令提示符中执行命令gradle assemble，我们可以看到以下输出：

```
> gradle assemble
:compileJava
:processResources
:classes
:jar
:assemble

BUILD SUCCESSFUL

Total time: 3.163 secs
```

如果我们在命令提示符中执行命令gradle build，我们可以看到以下输出：

```
> gradle build
:compileJava
:processResources
:classes
:jar
:assemble
:compileTestJava
:processTestResources
```

```
:testClasses
:test
:check
:build

BUILD SUCCESSFUL

Total time: 3.01 secs
```

这些命令的输出表明了它们的区别：

- assemble任务仅仅执行项目打包所必须的任务集。
- build任务执行项目打包所必须的任务集，以及执行自动化测试。这两个命令都会在build/libs目录中创建一个file-java-project.jar文件。默认创建的Jar文件名称是由这个模版决定的：[projectname].jar，此外，项目的默认名称和其所处的目录名称是一致的。因此如果你的项目目录名称是first-java-project，那么创建的Jar文件名称就是first-java-project.jar。

现在，我们尝试使用以下命令运行我们的程序：

```
java -jar first-java-project.jar
```

我们可以看到以下输出：

```
> java -jar first-java.project.jar
No main manifest attribute, in first-java-project.jar
```

问题出在，我们没有在manifest文件中配置Jar文件的主类，让我们继续看看怎样解决这个问题。

## 配置Jar文件的主类

Java插件在我们的项目中加入了一个Jar任务，每一个Jar对象都一个manifest属性，这个属性是Manifest的一个实例。

我们可以对生成的Jar文件的主类进行配置，使用Manifest接口的attributes()方法。换句话说，我们可以使用一个包含键值对的map结构指定加入到manifest文件的属性集。

我们能够通过设置Main-Class属性的值，指定我们程序的入口点。在我们对build.gradle文件

进行必要的改动后，代码如下：

```
apply plugin: 'java'

jar {
  manifest {
      attributes 'Main-Class': 'net.petrikainulainen.gradle.HelloWorld'
  }
}
```

（JavaSE教程提供了关于manifest文件的更多信息。）

在我们执行gradle assemble或gradle build命令生成一个新的jar文件之后，我们可以执行以下命令运行jar文件：

```
java -jar first-java-project.jar
```

当我们运行程序时，System.out会打印出以下信息：

```
> java -jar first-java-project.jar
Hello World!
```

这就是我们今天所有的内容，我们看一下我们学到了什么。

## 总结

我们已经通过Gradle创建了一个简单的Java项目，这篇教程教会了我们四点：

- 我们了解了可以使用Gradle的Java插件创建一个Java项目。
- 我们知道了Java项目的默认结构和Maven项目的默认结构是一样的。
- 我们知道了构建所生成的所有文件都能在build目录下找到。
- 我们知道了我们可以自定义加入到manifest文件中的属性。

P.S. 这篇教程的示例代码可以在Github找到。

来源：伯乐在线 - Justin Wu
链接：http://android.jobbole.com/72558/

---

「安卓应用频道」专注 Android 技术、设计和市场推广相关的内容分享。如果你期望了解Android应用的全流程知识，欢迎关注我们。

微信号： **AndroidPD**



（长按上图↑可自动识别二维码）

---



阅读原文

微信扫一扫
关注该公众号