和Maven一样，Gradle只是提供了构建项目的一个框架，真正起作用的是Plugin。Gradle在默认情况下为我们提供了许多常用的Plugin，其中包括有构建Java项目的Plugin，还有War，Ear等。与Maven不同的是，Gradle不提供内建的项目生命周期管理，只是java Plugin向Project中添加了许多Task，这些Task依次执行，为我们营造了一种如同Maven般项目构建周期。

现在我们都在谈领域驱动设计，Gradle本身的领域对象主要有Project和Task。Project为Task提供了执行上下文，所有的Plugin要么向Project中添加用于配置的Property，要么向Project中添加不同的Task。一个Task表示一个逻辑上较为独立的执行过程，比如编译Java源代码，拷贝文件，打包Jar文件，甚至可以是执行一个系统命令或者调用Ant。另外，一个Task可以读取和设置Project的Property以完成特定的操作。

让我们来看一个最简单的Task，创建一个build.gradle文件，内容如下：

```
task helloWorld << {
   println "Hello World!"
}
```

这里的“<<”表示向helloWorld中加入执行代码——其实就是groovy代码。Gradle向我们提供了一整套DSL，所以在很多时候我们写的代码似乎已经脱离了groovy，但是在底层依然是执行的groovy。比如上面的task关键字，其实就是一个groovy中的方法，而大括号之间的内容则表示传递给task()方法的一个闭包。除了“<<”之外，我们还很多种方式可以定义一个Task，我们将在本系列后续的文章中讲到。

在与build.gradle相同的目录下执行：

```
gradle helloWorld
```

命令行输出如下：

```
:helloWorld
Hello World!

BUILD SUCCESSFUL

Total time: 2.544 secs
```

在默认情况下，Gradle将当前目录下的build.gradle文件作为项目的构建文件。在上面的例子中，我们创建了一个名为helloWorld的Task，在执行gradle命令时，我们指定执行这个helloWorld Task。这里的helloWorld是一个DefaultTask类型的对象，这也是定义一个Task时的默认类型，当然我们也可以显式地声明Task的类型，甚至可以自定义一个Task类型（我们将在本系列的后续文章中讲到）。

比如，我们可以定义一个用于文件拷贝的Task：

```
task copyFile(type: Copy) {
   from 'xml'
   into 'destination'
}
```

以上copyFile将xml文件夹中的所有内容拷贝到destination文件夹中。这里的两个文件夹都是相对于当前Project而言的，即build.gradle文件所在的目录。

Task之间可以存在依赖关系，比如taskA依赖于taskB，那么在执行taskA时，Gradle会先执行taskB，然后再执行taskA。声明Task依赖关系的一种方式是在定义一个Task的时候：

```
task taskA(dependsOn: taskB) {
   //do something
}
```

Gradle在默认情况下为我们提供了几个常用的Task，比如查看Project的Properties、显示当前Project中定义的所有Task等。可以通过一下命令查看Project中所有的Task：

```
gradle tasks
```

输出如下：

```
:tasks

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

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

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

Other tasks
-----------
copyFile
helloWorld

To see all tasks and more detail, run with --all.

BUILD SUCCESSFUL

Total time: 2.845 secs
```

可以看到，除了我们自己定义的copyFile和helloWorld之外，Gradle还默认为我们提供了dependencies、projects和properties等Task。dependencies用于显示Project的依赖信息，projects用于显示所有Project，包括根Project和子Project，而properties则用于显示一个Project所包含的所有Property。

在默认情况下，Gradle已经为Project添加了很多Property，我们可以调用以下命令进行查看：

```
gradle properties
```

输出如下：

```
```

```
:properties

------------------------------------------------------------
Root project
------------------------------------------------------------

allprojects: [root project 'gradle-blog']
ant: org.gradle.api.internal.project.DefaultAntBuilder@1342097

buildDir: /home/davenkin/Desktop/gradle-blog/build
buildFile: /home/davenkin/Desktop/gradle-blog/build.gradle
...
configurations: []
convention: org.gradle.api.internal.plugins.DefaultConvention@11492ed
copyFile: task ':copyFile'
...
ext: org.gradle.api.internal.plugins.DefaultExtraPropertiesExtension@1b5d53a
extensions: org.gradle.api.internal.plugins.DefaultConvention@11492ed
...
helloWorld: task ':helloWorld'
...
plugins: [org.gradle.api.plugins.HelpTasksPlugin@7359f7]
project: root project 'gradle-blog'
...
properties: {...}
repositories: []

tasks: [task ':copyFile', task ':helloWorld']
version: unspecified

BUILD SUCCESSFUL

Total time: 2.667 secs
```

在以上Property中，allprojects表示所有的Project，这里只包含一个根Project，在多项目构建中，它将包含多个Project；buildDir表示构建结果的输出目录；我们自己定义的helloWorld和copyFile也成为了Project中的Property。另外，Project还包括用于执行Ant命令的DefaultAntBuilder（Property名为ant）和Project的描述属性description。