Gradle Handson Cheatsheet

1. Create a new project, and run `gradle init`. It will create a gradle project layout for you. It will generate a few files like the following and it is equivalent to you create a gradle project in eclipse.

*build.gradle gradle gradlew gradlew.bat settings.gradle*

then the build.gradle will look like this,

apply plugin: 'java'

repositories {

jcenter()

}

dependencies {

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

testCompile 'junit:junit:4.12'

}

1. Run build

Assuming the default project layout of the project is like layouted like the following,

*src/main/java contains the Java source code*

*src/test/java contains the Java tests*

then run,

`gradle build`

1. Specify jvm heap size for gradle in gradlew

# Add default JVM options here. You can also use JAVA\_OPTS and GRADLE\_OPTS to pass JVM options to this script.

1. DEFAULT\_JVM\_OPTS=""
2. What stages

$ gradle build

:compileJava NO-SOURCE

:processResources NO-SOURCE

:classes UP-TO-DATE

:jar

:assemble

:compileTestJava NO-SOURCE

:processTestResources NO-SOURCE

:testClasses UP-TO-DATE

:test NO-SOURCE

:check UP-TO-DATE

:build

$ gradle assemble

:compileJava NO-SOURCE

:processResources NO-SOURCE

:classes UP-TO-DATE

:jar UP-TO-DATE

:assemble UP-TO-DATE

BUILD SUCCESSFUL in 1s

1 actionable task: 1 up-to-date

**Add a task, and run a task**

xli@T550-13870 MINGW64 /d/tmp

$ gradle hello

The Task.leftShift(Closure) method has been deprecated and is scheduled to be removed in Gradle 5.0. Please use Task.doLast(Action) instead.

at build\_2h6qdg4wp81us39exn7yrj3d9.run(D:\tmp\build.gradle:32)

:hello

tutorial point

BUILD SUCCESSFUL in 1s

1 actionable task: 1 executed

-q, silent the warning.

xli@T550-13870 MINGW64 /d/tmp

$ gradle -q hello

tutorial point

change the above task to,

1/ task hello

Println hello.name

Println project.hello.name

2/ task hello {

Println hello.name

Println project.hello.name

}

3 task hello() {

Println hello.name

Println project.hello.name

}

4/ task hello() {

Println hello.name

Println project[‘hello’].name

}

1. Dependency

task taskX << { println 'taskX' }

task taskY(dependsOn: 'taskX') << { println "taskY" }

**Gradle library dependency download**

Gradle downloaded the dependent jars and put it in $HOME/.gradle/caches/modules-2.

To remove it manually, just remove $HOME/.gradle/caches.

A sample build.gradle file is provided as the following,

// Apply the java-library plugin to add support for Java Library

apply plugin: 'java-library'

// In this section you declare where to find the dependencies of your project

repositories {

// Use jcenter for resolving your dependencies.

// You can declare any Maven/Ivy/file repository here.

mavenCentral()

}

dependencies {

// This dependency is exported to consumers, that is to say found on their compile classpath.

api 'org.apache.commons:commons-math3:3.6.1'

// This dependency is used internally, and not exposed to consumers on their own compile classpath.

implementation 'com.google.guava:guava:21.0'

compile("org.springframework.boot:spring-boot-starter-web:1.5.7.RELEASE")

compile('org.springframework:spring-beans:4.2.3.RELEASE')

// Use JUnit test framework

testImplementation 'junit:junit:4.12'

task getDependencies(type: Copy) {

from configurations.compile into 'lib/'

}

}

To get dependencies jar download to a directory, you need to add the above task, and run the following command,

*‘Gradle getDependencies’*

**Tasks and commands**

`gradle tasks ` will list all the system commands.

`gradle clean, Javadoc, check, test` are very easy to understand. Make sure understand the sequence of these commands.

`gradle classes`, will compile and output classes files.

`gradle jar`, will zip and output jar files from the class files.

`gradle assemble`, which is the step right after jar, you can think of it is to put things together.

`gradle testClasses` is to compile and output test java files.

`gradle test`, is the step after `testClasses`, which will run the generated test file class files and generate output in the console.

`gradle build` is the final step, which means assemble + test.

**A sample command list.**

PS C:\Users\xli\eclipse-workspace\SpringBoot> gradle tasks

> Task :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 main classes.

clean - Deletes the build directory.

jar - Assembles a jar archive containing the main classes.

testClasses - Assembles test classes.

Build Setup tasks

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

init - Initializes a new Gradle build.

wrapper - Generates Gradle wrapper files.

Documentation tasks

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

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

Help tasks

----------

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

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

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

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

dependentComponents - Displays the dependent components of components in root project 'SpringBoot'. [incubating]

help - Displays a help message.

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

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

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

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

Verification tasks

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

check - Runs all checks.

test - Runs the unit tests.

Rules

-----

Pattern: clean<TaskName>: Cleans the output files of a task.

Pattern: build<ConfigurationName>: Assembles the artifacts of a configuration.

Pattern: upload<ConfigurationName>: Assembles and uploads the artifacts belonging to a configuration.

To see all tasks and more detail, run gradle tasks --all

To see more detail about a task, run gradle help --task <task>

BUILD SUCCESSFUL in 0s

==========================End of the File =========================================