DepenDancy Managment usig Gradel

1. **Add a gradle dependency and its related repository url.**
   Step-1 open the project
   Step-2 in intellij idea in terminal run command- $ gradle build

```
Total time: 1.115 secs
udit@machine-control:JavaGradle $ gradle build
Cleaned up directory '/home/udit/geek practice/java-playground/JavaGradle/build/classes/main'
:compileJava
:processResources NO-SOURCE
:classes
:jar UP-TO-DATE
:assemble UP-TO-DATE
:compileTestJava NO-SOURCE
:processTestResources NO-SOURCE
:testClasses UP-TO-DATE
:test NO-SOURCE
:check UP-TO-DATE
:build UP-TO-DATE

BUILD SUCCESSFUL

Total time: 1.12 secs
udit@machine-control:JavaGradle $
```

build.gradle | SaleEvent.java

You can configure Gradle wrapper to use distribution with sources. It will provide IDE with Gradle API/DSL documentation. Hide the tip Ok, apply suggestion!

```
1  plugins {
2      id 'java'
3  }
4
5  group 'org.example'
6  version '1.0-SNAPSHOT'
7
8  sourceCompatibility = 1.8
9
10 repositories {
11     mavenCentral()
12 }
13
14 dependencies {
15     testCompile group: 'junit', name: 'junit', version: '4.12'
16
17     compile 'com.google.code.gson:gson:2.8.0'
18 }
19
```

2. **Using java plugin, make changes in the manifest to make the jar executable. Using java -jar JAR_NAME, the output should be printed as "Hello World"**

step1-add a java class demo in src>main>java

```
public class Demo {
public static void main(String[] args){
    System.out.println("hello world");
}
}
```

step-2 override sourceSets

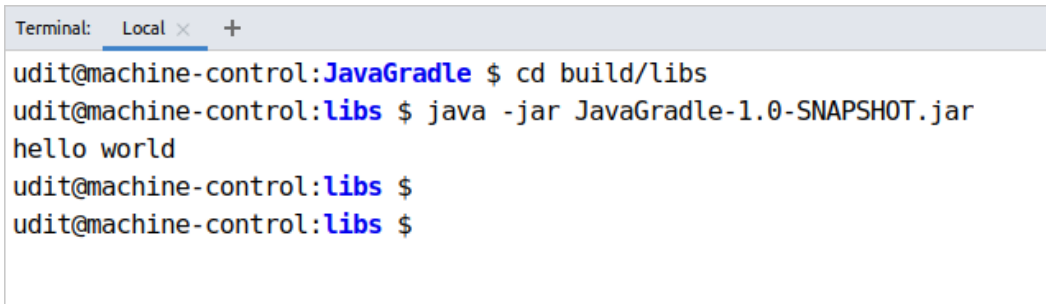
```
// default sourceSets
sourceSets{
    main{
        java.srcDirs=['src']

    }
}
```

Step-3 make jar executable to run demo class

```
jar{
    manifest{
        attributes 'Main-Class':'Demo'
    }
}
```

Step-4- switch to jar directory run the jar by running the command

java -jar <jar-name>



3. **Differentiate between the different dependency scopes: compile, runtime, testCompile, testRuntime using different dependencies being defined in your build.gradle.**

compile-->Dependencies required at compile time but never required at runtime, such as source-only annotations or annotation processors. Compile dependencies are available in all classpaths of a project.these dependencies are also transitive available to sub projects:

runtime-->The dependencies with this scope are required at runtime, but they're not needed for compilation of the project code. Because of that, dependencies marked with the *runtime* scope will be present in runtime and test classpath, but they will be missing from compile classpath.

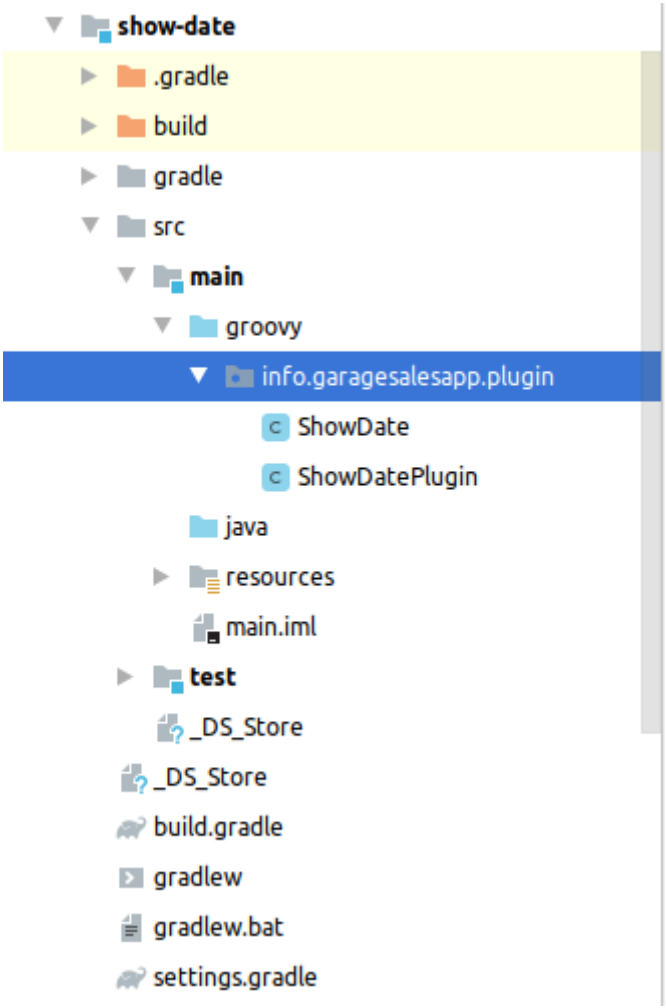
testCompile--> testCompile is a group of dependencies that might be needed only for testing while compile is the group of dependencies that might be needed to build the application . The testCompile configuration contains the dependencies which are required to compile the tests of the project. This configuration contains the compiled classes of the project and the dependencies added to the compile configuration.

runCompile: The *testRuntime* contains the dependencies which are required when our tests are run. This configurations contains the dependencies added to *compile*, *runtime*, and *testCompile* configurations.

```
dependencies {
    testCompile group: 'junit', name: 'junit', version: '4.12'
    compile 'com.google.code.gson:gson:2.8.0'
    testRuntime 'com.google.code.gson:gson:2.8.0'
    runtime 'com.google.code.gson:gson:2.8.0'
}
```

4. **Create a custom plugin which contains a custom task which prints the current date-time. Using that plugin in your project, execute that task after the jar task executes.**

Step -1 create a module inside root module



step-2 write the plugin ShowDate ShowDatePlugin

```groovy
package info.garagesalesapp.plugin

import org.gradle.api.DefaultTask
import org.gradle.api.tasks.TaskAction

class ShowDate extends DefaultTask {

    String dateMessage = 'Current date: '

    @TaskAction
    def showDate() {
        println ""
        println dateMessage + new Date()
        println ""
    }
}
```

```groovy
package info.garagesalesapp.plugin

import ...

/**
 * Created by jamesharmon on 5/3/17.
 */
class ShowDatePlugin implements Plugin<Project> {

    @Override
    void apply(Project project) {
        project.task( name: 'showDate', type: ShowDate)
    }
}
```

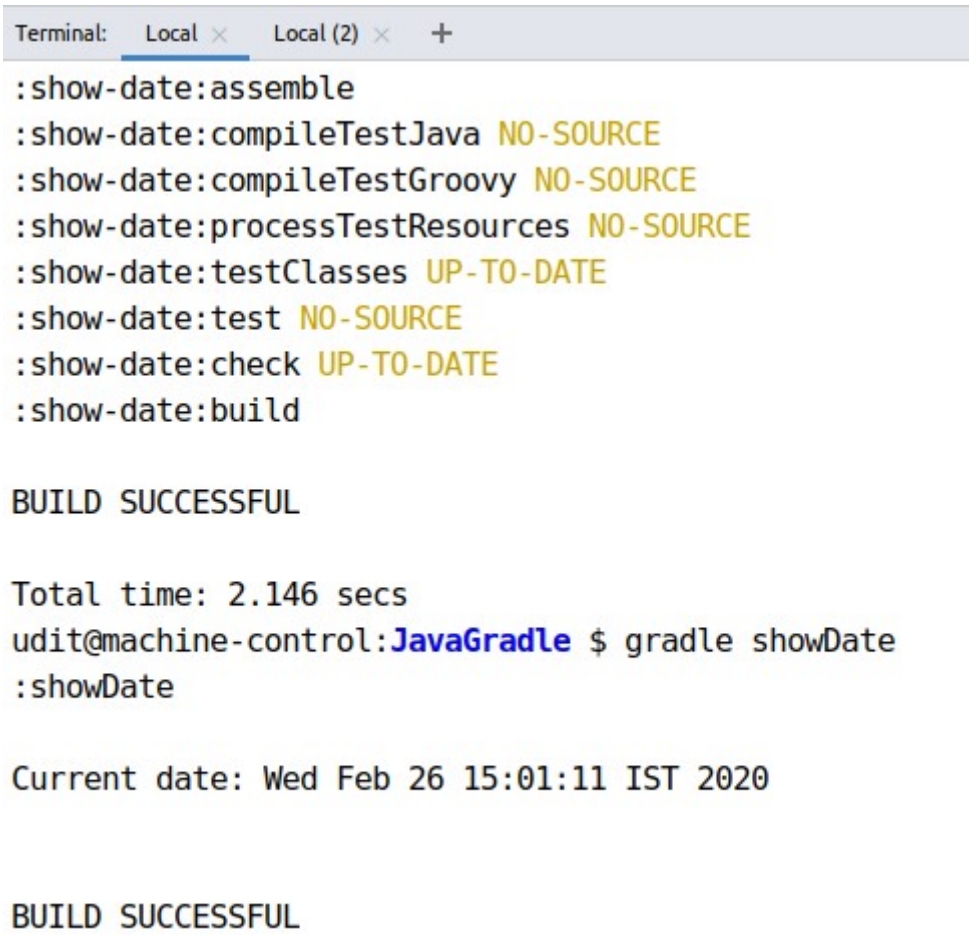
step 3- make changes build.gradle(:show-date) and build.gradle(JavaGradle)

```
group 'info.garagesalesapp'
version '1.0-SNAPSHOT'

apply plugin: 'groovy'

dependencies {
    compile gradleApi()
    compile localGroovy()
}
```

```
// Q-2 crating java plugin
buildscript {
    dependencies {
        classpath files('show-date/build/libs/show-date-1.0-SNAPSHOT.jar')
    }
}
```

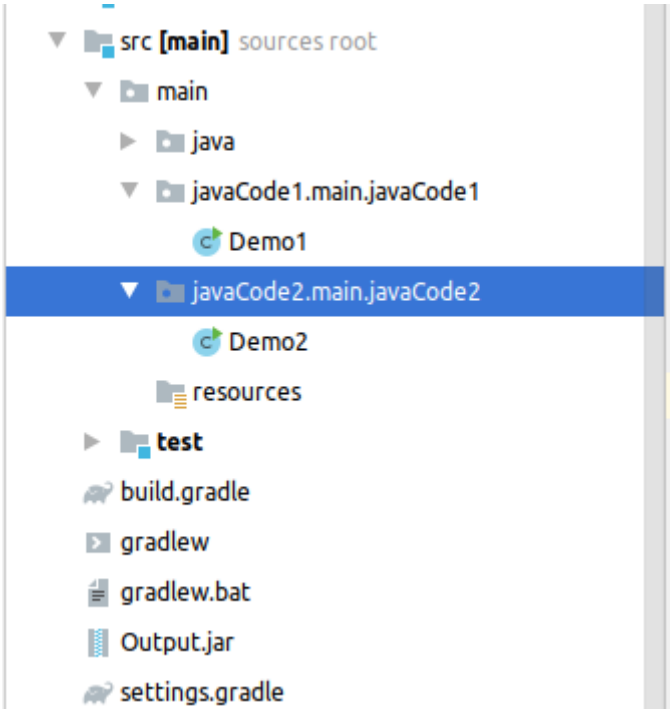
```
//Q-4
apply plugin:'show-date-plugin'

//Q-6
```

step 4- Run plugin with command $gradle showDate



5. **Instead of using default source set, use src/main/javaCode1, src/main/javaCode2 to be taken as code source. Make sure that the JAR created contains files from both the directories and not from src/main/java.**

Step-1 create directories javaCode1 and java Code2 in src/main



Step-2 overwrite the default source set

```
//Q-5 overwritten sourceSets
sourceSets{
    main{
                    java.srcDirs 'src/main/javaCode1'
                    java.srcDirs 'src/main/javaCode2'
    }
}
```

Step-3 clean build the gradle with $./gradlew clean build
Step-4 see the content of the jar file with command

```
main/javaCode1/
main/javaCode1/Demo1.class
Demo.class
udit@machine-control:libs $ jar tf JavaGradle-1.0-SNAPSHOT.jar
META-INF/
META-INF/MANIFEST.MF
SaleEvent.class
main/
main/javaCode2/
main/javaCode2/Demo2.class
main/javaCode1/
main/javaCode1/Demo1.class
Demo.class
udit@machine-control:libs $
```

6. **Override the Gradle Wrapper task to install a different version of gradle. Make sure that the task written in Q4 also executes with it.**

Step-1 create a wrapper task to overwrite the default wrapper task

```
//Q-6
wrapper {
    gradleVersion = '5.2.1'
    distributionUrl = distributionUrl.replace("bin", "all")
}
```

Step-2 check the gradle version in gradle>wrapper>gradle-wrapper.properties

build.gradle (JavaGradle) | build.gradle (:show-date) | ShowDate.groovy | ShowDatePlugin.groovy | gradle-wrapper.properties

```
#Wed Feb 26 13:35:05 IST 2020
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
distributionUrl=https\://services.gradle.org/distributions/gradle-5.2.1-all.zip
```

step-3 after gradle build. Run custom plugin showDate with command $gradle showDate

Terminal: Local | Local (2) | +

```
BUILD SUCCESSFUL in 31s
7 actionable tasks: 7 executed
udit@machine-control:JavaGradle $ gradle showDate
Cleaned up directory '/home/udit/geek practice/java-playground/JavaGradle/show-date/build/resources/main'
:showDate

Current date: Wed Feb 26 13:31:36 IST 2020


BUILD SUCCESSFUL

Total time: 1.092 secs
udit@machine-control:JavaGradle $
```

7. **Run the gradle profile command and attach the resulting files.**

   Step-1 run the command $gradle buld --profile to crate profile of the project

Profile report

Profiled build: build

Started on: 2020/02/26 - 15:45:23

Summary Configuration Dependency Resolution Task Execution

Description	Duration
Total Build Time	2.840s
Startup	0.779s
Settings and BuildSrc	0.001s
Loading Projects	0.008s
Configuring Projects	0.569s
Task Execution	1.459s