Gradle Command Line Interface Basics

If you want to interact with gradle outside the ide, then use the command line interface!

./gradlew.bat [taskName…] [--option-name]

./gradlew.bat taskName –exampleOption=exampleValue

Settings File Basics

The settings file is the entry point of every gradle project!!

This is in the project- and it is named settings.gradle

What is the purpose of the settings.gradle file ?

It is to add subprojects to the build

Therefore, the settings file is mandatory only for multi-project builds.

Settings.gradle example

rootProject.name = 'root-project'

include('sub-project-a')

include('sub-project-b')

include('sub-project-c')

Build file basics

A build file/script details build config, tasks and plugins

A build file can contain 2 types of dependencies

1. Dependencies of the build script
2. Dependencies of the source code

Build.gradle example

plugins {

id 'application'

}

application {

mainClass = 'com.example.Main'

}

In this case, we have added a plugin called application

This makes additional functionality available to a build.

This application plugin that we have added above, allows creating an executable JVM application

Inherently, the application plugin also adds the java plugin.

The java plugin brings in the following functionalities – compilation,testing,bundling

In the above, after we have added the “application” plugin,

This particular plugin, what It does is that it adds the “run” task to the build.

The “run” task, however, needs to know where is the mainClass and hence this is added in the below application curly braces

Dependency Management basics

What is dependency management?

It is an automated technique for declaring and resolving external resources required by a project

Dependencies include: JARs, plugins, libraries, source code

What is a version catalog?

This basically refers to libs.versions.toml file

Example of libs.versions.toml file

[versions]

androidGradlePlugin = "7.4.1"

mockito = "2.16.0"

[libraries]

googleMaterial = { group = "com.google.android.material", name = "material", version = "1.1.0-alpha05" }

mockitoCore = { module = "org.mockito:mockito-core", version.ref = "mockito" }

[plugins]

androidApplication = { id = "com.android.application", version.ref = "androidGradlePlugin" }

this version catalog allows for enforcement of versions of libraries and plugins in large projects

Declaring dependencies

plugins {

alias(libs.plugins.androidApplication)

}

dependencies {

// Dependency on a remote binary to compile and run the code

implementation(libs.googleMaterial)

// Dependency on a remote binary to compile and run the test code

testImplementation(libs.mockitoCore)

}

As can be seen above, the dependencies can be added by referring to the version catalog.

In the above we can see that there are two types of dependencies

1. Implementation 🡺 used for compiling and running production code
2. testImplementation 🡺 used for compiling and running test code

how to view all the dependencies of a project?

./gradlew :app:dependencies

Task Basics

What is a task? – it is an independent unit of work

An example is compiling,

Or generating a JAR

Or publishing archives to a repo

For example the build is a gradle task

And you run it by

./gradlew.bat build

Available Tasks

All the available tasks of a project come from

* gradle plugins
* build scripts

how to see all tasks of a gradle project ?

./gradlew.bat tasks

How do I run tasks?

./gradlew.bat run

This will compile the code and run it – will NOT create a jar

What is task dependency ?

* a task requires another task to run first.
* As an example
* When you run ./gradlew.bat build
* This build task depends on the compileJava task

So when you run ./gradlew.bat build, the following is seen

$ ./gradlew build

> Task :app:compileJava

> Task :app:processResources NO-SOURCE

> Task :app:classes

> Task :app:jar

> Task :app:startScripts

> Task :app:distTar

> Task :app:distZip

> Task :app:assemble

> Task :app:compileTestJava

> Task :app:processTestResources NO-SOURCE

> Task :app:testClasses

> Task :app:test

> Task :app:check

> Task :app:build

BUILD SUCCESSFUL in 764ms

7 actionable tasks: 7 executed

Since the build task DEPENDS ON the compileJava task, the compileJava task has to run first!