## Links for reference

|  |  |
| --- | --- |
| Junit user guide | <https://junit.org/junit5/docs/current/user-guide/#writing-tests> |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Gradle

Its very fast (it will keep a track of what is recently changed recently so it will compile only changed files ), incremental builds,

Ex:- if code is not changed and only if test cases are changed then it will just run the test cases , it wont compile the source code again

We can write the code of build.gradle in DSL (domain specific language) in either groovy or kotlin

we will write build.gradle in groovy syntax

note: - gradlew.bat will download the gradle software into our machine in “.gradle” folder, so we don’t need to install the gradle at all

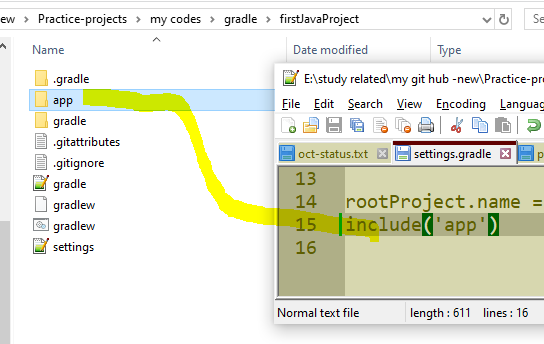
in “gradle-wrapper.properties” we will mention which gradle version we want

Gradle wrapper is responsible to download & install the mentioned gradle versions

Each project will have separate gradle wrapper

Important points

1. Even if build.gradle is not there in current directory, settings.gradle file will be there, where we mention the project name to include, then gradle will go into that folder



See here in this folder we don’t have build.gradle, but in settings.gradle we included the project named “app” inside that folder we will have the build.gradle

Gradle Commands

|  |  |
| --- | --- |
| To list/see all tasks | gradle tasks --all |
| To execute a task | gradle <task name> ex:- gradle test, gradle clean |
| To create gradle project – we can create java project with this command | Gradle init |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Tasks

Note:- u can print the data in both single or double quotes

1. Creating a basic task

Here manicopy is the task name

|  |  |
| --- | --- |
| task manicopy{  println 'manideep'  } | task manicopyD{  println "manideep"  } |

Predefined tasks:-

gradle <task-name>

|  |  |
| --- | --- |
| To compile | gradle app:compileJava - Compiles main Java source., here app is the project name |
| To run the test cases | gradle test // this will run the test task and execute test cases |
| To create jar file | gradle jar |
| To run the jar file | gradle run |
|  | Gradle build |

Plugin

1 plugin will extend another plugin and from 1 plugin we will get lot of tasks

Scopes

compileOnly(only for compilation purposes like lombok), runtimeOnly(use only if these jars are provided by server while deploying )

implementation, testImplementation

Build phases

1. Initialization : - it will verify the settings.gradle and checks if it is a single or multi module project
2. Configuration : here it will check all tasks and creates DAG- direct acyclic graph of all tasks which one to execute first and which to execute later
3. Execution:- it will execute all tasks

FQA

1. how to make gradle to use certain version of java to build

keep this in build.gradle

java **{  
 sourceCompatibility** = JavaVersion.***VERSION\_11* targetCompatibility** = JavaVersion.***VERSION\_11*}**

1. we dont need to mention the version of jar then how gradle is exactly fetching that single jar??

it will intenally use spring boot dependencies bom bill of materials that bom contains all the compatible jar versions 16

in gradle How to reference a jar that Is present locally

To add a jar locally

* paste that jar in libs/new folder
* add that lib/jar to project "module settings" -->"libraries"--> add that library (i mean fol
* add this entry in build.gradle

implementation fileTree(dir: 'libs/new', include: ['\*.jar'])

settings.gradle

repositories {

flatDir {

dirs("libs")

}

When external Libraries are not visible

then execute a plugin called "idea" then it will generate a iml file with all dependencies, then and then all those ext lib will be visible

plugins {

id 'java-library'

id 'maven-publish'

id 'jacoco'

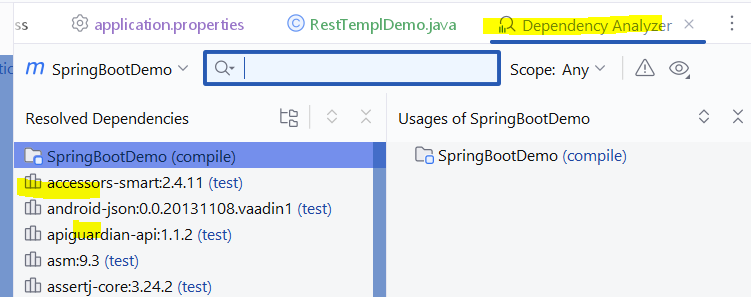
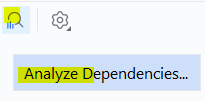
id 'idea'

}

command --> \gradlew.bat IDEA

How to see finalized dependencies in our project

1. u can see in left side Intellij External dependencies
2. using analyse dependencies icon in gradle tab and from which parent that jar came it is like



This is Intellij feature it seems

U can see all these dependencies / jars finally took into our project