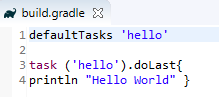
Groovy runs inside the jvm

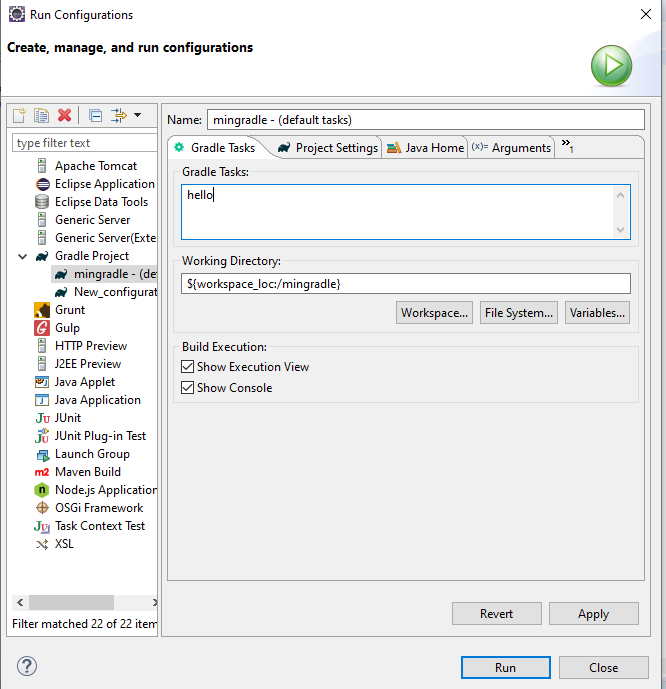
Each gradle Project will have a build.gradle file

Ex:

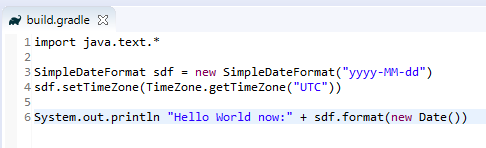


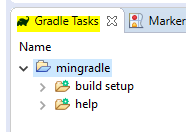


Default tasks will run the specified tasks without the need to call it explicitly

on the run configurations you can add the gradle tasks you want to run

Gradle allow java code:





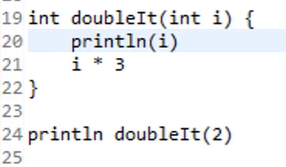
Import project as gradle to jave it in gradle task view (show gradle tasks view)

You can define methods and call them as in java in the gradle script

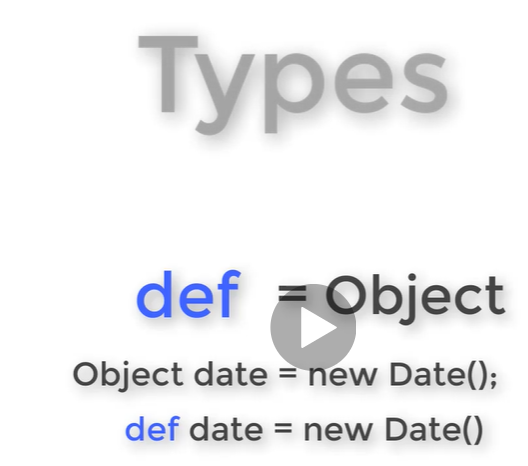




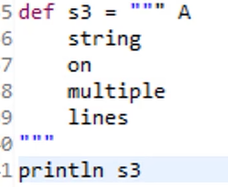
Return statement is optional in groovy, the last line of the method would be returned:



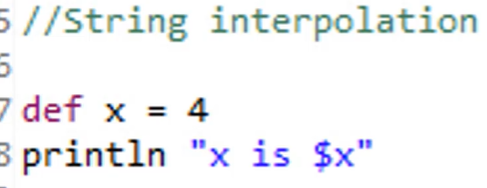
Groovy types

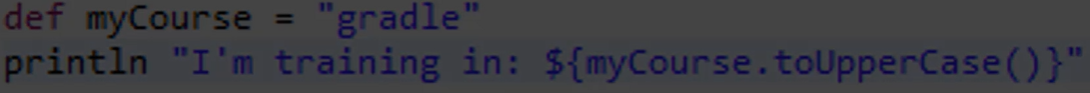


Groovy string

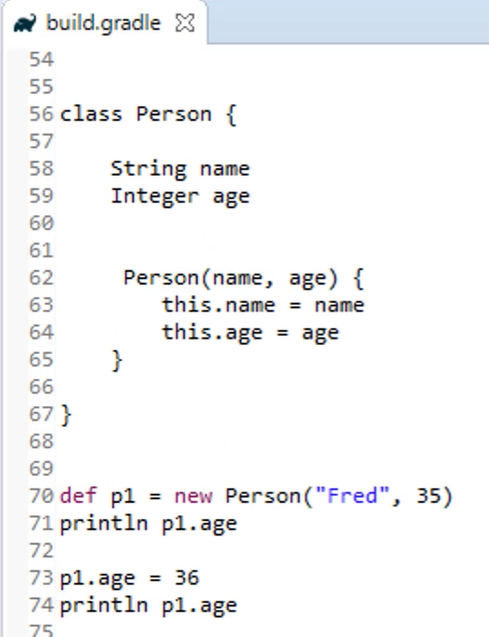


String interpolation

 = x is 4



Groovy objects

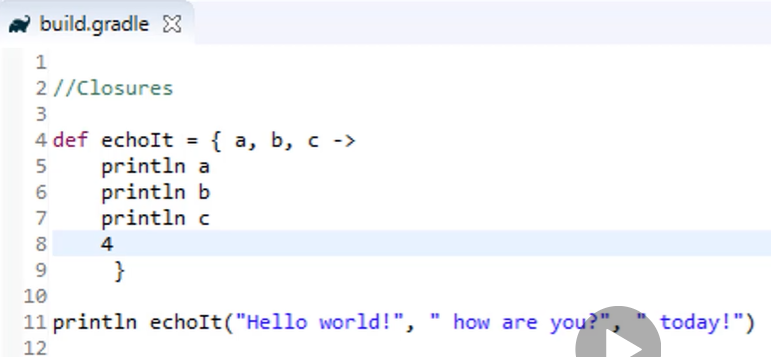


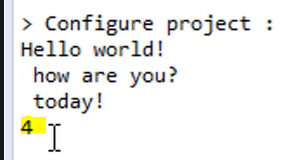
Groovy Closures { }

Similar to lambda

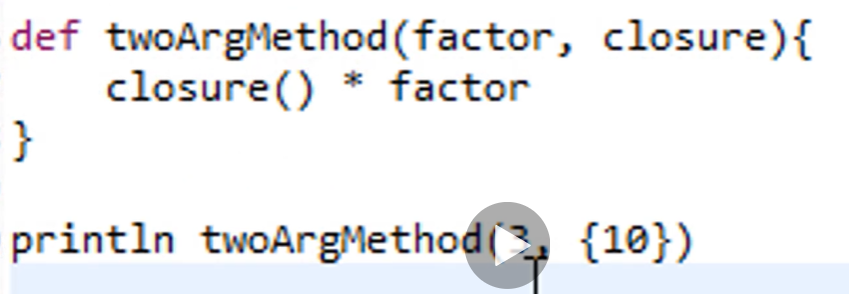
It says we want to invoke what is inside { }

Similar to methods the last line of the closure will be returned

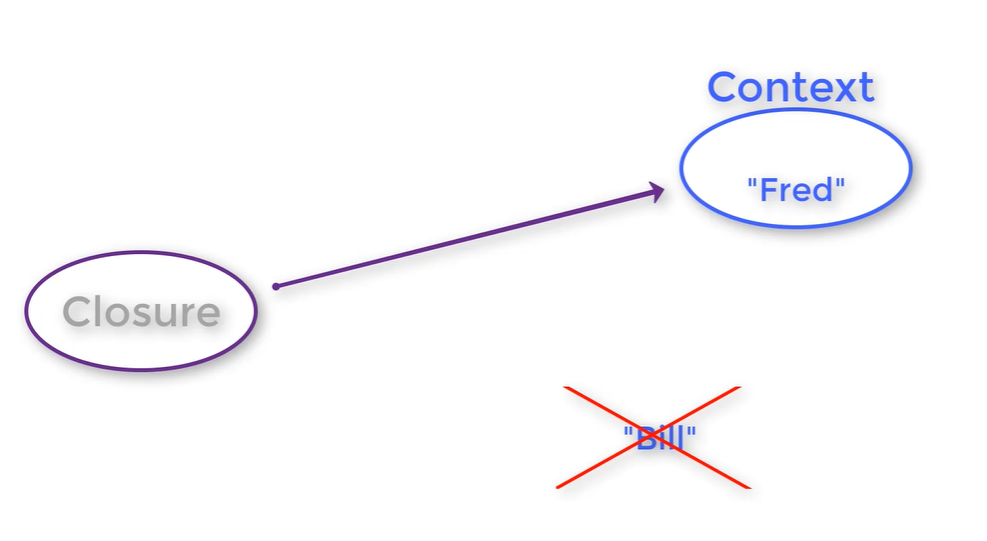




You can pass a closure as argument



New value for closure context



Don’t keep the value, keep a reference to the context in case it changes, it takes the updated one

Key Gradle interfaces

Script (any file ending with .gradle implement it)

Methods that are available with all scripts

Project

Gradle

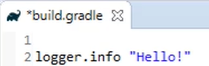
Settings

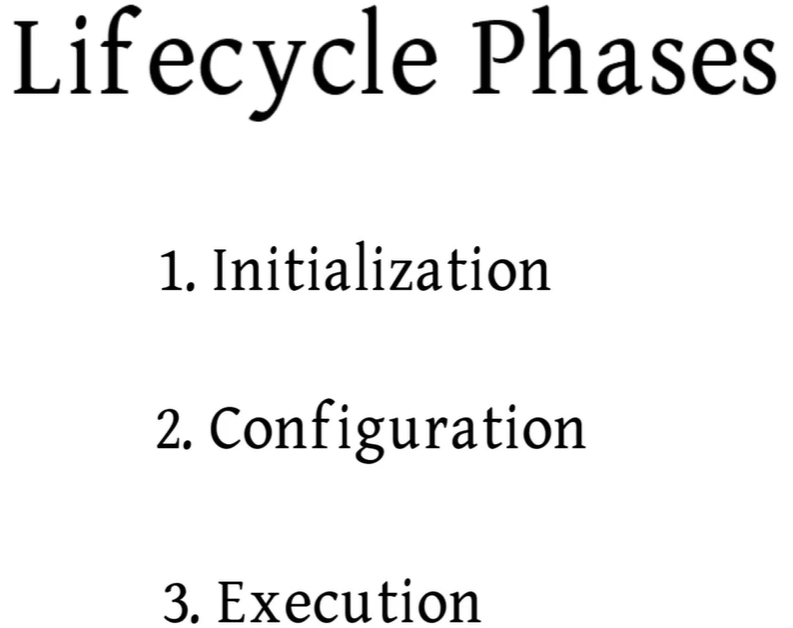
For multirojects

Task

Action

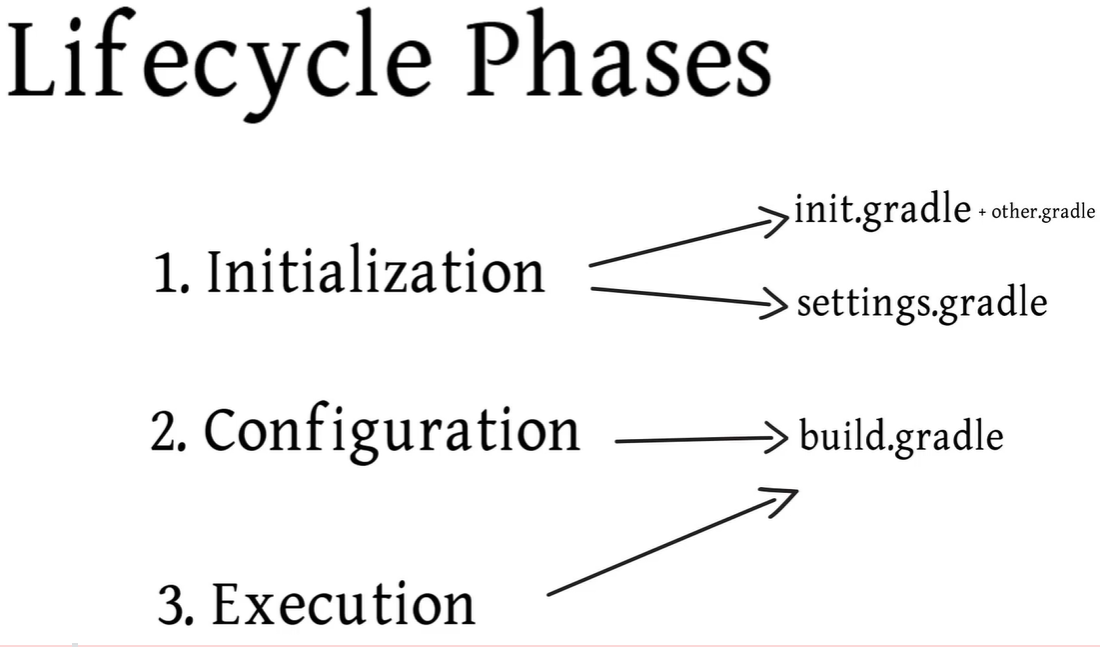
Logging, you can use directly the logger on the script thanks to the Script project





Initialization

Gradle determines which projects will be part of the build and creates a project instance for each these are part of the build

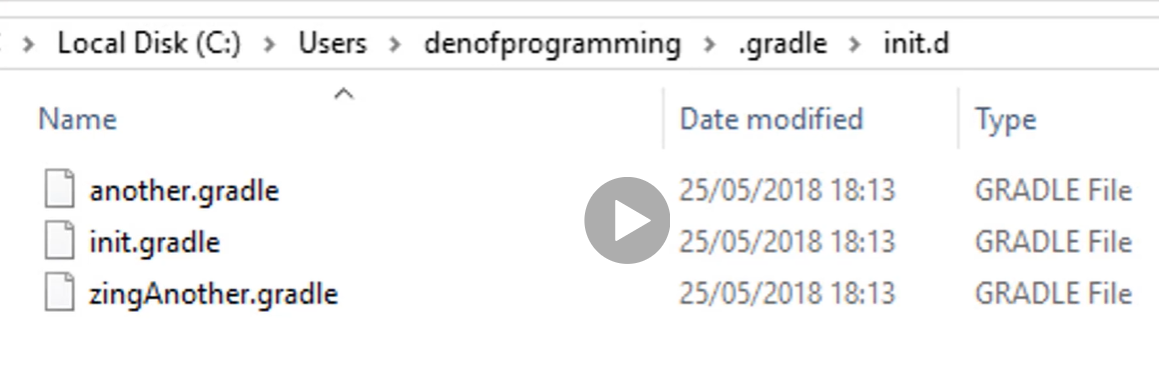


Initialization phase

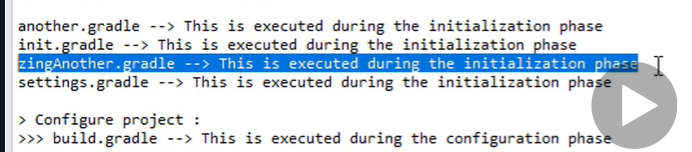
Runs the init.gradle and any other file in a specific located location with sintaxis <XXX>.gradle (except build.gradle and settings.gradle) help to setup the environment before executing our build

Settings.gradle

You can specify what projects will be included in the build



When the initialization phase starts, it looks in the .gradle/init.d location for gradle files to be executed at initialization, starting with alphabetical order of execution



Configuration Phase and Execution Phase

Relies on the build.gradle file, there needs to be one build.gradle for each project on the build

Gradle Object

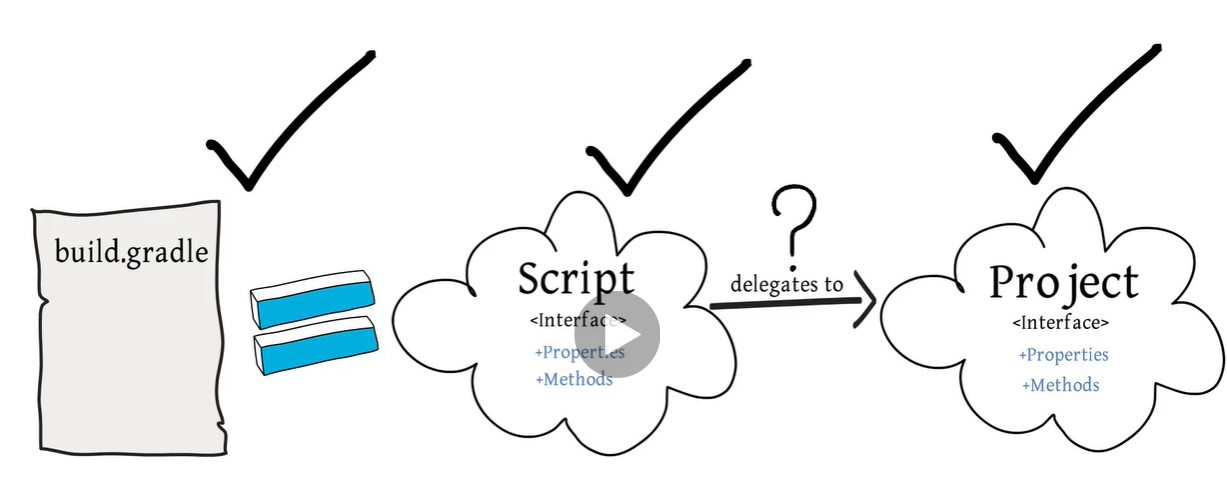
We have access to it for every script during the entire build lifecycle



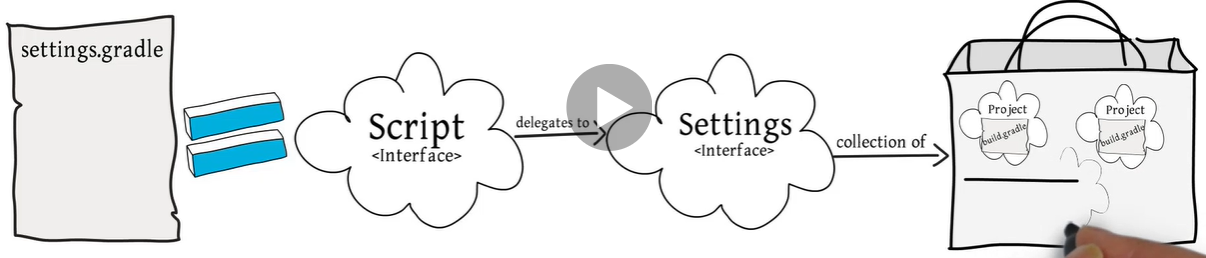
Build.gradle

Represents a build script

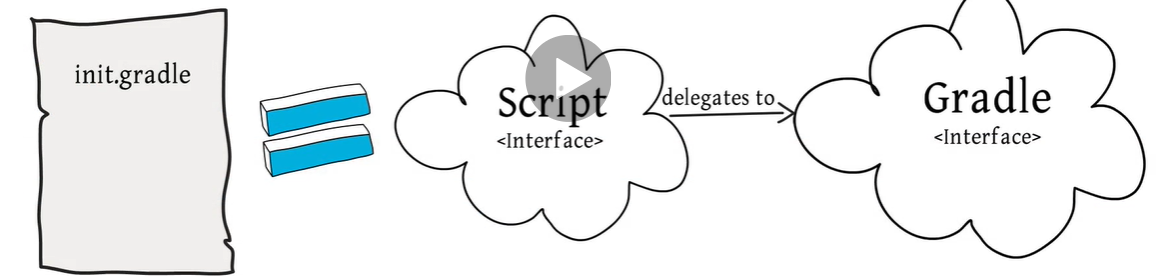
When we run the build, gradle creates an object which implements the Script interface for each file with .gradle extension



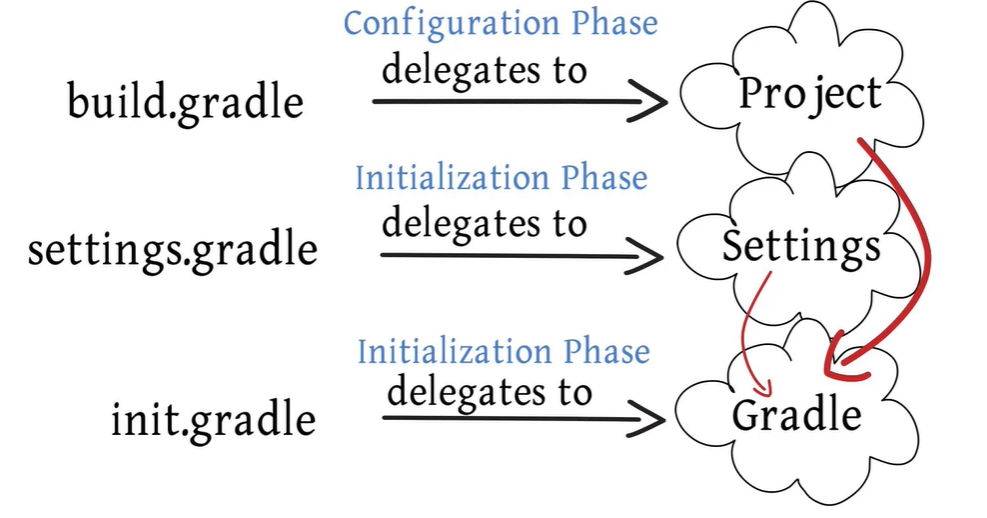
Project interface is only available at configuration phase (build.gradle)



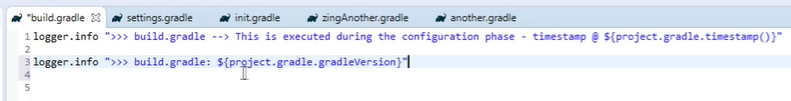
Settings interface available at initialization for settings.gradle



Gradle interface available at initialization phase for init build files

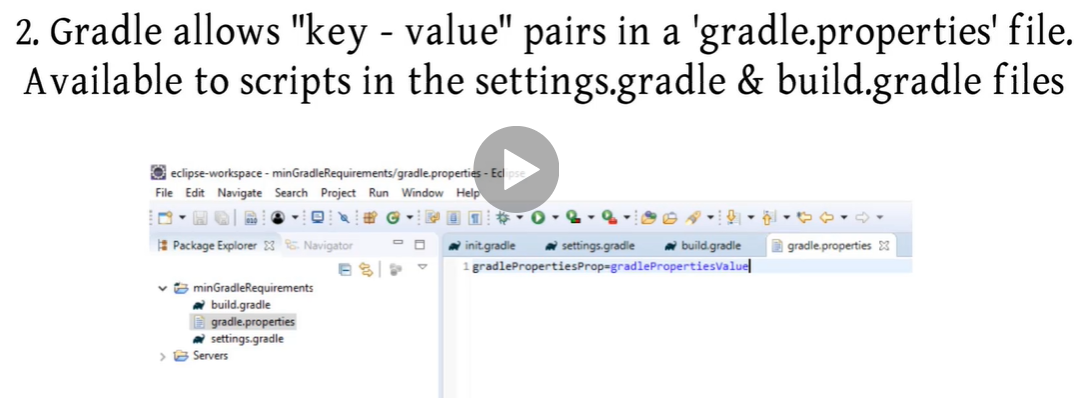


Project and Settings have access to the Gradle object



Gradle.properties

Available for all the scripts



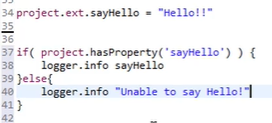
It can live in the root directory or in GRADLE\_USER\_HOME

Properties can be send from the command line as well



Extra properties extension

Allow properties to be added to existing domain objects



Execution Phase

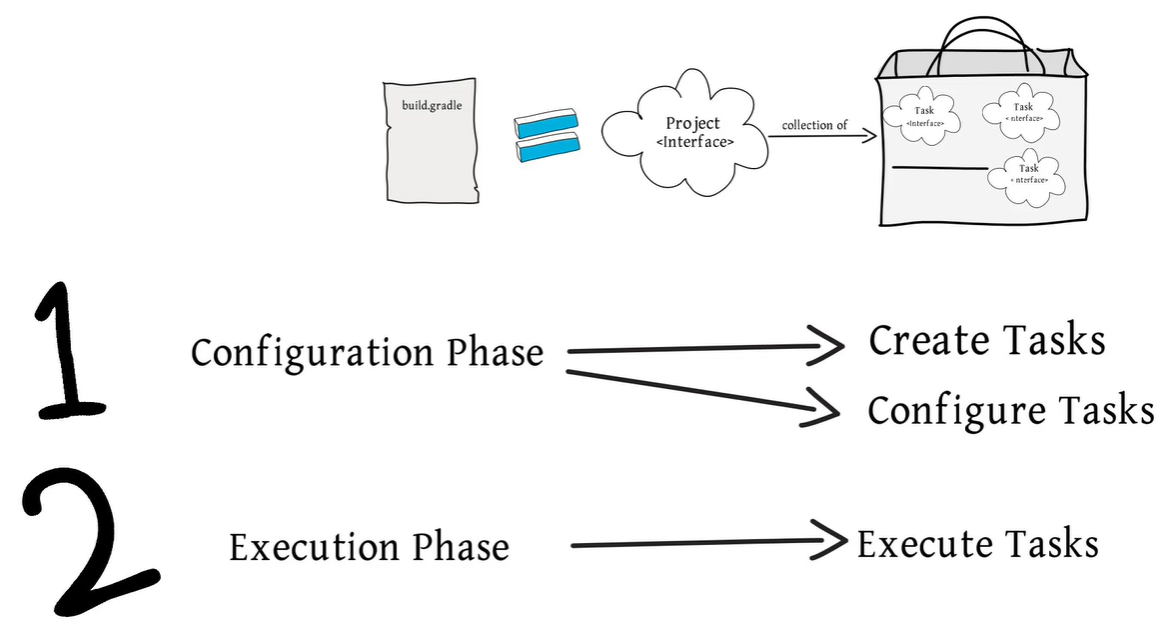
Every gradle build is one or more projects, a project represents depends on what we are trying to do with gradle (a thing that must be build, a thing that needs to be done)

A project is a collection of tasks

Tasks

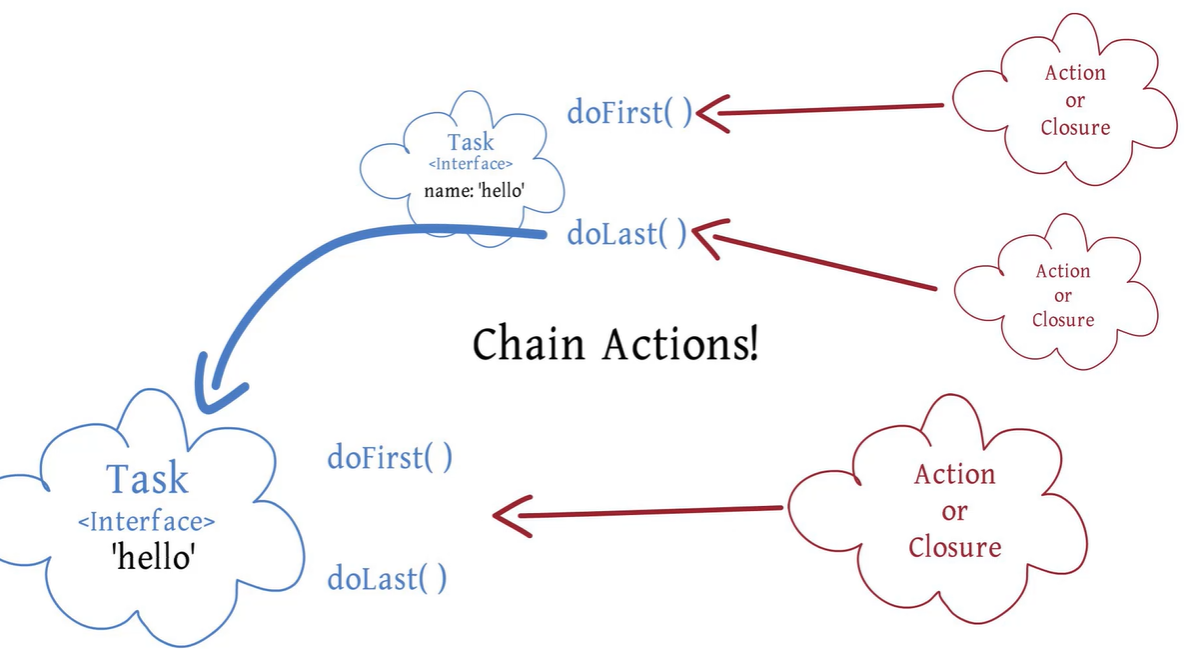
Represent an atomic piece of work to be performed by th build

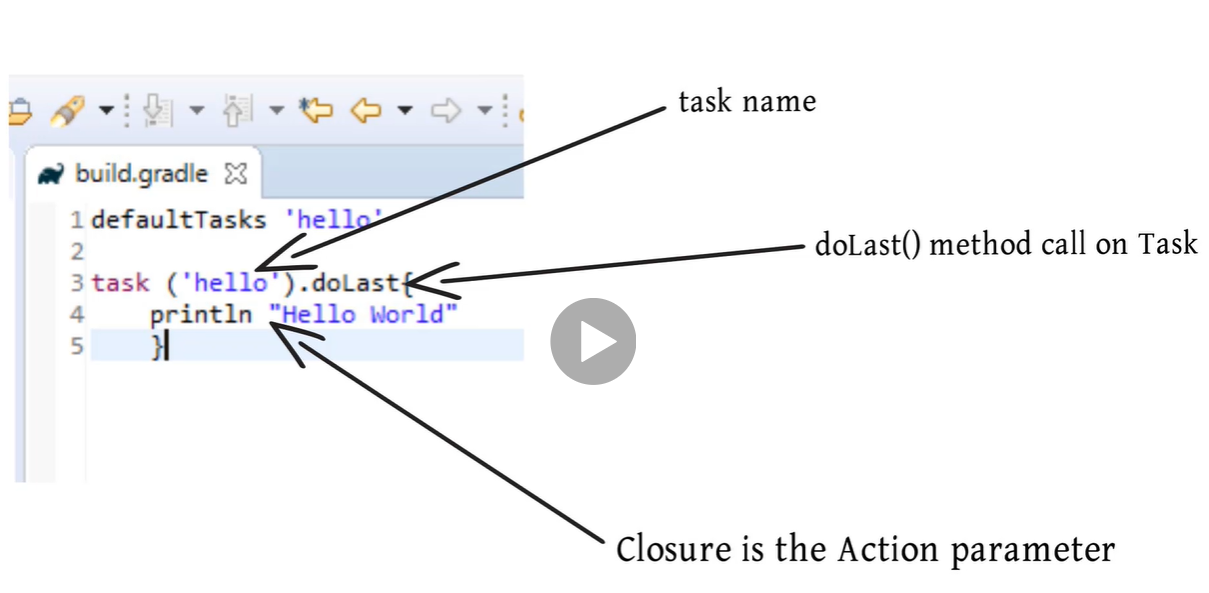
A task can be 0 or many actions (collection of actions)



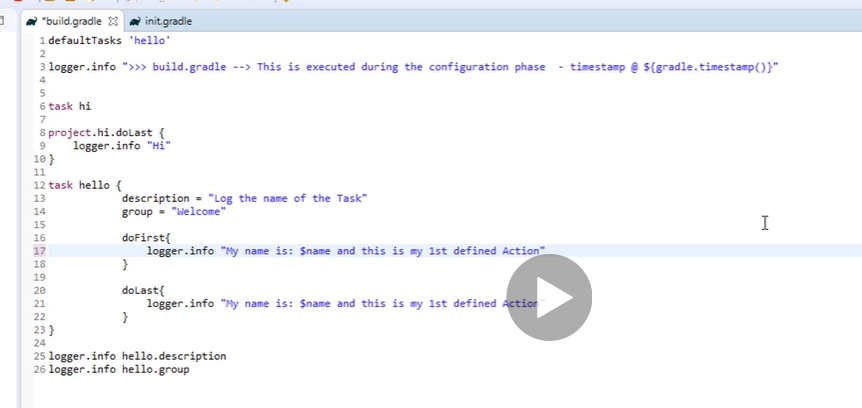
When a task is executed each action is executed

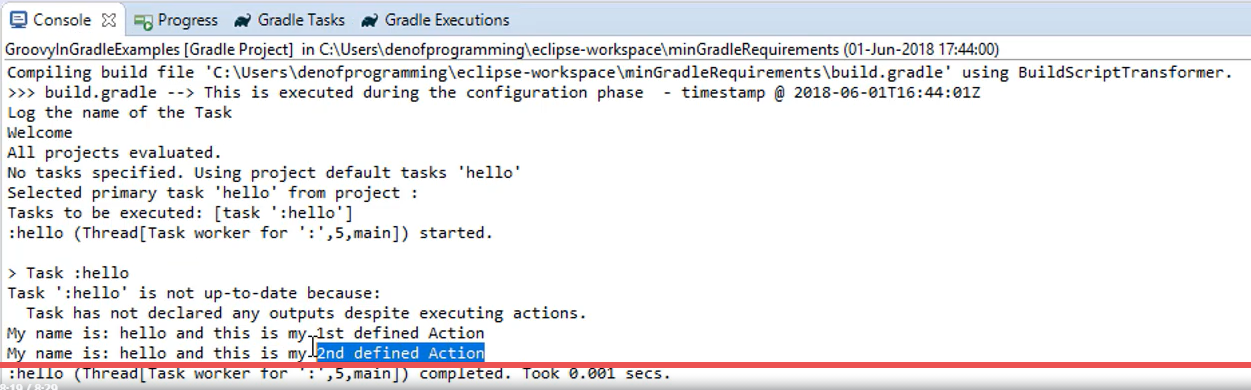
Actions



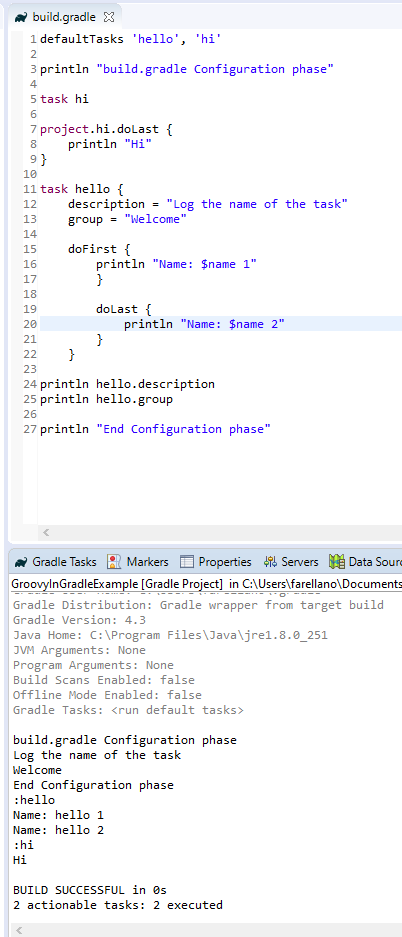


Example





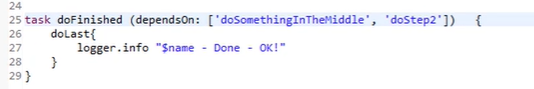
First configuration phase, will run the loggers, then since default tasks is only hello, this will be the only task executed at execution phase

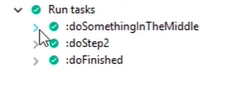


All tasks are first gathered depending there position first or last and after all have been gathered they are executed in execution phase

Sequence tasks

dependsOn the task will be run once the specified task is executed (it forces it to run)





If a task is mentioned twice in depends on on a different task it wont run 2 times, just the first time it was needed



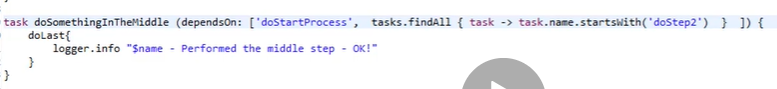


Depends with closure

The result of the closure must be a single task or a collection of task objects

# 

Filter from Task allows to get the tasks which are depending on a task for example



doSomethingInTheMiddle task will be dependent on doStartProcess and all the tasks which name starts with doStep2

equivalent

 = 



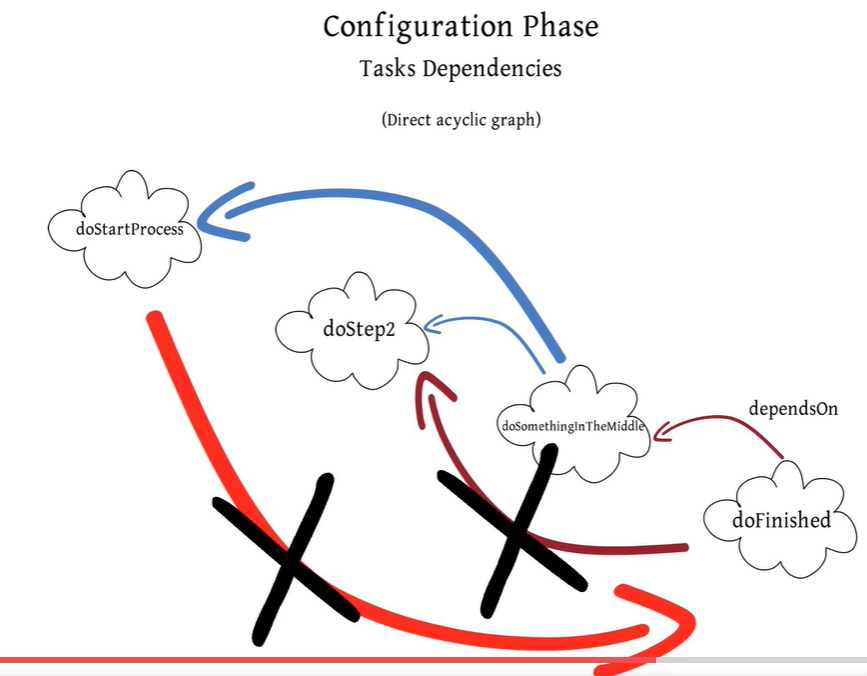
conditional dependsOn



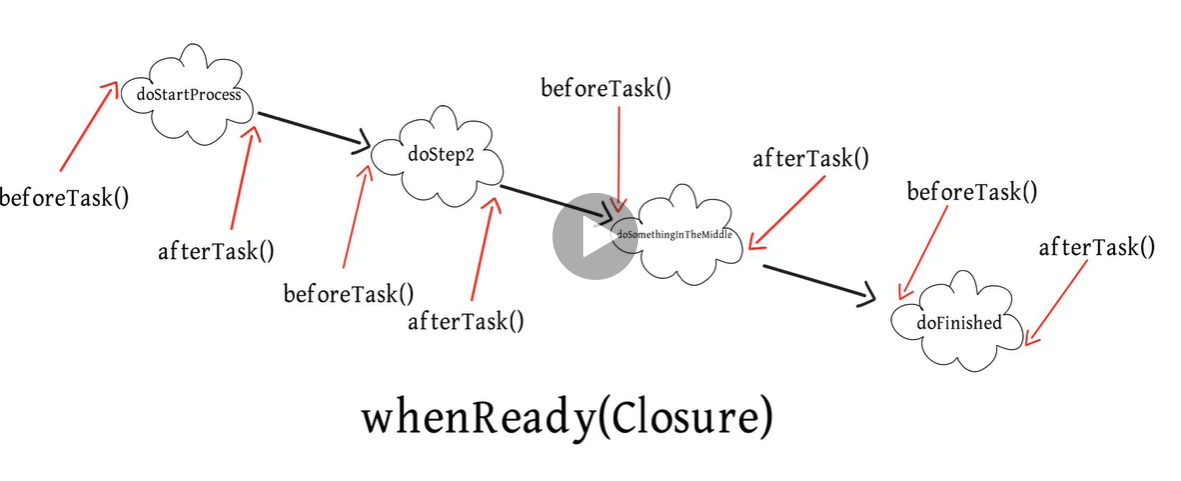
It might be useful if you validate a property being set at initialization to execute or not a task

Task dependency graph.

Help to identify when a task should be executed and not producing a infinite loop



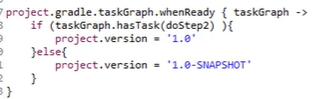
NOTE: if for example in the task you add a println but its directly inside the task, it will be executed at configuration phase, no matter if it is not called by the task, if you add it inside a doFirst or doLast, then it will only be executed when the task is called!



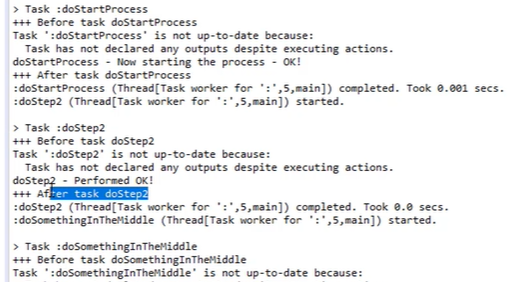
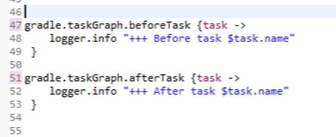






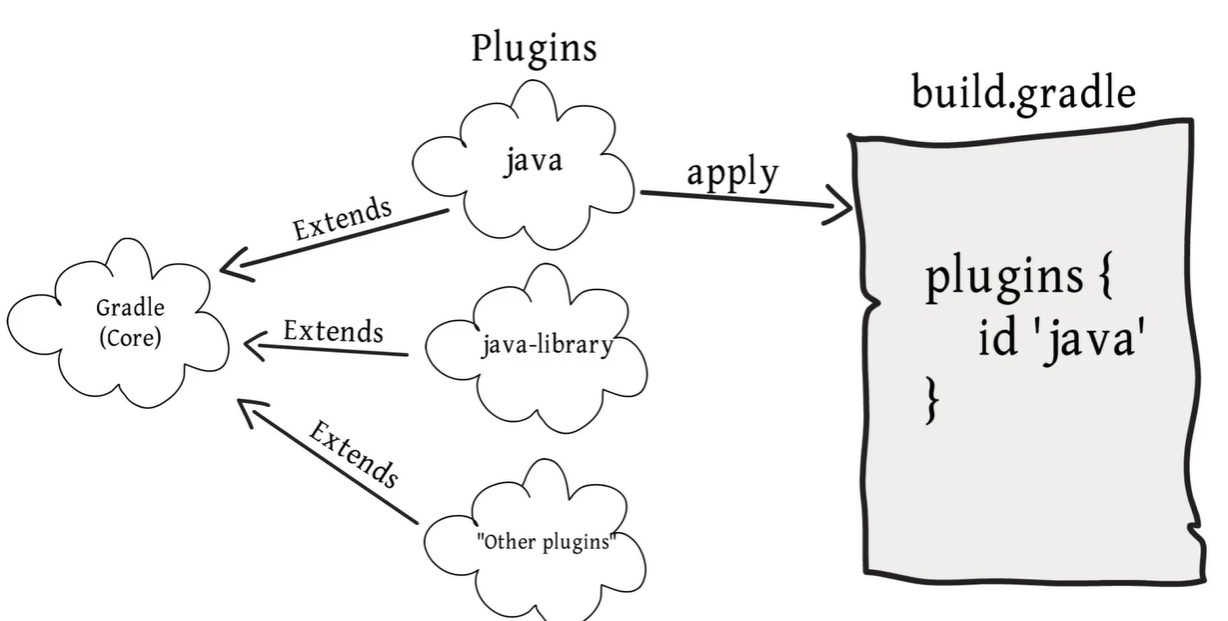


beforeTask, afterTask

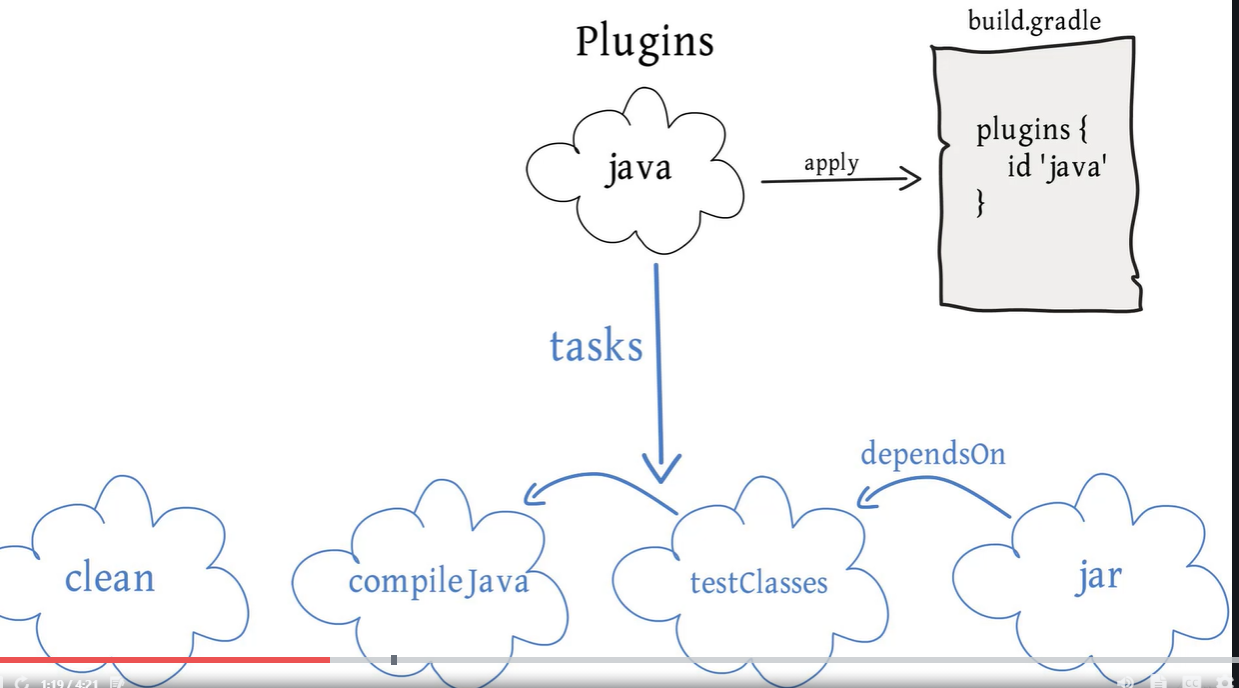
 

Plugins

Allows to extend the project capabilities



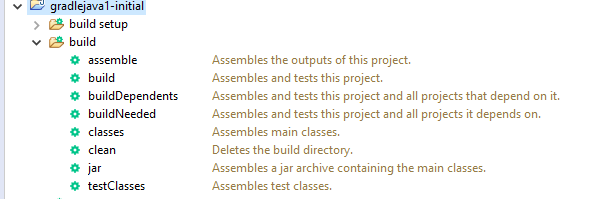
Some Tasks added by applying the java plugin



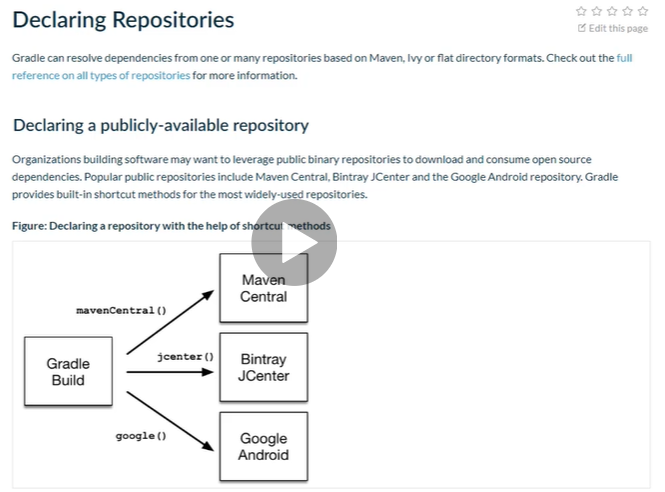
Add java to gradle

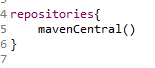
 OR 

It includes tasks, lifecycle tasks, clean, compile, etc



Repositories





Inside repositories it is a closure, a different context so you have access to differents methods like mavenCentral()



Compile, testCompile have been deprecated, use implementation and testImplementation

Execute jar from command line





Class that needs to be invoked when called from the comandline

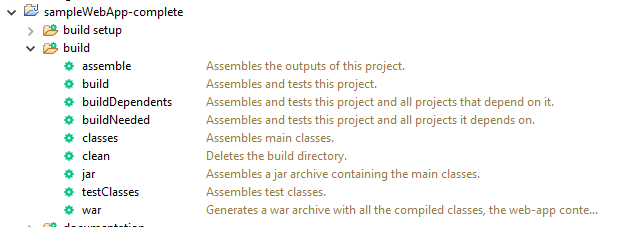
Jar task

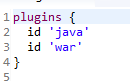


Gives us access to the jar task

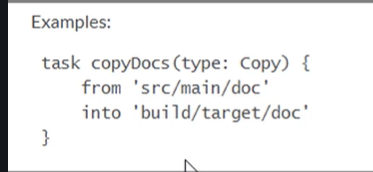


War task added by adding the plugin war to the build.gradle





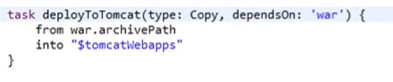
Adding task type

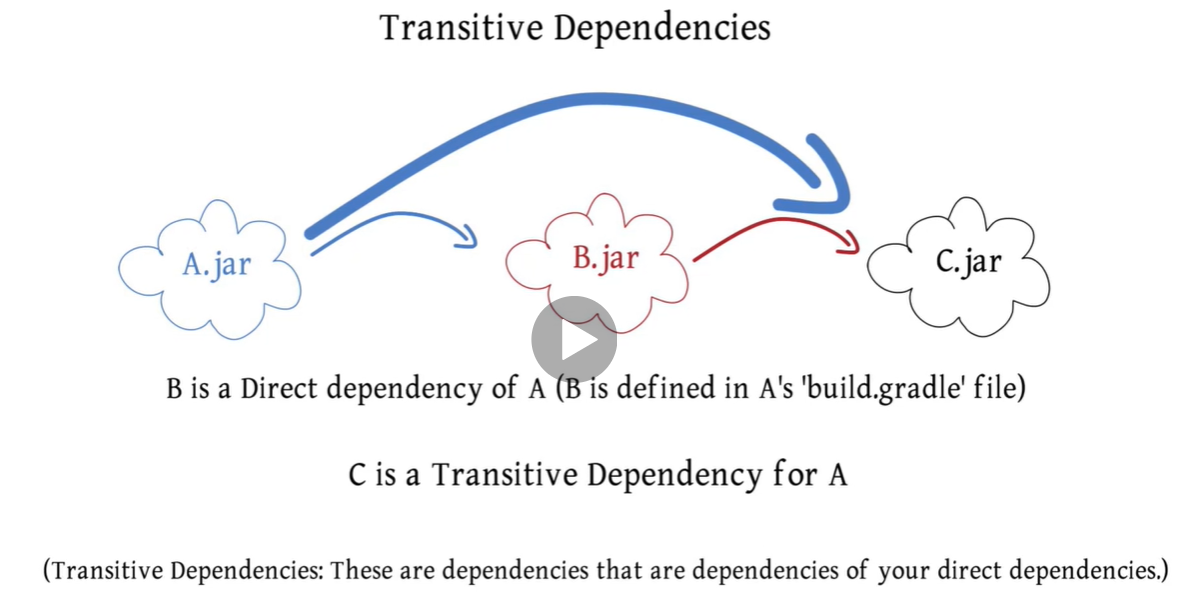


If we add a type for a task it will inherit the properties and methods the parent has

Ex deploy war to tomcat (copy the war file to the tomcat webapps directory)







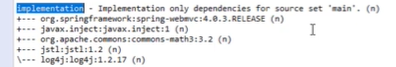
Transitive dependencies, dependencies from your dependencies

Analize dependencies

On the help task if run gives a list of dependencies



Implementation section, direct dependencies

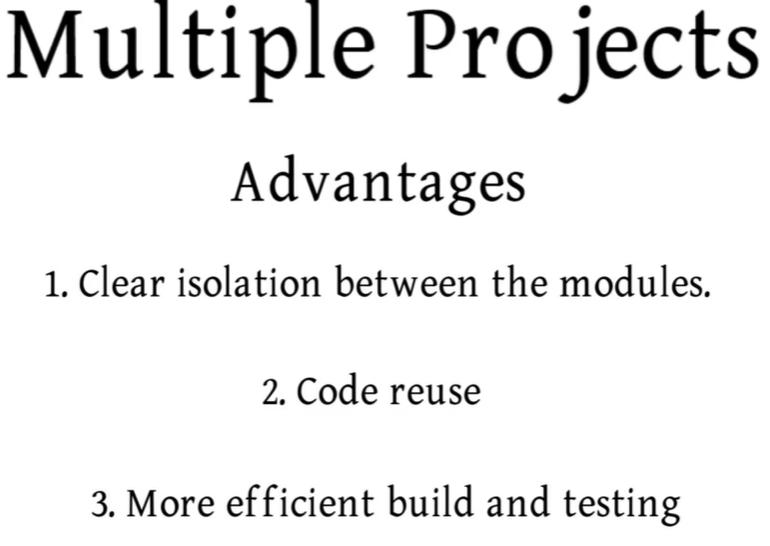


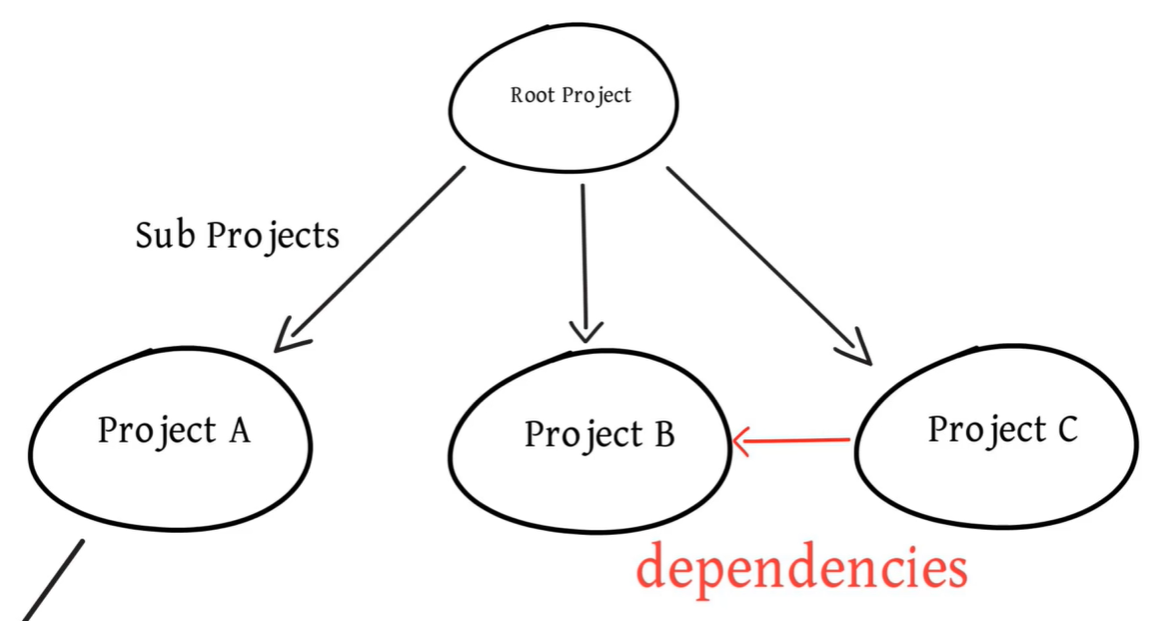
Exclude a transitive dependencies

That transitive dependencie wont be downloaded



Multiple projects

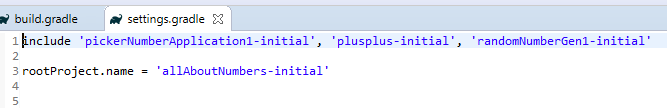




Settings.gradle

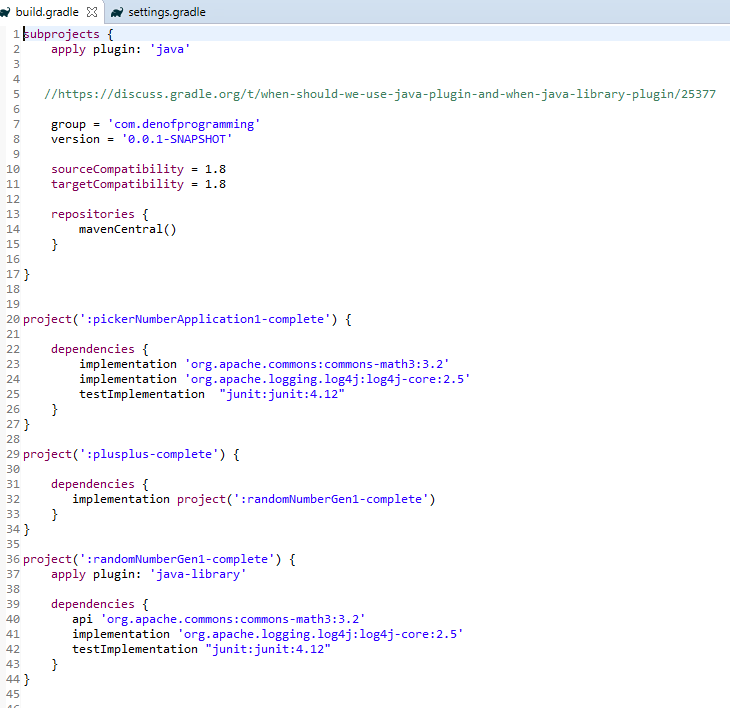
Mandatory for multi projects build

Not needed on a single project build



Include which projects will be part of the build (on the root project)

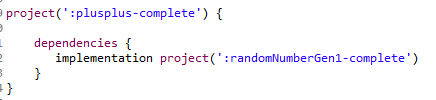
Root build.gradle



This will be inherit for the subprojects

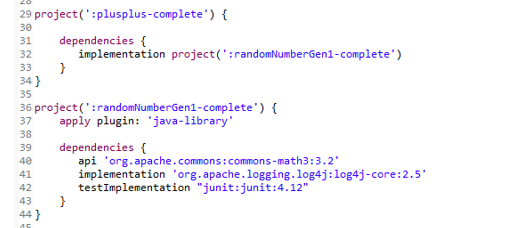
common config for all subprojects

config for a specific project



Dependencie with a project

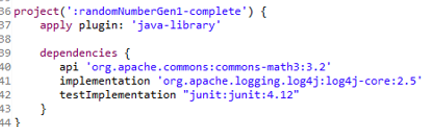
For transitive dependencies the project that is importing the other project wont have access to the transitive dependencies, so it needs to add aditional changes for them to be able to access the dependencies requiered by the project it is importing as a dependency



Plusplus-complete project has a depenency on randomNumberGen1-complete, but it wouldn’t have access to the dependencies that project needs if we use the normal way to get dependencies:



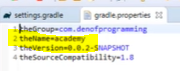
It needs to specify that commons-math is needed in the other project for it to work with the api instead of implementation scope



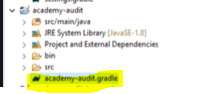
It needs the plugin to be changed from java to java-library in order to work since it contains the scope api

With this the consumer of the library will have access to the transitive dependencies

Using properties file gradle.properties



Rename build.gradle for subprojects (to avoid confusion)

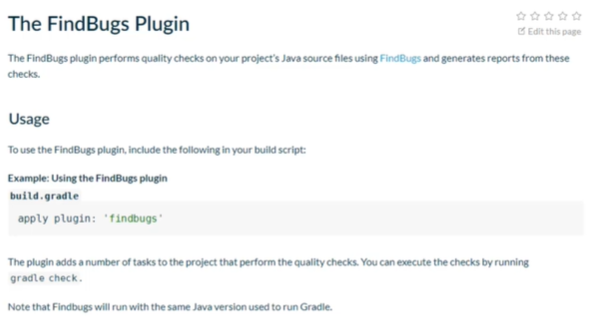


When taking a look to dependencies the dependecies task in help folder is helpful.

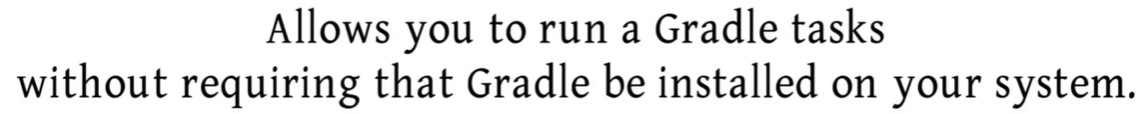
Take a look to the scope you are looking for, ex: implementation

If there seem to be dependencies extra, take a look to both the subproject gradle.build and the root project.build which might have those dependencies configured

FindBugs plugin



Gradle Wrapper



Project could be executed on a git download without the need to download gradle