Ignazio Gallo - PDM



Gradle





Install Gradle

- Prerequisites: requires only a Java Development Kit version 8 or higher to run.
- To check, run java -version

If all you want to do is run an **existing Gradle build**, then you don't need to install Gradle if the build has a Gradle Wrapper

- To install standalone Gradle: https://gradle.org/install/
- USAGE: gradle [option...] [task...]
- \$ gradle -v
- \$ gradle -h



Gradle

 Gradle is an open-source build automation tool focused on flexibility and performance.

Domain Specific Language

Gradle build scripts are written using a Groovy or Kotlin DSL.



Flexibility

Full control

Chaining of targets



Dependency management



Convention over configuration

Multimodule projects

Extensibility via plugins



Groovy DSL on top of Ant





Why use a build tool?

```
javac -cp /usr/local/hadoop-core-1.2.1.jar:/usr/l
ocal/hadoop/lib/commons-cli-1.2.jar:/home/user/program/libs/*.jar *.java
```

jar cmf myManifestFile myJarFile *.class

OPTIONS

-bootclasspath bootclasspath

Cross-compile against the specified set of boot classes. As with the user class

-classpath classpath

Set the user class path, overriding the user class path in the CLASSPATH env 1.2 Software under Solaris for more details.

If the -sourcepath option is not specified, the user class path is searched for sa

-d directory

Set the destination directory for class files. If a class is part of a package, java class is called con.mypackage.NyClass, then the class file is called c:/mycla

If -d is not specified, javac puts the class file in the same directory as the sour

Note that the directory specified by -d is not automatically added to your user

-depend

Use dependency information in a class file's constant table to determine if the drastically.



Apache ANT

- The first widely build tool of the Java world
- Ant is extremely flexible and does not impose coding conventions or directory layouts to the Java projects

```
cproject name="A2" default="compile" basedir=".">
     cproperty name="src" location="$(basedir)"/>
     cproperty name="classes" location="output"/>
     cproperty name="cupJar" location="C:\PROJ131&\A2\cupJar.jar"/>
     <description>
         Build File for Compiler Project
     </description>
     <target name="generate parser">
         <java jar="$(cupJar)" input="parser11.cup" fork="true" failonerror="true">
         </ri>
       </target>
     <target name="compile" depends="generate parser">
             <javac destdir="$(classes)" srcdir="$(src)" classpath="$(cupJar)"/>
         </target>
 </project>
```





Apache ANT cons

- Too flexible
- Write a lot of thing to do a simple build
- The projects have no a standard structure, everyone create their own structure

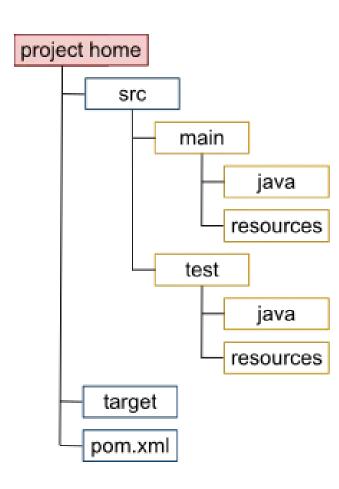




Apache Maven

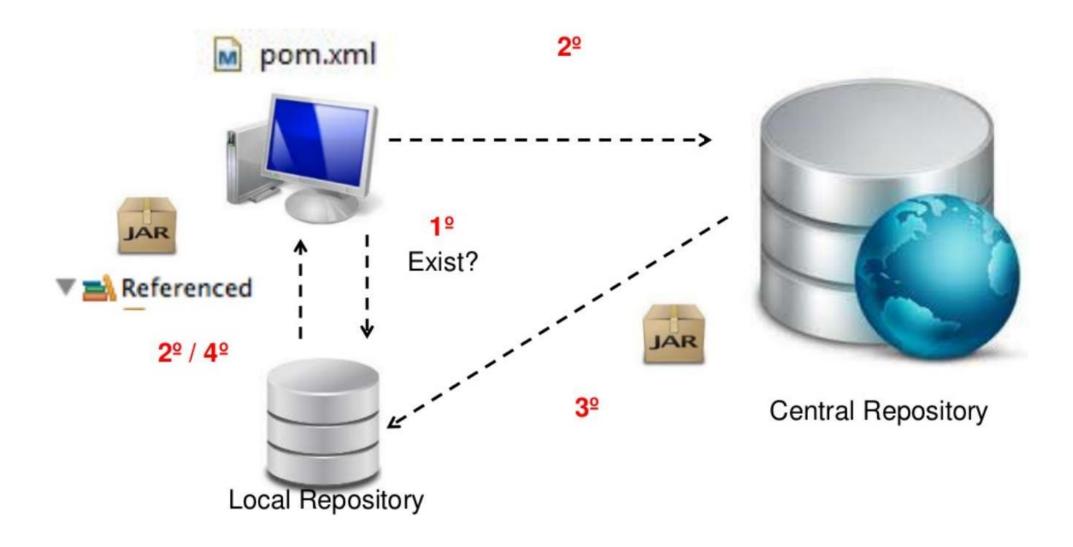
- Simple project setup that follows best practices
 - get a new project or module started in seconds
- Convention over Configuration
- Superior dependency management including automatic updating, dependency closures (also known as transitive dependencies)







Apache Maven Central Repository





Apache Maven cons

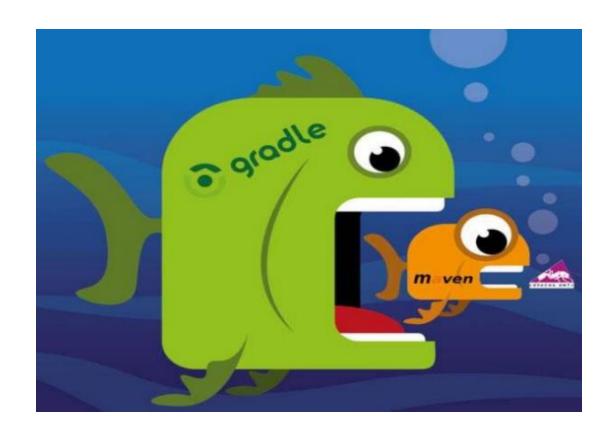
- Too hard / rigid
- Use of XML in the build file
- Maven works great for 90% of the most common things, but complicates a lot for those 10% of specifics things of your project





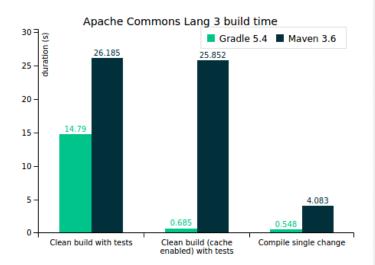
Why Gradle?

- Gradle combines
 good parts of both tools and
 builds on top of them with
 DSL and other improvements
- It has Ant's power and flexibility
- Maven's life-cycle and ease of use





Why Gradle?



```
eland waranapers, drive:
springed comestateple? setasits to at the setter. To
  "property home "wee" thintings arrimate/lave"/"
  oproparty tuans aroland lacations are tenteless.
  sproperty eases'suite" location building
  oproperty wasenights lacations $ Doring (1987)
  "property easen"corning" aglass"1 0-588958631 /c-
  math tip "clavinath compliate.
   opath, 1987/Catagam, beat's
   spottelement lacarines to harpore t-4.8.3. per co.
    spottstement lacarines (Smarrowness-lang-2-3, party
    spathelement lacations #(confact) //
    spottalament lacations $25x110)/classes co-
    spatialsment lacations $/helldp/rest-clauses'/s
  Starget bases table
   capate where bisocherize Lawrence.
   reports garwite parties / rest-classes 7/2.
  charget waser countle? muscular built's
   "DESERT STEELST" SCHOOL" SERVESTON SERVICES CVIANORS"
     milespath reforminespath results to
   TRABBLES.
  charged camer teerCamelle' deposite compile's
   Claves erestre Resortants sentete Tomolograms
      estatepath retractionapath.test?rv
 triangett.
  crarget camer'test' suproper'testion/flat-
   "just fark" yet" baltoufallure" yes":
     Chatchings factor's as '-
       efficient specification.
         Timble committee and a province
      SCHOOLSTALLS.
     sclasspitt: crttp: classpitt.tast !!
     Starmanter Tapertulators
    HATGETAN
  transport.
  "terpet name"High" magnific tact"
    CARGOS FIRST $1 $145777
   "jut jarfilar"$18101)/cnc-comparison(8)
      EXPERIMENTAL PROPERTY - PROPERTY - PROPERTY -
  CTATES CARRY CLASSIC.
   vanished attentionation
  SCENERALLY.
erecathete.
```

```
about serviner's ar mountage full-arre-
COTESTS OBSASE TITE : //AArro. apachs.org/FOM/W. #. 81
units.scie?http://www.wf.urg/7901/3945cbomp-
      Instante'
est schenylogstlate bits://moren.sparks.org/99874.E.
Title: c/movero.upache.org/manno-vit.d.m.und'+
  smodelywrstanist W. Recondalywrstans
  ograsplæggiærgrasplæ-
  cartifactlascoi-campariamnyantifactlas
  sparkagingspare/packagings
  THE PRINCIPLE BUSY APPROPRIESTS
  CREDENSING FREE
   * departments*
      egrouplatesumann-lange/grouplat.
      martifactExcussors-longeraritheride
      restaluable Stivarature
    *Chapmonia can
    -dypolomery)
      *groupf#+takstergroupld*
      PARTYFACTIVE STANDARD TO ANY CONTRACTION
      rescaluant R. Screen lane
      TREADWINEST COLUMN
    A Chesimodesic year
  steranding teas
  Chathan
    "Minghing
      SPINGIAN.
        "group litting assets seems play harry maple."
       tartifactionsen.complier.
       Wilnesser rettfaction
        exantiquest une
          TRANSCORD SCIENCES
         stargets. Exchargesy
        -Countigacations
      Alphagree
    ACRIMETARS.
  0740 This
 CERTIFICATION.
   maven
```

```
Bewjack I
 Assert/Accton "4-6-6"
 ATTIFACTIO COVE-CAMPACINAC
 groupld neld"
 section 1.6-DAPENT
 reperdenting 4
  dependency ("annuous lung romanos-lung I & )
  degendancy to hint think the A. A. A. L.
 Butha E
  BLAGIAN C
    Blugto 1
      groupld 'org.namin meses plagtan'
      artifactid massroomgive plugic
      CHATTEGRATISE I
        ASSESS LA
        target 15.5"
     Polyglot
   maven
```



Convention

http://kaczanowscy.pl/tomek

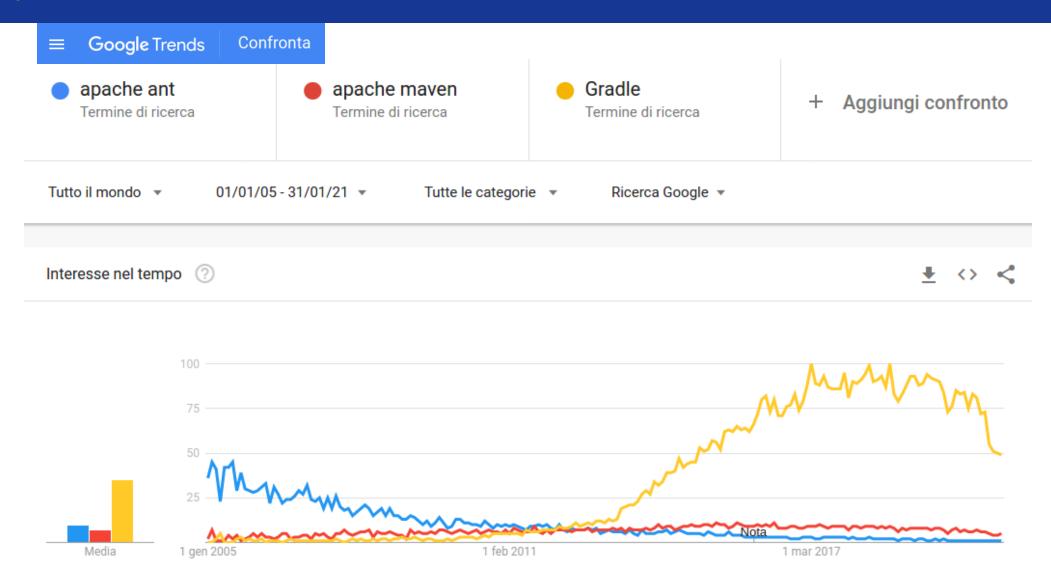


Gradle scripts, but...

```
def toCamelCase(String string) {
   String result = ""
                                                                                complex
   string.findAll("[^\\W]+") { String word ->
        result += word.capitalize()
   return result
afterEvaluate { project ->
   Configuration runtimeConfiguration = project.configurations.getByName('compile')
   ResolutionResult resolution = runtimeConfiguration.incoming.resolutionResult
   // Forces resolve of configuration
   ModuleVersionIdentifier module = resolution.getAllComponents().find { it.moduleVersion.name.equals("pl
   String prepareTaskName = "prepareS{toCamelCase("S{module.group} S{module.name} S{module.version}")}Lib
   File playServiceRootFolder = project.tasks.find { it.name.equals(prepareTaskName) }.explodedDir
   Task stripPlayServices = project.tasks.create(name: 'stripPlayServices', group: "Strip") {
        inputs.files new File(playServiceRootFolder, "classes.jar")
       outputs.dir playServiceRootFolder
        description 'Strip useless packages from Google Play Services library to avoid reaching dex limit'
       doLast {
           copy {
                from(file(new File(playServiceRootFolder, "classes.jar")))
                into(file(playServiceRootFolder))
               rename { fileName ->
                    fileName = "classes_orig.jar"
           tasks.create(name: "stripPlayServices" + module.version, type: Jar) {
                destinationDir = playServiceRootFolder
               archiveName = "classes.jar"
               from(zipTree(new File(playServiceRootFolder, "classes_orig.jar"))) {
                    exclude "com/google/ads/**"
                   exclude "com/google/android/gms/analytics/**"
                   exclude "com/google/android/gms/games/**"
                   exclude "com/google/android/gms/plus/**"
                   exclude "com/google/android/gms/drive/**"
                   exclude "com/google/android/gms/ads/**"
           }.execute()
```

```
plugins {
                                                             simple
android {
    compileSdk 32
 💡 defaultConfig ┨
        applicationId "com.example.myapplication"
       minSdk 22
        targetSdk 32
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
   buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt')
   compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
       targetCompatibility JavaVersion.VERSION_1_8
    kotlinOptions {
       jvmTarget = '1.8'
```

Why Gradle?





Who is using Gradle?





















































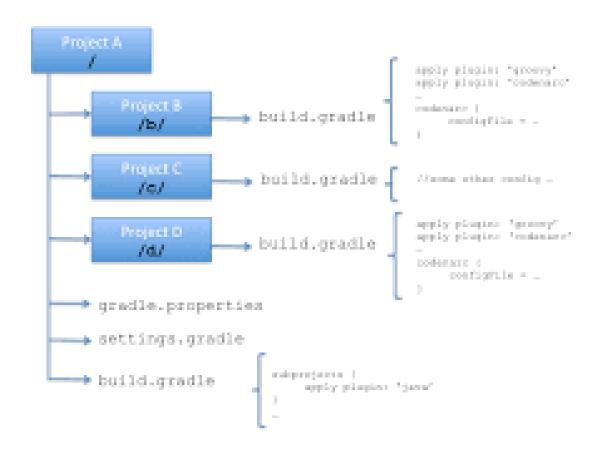






Gradle Introduction

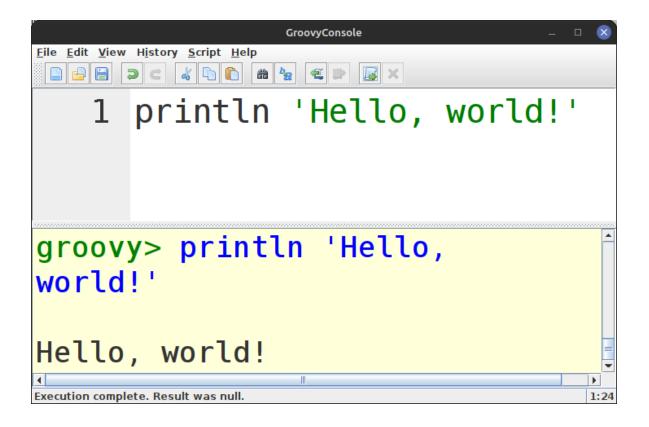
- Gradle is composed of two concepts:
 - **Projects and Tasks**
- A **Project** may represent the creation of a jar or a deploy of an application on the server.
 - Each project is composed of several Tasks
- Tasks represent some atomic job.

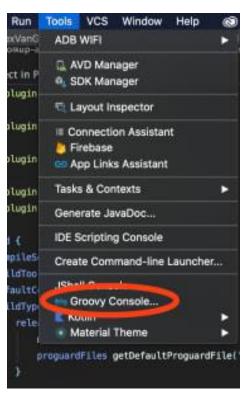




GroovyConsole

- Gradle uses Groovy-based DSL (Domain Specific Language).
- We need a little bit of familiarity with Groovy to work with Gradle.







Groovy

- Gradle uses Groovy-based DSL (Domain Specific Language).
- We need a little bit of familiarity with Groovy to work with Gradle.

Variable Declaration:

```
def str = "hello world" // string
def nameOfList = [] // list
def nameOfMap = [:] // map
```

Loops start.upto(end):

```
1.upto(5) {
    println "$it"
}
```

```
nameOfList.each { item ->
    println element
}
```

class MyClass {
 String str
}
MyClass obj = new MyClass()
obj.str = 'Ciao a tutti!'
println obj.str

Groovy: closures

 A closure in Groovy is an open, <u>anonymous</u>, <u>block of code</u> that can <u>take arguments</u>, <u>return a value</u> and can be <u>assigned to a variable</u>.

example

```
void doSomething(Closure closure) {
  closure.call()
}
doSomething {
  print "DID SOMETHING"
}
```

```
int square(int n) {
  return n*n
}
square = { n ->
     n*n
}
square(5)
```

Your build.gradle file is full of these codes

```
dataBinding {
    enabled = true
}
testOptions {
    animationsDisabled = true
}
```



Groovy: closures

 A closure in Groovy is an open, <u>anonymous</u>, <u>block of code</u> that can <u>take arguments</u>, <u>return a value</u> and can be <u>assigned to a variable</u>.

Qual è la differenza?

```
int x = 5
cSquare = {
   x*x
}
def mSquare(){
   return x*x
}
cSquare()
mSquare()
```

```
void faiQualcosa(Closure cl){
   cl()
}
faiQualcosa {
   println 'Ciao!'
}
```



Groovy: exercises

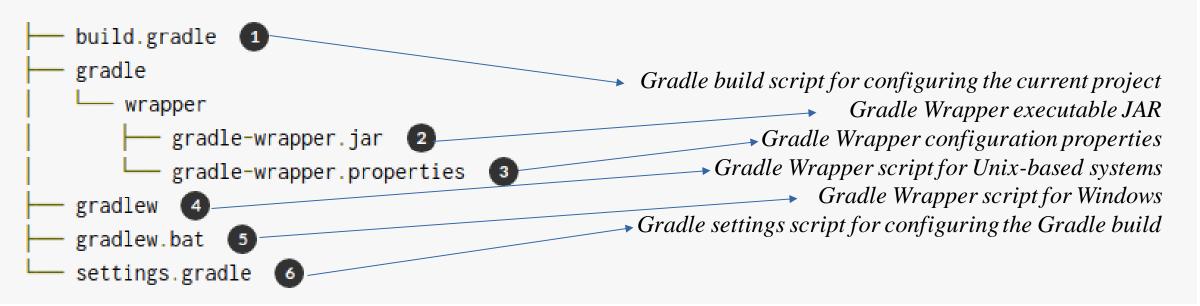
- Proporre una soluzione per gli esercizi su Groovy presenti su elearning.uninsubria.it
- 2. Alcune soluzioni selezionate a caso saranno messe a confronto



Creating New Gradle Builds

Initialize a project

```
$ mkdir basic-demo
$ cd basic-demo
$ gradle init
```





Creating New Gradle Builds

Initialize a project

```
$ mkdir basic-demo
$ cd basic-demo
$ gradle init
```

Provate ad aggiungere la seguente riga al file build.gradle apply plugin: 'java' provate a capire cosa è successo

- eseguendo gradle tasks
- eseguendo gradle build

aggiungere adesso la seguente riga al file build.gradle apply plugin: 'application' provate a capire cosa è successo

- eseguendo gradle tasks
- eseguendo gradle run

Che cosa rappresentano le righe che abbiamo aggiunto? Che differenza c'è tra le seguenti due righe?

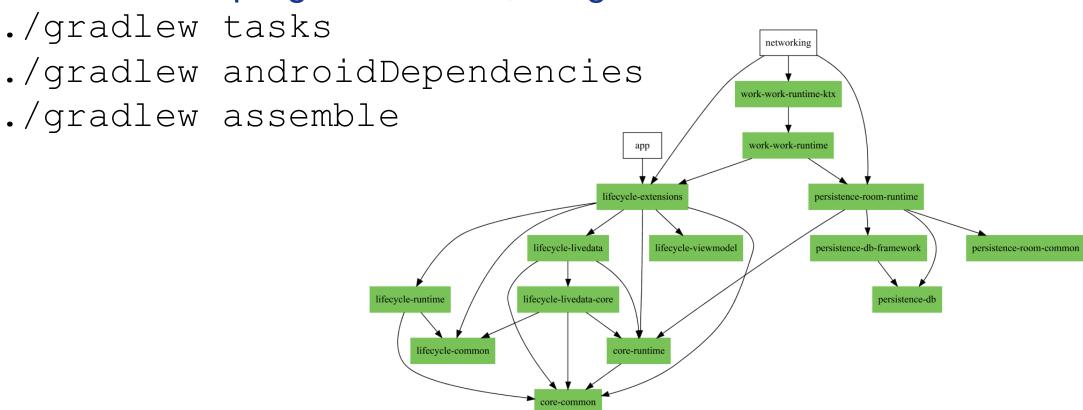
```
apply plugin: 'java'
project.apply([plugin: 'java'])
```

https://docs.gradle.org/current/dsl/org.gradle.api.Project.html



Gradle build.gradle in AndroidStudio

All'interno del progetto Android, eseguire





Create our task

- Dopo aver scritto il nuovo task, eseguire
- ./gradlew hello

```
task hello {
   doLast {
    println 'Hello!'
   }
}
```

```
task respond (dependsOn: hello) {
  doLast {
    println 'Goodbye!'
  }
}
```

./gradlew respond

Che succede se esguiamo:

```
task respond (dependsOn: hello) {
  println 'Configuration'
  doLast {
    println 'GoodBye!'
  }
}
```



Create a task: an example

- We can use the core type called Copy,
 which copies files from one location to another.
- From directory called src.
- Create a directory called dst.
- In your build.gradle file, define a task called copy.

```
task copy(type: Copy, description: "Copies sources to the dest directory") {
  from "src/main/java"
  into "dst"
}
```



Apply a plugin

- Gradle includes a range of plugins, and many, many more are available at the Gradle plugin portal. https://plugins.gradle.org/
- One of the plugins included with the distribution is the base plugin.
- Using this plugin, you can <u>create a zip archive of your project</u> with a configured name and location.

```
plugins {
   id "base"
}

task zip(type: Zip, group: "Archive", description: "Archives sources in a zip file") {
   from "src"
   setArchiveFileName "basic-demo-1.0.zip"
}
```



Gradle tasks

- \$ gradle tasks
- Applying the plugin's to the build file will automatically add set of build task to run
- assemble
 - The task to assemble the output(s) of the project
- check
 - The task to run all the checks.
- build
 - This task does both assemble and check
- clean
 - This task cleans the output of the project



Build Init Plugin

- The Build Init plugin can be used to <u>create a new Gradle build</u>.
- The build type can be specified by using the --type command-line option.
- For example, to create a Java library project run:

```
$ gradle init --type java-library
$ gradle init --type java-application
...
```

https://docs.gradle.org/current/userguide/build_init_plugin.html



Using Gradle on the command line

- Start Gradle build via the command line.
- Here is an overview of the important Android Gradle tasks:

Command	Description
./gradlew build	build project, runs both the assemble and check task
./gradlew clean build	build project complete from scratch
./gradlew clean build	build project complete from scratch
./gradlew test	Run the tests
./gradlew connectedAndroidTest	Run the instrumentation tests

