Gradle

Install Gradle

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

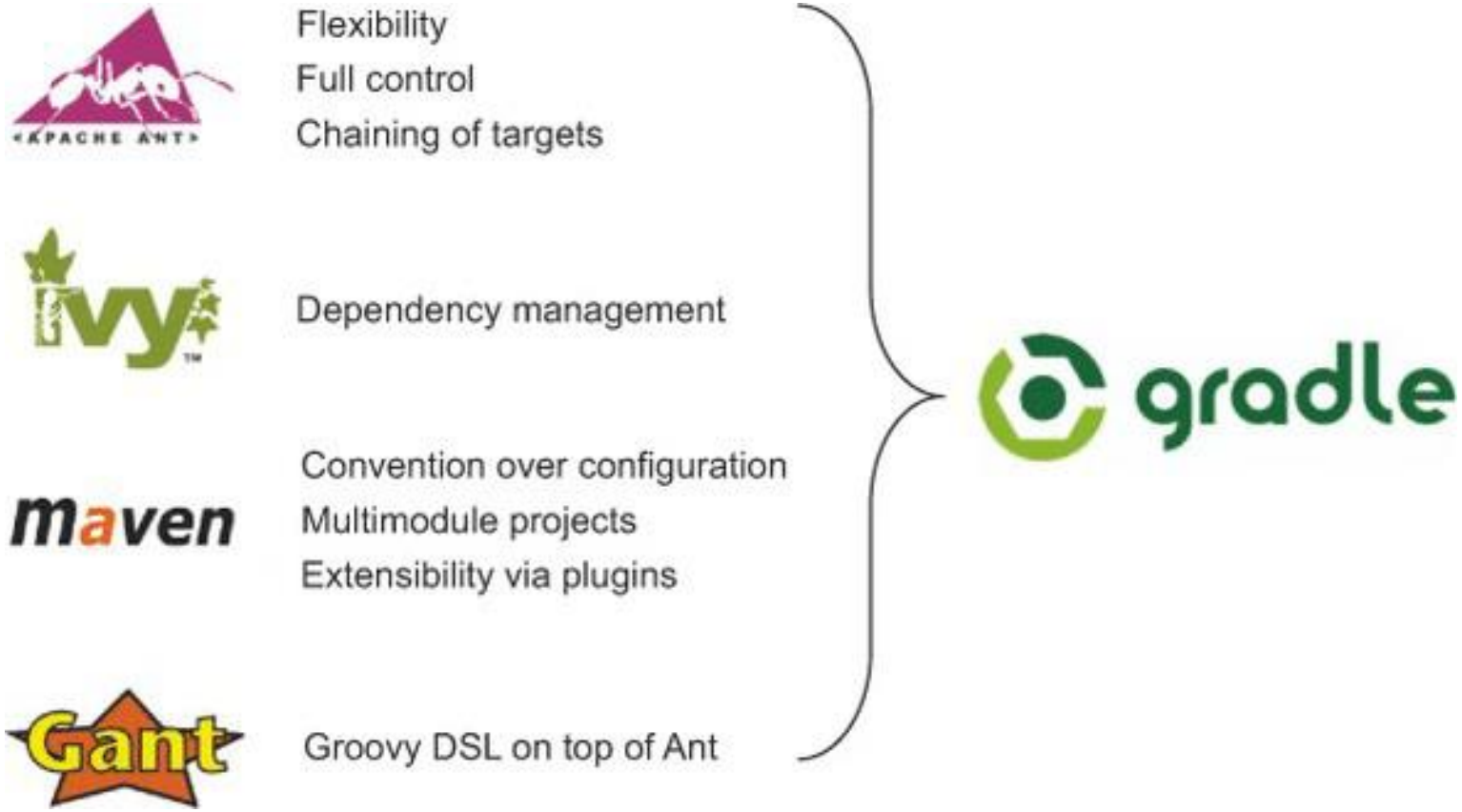
- If all you want to do is run an **existing Gradle build**, then you don't need to install Gradle if the build has a Gradle Wrapper

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

Domain Specific Language



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 so

-d *directory*
Set the destination directory for class files. If a class is part of a package, **java**
class is called `com.mypackage.MyClass`, 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

```
<project name="A2" default="compile" basedir=".">
    <property name="src" location="${basedir}"/>
    <property name="classes" location="output"/>
    <property name="cupJar" location="C:\PROJ131A\A2\cupJar.jar"/>
    <description>
        Build File for Compiler Project
    </description>


    <target name="generate_parser">
        <java jar="${cupJar}" input="parser11.cup" fork="true" failonerror="true">
        </java>
      </target>



    <target name="compile" depends="generate_parser">
            <javac destdir="${classes}" srcdir="${src}" classpath="${cupJar}"/>
        </target>


</project>
```

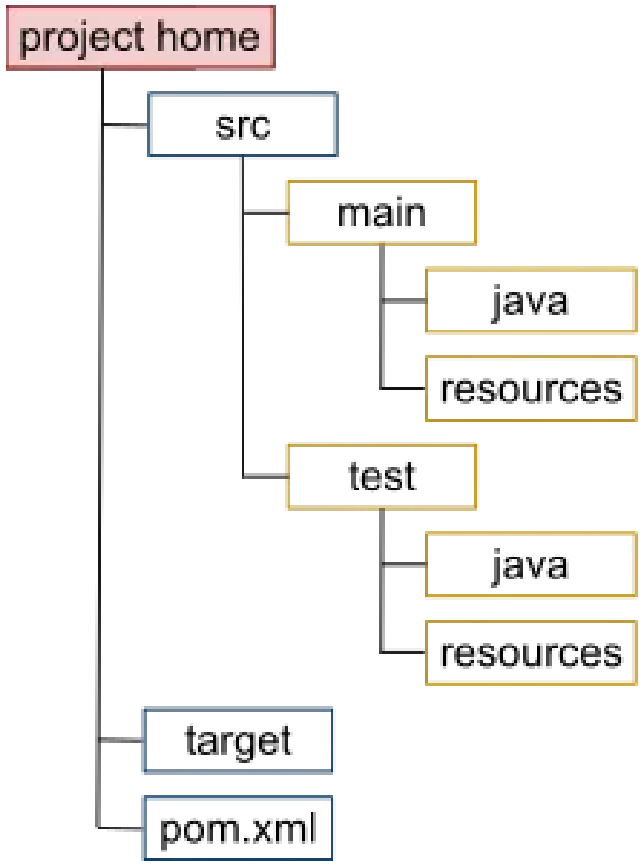
<APACHE ANT>

Apache ANT cons

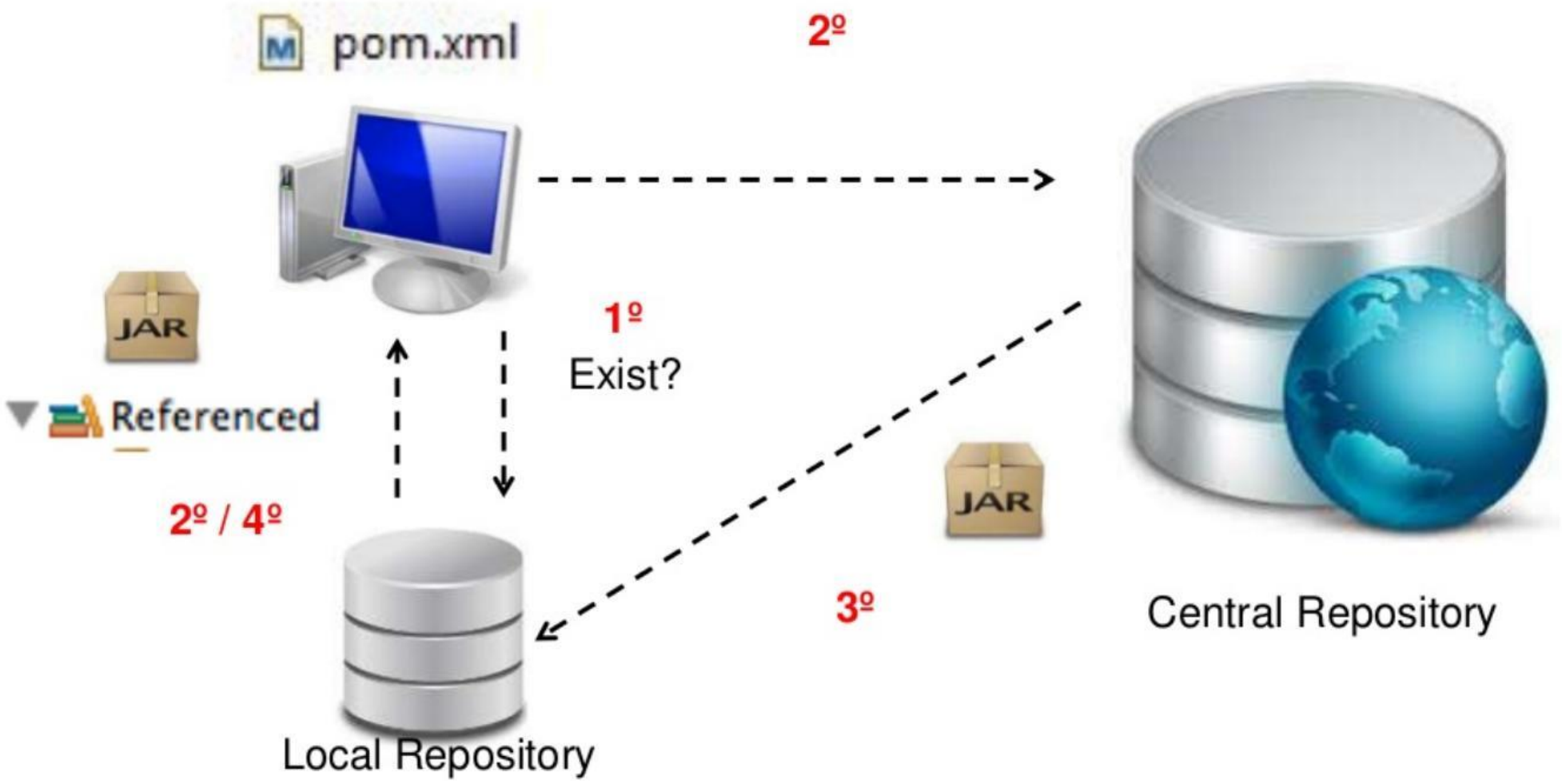
- Too flexible
- Write a lot of thing to do a simple build
- The projects have no a standard structure, everyone create their own structure

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



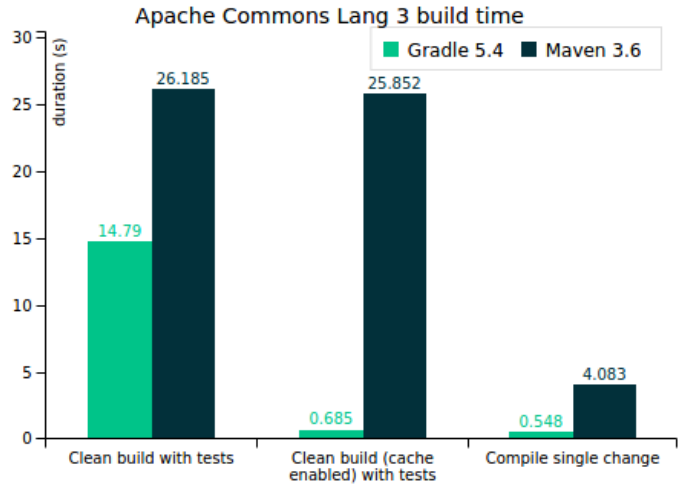
Apache Maven cons

- 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?



Gradle

Polyglot maven

maven

<APACHE ANT>

Convention Over Configuration

Tomek's Blog
http://kaczanowscy.pl/tomek

Gradle scripts, but...

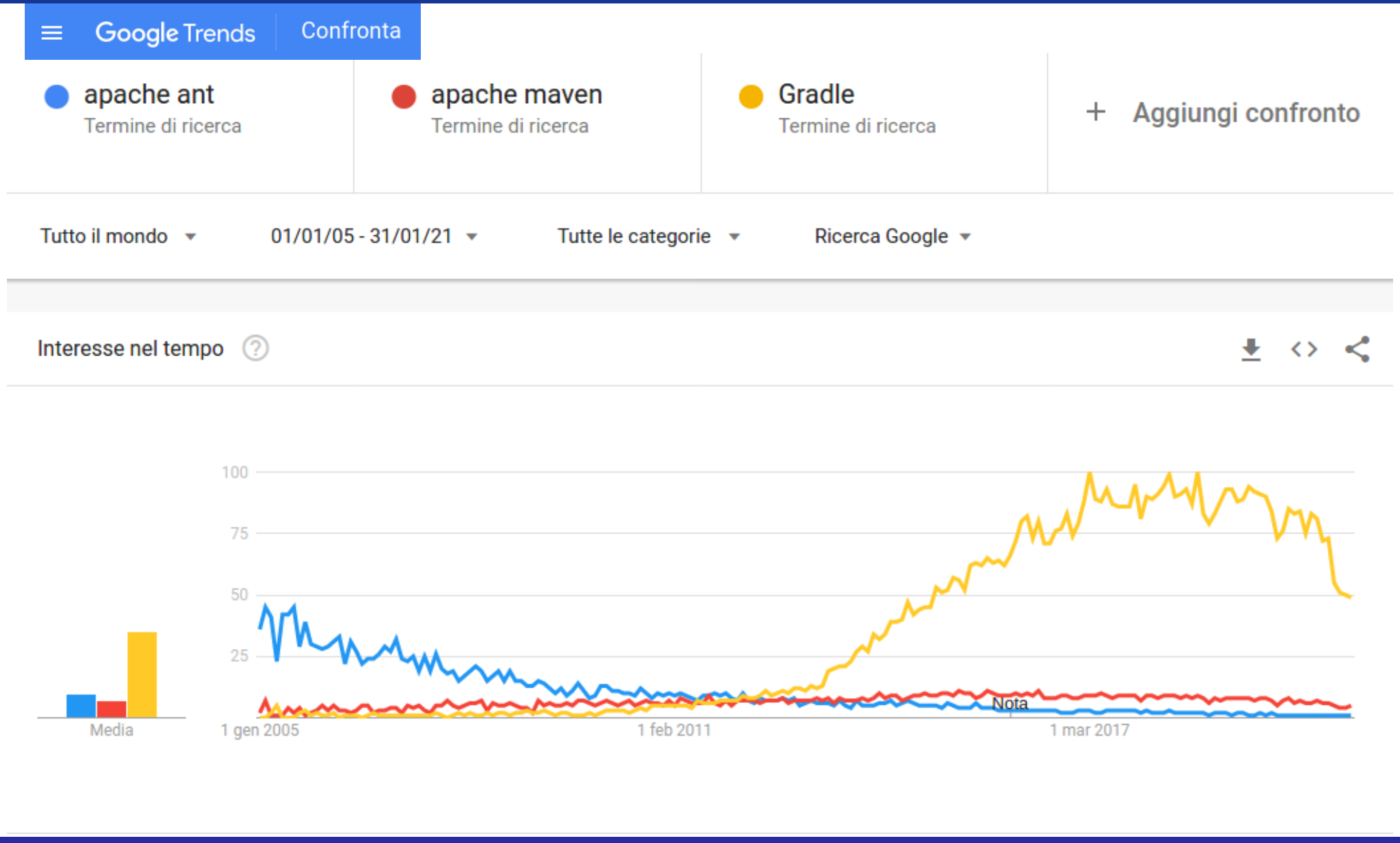
complex

```
def toCamelCase(String string) {
    String result = ""
    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 = "prepare${toCamelCase("${module.group} ${module.name} ${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()
```

simple

```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
}

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'
    }
}
```

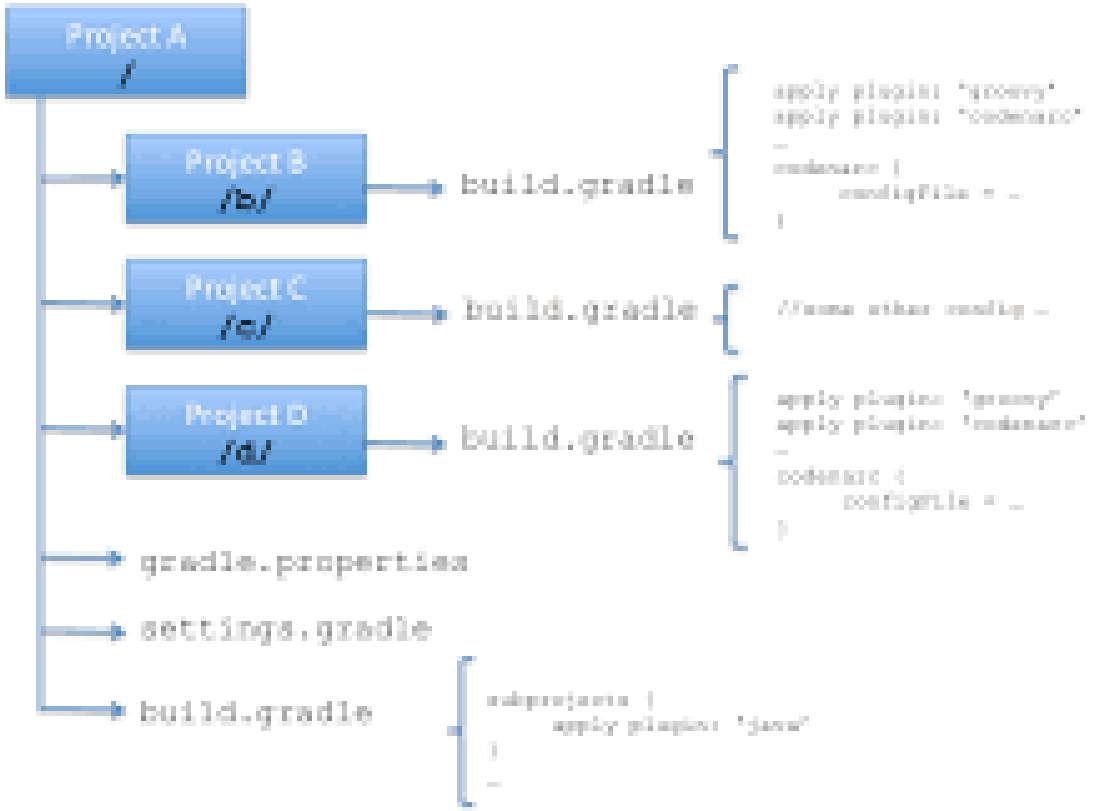
Why Gradle?


Google Trends
Confronta
apache ant
Termine di ricerca
apache maven
Termine di ricerca
Gradle
Termine di ricerca
Aggiungi confronto
Tutto il mondo
01/01/05 - 31/01/21
Tutte le categorie
Ricerca Google
Interesse nel tempo
100
75
50
25
Media
1 gen 2005
1 feb 2011
1 mar 2017
Nota

Who is using Gradle?

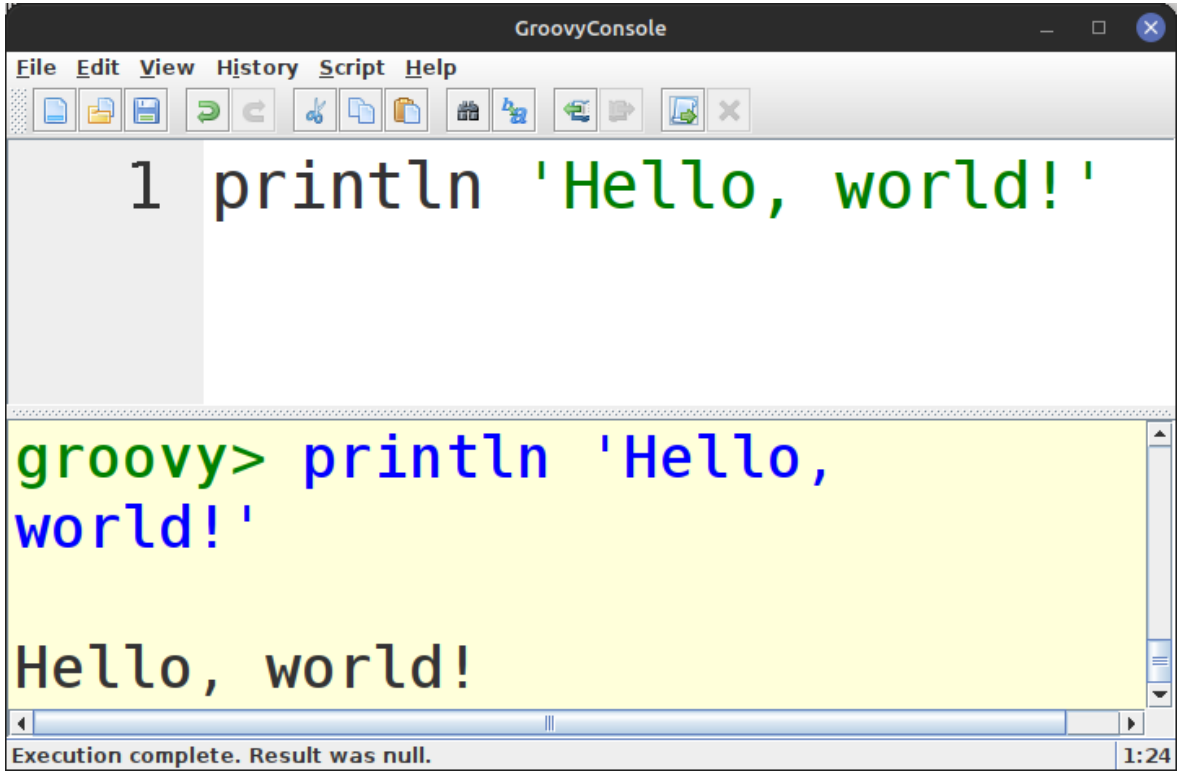
Gradle Introduction

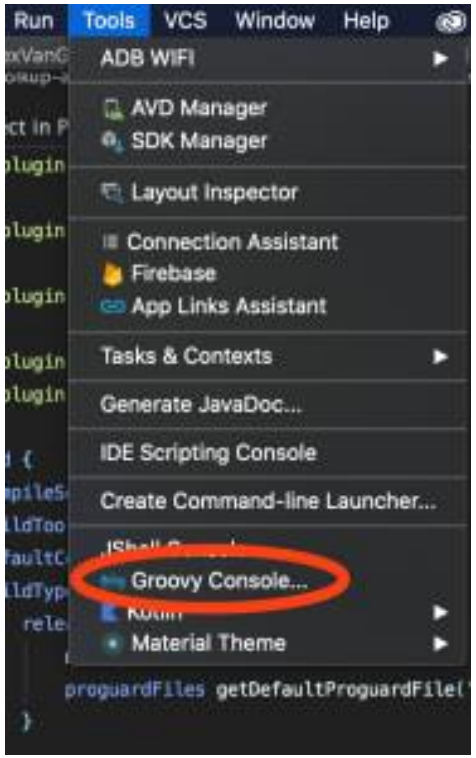
- Gradle is composed of two concepts:
  Projects and Tasks
- A **Project** may represent the creation of a jar or a deploy of an application on the server.
  - Each project is composed of several Tasks
- **Tasks** represent some atomic job.



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
}
```

- **Classes:**

```
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, anonymous, block of code that can take arguments, return a value and can be assigned to a variable.

```
int square(int n) {
   return n*n
}
square(5)
```

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

example

```
void doSomething(Closure closure) {
   closure.call()
}

doSomething {
   print "DID SOMETHING"
}
```

Your build.gradle file is full of these codes

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

Groovy: closures

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

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

1. 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
```

```
├── build.gradle        (1)
├── gradle
│   └── wrapper
│       ├── gradle-wrapper.jar        (2)
│       └── gradle-wrapper.properties (3)
├── gradlew             (4)
├── gradlew.bat         (5)
└── settings.gradle     (6)
```

1. *Gradle build script for configuring the current project*
2. *Gradle Wrapper executable JAR*
3. *Gradle Wrapper configuration properties*
4. *Gradle Wrapper script for Unix-based systems*
5. *Gradle Wrapper script for Windows*
6. *Gradle settings script for configuring the Gradle build*

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?

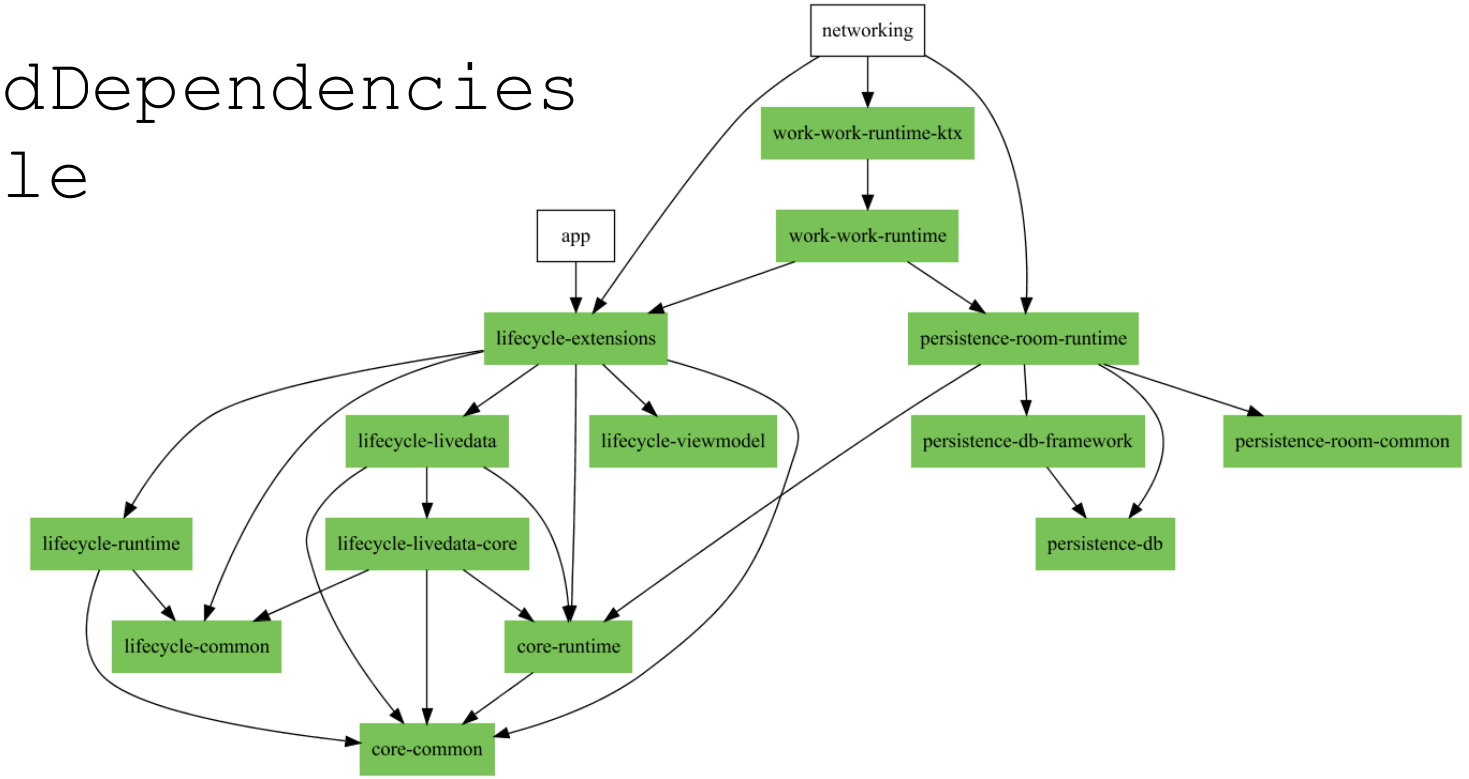
```
apply plugin: 'java'
project.apply([plugin: 'java'])
```

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

Gradle build.gradle in AndroidStudio

- All'interno del progetto Android, eseguire

```
./gradlew tasks
./gradlew androidDependencies
./gradlew assemble
```



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 create a zip archive of your project 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 create a new Gradle build.
- 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