COMP0104 Software Development Practice: GRADLE – Another Build Infrastructure

Jens Krinke

Centre for Research on Evolution, Search & Testing
Software Systems Engineering Group
Department of Computer Science
University College London

Build Automation

- MAKE: Simple, but powerful through recipes, can be adapted to anything
- ANT: Simple, actions implemented in task
- MAVEN: Huge, but limited in configuration, projects adapt to Maven (and not vice versa)
- ANT and MAVEN use XML – hard to read

Gradle:
Take the best of all…

- Convention over configuration
  relies on and supports best practices
- Project based
- Plugins

But

- Conventions can be changed and adapted
- Build scripts are code
- Domain specific language (Groovy based)

Gradle in a nutshell

- Human readable build file, declarative and programmatic.
- Graph of tasks (build steps), DAG
- Task execution:
  - Output of tasks are saved
  - If the output has changed, the dependent tasks are executed.
- Manages dependencies, uses repositories for external dependencies.
- Self-updating

Core Elements of Gradle

- Gradle's system model is defined by a build file using Groovy syntax.
- Gradle project object model:
  projects are represented by Groovy objects.
- Gradle uses Maven's repository and dependence management (actually, Apache Ivy).

How Gradle Scripts Look Like

… like Groovy

```
task helloWorld {
    doLast {
        println 'Hello, World!'
    }
}
```

How Gradle Scripts Look Like
… like Groovy with Dependences

```
defaultTasks 'v

task helloWorld
    doLast {
      println 'He
    }
}

task version (de
    doLast {
       println '(
    }
}
```

```
$ gradle -b hello2.gradle

> Task :helloWorld
Hello, World!

> Task :version
0.0.1-SNAPSHOT

BUILD SUCCESSFUL in 979ms
2 actionable tasks: 2 executed
$ _
```

How Gradle Scripts Look Like
… like Maven

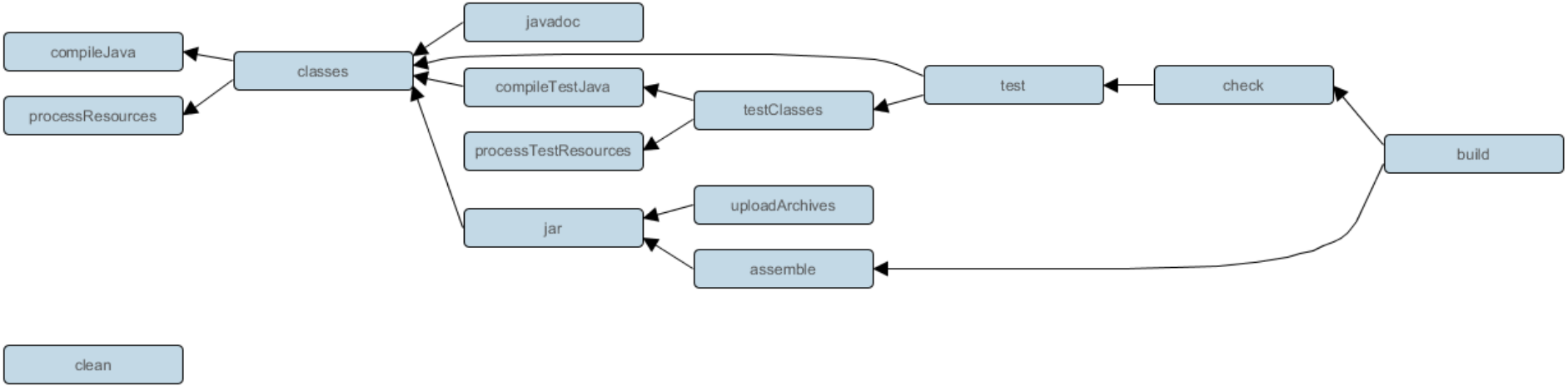
```
apply plugin : 'java-library'

group = "uk.ac.ucl.cs.comp0104.grdlexp"
version = "0.0.1-SNAPSHOT"

repositories {
    mavenCentral()
}

dependencies {
    implementation "joda-time:joda-time:2.10.6"
    testImplementation "junit:junit:4.13"
}
```

Gradle Java Plugin Tasks



Hello.java
src/main/java/hello/

```
package hello;

public class Hello {
   public String hello(String who) {
      return "Hello, " + who + "!";
   }
}
```

HelloWorld.java
src/main/java/hello/

```
package hello;

import org.joda.time.LocalTime;

public class HelloWorld {
  public static void main(String[] args) {
    LocalTime currentTime = new LocalTime();
    System.out.println("The current time is: " +
                       currentTime);
    Hello h = new Hello();
    System.out.println(h.hello("World"));
  }
}
```

$ gradle build

```
BUILD SUCCESSFUL in 1s
2 actionable tasks: 2 executed
```

Concepts

- Take the best from MAKE, ANT, MAVEN, …
- Put it in a Domain Specific Language, …
- You get the power and conventions without the restrictions.