Building Software with Gradle - An Introduction

Etienne Studer
Tooling Lead, Gradleware
etienne.studer@gradleware.com

Donat Csikos
Principal Engineer, Gradleware
donat.csikos@gradleware.com
Gradle

The Enterprise Build System

Highly tested frameworks  Multi-dimensional build variants  Thousands of dependent modules
Poll

Using Gradle?
Build Pain Points

• Build is difficult to read, maintain, and extend
• Weak support to model project dependencies
• Build system dictates project layout
• Slow build performance
• Incompatible build system versions
• Build system restricts project automation
Core Concepts

- Deeply model the domain
- Provide declarative domain-specific language
Core Gradle Objects

- Task
- Project
- Plugin
- Dependency
- Component
Tasks

- Unit of work
- Declarative
- Typed

```groovy
task testClassesJar(type: Jar) {
    from 'build/classes/test'
    classifier = 'test'
    manifest.attributes provider: 'Gradle'
}
```
Task Dependencies

- Directed
- Acyclic

```groovy
assemble.dependsOn testClassesJar
assemble.dependsOn 'sourcesJar'

task sourcesJar(type: Jar) {
    from sourceSets.main.java
    classifier = 'sources'
}
```
Task Graph (DAG)

- Each task to be executed is a node
- The \textit{dependsOn} relations define directed edges
- Each task executed once and only once
- Execution order is against the edge directions
Plugins

- Provide tasks
- Provide task types
- Configure tasks of other plugins
- Register build listeners

apply plugin: 'java'
apply plugin: 'application'
External Plugins

• Can be applied from any local/remote M2 repository

buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'nu.studer:gradle-jooq-plugin:1.0.5'
    }
}

apply plugin: 'nu.studer.jooq'
Projects

• Have plugins applied
• Contain child projects
• Contain tasks
• Contain configurations
Project Dependencies

- Project dependencies and external dep. declared in configurations
  - Configurations provided by plugin or defined in script

```gradle
repositories {
    maven {
        url 'https://my.repo.org'
        credentials {
            username 'user'
            password 'password'
        }
    }
}

dependencies {
    compile project('api')
    testCompile 'junit:junit:4.8.1'
}
```

http://www.gradle.org/docs/current/userguide/dependency_management.html
Demo
Gradle Object Model

- Object graph can be traversed

```groovy
allprojects {
    repositories {
        jcenter()
    }
}

allprojects {
    def zipTasks = tasks.withType(Zip)
    def imgZipTasks = zipTasks.matching { it.name.startsWith('images') }
    imgZipTasks.all {
        classifier = 'IMAGES'
    }
}

task imagesJar(type: Jar) {
    from 'images'
}
```
Implicit Task Dependencies

- Model the input of tasks as the output of other tasks so Gradle can build the correct task graph

```groovy
task docsZip(type: Zip) {
    into('java') {
        from project(':SUB1').javadoc
    }
    into('scala') {
        from scaladoc
    }
}
```
Task uptodate checks

Inputs —> Task —> Outputs

Only run task if its input or output has changed since the previous run

💡 Define inputs and outputs on your custom tasks!

http://www.gradle.org/docs/current/userguide/more_about_tasks.html
Demo
Multi-Project Build

• Gradle project structure defined in *settings.gradle*

• Arbitrary mapping from physical to logical structure

• Partial builds supported
  • Gradle tasks *buildDependents* and *buildNeeded*

http://www.gradle.org/docs/current/userguide/build_lifecycle.html
Dependency Management

- Most powerful dependency management today
- Many options for version conflict resolution

```groovy
repositories {
    jcenter()
    mavenCenter()
    mavenLocal()
    maven { url 'http://myrepo.org' }  
    flatDir { dirs 'lib' }
}

configurations.compile 'com.google.guava:guava:17.0'
```
Android-Specific Language

```java
android {
    compileSdkVersion 21
    buildToolsVersion '21.0.2'
    defaultConfig {
        applicationId 'ch.example.android.foo'
        minSdkVersion 15
        targetSdkVersion 21
    }
    buildTypes {
        release {
            runProguard false
            proguardFiles getDefaultProguardFile('proguard-android.txt')
        }
    }
}
```
Hidden build complexity
Gradle Plugin Portal

http://plugins.gradle.org
Custom plugins

- Reusable build logic
- Plain Java / Groovy project
- Package in .jar file with plugin descriptor
- Upload to repository (JCenter, MavenC, private)

`plugindev` plugin simplifies bundling & publishing

https://github.com/etiennestuder/gradle-plugindev-plugin

http://plugins.gradle.org/submit
Policies

- Enforce company policies through init scripts

```groovy
gradle.taskGraph.whenReady {
    allprojects { Project project ->
        def androidExtension = project.extensions.findByName('android')
        if (androidExtension) {
            def release = androidExtension.buildTypes.find { def buildType ->
                buildType.name == 'release'
            }
            if(release && !release.runProguard){
                def msg = "Build type '" + release.name + '" must run proGuard."
                throw new IllegalStateException(msg)
            }
        }
    }
}
```

http://www.gradle.org/docs/current/userguide/init_scripts.html
Gradle Wrapper

• No setup on CI/Dev machines required
• Different versions of Gradle for different projects
• Gradle version persisted in version control system

http://www.gradle.org/docs/current/userguide/gradle_wrapper.html
Gradle Daemon

- Reuse JVM with same settings between builds
  - ~/.gradle/gradle.properties
    org.gradle.daemon=true
  - GRADLE_OPTS
    -Dorg.gradle.daemon=true
  - Command Line
    gradlew tasks --daemon (--no-daemon)

http://www.gradle.org/docs/current/userguide/gradle_daemon.html
Tooling API

- Proxy for embedding Gradle
- Extension mechanism to provide custom models
- Forward and backward compatible

Client VM with gradle-tooling-api.jar | get build models | Gradle Daemon for Gradle build

invoke build tasks

http://www.gradle.org/docs/current/userguide/embedding.html
Unified Build

• Gradle as the single source of build logic

IDE → Tooling API → Gradle ← Cmd Line

CI Server
Demo
Coming soon

• Parallel task execution
• Configuration phase enhancements
• New JVM language plugins
Summary

Gradle is an extensible build language that targets usability, maintainability, extensibility, and performance for complex builds.
Resources

- Gradle Userguide
  http://www.gradle.org/docs/current/userguide/userguide_single.html

- Gradle Reference Guide
  http://www.gradle.org/docs/current/dsl/

- Gradle Forum
  http://forums.gradle.org/gradle
Conferences

• EclipseCon 2015
  https://www.eclipsecon.org/na2015

• Gradle Summit June 2015
  http://www.gradlesummit.com
Gradleware

• Gradle development
  • Open-Source

• Gradle training
  • Virtual training, On-site training

• Services and consulting
  • Gradle, Automation, Continuous Integration & Delivery

https://www.gradleware.com/
Building Software with Gradle - An Introduction

Etienne Studer
Tooling Lead, Gradleware
etienne.studer@gradleware.com

Donat Csikos
Principal Engineer, Gradleware
donat.csikos@gradleware.com