Gradle: The Basics and Beyond
Ken Kousen, Kousen IT, Inc.
Gradle

The Basics and Beyond
Contact Info

Ken Kousen
Kousen IT, Inc.
ken.kousen@kousenit.com
http://www.kousenit.com
http://kousenit.wordpress.com (blog)
@kenkousen
Videos (available on Safari)

O'Reilly video courses: See Safari Books Online for details

- Groovy Programming Fundamentals
- Practical Groovy Programming
- Mastering Groovy Programming
- Learning Android
- Practical Android
- Gradle Fundamentals
- Gradle for Android
- Spring Framework Essentials
- Advanced Java Development
Modern Java Recipes

Source code:

https://github.com/kousen/java_8_recipes
Installation

Install Java 7 or 8 (9 is supported by Gradle 4.2.1+)

Go to gradle.org and follow install link

https://gradle.org/install/

Download and unzip distribution

Add "bin" directory to your PATH
Gradle Wrapper

Gradle can generate a "wrapper" script

Will download and install Gradle into ~/.gradle/wrapper/dists

> gradle wrapper

--gradle-version <pick a version>

--distribution-type "all" or "bin"
Verify Gradle Distribution

> gradle -v

Shows:

- Gradle version
- Embedded Groovy version (for running Gradle)
- Java version
- OS info
Gradle Build File

Default build file is "build.gradle"

Gradle also uses:

- **settings.gradle** → discussed in multi-project builds
- **gradle.properties** → key/value pairs

Use `-b` command line flag to use different build file
Gradle Daemon

Background process that improves build performance

Enabled by default in Gradle 3.0+

Default TTL is 4 hours

> gradle --status

Shows process IDs and status

> jps (from Java) also works
Build Scans

Need to use build scan plugin (automatic for 4.3+)
Need to add license agreement

plugins {
    id 'com.gradle.build-scan' version '1.13.1'
}

buildScan {
    licenseAgreementUrl = 'https://gradle.com/terms-of-service'
    licenseAgree = 'yes'
}
Build Scans

Add plugin

Add license agreement

Run with \(--\text{scan}\) (3.4 and above) or \(-D\text{scan}\)

Check out results

Self-hosted version part of Gradle Enterprise
Ad-Hoc vs Typed tasks

Ad-hoc:

```groovy
task hello {
    doLast { println 'What up?' }
}
```

Typed:

```groovy
task copy(type: Copy) {
    from 'mydir'
    into 'otherDir'
}
```
Build Lifecycle

Init time

Config time

Execution time
Filesystem Task Types

Copy
Sync
Delete
Jar / Tar / Zip
Project Properties

- name
- buildDir
- logger
- projectDir
- tasks
- version
Task Properties

- name
- dependsOn
- enabled
- group, version
- inputs, outputs
- doFirst(), doLast(), onlyIf(), …
DAG

Gradle builds Directed Acyclic Graph

Graph based on dependencies

- dependsOn
- mustRunAfter
- shouldRunAfter
- finalizedBy

See execution path with -m or --dry-run

Each task executed only once
Command line flags

Exclude task with -x

Continuous build -t

Keep going --continue

Dry run -m or --dry-run

Parallel --parallel (for multi-project builds)
Creating Task Classes

Extend DefaultTask

Use `@org.gradle.api.tasks.TaskAction`
Access Task Graph

```java
gradle.taskGraph.whenReady { graph ->
    if (graph.hasTask(":mytask")) {
        if (!project.hasProperty("myprop")) {
            // do whatever
        }
    }
}

println gradle.taskGraph.allTasks
```
In build.gradle, use logger

```java
logger.info "An info message"
logger.warn 'A warning'
```

> gradle -i hello
Script Plugins

Like an import

apply from: 'other.gradle'

"from" value can be a full URL

Not cached -- if unavailable, build will fail
Binary Plugins

Class that implements `Plugin<Project>`

```java
define apply(Project project) {
    // ...
}
```

```groovy
apply plugin: 'org.foo.my-plugin'
apply plugin: org.foo.plugins.MyPlugin
```
apply plugin: 'org.foo.my-plugin'

META-INF/gradle-plugins/org.foo.my-plugin.properties:
    implementation-class=org.foo.plugins.MyPlugin
Ant integration

Gradle includes "ant" object → AntBuilder

Import Ant build files → Ant tasks become Gradle tasks

    ant.importBuild 'build.xml'
Dependency Resolution

`jcenter()` or `mavenCentral()`

Any Ivy or Maven url

Local file system
A configuration is a collection of dependencies.
Customizing

```
compile "org.hibernate:hibernate:3.1"  // full, transitive

compile('org.hibernate:hibernate:3.1') {
    exclude module: 'cglib'
}

transitive = false → no transitive dependencies

resolutionStrategy.failOnVersionConflict()
configurations.compile {
    resolutionStrategy.force 'org.hibernate:hibernate:3.1'
}
```
Checking dependencies

> gradle dependencies [--configuration <name>]

> gradle dependencyInsight --dependency <name> --configuration <name>
Publishing

Publish to Maven or Ivy repositories

```groovy
apply plugin: 'maven'

uploadArchives {
    repositories {
        mavenDeployer {
            repository(url: 'http://...')
        }
    }
}
```
Extra properties

ext.myprop → avoid conflicting with existing project properties

myprop now available

- throughout project
- in any subproject
Inputs and Outputs

files, directories, properties

In task class, annotate:

- `@InputFile, @InputFiles`
- `@InputDirectory`
- `@Input (for a String property)`
- `@OutputFile`
- `@OutputDirectory`
Inputs and Outputs

In the build file,

```java
task mytask {
    inputs.file file('myfile.txt')
    outputs.file file('output.txt')
    doLast {
        // ... whatever ...
    }
}
```
Inputs and Outputs

Task is **up to date** when:

- Inputs haven't changed
- Outputs still present and unchanged

**Input/Output files are hashed**

**Contents of directories are hashed**

**Values of properties are serialized**
IDE

Eclipse → **Buildship**

IntelliJ → Gradle support built-in

```groovy
apply plugin: 'eclipse' or apply plugin: 'eclipse-wtp'
apply plugin: 'idea'
```
Plugin Repository

Gradle Plugins Repository

http://plugins.gradle.org
Thank you