

GRADLE

The New Build Automation
Tool on the Block

www.gradle.org



What is Gradle?

- A general purpose build system
- Utilizes a Groovy DSL (Domain Specific Language) with a Java-based core
- Provides built-in support for Java, Groovy, and Scala builds
- Leverages selective capabilities of Ant, Ivy and Maven
- Useful for many more things than just Grails builds 😊

Why Groovy?

- ⦿ Because it isn't XML!
- ⦿ Creating DSLs with a dynamic language containing meta-programming support is easier
- ⦿ Groovy, Python, and Ruby would all be viable options, but Groovy supports the JVM ecosystem best
- ⦿ Did I mention that it beats XML for anything other than just configuration?

Groovy variances from Java

- Automatic Imports
 - `java.lang.*`, `java.io.*`, `java.util.*`, `java.net.*`
- Optional
 - Semicolons (DSL friendly)
 - Parenthesis (DSL friendly)
 - Return Statements (last line always returned)
 - Typing
 - Exception Handling
- Operator Overloading
- Safe Dereferencing (special null-test operators)
- Closures (Gradle DSL uses extensively)

Build Tool Timelines

- 2000 – Ant Released Standalone
- 2004 – Maven 1.0 Released
- 2005 – Ivy 1.0 Released
- 2005 – Maven 2.0 Released
- 2009 – Ivy 2.0. Released
- 2010 – Maven 3.0 Released
- 2012 – Gradle 1.0 Released

Several years of prior development

Key Differentiators

	Ant + Ivy	Maven	Gradle
CI Tool Plugins	YES	YES	YES
Build by convention		YES	YES
Versioning	YES	YES	YES
Import Ant/Maven Builds	OWN	OWN	BOTH
Built-In Multi-Artifact Builds	YES		YES
Multi-project dependency support			YES
Incremental Builds			YES
Plugin support	YES	YES	YES
In-script build extensions			YES
Custom Test Listeners		LIMITED	YES
Auto-import build file into IDE	YES	YES	LIMITED
Commercial Support		YES	YES

My Favorite Existing Features

- ⦿ Gradle Wrapper
- ⦿ Gradle Daemon
- ⦿ Flexible Dependency Management
- ⦿ Directed Acyclic Graph of Tasks
- ⦿ Ant Integration
- ⦿ Maven Integration
- ⦿ Lazy Collections
- ⦿ Sonar Plugin (among other plugins)
- ⦿ Build Comparison

Incubating/Future Features

- ⦿ Configuration on Demand
- ⦿ Gradle Plugin Portal
- ⦿ Improved Dependency Resolve Rules
- ⦿ Parallel and Distributed Builds
- ⦿ Native Client Support
- ⦿ Enhanced Build Comparison
- ⦿ C++ Plugins
- ⦿ Gradle Deployment Plugin (on Arquillian)

Ant and Maven Pain Points

- ⦿ Difficult to implement algorithms within build file (conditional and looping constructs are not achieved in a natural way)
- ⦿ Simple extension beyond existing functions/plugins limited without new plugin
- ⦿ Ant does not support “build by convention”
- ⦿ Maven penalizes attempts to diverge from “build by convention”
- ⦿ Multi-project (component) support is limited

Integration with Ant



- Ant is treated as a first class citizen
- Any Ant task can be used from within Gradle
 - `ant.delete dir: 'someDir'`
- Gradle ships with Ant
 - Usually the latest version available
- Gradle can import almost any Ant build
 - `ant.importBuild 'build.xml'`

Integration with Maven

- Auto-generation of pom.xml
- Deploy to local Maven repository
- Deploy to any remote Maven Repository
- Full maven metadata generation
- Compliant signing plugin for deployment

Build Phases



Initialization

-  Creates project instances
-  Determines single versus multi-project build via settings.gradle file (or absence of one)

Configuration

-  build.gradle file
-  DAG (Directed Acyclic Graph) of task dependencies created

Execution

-  build.gradle file
-  Tasks Executed

Tasks

- The logical unit of control for Gradle is called a task (similar to Ant targets)
- Task details processed in two separate phases, configuration and execution
- When importing an Ant build, those targets are transformed into full Gradle tasks that can then be augmented normally

How to define tasks

• The original way to create tasks

- `project.tasks.add('aTask').doLast({ println 'Verbose!' })`

• Gradle has its own DSL via groovy to make this nicer

- `task aTask << { println 'Less Verbose' }`
- `task aTask { doLast { println 'Less Verbose 2' } }`
- `task([:], 'aTask') { doLast { println 'Different' } }`
- `task aTask`
- `aTask.doLast { println 'Less Verbose 3' }`

• In groovy, methods do not require parens, and if the last argument is a closure it can be used outside of the parens anyway

Task Examples

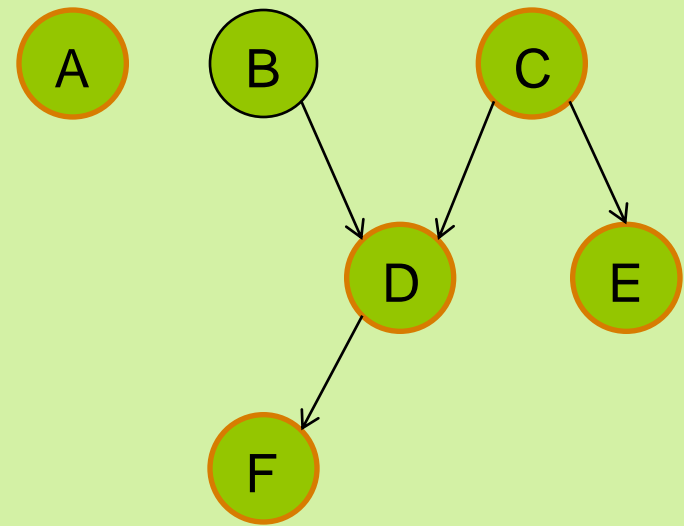
- task myTask { println 'Config phase' }
 - myTask.doLast { println 'Middle Exec' }
 - myTask << { println 'Last Exec' }
 - myTask.doFirst { println 'First Exec' }
 - task shortCutTask << { println 'Exec phase' }
-
- doFirst and doLast simply add an action block to the begin or end of a task. << is an alias for doLast

Task Execution

- Only required tasks are invoked during execution
 - Individual tasks are only ever executed once
- Independent tasks are run in arbitrary order
 - Currently it will choose based upon alphabetical, but do not depend upon this, as this will change in the future when tasks are allowed to run in parallel
- Tasks listed on the command line will be executed in order specified, but any dependent tasks will follow dependency ordering, and individual tasks will still only ever be invoked once

Task Execution

- gradle A C
 - A, F, D, E, C (current behavior)
 - A, F, E, D, C (allowed)
 - A, E, F, D, C (allowed)
- gradle D C A
 - F, D, E, C, A
- Tasks are only run once, and other than dependencies, order is irrelevant unless tasks explicitly listed upon command line.



Task Inputs/Outputs

- Should a clean be required before every build, or at least every build you trust to be correct?
 - Absolutely not!
- Gradle allows tasks to describe:
 - Input Files
 - Input Directories
 - Input Properties
 - Output Files
 - Output Directories

Task Annotations

```
class MyTask extends DefaultTask {  
    @InputFile File text  
    @InputFiles FileCollection path  
    @InputDirectory @Optional File templates  
    @Input String mode    // This is a serialized property  
    @OutputFile File result  
    @OutputDirectory transformedTemplates  
    boolean verbose        // Not serialized property  
  
    @TaskAction  
    generate() { ... }  
}
```

Task Processing

- Property Processing:
 - Exception if input files or directories do not exist
 - Validation disabled with `@Optional` annotation
 - Output directories are created before execution phase
- Gradle's Incremental build algorithm:
 - Hashes of input/output files are cached
 - Hashes for all files of input/output directories are cached.
 - Property values are cached (via serialization)
 - Cache values == Current values → Skip Task

Incremental Builds

- Not confused by network filesystem time issues
 - File modified in the future complaints anyone?
- Can quickly skip tasks that do not actually need to be run
 - Seeing UP-TO-DATE can feel good!
 - Recompile doesn't mean jar needs to be built
- Changes to key properties honored
 - Project version number change will rebuild jar

Tasks that always exist

- gradle help
 - Displays a help message
- gradle dependencies
 - Displays the dependencies of projects
- gradle dependencyInsight
 - Displays details of a specific dependency
- gradle projects
 - Displays root and subprojects
- gradle properties
 - Displays the properties of the current project
- gradle tasks
 - Displays the tasks for the current project

Cool Task Matching

- gradle t
 - Maps to gradle tasks if no other task starting with “t”
- gradle sCT
 - Maps to “superCoolTask” if it is the only task that matches that camel case
- gradle dep
 - Maps to dependencies task if no other task staring with “dep” without any capital characters, so no conflict with dependencyInsight

Plugins

- Two types of plugins
 - Build Script
 - Can be applied from almost anywhere
 - apply from: 'myplugin.gradle'
 - apply from: 'http://myurl/myplugin.gradle'
 - Binary Class implementing `org.gradle.api.Plugin`
 - Must be found in build script classpath
 - Can have id's via meta properties in jar
 - apply plugin: `org.myplugin.MyPlugin`
 - apply plugin: 'myplugin'

Core Plugins

Plugin	Description
java, groovy, scala	Most prominent JVM languages
antlr	Support for generating parsers with ANTLR
pmd, findbugs, checkstyle, jdepend	Code quality plugins, including codenarc for groovy code
eclipse, idea	Import gradle build into respective IDE
project-report	Detailed report information about gradle build
sonar	Integration with Sonar code quality tool
announce	Publish messages to things like twitter and growl
application	Run and bundle a project as an application (java/groovy)
maven	Deploy artifacts to Maven repositories
jetty	Deploy web applications within embedded jetty
osgi	Support for building OSGi bundles
war, ear	Assembling web/j2ee applications

Incubating Plugins

Plugin	Description
cpp	c++ source compilation
cpp-exec	c++ executable compilation
cpp-lib	c++ library compilation
java-library-distribution	Build a zip distribution for a java library
ivy-publish	Provides new DSL for publishing artifacts to Ivy repositories
maven-publish	Provides new DSL for publishing artifacts to Maven repositories
maven2Gradle	Adds support for converting an existing Maven build into a Gradle build

Repositories Support

- ④ Maven
- ④ Maven cache
- ④ Ivy
- ④ Flat directory

Repositories

Dependency Type	Description
Maven Central	A pre-configured repository that looks for dependencies in Maven Central.
Maven Local	A pre-configured repository that looks for dependencies in the local Maven repository (the .m2 cache)
Maven	A Maven repository located somewhere on the local filesystem or at a remote location.
Ivy	An Ivy repository located on the local filesystem or at a remote location.
Flat directory	A simple repository on the local filesystem. Does not support any meta-data formats (no .pom or .ivy files)

Repository Examples

```
repositories {  
    mavenCentral()  
    mavenLocal()  
    maven { url 'http://repo.mycompany.com/maven2' }  
  
    flatDir { dirs 'lib' }  
    flatDir { dirs 'lib1', 'lib2' }  
  
    ivy {  
        url 'http://repo.mycompany.com/repo'  
        layout 'maven'  
    }  
    ivy {  
        artifactPattern "$projectDir/repo/[organization]/[module]-  
[revision].[ext]"  
    }  
}
```

Dependencies Support

- Maven/Ivy Repositories
- Specified directly by files/path
- Project dependencies in multi-project build
- Artifacts produced from other tasks
- Supports transitive and non-transitive dependencies on a per-dependency basis

Dependencies

Dependency Type	Description
External module	A dependency on an external module in some repository.
Project	A dependency on another project in the same build.
File	A dependency on a set of files on the local filesystem.
Client module	A dependency on an external module, where the artifacts exists in a repository, but the module meta-data is specified by the local build.
Gradle API	A dependency on the API of the current Gradle version. Use for plugin development.
Local Groovy	A dependency on the Groovy version used by the current Gradle Version. Used for plugin development.

Mapped to Configurations

- ④ Dependencies are assigned to different configurations
- ④ Configurations are added by various plugins
- ④ Similar in concept to Maven Dependency Scopes, but more generalized

Added by Java Plugin

Configuration Name	Description
compile	Compile time dependencies.
runtime	Runtime dependencies. Extends compile.
testCompile	Additional dependencies for compiling tests. Extends compile.
testRuntime	Additional dependencies for running tests. Extends testCompile and runtime.
archives	Added by 'base' plugin, which is applied by 'java-base' which 'java' plugin extends from. This is artifacts produced, which for java plugin are jars.
default	Added by 'base' plugin like archives. Extends the runtime and contains the artifacts being generated just like the archives configuration. Used as the default configuration for a project dependency when not explicitly named.

External Module Dependencies

```
dependencies {
    runtime group: 'org', name: 'module', version: '2.5'
    runtime 'org:module:2.5', 'org:otherModule:4.5'
    runtime( [group: 'org', name: 'module', version: '2.5'],
             [group: 'org', name: 'otherModule', version: '4.5'] )
    runtime('org:special:3.0.5') { transitive = true }
    runtime group: 'org', name: 'special', version: '3.0.5',
            transitive: true
    runtime(group: 'org', name: 'special', version: '3.0.5') {
        transitive = true }

    runtime 'org:groovy:1.5.6@jar' // Artifact only notation
    runtime group: 'org', name: 'groovy', version: '1.5.6', ext: 'jar'

    compile 'org:groovy:1.5.6:jdk14@jar' // Dependency with classifier
    compile group: 'org', name: 'groovy', version: '1.5.6', classifier:
        'jdk14', ext: 'jar'
}
```

Project/File/Special Dependencies

```
dependencies {  
    compile project(':shared')    // default configuration  
    runtime project(path: ':shared', configuration:  
        'default')  
    testCompile project(path: ':shared', configuration:  
        'testExtras')  
    compile project(path: ':other', configuration: 'api')  
  
    runtime files('libs/a.jar', 'libs/b.jar')  
    runtime fileTree(dir: 'libs', include: '*.jar')  
  
    compile gradleApi()  
  
    groovy localGroovy()  
}
```

Gradle Documentation

- User Guide for getting started and demonstrating common usage
- DSL Reference for core types, task types, and build script blocks
- Javadoc and Groovydoc for detailed information about objects and methods
- [Local Gradle Documents](#)

Gradle Collaboration

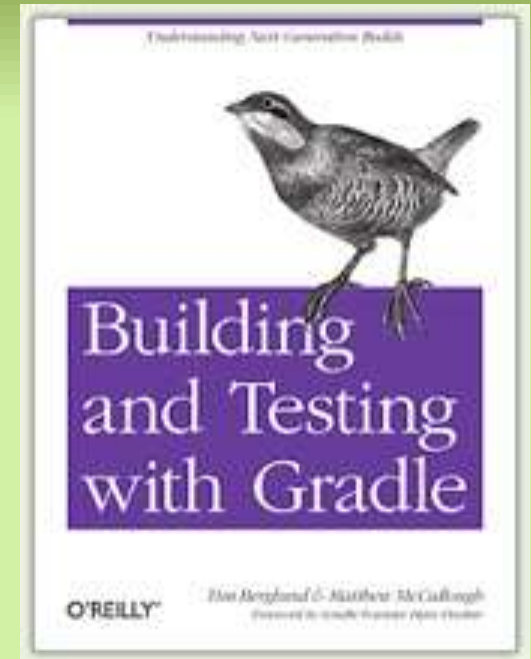
- If you have any type of question related to Gradle, head straight to the forums
 - <http://forums.gradle.org/gradle>
- Github source code repository
 - <https://github.com/gradle/gradle>
- Developer's mailing list
 - dev-subscribe@gradle.codehaus.org
 - Search Nabble or MarkMail for gradle-dev

Gradle Books

- 🌀 Building and Testing with Gradle
- 🌀 Gradle Effective Implementation Guide
- 🌀 Gradle In Action
- 🌀 Gradle DSLs/Programming Gradle

Building and Testing with Gradle

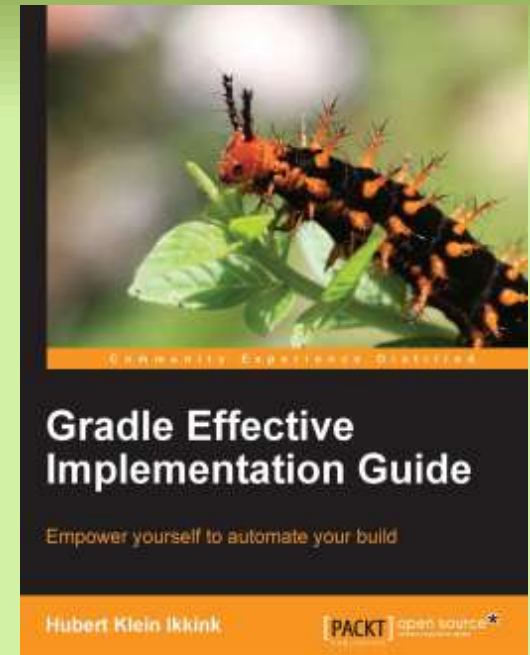
- Released in Mid 2011
- The first book published, and intended to be part of a mini-series of books
- Available for free now from gradleware
- Provides some introductory steps into how to use basic functions of Gradle



www.gradleware.com/registered/books/building-and-testing

Gradle Effective Implementation Guide

- Released in Late 2012
- Excellent supplement to online user's guide
- Part tutorial, part reference guide for practical problems
- Discounts from 2/12 – 2/28
- MREGE25 – 25% off e-copy
- MREGE15 – 15% off print copy



www.packtpub.com/gradle-effective-implementation-guide/book

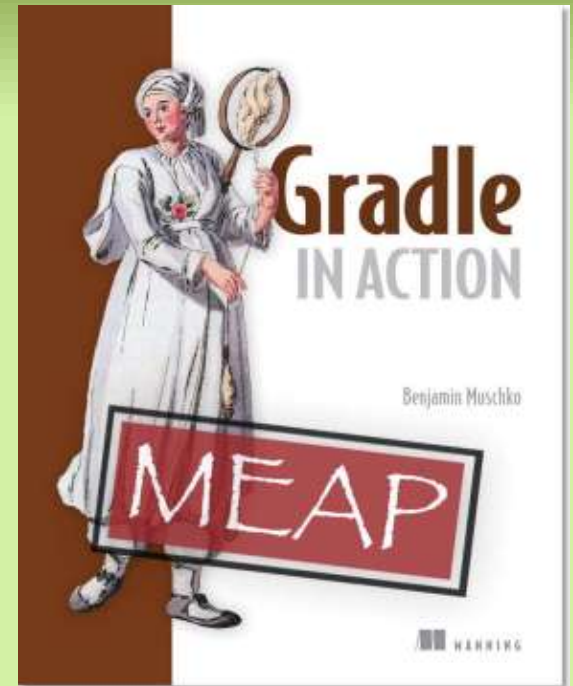
Gradle DSLs/Programming Gradle

- Slated for Mid 2013
- Currently in revision
- Another mini-series book
- Focuses on:
 - File Manipulation
 - Custom Plugins
 - Build Hooks
 - Dependency Management



Gradle In Action

- Slated for Fall 2013
- Preview and ability to provide review feedback available via Manning Early Access Program
- Special guest is author Benjamin Muschko!
- 13grad – 40% discount off books at Manning



www.manning.com/muschko/

Open Source Community



Commercial Support

