



# Gradle

## Leveraging Groovy for Building Java Applications

**Hans Dockter**  
**Gradle Project Lead**  
**mail@dockter.biz**



# About Me

- **Founder and Project Lead of Gradle**
- **Independent Consultant**
- **Trainer for Skills Matter (TTD, Patterns, DDD)**
- **Past**
  - **Senior Developer at Krugle and Volkswagen**
  - **Committer to JBoss and Founder of JBoss-IDE**



# Agenda

## ● **Why a new build system**

- **The problem with Ant & Maven**
- **The importance of Project Automation**
- **Why an internal DSL, why Groovy?**

## ● **Introduction to Gradle**

- **Dependency Based Programming**
- **Build-By-Convention**
- **Managing your Jars**
- **Managing Multi-Project Builds**
- **Organizing your Build Logic**



Why  
a  
new  
build system?



## Our Vision (Quote from Moshé Feldenkrais)

- **Make the impossible possible.**
- **Make the possible easy.**
- **Make the easy elegant.**



The old bulls ...



# Ant



# Ant - Pros

- **Flexible**

- You are in charge.
- Dependency based programming.

- **A wealth of helpful tasks.**



# Ant - Cons

- **No Build-By-Convention**
- **XML**
  - **Anemic Interaction with Java**
  - **Home-Baked general purpose elements**
    - **Steep learning curve**
    - **Pale imitation of a real language**



# Maven



## Maven - Pros

- **Build-By-Convention**
- **Many plugins (although a lot are for self-management)**
- **Some form of transitive dependency management**



# Maven - Cons

## ● XML

- Anemic Interaction with Java
- (Implicit) general purpose elements
- Extremely verbose

● **Maven does not offer dependency based programming.**

● **Frameworkitis ...**



## Frameworkitis ...

... is the disease that a framework wants to do too much for you or it does it in a way that you don't want but you can't change it. It's fun to get all this functionality for free, but it hurts when the free functionality gets in the way. But you are now tied into the framework. To get the desired behavior you start to fight against the framework. And at this point you often start to lose, because it's difficult to bend the framework in a direction it didn't anticipate. Toolkits do not attempt to take control for you and they therefore do not suffer from frameworkitis.

(Erich Gamma)



## Solution

Because the bigger the framework becomes, the greater the chances that it will want to do too much, the bigger the learning curves become, and the more difficult it becomes to maintain it. If you really want to take the risk of doing frameworks, you want to have small and focused frameworks that you can also probably make optional. If you really want to, you can use the framework, but you can also use the toolkit. That's a good position that avoids this frameworkitis problem, where you get really frustrated because you have to use the framework. Ideally I'd like to have a toolbox of smaller frameworks where I can pick and choose, so that I can pay the framework costs as I go. (Erich Gamma)



There  
are  
no  
simple builds.



# Project Automation

- **A build can do far more than just building the jar.**
- **Often repetitive, time consuming, boring stuff is still done manually.**
  - Deployment, release management, generating documentation
- **Manually means: less reliable and less frequently.**
- **Automated means:**
  - Continuous feedback, executable documents, no secret knowledge.
  - Essential for agile projects.
- **Many of those tasks are very company specific.**
- **Maven & Ant are not well suited for this.**



# Gradle Overview 1

- **First release in April 2008**
- **A flexible general purpose build tool**
  - Offers dependency based programming with a rich API.
- **Build-by-convention plugins on top**
- **Powerful multi-project support**
- **Powerful dependency management based on Apache Ivy.**



## Gradle Overview 2

- **Build Scripts are written in Groovy**
  - We get our general purpose elements from a full blown OO language.
  - The perfect base to provide a mix of:
    - Small frameworks, toolsets and dependency based programming.
  - Rich interaction with Java
  - Gradle is NOT a framework.
- **Gradle is mostly written in Java with a Groovy DSL layer on top.**
- **Offers very good documentation (70+ Pages user's guide)**
- **Committer -> Steven Devijver, Hans Dockter, Adam Murdoch, Tom Eyckmans, Russel Winder**



# Live Demo - Hello World



# Live Demo - Java



# Java Plugin - Phases

- **init**
- **resources**
- **compile**
- **testCompile**
- **test**
- **libs**
- **dists**



# Dependencies

- **DSL on top of Apache Ivy**
- **Integrates with Ivy/Maven infrastructure**
- **Support for client modules.**
- **Support for storing libs in SVN.**
- **Extremely powerful and flexible.**
- **Compile only against first level deps.**



# Client Modules

```
addClientModule(conf: ['compile'], id: "org.codehaus.groovy:groovy-all:1.5.4") {  
    addDependency("commons-cli:commons-cli:1.0:jar")  
    addClientModule("org.apache.ant:ant:1.7.0") {  
        addDependency("org.apache.ant:ant-launcher:1.7.0:jar")  
        addDependency("org.apache.ant:ant-junit:1.7.0")  
    }  
}
```

- **No need for XML descriptors.**
- **No need to set up your own repository.**



# Libs in Source Control System

```
addFlatDirResolver(lib, "$rootDir/lib")  
compile ":commons-io:1.4.1"  
testCompile ":junit:junit:4.4"
```

- **Out of the box experience**
- **Perfect match with client modules**



# Multi-Project Builds

- **Configuration Injection**
- **Partial builds**
- **Separate Config/Execution Hierarchy**
- **Arbitrary Multiproject Layout**



## ● ultimateApp

- api
- webservice

```
// build script ultimate app
subprojects {
    manifest.mainAttributes([
        'Implementation-Title': 'Gradle',
        'Implementation-Version': '0.1'
    ])
    dependencies {
        compile "commons-lang:commons-lang:3.1"
        testCompile "junit:junit:4.4"
    }
    sourceCompatibility = 1.5
    targetCompatibility = 1.5
    test {
        fork(dir: $projectDir)
        exclude '**/Abstract*'
    }
}
```



# Organizing Build Logic

- **No unnecessary indirections**
- **If build specific:**
  - Within the script
  - Build Sources
- **Otherwise: Jar**



# Gradle Wrapper

- **Use Gradle without having Gradle installed**
- **Useful for:**
  - CI
  - User friendly



There  
are  
really  
no  
simple builds.



# The Gradle Build

- **Gradle is build with Gradle**
- **Automatic release management**
- **Automatic user's guide generation**
- **Automatic distribution**
- **Behavior depends on task execution graph**



# Release Management

- **The version number is automatically calculated**
- **The dist is build and uploaded to codehaus**
- **For trunk releases, a new svn branch is created.**
- **A tag is created.**
- **A new version properties file is committed.**
- **The download links on the website are updated**



## User's Guide

- **The user's guide is written in LaTeX and generated by our build.ä**
- **The source code examples are mostly real tests and are automatically included.**
- **The expected output of those tests is automatically included.**
- **The tests are run.**
- **The current version is added to the title.**



# Uploading & Execution Graph

- **Based on the task graph we set:**
  - **Upload Destination**
  - **Version Number**



# Production Ready?

- **YES! (If you don't need a missing feature).**
- **The first big enterprise builds are migrating from Maven to Gradle.**
- **We are about to release 0.6.**
- **We are transparent:**
  - The user's guide list all missing features which might be known from other tools.
  - Roadmap.



# Roadmap

- see **<http://www.gradle.org/roadmap>**



# Call for Contribution

- **It is an exciting time for build systems.**
- **A rare opportunity to change and revolutionize a Java base technology.**
- **Use, Give Feedback, Provide Patches, Become a Committer**