Gradle

Standardizing and Administering your
Enterprise Build Environment with Gradle.

About Me

- Luke Daley
- Principal Engineer
- luke.daley@gradleware.com
- Located in London

Slides & Demos

Content can be downloaded from GitHub.

https://github.com/gradleware/gradle-talks/downloads

Download “`gradle-enterprise-admin.zip`”.

Contains:

- Example code
- HTML slides
- PDF slides

Upcoming Webinars

Upcoming Webinars can always be found @ http://gradle.org/webinars.

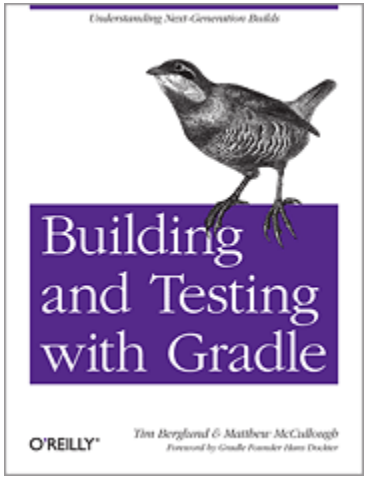
- Migrating and Upgrading to Gradle by Szczepan Faber

Past webinar recordings can always be found at http://gradleware.com/resources.

- The Gradle Roadmap by Hans Dockter
- A Gentle Introduction to Gradle by Tim Berglund
- Gradle Power Features by Szczepan Faber

The Gradle Book

The first of these three books was launched in July of 2011 and is 80 pages of the foundations of Gradle as written by Tim Berglund and Matthew McCullough.



Free HTML version online @ http://gradleware.com/resources.

Webinar Mechanics

Use the question feature of the UI to ask questions.

I'll stop periodically to review the questions.

If you don't get your question answered, email me afterwards.

Background

What are we talking about?

Automation is personal

- Only the simplest of software can be built in near identical ways.
- Convention over configuration let us stop inventing the wheel.
- Rigid conventions inhibit innovation and efficiency.
- Modern automation needs to be ambitious, effective and pervasive.

Whose conventions?

Conventions are great, and Gradle embraces them. However, whose conventions are these?

You might want to extend the conventions.

- e.g. all projects use CheckStyle for static analysis

You might want to replace general conventions with your own.

- e.g. “main” source is meaningless to me

Gradle is a toolkit

Gradle is a collection of tools, like Ant.

Unlike Ant, Gradle also contains tools for standardisation, reuse and conventions.

- Plugins
- Plugin stacks
- Configuration Rules
- DSL Extensions
- Init Scripts

Overview

- Deep API
- Configuration Rules
- Plugins
- Init Scripts
- Custom Distributions

Deep API

Many aspects of Gradle are configurable. It pays to dig through the API doc.

Example: How do I enforce that no one can compile against Hibernate?

- [demos/01-deep-api-dependency-rules]

Example: How do I verify the existence of a "`test`" task?

- [demos/02-deep-api-require-task]

Deep API (cont.)

A design philosophy of Gradle is to give you deep access to the entire model.

The API is often described as an Onion Skin API.

Reading the Gradle Javadoc does not mean you are doing something wrong.

Configuration Rules

Gradle provides many ways to configure future objects.

You often want to configure things that you don't know about.

Example: How do configure all test tasks to be as parallel as they can be?

- [demos/03-configuration-rule-parallel-tests]

Example: How do I disallow Hibernate as a dependency at all (e.g. `compile`, `test`, `runtime`)?

- [demos/04-configuration-rule-dependency-rules]

Configuration Rules (cont.)

Based on `DomainObjectCollection` and `NamedDomainObjectCollection`.

Java Collections that support “liveness.”

```
def numbers = new DefaultDomainObjectCollection(Number, [1, 2])

def evenNumbers = numbers.matching { it % 2 == 0 }
assert evenNumbers == [2]

numbers << 4
assert evenNumbers == [2, 4]
```

Configuration Rules (cont.)

Many objects you use in Gradle are live collections.

- `project.tasks` - TaskContainer
- `project.plugins` - PluginContainer
- `project.configurations` - ConfigurationContainer
- `project.repositories` - RepositoryHandler
- `project.sourceSets` - SourceSetContainer
- `project.«configuration».dependencies` - DependencySet

Configuration Rules (cont.)

Some common use cases…

Run some code against all things that exist now and in the future:

```
tasks.all { }
```

Do something when new things are added:

```
repositories.whenObjectAdded { }
```

Do something when things are removed:

```
repositories.whenObjectRemoved { }
```

Plugins

Plugins are just reusable build script code. That is, there is no functionality that only plugins can perform.

Plugins can:

- Add new capabilities/tools (often with defaults)
- Add new conventions or extend tighten existing conventions

Ideally, they do only one of those.

Combining these two inseparably leads to rigidity.

Adding new capabilities

Adding the ability to perform some new function. For example, to compile CoffeeScript.

Out of scope for this Webinar.

Keep an eye on http://gradle.org/news for announcements on a Webinar focussed on this topic.

Adding/Extending Conventions

Configures the project(s) to do useful work by adding/configuring tools and capabilities.

May provide defaults, may enforce standards.

Example conventions:

- Use JUnit 4.10 for tests
- Integration tests will be their own source set and task
- Dependencies always come from our corporate repository
- Source code must be Java 1.5 compatible

Plugin Types

Plugins can be packaged as either scripts or objects.

Script plugins are `.gradle` files.

- [05-plugins-script-plugin]

Object plugins are compiled and on the classpath.

- [06-plugins-object-plugin-buildSrc]
- [07-plugins-object-plugin-repo]

DSL Extensions

If you are creating conventions the are configurable or have options, consider wrapping them in DSL extensions.

DSL extensions add new constructs to the build language.

The canonical example of an extension would be the SourceSetContainer.

- [demos/08-dsl-extensions]

Plugin Stacks

Sophisticated plugins, like the Java plugin, are comprised of stacks.

`BasePlugin JavaBasePlugin JavaPlugin`

- `BasePlugin` = general build features (language agnostic)
- `JavaBasePlugin` = sensible defaults for Java tools
- `JavaPlugin` = conventions for a Java project

Your own 'java' plugin

If those conventions don't suit, you can just apply the `java-base` plugin and build your own conventions.

```
apply plugin: 'java-base'
```

However, not a lot of documentation on this at this time.

The source code and the Gradle Forums are your friends.

Init Scripts

Gradle init scripts facilitate cross project configuration.

Init scripts work with a `Gradle` instance, not a `Project`.

Looked for at:

- $USER_HOME/.gradle/init.gradle
- $USER_HOME/.gradle/init.d/*.gradle
- $GRADLE_HOME/init.d/*.gradle

Can also be specified with `-I` command line arg.

Init Scripts (cont.)

What can build scripts do?

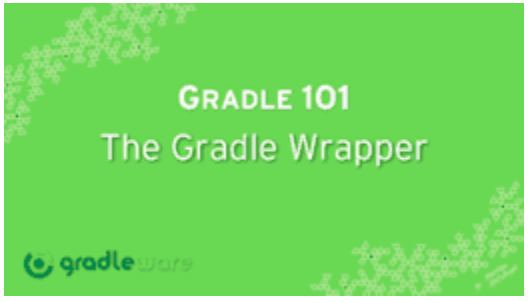
- Load plugins
- Specify plugin repositories
- Environmental configuration (CI server specific conf)
- Register listeners
- More…

Can also do anything that you can do in a build script.

- [09-init-script-plugin-repo]

The Gradle Wrapper

The Gradle Wrapper allows you to start Gradle via some scripts that you check in to source control. These scripts will go ahead and download Gradle if the user does not have it installed on their machine.



Check out the Gradle Wrapper screencast!

The wrapper doesn't have to download Gradle from our servers, it can download it from yours.

And the distribution may be customised.

Custom Distributions

You can build a custom distribution by packaging your init script inside the Gradle distribution.

This way, every build will implicitly use your own enterprise init script.

- [10-custom-distribution]

Custom Distributions (cont.)

In the future, this will be simpler. Something like…

```
task(type: Wrapper) {
  gradleVersion "1.0"
  initScript "http://com.com/init/corporate-plugin.gradle"
  initScript "http://com.com/init/other-init-script.gradle"
}
```

Then the Gradle wrapper will be responsible for fetching and managing the init scripts.

Gradleware

The company behind Gradle.

- Employs full time engineers
- Gradle consulting, support, development services etc.
- Training: online, public and in-house
- General build automation services

Germany, Australia, UK, Poland, Austria, Canada and the US.

http://www.gradleware.com

Questions?

- Questions on this webinar
- Feedback on this webinar (email me directly if you prefer)
- Ideas for future webinars
- Anything else?

More stuff...

- Gradle Training
  - Gradle In-Depth, Chicago, Oct 9-11 by Gradle Core Developer
- Gradle Webinars
  - Migrating and Upgrading to Gradle by Szczepan Faber
- Gradle News
- Gradle Forums
- Gradleware Resources