

G*ワークショップZ Gradleハンズオン

2013.03.15

須江信洋(@nobusue)

<http://d.hatena.ne.jp/nobusue>

<https://www.facebook.com/nobuhiro.sue>

自己紹介

- ▶ 須江 信洋(すえ のぶひろ)
 - ▶ Twitter: @nobusue
 - ▶ <https://www.facebook.com/nobuhiro.sue>
- ▶ 約10年ほどJavaEE関連の仕事をしています
- ▶ 最近はPhoneGap(Cordova)とかがメイン
- ▶ G*(Groovy関連技術)との関わり
 - ▶ JGGUGサポートスタッフ
 - ▶ 「プログラミングGROOVY」執筆チーム
 - ▶ 「Groovy イン・アクション」翻訳チーム
 - ▶ Groovyで作ったBot飼ってます(@hatena_groovy)



agenda

- ▶ ~~Gradle概要~~-(thanks to 林さん)
- ▶ Gradleインストール
- ▶ Hello Gradle
- ▶ Gradle Tools
- ▶ Gradle Quickstart
- ▶ Gradle Build Language概要
- ▶ Ant integration
- ▶ IDE integration
- ▶ Jenkins integration
- ▶ 便利なGradleプラグイン紹介
- ▶ 参考情報

本日のサンプルコード

- ▶ GitHubから入手してください
 - ▶ <https://github.com/nobusue/GradleHandson>
 - ▶ git clone
https://github.com/nobusue/GradleHandson
- ▶ より詳しい例はGradle公式サンプルを参照
 - ▶ <http://www.gradle.org/downloads>
 - ▶ gradle-1.4-all.zipの"samples"ディレクトリ

Gradleインストール

▶ 前提

- ▶ JDK1.5以上 (“java -version”で確認)

▶ GVM利用

- ▶ `curl -s get.gvmtool.net | bash`
- ▶ `gvm install gradle`
- ▶ 詳細は <http://gvmtool.net/> 参照
- ▶ Windowsの場合はcygwinが必要

▶ ZIPを展開

- ▶ <http://www.gradle.org/downloads>
- ▶ `gradle-1.4-all.zip`か`gradle-1.4-bin.zip`
- ▶ 適当なディレクトリに展開 (`$GRADLE_HOME`)
- ▶ `$GRADLE_HOME/bin` にパスを通しておく

動作確認

▶ gradle -v

```
C:\Users\nobusue>gradle -v
```

```
-----  
Gradle 1.4  
-----
```

```
Gradle build time: 2013年1月28日 (月曜日) 3時42分46秒 UTC  
Groovy: 1.8.6  
Ant: Apache Ant(TM) version 1.8.4 compiled on May 22 2012  
Ivy: 2.2.0  
JVM: 1.6.0 (IBM Corporation 2.4)  
OS: Windows 7 6.1 build 7601 Service Pack 1 amd64
```

GVMでインストールした場合は ~/.gvm/gradle/1.4 以下に導入され、
~/.gvm/gradle/current にシンボリックリンクが作成されます

Hello Gradle

- ▶ 適当な作業ディレクトリを作成し、カレントディレクトリを移動します
- ▶ 以下の内容で“build.gradle”を作成します

```
task helloWorld << {  
    println 'Hello world.'  
}
```

- ▶ “gradle helloWorld” を実行します

```
C:\Users\nobusue\work\hello>gradle helloWorld  
:helloWorld  
Hello world.  
  
BUILD SUCCESSFUL  
  
Total time: 2.605 secs
```

Hello Gradle解説

タスクの定義

タスクにクロージャを追加
※ leftShift()の省略記法

```
task helloWorld << {  
    println 'Hello world.'  
}
```

Groovyのprintln文

デフォルトGradleタスク

▶ gradle tasks

```
-----  
All tasks runnable from root project  
-----
```

```
Help tasks  
-----
```

```
dependencies - Displays all dependencies declared in root project 'hello'.  
dependencyInsight - Displays the insight into a specific dependency in root project 'hello'.  
help - Displays a help message  
projects - Displays the sub-projects of root project 'hello'.  
properties - Displays the properties of root project 'hello'.  
tasks - Displays the tasks runnable from root project 'hello' (some of the displayed tasks may belong to subprojects).
```

```
Other tasks  
-----
```

```
helloWorld
```

Gradle Tools

- ▶ gradle command
- ▶ gradle daemon
- ▶ gradle-gui

gradle command

► gradle --help

```
USAGE: gradle [option...] [task...]

-?, -h, --help            Shows this help message.
-a, --no-rebuild           Do not rebuild project dependencies.
-b, --build-file           Specifies the build file.
-C, --cache               Specifies how compiled build scripts should be cached. Possible values are: 'rebuild' and 'on'. Default value is 'on' [deprecated - Use '--rerun-tasks' or '--recompile-scripts' instead]
-c, --settings-file       Specifies the settings file.
--continue                Continues task execution after a task failure.
-D, --system-prop         Set system property of the JVM (e.g. -Dmyprop=myvalue).
-d, --debug               Log in debug mode (includes normal stacktrace).
--daemon                  Uses the Gradle daemon to run the build. Starts the daemon if not running.
--foreground              Starts the Gradle daemon in the foreground. [incubating]

-g, --gradle-user-home    Specifies the gradle user home directory.
--gui                     Launches the Gradle GUI.
-I, --init-script          Specifies an initialization script.
-i, --info                Set log level to info.
-m, --dry-run             Runs the builds with all task actions disabled.
--no-color                Do not use color in the console output.
--no-daemon               Do not use the Gradle daemon to run the build.
```

gradle commandの重要オプション

▶ ログ関連

- ▶ --quiet(-q) ログ出力を抑制
- ▶ --info(-i) / --debug(-d)
- ▶ --stacktrace(-s) / --full-stacktrace(-S)

▶ ファイル/ディレクトリ指定

- ▶ --build-file(-b) build.gradle以外
- ▶ --project-dir(-p) カレントディレクトリ以外

▶ テスト

- ▶ --dry-run(-m) タスクの実行順序のみ確認

環境まわりのまとめ

▶ 環境変数

- ▶ JAVA_OPTS ⇒ JVM全体に影響
- ▶ GRADLE_OPTS ⇒ gradleのみ影響

▶ 初期化スクリプト(init scripts)

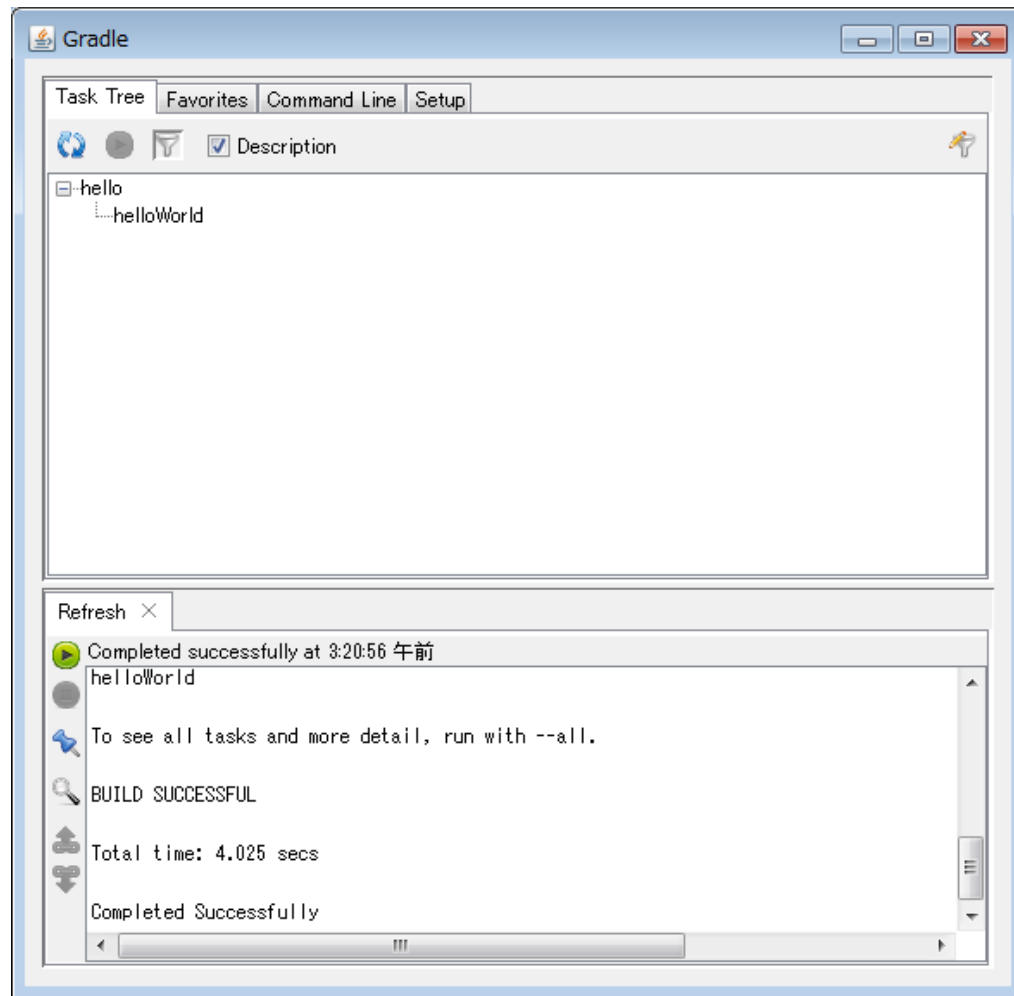
- ▶ ビルドの開始前に実行される
- ▶ ~/.gradle/init.gradle
- ▶ ~/.gradle/init.d/*.gradle
- ▶ \$GRADLE_HOME/init.d/*.gradle
- ▶ --init-script(-I)で指定することも可能

gradle daemon

- ▶ gradleのプロセスを常駐して起動を高速化
- ▶ 起動
 - ▶ `gradle --daemon helloWorld`
- ▶ 停止
 - ▶ `gradle --stop`
- ▶ 起動済みのプロセスを使わない
 - ▶ `gradle -no-daemon helloWorld`
- ▶ 常にdaemonを使う場合は以下いずれか
 - ▶ `alias gradled='gradle --daemon'`
 - ▶ `export GRADLE_OPTS="-Dorg.gradle.daemon=true"`

gradle-gui

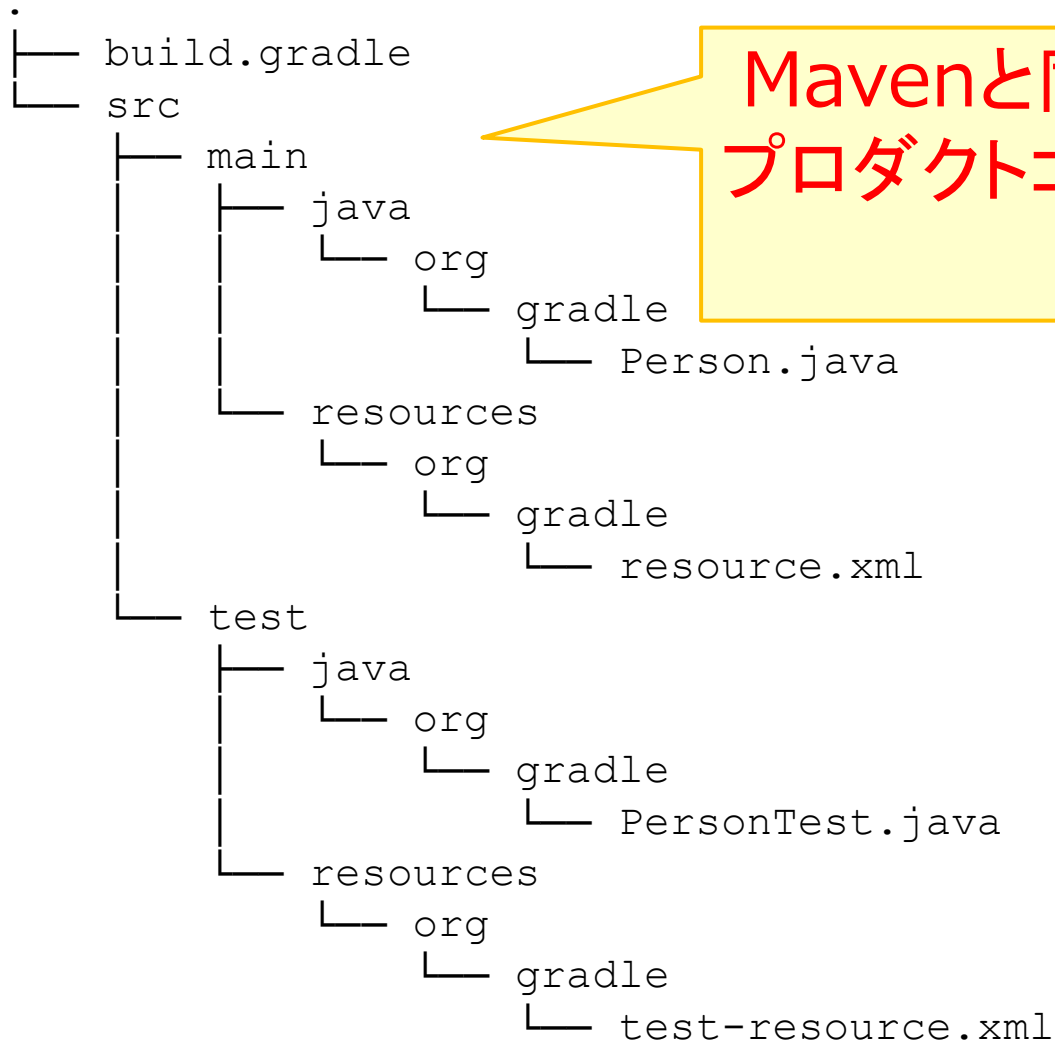
▶ gradle --gui



Gradle Quickstart

- ▶ Java project
- ▶ Groovy project
- ▶ Web project

Javaプロジェクトのレイアウト



Mavenと同様の規約に従って
プロダクトコードとテストコードを
配置

Javaプロジェクトのビルドスクリプト(抜粋)

```
apply plugin: 'java'
```

Javaプラグインを適用

```
repositories {  
    mavenCentral()  
}
```

依存性解決にMavenリポジトリを利用

```
dependencies {  
    compile(  
        group: 'commons-collections',  
        name: 'commons-collections',  
        version: '3.2')
```

プロダクトコードのコンパイル時の依存先

```
    testCompile(  
        group: 'junit',  
        name: 'junit',  
        version: '4.+')
```

テストコードのコンパイル時の依存先

```
}
```

Javaプラグインが追加するタスク

Build tasks

assemble - Assembles the outputs of this project.

build - Assembles and tests this project.

buildDependents - Assembles and tests this project and all projects that depend on it.

buildNeeded - Assembles and tests this project and all projects it depends on.

classes - Assembles the main classes.

clean - Deletes the build directory.

jar - Assembles a jar archive containing the main classes.

testClasses - Assembles the test classes.

Documentation tasks

javadoc - Generates Javadoc API documentation for the main source code.

Javaプラグインが追加するタスク

Upload tasks

uploadArchives - Uploads all artifacts belonging to configuration ':archives'

Verification tasks

check - Runs all checks.

test - Runs the unit tests.

Rules

Pattern: build<ConfigurationName>: Assembles the artifacts of a configuration.

Pattern: upload<ConfigurationName>: Assembles and uploads the artifacts belonging to a configuration.

Pattern: clean<TaskName>: Cleans the output files of a task.

Groovyプロジェクトのレイアウト

```
.
├── build.gradle
├── src
│   ├── main
│   │   ├── groovy
│   │   │   ├── org
│   │   │   │   └── gradle
│   │   │   │       └── Person.groovy
│   │   └── resources
│   │       ├── resource.txt
│   │       └── script.groovy
│   └── test
│       ├── groovy
│       │   ├── org
│       │   │   └── gradle
│       │   │       └── PersonSpec.groovy
│       └── resources
│           ├── testResource.txt
│           └── testScript.groovy
```

Groovyプロジェクトのビルドスクリプト

```
apply plugin: 'groovy'
```

Groovyプラグインを適用

```
repositories {  
    mavenCentral()  
}
```

依存性解決にMavenリポジトリを利用

```
dependencies {  
    compile 'org.codehaus.groovy:groovy-all:2.0.5'  
    testCompile "org.spockframework:spock-core:0.7-groovy-2.0"  
}
```

プロダクトコードのコンパイル時の依存先

テストコードのコンパイル時の依存先

Groovyプラグインが追加するタスク

Documentation tasks

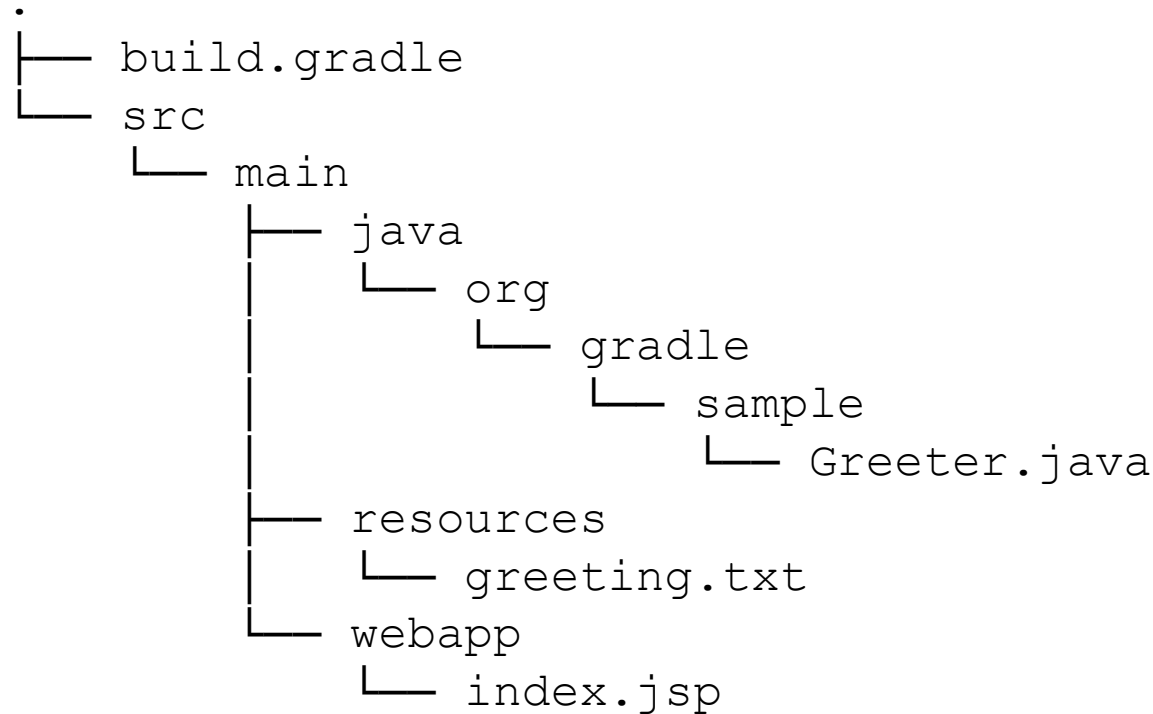
javadoc - Generates Javadoc API documentation for the main source code.

Documentation tasks

groovydoc - Generates Groovydoc API documentation for the main source code.

GroovyプラグインはJavaプラグインを拡張して作られているため、Javaプラグインが提供するタスクはそのまま利用可能
(ただし、一部のタスクはGroovy用に機能拡張されている)

Webappプロジェクトのレイアウト



Webappプロジェクトのビルドスクリプト

apply plugin: 'war'

Warプラグインを適用

apply plugin: 'jetty'

Jettyプラグインを利用して
ビルドスクリプト上でwarを実行可能

repositories {
 mavenCentral()
}

依存性解決にMavenリポジトリを利用

dependencies {
 compile group: 'commons-io', name: 'commons-io', version: '1.4'
 compile group: 'log4j', name: 'log4j', version: '1.2.15', ext: 'jar'
}

プロダクトコードのコンパイル時の依存先

httpPort = 8080
stopPort = 9451
stopKey = 'foo'

Jettyプラグインのパラメータ

Webappプラグインが追加するタスク

Build tasks

war - Generates a war archive with all the compiled classes, the web-app content and the libraries.

Web application tasks

jettyRun - Uses your files as and where they are and deploys them to Jetty.

jettyRunWar - Assembles the webapp into a war and deploys it to Jetty.

jettyStop - Stops Jetty.

WebappプラグインはJavaプラグインを拡張して作られているため、Javaプラグインが提供するタスクはそのまま利用可能
(ただし、一部のタスクはWebapp用に機能拡張されている)

Gradle Build Language概要

- ▶ Gradle DSL
- ▶ Gradle Domain Objects

Gradle DSLのエントリーポイント

▶ Gradleビルド言語リファレンス

- ▶ <http://gradle.monochromeroad.com/docs/dsl/index.html>

Home

はじめに
いくつかの基本
ビルドスクリプトの構造
核となる型
コンテナ型
補助タスク型
タスク型
Eclipse/IDEAモデル型
Eclipse/IDEAタスク型

ビルドスクリプトのブロック

Build script blocks

```
allprojects { }  
artifacts { }  
buildscript { }  
configurations { }
```

Gradleビルド言語リファレンス

Version 1.5-20130308082157+0000

はじめに

このリファレンスガイドは、Gradleビルド言語、DSLを構成する、様々な型について説明するものです。

いくつかの基本

あなたが理解すべきなのは少々の基本構想で、それらはあなたがGradleスクリプトを記述する際に助けになるでしょう。

タスク定義

```
task('hello') << {  
    println "hello"  
}
```

```
task('copy', type: Copy) {  
    from(file('srcDir'))  
    into(buildDir)  
}
```

```
task(hello) << {  
    println "hello"  
}
```

```
task(copy, type: Copy) {  
    from(file('srcDir'))  
    into(buildDir)  
}
```

```
task copy(type: Copy) {  
    description = 'Copies the resource directory to the target directory.'  
    from 'resources'  
    into 'target'  
    include('**/*.txt', '**/*.xml', '**/*.properties')  
}
```

デフォルトタスク

```
defaultTasks 'clean', 'run'
```

```
task clean << {  
    println 'Default Cleaning!'  
}
```

```
task run << {  
    println 'Default Running!'  
}
```

```
task other << {  
    println "I'm not a default task!"  
}
```

タスク依存関係(task利用)

```
task taskX << {  
    println 'taskX'  
}
```

```
task taskY << {  
    println 'taskY'  
}
```

```
taskX.dependsOn taskY
```

タスク名でtaskオブジェクトを参照

タスク依存関係(クロージャ利用)

```
task taskX << {  
    println 'taskX'  
}  
taskX.dependsOn {  
    tasks.findAll { task -> task.name.startsWith('lib') }  
}  
  
task lib1 << {  
    println 'lib1'  
}  
task lib2 << {  
    println 'lib2'  
}  
task notALib << {  
    println 'notALib'  
}
```

taskのコレクションを返すクロージャ

description、グループ化

動的タスク定義

```
4.times { counter ->
  task "task${counter}" << {
    println "${counter+1}番目の動的タスクです"
  }
}
```

```
Other tasks
-----
task0
task1
task2
task3
```

```
C:¥work¥gradle¥dynamic>gradle task1
:task1
2番目の動的タスクです

BUILD SUCCESSFUL
```

条件分岐

```
task "OsDependTask" << {  
    def os = System.getProperty("os.name")  
    if(os.contains("Windows")) {  
        println "Windows用の処理" }  
    else if(os.contains("Mac OS")) {  
        println "Mac OS用の処理" }  
    else { println "Linux/Unix用の処理" }  
}
```

```
C:¥work¥gradle¥condition>gradle OsDependTask  
:OsDependTask  
Windows用の処理  
  
BUILD SUCCESSFUL
```

Gradleの代表オブジェクト

- ▶ `org.gradle.api.Project`
 - ▶ ビルドスクリプト(`build.gradle`)に対応
 - ▶ 中核となるオブジェクト
 - ▶ ビルドスクリプト内では暗黙的に、もしくは`project`プロパティで参照

Gradleの代表オブジェクト

- ▶ `org.gradle.api.invocation.Gradle`
 - ▶ 実行中のビルドエンジンに対応
 - ▶ 初期化時の情報や、環境情報を保持
 - ▶ `Project.getGradle()` で取得可能

Gradleの代表オブジェクト

- ▶ `org.gradle.api.initialization.Settings`
 - ▶ `settings.gradle`に対応
 - ▶ マルチプロジェクト構成時にプロジェクト階層の情報を保持

便利メソッド/プロパティ

- ▶ file()
 - ▶ 相対/絶対パス、Fileオブジェクト、URLなど
 - ▶ PathValidationでいろいろ判定可能
- ▶ files()
 - ▶ ファイルのコレクション(filter可能)
 - ▶ 引数としてファイルを返すtaskを渡せる
- ▶ fileTree()
 - ▶ ファイルのツリー階層をトラバース(visit)可能
 - ▶ Antのpathelement式でinclude/exclude可能
- ▶ logger
 - ▶ SLF4J Loggerのインスタンス
 - ▶ 標準出力への出力はQUIETレベルにリダイレクト

拡張プロパティ

- ▶ Gradleのドメインモデルにプロパティを追加する際には、extプロパティ(extブロック)を使うこと
 - ▶ Gradle-1.0M9からこちらが強く推奨されています
 - ▶ 現在は互換性のため未定義のプロパティがあってもエラーにならないが、警告が出ます
- ▶ ローカル変数(def)と異なり、ドメインモデルのライフサイクル全体で利用できます
- ▶ 詳細はこちらの「13.4.2. 拡張プロパティ」を参照
 - ▶ http://gradle.monochromeroad.com/docs/userguide/writing_build_scripts.html

Ant integration

- ▶ Antタスクの利用
- ▶ Antビルド定義の利用

AntからGradleへ

- ▶ Gradleは既存のAnt資産を活用できる
 - ▶ Antのbuild.xmlをそのまま読み込んで実行可能
 - ▶ Antターゲット=Gradleタスク
 - ▶ AntタスクをGradleから直接利用可能
 - ▶ GroovyのAntBuilderが組み込まれている
 - ▶ AntタスクとGradleタスクを共存することも可能
 - ▶ 相互に依存するタスクも定義できる
 - ▶ AntタスクをGradleから拡張することもできる
- ▶ Gradleは「Better Ant」としても使える
 - ▶ Mavenとの大きな違い
 - ▶ Antから段階的にGradleへ移行できる

GradleでAntタスクを利用

```
task hello << {  
    ant.echo('Antタスクの実行')  
}
```

```
Other tasks  
-----  
hello
```

```
C:\work\gradle\anttask>gradle hello  
:hello  
[ant:echo] Antタスクの実行  
  
BUILD SUCCESSFUL
```

GradleでAntのビルド定義を利用

[build.gradle]

```
ant.importBuild 'build.xml'
```

[build.xml]

```
<project>  
  <target name="hello">  
    <echo>Antターゲットの実行</echo>  
  </target>  
</project>
```

```
Other tasks  
-----  
hello
```

```
C:\work\gradle\ant\import>gradle hello  
:hello  
[ant:echo] Antターゲットの実行  
  
BUILD SUCCESSFUL
```

IDE integration

- ▶ IntelliJ IDEA
- ▶ Eclipse

Jenkins integration

- ▶ Gradle wrapper
- ▶ Jenkins Gradle plugin

Jenkinsとの統合 - Gradle

- ▶ Gradle wrapperを利用
 - ▶ JDKさえ導入されていれば、Gradleを自動でインストールして実行してくれる
 - ▶ Jenkins上では汎用コマンドとして実行すればよい
- ▶ Gradle wrapperは、CI用途に限らず、開発者の環境構築負荷軽減にも有用

Gradle wrapper

[build.gradle]

```
task wrapper(type: Wrapper) {  
    gradleVersion = '1.4'  
}
```

Gradle導入済の環境で"gradle wrapper"を実行すると、Gradleのブートストラップ(gradlew)が生成される

```
build.gradle  
gradlew  
gradlew.bat  
  
└─ gradle  
    └─ wrapper  
        gradle-wrapper.jar  
        gradle-wrapper.properties
```



Jenkinsのビルド設定

ジョブ名

☒ **フリー スタイル・プロジェクトのビルド**

もっとも汎用性の高いJenkinsの中核機能です。任意のSCMからソースコードをチェックアウトし、任意のビルドシステムでプロジェクトがビルドできます。往々にして、ソフトウェアのビルド以外にも様々な仕事の自動化に利用することができます。

☐ **Maven2/3プロジェクトのビルド**

Maven2/3のプロジェクトをビルドします。JenkinsはPOMファイルから必要な情報を読み取るので、設定が必要な項目はごくわずかです。

☐ **外部ジョブの監視**

Jenkinsの外(含む別のマシン上)で実行されるプロセスの実行をJenkinsに記録します。これにより、既存の自動化システムの動作をJenkinsを使って監視できます。[詳しくはこのドキュメント](#)をご覧ください。

☐ **マルチ構成プロジェクトのビルド**

複数の環境でのテストや、プラットフォームごとのビルドなどといった、多数の異なる構成が必要なプロジェクトに適しています。

ビルド

Windowsバッチコマンドの実行

コマンド

[ビルドから利用可能な環境変数の一覧](#)

削除

ビルド手順の追加 ▼

Jenkinsとの統合 - Spock

▶ Spock

- ▶ レポートはJUnit互換なので、Jenkinsでそのまま利用可能

(root)の履歴

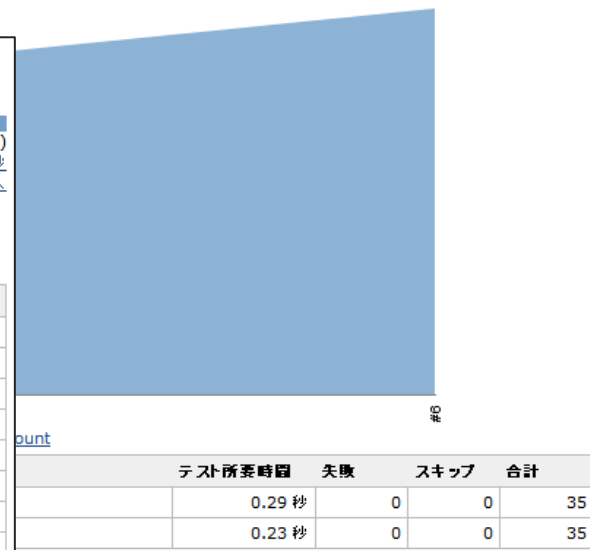
テスト結果 : (root)

0個の失敗 (±0)

35個のテスト (±0)
所要時間 0.29 秒
[説明を記入](#)

すべてのテスト


クラス	テスト所要時間	失敗	(差分)	スキップ	(差分)	合計	(差分)
DataDriven	16 ms	0		0		8	
DatabaseDriven	78 ms	0		0		1	
DerivedSpec	15 ms	0		0		2	
EmptyStack	31 ms	0		0		4	
HamcrestMatchers	31 ms	0		0		1	
HelloSpock	16 ms	0		0		1	
IncludeExcludeExtension	0 ms	0		0		3	
OrderedInteractions	32 ms	0		0		1	
PublisherSpec	47 ms	0		0		2	
StackWithOneElement	15 ms	0		0		4	
StackWithThreeElements	0 ms	0		0		4	
StepwiseExtension	0 ms	0		0		3	
UsingJUnitRules	16 ms	0		0		1	



Jenkinsのテストレポート設定

▶ JUnitと同様

ビルド後の処理

☒ JUnitテスト結果の集計 

テスト結果XML

[Antのファイルセットincludes属性の書式](#)に従ってビルドから生成されるXMLレポートファイル群を指定します(例: myproject/target/test-reports/*.xml)。ワークスペースルートからの相対パスです。

☐ Retain long standard output/error 

Gradleプラグイン紹介

Gradleプラグイン、順調に増殖中

- 26 Ivyxml Plugin
- 27 Javascript Library Plugin
- 28 JavaFx Plugin
- 29 JavaPropFile Plugin
- 30 JAXB Plugin
- 31 jDocbook Plugin
- 32 JDepend Plugin
- 33 Launch4J Plugin
- 34 MacAppBundle Plugin
- 35 PMD Plugin
- 36 Protocol Buffers Plugin
- 37 OneJar plugin
- 38 Release Plugin
- 39 SvnKit Plugin
- 40 Templates Plugin
- 41 Tomcat Plugin
- 42 Xslt Plugin
- 43 Workspace Plugin
- 44 PGP Plugin
- 45 org.linkedin plugins
 - 45.1.1 a. org.linkedin.userConfig
 - 45.1.2 b. org.linkedin.spec
 - 45.1.3 c. org.linkedin.repository
 - 45.1.4 d. org.linkedin.release
 - 45.1.5 e. org.linkedin.cmdline
- 46 JSLint Plugin
- 47 Ubuntu Packager Plugin
- 48 RPM Plugin
- 49 Writing Custom Plugins
 - 49.1 Example of a simple custom task within buildSrc
 - 49.2 Example of a simple custom plugin within buildSrc
 - 49.3 Building an external plugin (outside buildSrc)

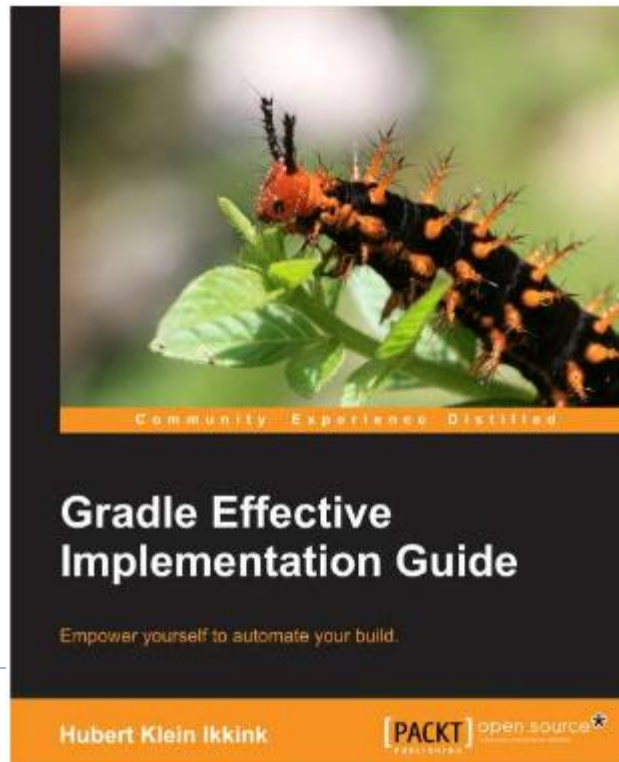
[http://wiki.
gradle.org
/display/G
RADLE/Plu
gins](http://wiki.gradle.org/display/GRADLE/Plugins)

Gradleの情報源

- ▶ Gradle 日本語ドキュメント(@literaliceさん)
 - ▶ <http://gradle.monochromeroad.com/docs/index.html>
- ▶ ビルドツールGradle スタートアップガイドの紹介
 - ▶ http://www.ntts.co.jp/publish/column/tec/java_03/index.html
- ▶ GVM:the Groovy enVironment Manager
 - ▶ <http://gvmtool.net/>
 - ▶ groovy/grails/gradle/griffon/vert.xに対応
- ▶ Gradleプロジェクトのソースコード
 - ▶ <https://github.com/gradle/gradle>
 - ▶ Gradleのdownloadにあるsrc.zipはビルドスクリプトがないので、ソースコードリーディングにはこちらを参照するべし

おすすめ書籍

- ▶ “Gradle Effective Implementation Guide” Hubert Klein Ikkink (mrhaki)
- ▶ <http://mrhaki.blogspot.de/search/label/Gradle%3AGoodness> の人の本です



ありがとうございました



powered by NekoFont

<http://nekofont.upat.jp/>