

Gradle勉強会

アジェンダ

- ・ Gradle(ビルド自動化)の歴史
- ・ Javaビルド
- ・ Gradleプラグイン

Gradleとは？

- ・ ビルド実行を自動化するためのツール
- ・ 従来のビルドツールの特徴を引き継ぎ、かつ記述が簡単
- ・ Android / Spring / Hibernateなどの有名Javaプロジェクトの公式ビルドツールに採用されている

資料

- https://github.com/kaakaa/Gradle_Hands-on



Javaビルドの歴史

ビルド自動化の歴史

ソフトウェア開発(Java)

コーディング

プログラミング

コンピュータが実行
しやすい形に

コンパイル

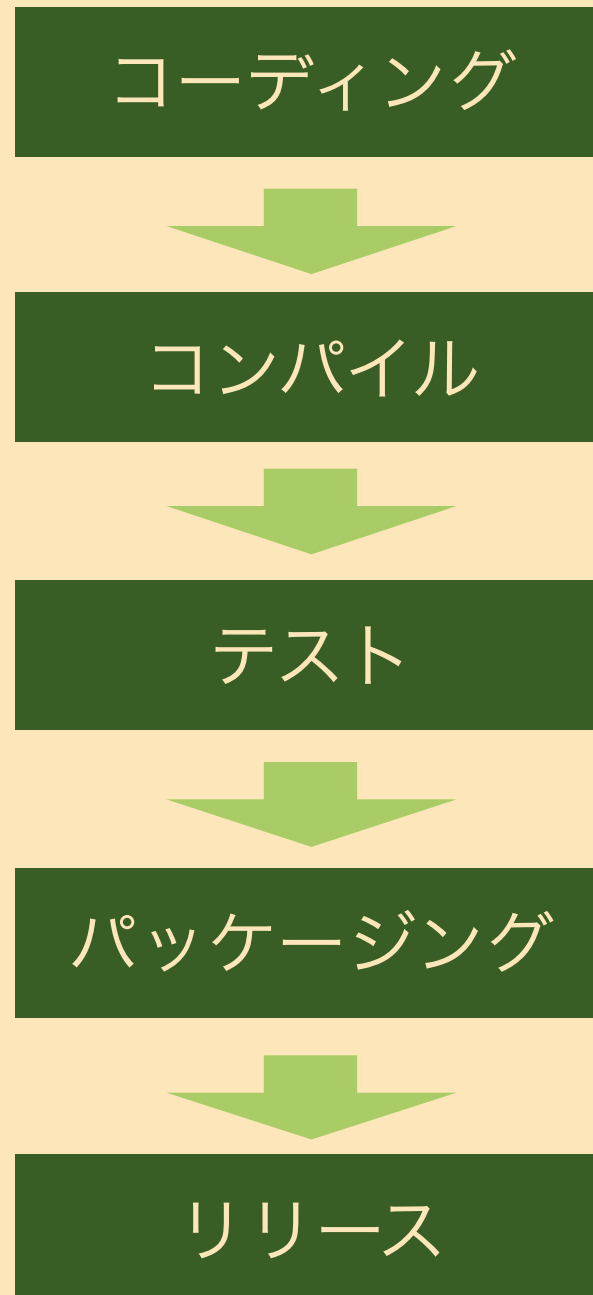
テスト

パッケージング

配布しやすい形に

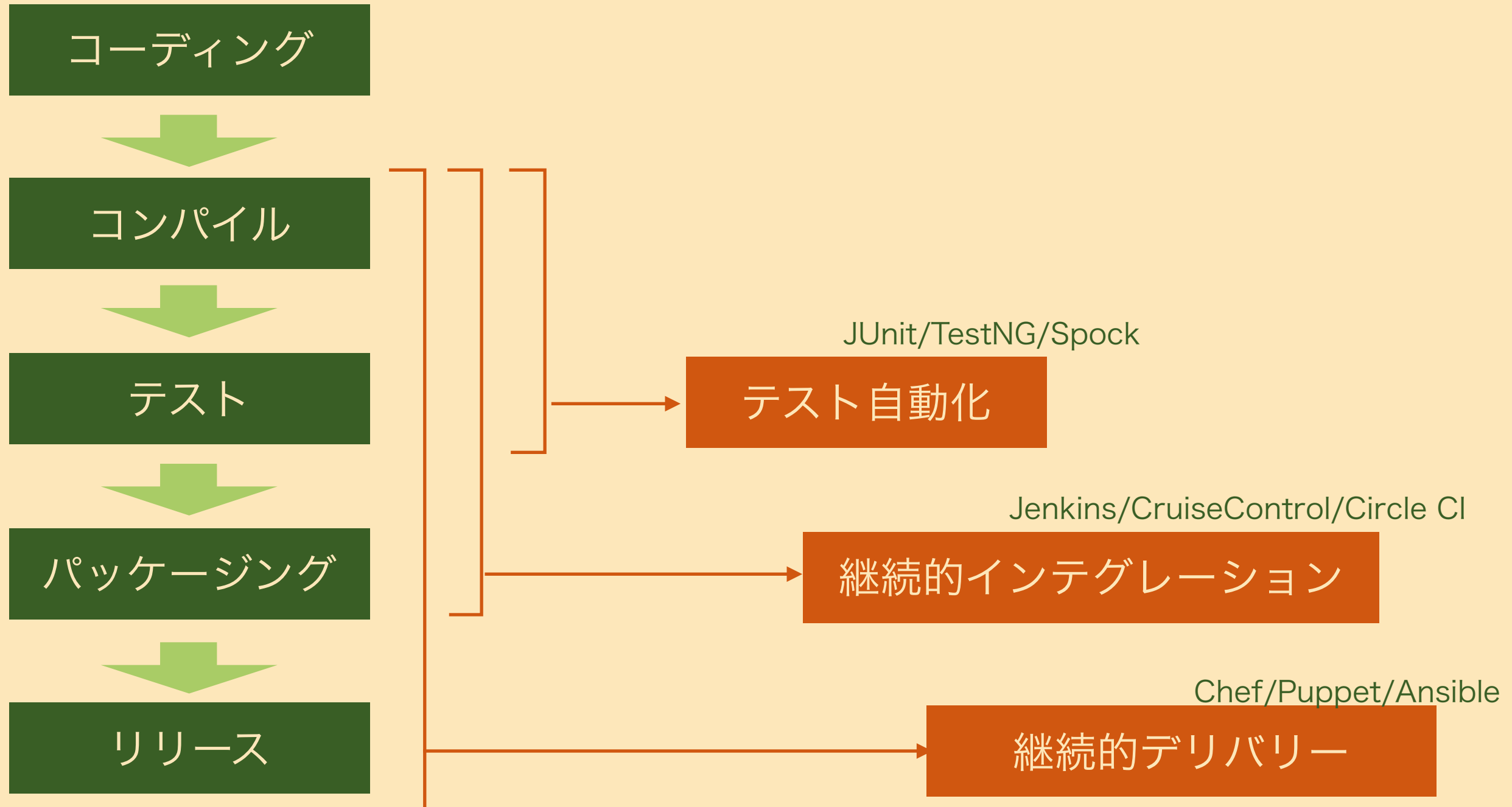
実行できる形に

リリース



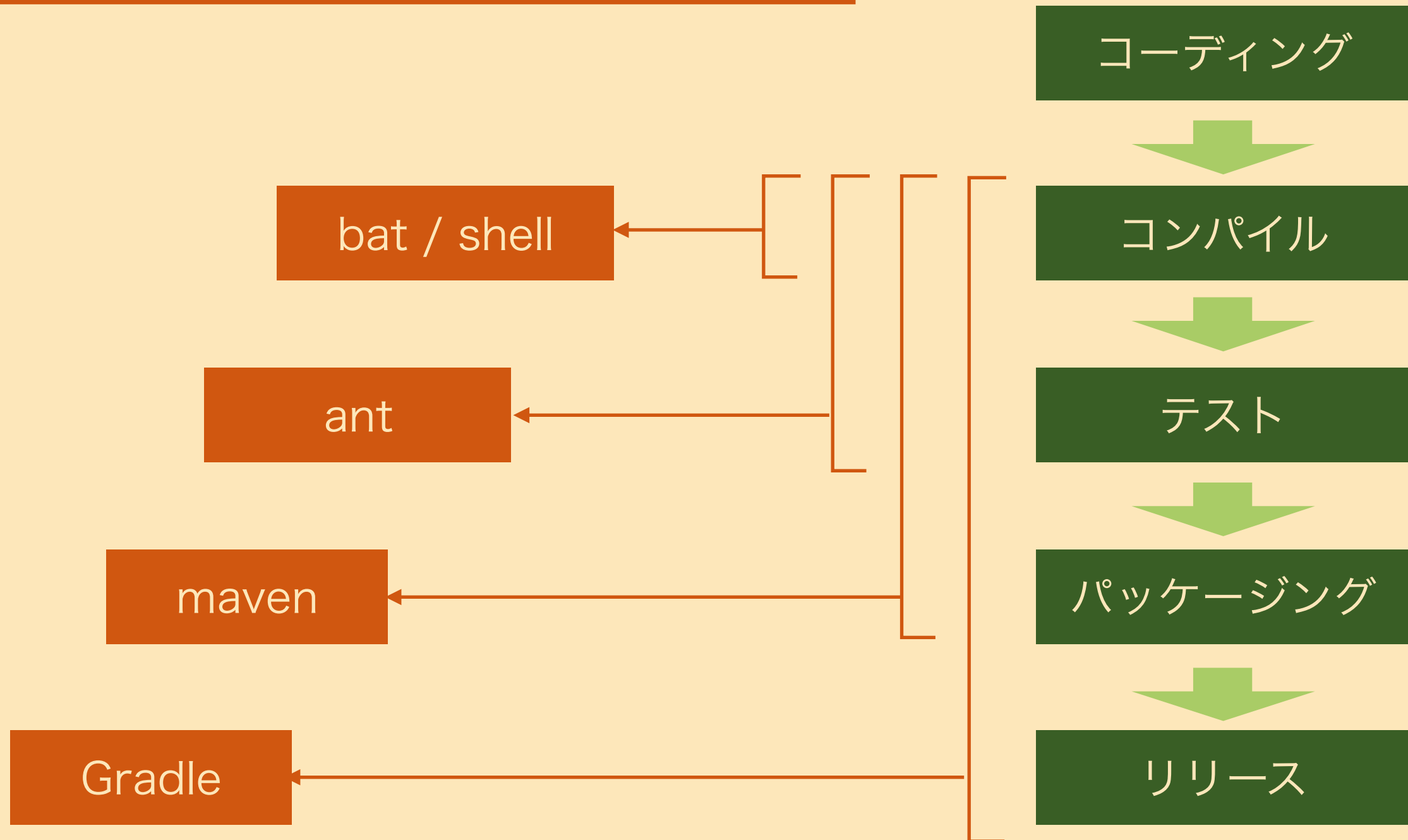
ソフトウェア開発 - 自動化の歴史

自動化の領域が広がっている



ソフトウェア開発 - ビルドスクリプト

ビルドスクリプトも進化している



※リリースされた時代背景のイメージであり、各ツールが対応するタスクの自動化を想定している訳ではないです

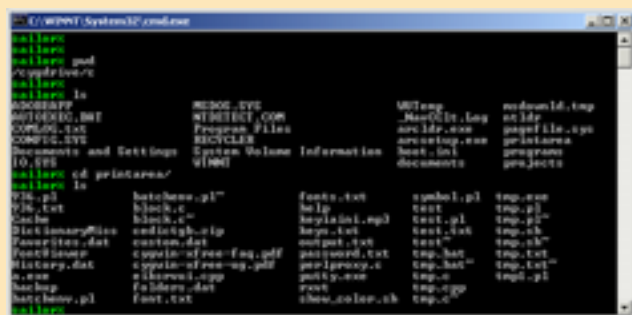
ビルドスクリプトの歴史



BAT / Shell

- ・ コンピュータで出来ることならほぼ何でも出来る
- ・ 統一的な記述方法がない

=> 属人性が高くなる(ビルド職人)



2000

2004

2012

Ant

- ・ 2000/7/19 Ant 1.1 リリース
=> Tomcat のビルドツールとして開発された
- ・ Ant = Another Neat Tool
=> Neat: さっぱりした (NEETではない)
- ・ XMLによる宣言的な記述
=> XML地獄の始まり

(参考)

http://en.wikipedia.org/wiki/Apache_Ant#History

<http://ant.apache.org/faq.html#ant-name>



2000

2004

2012

Maven

- 2004/7/13 Maven 1.1 リリース (最新ver 3.2.1)
- Maven Repository
 - => 依存関係問題を成果物リポジトリに任せる
 - => Antでも依存関係解決のために ivy が作られる
- Conversion Over Configuration (設定より規約)
 - => 規約通りに作ることによって設定項目を減らす
 - => しかし、XML地獄からは抜け出せず



2000

2004

2012

Gradle

- ・ 2012/6/12 Gradle 1.0 リリース
=> 2014/7/1 Gradle 2.0 公開
- ・ Groovy DSLによる簡潔な記述
=> Ant/Mavenと続いたXML地獄からの脱却
- ・ Mavenの思想(Repo, COC)を引き継ぐ
- ・ Ant/Mavenの設定ファイルをインポート可能
=> 過去の資産を無駄にしない



2000

2004

2012

ビルドスクリプトの比較



```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>MavenSample</groupId>
  <artifactId>MavenSample</artifactId>
  <version>1.0</version>
  <properties>
    <project.build.sourceEncoding>utf-8</project.build.sourceEncoding>
  </properties>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>2.3.2</version>
        <configuration>
          <source>1.7</source>
          <target>1.7</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```



```
apply plugin: 'java'
group = 'MavenSample'
version = '1.0'
artifactId = 'MavenSample'
sourceCompatibility = 1.7
targetCompatibility = 1.7
def defaultEncoding = 'UTF-8'
[compileJava, compileTestJava]*.options*.encoding
    = defaultEncoding

repositories{
    mavenCentral()
}
dependencies{
    testCompile 'junit:junit:4.11'
}
```



Antでは

- ・ソース格納場所の指定
- ・ビルドタスクの定義
- ・依存関係解決は別ファイルに記述

などにより、maven以上の記述量になるため割愛

2013-03-12 [Gradle] GradleとMavenとAnt+ivyの比較

=> http://d.hatena.ne.jp/kaakaa_hoe/20130312

インストール

必要要件

- ・ **JDK5以上** (Gradle2.0からはJDK6以上)
 - => 環境変数 “JAVA_HOME” で指定されたJavaを使う
 - => Groovyのインストールは必要ない

インストール

- ・ ダウンロード

=> <http://www.gradle.org/downloads>

- ・ 解凍

- ・ PATHを通す

=> 環境変数“GRADLE_HOME”を作っておくとアップデートが楽

Gradle wrapper

- Gradle実行環境を作成するスクリプト

=> ビルド実行に必要なファイルを自動でダウンロードするスクリプトを生成

=> チーム内でのGradleバージョン統一 / ビルドマシンでのGradleの使用 などに

=> 第61章 Gradleラッパー

http://gradle.monochromeroad.com/docs/userguide/gradle_wrapper.html

1. Gradle Wrapper作成

```
$ cat build.gradle
task wrapper(type: Wrapper) {
    gradleVersion = '1.12'
}

$ gradle wrapper
```

=> ラッパースクリプトと、その実行に必要なファイル群が生成される

2. Wrapper実行

```
$ gradlew build
```

=> Gradleの実行に必要なファイル群をDL
(初回実行時は数分かかる)

プロキシの設定

- ・ 社内からGradleを使用する時は必要

=> MavenCentralからライブラリをダウンロードする使い方が一般的

gradle.properties

=> カレントかホームディレクトリに置いておく

```
systemProp.proxySet=true  
systemProp.http.proxyHost=proxy.hogehoge.com  
systemProp.http.proxyPort=8080  
systemProp.http.proxyUser=kaakaa  
systemProp.http.proxyPassword=pass  
systemProp.http.nonProxyHosts=127.0.0.1|localhost
```

=> Gradleでプロキシの設定ってどうやるの？ - みちしるべ

<http://orangeclover.hatenablog.com/entry/20111207/1323184826>

=> 第20章 ビルド環境

http://gradle.monochromeroad.com/docs/userguide/build_environment.html



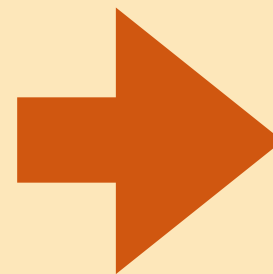
Javaビルド

タスク定義の基本

タスクの定義

build.gradle

```
task hello(description: 'sample') << {  
    println "Hello Gradle!"  
}
```



```
$ gradle hello
```

```
:hello  
Hello Gradle!
```

```
BUILD SUCCESSFUL
```

```
Total time: 5.997 secs
```

タスクの定義（依存関係）

build.gradle

```
task hello << {  
    println "Hello Gradle!"  
}  
task greet << {  
    println "Bye Gradle!"  
}  
  
greet.dependsOn hello
```



```
$ gradle greet -q  
Hello Gradle!  
Bye Gradle!
```

quietオプション

タスクの定義(色々なタスク)

build.gradle

```
task myCopy(type: Copy){  
    from 'resource'  
    into 'target'  
    include '**/*.txt'  
  
    logger.quiet 'Copy .txt files'  
}
```

ファイルコピータスク

デフォルトで利用できるロガー



重複していなければ、タスク名の
頭文字でタスクを指定可能

```
$ gradle mC -q  
Copy .txt files
```

=> 第15章 タスク詳解

http://gradle.monochromeroad.com/docs/userguide/more_about_tasks.html

タスクの確認

- 実行可能なタスクの確認コマンド

```
$gradle tasks -q
```

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

```
Build Setup tasks  
-----
```

```
init - Initializes a new Gradle build. [incubating]  
wrapper - Generates Gradle wrapper files. [incubating]
```

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

```
dependencies - Displays all dependencies declared in root project '03_task'.  
dependencyInsight - Displays the insight into a specific dependency in root project  
'03_task'.  
  
...
```

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

```
hello - sample task
```

カレントディレクトリで
実行できるタスク一覧

自分で定義したタスクの
説明は description で
していしたもの

Javaビルド (基本)

Javaビルド

- Javaプラグインを使う

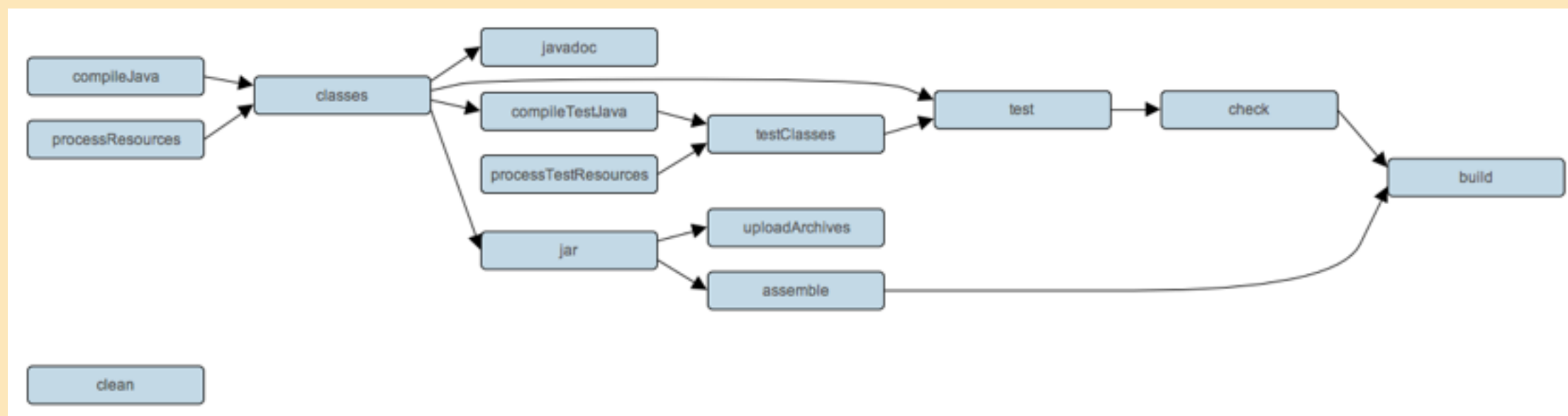
=> コンパイルやjar生成のタスクを記述しなくても良い (COC)

- COCにより基本的なタスク・設定は存在する

=> ソースの格納場所はsrc/main/java. テストソースはsrc/test/java

=> jar / test / build などのデフォルトタスク

Javaプラグインにより追加されるタスク群の依存関係



Javaビルド


- ・ 初回は **Build Init Plugin** が便利

=> 最小限のプロジェクトを自動生成してくれる

=> Chapter 47. Build Init Plugin

http://gradle.monochromeroad.com/docs/userguide/build_init_plugin.html

```
$ gradle init --type java-library
```

- 
- ・ java-library
 - ・ groovy-library
 - ・ scala-library
 - ・ 各言語の動作する最小プロジェクト
 - ・ basic
 - ・ Gradle関連のファイルのみ

Javaビルド

build.gradle

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

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

```
dependencies {  
    compile 'org.slf4j:slf4j-api:1.7.5'  
    testCompile "junit:junit:4.11"  
}
```

Javaプラグイン使用宣言

“dependencies”で指定された成果物を
探索する成果物リポジトリとして
MavenCentralリポジトリを指定

ビルドに必要なライブラリの指定

compile : ソースビルド

testCompile : テストソースビルド

Javaビルド

```
$ gradle build
```

```
:compileJava
```

```
:processResources
```

```
:classes
```

```
:jar
```

```
:assemble
```

```
:compileTestJava
```

```
:processTestResources
```

```
:testClasses
```

```
:test
```

```
:check
```

```
:build
```

```
BUILD SUCCESSFUL
```

```
Total time: 7.049 secs
```

ソースコンパイル

jarファイル生成

テストソースコンパイル

テスト実行

チェック

build.gradleにcheckstyleやfindbugsプラグインを
適用するだけで、各チェックツールを実行してくれる

Javaビルド (発展)

SourceSets

- ・ ソースセットの場所をデフォルトから変更する

build.gradle

```
sourceSets {  
    main.java.srcDir "src"  
    test.java.srcDir "test"  
}
```

- ・ jar生成時のマニフェストを記述する

build.gradle

```
jar {  
    manifest {  
        attributes("Implementation-Title": "Gradle",  
                    "Implementation-Version": version)  
    }  
}
```

Dependencies

- ・ 依存しているライブラリを指定する

build.gradle

```
repositories {  
    mavenCentral()  
    ivy {  
        url "http://repo.mycompany.com/repo"  
        layout "maven"  
    }  
}  
  
dependencies {  
    compile 'org.hibernate:hibernate-core:  
            4.2.14.Final'  
    testCompile "junit:junit:4.11"  
}
```

← MavenCentral以外のリポジトリを指定

← 依存ライブラリは
\$GroupId : \$artifactId : \$version
の形式で指定する

Dependencies

- ・ プロジェクトが依存しているライブラリの確認コマンド

```
$gradle dependencies -q
```

```
-----  
Root project  
-----
```

```
archives - Configuration for archive artifacts.  
No dependencies
```

```
compile - Compile classpath for source set 'main'.
```

```
\--- org.hibernate:hibernate-core:4.2.14.Final
```

```
    +--- antlr:antlr:2.7.7
```

```
    +--- org.jboss.logging:jboss-logging:3.1.0.GA
```

```
    +--- dom4j:dom4j:1.6.1
```

```
    +--- org.javassist:javassist:3.18.1-GA
```

```
    +--- org.jboss.spec.javax.transaction:jboss-transaction-api_1.1_spec:1.0.1.Final
```

```
    +--- org.hibernate.javax.persistence:hibernate-jpa-2.0-api:1.0.1.Final
```

```
\--- org.hibernate.common:hibernate-commons-annotations:4.0.2.Final
```

```
    \--- org.jboss.logging:jboss-logging:3.1.0.CR2 -> 3.1.0.GA
```

```
...
```

このプロジェクトは
hibernate-core に依存

hibernate-core が
依存してるライブラリ群

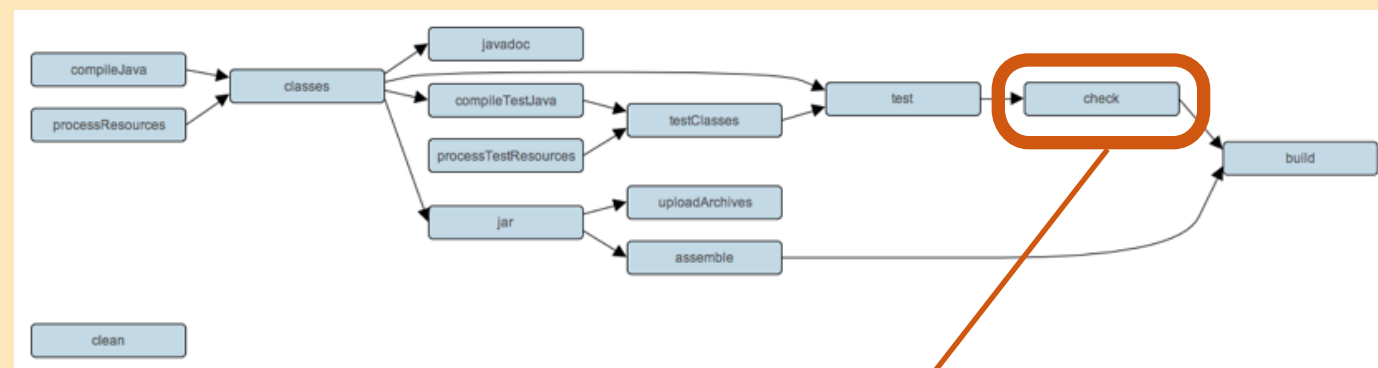
静的解析ツール

- Java開発でお馴染みのチェックツール群も簡単に使える

build.gradle

```
apply plugin: 'checkstyle'
apply plugin: 'findbugs'
apply plugin: 'pmd'
apply plugin: 'jdepend'
```

```
// チェックエラーでビルド失敗にしないため
[Checkstyle, FindBugs, Pmd].each { type ->
    tasks.withType(type) {
        ignoreFailures = 'true'
    }
}
```



各ツールのチェックタスクがJavaプラグインの
checkタスク加わる

※ checkstyleのみ規約ファイルが必要
=> \$ROOT/config/checkstyle/checkstyle.xml

便利な機能



IDE系 プラグイン

IDE系プラグイン

- 各IDEのプロジェクトファイルを生成する

=> 第38章 Eclipse プラグイン

http://gradle.monochromeroad.com/docs/userguide/eclipse_plugin.html

=> 第39章 IDEAプラグイン

http://gradle.monochromeroad.com/docs/userguide/idea_plugin.html

build.gradle

```
apply plugin: 'groovy'  
apply plugin: 'eclipse'  
apply plugin: 'idea'
```

```
$ gradle eclipse  
$ gradle idea
```

- どの言語のプラグインと一緒に利用するかで生成される内容が異なる
- 各々の好きなIDEで開発できる
- Build init pluginと併用するとプロト作成が捗る

既存ビルドツール との連携

Antタスクのインポート

- ・ build.xmlをGradleのタスクとして使える

=> 第17章 GradleからAntを使う

<http://gradle.monochromeroad.com/docs/userguide/ant.html>

build.gradle

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

Mavenプロジェクトの変換

- ・ pom.xmlをbuild.gradleに変換する

=> 47.3.1. "pom" (Maven conversion)

http://www.gradle.org/docs/current/userguide/build_init_plugin.html

```
$ gradle init --type pom
```

- ・ POMファイル生成やMavenデプロイが出来るプラグイン

=> 第52章 Mavenプラグイン

http://gradle.monochromeroad.com/docs/userguide/maven_plugin.html

=> Chapter 65. Maven Publishing (new)

http://gradle.monochromeroad.com/docs/userguide/publishing_maven.html#publishing_maven:apply_plugin

実行系 プラグイン

Applicationプラグイン

- ・ プログラムの実行や配布形式の作成

=> 第45章 アプリケーション プラグイン

http://gradle.monochromeroad.com/docs/userguide/application_plugin.html

build.gradle

```
apply plugin: 'application'
mainClassName = 'org.sample.Greet'
```

※ distZip / distTarタスクを実行すると、
依存ライブラリと起動スクリプトを含んだ
圧縮ファイルが生成される

```
$ gradle run
:compileJava
:processResources
:classes
:run
Hello application plugin

BUILD SUCCESSFUL

Total time: 8.837 secs
```

マルチプロジェクト

マルチプロジェクトの定義

- Gradleではマルチプロジェクトを簡単に定義できる

=> 第56章 マルチプロジェクトのビルド

http://gradle.monochromeroad.com/docs/userguide/multi_project_builds.html

settings.gradle

```
includeFlat 'ProjectA'
include 'ProjectB'
```

ビルド対象の
ディレクトリ名を指定

build.gradle

```
subprojects {
    apply plugin: 'java'
    repositories { mavenCentral() }
    dependencies {
        testCompile 'junit:junit:4.11'
    }
}
```

ProjectA / Project B
共通な設定を記述
(固有な設定は
project(:ProjectA) { ~ }
のように記述する)

```
├── ProjectA
│   └── src
└── Build
    ├── build.gradle
    ├── settings.gradle
    └── ProjectB
        └── src
```

マルチプロジェクトの実行

```
$ gradle build -q
```

```
:ProjectA:compileJava
```

```
:ProjectA:processResources UP-TO-DATE
```

```
...
```

```
:ProjectA:check
```

```
:ProjectA:build
```

```
:ProjectB:compileJava UP-TO-DATE
```

```
:ProjectB:processResources UP-TO-DATE
```

```
...
```

```
:ProjectB:check UP-TO-DATE
```

```
:ProjectB:build
```

```
BUILD SUCCESSFUL
```

ProjectAのビルドタスクが
実行される

ProjectBのビルドタスクが
実行される

※ProjectA/Bにファイルを
追加することなく、Gradleの
ビルドタスクを実行できる
(成果物は各プロジェクト内に
生成される)

まとめ



Gradleまとめ

- ・ 記述量が少なく、かつ細かな設定も可能

=> Ant/Mavenに比べるとビルドスクリプトの見通しが良くなる

- ・ 結構なんでもできる

=> Androidアプリ / マルチプロジェクトなど、他にも多数のトピック

=> 最近のアップデートでは cppプロジェクト関連の機能もよく追加されてる

- ・ COCによる暗黙の理解が増える

=> Groovyもそこまで市民権を得てる訳ではない…

=> 結局ビルド職人はなくなる？

参考資料

Gradle関連の書籍



Gradle in Action (2014/3/9)

- ・ 基本的な事柄からマルチプロジェクトビルドや継続的デリバリーなどについても言及あり
- ・ コードや図も多くて見やすいので英語苦手でも行けるかも



Gradle Effective Implementation Guide (2012/10/25)

- ・ 依存関係やマルチプロジェクトビルドについて、さらには継続的インテグレーションやGradleプラグインの作成方法などについての包括的な記述がある

Gradle Beyond the Basics (2013/7/16)

Building and Testing with Gradle (2011/7/16)

- ・ 日本語の書籍も執筆中との話…
- ・ Androidビルドについて書かれた書籍はまだない？

参考サイト

- ・ 公式のユーザガイドが充実してる

<http://www.gradle.org/documentation>

=> 有志による邦訳 : <http://gradle.monochromeroad.com/docs/>

- ・ 日本語資料はJGGUGさん周りを…

=> Gradle入門 - Qiita

<http://qiita.com/vvakame/items/83366fbfa47562fafbf4>

=> nobusue/GradleHandson - Github

<https://github.com/nobusue/GradleHandson>

終