

Gradle – lepszy system budowania projektów

Tomasz Kaczanowski

<http://kaczanowscy.pl/tomek>

Java Camp #3, Kraków, Kwiecień 2010



UWAGA

Wszystkie przykłady kodu zamieszczone w tej prezentacji działają poprawnie z Gradle w wersji **0.9-preview-1**. Istnieją spore szanse że będą w 100% zgodne z wersją stabilną 0.9.

Biorąc pod uwagę rozwój Gradle i planowane zmiany, jest jednak możliwe że nie będą one działać poprawnie w wersjach kolejnych.

Założenie

- Nie ma prostych buildów

Założenie

- **Nie ma prostych buildów**
 - Testy (wszelkiego rodzaju)
 - Stawianie środowiska
 - Dopalacze
 - AspectJ, Retrotranslator,
 - Deployment (releasy)
 - Do repozytorium artefaktów
 - Na WebDavs
 - Tagi, branche, etc.
 - Wersje testowe i produkcyjne
 - Wiele artefaktów
 - JAR/WAR/EAR/bundle + source, zip, tar.gz
 - pliki konfiguracyjne / deployment descriptors
 - Raporty, dokumentacja
 - PMD, Cobertura, Findbugs, ...
 - JVM to wiele języków
- **Automatyzacja jest nieodzowna**

Plan gry

- Podstawowe informacje o Gradle
- Konkurencja
- Opis Gradle – teoria + przykłady
 - Podstawy
 - Command line
 - Współpraca z Antem
 - Dependencje
 - Własna logika
 - Konfiguracja
 - Projekty wielomodułowe
 - Własna logika w buildzie
 - Różności
- Konkluzje
- Pytania

Gradle

- Open-source, licencja Apache
- Repo: git
- Project manager – Hans Docter
- Pierwszy release – kwiecień 2008
- Obecna stabilna wersja 0.9-preview-1
 - Marzec 2010
- <http://gradle.org>
- <http://gradle.biz>

Gradle

- Open-source, licencja Apache
- Repo: git
- Project manager – Hans Docter
- Pierwszy release – kwiecień 2008
- Obecna stabilna wersja 0.9-preview-1
 - Marzec 2010

- <http://gradle.org>
- <http://gradle.biz>

- UserGuide > 200 stron PDF
- Cookbook
- Prężna społeczność, aktywna lista mailingowa (user i dev)
- Wiki

Gradle

- Elastyczne narzędzie do budowania projektów
- Oparte na idei build-by-convention
- Skrypty pisane w Groovym (DSL)
- *making the impossible possible, the possible easy, and the easy elegant*

-- Moshé Feldenkrais

Gradle – lista projektów

- Grails <http://www.grails.org/>
- Mockito <http://www.mockito.org/>
- GPars (was GParallelizer) <http://gpars.codehaus.org>
- GWT Pectin <http://code.google.com/p/gwt-pectin/>
- Spock <http://code.google.com/p/spock/>
- GMOCK <http://code.google.com/p/gmock>
- ...
- + wdrożenia komercyjne

Konkurencja

- Ant
- Maven
- Buildr, Gant, ... ?

Ant

- Za
 - Elastyczność
 - Masz nad nim pełną kontrolę
 - Zależności między taskami
 - Mnóstwo dostępnych tasków

Ant

- Za
 - Elastyczność
 - Masz nad nim pełną kontrolę
 - Zależności między taskami
 - Mnóstwo dostępnych tasków

```
<target name="init">
```

```
...
```

```
</target>
```

```
<target name="compile"  
        depends="init">
```

```
...
```

```
</target>
```

Ant

- Za
 - Elastyczność
 - Masz nad nim pełną kontrolę
 - Zależności między taskami
 - Mnóstwo dostępnych tasków

```
<target name="init">
```

```
...
```

```
</target>
```

```
<target name="compile,,  
        depends="init">
```

```
...
```

```
</target>
```

BUnzip2, BZip2, Cab, Ear, Gunzip,
gzip, Jar, Jlink, Manifest, Rpm, SignJar,
Tar, ...

Attrib, Checksum, Chgrp, Chmod,
Chown, Concat, Copy, Copydir,
Copyfile, Delete, Deltree, Filter,
FixCRLF, Get, Mkdir, Move, ...

+ contrib

+ taski do uruchamiania różnych
narzędzi (PMD, Selenium, Cargo,
Cobertura,...)

Ant

- Za
 - Elastyczność
 - Masz nad nim pełną kontrolę
 - Zależności między taskami
 - Mnóstwo dostępnych tasków
- Przeciw
 - Brak build-by-convention
 - XML (nie ma typowania, zwracania wartości, trudno wyrazić logikę builda itp.)
 - Brak wsparcia dla multi-module builds
 - Przegadane pliki build.xml

Ant – przykład - if

```
<condition property="isMacOsButNotMacOsX">
  <and>
    <os family="mac"/>
    <not>
      <os family="unix"/>
    </not>
  </and>
</condition>
```

sets the property `isMacOsButNotMacOsX` if the current operating system is MacOS, but not MacOS X - which Ant considers to be in the Unix family as well.

Ant – przykład - if

```
<condition property="isMacOsButNotMacOsX">
  <and>
    <os family="mac"/>
    <not>
      <os family="unix"/>
    </not>
  </and>
</condition>
```

```
If (System.getProperty("os.name") ... .) {
  ...
} else {
  ...
}
```

Maven

- Za
 - Layout projektu
 - Build-by-convention
 - Wiele dostępnych pluginów

Maven

- Za
 - Layout projektu
 - Build-by-convention
 - Wiele dostępnych pluginów
- Przeciw
 - Convention-**instead of**-configuration
 - Sztywny lifecycle,
 - Dodawanie własnej logiki do buildów jest trudne,
 - Jak napisać własnego plugina ?
 - Jak napisać IF ?
 - Brak dostępu do Javy
 - pom.xml są przegadane – XML !
 - **frameworkitis**

Maven

- Za
 - Layout projektu
 - Build-by-convention
 - Wiele dostępnych pluginów
- Przeciw
 - Convention-**insteadof**-configuration
 - Sztywny lifecycle,
 - Dodawanie własnej logiki do buildów jest trudne,
 - Jak napisać własnego plugina ?
 - Jak napisać IF ?
 - Brak dostępu do Javy
 - pom.xml są przegadane – XML !
 - **frameworkitis**
- ?
 - Wsparcie dla multi-module

Maven – najpierw pomaga, a potem zaczyna szkodzić

- Frameworkitis – choroba polegająca na tym, że framework próbuje zrobić za dużo lub nie w taki sposób w jaki chciałbyś (ale nie możesz tego zmienić). Przyjemnie jest otrzymać tyle funkcjonalności za darmo, ale w pewnym momencie okazuje się, że aby móc uzyskać to co chcesz, musisz zmagać się z frameworkiem zamiast korzystać z niego. Na tym etapie zaczynasz przegrywać, bo ciężko jest zmusić framework do działania w kierunku nieprzewidzianym przez jego twórców.

(Własne, tendencyjne tłumaczenie na podstawie wywiadu z Erichem Gammą – www.artima.com)

Maven – najpierw pomaga, a potem zaczyna szkodzić

- Frameworkitis – choroba polegająca na tym, że framework próbuje zrobić za dużo lub nie w taki sposób w jaki chciałbyś (ale nie możesz tego zmienić). Przyjemnie jest otrzymać tyle funkcjonalności za darmo, ale w pewnym momencie okazuje się, że aby móc uzyskać to co chcesz, musisz zмагаć się z frameworkiem zamiast korzystać z niego. Na tym etapie zaczynasz przegrywać, bo ciężko jest zmusić framework do działania w kierunku nieprzewidzianym przez jego twórców.
- Framework vs. Toolkit
 - Toolkity są wolne od tej choroby, bo nie próbują przejąć kontroli (nie są mądrzejsze od Ciebie).
 - Frameworki przejmują kontrolę i mówią Ci co i kiedy masz wykonać. Toolkit dostarcza Ci elementów i daje swobodę w korzystaniu z nich.

(Własne, tendencyjne tłumaczenie na podstawie wywiadu z Erichem Gammą – www.artima.com)

Maven – najpierw pomaga, a potem zaczyna szkodzić

- Frameworkitis – choroba polegająca na tym, że framework próbuje zrobić za dużo lub nie w taki sposób w jaki chciałbyś (ale nie możesz tego zmienić). Przyjemnie jest otrzymać tyle funkcjonalności za darmo, ale w pewnym momencie okazuje się, że aby móc uzyskać to co chcesz, musisz zмагаć się z frameworkiem zamiast korzystać z niego. Na tym etapie zaczynasz przegrywać, bo ciężko jest zmusić framework do działania w kierunku nieprzewidzianym przez jego twórców.

(Własne, tendencyjne tłumaczenie na podstawie wywiadu z Erichem Gamma – www.artima.com)

A Gradle na to:

Our design decision is to embrace the unknown complexity of the enterprise and enable our users to do what they think is right. So we always looking for new use cases to work on that vision.

Hans Docter, Gradle mailing list

Narzędzia do budowania – czego chcemy

- Ant
 - Elastyczność
 - Zależność między taskami
 - Mieć kontrolę, pełna konfiguracja
 - Mnóstwo dostępnych tasków

Narzędzia do budowania – czego chcemy

- Ant
 - Elastyczność
 - Zależność między taskami
 - Mieć kontrolę, pełna konfiguracja
 - Mnóstwo dostępnych tasków
- Ivy
 - Zarządzanie dependencjami
- Buildr, Gant
 - DSL lepszy od XMLa

Narzędzia do budowania – czego chcemy

- Ant
 - Elastyczność
 - Zależność między taskami
 - Mieć kontrolę, pełna konfiguracja
 - Mnóstwo dostępnych tasków
- Ivy
 - Zarządzanie dependencjami
- Buildr, Gant
 - DSL lepszy od XMLa
- Maven
 - Convention-over-configuration
 - Multi-module build

Narzędzia do budowania – czego chcemy

- Ant
 - Elastyczność
 - Zależność między taskami
 - Mieć kontrolę, pełna konfiguracja
 - Mnóstwo dostępnych tasków
- Ivy
 - Zarządzanie dependencjami
- Buildr, Gant
 - DSL lepszy od XMLa
- Maven
 - Convention-over-configuration
 - Multi-module build
- Inne
 - Niezależność od platformy
 - Obsługa różnych języków (JVM)
 - Zgodność w tył
 - Szybkość
 - Łatwy do zrozumienia plik konfiguracyjny build
 - Przyjazny dla użytkownika

Gradle – pierwszy przykład

- command line
- zależności między taskami
- defaultTasks
- wyłączanie tasków

Współpraca z Antem – użycie tasków

- Wykorzystuje AntBuilder
- Taski Anta są „obywatelami pierwszej kategorii”
- W skrypcie builda dostępny obiekt ant

Współpraca z Antem – użycie tasków

```
task zip << {
    ant.zip(destfile: 'archive.zip') {
        fileset(dir: 'src') {
            include(name: '**.xml')
            exclude(name: '**.java')
        }
    }
}
```

Współpraca z Antem – użycie tasków

```
task zip << {
    ant.zip(destfile: 'archive.zip') {
        fileset(dir: 'src') {
            include(name: '**.xml')
            exclude(name: '**.java')
        }
    }
}
```

Podrasowane taski Anta

```
task dist(type: Zip) {
    from 'src/dist`
    from configurations.runtime
    into('libs`)
}
```

Współpraca z Antem – import build.xml

Współpraca z Antem – podsumowania

- Gradle to „Ant na dopalaczu”
- Pełna zgodność w tył
- Możliwość użycia tasków Anta (defaultowych i zewnętrznych)
- Możliwość importu build.xml

Dependencje – pełne wsparcie dla Mavena

- download

```
repositories {  
    mavenCentral()  
    mavenRepo urls: 'http://download.java.net/maven/2/'  
}
```

Dependencje – pełne wsparcie dla Mavena

- **download**

```
repositories {
    mavenCentral()
    mavenRepo urls: 'http://download.java.net/maven/2/'
}
```

- **upload**

```
dependencies {
    deployerJars "org.apache.maven.wagon:wagon-ssh:1.0-beta-2"
}

uploadArchives {
    repositories.mavenDeployer {
        configuration = configurations.deployerJars
        repository(url: "scp://myrepo.com/releases") {
            authentication(userName: "me", password:
                "myPassword")
        }
    }
}
```

Dependencje – Apache Ivy in action

- Łączenie różnych typów repozytoriów

```
repositories {  
    mavenCentral()  
    flatDir name: 'localRepository', dirs: 'lib'  
}
```

- Dowolny „pattern” repozytorium

```
// Maven2 layout  
someroot/[organisation]/[module]/[revision]/[module]-  
[revision].[ext]  
  
// Typical layout for an ivy repository  
someroot/[organisation]/[module]/[revision]/[types]/[a  
rtifact].[ext]  
  
//Simple layout (the organization is not used, no  
nested folders.)  
someroot/[artifact]-[revision].[ext]
```

Gradle – przykład dependencje

Gradle – przykład webapp

- Convention over configuration
 - Nic nie musisz pisać, a Gradle i tak wie co robić
 - Ale konfiguracja zawsze możliwa i prosta do uzyskania

Gradle – przykład webapp

- Convention over configuration
 - Nic nie musisz pisać, a Gradle i tak wie co robić
 - Ale konfiguracja zawsze możliwa i prosta do uzyskania

```
`-- src
  |-- main
  |   |-- groovy    // kod źródłowy pisany w Groovym
  |   |-- java     // kod źródłowy pisany w Javie
  |   |-- resources // zasoby aplikacji
  |   `-- webapp   // kod źródłowy aplikacji
webowej
  `-- test
      |-- groovy    // testy napisane w Groovym
      |-- java     // testy napisane w Javie
      `-- resources // zasoby testowe
```

Gradle – przykład webapp

- Tworzenie artefaktów Mavena

```
apply plugin: 'maven'
```

```
configure(install.repositories.mavenInstaller) {  
    pom.project {  
        version '1.0-Maven'  
        groupId 'myGroup'  
        artifactId 'myArtifact'  
    }  
}
```

```
gradle install
```

Gradle – przykład projekt wielomodułowy

- Dowolna struktura katalogów
 - settings.gradle
- Dependencje pomiędzy projektami i taskami

```
|  
|--build.gradle  
|--core  
|--ui  
    |--swing  
    |--web
```

Gradle – przykład projekt wielomodułowy

- Dowolna struktura katalogów
 - settings.gradle
- Dependencje pomiędzy projektami i taskami

```
|
|--build.gradle
|--core
|--ui
    |--swing
    |--web
        |--build.gradle
```

```
project (':ui:web') {
    apply plugin: 'war'
    apply plugin: 'war'
}
```

Gradle – przykład projekt wielomodułowy

- Sprytne budowanie projektów:
 - `build` – buduje wszystko co potrzeba, ale testuje tylko bieżący projekt
 - `buildNeeded` – buduje i testuje wszystkie potrzebne projekty
 - `-a build` – nie buduje potrzebnych projektów, bierze je z cache
 - `buildDependent` – buduje projekt i wszystkie od niego zależne (wraz z testami)

Pluginy - Gradle

- Java
- Groovy
- Scala
- War
- Jetty
- Maven
- Code Quality
- OSGi
- Eclipse
- Project Report

- IDE: Eclipse, IntelliJ
- CI: Hudson

- Wszystkie taski Anta (PMD, Findbugs, ...)
- Pluginy pisane przez użytkowników

Brak mi pluginów ☹ - Ant !

```
configurations {
    findbugsConf
}

dependencies {
    findbugsConf 'net.sourceforge.findbugs:findbugs:1.3.2',
    'net.sourceforge.findbugs:findbugs-ant:1.3.2'
}

task findbugs << {
    println 'Running findbugs static code analysis'
    ant {
        taskdef(name:'findbugs',
            classname:'edu.umd.cs.findbugs.anttask.FindBugsTask', classpath:
            configurations.findbugsConf.asPath)

        findbugs(home: '/home/tomek/bin/findbugs', output:'xml') {
            sourcePath(path: 'src/main/java')
            "class"(location: 'build/libs/findbugs-unspecified.jar')
        }
    }
}
```

Brak mi pluginów ☹ - ale pojawiają się i będą !

```
apply
  url:'http://github.com/breskeby/gradleplugins
    /raw/0.9-
    upgrade/aspectjPlugin/aspectJ.gradle'
```

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

```
dependencies{
  ajc "aspectj:aspectjtools:1.5.3"
  compile "aspectj:aspectjrt:1.5.3"
}
```

Logika w skrypcie budowania - zewnętrzne biblioteki

```
import org.apache.commons.codec.binary.Base64

buildscript {
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath group: 'commons-codec', name:
'commons-codec', version: '1.2'
    }
}

task encode << {
    def byte[] encodedString = new
Base64().encode('hello world\n' as byte[])
    println new String(encodedString)
}
```

Gradle – przykład – własny dodatkowy kod

- Jak uniknąć zaśmiecenia kodu build.gradle ?
 - W build.gradle umieść tylko wywołanie własnych tasków (ich konfiguracje)
 - Implementacja własnych tasków w katalogu buildSrc (src/main/groovy)

Zgodność w tył

- Repozytoria Mavena (i szerzej – Ivy)
 - Layout projektu
 - Taski Anta
 - Import build.xml Anta
-
- A co z importowaniem pom.xml ?

DAG, operowanie na grafie tasków

```
task release(dependsOn: assemble) << {  
    println 'We release now'  
}
```

```
build.taskGraph.whenReady { taskGraph ->  
    if (taskGraph.hasTask(':release')) {  
        version = '1.0'  
    } else {  
        version = '1.0-SNAPSHOT'  
    }  
}
```

Generowanie tasków

- Możliwe dynamiczne generowanie tasków
- **Przykład**
- Praca podzielona na WP (*work-package*) (1-5)
- Zadanie:
 - Deployować bundle z testami dla poszczególnych WP na frameworku OSGi
 - Każdy bundle posiada aktywator odpalający testy
- Chcę móc napisać:

```
gradle testWp1 testWp3 testWp5
```
- albo

```
gradle testWps
```

Generowanie tasków

```
testWp1 (description: "run Wp1 tests") << {  
  copy {  
    from configurations.testJars  
    into deployDir  
    include "wp1-integration-tests-*.jar"  
  }  
}  
testWps.dependsOn testWp1
```

Generowanie tasków

```
[1, 2, 3, 4, 5].each { wpsNb ->
  task "testWp${wpsNb} „ (description: "run
    Wp${wpsNb} tests") << {
    copy {
      from configurations.testJars
      into wkiDeployDir
      include "wp${wpsNb}-integration-
tests-*.jar„
    }
  }
testWps.dependsOn "testWp${wpsNb} "
}
```

Generowanie tasków

```
testWp1 (description: "run Wp1 tests") << {  
    copy {  
        from configurations.testJars  
        into deployDir  
        include "wp1-integration-tests-*.jar"  
    }  
}  
testWps.dependsOn "testWp1"
```

```
testWp2 (description: "run Wp2 tests") << {  
    copy {  
        from configurations.testJars  
        into deployDir  
        include "wp2-integration-tests-*.jar"  
    }  
}  
testWps.dependsOn "testWp2"
```

```
gradle testWp1 testWp4
```

```
gradle testWps
```

Czy to jest proste ?

- *making the impossible possible, the possible easy, and the easy elegant*

-- Moshé Feldenkrais

- Magia Groovy + Gradle
- Brak wiedzy o tym „jak należy to zrobić”
- TIMTWOTDI

A jak z szybkością ?

- Czy Gradle jest wolniejszy?
- Dla małego builda – tak
 - Groovy
 - Up-to-date checking
 - Budowa grafu tasków
 - Wołanie tasków Anta

A jak z szybkością ?

- Czy Gradle jest wolniejszy?
- Dla małego builda – tak
 - Groovy
 - Up-to-date checking
 - Budowa grafu tasków
 - Wołanie tasków Anta
- Dla wielu projektów – nie
 - -a build
 - smart skipping (-x)
- Obiecanki
 - Zrównoleglenie odpalania testów

Czy to jest proste ?

- *making the impossible possible, the possible easy, and the easy elegant*

-- Moshé Feldenkrais

- Magia Groovy + Gradle
- Brak wiedzy o tym „jak należy to zrobić”
- TMTWOTDI

- DSL wciąż rozwijany
- Cookbook się wypełnia
- Pluginów przybywa

Ale ja nie mam zainstalowanego Gradle...

Gradle is a new tool. You can't expect it to be installed on machines beyond your sphere of influence. An example are continuous integration server where Gradle is not installed and where you have no admin rights for the machine. Or what if you provide an open source project and you want to make it as easy as possible for your users to build it?

Ale ja nie mam zainstalowanego Gradle...

- gradle wrapper
 - Ściąga Gradle z sieci
 - Wywoływany identycznie jak gradle (te same parametry)
- Gradle jest budowany Gradlem (wrapperem)

Building Pectin from source

Pectin is built using the Gradle build system. It's built using version 0.8 for which you can find the user guide [here](#).

Please Note: You don't need to install gradle to build the project. The project uses a Gradle provided wrapper that automatically downloads the correct version for you.

Narzędzia do budowania – czego chcemy

- Ant
 - Elastyczność
 - Zależność między taskami
 - Mieć kontrolę, pełna konfiguracja
 - Mnóstwo dostępnych tasków
- Ivy
 - Zarządzanie dependencjami
- Buildr, Gant
 - DSL lepszy od XMLa
- Maven
 - Convention-over-configuration
 - Multi-module build
- Inne
 - Niezależność od platformy
 - Obsługa różnych języków (JVM)
 - Zgodność w tył
 - Szybkość
 - Łatwy do zrozumienia plik konfiguracyjny build
 - Przyjazny dla użytkownika

Podsumowanie

- Jeszcze nie 1.0, ale już używalny produkcyjnie
- Masa przydatnych featurów,
 - a Gradle dopiero się rozpędza !
- Jest tu co robić – dołącz !
- Ant nie przystaje do dzisiejszych czasów
- Maven zbyt sztywny
- Czy to ma szanse powodzenia ?
 - Jest fajny, ale czy to wystarczy ?
 - „Momentum”
 - Wymaga „mental switch”

Linki

- <http://gradle.org>
- <http://gradle.biz>
- <http://docs.codehaus.org/display/GRADLE/Cookbook>
- <http://docs.codehaus.org/display/GRADLE/Releases>
- <http://groovy.codehaus.org/Using+Ant+from+Groovy>

**Kolejny projekt
buduj Gradlem.**

