



Gradle

Miłosz Filus

ZTI 1.06.2020



Czym jest gradle?

Narzędzie do
zautomatyzowania procesu
budowania i zarządzania
projektem



Zalety

- Cache'owanie plików wyjściowych, bibliotek
- Wielowątkowa egzekucja
- Wielowątkowe pobieranie zależności
- Wykrywanie zmian w drzewie zadań
- Metryki dla wykonanych buildów
- i wiele wiele więcej



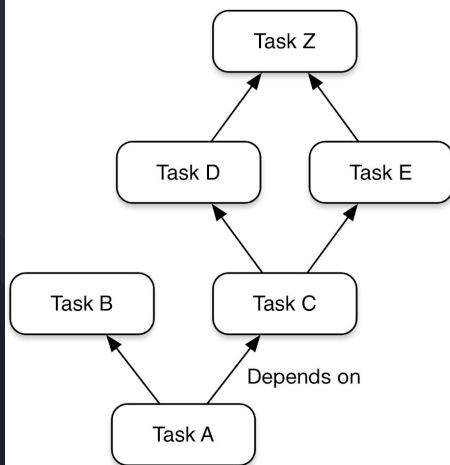
Aspekty Gradle 1/5

Możliwość budowania
praktycznie dowolnego
oprogramowania
C++/Java/Swift

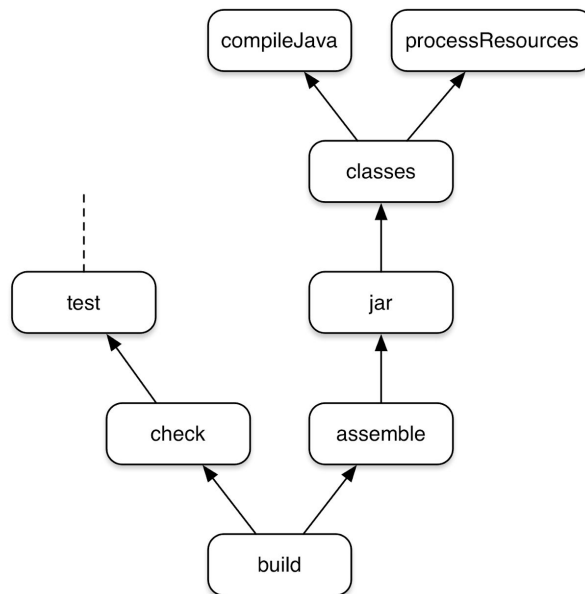
Mechanizm pluginów oraz
wchodzących w ich skład zadań

Aspekty Gradle 2/5

Generic task graph



Partial task graph for a standard Java build



Zadania - cały proces budowania jest oparty na stworzonych przez nas lub zawartych w pluginach zadaniach



Aspekty Gradle 3/5

Fazy budowania

- Inicjalizacja
- Konfiguracja
- Egzekucja

Aspekty Gradle 4/5

Rozszerzalność

- Własne zadania
- Dodatkowe właściwości
- Własne konwencje / model



Aspekty Gradle 5/5

Gradle API

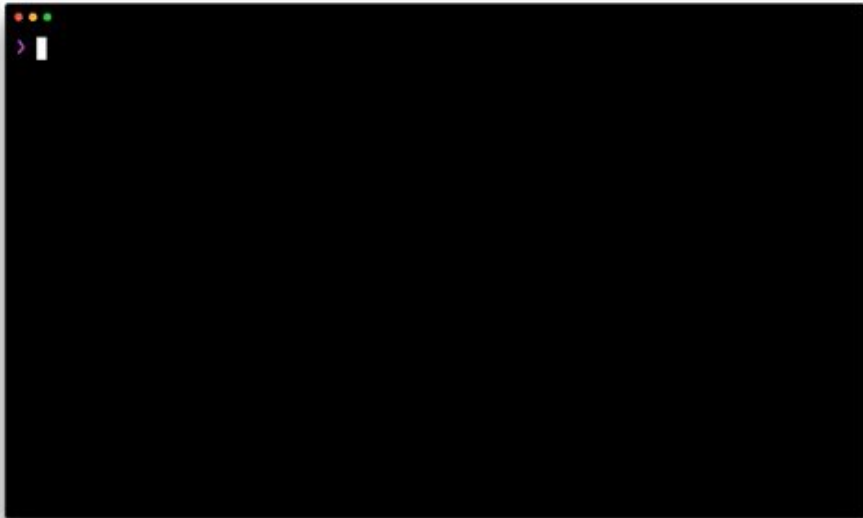


Czemu gradle?

- Łatwy w obsłudze
- Wysoce konfigurowalny
- Szybki
- POTĘŻNY
- Korzysta z języków programowania Kotlin / Groovy przez co nie trzeba używać XMLa



Gradle vs Maven



(Groovy | Kotlin) DSL (Domain-specific language)

Groovy to obiektowy język skryptowy wzorowany na składni Javy

Groovy może używać klas napisanych w Javie

Podobnie jak Kotlin, który może wywoływać klasy Javy

Przejrzystość i prostota

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5     <groupId>info.solidsoft.rnd</groupId>
6     <artifactId>spock-10-groovy-24-gradle-maven</artifactId>
7     <version>0.0.1-SNAPSHOT</version>
8     <properties>
9         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
10        <surefire.version>2.18.1</surefire.version>
11    </properties>
12    <build>
13        <plugins>
14            <plugin>
15                <groupId>org.codehaus.gmavenplus</groupId>
16                <artifactId>gmavenplus-plugin</artifactId>
17                <version>1.4</version>
18                <executions>
19                    <execution>
20                        <goals>
21                            <goal>compile</goal>
22                            <goal>testCompile</goal>
23                        </goals>
24                    </execution>
25                </executions>
26            </plugin>
27            <plugin>
28                <artifactId>maven-surefire-plugin</artifactId>
29                <version>${surefire.version}</version>
30                <configuration>
31                    <includes>
32                        <include>/**/*.Spec.java</include> <!-- Yes, .java extension -->
33                        <include>/**/*.Test.java</include> <!-- Just in case having "normal" JUnit tests -->
34                    </includes>
35                </configuration>
36            </plugin>
37        </plugins>
38    </build>
39    <dependencies>
40        <dependency>
41            <groupId>org.codehaus.groovy</groupId>
42            <artifactId>groovy-all</artifactId>
43            <version>2.4.1</version>
44        </dependency>
45        <dependency>
46            <groupId>org.spockframework</groupId>
47            <artifactId>spock-core</artifactId>
48            <version>1.0-groovy-2.4</version>
49            <scope>test</scope>
50        </dependency>
51    </dependencies>
52 </project>
```

pom.xml

```
1 apply plugin: 'groovy'
2
3 group = "info.solidsoft.rnd"
4 version = "0.0.1-SNAPSHOT"
5
6 repositories {
7     mavenCentral()
8 }
9
10 dependencies {
11     compile 'org.codehaus.groovy:groovy-all:2.4.1'
12     testCompile 'org.spockframework:spock-core:1.0-groovy-2.4'
13 }
14
15
16
17
18 rootProject.name = 'spock-10-groovy-24-gradle-maven'
```

build.gradle

settings.xml

maven

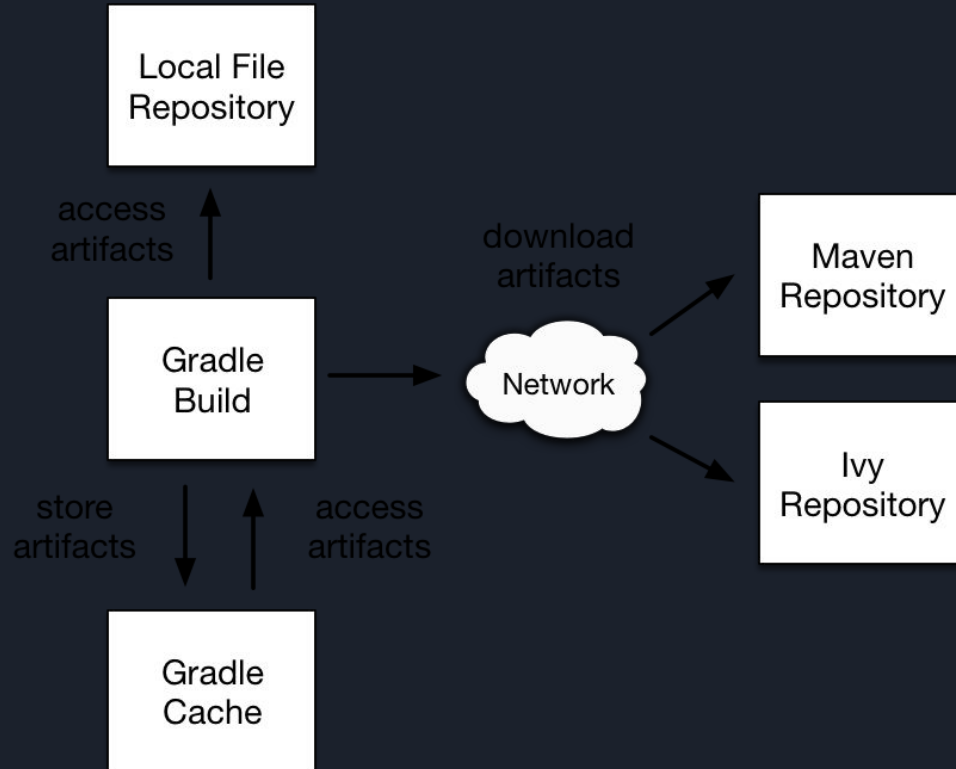




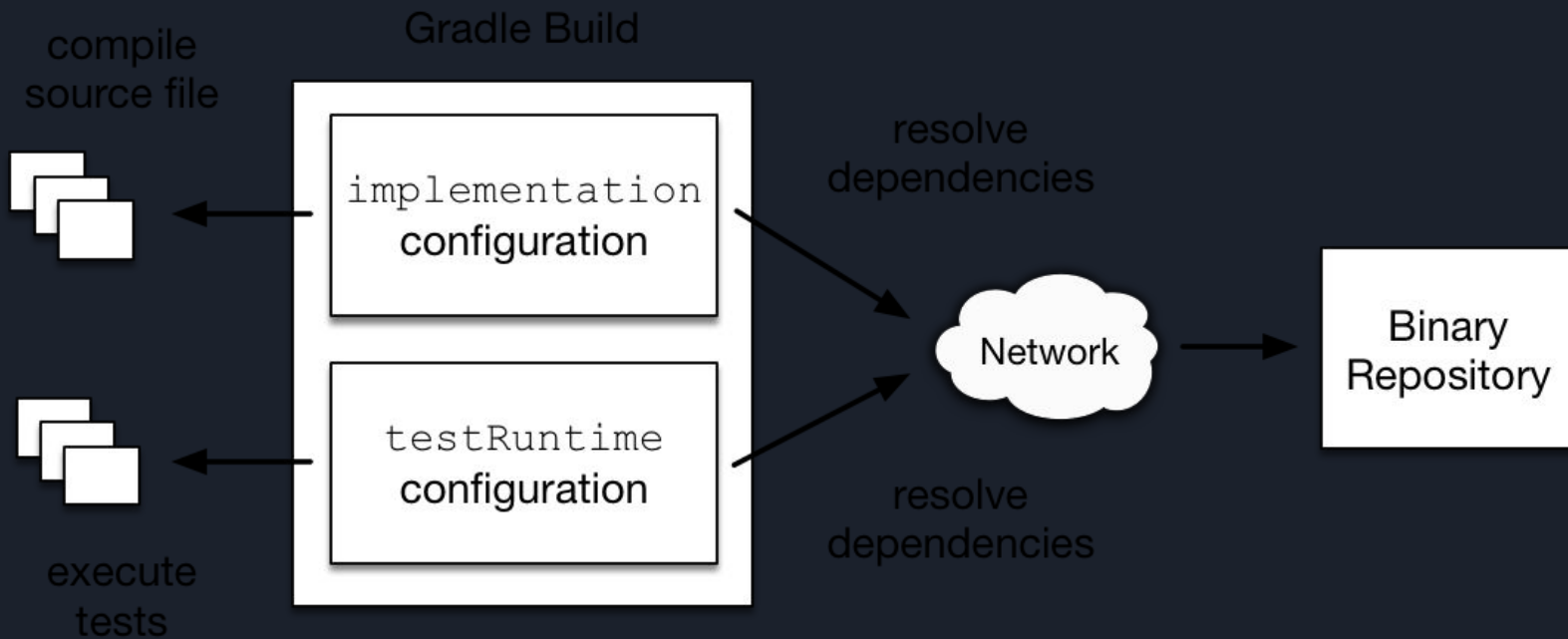
Mechanizm zapewniania zależności

- Repozytoria - czyli skąd
- Dependencje - czyli co

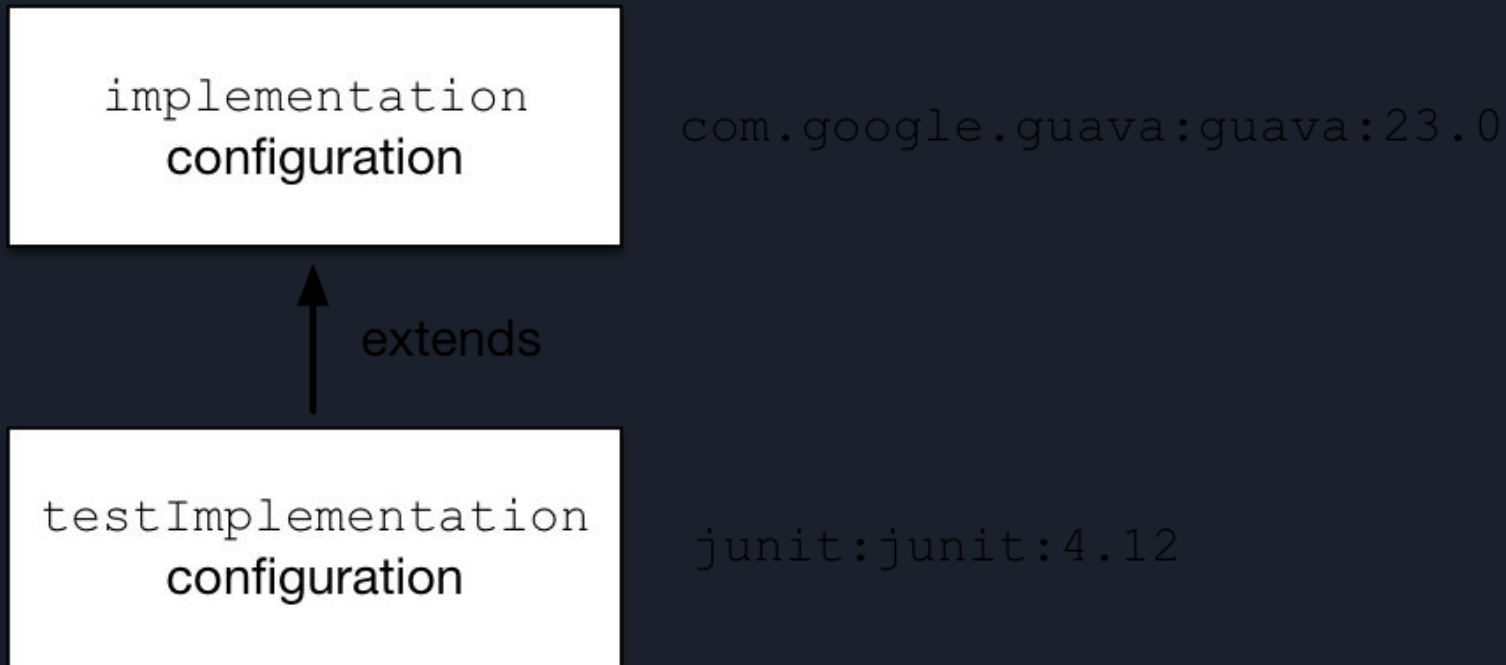
Menedžer dependencji



Dependencje 1/2



Dependencje 2/2





Główne pliki narzędzia Gradle

`settings.gradle` - reprezentuje architekturę projektu - jakie moduły wchodzi w skład projektu, jeden i globalny na cały projekt

`build.gradle` - poszczególny dla każdego modułu, może być również globalny, zawiera ustawienia, dependencje, repozytoria, pluginy

`properties.gradle` - nie jest tworzony domyślnie, posiada linie typu `klucz:wartość`, można w nim zdefiniować opcje przekazane do JVM'a, opcje do samego gradle'a czy też dane potrzebne do autoryzacji z bazą danych

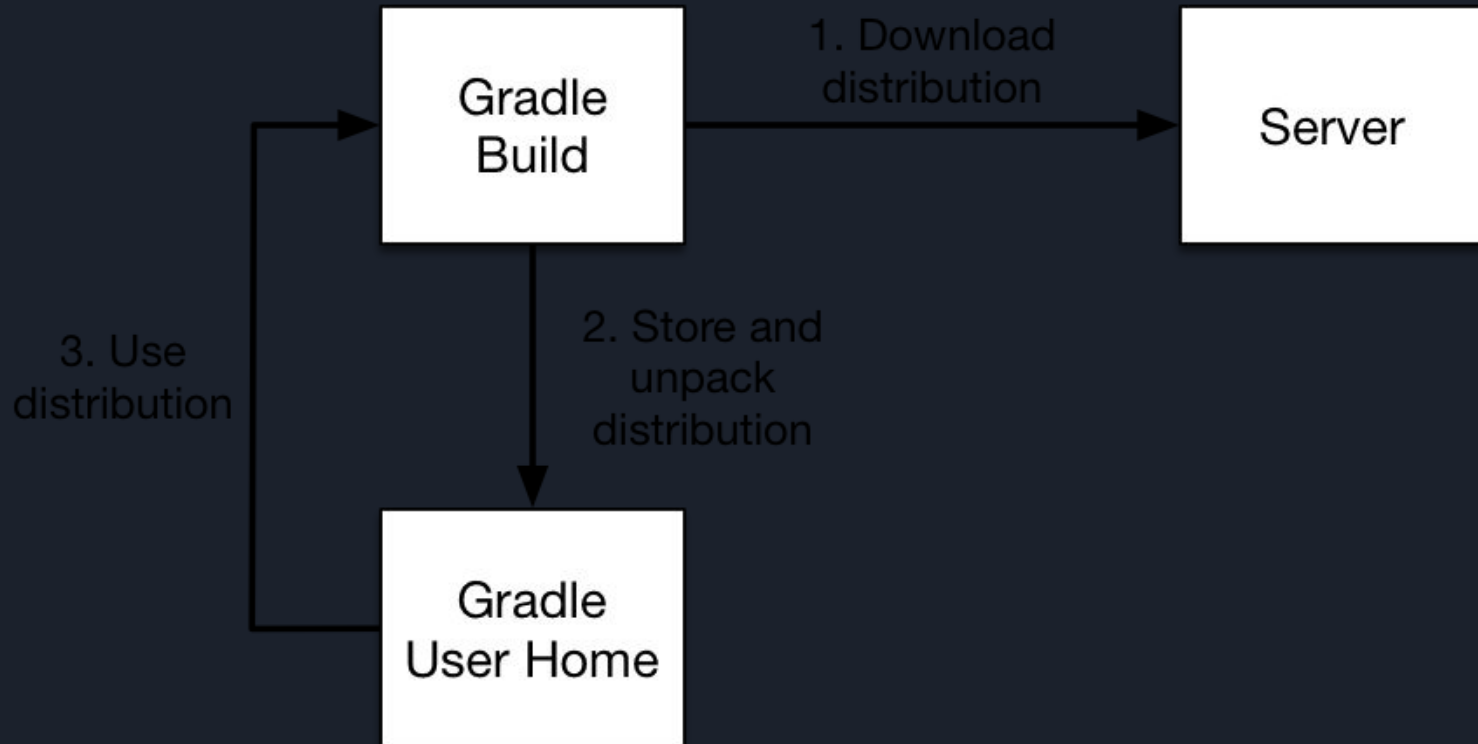


Przykład 0a

Init

Użycie gradle do swojego pierwszego projektu

Gradle wrapper (Gradlew)





Przykład 0b

Build Scan

Funkcjonalność stworzona w ramach
śledzenia etapu budowania



Przykłady

- Przykłady z praktycznej użycia do tworzenia konkretnych projektów (Pierwsze 4)
- Prezentacja bardziej zaawansowanych możliwości narzędzia Gradle (Kolejne)



Przykład 1/4

Tworzenie projektu opartego o gradle

- Tworzenie środowiska do budowania projektu
- Generowanie wyników UT
- Generowanie dokumentacji



Plugin 'java'

Głównie zadania

- compileJava
- classes
- compileTestJava
- jar
- javadoc
- test
- clean

Połączenie z bazowymi zadaniami

- assemble
- check
- build

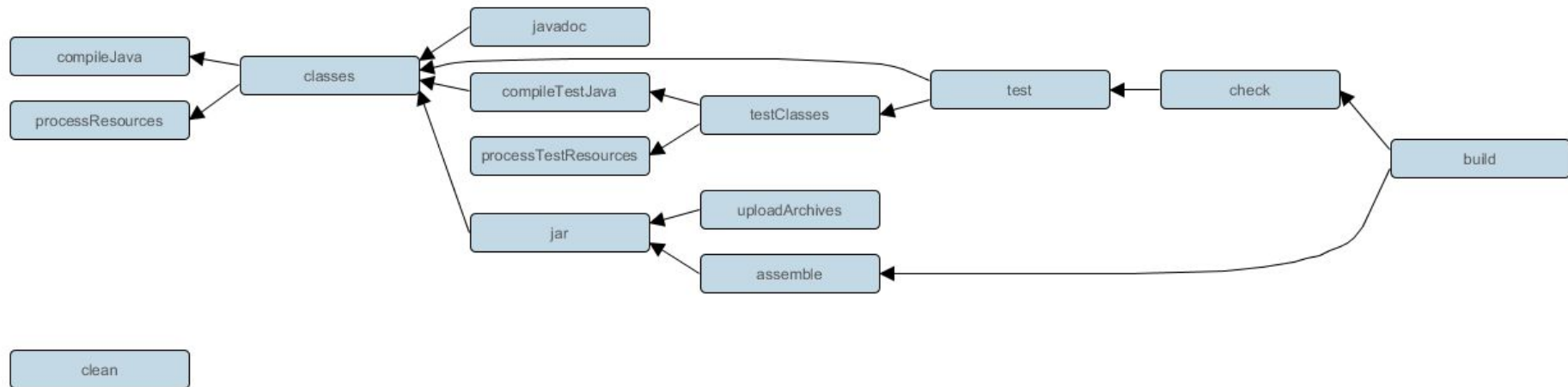
Struktura projektu

- src/main/java
- src/main/resources
- src/test/java
- src/test/resources

Dependencje

- implementation
- testImplementation
- runTimeOnly
- compileOnly

Plugin 'java' workflow





Plugin 'application'

Głównie zadania

- run
- startScripts

Wymaga podania klasy w której znajduje się Main



Przykład 2/4

Tworzenie biblioteki JAR

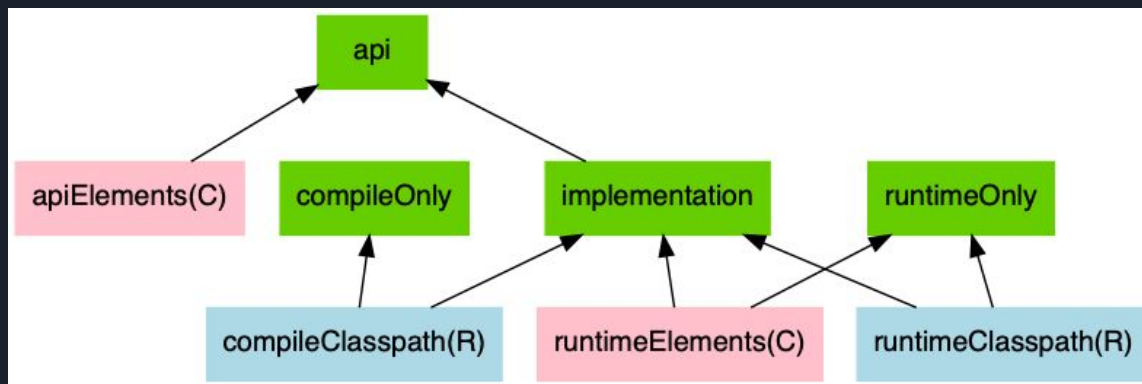
- Generowanie pliku JAR
- Pokrycie UT

Plugin 'java-library'

Rozszerza funkcjonalność pluginu 'java'

Dependencje

- api vs implementation





Przykład 3/4

Tworzenie aplikacji webowej

Plugin 'war'

Rozszerzenie pluginu 'java'

Głównie zadania

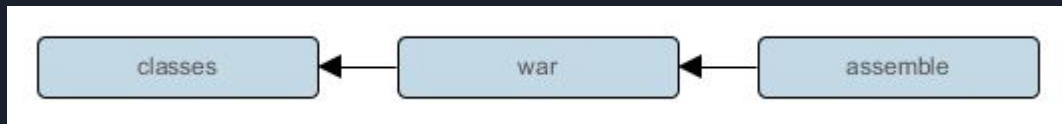
- war

Struktura projektu

- src/main/webapp

Dependencje

- providedCompile
- providedRuntime





Przykład 4/4

Tworzenie aplikacji Spring Boot



Pluginy 'spring-boot', 'spring-dependency-manager'

- Nie wchodzą w skład 'core' pluginów Gradle'a
- Zapewniają zadania do uruchomienia aplikacji
- Zapewniają wiele konfigurowalnych parametrów



Przykład: 1/2

- Definiowanie własnych zadań
- Zmiany domyślnych zadań
- Łączenie Javy i Groovy'ego
- Wykorzystanie Gradle API



Przykład: 2/2

- Stworzenie własnego pluginu
- Wykorzystanie `gradle.properties`



Zadanie:

Proszę przerobić jedno z wybranych zadań realizowanych na zajęciach laboratoryjnych z użycia Mavena na Gradle'a.



Dziękuję za uwagę!