Gradle

Miłosz Filus

ZTI 1.06.2020

Czym jest gradle?

Narzędzie do zautomatyzowania procesu budowania i zarządzania projektem

Zalety

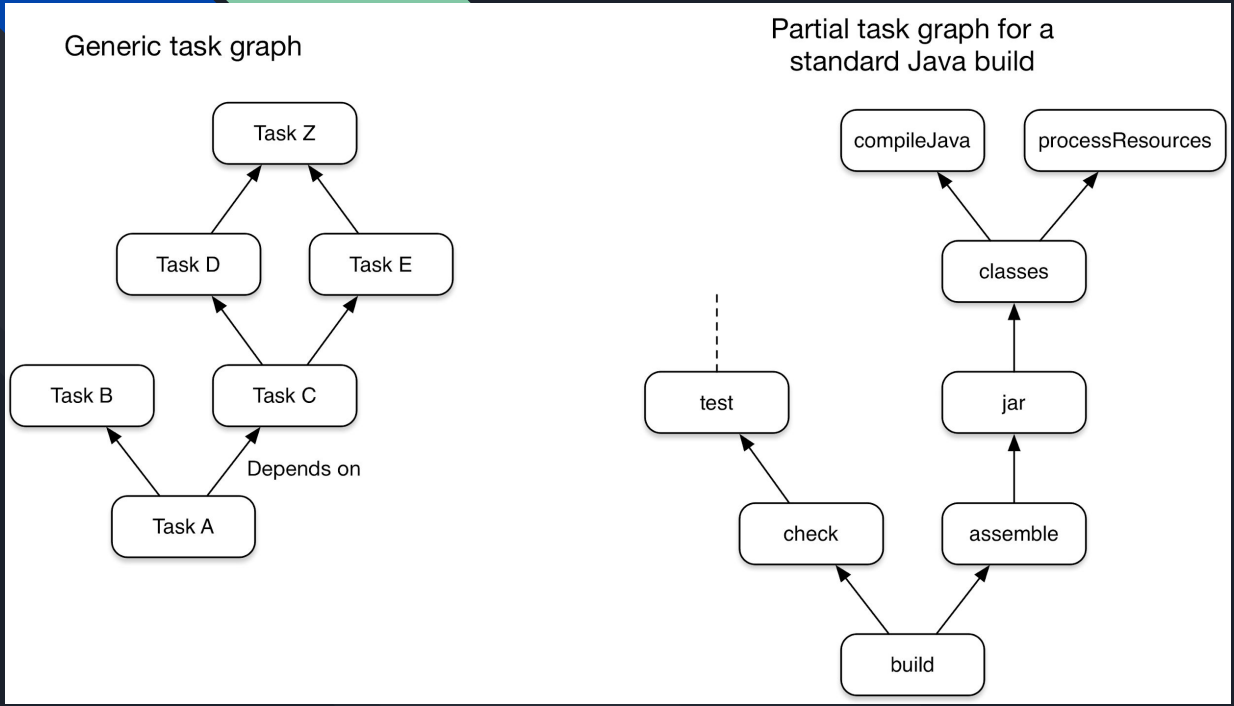
- Cache'owanie plików wyjściowych, bibliotek
- Wielowątkowa egzekucja
- Wielowątkowe pobieranie zależności
- Wykrywanie zmian w drzewie zadań
- Metryki dla wykonanych buildów
- i wiele wiele więcej

Aspekty Gradle 1/5

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

Mechanizm pluginów oraz wchodzących w ich skład zadań

Aspekty Gradle 2/5



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

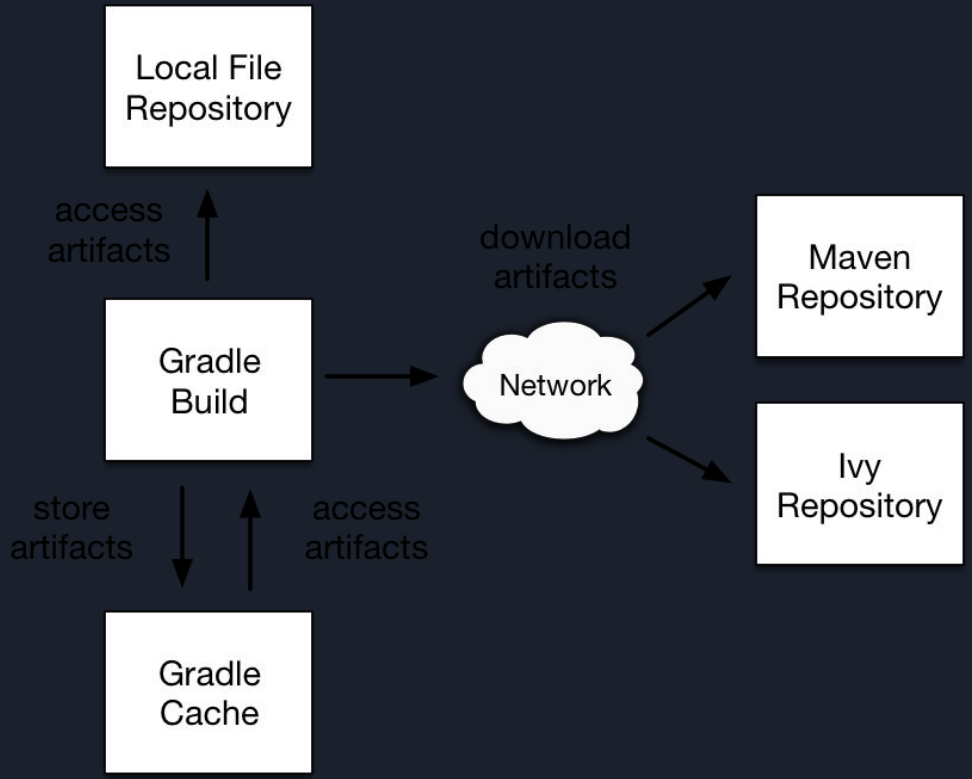
Podobnie jak Kotlin, który może wywoływać klasy Javy

Przejrzystość i prostota

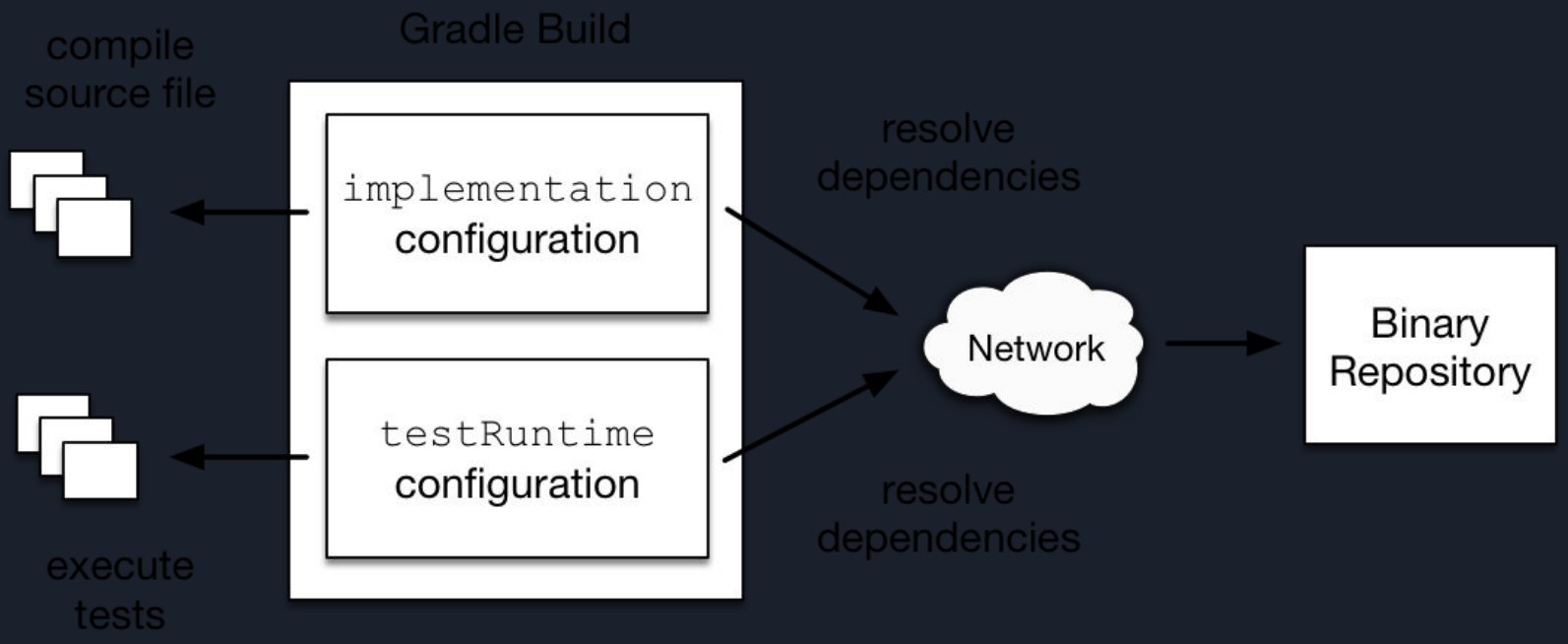
Mechanizm zapewniania zależności

- Repozytoria - czyli skąd
- Dependencje - czyli co

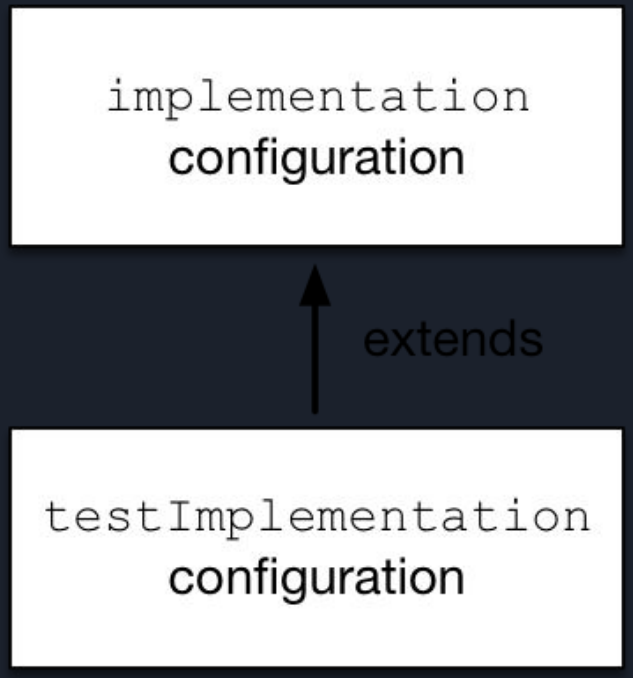
Menedżer dependencji



Dependencje 1/2



Dependencje 2/2




```
com.google.guava:guava:23.0
```

```
junit:junit:4.12
```

Główne pliki narzędzia Gradle

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

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

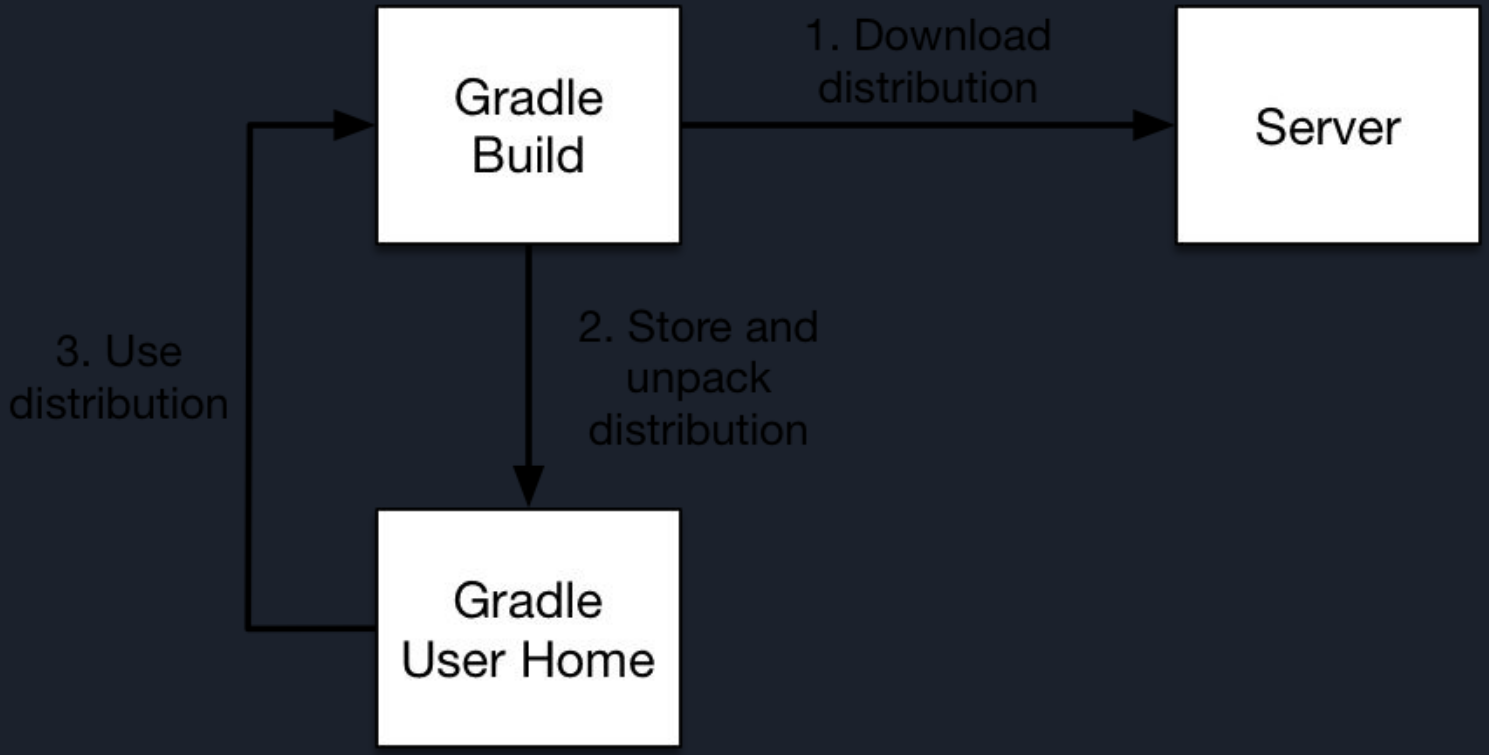
properties.gradle - nie jest tworzony domyślnie, posiada linie typu klucz:wartość, można w nim zdefiniować opcje przekazane do JVM'a, opcje do samego gradle'a czy też dane potrzebne do autoryzacji z bazą danych

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’

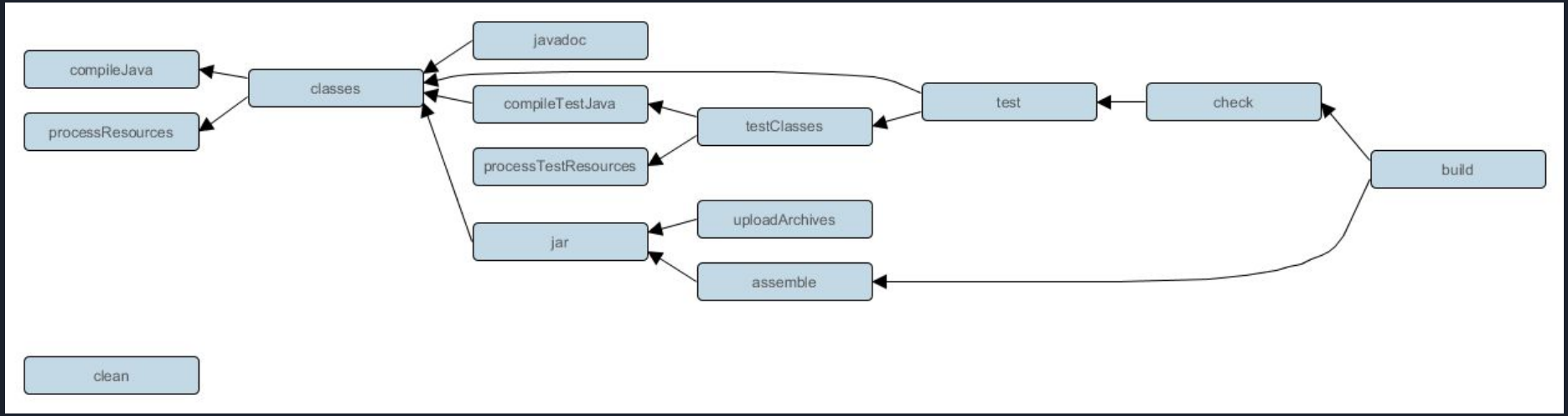
Głównie zadania
- compileJava
- classes
- compileTestJava
- jar
- javadoc
- test
- clean

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

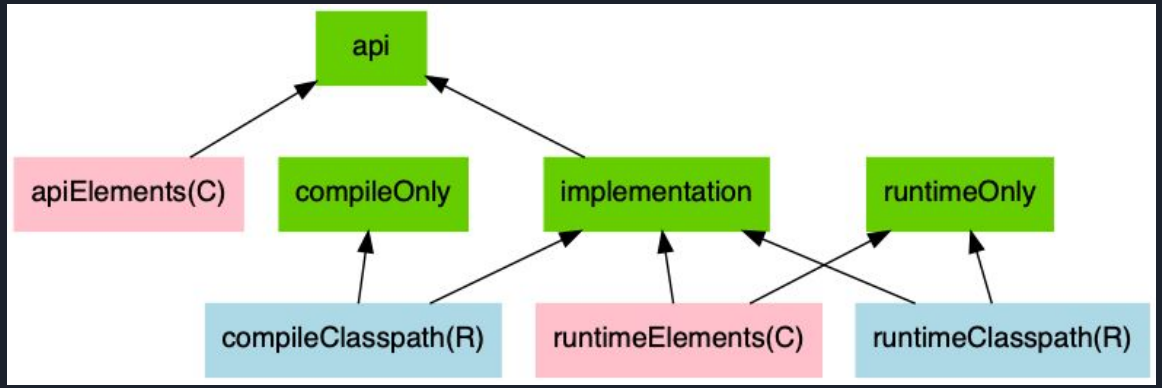
- Generowanie pliku JAR
- Pokrycie UT

Plugin ‘java-library’

Rozszerza funkcjonalność pluginu ‘java’

Dependencje

- api vs implementation



Przykład 3/4

Tworzenie aplikacji webowej

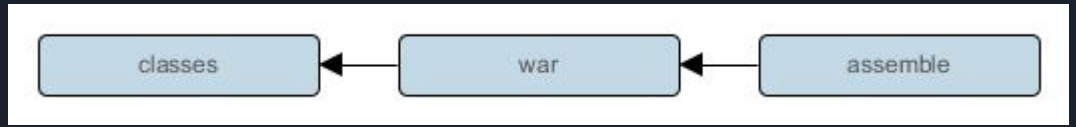
Plugin ‘war’

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-dependecy-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!