SYSTEMY BUDOWANIA APLIKACJI

MAVEN

Gdańsk, 15.01.2015

O MNIE

- Absolwent wydziału Fizyki Technicznej i Matematyki Stosowanej Politechniki Gdańskiej
- Programista w firmie Solwit
- Wykładowca na Uniwersytecie Gdańskim

- Blogger: blog.dragonia.org.pl
- Twitter: @Smoczysko
- LinkedIn: pl.linkedin.com/in/Smoczysko

PODSTAWOWE USTALENIA

- ✓ nie jestem alfą ani omegą

PODSTAWOWE USTALENIA

- ✓ nie jestem alfą ani omegą
- ✓ nie jestem sprzedawcą, a zwykłym developerem

PODSTAWOWE USTALENIA

- ✓ nie jestem alfą ani omegą
- ✓ nie jestem sprzedawcą, a zwykłym developerem
- ✓ pytania mile widziane, szczególnie w trakcie

PODSTAWOWE USTALENIA

- ✓ nie jestem alfą ani omegą
- ✓ nie jestem sprzedawcą, a zwykłym developerem
- ✓ pytania mile widziane, szczególnie w trakcie
- ✓ 2 spotkania - 3 narzędzia do budowania

PODSTAWOWE USTALENIA

- ✓ nie jestem alfą ani omegą
- ✓ nie jestem sprzedawcą, a zwykłym developerem
- ✓ pytania mile widziane, szczególnie w trakcie
- ✓ 2 spotkania - 3 narzędzia do budowania
- ✓ mam grypę - mogę niewyraźnie mówić

https://github.com/Smoczysko/introduction-to-build-tools

HISTORIA NARZĘDZI DO BUDOWANIA

- C/C++: kompilacja -> linkowanie -> aplikacja
- Java: javac class_name.java -> class_name.class
- Shell scripts
- Make, GNU Automake/Autoconf

HISTORIA NARZĘDZI DO BUDOWANIA

- C/C++: kompilacja -> linkowanie -> aplikacja
- Java: javac class_name.java -> class_name.class
- Shell scripts
- Make, GNU Automake/Autoconf

- …

HISTORIA NARZĘDZI DO BUDOWANIA

HISTORIA NARZĘDZI DO BUDOWANIA

maven

HISTORIA NARZĘDZI DO BUDOWANIA

maven

sbt

HISTORIA NARZĘDZI DO BUDOWANIA

maven

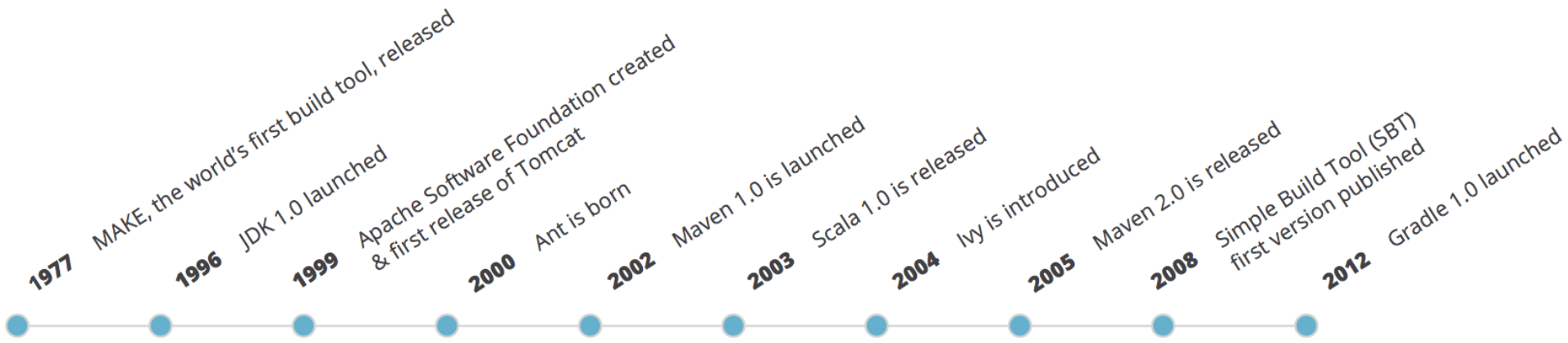
sbt

gradle

HISTORIA NARZĘDZI DO BUDOWANIA

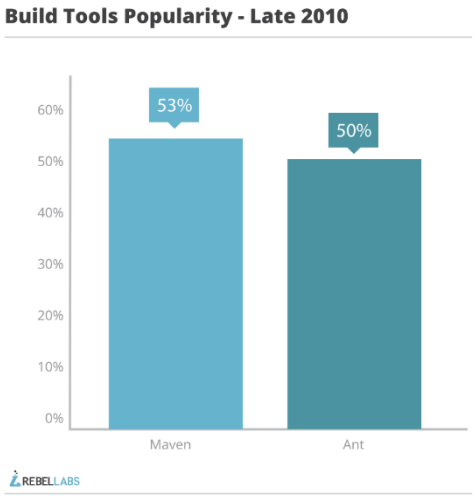
THE EVOLUTION OF BUILD TOOLS: 1977 - 2013 (AND BEYOND)

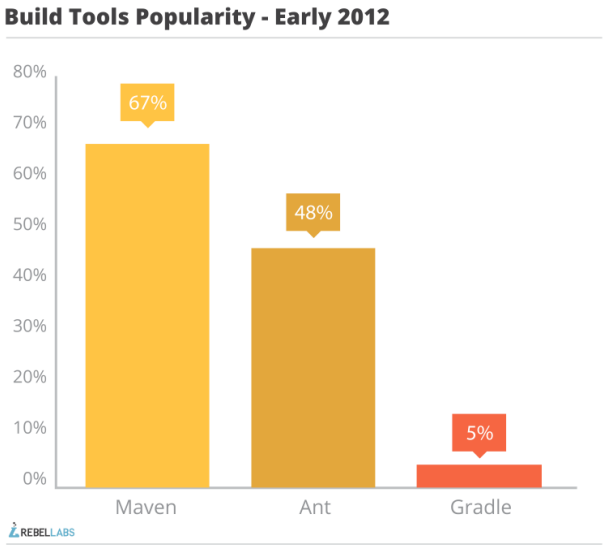
Visual timeline

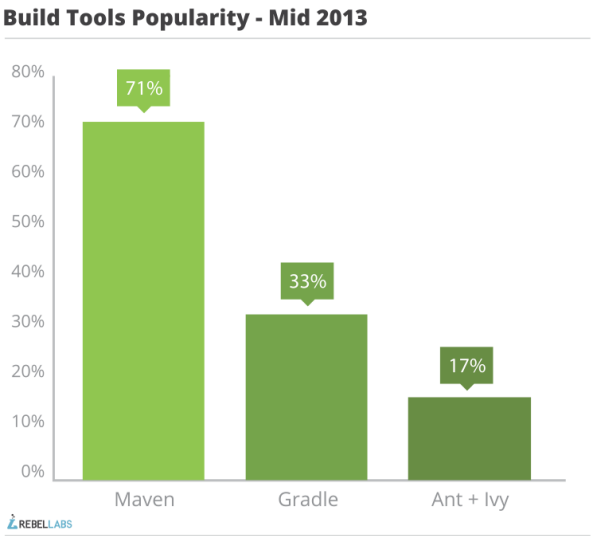


REBELLABS

HISTORIA NARZĘDZI DO BUDOWANIA







CO POWINIEN DOSTARCZAĆ BUILD TOOL

- ✓ zarządzanie zależnościami

CO POWINIEN DOSTARCZAĆ BUILD TOOL

- ✓ zarządzanie zależnościami
- ✓ inkrementalna rekompilacja (kompilacja wyłacznie tych fragmentów, które zostały zmodyfikowane)

CO POWINIEN DOSTARCZAĆ BUILD TOOL

- ✓ zarządzanie zależnościami
- ✓ inkrementalna rekompilacja (kompilacja wyłacznie tych fragmentów, które zostały zmodyfikowane)
- ✓ obsługiwanie zadań kompilacji oraz zarządzania zasobami

CO POWINIEN DOSTARCZAĆ BUILD TOOL

- ✓ zarządzanie zależnościami
- ✓ inkrementalna rekompilacja (kompilacja wyłacznie tych fragmentów, które zostały zmodyfikowane)
- ✓ obsługiwanie zadań kompilacji oraz zarządzania zasobami
- ✓ obsługa wielu profili (środowisk)

CO POWINIEN DOSTARCZAĆ BUILD TOOL

- ✓ zarządzanie zależnościami
- ✓ inkrementalna rekompilacja (kompilacja wyłacznie tych fragmentów, które zostały zmodyfikowane)
- ✓ obsługiwanie zadań kompilacji oraz zarządzania zasobami
- ✓ obsługa wielu profili (środowisk)
- ✓ dostosowywanie się do zmieniających się wymagań projektu

CO POWINIEN DOSTARCZAĆ BUILD TOOL

- ✓ zarządzanie zależnościami
- ✓ inkrementalna rekompilacja (kompilacja wyłacznie tych fragmentów, które zostały zmodyfikowane)
- ✓ obsługiwanie zadań kompilacji oraz zarządzania zasobami
- ✓ obsługa wielu profili (środowisk)
- ✓ dostosowywanie się do zmieniających się wymagań projektu
- ✓ zaprojektowane z myślą o automatyzacji procesu budowania

THE JOEL TEST

The Joel Test

1. Do you use source control?
2. Can you make a build in one step?
3. Do you make daily builds?
4. Do you have a bug database?
5. Do you fix bugs before writing new code?
6. Do you have an up-to-date schedule?
7. Do you have a spec?
8. Do programmers have quiet working conditions?
9. Do you use the best tools money can buy?
10. Do you have testers?
11. Do new candidates write code during their interview?
12. Do you do hallway usability testing?

źródło: http://www.joelonsoftware.com/articles/fog0000000043.html

APACHE MAVEN

- Project management tool
  - Budowanie (kompilacja, linkowanie, …)
  - Testowanie
  - Raportowanie
  - Dokumentacja

APACHE MAVEN

- Project management tool
  - Budowanie (kompilacja, linkowanie, …)
  - Testowanie
  - Raportowanie
  - Dokumentacja
- Zarządzanie i rozwiązywanie zależności

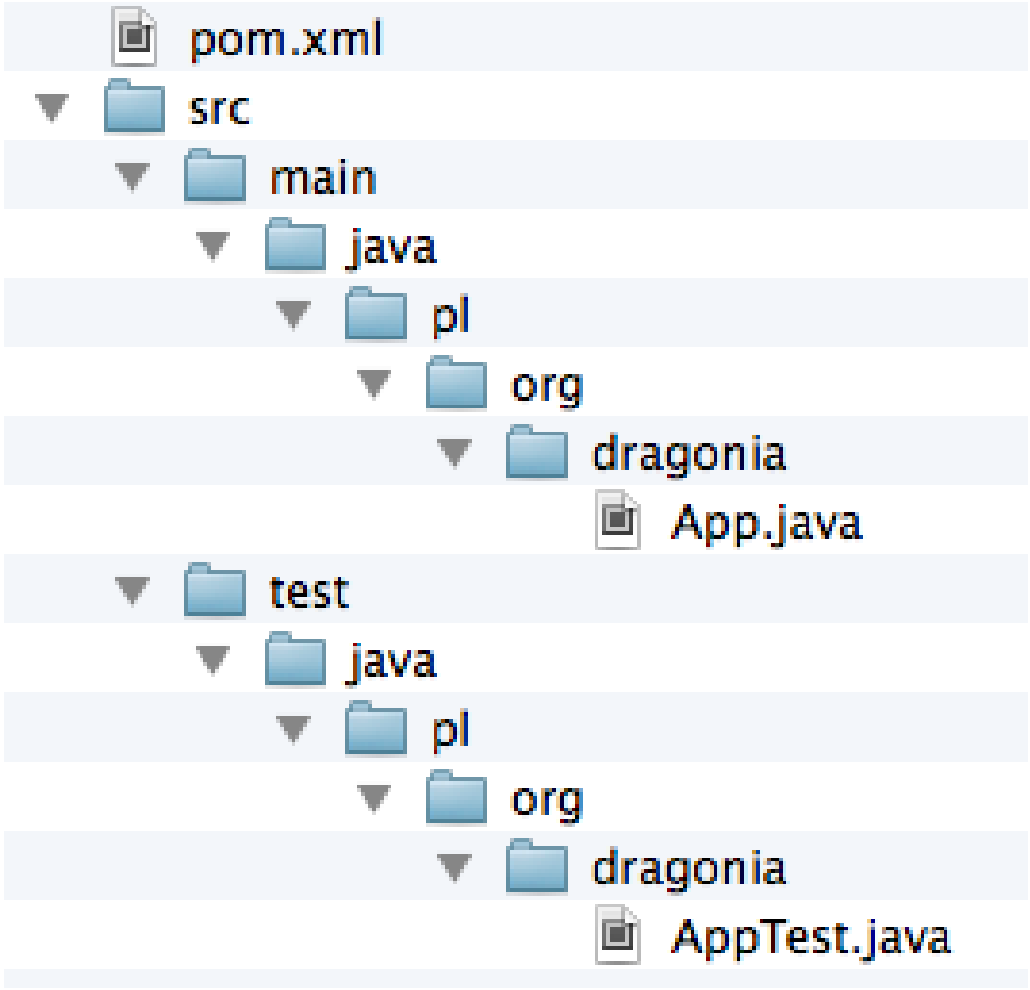
APACHE MAVEN

- Project management tool
  - Budowanie (kompilacja, linkowanie, …)
  - Testowanie
  - Raportowanie
  - Dokumentacja
- Zarządzanie i rozwiązywanie zależności
- Centralne miejsce informacji i konfiguracji projektu (plik POM)

KONFIGURACJA PROJEKTU

- ✓ Odpowiednia struktura katalogów i plików
- ✓ Plik pom.xml

STRUKTURA KATALOGÓW I PLIKÓW



PLIK POM.XML

- Project Object Model – fundamentalny plik przy pracy z projektem Maven’owym
- Plik XML zawierający podstawowe informacje o projekcie i jego konfiguracji

MINIMALISTYCZNY PLIK POM.XML

- Wymagane informacje:
  - modelVersion (4.0.0 – oznacza build zgodny z Maven 2)
  - groupId – ID grupy, do którego należy projekt (zazwyczaj pokrywa się z pakietem)
  - artifactId – ID artefaktu (projektu)
  - version - bieżąca wersja projektu

```
<project>
   <modelVersion>4.0.0</modelVersion>
   <groupId>pl.org.dragonia</groupId>
   <artifactId>sample-app</artifactId>
   <version>1.0-SNAPSHOT</version>
</project>
```

ARCHETYPY

- Gotowe do użycia projekty
- Automatyczne generowanie i podstawowa konfiguracja
- Ogromna ilość (setki!) gotowych i dostępnych archetypów
- Najczęściej wykorzystywane archetypy:
  - **maven-archetype-quickstart** - prosty i podstawowy szkielet projektu z odpowiednią strukturą katalogów
  - **maven-archetype-webapp** - aplikacja webowa z podstawową konfiguracją (w plikach XML)
  - **maven-archetype-j2ee-simple** - aplikacja JEE z podziałem na projekty i komponenty (w tym EJB w starej konfiguracji)

ARCHETYPY

- Gotowe do użycia projekty
- Automatyczne generowanie i podstawowa konfiguracja
- Ogromna ilość (setki!) gotowych i dostępnych archetypów
- Najczęściej wykorzystywane archetypy:
  - **maven-archetype-quickstart** - prosty i podstawowy szkielet projektu z odpowiednią strukturą katalogów
  - **maven-archetype-webapp** - aplikacja webowa z podstawową konfiguracją (w plikach XML)
  - **maven-archetype-j2ee-simple** - aplikacja JEE z podziałem na projekty i komponenty (w tym EJB w starej konfiguracji)

mvn archetype:generate

-DarchetypeGroupId=…
-DgroupId=…
-DartifactId=…

ZBUDOWANIE I URUCHOMIENIE

- Zbudowanie (kompilacja, stworzenie archiwum JAR)

mvn package

ZBUDOWANIE I URURCHOMIENIE

- Zbudowanie (kompilacja, stworzenie archiwum JAR)

mvn package

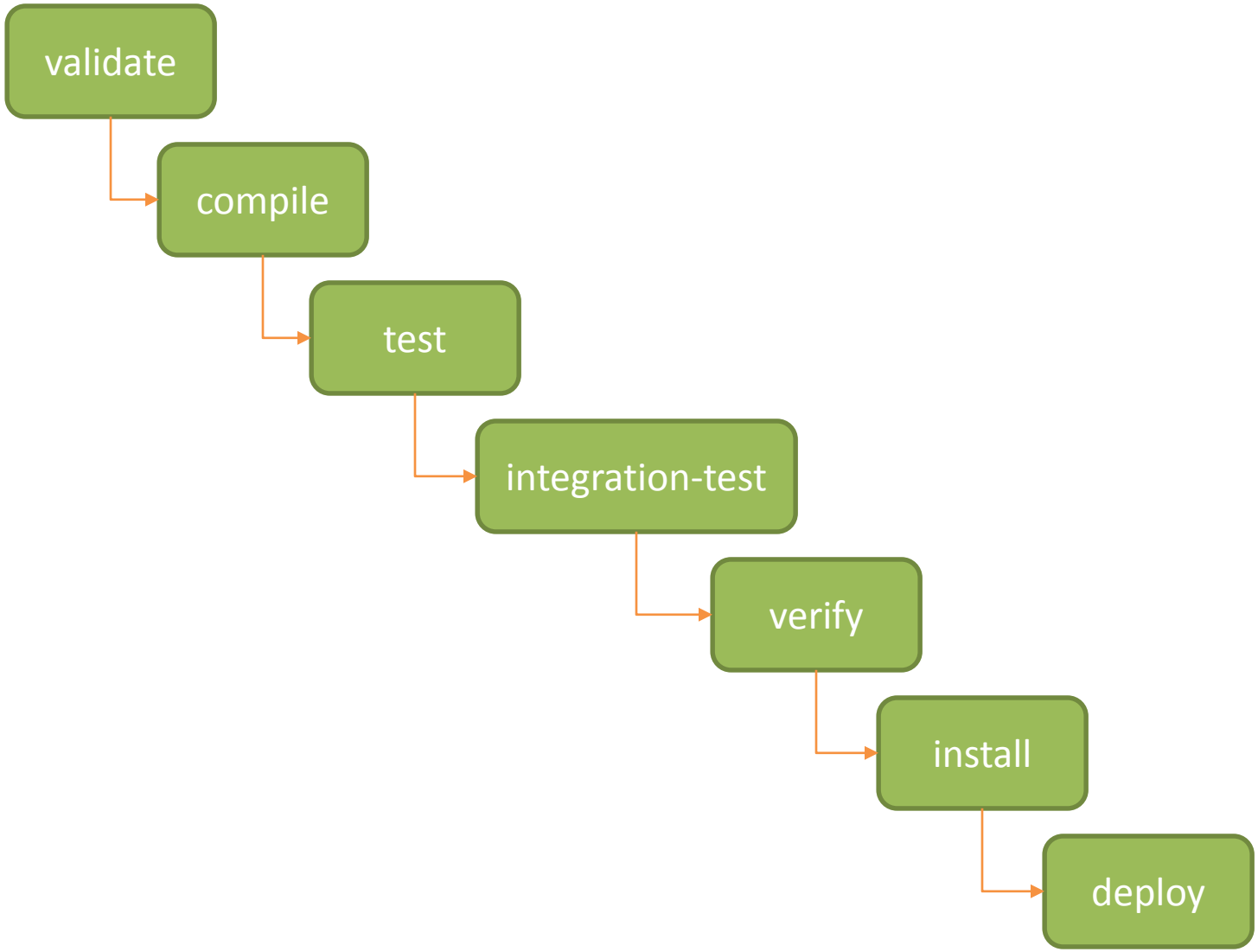
- Uruchomienie:

```
java -cp \
target/sample-app-1.0-SNAPSHOT.jar pl.org.dragonia.App
```

CYKL BUDOWANIA APLIKACJI

- Jasno określnone (i powtarzalne) reguły
- Wystarczy znajomość zaledwie kilku komend do efektywnego budowania aplikacji
- Wbudowane cykle budowania:
  - default - deployment aplikacji
  - clean - czyszczenie projektu
  - site - tworzenie dokumentacji

DEFAULT LIFECYCLE



FULL DEFAULT LIFECYCLE

- **validate**
- initialize
- generate-sources
- process-sources
- generate-resources
- process-resources
- **compile**
- process-classes
- generate-test-sources
- process-test-sources
- generate-test-resources
- process-test-resources
- test-compile
- process-test-classes
- **test**
- prepare-package
- **package**
- pre-integration-test
- integration-test
- post-integration-test
- **verify**
- **install**
- **deploy**

źródło: http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Lifecycle_Reference

ZARZĄDZANIE ZALEŻNOŚCIAMI

- Wbudowane w Maven'a
- Możliwość zdefiniowania odpowiedniego zakresu:
  - compile
  - provided
  - tests
  - system
- Nie chroni przed powtarzającymi (cyklicznymi) się zależnościami (tzw. **transitive dependency**)

- Zależności to takie same projekty Maven'owe jak nasz

BIBLIOTEKA DO TESTOWANIA – JUNIT 4

```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.11</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

PLUGINS

- zależności wykorzystywane w trakcie budowania zamiast w kodzie aplikacji
- rozbudowują możliwości narzędzia niemalże nieskończenie
- podstawowe funkcje (komendy/fazy) składają się z cyklicznie wywoływanych pluginów

- najpopularniejsze pluginy:
  - **maven-assembly-plugin**
  - maven-ant-plugin
  - maven-surefire-plugin
  - **maven-checkstyle-plugin**
  - gwt-maven-plugin
  - **jetty-maven-plugin**

EXECUTABLE JAR

```
<build>
    <plugins>
        <plugin>
            <artifactId>maven-assembly-plugin</artifactId>
            <configuration>
                <finalName>executable</finalName>
                <archive>
                    <manifest>
                        <mainClass>pl.org.dragonia.App</mainClass>
                    </manifest>
                </archive>
                <descriptorRefs>
                    <descriptorRef>jar-with-dependencies</descriptorRef>
                </descriptorRefs>
                <appendAssemblyId>false</appendAssemblyId>
            </configuration>
            <executions>
                <execution>
                    <id>make-assembly</id>
                    <phase>package</phase>
                    <goals>
                        <goal>single</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

RAPORTY

- ✓ Wbudowany mechanizm generowania raportów
- ✓ Raporty dają szczegółowy pogląd na projekt bez zaglądania w kod i konfiguracje aplikacji
- ✓ Możliwość rozszerzania zawartości raportów
  - maven-checkstyle-plugin
- ✓ Możliwość generowania raportów w innym formacie niż HTML
  - oraz z innym wyglądem/strukturą

MAVEN CHECKSTYLE PLUGIN

```
<reporting>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-checkstyle-plugin</artifactId>
            <version>2.13</version>
            <reportSets>
                <reportSet>
                    <reports>
                        <report>checkstyle</report>
                    </reports>
                </reportSet>
            </reportSets>
        </plugin>
    </plugins>
</reporting>
```

INFORMACJE O PLUGINIE (HELP)

- ✓ Funkcjonalność dostarczana przez plugin (**!!**): maven-help-plugin
- ✓ Wypisuje wszystkie atrybuty i/oraz cele (goals)

mvn help:describe -Dplugin=<groupId>:<artifactId>

INFORMACJE O PLUGINIE

mvn help:describe -Dplugin=<groupId>:<artifactId>

mvn help:describe
-Dplugin=org.apache.tomcat.maven:tomcat7-maven-plugin

ZEWNĘTRZNY PLIK PROPERTIES

- ✓ Funkcjonalność (jak zawsze) dostarczana przez plugin: maven-properties-plugin
- ✓ Pozwala zarównno wczytywać jak i zapisywać pliki properties (w standardowym dla Javy formacie)

```
<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>properties-maven-plugin</artifactId>
    <version>1.0-alpha-2</version>
    <executions>
        <execution>
            <phase>validate</phase>
            <goals>
                <goal>read-project-properties</goal>
            </goals>
            <configuration>
                <files>
                    <file>${project.basedir}/build.properties</file>
                </files>
            </configuration>
        </execution>
    </executions>
</plugin>
```

JAVA ENTERPRISE EDITION APPLICATION

- ✓ Konfiguracja za pomocą jednej zależności i dwóch pluginów
- ✓ Budowanie aplikacji za pomocą maven-war-plugin
- ✓ Serwer aplikacji dostępny w trybie in-memory (Jetty, Tomcat, GlassFish AS)
- ✓ Pełne pokrycie standardów JEE 5, 6 oraz 7

JEE6 APPLICATION - DEPENDENCY

```
<dependencies>
    <dependency>
        <groupId>javax</groupId>
        <artifactId>javaee-web-api</artifactId>
        <version>6.0</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

JEE6 APPLICATION – WAR PLUGIN

```xml
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-war-plugin</artifactId>
    <version>2.1.1</version>
    <configuration>
        <failOnMissingWebXml>false</failOnMissingWebXml>
    </configuration>
</plugin>
```

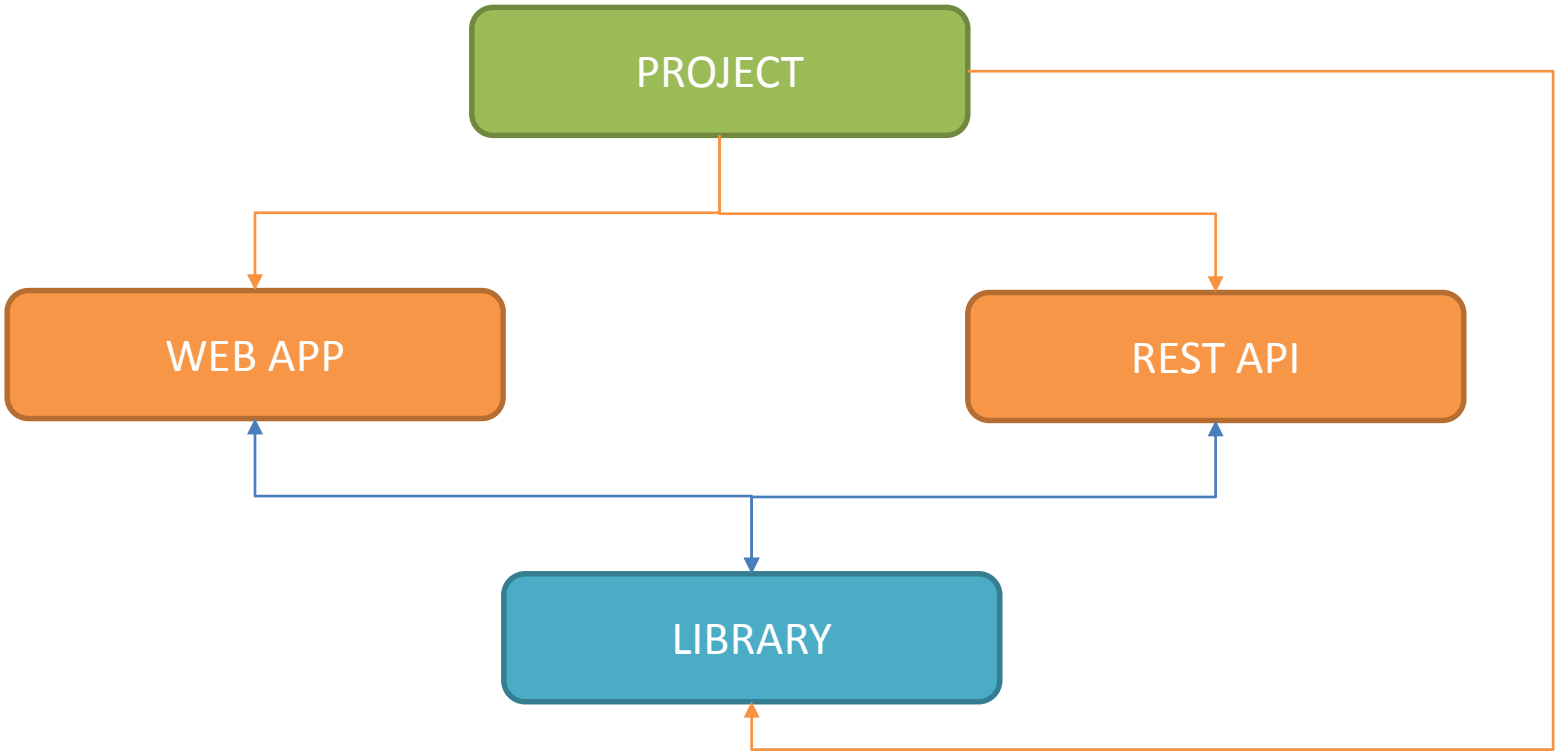
JEE6 APPLICATION – JETTY

```
<plugin>
    <groupId>org.mortbay.jetty</groupId>
    <artifactId>jetty-maven-plugin</artifactId>
    <version>8.0.3.v20111011</version>
    <configuration>
        <scanIntervalSeconds>3</scanIntervalSeconds>
        <webAppConfig>
            <contextPath>/${project.build.finalName}</contextPath>
        </webAppConfig>
    </configuration>
</plugin>
```

MODUŁY

- ✓ pozwalają w naturalny sposób podzielić strukturę projektu, wydzielić logiczne fragmenty (komponenty)
- ✓ bazowa konfiguracja i zależności zdefiniowane raz dla całego projektu
- ✓ możliwość ustawienia zależności między modułami - **uwaga!!**
- ✓ dowolna strultura (ilość zagnieżdżeń) modułów i ich podmodułów

ZŁY PRZYKŁAD



PYTANIA?

Q&A