Notatki - Maven - Fat Jar

Spis treści

Building Fat Jar

`Fat Jar` to pojedynczy plik `.jar`, który zawiera w sobie wszystkie skompilowane klasy z naszego projektu oraz wszystkie skompilowane (zgodnie z konfiguracją) klasy z plików `.jar`, które zostały określone jako dependencje naszego projektu. Plik taki jest o tyle wygodny, że nie musimy wskazywać położenia bibliotek, gdyż są one dołączone w ramach jednego pliku. Nie musimy wtedy wszystkich tych plików wskazywać na `classpath`.

Taki plik `.jar` możemy stworzyć przy pomocy `Maven`. Oczywiście potrzebna jest do tego odpowiednia konfiguracja. W tym celu wykorzystamy plugin `maven-assembly-plugin`. Przykładowa konfiguracja:

```
<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">

    <!-- Tutaj umieszczamy konfigurację, którą widzieliśmy już wcześniej -->
    ...

    <build>
        <finalName>zajavka-title-reader</finalName>
        <plugins>
            <!-- Inne pluginy -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-assembly-plugin</artifactId>
                <version>3.1.1</version>
                <configuration>
                    <archive>
                        <manifest>
                            <mainClass>pl.zajavka.MavenCompilingJsoupExamplesRunner</mainClass>
                        </manifest>
                    </archive>
                    <descriptorRefs>
                        <descriptorRef>jar-with-dependencies</descriptorRef>
                    </descriptorRefs>
                </configuration>
                <executions>
                    <execution>
                        <id>make-assembly</id>
                        <phase>package</phase>
                        <goals>
                            <goal>single</goal>
```

```
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

Dzięki określeniu:

```
<descriptorRef>jar-with-dependencies</descriptorRef>
```

mówimy pluginowi, że chcemy stworzyć `Fat Jar`, czyli `Jar with dependencies`. Tag `execution` jest analogiczny do poprzednich przykładów, tym razem jednak dodajemy do tego taga pole `id`. Możemy teraz uruchomić komendę:

```
mvn clean verify
```

Maven wyprodukuje `Fat Jar` w katalogu `target`, który jest domyślnym katalogiem, w którym `Maven` produkuje swoje produkty ☺. Schemat nazewnictwa takiego pliku wygląda następująco:

```
zajavka-title-reader-jar-with-dependencies.jar
```

Jeżeli nie dodalibyśmy taga `execution` z powyższą konfiguracją, należałoby uruchomić utworzenie `Fat Jar` za pomocą komendy:

```
mvn clean package assembly:single
```

Jeżeli natomiast chcemy dać możliwość uruchomienia naszego programu z poziomu pliku `.jar` (inaczej tzw. **executable jar**), należy do konfiguracji dodać fragment:

```
<archive>
    <manifest>
        <mainClass>pl.zajavka.MavenCompilingJsoupExamplesRunner</mainClass>
    </manifest>
</archive>
```

Inaczej przy próbie uruchomienia:

```
java -jar zajavka-title-reader-jar-with-dependencies.jar
```

Dostaniemy poniższy błąd:

```
no main manifest attribute, in zajavka-title-reader-jar-with-dependencies.jar
```

Profile

Profile budowania są wykorzystywane jeżeli chcemy wskazać różnice między sposobami budowania naszej aplikacji w zależności od tego, na które środowisko ma zostać ona wdrożona (DEV, TEST, PROD). Możemy wtedy włączać lub wyłączać konkretne operacje lub ustawienia, w zależności od środowiska, które nas interesuje. Mielibyśmy wtedy np. profil DEV, TEST oraz PROD i każdy profil przygotowywałby zbudowany projekt na dedykowane środowisko. Link do dokumentacji umieszczam tutaj.

Jeżeli chcemy zastosować profil, należy dodać tag profiles jak poniżej:

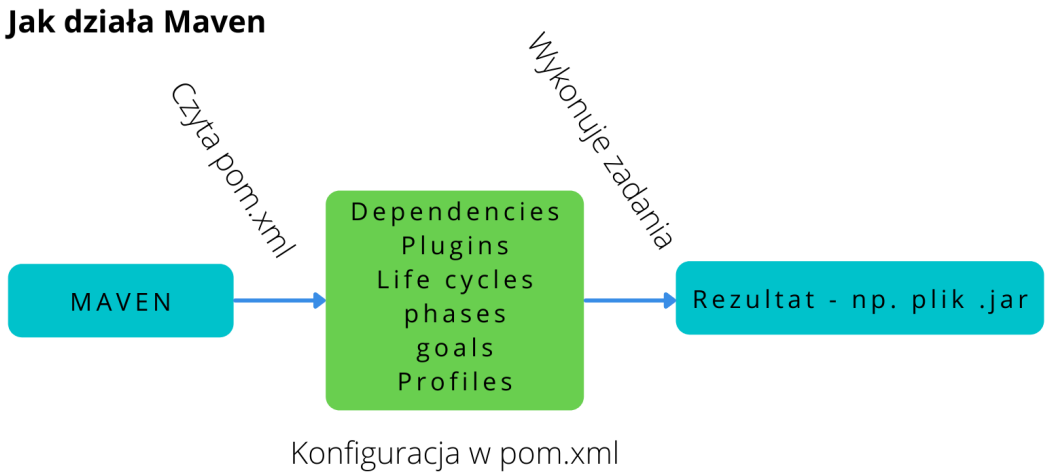
```
<project
    xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
        http://maven.apache.org/xsd/maven-4.0.0.xsd">

    ...
    <profiles>
        <profile>
            <id>TEST</id>
            <build>...</build>
            <modules>...</modules>
            <repositories>...</repositories>
            <dependencies>...</dependencies>
        </profile>
    </profiles>
    ...
</project>
```

Jeżeli następnie chcemy uruchomić komendę maven z wybranym profilem, możemy to zrobić w ten sposób:

```
mvn compile -P TEST
```

Podsumowanie

Podsumowanie działania Maven umieściłem na grafice poniżej.



Obraz 1. Jak działa Maven