Notatki - Maven - Scope

Spis treści

Czym jest scope? . . . 1
Snapshoty . . . 2

Czym jest scope?

Wróćmy jednak do jeszcze jednej kwestii, skoro już wiemy czym jest `classpath`. Czym jest scope? Widzieliśmy już taki zapis:

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.20</version>
    <scope>provided</scope>
</dependency>
```

Głębsze wyjaśnienie oczywiście można znaleźć w dokumentacji. Scope określa nam w jaki sposób mamy podchodzić do umieszczania pobranej dependencji na `classpath`. Przykłady poniżej odnoszą się też do sytuacji, gdy chcemy zbudować wynikowy plik `.jar`. Możemy wtedy zdecydować czy umieszczać zewnętrzne zależności w tym pliku `.jar`, lub czy zależności są nam potrzebne na etapie kompilacji, czy może dopiero w trakcie działania programu. Poniżej wyjaśniam w tabelce, nie podaję też wszystkich dostępnych `scope`.

Scope	Opis
compile	Defaultowy scope, czyli jak nie wybierzemy żadnego to będzie wybrany ten. Dependencje oznaczone jako compile będą dostępne na wszystkich classpathach i na etapie kompilacji i na etapie wykonania programu
provided	Podobny do scope compile, ale określa, że oczekujemy, że zależność jest konieczna na etapie kompilacji. Natomiast w trakcie działania programu oczekujemy, że dependencja zostanie dostarczona przez serwer, na którym uruchamiamy nasz program. Ustawimy też ten zakres jeżeli potrzebujemy dependencji na etapie budowania, ale w trakcie działania już nie. Dependencja w scope provided jest też dostępna w classpath dla testów
runtime	Dependencja nie jest potrzebna na etapie kompilacji, ale jest wymagana w trakcie działania programu.
test	Dependencja nie jest wymagana podczas normalnego działania programu, potrzebujemy jej natomiast jeżeli będziemy pisać testy do naszego programu. Najczęściej używane do bibliotek, które są stosowane tylko do testów.

Jeżeli chodzi o scope `test`, wspomnimy niedługo o testach, wtedy rozjaśni się to bardziej.

Snapshoty

Możliwe jest znalezienie dependencji, które w numerze wersji będą miały dopisek **SNAPSHOT**. Oznaczenie to służy do tego, żeby dać znać korzystającemu z biblioteki, że korzysta z wersji, która nie jest ostateczna i może ulegać zmianie. Czyli jeżeli pobraliśmy zawartość takiej biblioteki dzisiaj i zrobimy to samo jutro, to teoretycznie taka zawartość może się zmienić. Pewnie klasy i metody mogą zostać w tym czasie dodane, a inne mogą zostać usunięte. My również możemy w naszym opisie pliku `pom.xml` stosować dopisek **SNAPSHOT**.

Jeżeli chcemy napisać, że zależy nam na dependencji **SNAPSHOT** definiujemy to w taki sposób:

```
<version>1.0.0-SNAPSHOT</version>
```