Laboratorium
Maven – automatyzacja budowy oprogramowania

Głównym celem laboratoriów jest zapoznanie się z narzędziem Apache Maven służącym do automatyzacji budowy oprogramowanie na platformie Java. Zajęcia powinny pomoc studentom zapoznać się z zasadą działania tego narzędzia wraz z podstawowymi mechanizmami służącymi do organizacji, konfiguracji oraz zarządzania zależnościami między projektami.

Podstawowe informacje

http://coach.kis.p.lodz.pl/maven.pdf – opis narzędzia.

http://maven.apache.org/ – strona domowa Maven.

Zadania

1. Automatyczna generacja POMu i projektu

Pliki POM wymagają dostosowania do potrzeb projektu, jednak Maven pozwala na wygenerowanie podstawowego pliku dla dowolnego projektu. Plik ten wymaga ręcznej edycji, jednak o ile struktura projektu jest zgodna z konwencją, wystarcza on do wykonania podstawowych operacji.

Polecenie służące do generacji POMu wygląda następująco:

```
mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes
         -DgroupId=com.mycompany.app -DartifactId=my-app -DinteractiveMode=false
```

Używamy tu wtyczki (*plugin*) `archetype` z celem (*goal*) `create` , podając właściwości dotyczące wartości elementów (identycznie jak w Apache Ant, czyli poprzedzając je `-D`). *Output* polecenia powinien wyglądać następująco (zależnie od tego, czy uruchamiamy Mavena po raz pierwszy na danej maszynie, może zostać pominięte pobieranie domyślnych wtyczek i podstawowych artefaktów, także skrócone poniżej):

```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom >>
>
[INFO]
[INFO] <<< maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom <<
<
[INFO]
[INFO] --- maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom --
-
[INFO] Generating project in Batch mode
[INFO] No archetype defined. Using maven-archetype-quickstart (org.apache.maven.
archetypes:maven-archetype-quickstart:1.0)
Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mav
en-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar
Downloaded: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mave
n-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar (5 KB at 9.3 KB/se
c)
Downloading: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mav
en-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.pom
Downloaded: http://repo.maven.apache.org/maven2/org/apache/maven/archetypes/mave
n-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.pom (703 B at 5.3 KB/s
ec)
[INFO] ------------------------------------------------------------------------
---
[INFO] Using following parameters for creating project from Old (1.x) Archetype:
 maven-archetype-quickstart:1.0
[INFO] ------------------------------------------------------------------------
---
```

```
[INFO] Parameter: groupId, Value: com.mycompany.app
[INFO] Parameter: packageName, Value: com.mycompany.app
[INFO] Parameter: package, Value: com.mycompany.app
[INFO] Parameter: artifactId, Value: my-app
[INFO] Parameter: basedir, Value: C:\Users\Tomek\Dropbox\Dydaktyka\PSOiR\Maven\s
olution\generate
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\Users\Tomek\Dropbox\D
ydaktyka\PSOiR\Maven\solution\generate\my-app
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 4.226s
[INFO] Finished at: Mon Mar 04 13:17:09 CET 2013
[INFO] Final Memory: 12M/108M
[INFO] ------------------------------------------------------------------------
```

W efekcie otrzymujemy gotową strukturę projektu (nazwy katalogów zgodne z podanymi przez nas `groupId` i `artifactId`):

```
my-app
  |-pom.xml
  |-src
      |-main
      |   |-java
      |       |-com
      |          |-mycompany
      |              |-app
      |                  |-App.java
      |-test
          |-java
              |-com
                 |-mycompany
                     |-app
                        |-AppTest.java
```

A wygenerowany plik POM zawiera:

```
<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">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>my-app</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

2. Zdalne repozytorium

Domyślnym repozytorium Mavena jest http://repo1.maven.org/maven2/ (można skonfigurować własną listę). Należy zapoznać się z jego strukturą i rozpoznać grupy, artefakty i wersje, a także typy artefaktów.

3. Lokalne repozytorium

Maven pobiera artefakty i wtyczki do lokalnego repozytorium zlokalizowanego w katalogu domowym użytkownika w podkatalogu *.m2* (*~/.m2*). Należy zapoznać się z jego strukturą i porównać ze strukturą repozytorium zdalnego. Powinny być identyczne, jednak repozytorium lokalne zawiera jedynie artefakty potrzebne do wykonania *buildów* (czyli tylko takie, które chociaż raz zostały użyte). Dzięki temu nie istnienie potrzeba pobierania ich za każdym razem, wykorzystywany jet lokalny *cache*.

4. Prosty projekt

Używając automatycznie wygenerowanego projektu z polecenia 1., należy:

1. Wykonać kompilację kodu (`mvn compile`) i sprawdzić, gdzie znajdują się utworzone pliki *.class*.
2. Wykonać pakietowanie (`mvn package`) i sprawdzić, gdzie znajdują się wygenerowane artefakty. Proszę zwrócić uwagę na automatycznie wykonane test i ich raporty.
3. Wyczyścić projekt (`mvn clean`) i sprawdzić, co zostało usunięte.
4. Wygenerować artefakty bez pakietowania (`mvn jar:jar`) i porównać efekt z pełnym pakietowaniem.
5. Zainstalować artefakty w lokalnym repozytorium (`mvn install`), aby stały się dostępne dla Mavena. Należy sprawdzić, czy artefakty przykładowego projektu znajdują się w odpowiednim miejscu lokalnego repozytorium.
6. Uruchomić przykładową aplikację w oparciu o zainstalowane artefakty (`mvn exec:java -Dexec.mainClass=com.mycompany.app.App`).

5. Projekt wielomodułowy

Do nowego folderu należy ***wyeksportować*** (w *tortoiseSVN* opcja *Export...*) przykładowy projekt wielo-modułowy z repozytorium svn:
http://team.kis.p.lodz.pl:8080/svn/samples/trunk/psoir/maven-multi/

Następnie:

1. Zapoznać się z zawartością pliku POM, rozpoznać zależności i moduły projektu, a także fizyczną organizację plików projektu.
2. Wygenerować artefakty projektu głównego z pominięciem testów (opcja `-Dmaven.test.skip=true`). Proszę zwrócić uwagę na automatyczną kompilację i generację artefaktów poszczególnych modułów.
3. Zainstalować artefakty projekty głównego z pominięciem testów i sprawdzić, czy zostały umieszczone w lokalnym repozytorium.
4. Uruchomić aplikację z projektu *dziecko2*.

6. Wykorzystanie bibliotek

Pracując wciąż na tym samym projekcie, należy usunąć błędy znajdujące się w kodzie, wykorzystując zewnętrzną bibliotekę ***Joda Time*** dostarczającą funkcjonalności związane z datą i czasem. W tym celu należy:

1. Przetestować projekty z poziomu głównego projektu. Zlokalizować test wykonany z niepowodzeniem oraz znaleźć odpowiedzialną za błędy metodę.
2. Kod odpowiedniej metody zastąpić:

```
LocalDate lFromDate = new LocalDate(fromDate);
LocalDate newYear = lFromDate.plusYears(1).withDayOfYear(1);
return Days.daysBetween(lFromDate, newYear).getDays();
```

3. Do klasy należy dodać „importy” wykorzystywanych klas biblioteki ***Joda Time***:

```
import org.joda.time.LocalDate;
import org.joda.time.Days;
```

4. Do pliku POM projektu wykorzystującego bibliotekę ***Joda Time*** należy dodać informacje o zależności podając odpowiednie parametry artefaktu opisujące wersję tego projektu (najlepiej najnowszą). Należy wykorzystać wyszukiwarkę repozytorium Maven.
5. Zainstalować artefakty projekty głównego, zweryfikować stan testów oraz uruchomić ponownie aplikację.