



UNIWERSYTET IM. ADAMA MICKIEWICZA W POZNANIU

Wydział Matematyki i Informatyki

Automatyzacja budowania



Rys historyczny

Potrzeba automatyzacji procesu wynikła dawno temu – proces budowania musi być powtarzalny i zapewniać jakość.

Historycznie rzecz ujmując powstały w tym celu:

- skrypty powłoki
 - make
 - autotools (autoconf, automake, libtool)
 - narzędzia zarządzające także zależnościami (Maven, pip, sbt, stack, ...)
-



Kaçık humoru



Tweet



Tim Martin
@timmartin2



I saw a book entitled "Die GNU Autotools" and I thought "My feelings exactly". Turns out the book was in German.

2:07 PM · Jan 7, 2011 · TweetDeck

2,388 Retweets **31** Quote Tweets **1,327** Likes



W języku Java

Historycznie:

- Ant
 - narzędzie do kompilacji i budowania paczek
 - konfiguracja w pliku XML
 - ant jak antyk

Aktualnie:

- Maven
 - jak wyżej: narzędzie do kompilacji i budowania paczek
 - ale: zarządza zależnościami
 - pozwala dzielić się artefaktami (np. plik JAR)
 - narzuca schematy
-



W języku Java

Aktualnie cd.:

- Maven cd.
 - ma niezliczoną liczbę wtyczek rozszerzających możliwości
 - dostarcza repozytorium z zależnościami
 - konfiguracja dalej w XML
 - gradle
 - napisane w języku Groovy
 - w pewnym sensie bazuje na Mavena (repozytoria, schemat)
 - ma podobne możliwości (gdzieniegdzie większe)
 - plik projektu jest plikiem **wykonywalnym** (tak ma być!)
 - inne: sbt, coursier – przemilczmy
-



Proszę założyć repozytorium

<https://git.wmi.amu.edu.pl/>

Repozytorium o nazwie:

06-ZPRPLI0



Struktura repozytorium

06-ZPRPLI0

- ćwiczenia/
 - 01/
 - 1
 - src/... (za chwilę)
 - 02/...
- projekt/
 - src/...



Ćwiczenia

Ćwiczenia nie są obowiązkowe, ale:

- pomagają przyswoić i zrozumieć materiał
 - liczą się jako aktywność (dodatkowe punkty)
 - code review i (kulturalne!) sugestie dla innych także się liczą
 - Uwaga: sugestie, nie napisanie kodu za daną osobę
 - mogą korzystać z dowolnej technologii, o ile pozwala pokryć materiał
-



Maven

Struktura projektu Maven:

katalog projektu

```
|--- src
|   |--- main
|   |   |-- java
|   |   |-- resources
|   |
|   |___ test
|       |-- java
|       |-- resources
|___ pom.xml
```



Przykładowy plik projektu

```
<?xml version="1.0" encoding="UTF-8"?>
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.example</groupId>
  <artifactId>cw1</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>cw1</name>
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <java.version>11</java.version>
    <maven.compiler.source>${java.version}</maven.compiler.source>
    <maven.compiler.target>${java.version}</maven.compiler.target>
    <maven.compiler.release>${java.version}</maven.compiler.release>

    <junit>5.6.2</junit>

    <!-- Plugin versions -->
    <maven.shade>3.2.2</maven.shade>
    <maven.clean>3.1.0</maven.clean>
    <maven.resources>3.1.0</maven.resources>
    <maven.compiler>3.8.1</maven.compiler>
    <maven.surefire>3.0.0-M5</maven.surefire>
    <maven.jar>3.2.0</maven.jar>
  </properties>
</project>
```

```
[INFO] Parameter: package, value: org.example
[INFO] Parameter: groupId, Value: org.example
[INFO] Parameter: artifactId, Value: cw1
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[WARNING] Don't override file /tmp/archetypetmp/cw1/pom.xml
```



Struktura pom.xml

`groupId`: unikalna nazwa wytwórcy, polecam:

- `pl.edu.amu.<imie_nazwisko>`

`artifactId`: nazwa artefaktu (tu pliku jar), polecam:

- `cw<nr_ćwiczenia>` (np. 1) dla ćwiczeń
- `<nazwa_wymyślonego_projektu>` dla projektu

`version`: numer wersji projektu, polecam:

- `0.1-SNAPSHOT` (konwencja, numer wersji zmienny)
- lub `1.0-RELEASE` (gdy projekt jest gotowy i chcemy go wydać)

`packaging`: jak spakować artefact (do wyboru JAR, WAR, ZIP):

- w praktyce: JAR
-



Podstawowe komendy Maven

`mvn test`

- uruchamia testy

`mvn compile`

- kompiluje projekt

`mvn clean`

- usuwa wcześniejsze wytwory

`mvn package`

- buduje archiwum uruchamiając testy i kompilację

Co robi `mvn clean package`?



Ćwiczenia

1. Tworzenie projektu w IntelliJ Idea z archetypu
 - <https://link.wmi.amu.edu.pl/oKsjus>
 2. Tworzenie projektu w IntelliJ Idea z Spring Initizr
 - <https://link.wmi.amu.edu.pl/ONddlf>
 3. Tworzenie projektu przez kopiowanie
 - <https://link.wmi.amu.edu.pl/ICSHOM>
-



Maven

Literatura:

Maven in 5 minutes

Apache Maven Tutorial

Spring.io: Building Maven Projects with Java



UNIWERSYTET IM. ADAMA MICKIEWICZA W POZNANIU

Wydział Matematyki i Informatyki

Na razie tyle, reszta „przy okazji”