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, ...)

Kącik 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:

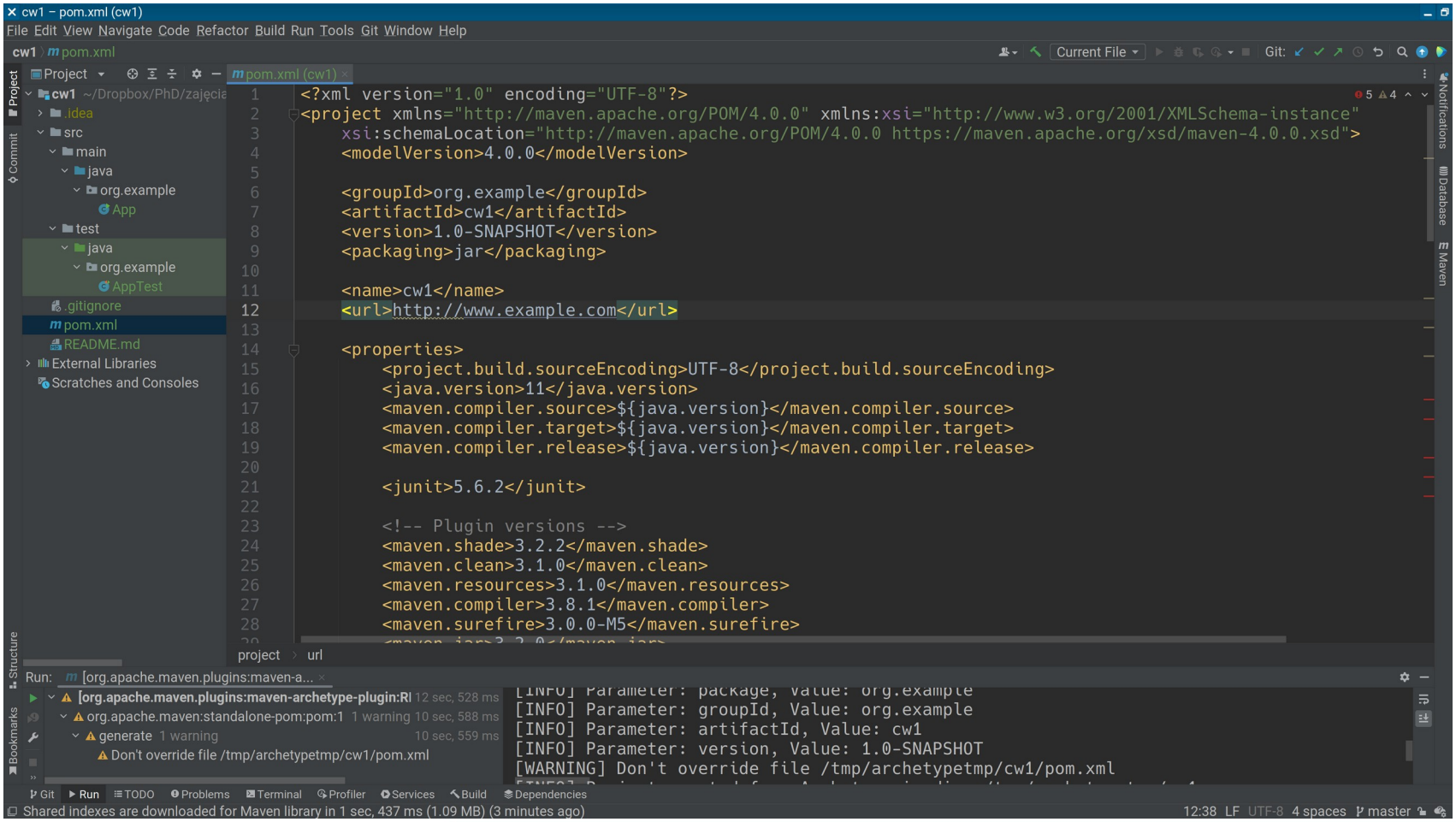
- pomagają przyswoić i zrozumieć materiał
- liczą się jako aktywność (dodatkowe punkty)
- code review i (kulturalne!) sugestie dla innych także się liczą
  - Uwaga: sugestie, nie napisanie kodu za daną osobę
- mogą korzystać z dowolnej technologii, o ile pozwala pokryć materiał

Maven

Struktura projektu Maven:

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

Przykładowy plik projektu



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 Initlzr
   - https://link.wmi.amu.edu.pl/ONddlf
3. Tworzenie projektu przez kopiowanie
   - https://link.wmi.amu.edu.pl/lCSHOM

Maven

Literatura:

Maven in 5 minutes
Apache Maven Tutorial
Spring.io: Building Maven Projects with Java

Na razie tyle, reszta „przy okazji”