Budowa aplikacji webowej w oparciu o Maven2 oraz przykłady testów jednostkowych.

Wykonał Marcin Gadamer

Maven 2 – podstawowe informacje

- **Apache Maven** jest narzędziem automatyzującym budowę oprogramowania na platformę Java

- Wtyczki są pobierane automatycznie przy pierwszym wykorzystaniu
- Sposób budowy aplikacji zawarty jest w pliku POM (ang. *Project Object Model*)

- Apache Maven2 jest wydany na **The Apache Software License, Version 2.0**

Maven 2 – źródła oraz instalacja

- Najnowsze źródła oraz szczegółowy opis instalacji można znaleść pod adresem:
  - http://maven.apache.org/download.html

Maven 2 – cykl życia

- **validate** - sprawdzenie, czy projekt jest poprawny i czy wszystkie niezbędne informacje zostały określone
- **compile** - kod źródłowy jest kompilowany
- **test** - przeprowadzane są testy jednostkowe
- **package** - budowana jest paczka dystrybucyjna
- **integration-test** - zbudowany projekt umieszczany jest w środowisku testowym, gdzie przeprowadzane są testy integracyjne
- **verify** - sprawdzenie, czy paczka jest poprawna
- **install** - paczka umieszczana jest w repozytorium lokalnym - może być używana przez inne projekty jako zależność
- **deploy** - paczka umieszczana jest w repozytorium zdalnym (opublikowana)

Maven 2 – *Project Object Model*

- **POM**, czyli *Project Object Model*, to dokument xml kompleksowo opisujący projekt. POM nie tylko precyzuje szczegóły budowy produktu, ale też może przechowywać informacje o zespole programistów, zastosowanych systemach wspomagających rozwój oprogramowania.

- Szczegółowy opis tworzenia pliku pom.xml można znaleźć pod adresem:
  - http://maven.apache.org/guides/introduction/introduction-to-the-pom.html

Maven 2 budowa podstawowej aplikacji

- Wydajemy polecenie:

```
mvn archetype:generate
   -DgroupId=pl.gadamer.maven.firstApp
   -DartifactId=FirstApp
   -Dversion=1.0
   -DarchetypeArtifactId=maven-archetype-
                                  quickstart
```

- **I czekamy…**

…Maven zajmie się resztą

Maven 2 – budowa aplikacji webowej cz I

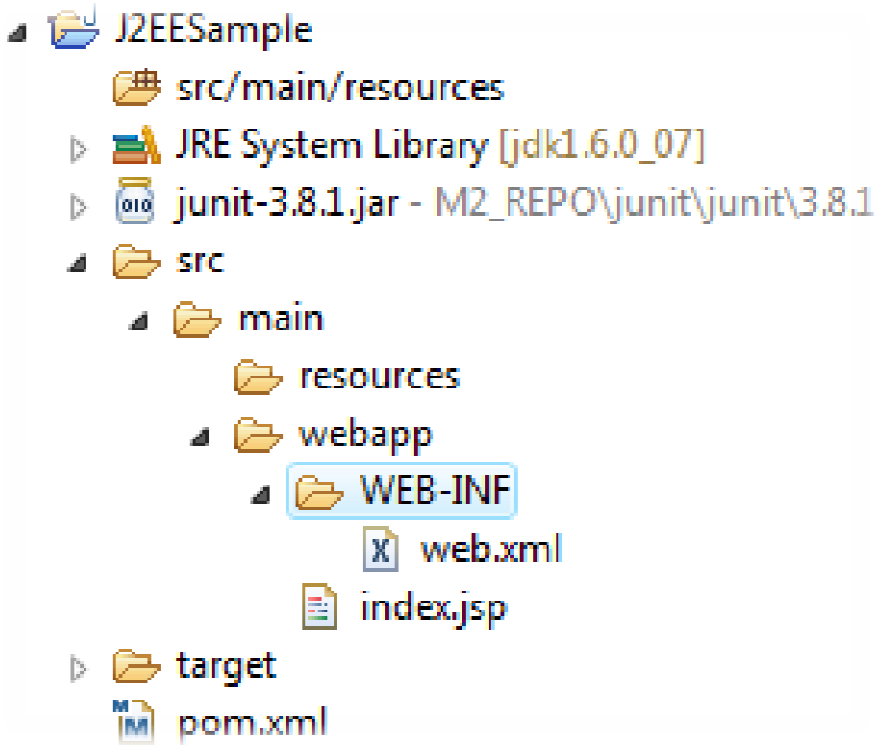
- Wydajemy polecenie:

```
mvn archetype:generate
    -DgroupId=pl.gadamer.maven.firstApp
    -DartifactId=FirstJ2EEApp
    -Dversion=1.0
    -DarchetypeArtifactId=maven-archetype-webapp
```

- **I czekamy…**

Maven 2 – budowa aplikacji webowej cz II

- Stworzona zostanie taka struktura katalogów:



Maven 2 - podsumowanie

- **Apache Maven** jest narzędziem automatyzującym budowę oprogramowania na platformę Java
- Zarządzanie zależnościami projektu w jednym pliku
- Cały etap budowy aplikacji w jednym poleceniu
  - Kompilacja,
  - Testy,
  - Budowa paczki

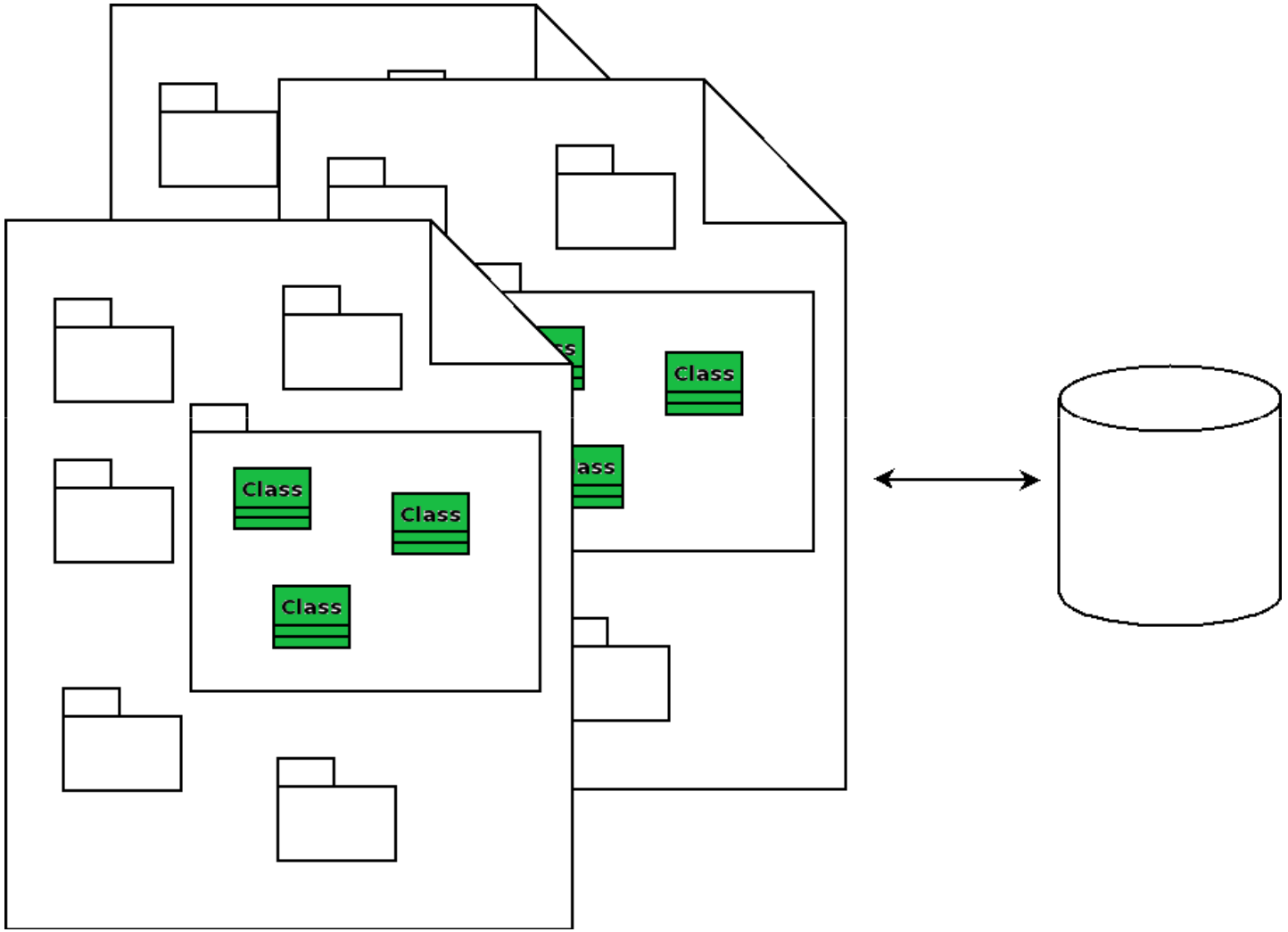
Testy jednostkowe – wszyscy testujemy

- Jak testujemy?
  - System.out.println / cout / loggery: log4j
  - debugujemy
  - przeklikujemy się
- problemy:
  - jednorazowe, niepowtarzalne,
  - niemierzalne,
  - brak precyzyjnego określenia miejsca,
  - wystąpienia błędu w kodzie,
  - ręczne,
  - kosztowne czasowo,
  - oddzielone od procesu budowania projektu

Rodzaje testów

- testy:
  - **Jednostkowe,**
  - Integracyjne,
  - modułowe**,**
  - funkcjonalne,
  - systemowe,
  - użytkownika,
  - akceptacyjne,
  - itd.

Testy jednostkowe – czyli gdzie?



Testy – do czego one służą?

- procent = czesc/calosc;
- x.getY().getZ().doSth();
- offset = +perPage;
- if (cos > cosInnego)

- procent = czesc/0;
- x.getY().null.doSth();
- offset += perPage;
- if (cos >= cosInnego)

Testy – po co one w projekcie?

- znajdź błędy nim one cię znajdą,
- bezproblemowy refaktoring,
- szybsze tworzenie kodu,
- spokojny sen,
- dokumentacja,
- wyższa jakość kodu,
- testy jako sposób na design

Test jednostkowy - cechy

- testujemy zachowania klas,
- wyniki testów mierzalne,
- izolacja / niezależność
  - od innych elementów systemu
  - od innych testów
- prostota,
- wydajność,
- wysoka jakość,
- automatyczne,
- szybkie,
- powtarzalne,
- włączone w proces budowania projektu

Co testować?

- wartości **spodziewane**
  - Kilka
- wartości **brzegowe**
  - dokładnie !
- wartości **“tego na pewno nie wpisze user”**

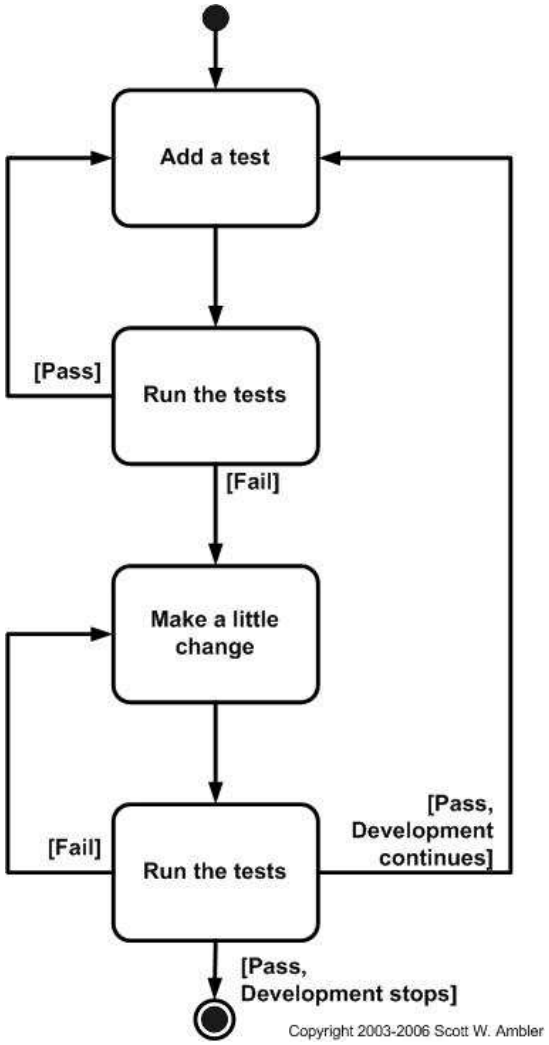
Kiedy pisać test ?

- zawsze przed napisaniem kodu (TDD – *Test-Driven Development*)

- gdy pojawi się błąd,
- gdy dodajemy nową funkcjonalność

- jeżeli masz trudność z napisaniem testu, to znaczy że napisałeś zły kod !

TDD – najlepsze co może być!



Testy - obiekty współpracujące

- potrzebuję tylko żeby ten obiekt pomocniczy istniał (*Dummy*)
- potrzebuję żeby obiekt pomocniczy coś mi dał (*Stub*)
- potrzebuje sprawdzić czy testowany kod wywołuje odpowiednie metody obiektu pomocniczego (*Mock / TestSpy*)

Podsumowanie testów

- testy jednostkowe testują klasy w IZOLACJI,
- testy jednostkowe podstawą życia kodu,
- testowanie stanu to banał,
- testowanie zachowania bywa interesujące,
- obiekty współpracujące
  - dummy,
  - stuby,
  - mocki

Dziękuję za uwagę

- Proszę o pytania/spostrzeżenia/uwagi

Marcin Gadamer
marcin.gadamer@gmail.com