BOTTEGA IT minds

Program szkolenia:

Tworzenie i testowanie aplikacji z użyciem Test Driven Development (TDD) w Spock (lub JUnit 5)

Informacje:

Nazwa:	**Tworzenie i testowanie aplikacji z użyciem Test Driven Development (TDD) w Spock (lub JUnit 5)**
Kod:	**craft-test-TDD**
Kategoria:	Testowanie automatyczne
Odbiorcy:	developerzy
Czas trwania:	3-4 dni
Forma:	40% wykłady / 60% warsztaty

Szkolenie prezentuje techniki tworzenia i testowania aplikacji z użyciem Test Driven Development (TDD). Dodatkowo uczestnicy zdobędą/rozszerzą wiedzę dotyczącą programowania w parach, skutecznego refaktoringu, separacji zależności (mockowania) czy testowania integracyjnego.

Materiały wstępne

Przed szkoleniem możesz zapoznać się z serią naszych artykułów: Testowanie automatyczne.

Zalety szkolenia:

- nacisk na praktyczne "poczucie" tematu
- efektywne techniki testowania
- dobre praktyki i wzorce
- nowoczesne narzędzia: Spock lub JUnit 5

BOTTEGA IT minds

Szczegółowy program:

1. Jednostkowe testowanie kodu - niezbędnik do pisania dobrych testów w TDD

- 1.1. dobre testy jednostkowe - FIRST
- 1.2. struktura testów jednostkowych (given-when-then/arrange-act-assert)
- 1.3. nazewnictwo testów (metod testowych)
- 1.4. konstrukcje wspierające testowalność kodu (m.in. dziedziczenie -> kompozycja, małe klasy, wstrzykiwanie zależności)
- 1.5. antywzorce dla testowalnego kodu (m.in. singletony, elementy statyczne, pola i metody final)
- 1.6. zadania frameworku testowego
- 1.7. formułowanie naturalnie wyglądających asercji (AssertJ)
- 1.8. ćwiczenia praktyczne (w trakcie - przeplatane z teorią)

2. Spock Framework - pisanie testów szybciej, zwięźlej i czytelniej (alternatywnie JUnit 5)

- 2.1. nowa jakość w testowaniu kodu - dlaczego warto poznać Spocka
- 2.2. konstrukcje i techniki upraszczające kod testowy (Groovy w pigułce na potrzeby pisania testów)
- 2.3. podział testu na bloki/sekcje
- 2.4. testy parametryzowane
- 2.5. testowanie wyjątków
- 2.6. warunkowe wykonywanie testów
- 2.7. inicjowanie i sprzątanie w testach
- 2.8. porównanie trio JUnit/Mockito/AssertJ ze Spockiem
- 2.9. efektywne przejście w projekcie z JUnit/TestNG na Spocka
- 2.10. ćwiczenia praktyczne (w trakcie - przeplatane z teorią)

3. Test Driven Development

- 3.1. historia TDD

BOTTEGA IT minds

- 3.2. podstawowe założenia
- 3.3. cykl red-green-refactor
- 3.4. przejrzysta struktura testu
- 3.5. wybór kolejnych funkcji do zaimplementowania
- 3.6. sprawne uruchamianie testów z IDE (przydatne wtyczki, skróty klawiaturowe, ciągłe uruchamianie testów)
- 3.7. ćwiczenia praktyczne z TDD w formie Kata/Coding Dojo
- 3.8. korzyści ze stosowania TDD

4. Bezpieczny refaktoring

- 4.1. cel refaktoringu
- 4.2. przydatne przekształcenia kodu
- 4.3. automatyczny refaktoring w IDE z użyciem skrótów klawiaturowych (Idea lub Eclipse)
- 4.4. ćwiczenia praktyczne - Golf refaktoryzacyjny

5. Stosowanie separacji od obiektów współpracujących przy TDD

- 5.1. potrzeba, kiedy i dlaczego warto
- 5.2. testowe zastępniki obiektów współpracujących
- 5.3. mockowanie ze Spock Framework
- 5.4. mockowanie z Mockito (mocki w Spocku mają swoje ograniczenia)
- 5.5. ćwiczenia praktyczne (w trakcie - przeplatane z teorią)

6. TDD i testowanie integracyjne (Spring Framework)

- 6.1. dlaczego tylko testy jednostkowe nie wystarczą
- 6.2. wsparcie w Spock/JUnit
- 6.3. konfiguracja i zarządzanie kontekstem Springa
- 6.4. wstrzykiwanie zależności
- 6.5. odcinanie zależności w kontekście Springa (wstrzykiwanie zależności testowych, w tym mocków)

6.6. ręczne tworzenie kontekstu w teście (do testowania specyficznych przypadków)

6.7. cache'owanie kontekstu między testami (i problemy z tym związane)

6.8. wydzielanie wspólnej konfiguracji dla wielu testów

6.9. ćwiczenia praktyczne (w trakcie - przeplatane z teorią)

7. Test Driven Development - dobre praktyki i techniki unikania typowych problemów

7.1. małe kroki a efektywność pracy

7.2. TDD i programowanie w parach (ang. pair programming)

7.3. test-first vs test-last

7.4. wykorzystanie TDD do pracy nad dobrym designem systemu

7.5. wybór właściwego poziomu odcięcia zależności

7.6. wykorzystanie TDD/BDD na poziomie testowania akceptacyjnego (wstęp)

7.7. wprowadzenie TDD w organizacji

7.8. zestaw ćwiczeń praktycznych uwypuklających typowe problemy spotykane przy tworzeniu kodu produkcyjnego i testów z pomocą TDD oraz pokazanie sposób ich unikania /rozwiązywania

8. Testowanie - zaawansowane zagadnienia (podzbiór ustalany indywidualnie)

8.1. techniki testowania asynchronicznych wywołań (Awaitility, Spock)

8.2. badanie jakości testów z testowaniem mutacyjnym (PIT)

8.3. testowanie kodu wykorzystującego mechanizm feature toogle

8.4. techniki przyśpieszania testów

8.5. uproszczenie niektórych aspektów testowania z Java 8

8.6. testowanie z losowymi danymi wejściowymi

8.6.1. powstarzalność w testach z danymi losowymi

8.6.2. property based testing - QuickCheck

8.7. zaawansowany Spock

8.7.1. tworzenie własnych rozszerzeń

BOTTEGA IT minds

- 8.8. miarodajne microbeanchmarki
  - 8.8.1. kiedy i dlaczego warto stosować
  - 8.8.2. naiwne podejście i jego znikoma przydatność
  - 8.8.3. optymalizacja kodu w JVM
  - 8.8.4. ćwiczenia praktyczne z JMH (Java Microbenchmarking Harness)
- 8.9. testowanie reaktywnego kodu RxJava
- 8.10. system legacy bez testów - jak do tego podejść?
- 8.11. testowanie architektury Event-Driven
- 8.12. testowanie architektury mikroserwisów - TDD na poziomie architektury
- 8.13. consumer-Driven-Contracts
- 8.14. testowanie projektów stosujących Event Sourcing