Тестирование Java-программ

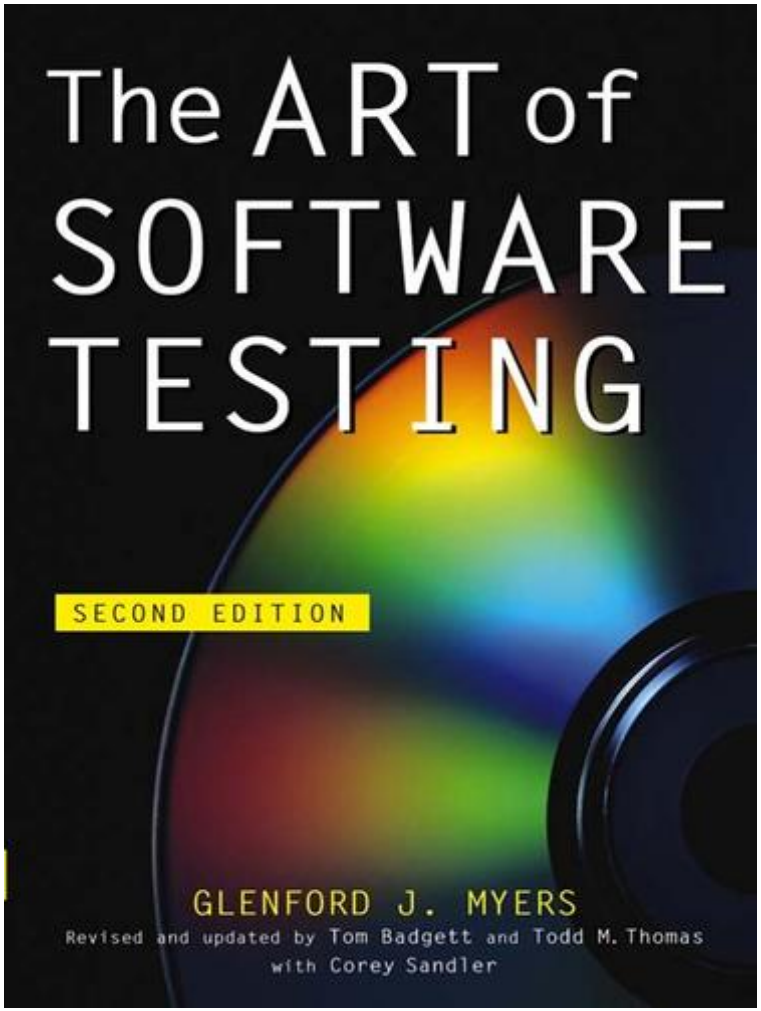
Алексей Владыкин

10 ноября 2014

1. Основные идеи

2. Модульное тестирование
   - JUnit
   - FEST-Assert
   - Mockito
   - JaCoCo

3. Тестирование производительности
   - JMH



Фундаментальная книга по тестированию

Виды тестирования

- Модульное тестирование — проверка работы программы на уровне отдельных модулей (классов, методов)
- Интеграционное тестирование — проверка совместной работы нескольких модулей
- Системное тестирование — проверка работы системы в целом

Виды тестирования

- Функциональное тестирование
- Тестирование производительности
- Тестирование удобства использования
- Тестирование безопасности

Виды тестирования

- Тестирование «черного ящика»
- Тестирование «белого ящика»


FAILURE?
MY CODE DOESN'T FAIL!
I'M A PRO-FESSIONAL

Инструменты модульного тестирования

- Инфраструктуры для написания и запуска тестов
  JUnit, TestNG
- Библиотеки проверок
  FEST Assert, Hamcrest, XMLUnit, HttpUnit
- Библиотеки для создания тестовых дублеров
  Mockito, JMock, EasyMock

- “JUnit is a simple, open source framework to write and run repeatable tests.”
- http://junit.org/

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.11</version>
    <scope>test</scope>
</dependency>
```

- mvn test

```java
package ru.csc.java2014.testing;

import org.junit.Test;
import static org.junit.Assert.*;

public class StringTest {

    @Test
    public void substring() {
        assertEquals("llo", "Hello".substring(3));
    }
}
```

```
org.junit.ComparisonFailure: expected:<l[l]o> but was:<l[]o>
    at org.junit.Assert.assertEquals(Assert.java:125)
    at org.junit.Assert.assertEquals(Assert.java:147)
    at ru.csc.java2014.testing.StringTest.substring(StringTest.java:10)
    ...
```

org.junit.Assert

- assertTrue
  assertFalse
- assertEquals
  assertArrayEquals
  assertNotEquals
- assertSame
  assertNotSame
- fail
- Варианты с текстом ошибки и без

assert

```
assert "llo".equals("Hello".substring(3));

assert 1 == 1 : "Arithmetics broken";
```

- Поддерживаются только булевские условия
- В исключении нет описания проблемы
- Надо включать флагом JVM `-ea`

Структура теста

- (Given) Подготовка тестового окружения
- (When) Выполнение тестового сценария
- (Then) Проверки

Жизненный цикл теста

- `@BeforeClass`
- Для каждого `@Test`-метода:
  - создание экземпляра тестового класса
  - `@Before`
  - `@Test`
  - `@After`
- `@AfterClass`

Test Driven Development

1. Пишем простейший тест, ломающий программу
2. Пишем простейшую реализацию, достаточную для прохождения теста
3. Улучшаем написанный код, не ломая тесты. Возвращаемся к пункту 1

- “FEST-Assert provides a fluent interface for assertions.”
- https://github.com/alexruiz/fest-assert-2.x

```
<dependency>
    <groupId>org.easytesting</groupId>
    <artifactId>fest-assert-core</artifactId>
    <version>2.0M10</version>
    <scope>test</scope>
</dependency>
```

- “Mockito is a mocking framework that tastes really good.”
- https://code.google.com/p/mockito/

```
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>1.10.0</version>
    <scope>test</scope>
</dependency>
```

- “JaCoCo is a free Java code coverage library”
- http://www.eclemma.org/jacoco/trunk/index.html

```
<plugin>
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>0.7.2.201409121644</version>
</plugin>
```

- “JMH is a Java harness for building, running, and analysing nano/micro/milli/macro benchmarks written in Java and other languages targetting the JVM.”
- http://openjdk.java.net/projects/code-tools/jmh/

Что сегодня узнали

- Важно писать модульные тесты
- Существует достаточно библиотек, помогающих в этом деле
- Можно тестировать не только корректность, но и производительность