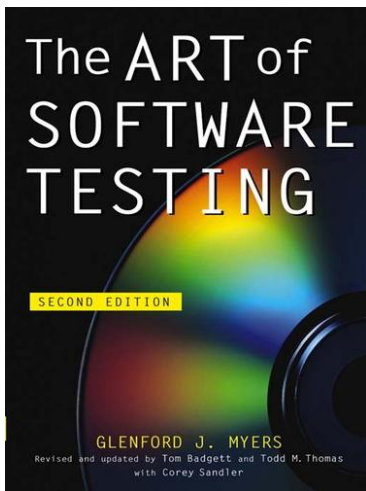


# Модульное тестирование на Java

Алексей Владыкин

11 ноября 2013

- 1 Основные идеи
- 2 JUnit
- 3 Java Logging API



Фундаментальная книга по тестированию

# Уровни 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

# 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

# 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



```
package ru.compscicenter.java2013.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:<1[1]o> but was:<1[]o>  
  at org.junit.Assert.assertEquals(Assert.java:125)  
  at org.junit.Assert.assertEquals(Assert.java:147)  
  at ru.compscicenter.java2013.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

# FEST Assert

DEMO

# Mockito

DEMO





- Пакет `java.util.logging`
- Центральная сущность — `java.util.logging.Logger`
- Логгеры образуют иерархию

```
Logger logger = Logger.getLogger(  
    "ru.compscicenter.java2013.logging");  
  
logger.info("I'm logging!");
```

- Уровни логирования:  
SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST

```
logger.setLevel(Level.INFO);  
  
logger.fine("I'm still logging");  
  
logger.log(Level.WARNING,  
           "Houston, we have a problem!");
```

# java.util.logging.Handler

- Обработчик сообщения  
Определяет, куда будет записано сообщение
- `java.util.logging.ConsoleHandler`
- `java.util.logging.FileHandler`

# java.util.logging.Formatter

- Оформитель сообщения  
Определяет формат вывода
- `java.util.logging.SimpleFormatter`
- `java.util.logging.XMLFormatter`

## Что сегодня узнали

- Важно писать модульные тесты
- Существует достаточно библиотек, помогающих в этом деле
- Есть простой стандартный API для логирования