LUXOFT TRAINING

Разработка через тестирование BDD

Test Driven Development

Ivan Dyachenko <IDyachenko@luxoft.com>

Module 7

Содержание

LUXOFT Training Center

Что такое BDD?

LUXOFT Training Center

BDD – это правильное TDD

Что такое BDD?

Разработка, основанная на тестировании (Test-Driven Development) – это отличная практическая идея, но некоторые разработчики не могут преодолеть концептуальную пропасть, с которой у них ассоциируется слово тестирование.

Что такое BDD?

LUXOFT Training Center

- Тестирование, выполняемое разработчиком – это очевидно полезная вещь.
- Тестирование на ранней стадии, например, во время написания кода – еще лучше, особенно когда оно приводит к повышению качества кода.
- Напишите тесты заранее – и вы имеете шанс выиграть «голубую ленту» победителя регаты.

Что такое BDD?

Дополнительные возможности для проверки функционирования кода и его предварительной отладки, без всякого сомнения, повышают скорость разработки

Что такое BDD?

Но даже зная все это, мы еще очень далеки от того времени, когда написание тестов до написания кода станет общим стандартом.

Точно так же как TDD стало следующим этапом эволюции развития экстремального программирования (eXP) и выдвинуло на первый план инфраструктуры для unit-тестирования, следующий скачок эволюции будет сделан с того уровня, где находится TDD.

BDD

LUXOFT Training Center

Dan North

Behavior Driven Development (BDD)

Проблемы TDD

LUXOFT Training Center

У меня была проблема. При использовании и обучении Test-Driven Development (TDD) в разных проектах, я каждый раз сталкивался с непониманием со стороны разработчиков.

- Dan North

Introducing BDD - Dan North
http://dannorth.net/introducing-bdd

Что такое BDD?

Программисты не понимают:

- С чего начать писать тест?
- Что тестировать а что нет?
- Как много тестировать в одном тесте?
- Как именовать тесты?
- Как понять, почему тест не пройден?

Что такое BDD?

LUXOFT Training Center

“The deeper I got into TDD, the more I felt that my own journey had been less of a wax-on, wax-off process of gradual mastery than a series of blind alleys. I remember thinking ‘If only someone had told me that!’ far more often than I thought ‘Wow, a door has opened.’

I decided it must be possible to present TDD in a way that gets straight to the good stuff and avoids all the pitfalls.”

Что такое BDD?

LUXOFT Training Center

“My response is behaviour-driven development (BDD). It has evolved out of established agile practices and is designed to make them more accessible and effective for teams new to agile software delivery.

Over time, BDD has grown to encompass the wider picture of agile analysis and automated acceptance testing.”

Основная задача модульных тестов

LUXOFT Training Center

- Проверить правильность работы того или иного аспекта модуля, и обнаружить дефекты в программном обеспечении
- Юнит тест представляет собой смесь состоящую из бизнес задач и механизмов тестирования этих задач
- Юнит тест позволяет проверить то, что созданное ПО работает корректно, но отнюдь не то, что ПО является корректным.

Executable specifications

LUXOFT Training Center

- Идея BDD подхода в описании требований (спецификаций), которые могут быть выполнены - executable specifications.
- Основное отличие от юнит тестов в том, что юнит тест верифицирует правильность работы модуля, а executable specification - описывает желаемое поведение модуля

Given-when-then

LUXOFT Training Center

- Given-when-then
- Four Phase Test Approach
- Arrance-Act-Assert

Шаблон теста

LUXOFT Training Center

```
Test...
{
    // Arrange
    ...
    // Action
    ...
    // Assetion
    ...
}
```

```
Should...
{
    // Given
    ...
    // When
    ...
    // Then
    ...
}
```

Scenarios and context/specification style

Context/Specification стиль предлагает рассматривать поведения не как плоский список, а фокусироваться на различных сценариях, в рамках которых проявляются те или иные поведения

Flavors of BDD approach

BDD подход можно разделить на два вида: xSpec и xBehave.

- xSpec - это применение BDD на уровне модулей (unit level), то есть описание спецификаций по отношению к модулям
- xBehave - это применение BDD для описания высокоуровневых пользовательских историй (user stories) в виде приемочных критериев (acceptance criteria) при помощи given-when-then синтаксиса.

xSpec

LUXOFT Training Center

Specification-oriented BDD Frameworks

До конца этого доклада будем рассматривать xSpec подход

xSpec

LUXOFT Training Center

- Обычно используются те же технические инструменты, что и для TDD
- Некоторые BDD framework'и предлагают описывать спецификации, на основании которых потом генерируется код тестов
- Unit-level спецификации применимы на этапе дизайна и реализации ПО
- Отличительной чертой является выделение DSL и использование библиотек для написания тестов с использованием DSL

Эволюция TDD

LUXOFT Training Center

Test - Driven Development

BDD

LUXOFT Training Center

↓ Действительно ли мы говорим о тестах?

Test - Driven Development

BDD

Нет – обычно мы говорим о **Software Design**

~~Test~~ - Driven Development

BDD

LUXOFT Training Center

Behavior

~~Test~~ - Driven Development

BDD

BDD был представлен как набор соглашений поверх TDD:

- Тестовые методы должны быть предложением

```
void testFindCustomerById()
void testFailsForDuplicateCustomers()
```

- Тестовые методы должны начинаться с “should”

```
void shouldFindCustomerById()
void shouldFailsForDuplicateCustomers
```

- TastCase class должен быть существительным в предложении тестового метода

```
public class CustomerTableTest {
    @Test
    public void shouldFindCustomerById() {

    }
}
```

BDD Assertions

LUXOFT Training Center

TESTing (TDD)

```
assertEquals(expected, actual)
assertGreaterThan(expected, actual)
assertInstanceOf(clazz, actual)
```

Describing (BDD)

```
actual should be Equals to expected
actual should be GreaterThen expected
actual should be InstanceOf clazz
```

BDD Assertions

LUXOFT Training Center

- Что бы полноценно использовать BDD нужно, что бы язык программирования поддерживал созданиеDomain Specific Language
- В этом плане Java не подходит для написания тестов используя specification based Domain Specific Language
- Но с этой задачей отлично справляются JVM based языки программирования такие как Groovy & Scala

Пример на Java (Instinct)

LUXOFT Training Center

```
import static com.googlecode.instinct.expect.Expect.expect;

class AnEmptyStack {
    ...
    @Specification
    void isEmpty() {
        expect.that(stack.isEmpty()).isTrue();
    }
    ...
}
```

Пример на Scala

LUXOFT Training Center

```
import collection.mutable.Stack
import org.scalatest._

class StackSpec extends FlatSpec with ShouldMatchers {

  "A Stack" should "pop values in last-in-first-out order" in {
    val stack = new Stack[Int]
    stack.push(1)
    stack.push(2)
    stack.pop() should equal (2)
    stack.pop() should equal (1)
  }

  it should "throw NoSuchElementException if an empty stack is popped" in {
    val emptyStack = new Stack[String]
    evaluating { emptyStack.pop() } should produce [NoSuchElementException]
  }
}
```

BDD Frameworks

LUXOFT Training Center

Framework	Good	Bad
Instinct	ASF license	Mar 2010
JDave	Jan 2011, ASF license	comes with matchers and mocks
easyb	Groovy, Story+Specs	Oct 2010
beanspec		2007
bdoc		Jan 2010
spock	Very active, very cool	Bit extreme
jbehave	The "mother" of all BDD in Java	

Spock Framework

LUXOFT Training Center

Why Spock?

LUXOFT Training Center

```
import spock.lang.*

class EmptyStack extends Specification {
  def stack = new Stack()

  def "size"() {
    expect: stack.size() == 0
  }

  def "pop"() {
    when: stack.pop()
    then: thrown(EmptyStackException)
  }

  def "peek"() {
    when: stack.peek()
    then: thrown(EmptyStackException)
  }

  def "push"() {
    when:
    stack.push("elem")

    then:
    stack.size() == 1
    stack.peek() == "elem"
  }
}
```

Why Spock?

LUXOFT Training Center

- Easy to learn
- Powered by Groovy
- Eliminates waste
- Detailed information
- Designed for use
- Open-minded
- Beautiful language
- Extensible for everyone
- Compatible with Junit
- Learns from the history

Specification

```
class MyFirstSpecification extends Specification {
    // fields
    // fixture methods
    // feature methods
    // helper methods
}
```

Спецификация должна быть унаследована от spock.lang.Specification. Она может содержать поля, установочные методы(fixture methods), сценарии требований(feature methods), вспомогательные методы(helper methods).

Fields

LUXOFT Training Center

```
def obj = new ClassUnderSpecification()
def coll = new Collaborator()


@Shared res = new VeryExpensiveResource()
```

Instance fields are a good place to store objects belonging to the specification's fixture. It is good practice to initialize them right at the point of declaration.

Fixture Methods

LUXOFT Training Center

```
def setup() {}          // run before every feature method
def cleanup() {}        // run after every feature method
def setupSpec() {}      // run before the first feature method
def cleanupSpec() {}    // run after the last feature method
```

Установочные методы это:

- setup() — аналог @Before в JUnit, выполняется перед каждым сценарием
- cleanup() — аналог After в JUnit, выполняется после каждого сценария
- setupSpec() — аналог @BeforeClass в JUnit, выполняется до первого сценария в спецификации
- cleanupSpec() — аналог AfterClass в JUnit, выполняется после последнего сценария в спецификации

Feature Methods


```
def "pushing an element on the stack"() {
  // blocks go here
}
```

Blocks

- **setup** — то же самое что и установочный метод setup(), только применяется к конкретному сценарию. Обязательно должен находится до остальных блоков и не должен повторяться. Метка setup может отсутствовать, также вместо setup можно писать given, это сделано для лучшей читаемости(given-when-then)

- **cleanup** — то же самое что и метод cleanup(), только применяется к конкретному сценарию. Обязательно должен находиться в конце сценария до блока where, если он есть, и не должен повторяться

- **when-then** — это причина и условие выполнения. В when части обычно объявляются переменные, выполняются необходимые действия, в then части проверяются некоторые условия. Это могут быть проверки условий, проверка выброса исключения либо ожидания выполнения некоторых методов у мок-объектов. Данный блок может повторяться, но авторы фреймворка рекомендуют не увлекаться, хороший сценарий должен содержать от 1 до 5 таких блоков

- **expect** — это упрощенный when-then блок, где действие и проверка находятся в одном выражении

- **where** — это аналог @DataProvider из TestNG, предназначен для создания набора данных для теста

Extensions

LUXOFT Training Center

- **IMethodInterceptor, IMethodInvocation** — первый для проксирования методов спецификации, позволяет добавлять свой код до и после вызова метода, для упрощения работы можно воспользоваться классом AbstractMethodInterceptor. Второй доступен из первого, служит для работы с оригинальным(проксируемым) методом

- **IGlobalExtension** — позволяет работать с метаданными спецификации(SpecInfo), здесь можно посмотреть метаданные по любым компонентам спецификации(поля, установочные методы, сценарии требований) и добавить им свои интерсепторы

- **IAnnotationDrivenExtension** — тоже что и предыдущий, только упрощает задачу, если наше расширение привязано к какой-то конкретной аннотации, для упрощения работы можно воспользоваться классом AbstractAnnotationDrivenExtension

Bowling Game Kata

LUXOFT Training Center

```
package kata

import spock.lang.Specification

class BowlingSpecification extends Specification {

    BowlingGame game

    def setup() {
        game = new BowlingGame()
    }

    def "should score zero on gutter game"() {
        when: rollMany 20, 0
        then: game.score() == 0
    }

    def "should score twenty when all one"() {
        when: rollMany 20, 1
        then: game.score() == 20
    }

    def "should score spare correctly"() {
        when:
        rollSpare()
        game.roll 3
        rollMany 17, 0

        then:
        game.score() == 16
    }
```

Bowling Game Kata

```
def "should score strike correctly"() {
        when:
        rollStrike()
        game.roll 4
        game.roll 3
        rollMany 16, 0

        then:
        game.score() == 24
    }

    def "should score perfect game"() {
        when: rollMany 12, 10
        then: game.score() == 300
    }

    void rollMany(int n, int pins) {
        (1..n).each { game.roll pins }
    }

    void rollSpare() {
        game.roll 5
        game.roll 5
    }

    void rollStrike() {
        game.roll 10
    }

}
```

Workshop

LUXOFT Training Center

Преимущества BDD

LUXOFT Training Center

- Уменьшение количества сценариев по отношению к количеству юнит-тестов.
- Поведения группируются относительно сценариев в которых они проявляются. Легче стало проверять одно и тоже поведение, которое проявляется в разных сценариях
- Легче стало применять spec(test)-first подход. Описания спецификаций меньше привязаны API модуля, чем юнит тесты
- Спецификации легче в сопровождении. Возвращаясь через некоторое время к спецификации, бывает достаточно прочитать ее описание для понимания поведения, и реже для этого приходиться лезть в код модуля

BDD это ...

LUXOFT Training Center

BDD IS A
MINDSET
NOT A TOOLSET

Вопросы ?

Разработка через тестирование

IDyachenko@luxoft.com

git clone git://github.com/ivan-dyachenko/Trainings.git

https://github.com/ivan-dyachenko/Trainings