JUnit Assertions

JSP & Servlets
3 уровень, 5 лекция

ОТКРЫТА

6.1 assert’ы

Ассерты (asserts) — это **специальные проверки**, которые можно вставить в разные места кода. Их задача определять, что что-то пошло не так. Вернее, проверять, что все идет как нужно. Вот это “как нужно” они и позволяют задать различными способами.

С некоторыми ассертами ты уже сталкивался в коде выше. Первый из них – проверка объектов на равенство. Если объекты не равны — кинется исключение и тест будет провален.

Тут важен порядок сравнения, ведь JUnit в итоговом отчете напишет что-то типа “получено значение 1, а ожидалось 3”. Общий формат такой проверки имеет вид:

```
assertEquals(эталон, значение)
```

Пример:

```
@Test
public void whenAssertingEquality () {
    String expected = "3.1415";
    String actual = "3";

    assertEquals(expected, actual);
}
```

6.2 Методы assertEquals, assertTrue, assertFalse

Ниже я приведу список самых популярных методов — ассертов. По их названиям вполне можно догадаться как они работают. Но все же напишу краткое пояснение:

assertEquals	Проверяет, что два объекта равны
assertArrayEquals	Проверяет, что два массива содержат равные значения
assertNotNull	Проверяет, что аргумент не равен null
assertNull	Проверяет, что аргумент равен null
assertNotSame	Проверят, что два аргумента — это не один и тот же объект
assertSame	Проверят, что два аргумента — это один и тот же объект
assertTrue	Проверяет, что аргумент равен *true*
assertFalse	Проверяет, что аргумент равен *false*

Некоторые из этих методов кажутся излишними. Зачем использовать `assertSame(a, b)`, если можно просто написать `assertTrue(a == b)`?

Все дело в том, что `assert` — это очень умный метод. Он делает много чего полезного и, в том числе, пишет в лог информацию об ошибке. В первом случае он напишет, что ожидался объект А, а получен объект Б. Во втором случае просто напишет, что ожидалось *true*.

Когда у тебя сотни тестов, особенно выполняемые на специальном тестовом сервере, то наличие детальных логов может быть суперполезным. Думаю,ты понимаешь, о чем я.

Пример сравнения массивов:

```
@Test
public void whenAssertingArraysEquality() {
    char[] expected = {'J','u','n','i','t'};
    char[] actual = "Junit".toCharArray();

    assertArrayEquals(expected, actual);
}
```

6.3 Метод assertAll

Как уже говорилось выше, метод `assert` не просто выполняет проверку, но и пишет в лог много информации о сравнимых объектах.

Допустим выполняется сравнение:

```
Address address = unitUnderTest.methodUnderTest();
assertEquals("Вашингтон", address.getCity());
assertEquals("Oracle Parkway", address.getStreet());
assertEquals("500", address.getNumber());
```

Но если один из параметров не совпадет, то проверки остальных не произойдет. А хотелось бы чтобы они все-таки произошли и их результаты записались в лог. Но при этом, если хотя бы одна проверка не прошла, то тест все-таки был провален.

Для этого есть специальный метод — `assertAll()`. В качестве первого аргумента он принимает комментарий, который нужно записать в лог, а дальше — любое количество функций-ассертов.

Вот как будет переписан наш пример с его помощью:

```
Address address = unitUnderTest.methodUnderTest();
assertAll("Сложный сценарий сравнение адреса",
    () -> assertEquals("Вашингтон", address.getCity()),
    () -> assertEquals("Oracle Parkway", address.getStreet()),
    () -> assertEquals("500", address.getNumber())
);
```

Тогда если адрес будет неправильный, в лог будет написано что-то типа:

```
Сложный сценарий сравнение адреса (3 failures)
expected: <Вашингтон> but was: <Сиэтл>
expected: <Oracle Parkway> but was: <Main Street>
expected: <500> but was: <5772>
```

6.4 Метод assertTimeout

Помнишь аннотацию **@Timeout**? Она позволяла контролировать время выполнения всего метода. Но иногда бывает полезно задать ограничения на выполнения некоторой части кода внутри метода. Для этого можно использовать `assertTimout()`.

В качестве первого параметра в него передается время, а в качестве второго — код (функция), который должен выполниться за указанное время. Пример:

```
@Test
public void whenAssertingTimeout() {
    assertTimeout(
    ofSeconds(2),
    () -> {
        // пауза в одну секунду
        Thread.sleep(1000);
    }
    );
}
```

У класса `Assert` есть 12 вариантов метода `assertTimeout()`. Если хочешь ознакомиться с ними подробнее, добро пожаловать в официальную документацию.

6.5 Метод assertThrows

Очень часто бывают ситуации, когда тебе нужно убедиться, что в определенной ситуации код кидает нужное исключение: определил ошибку и кинул нужное исключение. Это очень распространенная ситуация.

На этот случай есть еще один полезный метод `assert` — это `assertThrows()`. Общий формат его вызова имеет вид:

```
assertThrows(исключение, код)
```

По сути, он очень похож на метод `assertTimeout()`, только он проверяет, чтобы указанный код выкинул нужное исключение. Пример:

```
@Test
void whenAssertingException() {
    Throwable exception = assertThrows(
    IllegalArgumentException.class,
    () -> {
        throw new IllegalArgumentException("Exception message");
    }
    );
    assertEquals("Exception message", exception.getMessage());
}
```

Предыдущая лекция

Следующая лекция

Введите текст комментария

Олег Уровень 65, Уфа, Россия 13 сентября, 06:55

В коде п. 6.2 опечатка (метод некорректно задан)

Ответить +4

Mentor-02 Backend Developer в **JavaRush** MENTOR 1 ноября, 10:19

Спасибо, поправили

Ответить 0

ОБУЧЕНИЕ

Курсы программирования

Курс Java

Помощь по задачам

Подписки

Задачи-игры

СООБЩЕСТВО

Пользователи

Статьи

Форум

Чат

Истории успеха

Активности

КОМПАНИЯ

О нас

Контакты

Отзывы

FAQ

Поддержка

RUSH

JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

Русский

СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ