Выявление определенного поведения у объектов

JSP & Servlets
4 уровень, 4 лекция

ОТКРЫТА

5.1 Метод verify()

Кроме создания виртуальных объектов, часто возникает еще одна интересная задача – убедиться, что тестируемый класс вызвал нужные методы нужных объектов. Более того, вызвал нужное число раз, с правильными параметрами и тому подобное.

Для этого в Mockito тоже есть немного магии – семейство методов `Mockito.verify(…)`. Общее правило, которым задается проверка вызова метода, имеет вид:

```
Mockito.verify(объект).имяМетода(параметр);
```

Пример:

```
@ExtendWith(MockitoExtension.class)
class VerifyTest {
    @Mock
    List<String> mockList;

    @Test
    public void whenMockAnnotation() {
        //вызов метода
        String name = mockList.get(10);

        //проверяем вызывался ли метод
        Mockito.verify(mockList).get(10);
    }
}
```

Во время вызова метода `verify()` мы задали правило, что у объекта `mockitoList` должен вызваться метод `get()` с параметром 10.

5.2 Метод verify() с проверкой количества вызовов

Иногда бывают более сложные сценарии проверки. Например, тебе нужно проверить не просто факт, что метод вызывался, а например, что он вызывался 3 раза. Или он у тебя вызывался в цикле и значит должен был вызваться N раз.

Мы не будем спрашивать, можно ли это сделать, мы сразу спросим: как записать такое правило? И опять Mockito нас не подводит. Правило можно задать в виде:

```
Mockito.verify(объект,количество).имяМетода(параметр);
```

Важно! Количество – это не тип `int`, а специальный объект, которые может задавать различные шаблоны. Помнишь разные варианты метода `any()` ? Тут тоже самое – есть специальные методы, с помощью которых можно задавать различные сценарии:

	Метод	**Описание**
1	never()	Метод никогда не должен вызываться
2	times(n)	n раз
3	atLeast(n)	n или больше раз
4	atLeastOnce()	1 или больше раз
5	atMost(n)	n или меньше раз
6	only()	Должен быть только один вызов и только к этому методу

Пример:

```
String name1 = mockList.get(1);  //вызов метода
String name2 = mockList.get(2);  //вызов метода
String name3 = mockList.get(3);  //вызов метода

//проверяем, что метод get() вызывался 3 раза
Mockito.verify(mockList, times(3)).get(anyInt());
```

Ты также можешь потребовать, чтобы кроме указанных вызовов метода, **никаких других обращений к объекту не было**. Для этого есть правило:

```
Mockito.verifyNoMoreInteractions(объект);
```

5.3 Порядок вызова методов

Предыдущие правила никак не регламентировали порядок вызова методов. Правило просто должно выполниться и все. Но бывают ситуации, когда порядок вызовов методов важен и для этого у Mockito тоже есть решение.

Жесткий порядок вызова методов можно задать с помощью специального объекта `InOrder`. Сначала его нужно создать:

```
InOrder inOrder = Mockito.inOrder(объект);
```

А затем уже ему добавлять правила посредством вызова методов `verify()`.

Пример:

```
List<String> mockedList = mock(MyList.class);
mockedList.size();
mockedList.add("a parameter");
mockedList.clear();

InOrder inOrder = Mockito.inOrder(mockedList);
inOrder.verify(mockedList).size();

```

```
inOrder.verify(mockedList).add("a parameter");
inOrder.verify(mockedList).clear();
```

5.4 Проверка исключений в Mockito

Факт того, что исключения возникли, проверяется немного по другому. Для этого нужно использовать метод `assertThrows()`. Общий формат такой проверки имеет вид:

```
Assertions.assertThrows(исключение.class, () -> объект.имяМетода());
```

Ничего сложного.

Пример:

```
@ExtendWith(MockitoExtension.class)
class ThenThrowTest {
    @Mock
    List mockList;

    @Test
    public void whenMockAnnotation() {
        //задаем поведение метода (нужно только для демонстрации)
        Mockito.when(mockList.size()).thenThrow(IllegalStateException.class);

        //проверяем бросится ли IllegalStateException при вызове метода size
        assertThrows(IllegalStateException.class, () -> mockList.size());
    }
}
```

+14

Комментарии (5) популярные новые старые

JavaCoder

Введите текст комментария

Stas S Уровень 76, Гродно, Беларусь 8 сентября, 11:52

```
Во время вызова метода verify() мы задали правило, что у объекта mockitoList долже
```

Мы НЕ задаем никаких правил. Простой вызов метода с параметром и проверка, что он был вызван с данным параметром.

Вызов метода в assert-ах тоже считается, в отличии от вызова метода в служебных выражениях самого Mockito. (when - do и т.д.)

Ответить 0

Konstantin Уровень 86 26 июня, 18:09

Есть подозрения, что метод

```
Mockito.verify(объект).имяМетода(параметр);
```

надо вызывать после выполнения проверяемых вызовов.

Код из примеров лекции (и не только) у меня корректно отрабатывает только при вышеописанном условии!
Например, первый тест падает, второй норм

```
@ExtendWith(MockitoExtension.class)
class TestClass {
    @Mock
    List mockList;

    @Test
    public void whenMockAnnotation1() {
        Mockito.doReturn("Иван").when(mockList).get(10);
        Mockito.verify(mockList).get(10); // проверка до вызова метода
        String name = (String) mockList.get(10);
    }

    @Test
    public void whenMockAnnotation2() {
        Mockito.doReturn("Иван").when(mockList).get(10);
        String name = (String) mockList.get(10);
        Mockito.verify(mockList).get(10); // проверка после вызова метода
    }
}
```

Ответить +12

Vladislav Shamshurin Уровень 1, Russian Federation 2 августа, 14:24

Да, всё верно.

Ответить 0

OB11TO Backend Developer в **Мечтах** 20 июня, 17:49

Исправьте

```
String name = mockList).get( 1 );  //вызов метода
String name = mockList).get( 2 );  //вызов метода
String name = mockList).get( 3 );  //вызов метода
```

Ответить +3

dimarik236 System Engineer в **Tinkoff.ru** 18 июня, 09:15

```
String name = mockList).get(10);
```

скобочка кажется лишняя

Ответить +5

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

Русский

СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ

ДОСТУПНО В Google Play

Загрузите в App Store