

Основные аннотации Mockito

JSP & Servlets
4 уровень, 1 лекция

ОТКРЫТА

2.1 Аннотация @Mock

Есть два способа работы с мок-объектами в Mockito. Первый – это **создать полностью виртуальный объект**, второй – это **обернуть существующий объект** в некую обертку. Начнем с первого.

Чтобы создать полностью виртуальный объект, нужно написать код:

```
ИмяКласса имяПеременной = Mockito.mock(ИмяКласса.class);
```

Давайте ради примера создадим мок класса ArrayList:

```
1  @ExtendWith(MockitoExtension.class)
2  class MockTest {
3      @Test
4      public void whenNotUseMockAnnotation_thenCorrect() {
5          List mockList = Mockito.mock(ArrayList.class);
6          //эти методы не будут ничего делать – это заглушки
7          mockList.add("one");
8          mockList.add("two");
9      }
10 }
```

В этом примере мы создаем фейковый объект типа ArrayList и сохраняем ссылку на него в переменную mockList. Методы этого объекта ничего не делают.

Кстати, этот код можно записать еще короче, так как для этого есть специальная аннотация `@Mock`.

```
1  @ExtendWith(MockitoExtension.class)
2  class MockTest {
3      @Mock
4      List mockList;
5
6      @Test
7      public void whenNotUseMockAnnotation_thenCorrect() {
8          //эти методы не будут ничего делать – это заглушки
9          mockList.add("one");
10         mockList.add("two");
11     }
12 }
```

Во втором случае `MockitoExtension` сам проанализирует код класса и создаст нужные заглушки. Вызывать метод `Mockito.mock()` не нужно. Одна аннотация и виртуальный объект готов. Красота.

2.2 Аннотация @Spy

Второй важный тип объектов в Mockito – это обертки над существующими объектами. Они позволяют с одной стороны пользоваться уже существующими классами, а с другой – перехватывать обращение ко всем методам и переменным таких объектов: подкорректировать их работу, где это нужно. Используются так же часто, как и Mock-объекты.

Чтобы создать обертку над объектом, нужно написать код:

ИмяКласса имяПеременной = Mockito.spy(объект);

Пример с оберткой вокруг класса ArrayList:

```
1  @ExtendWith(MockitoExtension.class)
2  class SpyTest {
3      @Test
4      public void whenMockAnnotation() {
5          List<String> mockList = Mockito.spy(new ArrayList<String>());
6          //эти методы будут работать!
7          mockList.add("one");
8          mockList.add("two");
9      }
10 }
```

В самом простом варианте обращение к объекту-обертке просто перенаправляет вызовы к оригинальному объекту, ссылку на который он хранит у себя внутри. Все будет работать, как и с оригинальным объектом.

Создать обертку так же можно с помощью аннотации – @Spy.

```
1  @ExtendWith(MockitoExtension.class)
2  class SpyTest {
3      @Spy
4      List mockList = new ArrayList<String>();
5
6      @Test
7      public void whenMockAnnotation() {
8          // эти методы будут работать!
9          mockList.add("one");
10         mockList.add("two");
11     }
12 }
```

Эти два примера кода эквиваленты.

[← Предыдущая лекция](#)

[Следующая лекция →](#)

–

+19

+

Комментарии (14)

популярные новые старые

Введите текст комментария

Караганов Михаил

Уровень 68, Москва, Россия

29 августа, 12:47

...

"В самом простом варианте обращение к объекту-обертке просто перенаправляет вызовы к оригинальному объекту, ссылку на который он хранит у себя внутри. Все будет работать, как и с оригинальным объектом."

А в более сложном варианте? И если вызовы перенаправляются к оригиналу, то зачем обертка, раз методы обрабатывают точно так же?

Ответить

-

+2

+

Алексей

Уровень 79, Москва

25 июля, 16:14

...

1

@Mock

2

List mockList;

Когда мы пишем так, какой тип List и с каким дженериком создает Mockito?

Ответить

-

+3

+

On1k

Уровень 44, Krasnogorsk, Russian Federation

6 августа, 09:32

...

Может эквивалентно второй половине?
new ArrayList<String>();

Ответить

-

+2

+

Алексей

Уровень 79, Москва

7 августа, 10:24

...

Вопрос про те примере где нет второй половины... В примерах задано объявление с аннотацией и потом сразу метод add

Ответить

-

+2

+

Justinian

Judge в Mega City One

MASTER

9 августа, 08:45

...

Относительно дженерика, у нас непараметризированная коллекция, поэтому там Object ы будут, все остальное через правила кастинга.

Относительно типа List, а зачем нам тип List?
Мокито создаст свой чайлд класс, который заэкстендит от List и имплементирует соответствующие методы интерфейса.
Мы ведь все-равно другие методы не сможем вызывать кроме тех, которые в List.

Если нам нужна конкретная коллекция, то будем мокать конкретную.

Ответить

-

+6

+

Иван Голубев

Уровень 79, Москва, Россия

9 июля, 21:21

...

для junit 5 @ExtendWith(MockitoExtension.class) вместо @RunWith(MockitoJUnitRunner.class)

Ответить

-

+4

+

Булат

Уровень 90, Ural, Россия

8 июля, 15:32

...

@Spy
List mockList = new ArrayList<String>();

Здесь же создается новый объект, где подмена?

Ответить

-

+3

+

Anton Nikolaev

System Engineer

20 июля, 17:42

...

Аннотация Spy над созданным объектом коллекции говорит о том, что Mockito контролирует его вызовы с помощью рефлексии и работает как прокси.

Ответить

-

+7

+

Максим

Уровень 90, Саранск, Россия

21 июня, 23:58

...

List mockList = new ArrayList<String>(); Пропустили в начале дженерик

Ответить

-

+4

+

Андрей Пазюк

Уровень 70, Винница, Украина

21 июля, 20:20

...

Максим, его писать не обязательно, потому что сам объект уже типизирован.

Ответить

-

+2

+

Максим

Уровень 90, Саранск, Россия

21 июля, 20:29

...

Тогда нужно убрать от сюда "<String>"

1

new ArrayList<String>()

Ответить

+2

Андрей Пазюк

Уровень 70, Винница, Украина

21 июля, 20:39

[Максим](#), не нужно. Ведь именно оно и задает тип дженерика. Для удобства в Java разрешено упускать указание типа дженерику, если он уже указан где-то один раз. Например эти три примера эквивалентны:

```
1 List mockList = new ArrayList<String>();
2 List<String> mockList = new ArrayList<>();
3 List<String> mockList = new ArrayList<String>();
```

Ответить

+4

Максим

Уровень 90, Саранск, Россия

21 июля, 21:03

Да, ты прав, я всегда думал что есть только 2 вида записи

```
1 List<String> mockList = new ArrayList<>();
2 List<String> mockList = new ArrayList<String>();
```

Ответить

+2

Александр Огарков

Уровень 71, Мытищи, Россия

23 июля, 14:46

В первом случае получишь сырой тип.

Ответить

+3

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



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

