Карта квестов

**CS50** 

**Android** 

# Продвинутое тестирование с помощью Mockito

JSP & Servlets 4 уровень, 0 лекция

ОТКРЫТА

Лекции

### 1.1 Библиотека Mockito

Сегодня мы познакомимся с продвинутым тестированием. А точнее с библиотекой **Mockito**. Даже и не думайте увильнуть от этого дела.

Поиск

Во-первых, эта библиотека – это стандарт в тестирование **Spring'a**. Который фактически является стандартом в отрасли Javabackend разработки.

Во-вторых, тебе придется писать тесты для своего Spring-кода. Единственный способ понять, что написанный тобой бекенд работает как нужно – это вызывать методы его API. И сделать это с помощью тестов в 10 раз легче, чем без них. Ты сам в этом убедишься.

Добавить библиотеку Mockito в ваш ром.xml можно с помощью кода:

Весь исходный код проекта Mockito можно найти на GitHub.

### 1.2 Моск-объекты

Так что же такое эта Mockito и чем она так хороша?

В процессе развития и становления тестирования очень часто возникала необходимость вместо реального объекта подсунуть коду какую-нибудь «заглушку».

Например, тестируется код, который работает с базой данный и что-то там меняет. Хорошо чтобы перед каждым тестом состояние этой базы было одним и тем же (иначе тесты будут разные). И базу хотелось бы попроще, чтобы быстро эти состояния откатывать.

Или, например, ты тестируешь код, который рассылает полезные SMS. А для непосредственно рассылок он использует какойнибудь платный SMS Gateway. Хорошо бы для тестирования кода подсунуть ему некий виртуальный Gateway, чтобы не рассылать сотни SMS непонятно кому.

Или твой код запрашивает данные у других веб-серверов, которые банально недоступны на тестовом сервере. Или ты пишешь код для интернет-платежей, который нужно 50 раз протестировать, и только потом допускать к реальным финансовым каналам.

Думаю, вы поняли... Виртуальные объекты или как их еще называют объекты-заглушки очень полезная штука.

И тут возникает сложность — в Java-то статическая типизация. Значит, чтобы вместо объекта типа ReadDatabase присвоить переменной ссылку на объект VirtualDatabase(), нужно унаследовать класс VirtualDatabase от RealDatabase.

Затем оказывается, что у класса RealDatabase куча приватных методов и переменных, которые хранят ссылки на другие реальные объекты, и нормальную заглушку таким образом не напишешь. В теории хороший, но на практике тупиковый

вариант.

И тут нам на помощь приходит DynamicProxy (более подробно можно почитать), которые появились еще в Java 5. Она позволяет создавать виртуальные объекты, к которым у компилятора нет претензий.

Такие виртуальные объекты называют **mock'ами** (от слова mock – макет). Библиотека Mockito смогла вознести работу с этими моками на небывалую высоту. Отсюда, кстати, и название библиотеки.

# 1.3 Аннотация @ExtendWith

Библиотека Mockito отлично работает вместе с JUnit, ее можно даже рассматривать как его расширение.

Есть два способа активировать работу библиотеки Mockito в ваших unit-тестах. Первый способ – это добавить специальную аннотацию:

```
1  @ExtendWith(MockitoExtension.class)
2  public class MockitoAnnotationTest {
3    ...
4 }
```

Второй способ – включить ее работу вызвав метод openMocks():

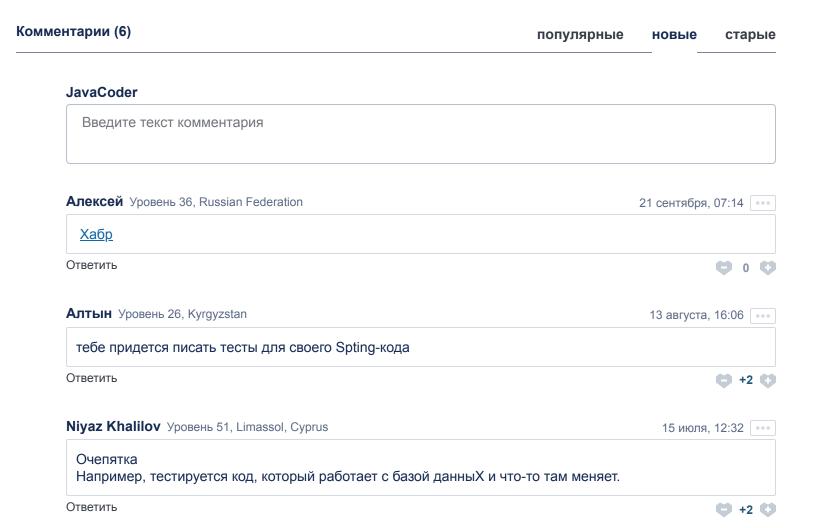
```
public class MockitoAnnotationTest {
    @BeforeEach
    public void init() {
        MockitoAnnotations.openMocks(this);
    }
}
```

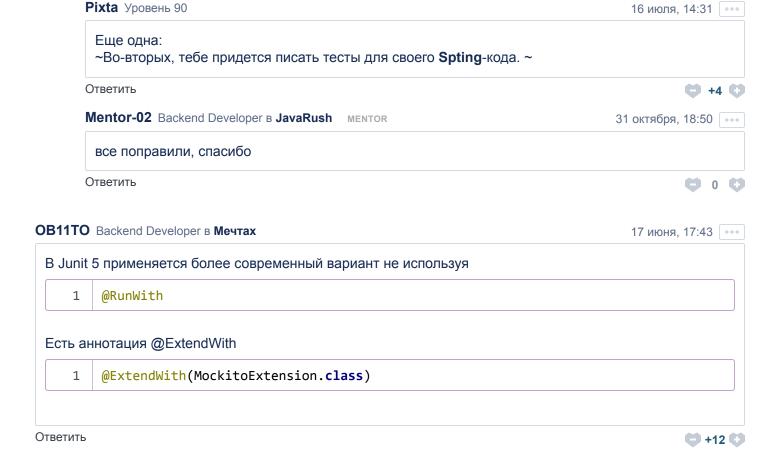
Чаще всего ты будешь видеть первый вариант, но иногда полезно знать, что есть и второй.

< Предыдущая лекция

Следующая лекция >







ОБУЧЕНИЕ СООБЩЕСТВО КОМПАНИЯ Пользователи О нас Курсы программирования Kypc Java Статьи Контакты Помощь по задачам Форум Отзывы Подписки Чат **FAQ** Задачи-игры Истории успеха Поддержка Активности



## 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



## СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ





