# Настройка окружения теста в JUnit

JSP & Servlets 3 уровень, 2 лекция

ОТКРЫТА

## 3.1 Аннотации @BeforeEach, @AfterEach

Обрати внимание, что в предыдущем примере нам в каждом методе приходилось писать код для создания объекта Calculator.

Конечно, это всего одна строчка, но если мы будем тестировать реальные системы, то часто будет возникать ситуация, когда нужно создать и сконфигурировать несколько объектов, что может занять несколько десятков строк кода. Пример:

```
1
     //Создаем объект HttpClient
2
        HttpClient client = HttpClient.newBuilder()
3
              .version(Version.HTTP_1_1)
              .followRedirects(Redirect.NORMAL)
4
              .connectTimeout(Duration.ofSeconds(20))
5
              .proxy(ProxySelector.of(new InetSocketAddress("proxy.example.com", 80)))
6
7
              .authenticator(Authenticator.getDefault())
8
              .build();
9
        //Создаем объект HttpRequest
10
11
       HttpRequest request = HttpRequest.newBuilder()
           .uri(new URI("https://javarush.ru"))
12
           .headers("Content-Type", " application/octet-stream")
13
           .POST( HttpRequest.BodyPublishers. ofInputStream ( () -> is; ))
14
           .build();
15
16
17
        //Вызываем метод send()
        HttpResponse response = client.send(request, BodyHandlers.ofString());
18
19
        System.out.println(response.statusCode());
20
        System.out.println(response.body());
```

В примере выше мы создали и сконфигурировали объект HttpClient и хотим протестировать работу метода send().

Чтобы каждый раз в тестовом методе не писать создание объекта HttpClient, его можно вынести в отдельный метод и поставить ему специальную аннотацию @BeforeEach. Тогда Junit будет вызывать этот метод перед каждым тестовым методом. Пример:

```
1
     class HttpClientTest {
2
             public HttpClient client;
3
         @BeforeEach
4
             public void init(){
5
             client = HttpClient.newBuilder()
6
                  .version(Version.HTTP 1 1)
7
                  .followRedirects(Redirect.NORMAL)
8
9
                  .connectTimeout(Duration.ofSeconds(20))
                  .proxy(ProxySelector.of(new InetSocketAddress("proxy.example.com", 80)))
10
                  .authenticator(Authenticator.getDefault())
11
```

```
.build();
12
              }
13
14
         @Test
15
              public void send200() throws Exception {
16
17
                 //Создаем объект HttpRequst()
                  HttpRequest request = HttpRequest.newBuilder(new URI("https://javarush.ru")).build();
18
19
                 //Вызываем метод send()
20
21
                 HttpResponse response = client.send(request, BodyHandlers.ofString());
                  assertEquals(200, response.statusCode());
22
23
              }
24
         @Test
25
              public void send404() throws Exception {
26
                 //Создаем объект HttpRequst()
27
                  HttpRequest request = HttpRequest.newBuilder(new URI("https://javarush.ru/unknown")).buil
28
29
30
                 //Вызываем метод send()
                 HttpResponse response = client.send(request, BodyHandlers.ofString());
31
32
                  assertEquals(404, response.statusCode());
33
              }
34
     }
```

Также можно создать специальный метод, который будет вызываться каждый раз после очередного тестового метода, и подчищать использованные ресурсы, писать что-то в лог и т. п. Такой метод нужно пометить аннотацией **@AfterEach**.

Если у тебя есть 3 тестовых метода | test1() |, | test2() | и | test3() |, то порядок вызова будет таким:

- BeforeEach-метод
- test1()
- AfterEach-метод
- BeforeEach-метод
- test2()
- AfterEach-метод
- BeforeEach-метод
- test3()
- AfterEach-метод

## 3.2 Аннотации @BeforeAll, @AfterAll

JUnit также позволяет добавить метод, который будет вызван один раз перед всеми тестовыми методами. Такой метод нужно пометить аннотацией @BeforeAll. Для нее так же существует парная аннотация @AfterAll. Метод, помеченный ею, JUnit вызовет после всех тестовых методов.

Давай напишем специальный пример, который позволяет лучше понять, как это все работает. За основу возьмем тестирование нашего калькулятора:

```
class CalculatorTest {
1
        private Calculator calc = new Calculator();
2
3
4
        @BeforeAll
5
         public static void init(){
             System.out.println("BeforeAll init() method called");
6
7
         }
8
9
         @BeforeEach
```

```
public void initEach(){
10
              System.out.println("BeforeEach initEach() method called");
11
12
          }
13
         @Test
14
          public void add(){
15
              System.out.println("Testing Addition");
16
17
          }
18
         @Test
19
          public void sub() {
20
              System.out.println("Testing Subtraction");
21
          }
22
23
24
         @Test
          public void mul(){
25
              System.out.println("Testing Multiplication");
26
27
          }
28
29
         @Test
          public void div() {
30
              System.out.println("Testing Division");
31
32
          }
33
     }
```

Этот тест напечатает в консоль такой текст:

```
BeforeAll init() method called
BeforeEach initEach() method called
Testing Addition

BeforeEach initEach() method called
Testing Subtraction

BeforeEach initEach() method called
Testing Multiplication

BeforeEach initEach() method called
Testing Division
```

< Предыдущая лекция

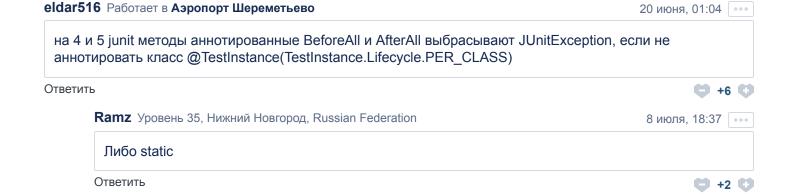
Следующая лекция >

+23

Комментарии (2) популярные новые старые

JavaCoder

Введите текст комментария



Курсы программирования Пользователи Онас Контакты Kypc Java Статьи Форум Отзывы Помощь по задачам Чат **FAQ** Подписки Задачи-игры Истории успеха Поддержка Активности

СООБЩЕСТВО

КОМПАНИЯ



ОБУЧЕНИЕ

### RUSH

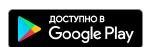
JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном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

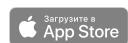
### ПОДПИСЫВАЙТЕСЬ

#### ЯЗЫК ИНТЕРФЕЙСА



### СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ







"Программистами не рождаются" © 2022 JavaRush