链滴

用 Spock 单元测试框架替代 JUnit

作者：ZephyrJung
原文链接：https://ld246.com/article/1542954344002
来源网站：链滴

Spock是一个Java及Groovy应用的测试框架。它之所以能从众多框架中脱颖而出，是由于它富有表力的语言。通过JUnit runner，Spock能够与大多数IDE、构建工具及集成测试服务兼容。Spock的诞受到了JUnit, jMock, RSpec, Groovy, Scala, Vulcans的启发。

Groovy VS Java

此处不做细节上的比较，只提在写单元测试中用到的代码

1. 类型推断

```
orderData.setOrderId(123456L)
orderMoney.setInitFactPrice(new BigDecimal("3.2"));

orderData.orderId = 123456
orderMoney.initFactprice = 3.2
```

2. 输出结果

```
System.out.println("Hello world");

println("Hello world")
```

3. 创建List

```
List<String> list = new ArrayList<>();
list.add("test1");
list.add("test2");
// 或使用guava
List<String> list = Lists.newArrayList("test1","test2");

OrderData odtest1 = new OrderData();
odtest1.setOrderId(123456L);
odtest1.setStoreId(123L);
OrderData odtest2 = new OrderData();
...
List<OrderData> orderDataList = Lists.newArrayList();
orderDataList.add(odtest1);
orderDataList.add(odtest2);


def list = ["test1","test2"]
def orderDataList = [new OrderData(orderId:123456,storeId:123),
                     new OrderData(orderId:123456,storeId:234)
                    ]
// 或
orderDataList << new OrderData(orderId:123456,storeId:123)
orderDataList << new OrderData(orderId:123456,storeId:124)
```

4. 对象比较

```
odtest1.getOrderId() == odtest2.getOrderId();
odtest1.getInitFactPrice().compareTo(odtest2.getInitFactPrice()) == 0;
//不适用java 8 提供的Stream方法下，判断数组的步骤要更多，如size比较，对应对象或值比较，此意会即好
for(int i=0;i<list1.size;i++){
```

```
    list1.get(i).equals(list2.get(i))
}

odtest1 == odtest2
test1 == test2
```

更简洁的书写语法，能够让你在对测试有限的热情里写出更多的测试用例

Get Started With Spock

```
import spock.lang.*
class MyFirstSpecification extends Specification {
    // fields
    def coll = new Collaborator()
    @Shared res = new VeryExpensiveResource()
    // fixture methods
    def setup() {}          // run before every feature method
    def cleanup() {}        // run after every feature method
    def setupSpec() {}      // run before the first feature method
    def cleanupSpec() {}    // run after the last feature method
    // feature methods
    def "pushing an element on the stack"() {
      // blocks go here
    }
    // helper methods
    def testHelper(){

    }
}
```

blocks

setup -> 测试准备

clean -> 测试收尾

where -> 循环遍历

when/then : 给定……则满足……

expect：应满足……

Spock VS JUnit

简单的单元测试，测试代码的差异主要在Java和Groovy的语法差异上，而参数化的单元测试，能够显的感受到两个测试框架的不同，故此处以参数化测试为例：

```
// 24 lines
@RunWith(Parameterized.class)
public class MyTest {
    private String name;
    private String family;
```

```
    public MyTest(String name,String family) {
        super();
        this.name = name;
        this.family = family;
    }

    @Parameterized.Parameters
    public static Collection input() {
        return Arrays.asList(new Object[][]{
                {"Zephyr","Jung"}, {"Y","Z"}
        });
    }

    @Test
    public void test(){
        System.out.println(name+" "+family);
    }
}

// 10 lines
@Unroll("#name #family")
class MyTest2 extends Specification {
    def "test"() {
        expect:
        println(name + " " + family)
        where:
        name     | family
        "zephyr" | "Jung"
        "Y"      | "Z"
    }
}
```

这个单元测试中的参数是简单的字符串，我们可以看到，使用jUnit来实现参数化的单元测试，写法要Spock更为复杂，当参数是对象时，又会附加上Java代码的复杂性，大量的时间用在了模板性质的代上。

通过@Unroll注解，能够将每一个参数作为标题展现在测试结果中

在SpringBoot项目中添加Spock测试

```
<dependency>
    <groupId>org.spockframework</groupId>
    <artifactId>spock-spring</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>

@SpringBootTest
```

```
@ContextConfiguration(classes = Application.class)
class MyTest extends Specification {
    @Autowired
    private TicketCenter ticketCenter
    @Autowired
    private Train train

    def "testSpring"() {
        when:
        Ticket ticket = ticketCenter.getTicket(new Path(start, end, date))
        train.setTicket(ticket)
        train.travel()
        then:
        true
        where:
        start       | end         | date
        "zhengzhou" | "shanghai"  | new Date()
        "shanghai"  | "zhengzhou" | new Date()

    }
}
```

原文链接：用 Spock 单元测试框架替代 JUnit