Mockito实现单测

创建： 王丽月，最新修改于： 六月 21, 2017

为了重构、迭代开发，不影响到已有功能，我们需要快速回归已有的功能点。因此，单测显得尤为重要。然而，实际上，大多数为了覆盖率而编写的单测起不到应有的效果。

我认为，单测的目标：

1. 底层CRUD，最真实的DB操作，及时发现含DB类型错误等问题，可以采用事务回滚，使测试用例间不相互干扰，不产生脏数据；
2. 中间业务层，可以mock远程调用，只遍历内部业务逻辑分支；但业务内部，也可以分布真是调用，确保业务全流程正常；
3. 上层api， 根据业务场景，准备数据，真实地跑过各个服务，而不是mock服务；

简单介绍mockito的用法，以及之前踩的几个坑。

使用mockito，可以将远程调用mock

1. 建一个基类，准备单测的上下文环境及初始化操作

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations ="classpath:application-context.xml")
public abstract class BaseTest extends AbstractTransactionalJUnit4SpringContextTests {
    protected static Logger logger = LogUtils.get();
    @BeforeClass
    public static void init() {
        AppEnvProfileLoader.initialize();
    }
    @Before
    public void mock() {
        MockitoAnnotations.initMocks(this);
    }
}
```

AbstractTransactionalJUnit4SpringContextTests：以事务方式执行单测

AbstractJUnit4SpringContextTests：非事务

2. 具体类中，导入并**mock**远程服务

```
@Import({ OrderAptServiceClient.class, OrderBaseServiceClient.class, OrderSettlementCl
        ProductCacheServiceClient.class, ProductInventoryClient.class, RiskControlClie
        UgcBaseServiceClient.class, UserBaseServiceClient.class, VoucherBaseClient.cla
public class OrderSubmitFacadeTest extends BaseTest {
    @InjectMocks
    @Autowired
    private OrderSubmitFacade orderSubmitFacade;
    @Mock
    private OrderBaseServiceClient orderBaseServiceClient;
    @Mock
    private PhxPayClient phxPayClient;
    @Mock
    private ProductCacheServiceClient productCacheServiceClient;
    @Mock
    private UserBaseServiceClient userBaseServiceClient;
```

```
    @Mock
    private VoucherBaseClient voucherBaseClient;

    @Test
    public void testXXX() {
        // do some test
        OrderCreateParam orderCreateParam = MockData.getExistOrderCreateParam();
        when(orderBaseServiceClient.getOrderStatus(orderCreateParam.getOrderId()))
            .thenReturn(OrderStatusEnum.APPLY_SUC.getCode());
        when(orderBaseServiceClient.getPaymentById(orderCreateParam.getOrderId()))
            .then(OrderBaseAnswer.getMtPaymentModel);
        doNothing().when(orderBaseServiceClient).createPayment(any(PaymentDetailModel.
        doNothing().when(orderBaseServiceClient).transferOrderStatus(any(OrderTransfer
        OrderPaymentModel orderPaymentModel = orderSubmitFacade.submitOrder(orderCreat
        assert orderPaymentModel != null;
    }
}
```

注意，orderSubmitFacade是个既需要真实类注入，又需要mock对象的实例，因此需要@InjectMocks和@Autowired双重注解。而orderBaseServiceClient等，则是我们mock的对象，它们需要配合@Import，先将Class导入，下面才可以作为mock注入的实例。

3. **mockito基本语法**

由具体的场景来决定when...then...或者doNoting.when...。

doNothing().when(xxxClass).xxxMethod(xxxParam)

doThrow(xxxException).when(xxxClass).xxxMethod(xxxParam)

when(xxxClass.xxxMethod(xxxParam)).then(xxxAnswer)

when(xxxClass.xxxMethod(xxxParam)).thenReturn(xxxReturn)

赞 成为第一个赞同者 无标签