单元测试

2021年10月3日 10:49

单元测试的对象

Model对象：

Controller、Service、Dao对象

值对象：

VO、BO、PO对象

切片测试

在这个例子中我们分离Controller层和Service层

Controller层使用Resultful API来进行数据的输入和合法性检查

这里我们使用Mockito的MockBean，在SpringBoot的环境下把Service层做成一个模拟的Bean对象

集成测试

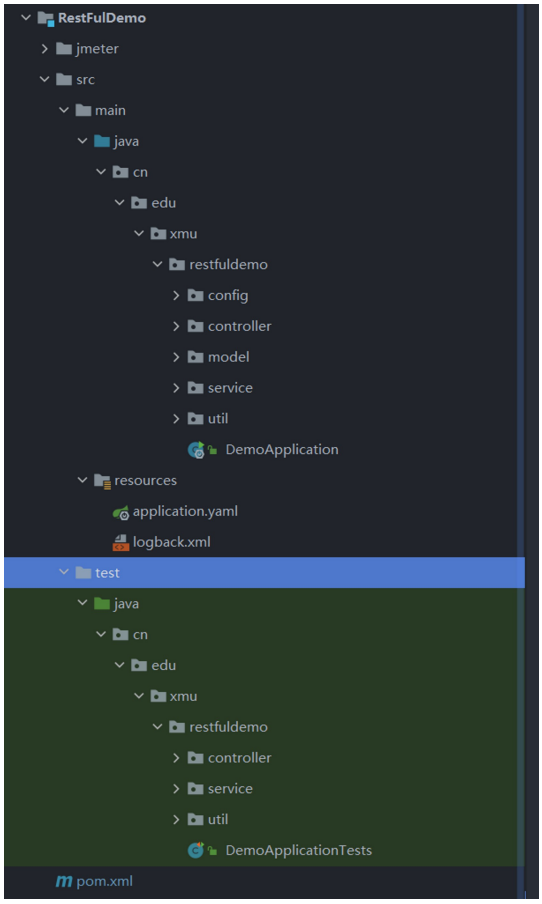
在这个例子中，将Controller层和Service层整合测试

性能测试

在这个例子中，我们选择特定的API来进行性能测试

使用JMeter

以Resultfuldemo为例



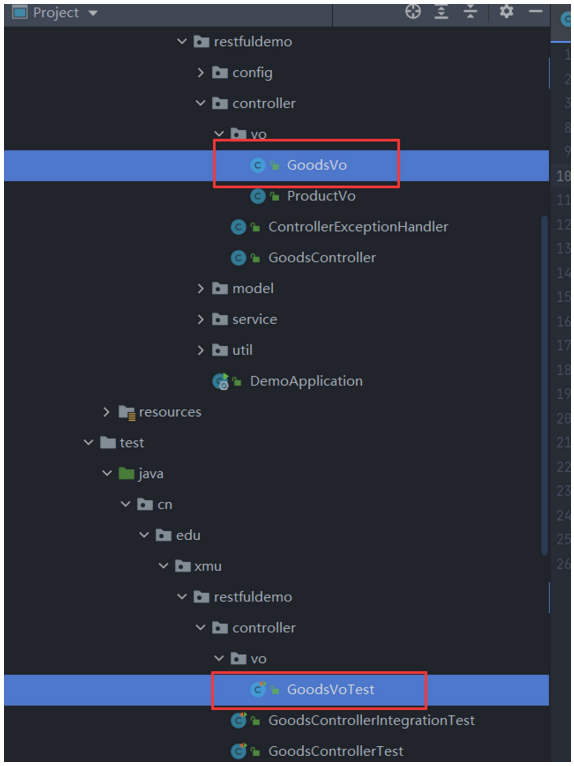
测试代码都在test目录下

结构与生产代码main目录要保持一致

要测某个类，它在main目录下是什么路径，在test目录下就要是什么路径

如GoodsVo在main/java/cn/edu/xmu/restfuldemo/controller/vo下

那么对应的测试类应该在test/java/cn/edu/xmu/restfuldemo/controller/vo下



测试类的命名必须对应且以Test结尾
比如我们命名为GoodsVoTest，说明我们是去测试GoodsVo的

```
    */
    public Goods createGoods(){
        Goods goods = new Goods();
        goods.setName(this.name);
        goods.setBrief(this.brief);
        goods.setUnit(this.unit);
        goods.setSpecList(this.specList);
        goods.setBrandId(this.brandId);
        goods.setCategoryId(this.categoryId);
        return goods;
    }
```

我们是要去测试GoodsVo中的createGoods方法的，因此测试类中也有一个方法叫做createGoodsTest，方法前面加上@Test标签
在测试之前，我们要先把VO对象创建出来，因此我们使用工厂设计模式来创建对象，详见GoodsFactory类
GoodsFactory是为了测试而写的类，因此我们把它放在test的util下，而没有放在生产代码下，因此打jar包时是没有GoodsFactory的

```
public class GoodsVoTest {

    @Test
    public void createGoodsTest(){
        GoodsVo goodsVo= GoodsFactory.getInstance().createGoodsVo();
        Goods goods = goodsVo.createGoods();
        assertEquals( expected: "红米4X", goods.getName() );
        assertEquals( expected: "红米4X是个好用便宜的手机", goods.getBrief() );
        assertEquals( expected: "台", goods.getUnit());
        assertEquals( expected: 11, goods.getCategoryId());
        assertEquals( expected: 12, goods.getBrandId());
        assertEquals( expected: 3, goods.getSpecList().size());
    }
}
```

补充：工厂设计模式
工厂设计模式，顾名思义，就是用来生产对象的，在java中，万物皆对象，这些对象都需要创建，如果创建的时候直接new该对象，就会对该对象耦合严重，假如我们要更换对象，所有new对象的地方都需要修改一遍，这显然违背了软件设计的开闭原则，如果我们使用工厂来生产对象，我们就只和工厂打交道就可以了，彻底和对象解耦，如果要更换对象，直接在工厂里更换该对象即可，达到了与对象解耦的目的；所以说，工厂模式最大的优点就是：解耦

这里使用assertEquals来看比较两个值，如果是则测试通过，不是则报错
（也可以使用assertTrue）

这里的单元测试是用Junit进行的，不依赖于Spring框架
但如果我们需要测试dao对象或者其他的，就需要依赖Spring框架