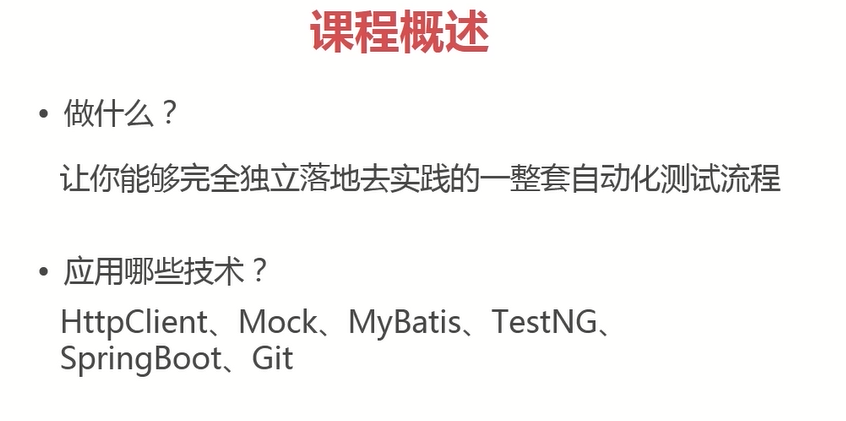
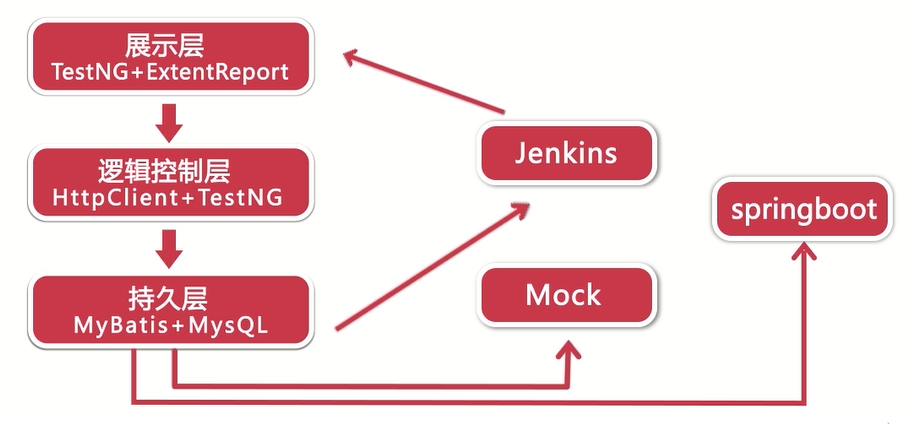
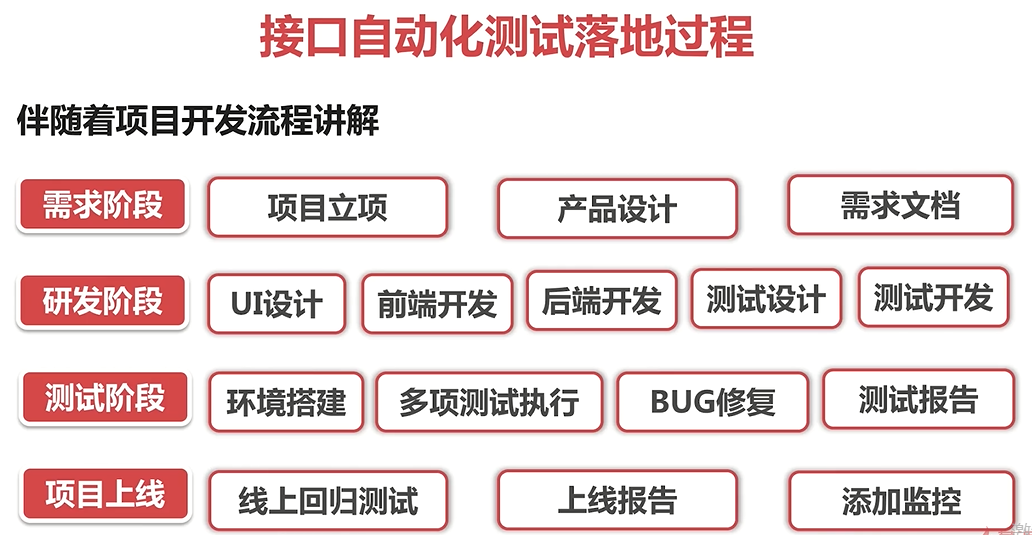
1-1导学





持久层管理测试用例

1-5接口测试落地过程



## 2-1接口测试范围





Null：对象为空

“”：空字符串

数据类型：注意负数，小数点

负载均衡架构：

1个Nginx+多个tomcat，其中一个tomcat挂掉，是否不影响

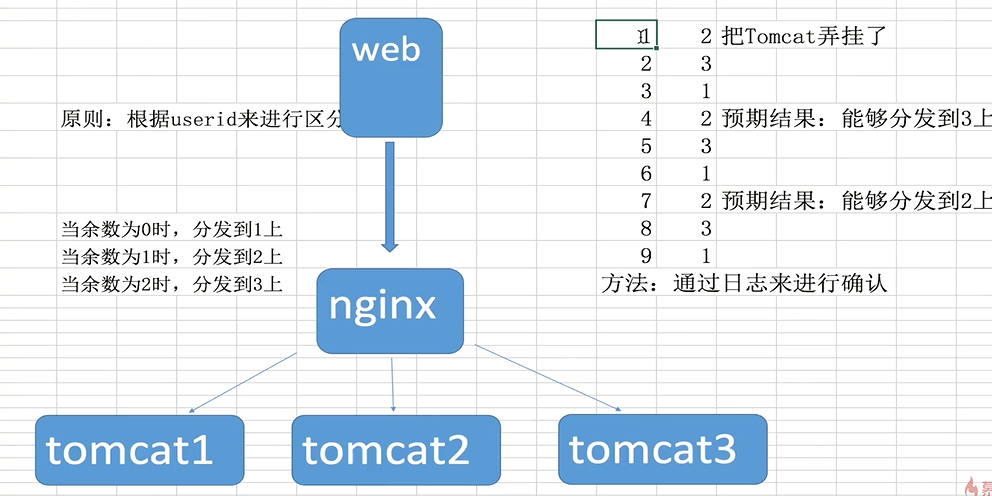
Nginx负责分发

冷热备份：

主tomcat挂掉，从tomcat是否能接替，回复后是否正常

## 2-2接口测试用例个设计

## 2-3环境异常该怎么测试



3手工接口测试到自动化框架设计

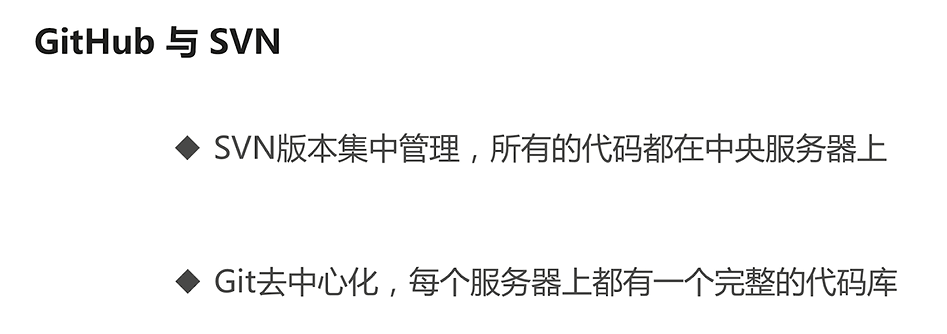
3-1手工接口测试的常用工具

3-2自动化框架设计简介



# 4 git应用

## 4-1为什么使用git



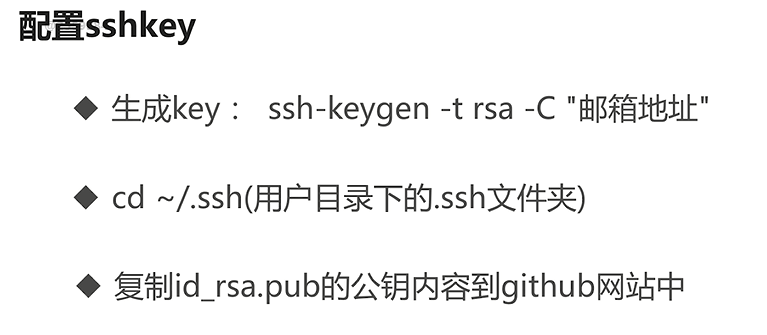
## 4-2git客户端安装

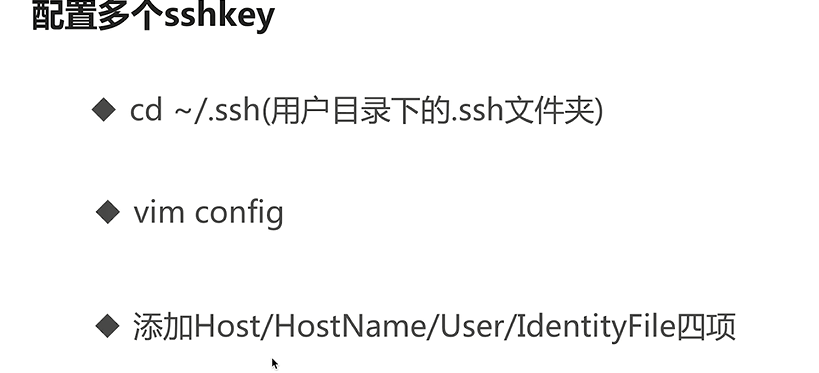


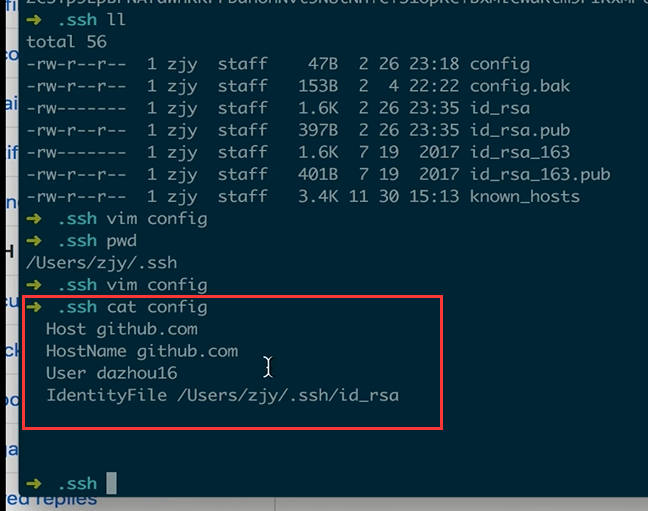


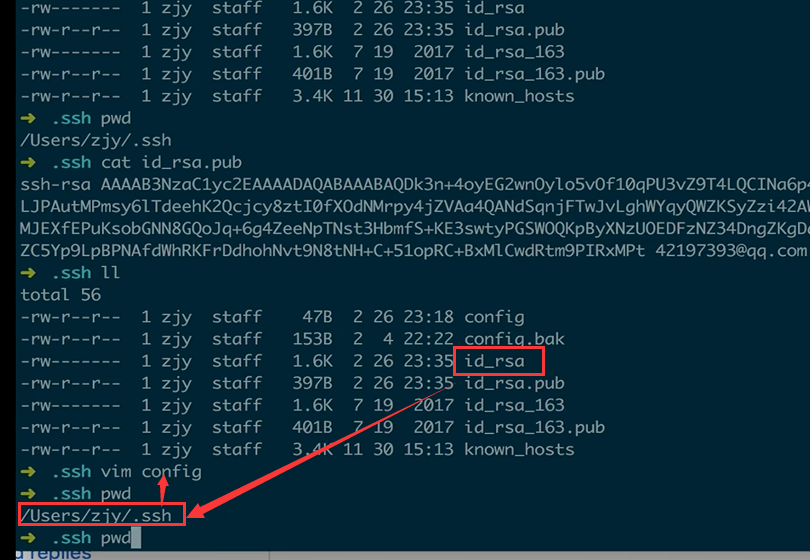
## 4-3拥有一个git账号

## 4-4github的基本使用ssh-key的客户端配置



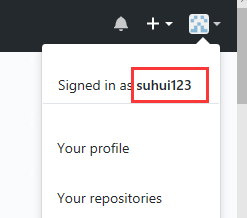




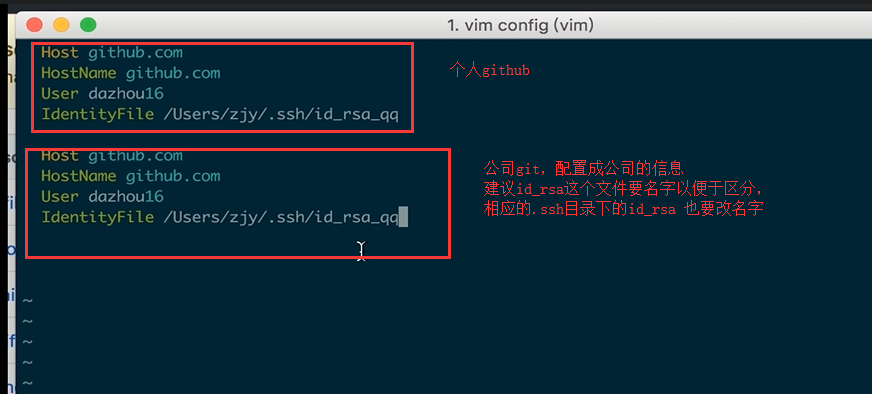


IdentityFile的配置是id\_rsa文件的路径如上图中的配置

Users的配置是guihub上的名字，如下图



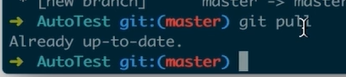
## 4-5多个github网站账号的配置



## 4-6创建git仓库以及克隆，拉去推送

拉取命令

Git pull



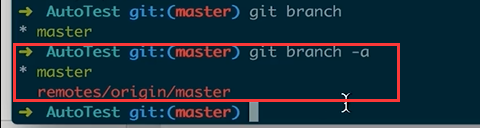
如果是already up-to-date 表示本地和远端一致

## 4-7创建分支

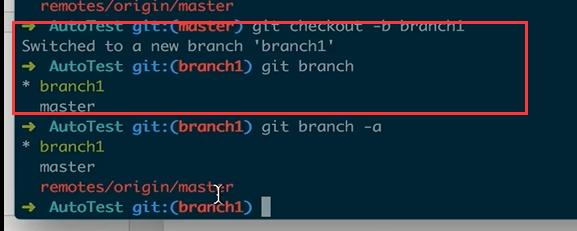
查看本地分支



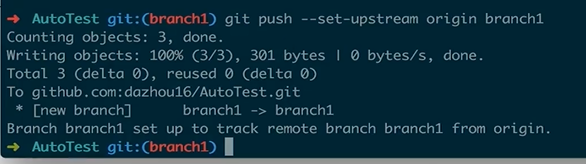
查看远端分支



创建本地分支



本地分支提交到远端



## 4-8分支删除

命令git branch -d 分支名

分支作用场景：

1修复bug时会给开发人员，从master拉出来一个分支，供其修改，bug修复后合并到master上，这时这个分支没用了，可以删除

2 太多没用的分支需要删除

### **删除分支branch1的流程**：

1查看本地分支git branch

2查看远程分支 git branch -a

3本地分支切换到master git checkout master

3删除本地分支 git branch -d branch1

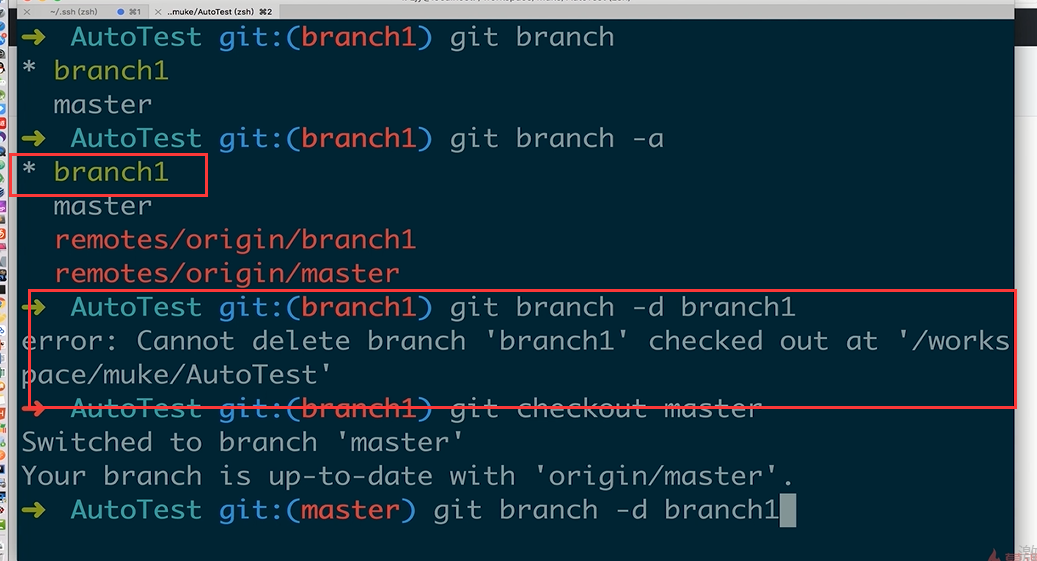
4 删除远程分支 git branch -r -d origin/brangch1

5推送 git push origin :branch1

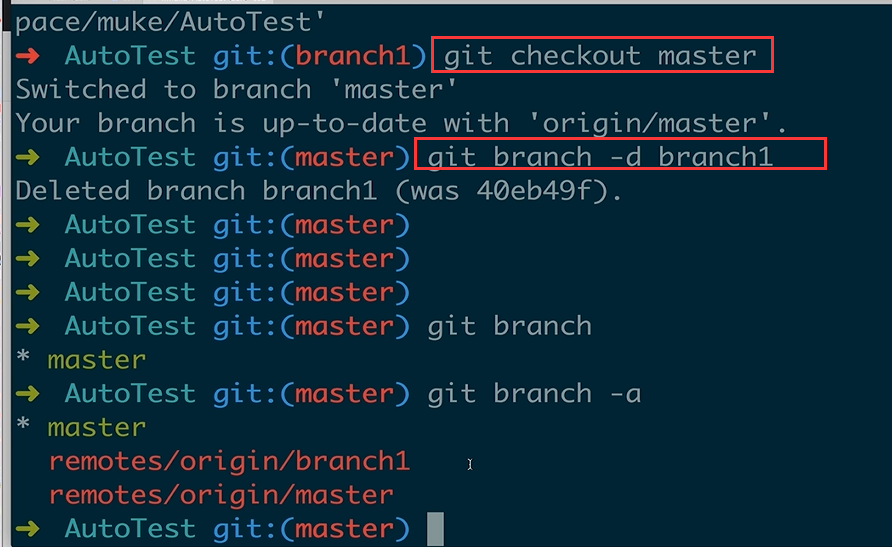
### 删除分支时的坑

删除分支的坑1：删除本地的分支时，没有切换目录

删除本地分支 git branch -d branch1

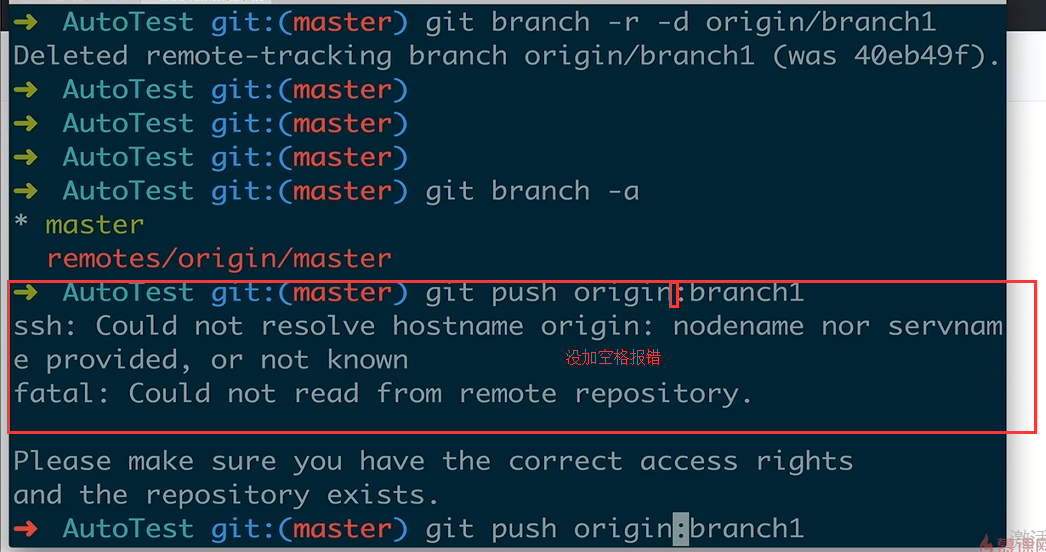


上图：不能删除当前工作的分支，需要先切换git checkout master在删除如下图：



删除分支坑2：删除远程分只后，执行git push origin :branch1时，在origin和:之间没有输入空格

删除远程分支：git branch -r -d origin/brangch1



## 4-9合并分支

现在有master 和分支mergedemo，把mergedemo合并到master上：

1切的换到master :git checkout master

2 合并分支：git merge mergedemo



## 4-10 解决合并分支的冲突问题

冲突原因：合并分支时，对于同一行代码，每个分支上的内容不一样



解决办法，手动修改

然后在master上 git 。。。Commit。。。Push

## 4-11项目版本操作

版本操作最常用的是版本回退

回退一般都是本地回退，然后提交到远程

### 回退到上一版本

回退命令：

Git reset --hard HEAD^



### 回到指定版本

查询所有版本号：git reflog 得到如下图：

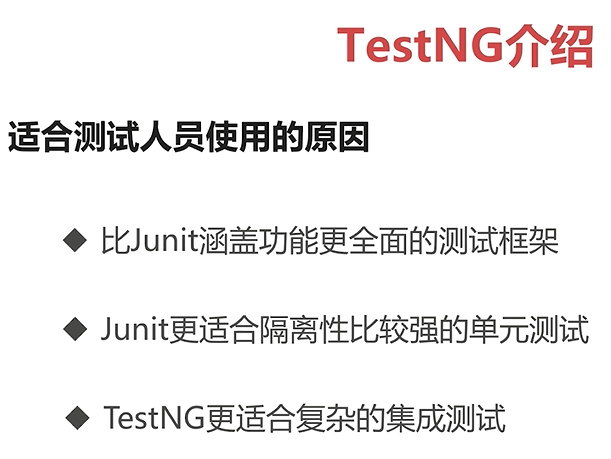


找到指定版本号后执行 git reset --hard 版本号



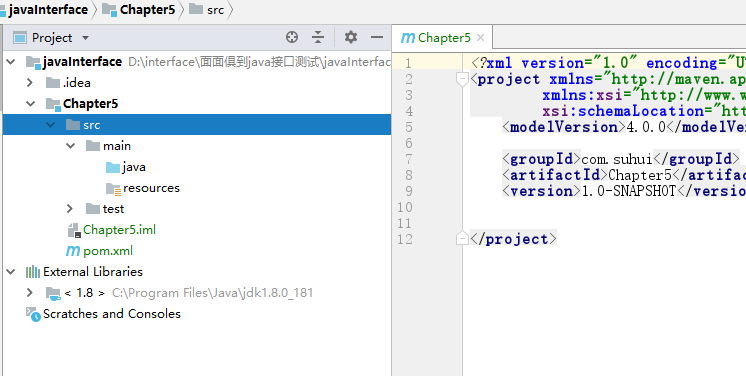
# 5 Testng

## 5-1 testng基本介绍和如何在maven中引用

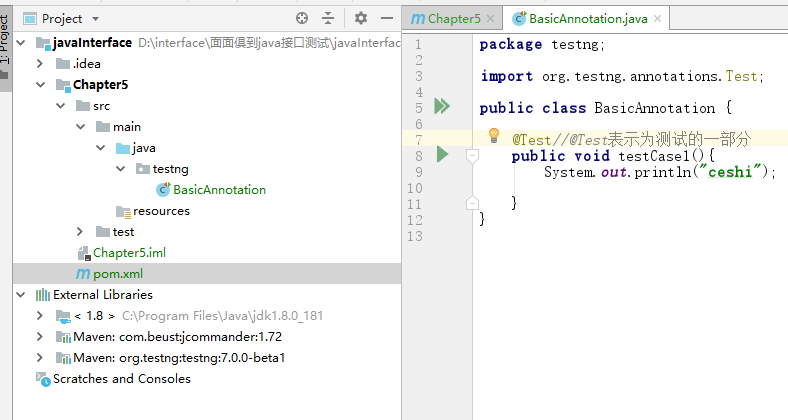


## 5-2testng实战在idea中创建module

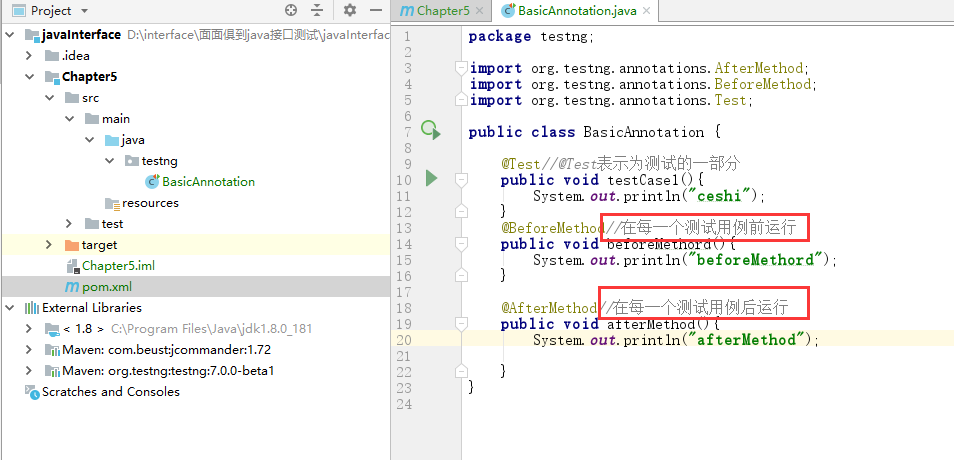
如图创建：



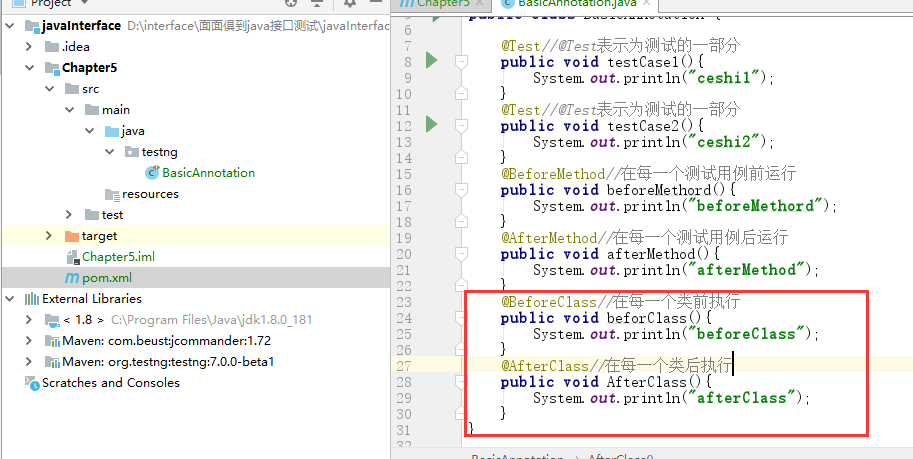
## 5-3注解实战@Test标签



## 5-4注解实战BeforeMethod和AfterMethord



## 5-5注解实战BeforeClass和AfterClass



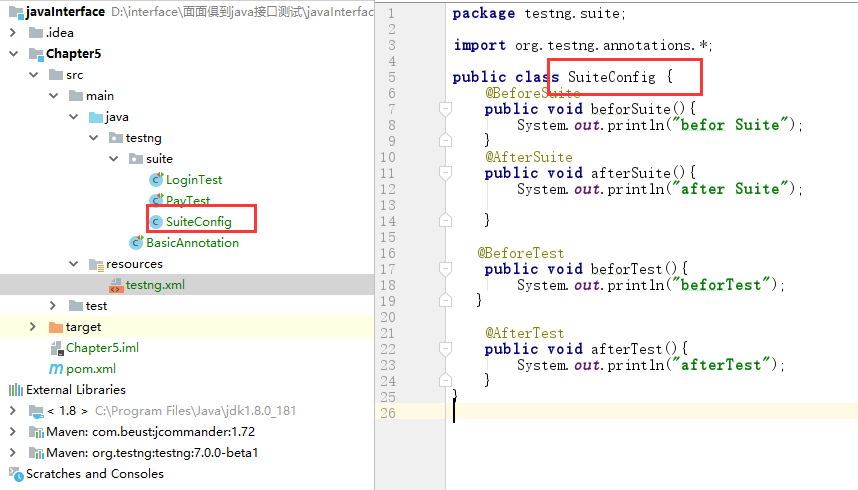
## 5-6注解实战BeforeSuite和AfterSuite

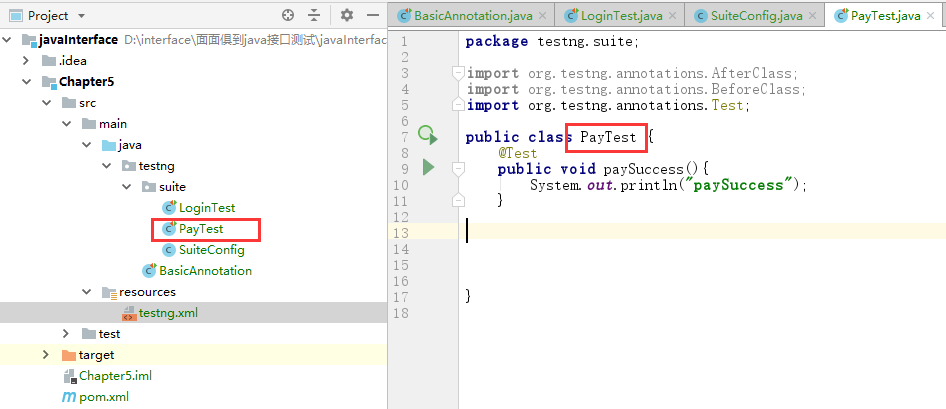


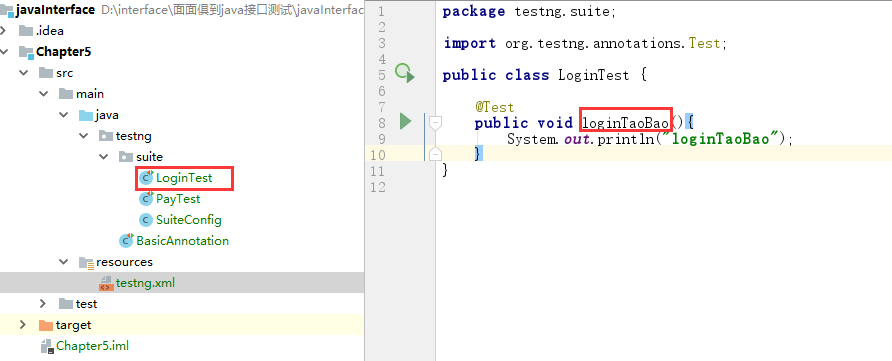
## 5-7套件测试

主要是利用testng.xml进行测试，配置testng.xml

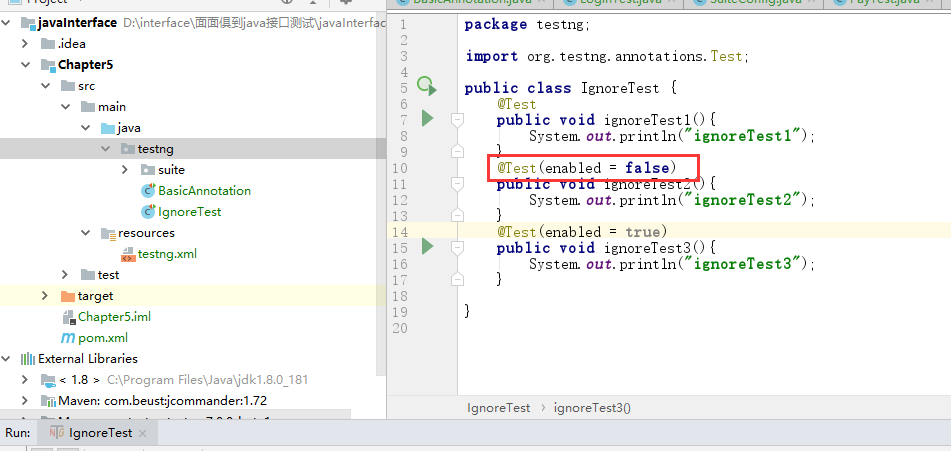




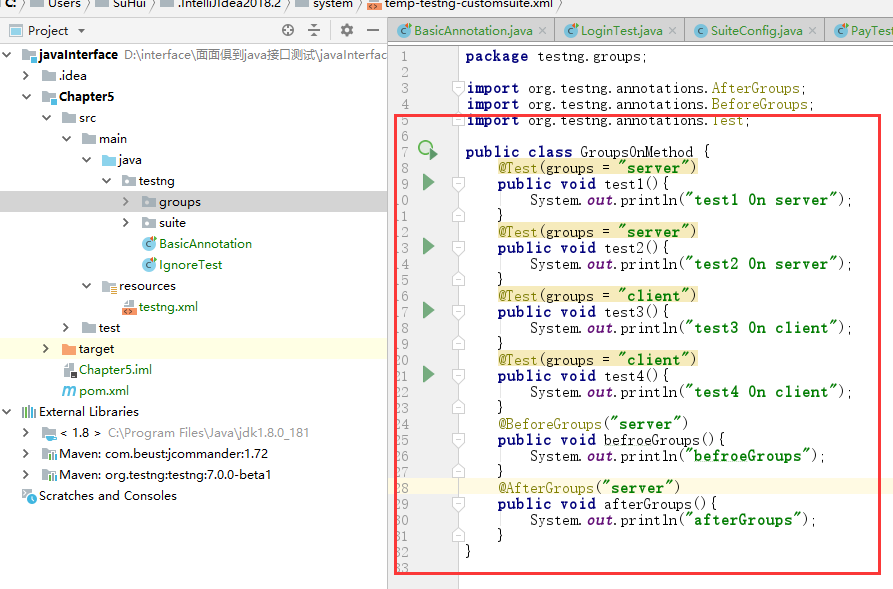


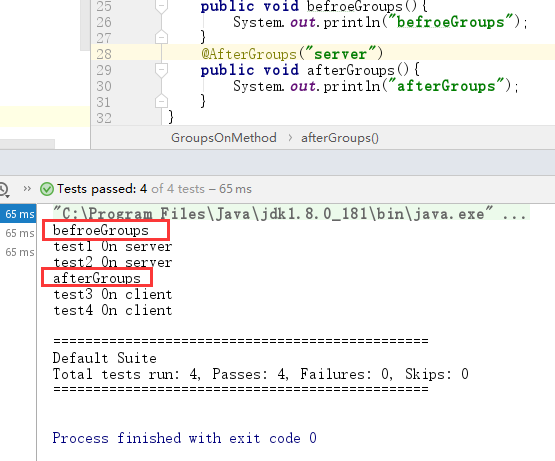


## 5-8忽略测试



## 5-9组测试中的-方法分组测试

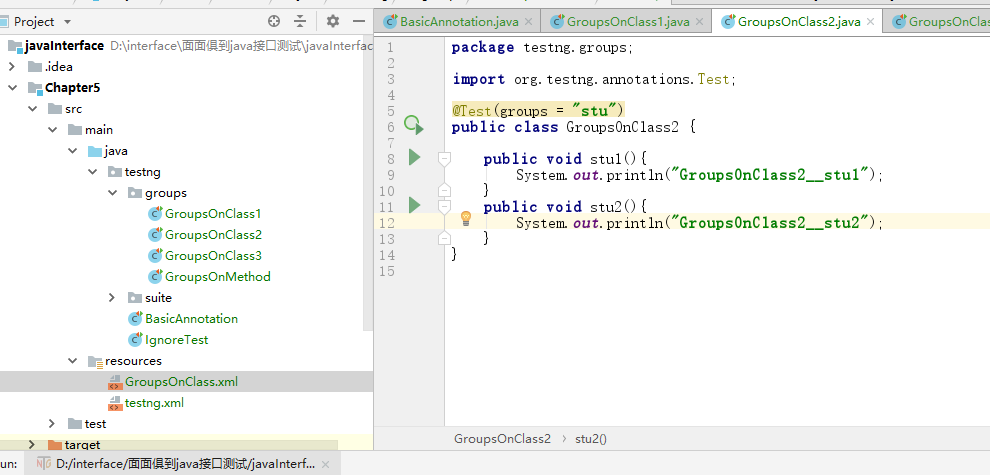


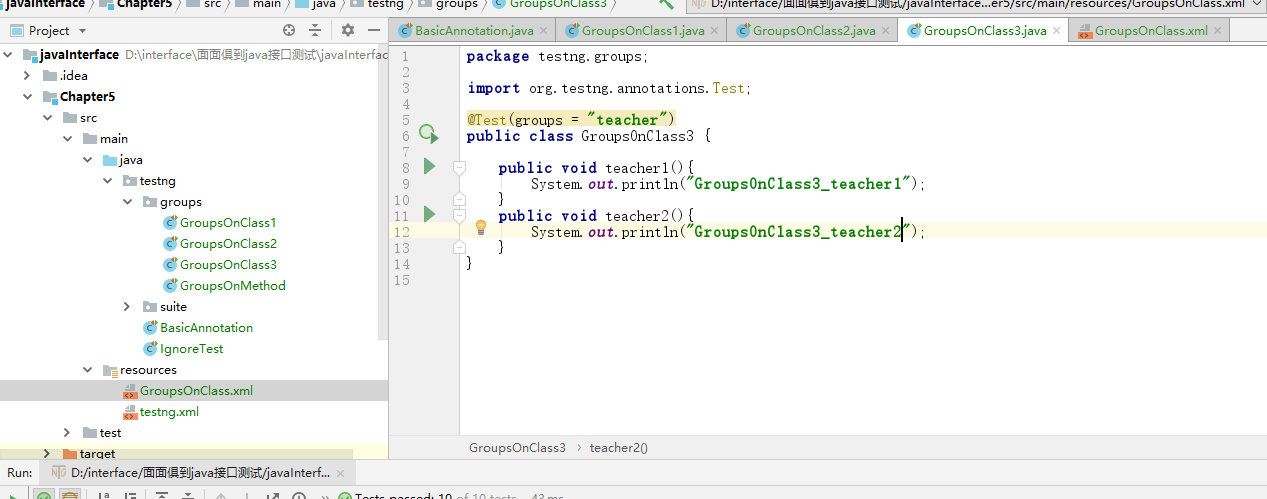


## 5-10组测试中的-类分组测试

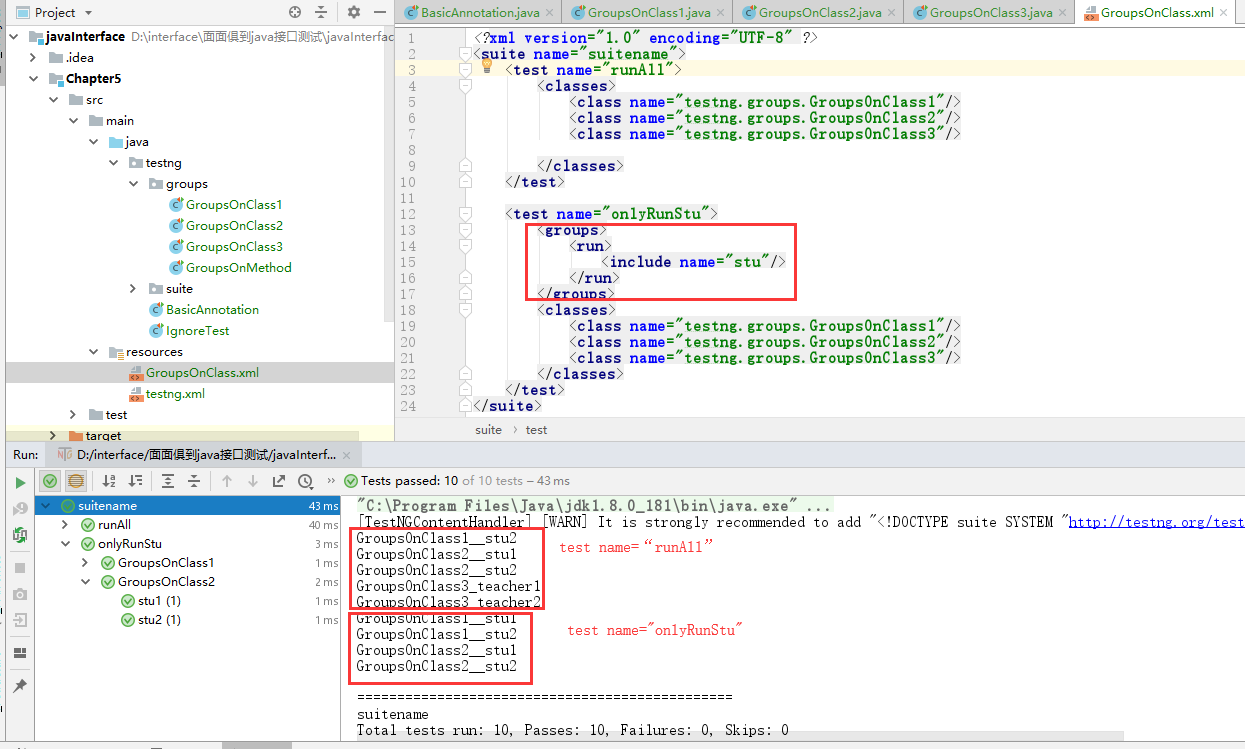
新建三个类如下图：







新建xml运行如下图，结果如下：



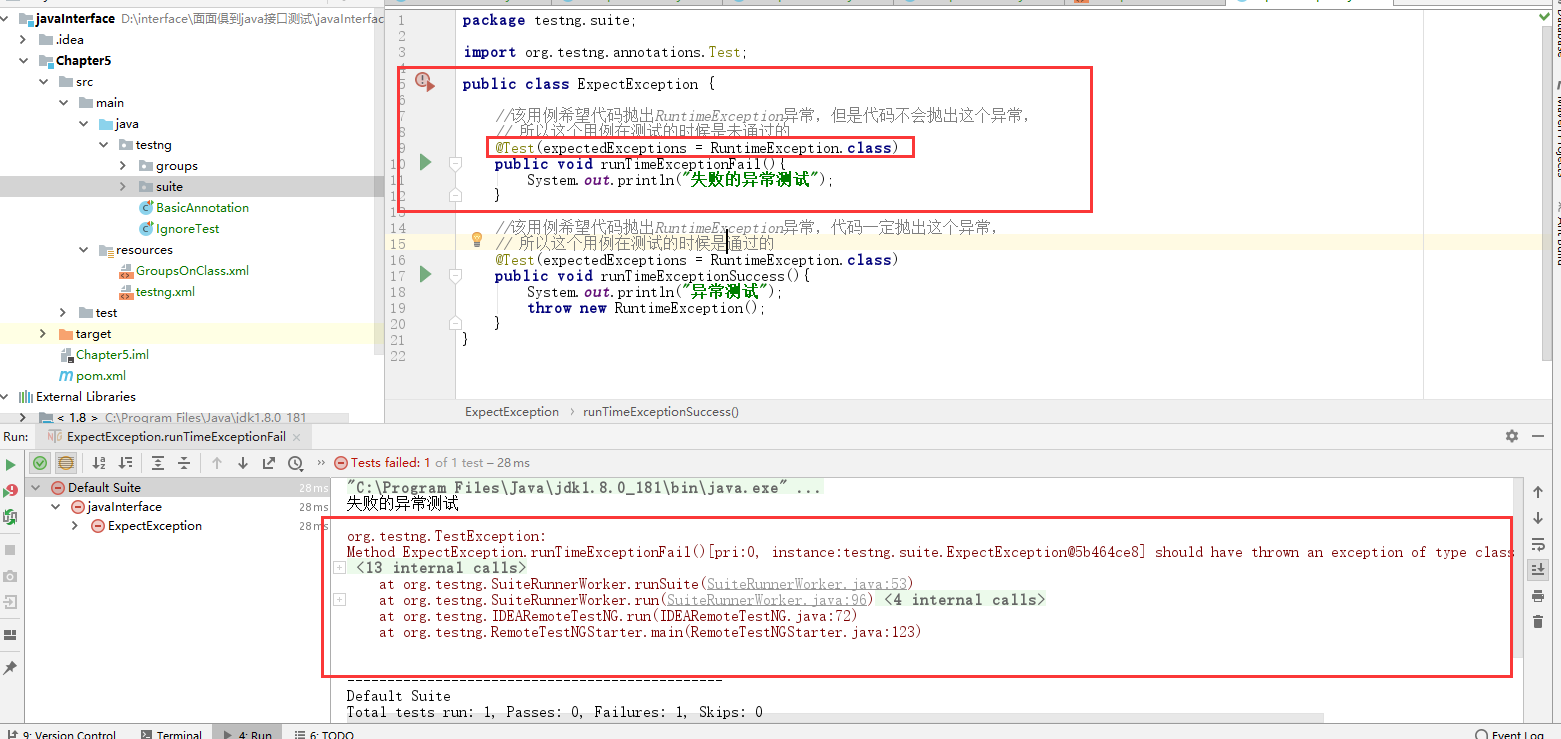
结论：

在类前面增加@Test（groups=“xx”）

在xml中增加标签，可以指定运行xx类下的测试用例

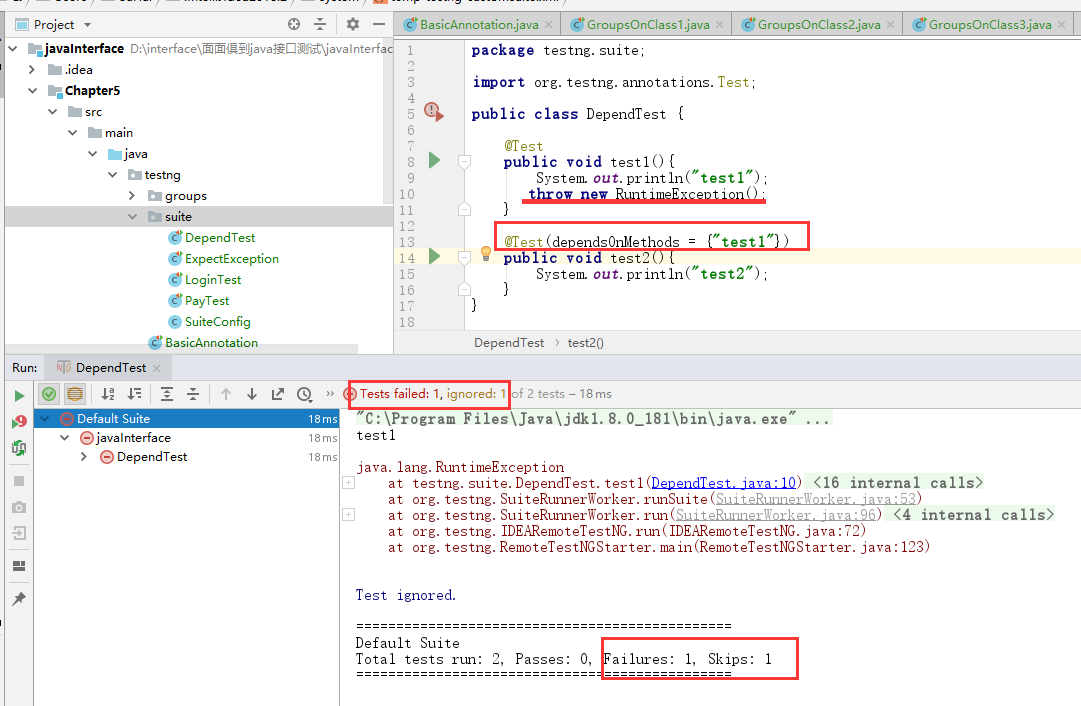
<**groups**>  
 <**run**>  
 <**include name="xx"**/>  
 </**run**>  
</**groups**>

## 5-11testng异常测试

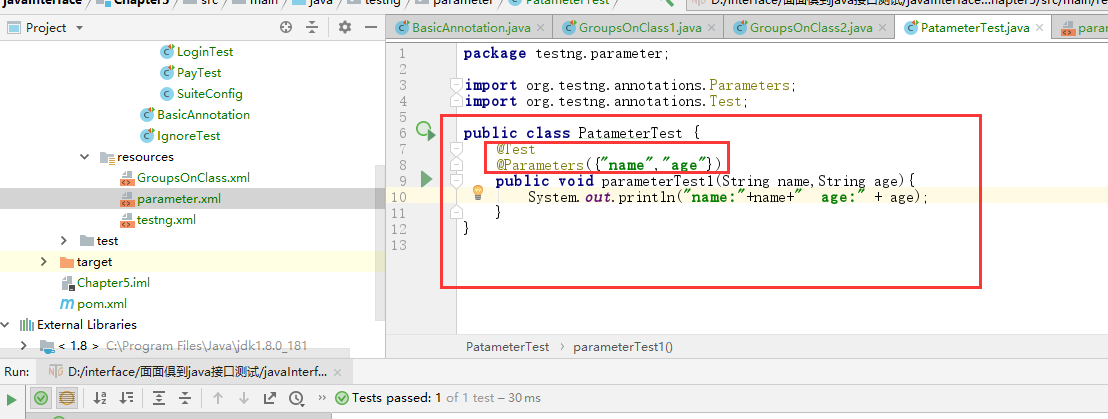


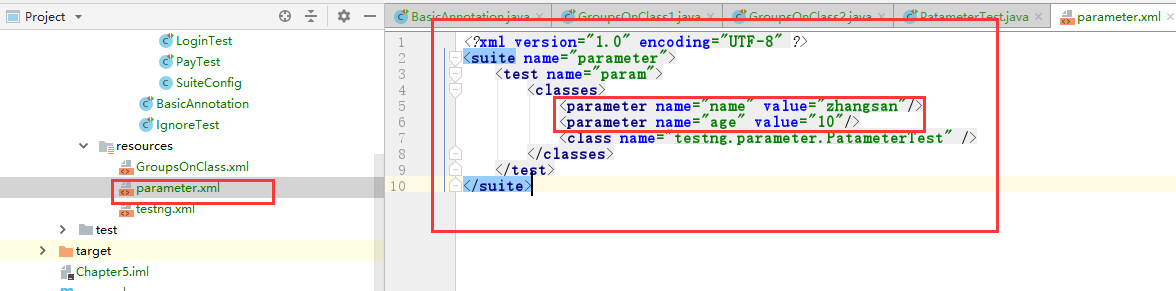


## 5-12testng依赖测试



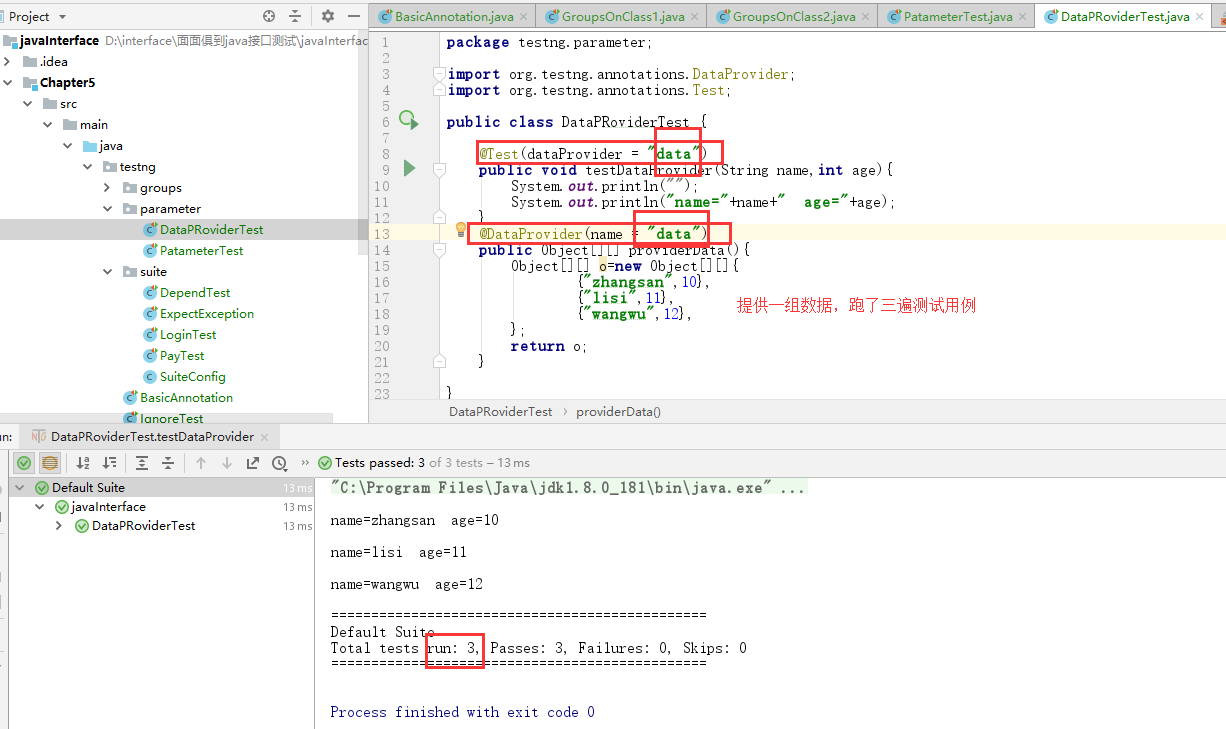
## 5-13testng参数化测试-XML文件参数化



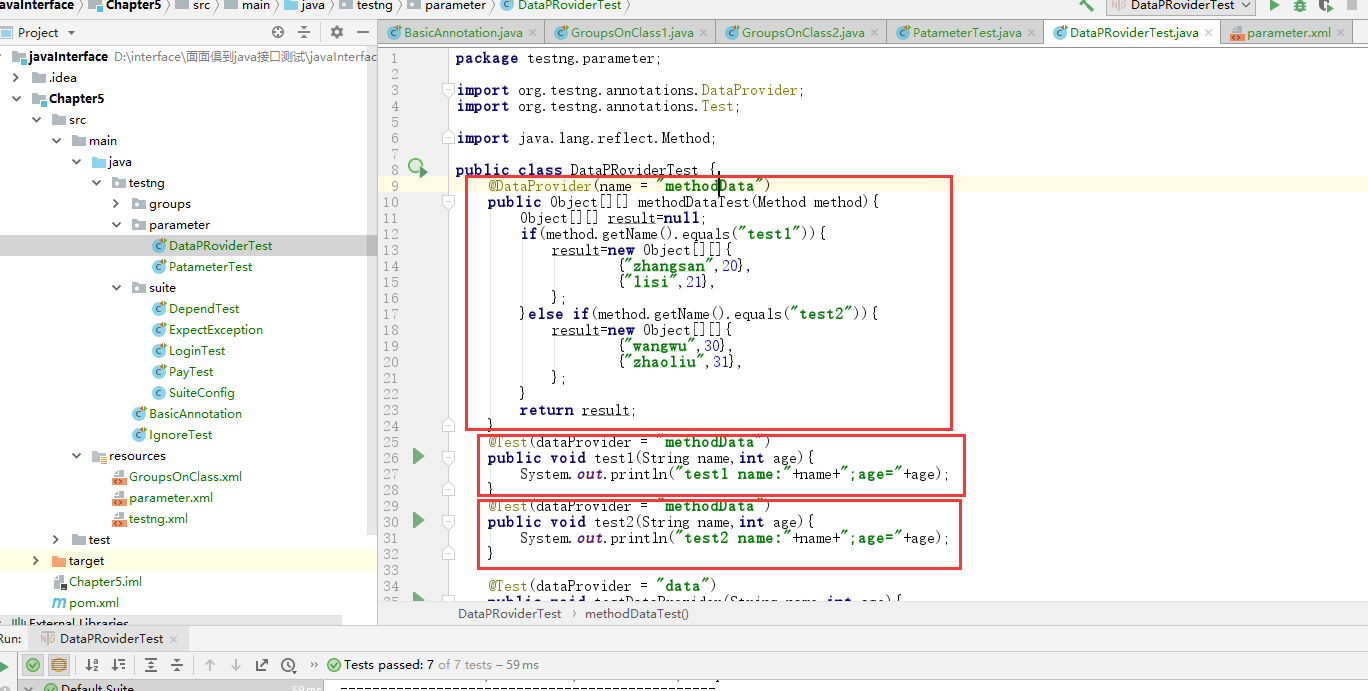


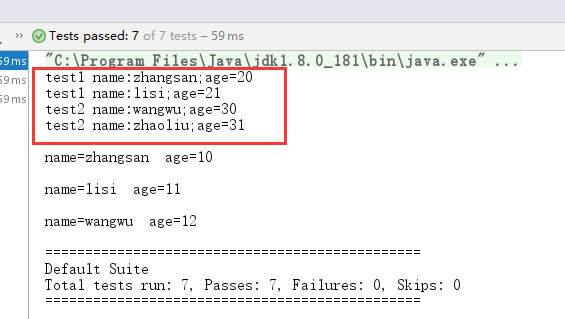
## 5-14testng参数化测试-DataProvider文件参数化

下图是针对参数进行测试

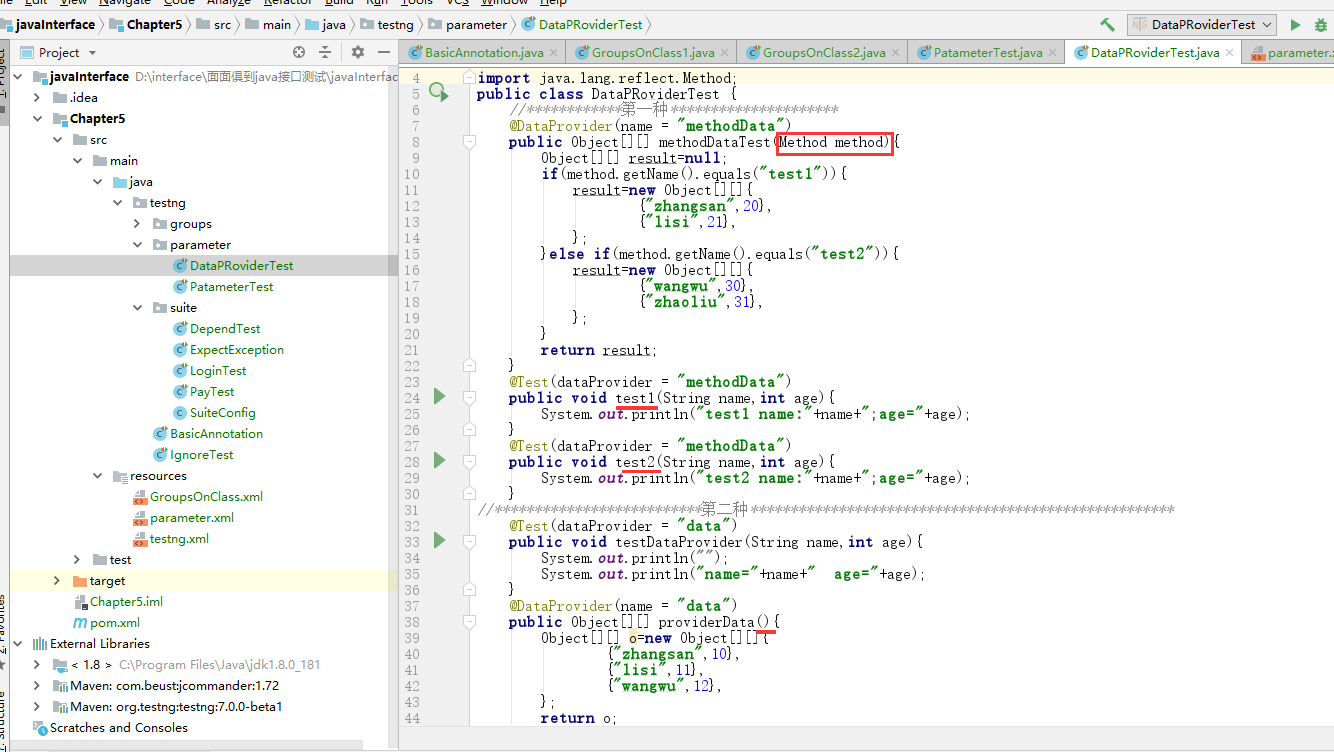


下图是针对方法进行测试





上面两种方法的总体代码如下图：



## 5-15多线程测试-注解方式实现

