AutoTest操作手册

新特性：

1.自动生成测试用例

2.测试代码更简洁，面对对象编码

3.数据准备更方便，可自动生成csv

4.数据验证更容易，直接断言数据库对象

5.配置文件更简单，只需一个数据库连接配置

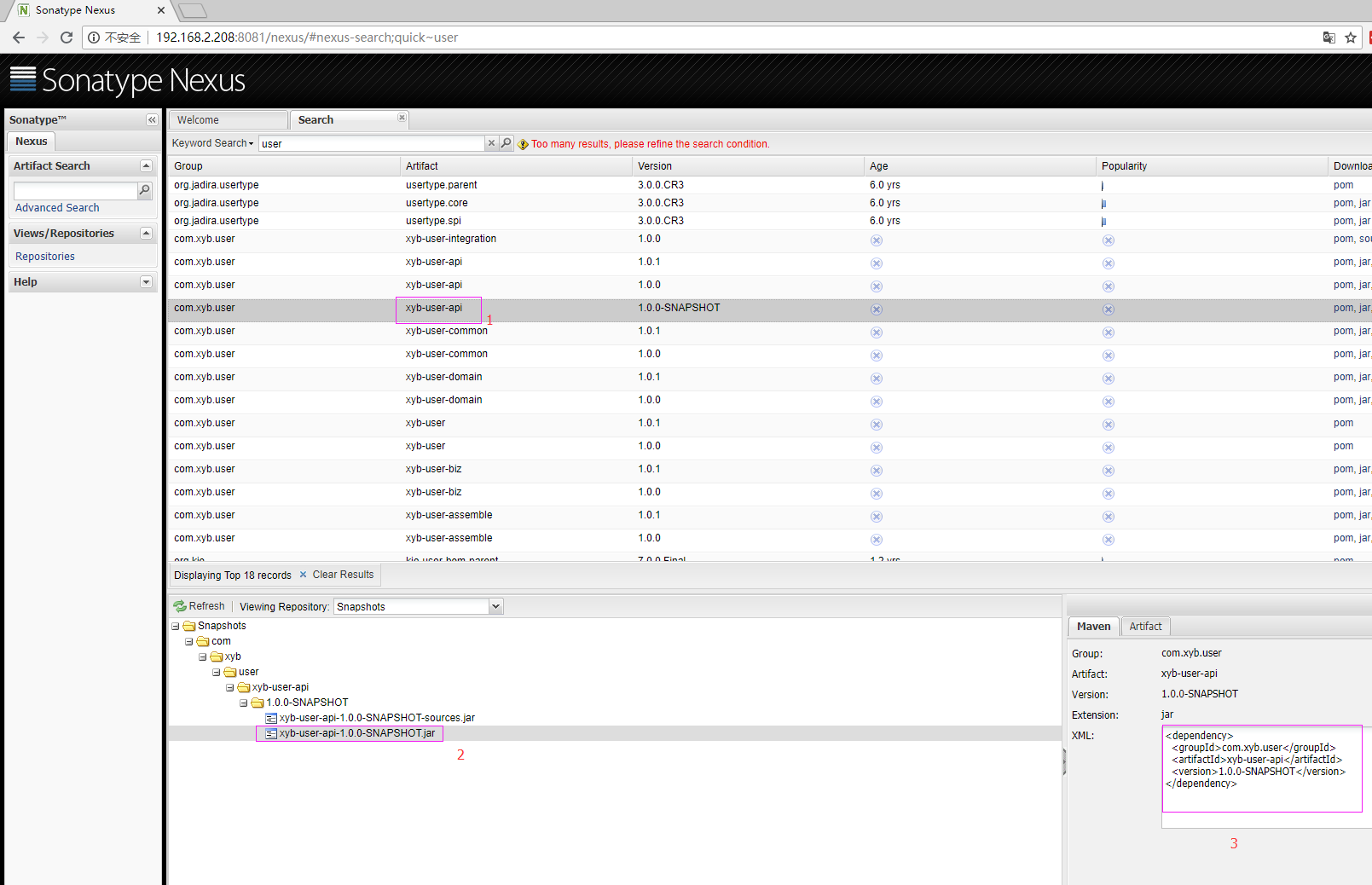
项目地址：

<http://192.168.2.221/test/autotest.git>

# 一、添加依赖包

仓库地址：

[http://192.168.2.220/#browse/search/maven](http://192.168.2.220/" \l "browse/search/maven)



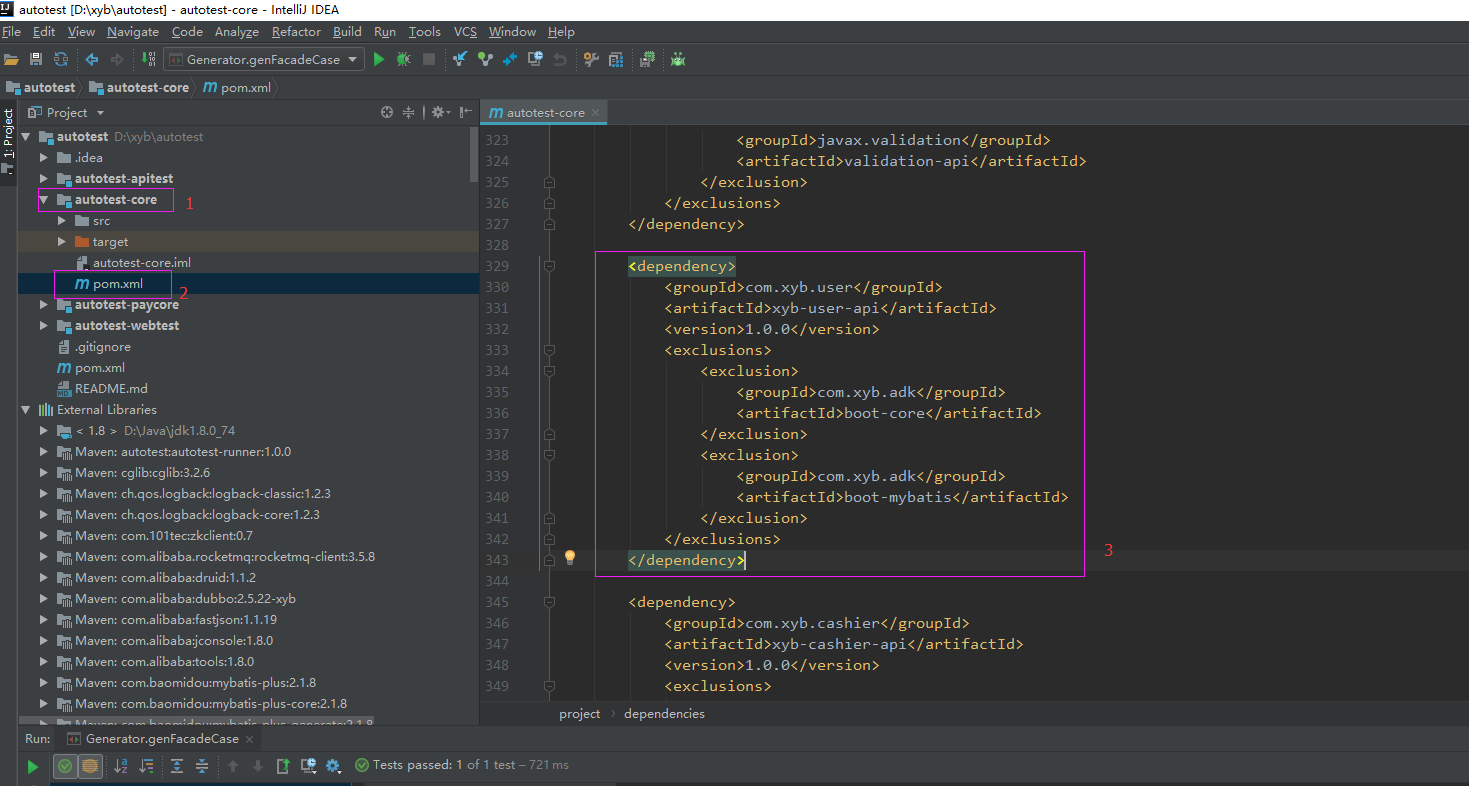
<dependency>

<groupId>com.xyb.user</groupId>

<artifactId>xyb-user-api</artifactId>

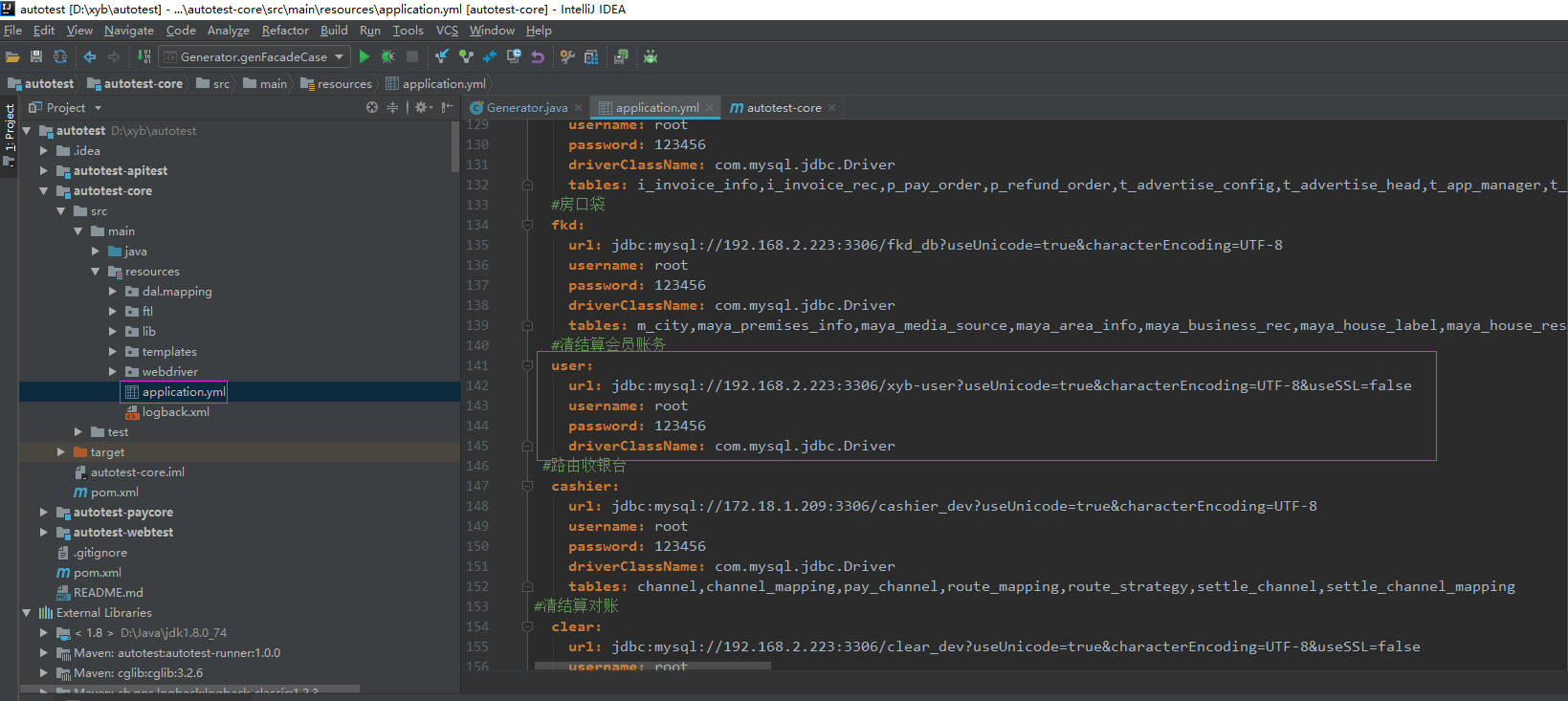
<version>1.0.0-SNAPSHOT</version>

</dependency>



# 二、配置

1.数据库连接



#清结算会员账务

**user**:

url: jdbc:mysql://192.168.2.223:3306/xyb-user?useUnicode=true&characterEncoding=UTF-8&useSSL=false

username: root

password: 123456

driverClassName: com.mysql.jdbc.Driver

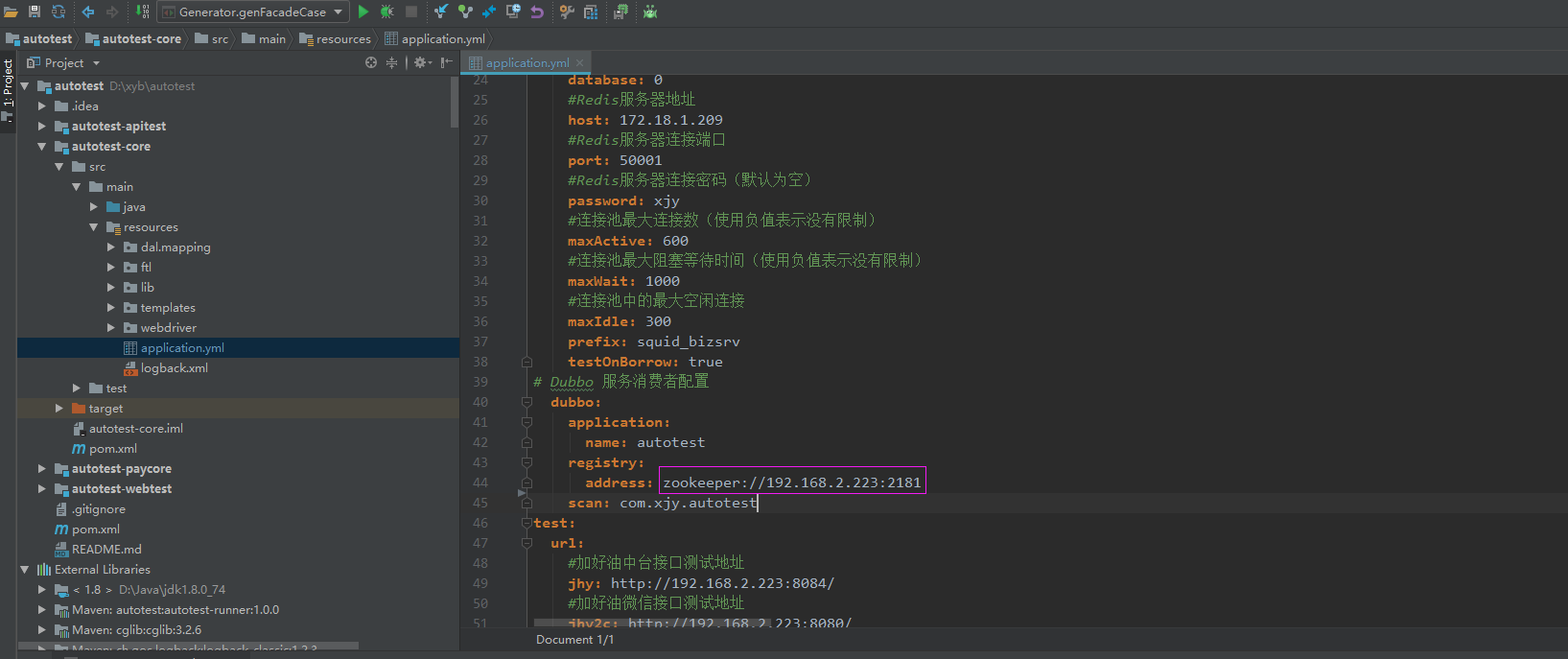
# text类型转string

columnOverride: trans\_req\_ids

#生成双入参删查改方法

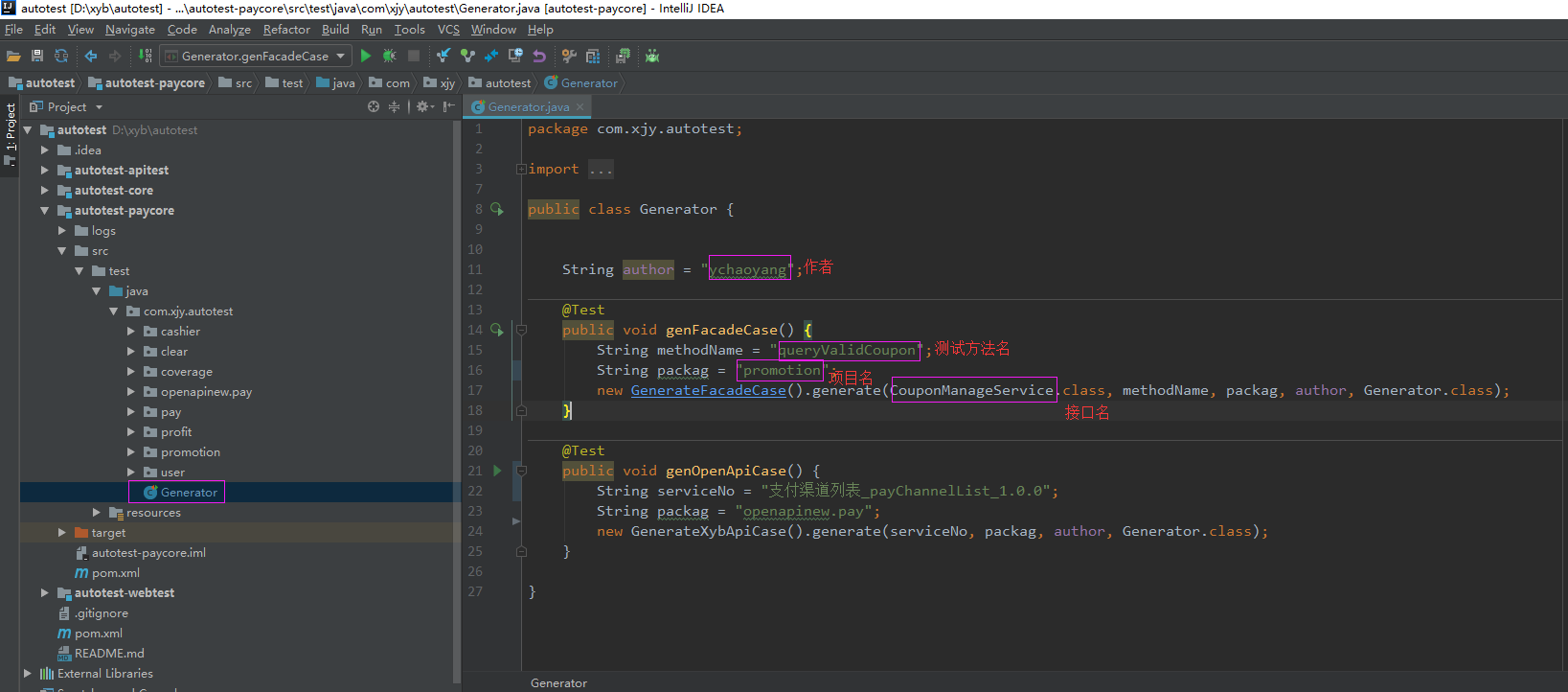
parms: account(user\_id+tag)

2.dubbo配置



# 三、开始

3.1 生成测试用例

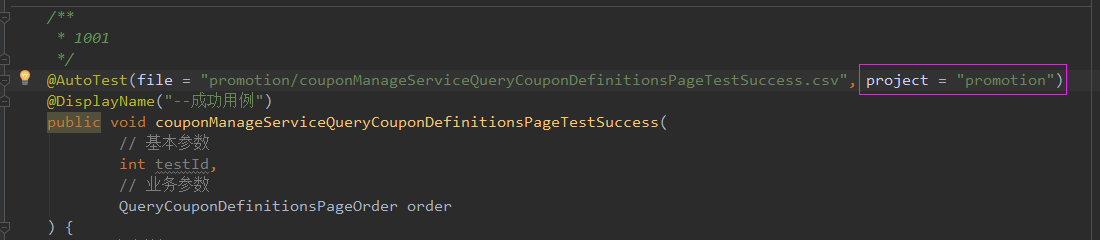


3.2 生成CSV文件

运行测试用例时，若没有找到csv文件则会生成对应的文件，如存在则不会生成

3.3 生成数据库操作类

在AutoTest注解中添加project，值填项目名（必须要与yml配置文件中的配置项名称一致，生成完成后去掉

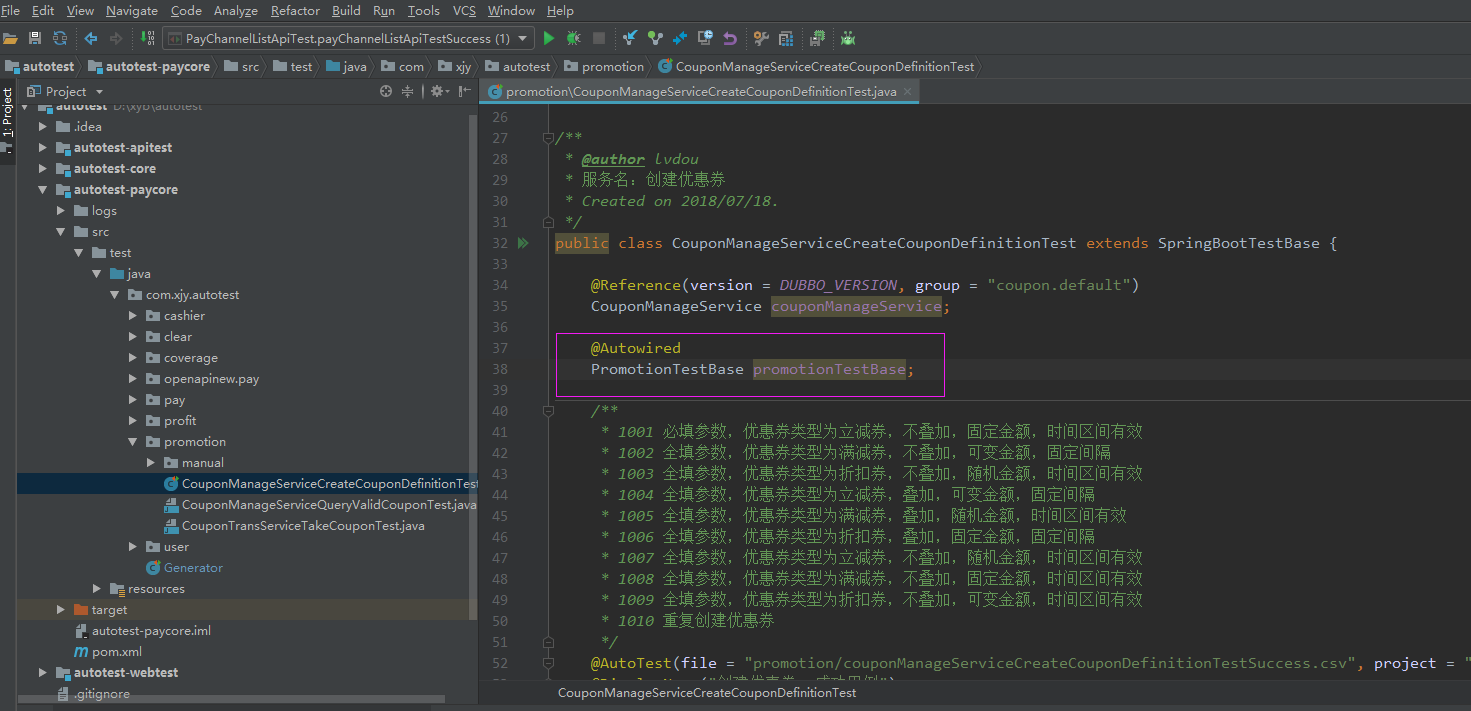


3.4 操作数据库

3.4.1 引入Base类

@Autowired

PromotionTestBase promotionTestBase;



3.4.2 增删查改方法

插入方法

insert+数据库表名

删除方法

clean+数据库表名+By+字段名

查询方法

find+数据库表名+By+字段名

更新方法

update+数据库表名+By+字段名

# 四、测试用例

**4.1 测试用例步骤**

1 清除数据：排除影响测试用例执行失败的数据项，保证测试用例每次执行都能通过

2 准备数据：准备测试用例需要的前置数据

3 测试过程：准备测试用例需要的入参

4 调用接口：调用需要测试的接口

5 结果验证：同步返回结果校验

6 数据验证：数据库校验，异步结果校验

7 清除数据：清理测试用例产生的数据，防止垃圾数据的遗留

**4.2 用例要求**

1、测试用例可重复执行，每次结果都是一样

2、测试用例简洁，可维护性高

3、尽量多的覆盖测试点

4、添加适当的注释

# 五、页面自动化—Selenide

## 5.1 打开浏览器

open("www.baidu.com");

## 5.2 关闭浏览器

close();

## 5.3 查找页面元素

### 5.3.1 通过Id查找

$("#kw")

### 5.3.2 通过className查找

$(".s\_ipt")

### 5.3.3 通过tagName查找

$("input")

### 5.3.4 通过元素属性查找

$("input[name=\"wd\"]")

### 5.3.5 通过文本文字查找

$(byText("百度快照"))

$(byPartialLinkText("百度首页"))

### 5.3.6 通过name查找

$(byName("百度快照"))

### 5.3.7 通过value查找

$(byValue("百度一下"))

### 5.3.8 通过标签名查找第几个

$("input",2)

## 5.4 输入数据

$("#kw").setValue("selenide");

## 5.5选项选择

$(By.id("userType")).selectOptionByValue("PERSONAL");

## 5.6 刷新页面

refresh();

## 5.7 截图

takeScreenShot("登录页面");

## 5.8 页面文本断言

$("#TANGRAM\_\_PSP\_10\_\_footerULoginBtn").shouldHave(text("用户名登陆"));

5.9 操作下一个页面

switchTo().window(1);

# 六、自定义sql

自定义sql适用场景：

1. 查询等方法入参超过3个及以上；
2. 需要用到时间段等复杂查询
3. 需要查询统计信息sum、count等
4. 需要多表联合查询

## 6.1 操作类

类型：interface接口

命名：数据库表名 + Mapper

类路径：与DAO文件相同

例子：

package dal.dao.gas\_silverbolt;

public interface GasTradeOrderMapper{

}

使用时，直接引入该接口即可用

@Autowired

GasTradeOrderMapper gasTradeOrderMapper;

## 6.2操作方法

命名：find/update/delete + 表名 + By +字段名

注解：@Select @Update @Delete

例子：

@Select("select \* from gas\_trade\_order t where " +

"t.partner\_id =#{partnerId} " +

"and t.trade\_type =#{tradeType} " +

"and t.station\_id =#{stationId} ")

public List<GasTradeOrderDO> findGasTradeOrderByPartnerIdAndTradeTypeAndStationId(

@Param("partnerId") String partnerId,

@Param("tradeType") String tradeType,

@Param("stationId") String stationId);

## 6.3 返回值

1.返回全表信息

List<GasTradeOrderDO>

2.返回数组

List<String>

3.返回自定义字段

Map<String, Object>

4.返回自定义数组

List<Map<String, Object>>

## 6.4 注意事项

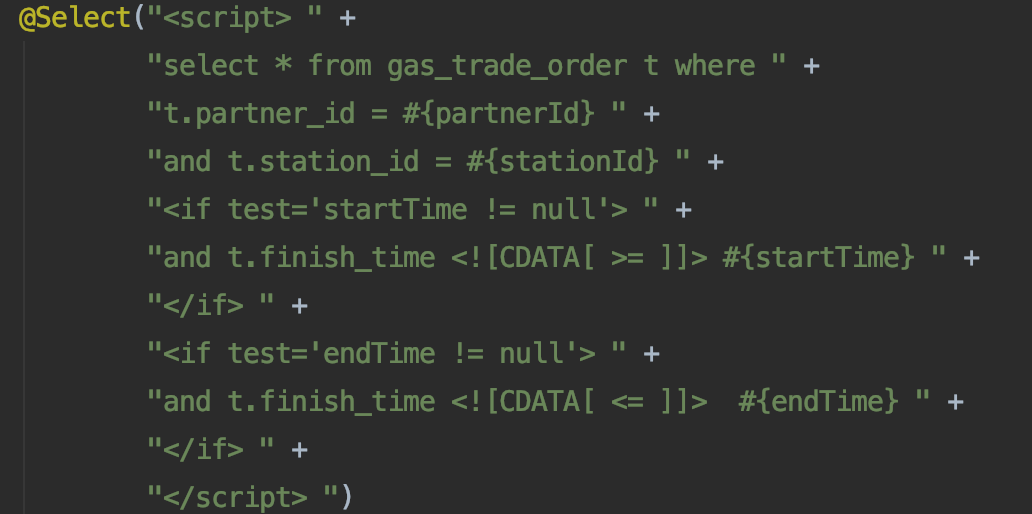
1.当sql中需要用到in条件查询时需要引入注解

@Lang(SimpleSelectInDriver.class)

2.当方法入参名称与实体中定义不一致时需要添加参数注解

@Param("tradeType") String tradeType

3.sql需要做非空判断时，需要在sql中加入<script>标签包围，写法与xml类似



4.当<script>标签里成对出现大于小于符号时需要做下转换（xml会当成尖括号）

<=需要写成<![CDATA[ <= ]]>

>=需要写成<![CDATA[ >= ]]>

5.添加好注释信息，方便使用

6.实例请参照

<http://192.168.2.241/test/autotest/blob/nuomi2/autotest-core/src/main/java/dal/dao/gas_silverbolt/GasTradeOrderMapper.java>