# 单元测试的问题和总体解决思路

在msa层，单元测试会遇到如下问题

1. 数据库依赖。在本地安装mysql支撑DAO层，每个测试类负责填充和销毁自己所需要的测试数据。也可以考虑使用内存数据库，例如H2 Database模拟，优点是不需要在每个打包环境安装mysql，缺点是在DML和DDL层面都会有一些不兼容的地方，如果遇到不兼容的地方，则需要在mybatis映射中处理，测试用例才能通过。
2. 消息队列依赖。使用spring-cloud-stream的TestSupportBinder完成，不需要安装特定的消息队列支撑。Spring工具支持捕获应用层析发出的消息以及给Listener注入消息。
3. 服务间依赖。服务间一般不存在直接依赖，更多是通过消息队列完成，因此可以通过第二点完成测试：消息发送端通过spring工具检测消息发送是否成功，消息接收端通过spring工具注入消息，驱动任务。在少量存在服务间接口依赖的情况，通过打桩方式完成测试。
4. 定时任务测试。
   * 定时任务一般需要调用msa服务，但是不提供Controller接口。定义任务正常执行的依据是业务逻辑是否正确，是否有调用msa的接口，因此可以使用打桩方式提供msa接口，并提供检测方式判断接口是否被触发。
   * 保证定时任务能够按照预期方式调度的代码，一般位于定式服务框架中。框架通过一些监测点数据，可以完成单元测试

在mt层，单元测试的主要问题是对msa以及SOA网关的依赖。建议按照以下方式执行单元测试：

1. Msa依赖。首先由msa定义接口，mt完成打桩实现。在mt代码中，通过Profile方式，在测试环境和运行环境注入不同的实现，以实现功能。
2. Soa网关依赖的接口一般是线程接口。为了保证单元测试能够独立运行，最好提供打桩实现。

在mt中推荐使用打桩执行单元测试，主要原因是

1. 直接依赖msa实现，需要了解数据库设计才能完成测试数据注入。导致聚合曾测试用例强依赖服务层实现，如果数据库结构或者service实现上有变化，mt的测试用例会失败
2. 在msa实现完成之间，mt无法自测代码，也不能给APP提供模拟数据测试。

# MSA测试

## msa-unit-test模块

服务层测试均需要以test scope添加对msa-unit-test的依赖。这个模块的pom如下



其中，dbunit用于完成数据库注入和检查、spring-cloud-stream-test-support用于完成消息注入和检测。

模块提供了测试基础类，用于完成对spring-dbunit的配置、MockMvc的初始化、公共测试配置注入等。

服务层Controller测试均继承这个类，以简化并统一测试配置。MockMvc使用webApplictionContextSetup测试方式并添加了CachingRequestBodyFilter，使测试环境和正式环境具有具有相同的filter、interceptor和advice环境。



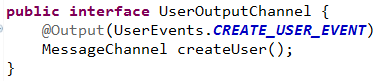
## 测试案例

在shop-service-provider下建立了一个UserController用于演示测试。这个服务演示了数据库插入、更新测试；消息队列发送和侦听测试。

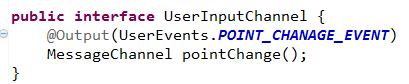
### 待测试组件实现

1. 消息队列组件

定义了消息发送topic，createUserEvent，每次创建一个用户，向这个topic发送一个消息，

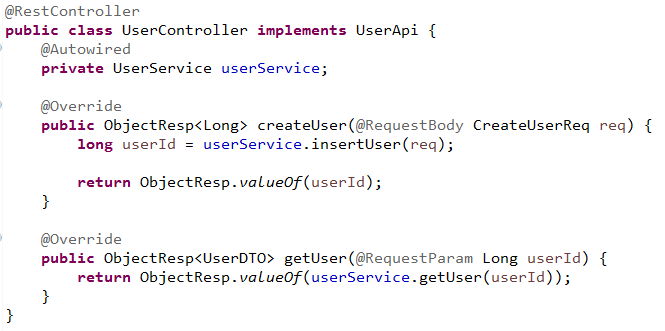


以及消息定于topic，pointChangeEvent，每次监听到积分变化，会根据消息内容修改数据库。



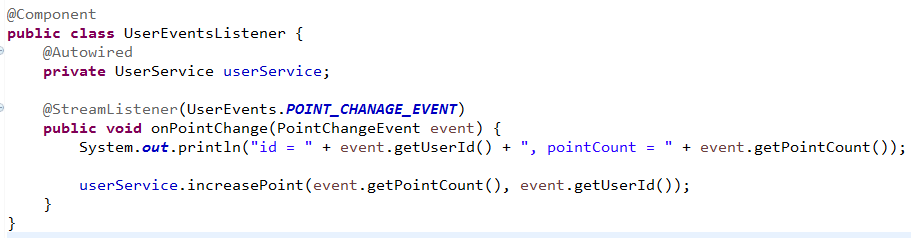
1. Controller

Controller实现简单，提供了一个创建用户和一个查询用户的接口。

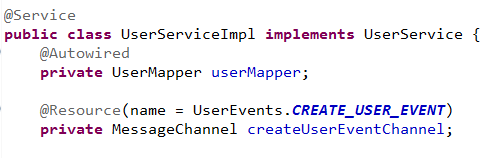


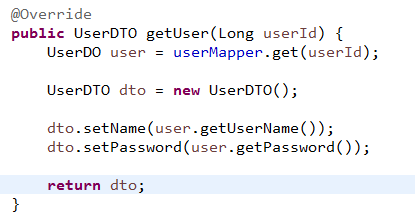
1. 消息侦听

消息监听方法每次收到一个消息修改用户积分。

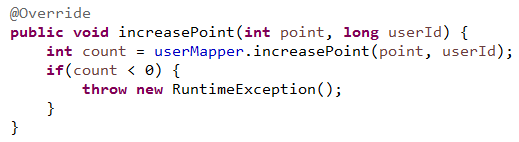


1. 服务层实现





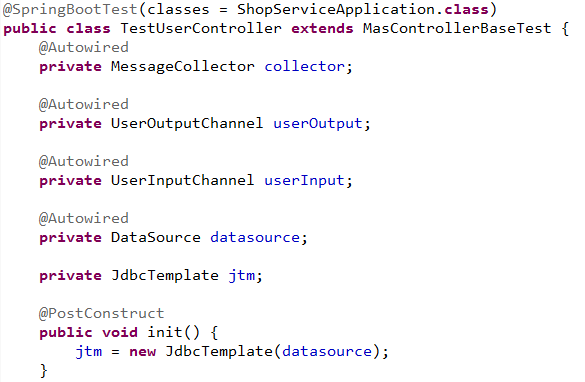




### 测试

1. 测试准备

为了测试消息队列、使用SpringTestUtils数据库检测工具，这里注入了MssageCollector、DataSource、JdbcTemplate等组件。



1. 测试创建用户

创建用户需要测试两个方面，数据库是否插入成功，消息是否发送成功。测试步骤如下

1. 构造请求，以json格式发送给/v1/shop/user/create。使用jsonPath组件判断返回码和返回的Id是否正确；
2. 使用JdbcTestUtils工具，判断vcity\_user的数据行数是否为1。由于初始数据为空，因此插入插入数据成功即等价于数据行数为1；
3. 使用MessageCollector组件，检测UserOutputChannel#createUser topic中是否有消息。



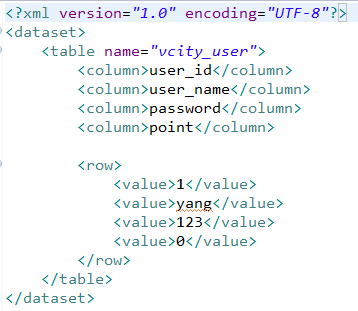
在上面的测试中，在testInsert方法上使用了@Transactional标签，用以启动测试事务，Spring在测试完成之后，会自动回滚事务，以保证被测试方法如果有修改数据库的行为，在测试完成后行为被清空，不影响后续测试。

1. 测试查询用户

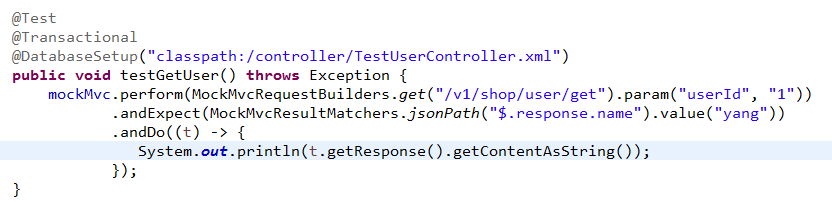
为了测试查询用户，需要给数据库注入数据。可以写sql脚本，[然后用@Sql标签注入。这里选择使用DBUnit完成数据注入。DBUnit](mailto:然后用@Sql标签注入。这里选择使用DBUnit完成数据注入。DBUnit)具有两点优势：

1. 使用XML描述数据，结构清楚并且与数据库无关；
2. 提供了检测数据库字段变化的方法。后面测试积分更新会演示这种用法。

DBUnit注入的测试数据如下

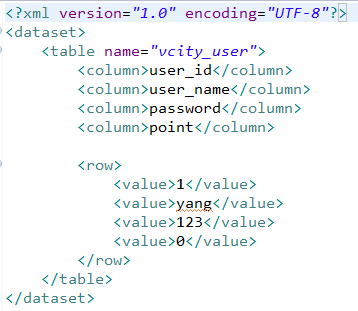


在测试中，使用userId = 1调用/v1/shop/user/get接口，并检测name字段。

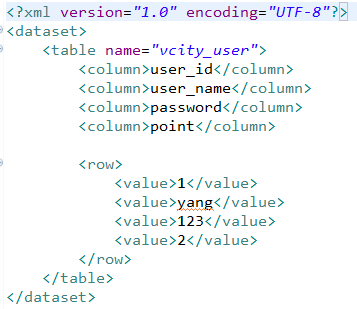


1. 积分更新测试

积分更新测试需要注入测试数据，并且检测测试完成后，数据库的修改是否满足要求。测试数据如下：



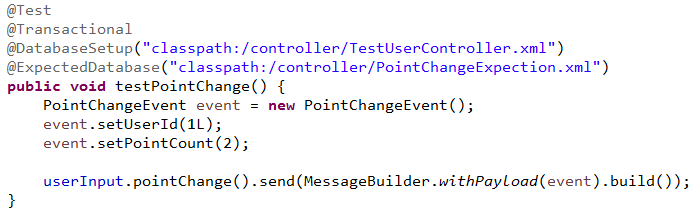
为了检验测试结果，需要额外写一个数据脚本，例如：



这个测试结果认为，在测试完成后，积分增加了两点。

测试中，直接给UserInputChannel发送了一个事件，通知id=1的用户积分增加了两点。这个事件将驱动事件监听任务，如果任务成功，数据库积分修改2点。

这里使用了@ExpectedDatabase检测结果是否与预置的数据集相同。



# MT测试

MT的测试与MSA不同。MT不直接面向数据库、消息队列等，而是调用底层接口或者第三方接口。

为了独立测试MT，不依赖MSA和第三方接口，需要执行打桩测试，主要用到的工具是Mockito。

与MSA测试一样，MT提供了公共组建mt-unit-test，需要执行单元测试模块都需要以test scope依赖这个组建。

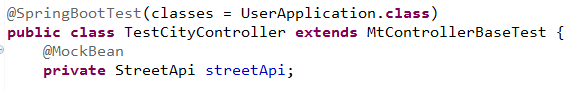
这里以测试CityController和GeoInfoApi演示了如何进行打桩测试。

## CityController接口测试

下面演示如何打桩测试CityController获得精品街道列表的接口和检测区域是否3d话的接口。

1. 测试准备

CityController中，使用@Autowired注入了Feign实现的StreetApi。为了打桩测试，需要使用@MockBean标签，将Mockito的打桩类注入到CityController中。如下



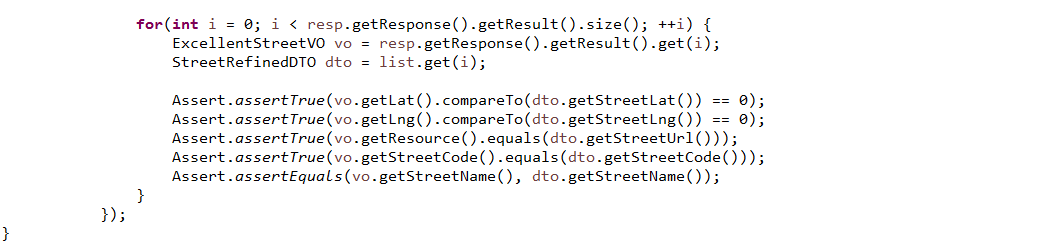
在调用streetApi获得精品街道时，可能获得三种结果：有精品街道、精品街道为空、服务器发生异常，下面分别测试。

1. 有精品街道

测试步骤如下

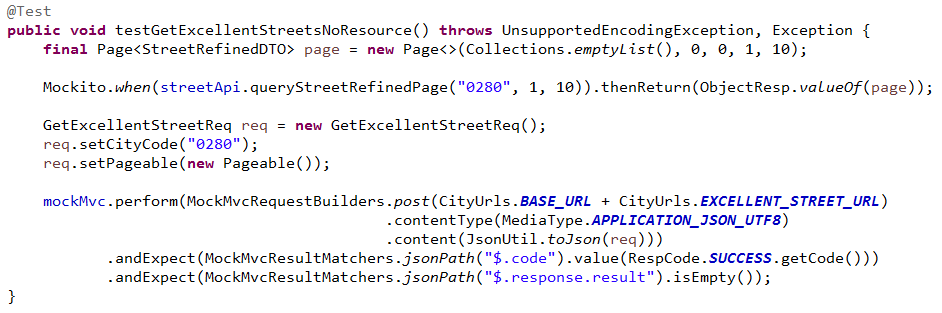
1. 准备测试数据。这里准备了一个street dto的list，模拟MSA返回的数据。
2. 使用Mockito，给streetApi#queryStreetRefinedPage打桩。
3. 调用CityController接口，检测CityController返回的ObjectResp<Page<ExcellentStreetVO>>是否正确。



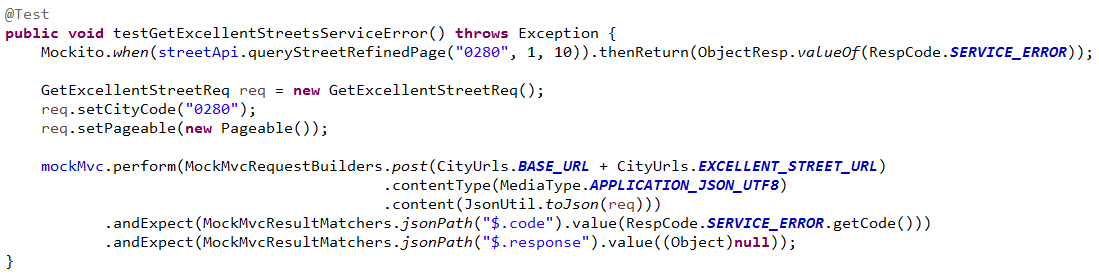


1. 没有精品街道

模拟MSA返回了一个空列表的Page。



1. 模拟MSA异常



1. 区域是否3d化

检测区域是否3d化，需要SOA接口。这里讲测试分为两个部分：

1. CityController实现是否正确。
2. SOA网关调用是否正确。这一部分在下一节演示。

CityController的测试与测试精品街道一样，只需要使用@MockBean注入GeoInfoApi，然后给getGeo3dCoordinate等打桩。



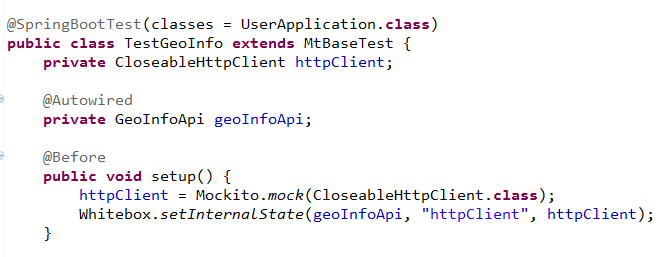
## 测试GeoInfoApi实现

GeoInfoApi实现依赖于SOA。在内部完成数据封装、签名检验之后，最后调用HttpClient访问SOA获得地理信息数据。

单元测试可以测试数据封装过程是否满足预期，对返回数据的解析是否正确。但是不能检测签名校验等问题。

1. 测试准备

GeoInfoApi实现依赖私有的HttpClient，因此首先需要将这个字段替换为打桩实现。



1. 测试3d坐标转换

在测试过程中，可以在打桩实现里面检测http请求格式是否满足要求。这里简化了测试，只检测了返回值解析是否正确。对于，参数封装过程，只能保证没有异常。

测试过程如下：

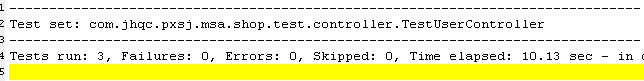
1. 首先打桩getStatusLine和getEntity方法。返回正常数据。
2. 然后，调用接口，并检测返回值是否符合预期。



# 测试插件

在mt-parent和msa-parent pom中，都配置了下面两种测试插件。

Surefile：用于生成测试报告。报告结果，在target/surefire-reports目录中。这个结果只会描述简单信息，例如测试了多个个case、成功和失败数量，并且会列举出测试失败的case



Jacoco：用于生成测试覆盖率报告。在test的所有case通过之后，jacoco会在target/site/jacoco-ut生成可视化的覆盖率测试报告包括行覆盖率和分支覆盖率。例如

