# 05 belt项目测试用例开发快速上手（一）

# 概述

在本文中，会基于belt的crd项目，描述测试用例的开发流程

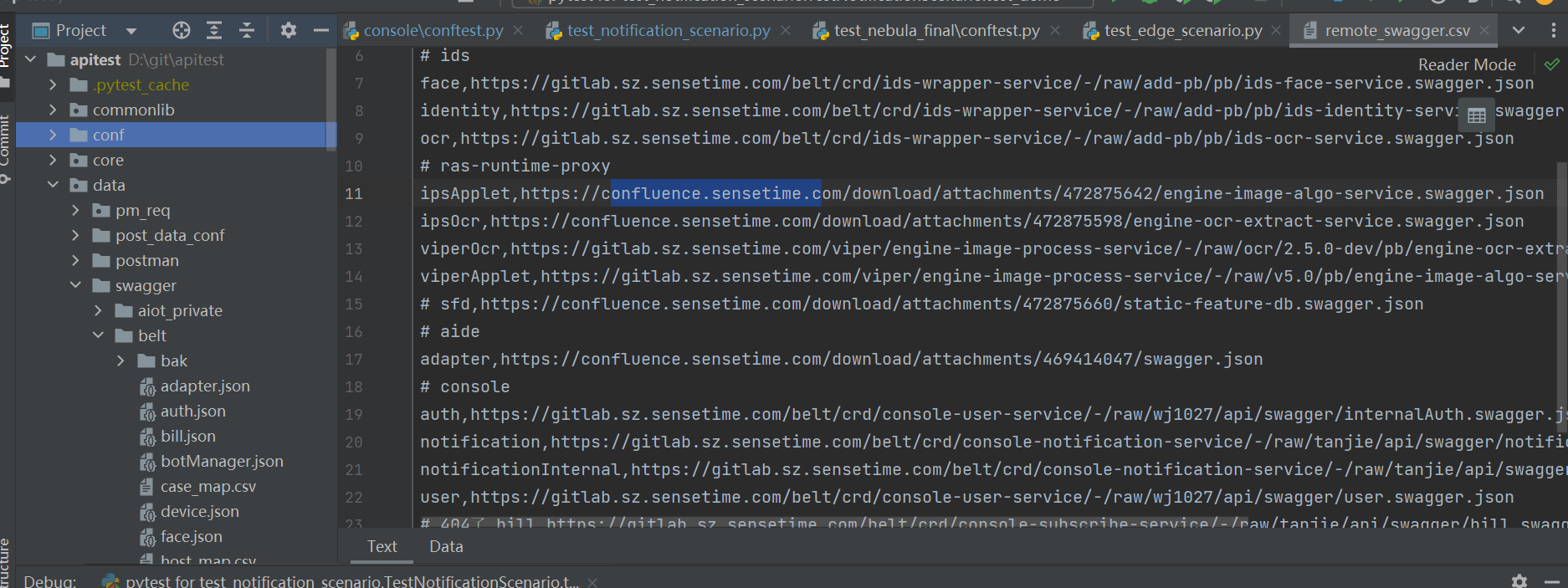
# 基础代码生成

根据swagger接口文档，生成基础代码

|  |  |  |
| --- | --- | --- |
| 时期 | 描述 | 命令 |
| 测试初期 | 生成一套基础代码并提交 @wangan | 更新本地swagger.json：**python main.py update\_swagger\_json -d belt**  json文件生成目录:apitest/data/swagger/belt/  生成代码：**python main.py gen\_all\_template -s belt**  生成代码目录：apitest/temp/templates/belt |
| 测试中期 | 当负责的模块接口文档有改动，单独更新相应的代码 @all |

**备注： update\_swagger\_json命令会检查线上的swagger跟本地的swagger是否一致，一致则说明开发没有更新（存在一个小问题：由于是检测的文件md5， 首次一定会不一致）**

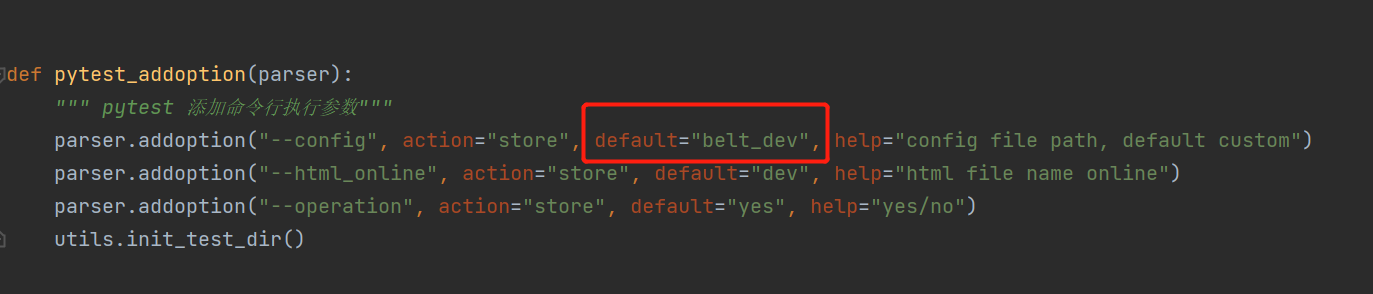
参考：[01 基础代码生成](https://ones.sensetime.com/wiki/#/team/JNwe8qUX/space/EBjCDQGt/page/GB3TdmuE)



# config配置文件

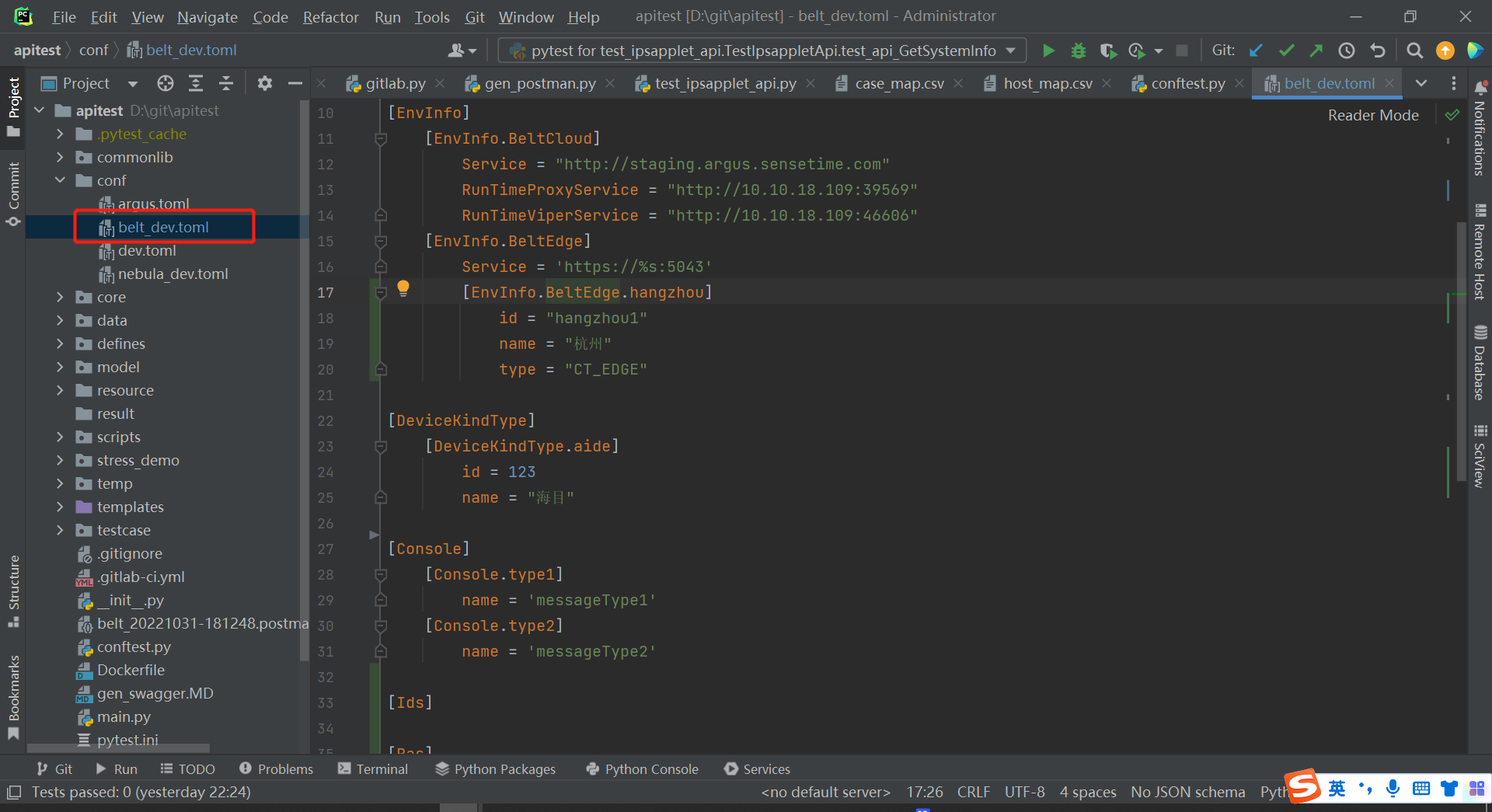
用于配置环境信息，如host，指定配置信息，如预制的一些类型名称等

1.默认的配置文件名称为belt\_dev, 指定运行的**配置文件名**是在apitest/conftest.py:pytest\_addoption中修改，方便用于IDE写代码时候debug



2.也可以使用pytest命令行执行测试用例的时候，通过pytest --config=belt\_dev来使用该config

3.该配置文件实际位置：apitest/conf/belt\_dev.toml  (配置文件是toml格式)

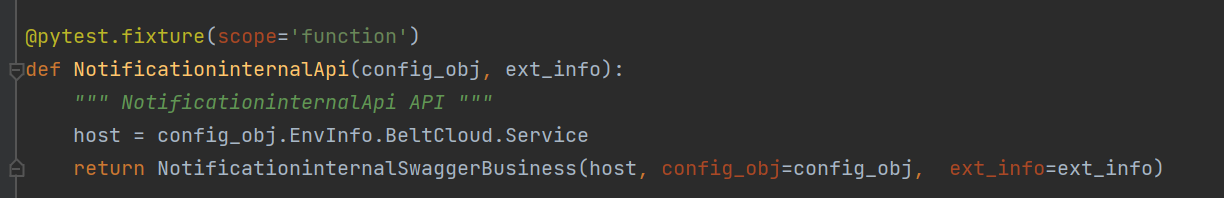


4. 配置文件中预先设置了公共使用的区域，及分模块自定义的区域：

|  |  |  |  |
| --- | --- | --- | --- |
| 区域 |  | 说明 |  |
| 公共区域 | ConfigMap | 联动pytest的fixture，实现动态参数化功能，详细可参考 [01 configObj配置文件对象](https://ones.sensetime.com/wiki/#/team/JNwe8qUX/space/EBjCDQGt/page/Dr69PckY) |  |
| EnvInfo | 测试环境信息  BeltCloud：中心云各个服务的host地址  BeltEdge: 边缘云信息，包含cluster（接入点）等 |  |
| DeviceKindType | 设备类型（内部定义的） |  |
| 模块区域 | Ras | ras自定义配置 |  |
| Ids | ids自定义配置 |  |
| Console | console自定义配置 |  |

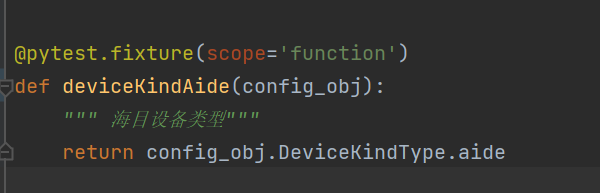
5.配置文件参数在代码中使用：

1）在conftest中使用，读取host：

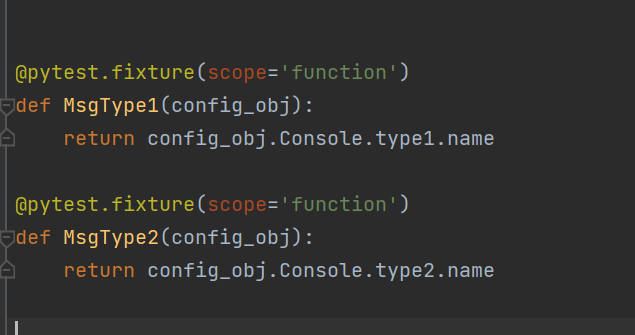


2）读取配置文件的指定项：

读取设备类型



读取console自定义的消息类型



**总结：无论是conftest还是testcase里，都可以通过config\_obj这个fixture获取到配置文件中的信息**

6.不同测试环境配置文件的切换

1）复制apitest/conf/belt\_dev.toml 为belt\_online.toml

2）修改belt\_online.toml中的环境信息，比如host等

3）执行pytest --config=belt\_online

# conftest.py文件

1.belt项目级contest.py

用于共享使用belt项目下的fixture， 比如api接口实例

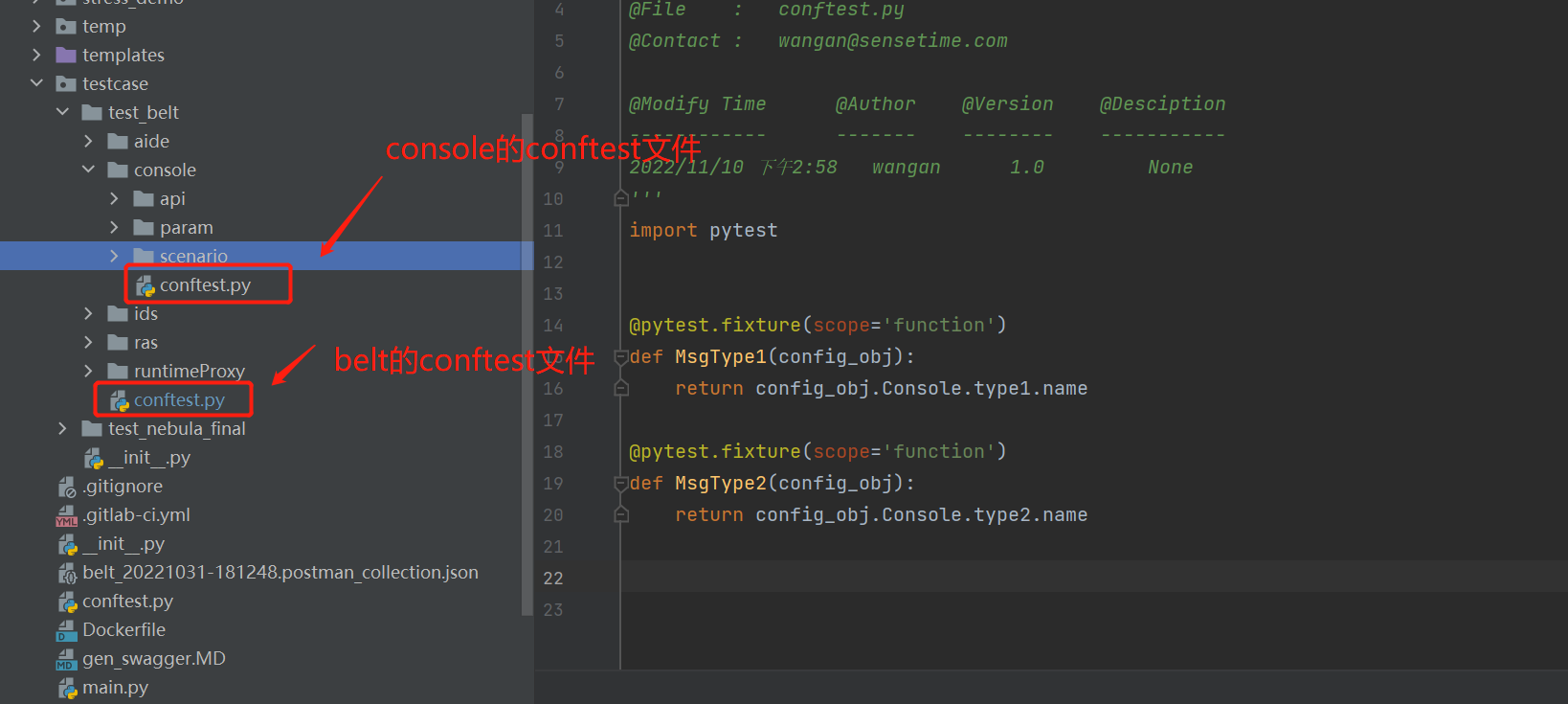
我们在写场景测试用例的时候，可以在一个测试用例中使用多个接口实例



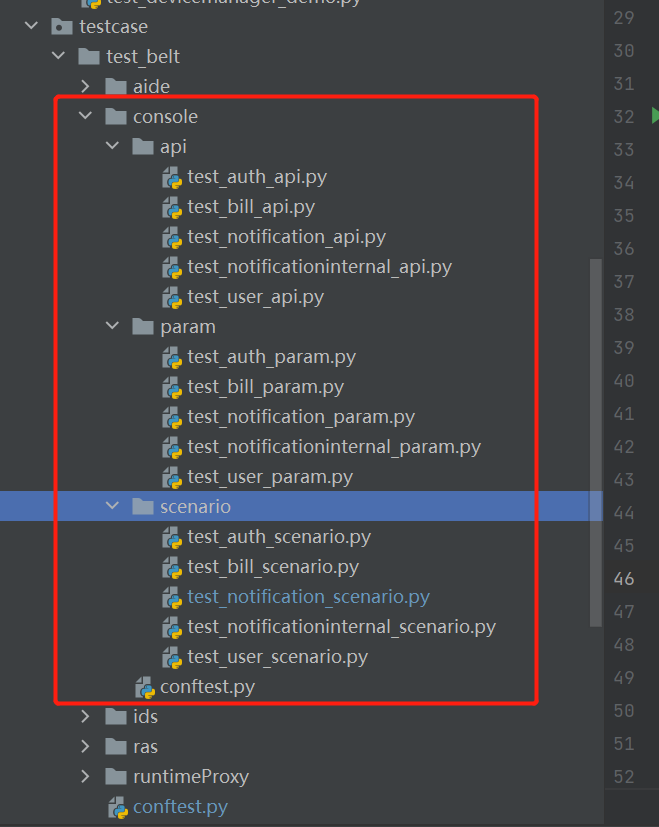
2.定义模块级conftest.py

在自己的项目下，创建一个conftest.py文件即可， 如果有同名的fixture会优先使用最近的conftest中的代码

以console举例：



# 测试用例编写



我们以console举例：

生成的代码包含了api, param, scenario三个目录，测试用例含义如下：

api：单接口测试

param：参数校验测试

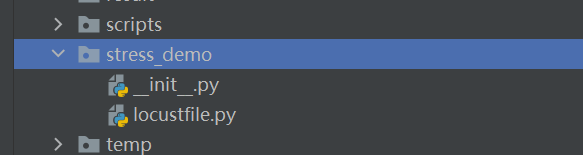
scenario：场景测试

这里就不一一举例了， 可以参考 [03 单接口测试用例开发](https://ones.sensetime.com/wiki/#/team/JNwe8qUX/space/EBjCDQGt/page/HRizHNZC)  [04 接口参数边界值测试用例开发](https://ones.sensetime.com/wiki/#/team/JNwe8qUX/space/EBjCDQGt/page/DAC8rG6u) [05 场景测试用例开发](https://ones.sensetime.com/wiki/#/team/JNwe8qUX/space/EBjCDQGt/page/B1Adstxh)

另外，我每天都在更新ras的测试用例代码，可以参考，同时也可以针对具体的case来问我，我可以提供1:1的代码编写演示

当然，除了以上三种，我们还可以根据实际情况，增加更多的测试类型的测试用例，比如stress等

我们的框架中，额外提供了locust压力测试的的demo，遗憾的是，目前还不能自动生成代码，未来会考虑支持



# 业务方法抽象

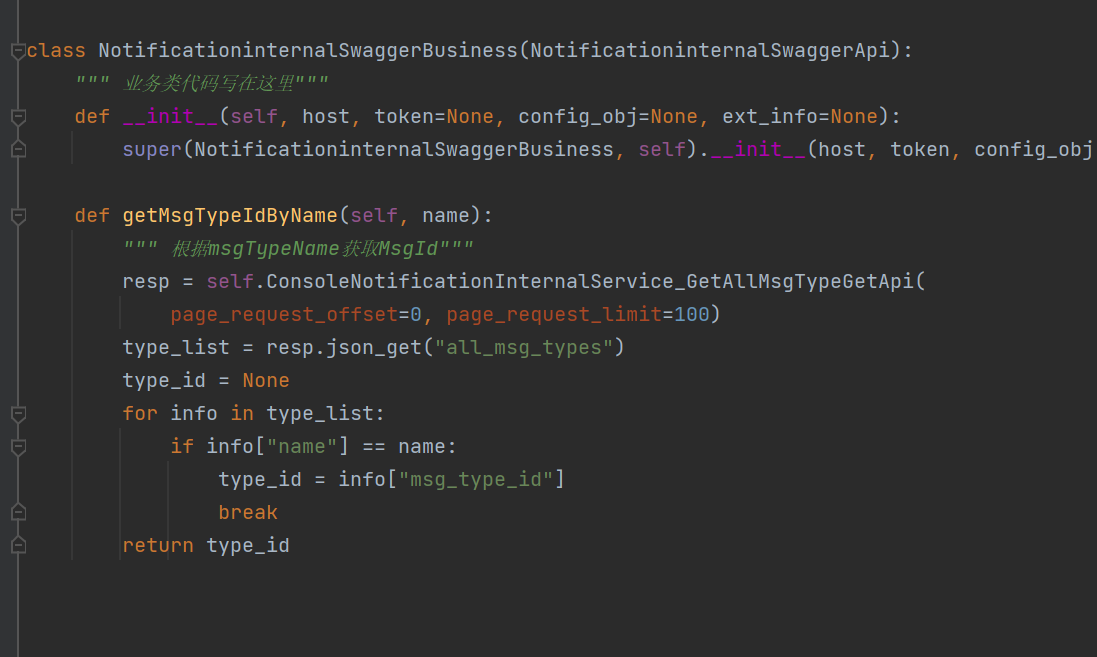
* ~~通常来说，单个接口只是实现单一的功能~~
  + ~~比如创建，~~
  + ~~比如查询，~~
  + ~~比如更新，~~
  + ~~比如删除~~
* ~~往往为了实现一个简单的业务场景，需要调用多次接口，并且接口之间是有关联关系的~~
  + ~~比如创建后查询是否创建成功，~~
  + ~~比如更新后查询是否更新成功，~~
  + ~~比如删除后查询是否删除成功。~~
* ~~而为了测试更加复杂的且数量较多的业务场景，我们需要多次调用某个业务方法~~
  + ~~创建时，不关键的参数默认取值，比如随机一个name~~
  + ~~创建后，实现自动清理clearUp功能，保证环境的干净，即便测试中程序报错~~

以上就是业务方法抽象的心路历程，简单一句话来概括：**抽象业务流程为方法，方便后续测试的调用**

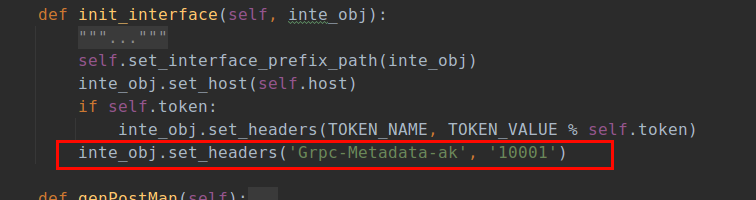
专门写业务方法抽象的文件为**apitest/defines/belt/xxxx\_service\_business.py**

举例：注册中心根据名称查询消息类型ID的封装：

*开发没有提供****根据名字查询消息类型ID****的接口，但我们的测试用例中需要反复使用，因此我们自己封装一个*



举例2: 统一修改接口的header:

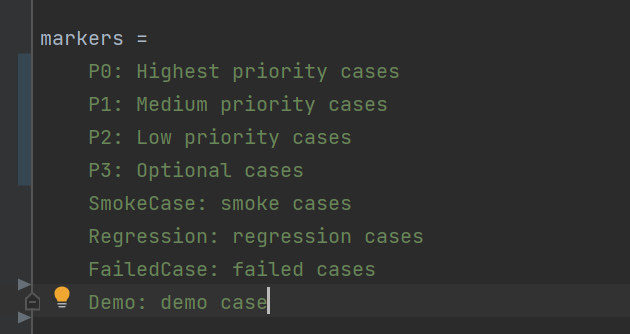


# 为测试用例打标签

打标签的方法：

在测试用例类或方法上添加装饰器： @pytest.mark.xxx

我们预定义的标签在apitest/pytest.ini 文件中， 如下：



执行的方法：

pytest -m "P0 and Regression"

定义一些常用的组合

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 用途 | 代码特点 | 标记 |  |
| 主流程 | 1. p0级功能点 2. 主要流程 3. 迭代回归Case 4. 线上回归Case | 1. 代码自动化 2. 仅依赖toml配置文件 3. 主要出现在场景测试用例中 | P0 + Regression |  |
| 手工测试 | 1. 手工测试用例 | 1. 不支持回归 2. 方便与开发配合测试 3. 方法模块测试使用 | skip |  |
| 其他组合 | 对于其他组合没有要求可以自行定义 |  |  |  |

参考文档：<https://www.sohu.com/a/428939975_216613>