**10.2异常处理·业务错误码使用样例**

上文已经定义了业务错误码的基类，并定义了一些全局的错误码。本文会继续上文的内容，新增具体的业务错误码样例。这里以用户服务为例。

# 旧的业务处理方式

可参考user\_service.py的login方法

def login(self, user\_name, password):  
 """  
 用户密码登录  
 :param user\_name:  
 :param password:  
 :return:  
 """  
 u = self.db.session.query(User).filter(User.user\_name == user\_name).first()  
 if u is None:  
 raise Exception("用户名或密码错误")  
 if u.password != password:  
 raise Exception("用户名或密码错误")  
 res = u.to\_dict(camel=True)  
 del res['password']  
 del res['createTime']  
 del res['updateTime']  
 del res['isDeleted']  
 return res

这里没有使用自定义异常，直接使用了Exception，当用户名密码错误时，报错如下：

{  
 "code": 99999999,  
 "msg": "服务器异常"  
}

没有具体的错误码和错误描述信息，当然我们也可以像之前一样，在全局异常处理那里把描述信息输出，但是这样依然不够优雅，不方便前端定制状态码与消息描述的关系。这里采用了新的方式。

# 新的业务处理方式

## 新增业务错误码枚举类

### services/enums/\_\_init\_\_.py

### services/enums/user\_error\_enum.py

新增的业务错误码基类需要继承错误码基类

from mexception import ErrorEnum  
  
  
class UserErrorEnum(ErrorEnum):  
 """  
 用户模块错误码  
 """  
 U80009001 = (80009001, "用户不存在")  
 U80009002 = (80009002, "用户名或者密码错误")

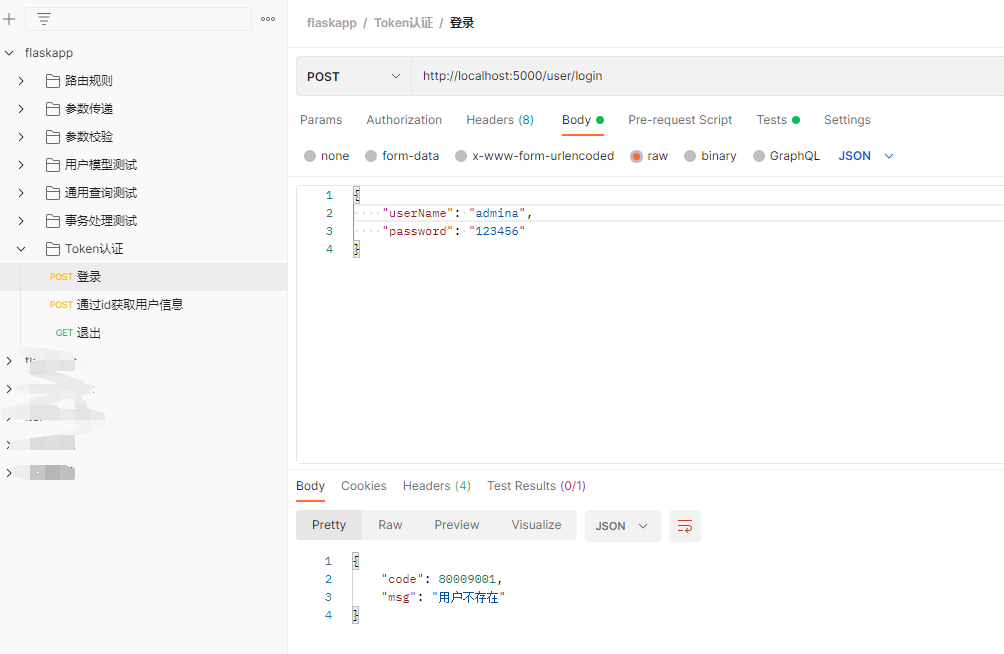
## 登录代码修改

修改services/user\_service.py

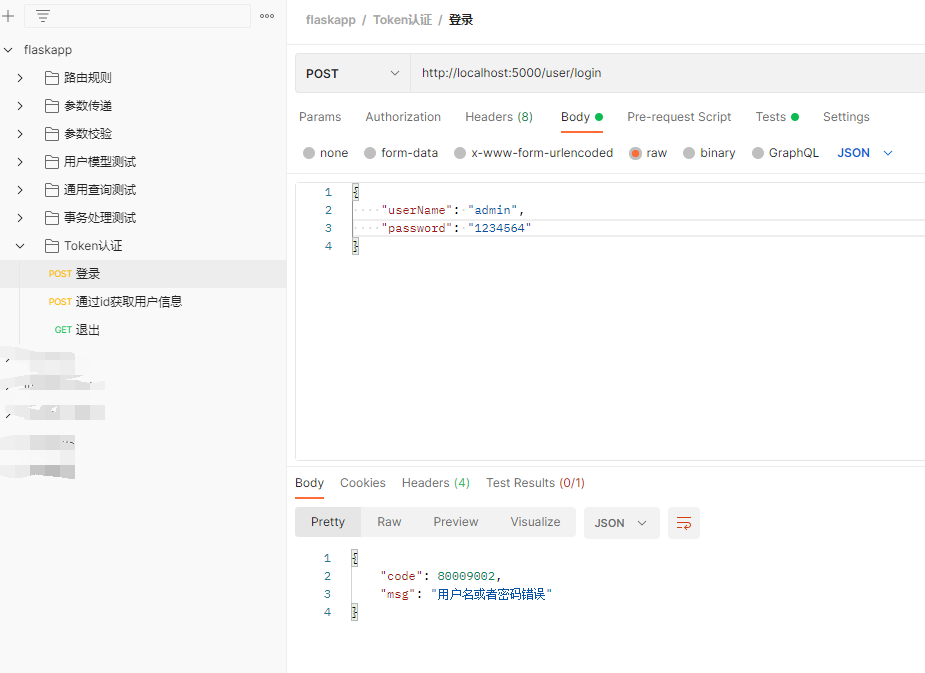
def login(self, user\_name, password):  
 """  
 用户密码登录  
 :param user\_name:  
 :param password:  
 :return:  
 """  
 u = self.db.session.query(User).filter(User.user\_name == user\_name).first()  
 if u is None:  
 AssertTool.raise\_biz(UserErrorEnum.U80009001)  
 if u.password != password:  
 AssertTool.raise\_biz(UserErrorEnum.U80009002)  
 res = u.to\_dict(camel=True)  
 del res['password']  
 del res['createTime']  
 del res['updateTime']  
 del res['isDeleted']  
 return res

# 使用Postman接口测试工具访问

## 访问/user/login登录-用户名不存在时



## 访问/user/login登录-密码错误时



# 一些说明

● 业务错误码枚举可以与service一对一对应

1. UserErrorEnum

2. RoleErrorEnum

● 业务错误码也可以与业务模块相对应

1. SysErrorEnum

2. CmsErrorEnum

3. OmsErrorEnum

● 业务错误码共8位长度，大家可以根据业务需要，指定错误码区间

● 当然，如果觉得业务错误码太长，可以缩减到3位或者4位