

退出登录就是删除用户相关的cookie信息（有session也需要删除session），达到未登录状态

The screenshot shows the PyCharm IDE interface. The left sidebar displays a project structure with a tree view of files and folders. A red arrow points from the 'User.py' file in the tree view to its corresponding tab in the top bar. Another red arrow points from the 'User.py' tab in the top bar down to the code editor area. The code editor contains Python code for a 'Logout' view. A yellow box highlights the URL routing section at the bottom of the code. The status bar at the bottom of the IDE shows various tabs like TODO, Problems, Version Control, Python Packages, Python Console, Terminal, Endpoints, and Event Log.

```
class Logout(views.MethodView):
    def get(self):
        """
        退出登录
        退出登录后将会回到登录页
        :return:
        """
        # 1.构建登录也响应对象
        # from common.lib.UrlManager import UrlManager
        response = make_response(redirect(UrlManager.buildUrl("/user/login"))) # 重定向到登录页
        # 2.删除 AUTH_COOKIE_NAME cookie
        response.delete_cookie(app.config.get("AUTH_COOKIE_NAME")) # 删除 cookie
        # 3.响应
        return response

# 使用add_url_rule方法来构造与视图关联的请求url
# 继承自模板的as_view方法可以将类转换为可以为路由注册的视图函数
# as_view('login') --- 传入的是站点名(标准操作是视图名小写 login -- login)
#         <Rule '/user/login' (GET, HEAD, OPTIONS) -> user_page.login>] user_page.login
route_user.add_url_rule('/logout', view_func=Logout.as_view('logout'))
```

```
# -*- encoding: utf-8 -*-
"""
File       : User.py
teaching   :
    用户模块 -- 登录，退出登录
"""

from flask import Blueprint, render_template, views, request, jsonify, make_response, url_for, redirect
from common.lib.UrlManager import UrlManager
from web.models.User import User # 模型类导出
from common.lib.UserService import UserService
from application import app
import json

# 构建蓝图对象
route_user = Blueprint('user_page', __name__)

# 登录视图
# 继承MethodView类可以重载，get/post/delete/put的请求方法。来实现针对某一类请求的视图接收。
class Login(views.MethodView):
    def get(self):
        return render_template("user/login.html")

    def post(self):
        # 1.注意导入请求上下文 request from flask import request
        # 2.获取form表单数据
        data = request.values
```

```
# 3.获取用户名及密码
login_name = data.get('login_name')
login_pwd = data.get('login_pwd')

# 4.构建响应数据对象
resp = {
    "code": 200, # 状态码
    "msg": "登录成功", # 响应信息
    "data": {} # 响应数据
}

# 5.判断数据是否获取及其是否满足要求
if (not login_name) or len(login_name) < 1:
    # 用户名没有数据 或者 用户名长度小于1
    resp["code"] = -1
    resp["msg"] = "请输入正确的登录用户名"
    # 导入 jsonify --- from flask import jsonify
    return jsonify(resp)

if (not login_pwd) or len(login_pwd) < 1:
    # 密码没有数据 或者 密码长度小于1
    resp["code"] = -1
    resp["msg"] = "请输入正确的密码"
    return jsonify(resp) # 响应json数据

# 6.根据用户名查询用户是否存在与数据库
user_info = User.query.filter_by(login_name=login_name).first()
if not user_info: # 数据库查不到 -- 即不存在
    resp["code"] = -1
    resp["msg"] = "请输入正确的用户名或密码"
    return jsonify(resp) # 响应json数据

# 7.用户存在 -- 加密密码
salt = user_info.login_salt # 根据user_info对象数据 获取数据库中的用户密码 加密盐
# 加密方法 from common.lib.UserService import UserService
gen_pwd = UserService.genPwd(login_pwd, salt) # 跟注册时相同的加密机制 -- 获得加密数据

# 8.将秘密加密数据 与 数据库密码数据进行比对

if user_info.login_pwd != gen_pwd:
    # 不相同，说明密码或者用户名不对
    resp["code"] = -1
    resp["msg"] = "请输入正确的用户名或密码"
    return jsonify(resp)

# 9.登录成功 重定向首页 from flask import url_for, redirect
# return redirect(url_for('index_page.index'))

# return jsonify(resp)

# 9.构建响应对象 --- 此处根据make_response获得的响应对象
# 注意导包
# from flask import make_response
# import json
res = make_response(json.dumps(resp))

# 10.构建用户对象加密数据
# 检查是否导入UserService --- from common.lib.UserService import UserService
```

```

cokie_str = UserService.genAuthCode(user_info) + "#" + str(user_info.uid)

# 11.设置cookie
# from application import app
# app.config.get("AUTH_COOKIE_NAME") 获取配置中配置的通用AUTH_COOKIE_NAME cookie键名
res.set_cookie(app.config.get("AUTH_COOKIE_NAME"), cokie_str)

# 响应
return res


class Logout(views.MethodView):
    def get(self):
        """
        退出登录
        退出登录后将会回到登录页
        :return:
        """

        # 1.构建登录也响应对象
        # from common.lib.UrlManager import UrlManager
        response = make_response(redirect(UrlManager.buildUrl("/user/login"))) # 重定向
        # 2.删除 AUTH_COOKIE_NAME cookie
        response.delete_cookie(app.config.get("AUTH_COOKIE_NAME")) # 删除cookie
        # 3.响应
        return response

# 使用add_url_rule方法来构造与视图关联的请求url
# 继承自模板的as_view方法可以将类转换为可以为路由注册的视图函数
# as_view('login') --- 传入的是站点名 (标准操作是视图名小写 Login -- login)
#           <Rule '/user/login' (GET, HEAD, OPTIONS) -> user_page.login>] user_page.login中的login就是
# as_view指明的站点名
route_user.add_url_rule('/login', view_func=Login.as_view('login'))
route_user.add_url_rule('/logout', view_func=Logout.as_view('logout'))

```

测试

1.启动项目，访问首页（未登录进行登录访问）

<http://127.0.0.1:8888/>

2.点击右上角头像位置，退出登录操作如下图



欢迎使用编程浪子订餐管理后台

姓名: xxxx 编辑
手机号码: xxxx

营收概况 白统计 1005.00 近30日: 31177.00

订单 白统计 988 近30日: 29383

会员 白统计 358 今日新增: 77 近30日新增: 2454
近30日: 45980

使用highchart画图

使用highchart画图

```
javascript:void(0);
```