如何用 Python 在京东上抢口罩

原创 極光 Python技术 2月25日

全国抗"疫"这么久终于见到曙光，在家待了将近一个月，现在终于可以去上班了，可是却发现出门必备的口罩却一直买不到。最近看到京东上每天都会有口罩的秒杀活动，试了几次却怎么也抢不到，到了抢购的时间，浏览器的页面根本就刷新不出来，等刷出来秒杀也结束了。现在每天只放出一万个，却有几百万人在抢，很想知道别人是怎么抢到的，于是就在网上找了大神公开出来的抢购代码。看了下代码并不复杂，现在我们就报着学习的态度一起看看。

使用模块

首先打开项目中 `requirements.txt` 文件，看下它都需要哪些模块：

- requests：类似 `urllib`，主要用于向网站发送 HTTP 请求。
- beautifulsoup4：`HTML` 解析器，用于将 `HTML` 文档转换成一个复杂的树形结构。
- pillow：Python 图像处理标准库，用于识别验证码。

配置文件

一般项目中我们都需要把一些可配置的内容放到配置文件中，现在我们来看下这里主要配置项：

```
# 邮寄地所属地区ID
area = 123456

# 这是配置的商品的ID
skuid = 6828101

# 打码服务器的地址
captchaUrl = http://xxx/pic

# 通知邮箱
mail = xxxxxx@qq.com

# cookie的设置
cookies_String = shshshfpa21jsda8923892949204923123
```

OK，有了配置文件，那我们就得有一段读取配置文件的代码，这段代码实现将配置内容加载到内存中。

```
import os
import configparser

# 加载配置文件
class Config(object):
    def __init__(self, config_file='configDemo.ini'):
        self._path = os.path.join(os.getcwd(), config_file)
        if not os.path.exists(self._path):
            raise FileNotFoundError("No such file: config.ini")
```

```
        self._config = configparser.ConfigParser()
        self._config.read(self._path, encoding='utf-8-sig')
        self._configRaw = configparser.RawConfigParser()
        self._configRaw.read(self._path, encoding='utf-8-sig')

    def get(self, section, name):
        return self._config.get(section, name)

    def getRaw(self, section, name):
        return self._configRaw.get(section, name)
```

主程序模块

我看 `GitHub` 上也有实现了运行程序后通过京东 App 扫码登陆，然后再通过登陆 `Cookie` 访问网站的，不过这里并没有使用这种方式，毕竟我们打开浏览器开发者工具也能很容易获取到登陆的 `Cookie`，这里就是将 `Cookie` 直接放到配置文件里的方式。

```
# 主程序入口
# 检查是否存在要抢购的端口，然后进入循环扫描
if len(skuids) != 1:
    logger.info('请准备一件商品')
skuId = skuids[0]
flag = 1

# 循环扫描该商品是否有货，有库存即会自动下单，无库存则休眠后继续扫描
while (1):
    try:
        # 初始化校验
        if flag == 1:
            logger.info('当前是V3版本')
            validate_cookies()   # 校验登陆状态
            getUsername()        # 获取登陆用户信息
            select_all_cart_item()   # 全选购物车
            remove_item()           # 删除购物车
            add_item_to_cart(skuId)   # 增加抢购的商品
        # 检测配置文件修改
        if int(time.time()) - configTime >= 60:
            check_Config()
        logger.info('第' + str(flag) + '次 ')
        # 计数器
        flag += 1
        # 检查库存模块
        inStockSkuid = check_stock(checksession, skuids, area)
        # 自动下单模块
        V3AutoBuy(inStockSkuid)
        # 休眠模块
        timesleep = random.randint(1, 3) / 10
        time.sleep(timesleep)
        # 校验是否还在登录模块
        if flag % 100 == 0:
```

```
            if flag % 100 == 0:
                V3check(skuId)
        except Exception as e:
            print(traceback.format_exc())
            time.sleep(10)
```

以上就是该项目主程序，我已经将代码在原来基础上增加了些注释，可以让我们更容易明白代码的含义。下面我们就选择几个比较关键的代码分析一下。

登陆状态校验

```
# 校验登陆状态
def validate_cookies():
    for flag in range(1, 3):
        try:
            targetURL = 'https://order.jd.com/center/list.action'
            payload = {
                'rid': str(int(time.time() * 1000)),
            }
            resp = session.get(url=targetURL, params=payload, allow_redirects=False)
            if resp.status_code == requests.codes.OK:
                logger.info('登录成功')
                return True
            else:
                logger.info('第【%s】次请重新获取cookie', flag)
                time.sleep(5)
                continue
        except Exception as e:
            logger.info('第【%s】次请重新获取cookie', flag)
            time.sleep(5)
            continue
    message.sendAny('脚本登录cookie失效了，请重新登录')
    sys.exit(1)
```

以上代码是每次调用时，循环两次获取通过 `session` 获取当前登陆状态，如果两次后依然失败则退出程序。

添加商品到购物车

接下来我们再看下如果添加商品到购物车的，代码如下：

```
def add_item_to_cart(sku_id):
    # 请求添加商品url
    url = 'https://cart.jd.com/gate.action'
    payload = {
        'pid': sku_id,
        'pcount': 1,
        'ptype': 1,
    }
    # 返回结果
```

```
    # 返回结果
    resp = session.get(url=url, params=payload)
    # 套装商品加入购物车后直接跳转到购物车页面
    if 'https://cart.jd.com/cart.action' in resp.url:
        result = True
    else:
     # 普通商品成功加入购物车后会跳转到提示 "商品已成功加入购物车！" 页面
        soup = BeautifulSoup(resp.text, "html.parser")
        result = bool(soup.select('h3.ftx-02'))  # [<h3 class="ftx-02">商品已成功加入购物车！</h3>]

    if result:
        logger.info('%s  已成功加入购物车', sku_id)
    else:
        logger.error('%s 添加到购物车失败', sku_id)
```

在这里，只是简单几行代码就能将端口添加到购物车了，而且这里还区分了不同类型商品添加到购物车返回的页面结果是不同的，所以要进行区别处理。

购买商品

将商品添加到购物车了，接下来我们就得提交结算页了，也就是将商品提交到付款页面，这段代码有点多，我简化了下并加了些注释：

```
def submit_order(session, risk_control, sku_id, skuids, submit_Time, encryptClientInfo, is_Submit_ca
                 submit_captcha_text, submit_captcha_rid):
    # 提交端口的url
    url = 'https://trade.jd.com/shopping/order/submitOrder.action'

    # 提交参数
    data = {
        'overseaPurchaseCookies': '',
        'vendorRemarks': '[]',
        'submitOrderParam.sopNotPutInvoice': 'false',
        'submitOrderParam.trackID': 'TestTrackId',
        'submitOrderParam.ignorePriceChange': '0',
        'submitOrderParam.btSupport': '0',
        'riskControl': risk_control,
        'submitOrderParam.isBestCoupon': 1,
        'submitOrderParam.jxj': 1,
        'submitOrderParam.trackId': '9643cbd55bbbe103eef18a213e069eb0',  # Todo: need to get trackId
        'submitOrderParam.needCheck': 1,
    }

    # 如果用到京豆会需要输入支付密码
    def encrypt_payment_pwd(payment_pwd):
        return ''.join(['u3' + x for x in payment_pwd])

    # 校验支付密码
    if len(payment_pwd) > 0:
        data['submitOrderParam.payPassword'] = encrypt_payment_pwd(payment_pwd)
```

```
    data['submitOrderParam.payPassword'] = encrypt_payment_pwd(payment_pwd)

    # 请求报文头
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gec
        "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q
        "Referer": "http://trade.jd.com/shopping/order/getOrderInfo.action",
        "Connection": "keep-alive",
        'Host': 'trade.jd.com',
    }

    # 订单提交会尝试两次
    for count in range(1, 3):
        logger.info('第[%s/%s]次尝试提交订单', count, 3)
        try:
            # 可能会存在的校验码
            if is_Submit_captcha:
                captcha_result = page_detail_captcha(session, encryptClientInfo)
                # 验证码服务错误
                if not captcha_result:
                    logger.error('验证码服务异常')
                    continue
                data['submitOrderParam.checkcodeTxt'] = submit_captcha_text
                data['submitOrderParam.checkCodeRid'] = submit_captcha_rid
            # 提交订单
            resp = session.post(url=url, data=data, headers=headers)
            resp_json = json.loads(resp.text)
            logger.info('本次提交订单耗时[%s]毫秒', str(int(time.time() * 1000) - submit_Time))
            # 判断是否提交成功
            if resp_json.get('success'):
                logger.info('订单提交成功! 订单号：%s', resp_json.get('orderId'))
                return True
            else:
                # 提交失败返回的多种原因
                resultMessage, result_code = resp_json.get('message'), resp_json.get('resultCode')
                if result_code == 0:
                    # self._save_invoice()
                    if '验证码不正确' in resultMessage:
                        resultMessage = resultMessage + '(验证码错误)'
                        logger.info('提交订单验证码[错误]')
                        continue
                    else:
                        resultMessage = resultMessage + '(下单商品可能为第三方商品，将切换为普通发票进行尝
                elif result_code == 60077:
                    resultMessage = resultMessage + '(可能是购物车为空 或 未勾选购物车中商品)'
                elif result_code == 60123:
                    resultMessage = resultMessage + '(需要在payment_pwd参数配置支付密码)'
                elif result_code == 60070:
                    resultMessage = resultMessage + '(省份不支持销售)'
                    skuids.remove(sku_id)
                    logger.info('[%s]类型口罩不支持销售', sku_id)
```

```
                logger.info('订单提交失败，错误码：%s，返回信息：%s', result_code, resultMessage)
                logger.info(resp_json)
                return False
        except Exception as e:
            print(traceback.format_exc())
            continue

```

以上代码实现了商品自动提交到结算页面，这段明显比添加购物车要复杂，果然跟钱有关的都不简单。好了，到了结算页面剩下就是付款了，这个就不需要再抢了，毕竟也没人会抢着给你付钱的。

总结

本文为大家介绍了一个京东抢购的小工具，它实现了扫描是否有库存，发现有库存就自动下单，并且可以自动提交到结算页面。而它所实现方式也并不算太复杂，进一步分析了它的部分代码，有兴趣的小伙伴可以去文末 GitHub 项目网址上了解更多，再次感谢开发者的付出和分享。

参考

GitHub项目网址：https://github.com/cycz/jdBuyMask