crifan的折腾精神、学习能力和逻辑能力的体现

- 最后更新：`20190525`

说明

- 内容说明
  - 此文档专门整理出一些相关内容，以体现出crifan的`折腾精神`，`学习能力`和`逻辑能力`
- 本文目的
  - 证明自己的特长
    - 逻辑性比较强
    - 有足够技术敏感度
    - 有很强的折腾精神
    - 善于总结
  - 从而才能
    - 能够快速学习新的技术
    - 善于和能够解决复杂的技术问题
    - 找到问题的根本原因
      - 追根溯源
    - 能想办法提高做事情的效率
      - 详见：
        - 如何提高工作效率
    - 总结出
      - 技术学习的思路和方法的经验与总结
      - 和其他各种技术和非技术的教程
        - Crifan的电子书的使用说明

- 内容历史
  - 之前最早是发布在crifan的折腾精神 – 在路上
  - 现在合并整理到此文档
    - 加上其他部分，如学习能力、逻辑能力等
- 发布形式
  - 代码仓库：
    - https://github.com/crifan/crifan_play_learn_logic_spirit.git
      - 源码是：`README.md`
  - HTML：从`README.md`中生成（的`README.html`，并改名为index.html）
    - 文件：https://github.com/crifan/crifan.github.io/blob/master/crifan_play_learn_logic_spirit/index.html
    - 在线查看页面：https://crifan.github.io/crifan_play_learn_logic_spirit/
      - 会自动跳转到：https://crifan.github.io/crifan_play_learn_logic_spirit/index.html
  - PDF：从`README.md`中生成（的`README.pdf`）
    - 文件：https://github.com/crifan/crifan_play_learn_logic_spirit/blob/master/README.pdf
    - 下载：https://github.com/crifan/crifan_play_learn_logic_spirit/raw/master/README.pdf

crifan的折腾精神==解决复杂问题的能力

下面整理出crifan的折腾各种复杂问题的过程。

其中部分内容额外体现了需要一定的技术敏感度，才容易从发现问题细

节，找到问题根源，最终解决问题。

CentOS7中安装Python的pycurl和PySpider的pipenv的pycurl

期间各种折腾，遇到各种问题

- ModuleNotFoundError No module named pycurl
- ImportError pycurl libcurl link-time version (7.29.0) is older than compile-time version (7.64.1)
- ImportError pycurl libcurl link-time ssl backend (nss) is different from compile-time ssl backend (openssl)
    - 且此同一种问题反复出现多次

最后才发现，其实涉及到了多个环境：

- CentOS中 `Python2`
- CentOS中 `Python3`
- CentOS中 `PySpider` 中 `pipenv`

最终解决问题的关键点之一是：

- 思路不要僵化
    - 通过仔细和思考或发现，当把curl变成openssl失败，行不通后
    - 转而考虑保留 `curl` 为 `nss` ，让 `pycurl` 配合弄成 `nss`
        - 之后即可通过源码或 `pip` 安装出，版本匹配的， `nss` 的 `pycurl`
            - 最终 `python` 和 `PySpider` 的 `pipenv` 都可以 `import pycurl` 了

最后不仅解决了问题，还整理出心得了：

- 【整理】CentOS7中安装pycurl的心得

期间的过程详见：

- 【未解决】CentOS7中安装nss的PyCurl
- 【未解决】CentOS7中Pyspider运行出错：ModuleNotFoundError No module named pycurl
- 【未解决】CentOS7中卸载自带nss的curl并升级更换为openssl的curl
- 【已解决】CentOS7中PySpider运行出错：ImportError pycurl libcurl link-time ssl backend (nss) is different from * compile-time ssl backend (openssl)
- 【未解决】CentOS7中尝试通过更换so库把默认nss的curl更换为openssl的curl
- 【已解决】CentOS7中通过源码重新编译和安装openssl版本的curl
- 【已解决】CentOS7中旧版本backend是nss的curl的libcurl的库在哪里
- 【已解决】CentOS7中已经安装的openssl的lib路径是什么
- 【未解决】用远程阿里云ECS中CentOS服务器中运行PySpider批量下载数据
- 【已解决】CentOS7中PySpider运行出错：ImportError pycurl libcurl link-time version (7.29.0) is older than * compile-time version (7.64.1)
- 【已解决】CentOS7中pipenv去安装PySpider期间pycurl出错：__main__.ConfigurationError Could not run curl-config * Errno 2 No such file or directory
- 【已解决】CentOS7中用安装Python的curl出错：src/pycurl.h fatal error Python.h No such file or directory

PySpider中模拟访问小花生接口其他参数都正确但始终是500 Internal Server Error

- 折腾期间的关键点
  - 技术敏感度=细心
    - 能发现`PySPider`的调试界面中的data的参数和postman中

json参数不同
        - `PySPider` 中：`J=%7B%22userId%22%3A%22...`
        - `Postman` 中：`{"J":"{\"userId\":\"1134723\",...`
    - 具备对应的基础知识
        - 能从 `J=%7B%22userId%22%3A%22...` 之类的数据中推断和猜测出：dict字典的json被额外encode了，是encoding编码后的字符串
            - 才能想到去找不
              让 `PySpider` 中 `self.crawl` 的 `POST` 的 `data` 不被 `encode` 编码
                - 最终解决了问题，获取到希望的数据
        - -》具体涉及到了哪些知识
            - 主流数据格式：JSON
            - HTTP知识总结
            - 字符编码详解

详见：

- 【已解决】PySpider模拟请求小花生api接口出错：requests.exceptions.HTTPError HTTP 500 Internal Server Error

小花生的app的破解

- 最关键的前提：
    - v3.4.8
        - 没有被加固到
            - 否则dex导出了jar时（估计）就会报错
                - 不会这么轻易的导出可用的dex
        - 没有被混淆
            - 否则即使jar导出了源码，也无法看清原始代码中的加密逻辑
- 自己折腾过程中：

- 关键点：
  - 在最新版v3.6.9用FDex2导出（200多B的无效的）dex无果后
    - 能想到去试试其他的旧版本
  - 以及在试试旧版本期间
    - v1.5虽然可以导出dex（dex导出jar，jar导出源码）
      - 主要是其中代码都是错误bad dex opcode
        - 无法找到源码
      - 其次是代码被混淆了
        - 即使找到，也不容易看清楚源码逻辑
- 中等难度的地方
  - 如何搞清楚apktool和dex2jar、jd-gui等之间的关系
    - 搞清楚如何利用导出的文件，后续用什么工具，如何去处理
  - 如何正确的使用各种工具
    - root了的安卓 + XPosed
      - 用的是之前破解安卓app中https的ssl证书而搭建的环境：
        - 夜神安卓模拟器
        - XPosed
    - 再次基础上再去安装和使用工具
      - FDex2
        - 才能继续导出dex文件
      - 才能继续用夜神中文件管理器导出文件
        - 自己要解决夜神和mac的共享目录的问题
          - 【已解决】夜神安卓模拟器中导出文件到mac电脑
          - 【已解决】Nox夜神安卓模拟器中/mnt/shared对应Mac的共享目录在哪里
    - 最后才是用工具查看jar包，导出源码
      - 用jd-gui导出源码
        - 【已解决】mac版JD-GUI查看并导出jar包的

java源代码
        - 也顺带去试了其他工具，比如：
            - jadx
            - Procyon：命令行工具
                - Luyten：基于Procyon的GUI工具
                    - 【已解决】用基于Procyon的Luyten反编译安卓jar包得到java源码
    - 最终从v3.4.8的hook出的dex，dex转jar，jar导出源码，找到了J字段的解密逻辑

MongoDB无法连接

期间，要有足够的技术敏感度，才能及时想到可能的原因，然后才能证实和快速解决。

详见：

- 【已解决】公司Wi-Fi更换运营商导致IP变化导致远程Mongo连不上

wordpress主页菜单加指示条

折腾期间，能想到利用：

网址是wordpress，然后再去搜wordpress中是否有和当前页面方面的标示，还真的巧了找到了current-menu-item

之后，才能通过css去控制current-menu-item，达到要显示的效果。

详见：

- 【已解决】给wordpress顶部主菜单底部加上指示条表示当前所处页面

enfold-child子主题中手机端顶部菜单点击显示异常

开始时最直接的反应，以为是以为缺少什么css呢，所以就去对比css，一点点的找，到底是哪些css不同而导致的异常

后来对比调试+细心发现，加上了is-active后，菜单可正常显示，说明不是缺少css

（重点：如果不是细心发现其实只是加上is-active即可，不知道后续还要在错误道路上，继续调试css多久）

而最开始想要调试，也没法调试，是无意间搜到网上帖子，得知是Enfold的avia-merged-styles-f39bxxxx773.css这种是合并后的

所以想到了，是不是可以有合并的参数设置，后来果然找到了

然后取消合并后，得到分别的独立的css(以及js)

从而后续可以单独看到js源码调试了

（重点：如果不是找到取消合并，则后续无法准确调试js到底执行了什么）

后来以为avia.js中的burger_wrap.click的代码执行有误呢，然后经过添加log日志，最终确定代码没问题

期间看到了加上了is-active，但是后来又没了

以为是jquery的AddClass失效了呢

而期间调试了N多次，始终有问题。

后来是第二天无意间重启了Mac的web server即mamp后，本地代码好像正常工作了

（重点：如果不是重启mamp，还不知道要继续浪费多少时间）

才调试发现

```
burger.addClass(“is-active”);
```

是正常执行的，是的确添加了is-active的class

而执行了后面的：

```
htmlEL.addClass(“av-burger-overlay-active”);
```

却导致菜单不正常显示的

就以为是：html的class中加了av-burger-overlay-active导致其他什么css生效，导致不正常显示呢

后来发现这个是正常现象

后来继续对比调试。以为是：

```
burger_wrap.click的 e.preventDefault();
```

没有执行到，导致burger_wrap.click被执行了2次

后来发现不是，而是通过Chrome调试期间，细心的注意到了：

前后的两个avia.js是enfodl父主题和子主题enfold-child两个独立的文件的相同函数

（重点：如果不是注意到是两个不同文件的avis.js中的burger_wrap.click，则解决问题的方向就偏了，还不知道要继续花多少时间才能回到正确方向上）

不是同一个avia.js中的两次执行相同的函数

从而确定是由于先后两次加载了都带burger_wrap.click的avai.js，而导致

burger_wrap.click被执行了2次

最终经过Beyond Compare对比发现，enfold-child本身配置是相同的，而新旧两个Enfold主题，是版本不同，所以问题还是出在enfold主题。

然后自己通过间接的注释掉enfold-child的avia.js，才规避问题。

具体过程详见：

- 【已解决】WordPress的网站Enfold主题在手机端顶部菜单异常
- 【已搞懂】WordPress中enfold-child主题中为何avia.js的burger_wrap.click执行了2次
- 【已解决】搞懂Enfold中burger_wrap.click时什么原因导致正常显示的菜单又消失异常
- 【已解决】WordPress的Enfold主题中合并后的css和js文件是如何生成的
- 【已解决】WordPress主题Enfold中如何拆分之前合并了的css和js文件
- 【已解决】确认是否是缺少css导致手机端WordPress主题Enfold的主菜单显示异常
- 【已解决】确认是否是js没有正确运行导致手机端WordPress主题Enfold的主菜单显示异常

Azure的token出错：Out of call volume quota

如果只是从问题的表面现象，很难想到根本原因。

幸好是从繁杂的信息中，找到了一个帖子，有个提示。

经过尝试最终发现是这个原因：

> 微软Azure，打着鼓励你用免费F0套餐，且免费的额度很多很多，但是实际上你使用了一点点后，就不给你继续使用，就告诉你超额了。然后你只能升级换成收费的套餐，才能正常继续使用。

详见：

- 【已解决】调用微软Azure的cognitive的sts/tts的api生成token时出错：Out of call volume quota. Quota will be replenished in

小程序页面空白出错：SyntaxError Unexpected EOF

关键点：

即使知道原因是：MongoDB中某些text中有特殊字符，导致显示小程序json解析出错，导致页面无法显示的问题

但是如果不懂这个是不可见的控制字符，以及如何去除，以及应该去掉哪些，那也是没法彻底的（去写代码，批量）解决问题的。

详见：

- 【已解决】测评系统小程序出错：SyntaxError Unexpected EOF 0/page-frame.html

Netgear R6220路由器 变砖尝试修复的过程

虽然最后没有把变砖的路由器救活，但是期间能够从网上繁杂的信息中，找到真正的串口的位置，以及最终买电烙铁和找到并买到合适的ttl的线，也算是不容易了。

详见：

- 【未解决】尝试通过接串口和重新刷机去修复变砖的Netgear R6220路由器

nginx的https的ssl证书无效，https域名的网页地址打不开

nginx中配置了https的ssl证书，结果始终不起效果，打开https的地址 https://www.naturling.com/ 始终出现：

> 无法访问此网站，拒绝了我们的请求。
> 请尝试以下办法：
> 检查网络连接
> 检查代理服务器和防火墙

之类的错误

-》经过一点点问题的排除，包括但不限于：

cert和key的文件访问权限：从root改为nginx的www用户和组

ssl的各种参数配置，包括listen 80和listen，server_name，ssl_ciphers等等等等

-》最终发现：

nginx在listen 443同时如果加入了80，则http页面是可以打开的，有access的log的

-》但是https的访问，始终没有log

-》好像是https的请求，根本都没进入nginx

-》所以才怀疑是不是端口问题

-》但是阿里云的ECS的安全组中，已确保了添加了443端口了

（本身新建ECS时勾选了默认系统建了安全组就包括优先级110的443，担心有影响，又自己新建一个更高的优先级1的443的规则，且删除了系统的443规则）

但是还是不行。

-》最终是：（去CentOS中用firewalld去）添加防火墙规则，允许https的443端口入方向被访问
才使得https地址 https://www.naturling.com/ 能正常打开。

详见：

- 【已解决】小程序中如何让api服务器满足要求：已备案的带域名的https
- 【已解决】给阿里云的带域名的服务器加https
- 【已解决】使用已购买的阿里云免费SSL证书即去服务器中配置nginx的https证书
- 【已解决】nginx中配置了https的ssl证书后不起效果
- 【已解决】CentOS 7中如何通过iptables添加https的443端口
- 【已解决】CentOS 7中如何通过firewalld去添加https的443端口

相关：

- 【整理】https证书 SSL证书基本知识
- 【已解决】购买阿里云首年免费的https证书：Symantec免费型DV SSL证书
- 【已解决】nginx中如何强制所有的80的http都强制转发到443的https

Charles抓包https的过程

- 先是小坑：用有线网络 解决app无法上网
  - 也是看到别人帖子，但是不容易找到这样的帖子，因为网上很少提到
  - 去试了试，发现才有用的
- 最终是：无意间发现 单独设置ssl的过滤网址 才能工作
  - 也是参考别人帖子的尝试后 无意间发现的
    - 归根到底，感觉应该算是Charles的bug了，*:*按照道理应该工作才对
- 期间是：几个大大小小大坑，都分别靠自己的自信和整理网上大量的

资料，最终解决掉了，比如：

- 虽然提示证书安装成功，但是实际上没有安装进去
  - 先是自己仔细，去受信任凭据中没有找到
  - 后来是参考别的帖子，而确定了，证书的确没有安装成功
- 自己特殊的锤子M1L无法root导致无法解决证书问题
- 搞清楚Android 7之后，无法抓包https的问题
  - 幸好之前弄过Android开发，否则不知道Android官网和别人所提及的AndroidManifest.xml，其实指的是你自己是app的开发者，有源码，才能干的事情
    - 而自己非APP的开发者，而是抓包者
  - 而且当时参考别人帖子，找到并使用工具去给已有apk加上支持https的抓包
    - 估计内部就是改动了xml中相关配置后重新打包
    - 但是当时还不懂，没搞清楚是什么意思
    - 最后是在整理
      - 【整理】Mac中用Charles抓包iOS或Android手机app中包括https的数据
    - 期间，才搞懂该工具是用来干啥的，以及使用的前提和场景：
      - 就是此处用Charles的，非app的开发者，而是抓包者，可以用这个工具

其中包括：

网上更多的人说安卓手机中安装Charles证书时，类型选择WLAN，结果被坑了，最后是换成少数人提到但是自己没试过的：VPN和应用，最后才正常安装证书，但是还不是安装到受信任凭据的系统中，而是用户中，以为没用，但是后来发现是有用的

-》规避了必须要root安卓手机的问题

-》也可以实现普通的https抓包解密未明文的效果了

最后把完整的操作步骤和中间遇到的大大小小的坑，都详细记录并整理到帖子里了，详见：

- 【整理】Mac中用Charles抓包iOS或Android手机app中包括https的数据

并且，后续又遇到：

部分https能抓包，但是其他特殊https无法抓包

期间也试了试其他路：找改安卓app的旧版本，希望万幸可以没有https的ssl pinning，最后失败

从：

Charles proxy fails on SSL Connect Method – Stack Overflow

以及其他一些帖子，基本上确定了此处无法破解的https是ssl pinning

而关于ssl pinning的办法，网上很多帖子，各种说法都很复杂，包括从apk逆向工程得到代码，再改动代码去破解的，所以放弃这些复杂的办法。

Android Security: SSL Pinning – Matthew Dolan – Medium

提到了之前Charles调试期间看到的，那个特殊的https是OkHttp，也知道旧版本貌似有bug

但是此处是最新的okhttp/3.10.0，没bug，所以也无法破解，也找不到其他相关的的办法。

后来终于找到一个相对解释的比较全的帖子，其中介绍了破解的办法：

Four Ways to Bypass Android SSL Verification and Certificate Pinning

但是却也没有给出有效且方便的办法。

而方便的办法，则是之前很多帖子中，断断续续提及的，包括这里也提到了：

如何对使用了ssl pinning的APP（如知乎）进行抓包？ – 知乎

以及Android Security: SSL Pinning – Matthew Dolan – Medium 然后才知道，对于破解ssl pinning的办法：

- Android：
  - iSECPartners/Android-SSL-TrustKiller: Bypass SSL certificate pinning for most applications
  - 或
  - Fuzion24/JustTrustMe: An xposed module that disables SSL certificate checking for the purposes of auditing an app with cert pinning

通过

- Charles Proxy now available on iOS | Hacker News
- one of the best tools for reverse engineering mobile apps. I’m just having probl… | Hacker News

知道的：

- iOS
  - nabla-c0d3/ssl-kill-switch2: Blackbox tool to disable SSL certificate validation – including certificate pinning – within iOS and OS X Apps
  - 或：
  - iSECPartners/ios-ssl-kill-switch: Blackbox tool to disable SSL certificate validation – including certificate pinning – within iOS Apps

但是需要去root手机才行

然后对于手上的手机想办法去root：

- 锤子M1L：最终确定官网就不支持root
- 红米5A：
  - 本来以为简单的下载个root工具，随便即可root。
  - 结果试了半天官网的解锁的办法，未果
  - 最终证明是：
    - 小米很垃圾的做法，限制解锁时间，要1一月后才能解锁，否则无法继续root
    - 暂时只能放弃
  - 注
    - 期间也试过 音量键减 + 电源键的FastBoot，和 音量键加 + 电源键的Mi-Recovery模式，都要先解锁才能继续root。
- 结果就是：手头的安卓手机都不支持root

那么实在不行，考虑去购买个，便宜点的，比如1000以内中低端手机，应该都可以root的

然后就去研究便宜的可以root的安卓手机

结果发现，通过网上很多个root工具的支持机型，再去找手机，都找不到，因为都是旧型号手机，现在京东和天猫等都买不到了

再去单独从京东或天猫中找最新出的，1000以内的手机，再去找每个手机是否方便root，结果却又发现原本以为的常见的品牌，包括小胡，华为，中兴，Oppo，Vivo等等手机，却要么是之前可以root，但是最新都不支持了，比如华为的，之前可以申请解锁现在不支持了，要么是太贵了，总之现在都不论便宜和贵的，都很难买到一个手机，确保能顺利root的。

所以放弃。

后来的后来，突然想起来：去淘宝买个二手的手机吧，结果无意间发现，有人卖这种二手老手机且帮忙弄好root的手机，所以就去买二手的小米4，

卖家帮忙先root好（其实自己也可以用工具去root，因为都是老的安卓系统，很多现有root工具都支持root的）

不过后来，突然想到：

貌似听某些人说，Charles的代理，也可以用安卓模拟器的

以及如何对使用了ssl pinning的APP（如知乎）进行抓包？ – 知乎又提到了安卓模拟器，所以才想到另外这条路：

找个安卓模拟器，这样应该就容易解决root的问题了

最后经过尝试，在Mac中好用的，支持Wifi网络设置Charles的代理的，支持root权限的安卓模拟器是夜神安卓模拟器，

其中还有个细节：夜神模拟器的Wifi直接点击也无法设置代理，无意间（也包括之前自己用过Android，巧了有过类似精力）长按Wifi，才找到Wifi代理设置的

之后的路，就相对不那么难了，但是还有点小小曲折：

正常去夜神模拟器中安卓Charles的证书，

正常去模拟器中通过apk安装安卓的app

模拟器中安装xposed框架，结果最开始安装的夜神应用中心（按理说，系统自带的应用市场，肯定是最匹配，且效果最好的），安装的是5.1.1版本，结果后来证明是不兼容，不支持此夜神模拟器的

后来的后来，即使从官网或别处下载到正确的4.4的版本，去安装，也还是有问题

最后是自己意识到，可能需要先卸载已有的版本再安装才可以?

试了下先卸载5.1.1.的Xposed，再安装支持4.4的Xposed，终于可以正常安装了。

最后的最后，终于可以绕开ssl pinning，实现特殊的https也可以抓包解密看到明文了。

详见：

- 【已解决】Charles无法抓包部分加了SSL Certificate Pinning的https包

注：后来又去整理出独立的教程了：

- app抓包利器：Charles

找出supervisor+gunicorn的gevent单worker的Flask的app中额外的2个进程是从哪里来的

虽然用gunicorn的gevent解决了Flask的app的单例问题，但是却发现另外还有2个线程，导致单例失效

而对于为何有这两个线程，其实开始是一点头绪是没有的。

而足够多的折腾精神和敏锐，让我找到了个思路：

可以从另外2个线程的log信息中，找到所对应的文件

这样就可以找到最开始打印log的文件

对于找到最终的线程的来源，应该会有帮助。

然后就找到了都是：

`common/FlaskLogSingleton.py`

```
log.info("LoggerSingleton inited, logSingleton=%s", logSingleton)
```

所打印出来的log：

```
[2018-08-30 13:28:35,272 INFO 26049 MainProcess 139969090553664 Mai
```

然后根据自己之前的代码，反推出，应该是别的模块中，调用了：

```
from common.FlaskLogSingleton import log
```

而触发上述的log的。

但是import log的地方也很多，并不容易找到是哪里的最开始引入的，以及也不容易因此就发现线程是如何创建的。

只是经验加上直觉，觉得最大的嫌疑是：

和Flask的app，感觉逻辑上属于并列的关系的celer

-》因为：

supervisor去管理和部署Flask的APP之外，还管理了celery：

```
[program:robotDemo_CeleryWorker]
command=/root/.local/share/virtualenvs/robotDemo-dwdcgdaG/bin/celer

[program:robotDemo_CeleryBeat]
command=/root/.local/share/virtualenvs/robotDemo-dwdcgdaG/bin/celer
```

猜测其中的：

`celery worker -A resources.tasks.celery`

和

`celery beat -A resources.tasks.celery`

导致了另外两个的process的产生

接着后来再去找更多的日志信息，最后发现：

`/celery-beat-robotDemo_CeleryBeat-stderr.log`

```
[20180830 01:28:35 INFO 26049 MainProcess 139969090553664 MainThrea
```

和：

`celery-worker-robotDemo_CeleryWorker-stderr.log`

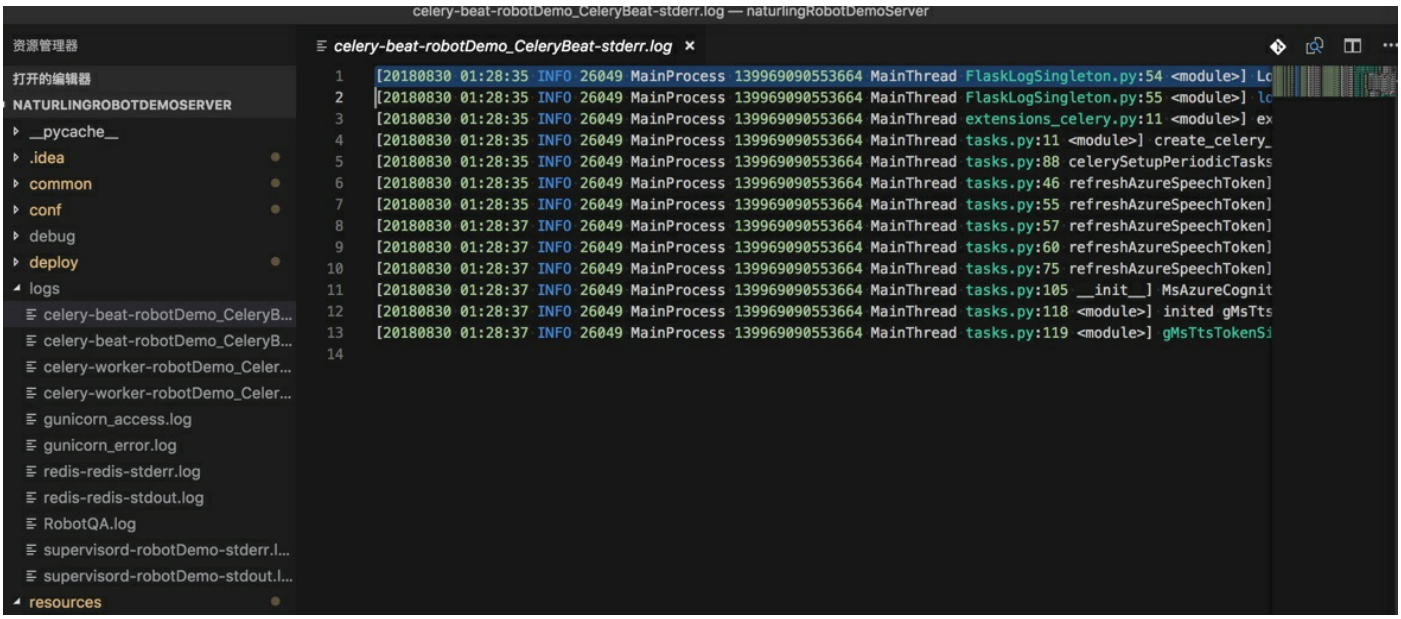
```
[20180830 01:28:35 INFO 26052 MainProcess 140308360062784 MainThrea
```

验证了之前的推测：

因为对应的log的第一条，就是我们之前找到的import log而输出了logSingleton的日志信息：

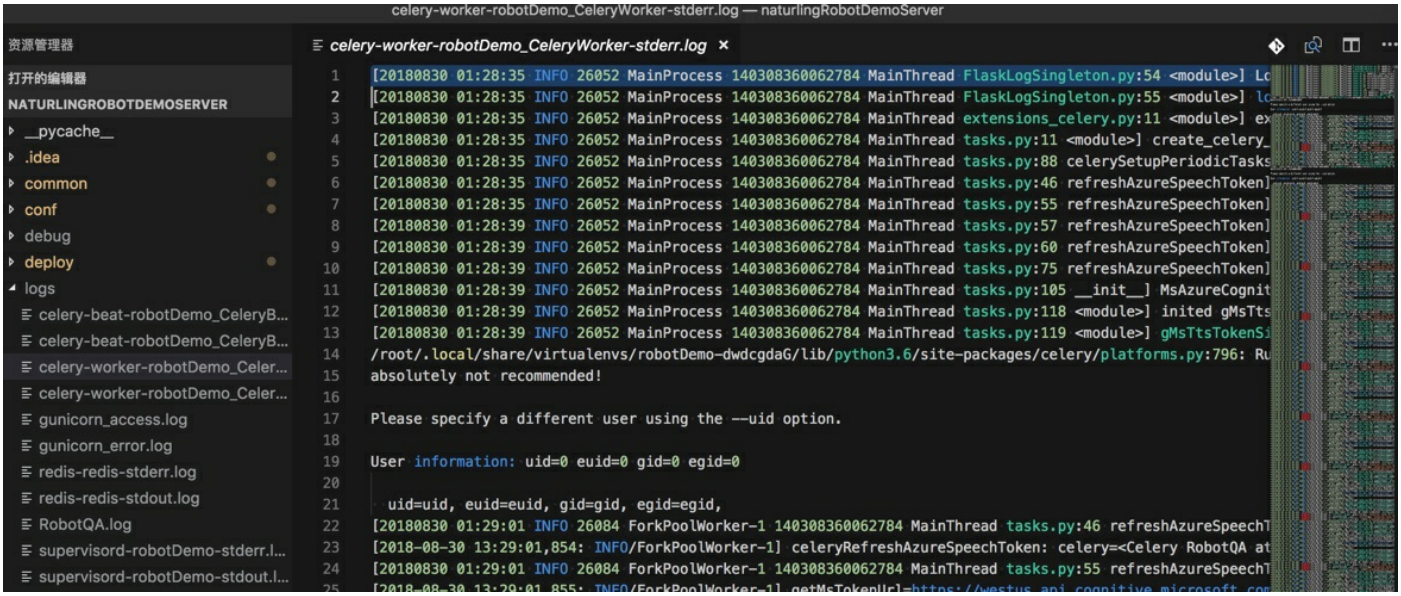
`celery-beat-robotDemo_CeleryBeat-stderr.log`

```
[20180830 01:28:35 INFO 26049 MainProcess 139969090553664 MainThrea
```



`celery-worker-robotDemo_CeleryWorker-stderr.log`

```
[20180830 01:28:35 INFO 26052 MainProcess 140308360062784 MainThrea
```



而其中：

- celery的woker的proceed的id是：26049
- celery的beat的proceed的id是：26052

就是最早发现的3个进程中的其中2个Process的ID的值：

```
[2018-08-30 13:28:37,129 INFO 26049 MainProcess 139969090553664 Mai

[2018-08-30 13:28:38,078 INFO 26063 MainProcess 140140210039848 Mai

[2018-08-30 13:28:39,545 INFO 26052 MainProcess 140308360062784 Mai
```

最终，而找到了：

除了supervisor+gunicorn去启动了Flask的app是单个Process之外：

supervisor还启动了Celery的worker和beat，这2个额外的Process

共3个线程，从而导致，虽然Flask的app中是单个Process，单例正常工作，

但是加上额外2个Process，导致单例失效：每个Process中初始化的实例都不同，无法保证单例的效果了。

总结：

此处之所以能够从大量的log日志中，最终分析找到产品额外2个进程的原因，主要是靠：

先是要了解自己写的代码的逻辑关系：此处涉及到近10个文件，以及好几个配置文件

其次要足够仔细和认真：要能否思路活跃，看到相关的日志信息后，能够实现基本的逻辑推理

一定的敏感度：能否在推理的基础上，思维活跃，偶尔联想到，猜到，可能和其他哪些模块有关系

最终通过 熟悉代码+足够认真+思维敏感 而找到问题原因并解决。

详见：

- 【已解决】用gunicorn的gevent解决之前多worker多Process线程的单例的数据共享

对于cygwin下编译buildroot时libtool的配置期间出错的折腾

先后尝试了可算达到上百个点了。

相比而言，之前的折腾时遇到比较多的，也就三五十个尝试的点，也就把问题搞定了。

期间有几次都打算放弃了，但是后来还是坚持继续找问题原因，最终功夫不负有心人，终于搞定了：

详见：

- 【已解决】cygwin下make编译buildroot时在libtool-2.2.10时出错：configure: error: C compiler cannot create executables
- 【已 解决】再次研究：Cygwin下编译Buildroot时在编译libtool-2.2.10时出错：/usr/lib/gcc/i686-pc- cygwin/4.7.3/cc1.exe: error while loading shared libraries: ?: cannot open shared object file: No such file or directory
- 【记 录】第三次去研究：Cygwin下编译Buildroot时在编译libtool-2.2.10时出错：/usr/lib/gcc/i686-pc- cygwin/4.7.3/cc1.exe: error while loading shared libraries: ?: cannot open shared object file: No such file or directory

对于cygwin下编译docbook的webhelp用到makefile调用java编译webhelp结果出错

期间，也基本是，都差不多放弃了

因为实在找不到是什么原因

而且网上也没有类似的参考资料

其他找到的资料，也没太大参考价值

最后，还是自己巧了，试了试java的classpath改为分好分隔后，虽然不行，但是想到了加上引号试试，结果才搞定的。

然后再回头找原因，才找到了该问题的根据原因并解决的。

详见：

- 【已解决】docbook中去make webhelp编译webhelp结果出错：Error: Could not find or load main class com.nexwave.nquindexer.IndexerMain

ReactNative iOS给导航栏添加图标

详见：

- 【总结】能代表自己的折腾精神的过程：React Native iOS中给导航栏中添加图标

折腾Flask-RQ2 + Redis

在折腾：

- ［已解决］Flask-RQ2＋redis的后台进程不工作

期间，就在迷茫的时候 能想到去试试

`rq worker`

最终明白`flask-rq2`是需要`rq worker`的后台服务才能工作的

Antd Pro中前端列表页面loading加载很慢

antd pro中，前端页面中列表的loading很慢：

开始就知道后端Django有一次性返回所有页面的数据，而不是当前页面数据的问题

但是发现好像是antd pro的loading的绑定有问题，后来发现不是

又以为和antd pro的yield 或call有问题，发现也不是

又以为是js的fetch有问题，发现早就返回response了

又以为是fetch后的response去json()数据量大时，很耗时

结果去花精力解决了后端Django只返回当前页数据后，依旧很慢，发现不是json()慢

再后来是，antd pro的reactjs前端的js的console的log 和 Django的后端的api请求 联合对此，最终发现：

Django后端的代码耗时太长，很多的mysql的查询和其他操作，导致很慢

具体点就是：

- 先是检索Script对象的history，很慢：要4秒
- 而得到的所有的页面的数据再去全部序列化serialize，很慢：要5秒

所以加起来要8，9秒。

所以需要去优化原有的处理逻辑：

- 搞清楚对于history的逻辑的处理，是否可以再优化
  - 后来搞清楚了：
    - 根据筛选条件过滤出所需要的所有的Script后，去获取每个Script的历史中版本号version最大的一个
      - 优化了此段逻辑，不需要去检索Script的History，从而时间上从4秒优化为不到1秒
- 只获取当前页面的数据（可以借用Django中Pagination，获得当前页面的object_list，然后再去序列化，就可以少很多时间了，从5秒优化为不到1秒

详见：

- 【已解决】Antd Pro中前端列表页面loading加载很慢

pipenv中运行PySpider出错：ImportError pycurl

libcurl link-time ssl backend (openssl) is different from compile-time ssl backend (none/other)

之前类似错误，简单的就已通过：

【已解决】pyspider运行出错：ImportError pycurl libcurl link-time ssl backend (openssl) is different from compile-time ssl backend (none/other) – 在路上

就解决了。

而此处的问题，是同事另外一台Mac。

折腾和尝试了各种思路和方向，都没有结果，详见：

【已解决】Mac中pipenv中运行PySpider出错：ImportError pycurl libcurl link-time ssl backend (openssl) is different from compile-time ssl backend (none/other)

而此处真正对解决问题的有帮助的点是：

除了之前已有的类似的经历，还要加上：

之前经历过2种类似和相关问题：

【已解决】pyspider运行出错：ImportError pycurl libcurl link-time ssl backend (openssl) is different from compile-time ssl backend (none/other) – 在路上

【已解决】Mac中编译安装pycurl失败：error: command ‘gcc’ failed with exit status 1

还要加上足够细心和敏感才能注意到：

【已解决】Mac中编译安装pycurl失败：error: command ‘gcc’ failed with

exit status 1

中是用的LibreSSL

以及也注意到了旧Mac中用的是OpenSSL

由此才能想到可能是openssl内部调用的库，不同：

- 旧的：OpenSSL
- 新的：LibreSSL

以及又（有想要去了解新技术的动力，所以才）去找了相关的解释：

tls – What are the main advantages of using LibreSSL in favor of OpenSSL – Information Security Stack Exchange

然后看到提到了是10.11的OS X之后也换用了LibreSSL

所以才想到这个点，可能是解决问题的方向

->最终经过升级Mac系统到最新版本Mojave而真正解决问题。

总结起来就是说：

能解决此问题有很多必要因素：

- 自己之前巧了遇到相关现象的问题
- 以及与之相关的类似其他的2个问题
  - 以及当时在
  - 【已解决】Mac中编译安装pycurl失败：error: command 'gcc' failed with exit status 1
  - 顺带去看了openssl的version信息
    - 才能有内部用的库的说明
- 并且这几个帖子都记录了详细过程
  - 包括brew install/reinstall openssl的详细过程，否则也不容易对比

发现
    - 新版mac中openssl是
        - https://homebrew.bintray.com/bottles/openssl-1.0.2p.sierra.bottle.tar.gz
    - 旧版mac中openssl是
        - https://homebrew.bintray.com/bottles/openssl-1.0.2p.high_sierra.bottle.tar.gz
            - 说明新系统high sierra是和旧的不一样的
- 够仔细和敏感
    - 能发现新旧问题中用的库是不同的
    - 旧的：OpenSSL
    - 新的：LibreSSL
- 有学习新技术的冲动：
    - 才会想起来去找OpenSSL的LibreSSL的区别
    - 才能找到：
    - tls – What are the main advantages of using LibreSSL in favor of OpenSSL – Information Security Stack Exchange
    - 的解释
        - 才能看到提到OS X 10.11之后也改用LibreSSL了
            - 最终才想到，会不会是系统问题
            - 才让同事升级系统到最新的macOS Mojave，才解决了此问题

crfian的学习能力

折腾IP代理池的过程

第一次折腾IP代理池时，对于各种相关概念不了解，连想要选购合适产品都不容易。

但是通过一些IP代理池服务商的网站产品的简单介绍：

- 【已解决】找个好用的IP代理池实现防止大众点评网站的反扒

自己悟出和理解出相关含义和区别，并整理出来了：

- 【已解决】搞懂IP代理池相关概念和逻辑

然后就可以购买和使用合适自己需求的代理了：

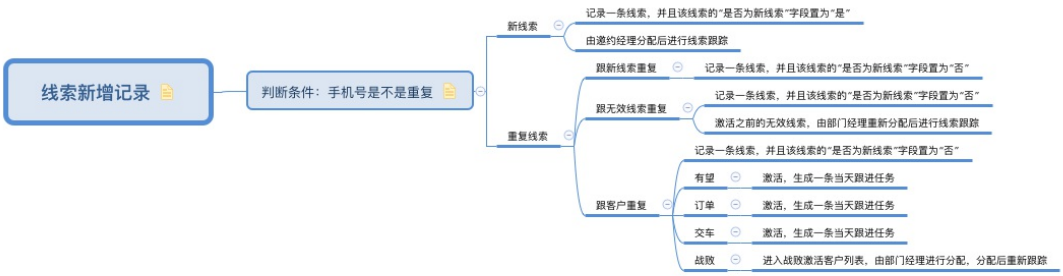
- 【已解决】购买多贝云IP代理池
- 【已解决】用Python代码测试多贝云代理IP是否生效
- 【已解决】PySpider中使用多贝云IP代理池实现每次请求IP都不同

crifan的逻辑能力

汽车销售领域内客户和线索逻辑的再优化

比如在 汽车销售领域内整理客户和线索的逻辑和流程时把已有的：

潜客和线索的关系和操作逻辑：



稍加整理，变为逻辑更加清楚的：