**系统详细设计**

--线下最终版

--24组

--姓名：葛萌萌（组长）

--学号：2017210461

--班级：2017211102

--小组其他成员：罗平

--时间：2019.06.11

目录

[1、系统概述 2](#_Toc11182342)

[2、数据结构说明 4](#_Toc11182343)

[3、模块设计 5](#_Toc11182344)

[4、接口设计 19](#_Toc11182345)

[5、数据库设计 20](#_Toc11182346)

[6、系统出错处理 21](#_Toc11182347)

[7、其他设计以及外来改进方向 22](#_Toc11182348)

# 1、系统概述

**1.1、系统简介**

基本情况：专注于爬虫方面开发，爬取知乎、微博、即刻APP上的用户信息，并且进行性能优化，项目部署管理，以及简要数据分析。

背景：互联网时代为我们提供了海量的数据资源，而这些资源并非直接呈现，需要采用合法合理高效的手段进行提取，所以爬虫应运而生，而克服多种反扒机制以及提升爬虫性能和简易化实现爬虫项目分发部署在实际中尤为重要。用户信息也是现在的热门话题，而各大社交平台提供了海量的公开用户信息，通过挖掘分析这些信息可以发现跟多有趣的是事情。

因此便有了我们**PowerFul crawler**项目组的“**用户信息爬虫**”项目。

**1.2、术语表**

|  |  |  |
| --- | --- | --- |
| **序号** | **术语或缩略语** | **说明性定义** |
| 1 | Pycharm vscode | 集成开发环境作为基本的编码调试运行工具 |
| 2 | Scrapy | 项目主要依赖的python高效爬虫库 |
| 3 | Redis mongodb | 数据库 |
| 4 | Robomongo RedisDesktopManager | 数据库连接工具 |
| 5 | Mongodb-charts | 数据库可视化工具 |
| 6 | Appium | APP自动化操作工具 |
| 7 | Charles mitmproxy | App抓包工具 |
| 8 | Xshell | 远程ssh连接工具 |
| 9 | Git | 项目管理工具 |
| 10 | Docker | 项目发布部署工具 |
| 11 | Genymotion | PC端的安卓模拟器 |
| 12 | Andriod\_SDK | 主要借助其中adb工具进行移动端控制 |
| 13 | VMware VirtualBox | PC端的虚拟机支持软件 |
| 14 | GitHub | 程序员交友平台。。。 |
| 15 | Python库 | 项目开发需要用到的python库不一一列举 |

**1.3、系统运行环境**

**硬件平台：**

安装docker的主机（容器的存在使项目部署如此便捷）+云主机（分布式支持）

**操作系统：**

不限系统，项目最终发布的容器内集成了需要的python环境

**数据库系统：**

redis mongodb

有一台主机必须要有redis（分布式调度主机）

其余主机需要有容器技术实现redis mongodb环境的安装

**编程平台（开发工具）：**

Pycharm Vscode Git

**网络协议：**

http https

**1.4、开发环境**

系统分析、程序设计工具：

云笔记，xmind8，Visio

程序开发：

工程工具：

pycharm vscode git docker redis mongodb 数据库连接可视化工具appium charles mitmproxy Andriod\_SDK xshell Genymotion VMware VirtualBox

开发语言：

python3

工具软件的名称、版本：

Pycharm vscode 集成开发环境作为基本的编码调试运行工具

Redis mongodb 以及一些链接客户端作为数据库以及数据库操作可视化工具

Appium app 自动化操作工具

Charles mitmproxy App抓包工具

Xshell 远程ssh连接工具

Git 项目管理工具

Docker 项目发布部署工具

Genymotion PC端的安卓模拟器

Andriod\_SDK 主要借助其中adb工具进行移动端控制

VMware VirtualBox PC端的虚拟机

# 2、数据结构说明

**全局数据常量：**

通用的全局变量定义在了项目的每个组成部分(model)的config.py文件中

在proxypool cooliespool appspider的model中定义在了config.py中

在基于scrapy两个model—zhihuspider weibospider定义在了settings.py中

这些全局变量主要是**数据库连接配置信息，pool所提供的功能控制开关，pool接口连接信息，程序运行所需要的全局配置信息，scrapy的基本配置信息，APP连接控制所需信息**等等…

而通过这些独立的配置文件而控制的全局配置信息，可以帮户用户/开发人员轻松地变更更新扩展程序，这边是基于配置读取配置的力量。

**变量：**

定义在程序代码中，由于基于面对对象思想编程，大多为类内变量，当然在mitmdump处理脚本中由于程序支持需要定义了一部分局部变量属于面向过程程序设计，具体变量过于繁多不一一列举，包括数据库连接对象，request response对象及其属性而单独定义的对象，webdriver驱动app的对象，cookies proxy对象，异常对象，以及大量中间临时局部变量等等…

**数据结构：**

**爬虫逻辑数据结构**

用户信息爬虫套路相似，基于一个用户的粉丝关注列表实现递归爬去，本项目的知乎微博爬虫都是采用队列维护生成的request从而顺序爬取，而request的生成基于当前用户的关注粉丝列表，所以整体上呈现广度优先搜索----由于基于队列维护请求并且进行爬取；APP爬虫由于需要自动化操作所以不便与实现递归爬去，所以采用生成初始用户队列基于这些用户的粉丝关注列表进行用户信息获取，相当于简单的队列遍历。

**Pool数据结构**

在proxypool cookiespool部分使用维护的redis中的sortedset以及基于key-value的hashset从而实现去重存储。

**数据库存储数据结构**

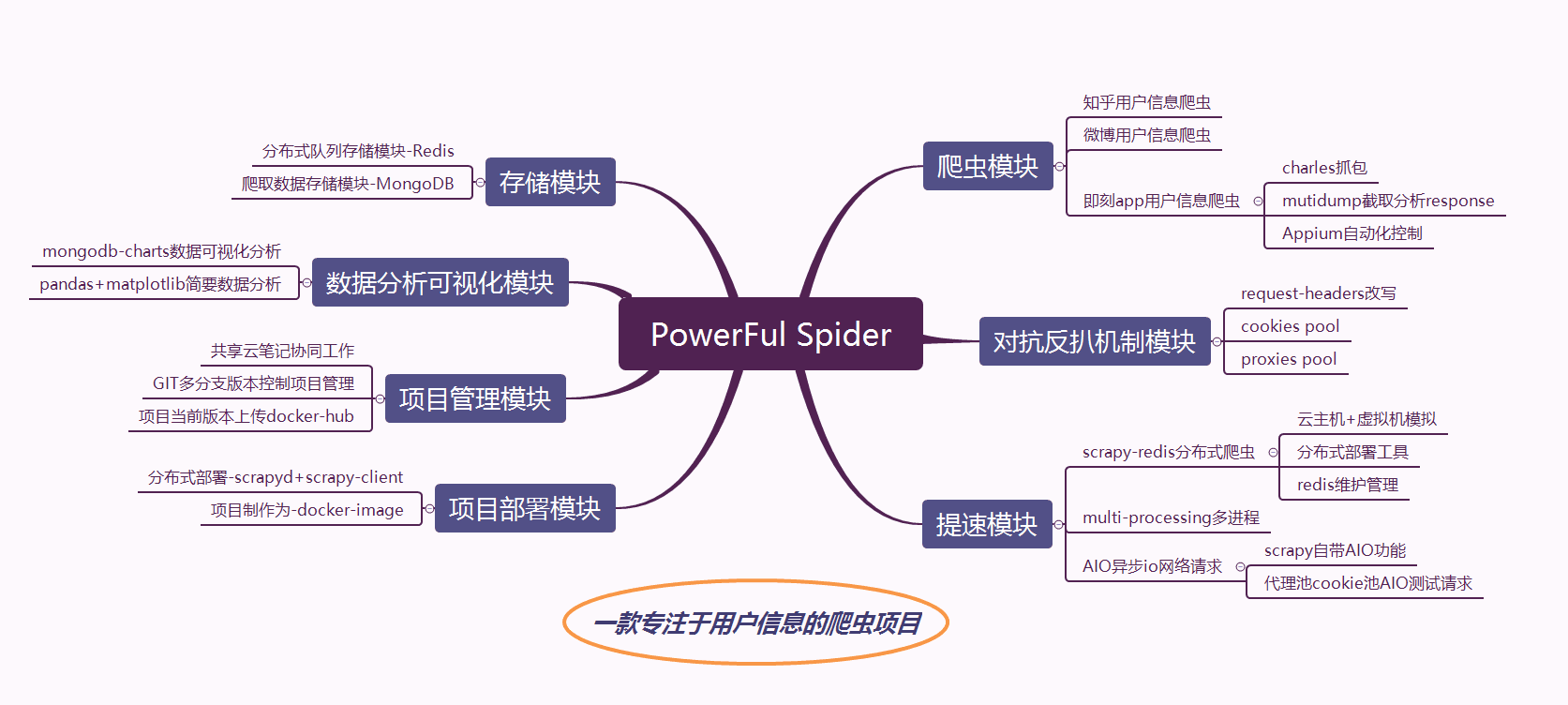
数据库（mongodb redis）内部所使用的高效数据存储结构比如B,B-,B+树等等不赘述，nosql中比如mongodb的keyvalue存储方式很高效。

**响应内容分析及数据存储数据结构**

本项目中由于现有网络请求返回的response.text大多为json字符串，所以json数据结构的分析提取是本项目中最关键最基础的部分，所谓json即嵌套的keyvalue字典结构，通过遍历key可以迅速得到value部分进行数据提取，并且支持空数据字段，使用python提供的json模块可以与txt格式文本迅速转换实现分析及存储。而在数据存储中由于mongodb支持keyvalue形式存储，所以将提取到的信息转换为json直接实现数据库存储，方便快捷。

# 3、模块设计

**3.1、软件基本结构**



**3.2、功能设计说明**

我们所做的项目在爬虫方面可以实现三部分内容：知乎用户信息爬虫、微博用户信息爬虫以及即刻app用户信息爬虫。

而在爬取微博、知乎平台时会遇到反爬机制，我们通过request-headers改写请求头以及proxypool，cookiespool实现ip以及cookies切换,从而对抗知乎微博反爬机制。

在爬取过程中由于网络IO以及CPU限制，导致爬取速度受到限制，于是我们采用scrapy-redis实现分布式爬虫、multi-processing多进程和AIO异步IO网络请求来进行提速。

在数据存储方面，我们用分布式队列存储，将scrapy生成的请求存储到中心主机的redis队列中，每个从节点主机获取中心主机的request队列进行爬取，并更新该队列，实现分布式调度，并将爬取到的数据存储到数据库MongoDB中。

在数据可视化分析方面，用mongodb-charts以及pandas+matplotlib（待完成）对数据进行可视化方便简要分析。

在项目管理部分，用共享云笔记协同工作、GIT多分布版本控制项目管理，然后将项目上传至docker-hub。

在项目部署方面，采用scrapyd+scrapy-client进行分布式部署并且通过docker容器技术，制作docker-image实现项目发布，便捷部署。

**3.3、模块1**

**爬虫模块**

三个爬虫，分别是基于scrapy的知乎以及微博用户信息爬虫，以及基于charles抓包分析，appium自动化操作，mitmdump抓包脚本处理的即刻APP爬虫。

[1] 知乎爬虫 微博爬虫

**思路：**

基于用户🡪粉丝列表关注列表+翻页🡪用户

实现递归爬去 递归生成请求 保存到scrapy的request队列中

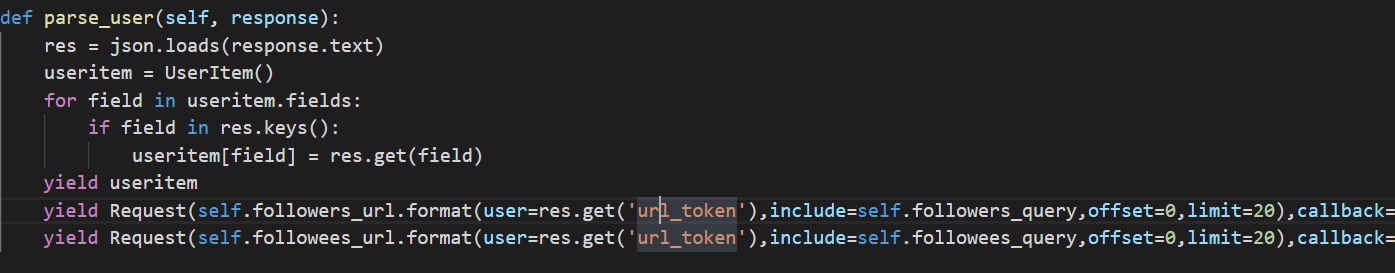
**具体代码剖析：**

**（1）爬虫代码**

知乎爬虫：

以下是在用户解析，粉丝列表关注列表解析的函数

用户解析函数需要填充对象field 从而生成需要的item

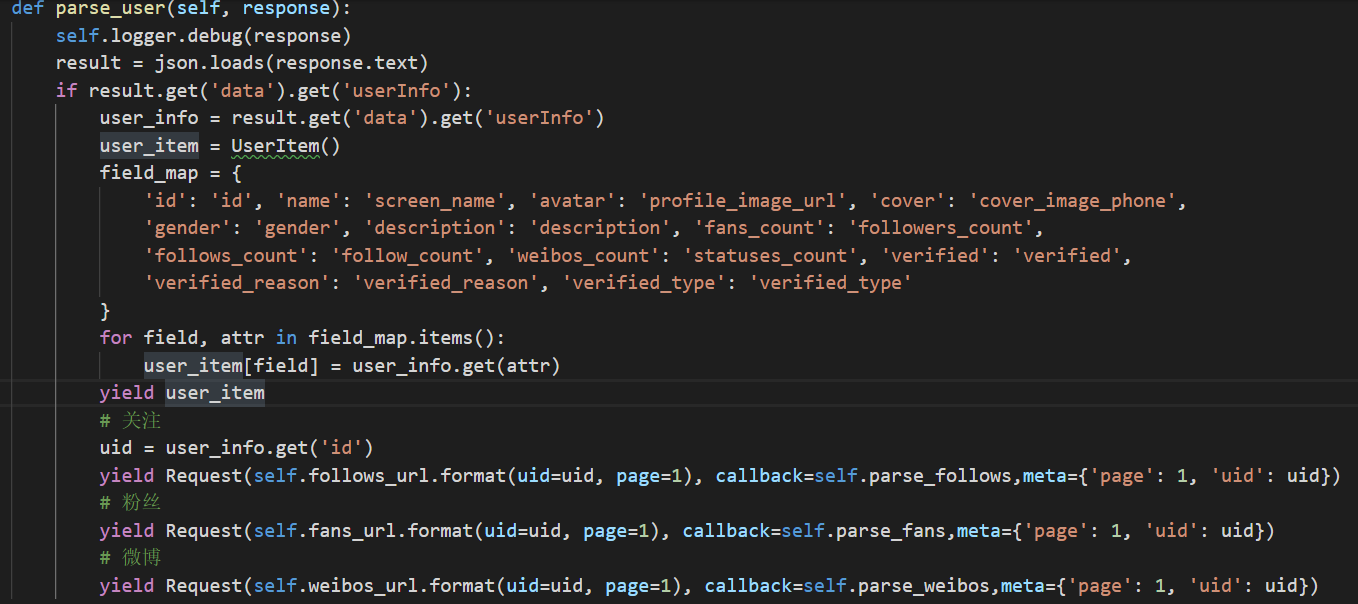


粉丝列表关注列表函数需要对于每个用户回调用户解析函数，并且翻页



微博爬虫：

类似上面的逻辑，只是生成对象item使用了预先定义的字典映射的方式，从而填充item的每个field，返回item并且发起粉丝列表关注列表以及微博正文的请求



粉丝关注列表解析函数类似知乎爬虫，只是额外改进添加了对于粉丝列表关注列表进行单独保存的功能，并且为了保证能够正确异步存储粉丝关注列表，需要在构造的请求中通过meta传递元数据，保证正确存储。

基本功能：生成用户request ，关注列表/粉丝列表生成 ，请求下一页



粉丝列表函数类似不再赘述

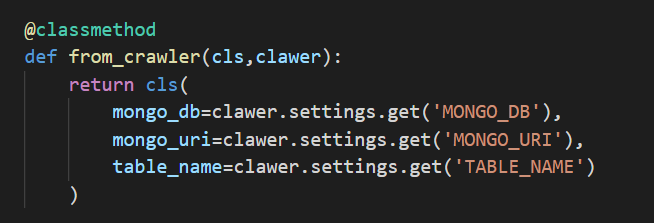
**（2）处理代码**

处理部分包括对于item的处理，以及downloader的处理

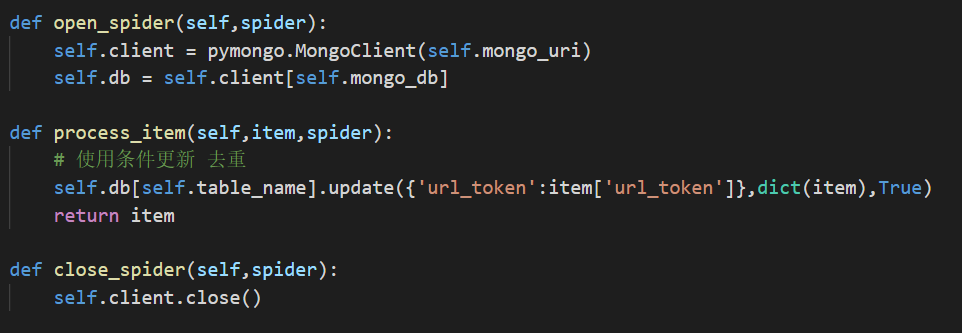
所以需要用到ItemPipeline + DownloaderMiddleware

在itempipline中需要做到存储item到mongodb中，并且对于数据进行清洗格式化，并且对于微博爬虫来说，需要对于生成的关注列表粉丝列表item对象进行单独处理，实现异步存储，以下是知乎中itempipline代码：

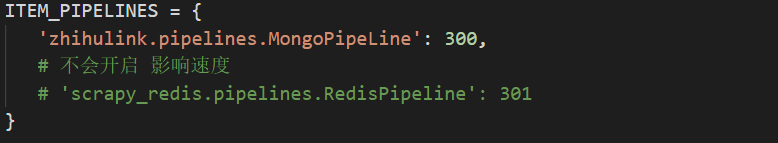
读取配置



过程处理+mongodb存储



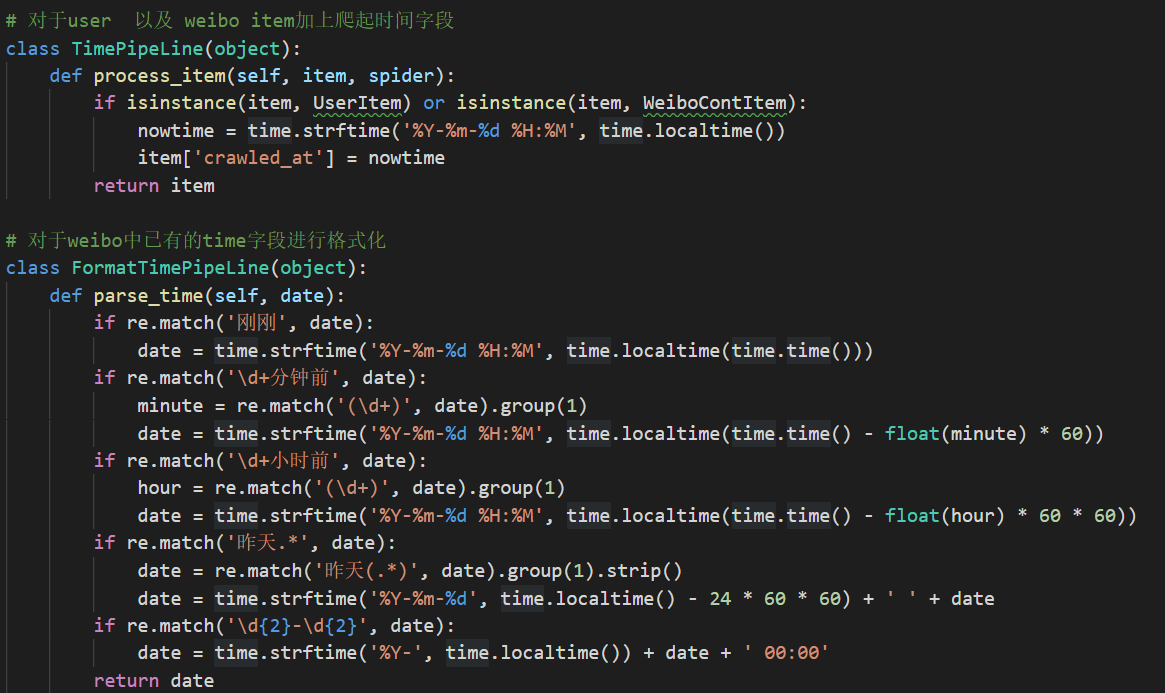
Settings.py中调用



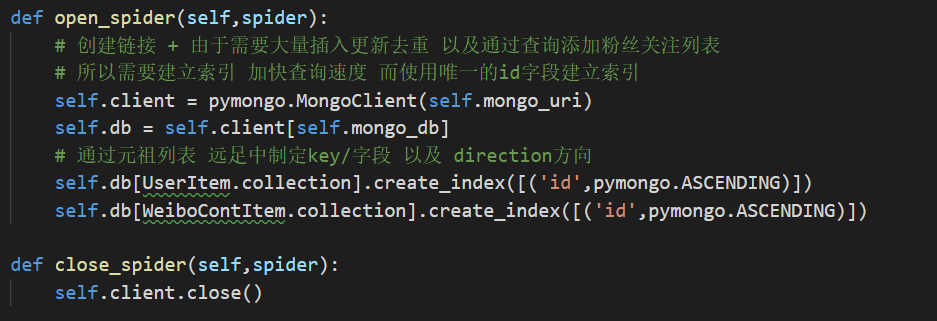
------------------------------------------------------------------------------------------

以下是微博中itempipeline代码

数据清洗时间处理



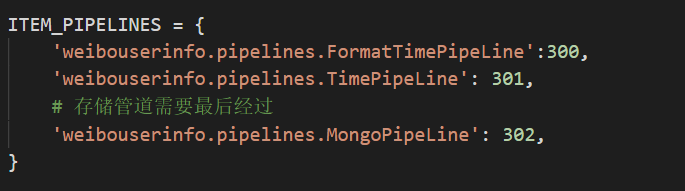
打开关闭mongodb链接



进行实际的mongodb存储---需要针对不同的item进行类别检查，从而选用合理的处理方式存储方式--------useritem/weibocontentitem--------------RelationItem

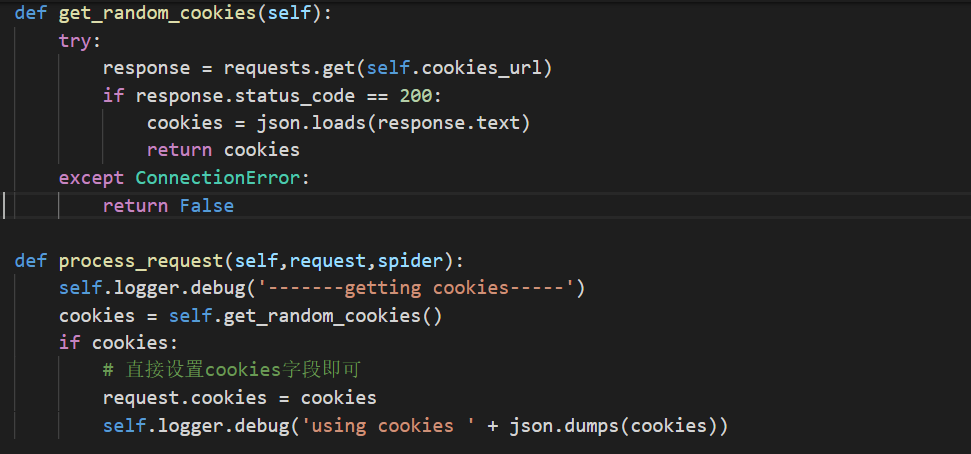


Settings.py中的通过配置调用



对于downloadmiddleware需要在爬取过程中检查响应状态从而实时通过flask接口获取代理/cookies从而切换cookies/ip伪装headers，实现批量爬去，应对微博知乎的强反爬虫机制

随机获取cookies并且直接每次请求更换cookies

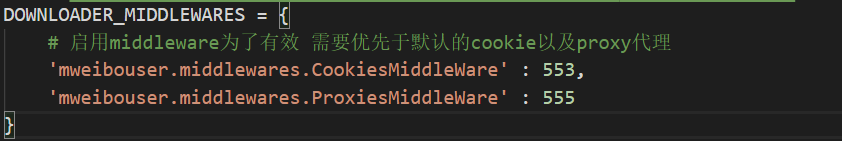


通过检查当前retry\_times字段值从而确定是否需要获取代理，更换ip

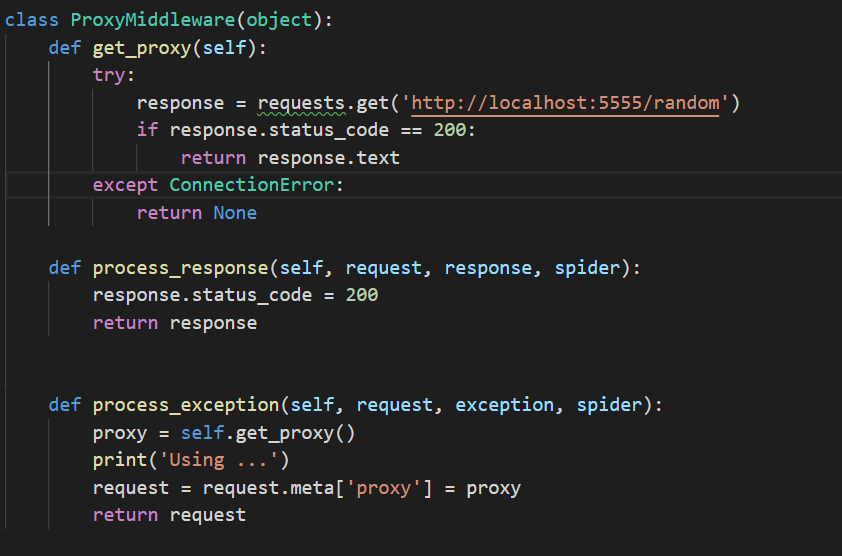
微博爬虫



在settings中进行配置性的调用



知乎爬虫类似，只是通过检查下载过程中exception实现使用代理切换ip



**3.4、模块2**

**代理池，cookies池模块**

由于知乎微博强反扒机制，通过购买微博账号搭建cookies池

从网络上获取免费代理ip构建代理池，实现ip,cookies切换从而有效爬取

Cookies池尚未完成，所以使用的是手动登录，实现cookies获取

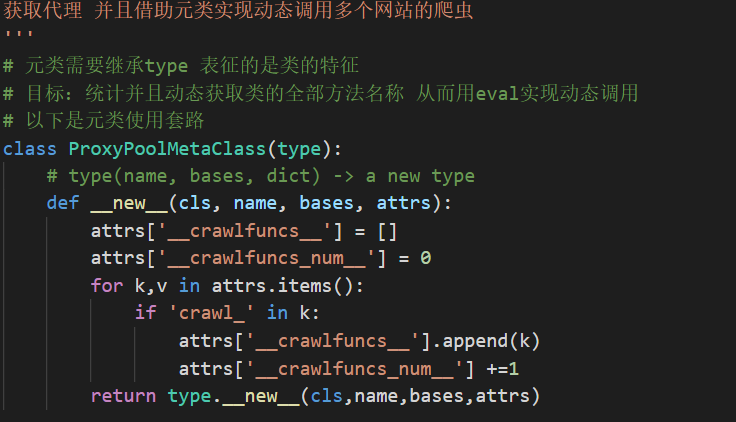
Ip/cookies pool二者逻辑相同，分为三个子成分，getter tester interface

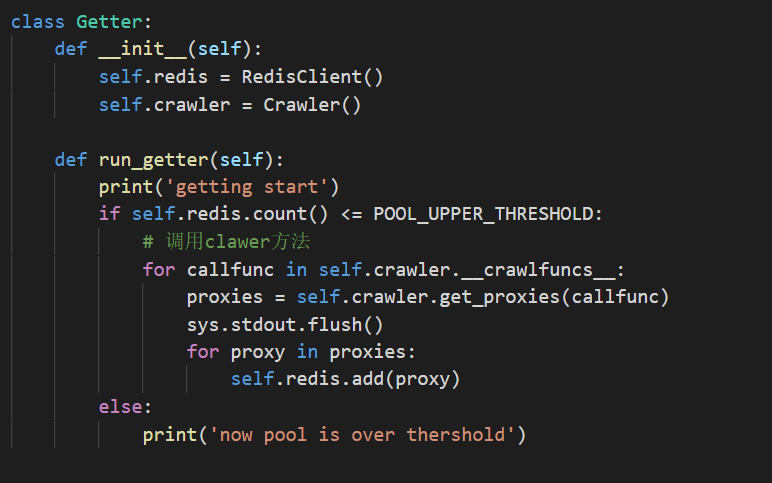
**[1] getter**

代理池通过爬虫获取免费代理网站代理

Cookies通过selenium使用redis中的账号进行模拟登陆，从而获取cookies

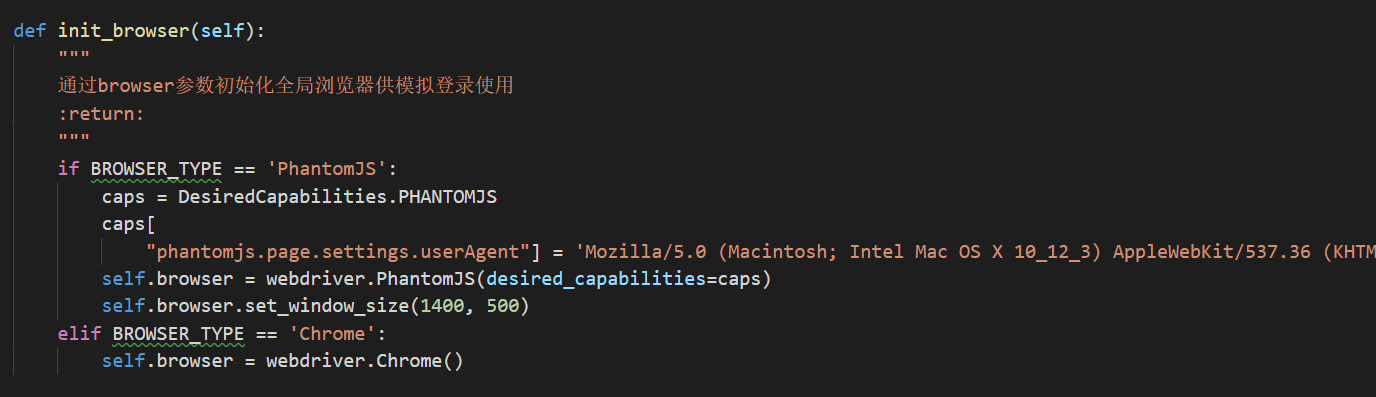
代理池getter





Cookies池getter

初试化selenium浏览器



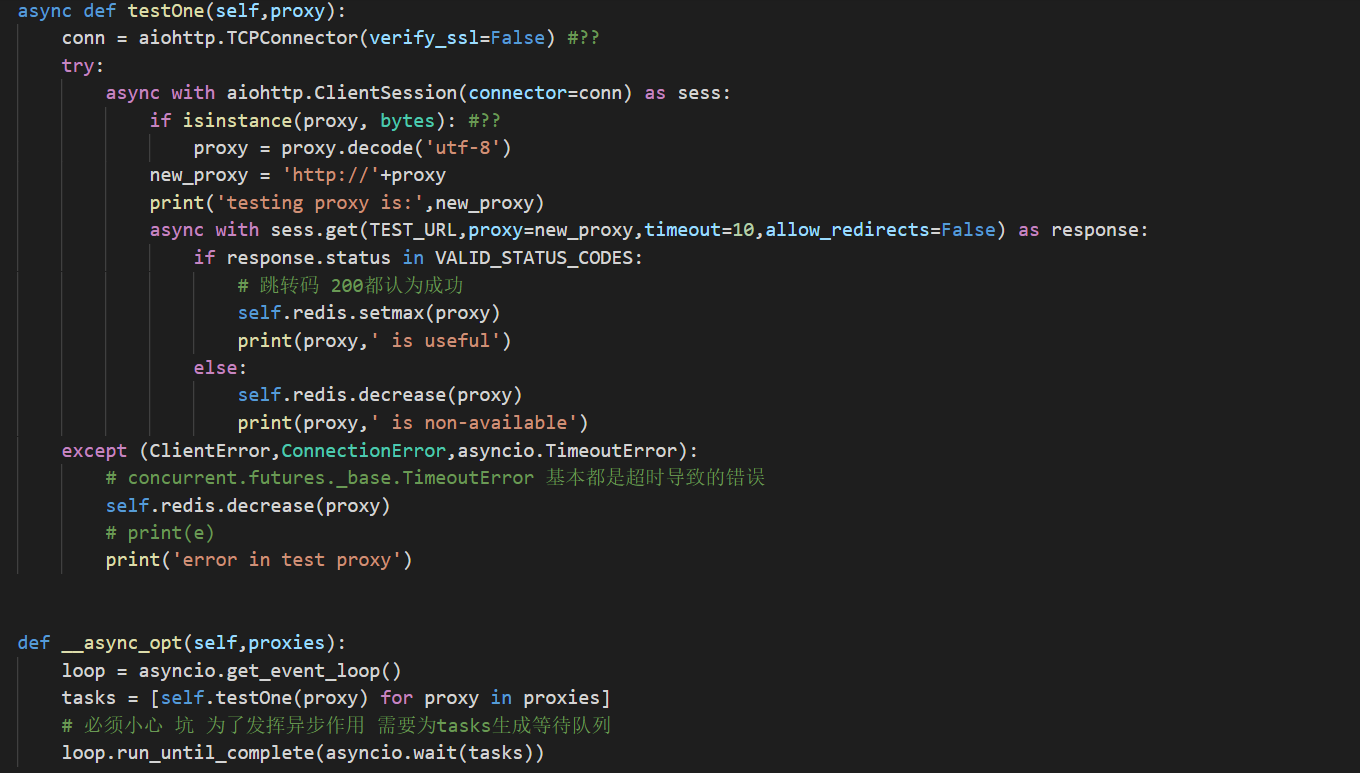
进行模拟登陆



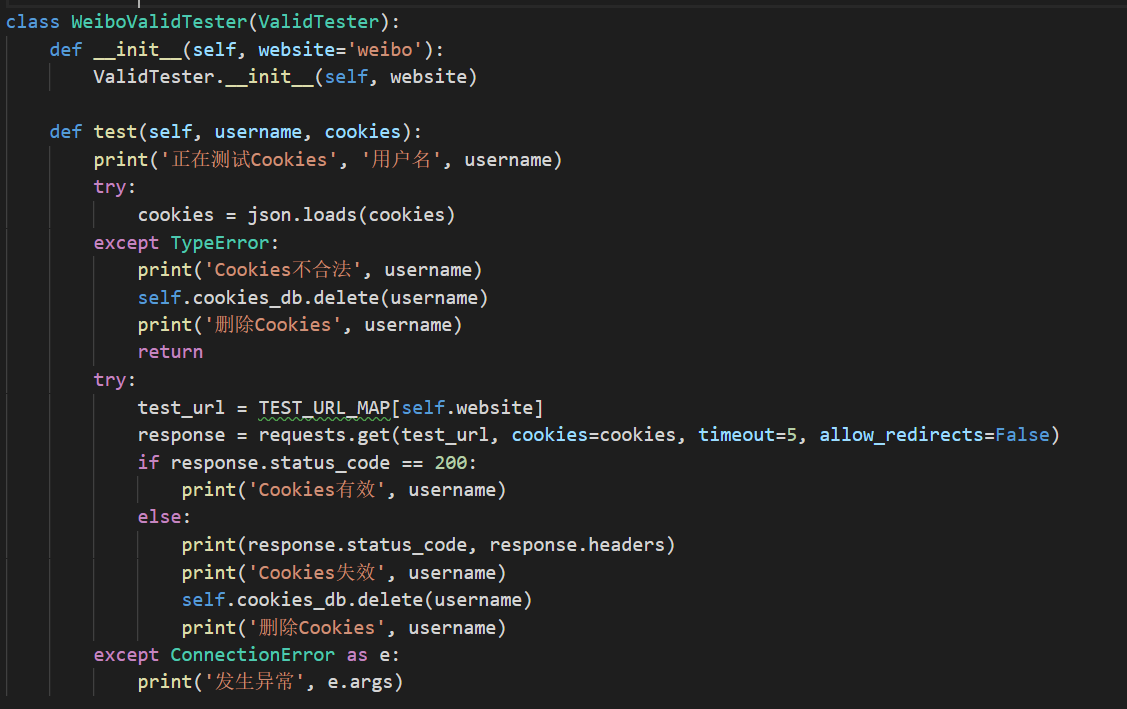
**[2]tester**

Proxy pool tester

使用异步方式进行网络请求，测试代理/cookies可用性，由于网络IO是主要性能障碍，所以异步aiohttp+asyncio有效提升了速度



Cookies pool tester



通过继承关系，使cookies不局限于一个微博网站，可以通过继承父类实现子类，边界扩展，但是代理池由于切换的是通用的IP所以不需要进行扩展的考虑

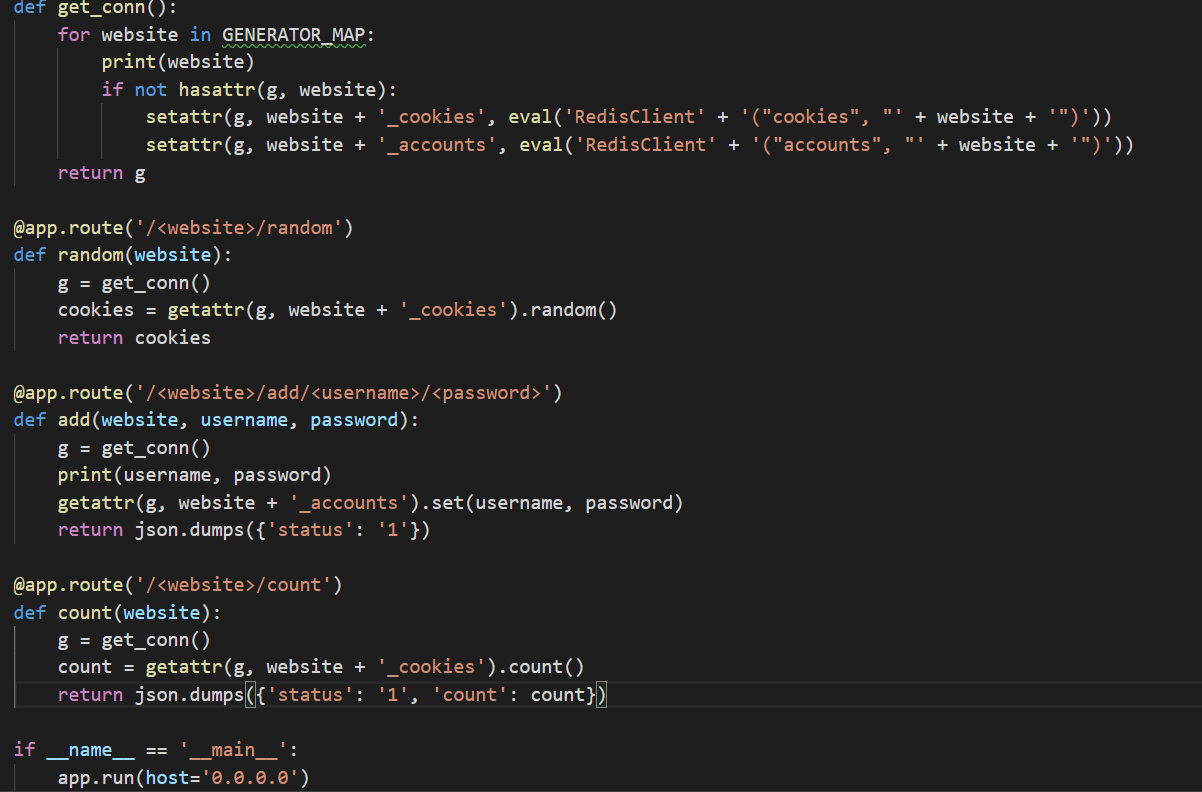
**[3]interface**

使用flask这个简便的web框架实现为redis中的proxy/cookies提供对外访问获取接口，而爬虫只需要通过访问接口拿到数据即刻获得proxy/cookies

代理池接口

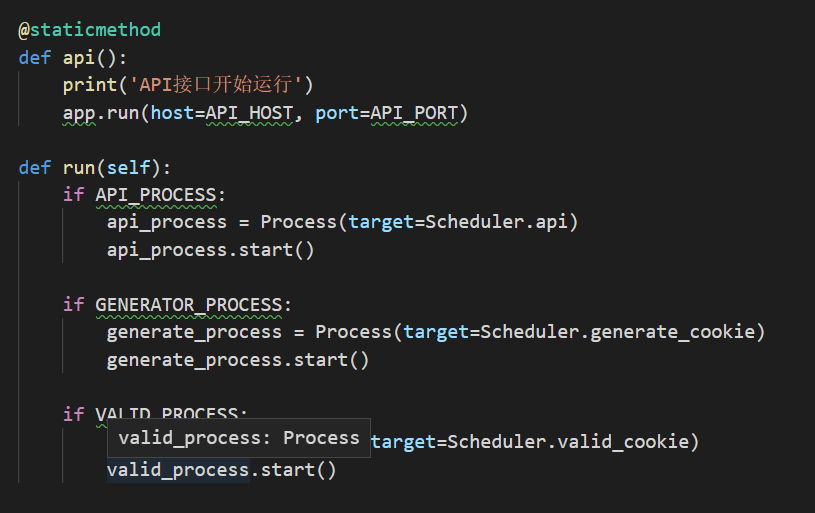


Cookies池接口



**最后通过调度器使用多进程multiprocessing，并且读取配置文件中的启动信息，从而进行统一调度**





**3.5、模块3**

**提速模块**

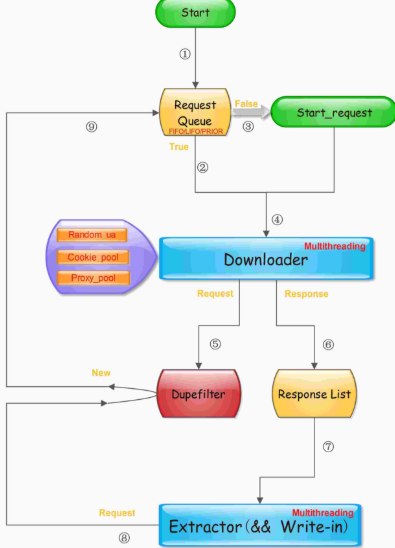
在此模块中，我们通过三方面对爬虫爬取速度进行提速：

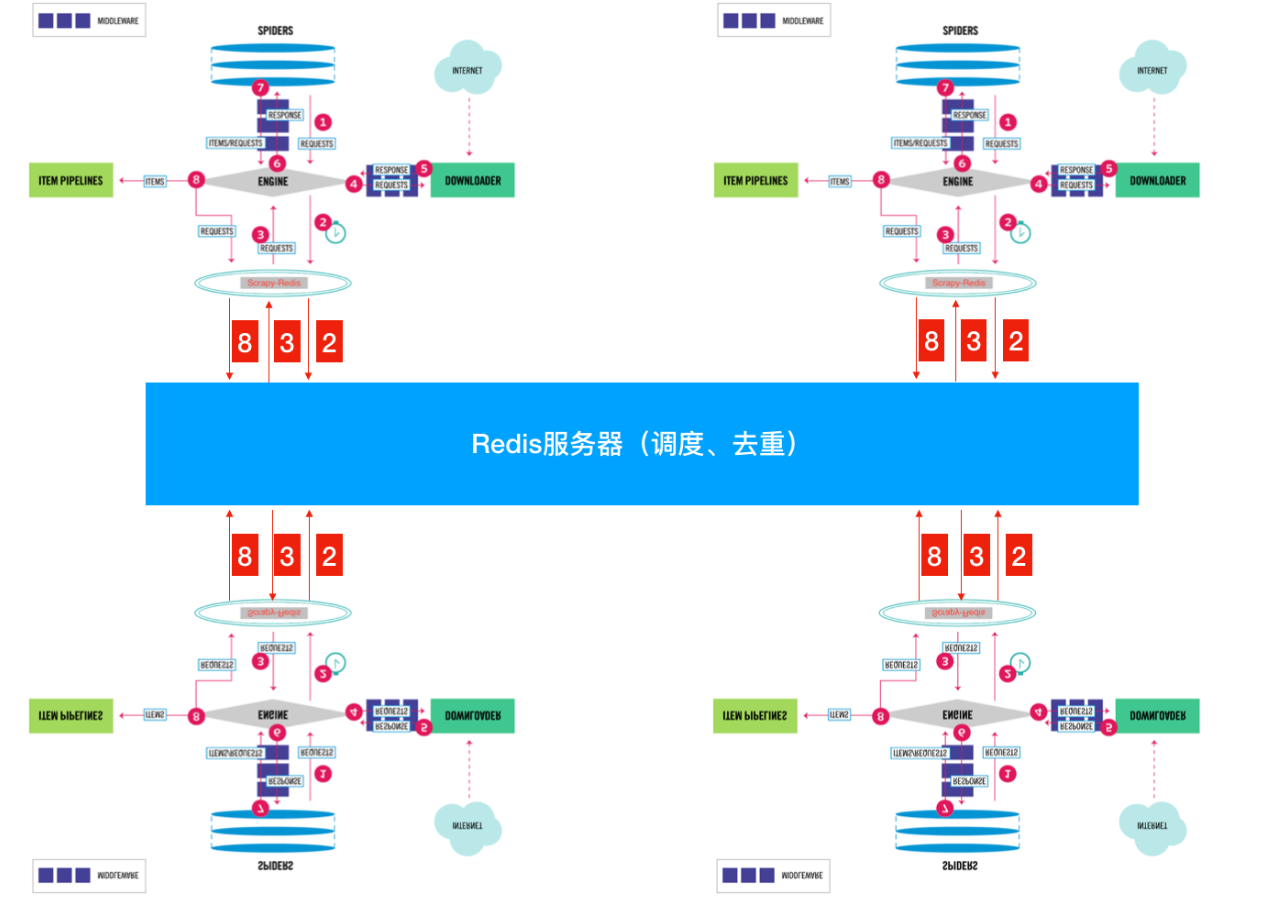
scrapy-redis分布式爬虫

multi-processing多进程

AIO异步IO网络请求

**[1]Scrapy-redis分布式爬虫:**

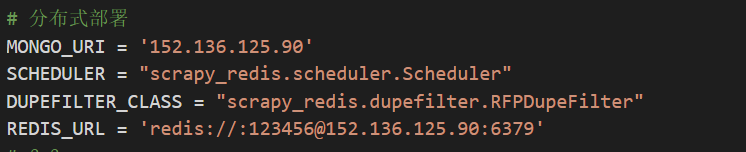




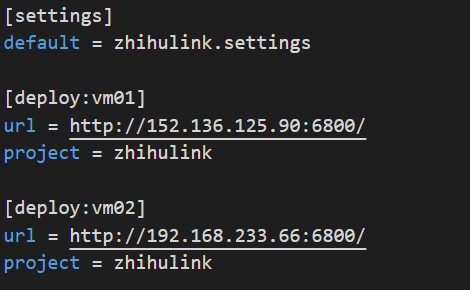
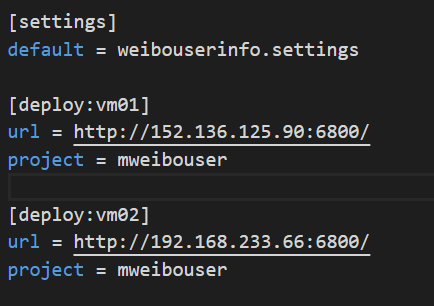
**原理图如上**

**以下介绍分为分布式代码配置部分+分布式部署部分**

**配置**



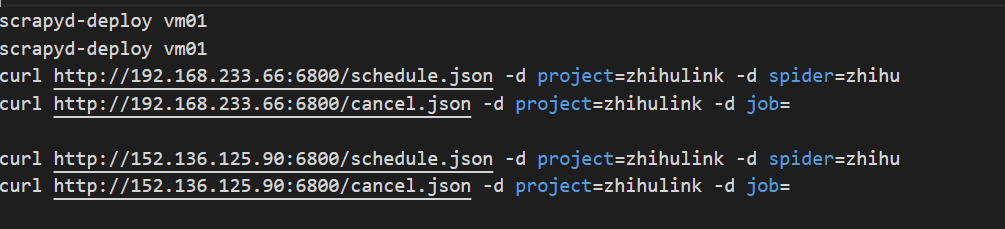
**部署**



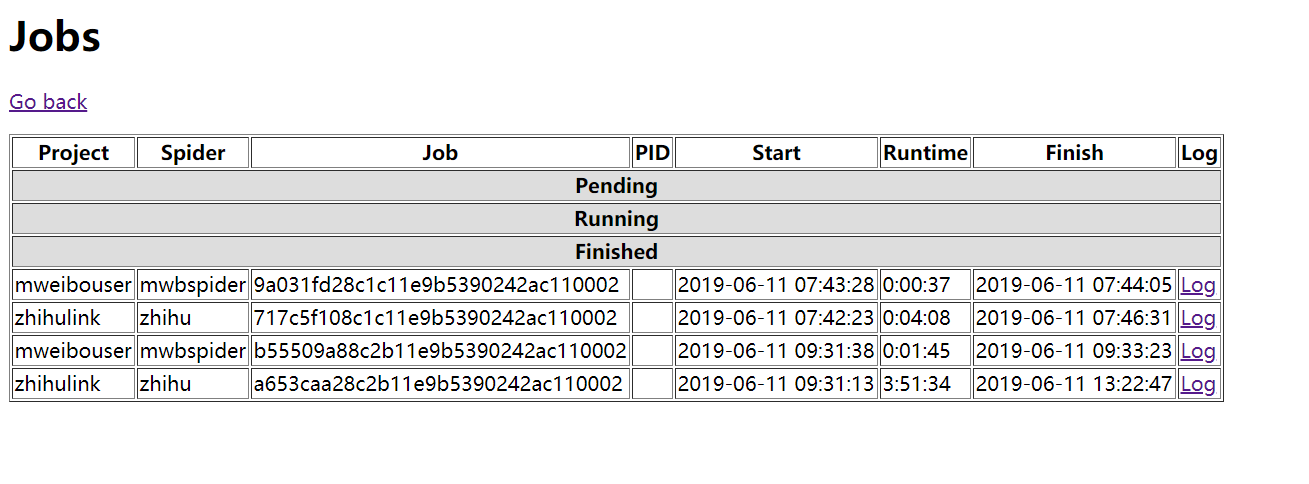
分布式调度

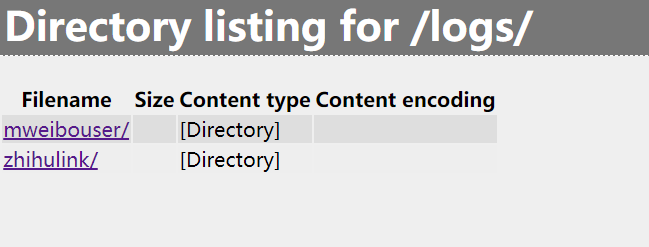
首先需要在目标主机上运行scrapyd并且监听来自外部的请求在6800端口

之后再本地主机上运行以下命令，即可实现从项目打包部署，到爬虫运行启动终止调度



并且可以通过浏览器访问相应分布式节点的scrapyd接口，查看jobs 以及 logs信息

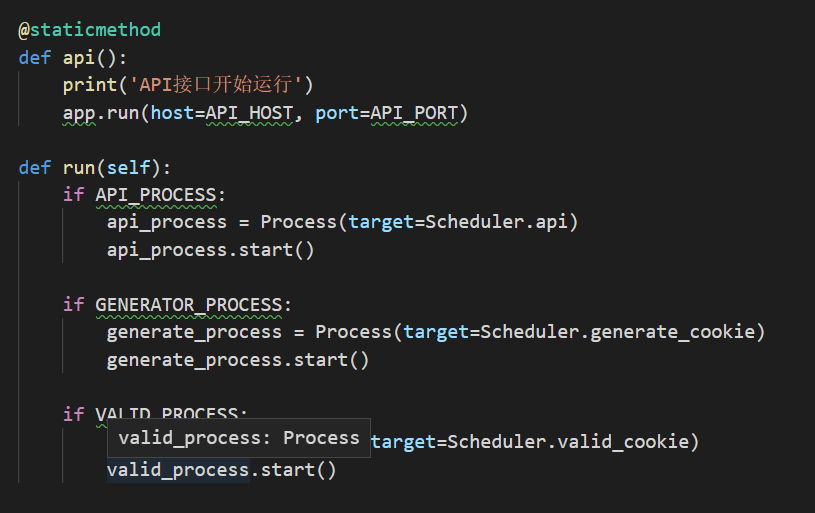




**[2]Multi-processing多进程实现pool调度：**

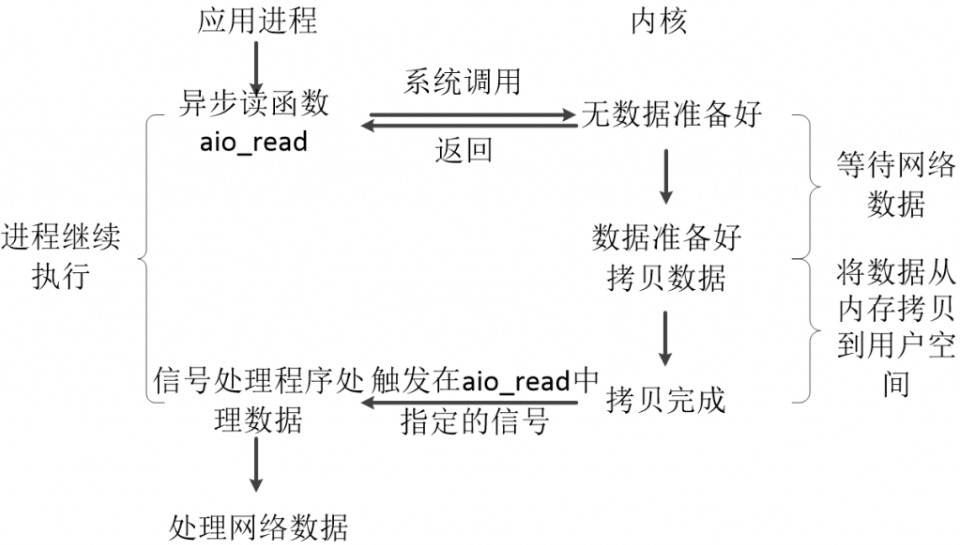
通过多进程实现多个模块同时开启进行调用，减少阻塞所等待的时间，实现同步运行





**[3]AIO异步io网络请求：**

**AIO原理**



在一个进程中为了执行多个 I/O 请求而对计算操作和 I/O 处理进行重叠处理的能力利用了处理速度与 I/O 速度之间的差异。当一个或多个 I/O 请求挂起时，CPU 可以执行其他任务；或者更为常见的是，在发起其他 I/O 的同时对已经完成的 I/O 进行操作。



**AIO实现**

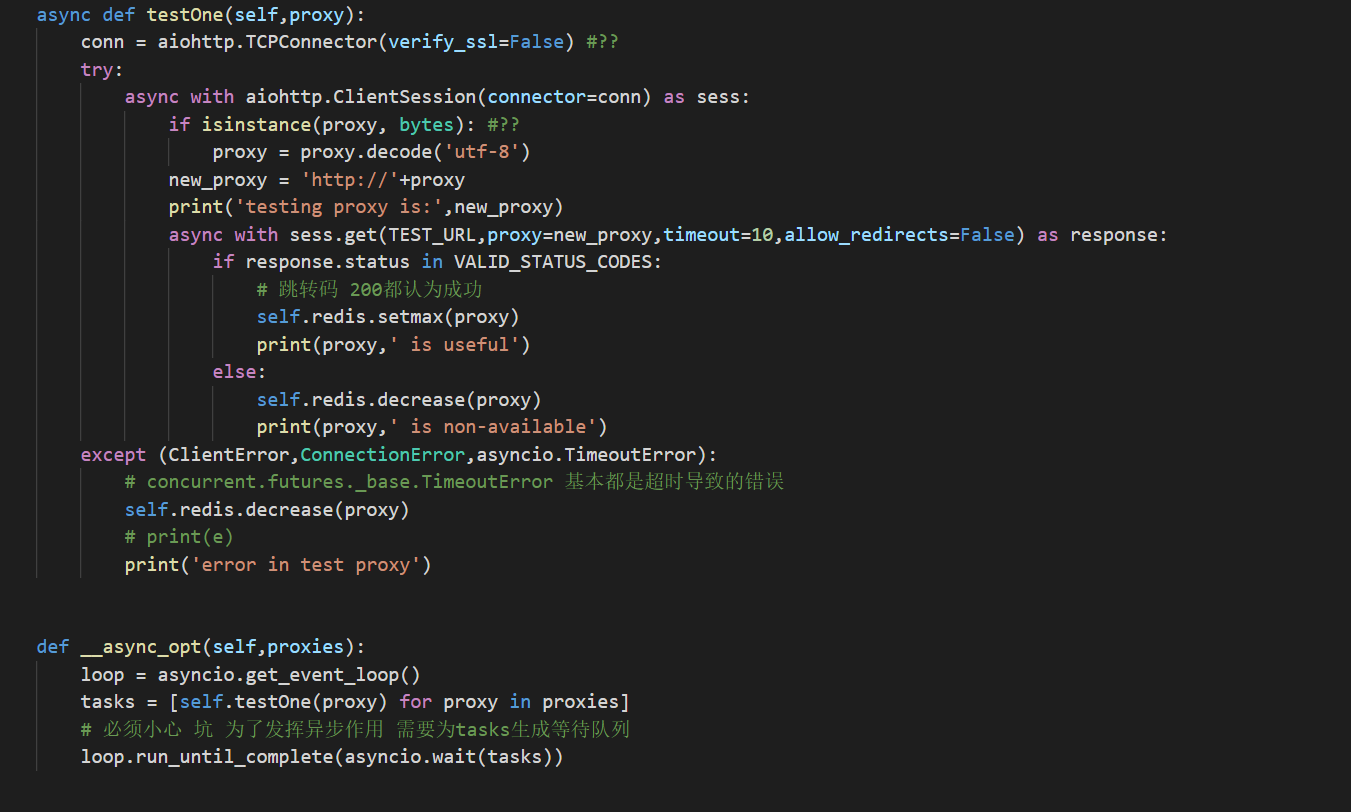
爬虫部分的AIO的通过scrapy自带的爬虫异步请求功能得以实现

scrapy基于异步框架twistted

**NOTE:分布式部署需要使用18.9.0而不能是19.2.0的twisted，否则无法分布式部署**

同时在代理池cookies池中使用异步请求aiohttp模块实现异步网络请求验证cookie以及ip有效性，从而缓解网络IO带来的性能瓶颈。





**3.6、模块4**

**数据存储及可视化模块**

**[1]数据存储：**

1. 分布式队列存储-redis：

redis是一个开源的、使用C语言编写的、支持网络交互的、可基于内存也可持久化的Key-Value数据库。Redis将数据存储到内存中，使得读写速度非常快。使用redis列表类型保存数据，队列采用左进右出的模式保证队列消息的顺序性。

1. 将爬取到的数据存储到MongoDB中：

MongoDB 是由C++语言编写的，是一个基于分布式文件存储的开源数据库系统。MongoDB 将数据存储为一个文档，数据结构由键值(key=>value)对组成



**[2]数据分析可视化：**

1. mongodb-charts可视化分析：mongodb-charts是在mongodb数据上构建可视化最快的方法。我们通过添加数据源、创建一个仪表板和创建图表三个步骤连接到mongodb图表服务器，从而实现数据的可视化
2. pandas+matplotlib对数据进行可视化简要分析：Pandas与Matplotlib配合绘制出折线图, 散点图, 饼图, 柱形图, 直方图等五大基本图形，对数据进行了可视化，从而方便我们的简单分析。（目前要根据进度来看，不一定会实现）

**3.7、模块5**

**项目管理及部署模块：**

[1]我们通过共享云笔记进行协同合作，推进实验的进行，有什么问题在云端进行讨论与解决，当然，实验课时间通过面对面交讨论问题。

[2]通过GIT多分布版本控制项目管理：从master->zhihu->weibo->v1->master

通过分支合并保证master分支可用性，而在从分支上进行拓展开发，当开发完成与master分支进行合并，模拟了产品更新升级的过程，而且必要时候可以实现版本回退，git分支管理的意义不言而喻

[3]将项目打包成docker-image提交到docker-hub以及阿里云提供的docker registry

利用容器技术实现项目部署，分别有scrapyd的docker-images，以及知乎爬虫，微博爬虫的docker-images

[4]分布式部署scrapyd+scrapy-client

通过多台机器爬取使用scrapyd来监听请求命令，进行对多个爬虫的上传、启动、暂停等调度管理。

# 4、接口设计

**4.1、 用户、外部接口**

用户只需要用安装有docker的主机即可通过pull docker-image的方式获取项目运行环境，并通过架设多台主机以及一台装有redis的主机实现分布式部署，但是过程中需要对得到的docker镜像中的项目代码中的配置文件进行简要修改，从而适应于分布式以及远程连接环境。

**4.2、 内部接口**

4.2.1、 接口说明

[1]Proxypool以及cookiespool通过Flask提供爬虫爬取过程中需要的cookie以及proxy的获取接口

[2]数据库接口redis+mongodb

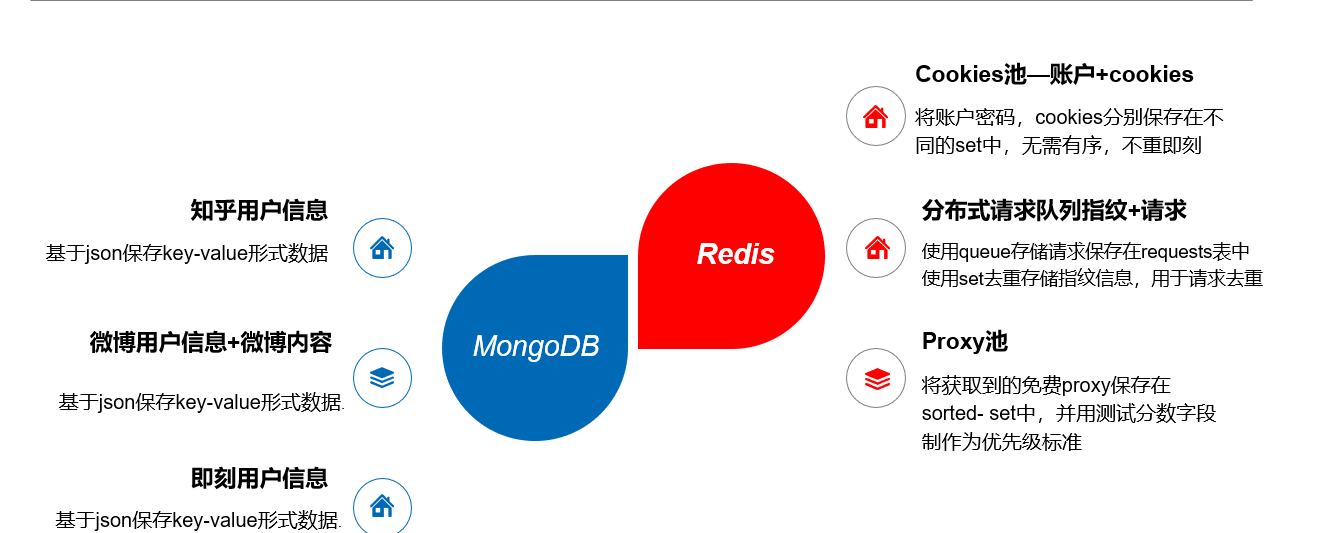
4.2.2、 调用方式

[1]进入项目目录中，之后单独进入cookies池以及proxy池中，修改config.py中设置的开关接口，打开api接口，运行池，之后通过在爬虫通过访问flask提供的网络借口比如random获取网页内容—proxyip/cookies。

后续可以将这些调用过程单独做成项目中的一个启动模块实现快速调度。

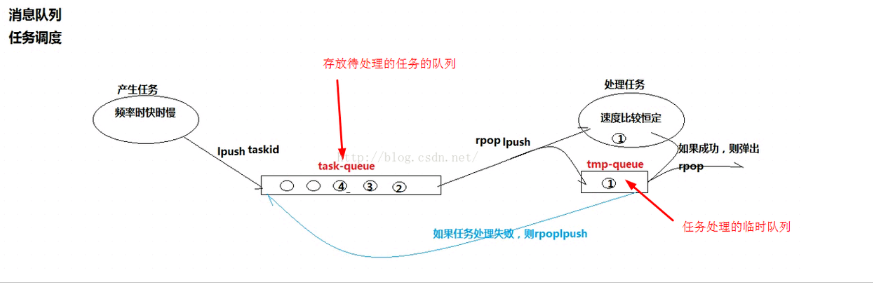
[2]数据库接口通过pymongo/Redis等python模块获取数据库连接并且进行数据存取操作，还可以通过数据库客户端实现数据库数据可视化分析。

# 5、数据库设计



[1]分布式以及pool中redis的应用:

我们用redis分布式队列存储对爬取过程中的requests以及指纹信息进行存储，队列采用左进右出的模式保证队列消息的顺序性，并且保存在set中防止重复；对于proxy pool使用sorted set保证有序性以及非重复性。



Redis依赖于链表 字典 跳跃表等数据结构dict Sds ziplist quicklist skiplist

[2]爬取到的数据存储应用：

我们用mongodb数据库对爬取到的数据进行数据存储。使用mongodb-charts进行对存储的数据进行可视化处理。

Mongodb基于key-value进行存储，并且基于json结构，与网络请求返回的数据极其相似，无需过多的转换便可以存储，并且符合面对对象的逻辑，嵌套的kv结构使数据意义明确且编程控制方便

Mongodb依赖于**B-树实现索引**

# 6、系统出错处理

**6.1、 出错信息**

[1]反扒措施导致的报错

[2]Ip被封

[3]Cookies被封（账号被封）

[4]Pool网络接口崩溃

[5]数据库IO错误

[6]自动化操作延时错误

[7]项目部署过程中出现的环境依赖不足问题

[8]APP的https无法有效抓取分析

[9]爬虫过程中可能出现的IOError UrlError HttpError等等错误

**6.2、 补救措施**

[1]基础的HEADERS伪装 包括useragent auth cookies proxy的添加

[2][3]购买大量微博账号，手机网上免费代理，构建自动检测并且动态更新获取提供可用proxy及cookies的获取接口的cookies-pool,proxy-pool，并通过检测reponse响应状态码进行proxycookies切换保证爬取顺利进行

[4]Pool网络借口由于分布式爬虫中的大量访问可能出现响应不及时甚至崩溃现象，在项目后续大规模扩展中可以更换flask为支持异步请求的tarnado实现更快的接口，并且通过nginx反向代理，实现多个API接口服务器的负载均衡技术。

[5]数据库IO错误，检查链接信息，重启数据库，查看数据库日志分析等等

[6]自动化操作中延时导致的问题，通过selenium的Waite部分实现元素加载等待机制，知道元素出现在进行操作，并且手动通过time模块设置延时实现翻页等操作中的认为等待操作。

[7]项目部署的环境依赖问题确实比较严重，但是随着虚拟化技术的发展，容器的诞生极大地简化了这个过程， 通过docker可以实现方便快捷的项目部署，妈妈再也不用担心我的环境安装了，docker在手，开箱即用。

[8]首先在抓包软件端进行设置，其次在安卓模拟器端安装CA证书，由于安卓7.0以上不支持私人安装的CA证书，所以需要使用安卓6.0以下版本，本项目考虑到APP支持的安卓版本兼容问题，使用安卓6.0版本进行

[9]用trycatch语句包装，logging以及ctx模块日志记录错误信息提取，在符合程序主逻辑下的忽略请求，重新请求处理，通过检测异常类型使用pool等方式对异常进行有效的检测处理保证程序的持续运行。

# 7、其他设计以及外来改进方向

**补充设计：**

对于可能出现的验证码识别问题—如传统的数字英文字母验证码

通过Tensorflow训练CNN神经网络进行训练，实现对验证码的识别。

**拓展设计（这是项目未来的发展完善方向）：**

[1]变免费ip为使用ADSL拨号主机的动态ip获取方式

[2]Pool接口的异步网络框架tarnado使用以及nginx反向代理实现负载均衡

[3]用户数据进一步数据分析可视化模块

[4]项目部署可视化模块比如gerapy模块的使用

[5]通过twitter,facebook提供的用户接口实现国外社交网站的用户信息爬取

[6]分布式数据存储hdfs hbase等实现爬取数据可靠并发存储