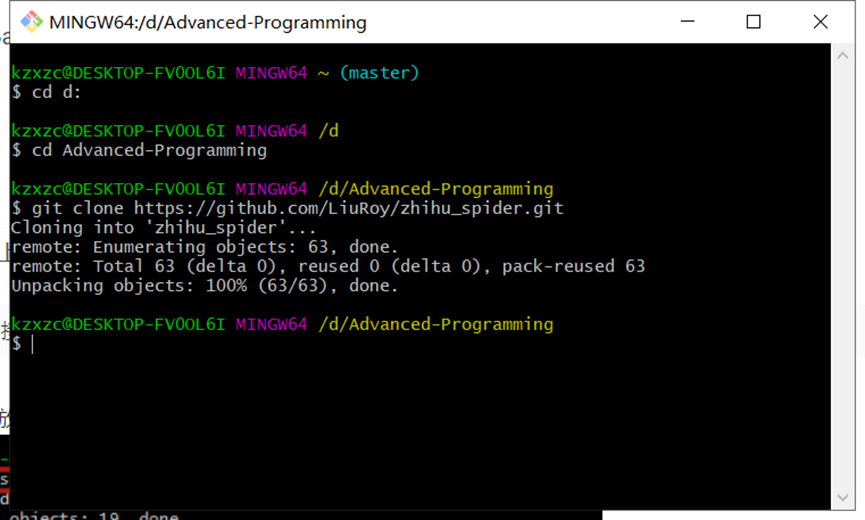
这个项目需要下载的软件和需要设定的环境较多，而且比较令人疑惑。我**分享与项目相关介绍和代码个人解读**。

**Step1: 将整个文件下载到本地**



**Step2：下载会用到的软件**

①MongoDB



②Erlang

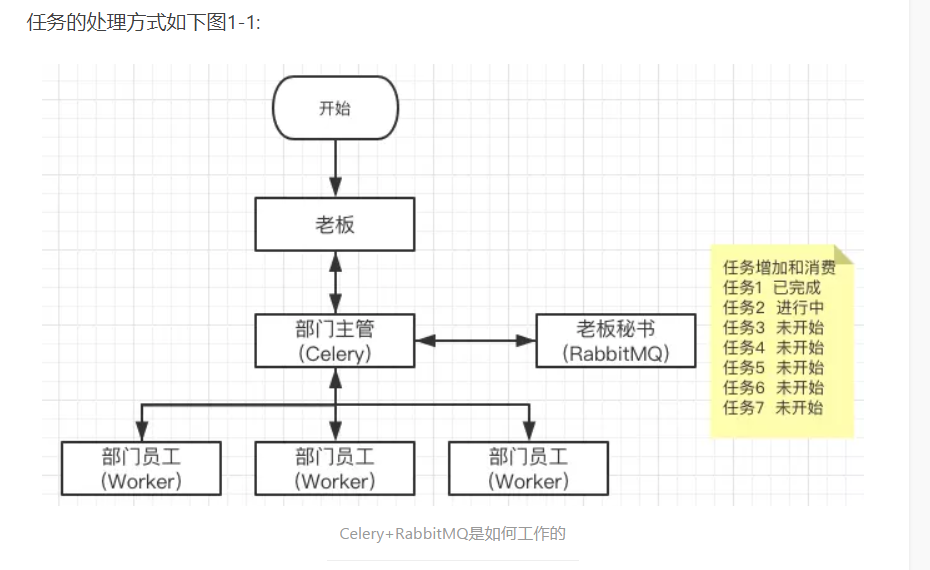
③RabbitMQ（依赖于Erlang，所以要先装②）



④Celery



Celery+RabbitMQ工作模式



在分配爬取任务时需要解决的问题是为每个爬虫分配不重复的爬取任务，Celery+RabbitMQ给出的解决方案是把所有的爬取任务放在一起，并且在获取任务时进行去重。

Celery+RabbitMQ为多个爬虫分配爬取任务的方式是：让所有爬虫（即图上3-1的worker）共享一个存在于RabbitMQ中的请求队列，用来替代各爬虫独立的请求队列，每个爬虫从请求队列中获取爬取任务进行数据采集，Celery是RabbitMQ中任务的生产者，各个爬虫（worker）是任务的消费者。

⑤Docker



分享我曾写的简单爬虫……



https://yz.chsi.com.cn/sch/search.do?ssdm=12&yxls=

**settings.py**

**# profile.py**

import os

import re

import json

from urllib import urlencode

from scrapy import log

from scrapy.spiders import CrawlSpider

from scrapy.selector import **Selector**

from scrapy.http import Request, FormRequest

from **zhihu.items** import ZhihuPeopleItem, ZhihuRelationItem

from **zhihu.constants** import Gender, People, HEADER

class ZhihuSipder(CrawlSpider):

name = "zhihu"

allowed\_domains = ["www.zhihu.com"]

start\_url = "https://www.zhihu.com/people/weizhi-xiazhi"

def \_\_init\_\_(self, \*args, \*\*kwargs):

super(ZhihuSipder, self).\_\_init\_\_(\*args, \*\*kwargs)

self.xsrf = ''

def start\_requests(self):

"""

**登陆页面 获取xrsf**

"""

return [Request(

"https://www.zhihu.com/#signin",

meta={'cookiejar': 1},

callback=self.post\_login

)]

def post\_login(self, response):

"""

解析登陆页面，发送登陆表单

"""

**self.xsrf = Selector(response).xpath(**

**'//input[@name="\_xsrf"]/@value'**

).extract()[0]

return [FormRequest(

'https://www.zhihu.com/login/email',

method='POST',

meta={'cookiejar': response.meta['cookiejar']},

formdata={

'\_xsrf': self.xsrf,

'email': 'xxxxxxxxx',

'password': 'xxxxxxxxx',

'remember\_me': 'true'},

callback=**self.after\_login**

)]

def after\_login(self, response):

"""

登陆完成后从第一个用户开始爬数据

"""

return [Request(

self.start\_url,

meta={'cookiejar': response.meta['cookiejar']},

callback=self.**parse\_people,**

errback=self.parse\_err,

)]

def parse\_people(self, response):

"""

**解析用户主页**

"""

selector = Selector(response)

nickname=selector.xpath(

'//div[@class="title-section ellipsis"]/span[@class="name"]/text()'

).extract\_first()

zhihu\_id=os.path.split(response.url)[-1]

location=selector.xpath(

'//span[@class="**location item**"]/@title'

).extract\_first()

business=selector.xpath(

'//span[@class="**business item**"]/@title'

).extract\_first()

gender = selector.xpath(

'//span[@class="**item gender**"]/i/@class'

).extract\_first()

if gender is not None:

gender = Gender.FEMALE if u'female' in gender else Gender.MALE

employment =selector.xpath(

'//span[@class="**employment item**"]/@title'

).extract\_first()

position = selector.xpath(

'//span[@class="**position item**"]/@title'

).extract\_first()

education = selector.xpath(

'//span[@class="**education-extra item**"]/@title'

).extract\_first()

followee\_count, follower\_count = tuple(selector.xpath(

'//div[@class="zm-profile-side-following zg-clear"]/a[@class="item"]/strong/text()'

).extract())

followee\_count, follower\_count = int(followee\_count), int(follower\_count)

image\_url = selector.xpath(

'//div[@class="body clearfix"]/img/@srcset'

).extract\_first('')[0:-3]

follow\_urls = selector.xpath(

'//div[@class="zm-profile-side-following zg-clear"]/a[@class="item"]/@href'

).extract()

for url in follow\_urls:

complete\_url = 'https://{}{}'.format(self.allowed\_domains[0], url)

yield Request(complete\_url,

meta={'cookiejar': response.meta['cookiejar']},

callback=self**.parse\_follow,**

errback=self.parse\_err)

item = ZhihuPeopleItem(

nickname=nickname,

zhihu\_id=zhihu\_id,

location=location,

business=business,

gender=gender,

employment=employment,

position=position,

education=education,

followee\_count=followee\_count,

follower\_count=follower\_count,

image\_url=image\_url,

)

yield item

def parse\_follow(self, response):

"""

**解析follow数据,解析用户列表**

"""

selector = Selector(response)

people\_links = selector.xpath('//a[@class="zg-link"]/@href').extract()

people\_info = selector.xpath(

'//span[@class="zm-profile-section-name"]/text()').extract\_first()

people\_param = selector.xpath(

'//div[@class="zh-general-list clearfix"]/@data-init').extract\_first()

re\_result = re.search(r'\d+', people\_info) if people\_info else None

people\_count = int(re\_result.group()) if re\_result else len(people\_links)

if not people\_count:

return

people\_param = json.loads(people\_param)

post\_url = 'https://{}/node/{}'.format(

self.allowed\_domains[0], people\_param['nodename'])

# 去请求所有的用户数据

start = 20

while start < people\_count:

payload = {

u'method': u'next',

u'\_xsrf': self.xsrf,

u'params': people\_param[u'params']

}

payload[u'params'][u'offset'] = start

payload[u'params'] = json.dumps(payload[u'params'])

HEADER.update({'Referer': response.url})

start += 20

yield Request(post\_url,

method='POST',

meta={'cookiejar': response.meta['cookiejar']},

headers=HEADER,

body=urlencode(payload),

priority=100,

callback=self.parse\_post\_follow)

**# 请求所有的人**

zhihu\_ids = []

for people\_url in people\_links:

zhihu\_ids.append(os.path.split(people\_url)[-1])

yield Request(people\_url,

meta={'cookiejar': response.meta['cookiejar']},

**callback=self.parse\_people,**

errback=self.parse\_err)

# 返回数据

url, user\_type = os.path.split(response.url)

user\_type = People.Follower if user\_type == u'followers' else People.Followee

item = ZhihuRelationItem(

zhihu\_id=os.path.split(url)[-1],

user\_type=user\_type,

user\_list=zhihu\_ids

)

yield item

def parse\_post\_follow(self, response):

"""

获取动态请求拿到的人员

"""

body = json.loads(response.body)

people\_divs = body.get('msg', [])

# 请求所有的人

zhihu\_ids = []

for div in people\_divs:

selector = Selector(text=div)

link = selector.xpath('//a[@class="zg-link"]/@href').extract\_first()

if not link:

continue

zhihu\_ids.append(os.path.split(link)[-1])

yield Request(link,

meta={'cookiejar': response.meta['cookiejar']},

callback=self.parse\_people,

errback=self.parse\_err)

url, user\_type = os.path.split(response.request.headers['Referer'])

user\_type = People.Follower if user\_type == u'followers' else People.Followee

zhihu\_id = os.path.split(url)[-1]

yield ZhihuRelationItem(

zhihu\_id=zhihu\_id,

user\_type=user\_type,

user\_list=zhihu\_ids,

)

def parse\_err(self, response):

log.ERROR('crawl {} failed'.format(response.url))