

[索引](#)

开源项目

[如何通过 Scrapy + ScrapyWeb 简单高效地部署和监控分布式爬虫项目](#)

[LogParser v0.8.0 发布：一个用于定期增量式解析 Scrapy 爬虫日志的 Python 库，配合 ScrapyWeb 使用可实现爬虫进度可视化](#)

[如何免费创建云端爬虫集群](#)

[时隔五年，Scrapy 终于原生支持 basic auth](#)

JS 分析

***** [\[JS 分析\] 快速定位 JS 代码，还原被混淆压缩的 JS 代码](#)

Scrapy

[Scrapy 隐含 bug: 强制关闭爬虫后从 requests.queue 读取的已保存 request 数量可能有误](#)

[Scrapy 扩展中间件: 针对特定响应状态码，使用代理重新请求](#)

[Scrapy 扩展中间件: 同步/异步提交批量 item 到 MySQL](#)

[scrapy 相关 通过设置 FEED_EXPORT_ENCODING 解决 unicode 中文写入 json 文件出现 '\uXXXX'](#)

[scrapy 通过 FormRequest 模拟登录再继续](#)

Django

[Django DetailView 多重继承 关系整理](#)

Scrapy_redis

[scrapy_redis 相关: 查看保存的数据](#)

[scrapy_redis 相关: 将 jobdir 保存的爬虫进度转移到 Redis](#)

[scrapy_redis 相关: 多线程更新 score/request.priority](#)

Scrapyd

[Scrapyd 改进第一步: Web Interface 添加 charset=UTF-8, 避免查看 log 出现中文乱码](#)

[Scrapyd 改进第二步: Web Interface 添加 STOP 和 START 超链接, 一键调用 Scrapyd API](#)

代理/登录/验证码

[python 代理](#)

[python之cookie, cookiejar 模拟登录绕过验证](#)

[python之基于libsvm识别数字验证码](#)

[python之验证码识别 特征向量提取和余弦相似性比较](#)

网页解析

[lxml.etree.HTML\(text\) 解析HTML文档](#)

[Scrapy Selectors 选择器](#)

[CSS/Xpath 选择器 第几个子节点/父节点/兄弟节点](#)

[BeautifulSoup总结](#)

[pyspider和pyquery总结](#)

编码

[编码 ASCII, GBK, Unicode+utf-8](#)

[requests之headers 'Content-Type': 'text/html'误判encoding为'ISO-8859-1'导致中文text解码错误](#)

阅读:

- [Unicode In Python, Completely Demystified](#)
 - Decode early, Unicode everywhere, encode late
- [立即停止使用 setdefaultencoding\('utf-8'\), 以及为什么](#)
- [也谈 Python 的中文编码处理](#)

- [PYTHON-进阶-编码处理小结](#)

动态加载

[scrapy相关: splash 实践](#)

[scrapy相关: splash安装 A javascript rendering service 渲染](#)

[selenium执行JavaScript语句: 控制滚动条 聚焦元素 改变下拉选项](#)

[python模拟鼠标键盘操作 GhostMouse tinytask 调用外部脚本或程序 autopsy右键另存为](#)

[python 通过js控制滚动条拉取全文 通过psutil获取pid窗口句柄, 通过win32gui使程序窗口前置 通过autopsy实现右键菜单和另存为操作](#)

数据提取

[python之re正则简单够用](#)

[HTML 中的预留字符\(如标签的小于号 <\)必须被替换为字符实体\(< \)。不间断空格\(\)](#)

[HTML转义字符 表示non-breaking space, unicode编码为u'\xa0',超出gbk编码范围?](#)

[python提取网页表格并保存为csv](#)

[python之使用 wkhtmltopdf 和 pdftk 批量加载html生成pdf, 适用于博客备份和官网文档打包](#)

数据库

[python之MySQL MySQLdb 推荐使用姿势, 解决中文乱码](#)

[MongoDB 及 scrapy 应用](#)

posted @ 2017-07-25 12:58 [my8100](#) [阅读\(...\)](#) [评论\(...\)](#) [编辑](#) [收藏](#)


时隔五年，Scrapyd 终于原生支持 basic auth

Issue in 2014

[scrapy/scrapyd/issues/43](https://github.com/scrapy/scrapyd/issues/43)


Authorization? #43


Open mattes opened this issue on Apr 17, 2014 · 15 comments



mattes commented on Apr 17, 2014

Are there any plans concerning authorization of API requests?

 2



pablohoffman commented on Apr 19, 2014

No concrete plans, but I'll gladly accept a pull request that adds HTTP auth!

Pull request in 2019

[scrapy/scrapyd/pull/326](https://github.com/scrapy/scrapyd/pull/326)

Support basic authentication #326


Open my8100 wants to merge 1 commit into scrapy:master from my8100:add_basic_auth

Conversation 4

Commits 1


Checks 0

Files changed 5



my8100 commented 12 days ago

This PR introduces basic authentication, which would be enabled if both `username` and `password` options are set up.

 1

Support basic authentication

my8100 referenced this pull request 12 days ago

Authorization? #43

试用

1. 安装： `pip install -U git+https://github.com/my8100/scrapyd.git@add_basic_auth`
2. 更新配置文件 `scrapyd.conf`，其余配置项详见[官方文档](#)

```
[scrapyd]
username = yourusername
password = yourpassword
```

3. 启动： scrapyd

```
In [1]: import requests

In [2]: requests.get('http://127.0.0.1:6800/').status_code
Out[2]: 401

In [3]: requests.get('http://127.0.0.1:6800/', auth=('admin', 'admin')).status_code
Out[3]: 401

In [4]: requests.get('http://127.0.0.1:6800/', auth=('yourusername', 'yourpassword')).status_code
Out[4]: 200
```

4. 由于 Scrapyd 的 GitHub 最新提交已经[重构了 Jobs 页面](#)，如果正在使用 [ScrapydWeb](#) 管理 Scrapyd，则需同步更新 ScrapydWeb：

```
pip install -U git+https://github.com/my8100/scrapydweb.git
```

[my8100/scrapyd](#)

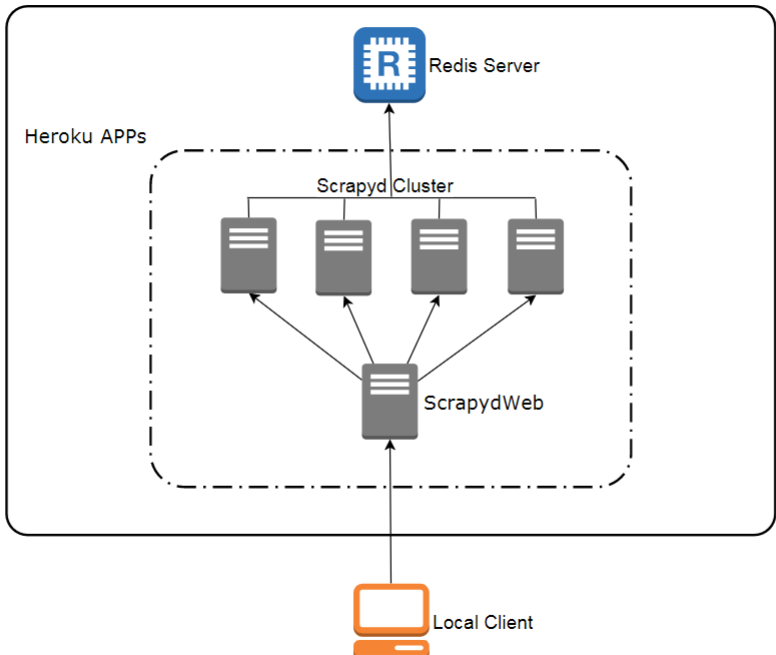
posted @ 2019-05-09 16:14 [my8100](#) [阅读\(...\)](#) [评论\(...\)](#) [编辑](#) [收藏](#)

如何免费创建云端爬虫集群

在线体验

scrapydwweb.herokuapp.com

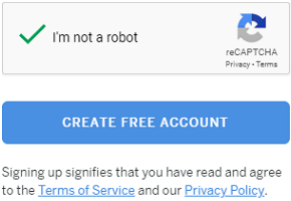
网络拓扑图



注册帐号

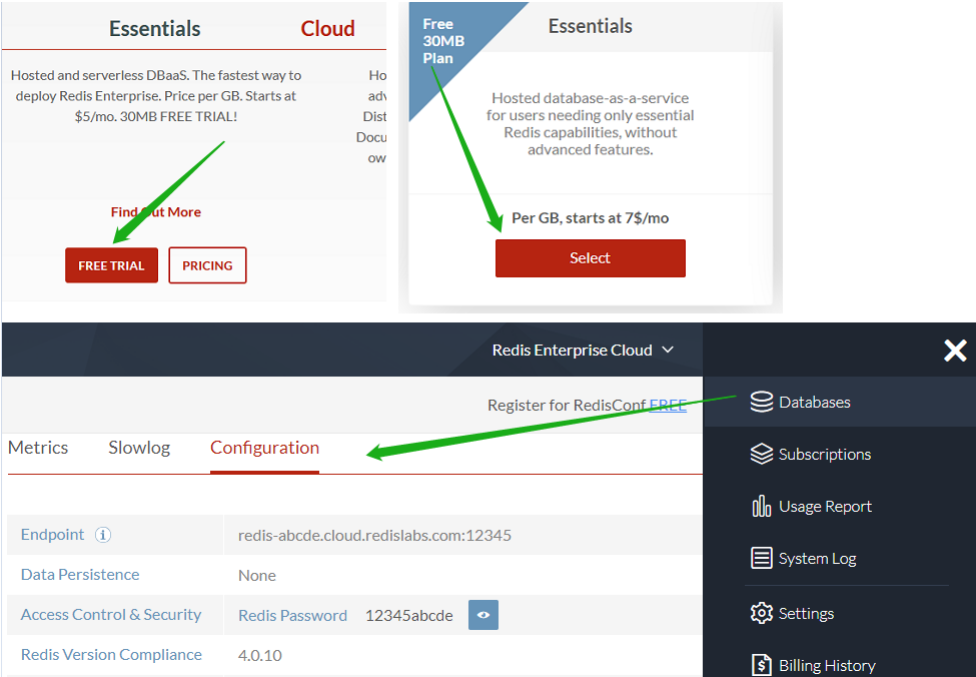
1. Heroku

访问 heroku.com 注册免费账号（注册页面需要调用 google recaptcha 人机验证，登录页面也需要科学地进行上网，访问 app 运行页面则没有该问题），免费账号最多可以创建和运行5个 app。



2. Redis Labs（可选）

访问 redislabs.com 注册免费账号，提供30MB 存储空间，用于下文通过 scrapy-redis 实现分布式爬虫。



通过浏览器部署 Heroku app

1. 访问 [my8100/scrapyd-cluster-on-heroku-scrapyd-app](https://my8100.com/scrapyd-cluster-on-heroku-scrapyd-app) 一键部署 Scrapyd app。（注意更新页面表单中 Redis 服务器的主机，端口和密码）
2. 重复第1步完成4个 Scrapyd app 的部署，假设应用名称为 `svr-1`，`svr-2`，`svr-3` 和 `svr-4`
3. 访问 [my8100/scrapyd-cluster-on-heroku-scrapydweb-app](https://my8100.com/scrapyd-cluster-on-heroku-scrapydweb-app) 一键部署 ScrapydWeb app，取名 `myscrapydweb`
4. （可选）点击 dashboard.heroku.com/apps/myscrapydweb/settings 页面中的 Reveal Config Vars 按钮相应添加更多 Scrapyd server，例如 KEY 为 `SCRAPYD_SERVER_2`，VALUE 为 `svr-2.herokuapp.com:80#group2`
5. 访问 myscrapydweb.herokuapp.com
6. 跳转 [部署和运行分布式爬虫](#) 章节继续阅读。

自定义部署

安装工具

1. [Git](#)
2. [Heroku CLI](#)
3. [Python client for Redis](#): 运行 `pip install redis` 命令即可。

下载配置文件

新开一个命令行提示符：

```
git clone https://github.com/my8100/scrapyd-cluster-on-heroku
cd scrapyd-cluster-on-heroku
```

登录 Heroku

```
heroku login
# outputs:
# heroku: Press any key to open up the browser to login or q to exit:
# Opening browser to https://cli-auth.heroku.com/auth/browser/12345-abcde
# Logging in... done
# Logged in as username@gmail.com
```

创建 Scrapyd 集群

1. 新建 Git 仓库

```
cd scrapyd
git init
# explore and update the files if needed
git status
git add .
git commit -a -m "first commit"
git status
```

2. 部署 Scrapyd app

```
heroku apps:create svr-1
heroku git:remote -a svr-1
git remote -v
git push heroku master
heroku logs --tail
# Press ctrl+c to stop logs outputting
# Visit https://svr-1.herokuapp.com
```

3. 添加环境变量

◦ 设置时区

```
# python -c "import tzlocal; print(tzlocal.get_localzone())"
heroku config:set TZ=Asia/Shanghai
# heroku config:get TZ
```

◦ 添加 Redis 账号（可选，详见 *scrapy_redis_demo_project.zip* 中的 *settings.py*）

```
heroku config:set REDIS_HOST=your-redis-host
heroku config:set REDIS_PORT=your-redis-port
```

```
heroku config:set REDIS_PASSWORD=your-redis-password
```

4. 重复上述第2步和第3步完成余下三个 Scrapyd app 的部署和配置：svr-2，svr-3 和 svr-4

创建 ScrapydWeb app

1. 新建 Git 仓库

```
cd ..  
cd scrapydweb  
git init  
# explore and update the files if needed  
git status  
git add .  
git commit -a -m "first commit"  
git status
```

2. 部署 ScrapydWeb app

```
heroku apps:create myscrapyweb  
heroku git:remote -a myscrapyweb  
git remote -v  
git push heroku master
```

3. 添加环境变量

- 设置时区

```
heroku config:set TZ=Asia/Shanghai
```

- 添加 Scrapyd server（详见 *scrapydweb* 目录下的 *scrapydweb_settings_v8.py*）

```
heroku config:set SCRAPYD_SERVER_1=svr-1.herokuapp.com:80  
heroku config:set SCRAPYD_SERVER_2=svr-2.herokuapp.com:80#group1  
heroku config:set SCRAPYD_SERVER_3=svr-3.herokuapp.com:80#group1  
heroku config:set SCRAPYD_SERVER_4=svr-4.herokuapp.com:80#group2
```

4. 访问 myscrapyweb.herokuapp.com

ScrapydWeb

svr-1.herokuapp.com:80

1 / 4

Overview

Servers

Jobs

Timer Tasks

Operations

Deploy Project

Run Spider

Files

Projects

Logs

Log Parser

Items

System

Settings

Mobile UI

Set 'ENABLE_AUTH = True' to enable basic auth for web UI

Set 'ENABLE_EMAIL = True' to enable email notice

Monitor and control all of your Scrapyd servers.

Deploy Project

Run Spider

Stop Job

List Stats

List Projects

List Versions

List Spiders

List Running Jobs

Delete Version

Delete Project

Multinode Deploy Project

Invert selection

All Columns

Q type to filter

invert filter

4total

4displayed

0selected

<input type="checkbox"/>	Index	Selected	Group	Scrapyd server	Hostname	Status / List info	Pending	Running
<input type="checkbox"/>	1	no		svr-1.herokuapp.com:80	f3cd02bc-4117-4d25-9db0-f74101f4b4de	ok	0	0
<input type="checkbox"/>	2	no	group1	svr-2.herokuapp.com:80	365c1e06-bd17-4903-af2e-bed42df322ee	ok	0	0
<input type="checkbox"/>	3	no	group1	svr-3.herokuapp.com:80	a79e5434-f30f-4fe6-905c-98a2a83525f8	ok	0	0
<input type="checkbox"/>	4	no	group2	svr-4.herokuapp.com:80	d6cf94d5-324e-4def-a1ab-e7ee2aaca45a	ok	0	0

v1.2.0

scrapydweb

logparser

部署和运行分布式爬虫

- 1. 上传 demo 项目，即 `scrapyd-cluster-on-heroku` 目录下的压缩文档 `scrapy_redis_demo_project.zip`
- 2. 将种子请求推入 `mycrawler:start_urls` 触发爬虫并查看结果

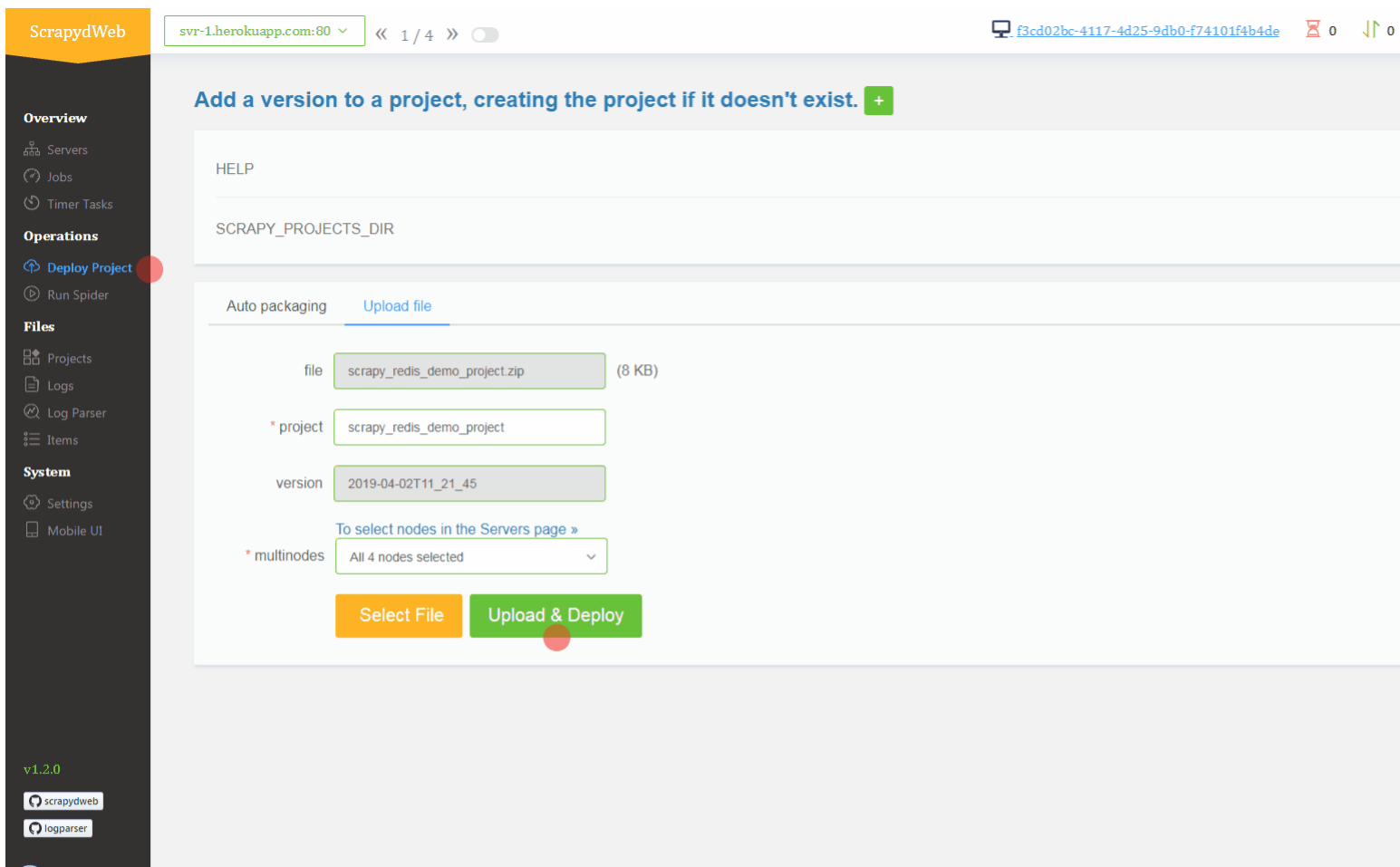
```
In [1]: import redis # pip install redis

In [2]: r = redis.Redis(host='your-redis-host', port=your-redis-port, password='your-redis-password')

In [3]: r.delete('mycrawler_redis:requests', 'mycrawler_redis:dupefilter', 'mycrawler_redis:items')
Out[3]: 0

In [4]: r.lpush('mycrawler:start_urls', 'http://books.toscrape.com', 'http://quotes.toscrape.com')
Out[4]: 2

# wait for a minute
In [5]: r.lrange('mycrawler_redis:items', 0, 1)
Out[5]:
[b{'url': 'http://quotes.toscrape.com/', 'title': 'Quotes to Scrape', 'hostname': 'd6cf94d5-324e-4def-a1ab-e7ee2aaca45a'},
 b{'url': 'http://books.toscrape.com/index.html', 'title': 'All products | Books to Scrape - Sandbox', 'hostname': 'f3cd02bc-4117-4d25-9db0-f74101f4b4de'}]
```



总结

- 优点
 - 免费
 - 可以爬 Google 等外网
 - 可扩展（借助于 [ScrapyWeb](#)）
- 缺点
 - 注册和登录需要科学地进行上网
 - Heroku app 每天至少自动重启一次并且重置所有文件，因此需要外接数据库保存数据，详见 [devcenter.heroku.com](#)

GitHub 开源

[my8100/scrapy-d-cluster-on-heroku](#)

posted @ 2019-04-05 15:10 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

[如何通过 Scrapy + ScrapyWeb 简单高效地部署和监控分布式爬虫项目](#)

[移步 GitHub](#)

posted @ 2019-03-16 22:08 [my8100](#) [阅读\(...\)](#) [评论\(...\)](#) [编辑](#) [收藏](#)

[LogParser v0.8.0 发布：一个用于定期增量式解析 Scrapy 爬虫日志的 Python 库，配合 ScrapydWeb 使用可实现爬虫进度可视化](#)

[移步 GitHub](#)

posted @ 2019-01-24 11:53 [my8100](#) [阅读\(...\)](#) [评论\(...\)](#) [编辑](#) [收藏](#)

Django DetailView 多重继承 关系整理

0.参考

<https://docs.djangoproject.com/en/2.1/topics/class-based-views/mixins/>

1.版本信息

```
In [157]: import sys
```

```
In [158]: sys.version
```

```
Out[158]: '3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:32:41) [MSC v.1900 64 bit (AMD64)]'
```

```
In [159]: import django
```

```
In [160]: django.__version__
```

```
Out[160]: '2.1'
```

```
In [161]: from django.views.generic.detail import DetailView
```

```
In [162]: DetailView.__mro__
```

```
Out[162]:
```

```
(django.views.generic.detail.DetailView,  
 django.views.generic.detail.SingleObjectTemplateResponseMixin,  
 django.views.generic.base.TemplateResponseMixin,  
 django.views.generic.detail.BaseDetailView,  
 django.views.generic.detail.SingleObjectMixin,  
 django.views.generic.base.ContextMixin,  
 django.views.generic.base.View,  
 object)
```

1.多重继承关系和 MRO

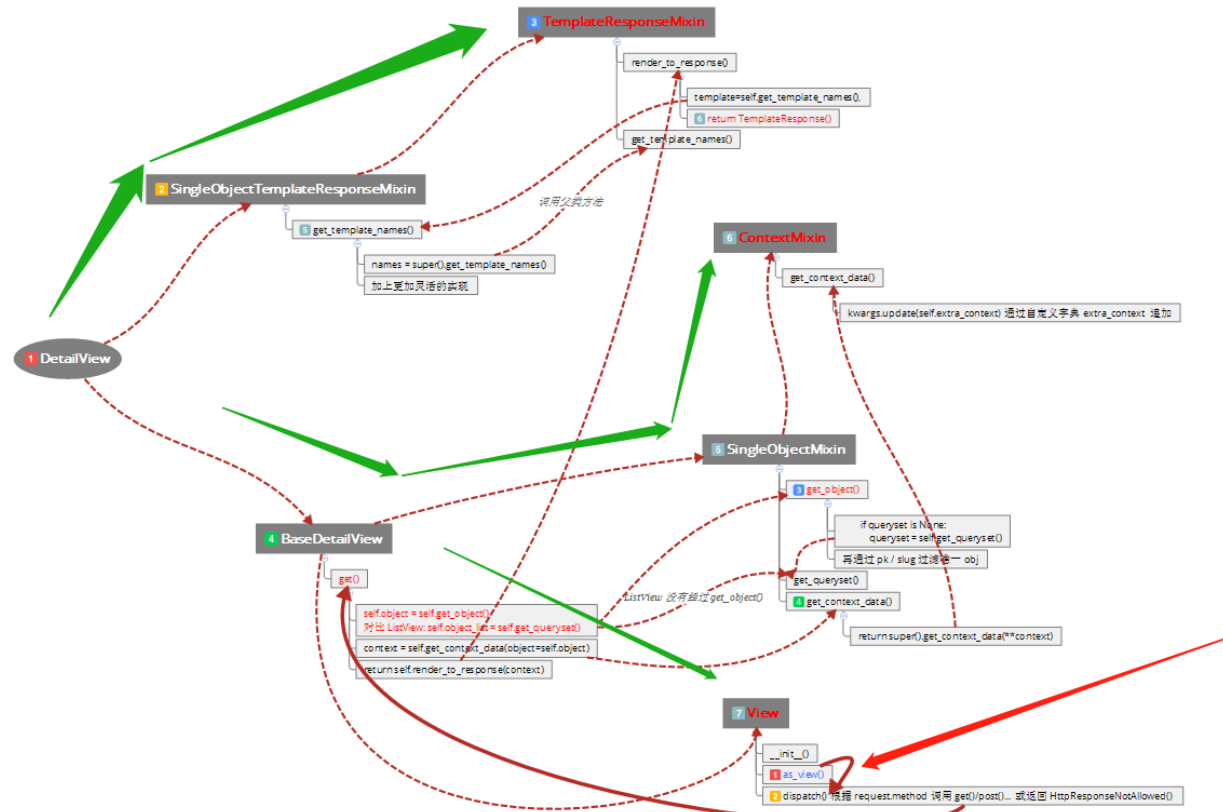
E:\ProgramData\Anaconda3\envs\py3\Lib\site-packages\django\views\generic\detail.py

E:\ProgramData\Anaconda3\envs\py3\Lib\site-packages\django\views\generic\list.py

E:\ProgramData\Anaconda3\envs\py3\Lib\site-packages\django\views\generic\base.py

右下角**红色箭头**为 path 定义的 .as_view() 入口

绿色箭头代表 MRO, 顺序从左上到右下, 由粗到细



[scrapy_redis 相关: 多线程更新 score/request.priority](#)

0.背景

使用 scrapy_redis 爬虫, 忘记或错误设置 request.priority(Rule 也可以通过参数 process_request 设置 request.priority), 导致提取 item 的 request 排在有序集 xxx:requests 的队尾, 持续占用内存。

1.代码实现

遍历 SortedSet 的所有 item 并根据预定义字典对 data 中的 url 进行正则匹配, 更新 score 并复制到临时 newkey, 最后执行 rename

```
# -*- coding: UTF-8 -*-
import sys
import re
from multiprocessing.dummy import Pool as ThreadPool
from functools import partial

try:
    input = raw_input #For py2
except NameError:
    pass

import redis

def print_line(string):
    print('\n{symbol}{space}{string}'.format(symbol='#'*10, space=' '*5, string=string))

def check_key_scores(key):
    try:
        total = redis_server.zcard(key)
    except redis.exceptions.ResponseError:
        print("The value of '{key}' is not a SortedSet".format(key=key))
        sys.exit()
    except Exception as err:
        print(err)
        sys.exit()

    if total == 0:
        print("key '{key}' does not exist or has no items".format(key=key))
        sys.exit()

    __, min_score = redis_server.zrange(key, 0, 0, withscores=True)[0]
    __, max_score = redis_server.zrange(key, -1, -1, withscores=True)[0]

    print('score amount')
    total_ = 0
    # Assuming that score/request.priority is an integer, rather than float number like 1.1
    for score in range(int(min_score), int(max_score)+1):
        count = redis_server.zcount(key, score, score)
        print(score, count)
        total_ += count
    print("{total_}/{total} items of key '{key}' have an integer priority".format(
        total_=total_, total=total, key=key))

def zadd_with_new_score(startstop, total_items):
    data, ori_score = redis_server.zrange(key, startstop, startstop, withscores=True)[0]
    for pattern, score in pattern_score:
        # data eg: b'\\x80\\x02}q\\x00(X\\x03\\x00\\x00\\x00urlq\\x01X\\x13\\x00\\x00\\x00http://httpbi
        # See /site-packages/scrapy_redis/queue.py
        # We don't use zadd method as the order of arguments change depending on
        # whether the class is Redis or StrictRedis, and the option of using
        # kwargs only accepts strings, not bytes.
        m = pattern.search(data.decode('utf-8', 'replace'))
        if m:
            redis_server.execute_command('ZADD', newkey, score, data)
            break
    else:
        redis_server.execute_command('ZADD', newkey, ori_score, data)
    print('{startstop} / {total_items}'.format(
        startstop=startstop+1, total_items=total_items))
```

```

if __name__ == '__main__':

    password = 'password'
    host = '127.0.0.1'
    port = '6379'
    database_num = 0

    key = 'test:requests'
    newkey = 'temp'
    # Request whose url matching any key of keyword_score would be updated with the corresponding value
    # Smaller value/score means higher request.priority
    keyword_score = {'httpbin': -12, 'apps/details': 1}
    pattern_score = [(re.compile(r'url.*?%s.*?callback'%k), v) for (k, v) in keyword_score.items()]

    threads_amount = 10

    redis_server = redis.StrictRedis.from_url('redis://:{password}@{host}:{port}/{database_num}'.format(
        password=password, host=host,
        port=port, database_num=database_num))

    print_line('Step 0: pre check')
    check_key_scores(key)

    print_line('Step 1: copy items and update score')
    # total_items = redis_server.zlexcount(key, '-', '+')
    total_items = redis_server.zcard(key)
    input("Press Enter to copy {total_items} items of '{key}' into '{newkey}' with new score".format(
        total_items=total_items, key=key, newkey=newkey))
    p = ThreadPool(threads_amount)
    p.map(partial(zadd_with_new_score, total_items=total_items), range(total_items))
    p.close()    #Prevents any more tasks from being submitted to the pool. Once all the tasks have been
    p.join()     #Wait for the worker processes to exit. One must call close() or terminate() before using

    # For py3
    # https://stackoverflow.com/questions/5442910/python-multiprocessing-pool-map-for-multiple-argument
    # with ThreadPool(threads_amount) as pool:
    #     # pool.map(partial(zadd_with_new_score, total_items=total_items), range(total_items))
    # print('zadd_with_new_score done')

    print_line('Step 2: check copy result')
    check_key_scores(key)
    check_key_scores(newkey)

    print_line('Step 3: delete, rename and check key')
    input("Press Enter to DELETE '{key}' and RENAME '{newkey}' to '{key}'".format(
        key=key, newkey=newkey))
    print(redis_server.delete(key))
    print(redis_server.rename(newkey, key))
    check_key_scores(key)
    check_key_scores(newkey)

```

2.运行结果


```
PAUSE
##### Step 0: pre check
score amount
-10 10
10/10 items of key 'test:requests' have an integer priority

##### Step 1: copy items and update score
Press Enter to copy 10 items of 'test:requests' into 'temp' with new score
1 / 10
10 / 10
8 / 10
5 / 10
2 / 10
7 / 10
9 / 10
6 / 10
4 / 10
3 / 10

##### Step 2: check copy result
score amount
-10 10
10/10 items of key 'test:requests' have an integer priority
score amount
-12 10
10/10 items of key 'temp' have an integer priority

##### Step 3: delete, rename and check key
Press Enter to DELETE 'test:requests' and RENAME 'temp' to 'test:requests'
1
True
score amount
-12 10
10/10 items of key 'test:requests' have an integer priority
key 'temp' does not exist or has no items

请按任意键继续...
```

Scrapy 扩展中间件: 针对特定响应状态码, 使用代理重新请求

0.参考

<https://doc.scrapy.org/en/latest/topics/downloader-middleware.html#module-scrapy.downloadermiddlewares.redirect>

<https://doc.scrapy.org/en/latest/topics/downloader-middleware.html#module-scrapy.downloadermiddlewares.httpproxy>

1.主要实现

实际爬虫过程中如果请求过于频繁, 通常会被临时重定向到登录页面即302, 甚至是提示禁止访问即403, 因此可以对这些响应执行一次代理请求:

(1) 参考原生 redirect.py 模块, 满足 dont_redirect 或 handle_httpstatus_list 等条件时, 直接传递 response

(2) 不满足条件(1), 如果响应状态码为 302 或 403, 使用代理重新发起请求

(3) 使用代理后, 如果响应状态码仍为 302 或 403, 直接丢弃

2.代码实现

保存至 /site-packages/my_middlewares.py

```
from w3lib.url import safe_url_string
from six.moves.urllib.parse import urljoin

from scrapy.exceptions import IgnoreRequest
```

```
class MyAutoProxyDownloaderMiddleware(object):
```

```
    def __init__(self, settings):
        self.proxy_status = settings.get('PROXY_STATUS', [302, 403])
        # See https://doc.scrapy.org/en/latest/topics/downloader-middleware.html?highlight=proxy#module
        self.proxy_config = settings.get('PROXY_CONFIG', 'http://username:password@some_proxy_server:port')
```

```
    @classmethod
    def from_crawler(cls, crawler):
        return cls(
            settings = crawler.settings
        )
```

```
    # See /site-packages/scrapy/downloadermiddlewares/redirect.py
```

```
    def process_response(self, request, response, spider):
        if (request.meta.get('dont_redirect', False) or
            response.status in getattr(spider, 'handle_httpstatus_list', []) or
            response.status in request.meta.get('handle_httpstatus_list', []) or
            request.meta.get('handle_httpstatus_all', False)):
            return response
```

```
        if response.status in self.proxy_status:
            if 'Location' in response.headers:
                location = safe_url_string(response.headers['location'])
                redirected_url = urljoin(request.url, location)
            else:
                redirected_url = ''
```

```
        # AutoProxy for first time
```

```
        if not request.meta.get('auto_proxy'):
            request.meta.update({'auto_proxy': True, 'proxy': self.proxy_config})
            new_request = request.replace(meta=request.meta, dont_filter=True)
            new_request.priority = request.priority + 2
```

```
            spider.log('Will AutoProxy for <{} {}> {}'.format(
                response.status, request.url, redirected_url))
            return new_request
```

```
        # IgnoreRequest for second time
```

```
        else:
            spider.logger.warn('Ignoring response <{} {}>: HTTP status code still in {} after AutoProxy'
                               .format(response.status, request.url, self.proxy_status))
            raise IgnoreRequest
```

```
return response
```

3.调用方法

(1) 项目 `settings.py` 添加代码，注意必须在默认的 `RedirectMiddleware` 和 `HttpProxyMiddleware` 之间。

```
# Enable or disable downloader middlewares
# See https://doc.scrapy.org/en/latest/topics/downloader-middleware.html
DOWNLOADER_MIDDLEWARES = {
    # 'scrapy.downloadermiddlewares.redirect.RedirectMiddleware': 600,
    'my_middlewares.MyAutoProxyDownloaderMiddleware': 601,
    # 'scrapy.downloadermiddlewares.httpproxy.HttpProxyMiddleware': 750,
}
PROXY_STATUS = [302, 403]
PROXY_CONFIG = 'http://username:password@some_proxy_server:port'
```

4.运行结果

```
2018-07-18 18:42:35 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://httpbin.org/> (referer: None)
2018-07-18 18:42:38 [test] DEBUG: Will AutoProxy for <302 http://httpbin.org/status/302> http://httpbir
2018-07-18 18:42:43 [test] DEBUG: Will AutoProxy for <403 https://httpbin.org/status/403>
2018-07-18 18:42:51 [test] WARNING: Ignoring response <302 http://httpbin.org/status/302>: HTTP status
2018-07-18 18:42:52 [test] WARNING: Ignoring response <403 https://httpbin.org/status/403>: HTTP status
```

代理服务器 log:

```
squid [18/Jul/2018:18:42:53 +0800] "GET http://httpbin.org/status/302 HTTP/1.1" 302 310 "-" "Mozilla/5.
squid [18/Jul/2018:18:42:54 +0800] "CONNECT httpbin.org:443 HTTP/1.1" 200 3560 "-" "-" TCP_TUNNEL:HIER_
```

posted @ 2018-07-18 18:47 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

Scrapy 扩展中间件: 同步/异步提交批量 item 到 MySQL

0.参考

<https://doc.scrapy.org/en/latest/topics/item-pipeline.html?highlight=mongo#write-items-to-mongodb>

20180721新增: 异步版本

<https://twistedmatrix.com/documents/15.3.0/core/howto/rdbms.html>

<https://twistedmatrix.com/documents/18.7.0/api/twisted.python.failure.Failure.html>

<https://twistedmatrix.com/documents/12.1.0/core/howto/time.html>

1.主要实现

- (1) 连接超时自动重连 MySQL server
- (2) 通过 item_list 收集 item, 达到阈值后批量提交至数据库
- (3) 通过解析异常, 自动移除存在异常的数据行, 重新提交 item_list
- (4) shutdown 之前在 close_spider() 中提交当前 item_list
- (5) 20180721新增: 异步版本

2.同步版本

保存至 /site-packages/my_pipelines.py

```
from socket import gethostname
import time
import re
from html import escape

import pymysql
pymysql.install_as_MySQLdb()
from pymysql import OperationalError, InterfaceError, DataError, IntegrityError

class MyMySQLPipeline(object):

    hostname = gethostname()

    def __init__(self, settings):
        self.mysql_host = settings.get('MYSQL_HOST', '127.0.0.1')
        self.mysql_port = settings.get('MYSQL_PORT', 3306)
        self.mysql_user = settings.get('MYSQL_USER', 'username')
        self.mysql_passwd = settings.get('MYSQL_PASSWD', 'password')
        self.mysql_reconnect_wait = settings.get('MYSQL_RECONNECT_WAIT', 60)

        self.mysql_db = settings.get('MYSQL_DB')
        self.mysql_charset = settings.get('MYSQL_CHARSET', 'utf8') #utf8mb4
        self.mysql_item_list_limit = settings.get('MYSQL_ITEM_LIST_LIMIT', 30)

        self.item_list = []

    @classmethod
    def from_crawler(cls, crawler):
        return cls(
            settings = crawler.settings
        )

    def open_spider(self, spider):
        try:
            self.conn = pymysql.connect(
                host = self.mysql_host,
                port = self.mysql_port,
                user = self.mysql_user,
                passwd = self.mysql_passwd,
                db = self.mysql_db,
```

```

        charset = self.mysql_charset,
    )
except Exception as err:
    spider.logger.warn('MySQL: FAIL to connect {} {}'.format(err.__class__, err))
    time.sleep(self.mysql_reconnect_wait)
    self.open_spider(spider)
else:
    spider.logger.info('MySQL: connected')

    self.curs = self.conn.cursor(pymysql.cursors.DictCursor)
    spider.curs = self.curs

def close_spider(self, spider):
    self.insert_item_list(spider)
    self.conn.close()
    spider.logger.info('MySQL: closed')

def process_item(self, item, spider):
    self.item_list.append(item)
    if len(self.item_list) >= self.mysql_item_list_limit:
        self.insert_item_list(spider)

    return item

def sql(self):
    raise NotImplementedError('Subclass of MyMySQLPipeline must implement the sql() method')

def insert_item_list(self, spider):
    spider.logger.info('insert_item_list: {}'.format(len(self.item_list)))
    try:
        self.sql()
    except (OperationalError, InterfaceError) as err:
        # <class 'pymysql.err.OperationalError'>
        # (2013, 'Lost connection to MySQL server during query ([Errno 110] Connection timed out)')
        spider.logger.info('MySQL: exception {} err {}'.format(err.__class__, err))
        self.open_spider(spider)
        self.insert_item_list(spider)
    except Exception as err:
        if len(err.args) == 2 and isinstance(err.args[1], str):
            # <class 'pymysql.err.DataError'>
            # (1264, "Out of range value for column 'position_id' at row 2")
            # <class 'pymysql.err.InternalError'>
            # (1292, "Incorrect date value: '1977-06-31' for column 'release_day' at row 26")
            m_row = re.search(r'at\s+row\s+(\d+)\$', err.args[1])
            # <class 'pymysql.err.IntegrityError'>
            # (1048, "Column 'name' cannot be null") films 43894
            m_column = re.search(r"Column\s'(.+)'", err.args[1])

            if m_row:
                row = m_row.group(1)
                item = self.item_list.pop(int(row) - 1)
                spider.logger.warn('MySQL: {} {} exception from item {}'.format(err.__class__, err,
                    self.insert_item_list(spider)))
            elif m_column:
                column = m_column.group(1)
                item_list = []
                for item in self.item_list:
                    if item[column] == None:
                        item_list.append(item)
                for item in item_list:
                    self.item_list.remove(item)
                    spider.logger.warn('MySQL: {} {} exception from item {}'.format(err.__class__,
                        self.insert_item_list(spider)))
            else:
                spider.logger.error('MySQL: {} {} unhandled exception from item_list: \n{}'.format(
                    err.__class__, err, self.item_list))
        else:
            spider.logger.error('MySQL: {} {} unhandled exception from item_list: \n{}'.format(
                err.__class__, err, self.item_list))
    finally:
        self.item_list.clear()

```

3.调用方法

Scrapy 项目 project_name

MySQL 数据库 database_name， 表 table_name

(1) 项目 pipelines.py 添加代码:

```
from my_pipelines import MyMySQLPipeline

class MySQLPipeline(MyMySQLPipeline):

    def sql(self):
        self.curs.executemany("""
            INSERT INTO table_name (
                position_id, crawl_time)
            VALUES (
                %(position_id)s, %(crawl_time)s)
            ON DUPLICATE KEY UPDATE
                crawl_time=values(crawl_time)
            """, self.item_list)

        self.conn.commit()
```

(2) 项目 settings.py 添加代码:

```
# Configure item pipelines
# See https://doc.scrapy.org/en/latest/topics/item-pipeline.html
ITEM_PIPELINES = {
    # 'project_name.pipelines.ProxyPipeline': 300,
    'project_name.pipelines.MySQLPipeline': 301,
}

MYSQL_HOST = '127.0.0.1'
MYSQL_PORT = 3306
MYSQL_USER = 'username'
MYSQL_PASSWD = 'password'
MYSQL_RECONNECT_WAIT = 60

MYSQL_DB = 'database_name'
MYSQL_CHARSET = 'utf8' #utf8mb4
MYSQL_ITEM_LIST_LIMIT = 3 #100
```

4.运行结果

自动移除存在异常的数据行，重新提交 item_list:

```
2018-07-18 12:35:52 [scrapy.core.scrapers] DEBUG: Scraped from <200 http://httpbin.org/>
{'position_id': 103, 'crawl_time': '2018-07-18 12:35:52'}
2018-07-18 12:35:52 [scrapy.core.scrapers] DEBUG: Scraped from <200 http://httpbin.org/>
{'position_id': None, 'crawl_time': '2018-07-18 12:35:52'}
2018-07-18 12:35:52 [scrapy.core.scrapers] DEBUG: Scraped from <200 http://httpbin.org/>
{'position_id': 104, 'crawl_time': '2018-02-31 17:51:47'}
2018-07-18 12:35:55 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://httpbin.org/> (referer: http:
2018-07-18 12:35:55 [test] INFO: insert_item_list: 3
2018-07-18 12:35:55 [test] WARNING: MySQL: <class 'pymysql.err.IntegrityError'> (1048, "Column 'positio
2018-07-18 12:35:55 [test] INFO: insert_item_list: 2
2018-07-18 12:35:55 [test] WARNING: MySQL: <class 'pymysql.err.InternalError'> (1292, "Incorrect dateti
2018-07-18 12:35:55 [test] INFO: insert_item_list: 1
2018-07-18 12:35:55 [scrapy.core.scrapers] DEBUG: Scraped from <200 http://httpbin.org/>
```

提交结果:

id	position_id	keyword	keywords	crawl_time	date	latitude
4	28 <NULL>		<NULL>	2018-07-18 12:35:48	<NULL>	<NULL>
5	101 <NULL>		<NULL>	2018-07-18 12:35:48	<NULL>	<NULL>
6	20 <NULL>		<NULL>	2018-07-18 12:35:52	<NULL>	<NULL>
7	103 <NULL>		<NULL>	2018-07-18 12:35:52	<NULL>	<NULL>

在 `self.item_list.append(item)` 之后 添加代码 `spider.logger.info('process_item: {}'.format(len(self.item_list)))` 打印添加 item 后的当前 `item_list` 元素个数

连续 yield 5个 item, 累计3个则触发 insert, 红框 insert 部分**将会阻塞**绿框中后续 yield 部分:

```
2018-07-21 13:33:03 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://httpbin.org/> (referer: None)
2018-07-21 13:33:03 [test] INFO: process_item: 1
2018-07-21 13:33:03 [scrapy.core.scrapy] DEBUG: Scraped from (200 http://httpbin.org/)
<position_id': 4, 'crawl_time': '2018-07-21 13:33:03'>
2018-07-21 13:33:03 [test] INFO: process_item: 2
2018-07-21 13:33:03 [scrapy.core.scrapy] DEBUG: Scraped from (200 http://httpbin.org/)
<position_id': 100, 'crawl_time': '2018-07-21 13:33:03'>
2018-07-21 13:33:03 [test] INFO: process_item: 3
2018-07-21 13:33:03 [test] INFO: insert_item_list: 3
2018-07-21 13:33:03 [test] WARNING: MySQL: <class 'pymysql.err.DataError'> (1264, "Out of range value for column 'position_id' at row 3")
2018-07-21 13:33:03 [test] INFO: insert_item_list: 2
2018-07-21 13:33:03 [scrapy.core.scrapy] DEBUG: Scraped from (200 http://httpbin.org/)
<position_id': 120000000, 'crawl_time': '2018-07-21 13:33:03'>
2018-07-21 13:33:03 [test] INFO: process_item: 1
2018-07-21 13:33:03 [scrapy.core.scrapy] DEBUG: Scraped from (200 http://httpbin.org/)
<position_id': None, 'crawl_time': '2018-07-21 13:33:03'>
2018-07-21 13:33:03 [test] INFO: process_item: 2
2018-07-21 13:33:03 [scrapy.core.scrapy] DEBUG: Scraped from (200 http://httpbin.org/)
<position_id': 100, 'crawl_time': '2018-07-21 17:51:47'>
2018-07-21 13:33:03 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://httpbin.org/headers> (referer: http://httpbin.org/)
```

5.异步版本

(1) 保存至 /site-packages/my_pipelines.py

```
# -*- coding: utf-8 -*-
from socket import gethostname
import time
import re

# https://twistedmatrix.com/documents/15.3.0/core/howto/rdbms.html
# twisted.enterprise.adbapi: Twisted RDBMS support
from twisted.enterprise import adbapi
import pymysql
from pymysql import OperationalError, InterfaceError, DataError, InternalError, IntegrityError
```

```
class MyMySQLPipeline(object):

    hostname = gethostname()

    def __init__(self, spider, settings):
        self.spider = spider

        self.dbpool = adbapi.ConnectionPool('pymysql',
            host = settings.get('MYSQL_HOST', '127.0.0.1'),
            port = settings.get('MYSQL_PORT', 3306),
            user = settings.get('MYSQL_USER', 'username'),
            passwd = settings.get('MYSQL_PASSWD', 'password'),
            db = settings.get('MYSQL_DB', 'test'),
            charset = settings.get('MYSQL_CHARSET', 'utf8'), #utf8mb4

            cursorclass = pymysql.cursors.DictCursor
        )

        self.mysql_reconnect_wait = settings.get('MYSQL_RECONNECT_WAIT', 60)
        self.mysql_item_list_limit = settings.get('MYSQL_ITEM_LIST_LIMIT', 30)
        self.item_list = []

    @classmethod
    def from_crawler(cls, crawler):
        return cls(
            spider = crawler.spider,
            settings = crawler.settings
        )

    def close_spider(self, spider):
        self._sql(list(self.item_list))

    def process_item(self, item, spider):
        self.item_list.append(item)

        if len(self.item_list) >= self.mysql_item_list_limit:
            spider.log('item_list: %s'%len(self.item_list))
```

```

        self._sql(list(self.item_list))
        self.item_list.clear()

    return item

def sql(self, txn, item_list):
    raise NotImplementedError('Subclass of MyMySQLPipeline must implement the sql() method')

def _sql(self, item_list, retrying=False):
    d = self.dbpool.runInteraction(self.sql, item_list)
    d.addCallback(self.handle_result, item_list)
    d.addErrback(self.handle_error, item_list, retrying)

def handle_result(self, result, item_list):
    self.spider.logger.info('{} items inserted with retcode {}'.format(len(item_list), result))

def handle_error(self, failure, item_list, retrying):
    # https://twistedmatrix.com/documents/18.7.0/api/twisted.python.failure.Failure.html
    # r = failure.trap(pymysql.err.InternalError)

    args = failure.value.args

    # <class 'pymysql.err.OperationalError'> (1045, "Access denied for user 'username'@'localhost'
    # <class 'pymysql.err.OperationalError'> (2013, 'Lost connection to MySQL server during query
    # <class 'pymysql.err.OperationalError'> (2003, "Can't connect to MySQL server on '127.0.0.1'
    # <class 'pymysql.err.InterfaceError'> (0, '') # after crawl started: sudo service mysqld st
    if failure.type in [OperationalError, InterfaceError]:
        if not retrying:
            self.spider.logger.info('MySQL: exception {} {} \n{}'.format(
                failure.type, args, item_list))
            self.spider.logger.debug('MySQL: Trying to recommit in %s sec'%self.mysql_reconnect_wai

            # self._sql(item_list)
            # https://twistedmatrix.com/documents/12.1.0/core/howto/time.html
            from twisted.internet import task
            from twisted.internet import reactor
            task.deferLater(reactor, self.mysql_reconnect_wait, self._sql, item_list, True)
        else:
            self.spider.logger.warn('MySQL: exception {} {} \n{}'.format(
                failure.type, args, item_list))

    return

    # <class 'pymysql.err.DataError'> (1264, "Out of range value for column 'position_id' at row 2"
    # <class 'pymysql.err.InternalError'> (1292, "Incorrect date value: '1977-06-31' for column 're
    elif failure.type in [DataError, InternalError]:
        m_row = re.search(r'at\s+row\s+(\d+)\$', args[1])
        row = m_row.group(1)
        item = item_list.pop(int(row) - 1)
        self.spider.logger.warn('MySQL: {} {} exception from item {}'.format(failure.type, args, it

        self._sql(item_list)
        return

    # <class 'pymysql.err.IntegrityError'> (1048, "Column 'name' cannot be null") films 43894
    elif failure.type in [IntegrityError]:
        m_column = re.search(r"Column\s'(.+)'", args[1])
        column = m_column.group(1)
        some_items = [item for item in item_list if item[column] is None]
        self.spider.logger.warn('MySQL: {} {} exception from some items: \n{}'.format(
            failure.type, args, some_items))

        self._sql([item for item in item_list if item[column] is not None])
        return

    else:
        self.spider.logger.error('MySQL: {} {} unhandled exception from item_list: \n{}'.format(
            failure.type, args, item_list))

    return

```


(2) 项目 pipelines.py 添加代码: 注意 dbpool.runInteraction 是自动提交的 transaction

```
from my_pipelines import MyMySQLPipeline

class MySQLPipeline(MyMySQLPipeline):

    def sql(self, txn, item_list):
        return txn.executemany("""
            INSERT INTO table_name (
                position_id, crawl_time)
            VALUES (
                %(position_id)s, %(crawl_time)s)
            ON DUPLICATE KEY UPDATE
                crawl_time=values(crawl_time)
            """, item_list)
```

(3) 项目 settings.py

见上文同步版本 3(1)

(4) 运行结果

在 self.item_list.append(item) 之后 添加代码 spider.logger.info('process_item: {}'.format(len(self.item_list))) 打印添加 item 后的当前 item_list 元素个数

连续 yield 5个 item, 累计3个则触发 insert, 红框 insert 部分 **并不会阻塞** 绿框中后续 yield 部分:

```
2018-07-21 13:34:02 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://httpbin.org/> (referer: None)
2018-07-21 13:34:02 [test] INFO: process_item: 1
2018-07-21 13:34:02 [scrapy.core.scraper] DEBUG: Scraped from <200 http://httpbin.org/>
{'position_id': 2, 'crawl_time': '2018-07-21 13:34:02'}
2018-07-21 13:34:02 [test] INFO: process_item: 2
2018-07-21 13:34:02 [scrapy.core.scraper] DEBUG: Scraped from <200 http://httpbin.org/>
{'position_id': 100, 'crawl_time': '2018-07-21 13:34:02'}
2018-07-21 13:34:02 [test] INFO: process_item: 3
2018-07-21 13:34:02 [test] DEBUG: item_list: 3
2018-07-21 13:34:02 [scrapy.core.scraper] DEBUG: Scraped from <200 http://httpbin.org/>
{'position_id': 17000000, 'crawl_time': '2018-07-21 13:34:02'}
2018-07-21 13:34:02 [test] INFO: process_item: 1
2018-07-21 13:34:02 [scrapy.core.scraper] DEBUG: Scraped from <200 http://httpbin.org/>
{'position_id': None, 'crawl_time': '2018-07-21 13:34:02'}
2018-07-21 13:34:02 [test] INFO: process_item: 2
2018-07-21 13:34:02 [scrapy.core.scraper] DEBUG: Scraped from <200 http://httpbin.org/>
{'position_id': 100, 'crawl_time': '2018-02-31 12:51:42'}
2018-07-21 13:34:02 [test] WARNING: MySQL: <Class 'pymysql.err.DataError'> (1264, "Out of range value for column 'position_id' at row 3")
2018-07-21 13:34:02 [test] INFO: 2 items inserted with retcode 4
2018-07-21 13:34:02 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://httpbin.org/> (referer: http://httpbin.org/)
```

另外可见使用了连接池

```
mysql> show processlist;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Id | User | Host | db | Command | Time | State | Info |
+-----+-----+-----+-----+-----+-----+-----+
| 5 | scrapy | 26.114:6403 | lianjia | Sleep | 36 | | NULL |
| 6 | scrapy | 26.114:6428 | test | Query | 0 | starting | show processlist |
| 7 | scrapy | 26.114:6431 | lianjia | Sleep | 0 | | NULL |
| 8 | scrapy | 26.114:6434 | googleplay | Sleep | 69 | | NULL |
| 9 | scrapy | 112.20:6442 | lianjia | Sleep | 66 | | NULL |
| 12 | scrapy | 26.114:6460 | googleplay | Sleep | 12 | | NULL |
| 13 | scrapy | 26.114:6480 | googleplay | Sleep | 128 | | NULL |
+-----+-----+-----+-----+-----+-----+-----+
7 rows in set (0.01 sec)
```

posted @ 2018-07-18 12:55 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

Scrapy 隐含 bug: 强制关闭爬虫后从 requests.queue 读取的已保存 request 数量可能有误

问题描述和解决方案已提交至 **Scrapy issues**:

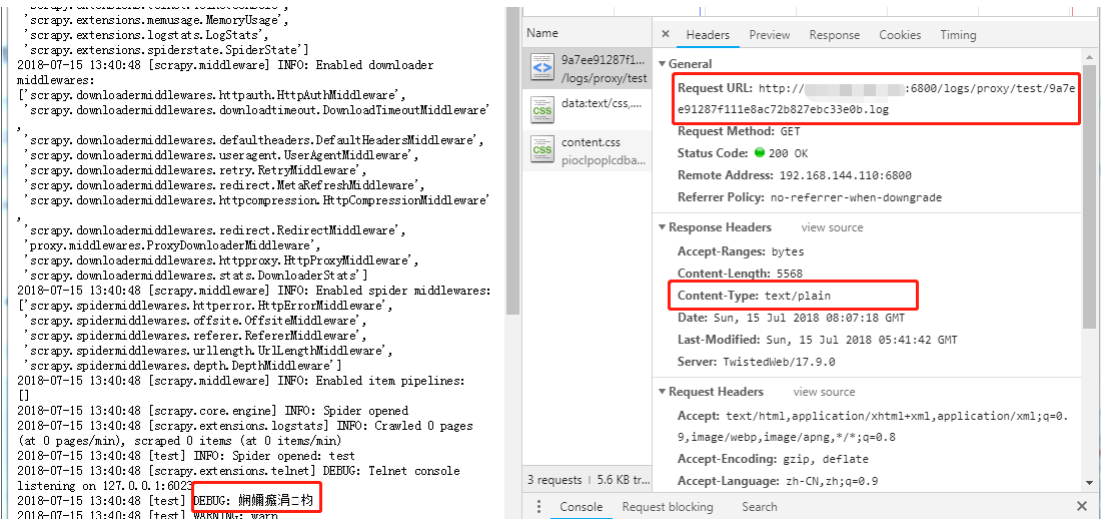
[The size of requests.queue may be wrong when resuming crawl from unclean shutdown. #3333](#)

posted @ 2018-07-16 09:39 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

Scrapy 改进第一步: Web Interface 添加 charset=UTF-8, 避免查看 log 出现中文乱码

0.问题现象和原因

如下图所示, 由于 Scrapy 的 Web Interface 的 log 链接直接指向 log 文件, Response Headers 的 Content-Type 又没有声明字符集 charset=UTF-8, 因此通过浏览器查看 log 会出现非 ASCII 乱码。



1.解决思路

- (1) 如下图所示, 在 Jobs 页面添加带有项目信息的 UTF-8 超链接, 如 http://127.0.0.1:6800/logs/UTF-8.html?project=proxy&spider=test&job=cd2cc82a87f111e8ac72b827ebc33e0b
- (2) 在 Scrapy 的 logs 目录新建 UTF-8.html, 通过 <meta charset="UTF-8"> 声明编码
- (3) 新页面打开超链接后, 通过 JS 获取 url 查询对, 然后更新 UTF-8.html 页面的 iframe 的 src 属性, 如 <iframe src="/logs/proxy/test/9a7ee91287f111e8ac72b827ebc33e0b.log" width="100%" height="100%"></iframe>
- (4) 浏览器自动加载 iframe 获取 log 文件

Jobs

Go back

Project	Spider	Job	PID	Start	Runtime	Finish	Log
Pending							
Running							
	test	cd2cc82a87f111e8ac72b827ebc33e0b	12853	2018-07-15 13:42:07	0:04:57		Log UTF-8
	apps_redis	63a293b687f211e8ac72b827ebc33e0b	12924	2018-07-15 13:46:15	0:00:49		Log UTF-8
	ershoufang	7048fdbcb87f211e8ac72b827ebc33e0b	12939	2018-07-15 13:46:37	0:00:27		Log UTF-8
	films_redis	7729d7dc87f211e8ac72b827ebc33e0b	12949	2018-07-15 13:46:48	0:00:16		Log UTF-8
	jobs	6a0669bc87f211e8ac72b827ebc33e0b	12952	2018-07-15 13:46:52	0:00:12		Log UTF-8
Finished							
	test	9a7ee91287f111e8ac72b827ebc33e0b		2018-07-15 13:40:38	0:01:04	2018-07-15 13:41:42	Log UTF-8

2.修改 Scrapy 代码

/site-packages/scrapy/webinterface.py

改动位置:

- (1) table 添加最后一列, 见红色代码

```
def render(self, txrequest):
    cols = 9 ##### 8
    s = "<html><head><meta charset='UTF-8'><title>Scrapy</title></head>"
    s += "<body>"
    s += "<h1>Jobs</h1>"
    s += "<p><a href='..'>Go back</a></p>"
    s += "<table border='1'>"
```

```

s += "<tr><th>Project</th><th>Spider</th><th>Job</th><th>PID</th><th>Start</th><th>Runtime</th>"
if self.local_items:
    s += "<th>Items</th>"
    #cols = 9 #####
    cols += 1 #####

```

(2) 有两处需要添加 UTF-8 超链接，分别对应 Running 和 Finished，见红色代码

```

s += "<td><a href='/logs/%s/%s/%s.log'>Log</a></td>" % (p.project, p.spider, p.job)
s += "<td><a href='/logs/UTF-8.html?project=%s&spider=%s&job=%s' target='_blank'>UTF-8</a><"

```

(3) 完整代码:

▣

```

from datetime import datetime

import socket

from twisted.web import resource, static
from twisted.application.service import IServiceCollection

from scrapy.utils.misc import load_object

from .interfaces import IPoller, IEggStorage, ISpiderScheduler

from six.moves.urllib.parse import urlparse

class Root(resource.Resource):

    def __init__(self, config, app):
        resource.Resource.__init__(self)
        self.debug = config.getboolean('debug', False)
        self.runner = config.get('runner')
        logsdir = config.get('logs_dir')
        itemsdir = config.get('items_dir')
        local_items = itemsdir and (urlparse(itemsdir).scheme.lower() in ['', 'file'])
        self.app = app
        self.nodename = config.get('node_name', socket.gethostname())
        self.putChild(b'', Home(self, local_items))
        if logsdir:
            self.putChild(b'logs', static.File(logsdir.encode('ascii', 'ignore'), 'text/plain'))
        if local_items:
            self.putChild(b'items', static.File(itemsdir, 'text/plain'))
        self.putChild(b'jobs', Jobs(self, local_items))
        services = config.items('services', ())
        for servName, servClsName in services:
            servCls = load_object(servClsName)
            self.putChild(servName.encode('utf-8'), servCls(self))
        self.update_projects()

    def update_projects(self):
        self.poller.update_projects()
        self.scheduler.update_projects()

    @property
    def launcher(self):
        app = IServiceCollection(self.app, self.app)
        return app.getServiceNamed('launcher')

    @property
    def scheduler(self):
        return self.app.getComponent(ISpiderScheduler)

    @property
    def eggstorage(self):
        return self.app.getComponent(IEggStorage)

    @property
    def poller(self):
        return self.app.getComponent(IPoller)

```

```

class Home(resource.Resource):

    def __init__(self, root, local_items):
        resource.Resource.__init__(self)
        self.root = root
        self.local_items = local_items

    def render_GET(self, txrequest):
        vars = {
            'projects': ', '.join(self.root.scheduler.list_projects())
        }
        s = """
<html>
<head><meta charset='UTF-8'><title>Scrapyd</title></head>
<body>
<h1>Scrapyd</h1>
<p>Available projects: <b>%(projects)s</b></p>
<ul>
<li><a href="/jobs">Jobs</a></li>
""" % vars
        if self.local_items:
            s += '<li><a href="/items/">Items</a></li>'
            s += """
<li><a href="/logs/">Logs</a></li>
<li><a href="http://scrapyd.readthedocs.org/en/latest/">Documentation</a></li>
</ul>

<h2>How to schedule a spider?</h2>

<p>To schedule a spider you need to use the API (this web UI is only for
monitoring)</p>

<p>Example using <a href="http://curl.haxx.se/">curl</a>:</p>
<p><code>curl http://localhost:6800/schedule.json -d project=default -d spider=somespider</code></p>

<p>For more information about the API, see the <a href="http://scrapyd.readthedocs.org/en/latest/">Scra
</body>
</html>
""" % vars
        return s.encode('utf-8')

class Jobs(resource.Resource):

    def __init__(self, root, local_items):
        resource.Resource.__init__(self)
        self.root = root
        self.local_items = local_items

    def render(self, txrequest):
        cols = 9 ##### 8
        s = "<html><head><meta charset='UTF-8'><title>Scrapyd</title></head>"
        s += "<body>"
        s += "<h1>Jobs</h1>"
        s += "<p><a href='../>Go back</a></p>"
        s += "<table border='1'>"
        s += "<tr><th>Project</th><th>Spider</th><th>Job</th><th>PID</th><th>Start</th><th>Runtime</th>"
        if self.local_items:
            s += "<th>Items</th>"
            #cols = 9 #####
            cols += 1 #####
        s += "</tr>"
        s += "<tr><th colspan='%s' style='background-color: #ddd'>Pending</th></tr>" % cols
        for project, queue in self.root.poller.queues.items():
            for m in queue.list():
                s += "<tr>"
                s += "<td>%s</td>" % project
                s += "<td>%s</td>" % str(m['name'])
                s += "<td>%s</td>" % str(m['_job'])
                s += "</tr>"
            s += "<tr><th colspan='%s' style='background-color: #ddd'>Running</th></tr>" % cols
        for p in self.root.launcher.processes.values():
            s += "<tr>"
            for a in ['project', 'spider', 'job', 'pid']:
                s += "<td>%s</td>" % getattr(p, a)
            s += "<td>%s</td>" % p.start_time.replace(microsecond=0)
            s += "<td>%s</td>" % (datetime.now().replace(microsecond=0) - p.start_time.replace(microsec

```

```

        s += "<td></td>"
        s += "<td><a href='/logs/%s/%s/%s.log'>Log</a></td>" % (p.project, p.spider, p.job)
        s += "<td><a href='/logs/UTF-8.html?project=%s&spider=%s&job=%s' target='_blank'>UTF-8</a></td></tr>"
        if self.local_items:
            s += "<td><a href='/items/%s/%s/%s.jl'>Items</a></td>" % (p.project, p.spider, p.job)
        s += "</tr>"
    s += "<tr><th colspan='%s' style='background-color: #ddd'>Finished</th></tr>" % cols
    for p in self.root.launcher.finished:
        s += "<tr>"
        for a in ['project', 'spider', 'job']:
            s += "<td>%s</td>" % getattr(p, a)
        s += "<td></td>"
        s += "<td>%s</td>" % p.start_time.replace(microsecond=0)
        s += "<td>%s</td>" % (p.end_time.replace(microsecond=0) - p.start_time.replace(microsecond=0))
        s += "<td>%s</td>" % p.end_time.replace(microsecond=0)
        s += "<td><a href='/logs/%s/%s/%s.log'>Log</a></td>" % (p.project, p.spider, p.job)
        s += "<td><a href='/logs/UTF-8.html?project=%s&spider=%s&job=%s' target='_blank'>UTF-8</a></td></tr>"
        if self.local_items:
            s += "<td><a href='/items/%s/%s/%s.jl'>Items</a></td>" % (p.project, p.spider, p.job)
        s += "</tr>"
    s += "</table>"
    s += "</body>"
    s += "</html>"

    txrequest.setHeader('Content-Type', 'text/html; charset=utf-8')
    txrequest.setHeader('Content-Length', len(s))

    return s.encode('utf-8')

```

[View Code](#)

3.新建 UTF-8.html 页面

根据 <http://scrapy.readthedocs.io/en/stable/config.html> 确定 Scrapy 所使用的 logs_dir，在该目录下添加如下文件 UTF-8.html

```

<html>
<head><meta charset="UTF-8"></head>
<iframe src="" width="100%" height="100%"></iframe>

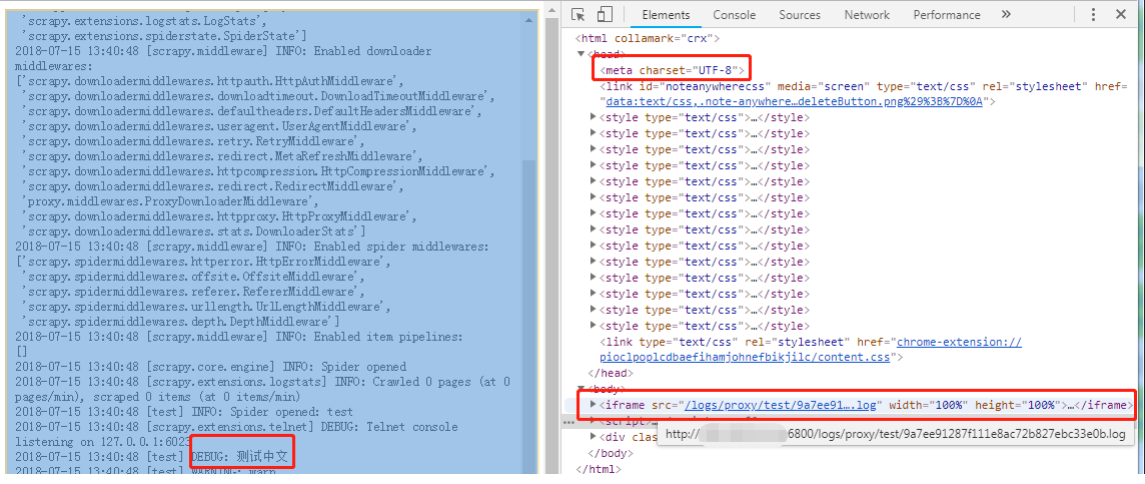
<script>
function parseQueryString(url) {
    var urlParams = {};
    url.replace(
        new RegExp("([^\?=&]+)(=([^\&]*))?", "g"),
        function($0, $1, $2, $3) {
            urlParams[$1] = $3;
        }
    );
    return urlParams;
}

var kwargs = parseQueryString(location.search);
document.querySelector('iframe').src = "/logs/" + kwargs.project + "/" + kwargs.spider + "/" + kwargs.job + ".log"
</script>

</html>

```

4.实现效果



posted @ 2018-07-15 16:18 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

scrapy_redis 相关: 将 jobdir 保存的爬虫进度转移到 Redis

0.参考

[Scrapy 隐含 bug: 强制关闭爬虫后从 requests.queue 读取的已保存 request 数量可能有误](#)

1.说明

Scrapy 设置 jobdir, 停止爬虫后, 保存文件目录结构:

```
crawl/apps/
├── requests.queue
│   ├── active.json
│   ├── p0
│   └── p1
├── requests.seen
└── spider.state
```

requests.queue/p0 文件保存 priority=0 的未调度 request, **p-1** 对应实际 priority=1 的高优先级 request, 转移到 redis 有序集合时, score 值越小排序越靠前, 因此取 score 为 -1。以此类推, p1 对应 priority=-1 的低优先级 request。

requests.seen 保存请求指纹过滤器对已入队 request 的 hash 值, 每行一个值。

spider.state 涉及自定义属性的持久化存储, 不在本文处理范围以内。

2.实现代码

```
import os
from os.path import join
import re
import struct

import redis

def sadd_dupefilter(jobdir, redis_server, name):
    """See python/lib/site-packages/scrapy/dupefilters.py"""

    file = join(jobdir, 'requests.seen')
    with open(file) as f:
        print('Processing %s, it may take minutes...' % file)
        key = '%s:dupefilter' % name
        for x in f:
            redis_server.sadd(key, x.rstrip())
    print('Result: {} {}'.format(key, redis_server.scard(key)))

def zadd_requests(jobdir, redis_server, name):
    """See python/lib/site-packages/queuelib/queue.py"""

    SIZE_FORMAT = ">L"
    SIZE_SIZE = struct.calcsize(SIZE_FORMAT)

    key = '%s:requests' % name
    queue_dir = join(jobdir, 'requests.queue')
    file_list = os.listdir(queue_dir)
    file_score_dict = dict([(f, int(f[1:])) for f in file_list
                             if re.match(r'^p-?\d+$', f)])

    for (file, score) in file_score_dict.items():
        print('Processing %s, it may take minutes...' % file)
        f = open(join(queue_dir, file), 'rb+')
        qsize = f.read(SIZE_SIZE)
        total_size, = struct.unpack(SIZE_FORMAT, qsize)
        f.seek(0, os.SEEK_END)

        actual_size = 0
        while True:
            if f.tell() == SIZE_SIZE:
                break
            f.seek(-SIZE_SIZE, os.SEEK_CUR)
            size, = struct.unpack(SIZE_FORMAT, f.read(SIZE_SIZE))
            f.seek(-size-SIZE_SIZE, os.SEEK_CUR)
            data = f.read(size)
            redis_server.execute_command('ZADD', key, score, data)
            f.seek(-size, os.SEEK_CUR)
```



```
        actual_size += 1
    print('total_size {}, actual_size {}, score {}'.format(
        total_size, actual_size, score))
    print('Result: {} {}'.format(key, redis_server.zlexcount(key, '-', '+')))
```

```
if __name__ == '__main__':
    name = 'test'
    jobdir = '/home/yourproject/crawl/apps'
    database_num = 0
    # apps/
    #   requests.queue
    #   active.json
    #   p0
    #   p1
    #   requests.seen
    #   spider.state

    password = 'password'
    host = '127.0.0.1'
    port = '6379'
    redis_server = redis.StrictRedis.from_url('redis://:{password}@{host}:{port}/{database_num}'.format(
        password=password, host=host,
        port=port, database_num=database_num))

    sadd_dupefilter(jobdir, redis_server, name)
    zadd_requests(jobdir, redis_server, name)
```

3.运行结果

```
Processing . apps/requests.seen, it may take minutes...
Result: test:dupefilter 303850
Processing p0, it may take minutes...
total_size 233490, actual_size 233490, score 0
Result: test:requests 233490
Processing p1, it may take minutes...
total_size 17, actual_size 17, score 1
Result: test:requests 233507
```

posted @ 2018-07-11 19:07 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

[lxml.etree.HTML\(text\) 解析HTML文档](#)

0.参考

<http://lxml.de/tutorial.html#the-xml-function>

There is also a corresponding function `HTML()` for HTML literals.

```
>>> root = etree.HTML("<p>data</p>")
>>> etree.tostring(root)
b'<html><body><p>data</p></body></html>'
```

1.基本用法

```
from lxml import etree
# Parses an HTML document from a string constant. Returns the root nodd
root = etree.HTML(r.text) #<Element html at 0x7bb8208>
```

1.1 xpath 和 cssselect 获取文字和属性

```
In [83]: for item in root.xpath('//button')[:1]:
...:     print(item)
...:     print(item.text)
...:     print(item.xpath('./@id'))
...:
<Element button at 0x84277c8>
Requests Generator
['btn_requests']
###
In [84]: for item in root.cssselect('button')[:1]:
...:     print(item)
...:     print(item.text)
...:     print(item.cssselect('::attr(id)'))
...:
...:
<Element button at 0x84277c8>
Requests Generator
ExpressionError: Pseudo-elements are not supported.
###
In [92]: for item in root.cssselect('button')[:1]:
...:     print(item.get('id', ''))
...:
btn_requests
###
In [93]: for item in root.cssselect('button')[:1]:
...:     print(item.xpath('./@id'))
...:
['btn_requests']
```

1.2 美化打印

```
print(etree.tostring(root, pretty_print=True).decode('utf-8')) # 美化打印
# You can also serialise to a Unicode string without declaration by
# passing the ``unicode`` function as encoding (or ``str`` in Py3),
# or the name 'unicode'. This changes the return value from a byte
# string to an unencoded unicode string.
print(etree.tostring(root, encoding=str, pretty_print=True)) #py3 使之返回 text
print(etree.tostring(root, encoding=unicode, pretty_print=True)) #py2 使之返回 unicode
```

1.3 自动补全

```
In [109]: rt = etree.HTML('<html><p>123</p></html>') #自动补全
In [110]: print(etree.tostring(rt, encoding=str, pretty_print=True))
<html>
  <body>
    <p>123</p>
  </body>
</html>
```

1.4 fromstring 不支持残缺片段，不会自动补全

```
In [115]: rt = etree.fromstring('<html><p>456</html>') #fromstring 不支持残缺片段，不会自动补全
XMLSyntaxError: Opening and ending tag mismatch: p line 1 and html, line 1, column 20
In [116]: rt = etree.fromstring('<html><p>456</p></html>')
In [117]: print(etree.tostring(rt, encoding=str, pretty_print=True))
<html>
  <p>456</p>
</html>
```

.

posted @ 2018-06-01 16:38 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

CSS/Xpath 选择器 第几个子节点/父节点/兄弟节点

0.参考

1.初始化

```
In [325]: from scrapy import Selector
```

```
In [326]: text="""
...: <div>
...:     <a>1a</a>
...:     <p>2p</p>
...:     <p>3p</p>
...: </div>"""
```

```
In [327]: sel=Selector(text=text)
```

```
In [328]: print(sel.extract())
```

```
<html><body><div>
  <a>1a</a>
  <p>2p</p>
  <p>3p</p>
</div></body></html>
```

2.Xpath 父节点/上一个下一个兄弟节点

```
In [329]: sel.xpath('//a/parent::*</p>').extract()
```

```
Out[329]: ['<p>2p</p>', '<p>3p</p>']
```

```
In [330]: sel.xpath('//p/preceding-sibling::a').extract()
```

```
Out[330]: ['<a>1a</a>']
```

```
In [331]: sel.xpath('//a/following-sibling::p').extract()
```

```
Out[331]: ['<p>2p</p>', '<p>3p</p>']
```

3.CSS 第几个子节点

3.1 通用

#完整子节点列表，从第一个子节点开始计数，并且满足子节点tag限定

```
In [332]: sel.css('a:nth-child(1)').extract()
```

```
Out[332]: ['<a>1a</a>']
```

#完整子节点列表，从最后一个子节点开始计数，并且满足子节点tag限定

```
In [333]: sel.css('a:nth-last-child(1)').extract()
```

```
Out[333]: []
```

```
In [334]: sel.css('p:nth-child(1)').extract()
```

```
Out[334]: []
```

```
In [335]: sel.css('p:nth-child(2)').extract()
```

```
Out[335]: ['<p>2p</p>']
```

```
In [336]: sel.css('p:nth-child(3)').extract()
```

```
Out[336]: ['<p>3p</p>']
```

```
In [337]: sel.css('p:nth-last-child(1)').extract()
```

```
Out[337]: ['<p>3p</p>']
```

```
In [338]: sel.css('p:nth-last-child(2)').extract()
```

```
Out[338]: ['<p>2p</p>']
```

```
In [339]: sel.css('p:nth-last-child(3)').extract()
```

```
Out[339]: []
```

3.2 特别指代

```
In [340]: sel.css('a:first-child').extract()
```

```
Out[340]: ['<a>1a</a>']
```

```
In [341]: sel.css('a:last-child').extract()
```

```
Out[341]: []
```

```
In [342]: sel.css('p:first-child').extract()
```

```
Out[342]: []
```

```
In [343]: sel.css('p:last-child').extract()
```

```
Out[343]: ['<p>3p</p>']
```

3.3 上述 -child 修改为 -of-type，仅对 过滤后的相应子节点列表 进行计数

4.Xpath 第几个子节点

```
In [344]: sel.xpath('//div').extract()
```

```
Out[344]: ['<div>\n      <a>1a</a>\n      <p>2p</p>\n      <p>3p</p>\n</div>']
```

```
In [345]: sel.xpath('//div/*').extract()
```

```
Out[345]: ['<a>1a</a>', '<p>2p</p>', '<p>3p</p>']
```

```
In [346]: sel.xpath('//div/node()').extract()
```

```
Out[346]: ['\n      ', '<a>1a</a>', '\n      ', '<p>2p</p>', '\n      ', '<p>3p</p>', '\n']
```

```
In [347]: sel.xpath('//div/a').extract()
```

```
Out[347]: ['<a>1a</a>']
```

```
In [348]: sel.xpath('//div/p').extract()
```

```
Out[348]: ['<p>2p</p>', '<p>3p</p>']
```

```
In [349]:
```

```
In [349]: sel.xpath('//div/a[1]').extract()
```

```
Out[349]: ['<a>1a</a>']
```

```
In [350]: sel.xpath('//div/a[last()]').extract()
```

```
Out[350]: ['<a>1a</a>']
```

```
In [351]:
```

```
In [351]: sel.xpath('//div/p[1]').extract()      #相当于过滤后的子节点列表
```

```
Out[351]: ['<p>2p</p>']
```

```
In [352]: sel.xpath('//div/p[last()]').extract()
```

```
Out[352]: ['<p>3p</p>']
```

```
In [353]: sel.xpath('//div/p[last()-1]').extract()
```

```
Out[353]: ['<p>2p</p>']
```

```
In [354]:
```

```
In [354]: sel.xpath('//div/*[1]').extract()      #完整子节点列表
```

```
Out[354]: ['<a>1a</a>']
```

```
In [355]: sel.xpath('//div/*[last()]').extract()
```

```
Out[355]: ['<p>3p</p>']
```

```
In [356]:
```

```
In [356]: sel.xpath('//div/node()[1]').extract() #包括纯文本
```

```
Out[356]: ['\n      ']
```

```
In [357]: sel.xpath('//div/node()[last()]').extract()
```

```
Out[357]: ['\n']
```

posted @ 2018-05-03 12:13 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

[scrapy_redis 相关: 查看保存的数据](#)

0.参考资料

<https://redis.io/topics/data-types-intro> An introduction to Redis data types and abstractions

<http://redisdoc.com/> Redis 命令参考

1.scrapy_redis

2.redis-cli 查看数据

2.1 匹配数据库内所有 key

redis-cli

```
127.0.0.1:6379> KEYS *
1) "mycrawler_redis:dupefilter"
2) "mycrawler_redis:requests"
6) "mycrawler_redis:items"
```

2.2 List (列表)

```
127.0.0.1:6379> type mycrawler_redis:items
list
127.0.0.1:6379> llen mycrawler_redis:items
(integer) 701
127.0.0.1:6379> LRANGE mycrawler_redis:items 0 1
1) "{\"text\": \"\\u201cA woman is like a tea bag; you never know how strong it is until it's in hot water.\\u201d\", \"crawled\": \"2018-02-21 03:38:17\", \"spider\": \"mycrawler_redis\", \"author\": \"Eleanor Roosevelt\"}"
2) "{\"text\": \"\\u201cThe world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.\\u201d\", \"crawled\": \"2018-02-21 03:38:17\", \"spider\": \"mycrawler_redis\", \"author\": \"Albert Einstein\"}"
127.0.0.1:6379> LRANGE mycrawler_redis:items -2 -1
1) "{\"text\": \"\\u201cThe opposite of love is not hate, it's indifference. The opposite of art is not ugliness, it's indifference. The opposite of faith is not heresy, it's indifference. And the opposite of life is not death, it's indifference.\\u201d\", \"crawled\": \"2018-02-21 03:43:34\", \"spider\": \"mycrawler_redis\", \"author\": \"Elie Wiesel\"}"
2) "{\"text\": \"\\u201cIt is not a lack of love, but a lack of friendship that makes unhappy marriages.\\u201d\", \"crawled\": \"2018-02-21 03:43:34\", \"spider\": \"mycrawler_redis\", \"author\": \"Friedrich Nietzsche\"}"
```

2.3 Set (集合)

PS: size是容量，但cardinality是「基数」，是集合论中的术语

```
127.0.0.1:6379> type mycrawler_redis:dupefilter
set
127.0.0.1:6379> SCARD mycrawler_redis:dupefilter
(integer) 18603
127.0.0.1:6379> SRANDMEMBER mycrawler_redis:dupefilter
"5faa874e145528c84d636d5a95959583301e18f2"
127.0.0.1:6379> SRANDMEMBER mycrawler_redis:dupefilter
"68f9f6842efcd0392236b953ba6cf5c4616d4c91"
```

2.4 SortedSet (有序集合)

20180726 更新: 也可通过 ZCARD key命令返回有序集 key 的基数。

```
127.0.0.1:6379> type mycrawler_redis:requests
zset
127.0.0.1:6379> ZLEXCOUNT mycrawler_redis:requests - +
(integer) 18199
127.0.0.1:6379> ZRANGE mycrawler_redis:requests 0 1 WITHSCORES
1) "\x80\x02;q\x01(U\x04bodyq\x02U\x00U\t_encodingq\x03U\x05utf-8q\x04U\acookiesq\x05;q\x06U\x04metaq\aq\b(U\x05depthq\tK\x02U\tlink_textq\nc\xml.etree\n_ElementStringResult\nq\x0bU\x0cspiritualityq\x0c\x8https://www.goodreads.com/quotesq\x19asU\x03urlq\x1aX1\x00\x00\x00https://www.goodreads.com/quotes/tag/spiritualityU\x0bdont_filterq\x1b\x89U\x06methodq!U\x03GETq!\"U\acrnbackq#Nu."
2) "0"
3) "\x80\x02;q\x01(U\x04bodyq\x02U\x00U\t_encodingq\x03U\x05utf-8q\x04U\acookiesq\x05;q\x06U\x04metaq\aq\b(U\x05depthq\tK\x02U\tlink_textq\nc\xml.etree\n_ElementStringResult\nq\x0bU\rrChoice
```


Awardsq\x0c\x85\x81q\r;q\x0e(U\ a_ parentq\x0fN\x0cis_ attributeq\x10\x89U\battnameq\x11N\ais_ textq\x12\x89U\ais_ tailq\x13\x89ubU\x04ruleq\x14\x89ubU\x04ruleq\x15\x89ubU\x04ruleq\x16\x89ubU\x04ruleq\x17\x89ubU\x04ruleq\x18\x89ubU\x04ruleq\x19\x89ubU\x04ruleq\x1a\x89ubU\x04ruleq\x1b\x89ubU\x04ruleq\x1c\x89ubU\x04ruleq\x1d\x89ubU\x04ruleq\x1e\x89ubU\x04ruleq\x1f\x89ubU\x04ruleq\x20\x89ubU\x04ruleq\x21\x89ubU\x04ruleq\x22\x89ubU\x04ruleq\x23\x89ubU\x04ruleq\x24\x89ubU\x04ruleq\x25\x89ubU\x04ruleq\x26\x89ubU\x04ruleq\x27\x89ubU\x04ruleq\x28\x89ubU\x04ruleq\x29\x89ubU\x04ruleq\x2a\x89ubU\x04ruleq\x2b\x89ubU\x04ruleq\x2c\x89ubU\x04ruleq\x2d\x89ubU\x04ruleq\x2e\x89ubU\x04ruleq\x2f\x89ubU\x04ruleq\x30\x89ubU\x04ruleq\x31\x89ubU\x04ruleq\x32\x89ubU\x04ruleq\x33\x89ubU\x04ruleq\x34\x89ubU\x04ruleq\x35\x89ubU\x04ruleq\x36\x89ubU\x04ruleq\x37\x89ubU\x04ruleq\x38\x89ubU\x04ruleq\x39\x89ubU\x04ruleq\x3a\x89ubU\x04ruleq\x3b\x89ubU\x04ruleq\x3c\x89ubU\x04ruleq\x3d\x89ubU\x04ruleq\x3e\x89ubU\x04ruleq\x3f\x89ubU\x04ruleq\x40\x89ubU\x04ruleq\x41\x89ubU\x04ruleq\x42\x89ubU\x04ruleq\x43\x89ubU\x04ruleq\x44\x89ubU\x04ruleq\x45\x89ubU\x04ruleq\x46\x89ubU\x04ruleq\x47\x89ubU\x04ruleq\x48\x89ubU\x04ruleq\x49\x89ubU\x04ruleq\x4a\x89ubU\x04ruleq\x4b\x89ubU\x04ruleq\x4c\x89ubU\x04ruleq\x4d\x89ubU\x04ruleq\x4e\x89ubU\x04ruleq\x4f\x89ubU\x04ruleq\x50\x89ubU\x04ruleq\x51\x89ubU\x04ruleq\x52\x89ubU\x04ruleq\x53\x89ubU\x04ruleq\x54\x89ubU\x04ruleq\x55\x89ubU\x04ruleq\x56\x89ubU\x04ruleq\x57\x89ubU\x04ruleq\x58\x89ubU\x04ruleq\x59\x89ubU\x04ruleq\x5a\x89ubU\x04ruleq\x5b\x89ubU\x04ruleq\x5c\x89ubU\x04ruleq\x5d\x89ubU\x04ruleq\x5e\x89ubU\x04ruleq\x5f\x89ubU\x04ruleq\x60\x89ubU\x04ruleq\x61\x89ubU\x04ruleq\x62\x89ubU\x04ruleq\x63\x89ubU\x04ruleq\x64\x89ubU\x04ruleq\x65\x89ubU\x04ruleq\x66\x89ubU\x04ruleq\x67\x89ubU\x04ruleq\x68\x89ubU\x04ruleq\x69\x89ubU\x04ruleq\x6a\x89ubU\x04ruleq\x6b\x89ubU\x04ruleq\x6c\x89ubU\x04ruleq\x6d\x89ubU\x04ruleq\x6e\x89ubU\x04ruleq\x6f\x89ubU\x04ruleq\x70\x89ubU\x04ruleq\x71\x89ubU\x04ruleq\x72\x89ubU\x04ruleq\x73\x89ubU\x04ruleq\x74\x89ubU\x04ruleq\x75\x89ubU\x04ruleq\x76\x89ubU\x04ruleq\x77\x89ubU\x04ruleq\x78\x89ubU\x04ruleq\x79\x89ubU\x04ruleq\x7a\x89ubU\x04ruleq\x7b\x89ubU\x04ruleq\x7c\x89ubU\x04ruleq\x7d\x89ubU\x04ruleq\x7e\x89ubU\x04ruleq\x7f\x89ubU\x04ruleq\x80\x89ubU\x04ruleq\x81\x89ubU\x04ruleq\x82\x89ubU\x04ruleq\x83\x89ubU\x04ruleq\x84\x89ubU\x04ruleq\x85\x89ubU\x04ruleq\x86\x89ubU\x04ruleq\x87\x89ubU\x04ruleq\x88\x89ubU\x04ruleq\x89\x89ubU\x04ruleq\x8a\x89ubU\x04ruleq\x8b\x89ubU\x04ruleq\x8c\x89ubU\x04ruleq\x8d\x89ubU\x04ruleq\x8e\x89ubU\x04ruleq\x8f\x89ubU\x04ruleq\x90\x89ubU\x04ruleq\x91\x89ubU\x04ruleq\x92\x89ubU\x04ruleq\x93\x89ubU\x04ruleq\x94\x89ubU\x04ruleq\x95\x89ubU\x04ruleq\x96\x89ubU\x04ruleq\x97\x89ubU\x04ruleq\x98\x89ubU\x04ruleq\x99\x89ubU\x04ruleq\x9a\x89ubU\x04ruleq\x9b\x89ubU\x04ruleq\x9c\x89ubU\x04ruleq\x9d\x89ubU\x04ruleq\x9e\x89ubU\x04ruleq\x9f\x89ubU\x04ruleq\xa0\x89ubU\x04ruleq\xa1\x89ubU\x04ruleq\xa2\x89ubU\x04ruleq\xa3\x89ubU\x04ruleq\xa4\x89ubU\x04ruleq\xa5\x89ubU\x04ruleq\xa6\x89ubU\x04ruleq\xa7\x89ubU\x04ruleq\xa8\x89ubU\x04ruleq\xa9\x89ubU\x04ruleq\xaa\x89ubU\x04ruleq\xab\x89ubU\x04ruleq\xac\x89ubU\x04ruleq\xad\x89ubU\x04ruleq\xae\x89ubU\x04ruleq\xaf\x89ubU\x04ruleq\xb0\x89ubU\x04ruleq\xb1\x89ubU\x04ruleq\xb2\x89ubU\x04ruleq\xb3\x89ubU\x04ruleq\xb4\x89ubU\x04ruleq\xb5\x89ubU\x04ruleq\xb6\x89ubU\x04ruleq\xb7\x89ubU\x04ruleq\xb8\x89ubU\x04ruleq\xb9\x89ubU\x04ruleq\xba\x89ubU\x04ruleq\xbb\x89ubU\x04ruleq\xbc\x89ubU\x04ruleq\xbd\x89ubU\x04ruleq\xbe\x89ubU\x04ruleq\xbf\x89ubU\x04ruleq\xc0\x89ubU\x04ruleq\xc1\x89ubU\x04ruleq\xc2\x89ubU\x04ruleq\xc3\x89ubU\x04ruleq\xc4\x89ubU\x04ruleq\xc5\x89ubU\x04ruleq\xc6\x89ubU\x04ruleq\xc7\x89ubU\x04ruleq\xc8\x89ubU\x04ruleq\xc9\x89ubU\x04ruleq\xca\x89ubU\x04ruleq\xcb\x89ubU\x04ruleq\xcc\x89ubU\x04ruleq\xcd\x89ubU\x04ruleq\xce\x89ubU\x04ruleq\xcf\x89ubU\x04ruleq\xd0\x89ubU\x04ruleq\xd1\x89ubU\x04ruleq\xd2\x89ubU\x04ruleq\xd3\x89ubU\x04ruleq\xd4\x89ubU\x04ruleq\xd5\x89ubU\x04ruleq\xd6\x89ubU\x04ruleq\xd7\x89ubU\x04ruleq\xd8\x89ubU\x04ruleq\xd9\x89ubU\x04ruleq\xda\x89ubU\x04ruleq\xdb\x89ubU\x04ruleq\xdc\x89ubU\x04ruleq\xde\x89ubU\x04ruleq\xdf\x89ubU\x04ruleq\xe0\x89ubU\x04ruleq\xe1\x89ubU\x04ruleq\xe2\x89ubU\x04ruleq\xe3\x89ubU\x04ruleq\xe4\x89ubU\x04ruleq\xe5\x89ubU\x04ruleq\xe6\x89ubU\x04ruleq\xe7\x89ubU\x04ruleq\xe8\x89ubU\x04ruleq\xe9\x89ubU\x04ruleq\xea\x89ubU\x04ruleq\xeb\x89ubU\x04ruleq\xec\x89ubU\x04ruleq\xed\x89ubU\x04ruleq\xee\x89ubU\x04ruleq\xef\x89ubU\x04ruleq\xf0\x89ubU\x04ruleq\xf1\x89ubU\x04ruleq\xf2\x89ubU\x04ruleq\xf3\x89ubU\x04ruleq\xf4\x89ubU\x04ruleq\xf5\x89ubU\x04ruleq\xf6\x89ubU\x04ruleq\xf7\x89ubU\x04ruleq\xf8\x89ubU\x04ruleq\xf9\x89ubU\x04ruleq\xfa\x89ubU\x04ruleq\xfb\x89ubU\x04ruleq\xfc\x89ubU\x04ruleq\xfd\x89ubU\x04ruleq\xfe\x89ubU\x04ruleq\xff\x89ubU\x04ruleq

https://www.goodreads.com/quotesq\x19asU\x03urlq\x1aX&\x00\x00\x00https://www.goodreads.com/choiceawardsU\x0bdont_filterq\x1b\x89U\bpriorityU\x06methodq!\U\x03GETq!\U\ aerrbackq#Nu."

4) "0"

127.0.0.1:6379> **ZRANGE mycrawler_redis:requests -2 -1 WITHSCORES**

1) "\x80\x02;q\x01(U\x04bodyq\x02U\x00U\t_encodingq\x03U\x05utf-8q\x04U\acookiesq\x05;q\x06U\x04metaq\ a;q\b(U\tlink_ textq\tX\x00\x00\x00\x00U\x04ruleq\nK\x00U\x10download_timeoutq\x0bG@f\x80\x00\x00\x00U\x12;q\x13U\x02enq\x14aU\ aRefererq\x15;q\x16U\x17https://scrapinghub.comq\x17aU\x0fAccept-Encodingq\x18;q\x19U\x0cgzip,deflateq\x1aaU\x06Acceptq\x1b;q\x1cU?text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8q\x1daU\nUser-Agentq\x1e;q\x1fU7scrapy-redis (+https://github.com/rolando/scrapy-redis)qauU\x03urlq!X#\x00\x00\x00https://www.youtube.com/scrapinghubU\x0bdont_filterq!\x88U\bpriorityq#J\xff\xff\xff\xffU\bcallbackq\$U\x14_response_2) "1"

3) "\x80\x02;q\x01(U\x04bodyq\x02U\x00U\t_encodingq\x03U\x05utf-8q\x04U\acookiesq\x05;q\x06U\x04metaq\ a;q\b(U\tlink_ textq\tX\x00\x00\x00\x00U\x04ruleq\nK\x00U\x10download_timeoutq\x0bG@f\x80\x00\x00\x00U\x12;q\x13U\x02enq\x14aU\ aRefererq\x15;q\x16U\x17https://scrapinghub.comq\x17aU\x0fAccept-Encodingq\x18;q\x19U\x0cgzip,deflateq\x1aaU\x06Acceptq\x1b;q\x1cU?text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8q\x1daU\nUser-Agentq\x1e;q\x1fU7scrapy-redis (+https://github.com/rolando/scrapy-redis)qauU\x03urlq!X%\x00\x00\x00https://www.facebook.com/ScrapingHub/U\x0bdont_filterq!\x88U\bpriorityq#J\xff\xff\xff\xffU\bcallbackq\$U\x14_respon4) "1"

scrapy 通过FormRequest模拟登录再继续

1.参考

https://doc.scrapy.org/en/latest/topics/spiders.html#scrapy.spiders.Spider.start_requests

自动提交 login.php 返回表单

<https://doc.scrapy.org/en/latest/topics/request-response.html#using-formrequest-from-response-to-simulate-a-user-login>

2.模拟登录雪球

```
# -*- coding: utf-8 -*-
import os
import scrapy
from scrapy.shell import inspect_response

# https://doc.scrapy.org/en/latest/topics/spiders.html start_requests() 章节

class LoginSpider(scrapy.Spider):
    name = 'login'
    allowed_domains = ['xueqiu.com']
    # start_urls = ['http://xueqiu.com/'] #The default implementation generates Request(url, dont_filter=True)

    url_login = 'https://xueqiu.com/snowman/login',
    url_somebody = 'https://xueqiu.com/u/6146070786'
    data_dict = {
        'remember_me': 'true',
        # 'username': 'fake', #返回200 {"error_description":"用户名或密码错误","error_uri":"/provider/oauth/token?error=invalid_grant"}
        'username': os.getenv('xueqiu_username'),
        'password': os.getenv('xueqiu_password'),
    }

    def start_requests(self):
        return [scrapy.FormRequest(url = self.url_login,
                                   headers={'X-Requested-With': 'XMLHttpRequest'}, #否则404将导致退出,并返回{"error_description":"用户名或密码错误","error_uri":"/provider/oauth/token?error=invalid_grant"}
                                   meta={'proxy': 'http://127.0.0.1:8888'}, #否则fiddler导致返回缓慢
                                   formdata = self.data_dict,
                                   callback=self.logged_in)]

    def logged_in(self, response):
        # inspect_response(response, self)
        assert os.getenv('xueqiu_nickname') in response.text #AssertionError 将导致退出
        return scrapy.Request(self.url_somebody, dont_filter=True, meta={'proxy': 'http://127.0.0.1:8888'})

    def parse(self, response):
        # inspect_response(response, self)
        self.log(os.getenv('xueqiu_nickname') in response.text)
```

posted @ 2017-12-27 16:14 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

python2 python3 转换, 兼容

0.

1.参考

<https://docs.python.org/3/library/urllib.html>

`urllib` is a package that collects several modules for working with URLs:

- [urllib.request](#) for opening and reading URLs
- [urllib.error](#) containing the exceptions raised by [urllib.request](#)
- [urllib.parse](#) for parsing URLs
- [urllib.robotparser](#) for parsing `robots.txt` files

[HOWTO Fetch Internet Resources Using The urllib Package](#)

[Python同时兼容python2和python3的8个技巧分享](#)

```
D:\Python36\Tools\scripts>python3 2to3.py "G:\xxx.py"
```

仅输出变化, 加 `-w` 则备份和生成替换文件

<https://pythonhosted.org/six/>

<https://docs.python.org/3/howto/pyporting.html> Porting Python 2 Code to Python 3

google: `urllib compatible python3`

[Cheat Sheet: Writing Python 2-3 compatible code](#)

[Supporting Python 2 and 3 without 2to3 conversion](#)

[Targeting python 2 and 3 at the same time.](#)

2.

3.

posted @ 2017-12-26 16:57 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

[nginx 限制并发访问及请求频率](#)

0.

1.参考

[【工作】Nginx限制IP并发连接数和请求数的研究](#)

[Module ngx_http_limit_conn_module](#)

[Module ngx_http_limit_req_module](#)

[漏桶算法和 NGINX 的 limit_req 模块](#)

漏桶这个名字，其实就非常形象的描述了算法本身的原理。大家都知道，一个身上打了 n 个眼儿的桶，无论你倒进桶里的水多还是少，漏出来的水的流速也会保持稳定，这就是此算法的本质。再以 NGINX + PHP-FPM 为例，我们在 NGINX 配置里定义一个最大处理请求的速度，如果 PHP-FPM 的稳定处理速度峰值是 1000 RPS，那就在 NGINX 里定义处理请求速度最大为 1000 RPS。当 RPS 已经大于这个值的时候，多出来的请求就被 NGINX 这个桶暂时储存起来，排着队等待处理。在 NGINX 的精心照料下，PHP-FPM 会相对稳定的处理来自 NGINX 的请求，而不会出现突然暴增的请求让 PHP-FPM 处理不过来，甚至挂掉。

然而桶也有大小，NGINX 也一样，假如请求太多太多，桶都装不下了，那么桶将会把多出来的请求直接漏掉，返回 503 错误。

[php-fpm 与 Nginx优化总结](#)

最大请求数max_requests

最长执行时间request_terminate_timeout

2.vi /etc/nginx/nginx.conf

在 http{} 添加：

```
## 2017-12-18 【工作】Nginx限制IP并发连接数和请求数的研究 http://www.jiagoumi.com/work/718.html
#调整为1, nginx -t 检查, 再reload, 查看 error.log
#默认为503 Service Unavailable, 由于临时的服务器维护或者过载,
#可以429 Too Many Requests 用户在给定的时间内发送了太多的请求

# [error] limiting connections by zone "conn_ip"
limit_conn_zone $binary_remote_addr zone=conn_ip:10m;
limit_conn conn_ip 10; #如果这里设置为限制1个ip只能1个连接, log 显示 req_freq_ip 相关控制信息

# [error] limiting connections by zone "conn_server"
limit_conn_zone $server_name zone=conn_server:10m;
limit_conn conn_server 1000;

limit_req_zone $binary_remote_addr zone=req_freq_ip:10m rate=3r/s;
#这样相当于容量为0, 都会被503, 没有意义
#limit_req zone=req_freq_ip; #By default, the maximum burst size is equal to zero

#超过每秒3个请求, 放入最多容纳10个的缓冲区, 或者理解为10个令牌?
#超过3.x, 则log显示 [warn] delaying; 超过10.x, 则log显示 [error] limiting, 会被503
limit_req zone=req_freq_ip burst=10;

#后面加 nodelay 则未超过10.x的拿到令牌的请求不会被延迟
#limit_req zone=req_freq_ip burst=10 nodelay;

limit_req_status 429;

limit_conn_log_level error; #info | notice | warn | error 不支持 debug
```

3.这里打开一个网页，实际上建立了多个连接，其中也包括连接复用。

Source port	Info
53694	GET /uninum/ HTTP/1.1
53694	GET /uninum/static/css/app.0a79551cb9bde65b86ae3c40db7fea9c.css HTTP/1.1
53693	GET /uninum/static/js/manifest.e4600e5493b15575fc55.js HTTP/1.1
53866	GET /uninum/static/js/vendor.f5d39a0c9f706a328c24.js HTTP/1.1
53908	GET /uninum/static/js/app.ec70eb387fa5f91aa026.js HTTP/1.1
53694	GET /uninum/static/js/manifest.e4600e5493b15575fc55.js.map HTTP/1.1
53908	GET /uninum/static/css/app.0a79551cb9bde65b86ae3c40db7fea9c.css.map HTTP/1.1
53866	GET /uninum/static/js/vendor.f5d39a0c9f706a328c24.js.map HTTP/1.1
53695	GET /uninum/static/js/app.ec70eb387fa5f91aa026.js.map HTTP/1.1

posted @ 2017-12-18 15:22 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

MySQL 查询性能优化相关

0.

1.参考

[提升网站访问速度的 SQL 查询优化技巧](#)

缓存一切数据，读取内存而不是硬盘IO

如果你的服务器默认情况下没有使用MySQL查询缓存，那么你应该开启缓存。开启缓存意味着MySQL 会把所有的语句和语句执行的结果保存下来，如果随后有一条与缓存中完全相同的语句需要执行，那么MySQL 就会返回缓存的结果。缓存不会过时，因为MySQL 会在表数据更新后刷新缓存。

[MySQL缓存--服务器缓存query cache](#)

<https://dba.stackexchange.com/questions/109030/cant-enable-mysql-5-6-query-cache>

Look for the [mysqld] group header in my.cnf and put those lines under it

```
[mysqld]
query_cache_type = 1
query_cache_size = 4096M
query_cache_limit = 2M
query_cache_strip_comments =1
```

Every change to a table requires scanning the 4GB to purge entries for that table. 一旦数据更新，会清空内存的所有缓存并更新，所以应该合理设置缓存总大小 query_cache_size

2.MySQL 查询缓存

当前设置

```
mysql> show variables like "query_cache%";
```

```
+-----+-----+
```

```
| Variable_name | Value |
```

```
+-----+-----+
```

```
| query_cache_limit | 1048576 | #单条最大 1MB，1000条数据返回45KB，这里设置为 51200 比较合适
```

```
| query_cache_min_res_unit | 4096 | #100条数据返回4.8KB，这里设置为 5120 比较合适，不够会再申请一块
```

```
| query_cache_size | 1048576 | #总共分配1MB，修改为 52428800，即50MB，能
```

够缓存 1w 个 100条查询结果。

| query_cache_type | OFF | #修改为 1 或 ON 缓存除了以 SELECT SQL_NO_CACHE 开头的所有查询结果。另一个选项是 2 或 DEMAND 只缓存以 SELECT SQL_CACHE 开头的查询结果。

| query_cache_wlock_invalidate | OFF | #表锁定时认为缓存不可用，修改为 ON
+-----+-----+
5 rows in set (0.01 sec)

mysql> select @@query_cache_type;
+-----+
| @@query_cache_type |
+-----+
| OFF |
+-----+
1 row in set, 1 warning (0.00 sec)

mysql> set @@query_cache_type=ON;
ERROR 1651 (HY000): Query cache is disabled; restart the server with query_cache_type=1 to enable it
mysql> exit;

修改缓存设置 vi /etc/my.cnf 重启 service mysqld restart

query_cache_limit = 50K
query_cache_min_res_unit = 5K
query_cache_size = 50M
query_cache_type = 1
query_cache_wlock_invalidate = ON

查询缓存状态

mysql> show status like "Qcache%";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Qcache_free_blocks | 1 | #太多碎片，最小分配内存单位设置不合理？
| Qcache_free_memory | 52365352 |
| Qcache_hits | 0 | #命中缓存的查询次数
| Qcache_inserts | 12 | #插入次数
| Qcache_lowmem_prunes | 0 | #总缓存空间不足？
| Qcache_not_cached | 0 |
| Qcache_queries_in_cache | 12 | #现有缓存个数
| Qcache_total_blocks | 26 |
+-----+-----+
8 rows in set (0.00 sec)

设置 query_cache_min_res_unit 为 5K 或 10K 下面结果都是1： 2，所以还是设置为 5K？ ？ ？

```
Qcache_queries_in_cache | 1152 |  
| Qcache_total_blocks | 2307 |
```

3.

posted @ 2017-12-15 17:19 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

[PHP查询数据库较慢, nginx 超时 返回 504: Sorry, the page you are looking for is currently unavailable.](#)

现象:

PHP查询数据库较慢, 大约 60s 后 nginx 返回 504: Sorry, the page you are looking for is currently unavailable.

检查log:

从 /etc/nginx/nginx.conf 找到 /var/log/nginx/access.log 和 /var/log/nginx/error.log

log 显示 upstream timed out (110: Connection timed out) while reading response header from upstream

解决办法:

[Nginx upstream timed out \(why and how to fix\)](#)

添加带有注释的语句

```
location ~ /\.php$ {
#    root            html;
    fastcgi_read_timeout 300; #####https://distinctplace.com/2017/04/22/nginx-upstream-timed-c
    fastcgi_pass      127.0.0.1:9000;
    fastcgi_index      index.php;
    fastcgi_param      SCRIPT_FILENAME    $document_root$fastcgi_script_name;
    include             fastcgi_params;
}
```

执行 service nginx reload 之后, 执行查询, 返回页面没有显示 table, 继续检查log

"PHP message: PHP Warning: file_get_contents(http://127.0.0.1/xxx): failed to open stream: HTTP request failed! in /home/www/uninum/web.php on line 50

PHP message: PHP Notice: Trying to get property of non-object in xxx.php on line 54" while reading response header from upstream

同步设置 file_get_contents 超时

[php file_get_contents 获取文件超时的处理方法](#) 以及post 和 多次重试

```
$opts = array(
    'http'=>array(
        'method'=>"GET",
        'timeout'=>300,
    ) );
$context = stream_context_create($opts);
$html =file_get_contents('http://www.example.com', false, $context);
```

posted @ 2017-12-13 18:13 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

编码 ASCII, GBK, Unicode+utf-8

0.

1.参考

[网页编码就是那点事](#)

[阮一峰 字符编码笔记：ASCII, Unicode 和 UTF-8](#)

2.总结

美国 ASCII 码 发音： [/ˈæski/](#)：128个字符，只占用了一个字节的后面7位，最前面的一位统一规定为0。

非 ASCII 编码：

- 欧洲：ISO 8859-1，又称Latin-1或“西欧语言”，以ASCII为基础，利用一个字节的的最前面1位，加入了96个字母和符号。
- 中国：gb2312 >>> gbk（2万多，支持繁体，1字节或双字节）>>> gb18030

全球大一统 Unicode编码系统：可分为编码方式和实现方式两个层次。

- 编码空间有100多万个码位 code point 可用来映射字符，已收录十万个字符
- 存储和传输的具体实现方式称为Unicode转换格式（Unicode Transformation Format，简称为UTF）
 - 从字节串判断若干个字节表示对应一个字符，如何节省空间） utf-8 变长编码（1字节或大部分3字节） 或 utf-16（2或4字节，不兼容ASCII编码）

3.资料

<https://zh.wikipedia.org/wiki/%E9%80%9A%E7%94%A8%E5%AD%97%E7%AC%A6%E9%9B%86>

通用字符集（英语：Universal Character Set, UCS）

表示一个UCS或Unicode值的十六进制数通常在前面加上“U+”，例如“U+0041”代表字符“A”。

<https://zh.wikipedia.org/wiki/Unicode>

Unicode编码系统可分为编码方式和实现方式两个层次。

统一码以一种抽象的方式（即数字）来处理字符，并将视觉上的演绎工作（例如字体大小、外观形状、字体形态、文体等）留给其他软件来处理，例如网页浏览器或是文字处理器。

在基本多文种平面（英文：Basic Multilingual Plane，简写BMP。又称为“零号平面”、plane 0）里的所有字符，要用四个数字（即两个char,16bit,例如U+4AE0，共支持六万多个字符）；在零号平面以外的字符则需要使用五个或六个数字。

一个字符的Unicode编码是确定的。但是在实际传输过程中，出于节省空间的目的，对Unicode编码的实现方式有所不同。Unicode的实现方式称为Unicode转换格式（Unicode Transformation Format，简称为UTF）

UTF-8编码，这是一种变长编码

此外Unicode的实现方式还包括UTF-7、Punycode、CESU-8、SCSU、UTF-32、GB18030等，这些实现方式有些仅在一定的国家和地区使用，有些则属于未来的规划方式。目前通用的实现方式是UTF-16小端序（LE）、UTF-16大端序（BE）和UTF-8。在微软公司Windows XP附带的记事本（Notepad）中，“另存为”对话框可以选择的四种编码方式除去非Unicode编码的ANSI（对于英文系统即ASCII编码，中文系统则为GB2312或Big5编码）外，其余三种为“Unicode”（对应UTF-16 LE）、“Unicode big endian”（对应UTF-16 BE）和“UTF-8”。

目前辅助平面的工作主要集中在第二和第三平面的中日韩统一表意文字中，因此包括GBK、GB18030、Big5等简体中文、繁体中文、日文、韩文以及越南喃字的各种编码与Unicode的协调性被重点关注。考虑到Unicode最终要涵盖所有的字符。从某种意义上而言，这些编码方式也可视作Unicode的出现于其之前的既成事实的实现方式，如同ASCII及其扩展Latin-1一样，后两者的字符在16位Unicode编码空间中的编码第一字节各位全为0，第二字节编码与原编码完全一致。但上述东亚语言编码与Unicode编码的对应关系要复杂得多。

UTF-8就是以8位为单元对UCS进行编码

提到“Unicode定义的区域，U+0000到U+10FFFF”，（注：自行折算为1114111，100多万。）

- **128个US-ASCII字符只需一个字节编码**（Unicode范围由U+0000至U+007F）。
- 其他基本多文种平面（BMP）中的字符（这包含了大部分常用字，如大部分的汉字）**使用三个字节编码**（Unicode范围由U+0800至U+FFFF）。

Unicode 和 UTF-8 之间的转换关系表 (x 字符表示码点占据的位)									
码点的位数	码点起值	码点终值	字节序列	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
7	U+0000	U+007F	1	0xxxxxxx					
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx				
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx			
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx		

Unicode的编码空间从U+0000到U+10FFFF，共有1,112,064个码位（code point）可用来映射字符。

因为这个字超过U+FFFF所以无法用UCS-2的格式编码

16进制编码范围	UTF-16表示方法（二进制）	10进制码范围	字节数量
U+0000---U+FFFF	xxxxxxxx xxxxxxxx yyyyyyyy yyyyyyyy	0-65535	2
U+10000---U+10FFFF	110110yyyyyyyyyy 110111xxxxxxxxxx	65536-1114111	4

UTF-16比起**UTF-8**，好处在于大部分字符都以固定长度的字节（2字节）存储，但UTF-16却无法兼容于**ASCII**编码。

UTF-16可看成是**UCS-2**的父集。在没有辅助平面字符（surrogate code points）前，UTF-16与UCS-2所指的是同一的意思。但当引入辅助平面字符后，就称为UTF-16了。

GB 2312标准共收录6763个汉字，不支持繁体。

GBK的**K**为汉语拼音Kuo Zhan（扩展）中“扩”字的声母。

GBK共收录21886个汉字和图形符号。支持GB2312-80编码不支持的中文繁体。

GBK是一种编码方式并向下兼容GB2312。

字符有一字节和双字节编码，00-7F范围内是第一个字节，和**ASCII**保持一致，此范围内严格上说有96个文字和32个控制符号。

对GB 2312-1980完全向后兼容，与GBK基本向后兼容；支持GB 13000（Unicode）的所有码位；共收录汉字70,244个。

采用变长多字节编码，每个字可以由1个、2个或4个字节组成。

https://zh.wikipedia.org/wiki/ISO/IEC_8859-1

ISO 8859-1

正式编号为ISO/IEC 8859-1:1998，又称Latin-1或“西欧语言”，是国际标准化组织内ISO/IEC 8859的第一个8位字符集。它以ASCII为基础，在空置的0xA0-0xFF的范围内，加入96个字母及符号，藉以供使用附加符号的拉丁字母语言使用。

<https://zh.wikipedia.org/wiki/%E5%85%A8%E5%BD%A2%E5%92%8C%E5%8D%8A%E5%BD%A2>

全角和半角，是计算机中，中、日、韩文的CJKV字符的显示格式。

传统上，英语或拉丁字母语言使用的电脑系统，每一个字母或符号，都是使用一字节的空间（一字节由8比特组成，共256个编码空间）来储存；

而汉语、日语及韩语文字，由于数量大大超过256个，故惯常使用两字节来储存一个字符。

posted @ 2017-11-01 14:33 [my8100](#) [阅读\(...\)](#) [评论\(...\)](#) [编辑](#) [收藏](#)

[python之re正则简单够用](#)

0.

1.参考

[Python正则表达式指南](#)

<https://docs.python.org/2/library/re.html>

<https://docs.python.org/2/howto/regex.html>

<https://docs.python.org/3/library/re.html>

string	re	备注
	<code>re.match(pattern, string, flags=0)</code>	at the start of the string
<code>S.find(sub [,start [,end]]) -> int</code>	<code>re.search(pattern, string, flags=0)</code>	Scan through string looking for a match
<code>S.replace(old, new[, count]) -> string</code>	<code>re.findall(pattern, string, flags=0)</code>	<code>re.finditer</code>

2.分组 `m.group()`

xx

```
In [560]: m.group?
Docstring:
group([group1, ...]) -> str or tuple.
Return subgroup(s) of the match by indices or names.
For 0 returns the entire match.
Type:      builtin_function_or_method
```

```
In [542]: m=re.search(r'(-{1,2}(gr))','pro---gram-files')
```

```
In [543]: m.group()    #自带
Out[543]: '--gr'
```

```
In [544]: m.group(0)    #自带，返回整个匹配到的字符串 For 0 returns the entire match. 注意 m.string 是被检索的完
Out[544]: '--gr'
```

```
In [545]: m.group(1)
Out[545]: '--gr'
```

```
In [546]: m.group(2)
Out[546]: 'gr'
```

```
In [547]: m.group(3)    #加的 （ 不满足则报错
```

```
-----
IndexError                                Traceback (most recent call last)
<ipython-input-547-71a2c7935517> in <module>()
----> 1 m.group(3)
```

```
IndexError: no such group
```

```
In [548]: m.group(1,2)    #选择多个分组，返回tuple
Out[548]: ('--gr', 'gr')
```

```
In [549]: m.groups()    #选择所有分组
Out[549]: ('--gr', 'gr')
```

`m.groupdict` 用于命名分组

```
In [557]: m.groupdict?
Docstring:
groupdict([default=None]) -> dict.
Return a dictionary containing all the named subgroups of the match,
keyed by the subgroup name. The default argument is used for groups
that did not participate in the match
Type:      builtin_function_or_method
```

```
In [558]: m=re.search(r'(-{1,2}(?P<GR>gr))','pro---gram-files')
```

```
In [559]: m.groupdict()
Out[559]: {'GR': 'gr'}
```

3.提取 re.findall()

re.findall(pattern, string, flags=0)

```
In [97]: text = "He was carefully disguised but captured quickly by police."
```

```
In [98]: re.findall(r"\w+ly", text)    #相当于 m.group(0)
Out[98]: ['carefully', 'quickly']
```

```
In [99]: re.findall(r"(\w+ly)", text)   #手动加单个括号限定内容，相当于返回 m.group(1)
Out[99]: ['careful', 'quick']
```

```
In [100]: re.findall(r"((\w+)(ly))", text)    #多个括号，从左到右数 (, 相当于返回 m.groups()
Out[100]: [('carefully', 'careful', 'ly'), ('quickly', 'quick', 'ly')]
```

```
In [102]: re.findall(r"((\w+)(ly))", text)
Out[102]: []
```

4.替换 re.sub()

re.sub(pattern, repl, string, count=0, flags=0)

repl 里面的 前向引用 Backreferences, such as \6, are replaced with the substring matched by group 6 in the pattern. 也可以通过 func 实现。

注意 mysql regexp 不支持 \1

<https://stackoverflow.com/questions/4122393/negative-backreferences-in-mysql-regex> 提到 unless you can install/use LIB_MYSQLUDF_PREG.

<https://stackoverflow.com/questions/7058209/reference-to-groups-in-a-mysql-regex>

```
In [158]: def func(m):
...:     return m.group('DEF')+' '+m.group(2)    #别名
...:
```

```
In [159]: re.sub(r'(?P<DEF>def)\s+([a-z+])\s*\(\s*\):', func, 'def func(): def f():')
Out[159]: 'def func def f'
```

```
In [160]: re.sub(r'(?P<DEF>def)\s+([a-z+])\s*\(\s*\):', r'\1 \2', 'def func(): def f():')    #不支持 \别名
Out[160]: 'def func def f'
```

5. Backreferences 前向引用在 pattern

5.1扑克牌找对子

```
In [204]: re.search(r'(.).*\1', 'ab123')
```

```
In [205]: re.search(r'(.).*\1', 'ab121')
Out[205]: <_sre.SRE_Match at 0x71ca120>
```

```
In [206]: _.group()
Out[206]: '121'
```

5.2连续多个相同

```
In [207]: re.search(r'.{3}', '1122')    #错误
Out[207]: <_sre.SRE_Match at 0x71b94a8>
```

```
In [208]: re.search(r'(.){3}', '1122')  #错误
Out[208]: <_sre.SRE_Match at 0x71ca198>
```

```
In [209]: re.search(r'(\1){3}', '1122') #正确
```

```
In [210]: re.search(r'(\1)\1\1', '1112')
Out[210]: <_sre.SRE_Match at 0x71ca210>
```

```
In [211]: re.search(r'(\1){2}', '1112')
Out[211]: <_sre.SRE_Match at 0x71ca288>
```

```
In [212]: _.group()
```

Out[212]: '111'

posted @ 2017-10-30 10:34 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

0.

1.参考

iso-8859是什么？ 他又被叫做Latin-1或“西欧语言”

补丁:

2.原因

response text 异常

[illegible]

response headers有'content-type'而且没有charset而且有'text', 同时满足三个条件导致判定'ISO-8859-1'

参考文章说 python3 没有问题，实测有。

C:\Program Files\Anaconda2\Lib\site-packages\requests\utils.py

20180102 补充: # "Content-Type": "application/json" 对应 r.encoding 为 None

```
def get_encoding_from_headers(headers):
```

```

"""Returns encodings from given HTTP Header Dict.

:param headers: dictionary to extract encoding from.
:rtype: str
"""

content_type = headers.get('content-type')

if not content_type:
    return None

content_type, params = cgi.parse_header(content_type)

if 'charset' in params:
    return params['charset'].strip("'\"")

if 'text' in content_type:
    return 'ISO-8859-1'

```

C:\Program Files\Anaconda2\Lib\site-packages\requests\adapters.py

```

class HTTPAdapter(BaseAdapter):
    def build_response(self, req, resp):
        # Set encoding.
        response.encoding = get_encoding_from_headers(response.headers)

```

3.解决办法

参考文章打补丁或:

20180102 补充: if resp.encoding == 'ISO-8859-1': 修改为 if r.encoding == 'ISO-8859-1' and not 'ISO-8859-1' in headers.get('content-type', ''): 即只处理按照协议最后返回的 'ISO-8859-1'

```

if r.encoding == 'ISO-8859-1' and not 'ISO-8859-1' in headers.get('content-type', ''):
    encodings = requests.utils.get_encodings_from_content(resp.content) #re.compile(r'<meta.*?char
    if encodings:
        resp.encoding = encodings[0]
    else:
        resp.encoding = resp.apparent_encoding #models.py chardet.detect(self.content)['encoding']
    print 'ISO-8859-1 changed to %s'%resp.encoding

```

posted @ 2017-10-26 16:22 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

python提取网页表格并保存为csv

0.

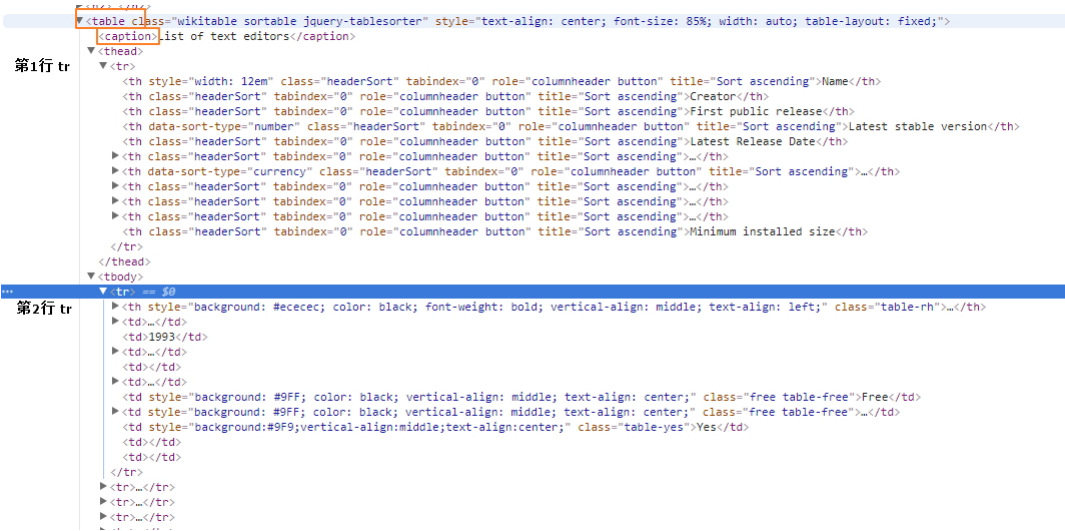
1.参考

W3C HTML 表格

表格标签

表格	描述
<table>	定义表格
<caption>	定义表格标题。
<th>	定义表格的表头。
<tr>	定义表格的行。
<td>	定义表格单元。
<thead>	定义表格的页眉。
<tbody>	定义表格的主体。
<tfoot>	定义表格的页脚。
<col>	定义用于表格列的属性。
<colgroup>	定义表格列的组。

表格元素定位



参看网页源代码并没有thead和tbody。。。

```
<table class="wikitable sortable" style="text-align: center; font-size: 85%; width: auto; table-layout:
<caption>List of text editors</caption>
<tr>
  <th style="width: 12em">Name</th>
  <th>Creator</th>
  <th>First public release</th>
  <th data-sort-type="number">Latest stable version</th>
  <th>Latest Release Date</th>
  <th><a href="/wiki/Programming_language" title="Programming language">Programming language</a></th>
  <th data-sort-type="currency">Cost  (<a href="/wiki/United_States_dollar" title="United States do
  <th><a href="/wiki/Software_license" title="Software license">Software license</a></th>
  <th><a href="/wiki/Free_and_open-source_software" title="Free and open-source software">Open sou
  <th><a href="/wiki/Command-line_interface" title="Command-line interface">Cli available</a></th>
  <th>Minimum installed size</th>
</tr>
<tr>
  <th>
```

2.提取表格数据

表格标题可能出现超链接，导致标题被拆分，也可能不带表格标题。。

```
<caption>Text editor support for remote file editing over
  <a href="/wiki/Lists_of_network_protocols" title="Lists of network protocols">network protocols</a>
</caption>
```

表格内容换行

```
<td>
  <a href="/wiki/Plan_9_from_Bell_Labs" title="Plan 9 from Bell Labs">Plan 9</a>
  and
  <a href="/wiki/Inferno_(operating_system)" title="Inferno (operating system)">Inferno</a>
</td>
```

tag 规律

```
table
thead tr1 th th th th
tbody tr2 td/th
tbody tr3 td/th td
tbody tr3 td/th
```

2.1提取所有表格标题列表

```
filenames = []

for index, table in enumerate(response.xpath('//table')):
    caption = table.xpath('string(./caption)').extract_first() #提取caption tag里面的所有text，包括子节
    filename = str(index+1)+'_'+caption if caption else str(index+1) #xpath 要用到 table 计数，从[1]开始
    filenames.append(re.sub(r'[\w\s()]+', '', filename)) #移除特殊符号
```

```
In [233]: filenames
Out[233]:
[u'1_List of text editors',
 u'2_Text editor support for various operating systems',
 u'3_Available languages for the UI',
 u'4_Text editor support for common document interfaces',
 u'5_Text editor support for basic editing features',
 u'6_Text editor support for programming features (see source code editor)',
 u'7_Text editor support for other programming features',
 '8',
 u'9_Text editor support for key bindings',
 u'10_Text editor support for remote file editing over network protocols',
 u'11_Text editor support for some of the most common character encodings',
 u'12_Right to left (RTL) bidirectional (bidi) support',
 u'13_Support for newline characters in line endings']
```

2.2每个表格分别写入csv文件

```
for index, filename in enumerate(filenames):
    print filename
    with open('%s.csv'%filename, 'wb') as fp:
        writer = csv.writer(fp)
        for tr in response.xpath('//table[%s]/tr'%(index+1)):
            writer.writerow([i.xpath('string(.)').extract_first().replace(u'\xa0', u' ').strip().encode('utf-8') for i in tr.xpath('td')])
```

代码处理 .replace(u'\xa0', u'')

[HTML转义字符 : 表示non-breaking space, unicode编码为u'\xa0',超出gbk编码范围?](#)

使用 'w' 写csv文件，会出现如下问题，使用'wb' 即可解决问题

[【已解决】Python中通过csv的writerow输出的内容有多余的空行 – 在路上](#)

所有表格写入同一excel文件的不同工作表 sheet，需要使用xlwt

[python：创建 excel 工作簿和倾倒 csv 文件作为工作表](#)

HTML转义字符 表示non-breaking space, unicode编码为u'\xa0',超出gbk编码范围?

0. 目录

- 1.参考
- 2.问题定位

不间断空格的unicode表示为 u\xa0',超出gbk编码范围?

- ### 3. 如何处理

```
.extract_first().replace(u'\xa0', u' ').strip().encode('utf-8','replace')
```

1.参考

Beautiful Soup and Unicode Problems

详细解释

`unicodedata.normalize('NFKD',string)` 实际作用???

Scrapy : Select tag with non-breaking space with xpath

```
>>> selector.xpath(u'''
...     //p[normalize-space()]
...     [not(contains(normalize-space(), "\u00a0"))]
... ''')
```

normalize-space() 实际作用？？？

```
In [244]: sel.css('.content')
```

```
Out[244]: [<Selector xpath=u"descendant-or-self::*[@class and contains(concat(' ', normalize-space(@class), ' '), ' ')]>
```

BeautifulSoup下Unicode乱码解决

```
s.replace(u'\xa0', u'').encode('utf-8')
```

2.问题定位

https://en.wikipedia.org/wiki/Comparison_of_text_editors

定位元素显示为 sp;

[illegible]

网页源代码表示为

```
<tr>
<td style="background: #FFD; color: black; vertical-align: middle; text-align: center;" class="partial
<td>= Limited by available memory &#160;&#160;</td>
<td style="background:#F99;vertical-align:middle;text-align:center;" class="table-no">No (64&#160;KB)</
<td>= Some limit less than available memory (give max size if known)</td>
</tr>
</table>
```

实际传输Hex为:

0 [0x0]		Readonly										
Transformer	Headers	TextView	SyntaxView	ImageView	HexView	WebView	Auth	Caching	Cookies	Raw	JSON	XML
000649C5	64 3E 0A 3C 74 64 3E 3D 20 4C 69 6D 69 74 65 64 20 62 79 20 61 76 61 69 6C 61 62	d>.<td>= Limited by availab										
000649E0	6C 65 20 6D 65 6D 6F 72 79 20 26 23 31 36 30 3B 26 23 31 36 30 3B 3C 2F 74 64 3E	le memory </td>										
000649FB	0A 3C 74 64 20 73 74 79 6C 65 3D 22 62 61 63 6B 67 72 6F 75 6E 64 3A 23 46 39 39	.<td style="background:#F99										
00064A16	3B 76 65 72 74 69 69 61 6C 2D 61 6C 69 67 6E 3A 6D 69 64 64 6C 65 3B 74 65 78 74	.vertical-align:middle;text										
00064A31	2D 61 6C 69 67 6E 3A 63 65 6E 74 65 72 3B 22 20 63 6C 61 73 73 3D 22 74 61 62 6C	-align:center;" class="tabl										
00064A4C	65 2D 6E 6F 22 3E 4E 6F 20 28 36 34 26 23 31 36 30 3B 4B 42 29 3C 2F 74 64 3E 0A	e-no">No (64 KB)</td>										
00064A67	3C 74 64 3E 3D 20 53 6F 6D 65 20 6C 69 6D 69 74 20 6C 65 73 73 20 74 68 61 6E 20	<td>= Some limit less than										
00064A82	61 76 61 69 6C 61 62 6C 65 20 6D 65 6D 6F 72 79 20 28 67 69 76 65 20 6D 61 78 20	available memory (give max										
00064A9D	73 69 7A 65 20 69 66 20 6B 6E 6F 77 6E 29 3C 2F 74 64 3E 0A 3C 2F 74 72 3E 0A 3C	size if known)</td>.</tr>.<										
00064AB8	2F 74 61 62 6C 65 3E 0A 3C 7D 3E 49 6E 20 67 65 6E 65 72 61 6C 2C 20 6D 6F 73 74	/table>.<p>In general, most										

不间断空格的unicode表示为 u'xa0'，保存的时候编码 utf-8 则是 '\xc2\xa0'

```
In [211]: for tr in response.xpath('//table[8]/tr[2]'):
...:     print [u''.join(i.xpath('..//text()').extract()) for i in tr.xpath('.*')]
...:
```

[u'memory', u'= Limited by available memory \xa0\xa0', u'No (64\xa0KB)', u'= Some limit less than avail

```
In [212]: u'No (64\xa0KB)'.encode('utf-8')
Out[212]: 'No (64\xc2\xa0KB)'
```

```
In [213]: u'No (64\xa0KB)'.encode('utf-8').decode('utf-8')
Out[213]: u'No (64\xa0KB)'
```

保存 csv 直接使用 excel 打开会有乱码（默认ANSI gbk 打开？？？，u'xa0' 超出 gbk 能够编码范围？？？），使用记事本或notepad++能够自动以 utf-8 正常打开。

D2		= Some limit less than available memory (give max size if known)			
	A	B	C	D	E
1	Yes	= Larger than 4GB (LFS)	2 GB	= Larger than 1GB, not limited by memory	
2	memory	#NAME?	No (64KB)	#NAME?	
3					

使用记事本打开csv文件，另存为 ANSI 编码，之后 excel 正常打开。超出 gbk 编码范围的替换为''

C2		No (64?KB)			
	A	B	C	D	E
1	Yes	= Larger than 4?GB (LFS)	2 GB	= Larger than 1?GB, not limited by memory	
2	memory	= Limited by available memory ??	No (64?KB)	#NAME?	
3					

3.如何处理

```
.extract_first().replace(u'xa0', u'').strip().encode('utf-8','replace')
```

Scrapy Selectors 选择器

0.

1.参考

[《用Python写网络爬虫》——2.2 三种网页抓取方法](#) re / lxml / BeautifulSoup

需要注意的是，lxml在内部实现中，实际上是将CSS选择器转换为等价的XPath选择器。

从结果中可以看出，在抓取我们的示例网页时，Beautiful Soup比其他两种方法慢了超过6倍之多。实际上这一结果是符合预期的，因为lxml和正则表达式模块都是C语言编写的，而BeautifulSoup则是纯Python编写的。一个有趣的事实是，lxml表现得和正则表达式差不多好。由于lxml在搜索元素之前，必须将输入解析为内部格式，因此会产生额外的开销。而当抓取同一网页的多个特征时，这种初始化解析产生的开销就会降低，lxml也就更具竞争力。这真是一个令人惊叹的模块！

2.Scrapy Selectors 选择器

<https://doc.scrapy.org/en/latest/topics/selectors.html#topics-selectors>

- BeautifulSoup缺点：慢
- lxml:基于 ElementTree
- Scrapy selectors: parsel library, 构建于 lxml 库之上，这意味着它们在速度和解析准确性上非常相似。

.css() .xpath() 返回 SelectorList, 即 a list of new selectors

.extract() .re() 提取 过滤 tag data

```
import scrapy
```

```
C:\Program Files\Anaconda2\Lib\site-packages\scrapy\__init__.py
```

```
from scrapy.selector import Selector
```

```
C:\Program Files\Anaconda2\Lib\site-packages\scrapy\selector\__init__.py
```

```
from scrapy.selector.unified import *
```

```
C:\Program Files\Anaconda2\Lib\site-packages\scrapy\selector\unified.py
```

```
from parsel import Selector as _ParselSelector
```

```
class Selector(_ParselSelector, object_ref):
```

```
>>> from scrapy.selector import Selector
```

```
>>> from scrapy.http import HtmlResponse
```

如此导入 Selector, 实例化 Selector 的时候第一个参数是 HtmlResponse 实例, 如果要通过 str 实例化 Selector , 需要 se

```
xx
```

```
In [926]: from parsel import Selector
```

```
In [927]: Selector?
```

```
Init signature: Selector(self, text=None, type=None, namespaces=None, root=None, base_url=None, _expr=None)
Docstring:
```

```
:class: Selector` allows you to select parts of an XML or HTML text using CSS
or XPath expressions and extract data from it.
```

```
`text` is a `unicode` object in Python 2 or a `str` object in Python 3
```

```
`type` defines the selector type, it can be `html`, `xml` or `None` (default).
If `type` is `None`, the selector defaults to `html`.
```

```
File: c:\program files\anaconda2\lib\site-packages\parsel\selector.py
```

```
Type: type
```

```
xx
```

```
doc=u"""
<div class="quote" itemscope itemtype="http://schema.org/CreativeWork">
    <span class="text" itemprop="text">"I have not failed. I've just found 10,000 ways that won't w
    <span>by<small class="author" itemprop="author">Thomas A. Edison</small>
```



```
<a href="/author/Thomas-A-Edison">(about)</a>
"""

sel = Selector(doc)

sel.css('div.quote')
[<Selector xpath=u"descendant-or-self::div[@class and contains(concat(' ', normalize-space(@class), ' '
"
```

3.使用 scrapy shell 调试

<https://doc.scrapy.org/en/latest/intro/tutorial.html#extracting-data>

```
G:\pydata\pycode\scrapy\splash_cnblogs>scrapy shell "http://quotes.toscrape.com/page/1/"
```

3.1Xpath VS CSS

对比

	CSS	Xpath	备注
含有属性	response.css('div[class]')	response.xpath('//div[@class]')	css可以简写为 div.class 甚至 .class, div#abc 或 #abc 则对应于id=abc
匹配属性值	response.css('div[class="quote"]')	response.xpath('//div[@class="quote"]')	response.xpath('//small[text()="Albert Einstein"]')
匹配部分属性值	response.css('div[class*="quo"]')	response.xpath('//div[contains(@class,"quo")]')	response.xpath('//small[contains(text(),"Einstein")]')
提取属性值	response.css('small::attr(class)')	response.xpath('//small/@class')	css里面text排除在attr以外，所以不支持上面两个过滤text???
提取文字	response.css('small::text')	response.xpath('//small/text()')	

使用

```
In [135]: response.xpath('//small[@class="author"]').extract_first()
In [122]: response.css('small.author').extract_first()
Out[122]: u'<small class="author" itemprop="author">Albert Einstein</small>'
```

```
In [136]: response.xpath('//small[@class="author"]/text()').extract_first()
In [123]: response.css('small.author::text').extract_first()
Out[123]: u'Albert Einstein'
```

```
In [137]: response.xpath('//small[@class="author"]/@class').extract_first() #class也是属性
In [124]: response.css('small.author::attr(class)').extract_first()
Out[124]: u'author'
```

```
In [138]: response.xpath('//small[@class="author"]/@itemprop').extract_first()
In [125]: response.css('small.author::attr(itemprop)').extract_first()
Out[125]: u'author'
```

class 是一个特殊属性，允许多值 class="row header-box"

匹配多值中的某一个值

```
In [228]: response.css('div.row')
Out[228]:
[<Selector xpath=u"descendant-or-self::div[@class and contains(concat(' ', normalize-space(@class), ' '
<Selector xpath=u"descendant-or-self::div[@class and contains(concat(' ', normalize-space(@class), ' '
"
```

```
In [232]: response.css('div.ro')
Out[232]: []
```

整个class属性值，匹配全部字符串

```
In [226]: response.css('div[class="row"]')
Out[226]: [<Selector xpath=u"descendant-or-self::div[@class = 'row']" data=u'<div class="row">\n
In [240]: response.xpath('//div[@class="row header-box"]')
Out[240]: [<Selector xpath="//div[@class="row header-box"]' data=u'<div class="row header-box">\n >]
```

整个class属性值，匹配部分字符串

```
In [229]: response.css('div[class*="row"]')
Out[229]:
[<Selector xpath=u"descendant-or-self::div[@class and contains(@class, 'row')]" data=u'<div class="row
"
```

```
<Selector xpath=u"descendant-or-self::div[@class and contains(@class, 'row')]" data=u'<div class="row"
In [230]: response.xpath('//div[contains(@class,"row")])')
Out[230]:
[<Selector xpath='//div[contains(@class,"row")]' data=u'<div class="row header-box">\n          ',
 <Selector xpath='//div[contains(@class,"row")]' data=u'<div class="row">\n          <div class="col-md">]

In [234]: response.css('div[class*="w h"]')
Out[234]: [<Selector xpath=u"descendant-or-self::div[@class and contains(@class, 'w h')]" data=u'<div c
In [235]: response.xpath('//div[contains(@class,"w h")])')
Out[235]: [<Selector xpath='//div[contains(@class,"w h")]' data=u'<div class="row header-box">\n
```

3.2提取数据

- 提取data

```
selectorList / selector.extract(), extract_first()
```

selector.extract() 返回一个str, selector.extract_first() 报错

selectorList.extract() 对每一个selector执行selector.extract, 返回 list of str, selectorList.extract_first() 取前面list的第一个。

- 提取data同时过滤

```
selectorList / selector.re(r'xxx'), re_first(r'xxx')
```

selector.re() 返回 list, selector.re_first() 取第一个str

selectorList.re() 对每一个selector执行selector.re, 每个list结果（**注意并非每个selector都会match**）合并为一个list, **selectorList.re_first()** 取前面合并list的第一个str。。。

使用 extract

```
In [21]: response.css('.author') #内部转为 xpath, 返回 SelectorList 实例
Out[21]:
[<Selector xpath=u"descendant-or-self::*[@class and contains(concat(' ', normalize-space(@class), ' '),
 <Selector xpath=u"descendant-or-self::*[@class and contains(concat(' ', normalize-space(@class), ' '),
 <Selector xpath=u"descendant-or-self::*[@class and contains(concat(' ', normalize-space(@class), ' '),
 <Selector xpath=u"descendant-or-self::*[@class and contains(concat(' ', normalize-space(@class), ' '),
 <Selector xpath=u"descendant-or-self::*[@class and contains(concat(' ', normalize-space(@class), ' '),
 <Selector xpath=u"descendant-or-self::*[@class and contains(concat(' ', normalize-space(@class), ' '),
 <Selector xpath=u"descendant-or-self::*[@class and contains(concat(' ', normalize-space(@class), ' '),
 <Selector xpath=u"descendant-or-self::*[@class and contains(concat(' ', normalize-space(@class), ' '),
 <Selector xpath=u"descendant-or-self::*[@class and contains(concat(' ', normalize-space(@class), ' '),
 <Selector xpath=u"descendant-or-self::*[@class and contains(concat(' ', normalize-space(@class), ' '),
```

```
In [22]: response.css('.author').extract() #提取上面的 data 部分, 对 SelectorList 中的每个 Selector 执行 ex
Out[22]:
[u'<small class="author" itemprop="author">Albert Einstein</small>',
 u'<small class="author" itemprop="author">J.K. Rowling</small>',
 u'<small class="author" itemprop="author">Albert Einstein</small>',
 u'<small class="author" itemprop="author">Jane Austen</small>',
 u'<small class="author" itemprop="author">Marilyn Monroe</small>',
 u'<small class="author" itemprop="author">Albert Einstein</small>',
 u'<small class="author" itemprop="author">Andr\xe9 Gide</small>',
 u'<small class="author" itemprop="author">Thomas A. Edison</small>',
 u'<small class="author" itemprop="author">Eleanor Roosevelt</small>',
 u'<small class="author" itemprop="author">Steve Martin</small>']
```

```
In [23]: response.css('.author').extract_first() #只取第一个, 可能返回 None ,可能报错 response.css('.author
Out[23]: u'<small class="author" itemprop="author">Albert Einstein</small>'
```

```
In [24]: response.css('.author::text').extract_first() #定位到 tag 内部的 text
Out[24]: u'Albert Einstein'
```

使用 re

```
In [46]: response.css('.author::text')[0]
Out[46]: <Selector xpath=u"descendant-or-self::*[@class and contains(concat(' ', normalize-space(@class
```

```
In [47]: response.css('.author::text')[0].re(r'\w+')
Out[47]: [u'Albert', u'Einstein']
```

```
In [48]: response.css('.author::text')[0].re_first(r'\w+')
Out[48]: u'Albert'
```

```
In [49]: response.css('.author::text')[0].re(r'((\w+)\s(\w+))') #按照左边括号顺序输出
Out[49]: [u'Albert Einstein', u'Albert', u'Einstein']
```

3.

posted @ 2017-10-20 17:33 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

[python之使用 wkhtmltopdf 和 pdftkit 批量加载html生成pdf, 适用于博客备份和官网文档打包](#)

0.

1.参考

[Python 爬虫：把廖雪峰教程转换成 PDF 电子书](#)

https://github.com/lzjun567/crawler_html2pdf

wkhtmltopdf 就是一个非常好的工具，它可以用适用于多平台的 html 到 pdf 的转换，pdftkit 是 wkhtmltopdf 的Python封装包。

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/#>

也可以通过 BeautifulSoup 插入删除tag

```
soup.insert  
soup.decompose
```

2.安装 wkhtmltopdf

windows 安装：

<https://wkhtmltopdf.org/downloads.html>

下载版本 Windows (MinGW) 0.12.4 32-bit / 64-bit for Windows XP/2003 or later; standalone

添加路径 D:\Program Files\wkhtmltopdf\bin

需要重新打开cmd以及notepad++。。。

###centos 7 安装：

<https://gist.github.com/AndreasFurster/ebe3f163d6d47be43b72b35b18d8b5b6>

<https://gist.github.com/calebbrewer/aca424fb14618df8aadd> rpm 安装，依赖包

<https://yq.aliyun.com/articles/86256> 为了能从任意路径执行程序，将 wkhtmltopdf 安装到 /usr/bin 目录下。

确认最新下载地址 <https://wkhtmltopdf.org/downloads.html> Linux 0.12.4 32-bit / 64-bit depends on: zlib, fontconfig, freetype, X11 libs (libX11, libXext, libXrender)

```
wget https://github.com/wkhtmltopdf/wkhtmltopdf/releases/download/0.12.4/wkhtmltox-0.12.4_linux-generic  
unxz wkhtmltox-0.12.4_linux-generic-amd64.tar.xz  
tar -xvf wkhtmltox-0.12.4_linux-generic-amd64.tar  
mv wkhtmltox/bin/* /usr/local/bin/  
rm -rf wkhtmltox  
rm -f wkhtmltox-0.12.4_linux-generic-amd64.tar
```

上述方法安装后 wkhtmltopdf 提示 未找到命令

检查

```
[root@iz2zeb4wal7uttzolpwmbez bin]# which wkhtmltopdf  
/usr/bin/which: no wkhtmltopdf in (/sbin:/bin:/usr/sbin:/usr/bin)  
[root@iz2zeb4wal7uttzolpwmbez bin]# which mysqld  
/sbin/mysqld
```

解决办法

```
mv /usr/local/bin/wkhtmltopdf /usr/bin/wkhtmltopdf
```

```
mv /usr/local/bin/wkhtmltoimage /usr/bin/wkhtmltoimage
```

确认

```
[root@iz2zeb4wal7uttzolpwmbez /]# which wkhtmltopdf  
/bin/wkhtmltopdf  
[root@iz2zeb4wal7uttzolpwmbez /]# which wkhtmltoimage  
/bin/wkhtmltoimage
```

验证 wkhtmltopdf <http://httpbin.org/temp.pdf>

###centos 7 需要安装字体输出中文:

<http://www.liumapp.com/articles/2017/04/10/1491811668985.html> CentOS安装wkhtmltopdf并解决中文字体的问题

其中字体到这里找 C:\Windows\Fonts simsun.ttc msyh.ttf

###安装 python 库

```
pip install pdfkit
```

API <https://pypi.python.org/pypi/pdfkit>

定制options, 搜索关键字 <https://wkhtmltopdf.org/usage/wkhtmltopdf.txt>

```
options = {
    'page-size': 'Letter',
    'margin-top': '0.75in',
    'margin-right': '0.75in',
    'margin-bottom': '0.75in',
    'margin-left': '0.75in',
    'encoding': "UTF-8",    #支持中文
    'custom-header' : [
        ('Accept-Encoding', 'gzip')
    ]
    'cookie': [
        ('cookie-name1', 'cookie-value1'),
        ('cookie-name2', 'cookie-value2'),
    ],
    'no-outline': None
}

pdfkit.from_url('http://google.com', 'out.pdf', options=options)
```

3.背景知识

3.1url 相对路径 绝对路径

```
In [323]: urlparse.urljoin('https://doc.scrapy.org/en/latest/index.html', 'intro/overview.html')    #相当
Out[323]: 'https://doc.scrapy.org/en/latest/intro/overview.html'

In [324]: urlparse.urljoin('https://doc.scrapy.org/en/latest/intro/overview.html', '#walk-through-of-ar
Out[324]: 'https://doc.scrapy.org/en/latest/intro/overview.html#walk-through-of-an-example-spider'

In [326]: urlparse.urljoin('https://doc.scrapy.org/en/latest/intro/overview.html', 'install.html')    #相
Out[326]: 'https://doc.scrapy.org/en/latest/intro/install.html'

In [327]: urlparse.urljoin('https://doc.scrapy.org/en/latest/intro/overview.html', '../topics/commands.
Out[327]: 'https://doc.scrapy.org/en/latest/topics/commands.html'
```

<https://doc.scrapy.org/en/latest/index.html>

这一类官方文档一般页脚都为:

© Copyright 2008-2016, Scrapy developers. Revision 65ac0b06.

Built with [Sphinx](#) using a [theme](#) provided by [Read the Docs](#).

3.2页面布局规律

- 点击左上角 home 图标转到首页
- 左边栏页面导航
 - `<div class="wy-menu wy-menu-vertical" data-spy="affix" role="navigation" aria-label="main navigation">`
 - `Scrapy at a glance`
- 正文主体
 - `<div class="rst-content">`

3.3转换pdf时注意事项

- 提取正文主体后，可以直接将 <div xxxx </div> 保存html，不需要补全 <html>
- 图片链接相对路径需要转换为绝对路径，才会自动加载图片
 -
- pdfkit.from_file 第一个参数 input 为 html文件路径列表，文件名不能是中文。。。
 - pdfkit.from_file(self.htmls_saved, self.netloc+'.pdf', options=options)
- pdf会根据<h1><h2>等标题 tag 自动生成目录

4.完整代码

```
#!/usr/bin/env python
#coding:utf-8

import os
import sys
import time
import traceback
import re
import urlparse
import threading
import Queue

import requests
from scrapy import Selector
import pdfkit

s = requests.Session()
# s.headers.update({'user-agent':'Mozilla/5.0 (iPhone; CPU iPhone OS 9_3_5 like Mac OS X) AppleWebKit/6
s.headers.update({'user-agent':'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, lik
# s.headers.update({'Referer':'https://servicewechat.com/wx55b926152a8c3bef/14/page-frame.html'})
s.verify = False
s.mount('http://', requests.adapters.HTTPAdapter(pool_connections=1000, pool_maxsize=1000))
s.mount('https://', requests.adapters.HTTPAdapter(pool_connections=1000, pool_maxsize=1000))
import copy
sp = copy.deepcopy(s)
proxies = {'http': 'http://127.0.0.1:1080', 'https': 'https://127.0.0.1:1080'}
sp.proxies = proxies

from urllib3.exceptions import InsecureRequestWarning
from warnings import filterwarnings
filterwarnings('ignore', category = InsecureRequestWarning)

html_template = u"""
<!DOCTYPE html>

<html>
    <head>
        <meta charset="utf-8" />
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    </head>
    <body>
        <!-- <center><h1>{title}</h1></center> -->
        {content}
    </body>
</html>
"""

# https://wkhtmltopdf.org/usage/wkhtmltopdf.txt

options = {
    'page-size': 'A4', # Letter
    'minimum-font-size': 25, ###
    # 'image-dpi':1500, ###

    'margin-top': '0.1in', #0.75in
    'margin-right': '0.1in',
    'margin-bottom': '0.1in',
    'margin-left': '0.1in',
    'encoding': 'UTF-8', #支持中文
    'custom-header': [
        ('Accept-Encoding', 'gzip')
    ],
    'cookie': [
        ('cookie-name1', 'cookie-value1'),
        ('cookie-name2', 'cookie-value2'),
    ],
}
```

```

'outline-depth': 10,
}

# 验证 re.findall(pattern, s)
# ) 为 (") 或(".*?alt)或(".*?style)确认问题, 所以要加 re.S
# <div style="text-align:center;">
# <div style="text-align:center; margin: 0 auto 0 auto; "></div>

pattern_img_scr = re.compile(r'(<img\s+[\^>]*?src\s*=\s*"") (?P<src>.*?) (".*?>)', re.S) #最后不能简写成 " ,

# <a class="reference external" href="http://code.google.com/p/selenium/issues/detail?id=1008">issue 10
# text为空, 也能匹配到 m.group(4)=''
pattern_a_href = re.compile(r'(<a\s+[\^>]*?href\s*=\s*"") (?P<href>.*?) (".*?>) (?P<text>.*?) (</a>)', re.S)

# http://www.seleniumhq.org/docs/ 合体。。。text为 第一章、架設伺服器前的準備工作</span> #要变h1
# 错误: <span class="text_h1">1.1 前言: Linux 有啥功能</span> #要变h2
# 更正: <h1>第七章、Linux 磁碟與檔案系統管理</h1>
# 复杂嵌套:<span class="text_head0"><span class="text_head_en">Linux </span>基礎</span>
# 这里只处理最普通情况
pattern_span_head = re.compile(r'<span\s+class\s*=\s*"text_h(?:ead)?(\d)">([^\^<]*?)</span>') #(?:xxx)数量

```

```

class HTMLtoPDF(object):

```

```

    def __init__(self, seed_url, proxy=False, pdf='', page=1, font_size=25, css_links='div[class="wy-me
        css_content='div.rst-content', threads_count=30):
        self.seed_url = seed_url
        self.session = sp if proxy else s

        options['minimum-font-size'] = font_size

        self.pdf = pdf
        self.netloc = urlparse.urlparse(seed_url).netloc
        print self.netloc
        self.folder = os.path.join(sys.path[0], '{}({})'.format(self.netloc, self.pdf))
        self.folder_temp = os.path.join(sys.path[0], 'temp')
        for f in [self.folder, self.folder_temp]:
            if not os.path.isdir(f):
                os.mkdir(f)

        self.css_content = css_content
        self.css_links = css_links

        self.threads_count = threads_count
        # self.lock = threading.Lock()
        self.links_queue = Queue.Queue()
        self.links_seen = []

        for p in range(1, page+1):
            url = re.sub(r'page=\d+', 'page=%s'%p, self.seed_url) #本来无page, 原样返回
            self.links_queue.put((str(len(self.links_seen)), url))
            self.links_seen.append(url)
            self.get_links(url)
        self.links_queue_size = self.links_queue.qsize()
        self.htmls_saved = []

    def get_links(self, url):
        encoding, text = self.load_page(url)
        sel = Selector(text=text)

        # [u'#selenium-documentation',
        # u'00_Note_to-the-reader.jsp',
        # u'01_introducing_selenium.jsp',

```

```

# u'01_introducing_selenium.jsp#test-automation-for-web-applications',

links = [re.sub(r'#$', '', i) for i in sel.css(self.css_links).extract()]
print links
for link in links:  #set(links) 会导致乱序,使用urls_seen 去重
    link_abs = urlparse.urljoin(url, link)
    if link_abs not in self.links_seen:
        self.links_queue.put((str(len(self.links_seen)), link_abs))
        self.links_seen.append(link_abs)

def run(self):
    threads = []
    for i in range(self.threads_count):
        t = threading.Thread(target=self.save_html)
        threads.append(t)

    for t in threads:
        t.setDaemon(True)
        t.start()

    self.links_queue.join()
    print 'load done'

def func(filename):
    _, filename = os.path.split(filename)
    return int(filename[:filename.index('_')])

self.htmls_saved.sort(key=lambda x:func(x))
output = u'{}({}) {}.pdf'.format(self.netloc, self.pdf, time.strftime('%Y%m%d_%H%M%S'))
pdfkit.from_file(self.htmls_saved, output, options=options)
print output

def save_html(self):
    while True:
        try:
            (num, url) = self.links_queue.get()
            meta_encoding, text = self.load_page(url)
            # http://linux.vbird.org/linux_desktop/index.php
            # meta charset 是 big5
            # pdfkit 已经设置 utf-8, 所以temp文件需要编码为 utf-8
            # 用于浏览的保存页面, 如果编码为 utf-8, 不符合meta charset 冲突,
            # 需要通过firefox手动指定编码

            encoding_to = meta_encoding if meta_encoding else 'utf-8'

            text = self.modify_text(url, text)  ##### 含有原始 <html meta charse
            # text = self.modify_content2(url, text)
            # text1 = text

            title, content = self.parse_page(url, text)

            filename_cn = u'{}_{}.html'.format(num, re.sub(ur'^\u4e00-\u9fa5\w\s()_-]', '', title))
            filename = u'{}_{}_{}.html'.format(num, re.sub(r'^\w\s()_-]', '', title), int(time.

            with open(os.path.join(self.folder, filename_cn), 'wb') as fp:
                fp.write(text.encode(encoding_to, 'replace'))
            f = os.path.join(self.folder_temp, filename)
            with open(f, 'wb') as fp:
                fp.write(content.encode('utf-8', 'replace'))
                # fp.write(html_template.format(content=content, title=title).encode('utf-8', 'repla
                self.htmls_saved.append(f)
                print '{}/{} {}'.format(len(self.htmls_saved), self.links_queue_size, url)

            self.links_queue.task_done()
        except Exception as err:
            print '#####{} {} {}'.format(url, err, traceback.format_exc())

def load_page(self, url):
    resp = self.session.get(url)
    # http://linux.vbird.org/linux_desktop/index.php
    # meta charset 是 big5
    meta_encoding = None
    if resp.encoding == 'ISO-8859-1':

```



```

        encodings = requests.utils.get_encodings_from_content(resp.content) #re.compile(r'<meta.*?
    if encodings:
        resp.encoding = encodings[0]
        meta_encoding = resp.encoding
    else:
        resp.encoding = resp.apparent_encoding #models.py chardet.detect(self.content)['encod
    print 'ISO-8859-1 changed to %s'%resp.encoding

    return (meta_encoding, resp.text)

def modify_text(self, url, text):
    # m.group(1)='abc' SyntaxError: can't assign to function call 不能直接赋值

    # https://doc.scrapy.org/en/latest/topics/firebug.html
    # ../_images/firebug1.png
    # 异常 urlparse.urljoin(self.seed_url, src)

    # r'(<img\s+[^>]*?src\s*=\s*"") (?P<src>.*?) (".*?>)'
    def func_src(m):
        src = m.group('src') #别名
        if not src.startswith('http'):
            src = urlparse.urljoin(url, src)
        # print u'{}'.format(m.group(1), src, m.group(3))
        return u'{}'.format(m.group(1), src, m.group(3))
    text = re.sub(pattern_img_scr, func_src, text)

    def func_head(m):
        # re.compile(r'<span class="text h.*?(\d)">(.*?)</span>', re.S)
        # 错误: <span class="text head0">第一章、架設伺服器前的準備工作</span>
        # 更正: <h1>第七章、Linux 磁碟與檔案系統管理</h1>
        return u'<h{num}>{word}</h{num}>'.format(num=int(m.group(1))+1, word=m.group(2))
    text = re.sub(pattern_span_head, func_head, text)

    # re.compile(r'(<a\s+[^>]*?href\s*=\s*"") (?P<href>.*?) (".*?>) (?P<text>.*?) (</a>)'
    def func_href(m):
        href = m.group('href')
        text = m.group('text')
        if not href.startswith('#'):
            if not href.startswith('http'):
                href = urlparse.urljoin(url, href)
            # text = u'{text} ({href})'.format(text=text, href=href)
            return u'{g1}{href}{g3}{text}{g5}'.format(g1=m.group(1), g3=m.group(3), g5=m.group(5), href

            #m.string是content全文。。。也不能 return m
    text = re.sub(pattern_a_href, func_href, text)

    return text

def parse_page(self, url, text):
    sel = Selector(text=text)
    title = sel.css('head title::text').extract_first() or '' #固定css, 匹配不到 extract()返回[],ext
    content = sel.css(self.css_content).extract_first() or '' #'div.rst-content'

    content = self.clean_content(content)

    return title, content

def clean_content(self, content):
    # <a class="headerlink" href="#check" title="Permalink to this headline">¶</a> headline 自帶图
    content = content.replace(u'>\xb6<', u'><')

    # selenium LanguagePreference
    sel = Selector(text=content)
    # content = content.replace(sel.css('div#codeLanguagePreference').extract_first(), '') #可能是No
    for div in sel.css('div#codeLanguagePreference').extract():
        content = content.replace(div, '')

    for lang in ['java', 'csharp', 'ruby', 'php', 'perl', 'javascript']:
        for div in sel.css('div.highlight-%s'%lang).extract():
            # print len(content)
            content = content.replace(div, '')

```

```

# liaoxuefeng comment
content = content.replace('<h3>Comments</h3>', '')
content = content.replace('<h3>Make a comment</h3>', '')

# http://lxml.de/
for div in sel.css('div.sidemenu').extract():
    content = content.replace(div, '')

return content

# 未使用, 下面解释了, extract() 执行 Serialize 导致 tag 内部 \n 丢失, replace(tag, tag_new) 失效
def modify_content2(self, url, content):
    sel = Selector(text=content)

    # 修改图片链接为绝对链接, 否则pdf无法图片
    # 

    for i in sel.css('img[src]'):
        tag = i.extract()
        src = i.xpath('./@src').extract_first()
        if not src.startswith('http'):
            src_abs = urlparse.urljoin(url, src)
            # print src, src_abs
            tag_new = tag.replace(src, src_abs)

            # In [392]: sel.extract?
            # Signature: sel.extract()
            # Docstring:
            # Serialize and return the matched nodes in a single unicode string.
            # Percent encoded content is unquoted.
            # File:      c:\program files\anaconda2\lib\site-packages\parsel\selector.py
            # Type:      instancemethod
            # Serialize.....
            # print tag #换行被自动省略, 后面的replace 失效..
            # print tag_new
            # 
    for i in sel.css('a[href]'):
        tag = i.extract()
        href = i.xpath('./@href').extract_first()
        text = i.xpath('./text()').extract_first()

        # 补全内部链接, 忽略本页面的#定位
        if not href.startswith('http') and not href.startswith('#'):
            href_abs = urlparse.urljoin(url, href)
            # print href, href_abs
            tag_new = tag.replace(href, href_abs)
        else:
            href_abs = href
            tag_new = tag

        # 图标链接, 如果text为None, replace表现异常
        if text and not href.startswith('#'):
            text_new = u'{} ({} )'.format(text, href_abs)
            # print text.encode('gbk', 'replace'), text_new.encode('gbk', 'replace')
            tag_new = tag_new.replace(text, text_new)

        # 保证整体替换
        content = content.replace(tag, tag_new)

    return content

```

```

if __name__ == '__main__':

```

```

url = 'http://www.cnblogs.com/my8100/default.html?page=1'

```

```
obj = HTMLtoPDF(url, page=7, pdf='my8100', css_links='div#mainContent div.postTitle a::attr(href)',
obj.run()
```

5.更多应用

```
if __name__ == '__main__':

    url = 'https://doc.scrapy.org/en/latest/index.html'
    # obj = HTMLtoPDF(url, proxy=True)

    url = 'http://python3-cookbook.readthedocs.io/zh_CN/latest/index.html'
    # obj = HTMLtoPDF(url, font_size=20, css_links='div[class="toctree-wrapper compound"] a::attr(href)'

    url = 'http://www.seleniumhq.org/docs/'
    # obj = HTMLtoPDF(url, proxy=True, css_links='div#selenium-documentation a::attr(href)', css_conter

    url = 'https://www.crummy.com/software/BeautifulSoup/bs4/doc/'
    # obj = HTMLtoPDF(url, pdf='BeautifulSoup', css_links='div.sphinxsidebarwrapper a::attr(href)', css

    url = 'https://www.liaoxuefeng.com/wiki/001374738125095c955c1e6d8bb493182103fac9270762a000'
    # obj = HTMLtoPDF(url, proxy=True, pdf='python2', css_links='ul#x-wiki-index a::attr(href)', css_cc

    url = 'https://www.liaoxuefeng.com/wiki/0014316089557264a6b348958f449949df42a6d3a2e542c000'
    # obj = HTMLtoPDF(url, proxy=True, pdf='python3', css_links='ul#x-wiki-index a::attr(href)', css_cc

    url = 'https://docs.python.org/2/howto/index.html'
    # obj = HTMLtoPDF(url, pdf='python2howto', css_links='div[class="toctree-wrapper compound"] a::attr

    url = 'http://lxml.de/'
    # obj = HTMLtoPDF(url, pdf='lxml', css_links='div.menu a::attr(href)', css_content='div.document')

    url = 'http://docs.python-requests.org/zh_CN/latest/'
    # obj = HTMLtoPDF(url, pdf='requests', css_links='div[class="toctree-wrapper compound"] a::attr(hre

    url = 'http://twistedmatrix.com/documents/current/core/howto/index.html'
    # obj = HTMLtoPDF(url, pdf='twisted_core_guide', css_links='div[class="body"] a::attr(href)', css_c

    url = 'http://linux.vbird.org/linux_basic/'
    # obj = HTMLtoPDF(url, pdf='linux_basic', css_links='div[class="mainarea"] a::attr(href)', css_cont

    url = 'http://linux.vbird.org/linux_server/'
    # obj = HTMLtoPDF(url, pdf='linux_server', css_links='td[width="718"] a::attr(href)', css_content='

    url = 'http://linux.vbird.org/linux_desktop/'
    # obj = HTMLtoPDF(url, pdf='linux_desktop', css_links='td[width="718"] a::attr(href)', css_content=
```

[Scrapy 改进第二步: Web Interface 添加 STOP 和 START 超链接, 一键调用 Scrapy API](#)

0.提出问题

Scrapy 提供的开始和结束项目的API如下, 参考 [Scrapy 改进第一步: Web Interface 添加 charset=UTF-8, 避免查看 log 出现中文乱码](#), 准备继续在页面上进一步添加 START 和 STOP 超链接。

<http://scrapy.readthedocs.io/en/stable/api.html#schedule-json>

Example request:

```
$ curl http://localhost:6800/schedule.json -d project=myproject -d spider=somespider
```

Example response:

```
{"status": "ok", "jobid": "6487ec79947edab326d6db28a2d86511e8247444"}
```

<http://scrapy.readthedocs.io/en/stable/api.html#cancel-json>

Example request:

```
$ curl http://localhost:6800/cancel.json -d project=myproject -d job=6487ec79947edab326d6db28a2d86511e8247444
```

Example response:

```
{"status": "ok", "prevstate": "running"}
```

1.解决思路

尝试直接通过浏览器地址栏 GET 请求页面 <http://localhost:6800/schedule.json?project=myproject&spider=somespider>

返回提示需要使用 POST 请求

```
{"node_name": "pi-desktop", "status": "error", "message": "Expected one of [b'HEAD', b'object', b'POST']"}
```

那就继续[通过 URL 查询对传参, 通过 JS 发起 POST 异步请求吧](#)

2.修改 Scrapy 代码

[/site-packages/scrapy/webui.py](#)

改动位置:

(1) table 添加最后两列, 分别用于 UTF-8 和 STOP/START 超链接, 见红色代码

```
def render(self, txrequest):
    cols = 10 ##### 8
    s = "<html><head><meta charset='UTF-8'><title>Scrapy</title></head>"
    s += "<body>"
    s += "<h1>Jobs</h1>"
    s += "<p><a href='..'>Go back</a></p>"
    s += "<table border='1'>"
    s += "<tr><th>Project</th><th>Spider</th><th>Job</th><th>PID</th><th>Start</th><th>Runtime</th>"
    if self.local_items:
        s += "<th>Items</th>"
        #cols = 9 #####
        cols += 1 #####
```

(2) 有两处需要添加 UTF-8 超链接, 分别对应 Running 和 Finished, 见红色代码

前面 Running 部分添加 UTF-8 超链接后继续添加 STOP 超链接, 见蓝色代码

```
s += "<td><a href='/logs/%s/%s/%s.log'>Log</a></td>" % (p.project, p.spider, p.job)
s += "<td><a href='/logs/UTF-8.html?project=%s&spider=%s&job=%s' target='_blank'>UTF-8</a><"
s += "<td><a href='/logs/scrapy.html?opt=cancel&project=%s&job_or_spider=%s' target='_blank'>STOP</a></td>" % (p.project, p.spider, p.job)
```

后面 Finished 部分添加 UTF-8 超链接后继续添加 START 超链接, 见蓝色代码

```
s += "<td><a href='/logs/%s/%s/%s.log'>Log</a></td>" % (p.project, p.spider, p.job)
```

```
s += "<td><a href='/logs/UTF-8.html?project=%s&spider=%s&job=%s' target='_blank'>UTF-8</a><td><a href='/logs/scrapy.html?opt=schedule&project=%s&job_or_spider=%s' target='_bl
```

(3) 完整代码

▣

```
from datetime import datetime

import socket

from twisted.web import resource, static
from twisted.application.service import IServiceCollection

from scrapy.utils.misc import load_object

from .interfaces import IPoller, IEggStorage, ISpiderScheduler

from six.moves.urllib.parse import urlparse

class Root(resource.Resource):

    def __init__(self, config, app):
        resource.Resource.__init__(self)
        self.debug = config.getboolean('debug', False)
        self.runner = config.get('runner')
        logsdir = config.get('logs_dir')
        itemsdir = config.get('items_dir')
        local_items = itemsdir and (urlparse(itemsdir).scheme.lower() in ['', 'file'])
        self.app = app
        self.nodename = config.get('node_name', socket.gethostname())
        self.putChild(b'', Home(self, local_items))
        if logsdir:
            self.putChild(b'logs', static.File(logsdir.encode('ascii', 'ignore'), 'text/plain'))
        if local_items:
            self.putChild(b'items', static.File(itemsdir, 'text/plain'))
        self.putChild(b'jobs', Jobs(self, local_items))
        services = config.items('services', ())
        for servName, servClsName in services:
            servCls = load_object(servClsName)
            self.putChild(servName.encode('utf-8'), servCls(self))
        self.update_projects()

    def update_projects(self):
        self.poller.update_projects()
        self.scheduler.update_projects()

    @property
    def launcher(self):
        app = IServiceCollection(self.app, self.app)
        return app.getServiceNamed('launcher')

    @property
    def scheduler(self):
        return self.app.getComponent(ISpiderScheduler)

    @property
    def eggstorage(self):
        return self.app.getComponent(IEggStorage)

    @property
    def poller(self):
        return self.app.getComponent(IPoller)

class Home(resource.Resource):

    def __init__(self, root, local_items):
        resource.Resource.__init__(self)
        self.root = root
        self.local_items = local_items

    def render_GET(self, txrequest):
        vars = {
            'projects': ', '.join(self.root.scheduler.list_projects())
        }
```

```

s = ""

<html>
<head><meta charset='UTF-8'><title>Scrapyd</title></head>
<body>
<h1>Scrapyd</h1>
<p>Available projects: <b>%(projects)s</b></p>
<ul>
<li><a href="/jobs">Jobs</a></li>
""" % vars
    if self.local_items:
        s += '<li><a href="/items/">Items</a></li>'
        s += ""
<li><a href="/logs/">Logs</a></li>
<li><a href="http://scrapyd.readthedocs.org/en/latest/">Documentation</a></li>
</ul>

<h2>How to schedule a spider?</h2>

<p>To schedule a spider you need to use the API (this web UI is only for
monitoring)</p>

<p>Example using <a href="http://curl.haxx.se/">curl</a>:</p>
<p><code>curl http://localhost:6800/schedule.json -d project=default -d spider=somespider</code></p>

<p>For more information about the API, see the <a href="http://scrapyd.readthedocs.org/en/latest/">Scra
</body>
</html>
""" % vars
    return s.encode('utf-8')

class Jobs(resource.Resource):

    def __init__(self, root, local_items):
        resource.Resource.__init__(self)
        self.root = root
        self.local_items = local_items

    def render(self, txrequest):
        cols = 10 ##### 8
        s = "<html><head><meta charset='UTF-8'><title>Scrapyd</title></head>"
        s += "<body>"
        s += "<h1>Jobs</h1>"
        s += "<p><a href='../>Go back</a></p>"
        s += "<table border='1'>"
        s += "<tr><th>Project</th><th>Spider</th><th>Job</th><th>PID</th><th>Start</th><th>Runtime</th>"
        if self.local_items:
            s += "<th>Items</th>"
            #cols = 9 #####
            cols += 1 #####
        s += "</tr>"
        s += "<tr><th colspan='%s' style='background-color: #ddd'>Pending</th></tr>" % cols
        for project, queue in self.root.poller.queues.items():
            for m in queue.list():
                s += "<tr>"
                s += "<td>%s</td>" % project
                s += "<td>%s</td>" % str(m['name'])
                s += "<td>%s</td>" % str(m['_job'])
                s += "</tr>"
            s += "<tr><th colspan='%s' style='background-color: #ddd'>Running</th></tr>" % cols
        for p in self.root.launcher.processes.values():
            s += "<tr>"
            for a in ['project', 'spider', 'job', 'pid']:
                s += "<td>%s</td>" % getattr(p, a)
            s += "<td>%s</td>" % p.start_time.replace(microsecond=0)
            s += "<td>%s</td>" % (datetime.now().replace(microsecond=0) - p.start_time.replace(microsec
            s += "<td></td>"
            s += "<td><a href='/logs/%s/%s/%s.log'>Log</a></td>" % (p.project, p.spider, p.job)
            s += "<td><a href='/logs/UTF-8.html?project=%s&spider=%s&job=%s' target='_blank'>UTF-8</a><"
            s += "<td><a href='/logs/scrapyd.html?opt=cancel&project=%s&job_or_spider=%s' target='_blar"
            if self.local_items:
                s += "<td><a href='/items/%s/%s/%s.jl'>Items</a></td>" % (p.project, p.spider, p.job)
            s += "</tr>"
            s += "<tr><th colspan='%s' style='background-color: #ddd'>Finished</th></tr>" % cols
        for p in self.root.launcher.finished:
            s += "<tr>"
            for a in ['project', 'spider', 'job']:

```

```

        s += "<td>%s</td>" % getattr(p, a)
    s += "<td></td>"
    s += "<td>%s</td>" % p.start_time.replace(microsecond=0)
    s += "<td>%s</td>" % (p.end_time.replace(microsecond=0) - p.start_time.replace(microsecond=0))
    s += "<td>%s</td>" % p.end_time.replace(microsecond=0)
    s += "<td><a href='/logs/%s/%s/%s.log'>Log</a></td>" % (p.project, p.spider, p.job)
    s += "<td><a href='/logs/UTF-8.html?project=%s&spider=%s&job=%s' target='_blank'>UTF-8</a></td>" % (p.project, p.spider, p.job)
    s += "<td><a href='/logs/scrapydl.html?opt=schedule&project=%s&job_or_spider=%s' target='_blank'>Scrapydl</a></td>" % (p.project, p.spider, p.job)
    if self.local_items:
        s += "<td><a href='/items/%s/%s/%s.jl'>Items</a></td>" % (p.project, p.spider, p.job)
    s += "</tr>"
s += "</table>"
s += "</body>"
s += "</html>"

txrequest.setHeader('Content-Type', 'text/html; charset=utf-8')
txrequest.setHeader('Content-Length', len(s))

return s.encode('utf-8')

```

/site-packages/scrapydl/website.py

3.新建 scrapydl.html

根据 <http://scrapydl.readthedocs.io/en/stable/config.html> 确定 Scrapydl 所使用的 logs_dir，在该目录下添加如下文件

□

```

<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>scrapydl</title>
</head>

<body>
<p>仅用于内网环境下执行 scrapydl API</p>
<div id="result"></div>

<script>
function parseQueryString(url) {
    var urlParams = {};
    url.replace(
        new RegExp("(^[?=&]+)(=([^\&]*))?", "g"),
        function($0, $1, $2, $3) {
            urlParams[$1] = $3;
        }
    );
    return urlParams;
}

function curl(opt, project, job_or_spider) {
    console.log(opt);
    console.log(project);
    console.log(job_or_spider);
    var formdata = new FormData();
    formdata.append('project', project);
    if(opt == 'cancel') {
        formdata.append('job', job_or_spider);
    } else {
        formdata.append('spider', job_or_spider);
    }

    var req = new XMLHttpRequest();
    req.onreadystatechange = function() {
        if (this.readyState == 4) {
            if (this.status == 200) {
                document.querySelector('#result').innerHTML = this.responseText;
            } else {
                alert('status code: ' + this.status);
            }
        } else {
            document.querySelector('#result').innerHTML = this.readyState;
        }
    };
}

```

```
req.open('post', window.location.protocol+'//'+window.location.host+'/'+opt+'.json', Async = true);
}

var kwargs = parseQueryString(location.search);
if (kwargs.opt == 'cancel' || kwargs.opt == 'schedule') {
    curl(kwargs.opt, kwargs.project, kwargs.job_or_spider);
}
}
</script>
</body>
</html>
```

scrapyd.html

4.实现效果

Jobs

[Go back](#)

Project	Spider	Job	PID	Start	Runtime	Finish	Log
Pending							
Running							
proxy	test	4b2e1376881211e8ac72b827ebc33e0b	22597	2018-07-15 17:34:38	0:00:11		Log UTF-8 STOP
Finished							

(1) 点击 STOP 超链接

```
仅用于内网环境下执行 scrapyd API

{"node_name": "pi-desktop", "status": "ok", "prevstate": "running"}
```

(2) 返回 Jobs 页面

Jobs

[Go back](#)

Project	Spider	Job	PID	Start	Runtime	Finish	Log
Pending							
Running							
Finished							
proxy	test	4b2e1376881211e8ac72b827ebc33e0b		2018-07-15 17:34:38	0:02:46	2018-07-15 17:37:24	Log UTF-8 START

(3) 点击 START 超链接

```
仅用于内网环境下执行 scrapyd API

{"node_name": "pi-desktop", "status": "ok", "jobid": "d1017358881211e8ac72b827ebc33e0b"}
```


(4) 返回 Jobs 页面

Jobs

[Go back](#)

Project	Spider	Job	PID	Start	Runtime	Finish	Log
Pending							
Running							
proxy	test	d1017358881211e8ac72b827ebc33e0b	22663	2018-07-15 17:38:27	0:00:08		Log UTF-8 STOP
Finished							
proxy	test	4b2e1376881211e8ac72b827ebc33e0b		2018-07-15 17:34:38	0:02:46	2018-07-15 17:37:24	Log UTF-8 START

posted @ 2018-07-15 18:47 [my8100](#) [阅读\(...\)](#) [评论\(...\)](#) [编辑](#) [收藏](#)

[scrapy相关: splash 实践](#)

0.

1.参考

<https://github.com/scrapy-plugins/scrapy-splash#configuration>

以此为准

[scrapy相关: splash安装 A javascript rendering service 渲染](#)

1. 启动 Docker Quickstart Terminal
2. 使用 putty 连接如下ip, 端口22, 用户名/密码: docker/tcuser
3. 开启服务:
 1. `sudo docker run -p 5023:5023 -p 8050:8050 -p 8051:8051 scrapinghub/splash`
4. 浏览器打开: <http://192.168.99.100:8050/>

docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at <https://docs.docker.com>

Start interactive shell

```
win7@win7-PC MINGW64 ~  
$
```

2.实践

2.1新建项目后修改 settings.py

ROBOTSTXT_OBEY 改为 False, 同时添加如下内容:

▣

```
'''https://github.com/scrapy-plugins/scrapy-splash#configuration'''  
# 1.Add the Splash server address to settings.py of your Scrapy project like this:  
SPLASH_URL = 'http://192.168.99.100:8050'  
  
# 2.Enable the Splash middleware by adding it to DOWNLOADER_MIDDLEWARES in your settings.py file  
# and changing HttpCompressionMiddleware priority:  
DOWNLOADER_MIDDLEWARES = {  
    'scrapy_splash.SplashCookiesMiddleware': 723,  
    'scrapy_splash.SplashMiddleware': 725,  
    'scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware': 810,  
}  
# Order 723 is just before HttpProxyMiddleware (750) in default scrapy settings.  
# HttpCompressionMiddleware priority should be changed in order to allow advanced response processing;  
# see https://github.com/scrapy/scrapy/issues/1895 for details.  
  
# 3.Enable SplashDeduplicateArgsMiddleware by adding it to SPIDER_MIDDLEWARES in your settings.py:  
SPIDER_MIDDLEWARES = {  
    'scrapy_splash.SplashDeduplicateArgsMiddleware': 100,  
}  
# This middleware is needed to support cache_args feature;  
# it allows to save disk space by not storing duplicate Splash arguments multiple times in a disk request  
# If Splash 2.1+ is used the middleware also allows to save network traffic by not sending these duplicate arguments  
  
# 4.Set a custom DUPEFILTER_CLASS:  
DUPEFILTER_CLASS = 'scrapy_splash.SplashAwareDupeFilter'  
  
# 5.If you use Scrapy HTTP cache then a custom cache storage backend is required.  
# scrapy-splash provides a subclass of scrapy.contrib.httpcache.FilesystemCacheStorage:  
HTTPCACHE_STORAGE = 'scrapy_splash.SplashAwareFSCacheStorage'  
# If you use other cache storage then it is necessary to subclass it  
# and replace all scrapy.util.request.request_fingerprint calls with scrapy_splash.splash_request_fingerprint  
  
# Note  
# Steps (4) and (5) are necessary because Scrapy doesn't provide a way to override request fingerprints  
# There are also some additional options available. Put them into your settings.py if you want to change them  
# SPLASH_COOKIES_DEBUG is False by default. Set to True to enable debugging cookies in the SplashCookieMiddleware  
# SPLASH_LOG_400 is True by default - it instructs to log all 400 errors from Splash. They are important for debugging  
# SPLASH_SLOT_POLICY is scrapy_splash.SlotPolicy.PER_DOMAIN by default. It specifies how concurrency &
```

2.2 编写基本 spider

```
# -*- coding: utf-8 -*-
import scrapy
from scrapy_splash import SplashRequest
from scrapy.shell import inspect_response
import base64
from PIL import Image
from io import BytesIO

class CnblogsSpider(scrapy.Spider):
    name = 'cnblogs'
    allowed_domains = ['cnblogs.com']
    start_urls = ['https://www.cnblogs.com/']

    def start_requests(self):
        for url in self.start_urls:
            yield SplashRequest(url, self.parse, args={'wait': 0.5})

    def parse(self, response):
        inspect_response(response, self) #####
```

调试 view(response) 是个txt。。。另存为 html 使用浏览器浏览即可。

2.3 编写截图 spider

同时参考 <https://stackoverflow.com/questions/45172260/scrapy-splash-screenshots>

```
def start_requests(self):
    splash_args = {
        'html': 1,
        'png': 1,
        #'width': 1024, #默认1027*768,4:3
        #'render_all': 1, #长图截屏, 不提供则是第一屏, 需要同时提供 wait, 否则报错
        #'wait': 0.5,
    }

    for url in self.start_urls:
        yield SplashRequest(url, self.parse, endpoint='render.json', args=splash_args)
```

<http://splash.readthedocs.io/en/latest/api.html?highlight=wait#render-png>

render_all=1 requires non-zero [wait](#) parameter. This is an unfortunate restriction, but it seems that this is the only way to make rendering work reliably with render_all=1.

<https://github.com/scrapy-plugins/scrapy-splash#responses>

Responses

scrapy-splash returns Response subclasses for Splash requests:

- SplashResponse is returned for binary Splash responses - e.g. for /render.png responses;
- SplashTextResponse is returned when the result is text - e.g. for /render.html responses;
- **SplashJsonResponse** is returned when the result is a JSON object - e.g. for /render.json responses or /execute responses when script returns a Lua table.

SplashJsonResponse provide extra features:

- response.data attribute contains response data decoded from JSON; you can access it like response.data['html'].

show 另存文件

```
def parse(self, response):
    # In [6]: response.data.keys()
    # Out[6]: [u'title', u'url', u'geometry', u'html', u'png', u'requestedUrl']

    imgdata = base64.b64decode(response.data['png'])
    img = Image.open(BytesIO(imgdata))
    img.show()
    filename = 'some_image.png'
```

```
with open(filename, 'wb') as f:
    f.write(imgdata)
inspect_response(response, self) #####
```

x

posted @ 2017-10-19 17:56 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

[wb 黑名单批量操作](#)

0. 参考

[yu961549745/WeiboBlackList 微博批量拉黑](#)

1. 代码 block.py

更新内容：多线程，urllib.request 改为 requests + session

改成从 firefox 或 chrome 读取 cookie 更方便，懒得改了

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-

import requests
from requests.exceptions import (ConnectionError, ConnectTimeout, ReadTimeout, SSLError,
                                 ProxyError, RetryError, InvalidSchema)

import threading
import Queue
import traceback

import logging
def get_logger():
    logger = logging.getLogger("threading_example")
    logger.setLevel(logging.DEBUG)

    # fh = logging.FileHandler("d:/threading.log")
    fh = logging.StreamHandler()
    fmt = '%(asctime)s - %(threadName)-10s - %(levelname)s - %(message)s'
    formatter = logging.Formatter(fmt)
    fh.setFormatter(formatter)

    logger.addHandler(fh)
    return logger

logger = get_logger()

def block():
    while True:
        try:
            uid = task_queue.get()
            data = dict(payload) #dict
            data.update({'uid': uid})
            resp = s.post(url, data=data)
        except (ConnectionError, ConnectTimeout, ReadTimeout, SSLError,
                ProxyError, RetryError, InvalidSchema) as err:
            task_queue.task_done() ##### 重新 put 之前需要 task_done , 才能保证释放 task_queue.j
            task_queue.put(uid)
        except Exception as err:
            logger.debug(u'\nuid: {}\nerr: {}\ntraceback: {}'.format(uid, err, traceback.format_exc()))
            task_queue.task_done() ##### 重新 put 之前需要 task_done , 才能保证释放 task_queue.j
            task_queue.put(uid)
        else:
            try:
                code = resp.json()['code']
                if code != '100000':
                    logger.debug(u'uid: {} code: {}'.format(uid, code))
                else:
                    logger.debug(u'uid: {}'.format(uid))
            except Exception as err:
                logger.debug(u'\nuid: {}\nresp: {}\nerr: {}\ntraceback: {}'.format(uid, resp.text, err,
                finally:
                    task_queue.task_done()

if __name__ == '__main__':

    # lines: request in raw format captured from Fiddler
    '''
    ['POST http://weibo.com/aj/filter/block?ajwvr=6 HTTP/1.1',
     'Host: weibo.com',
     'User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:56.0) Gecko/20100101 Firefox/56.0',
```

```
'Accept: */*',
'Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3',
'Accept-Encoding: gzip, deflate',
'Content-Type: application/x-www-form-urlencoded',
'X-Requested-With: XMLHttpRequest',
'Referer: http://weibo.com/u/5471246591?is_hot=1',
'Content-Length: 57',
'Cookie: your cookie#####',
'Connection: keep-alive',
'',
'uid=5471246591&filter_type=1&status=1&interact=1&follow=1']
'''
```

移除黑名单:

```
# In [317]: url2='http://weibo.com/aj/f/delblack?ajwvr=6'
# In [318]: resp = s.post(url2, data={'uid':'5209943797'})
```

```
# In [320]: resp.json()
# Out[320]: {'code': '100000', 'data': {}, 'msg': ''}
```

```
with open('uids.txt') as f:
    uids = [uid.strip() for uid in f.readlines()]
```

```
with open('headers.txt') as f:
    lines = [i.strip() for i in f.readlines()]
```

```
url = lines[0].split()[1]
```

```
headers = {}
for line in lines[2:-2]:
    k, v = line.split(':',1) #space
    headers[k] = v
```

```
data=lines[-1]
# datas = [re.sub(r'uid=\d+', 'uid=%s'%uid ,data) for uid in uids]
payload = dict([i.split('=',1) for i in data.split('&')])
# payloads = [payload.update({'uid': uid}) for uid in uids]
# payloads = []
# for uid in uids:
#     payload.update({'uid': uid})
#     payloads.append(dict(payload)) ### dict
```

```
# for payload in payloads:
#     r = requests.post(url, headers=headers, data=payload)
# r.text
# {"code":"100000","msg":"\u96b1\u85cf\u6210\u529f","data":{}}
# In [287]: r.json()['code']
# Out[287]: u'100000'
```

```
s = requests.Session()
s.headers = headers
s.mount('http://', requests.adapters.HTTPAdapter(pool_connections=1000, pool_maxsize=1000))
```

```
task_queue = Queue.Queue()
for uid in uids:
    task_queue.put(uid)
```

```
threads = []
for i in range(100):
    t = threading.Thread(target=block) #args接收元组, 至少(a,)
    threads.append(t)
```

```
for t in threads:
    t.setDaemon(True)
    t.start()
```

```
task_queue.join()
print 'task done'
```


[scrapy相关: splash安装 A javascript rendering service 渲染](#)

0.

splash: 美人鱼 溅, 泼

1.参考

[Splash使用初体验](#)

docker在windows下的安装

<https://blog.scrapinghub.com/2015/03/02/handling-javascript-in-scrapy-with-splash/>

[Splash](#) is our in-house solution for JavaScript rendering, implemented in Python using [Twisted](#) and [QT](#). 官方博客介绍, splash 是 scrapinghub 的内部解决方案? ? ?

<https://scrapinghub.com/>

We're the creators and the main maintainers of Scrapy. 创始人和维护者...背后的大佬

[github: scrapinghub/splash](#)

Splash is a javascript rendering service with an HTTP API. It's a lightweight browser with an HTTP API, implemented in Python 3 using Twisted and QT5.

It's fast, lightweight and state-less which makes it easy to distribute. 用于渲染js页面

<http://splash.readthedocs.io/en/latest/index.html>

splash 官方文档

[github: scrapy-plugins/scrapy-splash](#)

This library provides [Scrapy](#) and JavaScript integration using [Splash](#). 如何在 scrapy 中使用 splash

<http://splash.readthedocs.io/en/stable/api.html#request-filters>

Splash supports filtering requests based on [Adblock Plus](#) rules. 还没有搞定

2.安装使用

<https://stackoverflow.com/questions/30345623/scraping-dynamic-content-using-python-scrapy>

提到 [ScrapyJS](#), 但是链接地址跳转 <https://github.com/scrapy-plugins/scrapy-splash#installation>

<https://pypi.python.org/pypi/scrapyjs>

<https://pypi.python.org/pypi/scrapy-splash>

2.1 安装 scrapy-splash

```
C:\Users\win7>pip install scrapy-splash
Collecting scrapy-splash
  Downloading scrapy_splash-0.7.2-py2.py3-none-any.whl
Installing collected packages: scrapy-splash
Successfully installed scrapy-splash-0.7.2
```

2.2 通过 docker 安装 image: scrapinghub/splash

官网找到下载链接

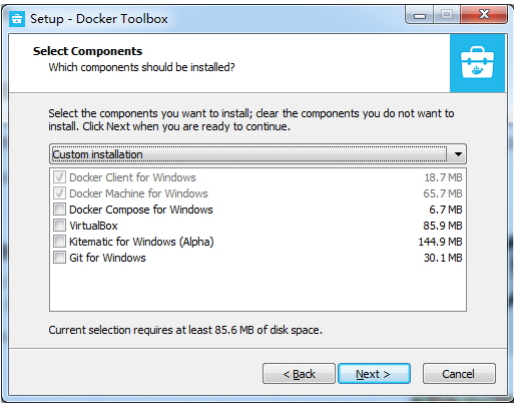
<https://store.docker.com/editions/community/docker-ce-desktop-windows>

Get Docker Community Edition for Windows

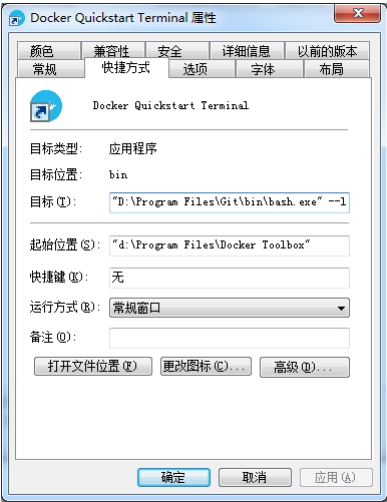
Docker for Windows is available for free.

Requires Microsoft Windows 10 Professional or Enterprise 64-bit. [For previous versions get Docker Toolbox.](#)

右键管理员安装，最好勾选非必要项？？？



右键管理员启动 Docker Quickstart Terminal ，提示没找到 bash.exe



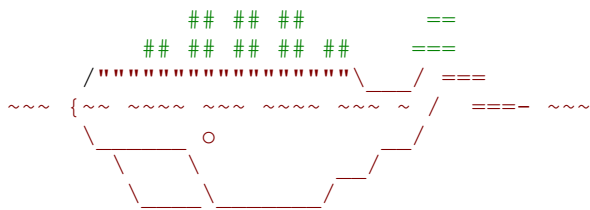
输出：

□

```
Creating CA: C:\Users\win7\.docker\machine\certs\ca.pem
Creating client certificate: C:\Users\win7\.docker\machine\certs\cert.pem
Running pre-create checks...
(default) Image cache directory does not exist, creating it at C:\Users\win7\.docker\machine\cache...
(default) No default Boot2Docker ISO found locally, downloading the latest release...
(default) Latest release for github.com/boot2docker/boot2docker is v17.09.0-ce
(default) Downloading C:\Users\win7\.docker\machine\cache\boot2docker.iso from https://github.com/boot2docker/boot2docker/releases/download/v17.09.0-ce/boot2docker.iso
(default) 0%....10%....20%....30%....40%....50%....60%....70%....80%....90%....100%
Creating machine...
(default) Copying C:\Users\win7\.docker\machine\cache\boot2docker.iso to C:\Users\win7\.docker\machine\boot2docker.iso
(default) Creating VirtualBox VM...
(default) Creating SSH key...
(default) Starting the VM...
(default) Check network to re-create if needed...
(default) Windows might ask for the permission to create a network adapter. Sometimes, such confirmation dialog is shown. If yes, please click on Yes.
(default) Found a new host-only adapter: "VirtualBox Host-Only Ethernet Adapter #2"
(default) Windows might ask for the permission to configure a network adapter. Sometimes, such confirmation dialog is shown. If yes, please click on Yes.
(default) Windows might ask for the permission to configure a dhcp server. Sometimes, such confirmation dialog is shown. If yes, please click on Yes.
(default) Waiting for an IP...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with boot2docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: D:\
```

##

.



docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at <https://docs.docker.com>

Start interactive shell

```
win7@win7-PC MINGW64 ~
$ docker info
Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
Images: 0
Server Version: 17.09.0-ce
Storage Driver: aufs
  Root Dir: /mnt/sda1/var/lib/docker/aufs
  Backing Filesystem: extfs
  Dirs: 0
  Dirperm1 Supported: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
  Volume: local
  Network: bridge host macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file logentries splunk syslog
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 06b9cb35161009dcb7123345749fef02f7cea8e0
runc version: 3f2f8b84a77f73d38244dd690525642a72156c64
init version: 949e6fa
Security Options:
  seccomp
   Profile: default
Kernel Version: 4.4.89-boot2docker
Operating System: Boot2Docker 17.09.0-ce (TCL 7.2); HEAD : 06d5c35 - Wed Sep 27 23:22:43 UTC 2017
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 995.8MiB
Name: default
ID: 033J:6GDF:AQ6P:RBM7:6KLF:OZHY:2N3J:QZKV:YIJT:G3AI:XCPD:NZ3G
Docker Root Dir: /mnt/sda1/var/lib/docker
Debug Mode (client): false
Debug Mode (server): true
  File Descriptors: 17
  Goroutines: 26
  System Time: 2017-10-18T09:58:42.414047781Z
  EventsListeners: 0
Registry: https://index.docker.io/v1/
Labels:
  provider=virtualbox
Experimental: false
Insecure Registries:
  127.0.0.0/8
Live Restore Enabled: false
```

```
win7@win7-PC MINGW64 ~
$ ipconfig
```

Windows IP 配置

以太网适配器 lan:

```
    连接特定的 DNS 后缀 . . . . . : 
    本地链接 IPv6 地址. . . . . : fe80::f950:bf55:726b:b7a6%14
```

```
IPv4 地址 . . . . . : 192.168.144.100
子网掩码 . . . . . : 255.255.255.0
默认网关 . . . . . : 192.168.144.254
```

以太网适配器 VirtualBox Host-Only Network #2:

```
连接特定的 DNS 后缀 . . . . . :
本地链接 IPv6 地址 . . . . . : fe80::1c18:13ad:7ed2:c0ff%29
IPv4 地址 . . . . . : 192.168.99.1
子网掩码 . . . . . : 255.255.255.0
默认网关 . . . . . :
```

隧道适配器 isatap.{CE007B04-2C7A-4A52-8BBF-1BCB4682EEB9}:

```
媒体状态 . . . . . : 媒体已断开
连接特定的 DNS 后缀 . . . . . :
```

隧道适配器 Teredo Tunneling Pseudo-Interface:

```
媒体状态 . . . . . : 媒体已断开
连接特定的 DNS 后缀 . . . . . :
```

隧道适配器 isatap.{93C68FD9-301C-484C-AFCB-5549CA24453B}:

```
媒体状态 . . . . . : 媒体已断开
连接特定的 DNS 后缀 . . . . . :
```

```
win7@win7-PC MINGW64 ~
$
```

[View Code](#)

里面重要信息:

```
(default) Copying C:\Users\win7\.docker\machine\cache\boot2docker.iso to C:\Users\win7\.docker\machine\
(default) Creating VirtualBox VM...
```

```
docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at https://docs.docker.com
```

putty 连接:

```
192.168.99.100
22
```

```
docker
tcuser
```

第一次需要从docker hub下载相关镜像文件

```
sudo docker pull scrapinghub/splash
```

后面每次启动splash服务，并通过http，https，telnet提供服务

```
#通常一般使用http模式 ，可以只启动一个8050就好
#Splash 将运行在 0.0.0.0 at ports 8050 (http), 8051 (https) and 5023 (telnet).
sudo docker run -p 5023:5023 -p 8050:8050 -p 8051:8051 scrapinghub/splash
```

浏览器打开

<http://192.168.99.100:8050>

posted @ 2017-10-19 17:45 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

scrapy相关 通过设置 FEED_EXPORT_ENCODING 解决 unicode 中文写入json文件出现\uXXXX

0.问题现象

爬取 item:

```
2017-10-16 18:17:33 [scrapy.core.scrapers] DEBUG: Scraped from <200 https://www.huxiu.com/v2_action/arti
{'author': u'\u5546\u4e1a\u8bc4\u8bba\u7cbe\u9009\u9a9',
 'cmt': 5,
 'fav': 194,
 'time': u'4\u5929\u524d',
 'title': u'\u96f7\u519b\u8c08\u5c0f\u7c73\u201c\u65b0\u96f6\u552e\u201d\u51a\u50cfZara\u4e00\u6837\u51
 'url': u'/article/217755.html'}
```

写入jsonline.jl 文件

```
{ "title": "\u8fd9\u4e00\u5468\u2011\u82b2\u7a77\u66b4\u51fb", "url": "/article/217997.html", "author":  
{ "title": "\u502a\u840d\u8001\u516c\u7684\u65b0\u620f\u6251\u8857\u4e86\u2011\u9ec4\u6e24\u6301\u80a1\u2011
```

item 被转 str, 默认 ensure ascii = True, 则非 ASCII 字符被转化为 '\uXXXX', 每一个 '{xxx}' 单位被写入文件

目标: 注意最后用 chrome 或 notepad++ 打开确认, firefox 打开 jl 可能出现中文乱码, 需要手动指定编码。

```
{ "title": "这一周：贫穷暴击", "url": "/article/217997.html", "author": "虎嗅", "fav": 8, "time": "2天前", '
{ "title": "倪萍老公的新戏扑街了，黄渤持股的公司要赔惨了", "url": "/article/217977.html", "author": "娱乐资本论"
```

1.参考资料

scrapy抓取到中文,保存到json文件为unicode,如何解决.


```
import json
import codecs

class JsonWithEncodingPipeline(object):

    def __init__(self):
        self.file = codecs.open('scraped_data_utf8.json', 'w', encoding='utf-8')

    def process_item(self, item, spider):
        line = json.dumps(dict(item), ensure_ascii=False) + "\n"
        self.file.write(line)
        return item

    def close_spider(self, spider):
        self.file.close()
```

View Code

scrapy中输出中文保存中文

Scrapy爬虫框架抓取中文结果为Unicode编码，如何转换UTF-8编码

[lidashuang / imax-spider](#)

以上资料实际上就是官方文档举的 pipeline 例子，另外指定 `ensure_ascii=False`

Write items to a JSON file

The following pipeline stores all scraped items (from all spiders) into a single `items.jsonl` file, containing one item per line serialized in JSON format:

```
import json

class JsonWriterPipeline(object):

    def open_spider(self, spider):
        self.file = open('items.json', 'w')

    def close_spider(self, spider):
```

```
self.file.close()
```

```
def process_item(self, item, spider):
    line = json.dumps(dict(item)) + "\n" #另外指定 ensure_ascii=False
    self.file.write(line)
    return item
```

Note

The purpose of JsonWriterPipeline is just to introduce how to write item pipelines. If you really want to store all scraped items into a JSON file you should use the [Feed exports](#).

2.更好的解决办法:

[scrapy 使用item export输出中文到json文件, 内容为unicode码, 如何输出为中文?](#)

<http://stackoverflow.com/questions/18337407/saving-utf-8-texts-in-json-dumps-as-utf8-not-as-u-escape-sequence> 里面有提到, 将 JSONEncoder 的 ensure_ascii 参数设为 False 即可。

而 scrapy 的 item export 文档里有提到

The additional constructor arguments are passed to the BaseItemExporter constructor, and the leftover arguments to the JSONEncoder constructor, so you can use any JSONEncoder constructor argument to customize this exporter.

因此就在调用 scrapy.contrib.exporter.JsonItemExporter 的时候额外指定 ensure_ascii=False 就可以啦。

3.根据上述解答, 结合官网和源代码, 直接解决办法:

1.可以通过修改 project settings.py 补充 FEED_EXPORT_ENCODING = 'utf-8'

2.或在cmd中传入 G:\pydata\pycode\scrapy\huxiu_com>scrapy crawl -o new.jl -s FEED_EXPORT_ENCODING='utf-8' huxiu

<https://doc.scrapy.org/en/latest/topics/feed-exports.html#feed-export-encoding>

FEED_EXPORT_ENCODING

Default: None

The encoding to be used for the feed.

If unset or set to None (default) it uses UTF-8 for everything except JSON output, which uses safe numeric encoding (\uXXXX sequences) for historic reasons.

Use utf-8 if you want UTF-8 for JSON too.

```
In [615]: json.dump?
Signature: json.dump(obj, fp, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, c
Docstring:
Serialize ``obj`` as a JSON formatted stream to ``fp`` (a
``.write()``-supporting file-like object).
```

If ``ensure_ascii`` is true (the default), all non-ASCII characters in the output are escaped with ``\uXXXX`` sequences, and the result is a ``str`` instance consisting of ASCII characters only. If ``ensure_ascii`` is ``False``, some chunks written to ``fp`` may be ``unicode`` instances. This usually happens because the input contains unicode strings or the ``encoding`` parameter is used. Unless ``fp.write()`` explicitly understands ``unicode`` (as in ``codecs.getwriter()``) this is likely to cause an error.

```
class JsonLinesItemExporter(BaseItemExporter):

    def __init__(self, file, **kwargs):
        kwargs.setdefault('ensure_ascii', not self.encoding)

class JsonItemExporter(BaseItemExporter):
    def __init__(self, file, **kwargs):
        kwargs.setdefault('ensure_ascii', not self.encoding)

class XmlItemExporter(BaseItemExporter):

    def __init__(self, file, **kwargs):
        if not self.encoding:
            self.encoding = 'utf-8'
```

posted @ 2017-10-16 18:30 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

python之PIL 二值图像处理和保存

0.

1.参考

<http://pszpcl.baike.com/article-77327.html>

windows 图片右键：属性 详细信息 位深度

位深度 用于指定图像中的每个像素可以使用的颜色信息数量。

位深度为 1 的图像的像素有两个可能的值：黑色和白色。

位深度为 8 的灰度模式图像有 256 个可能的灰色值。

RGB 图像由三个颜色通道组成。8 位/像素的 RGB 图像中的每个通道有 256 个可能的值，这意味着该图像有 1600 万个以上可能的颜色值。

有时将带有 8 位/通道 (bpc) 的 RGB 图像称作 24 位图像（8 位 x 3 通道 = 24 位数据/像素）。

2.结论：

载入黑白图片也可能是0/1二值

保存黑白图片优选 gif，自动转为0/1二值，且windows下显示正常。

3. 载入图片后的探索

#载入黑白图

```
In [82]: img.mode
Out[82]: 'P'
In [114]: img.getbands()
Out[114]: ('P',)
In [83]: img.getcolors()
Out[83]: [(1048, 0), (102, 1)]
In [84]: print list(img.getdata())[:10]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
In [90]: set(img.getdata())
Out[90]: {0, 1}
```

#载入彩色图

```
In [103]: imgc.mode
Out[103]: 'RGBA'
In [100]: imgc.getbands()
Out[100]: ('R', 'G', 'B', 'A')
In [106]: imgc.getcolors()
Out[106]:
[(34, (255, 255, 255, 255)),
 (11, (250, 215, 245, 255)),
```



```
In [112]: list(imgc.getdata())
Out[112]:
[(244, 245, 245, 255),
 (245, 245, 245, 255),
```

4. 保存二值黑白图片要区分 gif / png

```
In [176]: img=Image.open('split_image/61234.png')
In [177]: img.show()
```

```
In [178]: img.getcolors() #注意原图 0/1 足以显示为黑白图像
Out[178]: [(1048, 0), (102, 1)]
```

```
In [179]: gray = img.convert('L')
In [180]: gray_array = np.array(gray)
In [181]: Image.fromarray(gray_array).show()
In [182]: gray_array
Out[182]:
array([[249, 249, 249, ..., 249, 249, 249],
       [249, 249, 249, ..., 249, 249, 249],
       [249, 249, 249, ..., 249, 249, 249],
       ...,
       [249, 249, 249, ..., 249, 249, 249],
       [249, 249, 249, ..., 249, 249, 249],
       [249, 249, 249, ..., 249, 249, 249]], dtype=uint8)
```

```
In [183]: bilevel = Image.fromarray(np.where(gray_array<100,0,200))
In [184]: bilevel.getcolors()
Out[184]: [(1048, 200), (102, 0)]
```

```
In [185]: bilevel.save('png.png')
In [186]: bilevel.save('gif.gif')
```

```
In [187]: Image.open('png.png').getcolors() #png灰度值得以保留
Out[187]: [(1048, 200), (102, 0)]
```

```
In [188]: Image.open('gif.gif').getcolors() #gif只剩0/1
Out[188]: [(102, 0), (1048, 1)]
```

原图png

61234

PIL处理后保存的gif:

61234

PIL处理后保存的png，在windows中也显示为黑图，PIL show显示正常:



d

posted @ 2017-08-25 15:45 [my8100](#) 阅读(...) 评论(...) [编辑](#) [收藏](#)

