Day09回顾

settings.py常用变量

```
# 1、设置日志级别
LOG_LEVEL = ''
# 2、保存到日志文件(不在终端输出)
LOG_FILE = ''
# 3、设置数据导出编码(主要针对于json文件)
FEED_EXPORT_ENCODING = ''
# 4、非结构化数据存储路径
IMAGES_STORE = '路径'
# 5、设置User-Agent
USER_AGENT = ''
# 6、设置最大并发数(默认为16)
CONCURRENT_REQUESTS = 32
# 7、下载延迟时间(每隔多长时间请求一个网页)
DOWNLOAD_DELAY = 3
# 8、请求头
DEFAULT_REQUEST_HEADERS = {}
# 9、添加项目管道
ITEM_PIPELINES = {}
# 10、添加下载器中间件
DOWNLOADER_MIDDLEWARES = {}
```

非结构化数据抓取

```
1、spider
   yield item['链接']
2、pipelines.py
   from scrapy.pipelines.images import ImagesPipeline
   import scrapy
   class TestPipeline(ImagesPipeline):
      def get_media_requests(self,item,info):
            yield scrapy.Request(url=item['url'],meta={'item':item['name']})
      def file_path(self,request,response=None,info=None):
            name = request.meta['item']
            filename = name
            return filename
3、settings.py
   IMAGES_STORE = 'D:\\Spider\\images'
```

scrapy.Request()

```
# 参数
1、url
2、callback
3、headers
4、meta ： 传递数据,定义代理
5、dont_filter ： 是否忽略域组限制 - 默认False,检查allowed_domains['']
# request属性
1、request.url
2、request.headers
3、request.meta
4、request.method
# response属性
1、response.url
2、response.text
3、response.body
4、response.meta
5、response.encoding
```

设置中间件

随机User-Agent

```
# 1、middlewares.py
class RandomUaDownloaderMiddleware(object):
    def process_request(self,request,spider):
        request.header['User-Agent'] = xxx
# 2、settings.py
DOWNLOADER_MIDDLEWARES = {'xxx.middlewares.xxx':300}
```

随机代理

```
# 1、middlewares.py
class RandomProxyDownloaderMiddleware(object):
    def process_request(self,request,spider):
        request.meta['proxy'] = xxx

    def process_exception(self,request,exception,spider):
        return request
# 2、settings.py
DOWNLOADER_MIDDLEWARES = {'xxx.middlewares.xxx':200}
```

Day10笔记

item对象到底该在何处创建?

```
1、一级页面:  都可以,建议在for循环外
2、>=2级页面: for循环内
```

分布式爬虫

分布式爬虫介绍

- **原理**

```
多台主机共享1个爬取队列
```

- **实现**

```
重写scrapy调度器(scrapy_redis模块)
```

- **为什么使用redis**

```
1、Redis基于内存,速度快
2、Redis非关系型数据库,Redis中集合,存储每个request的指纹
3、scrapy_redis安装
    sudo pip3 install scrapy_redis
```

Redis使用

- **windows安装客户端使用**

```
1、服务端启动 : cmd命令行 -> redis-server.exe
   客户端连接 : cmd命令行 -> redis-cli.exe
```

scrapy_redis详解

- **GitHub地址**

```
https://github.com/rmax/scrapy-redis
```

- **settings.py说明**

```
# 重新指定调度器: 启用Redis调度存储请求队列
SCHEDULER = "scrapy_redis.scheduler.Scheduler"
```

```
3
4  # 重新指定去重机制：确保所有的爬虫通过Redis去重
5  DUPEFILTER_CLASS = "scrapy_redis.dupefilter.RFPDupeFilter"
6
7  # 不清除Redis队列：暂停/恢复/断点续爬
8  SCHEDULER_PERSIST = True
9
10 # 优先级队列 （默认）
11 SCHEDULER_QUEUE_CLASS = 'scrapy_redis.queue.PriorityQueue'
12 #可选用的其它队列
13 # 先进先出队列
14 SCHEDULER_QUEUE_CLASS = 'scrapy_redis.queue.FifoQueue'
15 # 后进先出队列
16 SCHEDULER_QUEUE_CLASS = 'scrapy_redis.queue.LifoQueue'
17
18 # redis管道
19 ITEM_PIPELINES = {
20     'scrapy_redis.pipelines.RedisPipeline': 300
21 }
22
23 #指定连接到redis时使用的端口和地址
24 REDIS_HOST = 'localhost'
25 REDIS_PORT = 6379
```

腾讯招聘分布式改写

1、正常项目数据抓取（非分布式）

2、改写为分布式（同时存入redis）

1、settings.py

```
1  # 使用scrapy_redis的调度器
2  SCHEDULER = "scrapy_redis.scheduler.Scheduler"
3  # 使用scrapy_redis的去重机制
4  DUPEFILTER_CLASS = "scrapy_redis.dupefilter.RFPDupeFilter"
5  # 是否清除请求指纹,True:不清除 False:清除
6  SCHEDULER_PERSIST = True
7  # 在ITEM_PIPELINES中添加redis管道
8  'scrapy_redis.pipelines.RedisPipeline': 200
9  # 定义redis主机地址和端口号
10 REDIS_HOST = '111.111.111.111'
11 REDIS_PORT = 6379
```

改写为分布式（同时存入mysql）

- **修改管道**

```
ITEM_PIPELINES = {
   'Tencent.pipelines.TencentPipeline': 300,
   # 'scrapy_redis.pipelines.RedisPipeline': 200
   'Tencent.pipelines.TencentMysqlPipeline':200,
}
```

- **清除redis数据库**

```
flushdb
```

- **代码拷贝一份到分布式中其他机器，两台或多台机器同时执行此代码**

腾讯招聘分布式改写- 方法二

- **使用redis_key改写**

```
# 第一步: settings.py无须改动
settings.py和上面分布式代码一致
# 第二步:tencent.py
from scrapy_redis.spiders import RedisSpider
class TencentSpider(RedisSpider):
    # 1. 去掉start_urls
    # 2. 定义redis_key
    redis_key = 'tencent:spider'
    def parse(self,response):
        pass
# 第三步:把代码复制到所有爬虫服务器，并启动项目
# 第四步
  到redis命令行，执行LPUSH命令压入第一个要爬取的URL地址
  >LPUSH tencent:spider 第1页的URL地址

# 项目爬取结束后无法退出，如何退出?
setting.py
CLOSESPIDER_TIMEOUT = 3600
# 到指定时间(3600秒)时,会自动结束并退出
```

scrapy - post请求

- **方法+参数**

```
scrapy.FormRequest(
    url=posturl,
    formdata=formdata,
    callback=self.parse
)
```

- **有道翻译案例实现**

1、创建项目+爬虫文件

```
scrapy startproject Youdao
cd Youdao
scrapy genspider youdao fanyi.youdao.com
```

2、items.py

```
result = scrapy.Field()
```

3、youdao.py

```

```

4、settings.py

```
1、ROBOTSTXT_OBEY = False
2、LOG_LEVEL = 'WARNING'
3、COOKIES_ENABLED = False
4、DEFAULT_REQUEST_HEADERS = {
      "Cookie": "OUTFOX_SEARCH_USER_ID=970246104@10.169.0.83; OUTFOX_SEARCH_USER_ID_NCOO=570559528.1224236; _ntes_nnid=96bc13a2f5ce64962adfd6a278467214,1551873108952; JSESSIONID=aaae9i7plXPlKaJH_gkYw; td_cookie=18446744072941336803; SESSION_FROM_COOKIE=unknown; ___rl__test__cookies=1565689460872",
      "Referer": "http://fanyi.youdao.com/",
      "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36",
    }
```

scrapy添加cookie的三种方式

```
# 1、修改 settings.py 文件
1、COOKIES_ENABLED = False  取消注释
2、DEFAULT_REQUEST_HEADERS = {}   添加Cookie

# 2、DownloadMiddleware
def process_request(self,request,spider):
    request.cookies={}

# 3、爬虫文件
def start_requests(self):
    yield scrapy.Request(url=url,cookies={},callback=xxx)
```

机器视觉与tesseract

作用

```
处理图形验证码
```

三个重要概念

- OCR

```
# 定义
OCR：光学字符识别(Optical Character Recognition)
# 原理
通过扫描等光学输入方式将各种票据、报刊、书籍、文稿及其它印刷品的文字转化为图像信息，再利用文字识别技术将图像信息转化为电子文本
```

- tesserct-ocr

```
OCR的一个底层识别库（不是模块，不能导入）
# Google维护的开源OCR识别库
```

- pytesseract

```
Python模块,可调用底层识别库
# 对tesseract-ocr做的一层Python API封装
```

安装tesseract-ocr

- Ubuntu

```
sudo apt-get install tesseract-ocr
```

- Windows

```
1、下载安装包
2、添加到环境变量(Path)
```

- 测试

```
# 终端 | cmd命令行
tesseract xxx.jpg 文件名
```

安装pytesseract

- 安装

```
sudo pip3 install pytesseract
```

- 使用

```
import pytesseract
# Python图片处理标准库
from PIL import Image

# 创建图片对象
img = Image.open('test1.jpg')
# 图片转字符串
result = pytesseract.image_to_string(img)
print(result)
```

- 爬取网站思路（验证码）

```
1、获取验证码图片
2、使用PIL库打开图片
3、使用pytesseract将图片中验证码识别并转为字符串
4、将字符串发送到验证码框中或者某个URL地址
```

在线打码平台

- **为什么使用在线打码**

```
tesseract-ocr识别率很低,文字变形、干扰，导致无法识别验证码
```

- **云打码平台使用步骤**

```
1、下载并查看接口文档
2、调整接口文档，调整代码并接入程序测试
3、真正接入程序，在线识别后获取结果并使用
```

- **破解云打码网站验证码**

1、下载并调整接口文档，封装成函数，打码获取结果

```

```

2、访问云打码网站，获取验证码并在线识别

```

```

Fiddler抓包工具

- **配置Fiddler**

```
# 添加证书信任
1、Tools - Options - HTTPS
   勾选 Decrypt Https Traffic 后弹出窗口，一路确认
# 设置只抓取浏览器的数据包
2、...from browsers only
# 设置监听端口（默认为8888）
3、Tools - Options - Connections
# 配置完成后重启Fiddler（重要）
4、关闭Fiddler,再打开Fiddler
```

- **配置浏览器代理**

```
1、安装Proxy SwitchyOmega插件
2、浏览器右上角：SwitchyOmega->选项->新建情景模式->AID1901(名字)->创建
   输入 ：HTTP://  127.0.0.1  8888
   点击 ： 应用选项
3、点击右上角SwitchyOmega可切换代理
```

- **Fiddler常用菜单**

```
1、Inspector ： 查看数据包详细内容
   整体分为请求和响应两部分
2、常用菜单
   Headers ： 请求头信息
   WebForms: POST请求Form表单数据 ： <body>
   GET请求查询参数: <QueryString>
   Raw
   将整个请求显示为纯文本
```

移动端app数据抓取

-------让我来告诉你-------