Day09回顾

日志级别

```
DEBUG < INFO < WARNING < ERROR < CRITICAL
```

数据持久化存储(MySQL、MongoDB)

```
1、settings.py  : 定义相关变量
2、pipelines.py : 新建类
3、settings.py  : 添加管道

# 注意 : process_item() 函数中一定要 return item
```

保存为csv、json文件

- 命令格式

```
scrapy crawl maoyan -o maoyan.csv
scrapy crawl maoyan -o maoyan.json
# settings.py  FEED_EXPORT_ENCODING = 'utf-8'
```

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 会影响 CONCURRENT_REQUESTS，不能使并发显现
# 有CONCURRENT_REQUESTS，没有DOWNLOAD_DELAY： 服务器会在同一时间收到大量的请求
# 有CONCURRENT_REQUESTS，有DOWNLOAD_DELAY 时，服务器不会在同一时间收到大量的请求
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
3、settings.py
```

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['']
```

设置中间件

整体步骤

```
# middlewares.py
class xxx(object):
    def process_request(self,request,spider):
        pass
```

随机User-Agent

```
class RandomUaDownloaderMiddleware(object):
    def process_request(self,request,spider):
        request.header['User-Agent'] = xxx
```

随机代理

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

    def process_exception(self,request,exception,spider):
        return request
```

Day10笔记

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

- **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"

# 重新指定去重机制：确保所有的爬虫通过Redis去重
DUPEFILTER_CLASS = "scrapy_redis.dupefilter.RFPDupeFilter"

# 不清除Redis队列：暂停/恢复/断点续爬
SCHEDULER_PERSIST = True

# 优先级队列 （默认）
SCHEDULER_QUEUE_CLASS = 'scrapy_redis.queue.PriorityQueue'
#可选用的其它队列
# 先进先出队列
SCHEDULER_QUEUE_CLASS = 'scrapy_redis.queue.FifoQueue'
# 后进先出队列
SCHEDULER_QUEUE_CLASS = 'scrapy_redis.queue.LifoQueue'

# redis管道
ITEM_PIPELINES = {
    'scrapy_redis.pipelines.RedisPipeline': 300
}

#指定连接到redis时使用的端口和地址（可选）
REDIS_HOST = 'localhost'
REDIS_PORT = 6379
```

腾讯招聘笔记分布式案例

正常项目数据抓取（非分布式）

MySQL数据库--建库建表

```
create database tencentdb charset utf8;
use tencentdb;
create table tencenttab(
name varchar(100),
type varchar(100),
duty varchar(5000),
requirement varchar(5000)
)charset=utf8;
```

- items.py

```
1
```

- tencent.py

```
1
```

- pipelines.py

```
1
```

- settings.py

```
1
```

改写为分布式（同时存入redis）

1. **settings.py**

```
# 使用scrapy_redis的调度器
SCHEDULER = "scrapy_redis.scheduler.Scheduler"
# 使用scrapy_redis的去重机制
DUPEFILTER_CLASS = "scrapy_redis.dupefilter.RFPDupeFilter"
# 爬取完成后是否清除请求指纹,True:不清除 False:清除
SCHEDULER_PERSIST = False
# 在ITEM_PIPELINES中添加redis管道
'scrapy_redis.pipelines.RedisPipeline': 200
# 定义redis主机地址和端口号
REDIS_HOST = '111.111.111.111'
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

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

```
def get_result(filename):
    # 用户名
    username    = 'yibeizi001'

    # 密码
    password    = 'zhanshen002'

```

```
    # 软件ＩＤ，开发者分成必要参数。登录开发者后台【我的软件】获得!
    appid       = 1

    # 软件密钥，开发者分成必要参数。登录开发者后台【我的软件】获得!
    appkey      = '22cc5376925e9387a23cf797cb9ba745'

    # 图片文件
    # filename    = 'getimage.jpg'

    # 验证码类型，# 例：1004表示4位字母数字，不同类型收费不同。请准确填写，否则影响识别率。在此查询所有类型 http://www.yundama.com/price.html
    codetype    = 5000

    # 超时时间，秒
    timeout     = 60

    # 初始化
    yundama = YDMHttp(username, password, appid, appkey)

    # 登陆云打码
    uid = yundama.login();

    # 查询余额
    balance = yundama.balance();

    # 开始识别，图片路径，验证码类型ID，超时时间（秒），识别结果
    cid, result = yundama.decode(filename, codetype, timeout);

    return result

##################################################################################
```

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-------