Day07回顾

创建项目流程

```
1、scrapy startproject Tencent
2、cd Tencent
3、scrapy genspider tencent tencent.com
4、items.py(定义爬取数据结构)
5、tencent.py (写爬虫文件)
6、pipelines.py(数据处理)
7、settings.py(全局配置)
8、终端: scrapy crawl tencent
```

响应对象属性及方法

```
# 属性
1、response.text ： 获取响应内容
2、response.body ： 获取bytes数据类型
3、response.xpath('')

# response.xpath('')调用方法
1、结果 ： 列表,元素为选择器对象
2、.extract() ： 提取文本内容,将列表中所有元素序列化为Unicode字符串
3、.extract_first() ： 提取列表中第1个文本内容
4、.get() ： 提取列表中第1个文本内容
```

爬虫项目启动方式

- 方式一

```
从爬虫文件(spider)的start_urls变量中遍历URL地址，把下载器返回的响应对象（response）交给爬虫文件的parse()函数处理
# start_urls = ['http://www.baidu.com/','http://www.sina.com.cn']
```

- 方式二

```
重写start_requests()方法，从此方法中获取URL，交给指定的callback解析函数处理

1、# 去掉start_urls变量
2、def start_requests(self):
      # 生成要爬取的URL地址，利用scrapy.Request()方法交给调度器 **
```

数据传递

```
1、数据交给管道 ： yield item
2、URL交给调度器 ： yield scrapy.Request(url,callback=解析函数名)
3、爬虫文件传递item到下一个解析函数
    yield scrapy.Request(
        url,
        meta={'item':item},
        callback=解析函数名
    }
```

settings.py常用变量

```
1、LOG_LEVEL = ''
2、LOG_FILE = ''
3、FEED_EXPORT_ENCODING = ''
4、IMAGES_STORE = '路径'
```

日志级别

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

数据持久化存储

```
1、settings.py定义常用变量
2、pipelines.py自定义管道类
    from .settings import *

    class TencentMongoPipeline(object):
        def open_spider(self,spider):
            # 打开数据库连接

        def process_item(self,item,spider):
            # item数据处理，必须return item

        def close_spider(self,spider):
```

```
            # 收尾工作，一般用于断开数据库连接

3、settings.py添加管道
    ITEM_PIPELINES = {'Tencent.pipelines.TencentxxxPipeline':1}
```

保存为csv、json文件

```
1、scrapy crawl tencnet -o xxx.json | xxx.csv
2、设置导出编码
   settings.py : FEED_EXPORT_ENCODING='utf-8'
```

Day08笔记

图片管道(360图片抓取)

- 目标

```
www.so.com -> 图片 -> 美女
# http://image.so.com/z?ch=beauty
```

- F12抓包

```
# 数据1: json地址
  url = 'http://image.so.com/zj?ch=beauty&sn={}&listtype=new&temp=1'

# 数据2: 查询参数 (QueryString)
   ch: beauty
   sn: 90
   listtype: new
   temp: 1
```

- 项目实现

1. items.py定义抓取的数据结构

```
# 图片链接
img_link = scrapy.Field()
```

2. so.py解析响应数据

```
1
```

3. pipelines.py数据处理

```
1
```

4. settings.py全局配置

```
1  # 定义存储图片的路径
2
```

5. begin.py运行爬虫

```
1  from scrapy import cmdline
2
3  cmdline.execute('scrapy crawl so'.split())
```

scrapy shell

- 基本使用

```
1  1、scrapy shell URL地址
2  *2、request.headers ： 请求头(字典)
3  *3、reqeust.meta    ： item数据传递，定义代理(字典)
4  4、response.text    ： 字符串
5  5、response.body    ： bytes
6  6、response.xpath('')
```

- scrapy.Request()

```
1  1、url
2  2、callback
3  3、headers
4  4、meta ： 传递数据,定义代理
5  5、dont_filter ： 是否忽略域组限制
6     默认False,检查allowed_domains['']
```

设置中间件(随机User-Agent)

少量User-Agent切换

- 方法一

```
# settings.py
USER_AGENT = ''
DEFAULT_REQUEST_HEADERS = {}
```

- 方法二

```
# spider
yield scrapy.Request(url,callback=函数名,headers={})
```

大量User-Agent切换（中间件）

- middlewares.py设置中间件

```
1、获取User-Agent
   # 方法1 ： 新建useragents.py,存放大量User-Agent, random模块随机切换
   # 方法2 ： 安装fake_useragent模块(sudo pip3 install fack_useragent)
       from fake_useragent import UserAgent
       ua_obj = UserAgent()
       ua = ua_obj.random
2、middlewares.py新建中间件类
    class RandomUseragentMiddleware(object):
        def process_request(self,reuqest,spider):
            ua = UserAgent()
            request.headers['User-Agent'] = ua.random
3、settings.py添加此下载器中间件
    DOWNLOADER_MIDDLEWARES = {'' : 优先级}
```

设置中间件(随机代理)

```
request.meta['proxy'] = 'http://127.0.0.1:8888'
** 使用代理尝试 **
```

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_ydm(filename):
    # 用户名
    username    = 'yibeizi001'
    # 密码
    password    = 'zhanshen001'
    # 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'
```

```
    # 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();
    # print('uid: %s' % uid)
    # 查询余额
    balance = yundama.balance();
    # print('balance: %s' % 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

```

```

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

```
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
```

- Ubuntu安装redis

```
# 安装
sudo apt-get install redis-server
# 启动
redis-server
# 连接
redis-cli -h IP地址
```

腾讯招聘笔记分布式案例

正常项目数据抓取（非分布式）

```
首先将项目以非分布式方式完成
```

改写为分布式（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"
# 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 = '172.40.91.129'
REDIS_PORT = 6379
```

改写为分布式（mongodb）

- 修改管道

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

- 清除redis数据库

```
1 flushall
```

- 代码拷贝一份到Ubuntu（分布式中其他机器）,两台机器同时执行此代码

移动端数据抓取

见 笔记中文件夹资料