## 爬虫分类

### 通用网络爬虫

* 搜索引擎引用，需要遵守robots协议
* 在网站后加入：robots.txt可以显示
  + 搜索引擎获取新网站的URL步骤
    - 网站主动向搜索引擎提供（百度站长平台）
    - 和DNS服务商（万网），快速收录新网站

### 聚焦网络爬虫

* 面向主题的爬虫、面向需求的爬虫

## 爬虫数据步骤及注意事项

* 确定要爬取的URL地址
* 通过HTTP/HTTPS协议来获取相应的HTML页面
* 提取HTML页面有用的数据
  1. 所需数据，保存
  2. 页面中有其他的URL，继续 第二步
* 在浏览器中匹配的不一定是真实的,此时
  1. 更换User-Agent为IE
  2. 把页面下载到本地,去查找位置
* 爬取的form表单等数据，一定要确保前后无空格

## Web知识需求

* <https://baidu.com/index.html> #detail【detail锚点】
* User-Agent：记录用户的浏览器、操作系统等，为了让用户获取更好的HTML页面效果
* Accept-Encoding:gzip----------浏览器压缩请求，程序里无法解压，所以不写
* Referrer----服务器检查是否爬虫的一种方式，显示你从什么页面而来
* Salt、sign一般是进行了js加密
* 浏览器分别
  + IE：Trident
  + Mozilla Firefox：（Gecko内核）
  + Linux:KTHML(like Gecko)
  + Apple:Webkit(like KHTML)
  + GooGle:Chrome(like Webkit)

## Fiddler

* Fiddler配置：Tools - > Options
  + HTTS:Decrypt Https Traffic(选中添加信任证书)
  + Connections:监听端口 8888
* 浏览器代理：SwitchOmega
  + 端口ip绑定为何Fiddler一样
  + 点击应用选项
* 设置手机抓包
  + 在手机上安装证书
    - 打开手机浏览器：http://电脑IP地址:8888
    - 在页面上下载(FiddlerRoot certificate)
      * 下载文件名：FiddlerRoot.cer
    - 安装
  + 设置手机代理
    - 打开手机已连接的无线，代理设置- >改成手动
    - 输入电脑的IP地址：8888

## request模块

* req = urlib.request.Request(url,data,headers)
  + 多看几次页面数据找出变化数据（可查看js加密）
  + Data（是字典，必须为bytes数据类型）----------- >form表单数据
* Response = urllib.request.urlopen(req)
  + 获取响应对象
* html = res.read().decode(‘utf-8’)
  + 并不是所有响应都是utf-8的格式，报错就换
* Dict = json.loads(html)
  + 将html转换为json格式

## csv模块

* import csv
* with open(‘test.csv’,’w’,newline=’’) as f:
  + newline=’’表格中不会有空行出现
* write = csv.writer(f)
* writer.writerow([])

## warnings模块

* warnings.filterwarnings('ignore’)
  + 忽略下方程序的警告

## 远程存入MySQL

* 开启MySQL远程连接
  + sudo –i
  + cd /etc/mysql/mysql.conf.d
  + subl mysqld.cnf
  + [mysqld]
  + #bind-adress = 127.0.0.1
  + /etc/init.d/mysql restart
* 添加授权用户
  + mysql>grant all privileges on \*.\* to ‘monkey’@’%’ indentified by ‘123’ with grant option;
  + mysql>flush privileges;
* 关闭Ubuntu防火墙
  + sudo ufw disable

## Ubuntu防火墙基本操作

* 查看状态：sudo ufw status
* 添加规则：sudo ufw allow 3306（允许外部访问3306端口）

## requests模块

* 安装：以管理员身份去打开Anaconda Prompt
  + conda install requests

### 常用方法

* 获取响应对象：get(url,headers=headers)
  + 响应对象res的属性
    - encoding：响应编码
    - text：字符串
    - content：bytes
    - status\_code：HTTP响应码
    - url：返回实际数据的URL地址
* 获取响应对象的方法
  + get(url,params,headers,proxies,auth,verify)
  + post(url,data,….)
    - data：字典，表示Form表单数据，不用编码或转码

### 查询参数

* res = requests.get(baseurl,params={},…)
* 自动对params字典进行编码，自动拼接URL地址

### 使用代理

* 获取IP地址的网站（快代理、全网代理）
* 高匿代理：看不到自身IP地址
* 普通代理
  + proxies={“协议”：”协议：//IP:端口号”}
* 私密代理
  + proxies={“http”:”http://用户名:密码@IP地址:端口号”}

### Web客户端验证(auth)

* auth=("用户名”,”密码”)

### SSL证书认证(verify)

* verify=True(默认) ----------进行CA证书认证 False：不进行认证

## 爬虫解析

### js加密

* 查看js文件，查找到和加密的键相同的名字，然后找到对应的加密算法
* Js：(new Date).getTime()-------获取当前的时间，数字长度13
  + Python：int(time.time()\*1000)刚好13位
* Js：navigator.appVersion
  + 获取浏览器的User-Agent属性（只是部分User-Agent的属性）
* Python：md5
  + form hashlib import md5
  + md = md5()
  + md.update(“a”.encode)
  + bv = s.hexdigest()--------------获得a的md5加密

### re

. 匹配任意字符（不能匹配\n）

\d 数字

\s 空白字符

[..] 包含[]内容： A[BCD]E -----> ABE ACE ADE

\w 字母、数字、\_

\* 0次或多次

？ 0或1次

+ 1次或多次

{m} m次： AB{1,3}C ----> ABC ABBC ABBBC

贪婪匹配（.\*）：在整个表达式匹配成功的前提下，尽可能匹配多的匹配\*（中间内容全无视，直到最后.\*后的条件为止）

非贪婪匹配(.\*?)：在整个表达式匹配成功的前提下，竟可能少的匹配\*

* 数据分类
  + 特点：有固定的格式，如：HTML、XML、JSON
* 非结构化
  + 示例：图片、视频、音频、这类数据一般都存储为二进制

**"""  
 先按照整体匹配出来,然后在匹配()中的  
 如果有２个或者多个(),则以元组的方式去显示  
"""***strings = "A B C D"  
p = re.compile("\w+\s+\w+")  
print(p.findall(strings))*

*p = re.compile("(\w+)\s+\w+")  
print(p.findall(strings))  
  
p = re.compile("(\w+)\s+(\w+)")  
print(p.findall(strings))*

### +xpath

* 在xml文档中查找信息的语言，同样适用于HTML文档的检索
* Xpath辅助工具
  + Chrome插件：XPath Helper
    - 打开或关闭：Ctrl + shift + X
  + Firefox插件：XPath Checker
  + XPath编辑工具：XML quire
* 匹配演示
  + 查找所有的book节点：//book
  + 查找所有book节点下的title子节点中，lang属性为”en”的节点：//book/title[@lang=”en”]
  + 查找bookstore下的第二个book节点下的title子节点：

/bookstore/book[2]/title

* 选取节点
  + //：从整个文档中查找节点
  + @：选取某个节点的属性值
* @的使用
  + 选取一类节点：//title[@lang=”en”]
  + 选取存在某个属性的节点：//title[@lang]
  + 选取节点的属性值：//title/@lang
* 匹配多路径
  + Xpath表达式1 | xpath表达式2
  + 获取所有book节点下的title子节点和price子节点
    - //book/title | //book/price子节点
* 函数
  + contains()：匹配一个属性中包含某些字符串的节点
    - //title[contains(@lang,”e”)]
    - //div[contains(@class,”njjzw”)]

<div class = “njjzw10389403”>

<div class = “njjzw10389432”>

* + text()：获取文本
    - title[contains(@lang,”e”)]/test()

### lxml库+xpath

* 安装
  + 管理员Prompt：conda install lxml
* 使用流程（只要调用了xpath，结果一定是列表）
  + from lxml import etree
  + 创建解析对象：parseHtml = etree.HTML(html)
  + 调用xpath：rList = parseHtml.xpath(‘xpath表达式’)
* 获取节点对象内容
  + Xpath表达式中：…/text()
  + 节点对象.txt for r in rList:print(r.text)

例： 

### BeautifulSoup(解析)

* HTML或者XML解析器，依赖于lxml
* 安装：conda install beautifulsoup4
* 使用流程
  + 导入模块：from bs4 import BeautifulSoup
  + 创建解析对象：soup = BeautifulSoup(html,’lxml’)
  + 查找节点对象：rList = soup.find\_all(‘div’,{‘id’:’’})
  + 查找该节点的子节点rList[0].find(‘div’,{‘class’:’’})
  + 获取当前节点（包含子节点）所有文本属性：rList[0].get\_text()
* 支持解析库
  + lxml：速度快，文档容错能力强（容错：比如<div>没有</div>的结尾）
  + html.paser：python标准库，速度一般，文档容错能力一般
  + xml：速度快，文档容错能力强

### Fiddler抓包工具设置

* 设置Fiddler软件
  + https：Tools – options – HTTPS - …from browsers only Actions 添加证书信任
  + connections：设置端口号 8888
  + 重启Fiddler软件
* 设置Chrome浏览器
  + 安装代理切换插件：Proxy SwitchOmega
  + 选项-新建情景模式-HTTP 127.0.0.1 8888 – 应用情景模式
  + 把代理切换到自己新建的情景模式上
* Fiddler常用菜单
  + Inspector：查看数据包详细内容
    - 分为请求（request）和响应（response）两部分
  + 常用子选项卡
    - Headers：显示请求头信息
    - WebForms：显示POST数据，在body中
    - Raw：将整个请求显示为纯文本

### json模块

* Json – > Python ：json.loads(html)
* Python – > Json ：Json.dumps(python)

## selenium+phantomjs/chrome

* 因为其效率低，多用于js加载和ajax动态加载

### selenium

* Web自动化测试工具，应用于Web自动化测试
* 特点
  + 可以运行在浏览器，根据指定命令操作浏览器，让浏览器自动加载页面
  + 只是工具，需要与第三方浏览器结合使用
* 安装：（管理员Anaconda Prompt）：conda install selenium

### Phantomjs

* 无界面浏览器（无头浏览器）
* 特点
  + 把网站加载到内存进行页面加载
  + 运行高效
* Windows安装：下载的exe文件，拷贝到python安装目录的Scripts目录下

### Chromedriver

* 查看Chrome浏览器版本
* 拷贝chromdriver.exe到Scripts目录下

### 浏览器对象driver的方法

* From selenium import webdriver
* driver = webdriver.PhantomJS(executable\_path)
  + executable \_path：webdriver的路径
  + 也可以driver=webdriver.Chrome()
* 发get请求获取响应：driver.get(url)
* 获取屏幕截图：driver.save\_screenshot(‘百度.png’)
* 获取html源码：driver.page\_source
* driver.page\_source.find(字符串)查找失败-1
* 单元素查找（节点对象，只找一个，search）
  + driver.find\_element\_by\_id(‘’)
  + driver.find\_element\_by\_name(‘’)
  + driver.find\_element\_by\_class\_name()
  + driver.find\_element\_by\_xpath(‘’)
* 多元素查找（返回值-列表）
  + driver.find\_elements\_by…()
  + 利用节点对象的text属性可获取文本内容（包含子节点文本内容）
    - Xpath的”/text()”只获取该节点下的纯文本
* 节点对象.send\_keys(‘’)
* 节点对象.click()
* 关闭浏览器：driver.quit()
* 执行js脚本：driver.execute\_script(‘window.srollTo(0,document.body.scrollHeight)’)下拉到底端

### 设置无界面

* opt = webdriver.ChromeOptions()
* opt.set\_headless()
* opt.add\_arguments(‘window-size=…’)
* driver = webdriver.Chrome(options=opt)

## 多线程爬虫

* 多进程：大量的密集计算
* 多线程：依赖网络I/O程序（网络IO、磁盘IO、爬虫）

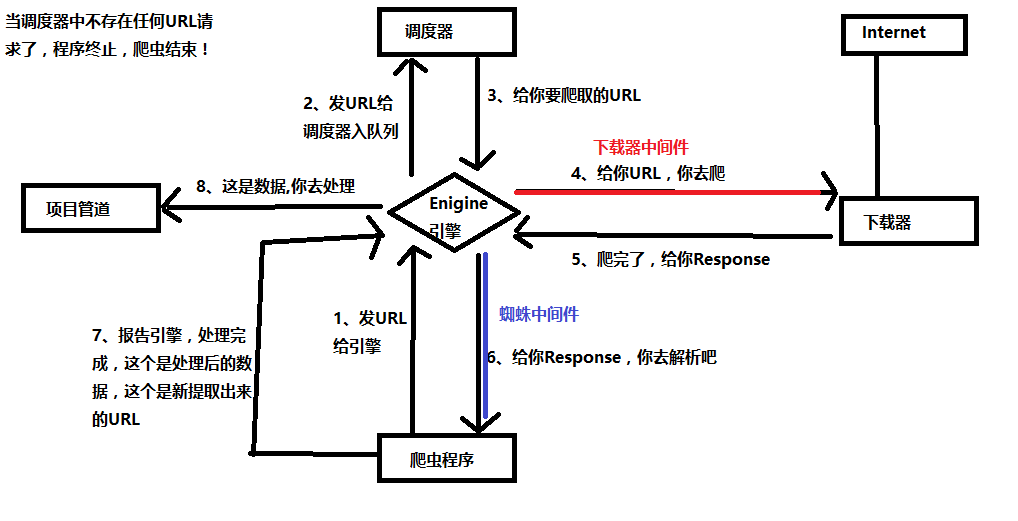
### 队列(from multiprocessing import Queue)

* UrlQueue = Queue()
* UrlQueue.put(url)
* UrlQueue().get(block=True,timeout=2) #没有值阻塞，2秒后异常
  + bloclk=false抛出异常
* UrlQueue.empty()

## Scrapy

* 异步处理框架，可配置和可扩展程序非常高
* 安装：
  + Windows
    - Anaconda Prompt : conda install Scrapy
  + Ubuntu
    - 1、安装依赖包 ：
      * sudo apt-get install python3-dev
      * sudo apt-get install python-pip
      * sudo apt-get install libxml2-dev
      * sudo apt-get install libxslt1-dev
      * sudo apt-get install zlib1g-dev
      * sudo apt-get install libffi-dev
      * sudo apt-get install libssl-dev
    - sudo pip3 install pyasn1==0.4.1
    - 安装Scrapy：sudo pip3 install Scrapy

### Scrapy框架组成



* 引擎Engine：整个框架核心
* 调度器Scheduler：接收从引擎发过来的URL，入队列
* 下载器Downloader：获取response，返回给爬虫程序
* 项目管道Item Pipeline：数据处理
* 中间件
  + 下载器中间件Downloader Middlewares：处理引擎与下载器之间的请求比如添加headers
  + 蜘蛛中间件Spider Middlewares：处理爬虫程序输入响应和输出结果以及新的请求

### 制作Scrapy爬虫项目步骤

* + 新建项目cmd命令：scrapy startproject 项目名
  + 明确目标：items.py
  + 制作爬虫程序
    - cd spiders
    - scrapy genspider 文件名 域名
  + 处理数据：pipelines.py
  + 全局配置：settings.py
  + 运行爬虫项目：scrapy crawl 爬虫名

例：

Baidu/

|-- scrapy.cfg : 项目基本配置,不用改

|-- begin.py :from scrapy import cmdline; cmdline.execute(‘scrapy crawl baidu’.split())

|-- Baidu/ : 项目目录

|--items.py : 定义爬取数据结构

|--middlewares.py : 2个中间件

|--pipelines.py : 管道文件(数据处理)

|--settings.py : 全局配置

|--spiders/

|--baidu.py : 爬虫程序

#### settings.py配置

* 设置User-Agent：USER\_AGENT = ‘Mozilla/5.0’
* 是否遵循robots协议，设置为False：ROBOTSTXT\_OBEY = False
* 最大并发量，默认16：CONCURRENT\_REQUESTS = 32
* 下载延迟时间：DOWNLOAD\_DELAY = 1
* 请求头（也可以设置User-Agent）
  + DEFAULT\_REQUEST\_HEADERS = {
    - 'User-Agent': 'Mozilla/5.0',
    - 'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8',
    - 'Accept-Language': 'en',}
* 下载器中间件
  + DOWNLOADER\_MIDDLEWARES={
    - ‘Baidu.middlewares.randomProxyMiddleware’:543,
    - ‘Baidu.middlewares.randomUserAgentMiddleware’:300,}
* 项目管道：优先级1-1000，数字越小优先级（谁先执行）越高
  + ITEM\_PIPELINES={
    - ‘Baidu.pipelines.BaiduPipeline’:300,
    - ‘Baidu.pipelines.BaiduMongoPipeLine’:200,
    - ‘Baidu.pipelines.BaiduMysqlPipeline’:100,}

#### Scrapy项目运行方法

* 配置pycharm的python环境
  + 在Baidu根目录目录下创建begin.py
    - form scrapy import cmdline
    - cmdline.execut(‘scrapy crawl baidu’.split())
  + Setting - > Project Interpreter - > 右上角…add - > existinig environment -> 选择你自己Anaconda中python的路径(C:\ProgramData\Anaconda3\python.exe) -> 点击OK
* 在项目根目录下，输入scrapy crawl baidu

#### Spyders下的py文件

* extract()：获取选择器对象中的文本内容
  + response.xpath(‘’)得到的结果为选择器对象的列表
* popelines.py中必须有个函数叫（返回的item是给下一个管道使用）：
  + def process\_item(self,item,spider):return item

#### MongoDB（MYSQL类似）

* 在setting.py中定义相关变量
  + MONGODB\_HOST =
  + MONGODB\_PORT =
  + MONGODB\_DATABASE =
  + MONGODB\_TABLE =
* 在pipelines.py中添加项目管理
  + class DaomuMongoPipeline(object):

def \_\_init\_\_(self):

def process\_item(self,item,spider):

* 在setting.py中设置项目管道
  + ITEM\_PIPELINES = {'Daomu.pipelines.DaomuMongoPipeline':200,}

#### Scrapy设置LOG

* 在setting.py中定义相关变量
  + 设置LOG级别：LOG\_LEVEL = “DEBUG”
    - 5层日志级别
      * CRITICAL ：严重错误
      * ERROR ：一般错误
      * WARNING ：警告信息（只显示WARNING/ERROR/CRITICAL信息）
      * DEBUG ：调试信息
      * INFO ：一般信息
  + LOG信息存放到本地log文件：LOG\_FILE = “daomu.log”

#### 保存为csv/json文件

* Json文件
  + scrapy crawl daomu –o daomu.json
  + 在setting设置导出的编码格式
    - FEED\_EXPORT\_ENCODING=’utf-8’
* csv文件
  + scrapy crawl daomu –o daomu.csv
    - FEED\_EXPORT\_ENCODING=’gb18030
    - csv出现空行解决方法（修改源码exporters.py）
      * Anaconda3/Lib/site-packages/scrapy/
      * self.stream = io.TextIOWrapper(file,newline=’’)

#### 调度器

* 交给调度器,并指定解析函数
  + yield scrapy.Request(url,callback = self.getHtml)
    - url、callback、headers、meta代理等相关信息
    - dont\_filter：是否忽略域组设置
      * 默认为False，检查allowed\_domains=[‘’]
      * 比如allowed\_domain爬取的是百度，request的url是其他网站就无效
  + 解析函数：def getHtml(self):
* 重写start\_requests()方法
  + 拼接url地址，发给调度器

#### 图片管道

* XXSpider类
  + from ..items import SoItem
* XXPipeline(ImagesPipeline)类
  + from scrapy.pipelines.images import ImagesPipeline
    - 重写方法get\_media\_requests(self,item,info):
    - 将要下载的图片路径交给调度器yield scrapy.Request(item[‘imgLink’])
    - 下载图片的位置在setting设置
      * IMAGES\_STORE

#### Scrapy shell

* scrapy shell 网址
* response.text
* response.body-----------bytes格式
* response.xpath(“//”)
* request.headers
* 定义代理等相关属性：request.meta

#### 下载中间器

* 没设置中间器之前，settings.py只能适用于少量user-agent
* middlewares.py设置中间件
  + 存放User-Agent的类
  + 新建RamdomUAMiddleware(‘object’)
    - def process\_request(self,request,spyder)
* 到setting文件载入中间件
  + DOWNLOADER\_MIDDLEWARES
* RandomProxyMiddleware:
  + def process\_request(self,request,spyder).
  + proxy\_list = [‘http://127.0.0.1:8888’,]
  + request.meta[“proxy”] = random.choice(proxy\_list)

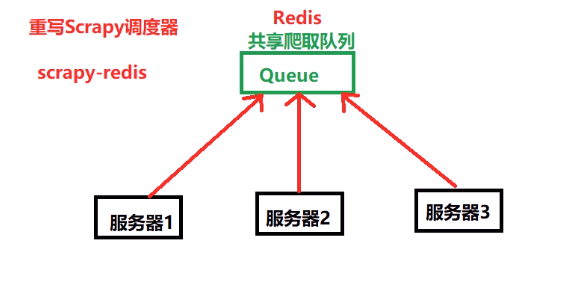
## 机器视觉与tesseract（验证码）

* OCR：光学字符识别，通过字符形状，转换为电子文本
* tesseract-ocr：OCR的一个底层识别库，不能导入
  + 下载安装、并添加到环境变量
    - <Http://sourceforge.net/projects/tesseract-ocr-alt/files/>
    - 添加环境变量：tesseract-ocr的安装路径添加进去
* pytesseract：python模块，做的封装
  + import pytesseract
  + from PIL import Image：图像处理标准库
  + img = Image.open(‘test1.jpg’)：创建图片对象
  + result=pytesseract.image\_to\_string(img)：图片转字符串
* tesseract-orc识别率很低，很多文字变形、干扰，导致无法识别
  + 在线打码
    - 云打码平台（如www.yundama.com）
      * 注册用户-充值-下载接口（开发文档）
      * 提分价格

## Redis缓存数据库

* 非关系型数据库，key-value形式存储，内存读写
* 服务端启动：cmd：redis-server.exe redis.conf
* 客户端连接：cmd:redis-cli.exe –h IP地址
* 存值：set ‘a’ “b”
* 刷新清除所有数据：flushall
* 安装图形界面管理工具（RedisDesktopManager）
  + Name:127.0.0.1

## 分布式爬取



* 多台主机共享一个爬取队列
* 实现：重写scrapy调度器（scrapy\_redis模块）
* 安装：
  + pip install scrapy\_redis
* 使用redis集合，存储每个request指纹
* GetHub搜索scrapy\_redis
  + Setting
    - 使用自己定义的调度器机制：
      * SCHEDULER = “scrapy\_redis.dupefilter.RFPDupeFilter”
    - 去重机制：DUPEFILTER\_CLASS
    - 不清楚指纹（指纹是用来判断是否重复url）
      * SCHEDULER\_PESIST = True
    - 数据库：
      * REDIS\_HOST =
      * REDIS\_PORT = 6379

### 手机抓包

* 1. 设置手机(见图)
* 1. 手动
* IP地址 ： 你电脑的IP(ipconfig)
* 端口号 ： 8888(和Fiddler保持一致)
* 2. 设置电脑(更改注册表)
* 3. 设置Fiddler
* 1. HTTPS选项卡 ：...from all processes
* 2. Connections选项卡
* 1. 端口号 ：8888(和手机保持一致)
* 2. Allow remote computers to Connect
* 3. 重启Fiddler

