# Day08回顾

## scrapy框架

* **五大组件+工作流程+常用命令**
* 【1】五大组件  
   1.1) 引擎（Engine）  
   1.2) 爬虫程序（Spider）  
   1.3) 调度器（Scheduler）  
   1.4) 下载器（Downloader）  
   1.5) 管道文件（Pipeline）  
   1.6) 下载器中间件（Downloader Middlewares）  
   1.7) 蜘蛛中间件（Spider Middlewares）  
     
  【2】工作流程  
   2.1) Engine向Spider索要URL,交给Scheduler入队列  
   2.2) Scheduler处理后出队列,通过Downloader Middlewares交给Downloader去下载  
   2.3) Downloader得到响应后,通过Spider Middlewares交给Spider  
   2.4) Spider数据提取：  
   a) 数据交给Pipeline处理  
   b) 需要跟进URL,继续交给Scheduler入队列，依次循环  
     
  【3】常用命令  
   3.1) scrapy startproject 项目名  
   3.2) scrapy genspider 爬虫名 域名  
   3.3) scrapy crawl 爬虫名

## 完成scrapy项目完整流程

* **完整流程**
* 【1】crapy startproject Tencent  
  【2】cd Tencent  
  【3】scrapy genspider tencent tencent.com  
  【4】items.py(定义爬取数据结构)  
   import scrapy  
   class TencentItem(scrapy.Item):  
   name = scrapy.Field()  
   address = scrapy.Field()  
     
  【5】tencent.py（写爬虫文件）  
   import scrapy  
   from ..items import TencentItem  
     
   class TencentSpider(scrapy.Spider):  
   name = 'tencent'  
   allowed\_domains = ['tencent.com']  
   start\_urls = ['http://tencent.com/']  
   def parse(self,response):  
   item = TencentItem()  
   xxx  
   yield item  
    
  【6】pipelines.py(数据处理)  
   class TencentPipeline(object):  
   def process\_item(self,item,spider):  
   return item  
     
  【7】settings.py(全局配置)  
   LOG\_LEVEL = '' # DEBUG < INFO < WARNING < ERROR < CRITICAL  
   LOG\_FILE = ''  
   FEED\_EXPORT\_ENCODING = ''  
     
  【8】run.py   
   from scrapy import cmdline  
   cmdline.execute('scrapy crawl tencnet'.split())

## 我们必须记住

* **熟练记住**
* 【1】响应对象response属性及方法  
   1.1) response.text ：获取响应内容 - 字符串  
   1.2) response.body ：获取bytes数据类型  
   1.3) response.xpath('')  
   1.4) response.xpath('').extract() ：提取文本内容,将列表中所有元素序列化为Unicode字符串  
   1.5) response.xpath('').extract\_first() ：序列化提取列表中第1个文本内容  
   1.6) response.xpath('').get() ： 提取列表中第1个文本内容(等同于extract\_first())  
     
  【2】settings.py中常用变量  
   2.1) 设置日志级别  
   LOG\_LEVEL = ''  
   2.2) 保存到日志文件(不在终端输出)  
   LOG\_FILE = 'xxx.log'  
   2.3) 设置数据导出编码(主要针对于json文件)  
   FEED\_EXPORT\_ENCODING = 'utf-8'  
   2.4) 设置User-Agent  
   USER\_AGENT = 'Mozilla/5.0'  
   2.5) 设置最大并发数(默认为16)  
   CONCURRENT\_REQUESTS = 32  
   2.6) 下载延迟时间(每隔多长时间请求一个网页)  
   DOWNLOAD\_DELAY = 0.1  
   2.7) 请求头  
   DEFAULT\_REQUEST\_HEADERS = {'Cookie':'','User-Agent':''}  
   2.8) 添加项目管道  
   ITEM\_PIPELINES = {'项目目录名.pipelines.类名' : 200}  
   2.9) cookie(默认禁用,取消注释-True|False都为开启)  
   COOKIES\_ENABLED = False  
   2.10) 非结构化数据存储路径  
   IMAGES\_STORE = '/home/tarena/images/'  
   FILES\_STORE = '/home/tarena/files/'  
   2.11) 添加下载器中间件  
   DOWNLOADER\_MIDDLEWARES = {'项目目录名.middlewares.类名' : 200}  
     
  【3】日志级别  
   DEBUG < INFO < WARNING < ERROR < CRITICAL

## 爬虫项目启动方式

* **启动方式**
* 【1】方式一:基于start\_urls  
   1.1) 从爬虫文件(spider)的start\_urls变量中遍历URL地址交给调度器入队列,  
   1.2) 把下载器返回的响应对象（response）交给爬虫文件的parse(self,response)函数处理  
    
  【2】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  
   2.1) 去掉start\_urls变量  
   2.2) def start\_requests(self):  
   # 生成要爬取的URL地址，利用scrapy.Request()方法交给调度器

## 数据持久化存储

* **MySQL-MongoDB-Json-csv**
* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*存入MySQL、MongoDB\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
    
  【1】在setting.py中定义相关变量  
  【2】pipelines.py中新建管道类，并导入settings模块  
   def open\_spider(self,spider):  
   # 爬虫开始执行1次,用于数据库连接  
     
   def process\_item(self,item,spider):  
   # 用于处理抓取的item数据  
   return item  
     
   def close\_spider(self,spider):  
   # 爬虫结束时执行1次,用于断开数据库连接  
     
  【3】settings.py中添加此管道  
   ITEM\_PIPELINES = {'':200}  
    
  【注意】 process\_item() 函数中一定要 return item  
    
  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*存入JSON、CSV文件\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  scrapy crawl maoyan -o maoyan.csv  
  scrapy crawl maoyan -o maoyan.json  
  【注意】  
   存入json文件时候需要添加变量(settings.py) : FEED\_EXPORT\_ENCODING = 'utf-8'

## 多级页面抓取之爬虫文件

* **多级页面攻略**
* 【场景1】只抓取一级页面的情况  
  """  
  一级页面: 名称(name)、爱好(likes)  
  """  
  import scrapy  
  from ..items import OneItem  
  class OneSpider(scrapy.Spider):  
   name = 'One'  
   allowed\_domains = ['www.one.com']  
   start\_urls = ['http://www.one.com']  
   def parse(self,response):  
   dd\_list = response.xpath('//dd')  
   for dd in dd\_list:  
   # 创建item对象  
   item = OneItem()  
   item['name'] = dd.xpath('./text()').get()  
   item['likes'] = dd.xpath('./text()').get()  
     
   yield item  
    
    
  【场景2】二级页面数据抓取  
  """  
  一级页面: 名称(name)、详情页链接(url)-需要继续跟进  
  二级页面: 详情页内容(content)  
  """  
  import scrapy  
  from ..items import TwoItem  
    
  class TwoSpider(scrapy.Spider):  
   name = 'two'  
   allowed\_domains = ['www.two.com']  
   start\_urls = ['http://www.two.com/']  
   def parse(self,response):  
   """一级页面解析函数,提取 name 和 url(详情页链接,需要继续请求)"""  
   dd\_list = response.xpath('//dd')  
   for dd in dd\_list:  
   # 有继续交给调度器入队列的请求,就要创建item对象  
   item = TwoItem()  
   item['name'] = dd.xpath('./text()').get()  
   item['url'] = dd.xpath('./@href').get()  
     
   yield scrapy.Request(  
   url=item['url'],meta={'item':item},callback=self.detail\_page)  
     
   def detail\_page(self,response):  
   item = response.meta['item']  
   item['content'] = response.xpath('//text()').get()  
     
   yield item  
     
     
  【场景3】三级页面抓取  
  """  
  一级页面: 名称(one\_name)、详情页链接(one\_url)-需要继续跟进  
  二级页面: 名称(two\_name)、下载页链接(two\_url)-需要继续跟进  
  三级页面: 具体所需内容(content)  
  """  
  import scrapy  
  from ..items import ThreeItem  
    
  class ThreeSpider(scrapy.Spider):  
   name = 'three'  
   allowed\_domains = ['www.three.com']  
   start\_urls = ['http://www.three.com/']  
     
   def parse(self,response):  
   """一级页面解析函数 - one\_name、one\_url"""  
   dd\_list = response.xpath('//dd')  
   for dd in dd\_list:  
   # 有继续发往调度器的请求,创建item对象的时刻到啦！！！  
   item = ThreeItem()  
   item['one\_name'] = dd.xpath('./text()').get()  
   item['one\_url'] = dd.xpath('./@href').get()  
   yield scrapy.Request(  
   url=item['one\_url'],meta={'meta\_1':item},callback=self.parse\_two)  
     
   def parse\_two(self,response):  
   """二级页面解析函数: two\_name、two\_url"""  
   meta1\_item = response.meta['meta\_1']  
   li\_list = response.xpath('//li')  
   for li in li\_list:  
   # 有继续交给调度器入队列的请求啦，所以创建item对象的时刻来临了！！！  
   item = ThreeItem()  
   item['two\_name'] = li.xpath('./text()').get()  
   item['two\_url'] = li.xpath('./@href').get()  
   item['one\_name'] = meta1\_item['one\_name']  
   item['one\_url'] = meta1\_item['one\_url']  
   # 交给调度器入队列  
   yield scrapy.Request(  
   url=item['two\_url'],meta={'meta\_2':item},callback=self.detail\_page)  
     
   def detail\_page(self,response):  
   """三级页面解析: 具体内容content"""  
   item = response.meta['meta\_2']  
   # 太好了！提具体内容了,没有继续交给调度器的请求了！所以,我不用再去创建item对象啦  
   item['content'] = response.xpath('//text()').get()  
     
   # 交给管道文件处理  
   yield item

# Day09笔记

## 文件管道使用方法

【1】爬虫文件: 将文件链接yield到管道  
【2】管道文件:  
 from scrapy.pipelines.files import FilesPipeline  
 class XxxPipeline(FilesPipeline):  
 def get\_media\_requests(self,xxx):  
 pass  
   
 def file\_path(self,xxx):  
 return filename  
   
【3】settings.py中:  
 FILES\_STORE = '路径'

## 图片管道使用方法

【1】爬虫文件: 将图片链接yield到管道  
【2】管道文件:  
 from scrapy.pipelines.images import ImagesPipeline  
 class XxxPipeline(ImagesPipeline):  
 def get\_media\_requests(self,xxx):  
 pass  
   
 def file\_path(self,xxx):  
 pass  
   
【3】settings.py中:  
 IMAGES\_STORE = '路径'

## 第一PPT模板下载 - 文件管道

* **项目概述**
* 【1】URL地址  
   1.1) http://www.1ppt.com/xiazai/  
   抓取所有栏目分类的所有页的PPT文件  
    
  【2】文件保存路径  
   /home/tarena/ppt/xxx/xxx.rar  
     
  【3】思路  
   3.1) 主页提取数据: 所有分类名称、所有分类链接  
   基准xpath: //div[@class="col\_nav clearfix"]/ul/li  
   分类名称: ./a/text()  
   分类链接: ./a/@href  
   3.2) 获取每个分类下的PPT总页数  
   获取'末页'节点,想办法提取 : //ul[@class="pages"]/li[last()]/a/@href  
   total = int(last\_page\_a.split('.')[0].split('\_')[-1])  
   3.3) 获取一页中所有PPT的名称、链接  
   基准xpath: //ul[@class="tplist"]/li  
   PPT名称: ./h2/a/text()  
   PPT链接: ./a/@href  
   3.4) 获取具体ppt下载链接  
   下载链接: //ul[@class="downurllist"]/li/a/@href

### 项目实现

* + **1 - 创建项目和爬虫文件**
  + scrapy startproject Ppt  
    cd Ppt  
    scrapy genspider ppt www.1ppt.com
  + **2 - 定义抓取的数据结构**
  + import scrapy  
      
    class PptItem(scrapy.Item):  
     # pipelines.py中所需数据：大分类名称、具体PPT文件名、PPT下载链接  
     parent\_name = scrapy.Field()  
     ppt\_name = scrapy.Field()  
     download\_url = scrapy.Field()
  + **3 - 爬虫文件提取数据**
  + # -\*- coding: utf-8 -\*-  
    import scrapy  
    from ..items import PptItem  
      
    class PptSpider(scrapy.Spider):  
     name = 'ppt'  
     allowed\_domains = ['www.1ppt.com']  
     start\_urls = ['http://www.1ppt.com/xiazai/']  
      
     def parse(self, response):  
     """一级页面解析函数: 提取大分类名称和链接"""  
     li\_list = response.xpath('//div[@class="col\_nav clearfix"]/ul/li')  
     for li in li\_list[1:]:  
     item = PptItem()  
     # 大分类名称、链接  
     item['parent\_name'] = li.xpath('./a/text()').get()  
     parent\_url = 'http://www.1ppt.com' + li.xpath('./a/@href').get()  
      
     # 依次将大分类链接交给调度器入队列  
     yield scrapy.Request(url=parent\_url, meta={'meta1':item}, callback=self.get\_total\_page)  
      
     def get\_total\_page(self, response):  
     """二级页面解析函数:获取总页数,并交给调度器入队列"""  
     meta1 = response.meta['meta1']  
     try:  
     # last\_page\_a : ppt\_jihua\_12.html  
     last\_page\_a = response.xpath('//ul[@class="pages"]/li[last()]/a/@href').get()  
     total = int(last\_page\_a.split('.')[0].split('\_')[-1])  
     url\_name = last\_page\_a.split('.')[0].split('\_')[-2]  
      
     page\_url = 'http://www.1ppt.com/xiazai/{}/ppt\_{}\_{}.html'  
     for page in range(1, total+1):  
     # 拼接此类别下的所有页的URL地址  
     url = page\_url.format(url\_name, url\_name, page)  
     yield scrapy.Request(url=url, meta={'meta2': meta1}, callback=self.get\_ppt\_info)  
     except Exception as e:  
     # 如果捕捉到异常,说明只有1页  
     yield scrapy.Request(url=response.url, meta={'meta2': meta1}, callback=self.get\_ppt\_info)  
      
     def get\_ppt\_info(self, response):  
     """提取PPT详情页链接,以及PPT名字"""  
     meta2 = response.meta['meta2']  
     li\_list = response.xpath('//ul[@class="tplist"]/li')  
     for li in li\_list:  
     item = PptItem()  
     ppt\_info\_url = 'http://www.1ppt.com' + li.xpath('./a/@href').get()  
     item['ppt\_name'] = li.xpath('./h2/a/text()').get()  
     item['parent\_name'] = meta2['parent\_name']  
      
     yield scrapy.Request(url=ppt\_info\_url, meta={'meta3': item}, callback=self.download\_ppt)  
      
     def download\_ppt(self, response):  
     """获取PPT下载链接"""  
     item = response.meta['meta3']  
     item['download\_url'] = response.xpath('//ul[@class="downurllist"]/li/a/@href').get()  
      
     yield item
  + **4 - 管道文件**
  + import scrapy  
    from scrapy.pipelines.files import FilesPipeline  
      
    class PptPipeline(FilesPipeline):  
     def get\_media\_requests(self, item, info):  
     yield scrapy.Request(url=item['download\_url'], meta={'item':item})  
      
     def file\_path(self, request, response=None, info=None):  
     item = request.meta['item']  
     filename = '{}/{}.rar'.format(item['parent\_name'], item['ppt\_name'])  
      
     return filename
  + **5 - 全局配置**
  + ROBOTSTXT\_OBEY = False  
    DOWNLOAD\_DELAY = 1  
    DEFAULT\_REQUEST\_HEADERS = {  
     'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8',  
     'Accept-Language': 'en',  
     'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.122 Safari/537.36',  
    }  
    ITEM\_PIPELINES = {  
     'Ppt.pipelines.PptPipeline': 300,  
    }  
    FILES\_STORE = '/home/tarena/ppt/'

## scrapy - post请求

* **方法+参数**
* scrapy.FormRequest(  
   url=posturl,  
   formdata=formdata,  
   callback=self.parse  
  )

### 抓取全国所有城市肯德基门店信息

* **目标说明**
* 【1】主页URL地址  
   http://www.kfc.com.cn/kfccda/storelist/index.aspx  
    
  【2】抓取所有城市的肯德基门店信息  
   2.1) 门店编号  
   2.2) 门店名称  
   2.3) 门店地址  
   2.4) 所属城市  
     
  【3】将所抓数据存储到MySQL数据库中  
  <a href=".\*?rel="(.\*?)">
* **步骤1 - 创建项目+爬虫文件**
* scrapy startproject Kfc  
  cd Kfc  
  scrapy genspider kfc www.kfc.com.cn
* **步骤2 - 定义要抓取的数据结构(items.py)**
* import scrapy  
    
  class KfcItem(scrapy.Item):  
   # 门店编号 + 门店名称 + 门店地址 + 所属城市  
   row\_num = scrapy.Field()  
   store\_name = scrapy.Field()  
   address\_detail = scrapy.Field()  
   city\_name = scrapy.Field()
* **步骤3 - 写爬虫程序(kfc.py)**
* # -\*- coding: utf-8 -\*-  
  import scrapy  
  import requests  
  import json  
  import re  
  from ..items import KfcItem  
    
    
  class KfcSpider(scrapy.Spider):  
   name = 'kfc'  
   allowed\_domains = ['www.kfc.com.cn']  
   index\_url = 'http://www.kfc.com.cn/kfccda/storelist/index.aspx'  
   post\_url = 'http://www.kfc.com.cn/kfccda/ashx/GetStoreList.ashx?op=cname'  
   headers = {'User-Agent':'Mozilla/5.0'}  
    
   # 经过分析为POST请求,故使用start\_requests()方法  
   def start\_requests(self):  
   """拼接多页地址,进行数据抓取"""  
   # 获取所有的城市  
   all\_city = self.get\_all\_city()  
   for city in all\_city:  
   # 获取每个城市的门店页数  
   total = self.get\_total\_page(city)  
   for i in range(1,total+1):  
   # 此为抓包抓到的Form表单数据  
   formdata = {  
   "cname": city,  
   "pid": "",  
   "pageIndex": str(i),  
   "pageSize": "10"  
   }  
   yield scrapy.FormRequest(url=self.post\_url,formdata=formdata,callback=self.parse)  
    
   def get\_all\_city(self):  
   """获取所有的城市列表"""  
   html = requests.get(url=self.index\_url,headers=self.headers).text  
   pattern = re.compile('<a href=".\*?rel="(.\*?)">',re.S)  
   all\_city = pattern.findall(html)  
    
   return all\_city  
    
   def get\_total\_page(self,city):  
   """获取某个城市的肯德基总数 - 向第1页发请求即可获取"""  
   data = {  
   "cname": city,  
   "pid": "",  
   "pageIndex": "1",  
   "pageSize": "10"  
   }  
   html = requests.post(url=self.post\_url,data=data,headers=self.headers).json()  
   kfc\_count = html['Table'][0]['rowcount']  
   total = kfc\_count//10 if kfc\_count%10==0 else kfc\_count//10 + 1  
    
   return total  
    
   def parse(self, response):  
   html = json.loads(response.text)  
   kfc\_shop\_list = html['Table1']  
   for kfc\_shop in kfc\_shop\_list:  
   item = KfcItem()  
   item['row\_num'] = kfc\_shop['rownum']  
   item['store\_name'] = kfc\_shop['storeName']  
   item['address\_detail'] = kfc\_shop['addressDetail']  
   item['city\_name'] = kfc\_shop['cityName']  
    
   yield item
* **步骤4 - 管道文件实现(pipelines.py)**
* # 存入MySQL管道  
  """  
  create database kfcdb charset utf8;  
  use kfcdb;  
  create table kfctab(  
  row\_num int,  
  store\_name varchar(100),  
  address\_detail varchar(200),  
  city\_name varchar(100)  
  )charset=utf8;  
  """  
  import pymysql  
  from .settings import \*  
    
  class KfcMysqlPipeline(object):  
   def open\_spider(self,spider):  
   self.db = pymysql.connect(MYSQL\_HOST,MYSQL\_USER,MYSQL\_PWD,MYSQL\_DB,charset=CHARSET)  
   self.cursor = self.db.cursor()  
   self.ins = 'insert into kfctab values(%s,%s,%s,%s)'  
    
   def process\_item(self, item, spider):  
   shop\_li = [  
   item['row\_num'],  
   item['store\_name'],  
   item['address\_detail'],  
   item['city\_name']  
   ]  
   self.cursor.execute(self.ins,shop\_li)  
   self.db.commit()  
    
   return item  
    
   def close\_spider(self,spider):  
   self.cursor.close()  
   self.db.close()
* **步骤5 - 全局配置(settings.py)**
* 【1】ROBOTSTXT\_OBEY = False  
  【2】DOWNLOAD\_DELAY = 0.1  
  【3】DEFAULT\_REQUEST\_HEADERS = {  
   'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8',  
   'Accept-Language': 'en',  
   'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.149 Safari/537.36',  
  }  
  【4】ITEM\_PIPELINES = {  
   'Kfc.pipelines.KfcMysqlPipeline': 300,  
  }  
  【5】MYSQL\_HOST = 'localhost'  
  【6】MYSQL\_USER = 'root'  
  【7】MYSQL\_PWD = '123456'  
  【8】MYSQL\_DB = 'kfcdb'  
  【9】CHARSET = 'utf8'
* **步骤6 - 运行爬虫(run.py)**
* from scrapy import cmdline  
    
  cmdline.execute('scrapy crawl kfc'.split())
* **练习**
* 请使用scrapy框架实现有道翻译案例

### 有道翻译案例实现

* **步骤1 - 创建项目+爬虫文件**
* scrapy startproject Youdao  
  cd Youdao  
  scrapy genspider youdao fanyi.youdao.com
* **步骤2 - items.py**
* result = scrapy.Field()
* **步骤3 - youdao.py**
* # -\*- coding: utf-8 -\*-  
  import scrapy  
  import time  
  import random  
  from hashlib import md5  
  import json  
  from ..items import YoudaoItem  
    
  class YoudaoSpider(scrapy.Spider):  
   name = 'youdao'  
   allowed\_domains = ['fanyi.youdao.com']  
   word = input('请输入要翻译的单词:')  
    
   def start\_requests(self):  
   post\_url = 'http://fanyi.youdao.com/translate\_o?smartresult=dict&smartresult=rule'  
   salt, sign, ts = self.get\_salt\_sign\_ts(self.word)  
   formdata = {  
   'i': self.word,  
   'from': 'AUTO',  
   'to': 'AUTO',  
   'smartresult': 'dict',  
   'client': 'fanyideskweb',  
   'salt': salt,  
   'sign': sign,  
   'ts': ts,  
   'bv': 'cf156b581152bd0b259b90070b1120e6',  
   'doctype': 'json',  
   'version': '2.1',  
   'keyfrom': 'fanyi.web',  
   'action': 'FY\_BY\_REALTlME'  
   }  
   # 发送post请求的方法  
   yield scrapy.FormRequest(url=post\_url,formdata=formdata)  
    
   def get\_salt\_sign\_ts(self, word):  
   # salt  
   salt = str(int(time.time() \* 1000)) + str(random.randint(0, 9))  
   # sign  
   string = "fanyideskweb" + word + salt + "n%A-rKaT5fb[Gy?;N5@Tj"  
   s = md5()  
   s.update(string.encode())  
   sign = s.hexdigest()  
   # ts  
   ts = str(int(time.time() \* 1000))  
   return salt, sign, ts  
    
   def parse(self, response):  
   item = YoudaoItem()  
   html = json.loads(response.text)  
   item['result'] = html['translateResult'][0][0]['tgt']  
    
   yield item
* **步骤4 - pipelines.py**
* class YoudaoPipeline(object):  
   def process\_item(self, item, spider):  
   print('翻译结果:',item['result'])  
   return item
* **步骤5 - settings.py**
* ROBOTSTXT\_OBEY = False  
  LOG\_LEVEL = 'WARNING'  
  COOKIES\_ENABLED = False  
  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",  
  }  
  ITEM\_PIPELINES = {  
   'Youdao.pipelines.YoudaoPipeline': 300,  
  }
* **步骤6 - run.py**
* from scrapy import cmdline  
    
  cmdline.execute('scrapy crawl youdao'.split())

### scrapy添加cookie的三种方式

【1】修改 settings.py 文件  
 1.1) COOKIES\_ENABLED = False -> 取消注释,开启cookie,检查headers中的cookie  
 1.2) DEFAULT\_REQUEST\_HEADERS = {} 添加Cookie  
  
【2】利用cookies参数  
 1.1) settings.py: COOKIES\_ENABLED = True # 修改为TRUE后，检查 Request()方法中cookies  
 1.2) def start\_requests(self):  
 yield scrapy.Request(url=url,cookies={},callback=xxx)  
 yield scrapy.FormRequest(url=url,formdata=formdata,cookies={},callback=xxx)  
   
【3】DownloadMiddleware设置中间件  
 3.1) settings.py: COOKIES\_ENABLED = TRUE # 找Request()方法中cookies参数  
 3.2) middlewares.py  
 def process\_request(self,request,spider):  
 request.cookies={}

## scrapy shell的使用

* **定义**
* 【1】调试蜘蛛的工具  
  【2】交互式shell，可在不运行spider的前提下,快速调试 scrapy 代码(主要测试xpath表达式)
* **基本使用**
* # scrapy shell URL地址  
  \*1、request.url : 请求URL地址  
  \*2、request.headers ：请求头(字典)  
  \*3、request.meta ：item数据传递，定义代理(字典)  
  \*4、request.cookies : Cookie  
    
  4、response.text ：字符串  
  5、response.body ：bytes  
  6、response.xpath('')  
  7、response.status : HTTP响应码  
     
  # 可用方法  
  shelp() : 帮助  
  fetch(request) : 从给定的请求中获取新的响应，并更新所有相关对象  
  view(response) : 在本地Web浏览器中打开给定的响应以进行检查
* **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['']  
  6、cookies

## 设置中间件(随机User-Agent)

* **少量User-Agent切换**
* 【1】方法一 : settings.py  
   1.1) USER\_AGENT = ''  
   1.2) DEFAULT\_REQUEST\_HEADERS = {}  
     
  【2】方法二 : 爬虫文件  
   yield scrapy.Request(url,callback=函数名,headers={})
* **大量User-Agent切换（middlewares.py设置中间件）**
* 【1】获取User-Agent方式  
   1.1) 方法1 ：新建useragents.py,存放大量User-Agent，random模块随机切换  
   1.2) 方法2 ：安装fake\_useragent模块(sudo pip3 install fack\_useragent)  
   from fake\_useragent import UserAgent  
   agent = UserAgent().random   
     
  【2】middlewares.py新建中间件类  
   class RandomUseragentMiddleware(object):  
   def process\_request(self,reuqest,spider):  
   agent = UserAgent().random  
   request.headers['User-Agent'] = agent  
     
  【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)

class RandomProxyDownloaderMiddleware(object):  
 def process\_request(self,request,spider):  
 request.meta['proxy'] = xxx  
   
 def process\_exception(self,request,exception,spider):  
 return request

* **练习**
* 有道翻译,将cookie以中间件的方式添加的scrapy项目中

## 今日作业

【1】URL地址  
 1.1) www.so.com -> 图片 -> 美女  
 1.2) 即: https://image.so.com/z?ch=beauty  
 1.3) 抓取5页即可，共计150张图片  
   
【2】图片保存路径  
 ./images/xxx.jpg  
   
【提示】: 使用 from scrapy.pipelines.images import ImagesPipeline 管道,并重写方法  
   
settings.py: IMAGES\_STORE = '路径'

## 答案

* **抓取网络数据包**
* 【1】通过分析，该网站为Ajax动态加载  
  【2】F12抓包，抓取到json地址 和 查询参数(QueryString)  
   2.1) url = 'https://image.so.com/zjl?ch=beauty&sn={}&listtype=new&temp=1'  
   2.2) 查询参数  
   ch: beauty  
   sn: 0 # 发现sn的值在变,0 30 60 90 120 ... ...  
   listtype: new  
   temp: 1

### 项目实现

* **1、创建爬虫项目和爬虫文件**
* scrapy startproject So  
  cd So  
  scrapy genspider so image.so.com
* **2、定义要爬取的数据结构(items.py)**
* img\_url = scrapy.Field()  
  img\_title = scrapy.Field()
* **3、爬虫文件实现图片链接+名字抓取**
* import scrapy  
  import json  
  from ..items import SoItem  
    
  class SoSpider(scrapy.Spider):  
   name = 'so'  
   allowed\_domains = ['image.so.com']  
   # 重写start\_requests()方法  
   url = 'https://image.so.com/zjl?ch=beauty&sn={}&listtype=new&temp=1'  
    
   def start\_requests(self):  
   for sn in range(0,91,30):  
   full\_url = self.url.format(sn)  
   # 扔给调度器入队列  
   yield scrapy.Request(url=full\_url,callback=self.parse\_image)  
    
   def parse\_image(self,response):  
   html = json.loads(response.text)  
   item = SoItem()  
   for img\_dict in html['list']:  
   item['img\_url'] = img\_dict['qhimg\_url']  
   item['img\_title'] = img\_dict['title']  
    
   yield item
* **4、管道文件（pipelines.py）**
* from scrapy.pipelines.images import ImagesPipeline  
  import scrapy  
    
  class SoPipeline(ImagesPipeline):  
   # 重写get\_media\_requests()方法  
   def get\_media\_requests(self, item, info):  
   yield scrapy.Request(url=item['img\_url'],meta={'name':item['img\_title']})  
    
   # 重写file\_path()方法,自定义文件名  
   def file\_path(self, request, response=None, info=None):  
   img\_link = request.url  
   # request.meta属性  
   filename = request.meta['name'] + '.' + img\_link.split('.')[-1]  
   return filename
* **5、全局配置(settings.py)**
* ROBOTSTXT\_OBEY = False  
  DOWNLOAD\_DELAY = 0.1  
  DEFAULT\_REQUEST\_HEADERS = {  
   'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8',  
   'Accept-Language': 'en',  
   'User-Agent': 'Mozilla/5.0',  
  }  
  ITEM\_PIPELINES = {  
   'So.pipelines.SoPipeline': 300,  
  }  
  IMAGES\_STORE = './images/'
* **6、运行爬虫(run.py)**
* from scrapy import cmdline  
    
  cmdline.execute('scrapy crawl so'.split())