爬虫实现-Scrapy

主讲：孙国元

华信培训

本章要点

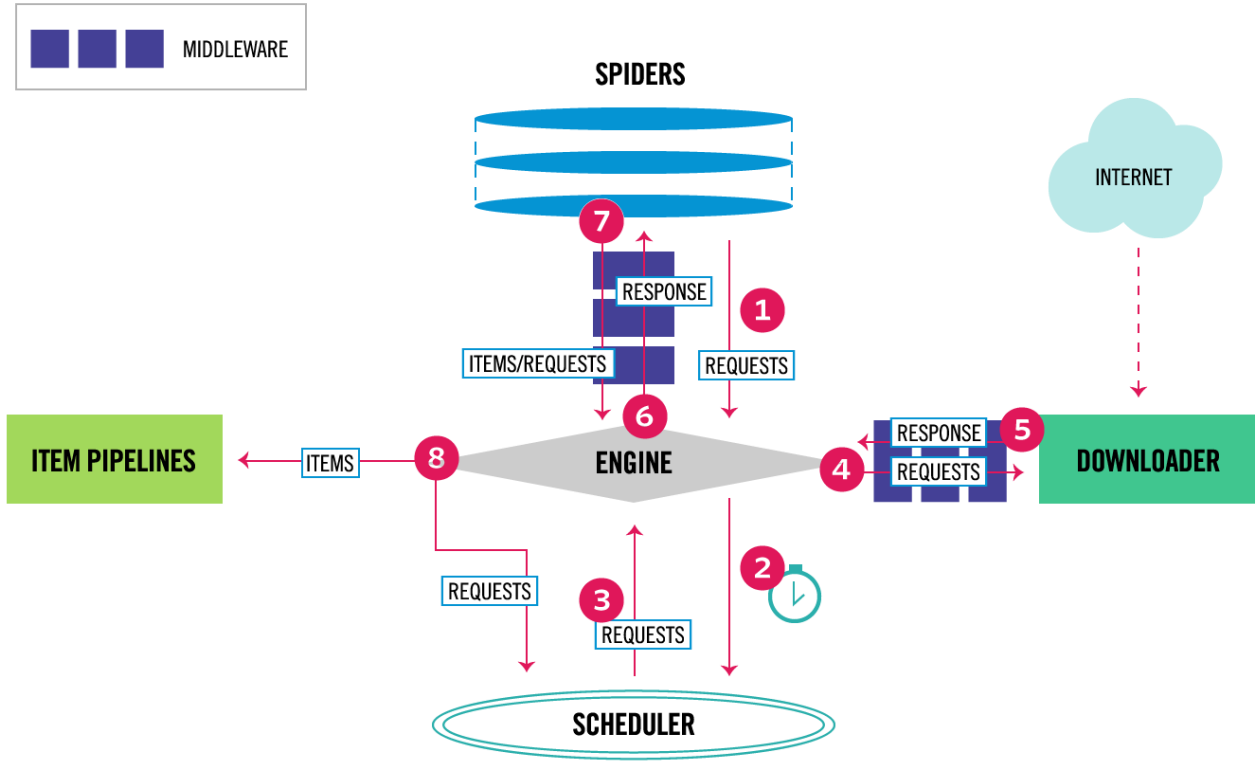
- Scrapy入门
- Scrapy进阶

1

Scrapy入门

概述

- Scrapy是用python写的一个爬虫框架，当然如果只是写一些简单爬虫，python自己就有做爬虫的库，scrapy只是更加流水线化，各部分分工更加清晰。



主要组件

- 引擎(Scrapy)
  - 用来处理整个系统的数据流, 触发事务(框架核心)
- 调度器(Scheduler)
  - 用来接受引擎发过来的请求, 压入队列中, 并在引擎再次请求的时候返回. 可以想像成一个URL（抓取网页的网址或者说是链接）的优先队列, 由它来决定下一个要抓取的网址是什么, 同时去除重复的网址
- 下载器(Downloader)
  - 用于下载网页内容, 并将网页内容返回给蜘蛛(Scrapy下载器是建立在twisted这个高效的异步模型上的)
- 爬虫(Spiders)
  - 用于从特定的网页中提取自己需要的信息, 即所谓的实体(Item)。用户也可以从中提取出链接,让Scrapy继续抓取下一个页面

主要组件

- 项目管道(Pipeline)
  - 负责处理爬虫从网页中抽取的实体，主要的功能是持久化实体、验证实体的有效性、清除不需要的信息。当页面被爬虫解析后，将被发送到项目管道，并经过几个特定的次序处理数据。
- 下载器中间件(Downloader Middlewares)
  - 位于Scrapy引擎和下载器之间的框架，主要是处理Scrapy引擎与下载器之间的请求及响应。
- 爬虫中间件(Spider Middlewares)
  - 介于Scrapy引擎和爬虫之间的框架，主要工作是处理蜘蛛的响应输入和请求输出。
- 调度中间件(Scheduler Middewares)
  - 介于Scrapy引擎和调度之间的中间件，从Scrapy引擎发送到调度的请求和响应。

安装

- 在python3.6以上版本安装scrapy框架是会报错缺少Microsoft Visual C++ Build Tools，需要首先手动安装twisted，
  http://www.lfd.uci.edu/~gohlke/pythonlibs/#twisted

```
pip install Twisted-18.4.0-cp36-cp36m-win_amd64.whl
```

- 在线安装Scrapy

```
pip install scrapy
```

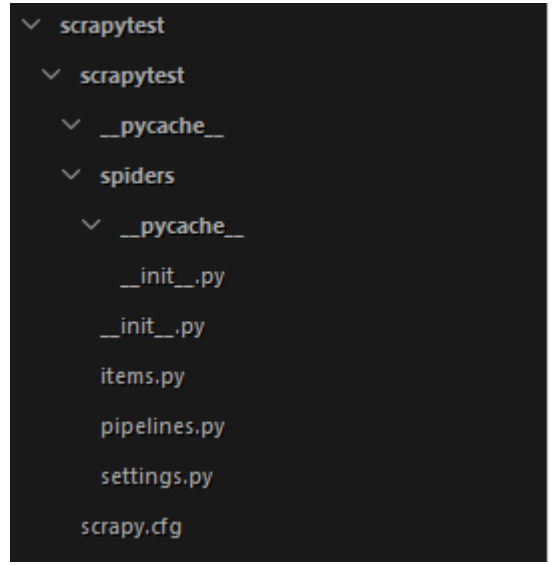
- 安装支持模块

```
pip install pypiwin32
```

工程建立

```
scrapy startproject scrapytest
```

- scrapy.cfg: 项目的配置文件
- scrapytest/: 该项目的python模块。之后您将在此加入代码。
- scrapytest/items.py: 项目中的item文件.
- scrapytest/pipelines.py: 项目中的pipelines文件.
- scrapytest/settings.py: 项目的设置文件.
- scrapytest/spiders/: 放置spider代码的目录.



创建爬虫文件

- 为了创建一个Spider，您必须继承 scrapy.Spider 类， 且定义一些属性:
  - name: 用于区别Spider。 该名字必须是唯一的，您不可以为不同的Spider设定相同的名字。
  - start_urls: 包含了Spider在启动时进行爬取的url列表。 因此，第一个被获取到的页面将是其中之一。 后续的URL则从初始的URL获取到的数据中提取。
  - parse() 是spider的一个方法。 被调用时，每个初始URL完成下载后生成的 Response 对象将会作为唯一的参数传递给该函数。 该方法负责解析返回的数据(response data)，提取数据(生成item)以及生成需要进一步处理的URL的 Request 对象。

MySpider.py

```
# -*- coding: utf-8 -*-
import scrapy


class MyspiderSpider(scrapy.Spider):
    name = 'MySpider'
    allowed_domains = ['dhee.com.cn']
    start_urls = ['http://www.dhee.com.cn/']

    def parse(self, response):
        print(response.body.decode('utf-8'))
```

xpath

- scrapy内部支持更简单的查询语法，帮助我们去html中查询我们需要的标签和标签内容以及标签属性。

表达式	描述
nodename	选取此节点的所有子节点。
/	从根节点选取。
//	从匹配选择的当前节点选择文档中的节点，而不考虑它们的位置。
.	选取当前节点。
..	选取当前节点的父节点。
@	选取属性。

xpath

- /html/head/title: 选择HTML文档中 <head> 标签内的 <title> 元素
- /html/head/title/text(): 选择上面提到的 <title> 元素的文字
- //td: 选择所有的 <td> 元素
- //div[@class="mine"]: 选择所有具有 class="mine" 属性的 div 元素

- plasmasturm.org/log/xpath101/

相关方法

- xpath(): 传入xpath表达式，返回该表达式所对应的所有节点的selector list列表 。
- css(): 传入CSS表达式，返回该表达式所对应的所有节点的selector list列表.
- extract(): 序列化该节点为unicode字符串并返回list。
- re(): 根据传入的正则表达式对数据进行提取，返回unicode字符串list列表。

MySpider.py

```
# -*- coding: utf-8 -*-
import scrapy


class MySpider(scrapy.Spider):
    name = 'MySpider'
    allowed_domains = ['dhee.com.cn']
    start_urls = ['http://dhee.com.cn/EVSelectIndex.do']

    def parse(self, response):
        titles = response.xpath("//title/text()").extract()
        print(titles)
```

定义输出数据

- 为了定义常用的输出数据，Scrapy提供了Item类。Item对象是种简单的容器，保存了爬取到得数据。 其提供了 类似于词典(dictionary-like)的API以及用于声明可用字段的简单语法。
- 我们在工程目录下可以看到一个items文件，我们可以更改这个文件或者创建一个新的文件来定义我们的item。

items.py

- 常用方法

```
import scrapy


class DemoItem(scrapy.Item):
    # define the fields for your item here like:
    # name = scrapy.Field()
    news_title = scrapy.Field()
    news_date = scrapy.Field()
    news_intro = scrapy.Field()
```

MySpider.py

```
import scrapy
import demo.items


class MySpider(scrapy.Spider):
    name = 'MySpider'
    allowed_domains = ['dhee.com.cn']
    start_urls = ['http://dhee.com.cn/EVSelectNews.do?newsType=1']

    def parse(self, response):
            for n in response.xpath("//div[@class='col-xs-12 col-md-6 item nobrd']"):
                item = demo.items.DemoItem()
                item['news_title'] = n.xpath(".//p[@class='tit f-toe']/text()").extract()[0].strip()
                item['news_date'] = n.xpath(".//p[@class='date']/text()").extract()[0].strip()
                item['news_intro'] = n.xpath(".//p[@class='intro']/text()").extract()[0].strip()
                yield item
```

输出数据

- 在settings文件中增加：
- FEED_EXPORT_ENCODING = 'utf-8’

- scrapy crawl MySpider -o output.json
- scrapy crawl MySpider -o output.json --nolog

运行项目

- 使用main

```
import scrapy.cmdline

def main():
    scrapy.cmdline.execute(argv=['scrapy', 'crawl', 'MySpider'])

if __name__ == '__main__':
    main()
```

使用pipeline处理数据

- 当Item在Spider中被收集之后，它将会被传递到pipeline，一些组件会按照一定的顺序执行对Item的处理。
- pipeline经常进行一下一些操作：
  - 清理HTML数据
  - 验证爬取的数据(检查item包含某些字段)
  - 查重(并丢弃)
  - 将爬取结果保存到数据库中

pipelines.py

- 在pipelines.py中增加

```
class MyPipeline(object):
    def open_spider(self, spider):
        self.file = open('data.txt', 'w', encoding='utf-8')

    def process_item(self, item, spider):
        line = item['news_title'] + ',' + item['news_date'] + ',' + item['news_intro']
        self.file.write(line + '\n')
        return item

    def close_spider(self, spider):
        self.file.close()
```

注册pipeline

- 在settings.py中注册

```
ITEM_PIPELINES = {
    ‘demo.pipelines.MyPipeline': 300,
}
```

- ’ 1’为该Pipeline的优先级，范围1～1000，越小越先执行。

2

Scrapy进阶

并发设置

- 在settings.py文件中添加

```
CONCURRENT_REQUESTS = 1
DOWNLOAD_DELAY = 5
```

设置请求头

- 在settings.py文件中添加

```
DEFAULT_REQUEST_HEADERS = {
    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.84
      Safari/537.36',
    'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8',
    'Accept-Encoding': 'gzip, deflate, sdch',
    'Accept-Language': 'en-US,en;q=0.8,zh-CN;q=0.6,zh;q=0.4',
}
```

下一页

```
class MySpider(scrapy.Spider):
    name = 'MySpider'
    allowed_domains = ['dhee.com.cn']
    start_urls = ['http://dhee.com.cn/EVSelectNews.do?newsType=1']
    page = 1

    def parse(self, response):
        …
        self.page = self.page + 1
        if response.xpath("//a[@href='/EVSelectNews.do?nowPage={0}']".format(self.page)).extract_first() is not None:
            nextpage = 'http://dhee.com.cn/EVSelectNews.do?nowPage={0}'.format(self.page)
            yield scrapy.Request(nextpage, callback=self.parse)
```

代理服务器

- 代理IP可以从这个几个网站获取:快代理（http://www.kuaidaili.com/）、代理66（http://www.66ip.cn/）、有代理（http://www.youdaili.net/Daili/）、西刺代理（http://www.xicidaili.com/nt）。如果出现像下面这种提示:“由于连接方在一段时间后没有正确答复或连接的主机没有反应，连接尝试失败”或者是这种，“ 由 于目标计算机积极拒绝，无法连接。”，那就是IP的问题，更换就行了。

代理设置-方法1

- 重写start_requests()方法

```
def start_requests(self):
        for url in self.start_urls:
            yield scrapy.Request(url, callback=self.parse, meta={'proxy': 'http://218.60.8.98:3129'})]
```

代理设置-方法2

- 使用DOWNLOAD中间件，修改middlewares.py，增加

```
class ProxyMiddleware(object):
    def process_request(self, request, spider):
        request.meta['proxy'] = "http://218.60.8.98:3129"
```

- 在settings.py中，注册中间件

```
demo.middlewares.ProxyMiddleware': 100
```

MySQL Pipeline

```
class MySQLPipeline(object):
    def __init__(self):
        config = {
            'host': '127.0.0.1',
            'port': 3306,
            'user': 'root',
            'password': '123456',
            'database': 'dhee_db',
            'charset': 'utf8'
        }
        self.conn = pymysql.connect(**config)

    def process_item(self, item, spider):
        try:
            with self.conn.cursor() as cursor:
                sql = "INSERT INTO T_DHEE(news_title, news_date, news_intro) VALUES(%s, %s, %s)"
                cursor.execute(sql, (item['news_title'], item['news_date'], item['news_intro']))
            self.conn.commit()
        except:
            self.conn.rollback()
            raise
        return item

    def spider_closed(self, spider):
        self.conn.close()
```

MySQL Pipeline

- 在settings注册

```
ITEM_PIPELINES = {
    ‘demo.pipelines.MySQLPipeline': 200
}
```

下载图片

- 安装pillow

```
pip install pillow
```

下载图片

- 在items.py文件中添加如下属性

```
image_url = scrapy.Field()
```

下载图片

- 在Pipelines.py增加

```
import scrapy
from scrapy.contrib.pipeline.images import ImagesPipeline

class ImagePipeline(ImagesPipeline):
    def get_media_requests(self, item, info):
        yield Request(item['image_urls'])
```

- get_media_requests(item, info)方法是通过抓取的图片url来返回一个Request，这个Request将对图片进行下载。在下载请求完成后(下载成功或失败)就会调用item_completed()方法。
- item_completed(results, items, info)，参数results包含三个项目
  - url->图片的url，
  - path->下载后保存地址，
  - checksum->图片内容的 MD5 hash

下载图片

- 注册ImagePipeline

```
ITEM_PIPELINES = {
    'demo.pipelines.ImagePipeline': 100
}
```

- 下载图片这里需要注册一下保存地址，还是在settings.py文件

```
IMAGES_STORE = 'D:\img\'
```

本章小结

- Scrapy入门
- Scrapy进阶

华信培训