



数据科学实训营 数据抓取

寒小阳
2017年2月

背景知识：Xpath与网页内容定位

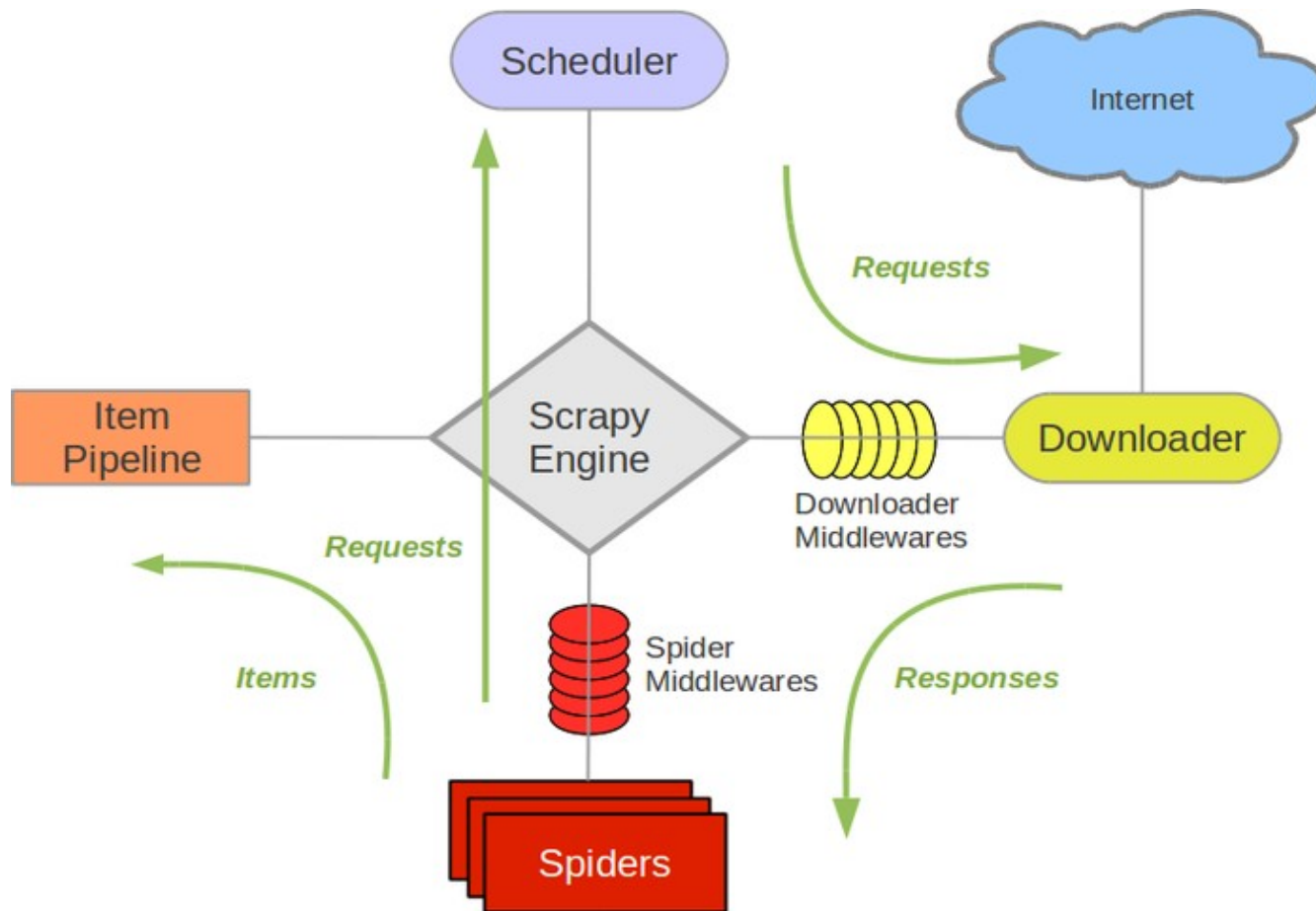
大家最爱开卷考试
请参阅课程所给xpath cheatsheet

Scrapy



<https://doc.scrapy.org/en/master/intro/overview.html>

Scrapy结构



<https://doc.scrapy.org/en/master/topics/architecture.html>

Scrapy结构

引擎(Scrapy Engine)

调度器(Scheduler)

下载器(Downloader)

蜘蛛(Spiders)

项目管道(Item Pipeline)

下载器中间件(Downloader Middlewares)

蜘蛛中间件(Spider Middlewares)

调度中间件(Scheduler Middlewares)

Scrapy工作方式

➤ 绿线是数据流向

- ◆ 从初始URL开始，Scheduler会将其交给Downloader进行下载
- ◆ 下载之后会交给Spider进行分析
- ◆ Spider分析出来的结果有两种
 - ◇ 一种是需要进一步抓取的链接，如“下一页”的链接，它们会被传回Scheduler；
 - ◇ 另一种是需要保存的数据，它们被送到Item Pipeline里，进行后期处理（详细分析、过滤、存储等）。
- ◆ 在数据流动的通道里还可以安装各种中间件，进行必要的处理。

组件spider与简单爬取

scrapy runspider spider.py -o xxx.json

```
import scrapy

class QuotesSpider(scrapy.Spider):
    name = "quotes"
    start_urls = [
        'http://quotes.toscrape.com/tag/humor/',
    ]

    def parse(self, response):
        for quote in response.css('div.quote'):
            yield {
                'text': quote.css('span.text::text').extract_first(),
                'author': quote.xpath('span/small/text()').extract_first(),
            }

        next_page = response.css('li.next a::attr("href")').extract_first()
        if next_page is not None:
            next_page = response.urljoin(next_page)
            yield scrapy.Request(next_page, callback=self.parse)
```

推荐json、xml、csv格式，方便导入数据库

Scrapy项目创建

```
→ Downloads scrapy startproject bigdatadigest_crawler
New Scrapy project 'bigdatadigest_crawler', using template directory '/Library/Python/2.7/site-packages/scrapy/templates/project',
created in:
/Users/Downloads/bigdatadigest_crawler

You can start your first spider with:
cd bigdatadigest_crawler
scrapy genspider example example.com
```

```
→ Downloads tree bigdatadigest_crawler
bigdatadigest_crawler
├── bigdatadigest_crawler
│   ├── __init__.py
│   ├── items.py
│   ├── middlewares.py
│   ├── pipelines.py
│   ├── settings.py
│   └── spiders
│       └── __init__.py
└── scrapy.cfg

2 directories, 7 files
```


scrapy shell http://你要调试xpath的网址

```
➔ Downloads scrapy shell http://edu.bigdatadigest.cn/
2017-02-01 21:23:33 [scrapy] INFO: Scrapy 1.2.2 started (bot: scrapybot)
2017-02-01 21:23:33 [scrapy] INFO: Overridden settings: {'LOGSTATS_INTERVAL': 0, 'DUPEFILTER_CLASS': 'scrapy.dupefilters.BaseDupeFilter'}
2017-02-01 21:23:33 [scrapy] INFO: Enabled extensions:
['scrapy.extensions.telnet.TelnetConsole',
 'scrapy.extensions.corestats.CoreStats']
2017-02-01 21:23:33 [scrapy] INFO: Enabled downloader middlewares:
['scrapy.downloadermiddlewares.httpauth.HttpAuthMiddleware',
 'scrapy.downloadermiddlewares.downloadtimeout.DownloadTimeoutMiddleware',
 'scrapy.downloadermiddlewares.defaultheaders.DefaultHeadersMiddleware',
 'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware',
 'scrapy.downloadermiddlewares.retry.RetryMiddleware',
 'scrapy.downloadermiddlewares.redirect.MetaRefreshMiddleware',
 'scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware',
 'scrapy.downloadermiddlewares.redirect.RedirectMiddleware',
 'scrapy.downloadermiddlewares.cookies.CookiesMiddleware',
 'scrapy.downloadermiddlewares.chunked.ChunkedTransferMiddleware',
 'scrapy.downloadermiddlewares.stats.DownloaderStats']
2017-02-01 21:23:33 [scrapy] INFO: Enabled spider middlewares:
['scrapy.spidermiddlewares.httperror.HttpErrorMiddleware',
 'scrapy.spidermiddlewares.offsite.OffsiteMiddleware',
 'scrapy.spidermiddlewares.referer.RefererMiddleware',
 'scrapy.spidermiddlewares.urllength.UrlLengthMiddleware',
 'scrapy.spidermiddlewares.depth.DepthMiddleware']
2017-02-01 21:23:33 [scrapy] INFO: Enabled item pipelines:
[]
2017-02-01 21:23:33 [scrapy] DEBUG: Telnet console listening on 127.0.0.1:6023
2017-02-01 21:23:33 [scrapy] INFO: Spider opened
2017-02-01 21:23:34 [scrapy] DEBUG: Crawled (200) <GET http://edu.bigdatadigest.cn/> (referer: None)
2017-02-01 21:23:34 [root] DEBUG: Using default logger
2017-02-01 21:23:34 [root] DEBUG: Using default logger
[s] Available Scrapy objects:
[s] scrapy scrapy module (contains scrapy.Request, scrapy.Selector, etc)
[s] crawler <scrapy.crawler.Crawler object at 0x1043cee10>
[s] item {}
[s] request <GET http://edu.bigdatadigest.cn/>
[s] response <200 http://edu.bigdatadigest.cn/>
[s] settings <scrapy.settings.Settings object at 0x1043cec90>
[s] spider <DefaultSpider 'default' at 0x104776590>
[s] Useful shortcuts:
[s] help() Shell help (print this help)
[s] fetch(req_or_url) Fetch request (or URL) and update local objects
[s] view(response) View response in a browser
In [1]:
```

scrapy组件spider

```
import scrapy

class QuotesSpider(scrapy.Spider):
    name = "quotes"
    start_urls = [
        'http://quotes.toscrape.com/tag/humor/',
    ]

    def parse(self, response):
        for quote in response.xpath('//div[@class="quote"]'):
            yield {
                'text': quote.xpath('span[@class="text"]/text()').extract_first(),
                'author': quote.xpath('span/small[@class="author"]/text()').extract_first(),
            }
```

scrapy组件spider

爬取流程

先初始化请求URL列表，并指定下载后处理response的回调函数。

在parse回调中解析response并返回字典,Item对象,Request对象或它们的迭代对象。

在回调函数里面，使用选择器解析页面内容，并生成解析后的结果Item。

最后返回的这些Item通常会被持久化到数据库中(使用Item Pipeline)或者使用Feed exports将其保存到文件中。

scrapy spider几种爬取方式

爬取1页内容

按照给定列表爬取多页

“下一页”类型

按照链接进行爬取

见课上代码讲解

不同类型spider

CrawlSpider

链接爬取蜘蛛

属性rules: Rule对象列表, 定义规则

```
rules = (  
    # 提取匹配正则式 '/group?f=index_group' 链接 (但是不能匹配 'deny.php')  
    # 并且会递归爬取 (如果没有定义callback, 默认follow=True).  
    Rule(LinkExtractor(allow=('/group?f=index_group', ), deny=('deny\.php', ))),  
    # 提取匹配 '/article/\d+/\d+.html' 的链接, 并使用parse_item来解析它们下载后的内容, 不递归  
    Rule(LinkExtractor(allow=('/article/\d+/\d+.html', ), callback='parse_item'),  
)
```

不同类型spider

XMLFeedSpider

XML订阅蜘蛛，通过某个指定的节点来遍历

```
from julyeduscrapy.items import BlogItem
import scrapy
from scrapy.spiders import XMLFeedSpider

class XMLSpider(XMLFeedSpider):
    name = "xml"
    namespaces = [('atom', 'http://www.w3.org/2005/Atom')]
    allowed_domains = ["github.io"]
    start_urls = [
        "http://www.pycoding.com/atom.xml"
    ]
    iterator = 'xml' # 缺省的iternodes，貌似对于有namespace的xml不行
    itertag = 'atom:entry'

    def parse_node(self, response, node):
        # self.logger.info('Hi, this is a <%s> node!', self.itertag)
        item = BlogItem()
        item['title'] = node.xpath('atom:title/text()')[0].extract()
        item['link'] = node.xpath('atom:link/@href')[0].extract()
        item['id'] = node.xpath('atom:id/text()')[0].extract()
        item['published'] = node.xpath('atom:published/text()')[0].extract()
        item['updated'] = node.xpath('atom:updated/text()')[0].extract()
        self.logger.info('|'.join([item['title'], item['link'], item['id'], item['published']]))
        return item
```

不同类型spider

CSVFeedSpider

类似XML订阅蜘蛛

逐行迭代，调用parse_row()解析

```
from julyeduscrapy.items import BlogItem
from scrapy.spiders import CSVFeedSpider

class CSVSpider(CSVFeedSpider):
    name = "csv"
    allowed_domains = ['example.com']
    start_urls = ['http://www.example.com/feed.csv']
    delimiter = ';'
    quotechar = '"'
    headers = ['id', 'name', 'description']

    def parse_row(self, response, row):
        self.logger.info('Hi, this is a row!: %r', row)
        item = BlogItem()
        item['id'] = row['id']
        item['name'] = row['name']
        return item
```

Scrapy组件Item

保存数据的地方

Item Loader可

```
class JulyeduStudent(scrapy.Item):  
    num = scrapy.Field()  
    age = scrapy.Field()  
    name = scrapy.Field()  
    last_updated = scrapy.Field(serializer=str)
```

```
from scrapy.loader import ItemLoader  
from myproject.items import JulyeduStudent  
  
def parse(self, response):  
    l = ItemLoader(item=Product(), response=response)  
    l.add_xpath('name', '//div[@class="student_name"]')  
    l.add_xpath('num', '//div[@class="student_num"]')  
    l.add_xpath('age', '//p[@id="age"]')  
    l.add_value('last_updated', 'today') # you can also use literal values  
    return l.load_item()
```


Scrapy组件Item Pipeline

当一个item被蜘蛛爬取到之后会被发送给Item Pipeline，然后多个组件按照顺序处理这个item

Item Pipeline常用场景

- 清理HTML数据

- 验证被抓取的数据(检查item是否包含某些字段)

- 重复性检查(然后丢弃)

- 将抓取的数据存储到数据库中

Scrapy组件Item Pipeline

定义一个Python类，实现方法`process_item(self, item, spider)`即可，返回一个字典或Item，或者抛出DropItem异常丢弃这个Item。

经常会实现以下的方法：

`open_spider(self, spider)` 蜘蛛打开时执行

`close_spider(self, spider)` 蜘蛛关闭时执行

`from_crawler(cls, crawler)` 可访问核心组件比如配置和信号，并注册钩子函数到Scrapy中

Scrapy组件Item Pipeline

```
import pymongo

class MongoPipeline(object):

    collection_name = 'scrapy_items'

    def __init__(self, mongo_uri, mongo_db):
        self.mongo_uri = mongo_uri
        self.mongo_db = mongo_db

    @classmethod
    def from_crawler(cls, crawler):
        return cls(
            mongo_uri=crawler.settings.get('MONGO_URI'),
            mongo_db=crawler.settings.get('MONGO_DATABASE', 'items')
        )

    def open_spider(self, spider):
        self.client = pymongo.MongoClient(self.mongo_uri)
        self.db = self.client[self.mongo_db]

    def close_spider(self, spider):
        self.client.close()

    def process_item(self, item, spider):
        self.db[self.collection_name].insert(dict(item))
        return item
```

代码与示例

请参考课上示例

法律声明

本课件以及大数据文摘网站和QQ群内的所有内容，包括但不限于演示文稿、软件、声音、图片、视频、代码等，由发布者本人和大数据文摘共同享有。未经大数据文摘的明确书面特别授权，任何人不得翻录、发行、播送、转载、复制、重制、改动或利用大数据文摘的局部或全部的内容或服务或在非大数据文摘所属的服务器上作镜像，否则以侵权论，依法追究法律责任。

本网站享有对用户在本网站活动的监督和指导权，对从事网上非法活动的用户，有权终止对其所有服务。

课程详情咨询

微信客服：shujupeixun

QQ客服：3530548572