Scrapy 是什么?

写爬虫，要导入和操作不同的模块，比如requests模块、gevent库、csv模块等。而在Scrapy里，你不需要这么做，因为很多爬虫需要涉及的功能，比如麻烦的异步，在Scrapy框架都自动实现了。

图解

上面的这张图是Scrapy的整个结构。你可以把整个Scrapy框架看成是一家爬虫公司。最中心位置的Scrapy Engine(引擎）就是这家爬虫公司的大boss，负责统筹公司的4大部门，每个部门都只听从它的命令，并只向它汇报工作。

Scheduler(调度器)

部门主要负责处理引擎发送过来的requests对象（即网页请求的相关信息集合，包括params，data，cookies，request headers…等），会把请求的url以有序的方式排列成队，并等待引擎来提取（功能上类似于gevent库的queue模块）。

Downloader（下载器）

部门则是负责处理引擎发送过来的requests，进行网页爬取，并将返回的response（爬取到的内容）交给引擎。它对应的是爬虫流程【获取数据】这一步。

Spiders(爬虫)部门

是公司的核心业务部门，主要任务是创建requests对象和接受引擎发送过来的response（Downloader部门爬取到的内容），从中解析并提取出有用的数据。它对应的是爬虫流程【解析数据】和【提取数据】这两步。

Item Pipeline（数据管道）部门

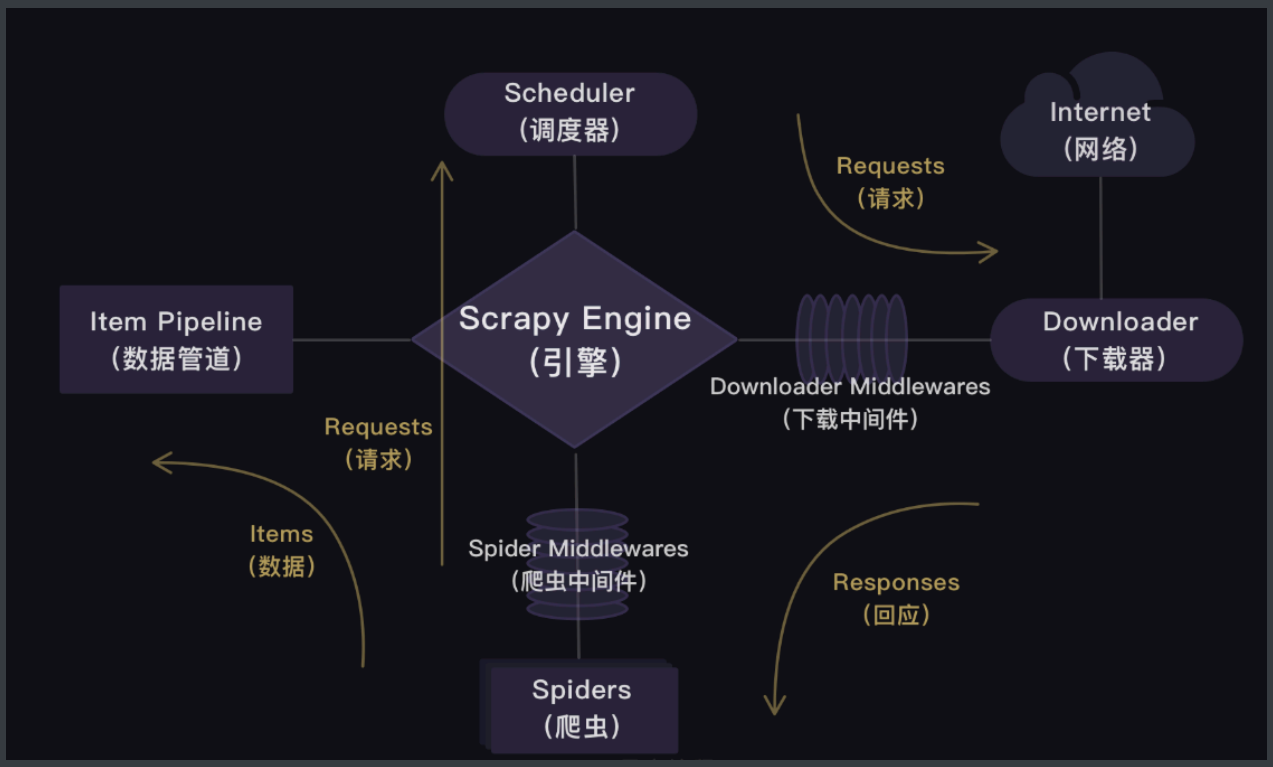
则是公司的数据部门，只负责存储和处理Spiders部门提取到的有用数据。这个对应的是爬虫流程【存储数据】这一步。

Downloader Middlewares（下载中间件）

的工作相当于下载器部门的秘书，比如会提前对引擎大boss发送的诸多requests做出处理。

Spider Middlewares（爬虫中间件）

的工作则相当于爬虫部门的秘书，比如会提前接收并处理引擎大boss发送来的response，过滤掉一些重复无用的东西。



用法图解

数据存储为xlsx的设置

setting.py里设置启用ITEM_PIPELINES，取消ITEM_PIPELINES的注释（删掉#）即可。

```
#取消`ITEM_PIPELINES`的注释后:

# Configure item pipelines
# See https://doc.scrapy.org/en/latest/topics/item-pipeline.html
ITEM_PIPELINES = {
     'jobuitest.pipelines.JobuitestPipeline': 300,
}
```

然后修改抓取协议设置

```
#需要修改的默认设置:

# Crawl responsibly by identifying yourself (and your website) on the user-agent
#USER_AGENT = 'douban (+http://www.yourdomain.com)'

# Obey robots.txt rules
ROBOTSTXT_OBEY = True
```

修改下载速度的设置,我们需要取消DOWNLOAD_DELAY = 0这行的注释（删掉#）。
DOWNLOAD_DELAY翻译成中文是下载延迟的意思，这行代码可以控制爬虫的速度。因为这个项目的爬取速度不宜过快，我们要把下载延迟的时间改成0.5秒。

```
# Configure a delay for requests for the same website (default: 0)
# See https://doc.scrapy.org/en/latest/topics/settings.html#download-delay
# See also autothrottle settings and docs
DOWNLOAD_DELAY = 0.5
```

去编辑pipelines.py文件。存储为Excel文件，我们依旧是用openpyxl模块来实现

实例代码

```
import openpyxl

class JobuiPipeline(object):
#定义一个JobuiPipeline类，负责处理item
    def __init__(self):
    #初始化函数 当类实例化时这个方法会自启动
        self.wb =openpyxl.Workbook()
        #创建工作薄
        self.ws = self.wb.active
        #定位活动表
        self.ws.append(['公司', '职位', '地址', '招聘信息'])
        #用append函数往表格添加表头

    def process_item(self, item, spider):
    #process_item是默认的处理item的方法，就像parse是默认处理response的方法
        line = [item['company'], item['position'], item['address'],
item['detail']]
        #把公司名称、职位名称、工作地点和招聘要求都写成列表的形式，赋值给line
        self.ws.append(line)
        #用append函数把公司名称、职位名称、工作地点和招聘要求的数据都添加进表格
        return item
        #将item丢回给引擎，如果后面还有这个item需要经过的itempipeline，引擎会自己调
度

    def close_spider(self, spider):
    #close_spider是当爬虫结束运行时，这个方法就会执行
        self.wb.save('./jobui.xlsx')
        #保存文件
        self.wb.close()
        #关闭文件
```