1 基于scrapy 框架的爬虫设计

* 1. 搭建scrapy框架

为了获取招聘网站页面数据需要使用python来进行编程。Scrapy框架是目前用于抓取页面数据的一个非常流行的框架。它集页面抓取、数据存储以及设置于一体。在这个框架下我不需要做过多的设计，仅仅需要设定一些特定的参数在其中。由于scrapy包不是python默认的包。首先，需要使用pip指令将必备的scrapy框架包装入python的安装目录下。

在安装完成scrapy工具包之后，在Windows CMD中执行如下指令来创建一个爬虫目录（对于两个网站页面信息的获取是一样的这里仅以jobtotal网站为例）：

scrapy startproject jobtotal

运行上述指令后，目录中会自动生成一个jobtotal的项目文件夹。根据进一步的提示指令，scrapy genspider crawl job jobtotal.com运行生成一个名字为job的爬虫文件。这一步的目的是将爬虫网页的域名限定在了jobtotal.com之中。至此，一个完整的scrapy框架就搭建完成了。该文件中会根据 crawl 模板生成自定的爬虫类 zhaopin.该类继承 scrapy.Spider 类的方法和属性。这里可以充分的显现出 python 的 scrapy 框架的易用性、高封装性。使得爬虫项目变 得更容易上手，减少重复而繁琐的前期工作，提高了开发效率。

* 1. 本项目中网站的反爬虫的通用解决方法

随着大数据的技术发展，人们开始越来越重视也更加需求数据本身的价值， 网络中页面爬虫工具也开始日益增多。网页数据开始成为大家相互争夺的重要资源之一。与此同时，网站也开始更加注重保护自己的数据资源，防止被竞争对手获取核心数据。由于爬虫机器人能够在短时间内会对网站进行大批量的访问因此容易导致网站服务器的超负荷运行。最糟糕的情况就是多人同时大批量的访问网站会导致服务器崩溃。因此，反爬虫的关键目标在于阻止批量获取网站信息。

在http请求中，用户通过网页浏览器正常浏览的过程中，数据请求体的头部信息会带上该浏览器的标识信息。为了能够在爬虫中模拟该请求，我先通过Chrome浏览器真实访问了需要访问的页面，通过所截取到的头部信息放入setting.py中。setting.py是爬虫的全局配置文件，通过设置内容scrapy会在运行时自动加载。具体配置信息如图1。如在 Headers 的 User-Agent 当中，显示的字段如下： Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML， like Gecko) Chrome/62.0.3202.75 Safari/537.36。该头部表明了，此次请求是通过 Mozilla浏览器发起的并附带诸如cookies等其他信息。通过这样的头部处理，达到了伪装浏览器请求的效果。使得在爬取数据的过 程中，网站服务器对该请求会识别为浏览器的常规请求。同时，由于请求次数较多，可以再进一 步的准备好一组浏览器头部列表，对后续每一次的请求随机的分配不同的请求头 部，使得网站在识别爬虫项目的大批量请求时，会认为是从多个浏览器中发起的。

此外，为了避免高频请求带来的反爬虫处理，在 scrapy 爬虫项目中可以通过在setting.py中设置请求频率来提高多个请求的间隔时间以此来规避异常请求行为。由于设置了请求间隔，虽然会对请求网页的效率造成一定影响，但能够避免对对方服务器造成不必要的负载。因此，我的每一次请求都能够成功。该操作在 scrapy 项目中已被高度封装，可以在 setting.py 中进行全局设置。如下：

设置下载延迟时间: DOWNLOAD\_DELAY = 1

设置最大请求延迟时间: AUTOTHROTTLE\_MAX\_DELAY = 60

通过上述应对网站反爬虫策略的设计，我发现对于jobtotal和51job这两个网站都能够成功解决反爬虫策略。

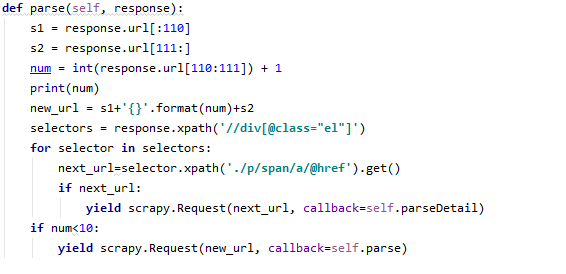
* 1. 网页信息的抓取

对于jobtotal和51job这两个招聘网站我所采取的信息抓取方式在总体上都使用同一个策略。首先通过关键字搜索职位，在本项目中为了保证数据有可对照性因此均选用了4个关键词。它们分别是python，java，JavaScript，PHP。选择它们的目的是在目前python以及java是目前程序员最常用的两种面向对象语言，它的求职面广以及职业机遇多是它们的特点。而后两者则是在网页制作的过程中两个常见的编程语言。通过搜索关键字之后网站会以一个列表的形式将职位信息展示出来。但是，在这张列表中获取到的信息量并不是特别详尽，求职信息更多的内容还是在岗位详情页中所体现。这也符合了我们在浏览器中正常浏览一个职位信息的操作。因此，在岗位列表页中，我选择先获取每一页的详情页面的URL，先将它们存储在容器中。在获取完这一个列表页的所有岗位信息详情页的URL之后，scrapy再对详情页面进行遍历访问，每一次的访问间隔都是1秒钟。

在本项目中对于网页信息的读取我采用xpath方法来实现。xpath 是一种用来确定 XML 结构文档信息的一门路径语言。本次爬取 Web 数 据的过程中，xpath 将读取 html 树状结构的信息，根据节点信息结构的不同主要分为三类：元素节点、属性节点和文本节点，xpath 根据指定好的路径信息在 html 的 DOM 结构中寻找相对应的节点。这样通过路径定位到需要的页面节点，通过此 方式来获取节点有用的信息，再对信息进行整理即可获得所需的源数据。

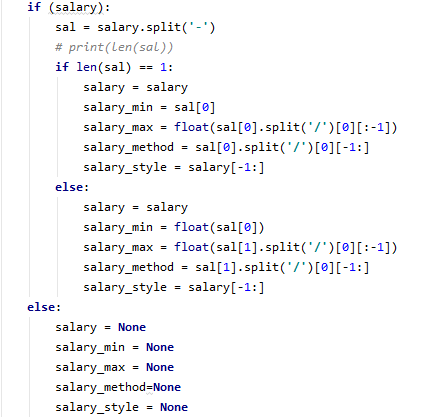
分析完所需的数据信息及如何解析节点信息后开始编写 scrapy 爬虫项目的 spider 模块。。由于自定义的类继承至 scrapy 的 Spider 类，因此该类中将定义爬取得初始动作， 以及后续是否跟进爬虫的逻辑判断。具体步骤如下：

1. 选取爬取的初始页面，在start\_urls 属性中填入该初始页面的 url 值。
2. parse 方法中将会获取到指定 URL请求返回的数据。在这个方法中将实现解 析详情页连接，并在存在有效链接的基础上继续请求详情页数据，在该请求中会指定 callback 指定其返回数据的回调函数 parseDetail，用于处理返回数据的解析操作。此外在这一函数中还将实现下一个列表页面的URL拼接。通过多次试验我发现在51job网页中下一个页面的URL仅仅与start\_url中的第111个数字有关所以每次仅需要替换这个字符即可。而在jobtotal网页中仅仅需要替换最后一个字符即可。最后一步通过观察到的最大页数设定翻页次数，以此来让程序实现自动翻页功能。

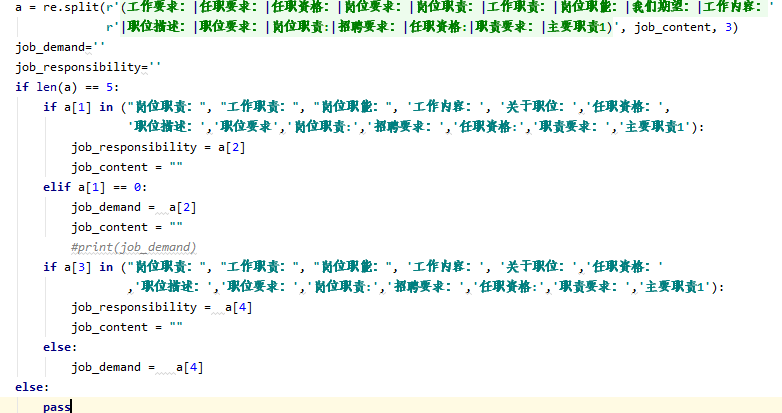


1. parseDetail方法中将获取到详情页的返回数据，通过上面表格中所示的 xpath 路径，在返回值 response 中提取所需的数据信息。经过简单的处理后将其绑定至定义的 Item 类属性中，返回给 pipelines 组件进行下一步的流程处理。

由于51job网站的数据内容结构较为统一，因此在获取数据的同时也进行了简单的处理。在这过程中由于搜集到的数据中的工资部分都是一个区间段的字符串，所以我将它们进行剥离来获得最低工资、最高工资，以及工资的支付方式（按年支付、按月支付、按日支付）。



除此之外，在岗位描述部分不难发现，大多数的岗位内容划分为岗位职责和岗位内容。通过使用正则表达式就能够将岗位职责与岗位内容进行区分开。



* 1. 网页信息的本地存储

从上述方式获得的数据只有通过存储到文件中才能够被更好地运用。在scrapy运行中，如果不进行本地输出，那么所有的数据都会一条一条地在CMD中显示出来，这对于我接下来的分析工作就毫无价值。Scrapy框架自己就有数据输出的方式，在运行scrapy时加上-o filename.csv就可以以CSV格式直接输出所获取到的网页信息。但是这一方法容易造成数据的冗余出现。例如，我在搜索关键词为python的岗位时有些岗位可能会出现Java这一个关键字，那么当我在搜索java时这个招聘信息里面也会有python的关键字。但事实上它们是同一家公司发出的同一条招聘信息。Scrapy框架还提供了将信息存储到数据库的方法。

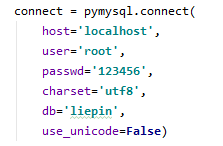
通过 spider 模块，已经将请求的数据返回给 pipelines 模块。在该模块中，将会进行数据的整合，以及连接数据库，转存等操作。根据上一节中对请求数据的处理，单条请求的数据信息已存入 Items 自定义的属性中。确定完数据结构后，进行连接数据库操作。在模块中引入 pymysql库，通过该库连接本地的mysql数据库，进行转存操作。可以通过简单的代码对远程数据库进行新增数据库，转存数据文档的操作。具体流程如下：

1. 引入pymysql包以及全局配置

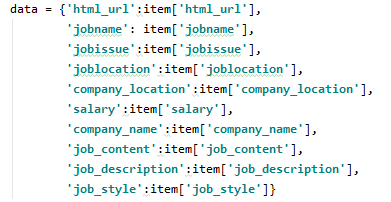
**import** pymysql

**from** scrapy.exceptions **import** DropItem

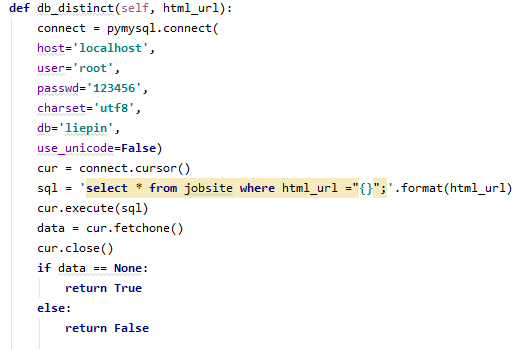
1. Connect local mysql database



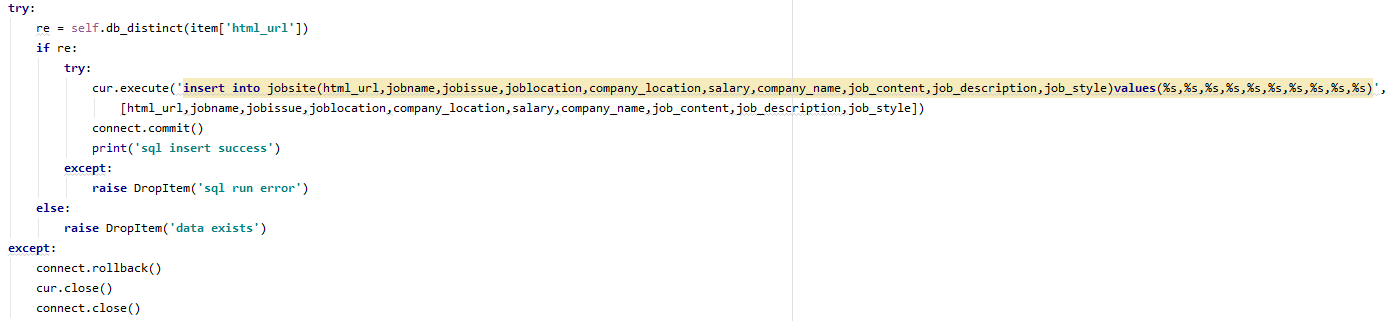
1. 在pipelines 的 process\_item 方法中，格式化获取的 item 数据



1. 执行db\_distinct function来判断是否为之前写入过的数据。在这一步就先完成数据去重工作，保证每一条被记录到mysql数据库中的数据都是唯一的。去重的方法时依据岗位详情页的URL来判断，为什么这么做的理由是，每一个岗位的详情页是有且仅有一个。



1. 将格式化后的数据运用sql语句存入mysql数据库中。执行db\_distinct function，并编写try函数判断写入是否成功。



Therefore, the data gathering from the website is stored into my local database. Then, the data can be exported from the Mysql Workbench 8.0 which is a Mysql Database visualization tool.