项目文档（作业二）

路线三 11 组
组长：
宋定杰 学号：171250628 手机：18851132226

组员：
姓名：李辰辉 学号：171250645
姓名：梁斌 学号：171830506
姓名：陈维烨 学号：171250599

组员分工

陈维烨：数据爬取，清洗，整理
梁斌：数据库搭建，后端数据持久层搭建，数据集成
宋定杰：服务端搭建
李辰辉：前端搭建

项目 git 地址

https://github.com/bbsngg/Data-Integration/tree/master/data

数据获取与数据整理

数据源

<http://pycs.greedyai.com/>：股票的公司基本信息以及发行信息数据源
<tushare 库>：股票行业及概念信息数据源

获取方式

通过 Scrapy 框架爬取股票的公司基本信息以及发行信息，通过直接调用 tushare 库的 API 获取股票行业及概念信息。

Scrapy 爬虫部分，首先在首页 http://pycs.greedyai.com/ 获取所提供的所有股票代码信息所在的网页链接，并将链接存储于文本文件中。

爬取具体数据之前，先从第一部分获取的链接文本文件中读取所有的 url，存放于临时数组中，在调用回调函数需要传入下一个爬取的页面链接时，就可以通过全局索引，从临时数组中获取下一个爬取页面的 url 并传递给回调函数。

在股票信息页面，观察到大部分的信息都以表格形式展现，因此可以获取页面所有的 td 标签，通过遍历判断的形式获取所需的信息。对于所有获取到的信息，为了保证一只股票相关所有字段信息均位于一行内，需要进行去空白格式化处理。

最后，由于部分股票的部分信息在页面上显示为空或"-"，因此在最终导入数据库前进行格式化预处理，将所有无效信息字段值替换为字符串"NULL"，至此完成数据获取。

关键代码

爬取所有股票信息所在页面的 url

```
class URLSpider(scrapy.Spider):
    name = "url"
    start_urls = ['http://pycs.greedyai.com/']

    def parse(self, response):
        global index
        # global currentUrl

        temp = response.css('a::attr(href)').extract()

        for link in temp:
            relative_url = link
            stop = relative_url
            with open(r'G:\scpy\MyDtScpy\MyDtScpy\url.txt', 'a') as namesFile:
                namesFile.writelines(relative_url+'\n')

        namesFile.close()
```

读取所有 url 至临时数组

```
with open(r'G:\scpy\MyDtScpy\MyDtScpy\url.txt', 'r') as urlsFile:
    urlsFromFile = urlsFile.readlines()
    urlsFile.close()

urlset = {urlsFromFile[0]}
urls = []

for ns in urlsFromFile:
    urlset.add(ns)

for s in urlset:
    urls.append(s)
```

爬取所需股票字段（遍历 td 标签）

```
company_regi = '-' #注册资金
company_addr = '-' #办公地址
company_boss = '-' #董事长

for trs in details:
    head = trs.css('strong::text').get()
    temp_msg = trs.css('span::text').get()
    if temp_msg is not None:
        if head=="公司名称：":
            company_name = trs.css('span::text').get().strip()
        elif head=="董事长：":
            company_boss = trs.css('span a::text').get().strip()
        elif head=="所属地域：":
            company_area = trs.css('span::text').get().strip()
        elif head=="所属行业：":
            company_type = trs.css('span::text').get().strip()
        elif head=="主营业务：":
            company_main = trs.css('span::text').get().strip()
        elif head=="控股股东：":
            company_host = trs.css('span::text').get().strip()
        elif head=="注册资金：":
            company_regi = trs.css('span::text').get().strip()
        elif head=="办公地址：":
            company_addr = trs.css('span::text').get().strip()
```

```
publish_pe = '-' #发行市盈率
publish_predict = '-' #预计募资
publish_first = '-' #首日开盘价
publish_rate = '-' #发行中签率
publish_actual = '-' #实际募资
publish_main = '-' #主承销商
publish_guar = '-' #上市保荐人

for trs in publish:
    head = trs.css('strong::text').get()
    temp_msg = trs.css('span::text').get()
    if temp_msg is not None and len(temp_msg.strip())>0:
        if head=="发行数量：":
            publish_number = trs.css('span::text').get().strip().strip('\t')
        elif head=="发行价格：":
            publish_price = trs.css('span::text').get().strip().strip('\t')
        elif head=="发行市盈率：":
            publish_pe = trs.css('span::text').get().strip().strip('\t')
        elif head=="预计募资：":
            publish_predict = trs.css('span::text').get().strip().strip('\t')
        elif head=="首日开盘价：":
            publish_first = trs.css('span::text').get().strip().strip('\t')
        elif head=="发行中签率：":
            publish_rate = trs.css('span::text').get().strip().strip('\t')
        elif head=="实际募资：":
            publish_actual = trs.css('span::text').get().strip().strip('\t')
        elif head=="主承销商：":
            publish_main = trs.css('span::text').get().strip().strip('\t')
```

获取下一个爬取页面 url 并调用回调函数

```
index = index + 1
if index < len(urls):
    next_page = urls[index]
    # next_page = response.urljoin(relative_path)
    yield Request(next_page, callback=self.parse_tastypage)
```