链家爬虫作业报告

实验环境

- 操作系统: Ubutun 20.04
- python 3.10.13
- scrapy 2.10.0

实验过程

新建爬虫文件:

```
scrapy startproject lianjia
```

在 `items.py` 文件中定义存储的数据:

```
# 新房
class NewHouseItem(scrapy.Item):
    name = scrapy.Field()  # 楼盘名称
    type = scrapy.Field()  # 类型
    location = scrapy.Field()  # 地点
    room_type = scrapy.Field()  # 房型
    area = scrapy.Field()  # 面积
    unit_price = scrapy.Field()  # 单价
    total_price = scrapy.Field()  # 总价


# 二手房
class OldHouseItem(scrapy.Item):
    community = scrapy.Field()  # 小区名称
    location = scrapy.Field()  # 地点
    room_type = scrapy.Field()  # 房型信息
    unit_price = scrapy.Field()  # 单价
    total_price = scrapy.Field()  # 总价
```

在 `spider` 文件中编写爬虫程序, 一共包含两个程序, 分别命名为 `new_house` 和 `old_house`

使用 XPath 定位网页元素, 并自定义管道, 新房爬虫代码如下:

```
class NewHouseSpider(scrapy.Spider):
    """
    爬取新房数据
    """
    name = "new_house"
    allowed_domains = ["bj.fang.lianjia.com"]
    start_urls = ['https://bj.fang.lianjia.com/loupan/']
    custom_settings = {
        'ITEM_PIPELINES': {"lianjia.pipelines.NewHouseItemPipeline": 300, }
    }

    def start_requests(self):
        for page in range(3, 3 + 5):
            yield Request(url=f'https://bj.fang.lianjia.com/loupan/pg{page}/', callback=self.par

    def parse(self, response: HtmlResponse, **kwargs):
        list_items = response.xpath(
            '/html/body/div[@class="resblock-list-container clearfix"]/ul[@class="resblock-list-
        for item in list_items:
            new_house_item = NewHouseItem()
            new_house_item['name'] = item.xpath(
                './div[@class="resblock-name"]/a[@class="name "]/text()').extract_first()
            new_house_item['type'] = item.xpath(
                './div[@class="resblock-name"]/span[@class="resblock-type"]/text()').extract_fir
            new_house_item['location'] = "".join(item.xpath('./div[@class="resblock-location"]/*
            new_house_item['room_type'] = item.xpath('./a[@class="resblock-room"]/span/text()')
            new_house_item['area'] = item.xpath('./div[@class="resblock-area"]/span/text()').ext
            new_house_item['unit_price'] = "".join(
                item.xpath('./div[@class="resblock-price"]/div[@class="main-price"]/*/text()').e
                .replace(u'\xa0', ' ')
            new_house_item['total_price'] = item.xpath('./div[@class="resblock-price"]/div[@clas
                .extract_first()
            yield new_house_item
```

在爬取二手房数据时, 小区信息和地址信息需要点进链接内再爬取, 所以爬取二手房的代码增加了一个回调函数:

```
class OldHouseSpider(scrapy.Spider):
    """
    爬取二手房数据，点进链接爬取
    """
    name = "old_house"
    allowed_domains = ["bj.lianjia.com"]
    start_urls = ['https://bj.lianjia.com/ershoufang/']
    custom_settings = {
        'ITEM_PIPELINES': {"lianjia.pipelines.OldHouseItemPipeline": 400, }
    }

    def start_requests(self):
        for page in range(3, 3 + 5):
            yield Request(url=f'https://bj.lianjia.com/ershoufang/pg{page}/', callback=self.pars

    """
    爬取地点详细信息
    """
    def location_info(self, response: HtmlResponse, **kwargs):
        location_item = response.xpath(
            '/html/body/div[@class="overview"]/div[@class="content"]/div[@class="aroundInfo"]')
        old_house_item = kwargs['item']
        old_house_item['community'] = location_item.xpath(
            './div[@class="communityName"]/a[@class="info "]/text()').extract_first()
        old_house_item['location'] = "".join(
            location_item.xpath('./div[@class="areaName"]/span[@class="info"]//text()').extract(
            .replace(u'\xa0', ' ')
        yield old_house_item

    def parse(self, response: HtmlResponse, **kwargs):
        list_items = response \
            .xpath('/html/body/div[@class="content "]/div[@class="leftContent"]/ul/li')
        for item in list_items:
            detail_info_url = item.xpath('./a[@class="noresultRecommend img LOGCLICKDATA"]/@hre
            item = item.xpath('./div[@class="info clear"]')
            old_house_item = OldHouseItem()
            old_house_item['room_type'] = item.xpath(
                './div[@class="address"]/div/text()').extract_first()
            old_house_item['total_price'] = item.xpath(
                './div[@class="priceInfo"]/div[@class="unitPrice"]/span/text()').extract_first(
            old_house_item['unit_price'] = "".join(
                item.xpath('./div[@class="priceInfo"]/div[@class="totalPrice totalPrice2"]/*/te
```

```
            yield Request(url=detail_info_url, dont_filter=True, callback=self.location_info,
                          cb_kwargs={'item': old_house_item})
```

在 `pipelines.py` 文件中定义两个管道, 分别用于处理新房和二手房数据

```python
class NewHouseItemPipeline:
    def __init__(self):
        self.new_house_file = None

    def open_spider(self, spider):
        try:
            self.new_house_file = open('new_house.json', 'w', encoding='utf-8')
        except Exception as e:
            print(e)

    def process_item(self, item, spider):
        dict_item = dict(item)
        json_str = json.dumps(dict_item, ensure_ascii=False) + "\n"
        self.new_house_file.write(json_str)
        return item

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


class OldHouseItemPipeline:
    def __init__(self):
        self.old_house_file = None

    def open_spider(self, spider):
        try:
            self.old_house_file = open('old_house.json', 'w', encoding='utf-8')
        except Exception as e:
            print(e)

    def process_item(self, item, spider):
        dict_item = dict(item)
        json_str = json.dumps(dict_item, ensure_ascii=False) + "\n"
        self.old_house_file.write(json_str)
        return item

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

在 `middlewares.py` 文件中定义一个随机 `User-Agent` 中间件, 用于随机生成 `User-Agent`：

```
class RandomUserAgentMiddleware(object):
    def __init__(self, user_agents):
        self.user_agents = user_agents

    @classmethod
    def from_crawler(cls, crawler):
        # 从settings.py中导入MY_USER_AGENT
        s = cls(user_agents=crawler.settings.get('USER_AGENT'))
        return s

    def process_request(self, request, spider):
        agent = random.choice(self.user_agents)
        request.headers['User-Agent'] = agent
        return None
```

在 settings.py 文件中添加配置, 如配置 User-Agent (见 settings.py 文件), 激活管道, 激活中间件, 确定数据编码等:

```
DOWNLOADER_MIDDLEWARES = {
    'lianjia.middlewares.RandomUserAgentMiddleware': 666,
}
ITEM_PIPELINES = {
    "lianjia.pipelines.NewHouseItemPipeline": 300,
    "lianjia.pipelines.OldHouseItemPipeline": 400,
}
AUTOTHROTTLE_ENABLED = True
FEED_EXPORT_ENCODING = "utf-8"
```

新建启动文件 begin.py , 启动两个爬虫程序:

```
from scrapy.crawler import CrawlerProcess
from scrapy.utils.log import configure_logging
from scrapy.utils.project import get_project_settings

if __name__ == "__main__":
    configure_logging()
    process = CrawlerProcess(get_project_settings())
    process.crawl("new_house")
    process.crawl("old_house")
    process.start()
```

启动程序:

```
python begin.by
```

实验结果

新房数据共 50 条, 存储在 `new_house.json` 文件中

二手房数据共 150 条, 存储在 `old_house.json` 文件中