scrapy的入门使用

学习目标：

1. 掌握 scrapy的安装
2. 应用 创建scrapy的项目
3. 应用 创建scrapy爬虫
4. 应用 运行scrapy爬虫
5. 应用 scrapy定位以及提取数据或属性值的方法
6. 掌握 response响应对象的常用属性

---

1 安装scrapy

命令:

sudo apt-get install scrapy

或者：

pip/pip3 install scrapy

2 scrapy项目开发流程

1. 创建项目:

   scrapy startproject sixstartText

2. 生成一个爬虫:

   scrapy genspider sixtartText sixstaredu.com

3. 提取数据:

   根据网站结构在spider中实现数据采集相关内容

4. 保存数据:

   使用pipeline进行数据后续处理和保存

3. 创建项目

> 通过命令将scrapy项目的的文件生成出来，后续步骤都是在项目文件中进行相关操作，下面以抓取传智师资库来学习scrapy的入门使用：https://www.sixstaredu.com/teacher?page=1

创建scrapy项目的命令：

scrapy startproject <项目名字>

示例：

scrapy startproject myspider

生成的目录和文件结果如下：

```
(python3) ➜  code tree
.
└── myspider
    ├── myspider
    │   ├── __init__.py
    │   ├── __pycache__
    │   ├── items.py          ——>自己预计需要爬取的内容
    │   ├── middlewares.py    ——>自定义中间件的文件
    │   ├── pipelines.py      ——> 管道，保存数据
    │   ├── settings.py       ——> 设置文件，UA，启动管道
    │   └── spiders           ——> 自己定义的spider的文件夹
    │       ├── __init__.py
    │       └── __pycache__
    └── scrapy.cfg            ——>项目的配置文件
```

4. 创建爬虫

> 通过命令创建出爬虫文件，爬虫文件为主要的代码作业文件，通常一个网站的爬取动作都会在爬虫文件中进行编写。

命令：

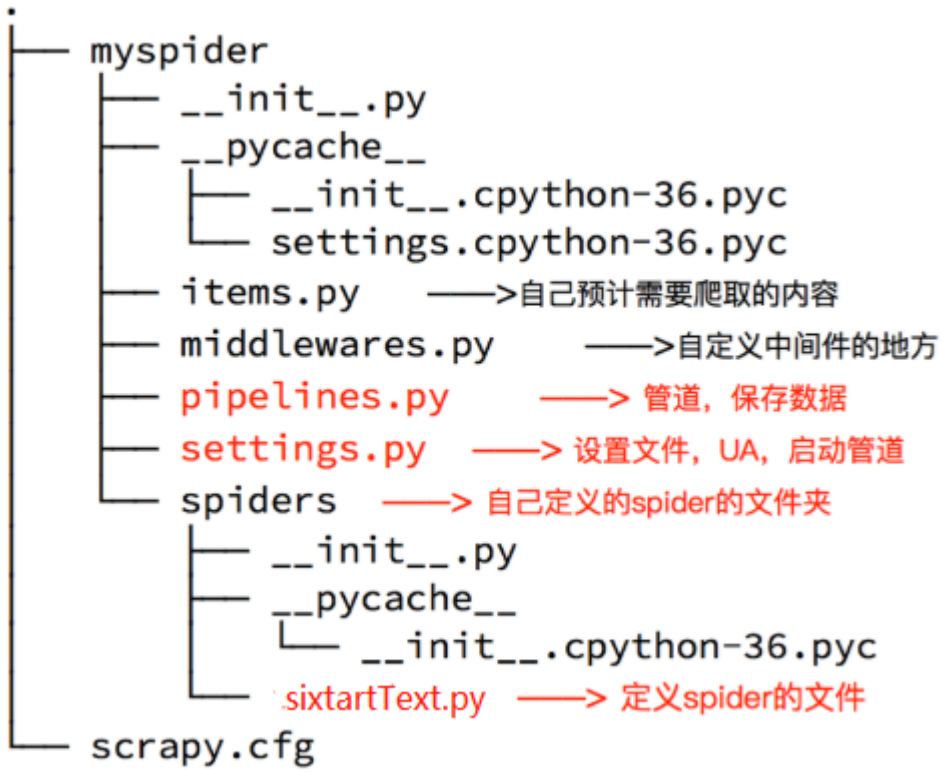
在项目路径下执行:
scrapy genspider <爬虫名字> <允许爬取的域名>

爬虫名字: 作为爬虫运行时的参数
允许爬取的域名: 为对于爬虫设置的爬取范围，设置之后用于过滤要爬取的url，如果爬取的url与允许的域不通则被过滤掉。

示例：

```
cd myspider
scrapy genspider sixtartText sixstaredu.com
```

生成的目录和文件结果如下：



5. 完善爬虫

> 在上一步生成出来的爬虫文件中编写指定网站的数据采集操作，实现数据提取

5.1 在/myspider/myspider/spiders/sixtartText.py中修改内容如下:

```
import scrapy


class SixtarttextSpider(scrapy.Spider):
    name = 'sixtartText'
    allowed_domains = ['sixstaredu.com']
    start_urls = ['http://sixstaredu.com/']

    # 数据提取方法，接受下载中间件传过来的response
    def parse(self, response):
        # scrapy的response对象可以直接进行xpath
        # names =
response.xpath('/html/body/div[1]/section[6]/div/div[2]/div[1]/div/div[1]/h3/a/text()').get
()
        # print(names)
        # 分组
        li_list = response.xpath("//div/div[@class='teacher-top']")
        for i in  li_list:
            item = {}
            item['name'] = i.xpath('./h3/a/text()').extract_first()
            print(item)
```

注意:

- scrapy.Spider爬虫类中必须有名为parse的解析

- 如果网站结构层次比较复杂，也可以自定义其他解析函数
- 在解析函数中提取的url地址如果要发送请求，则必须属于allowed_domains范围内，但是start_urls中的url地址不受这个限制，我们会在后续的课程中学习如何在解析函数中构造发送请求
- 启动爬虫的时候注意启动的位置，是在项目路径下启动
- parse()函数中使用yield返回数据，**注意：解析函数中的yield能够传递的对象只能是：BaseItem, Request, dict, None**

5.2 定位元素以及提取数据、属性值的方法

> 解析并获取scrapy爬虫中的数据: 利用xpath规则字符串进行定位和提取

1. response.xpath方法的返回结果是一个类似list的类型，其中包含的是selector对象，操作和列表一样，但是有一些额外的方法
2. 额外方法extract()：返回一个包含有字符串的列表
3. 额外方法extract_first()：返回列表中的第一个字符串，列表为空没有返回None

5.3 response响应对象的常用属性

- response.url：当前响应的url地址
- response.request.url：当前响应对应的请求的url地址
- response.headers：响应头
- response.requests.headers：当前响应的请求头
- response.body：响应体，也就是html代码，byte类型
- response.status：响应状态码

6 保存数据

> 利用管道pipeline来处理(保存)数据

6.1 在pipelines.py文件中定义对数据的操作

1. 定义一个管道类
2. 重写管道类的process_item方法
3. process_item方法处理完item之后必须返回给引擎

```
import json

class ItcastPipeline():
    # 爬虫文件中提取数据的方法每yield一次item，就会运行一次
    # 该方法为固定名称函数
    def process_item(self, item, spider):
        print(item)
        return item
```

6.2 在settings.py配置启用管道

```
ITEM_PIPELINES = {
   'sixstart.pipelines.SixstartPipeline': 300,
}
```

配置项中键为使用的管道类，管道类使用.进行分割，第一个为项目目录，第二个为文件，第三个为定义的管道类。
配置项中值为管道的使用顺序，设置的数值约小越优先执行，该值一般设置为1000以内。

7. 运行scrapy

命令：在项目目录下执行scrapy crawl <爬虫名字>

示例：scrapy crawl sixstartText

小结

1. scrapy的安装：pip install scrapy
2. 创建scrapy的项目: scrapy startproject myspider
3. 创建scrapy爬虫：在项目目录下执行 scrapy genspider sixstartText sixstartText.cn
4. 运行scrapy爬虫：在项目目录下执行 scrapy crawl sixstartText
5. 解析并获取scrapy爬虫中的数据：
    1. response.xpath方法的返回结果是一个类似list的类型，其中包含的是selector对象，操作和列表一样，但是有一些额外的方法
    2. extract() 返回一个包含有字符串的列表
    3. extract_first() 返回列表中的第一个字符串，列表为空没有返回None
6. scrapy管道的基本使用:
    1. 完善pipelines.py中的process_item函数
    2. 在settings.py中设置开启pipeline
7. response响应对象的常用属性
    1. response.url：当前响应的url地址
    2. response.request.url：当前响应对应的请求的url地址
    3. response.headers：响应头
    4. response.requests.headers：当前响应的请求头
    5. response.body：响应体，也就是html代码，byte类型
    6. response.status：响应状态码