

# day08-课堂笔记

day08-课堂笔记

1. meta参数的使用
2. item的使用
3. 管道的使用
4. 课堂作业

## 1. meta参数的使用

如果在爬虫中，获取到了新的url地址，要将新的url地址构造成一个request对象，再进行返回。

```
1 yield scrapy.Request(url, callback, meta)
```

重点再给大家讲一下 meta 参数的使用。

meta参数：实现数据在不同的解析函数之间的传递

伪代码：

```
1 def parse():
2     # 相当于是我们在scrapy框架中代码中那个 parse方法
3     # 数据提取
4     item = {"title": "陕西工院“三全”用“心” 喜迎7895名2021级新同学", "time": "2021-09-08"}
5
6     item = parse_detail(item)
7     print(item)
8
9
10 def parse_detail(item):
11     # 解析详情页数据的方法
12     item["news_detail"] = "9月1日-2日，2021级7895名新生正式来校报到，我校整体谋划、统筹安排、重视细节，用“三全”举措做好2021级学生迎新工作。"
13     return item
14
15
16 parse()
```

完整代码：

```
1 import scrapy
2
3
4 class SgySpider(scrapy.Spider):
5     name = 'sgy'
6     allowed_domains = ['www.sxpi.edu.cn']
7     start_urls = ['https://www.sxpi.edu.cn/xwzx/gyyw.htm']
8
9     def parse(self, response):
10         # 分组
11         li_list = response.xpath("//div[@class='list_box']/ul/li")
```

```

12         for li in li_list:
13             item = {}
14             item["title"] = li.xpath("./a/text()").extract_first()
15             # 新闻的详情页的url地址,
16             # ../info/1020/16754.htm
17             # https://www.sxpi.edu.cn/info/1020/16860.htm
18             item["detail_href"] = "https://www.sxpi.edu.cn" +
li.xpath("./a/@href").extract_first()[2:]
19             item["time"] = li.xpath("./span/text()").extract_first()
20             # 获取详情页的数据 要对详情页的url地址发送请求, 构造新的请求对象
21             yield scrapy.Request(
22                 url=item["detail_href"],
23                 callback=self.parse_detail,
24                 meta={"news_list": item}
25             )
26
27     def parse_detail(self, response):
28         # 获取其他解析方法传递来的数据
29         item = response.meta["news_list"]
30
31         # 获取新闻的详情数据
32         item["news_content"] =
response.xpath("//div[@class='v_news_content']//text()").extract()
33         yield item

```

## 2. item的使用

将到构造新的request对象, 在爬虫中要返回给引擎的时候, yield关键字进行返回, yield在框架中返回数据的时候只能返回 request、dict、item对象、None

item对象指的是, 我们通过在Scrapy项目中有一个文件 `items.py` 文件中, 会去定义一个 `item` 类, 通过这个类创建出来的示例对象就是 item对象。

作用: 提前的去定义好我们要抓取的字段的名字 (指的就是之前使用字典的时候key), 为了防止手误, 写错key名。

```

1 item = {}
2 item['name'] = "张三"
3 现在使用的是字典, 字典中key名字是可以随便起。

```

使用Item, 提前的定义好要抓取的字段, 并且, 我们在对字段进行赋值的时候, Scrapy框架会自动的去检查你的字段有没有写错。

在 `items.py` 文件中去定义

```

1 import scrapy
2
3
4 class SchoolItem(scrapy.Item):
5     # 语法格式: 字段名 = scrapy.Field()
6     title = scrapy.Field()
7     detail_href = scrapy.Field()
8     time = scrapy.Field()
9     news_content = scrapy.Field()

```

使用: 在爬虫文件中去使用

```

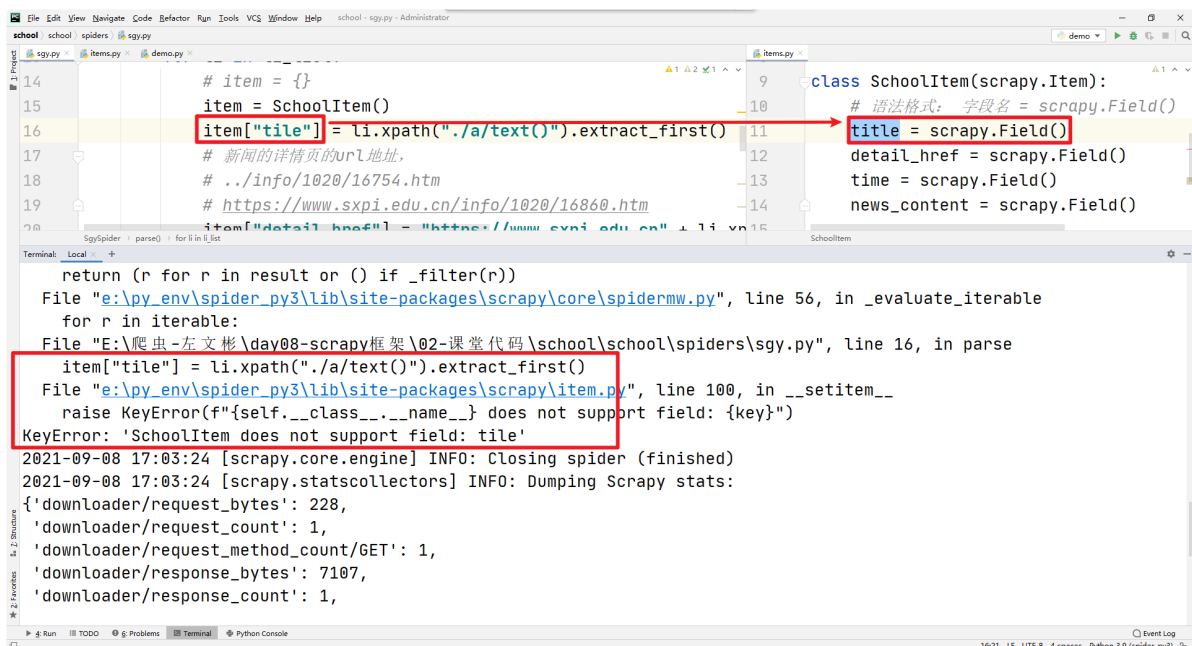
1 import scrapy
2 from school.items import SchoolItem
3

```

```

4
5 class SgySpider(scrapy.Spider):
6     name = 'sgy'
7     allowed_domains = ['www.sxpi.edu.cn']
8     start_urls = ['https://www.sxpi.edu.cn/xwzx/gyyw.htm']
9
10    def parse(self, response):
11        # 分组
12        li_list = response.xpath("//div[@class='list_box']/ul/li")
13        for li in li_list:
14            # item = {}
15            item = SchoolItem()
16            item["title"] = li.xpath("./a/text()").extract_first()
17            # 新闻的详情页的url地址,
18            # ../info/1020/16754.htm
19            # https://www.sxpi.edu.cn/info/1020/16860.htm
20            item["detail_href"] = "https://www.sxpi.edu.cn" +
li.xpath("./a/@href").extract_first()[2:]
21            item["time"] = li.xpath("./span/text()").extract_first()
22            # 获取详情页的数据 要对详情页的url地址发送请求, 构造新的请求对象
23            yield scrapy.Request(
24                url=item["detail_href"],
25                callback=self.parse_detail,
26                meta={"news_list": item}
27            )
28
29    def parse_detail(self, response):
30        # 获取其他解析方法传递来的数据
31        item = response.meta["news_list"]
32
33        # 获取新闻的详情数据
34        item["news_content"] =
response.xpath("//div[@class='v_news_content']//text()").extract()
35        print(item)
36        yield item

```



在使用 item对象的时候, 保存数据到MongoDB数据库中的时候, 注意: **需要将item对象先转换为dict, 之后再  
进行保存**

```

1 from pymongo import MongoClient
2 client = MongoClient()

```

```

3 collections = client.school98.news
4
5
6 class SchoolPipeline:
7     def process_item(self, item, spider):
8         # 因为 在爬虫中返回的数据是 item对象, 并不是一个 dict
9         # 所以在使用 insert_one()方法的时候, 报错
10        # 可以将item对象转换为一个字典之后, 在进行保存数据即可
11        # dict(item)
12        # 111 --> str(111) --> "111"
13        collections.insert_one(dict(item))
14        return item

```

同时实现翻页和详情页数据的抓取

```

1 import scrapy
2 from school.items import SchoolItem
3
4
5 class SgySpider(scrapy.Spider):
6     name = 'sgy'
7     allowed_domains = ['www.sxpi.edu.cn']
8     start_urls = ['https://www.sxpi.edu.cn/xwzx/gyyw.htm']
9
10    def parse(self, response):
11
12        print(f'当前获取数据的url地址是: {response.url}')
13
14        # 分组
15        li_list = response.xpath("//div[@class='list_box']/ul/li")
16        for li in li_list:
17            # item = {}
18            item = SchoolItem()
19            item["title"] = li.xpath("./a/text()").extract_first()
20            # 新闻的详情页的url地址,
21            # ../info/1020/16754.htm
22            # https://www.sxpi.edu.cn/info/1020/16860.htm
23            item["detail_href"] = "https://www.sxpi.edu.cn" +
18            li.xpath("./a/@href").extract_first()[2:]
24            item["time"] = li.xpath("./span/text()").extract_first()
25            # 获取详情页的数据 要对详情页的url地址发送请求, 构造新的请求对象
26            yield scrapy.Request(
27                url=item["detail_href"],
28                callback=self.parse_detail,
29                meta={"news_list": item}
30            )
31            # 获取一下 下一页的url地址
32            # next_url = response.xpath("//a[@class='Next']
33            [1]/@href").extract_first()
34            next_url = response.xpath("////a[text()='下页']/@href").extract_first()
35            # 将不完整的url地址拼接完整
36            if "gyyw" in next_url:
37                next_url = "https://www.sxpi.edu.cn/xwzx/" + next_url
38            else:
39                next_url = "https://www.sxpi.edu.cn/xwzx/gyyw/" + next_url
40            print(f'下一页的url地址是: {next_url}')
41            # 要将下一页的url地址构造成为一个新的request对象, 返回给引擎
42            yield scrapy.Request(
43                url=next_url,
44                callback=self.parse
45            )

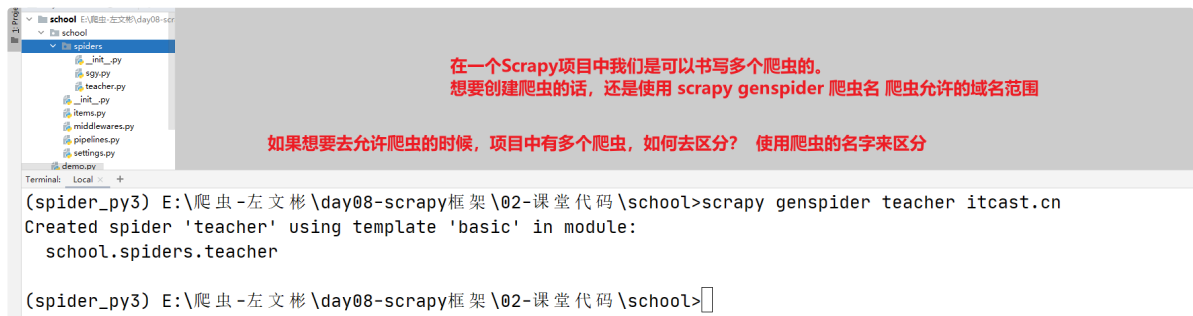
```

```

45
46     def parse_detail(self, response):
47         # 获取其他解析方法传递来的数据
48         item = response.meta["news_list"]
49
50         # 获取新闻的详情数据
51         item["news_content"] =
response.xpath("//div[@class='v_news_content']//text()).extract()
52         yield item

```

### 3. 管道的使用



在一个Scrapy项目中我们可以书写多个爬虫的，  
想要创建爬虫的话，还是使用 scrapy genspider 爬虫名 爬虫允许的域名范围

如果想要去允许爬虫的时候，项目中有多个爬虫，如何去区分？ 使用爬虫的名字来区分

```

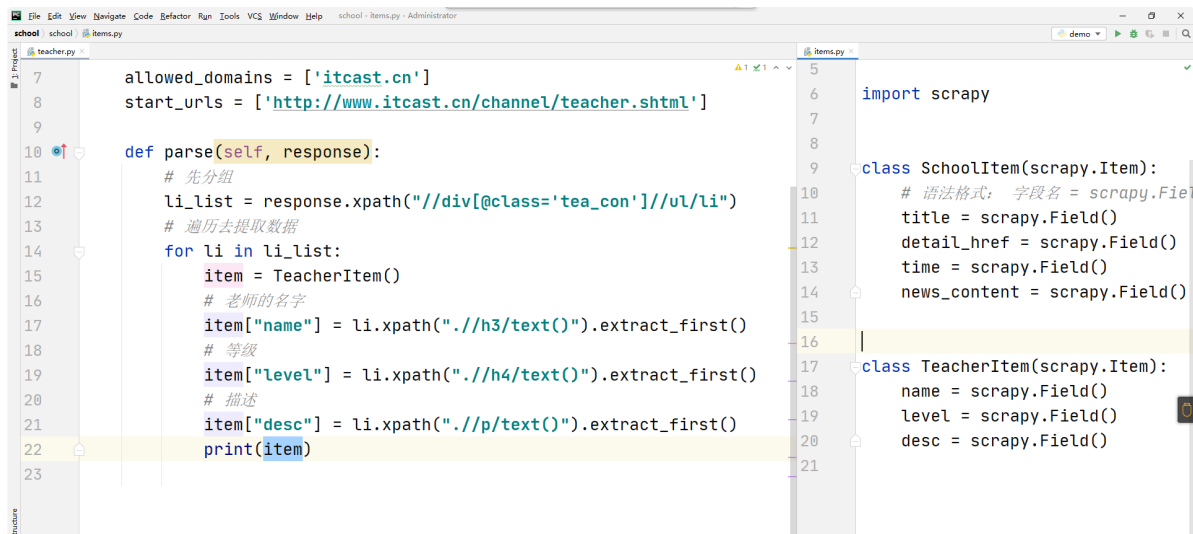
(spider_py3) E:\爬虫-左文彬\day08-scrappy框架\02-课堂代码\school>scrapy genspider teacher itcast.cn
Created spider 'teacher' using template 'basic' in module:
school.spiders.teacher

(spider_py3) E:\爬虫-左文彬\day08-scrappy框架\02-课堂代码\school>

```

讲解Scrapy框架的流程的时候，管道的作用：编写保存数据的代码。

基础的代码



```

7     allowed_domains = ['itcast.cn']
8     start_urls = ['http://www.itcast.cn/channel/teacher.shtml']
9
10    def parse(self, response):
11        # 先分组
12        li_list = response.xpath("//div[@class='tea_con']//ul/li")
13        # 遍历去提取数据
14        for li in li_list:
15            item = TeacherItem()
16            # 老师的名字
17            item["name"] = li.xpath("./h3/text()").extract_first()
18            # 等级
19            item["level"] = li.xpath("./h4/text()").extract_first()
20            # 描述
21            item["desc"] = li.xpath("./p/text()").extract_first()
22            print(item)
23

```

```

5
6 import scrapy
7
8
9 class SchoolItem(scrapy.Item):
10     # 语法格式： 字段名 = scrapy.Field()
11     title = scrapy.Field()
12     detail_href = scrapy.Field()
13     time = scrapy.Field()
14     news_content = scrapy.Field()
15
16
17 class TeacherItem(scrapy.Item):
18     name = scrapy.Field()
19     level = scrapy.Field()
20     desc = scrapy.Field()
21

```

在管道中除了有一个 `process_item()` 方法之外，还有另外两个方法

- `open_spider()` 在爬虫开始运行的时候只执行一次。
- `close_spider()` 在爬虫结束的时候只执行一次。

```

1 class TeacherPipeline(object):
2
3     def open_spider(self, spider):
4         """
5         此方法在爬虫开始运行的时候 只执行一次
6         一般在此方法中去实现一些初始化的操作： 1、 创建数据库连接 2. 打开文件
7         :param spider: 当前运行的爬虫对象
8         """
9         print("爬虫开启的时候只执行一次")
10        self.collections = client.itcast.teacher
11
12    def process_item(self, item, spider):
13        # 需求 将老师的信息保存到MongoDB数据库: itcast 集合 teacher
14        self.collections = client.itcast.taacher

```

```

15         self.collections.insert_one(dict(item))
16         return item
17
18     def close_spider(self, spider):
19         """
20         此方法是在爬虫关闭的时候只执行一次，一般用于销毁数据。
21         关闭文件对象  关闭数据库连接
22         :param spider: 当前运行爬虫对象
23         :return:
24         """
25         print("爬虫结束的时候只执行一次。。。。")

```

多爬虫，多管道的时候操作

```

1  from pymongo import MongoClient
2
3  client = MongoClient()
4
5
6  class SchoolPipeline:
7      def open_spider(self, spider):
8          """
9          此方法在爬虫开始运行的时候 只执行一次
10         一般在此方法中去实现一些初始化的操作： 1、 创建数据库连接 2. 打开文件
11         :param spider: 当前运行的爬虫对象
12         """
13         print("爬虫开启的时候只执行一次")
14         if spider.name == "sgy":
15             self.collections = client.school98.news
16
17     def process_item(self, item, spider):
18         # 因为 在爬虫中返回的数据是 item对象，并不是一个 dict
19         # 所以在使用 insert_one()方法的时候，报错
20         # 可以将item对象转换为一个字典之后，在进行保存数据即可
21         # dict(item)
22         # 111 --> str(111) --> "111"
23         if spider.name == "sgy":
24             self.collections.insert_one(dict(item))
25         return item
26
27
28  class TeacherPipeline(object):
29
30     def open_spider(self, spider):
31         """
32         此方法在爬虫开始运行的时候 只执行一次
33         一般在此方法中去实现一些初始化的操作： 1、 创建数据库连接 2. 打开文件
34         :param spider: 当前运行的爬虫对象
35         """
36         print("爬虫开启的时候只执行一次")
37         if spider.name == "teacher":
38             self.collections = client.itcast.teacher
39
40     def process_item(self, item, spider):
41         # 需求 将老师的信息保存到MongoDB数据库： itcast 集合 teacher
42         if spider.name == "teacher":
43             self.collections.insert_one(dict(item))
44         return item
45
46     def close_spider(self, spider):
47         """

```

```
48         此方法是在爬虫关闭的时候只执行一次，一般用于销毁数据。  
49         关闭文件对象    关闭数据库连接  
50         :param spider: 当前运行爬虫对象  
51         :return:  
52         ""  
53         print("爬虫结束的时候只执行一次。。。。")
```

## 4. 课堂作业

- 要求使用 scrapy实现
- 要抓取的数据地址：<http://www.shaanxi.gov.cn/hd/wxszsjh/lyxd/index.html>
- 要抓取的字段信息（必须使用 item 对象来完成存储，不要使用字典）
  - 信件标题
  - 回复单位
  - 处理时间
  - 状态
  - 详情页url地址
  - 提问的具体内容
  - 回答的的具体内容
- 翻页 18 页 抓取到18页所有数据。
- 将数据保存到MongoDB数据库，数据库名字 sxgov 集合的名字： datas