day06-课堂笔记

1. PyMongo使用

1.1 pymongo的介绍和安装

刚才给大家回顾的一些MongoDB的操作的命令，都是在 终端中进行的操作，但是，后续我们是要书写爬虫程序去抓取数据，抓取到的数据是在Python程序中，最终要将程序中的数据保存到MongoDB数据库。

Python程序 和 MongoDB 数据库，两个不同的东西。

如何在Python代码中去操作MongoDB数据库。

需要借助一个模块，这个模块就是我们要学习的Pymongo模块。这个模块就可以帮助我们在Python程序中去操作MongoDB数据库。

安装：

```
pip install pymongo
```

```
(spider_py3) E:\爬虫-左文彬\day06-MongoDB数据库和Scrapy入门\02-课堂代码>pip install pymongo
Collecting pymongo
  Downloading pymongo-3.12.0-cp39-cp39-win_amd64.whl (397 kB)
     |████████████████████████████████| 397 kB 1.1 MB/s
Installing collected packages: pymongo
Successfully installed pymongo-3.12.0

(spider_py3) E:\爬虫-左文彬\day06-MongoDB数据库和Scrapy入门\02-课堂代码>pip list
Package            Version
------------------ ---------
certifi            2021.5.30
charset-normalizer 2.0.4
idna               3.2
lxml               4.6.3
pip                21.2.4
pymongo            3.12.0
requests           2.26.0
selenium           3.141.0
setuptools         57.4.0
urllib3            1.26.6

(spider_py3) E:\爬虫-左文彬\day06-MongoDB数据库和Scrapy入门\02-课堂代码>
```

1.2 pymongo的使用

使用步骤:

- 导包
- 创建客户端的连接对象。
- 创建要操作的集合对象。
- 使用集合对象对数据进行操作
    - 插入数据（用的最多的，爬虫最终获取到的数据，是要保存到MongoDB，保存的意思的就是插入数据）
    - 查询

插入数据

```
# - 导包
from pymongo import MongoClient

# - 创建客户端的连接对象。
client = MongoClient()

# - 创建要操作的集合对象。   客户端 --> 数据库 --> 集合 --> 数据
# 方式一：  客户端对象[要操作数据库名][要操作集合名]
# collection = client['spider_96']['student']
# 方式二 ：  客户端对象.要操作的数据库名.要操作的集合名
collection = client.spider_96.student

# - 使用集合对象对数据进行操作
#    - 插入数据（用的最多的，爬虫最终获取到的数据，是要保存到MongoDB，保存的意思的就是插入数据）
# 插入数据 集合对象.insert(字典/列表)
# 插入一条数据
# collection.insert({"name": "李四", "age": 18, "gender": True})
# collection.insert_one({"name": "王五", "age": 19, "gender": False})

# 插入多条数据    insert([{},{},{},......])
# 准备多条数据
stu_list = [
    {"name": "郭靖", "age": 28, "gender": True},
    {"name": "黄蓉", "age": 20, "gender": False},
    {"name": "华筝", "age": 18, "gender": False},
    {"name": "欧阳锋", "age": 58, "gender": True}
]
# 一次性插入多条数据
# collection.insert(stu_list)
collection.insert_many(stu_list)
```

查询数据

```
# - 导包
from pymongo import MongoClient

# - 创建客户端的连接对象。
client = MongoClient()
```

```python

# - 创建要操作的集合对象。  客户端 --> 数据库 --> 集合 --> 数据
# 方式一：  客户端对象[要操作数据库名][要操作集合名]
# collection = client['spider_96']['student']
# 方式二 ：  客户端对象.要操作的数据库名.要操作的集合名
collection = client.spider_96.student

# - 使用集合对象对数据进行操作
#    - 插入数据（用的最多的，爬虫最终获取到的数据，是要保存到MongoDB，保存的意思的就是插入数据）
# 插入数据 集合对象.insert(字典/列表)
# 插入一条数据
# collection.insert({"name": "李四", "age": 18, "gender": True})
# collection.insert_one({"name": "王五", "age": 19, "gender": False})

# 插入多条数据   insert([{},{},{},.....])
# 准备多条数据
stu_list = [
    {"name": "郭靖", "age": 28, "gender": True},
    {"name": "黄蓉", "age": 20, "gender": False},
    {"name": "华筝", "age": 18, "gender": False},
    {"name": "欧阳锋", "age": 58, "gender": True}
]
# 一次性插入多条数据
# collection.insert(stu_list)
# collection.insert_many(stu_list)

#    - 查询
# 查询一个， 集合中的第一条数据   集合名.find_one()
# 查询到的数据是一个 字典类型。  {'_id': ObjectId('61359b58a19c64b3c465bb24'), 'name': '张三', 'age': 18, 'gender': True} <class 'dict'>
stu = collection.find_one()
# print(stu, type(stu))

# 查询多条  集合名.find()  返回一个 cursor 游标对象
# 返回的是一个Cursor 游标对象 <pymongo.cursor.Cursor object at 0x0000020B3B4750A0>
stus = collection.find()
# 1. 可以直接去遍历这个游标对象，每遍历一次 就是一个 字典
# for s in stus:
#     print(f"当前的学生信息是: {s},  数据的类型:  {type(s)}")

# 2. 可以使用 list(cursor) 就可以将 cursor游标对象 转换为一个Python的列表
student_list = list(stus)
print(student_list)
```

1.3 总结

- insert(): 插入的是一个字典，插入一条数据，如果插入的是一个列表[{},{},{}] 表示插入多条数据
- inser_one(): 只能插入一条数据，接收的是一个字典。
- insert_many(): 插入多条数据，接收的是一个列表[{},{},{}]
- find_one(): 查询出集合中的第一条数据，返回的是一个字典。
- find(): 查询的是集合中所有的数据，返回的是一个 cursor 游标对象，可以将游标对象 转换为 list(cursor) 转成列表，遍历这个列表。

2. 案例

需求： 抓取的url地址： https://wz.sun0769.com/political/index/politicsNewest

- 要抓取的字段
-

编号	状态	问政标题	响应时间	问政时间

- 要求抓取 10页的数据。
- 抓取到的数据要 [{},{}]. 最终保存到MongoDB数据库中， 数据库的名字sun， 集合的名字是 datas

```python
import requests
from lxml import etree
from pymongo import MongoClient

# 1. 准备url地址
url = "https://wz.sun0769.com/political/index/politicsNewest"
# 准备请求头
headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36"
}
# 创建数据库连接对象
client = MongoClient()
# 创建要操作的集合对象
collections = client.sun.datas


# 2. 发送请求 获取响应
response = requests.get(url, headers=headers)

# 3. 提取数据 就是从html中提取数据的
# 将html转换为element对象
element = etree.HTML(response.content.decode())
# 先分组  再提取
li_list = element.xpath("//ul[@class='title-state-ul']/li")
# 准备一个空列表，用于存放数据
data_list = []
for li in li_list:
    # 准备一个字典， 一个字典就是一条数据
    item = {}
    # 获取标号
    item['index'] = li.xpath("./span[1]/text()")[0]
    # 状态
    item['status'] = li.xpath("./span[2]/text()")[0].strip()
    # 标题
    item['title'] = li.xpath("./span[3]/a/text()")[0]
    # 响应时间
    item['response_time'] = li.xpath("./span[4]/text()")[0].strip()
    # 发布时间
    item['publish_time'] = li.xpath("./span[5]/text()")[0]
    data_list.append(item)

# 4、 保存数据
# 插入数据，多条数据
collections.insert_many(data_list)
```

抓取多页的数据

```python
import time

import requests
from lxml import etree
from pymongo import MongoClient

# 1. 准备url地址
url_temp = "https://wz.sun0769.com/political/index/politicsNewest?id=1&page={}"
# 先将要抓取的10个url地址 准备好  可以将10个URL地址放到一个列表中
url_list = []
for i in range(1, 11):
    page_url = url_temp.format(i)
    url_list.append(page_url)

# 准备请求头
headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36"
}
# 创建数据库连接对象
client = MongoClient()
# 创建要操作的集合对象
collections = client.sun.datas

for url in url_list:
    time.sleep(2)
    print(f'正在抓取的页面是： {url}')
    # 2. 发送请求 获取响应
    response = requests.get(url, headers=headers)

    # 3. 提取数据 就是从html中提取数据的
    # 将html转换为element对象
    element = etree.HTML(response.content.decode())
    # 先分组  再提取
    li_list = element.xpath("//ul[@class='title-state-ul']/li")
    # 准备一个空列表，用于存放数据
    data_list = []
    for li in li_list:
        # 准备一个字典， 一个字典就是一条数据
        item = {}
        # 获取标号
        item['index'] = li.xpath("./span[1]/text()")[0]
        # 状态
        item['status'] = li.xpath("./span[2]/text()")[0].strip()
        # 标题
        item['title'] = li.xpath("./span[3]/a/text()")[0]
        # 响应时间
        item['response_time'] = li.xpath("./span[4]/text()")[0].strip()
        # 发布时间
        item['publish_time'] = li.xpath("./span[5]/text()")[0]
        data_list.append(item)

    # 4、 保存数据
```

```python
    # 插入数据，多条数据
    collections.insert_many(data_list)
    print(f"{url}  中的数据保存完毕..............")
```

3. Scrapy框架

2.1 Scrapy框架的介绍与安装

是在Python中写爬虫的时候，使用的最出名的框架，使用的纯Python实现的开源的爬虫框架。

框架中已经集成 异步操作。框架在抓取数据的时候，速度是特别快的。

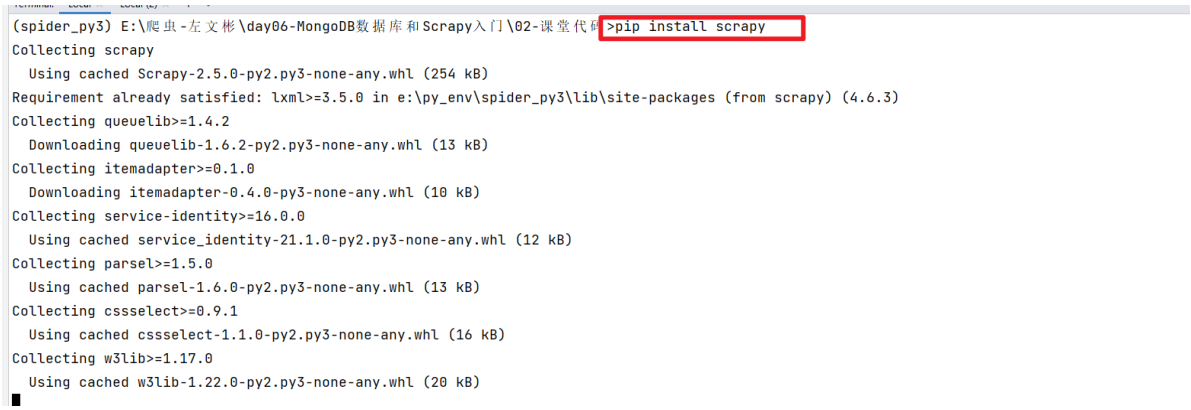
现在 有 100 个url地址， 使用同步 假设一个url地址请求 需要1秒， 将所有的url地址请求完毕 至少需要 100秒

如果现在使用的是异步： 在一秒钟可以发送 10 次请求， 10秒。

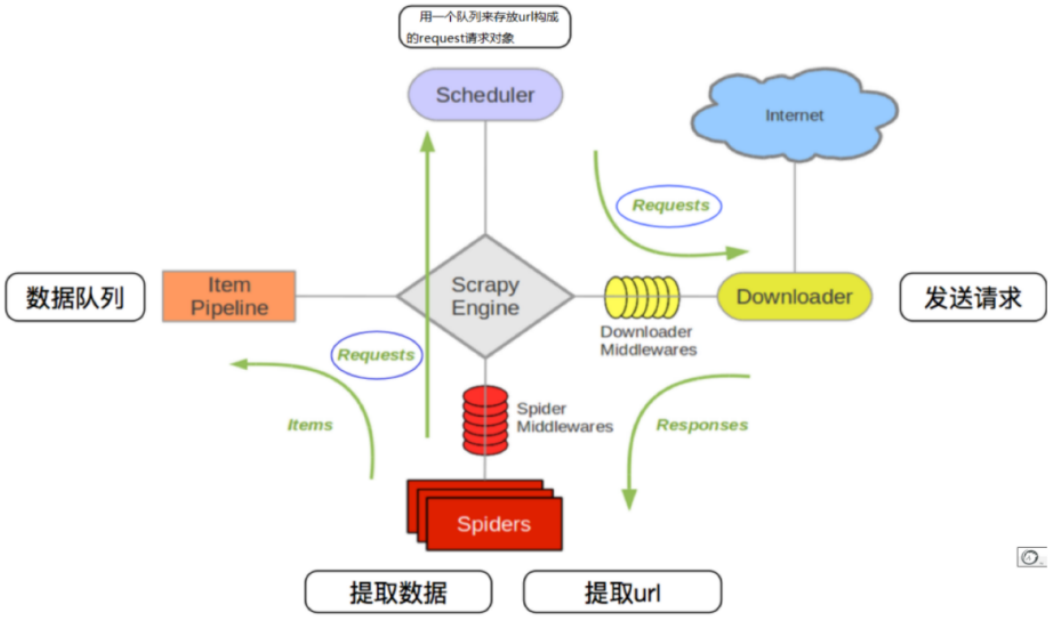
异步： twisted 异步框架来实现的，

安装：

```
pip install scrapy
```



2.2 Scrapy框架中每个模块的概念和作用



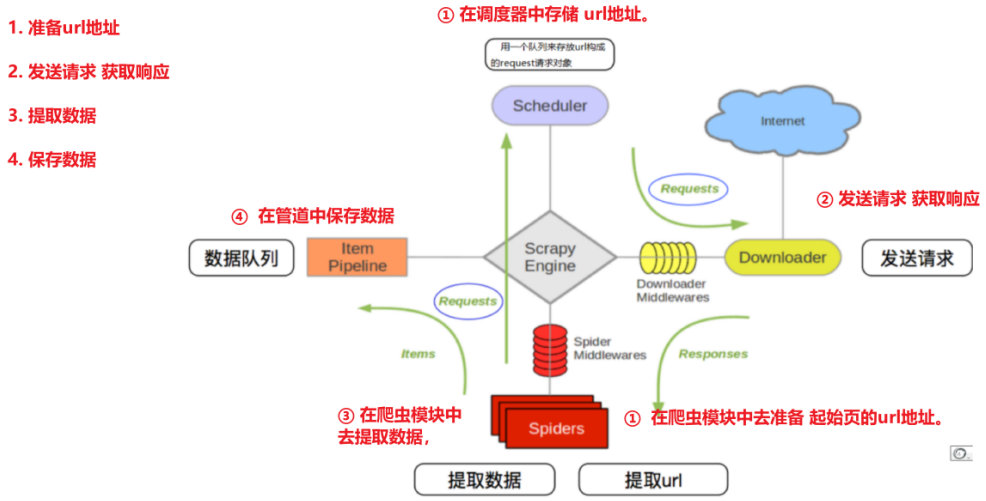
Scheduler： 调度器，用于存储 request 对象的 队列（容器）。

Downloader： 下载器，发送请求，获取响应的。

Spider： 爬虫模块，用于 书写爬虫的代码，数据的解析，提取新的url地址。

item pipeline： 数据管道，用于存储数据的。（并不是在这个地方去存储，而是在管道中去表写存储数据的代码-文件-数据库。）

Scrapy Engine： Scrapy的引擎，负责每个模块之间的调度。



Scheduler： 调度器，用于存储 url 地址的 队列（容器）。

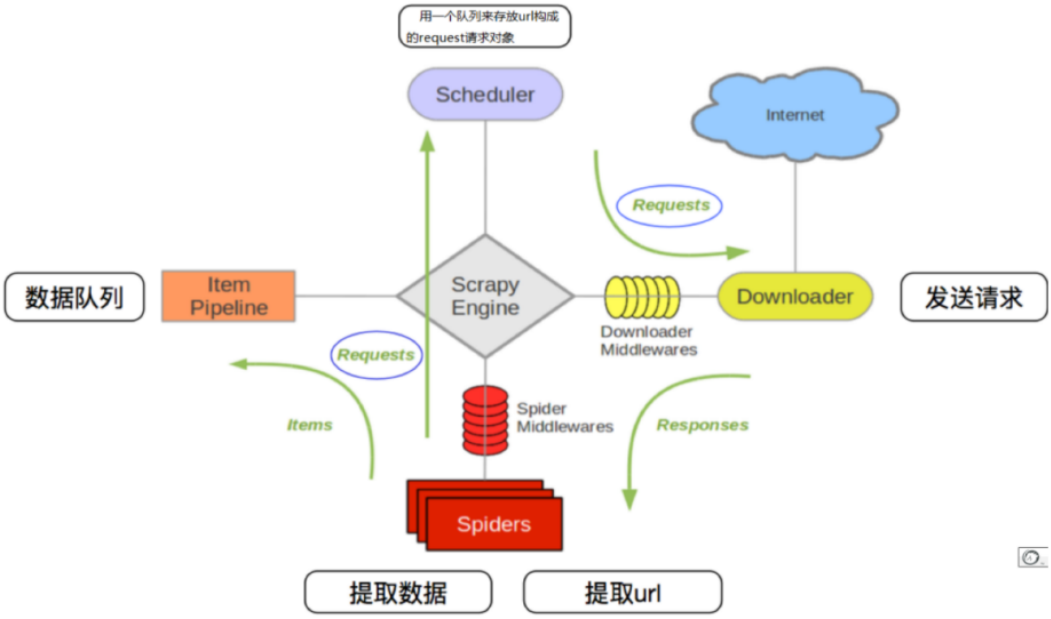
Downloader： 下载器，发送请求，获取响应的。

Spider： 爬虫模块，用于 书写爬虫的代码，数据的解析，提取新的url地址。

item pipeline： 数据管道，用于存储数据的。（并不是在这个地方去存储，而是在管道中去表写存储数据的代码-文件-数据库。）

Scrapy Engine： Scrapy的引擎，负责每个模块之间的调度。

2.3 Scrapy框架的流程-[重要]

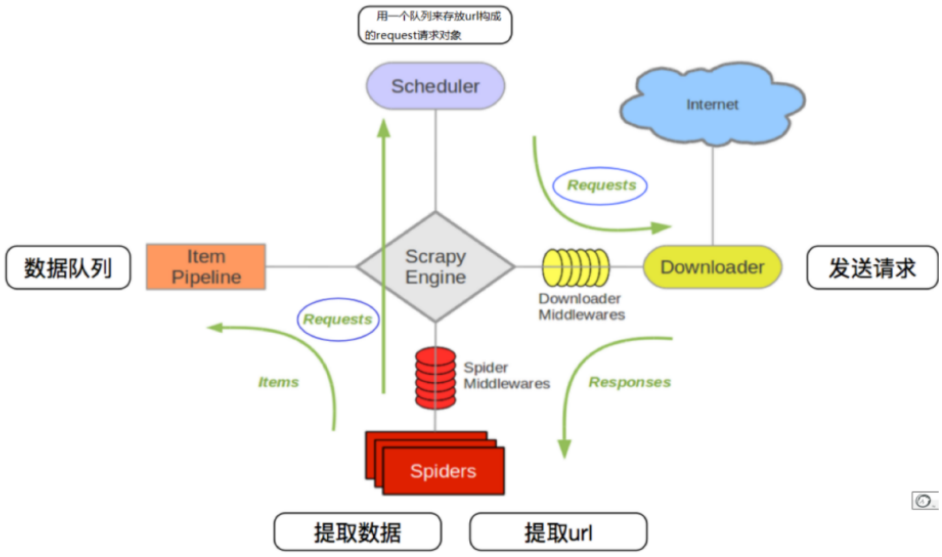


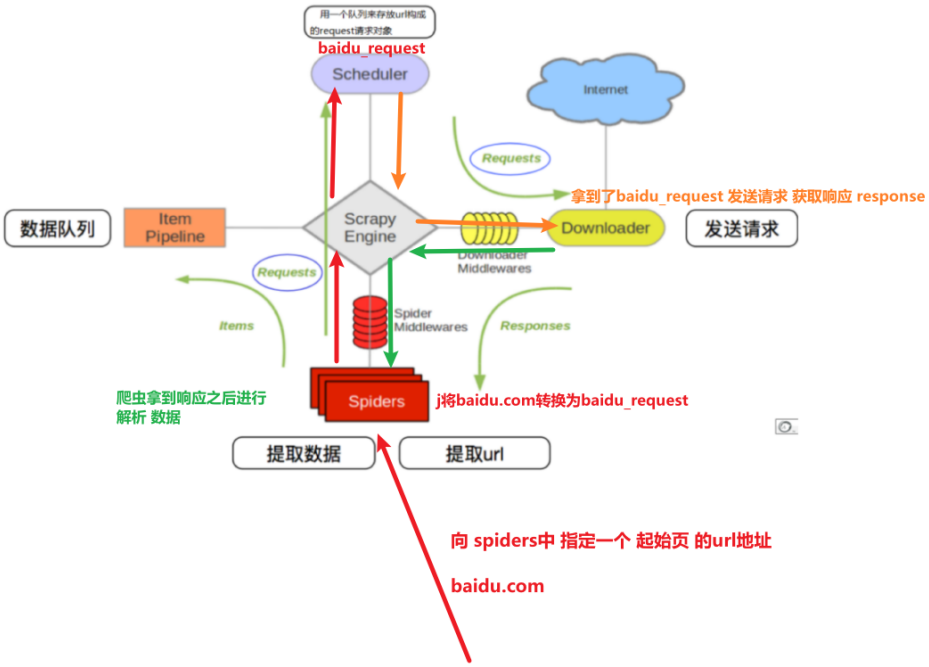
在上图中看到了每个模块之间都有一条直线进行连接，但是真实的情况是，每个模块之间并没有之间的连线，

在框架中 每个模块是 独立。但是，可以通过引擎 将每个独立的模块的串联起来。

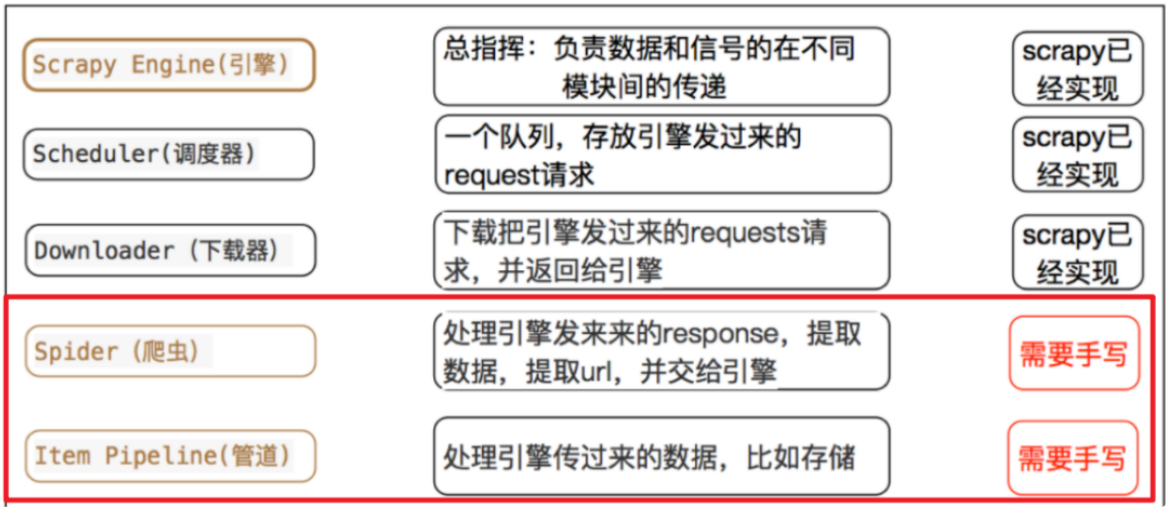
步骤：

- 从爬虫中开始， 在Spider模块中要去指定一个 起始页 的url地址 `http://www.baidu.com`， 在Spider模块中，会将 url地址 转换 request对象。
- 在爬虫中获取到了 request对象之后， 将 request对象返回给 引擎，引擎再讲request对象给了 调度器。 （1. 准备url地址）
- 将 调度器中的 request对象取出来，给了 引擎，引擎再将 request对象 交给 下载器 去发送请求 获取响应，那在下载器中 获取到一个 response对象。 （2. 发送请求获取响应）
- 要将下载器中获取到的 response对象 先交给 引擎 ，引擎再将 response对象 交给 spider。
- 在spider中去解析数据（3. 解析数据）
    - 解析出来的是数据： 解析出来数据之后要进行保存，要将解析出来的数据 先给了 引擎 ，引擎再将数据给了 管道（4. 保存数据）
    - 解析出来的是新的url地址（下一页）： 将新的url地址，转换为 request对象，---引擎 --- 调度器 -- 引擎--下载器--爬虫





- spider ---- 引擎 --- 调度器 ---引擎 --- 下载器 --- 引擎 ----spider ---- 引擎 ---item pipeline

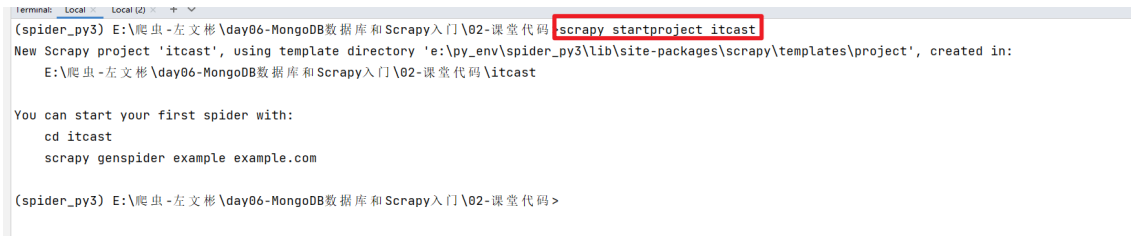


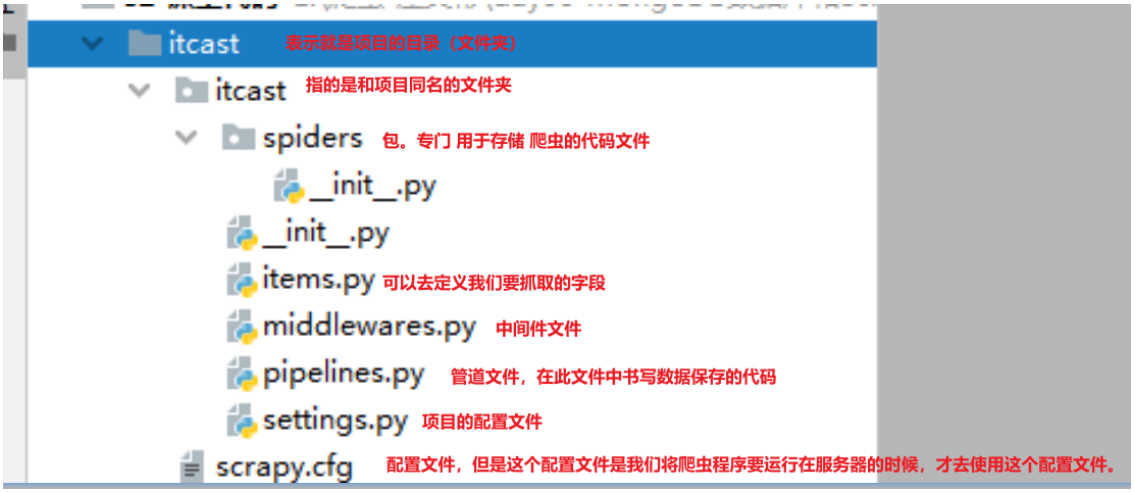
2.4 Scrapy框架的使用

- 创建项目

```
scrapy startproject 项目名

scrapy startproject itcast
```





- 创建爬虫

```
先切换到项目的路径下
cd 项目名
再创建项目
scrapy genspider 爬虫名 爬虫允许的域名范围
```

默认生成的代码 解析：

```
# 导入了scrapy
import scrapy


# 定义了一个爬虫类，继承了父类， scrapy.Spider
# scrapy.Spider 是 scrapy爬虫的基类，在scrapy中所有的爬虫必须要继承这个类
class TeacherSpider(scrapy.Spider):
    # 类属性， 爬虫的名字，我们在运行这个爬虫的时候，会使用到这个爬虫名字    可以去改
    name = 'teacher'
    # 爬虫允许的域名范围，列表，也就是说我们可以指定多个范围。 可以自己改
    allowed_domains = ['itcast.cn']
    # 起始页的url地址，列表， 起始页的url地址可以有多个。   必须是可以去改的
    start_urls = ['http://www.baidu.com/']

    # 这个方法 是专门用于去解析 起始页url地址的响应的方法， 其中的 response 参数，就是起始页的响应对象
    # 注意： 此方法必须有，并且 此方法的名字 必须叫做 parse
    def parse(self, response):
        pass
```

- 书写爬虫代码

```
# 导入了scrapy
import scrapy


# 定义了一个爬虫类，继承了父类， scrapy.Spider
# scrapy.Spider 是 scrapy爬虫的基类，在scrapy中所有的爬虫必须要继承这个类
class TeacherSpider(scrapy.Spider):
    # 类属性， 爬虫的名字，我们在运行这个爬虫的时候，会使用到这个爬虫名字    可以去改
    name = 'teacher'
    # 爬虫允许的域名范围，列表，也就是说我们可以指定多个范围。 可以自己改
    allowed_domains = ['itcast.cn']
    # 起始页的url地址，列表， 起始页的url地址可以有多个。   必须是可以去改的
    start_urls = ['http://www.itcast.cn/channel/teacher.shtml']

    # 这个方法 是专门用于去解析 起始页url地址的响应的方法， 其中的 response 参数，就是起始页的响应对象
    # 注意： 此方法必须有，并且 此方法的名字 必须叫做 parse
    def parse(self, response):

        print('爬虫允许起来了.....执行了这里.....')
        # 打印响应的html 等同于 前面学习的requests模块中的 response.content.decode()
```

```
        # print(response.body.decode())
        # 也是可以使用xpath方法来对数据进行解析
        # 在scrapy框架中想要使用xpath来解析数据直接使用  : response.xpath()
        # 先分组  再提取
        li_list = response.xpath("//div[@class='tea_con']//ul/li")
        for li in li_list:
            # 准备一个字典  来保存数据
            item = {}
            # 想要获取的数据  应该是  数据的字符串  这个方法extract_first() 就是去获取数据的文本字符串内容
            # extract()  获取到的是一个列表  列表中包含了文本的字符串
            item["name"] = li.xpath(".//h3/text()").extract_first()
            item["level"] = li.xpath(".//h4/text()").extract_first()
            item["desc"] = li.xpath(".//p/text()").extract_first()
            print(item)
```

- 保存数据
- 启动爬虫

```
scrapy crawl 爬虫名
```

作业

抓取的url地址： https://www.sxpi.edu.cn/xwzx/gyyw.htm

抓取的数据内容： 新闻的标题 以及发布时间，

要求：

- 使用 requests模块 和 xpath 完成，（如果有能力的话，再使用 scrapy框架去实现一遍）
- `[{}, {},{} ]`
- 需要将数据保存到MongoDB数据库。（截个图，显示下 当前电脑的时间， 还要在截图上写上自己的名字）
- 需要抓取 10 页的数据。