爬虫—scrapy爬虫框架

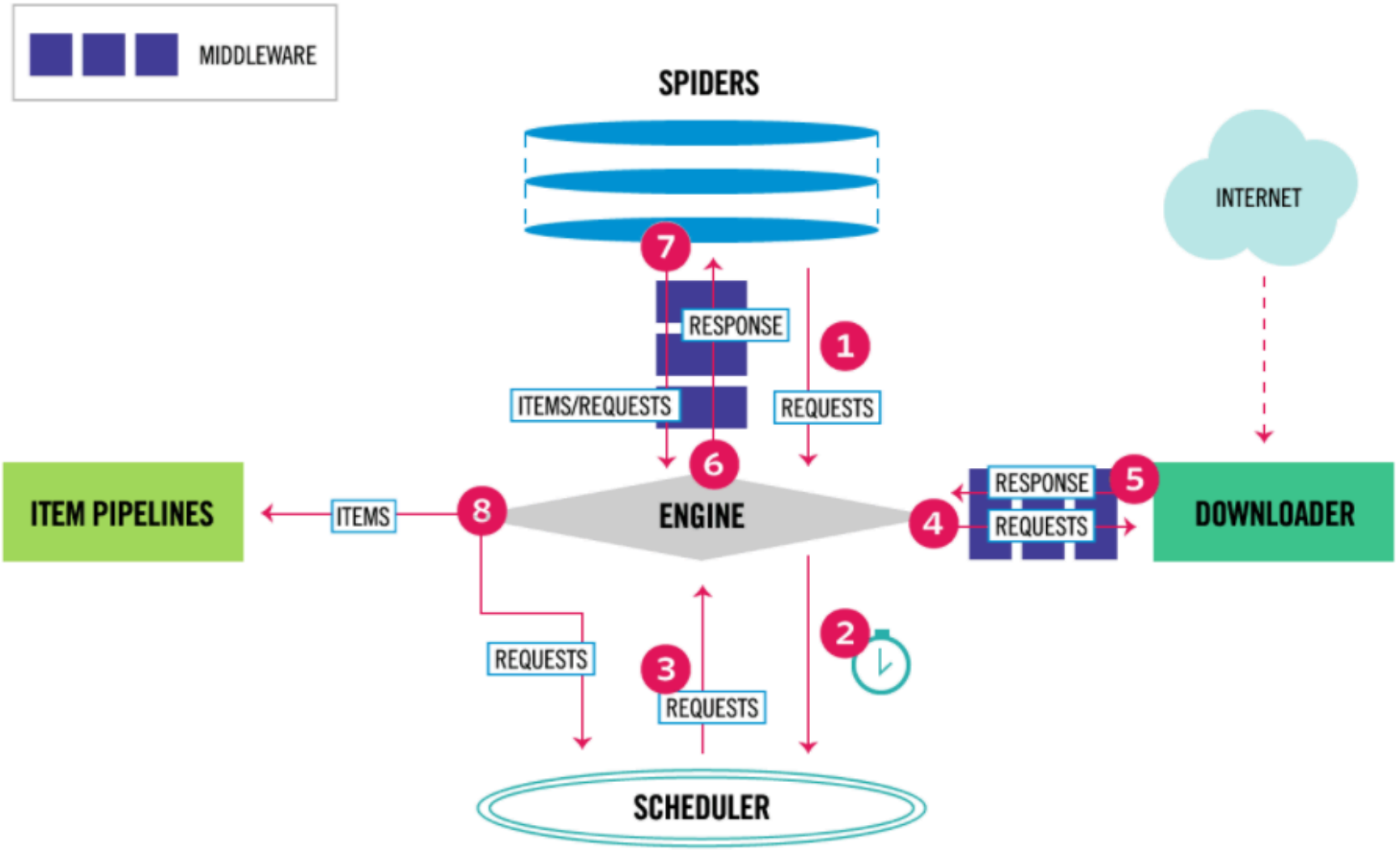
一、简介

1、基本功能

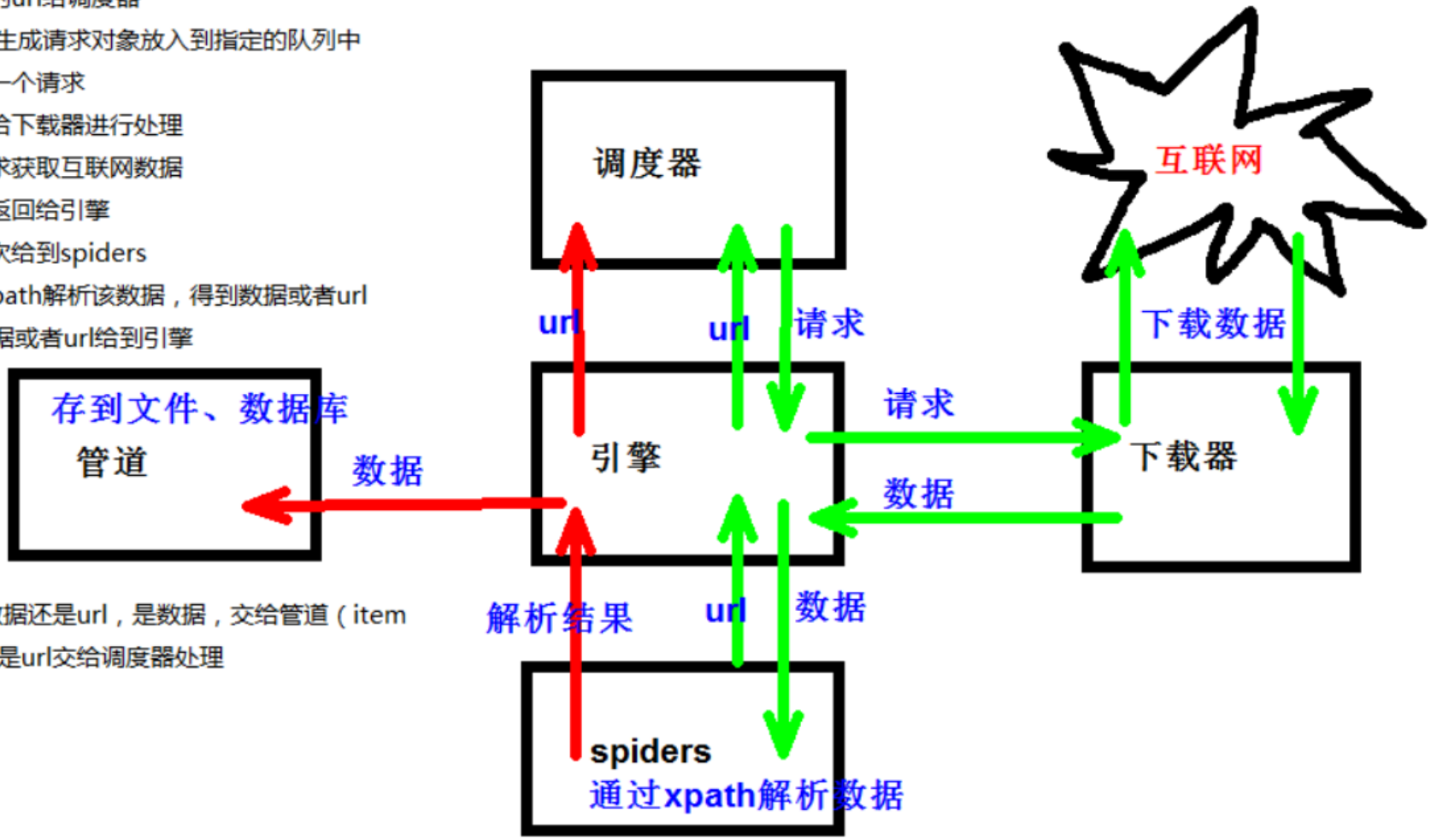
Scrapy是一个适用爬取网站数据、提取结构性数据的应用程序框架，它可以应用在广泛领域：Scrapy 常应用在包括数据挖掘，信息处理或存储历史数据等一系列的程序中。通常我们可以很简单的通过 Scrapy 框架实现一个爬虫，抓取指定网站的内容或图片。

2、架构

- ***Scrapy Engine(引擎)***：负责Spider、ItemPipeline、Downloader、Scheduler中间的通讯，信号、数据传递等。
- ***Scheduler(调度器)***：它负责接受引擎发送过来的Request请求，并按照一定的方式进行整理排列，入队，当引擎需要时，交还给引擎。
- ***Downloader（下载器）***：负责下载Scrapy Engine(引擎)发送的所有Requests请求，并将其获取到的Responses交还给Scrapy Engine(引擎)，由引擎交给Spider来处理。
- ***Spider（爬虫）***：它负责处理所有Responses,从中分析提取数据，获取Item字段需要的数据，并将需要跟进的URL提交给引擎，再次进入Scheduler(调度器)。
- ***Item Pipeline(管道)***：它负责处理Spider中获取到的Item，并进行进行后期处理（详细分析、过滤、存储等）的地方。
- ***Downloader Middlewares（下载中间件）***：一个可以自定义扩展下载功能的组件。
- ***Spider Middlewares（Spider中间件）***：一个可以自定扩展和操作引擎和Spider中间通信的功能组件。


MIDDLEWARE
SPIDERS
INTERNET
7
RESPONSE
1
ITEMS/REQUESTS
REQUESTS
6
ITEM PIPELINES
ITEMS
8
ENGINE
RESPONSE
5
4
REQUESTS
DOWNLOADER
REQUESTS
3
REQUESTS
2
SCHEDULER

1、引擎向spiders要url
2、引擎将要爬取的url给调度器
3、调度器会将url生成请求对象放入到指定的队列中
4、从队列中出队一个请求
5、引擎将请求交给下载器进行处理
6、下载器发送请求获取互联网数据
7、下载器将数据返回给引擎
8、引擎将数据再次给到spiders
9、spiders通过xpath解析该数据，得到数据或者url
10、spiders将数据或者url给到引擎



11、引擎判断该数据还是url，是数据，交给管道（item pipeline）处理，是url交给调度器处理

3、scrapy项目的结构

```
项目名字
    项目的名字
        spiders文件夹（存储的是爬虫文件）

```

```
    init
    自定义的爬虫文件          核心功能文件
init
items           定义数据结构的地方 爬虫的数据都包含哪些
middleware      中间件 代理
pipelines       管道   用来处理下载的数据
settings        配置文件 robots协议 ua定义等
```

二、scrapy环境搭建

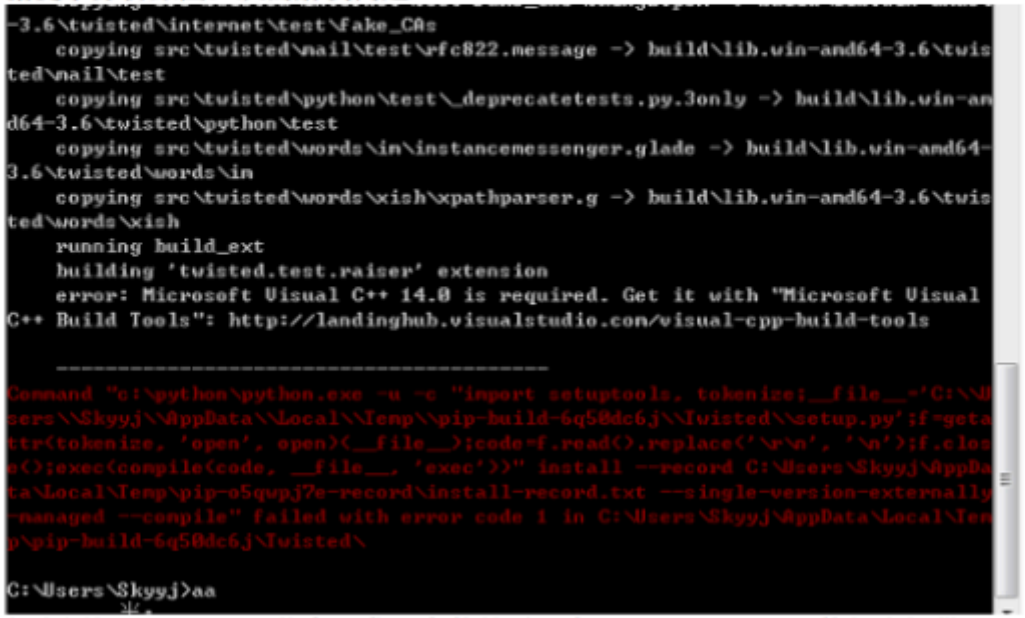
Scrapy 框架环境搭建

Scrapy 安装介绍

使用 pip 来安装 Scrapy，在命令行窗口执行如下命令即可：

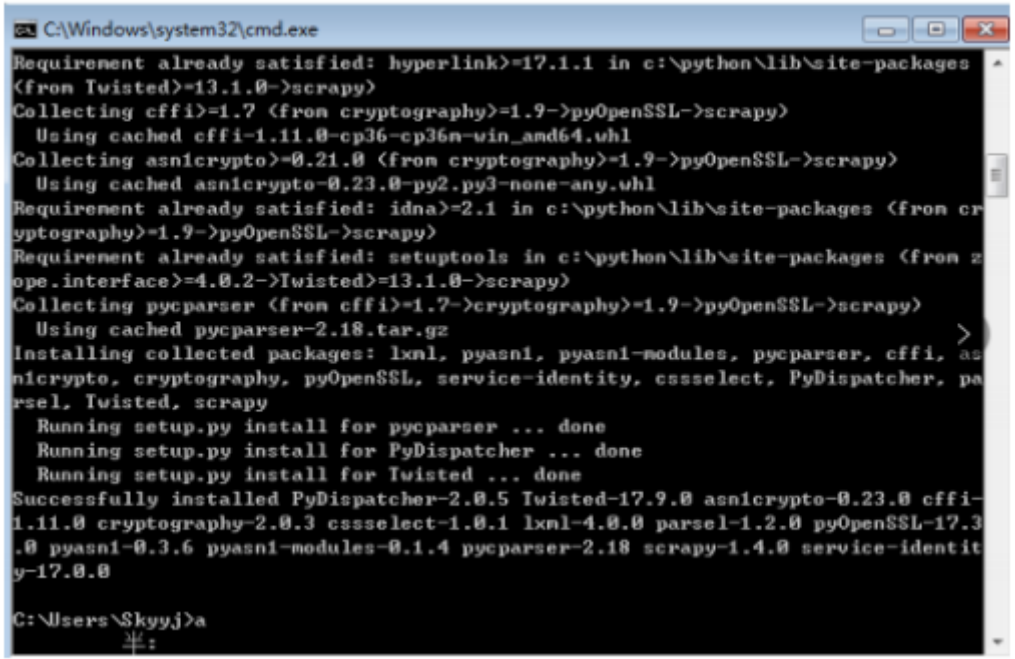
```
pip install Scrapy
```

安装过程可能会因为环境等问题出现终止，此时可以查看命令行显示提示信息，根据提示分析原因，从而解决问题。



如上图提示：error：信息，发现安装前需要有 Visual C++，而且他们也提供了下载地址，可以复制粘贴到地址栏，进行下载，并安装。

再次安装，如下图即可成功



三、如何开始

1、新建项目： 新建一个新的爬虫项目

```
打开cmd，输入scrapy startproject 项目的名字
(默认是在C:\Users\...这个目录下，你可以自行切换到对应的 文件下）
注意：项目的名字不允许使用数字开头 也不能包含中文
```

2、明确目标 （items.py）： 明确你想要抓取的目标

```
选择你需要爬取的内容，例如作者名字、小说名、封面图片等
在items.py文件中定义
```

```
import scrapy
class AdicrawlerItem(scrapy.Item):
    author = scrapy.Field()
    theme = scrapy.Field()
    # 以上定义了两个变量 分别是作者名、主题。
```

3、制作爬虫 （spiders/xxspider.py）：制作爬虫开始爬取网页

创建爬虫文件

```
        要在spiders文件在去创建爬虫文件
            cd 项目的名字\项目的名字\spiders
            eg : cd scrapy_baidu\scrapy_baidu\spiders

    创建爬虫文件
        scrapy genspider 爬虫文件的名字 要爬的网页
        eg : scrapy genspider baidu www.baidu.com
        一般情况下不需要添加http协议
        因为start_urls的值是根据allowed_domains修改的
```

爬虫文件的解释：

```
import scrapy
class BaiduSpider(scrapy.Spider):
    # 爬虫的名字 一般运行爬虫的时候 使用的值
    name = 'baidu'
    # 允许访问的域名
    allowed_domains = ['www.baidu.com']
    # 起始的url地址   指的是第一次要访问的域名

```

```
    # start_urls    是在allowed_domains的前面添加一个http://
    #               是在allowed_domains的后面添加一个/
    # 如果以html结尾 就不用加/ 否则网站进不去  报错
    start_urls = ['http://www.baidu.com/']

    # 是执行了start_urls之后  执行的方法
    # 方法中的response  就是返回的那个对象
    # 相当于 response = urllib.request.urlopen()
    #       response = requests.get()
    def parse(self, response):
        pass
```

response的属性和方法

- response.text
  获取的是响应的字符串
- response.body
  获取的是二进制数据
- response.xpath
  可以直接使用xpath方法来解析response中的内容
- response.extract()
  提取seletor对象的data属性值
- response.extract_first()
  提取的seletor列表的第一个数据

```
import scrapy
from AdiCrawler.items import AdicrawlerItem


```

```
class ThousandpicSpider(scrapy.Spider):
    name = 'thousandpic'
    allowed_domains = ['www.58pic.com']
    start_urls = ['http://www.58pic.com/c/']

    def parse(self, response):
        author = response.xpath('//div[@class="wrap-list fl"]//span[@class="fl info-h1"]/text()').extract()
        theme = response.xpath('//div[@class="wrap-list fl"]//span[@class="usernameColor"]/text()').extract()
        item = AdicrawlerItem(author=author,theme=theme)
        yield item
```

4、存储内容 （pipelines.py）：设计管道存储爬取内容

如果想使用管道的话 那么就必须在settings中开启管道

```
ITEM_PIPELINES = {
            # 管道可以有很多个 那么管道是有优先级 优先级的范围是1到1000 值越小优先级越高
           'scrapy_dangdang.pipelines.ScrapyDangdangPipeline': 300,
}
        # 将在settings.py中这段话取消注释，则打开了通道。
```

然后去pippelines.py中设计管道：

方法一：

```
class ScrapyDangdangPipeline:
        def process_item(self, item, spider):
        # 以下这种模式不推荐 因为每传递一个对象 那么就打开一次文件对文件的操作过于频繁
        # write方法必须要写一个字符串 而不能是其他的对象

```

```
        # w模式 会每一个对象都打开一次文件 覆盖之前的内容
        # 文件存储就不多讲啦
                with open('book.json','a',encoding='utf-8') as fp:
                fp.write(str(item))
                return item
```

方法二：（推荐）

```
class ScrapyDangdangPipeline:

    def open_spider(self,spider):
        self.fp = open('book.json','w',encoding='utf-8')

    # item就是yield后面的对象
    def process_item(self, item, spider):
        self.fp.write(str(item))
        return item

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

5、运行爬虫

> 一般在运行爬虫的时候仍然没有内容查询，则需要考虑将settings.py文件中的ROBOTSTXT_OBEY = True注释掉
> robots协议 注释之后就不遵守协议了 他是君子协议 一般情况下我们不遵守 # BOTSTXT_OBEY = True

```
在cmd中输入：scrapy crawl 爬虫的名字
  eg:scrapy crawl baidu
```

四、项目实战

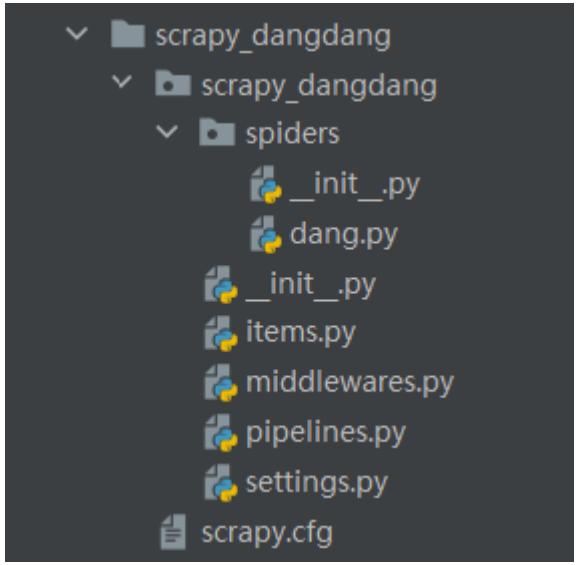
- 打开cmd，创建项目

> scrapy startproject scrapy_dangdang

- 创建爬虫文件

> 先到spiders文件下 :
> cd scrapy_dangdang\scrapy_dangdang\spiders
> 然后创建爬虫文件 :
> scrapy genspider dang category.dangdang.com

- 项目目录



- 确定需要下载的数据，去items.py文件中添加。这里我们准备存储图片、名字和价格

```
import scrapy


class ScrapyDangdangItem(scrapy.Item):
    # define the fields for your item here like:
    # name = scrapy.Field()
    # 通俗的说就是你要下载的数据都有什么
    # 图片
    src = scrapy.Field()
    # 名字
    name = scrapy.Field()
    # 价格
    price = scrapy.Field()
```

- 接下来我们就可以去爬虫文件中去爬取我们需要的内容了（这里是在dang.py文件中）

```
import scrapy
from scrapy_dangdang.items import ScrapyDangdangItem


class DangSpider(scrapy.Spider):
    # 爬虫的名字 一般运行爬虫的时候 使用的值
    name = 'dang'

    # 允许访问的域名
    # 如果是多页下载的话 那么必须要调整的是allowed_domains的范围 一般情况下只写域名
```

- 通过解析拿到数据之后，我们就可以去通道中添加保存的方法了（pippelines.py）
- 首先我们要去settings.py在打开通道和添加通道，完成之后进行下一步

```
ITEM_PIPELINES = {
    # 管道可以有很多个 那么管道是有优先级 优先级的范围是1到1000 值越小优先级越高
   'scrapy_dangdang.pipelines.ScrapyDangdangPipeline': 300,
   'scrapy_dangdang.pipelines.DangDangDownloadPiepline': 301,
}
```

- 通道打开后，在pippelines.py完成下列操作

```python
import os
# 如果想使用管道的话 那么就必须在settings中开启管道
class ScrapyDangdangPipeline:

    def open_spider(self,spider):
        self.fp = open('book.json','w',encoding='utf-8')

    # item就是yield后面的book对象
    def process_item(self, item, spider):
        # 一下这种模式不推荐 因为每传递一个对象 那么就打开一次文件对文件的操作过于频繁
        # # write方法必须要写一个字符串 而不能是其他的对象
        # # w模式 会每一个对象都打开一次文件 覆盖之前的内容
        # with open('book.json','a',encoding='utf-8') as fp:
        #     fp.write(str(item))

        self.fp.write(str(item))
        return item

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

# 多条管道开启
    # 定义管道类
    # 在settings中开启管道
    # 'scrapy_dangdang.pipelines.DangDangDownloadPiepline': 301,
import urllib.request


class DangDangDownloadPiepline:

    def process_item(self,item,spider):
        url = 'http:' + item.get('src')
```

```
        if not os.path.exists('./books/'):
            os.mkdir('./books/')
        filename = './books/' + item.get('name') + '.jpg'
        urllib.request.urlretrieve(url=url,filename=filename)
        return item
```

- 最后在cmd中输入：scrapy crawl dang
- 完成之后就开始下载了，全部完成之后你就会看到多了book.json文件和books文件夹在自己的项目中。里面有数据，则表示项目成功了。