关于scrapy过程的理解:

首先在spider中创建工程，可以以网页名来命名（如jobbole.py），其中爬虫的开始是以jobbole.py中的类xx在其底层（框架的底层，用户无法直接看到）对里面写的start-url发送

Request，而在类xx中的parse方法获取了网页服务器返回的response(可以理解为scrapy已经下载了网页)，之后则是在parse方法中对response进行操作，可以用response.xpath()或

Response.css()这些选择器来筛选已下载网页的标签（或者说是自己想要获取的部分，可以通过自己用浏览器查看该网页的源码自行分析来确定），获取的标签（或者属性/文本）可以通过extract或extract\_first方法来返回一个str的数组或str（即获取node/nodes的具体内容），

若想对获取的新的url进行爬取，则可用yield Request(url,callback=fuc),这可理解为让scrapy

继续对该url发送request请求,这就又回到了类似一开始的过程，但此时是callback中的fuc方法获取返回的respone继续进行刚刚流程的解析，所以我们只需要编写好fuc方法就能达到我们想要的目的，而此时我们在fuc方法的最后可以引入item，通过新建的在items.py中

创建xxitem的类的对象，将从网页中获取的数据传给xxitem(通过字典写入的方式)，之后xxitem会在scrapy底层将该对象传给pipelines(用于下载一些文件如访问图片的url路径下载image)，我们需要在setting.py中的ITEM\_PIPELINES中加入piplie类的路径(以包.模块.类的形式),而piplines默认是调用底层的”虚拟环境的路径+\Lib\site-packages\scrapy\pipelines”里的py文件中的类，如图片则是调用里面的\images.py的ImagesPipeline，在它的get\_media\_requests方法中获取到之前传过来的item，由item获取图片的路径(以key=images\_urls\_field来获取，因此要在setting.py中加入IMAGES\_URLS\_FIELD=’item对象中图片路径的key的值’，注意这个路径是一个数组),并遍历这个数组，对其中的url逐个发送request进行下载，它会在底层完成好，存储路径由setting中的IMAGES\_STORE指定，而后在Pipeline中的item\_completed方法中获取的result参数则会将其存储路径包装进去(result是一个二元元组，第一个表示是否下载成功,第二个是一个字典，里面存有key值为‘path’的图片的存储路径。由于scrapy的pipelines是可以有多个的，它会按照优先级从pipline1到pipeline，每次都传入item对象，在item\_completed方法中进行改造后再传给下一个pipline(因此记住该方法里一定要return item)，所以我们可以将文件的存储信息在相应进行下载的pipline中进行更新或保存，并在最后的pipline中将其存入数据库中,所以为了更好的满足我们的需求，我们可以自定义pipline，通过继承scrapy已有的pipline(如Imagepipeline)，并重写它的item\_completed方法。