1、引擎从调度器中取出一个链接(URL)用于接下来的抓取

2、引擎把URL封装成一个请求(Request)传给下载器

3、下载器把资源下载下来，并封装成应答包(Response)

4、爬虫解析Response

5、解析出实体（Item）,则交给实体管道进行进一步的处理

6、解析出的是链接（URL）,则把URL交给调度器等待抓取

参考网址：

<http://www.cnblogs.com/kongqi816-boke/p/5827243.html#_label0>

<http://blog.csdn.net/zjiang1994/article/details/52779537>

0、scrapy startproject MyScrapy 创建爬虫目录

1、MySpider类（自定义，继承scrapy.Spider），必须设置好爬虫名name和入口地址start\_urls

2、编写爬取方法parse(self, response),里面用for循环遍历每个课程，用正则表达式匹配url、title等数据，并存储到item里面（item类似于一个字典，需要在自建类CourseItem（自定义）里面定义好各个键，并在需要用的时候实例化出来）。如需要爬取多个页面，根据下一页的url规律，可以在parse函数的末尾yield scrapy.Request(nexturl, callback=self.parse)

3、将item 通过yield传给ImgPipeline类，然后再return传给MyPipeline类，是在settings中设置优先级ITEM\_PIPELINES决定先后顺序的

4、ImgPipeline（继承ImagesPipeline），get\_media\_requests方法实现下载图片。若需要自定义图片的名字，重写file\_path方法。图片下载成功/失败后执行item\_completed方法，实现保存图片。

5、MyPipeline（自定义），在process\_item方法里将文字数据以utf-8的格式写入到json文件中。

6、open\_spider和close\_spider方法分别在spider被开启和被关闭时执行。

7、settings文件中：

CONCURRENT\_ITEMS = 1 #与Scrapy性能问题有关

IMAGES\_STORE = 'E:\\Demo\\python\\scrapytest\\img\\' #图片保存目录的一部分

（或者保存在当前目录下IMAGES\_STORE = 'img\\'）

ITEM\_PIPELINES = { #设置优先级，越小越先运行

'scrapytest.MyPipelines.MyPipeline': 100,

'scrapytest.ImgPipelines.ImgPipeline': 1,}