**基于Scrapy框架的京东评论爬虫**

这里以jd\_spider为例来讲述爬取。

本篇教程中将带您完成下列任务:

1. 安装Scrapy
2. 分析目标
3. 创建一个Scrapy项目
4. 定义Item
5. 编写爬取网站的 spider 并提取 Item
6. 编写 Item Pipeline 来存储提取到的Item(即数据)

安装Scrapy

windows下scrapy 的安装

\*注：安装包类型一定要全部一致 32位或64位

**准备工作：**

**安装python**

根据你的需求下载python安装包，安装python

这里使用的是Anaconda - Python2.7

https://www.continuum.io/downloads#windows

在运行中输入"cmd"打开命令窗口，输入"python --version"，如果成功显示python版本号，则python安装成功！



**安装pywin32**

根据需要下载相应的pywin32安装包

https://sourceforge.net/projects/pywin32/files/pywin32/Build%20217/

**安装pip**

一般为python默认安装好的，在cmd命令窗口，输入"pip --version"，若显示pip版本号，则已安装。

**安装lxml**

若系统未安装vs2008或者其它版本的vs，则安装lxml 前，先安装vcforpython27。

在  https://pypi.python.org/pypi/lxml/3.6.0#downloads  中

下载lxml for 2.7，下载安装。

**安装pyOpenSSL**

在cmd命令窗口，用pip安装openSSL，输入命令"pip install pyOpenSSL"

**安装scrapy**

准备工作完成，安装scrapy。在cmd命令窗口，输入命令"pip install scrapy"

等待自动安装，安装完成后输入"scrapy"，提示scrapy的命令提示内容，则整个安装过程结束。

分析目标

这里我们爬的目标是京东商品荣耀8的用户评价



京东的评论采用的是AJAX技术，在该页面源代码中搜索不到评论。

这里F12打开浏览器的调试模式，选择网络-所有，搜索productPageComments，

找到评论页面的请求，获取到请求网址。



接下来在评论页面底部点击“下一页”，同样的方法获取到第二页的URL

https://club.jd.com/comment/productPageComments.action?&productId=3245078&score=0&sortType=5&page=0&pageSize=10

https://club.jd.com/comment/productPageComments.action?&productId=3245078&score=0&sortType=5&page=1&pageSize=10

URL中的callback=fetchJSON\_comment98vv21171与isShadowSku=0部分对实际访问时无影响，所以这里已经精简掉了。

现在分析这两个URL的特点：

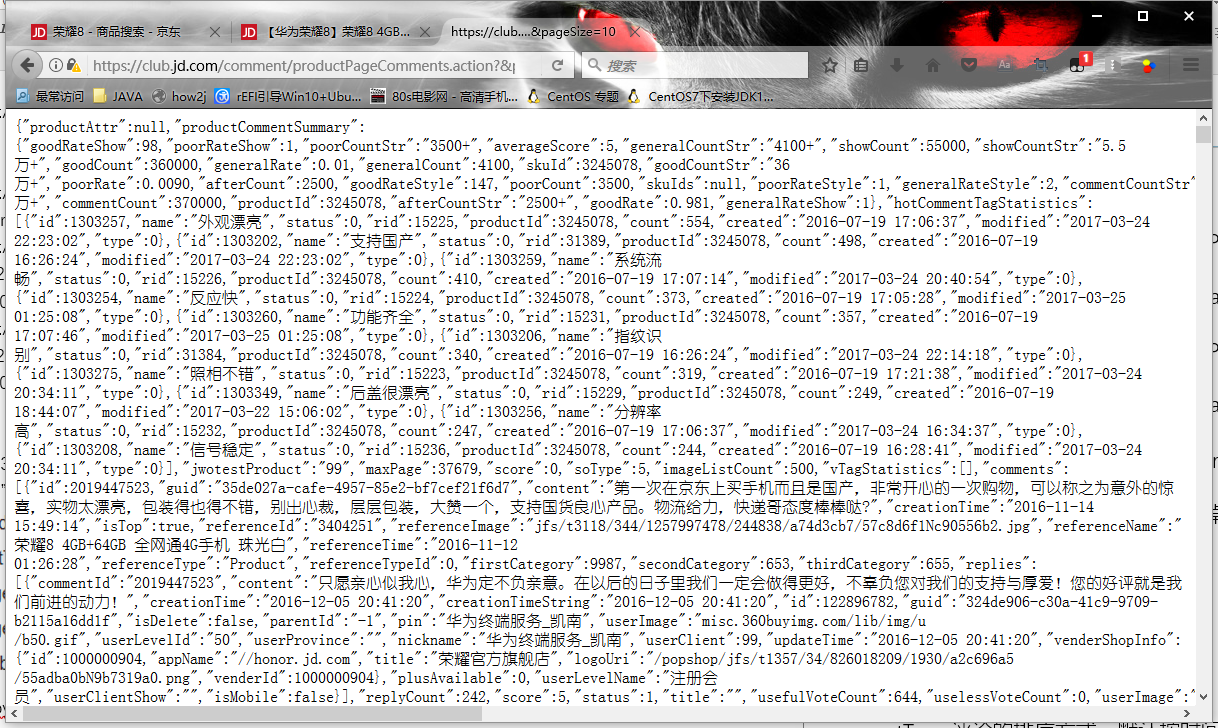
productId:商品的ID

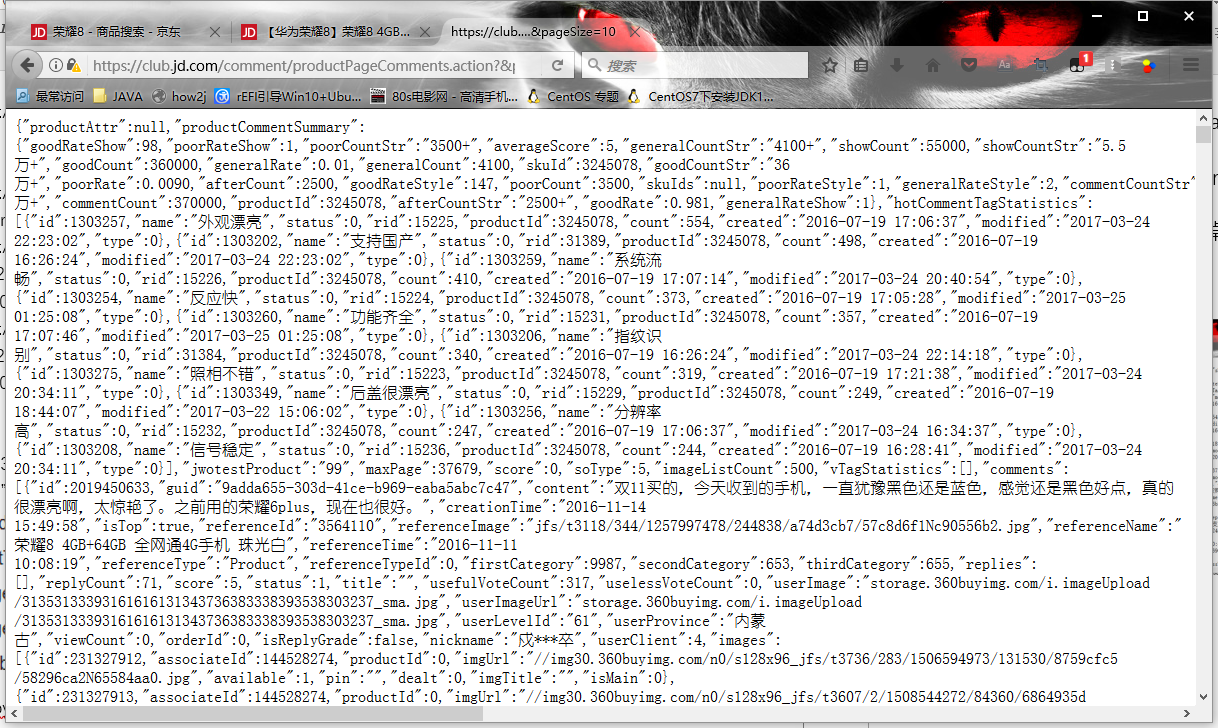
sortType:评论的排序方式，默认按时间排序，值为5

page：当前是第几页评论，从0开始递增

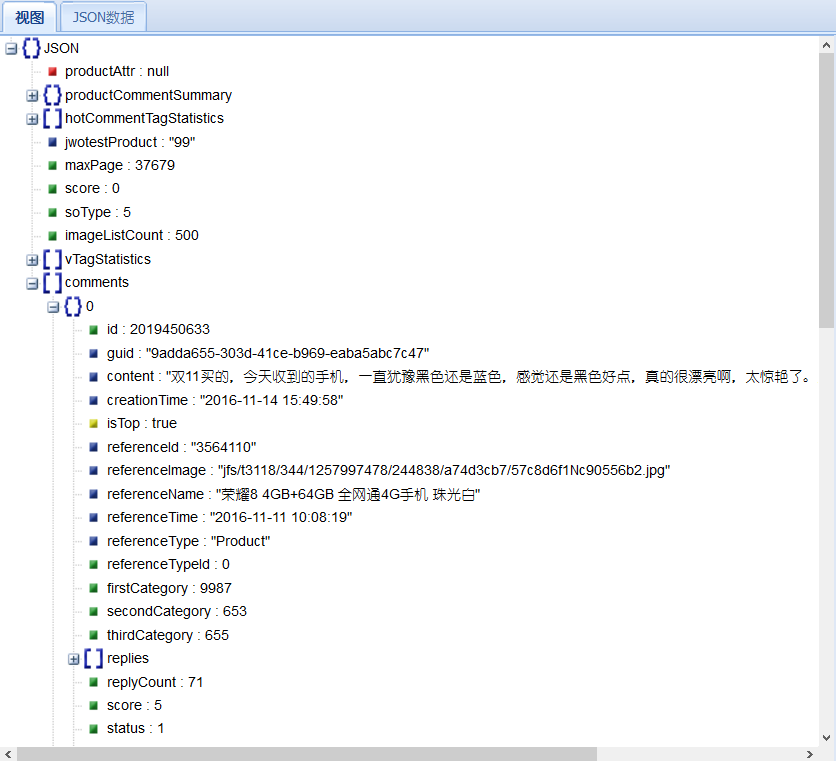
pageSize:指定每一页评论的数量，默认10

经过分析，我们在写spider时可以通过改变productId和page循环爬取评论。

下面是这两个页面的内容



我们发现数据是json格式的，将其进行在线解析



这为我们后面怎么定义Item，怎么写spider做准备

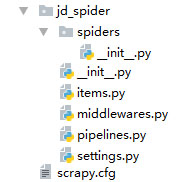
**创建项目**

在开始爬取之前，您必须创建一个新的Scrapy项目。 进入您打算存储代码的目录中，运行下列cmd命令:

scrapy startproject jd\_spider

*#命令中的jd\_spider为项目名称*

该命令将会创建包含下列内容的 jd\_spider 目录:



这些文件分别是:

* scrapy.cfg: 项目的配置文件
* spiders/: 该项目的python模块。之后您将在此加入代码。
* spiders/items.py: 项目中的item文件.
* spiders/pipelines.py: 对已抓取、解析后的内容的处理。 spiders/settings.py: 项目的设置文件.
* spiders/spiders/: 放置spider代码的目录.

**定义Item**

*Item* 是保存爬取到的数据的容器；其使用方法和python字典类似， 并且提供了额外保护机制来避免拼写错误导致的未定义字段错误。

并且定义类型为 scrapy.Field 的类属性来定义一个Item。

我们通过解析过的数据对item进行建模。这里我们要获取评论ID，评论时间，商品型号，评论内容。对此，在item中定义相应的字段。编辑 jd\_spider目录中的 items.py 文件:

**import** scrapy  
**class** JdSpiderItem(scrapy.Item):  
 *#用户ID* id = scrapy.Field()  
 *#评论时间* creationTime = scrapy.Field()  
 *#型号* referenceName = scrapy.Field()  
 *#评论内容* content = scrapy.Field()

**编写第一个爬虫(Spider)**

Spider是用户编写用于从单个网站(或者一些网站)爬取数据的类。

其包含了一个用于下载的初始URL，如何跟进网页中的链接以及如何分析页面中的内容， 提取生成 item 的方法。

为了创建一个Spider，您必须继承 scrapy.Spider 类， 且定义以下三个属性:

* name: 用于区别Spider。 该名字必须是唯一的，您不可以为不同的Spider设定相同的名字。
* start\_urls: 包含了Spider在启动时进行爬取的url列表。 因此，第一个被获取到的页面将是其中之一。 后续的URL则从初始的URL获取到的数据中提取。
* parse() 是spider的一个方法。 被调用时，每个初始URL完成下载后生成的 Response 对象将会作为唯一的参数传递给该函数。 该方法负责解析返回的数据(response data)，提取数据(生成item)以及生成需要进一步处理的URL的 Request 对象。

cmd进入jd\_spider/spiders文件夹下，运行以下命令创建名为jd的基本爬虫模板

Scrapy genspider -t jd club.jd.com

以下为我们的第一个Spider代码，保存在 jd\_spider/spiders 目录下的 jd.py 文件中:

**import** scrapy  
**import** json  
**from** scrapy.http **import** Request  
**from** jd\_spider.items **import** JdSpiderItem  
  
**class** JdSpider(scrapy.Spider):  
 name = **"jd"** allowed\_domains = [**"club.jd.com"**]  
 start\_urls = [**'https://club.jd.com'**]  
  
 **def** parse(self, response):  
url1 = **"https://club.jd.com/comment/productPageComments.action?&productId="** productId = **"3245078"** url2 = **"&score=0&sortType=5&page="** page = **"page"** url3 = **"&pageSize=10"  
 for** i **in** range(0, 10):  
 page = str(i)  
 url = url1+productId+url2+page+url3  
 **yield** Request(url=url,callback=self.page)  
 **def** page(self,response):  
js = json.loads(response.body\_as\_unicode())  
 *#json.loads(unicode(str, "utf-8"))*

*#用json.loads将数据解析成python可以操作的对象*

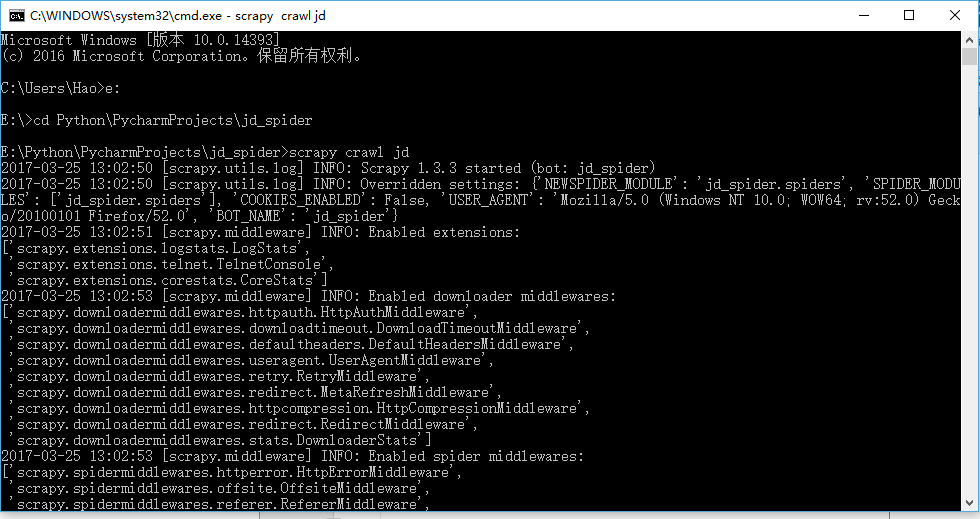
comments = js[**'comments'**]  
 *#items = []* **for** comment **in** comments:  
item = JdSpiderItem()  
 item[**'id'**] = comment[**'id'**]  
 item[**'creationTime'**] = comment[**'creationTime'**]  
 item[**'referenceName'**] = comment[**'referenceName'**]  
 item[**'content'**] = comment[**'content'**]  
**yield** item

**爬取**

进入项目的根目录，执行下列cmd命令启动spider:

scrapy crawl jd

crawl jd 启动用于爬取目标的spider，您将得到类似的输出:



**刚才发生了什么？**

Scrapy为Spider的每个URL创建了 scrapy.Request 对象，并将 parse 方法作为回调函数(callback)赋值给了Request。

Request对象经过调度，执行生成 [scrapy.http.Response](http://scrapy-chs.readthedocs.io/zh_CN/latest/topics/request-response.html#scrapy.http.Response) 对象并送回给spider [parse()](http://scrapy-chs.readthedocs.io/zh_CN/latest/topics/spiders.html#scrapy.spider.Spider.parse) 方法。

**提取数据**



从分析中可以看出，我们可以循环从comments节点提取id,content,creationTime,refernceType等节点

**使用item**

Item 对象是自定义的python字典。 您可以使用标准的字典语法来获取到其每个字段的值。(字段即是我们之前用Field赋值的属性):

>>> item = JdSpiderItem()

>>> item[**'id'**] = comment[**'id'**]

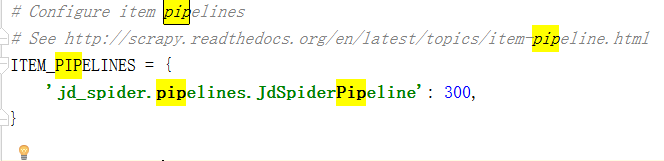
一般来说，Spider将会将爬取到的数据以 [Item](http://scrapy-chs.readthedocs.io/zh_CN/latest/topics/items.html#scrapy.item.Item) 对象返回。所以为了将爬取的数据返回，我们的代码将是:

js = json.loads(response.body\_as\_unicode())  
comments = js[**'comments'**]  
**for** comment **in** comments:  
 item = JdSpiderItem()  
 item[**'id'**] = comment[**'id'**]  
 item[**'creationTime'**] = comment[**'creationTime'**]  
 item[**'referenceName'**] = comment[**'referenceName'**]  
 item[**'content'**] = comment[**'content'**]

**yield** item

**保存爬取到的数据**

首先在settings中启用pipelines(去掉pipelines的注释即可)



在pipelines中编写将数据保存到excel中

**from** openpyxl **import** Workbook  
wb = Workbook()  
ws = wb.active  
*#添加列标题*ws.append([**'用户ID'**, **'评价时间'**, **'商品型号'**, **'用户评价'**])  
  
**class** JdSpiderPipeline(object):  
 **def** process\_item(self, item, spider):  
 *#创建行对象* line = [item[**'id'**], item[**'creationTime'**], item[**'referenceName'**], item[**'content'**]]  
 *#写入一行* ws.append(line)  
 *#保存* wb.save(**'jd\_spider.xlsx'**)  
 **return** item