多级页面的抓取与图片抓取的技巧

# 多级页面的抓取

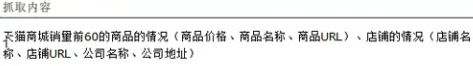
# 实用技巧1



# 需求分析

抓取网站：天猫商城

抓取内容：



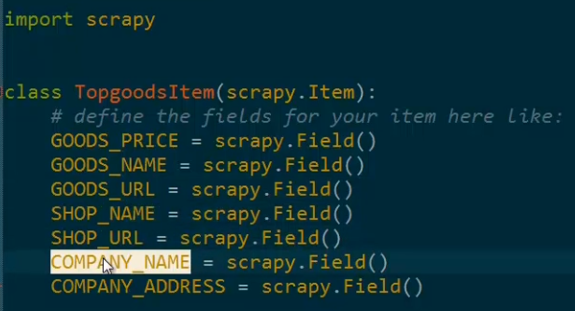
存储结构：Excel表格

# 项目实施

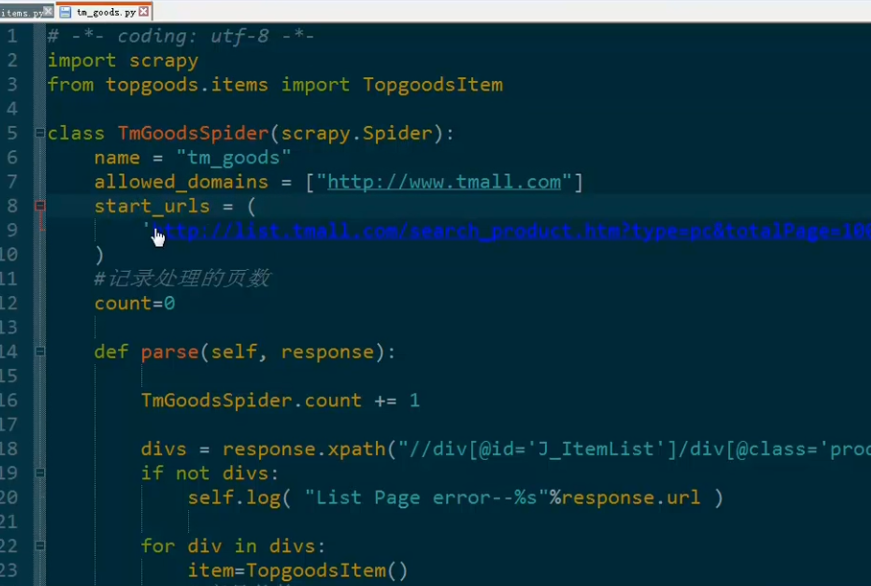
# 代码解读

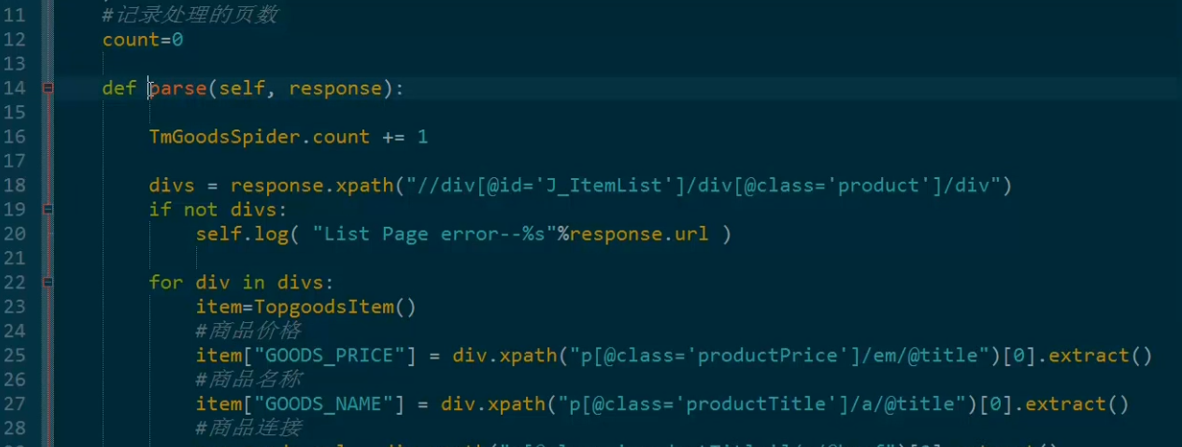


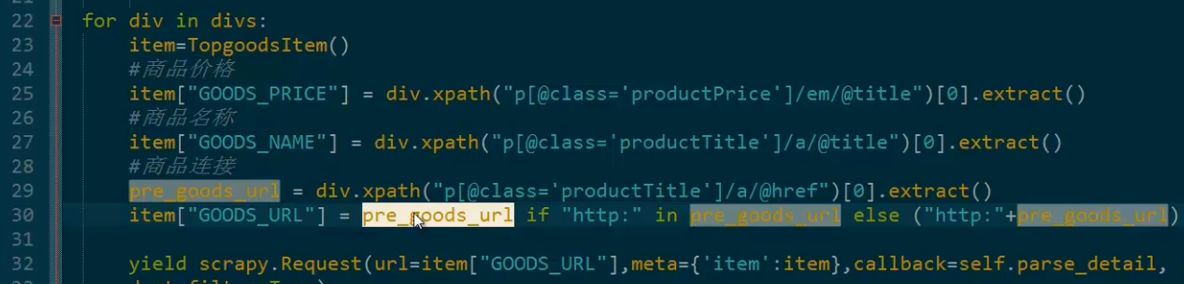
## item编写



## spider：



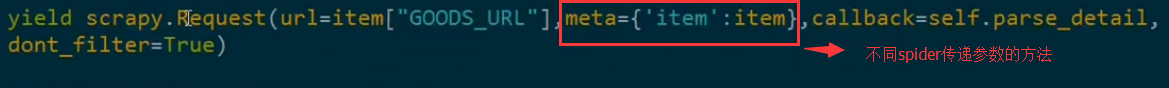






# 怎么在同一个spider的不同parse函数中传递数据？

**将数据如item通过meta参数传递到下一次的请求中，然后利用Response的meta获取该数据。**





# 项目工程已经打包保存

# 图片抓取的技巧

# 需求分析

目标网站： 天猫商城

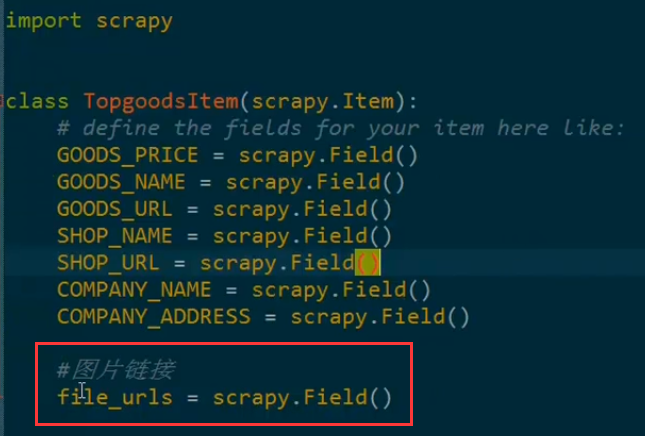
抓取内容： 添加“图片”



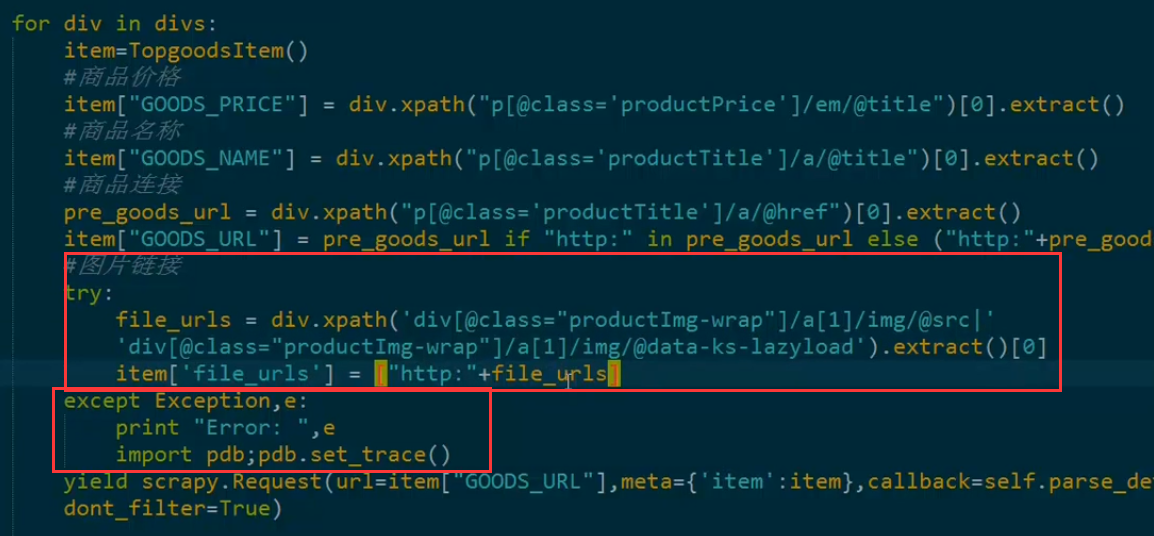
# 代码解读



**Item编写**：

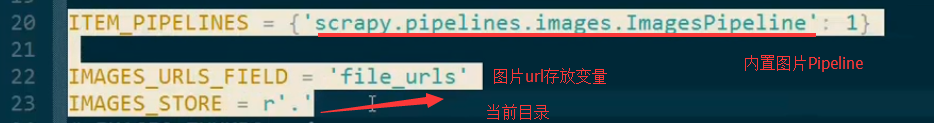


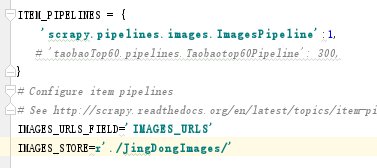
Spider编写：



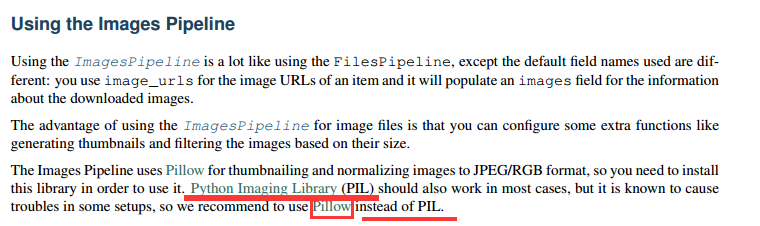
在**settings.py** 中设置Pipeline：

对于图片的下载，我们采用内置的**ImagesPipeline**：





# ImagesPipeline的介绍及使用方法：



使用**ImagesPipeline**之前，必须安装Pillow。

pip install pillow

其余的需要在settings.py 文件中设置：











总结：

**在settings.py 中至少三个参数：**

**ITEM\_PIPLINES={‘’}**

**IMAGES\_URLS\_FIELD=’’**

**IMAGES\_STORE=’’**

**错误异常：**

[**解决ValueError('Missing scheme in request url: %s' % self.\_url)**](http://blog.csdn.net/lcyong_/article/details/72858453)

**原因分析：指定的IMAGES\_URLS\_FIELD的类型必须是一个string的list类型，不能简单的传递url字符串，应该是[url]。如果是单个图片，一个url，就是一个元素的list，当然如果多个url，可以利用append方法添加。**

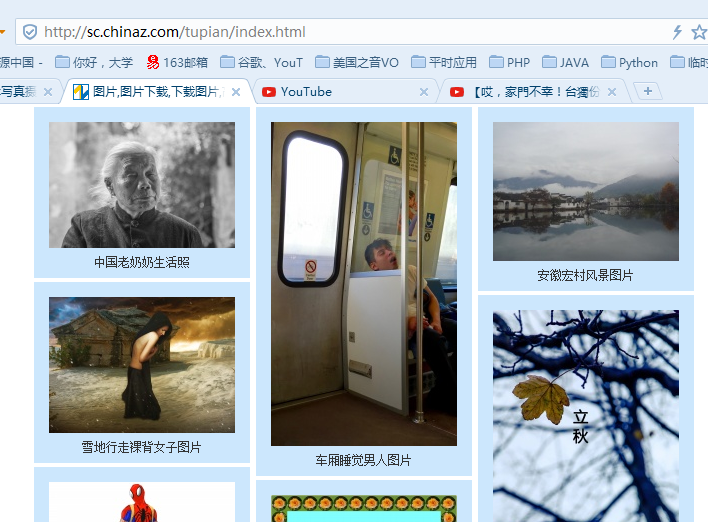
**如：**

item[**'IMAGES\_URLS'**] = [**"http:"**+li.xpath(**'div[@class="gl-i-wrap"]/div[@class="p-img"]/a[@target="\_blank"]**[**/img/@src'**).extract()[0](mailto:/img/@src').extract()%5b0)]];

item[**'IMAGES\_URLS'**] = “http://www.baidu.com/1.jpg”;//错误

item[**'IMAGES\_URLS'**] = [“<http://www.baidu.com/1.jpg>”];//错误

# 图片的抓取：



<http://sc.chinaz.com/tupian/index.html>

可以翻页: 翻页URL的特点 [http://sc.chinaz.com/tupian/index**\_num**.html](http://sc.chinaz.com/tupian/index_num.html)

# 方法1 ：利用内置的ImagesPipeline

确定：无法修改图片名字。

**不算是很复杂，将一个图片保存成一个item，返回给Pipeline处理即可。**

工程： **imageDown2工程**

**spider文件：**

*# -\*- coding: utf-8 -\*-***import** scrapy  
  
**import** logging  
**from** imageDown2.items **import** Imagedown2Item  
**import** os  
**import** re  
**import** random  
  
  
**class** Images2Spider(scrapy.Spider):  
 name = **'images2'  
 def** start\_requests(self):  
 start\_url = **'http://sc.chinaz.com/tupian/index.html'  
 yield** scrapy.Request(url=start\_url, callback=self.parse, meta={**"false"**: **True**})  
  
 **def** parse(self, response):  
 **if "flag" in** response.meta.keys():  
 flag = response.meta[**'flag'**]  
 **else**:  
 flag = **True  
 try**:  
 logging.info(response.url[-10:-5])  
 **if** flag:  
 self.writeInfo(**"index.html"**, response.text, **'utf-8'**)  
 **else**:  
 self.writeInfo(**"index"** + re.findall(**r'\d+'**, response.url[-10:-5])[0] + **".html"**, response.text,  
 **'utf-8'**)  
 item = Imagedown2Item()  
 imageBlock = response.xpath(**'//div[@id="container" and @class="clearfix psdk imgload"]'**);  
 imgList = imageBlock.xpath(**'div[@class="box picblock col3"]/div/a[@target="\_blank"]/img'**)  
 **for** img **in** imgList:  
 **try**:  
 item[**'IMAGES\_NAME'**] = img.xpath(**'@alt'**).extract()[0]  
 item[**'IMAGES\_URLS'**] = img.xpath(**'@src2'**).extract()  
 **yield** item  
 *# yield scrapy.Request(url=item['IMAGES\_URLS'], callback=self.parse\_picture, meta={"meta": item})* **except** Exception **as** ee:  
 logging.error(ee)  
 logging.info(**'BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB'**)  
 **continue  
 except** Exception **as** e:  
 logging.error(e)  
 logging.info(**'CCCCCCCCCCCCCCCCCCCCCCCCCCCCC'**)  
 **if** flag:  
 url\_raw = response.url  
 **for** i **in** range(2, 11):  
 url\_new = url\_raw[0:-5] + **"\_"** + str(i) + **".html"  
 yield** scrapy.Request(url=url\_new, callback=self.parse, meta={**'flag'**: **False**})  
  
 **def** writeInfo(self, fileName, text, bianma):  
 fDir = **r'./indexs/'  
 if not** os.path.exists(fDir):  
 os.makedirs(fDir)  
 **with** open(fDir + fileName, **'w'**, encoding=bianma) **as** f:  
 f.write(text);  
 f.close();

items：

**import** scrapy  
  
  
**class** Imagedown2Item(scrapy.Item):  
 *# define the fields for your item here like:  
 # name = scrapy.Field()* IMAGES\_NAME = scrapy.Field()  
 IMAGES\_URLS = scrapy.Field()

**setting文件：**

ITEM\_PIPELINES = {  
 **'scrapy.pipelines.images.ImagesPipeline'**:1,  
 *# 'imageDown2.pipelines.Imagedown2Pipeline': 300,*}  
IMAGES\_URLS\_FIELD=**'IMAGES\_URLS'**IMAGES\_STORE=**r'.'**

# 自己编写图片下载

\*（这里由于返回item，会打印日志，造成日志文件过大，采用直接在spider下下载图片）

发现，很多图片都是重复的，所以为了浪费存储空间，首先检查该图片是否已经下载过，下载过的图片，不再存储。

*# -\*- coding: utf-8 -\*-***import** scrapy  
**import** logging  
**from** imagesDown.items **import** ImagesdownItem  
**import** os  
**import** re  
**import** random  
  
**class** ImagesSpider(scrapy.Spider):  
 name = **'images'** *# allowed\_domains = ['chinaz.com']  
 # start\_urls = ['http://sc.chinaz.com/tupian/index.html']* **def** start\_requests(self):  
 start\_url = **'http://sc.chinaz.com/tupian/index.html'  
 yield** scrapy.Request(url = start\_url,callback=self.parse,meta={**"false"**:**True**})  
  
 **def** parse(self, response):  
 **if "flag" in** response.meta.keys():  
 flag = response.meta[**'flag'**]  
 **else**:  
 flag = **True  
 try**:  
 logging.info(response.url[-10:-5])  
 **if** flag:  
 self.writeInfo(**"index.html"**, response.text, **'utf-8'**)  
 **else**:  
 self.writeInfo(**"index"**+ re.findall(**r'\d+'**,response.url[-10:-5])[0]+**".html"**, response.text, **'utf-8'**)  
 item = ImagesdownItem()  
 imageBlock = response.xpath(**'//div[@id="container" and @class="clearfix psdk imgload"]'**);  
 imgList = imageBlock.xpath(**'div[@class="box picblock col3"]/div/a[@target="\_blank"]/img'**)  
 **for** img **in** imgList:  
 **try**:  
 item[**'IMAGES\_NAME'**] = img.xpath(**'@alt'**).extract()[0]  
 item[**'IMAGES\_URLS'**] = img.xpath(**'@src2'**).extract()[0]  
 **yield** scrapy.Request(url = item[**'IMAGES\_URLS'**], callback = self.parse\_picture, meta = {**"meta"**: item})  
 **except** Exception **as** ee:  
 logging.error(ee)  
 logging.info(**'BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB'**)  
 **continue  
 except** Exception **as** e:  
 logging.error(e)  
 logging.info(**'CCCCCCCCCCCCCCCCCCCCCCCCCCCCC'**)  
 **if** flag:  
 url\_raw = response.url  
 **for** i **in** range(2,11):  
 url\_new = url\_raw[0:-5]+**"\_"**+str(i)+**".html"  
 yield** scrapy.Request(url = url\_new,callback=self.parse,meta={**'flag'**:**False**})  
  
 **def** parse\_picture(self,response):  
 **try**:  
 contentByte = response.body  
 item = response.meta[**'meta'**]  
 filePath = **"./images/"**+ item[**'IMAGES\_NAME'**]+**".jpg"** *# # 利用while循环来保证重复的图片也能够下载  
 # while os.path.exists(filePath):  
 # filePath = filePath[0:-4]+ str(random.randint(1,100))+".jpg"* **if** os.path.exists(filePath):  
 **return ""**;*#若已经存在，则结束* **with** open(filePath,**'wb'**) **as** f:  
 f.write(contentByte)  
 logging.info(**'Successfuly'**)  
 f.close()  
 **except** Exception **as** e:  
 logging.error(e)  
 logging.info(**"DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD"**)  
  
 **def** writeInfo(self,fileName,text,bianma):  
 fDir = **r'./indexs/'  
 if not** os.path.exists(fDir):  
 os.makedirs(fDir)  
 **with** open(fDir+fileName, **'w'**, encoding=bianma) **as** f:  
 f.write(text);  
 f.close();

items.py

**import** scrapy  
**class** ImagesdownItem(scrapy.Item):  
 *# define the fields for your item here like:  
 # name = scrapy.Field()* IMAGES\_NAME =scrapy.Field()  
 IMAGES\_URLS =scrapy.Field()  
 IMAGES\_BODY =scrapy.Field()

# 编写Pipeline下载图片

也就是把第二个在spider中写的哪一部分，转移到Pipeline文件中，这里不再重复。