**基于Scrapy的爬虫**

# 一、实验目的

* 熟悉Scrapy
* 掌握基于Scrapy的数据爬取

# 二、实验内容

本实验主要是为了后续的大数据处理和Web电商系统准备数据，基于Scrapy从网络上爬取所需要的数据，学生需要尽可能的根据自己认为的需要获取网络上的数据内容，包括但不限于图书信息、图书分类信息、图书评价信息、图书销售信息等等。

# 三、实验知识点

* Scrapy
* 基于Scrapy的爬虫

# 四、实验环境

* Centos 7.9.2009 64位

# 五、实验步骤

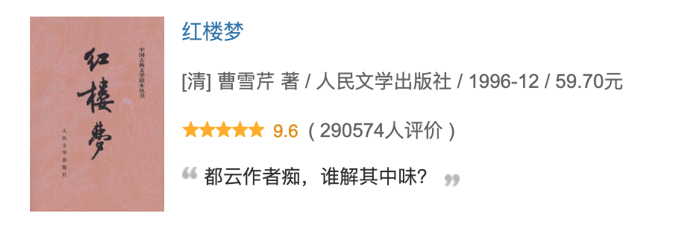
本实验给出样例程序使用Scrapy爬取豆瓣图书TOP250。爬取的内容为：图书id、书名、作者、出版日期、出版社、小图、中图、大图、类别id、类别名称、单价、短评。需要注意爬取豆瓣网图书，仅作为课程学习使用，不得用于任何商业目的。在爬取过程可能会被封IP。

* 采集工作有两层连接，第一层从https://book.douban.com/top250?start=0以25的位数递增，第二层为每本图书的详情页面。
* 采集方法可以使用：xpath方法或css方法。

## 1. 基于Xpath的方法

首先，使用浏览器开发者工具，定位元素后，查找xpath路径。

以<https://book.douban.com/top250?start=0>中第一本图书《红龙梦》为例。



可以确定每本图书的路径为："//[@id="content"]/div/div[1]/div/table"。

令book = response.xpath('//[@id="content"]/div/div[1]/div/table')， 那么，

标题路径为：标题路径：book.xpath("./tr/td[2]/div[1]/a/@title")，

评价人数路径为：book.xpath("./tr/td[2]/p[2]/span/text()")。

确定xpath之后，就根据先前创建的scrapy\_book项目，编写具体的爬虫代码。

（1）编写items

具体代码如下：

|  |
| --- |
| class ScrapyBookItem(scrapy.Item):  # 图书id  isbn = scrapy.Field()  # 书名  title = scrapy.Field()  # 作者  author = scrapy.Field()  # 出版日期  pub\_date = scrapy.Field()  # 出版社  publisher = scrapy.Field()  # 小图  s\_img = scrapy.Field()  # 中图  m\_img = scrapy.Field()  # 大图  b\_img = scrapy.Field()  # 类别id  category\_id = scrapy.Field()  # 类别名称  category = scrapy.Field()  # 单价  price = scrapy.Field()  # 图书一句话描述  scrible = scrapy.Field()  # 图书详情页的url  sub\_url = scrapy.Field() |

（2）编写基于xpath的spider

具体代码如下：

|  |
| --- |
| class SpiderForXPath(scrapy.Spider):  name = 'spider\_xpath\_douban'  def start\_requests(self):  for a in range(10):  url = 'https://book.douban.com/top250?start={}'.format(a \* 25)  yield scrapy.Request(url=url, callback=self.parse)  def parse(self, response):  items = []  for book in response.xpath('//\*[@id="content"]/div/div[1]/div/table'):  item = ScrapyBookItem()  title1 = book.xpath("./tr/td[2]/div[1]/a/@title").extract\_first().replace('\n', '').strip()  title2 = "无" if book.xpath("./tr/td[2]/div[1]/span/text()").extract\_first() == None else book.xpath(  "./tr/td[2]/div[1]/span/text()").extract\_first().replace('\n', '').strip()  item['title'] = title1 + "(" + title2 + ")"  item['s\_img'] = book.xpath("./tr/td[1]/a/img/@src").extract\_first().replace('\n', '').strip()  item['scrible'] = "无" if book.xpath("./tr/td[2]/p[2]/span/text()").extract\_first() == None else book.xpath(  "./tr/td[2]/p[2]/span/text()").extract\_first().replace('\n', '').strip()  sub\_url = book.xpath("./tr/td[2]/div/a/@href").extract\_first().replace('\n', '').strip()  items.append(item)  # meta={"item":item} 传递item引用SinaItem对象  yield scrapy.Request(url=sub\_url, callback=self.parse\_second, meta={"item": item})  def parse\_second(self, response):  item = response.meta["item"]  item["category\_id"] = ""  item["category"] = ""  # book = response.xpath('//div[@class="indent"]/div').extract\_first()  # item["author"] = book.xpath("./div[1]/a[1]/text()").extract\_first().replace('\n', '').strip()  # item["publisher"] = book.xpath("./div[1]/a/@href").extract\_first().replace('\n', '').strip()  # item["pub\_date"] = book.xpath("./div[1]/a/@href").extract\_first().replace('\n', '').strip()  # item["price"] = book.xpath("./div[1]/a/@href").extract\_first().replace('\n', '').strip()  # item["m\_img"] = book.xpath("./div[1]/a/@href").extract\_first().replace('\n', '').strip()  # item["b\_img"] = book.xpath("./div[1]/a/@href").extract\_first().replace('\n', '').strip()  # item["isbn"] = book.xpath("./div[2]/a[1]/text()").extract\_first().replace('\n', '').strip()  yield item |

（3）编写pipelines

具体代码如下：

|  |
| --- |
| import codecs  import json  class ScrapyBookPipeline(object):  def \_\_init\_\_(self):  self.file = codecs.open('data.json', 'w', encoding='utf-8')  def process\_item(self, item, spider):  line = json.dumps(dict(item), ensure\_ascii=False) + "\n"  self.file.write(line)  return item  def spider\_closed(self, spider):  self.file.close() |

（4）修改配置settings，在文件中修改user-agent等其他信息以伪装成浏览器。修改内容如下：

|  |
| --- |
| USER\_AGENT = 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36'  ITEM\_PIPELINES = {  'douban.pipelines.DoubanPipeline': 300,  } |

（5）编写start.py以直接调用

具体代码如下：

|  |
| --- |
| # -\*- coding: utf-8 -\*-  from scrapy import cmdline  # 执行基于xpath的spider  cmdline.execute('scrapy crawl spider\_xpath\_douban'.split())  # 执行基于css的spider  # cmdline.execute('scrapy crawl spider\_xpath\_douban'.split()) |

## 2. 基于CSS的方法

首先，使用浏览器开发者工具，定位元素后观察下方栏的css标签节点。如下图所示：



可以确定每本书的路径为：'tr.item'，令book=response.css('tr.item')，那么：

标题路径：book.css('div.pl2 a::text')，

图片路径：book.css('a.nbg img').xpath('@src')，其他的同理。

编写基于CSS的spider，具体代码如下：

|  |
| --- |
| class SpiderForCSS(scrapy.Spider):  name = 'spider\_css\_douban'  def start\_requests(self):  for a in range(10):  url = 'https://book.douban.com/top250?start={}'.format(a \* 25)  yield scrapy.Request(url=url, callback=self.parse)  def parse(self, response):  items = []  for book in response.css('tr.item'):  item = ScrapyBookItem()  item['title'] = book.css('div.pl2 a::text').extract\_first().replace('\n', '').strip()  item['score'] = book.css('div.star.clearfix span.rating\_nums::text').extract\_first().replace('\n',  '').strip()  item['scrible'] = book.css('p.quote span.inq::text').extract\_first().replace('\n', '').strip()  item['num'] = book.css('div.star.clearfix span.pl::text').extract\_first().strip("(").strip(")").replace(  '\n', '').strip()  item['img'] = book.css('a.nbg img').xpath('@src').extract\_first().replace('\n', '').strip()  items.append(item)  print(items)  return items |

至此，一个简单的图书爬虫就编写完了。学生还需要根据自己的调研，编写更加复杂的爬虫程序，以获取尽可能多的图书信息，为后续的大数据处理和Web图书电商系统做准备。