[利用Python爬取豆瓣电影](https://www.cnblogs.com/fangtaoa/p/8366528.html)

目标:使用Python爬取豆瓣电影并保存MongoDB数据库中

我们先来看一下通过浏览器的方式来筛选某些特定的电影:



我们把URL来复制出来分析分析:

https://movie.douban.com/tag/#/?sort=T&range=0,10&tags=%E7%94%B5%E5%BD%B1,%E7%88%B1%E6%83%85,%E7%BE%8E%E5%9B%BD,%E9%BB%91%E5%B8%AE

有3个字段是非常重要的:

　　1.sort=T

　　2.range=0,10

　　3.tags=%E7%94%B5%E5%BD%B1,%E7%88%B1%E6%83%85,%E7%BE%8E%E5%9B%BD,%E9%BB%91%E5%B8%AE

具体分析方法如下:

　　1.sort:表示排序方式,可以看到它有3中排序方式

　　根据上图可以直到每个字母表示的含义:

**T:热度排序,**

**R:时间排序,**

**S:评价排序:**

　　2.range=0,10;表示一个范围,具体是什么范围呢?



　　range参数我们也搞定了,它就是表示评分区间!

　　　　默认评分区间是:0-10

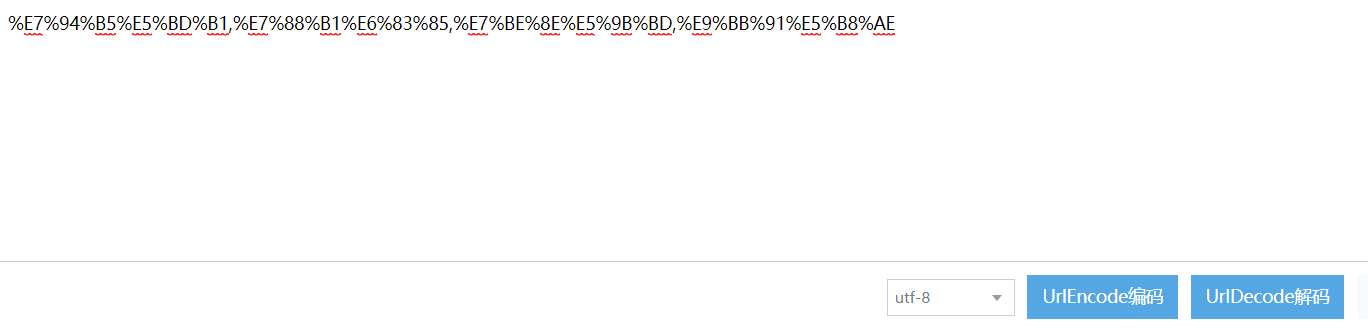
　　3.tags:同样的原理,这是一个标签

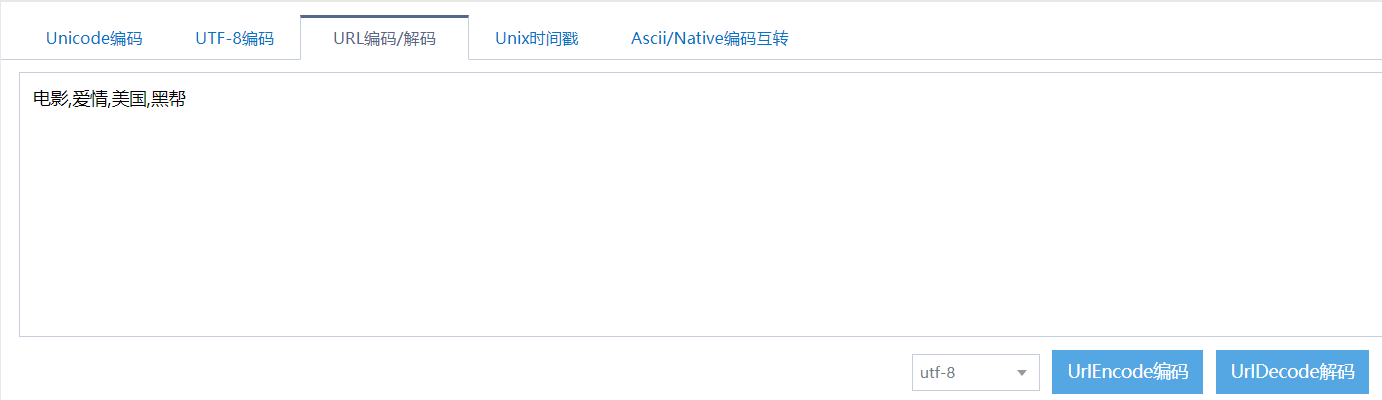
　　　　我们选中的标签有:电影,爱情,美国,黑帮4个标签,但是在tags里面我们看到的不是这写汉字,而是被编码过的形式!

　　　　%E7%94%B5%E5%BD%B1,%E7%88%B1%E6%83%85,%E7%BE%8E%E5%9B%BD,%E9%BB%91%E5%B8%AE)

　　　　那如何知道这些字符是表示什么呢?

　　我们可以到网上进行解码看看正不正确?



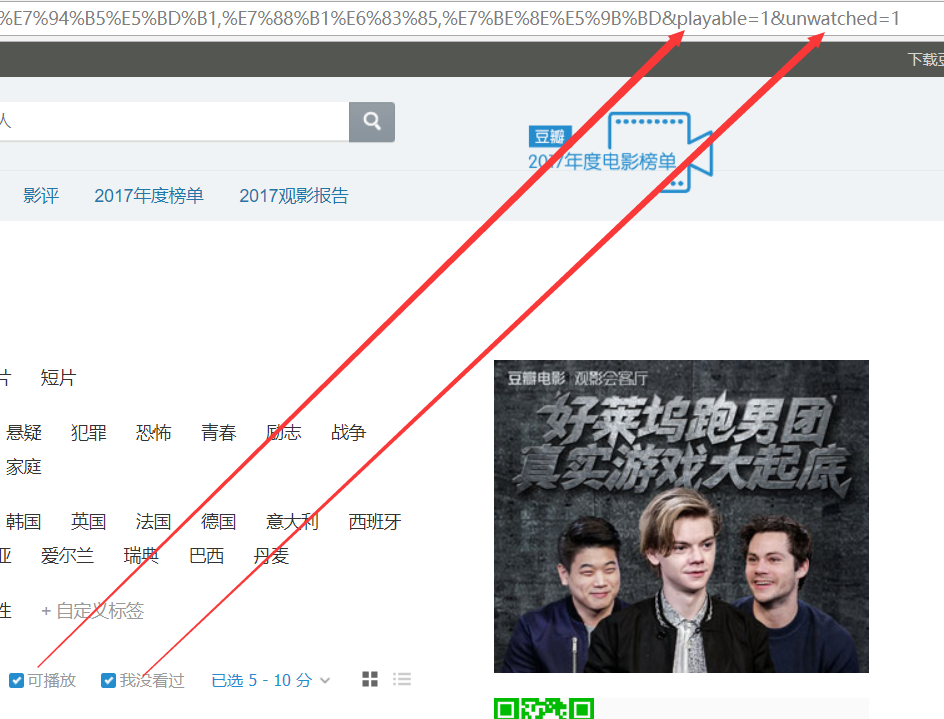


　　4.那么还有没有可以选择的参数呢?

　　我们还有2个参数可以选择!

　　　　playbale=1:表示可播放

　　　　unwatched=1:表示还没看过的



至此,我们就已经把URL中的查询参数全都弄明白了!

但是,又有一个问题了,当我们在浏览器中点击"加载更多"按钮时,这个地址栏中的URL并没有发生变化,但是电影信息可以加载出来了!这是为什么?

　　如果知道AJAX加载技术的读者可能知道这个原理,实际上就是异步加载,服务器不需要刷新整个网页,只需要刷新局部网页就可以把数据展示到网页中,这样不仅可以加快速度,也可以减少服务器的压力.

重点来了:

　　抓包结果:



看看浏览器地址栏的URL与Request URL有什么不一样的地方?

　　我们在浏览器地址栏中看到的URL是:

　　　　https://movie.douban.com/tag/#/?

　　　　　　sort=S&range=5,10&tags=%E7%94%B5%E5%BD%B1,%E7%88%B1%E6%83%85,%E7%BE%8E%E5%9B%BD&playable=1&unwatched=1

　　实际浏览器发送的Request URL是:

　　　　https://movie.douban.com/j/new\_search\_subjects?

　　　　　　sort=S&range=5,10&tags=%E7%94%B5%E5%BD%B1,%E7%88%B1%E6%83%85,%E7%BE%8E%E5%9B%BD&playable=1&unwatched=1&start=0

　　除了被红色标记的地方不同之外,其他地方都是一样的!那我们发送请求的时候应该是用哪一个URL呢?

　　在上面我就已提到了,在豆瓣电影中,是采用异步加载的方式来加载数据的,也就是说在加载数据的过程中,地址栏中的URL是一直保持不变的,那我们还能用这个URL来发送请求吗?当然不能了!

既然不能用地址栏中的URL来发送请求,那我们就来分析一下浏览器实际发送的Request URL:

　　我们把这个URL复制到浏览器中看看会发生什么情况!



我们可以看到这个URL的响应结果恰恰就是我们想要的数据,采用json格式.在Python中,我们可以利用一些工具把它转换成字典格式,来提取我们想要的数据.

距离我们成功还有一小步:

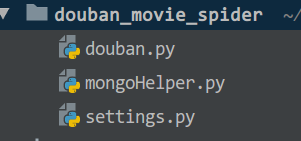
　　在这个URL中,我们看到还有一个参数:start,这个是干嘛的呢?

　　这个数值表示偏移量,来控制每一次加载的偏移位置是在哪里!比如我们把它设置成20,表示一次请求的电影数量.那么得到的结果如下:



到这里,该案例的思路,难点就已经全都捋清楚了,剩下的就是代码的事情了!

项目结构:



完整的代码如下:

settings.py

[复制代码](javascript:void(0);)

1 User\_Agents =[

2 'Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10\_6\_8; en-us) AppleWebKit/534.50 (KHTML, like Gecko) Version/5.1 Safari/534.50',

3 'Mozilla/5.0 (Windows; U; Windows NT 6.1; en-us) AppleWebKit/534.50 (KHTML, like Gecko) Version/5.1 Safari/534.50',

4 'Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0',

5 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:2.0.1) Gecko/20100101 Firefox/4.0.1',

6 'Opera/9.80 (Macintosh; Intel Mac OS X 10.6.8; U; en) Presto/2.8.131 Version/11.11',

7 'Opera/9.80 (Windows NT 6.1; U; en) Presto/2.8.131 Version/11.11',

8 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_7\_0) AppleWebKit/535.11 (KHTML, like Gecko) Chrome/17.0.963.56 Safari/535.11',

9 ]

[复制代码](javascript:void(0);)

mongoHelper.py

[复制代码](javascript:void(0);)

1 import pymongo

2

3

4 class MongoDBHelper:

5 """数据库操作"""

6

7 def \_\_init\_\_(self, collection\_name=None):

8 # 启动mongo

9 self.\_client = pymongo.MongoClient('localhost', 27017)

10 # 使用test数据库

11 self.\_test = self.\_client['test']

12 # 创建指定的集合

13 self.\_name = self.\_test[collection\_name]

14

15 def insert\_item(self, item):

16 """插入数据"""

17 self.\_name.insert\_one(item)

18

19 def find\_item(self):

20 """查询数据"""

21 data = self.\_name.find()

22 return data

23

24

25 def main():

26 mongo = MongoDBHelper('collection')

27 mongo.insert\_item({'a': 1})

28

29

30 if \_\_name\_\_ == '\_\_main\_\_':

31 main()

[复制代码](javascript:void(0);)

douban.py

[复制代码](javascript:void(0);)

1 import logging

2 import random

3 import string

4 import requests

5 import time

6 from collections import deque

7 from urllib import parse

8

9 from settings import User\_Agents

10 from MongDBHelper import MongoDBHelper

11

12

13 class DoubanSpider(object):

14 """豆瓣爬虫"""

15 def \_\_init\_\_(self):

16 # 基本的URL

17 self.base\_url = 'https://movie.douban.com/j/new\_search\_subjects?'

18 self.full\_url = self.base\_url + '{query\_params}'

19 # 从User-Agents中选择一个User-Agent

20 self.headers = {'User-Agent': random.choice(User\_Agents)}

21 # 影视形式(电影, 电视剧,综艺)

22 self.form\_tag = None # 类型

23 self.type\_tag = None # 地区

24 self.countries\_tag = None # 特色

25 self.genres\_tag = None

26 self.sort = 'T' # 排序方式,默认是T,表示热度

27 self.range = 0, 10 # 评分范围

28 self.playable = ''

29 self.unwatched = ''

30 # 连接数据库,集合名为douban\_movies

31 self.db = MongoDBHelper('douban\_movies')

32

33 def get\_query\_parameter(self):

34 """获取用户输入信息"""

35 # 获取tags参数

36 self.form\_tag = input('请输入你想看的影视形式(电影|电视剧|综艺...):')

37 self.type\_tag = input('请输入你想看的影视类型(剧情|爱情|喜剧|科幻...):')

38 self.countries\_tag = input('请输入你想看的影视地区(大陆|美国|香港...):')

39 self.genres\_tag = input('请输入你想看的影视特色(经典|冷门佳片|黑帮...):')

40

41 def get\_default\_query\_parameter(self):

42 """获取默认的查询参数"""

43 # 获取 sort, range, playable, unwatched参数

44 self.range = input('请输入评分范围[0-10]:')

45 self.sort = input('请输入排序顺序(热度:T, 时间:R, 评价:S),三选一:').upper()

46 self.playable = input('请选择是否可播放(默认不可播放):')

47 self.unwatched = input('请选择是否为我没看过(默认是没看过):')

48

49 def encode\_query\_data(self):

50 """对输入信息进行编码处理"""

51 if not (self.form\_tag and self.type\_tag and self.countries\_tag and self.genres\_tag):

52 all\_tags = ''

53 else:

54 all\_tags = [self.form\_tag, self.type\_tag, self.countries\_tag, self.genres\_tag]

55 query\_param = {

56 'sort': self.sort,

57 'range': self.range,

58 'tags': all\_tags,

59 'playable': self.playable,

60 'unwatched': self.unwatched,

61 }

62

63 # string.printable:表示ASCII字符就不用编码了

64 query\_params = parse.urlencode(query\_param, safe=string.printable)

65 # 去除查询参数中无效的字符

66 invalid\_chars = ['(', ')', '[', ']', '+', '\'']

67 for char in invalid\_chars:

68 if char in query\_params:

69 query\_params = query\_params.replace(char, '')

70 # 把查询参数和base\_url组合起来形成完整的url

71 self.full\_url = self.full\_url.format(query\_params=query\_params) + '&start={start}'

72

73 def download\_movies(self, offset):

74 """下载电影信息

75 :param offset: 控制一次请求的影视数量

76 :return resp:请求得到的响应体"""

77 full\_url = self.full\_url.format(start=offset)

78 resp = None

79 try:

80 resp = requests.get(full\_url, headers=self.headers)

81 except Exception as e:

82 # print(resp)

83 logging.error(e)

84 return resp

85

86 def get\_movies(self, resp):

87 """获取电影信息

88 :param resp: 响应体

89 :return movies:爬取到的电影信息"""

90 if resp:

91 if resp.status\_code == 200:

92 # 获取响应文件中的电影数据

93 movies = dict(resp.json()).get('data')

94 if movies:

95 # 获取到电影了,

96 print(movies)

97 return movies

98 else:

99 # 响应结果中没有电影了!

100 # print('已超出范围!')

101 return None

102 else:

103 # 没有获取到电影信息

104 return None

105

106 def save\_movies(self, movies, id):

107 """把请求到的电影保存到数据库中

108 :param movies:提取到的电影信息

109 :param id: 记录每部电影

110 """

111 if not movies:

112 print('save\_movies() error: movies为None!!!')

113 return

114

115 all\_movies = self.find\_movies()

116 if len(all\_movies) == 0:

117 # 数据库中还没有数据,

118 for movie in movies:

119 id += 1

120 movie['\_id'] = id

121 self.db.insert\_item(movie)

122 else:

123 # 保存已经存在数据库中的电影标题

124 titles = []

125 for existed\_movie in all\_movies:

126 # 获取数据库中的电影标题

127 titles.append(existed\_movie.get('title'))

128

129 for movie in movies:

130 # 判断数据库中是否已经存在该电影了

131 if movie.get('title') not in titles:

132 id += 1

133 movie['\_id'] = id

134 # 如果不存在,那就进行插入操作

135 self.db.insert\_item(movie)

136 else:

137 print('save\_movies():该电影"{}"已经在数据库了!!!'.format(movie.get('title')))

138

139 def find\_movies(self):

140 """查询数据库中所有的电影数目

141 :return: 返回数据库中所有的电影

142 """

143 all\_movies = deque()

144 data = self.db.find\_item()

145 for item in data:

146 all\_movies.append(item)

147 return all\_movies

148

149

150 def main():

151 """豆瓣电影爬虫程序入口"""

152 # 1. 初始化工作,设置请求头等

153 spider = DoubanSpider()

154 # 2. 与用户交互,获取用户输入的信息

155 spider.get\_query\_parameter()

156 ret = input('是否需要设置排序方式,评分范围...(Y/N):')

157 if ret.lower() == 'y':

158 spider.get\_default\_query\_parameter()

159 # 3. 对信息进行编码处理,组合成有效的URL

160 spider.encode\_query\_data()

161 id = offset = 0

162 while True:

163 # 4. 下载影视信息

164 reps = spider.download\_movies(offset)

165 # 5.提取下载的信息

166 movies = spider.get\_movies(reps)

167 # 6. 保存数据到MongoDB数据库

168 # spider.save\_movies(movies, id)

169 offset += 20

170 id = offset

171 # 控制访问速速

172 time.sleep(5)

173

174

175 if \_\_name\_\_ == '\_\_main\_\_':

176 main()

[复制代码](javascript:void(0);)

小结:在本次案例中,主要的难点有:查询参数的组合那部分和了解异步加载的原理从而找到真正的URL!查询参数的设置主要用到urlencode()方法,当我们不要把ASCII字符编码的时候,我们要设置safe参数为string.printable,这样只要把一些非ASCII字符编码就好了,同样quote()也是用来编码的,也有safe参数;那么本例中为什么要使用urlencode()方法呢?主要是通过观察URL是key=value的形式,所以才选用它!当我们把数据插入到数据库中时,如果是有相同的名字的电影,我们就不插入,这样也是处于对性能的考虑,合理利用资源!