链滴

Python3 网络爬虫实战：22、使用 Urllib：解析链接

作者：zhaolixiang

原文链接：https://ld246.com/article/1567222367451

来源网站：链滴

Urllib 库里还提供了 parse 这个模块，它定义了处理 URL 的标准接口，例如实现 URL 各部分的抽取合并以及链接转换。它支持如下协议的 URL 处理：file、ftp、gopher、hdl、http、https、imap、ailto、 mms、news、nntp、prospero、rsync、rtsp、rtspu、sftp、shttp、 sip、sips、snews、sn、svn+ssh、telnet、wais，本节我们介绍一下该模块中常用的方法来感受一下它的便捷之处。

1. urlparse()

urlparse() 方法可以实现 URL 的识别和分段，我们先用一个实例来感受一下：

```
from urllib.parse import urlparse

result = urlparse('http://www.baidu.com/index.html;user?id=5#comment')
print(type(result), result)
```

在这里我们利用了 urlparse() 方法进行了一个 URL 的解析，首先输出了解析结果的类型，然后将结也输出出来。

运行结果：

```
<class 'urllib.parse.ParseResult'>
ParseResult(scheme='http', netloc='www.baidu.com', path='/index.html', params='user', quer='id=5', fragment='comment')
```

观察可以看到，返回结果是一个 ParseResult 类型的对象，它包含了六个部分，分别是 scheme、netoc、path、params、query、fragment。

观察一下实例的URL：

```
http://www.baidu.com/index.html;user?id=5#comment
```

urlparse() 方法将其拆分成了六部分，大体观察可以发现，解析时有特定的分隔符，比如 :// 前面的是 scheme，代表协议，第一个 / 前面便是 netloc，即域名，分号 ; 前面是 params，代表参数。

所以可以得出一个标准的链接格式如下：

```
scheme://netloc/path;parameters?query#fragment
```

一个标准的 URL 都会符合这个规则，利用 urlparse() 方法我们可以将它解析拆分开来。

除了这种最基本的解析方式，urlopen() 方法还有其他配置吗？接下来看一下它的 API 用法：

```
urllib.parse.urlparse(urlstring, scheme='', allow_fragments=True)
```

可以看到它有三个参数：

- urlstring，是必填项，即待解析的 URL。
- scheme，是默认的协议（比如http、https等），假如这个链接没有带协议信息，会将这个作为默的协议。

我们用一个实例感受一下：

```
from urllib.parse import urlparse

result = urlparse('www.baidu.com/index.html;user?id=5#comment', scheme='https')
```

```
print(result)
```

运行结果：

```
ParseResult(scheme='https', netloc='', path='www.baidu.com/index.html', params='user', que
y='id=5', fragment='comment')
```

可以发现，我们提供的 URL 没有包含最前面的 scheme 信息，但是通过指定默认的 scheme 参数，回的结果是 https。

假设我们带上了 scheme 呢?

```
result = urlparse('http://www.baidu.com/index.html;user?id=5#comment', scheme='https')
```

结果如下：

```
ParseResult(scheme='http', netloc='www.baidu.com', path='/index.html', params='user', quer
='id=5', fragment='comment')
```

可见 scheme 参数只有在 URL 中不包含 scheme 信息时才会生效，如果 URL 中有 scheme 信息，就返回解析出的 scheme。

● allow_fragments，即是否忽略 fragment，如果它被设置为 False，fragment 部分就会被忽略，会被解析为 path、parameters 或者 query 的一部分，fragment 部分为空。

下面我们用一个实例感受一下：

```
from urllib.parse import urlparse

result = urlparse('http://www.baidu.com/index.html;user?id=5#comment', allow_fragments=F
lse)
print(result)
```

运行结果：

```
ParseResult(scheme='http', netloc='www.baidu.com', path='/index.html', params='user', quer
='id=5#comment', fragment='')
```

假设 URL 中不包含 parameters 和 query 呢?

再来一个实例看下：

```
from urllib.parse import urlparse

result = urlparse('http://www.baidu.com/index.html#comment', allow_fragments=False)
print(result)
```

运行结果：

```
ParseResult(scheme='http', netloc='www.baidu.com', path='/index.html#comment', params=''
 query='', fragment='')
```

可以发现当 URL 中不包含 params 和 query 时， fragment 便会被解析为 path 的一部分。

返回结果 ParseResult 实际上是一个元组，我们可以用索引顺序来获取，也可以用属性名称获取，实

如下：

```
from urllib.parse import urlparse

result = urlparse('http://www.baidu.com/index.html#comment', allow_fragments=False)
print(result.scheme, result[0], result.netloc, result[1], sep='\n')
```

在这里我们分别用索引和属性名获取了 scheme 和 netloc，运行结果如下：

```
http
http
www.baidu.com
www.baidu.com
```

可以发现二者结果是一致的，两种方法都可以成功获取。

2. urlunparse()

有了 urlparse() 那相应地就有了它的对立方法 urlunparse()。

它接受的参数是一个可迭代对象，但是它的长度必须是 6，否则会抛出参数数量不足或者过多的问题。

先用一个实例感受一下：

```
from urllib.parse import urlunparse

data = ['http', 'www.baidu.com', 'index.html', 'user', 'a=6', 'comment']
print(urlunparse(data))
```

参数 data 用了列表类型，当然你也可以用其他的类型如元组或者特定的数据结构。

运行结果如下：

```
http://www.baidu.com/index.html;user?a=6#comment
```

这样我们就成功实现了 URL 的构造。

3. urlsplit()

这个和 urlparse() 方法非常相似，只不过它不会单独解析 parameters 这一部分，只返回五个结果。面例子中的 parameters 会合并到 path中，用一个实例感受一下：

```
from urllib.parse import urlsplit

result = urlsplit('http://www.baidu.com/index.html;user?id=5#comment')
print(result)
```

运行结果：

```
SplitResult(scheme='http', netloc='www.baidu.com', path='/index.html;user', query='id=5', fr
gment='comment')
```

可以发现返回结果是 SplitResult，其实也是一个元组类型，可以用属性获取值也可以用索引来获取，例如下：

```
from urllib.parse import urlsplit

result = urlsplit('http://www.baidu.com/index.html;user?id=5#comment')
print(result.scheme, result[0])
```

运行结果：

```
http http
```

4. urlunsplit()

与 urlunparse() 类似，也是将链接的各个部分组合成完整链接的方法，传入的也是一个可迭代对象，如列表、元组等等，唯一的区别是，长度必须为 5。

用一个实例来感受一下：

```
from urllib.parse import urlunsplit

data = ['http', 'www.baidu.com', 'index.html', 'a=6', 'comment']
print(urlunsplit(data))
```

运行结果：

```
http://www.baidu.com/index.html?a=6#comment
```

同样可以完成链接的拼接生成。

5. urljoin()

有了 urlunparse() 和 urlunsplit() 方法，我们可以完成链接的合并，不过前提必须要有特定长度的对，链接的每一部分都要清晰分开。

生成链接还有另一个方法，利用 urljoin() 方法我们可以提供一个 base_url（基础链接），新的链接为第二个参数，方法会分析 base_url 的 scheme、netloc、path 这三个内容对新链接缺失的部分进补充，作为结果返回。

我们用几个实例来感受一下：

```
from urllib.parse import urljoin

print(urljoin('http://www.baidu.com', 'FAQ.html'))
print(urljoin('http://www.baidu.com', 'https://cuiqingcai.com/FAQ.html'))
print(urljoin('http://www.baidu.com/about.html', 'https://cuiqingcai.com/FAQ.html'))
print(urljoin('http://www.baidu.com/about.html', 'https://cuiqingcai.com/FAQ.html?question=
'))
print(urljoin('http://www.baidu.com?wd=abc', 'https://cuiqingcai.com/index.php'))
print(urljoin('http://www.baidu.com', '?category=2#comment'))
print(urljoin('www.baidu.com', '?category=2#comment'))
print(urljoin('www.baidu.com#comment', '?category=2'))
```

运行结果：

```
http://www.baidu.com/FAQ.html
https://cuiqingcai.com/FAQ.html
```

```
https://cuiqingcai.com/FAQ.html
https://cuiqingcai.com/FAQ.html?question=2
https://cuiqingcai.com/index.php
http://www.baidu.com?category=2#comment
www.baidu.com?category=2#comment
www.baidu.com?category=2
```

可以发现，base_url 提供了三项内容，scheme、netloc、path，如果这三项在新的链接里面不存在那么就予以补充，如果新的链接存在，那么就使用新的链接的部分。base_url 中的 parameters、query、fragments 是不起作用的。

通过如上的函数，我们可以轻松地实现链接的解析，拼合与生成。

6. urlencode()

我们再介绍一个常用的 urlencode() 方法，它在构造 GET 请求参数的时候非常有用，我们用实例感受一下：

```
from urllib.parse import urlencode

params = {
    'name': 'germey',
    'age': 22
}
base_url = 'http://www.baidu.com?'
url = base_url + urlencode(params)
print(url)
```

我们首先声明了一个字典，将参数表示出来，然后调用 urlencode() 方法将其序列化为 URL 标准 GET 请求参数。

运行结果：

```
http://www.baidu.com?name=germey&age=22
```

可以看到参数就成功由字典类型转化为 GET 请求参数了。

这个方法非常常用，有时为了更加方便地构造参数，我们会事先用字典来表示，要转化为 URL 的参时只需要调用该方法即可。

7. parse_qs()

有了序列化必然就有反序列化，如果我们有一串 GET 请求参数，我们利用 parse_qs() 方法就可以将转回字典，我们用一个实例感受一下：

```
from urllib.parse import parse_qs

query = 'name=germey&age=22'
print(parse_qs(query))
```

运行结果：

```
{'name': ['germey'], 'age': ['22']}
```

可以看到这样就成功转回为字典类型了。

8. parse_qsl()

另外还有一个 parse_qsl() 方法可以将参数转化为元组组成的列表，实例如下：

```
from urllib.parse import parse_qsl

query = 'name=germey&age=22'
print(parse_qsl(query))
```

运行结果：

```
[('name', 'germey'), ('age', '22')]
```

可以看到运行结果是一个列表，列表的每一个元素都是一个元组，元组的第一个内容是参数名，第二内容是参数值。

9. quote()

quote() 方法可以将内容转化为 URL 编码的格式，有时候 URL 中带有中文参数的时候可能导致乱码问题，所以我们可以用这个方法将中文字符转化为 URL 编码，实例如下：

```
from urllib.parse import quote

keyword = '壁纸'
url = 'https://www.baidu.com/s?wd=' + quote(keyword)
print(url
```

在这里我们声明了一个中文的搜索文字，然后用 quote() 方法对其进行 URL 编码，最后得到的结果下：

```
https://www.baidu.com/s?wd=%E5%A3%81%E7%BA%B8
```

这样我们就可以成功实现URL编码的转换。

10. unquote()

有了 quote() 方法当然还有 unquote() 方法，它可以进行 URL 解码，实例如下：

```
from urllib.parse import unquote

url = 'https://www.baidu.com/s?wd=%E5%A3%81%E7%BA%B8'
print(unquote(url))
```

这是上面得到的 URL 编码后的结果，我们在这里利用 unquote() 方法进行还原，结果如下：

```
https://www.baidu.com/s?wd=壁纸
```

可以看到利用 unquote() 方法可以方便地实现解码。

11. 结语

本节介绍了 parse 模块的一些常用 URL 处理方法，有了这些方法我们可以方便地实现 URL 的解析和造，建议熟练掌握。