概述

HTML是的HyperText Markup Language缩写，翻译为： 超文本标记语言，标准通用标记语言下的一个应用。

"超文本"就是指页面内可以包含图片、链接，甚至音乐、程序等非文字元素。

超文本标记语言的结构包括"头"部分（英语：Head）、和"主体"部分（英语：Body），其中"头"部提供关于网页的信息，"主体"部分提供网页的具体内容。

我们看一个基本的html的结构：

```
<html>
    <head>
        <title>我是标题</title>
    </head>
    <body>
        我是主体内容
    </body>
</html>
```

python html解析类:HTMLParser

HTML操作是编程中很重要的一块，下面介绍下Python3.x中的html.parser中的HTMLParser类。

> *HTMLParser的定义*
>
> class html.parser.HTMLParser(*, convert_charrefs=True)
>
> - HTMLParser主要是用来解析HTML文件（包括HTML中无效的标记）
> - 参数convert_charrefs表示是否将所有的字符引用自动转化为Unicode形式，Python3.5以后默认是True
> - HTMLParser可以接收相应的HTML内容，并进行解析，遇到HTML的标签会自动调用相应的handler（处理方法）来处理，用户需要自己创建相应的子类来继承HTMLParser，并且复写相应的handler方法
> - HTMLParser不会检查开始标签和结束标签是否是一对

下面我们看一看HTMLParser常用的方法

> *HTMLParser常用方法*
>
> - HTMLParser.feed(data)：接收一个字符串类型的HTML内容，并进行解析
> - HTMLParser.close()：当遇到文件结束标签后进行的处理方法。如果子类要复写该方法，需要首先调用HTMLParser累的close()
> - HTMLParser.reset():重置HTMLParser实例，该方法会丢掉未处理的html内容

- HTMLParser.getpos()：返回当前行和相应的偏移量
- HTMLParser.handle_starttag(tag, attrs)：对开始标签的处理方法。例如，参数tag指的是div，attrs指的是一个（name,Value)的列表,这里指(id, main)
- HTMLParser.handle_endtag(tag)：对结束标签的处理方法。例如，参数tag指的是div
- HTMLParser.handle_data(data)：对标签之间的数据的处理方法。test,data指的是"test"
- HTMLParser.handle_comment(data)：对HTML中注释的处理方法。

示例演示

下面我看使用HTMLParser来博客网首页的所有a（链接）节点进行解析出来。

```
#-*- coding:utf-8 -*-

__author__ = "苦叶子"

from html.parser import HTMLParser
import http.client

class BlogHTMLParser(HTMLParser):
    data = []
    data_key = ""

    def __init__(self):
        HTMLParser.__init__(self)
        self.is_a = False

    def handle_starttag(self, tag, attrs):
        # 处理开始为a的标签
        if tag == "a":
            self.is_a = True
            for name,value in attrs:
                if name == "href":
                    # 提取a的href属性值
                    self.data_key = value

    def handle_data(self, data):
        # 处理结束为a的标签
        if self.is_a and self.lasttag == "a":
            # 将a标签的href属性值作为key， a的文本作为data构建字典
            self.data.append({self.data_key : data})

    def handle_endtag(self, tag):
        # 处理a结束标签
        if self.is_a and self.lasttag == "a":
            self.is_a = False
```

```
    def get_data(self):
        # 返回所有从a中提取到的目标数据
        return self.data

if __name__ == "__main__":
    print("python HTML解析实例")

    print("访问博客网，获取首页html源码")

    # 构建博客园链接
    conn = http.client.HTTPSConnection("www.cnblogs.com")

    # 获取博客园首页html源码
    conn.request("GET", "/")
    r1 = conn.getresponse()
    data = r1.read().decode(encoding="utf-8")
    # print(data)

    # 解析博客园首页html源码，提取所有a的href和文本数据
    blogHtmlParser = BlogHTMLParser()
    blogHtmlParser.feed(data)
    links = blogHtmlParser.get_data()

    # 打印提取的结果
    print(links)
```

说明下上述代码运行的流程：

1. 运行__init__初始化实例
2. 执行handle_starttag
3. 执行handle_data
4. handle_endtag
5. 重复2、3、4直至把所有的a提取完毕

小结

本示例可以说是一个最最最简单的爬虫示例了，为什么要去掌握这个呢？我想可能会有以下简单的应用场景：

- 爬取目标html元素，自动构建xpath或css定位，用于UI级自动化测试
- 爬取目标URL下所有的链接或form表单相关资源，获取可能的接口测试目标
- 爬取感兴趣的资源，只要你感兴趣就好
- 其他应用场景

扫一扫关注微信公众号：