

Python爬虫基本认知

Python爬虫基本认知

- 一、课前准备
- 二、课堂主题
- 三、课堂目标
- 四、知识要点
 - 1、爬虫基本认知
 - 1.1 爬虫概念
 - 1.2 爬虫的基本流程
 - 2、网页基本认知
 - 2.1 浏览器工作原理
 - 2.2 Request
 - 2.3 Response
 - 2.4 网页数据类型
 - 2.5 网页的解析方式
 - 3、网页构成
 - 3.1 HTML
 - 3.2 CSS
 - 3.3 JS
 - 4、XPath的基本使用
 - 4.1 简介
 - 4.2 XPath语法
 - 4.3 基础使用
 - 4.4 XPath练习
- 五、总结

一、课前准备

1. 根据课程大纲复习Python语法；
2. 根据课程大纲提前预习Python爬虫；

二、课堂主题

本小节主要学习Python网页爬虫的基本流程及操作。

三、课堂目标

1. 理解Python网页爬虫流程及顺序。
2. 了解网页的构造及解析。
3. 使用XPath爬取网页数据。

四、知识要点

1、爬虫基本认知

1.1 爬虫概念

请求网站并提取数据的自动化程序。

1.2 爬虫的基本流程

- 发起请求

通过HTTP库目标站点发起请求，即发送一个Request，请求可以包含额外的headers等信息，等待服务器响应。

- 获取响应内容

如果服务器能够正常响应，会得到一个Response，Response的内容便是所要获取的页面内容，类型可能有HTML，Json字符串，二进制数据（如图片视频）等类型。

- 解析内容

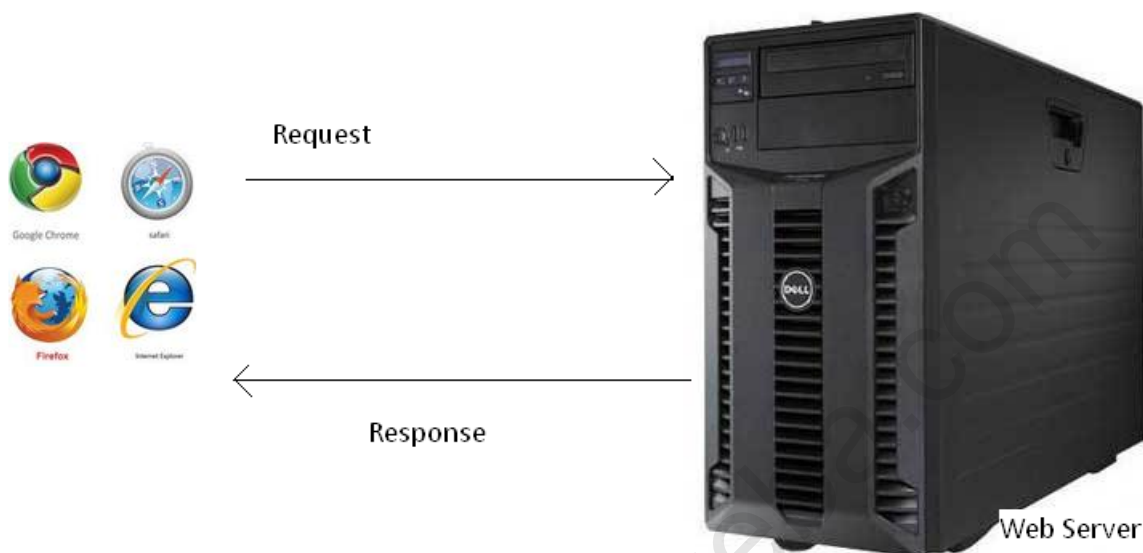
得到的内容可能是HTML，可以用正则表达式，网页解析库进行解析。可能是Json，可以直接转为Json对象解析，可能是二进制数据，可以做保存或者进一步的处理。

- 保存数据

保存形式多样，可以存为文本，也可以保存至数据库，或者保存特定格式的文件。

2、网页基本认知

2.1 浏览器工作原理



- 浏览器发送消息给该网址所在的服务器，这个过程叫做HTTP **Request**。
- 服务器收到浏览器发送的消息后，能够根据浏览器发送消息的内容，做响应处理，然后把消息回传给浏览器。这个过程叫做HTTP **Response**。
- 浏览器收到服务器的Response信息后，会对信息进行响应处理，然后展示。

查看浏览器的请求和响应：



Elements Console Sources Network Performance Application Security Audits					
View: [List Icon] [Filter Icon] [Group by frame] [Preserve log] [Disable cache] [Offline] No throttling					
[Hide data URLs] [All] XHR JS CSS Img Media Font Doc WS Manifest Other					
10 ms	20 ms	30 ms	40 ms	50 ms	60 ms
70 ms	80 ms	90 ms			
	Status	Type	Initiator	Size	T
gif?pid=201&pj=www&fm=behs&tab=bdlogo&path=https...=fa24207abed427e35e9...	200	gif	all_async_search_7935d39.js:7	289 B	6
gif?pid=201&pj=www&fm=tab&tab=felogo&rsv_platfor...=fa24207abed427e35e937...	200	gif	all_async_search_7935d39.js:7	289 B	6
gif?pid=201&pj=www&fm=behs&tab=bdlogo&path=https...=fa24207abed427e35e9...	200	gif	all_async_search_7935d39.js:7	289 B	2
gif?pid=201&pj=www&fm=tab&tab=felogo&rsv_platfor...=fa24207abed427e35e937...	200	gif	all_async_search_7935d39.js:7	289 B	2
gif?pid=201&pj=www&fm=behs&tab=bdlogo&path=https...=fa24207abed427e35e9...	200	gif	all_async_search_7935d39.js:7	289 B	5
gif?pid=201&ni=www&fm=tab&tab=felogo&rsv_platfor...=fa24207abed427e35e937...	200	nif	all_async_search_7935d39.js:7	289 B	1

2.2 Request

Request是请求,在浏览器输入地址,回车,就是一个请求。

- 请求方式

主要有get、post两种类型，另外还有HEAD、PUT、DELETE、OPTIONS等。

- 请求的URL

URL 全称统一资源定位符，如一个网页的文档、一张图片、一个视频等都可以用URL来确定。

- 请求头

包含请求时的头部信息，如User-Agent、host、Cookies等信息

- 请求体

请求时额外携带的数据如表单提交时的表单数据。

2.3 Response

Response是响应,服务器根据请求,返回数据到浏览器显示,就是一个响应。

- 响应状态

有多种响应状态，如200代表成功、301跳转、404找不到页面、502服务器错误。

- 响应头

如内容类型、内容长度、服务器信息、设置Cookie等等。

- 响应体

最主要的部分，包含了请求资源的内容，如网页HTML、图片二进制数据等。

例如：

```
1 # 导入网络请求模块 (该模块需要使用pip install requests 安装)
2 import requests
3
4 # 创建请求头
5 headers = {
6     "User-Agent": 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36
7     (KHTML, like Gecko) Chrome/74.0.3729.131 Safari/537.36'
8 }
9 # 发送网络请求
10 response = requests.get(url='https://www.baidu.com', headers = headers)
11
12 # 获取请求的内容
13 print(response.content.decode('utf-8'))
14 print(response.text)
15
16 # 获取响应头
17 print(response.headers)
18
19 # 状态码
20 print(response.status_code)
```

2.4 网页数据类型

- 网页文本

如HTML文档、Json格式的文本等。

- 图片

获取到的是二进制文件，保存为图片格式。

- 视频

获取到的是二进制文件，保存为视频格式。

例如：获取图片

```
1 import requests
2
3 # 创建请求头
4 headers = {
5     "User-Agent": 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36
6     (KHTML, like Gecko) Chrome/74.0.3729.131 Safari/537.36'
7 }
8 # 发送网络请求
9 response = requests.get(url='https://www.baidu.com/img/bd_logo1.png', headers =
10 headers)
11
12 # 将获取的二进制内容进行保存
13 with open('./baidu.png', 'wb') as f:
14     f.write(response.content)
```

2.5 网页的解析方式

- 直接处理

例如：请求回来的就是一段字符串，我们可以简单处理后直接保存

- Json解析

例如：Ajax请求回来的一般都是json格式的数据，我们就需要从json中拿出我们想要的数据。

- 正则表达式

- BeautifulSoup

- PyQuery

- XPath

解析库的选择可以根据爬取的网页具体情况酌情选择。

3、网页构成

3.1 HTML

超级文本标记语言 (HyperText Markup Language) 是标准通用标记语言下的一个应用，也是一种规范和标准，它通过标记符号来标记要显示的网页中的各个部分。网页文件本身是一种文本文件，通过在文本文件中添加标记符，可以告诉浏览器如何显示其中的内容（如：文字如何处理，画面如何安排，图片如何显示等）。

1.根元素

<code><doctype></code>	定义文档类型。
<code><html></code>	定义 HTML 文档。

2.元数据元素

<code><head></code>	定义关于文档的信息。
<code><meta></code>	定义关于 HTML 文档的元数据。
<code><link></code>	定义文档与外部资源之间的关系，一般用于引入样式表。
<code><base></code>	定义页面上所有链接的默认地址或默认目标。
<code><title></code>	定义文档标题。
<code><style></code>	定义文档的样式信息。

3.脚本元素

<code><script></code>	定义客户端脚本。
<code><noscript></code>	定义当浏览器不支持脚本的时候所显示的内容

4.块元素

<code><body></code>	定义文档的主体。
<code><h1>、<h2>...<h6></code>	定义文档标题。
<code><p></code>	定义文档段落。
<code><blockquote></code>	定义块引用。
<code>、、<dl></code>	定义列表。
<code><table></code>	定义表格。

5.列表元素

无序列表

<code></code>	定义无序的列表。
<code></code>	定义列表项。

有序列表

	定义有序的列表。
	定义列表项。

定义列表

<dl>	定义定义列表。
<dt>	定义定义术语。
<dd>	定义定义描述。

6.表格元素

<table>	定义表格。
<thead>	定义表格的页眉。
<tbody>	定义表格的主体。
<tfoot>	定义表格的页脚。
<th>	定义表格的表头行。
<tr>	定义表格的行。
<td>	定义表格单元。

7.文本元素

文本格式化元素

	定义着重文字。
	定义加重语气。
<sup>	定义上标字。
<sub>	定义下标字。
<ins>	定义插入字。
	定义删除字。
	定义粗体文本。
<i>	定义斜体文本。
<big>	定义大号字。
<small>	定义小号字。

8.链接与图像

<code><a></code>	定义超链接
<code></code>	定义图像。
<code><map></code>	定义图像地图。
<code><area></code>	定义图像地图中的可点击区域。

9. <div> 和

<code><div></code>	定义文档中的分区或节 (division/section) 。
<code></code>	定义 span, 用来组合文档中的行内元素。

10.表单元素

<code><form></code>	定义供用户输入的表单。
<code><input></code>	定义输入域。
<code><textarea></code>	定义文本域 (一个多行的输入控件)。
<code><table></code>	定义一个控制的标签。
<code><select></code>	定义一个选择列表。
<code><option></code>	定义下拉列表中的选项。
<code><optgroup></code>	定义选项组。
<code><button></code>	定义一个按钮。
<code><fieldset></code>	定义域。
<code><legend></code>	定义域的标题。

3.2 CSS

CSS是一种定义样式结构如字体、颜色、位置等的语言，用于描述网页上的信息格式化和现实的方式。CSS样式可以直接存储于HTML网页或者单独的样式单文件。

内联方式

样式定义在单个的 HTML元素中

内部样式表

样式定义在 HTML 页的头元素中

外部样式表

将样式定义在一个外部的 CSS 文件中 (.css 文件)

由 HTML 页面引用样式表文件

```

▶ <div id="u">...</div>
  <p class="s-skin-lm s-isindex-wrap"></p>
</div>

▶ <style index="newi" type="text/css">...</style>
▶ <style type="text/css" index="head">...</style>
▶ <style type="text/css" index="common">...</style>
▶ <style type="text/css" index="weather">...</style>

```

3.3 JS

JS, 全称JavaScript, 是一种属于网络的脚本语言, 已经被广泛用于Web应用开发, 常用来为网页添加各式各样的动态功能, 为用户提供更流畅美观的浏览效果。通常JavaScript脚本是通过嵌入在HTML中来实现自身的功能的。

1. js代码能直接嵌入网页的任何地方, 通常放在 <head></head> 中, 但会要求用户把所有js代码都下载解析执行以后在呈现内容;
2. 可以把代码单独的放在一个.js文件中, 同时, 多个页面可以引用同一份.js文件;
3. 页面中可以多次编写 <script></script>, 但是浏览器会按照顺序执行这些JavaScript代码;
4. JavaScript中每个语句以 ; 结尾 每个语句块以 {.....} 注释用//或/* */
5. 所有类型都是定义变量都是用 var, 如果一个变量没有通过var申明就被使用, 那么该变量就自动被申明为全局变量;

hello.html

```

1  <html>
2    <head>
3      <title>js</title>
4      <script type="text/javascript">
5        console.log("Hello javascript")
6      </script>
7    </head>
8    <body>
9    </body>
10 </html>

```

外部样式表

test.js

```
1 alert('Hello JS')
```

test.html

```
1 <html>
2
3   <head>
4     <title>js</title>
5     <script type="text/javascript" src='test.js'></script>
6   </head>
7   <body>
8   </body>
9 </html>
```

4、XPath的基本使用

4.1 简介

XPath 是一门在 XML 文档中查找信息的语言。使用路径表达式来选取 XML 文档中的节点或者节点集。

注意： xpath速度比较快，是爬虫在网页定位中的较优选择，但是很多网页前端代码混乱难以定位。

安装: `pip install lxml`

4.2 XPath语法

[xpath的更多语法](#)

```
1 '''
2 表达式      描述
3 nodename    选取此节点的所有子节点。
4 /           从根节点选取。路径的连接符
5 //          从匹配选择的当前节点选择文档中的节点，而不考虑它们的位置。
6 .           选取当前节点。
7 ..          选取当前节点的父节点。
8 @           选取属性。
9 '''
10
11 '''
12 通配符      描述
13 *           匹配任何元素节点。
14 @*          匹配任何属性节点。
15 node()      匹配任何类型的节点。
16 '''
```

4.3 基础使用

```
1 # 导入lxml的etree库
2 from lxml import etree
3
4 data_str = ""
```

```

5         <div>
6             <ul>
7                 <li class="item-0"><a href="link1.html">first item</a></li>
8                 <li class="item-1"><a href="link2.html">second item</a></li>
9                 <li class="item-inactive"><a href="link3.html">third item</a></li>
10                <li class="item-1"><a href="link4.html">fourth item</a></li>
11                <li class="item-0"><a href="link5.html">fifth item</a>
12            </ul>
13        </div>
14        """
15        # 注意: 该数据中缺少了一个li标签的闭合标签
16
17        # 利用etree.HTML可以将字符串或者bytes转化为Element python对象, 这个对象具有XPath的方法
18        html = etree.HTML(data_str)
19        # print(html)
20
21        # etree.tostring(html)可以自动修正HTML代码, 补全了缺胳膊少腿的标签
22        # 使用为了观察修改以后的html样子, 根据修改后的HTML去写XPath
23        result = etree.tostring(html)
24        print(result.decode("utf-8"))
25
26        # 获取class = item-1 的 a标签的href属性
27        result = html.xpath('//li[@class="item-1"]/a/@href')
28        print(result)

```

4.4 XPath练习

```

1  import requests
2  from lxml import etree
3  tiebaName = 'lol'
4  base_url = 'https://tieba.baidu.com/f?kw='+tiebaName+'&ie=utf-8&pn={}'
5  headers = {
6      'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36
7      (KHTML, like Gecko) Chrome/74.0.3729.131 Safari/537.36'
8  }
9  # 构建请求连接
10 url_list = [] # 存储连接的列表
11 for i in range(1): # 初始为第一页
12     url_list.append(base_url.format(i*50))
13
14 # 发送网络请求并得到相应内容
15 response = requests.get(url=url_list[0], headers = headers)
16
17 result_str = response.content.decode('utf-8')
18 # 由于爬取的内容被注释了, 所以将注释去掉
19 result_str = result_str.replace('<!--', '').replace('-->', '')
20
21 html = etree.HTML(result_str)
22
23 # 获取每一页的连接

```

```

24 links = html.xpath('//li[@class=" j_thread_list
    clearfix"]//div[@class="threadlist_title pull_left j_th_tit
    "]//a[@rel="nofollow"]/@href')
25 # 拼接成完整的连接
26 links = ['https://tieba.baidu.com/{}'.format(i) for i in links]
27
28 # 获取文字
29 texts = html.xpath('//li[@class=" j_thread_list
    clearfix"]//div[@class="threadlist_title pull_left j_th_tit
    "]//a[@rel="nofollow"]/text()')

```

五、总结

1. 掌握网页的基本构成及解析方式。
2. 掌握使用XPath爬取网页数据。