数据采集专题

金融科技协会

2020-12-30

框架

1. HTTP协议
2. HTML初步
3. XPath简介
4. Requests使用(By Demos)
    ① 选课查询
    ② 视频下载
    ③ 树洞采集

一、HTTP协议

1. 网络连接过程
2. HTTP协议及其特点
3. 基于HTTP协议的网络连接

1.1 网络连接过程

打开浏览器→地址栏输入网址→回车→浏览器显示相应网址

基于某种网络连接协议（如HTTP协议）、
向指定的服务器（IP地址+端口号）、
对某个资源进行请求（如index.html这个文件）；
服务器回应请求——返回一份HTML文档、
浏览器对该文档进行渲染——显示出来，成为我们看到的漂亮的网页。

URL(Uniform Resource Locator, 统一资源定位器)
&
URI(Uniform Resource Identifier, 统一资源标志符)

1.2 HTTP协议及其特点(1)

HTTP协议(HyperText Transfer Protocol)，全称“超文本传输协议”

↓ 什么是“超文本”？

“超文本”：即“可超链接文本”
该文档可以指向其他位置：当前文档中、局域网中的其他文档、全网的任何位置的文档……
这些文档组成了一个杂乱的信息网。

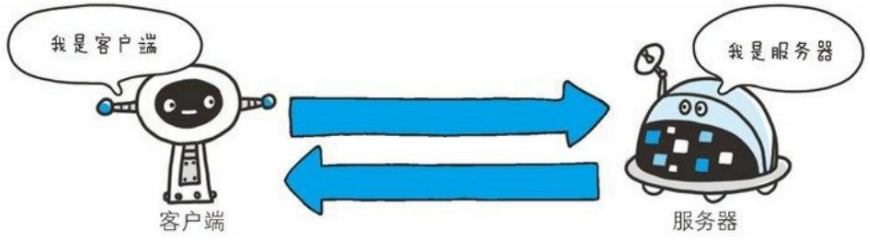
↓ 怎么实现“超文本”？

HTML(HyperText Markup Language)

↓ 传输“超文本”

HTTP协议就是一套关于HTML超文本的传输规定，用于服务器传输超文本到本地计算机。它包括了：客户端和服务端的数据传输格式的规定等等……

1.2 HTTP协议及其特点(2)

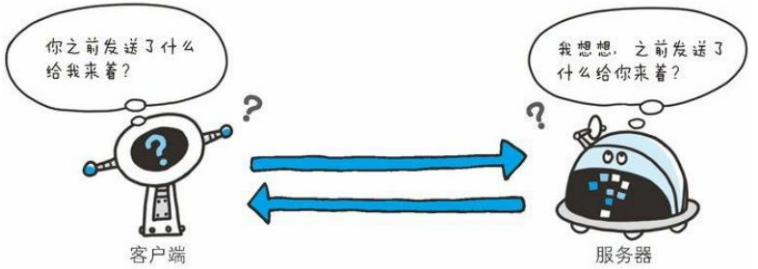


HTTP协议下，两台通信的计算机，必然是一台为客户端，另一台为服务器端：请求发起者/响应接收者为客户端，请求接收者/响应发出者为服务器端。

HTTP协议采用请求-响应（Request-Response）模型，请求永远由客户端发起，响应永远由服务器端回送。言外之意：服务器端不会自己给客户端发生响应。

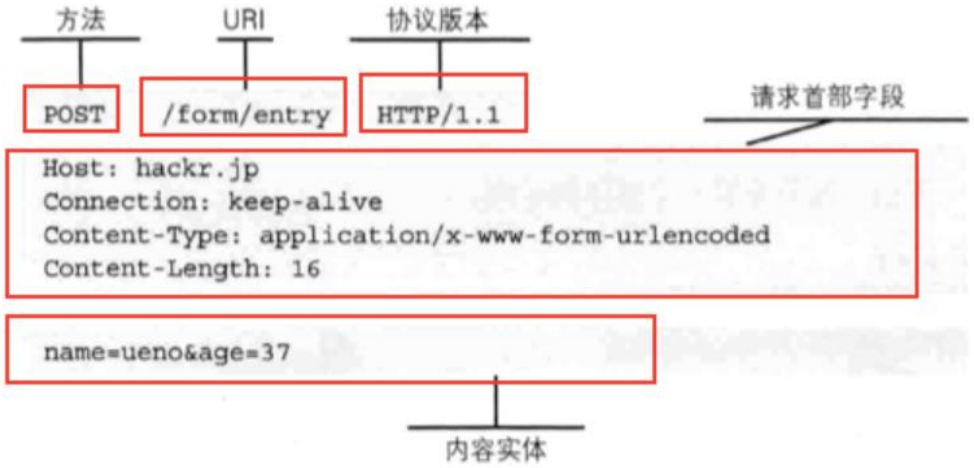
~~此处找不到合适的图（真是逼死强迫症）~~

HTTP是无连接的协议：连接不具有持久性，每次连接处理一个请求，服务器端返回资源，客户端接收到资源后，连接断开。



HTTP是无状态的协议：HTTP协议本身规定，通信双方均不保存此前的通信状态，是一种即无状态(stateless) 协议——But，登录状态的保存？Cookie

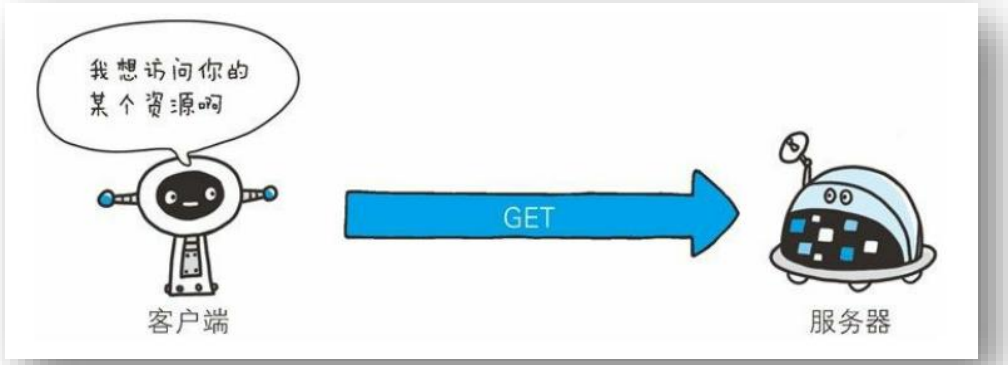
1.3 基于HTTP协议的网络连接(1, 请求报文)



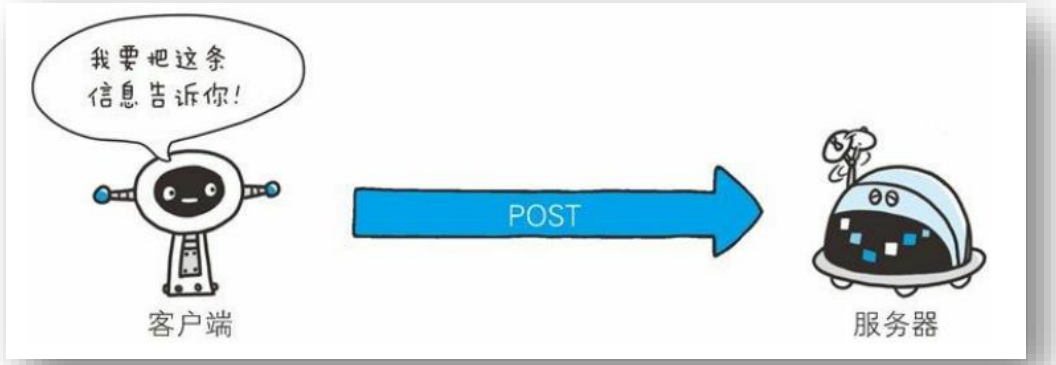
由客户端发出的关于请求的信息称为**请求报文**，由三部分构成：

- 请求行：Request line
  - 请求方法：GET，POST……
  - 请求的URI：指明请求访问的资源对象
  - 协议版本
- 请求头：Request headers。以下列出几个常见的：
  - Host：指定请求资源的Internet主机和端口号
  - User-Agent：请求发出的用户信息，包括使用的浏览器的基本信息，反爬第一道防线
  - Accept：指定客户端接受哪些类型的信息，如text/html表示客户端希望拿到html文本
  - Accept-Language：指定客户端接受哪种语言，不设置的话服务器默认任何语言都可以
  - Accept-Encoding：指定客户端接受哪种内容编码，爬虫设置不当可能返回乱码
- 请求正文：Request body：
  - 储存客户端向服务器端传递的数据
  - 如账号密码……

1.3 基于HTTP协议的网络连接(2, 请求方法)



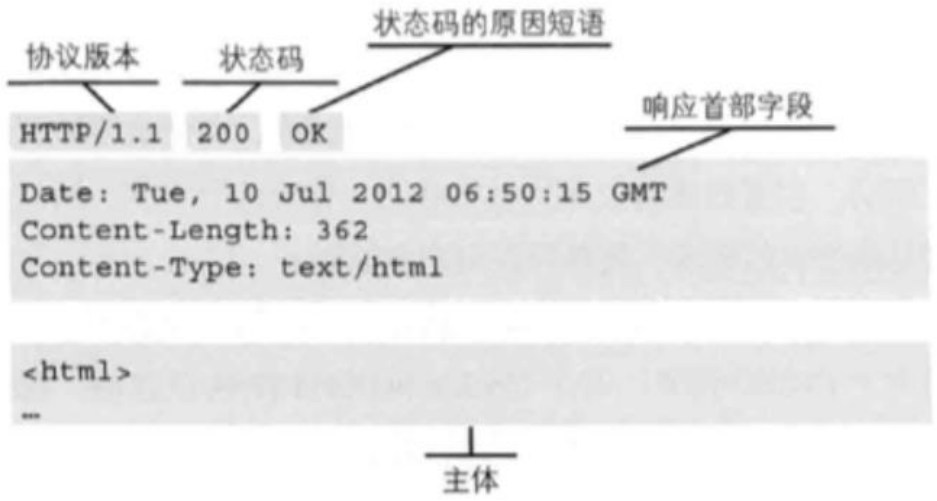
GET方法（获取资源）
“俺要啥啥啥，快给我！”



POST方法（提交信息）
“我告诉你那啥，密码是…”

GET方法也可以提交数据用以更好地描述自己想要什么或作为获取数据的验证，但这种提交数据不是GET的主要目的——因为GET方法传递数据的方式是把数据直接放在URL中：
http://wenshu.court.gov.cn/content/content?DocID=5e74db08-3b47-46d2-8f72-0522db986fba
（被偷窥？！）

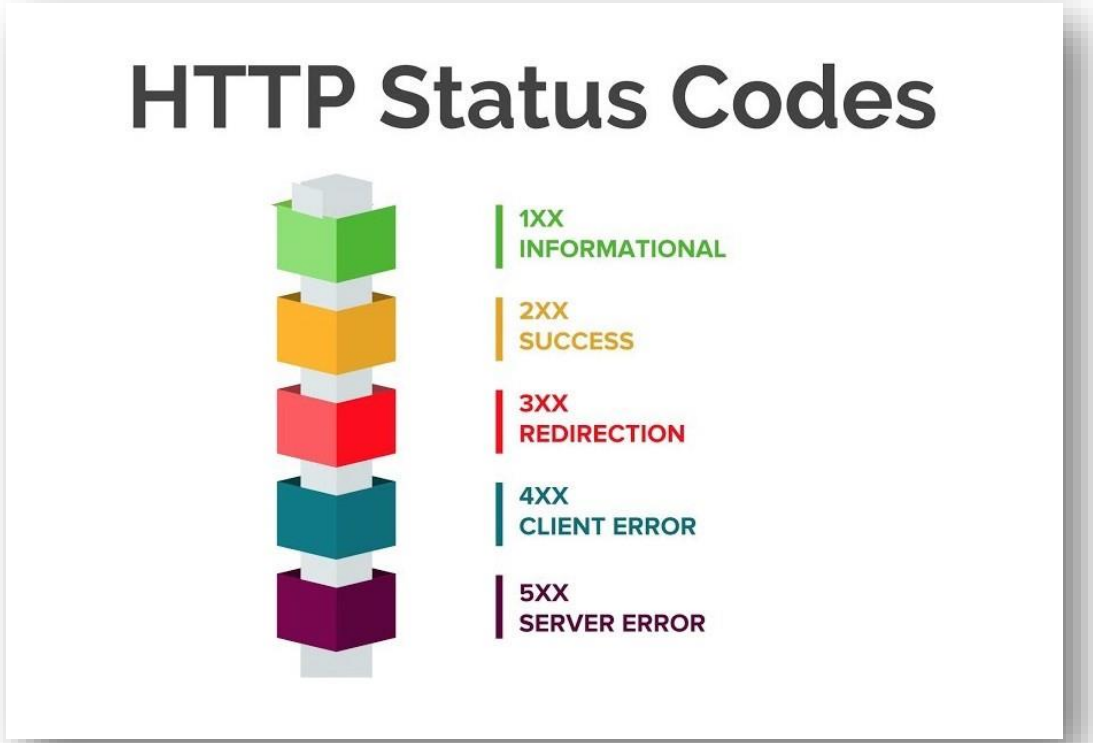
1.3 基于HTTP协议的网络连接(3, 响应报文)



由服务器端回送的对于请求的响应称为**响应报文**，由三部分构成：

- 状态行：Response line
  - 协议版本
  - 状态码
  - 原因短语：对状态码的描述
- 响应头：Response headers。以下列出几个常见的：
  - Content-Type：指明发送给客户端的实体正文的媒体类型，例如text/html; charset=utf-8表示响应实体是HTML文档，编码方式是UTF-8
  - Content-Length：实体正文的长度
- 响应正文： Response body：
  - 如HTML……

1.3 基于HTTP协议的网络连接(4, 状态码)



Informational Status Codes

100 – Continue [The server is ready to receive the rest of the request.]

101 – Switching Protocols [Client specifies that the server should use a certain protocol and the server will give this response when it is ready to switch.]

Client Request Successful

200 – OK [Success! This is what you want.]

201 – Created [Successfully created the URI specified by the client.]

202 – Accepted [Accepted for processing but the server has not finished processing it.]

203 – Non-Authoritative Information [Information in the response header did not originate from this server. Copied from another server.]

204 – No Content [Request is complete without any information being sent back in the response.]

205 – Reset Content [Client should reset the current document. Ie. A form with existing values.]

206 – Partial Content [Server has fulfilled the partial GET request for the resource. In response to a Range request from the client. Or if someone hits stop.]

Request Redirected

300 – Multiple Choices [Requested resource corresponds to a set of documents. Server sends information about each one and a URL to request them from so that the client can choose.]

301 – Moved Permanently [Requested resource does not exist on the server. A Location header is sent to the client to redirect it to the new URL. Client continues to use the new URL in future requests.]

302 – Moved Temporarily [Requested resource has temporarily moved. A Location header is sent to the client to redirect it to the new URL. Client continues to use the old URL in future requests.]

303 – See Other [The requested resource can be found in a different location indicated by the Location header, and the client should use the GET method to retrieve it.]

304 – Not Modified [Used to respond to the If-Modified-Since request header. Indicates that the requested document has not been modified since the the specified date, and the client should use a cached copy.]

305 – Use Proxy [The client should use a proxy, specified by the Location header, to retrieve the URL.]

307 – Temporary Redirect [The requested resource has been temporarily redirected to a different location. A Location header is sent to redirect the client to the new URL. The client continues to use the old URL in future requests.]

Client Request Incomplete

400 – Bad Request [The server detected a syntax error in the client's request.]

401 – Unauthorized [The request requires user authentication. The server sends the WWW-Authenticate header to indicate the authentication type and realm for the requested resource.]

402 – Payment Required [reserved for future.]

403 – Forbidden [Access to the requested resource is forbidden. The request should not be repeated by the client.]

404 – Not Found [The requested document does not exist on the server.]

405 – Method Not Allowed [The request method used by the client is unacceptable. The server sends the Allow header stating what methods are acceptable to access the requested resource.]

406 – Not Acceptable [The requested resource is not available in a format that the client can accept, based on the accept headers received by the server. If the request was not a HEAD request, the server can send Content-Language, Content-Encoding and Content-Type headers to indicate which formats are available.]

407 – Proxy Authentication Required [Unauthorized access request to a proxy server. The client must first authenticate itself with the proxy. The server sends the Proxy-Authenticate header indicating the authentication scheme and realm for the requested resource.]

408 – Request Time-Out [The client has failed to complete its request within the request timeout period used by the server. However, the client can re-request.]

409 – Conflict [The client request conflicts with another request. The server can add information about the type of conflict along with the status code.]

410 – Gone [The requested resource is permanently gone from the server.]

411 – Length Required [The client must supply a Content-Length header in its request.]

412 – Precondition Failed [When a client sends a request with one or more If... headers, the server uses this code to indicate that one or more of the conditions specified in these headers is FALSE.]

413 – Request Entity Too Large [The server refuses to process the request because its message body is too large. The server can close connection to stop the client from continuing the request.]

414 – Request-URI Too Long [The server refuses to process the request, because the specified URI is too long.]

415 – Unsupported Media Type [The server refuses to process the request, because it does not support the message body's format.]

417 – Expectation Failed [The server failed to meet the requirements of the Expect request-header.]

Server Errors

500 – Internal Server Error [A server configuration setting or an external program has caused an error.]

501 – Not Implemented [The server does not support the functionality required to fulfill the request.]

502 – Bad Gateway [The server encountered an invalid response from an upstream server or proxy.]

503 – Service Unavailable [The service is temporarily unavailable. The server can send a Retry-After header to indicate when the service may become available again.]

504 – Gateway Time-Out [The gateway or proxy has timed out.]

505 – HTTP Version Not Supported [The version of HTTP used by the client is not supported.]

Unused status codes

306- Switch Proxy

416- Requested range not satisfiable

506- Redirection failed

HTTP protocol version 1.1 Server Response Codes

http://www.w3.org/Protocols/rfc2616/rfc2616.html

Chart created September 5, 2000 by Suso Banderas(suso@suso.org). Most of the summary information was gathered from Appendix A of "Apache Server Administrator's Handbook" by Mohammed J. Kabir.

1.3 基于HTTP协议的网络连接(5, cookie技术)

Cookie技术

——通过在Request和Response中写入cookie信息来**控制客户端的状态**。

Cookie最开始由服务器端喂客户端

——Cookie会根据服务器端响应报文中响应头部信息中的Set-Cookie字段，通知客户端保存这一段cookie。

以后每次客户端见服务器都带着

——当下次客户端再与该服务器建立连接发送请求时，客户端会自动在请求报文的请求头信息中加入cookie值，即Cookie字段；服务器拿到cookie字段后会去解析它，识别出是哪一个客户端发来的请求，然后对比其上的记录，得到此前该客户端的状态信息——比如小本本上记着该客户端此前不久登陆过了，就返回登录后界面的html。

参考书籍

《图解HTTP》

作者: [日] 上野宣

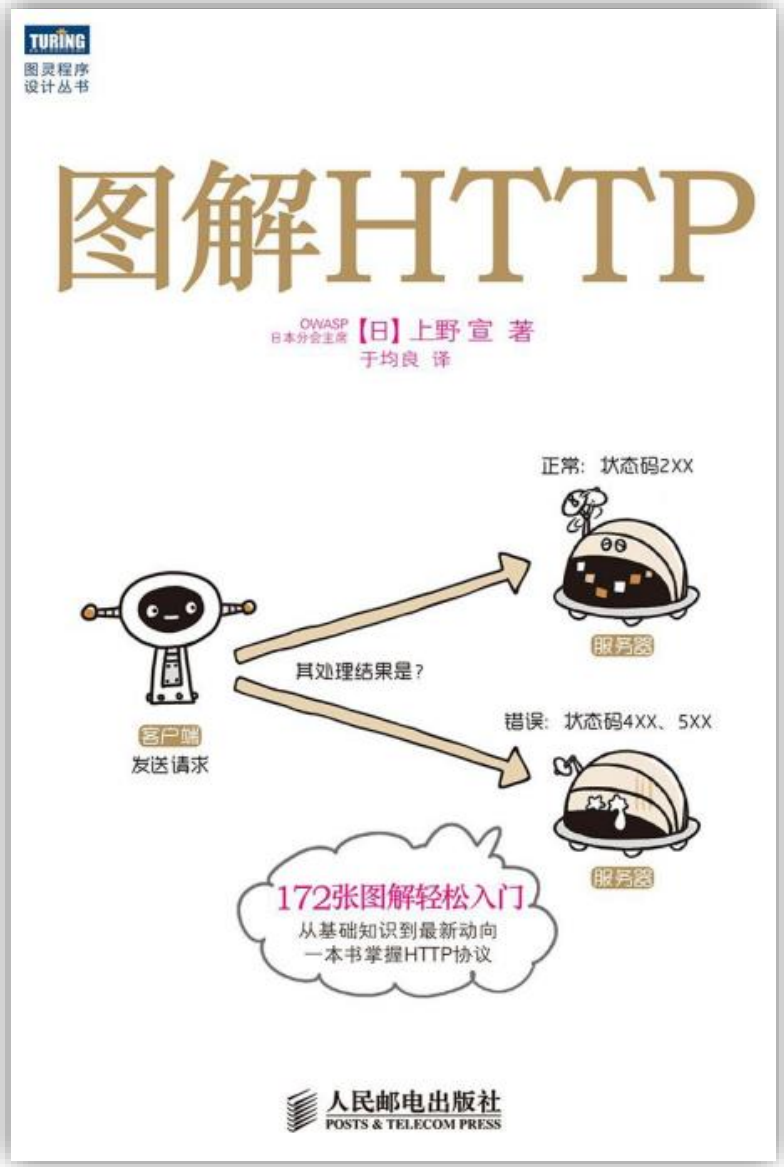
出版社: 人民邮电出版社

出品方: 图灵教育

出版年: 2014-4

页数: 308

ISBN: 9787115351531



二、HTML初步

1. 网页的基本结构
2. 常用标记
3. CSS简介

2.1 网页的基本结构(1)

```
<!doctype html>
<html>
        <head>
                <meta charset = "UTF-8">
                <title>......</title>
                ......
        </head>
        <body>
                ......
        </body>
</html>
```

几乎所有的网页代码均可抽象为上述简单结构

HTML

- HTML是一种语言，但不是编程语言，而是一种表示网页信息的符号标记语言。互联网中形形色色的网页就是用HTML撰写的，HTML是网页的基础。
- HTML全称为Hypertext Markup Language，译为“超文本标记语言”。Web浏览器会读取使用HTML语言撰写的文档，并于渲染后以网页的形式显示出它们。
- 注意，浏览器的“渲染”是指，浏览器不会显示HTML语言本身，而是根据HTML语言(标记)来解析页面内容，并将解析结果显示在电脑屏幕上。

注意

- 首先，html标记语言，对大小写不敏感，可以全部大写，也可以全部小写，也可以大小写混用。
- 其次，缩进不是必须的，但有缩进有助于观察网页结构，快速得到层次关系，及时发现错误。
- 最后，html代码撰写完毕后，需保存至扩展名为html的文档中再用浏览器打开，即可看到网页效果。

2.1 网页的基本结构(2)

```
<!doctype html>
<html>
        <head>
                <meta charset = "UTF-8">
                <title>......</title>
                ......
        </head>
        <body>
                ......
        </body>
</html>
```

几乎所有的网页代码均可抽象为上述简单结构

head：用于设置网页标题、网页编码等影响全局方面的内容，其中的内容不会在浏览器中显示；

body：主要包含将要显示的网页内容。

标记

- 左图中的一个个“<…>”被称为“标记”，这也是为什么html是一种标记语言——html使用标记来描述网页，一个html文档包含了html标记和文本内容，html标记是帮助浏览器解析怎样组织、怎样显示文本内容的工具，所以在你的浏览器里并不会显示html标记，而只会显示经过html标记组织后的文本内容。
- 标记 = 标签 = 元素 = tag，不同的称呼而已。
- 分类：
  - 双端标记：需要配对使用的标记，占html标记的绝大多数。他们将文本内容包裹在两个标记之间。注意结束标签中多一个斜杠。
  - 单端标记：不需配对使用的标记，少量，如上图中的<meta>。

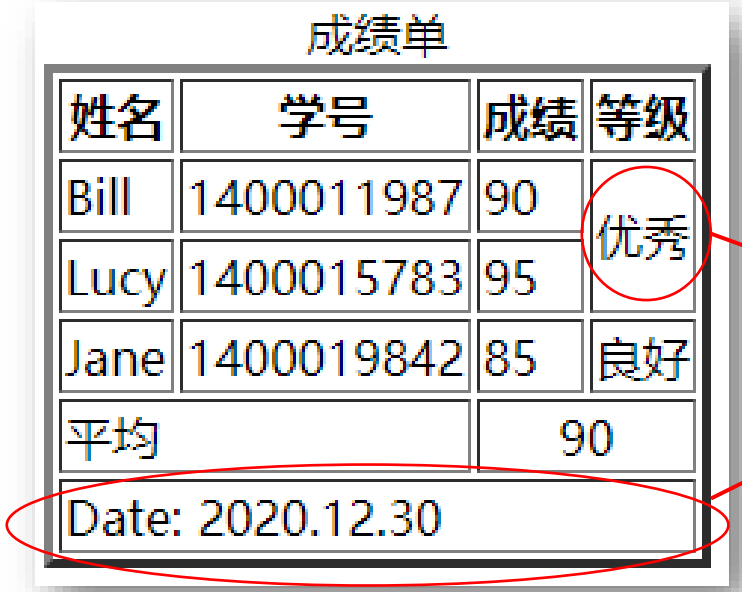
属性

- 即“标签的属性”，定义标签的特征。如上图中的单端标记<meta>就拥有一个属性charset，其属性值为UTF-8。
- 当标签有多个属性时，“属性=属性值”之间用空格隔开即可。
- Tip：绝大多数html标签支持style属性，该属性能够定义元素的显示效果。

2.2 常用标记(1)

标记	名称	类型	作用	常用属性
p	段落	双端	在此标记之间的内容自动与此标记前后的内容分行	\
br	换行	单端	类似于回车	\
img	图片	单端	在网页中插入图片	src=图片位置, alt=图片说明
a	超链接	双端	在网页中插入超链接	链接名称 target=_blank/_self…
div	块区域	双端	分区，其中的内容自动与上下文分行显示（显示效果类似于p）	style
span	行区域	双端	标记范围，没有像div那样的分行显示效果，即默认情况下其内部的内容与前后文内容在同一行显示	style
h1	标题	双端	标题样式：默认独占一行，且前后有一定的行距	\
ol	有序列表	双端	用阿拉伯数字对每行进行编号，配合	\
ul	无序列表	双端	用圆点对每行进行编号，配合	\
hr	横线	单端	用于产生横线	\

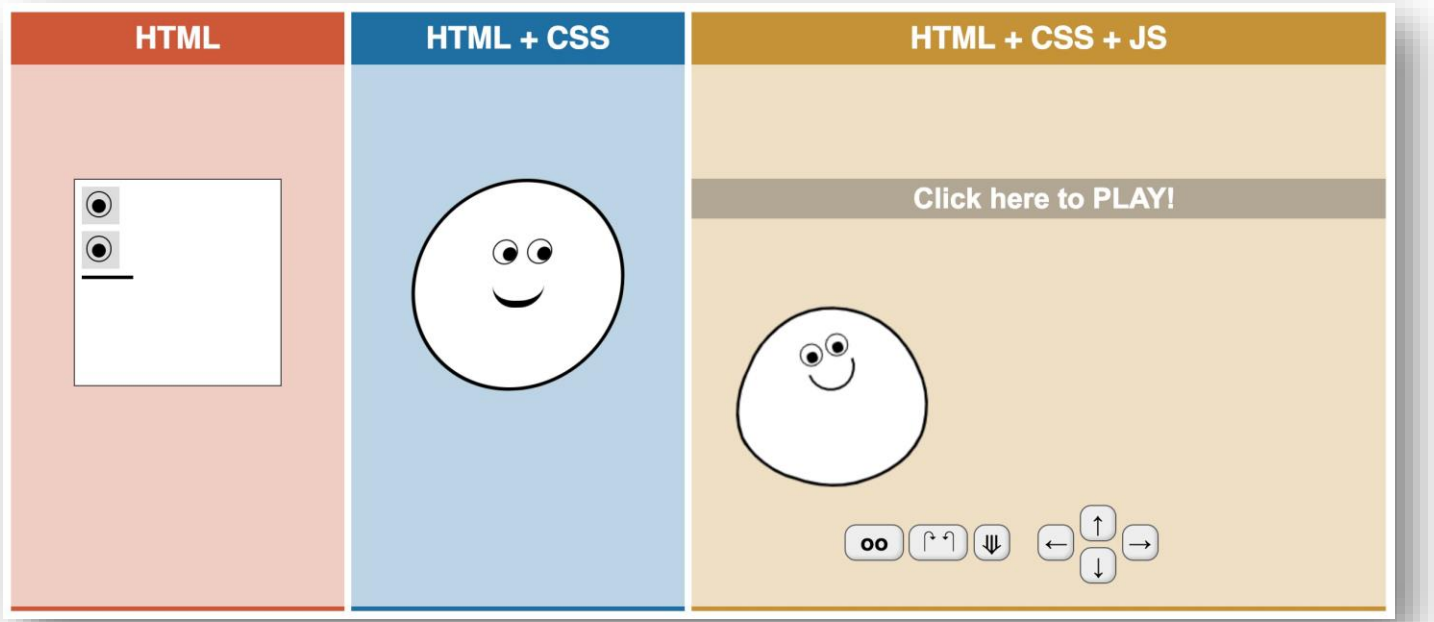
2.2 常用标记(2)



表格结构：<table>→<tr>→<td>，均双端

- tr：table row，即表格的行；td：table data，即表格的单元格；
- <table>的border属性：设置表格边框宽度；若省略该属性，则表格将没有边框；
- 跨行：<td>的rowspan属性
- 跨列：<td>的colspan属性
- 更完整的表格结构由标题<caption>，表头<thead>，表尾<tfoot>，主体<tbody>构成

2.3 CSS



CSS，全称为Cascading Style Sheets，译为“层叠样式表”。CSS定义如何显示html元素，因此常常配合html使用。

- 绝大多数html标签均支持style属性，可以对标签中的内容进行修饰，基本结构为：<标签名 style=“属性1:属性值1;属性2:属性值2;”>.....</标签名>——常见的style的属性有：颜色(color)，字体(font)，边框(border)，背景(background)，宽度(width)，高度 (height)等。上面放在style属性中的值，就是CSS。

CSS的使用方式

- 内联式，嵌入式，外部式

参考书籍

《 Head First HTML and CSS 》

作者: Elisabeth Robson / Eric Freeman

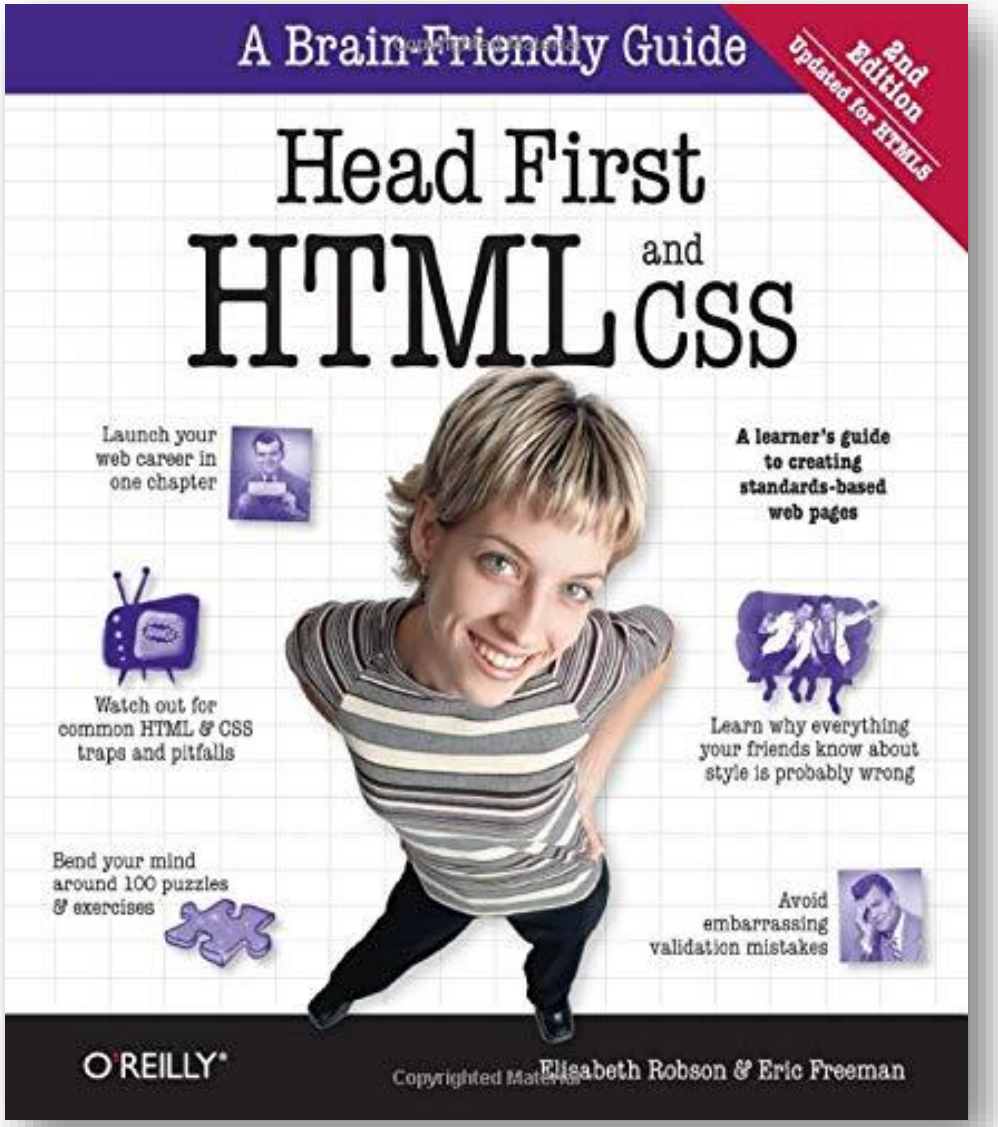
出版社: 中国电力出版社

译者: 徐阳 / 丁小峰

出版年: 2013-9

页数: 762

ISBN: 9787512344778



三、XPath简介

1. XPath概述
2. XPath语法
3. Python中的lxml库与chrome中的console

3.1 XPath概述(1)

“XPath (XML Path Language) is a query language for selecting nodes from an XML document.”

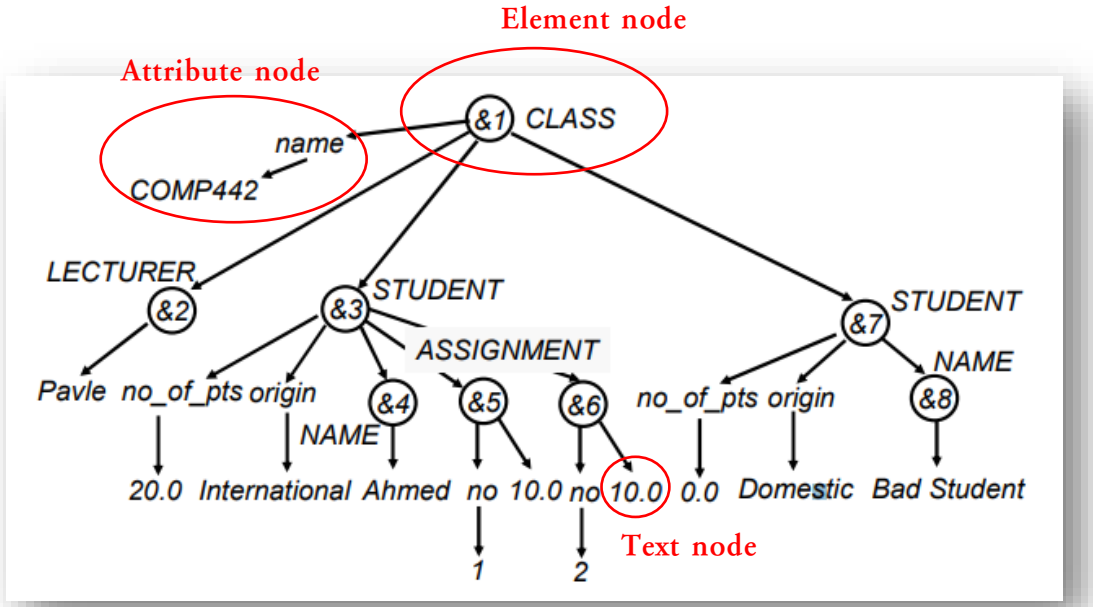
```xml
<?xml version="1.0" encoding="ISO-8859-1"?>

<bookstore>

<book>
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

</bookstore>
```

XML文档

- XML是“可扩展标记语言”，是一种用于标记电子文件使其具有结构性的标记语言——很像HTML——因此XPath也能工作于html文档；
- 一言蔽之，XPath的一大功能就是“定位”；
- Chrome：Copy Xpath；
- //*[@id="content"]/div/div[1]/ol/li[1]/div/div[1]/a/img
- 上面的类似于电脑文件路径表达式的式子，就是用XPath语言书写的定位表达式，用于在html文档中实现元素的搜索、查询和定位。

3.1 XPath概述(2)

```
<?xml version="1.0" standalone="yes"?>
<CLASS name="COMP442">
        <LECTURER>Pavle</LECTURER>
        <STUDENT no_of_pts="20.0" origin="International">
                <NAME>Ahmed</NAME>
                <ASSIGNMENT no="1">10.0</ASSIGNMENT>
                <ASSIGNMENT no="2">10.0</ASSIGNMENT>
        </STUDENT>
        <STUDENT no_of_pts="0.0" origin="Domestic">
                <NAME>Bad Student</NAME>
        </STUDENT>
</CLASS>
```



XPath将一个XML/HTML文档model成一棵挂满nodes的tree

Node的类型（7种）：element(元素), attribute(属性), text(文本), namespace, processing-instruction, comment, document nodes

3.2 Xpath语法(1)

- XPath的路径表达式有2×2=4种写法：（绝对路径，相对路径）×（简写，非简写）；
- 我们先看“相对路径&非简写”的写法。

$$\text{Path} := \text{Step}_1/\text{Step}_2/\cdots/\text{Step}_n$$

其中，Step := Axis::Node-test Predicate

- Step作用于一个node（称为context node），得到一个node set；
- Step作用于一个node set：将其中每一个node拿出来让Step作用得到一个个node set，再将结果union起来；
- 因此$\text{Step}_1/\text{Step}_2/\cdots/\text{Step}_n$最终得到一个node set（当然，可能为空集）。

3.2 Xpath语法(2, 轴-axis)

Axis specifies a direction in which to navigate through a document.

- child::，选择context node的子节点
- attribute::，选择context node的属性节点
- descendant:: ，选择context node的子孙节点
- parent:: ，选择context node的父节点
- self:: ，选择context node的自己
- 选择所有的话，用通配符*。例如，选择context node的所有子节点： child::*

```
<?xml version="1.0" standalone="yes"?>
<CLASS name="COMP442">
        <LECTURER>Pavle</LECTURER>
        <STUDENT no_of_pts="20.0" origin="International">
                <NAME>Ahmed</NAME>
                <ASSIGNMENT no="1">10.0</ASSIGNMENT>
                <ASSIGNMENT no="2">10.0</ASSIGNMENT>
        </STUDENT>
        <STUDENT no_of_pts="0.0" origin="Domestic">
                <NAME>Bad Student</NAME>
        </STUDENT>
</CLASS>
```

注意

element是attribute和text的父节点，但attribute和text却不是element的子节点

i.e. 元素和文本可以由子查父，但不能由父查子

3.2 Xpath语法(3, 节点测试-node test)

```
<?xml version="1.0" standalone="yes"?>
<CLASS name="COMP442">
        <LECTURER>Pavle</LECTURER>
        <STUDENT no_of_pts="20.0" origin="International">
                <NAME>Ahmed</NAME>
                <ASSIGNMENT no="1">10.0</ASSIGNMENT>
                <ASSIGNMENT no="2">10.0</ASSIGNMENT>
        </STUDENT>
        <STUDENT no_of_pts="0.0" origin="Domestic">
                <NAME>Bad Student</NAME>
        </STUDENT>
  </CLASS>
```

A Node-test specifies a simple test on nodes found along the axis.

- 沿轴筛选node;
- 最常用①：使用节点名称筛选元素节点。例如：child::STUDENT选中context node的子节点中的STUDENT元素节点;
- 若轴为child则可省略：…/child::STUDENT/…可简写为…/STUDENT/…
- 最常用②：attribute::*name*，选择context node中的*name*属性。可简写为：@*name*。

3.2 Xpath语法(4, 谓语-predicate)

The predicates are applied to the nodes selected by Node-test.

- 二次筛选;
- 具体而言，谓语是用运算符{=, <, >,<, >, !=}比较node property和某个值从而达到筛选的目的;
- node property:
  - 属性值
  - 索引值（return by position()）
- 例1：child::STUDENT[attribute::origin="Domestic"]，表示先选择context node的所有STUDENT元素子节点，然后从中再挑出属性origin的值为"Domestic"的那些nodes。可简写为STUDENT[@origin="Domestic"]。
- 例2：child::STUDENT[position()=2]，表示先选择context node的所有STUDENT元素子节点，然后取第2个。可简写为STUDENT[2]。

```
<?xml version="1.0" standalone="yes"?>
<CLASS name="COMP442">
        <LECTURER>Pavle</LECTURER>
        <STUDENT no_of_pts="20.0" origin="International">
                <NAME>Ahmed</NAME>
                <ASSIGNMENT no="1">10.0</ASSIGNMENT>
                <ASSIGNMENT no="2">10.0</ASSIGNMENT>
        </STUDENT>
        <STUDENT no_of_pts="0.0" origin="Domestic">
                <NAME>Bad Student</NAME>
        </STUDENT>
</CLASS>
```

3.2 Xpath语法(5, 其他)

- **绝对路径表达式：** 以/开头，表示root node，即整个文档
- **相对的：** 以//开头，表示从任意位置开始，而不只是像/表示的从文档最外层开始；
  - //STUDENT与/STUDENT
  - /CLASS/ASSIGNMENT与/CLASS//ASSIGNMENT（“跨层递进”）
- **.和..：** .相当于当前节点，即self::node()；..相当于父节点，即parent::node()；其中node()返回任何类型的节点。
  - .//STUDENT：当前节点的所有STUDENT子孙元素节点及自己（if）
  - ../LECTURER：当前节点的父节点的所有LECTURER子元素节点
- **text()函数：提取双端标签间的文本**
- **多个谓语：** and或or连接，如//student[@origin="domestic" or @orgin="Domestic"]

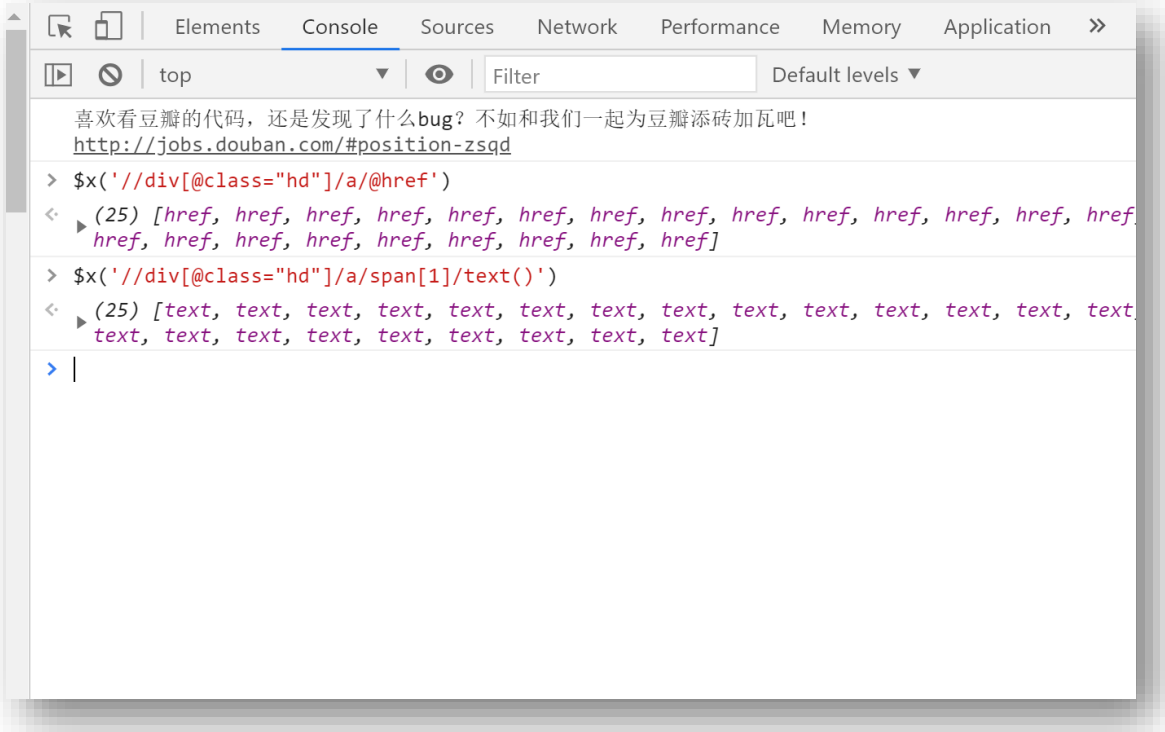
```
<?xml version="1.0" standalone="yes"?>
<CLASS name="COMP442">
        <LECTURER>Pavle</LECTURER>
        <STUDENT no_of_pts="20.0" origin="International">
                <NAME>Ahmed</NAME>
                <ASSIGNMENT no="1">10.0</ASSIGNMENT>
                <ASSIGNMENT no="2">10.0</ASSIGNMENT>
        </STUDENT>
        <STUDENT no_of_pts="0.0" origin="Domestic">
                <NAME>Bad Student</NAME>
        </STUDENT>
  </CLASS>
```

3.3 Python中的lxml库与chrome中的console

```python
import requests
from lxml import etree

url      = "https://movie.douban.com/top250"
headers  = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64
res      = requests.get(url, headers=headers)
html_str = res.text
root     = etree.HTML(html_str)

items = root.xpath("//div[@class='hd']")
for item in items:
    i_name = item.xpath("./a/span[1]/text()")[0]
    i_url  = item.xpath("./a/@href")[0]
    print(i_name, ': ', i_url, sep='')
```

```
问题  输出  调试控制台
unning] python -u "c:\Users\Alex Gao\Desktop\2020-12-30-数据采集\d
申克的救赎: https://movie.douban.com/subject/1292052/
王别姬: https://movie.douban.com/subject/1291546/
甘正传: https://movie.douban.com/subject/1292720/
个杀手不太冷: https://movie.douban.com/subject/1295644/
坦尼克号: https://movie.douban.com/subject/1292722/
丽人生: https://movie.douban.com/subject/1292063/
与千寻: https://movie.douban.com/subject/1291561/
```

使用lxml库中的etree进行xpath提取



使用chrome开发者工具中的console进行xpath调试

参考资料

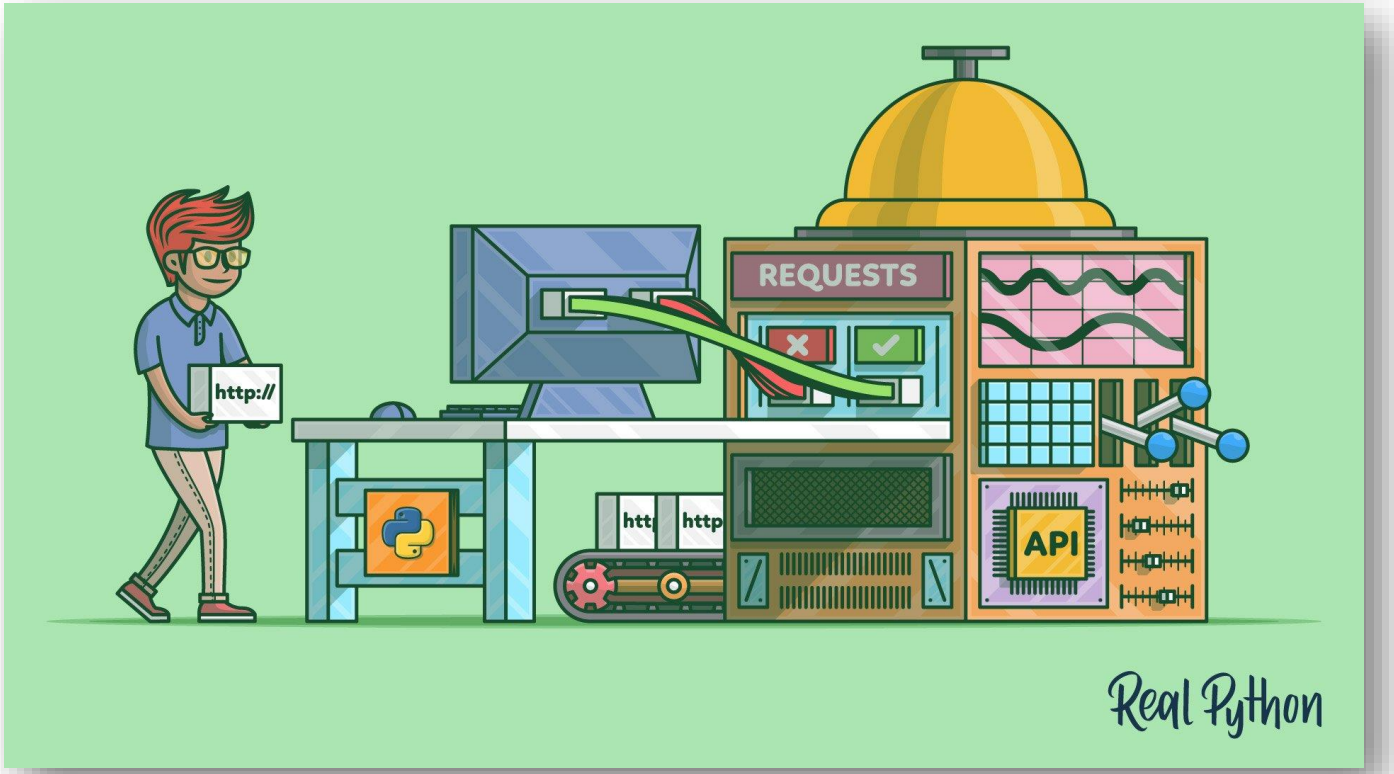
- XPath：
  - *An Introduction to XPath Language*——https://www.emse.fr/~beaune/websem/XPath.pdf
  - *Wiki：XPath*——https://en.wikipedia.org/wiki/Xpath
- lxml-etree：
  - *The lxml.etree Tutorial*——https://lxml.de/tutorial.html
- Chrome-console
  - https://developers.google.com/web/tools/chrome-devtools/console/utilities#xpath

四、Requests基础

1. 选课名单
2. 视频下载
3. 树洞采集（Charles的使用）

4.1 Requests

requests库是第三方库，因此需要手动安装——pip3 install requests。不要把.py文件的名字命名为requests.py!



官方文档：https://requests.readthedocs.io/zh_CN/latest/