第 2 章

◂ 爬虫基础快速入门 ▸

要学习网络爬虫，除了需要有 Python 编程的基础知识外，还需要对网络、HTTP、网页、爬虫原理、会话以及代理原理有一个全方位的认识。本章将着重讲述关于网络爬虫的这些基础知识，通过学习本章内容，读者会对爬虫的基础有一个更为深刻的认识，从而为后面的爬虫技术学习打下基础。

2.1 HTTP 基本原理

HTTP（Hyper Text Transfer Protocol，超文本传输协议）是人们平常使用最多的一种网络请求方式，常见的是在网络浏览器中输入一个网址。那么 HTTP 到底是怎么回事，发送请求、获取响应是怎么运作的？本节就来解决这个问题。

2.1.1 URI 和 URL

在了解 HTTP 基础原理之前，先来了解一下 URI（Uniform Resource Identifier，统一资源标志符）和 URL（Universal Resource Locator，统一资源定位符）这两个基础概念。

URL 是 URI 的子集，URI 还包括一个子类 URN（Universal Resource Name，统一资源名称）。URI 可被视为定位符 URL、名称 URN 或者两者兼备。URN 定义某事物的身份，URL 提供查找该事物的方法。URN 仅用于命名，而不指定地址。

提　示
在互联网中，URN 使用很少，几乎都是 URI 和 URL，所以一般网页可以称 URL 或 URI。

URL 有时用来定位那些包含指示器的资源，而这些指示器又指向其他资源。有时这些指示器用关系链接表示，在关系链接中，第二资源的位置表示符原则上“和那些除了带有次相关路径的表示符相同”。关系链接的使用依赖于包含分层结构的原始 URL，这是关系链接的基础。有些 URL 方案（例如 FTP、HTTP 和文件方案）包含的名字可以被认为是分层次的，这些层次之间用“/”分隔。

常见的一些方案列举如下：

- File Transfer Protocol（文件传输协议，FTP）
- HyperText Transfer Protocol（超文本传输协议，HTTP）
- Gopher The Gopher Protocol（Gopher 协议）
- Mailto Electronic Mail Address（电子邮件地址）
- News USENET News（USENET 新闻）
- NNTP USENET news using NNTP access（使用 NNTP 访问的 USENET 新闻）
- Telnet Reference to Interactive Sessions（交互式会话访问）
- Wais Wide Area Information Servers（广域信息服务系统）
- File Host-Specific File Names（特殊主机文件名）
- Prospero Directory Service（Prospero 目录服务）

2.1.2　超文本

浏览器中的网页是由超文本（Hypertext）解析而成。网页源代码是一系列 HTML 代码，里面包含一系列标签（如 img 显示图片、p 显示段落），浏览器解析这些标签后形成了我们平时看到的网页。网页的源代码比起普通文本能够描述更多的内容，包括网页的样式、网页的构成等，这些网页的源代码 HTML 就被称为超文本。

2.1.3　HTTP 和 HTTPS

HTTP（Hyper Text Transfer Protocol，超文本传输协议）是用于从网络传输超文本数据到本地浏览器的传输协议，它能保证高效而准确地传送超文本文档。

HTTPS（Hyper Text Transfer Protocol over Secure Socket Layer，超文本传输安全协议）是以安全为目标的 HTTP 通道，是 HTTP 的安全版，它在普通的 HTTP 下加入 TLS（Transport Layer Security，传输层安全协议）。TLS 是为网络通信提供安全及数据完整性的一种安全协议。

HTTPS 的安全基础是 SSL，通过它传输的内容都是 SSL 加密的，主要作用有两种：

- 一是建立一个信息安全通道，保证数据传输的安全。
- 二是确认网站的真实性，凡是使用 HTTPS 的网站都可以通过单击浏览器地址栏的锁头标志来查看网站认证之后的真实信息，也可以通过 CA 机构颁发的安全签章来查询。

一些网站虽然使用 HTTPS 协议，但还是会被浏览器提示不安全，如在 Chrome 浏览器中打开链接，它会提示“您的连接不是私密连接”，如图 2-1 所示。

图 2-1　Chrome 浏览器的提示

原因是某些网站的证书是不被官方机构认可的，所以证书验证不通过，但它的数据传输依然是 SSL 加密的。爬虫如果要爬取这样的站点，就需要设置忽略证书的选项，否则会提示 SSL 链接错误。

2.1.4　HTTP 请求过程

在浏览器输入一个 URL，按回车键后，在浏览器中观察页面内容，其中的过程是浏览器向网站所在服务器发送一个 Request（请求），网站服务器接收到 Request 后进行处理和解析，然后返回对应的 Response（响应），传回浏览器，Response 中包含页面的源代码等内容，浏览器再对其进行解析便会将网页呈现出来。

下面用 Chrome 浏览器的开发者模式下的 Network 监听组件进行演示。

步骤 01　首先使用 Chrome 浏览器输入网址：http://www.baidu.com。

步骤 02　使用快捷键 Ctrl+Shift+I，打开 Chrome 的开发者选项，单击“Network”选项，再按 F5 键刷新页面，将打开 Network 页面下的一个条目，这代表一次发送 Request 和接收 Response 的过程，如图 2-2 所示。

图 2-2　Chrome 的开发者选项

步骤 03 查看图 2-2 的结果可以发现网络请求记录这一项，其中各列的意义如下：

- 第一列 Name，即 Request 的名称。一般会用 URL 的最后一部分内容当作名称。
- 第二列 Method，说明请求的方法，通常有 GET、POST 等方式，图 2-2 中显示均为 GET 方式。
- 第三列 Status，即 Response 的状态码。这里显示为 200，代表 Response 是正常的，通过状态码可以判断发送了 Request 之后是否得到了正常的 Response。
- 第四列 Protocol，即为在网络中进行数据交换而建立的规则、标准或约定。用于不同系统中实体间的通信。这里均使用的是 HTTP1.1 协议。
- 第五列 Type，即 Request 请求的文档类型。第一行为 document，代表我们这次请求的是一个 HTML 文档，内容就是一些 HTML 代码。另外还有 gif、png 等其他类型。
- 第六列 Initiator，即请求源。用来标记 Request 是由哪个对象或进程发起的。其中有 Other、https://www.baidu.com 等内容。
- 第七列 Size，即从服务器下载的文件和请求的资源大小。如果是从缓存中取得的资源，该列就会显示 from memory cache。
- 第八列 Time，即发起 Request 请求到获取 Response 响应所用的总时间。单位 ms 为毫秒，0 代表从缓存中获取。
- 第九列 Waterfall，即网络请求的可视化瀑布流。如图 2-2 所示，是由不同颜色组成的可视化图。

步骤 04 单击条目可以看到更详细的信息，如图 2-3 所示。

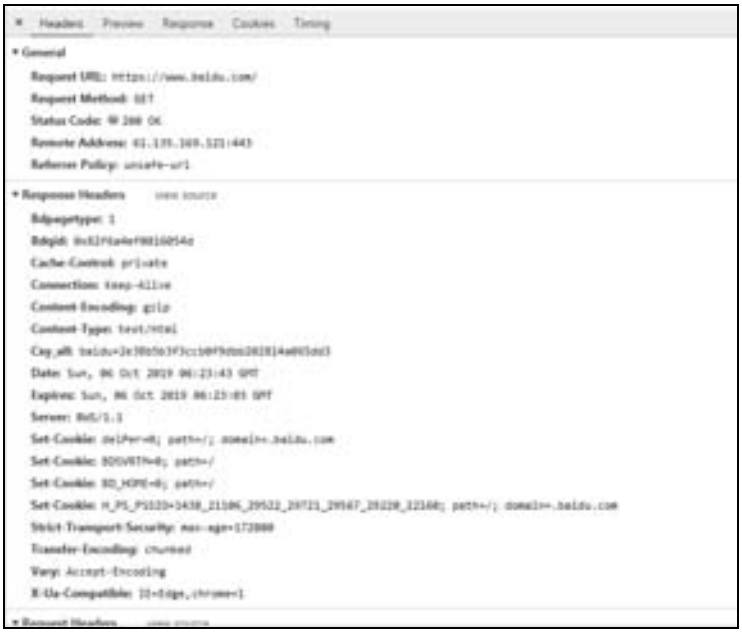

图 2-3　网络请求详细信息

图 2-3 所示的结果中的 General 部分，各个项目详细情况如下：

- Request URL 为 Request 请求的 URL。
- Request Method 为请求的方法。
- Status Code 为响应状态码。
- Remote Address 为远程服务器的地址和端口。
- Referrer Policy 为 Referrer 判别策略。

图 2-3 中的 Response Headers 和 Request Headers，分别代表响应头和请求头。

请求头里面带有许多请求信息，例如浏览器标识、Cookies、Host 等信息，这是 Request 的一部分，服务器会根据请求头内的信息判断请求是否合法，进而做出对应的响应，返回 Response，在图 2-3 中看到的 Response Headers 就是 Response 的一部分。例如，其中包含服务器的类型、文档类型、日期等信息，浏览器接收 Response 后，会解析响应内容，进而呈现网页内容。

2.1.5 请求

Request（请求）由客户端向服务端发出。可以将 Request 划分为 4 部分内容：Request Method（请求方式）、Request URL（请求 URL 地址）、Request Headers（请求头）和 Request Body（请求体）。

下面详细介绍请求的 4 部分内容。

（1）Request Method，常见的有两种；GET 和 POST。

在浏览器中直接输入一个 URL 并按回车键，便会发起一个 GET 请求，请求参数会直接包含到 URL 里，如在百度中搜索 Python，这就是一个 GET 请求，链接为：https://www.baidu.com/s?wd=Python。请求 URL 中包含请求的参数信息，这里的参数 wd 就是要搜索的关键词。

POST 请求大多为表单提交发起的，如登录表单、输入用户名和密码、单击“登录”按钮，通常会发起一个 POST 请求，其数据通常以 Form Data（表单形式）传输，不会体现在 URL 中。

GET 和 POST 请求方法的区别如下：

- GET 请求中的参数包含在 URL 里面，数据可以在 URL 中看到，而 POST 请求的 URL 不会包含这些数据，数据都是通过表单的形式传输的，会包含在 RequestBody 中。所以通常情况下，POST 请求比 GET 请求更安全。
- GET 方式请求提交的数据最多只有 1024 字节，而 POST 方式没有限制。

（2）Request URL，用 URL 可以唯一确定我们想请求的资源。

（3）Request Headers，用来说明服务器要使用的附加信息。一些常用的头部信息说明如下：

- Accept，请求报头域，用于指定客户端可接收哪些类型的信息。

- Accept-Language，指定客户端可接收的语言类型。
- Accept-Encoding，指定客户端可接收的内容编码。
- Host，用于指定请求资源的主机 IP 和端口号，其内容为请求 URL 的原始服务器或网关的位置。
- Cookie/Cookies，是网站为了辨别用户进行 Session 跟踪而存储在本地的数据。Cookies 的主要功能是维持当前访问会话，例如用户输入用户名和密码登录了某个网站，登录成功之后，服务器会用 Session 保存用户的登录状态信息，后面每次刷新或请求该站点的其他页面时会发现都是保持着登录状态的，在这里就是 Cookies 的功劳，Cookies 里有信息标识了用户所对应的服务器的 Session 会话，每次浏览器在请求该站点的页面时都会在请求头中加上 Cookies 并将其发送给服务器，服务器通过 Cookies 识别出用户，并且查出当前状态是登录的状态，所以返回的结果就是登录之后才能看到的网页内容。
- Referer，此内容用来标识这个请求是从哪个页面发过来的，服务器可以获取这个信息并进行相应的处理，如来源统计、防盗链处理等。
- User-Agent（UA），这是一个特殊字符串头，使得服务器能够识别客户使用的操作系统及版本、浏览器及版本等信息。在进行爬虫时，加上此信息可以伪装为浏览器，如果不加，就很可能会被识别为爬虫。
- Content-Type，即 Internet Media Type，互联网媒体类型，也叫作 MIME 类型，在 HTTP 协议消息头中，使用它来表示具体请求中的媒体类型信息。例如，text/html 代表 HTML 格式，image/gif 代表 GIF 图片，application/json 代表 JSON 类型，更多常见的对应关系如表 2-1 所示。

表 2-1　HTTP Content-Type 常见对照关系表

扩展名	Content-Type(Mime-Type)	扩展名	Content-Type(Mime-Type)
.doc	application/msword	.tif	image/tiff
.bmp	application/x-bmp	.bot	application/x-bot
.drw	application/x-drw	.dtd	text/xml
.htc	text/x-component	.htm	text/html
.html	text/html	.htt	text/webviewhtml
.img	application/x-img	.ins	application/x-internet-signup
.mocha	application/x-javascript	.movie	video/x-sgi-movie
.mp1	audio/mp1	.mp2	audio/mp2
.mp2v	video/mpeg	.mp3	audio/mp3
.mp4	video/mpeg4	.mpa	video/x-mpg
.pps	application/vnd.ms-powerpoint	.ppt	application/vnd.ms-powerpoint
.ppt	application/x-ppt	.pr	application/x-pr
.rtf	application/msword	.rtf	application/x-rtf
.wmv	video/x-ms-wmv	.xml	text/xml
.xsl	text/xml	.xslt	text/xml

提　示
Request Headers 是 Request 的重要组成部分，在编写爬虫程序时大部分情况下都需要设置 Request Headers。

（4）Request Body，一般承载的内容是 POST 请求中的 Form Data（表单数据），而对于 GET 请求，Request Body 则为空。

在 Request Headers 中指定 Content-Type 和 POST 提交数据方式的关系，在爬虫中，如果要构造 POST 请求，就要注意这几种 Content-Type，了解请求库的各个参数设置时使用的是哪种 Content-Type，不然可能会导致 POST 提交后得不到正常的 Response。

2.1.6　响应

Response（响应）由服务端返回给客户端。Response 可以划分为 3 部分：Response Status Code（响应状态码）、Response Headers（响应头）和 Response Body（响应体）。

（1）Response Status Code，表示了服务器的响应状态，如 200 表示服务器正常响应，404 代表页面未找到，500 代表服务器内部发生错误。在爬虫中可以根据状态码来判断服务器的响应状态，如判断为 200，就证明成功返回数据，再进行进一步处理，否则直接忽略。

（2）Response Headers，包含服务器对请求的应答信息，如 Context-Type、Server、Set-Cookie 等。一些常见的头信息说明如下：

- Date：标识 Response 产生的时间。
- Last-Modified：指定资源的最后修改时间。
- Content-Encoding：指定 Response 内容的编码。
- Serve：包含服务器的信息、名称、版本号等。
- Context-Type：文档类型，指定返回的数据类型。
- Set-Cookie：设置 Cookie，告诉浏览器要将此内容放在 Cookies 中，下次请求携带 Cookies 请求。
- Expires：指定 Response 的过期时间，使用它可以控制代理服务器或浏览器将内容更新到缓存中，再次访问时直接从缓存中加载，可以降低服务器负载，缩短加载时间。

（3）Response Body，非常重要，响应的正文数据都是在响应体中，如请求一个网页，它的响应体是网页的 HTML 代码，请求一张图片，响应体就是图片的二进制数据。在爬虫执行时，主要解析的内容就是 Response Body，通过 Response Body 可以得到网页的源代码、JSON 数据等，然后从中进行相应内容的提取。

2.2 网页基础

一个网站是由若干个网页组成的，可以说网页是构成整个网络的基石。本节就来了解一下有关网页的基础知识。

2.2.1 网页的组成

一般网站的页面都有文字、图片、超链接、表格、表单、动画及框架等。下面来详细介绍这些组成元素。

1. 框架

框架是网页的一种组织形式，将相互关联的多个网页的内容组织在一个浏览器窗口中显示出来。例如，我们可以在一个框架内放置导航栏，另一个框架中的内容可以随着单击导航栏中的链接而改变，这样我们只要制作一个导航栏的网页即可，而不必将导航栏的内容复制到各栏目的网页中去。

2. 文本

文本是网页中的主要信息。在网页中可以通过字体、字号、颜色、底纹以及边框等来设置文本属性。这里的文字指的是文本文字，而并非图片中的文字。在网页制作中，文字可以方便地设置成各种字体和大小，但是这里建议用于正文的文字不要太大，也不要使用太多的字体，中文文字使用宋体、9 磅或者 12、14 像素左右即可，因为文字过大在显示器中显示时线条不够平滑。颜色也不要使用得太过复杂，以免影响用户视觉。大段文本文字排列建议参考优秀的报纸杂志等。

3. 图片

我们能够看到丰富多彩的网页，都是因为网页中有了图片，可见图片在网页中的重要性。使用在网页上的图片一般为 JPG 和 GIF 格式的，即以.jpg 和.gif 为后缀的文件。

4. 超链接

超链接是整个网站的通道，它是把网页指向另一个目的端的链接，例如指向另一个网页或相同网页上的不同位置。这个目的端通常是另一个网页，也可以是图片、电子邮件地址、文件、程序，或者也可以是本网页的其他位置。超链接可以是文本或者图片。超链接广泛地存在于网页的图片和文字中，提供与图片和文字内容相关的链接。在超链接上单击，即可链接到相应地址（URL）的网页。有链接的地方，鼠标指到光标会变成小手的形状。可以说超链接是 Web 的主要特色。

5. 表格

表格是网页排版的灵魂，使用表格排版是网页的主要制作形式之一。通过表格可以精确地控制各网页元素在网页中的位置。表格并非指网页中直观意义的表格，范围要更广一些，它是 HTML 语言中一种元素。表格主要用于网页内容的排列，组织整个网页的外观，通过在表格中放置相应的内容即可有效地组合成符合设计效果的页面。有了表格的存在，网页中的元素得以方便地固定在设计位置上。一般表格的边线不在网页中显示。

6. 表单

表单是用来收集站点访问者信息的域集。站点访问者填写表单的方式有输入文本、单击单选按钮与复选框以及从下拉菜单中选择选项。在添写好表单之后，服务器将表单保存到数据库，该数据库就会根据所设置的表单处理程序，以各种不同的方式进行处理。

7. 动画

动画是网页上非常活跃的元素，通常制作优秀、有创意、出众的动画是吸引浏览者的有效方法。但太多的动画让人眼花缭乱，无心细看。这就使得对动画制作的要求越来越高。通常制作动画的软件有 Flash、Web Animator 等。Macromedia 的 Flash 虽然出现的时间不长，但已经成为重要的 Web 动画形式。Flash 不仅比 HTML 易学得多，而且有很多重要的动画特征，如关键帧、补间、运动路径、动画蒙版、形状变形和洋葱皮效果等。利用这个多才多艺的程序不仅可以制作 Flash 电影，而且可以把动画输出为 QuickTime 文件、GIF 文件以及其他不同格式的文件（PICT、JPEG、PNG 等）。

8. 其他

网页中除了这些基本元素外，还包括横幅广告、字幕、悬停按钮、日戳、计数器、音频及视频等。

2.2.2 网页的结构

很多时候学习网页制作开发第一看到的印象深刻的就是以.html 或.htm 为后缀的网页，那么 HTML 基本语言结构是怎么样的呢？

先看一下 HTML 结构：

```
    <html>
    <head>
    <title>放置文章标题</title>
    <meta http-equiv="Content-Type" content="text/html; charset=gb2312" /> //这里设置网页的编码，此时为 gb2312
    <meta name="keywords" content="关键字" />
    <meta name="description" content="本页描述或关键字描述" />
    </head>
    <body>
```

```
这里就是正文内容
</body>
</html>
```

一个完整的网页其 HTML 源代码包括 HTML DOCTYPE 声明、Title、Head、网页编码声明等内容。

最初 HTML 4 版本下完整的 HTML 源代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>标题部分</title>
<meta name="keywords" content="关键字" />
<meta name="description" content="本页描述或关键字描述" />
</head>
<body>
内容
</body>
</html>
```

HTML 5 版本结构完整的 HTML 源代码如下，注意其与 HTML4 版本在 DOCTYPE 方面的异同：

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
<meta charset="utf-8">
<title>网页标题</title>
<meta name="keywords" content="关键字" />
<meta name="description" content="此网页描述" />
</head>
<body>
网页正文内容
</body>
</html>
```

无论是 HTML 还是其他后缀的动态页面，其 HTML 语言结构都是这样的，只是在命名网页文件时以不同的后缀结尾。

（1）无论是动态还是静态页面，都是以“<html>”开始的，然后在网页最后以“</html>”结尾。

（2）“<html>”后接着“<head>”页头，其在<head></head>中的内容是在浏览器中无法显示的，这里是给服务器、浏览器链接外部 JavaScript、a 链接 CSS 样式的区域，而

“<title></title>”中放置的是网页标题，可在浏览器的标题栏看到，如图 2-4 所示。

图 2-4　<title></title>显示位置图

（3）“<meta name="keywords" content="关键字" /> <meta name="description" content="本页描述或关键字描述" /> ”这两个标签里的内容是给搜索引擎看的，说明本网页的关键字及主要内容等，SEO（Search Engine Optimization，搜索引擎优化）可以用到。

（4）接着就是正文“<body></body>”，也就是常说的 body 区，这里放置的内容可以通过浏览器呈现给用户，其内容可以是 table 表格布局格式的内容，可以 DIV 布局的内容，也可以直接是文字。这里是非常重要的区域，是网页内容的呈现区。

（5）最后以“</html> ”结尾，也就是网页闭合。

以上是一个完整的简单 HTML 语言基本结构，通过以上步骤可以增加更多的样式和内容充实网页。

注　意
网页一般根据 XHTML 标准都要求每个标签闭合，比如以<html>开始，以</html>闭合。如果没有闭合，如<meta name="keywords" content="关键字" />就没有</meta>，此时就要使用<meta 内容…/>来完成闭合。

以上是简单的 HTML 语言结构，如果需要查看更丰富的 HTML 语言结构，可打开一个网站的网页，然后在浏览器界面中使用快捷键 Ctrl+U 来查看网页源代码，这样可以根据此源代码来分析此网页的 HTML 语言结构与内容。

2.2.3　节点树及节点间的关系

HTML 网页文档从本质上来说是由 HTML 源代码构成的，而 HTML 源代码是一个树状结构，由一个个不同的节点构成了整个树。这一小节来介绍节点树及节点之间的关系。

1. DOM 节点树

首先来看 DOM 节点树。文档对象模型（Document Object Model，DOM）是 W3C（万维网联盟）组织推荐的处理可扩展置标语言的标准编程接口。这是一种与平台和语言无关的应用程序接口，DOM 可以动态地访问程序和脚本，更新其内容、结构和 WWW 文档的风格。

根据 W3C 的 DOM 标准，文档中的所有内容都是节点，节点是 DOM 的最小组成单元。浏览器会根据 DOM 模型将文档解析成一系列节点，这些节点组成一个树状结构，称为 DOM 节点树。

DOM 节点树体现了文档的层次结构，它有两种类型：一种是 DOM 文档节点树；另一种是 DOM 元素节点树。DOM 文档节点树包含文档中所有类型的节点，DOM 元素节点树只包含元素节点。

可以通过 DOM 提供的一些方法及属性来遍历文档节点树。

首先来遍历所有 Node 节点。遍历 DOM 文档节点树的方法兼容所有浏览器。

- parentNode：获取父节点。
- childNodes：获取所有子节点（包含元素节点和文本节点）。
- firstChild：获取第一个子节点。
- lastChild：获取最后一个子节点。
- nextSibling：获取元素之后紧跟的节点。
- previousSibling：获取元素之前紧跟的节点。

说　明
可以使用 JavaScript 配合以上方法来获取相应的节点。

以下属性在遍历元素节点树时会使用到。只遍历元素节点，其余类型的节点忽略。除了 children 方法外，其余方法在 IE 8 及以下版本的浏览器中都不兼容。

- parentElement：获取父元素。
- children：获取所有子元素。
- childElementCount：获取子元素的数量。
- firstElementChild：获取第一个子元素。
- lastElementChild：获取最后一个子元素。
- nextElementSibling：获取元素后紧跟的元素。
- previousElementSibling：获取元素前紧跟的元素。

下面的示例将演示如何使用演绎法获取一个元素的所有子节点。

【示例 2-1】获取元素的子节点

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
<title>获取元素子节点</title>
</head>
<body>
<ul>
<li>子元素 1</li>
<li>子元素 2</li>
```

```
<li>子元素 3</li>
<li>子元素 4</li>
<li>子元素 5</li>
<li>子元素 6</li>
<li>子元素 7</li>
</ul>
<script language="javascript">
  obj=document.getElementsByTagName("ul")[0]
  lis=obj.children
  num=lis.length
  for(i=0;i<num;i++)
  {
    l=lis[i]
    alert(l.innerText)
  }
</script>
</body>
</html>
```

以上代码使用 getElementsByTagName()方法获取标签“ul”元素，然后通过元素的 children 属性返回该元素的所有子元素节点，即其中的所有“li”。这里使用 children 是因为该属性只返回元素节点，而不会返回文本节点，这样便于更好地获取其中的内容。然后通过遍历来弹出每一个子元素节点的文本内容。将以上代码保存为：2-1.htm，执行该代码的结果如图 2-5 所示。

图 2-5　获取元素的子节点

2. DOM 节点之间的关系

DOM 将文档解析成一个由多层次节点构成的结构，即 DOM 结构。节点是 DOM 结构的基础。DOM 节点包含 12 种类型，每种类型的节点分别表示文档中不同的信息及标记。节点之间的关系构成了层次，整个文档表现为一个以特定节点为根节点的树形结构。节点关系类似于传统的家族关系，节点树相当于家谱。

提　示
DOM 的节点属性有一个特征，即 DOM 节点的关系属性都是只读的。

（1）parentNode，父级属性。每个节点都有一个 parentNode 属性，该属性指向当前节点在节点树中的父节点。对于一个节点来说，它的父节点只可能为 3 种类型，分别是元素节点、文档节点和文档片段节点。如果父节点不存在，就返回 null。

（2）parentElement，与 parentNode 属性不同的是，该属性指向当前节点在节点树中的父元素节点。IE 浏览器中只有元素节点有该属性，其他浏览器中所有类型的节点都有该属性。

下面通过一个示例来说明如何获取一个节点的父节点。

【示例 2-2】获取节点的父节点

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
<title>获取元素父节点</title>
</head>
<body>
<div>
    <strong></strong>
    <span></span>
</div>
<script language="javascript">
    var strong = document.getElementsByTagName("strong")[0];
    console.log(strong.parentElement);
</script>
</body>
</html>
```

以上代码先调用 getElementsByTagName()获取标签（返回一个列表中的第一个节点），然后使用节点的属性 parentElement 来获取节点的父节点，并将内容在控制台输出。将以上代码保存为：2-2.htm，执行该代码的结果如图 2-6 所示。

图 2-6　获取节点的父节点

（3）childNodes 子级属性，childNodes 属性返回一个只读的 NodeList 集合，保存着当前节点的第一层子节点。

将示例 2-2 的代码，修改如下：

```
<script>
    var div = document.getElementsByTagName("div")[0];
    console.log(div.childNodes);
</script>
```

以上代码使用 childNodes 来获取 div 的子节点。将修改后的代码保存为：2-3.htm，执行该代码的结果如图 2-7 所示。

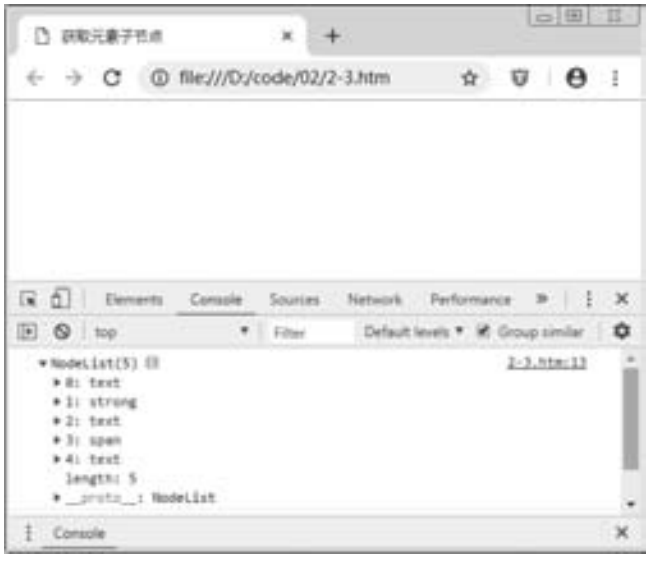

图 2-7　获取元素的子节点

查看图 2-7 中返回了 div 的所有子节点，分别是 text、strong、text、span、text，其 length 为 5。

children 返回一个只读的 HTMLCollection 集合，保存着当前节点的第一层子元素节点。在代码 2-1.htm 中已经对 children 属性做了说明，这里不再赘述。

childElementCount 属性返回当前节点下子元素节点的个数，相当于 children.length。IE 8 及以下版本的浏览器中不兼容。

将示例 2-2 的代码修改如下：

```
<script>
    var div = document.getElementsByTagName("div")[0];
    console.log(div.childElementCount); //2
</script>
```

以上代码使用节点的 childElementCount 属性来获取其所有子元素的数量。将代码保存为：2-4.htm，执行该代码的结果如图 2-8 所示。

图 2-8　获取子元素数量

- firstChild：当前节点的第一个子节点。IE 8 及以下版本的浏览器忽略空白文本节点。
- lastChild：当前节点的最后一个子节点。IE 8 及以下版本的浏览器忽略空白文本节点。
- firstElementChild：当前节点的第一个元素节点。IE 8 及以下版本的浏览器不兼容。
- lastElementChild：当前节点的最后一个元素节点。IE 8 及以下版本的浏览器不兼容。
- nextSibling：当前节点的后一个兄弟节点。IE 8 及以下版本的浏览器忽略空白文本节点。
- previousSibling：当前节点的前一个兄弟节点。IE 8 及以下版本的浏览器忽略空白文本节点。
- nextElementSibling：当前节点的后一个兄弟元素节点。IE 8 及以下版本的浏览器不兼容。
- previousElementSibling：当前节点的前一个兄弟元素节点。IE 8 及以下版本的浏览器不兼容。

除了以上属性之外，节点还有两个方法，分别为 hasChildNodes()方法和 contains()方法。

- hasChildNodes()：判断当前节点是否包含子节点，包含一个或多个子节点时返回 true。子节点可以是任意类型的。
- contains()：contains 方法接收一个节点作为参数，返回一个布尔值，表示参数节点是否为当前节点的后代节点。参数若为后代节点即返回 true，不一定是第一层子节点。IE 和 Safari 不支持 document.contains()方法，只支持元素节点的 contains()方法。

下面的代码将演示如何调用方法判断某一个节点是否有下级节点，以及一个节点是否包含在另一个节点中。

【示例 2-3】节点方法的调用

```
<!DOCTYPE html>
<html lang="zh-CN">
```

```
<head>
<title>节点方法的调用</title>
</head>
<body>
<div id="d1">
    <span id="s1"></span>
    <span id="s2"></span>
</div>
<script>
    console.log(d1.hasChildNodes());        //true
    console.log(s1.hasChildNodes());        //false
    console.log(d1.contains(s1));           //true
    console.log(s1.contains(s2));           //false
</script>
</body>
</html>
```

以上代码调用方法 hasChildNodes()判断 d1 与 s1 有无子节点，再调用方法 contains()判断某个节点是否是另一个节点的子节点。将代码保存为：2-5.htm，用浏览器打开，效果如图 2-9 所示。

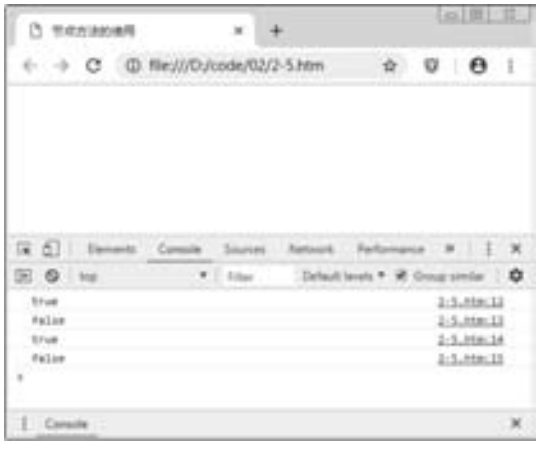

图 2-9　节点方法的使用

2.2.4　选择器

在 DOM 中要选择一个或多个节点，除了根据节点之间的各种父子兄弟关系选取之外，还可以通过 DOM 选择器来选择，DOM 选择器大致有以下几种：

（1）document.getElementByID()，该方法根据提供的 ID 名称返回唯一元素。元素 ID 在 IE 8 以下的浏览器中不区分字母大小写，而且返回匹配 name 属性的元素。

（2）getElementsByTagName()，该方法提供的标签名返回一组元素，如果要对其中一个

（即使是唯一一个）进行操作，就需要加上元素的数组引用标识，例如：

```
<div id = "only">123</div>
var div = document.getElementsByTagName('div');
```

把页面中的所有 div 都拿出来，放到一个类数组里，代表一个数组，代码如下：

```
div.style.background = "red"
```

这样会报错。为了不报错，把这个 div 取出来需要采取如下操作：

```
document.getElementsByTagName('div')[0];
```

尽管页面只有一个 div，但选出来的永远是一组，所以要加下标。

（3）getElementsByName()，根据提供的 name 返回一组元素，注意只有部分标签能够生效（表单、表单元素、img、iframe）。

（4）getElementsByClassName()，根据提供的 class 属性返回一组元素，IE 8 及以下版本不支持这样的操作。

关于这些方法的调用，在前面的例子中已经有所涉及，这里不再赘述。

2.3 爬虫的基本原理

前面介绍了 HTTP 基本原理及网页基础，这一节来介绍爬虫的基本原理，包括什么是爬虫、能抓取什么样的数据以及 JavaScript 渲染页面等内容。

2.3.1 爬虫概述

爬虫实际上就是采集网络上数据的一段程序。把这句话拆分一下，去掉其中的修饰词，就可以看到其实爬虫指的就是一段程序。这段程序的功能就是从网络上采集需要的数据。

一个爬虫的工作流程如下：

（1）发起请求。

（2）获取响应内容。

（3）解析内容。

（4）保存数据。

所以，爬虫就是从请求内容（即数据）到获取响应，接着解析内容，最后显示相应内容或者保持内容的过程。

2.3.2 能抓取什么样的数据

通过 2.3.1 小节的介绍，我们了解到爬虫的主要作用就是采集网络上的数据，那么究竟爬

虫能抓取什么样的数据呢？从广义上来说，只要是能请求并且能获取响应的数据都能够被抓取，具体包括以下几类：

（1）网页文本，如 HTML 文档、JSON 格式文本、XML 格式文本等。

（2）图片文件，如 JPEG、PNG 等图片文件，获取的是二进制文件，保存为相应格式的图片即可。

（3）视频文件，如 MP4、WMV 等视频文件，同样获取的是二进制文件，保存为相应的视频格式即可。

（4）其他文件。只要是能请求到的文件都能获取，比如 MP3 文件、Flash 文件以及其他各种类型的文件。

2.3.3 JavaScript 渲染页面

在介绍 JavaScript 渲染页面之前，我们先来了解一下浏览器渲染页面的过程。HTML 源代码由浏览器渲染为一个网页需要以下过程：

（1）浏览器根据 HTML 结构生成 DOM Tree（节点树）。

（2）浏览器根据 CSS（Cascading Style Sheets，层叠样式表）生成 CSSOM（样式表对象模型）。

（3）将 DOM 和 CSSOM 整合，形成 RenderTree（渲染树）。

（4）浏览器根据渲染树开始渲染页面。

在这个过程中遇到<script>时会执行并阻塞渲染，因为 JavaScript 有权利改变 DOM 结构，甚至也能改变对象的 CSS 样式，所以要等 JavaScript 脚本执行完毕，之后再继续渲染过程。

通过以上内容可以了解到，JavaScript 在整个页面渲染过程中起着举足轻重的作用，包括 JavaScript 在页面中的放置位置也十分重要。

如果 JavaScript 要对页面中的某个元素进行操作，就必须在这个元素加载之后才能进行，比如下面的代码。

【示例 2-4】JavaScript 渲染页面错误演示

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
<title>节点方法的调用</title>
</head>
<body>
<script>
  obj=document.getElementById("my_div")
  alert(obj.innerHTML)
</script>
```

```
<div id="my_div">1234567</div>
</body>
</html>
```

以上代码尝试获取页面中层里的内容并以弹出窗口的形式显示其内容。将以上代码保存为 2-6.htm，执行代码会发现，只会正常显示网页，但并没有任何弹出窗口。之所以会出现这种情况，就是因为<script>的出现中止了页面的渲染，导致获取不到<script>后面的层里的内容。

如果将代码修改如下：

```
<div id="my_div">1234567</div>
<script>
  obj=document.getElementById("my_div")
  alert(obj.innerHTML)
</script>
```

也就是将<script>放到要操作层之后，修改后将代码保存为 2-7.htm，再次执行代码就会看到弹出窗口的提示，执行结果如图 2-10 所示。

图 2-10　JavaScript 对页面渲染的影响

2.4 会话和 Cookies

Cookie 是一小段文本信息，伴随着用户请求和页面在 Web 服务器和浏览器之间传递。用户每次访问站点时，Web 应用程序都可以读取 Cookie 包含的信息。Cookie 的基本工作原理：如果用户再次访问站点上的页面，当该用户输入 URL 地址时，浏览器就会在本地硬盘上查找与该 URL 相关联的 Cookie。如果该 Cookie 存在，浏览器就将它与页面请求一起发送到用户请求的站点。Cookie 根本的用途是：Cookie 能够帮助 Web 站点保存有关访问者的信息。更简单地说，Cookie 是一种保持 Web 应用程序连续性（执行“状态管理”）的方法。

什么是会话（Session）？当用户访问站点时，服务器会为该用户创建唯一的会话，会话将一直延续到用户访问结束。

2.4.1 静态网页和动态网页

静态网页的网址形式通常是以.htm、.html、.shtml、.xml 等为后缀的。静态网页一般来说是最简单的 HTML 网页，服务器端和客户端是一样的，而且没有脚本和小程序，所以它不能动。在 HTML 格式的网页上，也可以出现各种动态的效果，如.gif 格式的动画、FLASH、滚动字母等，这些“动态效果”只是视觉上的，与后面将要介绍的动态网页是不同的概念。

静态网页有以下特点：

（1）静态网页每个网页都有一个固定的 URL，且网页 URL 以.htm、.html、.shtml 等常见形式为后缀，而不含有“?”。

（2）网页内容一经发布到网站服务器上，无论是否有用户访问，每个静态网页的内容都是保存在网站服务器上的，也就是说，静态网页是实实在在保存在服务器上的文件，每个网页都是一个独立的文件。

（3）静态网页的内容相对稳定，因此容易被搜索引擎检索。

（4）静态网页没有数据库的支持，在网站制作和维护方面工作量较大，因此当网站信息量很大时，完全依靠静态网页制作方式比较困难。

（5）静态网页的交互性较差，在功能方面有较大的限制。

动态网页是以.asp、.jsp、.php、.perl、.cgi 等形式为后缀的，并且在动态网页网址中有一个标志性的符号“?”。动态网页与网页上的各种动画、滚动字幕等视觉上的“动态效果”没有直接关系，动态网页可以是纯文字内容，也可以是包含各种动画的内容，这些只是网页具体内容的表现形式，无论网页是否具有动态效果，采用动态网页技术生成的网页都称为动态网页。动态网页也可以采用静动结合的原则，适合采用动态网页的地方采用动态网页，如果需要使用静态网页，就可以考虑用静态网页的方法来实现，在同一个网站上，动态网页内容和静态网页内容同时存在是很常见的事情。

动态网页具有以下特点：

（1）交互性，即网页会根据用户的要求和选择而动态改变和响应。例如，访问者在网页上填写表单信息并提交，服务器经过处理将信息自动存储到后台数据库中，并打开相应提示页面。

（2）自动更新，即无须手动操作，便会自动生成新的页面，可以大大节省工作量。例如，在论坛中发布信息，后台服务器将自动生成新的网页。

（3）随机性，即当不同的时间、不同的人访问同一网址时会产生不同的页面效果。例如，登录界面自动循环功能。

（4）动态网页中的“?”对搜索引擎检索存在一定的问题，搜索引擎一般不可能从一个网站的数据库中访问全部网页，或者出于技术方面的考虑，搜索蜘蛛不会去抓取网址中“?”后面的内容，因此采用动态网页的网站在进行搜索引擎推广时，需要做一定的技术处理才能适应搜索引擎的要求。

静态网页和动态网页各有特点，网站采用动态网页还是静态网页主要取决于网站的功能

需求和网站内容的多少，如果网站功能比较简单，内容更新量不是很大，采用纯静态网页的方式会更简单，如果网站功能复杂，那么最好采用动态网页技术来实现。

2.4.2　无状态 HTTP

HTTP（Hyper Text Transport Protocol，超文本传输协议）是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，它就必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时，它的应答就较快。

客户端与服务器进行动态交互的 Web 应用程序出现之后，HTTP 无状态的特性严重阻碍了这些应用程序的实现，毕竟交互是需要承前启后的，简单的购物车程序也要知道用户到底在之前选择了什么商品。于是，两种用于保持 HTTP 连接状态的技术就应运而生了，一个是 Cookie，另一个则是 Session。

Cookie 是通过客户端保持状态的解决方案。从定义上来说，Cookie 就是由服务器发给客户端的特殊信息，这些信息以文本文件的方式存放在客户端，然后客户端每次向服务器发送请求的时候都会带上这些特殊的信息。讲得更具体一些：当用户使用浏览器访问一个支持 Cookie 的网站的时候，用户会提供包括用户名在内的个人信息并且提交至服务器；接着，服务器在向客户端回传相应的超文本的同时也会发回这些个人信息，当然这些信息并不是存放在 HTTP 响应体中的，而是存放于 HTTP 响应头中的；当客户端浏览器接收到来自服务器的响应之后，浏览器会将这些信息存放在一个统一的位置。对于 Windows 操作系统而言，我们可以从：

```
[系统盘]:\Documents and Settings\[用户名]\Cookies
```

目录中找到存储的 Cookie。自此，客户端再向服务器发送请求的时候，都会把相应的 Cookie 再次发回至服务器。而这次，Cookie 信息则存放在 HTTP 请求头中。

有了 Cookie 这样的技术实现，服务器在接收到来自客户端浏览器的请求之后，就能够通过分析存放于请求头的 Cookie 得到客户端特有的信息，从而动态生成与该客户端相对应的内容。通常，我们可以从很多网站的登录界面中看到“请记住我”这样的选项，如果你勾选了该选项之后再登录，那么在下一次访问该网站的时候就不需要进行重复而烦琐的登录动作了，这个功能就是通过 Cookie 实现的。

与 Cookie 相对的一个解决方案是 Session，它是通过服务器来保持状态的。由于 Session 这个词汇包含的语义很多，因此需要在这里明确一下 Session 的含义。

首先，我们通常都会把 Session 翻译成会话，因此可以把客户端浏览器与服务器之间一系列的交互动作称为一个 Session。从这个语义出发，我们会想到 Session 持续的时间，会提到在 Session 过程中进行了什么操作，等等。

其次，Session 指的是服务器端为客户端所开辟的存储空间，在其中保存的信息就是用于保持状态的。从这个语义出发，我们会提到往 Session 中存放什么内容，如何根据键值从 Session 中获取匹配的内容，等等。

要使用 Session，第一步当然是创建 Session 了。那么 Session 在何时创建呢？当然是在服务器端程序运行的过程中创建的，不同语言实现的应用程序有不同的创建 Session 的方法，而在 Java 中是通过调用 HttpServletRequest 的 getSession 方法（使用 true 作为参数）来创建的。在创建了 Session 的同时，服务器会为该 Session 生成唯一的 Session ID，而这个 Session ID 在随后的请求中会被用来重新获得已经创建的 Session；在 Session 被创建之后，就可以调用 Session 相关的方法往 Session 中增加内容了，而这些内容只会保存在服务器中，发送到客户端的只有 Session ID；当客户端再次发送请求的时候，会将这个 Session ID 带上，服务器接收到请求之后就会依据 Session ID 找到相应的 Session，从而再次使用。正是这样一个过程，用户的状态就得以保持了。

综上所述，HTTP 本身是一个无状态的连接协议，为了支持客户端与服务器之间的交互，我们需要通过不同的技术为交互存储状态，而这些技术就是 Cookie 和 Session。

2.4.3 常见误区

在学习、理解无状态 HTTP 时要避免以下常见误区：

（1）有人在解释 HTTP 的无状态时，把它跟有连接对立，说是两种方式，也就是如果想不无状态，就必须有连接，但其实不然。

（2）有连接和无连接以及之后的 Keep-Alive 都是指 TCP 连接。

（3）有状态和无状态可以指 TCP，也可以指 HTTP。

（4）TCP 一直有状态，HTTP 一直无状态，但是应用为了有状态，就给 HTTP 加了 Cookie 和 Session 机制，让使用 HTTP 的应用也能有状态，但 HTTP 还是无状态。

（5）开始 TCP 有连接，后来 TCP 无连接，再后来也就是现在的 TCP 是 Keep-Alive，有点像有连接。

2.5 代理的基本原理

在进行网络爬虫时，经常会提到“代理”一词，那么究竟什么是代理？代理服务器是如何进行工作的？代理有什么作用？如何对代理服务进行设置？这一节就来解决这些问题。

2.5.1 基本原理

首先来看什么是代理。代理实际上指的就是代理服务器（Proxy Server），它的功能是代理网络用户获取网络信息。形象地说，代理服务器就是网络信息的中转站。

在用户正常请求一个网站时，发送了请求给 Web 服务器，Web 服务器把响应传回给用户。如果设置了代理服务器，实际上就是在本机和服务器之间搭建了一个桥，此时本机不是直接向 Web 服务器发起请求，而是向代理服务器发起请求，请求会发送给代理服务器，然后

由代理服务器再发送给 Web 服务器，接着由代理服务器把 Web 服务器返回的响应转发给本机。

这样用户同样可以正常访问网页，但在这个过程中，Web 服务器识别出的真实 IP 就不再是用户本机的 IP 了，成功实现了 IP 伪装，这就是代理的基本原理。

2.5.2　代理的作用

代理有什么作用呢？我们可以简单列举如下：

（1）代理可以突破自身 IP 访问限制，访问一些平时不能访问的站点。比如网上常说的翻墙、科学上网之类就是使用了代理技术。

（2）使用代理可以访问一些单位或团体的内部资源。比如使用的如果是教育网内地址段的免费代理服务器，就可以对教育网开放各类 FTP 下载、上传，以及各类资料查询、共享等服务。

（3）使用代理可以提高访问速度。通常代理服务器都会设置一个较大的硬盘缓冲区，当有外界的信息通过时，同时也会将其保存到缓冲区中，当其他用户再访问相同的信息时，则直接从缓冲区中取出信息传给用户，以提高访问速度。

（4）使用代理可以隐藏真实 IP。上网者可以通过代理服务隐藏自己的 IP，进而免受攻击。对于爬虫来说，使用代理就是为了隐藏自身 IP，防止自身的 IP 被封锁。因为通常大数据量的访问会引起对方主机的怀疑，从而有可能造成封锁 IP 的访问权限。

2.5.3　代理分类

按照代理不同的分类标准，可以对代理进行不同的分类，既可以根据协议区分，又可以根据其匿名程度区分。

根据代理的协议，代理可以分为如下类别：

（1）FTP 代理服务器：主要用于访问 FTP 服务器，一般有上传、下载以及缓存功能，端口一般为 21、2121 等。

（2）HTTP 代理服务器：主要用于访问网页，一般有内容过滤和缓存功能，端口一般为 80、8080、3128 等。

（3）SSL/TLS 代理：主要用于访问加密网站，一般有 SSL 或 TLS 加密功能（最高支持 128 位加密强度），端口一般为 443。

（4）RTSP 代理：主要用于访问 Real 流媒体服务器，一般有缓存功能，端口一般为 554。

（5）Telnet 代理：主要用于 Telnet 远程控制（黑客入侵计算机时常用于隐藏身份），端口一般为 23。

（6）POP3/SMTP 代理：主要用于 POP3/SMTP 方式收发邮件，一般有缓存功能，端口一般为 110/25。

（7）SOCKS 代理：只是单纯传递数据包，不关心具体协议和用法，所以速度快很多，一般有缓存功能，端口一般为 1080。SOCKS 代理协议又分为 SOCKS 4 和 SOCKS 5，前者只支持 TCP，而后者支持 TCP 和 UDP，还支持各种身份验证机制、服务器端域名解析等。简单来说，SOCK 4 能做到的 SOCKS 5 都可以做到，但 SOCKS 5 能做到的 SOCK 4 不一定能做到。

根据代理的匿名程度，代理可以分为如下类别：

（1）高度匿名代理：会将数据包原封不动地转发，在服务端看来就好像真的是一个普通的客户端在访问，而记录的 IP 是代理服务器的 IP。

（2）普通匿名代理：会在数据包上做一些改动，在服务端上有可能发现这是一个代理服务器，也有一定概率追查到客户端的真实 IP。代理服务器通常会加入的 HTTP 头有 HTTP_VIA 和 HTTP_X_FORWARDED_FOR。

（3）透明代理：不但改动了数据包，还会告诉服务器客户端的真实 IP。这种代理除了能用缓存技术提高浏览速度、能用内容过滤提高安全性之外，并无其他显著作用，常见的例子是内网中的硬件防火墙。

（4）间谍代理：指组织或个人创建的用于记录用户传输的数据，然后进行研究、监控等的代理服务器。

2.5.4 常见代理设置

常见的代理设置有以下几种：

（1）使用网上的免费代理。这种情况最好使用高匿代理，另外可用的代理不多，需要在使用前筛选一下可用代理，也可以进一步维护一个代理池。

（2）使用付费代理服务。互联网上存在许多代理商，可以付费使用，质量比免费代理好很多，而且速度更快，也更稳定，不过需要支付相应代理服务费。

（3）使用 ADSL 拨号。拨一次号换一次 IP，稳定性高，是一种比较有效的解决方案。

2.6 本章小结

本章对网络爬虫的基础做了一个相对全面、系统的介绍，包括 HTTP 基本原理、网页基础、爬虫的基本原理、会话和 Cookies 以及代理的基本原理等。通过本章的介绍，读者可以对爬虫有一个初步的认识，为接下来学习爬虫知识打下坚实的基础。