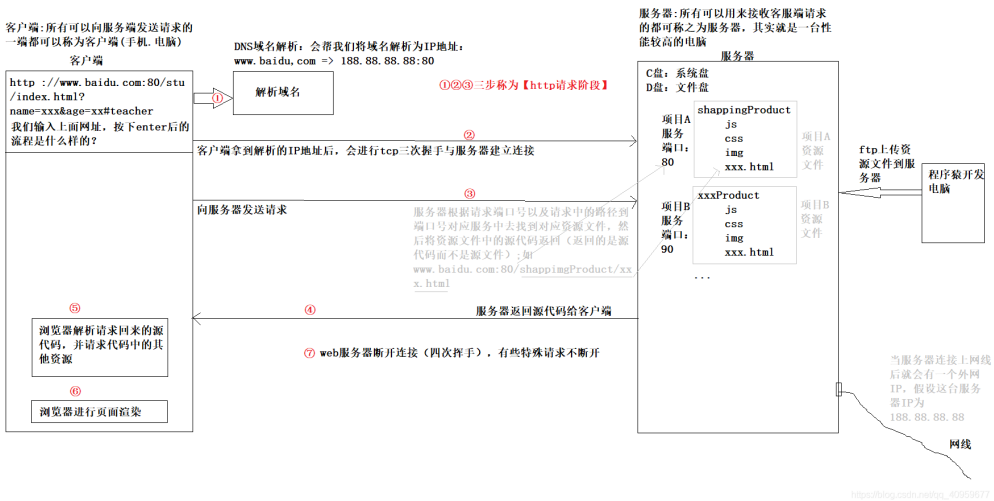
爬虫

一 、网络技术

HTTP和HTTPS

HTTP协议（HyperText Transfer Protocol，超文本传输协议）：是一种发布和接收 HTML页面的方法。
HTTPS（Hypertext Transfer Protocol over Secure Socket Layer）简单讲是HTTP的安全版，在HTTP下加入SSL层。
SSL（Secure Sockets Layer 安全套接层）主要用于Web的安全传输协议，在传输层对网络连接进行加密，保障在Internet上数据传输的安全。

浏览器发送HTTP请求的过程：



1.URL解析/DNS解析查找域名IP地址
2.网络连接发起HTTP请求
3.HTTP报文传输过程
4.服务器接收数据
5.服务器响应请求/MVC
6.服务器返回数据
7.客户端接收数据
8.浏览器加载/渲染页面

URL（Uniform / Universal Resource Locator的缩写）

定义：统一资源定位符，是用于完整地描述Internet上网页和其他资源的地址的一种标识方法。
基本格式：`scheme://host[:port#]/path/.../[?query-string][#anchor]`

- scheme：协议(例如：http, https, ftp)

- host：服务器的IP地址或者域名
- port#：服务器的端口（如果是走协议默认端口，缺省端口80）
- path：访问资源的路径
- query-string：参数，发送给http服务器的数据
- anchor：锚（跳转到网页的指定锚点位置）

客户端HTTP请求

- URL只是标识资源的位置，而HTTP是用来提交和获取资源。客户端发送一个HTTP请求到服务器的请求消息，包括以下格式：
  请求行、请求头部、空行、请求数据

一个典型的HTTP请求

1. Host (主机和端口号)

Host：对应网址URL中的Web名称和端口号，用于指定被请求资源的Internet主机和端口号，通常属于URL的一部分。

2. Connection (链接类型)

Connection：表示客户端与服务连接类型

Client 发起一个包含 Connection:keep-alive 的请求，HTTP/1.1使用 keep-alive 为默认值。

Server收到请求后：

如果 Server 支持 keep-alive，回复一个包含 Connection:keep-alive 的响应，不关闭连接；

如果 Server 不支持 keep-alive，回复一个包含 Connection:close 的响应，关闭连接。

如果client收到包含 Connection:keep-alive 的响应，向同一个连接发送下一个请求，直到一方主动关闭连接。

keep-alive在很多情况下能够重用连接，减少资源消耗，缩短响应时间，比如当浏览器需要多个文件时(比如一个HTML文件和相关的图形文件)，不需要每次都去请求建立连接。

3. Upgrade-Insecure-Requests (升级为HTTPS请求)

Upgrade-Insecure-Requests：升级不安全的请求，意思是会在加载 http 资源时自动替换成 https 请求，让浏览器不再显示https页面中的http请求警报。

HTTPS 是以安全为目标的 HTTP 通道，所以在 HTTPS 承载的页面上不允许出现 HTTP 请求，一旦出现就是提示或报错。

4. User-Agent (浏览器名称)

User-Agent：是客户浏览器的名称，以后会详细讲。
5. Accept (传输文件类型)

Accept：指浏览器或其他客户端可以接受的MIME（Multipurpose Internet Mail Extensions（多用途互联网邮件扩展））文件类型，服务器可以根据它判断并返回适当的文件格式。
举例：

Accept: */*：表示什么都可以接收。

Accept：image/gif：表明客户端希望接受GIF图像格式的资源；

Accept：text/html：表明客户端希望接受html文本。

Accept: text/html, application/xhtml+xml;q=0.9, image/*;q=0.8：表示浏览器支持的 MIME 类型分别是 html文本、xhtml和xml文档、所有的图像格式资源。

q是权重系数，范围 0 =< q <= 1，q 值越大，请求越倾向于获得其“;”之前的类型表示的内容。若没有指定q值，则默认为1，按从左到右排序顺序；若被赋值为0，则用于表示浏览器不接受此内容类型。

Text：用于标准化地表示的文本信息，文本消息可以是多种字符集和或者多种格式的；
Application：用于传输应用程序数据或者二进制数据。详细请点击
6. Referer (页面跳转处)

Referer：表明产生请求的网页来自于哪个URL，用户是从该 Referer页面访问到当前请求的页面。这个属性可以用来跟踪Web请求来自哪个页面，是从什么网站来的等。

有时候遇到下载某网站图片，需要对应的referer，否则无法下载图片，那是因为人家做了防盗链，原理就是根据referer去判断是否是本网站的地址，如果不是，则拒绝，如果是，就可以下载；
7. Accept-Encoding（文件编解码格式）

Accept-Encoding：指出浏览器可以接受的编码方式。编码方式不同于文件格式，它是为了压缩文件并加速文件传递速度。浏览器在接收到Web响应之后先解码，然后再检查文件格式，许多情形下这可以减少大量的下载时间。
举例：Accept-Encoding:gzip;q=1.0, identity; q=0.5, *;q=0

如果有多个Encoding同时匹配, 按照q值顺序排列，本例中按顺序支持 gzip, identity压缩编码，支持gzip的浏览器会返回经过gzip编码的HTML页面。 如果请求消息中没有设置这个域服务器假定客户端对各种内容编码都可以接受。
8. Accept-Language（语言种类）

Accept-Langeuage：指出浏览器可以接受的语言种类，如en或en-us指英语，zh或者zh-cn指中文，当服务器能够提供一种以上的语言版本时要用到。
9. Accept-Charset（字符编码）

Accept-Charset：指出浏览器可以接受的字符编码。
举例：Accept-Charset:iso-8859-1,gb2312,utf-8

ISO8859-1：通常叫做Latin-1。Latin-1包括了书写所有西方欧洲语言不可缺少的附加字符，英文浏览器的默认值是ISO-8859-1.

gb2312：标准简体中文字符集;

utf-8：UNICODE 的一种变长字符编码，可以解决多种语言文本显示问题，从而实现应用国际化和本地化。

如果在请求消息中没有设置这个域，缺省是任何字符集都可以接受。

10. Cookie （Cookie）

Cookie：浏览器用这个属性向服务器发送Cookie。Cookie是在浏览器中寄存的小型数据体，它可以记载和服务器相关的用户信息，也可以用来实现会话功能，以后会详细讲。

11. Content-Type (POST数据类型)

Content-Type：POST请求里用来表示的内容类型。

举例：Content-Type = Text/XML; charset=gb2312：

指明该请求的消息体中包含的是纯文本的XML类型的数据，字符编码采用“gb2312”。

常用请求报头

HTTP请求方法

序号	方法	描述
1	GET	请求指定的页面信息，并返回实体主体。
2	HEAD	类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头
3	POST	向指定资源提交数据进行处理请求（例如提交表单或者上传文件），数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。
4	PUT	从客户端向服务器传送的数据取代指定的文档的内容。
5	DELETE	请求服务器删除指定的页面。
6	CONNECT	HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。
7	OPTIONS	允许客户端查看服务器的性能。
8	TRACE	回显服务器收到的请求，主要用于测试或诊断。

主要方法get和post请求

- GET是从服务器上获取数据，POST是向服务器传送数据
- GET请求参数显示，都显示在浏览器网址上，HTTP服务器根据该请求所包含URL中的参数来产生响应内容，即“Get”请求的参数是URL的一部分。 例如： `http://www.baidu.com/s?wd=Chinese`
- POST请求参数在请求体当中，消息长度没有限制而且以隐式的方式进行发送，通常用来向HTTP服务器提交量比较大的数据（比如请求中包含许多参数或者文件上传操作等），请求的参数包含在“Content-Type”消息头里，指明该消息体的媒体类型和编码.

HTTP响应状态码

略

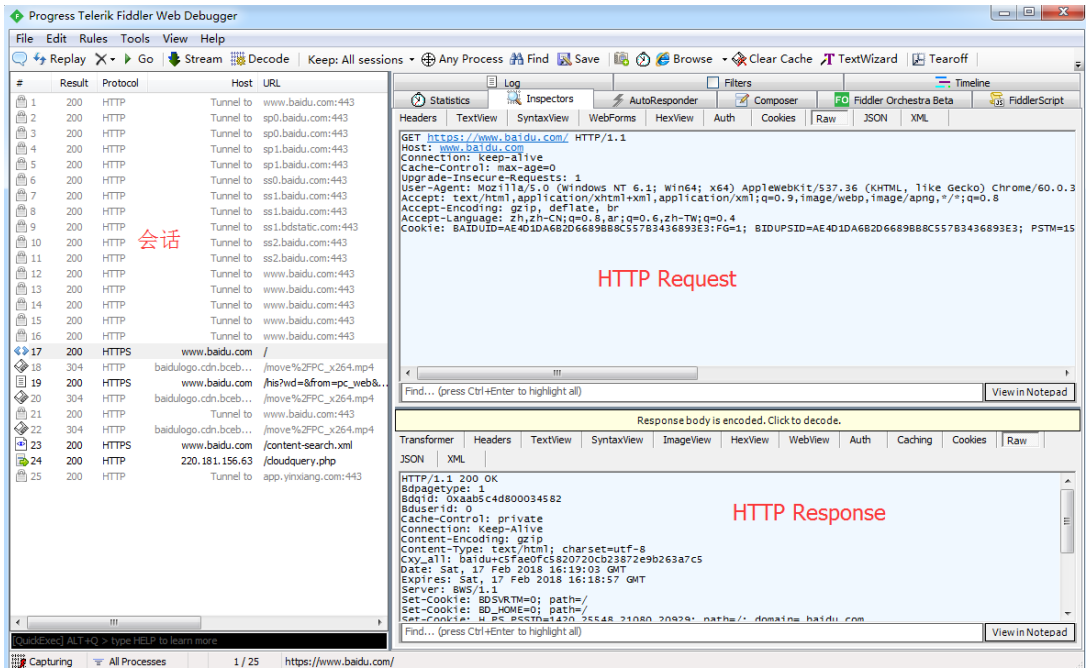
浏览器内核

浏览器	内核

IE	Trident
Chrome	Webkit
Firefox	Gecho
Opera	Pesto
Safari(Apple)	Webkit

HTTP代理工具Fiddler

Fiddler是一款强大Web调试工具，它能记录所有客户端和服务器的HTTP请求.



Request部分详解

> Headers —— 显示客户端发送到服务器的 HTTP 请求的 header，显示为一个分级视图，包含了 Web 客户端信息、Cookie、传输状态等。
> Textview —— 显示 POST 请求的 body 部分为文本。
> WebForms —— 显示请求的 GET 参数 和 POST body 内容。
> HexView —— 用十六进制数据显示请求。
> Auth —— 显示响应 header 中的 Proxy-Authorization(代理身份验证) 和 Authorization(授权) 信息.
> Raw —— 将整个请求显示为纯文本。
> JSON - 显示JSON格式文件。
> XML —— 如果请求的 body 是 XML 格式，就是用分级的 XML 树来显示它。

Responser部分详解

> Transformer —— 显示响应的编码信息。
> Headers —— 用分级视图显示响应的 header。
> TextView —— 使用文本显示相应的 body。
> ImageVies —— 如果请求是图片资源，显示响应的图片。
> HexView —— 用十六进制数据显示响应。
> WebView —— 响应在 Web 浏览器中的预览效果。
> Auth —— 显示响应 header 中的 Proxy-Authorization(代理身份验证) 和 Authorization(授权) 信息。
> Caching —— 显示此请求的缓存信息。
> Privacy —— 显示此请求的私密 (P3P) 信息。
> Raw —— 将整个响应显示为纯文本。

> JSON - 显示JSON格式文件。
> XML —— 如果响应的 body 是 XML 格式，就是用分级的 XML 树来显示它 。