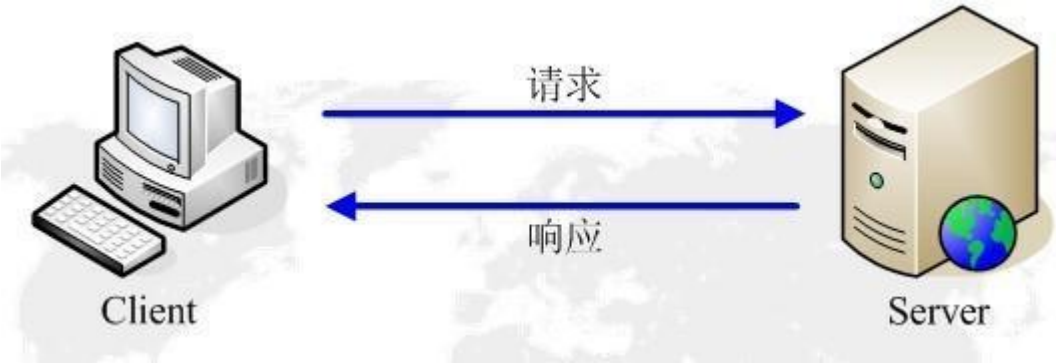
HTTP 协议

HTTP 简介

超文本传输协议（Hyper Text Transfer Protocol, HTTP）规定客户端如何从服务器端获取文档和向服务器端提交表单内容，以及服务器端如何响应这些请求和提交。



URL

统一资源定位符（Uniform Resource Locator, URL）是互联网上用来标识某一处资源的地址。以下面这个 URL 为例，介绍下普通 URL 的各部分组成：

```
http://www.haogu.com:8080/news/index.php?boardID=5&ID=24618&page=1#name
```

从上面的 URL 可以看出，一个完整的 URL 包括以下几部分：

1. 协议部分：该 URL 的协议部分为“http：”，这代表网页使用的是 HTTP 协议。在 Internet 中可以使用多种协议，如 HTTP，FTP 等等本例中使用的是 HTTP 协议。在"HTTP"后面的“//”为分隔符。

2. 域名部分：该 URL 的域名部分为“www.haogu.com”。一个 URL 中，也可以使用 IP 地址作为域名使用。

3. 端口部分：跟在域名后面的是端口，域名和端口之间使用“:”作为分隔符。端口不是一个 URL 必须的部分，如果省略端口部分，将采用默认端口 80

4. 虚拟目录部分：从域名后的第一个“/”开始到最后一个“/”为止，是虚拟目录部分。虚拟目录也不是一个 URL 必须的部分。本例中的虚拟目录是“/news/”

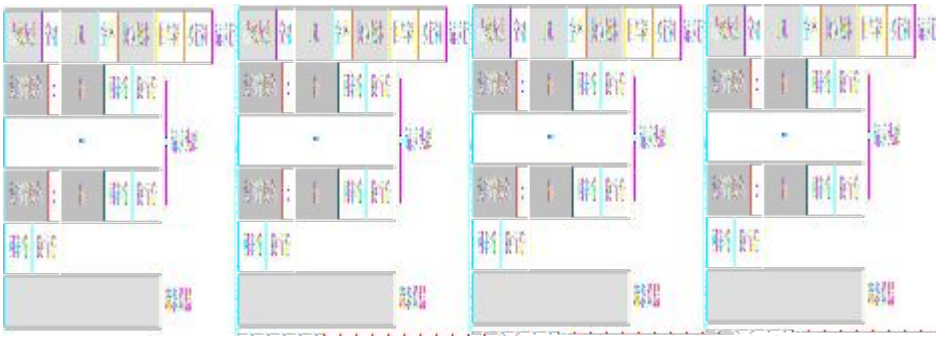
5. 文件名部分：从域名后的最后一个“/”开始到“？”为止，是文件名部分，如果没有“?”,则是从域名后的最后一个“/”开始到“#”为止，是文件部分，如果没有“？”和“#”，那么从域名后的最后一个“/”开始到结束，都是文件名部分。本例中的文件名是“index.asp”。文件名部分也不是一个 URL 必须的部分，如果省略该部分，则使用默认的文件名。

6. 锚部分：从“#”开始到最后，都是锚部分。本例中的锚部分是“name”。锚部分也不是一个 URL 必须的部分。

7. 参数部分：从“？”开始到“#”为止之间的部分为参数部分，又称搜索部分、查询部分。本例中的参数部分为“username=lisi&password=24618&page=1”。参数可以允许有多个参数，参数与参数之间用“&”作为分隔符。

请求消息 Request

客户端发送给服务器端的 HTTP 请求消息由**请求行（request line）**、**请求头部（header）**、**空行**和**请求数据**四个部分组成，如下图所示。



```
GET /index.html HTTP/1.1
Host: www.baidu.com
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36
Accept:text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9
Cookie:BAIDUID=002E605A2EE0AA27D0B7C2295B9D0242:FG=1;BIDUPSID=002E605A2EE0AA27D0B7C2295B9D0242;PSTM=1532306024;BD_UPN=12314753;BDORZ=B490B5EBF6F3CD402E515D22BCDA1598;delPer=0;BD_HOME=0;H_PS_PSSID=1465_26911_21121_26350_26925_20719
```

第一部分：请求行，用来说明请求类型，要访问的资源以及所使用的 HTTP 版本.

请求行以一个方法符号开头，以空格分开，后面跟着请求的 URI 和协议的版本。GET 指定请求类型为 GET，/index.html 为要访问的资源，该行的最后一部分说明使用的是 HTTP1.1 版本。

第二部分：请求头部，紧接着请求行（即第一行）之后的部分，用来说明服务器要使用的附加信息。

从第二行起为请求头部，HOST 将指出请求的目的地。User-Agent，服务器端和客户端脚本都能访问它,它是浏览器类型检测逻辑的重要基础.该信息由你的浏览器来定义,并且在每个请求中自动发送等等。

第三部分：空行，请求头部后面的空行是必须要有的

即使第四部分的请求数据为空，也必须有空行。

第四部分：请求数据也叫主体，可以添加任意的其他数据。

这个例子的请求数据为空。

POST 请求例子，使用 Charles 抓取的 request：

```
POST /api/students HTTP/1.1
Host: 192.168.0.130:3000
Connection: keep-alive
Content-Length: 245
Cache-Control: max-age=0
Origin: http://192.168.0.130:3000
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36
Accept:text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://192.168.0.130:3000/students/create
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9

clazz=%E7%81%AB%E8%8A%B112%E6%9C%9F&name=%E9%99%88%E6%A2%A6%E9%BE%99&gender=%E7%94%B7&age=20&hobby=%E7%9D%A1%E8%A7%89&hobby=%E6%89%93%E8%B1%86%E8%B1%86&tel=13834569928&address=%E5%8D%97%E9%98%B3&remark=%E6%96%B0%E5%90%8C%E5%AD%A6&date=2018-08-01
```

第一部分：请求行，第一行明了是 post 请求，以及 http1.1 版本。
第二部分：请求头部，第二行至第六行。
第三部分：空行，第七行的空行。
第四部分：请求数据，第八行。

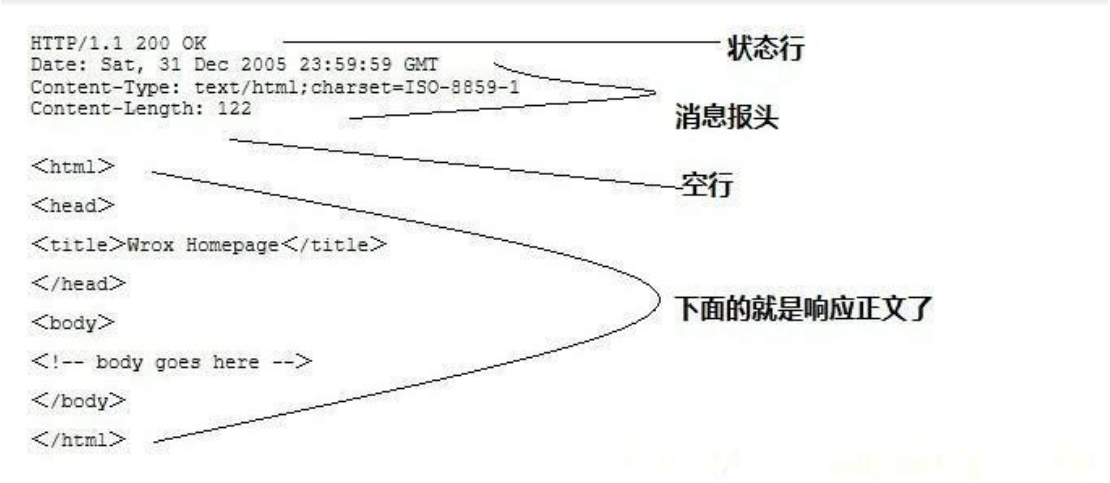
响应消息 Response

一般情况下，服务器接收到请求后，会对请求进行处理，然会返回给客户端一个 HTTP 的响应消息。HTTP 响应也由四个部分组成，分别是：**状态行、消息报头、空行和响应正文**。

```
HTTP/1.1 200 OK
Bdpagetype: 1
Bdqid: 0xc347763400004282
Cache-Control: private
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html
Cxy_all: baidu+5e802392ce41b42ad2771898f1006759
Date: Thu, 09 Aug 2018 01:36:14 GMT
```

```
Expires: Thu, 09 Aug 2018 01:36:06 GMT
Server: BWS/1.1
Set-Cookie: delPer=0; expires=Sat, 01-Aug-2048 01:36:06 GMT
Set-Cookie: BDSVRTM=0; path=/
Set-Cookie: BD_HOME=0; path=/
Set-Cookie:                 H_PS_PSSID=1465_26964_21121_26350_26925_20719;path=/;domain=.baidu.com
Strict-Transport-Security: max-age=172800
Vary: Accept-Encoding
X-Ua-Compatible: IE=Edge,chrome=1
Transfer-Encoding: chunked

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>百度一下</title>
</head>
<body>
  文档内容。。。
</body>
</html>
```



第一部分：状态行，由 HTTP 协议版本号， 状态码， 状态消息 三部分组成。

第一行为状态行，（HTTP/1.1）表明 HTTP 版本为 1.1 版本，状态码为 200，状态消息为（ok）

第二部分：消息报头，用来说明客户端要使用的一些附加信息

第二行和第三行为消息报头，Date:生成响应的日期和时间；Content-Type:指定了 MIME 类型的 HTML(text/html)，编码类型是 UTF-8

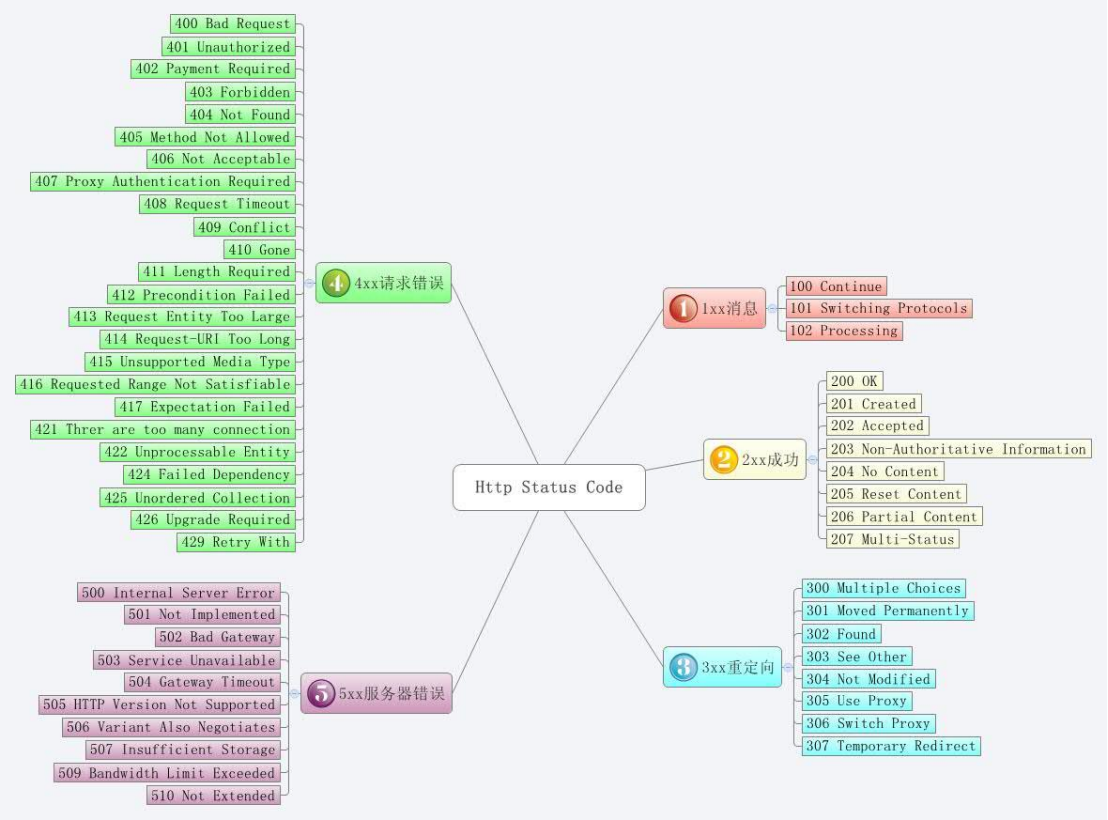
第三部分：空行，消息报头后面的空行是必须的

第四部分：响应正文，服务器返回给客户端的文本信息。

空行后面的 html 部分为响应正文。

HTTP 状态码

状态代码有三位数字组成，第一个数字定义了响应的类别，共分五种类别:



1xx：指示信息--表示请求已接收，继续处理
2xx：成功--表示请求已被成功接收、理解、接受
3xx：重定向--要完成请求必须进行更进一步的操作
4xx：客户端错误--请求有语法错误或请求无法实现
5xx：服务器端错误--服务器未能实现合法的请求

常见状态码:

```
200 OK                          //客户端请求成功
400 Bad Request                 //客户端请求有语法错误，不能被服务器所理解
401 Unauthorized                //请求未经授权，这个状态代码必须和 WWW-Authenticate 报头域一起使用
403 Forbidden                   //服务器收到请求，但是拒绝提供服务
404 Not Found                   //请求资源不存在，eg：输入了错误的 URL
500 Internal Server Error       //服务器发生不可预期的错误
503 Server Unavailable          //服务器当前不能处理客户端的请求，一段时间后可能恢复正常
```

HTTP 请求方法

根据 HTTP 标准，HTTP 请求可以使用多种请求方法。

HTTP1.0 定义了三种请求方法： GET, POST 和 HEAD 方法。
HTTP1.1 新增了五种请求方法：OPTIONS, PUT, DELETE, TRACE 和 CONNECT 方法。

- GET 请求指定的页面信息，并返回实体内容。
- 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
- 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。
- PUT 从客户端向服务器传送的数据取代指定的文档的内容。
- DELETE 请求服务器删除指定的页面。
- 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。
- OPTIONS 允许客户端查看服务器的性能。
- TRACE 回显服务器收到的请求，主要用于测试或诊断。

我们最常用的请求方法为 GET 和 POST。

GET 和 POST 请求的区别

发送数据的方式

GET 提交，请求的数据会附在 URL 之后（就是把数据放置在 HTTP 协议头中），以?分割 URL 和传输数据，多个参数用&连接；例 如：login.action?name=hyddd&password=idontknow&verify=%E4%BD%A0 %E5%A5%BD。如果数据是英文字母或者数字，原样发送，如果是空格，转换为+，如果是中文/其他字符，则直接把字符串用 BASE64 加密，得出如： %E4%BD%A0%E5%A5%BD，其中％XX 中的 XX 为该符号以 16 进制表示的 ASCII，Unicode。

POST 提交：把提交的数据放置在是 HTTP 数据包的请求体中。上文示例中红色字体标明的就是实际的传输数据。因此，GET 提交的数据会在地址栏中显示出来，而 POST 提交，地址栏不会改变。

传输数据的大小：

首先声明：HTTP 协议没有对传输的数据大小进行限制，HTTP 协议规范也没有对 URL 长度进行限制。
而在实际开发中存在的限制主要有：GET:特定浏览器和服务器对 URL 长度有限制，例如 IE 对 URL 长度的限制是 2083 字节(2K+35)。对于其他浏览器，如 Netscape、FireFox 等，理论上没有长度限制，其限制取决于操作系统的支持。因此对于 GET 提交时，传输数据就会受到 URL 长度的限制。

POST:由于不是通过 URL 传值，理论上数据不受限。但实际各个 WEB 服务器会规定对 post 提交数据大小进行限制，Apache、IIS6 都有各自的配置。

安全性

POST 的安全性要比 GET 的安全性高。比如：通过 GET 提交数据，用户名和密码将明文出现在 URL 上，因为(1)登

录页面有可能被浏览器缓存; (2)其他人查看浏览器的历史纪录, 那么别人就可以拿到你的账号和密码了, 除此之外, 使用 GET 提交数据还可能会造成 Cross-site request forgery 攻击。

POST 请求发送的数据放到请求体中，相对来说比较安全。

HTTP 工作原理

HTTP 协议定义 Web 客户端如何从 Web 服务器请求 Web 页面，以及服务器如何把 Web 页面传送给客户端。HTTP 协议采用了请求/响应模型。客户端向服务器发送一个请求报文，请求报文包含请求的方法、URL、协议版本、请求头部和请求数据。服务器以一个状态行作为响应，响应的内容包括协议的版本、成功或者错误代码、服务器信息、响应头部和响应数据。

以下是 HTTP 请求/响应的步骤:

1、客户端连接到 Web 服务器

一个 HTTP 客户端，通常是浏览器，与 Web 服务器的 HTTP 端口（默认为 80）建立一个 TCP 套接字连接。

2、发送 HTTP 请求

通过 TCP 套接字连接，客户端向 Web 服务器发送一个文本的请求消息，一个请求消息由请求行、请求头部、空行和请求数据 4 部分组成。

3、服务器接受请求并返回 HTTP 响应

Web 服务器解析请求消息，定位请求资源。服务器将资源副本写到 TCP 套接字，由客户端读取。一个响应由状态行、响应头部、空行和响应数据 4 部分组成。

4、释放 TCP 连接

若 connection 模式为 close，则服务器主动关闭 TCP 连接，客户端被动关闭连接，释放 TCP 连接;若 connection 模式为 keep alive，则该连接会保持一段时间，在该时间内可以继续接收请求;

5、客户端浏览器解析 HTML 内容

客户端浏览器首先解析状态行，查看状态代码，判断请求是否成功。然后解析每一个响应头，响应头告知以下为若干字节的 HTML 文档和文档的字符集。客户端浏览器读取响应数据 HTML，根据 HTML 的语法对其进行格式化，并在浏览器窗口中显示。

例如：在浏览器地址栏键入 URL，按下回车之后会经历以下流程:

1、浏览器向 DNS 服务器请求解析该 URL 中的域名所对应的 IP 地址;
2、解析出 IP 地址后，根据该 IP 地址和默认端口 80，和服务器建立 TCP 连接;
3、浏览器发出读取文件(URL 中域名后面部分对应的文件)的 HTTP 请求，该请求消息作为 TCP 三次握手的第三个报文的数据发送给服务器;
4、服务器对浏览器请求作出响应，并把对应的 html 文本发送给浏览器

5、释放 TCP 连接;
6、浏览器加载该 html 文本并显示内容;