HTTP协议类 HTTP协议的主要特点

简单快速　灵活　无连接　无状态

资源独立，访问某个资源输入URL

可以完成不同数据类型传输

连接一次就会断开

客户端和服务端是两种身份

HTTP协议类 HTTP 报文的组成部分

请求示例

▼ Request Headers　view parsed

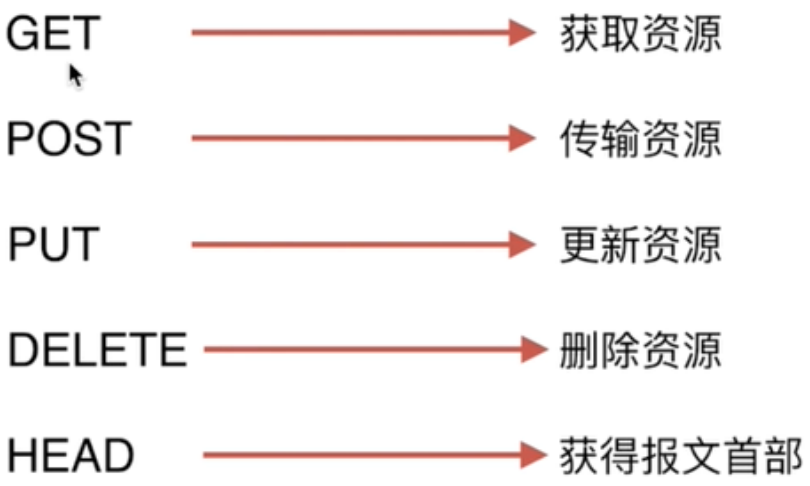
```
GET / HTTP/1.1
Host: www.imooc.com
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
DNT: 1
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8,en;q=0.6,ja;q=0.4
Cookie: imooc_uuid=9371e839-7a77-4193-bf11-f7a1860e530f; imooc_isnew_ct=1485229321; UM_distinctid=15b869e95feef-053c6486673a69-39687804-fa000-15b
869e9601d96; CNZZDATA1261110065=1104659954-1493298761-null%7C1493298761; Hm_lvt_c1c5f01e0fc4d75fd5cbb16f2e713d56=1493302513; Hm_lvt_f0cfcccd7b139
3990c78efdeebff3968=1494255721,1494256124,1494513448,1495377843; loginstate=1; apsid=NhNTU3ZDhhNGU4NWUyM2QyZWE2MmNkNDA1NGZlYWYAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAANTAyNzgxMgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA3MDY1Njk5MDZAcXEuY29tAAAAAAAAAAAAAAA
AAAAAADM5OGFjNTgyY2FkZWYxOTU1YWRkZjNmMTFiNmM1NTM4fhVVWX4VVVk%3DYj; last_login_username=706569906%40qq.com; mc_channel=banner1; mc_marking=afc221c
9f8cd2f1e0035b7a928e6252f; PHPSESSID=prsg0ood6b7c9ssjrmjtueabs4; IMCDNS=0; imooc_isnew=2; cvde=5972291c47ce0-104
```

HTTP协议类 HTTP 报文的组成部分

响应示例

```
HTTP/1.1 200 OK
Server: nginx
Date: Sat, 22 Jul 2017 15:55:01 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept-Encoding
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Set-Cookie: imooc_isnew=2; expires=Sun, 22-Jul-2018 15:55:00 GMT; path=/; domain=.imooc.com
Set-Cookie: cvde=5972291c47ce0-105; path=/; domain=.imooc.com
Content-Encoding: gzip
```

HTTP协议类 HTTP 方法



HTTP协议类 POST和GET的区别

- GET在浏览器回退时是无害的，而POST会再次提交请求
- GET产生的URL地址可以被收藏，而POST不可以
- GET请求会被浏览器主动缓存，而POST不会，除非手动设置
- GET请求只能进行url编码，而POST支持多种编码方式
- GET请求参数会被完整保留在浏览器历史记录里，而POST中的参数不会被保留
- GET请求在URL中传送的参数是有长度限制的，而POST没有限制
- 对参数的数据类型，GET只接受ASCII字符，而POST没有限制
- GET比POST更不安全，因为参数直接暴露在URL上，所以不能用来传递敏感信息
- GET参数通过URL传递，POST放在Request body中

HTTP协议类 HTTP状态码

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

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：服务器错误 – 服务器未能实现合法的请求

HTTP协议类 HTTP状态码

200 OK：客户端请求成功

206 Partial Content：客户发送了一个带有Range头的GET请求，服务器完成了它

301 Moved Permanently：所请求的页面已经转移至新的url

302 Found：所请求的页面已经临时转移至新的url

304 Not Modified：客户端有缓冲的文档并发出了一个条件性的请求，服务器告诉客户，原来缓冲的文档还可以继续使用

HTTP协议类 HTTP状态码

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：对被请求页面的访问被禁止

404 Not Found：请求资源不存在

500 Internal Server Error：服务器发生不可预期的错误原来缓冲的文档还可以继续使用

503 Server Unavailable：请求未完成，服务器临时过载或当机，一段时间后可能恢复正常

HTTP协议类 持久连接

HTTP 协议采用“请求-应答”模式，当使用普通模式，即非 Keep-Alive 模式时，每个请求/应答客户和服务器都要新建一个连接，完成之后立即断开连接（HTTP协议为无连接的协议）

当使用 Keep-Alive 模式（又称持久连接、连接重用）时，Keep-Alive 功能使客户端到服务器端的连接持续有效，当出现对服务器的后继请求时，Keep-Alive 功能避免了建立或者重新建立连接

HTTP协议类 管线化

在使用持久连接的情况下，某个连接上消息的传递类似于

请求1 -> 响应1 -> 请求2 -> 响应2 -> 请求3 -> 响应3

某个连接上的消息变成了类似这样

请求1 -> 请求2 -> 请求3 -> 响应1 -> 响应2 -> 响应3

HTTP协议类 管线化

- 管线化机制通过持久连接完成，仅 HTTP/1.1 支持此技术
- 只有 GET 和 HEAD 请求可以进行管线化，而 POST 则有所限制
- 初次创建连接时不应启动管线机制，因为对方（服务器）不一定支持 HTTP/1.1 版本的协议
- 管线化不会影响响应到来的顺序，如上面的例子所示，响应返回的顺序并未改变
- HTTP /1.1 要求服务器端支持管线化，但并不要求服务器端也对响应进行管线化处理，只是要求对于管线化的请求不失败即可
- 由于上面提到的服务器端问题，开启管线化很可能并不会带来大幅度的性能提升，而且很多服务器端和代理程序对管线化的支持并不好，因此现代浏览器如 Chrome 和 Firefox 默认并未开启管线化支持