# 网络部分

客户端与服务器交互采用Http协议，客户端采用OkHttp3框架，服务器采用SpringMVC+Mvbatis+MySQL框架

流程：

客户端发送Request-->收到Response信息-->分析Response信息

服务器侦听Request-->解析Request请求 --> 返回Response信息

## Http常用传输方式：

POST：

传输实体主体，即发送信息，

GET：

获取资源，即发送请求获取信息

二者区别主要是携带数据量不同,且GET为明文传输

PUT：

传输文件

PUT要求在请求报文的主体中包含文件内容，然后保存到请求URL指定的位置

处于安全考虑，一般web网站不使用此方法，若配合web的安全验证机制，或者架构采用REST标准的网站，就可能开放使用此方法

HEAD：

获得报文首部

HEAD和GET方法一样，只不过不返回报文主体部分，用于确认URI的有效性及资源更新的日期时间等

DELETE：

删除文件

DELETE是与PUT相反的方法，是按请求URI删除指定的资源

处于安全考虑，一般web网站不使用此方法，若配合web的安全验证机制，或者架构采用REST标准的网站，就可能开放使用此方法

OPTIONS：

询问支持的方法

用来查询针对请求URI指定的资源支持的方法

TRACE：

追踪路径

是让web服务器端将之前的请求通信还回给客户端的方法

发送请求时，在Max-Frowards首部字段中填入数值，每经过一个服务器端就-1，当数值为0时，停止传输，最后收到服务器返回状态码200 OK的响应

但是，这种方法基本很少使用，而且很容易引起XST（跨站追踪）攻击，就更不会用到了。

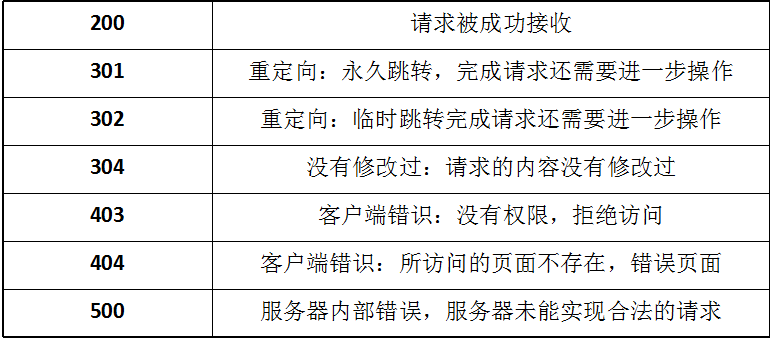
CONNECT：

要求采用隧道协议连接代理

该方法要求在于代理服务器通信时建立隧道，实现用隧道协议进行TCP通信，主要使用SSL（安全套接层）和TLS（传输层安全）协议把通信内容加密后经过网络传输。

## Http Status Code

|  | 已定义范围 | 分类 |
| --- | --- | --- |
| 1XX | 100-101 | 信息提示 |
| 2XX | 200-206 | 成功 |
| 3XX | 300-305 | 重定向 |
| 4XX | 400-415 | 客户端错误 |
| 5XX | 500-505 | 服务器错误 |



## URL 格式

 URL(Uniform Resource Locator) 地址用于描述一个网络上的资源,  基本格式如下

schema://host[:port#]/path/.../[?query-string][#anchor]

scheme               指定低层使用的协议(例如：http, https, ftp)

host                   HTTP服务器的IP地址或者域名

port#                 HTTP服务器的默认端口是80，这种情况下端口号可以省略。如果使用了别的端口，必须指明，例如 http://www.cnblogs.com:8080/

path                   访问资源的路径

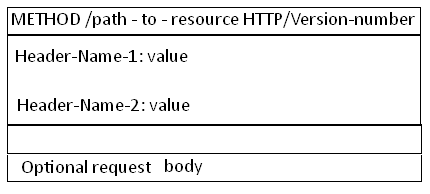
query-string       发送给http服务器的数据

anchor-             锚

## Http消息结构

Request

RequestHead



（1）METHOD：请求方法，比如POST/GET

Path-to-resource: 请求资源

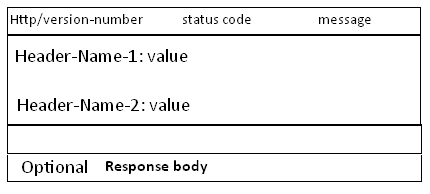
HTTP/Version-number:Http版本号

（2）Headers:该部分为Request的头部信息，包含有编码信息，请求客户端类型等等信息。

(3) Body：该部分含有Request的主体信息，与HTTP Request Header之间隔开一行。(OkHttp中使用RequestBody().builder()构建)

Response

ResponseHead



(1)Http/version-number:Http版本号

Status code:状态码，具体看上

Message:

(2)(3)和Request一致