# 1网络部分

网络交互较为成熟的有两种方式，其一是Socket，Socket是对TCP/UDP协议层的封装，也就是直接使用TCP/UDP协议传输数据，使用长连接

优点：可定制性强，灵活，直接以元数据方式传输便于解析，缺点：需要封装，使用麻烦

其二是http协议传输数据，也就是现在常用的互联网浏览器使用的方式，http是对为短连接，使用“请求-相应”方式√，使用简单

//如果有时间我在用Socket做。，。，

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（有个简化版Spring Boot）

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

## 1.1Http常用传输方式：

### 1.1.1POST：

传输实体主体，即发送信息，

### 1.1.2GET：

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

### 1.1.3PUT：

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

### 1.1.4HEAD：

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

### 1.1.5DELETE：

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

### 1.1.6OPTIONS：

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

### 1.1.7TRACE：

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。

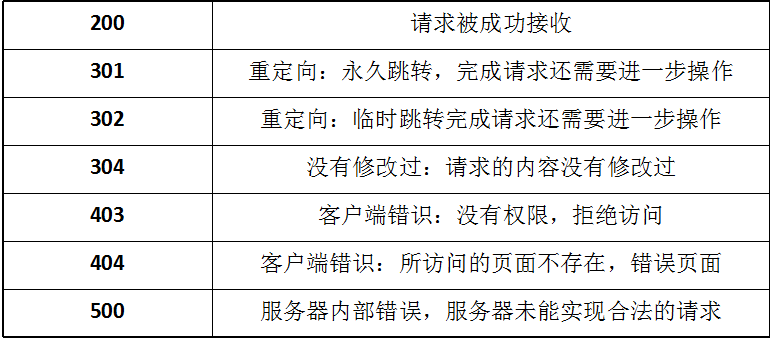
### 1.1.8CONNECT：

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。

## 1.2Http 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 |



## 1.3URL 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

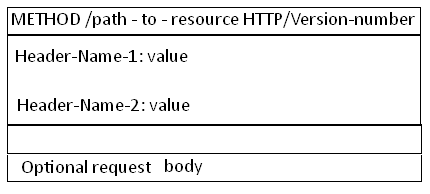
query-string       发送给http服务器的数据

anchor-             锚

## 1.4Http消息结构

### 1.4.1Request

RequestHead



（1）METHOD：请求方法，比如POST/GET

Path-to-resource: 请求资源

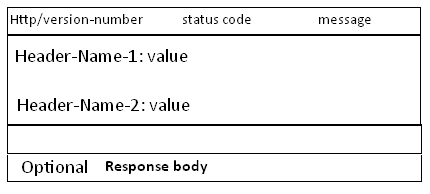
HTTP/Version-number:Http版本号

（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)

### 1.4.2Response

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

# 2线程部分

# 设计模式

## 单例设计模式：

有时候有些类只需要new一次即可，多new反而会有问题，这时候就需要单例设计模式

Singleton是一种创建型模式，指某个类采用Singleton模式，则在这个类被创建后，只可能产生一个实例供外部访问，并且提供一个全局的访问点。（注意和static类不同，当一个类中没有属性值，只是用来向外提供一个方法时，通常应使用单例模式，因为单例模式是面向对象的，可继承，可扩展。而且，最主要static类只能是内部类，有些情况是不适用的，顶级类不能是静态的）

例：

public class SingletonPattern{

public static SignletonPattern instance; //对外访问的实例

/\* 私有的构造方法，外部不可在new对象 \*/

private SignletonPattern{

}

/\*\*

\* 外部通过这个方法获取实例（这个方法负责创建保管实例）

\* 注意这个方法得是静态的，否则外部无法访问

\*/

public static void getInstance(){

if (instance == null){

//注意如果多线程使用的话需要线程同步，否则一个线程一个实例

instance = new SingletonPatton();

//线程同步

//synchronized (SingletonPatton.class){

// instance = new SingletonPatton();

//}

}

}

/\* 注意！单例模式类里的方法和属性最好用静态的\*/

}