http包的结构

http协议是用于从服务器传输超文本到本地浏览器位于传送协议，基于TCP/IP通信协议传递数据，数据的传递格式是http包。

http协议由请求和相应构成，总是由客户端向服务器发起请求，服务器收到请求想客户端返回响应。

1. http请求包的格式

客户端向服务器发送http的请求有三个部分构成：

- 起始行：请求方法+URL+协议/版本，对报文进行描述，后面接回车符+换行符

例子：GET /TEST/test.txt HTTP/1.1

- 头部：包含数据包的属性，可以为零个和多个

格式 { 字段：值 } 回车符+换行符

- 主体：请求正文

2. http的应答包

应答包的组成和请求包大致相同：

- 起始行：协议/版本+状态代码+状态描述

例子：HTTP/1.0 200 OK

- 头部
- 主体：应答正文

3. http的请求方法

- GET

最常用的方法，客户端请求服务器发送某个资源

- HEAD

和GET方法类似，但服务器只返回请求资源的头部，而不包含主体部分，使用HEAD可以在：捕获去资源的情况下了解资源、查看响应中的状态码判断某个对象是否存在、通过查看资源首部判断资源是否被修改

- PUT

与PUT方法相反，PUT请求会向服务器写入文档，请求服务器用数据包的主体部分创建或者替代（如果请求URL已经存在）一个由请求的URL命名的新文档

- POST

POST方法用来向服务器输入数据，在HTML表单中填好的数据会被发送给服务器，然后由服务器处理数据。

- TRACE

允许客户端查看请求的原始版本，判断数据包是否以及如何被修改过

- OPTIONS

请求web服务器告知其支持的各种功能。例如某些特殊资源支持哪些请求方法

- DELETE

请求服务器删除URL指定的资源，但是客户端无法宝恒删除操作一定会被执行。

4. 状态码

状态码方便的为客户端提供了服务器处理请求的结果

- 100-199——信息状态码

100：Continue，服务器收到请求的初始部分，请客户端继续。

101：Switching Protocal，服务器正在根据客户端的指定，将协议切换成Update首部所列的协议

- 200-299——成功状态码

200：OK，请求成功

201：Created，用于创建服务器对象的请求，返回这个请求说明对象创建成功

202：Accepted，请求已经被服务器接收，但是服务器还没有对请求执行任何动作。

203：Non-Authoritative Information，表示实体首部包含的信息不是来源于服务器本身，而是中间节点上面的副本

204：No Content，服务器的响应包包含若干首部和一个状态行，但是没有主体部分。

- 300-399——重定向状态码

300：Multiple Choices，客户端的请求指向多个URL

301：Moved Permanently，请求的URL已经被移除，响应包中的头部包含资源现在的URL

- 400-499——客户端错误状态码

400：Bad Request，告知客户端发送了一个错误的请求

401：Unauthorized，客户端需要认证

403：Forbidden，请求被服务器拒绝

404：Not Found，服务器无法找到所请求的URL

405：Method Not Allowed，请求方法不受服务器支持

- 500-599——服务器错误状态码

500：Internal Server Error，服务器收到阻碍

501 Not Implemented，请求超出服务器能力范围

502：Bad Gateway，代理或网关使用的服务器收到伪响应

4. 数据包首部

4.1 首部分类

- 通用首部：请求包和响应包都会包含

例子：Date： Tue, 3 Oct 1974 02：15:00 GMT

Connection：允许客户端和服务器指定与请求/响应连接有关的选项

Date：描述数据包创建的日期

MIME-Version：描述客户端的MIME版本

Trailer：数据包采用分块传输编码，用Trailer列出首部集合

Transfer-Encoding：数据包的编码格式

Update：客户端可能想要使用的新版本或新协议

Via：显示报文经过的中间节点

- 请求首部：提供更多有关请求的信息

例子：Accept: */*

Client-IP：客户端的IP地址

From：客户端的E-mail地址

Host：服务器的主机名和端口号

Referer：当前请求的URL

Accept：客户端能接受的mime

UA-Color：客户端显示的显示颜色

UA-CPU：客户端CPU的类型和制造商

UA-OS：客户端的操作系统和版本

User-Agent：发起请求的应用程序名称

- 响应首部：提供更多有关响应的信息

例子：Server: Tiki-Hut/1.0

Proxy-Connection：客户端和代理之间指定与连接有关的选项

Server：表示服务器的名字或者注释

- 实体首部：描述数据包主题的长度和内容

例子：Content-Type: text/html; charset=iso-latin-1

Content-Length：实体部分的长度

Content-type：实体对象的媒体类型

- 扩展首部：规范中没有定义的其他首部

Cookie：用于客户端识别和跟踪的扩展首部

4.2 首部延续行

较长的首部可以分为多行表示，多出来的行前面至少需要一个空格或者制表符

HTTP/1.0 200 OK

Content-Type: image/gif

Content-Length: 8527

Server: Test Server

 Version 1.0

Fiddler抓包

Fiddler专门捕获计算机与网络之间传送的http数据包，进行通过分析数据包可以查看接口是否调用正确，数据返回是否正确，还可以对http数据包进行重发、编辑和转存。

wireshark

wireshark可以截取链路层、网络层、传输层和应用层的所有数据包，并且显示网络数据包的详细信息。由于安全原因，wireshark只能查看网络包，不能修改。

1. 捕获过滤

wireshark捕获到的数据包数量非常大，为了便于分析，需要对对捕获的数据包进行过滤：

- IP过滤

ip.src eq 192.168.1.1 显示特定IP发来的数据包

ip.dst eq [ip-addr] 显示特定IP接收到的数据包

ip.addr == [ip-addr] 显示特定IP接受和发送的数据包

- 端口过滤

tcp.port == [port]

tcp.dstport == [port]

tcp.srcport == [port]

tcp.port >= [port] 过滤某范围的端口

- 协议过滤

直接输入协议名称： http/tcp/udp/ftp/icmp/ssl/dns/等

排除协议： !http 或者 not http

- 包长度过滤

tcp.len

udp.len

ip.len

frame.len

- http模式过滤

http.request.method == POST/GET/PUT... 按请求方法过滤

http.request.uri 按请求的资源标识符过滤

http contains "" 按http包中的内容过滤