1. 什么是http协议

HTTP协议是Hyper Text Transfer Protocol（超文本传输协议）的缩写,

是用于从万维网(WWW:World Wide We) 服务器传输超文本到本地浏览器的传送协议

HTTP是一个基于TCP/IP协议栈来传递数据(HTML 文件, 图片文件, 查询结果等)

HTTP是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体

信息系统

1. http协议的特点

简单快速: 客户端向服务端发起请求时, 只需要传输请求方法和路径, 常用的请求方法有: GET、POST、PUT、DELETE. 每种方法规定了客户与服务器联系的类型, 由于HTTP协议简单, 使得HTTP服务器的程序规模小, 因而通信速度很快

灵活: HTTP允许传输任意类型的数据对象, 正在传输的类型由Content-Type加以标记

无连接: 无连接的含义是限制每次连接只处理一个请求, 服务器处理完客户的请求, 并收到客户的响应后, 即断开连接, 采用这种方式可以节省传输时间

无状态: HTTP协议是无状态协议, 无状态是指协议对于事务处理没有记忆能力, 缺少状态意味着如果后续处理需要前面的信息, 则它必须重传, 这样可能导致每次连接传送的数据量增大

另一方面, 在服务器不需要先前信息时它的应答就较快

支持B/S, C/S模式:

B/S即Browser/Server, 浏览器/服务器

C/S即Client/Server, 客户端/服务器

1. http协议的作用?

HTTP协议是用于从WWW服务器传输超文本到本地浏览器的传送协议

它可以使浏览器更加高效, 使网络传输减少

它不仅保证计算机正确快速地传输超文本文档, 还确定传输文档中的哪一部分, 以及哪部分内容首先显示(如文本先于图形)等

1. http协议中的url:

我们在浏览器的地址栏里输入的网站地址叫做URL (Uniform Resource Locator)

uri: 统一资源标识符

url: 统一资源定位符 url是uri的子集

区别: <https://danielmiessler.com/study/url-uri/>

url的组成: 如: [http://www.exam.com:80/news/search.html?word=http#top](http://www.exam.com:80/search?word=http)

url由三部分组成:

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http后面的 “//” 为分隔符

1. 域名部分

该URL的域名部分为“www.exam.com”

在一个URL中，也可以使用IP地址作为域名使用

1. 端口部分

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URL必须的部分, 如果省略端口部分, 将采用默认端口

1. 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

虚拟目录不是URL必须的部分

1. 文件名部分

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, 是文件部分

文件名部分不是URL必须的部分

1. 锚部分

“#” (代表网页中的一个位置) 开始到最后, 都是锚部分, 锚部分不是URL必须的部分

1. 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

参数可以允许有多个参数, 参数与参数之间用 “&” 作为分隔符

参数部分不是URL必须的部分

1. http的request有什么?

客户端发送一个HTTP请求到服务器的请求消息包括以下格式:

请求行(request line)、请求头(header)、空行、请求数据四部分组成

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

由 请求类型, 请求URL, HTTP协议版本号 三部分组成

请求头: 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, 如

Host: 主机名

User-Agent: 浏览器基本资料

Accept: 浏览器能够识别的响应类型

Accept-Language: 浏览器默认语言

Accept-Encoding: 浏览器能够识别的压缩方式

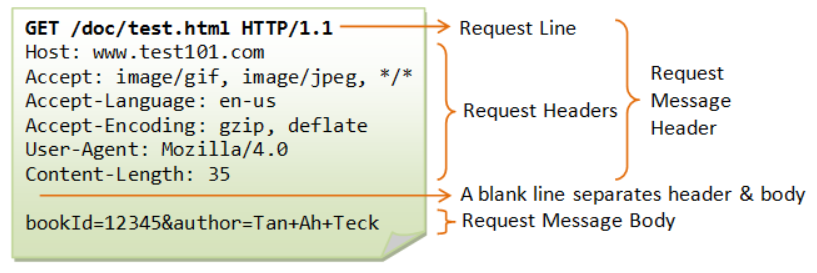
Referer: 来路页面

Connecton: 是否保持连接

空行: 请求头部后面的空行是必须的, 请求数据为空也必须有空行

请求数据: 请求数据也叫主体, 可以添加任意的其他数据, 可以为空

如图:



1. http的response有什么?

一般情况下, 服务器接收并处理客户端发过来的请求后会返回一个HTTP的响应消息

状态行、消息报头、空行和响应正文四部分组成

状态行: 由HTTP协议版本号, 状态码, 状态消息 三部分组成

消息报头: 用来说明客户端要使用的一些附加信息

Date: 响应的日期和时间

Content-Type: 指示响应的内容

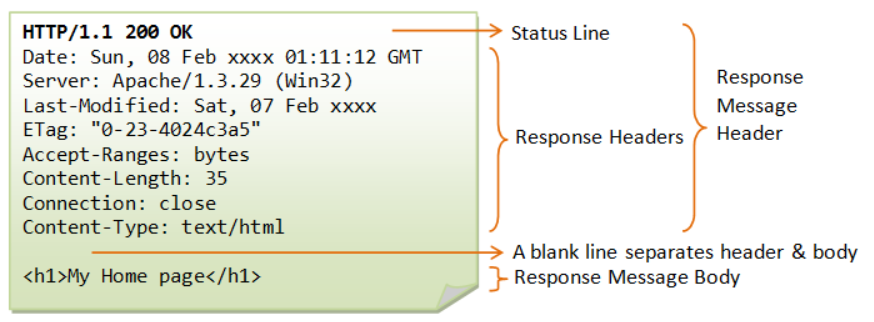
charset: 响应内容编码

Content-Length: 响应内容长度

空行: 消息报头后面的空行是必须的

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

如图:



1. http的请求方式有哪些?

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 获得资源: 请求指定的页面信息, 并返回实体主体

POST 传输实体主题: 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

HEAD 获取头部报文: 类似于GET请求, 返回的响应中没有具体的内容, 用于获取报头

PUT 上传文件: 从客户端向服务器传送的数据取代指定的文档的内容, 自身不带认证机制,

所以不安全

DELETE 删除文件: 请求服务器删除指定的页面, 与PUT相反, 也不带认证机制

PATCH 对资源进行部分修改: PUT也可以用于资源的修改, 但是会替换原来资源,

PATCH支持对资源的部分修改

OPTIONS 查询支持的方法: 查询指定URL支持的方法

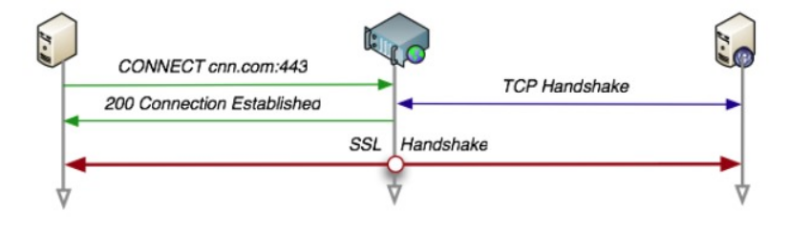
TRACE 追踪路径: 服务器会将通信路径返回给客户端, 请求时会在请求头插入Max-Forwards 字段, 每经过一个服务器都会减1, 到0停止传输, 可能会出现XST (Cross-Site Tracing)

跨站追踪攻击

CONNECT 与代理服务器通信时建立隧道: 使用 SSL (Secure Sockets Layer安全套接层) 和

TLS (Transport Layer Security传输层安全) 协议把通信内容加密后经网络隧道传输

CONNECT图:



1. http状态码有哪些?

1xx: Informational 信息状态码 -- 表示请求已接收, 继续处理

2xx: Success 成功状态码 -- 表示请求已被成功接收、理解、接受

3xx: Redirection 重定向状态码 -- 要完成请求必须进行更进一步的操作

4xx: Client Error 客户端错误状态码 -- 请求有语法错误或请求无法实现

5xx: Server Error 服务器错误状态码 -- 服务器未能实现合法的请求

100 Continue 表示当前请求正常, 可以继续请求或忽略

200 OK 客户端请求成功

204 No Content 请求已经处理, 返回的响应不包含实体部分 (head请求)

301 Moved Permanently 永久重定向

302 Found 临时重定向

303 See Other 和302几乎相同, 但是需要get请求

304 Not Modified 请求未改变, 使用缓存的数据

400 Bad Request 客户端请求有语法错误, 不能被服务器所理解

401 Unauthorized 认证失败

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

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

1. http首部

http有四种类型的首部字段: 通用首部字段、请求首部字段、响应首部字段、实体首部字段

通用首部字段:

Cache-Control 控制缓存的行为

Connection 控制不再转发给代理的首部字段, 管理持久连接

Date 创建报文的日期时间

Pragma 报文指令

Trailer 报文末端首部一览

Transfer-Encoding 指定报文主体传输的编码方式

Upgrade 升级为其他协议

Via 代理服务器相关信息

Warning 错误通知

...

请求首部字段:

Accept 用户代理可处理的类型

Accept-Charset 优先的字符集

Accept-Encoding 优先的内容编码

Accept-Language 优先的语言 (自然语言)

Authorization web认证信息

Expect 期待服务器的特定行为

From 用户电子邮箱地址

Host 请求资源所在服务器

If-Match 比较实体标记 (ETag)

If-Modified-Since 比较资源的更新时间

If-None-Match 比较实体标记 (与If-Match相反)

If-Range 资源未更新时, 发送实体Byte的请求范围

If-Unmodified-Since 比较资源的更新时间 (与If-Modified-Since相反)

Max-Forwards 最大传输逐条数

Proxy-Authorization 代理服务器要求客户端的认证信息

Range 实体字节请求范围

Referer 对请求URI的原始获取方

TE 传输编码的优先级别

User-Agent HTTP客户端的信息

...

响应首部字段:

Accept-Ranges 是否接受字节范围请求

age 推断资源创建经过时间

ETag 资源的匹配信息

Location 令客户端重定向到指定的URI

Proxy-Authenticate 代理服务器对客户端的认证信息

Retry-After 在此发起请求的时间要求

Server HTTP服务器的安装信息

Vary 代理服务器缓存的管理信息

WWW-Authenticate 服务器对客户端的认证信息

...

实体首部字段:

Allow 资源可支持的HTTP方法

Content-Encoding 实体主体适用的编码方式

Content-Language 实体主体的自然语言

Content-Length 实体主体的大小

Content-Location 替代对应资源的URI

Content-MD5 实体主体的报文摘要

Content-Range 实体主体的位置范围

Content-Type 实体主体的媒体类型

Expires 实体主体过期的日期时间

Last-Modified 资源最后修改的日期时间

...

1. http流水线是什么?

默认情况下, HTTP请求是按顺序发出的, 下一个请求只有在当前请求收到响应之后才会被出,

由于会受到网络延迟和带宽的限制, 在下一个请求被发送到服务器之前, 需要等待上一个请求

的响应, 这个时间可能会很长

流水线是在同一条长连接上发出连续的请求, 而不用等待响应返回, 这样可以避免连接延迟

1. http内容编码是什么?

内容编码是将内容主体压缩, 从而减少数据的传输量

常用的编码有: deflate、gzip、identity、compress

浏览器发送 Accept-Encoding 头信息, 其中包含有它所支持的压缩算法, 以及各自的优先级

服务器则从中选择一种, 使用该算法对响应的消息主体进行压缩, 并且发送 Content-Encoding

首部来告知浏览器它选择了哪一种算法, 由于该内容协商过程是基于编码类型来选择资源的展

现形式的, 在响应的 Vary 首部至少要包含 Content-Encoding

1. http请求范围是什么?

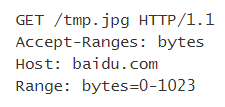
当网络传输过程中, 服务器只发送了一部分数据, 请求范围可以使客户端只请求服务端未发送 的数据, 避免服务端重新发送所有的数据

1. http请求范围如何设置?

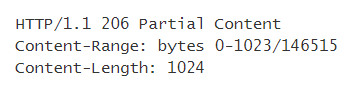
在请求报文添加Range和Accept-Ranges(可有可无) 字段,

Range指定请求范围,

Accept-Ranges用于告知客户端能否处理范围请求, 可以为bytes, 不可以为none



请求成功的话服务器返回206 Partial Content



1. http请求范围状态码有哪些?

范围请求成功: 206 Partial Content

范围请求失败(越界): 416 Requested Range Not Satisfiable

不支持范围请求: 200 OK

1. http的虚拟主机是什么?

HTTP/1.1添加了虚拟主机, 可以让一台服务器有多个域名, 达到逻辑上多台服务器的效果

1. http代理是什么?

代理服务器接收到请求从而转发给其他服务器

1. http代理的作用?

缓存

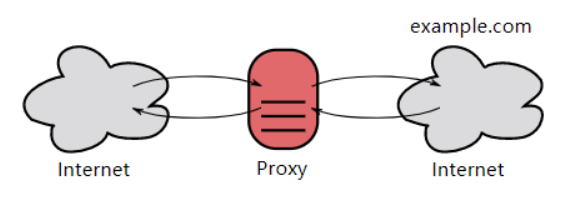
负载均衡

网络访问控制

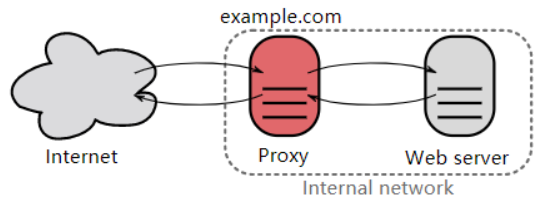
访问日志记录

1. 什么是正向代理和反向代理?

正向代理用户察觉的到:



反向代理位于内部网络用户察觉不到:



1. http网关是什么?

与代理服务器不同的是, 网关服务器会将http转化为其他协议进行通信, 从而请求其他非http服 务器的服务

1. http隧道是什么?

使用SSL等加密手段, 让客户端和服务端建立一条安全的通信线路

1. http请求方法的安全性:

安全的 HTTP 方法不会改变服务器状态, 也就是说它只是可读的

GET 方法是安全的, 而 POST 却不是, 因为 POST 的目的是传送实体主体内容, 这个内容可 能是用户上传的表单数据, 上传成功之后, 服务器可能把这个数据存储到数据库中, 因此状态也 就发生了改变

安全的方法除了 GET 之外还有HEAD、OPTIONS

不安全的方法除了 POST 之外还有 PUT、DELETE

1. http请求方法的幂等性:

幂等性: 同样的请求被执行一次与连续执行多次的效果是一样的, 服务器的状态也是一样的

所有的安全方法也都是幂等的, 在正常情况下: GET, HEAD, PUT 和 DELETE 等方法都是幂等 的, 而 POST 方法不是

1. http1.1的新特性:

默认长连接

支持流水线

支持同时打开多个 TCP 连接

支持虚拟主机

新增状态码 100 (Continue 表示当前请求正常, 请求初始化完毕)

支持分块传输编码

新增缓存处理指令 max-age

1. http1.x的缺陷有哪些:

客户端需要使用多个连接才能实现并发和缩短延迟

不会压缩请求和响应首部, 从而导致不必要的网络流量

不支持有效的资源优先级, 致使底层 TCP 连接的利用率低下

1. http的安全缺陷有哪些:

1 使用明文通信, 信息可能会被窃取

2 不认证通信方的身份, 通信方身份可能遭遇伪装

3 无法认证报文的完整性, 报文可能被篡改

1. http各个版本的区别?

0.9和1.0 使用非持续连接, 限制每次连接只处理一个请求, 服务器对客户端的请求做出响应后,

马上断开连接, 这种方式可以节省传输时间

1.1当前版本, 持久连接被默认采用, 并能很好地配合代理服务器工作, 还支持以管道方式同时发送多个请求, 以便降低线路负载, 提高传输速度

1.0和1.1的区别主要体现在:

1 缓存处理

2 带宽优化和网络连接的使用

3 错误通知的管理

4 消息在网络中的发送

5 互联网网址的维护

6 安全性及完整性

1. get和post的区别?
2. 提交方式不同

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),

如果数据是英文字母/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

如: %AA%E7%B2, 其中％XX中的XX为该符号以16进制表示的ASCII

最后一行为空行, 表示没有请求体

POST提交, 把提交的数据放置在是请求体中

因此, GET提交的数据会在URL中显示出来, 而POST提交, 不会显示到URL中

1. 传输数据的大小不同

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

GET: 特定浏览器和服务器对URL长度有限制

POST: 由于不是通过URL传值, 理论上数据不受限, 但实际各个WEB服务器会规定对提交数据大小进行限制

1. 安全性

POST的安全性要比GET的安全性高

如：通过GET提交数据, 用户名和密码将明文出现在URL上, 别人可以轻易的获得到你的 账号密码, 也可能会遭到CSRF攻击

1. http常用的三种Content-Type:

application/x-www-form-urlencoded: 数据被编码为名称/值对, 这是标准的编码格式, 最常用的一种格式

multipart/form-data: 数据被编码为一条消息, 页上的每个控件对应消息中的一个部分, 一般上传文件才会使用这种方式

text/plain: 数据以纯文本形式 (text/json/xml/html) 进行编码, 其中不含任何控件或格式字符, 一般向服务端发送json数据会使用这种方式

1. http和https的区别:

HTTP协议传输的数据都是未加密的, 也就是明文的, 因此使用HTTP协议传输隐私信息非常不安全, 为了保证这些隐私数据能加密传输, 于是网景公司设计了SSL (Secure Sockets Layer) 协议, 用于对HTTP协议传输的数据进行加密, 从而就诞生了HTTPS

1. http的Connection: Keep-Alive模式?

在HTTP1.0是默认close的, 在HTTP1.1默认启用了Keep-Alive模式

HTTP协议采用“请求-应答”模式, 在非Keep-Alive模式时, 每个请求/应答 客户端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, 启用Keep-Alive模式更高效, 性能更高

1. 长连接和短连接的区别?

当浏览器访问一个包含多张图片的 HTML 页面时, 除了请求访问 HTML 页面资源, 还会请求图片资源, 如果每进行一次 HTTP 通信就要新建一个 TCP 连接, 那么开销会很大, 长连接只需要建立一次 TCP 连接就能进行多次 HTTP 通信

才HTTP/1.1以前, 默认是短连接(Connection: close), 如果需要长连接, 可将Connection修改为Keep-Alive

从HTTP/1.1开始, 默认是长连接(Connection: Keep-Alive), 如果要断开连接, 需要服务器或客户端提出断开

1. Keep-Alive和WebSocket的区别:

WebSocket的H5提出的一个协议, 而Keep-Alive是将多个HTTP请求合并为一个

WebSocket协议和HTTP协议没有太大的关系

WebSocket是一个持久的协议, 而HTTP是一个非持久的协议

在HTTP/1.0中, HTTP只能发起一次请求接收一次响应

在HTTP/1.1后, HTTP可以发起多个请求, 接收多个响应, 但是他们是一一对应的, 并且不能主

动响应, 只能接收到请求后响应

在WebSocket中, 只要建立连接后, 服务端和客户端就能进行通信, 并可以主动发起, 而且不用

一一对应