项目HttpClient总结

一：了解Http协议

Http是超文本传输协议的缩写，是分布式，协作的，超媒体信息系统的应用层协议。

简单的说，HTTP协议就是请求/响应协议。

二：了解request-header域

Example:FireFox访问新浪抓包情况如下，

Request-header:

Host www.sina.com.cn

User-Agent Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN; rv:1.9.2.12) Gecko/20101026 Firefox/3.6.12

Accept text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

Accept-Language zh-cn,zh;q=0.5

Accept-Encoding gzip,deflate

Accept-Charset GB2312,utf-8;q=0.7,\*;q=0.7

Keep-Alive 115

Connection keep-alive

Cookie UOR=

Response-header:

Date Wed, 01 Dec 2010 09:00:25 GMT

Server Apache

Last-Modified Wed, 01 Dec 2010 09:00:07 GMT

Accept-Ranges bytes

X-Powered-By mod\_xlayout\_jh/0.0.1vhs.markIII.remix

Cache-Control max-age=60

Expires Wed, 01 Dec 2010 09:01:25 GMT

Vary Accept-Encoding

Content-Encoding gzip

X-UA-Compatible IE=EmulateIE7

Content-Type text/html

Age 47

Content-Length 115991

X-Cache HIT from nj75-165.sina.com.cn

Httpheader详解：

Accept: 响应可接受的媒体类型。如果不存在Accept头部域，表示客户端接收所有媒体类型。如果服务器不能依据Accept头部域的组合来发送任何可接受的响应，则返回406（No Acceptable）。Example,Accept:text/plain;q=0.5,text/html,text/x-dvi;q=0.8,text/x-c 优先级为：text/html——>text/x-c——>text/x-dvi——>text/plain

Accept-Charset:响应可接受的字符集。缺省可接受任何字符集。

Accept-Encoding：响应可接受的Content-Encoding。

Accept-Language：限制请求所选择的作为响应的自然语言集。

Accept-Ranges：允许服务器指出它对于某个资源请求的可接受范围。

Age：传送放对原始服务器生成响应以来所用总时间的估计值。

Allow：列出由Request-URI标识的资源所支持的方法集。Allow：GET，HEAD，PUT

Cache-Control：指定在请求/响应链上所有缓存机制服从的指令，简单的说就是控制网页缓存。值为private,no-cache,must-revalidate,打开新窗口访问时都会重新访问服务器，而如果指定了max-age的值，那么在此值内的时间里就不会重新访问服务器。Cache-Control:max-age=5.

Connection：允许发送方指定希望特定的链接选项。Connection：close 表示在完成响应后就会自动关闭连接。Connection：Keep-Alive 表示保持长连接。Http1.0默认是Connection:close；Http1.1默认是Connection:Keep-Alive 。

Content-Encoding：指出对entity-body的额外编码。允许文档被压缩。Content-Encoding：GZIP。

Content-Language：允许用户按用户自己所选的语言来标识并区分实体。

Content-Length：entity-body的二进制长度。

Content-Location：提供消息封装的实体的资源位置，由HTTP访问提供的URI地址的资源。

Content-MD5：实现对entity-body的完整性的检查。

Content-Range：断点下载的时候用到，用于指定整个实体中的一部分的插入位置，也指出了整个实体的长度。Bytes=100-300,意思是下载100 到 300 这部分数据。

Content-Type：指出发送方给接收方的entity-body的媒体类型，或者指出get请求将发送的媒体类型。

Date：表示消息发生的时间和日期。

ETag：用来判断请求的文件是否被修改。主要为了解决if-modify-since不能解决的问题。ETag可以是一个版本标记。

Except：指出客户端要求的特殊服务器行为。

Expires：表示存在时间，允许客户端在这个时间之前不去发请求，等同max-age的效果。但是如果同时存在，则被Cache –control的max-age覆盖。

From：发送邮件地址。

Host：被请求资源的主机和端口号。

If-Match：判断请求资源是否修改。

If-Modify-Since：所请求的资源在指定的时间以来不曾修改过，则服务器不会返回实体，而返回304.反之则返回200，并返回实体内容。

Last-Modify：服务器资源最后的修改时间。

Location：地址重定向 。

Max-Forwards：限制可以转发下个服务器或网关的数量。

Pragma：Pragma头域用来包含实现特定的指令，最常用的是Pragma:no-cache。在HTTP/1.1协议中，它的含义和Cache-Control:no-cache相同。

Range：字节范围。

Referer：简单的说就是告诉服务器我是从哪个页面链接过来的。（防盗链）

Server：原始服务器处理请求的所使用的软件的信息。

Transfer-Encoding：当不能确定报文的长度的时候不能用Content-Length来指明报文的长度。Transfer-Encoding：chunked表明采用chunked编码方式来进行报文传输。

User-Agent：标识请求者的一些信息。

三：Http协议状态码

**1×× 消息**

这一类型的状态码代表请求已经被接受，需要继续处理。这类响应是临时响应，只包含状态行和某些可选的响应头信息，并以空行结束。

100 Continue

客户端应当继续发送请求。这个临时的响应是通知客户端它的部分请求已经被服务器接收，且仍未被拒绝。客户端应当继续发送请求的剩余部分，如果客户端请求已经发送完成，忽略这个响应。服务器必须在请求完成后向客户端发送一个最终响应。

101 Switching Protocols

服务器已经理解了客户端的请求，并将通过Upgrade消息头通知客户端采用不同的协议来完成这个请求。

102 Processing

由WebDAV（RFC2518）扩展的状态码，代表处理奖杯继续进行。

**2×× 成功**

这一类型的状态码，代表请求已经成功被服务器接收，理解，并接受。

200 OK

请求已成功，请求所希望的响应体和数据体将随此响应返回。

201 Created

请求已经被实现，而且一个新的资源已经依据请求的需要而建立，且其URI已经随Location头信息返回。

202 Accepted

服务器已经接受请求，但尚未处理。正如它可能被拒绝一样，最终该请求可能会也可能不会被执行。在异步操作的场合下，没有比发送这个状态码更方便的做法了。

203 Non-Authoritative information

服务器已经成功处理了请求，但返回的实体头部元信息不是在服务器上有效的的确定集合，而是来自本地或者第三方的拷贝。

204 NoContent

服务器成功处理了请求，但不需要返回任何实体内容，并且希望返回更新了的元信息。

205 服务器成功处理了请求，且没有返回任何内容，返回此状态码的响应要求请求者重置文档视图。该响应主要用于接受用户输入后，立即重置表单，以便用户能够轻松地开始另一次输入。

206 Partial Content

服务器已经成功处理了Get请求。类似于FlashGet或者迅雷这类的Http下载工具都是此响应实现断点续传或者将一个大文档分解成多个部分下载。

207 Mutil-Status

由WebDAV（RFC2518）扩张的状态码，代表之后的信息体是一个XML信息，并且可能依照之前子请求的数量的不同，包含一系列的独立的响应代码/。

**3×× 重定向**

这类状态码代表客户端需要采取进一步的操作才能完成请求。通常这类代码用来重定向，后续的请求地址会在本次响应的Location域中指明。

300 Multiple Choices

被请求的资源有一系列的可供选择的回馈信息，每个都有自己特定的地址和浏览器驱动的商议信息。用户或浏览器能够自行选择一个首选的地址进行重定向。

301 Moved Permanently

被请求的资源已经被永久的移动到新的地址。

302 Found

请求的资源临时从不同的URI响应请求。

303 See Other

对应当请请求的响应可以在另一个URI上被找到，而且客户端应当采用Get请求方式访问那个资源。

304 Not Modified

如果客户端发送了一个带条件的Get请求，且该请求已经被允许，而文档的内容没有改变，则服务器返回这个状态码。

305 Use Proxy

被请求的资源必须通过指定的代理才能被访问。Location信息域中给出指定的代理所在的URI信息。

307 Temporary Redirect

请求的资源现在临时从不同的URI响应请求。

**4×× 请求错误**

这类的状态码代表了客户端看起来可能发生了错误，妨碍了服务器的处理。

400 Bad Request

1. 语义有误，当前请求无法被服务器理解
2. 请求参数有误

401 Unauthorized

当前请求需要用户验证。

403 Forbidden

服务器已经接受请求，但是拒绝执行它。

404 NotFound

请求失败，请求所希望得到的资源未被在服务器上发现。

405 Method Not Allowed

请求中指定的请求方法不能被用于请求相应的资源。该响应必须返回一个Allow 头信息用以表示出当前资源能够接受的请求方法的列表。

406 Not Acceptable

请求的资源的特性无法满足请求头中的条件，因而无法生成响应实体。

407 Proxy Authentication Required

与401类似，只不过客户端需要在代理服务器上进行身份验证。

408 Request Timeout

请求超时。

409 Confict

由于和被请求的资源的当前状态之间存在冲突，请求无法完成。

410 Gone

被请求的资源在服务器上已经不再可用了，而且没有任何已知的转发地址。

411 Length Required

服务器拒绝在没有定义Content-Length头的情况下接受请求。

412 Precondition Failed

服务器在验证在请求的头字段中给出先决条件时，没有能满足其中的一个或多个。

413 Request Entity Too Large

服务器拒绝处理当前请求，因为当前请求提交的实体数据大小超过了服务器愿意或者能够处理的范围。

414 Request-URI Too Long

请求的URL长度超过了服务器能够解释的长度，服务器拒绝对该请求的处理。

415 Unsupported Media Type

对于当前请求的放个和所请求的资源，请求中提交的实体不是服务器中所支持的格式。

416 Requested Range Not

如果请求中包含了Range请求头，并且Range 中指定的任何数据范围都与当前资源的可用范围不重合，同时请求中没有定义if-Range请求头。

417 Exceptation Failed

在请求头Except中指定的预期内容无法被服务器满足，或者这个服务器是一个代理服务器，它有明显的证据证明在当前路由的下一个节点上，Except的内容无法被满足。

421 There are too mang connections from your internet address

从当前客户端所在的Ip地址到服务器的链接数超过了服务器许可的最大范围。

422 Unprocessable Entity

请求格式正确，但是由于含有语义错误，无法响应。

424 Failed Dependency

由于之前的某个请求发生的错误，导致当前请求失败。

**5×× 服务器错误**

500 Intenet Server Error

服务器遇到了一个未曾预料的状况，导致了它无法完成对请求的处理。一般来说是服务器代码出错。

501 Not Implemented

服务器不支持当前请求所需要的某个功能。

502 Bad Getway

作为网关或者代理工作的服务器执行请求时，从上游服务器接收到无效的响应。

503 Service Unavaiable

由于临时的服务器维护或者过载，服务器当前无法处理请求。

504 Gatway Timeout

作为网关或者代理服务器处理请求时，没有从上游服务器或者辅助服务器收到响应。

505 HTTP Version Not Support

服务器不支持，或者拒绝支持在请求中使用的Http版本。

506 Variant Also Negotiates

代表服务器存在内部配置错误

507 Insufficient Storage

服务器无法存储完成请求所必须的内容。

509 Bandwidth Limit Exceeded

服务器达到带宽限制。

四．Chunked

概述：有时候Web服务器生成的Http Response是无法在header头中确定消息的大小的，这时一般来说服务器将不会提供 content-length的头消息，而采用Chunked编码动态的提供body内容的长度。

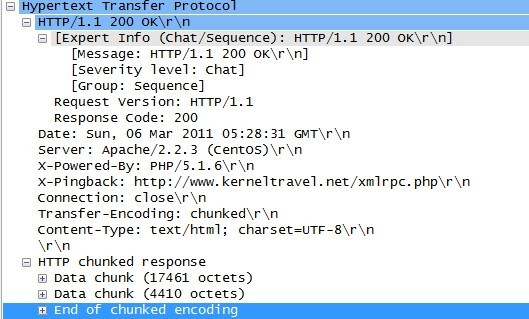
进行Chunked编码传输的Http Response 会在消息头部设置：

Transfer-Encoding:chunked

表示Content Body将用Chunked编码传输内容；

详细：Chunked编码使用若干个Chunk串连而成，由一个标明长度为0的Chunk标示结束。每个Chunk分为头部、正文两部分，头部内容指定下一段正文的字符总数（十六进制的数字）和数量单位（一般不写），正文部分就是指定长度的实际内容，两部分之间用回车换行（CRLF）隔开。在最后一个长度为0的Chunk中的内容是成为footer的内容，是一些附加的Header信息（通常可以直接忽略）；

分块技术：分块技术的意思就是说，实体被分成许多的块，也就是应用层的数据，TCP在传送的过程中，不对它们做任何的结束，而是把应用层产生的数据全部理解成二进制流，然后按照MSS的长度切成一份一份的，一股脑塞到TCP协议栈里面去，而具体这些二进制的数据如何做解释，需要应用层来完成，所以在这之前，一块整体应用层的数据需要等它分成的所有TCP Segment到达对方，重新组装后，应用程序才使用自己的解码方法还原它们。



Http1.1采用的持久的连接，也就是一次TCP的连接不马上释放，允许许多的请求跟响应在一个TCP的连接上发送，所以客户机与服务器需要某种方式来标示一个报文在哪里结束和下一个报文在哪里开始。简单的方法就是使用content-length，但这只有当报文长度可以预先判断的情况下才起作用，而对于动态的内容或者在发送数据前不能判断长度的情况下，可以使用分块的方法来传送编码。