20190126http协议学习

一.HTTP 简介

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 Web ）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通信协议来传递数据（HTML 文件, 图片文件, 查询结果等）。

二.HTTP 工作原理

- HTTP协议工作于客户端-服务端架构上。浏览器作为HTTP客户端通过URL向HTTP服务端即WEB服务器发送所有请求。
- Web服务器有：Apache服务器，IIS服务器（Internet Information Services）等。
- Web服务器根据接收到的请求后，向客户端发送响应信息。
- HTTP默认端口号为80，但是你也可以改为8080或者其他端口。

> HTTP三点注意事项:
>
> 1. HTTP是**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，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。
> 2. HTTP是**媒体独立的**：这意味着，只要客户端和服务器知道如何处理的数据内容，**任何类型的数据都可以通过HTTP发送**。客户端以及服务器指定使用适合的MIME-type内容类型。
> 3. HTTP是**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。

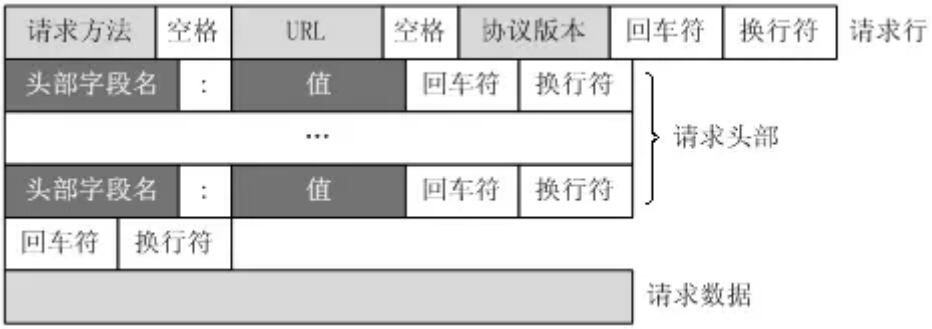
三.HTTP 消息结构

- HTTP是基于客户端/服务端（C/S）的架构模型，通过一个可靠的链接来交换信息，是一个无状态的请求/响应协议。

- **一个HTTP"客户端"是一个应用程序**（Web浏览器或其他任何客户端），通过连接到服务器达到向服务器发送一个或多个HTTP的请求的目的。
- **一个HTTP"服务器"同样也是一个应用程序**（通常是一个Web服务，如Apache Web服务器或IIS服务器等），通过接收客户端的请求并向客户端发送HTTP响应数据。
- HTTP使用**统一资源标识符（Uniform Resource Identifiers, URI）来传输数据和建立连接。**

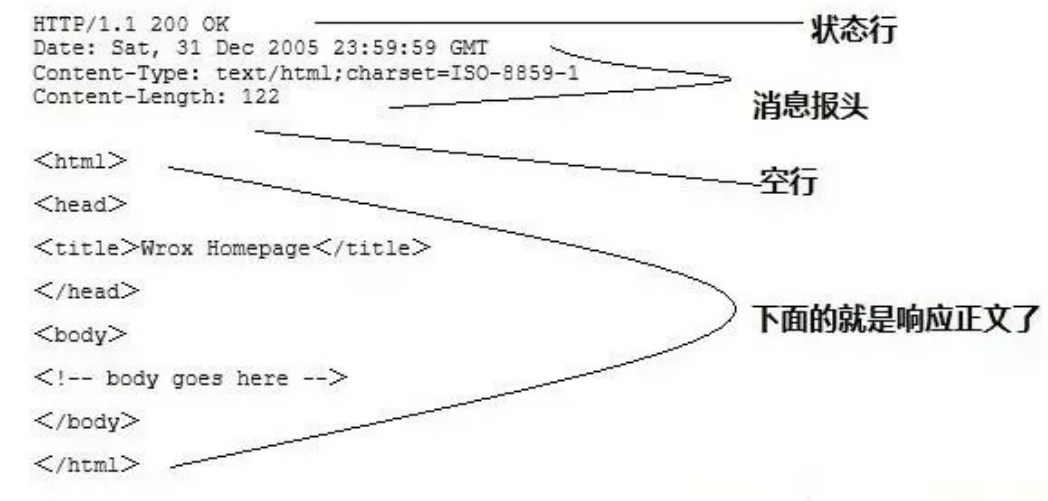
客户端请求消息

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）、**请求头部**（header）、**空行**和**请求数据**四个部分组成，下图给出了请求报文的一般格式。



服务器响应消息

HTTP响应也由四个部分组成，分别是：**状态行**、**消息报头**、**空行**和**响应正文**。



实例

客户端请求

```
GET /hello.txt HTTP/1.1
User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3
Host: www.example.com
Accept-Language: en, mi
```

服务端响应

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
```

> Server: Apache
> Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
> ETag: “34aa387-d-1568eb00”
> Accept-Ranges: bytes
> Content-Length: 51
> Vary: Accept-Encoding
> Content-Type: text/plain

输出结果

> Hello World! My payload includes a trailing CRLF.

四.HTTP请求方法

HTTP 协议中共定义了八种方法或者叫“动作”来表明**对 Request-URI 指定的资源的不同操作方式**，具体介绍如下：

序号	方法	描述
1	GET	**请求制定的页面信息，并返回实体主体**
2	HEAD	类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头
3	POST	**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。
4	PUT	从客户端向服务器传送的数据取代指定的文档的内容。
5	DELETE	请求服务器删除指定的页面。
6	CONNECT	HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。
7	OPTIONS	允许客户端查看服务器的性能。
8	TRACE	回显服务器收到的请求，主要用于测试或诊断。

虽然 HTTP 的请求方式有 8 种，但是我们**在实际应用中常用的也就是 get 和 post，其他请求方式也都可以通过这两种方式间接的来实现。**

五.HTTP响应头信息

应答头	说明
Allow	服务器支持哪些请求方法（如GET、POST等）。

Content-Encoding	**文档的编码（Encode）方法。**只有在解码之后才可以得到Content-Type头指定的内容类型。利用gzip压缩文档能够显著地减少HTML文档的下载时间。Java的GZIPOutputStream可以很方便地进行gzip压缩，但只有Unix上的Netscape和Windows上的IE 4、IE 5才支持它。因此，Servlet应该通过查看Accept-Encoding头（即request.getHeader(“Accept-Encoding”)）检查浏览器是否支持gzip，为支持gzip的浏览器返回经gzip压缩的HTML页面，为其他浏览器返回普通页面。
Content-Length	表示**内容长度。**只有当浏览器**使用持久HTTP连接时才需要这个数据。**如果你想要利用持久连接的优势，可以把输出文档写入 ByteArrayOutputStream，完成后查看其大小，然后把该值放入Content-Length头，最后通过byteArrayStream.writeTo(response.getOutputStream()发送内容。
Content-Type	表示后面的文档属于什么MIME类型。Servlet默认为text/plain，但通常需要显式地指定为text/html。由于经常要设置Content-Type，因此HttpServletResponse提供了一个专用的方法setContentType。
Date	**当前的GMT时间。**你可以用setDateHeader来设置这个头以避免转换时间格式的麻烦。
Expires	**应该在什么时候认为文档已经过期，从而不再缓存它？**
Last-Modified	**文档的最后改动时间。**客户可以通过If-Modified-Since请求头提供一个日期，该请求将被视为一个条件GET，只有改动时间迟于指定时间的文档才会返回，否则返回一个304（Not Modified）状态。Last-Modified也可用setDateHeader方法来设置。
Location	**表示客户应当到哪里去提取文档。**Location通常不是直接设置的，而是通过HttpServletResponse的sendRedirect方法，该方法同时设置状态代码为302。
Refresh	**表示浏览器应该在多少时间之后刷新文档**，以秒计。除了刷新当前文档之外，你还可以通过setHeader(“Refresh”, “5; URL=http://host/path“)让浏览器读取指定的页面。 注意这种功能通常是通过设置HTML页面HEAD区的 < META HTTP-EQUIV=”Refresh” CONTENT=”5;URL=http://host/path“ > 实现，这是因为，自动刷新或重定向对于那些不能使用CGI或Servlet的HTML编写者十分重要。但是，对于Servlet来说，直接设置Refresh头更加方便。 注意Refresh的意义是”N秒之后刷新本页面或访问指定页面”，而不是”每隔N秒刷新本页面或访问指定页面”。因此，连续刷新要求每次都发送一个Refresh头，而发送204状态代码则可以阻止浏览器继续刷新，不管是使用Refresh头还是 < META HTTP-EQUIV=”Refresh” … > 。 注意Refresh头不属于HTTP 1.1正式规范的一部分，而是一个扩展，但Netscape和IE都支持它。
Server	**服务器名字。**Servlet一般不设置这个值，而是由Web服务器自己设置。
Set-Cookie	设置和页面关联的Cookie。Servlet不应使用response.setHeader(“Set-Cookie”, …)，而是应使用HttpServletResponse提供的专用方法addCookie。参见下文有关Cookie设置的讨论。

WWW-Authenticate	客户应该在Authorization头中提供什么类型的授权信息？在包含401（Unauthorized）状态行的应答中这个头是必需的。例如，response.setHeader(“WWW-Authenticate”, “BASIC realm= \ ”executives \ ”“)。注意Servlet一般不进行这方面的处理，而是让Web服务器的专门机制来控制受密码保护页面的访问（例如.htaccess）。

六.HTTP状态码

当浏览者访问一个网页时，浏览者的浏览器会向网页所在服务器发出请求。当浏览器接收并显示网页前，此网页所在的服务器会返回一个包含HTTP状态码的信息头（server header）用以响应浏览器的请求。

HTTP状态码的英文为HTTP Status Code。

下面是常见的HTTP状态码：

- 200 - 请求成功
- 301 - 资源（网页等）被永久转移到其它URL
- 404 - 请求的资源（网页等）不存在
- 500 - 内部服务器错误

HTTP状态码分类
HTTP状态码由**三个十进制数字组成**，第一个十进制数字定义了状态码的类型，后两个数字没有分类的作用。HTTP状态码共分为5种类型：

分类	**分类描述**
1**	信息，服务器收到请求，需要请求者继续执行操作
2**	成功，操作被成功接收并处理
3**	重定向，需要进一步的操作以完成请求
4**	客户端错误，请求包含语法错误或无法完成请求
5**	服务器错误，服务器在处理请求的过程中发生了错误

七.HTTP之URL

HTTP使用统一资源标识符（Uniform Resource Identifiers, URI）来传输数据和建立连接。URL是一种特殊类型的URI，包含了用于查找某个资源的足够的信息

URL,全称是UniformResourceLocator, 中文叫**统一资源定位符**,是互联网上用来标识某一处资源的地址。以下面这个URL为例，介绍下普通URL的各部分组成：

http://www.aspxfans.com:8080/news/index.asp?boardID=5&ID=24618&page=1#name
从上面的URL可以看出，一个完整的URL包括以下几部分：

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Internet中可以使用多种协议，如HTTP，FTP等等本例中使用的是HTTP协议。在”HTTP”后面的“//”为分隔符
2. **域名部分**：该URL的域名部分为**“www.aspxfans.com”**。一个URL中，也可以使用IP地址作为域名使用
3. **端口部分**：跟在域名后面的是端口，域名和端口之间使用**“:”**作为分隔符。**端口不是一个URL必须的部分，如果省略端口部分，将采用默认端口**
4. **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。**虚拟目录也不是一个URL必须的部分。**本例中的虚拟目录是**“/news/”**
5. **文件名部分**：从域名后的最后一个“/”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，都是文件名部分。本例中的文件名是**“index.asp”**。**文件名部分也不是一个URL必须的部分，如果省略该部分，则使用默认的文件名**
6. **锚部分**：从“#”开始到最后，都是锚部分。本例中的锚部分是**“name”**。**锚部分也不是一个URL必须的部分**
7. **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。本例中的参数部分为**“boardID=5&ID=24618&page=1”**。参数可以允许有多个参数，参数与参数之间用“&”作为分隔符。