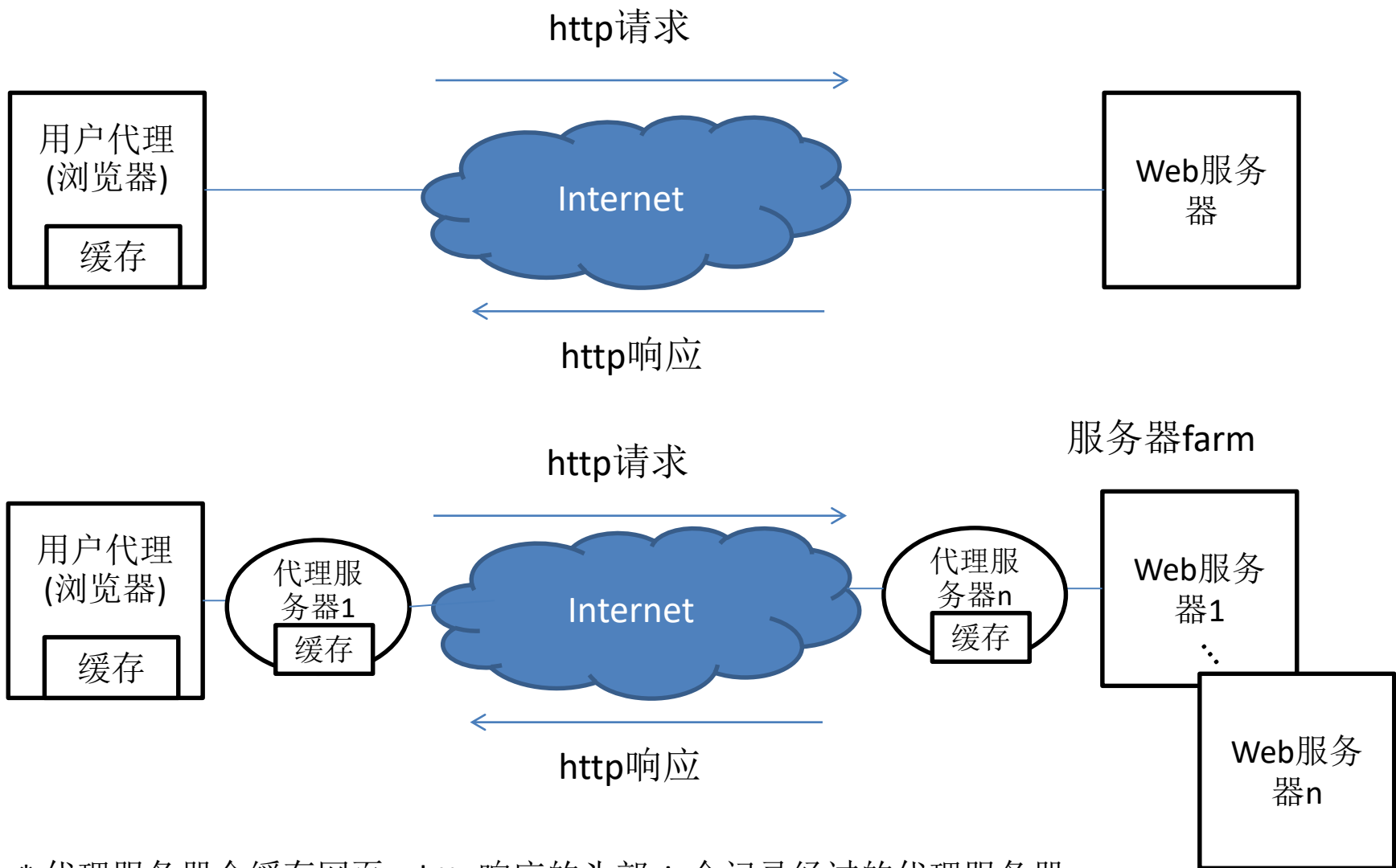


HTTP协议

2018.1.9

概述



* 代理服务器会缓存网页。http响应的头部via会记录经过的代理服务器。

HTTP请求--get

● 程序welcome.jsp



The screenshot shows a web browser window with the title "Welcome". The address bar displays the URL "http://202.116.76.22:8080/jsp/welcome.jsp". The browser's navigation bar includes back, forward, and refresh buttons, along with the current URL and icons for bookmarks, settings, and extensions. The main content area displays a small image of a landscape. A red line points from the text "文件字节数: 1469" to the image. Another blue line points from the text "文件字节数: 204" to the code snippet below. The code snippet is a JSP file named "welcome.jsp" that sets a session attribute "aaa" to 1 and includes an image "img01.jpg".

文件字节数: 1469

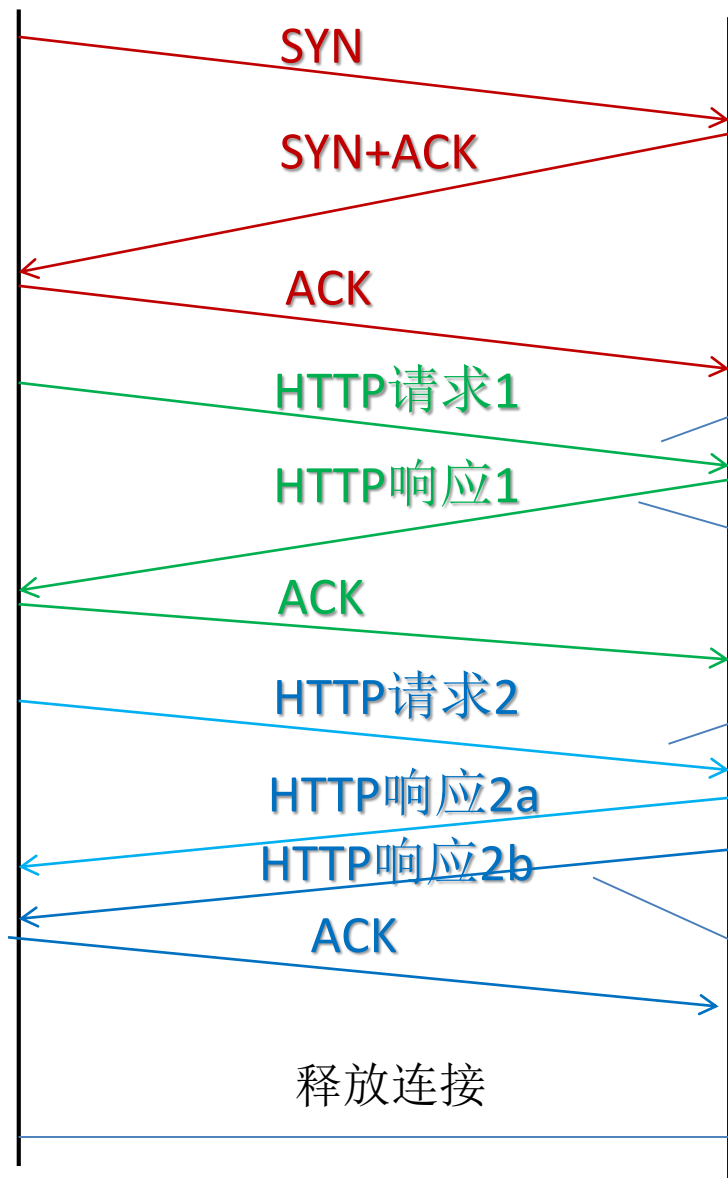
文件字节数: 204

```
<%@ page contentType="text/html; charset=gb2312"%>
<html>
<head>
  <title>Welcome</title>
</head>
<body>
  <% session.setAttribute("aaa",1); %>
  
</body>
</html>
```

● 读取网页: <http://202.116.76.22:8080/jsp/welcome.jsp>

浏览器

Web服务器



打开浏览器第一次运行

* 故意修改了图片

(1)浏览器产生http请求(GET), URL:
<http://202.116.76.22:8080/jsp/welcome.jsp>

(2) Web服务器执行welcome.jsp产生的http
响应, 包含html文件, 浏览器收到后显
示该文件, 并在分析该文件后产生了
下一个请求(取一个图像文件)

(3)浏览器收到响应并分析后产生http请求
(GET), URL:
<http://202.116.76.22:8080/jsp/file/img01.jpg>

(4) Web服务器直接形成http响应(包含
img01.jpg)

浏览器收到响应后将显示该图像

响应2a TCP data
len:1460B

响应2b TCP data
len: 238B

释放连接

202.116.76.22-Server 192.168.1.64-Client

	Source	Destination	Protocol	Info
1	192.168.1.64	202.116.76.22	TCP	24892→8080 [SYN] Seq=0 Win=65535 Len=0 M
2	202.116.76.22	192.168.1.64	TCP	8080→24892 [SYN, ACK] Seq=0 Ack=1 Win=81
3	192.168.1.64	202.116.76.22	TCP	24892→8080 [ACK] Seq=1 Ack=1 Win=262144
4	192.168.1.64	202.116.76.22	HTTP	GET /jsp/welcome.jsp HTTP/1.1 (A)
5	202.116.76.22	192.168.1.64	HTTP	HTTP/1.1 200 OK (text/html) (B)
6	192.168.1.64	202.116.76.22	TCP	24892→8080 [ACK] Seq=602 Ack=264 Win=261
7	192.168.1.64	202.116.76.22	HTTP	GET /jsp/file/img01.jpg HTTP/1.1 (C)
8	202.116.76.22	192.168.1.64	TCP	[TCP segment of a reassembled PDU]
9	202.116.76.22	192.168.1.64	HTTP	HTTP/1.1 200 OK (JPEG JFIF image) (D)
10	192.168.1.64	202.116.76.22	TCP	24892→8080 [ACK] Seq=1360 Ack=1935 Win=2
11	202.116.76.22	192.168.1.64	TCP	8080→24892 [FIN, ACK] Seq=1935 Ack=1360
12	192.168.1.64	202.116.76.22	TCP	24892→8080 [ACK] Seq=1360 Ack=1936 Win=2
13	192.168.1.64	202.116.76.22	TCP	24892→8080 [FIN, ACK] Seq=1360 Ack=1936
14	192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24892→8080 [FIN, AC
15	192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24892→8080 [FIN, AC
16	192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24892→8080 [FIN, AC
17	192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24892→8080 [FIN, AC
18	192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24892→8080 [FIN, AC
19	192.168.1.64	202.116.76.22	TCP	24892→8080 [RST, ACK] Seq=1361 Ack=1936

服务器在很短的一段时间内没有再收到http请求，则会发送FIN包并关闭连接。

Source	Destination	Protocol	Info
192.168.1.64	202.116.76.22	TCP	24892→8080 [SYN] Seq=0 Win=65535 Len=0 M
202.116.76.22	192.168.1.64	TCP	8080→24892 [SYN, ACK] Seq=0 Ack=1 Win=81
192.168.1.64	202.116.76.22	TCP	24892→8080 [ACK] Seq=1 Ack=1 Win=262144
192.168.1.64	202.116.76.22	HTTP	GET /jsp/welcome.jsp HTTP/1.1
202.116.76.22	192.168.1.64	HTTP	HTTP/1.1 200 OK (text/html)
192.168.1.64	202.116.76.22	TCP	24892→8080 [ACK] Seq=602 Ack=264 Win=261
192.168.1.64	202.116.76.22	HTTP	GET /jsp/file/img01.jpg HTTP/1.1
202.116.76.22	192.168.1.64	TCP	[TCP segment of a reassembled PDU]
202.116.76.22	192.168.1.64	HTTP	HTTP/1.1 200 OK (JPEG JFIF image)
192.168.1.64	202.116.76.22	TCP	24892→8080 [ACK] Seq=1360 Ack=1935 Win=2
202.116.76.22	192.168.1.64	TCP	8080→24892 [FIN, ACK] Seq=1935 Ack=1360
192.168.1.64	202.116.76.22	TCP	24892→8080 [ACK] Seq=1360 Ack=1936 Win=2
192.168.1.64	202.116.76.22	TCP	24892→8080 [FIN, ACK] Seq=1360 Ack=1936
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24892→8080 [FIN, AC
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24892→8080 [FIN, AC
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24892→8080 [FIN, AC
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24892→8080 [FIN, AC
192.168.1.64	202.116.76.22	TCP	24892→8080 [RST, ACK] Seq=1361 Ack=1936

(A)浏览器产生**http请求 (GET)** , URL: http://202.116.76.22:8080/jsp/welcome.jsp

GET /jsp/welcome.jsp HTTP/1.1

方法 文件路径 支持的**http**最高版本

Accept: text/html, application/xhtml+xml, image/jxr, */*

可接受的媒体类型

Accept-Language: zh-CN

可接受的语言

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) ... Edge/13.10586

Accept-Encoding: gzip, deflate

可接受的编码

Host: 202.116.76.22:8080

URL中的主机名 (域名或IP地址)

Connection: Keep-Alive

发完**http**响应后是否还保持连接

Cookie: username=admin; password=3214

(空行)

(B)Web服务器执行welcome.jsp, 产生**http响应** (包含html文件)

HTTP/1.1 200 OK

支持的**http**最高版本 响应码 描述

Server: Apache-Coyote/1.1

Web服务器类型和版本

Set-Cookie: JSESSIONID=9FB94A041ECF73217D8A3D147BE1CBA3; Path=/jsp/; HttpOnly

Content-Type: text/html; charset=gb2312

内容的类型

Content-Length: 119

内容的长度

Date: Sun, 24 Apr 2016 12:21:54 GMT

产生本响应的时间

(空行)

<html>

<head>...<body></body>

</html>

} 内容(content)

* **User-Agent**(浏览器): Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2486.0 Safari/537.36 Edge/13.10586

* 媒体类型采用MIME表示(Multipurpose Internet Mail Extensions), image/jxr - JPEG类型

* gzip, deflate: 两种压缩方式

* Host可以用于区分虚拟主机

* 头部含义还可以参见附录

(C)浏览器收到后分析html文件，产生一个http请求（GET）去获取img01.jpg，URL：
http://202.116.76.22:8080/jsp/file/img01.jpg

q为偏好系数（默认为1）

GET /jsp/file/img01.jpg HTTP/1.1

Accept: image/png, image/svg+xml, image/jxr, image/*;q=0.8, */*;q=0.5

Referer: http://202.116.76.22:8080/jsp/welcome.jsp 从哪个网页发出本http请求

Accept-Language: zh-CN

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) ...Edge/13.10586

Accept-Encoding: gzip, deflate

Host: 202.116.76.22:8080

If-Modified-Since: Sun, 24 Apr 2016 11:14:50 GMT

已缓存文件的修改时间
该文件的实体标签(ETags)

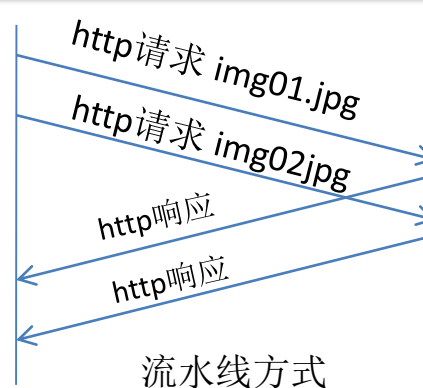
If-None-Match: W/"5295-1461496490242"

Connection: Keep-Alive

Cookie: JSESSIONID=9FB94A041ECF73217D8A3D147BE1CBA3; username=admin;
password=3214

(空行)

- * 缓存文件的修改时间ji就是在Web服务器上保存的该文件的修改时间。
- * ETag(Entity Tag)由服务器生成，比文件的修改时间粒度更小，适用于文件修改太频繁以至于修改时间不能反映出来的情况。
- * 如果html文件中包含多个Web对象，例如，还有图像img02.jpg的img元素，浏览器会连续发出多个http请求，即采用流水线方式进行请求和响应。
- * 如果Connection取值close，Web服务器发出http响应之后会立即关闭连接。



(D)Web服务器收到（3）的请求后，产生一个http响应(包含文件img01.jpg)。浏览器收到该响应后将显示该图像。* 该图像在上次下载后被修改过

HTTP/1.1 200 OK

Server: Apache-Coyote/1.1

Accept-Ranges: bytes

ETag: W/"1469-1461500473349"

Last-Modified: Sun, 24 Apr 2016 12:21:13 GMT

Content-Type: image/jpeg

Content-Length: 1469

内容(img01.jpg)的长度

Date: Sun, 24 Apr 2016 12:21:54 GMT

(空行)

.....JFIF.....H.H....."Exif..MM.....

.....

(省略)

....Ky...@Y|.....r\$...6..\..(...h...:...*[s..

不可显示字符

img01.jpg

0000	ff d8 ff e0 00 10 4a 46 49 46 00 01 01 01 00 48JFIF.....H
0010	00 48 00 00 ff e1 00 22 45 78 69 66 00 00 4d 4d	.H....."Exif..MM
0020	00 2a 00 00 00 08 00 01 01 12 00 03 00 00 00 01	.*.....
0030	00 01 00 00 00 00 00 00 ff db 00 43 00 02 01 01C....
0040	02 01 01 02 02 02 02 02 02 02 03 05 03 03 03
0050	03 03 06 04 04 03 05 07 06 07 07 07 06 07 07 08
.....		
0560	df 3b a0 c9 dc 36 b1 2d 45 15 f7 18 7e 2e cc ab	.;...6.-E...~...
0570	41 d3 af 25 24 ef d2 dd 52 fb 36 3e 66 b7 0b e5	A..%\$...R.6>f...
0580	f8 79 2a 98 78 72 b5 e7 7f fd 2a e6 c5 fe 8d aa	.y*.xr....*.....
0590	d8 88 da 1b 4b 79 92 d9 d5 40 59 7c cd 98 8c e0Ky...@Y
05a0	87 ea 72 24 c7 cd f3 36 c2 f8 5c e1 ca 28 a2 a4	..r\$...6..\..(...
05b0	a3 68 b7 15 aa 3a e9 e2 2a 5b 73 ff d9	.h...:...*[s..

img01.jpg

●刷新

按下浏览器刷新按钮：<http://202.116.76.22:8080/jsp/welcome.jsp>

Source	Destination	Protocol	Info
192.168.1.64	202.116.76.22	TCP	24831→8080 [SYN] Seq=0 win=65535 Len=0 MSS
202.116.76.22	192.168.1.64	TCP	8080→24831 [SYN, ACK] Seq=0 Ack=1 win=8192
192.168.1.64	202.116.76.22	TCP	24831→8080 [ACK] Seq=1 Ack=1 win=262144 Le
192.168.1.64	202.116.76.22	HTTP	GET /jsp/welcome.jsp HTTP/1.1
202.116.76.22	192.168.1.64	HTTP	HTTP/1.1 200 OK (text/html)
192.168.1.64	202.116.76.22	TCP	24831→8080 [ACK] Seq=602 Ack=264 win=26163
192.168.1.64	202.116.76.22	HTTP	GET /jsp/file/img01.jpg HTTP/1.1
202.116.76.22	192.168.1.64	HTTP	<u>HTTP/1.1 304 Not Modified</u>
192.168.1.64	202.116.76.22	TCP	24831→8080 [ACK] Seq=1360 Ack=387 win=2616
202.116.76.22	192.168.1.64	TCP	8080→24831 [FIN, ACK] Seq=387 Ack=1360 win
192.168.1.64	202.116.76.22	TCP	24831→8080 [ACK] Seq=1360 Ack=388 win=2616
192.168.1.64	202.116.76.22	TCP	24831→8080 [FIN, ACK] Seq=1360 Ack=388 win
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24831→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24831→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24831→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24831→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24831→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	24831→8080 [RST, ACK] Seq=1361 Ack=388 win

```
GET /jsp/file/img01.jpg HTTP/1.1
Accept: image/png, image/svg+xml, image/jxr, image/*;q=0.8, */*;q=0.5
Referer: http://202.116.76.22:8080/jsp/welcome.jsp
Accept-Language: zh-CN
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) ... Edge/13.10586
Accept-Encoding: gzip, deflate
Host: 202.116.76.22:8080
If-Modified-Since: Sun, 24 Apr 2016 12:21:13 GMT
If-None-Match: W/"1469-1461500473349" — ETags
Connection: Keep-Alive
Cookie: JSESSIONID=9FB94A041ECF73217D8A3D147BE1CBA3; username=admin;
        password=3214
(空行)
```

```
HTTP/1.1 304 Not Modified
Server: Apache-Coyote/1.1
ETag: W/"1469-1461500473349"
Date: Sun, 24 Apr 2016 15:33:19 GMT
(空行)
```

- * If-Modified-Since记录了缓存的图像img01.jpg的修改时间(上次取回该图像传回来的)。
- * If-None-Match记录了缓存的图像img01.jpg的Etag(上次取回该图像传回来的)。
- * 如果服务器保存的图像img01.jpg的修改时间和Etag均没有变化，则不取回图像，直接使用浏览器缓存的图像。

●按回车键

在浏览器地址栏内按回车: <http://202.116.76.22:8080/jsp/welcome.jsp>

Source	Destination	Protocol	Info
192.168.1.64	202.116.76.22	TCP	24787→8080 [SYN] Seq=0 win=65535 Len=0 MSS=
202.116.76.22	192.168.1.64	TCP	8080→24787 [SYN, ACK] Seq=0 Ack=1 win=8192
192.168.1.64	202.116.76.22	TCP	24787→8080 [ACK] Seq=1 Ack=1 win=262144 Len=0
192.168.1.64	202.116.76.22	HTTP	GET /jsp/welcome.jsp HTTP/1.1
202.116.76.22	192.168.1.64	HTTP	HTTP/1.1 200 OK (text/html)
192.168.1.64	202.116.76.22	TCP	24787→8080 [ACK] Seq=602 Ack=264 win=261632
202.116.76.22	192.168.1.64	TCP	8080→24787 [FIN, ACK] Seq=264 Ack=602 win=0
192.168.1.64	202.116.76.22	TCP	24787→8080 [ACK] Seq=602 Ack=265 win=261632
192.168.1.64	202.116.76.22	TCP	24787→8080 [FIN, ACK] Seq=602 Ack=265 win=0
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24787→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24787→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24787→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24787→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 24787→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	24787→8080 [RST, ACK] Seq=603 Ack=265 win=0

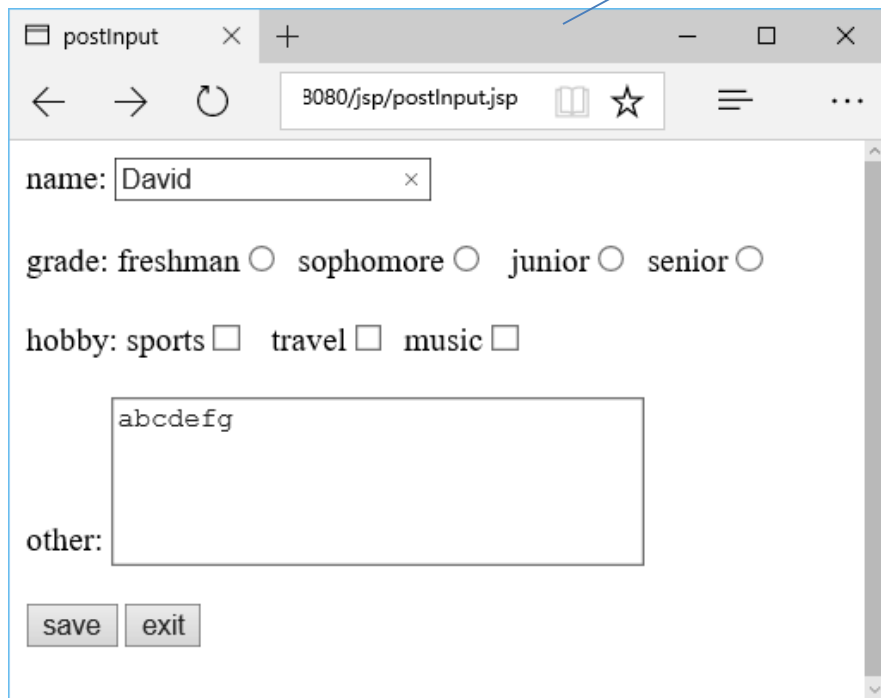
* 这里直接从浏览器缓存中取图像，而没有去Web服务器取图像。

HTTP请求-post

- 操作过程

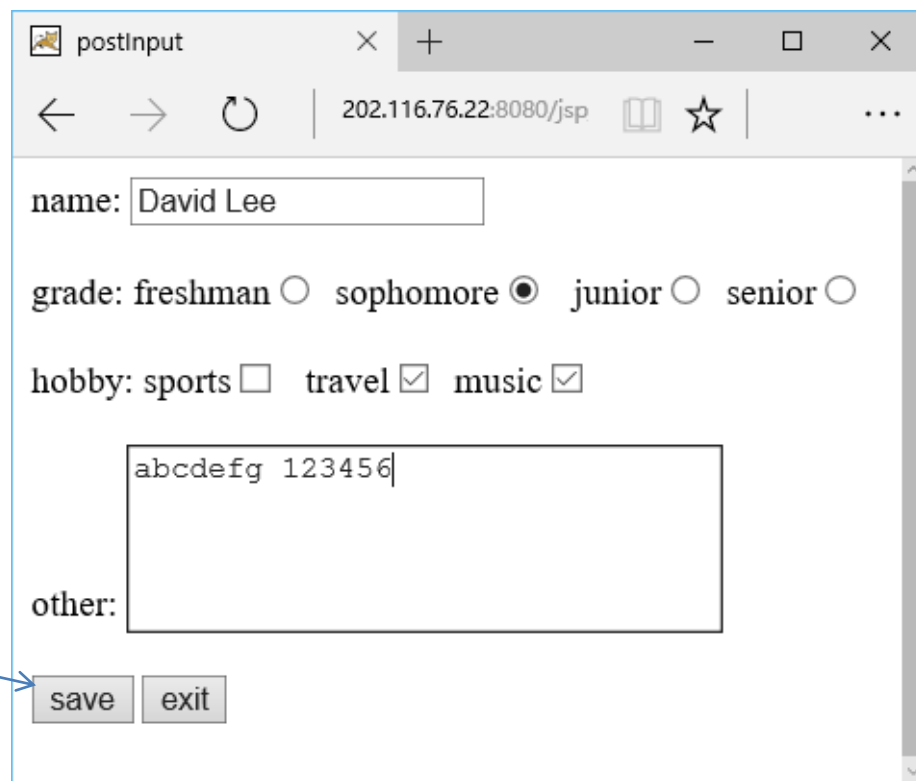
URL:http://202.116.76.22:8080/jsp/postInput.jsp

(1) 显示网页:



A screenshot of a web browser window titled 'postInput'. The address bar shows '3080/jsp/postInput.jsp'. The form contains the following elements: a text input for 'name' with the value 'David'; a group of radio buttons for 'grade' with options 'freshman', 'sophomore', 'junior', and 'senior'; a group of checkboxes for 'hobby' with options 'sports', 'travel', and 'music'; a text area for 'other' containing the text 'abcdefg'; and two buttons at the bottom labeled 'save' and 'exit'.

(2) 输入数据:

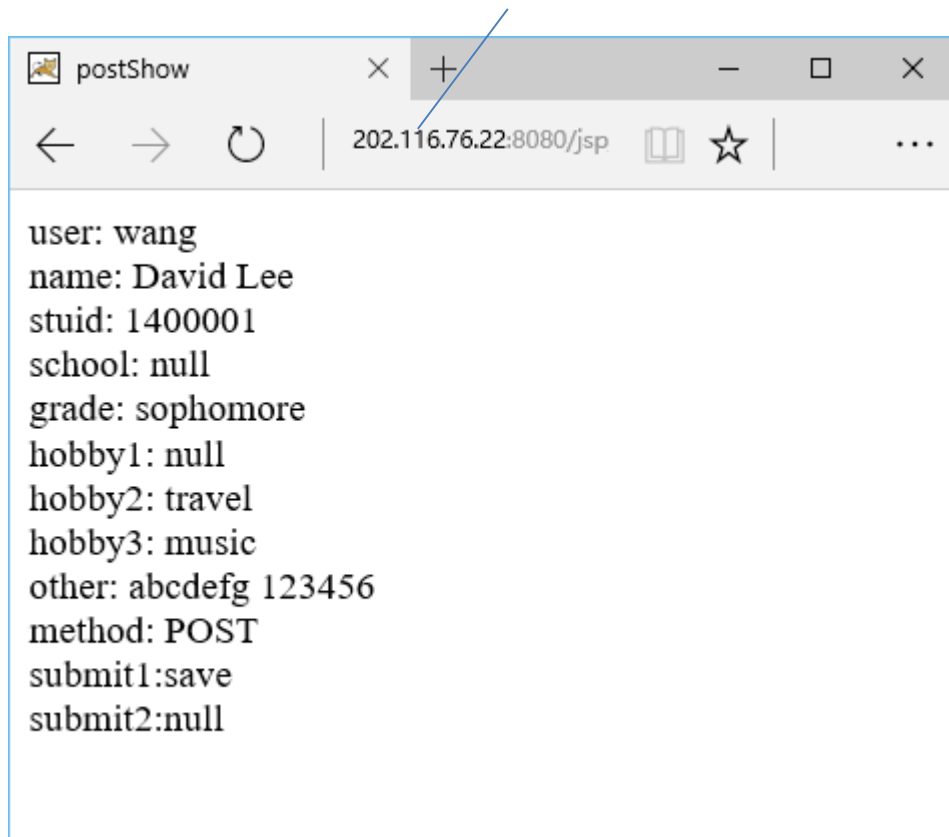


A screenshot of the same web browser window after data entry. The 'name' field now contains 'David Lee'. The 'grade' radio buttons have 'sophomore' selected. The 'hobby' checkboxes have 'travel' and 'music' checked. The 'other' text area now contains 'abcdefg 123456'. The 'save' and 'exit' buttons remain at the bottom.

点击提交

(3) 提交后显示:

URL:http://202.116.76.22:8080/jsp/postShow.jsp



• JSP程序

postInput.jsp

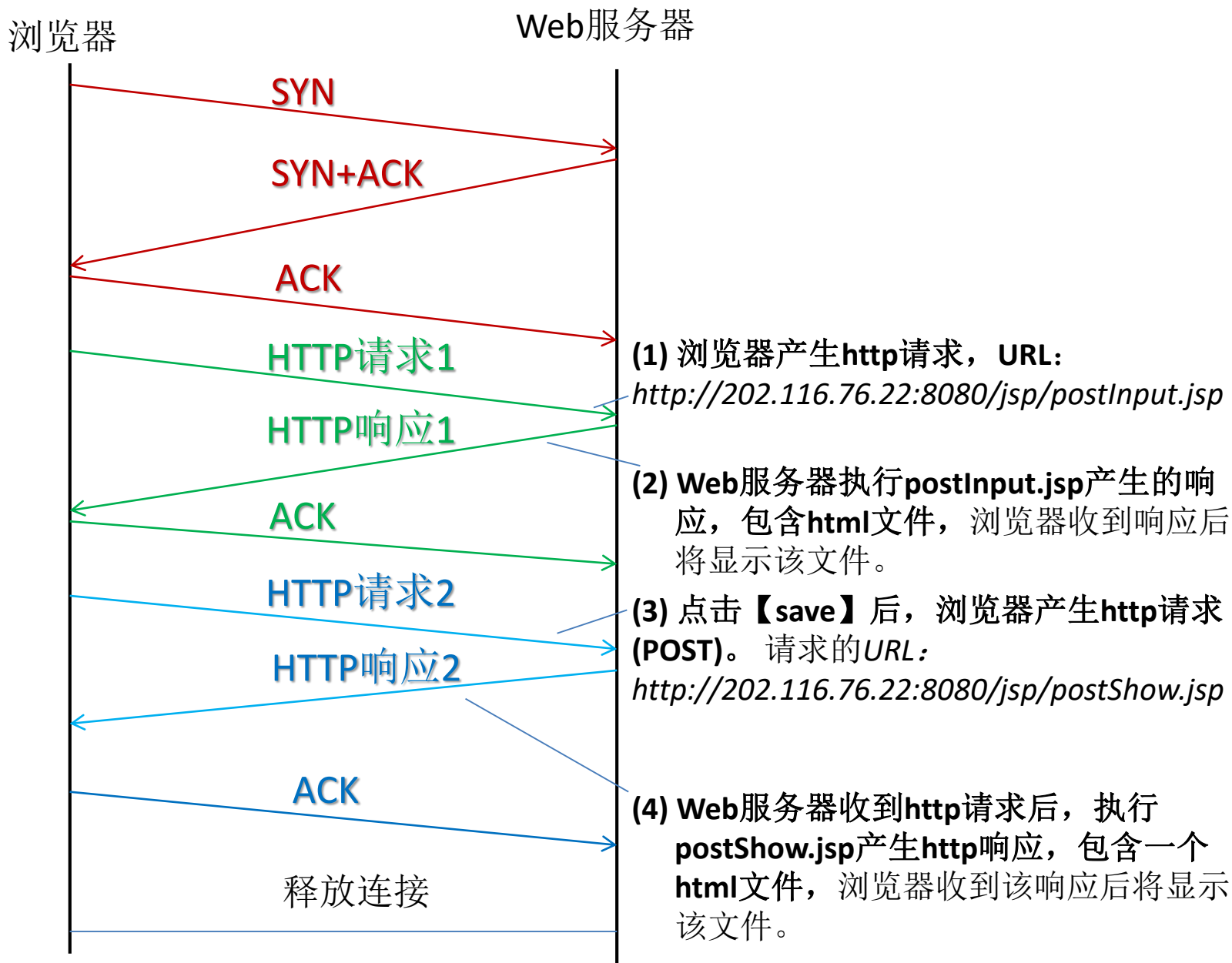
```
<%@ page language="java" import="java.util.*"
    contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>
<!DOCTYPE HTML>
<html>
    <head>
        <title>postInput</title>
    </head>
    <body>
        <form action="postShow.jsp?user=wang" method="post">
            <input type="hidden" name="stuid" value="1400001" />
            name: <input type="text" name="name" value="David"/>
            grade: freshman<input type="radio" name="grade" value="freshman"/>
                  sophomore<input type="radio" name="grade" value="sophomore"/>
                  junior<input type="radio" name="grade" value="junior"/>
                  senior<input type="radio" name="grade" value="senior"/>
            hobby: sports<input type="checkbox" name="hobby1" value="sports"/>
                  travel<input type="checkbox" name="hobby2" value="travel"/>
                  music<input type="checkbox" name="hobby3" value="music"/>
            other: <textarea rows="5" cols="30" name="other">abcdefg</textarea>
            <input type="submit" name="submit1" value="save" />
            <input type="submit" name="submit2" value="exit" />
        </form>
    </body>
</html>
```

form没有action时，会提交给本网页，如果method为get（默认），则会自动把输入键值对作为url的参数（IE会对汉字进行编码而Chrome则不会）。

postShow.jsp

```
<%@ page language="java" import="java.util.*"
        contentType="text/html; charset=utf-8"%>
<%request.setCharacterEncoding("utf-8");%>
<!DOCTYPE HTML>
<html>
    <head>
        <title>postShow</title>
    </head>
    <body>
        user:    <%= request.getParameter("user")    %>
        name:    <%= request.getParameter("name")    %>
        stuid:   <%= request.getParameter("stuid")   %>
        school:  <%= request.getParameter("school")  %>
        grade:   <%= request.getParameter("grade")   %>
        hobby1:  <%= request.getParameter("hobby1")  %>
        hobby2:  <%= request.getParameter("hobby2")  %>
        hobby3:  <%= request.getParameter("hobby3")  %>
        other:   <%= request.getParameter("other")   %>
        method:  <%= request.getMethod()              %>
        submit1: <%= request.getParameter("submit1") %>
        submit2: <%= request.getParameter("submit2") %>
    </body>
</html>
```

• http过程分析



202.116.76.22-Server 192.168.1.64-Client

Source	Destination	Protocol	Info
192.168.1.64	202.116.76.22	TCP	26515→8080 [SYN] Seq=0 win=65535 Len=0 MSS=
202.116.76.22	192.168.1.64	TCP	8080→26515 [SYN, ACK] Seq=0 Ack=1 win=8192
192.168.1.64	202.116.76.22	TCP	26515→8080 [ACK] Seq=1 Ack=1 win=262144 Len=
192.168.1.64	202.116.76.22	HTTP	GET /jsp/postInput.jsp HTTP/1.1 (1)
202.116.76.22	192.168.1.64	TCP	8080→26515 [ACK] Seq=1 Ack=571 win=65536 Len=
202.116.76.22	192.168.1.64	HTTP	HTTP/1.1 200 OK (text/html) (2)
192.168.1.64	202.116.76.22	TCP	26515→8080 [ACK] Seq=571 Ack=1265 win=26080
192.168.1.64	202.116.76.22	HTTP	POST /jsp/postShow.jsp?user=wang HTTP/1.1 (3)
202.116.76.22	192.168.1.64	HTTP	HTTP/1.1 200 OK (text/html) (4)
192.168.1.64	202.116.76.22	TCP	26515→8080 [ACK] Seq=1405 Ack=1832 win=2621
202.116.76.22	192.168.1.64	TCP	8080→26515 [FIN, ACK] Seq=1832 Ack=1405 win=
192.168.1.64	202.116.76.22	TCP	26515→8080 [ACK] Seq=1405 Ack=1833 win=2621
192.168.1.64	202.116.76.22	TCP	26515→8080 [FIN, ACK] Seq=1405 Ack=1833 win=
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 26515→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 26515→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 26515→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 26515→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 26515→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	26515→8080 [RST, ACK] Seq=1406 Ack=1833 win=

(1)客户端发出的HTTP请求，请求的URL: *http://202.116.76.22:8080/jsp/postInput.jsp*

GET /jsp/postInput.jsp HTTP/1.1

Accept: text/html, application/xhtml+xml, image/jxr, */*

Accept-Language: zh-CN

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) ...Edge/13.10586

Accept-Encoding: gzip, deflate

Host: 202.116.76.22:8080

Connection: Keep-Alive

Cookie: JSESSIONID=D6F57FCC8AFAE8EE26D1533B41321156; username=*admin*;...
(空行)

(2) 服务器发出http响应，包含执行postInput.jsp产生的html文件， 浏览器收到后将显示出来

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html; charset=utf-8
Content-Length: 1120
Date: Sun, 24 Apr 2016 18:25:40 GMT
(空行)
<!DOCTYPE HTML>
<html>
  <head>
    <title>postInput</title>
  </head>
  <body>
    <form action="postShow.jsp?user=wang" method="post">
      <input type="hidden" name="stuid" value="1400001" />
      name: <input type="text" name="name" value="David"/><br><br>
      grade: freshman<input type="radio" name="grade" value="freshman"/>&nbsp;
             sophomore<input type="radio" name="grade" value="sophomore"/> &nbsp;
             junior<input type="radio" name="grade" value="junior"/>&nbsp;
             senior<input type="radio" name="grade" value="senior"/><br><br>
      hobby: sports<input type="checkbox" name="hobby1" value="sport"/> &nbsp;
             travel<input type="checkbox" name="hobby2" value="travel"/>&nbsp;
             music<input type="checkbox" name="hobby3" value="music"/><br><br>
      other: <textarea rows="5" cols="30" name="other">abcdefg</textarea><br><br>
      <input type="submit" name="submit1" value="save" />
      <input type="submit" name="submit2" value="exit" /><br><br>
    </form>
  </body>
</html>
```

(3) 点击save按钮后，客户端发出的HTTP请求，请求的URL: *http://202.116.76.22:8080/jsp/postShow.jsp*

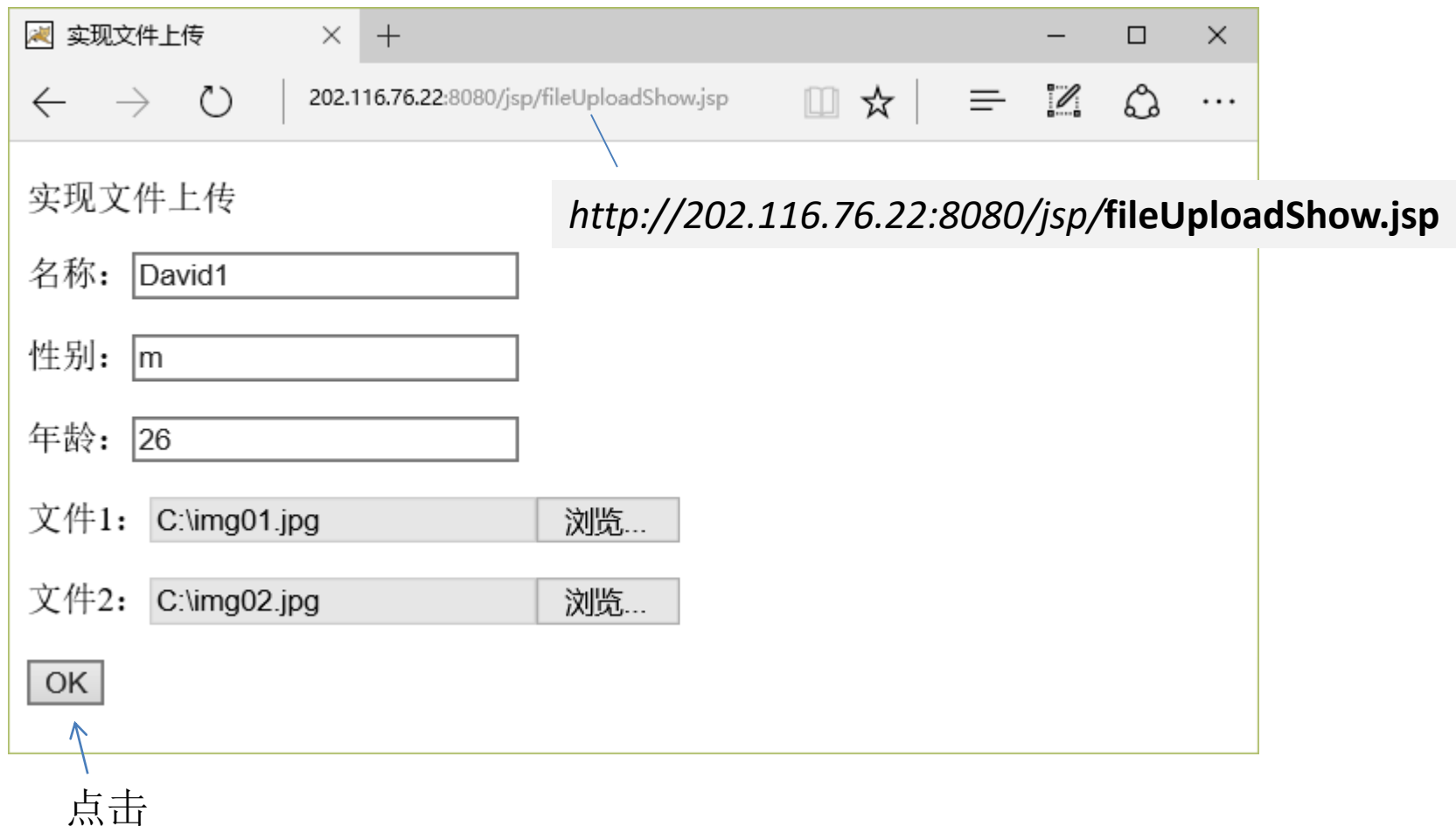
```
POST /jsp/postShow.jsp?user=wang HTTP/1.1
Accept: text/html, application/xhtml+xml, image/jxr, */*
Referer: http://202.116.76.22:8080/jsp/postInput.jsp
Accept-Language: zh-CN
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) ...Edge/13.10586
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: 202.116.76.22:8080
Content-Length: 105
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: JSESSIONID=D6F57FCC8AFAE8EE26D1533B41321156; username=admin;...
(空行)
stuid=1400001&name=David+Lee&grade=sophomore&hobby2=travel&hobby3=music&other=abcdefg+123456&submit1=save
```

(4) 服务器发出的HTTP响应，包含执行`postShow.jsp`产生的`html`文件

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=utf-8
Content-Length: 424
Date: Sun, 24 Apr 2016 18:25:51 GMT
(空行)
<!DOCTYPE HTML>
<html>
  <head>
    <title>postShow</title>
  </head>
  <body>
    user:   wang <br>
    name:   David Lee <br>
    stuid:  1400001 <br>
    school: null <br>
    grade:  sophomore <br>
    hobby1: null <br>
    hobby2: travel <br>
    hobby3: music <br>
    other:  abcdefg 123456 <br>
    method: POST <br>
    submit1:save <br>
    submit2:null <br>
  </body>
</html>
```

HTTP请求-post(multipart)

● 操作过程



实现文件上传

名称:

性别:

年龄:

文件1: 浏览...

文件2: 浏览...

点击

<http://202.116.76.22:8080/jsp/fileUploadShow.jsp>

http://202.116.76.22:8080/jsp/fileupload.jsp



● 源程序

fileUploadShow.jsp

```
<%@ page language="java" import="java.util.*" contentType="text/html;
charset=utf-8"%>
<!DOCTYPE HTML>
<html>
<head>
    <title>实现文件上传</title>
</head>
<body>
    <%request.setCharacterEncoding("utf-8");%>
    <p>实现文件上传</p>
    <form name="fileupload" action="fileupload.jsp" method="POST"
        enctype="multipart/form-data">
        <p>名称: <input type="text" name="name" size=24 value="David"></p>
        <p>性别: <input type="text" name="sex" SIZE=24 value="male"></p>
        <p>年龄: <input type="text" name="age" SIZE=24 value="28"></p>
        <p>文件1: <input type="file" name="file1" size=24></p>
        <p>文件2: <input type="file" name="file2" size=24></p>
        <p><input type="submit" name="submit" value="OK"></p>
    </form>
</body>
</html>
```


fileupload.jsp






```
<%@ page pageEncoding="utf-8" contentType="text/html; charset=utf-8"%>
<%@ page import="java.io.*"%>
<%@ page import="java.util.*"%>
<%@ page import="org.apache.commons.io.*"%>
<%@ page import="org.apache.commons.fileupload.*"%>
<%@ page import="org.apache.commons.fileupload.disk.*"%>
<%@ page import="org.apache.commons.fileupload.servlet.*"%>
<html><head><title>文件传输例子</title></head>
<body> <%request.setCharacterEncoding("utf-8");%>
    <% boolean isMultipart = ServletFileUpload.isMultipartContent(request); //检查
表单中是否包含文件
    if (isMultipart) {
        FileItemFactory factory = new DiskFileItemFactory();
        //factory.setSizeThreshold(yourMaxMemorySize); 此处可设置使用的内存最大值
        //factory.setRepository(yourTempDirectory); 文件临时目录
        ServletFileUpload upload = new ServletFileUpload(factory);
        //upload.setSizeMax(yourMaxRequestSize); 允许的最大文件尺寸
        List items = upload.parseRequest(request);
        for (int i = 0; i < items.size(); i++) {
            FileItem fi = (FileItem) items.get(i);
            if (fi.isFormField()) { //如果是表单字段
                %>
```

```

    <%=fi.getFieldName()%>:<%=fi.getString("utf-8")%>
<%} else { //如果是文件
    DiskFileItem dfi = (DiskFileItem) fi;
    if (!dfi.getName().trim().equals("")) { //getName()返回文件名称, 如果是空字
        符串, 说明没有选择文件。FilenameUtils.getName(filename);
    }
    %>
    文件被上传到服务上的实际位置:
    <%=new File(application.getRealPath("/file")
        + System.getProperty("file.separator")
        + FilenameUtils.getName(dfi.getName())).getAbsolutePath()%>
    <% dfi.write(new File(application.getRealPath("/file")
        + System.getProperty("file.separator")
        + FilenameUtils.getName(dfi.getName())));
    } } } } %>
</body>
</html>

```

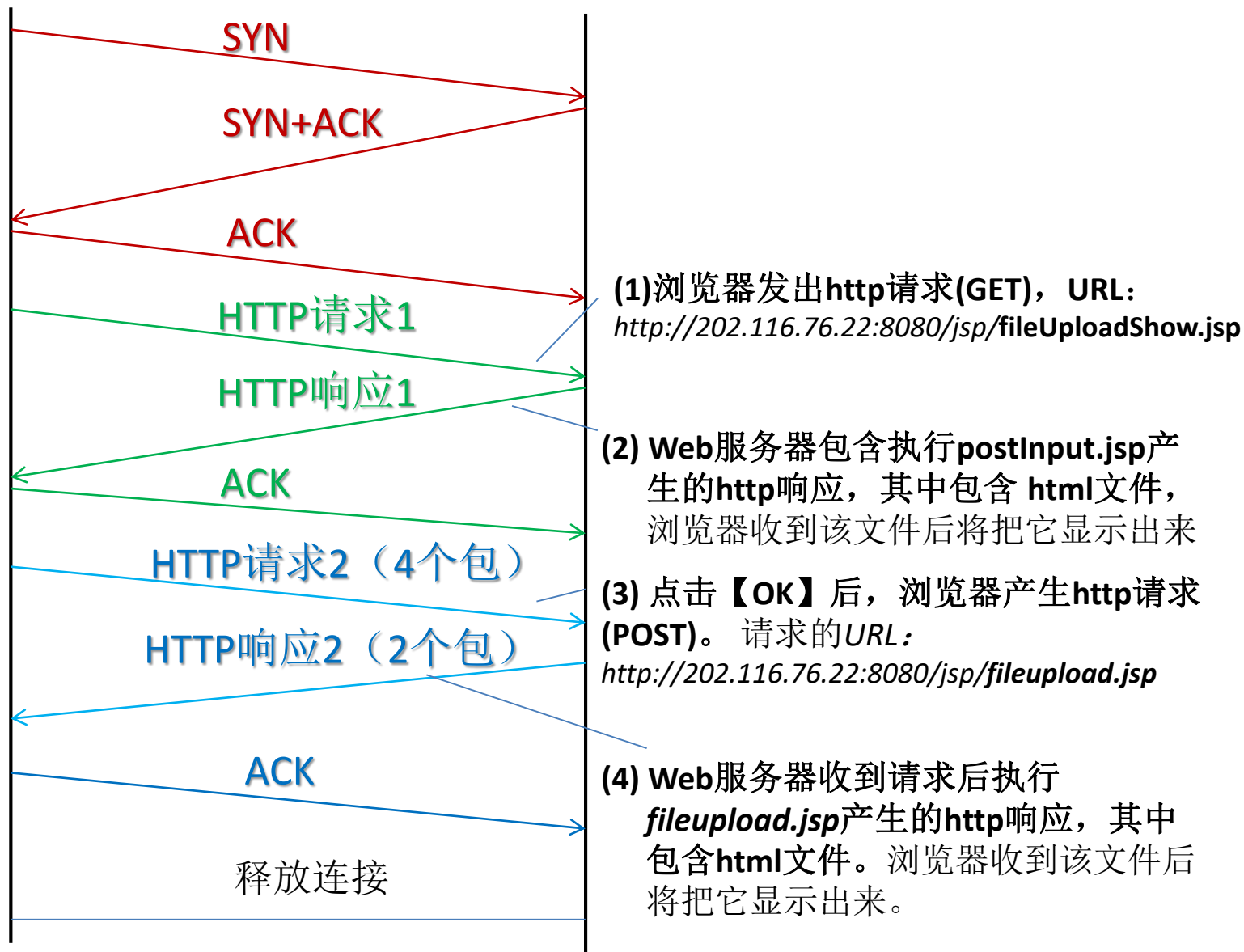
使用commons-io-1.4.jar和commons-fileupload-1.2.1.jar上传

Program Files > Apache Software Foundation > Tomcat 8.0 > webapps > jsp > WEB-INF > lib				
名称	修改日期	类型	大小	
 commons-fileupload-1.2.1.jar	2008/1/18 22:35	Executable Jar File	57 KB	
 commons-io-1.4.jar	2008/1/17 3:14	Executable Jar File	107 KB	
 jstl-1.2.jar	2015/9/13 16:42	Executable Jar File	405 KB	
 mysql-connector-java-5.1.36-bin.jar	2015/9/12 11:07	Executable Jar File	950 KB	
 standard-1.1.2.jar	2015/9/13 16:47	Executable Jar File	385 KB	

● 分析

客户端：浏览器

服务器端：Web服务器



Source	Destination	Protocol	Info
192.168.1.64	202.116.76.22	TCP	26005→8080 [SYN] Seq=0 win=65535 Len=0 MSS=
202.116.76.22	192.168.1.64	TCP	8080→26005 [SYN, ACK] Seq=0 Ack=1 win=8192
192.168.1.64	202.116.76.22	TCP	26005→8080 [ACK] Seq=1 Ack=1 win=262144 Len
192.168.1.64	202.116.76.22	HTTP	GET /jsp/fileUploadShow.jsp HTTP/1.1 (1)
202.116.76.22	192.168.1.64	HTTP	HTTP/1.1 200 OK (text/html) (2)
192.168.1.64	202.116.76.22	TCP	26005→8080 [ACK] Seq=609 Ack=779 win=261120
192.168.1.64	202.116.76.22	TCP	[TCP segment of a reassembled PDU]
192.168.1.64	202.116.76.22	TCP	[TCP segment of a reassembled PDU]
192.168.1.64	202.116.76.22	TCP	[TCP segment of a reassembled PDU]
192.168.1.64	202.116.76.22	HTTP	POST /jsp/fileupload.jsp HTTP/1.1 (JPEG JF (3)
202.116.76.22	192.168.1.64	TCP	8080→26005 [ACK] Seq=779 Ack=2865 win=65536
202.116.76.22	192.168.1.64	TCP	8080→26005 [ACK] Seq=779 Ack=4990 win=65536
202.116.76.22	192.168.1.64	HTTP	HTTP/1.1 200 OK (text/html) (4)
192.168.1.64	202.116.76.22	TCP	26005→8080 [ACK] Seq=4990 Ack=1361 win=2606
202.116.76.22	192.168.1.64	TCP	8080→26005 [FIN, ACK] Seq=1361 Ack=4990 win
192.168.1.64	202.116.76.22	TCP	26005→8080 [ACK] Seq=4990 Ack=1362 win=2606
192.168.1.64	202.116.76.22	TCP	26005→8080 [FIN, ACK] Seq=4990 Ack=1362 win
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 26005→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 26005→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 26005→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 26005→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	[TCP Retransmission] 26005→8080 [FIN, ACK]
192.168.1.64	202.116.76.22	TCP	26005→8080 [RST, ACK] Seq=4991 Ack=1362 win

(3) 点击【OK】后，浏览器产生http请求(POST)。
请求的URL: *http://202.116.76.22:8080/jsp/fileupload.jsp*

```
POST /jsp/fileupload.jsp HTTP/1.1
Accept: text/html, application/xhtml+xml, image/jxr, */*
Referer: http://202.116.76.22:8080/jsp/fileUploadShow.jsp
Accept-Language: zh-CN
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) ...Edge/13.10586
Content-Type: multipart/form-data; boundary=-----
                                     7e0e225a24c0

Accept-Encoding: gzip, deflate
Host: 202.116.76.22:8080
Content-Length: 3585
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: JSESSIONID=F8EE2044A4082AF186E64069A140624F; username=admin;...
(空行)
-----7e0e225a24c0
Content-Disposition: form-data; name="name"
(空行)
David1
-----7e0e225a24c0
Content-Disposition: form-data; name="sex"
(空行)
m
```

* **Referer**指出产生该请求的网页的URL，从中可以提取**host**。可用于判断是否盗链，即是否从其它网站链接到这里。

-----7e0e225a24c0
Content-Disposition: form-data; name="age"

(空行)

26

-----7e0e225a24c0
Content-Disposition: form-data; name="file1"; filename="img01.jpg"
Content-Type: image/jpeg

(空行)

JFIFHH"ExifMM*C

C'"

}!1AQa"q2#BR\$3br

%&'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz

w!1AQaq"2B #3Rbr

\$4%&'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz?H>xN\$m&(1agxkh5e#q/J?`Ho/o
z9x^hw\$FpIR7caWB|kkdfG AqGI.X>TQ&nmZf%ZUQkMM5NKKkXb.eU

7|9<G~Zn0,ojA.F*05'O_ZrK>soFWD\$, (@TI1t?4k^.uv:*v #Y9

=|?[1m0>d6[J+]A:G8v2N@(E5xLvvyo?6BE-

9\$)+pjKf<u)Gj}x\$uU[(\$sj;r,~PxY&qBNk&w6vE<]pF*<e0

\3Lx\Tu>e|Dr*?jc" xAi],/smh0NVNj(qfaZIz[k1QjB6hT~7s(b:a[

-----7e0e225a24c0

Content-Disposition: form-data; name="file2"; filename="img02.jpg"

Content-Type: image/jpeg

(空行)

JFIFHH"ExifMM*C

C"

}!1AQa"q2#BR\$3br

%&'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz

w!1AQa"2B #3Rbr

\$4%&'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz?H>xN\$m&(1agxAUkGk7ZZ^KkpO\$r

r 7lo/oz9x^hw\$FpIR7c}T!j|P]dG`\$`o`_^#K-AB2%{<WFtb90XkG|a x:"

mlnc/QD7;`yj@k'~7VIfPi'ed,_|r@qM~_.)xw5:e|kGjP`;4bAA:|oplRR)8uQ]/'Ju)ESMKo

&K[{}I:8v2N@(M5h{=2[IQbZ rISE|NE4'Y{HGG^ ,`4]jVt(G+I6K/_m9i|S%H;[

i(JEWey+.zSCO;|ux,(N7c;>_Y~y#mYn"[&)cJ8H

I\$(qN?On|+XyFTO7K?-;wU_~ 7s(OVt;

-----7e0e225a24c0

Content-Disposition: form-data; name="submit"

(空行)

OK

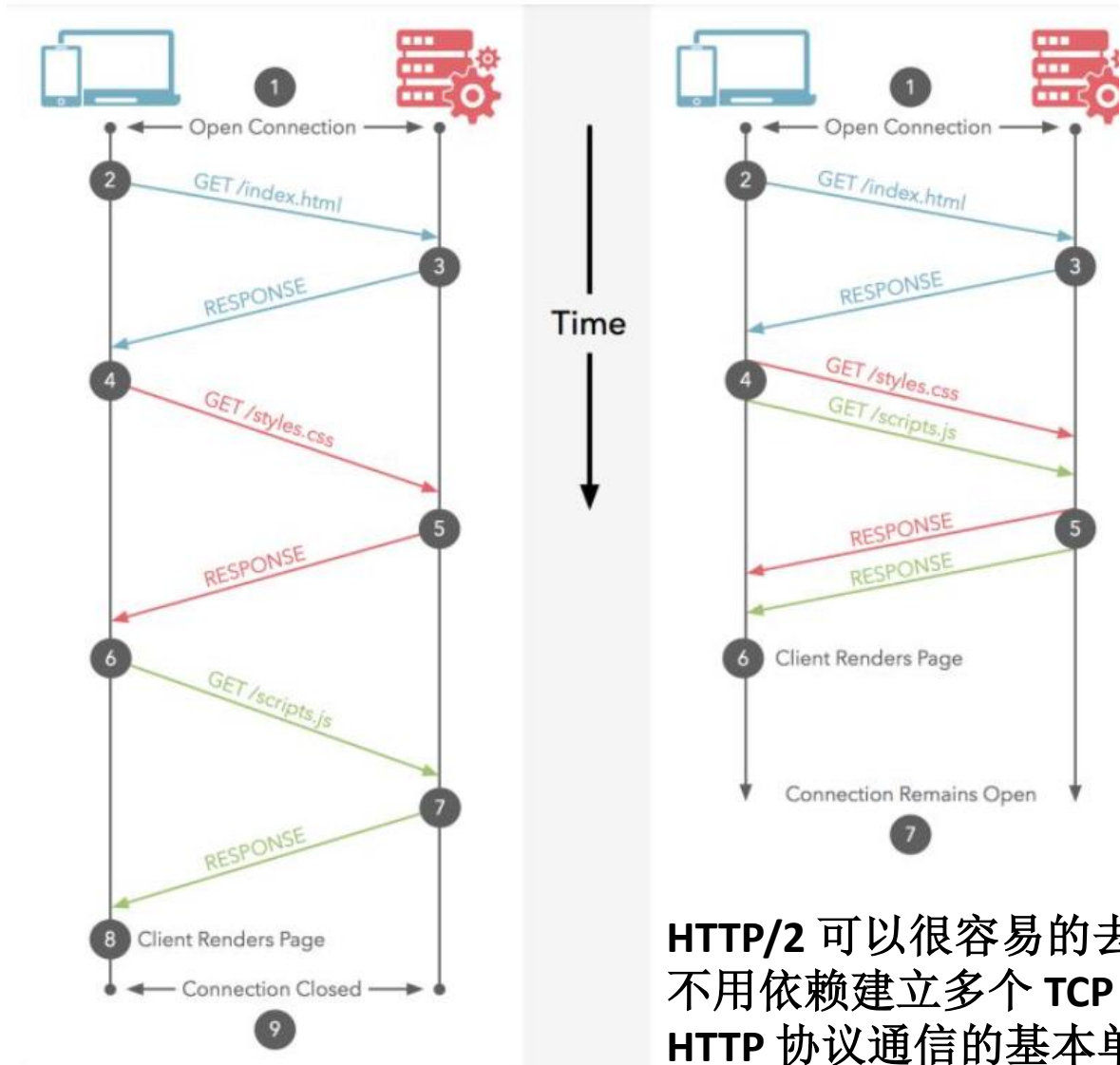
-----7e0e225a24c0--

HTTP 2.0

- HTTP 2.0采用二进制格式而非文本格式。二进制协议解析起来更高效、“线上”更紧凑，更重要的是错误更少。
- HTTP 2.0是完全多路复用的，而非有序并阻塞的。因为它能同时处理很多消息的请求和响应;甚至可以在传输过程中将一个消息跟另外一个掺杂在一起。
- HTTP 2.0使用报头压缩来降低开销。头部的轻微压缩都可以使很多http请求用一个包就可以解决，从而减少了发送数据包的数量。
- HTTP 2.0让服务器可以将响应主动“推送”到客户端缓存中。当浏览器请求一个网页时，服务器将会发回HTML，在服务器开始发送JavaScript、图片和CSS前，服务器需要等待浏览器解析HTML和发送所有内嵌资源的请求。服务器推送服务通过“推送”那些它认为客户端将会需要的内容到客户端的缓存中，以此来避免往返的延迟。

[参考](#) [参考](#) [参考](#)

HTTP/2.0 相比1.0有哪些重大改进？



HTTP/2 可以很容易的去实现多流并行而不用依赖建立多个 **TCP** 连接，**HTTP/2** 把 **HTTP** 协议通信的基本单位缩小为一个一个的帧。

附录1、HTTP状态码

- (1) 100~199: 表示成功接收请求, 要求客户端继续提交下一次请求才能完成处理。
- (2) 200~299: 表示成功接收请求, 并已经完成处理。
- (3) 300~399: 为完成请求, 客户端需要进一步细化请求。
- (4) 400~499: 客户端请求有错误。
- (5) 500~599: 服务器端发送错误。

状态码	消息	描述
100	Continue	只有一部分请求被服务器接收, 但只要没被服务器拒绝, 客户端就会延续这个请求
101	Switching Protocols	服务器交换机协议
200	OK	请求被确认
201	Created	请求时完整的, 新的资源被创建
202	Accepted	请求被接受, 但未处理完
203	Non-authoritative Information	
204	No Content	
205	Reset Content	
206	Partial Content	
300	Multiple Choices	一个超链接表, 用户可以选择一个超链接并访问, 最大支持5个超链接
301	Moved Permanently	被请求的页面已经移动到了新的URL下
302	Found	被请求的页面暂时性地移动到了新的URL下
303	See Other	被请求的页面可以在一个不同的URL下找到
304	Not Modified	
305	Use Proxy	

状态码	消息	描述
306	<i>Unused</i>	已经不再使用此状态码，但状态码被保留
307	Temporary Redirect	被请求的页面暂时性地移动到了新的URL下
400	Bad Request	服务器无法识别请求
401	Unauthorized	被请求的页面需要用户名和密码
402	Payment Required	<i>目前还不能使用此状态码</i>
403	Forbidden	禁止访问所请求的页面
404	Not Found	服务器无法找到所请求的页面
405	Method Not Allowed	请求中所指定的方法不被允许
406	Not Acceptable	服务器只能创建一个客户端无法接受的响应
407	Proxy Authentication Required	在请求被服务前必须认证一个代理服务器
408	Request Timeout	请求时间超过了服务器所能等待的时间，连接被断开
409	Conflict	请求有矛盾的地方
410	Gone	被请求的页面不再可用
411	Length Required	"Content-Length"没有被定义，服务器拒绝接受请求
412	Precondition Failed	请求的前提条件被服务器评估为false
413	Request Entity Too Large	因为请求的实体太大，服务器拒绝接受请求
414	Request-url Too Long	服务器拒绝接受请求，因为URL太长。多出现在把"POST"请求转换为"GET"请求时所附带的大量查询信息
415	Unsupported Media Type	服务器拒绝接受请求，因为媒体类型不被支持
417	Expectation Failed	
500	Internal Server Error	请求不完整，服务器遇见了出乎意料的状态
501	Not Implemented	请求不完整，服务器不提供所需要的功能
502	Bad Gateway	请求不完整，服务器从上游服务器接受了一个无效的响应
503	Service Unavailable	请求不完整，服务器暂时重启或关闭
504	Gateway Timeout	网关超时
505	HTTP Version Not Supported	服务器不支持所指定的HTTP版本

附录2、请求和响应头部

- HTTP1.1对HTTP1.0的改进：
 - (1) 一个TCP连接可以传送多个HTTP请求和响应。
 - (2) 可以采用流水线方式，即多个请求和响应过程可以重叠。
 - (3) 增加了更多的请求头和响应头。

- 请求方式:

- ✓ 主要的请求方式:

GET	请求获取URI所标识的资源
POST	在URI所标识的资源后附加新的数据
HEAD	请求获取由URI所标识的资源的响应消息报头
PUT	请求服务器存储一个资源，并用URI作为其标识
DELETE	请求服务器删除URI所标识的资源
TRACE	请求服务器回送收到的请求信息，主要用于测试或诊断
CONNECT	保留将来使用
OPTIONS	请求查询服务器的性能，或者查询与资源相关的选项和应用举例

- ✓ GET请求方式的请求参数加在请求行中，一般在1K以下，在传递请求参数时，在浏览器的url地址后以"?"分隔GET的请求参数，参数之间使用"&"分隔。
 - ✓ POST请求方式的请求参数在请求消息的内容中，大小无限制。
 - ✓ GET方法：在浏览器的地址栏中输入网址的方式访问网页时，浏览器采用GET方法向服务器获取资源，eg:GET /form.html HTTP/1.1 (CRLF)
 - ✓ POST方法要求被请求服务器接受附在请求后面的数据，常用于提交表单。

- HTTP通用信息头:

既能用于请求消息,也能用于响应消息,包括一些与被传输的实体内容没有关系的常用信息头字段。

- | | |
|---|------------------------|
| (1) Cache-control:no-cache | 不缓存。 |
| (2) Connection:close | 不保持连接。保持连接: keep-alive |
| (3) Date:Tue,11 Jul 2010 18:23:51 GMT | 请求或响应时间。 |
| (4) Pragma:no-cache | 不缓存消息。 |
| (5) Transfer-Encoding:chunked | 编码方式(分段编码传输)。 |
| (6) Via:HTTP/1.1 Proxy1,HTTP/1.1 Proxy2 | 指定途径的代理服务器。 |
| (7) Keep-Alive: 300 | 要求服务器保持连接300秒的连接。 |

* Connection为keep-alive时才能用Keep-Alive要求Web服务器保持连接多少秒

* Cache-Control(请求): 见后面。

- 请求头:

用于客户端发送的请求消息的头部。 * 标准日期: Tue, 06 May 2008 02:42:43 GMT

- | | |
|--|-------------------------|
| (1) Accept:text/html,image/* | 客户端可以接收的媒体类型。 |
| (2) Accept-Charset:Unicode-1-1.ISO8859-1 | 客户端接受的字符编码方式。 |
| (3) Accept-Encoding:gzip,compress | 客户端支持的数据压缩格式。 |
| (4) Accept-Language:en-gb,zh-cn | 客户端支持的语言。 |
| (5) Authorization:Basic enJEYMZQING== | 验证身份(Base64编码)。 |
| (6) Expect:100-continue | 需要服务器进一步操作。 |
| (7) Max-Forward:1 | 最大代理服务器数。 |
| (8) Accept-Range:bytes | 可以返回部分正文, none表示不接受 |
| (8) Range:bytes=100-599,800- | 返回100到599和800以后的正文。 |
| (9) Referer:http://www.google.com | 从哪个网页发出本http请求。 |
| (10) User-Agent:Mozilla/4.0 | 指定浏览器或客户端的类型。 |
| (11) Host:www.abc.com | 客户端想访问的域名和端口号 |
| (12) If-Match: w"1469-1461500473349" | 对象的 ETag 没有改变时才执行请求的动作。 |
| (13) If-None-Match | 对象的 ETag改变时才执行请求的动作。 |
| (14) If-Modified-Since: 标准日期 | 对象时间被修改时才执行 |
| (15) If-Unmodified-Since:标准日期 | 对象时间被修改时才执行 |
| (16) If-Range | 没改变则按照Range发送, 否则全部发送 |

- 响应头:

用作实体内容的元信息。

(1) Allow:GET,POST	允许客户端请求方式。
(2) Content-Encoding:gzip	压缩编码方式。
(3) Content-Language:zh-cn	服务器返回的文档语言。
(4) Content-Length:80	实体内容大小。
(5) Content-Location:http://www.google.com	请求资源所在位置。
(6) Content-Range:bytes 2543-4532	返回指定部分内容。
(7) Content-Type:text/html;charset=utf-8	实体内容的MIME类型和编码。
(8) Age:2	代理服务器响应的实体的缓存时间(s)
(9) ETag:1469-1461500473349	文件的标记，文件改变ETag也会改变
(10) Expires:Sat, 23 May 2009 10:02:12 GMT	过期时间
(11) Last-Modified: Tue, 06 May 2008 02:42:43 GMT	对象的最后修改时间
(12) Location	告诉客户端对象的新位置
(13) Pragma: no-cache	相当于 Cache-Control: no-cache
(14) Server: Apache/2.0.61 (Unix)	Web服务器版本
(15) Transfer-Encoding: chunked	响应体的编码方式
(16) Vary: Accept-Encoding	告诉 Cache 服务器响应后续请求的条件

*ETag用于文件改变过快一直modified检测不到（秒）

- 扩展头:

在HTTP1.1规范中没有定义的头字段，被当做实体扩展头处理。

(1) Refresh:1 告诉浏览器每1秒刷新访问1次。

Refresh:1;url=http://www.google.com 过1秒，跳转到指定页面。

(2) Content-Type:application/octet-Stream

Content-Disposition:attachment;filename=aaa.zip

让用户将响应的内容保存在一个文件中。

附录3、cache-control

网页缓存由 HTTP消息头中的Cache-control控制，常见取值有private、no-cache、max-age、must-revalidate等，默认为private。

* Cache-Control(请求): 。

- no-cache 不要缓存的实体，要求现在从WEB服务器去取
- max-age 只接受 Age 值小于 max-age 值，并且没有过期的对象
- max-stale 可以接受过去的对象，但是过期时间必须小于 max-stale 值
- min-fresh 接受其新鲜生命期大于其当前 Age 跟 min-fresh 值之和的缓存对象
- no-store 不允许缓存

* Cache-Control(响应):

- public 可以用 Cached 内容回应任何用户
- private 只能用缓存内容回应先前请求该内容的那个用户
- no-cache 可以缓存，但是只有在跟WEB服务器验证了其有效后，才能返回给客户端
- max-age 本响应包含的对象的过期时间
- no-store 不允许缓存
- must-revalidate: 强制页面不缓存，作用与no-cache相同，但更严格，强制意味更明显。

其作用根据不同的重新浏览方式，分为以下几种情况：

(1) 打开新窗口

值为**private**、**no-cache**、**must-revalidate**，那么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**值，那么在此值内的时间里就不会重新访问服务器，例如：**Cache-control: max-age=5**(表示当访问此网页后的5秒内再次访问不会去服务器)

(2) 在地址栏回车

值为**private**或**must-revalidate**则只有第一次访问时会访问服务器，以后就不再访问。值为**no-cache**，那么每次都会访问。值为**max-age**，则在过期之前不会重复访问。

(3)、按后退按钮

值为**private**、**must-revalidate**、**max-age**，则不会重访问，值为**no-cache**，则每次都重复访问

(4) 按刷新按钮

无论为何值，都会重复访问。**Cache-control**值为“**no-cache**”时，访问此页面不会在Internet临时文章夹留下页面备份。另外，通过指定“**Expires**”值也会影响到缓存。例如，指定**Expires**值为一个早已过去的时间，那么访问此网时若重复在地址栏按回车，那么每次都会重复访问：**Expires: Fri, 31 Dec 1999 16:00:00 GMT**

CacheControl :no-cache

Pragma:no-cache

Expires : -1

Expires是个好东东，如果服务器上的网页经常变化，就把它设置为-1，表示立即过期。如果一个网页每天凌晨1点更新，可以把**Expires**设置为第二天的凌晨1点。当HTTP1.1服务器指定 **CacheControl = no-cache**时，浏览器就不会缓存该网页。旧式 HTTP 1.0 服务器不能使用 **Cache-Control** 标题。所以为了向后兼容 HTTP 1.0 服务器，IE使用**Pragma:no-cache** 标题对 HTTP 提供特殊支持。

如果客户端通过安全连接 (<https://>)/与服务器通讯，且服务器在响应中返回 **Pragma:no-cache** 标题，则 Internet Explorer不会缓存此响应。注意：**Pragma:no-cache** 仅当在安全连接中使用时才防止缓存，如果在非安全页中使用，处理方式与 **Expires:-1**相同，该页将被缓存，但被标记为立即过期。

```
<head>
  <base href="<%=basePath%>">

  <title>My JSP 'index.jsp' starting page</title>
    <meta http-equiv="pragma" content="no-cache">
    <meta http-equiv="cache-control" content="no-cache">
    <meta http-equiv="expires" content="0">
    <meta http-equiv="keywords"
content="keyword1,keyword2,keyword3">
    <meta http-equiv="description" content="This is my page">
    <!--
    <link rel="stylesheet" type="text/css" href="styles.css">
    -->
</head>
```

附录4、参考资料

- <http://blog.csdn.net/wiwipetter/article/details/4559183>
- <http://xiejiaming.com/195.html>
- <http://www.studyofnet.com/news/166.html>
- <http://www.45it.com/net/201202/27993.htm>
- <http://www.jb51.net/article/34017.htm>