

# 网络原理之Http

- 掌握 http 原理, 重点掌握 http Request & Response 格式
- 掌握 http 中相关重点知识, 如请求方法, 属性, 状态码等
- 使用 java socket 编写简易版本 http server, 深刻理解原理
- 掌握 session 和 cookie
- 编写简单的 html, 感受前后端语言差别

## 1. Http 原理

### 1.1 理解为何要有应用层?

我们已经学过 TCP/IP, 已经知道目前数据能从客户端进程经过路径选择跨网络传送到服务器端进程[ IP+Port ], 可是, 仅仅把数据从A点传送到B点就完了吗? 这就好比, 在淘宝上买了一部手机, 卖家[ 客户端 ]把手机通过顺丰[ 传送+路径选择 ]送到买家[ 服务器 ]手里就完了吗? 当然不是, 买家还要使用这款产品, 还要在使用之后, 给卖家打分评论。所以, 我们把数据从A端传送到B端, TCP/IP 解决的是顺丰的功能, 而两端还要对数据进行加工处理或者使用, 所以我们还需要一层协议, 不关心通信细节, 关心应用细节!

这层协议叫做应用层协议。而应用是有不同的场景的, 所以应用层协议是有不同种类的, 其中经典协议之一的HTTP就是其中的佼佼者。那么, Http 是解决什么应用场景呢?

早期用户, 上网使用浏览器来进行上网, 而用浏览器上网阅读信息, 最常见的是查看各种网页【其实也是文件数据, 不过是一系列的 html 文档, 当然还有其他资源如图片, css, js 等】, 而要把网页文件信息通过网络传送到客户端, 或者把用户数据上传到服务器, 就需要 Http 协议【当然, http 作用不限于此】

### 1.2 再谈 "协议"

那如何理解应用层协议呢? 再回到我们刚刚说的买手机的例子, 顺丰相当于 TCP/IP 的功能, 那么买回来的手机都附带了说明书【产品介绍, 使用介绍, 注意事项等】, 而该说明书指导用户该如何使用手机【虽然我们都不看, 但是父母辈有部分是有看说明书的习惯的: )】, 此时的说明书可以理解为用户层协议

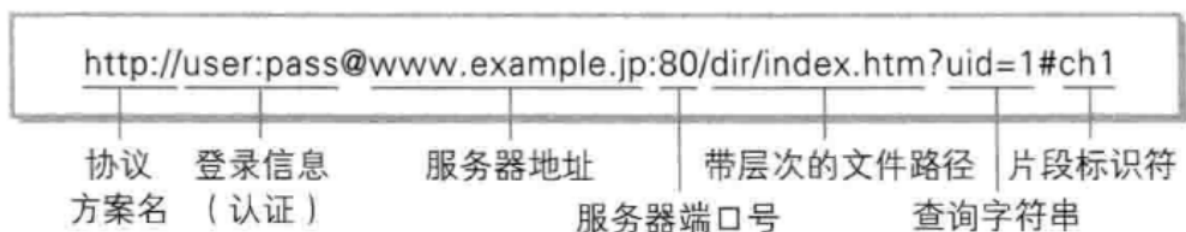
### 1.3 HTTP协议

虽然我们说, 应用层协议是我们程序猿自己定的。

但实际上, 已经有大佬们定义了一些现成的, 又非常好用的应用层协议, 供我们直接参考使用。HTTP(超文本传输协议)就是其中之一。

### 1.4 认识URL

平时我们俗称的 "网址" 其实就是说的 URL



## 1.5 urlencode和urldecode

像 / ? : 等这样的字符, 已经被url当做特殊意义理解了. 因此这些字符不能随意出现.  
比如, 某个参数中需要带有这些特殊字符, 就必须先对特殊字符进行转义.

转义的规则如下:

将需要转码的字符转为16进制, 然后从右到左, 取4位(不足4位直接处理), 每2位做一位, 前面加上%, 编码成%XY格式

例如:

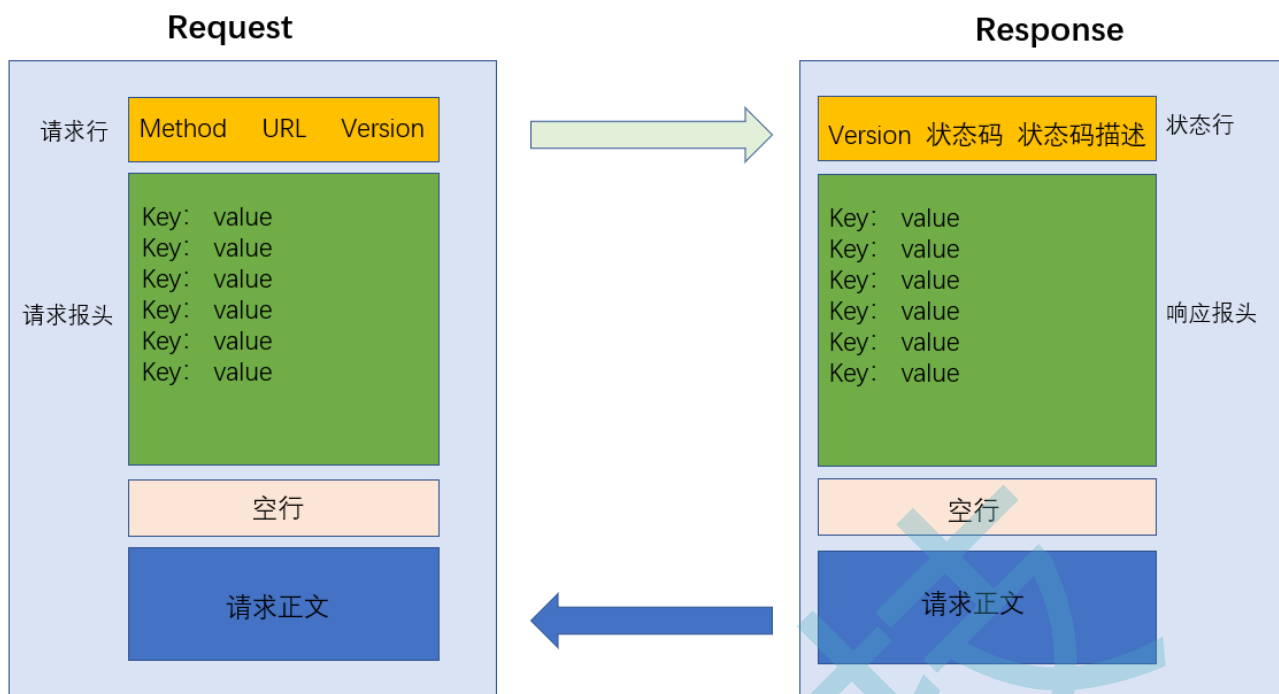


"+" 被转义成了 "%2B"

urldecode就是urlencode的逆过程;

[urlencode工具](#)

## 1.6 HTTP协议格式



## HTTP请求

```
POST http://job.xjtu.edu.cn/companyLogin.do HTTP/1.1
Host: job.xjtu.edu.cn
Connection: keep-alive
Content-Length: 36
Cache-Control: max-age=0
Origin: http://job.xjtu.edu.cn
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/61.0.3163.100 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/
png,*/*;q=0.8
Referer: http://job.xjtu.edu.cn/companyLogin.do
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8
Cookie: JSESSIONID=D628A75845A74D29D991DB47A461E4FC;
Hm_lvt_783e83ce0ee350e23a9d389df580f658=1504963710,1506661798;
Hm_lpv_783e83ce0ee350e23a9d389df580f658=1506661802

username=hgtz2222&password=22222222
```

- 首行: [方法] + [url] + [版本]
- Header: 请求的属性, 冒号分割的键值对; 每组属性之间使用\n分隔; 遇到空行表示Header部分结束
- Body: 空行后面的内容都是Body. Body允许为空字符串. 如果Body存在, 则在Header中会有一个Content-Length属性来标识Body的长度;

## HTTP响应

```
HTTP/1.1 200 OK
Server: YxlinkWAF
Content-Type: text/html; charset=UTF-8
Content-Language: zh-CN
Transfer-Encoding: chunked
Date: Fri, 29 Sep 2017 05:10:13 GMT

<!DOCTYPE html>
<html>
<head>
<title>西安交通大学就业网</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="IE=Edge">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="shortcut icon" href="/renovation/images/icon.ico">
<link href="/renovation/css/main.css" rel="stylesheet" media="screen" />
<link href="/renovation/css/art_default.css" rel="stylesheet" media="screen" />
<link href="/renovation/css/font-awesome.css" rel="stylesheet" media="screen" />
<script type="text/javascript" src="/renovation/js/jquery1.7.1.min.js"></script>
<script type="text/javascript" src="/renovation/js/main.js"></script><!--main-->
<link href="/style/warmTipsstyle.css" rel="stylesheet" type="text/css">
</head>
```

- 首行: [版本号] + [状态码] + [状态码解释]
- Header: 请求的属性, 冒号分割的键值对; 每组属性之间使用\n分隔; 遇到空行表示Header部分结束
- Body: 空行后面的内容都是Body. Body允许为空字符串. 如果Body存在, 则在Header中会有一个Content-Length属性来标识Body的长度; 如果服务器返回了一个html页面, 那么html页面内容就是在body中.

## 1.7 HTTP的方法

方法	说明	支持的HTTP 协议版本
GET	获取资源	1.0、1.1
POST	传输实体主体	1.0、1.1
PUT	传输文件	1.0、1.1
HEAD	获得报文首部	1.0、1.1
DELETE	删除文件	1.0、1.1
OPTIONS	询问支持的方法	1.1
TRACE	追踪路径	1.1
CONNECT	要求用隧道协议连接代理	1.1
LINK	建立和资源之间的联系	1.0
UNLINE	断开连接关系	1.0

其中最常用的就是GET方法和POST方法.

## 1.8 HTTP的状态码

	类别	原因短语
1XX	Informational ( 信息性状态码 )	接收的请求正在处理
2XX	Success ( 成功状态码 )	请求正常处理完毕
3XX	Redirection ( 重定向状态码 )	需要进行附加操作以完成请求
4XX	Client Error ( 客户端错误状态码 )	服务器无法处理请求
5XX	Server Error ( 服务器错误状态码 )	服务器处理请求出错

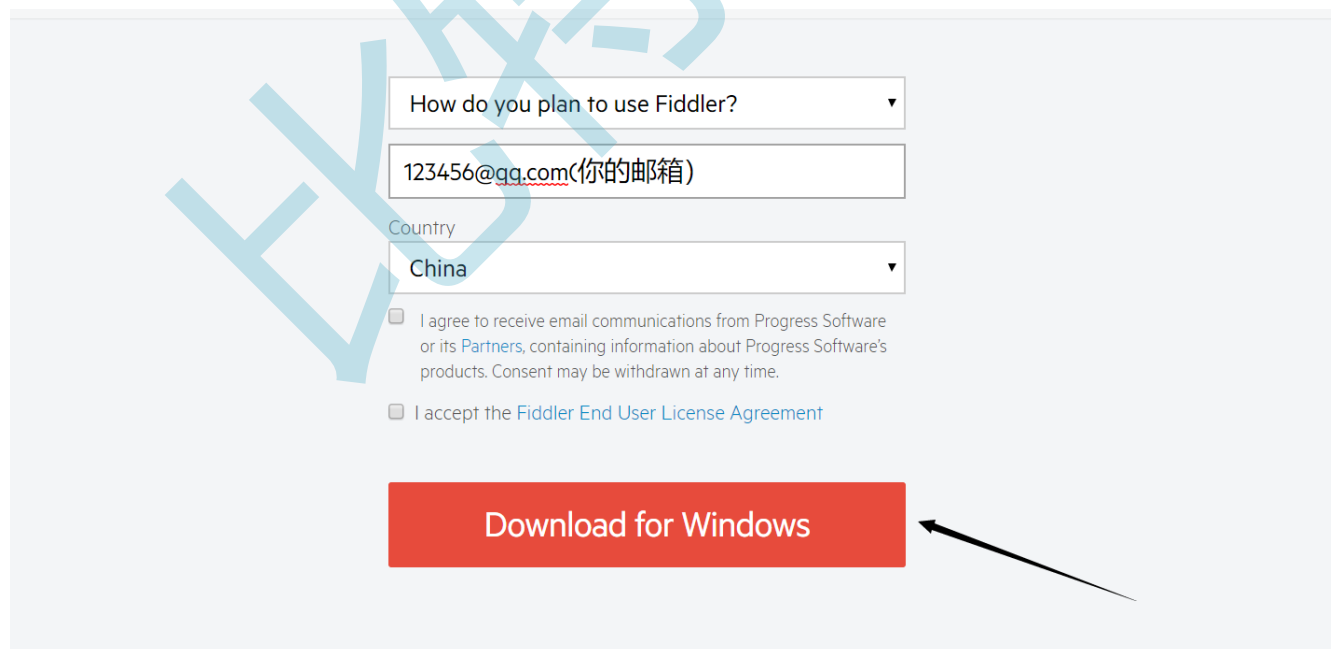
最常见的状态码, 比如 200(OK), 404(Not Found), 403(Forbidden), 302(Redirect, 重定向), 504(Bad Gateway)

## 1.9 HTTP常见Header

- Content-Type: 数据类型(text/html等)
- Content-Length: Body的长度
- Host: 客户端告知服务器, 所请求的资源是在哪个主机的哪个端口上;
- User-Agent: 声明用户的操作系统和浏览器版本信息;
- referer: 当前页面是从哪个页面跳转过来的;
- location: 搭配3xx状态码使用, 告诉客户端接下来要去哪里访问;
- Cookie: 用于在客户端存储少量信息. 通常用于实现会话(session)的功能;

## 1.10. 抓包工具 fiddler

为了研究http协议相关协议细节【主要是request和response, 状态码, 以及后续的cookie和session】, 我们引进一款抓包工具fiddler, 同学们可以去官网下载安装: <https://www.telerik.com/fiddler/>



How do you plan to use Fiddler? ▼

123456@qq.com(你的邮箱)

Country

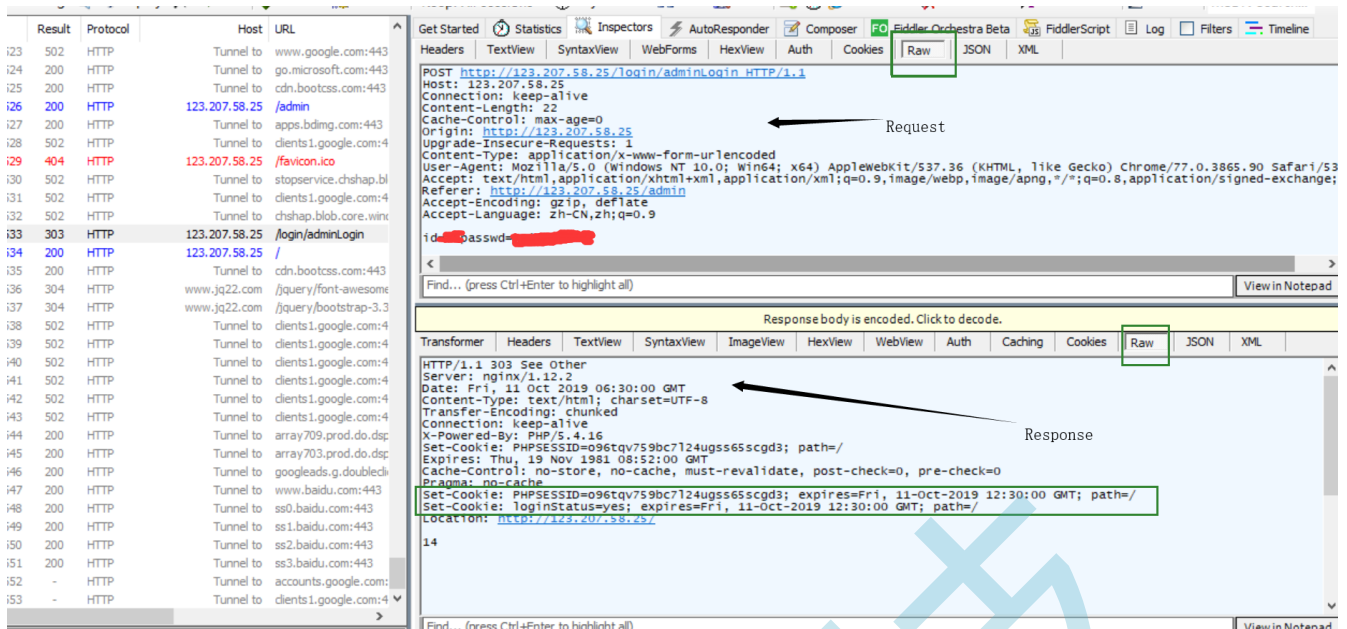
China ▼

☐ I agree to receive email communications from Progress Software or its [Partners](#), containing information about Progress Software's products. Consent may be withdrawn at any time.

☐ I accept the [Fiddler End User License Agreement](#)

**Download for Windows**

下载好之后, 打开之后, 就可以直接进行使用了



同学们可以根据实际情况，演示查看其他功能，比如重定向，404，上传文件等

## 1.11. 其他

[User-Agent里的历史故事](#)

## 2. Http Server

//简易版的http server v1 + 多线程 + 手动构建response 展示302 307 404效果

```
public class Server {

    /**
     * 端口号
     */
    private static final int PORT = 9999;

    /**
     * 统一编码
     */
    private static final String CHARSET = "UTF-8";

    public static void main(String[] args) throws IOException {
        ServerSocket serverSocket = new ServerSocket(PORT);
        ExecutorService POOL = Executors.newCachedThreadPool();
        try {
            while (true) {
                socket socket = serverSocket.accept();
                POOL.submit(new Runnable() {
                    @Override
                    public void run() {
                        try {
                            BufferedReader reader = new BufferedReader(
                                new InputStreamReader(socket.getInputStream()));

                            // 解析Http请求行
```



```

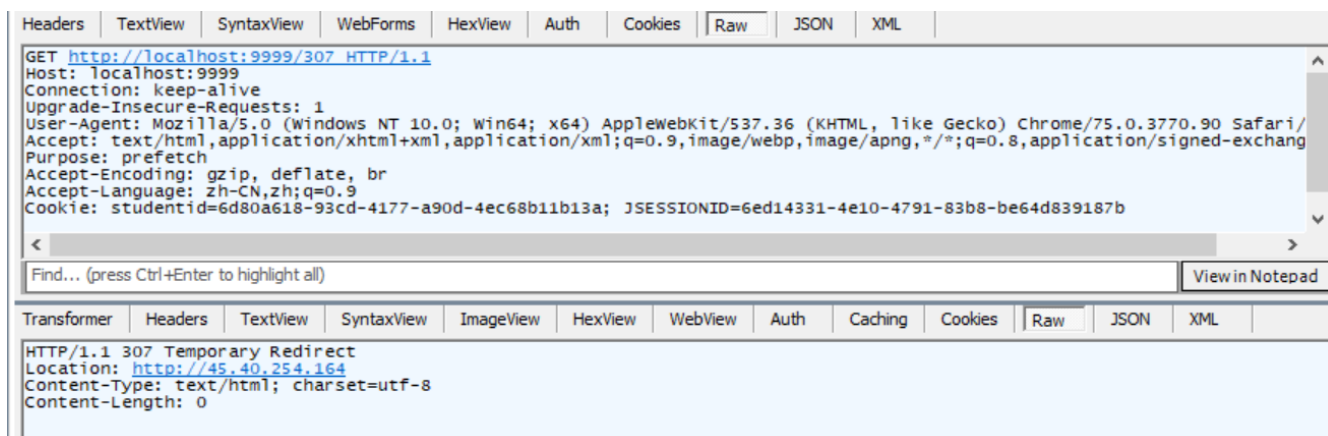
String httpLine = reader.readLine();
System.out.println("===="+httpLine);
String[] httpLineArray = httpLine.split(" ");
String requestMethod = httpLineArray[0];
String requestUri = httpLineArray[1];
String requestVersion = httpLineArray[2];

// 解析请求头
String requestHeader;
Map<String, String> headers = new HashMap<>();
while ((requestHeader = reader.readLine()) != null &&
requestHeader.length() != 0) {
    String[] headerArray = requestHeader.split(":");
    headers.put(headerArray[0].trim(), headerArray[1].trim());
}

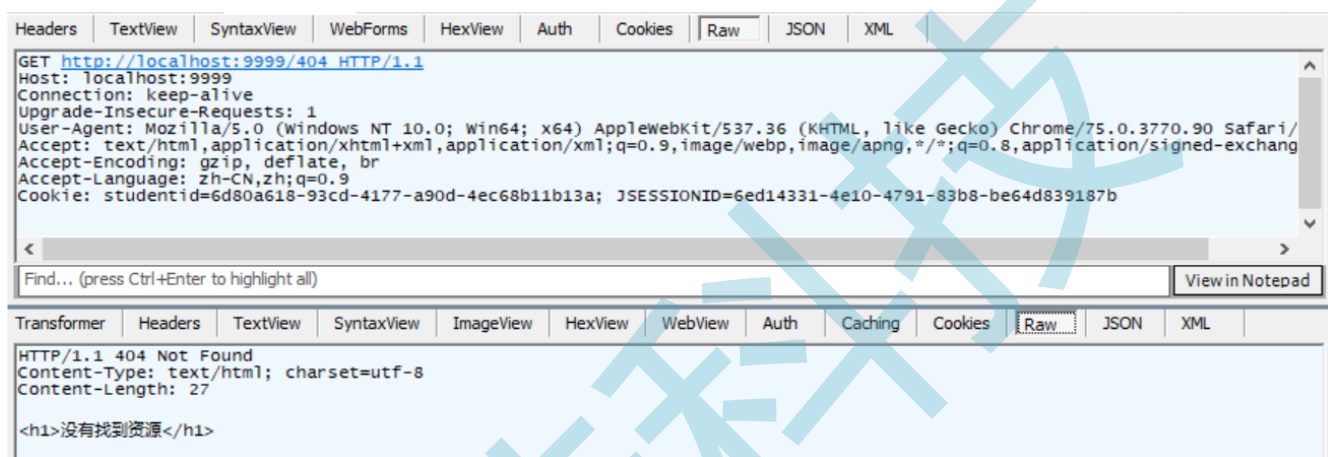
PrintWriter writer = new PrintWriter(new OutputStreamWriter(
    socket.getOutputStream(), CHARSET), true);
String content;
if("/307".equals(requestUri)) {
    writer.println("HTTP/1.1 307 Temporary Redirect");
    writer.println("Location: http://45.40.254.164");
    content = "";
}else if("/404".equals(requestUri)){
    writer.println("HTTP/1.1 404 Not Found");
    content = "<h1>没有找到资源</h1>";
}else{
    writer.println("HTTP/1.1 200 OK");
    content = "<h1>My Http Server</h1>";
}
writer.println("Content-Type: text/html; charset=utf-8");
writer.println("Content-Length:
"+content.getBytes(CHARSET).length);
writer.println();
writer.println(content.toString());
socket.close();
}catch(Exception e){
    e.printStackTrace();
}
    }
    });
}
}catch(Exception e){
    e.printStackTrace();
}
}
}

```

查看307重定向: <http://localhost:9999/307>



查看404找不到资源: <http://localhost:9999/404>



### 3. session和cookie

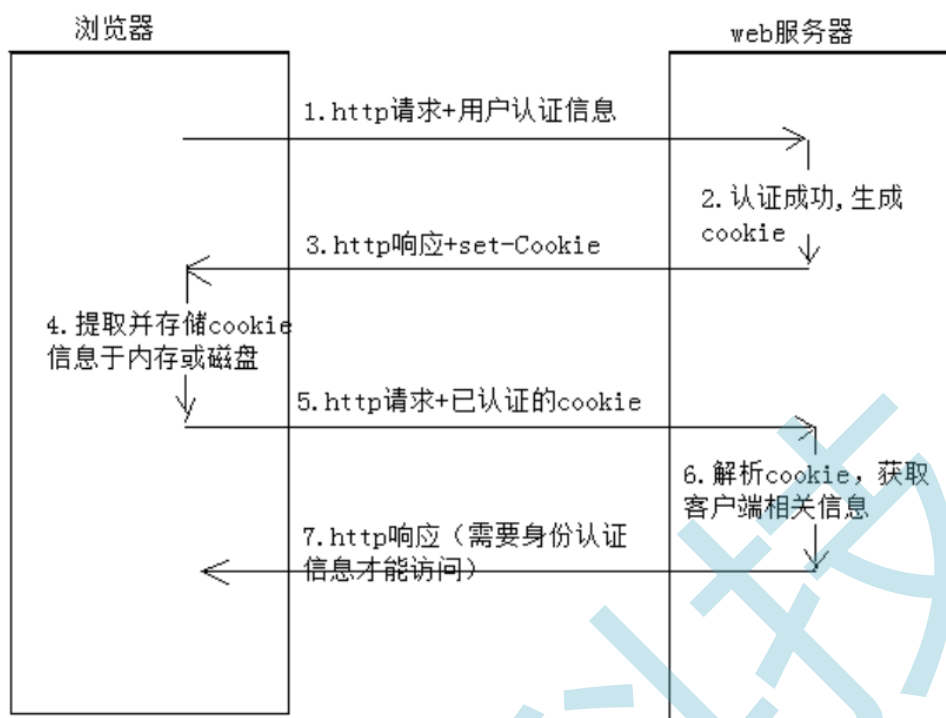
#### 3.1 用户信息

Http 是一个无状态协议, 就是说这一次请求和上一次请求是没有任何关系的, 互不认识的, 没有关联的。这种无状态的好处是快速。坏处是需要进行用户状态保持的场景时[比如, 登陆状态下进行页面跳转, 或者用户信息多页面共享等场景], 必须使用一些方式或者手段比如: session 和 cookie

#### 3.2 cookie

如上所述, Http 是一个无状态的协议, 但是访问有些资源的时候往往需要经过认证的账户才能访问, 而且要一直保持在线状态, 所以, cookie 是一种在浏览器端解决的方案, 将登陆认证之后的用户信息保存在本地浏览器中, 后面每次发起http请求, 都自动携带上该信息, 就能达到认证用户, 保持用户在线的作用, 具体如下图:

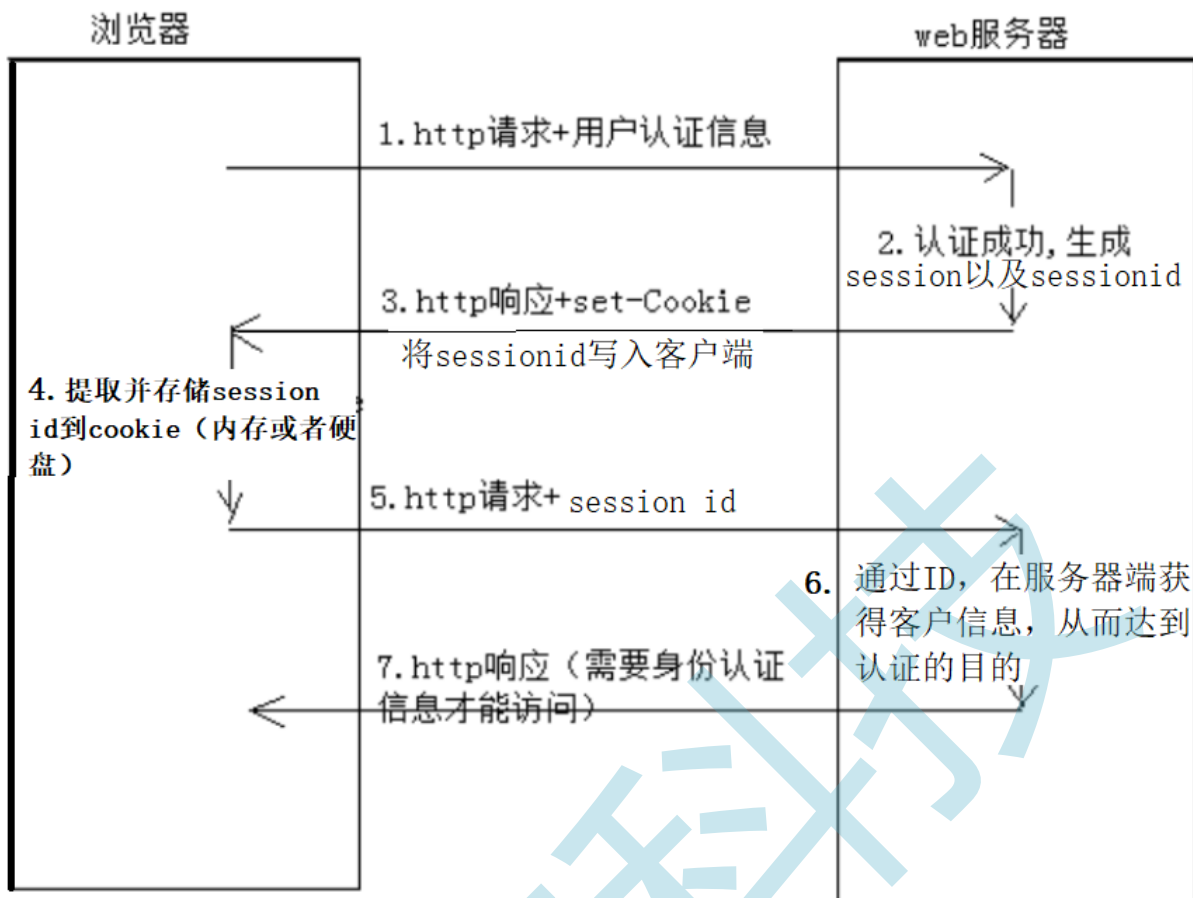




设置cookie的方法在 Http 的 Response 报头中可以携带 Set-Cookie 字段来完成，后面会有演示。

### 3.3 session

而将用户敏感信息放到本地浏览器中，能解决一定的问题，但是又引进了新的安全问题，一旦cookie丢失，用户信息泄露，也很容易造成跨站攻击，所以有了另一种解决方法，将用户敏感信息保存至服务器，而服务器本身采用md5算法或相关算法生成唯一值（session id），将该值保存至客户端浏览器，随后，客户端的后续请求，浏览器都会自动携带该id，进而再在服务器端认证，进而达到状态保持的效果



### 3.4 cookie vs session

两者有什么区别呢？

- Cookie以文本文件格式存储在浏览器中，而session存储在服务端
- 因为每次发起http请求，都要携带有效Cookie信息，所以Cookie一般都有大小限制，以防止增加网络压力，一般不超过4k
- 可以轻松访问cookie值但是我们无法轻松访问会话值，因此session方案更安全

### 3.5 本地禁止cookie

经过上面的学习，我们能看出来，要使用session，其实还是需要使用cookie机制来保存session id的，那么万一在客户端cookie机制被禁掉了，那session貌似也就无法使用了？其实替代方法是有的

- 经常被使用的一种技术叫做URL重写，就是把session id直接附加在URL路径的后面。
- 还有一种技术叫做**表单隐藏字段**。就是服务器会自动修改表单，添加一个隐藏字段，以便在表单提交时能够把session id传递回服务器

## 4. Http Server + session&cookie

```
public class Request {  
  
    /**  
     * 请求方法  
     */  
}
```

```

private String method;

/**
 * 请求路径
 */
private String uri;

/**
 * Http版本
 */
private String version;

/**
 * 请求头
 */
private Map<String, String> headers = new HashMap<>();

/**
 * 请求参数
 */
private Map<String, String> parameters = new HashMap<>();

/**
 * 统一编码
 */
private static final String CHARSET = "UTF-8";

private Request() {
}

/**
 * 解析HttpRequest并生成请求对象
 *
 * @param is HttpRequest流
 * @return 请求对象
 * @throws IOException
 */
public static Request build(InputStream is) throws IOException {
    Request request = new Request();
    BufferedReader reader = new BufferedReader(new InputStreamReader(is, CHARSET));
    // 处理请求行
    String[] requestLineArray = reader.readLine().split(" ");
    request.setMethod(requestLineArray[0]);
    request.setUri(requestLineArray[1]);
    request.setVersion(requestLineArray[2]);

    // 处理请求头
    String requestHeader;
    while ((requestHeader = reader.readLine()) != null && requestHeader.length() != 0)
    {
        String[] headerArray = requestHeader.split(":");
        request.addHeader(headerArray[0].trim(), headerArray[1].trim());
    }
}

```

```
    }  
    return request;  
}  
  
public void addHeader(String key, String value) {  
    headers.put(key, value);  
}  
  
public void addParameter(String key, String value) {  
    parameters.put(key, value);  
}  
  
@Override  
public String toString() {  
    return "HttpRequest{" +  
        "method='" + method + '\'' +  
        ", uri='" + uri + '\'' +  
        ", version='" + version + '\'' +  
        ", headers=" + headers +  
        ", parameters=" + parameters +  
        '}';  
}  
  
public String getMethod() {  
    return method;  
}  
  
public void setMethod(String method) {  
    this.method = method;  
}  
  
public String getUri() {  
    return uri;  
}  
  
public void setUri(String uri) {  
    this.uri = uri;  
}  
  
public String getVersion() {  
    return version;  
}  
  
public void setVersion(String version) {  
    this.version = version;  
}  
  
public Map<String, String> getHeaders() {  
    return headers;  
}  
  
public String getHeader(String key) {  
    return headers.get(key);  
}
```

```
}

public void setHeaders(Map<String, String> headers) {
    this.headers = headers;
}

public Map<String, String> getParameters() {
    return parameters;
}

public String getParameter(String key) {
    return parameters.get(key);
}

public void setParameters(Map<String, String> parameters) {
    this.parameters = parameters;
}
}
```

```
public class Response {

    /**
     * 响应状态码
     */
    private int status;

    /**
     * 响应信息
     */
    private String message;

    /**
     * 响应头
     */
    private Map<String, String> headers = new HashMap<>();

    /**
     * 响应打印流
     */
    private PrintWriter writer;

    /**
     * 统一编码
     */
    private static final String CHARSET = "UTF-8";

    /**
     * 响应内容
     */
    private StringBuilder body = new StringBuilder();

    private Response(){
```

```

}

public static Response build(OutputStream os) throws UnsupportedOperationException {
    Response response = new Response();
    response.writer = new PrintWriter(new OutputStreamWriter(
        os, CHARSET), true);
    return response;
}

/**
 * 返回信息
 * @param content
 */
public Response println(String content){
    body.append(content+"\r\n");
    return this;
}

public void flush() throws IOException {
    writer.println("HTTP/1.1 "+status+" "+message);
    writer.println("Content-Type: text/html; charset=utf-8");
    writer.println("Content-Length: "+body.toString().getBytes(CHARSET).length);
    headers.forEach((k,v)->{
        writer.println(k+": "+v);
    });
    writer.println();
    writer.println(body.toString());
}

public void addHeader(String key, String value) {
    headers.put(key, value);
}

public int getStatus() {
    return status;
}

public void setStatus(int status) {
    this.status = status;
}

public String getMessage() {
    return message;
}

public void setMessage(String message) {
    this.message = message;
}

public Map<String, String> getHeaders() {
    return headers;
}

```



```

public void setHeaders(Map<String, String> headers) {
    this.headers = headers;
}

public Writer getWriter() {
    return writer;
}

public void setWriter(PrintWriter writer) {
    this.writer = writer;
}
}

public class Server {

    /**
     * 端口号
     */
    private static final int PORT = 9999;

    /**
     * 统一编码
     */
    private static final String CHARSET = "UTF-8";

    public static void main(String[] args) throws IOException {
        ServerSocket serverSocket = new ServerSocket(PORT);
        ExecutorService POOL = Executors.newCachedThreadPool();
        try {
            while (true) {
                Socket socket = serverSocket.accept();
                POOL.submit(new Runnable() {
                    @Override
                    public void run() {
                        try {
                            Request request = Request.build(socket.getInputStream());
                            Response response = Response.build(socket.getOutputStream());

                            PrintWriter writer = new PrintWriter(new OutputStreamWriter(
                                socket.getOutputStream(), CHARSET), true);
                            if("/307".equals(request.getUri())) {
                                response.setStatus(307);
                                response.setMessage("Temporary Redirect");
                                response.addHeader("Location", "http://45.40.254.164");
                                response.println("");
                            } else if("/404".equals(request.getUri())) {
                                response.setStatus(404);
                                response.setMessage("Not Found");
                                response.println("<h1>没有找到资源</h1>");
                            } else if("/setCookie".equals(request.getUri())) {
                                response.setStatus(200);
                                response.setMessage("OK");
                            }
                        } catch (Exception e) {
                            e.printStackTrace();
                        }
                    }
                });
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

        response.addHeader("Set-Cookie", "studentid=" +
UUID.randomUUID());

        response.println("<h1>Set Cookie OK</h1>");
    }else{
        response.setStatus(200);
        response.setMessage("OK");
        response.println("<h1>My Http Server</h1>");
    }
    response.flush();
    socket.close();
}catch(Exception e){
    e.printStackTrace();
}

    }
}

});
}

}catch(Exception e){
    e.printStackTrace();
}

}
}

```

设置Cookie: <http://localhost:9999/setCookie>

The screenshot shows a web browser window with the address bar displaying 'localhost:9999/setCookie'. The page content shows 'Set Cookie OK' in large black text. A red arrow points to the address bar with the text '点击打开' (Click to open). A cookie management popup is open on the right side of the browser window. The popup has a title '正在使用的 Cookie' (Cookies in use) and two tabs: '允许' (Allow) and '已禁止' (Already prohibited). The '允许' tab is selected. Below the tabs, it says '以下 Cookie 是系统在您查看此网页时设置的' (The following cookies are set by the system when you view this webpage). A list of cookies is shown, with 'localhost' expanded to show a 'Cookie' folder containing a cookie named 'studentid'. Below this, a table provides details for the 'studentid' cookie:

名称	studentid
内容	e7a98111-ce49-41e0-b992-b7ba3b90 added
域名	localhost
路径	/
为何发送	各种连接
创建时间	2019年10月24日星期四 下午2:27:47
到期时间	浏览会话结束时

At the bottom of the popup, there are three buttons: '禁止' (Prohibit), '删除' (Delete), and '完成' (Done).

```
Headers  TextView  SyntaxView  WebForms  HexView  Auth  Cookies  Raw  JSON  XML
GET http://localhost:9999/setCookie HTTP/1.1
Host: localhost:9999
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.90 Safari/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchang
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9

Find... (press Ctrl+Enter to highlight all) View in Notepad

Transformer  Headers  TextView  SyntaxView  ImageView  HexView  WebView  Auth  Caching  Cookies  Raw  JSON  XML
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 24
Set-Cookie: studentid=e7a98111-ce49-41e0-b992-b7ba3b90fdd0
<h1>Set Cookie OK</h1>
```

设置了Cookie后，访问该域名下的url时，都可以看到该Cookie值：<http://localhost:9999/200>

localhost:9999/200

localhost:9999/200

应用 企业邮箱 IPAddress 百度 搜狗搜索

# My Http Server

正在使用的 Cookie

允许 已禁止

以下 Cookie 是系统在您查看此网页时设置的

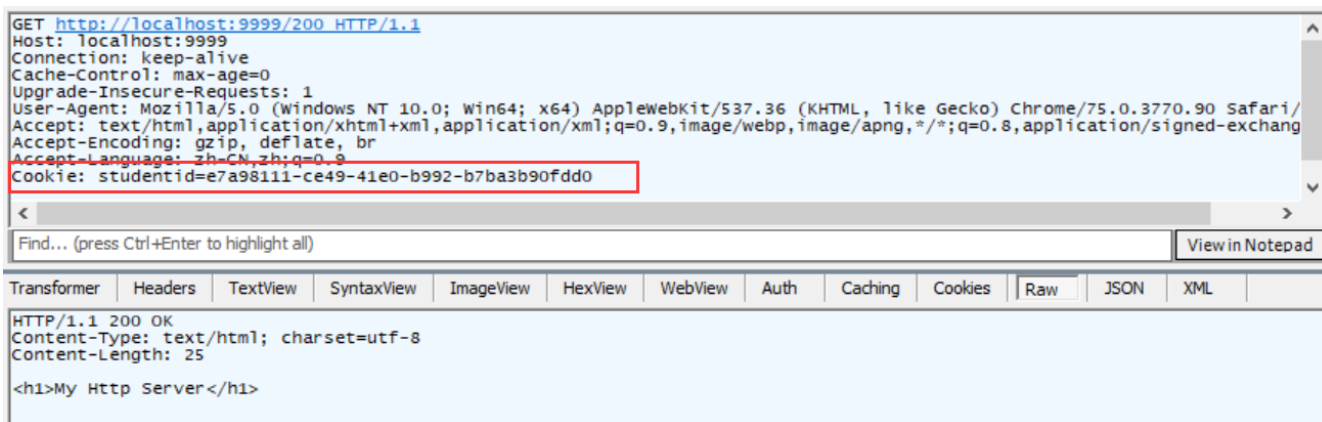
localhost

Cookie

studentid

名称	studentid
内容	e7a98111-ce49-41e0-b992-b7ba3b90fdd0
域名	localhost
路径	/
为何发送	各种连接
创建时间	2019年10月24日星期四 下午2:27:47
到期时间	浏览会话结束时

禁止 删除 完成



## 5. 简单的html

### 5.1 html 介绍

HTML是用于创建网页的语言。我们通过使用HTML标记标签创建html文档来创建网页。HTML代表超文本标记语言。HTML是一种标记语言，它是标记标签的集合。

HTML标签是由尖括号括起来的词，如 `<html>`，`<body>`。标签通常成对出现，例如 `<html>`和`</html>`。

一对中的第一个标签是开始标签;第二个标签是结束标签。如是开始标签，而`</html>`是结束标签，我们还可以将开始标签称为起始标签，结束标签称为闭合标签。

HTML文档结构至少要包括head, body两部分.如:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>html基础</title>
  </head>
  <body>
    <h3>同学们，我们要开启前端学习之旅啦...</h3>
  </body>
</html>
```

### 5.2 html 在 http server 中的显示

//后端接入html文件资源版的 v3

后面讲解html，直接使用我们的http server来进行演示

### 5.3 html 常见标签

HTML是由一套标记标签(Markup Tag)组成的,通常叫做标签.标签一般由开始标签和结束标签组成.部分标签为单标签或者空标签,即没有结束标签 开始标签 结束标签 单标签或者空标签

#### 文本标签

##### 1. 段落标签

段落标签使用p标签, 是paragraph的缩写, 自带换行效果。如:

```
<p>段落标签使用p表示,是paragraph的缩写</p>
<p>我爱高圆圆,虽然她已经39岁了</p>
```

2. **标题标签h1 ~ h6** 标题标签从名字就能看出,是用来定义文字标题的,包括h1-h6,数字越小对应的字体越大。如下:

```
<h1>我是一级标题</h1>
<h2>我是二级标题</h2>
...
<h6>我是老六</h6>
```

## 表单标签

1. **文本框** 单行的文本使用  标签, input标签有很多属性, 如下:

- type, 表示文本的类型;
- name, 表示文本的名称,后端使用name来获取框中的属性值;
- value, 默认属性框的填充值, 用户输入后显示输入的内容
- placeholder, 文本框内容为空显示的内容;
- size, 文本框的长度

```
<input type="text" name="name" value="高圆圆" placeholder="请输入姓名" size="10">
```

2. **密码框** 密码框与文本框类似, 区别在于type取值不同, 取值为password, 输入时候自动显示为星号,如:

```
<input type="password" name="password" placeholder="请填写密码">
```

3. **普通按钮**  表示一般按钮, 如下:

```
普通按钮<input type="button" value="登录">
```

4. **提交按钮** 当需要提交表单的时候, 需要使用提交按钮。提交按钮需要配合form表单才能向服务器提交数据。如下:

```
提交按钮<input type="submit" value="提交">
```

5. **表单** 表单使用form来表示,表示提交到的服务器的信息,一般在注册或者登陆两个应用场景中使用 如下:

```
<form method="post" action="http://how2j.cn/study/login.jsp">
账号: <input type="text" name="name"> <br/>
密码: <input type="password" name="password" > <br/>
<input type="submit" value="登陆">
</form>
```

action表示将表单提交的后台地址,method="post"表示提交的方式,提交方式有get方式和post方式两种 get和post方式区别 get方式将信息提交到url的后面 提交大小有限制 不适合私有数据 post方式信息是提交实体内容 提交大小无限制 适合私有数据

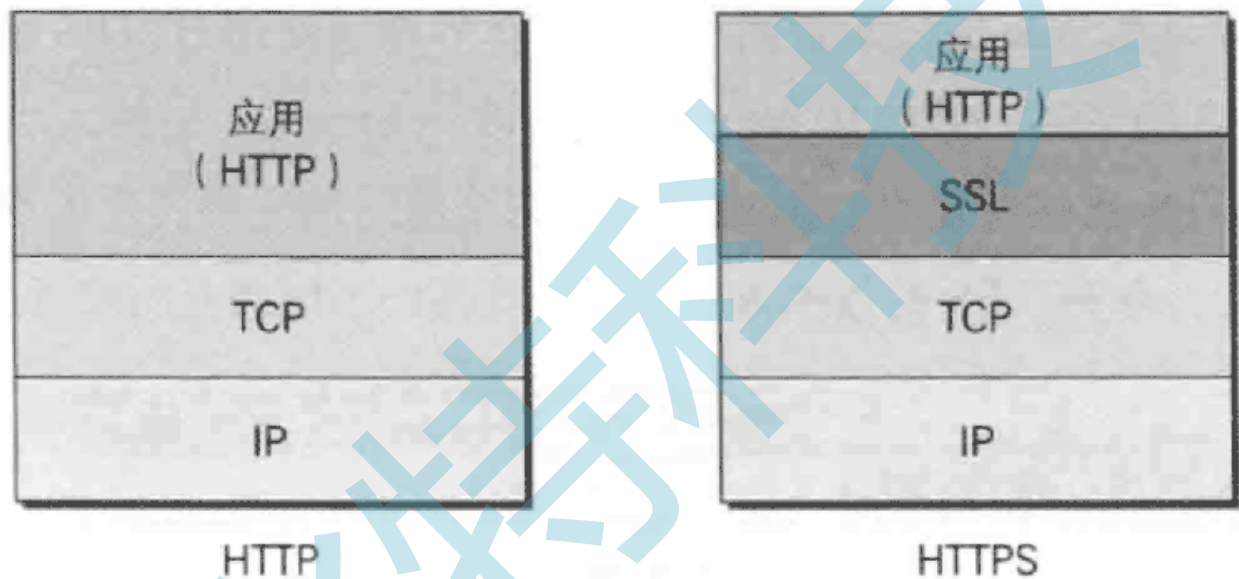
## 5.4 form http上提交, 看格式

表单提交的抓包截图, GET和POST区别

## 6. 补充主题

### 6.1. Http VS Https

简单理解,http传送数据(包括账号和密码),都是明文传送,很容易被窃取或者侦听,在现有的互联网应用中,很明显有不安全因素,所以有了https,可以简单理解成https多了一层加密解密层,在发送前加密,在收到后解密,在网络里传输的都是经过加密的数据



详细介绍: <https://blog.csdn.net/xiaoming100001/article/details/81109617>

<https://www.cnblogs.com/Rawls/p/10725703.html>

### 6.2. Http 新特性

- http2.0
- http3.0

### 6.3. 市场上常见的 Http 服务器

- Tomcat(web 应用服务器)
- Nginx
- Apache
- Lighttpd

## 7. Http Server + HTML + session&cookie



```
public class Request {

    /**
     * 请求方法
     */
    private String method;

    /**
     * 请求路径
     */
    private String uri;

    /**
     * Http版本
     */
    private String version;

    /**
     * 请求头
     */
    private Map<String, String> headers = new HashMap<>();

    /**
     * 请求参数
     */
    private Map<String, String> parameters = new HashMap<>();

    private Map<String, String> cookie = new HashMap<>();

    /**
     * 统一编码
     */
    private static final String CHARSET = "UTF-8";

    private Request() {

    }

    /**
     * 解析HttpRequest并生成请求对象
     *
     * @param is HttpRequest流
     * @return 请求对象
     * @throws IOException
     */
    public static Request build(InputStream is) throws IOException {
        Request request = new Request();
        BufferedReader reader = new BufferedReader(new InputStreamReader(is, CHARSET));
        String line = reader.readLine();
        System.out.println("请求行:" + line);
        // 处理请求行
        if (line != null) {
            String[] requestLineArray = line.split(" ");
        }
    }
}
```

```

        request.setMethod(requestLineArray[0]);
        request.setUri(requestLineArray[1]);
        request.setVersion(requestLineArray[2]);
    }

    // 处理请求头
    while ((line = reader.readLine()) != null && line.length() != 0) {
        System.out.println("请求头: "+line);
        String[] headerArray = line.split(":");
        request.addHeader(headerArray[0].trim(), headerArray[1].trim());
        if("Cookie".equalsIgnoreCase(headerArray[0].trim())){
            String[] cookies = headerArray[1].trim().split(";");
            for(String cookie : cookies){
                String[] cookieArray = cookie.split("=");
                request.cookie.put(cookieArray[0].trim(), cookieArray[1].trim());
            }
        }
    }
    // 处理请求体
    if ("POST".equals(request.getMethod())) {
        int len = Integer.parseInt(request.getHeader("Content-Length"));
        char[] chars = new char[len];
        reader.read(chars, 0, len);
        System.out.println("请求体: "+String.valueOf(chars));
        for (String requestParam : String.valueOf(chars).split("&")) {
            String[] paramArray = requestParam.split("=");
            request.addParameter(paramArray[0], paramArray[1]);
        }
    }
    return request;
}

public void addHeader(String key, String value) {
    headers.put(key, value);
}

public String getHeader(String key) {
    return headers.get(key);
}

public void addParameter(String key, String value) {
    parameters.put(key, value);
}

public String getParameter(String key) {
    return parameters.get(key);
}

public String getCookie(String key){

    return cookie.get(key);
}

```

```
@Override
public String toString() {
    return "HttpRequest{" +
        "method='" + method + '\'' +
        ", uri='" + uri + '\'' +
        ", version='" + version + '\'' +
        ", headers=" + headers +
        ", parameters=" + parameters +
        '}';
}

public String getMethod() {
    return method;
}

public void setMethod(String method) {
    this.method = method;
}

public String getUri() {
    return uri;
}

public void setUri(String uri) {
    this.uri = uri;
}

public String getVersion() {
    return version;
}

public void setVersion(String version) {
    this.version = version;
}

public Map<String, String> getHeaders() {
    return headers;
}

public void setHeaders(Map<String, String> headers) {
    this.headers = headers;
}

public Map<String, String> getParameters() {
    return parameters;
}

public void setParameters(Map<String, String> parameters) {
    this.parameters = parameters;
}

public Map<String, String> getCookie() {
```

```

        return cookie;
    }

    public void setCookie(Map<String, String> cookie) {
        this.cookie = cookie;
    }
}

public class Response {

    /**
     * 响应状态码
     */
    private int status;

    /**
     * 响应信息
     */
    private String message;

    /**
     * 响应头
     */
    private Map<String, String> headers = new HashMap<>();

    /**
     * 响应打印流
     */
    private PrintWriter writer;

    /**
     * 统一编码
     */
    private static final String CHARSET = "UTF-8";

    /**
     * 响应内容
     */
    private StringBuilder body = new StringBuilder();

    private Response(){

    }

    public static Response build(OutputStream os) throws UnsupportedEncodingException {
        Response response = new Response();
        response.writer = new PrintWriter(new OutputStreamWriter(
            os, CHARSET), true);
        return response;
    }

    /**
     * 返回信息

```

```
* @param content
*/
public Response println(String content){
    body.append(content+"\r\n");
    return this;
}

public void flush() throws IOException {
    writer.println("HTTP/1.1 "+status+" "+message);
    writer.println("Content-Type: text/html; charset=utf-8");
    writer.println("Content-Length: "+body.toString().getBytes(CHARSET).length);
    headers.forEach((k,v)->{
        writer.println(k+": "+v);
    });
    writer.println();
    writer.println(body.toString());
}

public void addHeader(String key, String value) {
    headers.put(key, value);
}

public int getStatus() {
    return status;
}

public void setStatus(int status) {
    this.status = status;
}

public String getMessage() {
    return message;
}

public void setMessage(String message) {
    this.message = message;
}

public Map<String, String> getHeaders() {
    return headers;
}

public void setHeaders(Map<String, String> headers) {
    this.headers = headers;
}

public Writer getWriter() {
    return writer;
}

public void setWriter(PrintWriter writer) {
    this.writer = writer;
}
```

```

}

public class Server {

    /**
     * 端口号
     */
    private static final int PORT = 9999;

    private static final Map<String, Object> SESSION = new HashMap<>();

    public static void main(String[] args) throws IOException {
        ServerSocket serverSocket = new ServerSocket(PORT);
        ExecutorService POOL = Executors.newCachedThreadPool();
        try {
            while (true) {
                Socket socket = serverSocket.accept();
                POOL.submit(new Runnable() {
                    @Override
                    public void run() {
                        try {
                            Request request = Request.build(socket.getInputStream());
                            Response response = Response.build(socket.getOutputStream());

                            if("/307".equals(request.getUri())) {
                                response.setStatus(307);
                                response.setMessage("Temporary Redirect");
                                response.addHeader("Location", "http://45.40.254.164");
                                response.println("");
                            } else if("/setCookie".equals(request.getUri())) {
                                response.setStatus(200);
                                response.setMessage("OK");
                                response.addHeader("Set-Cookie", "studentid=" +
UUID.randomUUID());
                                response.println("<h1>Set Cookie OK</h1>");
                            } else if("GET".equals(request.getMethod()) &&
"/login.html".equals(request.getUri())) {
                                response.setStatus(200);
                                response.setMessage("OK");
                                BufferedReader reader = new BufferedReader(
                                    new InputStreamReader(
Server.class.getClassLoader().getResourceAsStream("login.html")));
                                String line;
                                while ((line = reader.readLine()) != null) {
                                    response.println(line);
                                }
                            } else if("POST".equals(request.getMethod()) &&
"/login".equals(request.getUri())) {
                                response.setStatus(200);
                                response.setMessage("OK");
                                response.println("<h2>欢迎您, 用
户"+request.getParameter("username")+"</h2>");

```



```

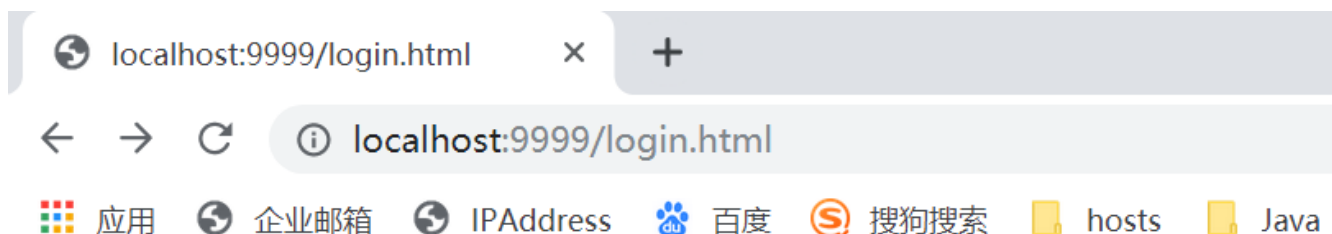
        String sessionId = UUID.randomUUID().toString();
        response.addHeader("Set-Cookie", "JSESSIONID="+sessionId);
        SESSION.put(sessionId, "用户名: "+request.getParameter("username")+", 计算机系2020届学生");
    }else if("/getSession".equals(request.getUri())){
        response.setStatus(200);
        response.setMessage("OK");
        response.println("<h1>"+SESSION.get(request.getCookie("JSESSIONID"))+"</h1>");
    }else{
        response.setStatus(404);
        response.setMessage("Not Found");
        response.println("<h1>没有找到资源</h1>");
    }
    response.flush();
    socket.close();
} catch (Exception e){
    e.printStackTrace();
}
});
}
} catch (Exception e){
    e.printStackTrace();
}
}
}

```

查看登录网页: <http://localhost:9999/login.html>



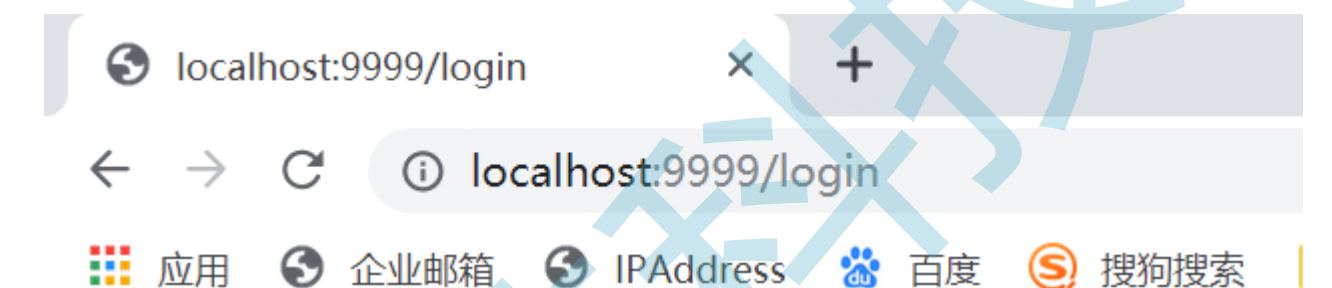
点击提交按钮:



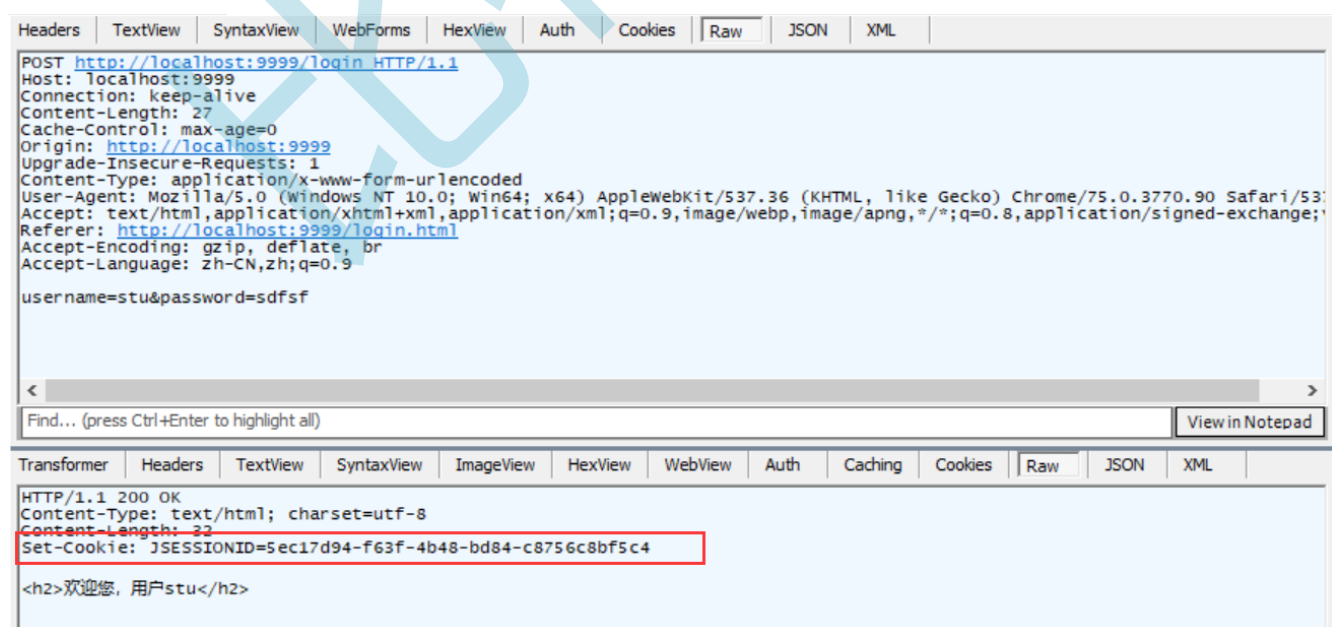
## 用户注册

姓名:  密码:

跳转成功:



## 欢迎您, 用户stu



此时设置了Session在服务器端，保存了用户信息，查看Session信息：

← → ↻ ⓘ localhost:9999/getSession

应用 企业邮箱 IPAddress 百度 搜狗搜索 百度网盘 百度翻译 hosts Java 工!

# 用户名：stu，计算机系2020届学生

HeadersTextViewSyntaxViewWebFormsHexViewAuthCookiesRawJSONXML

GET http://localhost:9999/getSession HTTP/1.1  
Host: localhost:9999  
Connection: keep-alive  
Upgrade-Insecure-Requests: 1  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.90 Safari/537.5  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.9  
Accept-Encoding: gzip, deflate, br  
Accept-Language: zh-CN,zh;q=0.9  
Cookie: JSESSIONID=Sec17d94-f63f-4b48-bd84-c8756c8bf5c4

< >

Find... (press Ctrl+Enter to highlight all) View in Notepad

TransformerHeadersTextViewSyntaxViewImageViewHexViewWebViewAuthCachingCookiesRawJSONXML

HTTP/1.1 200 OK  
Content-Type: text/html; charset=utf-8  
Content-Length: 54  
  
<h1>用户名：stu，计算机系2020届学生</h1>