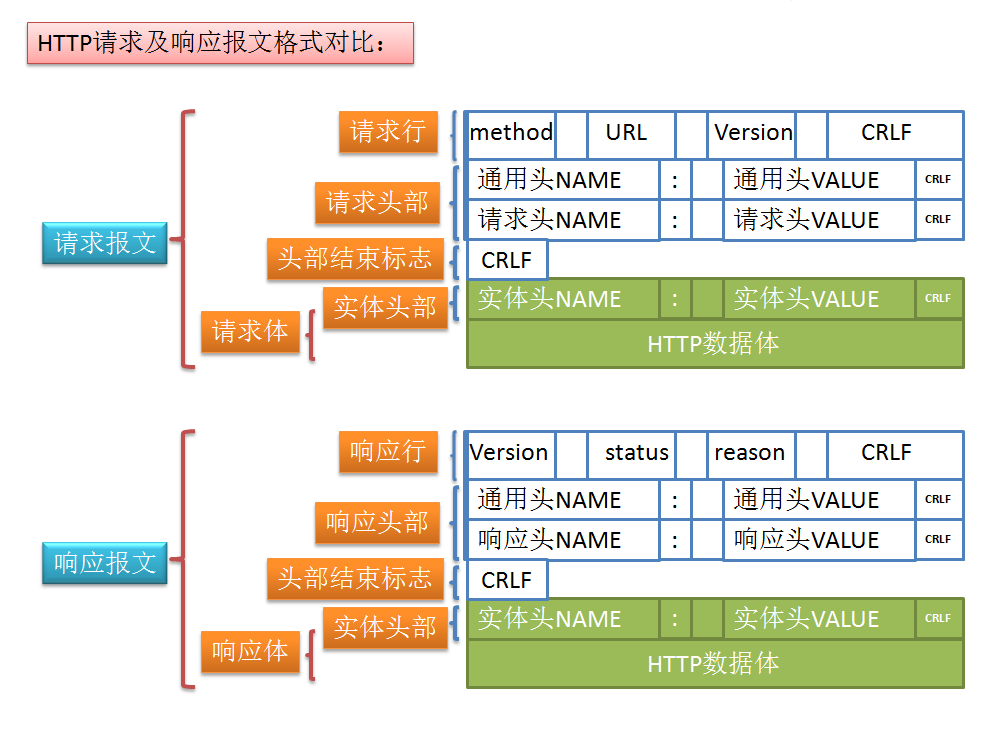
# 1.HTTP基本介绍



http协议版本:

http1.0:当浏览器和服务器连接之后,在一次连接中只能发出一个请求

http1.1:当浏览器和服务器连接之后,在一次连接中可以发出多次请求

URI:统一资源描述符和协议无关,可以描述任何文件,例如/web/hello

URL:统一资源定位符,前面要加上协议,例如<http://localhost:8080/web/hello>

GET:

1)提交的参数显示到地址栏

2)不适合放敏感数据

3)不超过1KB

原理:post提交的参数跟在请求的实体内容中

GET /day08/testMethod.html?name=jacky&password=123456 HTTP/1.1

POST:

1)提交的参数不会显示到地址栏

2)适合敏感数据

3)提交的大小没有限制

POST /day08/testMethod.html HTTP/1.1

...

name=jacky&password=123456//实体内容

# 2.HTTP请求

## 2.1HttpServletRequest接口

常用的方法:

请求行:

请求方式:req.getMethod()

请求资源:req.getRequestURI()/req.getRequest.URL

http协议版本:req.getProtocol()

请求头:

req.getHeader(“字段名”)

req.getHeaderNames()

实体内容:

req.getInputStream()

范例:

|  |
| --- |
| public class HelloServlet extends HttpServlet {  //1)tomcat服务器接受到一个浏览器发送的请求的数据  //2)tomcat服务器会把这些请求数据封装成一个HttpServletRequest对象  //3)tomcat服务器调用doGet方法,会把request对象传入servlet  protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  //doGet方法只能接受get提交的请求  //4)通过request对象获取请求数据  /\*\*  \* 请求数据  \*/  //4.1请求行  //4.1.1请求方式  System.out.println("请求方式:" + req.getMethod());  //4.1.2请求资源  System.out.println("URI:" + req.getRequestURI());  System.out.println("URL:" + req.getRequestURL());  //4.1.3qhttp协议版本号  System.out.println("请求协议" + req.getProtocol());  //4.2请求头  String value = req.getHeader("host");  System.out.println("host:" + value);  //这个Enumeration和Iterator差不多  Enumeration<String> enums = req.getHeaderNames();  while(enums.hasMoreElements()){  String headerName = enums.nextElement();  String headerValue = req.getHeader(headerName);  System.out.println(headerName + " : " + headerValue);  }  }  @Override  protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  //doPost接受post提交的请求  //4.3实体内容  //注意:post提交的数据才会出现在实体内容中  InputStream input = req.getInputStream();  byte buf[] = new byte[1024];  int len = 0;  while((len=input.read(buf))!=-1){  System.out.println(new String(buf,0,len));  }  }  } |

## 2.2.service方法

这个方法是HttpServlet里的,

service和doXXX方法的关系:

1. service方法是程序的入口
2. 在HttpService的service方法源码中,根据不同请求方式调用不同doXXX方法,

所以在开发servlet的时候,就不需要去覆盖service方法,只需要覆盖doXXX方法

## 2.3.字段

### 2.3.1.user-agent

**chrome:**

Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/537.36

**firefox:**

Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:56.0) Gecko/20100101 Firefox/56.0

**Edge:**

Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36 Edge/15.15063

范例:

|  |
| --- |
| protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  String userAgent = request.getHeader("user-agent");  response.setContentType("text/html;charset=utf-8");  if(userAgent.contains("Firefox")){  response.getWriter().write("你正在使用狐火浏览器");  }else if(userAgent.contains("Edge")){  response.getWriter().write("你正在使用edge浏览器");  }else{  response.getWriter().write("你正在使用谷歌浏览器");  }  } |

### 2.3.2referer

referer:表示当前请求来自哪里

用途:防止非法连接

注意:referer只会通过超链接(即a标签)才能产生这个字段

范例:

|  |
| --- |
| protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  response.setContentType("text/html;charset=utf-8");  String referer = request.getHeader("referer");  if(referer==null || !referer.contains("/adv.html")){  response.getWriter().write("你当前请求你是非请求,请转到首页<a href='/web/adv.html'>首页</a>");  }else {  response.getWriter().write("资源正在下载...");  }  } |

## 2.4.获取参数数据

get方式里:request. getQueryString()能返回提交的参数

post:方式里:通过request.getInputStream()在通过read方法也能获取提交的数据,

但是以上要分两种,太麻烦了,不使用

通过一以下方法来获取参数更简单,get,post都适用:

不过,当form里的method为post, enctype="multipart/form-data”时,getParameter是获取不到数据的,这时,只能用getInputStream()了

request.**getParameter**(“name”):获取一个值的参数

request.**getParameterValue**(“name”):获取多个值的参数

request.**getParameterNames**():获取所有参数名称

request.**getParameterMap**():获取所有参数的对象

|  |
| --- |
| String name = request.getParameter("name");  System.out.println(name);  Enumeration<String> enums = request.getParameterNames();  while(enums.hasMoreElements()){  String paramName = enums.nextElement();  String paramValue = request.getParameter(paramName);  System.out.println(paramName + "=" + paramValue);  }  System.out.println("===========");  Map<String,String[]> map = request.getParameterMap();//获取参数对象集合(map)  Set<Map.Entry<String, String[]>> entrySet = map.entrySet();  for(Map.Entry<String,String[]> entry : entrySet){  String key = entry.getKey();  String[] value = entry.getValue();  System.out.println(key+"=" + value[0]);  }  String [] hobits = request.getParameterValues("hobit");  for(String s : hobits){  System.out.println(s + ",");  } |

# 3.编码问题

问题:通过html的form表单提交含有中文的参数时,在通过上述代码获取到值打印时是乱码

原因推理:tomcat将获取的参数用iso8859-1进行编码,默认解码是根据ide的编码来进行解码的,所以是utf-8

解决方法:

1. request.setCharacterEncoding("utf-8");

注意:这个方法只对post提交的参数有用,因为这个方法只能设置对请求实体内容的解码

时查询的码表,post提交的参数是放在实体内容中,所以对post有效,对get无效

2)先将获取到的字符串按照iso-8859-1编码为字节码,再用utf-8进行解码

new String(str.getBytes("iso-8859-1"),"utf-8")

# 4.HTTP响应

## 4.1状态码

200: 请求处理完成，成功返回

302:需要浏览器进一步,才能完成

404:浏览器端的错误,没有找到资源

500:服务器端的错误

## 4.2HttpServletResponse类

1.tocmat服务器提供了一个HttpServletResponse对象,用户给开发者修改响应数据

2.通过service方法把response对象传入servlet中

3.通过response对象修改响应数据

4.tomcat服务器把response对象转化为响应格式的字符串,发送给浏览器

response.setStaus(404) 设置状态码

response.setHeader(“name”,”value”) 修改响应头

response.getWriter().write() 以字符形式发送实体内容

response.getOutputStream().write() 以字节形式发送实体内容

|  |
| --- |
| public class ServletDemo1 extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  response.setContentType("text/html;charset=utf-8");  //1.响应行  // response.setStatus(404);  // response.sendError(404);//404 + 404错误页面  //2.响应头  response.setHeader("server","webLogic");  //3.实体内容  response.getWriter().write("this is a content哈哈哈"); //字符流,发送文字  // response.getOutputStream().write("this is a content".getBytes());//字节流,如果发送图片等,应该发送字节流  }  } |

## 4.3字段

### 4.3.1location请求重定向

**原理**:浏览器拿到302状态码,会再次向服务器发出一个新的请求,请求的地址是location的地址

注意:一次请求重定向,一共向服务器发起了两次请求

范例:

|  |
| --- |
| public class ServletDemo1 extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  response.setContentType("text/html;charset=utf-8");  //原始版本  //设置302状态码  response.setStatus(302);  //设置location响应头  response.getWriter().write("hello,world");  response.setHeader("location","/web/01.html");  //简化本本,以后就用这个  response.sendRedirect("/web/01.html");  }  } |

### 4.3.2refresh定时刷新或每隔n秒跳转页面

|  |
| --- |
| public class ServletDemo1 extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  response.setContentType("text/html;charset=utf-8");  //设置秒数  // response.setHeader("refresh","2");  //每隔n秒跳转页面  response.getWriter().write("注册成功!3秒后跳转到主页");  //设置refresh  response.setHeader("refresh","3;/web/01.html");  }  } |

### 4.3.3content-type

服务器发送给浏览器的数据类型和数据编码格式

|  |
| --- |
| public class ServletDemo1 extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  //实体编码,默认编码为iso-8599-1  response.setCharacterEncoding("utf-8");  //指定浏览器使用什么样的编码进行解码  response.setHeader("content-type","text/html;charset=EUC-JP");  // response.setContentType("text/html;charset=utf-8"); //建议使用这个方法  response.getWriter().write("你好啊,hello,world,呃呃呃");  }  } |

### 4.3.4Content-Disposition

以下载方式打开资源attchment;filename=aaa.zip

|  |
| --- |
| public class ServletDemo1 extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  /\*\*  \* 告诉浏览器以下载方式去打开  \*/  File file = new File("F:\\共享目录\\server\\apache-tomcat-8.5.23\\apache-tomcat-8.5.23\\webapps\\web\\1.PNG");  response.setHeader("content-disposition","attachment;filename=" + file.getName());  //读取本地文件  InputStream in = new FileInputStream(file);  //写出给浏览器(字节内容)  OutputStream out = response.getOutputStream();  byte[] buf = new byte[1024];  int len = 0;  //边读边写  while ((len=in.read(buf))!=-1){  out.write(buf,0,len);  }  out.close();  in.close();  }  } |