

* 学习目标

* 能够编写下载文件案例

* Response:

```
* resp.setHeader("content-disposition", "attachment;filename="+
URLLEncoder.encode("美女.jpg","UTF-8"));
```

* 能够编写验证码案例

* BufferedImage

* 能够掌握validateCode验证码的框架

* ValidateCode

* 能够理解请求转发和重定向的区别

* 请求转发是一次请求

* 重定向两次请求

* 能够解决中文乱码问题

* Request

* Response

```
* resp.setContentType("text/html;charset=utf-8")
```

```
req.setCharacterEncoding("UTF-8");
```

* 能够理解会话的概念

* Cookie , Session

* 能够掌握Cookie技术

* 写一个Cookie客户端，在服务能获得Cookie

* 回顾

* ServletContext

* 属性：setAttribute , getAttribute , xxx

* 参数：全局配置参数

* 获得请求转发器，进行转发

- * 通过相对路径获得绝对路径，获得相对输入流，根据文件名获得MIME类型， ...

- * 登录的案例

- * MVC构建和Web三层架构

- * bean (entity) ,dao,service(bussiness),controller,view

- * Lombok

- * Request

- * ServletRequest

- * 属性：setAttribute , getAttribute , xxx

- * 参数：getParameter(name),getParameterNames(),getParameterValues(name)

- getParameterMap--》 Map<String,String[]>

- * 名字和类属性名字对应，MBeanUtils

- * 获得请求转发器

- * 获得远程地址，端口，主机

- * 获得请求输入流,....

- * HttpServletRequest

- *

- getHeader,getMethod,getRequestUrl,getRequestUri,getContextPath,getQueryString,

- getCookie,getSession

- * Response

- * ServletResponse

- * 设置编码：setCharacterEncoding,setContentType("text/html;charset='UTF-8'");

- * 获得输出流:getOutputStream,getWriter

- * HttpServletResponse

- * 响应Cookie：addCookie

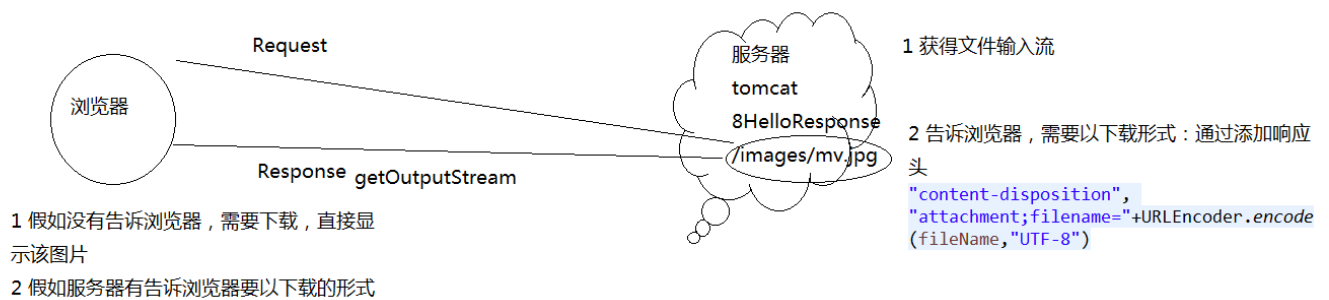
- * 编码url:encodeUrl

- * 重定向：sendRedirect()

- * 响应头：addHeader , setHeader

- * 状态码:setStatus():200,404,503

* 能够编写下载文件案例



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>下载</title>
6 </head>
7 <body>
8     <form action="/lgweb/download" method="post">
9         <input type="text" name="filename" value="banner.jpg"/>
10        <input type="submit" value="下载"/>
11    </form>
12 </body>
13 </html>
14
15 @WebServlet("/download")
16 public class DownServlet extends HttpServlet {
17     @Override
18     protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
19         String filename = req.getParameter("filename");
20         String path="/image/"+filename;
21         resp.setHeader("content-disposition", "attachment;filename="+ URLEncode
22         InputStream is = req.getServletContext().getResourceAsStream(path);
23         ServletOutputStream os = resp.getOutputStream();
24         byte[] b=new byte[1024*8];
25         int len=-1;
26         while((len=is.read(b))!=-1){
```

```

27         os.write(b,0,len);
28     }
29
30 }
31
32 @Override
33 protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
34     doGet(req, resp);
35 }
36 }
37

```

* 能够编写验证码案例

* 思路：

* 输出一个图片，在内存中构建一张图片（自己绘制）

* API

* BufferedImage:内存中图片（width，height，图片形式）

* Graphics：画笔

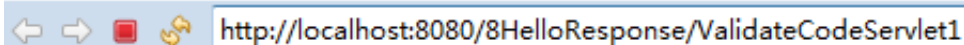
* 开发细节

* 构建内存图片（width，height，图片形式），并且输出到浏览器

```

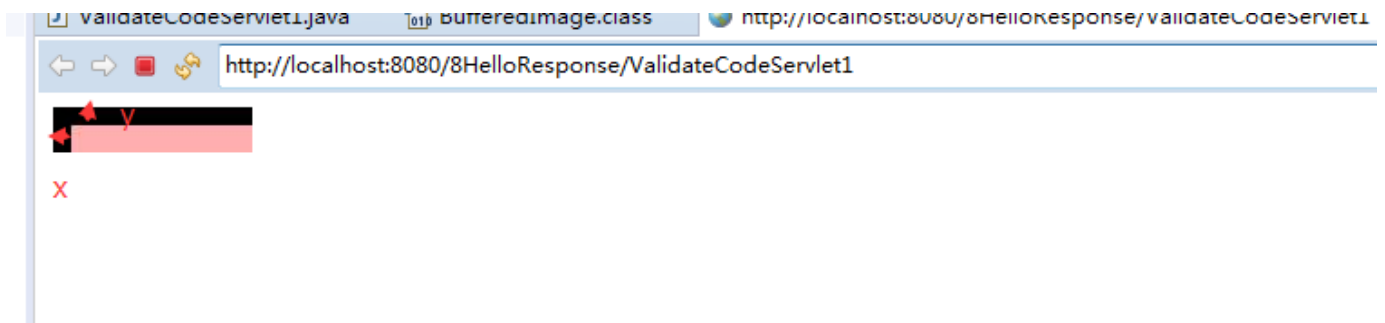
1 // 1 构建内存图片（画板）
2     int width=110;
3     int height=25;
4     BufferedImage image=new BufferedImage(width, height, BufferedImage.TYPE
5 //2 获取画笔
6     Graphics g = image.getGraphics();
7     ImageIO.write(image, "jpg", response.getOutputStream());

```

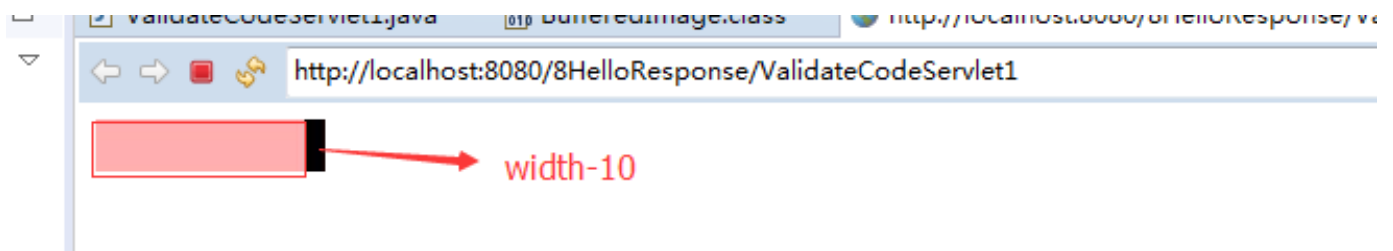



* 设置背景：参数测试（x，y，width，height）

```
1 //3 通过画笔设置背景的颜色
2     g.setColor(Color.PINK);
3     g.fillRect(10, 10, width, height);
```

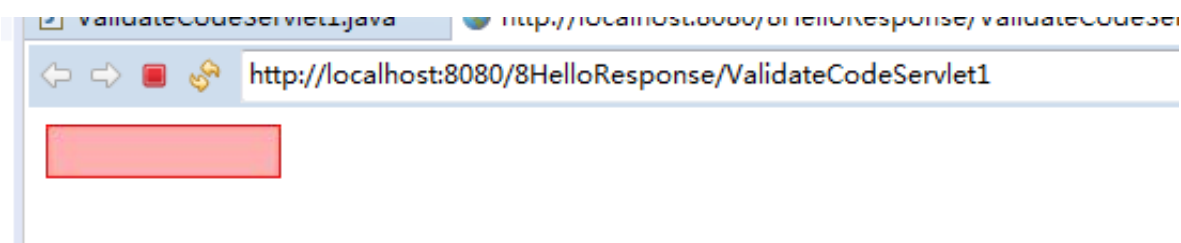


```
1 g.fillRect(0, 0, width-10, height);
```



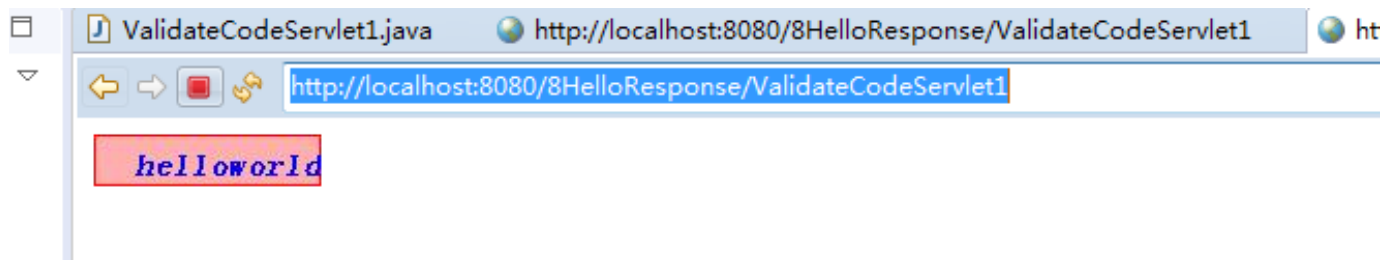
* 设置边框

```
1 //3 通过画笔设置背景的颜色
2     g.setColor(Color.PINK);
3     g.fillRect(1, 1, width-1, height-1);
4 //4 边框
5     g.setColor(Color.RED);
6     g.drawRect(0, 0, width-1, height-1);
```



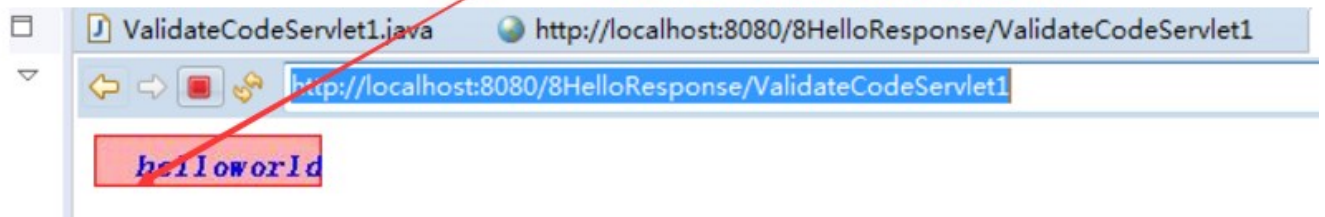
* 设置字体，画文本

```
1 g.setColor(Color.BLUE);
2 g.setFont(new Font("宋体", Font.BOLD|Font.ITALIC, 15));
3 g.drawString("helloworld", 20, 20);
```



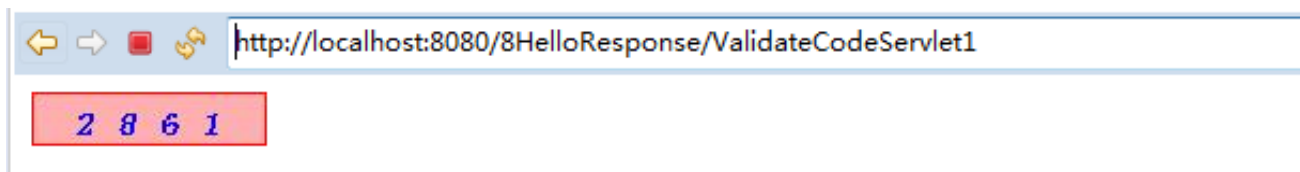
```
1 g.setColor(Color.BLUE);
2 g.setFont(new Font("宋体", Font.BOLD|Font.ITALIC, 15));
3 g.drawString("helloworld", 20, 20);
```

注意位置：假如设置为0的话，看到字体



* 绘制随机验证码：

```
1 Random r = new Random();
2     int xPos=20;
3     for (int i = 0; i < 4; i++) {
4         g.drawString(r.nextInt(10)+"", xPos, 20);
5         xPos+=20;
6     }
```



* 细节设置

```
1 response.setHeader("refresh", "5");//设置refresh响应头控制浏览器每隔5秒钟刷新
2 // 2.设置响应头控制浏览器浏览器以图片的方式打开
3 response.setContentType("image/jpeg");// 等同于response.setHeader("Content-Type","image/jpeg")
4 // 3.设置响应头控制浏览器不缓存图片数据
5 response.setDateHeader("expries", -1);
6 response.setHeader("Cache-Control", "no-cache");
7 response.setHeader("Pragma", "no-cache");
```

* 能够掌握ValidateCode验证码的框架

```
1 * 引入ValidateCode框架
2 * 代码
3 @WebServlet("/validateCode")
4 public class ValidateCodeServlet extends HttpServlet {
5     @Override
6     protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
7         int width=150;
8         int height=30;
9         int codeCount=5;
10        int lineCount=5;//干扰线
11        ValidateCode validateCode=new ValidateCode(width,height,codeCount,lineCount);
12        String code = validateCode.getCode();
13        System.out.println("code = " + code);
14        validateCode.write(resp.getOutputStream());
15    }
16
17    @Override
18    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
19        doGet(req, resp);
20    }
```

21 }

22

会员登录

您还不是我们的会员？[立即注册](#)

用户名:

密 码:

验证码: 

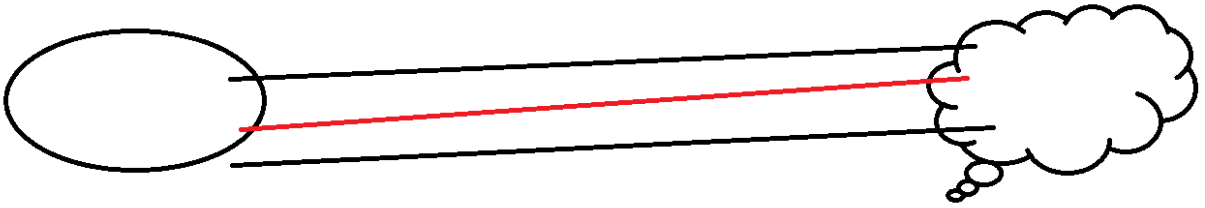
[立即登录](#)

* 能够理解请求转发和重定向的区别

* 重定向：

* 一个web资源收到客户端请求后，通知客户端去访问另外一个web资源，这称之为请求重定向。


重定向 多次请求



```
1 @WebServlet("/test14")
2 public class TestServlet14 extends HttpServlet {
3     @Override
4     protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
5         //
6         System.out.println("老孙到此一游");
7     //     //302:告诉浏览器: 访问这个位置是暂时, 它会访问到另一个位置(302)
8     //     resp.setStatus(HttpServletResponse.SC_FOUND);
9     //     resp.setHeader("Location", "/lgweb/login.html");
10    resp.sendRedirect("/lgweb/login.html");
11 }
```

← → ↻ ⓘ localhost:8080/lgweb/login.html

应用 开服表 【亮哥】Android ... 网址安全导航_安全... Maven Repository... 微信公众平台 消息中心 - 哔哩哔哩... 菜鸟教程 - 学的

 小米商城
让每个人都能享受科技的乐趣

会员登录 您还不是我们

Elements Console Sources Network Performance Memory Security Application Audits

Filter ☐ Hide data URLs ☒ All XHR JS CSS Img Media Font Doc WS Manifest Other

Name

- test14
- login.html
- mistore_logo.png
- validateCode
- ghs.png
- login_bg.jpg

test14

General

Request URL: http://localhost:8080/lgweb/test14

Request Method: GET

Status Code: 302 Found

Remote Address: [::1]:8080

Referrer Policy: no-referrer-when-downgrade

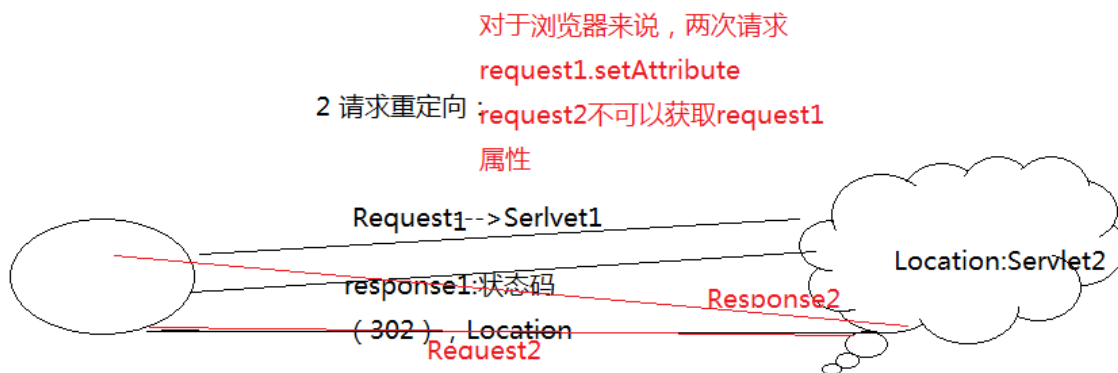
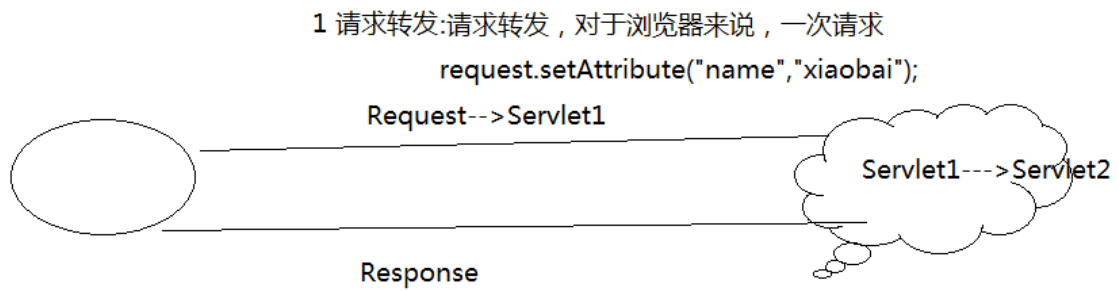
Response Headers view source

Content-Length: 0

Date: Thu, 21 Nov 2019 02:51:16 GMT

6 requests | 902 B transferred | 83.1 KB resources | Finish: 100 ms | DOMCont

* 对比请求转发和重定向的区别



* 简单案例

```
1 * Servlet A
2 req.setAttribute("name","xiaohei");
3 //      req.getRequestDispatcher("/code").forward(req,resp);
4 resp.sendRedirect("/lgweb/code");
5
6
7 * Servlet B
8 Object obj = req.getAttribute("name");
9     if(obj==null){
10         System.out.println("获取不到数据");
11     }else{
12         System.out.println(obj);
13     }
```

* 能够解决中文乱码问题

1 测试响应

问题：

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException {  
    response.getWriter().append("中国");  
}
```

http://localhost:8080/8HelloResponse/CodeServlet1

??

响应回来的时候，存在乱码

解决方案：

```
/**  
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)  
 */  
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException {  
    // 第一种设置编码方式  
    // 设置MIME类型  
    response.setContentType("text/html");  
    // 设置响应编码  
    response.setCharacterEncoding("UTF-8");  
  
    // 第二种设置编码方式  
    response.setContentType("text/html; charset=utf-8");  
    response.getWriter().append("中国");  
}
```

```
1 // 第一种设置编码方式  
2 // 设置MIME类型  
3 response.setContentType("text/html");  
4 // 设置响应编码  
5 response.setCharacterEncoding("UTF-8");  
6  
7 // 第二种设置编码方式  
8 response.setContentType("text/html; charset=utf-8");  
9 response.getWriter().append("中国");
```

http://localhost:8080/8HelloResponse/CodeServlet1

中国

2 测试获取Get方式请求参数：

1 在tomcat8(Servlet3.0)里面：默认编码UTF-8，以前是ISO-8859-1（当Servlet版本是3.0的时候，就不用处理get乱码）

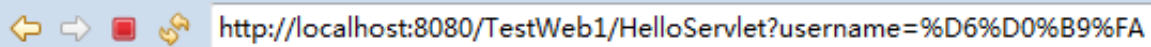
```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Insert title here</title>
5 </head>
6 <body>
7     <form action="/8HelloResponse/CodeServlet1" method="get">
8         <input type="text" value="中国" name="username"/><br>
9         <input type="submit" value="测试中文乱码"/>
10    </form>
11 </body>
12 </html>
```

```
1 protected void doGet(HttpServletRequest request, HttpServletResponse response)
2     throws ServletException, IOException {
3     response.setContentType("text/html;charset=utf-8");
4     // 请求方式为get的处理方式
5     String method = request.getMethod();
6     String name = "";
7     if ("GET".equals(method)) {
8         name = request.getParameter("username");
9         System.out.println(name);
10    }
11    response.getWriter().append(name);
12 }
13
```

```

protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=utf-8");
    // 请求方式为get的处理方式
    String method = request.getMethod();
    String name = "";
    if ("GET".equals(method)) {
        // 默认以前是ISO-8859-1, tomcat8: utf-8
        // 在tomcat7还是存在乱码ISO-8859-1
        name = request.getParameter("username");
        System.out.println(name);
    }
    response.getWriter().append(name);
}

```



http://localhost:8080/TestWeb1/HelloServlet?username=%D6%D0%B9%FA

中国

2 在tomcat7(Servlet2.4) 是存在乱码：默认ISO-8859-1

```

1 protected void doGet(HttpServletRequest request, HttpServletResponse response)
2     throws ServletException, IOException {
3     response.setContentType("text/html;charset=utf-8");
4     // 请求方式为get的处理方式
5     String method = request.getMethod();
6     String name = "";
7     if ("GET".equals(method)) {
8         // 默认以前是ISO-8859-1, tomcat8: utf-8
9         // 在tomcat7还是存在乱码ISO-8859-1
10        name = request.getParameter("username");
11        name=new String(name.getBytes("ISO-8859-1"), "gbk");
12        System.out.println(name);
13    }
14    response.getWriter().append(name);
15 }

```

```

protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=utf-8");
    // 请求方式位get的处理方式
    String method = request.getMethod();
    String name = "";
    if ("GET".equals(method)) {
        // 默认以前是ISO-8859-1, tomcat8: utf-8
        // 在tomcat7还是存在乱码ISO-8859-1
        name = request.getParameter("username");
        name=new String(name.getBytes("ISO-8859-1"),"gbk");
        System.out.println(name);
    }
    response.getWriter().append(name);
}

```

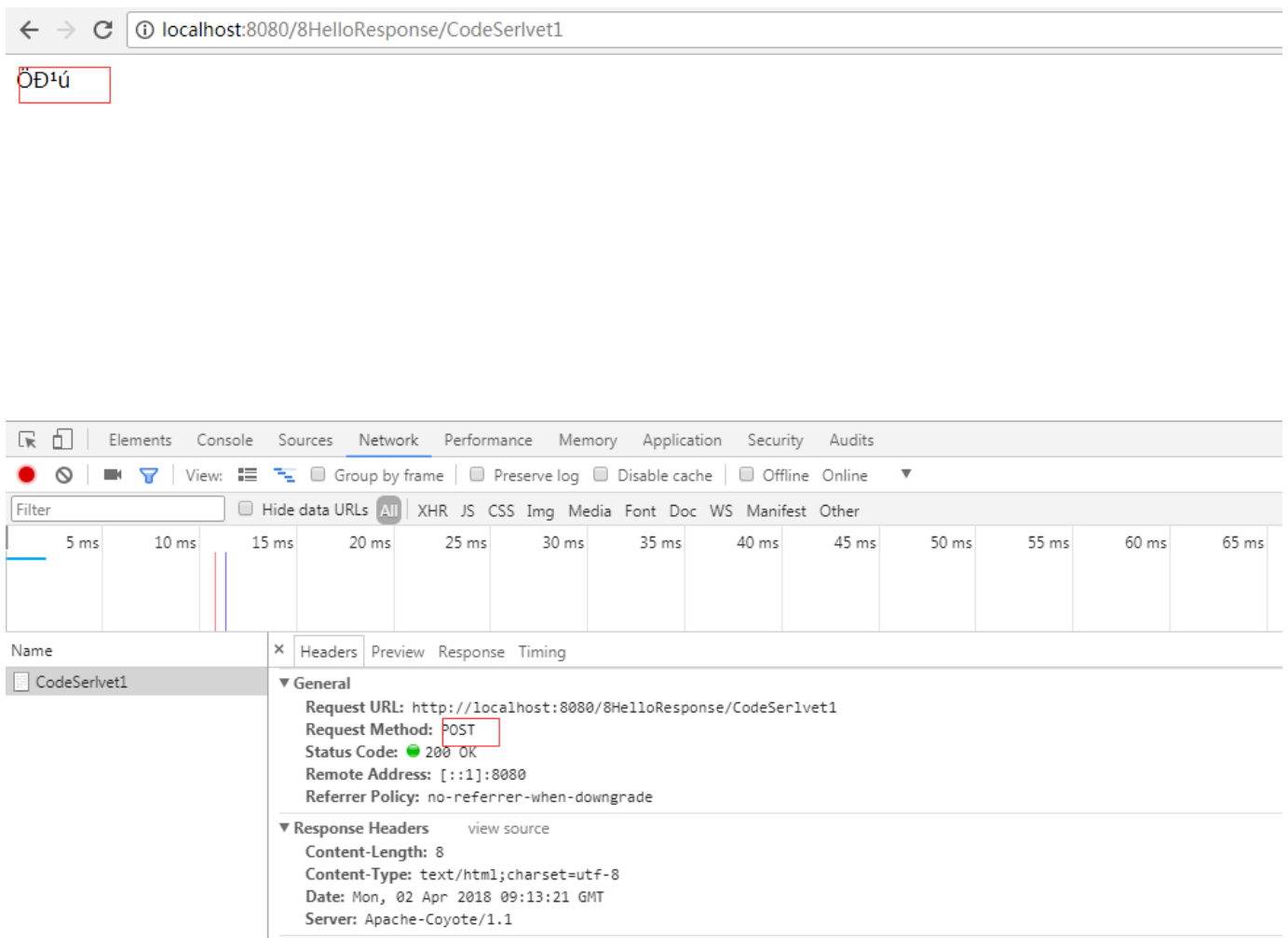
3 总结：不同Servlet版本，默认编码不一样：Servlet2.4和Servlet3.0

3 测试获取Post方式请求参数：

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Insert title here</title>
5 </head>
6 <body>
7     <form action="/8HelloResponse/CodeServlet1" method="post">
8         <input type="text" value="中国" name="username"/><br>
9         <input type="submit" value="测试中文乱码"/>
10    </form>
11 </body>
12 </html>

```



```
// response.setCharacterEncoding("UTF-8");
```

```
// 第二种设置编码方式
```

```
response.setContentType("text/html; charset=utf-8");
```

```
// 请求方式为get的处理方式
```

```
String method = request.getMethod();
```

```
String name = "";
```

```
if ("GET".equals(method)) {
```

```
    // 假如用Servlet3.1编译，不用做任何处理
```

```
} else if ("POST".equals(method)) {
```

```
    request.setCharacterEncoding("gbk");
```

```
    name = request.getParameter("username");
```

```
}
```

```
System.out.println(name);
```

```
response.getWriter().append(name);
```

```
}
```

post解决乱码的方式

```
1 protected void doGet(HttpServletRequest request, HttpServletResponse response)
2     throws ServletException, IOException {
3     // // 第一种设置编码方式
4     // // 设置MIME类型
5     // response.setContentType("text/html");
```

```
6      // // 设置响应编码
7      // response.setCharacterEncoding("UTF-8");
8
9      // 第二种设置编码方式
10     response.setContentType("text/html;charset=utf-8");
11
12     // 请求方式位get的处理方式
13     String method = request.getMethod();
14     String name = "";
15     if ("GET".equals(method)) {
16         // 假如用Servlet3.1 编译，不用做任何处理
17     } else if ("POST".equals(method)) {
18         request.setCharacterEncoding("gbk");
19         name = request.getParameter("username");
20     }
21     System.out.println(name);
22     response.getWriter().append(name);
23 }
24
```

← → ↻ ⓘ localhost:8080/8HelloResponse/CodeServlet1

中国

* 能够理解会话的概念

* 概述

* 会话可简单理解为：用户开一个浏览器，点击多个超链接，访问服务器多个web资源，然后关闭浏览器，整个过程称之为一个会话。有状态会话：一个同学来过教室，下次再来教室，我们会知道这个同学曾经来过，这称之为有状态会话。

* 每个用户在使用浏览器与服务器进行会话的过程中，不可避免各自会产生一些数据，程序要想办法为每个用户保存这些数据。

* 保存会话的技术

* 客户端Cookie

* Cookie是客户端技术，程序把每个用户的数据以cookie的形式写给用户各自的浏览器。当用户使用浏览器再去访问服务器中的web资源时，就会带着各自的数据去。这样，web资源处理的就是用户各自的数据了。

* 服务端Session

* Session是服务器端技术，利用这个技术，服务器在运行时可以为每一个用户的浏览器创建一个其独享的session对象，由于session为用户浏览器独享，所以用户在访问服务器的web资源时，可以把各自的数据放在各自的session中，当用户再去访问服务器中的其它web资源时，其它web资源再从用户各自的session中取出数据为用户服务。

* 购物车--Request.setAttribute,Session.setAttribute, ServletContext.setAttribute()

* 能够掌握Cookie技术



```
1 * 获取上一次的访问时间
2 @WebServlet("/lastAccess")
3 public class LastAccessServlet extends HttpServlet {
4
```

```
5  @Override
6  protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
7      resp.setContentType("text/html;charset=utf-8");
8      req.setCharacterEncoding("utf-8");
9      // 获得Cookie
10     Cookie[] cookies = req.getCookies();
11     boolean isFound=false;
12     Cookie c=null;
13     if(cookies!=null){
14         for (Cookie cookie : cookies) {
15             if("lastAccessTime".equals(cookie.getName())){
16                 c=cookie;
17                 isFound=true;
18                 break;
19             }
20         }
21         if(!isFound){
22             resp.getWriter().write("<h3>您是第一次访问该NB网站</h3>");
23         }else{
24             resp.getWriter().write("<h1>welcome come back</h1>: ");
25             String value = c.getValue();
26             Date date=new Date(Long.parseLong(value));
27             resp.getWriter().write(date.toLocaleString());
28         }
29     }else{
30         resp.getWriter().write("<h3>您是第一次访问该NB网站</h3>");
31     }
32     // 写个Cookie
33     //lastAccessTime=1574310365666;
34     // Version=1;
35     // Comment="hello cookie";
36     // \Domain=localhost;
37     // Max-Age=86400; E
38     // xpires=Fri, 22-Nov-2019 04:26:05 GMT;
39     // Path=/
40
41     Cookie cookie=new Cookie("lastAccessTime",System.currentTimeMillis()+"
42     cookie.setComment("hello cookie");
43     cookie.setDomain("localhost");
44     cookie.setPath("/");
```


```
45         cookie.setMaxAge(24*60*60);
46         resp.addCookie(cookie);
47     }
48
49     @Override
50     protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException {
51         doGet(req, resp);
52     }
53 }
54
```

← → ↻ ⓘ localhost:8080/lgweb/lastAccess

 应用  开服表  【亮哥】Android -...  网址安全导航_安全...  MVN |

您是第一次访问该NB网站

← → ↻ ⓘ localhost:8080/lgweb/lastAccess

 应用  开服表  【亮哥】Android -...  网址安全导航_安全...  MVN |

welcome come back

: 2019-11-21 12:37:49

→ ↻ ⓘ localhost:8080/lgweb/lastAccess

应用 开报表 【亮哥】Android -... 网址安全导航_支

elcome come back

19-11-21 12:37:49

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

Cookie

- Hm_lvt_080836300300be57b7f34f4b3e97d911
- Hm_lvt_aa5c701f4f646931bf78b6f40b234ef5
- Idea-f6e20ec0
- JSESSIONID
- lastAccessTime

名称	lastAccessTime
内容	1574311076517
域名	localhost
路径	/
为何发送	各种连接
创建时间	2019年11月21日星期四 下午12:37:56
到期时间	2019年11月22日星期五 下午12:37:56

禁止 删除 完成

* 登录案例（作业）

* 记住我，使用Cookie做





☒

记住我

登录

