

知识点列表

编号	名称	描述	级别
1	购物车	熟练掌握购物车的写法	**
2	URL 重写	理解并掌握 URL 重写的应用	**
3	什么是过滤器	了解过滤器	**
4	怎样写一个过滤器	掌握些过滤器的步骤	**
5	配置初始化参数	掌握配置初始化参数的方法	**
6	课堂练习	通过练习掌握过滤器	**
7	过滤器的优先级	了解	*
8	过滤器的优点	了解	**
9	如何写监听器	掌握些监听器的步骤	**
10	使用工具完成上传文件	了解文件上传功能的实现步骤	*

注： **"理解级别 ***"掌握级别 ****"应用级别

目录

1. 【案例 1】购物车 02*	错误！未定义书签。
1.1. 功能 **	3
1.2. 实现 **	3
2. 用户禁止 cookie 以后，如何继续使用 session**	10
2.1. 【案例 2】URL 重写 01 **	11
2.2. 【案例 3】URL 重写 02 **	12
3. 过滤器**	18
3.1. 什么是过滤器**	19
3.2. 怎样写一个过滤器**	20
3.3. 配置初始化参数**	27
3.4. 课堂练习**	27
3.5. 过滤器的优先级*	30
3.6. 过滤器的优点*	30
【案例 4】过滤器**	30
4. 监听器**	37
4.1. 如何写监听器**	38
4.2. 上传文件（扩展）*	55

1. 【案例 1】购物车 02 **

1.1. 功能 **

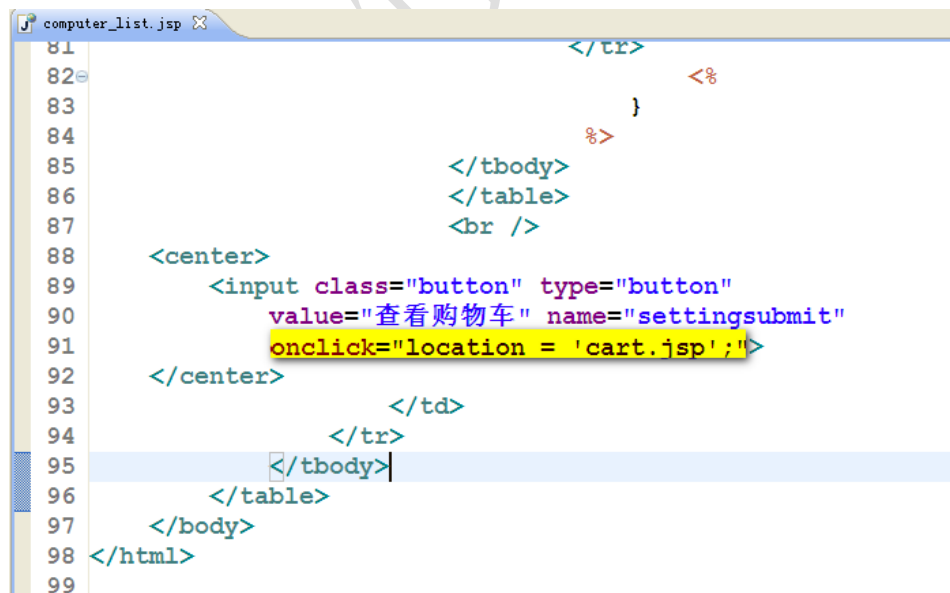
- a. 商品列表
- b. 购买商品
- c. 查看购物车
- d. 删除购物车当中的商品
- e. 修改购物车当中的商品的数量
- f. 删除购物车中的所有商品
- g. 购物车商品总价

1.2. 实现 **

功能 3：查看购物车 **

1) 修改 computer_list.jsp

添加查看购物车地址



```
81                                     </tr>
82                                     <%
83                                     }
84                                     %>
85                                     </tbody>
86                                     </table>
87                                     <br />
88                                     <center>
89                                     <input class="button" type="button"
90                                     value="查看购物车" name="settingsubmit"
91                                     onclick="location = 'cart.jsp';">
92                                     </center>
93                                     </td>
94                                     </tr>
95                                     </tbody>
96                                     </table>
97                                     </body>
98                                     </html>
99
```

2) 拷贝 cart.jsp 到项目中，修改为 cart.jsp

```
<%@page encoding="utf-8" contentType="text/html; charset=utf-8"%>
<%@page import="bean.*java.util.*" %>
<html>
    <head>
        <meta http-equiv=Content-Type content="text/html; charset=utf-8" />
        <link href="css/main/style.css"
            type="text/css" rel="stylesheet" />
    </head>

    <body topMargin="10">
        <div id="append_parent"></div>
        <table cellSpacing="6" cellPadding="2" width="100%" border="0">
            <tbody>
                <tr>
                    <td>

<table class="guide" cellSpacing="0"
cellPadding="0" width="100%"
border="0">
    <tbody>
        <tr>
            <td>
                <a href="#">主页</a>&nbsp;&nbsp;&nbsp;/&nbsp;&nbsp;&nbsp;
                <a href='computer_list.html'>
笔记本订购(WEB007)</a>&nbsp;&nbsp;&nbsp;/&nbsp;&nbsp;&nbsp;购物车信息
            </td>
        </tr>
    </tbody>
</table>

<br />

<table class="tableborder" cellSpacing="0" cellPadding="0"
width="100%" border="0">
    <tbody>
        <tr class="header">
            <td class="albtg2" colspan="6">
                购物车信息
```

```

        </td>
    </tr>
</tbody>
<tbody>
    <tr>
        <td class="altbg1" width="20%">
            <b>型号</b>
        </td>
        <td class="altbg1" width="20%">
            <b>价格</b>
        </td>
        <td class="altbg1" width="10%">
            <b>数量</b>
        </td>
        <td class="altbg1" width="30">
            &nbsp;
        </td>
        <td class="altbg1" width="10%">
            &nbsp;
        </td>
        <td class="altbg1">
            &nbsp;
        </td>
    </tr>
</tbody>

<tbody>
    <%
        Cart cart = (Cart)session.getAttribute("cart");
        if(cart != null && cart.list().size() > 0){
            List<CartItem> items = cart.list();
            for(int i=0;i<items.size();i++){
                CartItem item = items.get(i);
            }
        %>
    <tr>
        <td class="altbg2">
            <%=item.getC().getModel()%>
        </td>
    </tr>
</tbody>

```

```
<td class="altbg2">
    <%=item.getC().getPrice()%>
</td>
<td class="altbg2">
    <%=item.getQty()%>
</td>
<td class="altbg2">
    <input type="text" size="3" value=""
        id="num_<%=item.getC().getId()%>" />
</td>
<td class="altbg2">
    <a href="javascript:;"
        onclick="location='update.do?id=<%=item.getC().getId()%>
            &qty=' + document
            .getElementById(
            'num_<%=item.getC().getId()%>').value;'>更改数量</a>
</td>
<!--注意：请调整好加粗代码的格式，否则不起作用-->
<td class="altbg2">
    <a href="delete.do?id=<%=item.getC().getId()%>">删除</a>
</td>
</tr>

    <%
    }
    %>
<tr>
    <td class="altbg1" colspan="6">
        <b>总价格：¥<%=cart.total()%></b>
    </td>
</tr>
    <%
}
else{
    //还没有购买商品
    %>
    <tr>
        <td class="altbg2" colspan="6">
            <b>还没有选购商品</b>
        </td>
    </tr>
}
%>
</table>
```

```

        </tr>
        <%
        }
        %>
    </tbody>
</table>

    <br/>

<center>
    <input class="button" type="button" value="返回商品列表"
        name="settingsubmit" onclick="location = 'list.do';">
    <input class="button" type="button" value="清空购物车"
        name="settingsubmit"
        onclick="location = 'clear.do';">
</center>

        </td>
    </tr>
</tbody>
</table>

</body>
</html>

```

3) 测试

a. 访问 http://localhost:8080/web07_shopping/list.do

b. 点击“查看购物车”

跳转到 http://localhost:8080/web07_shopping/cart.jsp

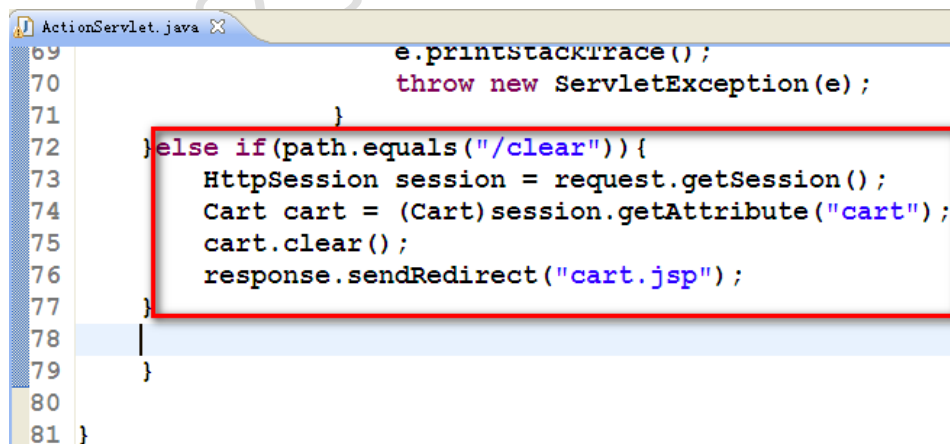


查看功能完成：)

功能 4：删除购物车中所有商品 **

4) 修改 ActionServlet

增加清空购物车业务



5) 重新部署，测试

购物车信息		
型号	价格	数量
x200	2000.0	1 <input type="text"/> 更改数量
总价格：¥ 2000.0		
<div> 返回商品列表 清空购物车 </div>		

功能 5：删除购物车中指定商品 **

6) 修改 ActionServlet 增加删除指定商品业务

```

71         }
72     }else if(path.equals("/clear")){
73         HttpSession session = request.getSession();
74         Cart cart = (Cart)session.getAttribute("cart");
75         cart.clear();
76         response.sendRedirect("cart.jsp");
77     }else if(path.equals("/delete")){
78         long id = Long.parseLong(request.getParameter("id"));
79         HttpSession session = request.getSession();
80         Cart cart = (Cart)session.getAttribute("cart");
81         cart.delete(id);
82         response.sendRedirect("cart.jsp");
83     }
84
85 }
86
87 }
    
```

7) 重新部署，测试

主页 / 笔记本订购(WEB007) / 购物车信息				
购物车信息				
型号	价格	数量		
x500	4000.0	1	<input type="text"/> 更改数量	删除
x600	6000.0	1	<input type="text"/> 更改数量	删除
总价格：¥ 10000.0				
<input type="button" value="返回商品列表"/> <input type="button" value="清空购物车"/>				

功能 6：修改购物车当中的商品的数量 **

8) 修改 ActionServlet

增加更新数量的代码

```

ActionServlet.java
77 }else if(path.equals("/delete")){
78     long id = Long.parseLong(request.getParameter("id"));
79     HttpSession session = request.getSession();
80     Cart cart = (Cart)session.getAttribute("cart");
81     cart.delete(id);
82     response.sendRedirect("cart.jsp");
83 }else if(path.equals("/update")){
84     long id = Long.parseLong(request.getParameter("id"));
85     int qty = Integer.parseInt(request.getParameter("qty"));
86     HttpSession session = request.getSession();
87     Cart cart = (Cart)session.getAttribute("cart");
88     cart.modify(id, qty);
89     response.sendRedirect("cart.jsp");
90 }
91
92 }
93
94 }
    
```

9) 重新部署，测试

购物车信息			
型号	价格	数量	
x200	2000.0	1	<input type="text"/> 更改数量
x500	4000.0	1	<input type="text"/> 更改数量
x600	6000.0	234	<input type="text"/> 更改数量
总价格：¥ 1410000.0			
<div> 返回商品列表 清空购物车 </div>			

更改数量功能完成：)

本版本存在的问题

当用户关闭浏览器，重新访问时，购物车中的信息就会丢失。

解决方法是使用 Session 技术，今天不做讲解。

(购物车案例结束)

2. 用户禁止 cookie 以后，如何继续使用 session **

1) 解决方式

使用 url 重写机制。

2) 什么是 url 重写。

如果要访问的 web 组件(jsp/servlet)需要 session 机制的支持,那么,不能够直接输入该 web 组件的地址,而应该使用服务器生成的包含有 sessionId 的地址。

3) 编程

如何生成包含有 sessionId 的地址?

方式 1 (适用于链接、表单提交)

```
response.encodeURL(String url);
```

方式 2 (适用于重定向)

```
response.encodeRedirectURL(String url);
```

注：转发不用

【案例 2】URL 重写 01 **

案例描述

当用户禁用 Cookie 后，使用 URL 重写**方式 1**（适用于链接、表单提交）
response.encodeURL(String url); 继续使用 session

参考代码

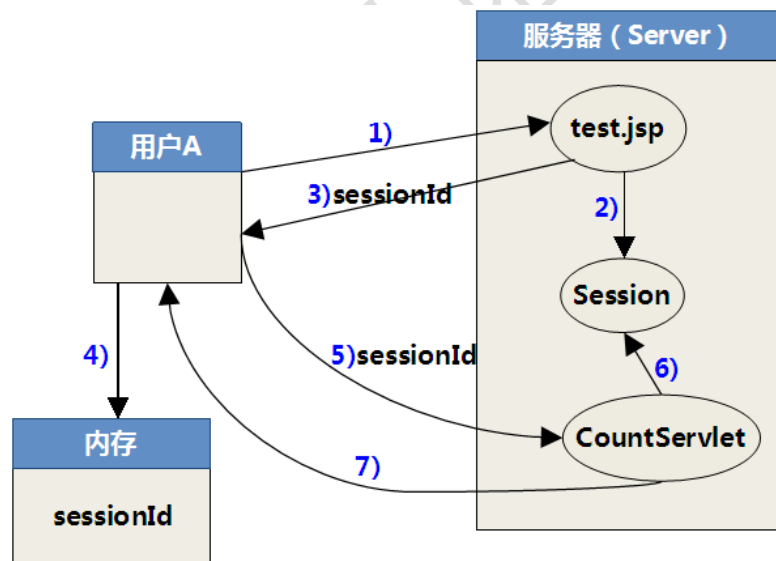
- 1) 请下载 web06_session.zip
- 2) 新建项目 web08_session，拷贝并部署

当 Cookie 没有被禁止时

- 3) 新建 test.jsp
- a. 访问 CountServlet

```
test.jsp x
1 <body style="font-size:30px;">
2   <a href="count">
3     visit countServlet</a>
4 </body>
5
```

- b. 访问 CountServlet 流程图（Cookie 被禁止时就会失效）



- ① 浏览器访问 test.jsp
- ② 服务器为该浏览器用户创建一个 Session 对象，用于保存此次会话的数据
- ③ 服务器将 sessionId 返回给浏览器并返回生成的显示给用户的页面
- ④ 浏览器将服务器传回的 sessionId 保存到内存中

- ⑤ 当用户点击“链接” `` 时，浏览器发送带 `sessionId` 的请求给服务器
- ⑥ 服务器中的 `CountServlet` 通过该 `sessionId` 找到该用户对应的 `Session` 并做计数操作
- ⑦ `CountServlet` 将计数结果和页面返回给用户浏览器

当 Cookie 被禁止时

方式 1 (适用于链接、表单提交)

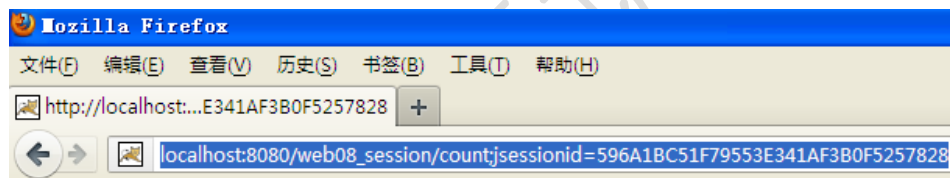
4) 修改 test.jsp

a. 使用 URL 重写方式，继续使用 session

```
test.jsp
1 <body style="font-size:30px;">
2   <a href="<%=response.encodeURL("count")%>">
3     visit countServlet</a>
4 </body>
5
```

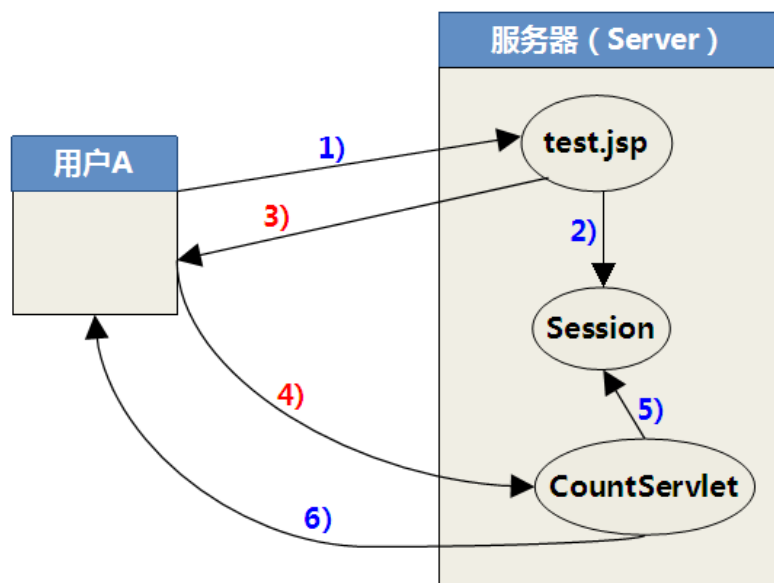
b. 点击链接，访问地址变为

`http://localhost:8080/web08_session/count;jsessionid=596A1BC51F79553E341AF3B0F5257828`



你是第 4 次访问

c. 访问 CountServlet 流程图 (Cookie 被禁止时 session 仍然起作用)



- ① 浏览器访问 test.jsp
- ② 服务器创建 Session 对象
- ③ test.jsp 将页面和 SessionId 返回给浏览器
因为浏览器禁用了 Cookie，所以浏览器并不保存 sessionId，
为了能继续使用 Session，我们在 test.jsp 中重写了 URL，
所以此时 test.jsp 返回给浏览器一个带有 sessionId 的链接地址：
`http://localhost:8080/web08_session/countjsessionid=596A1BC51F79553E341AF3B0F5257828`
- ④ 当用户点击“链接”，向服务器发送的请求中包含了 sessionId
- ⑤ 服务器通过这个 sessionId 可以找到对应的 Session
- ⑥ CountServlet 将计数结果和页面返回给用户浏览器

● form 表单的 URL 重写

```

1 <form action="<%=response.encodeURL("abc.do") %>">
2 </form>
    
```

【案例 3】URL 重写 02 **

案例描述

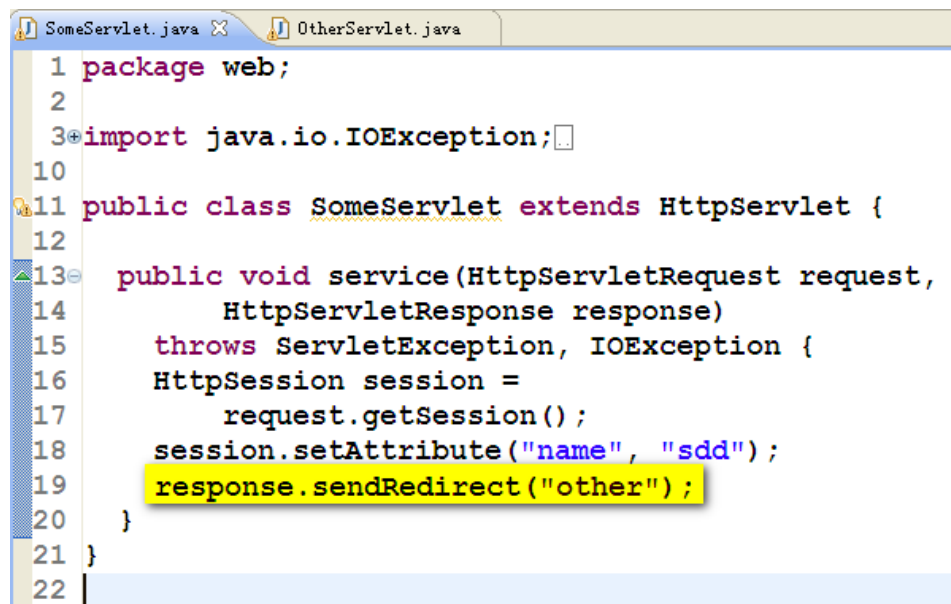
当用户禁用 Cookie 后，使用 URL 重写方式 2（适用于重定向）

`response.encodeRedirectURL(String url);` 使 Session 能够继续使用

注：转发不用

参考代码

1) 新建 SomeServlet



```
1 package web;
2
3 import java.io.IOException;
4
11 public class SomeServlet extends HttpServlet {
12
13     public void service(HttpServletRequest request,
14                         HttpServletResponse response)
15         throws ServletException, IOException {
16         HttpSession session =
17             request.getSession();
18         session.setAttribute("name", "sdd");
19         response.sendRedirect("other");
20     }
21 }
22
```

2) 新建 OtherServlet

```

1 package web;
2
3 import java.io.IOException;
4
11
12 public class OtherServlet extends HttpServlet {
13
14     public void service(HttpServletRequest request,
15                         HttpServletResponse response)
16         throws ServletException, IOException {
17
18         response.setContentType("text/html");
19         PrintWriter out = response.getWriter();
20         HttpSession session = request.getSession();
21         String name =
22             (String) session.getAttribute("name");
23         out.println(name);
24         out.close();
25     }
26 }
27

```

3) web.xml

```

1 web.xml
2
6     http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd
7 <servlet>
8     <servlet-name>SomeServlet</servlet-name>
9     <servlet-class>web.SomeServlet</servlet-class>
10 </servlet>
11 <servlet>
12     <servlet-name>OtherServlet</servlet-name>
13     <servlet-class>web.OtherServlet</servlet-class>
14 </servlet>
15 <servlet-mapping>
16     <servlet-name>SomeServlet</servlet-name>
17     <url-pattern>/some</url-pattern>
18 </servlet-mapping>
19 <servlet-mapping>
20     <servlet-name>OtherServlet</servlet-name>
21     <url-pattern>/other</url-pattern>
22 </servlet-mapping>
23 </web-app>
24

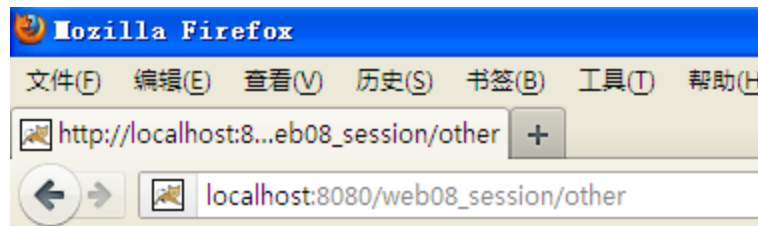
```


浏览器没有禁用 Cookie

4) 部署项目

5) 访问 http://localhost:8080/web08_session/some

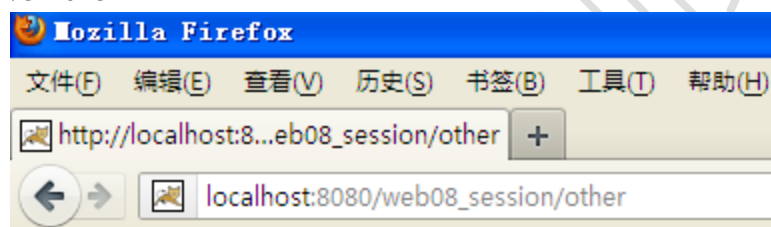
重定向后可以显示放入 Session 中的内容



sdd

禁用 Cookie

则找不到要访问的 Session



null

- 使用 URL 重写

6) 修改 SomeServlet

```

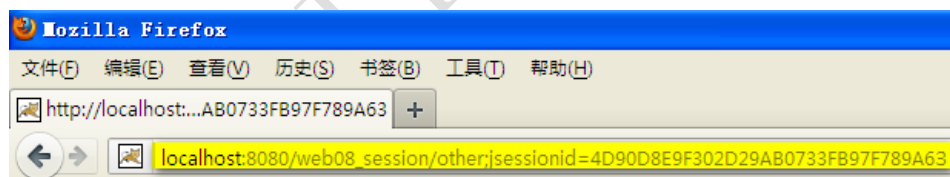
1 package web;
2
3 import java.io.IOException;
4
5
6
7
8
9
10 public class SomeServlet extends HttpServlet {
11
12     public void service(HttpServletRequest request,
13                         HttpServletResponse response)
14         throws ServletException, IOException {
15         HttpSession session =
16             request.getSession();
17         session.setAttribute("name", "sdd");
18         response.sendRedirect(
19             response.encodeRedirectURL("other"));
20     }
21 }
22
23

```

禁用 Cookie

7) 访问 http://localhost:8080/web08_session/some

访问地址中将加入 SessionId



sdd

8) 案例结束

3. 过滤器 **

1) 什么是过滤器

servlet 规范当中定义的一种特殊的类，用于对 servlet 容器的调用过程进行拦截。

2) 怎样写一个过滤器

step1

写一个 java 类，实现一个 Filter 接口。

step2

在 doFilter 方法里，实现过滤的逻辑。

step3

配置(web.xml)。

3) 配置初始化参数

step1

web.xml 中，使用<init-param>元素来配置初始化参数

step2

在 Filter 类中，使用 FilterConfig.getInitParameter(String paraName);获得初始化参数。

4) 过滤器的优先级

当有多个过滤器都满足过滤的条件时，依据<filter-mapping>的先后顺序依次执行。

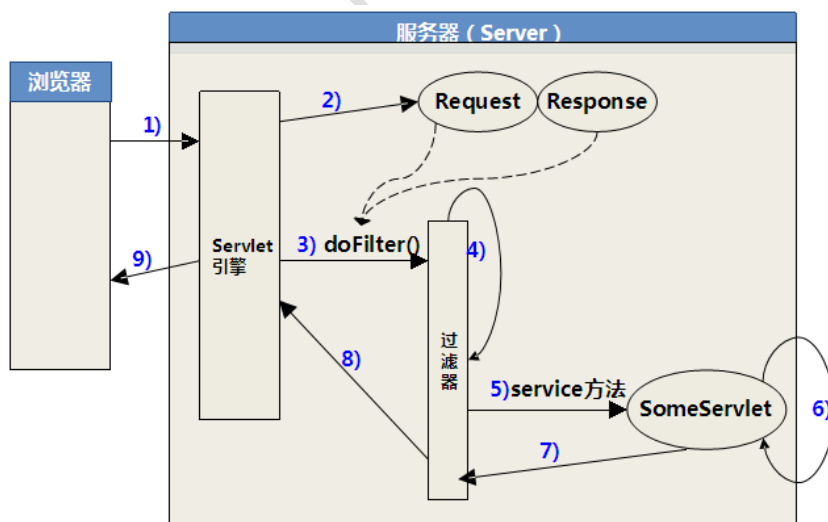
5) 过滤器的优点

- 可以将多个 web 组件相同的逻辑写在一个过滤器当中，方便代码的维护
- 可实现代码的“可插拔性”。
给一个软件增加或者减少某个功能不会影响已经存在的功能。

3.1. 什么是过滤器

servlet 规范当中定义的一种特殊的类，用于对 servlet 容器的调用过程进行拦截。

图示演示



- 1) 浏览器发送请求给服务器
- 2) 服务器的 Servlet 引擎创建 Request 对象&&Response 对象

- 3) Servlet 引擎先调用过滤器的 doFilter 方法，该方法有两个参数 request 和 response，（在过滤器中可以访问到 Request 对象和 Response 对象）
- 4) 过滤器对拦截的内容进行处理
- 5) 之后调用 SomeServlet 的 service 方法
- 6) service 方法执行
- 7) service 方法执行结束后，将结果返回到过滤器
- 8) 过滤器将 service 方法返回的结果再次进行过滤
- 9) 最后，Servlet 引擎将结果返回给浏览器

3.2. 怎样写一个过滤器 **

step1

写一个 java 类，实现一个 Filter 接口。

step2

在 doFilter 方法里，实现过滤的逻辑。

step3

配置(web.xml)。

演示

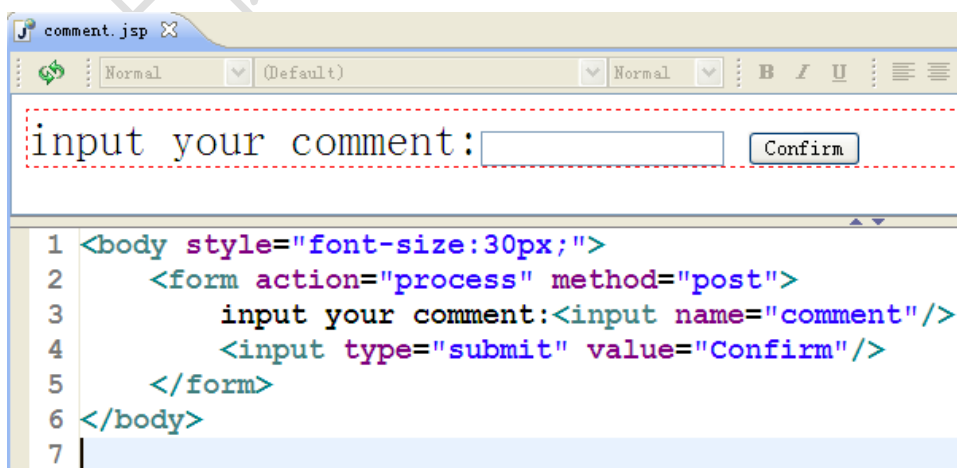
过滤器

对用户评论进行过滤，如果用户输入带“dog”的字符串，则告诉用户“不合法”。

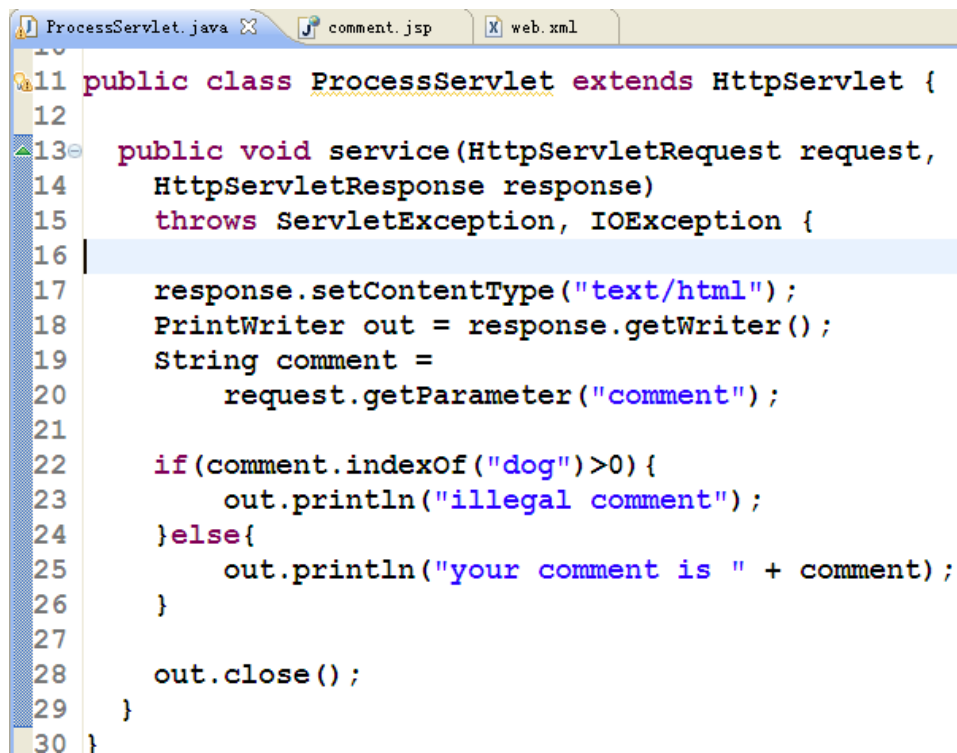
● 演示 1

不添加过滤器，在 Servlet 中验证

- 1) 新建工程 web08
- 2) 新建 comment.jsp



3) 新建 ProcessServlet



The screenshot shows an IDE with three tabs: ProcessServlet.java, comment.jsp, and web.xml. The ProcessServlet.java tab is active, displaying the following Java code:

```
11 public class ProcessServlet extends HttpServlet {
12
13     public void service(HttpServletRequest request,
14         HttpServletResponse response)
15         throws ServletException, IOException {
16
17         response.setContentType("text/html");
18         PrintWriter out = response.getWriter();
19         String comment =
20             request.getParameter("comment");
21
22         if(comment.indexOf("dog")>0){
23             out.println("illegal comment");
24         }else{
25             out.println("your comment is " + comment);
26         }
27
28         out.close();
29     }
30 }
```

4) web.xml

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="2.4"
3     xmlns="http://java.sun.com/xml/ns/j2ee"
4     xmlns:xsi="http://www.w3.org/2001/XMLSchema-in:
5     xsi:schemaLocation="http://java.sun.com/xml/ns,
6     http://java.sun.com/xml/ns/j2ee/web-app_2_4.xs
7 <servlet>
8     <servlet-name>ProcessServlet</servlet-name>
9     <servlet-class>web.ProcessServlet</servlet-cla
10 </servlet>
11
12 <servlet-mapping>
13     <servlet-name>ProcessServlet</servlet-name>
14     <url-pattern>/process</url-pattern>
15 </servlet-mapping>
16
17 </web-app>

```

5) 部署项目

6) 测试

a. 访问 <http://localhost:8080/web08/comment.jsp>

输入合法的字符串 "aaa"

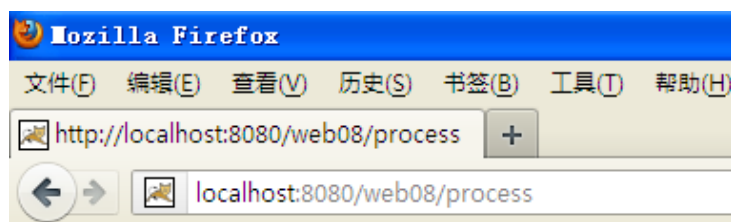


input your comment:

aaa

Confirm

b. 显示



your comment is aaa

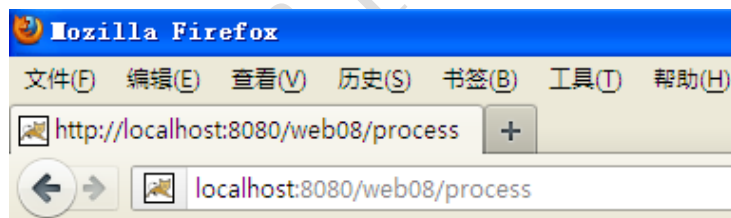
c. 输入不合法字符串 “who let the dogs out?”



input your comment:

who let the dogs out?
Confirm

d. 显示



illegal comment

● 演示 2

由过滤器负责过滤敏感字

7) 修改 ProcessServlet

```
package web;
```

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ProcessServlet extends HttpServlet {

    public void service(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        System.out.println("servlet begin process...");
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String comment = request.getParameter("comment");
        out.println("your comment is " + comment);
        out.close();
        System.out.println("servlet process finished.");
    }
}
```

8) 创建 CommentFilter1

```
package web;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```



```
public class CommentFilter1 implements Filter{

    /**
     * 在容器删除 Filter 实例之前，调用该方法。
     * 只会执行一次。
     */
    public void destroy() {
        System.out.println("destroy...");
    }

    /**
     * 当请求到达容器，容器会调用 doFilter 方法。
     * 容器会将事先创建好的 request,response 对象作为
     * 参数传递进来。
     * FilterChain：过滤器链。
     * 如果调用了 FilterChain.doFilter 方法，表示调用
     * 后续的过滤器。如果没有过滤器了，则调用对应的
     * web 组件。
     */
    public void doFilter(ServletRequest arg0,
        ServletResponse arg1, FilterChain arg2)
        throws IOException, ServletException {
        System.out.println("Filter1 begin filter...");
        HttpServletRequest request =
            (HttpServletRequest)arg0;
        HttpServletResponse response =
            (HttpServletResponse)arg1;
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String comment = request.getParameter("comment");
        if(comment.indexOf("dog") > 0){
            out.println("illegal comment.");
        }else{
            //调用后续的过滤器，如果没有了，则调用 web 组件
            arg2.doFilter(arg0, arg1);
        }
        System.out.println("Filter1 process finished.");
    }
}
```

```

    }

    /**
     * servlet 容器在创建好 Filter 实例之后，会立即调用
     * init()方法。容器会创建 FilterConfig 实例，通过该
     * 实例，可以访问 Filter 的初始化参数。
     * String FilterConfig.getInitParameter(String paraName);
     * init()方法只会执行一次。
     */
    public void init(FilterConfig arg0) throws ServletException {
        System.out.println("init...");
    }
}

```

9) 修改 web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4"
    xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <filter>
        <filter-name>filter1</filter-name>
        <filter-class>web.CommentFilter1</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>filter1</filter-name>
        <url-pattern>/process</url-pattern>
    </filter-mapping>
    <servlet>
        <servlet-name>ProcessServlet</servlet-name>
        <servlet-class>web.ProcessServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>ProcessServlet</servlet-name>
        <url-pattern>/process</url-pattern>
    </servlet-mapping>

```

```
</web-app>
```

10) 部署项目

11) 测试

测试结果相同

项目完成

3.3. 配置初始化参数 **

除了在过滤器代码中过滤“敏感字”以外，还可以配置初始化参数。

step1

web.xml 中，使用<init-param>元素来配置初始化参数

step2

在 Filter 类中，使用 FilterConfig.getInitParameter(String paraName);获得初始化参数。

演示 3

为过滤器配置初始化参数

12) 修改 web.xml

13) 修改 CommentFilter1

```
package web;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class CommentFilter1 implements Filter{
    private FilterConfig config;
```

```

/**
 * 在容器删除Filter实例之前，调用该方法。
 * 只会执行一次。
 */
public void destroy() {
    System.out.println("destroy...");
}

/**
 * 当请求到达容器，容器会调用doFilter方法。
 * 容器会将事先创建好的request,response对象作为
 * 参数传递进来。
 * FilterChain：过滤器链。
 * 如果调用了FilterChain.doFilter方法，表示调用
 * 后续的过滤器。如果没有过滤器了，则调用对应的
 * web组件。
 */
public void doFilter(ServletRequest arg0,
    ServletResponse arg1, FilterChain arg2)
    throws IOException, ServletException {
    System.out.println("Filter1 begin filter...");
    HttpServletRequest request =
        (HttpServletRequest)arg0;
    HttpServletResponse response =
        (HttpServletResponse)arg1;
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String comment = request.getParameter("comment");
    String illegalStr =
        config.getInitParameter("illegalStr");
    if(comment.indexOf(illegalStr) > 0){
        out.println("illegal comment.");
    }else{
        //调用后续的过滤器，如果没有了，则调用web组件
        arg2.doFilter(arg0, arg1);
    }
    System.out.println("Filter1 process finished.");
}

```

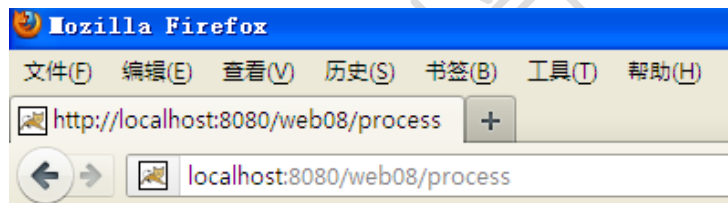
```
/**
 * servlet容器在创建好Filter实例之后，会立即调用
 * init()方法。容器会创建FilterConfig实例，通过该
 * 实例，可以访问Filter的初始化参数。
 * String FilterConfig.getInitParameter(String paraName);
 * init()方法只会执行一次。
 */
public void init(FilterConfig arg0) throws ServletException {
    System.out.println("init...");
    config = arg0;
}
}
```

14) 部署项目

15) 测试

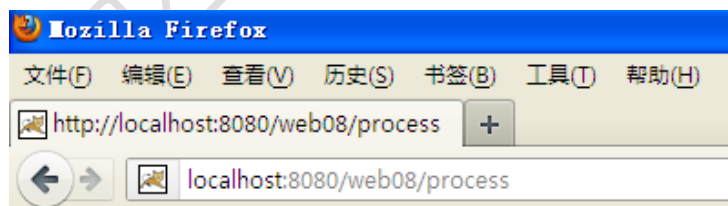
当用户输入带“cat”的字符串时提示“不合法”

a. 当用户输入“aaadog”没有关系



your comment is aaadog

b. 当用户输入“aaacat”有关系



illegal comment.

3.4. 课堂练习 **

练习描述

写一个 CommentFilter2 过滤器，该过滤器会检查 comment 参数值是否长度超过指定的范围(该范围使用初始化参数来配置，比如 20)。如果超过长度范围，提示用户，否则，显示 comment 参数值。

参考代码

请参考本文档中的【案例 4】过滤器

3.5. 过滤器的优先级 *

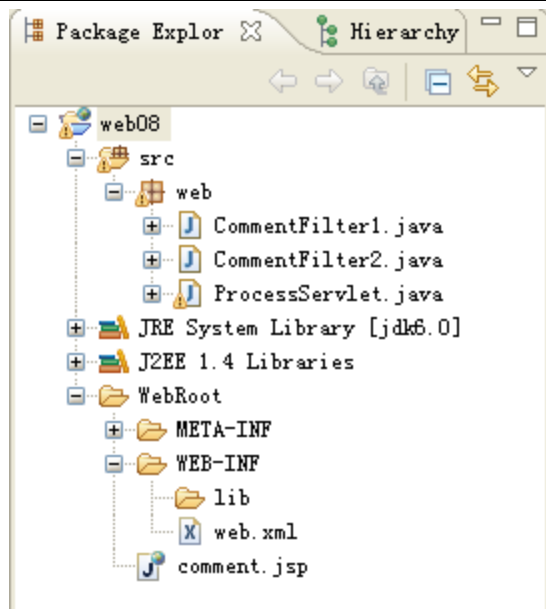
当有多个过滤器都满足过滤的条件时，依据<filter-mapping>的先后顺序依次执行。

3.6. 过滤器的优点 *

- a. 可以将多个 web 组件相同的逻辑写在一个过滤器当中，方便代码的维护
- b. 可实现代码的“可插拔性”。
给一个软件增加或者减少某个功能不会影响已经存在的功能。

【案例 4】过滤器 **

1) 项目结构



2) ProcessServlet

```
package web;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ProcessServlet extends HttpServlet {

    public void service(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        System.out.println("servlet begin process...");
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String comment = request.getParameter("comment");
        out.println("your comment is " + comment);
    }
}
```

```
        out.close();
        System.out.println("servlet  process finished.");
    }

}
```

3) CommentFilter1

```
package web;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class CommentFilter1 implements Filter{
    private FilterConfig config;
    /**
     * 在容器删除 Filter 实例之前，调用该方法。
     * 只会执行一次。
     */
    public void destroy() {
        System.out.println("destroy...");
    }

    /**
     * 当请求到达容器，容器会调用 doFilter 方法。
     * 容器会将事先创建好的 request,response 对象作为
     * 参数传递进来。
     * FilterChain：过滤器链。
     * 如果调用了 FilterChain.doFilter 方法，表示调用
```



```

    * 后续的过滤器。如果没有过滤器了，则调用对应的
    * web 组件。
    */
    public void doFilter(ServletRequest arg0,
        ServletResponse arg1, FilterChain arg2)
        throws IOException, ServletException {
        System.out.println("Filter1 begin filter...");
        HttpServletRequest request =
            (HttpServletRequest)arg0;
        HttpServletResponse response =
            (HttpServletResponse)arg1;
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String comment = request.getParameter("comment");
        String illegalStr = config.getInitParameter("illegalStr");
        if(comment.indexOf(illegalStr) > 0){
            out.println("illegal comment.");
        }else{
            //调用后续的过滤器，如果没有了，则调用 web 组件
            arg2.doFilter(arg0, arg1);
        }
        System.out.println("Filter1 process finished.");
    }

    /**
    * servlet 容器在创建好 Filter 实例之后，会立即调用
    * init()方法。容器会创建 FilterConfig 实例，通过该
    * 实例，可以访问 Filter 的初始化参数。
    * String FilterConfig.getInitParameter(String paraName);
    * init()方法只会执行一次。
    */
    public void init(FilterConfig arg0) throws ServletException {
        System.out.println("init...");
        config = arg0;
    }
}

```

4) CommentFilter2

```
package web;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class CommentFilter2 implements Filter{
    private FilterConfig config;
    public void destroy() {
    }

    public void doFilter(ServletRequest arg0,
        ServletResponse arg1, FilterChain arg2)
        throws IOException, ServletException {
        System.out.println("Filter2 begin process...");
        HttpServletRequest request =
            (HttpServletRequest)arg0;
        HttpServletResponse response =
            (HttpServletResponse)arg1;
        String comment = request.getParameter("comment");
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        int size = Integer.parseInt(
            config.getInitParameter("size"));
        if(comment.length() > size){
            out.println("illegal size.");
        }else{
```

```

        arg2.doFilter(arg0, arg1);
    }
    System.out.println("Filter2 process finished.");
}

public void init(FilterConfig arg0) throws ServletException {
    config = arg0;
}
}

```

5) web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4"
    xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

    <filter><body style="font-size:30px;">

        <form action="process" method="post">

            input your comment:<input
name="comment"/>

            <input type="submit" value="Confirm"/>

        </form>

    </body>

    <filter-name>filter1</filter-name>
    <filter-class>web.CommentFilter1</filter-class>
    <init-param>
        <param-name>illegalStr</param-name>
        <param-value>cat</param-value>

```

```

    </init-param>
</filter>
<filter>
    <filter-name>filter2</filter-name>
    <filter-class>web.CommentFilter2</filter-class>
    <init-param>
        <param-name>size</param-name>
        <param-value>20</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>filter1</filter-name>
    <url-pattern>/process</url-pattern>
</filter-mapping>
<!--
<filter-mapping>
    <filter-name>filter2</filter-name>
    <url-pattern>/process</url-pattern>
</filter-mapping>
-->
<servlet>
    <servlet-name>ProcessServlet</servlet-name>
    <servlet-class>web.ProcessServlet</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>ProcessServlet</servlet-name>
    <url-pattern>/process</url-pattern>
</servlet-mapping>

</web-app>

```

6) comment.jsp

```

<body style="font-size:30px;">
    <form action="process" method="post">
        input your comment:<input name="comment"/>
        <input type="submit" value="Confirm"/>
    </form>

```

```
</form>
</body>
```

4. 监听器 **

1) 什么是监听器?

servlet 规范当中定义的一种特殊的类，作用是监听容器当中产生的一些事件并进行相应的处理。

容器产生的事件指的是两大类事件：

第一大类

生命周期相关的事件，指的是当容器创建或者销毁 request,session,ServletContext 对象时产生的事件。

第二大类

绑定事件，指的是当调用 request,session,ServletContext 对象的 setAttribute,removeAttribute 时产生的事件。

2) 如何写监听器

step1

写一个 java 类，实现特定的监听器接口类(依据要监听的事件类型)。

step2

在接口声明的方法中，实现监听的逻辑。

step3

配置(web.xml)。

3) ServletContext 接口

web 服务器在启动时，会为每一个已经部署的应用创建唯一的一个 ServletContext 实例。

该实例会一直存在，除非服务器关闭或者应用被删除。

注意：每个应用对应唯一的一个 ServletContext 实例

a. 如何获得 ServletContext 实例。

GenericServlet 提供了 getServletContext()方法。

HttpSession 提供了 getServletContext()方法。

ServletConfig 提供了 getServletContext()方法。

b. 常用方法

✓ 绑定数据

```
setAttribute(String name,Object obj);
getAttribute(String name);
removeAttribute(String name);
```

✓ 配置全局的初始化参数

step1

在 web.xml 中,使用<context-param>配置的参数,可以被所有的 servlet 共享。

step2

使用 String ServletContext.getInitParameter(String paraName);

- ✓ 依据逻辑路径获得实际部署时的物理路径。

String ServletContext.getRealPath(String url);

4) 上传文件(扩展)

step1

在 form 中,设置 method="post",设置 enctype="multipart/form-data"。enctype 属性用于设置表单的编码方式,对于文件上传,必须设置成"multipart/form-data"。

step2

在 servlet 类当中,不能够使用 request.getParameter 方法来获得参数值。要使用 InputStream request.getInputStream();分析 InputStream 来获得参数值。直接分析 InputStream 比较复杂,一般使用一些封装好的工具(比如 apache 提供的 commons-fileupload.jar)来获得参数值。

4.1. 如何写监听器 **

实现监听的步骤

step1

写一个 java 类,实现特定的监听器接口类(依据要监听的事件类型)。

step2

在接口声明的方法中,实现监听的逻辑。

step3

配置(web.xml)。

案例演示

ServletContext&&监听器

功能描述

统计在线人数。

实现步骤

- ServletContext 演示

1) 新建 SomeServlet

```

12 public class SomeServlet extends HttpServlet {
13
14     public void service(HttpServletRequest request,
15                         HttpServletResponse response)
16         throws ServletException, IOException {
17
18         response.setContentType(
19             "text/html;charset=utf-8");
20         PrintWriter out = response.getWriter();
21         //获得ServletContext:
22         ServletContext sctx =
23             getServletContext();
24         sctx.setAttribute("username", "user1");
25         out.println(
26             "已经将username绑定" +
27             "到了ServletContext上面");
28         out.close();
29     }
30 }
31

```

2) 新建 OtherServlet

```

1 package web;
2
3 import java.io.IOException;
4
11
12 public class OtherServlet extends HttpServlet {
13
14     public void service(HttpServletRequest request,
15                         HttpServletResponse response)
16         throws ServletException, IOException {
17         response.setContentType(
18             "text/html;charset=utf-8");
19         PrintWriter out = response.getWriter();
20         ServletContext sctx = getServletContext();
21         String username =
22             (String) sctx.getAttribute("username");
23         out.println("username:" + username);
24         out.close();
25     }
26 }

```

3) web.xml

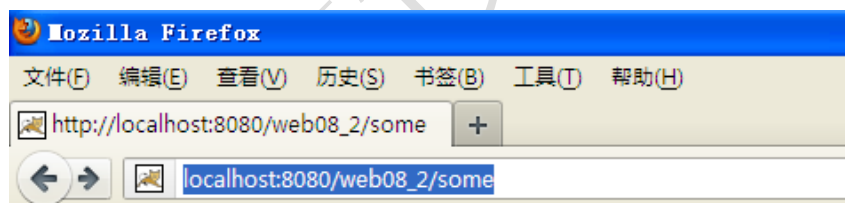
```

6      http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd
7  <servlet>
8      <servlet-name>SomeServlet</servlet-name>
9      <servlet-class>web.SomeServlet</servlet-class>
10 </servlet>
11 <servlet>
12     <servlet-name>OtherServlet</servlet-name>
13     <servlet-class>web.OtherServlet</servlet-class>
14 </servlet>
15 <servlet-mapping>
16     <servlet-name>SomeServlet</servlet-name>
17     <url-pattern>/some</url-pattern>
18 </servlet-mapping>
19 <servlet-mapping>
20     <servlet-name>OtherServlet</servlet-name>
21     <url-pattern>/other</url-pattern>
22 </servlet-mapping>
23 </web-app>
24

```

4) 测试

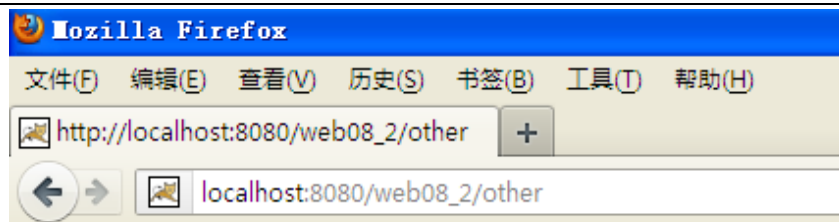
a. 访问 http://localhost:8080/web08_2/some



已经将username绑定到了ServletContext上面

b. 重启浏览器

c. 访问 http://localhost:8080/web08_2/other



username:user1

和 Session 不同（关闭浏览器后 session 对象即消失），除非把服务器关闭，否则在 servletContext 实例中保存的数据会一直存在

- 全局初始化变量演示

- 5) 修改 web.xml

配置全局初始化变量

a. 这样的配置只能被 SomeServlet 访问，不能被其他 Servlet 访问



b. 全局初始化参数，能被所有 Servlet 访问到

注意：要写在<servlet>之前

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app version="2.4"
3      xmlns="http://java.sun.com/xml/ns/j2ee"
4      xmlns:xsi="http://www.w3.org/2001/XMLSchema-in
5      xsi:schemaLocation="http://java.sun.com/xml/ns,
6      http://java.sun.com/xml/ns/j2ee/web-app_2_4.xs
7
8  <!-- 全局初始化参数 -->
9  <context-param>
10     <param-name>version</param-name>
11     <param-value>1.0</param-value>
12 </context-param>
13 <servlet>
14     <servlet-name>SomeServlet</servlet-name>
15     <servlet-class>web.SomeServlet</servlet-class>
16 </servlet>
17 <servlet>
18     <servlet-name>OtherServlet</servlet-name>
19     <servlet-class>web.OtherServlet</servlet-class:
20 </servlet>
21 <servlet-mapping>

```

访问全局初始化变量

6) 修改 SomeServlet

```

SomeServlet.java X OtherServlet.java
13
14 public void service(HttpServletRequest request,
15                     HttpServletResponse response)
16                     throws ServletException, IOException {
17
18     response.setContentType(
19         "text/html;charset=utf-8");
20     PrintWriter out = response.getWriter();
21     //获得ServletContext:
22     ServletContext sctx = getServletContext();
23     sctx.setAttribute("username", "user1");
24     out.println(
25         "已经将username绑定到了" +
26         "ServletContext上面");
27     String version =
28         sctx.getInitParameter("version");
29     out.println("version:" + version);
30     out.close();
31 }
32
33 }

```

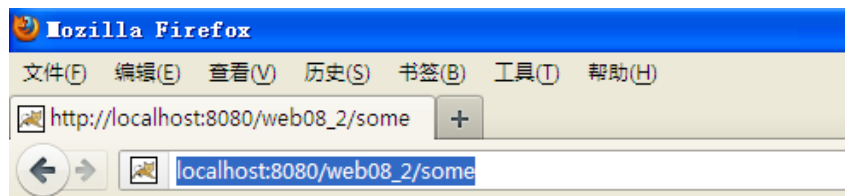
7) 修改 OtherServlet

```

OtherServlet.java X
11
12 public class OtherServlet extends HttpServlet {
13
14 public void service(HttpServletRequest request,
15                     HttpServletResponse response)
16                     throws ServletException, IOException {
17     response.setContentType(
18         "text/html;charset=utf-8");
19     PrintWriter out = response.getWriter();
20     ServletContext sctx = getServletContext();
21     String username = (String)
22     sctx.getAttribute("username");
23     out.println("username:" + username);
24     String version =
25         sctx.getInitParameter("version");
26     out.println("version:" + version);
27     out.close();
28 }
29 }

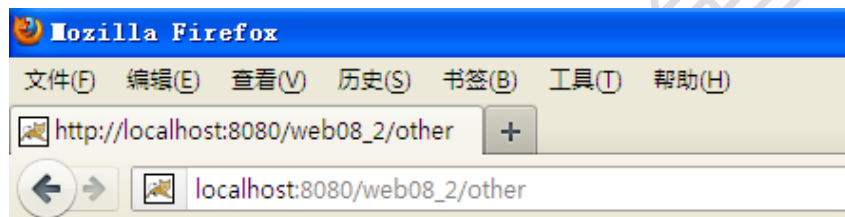
```

8) 访问 http://localhost:8080/web08_2/some



已经将username绑定到了ServletContext上面
version:1.0

9) 访问 http://localhost:8080/web08_2/other



username:user1 version:1.0

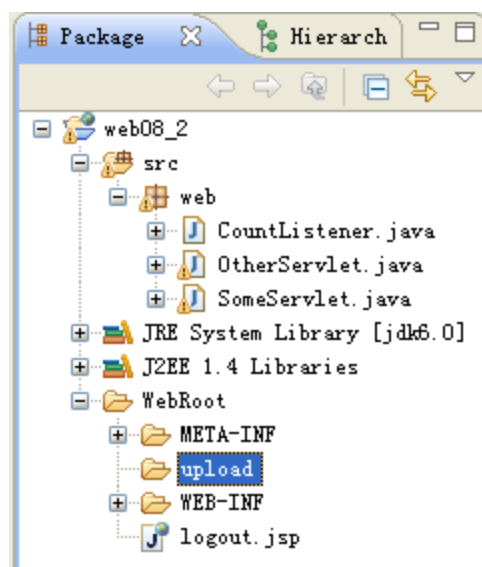
10) 配置多个初始化参数

注意：每个<context-param>只能配一对<param>

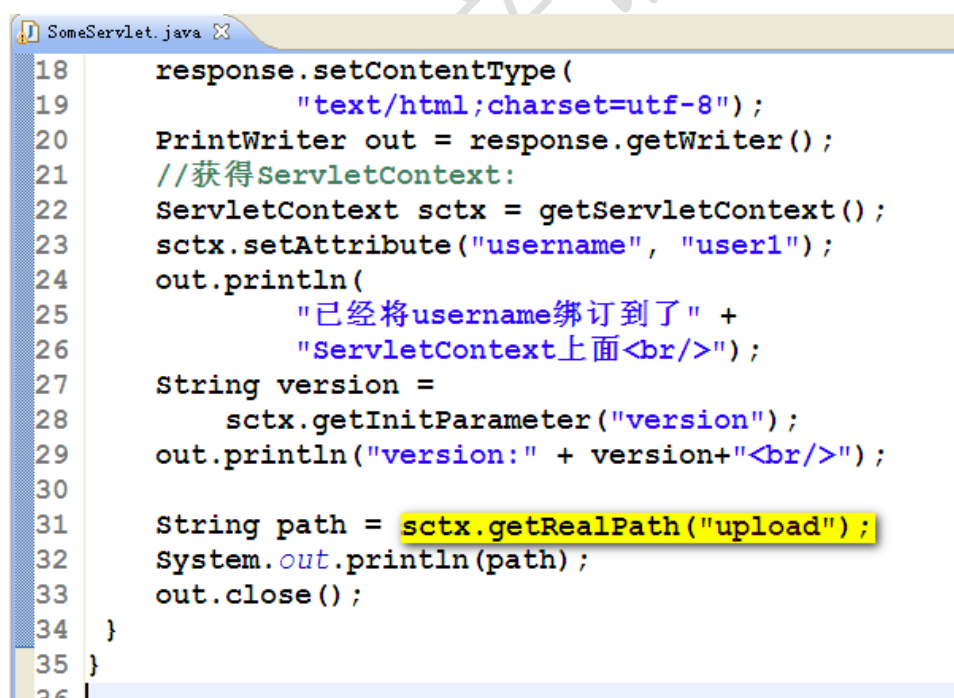


获得应用的物理路径

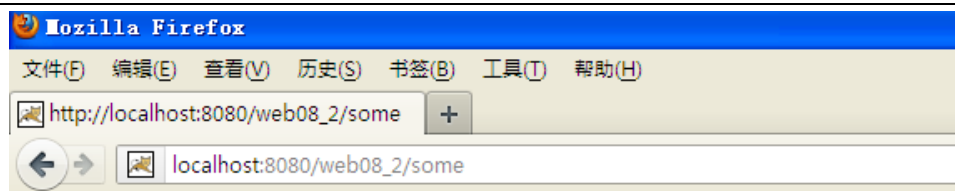
11) 新建目录 upload



12) 修改 SomeServlet



13) 访问 http://localhost:8080/web08_2/some

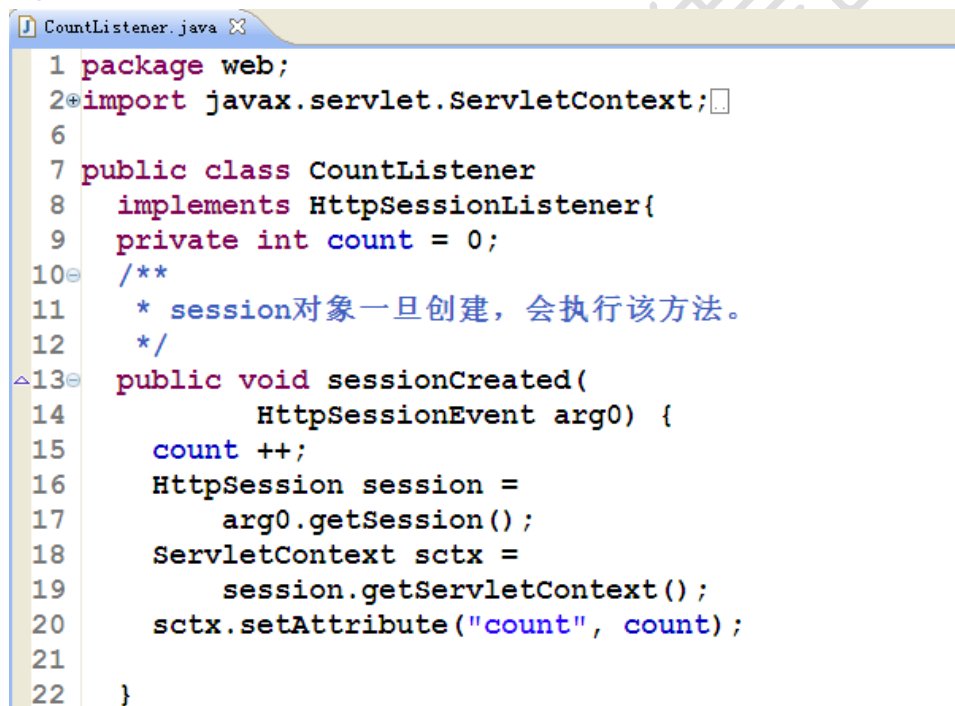


已经将username绑定到了ServletContext上面
version:1.0

D:\apache-tomcat-5.5.23\webapps\web08_2\upload

- 实现统计在线人数

14) 新建 CountListener



```

23  /**
24   * session对象被销毁，会执行该方法
25   */
26  public void sessionDestroyed(
27      HttpSessionEvent arg0) {
28      count --;
29      HttpSession session =
30          arg0.getSession();
31      ServletContext sctx =
32          session.getServletContext();
33      sctx.setAttribute("count", count);
34  }
35 }

```

15) 配置 web.xml

配置监听器

注意：有先后顺序的要求，在<context-param>之后，<servlet>之前；
<filter>在<listener>之后，<servlet>之前



```

web.xml CountListener.java
4      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5      xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
6      http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
7      <!-- 全局的初始化参数 -->
8      <context-param>
9          <param-name>version</param-name>
10         <param-value>1.0</param-value>
11     </context-param>
12     <!-- 配置监听器 -->
13     <!-->
14     <listener>
15         <listener-class>web.CountListener</listener-class>
16     </listener>
17     <!-- 配置过滤器 -->
18     <!-- 过滤器在监听器之后配置 -->
19     <!-->
20     <servlet>
21         <servlet-name>SomeServlet</servlet-name>
22         <servlet-class>web.SomeServlet</servlet-class>
23     </servlet>
24

```

小知识

首页

16) 配置当用户访问地址 http://localhost:8080/web08_2 的默认页

```

index1.jsp  web.xml
28  </servlet-mapping>
29  <servlet-mapping>
30      <servlet-name>OtherServlet</servlet-name>
31      <url-pattern>/other</url-pattern>
32  </servlet-mapping>
33  <welcome-file-list>
34      <welcome-file>index1.jsp</welcome-file>
35  </welcome-file-list>
36 </web-app>
37

```

17) 新建 index1.jsp

```

index1.jsp  web.xml
1  <%@page pageEncoding="utf-8"
2  contentType="text/html; charset=utf-8" %>
3  <body style="font-size:30px;">
4      当前系统在线人数是:
5      <%=application.getAttribute("count") %>
6      <br/>
7      <a href="logout.jsp">退出系统</a>
8  </body>
9

```

18) 新建 logout.jsp

```

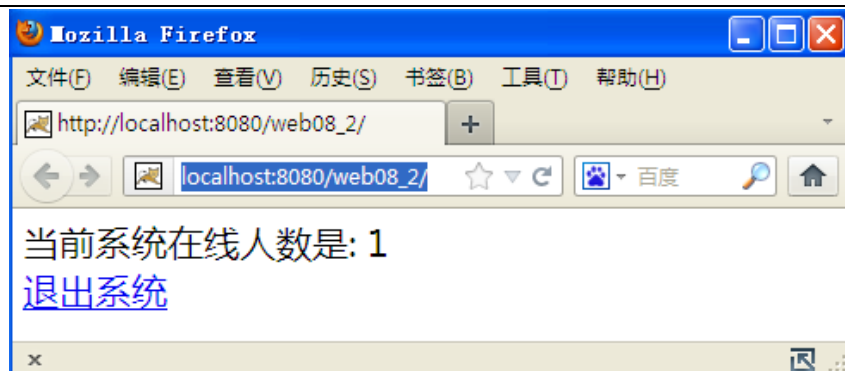
logout.jsp
1  <%
2      session.invalidate();
3  %>
4

```

19) 部署项目；重启服务器

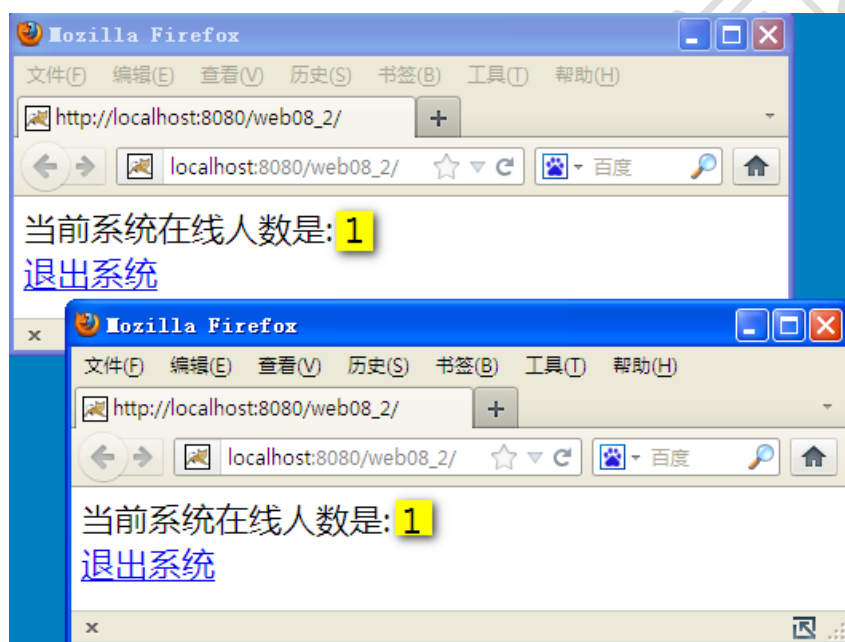
20) 测试

a. 访问 http://localhost:8080/web08_2/



b. 启动两个火狐浏览器

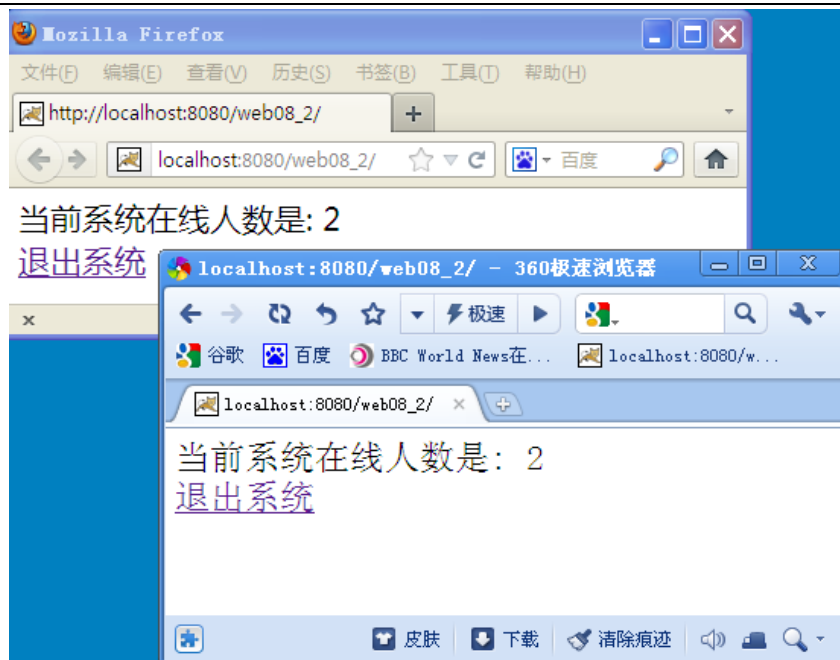
注意：因为多个火狐浏览器在一台计算机共享一块内存，所以不论启动多少个窗口都显示是一个用户



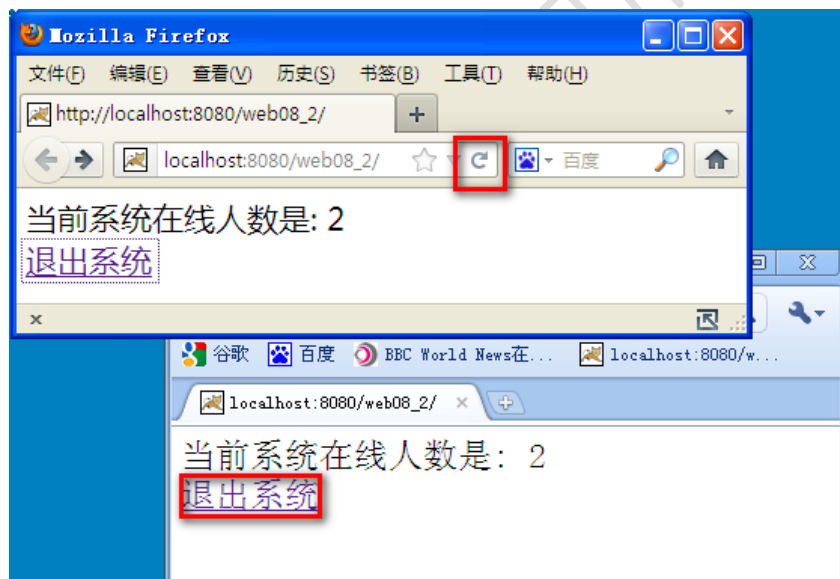
c. 重启服务器

d. 同时使用 FF 和其它浏览器打开

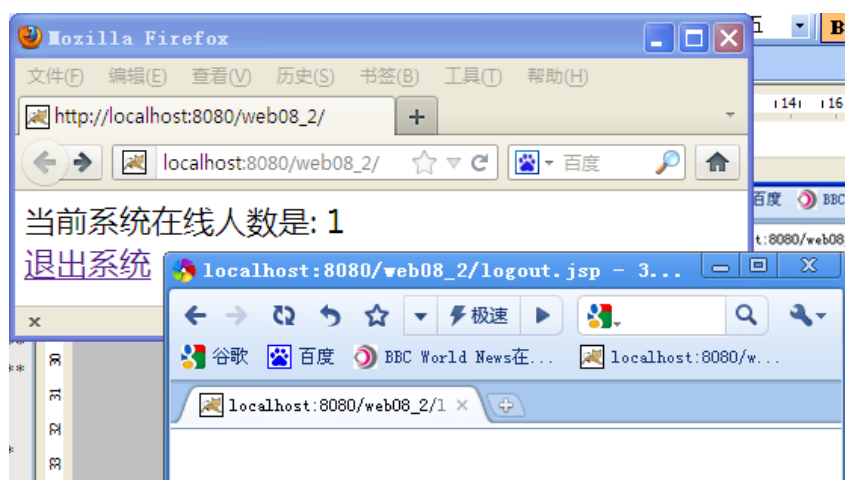
因为使用的不是同一块内存



e. 360 浏览器点击“退出系统”；同时“刷新”火狐浏览器



f. 显示结果



【案例 5】监听器 **

案例描述

要统计在线人数。

参考代码

- 1) 新建工程 web08_2
- 2) SomeServlet

```
package web;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class SomeServlet extends HttpServlet {

    public void service(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
```

```
response.setContentType(
    "text/html;charset=utf-8");
PrintWriter out = response.getWriter();
//获得 ServletContext:
ServletContext sctx = getServletContext();
sctx.setAttribute("username", "user1");
out.println(
    "已经将 username 绑定到了" +
    "ServletContext 上面<br/>");
String version =
    sctx.getInitParameter("version");
out.println("version:" + version+"<br/>");

String path = sctx.getRealPath("upload");
out.println(path);
out.close();
}
}
```

3) OtherServlet

```
package web;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class OtherServlet extends HttpServlet {

    public void service(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType(
```

```

        "text/html;charset=utf-8");
        PrintWriter out = response.getWriter();
        ServletContext sctx = getServletContext();
        String username = (String)
            sctx.getAttribute("username");
        out.println("username:" + username);
        String version =
            sctx.getInitParameter("version");
        out.println("version:" + version);
        out.close();
    }
}

```

4) CountListener

```

package web;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpSession;
import javax.servlet.http.HttpSessionEvent;
import javax.servlet.http.HttpSessionListener;

public class CountListener
    implements HttpSessionListener{
    private int count = 0;
    /**
     * session 对象一旦创建，会执行该方法。
     */
    public void sessionCreated(
        HttpSessionEvent arg0) {
        count ++;
        HttpSession session =
            arg0.getSession();
        ServletContext sctx =
            session.getServletContext();
        sctx.setAttribute("count", count);
    }
    /**
     * session 对象被销毁，会执行该方法
     */
}

```

```

*/
public void sessionDestroyed(
    HttpSessionEvent arg0) {
    count--;
    HttpSession session =
        arg0.getSession();
    ServletContext sctx =
        session.getServletContext();
    sctx.setAttribute("count", count);
}
}

```

5) web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4"
    xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
        http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <!-- 全局的初始化参数 -->
    <context-param>
        <param-name>version</param-name>
        <param-value>1.0</param-value>
    </context-param>
    <!-- 配置监听器 -->
    <listener>
        <listener-class>web.CountListener</listener-class>
    </listener>
    <!-- 配置过滤器 -->
    <servlet>
        <servlet-name>SomeServlet</servlet-name>
        <servlet-class>web.SomeServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>OtherServlet</servlet-name>
        <servlet-class>web.OtherServlet</servlet-class>
    </servlet>
    <servlet-mapping>

```

```
<servlet-name>SomeServlet</servlet-name>
<url-pattern>/some</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>OtherServlet</servlet-name>
  <url-pattern>/other</url-pattern>
</servlet-mapping>
<welcome-file-list>
  <welcome-file>index1.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

6) index1.jsp

```
<%@page pageEncoding="utf-8"
contentType="text/html;charset=utf-8" %>
<body style="font-size:30px;">
  当前系统在线人数是:
  <%=application.getAttribute("count")%>
  <br/>
  <a href="logout.jsp">退出系统</a>
</body>
```

7) logout.jsp

```
<%
  session.invalidate();
%>
```

4.2. 上传文件(扩展) *

仅作了解，因为在今后的开发中，上传文件的操作已经被一些框架封装好了，直接调用其提供的借口即可。

步骤

step1

在 form 中，设置 method="post",设置 enctype="multipart/form-data"。enctype 属性用于设置表单的编码方式，对于文件上传，必须设置成"multipart/form-data"。

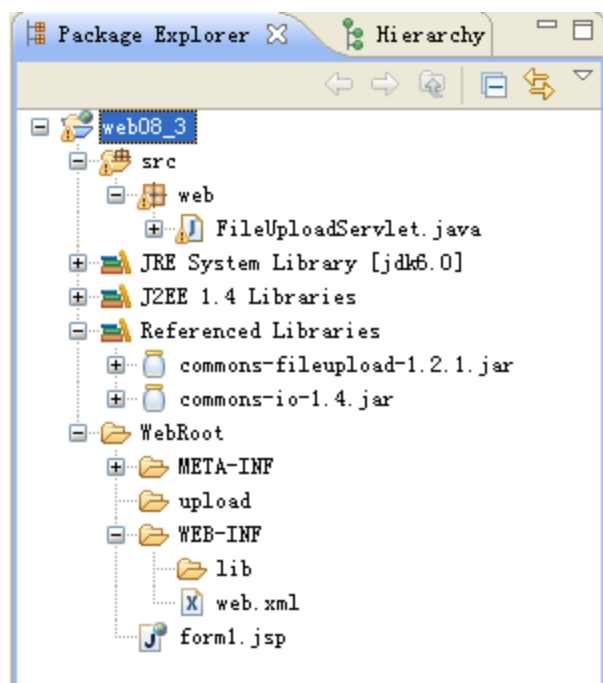
step2

在 servlet 类当中，不能够使用 request.getParameter 方法来获得参数值。要使用 `InputStream request.getInputStream()` 分析 `InputStream` 来获得参数值。直接分析 `InputStream` 比较复杂，一般使用一些封装好的工具(比如 apache 提供的 `commons-fileupload.jar`)来获得参数值。

请下载 `commons-fileupload-1.2.1-javadoc.zip`

【案例 6】使用工具完成上传文件 *

- 1) 请下载 `upload_lib.zip`
- 2) 导入 Jar 包
- 3) 项目结构如下



4) 新建 form1.jsp

```
<%@page pageEncoding="utf-8" contentType="text/html; charset=utf-8" %>
<html>
<head></head>
<body style="font-size:30px;">
    <form action="fileupload" method="post"
        enctype="multipart/form-data">
```



```
username:<input name="username"/>
<br/>
phone:<input type="file" name="file1"/>
<br/>
<input type="submit" value="Confirm"/>
</form>
</body>
</html>
```

5) 新建 FileUploadServlet

```
package web;

import java.io.File;
import java.io.IOException;
import java.util.List;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;

public class FileUploadServlet extends HttpServlet {

    public void service(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        //step1 创建一个工厂类的实例，该实例
        //为解析器提供了缺省的配置。
        DiskFileItemFactory factory =
            new DiskFileItemFactory();
        //step2 创建一个解析器
        ServletFileUpload sfu =
            new ServletFileUpload(factory);
```

//step3 使用解析器解析

```
try {
```

```
    //解析之后，会将表单中的数据转换成一个个
```

```
    //FormItem 对象。一个表单域中的数据对应于一个
```

```
    //FormItem 对象。
```

```
    List<FormItem> items =
```

```
        sfu.parseRequest(request);
```

//step4 遍历 items 集合

```
    for(int i=0;i<items.size();i++){
```

```
        FormItem item = items.get(i);
```

```
        //读表单域中的数据时，要区分表单域的类型
```

```
        if(item.isFormField()){
```

```
            //普通表单域
```

```
            String username = item.getString();
```

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

```
        }else{
```

```
            //文件上传表单域
```

```
            ServletContext sctx =
```

```
                getServletContext();
```

```
            String path = sctx.getRealPath("upload");
```

```
            //获得文件名
```

```
            String fileName = item.getName();
```

```
            File file = new File(path + "\\ " + fileName);
```

```
            item.write(file);
```

```
        }
```

```
    }
```

```
    } catch (Exception e) {
```

```
        e.printStackTrace();
```

```
    }
```

```
}
```

```
}
```

6) web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<web-app version="2.4"
```

```
    xmlns="http://java.sun.com/xml/ns/j2ee"
```

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```

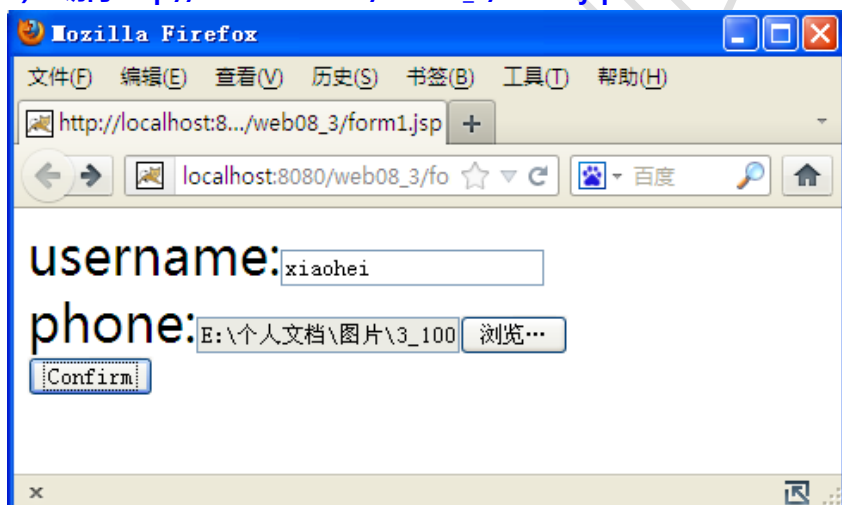
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
<servlet>
  <servlet-name>FileUploadServlet</servlet-name>
  <servlet-class>web.FileUploadServlet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>FileUploadServlet</servlet-name>
  <url-pattern>/fileupload</url-pattern>
</servlet-mapping>
</web-app>

```

7) 部署项目

8) 访问 http://localhost:8080/web08_3/form1.jsp



a. 选择文件并上传

b. 在 tomcat 服务器的目录下 ([D:\apache-tomcat-5.5.23\webapps\web08_3\upload](#)) 会显示你上传的内容