**实验十二 Java Web高级应用**

一、实验目的

1. 了解可以产生事件的对象；

2. 掌握ServletContext和HttpSession对象上的事件处理方法；

3. 了解在ServletRequest对象上的事件及处理方法；

4. 了解过滤器的作用；

5. 掌握过滤器的开发与部署的步骤；

6. 了解过滤器链。

二、实验原理

在Web应用程序中，事件的处理也是通过事件监听器接口处理的。Web应用事件处理的原理为：当Web应用中某些状态改变时，Servlet容器就产生某种事件，如ServletContext对象初始化时会产生ServletContextEvent事件，此时Servlet容器就会从注册的事件监听器中寻找处理该事件的监听器对象，并执行相应的代码。

在Servlet 2.4规范中共定义了6种事件类型和8个事件监听器接口，它们可以处理三种对象上的事件，如表12-1所示：

表12-1 监听器接口与事件类

|  |  |  |
| --- | --- | --- |
| 监听对象 | 监听器接口 | 监听事件 |
| ServletContext | ServletContextListener | ServletContextEvent |
| ServletContextAttributeListener | ServletContextAttributeEvent |
| HttpSession | HttpSessionListener | HttpSessionEvent |
| HttpSessionActivationListener |
| HttpSessionAttributeListener | HttpSessionBindingEvent |
| HttpSessionBindingListener |
| ServletRequest | ServletRequestListener | ServletRequestEvent |
| ServletRequestAttributeListener | ServletRequestAttributeEvent |

过滤器是web服务器上的组件，它们对客户和资源之间的请求和响应进行过滤。

过滤器的工作原理是：当servlet容器接收到对某个资源的请求，它要检查是否有过滤器与之关联。如果有过滤器与该资源关联，servlet容器将把该请求发送给过滤器。在过滤器处理完请求后，它将做下面3件事：

* 产生响应并将其返回给客户；
* 如果有过滤器链，它将把（修改过或没有修改过）请求传递给下一个过滤器；
* 将请求传递给不同的资源。

当请求返回到客户时，它是以相反的方向经过同一组过滤器返回。过滤器链中的每个过滤器够可能修改响应。

过滤器API主要包括：Filter、FilterConfig和FilterChain接口。

三、实验内容与步骤

**1、编写一个ServletContext事件监听器。该事件监听器的功能是当Web应用程序初始化和销毁时以及在ServletContext对象上添加属性、删除属性和替换属性时，在Tomcat日志中记录有关信息。**

【步骤1】编写监听器类

**（1）代码**

package com.demo;

import java.util.Date;

import javax.servlet.ServletContext;

import javax.servlet.ServletContextAttributeEvent;

import javax.servlet.ServletContextAttributeListener;

import javax.servlet.ServletContextEvent;

import javax.servlet.annotation.WebListener;

@WebListener

public class SCListener implements javax.servlet.ServletContextListener, ServletContextAttributeListener {

private ServletContext context = null;

public SCListener() {}

public void attributeAdded(ServletContextAttributeEvent scae) {

context = scae.getServletContext();

context.log("添加属性" + scae.getName() + ":" +scae.getValue());

}

public void attributeRemoved(ServletContextAttributeEvent scae) {

context = scae.getServletContext();

context.log("删除属性" + scae.getName() + ":" +scae.getValue());

}

public void contextDestroyed(ServletContextEvent sce) {

context = sce.getServletContext();

context.log("销毁：" + new Date());

}

public void attributeReplaced(ServletContextAttributeEvent scae) {

context = scae.getServletContext();

context.log("替换属性" + scae.getName() + ":" +scae.getValue());

}

public void contextInitialized(ServletContextEvent sce) {

context = sce.getServletContext();

context.log("初始化：" + new Date());

}

}

【步骤2】完成web.xml，实现注册监听器类。

**（2）代码**

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

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd" id="WebApp\_ID" version="3.1">

<display-name>ch12</display-name>

<listener>

<listener-class>com.demo.SCListener</listener-class>

</listener>

</web-app>

【步骤3】完成测试类EventTestServlet.java，实现添加属性（obj=100）、替换属性（obj=当前日期）和删除属性（obj）。

**（3）代码**

package com.demo;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Date;

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 EventTestServlet extends HttpServlet{

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

ServletContext context =null;

context = getServletConfig().getServletContext();

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

PrintWriter out = response.getWriter();

out.println("<html><head><title>EventTestServlet</title></head>");

context.setAttribute("obj",100); //添加属性

context.setAttribute("obj",new Date()); //替换属性

context.removeAttribute("obj"); //删除属性

out.println("<h3 align=center>已在ServletContext对象上添加、替换和删除了属性</h3>");

out.println("</body></html>");

out.close();

}

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doGet(request, response);

}

}

【步骤4】完成web.xml，实现注册Servlet。

**（4）代码**

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

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd" id="WebApp\_ID" version="3.1">

<display-name>ch12</display-name>

<listener>

<listener-class>com.demo.SCListener</listener-class>

</listener>

<servlet>

<servlet-name>EventTest</servlet-name>

<servlet-class>com.demo.EventTestServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>EventTest</servlet-name>

<url-pattern>/EventTestServlet</url-pattern>

</servlet-mapping>

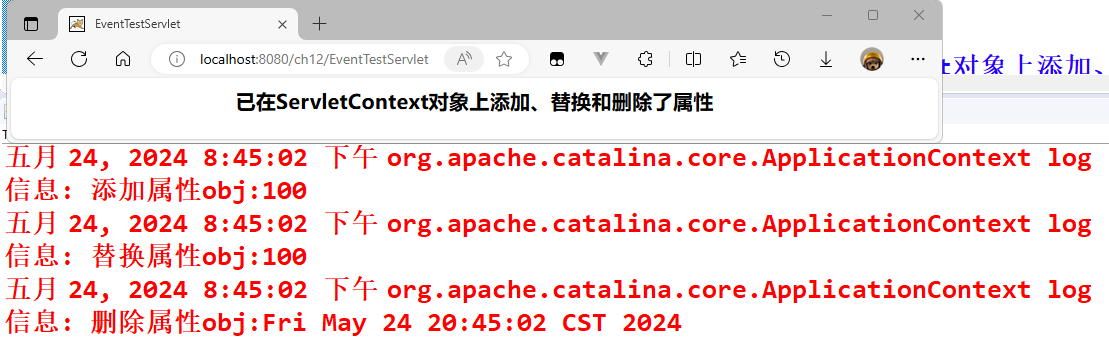
</web-app>

【步骤5】启动Tomcat，访问EventTestServlet。



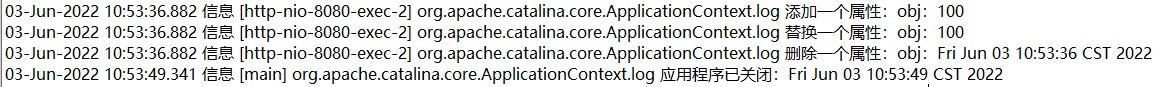
图12-1 EventTestServlet程序的运行结果

1. **运行截图**

****

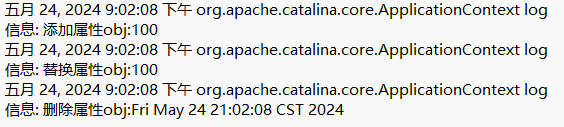
【步骤6】检查日志文件

打开<CATALINA\_HOME>\logs目录中的localhost.XXXXXX.log日志文件，在其中可以找到下面这些行。



上述信息是执行事件监听器后写到日志文件中的。

**（6）运行截图**



**2、编写HttpSession事件监听器用来记录当前在线人数。**

【步骤1】编写HttpSessionListener监听器类

**（1）代码**

package com.test2;

import javax.servlet.annotation.WebListener;

import javax.servlet.http.HttpSessionEvent;

import javax.servlet.http.HttpSessionListener;

@WebListener

public class HSListener implements HttpSessionListener {

private int count = 0;

public void sessionCreated(HttpSessionEvent se) {

count++;

se.getSession().getServletContext().setAttribute("count",count);

}

public void sessionDestroyed(HttpSessionEvent se) {

count--;

se.getSession().getServletContext().setAttribute("count",count);

}

}

【步骤2】完成web.xml，实现注册事件监听器。

**（2）代码**

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

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd" id="WebApp\_ID" version="3.1">

<display-name>ch12</display-name>

<listener>

<listener-class>com.test2.HSListener</listener-class>

</listener>

</web-app>

【步骤3】编写测试类OnlineCountServlet。

**（3）代码**

package com.test2;

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;

import javax.servlet.http.HttpSession;

public class OnlineCountServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

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

PrintWriter out = response.getWriter();

ServletContext application=request.getServletContext();

HttpSession session = request. getSession();

out.println("<html><head><title>OnlineCountServlet</title></head>");

out.println("<h2> 当前在线人数 ："+(Integer)application.getAttribute("count")+"</h2>");

out.println("</body></html>");

out.close();

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

}

【步骤4】完成web.xml文件，实现注册Servlet。

**（4）代码**

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

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd" id="WebApp\_ID" version="3.1">

<display-name>ch12</display-name>

<servlet>

<servlet-name>OnlineCount</servlet-name>

<servlet-class>com.test2.OnlineCountServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>OnlineCount</servlet-name>

<url-pattern>/OnlineCountServlet</url-pattern>

</servlet-mapping>

</web-app>

【步骤5】访问OnlineCountServlet

再启动两个浏览器窗口访问该Servlet，计数器就会增加，而点击浏览器的“刷新”按钮，计数器不会增加（因为没有开始新会话）。

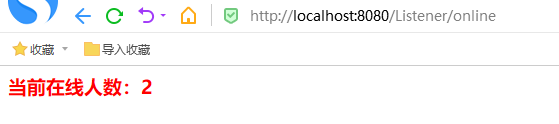
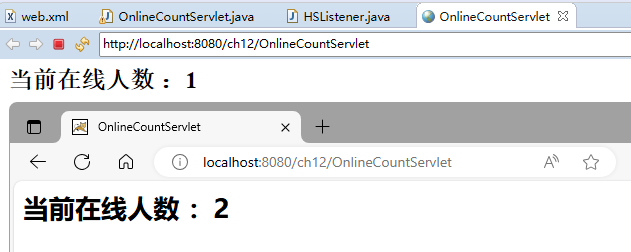


图12-2 OnlineCountServlet程序的运行结果

1. **运行截图**

****

**3、编写过滤器审计用户对资源的访问。**

【步骤1】完成AuditFilter.java过滤器，实现当用户访问应用程序任何资源时，将用户的IP地址和主机名写入日志文件中。

**（1）代码**

package com.test3;

import java.io.IOException;

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.annotation.WebFilter;

@WebFilter("/AuditFilter")

public class AuditFilter implements Filter {

public void destroy() {

}

public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {

String ip = request.getRemoteAddr();

String host = request.getRemoteHost();

request.getServletContext().log("用户IP地址："+ ip );

request.getServletContext().log("用户主机名："+ host );

chain.doFilter(request, response);

}

public void init(FilterConfig fConfig) throws ServletException {

}

}

【步骤2】在部署描述文件中配置过滤器，实现使用URL模式为/\*，将该过滤器映射到Web应用程序的任何资源。

**（2）代码**

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

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd" id="WebApp\_ID" version="3.1">

<display-name>ch12</display-name>

<filter>

<filter-name>AuditFilter</filter-name>

<filter-class>com.test3.AuditFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>AuditFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

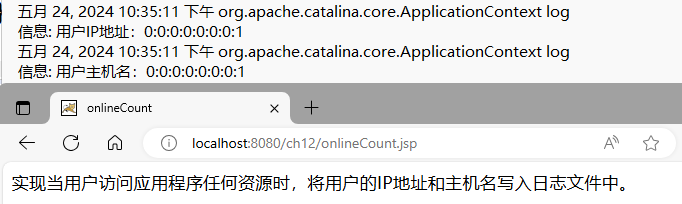
</web-app>

【步骤3】创建onlineCount.jsp后进行访问。

打开<CATALINA\_HOME>\logs目录中的localhost.XXXX.log文件中有下面一行（访问多个资源就会有多行）：



**（3）运行效果**



**4、编写过滤器实现改变请求编码。**

【步骤1】完成loginform.html，实现利用表单获取用户名和密码，将用户输入内容提交至由servlet/check映射的Servlet处理。



图12-3 loginform.html程序的运行结果

**（1）代码**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>loginform</title>

</head>

<body>

<h1>请输入用户名和口令：</h1>

<form action="/ch12/CheckParamServlet" method="post">

用户名：<input name="username" /> <br /> 口令：<input name="password" /> <br />

<input type="submit" /><input type="reset" />

</form>

</body>

</html>

【步骤2】完成CheckParamServlet.java，实现处理请求参数，并响应至前端显示用户名和口令。

**（2）代码**

package com.test4;

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 CheckParamServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

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

PrintWriter out = response.getWriter();

out.print("<!Doctype html><html><body>");

out.print("你的用户名为："+request.getParameter("username")+"<br/>");

out.print("你的口令为："+request.getParameter("password")+"<br/>");

out.print("</body></html>");

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

}

【步骤3】完成web.xml，实现注册Servlet。

**（3）代码**

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

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd" id="WebApp\_ID" version="3.1">

<display-name>ch12</display-name>

<servlet>

<servlet-name>CheckParam</servlet-name>

<servlet-class>com.test4.CheckParamServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>CheckParam</servlet-name>

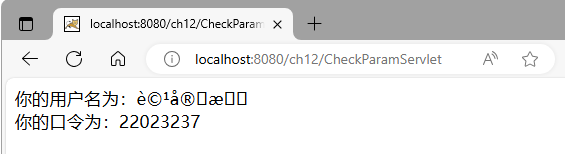
<url-pattern>/CheckParamServlet</url-pattern>

</servlet-mapping>

</web-app>

【步骤4】访问loginform.html。

**（4）运行截图**



点击“提交”按钮，经CheckParamServlet处理后返回的结果如下图所示：

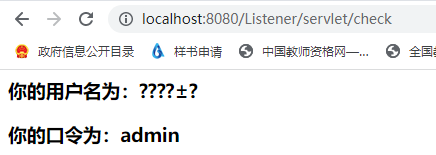


图12-4 CheckParamServlet程序的运行结果

可以看到，从服务器返回的汉字成了乱码。原因是没有指定request的编码。

接下来通过编写过滤器改变请求编码。

【步骤5】完成EncodingFilter.java，实现过滤器改变请求编码。

**（5）代码**

package com.test4;

import java.io.IOException;

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.annotation.WebFilter;

@WebFilter("/EncodingFilter")

public class EncodingFilter implements Filter {

private String encoding = "UTF-8"; // 设置编码为UTF-8

public void destroy() {

}

public void init(FilterConfig filterConfig) throws ServletException {

}

public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)

throws IOException, ServletException {

request.setCharacterEncoding(encoding);

response.setCharacterEncoding(encoding);

chain.doFilter(request, response);

}

}

【步骤6】完成web.xml，实现配置过滤器。

**（6）代码**

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

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd" id="WebApp\_ID" version="3.1">

<display-name>ch12</display-name>

<filter>

<filter-name>EncodingFilter</filter-name>

<filter-class>com.test4.EncodingFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>EncodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

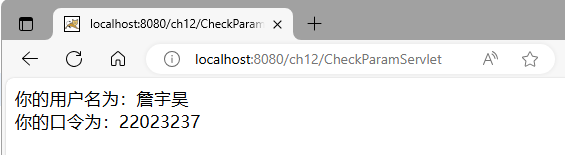
</web-app>

【步骤7】重复第（4）步操作，访问loginForm.html。



图12-5 EncodingFilter程序的运行结果

1. **运行截图**

****