**JavaEE第十三章 过滤器 监听器**

1. **过滤器**
2. **是JavaEE的Web组件之一**
3. **用来过滤发往Servlet的请求**

WEB服务器

被请求的资源

过

滤

器

请求

响应



**二．功能实现**

**1.Log4j记录日志：**

public class Log4jFilter implements Filter

{

    @Override

    public void doFilter(ServletRequest arg0, ServletResponse arg1,FilterChain arg2) throws IOException, ServletException

    {

           Logger log = Logger.getLogger(Log4jFilter.class);

           HttpServletRequest hsr = (HttpServletRequest)arg0;

           log.debug("客户IP："+ hsr.getRemoteAddr()+",访问了"+hsr.getRequestURL());

           arg2.doFilter(arg0, arg1);

    }

}

**Web.xml配置文件:**

<filter>

<filter-name>log4jFilter</filter-name>

<filter-class>com.igeek.peakedness.filter.Log4jFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>log4jFilter</filter-name>

<url-pattern>/TestServlet</url-pattern>

</filter-mapping>

**2.权限过滤器：**

@WebFilter("/PrettyServlet")

public class PrivacyFilter implements Filter {

/\*\*

\* @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)

\*/

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

// TODO Auto-generated method stub

// place your code here

    HttpServletRequest hsr = (HttpServletRequest)request;

    User user = (User)hsr.getSession().getAttribute("user");

    if(user == null) {

        hsr.getRequestDispatcher("Login.jsp").forward(hsr, (HttpServletResponse)response);

        return;

    }

chain.doFilter(request, response);

}

}

**3.Spring编码过滤器:**

<filter>

    <filter-name>characterEncodingFilter</filter-name>

    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

    <init-param>

      <param-name>encoding</param-name>

      <param-value>UTF-8</param-value>

    </init-param>

    <init-param>

      <param-name>forceEncoding</param-name>

      <param-value>true</param-value>

    </init-param>

  </filter>

  <filter-mapping>

    <filter-name>characterEncodingFilter</filter-name>

    <url-pattern>/\*</url-pattern>

  </filter-mapping>

1. **监听器**
2. **Servlet监听器也是Web应用程序事件模型的一部分**
3. **用来监听三大作用域（ServletContext,Session,Request）的状态或值，发生改变时触发的事件**

**SessionListener例子:**

**接口HttpSessionListener监听Session创建和销毁**

**接口HttpSessionAttributeListener监听Session创建，修改，删除参数**

@WebListener

public class SessionListener implements HttpSessionListener, HttpSessionAttributeListener {

    /\*\*

     \* Default constructor.

     \*/

    public SessionListener() {

        // TODO Auto-generated constructor stub

    }

/\*\*

     \* @see HttpSessionListener#sessionCreated(HttpSessionEvent)

     \*/

    public void sessionCreated(HttpSessionEvent se)  {

         // TODO Auto-generated method stub

        System.out.println("创建session");

    }

/\*\*

     \* @see HttpSessionListener#sessionDestroyed(HttpSessionEvent)

     \*/

    public void sessionDestroyed(HttpSessionEvent se)  {

         // TODO Auto-generated method stub

        System.out.println("销毁session");

    }

/\*\*

     \* @see HttpSessionAttributeListener#attributeAdded(HttpSessionBindingEvent)

     \*/

    public void attributeAdded(HttpSessionBindingEvent se)  {

         // TODO Auto-generated method stub

        System.out.println("Session添加参数，参数名："+se.getName()+",参数值:"+se.getValue());

    }

/\*\*

     \* @see HttpSessionAttributeListener#attributeRemoved(HttpSessionBindingEvent)

     \*/

    public void attributeRemoved(HttpSessionBindingEvent se)  {

         // TODO Auto-generated method stub

        System.out.println("Session删除参数，参数名："+se.getName());

    }

/\*\*

     \* @see HttpSessionAttributeListener#attributeReplaced(HttpSessionBindingEvent)

     \*/

    public void attributeReplaced(HttpSessionBindingEvent se)  {

         // TODO Auto-generated method stub

        System.out.println("Session修改参数，参数名："+se.getName()+",参数值:"+se.getValue());

    }

}