

Librerías en el managedbean

import ec.mil.he1.he1.modelo.seguridad.VMenuPrincipal;

import ec.mil.he1.he1.remotoseguridad.SegArbolMenuRemote;

import java.io.IOException;

import java.io.Serializable;

import java.util.List;

import javax.ejb.EJB;

import javax.faces.context.ExternalContext;

import javax.faces.context.FacesContext;

import javax.faces.event.ActionEvent;

import org.primefaces.component.commandlink.CommandLink;

Las variables dentro del managedbean

@EJB

private SegArbolMenuRemote segArbolMenu;

String usu = "1238";

String URL = "";

Los métodos que jalan el árbol

public List<VMenuPrincipal> getListagrupomenu() {

return segArbolMenu.listGrupoByUsuId(usu);

}

public List<VMenuPrincipal> getListagrupomenu(String gruid) {

return segArbolMenu.listSistemasByGruId(usu, gruid);

}

public List<VMenuPrincipal> getListamenu1(String sisid) {

return segArbolMenu.listMenuNivel1ByGruId(usu, sisid);

}

public List<VMenuPrincipal> getListamenu2(String opcid) {

return segArbolMenu.listMenuNivel2ByOpcNivel(usu, opcid);

}

public String lnkRedirect() throws IOException {

return URL;

}

public void lnkActionEvent(ActionEvent actionEvent) throws IOException {

CommandLink component = (CommandLink) actionEvent.getComponent();

Object value = component.getValue();

List<VMenuPrincipal> listOpcionByDescripcion = segArbolMenu.listOpcionByDescripcion(value.toString());

for (VMenuPrincipal listOpcionByDescripcion1 : listOpcionByDescripcion) {

URL = listOpcionByDescripcion1.getCOpcUrlSubmenu();

}

//si esta en un contexto o página externa se redireciona desde aquí

//caso contratio se puede enviar en el action

if (URL.substring(0, 5).equalsIgnoreCase("http:")) {

ExternalContext ec = FacesContext.getCurrentInstance().getExternalContext();

ec.redirect(URL);

}

}

En el HTML

xmlns:h="http://xmlns.jcp.org/jsf/html" xmlns:p="http://primefaces.org/ui" xmlns:c="http://xmlns.jcp.org/jsp/jstl/core"

En el body

<h:form>

<p:growl id="msgs" showDetail="true" escape="false"/>

<h:panelGrid style="font-size: 10px; width: 33%" >

<p:panelMenu style="font-size: 16px;" >

<c:forEach items="#{dibujaArbolController.listagrupomenu}" var="grupos" >

<p:submenu label="#{grupos.CGruDescripcion}" style="font-size: 15px;" >

<c:forEach items="#{dibujaArbolController.getListagrupomenu( grupos.CGruId)}" var="sistemas" >

<p:submenu label="#{sistemas.CGruDescripcionSistema}" style="font-size: 13px;" >

<c:forEach items="#{dibujaArbolController.getListamenu1( sistemas.CGruIdSistema )}" var="menu1" >

<p:submenu label="#{menu1.COpcNombreOpcion}" style="font-size: 10px;" >

<c:forEach items="#{dibujaArbolController.getListamenu2( menu1.COpcId )}" var="menu2" >

<p:menuitem style="width: 100%">

<p:commandLink id="cmlMenuOpciones"

style="font-size: 9px;"

action="#{dibujaArbolController.lnkRedirect()}"

value="#{menu2.COpcNombreOpcionSubmenu}"

actionListener="#{dibujaArbolController.lnkActionEvent}"/>

</p:menuitem>

</c:forEach>

</p:submenu>

</c:forEach>

</p:submenu>

</c:forEach>

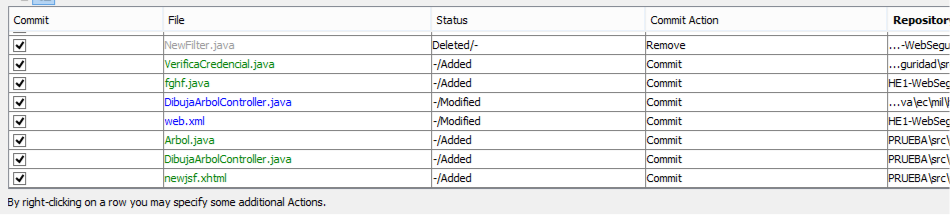
</p:submenu>

</c:forEach>

</p:panelMenu>

</h:panelGrid>

</h:form>



Filtro

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package ec.mil.he1.he1.servlet;

import ec.mil.he1.he1.modelo.seguridad.SegUsuario;

import ec.mil.he1.he1.remotoseguridad.SegUsuarioFacadeRemote;

import java.io.IOException;

import java.io.PrintStream;

import java.io.PrintWriter;

import java.io.StringWriter;

import java.math.BigDecimal;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.annotation.security.PermitAll;

import javax.naming.Context;

import javax.naming.InitialContext;

import javax.naming.NamingException;

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;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

/\*\*

\*

\* @author christian\_ruiz

\*/

@WebFilter(filterName = "NewFilter", urlPatterns = {"/\*"})

@PermitAll

public class FiltroSeguridad implements Filter {

SegUsuarioFacadeRemote segUsuarioFacade = lookupSegUsuarioFacadeRemote();

private static final boolean debug = true;

// The filter configuration object we are associated with. If

// this value is null, this filter instance is not currently

// configured.

private FilterConfig filterConfig = null;

public FiltroSeguridad() {

}

private void doBeforeProcessing(ServletRequest request, ServletResponse response)

throws IOException, ServletException {

if (debug) {

log("NewFilter:DoBeforeProcessing");

}

// Write code here to process the request and/or response before

// the rest of the filter chain is invoked.

// For example, a logging filter might log items on the request object,

// such as the parameters.

String parameter = request.getParameter("dhBvFABCCCSSASDASDASDAGGDDujGTR");

log("Desde filtro parameter---------------- = " + parameter);

}

private void doAfterProcessing(ServletRequest request, ServletResponse response)

throws IOException, ServletException {

try {

HttpSession session = ((HttpServletRequest) request).getSession();

if (session.getAttribute("personaLogin") == null) {

String parameter = request.getParameter("dhBvFABCCCSSASDASDASDAGGDDujGTR");

SegUsuario personaLogin = lookupSegUsuarioFacadeRemote().find(new BigDecimal(lookupSegUsuarioFacadeRemote().decrypta\_dinamico(parameter)));

session.setAttribute("personaLogin", personaLogin);

}

} catch (Exception e) {

log("" + e.getLocalizedMessage());

}

}

/\*\*

\*

\* @param request The servlet request we are processing

\* @param response The servlet response we are creating

\* @param chain The filter chain we are processing

\*

\* @exception IOException if an input/output error occurs

\* @exception ServletException if a servlet error occurs

\*/

public void doFilter(ServletRequest request, ServletResponse response,

FilterChain chain)

throws IOException, ServletException {

if (debug) {

log("NewFilter:doFilter()");

}

doBeforeProcessing(request, response);

Throwable problem = null;

try {

chain.doFilter(request, response);

} catch (Throwable t) {

// If an exception is thrown somewhere down the filter chain,

// we still want to execute our after processing, and then

// rethrow the problem after that.

problem = t;

t.printStackTrace();

}

doAfterProcessing(request, response);

// If there was a problem, we want to rethrow it if it is

// a known type, otherwise log it.

if (problem != null) {

if (problem instanceof ServletException) {

throw (ServletException) problem;

}

if (problem instanceof IOException) {

throw (IOException) problem;

}

sendProcessingError(problem, response);

}

}

/\*\*

\* Return the filter configuration object for this filter.

\*/

public FilterConfig getFilterConfig() {

return (this.filterConfig);

}

/\*\*

\* Set the filter configuration object for this filter.

\*

\* @param filterConfig The filter configuration object

\*/

public void setFilterConfig(FilterConfig filterConfig) {

this.filterConfig = filterConfig;

}

/\*\*

\* Destroy method for this filter

\*/

public void destroy() {

}

/\*\*

\* Init method for this filter

\*/

public void init(FilterConfig filterConfig) {

this.filterConfig = filterConfig;

if (filterConfig != null) {

if (debug) {

log("NewFilter:Initializing filter");

}

}

}

/\*\*

\* Return a String representation of this object.

\*/

@Override

public String toString() {

if (filterConfig == null) {

return ("NewFilter()");

}

StringBuffer sb = new StringBuffer("NewFilter(");

sb.append(filterConfig);

sb.append(")");

return (sb.toString());

}

private void sendProcessingError(Throwable t, ServletResponse response) {

String stackTrace = getStackTrace(t);

if (stackTrace != null && !stackTrace.equals("")) {

try {

response.setContentType("text/html");

PrintStream ps = new PrintStream(response.getOutputStream());

PrintWriter pw = new PrintWriter(ps);

pw.print("<html>\n<head>\n<title>Error</title>\n</head>\n<body>\n"); //NOI18N

// PENDING! Localize this for next official release

pw.print("<h1>The resource did not process correctly</h1>\n<pre>\n");

pw.print(stackTrace);

pw.print("</pre></body>\n</html>"); //NOI18N

pw.close();

ps.close();

response.getOutputStream().close();

} catch (Exception ex) {

}

} else {

try {

PrintStream ps = new PrintStream(response.getOutputStream());

t.printStackTrace(ps);

ps.close();

response.getOutputStream().close();

} catch (Exception ex) {

}

}

}

public static String getStackTrace(Throwable t) {

String stackTrace = null;

try {

StringWriter sw = new StringWriter();

PrintWriter pw = new PrintWriter(sw);

t.printStackTrace(pw);

pw.close();

sw.close();

stackTrace = sw.getBuffer().toString();

} catch (Exception ex) {

}

return stackTrace;

}

public void log(String msg) {

filterConfig.getServletContext().log(msg);

}

private SegUsuarioFacadeRemote lookupSegUsuarioFacadeRemote() {

try {

Context c = new InitialContext();

return (SegUsuarioFacadeRemote) c.lookup("java:global/HE1-ServicioSeguridad/SegUsuarioFacade");

} catch (NamingException ne) {

Logger.getLogger(getClass().getName()).log(Level.SEVERE, "exception caught", ne);

throw new RuntimeException(ne);

}

}

}