

SSOEnablerBean

```

package sso;
/*
 * SSOEnablerBean.java
 *
 * Copyright 1999-2001 by Oracle Corporation
 * 500 Oracle Parkway, Redwood Shores, California, 94065, U. S. A.
 * All rights reserved.
 *
 * This software is the confidential and proprietary information
 * of Oracle Corporation ("Confidential Information").
 * You shall not disclose such Confidential Information and shall
 * use it only in accordance with the terms of the license agreement
 * you entered into with Oracle Corporation.
 */

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.Cookie;

import java.net.URL;
import java.net.InetAddress;

import java.sql.DriverManager;
import java.sql.Connection;

import oracle.jdbc.pool.OracleConnectionCacheImpl;

import oracle.security.sso.enabler.SSOEnabler;
import oracle.security.sso.enabler.SSOUserInfo;
import oracle.security.sso.enabler.SSOEnablerUtil;
import oracle.security.sso.enabler.SSOEnablerException;

public class SSOEnablerBean
{
    private String m_listenerToken    = null;
    private String m_requestedUrl      = null;
    private String m_onCancelUrl       = null;

    private String m_pappCookieName    = null;
    private String m_pappCookieDomain = null;
    private String m_pappCookieScope   = null;

    private OracleConnectionCacheImpl m_connCache = null;

    /**
     * Default constructor
     */
    public SSOEnablerBean()
    {
    }

    /**
     * Set listener token
     */
    public void setListenerToken(String p_listenerToken)
    {
        m_listenerToken = p_listenerToken;
    }

    /**
     * Set requested and cancel url
     */
    public void setUrls(String p_requestedUrl, String p_cancelUrl)
    {
        m_requestedUrl = p_requestedUrl;
        m_onCancelUrl  = p_cancelUrl;
    }
}

```

```

SS0Enabl erBean

/**
 * Set applicati on cooki e informati on
 */
public void setAppCooki eInfo(String p_name, String p_domain, String
p_path)
{
    m_pappCooki eName      = p_name;
    m_pappCooki eDomai n   = p_domai n;
    m_pappCooki eScope     = p_path;
}

public void setDbConnecti onInfo(String p_schema , String p_password,
String p_hostname, int p_port, String p_sid, int p_dbPool Si ze)
{
    try
    {
        m_connCache = new Oracl eConnecti onCacheI mpl ();
        // m_connCache. setURL("j dbc: oracl e: oci 8: @");

        Class. forName("oracl e. j dbc. dri ver. Oracl eDri ver");
        m_connCache. setURL("j dbc: oracl e: thi n: @"
            + p_hostname + ":" + p_port + ":" + p_sid );

        m_connCache. setUser(p_schema);
        m_connCache. setPassword(p_password);

        m_connCache. setMaxLi mi t(p_dbPool Si ze);

        // aggi unto da L. S.

m_connCache. setCacheScheme(Oracl eConnecti onCacheI mpl . FI XED_WAI T_SCHEME);

    }
    catch(Excepti on e)
    {
        m_connCache = null ;
    }
}

/**
 * This method will return SS0 user informati on. If the user is not
authenticat ed against
 * SS0 server then it will redirect user to the SS0 Server for
authenticati on
 */
public String getSS0UserI nfo(HttpServletRequest p_request,
HttpServletResponse p_response)
    throws SS0Enabl erExcepti on
{
    String l_userName = null ;

    if(p_response == null || p_response == null )
    {
        throw new SS0Enabl erExcepti on("Http obj ects are null");
    }

    if(m_li stenerToken == null )
    {
        throw new SS0Enabl erExcepti on("Li stener token is null");
    }

    if(m_requeste dUrl == null || m_onCancel Url == null )
    {
        throw new SS0Enabl erExcepti on("Requeste d url and cancel
url must be set");
    }

    try

```

```

        SS0EnablerBean
    {
        // Get database connection
        Connection l_db_con = m_connCache.getConnection();

        // Try to get user information from application cookie
        l_userName = getUserInfo(p_request);
        System.out.println("ATTENZIONE: "+ l_userName);
        if(l_userName == null)
        {
            // Create SS0Enabler object
            SS0Enabler l_ssoEnabler = new
SS0Enabler(l_db_con);
            // Create redirect url to the SS0 server for
            user authentication
            String l_redirectUrl =

l_ssoEnabler.generateRedirect(m_listenerToken, m_requestedUrl, m_onCancelUrl);
            // close database connection
            l_db_con.close();

            // p_response.sendRedirect(l_redirectUrl);

            // Since the redirect url is usually large so
            send the redirect url input
            // parameters using HTTP post method instead of
            usual GET method of
            // HttpServletResponse.sendRedirect
            //String htmlPostForm =
SS0EnablerUtil.genHtmlPostForm(l_redirectUrl);
            //System.out.println(htmlPostForm);
            //p_response.getWriter().println(htmlPostForm);
            p_response.sendRedirect(l_redirectUrl);

            return null;
        }
        else
        {
            // We got this user information from application
            cookie
            SS0EnablerUtil l_ssoAppUtil = new
SS0EnablerUtil(l_db_con);
            return
l_ssoAppUtil.unbakeAppCookie(m_listenerToken, l_userName);
        }
        catch(Exception e)
        {
            throw new SS0EnablerException(e.toString());
        }
    }

/**
 * Get user information from application cookie
 */
private String getUserInfo(HttpServletRequest p_request)
    throws SS0EnablerException
{
    boolean l_gotPappCookie = false;
    String l_userInfo = null;

    if(m_pappCookieName == null)
        throw new SS0EnablerException("Cookie name is null");
    try
    {
        Cookie[] l_cookies = p_request.getCookies();
        for(int i=0; i < l_cookies.length; i++)
        {
            Cookie l_pappCookie = l_cookies[i];

```

```

        SS0EnablerBean
        if
        (l_pappCookie.getName().equals(m_pappCookieName))
        {
            l_gotPappCookie = true;
            l_userInfo =
        l_pappCookie.getValue();
        if (l_userInfo.equalsIgnoreCase("logout")){l_userInfo = null;}
            break;
        }
    }
    catch(Exception e)
    {
        return null;
    }

    if( (l_userInfo != null) && (l_userInfo.length() > 0) )
    {
        if (l_userInfo.equalsIgnoreCase("logout")){l_userInfo = null;}
        System.out.println("ATT2: " + l_userInfo);
        return l_userInfo;
    }
    else
    {
        return null;
    }
}

/**
 * This method will set application cookie from SS0 server token and
then redirect
 * user to the application
 */
public void setPartnerAppCookie(HttpServletRequest p_request,
HttpServletResponse p_response)
    throws SS0EnablerException
{
    if(p_response == null || p_request == null)
    {
        throw new SS0EnablerException("Http objects are null");
    }

    if(m_listenerToken == null)
    {
        throw new SS0EnablerException("Listener token is null");
    }

    if( m_pappCookieName == null
        || m_pappCookieDomain == null
        || m_pappCookieScope == null)
    {
        throw new SS0EnablerException("Application cookie
information is not available");
    }

    SS0UserInfo l_ssoUserInfo = null;
    try
    {
        String l_urlParam = p_request.getParameterValues("urlc")[0];
        if(l_urlParam != null)
        {
            // Get database connection
            Connection l_db_con = m_connCache.getConnection();

            // Create SS0Enabler object
            SS0Enabler l_ssoEnabler = new SS0Enabler(l_db_con);

            // Get IP address of the client

```

```

        SSOEnablerBean
        InetAddress l_clientIp =
InetAddress.getByName(p_request.getRemoteAddr());
        l_ssoUserInfo =
l_ssoEnabler.getSSOUserInfo(m_listenerToken, l_urlParam, l_clientIp);

        // Set application cookie
SSOEnablerUtil l_ssoAppUtil = new
SSOEnablerUtil(l_db_con);
        String l_bakedAppCookie =
        l_ssoAppUtil.bakeAppCookie(m_listenerToken,
l_ssoUserInfo.getSSOUserName());
        // Close database connection
l_db_con.close();

        // Create application cookie and set it
        // ** IMPORTANT **
        // Time stamp **must** be added in this cookie and
should implement
in-activity etc.
        // application cookie time out based on user
Cookie l_AppCookie = new Cookie(m_pappCookieName,
l_bakedAppCookie);
        //l_AppCookie.setDomain(m_pappCookieDomain);
        // In-memory cookie for better security
l_AppCookie.setMaxAge(-1);
        l_AppCookie.setPath(m_pappCookieScope);
p_response.addCookie(l_AppCookie);

        String reqRedirectHtmlStr =
SSOEnablerUtil.genRedirect(l_ssoUserInfo.getUrlRequested());

        p_response.getWriter().println(reqRedirectHtmlStr);
    }
    else
    {
        throw new SSOEnablerException("SSO server returned
null user information");
    }
}
catch(Exception e)
{
    throw new SSOEnablerException(e.toString());
}
}

/**
 * Remove application cookie to end user application session
 */
public void removeAppCookie(HttpServletResponse p_response)
    throws SSOEnablerException
{
    if(p_response == null)
    {
        throw new SSOEnablerException("HttpServletResponse is
null");
    }

    if( m_pappCookieName == null
        || m_pappCookieDomain == null
        || m_pappCookieScope == null )
    {
        throw new SSOEnablerException("Application cookie
information is not available");
    }

    Cookie l_AppCookie = new Cookie(m_pappCookieName, "Logout");
    //l_AppCookie.setDomain(m_pappCookieDomain);
    l_AppCookie.setMaxAge(0);

```

```

                                SS0Enabl erBean
l_AppCooki e. setPath(m_pappCooki eScope);
p_response. addCooki e(l_AppCooki e);
}

public void close()
{
    try
    {
        m_connCache. cl ose();
    }
    catch(Excepti on e)
    {
    }
}

}

```