```
package sso;
/**
   This class has been derived for CONSIP from
   SSOEnablerServletBean
*/

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

/* added for CONSIP */
import java.sql.DriverManager;
import java.sql.*;
import oracle.jdbc.pool.OracleConnectionCacheImpl;
/* end added */

import oracle.security.sso.enabler.SSOEnablerException;

public class Test_SSO
{
   /** Start configuration parameters
        *
        */

        // Listener token for this partner application name
        private static String m_listenerToken  = "";
        private static String m_listenerToken_e  = "";
        // Partner application  session cookie name
        private static String m_cookieName     = "" ;
        private static String m_cookieName_e     = "" ;
        // Partner application  session domain
        private static String m_cookieDomain   = "" ;
        // Partner application  session path scope
        private static String m_cookiePath     = "/";

        // Host name of the database
        private static String m_dbHostName     = "" ;
        // Port for database
        private static int    m_dbPort         = 0 ;
        // Sehema name
        private static String m_dbSchemaName   = "" ;
        // Schema password
        private static String m_dbSchemaPasswd = "" ;
        // Database SID name
        private static String m_dbSID          = "" ;
        // Database connection pool size
        private static int    m_dbPoolSize     = 10;

        // Requested URL (User requested page)
        private static String m_requestUrl     = "";
        private static String m_requestUrl_e     = "";
        // Cancel URL(Home page for this application which don't require authentication)
        private static String m_cancelUrl      = "" ;
        private static String m_cancelUrl_e      = "" ;
        /** customization added for CONSIP */
        /** GRP stand for groups... custom code has been added
           for group identification                          */

        private static String m_dbHostNameGRP     = "" ;
        // Port for database
        private static int    m_dbPortGRP         = 0 ;
        // Sehema name
        private static String m_dbSchemaNameGRP   = "" ;
        // Schema password
        private static String m_dbSchemaPasswdGRP = "" ;
        // Database SID name
        private static String m_dbSIDGRP          = "" ;
        // Database connection pool size
        private static int    m_dbPoolSizeGRP     = 10;

        private OracleConnectionCacheImpl  m_connCacheGRP = null;
        /** end of customization for CONSIP */

        /** End of configuration parameters */

        // Enabler object (Don't change)
        private sso.SSOEnablerBean  m_enablerBean = null;
  public static String logout_url = null;
  public static String logout_url_ext = null;
//public static String from = null;
   /**
        *  Default constructor
        */
        public Test_SSO(String from)
        {
```

```
		// PF

		java.util.PropertyResourceBundle prb =
(java.util.PropertyResourceBundle)java.util.PropertyResourceBundle.getBundle("sso.portal");

		for (java.util.Enumeration enum = prb.getKeys() ; enum.hasMoreElements() ; ) {

			String chiave = (String) enum.nextElement() ;
			if (chiave.equals("username")) {
				sso.Test_SSO.m_dbSchemaNameGRP = (String)prb.getString(chiave) ;
			}
			if (chiave.equals("password")) {
				sso.Test_SSO.m_dbSchemaPasswdGRP = (String)prb.getString(chiave) ;
			}
			if (chiave.equals("host")) {
				sso.Test_SSO.m_dbHostNameGRP = (String)prb.getString(chiave) ;
			}
			if (chiave.equals("port")) {
				sso.Test_SSO.m_dbPortGRP =
Integer.parseInt((String)prb.getString(chiave)) ;
			}
			if (chiave.equals("sid")) {
				sso.Test_SSO.m_dbSIDGRP = (String)prb.getString(chiave) ;
			}
		}

		java.util.PropertyResourceBundle prb2 =
(java.util.PropertyResourceBundle)java.util.PropertyResourceBundle.getBundle("sso.login");

		for (java.util.Enumeration enum2 = prb2.getKeys() ; enum2.hasMoreElements() ; ) {

			String chiave = (String) enum2.nextElement() ;
			if (chiave.equals("username")) {
				sso.Test_SSO.m_dbSchemaName = (String)prb2.getString(chiave) ;
			}
			if (chiave.equals("password")) {
				sso.Test_SSO.m_dbSchemaPasswd = (String)prb2.getString(chiave) ;
			}
			if (chiave.equals("host")) {
				sso.Test_SSO.m_dbHostName = (String)prb2.getString(chiave) ;
			}
			if (chiave.equals("port")) {
				sso.Test_SSO.m_dbPort =
Integer.parseInt((String)prb2.getString(chiave)) ; ;
			}
			if (chiave.equals("sid")) {
				sso.Test_SSO.m_dbSID = (String)prb2.getString(chiave) ;
			}
			if (chiave.equals("cookie_name")) {
				sso.Test_SSO.m_cookieName = (String)prb2.getString(chiave) ;
			}
			if (chiave.equals("cookie_name_ext")) {
				sso.Test_SSO.m_cookieName_e = (String)prb2.getString(chiave) ;
			}
			if (chiave.equals("cookie_domain")) {
				sso.Test_SSO.m_cookieDomain = (String)prb2.getString(chiave) ;
			}
			if (chiave.equals("cookie_path")) {
				sso.Test_SSO.m_cookiePath = (String)prb2.getString(chiave) ;
			}
			if (chiave.equals("request_url")) {
				sso.Test_SSO.m_requestUrl = (String)prb2.getString(chiave) ;
			}
			if (chiave.equals("cancel_url")) {
				sso.Test_SSO.m_cancelUrl = (String)prb2.getString(chiave) ;
			}
			if (chiave.equals("request_url_ext")) {
				sso.Test_SSO.m_requestUrl_e = (String)prb2.getString(chiave) ;
			}
			if (chiave.equals("cancel_url_ext")) {
				sso.Test_SSO.m_cancelUrl_e = (String)prb2.getString(chiave) ;
			}
			if (chiave.equals("listener_token")) {
				sso.Test_SSO.m_listenerToken = (String)prb2.getString(chiave) ;
			}
			if (chiave.equals("listener_token_ext")) {
				sso.Test_SSO.m_listenerToken_e = (String)prb2.getString(chiave) ;
			}
			if (chiave.equals("logout_url")) {
				sso.Test_SSO.logout_url = (String)prb2.getString(chiave) ;
			}
			if (chiave.equals("logout_url_ext")) {
				sso.Test_SSO.logout_url_ext = (String)prb2.getString(chiave) ;
			}
		}
```

```
                //    fine PF
                m_enablerBean = new sso.SSOEnablerBean();
                if (from.equalsIgnoreCase("I")) {
                    m_enablerBean.setListenerToken(m_listenerToken);
                    m_enablerBean.setUrls(m_requestUrl, m_cancelUrl);
                    m_enablerBean.setAppCookieInfo(m_cookieName, m_cookieDomain, m_cookiePath);
        }else{
                    m_enablerBean.setListenerToken(m_listenerToken_e);
                    m_enablerBean.setUrls(m_requestUrl_e, m_cancelUrl_e);
                    m_enablerBean.setAppCookieInfo(m_cookieName_e, m_cookieDomain, m_cookiePath);
             }
    m_enablerBean.setDbConnectionInfo(m_dbSchemaName, m_dbSchemaPasswd, m_dbHostName , m_dbPort ,
m_dbSID, m_dbPoolSize);

                /* added for CONSIP: set connection info to retrieve
                   group information */
                setDbConnectionInfoGRP(m_dbSchemaNameGRP, m_dbSchemaPasswdGRP,
                                                          m_dbHostNameGRP , m_dbPortGRP ,
m_dbSIDGRP, m_dbPoolSizeGRP);
        }

        /* added for CONSIP */
        public void setDbConnectionInfoGRP(String p_schema , String p_password,
                String p_hostname, int p_port, String p_sid, int p_dbPoolSize)
        {
                try
                {
                   m_connCacheGRP = new OracleConnectionCacheImpl();

                   Class.forName("oracle.jdbc.driver.OracleDriver");

                   m_connCacheGRP.setURL("jdbc:oracle:thin:@"
                                + p_hostname + ":" + p_port + ":" + p_sid );

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

                   m_connCacheGRP.setMaxLimit(p_dbPoolSize);


                   m_connCacheGRP.setCacheScheme(OracleConnectionCacheImpl.FIXED_WAIT_SCHEME);
                }
                catch(Exception e)
                {
                   m_connCacheGRP = null;
                }
        }

        public String getUserGroups(String usr) throws Exception
        {
           // borrow a DB connection from ConnPool
           Connection l_db_con = m_connCacheGRP.getConnection();

           // code has been written in order to use BIND variables...
           // It will be executed many times !!... avoid filling shared pool
           String ids = "<br>";
           String sIstr = "SELECT NAME FROM PORTAL30.WWSEC_GROUP$ A, PORTAL30.WWSEC_MEMBER$ B,
PORTAL30.WWSEC_PERSON$ C " +
                                          "WHERE C.USER_NAME = :1 AND C.ID = B.MEMBER_PERSON_ID " +
                                          "AND B.GROUP_ID = A.ID ORDER BY NAME ASC";

           // use it
           PreparedStatement stmt = l_db_con.prepareStatement(sIstr);

           stmt.setString(1, usr);

           ResultSet rSet = stmt.executeQuery();

           while (rSet.next())
           {
                  ids = ids + rSet.getString(1);
                  ids = ids + "<br>";
           }

           // release it
           rSet.close();
           stmt.close();

           l_db_con.close();

           return ids;
        }

        /* end of methods added for CONSIP */
```

```
        public String getSSOUserInfo(HttpServletRequest p_request, HttpServletResponse p_response)
                throws SSOEnablerException
        {
                return m_enablerBean.getSSOUserInfo(p_request, p_response);
        }

        public void setPartnerAppCookie(HttpServletRequest p_request, HttpServletResponse
p_response)
                throws SSOEnablerException
        {
                m_enablerBean.setPartnerAppCookie(p_request, p_response);
        }

        public void removeServletAppCookie(HttpServletResponse p_response)
                throws SSOEnablerException
        {
                m_enablerBean.removeAppCookie(p_response);
        }
}
```