AD域登录示例

使用AD域登录示例，请先查看 《与第三方系统用户集成》的第 4、使用第三方系统的用户验证

AD域登录示例

```
package smartbi.usermanager.auth.impl;
import java.io.IOException;
import java.io.InputStream;
import java.util.*;
import javax.naming.*;
import javax.naming.directory.*;
import javax.naming.ldap.InitialLdapContext;
import javax.naming.ldap.LdapContext;
import javax.servlet.ServletContext;
import org.apache.log4j.Logger;
import smartbi.usermanager.auth.*;
import smartbi.util.StringUtil;
/**
 * MicrosoftAD<br>
 * ad.propertieswarWEB-INF<br>
 *
 * <pre>
 * #
 *                 initial_context_factory=com.sun.jndi.ldap.LdapCtxFactory
 * #
 *         provider_url=ldap://ADServer:3268
 * #
 *                 principal_prefix=MyDomain\\
 * #
 *         login_user=username
 * #
 *         login_password=password
 * </pre>
 */
public class ADAuthentication implements IAuthentication {
        private static final Logger log = Logger.getLogger(ADAuthentication.class);
        private String initialContextFactory;
        private String providerUrl;
        private String prefix;
        private String loginUser;
        private String loginPassword;
        private String baseName;
        private String filterPrefix;
        private String filterSuffix;
        /**
         *
         *
         * @param ctx
         *
         * @throws IOException
         *
         */
        public void config(ServletContext ctx) throws IOException {
                init(ctx.getResourceAsStream("/WEB-INF/ad.properties"));
        }
        /**
         *
         *
         * @param is
         *
         * @throws IOException
         *
         */
        private void init(InputStream is) throws IOException {
                if (is != null) {
                        Properties prop = new Properties();
                        prop.load(is);
                        initialContextFactory = prop.getProperty("initial_context_factory");
```

```
                providerUrl = prop.getProperty("provider_url");
                prefix = prop.getProperty("principal_prefix");
                loginUser = prop.getProperty("login_user");
                loginPassword = prop.getProperty("login_password");
                baseName = StringUtil.nullToEmpty(prop.getProperty("baseName"));
                filterPrefix = StringUtil.nullToEmpty(prop.getProperty("filterPrefix"));
                filterSuffix = StringUtil.nullToEmpty(prop.getProperty("filterSuffix"));
                is.close();
        }
}
/**
 *
 *
 * @param userName
 *
 * @param password
 *
 * @return truefalse
 * @throws IOException
 *
 */
public boolean isPasswordValidate(String userName, String password) throws IOException {
        if (StringUtil.isNullOrEmpty(password)) {
                return false;
        }
        try {
                Hashtable<String, String> env = initEnv(userName, password);
                LdapContext context = new InitialLdapContext(env, null);
                context.close();
                return true;
        } catch (NamingException e) {
                return false;
        }
}
/**
 *
 *
 * @param userName
 *
 * @param password
 *
 * @return
 */
private Hashtable<String, String> initEnv(String userName, String password) {
        Hashtable<String, String> env = new Hashtable<String, String>();
        env.put(Context.INITIAL_CONTEXT_FACTORY, initialContextFactory);
        env.put(Context.PROVIDER_URL, providerUrl);
        env.put(Context.SECURITY_PRINCIPAL, prefix + userName);
        env.put(Context.SECURITY_CREDENTIALS, password);
        return env;
}
/**
 *
 *
 * @param userName
 *
 * @return
 * @throws IOException
 *
 * @throws UserNotExistException
 *             smartbi
 */
public UserInfo getUser(String userName) throws IOException, UserNotExistException {
        try {
                // AD-
                Hashtable<String, String> env = initEnv(loginUser, loginPassword);
                LdapContext context = new InitialLdapContext(env, null);
                // baseName Softerra LDAP Browser
                String base = baseName;
                // AD-
                String filter = filterPrefix + userName + filterSuffix;
```

```
                        SearchControls controls = new SearchControls();
                        controls.setSearchScope(SearchControls.SUBTREE_SCOPE);
                        // AD-
                        controls.setReturningAttributes(new String[] { "sAMAccountName", "displayName",
"department" });
                        NamingEnumeration<SearchResult> answer = context.search(base, filter, controls);
                        if (answer.hasMore()) {
                                //
                                SearchResult result = answer.next();
                                Attributes attrs = result.getAttributes();
                                Attribute attr = attrs.get("sAMAccountName");
                                String name = attr == null || attr.get() == null ? null : attr.get().toString();
                                attr = attrs.get("displayName");
                                String alias = attr == null || attr.get() == null ? null : attr.get().
toString();
                                attr = attrs.get("department");
                                String dept = attr == null || attr.get() == null ? null : attr.get().toString();
                                answer.close();
                                // smartbi
                                UserInfo info = new UserInfo(name, alias, alias);
                                //
                                GroupInfo group = new GroupInfo();
                                group.setName(dept);
                                // nullsmartbi""
                                // nullsmartbi
                                group.setParentGroup(null);
                                // smartbiList<GroupInfo>
                                List<GroupInfo> groups = new ArrayList<GroupInfo>();
                                groups.add(group);
                                //
                                info.setGroups(groups);
                                // ADnullsmartbi
                                info.setRoles(null);
                                return info;
                        }
                        //
                        throw new UserNotExistException(userName);
                } catch (NamingException e) {
                        log.error("getUser fail.", e);
                        return null;
                }
        }
        /**
         * Authentication
         *
         * @param userName
         *
         * @return truefalse
         * @throws IOException
         *
         */
        public boolean shallUserValidateInAuthentication(String userName) throws IOException {
                // admin
                if ("admin".equals(userName))
                        return false;
                else
                        return true;
        }
        /**
         *
         *
         * @param oldPassword
         *
         * @param newPassword
         *
         * @return
         * @throws IOException
         *
         * @throws UnsupportedOperationException
         *
         */
```

```
        public boolean changePassword(String userName, String oldPassword, String newPassword) throws
IOException,
                        UnsupportedOperationException {
                throw new UnsupportedOperationException();
        }
}
```