LDAP sample application (1.2 Ok)

{scrollbar}

top
Geronimo uses the Apache Directory Server for its directory service, this is part of the Apache Directory Project. Geronimo implements the following two projects from the ApacheDS project.

- ApacheDS Core:
  Server's core contains all backend subsystems. It depends on protocol and uses it with seda to service LDAP requests. The core contains the JNDI provider, interceptor framework, interceptor services, the schema subsystem and the database subsystem. Hence the core is the heart of the server.

- ApacheDS Shared:
  Created to eliminate cyclic project dependencies between the core and the maven plug-in. Any code shared across modules in general can go here so long as it does not depend on other modules.

More information about these two projects can be found at the ApacheDS project URL:

http://directory.apache.org/subprojects/apacheds/projects/index.html

At this point in time, Geronimo only provides LDAP viewing capabilities, editing is not there yet but adding this feature is in plan for the next releases of Geronimo. You will have to use an external LDAP client such as ldapbrowser/editor, jxplorer or gq for editing the configurations of the Directory Server in Geronimo.

Starting the LDAP server

In Geronimo v1.2 the Apache Directory v0.92 is already included with the distribution although it is not started by default. You can either start the server from command line using the deployer tool or via the Geronimo Administration Console.

Using the Administration Console click on **System Modules** on the navigation menu from the left and look for the component name **org.apache.geronimo.configs/directory** in the **Installed System Modules** portlet. You will see the current status and available commands for this particular component.

As we already mentioned, this component is stopped by default, click on **Start** to make this service available.

LDAP sample application

For your conveniece we have prodived the sample application and deployment plans packaged into a zip file.
Download the sample application from the following URL:

ldap-sample-app.zip

After extracting the zip file a **`ldap-sample-app`** directory is created, from now on this directory will be referred as <ldap_home>.

At this point it is assumed that you have installed an LDAP client and you are capable of exporting/importing an **`.ldif`** file to a directory server.

Back to Top

Add LDAP entries

Ensure that Geronimo is up and running and the Directory service is started. Start your LDAP client and create a new connection profile with the following values:

Host:	<localhost>
Port:	1389
Base DN:	ou=system
User DN:	uid=admin, ou=system
Password:	secret

Once you connect to the Geronimo Directory server you will see the initial configuration, this configuration can be exported as a backup in a ldif file. Depending the LDAP client you are using the export/import steps will be different. When you export the initial configuration you get an ldif file with a content similar as the one shown in the following example.

solidexport.ldif dn: ou=system ou: system objectClass: organizationalUnit objectClass: top dn: uid=admin, ou=system displayName: Directory Superuser uid: admin userPassword:: c2VjcmV0 objectClass: inetOrgPerson objectClass: organizationalPerson objectClass: person objectClass: top sn: administrator cn: system administrator dn: ou=users, ou=system ou: users objectClass: organizationalUnit objectClass: top dn: ou=groups, ou=system ou: groups objectClass: organizationalUnit objectClass: top dn: ou=configuration, ou=system ou: configuration objectClass: organizationalUnit objectClass: top

dn: ou=partitions, ou=configuration, ou=system ou: partitions objectClass: organizationalUnit objectClass: top dn: ou=services, ou=configuration, ou=system ou: services objectClass: organizationalUnit objectClass: top dn: ou=interceptors, ou=configuration, ou=system ou: interceptors objectClass: organizationalUnit objectClass: top dn: prefNodeName=sysPrefRoot, ou=system objectClass: extensibleObject prefNodeName: sysPrefRoot

Back to Top

Now you need to import the entries needed to run the sample application. Packaged with the sample application is a sample **`.ldif`** file with all the entries necessary to run the LDAP sample application, this file is located in **<ldap_home>/ldap-sample.ldif**.

The following example shows the content of the **`ldap-sample.ldif`** file.

solidldap-sample.ldif # User: system dn: uid=system,ou=users,ou=system cn: John Doe sn: Doe givenname: John objectclass: top objectclass: person objectclass: organizationalPerson objectclass: inetOrgPerson ou: Human Resources ou: People l: Las Vegas uid: system mail: system@apachecon.comm telephonenumber: +1 408 555 5555 facsimiletelephonenumber: +1 408 555 5556 roomnumber: 4613 userPassword: manager # User: user1 dn: uid=user1, ou=users,ou=system cn: User sn: One givenname: User1 objectclass: top objectclass: person objectclass: organizationalPerson objectclass: inetOrgPerson ou: Human Resources ou: People l: Las Vegas uid: user1 mail: user1@apachecon.comm telephonenumber: +1 408 555 5555 facsimiletelephonenumber: +1 408 555 5556 roomnumber: 4613 userPassword: p1 # User: user2 dn: uid=user2,ou=users,ou=system cn: User sn: Two givenname: User2 objectclass: top objectclass: person objectclass: organizationalPerson objectclass: inetOrgPerson ou: Human Resources ou: People l: Las Vegas uid: user2 mail: user2@apachecon.comm telephonenumber: +1 408 555 5555 facsimiletelephonenumber: +1 408 555 5556 roomnumber: 4613 userPassword: p2 # Group: admin dn: cn=admin,ou=groups,ou=system objectClass: groupOfUniqueNames uniqueMember: uid=system,ou=users, ou=system uniqueMember: uid=user2,ou=users,ou=system cn: admin # Group: guest dn: cn=guest,ou=groups,ou=system objectClass: groupOfUniqueNames uniqueMember: uid=user1,ou=users,ou=system cn: guest

Once the file is imported you should get a confirmation that five entries were successfully imported.

Back to Top

Deploy the LDAP realm

The LDAP sample application provides a security realm that needs to be deployed before the deployment of the application itself. This realm is located in **<ldap_home>/ldap-realm.xml** and the content is illustrated in the following example.

xmlsolidldap-realm.xml <module xmlns="http://geronimo.apache.org/xml/ns/deployment-1.2"> <environment> <moduleId> <groupId>console.realm< /groupId> <artifactId>LDAP_Sample_Realm</artifactId> <version>1.0</version> <type>car</type> </moduleId> <dependencies> <dependency> <groupId>org.apache.geronimo.configs</groupId> <artifactId>j2ee-security</artifactId> <type>car</type> </dependency> </dependencies> < /environment> <gbean name="LDAP_Sample_Realm" class="org.apache.geronimo.security.realm.GenericSecurityRealm" xsi:type="dep:gbeanType" xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <attribute name="realmName" >LDAP_Sample_Realm</attribute> <reference name="ServerInfo"> <name>ServerInfo</name> </reference> <reference name="LoginService"> <name>JaasLoginService</name> </reference> <xml-reference name="LoginModuleConfiguration"> <log:login-config xmlns:log="http://geronimo.apache. org/xml/ns/loginconfig-1.2"> <log:login-module control-flag="REQUIRED" server-side="true" wrap-principals="false"> <log:login-domain- name>LDAP_Sample_Realm</log:login-domain-name> <log:login-module-class>org.apache.geronimo.security.realm.providers.LDAPLoginModule</log: login-module-class> <log:option name="initialContextFactory">com.sun.jndi.ldap.LdapCtxFactory</log:option> <log:option name="connectionURL" >ldap://localhost:1389</log:option> <log:option name="connectionUsername">uid=admin,ou=system</log:option> <log:option name=" connectionPassword">secret</log:option> <log:option name="authentication">simple</log:option> <log:option name="userBase">ou=users,ou=system< /log:option> <log:option name="userSearchMatching">uid={0}</log:option> <log:option name="userSearchSubtree">false</log:option> <log:option name=" roleBase">ou=groups,ou=system</log:option> <log:option name="roleName">cn</log:option> <log:option name="roleSearchMatching">(uniqueMember= {0})</log:option> <log:option name="roleSearchSubtree">false</log:option> </log:login-module> <log:login-module control-flag="OPTIONAL" server-side=" true" wrap-principals="false"> <log:login-domain-name>LDAP_Sample_Realm-Audit</log:login-domain-name> <log:login-module-class>org.apache. geronimo.security.realm.providers.FileAuditLoginModule</log:login-module-class> <log:option name="file">var/log/login-attempts.log</log:option> </log: login-module> </log:login-config> </xml-reference> </gbean> </module>

To deploy the ldap-realm.xml run the following command from the <geronimo_home>/bin directory:

```
java -jar deployer.jar --user system --password manager deploy <ldap_home>/ldap-realm.xml
```

Once deployed you should see a confirmation message similar to the following example:

Back to Top

For further details refer to the LDAP Realm section.

Deployment plans

The deployment plans are located in the **<ldap_home>/WEB-INF** directory. Clearly, **geronimo-web.xml** is the Geronimo specific deployment plan. It provides the details on what security realm to use and user role mappings as well as the Geronimo specific namespace used to identify the elements in the security configuration. Common to other types of applications, not just security, the deployment plan also provides the main namespace for the deployment plan, a module identification (optional), a parent module configuration ID (also optional) and a context root. The following example illustrates the Geronimo specific deployment plan.

xmlsolidgeronimo-web.xml <?xml version="1.0" encoding="UTF-8"?> <web-app xmlns="http://geronimo.apache.org/xml/ns/j2ee/web-1.2"> <environment> <moduleId> <groupId>samples</groupId> <artifactId>LDAP_Sample</artifactId> <version>1.2</version> </moduleId> </environment> <context-root> /LDAP_Sample</context-root> <security-realm-name>LDAP_Sample_Realm</security-realm-name> <security> <default-principal realm-name=" LDAP_Sample_Realm"> <principal class="org.apache.geronimo.security.realm.providers.GeronimoUserPrincipal" name="system"/> </default-principal> <role-mappings> <role role-name="content-administrator"> <realm realm-name="LDAP_Sample_Realm"> <principal class="org.apache.geronimo.security. realm.providers.GeronimoGroupPrincipal" name="admin" designated-run-as="true"/> <principal class="org.apache.geronimo.security.realm.providers.

GeronimoUserPrincipal" name="system"/> </realm> </role> <role role-name="guest"> <realm realm-name="LDAP_Sample_Realm"> <principal class="org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal" name="guest" designated-run-as="true"/> <principal class="org.apache.geronimo.security.realm.providers.GeronimoUserPrincipal" name="user1"/> </realm> </role> </role-mappings> </security> </web-app>

Note that these role mappings will be overridden by the actual roles (what users pertaining to what groups) defined in the LDAP server. Ultimately it is the realm defined in the application deployment plan who determines the velidation method. Nevertheless, for this particular example, you still need to define principals and role mappings as determined in the XML schemas

Back to Top

The **web.xml** deployment descriptor shown in the following example (also located in the **<ldap_home>/WEB-INF** diretory) adds security constraints based on the location of the files.

xmlsolidweb.xml <?xml version="1.0" encoding="ISO-8859-1"?> <web-app xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4"> <welcome-file-list> <welcome-file>index.html</welcome-file> </welcome-file-list> <security-constraint> <web-resource-collection> <web-resource-name>Admin Role</web-resource-name> <url-pattern>/protect/*</url-pattern> </web-resource-collection> <auth-constraint> <role-name>content-administrator</role-name> </auth-constraint> </security-constraint> <security-constraint> <web-resource-collection> <web-resource-name>No Access</web-resource-name> <url-pattern>/forbidden/*</url-pattern> </web-resource-collection> <auth-constraint/> </security-constraint> <login-config> <auth-method>FORM</auth-method> <realm-name>ldap-realm-1</realm-name> <form-login-config> <form-login-page>/auth/logon.html?param=test</form-login-page> <form-error-page>/auth/logonError.html?param=test</form-error-page> </form-login-config> </login-config> <security-role> <role-name>content-administrator</role-name> </security-role> </web-app>

Back to Top

Package the sample application

Now that all the elements have been identified, it is necessary to package the sample application in a Web application Archive (.war). Open a command line window, change directory to <ldap_home> and run the following command:

```
jar -cvf ldap-demo.war *
```

This command will package all the existing files and directories inside **<ldap_home>**. Although not needed inside the .war file, the ldap-realm.xml and ldap-sample.ldif files will also be included.

Back to Top

Deploy the application

To deploy the LDAP sample application make sure the Geronimo server is up and running. Open a command line window, change directory to <geronimo_home>/bin and run the following command:

```
java -jar deployer.jar --user system --password manager deploy <ldap_home>/ldap-demo.war
```

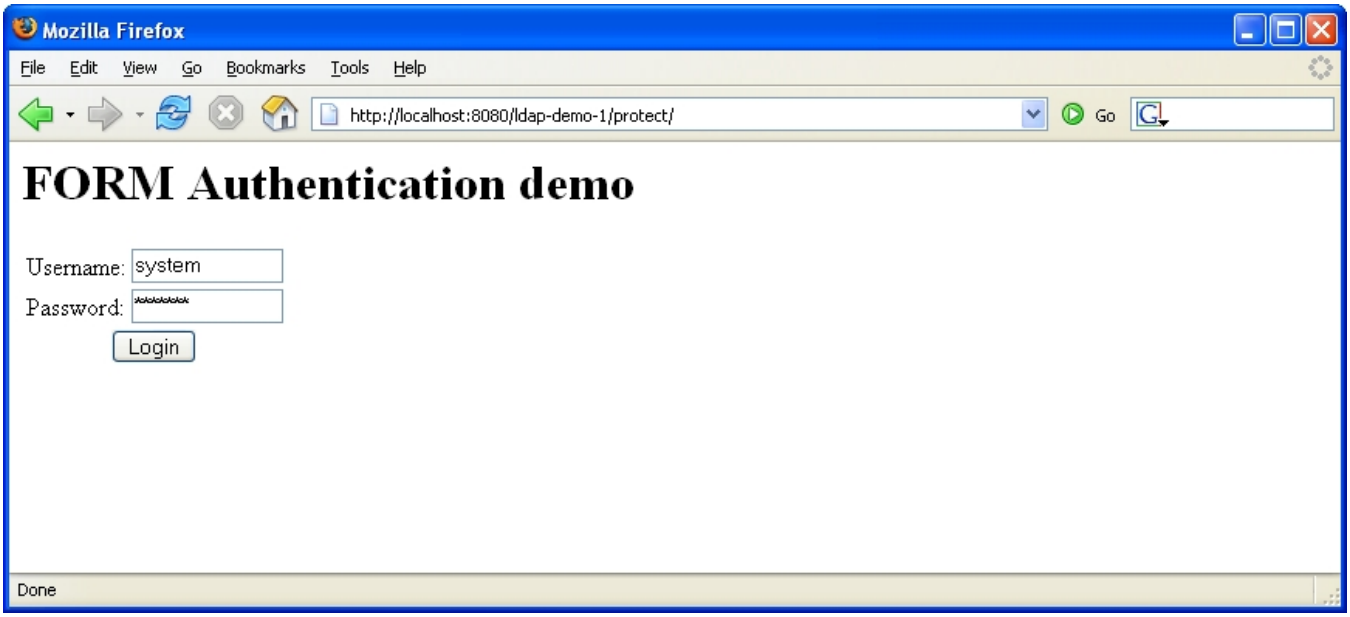
Once the Web application is successfully deployed you should see a confirmation message similar as the one shown in the following example:

To test the LDAP application open a Web browser and access the following URL:

http://localhost:8080/LDAP_Sample

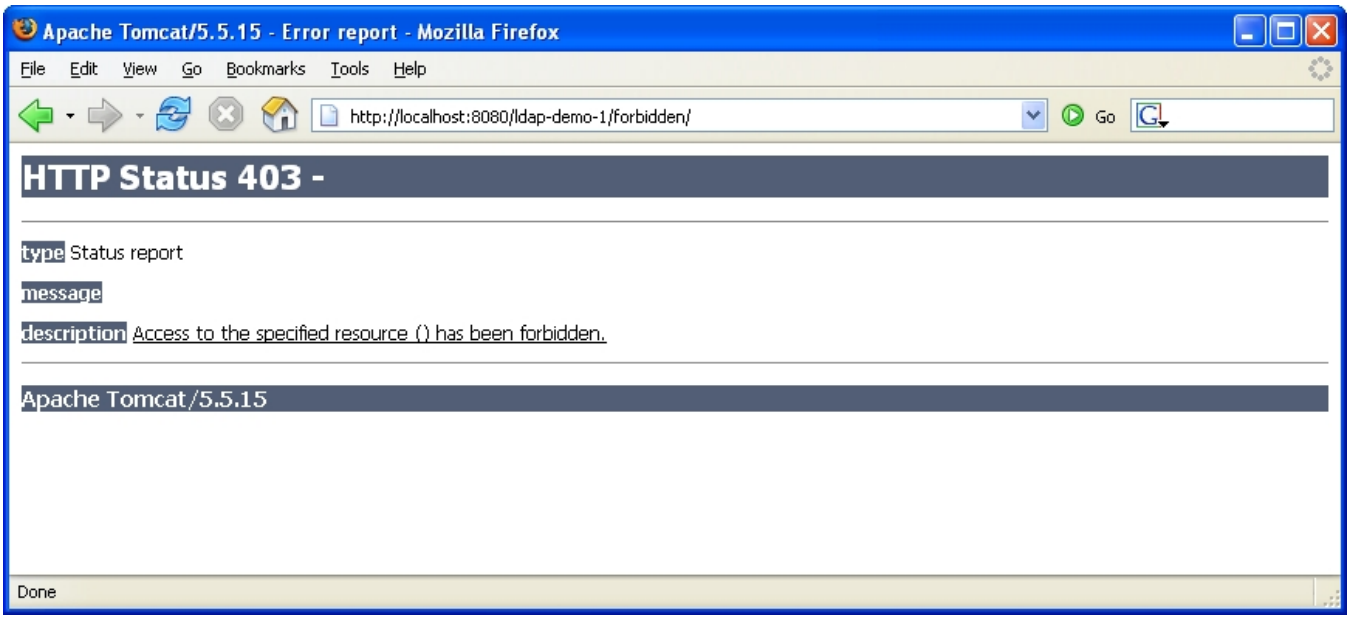
The following figure shows the welcome page for the LDAP sample application.

Click on Protect to validate against the LDAP Directory Server.



Enter **system** as the username and **manager** as the password and click **Login**. The username and password you provide here is the same you use to access the Geronimo Web console and it is stored in the Directory Server database. Once you are logged in you should see the following screen.

At this point you have an application that is validating username and passwords against an LDAP Directory Server database based on the security configuration you provided earlier in the LDAP realm. Now, if you go back to the welcome page and click on Forbidden you should receive a 403 - Forbidden HTTP error similar to the one shown in the following figure.



Depending on the web container you are using (that is Jetty or Tomcat) the presentation of that screen may be slightly different.

To further test this example you could now try the different users provided in the **`ldap-sample.ldif`**, use your LDAP client and add/remove users from the different groups. You will notice the changes immediatly (you may need to close your web browser).

Back to Top