ApacheDS as a Tomcat Realm

ApacheDS Tomcat Realm

What is it, and who needs it?

Within Apache Tomcat, a realm is (cite) *a "database" of usernames and passwords that identify valid users of a web application (or set of web applications), plus an enumeration of the list of roles associated with each valid user.* Learn more about them in the Realm Configuration HOW-TO from the Tomcat website.

In order to use Apache Directory Server (ApacheDS) as a realm, the straight forward approach is to use the JNDIRealm shipped with Tomcat. It can handle arbitrary LDAP servers, among them ApacheDS.

Because ApacheDS is 100% pure Java and embeddable, there is another option: Implement the Realm interface from Apache Tomcat and run *within* Tomcat. There are (at least) two interesting things about this approach:

- Tomcat can use ApacheDS without the wire protocol, because the realm has a handle to the "in process" API of ApacheDS
- ApacheDS runs within Tomcat and starts and stops automatically along with the web application server

While the first point promises some performance advantages, which are irrelevant in most situations (because LDAP servers like ApacheDS are optimized for read operations anyway), the second reason makes this approach a good option for development environments.

Current state

The current implementation of the realm has started as a proof of concept, but already works quite well. You can find the sources here:

https://svn.apache.org/repos/asf/directory/sandbox/szoerner/apacheds-tomcatrealm

It is not an official artifact of Apache Directory yet. Although if some people are interested, it soon can be. In the meantime, there are no official releases etc.

How to build it

After checking out the sources from subversion,

```
mvn install
```

builds a jar file which contains the Realm class. In order to use it, one has to copy it to the Tomcat server lib directory, along with the required ApacheDS and dependency jars.

How to enable and configure it

Activation in Tomcat server.xml is simply

```
<Realm className="org.apache.directory.tomcatrealm.EmbeddedApacheDsRealm" />
```

within the context the realm is intended to act.

When Tomcat starts up, ApacheDS starts as well and opens an LDAP port which defaults to 10389. You can use any LDAP client to connect to the embedded ApacheDS. We recommend Apache Directory Studio for this tasks.

Within the realm definition in Tomcats *server.xml*, you can add some attributes to configure ApacheDS:

Parameter	Description	Default value
ldapPort	the port number ApacheDS will listen to via LDAP	10389
workingDirectory	the working directory where ApacheDS stores its data	"*catalina.home*/apacheDsWorkingDir"
userSearchBase	search base used for user searches	"ou=system"
userSearchFilter	filter expression used for user searches	"(&(objectClass=inetOrgPerson)(uid={0}))"
groupSearchBase	search base used for group searches	"ou=system"

groupSearchFilter	filter expression used for group searches	"(&(objectClass=groupOfNames)(member={0}))"

Example:

```
<Realm
  className="org.apache.directory.tomcatrealm.EmbeddedApacheDsRealm"
  ldapPort="389"
  userSearchBase="ou=users,ou=system"
  groupSearchBase="ou=users,ou=system"
/>
```

How to use it

In order to use the realm, you have to add groups and users to the directory. Apache Directory Studio is a perfect solution for this task.

Connecting to the embedded ApacheDS

There is no difference to connecting to any other LDAP server. By default, the following connection properties will do:

Parameter	value
Hostname	*your hostname*, e.g. "localhost"
Port	10389
Bind DN (user)	"uid=admin,ou=system"
Password	"secret"

Adding a user

Use Studio to create a new entry with object class *inetOrgPerson* (plus super classes).

A user *tomcat* may look like this in LDIF.

```
dn: uid=tomcat,ou=users,ou=system
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
cn: Tomcat Administrator
sn: Tomcat
uid: tomcat
userPassword: tomcat
```

Adding a group

Use Studio to create a new entry with object class *groupOfNames*.

A group *manager*, which contains the previous user, may look like this in LDIF.

```
dn: cn=manager,ou=groups,ou=system
objectClass: groupOfNames
objectClass: top
cn: manager
member: uid=tomcat,ou=users,ou=system
```

Trying it out

Browse to the Manager web application within your Tomcat installation, http://localhost:8080/manager/html
Using the user and password "tomcat" should lead you to the application, all other input should fail (401 or 403).

Next steps

Some ideas I have:

- Make it an official artifact within Apache Directory
- Create a single jar which contains all dependencies necessary for running within Tomcat 5.5 and/or Tomcat 6.0
- Create some users and groups by default (at least "tomcat" within role "admin")
- Add the ability to define your own suffix ("dc=yourcompany,dc=com")

Provide feedback

Feel free to check it out and provide feedback. Further discussions about whether we provide this as official artifact will be on the dev-list of Apache Directory project.