spring-security-ldap - Reference Documentation

Authors: Burt Beckwith

Version: 0.1.1

Table of Contents

1. Introduction

The LDAP plugin adds support for LDAP and Active Directory authentication to a Grails application that uses Spring Security. It depends on the Spring Security Core plugin.
Once you have configured your Grails application as an LDAP client you can delegate authentication to LDAP and not have to manage users' authentication information in your application. By default roles are inferred from LDAP group membership and you can store additional application-specific roles in your database.
Please refer to the Spring Security LDAP documentation for details of the underlying implementation.

1.1 History

- Version 0.1
  - released 06/18/2010

2. Usage

Configuring your LDAP server is beyond the scope of this document. There are many different approaches and this will most likely be done by IT staff. It's assumed here that you already have a running LDAP or Active Directory server.

There isn't much that you need to do in your application to use LDAP. Just install this plugin, and configure any required parameters and whatever optional parameters you want in Config.groovy. These are described in detail in Chapter 3 but typically you only need to set these properties

```
grails.plugins.springsecurity.ldap.context.managerDn = 'uid=admin,ou=system'
grails.plugins.springsecurity.ldap.context.managerPassword = 'secret'
grails.plugins.springsecurity.ldap.context.server = 'ldap://localhost:10389'
grails.plugins.springsecurity.ldap.authorities.groupSearchBase = 'ou=groups,dc=yourcompany,d
grails.plugins.springsecurity.ldap.search.base = 'dc=yourcompany,dc=com'
```

Often all role information will be stored in LDAP, but if you want to also assign application-specific roles to users in the database, then add this

```
grails.plugins.springsecurity.ldap.authorities.retrieveDatabaseRoles = true
```

to do an extra database lookup after the LDAP lookup.
Depending on how passwords are encrypted in LDAP you may also need to configure the encryption algorithm, e.g.

```
grails.plugins.springsecurity.password.algorithm = 'SHA-256'
```

Custom UserDetailsContextMapper
There are three options for mapping LDAP attributes to `UserDetails` data (as specified by the `grails.plugins.springsecurity.ldap.mapper.userDetailsClass` config attribute) and hopefully one of those will be sufficient for your needs. If not, it's easy to implement UserDetailsContextMapper yourself.
Create a class in `src/groovy` or `src/java` that implements UserDetailsContextMapper and register it in `grails-app/conf/spring/resources.groovy`:

```
import com.mycompany.myapp.MyUserDetailsContextMapper
beans = {
   ldapUserDetailsMapper(MyUserDetailsContextMapper) {
      // bean attributes
   }
}
```

3. Configuration

Any property overrides must be specified in `grails-app/conf/Config.groovy` using the `grails.plugins.springsecurity` suffix, for example

```
grails.plugins.springsecurity.ldap.search.searchSubtree = true
```

There are several configuration options for the LDAP plugin. In practice the defaults are fine and only a few will need to be overridden.

Name	Default	Meaning
ldap.search.searchSubtree	`true`	If `true` then searc identified by conte then only searches context.
ldap.search.base	''	Context name to se of the configured C 'dc=example,dc=co 'ou=users,dc=exam
ldap.search.filter	`'(uid={0})'`	The filter expressio
ldap.search.derefLink	`false`	Enables/disables li search
ldap.search.timeLimit	`0` (unlimited)	The time to wait be
ldap.search.attributesToReturn	`null` (all)	The attributes to re
ldap.authenticator.useBind	`true`	if `true` uses a `Bi` bind as the authent a `PasswordCompa` to lookup the user passwords
ldap.authenticator.attributesToReturn	`null` (all)	names of attribute return all and an en
ldap.authenticator.dnPatterns	`null` (none)	optional pattern(s) patterns, e.g. "cn={
ldap.authenticator.passwordAttributeName	`'userPassword'`	the name of the pas when `useBind` =
ldap.mapper.convertToUpperCase	`true`	whether to upperca (will also be prefix
ldap.mapper.passwordAttributeName	`'userPassword'`	password attribute the `UserDetail`
ldap.mapper.userDetailsClass	`null` (create an LdapUserDetailsImpl)	use `'person'` to `'inetOrgPerso` `InetOrgPerson` `LdapUserDetai`
ldap.mapper.roleAttributes	`null`	optional names of

ldap.auth.hideUserNotFoundExceptions	`true`	if `true` throw a ne `BadCredential` throw the original `UsernameNotFo`
ldap.auth.useAuthPassword	`true`	If `true` use the su credentials in the a otherwise obtain th UserDetails object read the password
ldap.context.managerDn	`'cn=admin,dc=example,dc=com'`	DN to authenticate
ldap.context.managerPassword	`'secret'`	username to auther
ldap.context.server	`'ldap://localhost:389'`	address of the LDA
ldap.context.contextFactoryClassName	`'com.sun.jndi.ldap.LdapCtxFactory'`	class name of the `InitialContex`
ldap.context.dirObjectFactoryClassName	DefaultDirObjectFactory	class name of the D
ldap.context.baseEnvironmentProperties	none	extra context prope
ldap.context.cacheEnvironmentProperties	`true`	whether environme cached between re
ldap.context.anonymousReadOnly	`false`	whether an anonym be used for read-or
ldap.context.referral	`null` ('ignore')	the method to hand `'ignore'` or `'fo` to be automatically
ldap.authorities.retrieveGroupRoles	`true`	whether to infer rol membership
ldap.authorities.retrieveDatabaseRoles	`false`	whether to retrieve database using the
ldap.authorities.groupRoleAttribute	`'cn'`	The ID of the attrib name for a group
ldap.authorities.groupSearchFilter	`'uniquemember={0}'`	The pattern to be u is the user's DN
ldap.authorities.searchSubtree	`true`	If `true` a subtree s performed, otherwi used
ldap.authorities.groupSearchBase	`'ou=groups,dc=example,dc=com'`	The base DN from membership should
ldap.authorities.ignorePartialResultException	`false`	Whether `Partial` should be ignored i with Active Direct have a problem wit
ldap.authorities.defaultRole	none	An optional defaul users