

Flexible JDBC Realm for glassfish

by Erik Brakkee

Table of contents

1 What's this?	2
2 Platforms.....	3
3 License.....	3

News:

- 7-Feb-2010: Version 0.4 available. Improved logging. Basically traffic related logging (i.e. users being authenticated) is only done with log level FINEST> In addition there are improvements in the internal design.
- 15-Feb-2009: Version 0.3 available. Fixed a problem in hex encoding so that a hex number is always at least 32 characters long and padded with leading zeroes if needed.
- 6-Dec-2008: Version 0.2 available. Extension with user-specific password seeding as an option to avoid dictionary attacks. The features in this version are based on feedback from and discussions with Arnaud Rolly from gedial.com.
- 29-Jan-2008: Version 0.1 available!

1. What's this?

In many Java EE applications declarative security is required where user and group information is stored in a database. To support this, an application server must support a security realm based on a JDBC datasource.

Glassfish V2 and V3 application server also supports a configuration like this through the JDBCRealm. Unfortunately, this JDBCRealm is restrictive in various ways:

- It assumes a data model where groups are modeled as value objects in the sense of Eric Evan's terminology in Domain Driven Design. Specifically, if a group should have more properties apart from its name, then this should be modeled in a different database structure with the group name as a key.
- A very specific datamodel is assumed. Two tables are used: One with for every user the encoded password and another with pairs of usernames and groupnames to define to which groups a user belongs.
- It is static in that it assumes that a user will always be a part of the same groups over time. After retrieving the groups for the first time, it caches them indefinitely. This makes the JDBCRealm of glassfish unsuitable for dynamic applications where users can join or leave groups.

As is clear, the JDBCRealm of glassfish either fits your purpose and you are done, or it doesn't and you have to either work around it in your application or create a separate more flexible JDBC security realm yourself. Since I had a stable application that I wasn't intending on modifying, I decided to do the latter.

The FlexibleJdbcRealm is a JDBC security realm which is similar to the approach used in JBoss application server. Instead of depending on a fixed database structure with only limited configuration, it is configured with two queries instead:

- One query for determining the (encoded) password of the user based on the user name.

- One query for determining the groups the user belongs to based on the user name.

In other words, instead of assuming a certain type of data model with configuration of some column and table names and constructing the two JDBC queries for passwords and groups as JDBCRealm does, the FlexibleJDBCRealm is configured with the two queries. As a result, FlexibleJDBCRealm is more general than JDBCRealm since it can handle any datamodel that JDBCRealm can.

In particular, in the application that triggered this, I had a datamodel that did not fit the one assumed by JDBCRealm. In my design I am using surrogate keys and have three tables:

- a Users table with primary key, user name, encoded password, and other user attributes
- a groups table with primary key, group name, and other group attributes
- a user_groups table with a mapping of users to groups (based on primary key)

This datamodel can easily be handled using FlexibleJdbcRealm but would have required a redesign of the application if I would have used JDBCRealm.

2. Platforms

The FlexibleJdbcRealm only works with Java 5 or 6 and has been developed specifically for Glassfish V2 and is also known to work on Glassfish V3. I have been in contact with SUN in the hope that FlexibleJdbcRealm (or something with similar flexibility) will be added to a future version of glassfish.

3. License

The software is available as open source and is covered by the [Apache Software Foundation License version 2](#)