Getting Started with Grouper Provisioning Tests in Eclipse

Getting Started with Grouper Provisioning Tests in Eclipse

To test provisioning from Grouper to LDAP or from LDAP to Grouper, you will need a working Grouper API installation and an LDAP DSA.

These instructions assume that you have the Grouper and provisioning projects in your Eclipse workspace, and that you are using Maven to build Grouper. See Grouper Development Environment Using Maven.

Getting Started with the Grouper API

Start HSQL or configure a database for Grouper. The following example is for PostgreSQL.

conf/grouper.hibernate.properties

```
hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect
hibernate.connection.driver_class = org.postgresql.Driver
hibernate.connection.url = jdbc:postgresql://127.0.0.1:5432/grouperTRUNK
hibernate.connection.username = postgres
```

If you configure a database, whitelist the db connection. Since we are using maven instead of ant, Grouper will warn regarding jarfile mismatches. You may wish to turn off these warnings.

conf/grouper.properties

```
configuration.detect.errors = false


db.change.allow.user.0 = postgres
db.change.allow.url.0 = jdbc:postgresql://127.0.0.1:5432/grouperTEST
```

PostgreSQL requires a modification to `conf/sources.xml`, scroll down until you see the following and cut & paste the correct SQL.

conf/sources.xml

```
 <\!-\-  for postgres, use this query since no concat() exists:
```

Logging to stdout may be helpful, add `grouper_stdout` to the `log4j.rootLogger`.

conf/log4j.properties

```
log4j.rootLogger  = ERROR, grouper_error, grouper_stdout
```

The Grouper database needs to be initialized. I usually run `edu.internet2.middleware.grouper.app.gsh.GrouperShell` via a right-click and select `Run As` -> `Java Application`. This will fail since the Grouper database has not been initialized. I copy the `GrouperShell` run configuration via the menu `Run` -> `Run Configurations...`, add the `-registry -runscript` argument, and run this new run configuration.

Running the original `GrouperShell` run configuration should result in a `gsh` prompt.

```
Type help() for instructions
gsh 0%
```

The Grouper API installation is now ready.

Getting Started with Provisioning Tests

There are several example provisioning projects, named `psp-example-*`.

Configuration files are located in the `src/test/resources` directory. Please note that when running tests from the `psp-example-*` projects, the `src/test/resources` directory takes classpath precedence over the `grouper/conf` directory. For the curious, the configuration files in `src/test/resources` are included in the psp distribution.

The Apache Directory Studio plugin to Eclipse, available via the Eclipse Marketplace, is helpful when testing LDAP provisioning.

Test Provisioning from Grouper to LDAP

The `psp-example-grouper-to-ldap` project tests provisioning from Grouper to LDAP.

Adjust `ldap.properties` appropriately for your LDAP server.

> **Warning**
>
> All ldap entries under `edu.vt.middleware.ldap.base` will be deleted during testing !

Right-click on `edu.internet2.middleware.psp.GrouperToLdapTest.java` and `Run As` -> `Java Application` or `Junit Test`.

The `src/test/resources/data` directory contains ldif and xml files. The ldif files are used to initialize the LDAP directory for testing or to verify that the LDAP directory was correctly provisioned. The xml files are used to verify that the psp returned the "correct" SPMLv2 responses. "Correct" is in quotes since some psp messages are custom and outside of the SPMLv2 specification, for example, CalcRequest, DiffRequest, SyncRequest, etc.