Grouper loader with CSV data sources

Introduction

Inspired by GRP-1602- "add ability to load groups via csv". With a proper JDBC driver, CSV data sources can be used as a SQL data source in the grouper loader. Three drivers were tried. HSQL is already supplied with Grouper, and the other two – csvjdbc and jdbc-driver-csv – are open source projects. The significant differences between HSQL and the latter two are:

- HSQL requires a running server in order to map the table name to the csv file
- HSQL is read/write, and the others are read-only
- HSQL does not have column names as the first row
- HSQL has the SQL driver automatically loaded, while the others need the driver property specified in grouper-loader.properties
- HSQL can map a specific file to a table; the others use a directory as the url, with SQL "tables" mapped to files within the directory

HSQL

HSQL requires a running server in order to map the table name to the csv file. If the csv is an absolute file path outside of the local database directory, it also requires it to run with a special security flag, `textdb.allow_full_path=true`.

Server:

```
java -Dtextdb.allow_full_path=true -cp ./lib/jdbcSamples/hsqldb-2.3.5.jar org.hsqldb.server.Server --database.0
file:c:/temp/hsqlgrouper --dbname.0 grouper
```

Client:

```
java -jar sqltool-2.3.5.jar --inlineRc=url=jdbc:hsqldb:hsql://127.0.0.1/grouper,user=sa,password=
CREATE TEXT TABLE userList (groupName varchar(30), userid varchar(30));


SET TABLE userList SOURCE "c:\\Temp\\userlist.csv";


insert into userList values ('app:test:csvSync', '800000002');
insert into userList values ('app:test:csvSync', '800000003');
insert into userList values ('app:test:csvSync', '800000004');
...


select distinct groupName from userList;
```

grouper-loader.properties

```
db.hsql.url=jdbc:hsqldb:hsql://127.0.0.1/grouper
db.hsql.user=sa
db.hsql.pass=
```

CSVJDBC (recommended)

jar file tested: csvjdbc-1.0.34.jar

Homepage

Documentation

This is a read-only JDBC driver, with some basic column and aggregate functions, sorting and result limits. It also supports many connection parameters, to handle a range of different csv styles. The URL will point to the directory containing one or more csv files. Within the directory, files ending in *.csv will become the table names, minus the file extension. The first row will contain column names. If some rows contain fewer than all columns, add property `missingValue=___` to the URL.

grouper-loader.properties

```
db.csvjdbc.driver=org.relique.jdbc.csv.CsvDriver
db.csvjdbc.url=jdbc:relique:csv:/tmp/csvjdbc?missingValue=false

# username and password aren't needed, but expected by Grouper
db.csvjdbc.user=sa
db.csvjdbc.pass=
```

Note that this jar also has a convenience class to output a Resultset as csv to a stream (including stdout), which is useful for general debugging in gsh:

```
import java.sql.*;
Connection conn = DriverManager.getConnection("jdbc:relique:csv:/tmp/csvjdbc")
Statement stmt = conn.createStatement();
ResultSet results = stmt.executeQuery("select distinct userid as SUBJECT_ID, 'people' AS SUBJECT_SOURCE_ID from
userList")

import org.relique.jdbc.csv.CsvDriver
CsvDriver.writeToCsv(results, System.out, true) // true will print column headings
```

JDBC-DRIVER-CSV

jar file tested: jdbc-driver-csv-1.2.0.jar

Homepage

This project is based on the above csvjdbc, with only a few changes. It has been tested and works.

grouper-loader.properties

```
db.xbib.driver=org.xbib.jdbc.csv.CsvDriver
db.xbib.url=jdbc:xbib:csv:/Temp/csvjdbc
db.xbib.user=sa
db.xbib.pass=
```

: