Repast Simphony Database Outputter Plugin - Documentation

Sascha Holzhauer, Center for Environmental Systems Research

October 15, 2010

1 Introduction

This documentation version is related to the plugin version 0.8. The database outputter works analogous to a file outputter extending log4j's `JDBCAppender`. Like the file outputter, the database outputter is configured via a 2-step wizard reusing/extending the file outputter and JDBC freezedryer wizards. The code should integrate well into simphony as a plugin.

The performance of writing to a DB compared to a file is addressed by adding the option to cache a certain number of SQL statements and execute these at once. This might reduce the additional time to a negligible amount. The advantage is to directly put data in the database without getting confused about numerous files somewhere on the hard disk.

Features include:

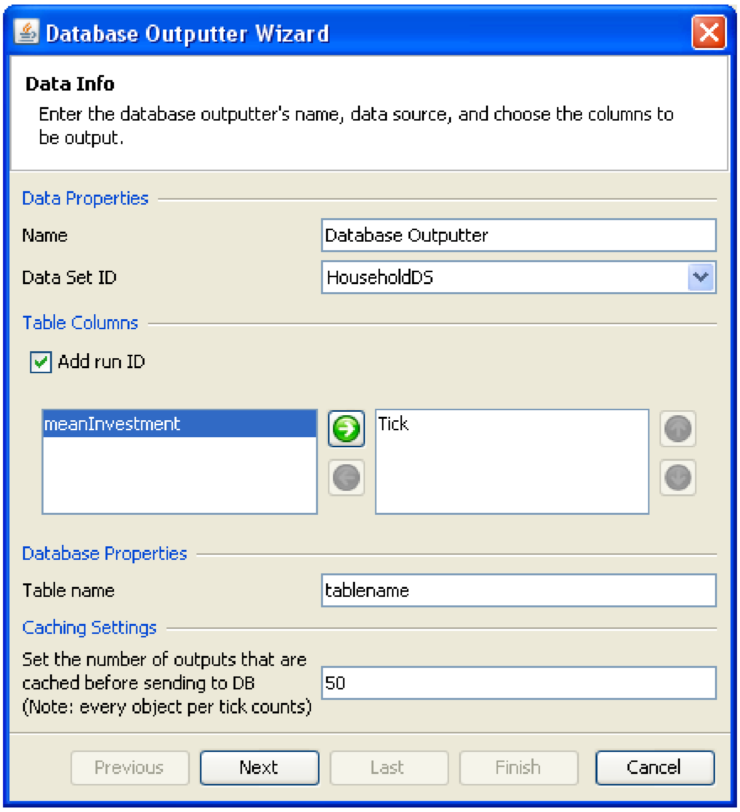
- Caching of a definable number of outputs before SQL execution
- Configuration via 2-step wizard
    - Auto-completion for URL and driver in database properties wizard step
    - Ability to test MySQL connection
    - Choose whether to store login information or to be prompted
- Integrates well in Repast Simphony as plugin folder
- Creates MySQL table if it does not exist

2 Installation

Just extract the zip file into your eclipse installation folder. The result should be a new folder called `repast.simphony.data.db_1.2.0` within the plugin folder.

3 Setting up a Database Outputter Action

A Database outputter action is configured similar to a file outputter action. After you configured one or more data sets, in the scenario tree, right-click on "Outputters" and choose "Add DB Outputter". The wizard's first step page (figure 1) appears:



Data Info

Enter the database outputter's name, data source, and choose the columns to be output.

Name: Database Outputter

Data Set ID: HouseholdDS

☑ Add run ID

meanInvestment

Tick

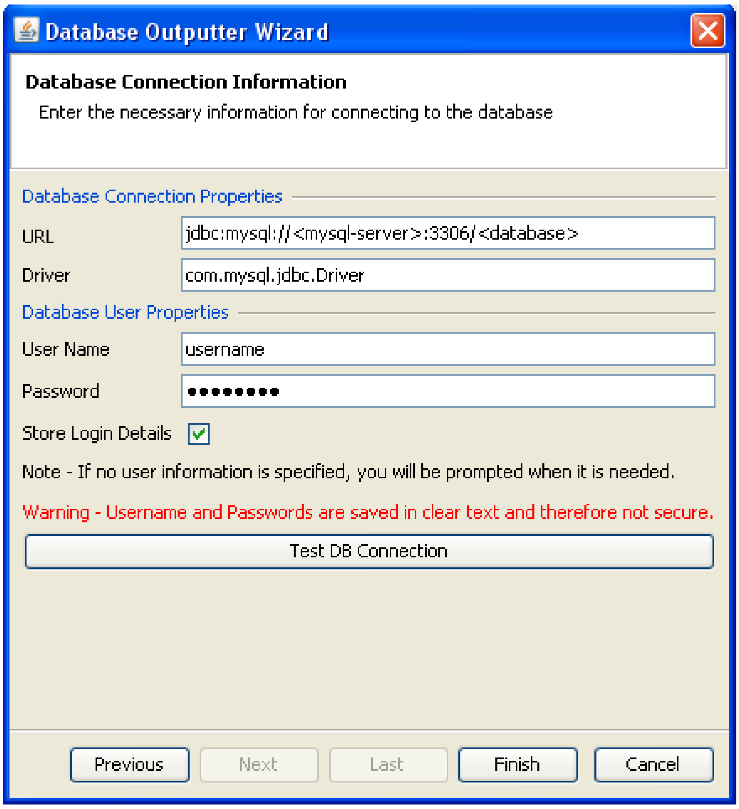
Table name: tablename

Set the number of outputs that are cached before sending to DB (Note: every object per tick counts): 50

Figure 1: Wizard Step 1

- Data Properties
  - **Name:** Choose an arbitrary name for the outputter action that itentifies the outputter in the sceanrio tree.
  - **Data Set ID:** Use the drop-down menu to choose a previously configured data set.
- Table Columns
  - **Add run ID:** This is important for batch runs. In case it is activated the plugin adds a column “runID” to each data row. The highest runID will be queried from the table and increased. In case of a new table it start with 1.
  - Move the columns that shall be stored in DB to the left.
- Data Base Properties
  - **Table name:** Type in the name of the table the data shall be stored in. NOTE: If the table does not exist, the plugin tries to create a new table.
- Caching Settings

  - Specify the number of outputs that are cached before the data is sent to the database. This can save an enormous amount of time since a single database query could be very time consuming. Every object per tick counts. I.e., if the data set is defined to gather data from 5 agents every second tick, data will be passed to the database after 20 ticks when the number of outputs to cache is 50. Note that the interval of data storage is defined during the configuration of the data sets.



Database Connection Information

Enter the necessary information for connecting to the database

Database Connection Properties

URL: jdbc:mysql://<mysql-server>:3306/<database>

Driver: com.mysql.jdbc.Driver

Database User Properties

User Name: username

Password: ●●●●●●●●●

Store Login Details ☑

Note - If no user information is specified, you will be prompted when it is needed.

Warning - Username and Passwords are saved in clear text and therefore not secure.

Test DB Connection

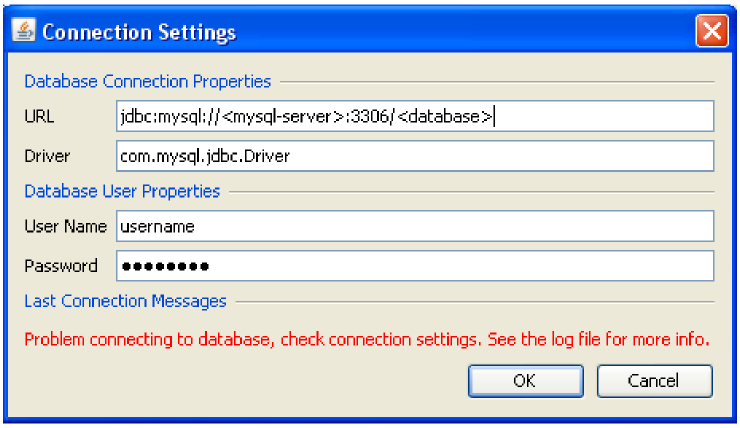
Figure 2: Wizard Step 2

The second wizard step (figure 2) deals with database connection properties:

- Database Connection Properties
  - **URL:** State the URL of the database you want to connect to, e.g. `jdbc:mysql://mysql:3306/simulations`. This text field features auto completion.
  - **Driver:** Specify the JDBC driver class according to your database, for instance `com.mysql.jdbc.Driver`. Make sure that the driver is available at the classpath. The MySQL Connector-J library containing the driver may be downloaded from

`http://www.mysql.com/downloads/connector/j/`. This text field features auto completion.

- Database User Properties
  - **User:** The username to connect with.
  - **Password:** Specify the password to log on at the database if there are no security concerns since passwords are stored in clear text in the outputter deciption XML file within the model configuration folder (*.rs). If there are concers, uncheck to following box.
  - **Store Login Details:** If it is checked username and password will be stored in clear text to the configuration file. If the box is unchecked, input fields for username and password are disabled. Furthermore, the “Test DB Connection”-Button is diabled since it is not possible to check the connection without username and password. However, it is possible to check the box, fill in username and password, check the connection, and uncheck the box to prevent storing login data. When no username and password are given, the plugin prompts for login data when the simulations is initialised. Then, also a connection test is performed (figure 3).
  - **Test DB Connection-Button:** Press the button to test a connection to the MySQL-Database with given parameters. If the test fails, a dialog (figure 3) appears and gives the opportunity to correct the data. If the test passes, the newly entered data is passed to the second wizard step.



Database Connection Properties

URL jdbc:mysql://<mysql-server>:3306/<database>

Driver com.mysql.jdbc.Driver

Database User Properties

User Name username

Password ••••••••

Last Connection Messages

Problem connecting to database, check connection settings. See the log file for more info.

Figure 3: Wizard Step 2 Communication Settings

4 Implementation

4.1 DefaultDBOutputter

`DefaultDBOutputter. CachedJdbcAppender` extends `JDBCAppender` and uses most of the functionality. To enable chaching, `DefaultDBOutputter. CachedJdbcAppender` overwrites `JDBCAppender#execute()` and stores all sql statements into a buffer until the limit is reached. Furthermore, it adds the column definitions as the first part of SQL statement. To flush the buffer on `close()`, `DefaultDBOutputter.CachedJdbcAppender` also overwrites `JDBCAppender#close()` and directly calls `JDBCAppenderexecute()` to by-pass caching.

Overwriting `JDBCAppender#getLogStatement()` is used to generate the SQL peace for every column defined in the `DbOutputterDescriptor. CachedJdbcAppender` and also adds the `runID` if activated.

To ensure all cache is sent to the database, the `DbOutputterDescriptor.CachedJdbcAppender`'s `close()` method is scheduled at the last tick with priority `ScheduleParameters. LAST_PRIORITY`. Otherwise, `close()` is not called before the simualtion is reset, and in case the user closes the application without resetting the cached data is missed.

4.2 Error Handling

The plugin does not throw `SQLExceptions` in order not to interrupt simulations because of output errors. Instead, a ERROR-Logging to the Repast Simphony Message Center is triggered.

5 Things to Do

- Adjust dialog dimensions
- Check for column types using `DbOutputterDescriptor.CachedJdbcAppender#doesTableDefinitionExist(String tableName, Map<String, Object> cols)`. Since the `DataGathererDescriptor` does not contain any type information this is not straight forward.
- Create additional table columns if requiered
- Handle more objects (e.g. arrays) to log
- Check for valid characters in table name.

6 Contact

Any suggestions and bug reports are appreciated. Please, send an eMail to `holzhauer@usf.uni-kassel.de`.