API documentation

Class MySQLlib: This is the class implementing the interface SqlRunner and contains the API functions.

Read README for build and run instructions.

The description of the methods are as follows:

1. void initConnection(String userName, String password, String url): This method connects the user to the database.

2. void closeConnection(): Closes the connection to the database

3. void setXMLCommandPath(String xmlCommandPath): Used to modify default queries.xml file path

4. void printTableColumnTypes(String table_name): Used to examine a table in the database by printing column types and the tabel

5. public <T, R> R selectOne(String queryId, T queryParam, Class<R> resultType): Used to return an object of generic type R by running select query returning single row. @Return: Sets fields to null if no return value, null if error

6. public <T, R> List<R> selectMany(String queryId, T queryParam, Class<R> resultType): Used to return an object of generic List<R> by running select query returning multiple rows. Returns empty list if no row is returned, else null for error

7. public <T> int update(String queryId, T queryParam): Used to update database. Returns -1 in case of an error.

8. public <T> int insert(String queryId, T queryParam): Insert rows in database. Returns -1 in case of an error

9. public <T> int update(String queryId, T queryParam): Delete rows. Returns -1 in case of an error.

10. Field boolean autoCommit: Set this field to false if you do not want SQL sessions to change the database. Field to be set before calling initConnection()

Data types allowed:

1. Data Types allowed for queryParam argument:
    a. Scalar data types: Types such as Integer, Float, String, etc defined in java.lang.* are allowed.(Note: int, float, (primitive java types) etc cannot be type casted into generic objects as required. Thus, use of only wrapper classes from java.lang such as Integer, Float, etc is permitted. This is similar to the constraints several Java APIs put on such as Java List, for example List<int> is not allowed but Java<Integer> is).
    b. List<> of scalar types(from point a) from java.util.ArrayList. This is used to replace the individual parameters in SQL queries in their order left to right. Number of elements in the list should be equal to the number of parameters.
    c. Java array (eg Integer arr[]) of scalar types(in point a). This is used to replace the parameters in SQL queries in order left to right. Number of elements in the array should be equal to the number of parameters.
    d. Custom Java POJOs with scalar and public fields, with the fields having the same name as the SQL table columns. The values are mapped using this property.
    e. null in case SQL query has no parameters.
    f. Timestamp, Date time, Date objects should be supplied as Strings since preparedStatement JDBC API cannot insert it as a parameter.

2. resultType of selectOne and selectMany: A Java POJO with the public fields and data type *exactly* same as the data types of SQL table columns.

XML file format:

1. paramType field:
- It should contain the Fully Qualified Name of the parameter object(obtained from obj.getClass().getName() function).
  Eg: java.util.ArrayList for ArrayList
- In case there is no parameter, set it as an empty string

2. To insert Arrays into one SQL column in SQL queries, the array structure(appropriate brackets, commas, etc) should be laid out by the user, with individual array elements being marked as ${<value_at_the_index>}. The ${<value_at_the_index>} are to be supplied in separate scalar variables.