JUDIE Tutorial

2008-07-14, Thomas Kestler

Overview

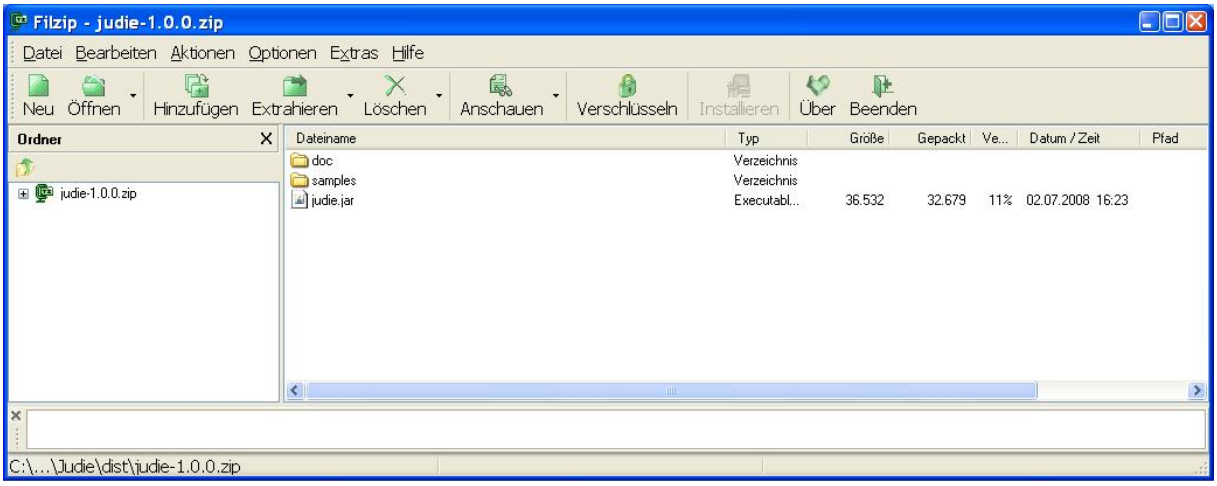
JUDIE stands for Java Universal Database Import and Export. It was planned to allow users and developers to export data from one database to XML and import from this XML into another database. There are many use cases like data migration or setup of test databases. XML allows you to manipulate the data, maybe you already have XML export from legacy system and you can now transform this XML files (using XSLT) to JUDIE format. Also export from other tools (like db/Torque) can be easily transformed into JUDIE format and vice versa (see functional specification for more information about XML Schema/DTD).

JUDIE is OpenSource and published under LGPL. Everyone is free to use it with own applications. JUDIE is based on JDBC and allows to connect to a large number of database systems.

JUDIE comes with a full API, a command line tool and with ant support. Also there is an Eclipse Plugin called JUDIEclipse4QuantumDB, see below.

Installation

Installation is easy, download the distribution archive (ZIP/TAR) and extract to any folder. There will be docs/ folder for Javadoc and lib/ folder for judie-N_N_N.jar (depending on version, e.g. judie-1_0_0.jar). The sample/ folder contains samples for command line usage or ant tasks.



The easiest way to run JUDIE is the command line tool. Copy one to the samples and modify to your personal settings (JDBC driver, database connection, classpath and so on).

First steps

For the first steps you need at least one database running, for example MySQL. Create two users there with privileges to create tables. Then login as user1 and create a sample table:

```
CREATE TABLE sample1 (
 id INT PRIMARY KEY,
 name VARCHAR(100)
);

INSERT INTO sample1 VALUES (1, 'Just a test ');
INSERT INTO sample1 VALUES (2, 'One more test');
```

Now modify your command line script to the right classpath of your database driver and JDBC connection settings. It should look similar to this:

```
java -classpath ../libs/commons-cli-
1.1.jar;../dist/judie.jar;C:\java\eclipse-workspace\Judie\libs\mysql-
connector-java-5.1.6-bin.jar de.onoffshoresoft.judie.cmd.JUDIECmd -e -d
com.mysql.jdbc.Driver -c jdbc:mysql://192.168.1.103:3306/testdb -u
judietest -p judietest -t sample1 sample1.xml
```

After this the export to XML should be done and your output file should look similar to this

```
<?xml version="1.0" encoding="UTF-8"?>
<dbexport xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://judie.sourceforge.net/judie.xsd">
<dbtable name="sample1"  dbschema="judietest@192.168.1.101">
<tableinfo>
    <colinfo name="id" type="INT"  sqltype="4"  scale="0"  precision="11"
isNull="false" isPrimary="true"></colinfo>
    <colinfo name="name" type="VARCHAR"  sqltype="12"  scale="0"
precision="100"  isNull="true" isPrimary="false"></colinfo>
</tableinfo>
<rowset>
<row>
<column name="id">1</column>
<column name="name">Just a test</column>
</row>
<row>
<column name="id">2</column>
<column name="name">One more test</column>
</row>
</rowset>
</dbtable>
</dbexport>
```

Very good, it works! So let's quickly test the other direction: import. Modify your command line script similar to this to do the import:

```
java -classpath ../libs/commons-cli-
1.1.jar;../dist/judie.jar;C:\java\eclipse-workspace\Judie\libs\mysql-
```

```
connector-java-5.1.6-bin.jar de.onoffshoresoft.judie.cmd.JUDIECmd -i -d
com.mysql.jdbc.Driver -c jdbc:mysql://192.168.1.103:3306/testdb -u
judietest -p judietest -t sample1 -v judietest2  sample1.xml
```

When you run this import script against MySQL you may see outcome like this:

```
CREATE TABLE sample1
(
        id INT(11)      NOT NULL,
        name VARCHAR(100)
)

INSERT INTO sample1(id,name) VALUES (?,?)

de.onoffshoresoft.judie.service.JUDIEException: Error occured while trying to in
sert a record
```

Ooops, why? Please note that MySQL has no concept of schemas, so the data from XML will be inserted into existing table sample1. You may wonder why no error on CREATE TABLE if the table sample1 already exists: default import mode is ignore error if table exists (see FS for details). So JUDIE tries to insert the rows, but fails because of PRIMARY KEY constraint.

Okay, we could manually login and issue a SQL to DROP the table before the import. The command line tool will then output something like this:

```
CREATE TABLE sample1
(
        id INT(11)      NOT NULL,
        name VARCHAR(100)
)

INSERT INTO sample1(id,name) VALUES (?,?)

INSERT INTO sample1(id,name) VALUES (?,?)

imported 2 rows into 1 tables (0 seconds)
```

Another way is to use the import mode 4 by adding option `-m 4` to the command line which results in a `DROP TABLE` command before `CREATE TABLE`.

The command line tool

The command line tool enables administrators or developers to easily export/import data. The syntax is similar to other Unix style commands:

```
judie.sh {options} {file}
judie.bat {options} {file}
```

The script just calls the main class `de.onoffshoresoft.judie.cmd.JudieCmd` with the parameters provided to the script.

Options

`-i` **i**mport
`-e` **e**xport

one of both must be supplied

`-d driver` **d**river class like `oracle.jdbc.driver.OracleDriver`, this class or JAR must be included into classpath.

`-c url` JDB**C** Url like `jdbc:oracle:thin:@localhost:1521:XE`

`-u user` **u**ser name

`-p password` **p**assword, if not present, the command reads the password from standard input after prompting (“enter password: “).

`-t …` tables export and import: This can be one **t**able name or a list of tables, separated by comma. Tables names can be fully qualified names like catalog.schema.table or simple names. This just depends on use case. Normally you access tables from your (implicit) user schema by simple names. If you want to dump tables from other schemas or catalogs, you use FQN. If missing on import, then **all** tables in XML file will be imported into target database!

`-s sql` export: **S**QL to extract data.

`-S alias` export: alia**s** table name for the SQL result, used as table name in XML file. If not supplied the default alias is “sql”.

`-o` export: **o**mit metadata. Normally table metadata is exported. This option turns off export of table metadata.

`-r N` export: limit number of rows exported per table, import: commit after N **r**ows. N must be a positive number > 0, but below MAXINT (2^31-1). Per default auto commit is on, which is similar to N=1. The row counter is incremented over all tables, so setting it to a very high number would result in one large transaction for the whole import.

-n import: validato**n**, activates XML validation on import (on export never any validation is done). By default validation is off. XSD file is referrenced via http URL `http://judie.sourceforge.net/judie.xsd`

`-m 1|2|3|4` import: error handling **m**ode, see above. Default is 3.

-x e**x**port: do not order tables for export according to referential constraints, default is on.

-z export: **z**ip the output stream, import: unzip import stream. Default: off. If exported to file, the file of the ZIP archive is that supplied by the user. It is recommended to use the extension .xml.zip. The created ZIP archive will contain one entry `judie.xml` with the exported data. On import the program will open the given file and presume it is a ZIP archive and will read the first entry as the XML file with exported data.

-f format export: custom **f**ormat for date (dt), time(ti) or timestamp (ts) followed by colon and a JAVA format pattern. This option can appear multiple times.

Example 1: …`–f "ts:dd.MM.yyyy HH:mm:ss" –f "dt:dd.MM.yy" –f "ti: hh:mm"`
Example 2: …`–f "ts:dd.MM.yy HH:mm" "ti:hh:mm"`

If supplied this format string will be stored in `format` attribute of `column` element in exported XML. On import this attribute will be recognized and used for conversion. This is for users convenience to get exported data in XML in his native format.

-v schema import: o**v**erride schema – by default (without this option) schema gets preserved when importing tables, that is, JUDIE tries to import table in same schema for where it was exported. If EMP was exported from SCOTT, JUDIE will import into table SCOTT. Using " …–v TEST …" will override schema information and import into TEST.EMP. Same is true for table creation.

-b SQL import: issue this SQL **b**efore importing each table, Milestone 2.

-a SQL import: issue this SQL **a**fter importing each table, Milestone 2

File

Normally user provides the name of the XML file to export into or to read from. If file is missing, export goes to `System.out` or import reads from `System.in`. This is usual Unix behaviour.

Examples

Export table EMP from Oracle database schema SCOTT to file emp.xml:

```
judie.sh –e –t EMP –d oracle.jdbc.driver.OracleDriver –c
jdbc:oracle:thin:@localhost:1521:XE –u scott –p tiger emp.xml

judie.sh –e –t SCOTT.EMP,SCOTT.DEPT –d oracle.jdbc.driver.OracleDriver –c
jdbc:oracle:thin:@localhost:1521:XE –u scott –p tiger emp_and_dept.xml

judie.sh –e –s "SELECT * FROM DEPT WHERE location = 'NY'" –S DEPT –d
oracle.jdbc.driver.OracleDriver –c jdbc:oracle:thin:@localhost:1521:XE –u
scott –p tiger dept.xml

judie.sh –i –d oracle.jdbc.driver.OracleDriver –c
jdbc:oracle:thin:@localhost:1521:XE –u scott –p tiger emp.xml

judie.sh –i –t SCOTT.EMP –d oracle.jdbc.driver.OracleDriver –c
jdbc:oracle:thin:@localhost:1521:XE –u scott –p tiger emp_and_dept.xml
```

You can find some similar examples in the `sample_files` folder in the distribution ZIP.

Writing your own application

If you want to write your own application based on JUDIE API you have to add judie.jar to your project and classpath. There is one central interface called `JUDIEService` and an implementation class `JUDIEServiceImpl`.

All in all it is pretty simple to use JUDIE:

```
…
JUDIEParams judieParams = new JUDIEParams();
JudieParams.set…
JUDIEServiceImpl judieService = new JUDIEServiceImpl();
judieService.importData(connection, judieParams, inputStream,
                                    new PrintWriter(System.out));
…
```

Please note that you have to catch the `JUDIEException` thrown by service methods. Whenever such an exception is thrown a fatal failure has occurred and there is no way to continue processing.

The Eclipse Plugin

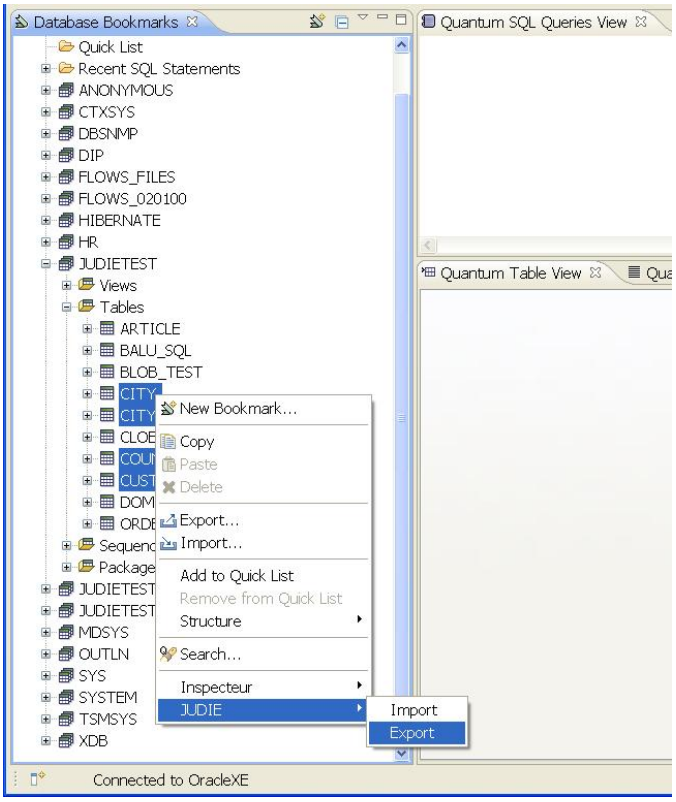
Actually there is one plugin based on QuantumDB plugin. To use this you first have to install QuantumDB plugin and then install the JUDIE Plugin for QuantumDB by simply unpacking the distribution zip archive into your eclipse folder. After restart of eclipse you should find a new context menu “JUDIE” in the QuantumDB bookmark view. This menu has two items export and import.

Depending on what nodes you have selected in the bookmark view you can import or export these tables. If you select a single table, then this table will be imported or exported. If you select several tables, the these will be processed. If you select a higher level node, like the schema node, then all tables below will be selected implicitly. You can see the list of selected tables in the wizards.

You can even select tables from different schemas as long as you have the permission to read/write them.

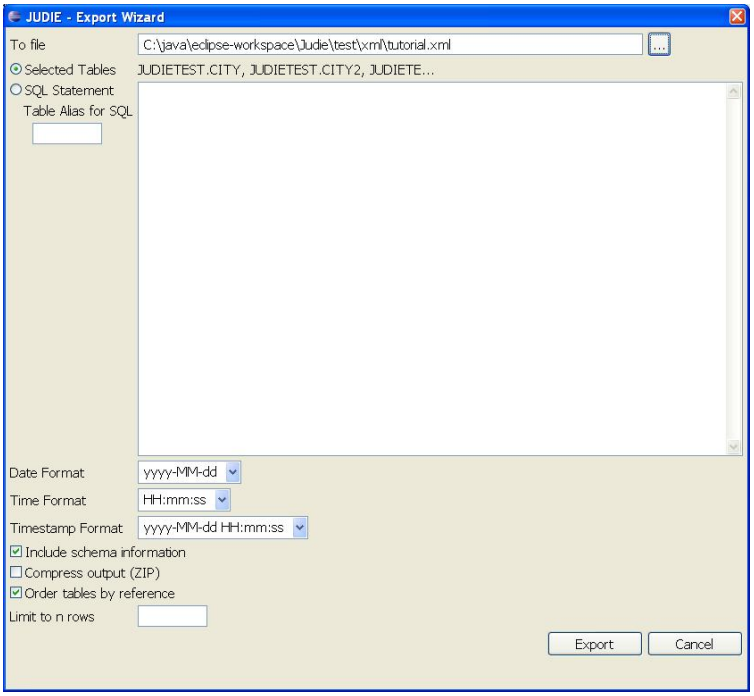
If you don’t select any table on import the “all tables” radio button will be selected and this means all tables from XML file will be imported.

Here some simple steps with the plugin: first how to export a couple of tables:



You select the tables to export in the bookmark view. Then choose Export menu item from the JUDIE menu. Please note that QuantumDB plugin has its own export feature which is not related to JUDIE.

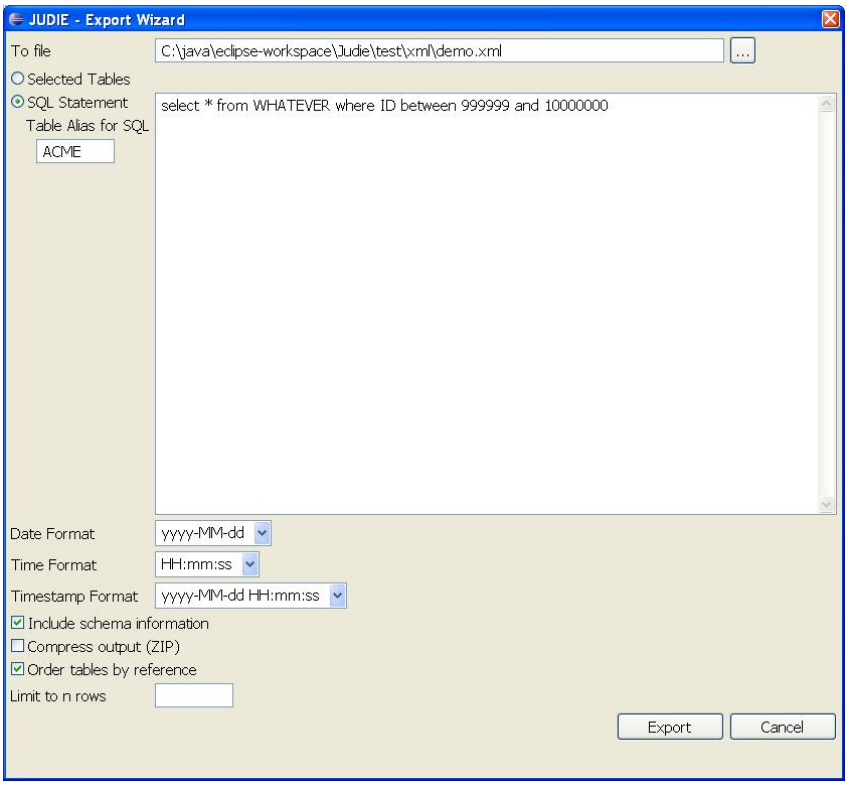
After doing so, the export wizard appears:



You have to select the output file and you can set some options. You can set the format for date, time and datetime columns and you can decide whether to include schema information (default is on, a good idea). You can compress the output as ZIP

archive which saves space. Then pls name your output file to a name with extension .zip. Because tables in an RDBMS often have relations it makes sense to export them in an order that allows later import without trouble caused by relational constraints (you cannot import CUSTOMER before CITY is imported). By default this ordering is on (again, a good idea). If you want to export only a subset of your data, than you can limit the number of rows per table(!) to be exported.

If you want to export data based on a SQL statement, you can also do. Just choose the SQL statement radio button and insert your statement. Please note, this statement will be executed, so don't type in some foolish like DROP DATABASE! There is no extra check. Also you can input only one statement at a time. Because JUDIE thinks in tables you must give the resulting `dbtable` Element a name (otherwise we do for you, the default name is sql):



Advanced topics

Schema handling

Most databases support the concept of schema, that means, a table `CUSTOMER` can exist in schema `SALES` and also another table `CUSTOMER` can exist in schema `MARY`. Both tables are different entities. Therefore the schema is needed to identify the entity `CUSTOMER`. The schema is an attribute of `dbtable` Element. If you import a XML file then JUDIE would use this schema from attribute. If you want to import table data into another schema, you could manipulate the XML file using an editor or (much better) you use the schema override option.

Please note that MySQL and other databases don't support different schemas

Transforming the XML file

The power of XML is its flexibility. You can easily transform the JUDIE XML output into other XML formats using XSTL scripts. Actually we don't supply such sample scripts, but hopefully the community will provide some soon.

ZIP handling

JUDIE can write output to ZIP file and read from zipped XML file. If you choose the option for ZIP on export than your file will be a ZIP archive containing exactly one XML file as entry named judie.xml. If you choose ZIP option on import JUDIE expects the ZIP file to be of same structure (it will look for the first entry and try to import).

Support and Bug Reporting

First of all please read the available documentation and FAQ. You can issue a support request of bug/feature request at the SourceForge project websites. Actually there is no commercial support. Useful contributions are welcome.