Scriptella ETL Frequently Asked Questions

Table of contents

1. General

1.1. Where can I find documentation on script syntax for driver FOO/BAR/...?

An up to date drivers documentation is available in the Scriptella Javadoc. Reference Documentation also covers drivers usage and scripts syntax.

1.2. Does Scriptella support Microsoft Access?

Microsoft Access and other databases with ODBC interface are supported via ODBC-JDBC bridge driver which comes with Sun's JRE. Download ODBC example for more details. Connection declaration example:

```
                <connection url="jdbc:odbc:DRIVER={Microsoft Access Driver
 (*.mdb)};DBQ=Northwind.mdb"/>
```

1.3. What is the difference between $variable / ${expression} and ?variable / ?{expression} syntax.

Binding variables syntax varies between drivers. JDBC drivers use the following rules for properties substitution:

- $variable - inserts a value of the variable as text content.
- ${expression} - braces are used for JEXL expressions. For example ${column1+column2} inserts a sum of 2 columns.
- ?variable, ?{expression} - syntax is the same as in 2 previous examples, but the result of evaluation is set as a prepared statement parameter, thus increasing the performance and eliminating the need to escape values.
  Please note that currently ?{} syntax is only supported by the JDBC drivers.

See Reference Manual for additional details.

1.4. How to share a mutable variable between script/query elements of an ETL file?

In general such practice in not recommended because often it's a sign of a bad design. It's like using GOTOs or global variables in programming languages. But in several cases using a global variable may help to achieve required goal. We propose 2 approaches:

1. Use etl.globals map to work with global variables. Example ${etl.globals['globalVar']}
2. Another approach is similar to the technique utilized in anonymous inner classes to modify a single-element array declared as a final variable.

The following example demonstrates both approaches:

```
<etl>
    <connection driver="script" id="js"/>
    <connection url="jdbc:...." id="db"/>

    <!-- Set number of records as a global variable -->
    <!-- Note that JEXL syntax etl.globals['globalVar'] does not work in JavaScript -->
    <query connection-id="db">
        select count(id) as c from Errors
        <script connection-id="js">
            etl.globals.put('errorsCount', c);
        <script>
    </query>
    <!-- Then reuse this variable in other parts of ETL file -->
    <script connection-id="js" if="etl.globals.get('errorsCount') gt 0">
        java.lang.System.out.println('errors count ='+etl.globals.get('errorsCount'));
    </script>

    <!-- Alternatively an outer query can be used to share an array based variable between scripts
 -->
    <query connection-id="js">
        var pseudoGlobalVar = []; //Declare pseudo-global variable available to nested element
        query.next(); //Executes child scripts
        <script> //Updates the variable
           pseudoGlobalVar[0] = 1;
        </script>
        <script> //Outputs updated value of global variable
             java.lang.System.out.println('pseudoGlobalVar[0]='+pseudoGlobalVar[0]);
        </script>
    </query>
</etl>
```

1.5. How to call a static method in JEXL expression?

Load a class by name by using class:forName Scriptella function

Example. Call System.getProperty:

```
${class:forName('java.lang.System').getProperty('propName')}
```

1.6. How to escape spaces in field names?

You can use `etl context variable` ${etl.getParameter('var name')} to reference any column names. Additionally several drivers including CSV and JDBC allow referencing columns by an index, i.e. $1, $2 ... $n.
Example:

```
<query connection-id="csv"> <!-- Read CSV file content -->
    <script connection-id="text">
        <!-- Print columns, column 2,3 are referenced by name, 1,4 - by index -->
        $1,$secondColumn,${etl.getParameter('My Column')},$4
    </script>
</query>
```

2. SQL

2.1. How to create an Oracle trigger (run a PL/SQL block)?

To recognize Oracle PL/SQL statement blocks you'd have to specify `plsql=true` connection property (supported only by Scriptella Adapter for Oracle). In this case a slash(/) on a single line is used as a statement separator:

```
<connection driver="oracle" ...>
    plsql=true
</connection>
<script>
    CREATE OR REPLACE TRIGGER secure_del_trigger
       BEFORE DELETE
       ON emp
       FOR EACH ROW
    DECLARE
       unauthorized_deletion   EXCEPTION;
    BEGIN
       IF <your business rule is violated> THEN
         RAISE unauthorized_deletion;
       END IF;
    EXCEPTION
       WHEN unauthorized_deletion
       THEN
          raise_application_error (-20500,
            'This record cannot be deleted');
    END;
    /

    -- Other statements separated with a slash on a single line
</script>
```

If you are using Oracle JDBC driver directly then set the following configuration properties:

```
<connection driver="oracle.jdbc.driver.OracleDriver" ...>
    statement.separator=/
    statement.separator.singleline=true
</connection>
```

This idea is similar to Ant solution.

2.2. How to control commit options and transactions?

The following connection parameters are supported by the JDBC bridge:

- `transaction.isolation` - Transaction isolation level name
- `autocommit` - Enables/disables auto-commit mode.
- `autocommit.size` - Enables/disables auto-commit mode.

> **Note:**
> In general avoid using `autocommit.size`, because in this case an ETL process cannot be rolled back correctly. Use this parameter only for performance critical operations (bulk inserts etc.).
> If the `autocommit` is true, then setting `autocommit.size` has no effect.

Example:

```
<connection driver="auto" url="jdbc:hsqldb:mem:test">
    autocommit.size=4 <!-- Automatically commit after every 4th statement -->
    transaction.isolation=SERIALIZABLE <!-- Sets TX level to Serializable -->
</connection>
```

2.3. How to UPSERT (update or insert into a table?)

You can leverage database-specific SQL statements like MERGE or UPSERT. See answers on StackOverflow:

- Oracle
- MySQL
- MS SQL Server

An alternative approach, which is universal but is typically slower, is to use count(*) statement and do insert or update accordingly:

```
<query connection-id="in">
    SELECT * FROM Persons_In
    <query connection-id="out">
       SELECT COUNT(*) as cnt FROM Person_Out WHERE Person_ID=?Person_ID
       <!-- If nothing found - insert -->
       <script if="cnt==0">
          INSERT INTO Person_OUT VALUES (Person_ID, Person_Name, ...);
       </script>
       <!-- Otherwise - update -->
       <script if="cnt gt 0">
          UPDATE Person_OUT SET Person_Name=?Person_Name WHERE Person_ID = ?Person_ID;
       </script>
    </query>
</query>
```

3. CSV/Text

3.1. How to output null value as empty string in a CSV/Text file.

Suppose we have 3 variables - a='valueA';b=null and c='valueC'.

Here is a CSV script which outputs them in a single line:

```
                    $a,$b,$c
```

The output for this script:

```
         valueA,$b,valueC
```

By default NULL variables are not substituted, because Scriptella substitution engine cannot distinguish null value from undeclared variable. This behaviour is not always desired. As a workaround you can specify a null_string connection property as follows:

```
<connection driver="text">
    null_string=<!-- Expand nulls to empty string -->
</connection>
```

And the output is:

```
valueA,,valueC
```