Job JobSchedulerPLSQLJob

Introduction

The JobSchedulerPLSQLJob JITL job provides a standardized and parameterized interface for executing Oracle PL/SQLs statements. The JobScheduler offers out of the box capability to execute PL/SQLs, passing parameters to the PL/SQL or collecting and passing on the results of a PL/SQL execution to next job step as a JobScheduler Order parameter. The JobSchedulerPLSQLJob can be used to execute existing PL/SQL files just by referring them in the command parameter.

A Simple JITL PL/SQL Job Example

The following example shows a basic example of the JobSchedulerPLSQLJob. It executes PL/SQL anonymous code blocks - selecting the current system date and displaying it on stdout as *order_date*.

Simple JobSchedulerPLSQLJob

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<job  title="Execute PL/SQL procedure" order="yes">
    <description >
        <include  file="jobs/JobSchedulerPLSQLJob.xml"/>
    </description>
    <params >
        <!-- Database connection parameters i.e. db_url, db_user, db_password -->
        <param  name="db_url"      value="jdbc:oracle:thin:@:1521:DORCL01"/>
        <param  name="db_user"     value="sos_scheduler"/>
        <param  name="db_password" value="sos"/>

       <!-- PL/SQL Code -->
                <param  name="command"     value="
                DECLARE
                   v_order_date DATE := SYSDATE;
                BEGIN
                        SELECT SYSDATE
                        INTO v_order_date
                        FROM DUAL;
                        DBMS_OUTPUT.PUT_LINE(' +++              +++');
                        DBMS_OUTPUT.PUT_LINE('SET order_date IS '|| v_order_date);
                        DBMS_OUTPUT.PUT_LINE(' +++              +++');
                END;
                "/>

        <!-- dbms_output to JobScheduler Order parameter parser regex -->
        <param  name="variable_parser_reg_expr" value="^SET\s+([^\s]+)\s*IS\s+(.*)$"/>
    </params>
    <script  language="java" java_class="sos.scheduler.db.JobSchedulerPLSQLJobJSAdapterClass"/>
    <run_time />
</job>
```

Parameters

The JobSchedulerPLSQLJob requires the following parameters:

Name	Title	Mandatory	Default	Example
command	PL/SQL statements to be executed	true		`select sysdate from dual`
db_url	JDBC connection string	true		`jdbc:oracle:thin:@localhost:1521:XE`
db_user	User name for database access	true		db username
db_password	Password for database access	true		db password
variable_parser_reg_expr	Regular expression to parse dbms_output and set order parameters for next job steps	false	^SETs+(\\s)\\s*ISs(.*)$	

command

- The PL/SQL code can be:
  - saved to a separate file such as get_order.sql . This file can subsequently be referred to as the value of the "command" job parameter. This is a recommended approach for achieving "separation of concern" in application architecture.

    ```
    <param  name="command" value="config/live/commn/sqls/get_order.sql"/>
    ```

  - PL/SQL code can also be specified as the value of the *command* parameter, with the entire PL/SQL being written as part of the Job. XML. This approach is preferred if the PL/SQL code is very small and only used by a single job.

```
<param  name="command" value="
               DECLARE
                  v_order_date DATE := SYSDATE;
               BEGIN
                       SELECT SYSDATE
                       INTO v_order_date
                       FROM DUAL;
                       DBMS_OUTPUT.PUT_LINE(' +++              +++');
                       DBMS_OUTPUT.PUT_LINE('SET order_date IS '|| v_order_date);
                       DBMS_OUTPUT.PUT_LINE(' +++              +++');
               END;
      "/>
```

db_url

JITL needs a standard JDBC database connection string such as *jdbc:oracle:thin:@localhost:1521:XE*

db_user

DB Username which has necessary database permission for executing the PL/SQL code.

db_password

The password for the DB user defined in the db_user parameter.

variable_parser_reg_expr

This parameter defines a regular expression for parsing the dbms_output from the PL/SQL execution and sets the order parameters for subsequent job steps. For example, the dbms ouput DBMS_OUTPUT.PUT_LINE('SET order_date IS '|| v_order_date) displays the output on console SET order_date is 20140915, it will be parsed by regular expression *^SETs+(\ls)\ls*\Ss(.*)$* will result as order parameter *order_date="20140915".*

Saving Database Connection Settings in a Parameter File

It strongly recommend that a db_connection parameter file such as *database_connection.parameter.xml* is used to store all the database connection settings in a common location. This approach enables the user to manage settings at central location which can then be reused by multiple jobs.

This approach also makes it easy to maintain different settings for development, integration and production environments.

The following shows an example database connection parameter file:

database_connection.parameter.xml

```
<params >
       <param  name="db_url"      value="jdbc:oracle:thin:@:1521:DORCL01"/>
       <param  name="db_user"     value="sos_scheduler"/>
       <param  name="db_password" value="sos"/>
<params >
```

The next example shows a JITL job where the database connection parameters are stored in an external file. In this example a "*common_settings /database"* directory has been created inside the JobScheduler's *live* folder.

JobSchedulerPLSQLJob with database_connection_settings file

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<job  title="Execute PL/SQL procedure" order="no">
    <description >
        <include  file="jobs/JobSchedulerPLSQLJob.xml"/>
    </description>
    <params >

       <!-- Database connection parameters i.e. db_url, db_user, db_password -->
       <include  live_file="../common_settings/database/database_connection.params.xml" node=""/>

      <!-- PL/SQL Code -->
       <param  name="command"     value="
                DECLARE
                   v_order_date DATE := SYSDATE;
                BEGIN
                        SELECT SYSDATE
                        INTO v_order_date
                        FROM DUAL;
                        DBMS_OUTPUT.PUT_LINE(' +++              +++');
                        DBMS_OUTPUT.PUT_LINE('SET order_date IS '|| v_order_date);
                        DBMS_OUTPUT.PUT_LINE(' +++              +++');
                END;
                "/>

        <!-- dbms_output to JobScheduler Order parameter parser regex -->
        <param  name="variable_parser_reg_expr" value="^SET\s+([^\s]+)\s*IS\s+(.*)$"/>
    </params>
    <script  language="java" java_class="sos.scheduler.db.JobSchedulerPLSQLJobJSAdapterClass"/>
    <run_time />
</job>
```

Passing parameters to the PL/SQL

JobScheduler order parameters can be passed to the PL/SQL. PL/SQL code can be parameterized by defining variables such as ${SCHEDULER_PARAM_*VARIABLE_NAME*}. Variables can be set using environment variables, JobScheduler task parameters (as described in the following example) or from JobScheduler order parameters.

Passing variables to the PL/SQL

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<job  title="Execute PL/SQL procedure" order="no">
    <settings >
        <log_level ><![CDATA[debug9]]></log_level>
    </settings>
    <description >
        <include  file="jobs/JobSchedulerPLSQLJob.xml"/>
    </description>
    <params >

            <!-- Database connection parameters i.e. db_url, db_user, db_password -->
        <include  live_file="../common_settings/database/database_connection.params.xml" node=""/>

            <!-- Parameter can be passed by task or as order param -->
                <param  name="date_mask" value="YYYYMMDD_HH24MI"/>

       <!-- PL/SQL Code -->
       <param  name="command" value="
                DECLARE
                        v_order_date VARCHAR2(16) := SYSDATE;
                BEGIN
                    /* recommended to set variables in the PL/SQL is with  ${SCHEDULER_PARAM_VARIABLE_NAME} */
                        SELECT to_char(SYSDATE, '\${SCHEDULER_PARAM_DATE_MASK}' )
                          INTO v_order_date
                          FROM DUAL;
                        DBMS_OUTPUT.PUT_LINE(' +++              +++');
                        DBMS_OUTPUT.PUT_LINE('SET order_date IS '|| v_order_date);
                        DBMS_OUTPUT.PUT_LINE(' +++              +++');
                END;   "/>

        <!-- dbms_output to JobScheduler Order parameter parser regex -->
        <param  name="variable_parser_reg_expr" value="^SET\s+([^\s]+)\s*IS\s+(.*)$"/>
    </params>
    <script  language="java" java_class="sos.scheduler.db.JobSchedulerPLSQLJobJSAdapterClass"/>
    <run_time />
</job>
```

Parameters can also be defined with following syntax:

- %parameter_name%
- ${SCHEDULER_PARAM_parameter_name}

Parameters are not case sensitive.

When PL/SQL code is part of Job XML file, then parameters should be defined in the form \${SCHEDULER_PARAM_PARAMETER_NAME}. If PL/SQL code is read from file system, the parameter can be defined without the "**\"**

PL/SQL script as an External File

PL/SQL code can be defined directly inside the Job xml as *a command* parameter value but is generally better stored on the file system. JITL jobs can be configured to read PL/SQL scripts from the file system by defining the script path as a value for the *command* parameter i.e.

In the following example the PL/SQL code is saved to the filesystem in *C:\app\executables\plsql\get_order_date.sql* and subsequently referenced using the *command* parameter.

Passing variables to the PL/SQL

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<job  title="Execute PL/SQL procedure" order="no">
    <settings >
        <log_level ><![CDATA[debug9]]></log_level>
    </settings>
    <description >
        <include  file="jobs/JobSchedulerPLSQLJob.xml"/>
    </description>
    <params >

            <!-- Database connection parameters i.e. db_url, db_user, db_password -->
        <include  live_file="../common_settings/database/database_connection.params.xml" node=""/>

            <!-- Parameter can be passed by task or as order param -->
                <param  name="date_mask" value="YYYYMMDD_HH24MI"/>

        <!-- PL/SQL script from filesystem -->
            <param  name="command" value="C:/app/executables/plsql/get_order_date.sql"/>

        <!-- dbms_output to JobScheduler Order parameter parser regex -->
        <param  name="variable_parser_reg_expr" value="^SET\s+([^\s]+)\s*IS\s+(.*)$"/>
    </params>
    <script  language="java" java_class="sos.scheduler.db.JobSchedulerPLSQLJobJSAdapterClass"/>
    <run_time />
</job>
```

Passing PL/SQL results to subsequent job steps as parameters

JobScheduler jobs can create and update JobScheduler Order parameters. The JobSchedulerPLSQLJob can also pass on the result of PL/SQL execution i.e. calculated dates, parameters calculated from tables, etc. By default the JobSchedulerPLSQL job defines a regular expression to parse dbms_output from the execution of PL/SQLs and sets order parameters for subsequent job steps. For example, the DBMS_OUTPUT.PUT_LINE('SET order_date IS '|| v_order_date) dbms ouput displays the output on console; if SET order_date is 20140915, it will be parsed by regular expression *^SETs+(Is)Is*ISs(.*)$* and return the *order_date="20140915"* order parameter. All dbms_output statements matching the *^SETs+(Is)Is*ISs(.*)$* regular expression will be set as order_parameters.

Advanced Configuration

Generic job for executing multiple PL/SQLs

The JobSchedulerPLSQLJob can be configured as a generic node inside a job chain and executable PL/SQL script can be defined as an order parameter. The following example shows such a generic job. The job chain has a job node - *execute_plsql* - two orders - *get_order_date* and *get_last_booking_date*. Each order is scheduled to be executed at a different time. Both the orders are configured to use a different PL/SQL script file i.e. *get_order_date.sql* and *get_last_booking_date.sql*.

- JobChain

  JITL-PLSQL.job_chain.xml

  ```
  <?xml version="1.0" encoding="ISO-8859-1"?>
  <job_chain  orders_recoverable="yes" visible="yes">
      <job_chain_node  state="execute_plsql" job="JITL-PLSQL" next_state="sucess" error_state="error"/>
      <job_chain_node  state="sucess"/>
      <job_chain_node  state="error"/>
  </job_chain>
  ```

- Job

JITL-PLSQL.job.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<job  title="Execute PL/SQL procedure" order="yes">
    <settings >
        <log_level ><![CDATA[debug9]]></log_level>
    </settings>
    <description >
        <include  file="jobs/JobSchedulerPLSQLJob.xml"/>
    </description>
    <params >
         <!-- Parameter can be passed by task or as order param -->
        <param  name="date_mask" value="YYYYMMDD_HH24MI"/>

        <!-- Database connection parameters i.e. db_url, db_user, db_password -->
        <include  live_file="../common_settings/database/database_connection.params.xml" node=""/>

        <!-- dbms_output to JobScheduler Order parameter parser regex -->
        <param  name="variable_parser_reg_expr" value="^SET\s+([^\s]+)\s*IS\s+(.*)$"/>
    </params>
    <script  language="java" java_class="sos.scheduler.db.JobSchedulerPLSQLJobJSAdapterClass"/>
    <run_time />
</job>
```

- Order : get_order_date

JITL-PLSQL,get_order_date.order.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<order  title="Calculate Order Date">
    <params >
       <!-- PL/SQL script file -->
        <param  name="command" value="C:/app/executables/plsql/get_last_booking_date.sql"/>
    </params>
    <run_time  let_run="no">
        <period  single_start="08:00"/>
    </run_time>
</order>
```

- Order : get_last_booking_date

JITL-PLSQL,get_last_order_date.order.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<order  title="Calculate last booking date">
    <params >
        <!-- PL/SQL script file -->
        <param  name="command" value="C:/app/executables/plsql/get_last_booking_date.sql"/>
    </params>
    <run_time  let_run="no">
        <period  single_start="11:00"/>
    </run_time>
</order>
```

Standalone PL/SQL Jobs

If PL/SQL code needs to be parameterized by a job parameter the syntax for parameter substitute is different compare to order jobs.

Since the syntax of suffixing an order parameter for SCHEDULER_PARAM is not required the parameter name can directly be substituted in the PL/SQL code.

See the following example for two variant for standalone PL/SQL code.

PL/SQL Code as script

plsql_job_param_script.job.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<job  order="no" title="test">
    <settings >
        <log_level ><![CDATA[debug9]]></log_level>
    </settings>
    <params >
        <param  name="testparam" value="test"/>
        <param  name="db_class" value="SOSOracleConnection"/>
        <param  name="db_driver" value="oracle.jdbc.driver.OracleDriver"/>
        <param  name="db_url" value="jdbc:oracle:thin:@8of9:1521:TEST"/>
        <param  name="db_user" value="scheduler"/>
        <param  name="db_password" value="scheduler"/>
    </params>
    <script  language="java" java_class_path="" java_class="sos.scheduler.db.JobSchedulerPLSQLJobJSAdapterClass"
>
        <![CDATA[
         BEGIN
                     INSERT INTO T_SOS_TEST VALUES ('${testparam}');
             END;
        ]]>
    </script>
    <run_time />
</job>
```

PL/SQL Code as command

plsql_job_param_command.job.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<job  order="no" title="test">
    <settings >
        <log_level ><![CDATA[debug9]]></log_level>
    </settings>
    <params >
        <param  name="testparam" value="test_command"/>
        <param  name="db_class" value="SOSOracleConnection"/>
        <param  name="db_driver" value="oracle.jdbc.driver.OracleDriver"/>
        <param  name="db_url" value="jdbc:oracle:thin:@8of9:1521:TEST"/>
        <param  name="db_user" value="scheduler"/>
        <param  name="db_password" value="scheduler"/>
        <param  name="command" value="BEGIN             INSERT INTO T_SOS_TEST VALUES
('\${testparam}');             END;"/>
    </params>
    <script  language="java" java_class_path="" java_class="sos.scheduler.db.JobSchedulerPLSQLJobJSAdapterClass"
/>
    <run_time />
</job>
```

Return parameters created by the JobSchedulerPLSQLJob

The JobScheduler automatically creates the following order parameters, which will be available to subsequent job steps as order parameters.

sql_error

- The sql_error parameter contains all the error messages generated during the PL/SQL execution. This parameter will be empty if no errors occur.

std_out_output

- The std_out_output parameter contains all the messages spooled to stdout by PL/SQL.

See also:

- How to work with PL/SQL and the Oracle DBMS
- How to run Oracle Stored Procedures using PL/SQL
- The JobSchedulerPLSQLJob JITL job documentation