

How to run Oracle Stored Procedures using PL/SQL

This article describes how to launch an Oracle™ RDBMS stored procedure using the JobSchedulerPLSQLJob JITL Job. This job is used standalone or triggered by orders to execute [pl/sql](#) statements in a database. These can be stored procedures or [SQL](#) statements as well.

The example also shows, how to send the result (output) of an [pl/sql](#) job as an email.

The documentation of the job [JobSchedulerPLSQLJob](#) can be found in the `./jobs` directory of the JobScheduler installation.

The [SQL](#) commands are defined using the `command` parameter.

It is possible to define more than one command as value of the `command` parameter.

Such commands are then carried out in the order in which they are written and must be separated by a semicolon and a subsequent new line.

You have to use the character sequence `
` for a newline.

Example of a stored procedure:

```
CREATE OR REPLACE PROCEDURE myTestProc IS
vCounter    NUMBER := 0;
BEGIN
  select count(*) into vCounter from SCHEDULER_HISTORY;
  if vCounter>0 then
    dbms_output.put_line ('Set plsqli_result IS The value of variable "vCounter" is: [|vCounter|]');
  end if;
END;
```

Please note that the output begins with "Set plsqli_result". This will create an order parameter which can be used for example in the body of an email.

The example defines a job chain with two steps. First step is executing the [pl/sql](#) job and the second step is sending an email.

The following is an example of a job in which the command contains one statement.

Parameter

Parameter	Environment	Includes
Name	command	Value
		begin myTestProc; end;
Name	db_url	Value
	command	begin myTestProc; end;
	db_url	jdbc:oracle:thin:@ur-lAsss:1521:XE
	db_user	scheduler
	db_password	scheduler



The xml configuration of the [pl/sql](#) job

```
<job title="Execute PL/SQL procedure" order="yes" name="TestSQL">
  <params >
    <param name="command" value="begin myTestProc; end;"/>
    <param name="db_url" value="jdbc:oracle:thin:@ur-lAsss:1521:XE"/>
    <param name="db_user" value="scheduler"/>
    <param name="db_password" value="`getDBPassword`"/>
  </params>
  <script language="java" java_class="sos.scheduler.db.JobSchedulerPLSQLJobJSAdapterClass"/>
</job>
```

Running this job will produce output in the log

```
2013-04-03 10:29:21.477 [info] Set plsql_result IS The value of variable "vCounter" is: [5495]
```

The SendEmail job comes from the JITL library.

SendMail - Send Mails

General Job Attributes

Job Name	SendMail
Job Title	Send Mails
Process Class	>
Language	java

Script [Java](#) [Includes](#) [Process File](#) [Options](#) [eMail](#) [SetBack](#) [Documentation](#) [XML](#)

Classname	sos.scheduler.managed.JobSchedulerManagedMailJob
Classpath	
Java Options	

```
<job title="Send Mails" order="yes" stop_on_error="no" name="SendMail">
  <description >
    <include file="jobs/JobSchedulerManagedMailJob.xml" />
  </description>
  <params >
    <param name="to" value="info@sos-berlin.com"/>
    <param name="subject" value="Result from oracle"/>
    <param name="host" value="smtp_host"/>
  </params>
  <script language="java" java_class="sos.scheduler.managed.JobSchedulerManagedMailJob"/>
  <monitor name="configuration_monitor" ordering="0">
    <script java_class="sos.scheduler.managed.configuration.ConfigurationOrderMonitor" language="java"/>
  </monitor>
  <run_time />
</job>
```

These jobs will be chained in a job chain:

Jobs

- SendMail - Send Mails
- TestSQL - Execute PL/SQL

Job Chains

- plsql
 - Steps/Nodes
 - TestSQL - Exec...
 - SendMail - Sen...
 - Nested Job Chains
 - Orders

Job Chain Orders

Job > [] Browse

Next State [] Delay [] Import Job

Error State [] On Error [] New Chain Node

Full Node
 End Node
 File Sink
 Remove File
 Move to []

State	Node	Job Directory	Next State	Error State	On f
100	Job	TestSQL	200	error	
200	Job	SendMail	success	error	
success	Endnode				
error	Endnode				

Reorder
 Parameter
 Add Missing Nodes
 Remove Node

File Order Sources for plsql

The second step of the job chain defines a node parameter for the body (the other parameters like subject, smtp-server are defined as job parameters).

Detail Parameter

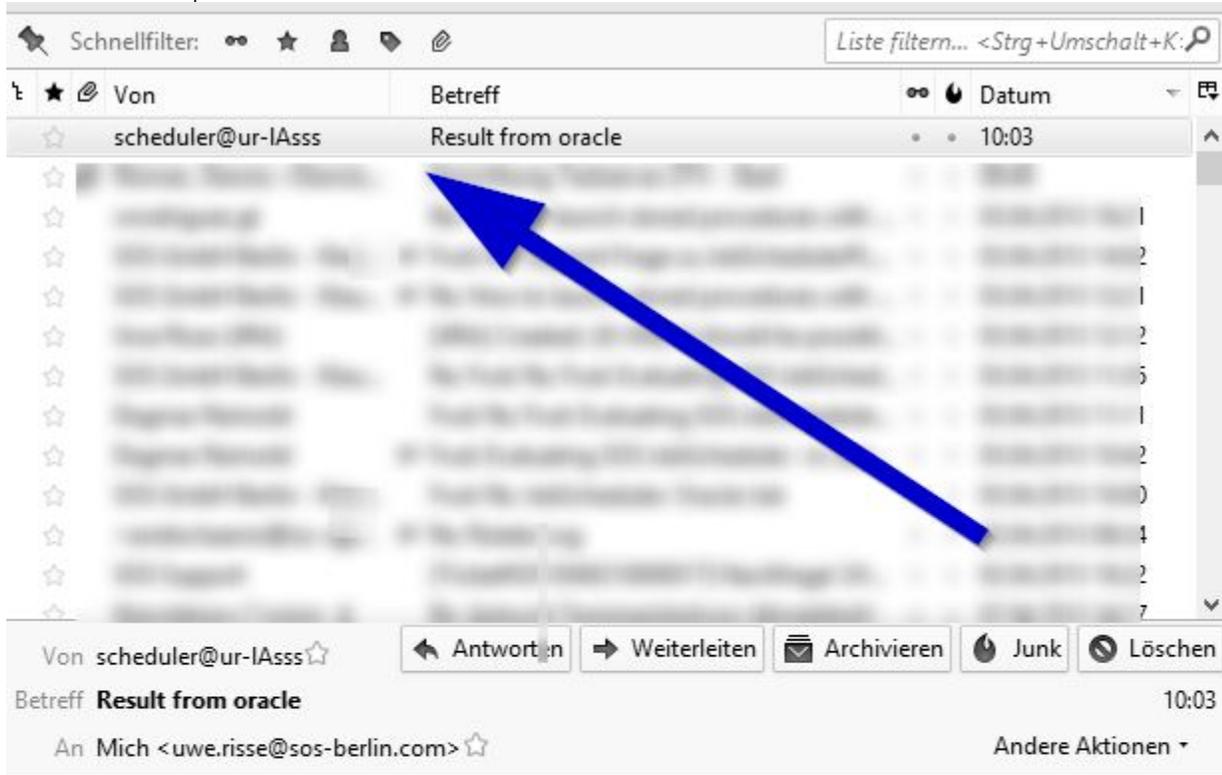
Name [] Value [] Text [] Apply []

Name	Value	Text
body	#{plsql_result}	

New []
 []
 []
 Wizard []
 Apply Details []
 Cancel []
 de []

Using the output from pl/sql job as content of the body

When the second step is executed an email will be sent.



The screenshot shows an email client interface. At the top, there is a search bar labeled 'Schnellfilter:' and a filter button 'Liste filterm... <Strg+Umschalt+K:'. Below this is a table of email headers with columns 'Von', 'Betreff', and 'Datum'. The selected email is from 'scheduler@ur-lAsss' with the subject 'Result from oracle' and the time '10:03'. A blue arrow points from the bottom right towards the subject line. Below the header table is a toolbar with buttons for 'Antworten', 'Weiterleiten', 'Archivieren', 'Junk', and 'Löschen'. Below the toolbar, the email details are shown: 'Von scheduler@ur-lAsss', 'Betreff Result from oracle', and 'An Mich <uwe.risse@sos-berlin.com>'. The time '10:03' is also displayed.

The value of variable vCounter is: [5387]



See also:

If you want to run [SQL*Plus™](#) scripts, you can use the [SqlPlus-Job: SOSSQLPlusJob](#)

If you are not using Oracle™ RDBMS and/or you want to execute just a (simple) `{{sql}}` command, please see the [JobSchedulerManagedDatabaseJobSO SHibernate](#) job.