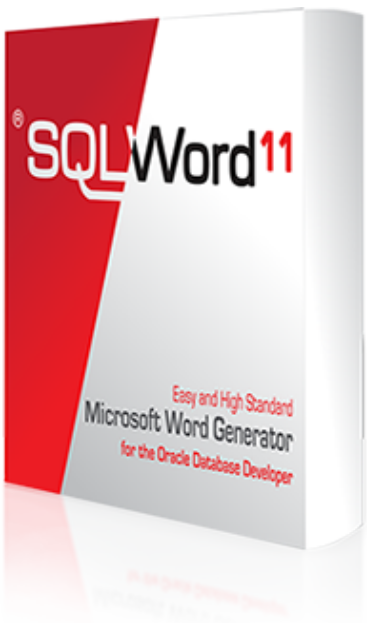
SQLWord 11

User's Guide and Reference

Release 11.0.5

Februari 2021



Contents

General information

Introduction

All Oracle database users know that it is quite difficult to retrieve Oracle data into Microsoft Word documents. Sequel Services has developed a report generator which gives you a powerful tool to solve this problem.

You can create your source documents in Microsoft Word and use SQLWord to generate Microsoft Word documents merged with data from your Oracle database.

The existing reporting tools mostly use their own specific format and don't integrate at all with Microsoft Word documents. Mailmerge in Microsoft Word has very limited possibilities and is difficult to use. It is not possible to create master-detail documents.

When using SQLWord you can create your own standard letters, contracts and reports, integrating with the data of your Oracle database.

Using the SQLWord Developer application, you can place several PL/SQL-statements enclosed by <% tag %> scriptlets inside the text of a Microsoft Word document. SQLWord follows the syntax of Oracle PSP (PL/SQL Server Pages) <% tag %> declarations.

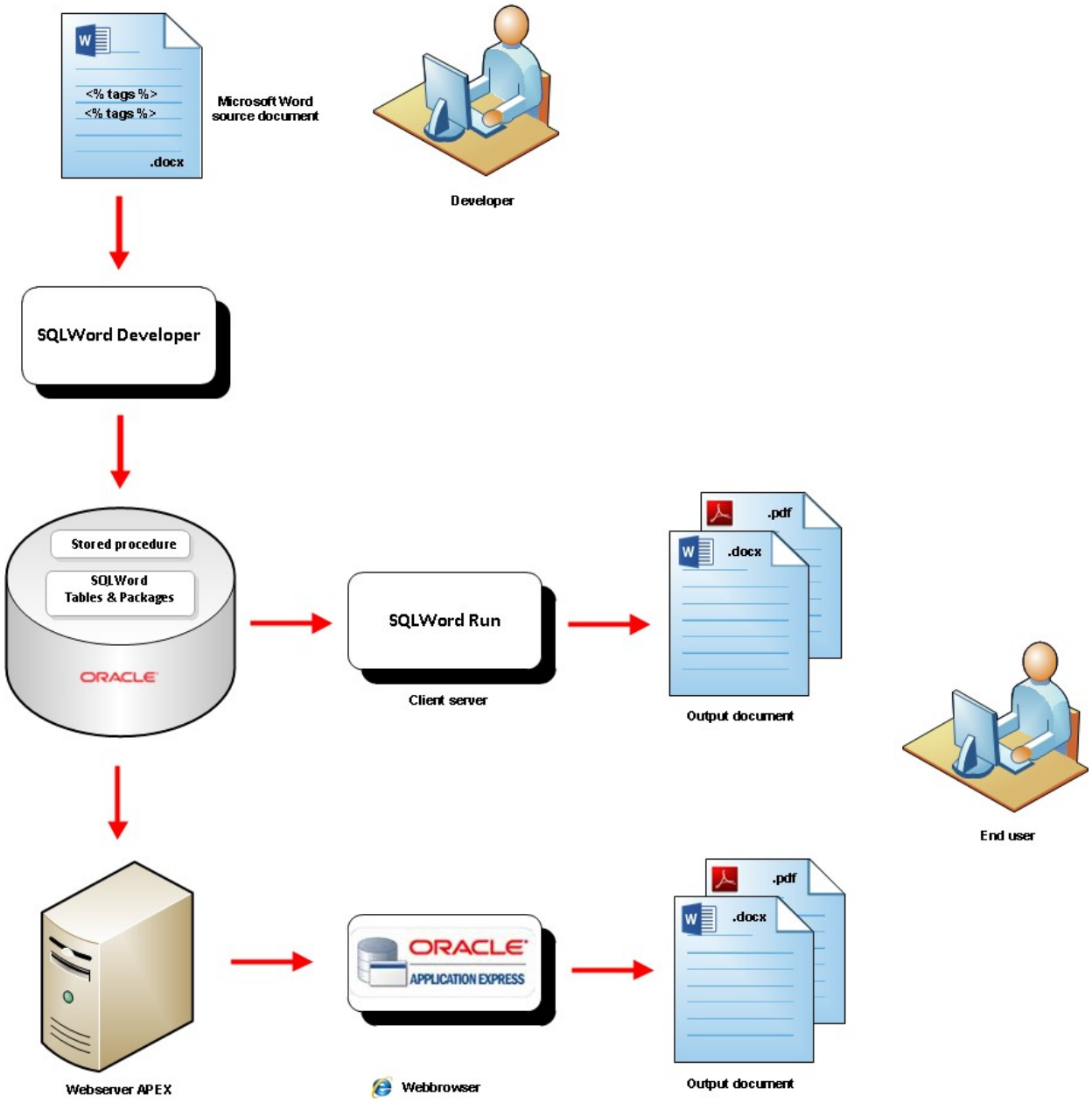
Microsoft Word documents with <% tag %> scriptlets can be stored by the SQLWord Developer application into your Oracle database, compiled as PL/SQL-procedures. By calling the PL/SQL procedure that you created from your Microsoft Word document, SQLWord retrieves the data from your Oracle database and integrates it in the generated output document.

If you run SQLWord "client-server" by using the SQLWord Run application, the output document is created on the LAN and is opened with Microsoft Word.

If you run SQLWord from an Apex application, the output document is send to the webbrowser on the client, where the output document is opened with Microsoft Word. An Apex5 demo implementation example is included.

SQLWord uses the **DOCX format**. This format is internal based on XML.

SQLWord architecture



Software requirements

Server side

- Oracle 11g / Oracle 12c

Client

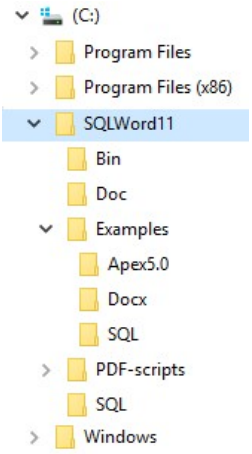
- Windows7, Windows8, Windows10. SQLWord can also run as a 32-bits application on Windows 64-bit OS.
- Microsoft Word 2016 / 2013 / 2010 / 2007 / 2003. For Microsoft Word 2003 you need to install the Microsoft Office Compatibility Pack 2007 for supporting the DOCX-format.
- 32-bits Oracle Client (11g / 12c) for Microsoft Windows. SQLWord is a 32-bits application and cannot connect to an 64-bit Oracle client.

Installation

Setup

From the file manager double-click on the file named install_sqlword11.exe or install_sqlword11_eval.exe to start the setup program. Please follow the on-screen installation directions.

All necessary files will be installed by default in a folder at *C:\SQLWord11*



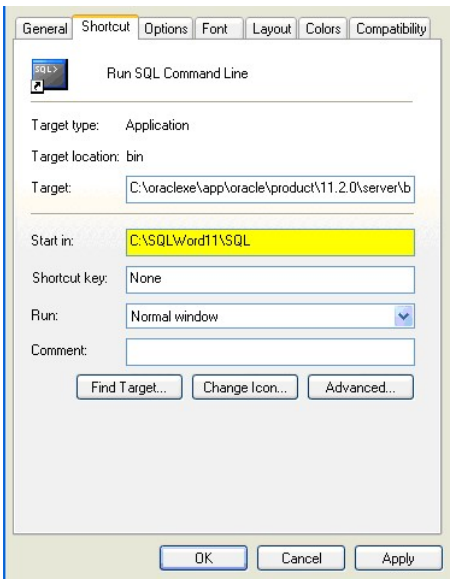
Create SQLWord tables and the SQLWord packages

Before installing the SQLWord tables and package we suggest to create a new user SQLWORD_DEMO for evaluation purposes with the good old SQL*Plus:

```
SQL> connect to a user with DBA-rights …

SQL> create user sqlword_demo identified by sqlword_demo
     default tablespace USERS;

SQL> grant connect, resource to sqlword_demo;
SQL> grant create view to sqlword_demo;
```

Create a shortcut on your desktop for SQL*Plus and specify the default directory

Start in: C:\SQLWord11\SQL



- If your Oracle database is **XE** then first you must grant several sys-privileges to the SQLWORD_DEMO user. Start SQL*Plus from this shortcut and run script **sys_grants.sql** (as sysdba).

```
SQL> @sys_grants.sql

Grant succeeded.
Grant succeeded.
Grant succeeded.
Grant succeeded.
Grant succeeded.
```

- Now start SQL*Plus from this shortcut, connect to user SQLWORD_DEMO and run the installation script **install_sqlword11.sql**

```
SQL> connect sqlword_demo/sqlword_demo

SQL> @install_sqlword11.sql

Table created.
Index created.
Table altered.
Table created.
Index created.
Table altered.
Table altered.
Table created.
Index created.
Table altered.
Table altered.
Table created.
Index created.
Table altered.
Package created.
Package body created.
etc …
```

When the installation script is finished check the log-files **install_sqlword11.log** and **install_sqlword11_demo.log** for any errors.

Upgrade from previous release

If you already have installed a previous SQLWord 11 release then you only need to update the sqlword package.

- Start SQL*Plus, connect to user SQLWORD_DEMO and run the upgrade script **upgrade_sqlword11.sql**

```
SQL> connect sqlword_demo/sqlword_demo

SQL> @upgrade_sqlword11.sql

Upgrading SQLWord 11.0.4.0 ...

Package created.

Package-body created.
```

License key

To install the SQLWord license key you need to run the supplied license script **sqlword_license.sql** using SQL*Plus.

```
SQL> @sqlword_license.sql
```

The SQLWord license key is inserted into table SQLWORD_PARAMETER and is verified every time you run SQLWord.

You can display the license information with SQL*Plus:

```
SQL> select sqlword.show_license from dual;

SHOW_LICENSE
--------------------------------------------------------------------------
SQLWord is licensed to <company-name> for <number> users on Oracle
database server <oracle-database-server-name>.
```

SQLWord Developer

Introduction

SQLWord Developer is a 32-bits Windows application to support users in the development of Microsoft Word template documents.

The SQLWord Developer application window is always displayed on top of all other applications so you can edit Microsoft Word documents and always have access to the SQLWord Developer application.

The SQLWord Developer application contains a button toolbar and a work area where you can edit <% tag %> scriptlets with PL/SQL-statements.

Toolbar buttons:

Shows a submenu where you can:

- Connect to your Oracle database.
- Display the options-window (described later in this section).
- Display this Users Guide & Reference.
- Show the about box.

Create a new Word document.

Open a Word document. If you select multiple files you will get a new screen where you can compile the selected files in one run.

Save the active Word document.

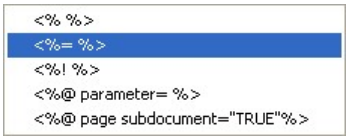
Undo the last Word command.

Find <% tag %> scriptlets in the active Word document and copy to the work area.

Paste the <% tag %> scriptlets from the work area into the active Word document.

Clear the work area.

Insert standard <% tag %> scriptlets into the work area from a submenu.



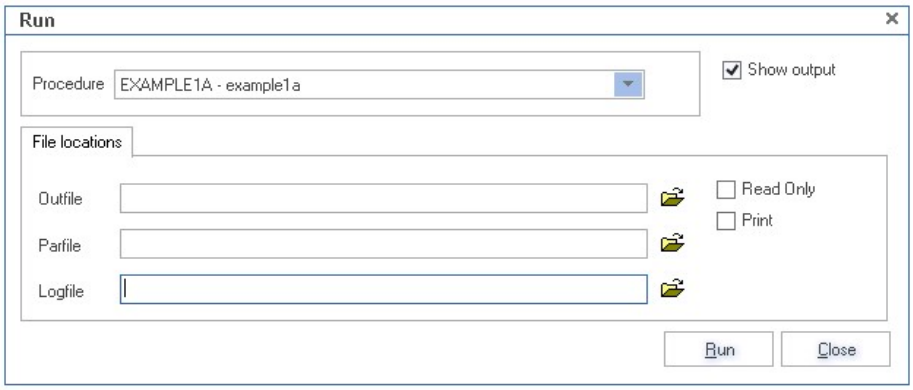
Clear all <% tag %> scriptlets in the active Word document from invisible formatting code.

Compile the active Word document to a stored procedure.

Show the stored procedure source code from the active Word document and edit several parameters (descriptions & lookup SQL-statements).

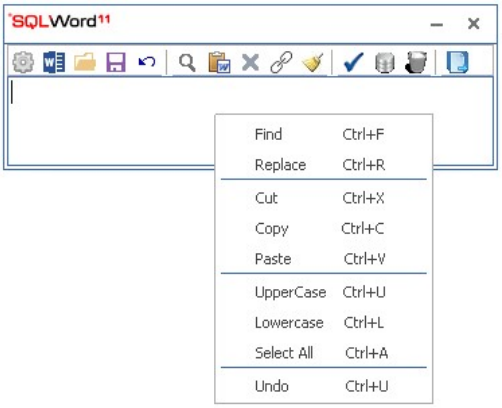
Remove stored procedures & content from the Oracle database.

Run SQLWord. The screen below shows up where you can select a report and specify values for file locations. When pressing on the Run button the screen below shows up where you can specify the input-parameters.



Popup-menu

When pressing on the right mouse button in the work area a popup-menu appears:



Scriptlets

SQLWord follows the syntax of Oracle PSP (PL/SQL Server Pages) <% tag %> declarations.

Scripting tags are enclosed within the **<%** and **%>** delimiters and the first character(s) after the opening delimiter **<%** determine the type of the scripting tag.

The following describe each scripting tag in detail.

Declarations <%! {plsql_declaration} %>

The declaration tag can be used to declare types, cursors and also define local procedures as well. Note that you need the **!** sign in this syntax.

Example: Declaring variables.

```
<%!
--
v1   number;
v2   varchar2(10) := '1234567890';
--

%>
```

Example: Declaring a cursor c1.

```
<%!
--
cursor c1
is
select emp.first_name || ' ' || emp.last_name as employee
,      emp.salary
,      job.job_title
from   employees emp
,      jobs job
where  emp.job_id = job.job_id
order by emp.last_name;
--
%>
```

Statements <% {plsql_statement} %>

All PL/SQL statements can be used such as for loops, assignments, calls to other stored procedures, etc. Note that a terminating semicolon is needed where PL/SQL requires it.

Example: for loop.

```
<%for r1 in c1 loop%>

<%end loop;%>
```

NB: Always place every loop statement on a new line !!!

and <

Example: assignments.

```
<%
--
a := 'ABC' || 'DEF';
b := a || 'GHI';
c := my_procedure(a, b);
--
%>
```

Example: local block.

```
<%
--
declare
  a varchar2(10) := 'ABCDEF';
  b varchar2(10);
begin
  b := a || 'GHI';
end;
--
%>
```

Example: exception handler.

```
<%
exception
  when NO_DATA_FOUND then null;
%>
```

Expressions <%= {plsql_expression} %>

The expression tag returns the value of any PL/SQL expression including PL/SQL function calls and places the return value into the output document. Note that a terminating semicolon is not allowed.

Example:

```
<%= 10 + 2 %>
<%= 'ABC' || 'EFG' %>
<%= to_char(sysdate, 'dd.mm.yyyy hh24:mi:ss') %>
<%= r1.job%>
```

Example:

```
<%!
--
cursor c1
is
select emp.first_name || ' ' || emp.last_name as employee
,      emp.salary
,      job.job_title
from   employees emp
,      jobs job
where  emp.job_id = job.job_id
order by emp.last_name;
--
%>

<%for r1 in c1 loop%>

<%= r1.employee%>     <%= r1.salary%>   <%= r1.job_title%>

<%end loop;%>
```

Parameters <%@plsql parameter={ ... } %>

The parameter tag declares an input parameter to the document:

parameter="<name>" [type="(varchar2 | number | date)"] [default="<default_value>"]

The VARCHAR2 is default type. Default text values must be enclosed within single quotes within the double quotes, eg default="'xyz'".

Example:

```
<%@ plsql parameter="P_EMPLOYEE_ID" type="number"%>
<%@ plsql parameter="P_LAST_NAME" default="'KING'" %>
```

Include <%include file={filename}%>

The include tag can be used to include PLSQL declarations from an external file:

file={filename}

Example:

```
<%@ include file="example1b.plsql"%>
```

Hints

#SQLWORD_FORMAT

You can add the #SQLWORD_FORMAT hint in select statements to inform SQLWord that you want to add extra XML formatting code (ie to set text bold or set colors, etc).

This option is meant for advanced MSWord users who are well known with the Microsoft Word XML formatting language. For more information about XML formatting check this nice website: http://www.lenzconsulting.com/wordml

Example:

```
select initcap(emp.first_name || ' ' || emp.last_name) as employee
,      case
         when emp.salary > 3000 then
         '#SQLWORD_FORMAT</w:t><w:rPr><w:i/><w:color
          w:val="FF0000"/></w:rPr><w:t>'  ||
          trim(to_char(emp.salary, 'L999G999G999')) ||
          '</w:t></w:r><w:r><w:t>'
         else
           trim(to_char(emp.salary, 'L999G999G999'))
       end as salary
,      job.job_title
from   employees emp
,      jobs job
where  emp.manager_id = p_manager_id
and    emp.job_id = job.job_id
order by emp.last_name;
```

Examine example1d.docx to see how it works.

Examples

You can find several SQLWord templates at: C:\Program Files\Sqlword11\Examples\Docx

- Example1a.docx
- Example1b.docx
- Example1c.docx
- Example1d.docx
- Example2.docx
- Example3.docx
- Example4.docx
- Example5.docx
- Example6.docx
- HR_Dept.docx
- HR_Dept_Selection.docx
- HR_Employee_Contract.docx
- HR_Employee_Job_Offer.docx
- HR_Employee_Selection.docx
- HR_Manager.docx
- HR_Tables_Report.docx

Example1a.docx

Redwood, <%=to_char(sysdate,'fm dd month yyyy')%>

<%for r1 in c1(p_employee_id) loop%>
Dear <%=r1.manager%>,

We inform you about the current salary of your employees:

Employee	Job	Salary
<%for r2 in c2 (r1.manager_id) loop%><%=r2.employee%>	<%=r2.job_title%>	<%=r2.salary%>
<%end loop;%>		

Sincerely,

Larry Ellison
<%end loop;%>

```
<%@ plsql parameter="P_EMPLOYEE_ID" "type="number"%>
<%!
--
cursor c1 (p_employee_id number)
is
select p_employee_id as manager_id
,      initcap(first_name || ' ' || last_name) as manager
,      to_char(sysdate,'dd month yyyy') today
from   employees
where  employee_id = p_employee_id;
--
cursor c2 (p_manager_id number)
is
select initcap(emp.first_name || ' ' || emp.last_name) as employee
,      trim(to_char(emp.salary, 'L999G999G999'))         as salary
,      job.job_title
from   employees emp
,      jobs job
where  emp.manager_id = p_manager_id
and    emp.job_id = job.job_id
order by emp.last_name;
--
```

Example1a.docx shows how to generate a letter by using <% tags %> described in the previous section. It uses the following tags:

- Declaration-tag:

```
<%!
--
cursor c1 (p_employee_id number)
is
select p_employee_id as manager_id
,      initcap(first_name || ' ' || last_name) as manager
,      to_char(sysdate,'dd month yyyy') today
from   employees
where  employee_id = p_employee_id;
--
cursor c2 (p_manager_id number)
is
select initcap(emp.first_name || ' ' || emp.last_name) as employee
,      trim(to_char(emp.salary, 'L999G999G999'))       as salary
,      job.job_title
from   employees emp
,      jobs job
where  emp.manager_id = p_manager_id
and    emp.job_id = job.job_id
order by emp.last_name;
--
%>
```

- Parameter-tag:

```
<%@ plsql parameter="P_EMPLOYEE_ID" "type="number"%>
```

- Statement-tags:

```
<%for r1 in c1(p_employee_id) loop%>

<%for r2 in c2 (r1.manager_id) loop%><%=r2.employee%>

<%end loop;%>
```

- Assignment-tags:

```
<%=to_char(sysdate,'fm dd month yyyy')%>

<%=r1.manager%>

<%=r2.employee%>

<%=r2.job_title%>

<%=r2.salary%>
```

Compiling

When pressing the compile button SQLWord Developer creates a stored procedure with the name EXAMPLE1A. Examine the PL/SQL source code on the next page to see how the <% tags %> are placed.

Redwood, 3 januari 2014

Dear Steven King,

We inform you about the current salary of your employees:

Employee	Job	Salary
Gerald Cambrault	Sales Manager	€11.000
Lex De Haan	Administration Vice President	€17.000
Alberto Errazuriz	Sales Manager	€12.000
Adam Fripp	Stock Manager	€8.200
Michael Hartstein	Marketing Manager	€13.000
Payam Kaufling	Stock Manager	€7.900
Neena Kochhar	Administration Vice President	€17.000
Kevin Mourgos	Stock Manager	€5.800
Karen Partners	Sales Manager	€13.500
Den Raphaely	Purchasing Manager	€11.000
John Russell	Sales Manager	€14.000
Shanta Vollman	Stock Manager	€6.500
Matthew Weiss	Stock Manager	€8.000
Eleni Zlotkey	Sales Manager	€10.500

Sincerely,

Larry Ellison

Output document generated from example1a.docx

```
CREATE OR REPLACE PROCEDURE EXAMPLE1A
(      p_employee_id number)
is
--
cursor c1 (p_employee_id number)
is
select p_employee_id as manager_id
,      initcap(first_name || ' ' || last_name) as manager
,      to_char(sysdate,'dd month yyyy') today
from   employees
where  employee_id = p_employee_id;
--
cursor c2 (p_manager_id number)
is
select initcap(emp.first_name || ' ' || emp.last_name) as employee
,      trim(to_char(emp.salary, 'L999G999G999'))         as salary
,      job.job_title
from   employees emp
,      jobs job
where  emp.manager_id = p_manager_id
and    emp.job_id = job.job_id
order by emp.last_name;
--
BEGIN
--
if sqlword.init_report('EXAMPLE1A') then
--
sqlword.put_content('EXAMPLE1A',1);
--
for r1 in c1(p_employee_id) loop
sqlword.put_content('EXAMPLE1A',2);
sqlword.put_content('EXAMPLE1A',3);
sqlword.put_content('EXAMPLE1A',4);
sqlword.put_data(r1.manager);
sqlword.put_content('EXAMPLE1A',5);
sqlword.put_content('EXAMPLE1A',6);
--
for r2 in c2 (r1.manager_id) loop
sqlword.put_content('EXAMPLE1A',7);
sqlword.put_data(r2.employee);
sqlword.put_content('EXAMPLE1A',8);
sqlword.put_data(r2.job_title);
sqlword.put_content('EXAMPLE1A',9);
sqlword.put_data(r2.salary);
sqlword.put_content('EXAMPLE1A',10);
end loop;
--
sqlword.put_content('EXAMPLE1A',13);
end loop;
--
sqlword.put_content('EXAMPLE1A',16);
sqlword.put_content('EXAMPLE1A',17);
sqlword.put_content('EXAMPLE1A',18);
sqlword.put_content('EXAMPLE1A',19);
sqlword.put_content('EXAMPLE1A',20);
--
sqlword.merge_xml('word/header1.xml');
--
sqlword.put_content('EXAMPLE1A',21);
sqlword.put_data(to_char(sysdate,'fm dd month yyyy'));
sqlword.put_content('EXAMPLE1A',22);
sqlword.put_content('EXAMPLE1A',23);
--
end if;
--
sqlword.end_report;
--
```

Generated stored procedure for example1a.docx

Example1b.docx

Redwood, <%=to_char(sysdate,'fm dd month yyyy')%>

<%for r1 in c1(p_employee_id) loop%>
Dear <%=r1.manager%>,

We inform you about the current salary of your employees:

Employee	Job	Salary
<%for r2 in c2 (r1.manager_id) loop%><%=r2.employee%>	<%=r2.job_title%>	<%=r2.salary%>
<%end loop;%>		

Sincerely,

Larry Ellison

<%end loop;%>

```
<%@ plsql parameter="P_EMPLOYEE_ID" "type="number"%>
<%@ include file="example1b.plsql"%>
```

Example1b.docx is about the same as example1a.docx with the difference that the PL/SQL declarations are included by an external file. In this way you can edit large PL/SQL declarations much easier.

- include-tag:

  <%@ include file="example1b.plsql"%>

Example1c.docx

Redwood, <%=to_char(sysdate,'fm dd month yyyy')%>

<%for r1 in hr_cursors.c_mgr(p_employee_id) loop%>
Dear <%=r1.manager%>,

We inform you about the current salary of your employees:

Employee	Job	Salary
<%for r2 in hr_cursors.c_emp (r1.manager_id) loop%><%=r2.employee%>	<%=r2.job_title%>	<%=r2.salary%>
<%end loop;%>		

Sincerely,

Larry Ellison

<%end loop;%>
<%@ plsql parameter="P_EMPLOYEE_ID" "type="number"%>

Example1c.docx is about the same as example1a.docx with the difference that the PL/SQL declarations are included by a reference to the cursor declarations in package specification HR_CURSORS. In this way you can keep controll of all your SQLWord SQL statements and modify the cursors quickly in case of database changes.

- Statement-tags:

```
<%for r1 in hr_cursors.c_mgr(p_employee_id) loop%>

<%for r2 in hr_cursors.c_emp (r1.manager_id) loop%>

<%end loop;%>
```

```
CREATE OR REPLACE PACKAGE HR_CURSORS
IS
--
cursor c_mgr (p_employee_id number)
is
select p_employee_id as manager_id
,      initcap(first_name || ' ' || last_name) as manager
,      to_char(sysdate,'dd month yyyy') today
from   employees
where  employee_id = p_employee_id;
--
cursor c_emp (p_manager_id number)
is
select initcap(emp.first_name || ' ' || emp.last_name) as employee
,      trim(to_char(emp.salary, 'L999G999G999'))         as salary
,      job.job_title
from   employees emp
,      jobs job
where  emp.manager_id = p_manager_id
and    emp.job_id = job.job_id
order by emp.last_name;
--
```

Redwood, <%=to_char(sysdate,'fm dd month yyyy')%>

<%for r1 in c1(p_employee_id) loop%>
Dear <%=r1.manager%>,

We inform you about the current salary of your employees:

Employee	Job	Salary
<%for r2 in c2 (r1.manager_id) loop%><%=r2.employee%>	<%=r2.job_title%>	<%=r2.salary%>
<%end loop;%>		

Sincerely,

Larry Ellison

```
<%end loop;%>
<%@ plsql parameter="P_EMPLOYEE_ID" type="number"%>
<%!
--
cursor c1 (p_employee_id number)
is
select p_employee_id as manager_id
,      '#SQLWORD_FORMAT</w:t><w:rPr><w:b/></w:rPr><w:t>'|| chr(38) || '#160;' ||
       first_name || chr(38) || '#160;' || '</w:t></w:r><w:r><w:t>' ||
       last_name as manager
,      to_char(sysdate,'dd month yyyy') today
from   employees
where  employee_id = p_employee_id;
--
cursor c2 (p_manager_id number)
is
select initcap(emp.first_name || ' ' || emp.last_name) as employee
,      case
         when emp.salary > 3000 then
         '#SQLWORD_FORMAT</w:t><w:rPr><w:i/><w:color
          w:val="FF0000"/></w:rPr><w:t>'  ||
          trim(to_char(emp.salary, 'L999G999G999')) ||
          '</w:t></w:r><w:r><w:t>'
         else
           trim(to_char(emp.salary, 'L999G999G999'))
       end as salary
,      job.job_title
from   employees emp
,      jobs job
where  emp.manager_id = p_manager_id
and    emp.job_id = job.job_id
order by emp.last_name;
--
```

Example1d.docx is about the same as example1a.docx with the difference that extra XML format code is added by using the **#SQLWORD_FORMAT** hint. In this example the first name is formatted to **bold** and the salary color is set to red when greater than 3000.

For more information about XML formatting check this nice website: http://www.lenzconsulting.com/wordml

Dear **Adam** Fripp,

We inform you about the current salary of your employees:

Employee	Job	Salary
Mozhe Atkinson	Stock Clerk	€2.800
Laura Bissot	Stock Clerk	*€3.300*
Alexis Bull	Shipping Clerk	*€4.100*
Anthony Cabrio	Shipping Clerk	€3.000
Julia Dellinger	Shipping Clerk	*€3.400*
James Marlow	Stock Clerk	€2.500
Tj Olson	Stock Clerk	€2.100
Nandita Sarchand	Shipping Clerk	*€4.200*

Sincerely,

Larry Ellison

Output document generated from example1d.docx

Example2.docx

<%open c_emp; fetch c_emp into r_emp;%>

Dear Mr./Ms. <%=r_emp.last_name%>,

Human Resources Inc. is pleased to offer you the position of <%=r_emp.emp_job_title%>. Your skills and experience will be an ideal fit for our <%=r_emp.department%> department.

As we discussed, your starting date will be <%=r_emp.hire_date%>.
The salary scale for this job ranges from <%=r_emp.min_salary%> to <%=r_emp.max_salary%> per month.

The salary is <%=r_emp.year_salary%> per year and is paid on a monthly basis. Direct deposit is available.

Full family medical coverage will be provided through our company's employee benefit plan. Dental and optical insurance are also available.

Human Resource Inc. offers a flexible paid-time off plan which includes vacation, personal, and sick leave. Time off accrues at the rate of one day per month for your first year, then increases based on your tenure with the company.

We look forward to welcoming you to the Human Resource Inc. team.

Please let me know if you have any questions or I can provide any additional information.

Sincerely,
<%=r_emp.manager%>
<%=r_emp.mgr_job_title%>, department <%=r_emp.department%>
Human Resource Inc.
<%close c_emp;%>

```
<%@ plsql parameter="P_EMPLOYEE_ID" "type="number"%>
<%!
--
cursor c_emp
is
select emp.last_name
,      trim(to_char(emp.hire_date,'dd') || ' ' ||
       trim(to_char(emp.hire_date, 'month')) || ' ' ||
       to_char(emp.hire_date,'yyyy')) as hire_date
,      trim(to_char(emp.salary * 12 , 'L999G999G999'))
       as year_salary
,      job1.job_title as emp_job_title
,      trim(to_char(job1.min_salary, 'L999G999G999')) as min_salary
,      trim(to_char(job1.max_salary, 'L999G999G999')) as max_salary
,      trim(mgr.first_name || ' ' || mgr.last_name) as manager
,      job2.job_title as mgr_job_title
,      dept.department_name as department
,      loc.street_address || ' ' || loc.city as department_address
,      trim(to_char(sysdate,'dd') || ' ' ||
       trim(to_char(sysdate, 'month')) || ' ' ||
       to_char(sysdate,'yyyy')) as today
from   employees emp
,      employees mgr
,      departments dept
,      jobs job1
,      jobs job2
,      locations loc
where  emp.employee_id = P_EMPLOYEE_ID
and    emp.job_id = job1.job_id
and    mgr.job_id = job2.job_id
and    emp.department_id = dept.department_id (+)
and    emp.manager_id = mgr.employee_id (+)
and    dept.location_id = loc.location_id (+);
--
r_emp c_emp%rowtype;
--
%>
```

Example2.docx shows how to generate a job offer document for an employee.

- Declaration-tag:

```
<%!
--
cursor c_emp
is
select emp.last_name
,      trim(to_char(emp.hire_date,'dd') || ' ' ||
       trim(to_char(emp.hire_date, 'month')) || ' ' ||
       to_char(emp.hire_date,'yyyy')) as hire_date
,      trim(to_char(emp.salary * 12 , 'L999G999G999'))
       as year_salary
,      job1.job_title as emp_job_title
,      trim(to_char(job1.min_salary, 'L999G999G999')) as min_salary
,      trim(to_char(job1.max_salary, 'L999G999G999')) as max_salary
,      trim(mgr.first_name || ' ' || mgr.last_name) as manager
,      job2.job_title as mgr_job_title
,      dept.department_name as department
,      loc.street_address || ' ' || loc.city as department_address
,      trim(to_char(sysdate,'dd') || ' ' ||
       trim(to_char(sysdate, 'month')) || ' ' ||
       to_char(sysdate,'yyyy')) as today
from   employees emp
,      employees mgr
,      departments dept
,      jobs job1
,      jobs job2
,      locations loc
where  emp.employee_id = P_EMPLOYEE_ID
and    emp.job_id = job1.job_id
and    mgr.job_id = job2.job_id
and    emp.department_id = dept.department_id (+)
and    emp.manager_id = mgr.employee_id (+)
and    dept.location_id = loc.location_id (+);
--
r_emp c_emp%rowtype;
--
%>
```

- Parameter-tag:

```
<%@ plsql parameter="P_EMPLOYEE_ID" "type="number"%>
```

- Statement-tags:

```
<%open c_emp; fetch c_emp into r_emp;%>

<%close c_emp;%>
```

- Assignment-tags:

```
<%=r_emp.last_name%>

<%=r_emp.emp_job_title%>

<%=r_emp.department%>

<%=r_emp.hire_date%>

<%=r_emp.min_salary%>

<%=r_emp.max_salary%>

<%=r_emp.year_salary%>

<%=r_emp.manager%>

<%=r_emp.mgr_job_title%>

<%=r_emp.department%>
```

Dear Mr./Ms. Fripp,

Human Resources Inc. is pleased to offer you the position of Stock Manager. Your skills and experience will be an ideal fit for our Shipping department.

As we discussed, your starting date will be 10 april 2005.

The salary scale for this job ranges from €5.500 to €8.500 per month.
The salary is €98.400 per year and is paid on a monthly basis. Direct deposit is available.

Full family medical coverage will be provided through our company's employee benefit plan. Dental and optical insurance are also available.

Human Resource Inc. offers a flexible paid-time off plan which includes vacation, personal, and sick leave. Time off accrues at the rate of one day per month for your first year, then increases based on your tenure with the company.

We look forward to welcoming you to the Human Resource Inc. team.

Please let me know if you have any questions or I can provide any additional information.

Sincerely,

Steven King
President, department Shipping

Human Resource Inc.

Output document generated from example2.docx

Example3.docx

This is an example of an advanced Tables Report demonstrating some interesting constructions.

DEPARTMENTS

Departments table that shows details of departments where employees work.
Contains 27 rows; references with locations, employees, and job_history tables.

Columns

Name	Datatype	Nullable	Comments
DEPARTMENT_ID	number (4)	not null	Primary key column of departments table.
DEPARTMENT_NAME	varchar2 (30)	not null	A not null column that shows name of a department. Administration, Marketing, Purchasing, Human Resources, Shipping, IT, Executive, Public Relations, Sales, Finance, and Accounting.
MANAGER_ID	number (6)		Manager_id of a department. Foreign key to employee_id column of employees table. The manager_id column of the employee table references this column.
LOCATION_ID	number (4)		Location id where a department is located. Foreign key to location_id column of locations table.

Primary key

Name	Column	Status
DEPARTMENTS	department_id	enabled

Foreign keys

Name	Column	Related to table	Column	Status
DEPT_MGR_FK	manager_id	EMPLOYEES	employee_id	enabled
DEPT_LOC_FK	location_id	LOCATIONS	location_id	enabled

Check constraints

Name	Condition
DEPT_NAME_NN	"DEPARTMENT_NAME" IS NOT NULL

Indexes

Name	Uniqueness	Column	Tablespace	Status
DEPT_ID_PK	unique	department_id	system	valid
DEPT_LOCATION_IX	nonunique	location_id	system	valid

Output document generated from example3.docx

Example4.docx

This is an example to show if your Oracle database characterset works fine with the unicode characterset from Microsoft Word. It works 100% fine when you use the Oracle database characterset AL32UTF8.

Declaration-tag:

```
<%!
--
cursor c1
is
select parameter
,      value
from   nls_database_parameters
where  parameter in ( 'NLS_LANGUAGE', 'NLS_TERRITORY', 'NLS_CHARACTERSET')
order  by parameter;
--
cursor c2
is
select unistr( '\0627\0644\0639\0631\0628\064A\0629' )      as arabic
,      unistr( '\4E2D\6587' )                              as chinese
,      unistr( 'English' )                                 as english
,      unistr( 'Fran\00E7ais' )                            as french
,      unistr( 'Deutsch' )                                 as german
,      unistr( '\0395\03BB\03BB\03B7\03BD\03B9\03BA\03AC' ) as greek
,      unistr( '\05E2\05D1\05E8\05D9\05EA' )               as hebrew
,      unistr( '\65E5\672C\8A9E' )                         as japanese
,      unistr( '\D55C\AD6D\C5B4' )                         as korean
,      unistr( 'Portugu\00EAs' )                           as portuguese
,      unistr( '\0420\0443\0441\0441\043A\0438\0439' )      as Russian
,      unistr( 'Espa\00F1ol' )                             as Spanish
,      unistr( '\0E44\0E17\0E22' )                         as Thai
from   dual;
--
r2 c2%rowtype;
--
%>
```

Testing your characterset for unicode

Parameter	Value
NLS_CHARACTERSET	AL32UTF8
NLS_LANGUAGE	AMERICAN
NLS_TERRITORY	AMERICA

Language	Text
Arabic	العربية
Chinese	中文
English	English
French	Français
German	Deutsch
Greek	Ελληνικά
Hebrew	עברית
Japanese	日本語
Korean	한국어
Portuguese	Português

Output document generated from example4.docx

Example5.docx

This is an example showing that you can include a picture from file system or url.

Declaration-tag:

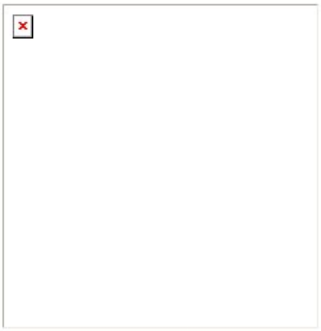
```
<%!
--
cursor c1
is
select initcap(first_name || ' ' || last_name) as manager
,      'http://www.sequel.nl/download/larry_ellison.jpg'   as image1
,      'file:///C:/SQLWord11/Examples/Docx/SQLWordBox.png' as image2
from   employees
where  employee_id = 100;
--
%>
```

<%forr1 in c1 loop%>
Dear <%=r1.manager%>,

This demo shows that you can include a picture from file system or url.
Examine the SQLWord manual where we explain how to do it!

Sincerely,

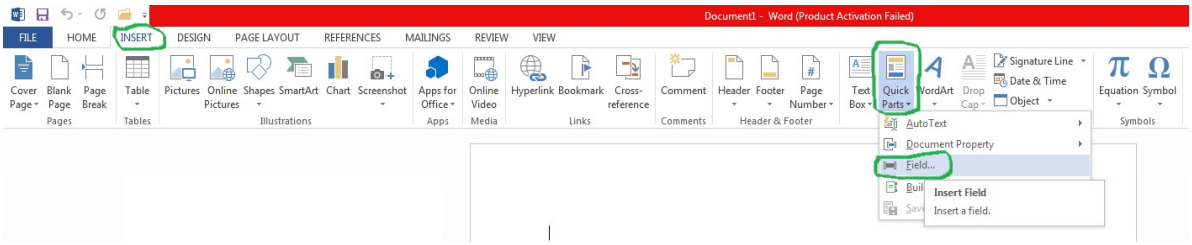
Larry Ellison



```
<%end loop;%>
<%!
--
cursor c1
is
select initcap(first_name || ' ' || last_name) as manager
,      'http://www.sequel.nl/download/larry_ellison.jpg'   as image1
,      'file:///C:/SQLWord11/Examples/Docx/SQLWordBox.png' as image2
from   employees
where  employee_id = 100;
--
%>
```

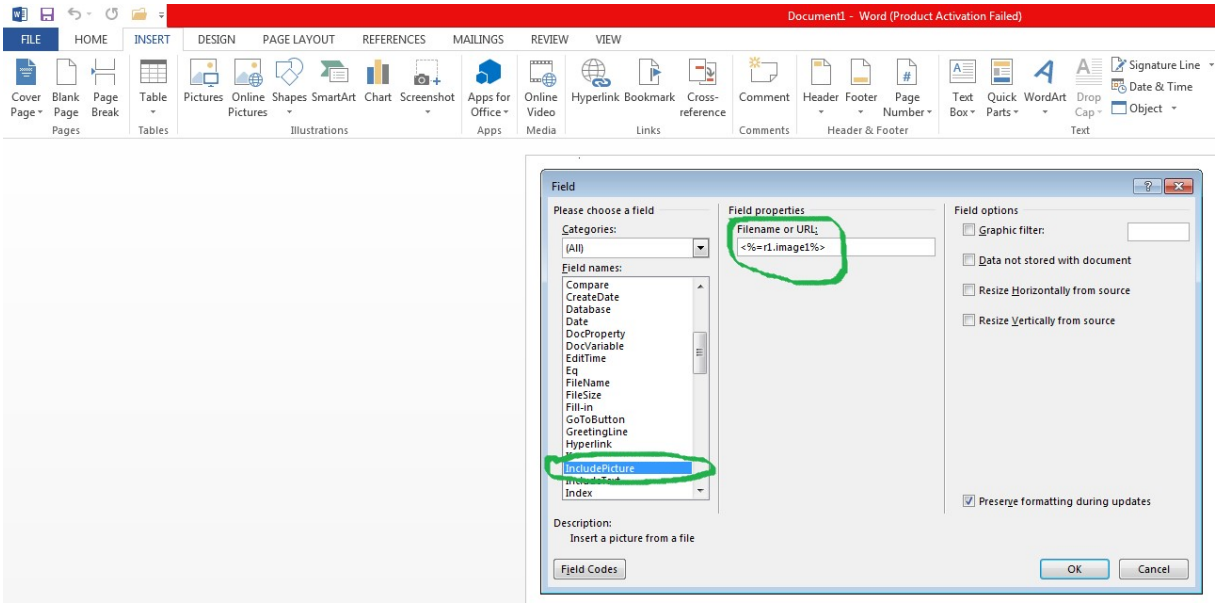
How to include the pictures dynamicly:

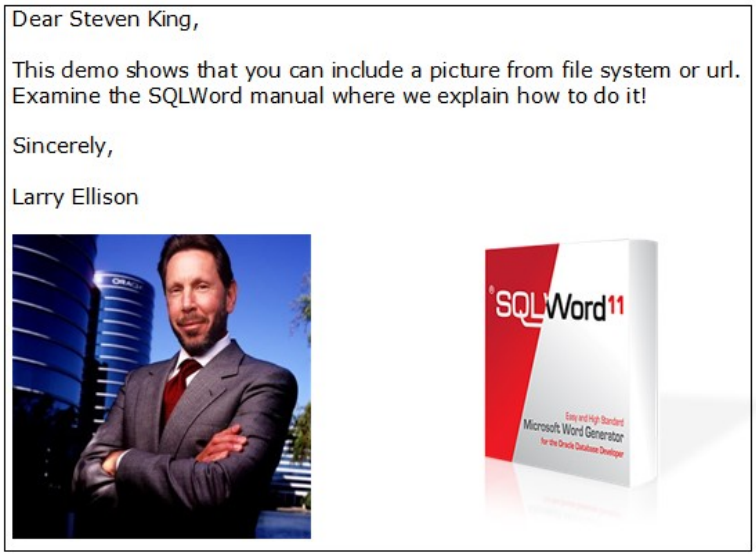
Step 1: insert a field into the Word document



Step 2: choose IncludePicture and fill in a scriptlet in field Filename or URL:

<%=r1.image1%> and <%=r1.image2%>





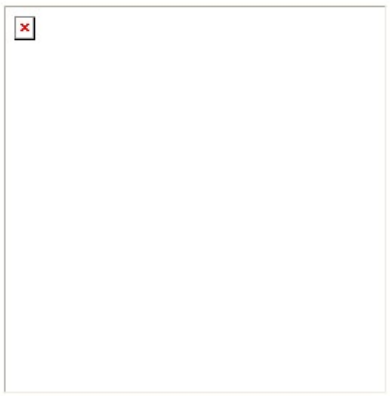
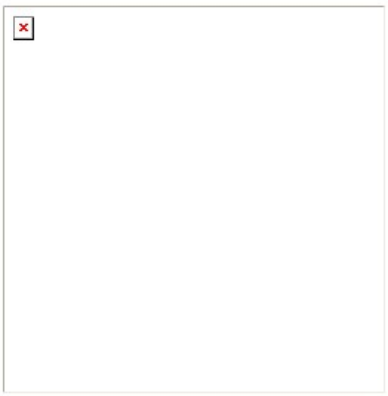
Output document generated from example5.docx

Example6.docx

This is an example showing that you can include a picture from a picture stored in the Oracle database as a BLOB. You do need to import file C:\SQLWord11\SQL\image_demo.dmp to get table IMAGES_DEMO with the two pictures !

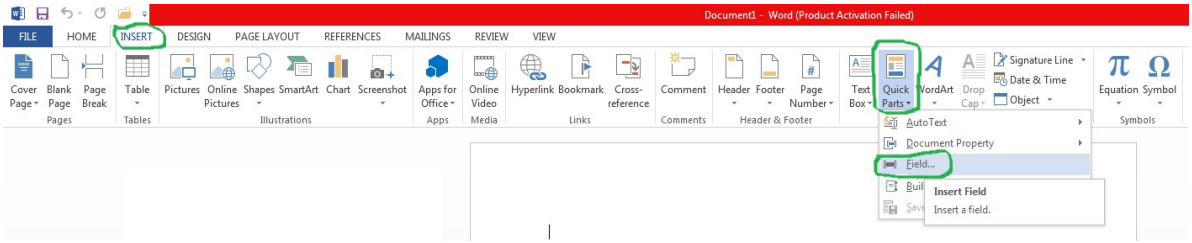
Declaration-tag:

```
<%!
function get_image (p_id in number) return blob
is
  l_blob blob;
begin
  --
  select data
  into   l_blob
  from   image_demo
  where  id = p_id;
  --
  return(l_blob);
  --
end;
%>
```

```
<%for r1 in c1 loop%>
Dear <%=r1.manager%>,

We send you the latest pictures from our Ocean race.




These pictures are stored in table IMAGES_DEMO.
You do need to import file C:\SQLWord11\SQL\image_demo.dmp.

Examine the SQLWord manual where we explain how to do it!

Sincerely,

Larry Ellison
<%end loop;%>
<%!
--
cursor c1
is
select initcap(first_name || ' ' || last_name) as manager
from   employees
where  employee_id = 100;
--
function get_image (p_id in number) return blob
is
  l_blob blob;
begin
  --
  select data
  into   l_blob
  from   image_demo
  where  id = p_id;
  --
  return(l_blob);
  --
end;
--
%>
```

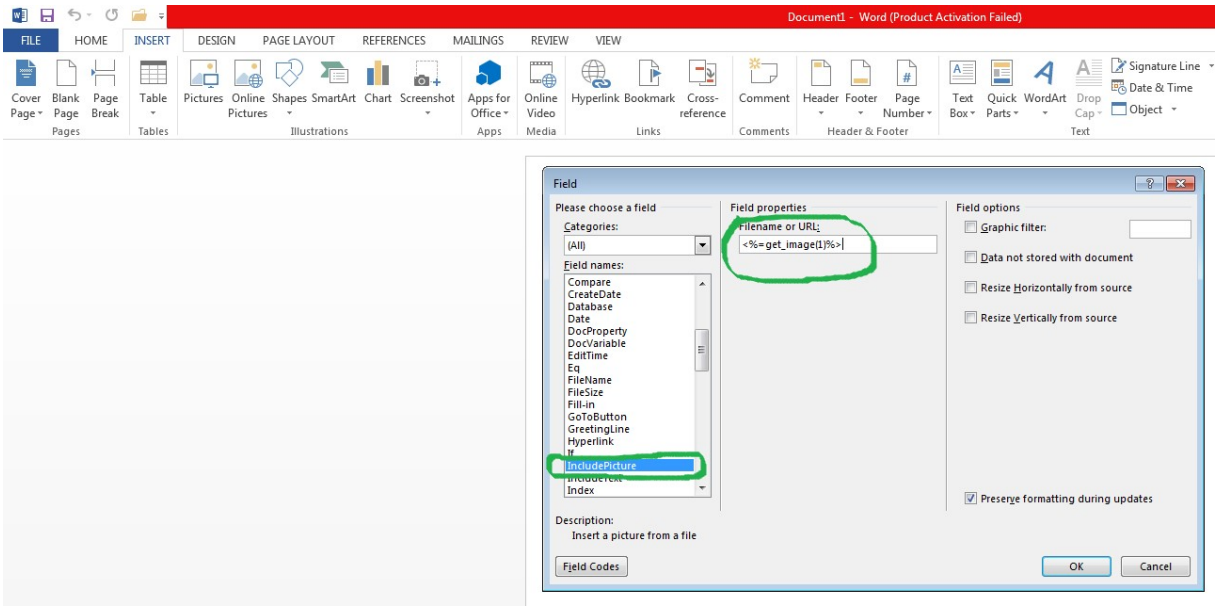
How to include the pictures dynamicly:

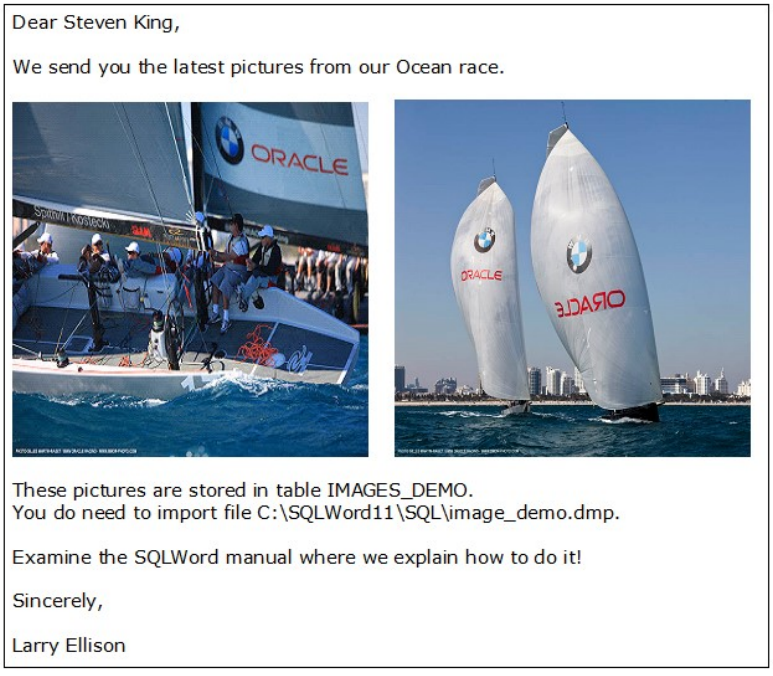
Step 1: insert a field into the Word document



Step 2: choose IncludePicture and fill in a scriptlet in field Filename or URL:

<%=get_image(1)%> and %=get_image(2)%>





Output document generated from example6.docx

HR_*.docx

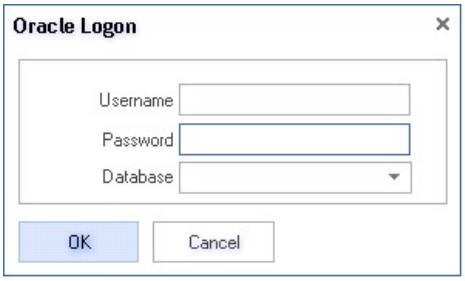
The HR_*.docx templates belong to the Apex demo application.

Steps to create a source document

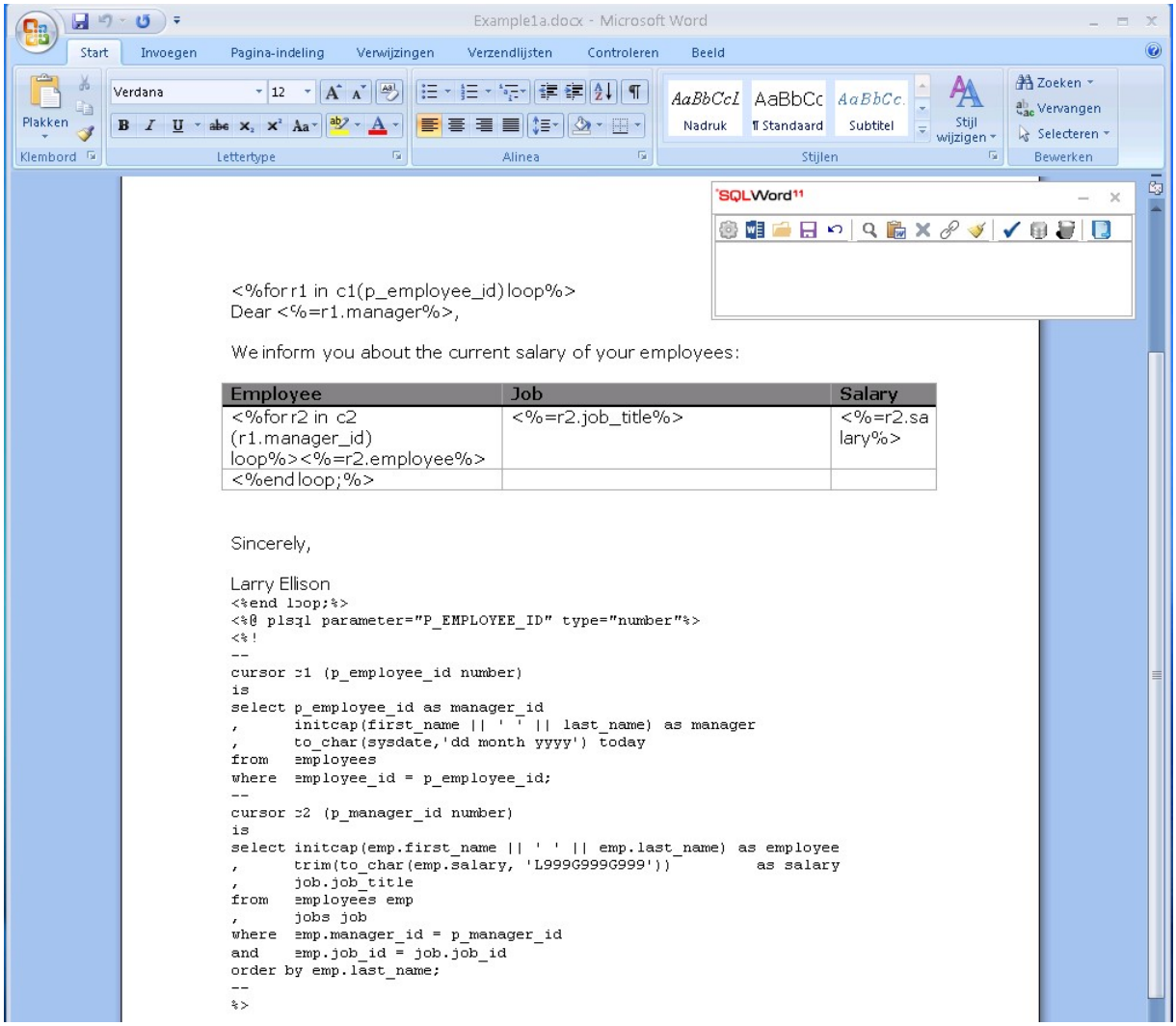
Step 1: Start SQLWord Developer

On starting SQLWord Developer the Oracle logon screen shows up.

Now connect to the Oracle schema where SQLWord tables and packages are installed by your DBA.



After connecting to the Oracle database the SQLWord Developer toolbar appears and Microsoft Word is started from the SQLWord Developer application.



The SQLWord Developer toolbar can be moved to another position on your desktop. The best place is to position it in the upper right corner of your screen.

The next time when SQLWord Developer is started the SQLWord Developer toolbar is positioned on the last positon where you left it.

Step 2: Create a new source document

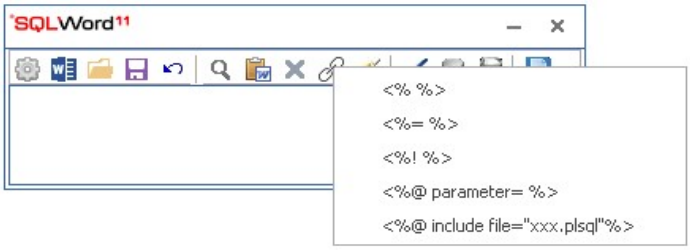
Press the  button in SQLWord Developer toolbar. A new document is opened in Microsoft Word.

Step 3: Type in your “static” content

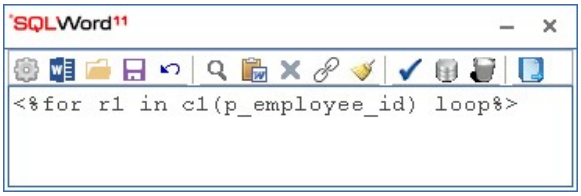
For this of course you can use all Microsoft Word features. Be sure that the layout of all “static” content is done before you go to the next step.

Step 4: Place <% tag %> scriptlets

Prepare your <% tag %> scriptlet in the work area. You can use the button to select the tag that you need and paste it from the submenu into the work area.



Type in the work-area your SQL-statements:



Now paste the prepared scriptlet from the work area to the Word source document by pressing the paste button. Do this for all the scriptlets that you need.

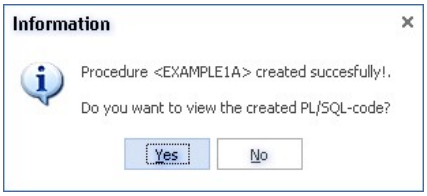
Step 5: Save the source document

Press the button to save the Word source document.

Step 6: Compile the source document to a stored procedure

Compile the active source document to a stored procedure by pressing the button.

After finishing this screen shows up if you did not make mistakes



If the generated stored procedure contains errors you probably want to see the invalid PL/SQL code. In that case choose "Yes" to examine the invalid read-only PL/SQL source code.

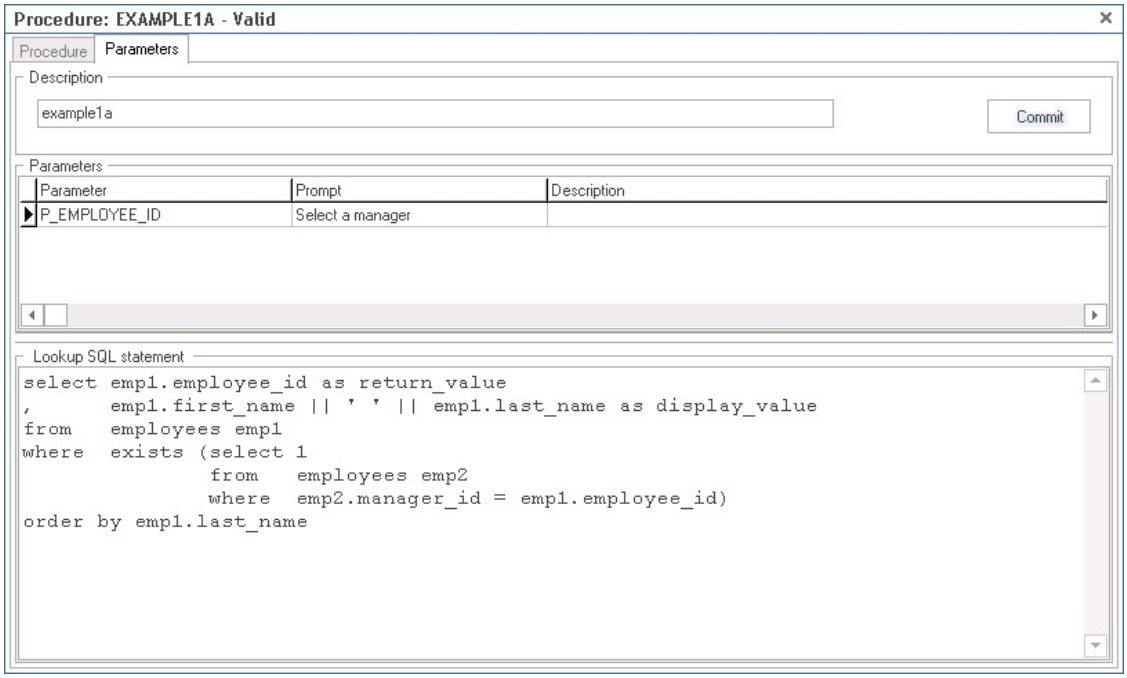
Always fix errors in your source document and recompile the source document until it is valid.

Step 7: Specify input-parameters (if necessary)

If you want to specify input parameters press the button. Change to the second tab "Parameters" to fill in a prompt and description or specify a lookup SQL-statement.

Note that lookup SQL-statements always must have two columns:

1. The first column must give an unique identifier that will be assigned to the input parameter after choosing.
2. The second column must give the description (varchar2) that is displayed in the lookup list.

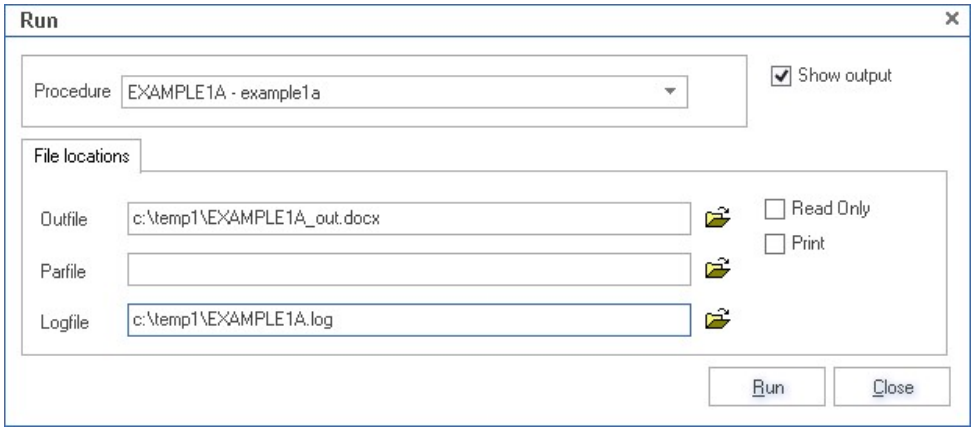


NB: Lookup's are only usefull when running SQLWord interactively "client-server".

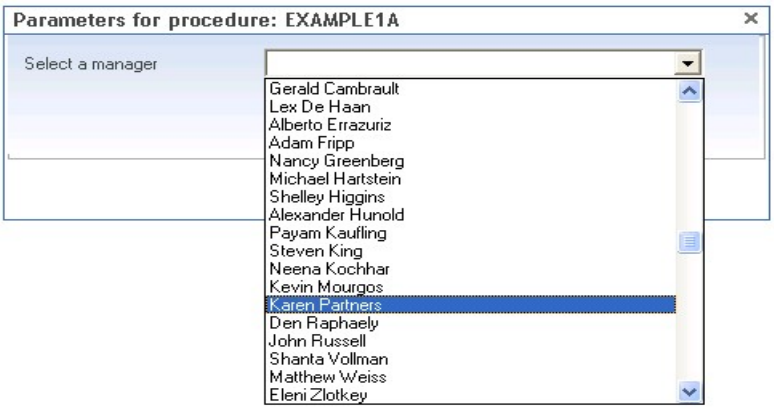
Step 8: Run it

You can run the report that you just created by pressing the Run button from the SQLWord Developer toolbar.

Run SQLWord. The screen below shows up where you can select a stored procedure and specify values for file locations.



When pressing the Run button a parameter screen appears where you can specify input parameters:



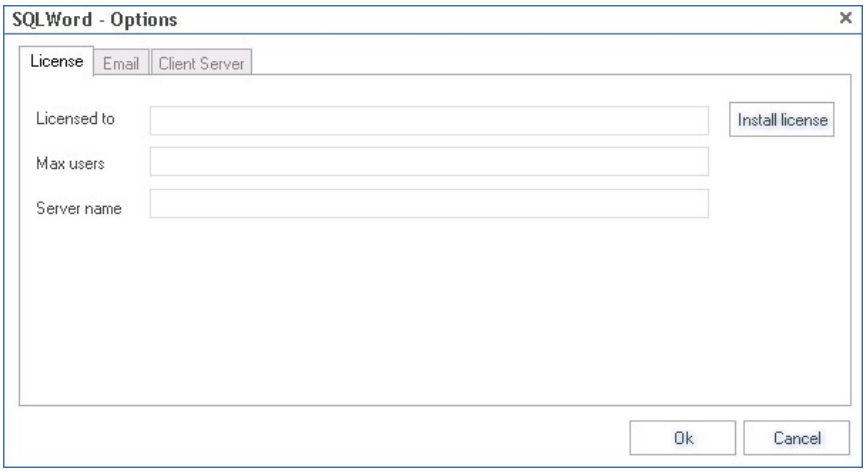
Options

Press the menu button from the SQLWord toolbar and choose “Options” from the submenu.

The options window shows up. This window contains 3 tabs, which are described below.

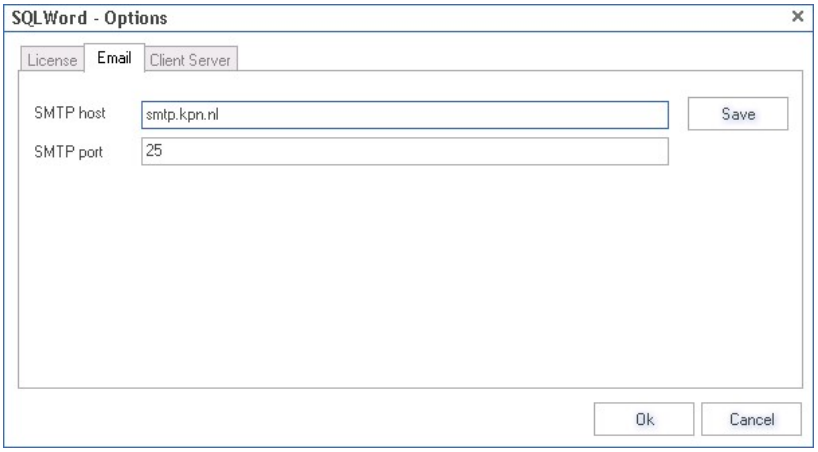
License tab

On the license tab you can see the license information or install your SQLWord license.



Email tab

On the email tab you can specify email settings in case you want to send an output document by email from your Oracle database. SQLWord uses the Oracle UTL_SMTP package.



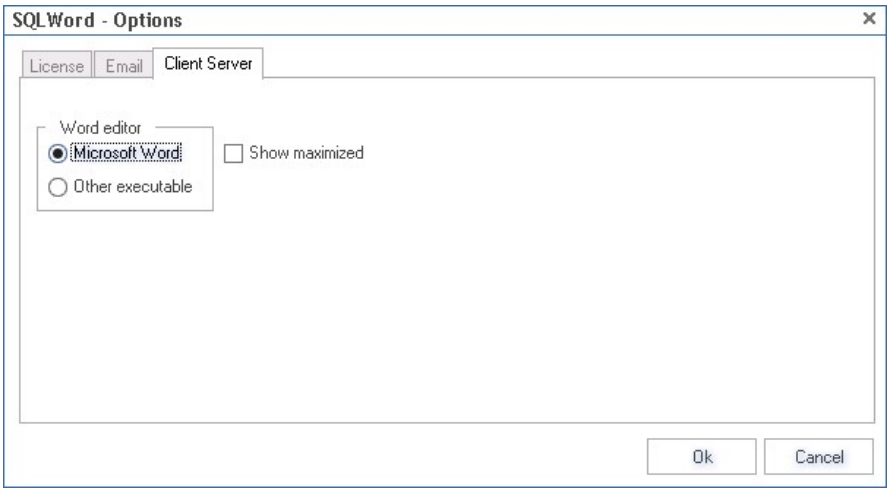
SMTP host: The name of your SMTP-server or IP-address.

SMTP port: The port number, usually this is port 25.

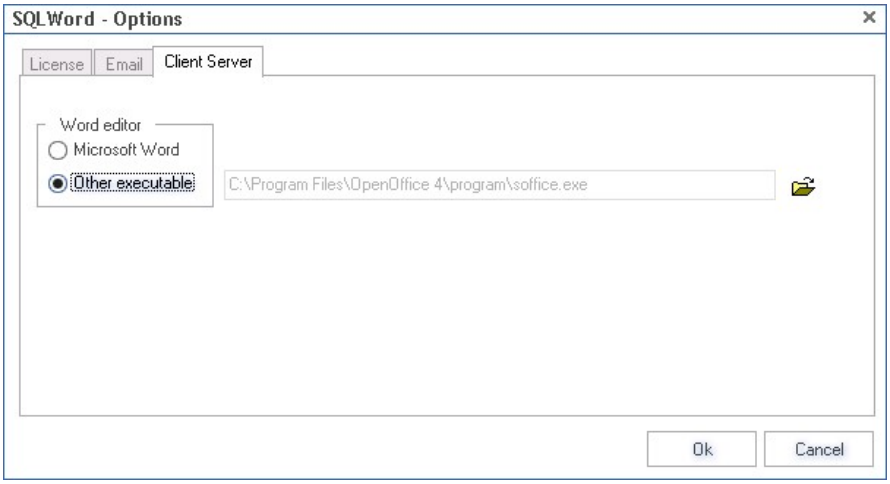
For more information how to send email, examine section “Frequently asked questions”.

Client Server tab

On the Client Server tab you can specify the client settings specific for your PC.



Or



Microsoft Word: If you choose this option SQLWord will use Microsoft Word as the default editor for all your Word documents.

Other executable: If you want to use a different program than Microsoft Word (for example Open Office) then you must specify which executable SQLWord should use to show the output document. SQLWord Developer doesn't use this editor for editing your docx-templates.

Show maximized: Indicates if Microsoft Word should maximize on opening.

SQLWord Run

Introduction

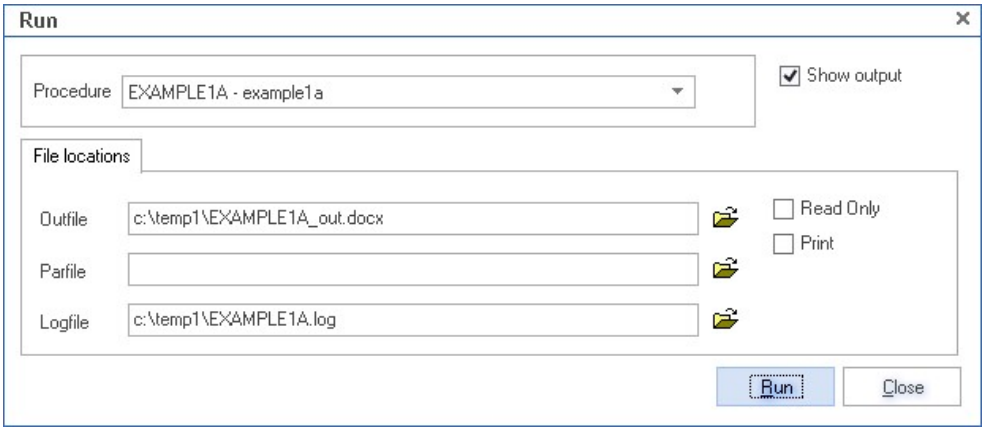
SQLWord Run is a 32-bits Windows application for running SQLWord reports interactively or in batch mode from the command line.

Menu options:

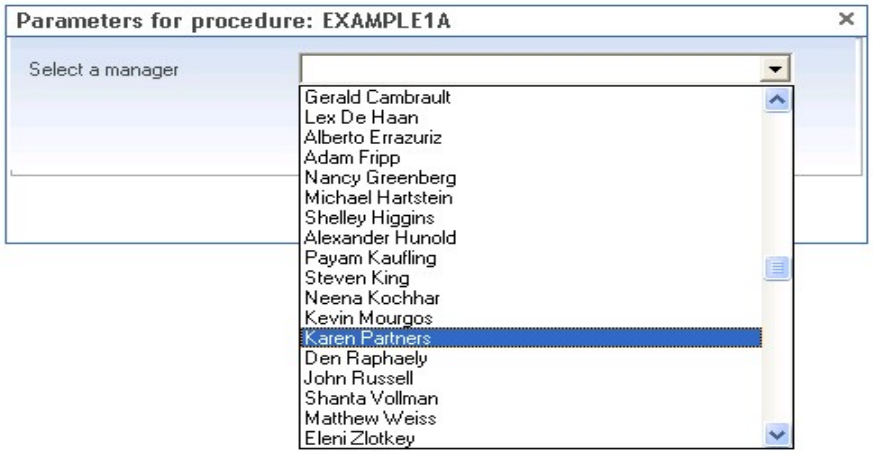
File: Shows a submenu where you can:

- Connect to your Oracle database.
- Display the options-window.

Run: The screen below shows up where you can select a stored procedure and specify values for file locations.

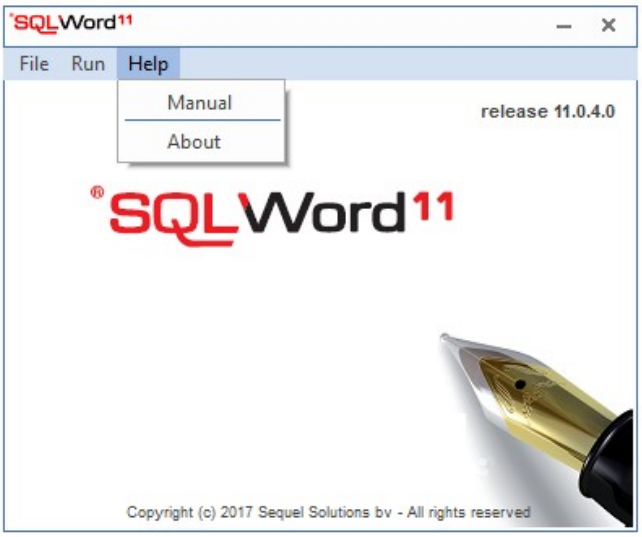


When pressing the Run button a parameter screen appears where you can specify input parameters:



Help: Shows a submenu where you can:

- Display this Users Guide & Reference.
- Show the about box.



Command line syntax

You can run the SQLWordRun executable from the command line with the following syntax:

SQLWordRun.exe keyword=<value> keyword=<value> keyword=<value> etc.

Keywords

Keyword	Description
userid=username/password@host	Oracle connect string.
procedure=stored-procedure-name	Name of the stored procedure to execute.
outfile=filename	Name and location of the output-file. In the run window the output file name can be specified and the directory can be choosen using the button on the right of the field.
[parfile=filename]	Name and location of the parameter file. In the run window the parameter file can be choosen using the button on the right of the field.
[logfile=filename]	Name and location of the log file. This file contains the logging information of the execution. In the run window the log file can be choosen, using then select button on the right of the field.
[editor=Y/N]	Open the output document with the default Word editor.
[readonly=Y/N]	Protect the output document by setting a secret password on the output document. This option is only available for Microsoft Word.
[print=Y/N]	Print the output document. This option is only available for Microsoft Word.
[copies=number]	The number of copies to print. This option is only available for Microsoft Word.
[printer=printername]	The name of the printer. This option is only available for Microsoft Word.
[showerror=Y/N]	To suppress all interactive messages (usefull for batch jobs).
[wordmacro=macroname]	Run a Word Macro on opening of the output document. This option is only available for Microsoft Word.
[role=rolename/password]	You can enable a database role when running SQLWord

The keywords between the straight [brackets] are optional keywords.

Example1:

```
SQLWordRun.exe" userid=sqlword_demo/sqlword_demo@my_db
procedure=example1a outfile="C:\Temp\example1a_out.docx"
parfile="C:\SQLWord11\Examples\Docx\example1.par"
```

Example2:

```
SQLWordRun.exe" userid=sqlword_demo/sqlword_demo@my_db
procedure=example1a outfile="C:\Temp\example1a_out.docx"
parfile="C:\SQLWord11\Examples\Docx\example1.par"
readonly=Y
```

Example3:

```
SQLWordRun.exe" userid=sqlword_demo/sqlword_demo@my_db
procedure=example1a outfile="C:\Temp\example1a_out.docx"
parfile="C:\SQLWord11\Examples\Docx\example1.par"
editor=N print=Y copies=1 printer=" HP OFFICEJET G SERIES"
```

You can find a sample batch script at:

C:\SQLWord11\Examples\Docx**run_cs_example1a.bat**

Parameter file

In a parameter file you can specify the values for the input parameters.

SQLWordRun reads the parameter file before execution the stored procedure and assigns the values to the input parameters to the called stored procedure.

The parameter file has the following syntax:

<PARAMETER>=<VALUE>

Example:

```
DEPTNO=10
HIRE_DATE=02-07-2009
ENAME='JONES'
```

You can find a sample parameter file at:

C:\SQLWord11\Examples\Templates**example1.par**

SQLExcel

How to generate Microsoft Excel XSLX

SQLWord11 supports generating Microsoft Excel files by package SQLEXCEL.

SQLEXCEL has several functions to create and write data to XLSX documents. For more information examine the package specifications.

You can find an example how to create a stored procedure for generating a Microsoft Excel file at:

C:\SQLWord11\Examples\SQL**excel_example1.sql**

You can find an example how to generate Microsoft Excel output at:

C:\SQLWord11\Examples\SQL\ **run_excel_example1.sql**

First you must use the Oracle “create directory” command (ask your DBA to do this).

```
-----------------------------------------------------------------
SQL> create or replace directory SQLWORD_OUTPUT_DIR as 'C:\Temp';
SQL> grant read, write on directory SQLWORD_OUTPUT_DIR to public;
-----------------------------------------------------------------

begin
  --
  excel_example1;
  --
  sqlexcel.save( p_directory => 'SQLWORD_OUTPUT_DIR'
               , p_filename  => 'excel_example1.xlsx');
  --
end;
```

Open the Excel-example1.xlsx file from the file location on your Oracle database server.

This spreadsheet has two tab sheets:

Region Countries Departments

FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW

A1 | Region

	A	B	C	D
1	**Region**	**Country**	**Departments**	
2	Americas	Argentina		
3		Brazil		
4		Canada	*Marketing*	
5		Mexico		
6		United States of America	*Accounting*	
7			*Administration*	
8			*Benefits*	
9			*Construction*	
10			*Contracting*	
11			*Control And Credit*	
12			*Corporate Tax*	
13			*Executive*	
14			*Finance*	
15			*Government Sales*	
16			*IT*	
17			*IT Helpdesk*	
18			*IT Support*	
19			*Manufacturing*	
20			*NOC*	
21			*Operations*	
22			*Payroll*	
23			*Purchasing*	
24			*Recruiting*	
25			*Retail Sales*	
26			*Shareholder Services*	
27			*Shipping*	
28			*Treasury*	
29	Asia	Australia		
30		China		
31		India		
32		Japan		
33		Malaysia		
34		Singapore		
35	Europe	Belgium		
36		Denmark		
37		France		
38		Germany	*Public Relations*	
39		Italy		
40		Netherlands		
41		Switzerland		
42		United Kingdom	*Human Resources*	
43			*Sales*	
44	Middle East and Africa	Egypt		
45		Israel		
46		Kuwait		
47		Nigeria		
48		Zambia		
49		Zimbabwe		
50				
51				

Regions Countries Departments | Departments Employees

Departments Employees

FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW

A1 Department

	A	B	C	D
1	Department	Employee	Job Title	
2	Accounting	William Gietz	Public Accountant	
3		Shelley Higgins	Accounting Manager	
4	Administration	Jennifer Whalen	Administration Assistant	
5	Benefits			
6	Construction			
7	Contracting			
8	Control And Credit			
9	Corporate Tax			
10	Executive	Lex De Haan	Administration Vice President	
11		Steven King	President	
12		Neena Kochhar	Administration Vice President	
13	Finance	John Chen	Accountant	
14		Daniel Faviet	Accountant	
15		Nancy Greenberg	Finance Manager	
16		Luis Popp	Accountant	
17		Ismael Sciarra	Accountant	
18		Jose Manuel Urman	Accountant	
19	Government Sales			
20	Human Resources	Susan Mavris	Human Resources Representative	
21	IT	David Austin	Programmer	
22		Bruce Ernst	Programmer	
23		Alexander Hunold	Programmer	
24		Diana Lorentz	Programmer	
25		Valli Pataballa	Programmer	
26	IT Helpdesk			
27	IT Support			
28	Manufacturing			
29	Marketing	Pat Fay	Marketing Representative	
30		Michael Hartstein	Marketing Manager	
31	NOC			
32	Operations			
33	Payroll			
34	Public Relations	Hermann Baer	Public Relations Representative	
35	Purchasing	Shelli Baida	Purchasing Clerk	
36		Karen Colmenares	Purchasing Clerk	
37		Guy Himuro	Purchasing Clerk	
38		Alexander Khoo	Purchasing Clerk	
39		Den Raphaely	Purchasing Manager	
40		Sigal Tobias	Purchasing Clerk	
41	Recruiting			
42	Retail Sales			
43	Sales	Ellen Abel	Sales Representative	
44		Sundar Ande	Sales Representative	
45		Amit Banda	Sales Representative	
46		Elizabeth Bates	Sales Representative	
47		David Bernstein	Sales Representative	
48		Harrison Bloom	Sales Representative	
49		Gerald Cambrault	Sales Manager	
50		Nanette Cambrault	Sales Representative	
51		Louise Doran	Sales Representative	

Regions Countries Departments | Departments Employees

Apex integration

An Apex demo application (for Apex 5.0.3 or higher) is available at: C:\SQLWord11\Examples\Apex5.0.

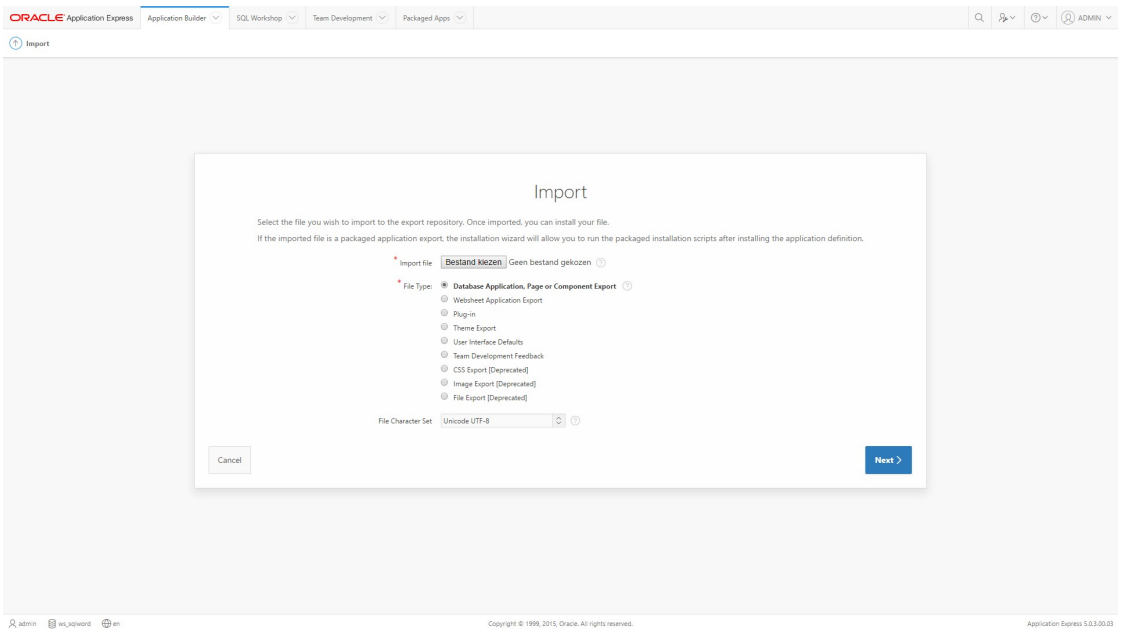
The Apex demo application is based on Oracle HR-tables and demonstrates how to integrate SQLWord within Apex.

Installation

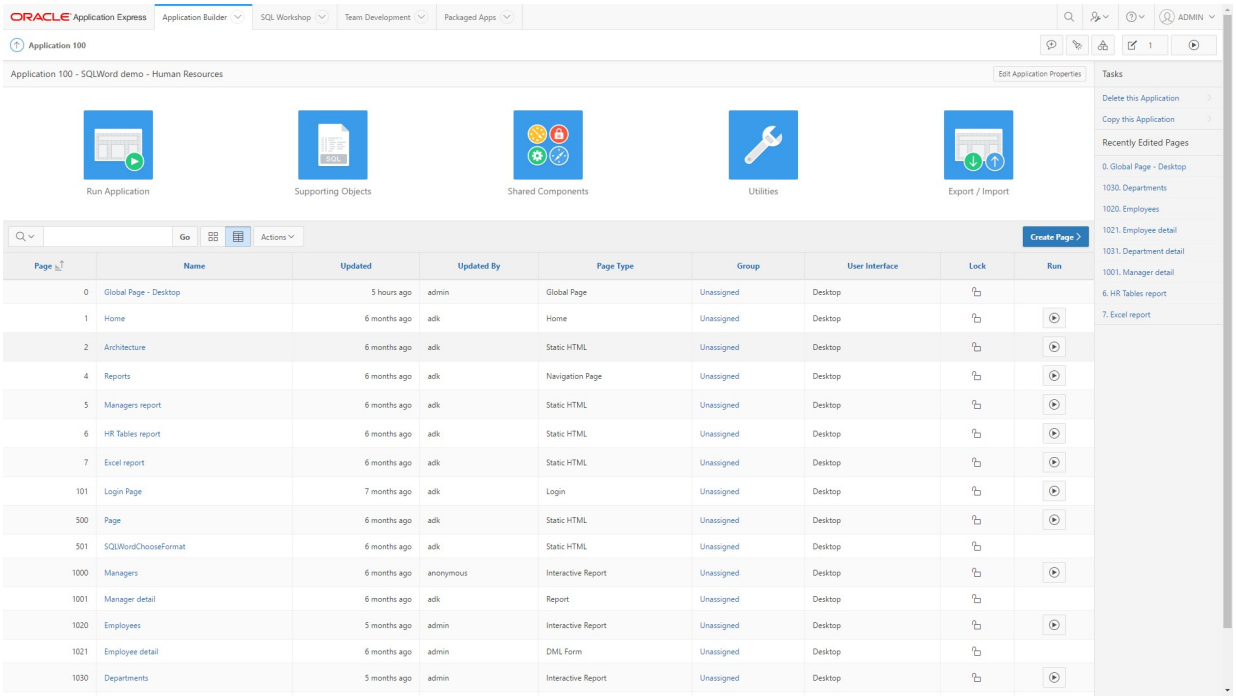
Step 1: Import the SQLWord11 HR DEMO application

Open the Apex Application Builder and import file:

C:\SQLWord11\Examples\Apex5.0**f100.sql**



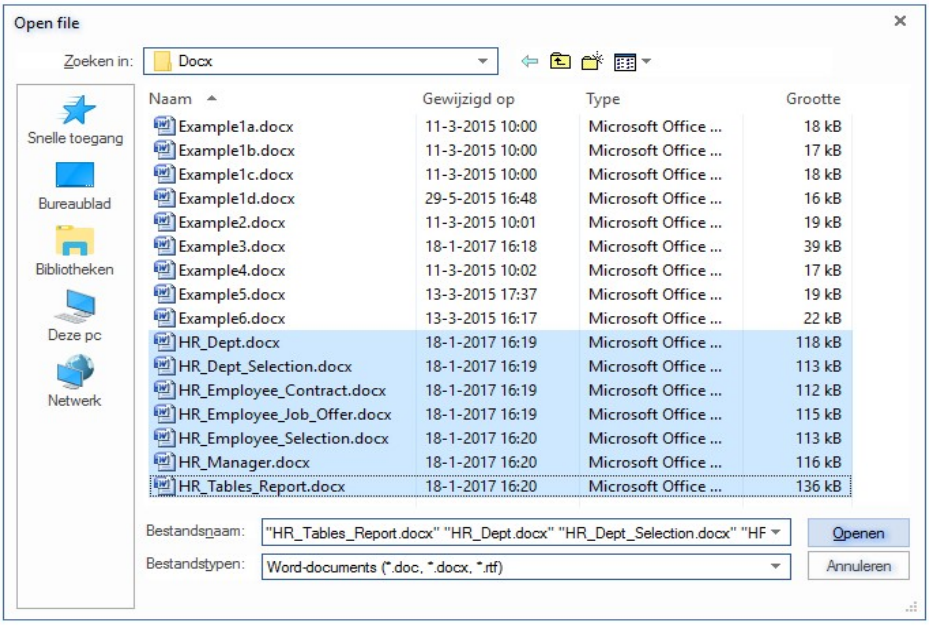
After finishing the import you should see these pages:



Step2: Compile all HR*.docx files

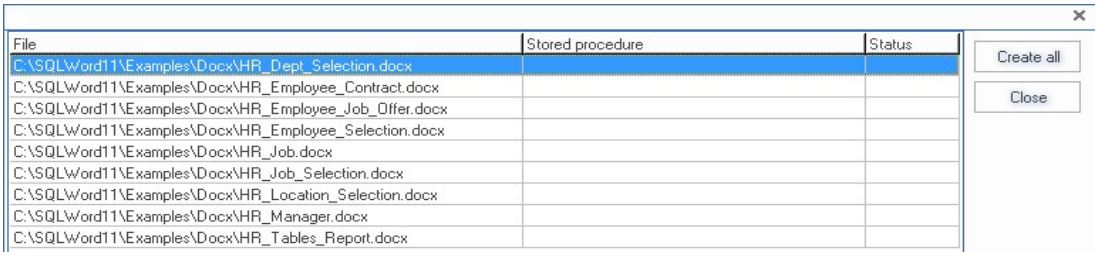
Start SQLWord Developer and connect to user SQLWORD_DEMO

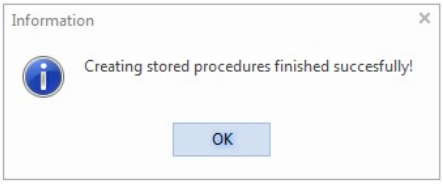
Click on button  in the SQLWord toolbar and select all HR_*.docx files at C:\SQLWord11\Examples\Docx

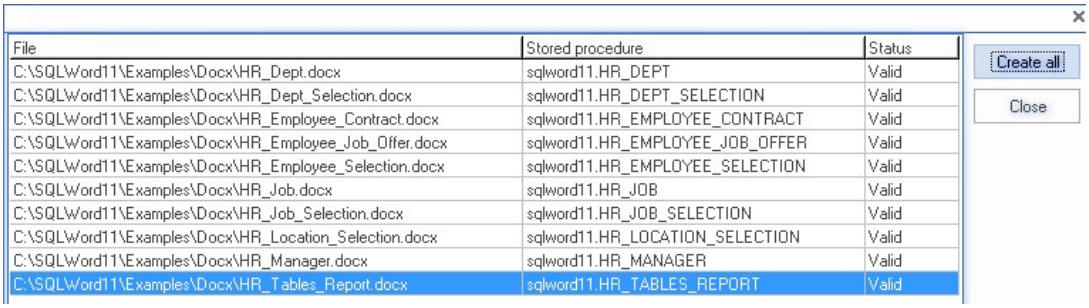


The selected files are displayed in this window.

Now press on button Create All





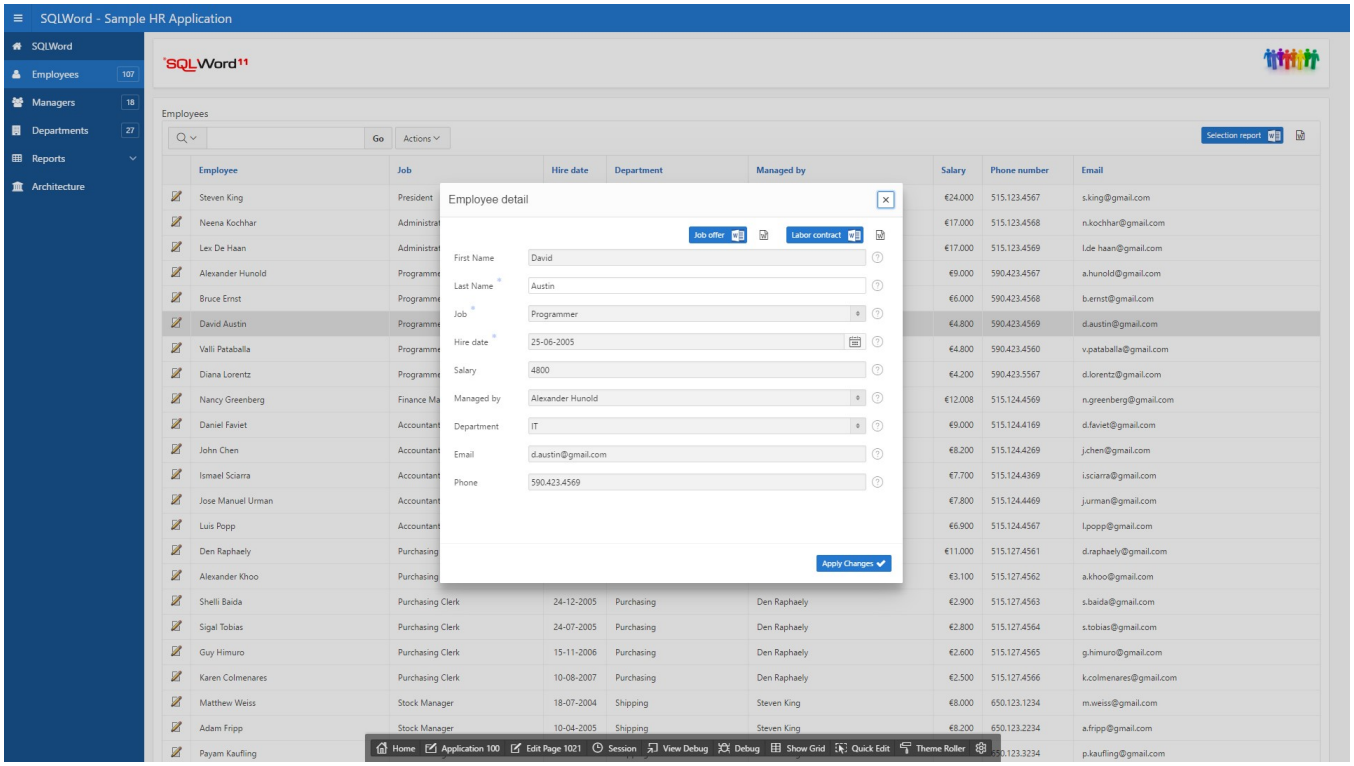


Implementation explained

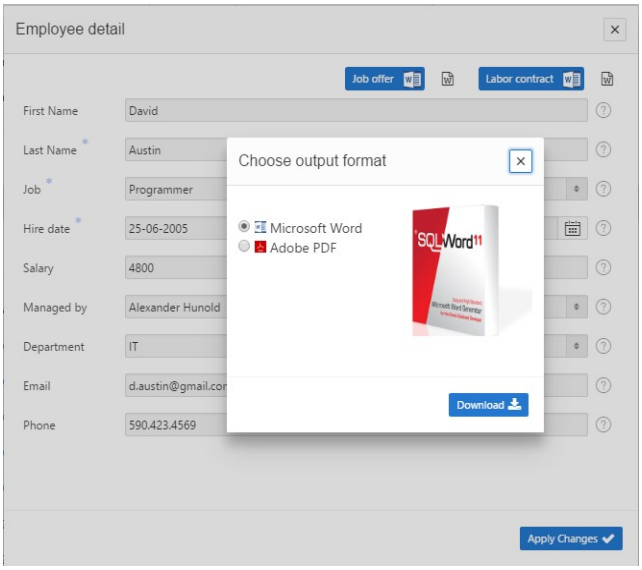
In this section the implemention of a SQLWord Job offer letter in Apex is explained.

Document HR_Employee_Job_Offer.docx is the source document for this letter.

- Start the SQLWord HR Demo application and choose Employees from the menu.
- Click on from employee David Austin to go to the Employee detail (modal) page 1021:



- Click on button Job Offer and choose option Microsoft Word to generate the Microsoft Word document.



Human Resources Inc.
Department IT
2014 Jabberwocky Rd
26192 Southlake
United States of America

Dear Mr./Ms. Austin,

Human Resources Inc. is pleased to offer you the position of Programmer. Your skills and experience will be an ideal fit for our IT department.

As we discussed, your starting date will be 25 june 2005.

The salary scale for this job ranges from $4,000 to $10,000 per month.

The salary is $57,600 per year and is paid on a monthly basis. Direct deposit is available.

Full family medical coverage will be provided through our company's employee benefit plan. Dental and optical insurance are also available.

Human Resource Inc. offers a flexible paid-time off plan which includes vacation, personal, and sick leave. Time off accrues at the rate of one day per month for your first year, then increases based on your tenure with the company.

We look forward to welcoming you to the Human Resource Inc. team.

Please let me know if you have any questions or I can provide any additional information.

Sincerely,

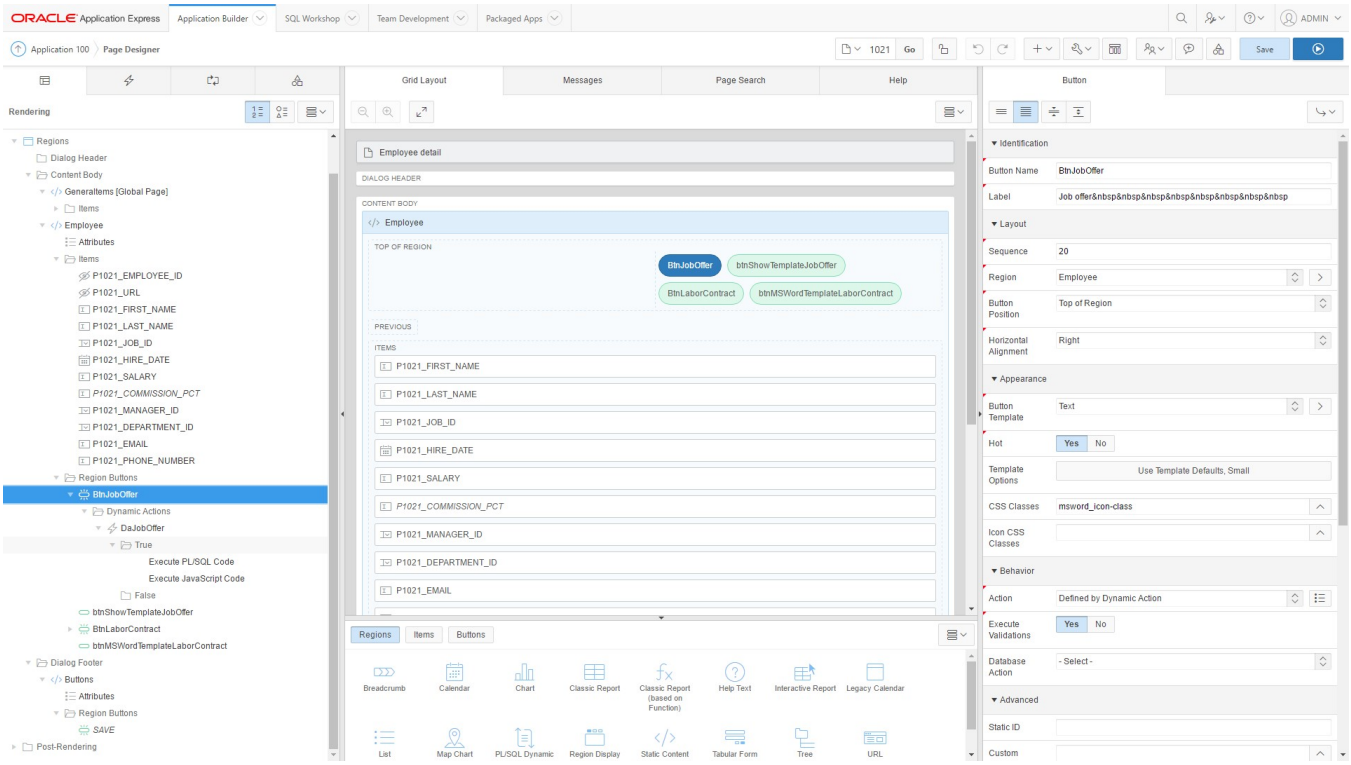
Alexander Hunold

Programmer, department IT

Human Resource Inc.

How does it work?

➢ Open page 1021. This page is a modal page called from page 1020



Button **BtnJobOffer** has a Dynamic Action **DaJobOffer** with 2 true actions:

- **Execute PL/SQL code**

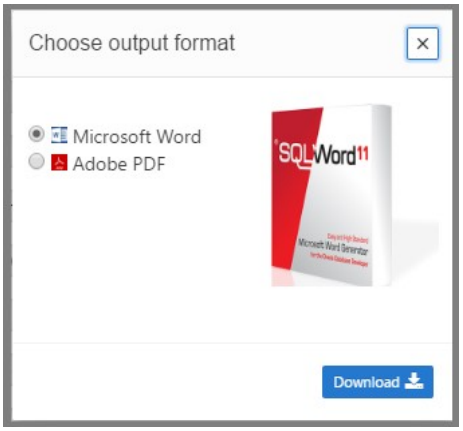
  The PL/SQL code fills page-item :P1021_URL with the url for calling page-501 in the next step.

  Page-0 items :P0_PK and P0_PROCEDURE are also set here.

- **Execute Javascript code**

```
var l_url = apex.item('P1021_URL').getValue();
eval(l_url);
```

This piece of javascript calls page 501 where the user is asked to choose an output format.



Button **BtnRun** on page 501 has a Dynamic Action **daRunSQLWord** with true action:

- **Execute Javascript code**

```
v_filetype = $('input[name=radioName]:checked', '#myForm').val();
parent.runSQLWord(v_filetype);
```

Here we call a javascript function runSQLWord from page 1020.

```
function runSQLWord(p_filetype) {
  apex.item('P0_FILETYPE').setValue(p_filetype);
  window.pageloadWait = apex.util.showSpinner();
  setTimeout(function(){ apex.event.trigger(document,'CustomEventRunSQLWord'); }, 10);
}
```

This function calls CustomEventRunSQLWord from page 1020 with true actions:

- **Execute PL/SQL code**

```
declare
  --
  l_blob          blob;
  l_filename      varchar2(4000);
  v_where_clause  varchar2(4000);
  v_report_id     varchar2(4000);
  --
begin
  --
  if apex_collection.collection_exists('SQLWORD_BLOB') then
    apex_collection.delete_collection('SQLWORD_BLOB');
  end if;
  --
  if :P0_PROCEDURE = 'HR_EMPLOYEE_SELECTION' then
    --
    l_filename := 'EmployeeSelection-' || to_char(sysdate,'dd-mm-yyyy-hh24.mi.ss') || '.' ||
                  lower(:P0_FILETYPE);
    --
    v_where_clause := 'where 1=1 ' || apex_ir_query.ir_query_where( app_id_in         => :APP_ID
                                                                  , page_id_in        => :APP_PAGE_ID
                                                                  , session_id_in     => :APP_SESSION
                                                                  , base_report_id_in => :P1020_REPORT_ID
                                                                  );
    --
    hr_employee_selection( p_where_clause => v_where_clause);
    --
  else
    --
    select substr(lower(replace(first_name || '_' || last_name || '.' ||
           lower(:P0_FILETYPE), ' ', '_')), 1, 4000)
    into   l_filename
    from   employees
    where  employee_id = :P0_PK;
    --
    if upper(:P0_PROCEDURE) = 'HR_EMPLOYEE_JOB_OFFER' then
      --
      l_filename := 'job_offer_' || l_filename;
      hr_employee_job_offer(p_employee_id => :P0_PK);
      --
    elsif upper(:P0_PROCEDURE) = 'HR_EMPLOYEE_CONTRACT' then
      --
      l_filename := 'employee_contract_' || l_filename;
      hr_employee_contract(p_employee_id => :P0_PK);
      --
    end if;
    --
  end if;
  --
  if upper(:P0_FILETYPE) = 'PDF' then
    --
    sqlword.save_output_pdf( p_utl_file_location => 'SQLWORD_OUTPUT_DIR'
                           , p_pdf_filename => l_filename );
    --
    l_blob := sqlword.get_output_pdf( p_utl_file_location =>'SQLWORD_OUTPUT_DIR'
                                    , p_pdf_filename => l_filename);
    --
  else
    l_blob := sqlword.get_output_docx;
  end if;
  --
  apex_collection.create_or_truncate_collection(p_collection_name => 'SQLWORD_BLOB');
  --
  apex_collection.add_member( p_collection_name => 'SQLWORD_BLOB'
                            , p_c001 => l_filename
                            , p_blob001 => l_blob );
  --
end;
```

As you can see in this piece of PL/SQL code we call generated SQLWord stored procedures and then get the output into a blob and put the blob into a collection which is picked up later from page 500.

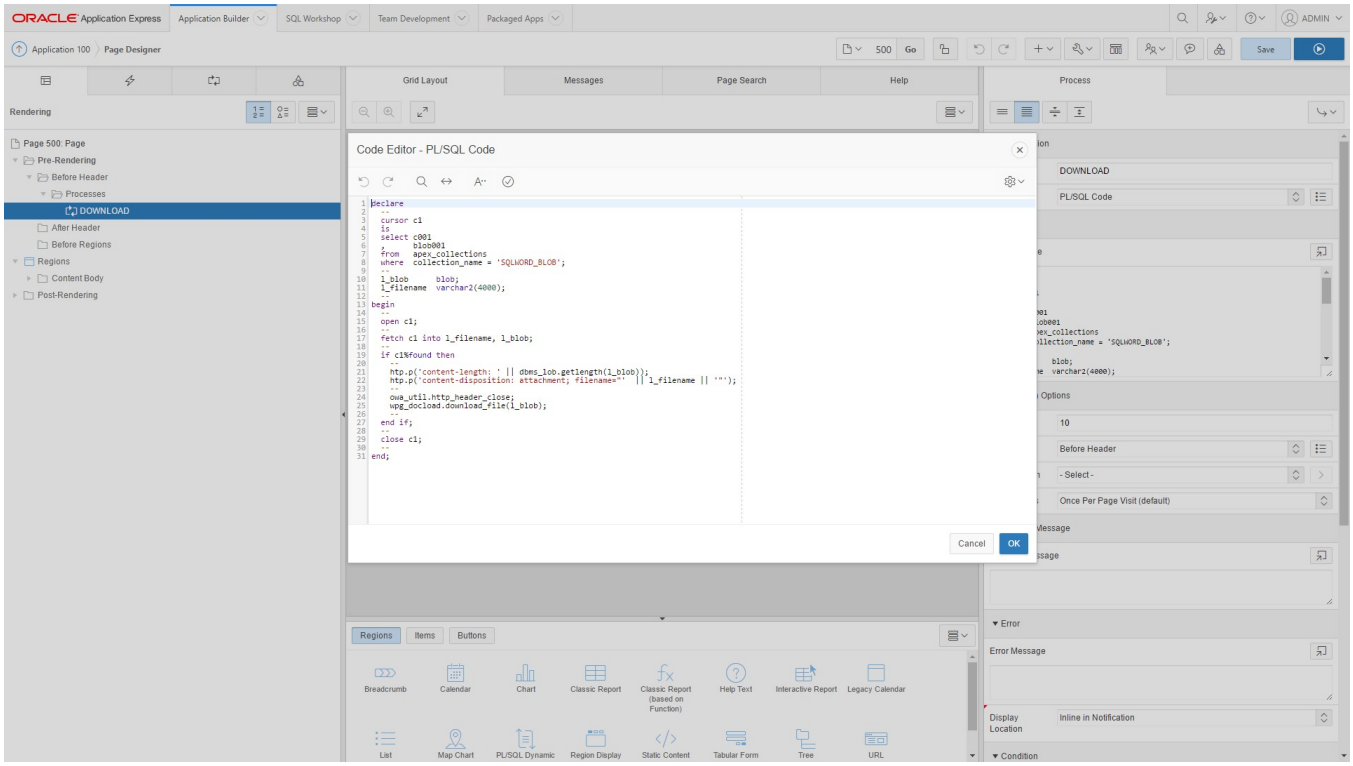
- **Execute Javascript code**

```
window.location = 'f?p=&APP_ID.:500:&APP_SESSION.::::::::';
window.pageloadWait.remove();
```

Here we call page 500 a generic page for downloading output documents.

➢ Open page 500

This is a empty page for downloading documents from a collection.



Examine the source code from Pre-Rendering/process/DOWNLOAD

```
declare
  --
  cursor c1
  is
  select c001
  ,      blob001
  from   apex_collections
  where  collection_name = 'SQLWORD_BLOB';
  --
  l_blob      blob;
  l_filename  varchar2(4000);
  --
begin
  --
  open c1;
  --
  fetch c1 into l_filename, l_blob;
  --
  if c1%found then
    --
    htp.p('content-length: ' || dbms_lob.getlength(l_blob));
    htp.p('content-disposition: attachment; filename="'  || l_filename || '"');
    --
    owa_util.http_header_close;
    wpg_docload.download_file(l_blob);
    --
  end if;
  --
  close c1;
  --
end;
```

Explanation

- The output document and filename is retrieved from the Apex collection SQLWORD_BLOB.
- The HTML header is prepared.
- By calling Apex procedure WPG_DOCLOAD.DOWNLOAD_FILE the download will start.
- Page 500 is closed and the browser returns back to page 1020.

PDF output (optional)

SQLWord can produce PDF output after the .docx generation by transforming the generated .docx to a .pdf file. This is done by calling a Java program on the Oracle server that uses the Aspose.Words for Java library. To use this functionality you need to buy a license at http://www.aspose.com. Default the evaluation version of Aspose.Words is installed which places some extra evaluation text in the pdf-file.

How does it work?

- First SQLWord generates a .docx file on the output directory on the Oracle server defined by a CREATE DIRECTORY command.
- When the .docx generation is finished then SQLWord runs an OS command using DBMS_SCHEDULER. It calls a shell script on the Oracle server that executes a Java program to transform the .docx file into a .pdf file.

Installation on Unix Oracle Server

We assume that you already have compiled all SQLWord demo examples by SQLWord Developer from C:\SQLWord11\Examples\Docx

Steps to follow:

- Create on the Oracle server a Unix user: sqlword

  All the scripts that we made assume that the home directory of the Unix sqlword user is /home/sqlword . If you have a different home directory (for example /opt/sqlword) then you need to edit these scripts:

  - sqlword_output_dir.sql
  - sqlword2pdf.sh
  - sqlword2pdf.java
- FTP the file sqlword2pdf.zip to the home directory of the sqlword user at /home/sqlword
- Unzip this file:

  $ cd /home/sqlword

  $ unzip sqlword2pdf.zip.

  Now a directory sqlword2pdf with 4 subdirectories are created (bin, log, output and sql)
- Change permissions:

  $ cd /home/sqlword

  $ chmod 777 sqlword2pdf

  $ cd /home/sqlword/sqlword2pdf

  $ chmod 777 bin output log

  $ cd /home/sqlword/sqlword2pdf/bin

  $ chmod +x *.sh
- Make sure that pam is properly installed. In particular make sure that the file libpam.so exists and is linked to a valid existing file.

  For exampe when you have Redhat OS create a link as root: # ln -s /lib64/libpam.so.0 /lib64/libpam.so
- Run script grants.sql as a DBA-user:

  SQL> @ grants.sql
- Edit script create_credential.and change the password to the password of the sqlword Unix user. Then run this script as the Oracle schema owner where you installed the SQLWord tables and packages

  SQL> @ create_credential.sql

- Run script sqlword_output_dir.sql as a DBA-user:

  SQL> @ sqlword_output_dir.sql

  Let's see if we now can create a .docx on the Oracle server by running script test_generate_docx.sql from the Oracle schema owner where you installed the SQLWord tables and packages

  SQL> @test_generate_docx.sql

  Now you should see the file example1a_out.docx at /home/sqlword/sqlword2pdf/output

- Download JDK 1.8 (or higher) from Oracle http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html and install it properly. Check if you have the right javac version

  $ javac –version should give back: javac 1.8 or higher

  NB: Make sure that the bin-directory of the JDK is in your PATH-environment variable !

  Run file compile.sh and check if the Java compilation created file sqlword2pdf.class.

  $ cd /home/sqlword/sqlword2pdf/bin

  $ ./compile.sh

  Now you have a Java executable that uses the evaluation version of Aspose.Words.

- If you bought an Aspose license then place your Aspose.Words.lic in directory /home/sqlword/sqlword2pdf/bin

  Edit file sqlword2pdf.java and remove the // comments.

  ```
  License license = new License();
  license.setLicense("Aspose.Words.lic");
  ```

  Again run file /home/sqlword/sqlword2pdf/bin/compile.sh and check if the Java compilation created

  a new class file at /home/sqlword/sqlword2pdf/bin/sqlword2pdf.class

- Now test if the Java progam sqlword2pdf.class can create a .pdf file:

  $ cd /home/sqlword/sqlword2pdf/bin

  $ ./sqlword2pdf.sh example1a_out.docx example1a_out.pdf

  Now you should see example1a_out.pdf in directory /home/sqlword/sqlword2pdf/output

- Now finally test if SQLWord now can generate pdf by running script test_generate_pdf.sql from the Oracle schema owner where you installed the SQLWord tables and packages.

  SQL> @test_generate_pdf.sql

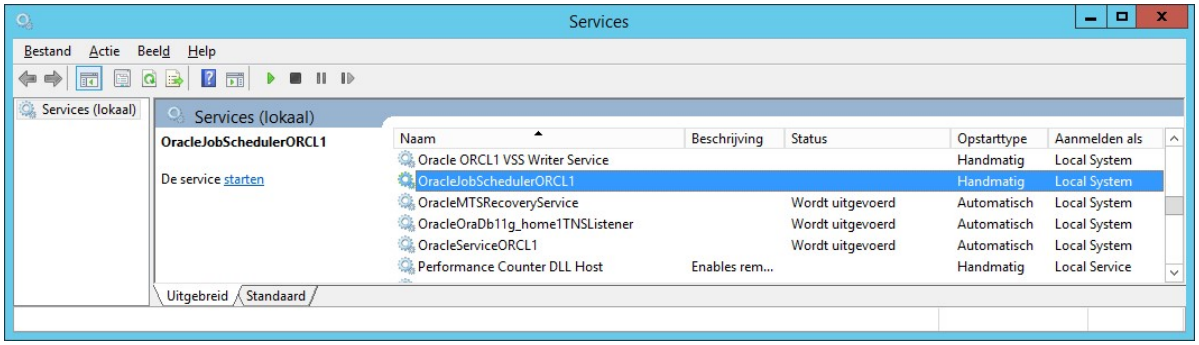
  Now you should see the file example1b_out.pdf at /home/sqlword/sqlword2pdf/output

Installation on Microsoft Windows Server

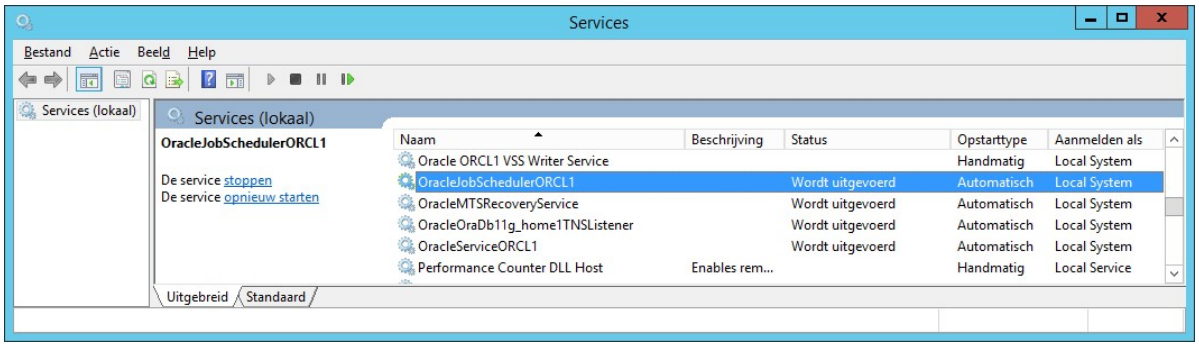
We assume that you already have compiled all SQLWord demo-examples with SQLWord Developer from C:\SQLWord11\Examples\Docx

Steps to follow:

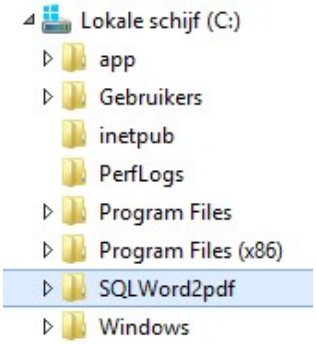
- Logon on your Windows Server as administrator and go to Services.



Start the Windows Service OracleJobScheduler%ORACLE_SID% and change the settings of this service to Automatic.



- Unzip the SQLWord2.zip to C:\SQLWord2pdf on your Windows server.



- Run script grants.sql as a DBA-user:

  SQL> @ grants.sql

- Run script sqlword_output_dir.sql as a DBA-user:

  SQL> @ sqlword_output_dir.sql

  Let's see if we now can create a .docx on the Oracle server by running script test_generate_docx.sql from Oracle schema owner where you installed the SQLWord tables and packages

  SQL> @test_generate_docx.sql

  Now you should see the file example1a_out.docx at C:\SQLWord2pdf\output

- Download JDK 1.8 (or higher) from Oracle http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html and install it properly.

  Check if you have the right javac version:

  NB: Make sure that the bin-directory of the JDK is in your PATH-environment variable !!!

  C:\SQLWord2pdf\bin> javac –version should give back: javac 1.8 or higher

- Run file compile.bat and check if the Java compilation created at C:\SQLWord2pdf\bin\sqlword2pdf.class

  C:\SQLWord2pdf\bin> compile.bat

  Now you have a Java executable that uses the evaluation version of Aspose.Words

- If you have an Aspose license then place your Aspose.Words.lic in directory C:\SQLWord2pdf\bin

  Edit file sqlword2pdf.java and remove the // comments.

  ```
  License license = new License();
  license.setLicense("Aspose.Words.lic");
  ```

  Again run file compile.bat and check if the Java compilation created

  a new class file at C:\SQLWord2pdf\bin\sqlword2pdf.class

- Now test if the Java progam sqlword2pdf.class can create a .pdf file:

  C:\SQLWord2pdf\bin> sqlword2pdf.bat example1a_out.docx example1a_out.pdf

  Now you should see example1a_out.pdf in directory C:\SQLWord2pdf\output

- Now finally test if SQLWord now can generate C:\SQLWord2pdf\output\example1b_out.pdf by running script test_generate_pdf.sql from the Oracle schema owner where you installed the SQLWord tables and packages

  SQL> @test_generate_pdf.sql

Apex implementation PDF expained

As already explained before there is a Custom event CustomEventRunSQLWord on page 1020 with a true action:

- **Execute PL/SQL code**

```
declare
  --
  l_blob          blob;
  l_filename      varchar2(4000);
  v_where_clause  varchar2(4000);
  v_report_id     varchar2(4000);
  --
begin
  --
  if apex_collection.collection_exists('SQLWORD_BLOB') then
    apex_collection.delete_collection('SQLWORD_BLOB');
  end if;
  --
  if :P0_PROCEDURE = 'HR_EMPLOYEE_SELECTION' then
    --
    l_filename := 'EmployeeSelection-' || to_char(sysdate,'dd-mm-yyyy-hh24.mi.ss') || '.' ||
                  lower(:P0_FILETYPE);
    --
    v_where_clause := 'where 1=1 ' || apex_ir_query.ir_query_where( app_id_in         => :APP_ID
                                                                  , page_id_in        => :APP_PAGE_ID
                                                                  , session_id_in     => :APP_SESSION
                                                                  , base_report_id_in => :P1020_REPORT_ID
                                                                  );
    --
    hr_employee_selection( p_where_clause => v_where_clause);
    --
  else
    --
    select substr(lower(replace(first_name || '_' || last_name || '.' ||
           lower(:P0_FILETYPE), ' ', '_')), 1, 4000)
    into   l_filename
    from   employees
    where  employee_id = :P0_PK;
    --
    if upper(:P0_PROCEDURE) = 'HR_EMPLOYEE_JOB_OFFER' then
      --
      l_filename := 'job_offer_' || l_filename;
      hr_employee_job_offer(p_employee_id => :P0_PK);
      --
    elsif upper(:P0_PROCEDURE) = 'HR_EMPLOYEE_CONTRACT' then
      --
      l_filename := 'employee_contract_' || l_filename;
      hr_employee_contract(p_employee_id => :P0_PK);
      --
    end if;
    --
  end if;
  --
  if upper(:P0_FILETYPE) = 'PDF' then
    --
    sqlword.save_output_pdf( p_utl_file_location => 'SQLWORD_OUTPUT_DIR'
                           , p_pdf_filename => l_filename );
    --
    l_blob := sqlword.get_output_pdf( p_utl_file_location =>'SQLWORD_OUTPUT_DIR'
                                    , p_pdf_filename => l_filename);
    --
  else
    l_blob := sqlword.get_output_docx;
  end if;
  --
  apex_collection.create_or_truncate_collection(p_collection_name => 'SQLWORD_BLOB');
  --
  apex_collection.add_member( p_collection_name => 'SQLWORD_BLOB'
                            , p_c001 => l_filename
                            , p_blob001 => l_blob );
  --
end;
```

- **Execute Javascript code**

```
window.location = 'f?p=&APP_ID.:500:&APP_SESSION.::::::::';
window.pageloadWait.remove();
```

Here we call page 500 a generic page for downloading output documents.

Explanation

- Procedure SQLWORD.SAVE_OUTPUT_PDF is called to create the PDF-file in the directory on your Oracle server.
- The output document is stored in a local variable L_BLOB by calling function SQLWORD.GET_OUPUT_PDF.
- An Apex collection is used to store the data.
- From the javascript we call page 500 where the PDF file will be downloaded.

Frequently asked questions

How can I create new pages in the output document?

You can call procedure SQLWORD.NEW_PAGE to create a new page.

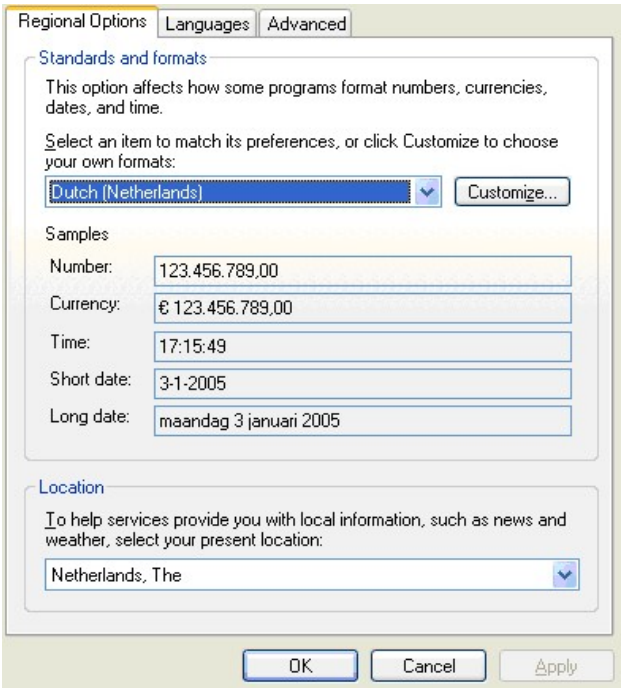
If you want to create new pages inside a for loop try this construction:

```
<%!
--
cursor c1
is
select ename
from   emp
order  by ename;
--
%>

<%for r1 in c1 loop%>
<%if c1%rowcount > 1 then sqlword.new_page; end if;%>
Employee: <%=r1.ENAME%>
<%end loop;%>
```

How can I change the presentation of decimal values in the output document?

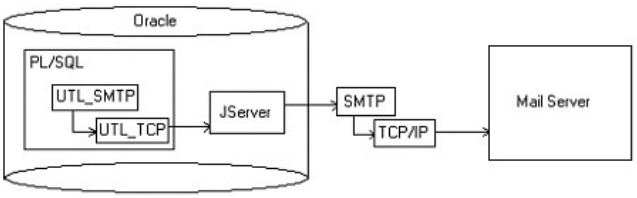
The presentation of decimal values in the output document depend on:

- The language settings from your Oracle database. Ask your DBA to check the value of parameter NLS_LANGUAGE (select * from V$NLS_PARAMETERS).
- The Microsoft Windows language settings on your PC. You can change this in the Windows Control panel in the “Regional and Language” options screen:



How can I send an output document by email from my Oracle database?

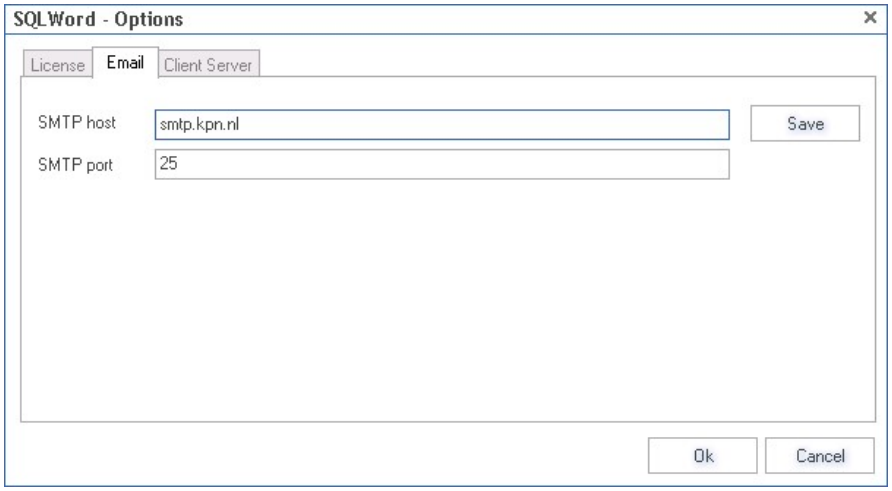
SQLWord can send an email from the Oracle database with the output document as an attachment. SQLWord uses the Oracle UTL_SMTP package:



- **ACL-access**

  Using UTL_SMTP to send email from your Oracle database has changed in Oracle 11g since accessing the remote network has changed. In Oracle 11g you have to configure (grant) each and every network access point using so called Access Control Lists (ACL's). Run SQL-script C:\SQLWord11\\SQL\ACL-access.sql as sysdba.

- Configure the email settings in the options screen from SQLWord Developer to your provider.



- Edit SQL-script C:\SQLWord11\Examples\SQL**send_email.sql** and change the address for the email_sender and email_recipients.

```
begin
  --
  example1a(p_employee_id => 121);
  --
  sqlword.send_email( p_from      => 'scott@tiger.com'
                    , p_to        => 'my_email@gmail.com'
                    , p_subject   => 'Hello, we send you a document
                                      generated by SQLWord 11'
                    , p_text_msg  => 'Hi,' || chr(10) || 'we send you a
                                      document generated by SQLWord 11'
                    , p_file_name => 'example1a_out.docx'
                    );
  --
end;
```

- Start SQL*Plus and run:

```
SQL> @ C:\SQLWord11\Examples\SQL\send_email.sql

PL/SQL-procedure successfully completed.

SQL>
```

- Now check your email and see if there is a new email with an MSWord-document attached to it.

How can I write an output document on the Oracle database server using UTL_FILE?

- Edit script C:\SQLWord11\Examples\SQL**write_utl_file.sql** and modify the file locations to your environment.

  First you must use the Oracle "create directory" command (ask your DBA to do this).

```
-----------------------------------------------------------------
create or replace directory SQLWORD_OUTPUT_DIR as 'C:\Temp';
grant read, write on directory SQLWORD_OUTPUT_DIR to public;
-----------------------------------------------------------------

begin
  --
  example1a(p_employee_id => 121);
  --
  sqlword.save_output_docx( p_utl_file_location => 'SQLWORD_OUTPUT_DIR'
                          , p_utl_file_filename => 'example1a_out.docx');
  --
end;
```

- Start SQL*Plus and run: **write_utl_file.sql**

```
SQL> @write_utl_file.sql

SQL> PL/SQL-procedure successfully completed.
```

- Check if the file is created on the specified file-location on your Oracle database server.

How can I save the output document into an Oracle table?

The output document is available as a BLOB through function GET_OUTPUT_DOCX which is available in package SQLWORD.

First create a table where you want to save the generated output

```
create table MY_OUTPUT
(       doc_name     varchar2(30)
,       dd_created   date
,       source       BLOB);
```

Now call stored procedure EXAMPLE1A and immediately get the generated Word document by calling function SQLWORD.GET_OUTPUT_DOCX.

```
begin
  --
  example1a(121);
  --
  insert into my_output
  (       doc_name
  ,       dd_created
```

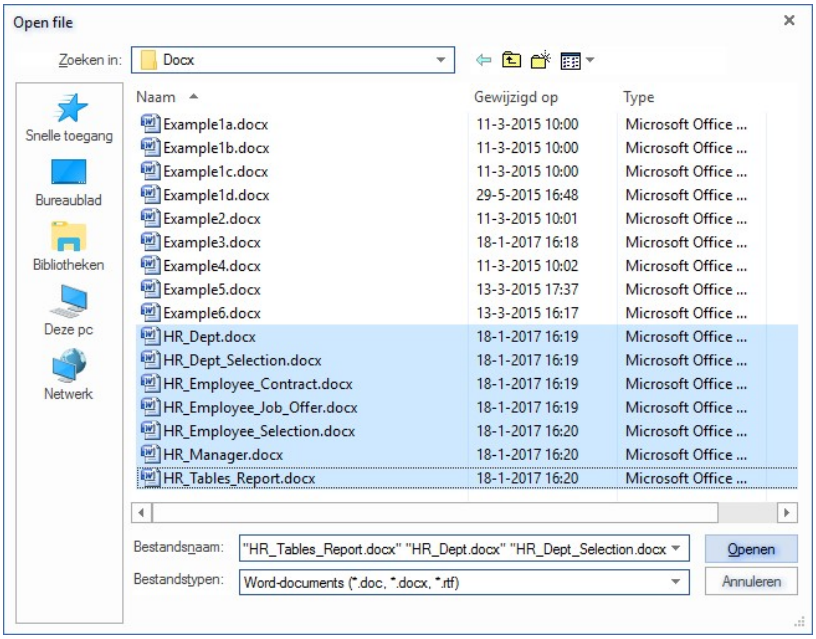
```
  ,       source)
  values
  (       'EXAMPLE1A_' || to_char(sysdate,'ddmmyyyy-hh24:mi:ss')
  ,       sysdate
  ,       sqlword.get_output_docx);
  --
  commit;
  --
end;
```

Hints & Tips

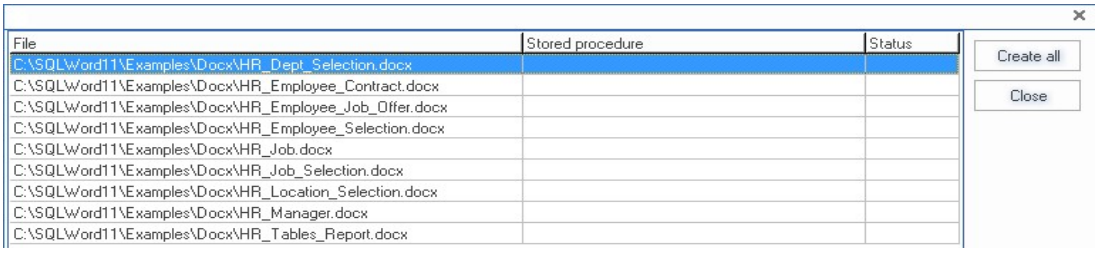
Compile multiple documents

You can quickly create stored procedures or recompile multiple source documents in one run.

Press the open file button from the SQLWord Developer toolbar and select the documents to compile:



The next screen shows up



Press the button "Create all" and the selected source documents are processed. If one or more is invalid after the compilation then the color of the row changes to red.

You can open a source document in Word by double-clicking on the row in the grid.

Clearing all scriptlets

All your scriptlets should be without any formatting code.

You can clear all <% tag %> scriptlets from invisible underwater formatting code by pressing the Clear all button on the SQLWord Developer toolbar.

Always place <% loop %> statements on a new line

To prevent a corrupt output document allways place loop statements on a new line.

Do not use constructions like this:

<%for r1 in c1 loop%><%=r1.department%><%end loop;%>

Use constructions like this:

<%for r1 in c1 loop%>

<%=r1.department%>

<%end loop;%>

Placing <%loop statements %> in a word table

To prevent a corrupt output document allways place <%for statements%> on the second row from a word table and put the <%end loop;%> on the last record.

Employee	Job	Salary
<%for r2 in c2 (r1.manager_id) loop%><%=r2.employee%>	<%=r2.job_title%>	<%=r2.salary%>
<%end loop;%>		

Employee<%for r2 in c2 (r1.manager_id) loop%>	Job	Salary
<%=r2.employee%>	<%=r2.job_title%>	<%=r2.salary%>

<%end loop;%>