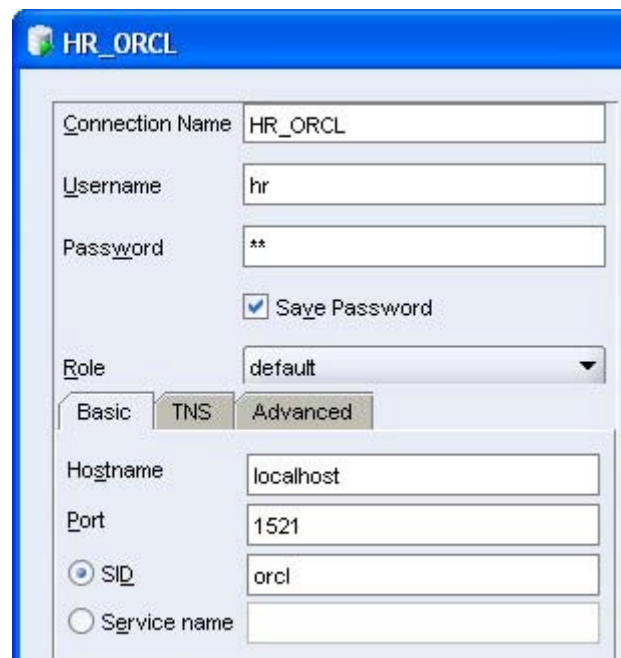


Remote Debugging with SQL Developer by Sue Harper

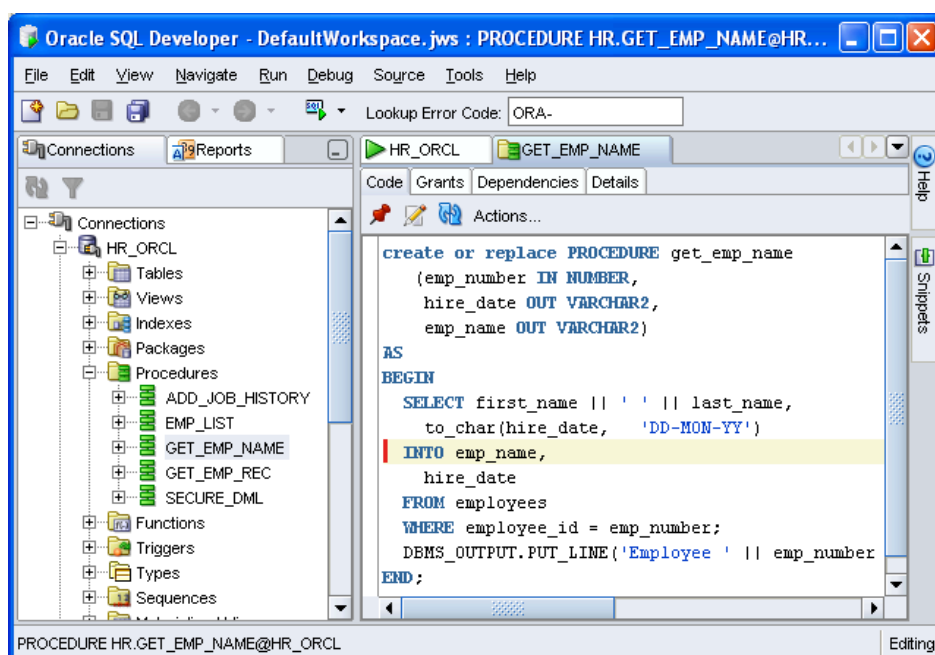
(available at : http://sueharper.blogspot.com/2006/07/remote-debugging-with-sql-developer_13.html)

It occurs to me that while we talk about using the remote debug facility in SQL Developer that you may not know how to use it. So in a few steps and with a few screen shots, I'd like to show you how it's done. (Please note, to see any of the images clearly, just double click on them.)

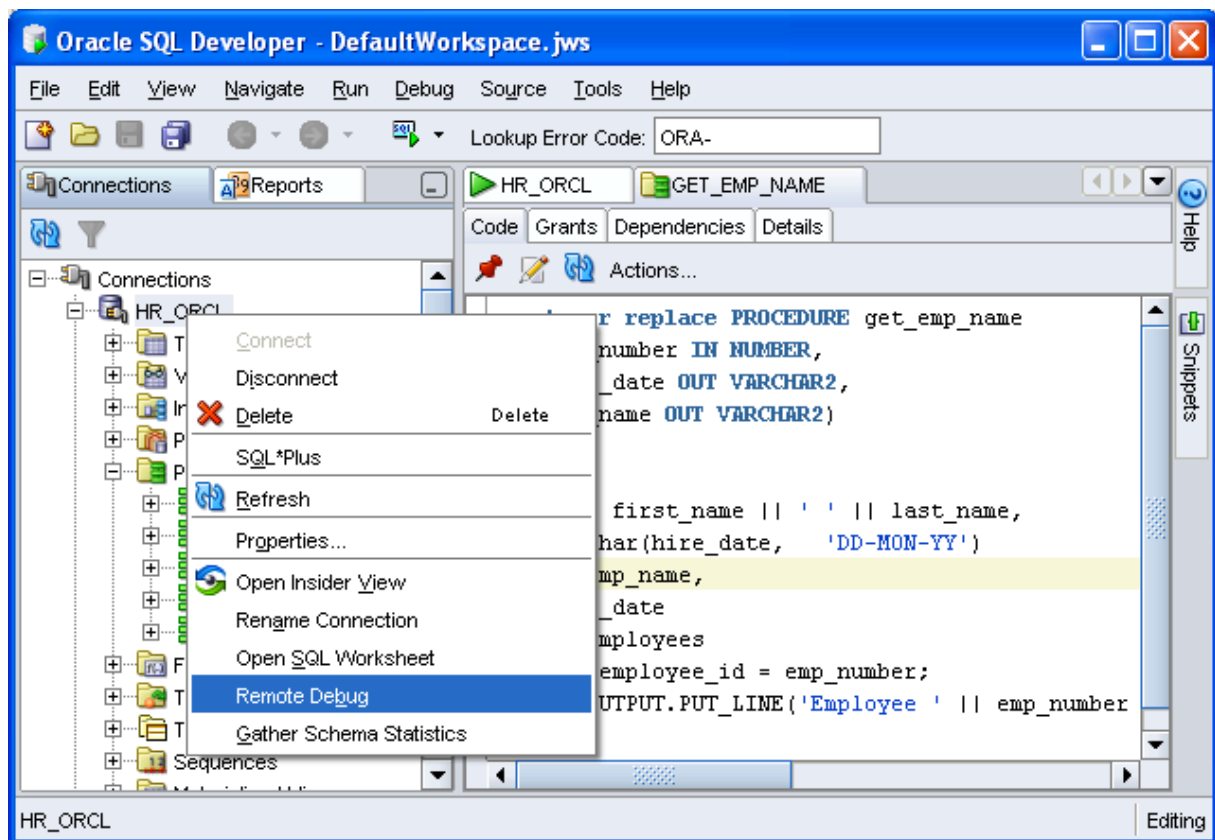
1. Let's start with a connection to the database. Create a database connection. (**File -> New Connection**) and complete the details. I only use the basic tab, so don't need to set up tnsnames or anything else. You'll see the database is on my own machine in this example, but it need not be.



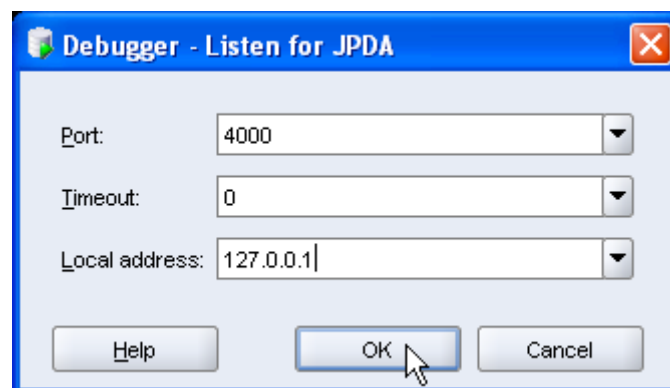
2. Now you can connect to any objects this user owns, using SQL Developer. You can browse the various objects the user has access to. I'm only interested in this procedure, which my user HR owns.



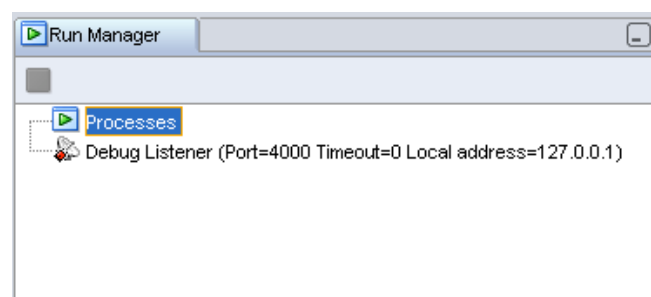
3. You can run and debug this procedure in SQL Developer, but that's not the purpose of the exercise. What we want to do is debug the procedure when it is executed from elsewhere, such as another programme or application. The starting point is to start a remote debug session in SQL Developer. You do this from the Database Connection as shown.



4. A dialog will display, requesting the listening port number and the IP address of the machine with the database. You can set the range of ports through a Preference in SQL Developer.



5. Once you have set the remote debug details, you should see the run manager display these.



6. Now you should start a remote session. Using a SQL *Plus command line session will do.

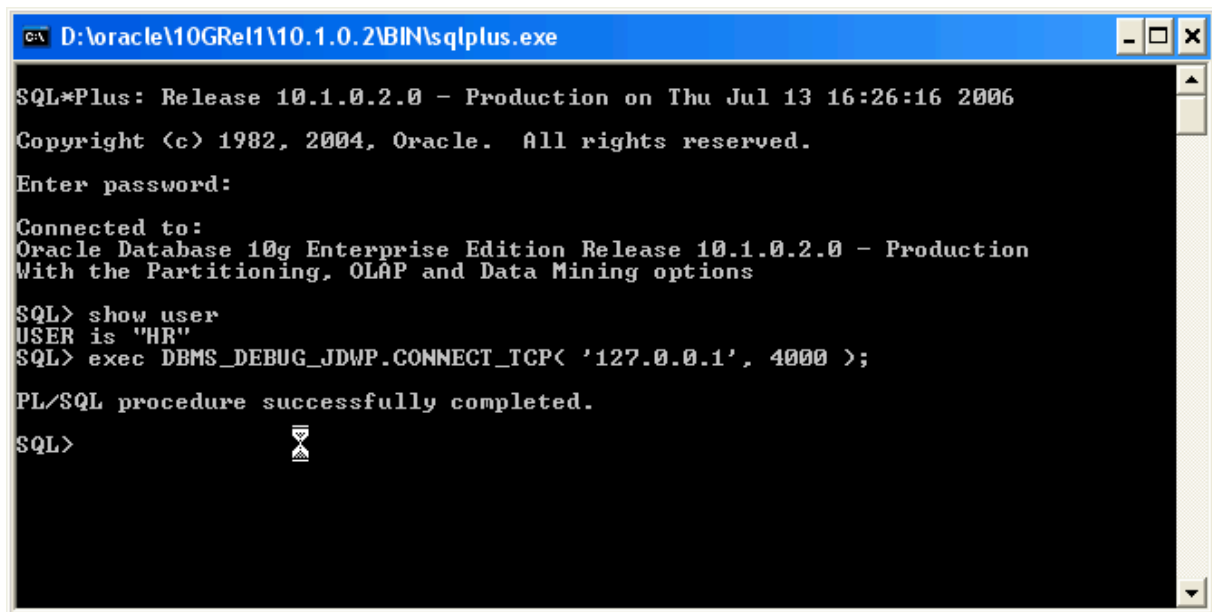
- * Invoke SQL *Plus for this user

- * In the SQL *Plus session enter the following command:

```
DBMS_DEBUG_JDWP.CONNECT_TCP('127.0.0.1', 4000 );
```

You should recognize the parameters from the previous dialog.

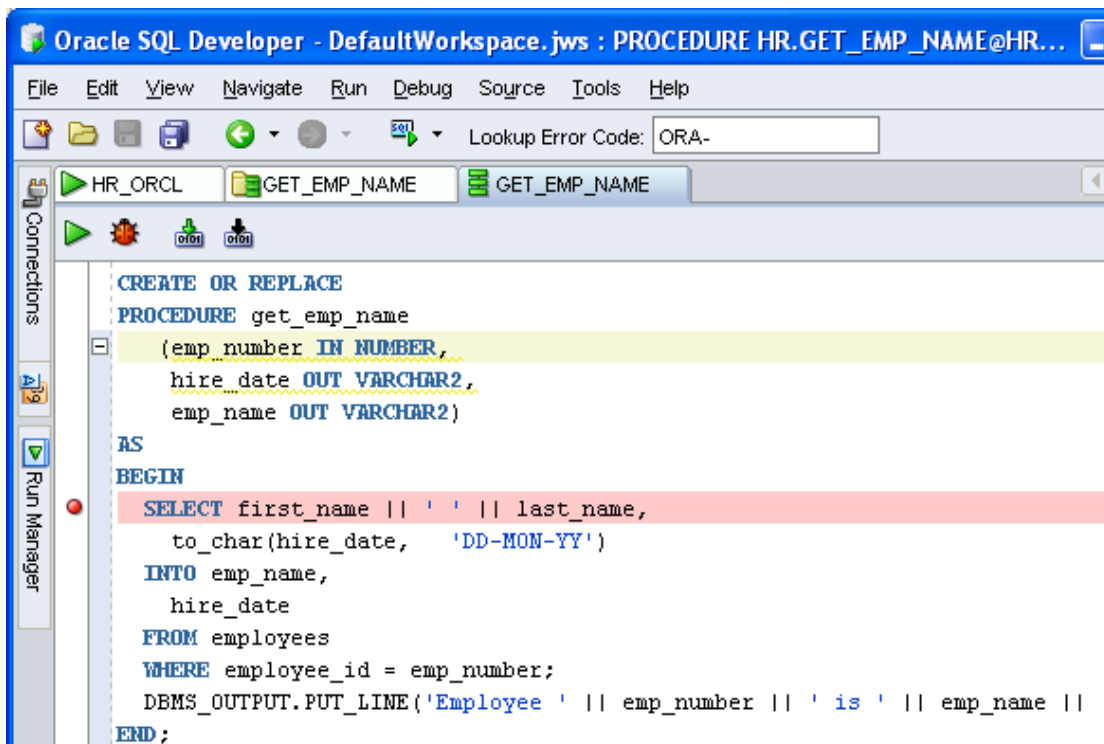
If you are debugging an application remotely, then this PL/SQL procedural call will need to go into that application, just for the debug purposes. You'll remove it afterwards.



The screenshot shows a Windows command prompt window titled "D:\oracle\10gRcl1\10.1.0.2\BIN\sqlplus.exe". The text inside the window is as follows:

```
SQL*Plus: Release 10.1.0.2.0 - Production on Thu Jul 13 16:26:16 2006
Copyright (c) 1982, 2004, Oracle. All rights reserved.
Enter password:
Connected to:
Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production
With the Partitioning, OLAP and Data Mining options
SQL> show user
USER is 'HR'
SQL> exec DBMS_DEBUG_JDWP.CONNECT_TCP('127.0.0.1', 4000 );
PL/SQL procedure successfully completed.
SQL>
```

7. Return to SQL Developer and set a breakpoint in the procedure.
Note: If you are debugging a procedure, you must remember to *Compile for Debug*, before you can start debugging.

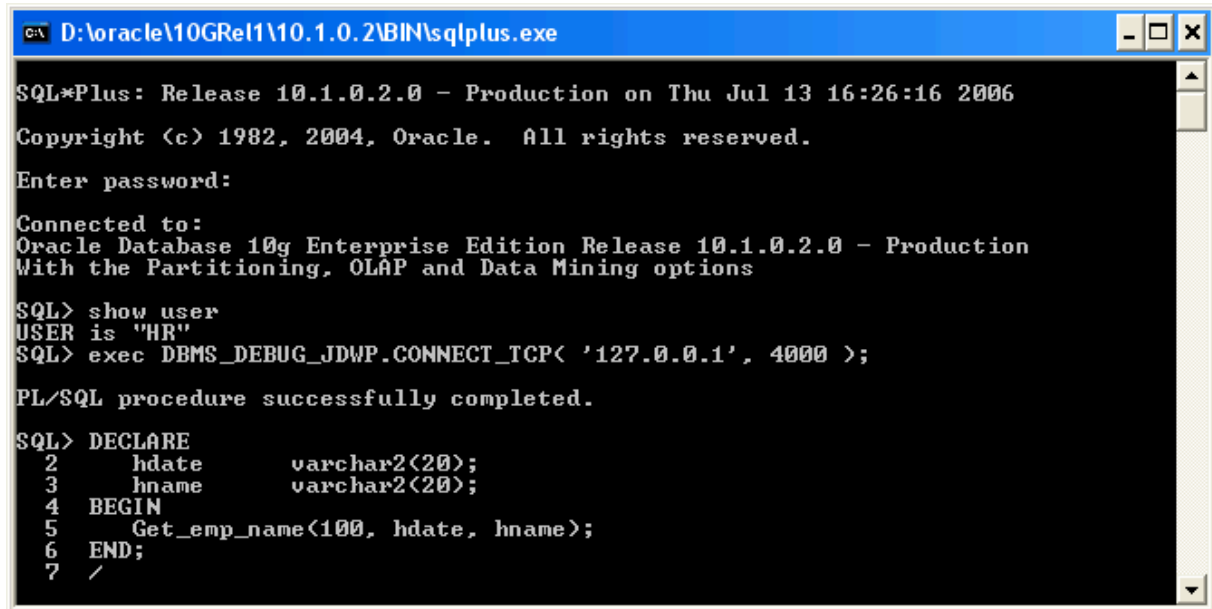


The screenshot shows the Oracle SQL Developer IDE. The title bar reads "Oracle SQL Developer - DefaultWorkspace.jws : PROCEDURE HR.GET_EMP_NAME@HR...". The menu bar includes File, Edit, View, Navigate, Run, Debug, Source, Tools, and Help. The toolbar has icons for file operations and a "Run" button. The "Connections" pane on the left shows a connection to "HR_ORCL". The main editor displays the following PL/SQL code:

```
CREATE OR REPLACE
PROCEDURE get_emp_name
(
  emp_number IN NUMBER,
  hire_date OUT VARCHAR2,
  emp_name OUT VARCHAR2
)
AS
BEGIN
  SELECT first_name || ' ' || last_name,
         to_char(hire_date, 'DD-MON-YY')
  INTO emp_name,
       hire_date
  FROM employees
  WHERE employee_id = emp_number;
  DBMS_OUTPUT.PUT_LINE('Employee ' || emp_number || ' is ' || emp_name || '
END;
```

8. Now you need to return to the SQL*Plus session and execute your procedure. If you debug in SQL Developer, an anonymous block is created for you to execute the procedure. In this case you'll need to write one to execute the procedure from SQL*Plus.

Note: In the procedure we have a DBMS_OUTPUT command, so you should also add the 'Set Serveroutput on' command.



```
C:\> D:\oracle\10GRel1\10.1.0.2\BIN\sqlplus.exe

SQL*Plus: Release 10.1.0.2.0 - Production on Thu Jul 13 16:26:16 2006
Copyright (c) 1982, 2004, Oracle. All rights reserved.

Enter password:

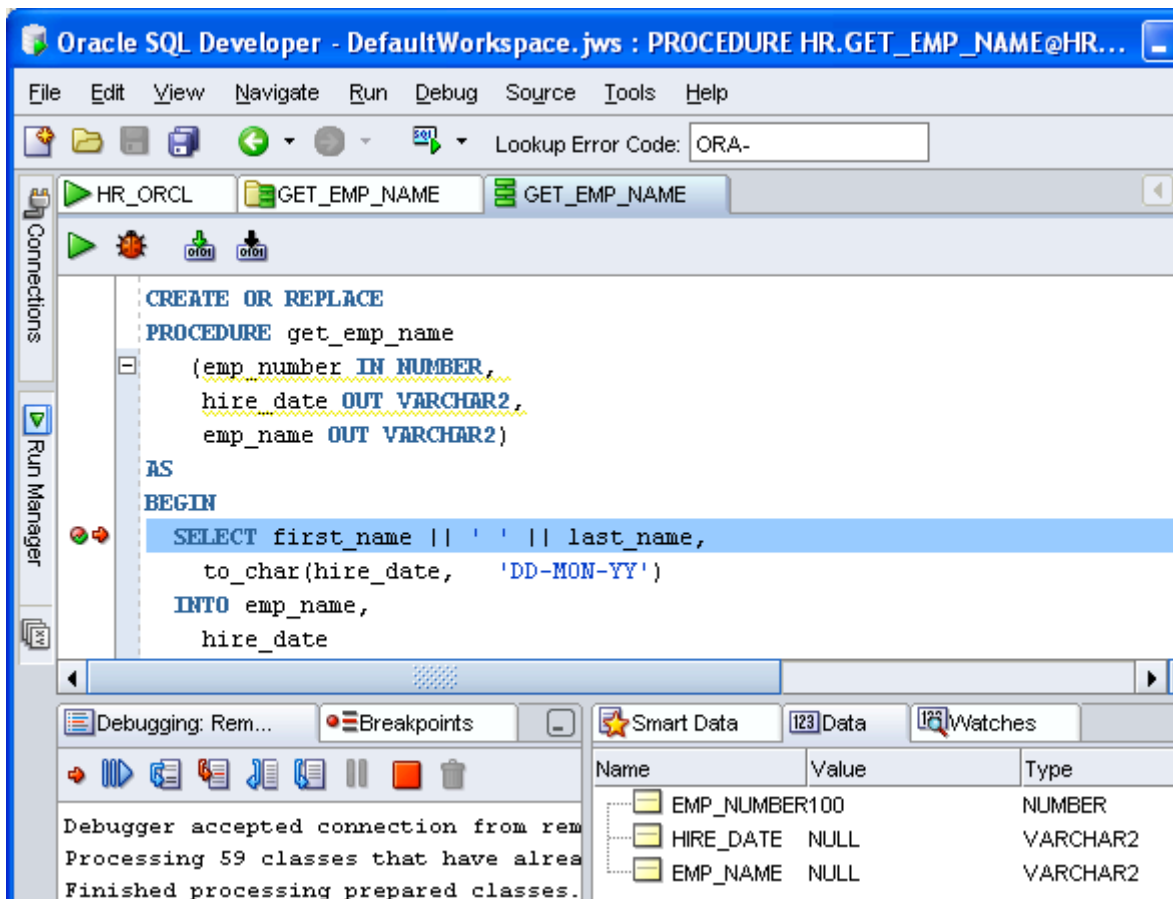
Connected to:
Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> show user
USER is 'HR'
SQL> exec DBMS_DEBUG_JDWP.CONNECT_TCP( '127.0.0.1', 4000 );

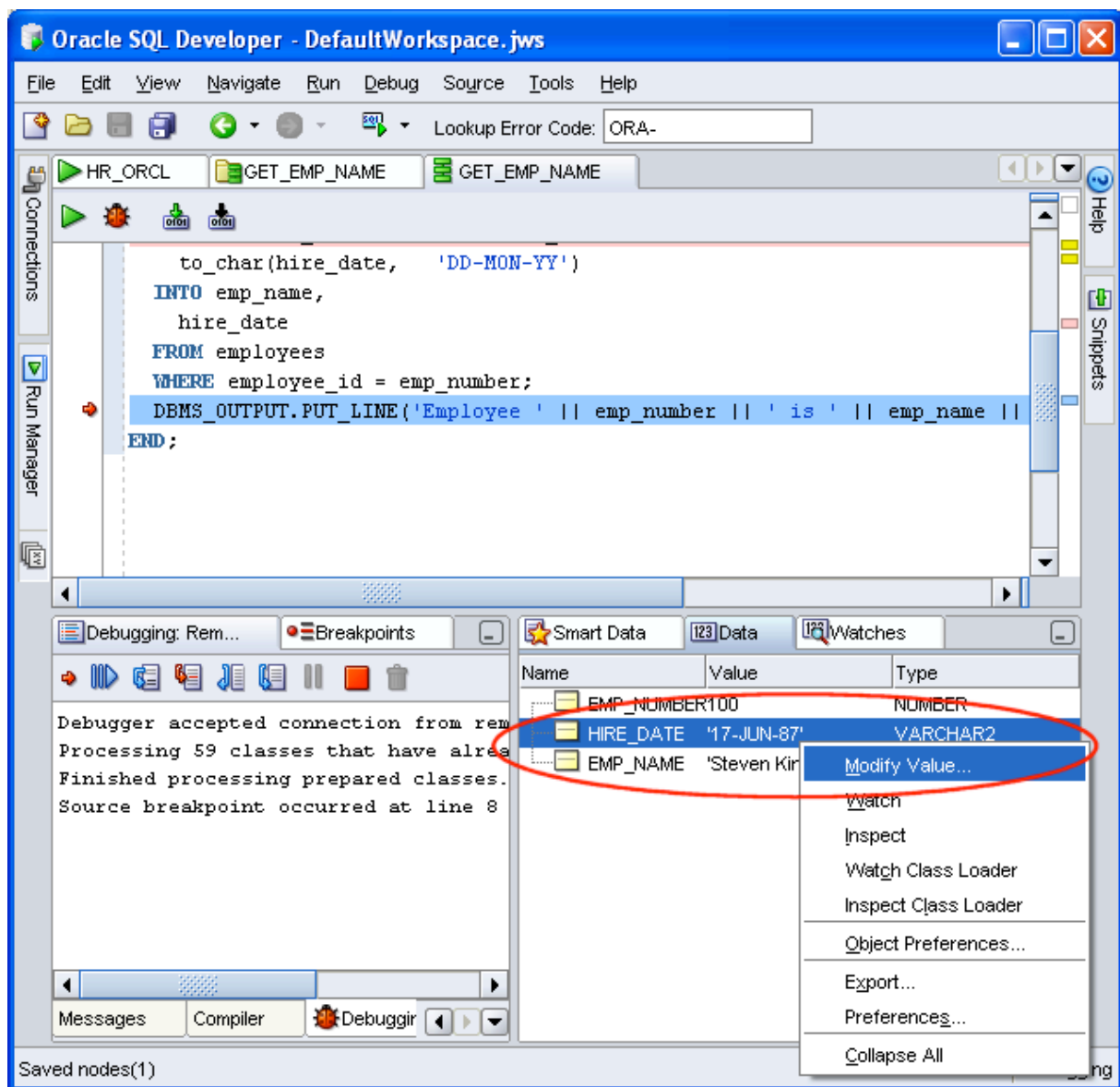
PL/SQL procedure successfully completed.

SQL> DECLARE
2      hdate      varchar2(20);
3      hname      varchar2(20);
4  BEGIN
5      Get_emp_name(100, hdate, hname);
6  END;
7  /
```

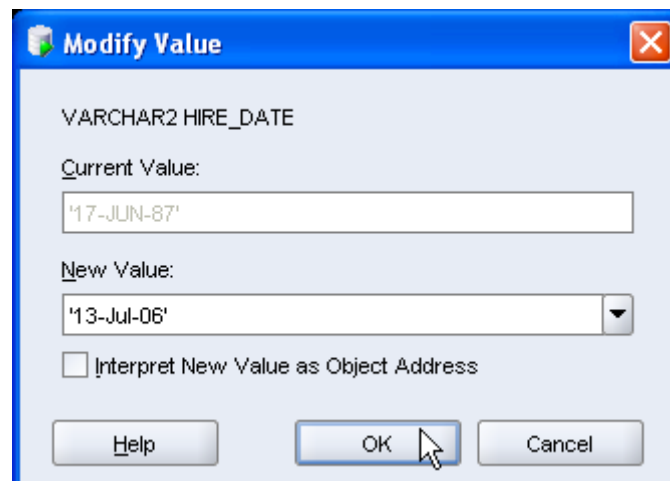
9. As soon as you execute the anonymous block, you'll *be* returned to SQL Developer to debug the session there. Step into the code as you would in a usual debug session.



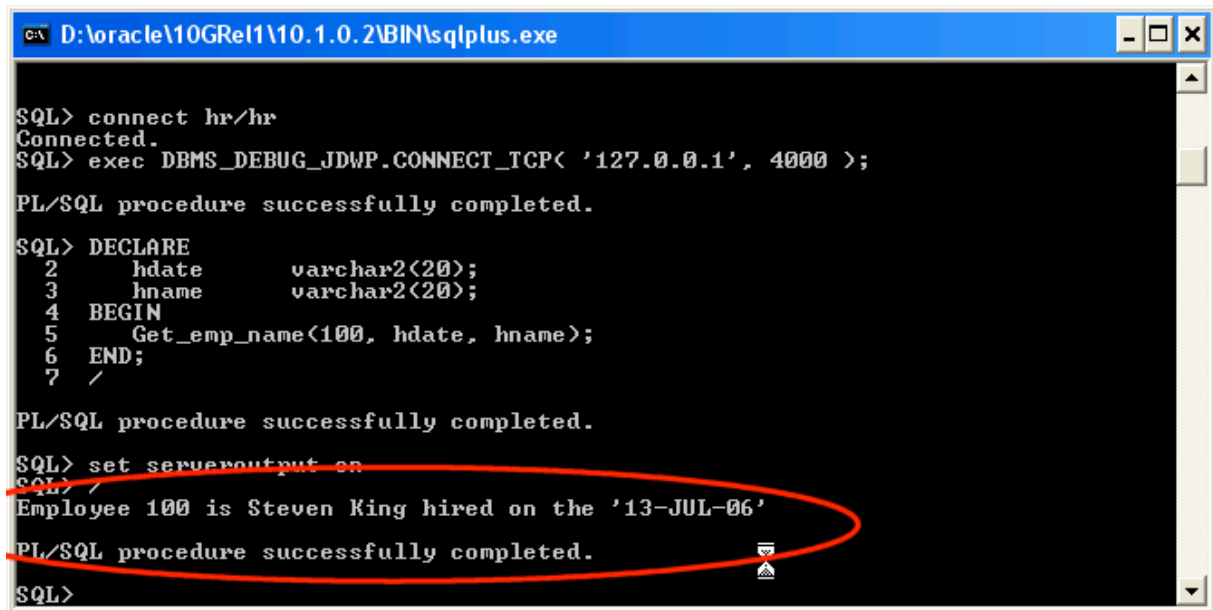
10. You can watch data and modify values in the same way as a normal debug session.



In this example, I'll modify the hire date.



11. Once you have reviewed the data, or made the modifications you want, resume debugging. Once the debug session is complete the control is returned to your remote session. In this case, SQL*Plus. You'll notice the modified date reflected in the output.



```
SQL> connect hr/hr
Connected.
SQL> exec DBMS_DEBUG_JDWP.CONNECT_TCP( '127.0.0.1', 4000 );

PL/SQL procedure successfully completed.

SQL> DECLARE
2     hdate      varchar2(20);
3     hname      varchar2(20);
4 BEGIN
5     Get_emp_name(100, hdate, hname);
6 END;
7 /

PL/SQL procedure successfully completed.

SQL> set serveroutput on
SQL> /
Employee 100 is Steven King hired on the '13-JUL-06'

PL/SQL procedure successfully completed.

SQL>
```

Quite a long piece, I know, but I think we often veer away from the unknown or untried. Hopefully once you have walked through an example, you might be able to make more use of this very useful feature.