

Testing with utPLSQL – Made Easy with SQL Developer

Philipp Salvisberg



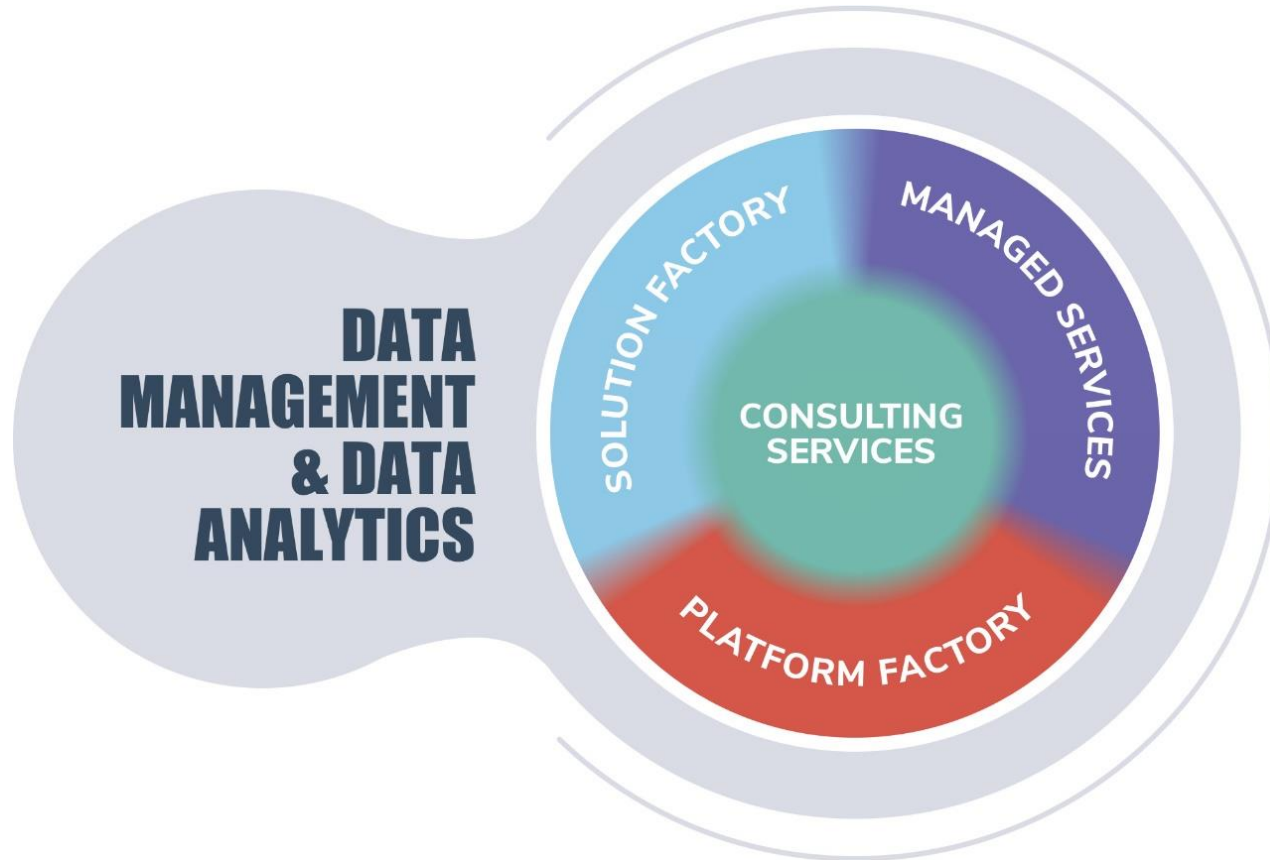
phsalvisberg

DOAG2018

trivadis
makes IT easier. ■ ■ ■



■ About Us – Added Value from Data



■ About Me

- Trivadian since April 2000
 - Senior Principal Consultant, Partner
 - Member of the Board of Directors
 - [@phsalvisberg](https://www.salvis.com/blog)
 - <https://www.salvis.com/blog>
 - <https://github.com/PhilippSalvisberg>
- Database centric development with Oracle database
- Model Driven Software Development
- Author of free SQL Developer Extensions PL/SQL Unwrapper, PL/SQL Cop, utPLSQL, plscope-utils, oddgen and Bitemp Remodeler



ORACLE®
ACE

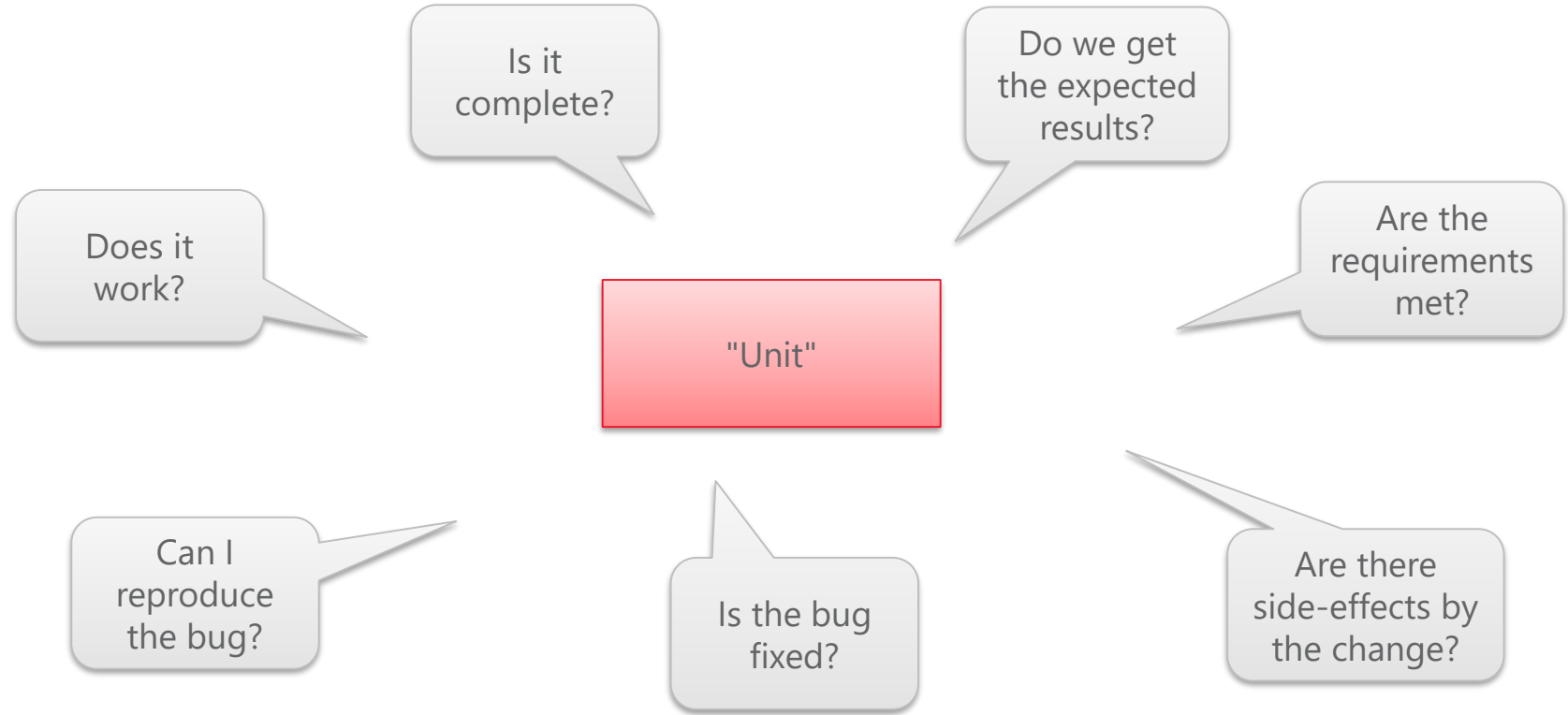
trivadis
makes IT easier. ■ ■ ■

■ Agenda

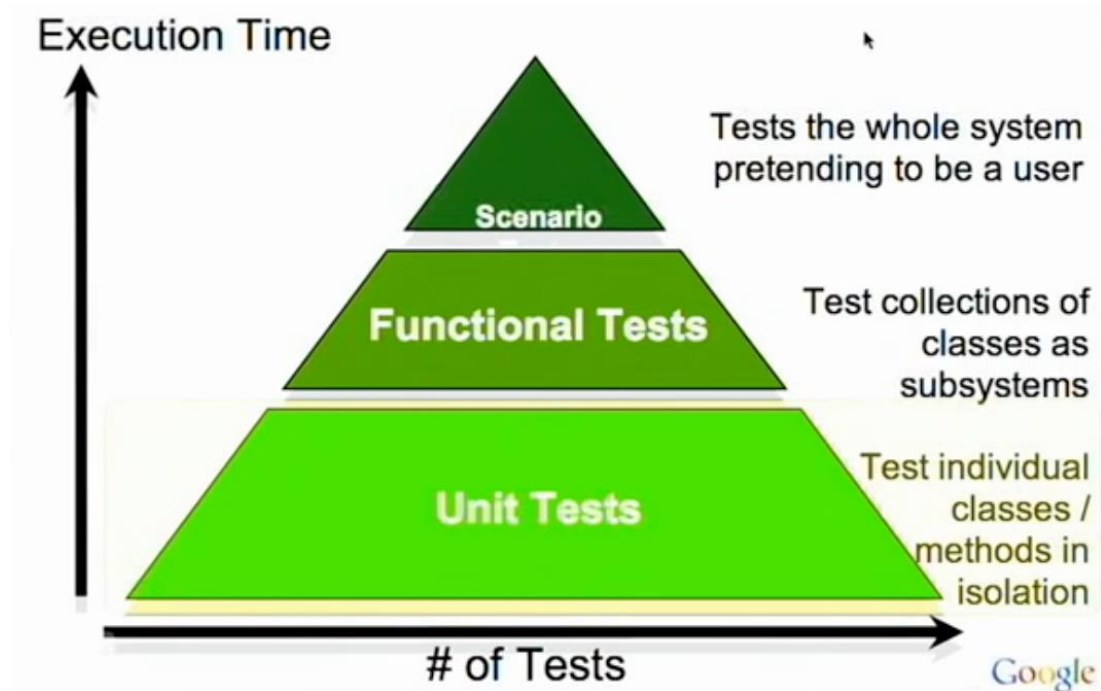
1. Introduction
2. Installation
3. Build & Run Tests in SQL Developer
4. Run Code Coverage Reports in SQL Developer
5. Core Messages

Introduction

■ Why?



■ utPLSQL Test Scope



Source: Miško Hevery, The Clean Code Talks, Unit Testing, October 30, 2008,
<https://www.youtube.com/watch?v=wEhu57pih5w&t=991>

■ GUI

■ API

■ Integration

■ Components

■ Unit

utPLSQL

■ utPLSQL Units Under Test

Primary

- Types
- Packages
- Procedures
- Functions



Secondary

- Non-PL/SQL Units
- Views
- Triggers
- Tables



■ utPLSQL Suite – Open Source – Apache 2.0 License

Mandatory

- Core Testing Framework
 - Schema installed in Oracle DB
 - No repository
 - Annotation based tests

Optional

- Command Line Client
- Maven Plugin
- SQL Developer Extension



■ Test Declaration

```
CREATE OR REPLACE PACKAGE test_package_name AS
```

```
  --%suite
```

```
  --%test
```

```
  PROCEDURE procedure_name;
```

```
END;
```

--%displayname(<description>)

--%test(<description>)

--%throws(<exception>[,...])

--%beforeall

--%afterall

--%beforeeach

--%aftereach

--%beforetest([...])

--%aftertest([...])

--%rollback(manual)

--%disabled

--%suite(<description>)

--%suitepath(<path>)

--%displayame(<description>)

--%beforeall([...])

--%afterall([...])

--%beforeeach([...])

--%aftereach([...])

--%rollback(manual)

--%disabled

--%context

--%endcontext

■ Test Implementation

```
CREATE OR REPLACE PACKAGE BODY test_package_name AS
  PROCEDURE procedure_name IS
    l_actual    INTEGER := 0;
    l_expected  INTEGER := 1;
  BEGIN
    ut.expect(l_actual).to_equal(l_expected);
  END procedure_name;
END;
```

Matcher:

be_between, be_empty, be_false,
be_greater_than, be_greater_or_equal,
be_less_or_equal, be_less_than, be_like,
be_not_null, be_null, be_true, **equal**,
have_count, match

Extended options for refcursor, object
type, nested table and varray:

- include(<items>)
- exclude(<items>)
- unordered
- join_by(<items>)

■ Test Run

```
SET SERVEROUTPUT ON SIZE UNLIMITED  
EXEC ut.run('test_package_name')
```

```
test_package_name  
  procedure_name [.003 sec] (FAILED - 1)
```

Failures:

1) procedure_name

Actual: 0 (number) was expected to equal: 1 (number)
at "TEST_PACKAGE_NAME.PROCEDURE_NAME", line 7 ut.expect(1_actual).to_equal(1_expected);

Finished in .007015 seconds

1 tests, 1 failed, 0 errored, 0 disabled, 0 warning(s)

Installation

■ Install utPLSQL Core Testing Framework

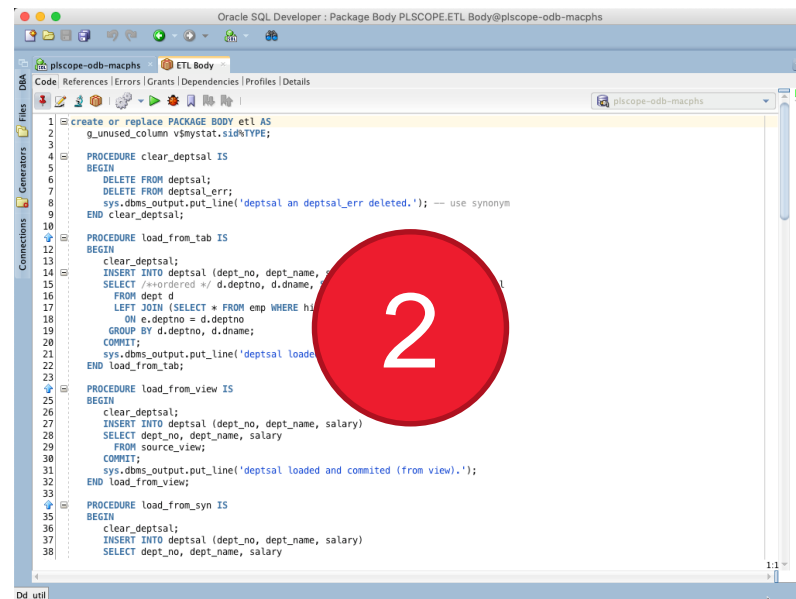
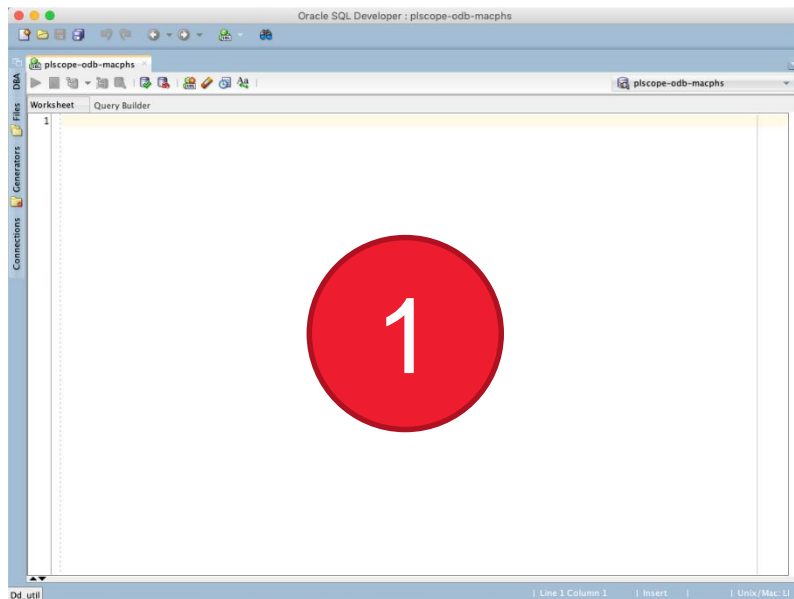
- Download utPLSQL.zip from <https://github.com/utPLSQL/utPLSQL/releases>
- Unzip utPLSQL.zip
- cd source
- sqlplus / as sysdba @install_headless.sql
 - User UT3
 - Password XNtxj8eEgA6X6b6f
 - Tablespace USERS

■ Install utPLSQL for SQL Developer

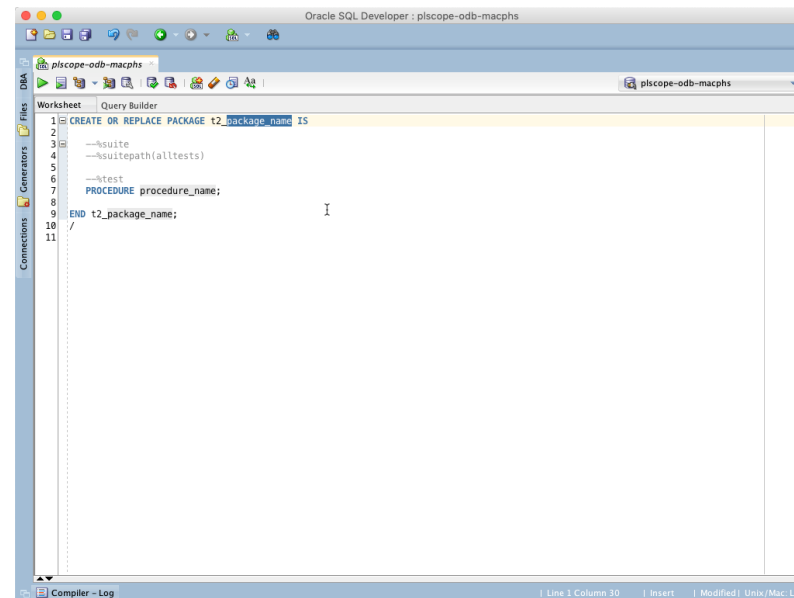
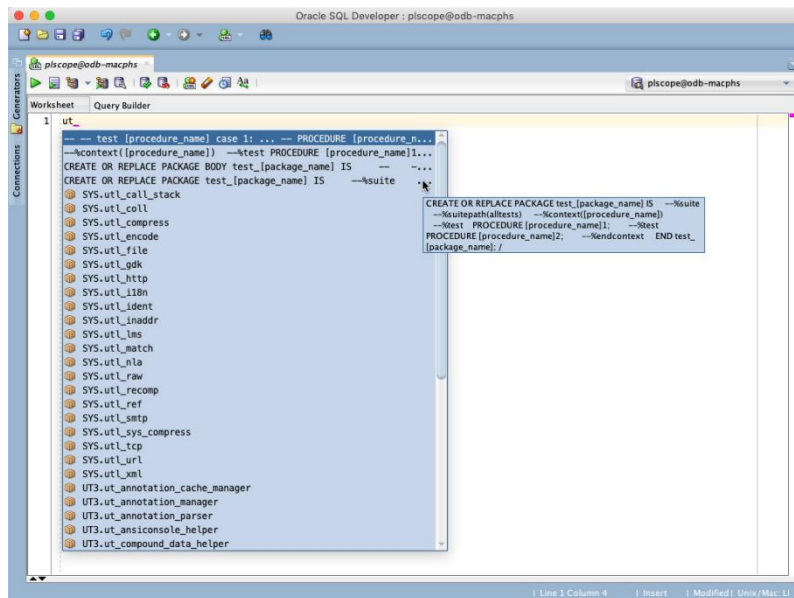
- Download utplsqli_for_SQLDev_*.zip
from <https://github.com/utPLSQL/utPLSQL-SQLDeveloper/releases>
- Start SQL Developer
- Select "Check for Updates..." in the help menu
- Use the "Install From Local File" option to install the previously downloaded "utplsqli_for_SQLDev_*.zip" file
 - User must have read/write access to SQL Developer installation directory (run as Administrator, if required)
- Restart SQL Developer

Build & Run Tests in SQL Developer

Starting Point?



Test First – Create Test from Template





Test First – Complete Test & Run



Oracle SQL Developer : plscope-odb-macphs

```
1 CREATE OR REPLACE PACKAGE t2_etl IS
2
3   --suite
4   --suitepath(alltests)
5   --rollback(manual)
6
7   --test
8   PROCEDURE load_from_tab;
9
10 END t2_etl;
11 /
12
13 CREATE OR REPLACE PACKAGE BODY t2_etl IS
14
15   PROCEDURE load_from_tab IS
16     l_actual sys_refcursor;
17     l_expected sys_refcursor;
18   BEGIN
19     -- populate actual
20     delete from deptsal;
21     etl.load_from_tab;
22     open l_actual for select * from deptsal;
23
24     -- populate expected
25     open l_expected for select * from source_view;
26
27     -- assert
28     ut.expect(l_actual).to_equal(l_expected);
29   END load_from_tab;
30
31 END t2_etl;
32 /
33
```

0.119 seconds

plscope-odb-macphs

Worksheet Query Builder

Run utPLSQL test

Oracle SQL Developer : plscope-odb-macphs-1

```
1 SET SERVEROUTPUT ON SIZE UNLIMITED
2 EXECUTE ut.run('t2_etl.load_from_tab');
```

Task completed in 0.438 seconds

alltests
t2_etl
load_from_tab [.195 sec]
deptsal an deptsal_err deleted.
deptsal loaded and committed (from table).

Finished in .199262 seconds
1 tests, 0 failed, 0 errored, 0 disabled, 0 warning(s)

PL/SQL procedure successfully completed.

0.438 seconds

plscope-odb-macphs-1

Worksheet Query Builder

Script Output

Compiler - Log



Configure utPLSQL



Preferences

Search: utp|

utPLSQL

Run utPLSQL test

Open an unshared worksheet? ☒

Reset package before running utPLSQL? ☐

Clear script output panel before running utPLSQL? ☐

Execute unit test automatically? ☒

Check availability of "Run utPLSQL test" menu option? ☐

Generate utPLSQL test

Test package prefix: t3_

Test package suffix:

Test unit prefix:

Test unit suffix:

Number of tests to generate per unit: 1

Generate comments? ☒

Disable tests? ☐

Suite path: alltests

Indent spaces: 3

Check availability of "Generate utPLSQL test" menu option? ☐ [Create code templates](#)

odtgen

Root folder in Generators view: utPLSQL

Generate files? ☒

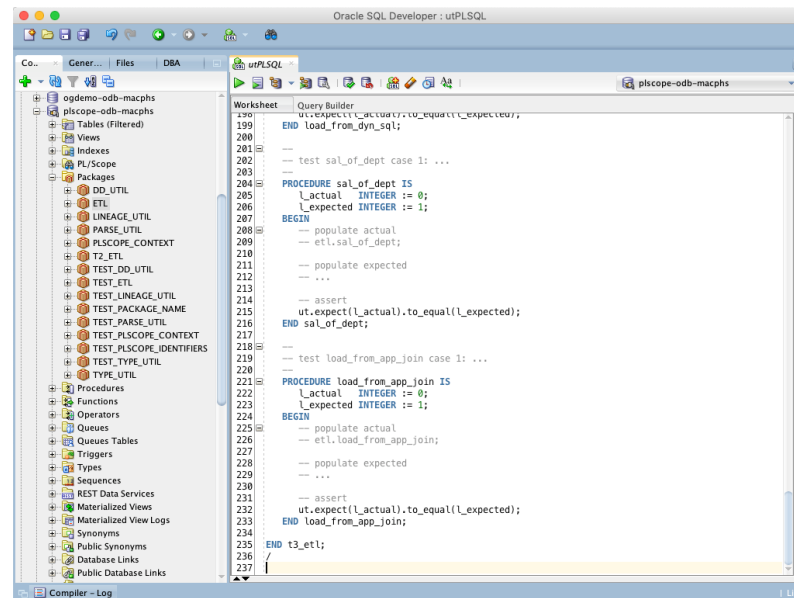
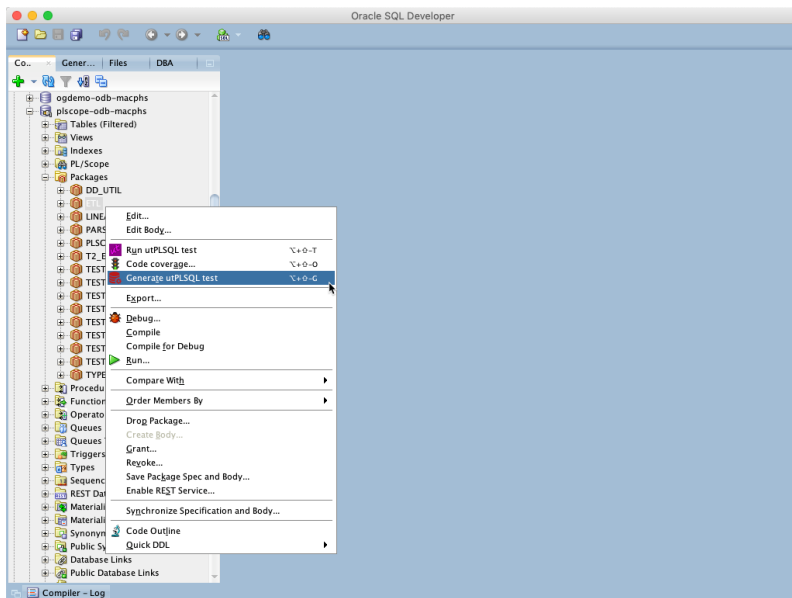
Output directory: /Users/phs/utplsqli/generated [Browse](#)

Delete existing files in output directory? ☐

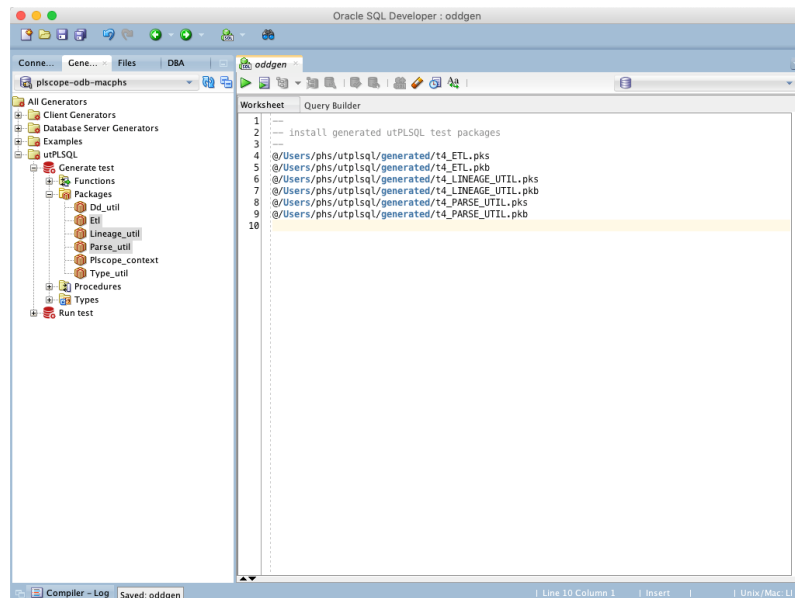
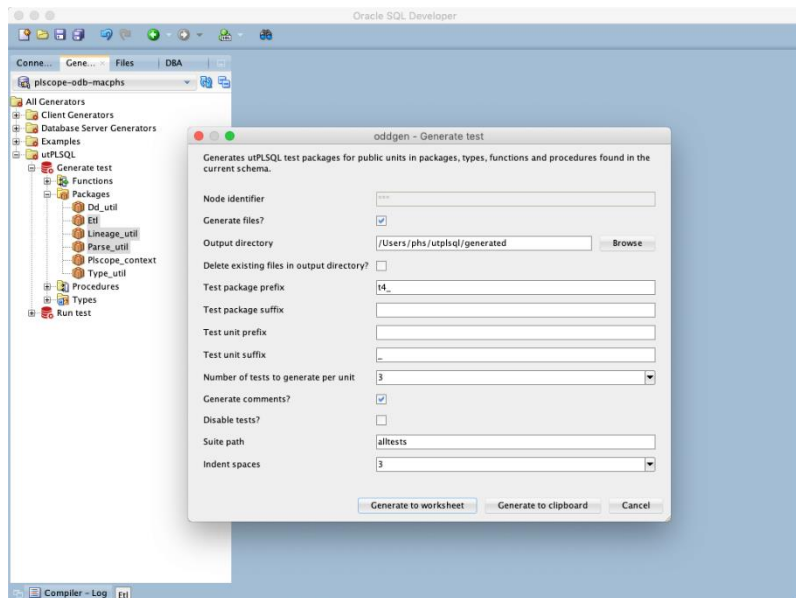
[Help](#) [OK](#) [Cancel](#)



Test Last – Create Test from Existing Code



Test Last – Generate Multiple Test Skeletons



Run Code Coverage Reports in SQL Developer

■ Code Coverage – Defintion

A measure used to describe the degree to which the source code of a program is executed when a particular test suite runs.

Source: https://en.wikipedia.org/wiki/Code_coverage

■ Line Coverage

```
CREATE OR REPLACE FUNCTION f(a IN INTEGER) RETURN INTEGER IS
BEGIN
    IF a IS NULL THEN
        RETURN 0;
    ELSE
        RETURN a*a;
    END IF;
END f;
/
```

Two test cases for
100% coverage

■ Code Block Coverage (12.2 and higher)

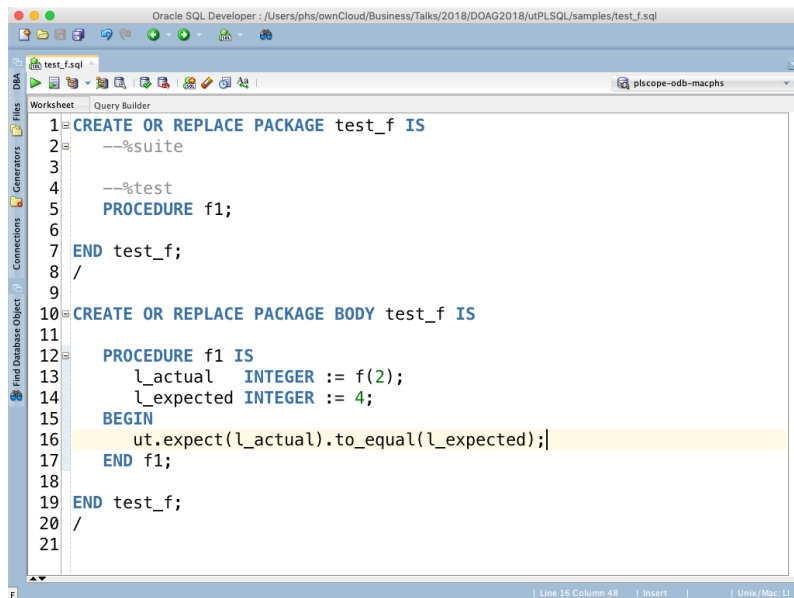
```
CREATE OR REPLACE FUNCTION f(a IN INTEGER) RETURN INTEGER IS
BEGIN
    IF a IS NULL THEN RETURN 0; ELSE RETURN a*a; END IF;
END f;
/
```

Two test cases for
100% coverage

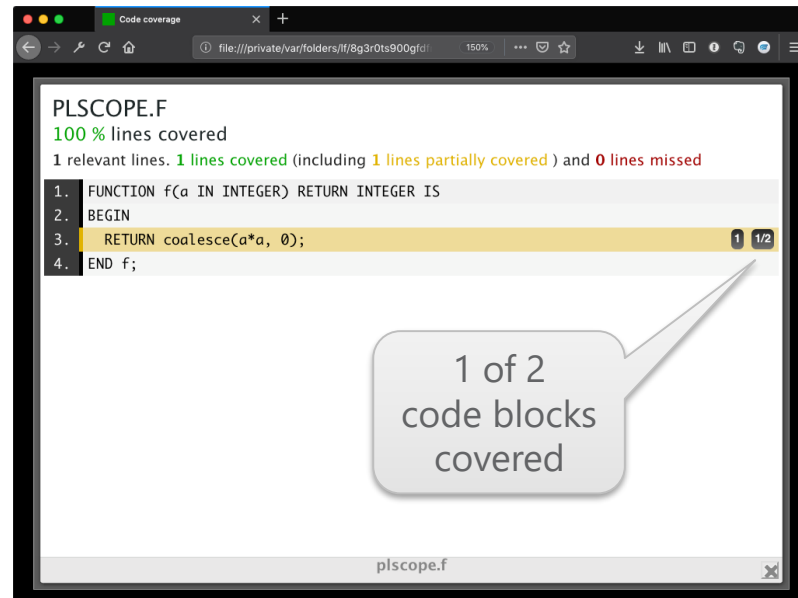
```
CREATE OR REPLACE FUNCTION f(a IN INTEGER) RETURN INTEGER IS
BEGIN
    RETURN coalesce(a*a, 0);
END f;
/
```

One test case for
100% coverage
when passing NULL

utPLSQL Combines Line & Code Block Coverage



```
1=CREATE OR REPLACE PACKAGE test_f IS
2  --%suite
3
4  --%test
5  PROCEDURE f1;
6
7 END test_f;
8 /
9
10=CREATE OR REPLACE PACKAGE BODY test_f IS
11
12  PROCEDURE f1 IS
13    l_actual  INTEGER := f(2);
14    l_expected INTEGER := 4;
15  BEGIN
16    ut.expect(l_actual).to_equal(l_expected);
17  END f1;
18
19 END test_f;
20 /
21
```



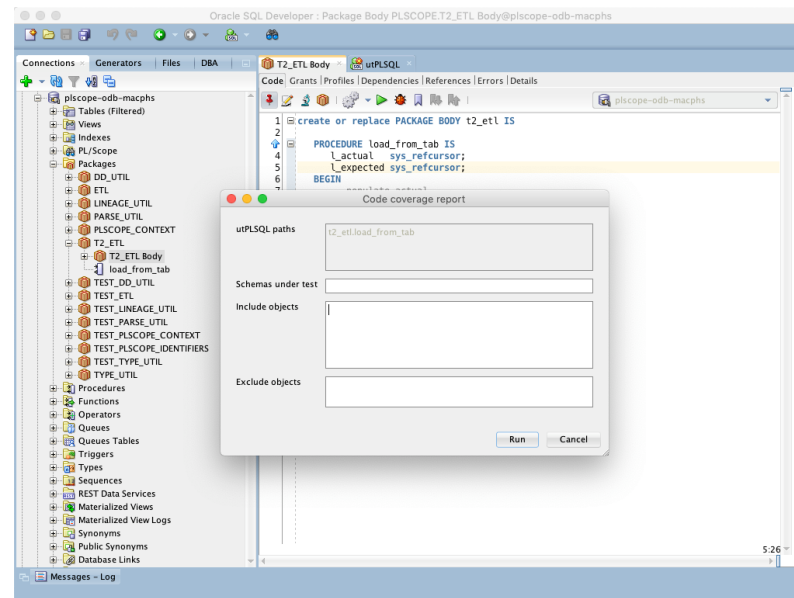
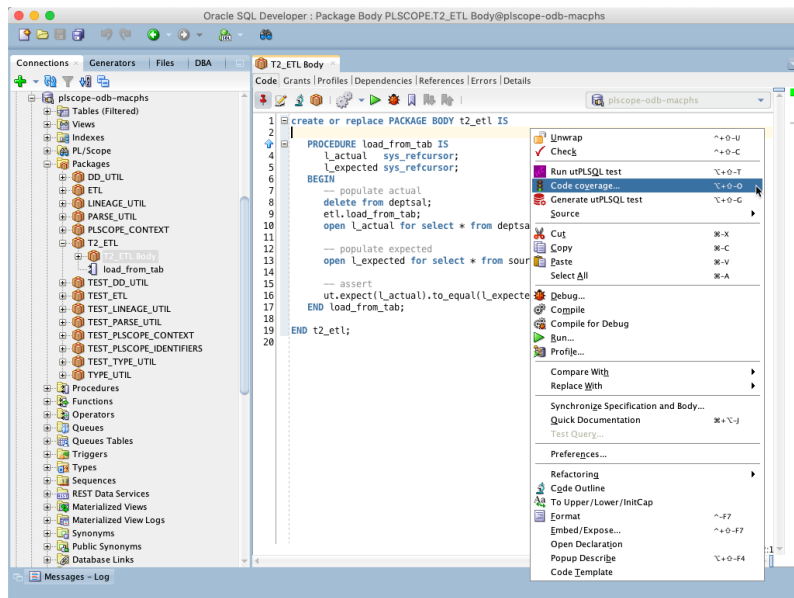
PLSCOPE.F
100 % lines covered
1 relevant lines. 1 lines covered (including 1 lines partially covered) and 0 lines missed

```
1. FUNCTION f(a IN INTEGER) RETURN INTEGER IS
2. BEGIN
3. RETURN coalesce(a*a, 0);
4. END f;
```

1 of 2
code blocks
covered



Run Code Coverage Report





Code Coverage Report



Code coverage

file:///private/var/folders/ff/8g3r0ts900gdfn2xxkn9yz0000...

All files (.77%) Generated less than a minute ago

All files (.77% lines covered at 0 hits/line)

6 files in total.
913 relevant lines. 7 lines covered and 906 lines missed.

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
plscope.dd_util	0 %	147	147	0	147	0
plscope.lineage_util	0 %	358	358	0	358	0
plscope.parse_util	0 %	216	216	0	216	0
plscope.plscope_context	0 %	43	43	0	43	0
plscope.type_util	0 %	96	96	0	96	0
plscope.etl	13.21 %	147	53	7	46	0

Showing 1 to 6 of 6 entries

Generated by utPLSQL v3.1.3-2292 -dev
Based on simplecov.html v0.10.0

Code coverage

file:///private/var/folders/ff/8g3r0ts900gdfn2xxkn9yz0000...

PLSCOPE.ETL
13.21 % lines covered
53 relevant lines. 7 lines covered and 46 lines missed

```
1. PACKAGE BODY etl AS
2.   g_unused_column v$mystat.sid%TYPE;
3.
4. PROCEDURE clear_deptsal IS
5. BEGIN
6.   DELETE FROM deptsal;
7.   DELETE FROM deptsal_err;
8.   sys.dbms_output.put_line('deptsal an deptsal_err deleted.'); -- use synonym
9. END clear_deptsal;
10.
11. PROCEDURE load_from_tab IS
12. BEGIN
13.   clear_deptsal;
14.   INSERT INTO deptsal (dept_no, dept_name, salary)
15.   SELECT /*+ordered */ d.deptno, d.dname, SUM(e.sal + NVL(e.comm, 0)) AS sal
16.   FROM dept d
17.   LEFT JOIN (SELECT * FROM emp WHERE hiredate > DATE '1980-01-01') e
18.   ON e.deptno = d.deptno
19.   GROUP BY d.deptno, d.dname;
20.   COMMIT;
21.   sys.dbms_output.put_line('deptsal loaded and committed (from table).');
22. END load_from_tab;
23.
```

plscope.etl

Core Messages

■ The First Step Is the Hardest

- Set up a test-friendly environment
 - Install utPLSQL core testing framework
 - Install SQL Developer for utPLSQL
- Start with tests
 - to reproduce bugs
 - for new requirements



Trivadis @ DOAG 2018

#opencompany

- Booth: 3rd floor – next to the escalator
- We share our know how!
Simply drop by, live presentations
and documents archive
- T-Shirts, contest and much more
- We look forward to your visit

