

Testing with utPLSQL

Made Easy with SQL Developer

Philipp



@phsalvisberg



<https://www.salvis.com/blog>

BASEL | BERN | BRUGG | BUKAREST | DÜSSELDORF | FRANKFURT A.M. | FREIBURG I.B.R. | GENÈVE
HAMBURG | KOPENHAGEN | LAUSANNE | MANNHEIM | MÜNCHEN | STUTTGART | WIEN | ZÜRICH

trivadis

Philipp

- Database centric development
- Model Driven Software Development
- Author of free SQL Developer Extensions
PL/SQL Unwrapper, PL/SQL Cop,
utPLSQL, plscope-utils, oddgen and
Bitemp Remodeler



 @phsalvisberg

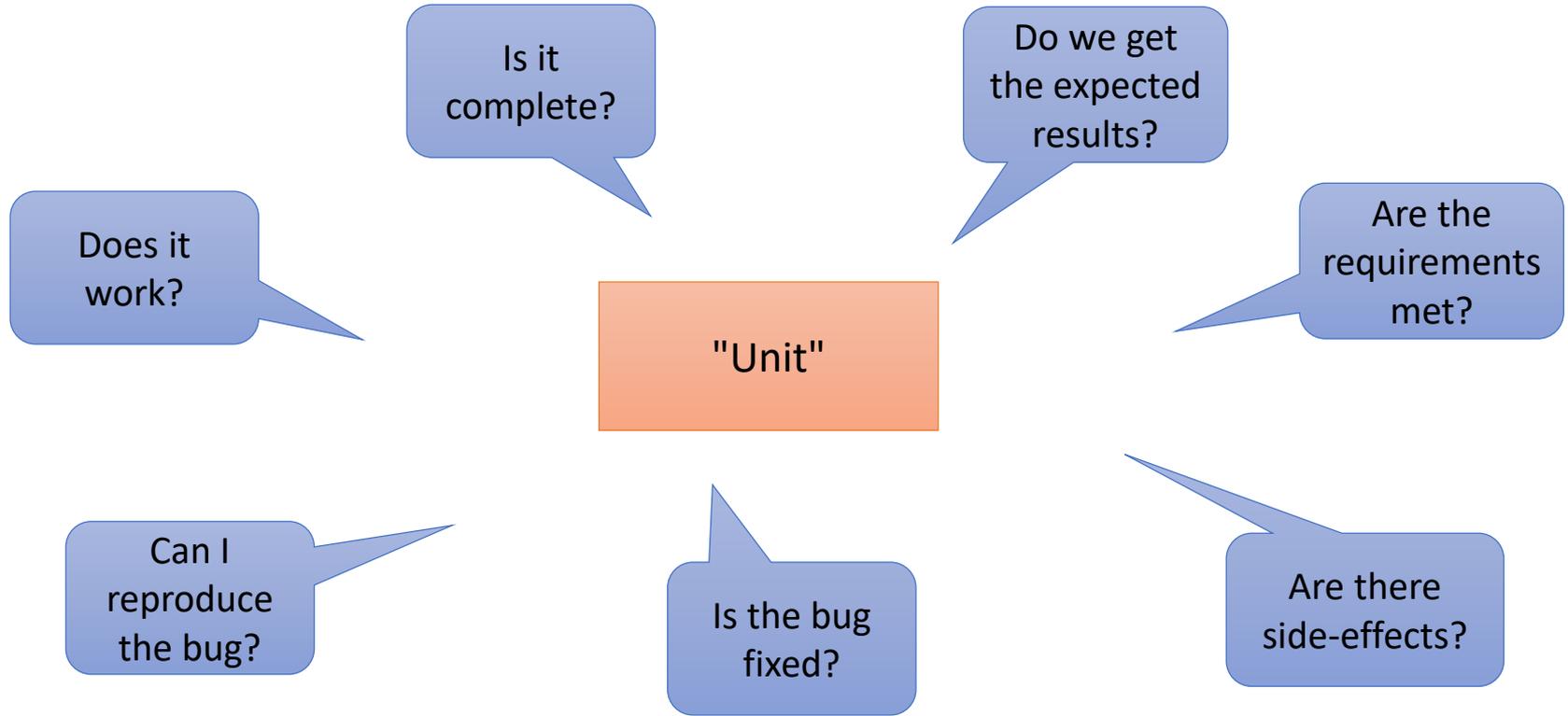
 <https://www.salvis.com/blog>

Agenda

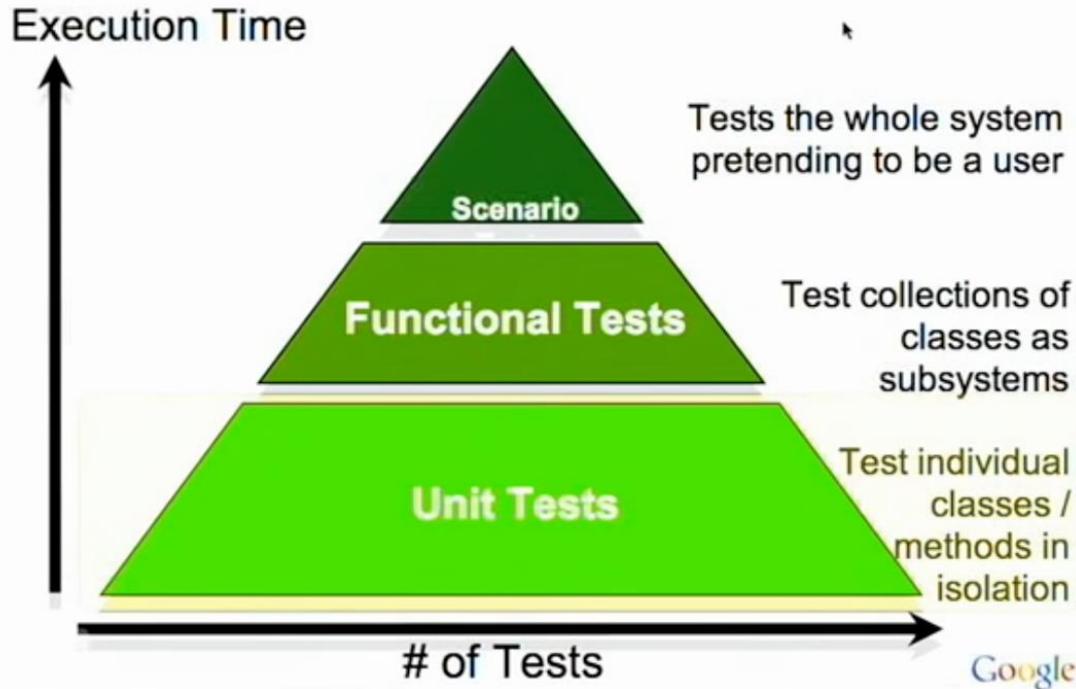
1. Introduction
2. Installation
3. Build & Run Tests in SQL Developer
4. Run Code Coverage Reports in SQL Developer
5. Core Messages

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>

utPLSQL

- GUI
- API
- Integration
- Components
- Unit

utPLSQL Units Under Test

Primary

- Types
- Packages
- Procedures
- Functions



Secondary

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



utPLSQL Suite – 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>
--%tags(<tag>[,...]
--%throws(<exception>[,...])
--%beforeall
--%afterall
--%beforeeach
--%aftereach
--%beforetest([...])
--%aftertest([...])
--%rollback(manual)
--%disabled

--%suite(<description>
--%suitepath(<path>
--%tags(<tag>[,...]
--%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, JSON, 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(l_actual).to_equal(l_expected);

Finished in .007015 seconds

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

Installation

Install utPLSQL Core Testing Framework

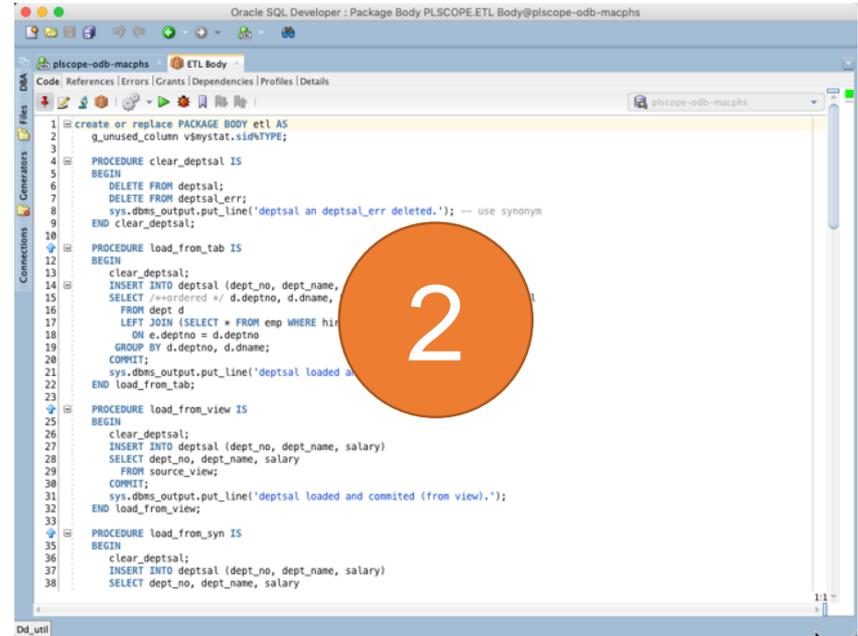
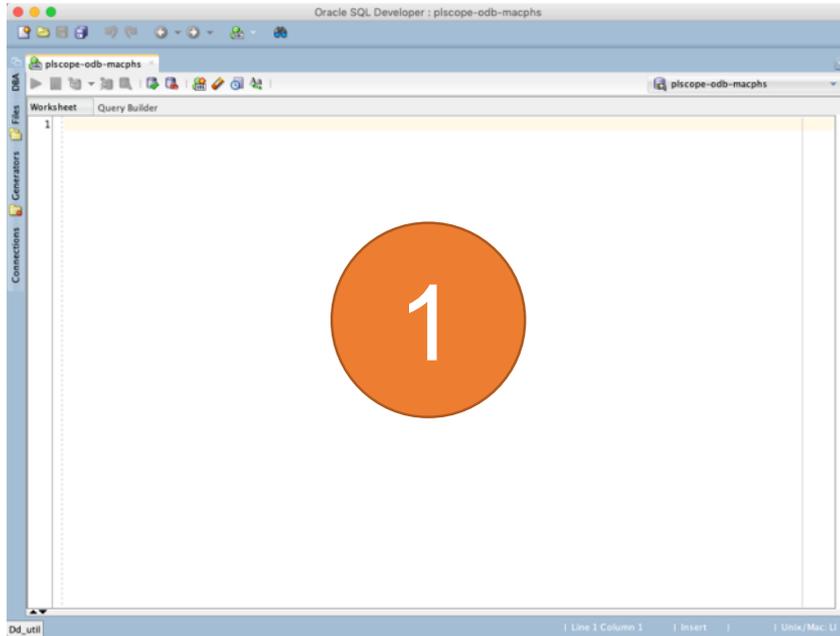
- Download utPLSQL.zip from <https://github.com/utPLSQL/utPLSQL/releases>
- Unzip utPLSQL.zip
- cd source
- sqlplus / as sysdba @install_headless.sql
 - User UT3
 - Password XNtxj8eEgA6X6b6f
 - Tablespace USERS

Install utPLSQL for SQL Developer

- Download utplsql_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 "utplsql_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

Oracle SQL Developer : plscope@odb-macphs

Worksheet

```
1 ut_
2 -- test {procedure_name} case 1: ... -- PROCEDURE {procedure_n...
3 --<context1[procedure_name] -->test PROCEDURE {procedure_name}1...
4 CREATE OR REPLACE PACKAGE BODY test_{package_name} IS -- ...
5 CREATE OR REPLACE PACKAGE test_{package_name} IS --%suite
6
7 SYS.ut_call_stack
8 SYS.ut_coll
9 SYS.ut_compress
10 SYS.ut_encode
11 SYS.ut_file
12 SYS.ut_gdk
13 SYS.ut_http
14 SYS.ut_isbn
15 SYS.ut_ident
16 SYS.ut_inaddr
17 SYS.ut_las
18 SYS.ut_match
19 SYS.ut_nla
20 SYS.ut_raw
21 SYS.ut_recomp
22 SYS.ut_ref
23 SYS.ut_sntp
24 SYS.ut_sys_compress
25 SYS.ut_tcp
26 SYS.ut_url
27 SYS.ut_xml
28
29 UT3.ut_annotation_cache_manager
30 UT3.ut_annotation_manager
31 UT3.ut_annotation_parser
32 UT3.ut_ansiconsole_helper
33 UT3.ut_compound_data_helper
```

CREATE OR REPLACE PACKAGE test_{package_name} IS --%suite
--%suitepath(alltests) --<context1[procedure_name]>
--%test PROCEDURE {procedure_name}1; --%test
PROCEDURE {procedure_name}2; --%endcontext END test_{
{package_name}; /

Oracle SQL Developer : plscope@odb-macphs

Worksheet

```
1 CREATE OR REPLACE PACKAGE t2_{package_name} IS
2
3 --%suite
4 --%suitepath(alltests)
5
6 --%test
7 PROCEDURE procedure_name;
8
9 END t2_{package_name};
10 /
11
```



Test First – Complete Test & Run

Oracle SQL Developer : plscope-odb-macphs-4

Worksheet Query Builder

```
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   -- test
16
17
18   PROCEDURE load_from_tab IS
19     l_actual INTEGER;
20     l_expected INTEGER := 4;
21
22   BEGIN
23     -- populate actual
24     delete from deptsal;
25     etl.load_from_tab;
26     select count(*) into l_actual from deptsal;
27
28     -- assert
29     ut.expect(l_actual).to_equal(l_expected);
30   END load_from_tab;
31
32 END t2_etl;
33 /
```

Run utPLSQL test

Oracle SQL Developer

Connections utPLSQL Snippets Generators

plscope-odb-macphs-4

23.04.25 (23plscope-odb-macphs) 0.097 seconds

Finished. 15 ms

Tests: 1/1 Failures: 0 Errors: 0 Disabled: 0 Warnings: 0 Info: 1

Description (alltests.t2_etl)	Time
load_from_tab	5 ms

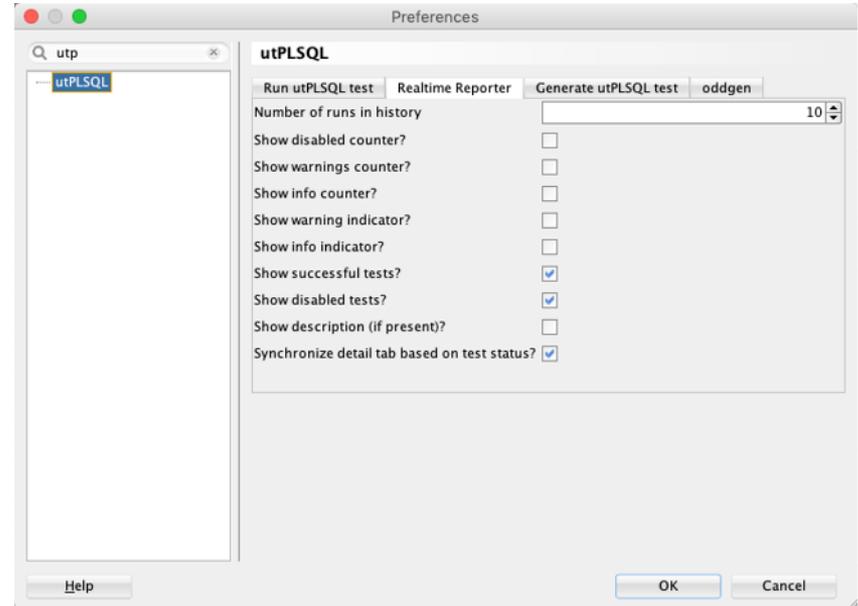
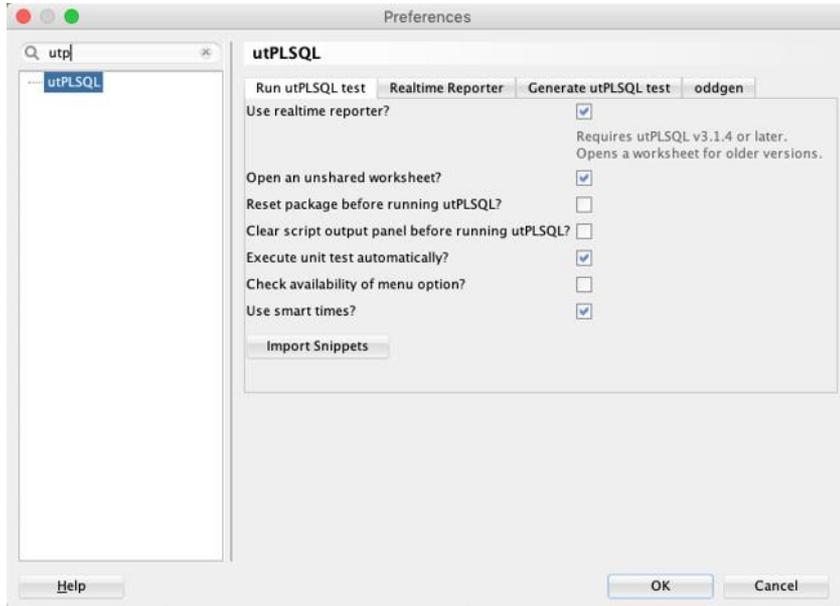
Test Failures Errors Warnings Info

deptsal an deptsal_err deleted.
deptsal loaded and committed (from table).

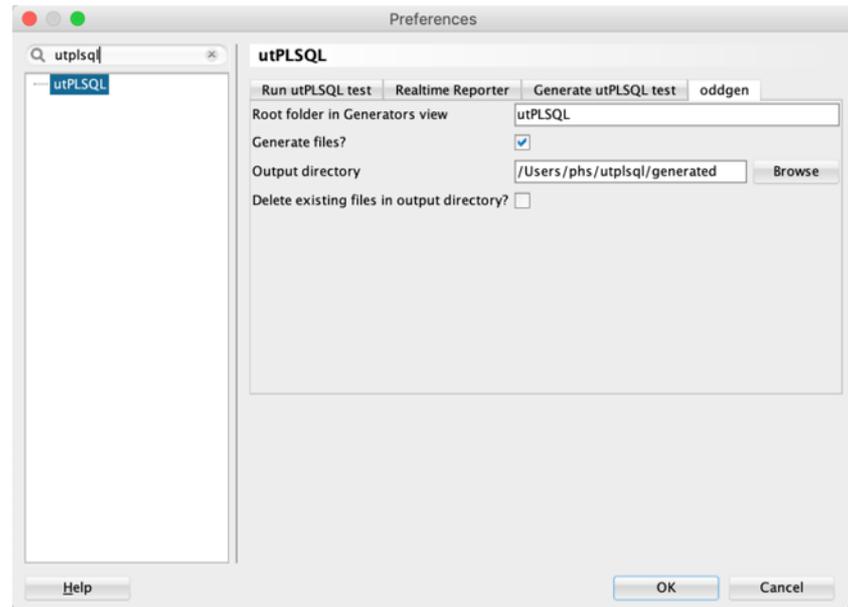
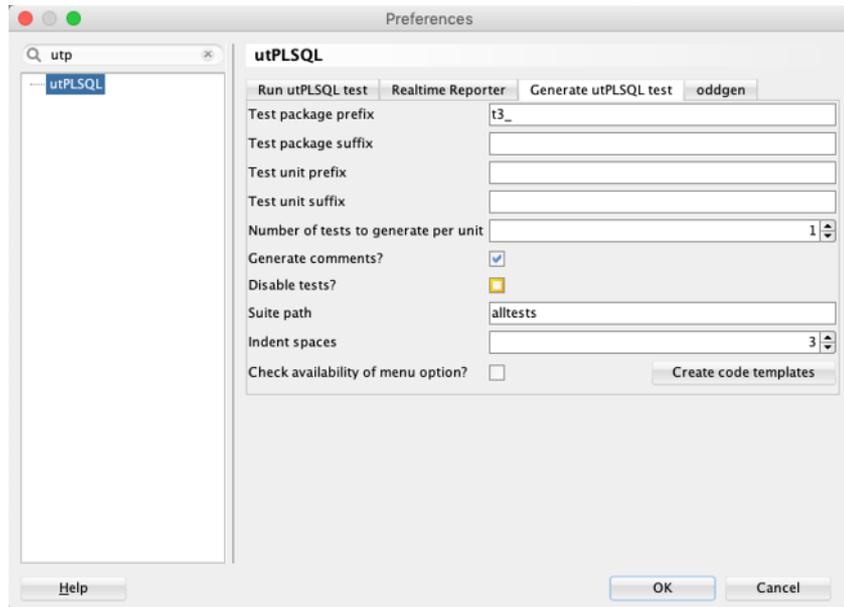
Worksheet Query Builder

```
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   -- test
16
17
18   PROCEDURE load_from_tab IS
19     l_actual INTEGER;
20     l_expected INTEGER := 4;
21
22   BEGIN
23     -- populate actual
24     delete from deptsal;
25     etl.load_from_tab;
26     select count(*) into l_actual from deptsal;
27
28     -- assert
29     ut.expect(l_actual).to_equal(l_expected);
30   END load_from_tab;
31
32 END t2_etl;
33 /
```

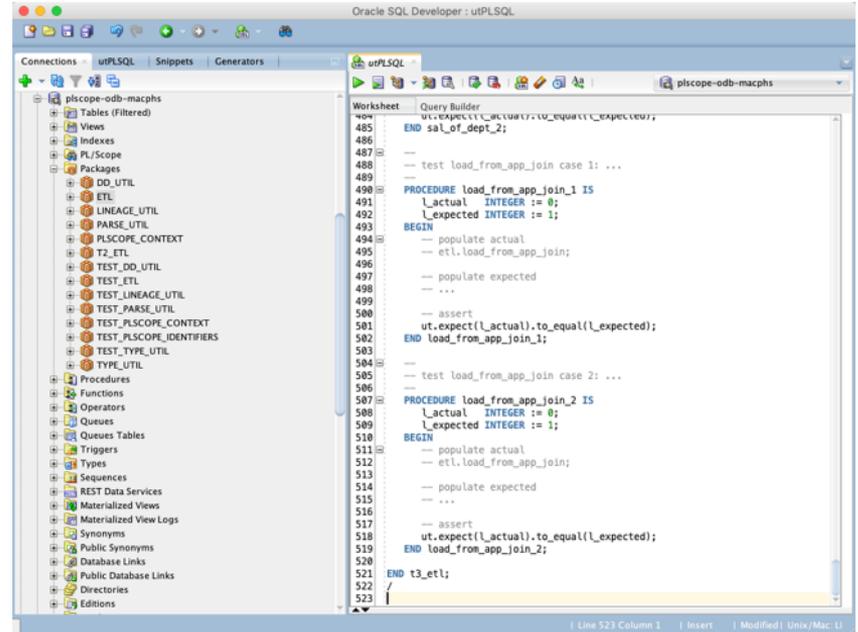
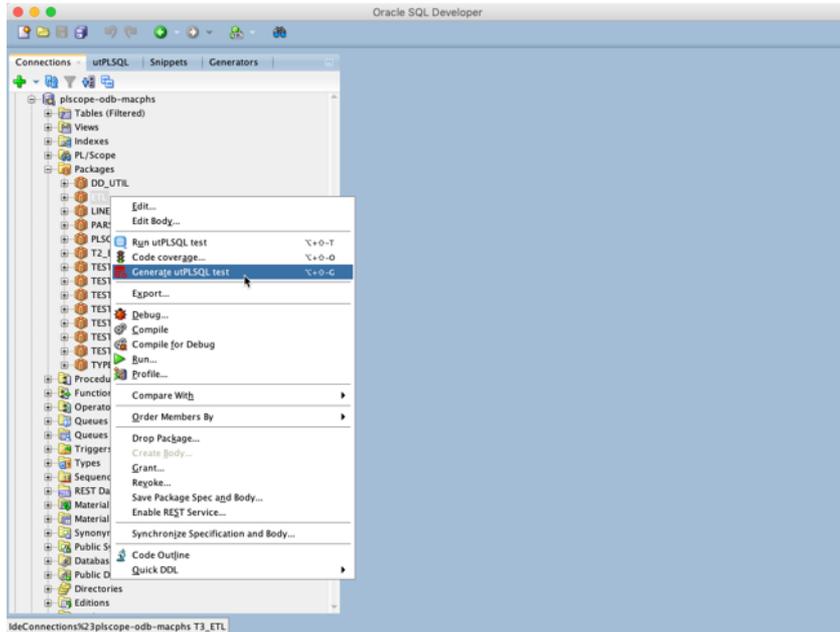
Configuration – Running utPLSQL Tests



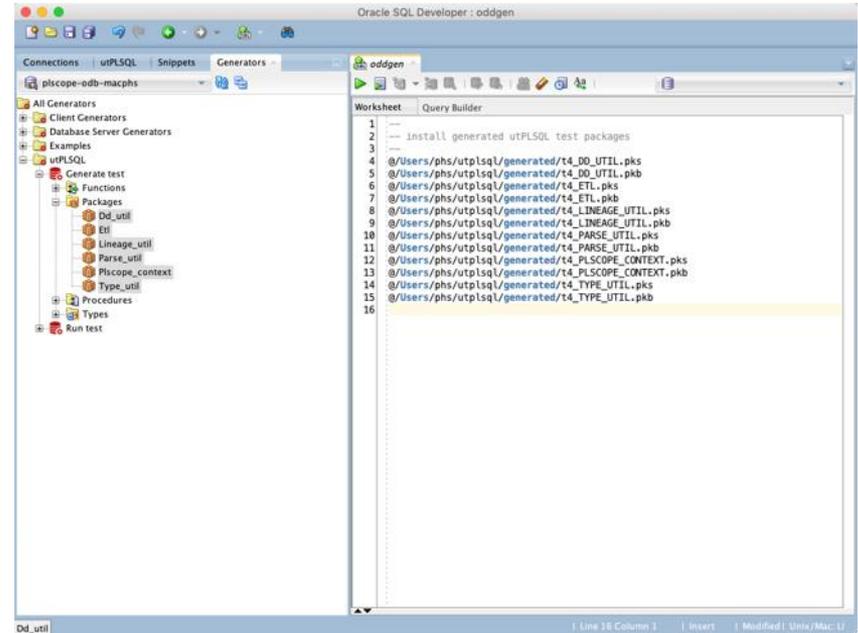
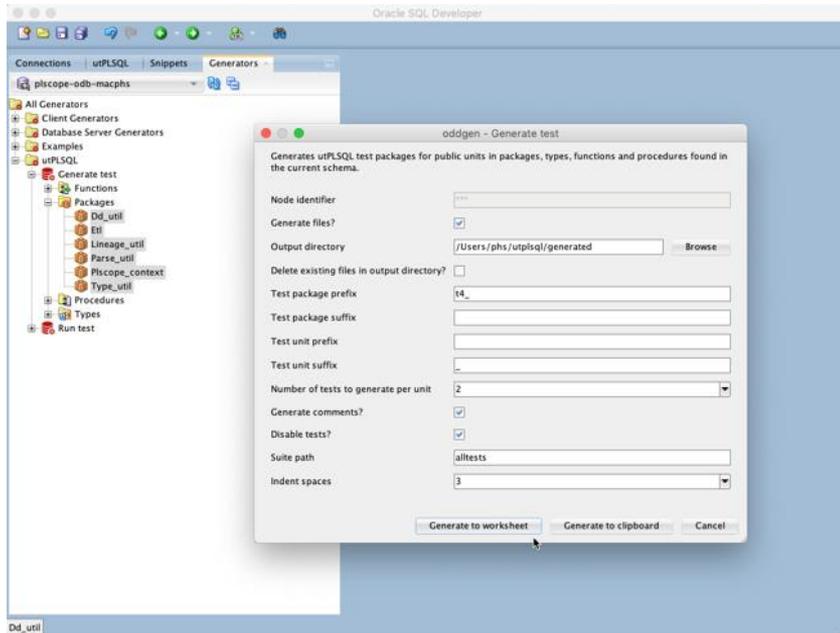
Configuration – Generating utPLSQL Tests



Test Last – Create Test from Existing Code



Test Last – Generate Multiple Test Skeletons



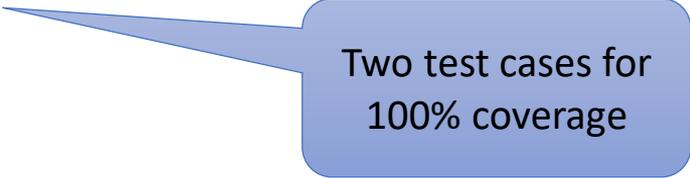
Run Code Coverage Reports in SQL Developer

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)

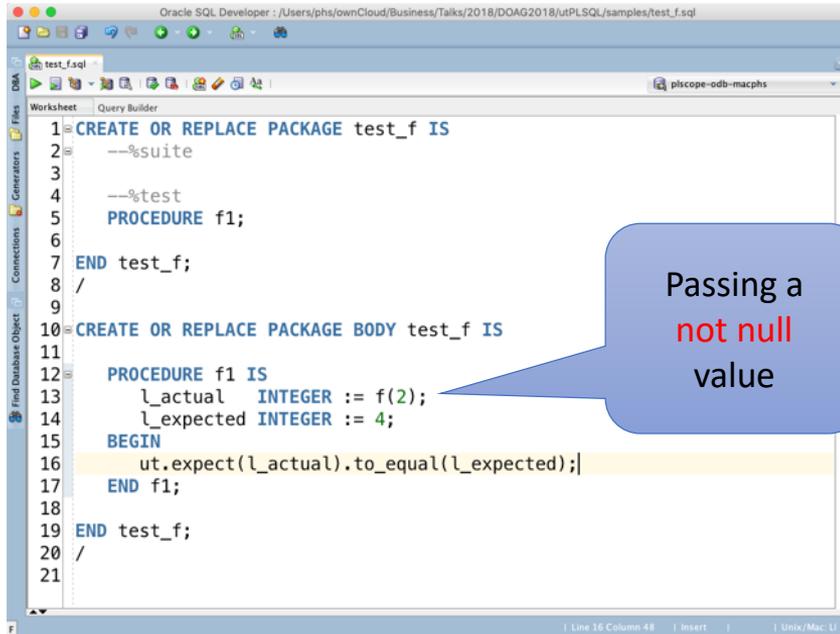
```
CREATE OR REPLACE FUNCTION f(a IN INTEGER) RETURN INTEGER IS
BEGIN
    IF a IS NULL THEN RETURN 0; ELSE RETURN a*a; END IF;
END f;
/
```

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;
/
```

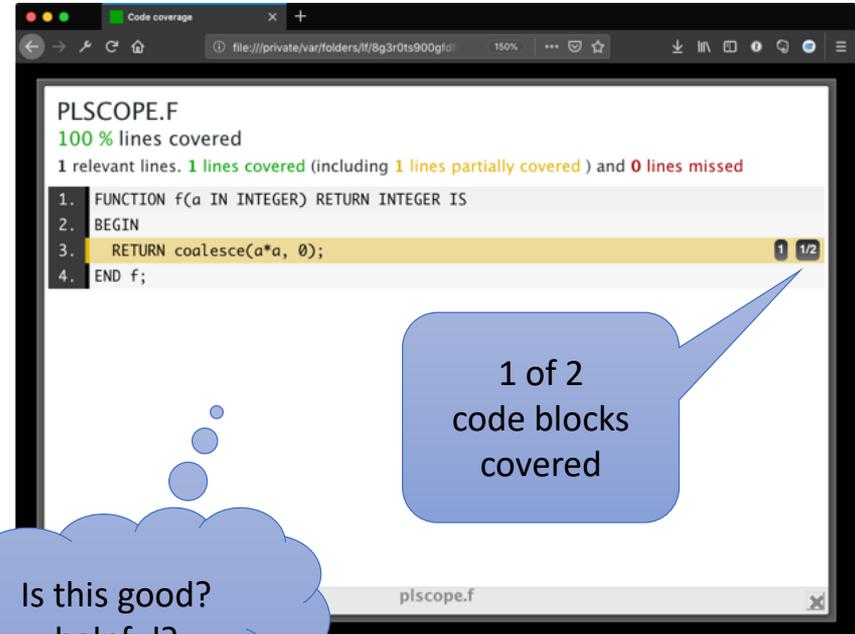
One test case for
100% coverage
when passing NULL

utPLSQL – 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
```

Passing a
not null
value



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

Is this good?
helpful?

Run Code Coverage Report

Oracle SQL Developer : Package Body PLSCOPE.T2_ETL.Body@plscope-odb-macchs

Code Profiles | Details | Dependencies | References | Grants | Errors

```
1 create or replace PACKAGE BODY t2_etl IS
2
3 -- test case: count rows
4
5
6 PROCEDURE load_from_tab_1 IS
7   l_actual INTEGER;
8   l_expected INTEGER := 4;
9 BEGIN
10  -- populate actual
11  delete from deptsal;
12  etl.load_from_tab;
13  select count(*) into l_actual
14  from deptsal;
15  -- assert
16  ut.expect(l_actual).to_equal(l_expected);
17 END load_from_tab_1;
18
19 -- test case: compare all rows
20
21
22 PROCEDURE load_from_tab_2 IS
23   l_actual sys_refcursor;
24   l_expected sys_refcursor;
25 BEGIN
26  -- populate actual
27  delete from deptsal;
28  etl.load_from_tab;
29  open l_actual for select * from deptsal;
30  -- populate expected
31  open l_expected for select * from source_view;
32  -- assert
33  ut.expect(l_actual).to_equal(l_expected);
34 END load_from_tab_2;
35
36 END t2_etl;
37
38 PACKAGE BODY t2_etl
```

Oracle SQL Developer : Package Body PLSCOPE.T2_ETL.Body@plscope-odb-macchs

Code Profiles | Details | Dependencies | References | Grants | Errors

```
1 create or replace PACKAGE BODY t2_etl IS
2
3 -- test case: count rows
4
5
6 PROCEDURE load_from_tab_1 IS
7   l_actual INTEGER;
8   l_expected INTEGER := 4;
9 BEGIN
10  -- populate actual
11  delete from deptsal;
12  etl.load_from_tab;
13  select count(*) into l_actual
14  from deptsal;
15  -- assert
16  ut.expect(l_actual).to_equal(l_expected);
17 END load_from_tab_1;
18
19 -- test case: compare all rows
20
21
22 PROCEDURE load_from_tab_2 IS
23   l_actual sys_refcursor;
24   l_expected sys_refcursor;
25 BEGIN
26  -- populate actual
27  delete from deptsal;
28  etl.load_from_tab;
29  open l_actual for select * from deptsal;
30  -- populate expected
31  open l_expected for select * from source_view;
32  -- assert
33  ut.expect(l_actual).to_equal(l_expected).unordered;
34 END load_from_tab_2;
35
36 END t2_etl;
37
38 PACKAGE BODY t2_etl
```

Code coverage report

utPLSQL paths: t2_etl

Schemas under test:

Include objects: UT3_LATEST_RELEASE.UT_EXPECTATION_COMPOUND, PLSCOPE.ETL_PLSCOPE.T2_ETL

Exclude objects:

Run Cancel



Code Coverage Report

Code coverage

file:///private/var/folders/ff/8g3r0ts900gdfn2xxkn9yz00000gn/T/utpl...

Apple LEO TVD PortfolioManagement TvdEmp Router

All files (13.21%) Generated less than a minute ago

All files (13.21% lines covered at 0 hits/line)

1 files in total.
53 relevant lines. **7 lines covered** and **46 lines missed**.

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
plscope.etl	13.21 %	147	53	7	46	0

Showing 1 to 1 of 1 entries

Generated by @PLSCOPE_V3.1.8-3189-@trivadis
Based on simulecor-hana v0.10.0

Code coverage

file:///private/var/folders/ff/8g3r0ts900gdfn2xxkn9yz00000gn/T/utpl...

Apple LEO TVD PortfolioManagement TvdEmp Router

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 on deptsal_err deleted.');
```

Showing 1 to 22 of 22 entries

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
- **utPLSQL will change how you code**
 - Write smaller units
 - Isolate code that is difficult to test





Making a **WORLD** possible
in which **intelligent IT**
facilitates **LIFE and WORK** as a
matter of course.