Testing with utPLSQL

Made Easy with SQL Developer

Philipp

@phsalvisberg https://www.salvis.com/blog

23.02.2021

Introduction

Test Automation

"The use of software
separate from the software being tested
to control the execution of tests
and the comparison of
actual outcomes with predicted outcomes."

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

Testing Scope in Database Development

Every component of an application
that is deployed in the database.

For example:
Types, packages, procedures, functions,
views, triggers, constraints.

utPLSQL

Core Testing Framework

- Schema in the database
- No repository
- Annotation based tests

ORACLE DATABASE

Development

- Realtime Reporter
- Code Coverage, Code Templates, etc.




Test Automation

- Command Line Client
- Maven Plugin Maven
- Various Reporters

JUnit  TFS

sonarqube COVERALLS 





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

trivadis

Test Implementation

```
CREATE OR REPLACE PACKAGE BODY test_package_name AS
   PROCEDURE procedure_name IS
      l_actual   SYS_REFCURSOR;
      l_expected SYS_REFCURSOR;
   BEGIN
      open l_actual for select * from dept where deptno != 30;
      open l_expected for select * from dept;
      ut.expect(l_actual).to_equal(l_expected).join_by('DEPTNO');
   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, contain, **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')
```

List of ...
- [schema.]package[.procedure]
- [schema]:suitepath[.context][.procedure]

List of ...
- tag (include)
- -tag (exclude)

```
test_package_name
  procedure_name [.03 sec] (FAILED - 1)

Failures:

  1) procedure_name
       Actual: refcursor [ count = 3 ] was expected to equal: refcursor [ count = 4 ]
       Diff:
       Rows: [ 1 differences ]
         PK <DEPTNO>30</DEPTNO> - Missing:  <DEPTNO>30</DEPTNO><DNAME>SALES</DNAME><LOC>CHICAGO</LOC>
       at "SCOTT.TEST_PACKAGE_NAME.PROCEDURE_NAME", line 8 ut.expect(l_actual).to_equal(l_expected)
                                                                              .join_by('DEPTNO');

Finished in .034792 seconds
1 tests, 1 failed, 0 errored, 0 disabled, 0 warning(s)
```

trivadis

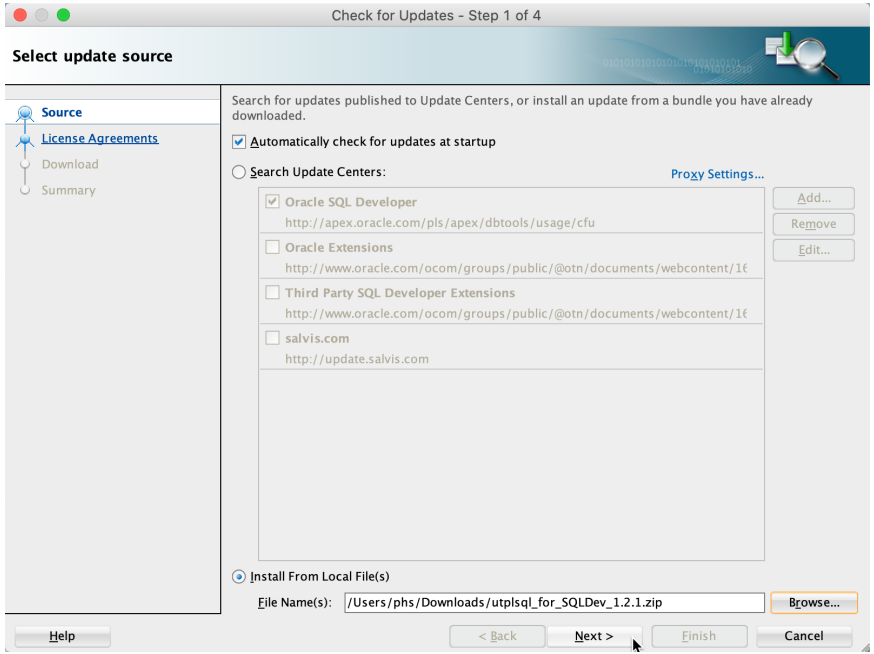
Installation

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 from File *)

- 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

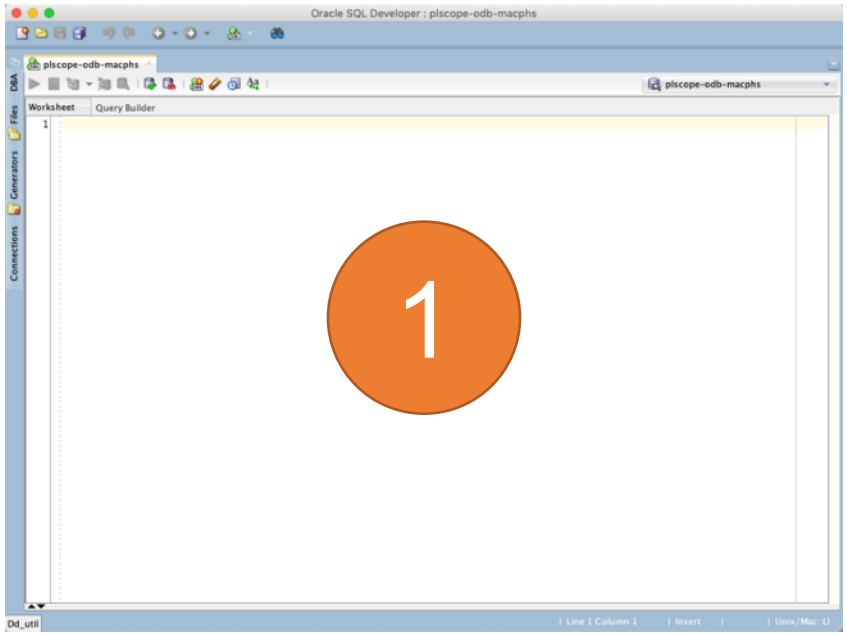


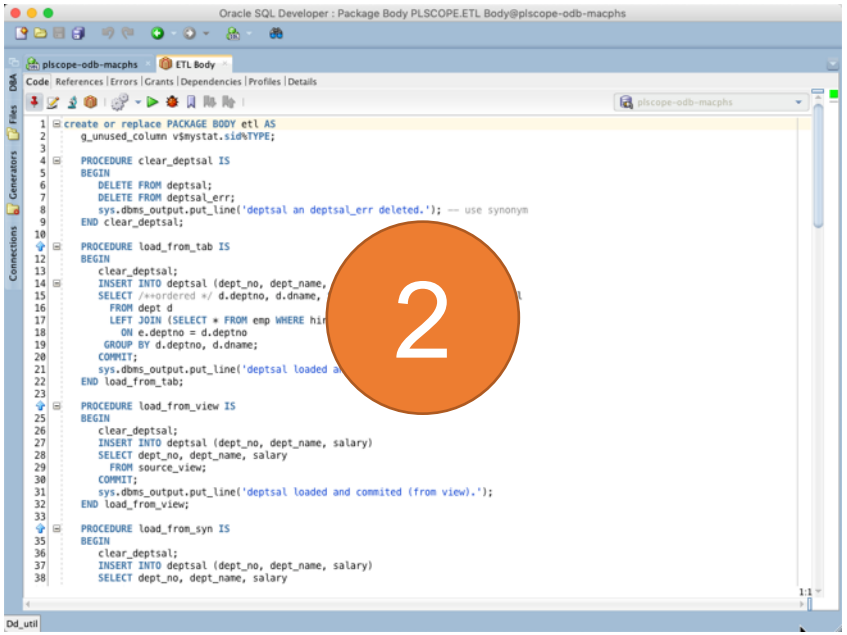
*) You can also configure an Update Center, see https://github.com/PhilippSalvisberg/sqldev-update

Build & Run Tests in SQL Developer

trivadis

Starting Point?





trivadis

Run Code Coverage Reports in SQL Developer

trivadis

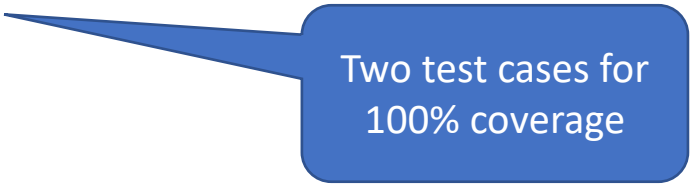
Code Coverage – Defintion

"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;
/
```



Code Block Coverage (12.2 and higher)

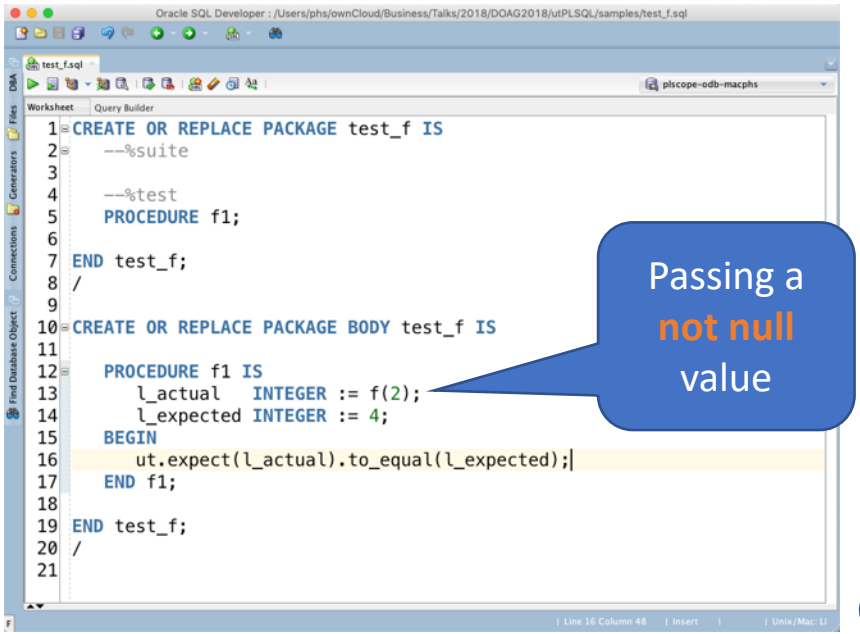
```
CREATE OR REPLACE FUNCTION f(a IN INTEGER) RETURN INTEGER IS
BEGIN
   IF a IS NULL THEN RETURN 0; ELSE RETURN a*a; END IF;
END f;
/
```

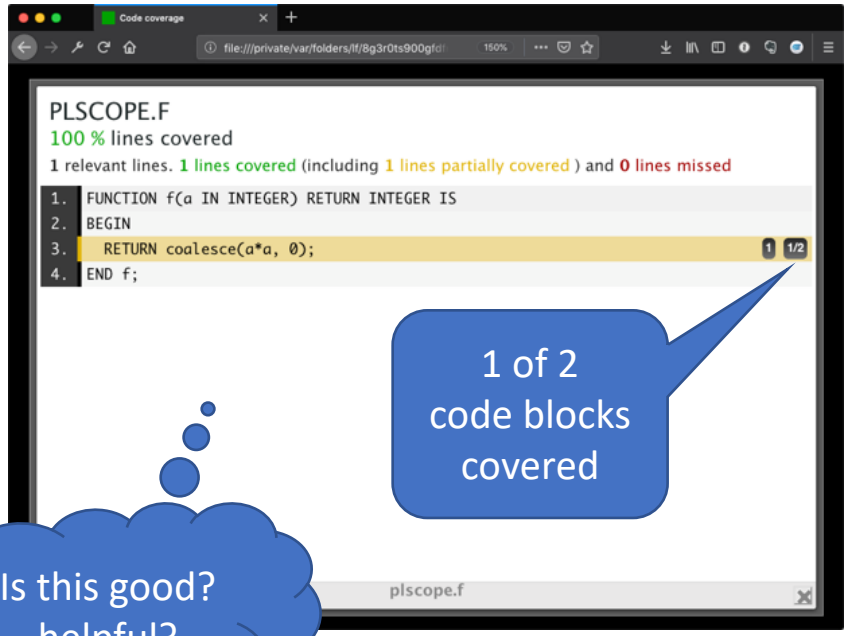
Two test cases for 100% coverage

```
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





Is this good? helpful?

trivadis

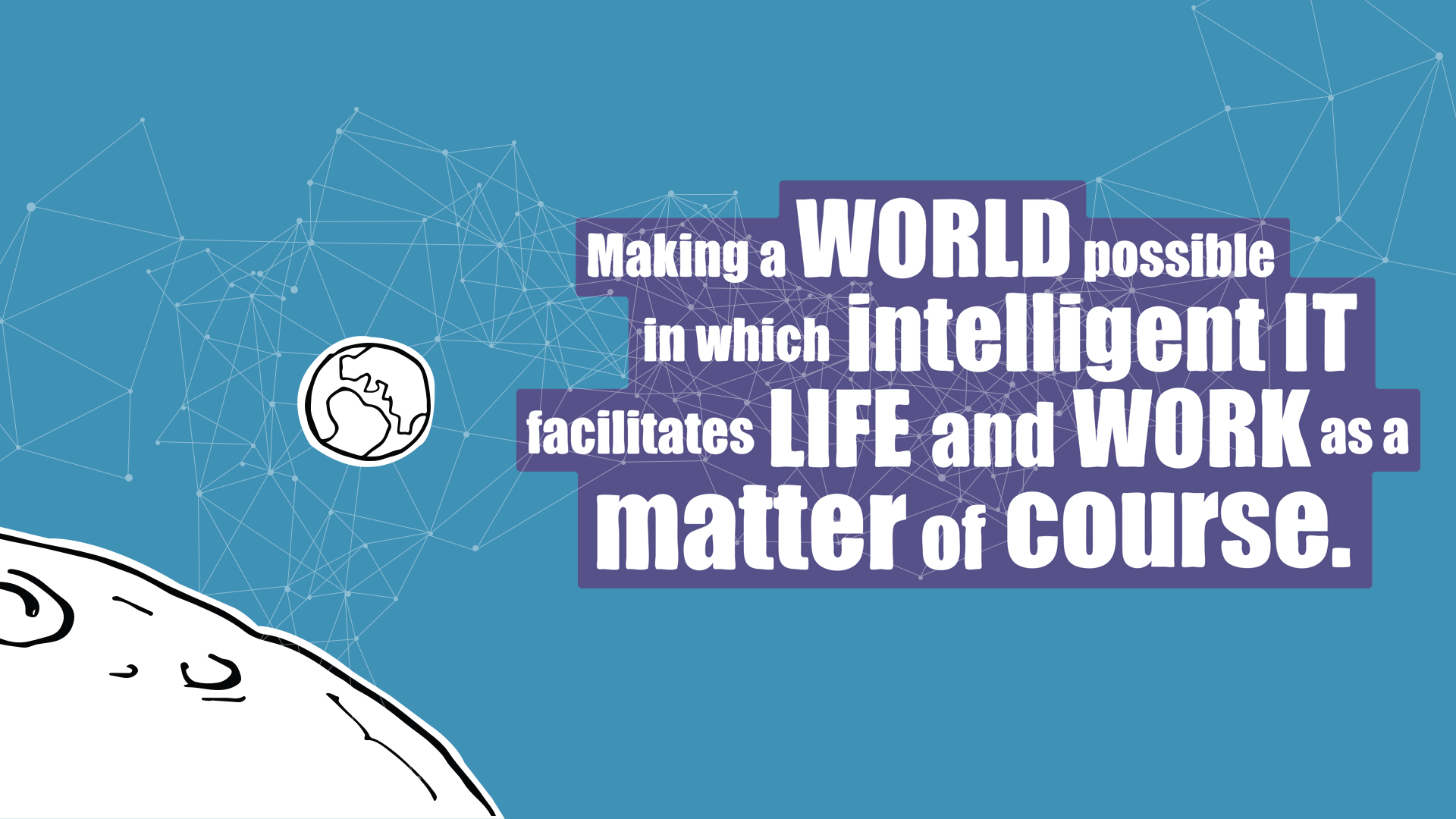
Core Messages

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.