utPLSQL 3.1.10 Cheat Sheet

by Samuel Nitsche (pesse) via cheatography.com/125418/cs/24133/

Define Test Suites with Annotations

Annotations are single-line comments and belong in the package header. Put a newline between package and procedure annotations.

Package Annotations

--%suite(description)	Package is a test-suite (*optional* description). *Mandatory*
--%suitepath(org.utplsql)	Groups suites in hierarchical namespaces
--%rollback(auto|manual)	Automatic (*default*) / manual transaction control
--%context(description) --%endcontext	Starts/ends sub-suite in suite. Can be nested.
--%disabled	Tests of suite/context won't be executed
--%beforeall(procedure [, ...]) --%afterall...	Procedure(s) to run before/after all tests in the suite/context
--%beforeeach(procedure [, ...]) --%aftereach...	Procedure(s) to run before/after each test in the suite/context
--%tags(myTag[, ...)	Label a test/context/suite

Procedure Annotations

--%test(description)	Procedure is a test (with optional description)
--%disabled	Test won't be executed
--%throws(exception [, ...])	Test expects exception(s) to be thrown
--%beforeall, --%afterall --%beforeeach, --%aftereach	Procedure to run before/after all/each tests in the suite/context
--%beforetest(procedure [, ...]) --%aftertest(procedure [, ...])	Procedure(s) to be run before/after annotated test
--%tags(myTag[, ...)	Label test

Run Tests

`exec ut.run();`	Run all tests to the default reporter (DBMS_OUTPUT)
`select * from table( ut.run());`	Run all tests and display output as result set

Consider using utplsql-cli to easily run tests

Run specific tests

`ut.run ('m y_s uit e_p c kg')`	Run only the specified suite by name (current schema)
`ut.run ('HR')`	Run all tests in specific schema
`ut.run('my_suite.test1, my_suite.test2')`	Run list of items
`ut.run(':my.suite.path')`	Run specific suite path

Run Parameters

`a_force_manual_rollback`	True/False. Run specified test and don't rollback changes (leaves open transaction)
`a_rand om_ tes t_order`	True/False. Run tests in random order
`a_colo r_c onsole`	True/False. Color-code output (for Windows see ANSICON)
`a_clie nt _ c ha rac - te r_set`	Define report's character-set
`a_tags`	Run tests with any specified tags. Use `a_tags =>' -tag1'` to exclude a tag

Reporting: Run alternative Reporter

```
exec ut.run (ut _ju nit _re por ter ());
```

Reporters

UT_DOCUMENTATION_REPORTER	Textual pretty-print of test results. *Default*
UT_COV ERA GE_ HTM L_R EP ORTER	HTML Coverage Report, includes SourceCode
UT_JUN IT_ REP ORTER	Test results conforming to JUnit 4 and above
UT_REA LTI ME_ REP ORTER	Test results and additional information as XML. Allows to show progress information
UT_SON AR_ TES T_R EPORTE R	JSON Test results designed for SonarQube

See full list of built-in reporters. See documentation for coverage and reporters.

By **Samuel Nitsche** (pesse)
cheatography.com/pesse/
cleandatabase.wordpress.com

Published 4th September, 2020.
Last updated 4th September, 2020.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Cheatography

utPLSQL 3.1.10 Cheat Sheet

by Samuel Nitsche (pesse) via cheatography.com/125418/cs/24133/

Expectation Syntax

Base expectation

```
ut.ex  pect( actua  l_v  alue ).to_( matc her );
```

Negated expectation

```
ut.ex  pect( actua  l_v  alue ).not_to_( matc her
);
```

Shortcut syntax (recommended)

```
ut.ex  pect( actua  l_v  alue ).to_matcher;
ut.expect( actua  l_v  alue ).not_  to_ m atc her;
```

Provide additional expectation description

```
ut.exp ect(1, 'Add i tional text' ).t o _ e qu  al(
3);
```

Simple Matchers

```
ut.exp ect (1= 1).to_be_true();
ut.exp ect(1 is null).to_be_false();
ut.exp ect (nu ll).to_be_null();
ut.exp ect (to _cl ob( 'AB C') ).to_be_not _null();
ut.ex  pect( 3 ).to_b e_b  etween( 1, 3 );
ut.ex  pect( 3 ).to_b e_g  rea  te r _ or  _equal(
2 );
ut.ex  pect( 2 ).to_b e_g  rea  te r _than( 1 );
ut.ex  pect( 3 ).to_b e_l  ess  _o r _ equal( 3 );
ut.ex  pect( 3 ).to_b e_l  ess  _than( 4 );
```

Equality Matchers

```
ut.ex  pect( 'a dog' ).to_equal('a dog',
  a_null s_a re_ equal => false );
```

a_nulls_are_equal is **true** by default

equal on objects

```
ut.expect(anydata.convertObject(l_expect))
  .to_equal(anydata.convertObject(l_actual));
```

equal on collections

```
ut.expect(anydata.convertCollection(l_expect))
  .to_equal(anydata.convertCollection(l_actual));
```

It's not possible to compare rowtypes and records directly. You have to wrap them in a `table of record`, select from them and use cursor comparsion

Advanced Matchers

be_like	`ut.expect( 'Lorem  _i m psum' ) .to_b e_like( '%re%su' );`
be_like with escape	`ut.ex  pect( 'Lorem  _i m psum' ) .to_be_like( a_mask => '%rem\_%', a_esca pe_char => '\' );` see Oracle like operator
match	`ut.exp ect ('some value') .to_m atch('^some.*');`
match with options	`ut.ex  pect( '123-4  56 - ABcd' ) .to_match( a_pattern=>'\d{3}-\d{3}-[a-z]', a_modifiers=>'i');` See regexp_like function

Cursor Comparison

Basic example

```
open l_expected for select * from all_objects;
open l_actual for select * from all_objects;
ut.expect(l_actual)
  .to_equal(l_expected)
  .join_by('COLUMN_NAME')
  .exclude(ut_varchar2_list('ID','SOMECOLUMN'))
```

Special Matchers

```
ut.exp ect (l_ act ual ).to_have_count(5);
ut.exp ect (l_ act ual ).to_be_empty();
```

Helpers

Type: List of strings

```
ut_var cha r2_ lis t(' String 1', 'String 2', ...)
```

Type: List of reporters

```
ut_rep ort ers (ut _do cum ent ati on_ rep ort er(),
_ rep ort er())
```

Select all tests

```
select * from table(
  ut_runner.get_suites_info(USER, 'PACKA GE_ NAME'))
  where item_type = 'UT_TEST';
```


Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com