Cheatography

utPLSQL v3.1.2 Cheat Sheet

by Jacek Gebal (jgebal) via cheatography.com/22528/cs/16396/

Common annotations

--%disabled	Suite / context / test will not execute
--%rollback(auto / manual)	Automatic(**default**) / manual transaction control
--%displayname(description)	Description of context / test / suite

Procedure annotations

--%test(description)	Procedure is a test (with description)
--%beforetest(procedure [, ...])	Procedure(s) to run before annotated test
--%aftertest(procedure [, ...])	Procedure(s) to run after annotated test
--%beforeall	Procedure to run before first test in suite/context
--%afterall	Procedure to run after last test in suite/context
--%beforeeach	Procedure to run before each test in suite/context
--%aftereach	Procedure to run after each test in suite/context
--%throws(exception [, ...])	Test expects exception(s) to be thrown

Package annotations

--%suite(description)	Package is a test suite (with description)
--%suitepath(com.acme.bomb)	Similar to classpath (Java) Group suites in namespaces
--%context(name)	Starts sub-suite in a suite
--%endcontext	Ends sub-suite in a suite

Package annotations (cont)

--%beforeall(procedure [, ...])	Procedure(s) to run before all tests in suite/context
--%afterall(procedure [, ...])	Procedure(s) to run after all tests in suite/context
--%beforeeach(procedure [, ...])	Procedure(s) to run before each tests in suite/context
--%aftereach(procedure [, ...])	Procedure(s) to run after each tests in suite/context

Annotations are sinlgle-line comments starting with a % sign.
Needed in package specification only (documentation)

Equality matcher

equal

```
ut.expect( 'a dog' ).to_equal(
  'a dog' , a_nulls_are_equal => false );
```

a_nulls_are_equal is **true** by default

equal with cursors

```
open l_expected for select * from all_objects;
open l_actual for select * from all_objects;
ut.expect( l_expected )
  .to_equal( l_actual )
  .exclude( 'owner' )
  .join_by( 'name' );
```

equal on objects

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

equal on collections

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

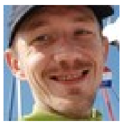
Expectation syntax

Base expectation

```
ut.expect( actual_value ).to_( matcher );
```

Negated expectation

```
ut.expect( actual_value ).not_to( matcher );
```



By **Jacek Gebal** (jgebal)
cheatography.com/jgebal/
www.oraclethoughts.com

Published 20th July, 2018.
Last updated 20th July, 2018.

Expectation syntax (cont)

Shortcuts syntax

```
ut.expect( actual_value ).to_matcher;
ut.expect( actual_value ).not_to_matcher;
```

Executing tests

`exec ut.run();`	All tests in my current schema
`alter session set current_schema='HR';` `exec ut.run();`	All tests in current schema after it was changed to HR
`exec ut.run('HR');`	All tests in specific schema
`exec ut.run('test_betwnstr');`	All tests in package of current schema
`exec ut.run('hr.test_betwnstr.big_end_position');`	Specific test only
`exec ut.run( 'hr.test_award_bonus, hr.test_betwnstr.big_end_position');`	Run several items
`exec ut.run(':com.my_org.my_project');`	Run using suitepath
`select * from table(ut.run());`	All tests as a select statement

Non-equality matchers

be_like

```
ut.expect( 'Lorem_impsum' ).to_be_like(
  a_mask => '%rem\_%', a_escape_char => '\' );
ut.expect( 'Lorem_impsum' ).to_be_like( '%re%su' );
```

a_mask, a_escape_char -> see Oracle like operator

match

```
ut.expect( '123-456-ABcd' ).to_match(
  a_pattern=>'\d{3}-\d{3}-[a-z]', a_modifiers=>'i'
);
ut.expect( 'some value' ).to_match( '^some.*' );
```

a_pattern, a_modifiers -> see regexp_like function

be_between

```
ut.expect( 3 ).to_be_between( 1, 3 );
```

Non-equality matchers (cont)

be_greater_or_equal

```
ut.expect( 3 ).to_be_greater_or_equal( 2 );
```

be_greater_than

```
ut.expect( 2 ).to_be_greater_than( 1 );
```

be_less_or_equal

```
ut.expect( 3 ).to_be_less_or_equal( 3 );
```

be_less_than

```
ut.expect( 3 ).to_be_less_than( 4 );
```

have_count

```
ut_expect( v_cursor ).to_have_count(10);
```

Unary matchers

be_empty

```
open l_cursor for select * from dual where 1 = 0;
ut.expect( l_cursor ).to_( be_empty() );
```

be_true

```
ut.expect( ( 1 = 1 ) ).to_( be_true() );
```

be_false

```
ut.expect( ( 1 = 0 ) ).to_( be_false() );
```

be_null

```
ut.expect( 1 ).to_( be_null() );
```

be_not_null

```
ut.expect( to_clob('ABC') ).to_( be_not_null() );
```

Reporting

Color output

```
exec ut.run(a_color_console=>true);
```

With sqlcl
or sqlPlus (Mac, Unix, Windows ANSICON)

JUnit reporter

```
exec ut.run(ut_junit_reporter());
```

JUnit-compatible XML report for CI servers

Coverage html reporter

```
exec ut.run(ut_coverage_html_reporter());
```

Produces HTML coverage report

Documentation for coverage and reporters