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

Package annotations (cont)

--%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_null s_a re_ equal => false );
```

a_nulls_are_equal is **true** by default

equal with cursors

```
open l_expected for select * from all_ob jects;
open l_actual for select * from all_ob jects;
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 );
```

Shortcuts syntax

```
ut.expect( actual _value ).to_matcher;
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.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

utPLSQL v3.1.2 Cheat Sheet

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

Executing tests

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

All tests in my current schema

```
alter session set curren t_s che ma= '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 ('t est _be twn str');
```

All tests in package of current schema

```
exec ut.run ('h r.t est _be twn str.bi g_e nd_ pos iti
```

Specific test only

```
exec ut.run(
    'hr.test_award_bonus, hr.tes t_b etw nst r.b ig_ end _to_ s
it ion');
```

Run several items

```
exec ut.run (': com.my _or g.m y_p roj ect');
```

Run using suitepath

```
select * from table( ut.r un());
```

All tests as a select statement

Non-equality matchers

Non-equality matchers (cont)

have_count

```
ut_expect( v_cursor ).to_h ave _co unt (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_clo b(' ABC') ).to_( be_not _null()
 );
```

Reporting

Color output

```
exec ut.run (a_ col or_ con sol e=> true);
```

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

JUnit reporter

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

JUnit-compatible XML report for CI servers

Coverage html reporter

```
exec ut.run (ut _co ver age _ht ml_ rep ort er
());
```

Produces HTML coverage report

Documentation for coverage and reporters

be_like

```
ut.expect( 'Lorem _im psum' ).to_be_like(
  a_mask => '%rem\_%', a_esca pe_char => '\' );
ut.expect( 'Lorem _im psum' ).to_b e_like( '%re%su'
 );
```

a_mask, a_escape_char -> see Oracle like operator

match

```
ut.expect( '123-4 56- ABcd' ).to_match(
  a_pattern=>'\d{3}-\d{3}-[a-z]', a_modifiers=>'i'
);
ut.expect( 'some value' ).to_m atch( '^some.*' );
```

a_pattern, a_modifiers -> see regexp_like function

be_between

```
ut.expect( 3 ).to_b e_b etween( 1, 3 );
```

be_greater_or_equal

```
ut.expect( 3 ).to_b e_g rea ter _or _equal( 2 );
```

be_greater_than

```
ut.expect( 2 ).to_b e_g rea ter _than( 1 );
```

be_less_or_equal

```
ut.expect( 3 ).to_b e_l ess _or _equal( 3 );
```

be_less_than

```
ut.expect( 3 ).to_b e_l ess _than( 4 );
```


Sponsored by **Readable.com**
Measure your website readability!
https://readable.com