Cheatography

utPLSQL v3 Cheat Sheet

by Jacek Gebal (jgebal) via cheatography.com/22528/cs/11565/

Unit Test Package definition

Name	Level	Info
--%suite	Package	The package is a unit test suite
--%suite(description)	Package	Gives description for suite
--%suitepath(path.with.dots)	Package	Similar to Java package. Allows grouping of suites in namespaces
--%test	Procedure	Defines procedure as a test
--%test(description)	Procedure	Gives description for test
--%beforetest(procedure_name)	Procedure	Name of procedure to execute before annotated test
--%aftertest(procedure_name)	Procedure	Name of procedure to execute after annotated test
--%beforeall	Procedure	Setup to be run before suite
--%afterall	Procedure	Cleanup to be run after suite
--%beforeeach	Procedure	Setup to be run for each test
--%aftereach	Procedure	Cleanup to be run aftrer each test
--%disabled	both	Identifes suite or test as disabled
--%rollback(auto)	both	Defines automatic rollback to savepoint for suite or test.**Default**

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

Published 24th April, 2017.
Last updated 25th April, 2017.

Unit Test Package definition (cont)

--%rollback(manual)	both	Defines manual transaction control. No rollback is performed

Annotations are comments starting with a % sign. They need to be defined in package specification only. Annotations in package body are ignored.
utPLSQL uses annotations to identify database package as a unit test suite.
Check documentation for details

Matchers

be_between

Validates that actual is betwen lower and upper bound

```
ut.expect( 3 ).to_( be_between( 1, 3 ) );
```

be_empty

Validates that the provided data-set is empty

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

be_false

Validates that the provided value is false

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

be_greater_or_equal

Validates that actual value is greater or equal expected

```
ut.expect( sysdate ).to_( be_greater_or_equal( sysdate - 1 ) );
```

be_greater_than

Validates that actual value is greater than expected

```
ut.expect( 2 ).to_( be_greater_than( 1 ) );
```

be_less_or_equal

Validates that actual value is less or equal expected

```
ut.expect( 3 ).to_( be_less_or_equal( 3 ) );
```

be_less_than

Validates that actual value is less than expected

```
exec ut.expect( 3 ).to_( be_less_than( 2 ) );
```

be_like

Validates that actual value is like the expected expression

```
ut.expect( 'Lorem_impsum' ).to_( be_like( a_mask => '%rem\_%', a_escape_char => '\' ) );
ut.expect( 'Lorem_impsum' ).to_( be_like( '%rem%sum' ) );
```

Parameters a_mask and a_escape_char represent a valid parameters of theOracle like operator

be_not_null

Validates that actual value is not null

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

be_null

Validates that actual value is null

```
ut.expect( cast(null as varchar2(100)) ).to_( be_null() );
```

Matchers (cont)

be_true

Validates that the provided value is false

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

equal

Validates that actual is equal expected

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

The a_nulls_are_equal parameter is **true by default**

equal on cursor data

```
declare
l_expected sys_refcursor;
l_actual sys_refcursor;
begin
open l_expected for select from dual;
open l_actual for select from dual where 1 = 0;
ut.expect( l_cursor ).to_( equal(l_actual) );
end;
```

equal on objects

```
declare
l_expected department := department('HR');
l_actual department := department('IT');
begin
ut.expect( anydata.convertObject(l_expected) ).to_( equal( anydata.convertObject(l_actual) ) );
end;
```

equal on collections

```
declare
l_expected departments := departments(department('HR'));
l_expected departments := departments(department('IT'));
begin
ut.expect( anydata.convertCollection(l_expected) ).to_( equal( anydata.convertCollection(l_actual) ) );
end;
```

match

Validates that actual value is matching the expected regular expression

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

Parameters a_pattern and a_modifiers represent a valid regexp pattern accepted by Oracle regexp_like function

negating matcher

Every matcher can be negated by simple replacement of `.to_()` with `.not_to()`

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

negating matcher

Every matcher can be negated by simple replacement of `.to_()` with `.not_to()`

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

Cheatography

utPLSQL v3 Cheat Sheet

by Jacek Gebal (jgebal) via cheatography.com/22528/cs/11565/

Executing tests

Run all unit tests in my current schema

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

Run all unit tests in current schema after it was changed to HR

```
alter session set current_schema='HR';
exec ut.run();
```

Run all unit tests in specific schema

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

Run all unit tests in specific package of current schema

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

Run all unit tests in specific schema.package

```
exec ut.run('hr.test_betwnstr');
```

Run one specific test only

```
exec ut.run('hr.test_betwnstr.big_end_position');
```

Combining the above

Run all tests from test_award_bonus package and one test procedure from test_betwnstr

```
exec ut.run(ut_varchar2_list('hr.test_award_bonus','hr.test_betwnstr.big_end_position'));
```

Run test using suitepath in current schema

```
exec ut.run(':com.my_org.my_project');
```

Runs all test packages that are on the suitepath `com.my_org.my_project...`

Run the tests as a select statement

```
select * from table(ut.run());
```

All the above syntaxes are still applicable

Reporting

Use color output for default reporter

Works using sqlPlus on Unix, Windows (with ANSICON)

Works on any platrowm when using Oracle sqlcl (new Oracle SQL console)

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

Run with XUnit reporter

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

Produces XML output compatible with JUnit for use with CI servers like Jenkins

Run with TeamCity reporter

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

Produces TeamCity-specific output

Run with Sonar Test reporter

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

Produces XML output to be consumed by Sonar server

By **Jacek Gebal** (jgebal)

cheatography.com/jgebal/

www.oraclethoughts.com

Published 24th April, 2017.

Last updated 25th April, 2017.

Reporting (cont)

Run with coverage html reporter

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

Produces HTML coverage report output.

See documentation for details

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

Published 24th April, 2017.
Last updated 25th April, 2017.