GCBLUnit tool

Simple Unit Testing for GnuCOBOL written in GnuCOBOL.

Features

- Assertions
- Reporting in JUnit format
- Continuous Integration
- No mainframe required
- GnuCOBOL Docker

Requirements

You may choose between *local* and *container* execution environment.

Local

GnuCOBOL `cobc` 2.2+ installed.

Container

GnuCOBOL Docker container up and running. The image includes GnuCOBOL and all required dependencies needed to debug or execute your code.

Installation

Simply download gcblunit.cbl file or install by COBOL Package Manager:

```
$ npm install -g cobolget
$ cobolget init
$ cobolget add --debug gcblunit
```

```
$ cobolget update
$ cobolget install
$ cobc -x -debug modules/gcblunit/gcblunit.cbl --job=-h
GCBLUnit 1.22.6  by Olegs Kunicins and contributors.

Usage:
  cobc -x -debug gcblunit.cbl first-test.cbl [next-test.cbl] --job='first-
test [next-test]'
  cobc -x -debug gcblunit.cbl --job=Options

Options:
  -h, -help                Print this help
  -v, --version            Print the version
  --stop-on-error          Stop on the first exception
  --stop-on-failure        Stop on the first failure
  --junit report.xml       Report in JUnit XML format
```

Usage

```
$ cobc -x -debug gcblunit.cbl tests/* --job='equals-test notequals-test'
GCBLUnit 1.22.6  by Olegs Kunicins and contributors.

..............................................................

Time: 00:00:00

OK
Tests: 0000000002, Skipped: 0000000000
Assertions: 0000000062, Failures: 0000000000, Exceptions: 0000000000
```

Writing Tests

Tests are simple COBOL programs that allow further execution (without `stop run`). There is no code-generation tricks nor injections. The assertions are GnuCOBOL programs and await two values - expected and actual, respectively:

```
call "assert-equals" using "expected", "actual".
```

This assertion, once included into the unit-testing, will lead to one failed test. More examples you may find in the `tests` directory.

At the moment these assertions are supported:

- assert-equals
- assert-notequals

GCBLUnit catches exceptions and stops. For instance, the statement `compute y = y / 0.` is getting reported this way:

```
GCBLUnit 1.22.6  by Olegs Kunicins and contributors.

There was an exception: EC-SIZE-OVERFLOW in exception-test; ; 33 on
COMPUTE

Time: 00:00:00

EXCEPTIONS!
Tests: 0000000001, Skipped: 0000000000
Assertions: 0000000000, Failures: 0000000000, Exceptions: 0000000001
```

Continuous Integration

GCBLUnit returns an exit-code of the execution that is usually enough for CI pipelines. Additional details you may export to a file in JUnit XML format by using `--junit` option.

Alternatives

GCBLUnit primarily focuses on Unit Testing - isolated GnuCOBOL functions and programs with an input and output.

Nonetheless, you may try two alternatives as well:

- `cobol-unit-test` - a paragraph-level Unit Testing framework, written by Dave Nicolette, hosted on GitHub.
- `COBOLUnit` - a full-featured Unit Testing framework for COBOL, written by Hervé Vaujour, hosted on Google Sites. Not updated since 2010.

TODO

- Assertion `assert-greater`
- Assertion `assert-less`
- Assertion `assert-contains`
- Assertion `assert-notcontains`
- Auto-discovery of the tests in the compilation group
- Integration with Debugger for GnuCOBOL

Your contribution is always welcome!