Unit Testing

Why and How to Write Unit Tests in KDE?

David Faure
faure@kde.org

Kévin Ottens
ervin@kde.org

aKademy 2007, Glasgow

What is a Unit Test?

- "Unit testing is a procedure used to validate that individual units of source code are working properly" (Wikipedia)
- A written contract that a given piece of code must satisfy
- In practice: steps and conditions
  - in a document
  - as code

TDD: What?

Test Driven Development

"Software development technique that involves repeatedly first writing a test case and then implementing only the code necessary to pass the test" (Wikipedia)

TDD: Why?

- Get immediate feedback
- Improve the design and code:
  - Close to Design by Contract
  - Modularized code
  - Easier refactoring
- Reduced need for a debugger
- Better trust in the code overall
  - Improved test coverage
  - Less defects

TDD: How?

1. Add a test
2. Run all tests and see the new one fail
3. Write some code
4. Run the automated tests and see them succeed, otherwise goto 3
5. Code cleanup, test should still pass
6. goto 1

In short: Rock climbing progression, each test is a carabiner

Automated tests

- Code which tests code
- As to be fast to be worthwile
- Green bar approach (PASS/FAIL)
- More advanced technics
  - Data driven tests
  - Mock objects

Unit tests & KDE

- Why?
  - Benefits for your design and code
  - Long term
    - Run them regularly on the EBN
    - Compute test coverage
- How?
  - Build them
  - Run them before commit and after update!
  - Implementation details following

aKademy 2007, Glasgow

QTestLib

- Part of Qt
- GPL (+commercial)
- One testcase -> one executable
- Uses slot introspection to run test methods

- QCOMPARE(a, b)
- QVERIFY(bool)
- QVERIFY2(bool, "some bug happened")

QTestLib example

```
#include <QtCore/QObject>

class KLocaleTest : public QObject
{
    Q_OBJECT

private Q_SLOTS:
    void testReadTime();
    ...
};
```

QTestLib example

```
#include "klocaletest.h"
#include <qtest_kde.h>
#include <klocale.h>

QTEST_KDEMAIN_CORE(KLocaleTest)

void KLocaleTest::readTime()
{
    KLocale* locale = KGlobal::locale();
    bool ok = false;
    QCOMPARE(locale->readTime("11:22:33", &ok),
                        QTime(11,22,33));
    QVERIFY(ok);
}
#include "klocaletest.moc"
```

CMake file

```
set(klocaletest_SOURCES klocaletest.cpp)
kde4_automoc(${klocaletest_SOURCES})
kde4_add_unit_test(klocaletest ${klocaletest_SOURCES})
target_link_libraries(klocaletest ${KDE4_KDECORE_LIBS}
                                  ${QT_QTTEST_LIBRARY})
```

Many tests -> use macro, see kdecore/tests for example

```
KDECORE_UNIT_TESTS(
 klocaletest
 klocalizedstringtest
 ...
)
```

QTestLib output

```
./klocaletest
********* Start testing of KLocaleTest *********
Config: Using QTest library 4.3.0, Qt 4.3.0
PASS   : KLocaleTest::initTestCase()
PASS   : KLocaleTest::readTime()
PASS   : KLocaleTest::cleanupTestCase()
Totals: 3 passed, 0 failed, 0 skipped
********* Finished testing of KLocaleTest *********
```

GREEN!

QTestLib output

```
./klocaletest
********* Start testing of KLocaleTest *********
Config: Using QTest library 4.3.0, Qt 4.3.0
PASS   : KLocaleTest::initTestCase()
FAIL!  : KLocaleTest::readTime() Compared values are not the same
   Actual (locale->readTime("11:22:33", &ok)): 11:22:33.000
   Expected (QTime(11,22,34)): 11:22:34.000
   Loc: [/d/kde/src/4/kdetoys/tests/klocaletest.cpp(13)]
PASS   : KLocaleTest::cleanupTestCase()
Totals: 2 passed, 1 failed, 0 skipped
********* Finished testing of KLocaleTest *********
```

RED!

Regression testing

- Run all tests before committing:

make && make test

```
Running tests...
Start processing tests
Test project /d/kde/build/4/kdelibs/kdecore/tests
  1/ 36 Testing klocaletest                    Passed
  2/ 36 Testing klocalizedstringtest           Passed
   [...]
 36/ 36 Testing kmimetypetest                  Passed

100% tests passed, 0 tests failed out of 36
```


Testing app code

Tests are not only for libraries

- Test self-contained app classes:

```
set(fooparsertest_SOURCES fooparsertest.cpp ../fooparser.cpp)
```

- Make static lib
  (cannot be used in shared lib)

- Make shared lib, versionned and installed
(already done by kdeinit_kmyapp)
  Needs FOO_TEST_EXPORT

More information

Tutorial by Brad Hards

http://developer.kde.org/documentation/tutorials/writingunittests/writingunittests.html

We're counting on you

Write unit tests!
Run existing unit tests!

Start today!
Du musst!
Fais-le maintenant!

aKademy 2007, Glasgow