QtTestLib
Qt Unit Testing Library

Harald Fernengel <harald@trolltech.com>
What Is It?

- Lightweight unit testing library
- Cross-platform, cross-compiler
- Tests are written in C++
- Tests are stand-alone executables
Features

• Data-driven testing
• Basic GUI testing
• Qt Signal/Slot introspection
• IDE integration (KDevelop, VS)
#include <QtTest/QtTest>

class QStringTest: public QObject
{
    Q_OBJECT
private slots:
    void toUpper()
    {
        QString str = "text";
        COMPARE(str.toUpper(), QString("TEXT"));
    }
};
QTTEST_MAIN(QStringTest)
Building it

• **Run** `qmake -project CONFIG+=qttest`
• `qmake && make`
Macros

• **VERIFY** - Verifies that the condition is true:
  
  \[
  \text{VERIFY} \ (i + j \ == \ 6);
  \]

• **COMPARE** - Compares two values:
  
  \[
  \text{COMPARE} \ (i + j, \ 6);
  \]
Run a test multiple times with different data:

```cpp
void toUpper_data(QtTestTable &t) {
    t.defineElement("QString", "string");
    t.defineElement("QString", "result");

    *t.newData("lower") << "kde" << "KDE";
    *t.newData("mixed") << "KdE" << "KDE";
}
```
• The same test, this time data-driven:

```c++
void toUpper()
{
    FETCH(QString, string);
    FETCH(QString, result);

    COMPARE(string.toUpper(), result);
}
```
Benefits

• Separation of logic and data
• Improved readability
• Easily extendable
• Eases testing of border cases
• Reduces copy-paste code in tests
GUI Testing

- Keyboard and Mouse simulation
- Sends Qt events (no X11 events)
- Supports clicking, double-clicking, pressing and releasing of keys and mouse movement
```cpp
void testGui()
{
    QLineEdit lineEdit;
    QTest::keyClicks(&lineEdit, "hi KDE");
    COMPARE(lineEdit.text(), "hi KDE");
}
```
GUI Testing: Mouse

- `mouseClick()`, `mousePress()` and `mouseRelease()` all take:
  - a widget
  - a mouse button
  - an optional modifier (Shift/Ctrl/Alt)
  - a position (default: center of widget)
  - an optional delay
GUI Testing: Keys

- `keyClick()`, `keyPress()` and `keyRelease()` all take:
  - a widget
  - a char or a `Qt::Key`
  - an optional keyboard modifier
  - an optional delay
GUI Testing: Testdata

- GUI events can be recorded:

```cpp
void guiTest_data(QtTestTable &t) {
    t.defineElement("QtTestEventList", "e");

    QTestEventListList list;
    list.addKeyClick('a');
    list.addKeyClick(Qt::Key_Backspace);

    *t.newData("there and back") << list;
}
```
GUI Testing: Replay

- A QTestEventList can be replayed multiple times:

```cpp
void guiTest()
{
    FETCH(QTestEventList, e);
    QLineEdit lineEdit;
    e.simulate(&lineEdit);
    VERIFY(lineEdit.text().isEmpty());
}
```
Signal introspection

- QSignalSpy is useful to introspect signals:

```cpp
QCheckBox box;
QSignalSpy spy(&box, SIGNAL(clicked(bool));
box.animateClick();
COMPARE(spy.count(), 1);
QList<QVariant> arguments = spy.takeFirst();
COMPARE(arguments.at(0).toBool(), true);
```
• **QSignalSpy** can connect to any signal from any `QObject`

• It can handle any kind of parameter as long as it is registered with `QMetaType`

• It is implemented as a list of list of `QVariant`

• It "fakes" slots at runtime, heavily misusing Qt's meta object system.
Test Output

- Output goes to stdout
- Outputs plain text or XML
- Supports colored output
- Messages are atomar and thread-safe
- IDE-friendly output
- Verbose output, Signal/Slot dumper
Other Good Stuff

- **EXPECT_FAIL** - Marks the next VERIFY/COMPARE as expected failure
- **SKIP** - Skips the test and outputs a message
- **VERIFY2** - Verbose VERIFY
- **ignoreMessage()** - swallows debug/warn messages
Summary

- Universal toolbox for testing Qt code
- Lightweight - 6000 LOC, 60 symbols ➞ Easy to learn, easy to maintain
- Tests in C++, standard executables ➞ No special environment/task-switch needed
- Self-contained, cross-platform, cross compiler ➞ Runs everywhere Qt does
That's It

Questions?