

Unit-testing PyGTK applications

Website: <http://gintas.pov.lt/gtktest>

by *Gintautas Miliauskas* <gintas@pov.lt>
Programmers of Vilnius

EuroPython 2005
Göteborg, Sweden

Outline

- Why is unit-testing GUI applications useful?
- How to use the *gtktest* library
- Things to remember while using *gtktest*
- Other GUI application testing tools

Why are GUI frameworks special?

- Lots of different types of objects
- Deep class hierarchies
- Complex interactions
- Much 'global' state

Why unit-testing?

- Test-driven development
(which implies test-driven design)
- Relative ease of achieving complete test coverage
- Simplicity

Basic goals

- Lightweight, unobtrusive and easy to use helpers
- Compatibility with the standard *unittest* and *doctest* libraries
- No actual widgets displayed during test runs
- Some support for stubbing, mock objects and logging

Simple PyGTK application

```
import gtk

class SimpleApp(object):

    def __init__(self):
        self.counter = 0
        self.button = gtk.Button(str(self.counter))
        self.button.connect('clicked', self.click)
        window = gtk.Window()
        window.add(self.button)
        window.show_all()

    def click(self, button):
        self.counter += 1
        self.button.set_label(str(self.counter))

if __name__ == '__main__':
    app = SimpleApp()
    gtk.main()
```

Simple unit test

```
import gtktest  
import unittest, doctest  
  
class TestSimpleApp(gtktest.GtkTestCase):  
  
    def test_SimpleApp(self):  
        from simple import SimpleApp  
        app = SimpleApp()  
        self.assertEquals(app.button.get_label(), '0')  
        app.button.clicked()  
        self.assertEquals(app.button.get_label(), '1')  
  
if __name__ == '__main__':  
    unittest.main()
```

Simple doctest

```
from gtktest import doctest_setup, doctest_tearDown
import unittest, doctest

def doctest_SimpleApp():
    """Test for SimpleApp.
        >>> from simple import SimpleApp
        >>> app = SimpleApp()
        >>> app.button.get_label()
        '0'
        >>> app.button.clicked()
        >>> app.button.get_label()
        '1'
    """

def test_suite():
    return doctest.DocTestSuite(setUp=doctest_setup,
                                tearDown=doctest_tearDown)

if __name__ == '__main__':
    unittest.main(defaultTest='test_suite')
```

The gtktest.gtkinfo singleton

- `main` – the callable invoked when code calls *gtk.main()*
- `level` – depth of main loop recursion
- `dlg_handler` – the callable invoked when `Dialog.run()` is called. The dialog is passed as an argument and the return value is the response.
- `dlg_response` – the response of `Dialog.run()`, if your `dlg_handler` would only be a single *return*.
- `calls` – the journal used by logging support

Conveniences

- the **mainloop_handler** decorator
Run the test instead of the main event loop

```
class TestFoo(gtktest.GtkTestCase):  
    @mainloop_handler(app.main)  
    def test_bar(self):  
        ...
```

- **overrides**
Replace a class or method with your provided object before starting a test, restore afterwards
- **logging**
Log method calls

Using overrides and logging

In unit tests

```
class TestMyApp(GtkTestCase):  
    overrides = {'gtk.Entry': EntryStub}  
    logging = {'gtk.Entry': ['set_text']}  
  
    def test_foo(self):  
        ...
```

In doctests:

```
def test_suite():  
    setUp = doctest_setUp_param(  
        overrides={'gtk.Entry': EntryStub}  
        logging={'gtk.Entry': ['set_text']})  
  
    return doctest.DocTestSuite(  
        setUp=setUp, tearDown=doctest_tearDown)
```

Things to have in mind

- *gtktest* must always be imported **before** the application or the real *gtk* module is imported
- Overriding objects that are used in top-level scope *does not work* (unless you update the references manually).
- Class, method overrides and logging do not work when the widget is created by *glade*.
- Doctest setup functions only work on Python 2.4.

Rough edges

- Windows and dialogs constructed by *glade* should not be visible by default, or you will see windows flashing while running tests.
- Windows and dialogs created by *glade* are proxied.
- *show()* and *show_all()* are overridden, but you can still display a window by accessing properties directly.

Unit-testing other GUI toolkits

- **QT**

- pyGUIUnit

- <http://sourceforge.net/projects/pyguiunit>

- **wxWidgets:**

- wxMock.py

- <http://dirtsimple.org/2004/12/mocking-wxpython.html>

- NDtestmaker

- http://wiki.wxpython.org/index.cgi/Unit_20Testing_20with_20wxPython

Functional testing tools

- **LDTP**
(GNU/Linux Desktop Testing Project)
http://gnomebangalore.org/ldtp/index.php/Main_Page
- **PyUseCase (with TextTest)**
<http://sourceforge.net/projects/pyusecase>

The End

Questions?

<http://gintas.pov.lt/gtktest>

Gintautas Miliauskas <gintas@pov.lt>
2005