Programming Qt + KDE with Python

Simon Edwards <simonedwards@kde.nl>
What is Python?

• High level, interpreted programming language.

• Features:
  – Simple & concise syntax
  – Built in lists, hash/dictionary datatypes
  – Strong dynamic typing
  – Object oriented
  – Garbage collection
  – Exceptions and stacktraces
  – Meta-programming + generators
  – Large standard library

• Benevolent dictator: Guido van Rossum.
What I've been doing with it...
What are language bindings?

- Glue between the object model, type system, calling conventions etc from one language to another. (KDE) C++ «» Python in this case.

- Components:
  - SIP: binding generator.
  - PyQt: bindings for Qt.
  - PyKDE: bindings for KDE.

- GPL, (commercial licenses available)

- Thanks go to Phil Thompson and Jim Bublitz.
KDE 3 Example

#!/usr/bin/python
# GPL. (C) 2005 Simon Edwards.
from qt import *
from kdecore import *
from kdeui import *
import sys

aboutdata = KAboutData("test","test","1.0", "A test", \
   KAboutData.License_GPL, "Copyright (C) 2005-7 Simon Edwards")
KCmdLineArgs.init(sys.argv,aboutdata)
kapp = KApplication()

label = QLabel("aKademy 2007",None)
label.adjustSize()
label.show()

kapp.setMainWidget(label)
kapp.exec_loop()
DCOP Example

• KDE's interprocess communication system.

#!/usr/bin/python
# GPL. (C) 2005 Simon Edwards.
import sys
import dcopext
from kdecore import *

aboutdata = KAboutData("dcop_test","dcop_test","1.0", "A test",
                      KAboutData.License_GPL, "Copyright (C) 2005-7 Simon Edwards")
KCmdLineArgs.init(sys.argv,aboutdata)
app  = KApplication()

dcop = app.dcopClient()
kwin = dcopext.DCOPApp("kicker", dcop)
ok,size = kwin.Panel.panelSize()
print "Panel size is: ",size
Subclass Example

# A read-only ListBox item that also uses the 'alternate' background
# colour as background.
class ROListBoxItem(QListBoxPixmap):
    def __init__(self,listbox,pix,text):
        QListBoxPixmap.__init__(self,listbox,pix,text)
        self.bgc = KglobalSettings.alternateBackgroundColor()
        self.setSelectable(False)

    def paint(self,p):
        boldfont = QFont(p.font())
        boldfont.setWeight(QFont.Bold)
        p.setFont(boldfont)
        p.setBackgroundColor(self.bgc)
        p.eraseRect(0,0,
                    self.listBox().width(),
                    self.height(self.listBox()))
        QListBoxPixmap.paint(self,p)
Qt-Designer
Performance & Cost

• GUI programs are typically I/O bound by the user, network, harddisk etc.

• Qt/KDE (C++) handles the GUI, menus, widgets etc.

• Overhead: Python Interpreter, PyQt and PyKDE.
Development Tools

- Eric3: Python IDE, using PyQt.
- Debugger.
- Unit testing.
- Version Control.
- Refactoring.
- Coverage.
Embedding & Plugins

![Graphical User Interface of Disk & Filesystems configuration](image)

**Available Disks and Filesystems:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Mount Point</th>
<th>Type</th>
<th>Device</th>
<th>Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optical Disk 'LITE-ON LTR-16102B'</td>
<td>/mnt/win_c</td>
<td>ntfs</td>
<td>/dev/hda1</td>
<td>Enabled</td>
</tr>
<tr>
<td>System</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disk 'Maxtor 6Y120LC'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Partition 20.0 Gb</td>
<td>/mnt/win_c</td>
<td>ntfs</td>
<td>/dev/hda1</td>
<td>Enabled</td>
</tr>
<tr>
<td>2 Partition 20.0 Gb</td>
<td>/</td>
<td>reiserfs</td>
<td>/dev/hda2</td>
<td>Enabled</td>
</tr>
<tr>
<td>3 Partition 1.9 Gb</td>
<td>swap</td>
<td>swap</td>
<td>/dev/hda3</td>
<td>Enabled</td>
</tr>
<tr>
<td>4 Partition 1 Gb</td>
<td>swap</td>
<td>swap</td>
<td>/dev/hda4</td>
<td>Enabled</td>
</tr>
<tr>
<td>5 Partition 27.9 Gb</td>
<td>/mnt/data</td>
<td>vfat</td>
<td>/dev/hda5</td>
<td>Enabled</td>
</tr>
<tr>
<td>6 Partition 44.6 Gb</td>
<td>/mnt/data2</td>
<td>reiserfs</td>
<td>/dev/hda6</td>
<td>Enabled</td>
</tr>
<tr>
<td>/mnt/cdrom</td>
<td>/mnt/cdrom</td>
<td>auto</td>
<td>/dev/hdc</td>
<td>Disabled</td>
</tr>
<tr>
<td>/mnt/floppy</td>
<td>/mnt/floppy</td>
<td>supermount none</td>
<td>Enabled</td>
<td></td>
</tr>
<tr>
<td>/mnt/blik_d</td>
<td>/mnt/blik_d</td>
<td>smbfs</td>
<td>\blick\D$</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

**Options:**

- New...
- Modify...
- Delete...
- Enable
- Disable
Side effects

- Rapid development
- Rapid debugging. Useful bug reports with exceptions, stacktraces and line numbers.
- Lowers the tinker
ing barrier
- ...and hence lowers the barrier for contributors
- Puts the fun back in development
Current Status

- PyQt 4.0 for Qt 4.1 released 10 June 2006
- Current version is PyQt v4.2 (Qt 4.2)
- Snapshots support Qt 4.3.
- PyKDE beta release expected around the first KDE 4 beta release.
Book

“Rapid GUI Programming with Python and Qt” by Mark Summerfield

• Covers Python 2.5, PyQt 4.2 & Qt 4.2.
• On safari.informit.com
• Printed version coming end October in the US.
• ~550 page.
• (Disclaimer: I haven't read it, but it sounds promising.)
Other Python Books

- “A Byte of Python” - beginners Python book
- “Dive into Python” - intermediate Python book