PyQt4中的事件和信号

在PyQt4教程的这部分中，我们将探讨应用中事件和信号的发生。

事件是GUI程序的重要部分，由用户或者系统产生。当我们调用应用的 `exec_()` 方法，应用进入主循环。主循环获取事件并把它们发往对象。Trolltech引入了独一无二的信号和槽机制。

事件是GUI程序的主要部分，所有GUI应用程序都是事件驱动的。应用在它的生命周期中产生的不同事件交互。事件主要由用户产生，但是它们也可以由其他方式产生，如：互联网，窗口管理器，定时器。在事件模型中，由三个参与者：

::

- event source
- event object
- event target

- 事件来源
- 事件对象
- 事件目标

事件对象 是指状态改变的对象，它产生了事件。 **事件对象** （Event）封装了事件元的状态改变。 **事件目标** 是事件对象想要 想要通知的对象。事件来源对象代理了事件的目标要处理的任务。

当我们调用了程序 `exec_()` 方法，程序进入了主循环。主循环捕获事件并把它们发往对象。信号和槽用于对象之间的通讯。 当一个特殊的事件发生时，将发射 **信号** ， **槽** 可以是任何Python调用，当链接到槽的信号发射，该槽将被调用。

信号和槽

这是一个演示PyQt4信号和槽的简单例子。

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# sigslot.py

import sys
from PyQt4 import QtGui, QtCore


class Example(QtGui.QWidget):

    def __init__(self):
        super(Example, self).__init__()

        self.initUI()

    def initUI(self):

        lcd = QtGui.QLCDNumber(self)
        slider = QtGui.QSlider(QtCore.Qt.Horizontal, self)

        vbox = QtGui.QVBoxLayout()
        vbox.addWidget(lcd)
        vbox.addWidget(slider)

        self.setLayout(vbox)
        self.connect(slider,  QtCore.SIGNAL('valueChanged(int)'), lcd,
            QtCore.SLOT('display(int)'))

        self.setWindowTitle('Signal & slot')
        self.resize(250, 150)


app = QtGui.QApplication(sys.argv)
ex = Example()
ex.show()
sys.exit(app.exec_())
```

在我们的例子中，我们显示了一个LCD数字和一个滑块，通过拖动滑块来改变LCD的值。

::

    self.connect(slider, QtCore.SIGNAL('valueChanged(int)'), lcd, QtCore.SLOT('display(int)'))

这里我们连接滑块的 `valueChanged()` 信号到LCD数字的 `display()` 槽。

`connect` 方法有4个参数， `sender` 是发送信号的对象， `signal` 是发射的信号， `receiver` 是接收信号的对象， 最后， `slog` 是对信号反应的方法。

图：信号和槽

重新实现事件处理程序

Events in PyQt4 are processed often by reimplementing event handlers.

PyQt4中的事件经常被重新实现。

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# escape.py

import sys
from PyQt4 import QtGui, QtCore


class Example(QtGui.QWidget):

    def __init__(self):
        super(Example, self).__init__()

        self.setWindowTitle('Escape')
        self.resize(250, 150)

    def keyPressEvent(self, event):
        if event.key() == QtCore.Qt.Key_Escape:
            self.close()


app = QtGui.QApplication(sys.argv)
ex = Example()
ex.show()
sys.exit(app.exec_())
```

在我们的例子中，我们重新实现了 `keyPressEvent()` 处理。

```
def keyPressEvent(self, event):
    if event.key() == QtCore.Qt.Key_Escape:
        self.close()
```

如果我们按下了escape按钮，程序将退出。

事件发送者

有时需要方便的知道哪个组件发出的信号，PyQt4有 `sender()` 方法。

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# sender.py

import sys
from PyQt4 import QtGui, QtCore


class Example(QtGui.QMainWindow):

    def __init__(self):
        super(Example, self).__init__()

        self.initUI()

    def initUI(self):

        button1 = QtGui.QPushButton("Button 1", self)
        button1.move(30, 50)

        button2 = QtGui.QPushButton("Button 2", self)
        button2.move(150, 50)

        self.connect(button1, QtCore.SIGNAL('clicked()'),
            self.buttonClicked)
```

```
        self.connect(button2, QtCore.SIGNAL('clicked()'),
            self.buttonClicked)

        self.statusBar().showMessage('Ready')
        self.setWindowTitle('Event sender')
        self.resize(290, 150)


    def buttonClicked(self):

        sender = self.sender()
        self.statusBar().showMessage(sender.text() + ' was pressed')

app = QtGui.QApplication(sys.argv)
ex = Example()
ex.show()
sys.exit(app.exec_())
```

这个了例子中由两个按钮，在 `buttonClicked()` 方法中我们通过调用 `sender()` 方法确定点击了哪个按钮。

```
self.connect(button1, QtCore.SIGNAL('clicked()'),
    self.buttonClicked)

self.connect(button2, QtCore.SIGNAL('clicked()'),
    self.buttonClicked)
```

两个按钮都连接了同一个信号。

```
def buttonClicked(self):

    sender = self.sender()
    self.statusBar().showMessage(sender.text() + ' was pressed')
```

通过调用 `sender()` 方法我们确定信号来源。在程序的状态栏，我们显示按下的按钮的标签。

Figure: Event sender

图：事件发送者

发射信号

从 `QtCore.QObject` 继承的对象可以发射信号。如果点击按钮，将产生一个 `clicked()` 信号。在接下来的例子中可以看到如何发射信号。

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# emit.py

import sys
from PyQt4 import QtGui, QtCore


class Example(QtGui.QWidget):

    def __init__(self):
        super(Example, self).__init__()

        self.initUI()

    def initUI(self):

        self.connect(self, QtCore.SIGNAL('closeEmitApp()'),
            QtCore.SLOT('close()'))

        self.setWindowTitle('emit')
        self.resize(250, 150)


    def mousePressEvent(self, event):
        self.emit(QtCore.SIGNAL('closeEmitApp()'))


app = QtGui.QApplication(sys.argv)
ex = Example()
ex.show()
```

```
sys.exit(app.exec_())
```

我们创建一个名为 `closeEmitApp()` 的新信号，在鼠标的按下实践中发射该信号。

```
def mousePressEvent(self, event):
    self.emit(QtCore.SIGNAL('closeEmitApp()'))
```

通过 `emit()` 方法发射信号。

```
self.connect(self, QtCore.SIGNAL('closeEmitApp()'),
    QtCore.SLOT('close()'))
```

这里我们把手工创建的 `closeEmitApp()` 信号和 `close()` 槽连接。

在PyQt4教程的这部分，我们涵盖了信号和槽。