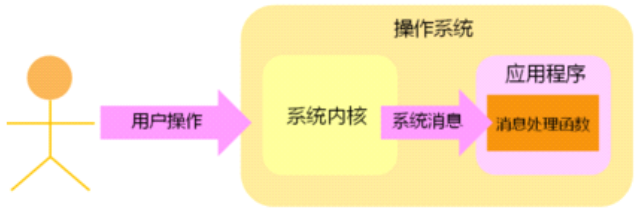
Qt 中的事件处理

2019年4月7日 星期日　20:56

1、图形界面应用程序的消息处理模型



特点：

- 基于**操作系统**才能运行
- GUI应用程序提供的功能必须**由用户触发**
- 用户操作界面时操作系统是第一个感知的
- 系统内核的消息通过事件处理转变成QT的信**号**

2．Qt中的事件处理

（1）在Qt中，**事件**被封装成一个个对象，**所有的事件均继承自抽象类QEvent**．

事件处理的核心包括事件

①产生

②分发

③接受和处理

①事件的产生

谁来产生事件？ 最容易想到的是我们的输入设备，比如键盘、鼠标产生的keyPressEvent，keyReleaseEvent，mousePressEvent，mouseReleaseEvent事件（他们被封装成QMouseEvent和QKeyEvent）。

②Qt中事件的分发

谁来负责分发事件？对于non-GUI的Qt程序，是由**QCoreApplication**负责将QEvent分发给QObject的子类-Receiver．

对于Qt GUI程序，由**QApplication**来负责

③事件的接受和处理

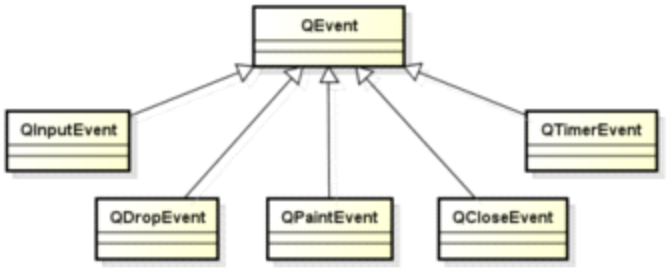
谁来接受和处理事件？答案是QObject。

QObject类是整个Qt对象模型的心脏，任何一个想要接受并处理事件的对象均须继承自QObject，可以选择重写QObject::event()函数或事件的处理权转给父类。

（2）Qt平台将**系统产生的消息转变成Qt事件**

- Qt**事件**是一个**QEvent（或子类）**的**对象**
- 有时一个事件包含多个**事件类型**，比如鼠标事件又可以分为鼠标**按下**、**双击**、和**移动**多种操作
- 事件类型由QEvent类的**枚举型QEvent::Type**来表示

- Qt事件用于描述**程序内部或外部发生的对应动作（**描述的是操作系统发生来的消息，一个系统消息对应着一个消息事件**)**
- 任意**QObject对象**都具备时间**处理**的能力



QEvent及其子类对象

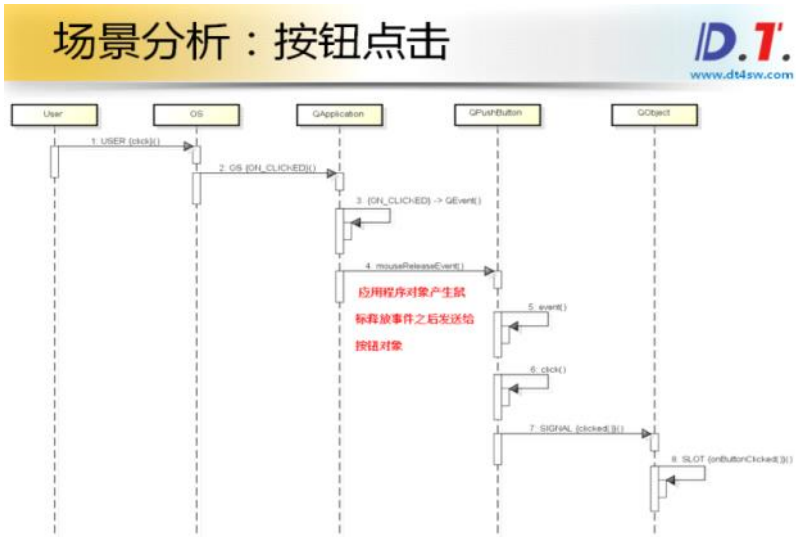
note：QEvent子类可以表示一个事件，但并不能处理这个事件

Qt 程序需要在main()函数创建一个QApplication对象，然后调用它的**exec()函数**。这个函数就是开始 **Qt 的事件循环**。在执行exec()函数之后，程序将进入事件循环来**监听**应用程序的**事件**。当事件发生时，Qt 将创建一个事件对象。Qt 中所有事件类都继承于QEvent。在事件对象创建完毕后，Qt 将这个事件对象传递给QObject的event()函数。event()函数并不直接处理事件，而是将这些事件对象按照它们不同的类型，分发给不同的事件处理器（event handler）。如上所述，event()函数主要用于事件的分发。

（3）GUI应用程序的事件处理方式

- Qt事件产生后会立即被分发到QWidget对象（QObject的子类，如按键QPushButton对象等）
- QWidget对象其内部会有一个**event(QEVent*)函数**被调用，进行事件处理
- event()**根据事件类型**调用不同的**事件处理函数**（默认的子函数）
- 在事件处理函数中发送Qt中预定义的信号
- 调用信号关联的槽函数

（4） QPushButton事件处理分析



①接收到鼠标事件

②QApplication调用QObject::event(QEvent*)成员函数来处理，进行事件的分派。

③调用QPushButton的mouseReleaseEvent(QMouseEvent*)成员函数

④QPushButton调用click()成员函数

⑤触发信号SIGNAL(clicked())

（5）实例

实例一：自定义事件处理函数

鼠标左键右键

wigth.h

```
#ifndef WIDGET_H
#define WIDGET_H

#include <QWidget>

class Widget : public QWidget
{
     Q_OBJECT
protected:
    void mousePressEvent(QMouseEvent *event);
public:
     Widget(QWidget *parent = 0);

     ~Widget();
 };

 #endif // WIDGET_H
```

widgth.cpp

```
#include "widget.h"
#include <QMouseEvent>
#include <QDebug>
#include <QMenu>

Widget::Widget(QWidget *parent)  : QWidget(parent)
{
}

void Widget::mousePressEvent(QMouseEvent *event)
{
     if(event->button()==Qt::LeftButton)
     {
         qDebug()<<"LeftButton clicked!";
     }
     else if(event->button()==Qt::RightButton)
     {
         qDebug()<<"RightButton clicked!";
     }
}

Widget::~Widget()
{
```

```
}
```

main.cpp

```
#include <QApplication>
#include "widget.h"

int main(int argc, char *argv[])
{
     QApplication a(argc, argv);
     Widget w;
     w.show();

     return a.exec();
 }
```

实例二：自定义事件处理函数

QMyPushButton.h

```
#ifndef _QMYPUSHBUTTON_H_
#define _QMYPUSHBUTTON_H_

#include <QPushButton>

typedef void (QButtonListener)(QObject*,QMouseEvent*);

class QMyPushButton : public QPushButton
{
     Q_OBJECT

protected:
     QButtonListener* m_listener;

     //重写QPushButton的事件处理函数 就有可能不会产生clicked信号
     void mouseReleaseEvent(QMouseEvent *e);
public:
     explicit QMyPushButton(QWidget* parent = 0,  QButtonListener* listener = 0);
};

#endif // _QMYPUSHBUTTON_H_
```

```
//QMyPushButton.cpp

 #include "QMyPushButton.h"
 #include <QMouseEvent>

 QMyPushButton::QMyPushButton(QWidget* parent, QButtonListener*
listener):QPushButton(parent)
 {
      m_listener = listener;
 }

 //重写改写事件处理函数，会改变程序的行为。
void QMyPushButton::mouseReleaseEvent(QMouseEvent *e)
```

```
{
     if(m_listener != NULL)
     {
          //调用自定义的事件处理函数，尽管按钮的clicked信号被连接到onMyButtonClicked槽函数，
          //但因自定义的m_listener函数里并不触发clicked信号，从而槽函数不会被调用。
          m_listener(this, e);
          e->accept();//事件被接收，就不再传递到父QWidget

          setDown(false); //按钮设置为“弹起”状态
     }
     else
     {
          //父类的mouseReleaseEvent会去调用clicked()，并触发SIGNAL(clicked())
          //从而调用到连接到该信号的槽函数（本例为onMyButtonClicked())
          QPushButton::mouseReleaseEvent(e); //调用父类
     }
 }
```

Widget.h

```
 #ifndef _WIDGET_H_
 #define _WIDGET_H_

 #include <QWidget>
 #include "QMyPushButton.h"

 class Widget : public QWidget
 {
     Q_OBJECT
     QMyPushButton myButton;

protected slots:
     void onMyButtonClicked();

public:
     Widget(QWidget *parent = 0);
     ~Widget();
};

#endif // _WIDGET_H_
```

Widget.cpp

```
 #include "Widget.h"
 #include <qDebug>

 //自定义事件处理函数
 void onMyButtonMouseRelease(QObject* sender, QMouseEvent* e)
 {
      qDebug() << "onMyButtonMouseRelease(QObject* sender, QMouseEvent* e)";
 }

Widget::Widget(QWidget *parent)
     : QWidget(parent),myButton(this, onMyButtonMouseRelease) //实验2：myButton(this,
0)
{
     myButton.setText("QMyPushButton");
```

```
    connect(&myButton, SIGNAL(clicked()), this, SLOT(onMyButtonClicked()));
}

//槽函数，用于接收按钮的clicked信号
void Widget::onMyButtonClicked()
{
    qDebug() << "onMyButtonClicked()" ;
}

Widget::~Widget()
{
}
```

main.cpp

```
#include "Widget.h"
#include <QApplication>

int main(int argc, char *argv[])
{
        QApplication a(argc, argv);
        Widget w;
        w.show();
   return a.exec();
}
```

（4）事件（QEvent）和信号（SIGNAL）的不同

	事件（QEvent）	信号（SIGNAL）
与QObject的关系	由具体对象**进行处理**	由具体对象**主动产生**
对程序影响	**改写事件处理函数可能导致**程序行为发生改变	信号**是否存在对应的槽函数不会改变**程序行为
两者的联系	一般而言，信号在具体的事件处理函数中产生	

例如：单击界面上的按钮，那么就会产生鼠标事件QMouseEvent（不是按钮产生的），由于按钮被按下了，所以他会发出一个单击信号clicked()信号（是按钮产生的），这里只考虑单击信号而不用考虑鼠标事件，但如果要设计一个按钮，或者当单击按钮时让它产生别的效果，此时就要考虑鼠标事件了，由此，信号和事件是两个不同层面的东西，发出者不同，作用不同。Qt中，所有的QObject的子类实例均可对事件接收和处理！

3．小结

（1）Qt中的**事件和信号**不同

（2）事件由QObject对象进行处理

（3）信号由QObject对象触发

（4）**重写**事件处理函数**可能改变程序行为**

（5）**信号的触发不会**对程序行为造成影响

（6）事件处理是在**实际工程开发中**应用非常普遍的