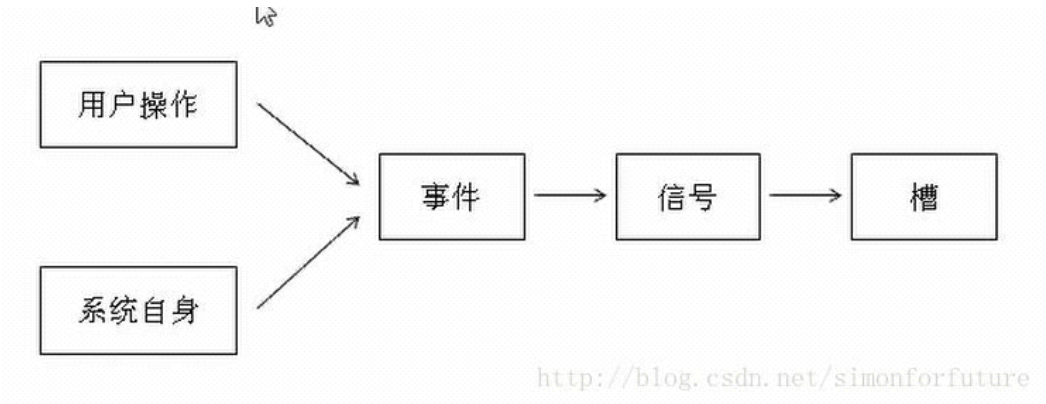
2019年4月7日 星期日　　22:12

Qt事件与信号(一)——重新实现事件处理器

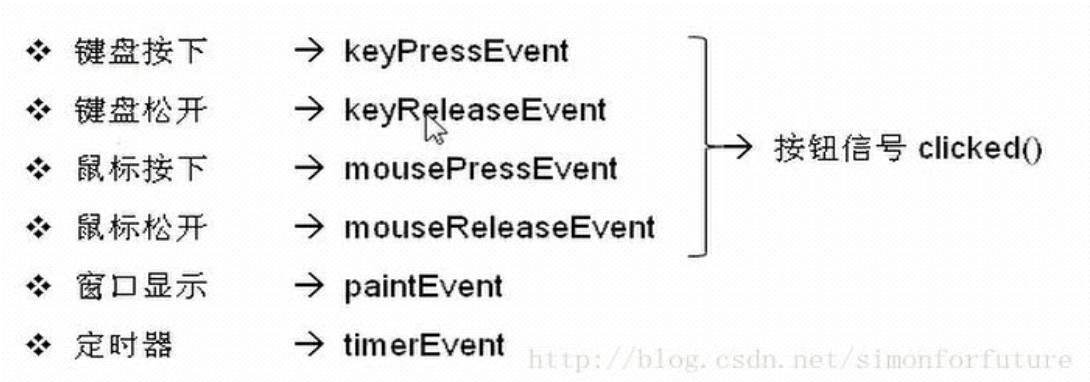
- 事件和信号
- 使用信号与槽的方式
- 重新实现事件处理器
- 重新实现paintEvent函数对事件进行处理
-

事件和信号

在Qt中，事件就是对象，派生自QEvent抽象类，用来表示在应用程序中发生的事件，或是应用程序需要处理的外部活动产生的事件。也就是用户操作可以产生事件，系统自身也可以产生事件。



事件如下：



有一些事件对应有信号，可以用信号与槽的方式处理，也可以直接处理事件。但是有些事件。比如：`paintEvent`和`timerEvent`，我们就只能自己编程进行处理了。

使用信号与槽的方式

从上图可以知道，鼠标、键盘的按下都可以看作是发出`clicked`信号，所以我们可以对这类事件采用信号与槽的方式进行处理。

- main.cpp

```
#include "dialog.h"
#include <QApplication>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Dialog w;
    w.show();

return a.exec();
}
```

- dialog.h

```
#ifndef DIALOG_H
#define DIALOG_H
#include <QDialog>
#include "sibutton.h"

namespace Ui {
class Dialog;
}

class Dialog : public QDialog
{
    Q_OBJECT

public:
    explicit Dialog(QWidget *parent = 0);
    ~Dialog();

private:
    Ui::Dialog *ui;
    siButton* sibutton;
private slots:
    void siButtonClicked();
};

#endif // DIALOG_H
```

- dialog.cpp

```
#include "dialog.h"
#include "ui_dialog.h"
#include <QtWidgets>

Dialog::Dialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Dialog)
{
    ui->setupUi(this);
```

```
    sibutton = new siButton;
    sibutton->setText(tr("Hello"));

connect(sibutton, SIGNAL(clicked()), this, SLOT(siButtonClicked()));

ui->horizontalLayout->addWidget(sibutton);
}

Dialog::~Dialog()
{
    delete ui;
}

void Dialog::siButtonClicked()
{
    QMessageBox::information(this, tr("Test"), tr("Hello Signal and
slots"));
}
```

使用信号与槽的方式，是在**界面**中对程序进行处理，也就是button只是作为一个对象，点击的事件应该是被界面接收，然后在界面中进行处理。

重新实现事件处理器

- sibutton.h

```
#ifndef SIBUTTON_H
#define SIBUTTON_H
#include <QPushButton>

class siButton : public QPushButton
{
public:
    siButton();
protected:
    void keyPressEvent(QKeyEvent*);
};

#endif // SIBUTTON_H
```

- sibutton.cpp

```
#include "sibutton.h"
#include <QtWidgets>
siButton::siButton()
{

}

void siButton::keyPressEvent(QKeyEvent *)
{
    QMessageBox::information(this, tr("Test"), tr("Hello,
keyPressEvent"));
```

```
}
```

使用事件处理的方式，是在继承`QPushButton`的文件中对点击事件进行处理，我认为也就是在`siButton`这个控件中对键盘按下事件进行处理。

重新实现`paintEvent`函数对事件进行处理

- dialog2.h

```
#ifndef DIALOG2_H
#define DIALOG2_H
#include <QDialog>
#include "siWidget.h"

namespace Ui {
class Dialog2;
}

class Dialog2 : public QDialog
{
    Q_OBJECT

public:
    explicit Dialog2(QWidget *parent = 0);
    ~Dialog2();

private:
    Ui::Dialog2 *ui;

private:
    siWidget *siwidget;

};

#endif // DIALOG2_H
```

- dialog2.cpp

```
#include "dialog2.h"
#include "ui_dialog2.h"
Dialog2::Dialog2(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Dialog2)
{
    ui->setupUi(this);

    siwidget = new siWidget;
    ui->horizontalLayout->addWidget(siwidget);

this->resize(800, 600);
}

Dialog2::~Dialog2()
```

```
{
    delete ui;
}
```

- siwidget.h

```
#ifndef SIWIDGET_H
#define SIWIDGET_H
#include <QWidget>
#include <QPaintEvent>

class siWidget : public QWidget
{
    Q_OBJECT

public:
    explicit siWidget(QWidget *parent = 0);

signals:

public slots:

protected:
    void paintEvent(QPaintEvent*);

};

#endif // SIWIDGET_H
```

- siwidget.cpp

```
#include "siwidget.h"
#include <QtWidgets>
siWidget::siWidget(QWidget *parent):
    QWidget(parent)
{
    update();
}

void siWidget::paintEvent(QPaintEvent* )
{
    QPainter painter(this);

    QPixmap pix;
    pix.load("images/0.bmp");  // note: this should be with elf file

    painter.drawPixmap(0, 0, pix);

}
```