1 事件处理过程

2 鼠标事件

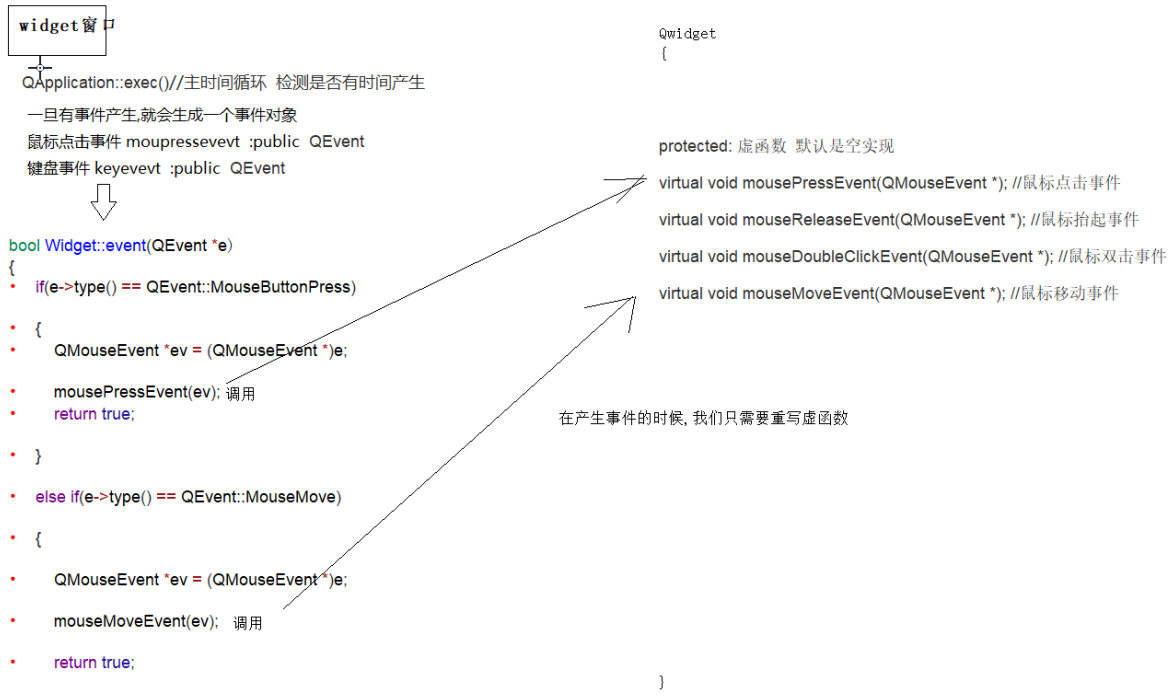
3 滚轮事件

4 键盘事件(QKeyEvent)

5 大小改变事件(QResizeEvent)

6 进入离开事件

1 事件处理过程



事件（event）是由系统或者 Qt 本身在不同的时刻发出的。当用户按下鼠标、敲下键盘，或者是窗口需要重新绘制的时候，都会发出一个相应的事件。

一些事件在对用户操作做出响应时发出，如键盘事件等；另一些事件则是由系统自动发出，如计时器事件。

事件处理过程：

1) 在Qt内部，Qt通过 QApplication::exec()启动的主事件循环不停的抓取事件队列中的事件。

2) 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。

3) 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，而是按照事件对象的类型分派给特定的事件处理函数（event handler）。

event()函数主要用于事件的分发：

```
bool Widget::event(QEvent *e)
{
    if(e->type() == QEvent::MouseButtonPress)
    {
        QMouseEvent *ev = (QMouseEvent *)e;
        mousePressEvent(ev);
        return true;
    }
    else if(e->type() == QEvent::MouseMove)
    {
        QMouseEvent *ev = (QMouseEvent *)e;
        mouseMoveEvent(ev);
        return true;
    }
    else if(e->type() == QEvent::MouseButtonDblClick)
    {
        QMouseEvent *ev = (QMouseEvent *)e;
        mouseDoubleClickEvent(ev);
        return true;
```

}

- return QWidget::*event*(e);

}

2 鼠标事件

```
鼠标事件(QMouseEvent)
//重写父类的虚函数
protected:
virtual void mousePressEvent(QMouseEvent *); //鼠标点击事件
virtual void mouseReleaseEvent(QMouseEvent *); //鼠标抬起事件
virtual void mouseDoubleClickEvent(QMouseEvent *); //鼠标双击事件
virtual void mouseMoveEvent(QMouseEvent *); //鼠标移动事件
```

.h

```
#ifndef WIDGET_H
#define WIDGET_H

#include <QWidget>

namespace Ui {
class Widget;
}

 class Widget : public QWidget
 {
  Q_OBJECT

 public:
  explicit Widget(QWidget *parent = 0);
  ~Widget();
 protected:
  void mousePressEvent(QMouseEvent *event);
  void mouseMoveEvent(QMouseEvent *event);

 private:
  Ui::Widget *ui;
```

```
};

#endif // WIDGET_H

```

.cpp

```
#include "widget.h"
#include "ui_widget.h"
#include <QMouseEvent>
#include <QDebug>
Widget::Widget(QWidget *parent) :
 QWidget(parent),
 ui(new Ui::Widget)
{
 ui->setupUi(this);
 }

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

  void Widget::mousePressEvent(QMouseEvent *event)
  {
  qDebug()<<"鼠标点击"<<event->x()<<event->y();
  if(event->button() ==Qt::LeftButton )
  {
  qDebug()<<"点击了左键";

  }
  else if(event->button() ==Qt::RightButton )
  {
  qDebug()<<"点击了右键";

  }

  }

  void Widget:: mouseMoveEvent(QMouseEvent *event)
  {

```

```
36  qDebug()<<"鼠标点击"<<event->x()<<event->y();
37
38  }
39
```

3 滚轮事件

```
1  void Widget:: mouseMoveEvent(QMouseEvent *event)
2  {
3
4  // qDebug()<<"鼠标点击"<<event->x()<<event->y();
5
6  }
7
8  void Widget::wheelEvent(QWheelEvent *event)
9  {
10  if(event->orientation()== Qt::Vertical) //如果方向是垂直
11  {
12  qDebug()<<"滚轮"<<event->x()<<event->y();
13  }
14
15
16
17  }
```

4 键盘事件(QKeyEvent)

参考代码：[code\day03\03_QKeyEvent]

//virtual void keyPressEvent(QKeyEvent *); //键盘按下事件

//virtual void keyReleaseEvent(QKeyEvent *); //键盘抬起事件

//键盘按下事件

void Widget::keyPressEvent(QKeyEvent *e)

{

QString str;

switch(e->modifiers()) //修饰键盘

{

case Qt::ControlModifier:

str = "Ctrl+";

break;

```
case Qt::AltModifier:
str = "Alt+";
break;
}
switch(e->key()) //普通键
{
case Qt::Key_Left:
str += "Left_Key Press";
break;
case Qt::Key_Right:
str += "Rigth_Key Press";
break;
case Qt::Key_Up:
str += "Up_Key Press";
break;
case Qt::Key_Down:
str += "Down_Key Press";
break;
case Qt::Key_Z:
str += "Z_Key Press";
break;
}
ui->label->setText(str);
}
```

5 大小改变事件(QResizeEvent)

当窗口大小发生变化时被调用，参考代码：

```
//virtual void resizeEvent(QResizeEvent *); //大小改变事件
//大小改变事件
void Widget::resizeEvent(QResizeEvent *e)
{
//变化前的窗口大小
qDebug() << "e->oldSize() = " << e->oldSize();
//变化后的窗口大小
```

```
qDebug() << "e->size() = " << e->size();
}
```

6 进入离开事件

```
/virtual void enterEvent(QEvent *); //进入事件
//virtual void leaveEvent(QEvent *); //离开事件
//进入事件
void Widget::enterEvent(QEvent *)
{
qDebug() << "enterEvent";
}
//离开事件
void Widget::leaveEvent(QEvent *)
{
qDebug() << "leaveEvent";
}
```