QT事件过滤器、事件重载和焦点事件

2019年4月8日 星期一　　21:57

常见的QT事件有很多，例如：

键盘事件: 按键按下和松开；

鼠标事件: 鼠标移动,鼠标按键的按下和松开.，双击；

拖放事件: 用鼠标进行拖放；滚轮事件: 鼠标滚轮滚动；

绘屏事件: 重绘屏幕的某些部分；

定时事件: 定时器到时触发；

焦点事件: 键盘焦点移动；

进入和离开事件: 鼠标移入widget之内,或是移出；

移动事件: widget的位置改变；

大小改变事件: widget的大小改变；

显示和隐藏事件: widget显示和隐藏；

窗口事件: 窗口是否为当前窗口；

socket事件,剪贴板事件,字体改变,布局改变等等.

这里主要通过三个示例来描述事件过滤器、事件重载和焦点事件，也是经常会用到的东西。

1、事件过滤器

QT提供了事件过滤器来在一个部件中监控其他多个部件的事件。事件过滤器与其他部件不同，它不是一个类，只是由两个函数组成的一种操作，来完成一个部件对其他部件的事件的监视。这两个函数是installEventFilter()函数和eventFilter()函数，都是Object类中的函数。

示例：

.h文件添加代码：

```
bool eventFilter(QObject *, QEvent *);
```

.cpp文件添加代码：

在构造函数中添加代码：

```
ui->register_lbl->installEventFilter(this);     //为register_lbl在窗口上安装过滤器
```

下面实现eventFilter()函数，这里实现了通过点击一个label来弹出相应的窗口：

```
/*使用事件过滤器来打开注册窗口*/
```

```
bool login::eventFilter(QObject *obj, QEvent *event)
{
    if(obj == ui->register_lbl)           // 判断部件
    {
        if(event->type() == QEvent::MouseButtonPress)          // 判断事件
        {
            Register * reg = new Register;
            this->close();
            reg->setModal(true);
            reg->show();
            // ui->userIDLineEdit->setFocus();
            return true;          // 该事件已经被处理
        }
        else
        {
            return false;         // 如果是其他事件可以进一步处理
        }
    }
    else
        return login::eventFilter(obj, event);       // 将事件交给上层对话框
}
```

2、重载event()函数

通过重载event()函数,我们可以在事件被特定的事件处理函数处理之前(象keyPressEvent())处理它. 比如, 当我们想改变tab键的默认动作时,一般要重载这个函数. 在处理一些不常见的事件(比如:LayoutDirectionChange)时,evnet()也很有用,因为这些函数没有相应的特定事件处理函数. 当我们重载event()函数时, 需要调用父类的event()函数来处理我们不需要处理或是不清楚如何处理的事件。

下面这个示例重载了event()函数,，实现改变Tab键的默认动作，默认的是键盘焦点移动到下一个控件上，代码如下：

```
bool CodeEditor::event(QEvent * event)
{
    if (event->type() == QEvent::KeyPress)     //如果是键盘按下
    {
        QKeyEvent *keyEvent = (QKeyEvent *) event;
        if (keyEvent->key() == Key_Tab)        //是否按下Tab键
        {
            insertAtCurrentPosition( '\t' );       //do somthing
            return true;
        }
    }
    return QWidget::event(event);
}
```

3、获得和失去焦点事件

我们可以通过使用事件过滤器来实现获得和失去焦点事件。Qt的事件模型中提供的事件过滤功能使得一个QObject对象可以监视另一个QObject对象中的事件，通过在一个QObject对象中安装事件过滤器可以在事件到达该对象前捕获事件，从而起到监视该对象事件的效果。实现类似功能的另一种方式是通过分别继承不同的控件类，并重构各控件的事件响应函数，但若窗体中包含大量不同的控件时，每一个控件都必须重新继承，然后分别重构不同的事件函数，实现比较复杂。事件过滤器可以实现在窗体中监视全部控件的不同事件，方便实现功能扩展。

下面这个示例实现了lineEdit获得和失去焦点时颜色会有相应的变化：

.h文件中添加代码：

```
bool eventFilter(QObject *,QEvent *);
```

.cpp文件中添加代码：

在构造函数中添加代码：

```
ui->lineEdit->installEventFilter(this);   //在窗体上为lineEdit安装过滤器
```

下面实现焦点事件，代码如下：

```
bool Widget::eventFilter(QObject *watched, QEvent *event)
{
    if (watched == ui->lineEdit)           //首先判断控件(这里指 lineEdit)
    {
        if (event->type()==QEvent::FocusIn)      //然后再判断控件的具体事件 (这里指获得焦点事件)
        {
            QPalette p=QPalette();
            p.setColor(QPalette::Base,Qt::green);
            ui->lineEdit->setPalette(p);
        }
        else if (event->type()==QEvent::FocusOut)     // 这里指 lineEdit 控件的失去焦点事件
        {
            QPalette p = QPalette();
            p.setColor(QPalette::Base,Qt::white);
            ui->lineEdit->setPalette(p);
        }
    }
    return QWidget::eventFilter(watched, event);      // 最后将事件交给上层对话框
}
```