<http://blog.csdn.net/qq_19528953/article/details/52577328>

今天在进行Qt编程的时候，学习了一下如何自定义ui的样式，有时候，你在修改别人程序的时候，可能会发现别人程序中的某个控件ui设计是直接拖上去的。这种情况下如果你想增加该控件的事件处理，比如拖拽功能。不太好办，今天在书上找到一个方法，就是使用Qt设计师里面的提升，将需要增加事件响应的那个控件提升到一个类里面，然后自己实现里面相关的事件响应函数，今天我成功实现了一个QTreeWidget的提升，然后把里面的数据实现了拖拽功能，代码如下：

#ifndef NEW\_QTREEWIDGET\_H

#define NEW\_QTREEWIDGET\_H

#include <QTreeWidget>

#include <QDragEnterEvent>

class new\_QTreeWidget:public QTreeWidget

{

Q\_OBJECT

public:

new\_QTreeWidget(QWidget \*parent = 0);

protected:

void dragEnterEvent(QDragEnterEvent \*e);

void dragMoveEvent(QDragMoveEvent \*e);

void dropEvent(QDropEvent \*e);

void mousePressEvent(QMouseEvent \*e);

void mouseMoveEvent(QMouseEvent \*e);

private:

QPoint \_startPos;

void performDrag();

};

#endif // NEW\_QTREEWIDGET\_H

#include "new\_qtreewidget.h"

#include <QMimeData>

#include <QDragEnterEvent>

#include <QDragMoveEvent>

#include <QDropEvent>

#include <QMouseEvent>

#include <QApplication>

#include <QDrag>

#include <QDebug>

new\_QTreeWidget::new\_QTreeWidget(QWidget \*parent):QTreeWidget(parent)

{

setAcceptDrops(true);

}

void new\_QTreeWidget::dragEnterEvent(QDragEnterEvent \*e)

{

new\_QTreeWidget \*source = qobject\_cast<new\_QTreeWidget \*>(e->source());

if(source)

{

e->setDropAction(Qt::MoveAction);

e->accept();

}

}

void new\_QTreeWidget::dragMoveEvent(QDragMoveEvent \*e)

{

new\_QTreeWidget \*source = qobject\_cast<new\_QTreeWidget \*>(e->source());

if(source)

{

e->setDropAction(Qt::MoveAction);

e->accept();

}

}

void new\_QTreeWidget::dropEvent(QDropEvent \*e)

{

new\_QTreeWidget \*source = qobject\_cast<new\_QTreeWidget \*>(e->source());

if(source)

{

new\_QTreeWidget \*source = qobject\_cast<new\_QTreeWidget \*>(e->source());

if(source)

{

QTreeWidgetItem \*item = this->itemAt(e->pos()); //当前位置的item

if( item == nullptr) //如果放下的位置没有item，则退出，没有这句话死机！

return;

if( -1 == this->indexOfTopLevelItem(item) && (-1 == this->indexOfTopLevelItem(currentItem())) ||

-1 != this->indexOfTopLevelItem(item) && (-1 != this->indexOfTopLevelItem(currentItem()))

) //如果“放下位置的item是顶层item，且原来的是顶层”或者“放下的不是顶层，且原来也不是顶层”

{

qDebug()<<"放下的文本是："<<e->mimeData()->text();

item->setText(currentColumn(),e->mimeData()->text());

e->setDropAction(Qt::MoveAction);

e->accept();

}

}

}

}

void new\_QTreeWidget::mousePressEvent(QMouseEvent \*e)

{

if(e->button() == Qt::LeftButton)

{

\_startPos = e->pos();

}

QTreeWidget::mousePressEvent(e);

}

void new\_QTreeWidget::mouseMoveEvent(QMouseEvent \*e)

{

if(e->buttons() & Qt::LeftButton)

{

int distance = (e->pos() - \_startPos).manhattanLength();

if(distance >= QApplication::startDragDistance()) //当拖动距离大于一个推荐抖动距离时，表示同意已经拖动操作了

performDrag();

}

QTreeWidget::mouseMoveEvent(e);

}

void new\_QTreeWidget::performDrag()

{

QTreeWidgetItem \*item = currentItem();

int column = currentColumn();

if(item) //必须是非顶层item才可以移动数据

// if(item)

{

QMimeData \*mineData = new QMimeData;

if(column != 4) //只有第四列才可以移动数据

return;

mineData->setText(item->text(column));

qDebug()<<item->text(column);

QDrag \*drag = new QDrag(this);

drag->setMimeData(mineData);

drag->exec(Qt::MoveAction);

}

}

这样就解决了当界面已经设置好后如何添加功能的难题！