[【实例】Qt之文本编辑（一）](http://blog.chinaunix.net/uid-25806493-id-3346294.html)

分类： C/C++

2012-09-12 18:29:10

**Qt之文本编辑（一）**

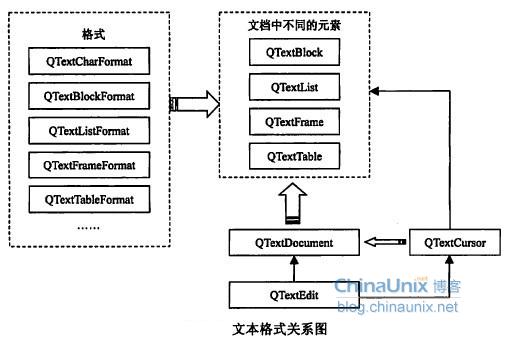
|  |  |
| --- | --- |
| 文档名称 | Qt之文本编辑（一） |
| 创建时间 | 2012-9-12 |
| 修改时间 | 2012-9-12 |
| 创建人 | Baifx |
| 简介（收获） | 设置字体字号、文本排序对齐 |

一、设置字体、字号等格式属性。

       【应用场景描述】在编辑框中输入一段文字，用鼠标选取文字，修改工具栏上的字体、字号大小、加粗、斜体等属性，选取的文字即发生相应的变化。

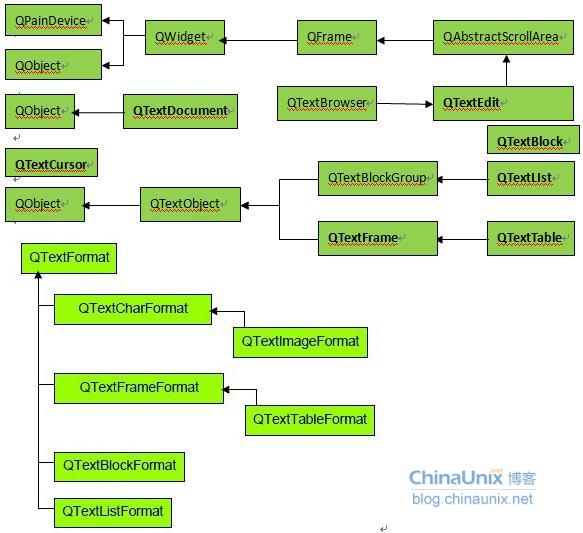
       【基本概念】在编写包含格式设置的文本编辑程序时，经常用到的Qt类有：QTextEdit、QTextDocument、QTextCharFormat、QTextCursor、QTextBlock、QTextList、QTextFrame、QTextTable、QTextBlockFormat、QTextListFormat、QTextFrameFormat、QTextTableFormat等。刚看到如此多的相关类可能会感到有些混乱，但只要弄清了它们之间的关系，运用起来就会非常方便，Qt已经为用户完成了几乎所有与编辑有关的具体工作，我们所要做的就是运用合适的类，调用合适的函数接口。

       首先，任何一个文本编辑器的程序都要用到QTextEdit作为输入文本的容器，在它里面输入的可编辑文本由QTextDocument作为载体，而QTextBlock、QTextList、QTextFrame等则用来表示QTextDocument的元素，也可理解为QTextDocument的不同表现形式，可能为字符串、段落、列表、表格或是图片等。每种元素都有自己的格式，这些格式则用QTextCharFormat、QTextBlockFormat、QTextListFormat、QTextFrameFormat等类来描述与实现。例如，QTextBlockFormat类对应于QTextBlock类，QTextBlock类用于表示一块文本，一般可以理解为一个段落，但并不只指段落，QTextBlockFormat类则用于表示这一块文本的格式，如缩进值、与四边的边距等。各类之间的划分与关系可用下图进行描述：

[](http://blog.chinaunix.net/attachment/201209/12/25806493_1347445833iK9u.jpg)

       从上图可知，QTextCursor类是一个非常重要也经常会用到的类，它用于表示编辑文本中的光标。这个类提供了对QTextDocument文档的修改接口，所有对文档格式的修改，说到底都与光标有关，如改变字符的格式，指的是改变光标处字符的格式；改变段落的格式，指的是改变光标所在段落的格式，因此，QTextCursor类在文档编辑类程序中有着重要的作用，所有对QTextDocument的修改能够通过QTextCursor类实现。

       【继承关系】

[](http://blog.chinaunix.net/attachment/201209/12/25806493_134744589259n9.jpg)

   【实例】

[](http://blog.chinaunix.net/attachment/201209/12/25806493_1347445976y5LW.jpg)

fontset.h代码：

class FontSet : public QMainWindow

{

       Q\_OBJECT

public:

       FontSet(QWidget \*parent = 0, Qt::WFlags flags = 0);

       ~FontSet();

protected slots:

       void slotFont( QString f );

       void slotSize( QString num );

       void slotBold();

       void slotItalic();

       void slotUnder();

       void slotColor();

       void slotCurrentFormatChanged( const QTextCharFormat & fmt );

private:

       void mergeFormat( QTextCharFormat fmt );

private:

       QLabel \* pLabel1;

       QLabel \* pLabel2;

       QFontComboBox \* pFontBox;

       QComboBox \* pSizeBox;

       QToolButton \* pBoldBtn;

       QToolButton \* pItalicBtn;

       QToolButton \* pUnderBtn;

       QToolButton \* pColorBtn;

       QTextEdit \* pText;

};

fontset.cpp代码：

FontSet::FontSet(QWidget \*parent, Qt::WFlags flags)

       : QMainWindow(parent, flags)

{

       setWindowTitle( tr( "Font" ) );

       pText = new QTextEdit( tr( "ajglkajdkgnakjfdsklaj" ), this );

       setCentralWidget( pText );

       QToolBar \* pToolBar = addToolBar( "Font" );

       //字体

       pLabel1 = new QLabel( tr( "ZiTi:" ) );

       pFontBox = new QFontComboBox();

       pFontBox->setFontFilters( QFontComboBox::ScalableFonts );

       pToolBar->addWidget( pLabel1 );

       pToolBar->addWidget( pFontBox );

       //字号

       pLabel2 = new QLabel( tr( "ZiHao:" ) );

       pSizeBox = new QComboBox();

       pToolBar->addWidget( pLabel2 );

       pToolBar->addWidget( pSizeBox );

       QFontDatabase db;

       foreach( int nSize, db.standardSizes() )

              pSizeBox->addItem( QString::number( nSize ) );

       pToolBar->addSeparator();

       //加粗、斜体、下划线、颜色

       pBoldBtn = new QToolButton();

       pBoldBtn->setIcon( QIcon( ".//Resources//bold.png" ) );

       pBoldBtn->setCheckable( true );

       pToolBar->addWidget( pBoldBtn );

       pItalicBtn = new QToolButton();

       pItalicBtn->setIcon( QIcon( ".//Resources//italic.png" ) );

       pItalicBtn->setCheckable( true );

       pToolBar->addWidget( pItalicBtn );

       pUnderBtn = new QToolButton();

       pUnderBtn->setIcon( QIcon( ".//Resources//underline.png" ) );

       pUnderBtn->setCheckable( true );

       pToolBar->addWidget( pUnderBtn );

       pToolBar->addSeparator();

       pColorBtn = new QToolButton();

       pColorBtn->setIcon( QIcon( ".//Resources//color.png" ) );

       pToolBar->addWidget( pColorBtn );

       //连接信号与槽函数

       connect( pFontBox, SIGNAL( activated( QString ) ), this, SLOT( slotFont( QString ) ) );

       connect( pSizeBox, SIGNAL( activated( QString ) ), this, SLOT( slotSize( QString ) ) );

       connect( pBoldBtn, SIGNAL( clicked() ), this, SLOT( slotBold() ) );

       connect( pItalicBtn, SIGNAL( clicked() ), this, SLOT( slotItalic() ) );

       connect( pUnderBtn, SIGNAL( clicked() ), this, SLOT( slotUnder() ) );

       connect( pColorBtn, SIGNAL( clicked() ), this, SLOT( slotColor() ) );

       connect( pText, SIGNAL( currentCharFormatChanged( const QTextCharFormat & ) ), this, SLOT( slotCurrentFormatChanged( const QTextCharFormat& ) ) );

}

FontSet::~FontSet()

{

}

void FontSet::slotFont( QString f )

{

       QTextCharFormat fmt;

       fmt.setFontFamily( f );

       mergeFormat( fmt );

}

void FontSet::slotSize( QString num )

{

       QTextCharFormat fmt;

       fmt.setFontPointSize( num.toFloat() );

       mergeFormat( fmt );

}

void FontSet::slotBold()

{

       QTextCharFormat fmt;

       fmt.setFontWeight( pBoldBtn->isChecked() ? QFont::Bold : QFont::Normal );

       mergeFormat( fmt );

       //pText->mergeCurrentCharFormat( fmt );

}

void FontSet::slotItalic()

{

       QTextCharFormat fmt;

       fmt.setFontItalic( pItalicBtn->isChecked() );

       mergeFormat( fmt );

}

void FontSet::slotUnder()

{

       QTextCharFormat fmt;

       fmt.setFontUnderline( pUnderBtn->isChecked() );

       mergeFormat( fmt );

}

void FontSet::slotColor()

{

       QColor color = QColorDialog::getColor( Qt::red, this );

       if ( color.isValid() )

       {

              QTextCharFormat fmt;

              fmt.setForeground( color );

              mergeFormat( fmt );

       }

}

/\*当光标所在处的字符格式发生变化时调用，函数根据新的字符格式把工具栏上的各个格式控件的显示更新\*/

void FontSet::slotCurrentFormatChanged( const QTextCharFormat & fmt )

{

       pFontBox->setCurrentIndex( pFontBox->findText( fmt.fontFamily() ) );

       pSizeBox->setCurrentIndex( pSizeBox->findText( QString::number( fmt.fontPointSize() ) ) );

       pBoldBtn->setChecked( fmt.font().bold() );

       pItalicBtn->setChecked( fmt.fontItalic() );

       pUnderBtn->setChecked( fmt.fontUnderline() );

}

/\*设置光标的选区，使格式作用于选区内的字符，若没有选区则作用于光标所在处的字符\*/

void FontSet::mergeFormat( QTextCharFormat fmt )

{

       QTextCursor cursor = pText->textCursor();

       if ( !cursor.hasSelection() )

              cursor.select( QTextCursor::WordUnderCursor );

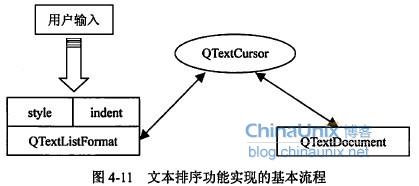
       cursor.mergeCharFormat( fmt );

       pText->mergeCurrentCharFormat( fmt );

}

       二、设置文本排序及对齐

       QTextListFormat主要用于描述文本排序的格式，它主要包括两个基本属性，一个味QTextListFormat::style，表示文本采用哪种排序方式；另一种为QTextListFormat::indent，表示排序后的缩进值。因此，要实现文本排序的功能只需设置好QTextListFormat的这两个属性，并把整个格式通过QTextCursor类应用到文本中即可。

[](http://blog.chinaunix.net/attachment/201209/12/25806493_1347445922865E.jpg)

    【实例】

[](http://blog.chinaunix.net/attachment/201209/12/25806493_13474459536SC7.jpg)

listalign.h代码：

class listAlign : public QMainWindow

{

       Q\_OBJECT

public:

       listAlign(QWidget \*parent = 0, Qt::WFlags flags = 0);

       ~listAlign();

protected slots:

       void slotAlignment( QAction \* );

       void slotList( int );

       void slotCursorPositionChanged();

private:

       QTextEdit \* pText;

       QLabel \* pLabel;

       QComboBox \* pListBox;

       QAction \* pLeftAction;

       QAction \* pRightAction;

       QAction \* pCenterAction;

       QAction \* pJustifyAction;

       QAction \* pRedoAction;

       QAction \* pUndoAction;

};

listalign.cpp代码：

listAlign::listAlign(QWidget \*parent, Qt::WFlags flags)

       : QMainWindow(parent, flags)

{

       QFont f( "ZYSong18030", 12 );

       setFont( f );

       setWindowTitle( tr( "List&Alignment" ) );

       QToolBar \* pToolBar = addToolBar( "List" );

       pLabel = new QLabel( tr( "List:" ) );

       pListBox = new QComboBox( pToolBar );

       pListBox->addItem( tr( "Standard" ) );

       pListBox->addItem( tr( "Bullet List (Disc)" ) );

       pListBox->addItem( tr( "Bullet List (Circle)" ) );

       pListBox->addItem( tr( "Bullet List (Square)" ) );

       pListBox->addItem( tr( "Ordered List (Decimal)" ) );

       pListBox->addItem( tr( "Ordered List (Alpha lower)" ) );

       pListBox->addItem( tr( "Ordered List (Alpha upper)" ) );

       pToolBar->addWidget( pLabel );

       pToolBar->addWidget( pListBox );

       pToolBar->addSeparator();

       QActionGroup \* pActGrp = new QActionGroup( this );

       pLeftAction = new QAction( QIcon( ".//Resources//left.png" ), tr( "left" ), pActGrp );

       pLeftAction->setCheckable( true );

       pCenterAction = new QAction( QIcon( ".//Resources//center.png" ), tr( "center" ), pActGrp );

       pCenterAction->setCheckable( true );

       pJustifyAction = new QAction( QIcon( ".//Resources//justify.png" ), tr( "justify" ), pActGrp );

       pJustifyAction->setCheckable( true );

       pRightAction = new QAction( QIcon( ".//Resources//right.png" ), tr( "right" ), pActGrp );

       pRightAction->setCheckable( true );

       pToolBar->addActions( pActGrp->actions() );

       QToolBar \* pEditBar = addToolBar( "Edit" );

       pUndoAction = new QAction( QIcon( ".//Resources//undo.png" ), tr( "undo" ), this );

       pEditBar->addAction( pUndoAction );

       pRedoAction = new QAction( QIcon( ".//Resources//redo.png" ), tr( "redo" ), this );

       pEditBar->addAction( pRedoAction );

       pText = new QTextEdit( this );

       pText->setFocus();

       setCentralWidget( pText );

       connect( pListBox, SIGNAL( activated( int) ), this, SLOT( slotList( int ) ) );

       connect( pActGrp, SIGNAL( triggered( QAction \* ) ), this, SLOT( slotAlignment( QAction \* ) ) );

       connect( pRedoAction, SIGNAL( triggered() ), pText, SLOT( redo() ) );

       connect( pUndoAction, SIGNAL( triggered() ), pText, SLOT( undo() ) );

       connect( pText->document(), SIGNAL( redoAvailable( bool ) ), pRedoAction, SLOT( setEnabled( bool ) ) );

       connect( pText->document(), SIGNAL( undoAvailable( bool ) ), pUndoAction, SLOT( setEnabled( bool ) ) );

       connect( pText, SIGNAL( cursorPositionChanged() ), this, SLOT( slotCursorPositionChanged() ) );

}

listAlign::~listAlign()

{

}

void listAlign::slotAlignment( QAction \* act )

{

       if ( act == pLeftAction )

              pText->setAlignment( Qt::AlignLeft );

       if ( act == pCenterAction )

              pText->setAlignment( Qt::AlignCenter );

       if ( act == pJustifyAction )

              pText->setAlignment( Qt::AlignJustify );

       if ( act == pRightAction )

              pText->setAlignment( Qt::AlignRight );

}

void listAlign::slotList( int index )

{

       QTextCursor cursor = pText->textCursor();

       if ( index != 0 )

       {

              QTextListFormat::Style style = QTextListFormat::ListDisc;

              switch ( index )

              {

                     default:

                     case 1:

                            style = QTextListFormat::ListDisc;

                            break;

                     case 2:

                            style = QTextListFormat::ListCircle;

                            break;

                     case 3:

                            style = QTextListFormat::ListSquare;

                            break;

                     case 4:

                            style = QTextListFormat::ListDecimal;

                            break;

                     case 5:

                            style = QTextListFormat::ListLowerAlpha;

                            break;

                     case 6:

                            style = QTextListFormat::ListUpperAlpha;

                            break;

              }

              cursor.beginEditBlock();

              QTextBlockFormat blockFmt = cursor.blockFormat();

              QTextListFormat listFmt;

              if ( cursor.currentList() )

              {

                     listFmt = cursor.currentList()->format();

              }

              else

              {

                     listFmt.setIndent( blockFmt.indent() + 1 );

                     blockFmt.setIndent( 0 );

                     cursor.setBlockFormat( blockFmt );

              }

              listFmt.setStyle( style );

              cursor.createList( listFmt );

              cursor.endEditBlock();

       }

       else

       {

               QTextBlockFormat bfmt;

         bfmt.setObjectIndex( -1 );

         cursor.mergeBlockFormat( bfmt );

       }

}

void listAlign::slotCursorPositionChanged()

{

       if ( pText->alignment() == Qt::AlignLeft)

          pLeftAction->setChecked(true);

    if (pText->alignment() == Qt::AlignCenter)

          pCenterAction->setChecked(true);

    if (pText->alignment() == Qt::AlignJustify)

          pJustifyAction->setChecked(true);

    if (pText->alignment() == Qt::AlignRight)

          pRightAction->setChecked(true);

}