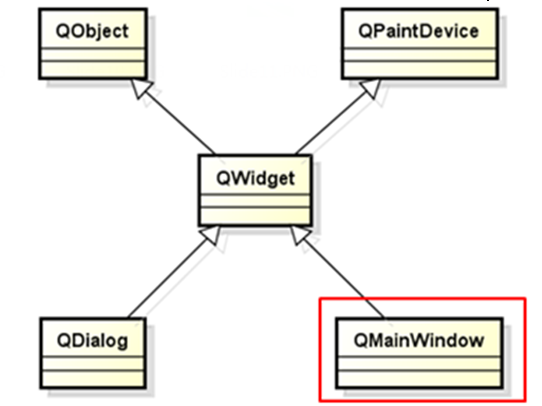
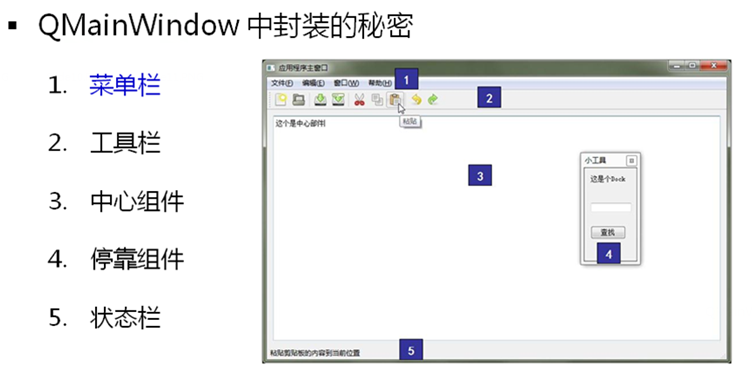
**QMainWindow介绍**

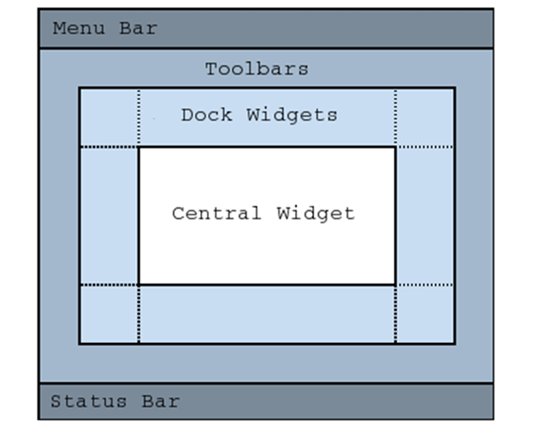
* 主窗口是与用户进行**长时间交互**的顶层窗口,比如记事本
* 主窗口通常是应用程序启动后显示的**第一个窗口**
* QMainWindow是Qt中主窗口的基类,继承于QWidget,如下图所示:



**QMainWindow和其它类不同,如下图所示:**

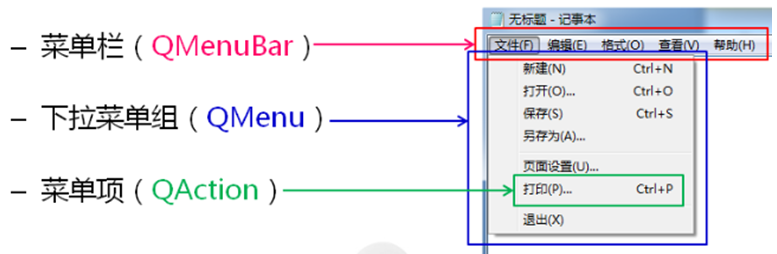


**QMainWindow组件布局**



**1.菜单栏**

QT中提供了预定义的与菜单相关的类组件:



**创建菜单示例:**

[复制代码](javascript:void(0);)

QMenuBar mb= menuBar(); 　　　　　　　　　　　　 //返回菜单栏,如果菜单栏没有创建,则返回一个空的菜单栏

QMenu\* menu= new QMenu("File(&F)", QMenuBar ); //(&F):表示添加alt+F键快捷键打开这个菜单

QAction\* action= new QAction("New(N)", QMenu); 　　　　　　　　//(N):表示添加N键快捷键

action-> setShortcut(QKeySequence (Qt::CTRL + Qt::Key\_N)); //为菜单项设置快捷键:ctrl+n

menu->addAction(action); //向菜单里添加菜单项"New"

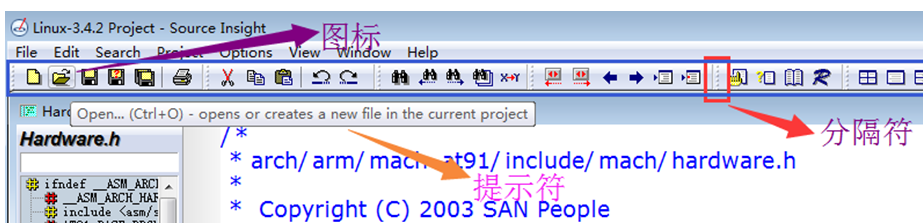
menu->addSeparator(); //添加菜单分隔符

mb->addMenu(menu); //向菜单栏里添加菜单File

[复制代码](javascript:void(0);)

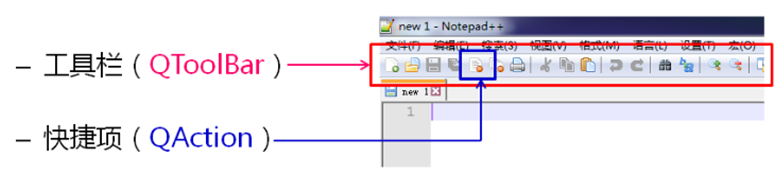
**2.工具栏**

* 工具栏中的元素可以是**各种窗口组件(QAction,Qwidget)**
* 工具栏中的元素通常以**图标方式**存在,
* 以**SI软件**为例,它的工具栏如下图所示:



**工具栏使用**

在Qt中,通过**QToolBar类**来创建工具栏,然后通过**QAction类**来创建工具项,如下图所示:



**QToolBar关键成员函数**

[复制代码](javascript:void(0);)

void setFloatable ( bool floatable );

　　　　 //设置工具栏是否可以当做独立的窗口,停靠在上下左右窗口旁,,默认为true

void setMovable(bool movable);

//设置工具栏是否可以自动移动,停靠在任意处,默认为true

void setIconSize ( const QSize & iconSize );

//设置工具栏中的每个图标大小

[复制代码](javascript:void(0);)

**创建工具栏示例:**

[复制代码](javascript:void(0);)

QToolBar \*tb = addToolBar("ToolBar"); //将"ToolBar"插入当前工具栏区域,并返回这个QToolBar对象

QAction \*action = new QAction("",QToolBar);

action->setToolTip("Open"); //信息提示,当鼠标靠近这个项,便会提示信息

action->setIcon(QIcon(":/res/pic/open.png")); //设置图标,冒号表示该文件处于资源文件指定的地方

tb->addAction(action); //将工具栏选项加入工具栏

[复制代码](javascript:void(0);)

**QToolBar中可以加入任意的QWidget组件,比如:文本框(用来查找文字)**

**示例,在MainWindow类的构造函数里写入:**

[复制代码](javascript:void(0);)

QToolBar\* tb = addToolBar("Tool Bar");

QAction\* action = new QAction("",QToolBar);

tb->setFloatable(false);

tb->setMovable(false);

action->setToolTip("Open");

action->setIcon(QIcon(":/Res/open.png"));

tb->addAction(action);

tb->addSeparator(); //添加间隔符

/\*添加QWidget组件\*/

QPushButton\* b = new QPushButton("Button");

QLabel\* l = new QLabel("Label",this);

QLineEdit\* e = new QLineEdit(this);

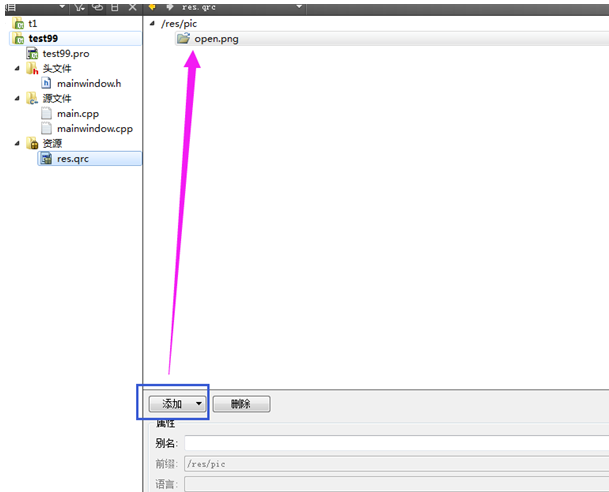
tb->addWidget(b);

tb->addWidget(l);

tb->addWidget(e);

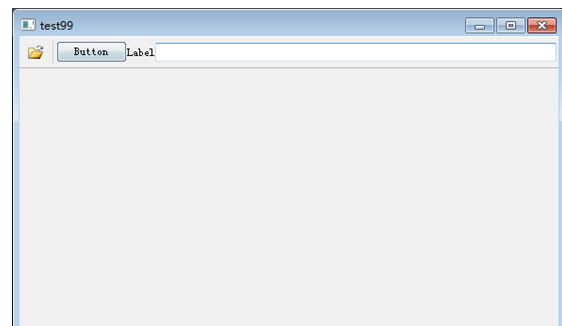
[复制代码](javascript:void(0);)

**添加Qt资源文件,将open.png加入程序指定的位置处:**



Qt资源文件的作用,就是告诉Qt Creator,该文件处于哪个位置

**编译运行:**



**3.状态栏**

* 状态栏一般位于主窗口最底部
* 用来显示简要信息

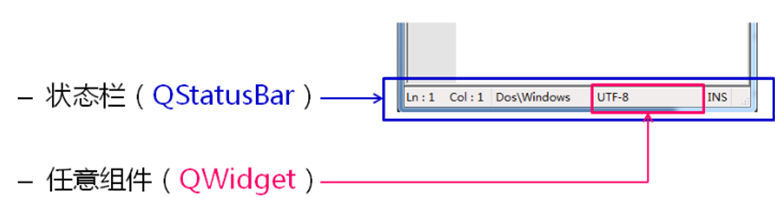
**显示的信息一般分为:**

**-实时信息**,如:  当前程序状态

**-永久信息**,如: 程序版本号,机构名称

**-进度信息**,如: 进度条提示,百分比提示

在Qt中,通过**QStatusBar类**来创建状态栏,这个类是个容器类组件,可以包含任意QWidget类,如下图所示:



**常用函数:**

[复制代码](javascript:void(0);)

QStatusBar \* statusBar ();

//获取当前类的状态栏,如果状态栏没有创建,则返回一个空的状态栏

void showMessage ( const QString & message, int timeout = 0 ) ;

//在状态栏下左下部显示信息, timeout 的单位是ms,timeout倒计时完会调用clearMessage()

void clearMessage ()

//清除消息

[复制代码](javascript:void(0);)

**创建状态栏示例:**

[复制代码](javascript:void(0);)

QStatusBar \*sb = statusBar();

QLabel\* status1 = new QLabel("Line: 1 Col: 1",this);

QLabel\* status2 = new QLabel("Lines: 0 Length: 0",this);

status1->setAlignment(Qt::AlignCenter);

status2->setAlignment(Qt::AlignCenter);

status1->setMinimumWidth(200); //设置最小宽度,避免与旁边的信息紧靠在一起

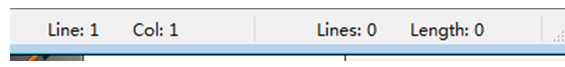
status2->setMinimumWidth(200);

sb->addPermanentWidget(status1);

sb->addPermanentWidget(status2);

[复制代码](javascript:void(0);)

效果:



**文本编辑组件**

**QT中提供了三种文本编辑组件**

* **QLineEdit :**单行文本编辑组件
* **QTextEdit:**多行富文本编辑组件(类似于word编辑,可以实现图片,个别文字的字体,颜色等编辑)
* **QPlainTextEdit:**多行普通文本编辑组件(类似于记事本编辑)

**Qt,在文本编辑组件里自带了右键式菜单**

可以实现复制,粘贴,剪切等,以QTextEdit为例:



**常用函数**

QMenu \* QLineEdit::createStandardContextMenu();

//获取右击菜单内容

menu->addAction(tr("My Menu Item"));

//添加右击菜单内容

addAction(QAction \*action, ActionPosition position);

　　　　　　　　　　　　　　　　　　//可以在编辑框的开始或者末尾添加一个action，方便我们操作，一个完美的搜索框顿时出现。

**以QPlainTextEdit组件为例,它的常用信号函数如下:**

[复制代码](javascript:void(0);)

void textChanged ();

//每当文档的内容发生变化时,则触发该信号,可以用来判断文本是否被修改,以及输入的字符是什么

void QPlainTextEdit::selectionChanged () ;

//当文本框鼠标选择改变时,触发该信号

void QPlainTextEdit::cursorPositionChanged ();

//每当光标的位置发生变化时，触发该信号

void undoAvailable ( bool available );

//当用户进行撤销操作时,则触发该信号,如果撤销成功则available 为true

void redoAvailable ( bool available );

//当用户进行重做操作时,则触发该信号,如果文本框为空则available 为true

void blockCountChanged ( int newBlockCount )

//每当按下回车或者删除回车(更新字符块),则newBlockCount计数,并触发该信号, newBlockCount 默认为1