基于QT的界面框架qcanpool使用教程

作者：马敏杰

文档版本控制记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 修改日期 | 修改章节 | 修改理由或内容 | 编写人 |
| V0.1.0 | 2018-03-02 |  | 创建文档 | 马敏杰 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[第1章 qcanpool是什么 3](#_Toc507794122)

[1.1 憧憬未来 3](#_Toc507794123)

[1.2 看破人生 5](#_Toc507794124)

[1.3 因何而生 5](#_Toc507794125)

[第2章 fancydemo介绍 6](#_Toc507794126)

[2.1 编译运行 6](#_Toc507794127)

[2.2 qcanpool库 7](#_Toc507794128)

[2.2.1 FancyBar 8](#_Toc507794129)

[2.2.2 FancyWindow 9](#_Toc507794130)

[2.2.3 FancyDialog 9](#_Toc507794131)

[2.2.4 FancyNavBar 9](#_Toc507794132)

[2.2.5 FancyTabWidget 10](#_Toc507794133)

[2.2.6 FancyTabBar 10](#_Toc507794134)

[2.2.7 IMode 11](#_Toc507794135)

[2.2.8 IModeBar 11](#_Toc507794136)

[2.2.9 MiniCard 12](#_Toc507794137)

[2.2.10 MiniStack 12](#_Toc507794138)

[2.2.11 ModeManager 12](#_Toc507794139)

[2.2.12 QuickAccessBar 13](#_Toc507794140)

[2.2.13 SimpleFilter 13](#_Toc507794141)

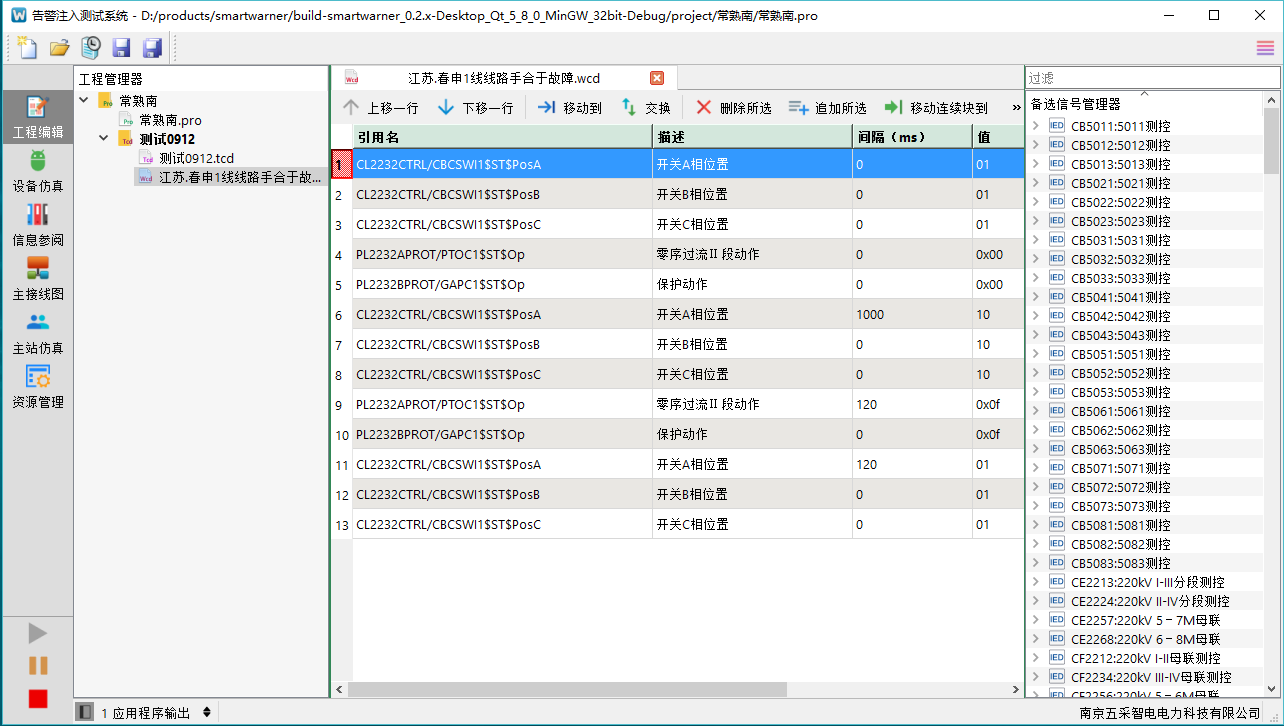
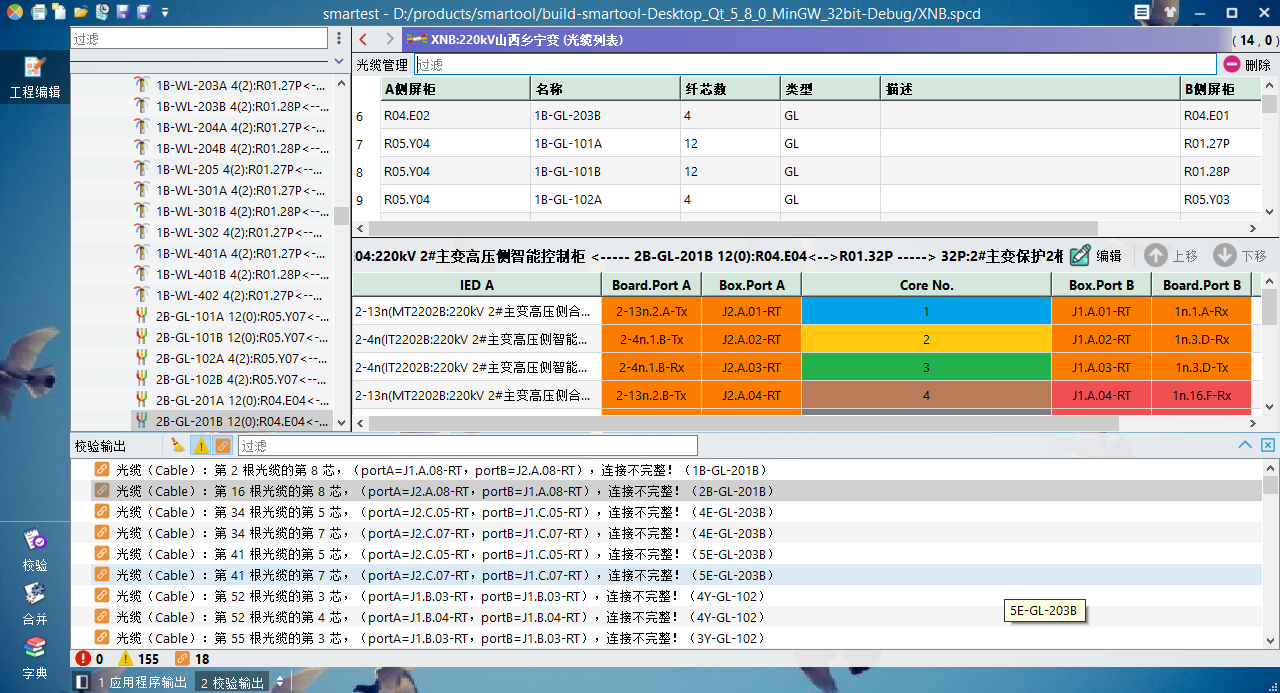
[2.2.14 TinyTabWidget 13](#_Toc507794142)

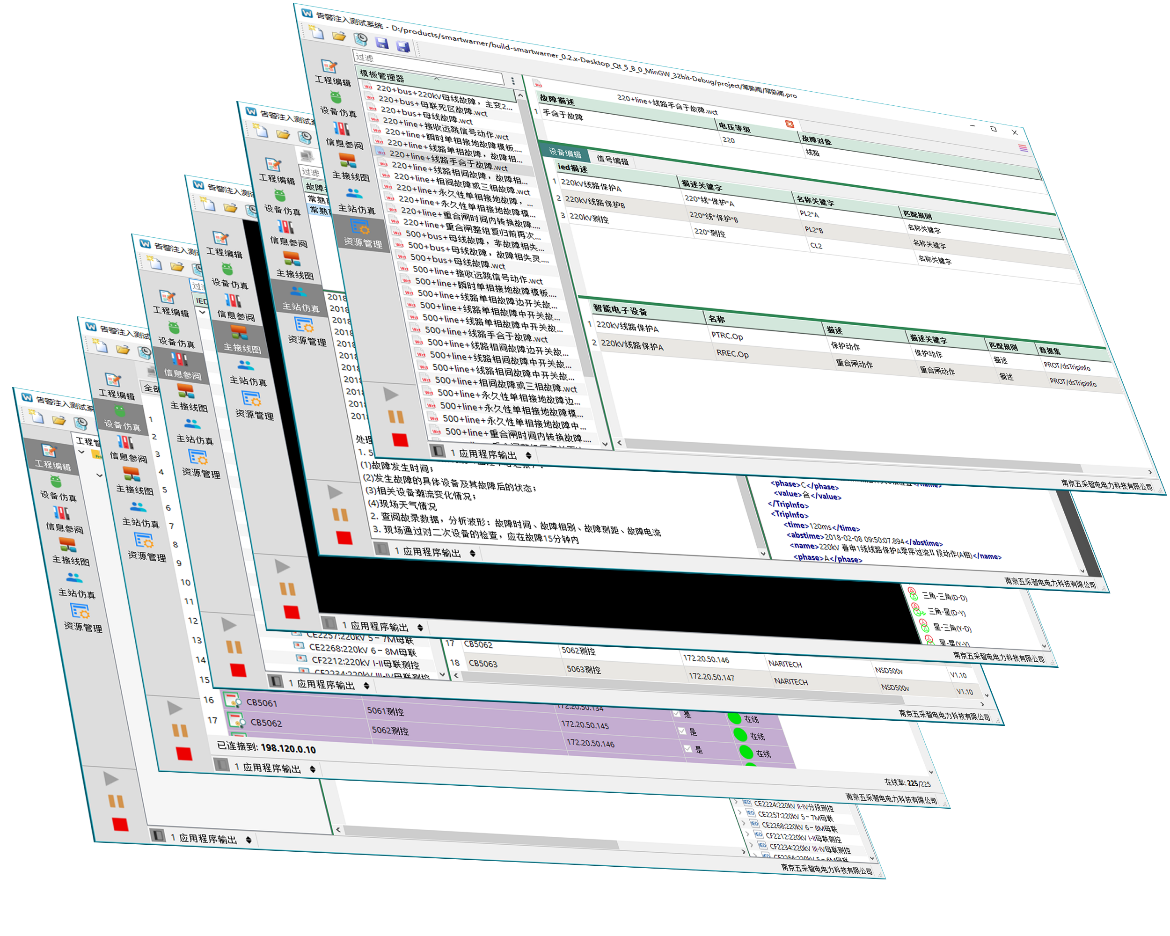
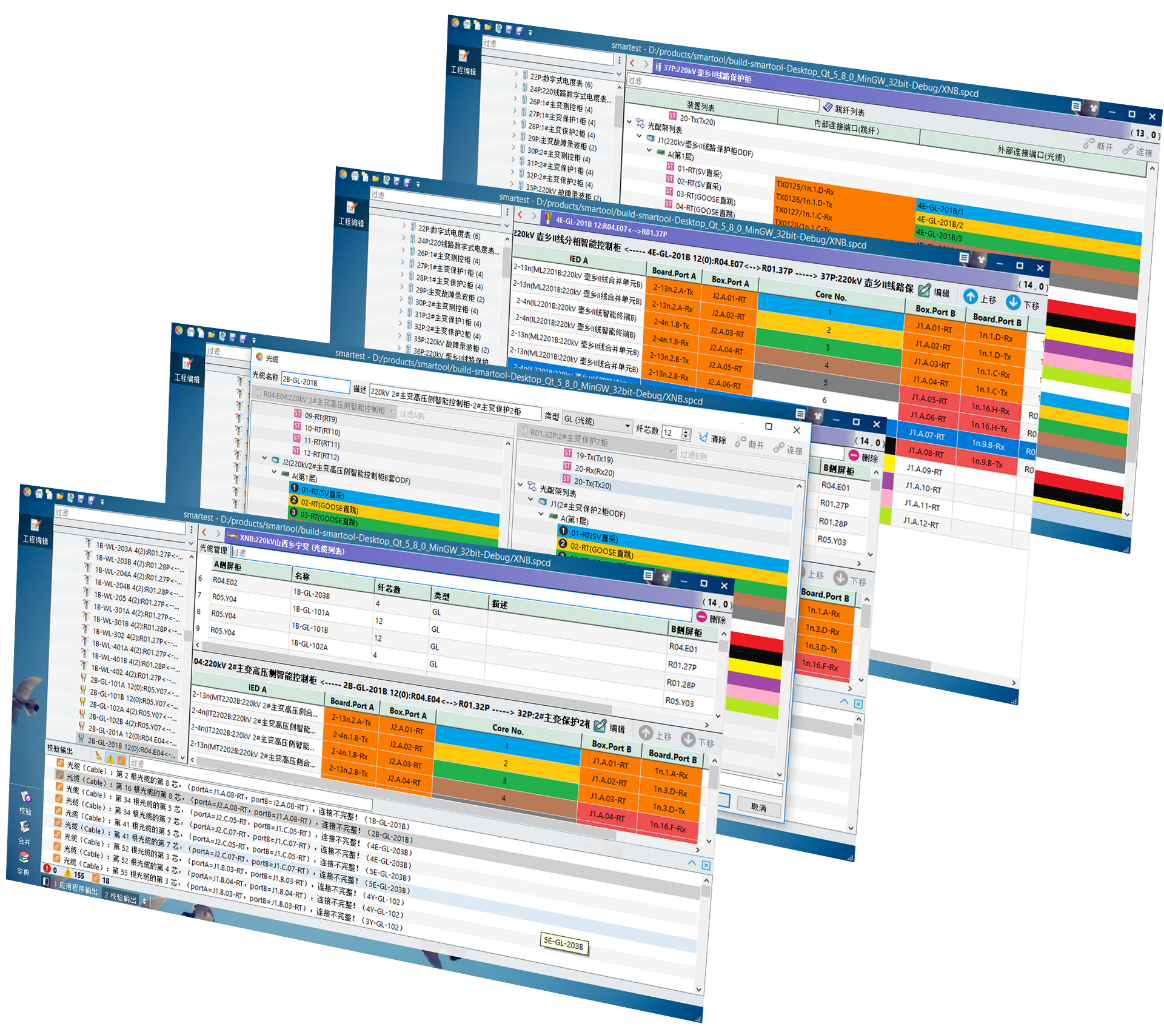
[2.3 fancydemo示例 13](#_Toc507794143)

# qcanpool是什么

## 憧憬未来

通过使用qcanpool做出来的软件，可以长成下图所示的样子，通过效果图也能看出，qcanpool适合于普通类的PC机软件，基于qtwidgets实现的。





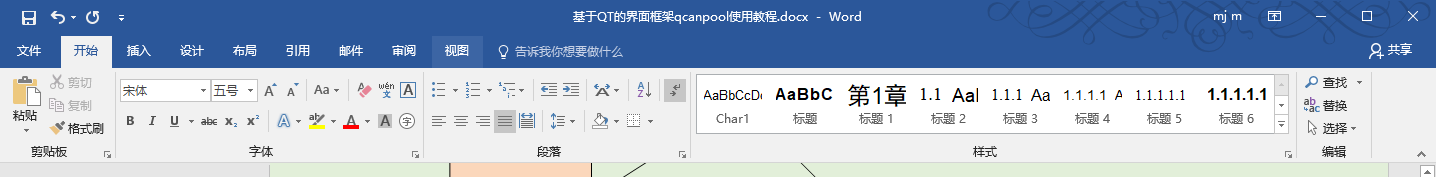
## 看破人生



前些年，网上都是这怎么破，那怎么破的。我在想，谁敢站出来说：我能破！于是我就想起个网名叫“能破”，能英文为can，就是can破，亦有看破人生的意味，破该怎么译呢？我想到了黑池大赛，你说你能破，他说他能破，那到底谁能破？不如到“黑池”里，看谁有能力破，因此就产生了能池，在这里能者居之。

## 因何而生

刚开始使用qt，就觉得windows下的qt，原生标题栏，没有颜色，又不能修改，好土。如何能像word、FoxitReader那样呢？于是就上网搜，搜出了qtitanribbon，可却是商用的，费用高达好几千，简直不舍得这么多money啊，没办法只能搜qt该怎么做？结果都是采用无边框的方式，我也试了一下，发现最大化时，简直不能忍啊，直接遮挡了windows的任务栏，然后就这么如何解决遮挡任务栏？如何通过鼠标改变窗体大小？如何多屏显示？……





在开发的过程中参考了360安全卫士、qtitanribbon、framelesshelper、qtcreator等。

# fancydemo介绍

环境：Qt5 mingw 32bit （windows qtcreator）

其他环境未严格测试过，若出现告警、错误，根据提示修复即可。

下面列出可能出现的问题：

1）如果编译器是vs2013、vs2015，可能会出现ShellExecuteA、Q\_ENUM之类的错误，可以注释掉，另外文件采用UTF-8编码，涉及中文的时候，编译可能报错，把中文改为英文，或者想法解决中文问题。

（1）一律采用英文描述，统一利用qt国际化翻译成中文

（2）文件采用UTF-8带BOM编码或其它（未测过）

（3）文件采用UTF-8带BOM编码或其它，定义宏变量，如:

#define APP\_NAME QTextCodec::codecForName(“GB2312”)->toUnicode(“演示”)

如果有些描述必须中文，我会采用第三种方式，将所有中文定义在一个文件中，然后针对不同的编译器，选择不同的中文文件，因为文件编码不同。

#if defined(Q\_CC\_MSVC)

#include "chinesedefines\_vs.h"

#else

#include "chinesedefines.h"

#endif

## 编译运行



状态栏中没有上图效果的，是之前的版本，需要添加如下几步，方能有上图的效果。

1. 在mainwindow.h中先声明FancyNavBar，即class FancyNavBar;。
2. 在MainWindow类中定义FancyNavBar的指针变量m\_pNavBar,即FancyNavBar \*m\_pNavBar;。
3. 在mainwindow.cpp中包含头文件，#include “qcanpool/fancynavbar.h”,

然后在createStatusBar()中添加下列代码：

m\_pNavBar = *new* FancyNavBar();

m\_pNavBar->setSideExpand(*true*);

*//* *m\_pNavBar->setVisible(false);*

QSplitter \*splitter = m\_modeStack->addCornerWidget(m\_pNavBar->panel(), FancyTabWidget::*Bottom*);

m\_pNavBar->setSplitter(splitter);

QTextBrowser \*tb1 = *new* QTextBrowser(*this*);

QTextBrowser \*tb2 = *new* QTextBrowser(*this*);

m\_pNavBar->add(tr("tb1"), tb1);

m\_pNavBar->add(tr("tb2"), tb2, *new* QLineEdit());

*//* *m\_pNavBar->remove(tb1);*

*//* *delete* *tb1;*

m\_statusBar->addWidget(m\_pNavBar);

具体意思，先不讲，后面再介绍。

## qcanpool库

|  |  |  |
| --- | --- | --- |
| 序号 | 类 | 说明 |
| 1 | FnacyBar | 定制的标题栏 |
| 2 | FancyButton | 定制的按钮，在FancyBar中使用 |
| 3 | FancyDialog | 定制的对话框，代替QDialog |
| 4 | FancyNavBar | 定制的导航栏，仿qtcreator状态栏中的导航栏 |
| 5 | FnacyNavButton | 定制的按钮，在FancyNavBar中使用 |
| 6 | FancyTabWidget | 仿qtcreator模式栏+中心区，类似QTabWidget |
| 7 | FancyTabBar | 仿qtcreator模式栏，类似QTabBar |
| 8 | FancyTab | 定制的按钮，在QTabBar中使用 |
| 9 | FancyWindow | 定制的窗口，代替QMainWindow |
| 10 | IMode | 仿qtcreator左侧的模式 |
| 11 | IModeBar | 定制的模式栏，类似FancyTabBar，较灵活 |
| 12 | IModeButton | 定制的按钮，在IModeBar中使用，类似FancyTab |
| 13 | MiniCard | 一种卡片式显示控件 |
| 14 | MiniStack | 一种可缩放的堆栈显示控件 |
| 15 | MiniSplitter | 简便的QSplitter |
| 16 | ModeManager | 管理由IMode构建的模式 |
| 17 | QCanpool | 包含通用的静态方法 |
| 18 | QuickAccessBar | 仿ribbon的快速访问栏 |
| 19 | ScreenHelper | 多屏显示处理助手 |
| 20 | SimpleFilter | 简单的过滤器，可由于QTableWidget、QTreeWidget、QTreeView |
| 21 | SkinItem | 皮肤项 |
| 22 | TinyTabWidget | 定制的TabWidget，代替QTabWidget |
| 23 | CentralArea | 简便的QScrollArea |
| 24 | Cursor | 鼠标状态处理类 |

### FancyBar

FancyBar类是定制的标题栏，ribbon的简化版，不单独使用，配合FancyWindow、FancyDialog使用。



提供如下一些接口：

1）QMenuBar \* menuBar() const;

获取菜单栏，然后进行菜单的操作

2）void showMenuBar(bool show = false);

控制菜单栏的显示

3）bool isMenuBarVisible() const;

判读菜单栏是否已经显示

4）QuickAccessBar \* quickAccessBar() const;

获取快速访问栏，然后进行快速访问按钮的操作

5）void showQuickAccess(bool show = true);

控制快速访问栏的显示

6）bool isQuickAccessVisible() const;

判读快速访问栏是否已经显示

7）void setHoverColor(const QColor &color);

设置按钮、菜单被鼠标滑过时的背景色

8）void setPressColor(const QColor &color);

设置按钮、菜单被鼠标左击时的背景色

9）void setTextColor(const QColor &color);

设置按钮、菜单的文本颜色

10）void setBackgroundColor(const QColor &color);

设置标题栏的背景色

11）void addAdditionalControl(QAction \*action, AdditionalControlPosition position);

设置额外的Action，位置可以是TitlePosition、MenuPostion

12）void addAdditionlControl(QWidget \*widget, AdditionalControlPosition position);

设置额外的widget，位置可以是TitlePosition、MenuPostion

13）void setApplicationWidget(const QString &label, QWidget \*widget);

设置应用程序按钮显示的widget，类似word2007的文件功能

14）void setApplicationButtonBkColor(const QColor &color);

设置应用程序按钮背景色

### FancyWindow

FancyWindow继承自QMainWindow，设置无边框样式，用FancyBar代替QMainWindow的菜单栏位置，因此FancyWindow的菜单栏，由FancyBar提供。



### FancyDialog

FancyDialog继承自QDialog，设置无边框样式，用FancyBar+windget代替原先中心区，widget成为新的中心区。



通过Qt::WindowFlags可以设置FancyBar的SystemButtons中的最大化、最小化按钮的显示。

### FancyNavBar

FancyNavBar定制的导航栏，仿qtcreator状态栏中的导航栏。





### FancyTabWidget

FancyTabWidget仿qtcreator模式栏+中心区，类似QTabWidget，一般作为FancyWindow的centralWidget。



添加FancyNavBar时，就是将FancyNavBar的panel添加到了BottomCorner位置。

### FancyTabBar

FancyTabBar仿qtcreator模式栏，类似QTabBar，一般不单独使用，单独使用会使用IModeBar。



### IMode

IMode仿qtcreator左侧的模式，在FancyTabBar和IModeBar中作为Mode或ModeMenu。

### IModeBar

IModeBar定制的模式栏，类似FancyTabBar，较灵活。用法如下：

void MainWindow::createModeBar()

{

// mode

m\_pEditMode = new EditMode();

m\_pDebugMode = new DebugMode();

m\_pReferMode = new ReferMode();

m\_pGraphMode = new GraphMode();

m\_pMasterMode = new MasterMode();

m\_pExplorerMode = new ExplorerMode();

m\_pModeBar->addMode(m\_pEditMode);

m\_pModeBar->addMode(m\_pDebugMode);

m\_pModeBar->addMode(m\_pReferMode);

m\_pModeBar->addMode(m\_pGraphMode);

m\_pModeBar->addMode(m\_pMasterMode);

m\_pModeBar->addMode(m\_pExplorerMode);

m\_pModeBar->selectMode(m\_pEditMode);

// m\_pModeBar->selectMode(m\_pExplorerMode);

m\_pModeBar->setEnabled(m\_pReferMode,false);

m\_pModeBar->setEnabled(m\_pDebugMode,false);

// m\_pModeBar->setVisible(m\_pGraphMode, false);

// m\_pModeBar->setVisible(m\_pMasterMode, false);

// mode action

QAction \*action = new QAction(QIcon(":/tools/start"), tr("Start"),this);

action->setShortcut(tr("Ctrl+R"));

action->setToolTip(tr("Start <i>Ctrl+R</i>"));

action->setEnabled(false);

m\_pModeBar->addAction(action);

m\_pDebugMode->addAction(action);

action = new QAction(QIcon(":/tools/suspend"), tr("Suspend"),this);

m\_pModeBar->addAction(action);

action->setEnabled(false);

m\_pDebugMode->addAction(action);

action = new QAction(QIcon(":/tools/stop"), tr("Stop"),this);

m\_pModeBar->addAction(action);

action->setEnabled(false);

m\_pDebugMode->addAction(action);

m\_pModeBar->setIconSize(QSize(24,24));

m\_pModeBar->setTextColor(QColor(10,10,10));

m\_pModeBar->setSelectedTextColor(QColor(255,255,255));

m\_pModeBar->setHoverColor(QColor(180,180,180));

}

### MiniCard

MiniCard一种卡片式显示控件。



### MiniStack

MiniStack一种可缩放的堆栈显示控件.



### ModeManager

ModeManager管理由IMode构建的模式，配合FancyTabWidget，通过objectAdded、objectRemoved接口可动态添加、删除模式，从而实现多功能软件。

### QuickAccessBar

QuickAccessBar继承自QToolBar，仿ribbon的快速访问栏。

### SimpleFilter

SimpleFilter简单的过滤器，可由于QTableWidget、QTreeWidget、QTreeView，构建时，需要传入控件类型，后续可通过相应的接口设置，实现同一过滤器共用。

### TinyTabWidget

TinyTabWidget定制的TabWidget，代替QTabWidget，用法同QTabWidget，只是添加颜色设置接口。

## fancydemo示例

CustomMode、ChartsMode、MenuMode都是继承自IMode，其中MenuMode可以实现菜单的功能。MainWindow中createWindow中的方法也都很好理解。