1、Qt 环境搭建

参考资料文件夹中的资料进行环境搭建

2、Qt 介绍

2.1 GUI 介绍

所谓 GUI（ Graphics User Interface），就是图形用户界面，采用 GUI 后，用户可直接对屏幕上的对象进行操作，如拖动、删除、插入以及放大和旋转等。用户执行操作后，屏幕能立即给出反馈信息或结果，称为所见即所得，用视、点（鼠标）代替了记、击（键盘），给用户带来了方便。通常所见的 GUI 都是位于 PC 机上的，但是在 PC 上 GUI 并不适合嵌入式系统。嵌入式设备有严格的资源要求（比如十分有限的存储空间）。同时嵌入式系统经常有一些特殊的要求，而普通的 PC 上的图形窗口系统是不能满足这些要求的。比如特殊的外观效果，控制提供给用户的函数，提高装载速度，特殊的低层图形或输入设备。因此可见嵌入式系统必定要有自己的 GUI。

综上所述，嵌入式 GUI 就是在嵌入式系统中为特定的硬件设备或环境而设计的图形用户界面系统。所以嵌入式 GUI 不但要具有以上有关 GUI 的特征，而且在实际应用中，嵌入式系统对它来说还有如下的基本要求：

体积小；
运行时耗用系统资源小；
上层接口与硬件无关，高度可移植；
高可靠性；
在某些应用场合应具备实时性。

下面我们就目前市场上常用的嵌入式 GUI 做一个简单的介绍。

1）MiniGUI

由北京飞漫软件技术有限公司开发的 MiniGUI(http://www.minigui.org),是国内为数不多的几大国际知名自由软件之一。MiniGUI 是面向实时嵌入式系统的轻量级图形用户界面支持系统，1999 年初遵循 GPL 条款发布第一个版本以来，已广泛应用于手持信息终端、机顶盒、工业控制系统及工业仪表、彩票机、金融终端等产品和领域。目前， MiniGUI 已成为跨操作系统的图形用户界面支持系统，可在 Linux/uClinux、 eCos、 uC/OS-II、 VxWorks、等操作系统上运行；
已验证的硬件平台包括 Intel x86、 ARM（ ARM7/AMR9/StrongARM/xScale）、PowerPC、 MIPS、 M68K（ DragonBall/ColdFire）等等。

MiniGUI 良好的体系结构及优化的图形接口，可确保最快的图形绘制速度。在设计之初，就充分考虑到了实时嵌入式系统的特点，针对多窗口环境下的图形绘制开展了大量的研究及开发，优化了 MiniGUI 的图形绘制性能及资源占有。MiniGUI 在大量实际系统中的应用，尤其在工业控制系统的应用，证明 MiniGUI 具有非常好的性能。

2）MicroWindows

MicroWindows（ http://microwindows.censoft.com)是一个开放源码的项目，目前由美国 Century Software 公司主持开发。该项目的开发一度非常活跃，国内也有人参与了其中的开发，并编写了 GB2312 等字符集的支持。但在 Qt/Embedded

发布以来，该项目变得不太活跃，并长时间停留在 0.89Pre7 版本。可以说，以开放源码形势发展的 MicroWindows 项目，基本停滞。

MicroWindows 是一个基于典型客户/服务器体系结构的 GUI 系统，基本分为三层。最底层是面向图形输出和键盘、鼠标或触摸屏的驱动程序；中间层提供底层硬件的抽象接口，并进行窗口管理；最高层分别提供兼容于 X Window 和 Windows CE（ Win32 子集）的 API。该项目的主要特色在于提供了类似 X 的客户/服务器体系结构，并提供了相对完善的图形功能，包括一些高级的功能，比如 Alpha 混合，三维支持， True Type 字体支持等。 MicroWindows 采用 MPL（ mozilla public license）条款发布。

3）DinX 非常适合于在很小的系统上运行多窗口程序，它简单、轻巧，并且快速。 DinX 并不是 X，它使用 Linux 核心的 framebuffer 视频驱动，采用 Client/Server 模式。为此，系统提供了两个界面： /dev/dinxsvr 和 /dev/dinxwin。

一个服务器程序连接到/dev/dinxsvr，并决定来自各程序窗口的 request 各占有视屏的各个部分。它也负责给各窗口发送像鼠标移动这样的事件消息。Clinet 程序连接到/dev/dinxwin，与 Server 进行消息通信等。 Server 进程还负责处理事件、窗口管理、调色板配置等功能。 DinX 是一个实验性的窗口系统，它处在发展阶段中，还存在一些缺陷和问题。DinX 的 license 属于 MPL，也可以转化为 GPL。这样，DinX 核心模块可以集成到 Linux 中，DinX 库可以链接到其他的 GPL 程序中。

4）OpenGUI

OpenGUI(http://www.tutok.sk/fastgl/)在 Linux 系统上存在已经很长时间了。最初的名字叫 FastGL，只支持 256 色的线性显存模式，但目前也支持其他显示模式，并且支持多种操作系统平台，比如 MS-DOS、 QNX 和 Linux 等等，不过目前只支持 x86 硬件平台。 OpenGUI 也分为三层。最低层是由汇编编写的快速图形引擎；中间层提供了图形绘制 API，包括线条、矩形、圆弧等，并且兼容于 Borland 的 BGI API。 第三层用 C++编写，提供了完整的 GUI 对象集。

OpenGUI 采用 LGPL 条款发布。OpenGUI 比较适合于基于 x86 平台的实时系统，可移植性稍差，目前的发展也基本停滞。

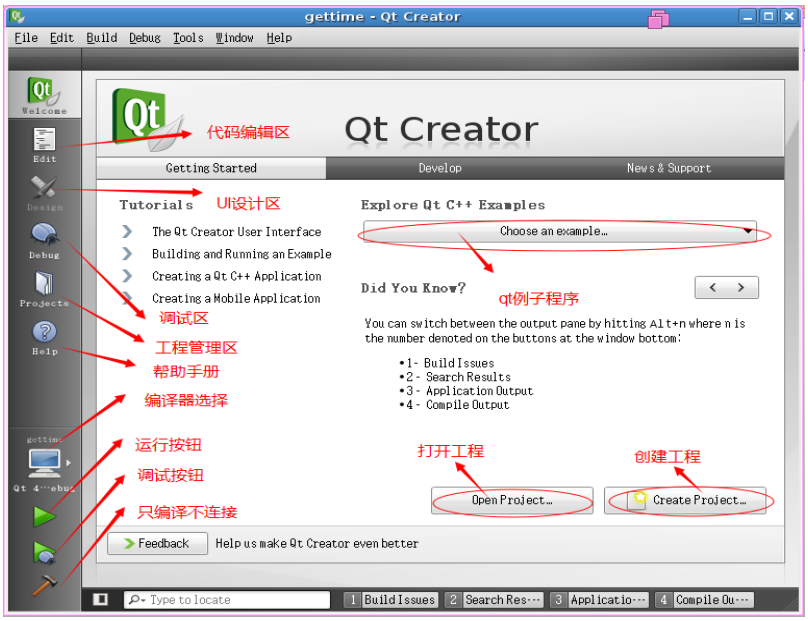
5）Qt/Embedded

Qt 是 nokia 公司(trolltech)的一个 C++图形库,可以跨平台使用,与 windows 下的 MFC 在同一个层次，但比 mfc 更好用。主要特点是比 mfc 要庞大，很多 Qt 是可以移植的。 Qt 最初主要是为跨平台的软件开发者提供统一的、精美的图形用户编程接口。 Qt 在 linuxX86 上通过 Qt lib 访问 linux 下的 x-server 与 x-lib，x-server 与底层硬件相连。 Qt 在 windows 上通过 Qt lib 访问 win api.

Qt 的嵌入式版本叫 Qt/Embedded，不需要 x-server 与 x-lib 的支持。可以直接写 framebuffer。所以很节省内存。

3 Qt creator 初始

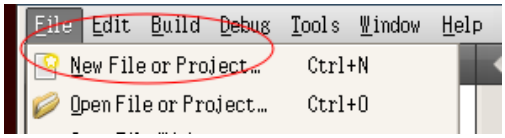
1）启动时欢迎界面



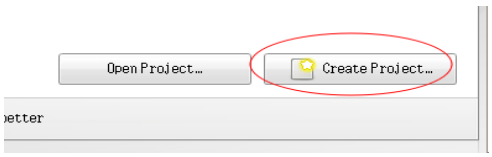
4 Qt 工程创建

4.1Hello Qt

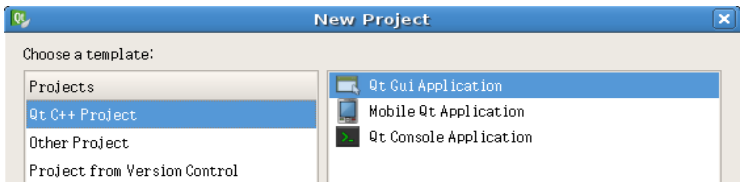
1）在菜单栏中 File-->New File or Project 新建工程



或者 在右下角点击 Create Project 按钮

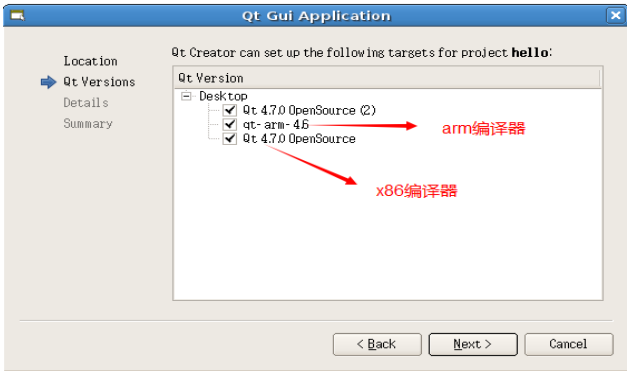


2）选择 Qt C++ project -->Qt Gui Application

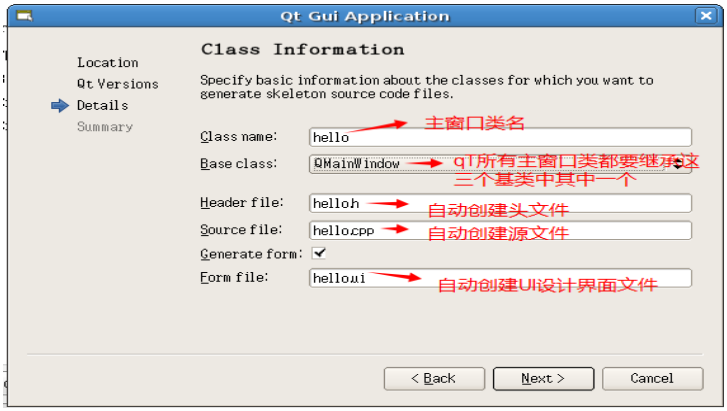


3)创建工程名和创建工程路径

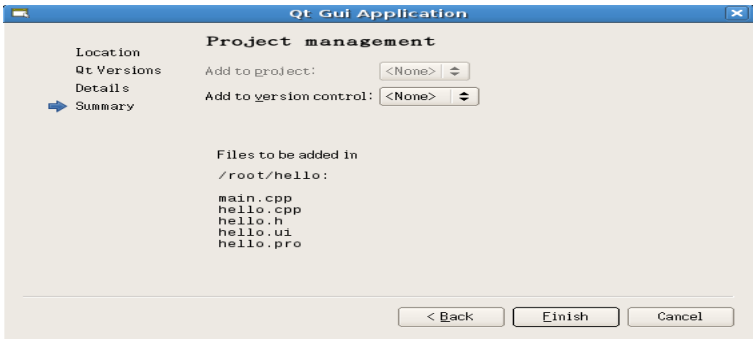
4）选择编译器



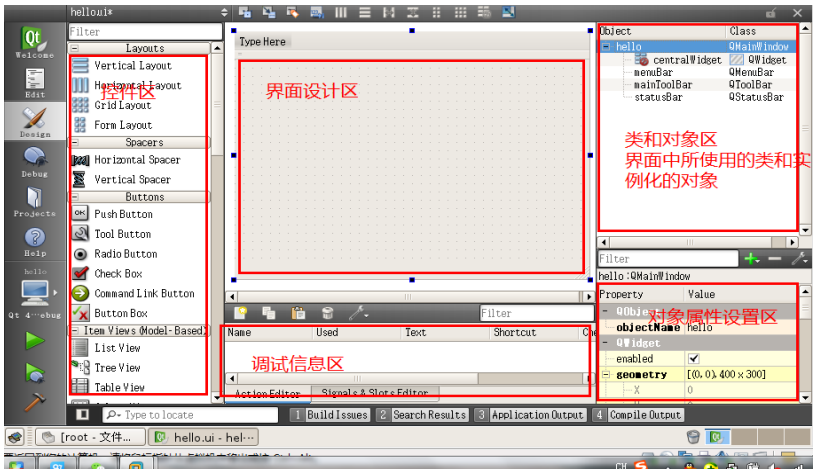
5）创建界面主窗口类



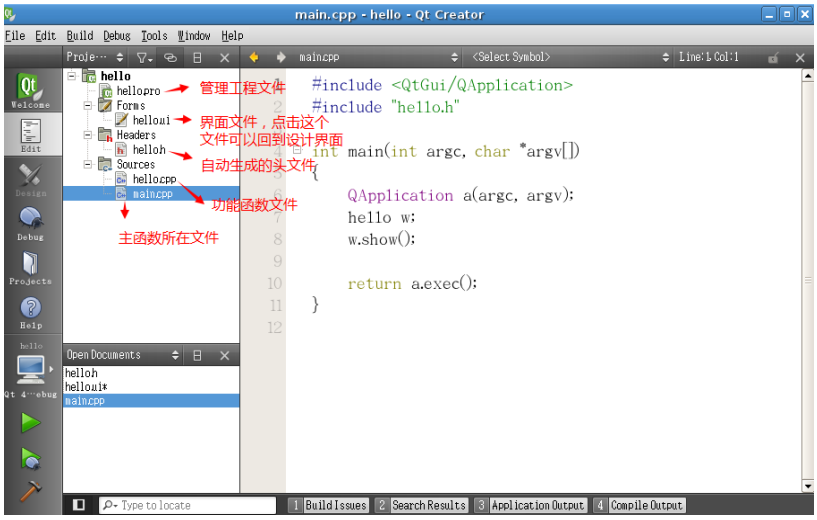
6）核查信息



7）UI 设计区



8）代码编辑区



9）在函数 man.cpp 中

#include <QtGui/QApplication> //qt 所需使用的类的头文件

#include "hello.h"

int main(int argc, char *argv[]) //是 main 函数标准写法

{

QApplication a(argc, argv);//创建对象，管理应用程序资源

hello w;//创建窗口对象

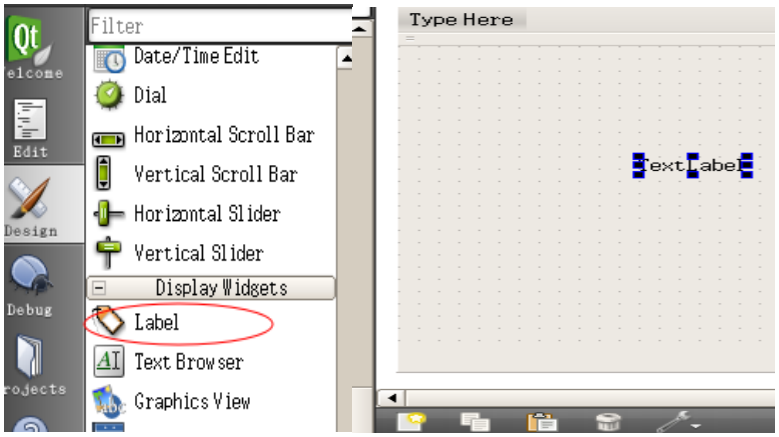
w.show();//显示主窗口

return a.exec();//进入消息循环，等待可能的菜单，工具条，鼠标等的输入，进行响应

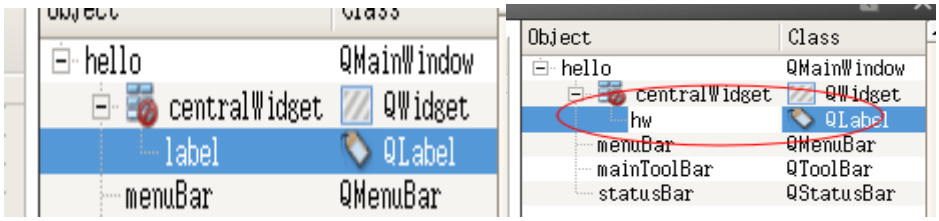
}

10）在设计界面添加控件显示 hello world！

在设计界面控件栏中选择 Lable 类，拖到界面设计区

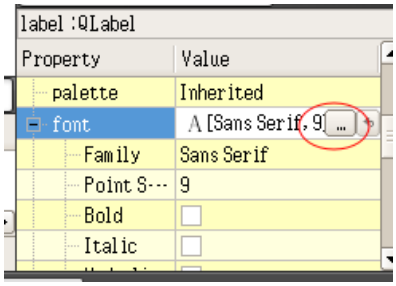


双击控件修改为 hello world！在类和对象区修改对象名



在属性栏中更改 hello world！字体大小 风格

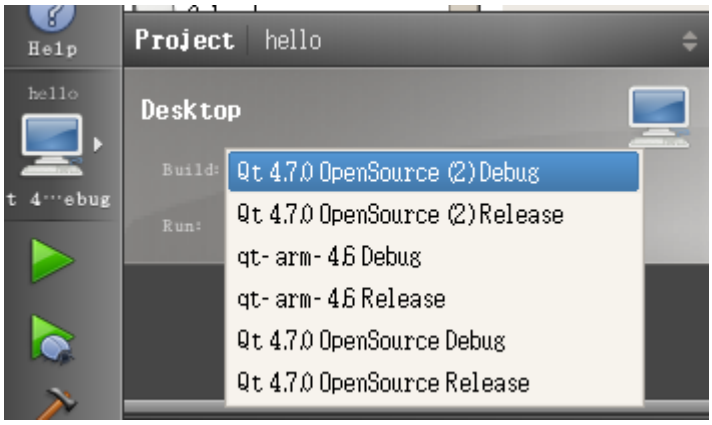
font-->



11)编译运行

选择编译器，如果在虚拟机中运行选择 Qt4.7.0OpenSource Debug

如果在 arm 开发板上运行选择 qt-arm-4.6 Debug



只编译不运行 ： 

编译并运行： 

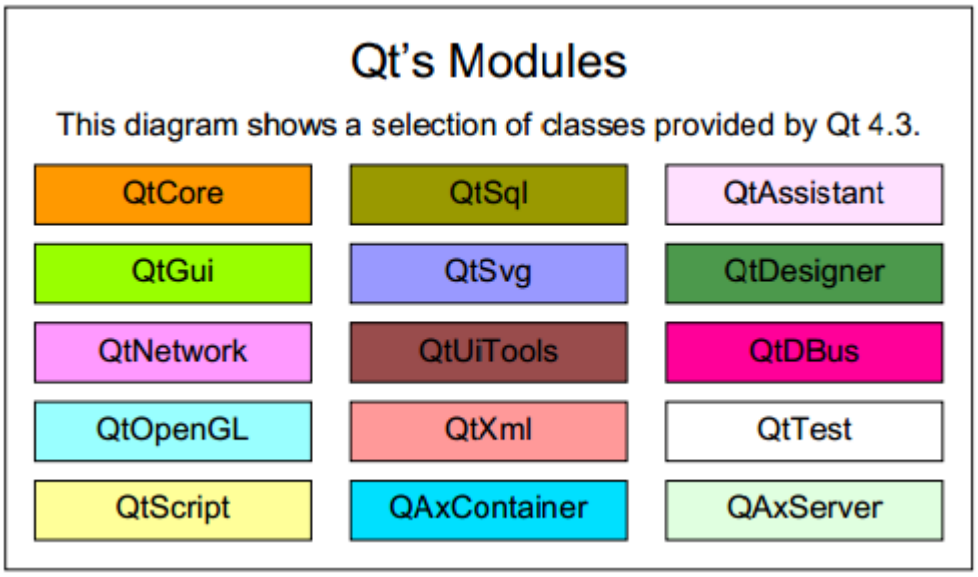
运行结果： 

5 Qt 模块和类库

5.1 Qt 模块

Qt4 提供了一些模块，他们分成了几个库的文件形式

Qt 常用模块及功能如下所示



1

1）QtCore：提供核心的非 GUI 功能，所有模块都需要这个模块。所有这些类都可以通过<QtCore>头文件引入。

2）QtGui：提供 GUI 程序的基本功能，包括与窗口系统的集成、事件处理、OpenGL 和 OpenGL ES 集成、2D 图像、字体、拖放等。这些类一般由 Qt 用户界面类内部使用，当然也可以用于访问底层的 OpenGL ES 图像 API。Qt Gui 模块提供的是所有图形用户界面程序都需要的通用功能。

3)QtNetwork：提供跨平台的网络功能。这些类可以通过 <QtNetwork> 引入，而且需要在 pro 文件中添加 QT += network。

4）QtOpenGL：提供对 OpenGL 的支持。

5）QtScript:提供对 Qt Scripts 的支持。

6）QtSql:提供对 SQL 数据库的支持。

7）QtSvg:提供对 SVG 文件的支持。

8）QtUiTools:用于在自己的引用程序中处理 Qt Designer 生成的 form 文件。

9）QtXml:XML 处理。

10)QtDesigner:用于扩展 Qt Designer。

11)QtTest:单元测试。

12)QAxContainer:用于访问 ActiveX 控件。

13)QAxServer:用于编写 ActiveX 服务器.QtDBus，使用 D-Bus 提供进程间交互。

5.2 类表

参考《Qt 类表图》，列出了类之间的继承关系

5.3 基于图形模式控件类

基于图形模式部件类提供了 QtCore 的扩展，包含有很多的部件 如下图

部件类	描述
QAbstractButton	抽象按钮基类部件提供了按钮的常用功能

QAbstractItemDelegate	用于在一个模型中编辑和显示数据功能
QAbstractItemView	项目视图类的基本功能
QAbstractPrintDialog	打印机的对话框
QAbstractScrollArea	使用滑块滑动区域
QAbstractSlider	整数滑块
QAbstractSpinBox	使用旋转框来编辑一个整数
QAbstractTextDocumentLayout	QTextDocuments 的自定义布局
QActionGroup	G 组动作
QApplication	用来管理图形应用程序的控制流与主设置
QBitmap	单色位图
QBoxLayout	子类水平与垂直的布局
QBrush	QPainter 的填充样式
QButtonGroup	按钮部件的组织容器
QCalendarWidget	日历部件
QCheckBox	复选框
QClipboard	访问窗口系统的剪切板
QColor	颜色类
QColorDialog	颜色对话框
QIcon	提供了关于图标的类
QIconEngine	QIcon 执行的基类
QIconEnginePlugin	提供了一些自定义的 QIconEngine
QImage	在硬件上绘画图片，并提供了关于访问像素的数据
QImageReader	用于格式化地从一个文件或者设备读界面
QImageWriter	用于格式化地写入文件或设备的基面
QSound	提供了基于平台的音频
QSpinBox	旋转框

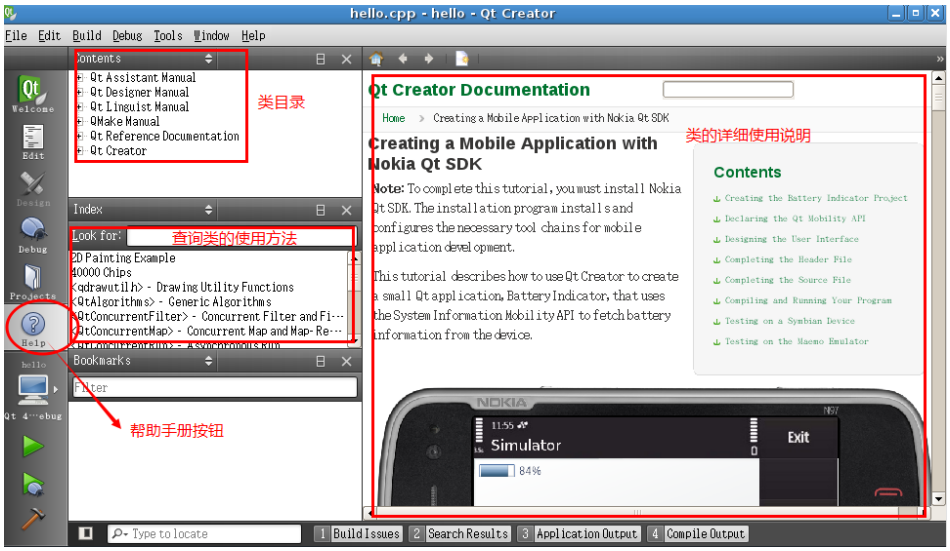
QStatusBar	状态条
QStyle	提供了图形设计的样式
QTabBar	制表条
QTableView	表格视图
QTextBrowser	富文本的浏览
QTextCursor	提供访问与调整文档的界面
QTextDocument	提供了一个格式化的文本编辑与显示
QTextEdit	提供了绘画与显示富文本
QTextFormat	提供了格式化的 QTextDocument 信息
QTextFragment	文本片段
QTextLength	文本长度
QTextLine	文本行
QTimeEdit	使用时间的编辑
QToolBar	工具条

QTreeView	树形视图
QTreeWidget	树形视图部件
QUndoStack	取消命令的对象堆栈
QUndoView	关于取消内容的显示

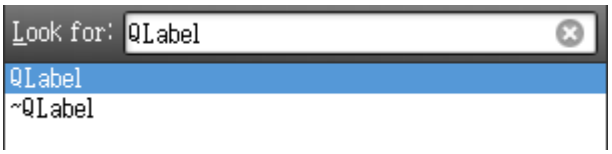
QVBoxLayout	横向的布局
QWidget	所有用户界面的基类
QWizard	向导框架
QWizardPage	创建向导页基类
QWorkspace	工作区域

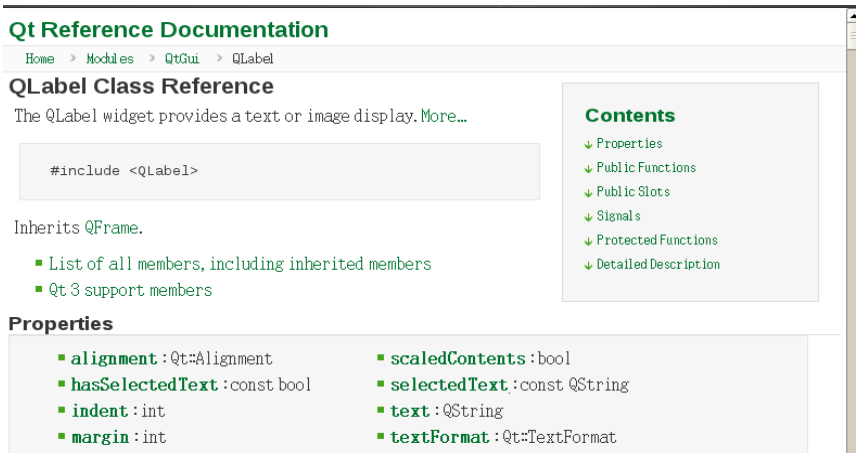
6、帮助手册的使用

关于 Qt 类库，Qt creator 提供了帮助手册，供开发人员参考



以 QLabel 类为例





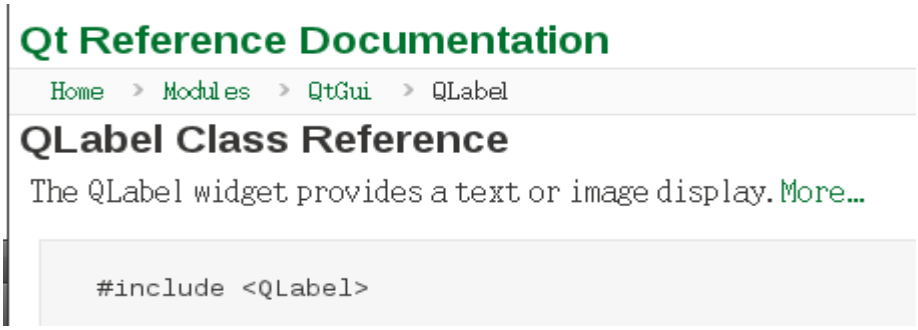
1) 类所属的模块

类所属于的模块

QLabel 属于 QtGui 模块

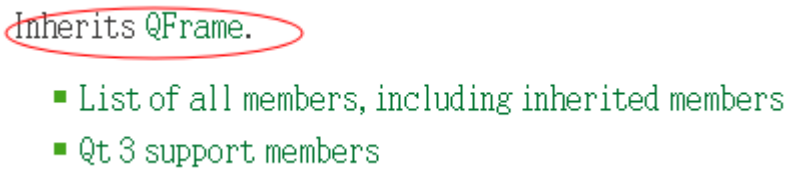
类的简单描述以及头文件

#include <QLabel>



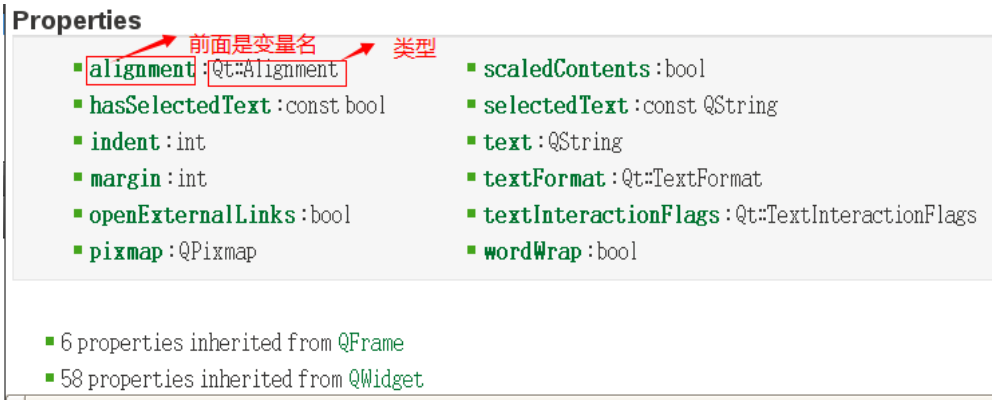
2) 类的继承关系

QLabel 继承自 QFrame 类



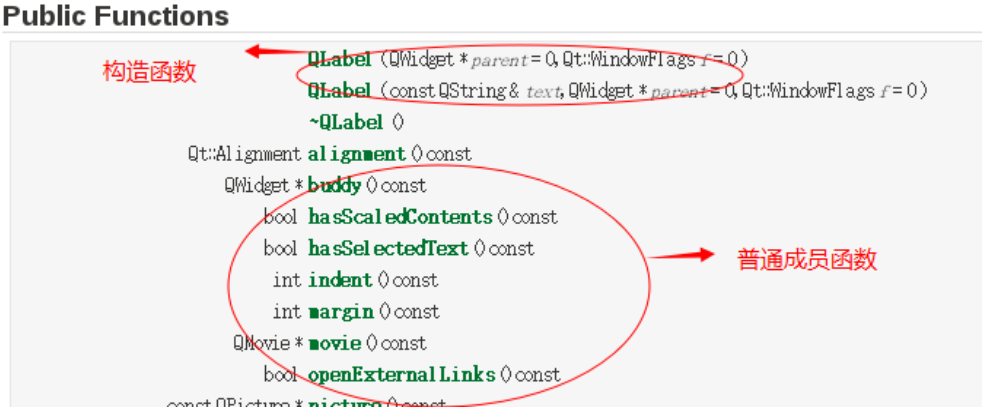
3) 成员变量

类自己的属性以及继承下来的属性



4)公有的成员函数

构造函数 普通成员函数 等



5) 重写公有成员函数

一般都是虚函数，可以提供覆盖版本实现多态

Reimplemented Public Functions

```
virtual int heightForWidth(int w)const
virtual QSize minimumSizeHint()const
virtual QSize sizeHint()const
```

- 14 public functions inherited from QFrame
- 217 public functions inherited from QWidget
- 29 public functions inherited from QObject
- 13 public functions inherited from QPaintDevice

6)槽函数和信号函数

主要用于组件之间的连接问题

Public Slots

```
void clear()
void setMovie(QMovie *movie)
void setNum(int num)
void setNum(double num)
void setPicture(const QPicture &picture)
void setPixmap(const QPixmap &)
void setText(const QString &)
```

- 19 public slots inherited from QWidget
- 1 public slot inherited from QObject

Signals

```
void linkActivated(const QString &link)
void linkHovered(const QString &link)
```

- 1 signal inherited from QWidget
- 1 signal inherited from QObject

7)保护函数

主要包含一些事件处理函数

Reimplemented Protected Functions

```
virtual void changeEvent(QEvent *ev)
virtual void contextMenuEvent(QContextMenuEvent *ev)
virtual bool event(QEvent *e)
virtual void focusInEvent(QFocusEvent *ev)
virtual bool focusNextPrevChild(bool next)
virtual void focusOutEvent(QFocusEvent *ev)
virtual void keyPressEvent(QKeyEvent *ev)
virtual void mouseMoveEvent(QMouseEvent *ev)
virtual void mousePressEvent(QMouseEvent *ev)
virtual void mouseReleaseEvent(QMouseEvent *ev)
virtual void paintEvent(QPaintEvent *)
```

- 3 protected functions inherited from QFrame

8)静态函数

Static Public Members

```
bool connect(const QObject *sender, const char *signal, const QObject *receiver, const char *method,
    Qt::ConnectionType type = Qt::AutoConnection)
bool disconnect(const QObject *sender, const char *signal, const QObject *receiver, const char *method)
const QMetaObject staticMetaObject
QString tr(const char *sourceText, const char *disambiguation = 0, int n = -1)
QString trUtf8(const char *sourceText, const char *disambiguation = 0, int n = -1)
```

9)类的详细描述

7、Qt 中文乱码的显示问题

查询 QTextCodec 类

1)获取编码对象

QTextCodec* coder = QTextCodec::codecForName("utf-8");

2)根据编码对象设置编码方式

QTextCodec::setCodecForTr(coder);

3)对显示的字符串内容进行翻译

QObject::tr("你好 Qt!"); => 返回翻译后的字符串内容

8、信号函数和槽函数的使用

信号和槽是 Qt 编程的一个重要部分。这个机制可以在对象之间彼此并不了解的情况下将它们的行为联系起来。

信号：当对象的内部状态发生改变，信号就被发射，在某些方面对于对象代理或者所有者也许是很有趣的。只有定义了一个信号的类和它的子类才能发射这个信号。

槽：当一个和槽连接的信号被发射的时候，这个操被调用。槽也是普通的 C++函数并且可以像它们一样被调用；它们唯一的特点就是它们可以被信号连接。槽的参数不能含有默认值，并且和信号一样，

例如一个按钮被单击时会激发一个“ clicked”信号，程序员通过建立一个函数（称作一个槽），然后调用 connect()函数把这个槽和一个信号连接起来，这样就完成了一个事件和响应代码的连接

链接函数：可以让信号函数和槽函数建立联系，在 QObject 类中有静态函数 connect

```
bool QObject::connect
(
    const QObject* sender,//信号发送者
    const char* signal,//发送信号内容
    const QObject* receiver,//接受者
    const char* method //响应方式
);
```

成功返回 true,失败返回 false

例如：

QPushButton 类中的信号函数：

```
void clicked();
```

QLabel 类中的槽函数：

```
bool close()
```

void hide()

调用 connect 函数建立连接：

QObject::connect(qp,SIGNAL(clicked()),&lab,SLOT(close()));

注意：

1）SIGNAL 和 SLOT 是带参的宏，进行函数调用到字符串的转换

2）建立关联时，信号函数和槽函数的参数类型，个数和顺序要一致

3）槽函数可以当做普通成员调用

4）信号函数和槽函数可以多对多，即一个信号函数对应多个槽函数，也可以一个槽函数对应一个信号函数

9、自定义信号函数和槽函数

9.1 格式：

public slots:

返回值类型 函数名(形参表);

public:signals:

返回值类型 函数名(形参表);

注意：

信号函数不用实现函数体；只需声明，由于信号函数机制比较复杂，编译器会提供函数体，

9.2 信号函数和槽函数不能直接连接的解决方案：

1)自定义一个槽函数连接

a.让信号函数和自定义槽函数连接

b.在自定义槽函数中调用不能连接的槽函数

2)自定义槽函数和信号函数作为中转

a.clicked 信号和自定义槽函数连接，自定义信号函数与 setText()链接

b.自定义槽函数内使用 emit 发射自定义信号信号

10、 使用 c++编程方式编写 Qt 程序

1) 编写头文件

a.自定义类继承自 QWidget/QMainWindow/QDialog

b.将窗口中的组件定义为类的属性

c.构造函数、析构函数的声明

d.可能需要自定义信号/槽函数

使用 Q_OBJECT

e.导入相关的头文件

2）编写实现文件

a.构造函数中初始化成员变量,调整位置

b.连接信号和槽函数的代码写在构造函数中

c.析构函数释放动态资源

d.实现槽函数的功能

3）主函数中实例化对象调用

创建自定义类型的对象，显示出来