2015-04-14 C++GUI Qt开发入门

B. Hello,Qt-创建第一个Qt程序

# Hello,Qt

本课时我们将开始创建第一个Qt应用程序.一般学习一门编程语言都是从打印Hello, World开始,所以我们今天会创建一个程序,显示Hello,Qt.

本课时概要:本课时将包括以下两个部分,首先我们会用基本的vi编辑器创建一个程序,程序的内容就如我们上面所说的,会显示一个字符串Hello,Qt.然后我们会介绍用qmake于make来编译构建程序.

## 创建一个Hello,Qt程序

#### 现在我们来实际演示创建Qt程序.这里我们暂时先不用Qt Creator,我们选择大部分操作系统都自带的vi编辑器来演示,然后使用qmake与make来编译构建.这次之所以不使用Qt Creator的目的,是为了让大家对于Qt程序的构建生成能有个了解,后面我们还是会使用Qt Creator来作为演示.

### 开始之前

##### 完成环境准备

###### Qt 5.4

###### Windows / OS X / Linux 任一操作系统

###### 完成Qt环境配置

##### 对Qt有个基本的了解

##### 会使用C++基本语法

#### 在开始写Qt程序之前,有一些必要的准备工作:

#### 首先要完成环境的准备,比如要安装Qt.建议至少是Qt5.2以上版本,这里我们使用的是前面课程中安装的Qt5.4.1.由于要在终端中使用qmake与make编译,完成Qt的环境变量配置也是必需的.另外,在操作系统方面你可以在Windows , OS X , Linux中随意选择你喜欢的,我们会在三个平台下分别演示.

#### 还有需要对Qt和C++能有个基本的了解,同时会使用循环,判断,选择,排序等基本的C++语法.

### 程序演示

#### 现在我们来实际演示下创建程序.首先我们会在OS X下进行演示.由于OS X对开发比较友好,所以我们的演示主要都在OS X下进行,对于不同平台存在差异的部分我们会分平台分别演示.

#### 这次我们先使用vi编辑器来创建程序,因为vi编辑器是在大部分平台上都内置了,而Windows上由于没有vi,所以在Windows上我们会使用自带的记事本程序.后面我们会再逐行讲解这个程序.

#### 下面我们来看程序:

#### 打开终端,先创建一个文件夹用来保存以后编写的代码,这里我们给文件夹取名为QtGuiStudy,保存在用户文件夹下.需要特别注意的是,Qt的所有项目,路径以及项目名都不能有中文,包括用户名,不然在编译的时候会报错,而且这个错误一般情况下很难发现.

#### 然后mkdir,创建一个hello文件夹用来保存本节课编写的源码.进入文件夹,输入vi main.cpp,由于文件夹中并没有这个文件,所以vi编辑器会自己创建一个并打开.

#### 首先我们输入的是注释.

#### 我们创建的Qt程序包含了两个部分,首先是一个GUI程序,然后会显示一个字符串Hello,Qt.我们选择使用Qt中的组件QLabel来显示这个字符串.所以这里我们要包含Qapplication与Qlabel的类定义.输入#include <Qapplication> , #include <Qlabel>.

#### 包含文件有了,然后是函数头, int main.

#### 下面是函数内容.我们先创建一个Qapplication对象,用来显示这个整个程序的UI.

#### 然后创建一个Qlabel对象,label,它是一个新的Qlabel组件.

#### 设置label的显示内容,Hello,Qt,然后使它显示出来.

#### 最后 return,程序到这里就结束了.

#### 按下esc进入vi的命令模式,输入:wq保存

#### 接下来我们来编译这个程序.输入qmake –project “Qt +=widgets”,如果提示找不到qmake,那么就是环境搭建有问题.这里正常的话不会有其他提示.

#### 输入ls,我们可以看到目录下生成了一个.pro格式的文件,这是一个与平台无关的项目文件.然后输入qmake hello.pro,执行完成,输入ls看看这次生成了什么.我们可以看到多了一个makefile文件,这是一个与平台相关的构建文件,供G++编译器调用.

#### 有了makefile文件,接下来就可以使用编译器编译了,输入 make,会自动开始编译构建程序.编译完成,输入ls,可以看到生成了一个hello.app文件,我们运行看看.输入open hello.app,程序打开,可以看到程序中显示了一个字符串,hello,qt.这里我们先不考虑程序的美化以及控件布局.

### 程序源码

// C++GUI Dev with Qt

// Hello,Qt by jkxy, 2015.4.14

// The First Qt Program

#include <QApplication>

#include <QLabel>

int main(int argc, char \*argv[])

{

QApplication hello(argc, argv);

QLabel \*label = new QLabel;

label->setText(“Hello,Qt”);

label->show();

return hello.exec();

}

#### 下面我们来逐行解释程序中用到的代码.

#### 前面三行,是文件开头的注释. 一个程序开头的注释应该包括项目名,文件名,创建者,创建时间,以及文件描述.这里我们只有这一个文件,所以文件名我就省略掉了.

#### 第四行跟第五行,包含了Qapplication与Qlabel类的定义. 每一个Qt类都会有一个以大写Q开头,并且与这个类同名的头文件,头文件中包括了对这个类的定义,所以程序中使用到的类都应该在程序开头包含.

#### 第六行是函数头, int main,我们知道所有的C++程序都要有一个main函数,程序会从这里开始执行.

#### 8-12行是函数内容.首先是创建Qapplication对象,用来管理整个程序的显示资源,这里我们给这个对象取名hello,当然你也可以叫他app,或者其他什么都可以.它需要2个参数argc, argv. Qt支持他自己的一些命令行参数.

#### 然后创建一个Qlabel对象,label.Qlabel是Qt的窗口组件widget中的一种,可以用来显示文本,同时也支持富文本显示.后面我们会详细讲解Qlabel以及Qt中的其他各种Qwidget.

#### 第十行我们设置Qlabel的显示文本,setText->Hello,Qt,然后我们设置它为显示.Qt中每个widget创建的时候都是隐藏的,我们应该先设置好组件的属性,显示内容等等再将它显示出来,这样可以避免显示以后再设置,这样会出现组件闪烁的情况.

#### 最后是return,这里将应用程序返回给Qt,这个时候程序会进入事件循环,从另一个角度去理解也可以认为现在程序在等待用户的下一步操作.在这个过程中,程序界面应该是不动的,除非接受到其他事件消息,这也是GUI程序与命令行程序的区别,命令行程序不用人为干预就可以自己处理完成输入,生 成结果然后终止退出.

#### 另外,如果要考虑节约内存,那么在函数最后应该对label做一个delete操作,但是对于一个这么小的程序,我们没有必要去考虑内存泄露的问题,在程序结束的时候这部分内存会被照常释放.

#### 好了,现在你可以自己试着写一个简单的Hello,Qt程序了,写的过程中尽量不要按照源码抄,也不要使用复制黏贴,可以试着自己写一遍,在编译或者运行的过程中遇到错误的话可以按照报错的内容去查找错误.

## qmake与make

##### qmake编译

##### IDE编译(Qt Creator)

##### 第三方编译工具

#### 下面我们将讲解下Qt程序的编译过程.Qt可以有三种编译方式,分别是使用qmake,使用IDE工具直接构建运行,也就是在Qt Creator中直接运行,这种方式最方便.还有使用第三方编译工具,这里我们将介绍使用qmake编译,使用Qt Creator 编译会在后面讲解,至于使用第三方工具我们不做介绍,因为Qt提供的工具已经足够使用了,而且足够方便.需要注意的是,如果需要使用第三方工具,建议使用可感知Qt的工具,会更加方便.

### Qmake编译

##### Qmake编译步骤

##### qmake –project

##### qmake

##### make

#### qmake工具跟随Qt一起提供,已经包括在了我们之前安装的QtSDK中,Qt本身,我们使用的QtCreator以及Qt自带的所有实例程序插件等等都是使用qmake编译的,包括我们以后编写的Qt程序.

#### 下面我们来说说Qt程序使用qmake的编译步骤.就像我前面做的,先使用qmake –project,这样可以生成一个pro文件.

#### Qmake是专门用来编译Qt的,所以对Qt很熟悉,就像前面说的,包含了调用 Qt内置代码生成工具 (moc,wc 和 rcc)的必要的逻辑规则.但是,qmake不仅仅只可以用于编译Qt,也可以用于编译其他C++程序,但是每一个使用qmake编译的程序都需要有一个pro文件,qmake工具可以使用这个与平台无关的 文件生成与平台相关的 makefile。

#### 下一步是使用输入qmake命令.这一步会创建一个makefile文件.

#### Qmake提供了很多的特性,包括创建可以递归调用其他 makefile 的 makefile文件,以及根据目前平台切换相关特性的开关状态等 .关于qmake的一些高级特性,目前不做过多介绍,我们会在以后需要用到相关知识的时候再做讲解.

#### 最后我们使用make命令,调用编译器来编译.而在Windows下,由于我们使用了MinGW编译器,所以make命令应该改成mingw32-make.

#### 正如前面所说,qmake可以生成一些触发 moc,wc和rcc的规则,正是因为这样,qmake的语法跟Qt一样简单明了,容易学习.

### Qmake创建工程文件

##### qmake –project //创建pro文件

##### qmake –tp vc hello.pro //创建VS工程文件

##### qmake –spec macx-xcode hello.pro //创建Xcode工程文件

##### qmake –spec macx-g++ hello.pro //创建特定平台与编译器的makefile文件

#### 一般情况下,我们使用qmake的主要目的是生成.pro文件的makefile供编译器调用,但是也可以在后面加上不同的参数生成不同的文件,比如供各个IDE使用的工程文件.这里列举了几种典型情况.如果在qmake后面加上参数-project,那么可以生成pro文件,qmake会自动扫描目录下的源码文件添加到pro文件中,.另外,如果使用参数-tp vc,可以生成对应pro文件的vs工程文件,这样就可以在Microsoft visual studio中打开这个工程继续开发,同样如果使用-spec macx-xcode,则会生成Xcode的工程文件.

#### 一般情况下,qmake会自动识别平台,但是在某些情况下我们需要指定使用的平台和编译器,qmake会根据-spec参数后面跟随的平台和编译器来生成相应的makefile文件.更多具体可用的规则,可以在Qt安装目录下的mkspecs目录中找到.

## Windows与Linux演示

#### 前面我们演示了在OS X中使用vi编辑器创建Hello,Qt程序.考虑到部分同学可能会使用Windows或者Linux系统,为了避免不同系统差异性带来的影响,接下来我们会在Windows与Linux上分别演示一次.

#### 首先是Windows系统.Windows上由于没有自带vi,所以我们使用自带的记事本程序.我们在C盘根目录新建一个文件夹, QtGuiStudy,进入然后再新建hellow文件夹,在Windows下也是一样无论是路径还是项目名都不能存在中文.然后右键,新建文本文档.需要注意的是Windows默认不会显示文件的扩展名,所以我们需要手动打开.Windows8以上的系统直接跟我这样打开就行了,Windows7需要按下Alt键再选择工具->文件夹选项.将新建文本文档重命名,main.cpp,然后打开,输入代码.由于前面已经讲解过一次,这里就不做过多解释,基本上都是一样的.完成后保存,Qt在Windows下提供了一个命令行工具,在开始菜单中找到这个工具,打开,会自动进入Qt环境,先进入源码所在目录,然后输入 qmake –project “Qt +=widgets “,然后后qmake,mingw32-make,编译完成,运行hello.exe,好了,程序运行.我们可以看到基本上跟OS X上是一样的.

#### 下面我们来演示在linux下面的操作.由于Linux上基本上与osx上一样,所以我们就不做过多演示,直接将代码拷贝过去,打开终端,进入项目文件夹,然后输入qmake –project “Qt +=widgets”,

#### 好了,本课时就到这里,现在你应该已经可以写出一个简单的Qt程序啦.下个课时我们会讲讲Qt Creater以及Qt文档.

# Qt Creator与高效开发

从前面的课程开始,我们就一直在将Qt Creator多好多好,它是一个多好用的IDE,能给Qt开发提高多少效率,但是一直没有个实际的概念.口说无凭,本课时我们就讲讲如何使用Qt Creator来快速高效的开发Qt程序,并且介绍下Qt的帮助文档.

本课时内容包括:Qt Creator的使用演示,包括使用Qt Creator创建一个Qt程序,这里我们还是选择Hello,Qt这个程序.因为它比较简单,而且大家都熟悉了.

然后我们会讲解Qt 工程中包括的文件,同时介绍一些简单的pro文件语法.最后我们会简单介绍Qt Assistant帮助程序以及Qt的示例,至于Qt Linguist语言家,我们会在后面用到的时候再做介绍.

## Qt Creator的使用

#### 这里我们还是在OS X下进行演示.至于在Windows与Linux中,Qt Creator与在OS X中相同,所以我们不做过多演示.

#### 下面我们运行Qt Creator,OS X可以在LaunchPad中直接打开,Windows也可以在开始菜单中找到,Linux如果没有启动器,则可以在终端中输入QtCreator中运行.

#### 下面我们来看Qt Creator

#### 启动QtCreator以后,我们首先看到的启动界面,这边有三个选项卡,项目,示例,和教程.示例中是Qt提供的一些代码示例,这些代码示例包括了很多Qt种典型的模块和类的使用,能给我们以He后的学习开发带来很高的参考价值.还有教程,我们之前也说过Qt的帮助文档是Qt最重要的一部分之一.

#### 左边的选项栏我们可以看到7种模式,分别是我们现在看到的启动界面,代码编辑模式,UI设计模式,Debug模式,项目管理,分析和帮助.具体的我们会一个个介绍.

#### 在上面的菜单栏我们可以看到Qt具有很多功能,比如说项目的构建调试,还有版本管理等等.

#### 下面我们就新建一个Hello,Qt程序来实际演示并讲解.

#### 使用Command+N或在文件->新建文件或项目来新建一个项目,这里我们选择Desktop模板,左边的项目中选择Application,可以看到这里有4种程序供我们选择,分别是我们这次用的Qt Widget Application,也就是Qt窗口组件程序,基于Qt Quick的Qt Quick程序,没有UI的控制台程序,和Qt Quick UI项目.这里我们选择第一项, Qt Widget Application.

#### 项目名称,我们输入HelloQt,路径选择之前创建的QtGUIStudy,并设为默认的项目路径,这样就不用以后每次都要重新选择.点击继续,这里需要选择构建套件,如果只安装了桌面版本,那么只会显示一个Desktop选项,如果安装了Android或者iOS版本这里也都会列出来,勾选需要的版本,这里我们选择Desktop,继续,这里我们保留默认,先取消勾选创建界面.至于使用Qt Design创建界面我们会在后面再讲.好了,我们可以看到自动切换到了编辑模式,并生成了四个文件…(念一遍文件名)

#### 我们先构建运行试试,后面再回来看这几个文件.按下Command+R或者点击左下角的运行按钮,可以看到运行了一个程序窗口,这个程序就是刚才创建的项目的程序,当然现在还没有任何东西,因为我们还没有加上.下面我们就来给他增加点东西.

#### 双击mainwindow.cpp打开,先包含Qlabel,然后输入 QLabel \*label = new QLabel,这里与我们之前创建的程序一样.

#### 然后设置内容,显示.完成了,我们再运行一次,IDE会自动帮你重新构建并运行.如果这一步保存,需要点击菜单栏中的构建->清理项目来清理一次构建过程中生成的文件.

#### 构建完成,我们看到生成的程序,与我们之前编写的一样,但是要更加方便了,可能因为这个程序实在太小了,我们不容易看出来,以后写的多了就会知道Qt Creator在使用Qt开发的过程中是必不可少的.

#### 我们还有另外一种更直观快捷的方式来生成应用程序的UI界面.就是使用Qt Designer工具.它可以创建一个供C++项目使用的UI文件.Qt Designer可以作为一个独立的应用运行,也可以在QtCreator中直接使用.我们把这个工程关掉,然后删除文件.它已经用不到了,我们有更高级的方法.重新建立一个工程.Command+N新建,还是选择Qt Widget Application,其他的都与之前的一样,不过记得要勾选创建界面.可以看到文件列表中多了一个界面文件mainwindow.ui.打开这个文件,会自动切换到设计模式.ui文件只能在设计模式下编辑修改.

#### 我们在左边的控件栏中找到label,拖动到右边的窗体中,在组件上双击就可以修改了.修改完成,我们可以按下Command+Option+R来预览.程序运行以后的效果就是这样子的.我们可以编译运行看看,跟预览效果一样.

#### 如果想要使用高大上一点的方式设置label里面的文字怎么办,因为一个组件可能需要在程序运行以后再做修改的.我们可以这样做,打开mainwindow.ccp,因为ui\_mainwindow.h的文件中已经包含了QLabel,所以这里就不用包含了,直接在下面新建一个QLabel对象,等于ui中的label对象,然后重新设置label显示的内容,就写Qt大法好!好了,然后运行,可以看到label显示的内容改变了.

## Qt项目文件

#### 下面我们讲解一下Qt项目中的文件.

#### 我们可以看到,新建一个Qt项目后生成了5个文件…(念一遍文件名),一个pro文件,一个头文件,两个cpp文件,一个界面文件.这就是一个Qt Widget开发的GUI持续的基本构成.其中程序都是从main.cpp开始运行的,它里面创建了一个Qapplication对象,app,同时包含了mainwindow的头文件,并创建了一个mainwindow对象,并显示了这个对象.而mainwindow.h中包含了对mainwindow.cpp文件中用到的公共变量,还有各个函数的定义,后面还会有信号和槽的定义.而mainwindow.ccp文件就是程序的主要源码了.我们可以看到mainwindow.cpp中除了包含了mainwindow.h,还包含了ui\_mainwindow.h文件.大家肯定在疑惑这个文件是怎么来的,因为我们的项目中并没有这个文件.而mainwindow.ui文件却没有在其他文件中被调用.我们可以打开项目文件夹看一下,返回上一级,可以看到多出了一个构建文件夹,将构建路径与源码路径分开 ,是为了保证程序源码路径的清洁.,我们叫做out-of source构建,也叫shadow-build.进入构建路径,我们看到这里果然有一个ui\_mainwindow.h文件.

#### 我们知道,.ui文件其实是一个xml描述文件,C++编译器是不认的,所以在编译之前,Qt会将ui文件预处理成一个编译器能认的.h文件并包含到程序中,这样就能在提高效率的同时,让开发者对ui所见既得,更直观高效.

## pro文件语法

#### 我们可以看到,工程中还有个.pro文件,而之前使用命令行构建的时候也是先生成了一个pro文件.这个文件是干嘛用的呢:

#### .pro文件是qmake的工程文件,它列举了工程中包含的源文件以及各种资源文件.由于直接使用qmake命令来构建程序过于繁琐,所以我们大多数情况下会选择使用相对简单的pro文件,再使用qmake来生成对应的makefile.

#### 下面我们简单介绍一些基本的pro文件语法

#### 通过观察Qt Creater生成的工程文件,我们可以很容易得到一些规则.

#### 首先,pro文件的注释以#开头,在行尾处结束.一个条目通常是从一个变量开始,然后中间是一个操作符,后面是值.操作符可以是=,+=,-=等等,其他额外的操作符是=操作符的补充,比如+=代表扩展变量的值,-=代表从当前变量中移除某个值,后面的参数会覆盖前面的.

#### 比如Qt += core gui ,后面有跟上Qt -=gui,那么就代表着实际上只增加了core,gui就被后面删除掉了.

#### 下面我们介绍下几个常用的变量.

#### TARGET代表的是项目名称,这里不能有任何中文,不然编译的时候会报错,这个名字也是生成的程序的名字.

#### TEMPLATE表示工程文件的类型,这里是app,代表这是一个应用程序,其他还可以是lib,表示编译的是一个库,还有subdirs递归编译.

#### QT变量表示在程序中使用了哪些Qt模块,这里没有注明的Qt模块在后面的文件中也没法使用相应的类.

#### 还有SOURCES变量代表工程中的源文件,同理HEADERS代表头文件, FORMS代表ui文件,还有RESOURCES代表资源文件, OTHER\_FILES代表其他文件.

#### 另外,这个文件中省略了一行CONFIG += qt,因为这个是默认值所以不用写.CONFIG变量后面的值也就是程序编译的时候链接相应的库,如果值是qt,也就意味着会链接qt的lib路径,而QT变量则是对这个的进一步细化,如果QT后的值是core,则会链接Qt头文件路径中的QtCore lib文件.

#### 最后要说两个额外的操作符.\*=操作会在一个变量上增加一个值,但是只有在这个值不在变量列表上的时候才会生效,而~=操作符则使用特定的值替换符合正则表达式的值.

## Qt帮助文档与示例程序的使用

##### 帮助文档

#### 说到帮助文档,还是要感叹一下,Qt的帮助文档真心好用,几乎包括了Qt里面的所有内容,不仅仅是各种开发工具的使用,每个模块或者类的详细使用说明都有.不得不说,不会看帮助文档就做不好Qt开发.

#### 我们说过Qt提供了一个专门的帮助浏览器Qt Assistant,你可以用它很方便的阅读Qt文档.实际上,Qt Assistant是一个可配置并重新发布的文档阅读器,你可以用它方便的定制你自己的程序的帮助系统,可以同时提供本地文档和在线帮助,它支持快速查找,全文搜索,索引和书签,相当好用.

#### 我们也可以在Qt Creator中打开Qt的帮助文档,在Qt Creator左侧点击帮助,即可进入,里面有Qt所有开发工具,类,示例的使用说明.

#### 如果要搜索某一个类或者组件的用法,只需要在这里输入关键字就可以了,有关的都会列出来,相当详细.

##### 示例程序

#### 我们打开Qt Creater,在欢迎界面第一个看到的就是Qt的程序示例,这些示例与程序demo包含了Qt中很多模块的使用方法,具有很高的参考价值,有些也是很实用的小工具.比如这个程序就是一个用Qt Quick编写的计算器程序.我们点击它首先会打开这个程序的帮助文档,包括了详细的文件代码说明以及用到的模块.我们可以编译运行看看.

# Qt Designer与窗口组件

在上个课时中,我们演示了如何使用Qt Creator创建一个程序,同时讲解了Qt项目中各个文件的作用等等.本课时我们将主要讲解Qt Designer.

本课时内容包括:Qt Designer的使用,比如如何用Qt Designer创建一个UI文件,并对Qt中常用的UI组件做一个演示.

## Qt Design的使用

#### Qt提供了一个独立的Designer程序来创建UI,我们也可以使用Qt Creator中集成的设计师来使用Qt Designer,两者实际上是一样的.

#### 在Qt Creator中创建Ui文件,需要按下Command+n或者文件->新建项目或文件,在左侧选择文件和类下的Qt类别,然后选择新建Qt Designer Form,新建一个Qt设计师窗体,模板可以根据需求选择,这里我们选择MainWindow来演示,文件创建完成后会自动打开,进入设计模式.UI文件实际上是一个xml描述文件,只能在设计模式中修改.

#### 为了方便后面的运行演示,我们关闭这个新建的UI文件,打开上个课时创建的Hello,Qt项目,并打开mainwindow.ui文件.整个设计模式被分成了几个部分,从左到右是组建工具箱,包括了一些常用的Qt组件,只需要拖动到中间的窗体上就可以使用.下面是动作和信号槽的编辑器,在这里可以直接用一种直观的方式创建和修改简单的动作和信号槽,具体的我们会在后面的课程中讲解.右侧的工具箱中,上半部分是对象选择器,下半部分是属性修改器,你可以在这里查看并修改组件的各种属性,比如控制一个组件是否是可用的.在组件上点右键可以改变对象名称,工具提示和样式表.工具提示就是鼠标移动到组件上面的时候显示的内容,举个例子,我们拉出来一个按钮,要让他鼠标移过的时候显示这是一个按钮,那么我们可以这样(演示).

#### 此外,还可以修改组件的样式表,qss样式表跟css比较类似,包括语法,选择权,以及可使用的属性.我要让这个文本框变成灰色背景,蓝色字,那么可以这样(演示)…

#### 修改完成后,可以按下shift+option+R来方便的预览改动,直观高效.

#### 我们可以将程序编译运行看看,效果与预览的时候是一样的.

## UI组件演示

#### 下面我们将演示Qt中采用的UI组件.熟悉这些组件的使用可以提高开发效率,避免在需要用到的时候还要到处找资料,最好能做到信手拈来.

#### ……

### 按钮(不用念)

##### QPushButton(标题还是不用念)

#### 我们先来看按钮,最上面的布局跟弹簧我们一会再看.按钮顾名思义,这个分组下面包含了Qt中用到的几种按钮.PushButton这个是我们用的最多的普通按钮.我们把它拉出来演示下,OS X下按下shift+option+R可以快速预览,或者在菜单栏中选择工具->Form Editor窗体编辑,这里还能用其他风格来预览.至于为什么要叫PushButton,我们知道push的意思是推,我们用OS X下大家都会用到的系统设置来讲解下为什么要叫QPushButton,这样能给大家一个更好理解的概念.打开系统设置,我们看到最上面的几个按钮.前进,后退跟显示所有,仔细看可以看到他们是突出来的, 这就是PushButton.打开节能器设置页,恢复成默认这个按钮就跟Qt默认的一模一样了,确切的说应该是Qt的UI模拟的是当前平台的UI风格.另外,QPushButton还可以理解成是命令按钮.

##### QToolButton

#### 下面这个按钮是ToolButton工具按钮.这个按钮本质上与QPushButton没有什么不同,只不过ToolButton细化了使用场景,它是一个用来做详细设置或者命令的快速入口.一个例子就是节能器设置里面的定时按钮.

##### QRadioButton & QCheckButton

#### 后面是两个常用的选择按钮,radio button单选按钮跟check button复选按钮.我们可以用预览来看下效果.那么怎么实现radio button跟check button的二选一效果呢,我们分别再拉一个radio button跟一个check button出来,预览,我们可以看到radio button已经自动分组了,只能选择一个,你选择了另一个就会取消这个的选择,这是因为在同一个widget下面的radio button是自动分组的.而check button就不行,那么我们可以手动给他们分组.选中要分组的按钮,右键,指定到按钮组->新建按钮组,可以看到右上角对象中多了一个button group,在上面点右键或者在按钮组的成员上右键,可以看到多了一个选项,按钮组button group,我们可以选择打断按钮组或者添加新成员.现在预览看看效果,check button也只能二选一了, 是不是很方便.如果一个widget中有很多radio button,也可以用这个方法给他们手动分组.

##### QCommandLinkButton

#### 下面这个按钮不太常用,叫Command link button.这个按钮是在Windows Vista中引入的一个新的控制按钮,用途类似于一个单选按钮,你可以用它在一组互斥的选项中选择一个,而不是直接使用这个按钮本身.具体用法大家可以自己研究.

##### QButtonBox

#### 最后是ButtonBox,严格来说这是一个按钮组而不是单个按钮.它默认包含了两个按钮,取消跟确定,从这两个按钮我们可以看出来它一般是用在对话框中,比如设置页面,我们也可以通过代码的方式修改里面的按钮,比如帮助按钮之类的.ButtonBox会默认符合一个适合当前widget样式的布局,并且可以通过信号槽方便的传递数据.

### Item View & Item Widget

#### 下面我们来说说几个item组件.这里可以使用两类组件来显示item项,分别是view视图跟widget组件.两者在大多数情况下是一样的,但是我们往往使用widget就够了.我们先来看第一个List,也就是列表.它分别有list view 跟list widget两种.我们都拉出来看看区别.我们可以看到这时候是一样的,那么区别在哪呢,我们再双击看看,可以看到list view双击是修改对象名,而list widget是可以直接修改列表项.view跟widget的区别在于,view可以自己建立一个模型来显示,你需要去自己维护显示的model,我们可以看到这里已经注明了model-based,而widget则已经维护好了一个模型,你只需要添加项,所以是item-based.对于新手,我们不推荐使用view,因为大部分情况下使用item widget已经足够,而item view要求对model框架比较了解.,但是它的优势也是显而易见的, 可以很大程度上降低数据冗余,提高程序的效率.下面的几个Tree跟Table也是一样的.所以我们目前只对Item Widget做一个了解,Item View暂时我们还用不到.

#### 现在我们来看List widget.顾名思义,这是一个列表,我们直接来看效果吧.双击添加几组数据,我比较懒就用1234代替好了.我们可以看到这里每一项的属性都是可以单独修改的.而列表组件的属性里面也提供了很多高级属性,比如选择列表模式与图标模式.这些具体的我们会在以后用到的时候讲解.

#### Tree widger与Table widget分别是树状列表与表格,我们这边做一个简单演示.最后我们预览看下效果.

### Containers

#### 接下来我们看下Qt中的容器组件,这些还是比较常用的.容器组件用来在UI窗体中更好的布局,也使界面布局能够有条理,避免在内容多的时候会显的凌乱.你可以随意把其他组件拉到这几个容器中,但是这些组件只会在这个容器里面显示,不会超出范围.

#### 下面我们来看下这几个组件的效果.

##### GroupBox

#### 首先是GroupBox.这是一个带标题的组件框,有些地方也会翻译成组群框或者组合框.它跟其他容器组件的区别就在于它有一个标题.

##### Scroll Area

#### Scroll Area可以实现滚动条,在它里面显示的内容如果超出范围会通过滚动条的方式显示,而不像其他组件会隐藏.

##### ToolBox

#### ToolBox可以实现一种类似于抽屉的效果,就像designer左侧组件栏的组件分组.它提供了一个包含控制集合的可移动的面板.

##### TabWidget

#### TabWidget可以实现带标题的选项卡切换,每个选项卡中的内容都是独立的.

##### StackedWidget

#### Stacked Widget也是一个选项卡组件,不过与TabWidget不同的是它没有选项卡标题,我们可以通过程序自己来维护页面间的切换.这是一个相当常用的功能,举个例子就是OS X的系统设置,我们可以点击不同的选项进入不同的设置页面.

##### QFrame & QWidget

#### QFrame是Qt中的一个带有边框部件的基类,下面还有很多子类,都是带有边框的.不过在这里它是一个窗口组件,主要用来实现不同的边框效果,比如形状和阴影.而QWidget看起来与QFrame有些类似,实际上QWidget类是所有用户界面的基类, 他们是Qt用户界面的基本组件.我们可以不断在一个widget上创建一个新的widget,但是他们都是按Z轴排序的.

##### QMdiArea

#### MdiArea是Mainwindow的一个中央部件,这个组件提供了一个可以绘制多窗口的区域,我们可以用它方便的创建并管理多窗口,比如可以在一个文本编辑器中打开多个文档并可以在各个标签页来回切换

##### Dock Widget

#### DockWidget是一个比较特殊也是比较实用的组件.打个比方,Qt Designer中的对象几个窗口都是实用dock widget实现的.它提供了一个可以锁定在主窗口中或者作为一个顶层组件自由悬浮的窗口.大多数时候它被用来作为工具面板,比如在PhotoShop中,因为你可以让他停靠在主窗口中的上下左右随便哪一边,也可以让它悬浮.

### 输入组件 & 显示组件

#### 下面我们简单演示下Qt的输入组件跟显示组件,因为这些组件的功能比较直观,所以只需要做一个简单的功能效果演示就可以了,对应的类的更多功能以及类的成员,我们会在后面用到的时候再详细讲解.

##### ComboBox & FontComboBox

#### ComboBox是一个下拉列表,你可以在上面双击然后添加你想要加入的项.而FontComboBox则是一个用来选择字体的下拉列表,默认它会读取系统的字体列表让后让你选择.

##### Line Edit & Text Edit &PlainTextEdit

#### 这三种是文本编辑组件,LineEdit是一个单行的文本编辑框,你可以选择输入模式是普通模式,不显示输入内容,密码模式,它会显示圆点来代替输入的内容,还有一种明文密码模式,输入密码但是会明文显示.你还可以让他显示一个一键清除输入内容的按钮. Text Edit是一个强大的单页面多信息文本编辑器.它支持HTML标记,而且所见既得.而QPlainTextEdit正好相反,它是专门用来显示编辑纯文本文档的, 他们都针对大文档和快速输入做了优化.如果我们要用来显示一个文本文件中的内容的时候,就可以使用PlainTextEdit.

##### spinBox & Double Spin Box

#### 后面是两种数字输入组件,分别对应整数和小数.他们只能输入数字,并且提供了一个增加减小按钮,我们可以给他们设置最大最小数,设置小数的精度也就是允许几位小数,以及每次点击按钮的时候增加或者减小的数值.另外他们主动维护了针对键盘跟鼠标的事件,方便快速更改.

##### Time Edit & Date Edit & DateTimeEdit

#### 后面是三种事件日期输入组件.具体用法跟上面的数字输入组件类似,另外还可以设置显示的格式,而且支持日期选择的组件都额外提供了一个日历选择窗口用来代替原先的增大/减小按钮.

##### QDial

#### QDial提供了一个类似于仪表盘速度计的圆形范围控制器,你可以用它来做一个时钟,表盘,或者一个控制列表快速滚动的旋钮,类似于iPod classic上面经典的Scrollwheel.还是蛮好玩的.弄个比较好玩的东西,顺便讲下后面的进度条.大家可以看到用起来很方便.

##### Scroll Bars & Sliders

#### 这是四种用来拖动进度的条,我们可以姑且这么理解吧,具体使用场景不一样,但是不管怎么说,他们都是拿来拖动的而且可以表示进度.恩,具体没什么好说的,就做个简单的演示.

##### keySequenceEdit

#### 下面这个比较屌,一个很方便的组件,专门用来输入快捷键,你输入组合键他会识别出来并且显示你输入的键.其他的用法暂时没有发现.我只能说这东西好任性….

##### Label & TextBrowser

#### 下面这些都是用来显示的了.显示组件用户只能看,不能操作.

#### Label可以用来显示文本或者图像,包括各种格式的图片,TextBrowser则跟QTextEditor比较类似,只不过没有了输入功能,它同样可以显示各种HTML标记.

##### graphicsView

#### graphicsView这个比较高级,它可以用来显示大量的2D图像,并且支持,比如旋转缩放.这个暂时我们不会用到.

##### Calender

#### Calender日历组件就是显示一个日历,上面我们讲到日期输入框的时候提到过,它下拉的日历选择其实就是这个.

##### LCDNumber

#### LCDNumber专门用来显示数字和一些特殊的符号,用的是一种LCD显示屏的风格,类似计算器.

##### ProgressBar & Line

#### 进度条上面我们演示过了,它会根据当前平台来显示不同的风格,比如Windows上就是一个很粗的绿色….

#### 下面我要重点讲一下这两根横线跟竖线,因为他们真的没什么好说的!你没看错,这就是两条线,什么都干不了.当然有着两条线也确实能方便很多,至少我们不用去在要用到的时候画两条线出来.简单理解就是类似于分割线,你还可以给它设置阴影.

##### OpenGLWidget & declarativeView & quickWidget & webView

#### 最后的几个组件只需要了解就行了,OpenGL Widget用来处理跟Open GL 3D绘图显示有关的, declarativeView可以显示 Qt Declarative 用户界面, Qt Declarative在触屏以及嵌入式设备方面更友好, quickWidget可以用来显示一个Qt Quick的用户界面, webView可以用来编辑显示html文档.显示的文档可以是本地的也可以是一个链接或者域名.是Qt WebKit的一个主要组件.

### 布局

#### 最后我们回过头说下布局.使用布局可以方便的构建一个排列整齐的应用界面,类似于Java Swing中的布局,不过Qt提供的布局要少一些,只有这四种,分别是纵向布局横向布局格栅布局跟表格布局.你可以从左边工具栏中拉一个布局出来然后再把组件拖进去组成布局,也可以选中组件然后点击上面的创建布局按钮,或者选中以后右键,布局.而下面的两种spacers弹簧你可以理解成是在布局中使用的一个空白控件,会自动根据大小样式以及布局元素调整高度或者宽度.更多关于布局的内容我们会在后面再深入讲解,现在只要会简单使用就可以了.

## 结尾彩蛋

#### 在课程的最后给大家带来一个结尾彩蛋,我们打开课时1生成的源码文件,将setText的内容改成”<h1 style=”color: red;”>Hello,Qt</h1>然后运行,查看效果,可惜我没有美工细胞,大家可以自己去试一试创建一个更加美观的Hello,Qt出来.至于这是为什么,我们会在下个课程中讲解.