# 快速上手Qt Creator

Qt 4.8.7/Qt Creator/C++/ubuntu16.04

* 第一个Qt程序
* 常用窗口部件
* 布局管理
* 样式外观
* 信号与槽
* Qt编程示例

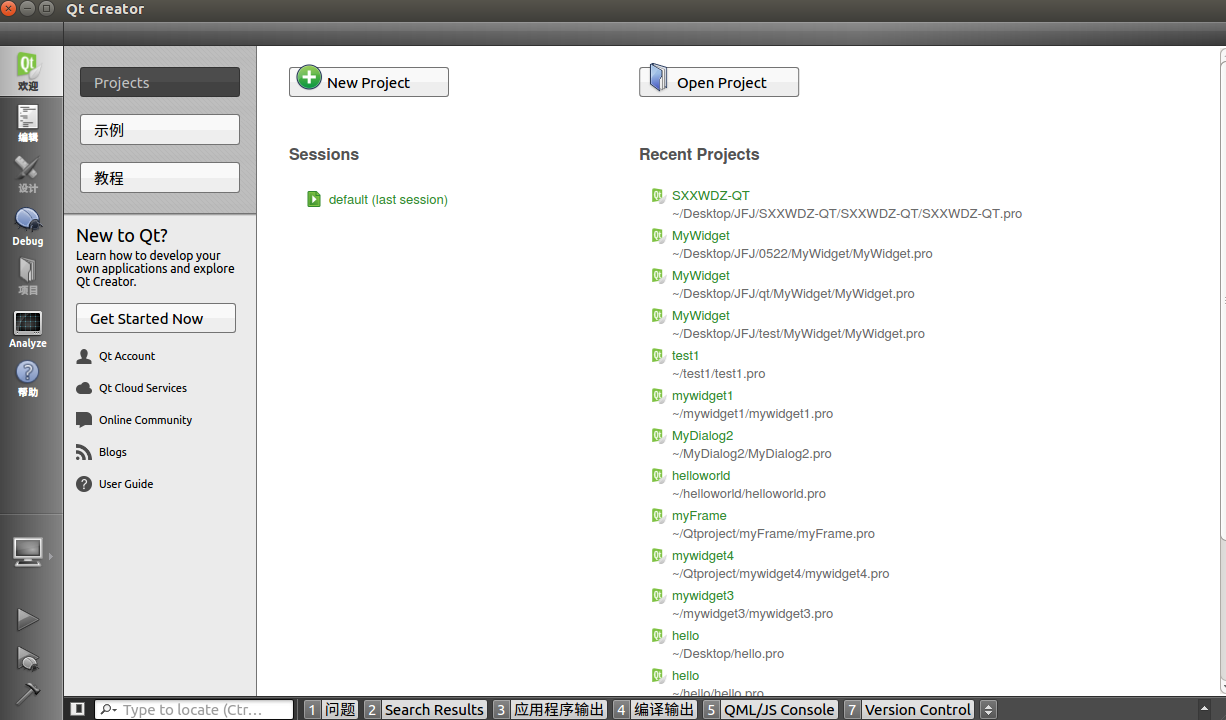
作 者：金凡杰

# 第一个Qt程序

Qt Creator是一个强大的跨平台IDE集编辑、编译、运行、调试功能于一体。

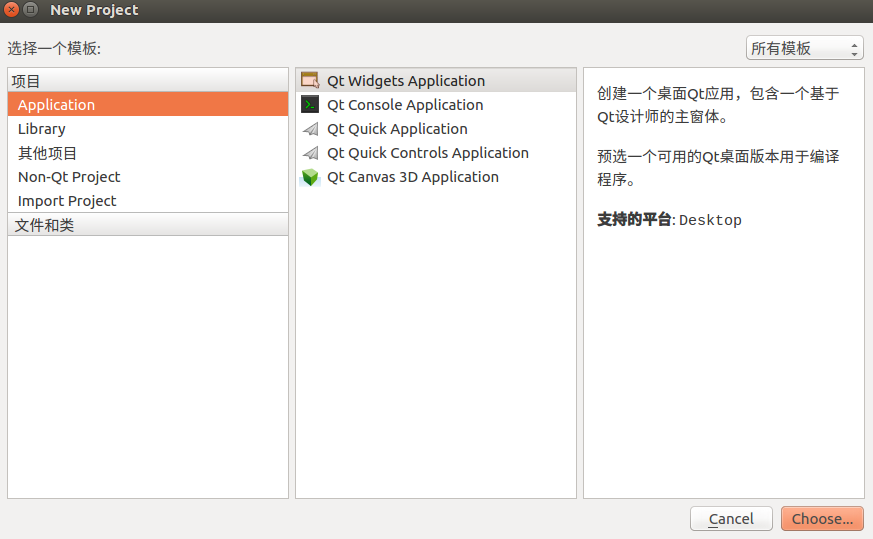
通过以下命令打开Qt Creator。

捕获1



**如果第一次使用请先配置Qt Creator（包括给Qt Creator设置Qt版本、编译器等，各类Qt书籍和网上均有方法，这里不做过多说明）。**

单击New Project，出现下面画面



选择Qt Widgets Application，点击Choose

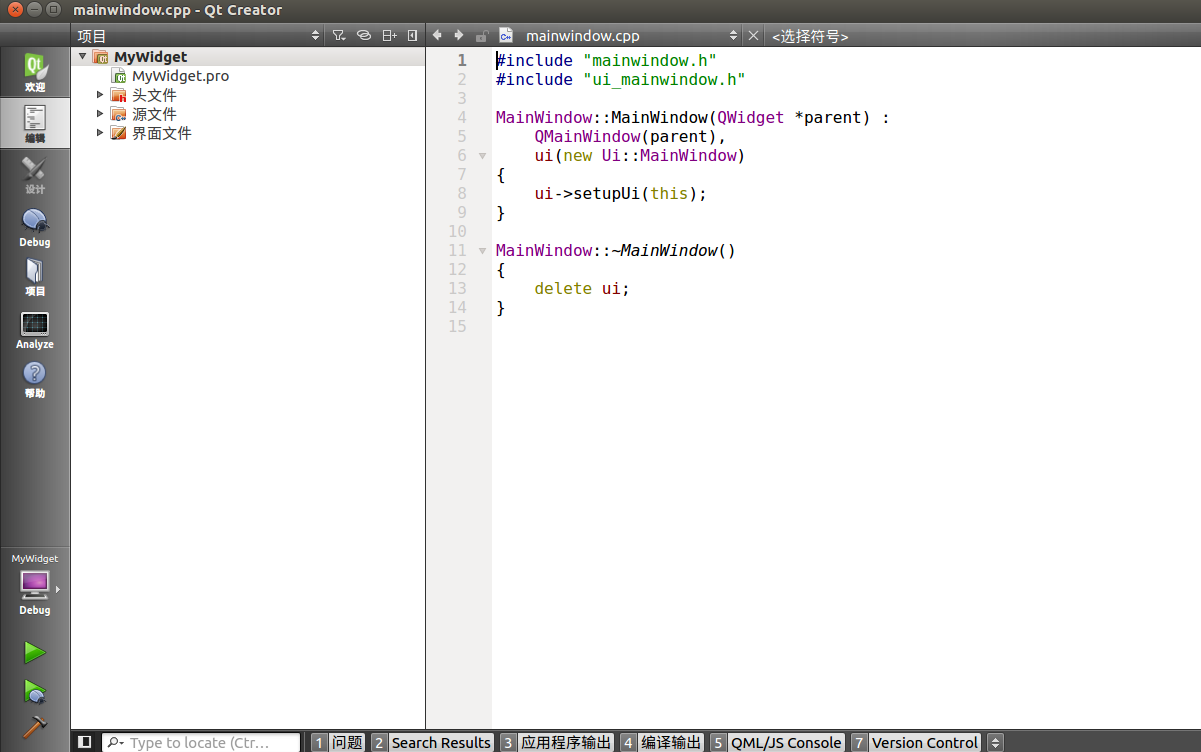


选择名称，路径后，点击下一步。

然后可以一路选择默认方式，点击下一步，直到出现下图

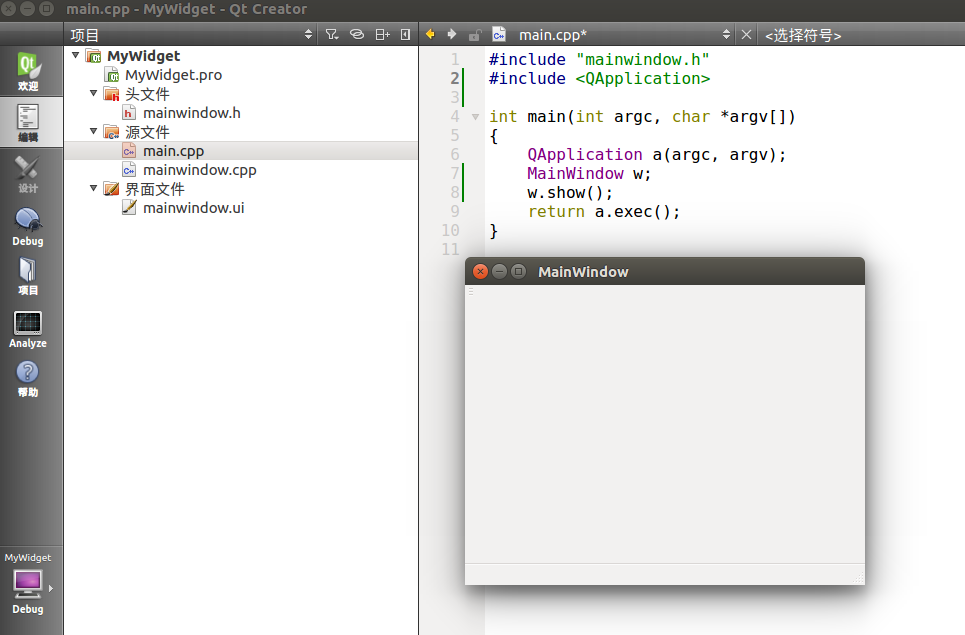


点击完成。



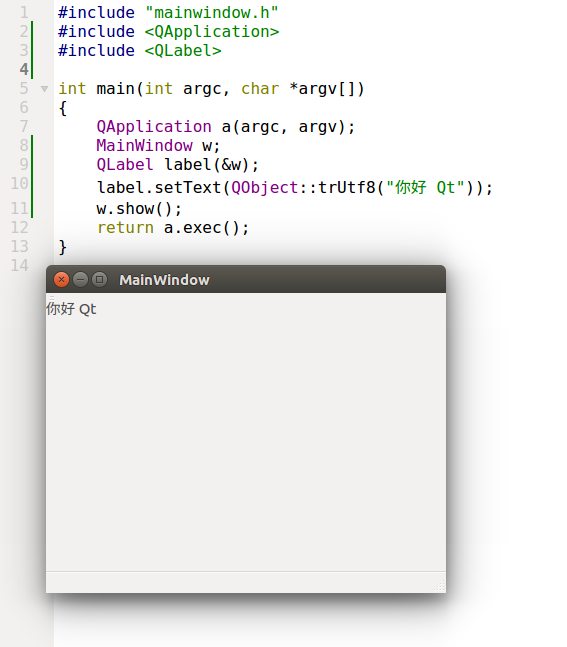
点击左下角的运行按钮，会弹出一个空白窗口。

点开源文件，看见main.cpp里面有如下代码



关闭MainWindow窗口，我们按照下列代码在窗口中添加一个文本“你好 Qt”。



Qt中的每一个类都有其同名的头文件，第三行添加了<QLabel>为使用QLabel类。**第九行label指定了父窗口为w，所以它是子部件（如果有父窗口，那它就是子部件，否则就是独立的窗口）。**第十行设置label对象的文本。默认情况下，新建的对象是不可见的，要用show()函数将他们显示出来。

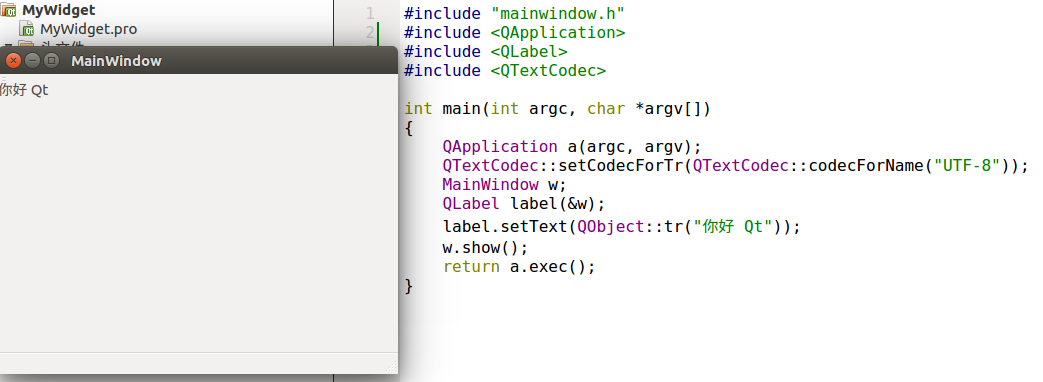
我们看到，中文在窗口中没有显示出来，这时候如果我们在括号里加上QObject::trUtf8，让它按UTF-8字符集进行显示，就可以使用中文了。

第7行QApplication xxx(argc,argv)创建一个QApplication对象，用来管理整个应用程序用到的资源，Qt支持命令行参数，QApplication构造函数需要两个参数argc，argv。

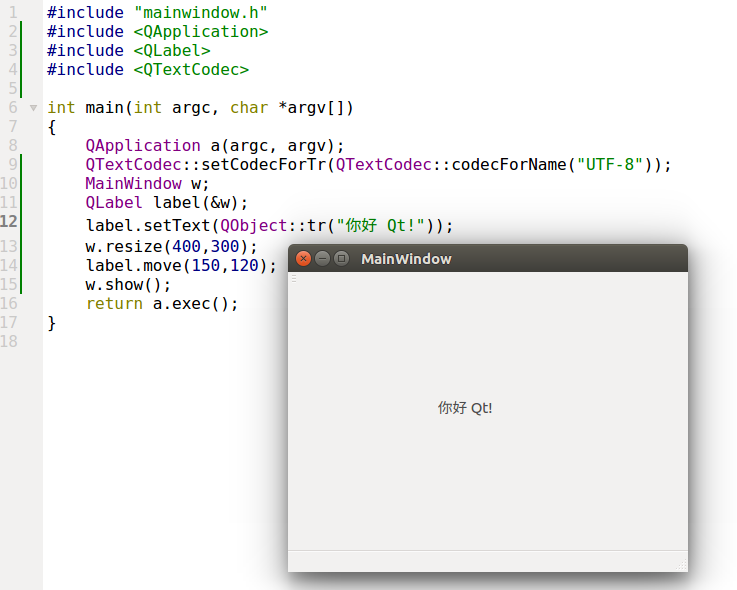
第12行 return xxx.exec()程序将进入一个循环，等待用户动作。例如鼠标点击和按键等操作。

这两行是每个Qt程序不可少的部分。

如果大量的文本都要使用UTF-8字符集，就使用以下办法



QTextCodec类提供了文本编码的转换功能，setCodecForTr函数用来设置QObject::tr使用的字符集。**Qt上所有显示到屏幕上的字符串最好都用tr括起来。**

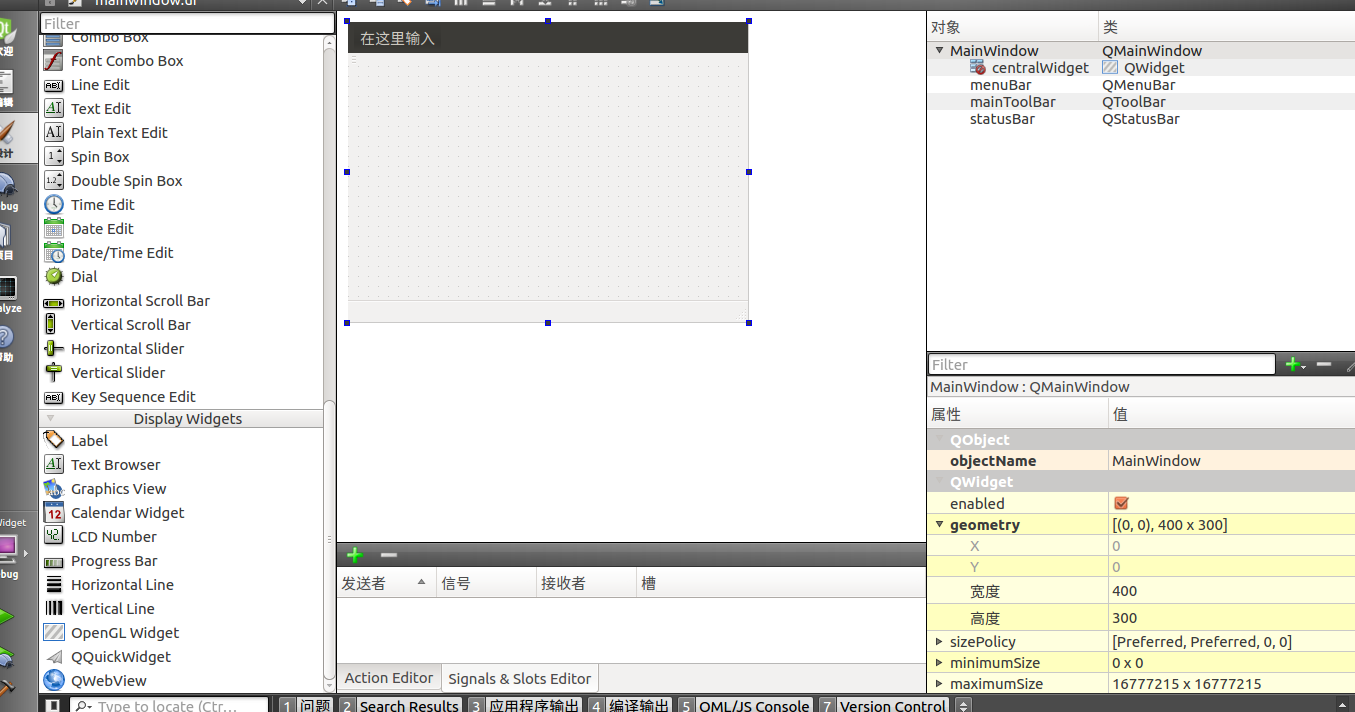


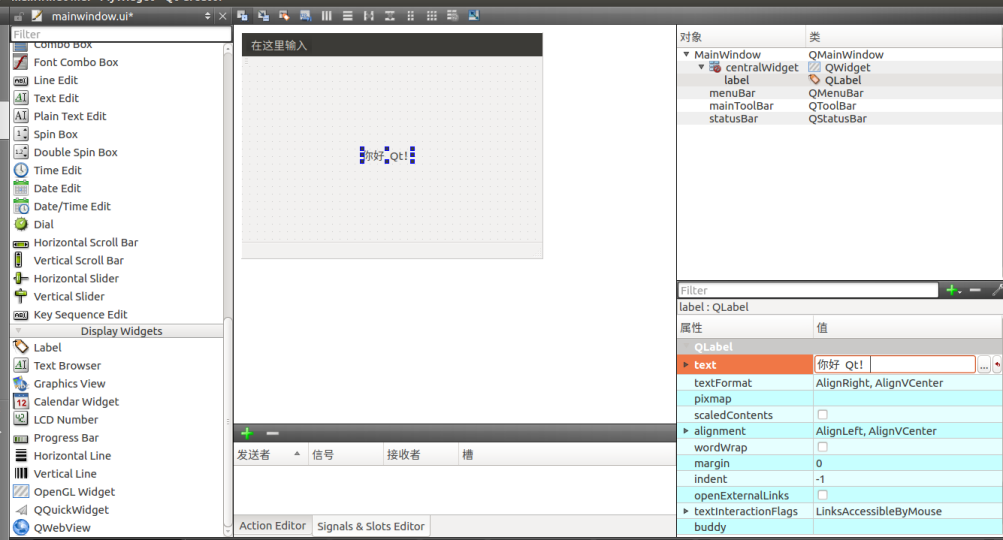
**第12行label.setText使用了 ” . ” ，说明setText是QLabel类的成员函数。第13、14、15、16都用到了成员函数，13行的resize用来设置窗口大小（单位是像素），move是设置窗口位置（单位也是像素）。**

屏蔽跟label有关的代码，使用可视化编辑器做相同的事情。双击mainwindow.ui界面文件。

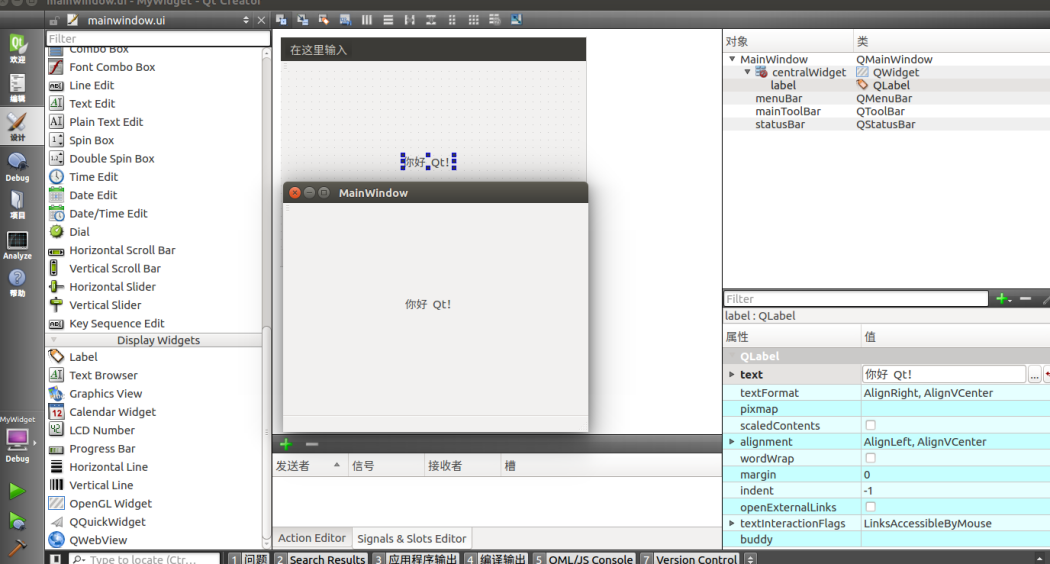
捕获12

双击mainwindow.ui以后出现以下窗口

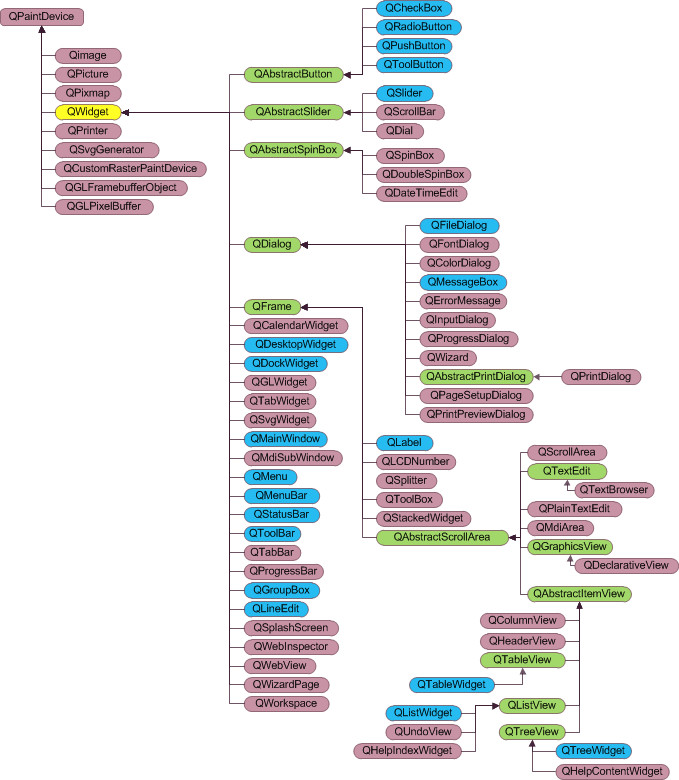


拖动左边侧栏的Label控件到中间的主页面中，然后点击Label控件设置文本。

运行。



# 常用窗口部件

****

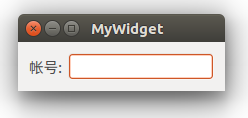
**窗口一般是没有父部件的部件，又叫顶级部件（一般用QWidget、QDialog、QMainWindow作为顶级部件即窗口）。相对的就是子部件，Qt中大部分部件都用作子部件嵌入在别的窗口中（按钮、标签、滑块、文本输入框等）。**

## **捕获3按钮**

1. #include <mainwindow.h>
2. #include <QApplication>
3. #include <QPushButton>
5. **int** main(**int** argc, **char** \*argv[])
6. {
7. QApplication a(argc, argv);
8. QWidget \*widget = **new** QWidget;
9. QPushButton \*button = **new** QPushButton("Quit",widget);//创建Quit按钮
10. widget->resize(300,150);//设置窗口大小
11. button->move((widget->width()-button->width())/2,(widget->height()-button->height())/2);//让按钮移动到窗口中间
12. widget->show();
13. **int** rec = a.exec() ;
14. **delete** widget ;
15. **return** rec ;
16. }

由于现在的程序未安排代码处理按钮的点击事件，所以当按钮按下时什么也不会发生。**而信号与槽能很方便地对按钮的点击和类似事件进行处理。**

## **标签和文本框**

1. #include "mainwindow.h"
2. #include <QApplication>
3. #include <QHBoxLayout>
4. #include <QLabel>
5. #include <QLineEdit>
6. #include <QTextCodec>
8. **int** main(**int** argc, **char** \*argv[])
9. {
10. QApplication a(argc, argv);
11. QTextCodec::setCodecForTr(QTextCodec::codecForName("UTF-8"));//设定字符集为UTF-8
13. QWidget \*widget = **new** QWidget ;
14. QLabel \*labelaccount = **new** QLabel(QObject::tr("帐号:"),widget);//创建帐号标签
15. QLineEdit \*lineaccount = **new** QLineEdit(widget) ;//创建帐号文本输入框
17. QHBoxLayout \*layoutaccount = **new** QHBoxLayout ;//创建水平布局管理器
18. layoutaccount->addWidget(labelaccount);
19. layoutaccount->addWidget(lineaccount);//从左至右放置部件
21. widget->setLayout(layoutaccount);
22. widget->show();
23. **int** rec = a.exec() ;
24. **delete** widget ;
25. **return** rec ;
26. }

为了使布局整齐，17行用到了布局管理器（第三部分布局管理会讲到）。

**24行：当你释放了一个父对象（父窗口）时，系统会自动释放它所有的子对象（子部件）。**

## **捕获2微调框与滑块**

1. #include "mainwindow.h"
2. #include <QApplication>
3. #include <QHBoxLayout>
4. #include <QSpinBox>
5. #include <QSlider>
7. **int** main(**int** argc, **char** \*argv[])
8. {
9. QApplication a(argc, argv);
11. QWidget \*widget = **new** QWidget ;
12. QSlider \*slider = **new** QSlider(Qt::Horizontal,widget);//创建滑块
13. QSpinBox \*spinbox = **new** QSpinBox(widget);//创建微调框
14. spinbox->setRange(0,100);//微调框设置有效范围
15. slider->setRange(0,100);//滑块设置有效范围
17. QHBoxLayout \*layout = **new** QHBoxLayout ;//创建水平布局管理器
18. layout->addWidget(spinbox);
19. layout->addWidget(slider);//从左至右放置部件
21. QObject::connect(spinbox,SIGNAL(valueChanged(**int**)),slider,SLOT(setValue(**int**)));
22. QObject::connect(slider,SIGNAL(valueChanged(**int**)),spinbox,SLOT(setValue(**int**)));
24. widget->setLayout(layout);
25. widget->show();
26. **int** rec = a.exec() ;
27. **delete** widget ;
28. **return** rec ;
29. }

为了让微调框与滑块同步，第21、22行用到了信号与槽关联这两个部件，当一个部件值改变了，就会发射valueChanged(int)信号，而另一个部件就会用这个新值调用它的setValue(int)槽（第五部分信号与槽会详细介绍）。

同样，为了使布局整齐，17行用到了布局管理器，这里使用的是水平布局，当使用addWidget()添加部件时就会从左至右放置部件。

### Qt Creator提供的默认基类只有QMainWindow、QWidget和QDialog三种，而这三种窗体也是以后用得最多的，QMainWindow是带有菜单栏和工具栏的主窗口类，QDialog是各种对话框的基类，而它们全部继承自QWidget。不仅如此，其实所有的窗口部件都继承自QWidget。

# 布局管理

### 布局管理器

QHBoxLayout：在水平方向排列窗口部件，从左到右；  
QVBoxLayout：在垂直方向排列窗口部件，从上到下；  
QGridLayout：把各个窗口部件排列在一个网格中。



1. #include "mainwindow.h"
2. #include <QApplication>
3. #include <QHBoxLayout>
4. #include <QLabel>
5. #include <QLineEdit>
6. #include <QPushButton>
7. #include <QTextCodec>
9. **int** main(**int** argc, **char** \*argv[])
10. {
11. QApplication a(argc, argv);
12. QTextCodec::setCodecForTr(QTextCodec::codecForName("UTF-8"));//设定字符集为UTF-8
14. QWidget \*widget = **new** QWidget ;
15. QLabel \*labelaccount = **new** QLabel(QObject::tr("帐号:"),widget);//创建帐号标签
16. QLabel \*labelpassword = **new** QLabel(QObject::tr("密码:"),widget);//创建密码标签
17. QLineEdit \*lineaccount = **new** QLineEdit(widget) ;//创建帐号文本输入框
18. QLineEdit \*linepassword = **new** QLineEdit(widget) ;//创建密码文本输入框
19. QPushButton \*buttonenter = **new** QPushButton(QObject::tr("确认"),widget);//创建确认按钮
20. QPushButton \*buttonquit = **new** QPushButton(QObject::tr("退出"),widget);//创建退出按钮
22. QHBoxLayout \*layoutaccount = **new** QHBoxLayout ;//创建水平布局管理器
23. layoutaccount->addWidget(labelaccount);
24. layoutaccount->addWidget(lineaccount);//从左至右放置部件
26. QHBoxLayout \*layoutpassword = **new** QHBoxLayout ;
27. layoutpassword->addWidget(labelpassword);
28. layoutpassword->addWidget(linepassword);
30. QVBoxLayout \*layoutleft = **new** QVBoxLayout ;//创建一个垂直布局管理器
31. layoutleft->addStretch();//添加分隔符
32. layoutleft->addLayout(layoutaccount);
33. layoutleft->addLayout(layoutpassword);
34. layoutleft->addStretch();
36. QVBoxLayout \*layoutright = **new** QVBoxLayout ;
37. layoutleft->addStretch();
38. layoutright->addWidget(buttonenter);
39. layoutright->addWidget(buttonquit);
40. layoutright->addStretch();
42. QHBoxLayout \*layoutmain = **new** QHBoxLayout ;
43. layoutmain->addLayout(layoutleft);
44. layoutmain->addLayout(layoutright);

47. widget->setLayout(layoutmain);
48. widget->show();
49. **int** rec = a.exec() ;
50. **delete** widget ;
51. **return** rec ;
52. }

**第31、34、37、40行添加了分隔符（或称伸展器），用它来占据剩余的空白区域，这样可以使窗口增大或缩小时，部件仍处于中间位置，不会随窗口变高变低而变化。**

**布局管理器既可以包含子部件，也可以包含子布局（例如第32、33、43、44、行），通过布局的不同嵌套组合，就可能构建相当复杂的布局层次。**

# 样式外观

### 捕获5setStyleSheet()设置窗口样式

1. #include <mainwindow.h>
2. #include <QApplication>
3. #include <QPushButton>
5. **int** main(**int** argc, **char** \*argv[])
6. {
7. QApplication a(argc, argv);
8. QWidget \*widget = **new** QWidget;
9. QPushButton \*button = **new** QPushButton("Start NewGame",widget);
10. widget->resize(300,150);//设置窗口大小
11. widget->setStyleSheet("background-color:rgba(247,192,193,255);");//设置样式
12. button->setStyleSheet("QPushButton{background-color:rgba(242,141,102,255);\
13. color:rgba(50,25,25,255);\
14. border:2px groove rgb(242,141,0);border-radius:20px;padding:2px 2px;\
15. background-color: qconicalgradient(cx:0.5, cy:0.522909, angle:179.9, stop:0.494318 rgba(242,141,102,255), \
16. stop:0.5 rgba(247,192,193,255));}"
17. "QPushButton:pressed{background-color:rgba(200,200,200,255);}");
18. button->move((widget->width()-button->width())/2,(widget->height()-button->height())/2);//让按钮移动到窗口中间
19. widget->show();
20. **int** rec = a.exec() ;
21. **delete** widget ;
22. **return** rec ;
23. }

setStyleSheet()功能非常强大，样式繁多。

### 捕获6setFont()设置字体

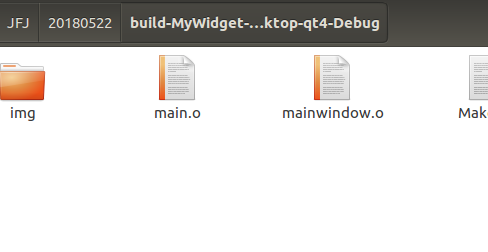
1. #include <mainwindow.h>
2. #include <QApplication>
3. #include <QPushButton>
5. **int** main(**int** argc, **char** \*argv[])
6. {
7. QApplication a(argc, argv);
8. QWidget \*widget = **new** QWidget;
9. QPushButton \*button = **new** QPushButton(QObject::trUtf8("新游戏"),widget);
10. widget->resize(300,150);//设置窗口大小
11. button->resize(100,50);
12. widget->setStyleSheet("background-color:rgba(247,192,193,255);");//设置样式
13. button->setStyleSheet("QPushButton{background-color:rgba(242,141,102,255);\
14. color:rgba(50,25,25,255);\
15. border:2px groove rgb(242,141,0);border-radius:20px;padding:2px 2px;\
16. background-color: qconicalgradient(cx:0.5, cy:0.522909, angle:179.9, stop:0.494318 rgba(242,141,102,255), \
17. stop:0.5 rgba(247,192,193,255));}"
18. "QPushButton:pressed{background-color:rgba(200,200,200,255);}");
19. QFont fontbutton(QObject::trUtf8("楷体"),20,75);//楷体，20号字体，加粗
20. button->setFont(fontbutton);
21. button->move((widget->width()-button->width())/2,(widget->height()-button->height())/2);//让按钮移动到窗口中间
22. widget->show();
23. **int** rec = a.exec() ;
24. **delete** widget ;
25. **return** rec ;
26. }

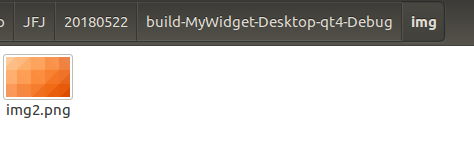
### 为窗口添加图片背景

第一步：添加资源文件。

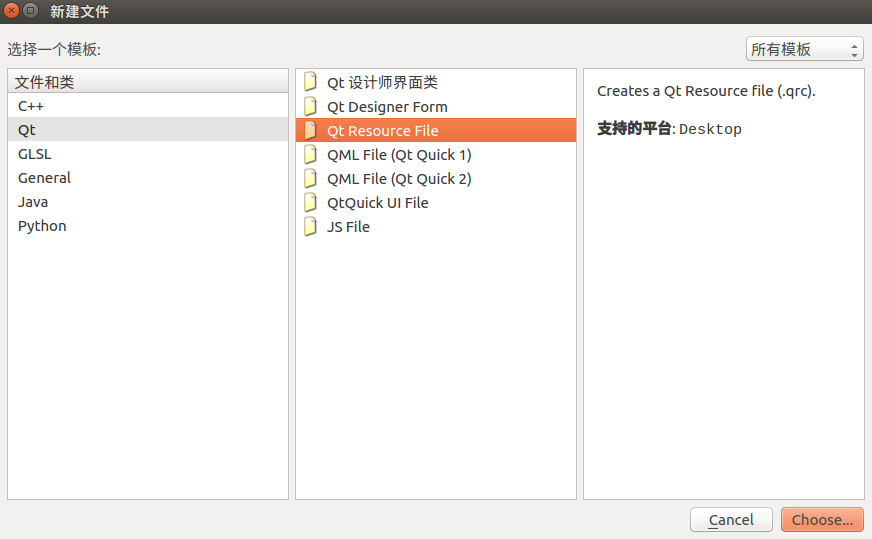
在你工程的当前目录里建一个文件夹，然后把图片放进这个文件夹里。



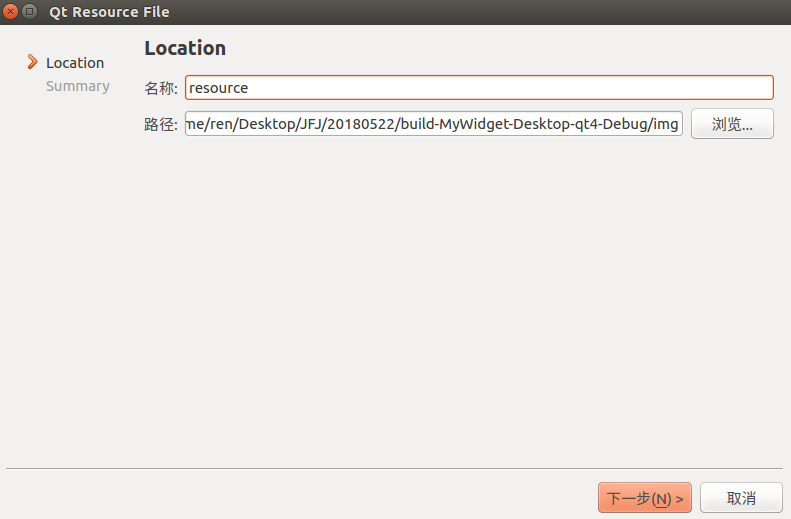




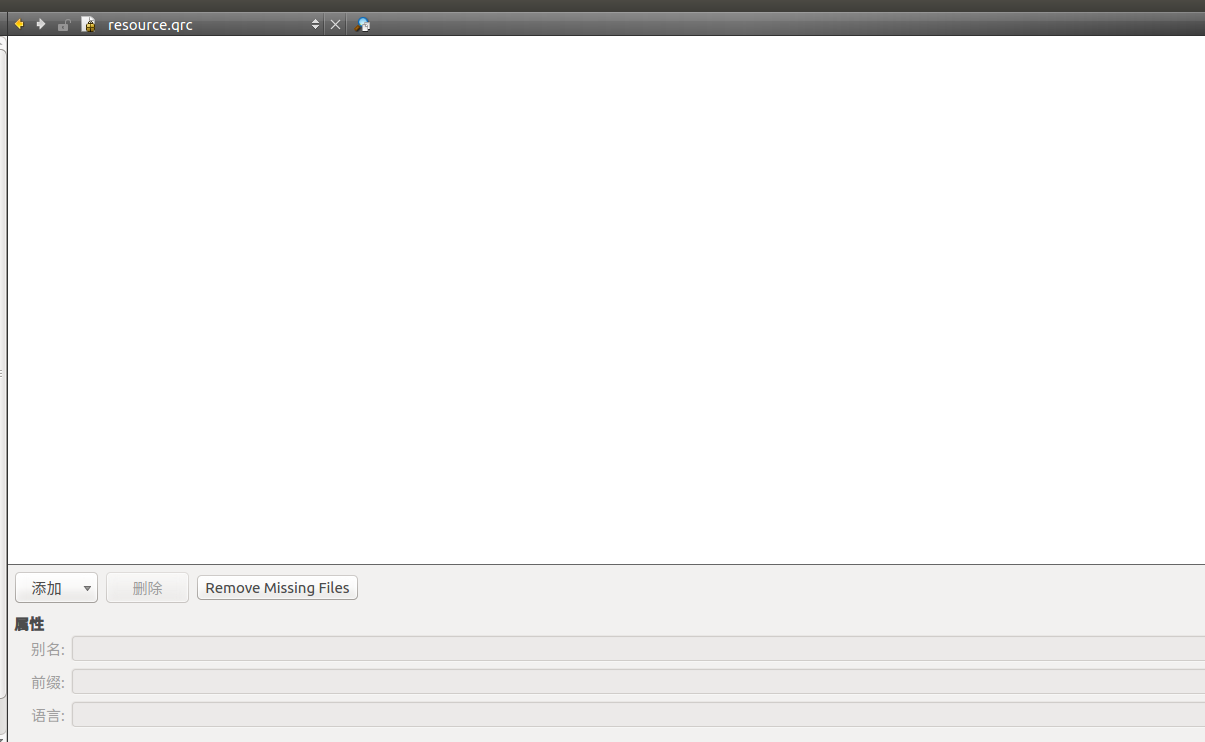
回到Ct Creator，右键自己的工程，添加新文件，出现以下窗口



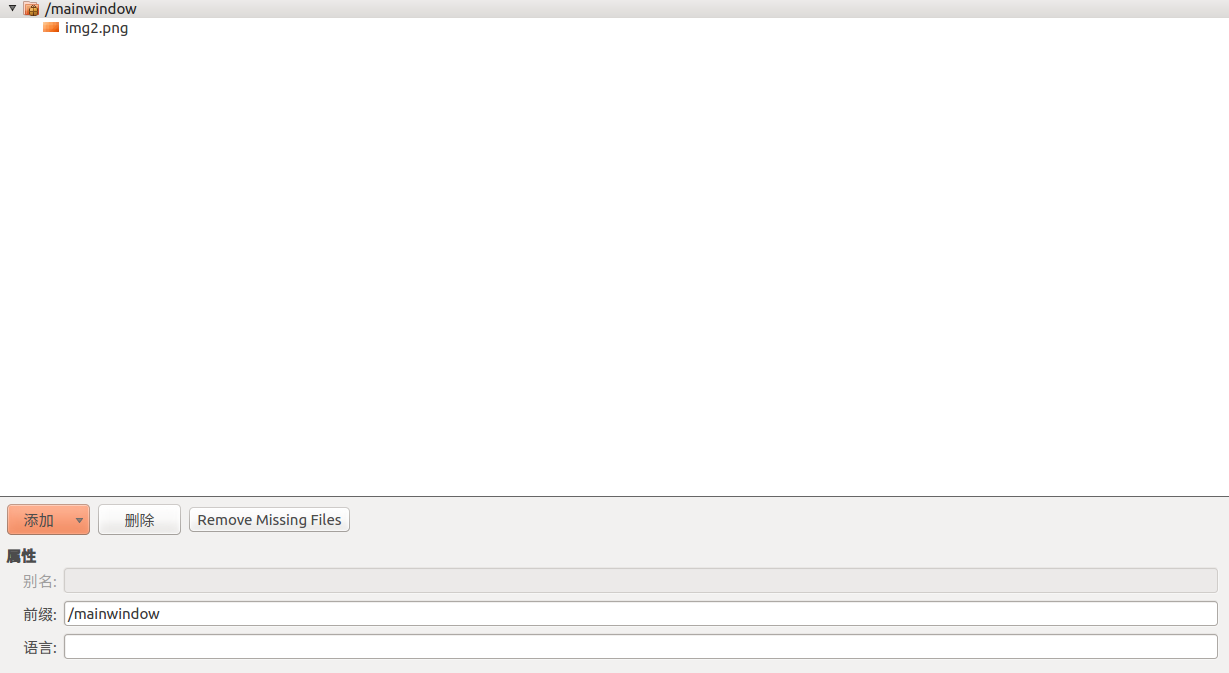
选好Qt Resource File后点击Choose，出现以下窗口



名称自取，路径选择刚才你放置图片的文件夹，点击下一步，然后出现的窗口就选择默认方式，直到出现以下画面



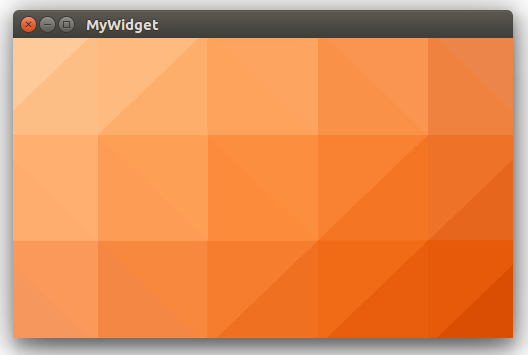
点击添加前缀（自取），前缀添加后再次点击添加，出现添加文件，点击添加文件，选择你的图片文件，这样资源文件就添加好了。



然后就可以在代码中使用图片了。

1. #include <mainwindow.h>
2. #include <QApplication>
3. #include <QPushButton>
5. **int** main(**int** argc, **char** \*argv[])
6. {
7. QApplication a(argc, argv);
8. QWidget \*widget = **new** QWidget;
9. widget->resize(500,300);
11. QPixmap pixmap(":/mainwindow/img2.png");
12. QPalette palette = widget->palette();
13. palette.setBrush(QPalette::Background,QBrush(pixmap.scaled(widget->size())));
14. widget->setPalette(palette);//给窗口添加图片背景

17. widget->show();
18. **int** rec = a.exec() ;
19. **delete** widget ;
20. **return** rec ;
21. }



### 窗口模式

**setWindowFlags(Qt::FramelessWindowHint);//设置无边窗口**

**setWindowFlags(Qt::WindowStaysOnTopHint);//设置窗口最上**

**setWindowState(Qt::WindowFullScreen);//设置窗口全屏**

**setModal(true);//设置模态**

**setFixedSize(const QSize&);//设置窗口固定大小**

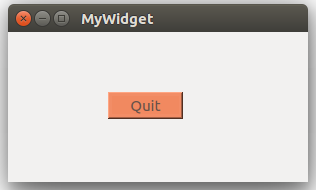
**setFocusPolicy(Qt::NoFocus);//无焦点**

**setAttribute(Qt::WA\_TranslucentBackground);//设置透明**

# 信号与槽

信号与槽可以实现两个对象之间的通信。例如：如果点击了关闭按钮，我们希望窗口调用close()函数退出。

信号与槽简单来说就是，一个事件发生会发出一个信号，收到这个信号后，会调用槽函数。下面看一个简单例子。



1. #include <mainwindow.h>
2. #include <QApplication>
3. #include <QPushButton>
4. #include <QDesktopWidget>
6. **int** main(**int** argc, **char** \*argv[])
7. {
8. QApplication a(argc, argv);
9. QWidget \*widget = **new** QWidget;
10. QPushButton \*button = **new** QPushButton("Quit",widget) ;
11. widget->resize(300,150);
12. button->move((widget->width()-button->width())/2,(widget->height()-button->height())/2);
13. button->setStyleSheet("background-color:rgba(242,141,102,255);");
15. QObject::connect(button,SIGNAL(clicked(**bool**)),widget,SLOT(close()));//信号与槽
17. widget->show();
18. **int** rec = a.exec() ;
19. **delete** widget ;
20. **return** rec ;
21. }

通过15行的QObject::connect（）函数，当我们按下按钮button的时候，就会关闭widget窗口了。

它的函数原型如下

bool QObject::connect(const QObject \* sender ,

const char \* signal,

const QObject \* receiver ,

const char \* method , Qt::ConnectionTypetype=Qt::AutoConnection)

第一个参数是发送信号的对象，第二个参数是发送的信号，第三个参数是接受信号的对象，第四个参数是槽函数。

对于信号与槽，必须使用SIGNAL()和SLOT()宏，它们可以将其参数转化为const char \*类型。conncet函数的返回值为bool类型，当关联成功时返回true。

注意：

* **信号与槽的参数只能有类型，不能有变量。**
* **基本原则要求信号的参数类型和槽的参数类型一致。信号的参数个数可以多于槽，但不能反过来。**
* **一个信号可以关联多个槽**
* **多个信号可以关联一个槽**
* **一个信号可以关联另一个信号**
* **信号没有返回值，只能是void类型**

1. 自定义信号与槽

简单实现一个程序：在一个窗口中创建一个对话框，在这个对话框中可以输入数值，当单击“确定”按钮时关闭对话框并将对话框的数值通过信号发送出去，在另一个窗口中接收信号并显示该数值。

新建项目，基类选择QWidget，类名选择Widget不变。项目建立完成后，向项目中添加新文件，模板选择Qt中的“Qt设计师界面类”，界面模板选择“Dialog without Buttons”，类名选择MyDialog。完成后首先在mydialog.h中添加一个信号

捕获21

信号只用声明，不需要也不能对它进行实现。声明信号用singals关键字，signals前不能加private、public、protect限定符。

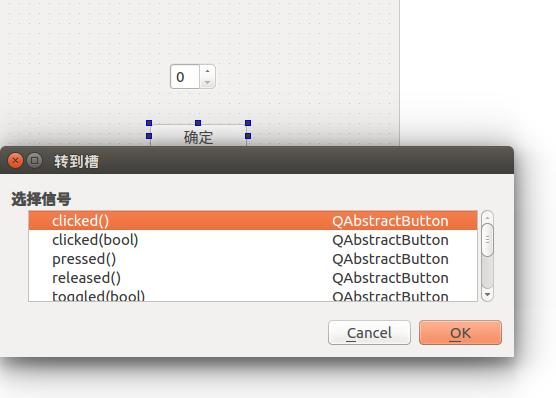
在widget.h中加入自定义的槽

捕获22

声明槽函数用slots关键字，slots前可以加private、public、protect限定符。

在mydialog.ui对应的界面中拖取一个PushButton 和一个Spinbox,控件名改为button和spinbox,button的text设置为“确定”。

右键确定按钮，选择“转到槽”，出现下图，选择clicked()。



点击OK。在出现的槽函数中加入如下代码

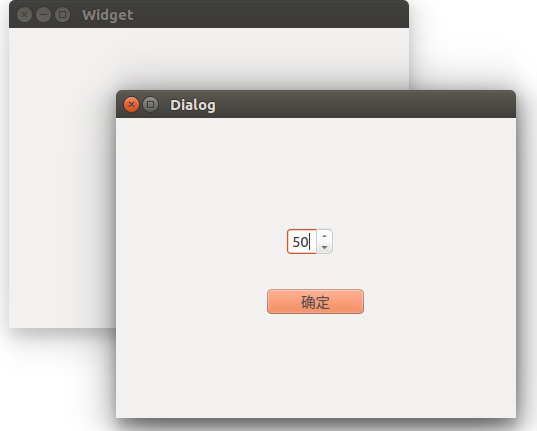
1. **void** MyDialog::on\_button\_clicked()
2. {
3. **int** value = ui->spinbox->value() ;
4. emit ValueReturn(value) ;//发送信号用emit关键字
5. close();
6. }

打开widget.ui界面，向界面上拖取一个Label控件，大小可调。然后往widget.cpp中加入 “mydialog.h”头文件，再往构造函数中加入如下代码

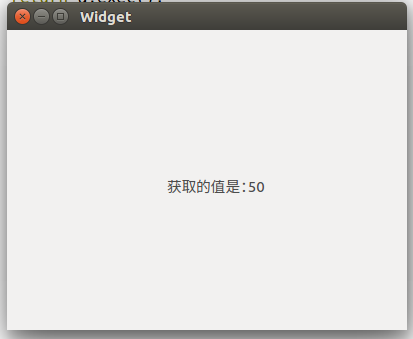
1. MyDialog \*dialog = **new** MyDialog(**this**) ;
2. QObject::connect(dialog,SIGNAL(ValueReturn(**int**)),**this**,SLOT(ShowValue(**int**)));
3. dialog->show();

在widget.cpp定义槽函数ShowValue(int)

1. **void** Widget::ShowValue(**int** value)
2. {
3. ui->label->setText(trUtf8("获取的值是：%1").arg(value));
4. }

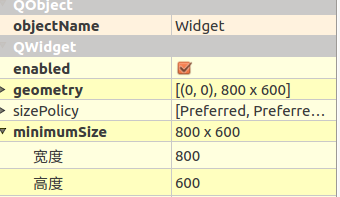
运行，在微调框中输入50。

点击确定。



# Qt编程示例

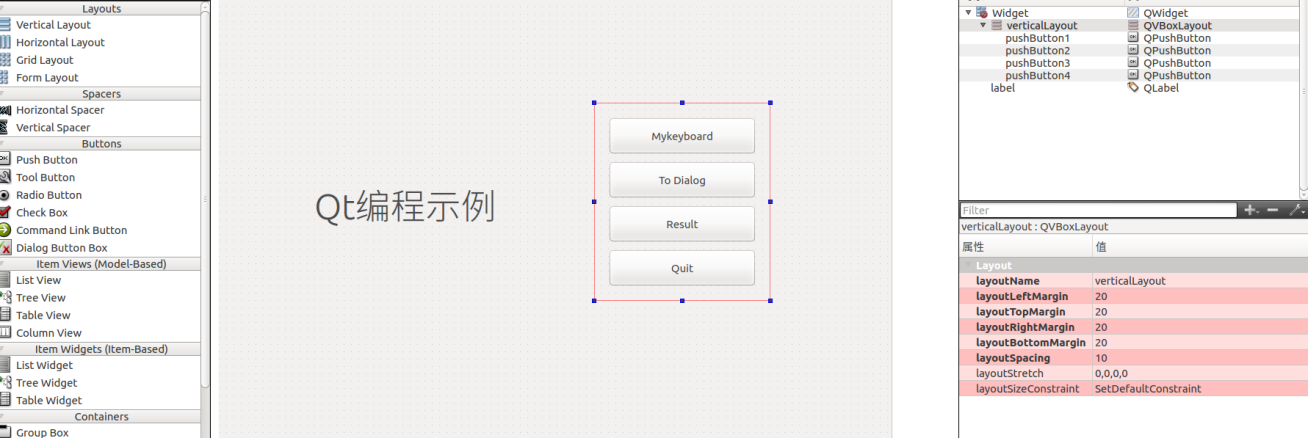
# 创建一个项目，基类选择QWidget，类名用Widget。创建完成后进入widget.ui界面。



点击主页面，找到右下角的minimumSize大小和geometry里面的大小，都设置800x600。

从左侧栏拖出一个Label控件和四个PushButton控件，Label控件设置窗口最小值和geometry的大小都为300x100,

PushButton控件都设置为200x50。右键Label控件，选择“改变样式表”，在窗口中输入font: 57 36pt "Ubuntu";Label文本更改为“Qt编程示例”，按钮的文本改为”Mykeyboard”、”To Dialog”、”Result”、”Quit”如图所示

从左侧栏拖出一个Vertical Layout，把四个按钮放进去，然后在右下角的vertical layout属性设置修改为如图所示

捕获4 从左侧栏拖出两个Horizontal Spacer放在左右两边，然后把两个个Horizontal Spacer 和 Label 以及刚才的Vertical Layout框全部选中，点击主页面上面的 这个按钮。右下角的layoutSpacing设置为50，完成后如图

再从左侧拖入两个Vertical Spacer放在上下两边，然后选中两个Vertical Spacer和中间这个Horizontal Spacer大框，点击上面的垂直布局按钮。最后，找块主页面的空白地方，再次点击上面的垂直布局或者水平布局（这步不可少）。完成后如下图



为使中文不显示乱码，可以在main.cpp中加入<QTextCodec>头文件，和代码QTextCodec::setCodecForTr(QTextCodec::codecForName("UTF-8"));

按前面说过的方法添加资源文件，在widget.cpp中使用如下代码给widget加入图片背景

1. QPixmap pixmap(":/img5.png");
2. QPalette palette = **this**->palette();
3. palette.setBrush(QPalette::Background,QBrush(pixmap.scaled(QApplication::desktop()->size())));
4. **this**->setPalette(palette);

第三行用到了desktop的函数要引入<QDesktopWidget>头文件。

运行



在刚才的项目中添加新文件，选择Qt设计师界面类，界面模板选择Dialog without Buttons，类名用Dialog,拖入一个Label，然后按照前面的方法进行格式设置（窗口大小，布局，字体等），完成后如下图



按照前面的代码在dialog.cpp中也加入一个图片背景。

在widget.h中加入#include "dialog.h"和一个成员Dialog \*mydialog,在widget.cpp的构造函数中加入mydialog = new Dialog(this) ;代码

然后进入widget.ui中，右键To Dialog按钮，选择转到槽，选择clicked()，确定后在跳转的函数体中加入mydialog->show();

此时我们运行程序，点击To Dialog按钮就能出现Dialog的窗口了。



继续在widget.ui中，右键Quit按钮，转到槽，选择clicked()，在函数体添加close()；

此时运行程序，点击Quit按钮就能关闭窗口了。

### 软键盘的实现

继续在刚才的项目中添加新文件，选择Qt设计师界面类，界面模板选择Dialog without Buttons，类名用Mykeyboard。

软键盘就是供鼠标点击使用的键盘，整体思想就是信号与槽实现按下某个键时（鼠标按下某个按钮），文本输入框上显示出当前按下的键。

当按下确定键时，将文本显示在Result窗口中。

mykeyboard.h中的代码

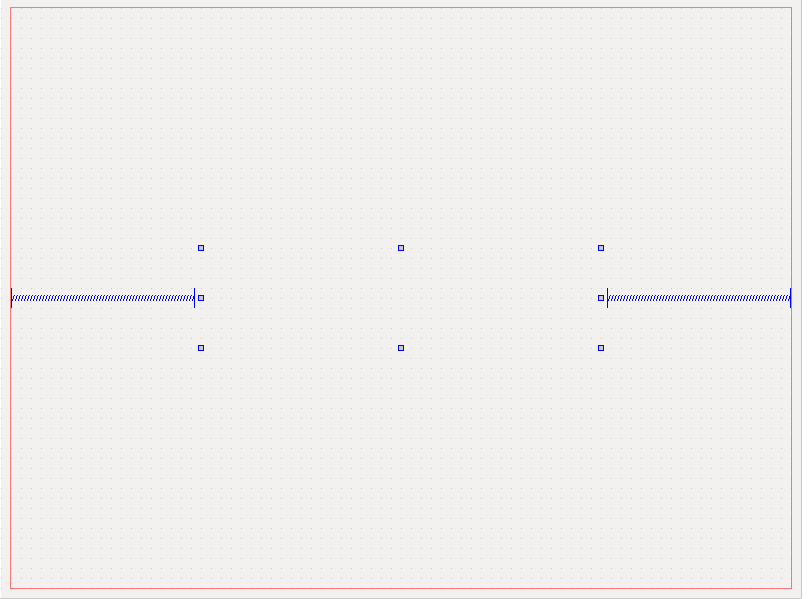
1. #ifndef MYKEYBOARD\_H
2. #define MYKEYBOARD\_H
4. #include <QDialog>
5. #include <QToolButton>
6. #include <QLineEdit>
8. **namespace** Ui {
9. **class** Mykeyboard;
10. }
12. **class** Mykeyboard : **public** QDialog
13. {
14. Q\_OBJECT
16. **public**:
17. **explicit** Mykeyboard(QWidget \*parent = 0);
18. ~Mykeyboard();
19. **friend** **class** Widget ;
21. **private**:
22. Ui::Mykeyboard \*ui;
23. QToolButton \*SoftKey[40] ;//所有的软键盘按钮
24. QLineEdit \*Line ;//文本输入框
25. QString CurString ;//按下确定后保存再CurString里
27. **private** slots:
28. **void** Init() ;
29. **void** Click1() ;
30. **void** Click2() ;
31. **void** Click3() ;
32. **void** Click4() ;
33. **void** Click5() ;
34. **void** Click6() ;
35. **void** Click7() ;
36. **void** Click8() ;
37. **void** Click9() ;
38. **void** Click0() ;
39. **void** ClickQ() ;
40. **void** ClickW() ;
41. **void** ClickE() ;
42. **void** ClickR() ;
43. **void** ClickT() ;
44. **void** ClickY() ;
45. **void** ClickU() ;
46. **void** ClickI() ;
47. **void** ClickO() ;
48. **void** ClickP() ;
49. **void** ClickA() ;
50. **void** ClickS() ;
51. **void** ClickD() ;
52. **void** ClickF() ;
53. **void** ClickG() ;
54. **void** ClickH() ;
55. **void** ClickJ() ;
56. **void** ClickK() ;
57. **void** ClickL() ;
58. **void** ClickBack() ;
59. **void** ClickClose() ;
60. **void** ClickZ() ;
61. **void** ClickX() ;
62. **void** ClickC() ;
63. **void** ClickV() ;
64. **void** ClickB() ;
65. **void** ClickN() ;
66. **void** ClickM() ;
67. **void** ClickPoint() ;
68. **void** ClickEnter() ;
69. };
71. #endif // MYKEYBOARD\_H

Mykeyboard.cpp中的代码

1. #include "mykeyboard.h"
2. #include "ui\_mykeyboard.h"
3. #include <QDesktopWidget>
5. Mykeyboard::Mykeyboard(QWidget \*parent) :
6. QDialog(parent),
7. ui(**new** Ui::Mykeyboard)
8. {
9. ui->setupUi(**this**);
10. setWindowFlags(**this**->windowFlags() | Qt::WindowStaysOnTopHint | Qt::FramelessWindowHint);
11. //WindowStaysOnTopHint设置窗口再最上端，FramelessWindowHint设置无边窗口
12. **this**->setFixedSize(QApplication::desktop()->width()/2,QApplication::desktop()->height()/2);//设置键盘的窗口大小
13. QString Toolbuttonformat = "QToolButton{background-color:rgba(242,141,102,255);\
14. color:rgba(50,25,25,255);\
15. border:2px groove rgb(242,141,102);border-radius:20px;padding:2px 2px;\
16. background-color: qconicalgradient(cx:0.5, cy:0.522909, angle:179.9, stop:0.494318 rgba(242,141,102,255), \
17. stop:0.5 rgba(242,141,102,255));}\
18. QToolButton:pressed{background-color:rgba(200,200,200,255);}";//把待会儿要用的窗口样式用字符串保存
20. Line = **new** QLineEdit(**this**) ;
21. QFont fontalnum("Microsoft YaHei",**this**->width()/30,50);
22. QFont fontfun("Microsoft YaHei",**this**->width()/35,50);
23. **for**(**int** i = 0 ; i < 40 ; ++i)
24. SoftKey[i] = **new** QToolButton(**this**) ;
25. **char** tmp[2] = {0} ;
26. **char** Alnum1[30] = "1234567890QWERTYUIOPASDFGHJKL";
27. **char** Alnum2[8] = "ZXCVBNM" ;
28. **for**(**int** i = 0 ; i < 29 ; ++i )
29. {
30. tmp[0] = Alnum1[i] ;
31. SoftKey[i]->setText(tmp) ;//设置文本
32. SoftKey[i]->setFont(fontalnum);//设置字体样式
33. }
34. **int** j = 0 ;
35. **for**(**int** i = 31; i < 38 && Alnum2[j] != '\0' ; ++i,++j )
36. {
37. tmp[0] = Alnum2[j] ;
38. SoftKey[i]->setText(tmp);
39. SoftKey[i]->setFont(fontalnum);
40. }
42. **for**(**int** i = 0 ; i < 40 ; ++i)
43. SoftKey[i]->setStyleSheet(Toolbuttonformat);//设置窗口样式
45. Line->setFixedSize(**this**->width()/1.2,**this**->height()/4);
46. Line->setStyleSheet("background-color:rgba(247,192,193,255);");
47. Line->setFont(fontalnum) ;
49. **for**(**int** i = 0 ; i < 40 ; ++i)
50. SoftKey[i]->resize(**this**->width()/10,**this**->height()/6);
52. Line->move((**this**->width()-Line->width())/2,(**this**->height()-4\*SoftKey[0]->height()-Line->height())/2);
54. **int** k = 0 ;
55. **for**( **int** i = 0 ; i < 4 ; ++i)
56. {
57. **for**( j = 0 ; j < 10 ; ++j)
58. {
59. SoftKey[k++]->move(j\*SoftKey[i]->width(),(i+2)\*SoftKey[i]->height());
60. }
61. }
63. SoftKey[29]->setIcon(QIcon(":/Backspace.png"));//给按钮对象添加图标
64. SoftKey[29]->setToolButtonStyle(Qt::ToolButtonIconOnly);//设置按钮只有图标没有文本
65. SoftKey[29]->setIconSize(QSize(SoftKey[29]->width()/2,SoftKey[29]->height()/2));
66. SoftKey[30]->setText(QObject::tr("关闭"));
67. SoftKey[30]->setFont(fontfun);
68. SoftKey[38]->setText(QObject::tr("."));
69. SoftKey[38]->setFont(fontalnum);
70. SoftKey[39]->setText(QObject::tr("确定"));
71. SoftKey[39]->setFont(fontfun);
72. Init();
73. }
75. **void** Mykeyboard::Init()
76. {
77. connect(SoftKey[0],SIGNAL(clicked(**bool**)),**this**,SLOT(Click1())) ;
78. connect(SoftKey[1],SIGNAL(clicked(**bool**)),**this**,SLOT(Click2())) ;
79. connect(SoftKey[2],SIGNAL(clicked(**bool**)),**this**,SLOT(Click3())) ;
80. connect(SoftKey[3],SIGNAL(clicked(**bool**)),**this**,SLOT(Click4())) ;
81. connect(SoftKey[4],SIGNAL(clicked(**bool**)),**this**,SLOT(Click5())) ;
82. connect(SoftKey[5],SIGNAL(clicked(**bool**)),**this**,SLOT(Click6())) ;
83. connect(SoftKey[6],SIGNAL(clicked(**bool**)),**this**,SLOT(Click7())) ;
84. connect(SoftKey[7],SIGNAL(clicked(**bool**)),**this**,SLOT(Click8())) ;
85. connect(SoftKey[8],SIGNAL(clicked(**bool**)),**this**,SLOT(Click9())) ;
86. connect(SoftKey[9],SIGNAL(clicked(**bool**)),**this**,SLOT(Click0())) ;
87. connect(SoftKey[10],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickQ())) ;
88. connect(SoftKey[11],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickW())) ;
89. connect(SoftKey[12],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickE())) ;
90. connect(SoftKey[13],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickR())) ;
91. connect(SoftKey[14],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickT())) ;
92. connect(SoftKey[15],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickY())) ;
93. connect(SoftKey[16],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickU())) ;
94. connect(SoftKey[17],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickI())) ;
95. connect(SoftKey[18],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickO())) ;
96. connect(SoftKey[19],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickP())) ;
97. connect(SoftKey[20],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickA())) ;
98. connect(SoftKey[21],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickS())) ;
99. connect(SoftKey[22],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickD())) ;
100. connect(SoftKey[23],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickF())) ;
101. connect(SoftKey[24],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickG())) ;
102. connect(SoftKey[25],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickH())) ;
103. connect(SoftKey[26],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickJ())) ;
104. connect(SoftKey[27],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickK())) ;
105. connect(SoftKey[28],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickL())) ;
106. connect(SoftKey[29],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickBack())) ;
107. connect(SoftKey[30],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickClose())) ;
108. connect(SoftKey[31],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickZ())) ;
109. connect(SoftKey[32],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickX())) ;
110. connect(SoftKey[33],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickC())) ;
111. connect(SoftKey[34],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickV())) ;
112. connect(SoftKey[35],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickB())) ;
113. connect(SoftKey[36],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickN())) ;
114. connect(SoftKey[37],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickM())) ;
115. connect(SoftKey[38],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickPoint())) ;
116. connect(SoftKey[39],SIGNAL(clicked(**bool**)),**this**,SLOT(ClickEnter())) ;
117. }
118. **void** Mykeyboard::Click1()
119. {
120. Line->setText(Line->text()+"1");
121. }
122. **void** Mykeyboard::Click2()
123. {
124. Line->setText(Line->text()+"2");
125. }
126. **void** Mykeyboard::Click3()
127. {
128. Line->setText(Line->text()+"3");
129. }
130. **void** Mykeyboard::Click4()
131. {
132. Line->setText(Line->text()+"4");
133. }
134. **void** Mykeyboard::Click5()
135. {
136. Line->setText(Line->text()+"5");
137. }
138. **void** Mykeyboard::Click6()
139. {
140. Line->setText(Line->text()+"6");
141. }
142. **void** Mykeyboard::Click7()
143. {
144. Line->setText(Line->text()+"7");
145. }
146. **void** Mykeyboard::Click8()
147. {
148. Line->setText(Line->text()+"8");
149. }
150. **void** Mykeyboard::Click9()
151. {
152. Line->setText(Line->text()+"9");
153. }
154. **void** Mykeyboard::Click0()
155. {
156. Line->setText(Line->text()+"0");
157. }
158. **void** Mykeyboard::ClickQ()
159. {
160. Line->setText(Line->text()+"Q");
161. }
162. **void** Mykeyboard::ClickW()
163. {
164. Line->setText(Line->text()+"W");
165. }
166. **void** Mykeyboard::ClickE()
167. {
168. Line->setText(Line->text()+"E");
169. }
170. **void** Mykeyboard::ClickR()
171. {
172. Line->setText(Line->text()+"R");
173. }
174. **void** Mykeyboard::ClickT()
175. {
176. Line->setText(Line->text()+"T");
177. }
178. **void** Mykeyboard::ClickY()
179. {
180. Line->setText(Line->text()+"Y");
181. }
182. **void** Mykeyboard::ClickU()
183. {
184. Line->setText(Line->text()+"U");
185. }
186. **void** Mykeyboard::ClickI()
187. {
188. Line->setText(Line->text()+"I");
189. }
190. **void** Mykeyboard::ClickO()
191. {
192. Line->setText(Line->text()+"O");
193. }
194. **void** Mykeyboard::ClickP()
195. {
196. Line->setText(Line->text()+"P");
197. }
198. **void** Mykeyboard::ClickA()
199. {
200. Line->setText(Line->text()+"A");
201. }
202. **void** Mykeyboard::ClickS()
203. {
204. Line->setText(Line->text()+"S");
205. }
206. **void** Mykeyboard::ClickD()
207. {
208. Line->setText(Line->text()+"D");
209. }
210. **void** Mykeyboard::ClickF()
211. {
212. Line->setText(Line->text()+"F");
213. }
214. **void** Mykeyboard::ClickG()
215. {
216. Line->setText(Line->text()+"G");
217. }
218. **void** Mykeyboard::ClickH()
219. {
220. Line->setText(Line->text()+"H");
221. }
222. **void** Mykeyboard::ClickJ()
223. {
224. Line->setText(Line->text()+"J");
225. }
226. **void** Mykeyboard::ClickK()
227. {
228. Line->setText(Line->text()+"K");
229. }
230. **void** Mykeyboard::ClickL()
231. {
232. Line->setText(Line->text()+"L");
233. }
234. **void** Mykeyboard::ClickBack()
235. {
236. QString s = Line->text();
237. **int** len = s.length();
238. QString s2 = s.remove(len-1,1);
239. Line->setText(s2);
240. }
241. **void** Mykeyboard::ClickClose()
242. {
243. Line->setText("");
244. **this**->close();
245. }
246. **void** Mykeyboard::ClickZ()
247. {
248. Line->setText(Line->text()+"Z");
249. }
250. **void** Mykeyboard::ClickX()
251. {
252. Line->setText(Line->text()+"X");
253. }
254. **void** Mykeyboard::ClickC()
255. {
256. Line->setText(Line->text()+"C");
257. }
258. **void** Mykeyboard::ClickV()
259. {
260. Line->setText(Line->text()+"V");
261. }
262. **void** Mykeyboard::ClickB()
263. {
264. Line->setText(Line->text()+"B");
265. }
266. **void** Mykeyboard::ClickN()
267. {
268. Line->setText(Line->text()+"N");
269. }
270. **void** Mykeyboard::ClickM()
271. {
272. Line->setText(Line->text()+"M");
273. }
274. **void** Mykeyboard::ClickPoint()
275. {
276. Line->setText(Line->text()+".");
277. }
279. **void** Mykeyboard::ClickEnter()
280. {
281. CurString = Line->text() ;
282. Line->setText("");
283. **this**->close();
284. }
286. Mykeyboard::~Mykeyboard()
287. {
288. **delete** ui;
289. **for**(**int** i = 0 ; i < 40 ; ++i)
290. **delete** SoftKey[i] ;
292. **delete** Line ;
293. }

通样继续在刚才的项目中添加新文件，选择Qt设计师界面类，界面模板选择Dialog without Buttons，类名用Result。

在result.h中加上友元friend class Widget。

在result.ui中拖入一个Label，去掉里面的text内容其他的设置参考前面的ui界面，如下图

在widget.h中添加

#include "mykeyboard.h"

#include "result.h"

成员添加

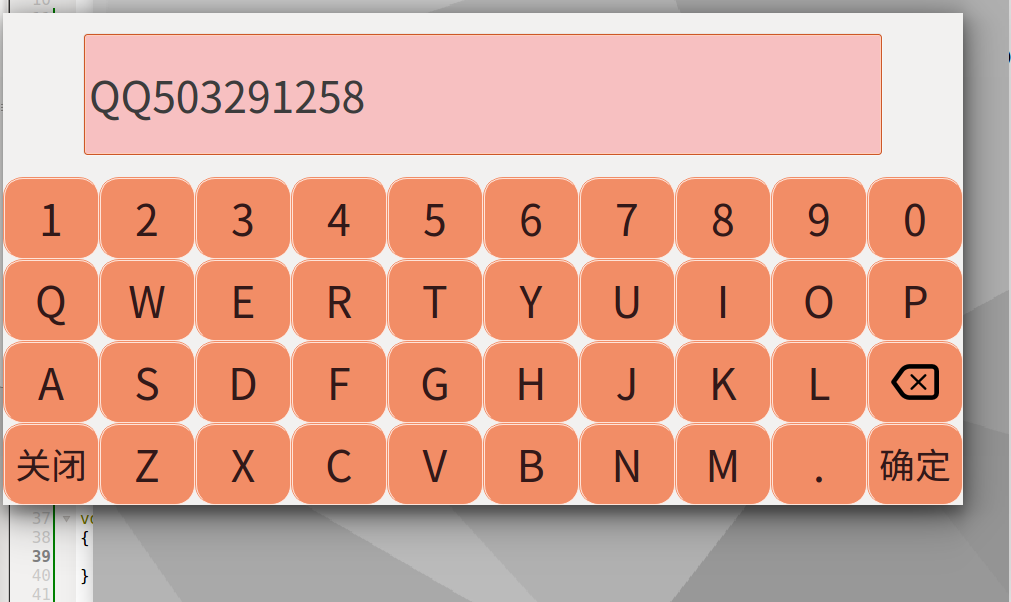
Mykeyboard \*mkb ;

Result \*curresult ;

进入widget.ui界面，右键Mykeyboard，转到槽，选择clicked()，在跳转的函数体中加入mkb->exec();右键Result，转到槽，选择clicked()，在跳转的函数体中加入 curresult->ui->label->setText(tr("The Result:%1").arg(mkb->CurString));

curresult->show();

至此，程序完成。运行程序，点击Mykeyboard，点击软键盘输入......



点击确定，回到主页后点击Result按钮，发现刚才的输入显示在窗口中

