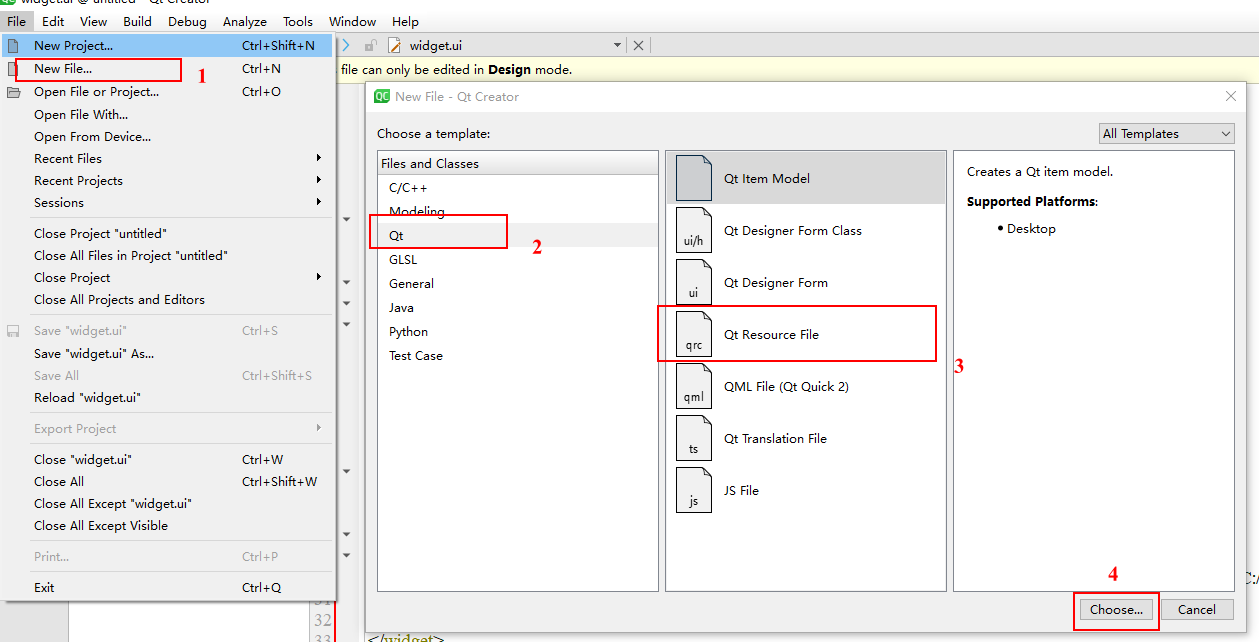
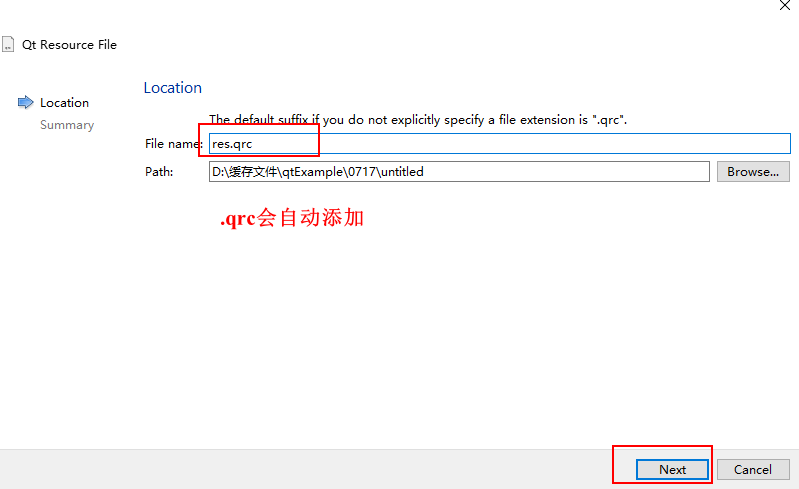
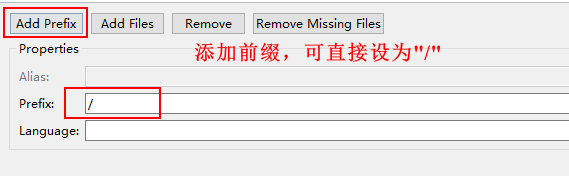
# 1. QmainWindow, Qwidget和Qdialog的区别

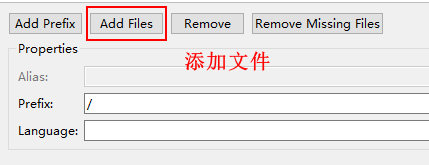
QMainWindow 是主窗口类，主窗口类具有菜单栏、工具栏和状态栏。  
QWidget 是所有界面组件类的基类， QWidget 可以作为独立的窗口，就是一个空白的窗口。  
QDialog 是对话框类，窗口具有对话框的显示效果，例如没有最大化按钮。

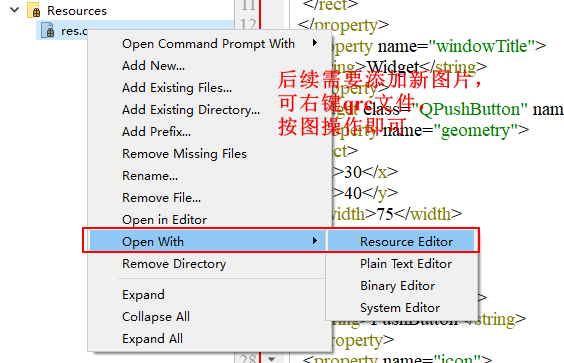
# 2.创建资源文件

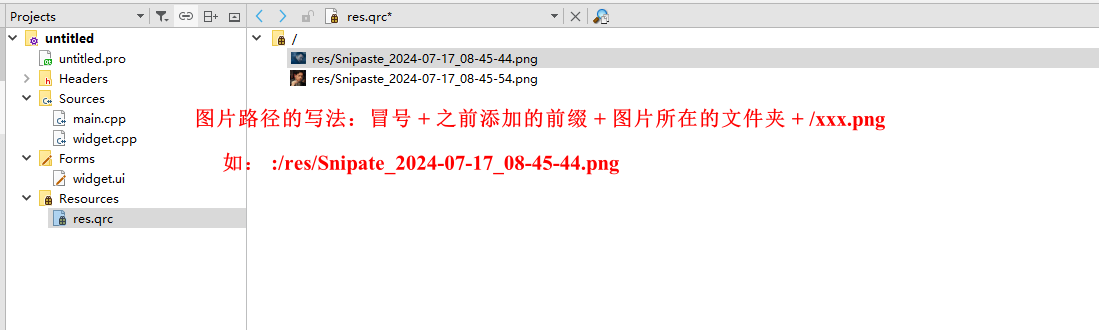














注意：

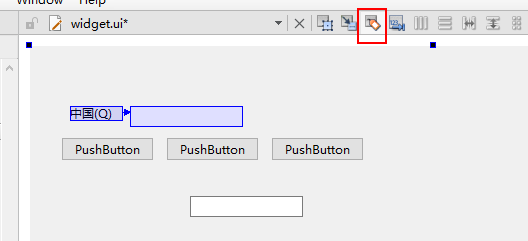
a.前缀若是不加”/”，程序会默认加上；

b.图片路径写法中，若添加前缀不是单一个”/”,在前缀与图片所在文件夹中应补充一个”/”；

c.之后项目打包发布的时候就不需要res文件夹了，所有的图片信息都以二进制形式存在xxx.qrc文件里了。

# **3.伙伴关系**

点击 Qt Designer 工具栏上的 Edit Buddies 按钮可以进入伙伴关系编辑状态，伙伴关系是指界面上一个标签和一个具有输入焦点的组件相关联。在伙伴关系编辑状态下，选中一个标签，按住鼠标左键，然后将其拖向一个编辑框（ QLineEdit 类），就可建立标签和编辑框的伙伴关系。  
利用伙伴关系，可以在程序运行时通过快捷键将输入焦点切换到某个组件上。例如，将标签的test属性设置为“中国(&Q)”，其中符号”&”用来指定快捷字符，界面上并不显示”&”。这里指定快捷字符为 Q，那么在程序运行时，用户按下 Alt+Q 快捷键，输入焦点就会快速切换到“中国”标签关联的编辑框内。



# 4.信号与槽机制

QObject::connect(sender, SIGNAL(signal()), receiver, SLOT(slot()));

其中， sender 是发射信号的对象的名称； signal()是信号，信号可以看作特殊的函数，需要带有括号，有参数时还需要指明各参数类型； receiver 是接收信号的对象的名称； slot()是槽函数，需要带有括号，有参数时还需要指明各参数类型。

SIGNAL 和 SLOT 是 Qt 的宏，分别用于指明信号和槽函数，并将它们的参数转换为相应的字符串。关于信号与槽的使用，有以下一些规则需要注意。

1.一个信号可以连接多个槽函数，例如：

connect(spinNum, SIGNAL(valueChanged(int)), this, SLOT(addFun(int)));

connect(spinNum, SIGNAL(valueChanged(int)), this, SLOT(updateStatus(int)));

当一个信号与多个槽函数关联时，槽函数按照建立连接时的顺序依次运行。

当信号和槽函数带有参数时，在函数connect()里要指明各参数的类型，但不用指明参数名称。

2.多个信号可以连接同一个槽函数。

connect(ui->radioBlack,SIGNAL(clicked()), this, SLOT(do\_setFontColor()));

connect(ui->radioRed, SIGNAL(clicked()), this, SLOT(do\_setFontColor()));

connect(ui->radioBlue, SIGNAL(clicked()), this, SLOT(do\_setFontColor()));

3.一个信号可以连接另一个信号，例如：

connect(spinNum, SIGNAL(valueChanged(int)), this, SIGNAL(refreshInfo(int)));

这样，当发射一个信号时，也会发射另一个信号，以实现某些特殊的功能。

4.严格的情况下，信号与槽的参数个数和类型需要一致，至少信号的参数不能少于槽的参数。如果参数不匹配，会出现编译错误或运行错误。

5.在使用信号与槽的类中，必须在类的定义中插入宏 Q\_OBJECT。

6.用窗口对象调用时可少一个参数，如下列两者等效：

this->connect(spinNum, SIGNAL(valueChanged(int)), SLOT(updateStatus(int)));

connect(spinNum, SIGNAL(valueChanged(int)), this, SLOT(updateStatus(int)));

this 表示窗口对象， updateStatus()是窗口类里定义的一个槽函数。

7.当一个信号被发射时，与其关联的槽函数通常被立即运行，就像正常调用函数一样。只有当信号关联的所有槽函数运行完毕后，才运行发射信号处后面的代码。

函数 connect()有多种参数形式，有一种常用的形式是不使用 SIGNAL 和 SLOT 宏，而是使用函数指针。

connect(lineEdit, &QLineEdit::textChanged, this, &Widget::do\_textChanged);

对于具有默认参数的信号，即信号名称是唯一的，不存在参数不同的其他同名的信号，可以使用这种函数指针形式进行关联

QLineEdit 有一个信号 textChanged(QString)，不存在参数不同的其他 textChanged()信号，自定义窗口类 Widget 里有一个槽函数 do\_textChanged(QString)。这样就可以用上面的语句将此信号与槽关联起来，无须出现函数参数。当信号的参数比较多时，这种写法简单一些。

如果在窗口类 Widget 里设计了如下的两个自定义槽函数：

void do\_click(bool checked);

void do\_click( );

那么使用如下的代码进行信号与槽的连接时，编译会出现错误。

connect(ui->checkBox, &QCheckBox::clicked, this, &Widget::do\_click);

这是因为 QCheckBox 的 clicked()信号是 overload 型信号， do\_click()是 overload 型槽函数，信号与槽函数无法匹配。这时需要使用模板函数 qOverload()来明确参数类型，如果写成下面的语句再编译时就没有问题。

connect(ui->checkBox, &QCheckBox::clicked, this, qOverload<bool>(&Widget::do\_click));

connect(ui->checkBox, &QCheckBox::clicked, this, qOverload<>(&Widget::do\_click));

第一行语句是将信号 clicked(bool)与槽函数 do\_click(bool)连接，第二行语句是将信号 clicked()与槽函数 do\_click()连接。模板函数 qOverload()的作用是明确 overload 型函数的参数类型。

因此，对于 overload 型信号，只要槽函数不是 overload 型， 就可以使用传递函数指针的 connect()来进行信号与槽的关联， Qt 会根据槽函数的参数自动确定使用哪个信号。我们在设计槽函数的时候一般也不会设计成 overload 型的。

Connect函数的第五个参数

• Qt::AutoConnection（默认值）：如果信号的接收者与发射者在同一个线程中，就使用Qt::DirectConnection 方式，否则使用 Qt::QueuedConnection 方式，在信号发射时自动确定

关联方式。

• Qt::DirectConnection：信号被发射时槽函数立即运行，槽函数与信号在同一个线程中。

• Qt::QueuedConnection：在事件循环回到接收者线程后运行槽函数，槽函数与信号在不同

的线程中。

• Qt::BlockingQueuedConnection：与 Qt::QueuedConnection 相似，区别是信号线程会阻塞，

直到槽函数运行完毕。当信号与槽函数在同一个线程中时绝对不能使用这种方式，否则会

造成死锁。

信号函数必须是无返回值的函数，但是可以有输入参数。信号函数无须实现，而只需在某些条件下被发射。例如，函数 incAge()中会发射信号 ageChanged()，其代码如下：

signals: void ageChanged( int value);

void QPerson::incAge()

{

m\_age++;

emit ageChanged(m\_age); //发射信号

}

void 类名::on\_控件名\_信号(参数类型) ｛...｝

直接命名槽函数名，在Qt中，按以上规则给槽函数命名可直接将控件的信号关联到槽函数：