第1章-认识Qt

author：岳石磊

1. Qt 简介

Qt（官方发音 [kju:t]，音同 cute）是一个**跨平台**的 C++ 开发库，主要用来开发**图形用户界面**（Graphical User Interface，GUI）程序，当然也可以开发不带界面的命令行（Command User Interface，CUI）程序。同时也是图像用户界面应用程序框架，它为应用程序开发者提供建立艺术级图形界面所需的功能。它是完全面向对象的，很容易扩展，并且允许真正的组件编程。
Qt 支持的操作系统有很多，例如通用操作系统 Windows、Linux、Unix，智能手机系统 Android、iOS、WinPhone， 嵌入式系统 QNX、VxWorks 等等。

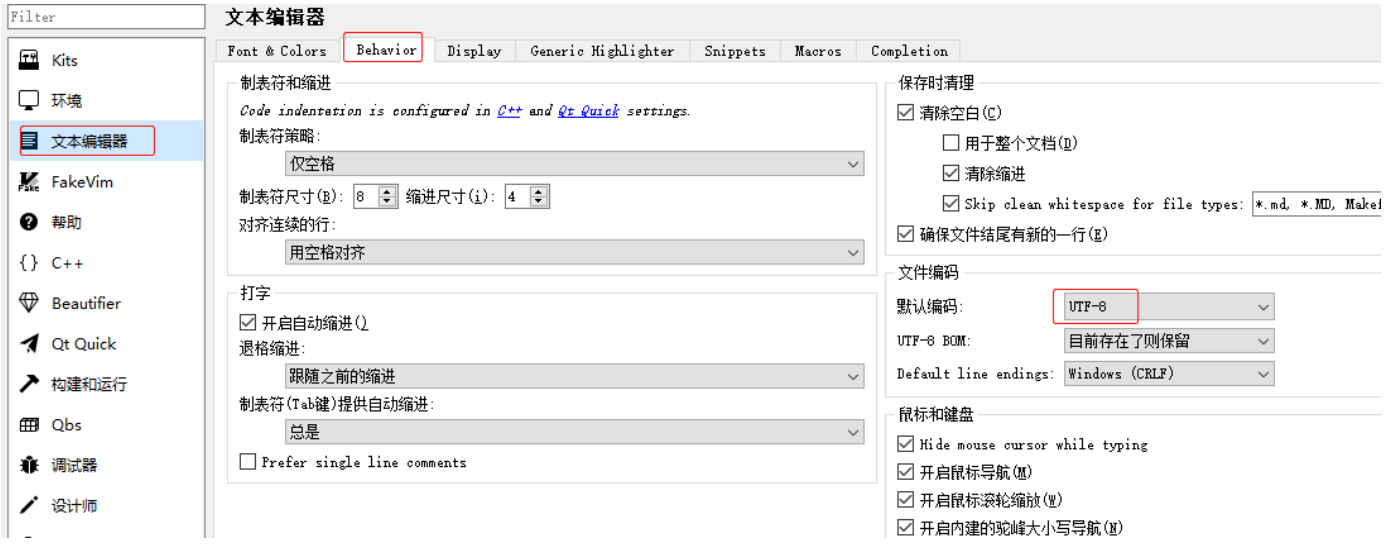
Qt 虽然经常被当做一个 GUI 库，用来开发图形界面应用程序，但这并不是 Qt 的全部；Qt 除了可以绘制漂亮的界面（包括控件、布局、交互），还包含很多其它功能，比如多线程、访问数据库、图像处理、音频视频处理、网络通信、文件操作等，这些 Qt 都已经内置了。

用 Qt 来开发 Windows 桌面程序有以下优点：
简单易学：Qt 封装的很好，几行代码就可以开发出一个简单的客户端，不需要了解 Windows API。
跨平台：如果你的程序需要运行在多个平台下，同时又希望降低开发成本，Qt 几乎是必备的。
漂亮的界面：Qt 很容易做出漂亮的界面和炫酷的动画，而 MFC、WTL、wxWidgets 比较麻烦。
独立安装：Qt 程序最终会编译为本地代码，不需要其他库的支撑，而 Java 要安装虚拟机，C# 要安装 .NET Framework。

2.Qt 建立工程

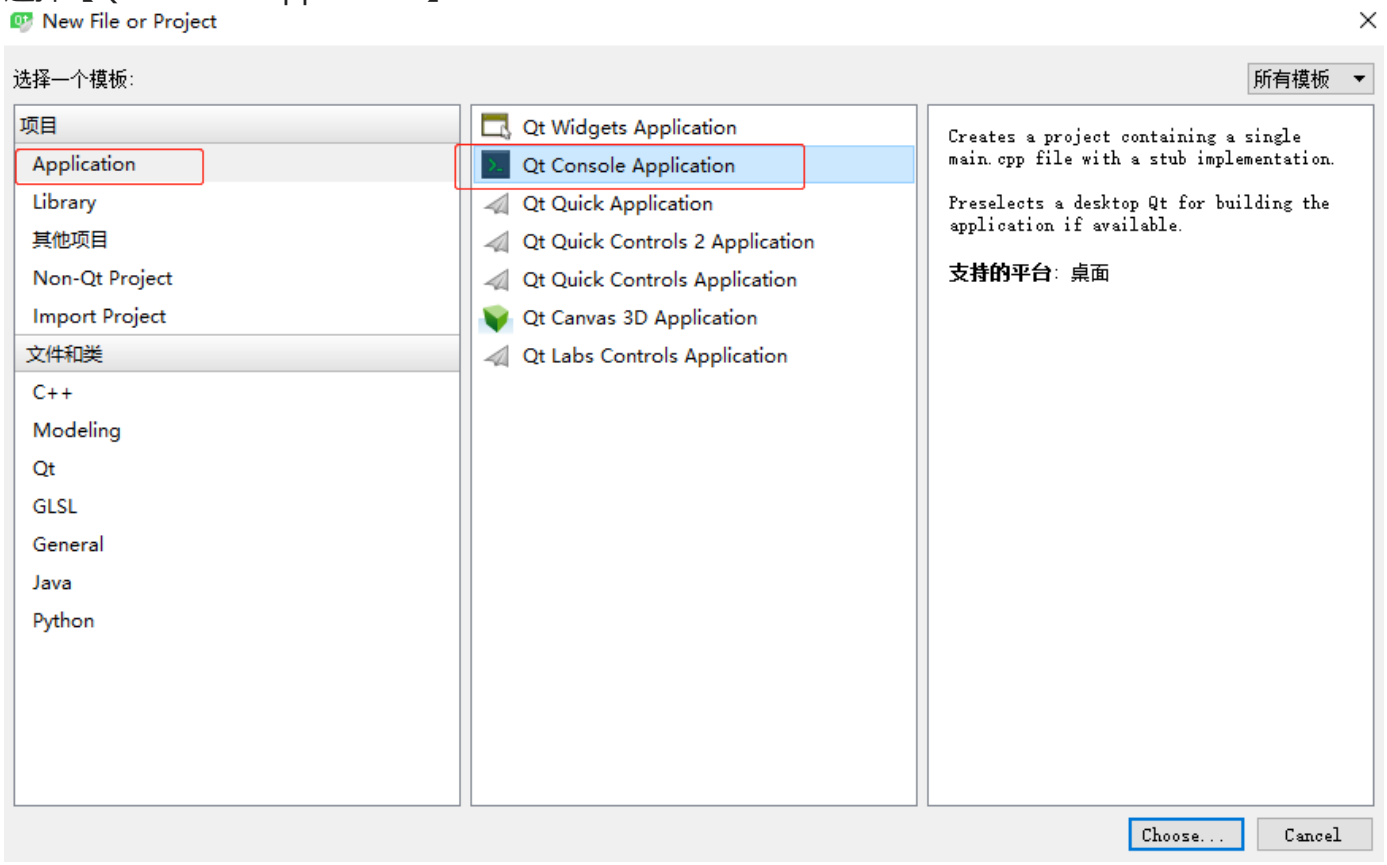
准备

为了方便开发和避免中文乱码问题，将文本编辑器的默认编码改成【UTF-8】

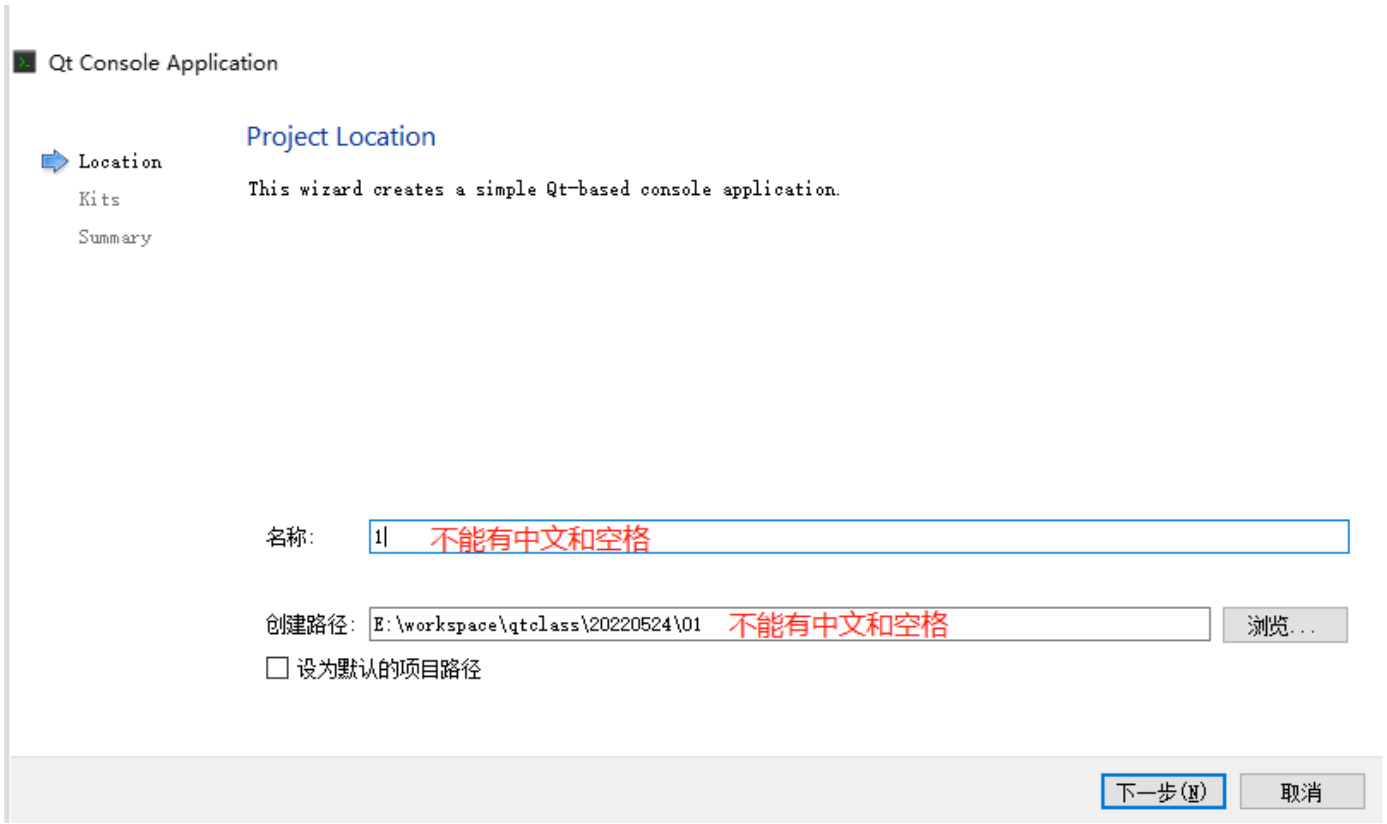


2.1 建立控制台程序

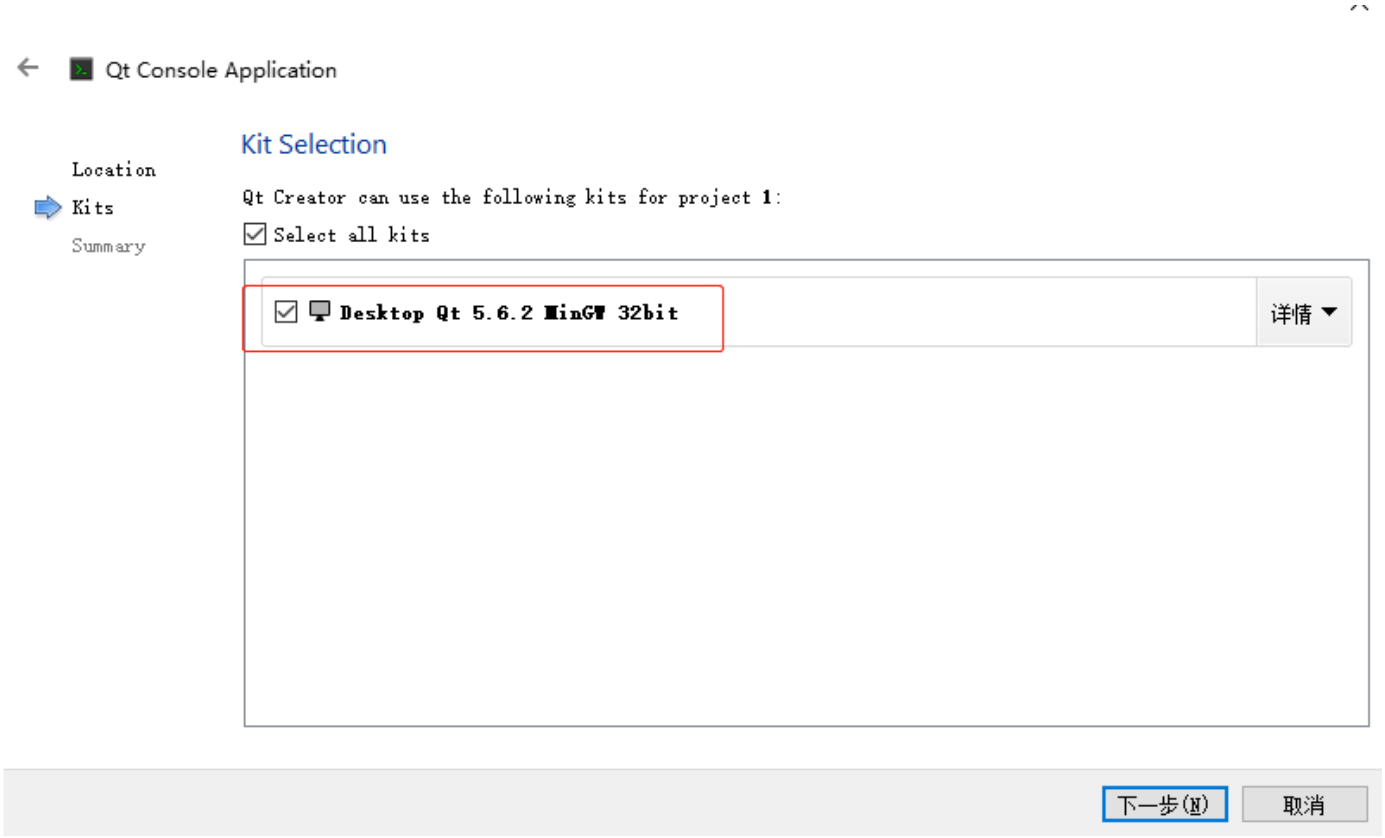
选择【Qt Console Application】：

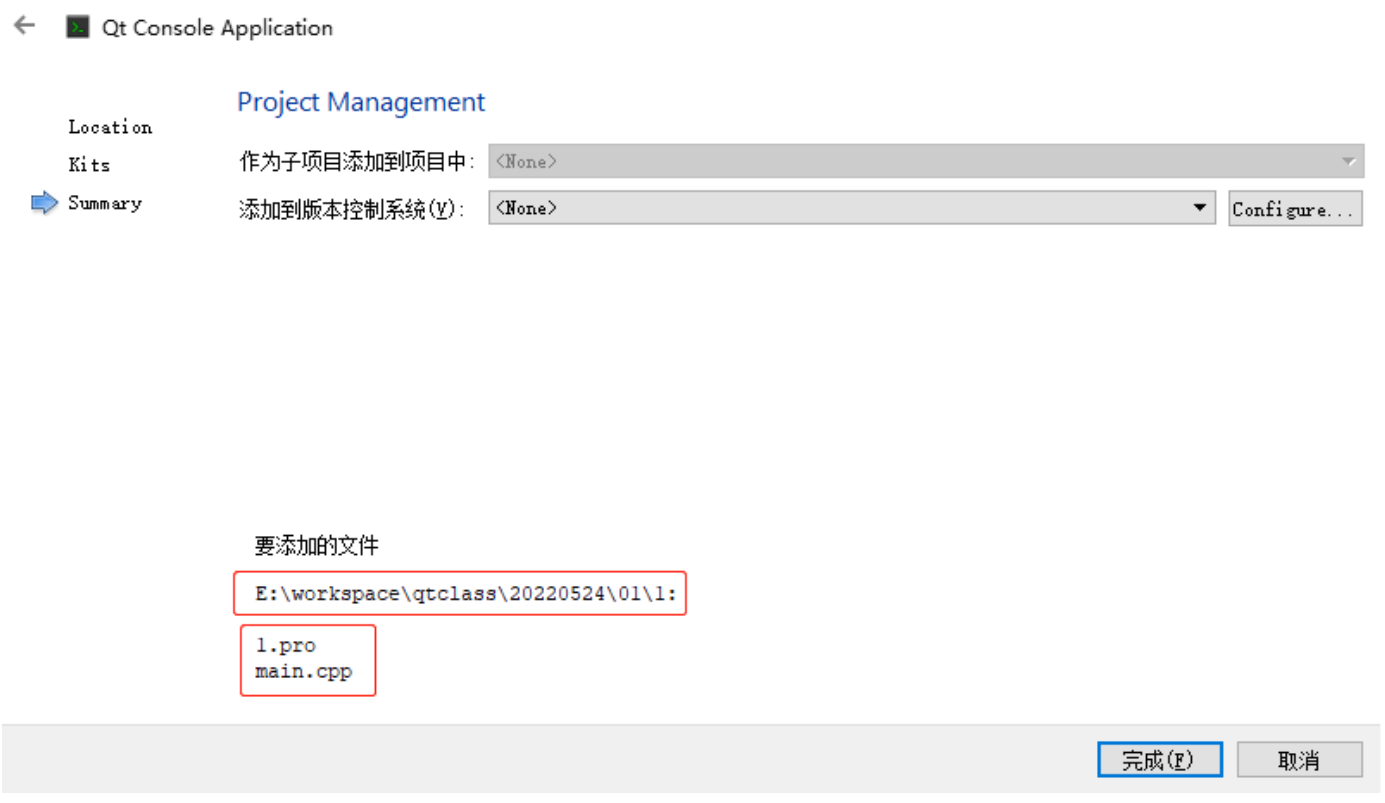


填写名称、选择路径（**名称和路径不能带中文、空格**）



选择一个合适的**编译套件**：



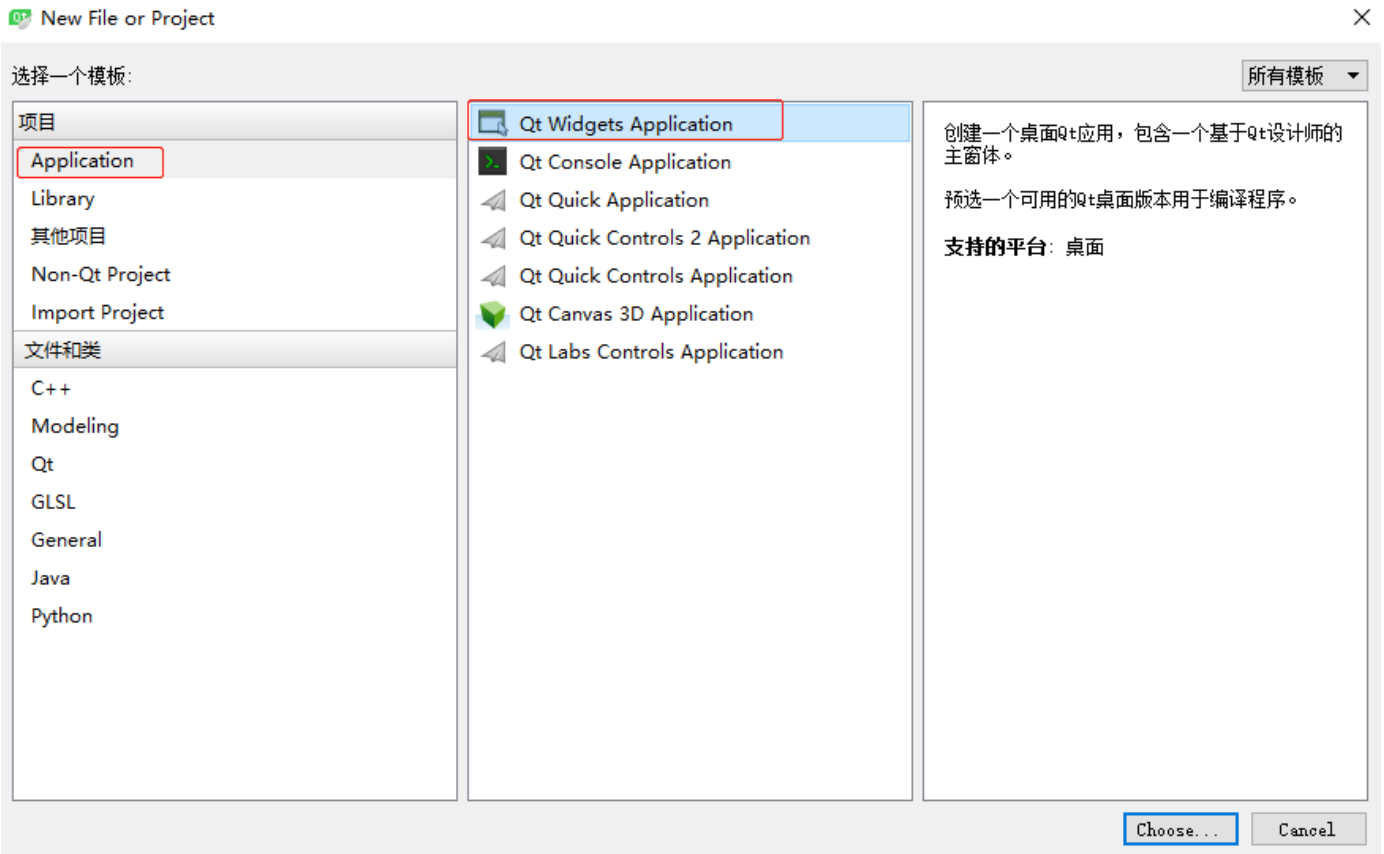


跟vs里默认的main 函数略有差别，多了命令行参数。

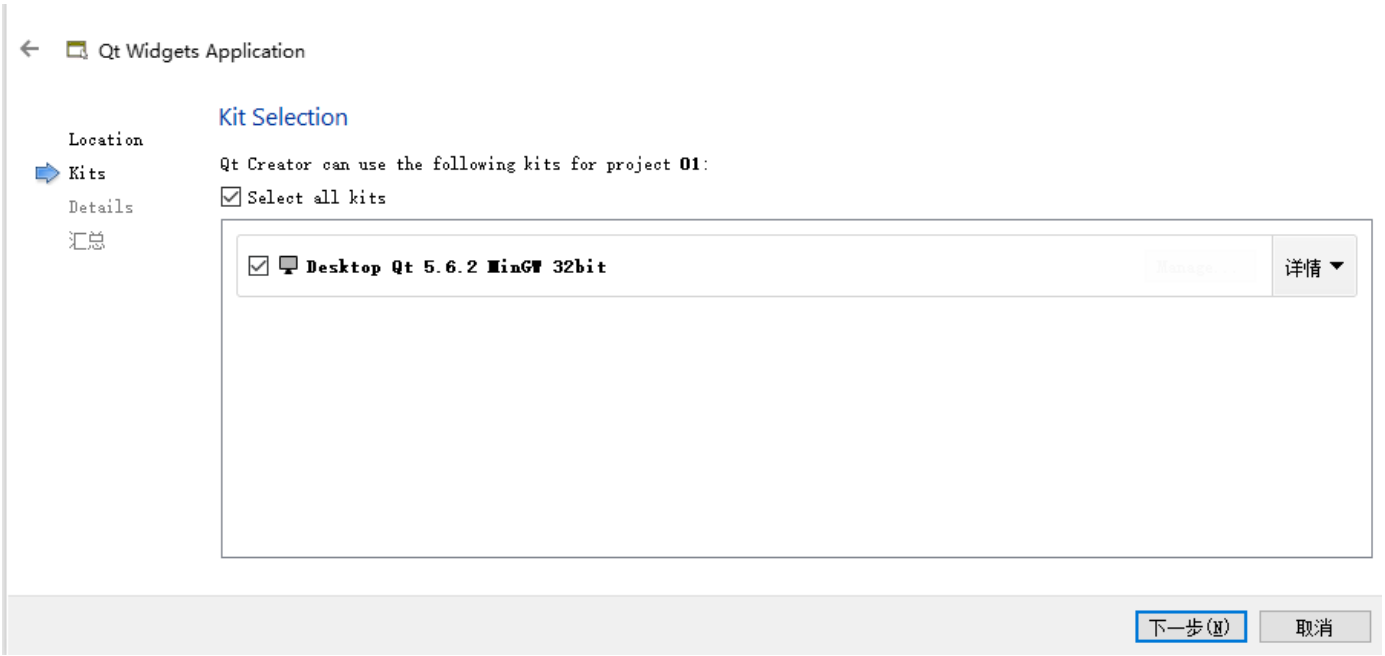
```
#include <QCoreApplication>
#include<iostream>
using namespace std;

```

```
5  int main(int argc, char *argv[])
6  {
7      QCoreApplication a(argc, argv);
8
9      cout<<"argc:"<<argc<<endl;    //指明了命令行参数的个数
10
11     for(int i=0;i<argc;i++){  //循环输出命令行参数
12         cout<<argv[i]<<endl;
13     }
14
15    return a.exec();  //阻塞
16 }
```

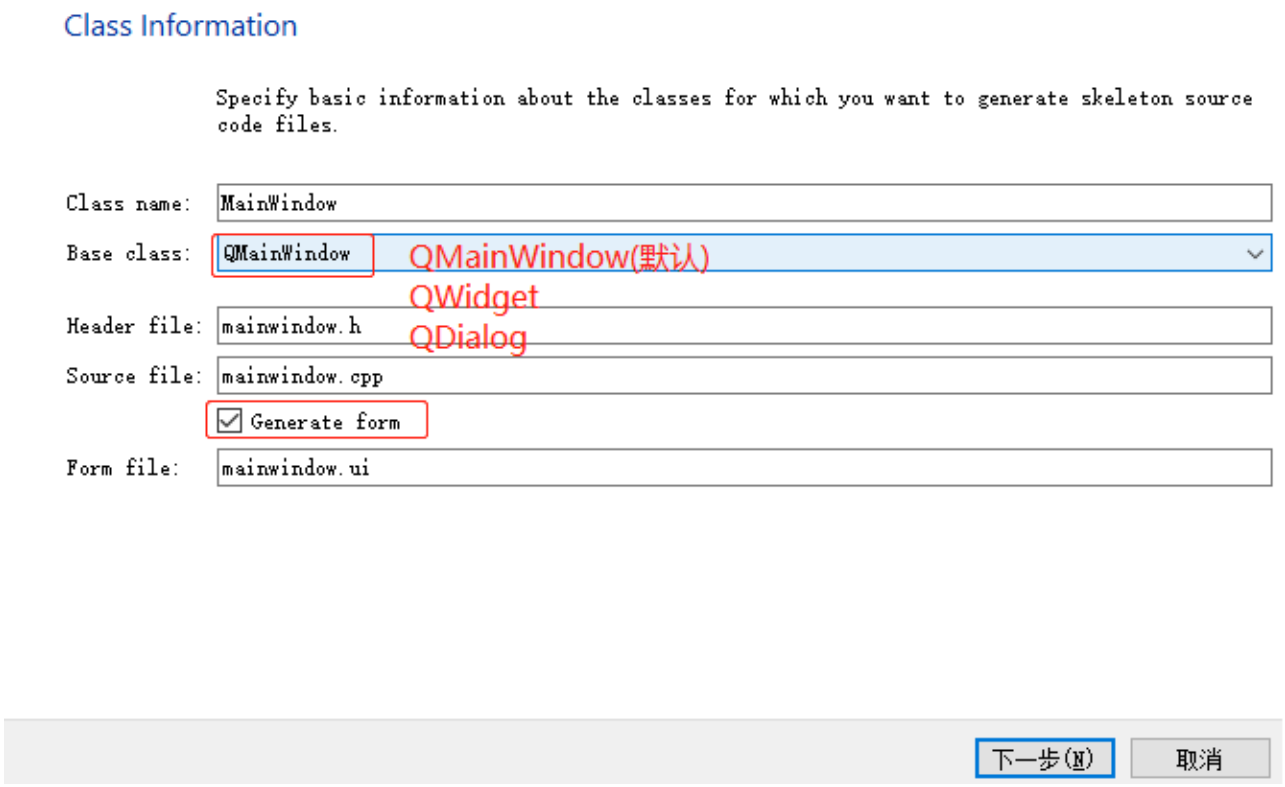
2.2 建立桌面窗口程序



填写名称、选择路径（**名称和路径不能带中文、空格**）

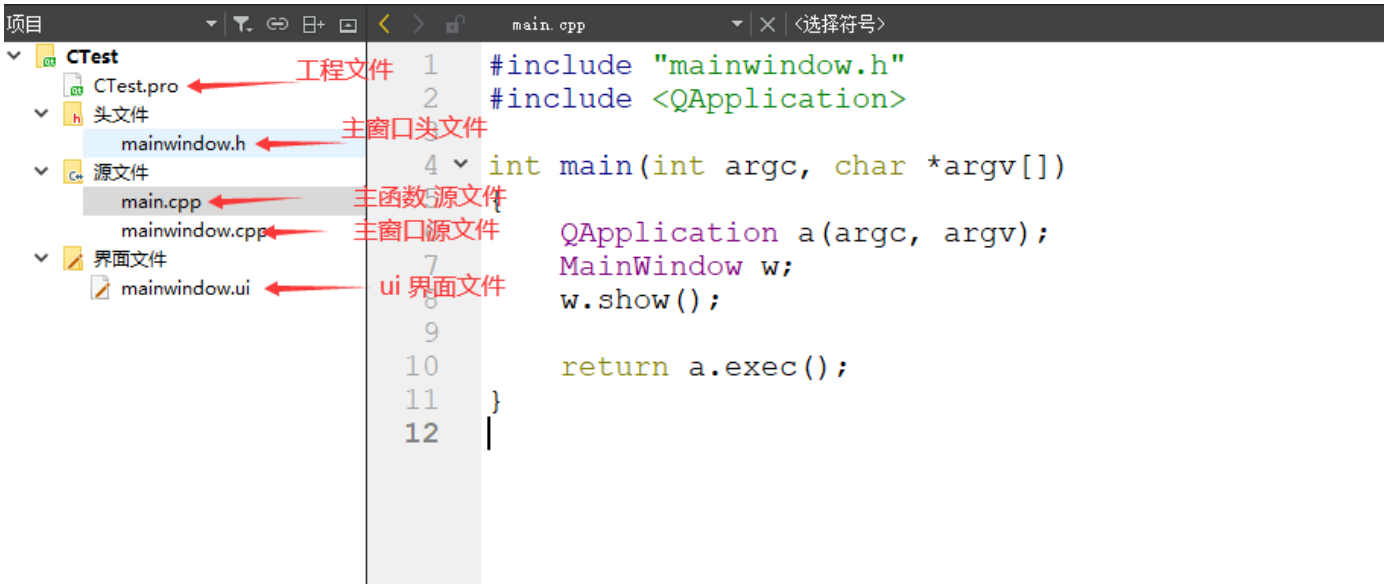


建立一个QMainWindow 窗口。

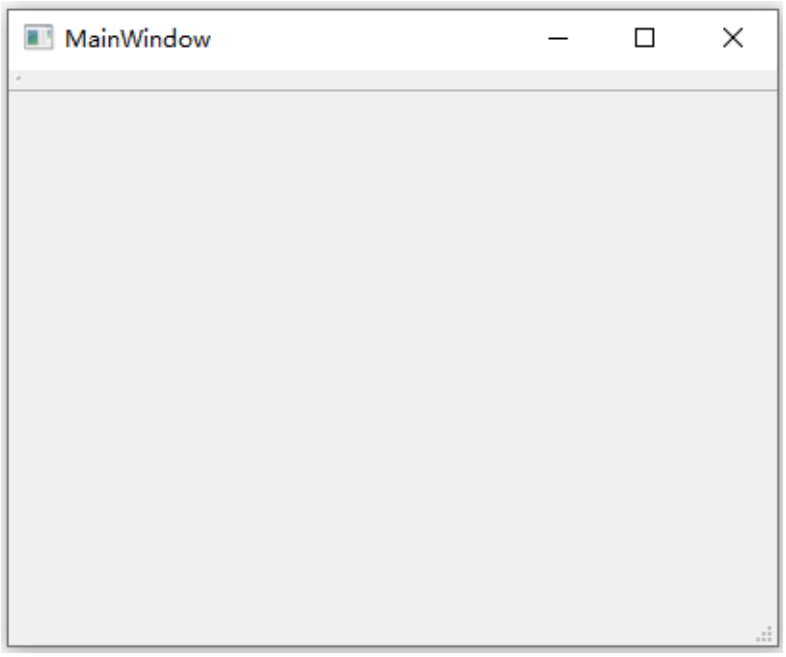

Class Information
Specify basic information about the classes for which you want to generate skeleton source code files.
Class name:
MainWindow
Base class:
QMainWindow
QMainWindow(默认)
QWidget
QDialog
Header file:
mainwindow.h
Source file:
mainwindow.cpp
Generate form
Form file:
mainwindow.ui

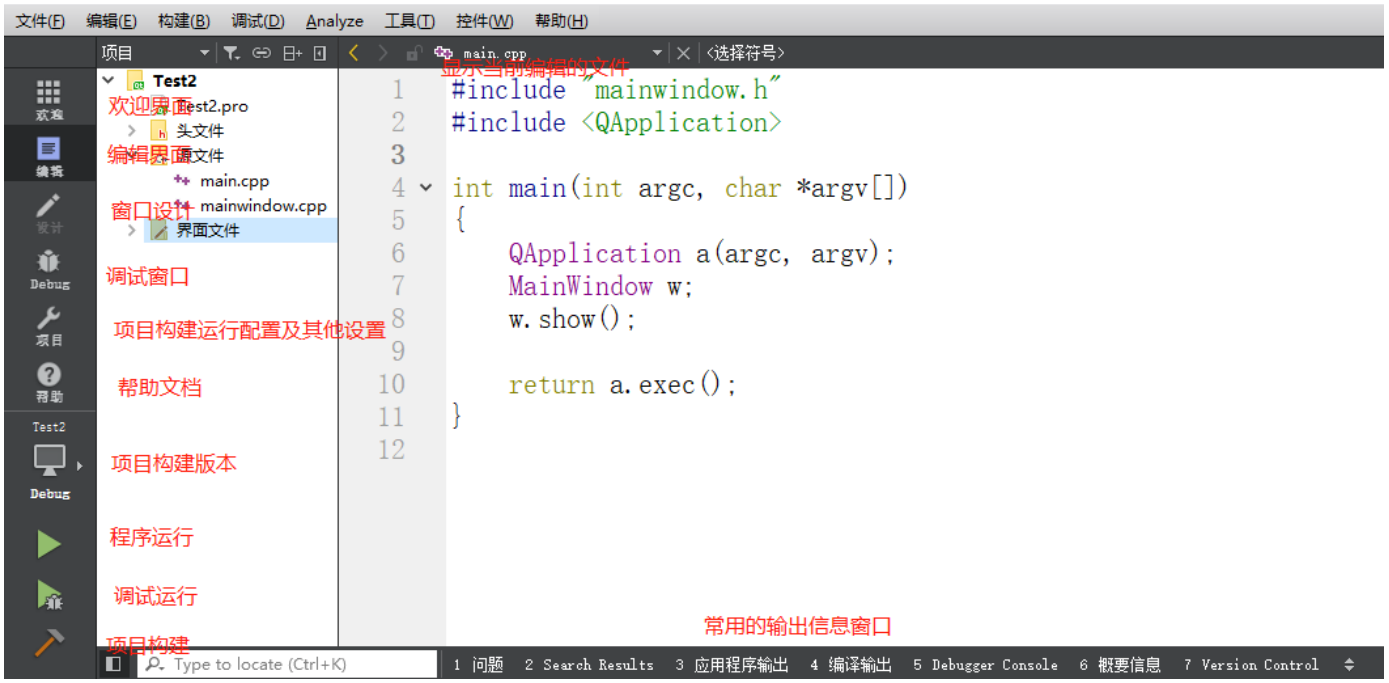
下一步(N)
取消


Qt Widgets Application
项目管理
Location
Kits
Details
汇总
作为子项目添加到项目中:
<None>
添加到版本控制系统(V):
<None>
Configure...
要添加的文件
C:\Users\1\Documents\untitled:
main.cpp
mainwindow.cpp
mainwindow.h
mainwindow.ui
untitled.pro
完成(F)
取消



运行的窗口如下：除了主窗体外(标题、边框、最大化、最小化、关闭)，包含了 菜单、工具栏、状态栏。





项目构建版本：

1. debug：调试版本，程序中带着符号信息，便于我们调试，该模式下会严格按照代码写的进行编译，运行速度相对较慢，程序比较大。
2. release：发布版本，不会包含调试信息，且会对代码进行优化，运行速度相对较快，程序比较小。
3. profile：是在debug 和 release 两种之中取一个平衡，兼顾性能和调试。

3.工程文件介绍

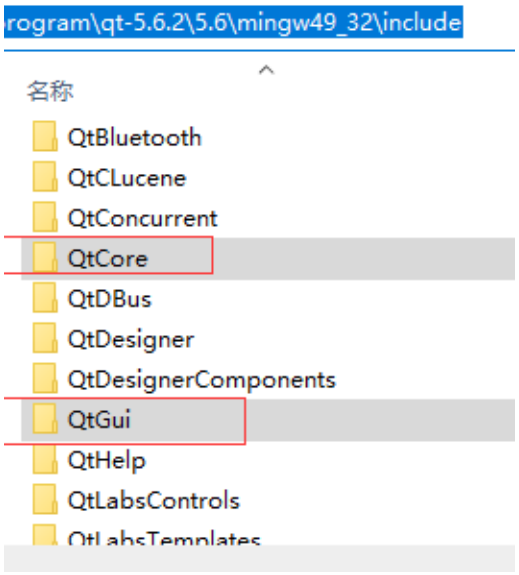
3.1 工程文件.pro

添加注释用 # ，而不是用 //

```
#-------------------------------------------------
#
# Project created by QtCreator 2021-09-05T15:02:01
#
#-------------------------------------------------

QT       += core gui                                    #包含的核心模块 分别是 core gui,是设计GUI设计必不可少的模块，如果是控制台程序，则不需要

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets  #条件执行语句，如果QT主版本大于4时，需要加入 widgets 模块

CONFIG += c++11

TARGET = TestMainWindow                                 #表示生成可执行文件的名称，TestWidget.exe 需要结合 TEMPLATE 配置
```

```
14  TEMPLATE = app                                  #表示项目使用的模板， ap
    p 指的是一般应用程序，还可指定 lib 、subdirs 等
15
16
17  SOURCES += main.cpp\                            # 项目中包含的所有源文
    件，多行用 \ 连接
18          mainwindow.cpp
19
20  HEADERS  += mainwindow.h                        # 项目中包含的头文件
21
22  FORMS    += mainwindow.ui                       # 项目中包含的ui文件
23
24  # Default rules for deployment.
25  qnx: target.path = /tmp/$${TARGET}/bin
26  else: unix:!android: target.path = /opt/$${TARGET}/bin
27  !isEmpty(target.path): INSTALLS += target
```

默认包含的核心模块对应磁盘上的 QtCore QtGui



FORMS 包含的ui界面文件，一般不需要我们手动修改，在UI编辑器中修改，构建后文件内容会自动更新。

3.2 mainwindow.h

主窗口的头文件，定义了 `MainWindow` 类，继承 `QMainWindow`，包含 窗口界面的指针 `Ui::MainWindow *ui`，`Ui::MainWindow` 的定义在`ui_mainwindow.h`中。

3.3 mainwindow.cpp

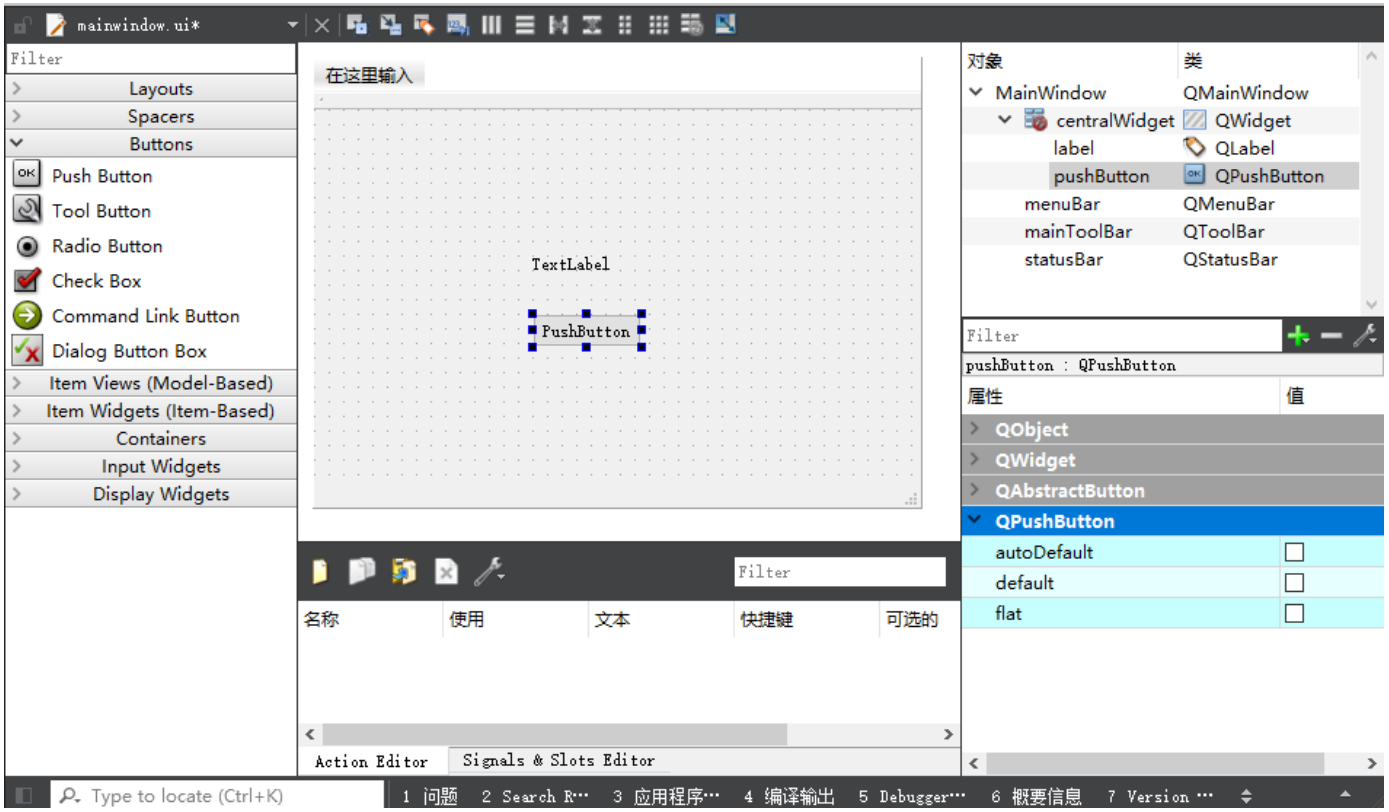
`mainwindow.h` 头文件对应的源文件，包含了 `MainWindow`类成员函数的实现。

3.4 main.cpp

包含了main主函数，定义了主窗口`MainWindow`类对象，并显示出来。

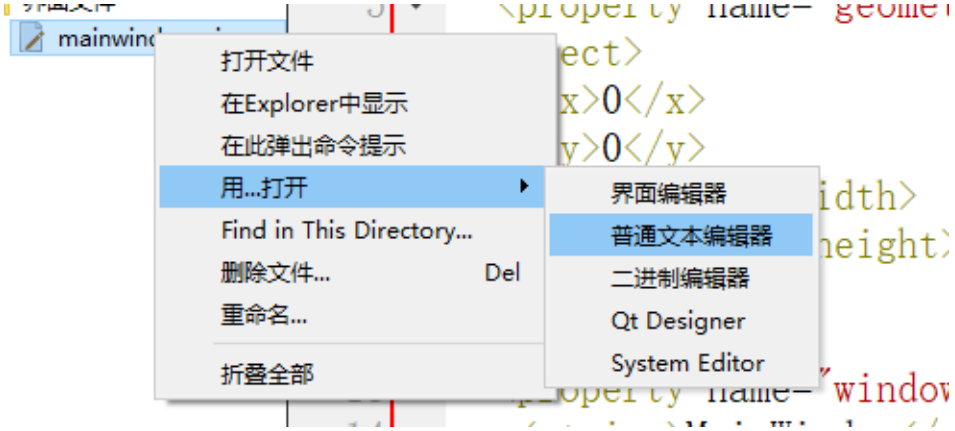
3.5 mainwindow.ui

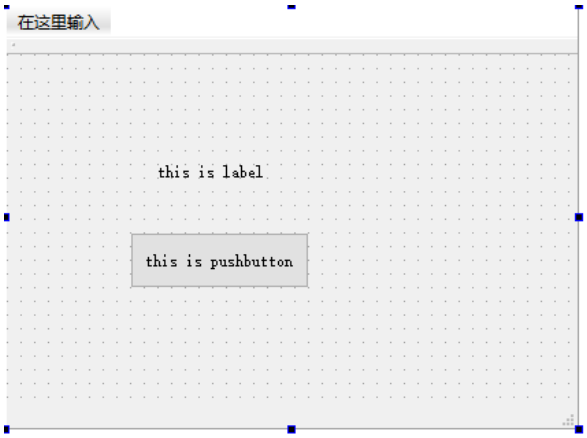
可视化设计的窗体的定义文件，这是一个GUI 界面文件，是一个 XML 文件，定义了窗口上的所有组件的属性设置、布局，及其信号与槽函数的关联等。双击文件会进入到一个**图形界面编辑器**，默认状态下是一个干净、简单的窗口（无菜单、工具栏、状态栏）。



1. **组件面板**：窗口左侧是界面设计组件面板，分为多个组，如`Layouts`、`Buttons`、`Display Widgets`等，界面设计的常见组件都可以在组件面板里找到。
2. **待设计的窗体**。如果要将某个组件放置到窗体上时，从组件面板上拖放一个组件到窗体上即可。例如，先放一个 `Label` 和一个 `Push Button` 到窗体上。
3. Signals **和** Slots **编辑器与** Action **编辑器**：位于待设计窗体下方的两个编辑器。Signals 和Slots 编辑器用于可视化地进行信号与槽的关联，Action 编辑器用于可视化设计 Action。
4. **布局和界面设计工具栏**：窗口上方的一个工具栏，工具栏上的按钮主要实现布局和界面设计。
5. **对象浏览器（**Object Inspector）：窗口右上方是 Object Inspector，用树状视图显示窗体上各组件之间的布局包含关系，视图有两列，组件的对象名称（ObjectName）和类名称（ClassName）。
6. **属性编辑器（**Property Editor）：窗口右下方是经常使用的属性编辑器。属性编辑器显示某个选中的组件或窗体的各种属性和对应值，可以在属性编辑器里修改这些属性的值。属性又分为多个组，表示了组件类的继承关系。

ui文件可以用文本编辑器打开，是一个 XML（Extensible Markup Language：可扩展标记语言） 格式文件。






```
<widget class="QWidget" name="centralWidget">
 <widget class="QLabel" name="label">
  <property name="geometry">
   <rect>
    <x>110</x>
    <y>70</y>
    <width>131</width>
    <height>31</height>
   </rect>
  </property>
  <property name="text">
   <string>this is label</string>
  </property>
 </widget>
 <widget class="QPushButton" name="pushButton">
  <property name="geometry">
   <rect>
    <x>90</x>
    <y>130</y>
    <width>131</width>
    <height>41</height>
   </rect>
  </property>
  <property name="text">
   <string>this is pushbutton</string>
  </property>
 </widget>
</widget>
```

把窗体及部件按照一定的规则集合在一起，在程序构建时编译器会将这个文件生成对应的ui_mainwindow.h 文件。这个文件在 mainwindow.cpp中用到了，但是并不包含在工程的目录中。之前提到的 Ui::MainWindow 类就在这个文件中定义的。所以 对于界面的改动，在 ui_mainwindow.h 中修改是无意义的。