



第1章 Qt概述

1.1 什么是Qt

1.2 Qt 5的安装

1.3 Qt 5开发步骤及实例





1.1 什么是Qt

Qt是一个跨平台的C++图形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用户界面所需的所有功能。

Qt是诺基亚公司的一个产品。1996年，Qt进入商业领域，已成为全世界范围内数千种成功的应用程序的基础。它也是目前流行的Linux桌面环境KDE 的基础，KDE是Linux发行版的主要一个标准组件。



1.1 什么是Qt

Qt支持的平台有：

MS/Windows—95、98、NT 4.0、ME、2000、XP和Vista；

UNIX/X11—Linux、Sun Solaris、HP-UX、Compaq Tru64 UNIX、IBM AIX、SGI IRIX和其他很多X11平台；

Macintosh—Mac OS X；

Embedded—有帧缓冲（framebuffer）支持的Linux平台、Windows CE；

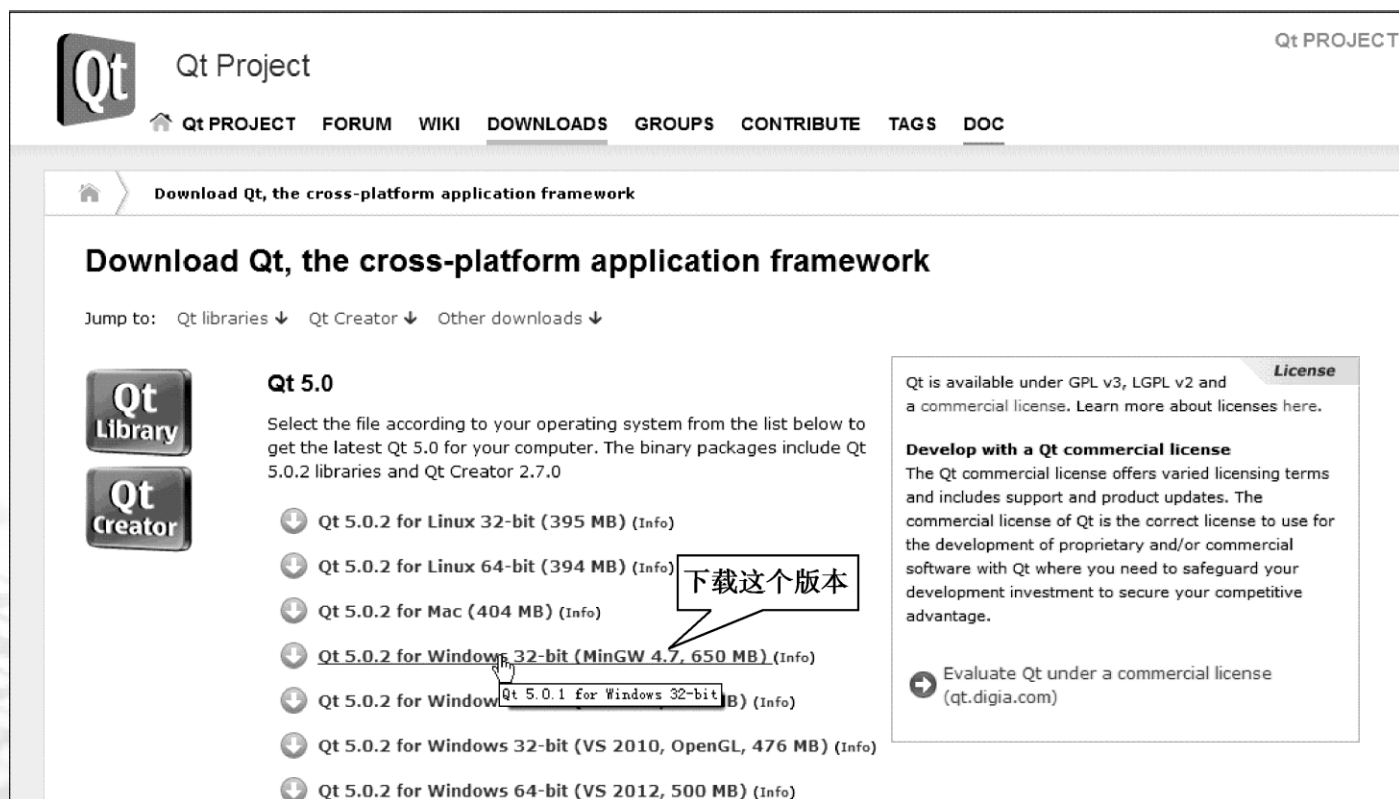
Symbian/S60—目前已经可以提供技术预览版本。



1.2 Qt 5的安装

1.2.1 下载Qt 5 Creator

下载地址：<http://qt-project.org/downloads>，下载页面如图1.1所示。





1.2.2 运行Qt 5 Creator

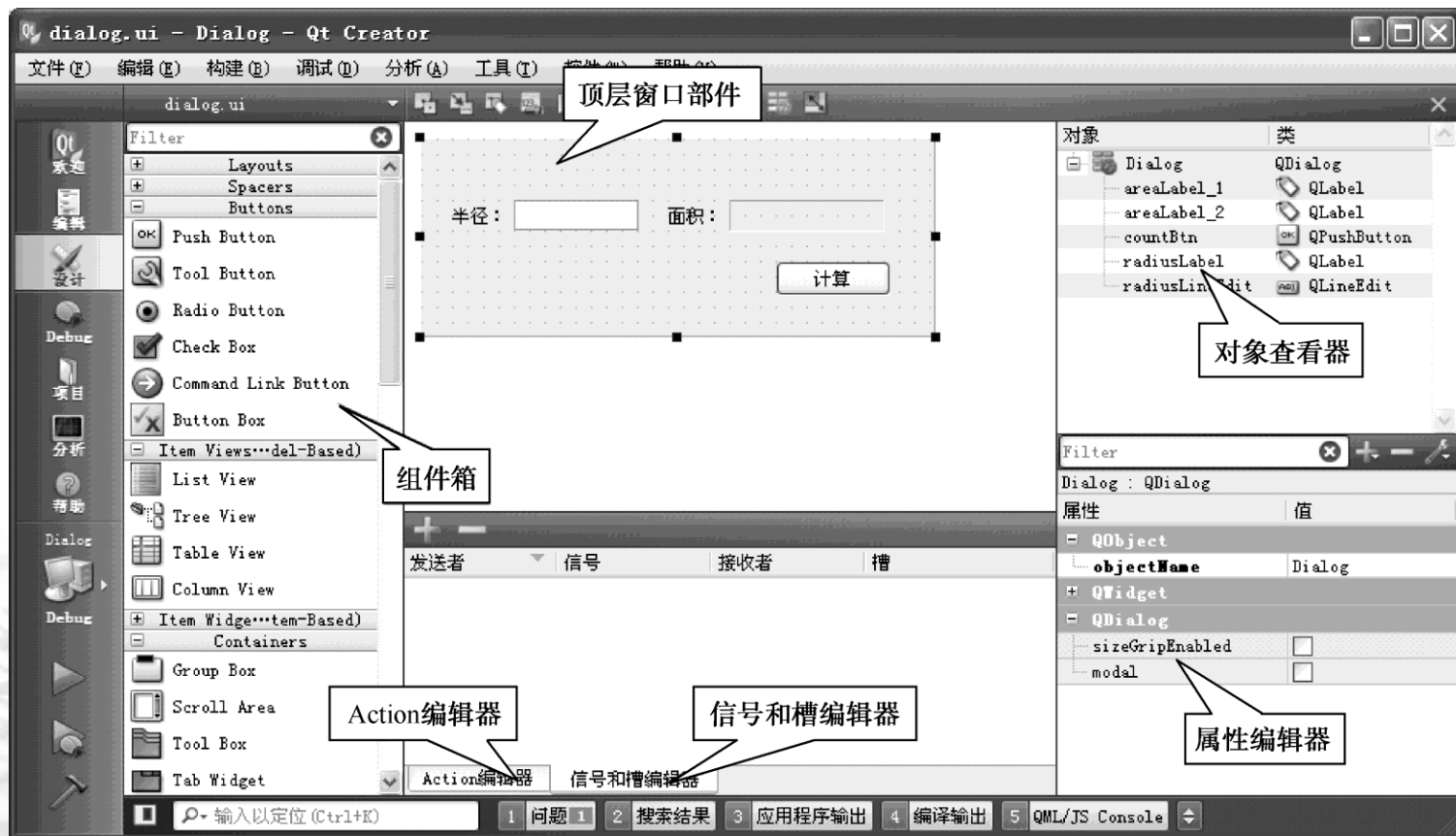
点击运行Qt Creator，出现欢迎界面，如图1.2所示。





1.2.3 Qt 5开发环境

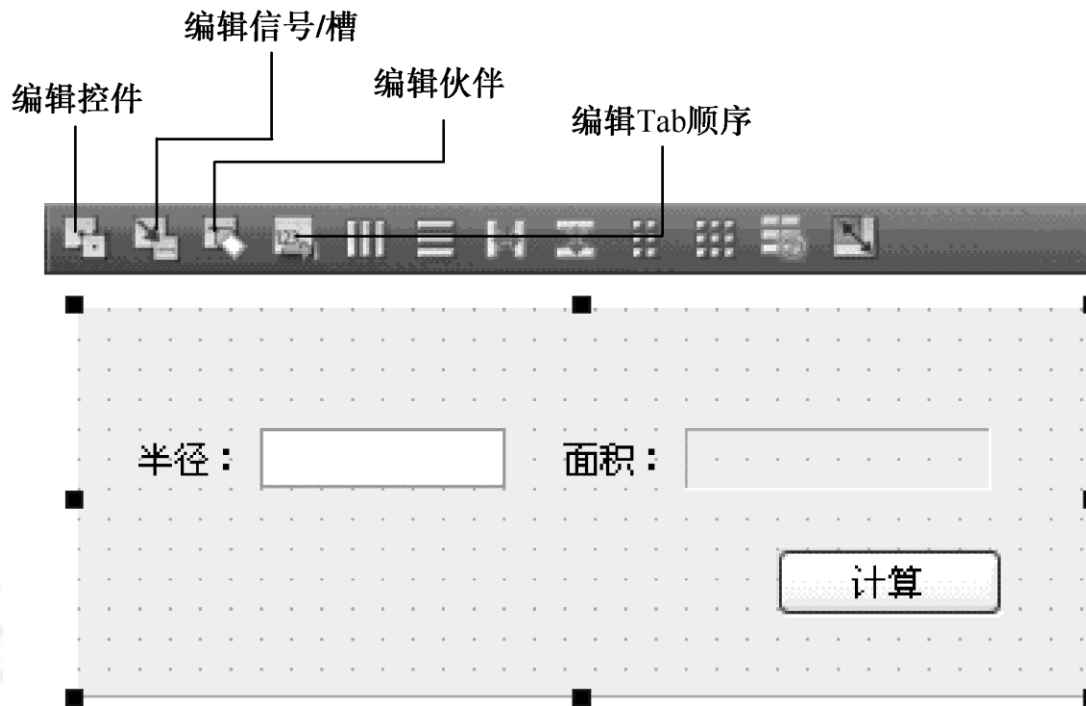
GUI用户界面设计 (Qt Designer) 界面如图1.3所示。





1.2.3 Qt 5开发环境

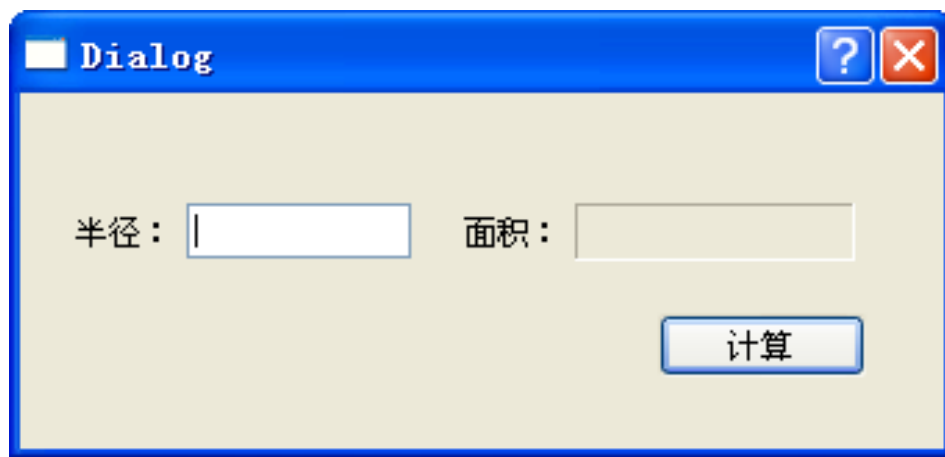
进入Qt设计器主界面后，看到的中间部分（如图1.4所示）就是将要设计的顶层窗口部件（顶层窗口部件是其他子窗口部件的载体）。





1.3 Qt 5开发步骤及实例

本实例要实现的功能是，当用户输入一个圆的半径后，可以显示计算后的圆的面积值。运行效果如图1.5所示。





1.3.1 设计器Qt 5 Designer实现

1. 界面设计

步骤如下。

(1) 单击运行Qt Creator，进入欢迎界面如图1.2所示。单击“文件”→“新建文件或项目...”命令，创建一个新的工程，如图1.6所示。





1.3.1 设计器Qt 5 Designer实现

(2) 单击选择“Qt Gui 应用”，单击“选择”按钮，进入下一步。
这里因为需要建立一个Gui项目，所以选择“Qt Gui 应用”，如图1.7所示。





1.3.1 设计器Qt 5 Designer实现

(3) 选择保存项目的路径并定义自己项目的名字。

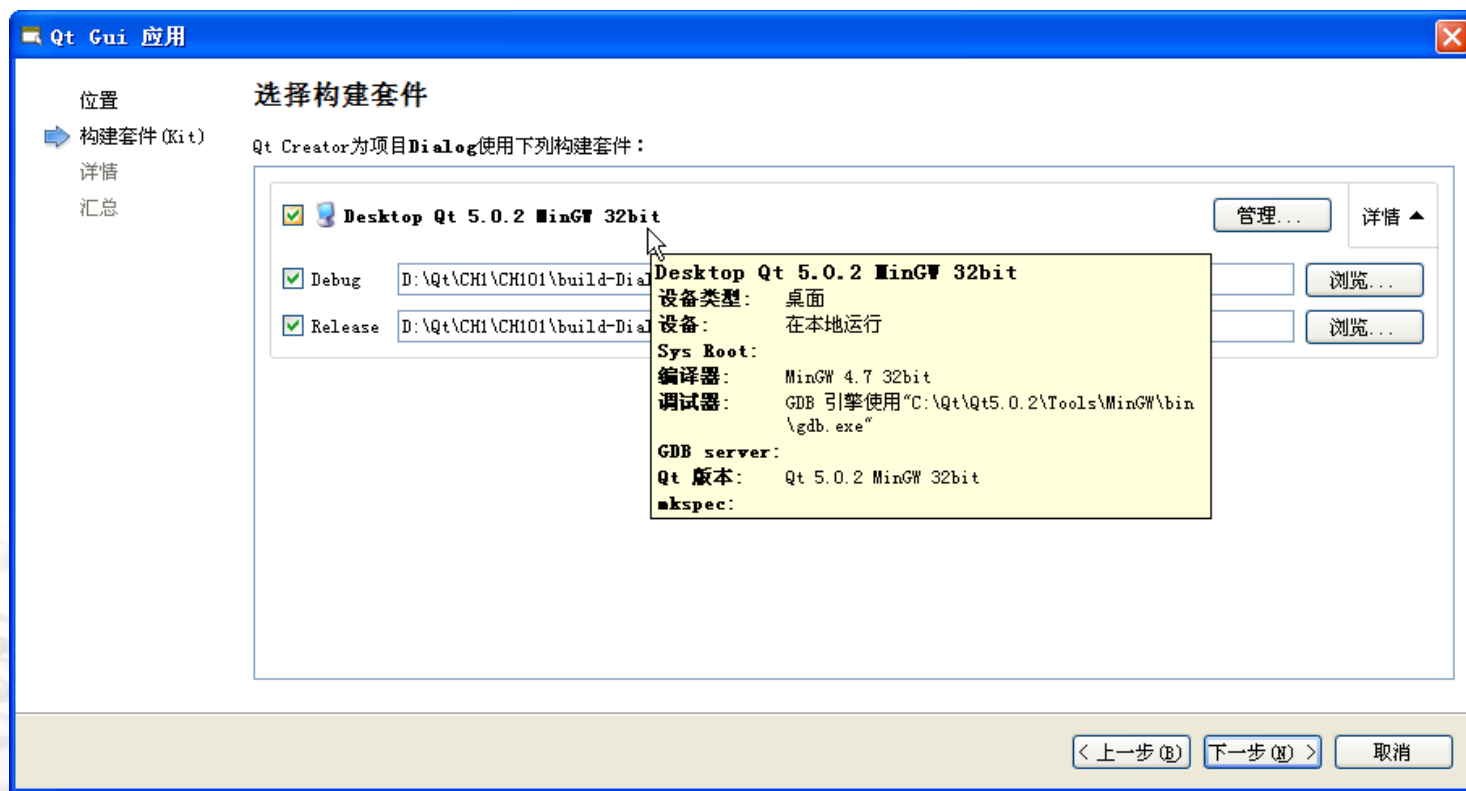
项目命名没有大小写要求，依据个人习惯命名即可。这里将项目命名为Dialog，保存路径为D:\Qt\CH1\CH101，如图1.8所示。单击“下一步”按钮进入下一步骤。





1.3.1 设计器Qt 5 Designer实现

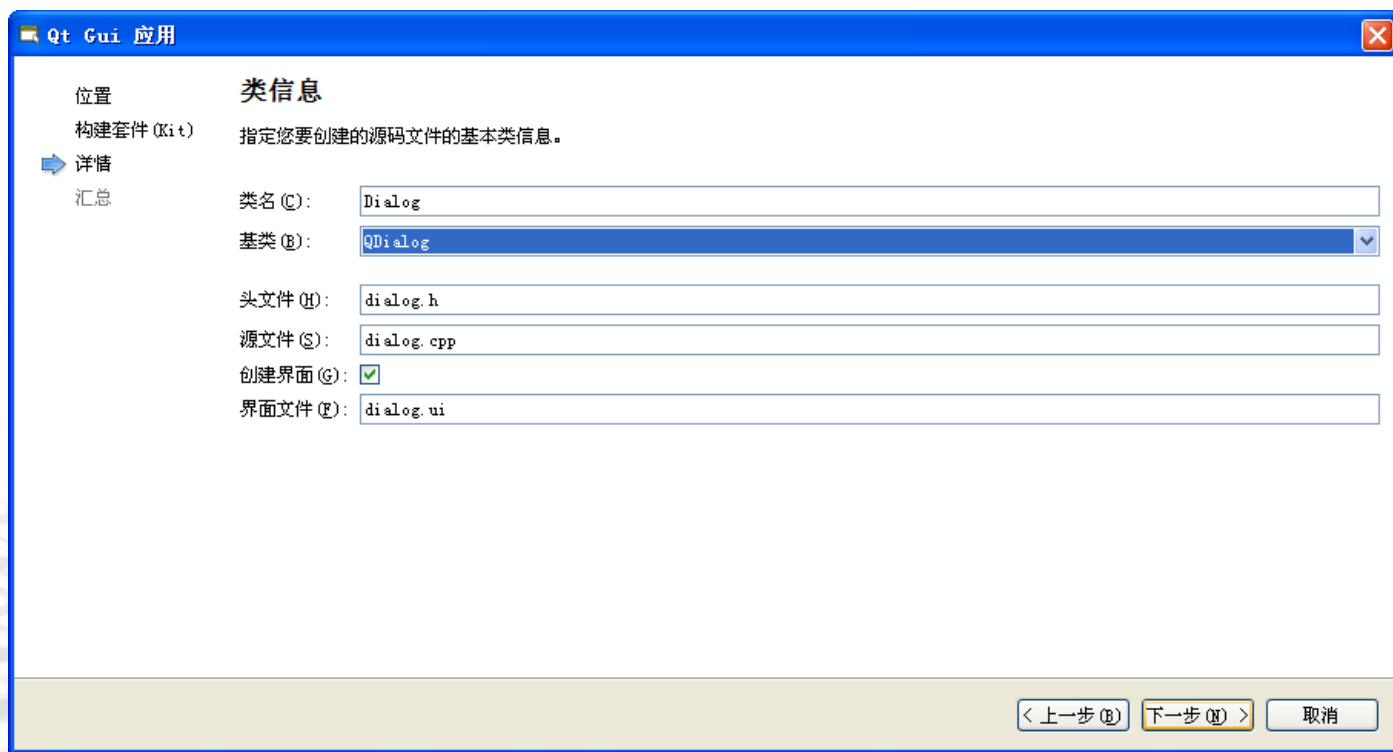
(4) 弹出“选择构建套件”界面，系统默认已指定C++的编译器和调试器，如图1.9所示，直接单击“下一步”按钮进入下一步骤即可。





1.3.1 设计器Qt 5 Designer实现

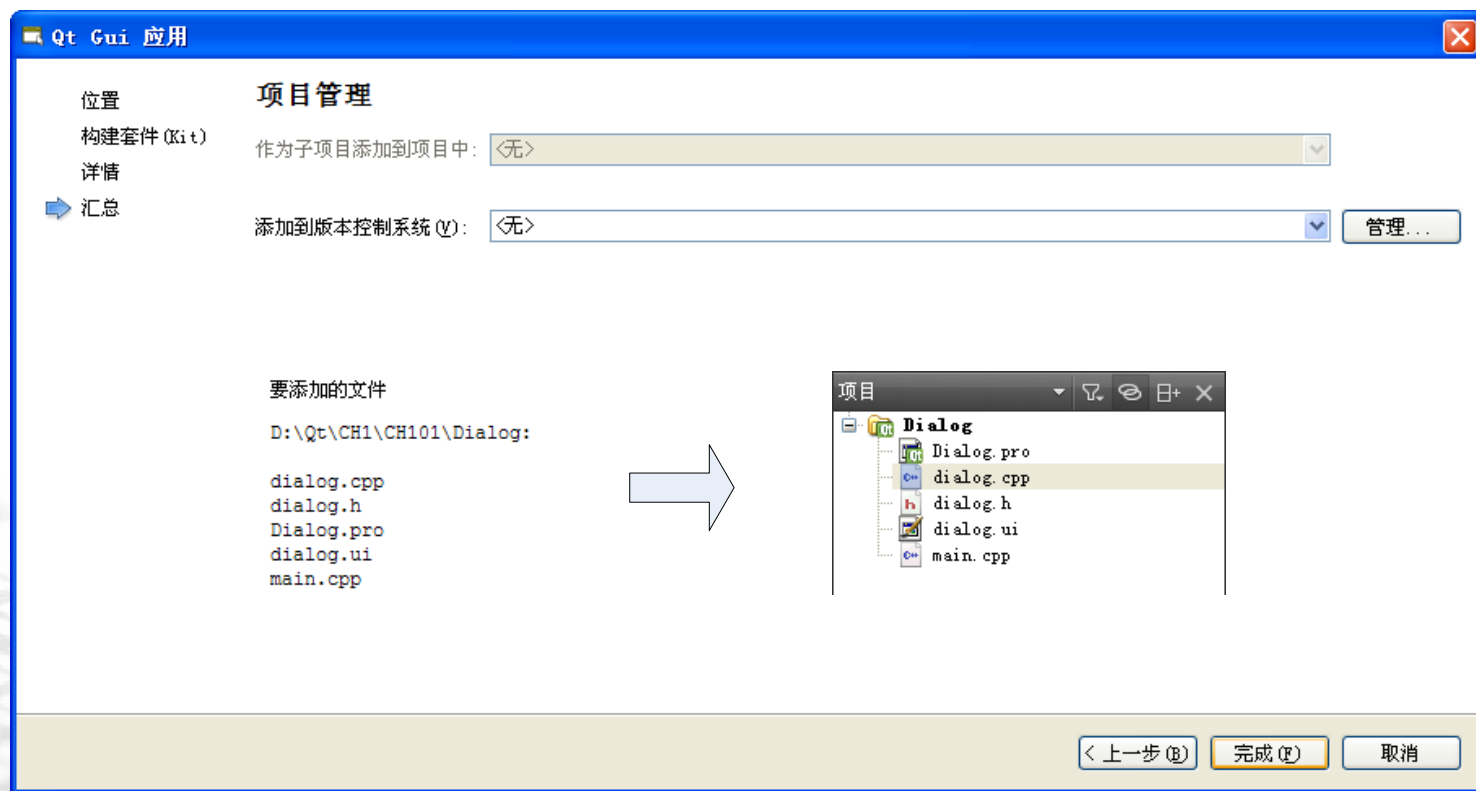
(5) 根据实际需要，选择一个“基类”。这里选择QDialog对话框类作为基类，这时“类名”、“头文件”、“源文件”及“界面文件”都出现默认的文件名。默认选中“创建界面”复选框，表示需要采用自带的界面设计器来设计界面，否则需要利用代码完成界面的设计，如图1.10所示。






1.3.1 设计器Qt 5 Designer实现

(6) 单击“完成”按钮完成创建，相应的文件自动加载到文件列表中，如图1.11所示。





1.3.1 设计器Qt 5 Designer实现

单击中间灰色一列工具栏中的过滤符号 () 后，弹出一个下拉列表，使两个项目都是被勾选后处于选中状态（默认选项“简化树形视图”没选中，“隐藏生成的文件”处于选中状态），如图1.12 (a) 所示。

若单击其中“简化树形视图”项取消选中状态，此时文件列表中的文件自动分类显示，如图1.12 (b) 所示。



(a)



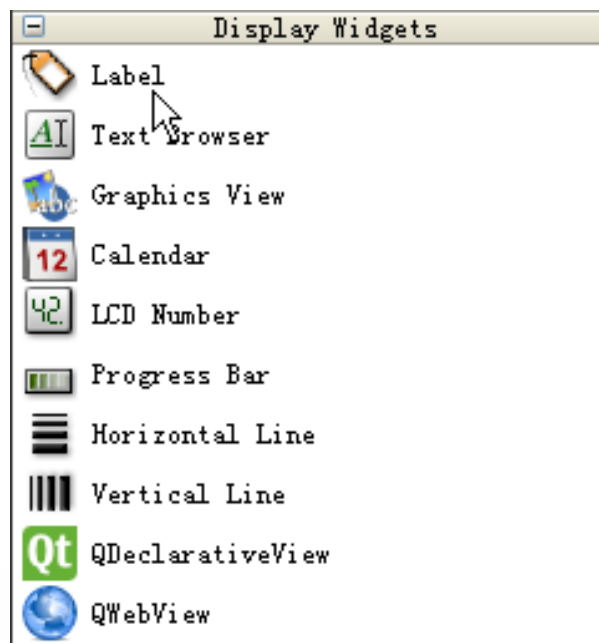
(b)



1.3.1 设计器Qt 5 Designer实现

(7) 双击dialog.ui，进入界面设计器Qt Designer编辑状态，开始进行设计器（Qt Designer）编程。

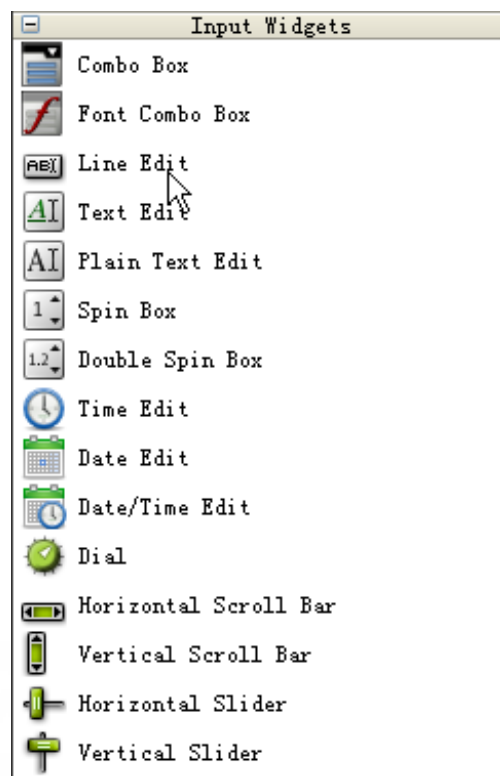
拖曳控件容器栏的滑动条，在最后的Display Widgets容器栏（如图1.13所示）中找到Label标签控件，拖曳三个此控件到中间的编辑框中。





1.3.1 设计器Qt 5 Designer实现

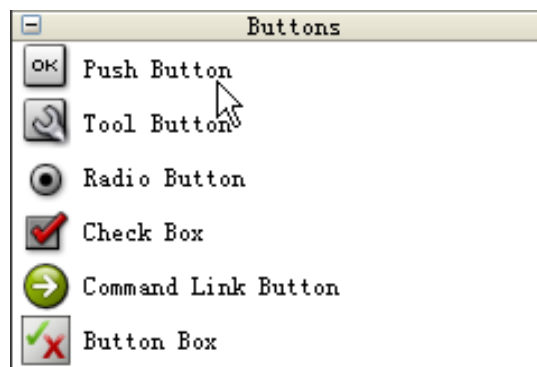
在Input Widgets容器栏（如图1.14所示）中找到LineEdit文本控件，拖曳此控件到中间的编辑框中，用于输入半径值；





1.3.1 设计器Qt 5 Designer实现

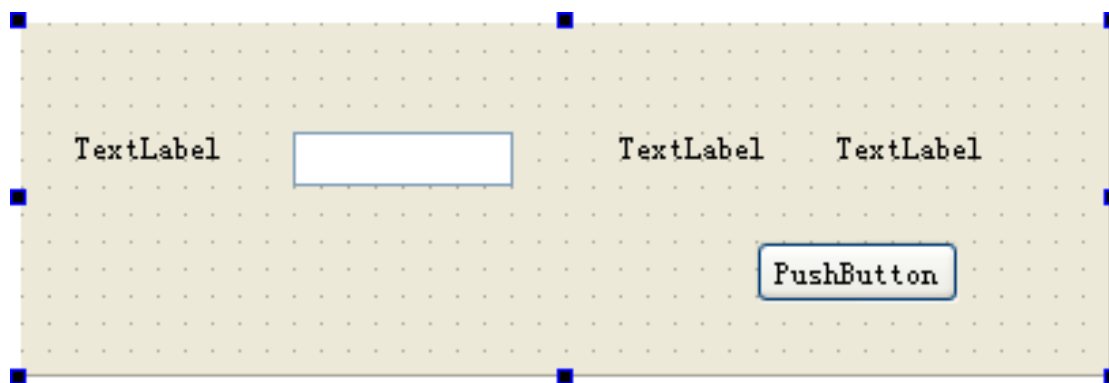
在Buttons容器栏（如图1.15所示）中找到PushButton按钮控件，拖曳此控件到中间的编辑框中，用于提交响应单击事件。





1.3.1 设计器Qt 5 Designer实现

调整各控件的位置，单击编辑框的空白处使编辑框处于被选中状态，拖曳右下角的小方框，调整整个框架的大小，直至调整到适当大小为止，调整后的布局如图1.16所示。





1.3.1 设计器Qt 5 Designer实现

下面将修改拖曳到编辑框中的各控件的属性，如图1.17所示，各控件属性见表1.1。

对象	类
Dialog	QDialog
areaLabel_1	QLabel
areaLabel_2	QLabel
countBtn	QPushButton
radiusLabel	QLabel
radiusLineEdit	QLineEdit

Class↵	text↵	objectName↵
QLabel↵	半径: ↵	radiusLabel↵
QLineEdit↵	↵	radiusLineEdit↵
QLabel↵	面积: ↵	areaLabel_1↵
QLabel↵	↵	areaLabel_2↵
QPushButton↵	计算↵	countBtn↵



1.3.1 设计器Qt 5 Designer实现

修改控件Text值的方法有如下两种。

- 直接双击控件本身即可修改。
- 在Qt Designer设计器的属性栏中修改，如修改表示半径的Label标签，如图1.18所示。

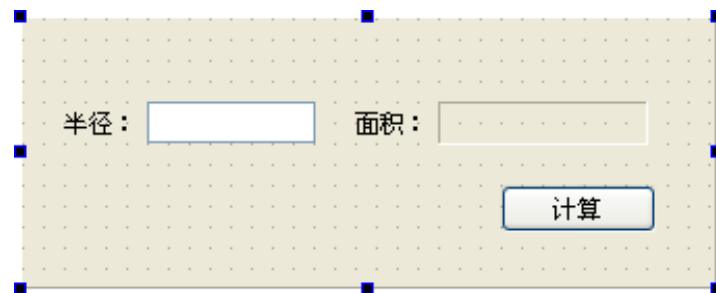
属性	值	= QLabel	
= QObject		+ text	半径：
objectName	radiusLabel	textFormat	AutoText
= QWidget		pixmap	



1.3.1 设计器Qt 5 Designer实现


最后，修改areaLabel_2的“frameShape”为Panel；“frameShadow”为Sunken，如图1.19所示。最终效果如图1.20所示。

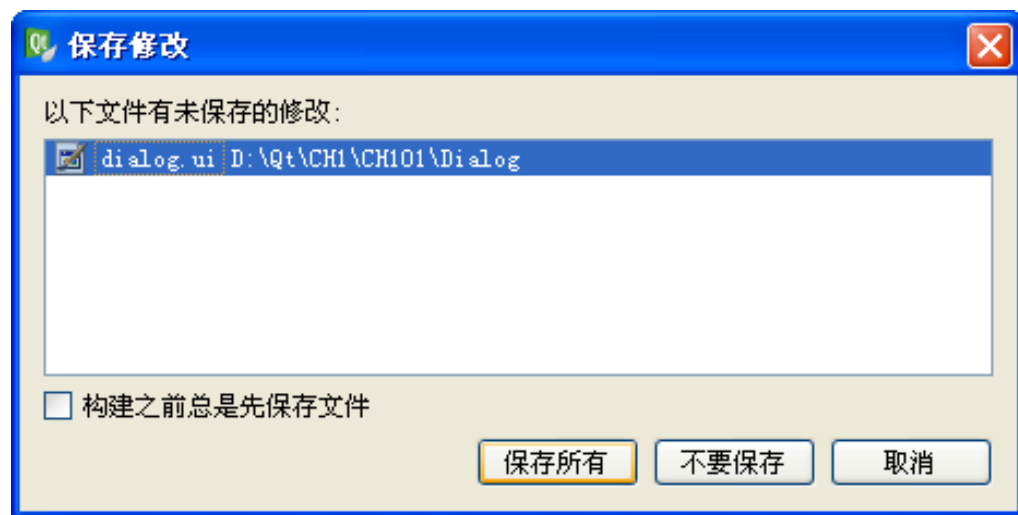
QFrame	
frameShape	Panel
frameShadow	Sunken
lineWidth	1
midLineWidth	0





1.3.1 设计器Qt 5 Designer实现

下面单击左下角的运行按钮（）或者使用组合键【Ctrl+R】运行程序，这时系统提示是否保存，单击“保存所有”按钮，如图1.21所示。





1.3.1 设计器Qt 5 Designer实现

2. 编写相应的计算圆面积代码

首先简单认识一下Qt编程环境。找到文件列表中自动添加的main.cpp文件，如图1.12所示。每个工程都有一个执行的入口函数，此文件中的main()函数就是此工程的入口。下面详细介绍一下main()函数的相关内容：

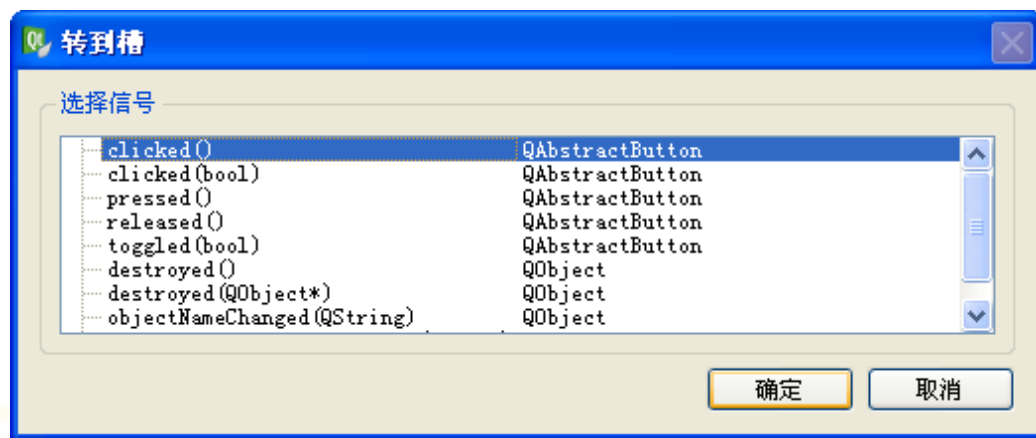
```
#include "dialog.h"
#include <QApplication>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Dialog w;
    w.show();
    return a.exec();
}
```




1.3.1 设计器Qt 5 Designer实现

方式1：在LineEdit文本框内输入半径值，然后单击“计算”按钮，则在areaLabel_2中显示对应的圆面积。编写代码步骤如下。

(1) 在“计算”按钮上单击鼠标右键，在弹出的下拉菜单中选择“转到槽...”命令，如图1.22所示。在弹出的对话框中选择“clicked()”信号，如图1.23所示。





1.3.1 设计器Qt 5 Designer实现

(2) 进入dialog.cpp文件中按钮单击事件的槽函数on_countBtn_clicked()。信号与槽连接的具体说明参照本书后面提供的知识点链接部分。在此函数中添加如下代码：

```
void Dialog:: on_countBtn_clicked()
{
    bool ok;
    QString tempStr;
    QString valueStr=ui->radiusLineEdit->text();
    int valueInt=valueStr.toInt(&ok);
    double area=valueInt*valueInt*PI;//计算圆面积
    ui->areaLabel_2->setText(tempStr.setNum(area));
}
```

(3) 在此文件开始处添加以下语句：

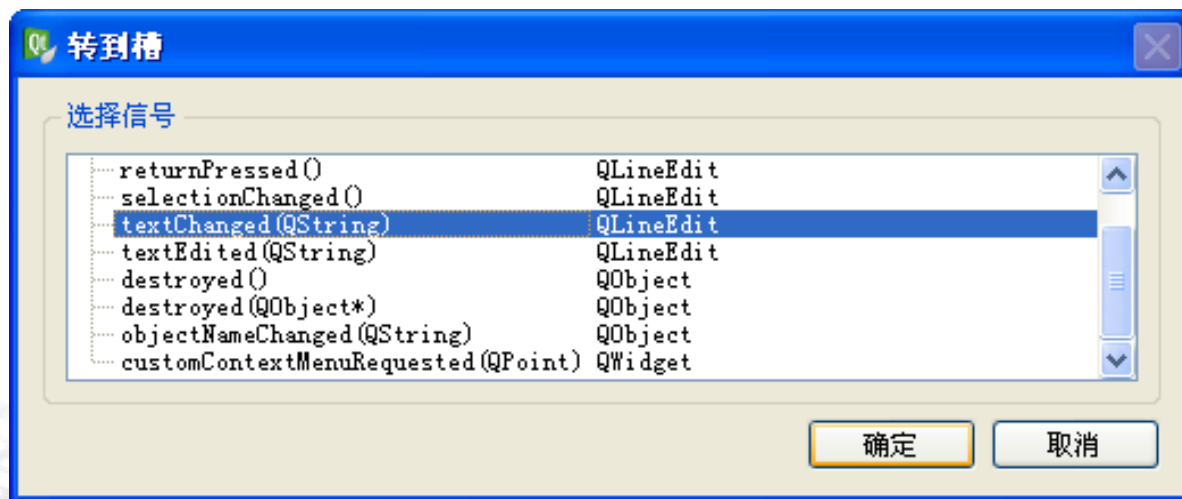
```
const static double PI=3.1416;
```



1.3.1 设计器Qt 5 Designer实现

方式2: 在LineEdit内输入半径值，不需要单击按钮触发单击事件，直接就在areaLabel_2中显示圆面积。编写代码步骤如下。

(1) 在“LineEdit”编辑框上单击鼠标右键，在弹出的下拉菜单中选择“转到槽...”菜单项，在弹出的对话框中选择“textChanged(QString)”信号，如图1.24所示。





1.3.1 设计器Qt 5 Designer实现

(2) 单击“确定”按钮，进入dialog.cpp文件中的文本编辑框，改变值内容事件的槽函数on_radiusLineEdit_textChanged(const QString &arg1)。在此函数中添加如下代码：

```
void Dialog::on_radiusLineEdit_textChanged(const QString &arg1)
{
    bool ok;
    QString tempStr;
    QString valueStr=ui->radiusLineEdit->text();
    int valueInt=valueStr.toInt(&ok);
    double area=valueInt*valueInt*PI;//计算圆面积
    ui->areaLabel_2->setText(tempStr.setNum(area));
}
```



1.3.2 代码实现

(1) 首先创建一个新工程。创建过程和本书1.3.1节中的第(1)步骤~第(6)步骤相同，只是在第(3)步骤中，项目命名为Dialog且保存路径为D:\Qt\CH1\CH102，在第(5)步骤中，取消“创建界面”复选框的选中状态。





1.3.2 代码实现

(2) 在上述工程的dialog.h中添加如下加黑代码:

```
class Dialog : public QDialog
{
    Q_OBJECT
public:
    Dialog(QWidget *parent = 0);
    ~Dialog();
private:
    QLabel *label1,*label2;
    QLineEdit *lineEdit;
    QPushButton *button;
};
```

此时要在文件最开始加入头文件:

```
#include <QLabel>
#include <QLineEdit>
#include <QPushButton>
```



1.3.2 代码实现

(3) 在dialog.cpp 中添加如下代码:

```
Dialog::Dialog(QWidget *parent)
    : QDialog(parent)
{
    label1=new QLabel(this);
    label1->setText(tr("请输入圆的半径: "));
    lineEdit=new QLineEdit(this);
    label2=new QLabel(this);
    button=new QPushButton(this);
    button->setText(tr("显示对应圆的面积"));
    QGridLayout *mainLayout=new QGridLayout(this);
    mainLayout->addWidget(label1,0,0);
    mainLayout->addWidget(lineEdit,0,1);
    mainLayout->addWidget(label2,1,0);
    mainLayout->addWidget(button,1,1);
}
```

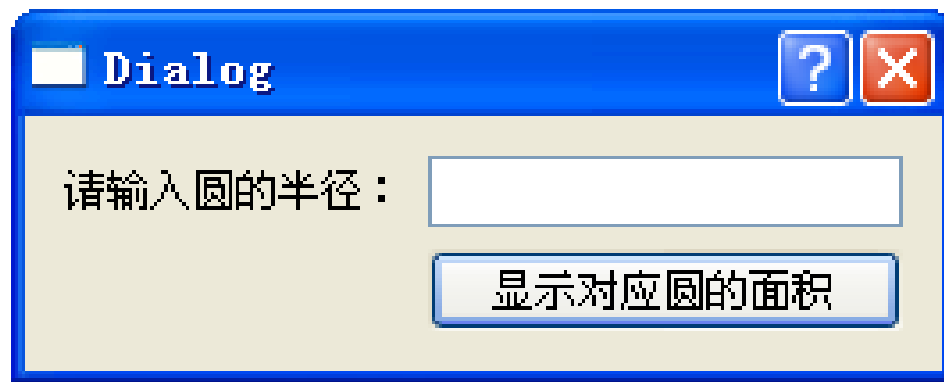


1.3.2 代码实现

(4) 在此文件一开始添加头文件:

```
#include <QGridLayout>
```

运行程序结果如图1.25所示。





1.3.2 代码实现

方式1: 在LineEdit文本框内输入所需圆的半径值，单击“显示对应圆的面积”按钮后，在label2中显示相对应的圆的面积值。

(1) 打开dialog.h文件，在类构造函数和控件成员声明后，添加如下代码：

```
class Dialog : public QDialog
{
    ... ..
    QPushButton *button;
private slots:
    void showArea();
};
```



1.3.2 代码实现

(2) 打开dialog.cpp 文件，在构造函数中添加如下加黑代码：

```
Dialog::Dialog(QWidget *parent)
    : QDialog(parent)
{
    ... ..
    mainLayout->addWidget(button,1,1);
    connect(button,SIGNAL(clicked()),this,SLOT(showArea()));
}
```



1.3.2 代码实现

(3) 在showArea()中实现显示圆面积功能，代码如下：

```
void Dialog::showArea()
{
    bool ok;
    QString tempStr;
    QString valueStr=lineEdit->text();
    int valueInt=valueStr.toInt(&ok);
    double area=valueInt*valueInt*PI;
    label2->setText(tempStr.setNum(area));
}
```

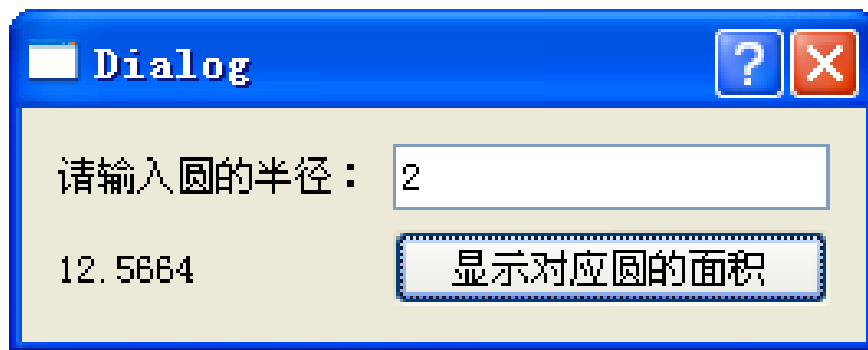


1.3.2 代码实现

(4) 在此文件开始处添加全局变量:

```
const static double PI=3.1416;
```

(5) 在LineEdit中输入圆半径值，单击“显示对应圆的面积”按钮后，在label2中显示圆面积值，如图1.26所示。





1.3.2 代码实现

方式2：在LineEdit文本框中输入所需圆的半径值后，不必单击“显示对应圆的面积”按钮，直接在label2中显示圆的面积值。操作步骤和方式1相同，只是在上述第（2）步骤中，添加的代码修改为如下加黑代码：

```
Dialog::Dialog(QWidget *parent)
    : QDialog(parent)
{
    ... ..
    mainLayout->addWidget(button,1,1);

connect(lineEdit,SIGNAL(textChanged(QString)),this,SLOT(showArea()));
}
```