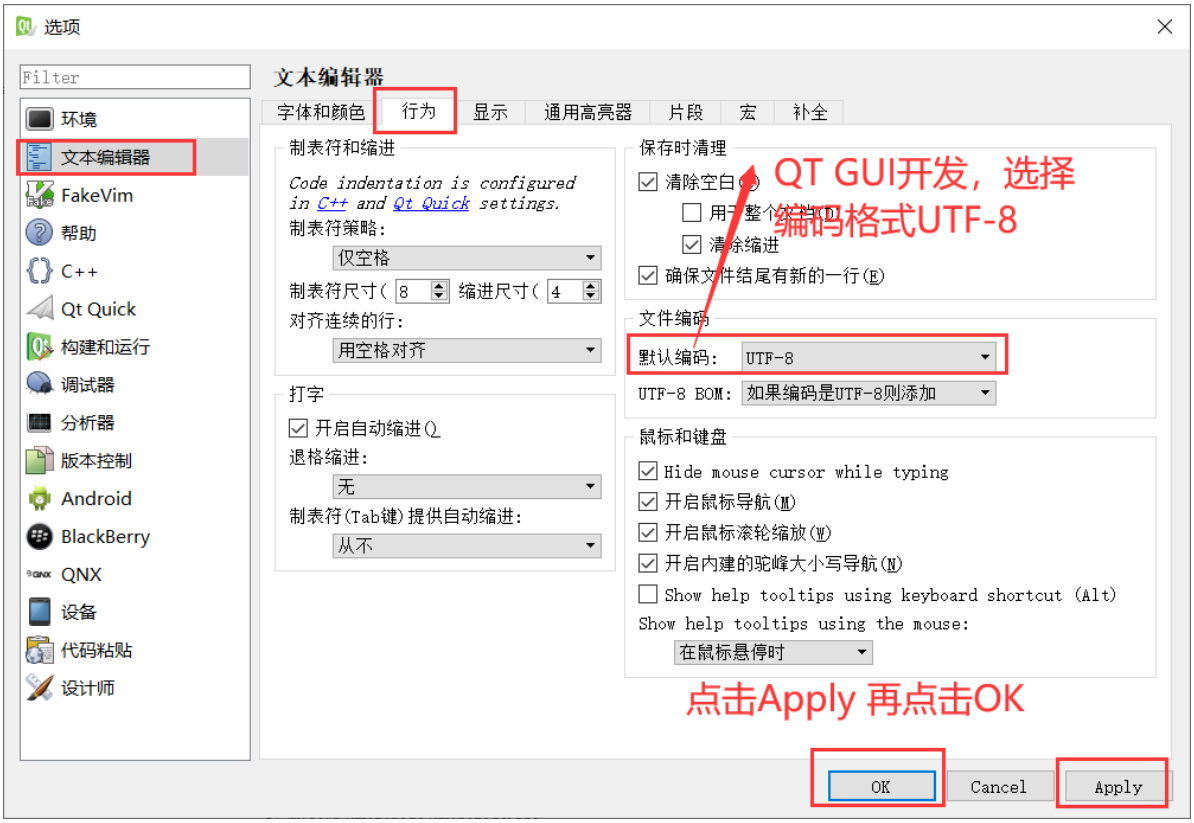
一、Qt简介

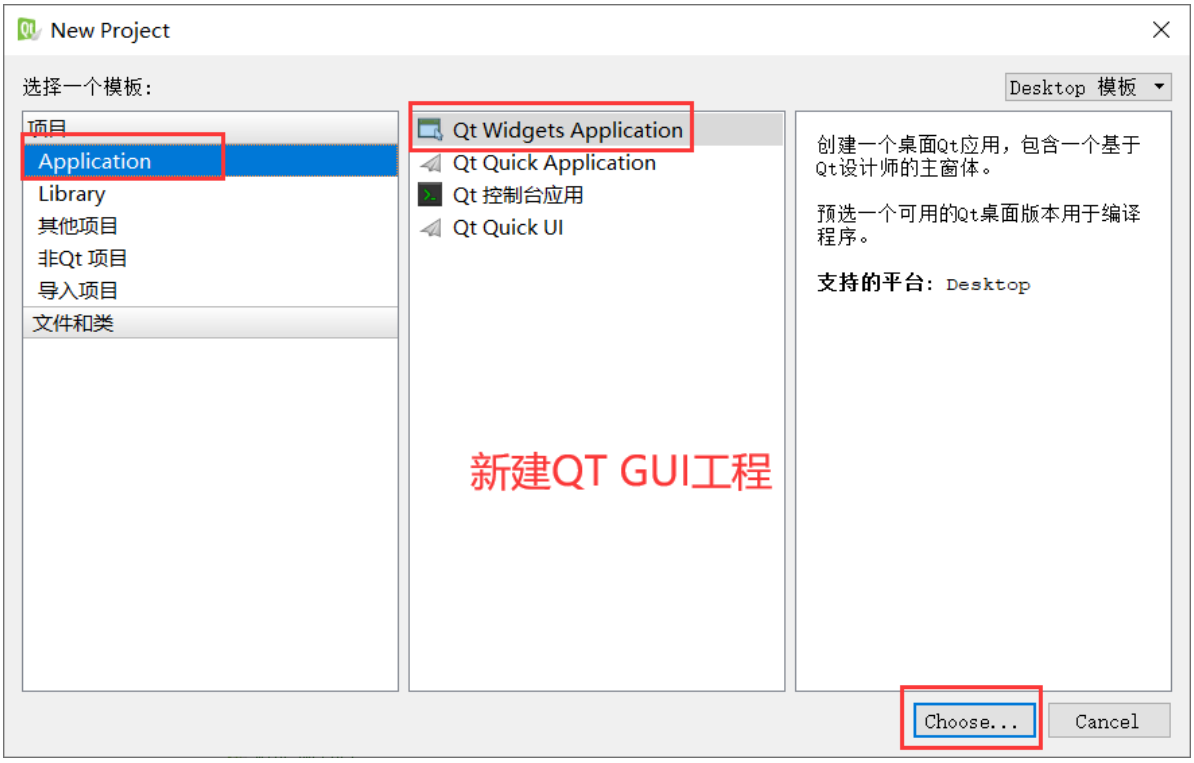
- Qt是一个1991年由Qt Company开发的 跨平台 C++ 图形用户界面应用程序开发框架。
- Qt内部包含了一组类库　我们可以使用他来开发GUI程序（非GUI程序）
- Qt是面向对象的框架，使用特殊的代码生成扩展（称为元对象编译器(Meta Object Compiler, moc)）

二、Qt优势

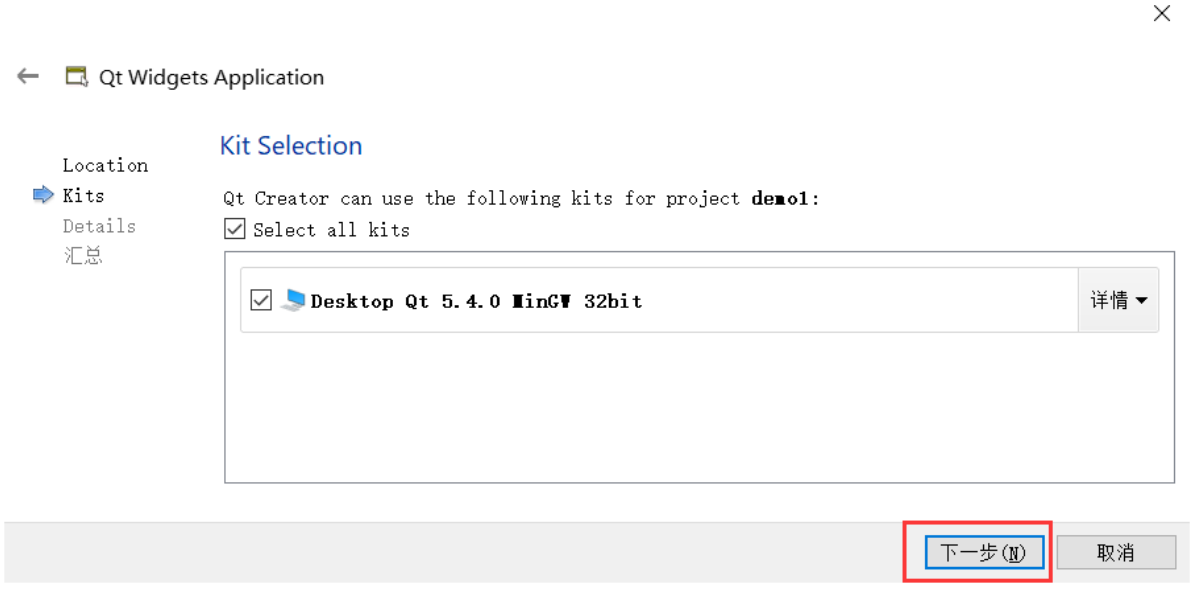
- 跨平台
- 面向对象的思想
- 信号与槽，实现各个对象之间的通信，与c++唯一的不同
- 很多的api函数

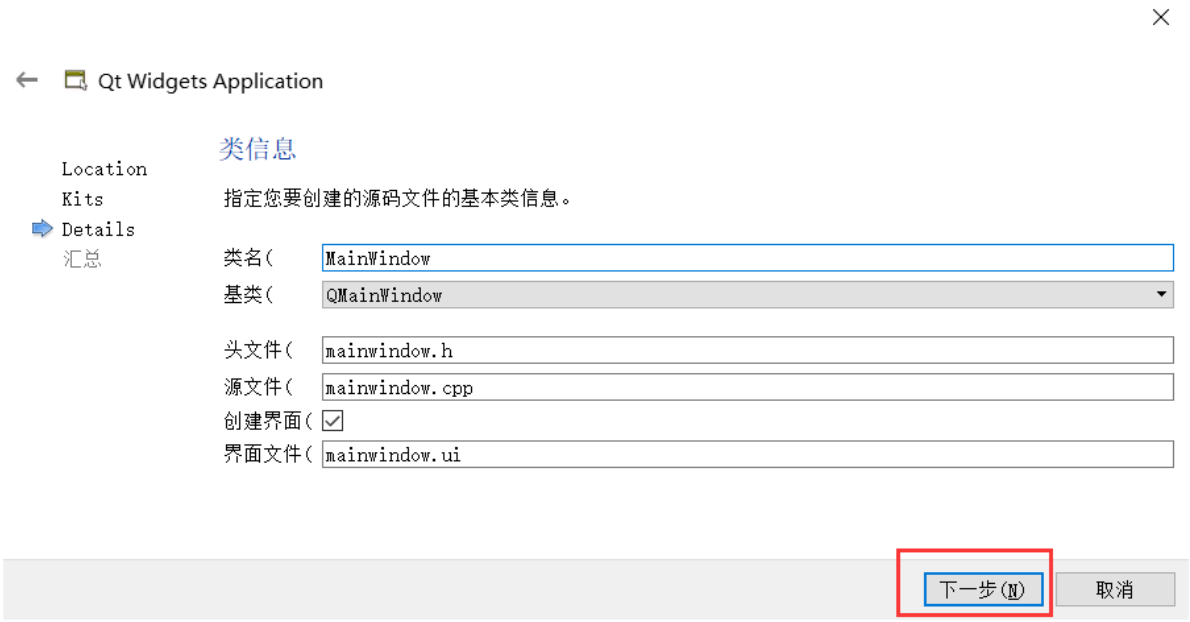
三、第一个Qt项目




New Project
选择一个模板：
Desktop 模板
项目
Application
Library
其他项目
非Qt 项目
导入项目
文件和类
Qt Widgets Application
Qt Quick Application
Qt 控制台应用
Qt Quick UI
创建一个桌面Qt应用，包含一个基于Qt设计师的主窗体。
预选一个可用的Qt桌面版本用于编译程序。
支持的平台：Desktop
新建QT GUI工程
Choose...
Cancel

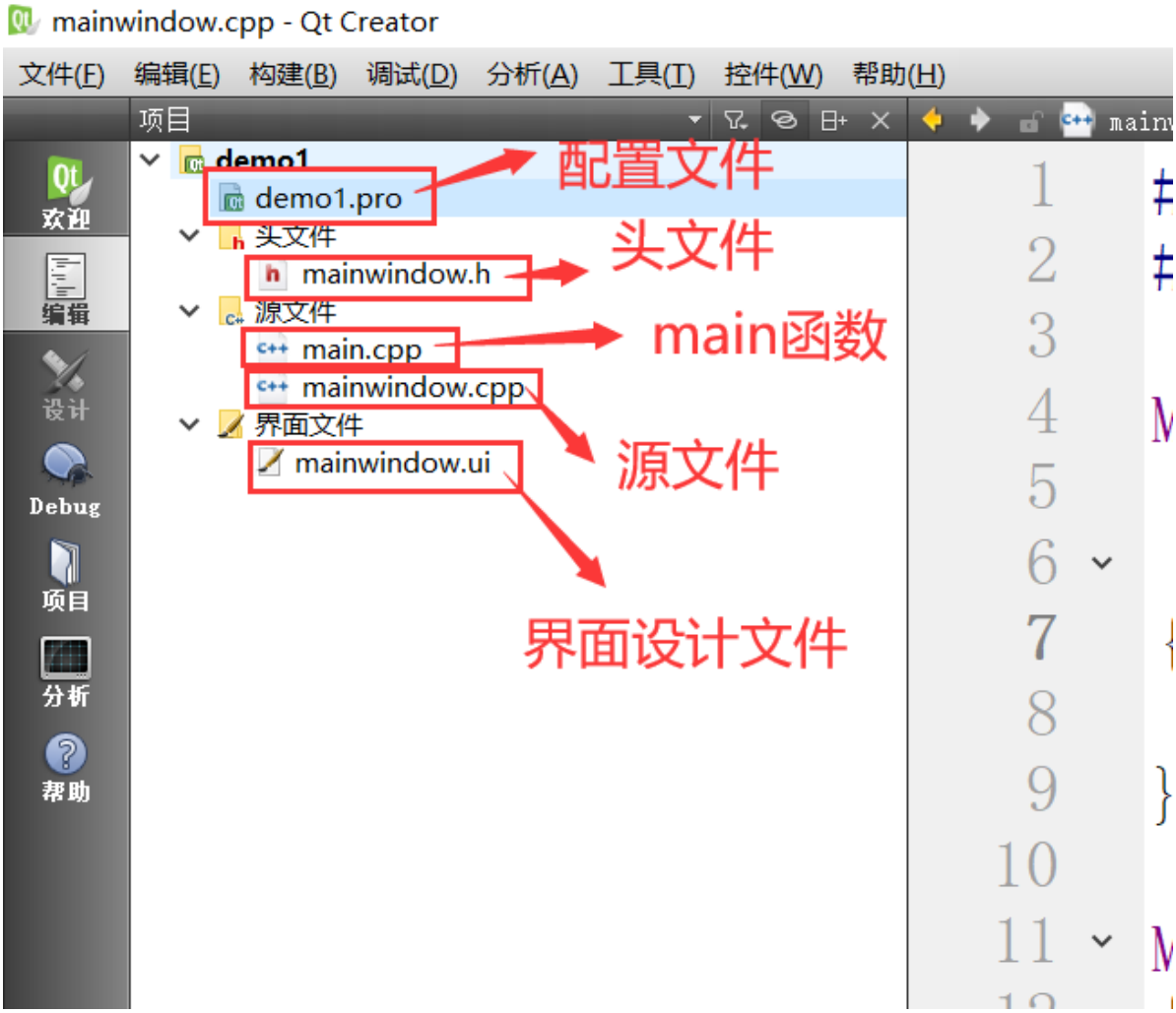

Qt Widgets Application
Location
Kits
Details
汇总
项目介绍和位置
This wizard generates a Qt Widgets Application project. The application derives by default from QApplication and includes an empty widget.
名称：
demo1
创建路径：
D:\mews
浏览...
设为默认的项目路径
下一步
取消


Qt Widgets Application
Location
Kits
Details
汇总
Kit Selection
Qt Creator can use the following kits for project demo1:
Select all kits
Desktop Qt 5.4.0 MinGW 32bit
详情
下一步(N)
取消


Qt Widgets Application
Location
Kits
Details
汇总
类信息
指定您要创建的源码文件的基本类信息。
类名(
MainWindow
基类(
QMainWindow
头文件(
mainwindow.h
源文件(
mainwindow.cpp
创建界面(
界面文件(
mainwindow.ui
下一步(N)
取消


Qt Widgets Application
Location
Kits
Details
汇总
项目管理
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
D:\mews\demo1:
demo1.pro
main.cpp
mainwindow.cpp
mainwindow.h
mainwindow.ui
完成(F)
取消

1. 工程目录介绍



2. 信号与槽

- 信号与槽是QT的核心机制
- 当一个对象的状态发生变化时，通过信号的方式通知其他对象，其他对象通过执行相应的槽函数来响应该信号。
- 信号与槽是QT提供的任意两个（QObject）对象之间的通信机制，常用来完成界面操作的响应。

2.1 信号

- 可以看作是一个请求或者一个动作的标志。
- 信号是属于对象的
- 当对象的状态改变时，发射信号

2.2 槽

- 槽函数：本质 就是类的成员函数，我们可以调用类的成员函数一样来调用槽函数
- 槽函数可以跟 信号建立起关联，而普通的成员函数不可以
- 信号槽工作的过程是：当一个对象发射一个信号的时候，则和其连接的对象的槽函数进行处理，等槽函数处理完成之后退出并执行接下来的内容

3. 初始UI设计步骤

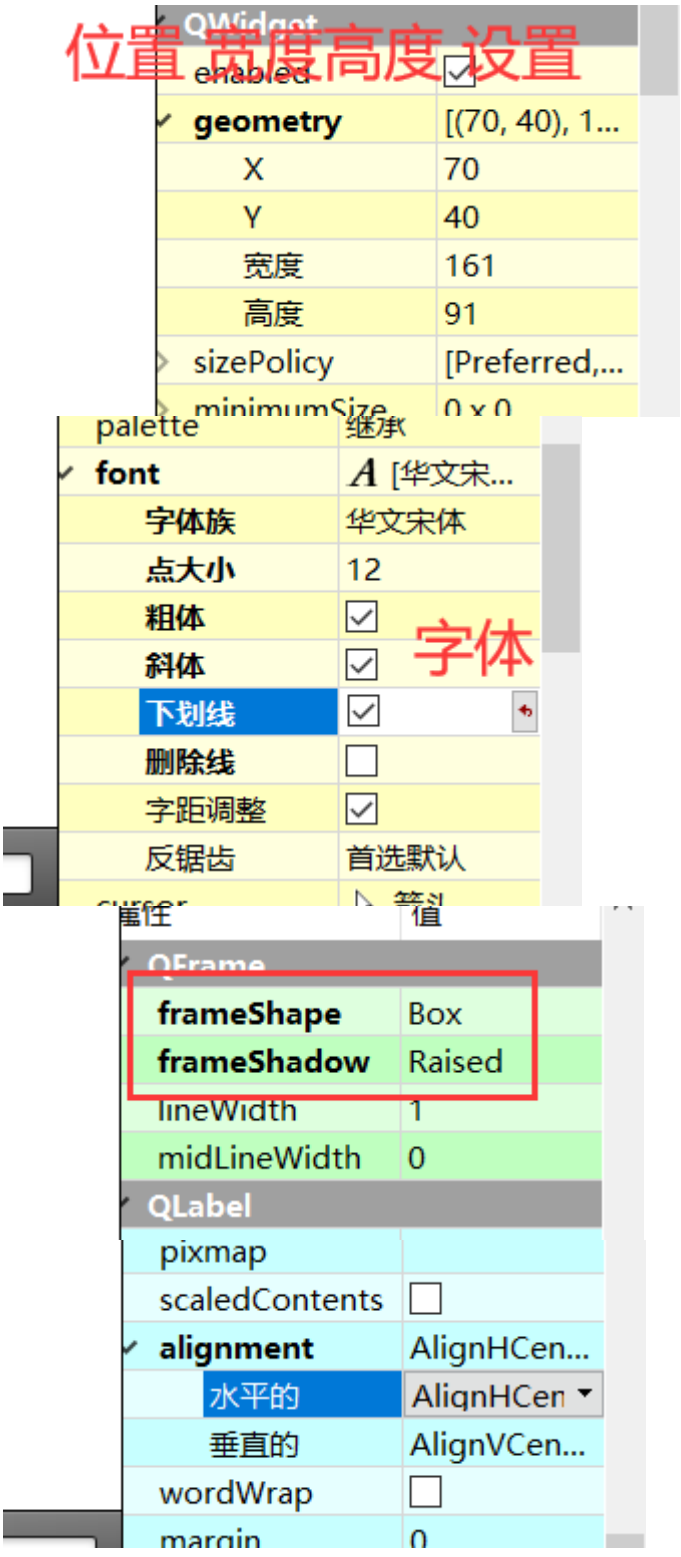
(1) 拖拽想要的控件至指定位置

label 显示文本
pushButton 按钮

(2) 更改objectName，以供代码操作使用

(3) 在代码中使用ui指针找到指定控件的objectName并进行操作

(4) 添加信号和槽等候用户操作



4. 信号定义

信号函数声明（信号函数不可以被实现）

```
signals:
        void signalTest();
```

5. 槽函数定义

```
访问权限+slots:
        槽函数声明
public slots:
    void slotTest();
```

6. 将信号与槽函数绑定

```
connect(ui->btn_1,SIGNAL(clicked()),this,SLOT(slotShowHello()));
connect(this,SIGNAL(signalTest()),this,SLOT(slotTest()));
```

7. pushButton和label控件

7.1 点击pushButton，label显示 字符串

```
//方法一
ui->labelShowText->setText("hello world");
//方法二
QString s = "hello world";
ui->labelShowText->setText(s);
```

7.2 点击pushButton，label显示 数字

```
//方法一
ui->labelShowNum->setNum(100);
//方法二
int num = 100;
ui->labelShowNum->setNum(num);
```

#练习1

```
重新建一个QT工程demo2
在拖拽两个按钮，共两个PushButton，一个label，实现功能
点击按钮1，label中显示字符串    你好
点击按钮2，label中显示变为数字 10086
```

#练习2

使用Label显示PushButton点击的次数，每点击一次PushButton，Label显示的数字加1

7.3 点击pushButton，label显示 图片

```
//label适应图片的大小
QImage img; //创建一个QImage对象
img.load("D:/images/heads/head1.jpg"); //加载图片，需要图片名称带有后缀名
ui->label->setPixmap(QPixmap::fromImage(img)); //将img图象显示在label中
ui->label->resize(img.width(),img.height()); //修改label的大小，设置图像的宽度和高度
```

```
//图片适应label大小
QImage img;//创建一个QImage对象
img.load("D:/images/heads/head1.jpg");//加载图片，需要图片名称带有后缀名
ui->label->resize(QSize(300,300));
QPainter painter;//创建一个画家对象
QImage nImg=img.scaled(ui->label->width(),ui->label->height());//对img缩放，按照label宽度和高度
//缩放之后，将缩放的图像保存到 nImg中    newImg
painter.drawImage(0,0,nImg);//用画家将nImg画出来
ui->label->setPixmap(QPixmap::fromImage(nImg)); //将nImg缩放后的图像显示在label中
```

```
//点击显示按钮，与之对应的槽函数
void MainWindow::on_pushButtonShow_clicked()
{

    //用label去适应图片的大小
    //    QImage img;//创建一个QImage图像对象
    //    if(!img.load("D:/images/heads/head22.jpg"))//如果加载失败返回值是false,!后进入条件句
    //    {
    //        qDebug() << "加载图片失败，你的路径写错了，仔细检查!!";
    //        exit(-1);//结束整个程序
    //    }
    //    ui->labelShowImage->setPixmap(QPixmap::fromImage(img));//将图像显示在label中
    //    //我们可以重置label的大小，让label的宽度和高度和 图像一样大
    //    ui->labelShowImage->resize(QSize(img.width(),img.height()));

    //对图片的大小进行缩放，去适应label的大小
    QImage img;//创建一个QImage图像对象
```

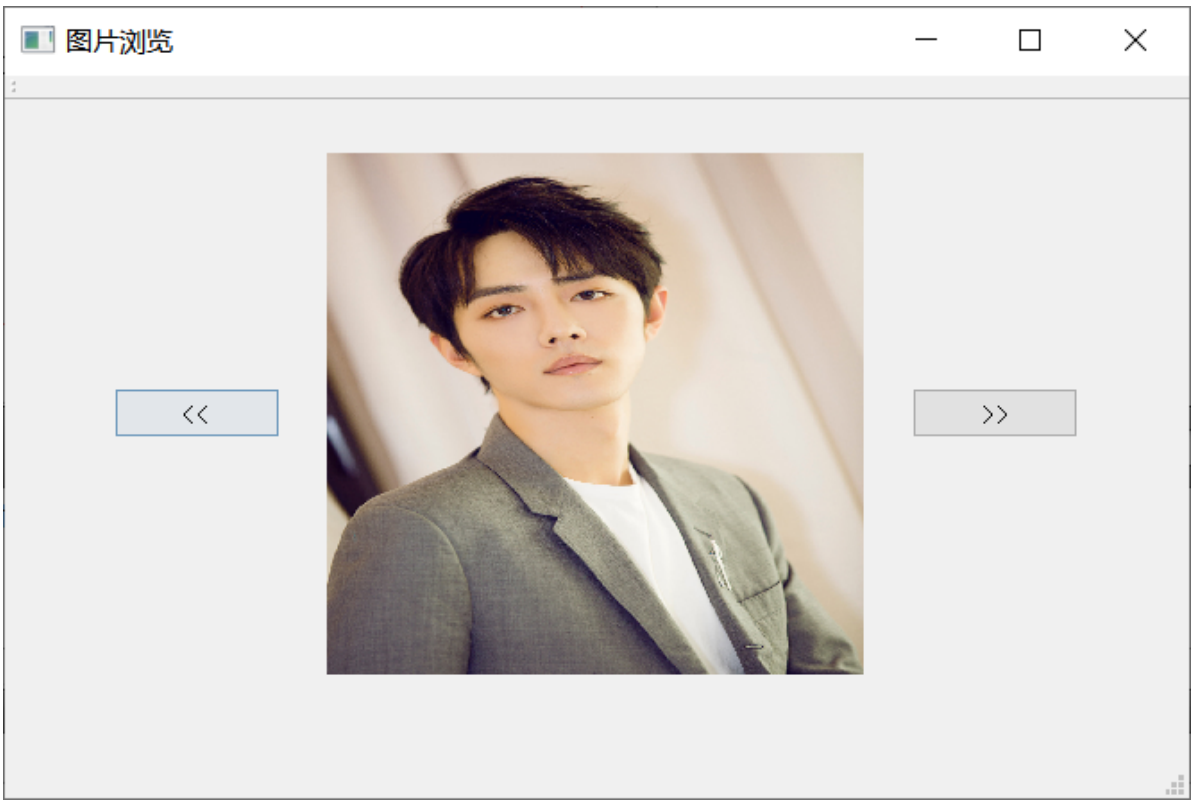
```
    if(!img.load("D:/images/heads/head22.jpg"))//如果加载失败返回值是
false,!后进入条件句
    {
        qDebug() << "加载图片失败，你的路径写错了，仔细检查!!";
        exit(-1);//结束整个程序
    }
    //缩放img图像的大小，按照label的宽度和高度进行缩放
    //将缩放后的图像保存的newImg中
    QImage newImg = img.scaled(ui->labelShowImage->width(), ui-
>labelShowImage->height());
    //用画家对象将新图像画出来
    QPainter painter;
    painter.drawImage(0,0,newImg);
    //将新的图像显示在label中
    ui->labelShowImage->setPixmap(QPixmap::fromImage(newImg));

}
```

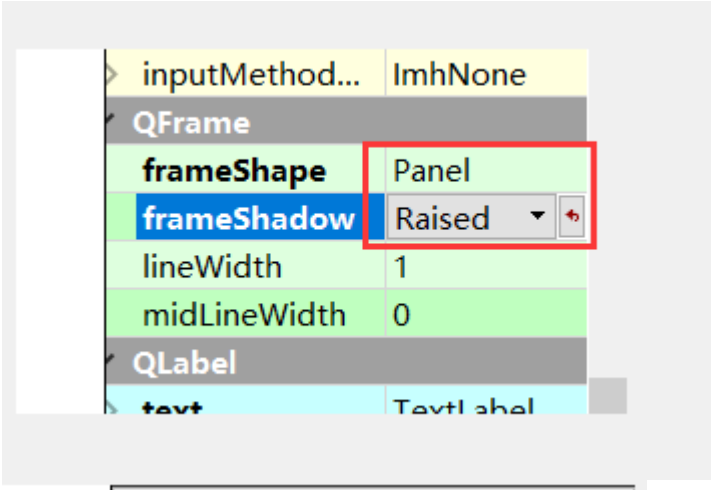
7.4 QDir获取目录中的所有文件

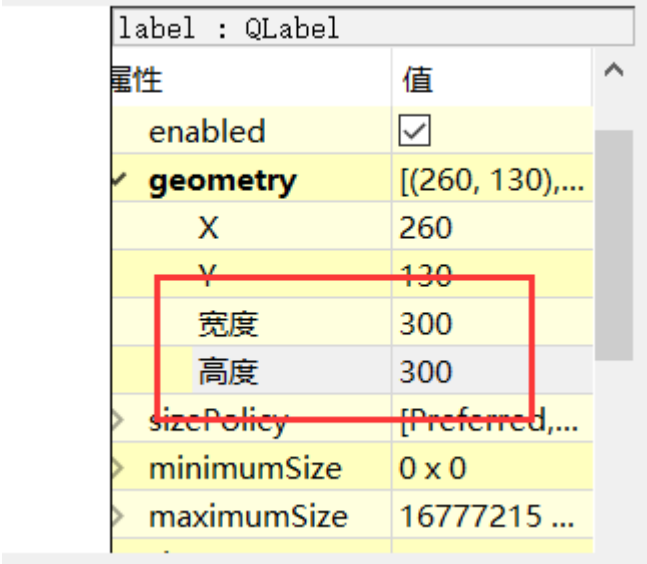
```
QString path = "D:/images/heads/";
QDir dir(path);
if(!dir.exists())
{
    qDebug() << path+"不存在!!";
    exit(-1);
}
QStringList fileList = dir.entryList(QDir::Files, QDir::Name);
for(int i = 0; i < fileList.size();i++)
{
    if(!fileList[i].endsWith(".jpg") && !fileList[i].endsWith(".png"))
        fileList.removeAt(i);
    qDebug() << fileList[i];
}
```

#练习3

制作图片浏览器


图片浏览
<<
>>


inputMethod...
ImhNone
QFrame
frameShape
Panel
frameShadow
Raised
lineWidth
1
midLineWidth
0
QLabel


label : QLabel
属性
值
enabled
geometry
[(260, 130),...
X
260
Y
130
宽度
300
高度
300
sizePolicy
[Preferred,...
minimumSize
0 x 0
maximumSize
16777215 ...

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public://下面是成员函数的声明
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
    void initFileList();//用来初始化图片名字列表
    void showImage();//显示图像


private slots://下面是槽函数的声明
    void on_pushButtonPre_clicked();
    void on_pushButtonNext_clicked();

private://下面是成员变量
    Ui::MainWindow *ui;
    QStringList fileList;//用来保存目录下所有图片文件的名称
    QString dirName;//保存被获取文件名的目录
    int index;//用来记录显示图片名字的下标
};

#endif // MAINWINDOW_H
```

mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDir>
#include <QDebug>
#include <QPainter>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow),
```

```
    dirName("D:/images/heads/"),
    index(0)
{
    ui->setupUi(this);
//    dirName = "D:/images/heads/";
//    index = 0;
    initFileList();//初始化图片名字列表
    showImage();//显示一张图片
}

MainWindow::~MainWindow()
{
    delete ui;
}
//用来初始化图片名字列表
void MainWindow::initFileList()
{
    QDir dir(dirName);
    if(!dir.exists())//如果dirName目录不存在
    {
        qDebug() << "目录文件不存在，请更换一个真实存在的路径!";
        exit(-1);//结束程序
    }
    fileList = dir.entryList(QDir::Files, QDir::Name);
    //让fileList中，只保留.png和.jpg结尾的文件名字
    for(int i = 0; i < fileList.size(); i++)
    {
        //如果不是以.jpg和.png结尾的，我们将其从fileList中删除
        if(!fileList[i].endsWith(".jpg") &&
!fileList[i].endsWith(".png"))
        {
            fileList.removeAt(i);//不是图片的名字，从列表中删除
        }
    }
}

//显示图像
void MainWindow::showImage()
{
    //pathanme = 目录名 + 图片名字
    QString pathname = dirName+fileList[index];
    QImage img;
    if(!img.load(pathname))
    {
        qDebug() << "加载图片失败，路径不正确!!";
        exit(-1);//结束程序
    }
    //缩放图片，按照label的大小进行缩放
```

```
    QImage newImg = img.scaled(ui->label->width(),ui->label->height());
    //画出缩放的图像
    QPainter painter;
    painter.drawImage(0,0,newImg);
    //将缩放后的图像显示在label中
    ui->label->setPixmap(QPixmap::fromImage(newImg));
}

//点击前一张图片，对应的槽函数
void MainWindow::on_pushButtonPre_clicked()
{
    index = index == 0 ? fileList.size()-1 : index-1;
    showImage();//显示图片
//    if(index == 0)//说明是第一张
//        index = fileList.size() - 1;//前一张是最后一张
//    else
//        index = index - 1;
}
//点击后一张图片，对应的槽函数
void MainWindow::on_pushButtonNext_clicked()
{
    index = index == fileList.size()-1 ? 0 : index+1;
    showImage();//显示图片
}
```

7.5 修改pushButton显示

```
ui->pushButtonTurn->setText("LED开");//设置pushButton上的文件
ui->pushButtonTurn->setStyleSheet("color:blue");//设置pushButton按钮 字体颜色
ui->pushButtonTurn->setStyleSheet("color: rgb(0,0,255);");//设置pushButton按钮 字体颜色
ui->pushButtonTurn->setStyleSheet("background-color: rgb(0,0,255);");//设置按钮 背景色
//设置按钮的背景图片
ui->pushButtonTurn->setStyleSheet("border-image: url(:/images/heads/bulb_off.png);");
//同时设置按钮的文字颜色及背景图片
ui->pushButtonTurn->setStyleSheet("color: blue; border-image: url(:/images/heads/bulb_off.png);");
```

```
//直接设置文字颜色
//    ui->pushButtonTextColor->setStyleSheet("color: green");//设置为绿色
//注意rgb的值，在0-255之间
ui->pushButtonTextColor->setStyleSheet("color: rgb(255,0,0);");//可以通过rgb设置自己想要的颜色
//同时设置字体颜色和背景颜色
ui->pushButtonBgColor->setStyleSheet("color: rgb(0,0,255); background-color: rgb(34,234,123);");//通过rgb设置背景颜色
//同时设置字体颜色和背景图片
ui->pushButtonBgImage->setStyleSheet("color: rgb(0,255,0); border-image: url(D:/images/heads/head1.jpg);");//设置背景图片
```

#练习4

```
bulb_off.png
bulb_on.png
默认pushButton按钮上显示的关灯背景图
第一次点击  显示开灯背景图
第二次点击  显示关灯背景图
第三次点击  显示开灯背景图
第四次点击  显示关灯背景图
..
..
如此循环往复
```

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private slots:
    void on_pushButtonLED_clicked();
```

```
private:
    Ui::MainWindow *ui;
    bool flag;//用来当做控制灯开关的标志位
};

#endif // MAINWINDOW_H
```

mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow),
    flag(true)
{
    ui->setupUi(this);
    //直接设置文字颜色
    //    ui->pushButtonTextColor->setStyleSheet("color: green");//设置为绿色
    //注意rgb的值，在0-255之间
    ui->pushButtonTextColor->setStyleSheet("color: rgb(255,0,0);");//可以通过rgb设置自己想要的颜色
    //同时设置字体颜色和背景颜色
    ui->pushButtonBgColor->setStyleSheet("color: rgb(0,0,255); background-color: rgb(34,234,123);");//通过rgb设置背景颜色
    //同时设置字体颜色和背景图片
    ui->pushButtonBgImage->setStyleSheet("color: rgb(0,255,0); border-image: url(D:/images/heads/head1.jpg);");//设置背景图片

    ui->pushButtonLED->setStyleSheet("border-image: url(D:/images/heads/bulb_off.png);");//设置背景图片
}

MainWindow::~MainWindow()
{
    delete ui;
}
//当点击LED开关 对应的 槽函数
void MainWindow::on_pushButtonLED_clicked()
{
    if(flag)
    {    //切换为开灯图片
        ui->pushButtonLED->setStyleSheet("border-image: url(D:/images/heads/bulb_on.png);");
        //给STM32发送一个开LED的命令
```

```
    }
    else
    {
        //切换为关灯图片
        ui->pushButtonLED->setStyleSheet("border-image:
url(D:/images/heads/bulb_off.png);");
        //给STM32发送一个关LED的命令
    }
    //每点击一次，都要切换一次标志位
    flag = !flag;//flag实现 true -> false  false -> true 如此循环往复
}
```

8. QTimer定时器

```
QTimer* timer = new QTimer; //创建一个定时器对象
timer->start(1000); //启动定时器 1000ms == 1s 每隔1s产生一个timeout信号
timer->stop(); //停止定时器
```

9. LCDNumber控件

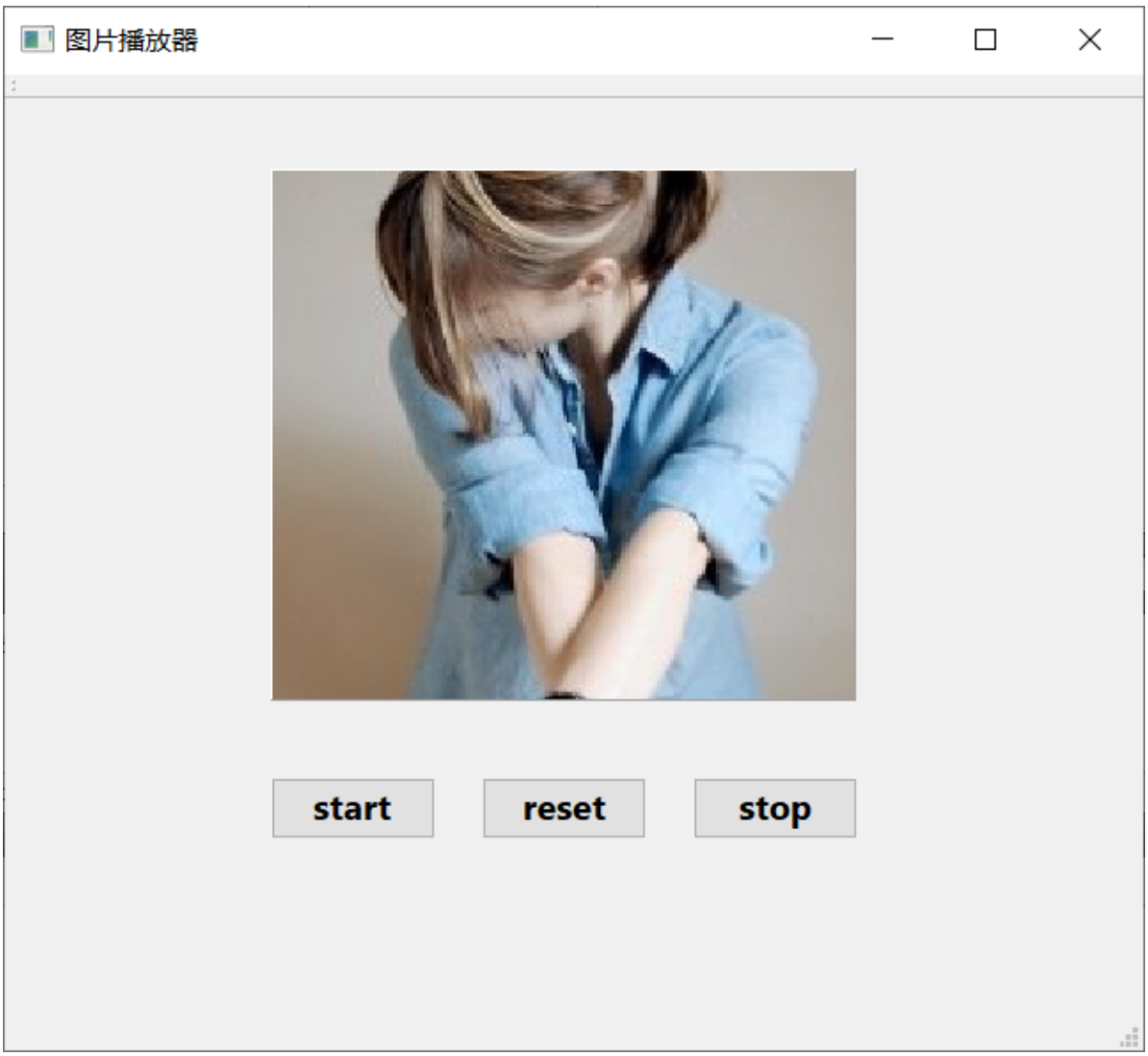
```
int count = 10;
ui->timeLCD->display(count);
QString time = "2022-12-19 22:22:22";
ui->lcdNumber->display(time);

QPalette textPalette;//创建一个调色板对象
//设置调色板颜色
textPalette.setColor(QPalette::Normal, QPalette::WindowText, Qt::red);
//设置LCDNumebr字体颜色
ui->timeLCD->setPalette(textPalette);
//获取当前的系统时间
QString time = QDateTime::currentDateTime().toString("yyyy-MM-dd
hh:mm:ss");
ui->lcdNumber->display(time);
```

#练习5

启动定时器，实现LCDNumber控件，每隔1s更新一次系统时间

#练习6

利用QTimer定时器，制作一个自动播放的图片相册



10. progressBar控件

```
int num = 10;
ui->progressBar->setValue(num);
```