前言

受到大佬的影响，决定未来多多记录自己的学习过程。本篇博客将把 Qt 开发数据库课程设计的过程详细介绍，适合初学者。可能因为太过详细所以请利用目录。

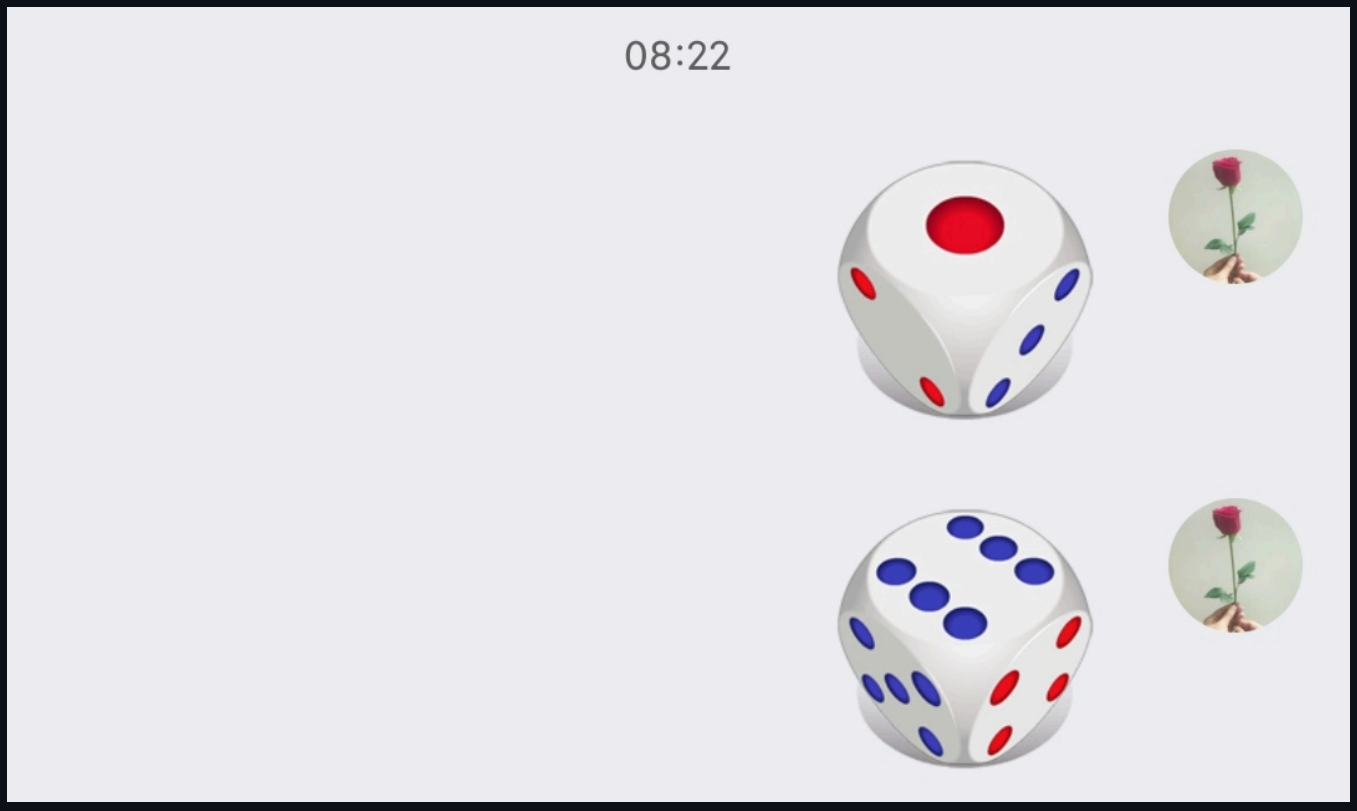
参考资料

如何在几个QT界面之间建立连接

QT5：一个UI实现多个界面切换(基础1)

选题

摇了个1，和 RongLin02 的选题一样。我当场重新摇，很快啊。（抄他的代码极易雷同！望周知！）



题目 员工培训管理系统

1 系统需求分析

1.1 系统功能分析

员工培训系统需要实现的主要功能包括：

- 企业总体培训**课程**的**设置和安排**。
- 允许**员工**根据自己的情况**选择**合适的**课程**和上课**时间**。
- 对**选课结果**进行**统计报表**。允许**员工对**最后选课**结果的查询**。
- 培训**考核成绩**的**输入和查询**。
- 员工**培训效果**的**综合报表**。
- 员工**个人信息**的**修改**。

1.2 系统功能模块设计（重点，结构）

本系统涉及到员工和培训管理部门之间的交流， 因此需要根据用户的不同分成两大功能模块。这两个模块的功能和使用的权限完全不同。本系统功能模块如图 1 所示。

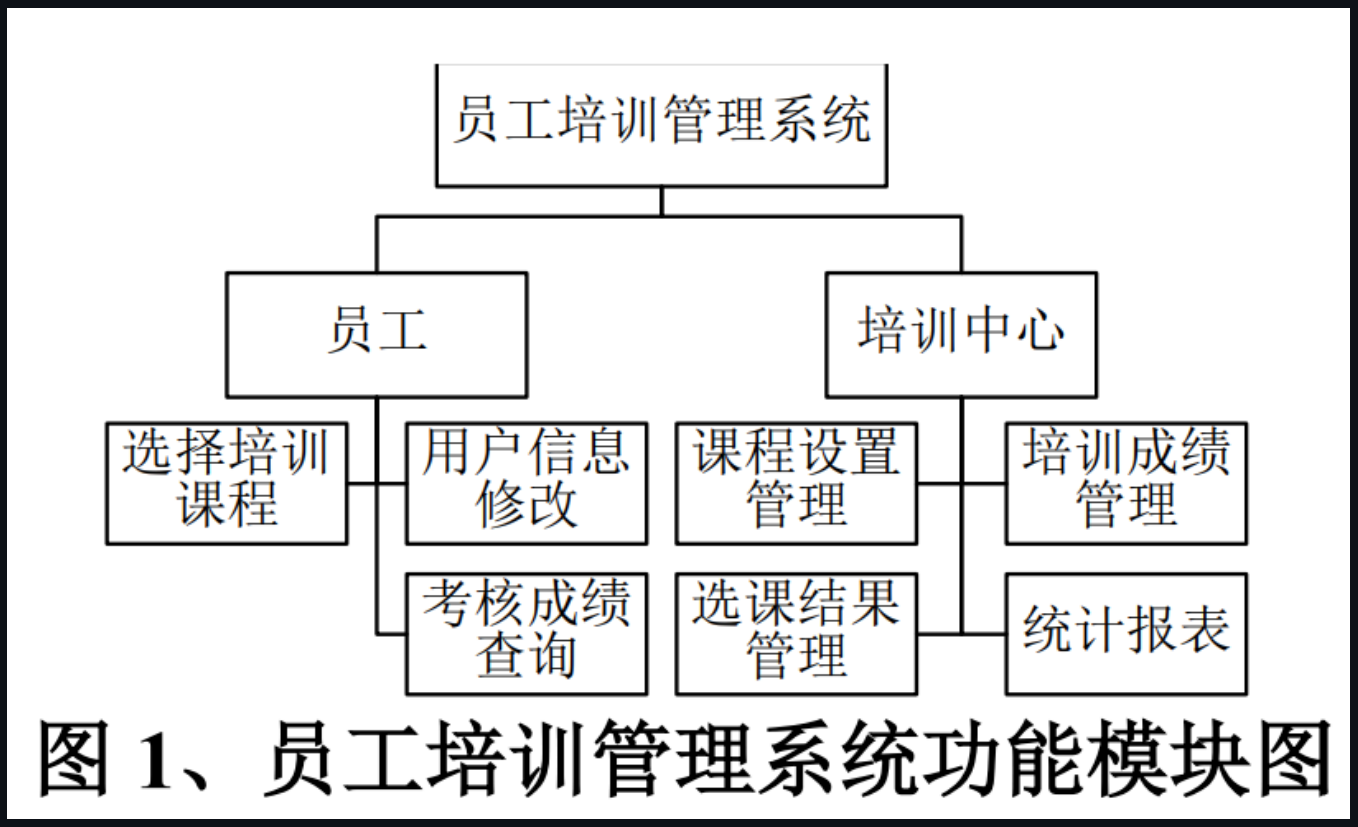


图 1、员工培训管理系统功能模块图

1.3 与其它系统的关系（无效信息）

员工培训系统可以为员工素质技能的评价提供可靠的依据，是职务评定的一个参考信息源。系统本身需要用到人事管理系统中的**员工基本信息和部门信息等辅助资料**，这些数据可以通过数据库直接读取。

1.4 数据流程图

员工培训管理系统的数据流程如图 2 所示。

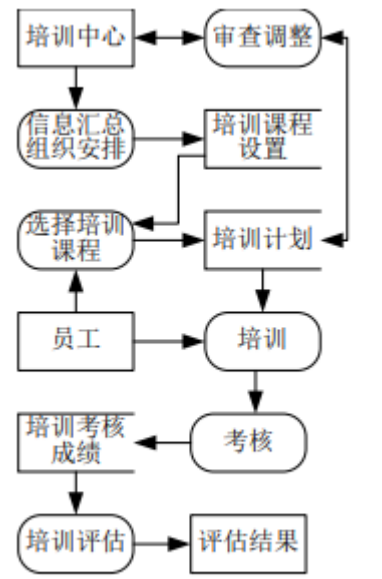


图 2、培训管理数据流程图

2 数据库设计

2.1 数据库需求分析

根据系统数据流程图，我们可以列出以下系统所需的数据项和数据结构：

- 课程设置：编号、名称、简介、所用教材、上课地点、人数、上课时间
- 选课结果：记录编号、员工、课程、考核成绩、评价、考核日期。

所需的外部数据支持：

- 人员信息：员工号、密码、权限、姓名、部门、当前状态等。
- 部门设置：部门编号、名称等。

2.2 数据库概念结构设计

图 3 是本系统所需数据的 E-R 模型图。

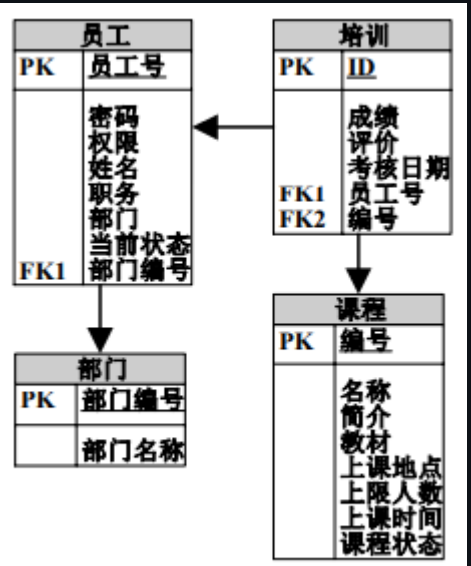


图 3、培训管理 E-R 图

2.3 数据库逻辑结构设计

根据 E-R 图和数据库需求分析，培训管理系统需要创建 2 个主要的数据表：课程设置表和培训安排表。对应这两个表中的个别代码字段，又需要创建 2 个代码表：课程状态代码表和考核评价代码表。这 4 个数据表的结构如表 1 至表 4 所示。员工信息和部门信息作为外部数据支持可以使用人事管理系统中建立的数据表，如表 5 和表 6 所示。

表 1　COURSE 课程设置表

字段名	数据类型	是否可空	说明
ID			课程编号
NAME			课程名
TEACHER			任课教师（外部关键字 PERSON）
INTRO			课程简介
BOOK			所用教材
CLASSROOM			上课地点
NUMBER			课程上限人数
CLASSTIME			开课时间
STATE			状态（外部关键字 COURSE_STATE）

表 2　TRAINING_PLAN 培训安排表

字段名	数据类型	是否可空	说明
ID			编号
PERSON			员工（外部关键字 PERSON）
COURSE			课程
SCORE			成绩
APPRAISEMENT			评价（外部关键字 APPRAISEMENT）
EXAM_DATE			考核日期

表3　COURSE_STATE 课程状态代码表

字段名	数据类型	是否可空	说明
CODE			状态代码
DESCRIPTION			描述

表4　APPRISEMENT 考核评价代码表

字段名	数据类型	是否可空	说明
CODE			评价代码
DESCRIPTION			描述

表5　PERSON 员工个人信息表

字段名	数据类型	是否可空	说明
ID			员工号（主关键字）
PASSWD			密码
AUTHORITY			用户权限
NAME			姓名
SEX			性别
BIRTHDAY			生日
DEPARTMENT			所在部门
JOB			职务
EDU_LEVEL			受教育程度
SPECIATY			专业技能
ADDRESS			家庭住址
TEL			联系电话
EMAIL			电子信箱
STATE			当前状态（T-员工，F-非员工）
REMARK			备注

表6　DEPARTMENT 部门信息表

字段名	数据类型	是否可空	说明
ID			部门编号
NAME			部门名称
MANAGER			部门经理
INTRO			简介

2.4 数据库的建立

2.4.1 数据库的建立（请设计者完成）

2.4.2 初始数据的输入

本系统中，初始数据包括课程状态代码和评价代码，如表 7 至表 8 所示。

表 7　课程状态代码

代码	说明
0	选课中
1	进行中
2	已结束

表 8　考核评价代码

代码	说明
0	未考核
1	不及格
2	及格
3	良好
4	优秀

3 各功能模块的设计与实现

3.1 功能说明

本管理系统主要分为两大部分：**培训管理应用程序**和**学员选课应用程序**。培训管理应用 程序主要用于培训中心的管理人员对培训课程和培训情况进行维护。此应用程序主要包括四项功能：课程设置、选课结果查询修改、成绩输入、培训成绩统计报表。另外，系统需要有登录窗口(用于权限认证)和导航窗口(用于连接各项功能)。学员选课应用程序包括个人信息修改、选课和成绩查询三项功能。

1. 培训管理管理应用程序功能说明
2. 学员选课应用程序功能说明

3.2 用户界面设计

完成数据库创建和功能说明以后，我们可以进行下一步工作，即设计用户界面。

1. 培训管理应用程序登录窗体的创建
2. 培训管理应用程序主窗体的创建

3. 课程设置窗体的创建
4. 选课结果查询窗体的创建
5. 学员名单报表窗体的创建
6. 考核评定结果窗体的创建
7. 培训统计窗体的创建
8. 培训成绩报表窗体的创建
9. 学员选课客户端界面的创建

3.3 各功能模块的实现

1. 培训管理应用程序数据模块的创建
2. 培训管理应用程序登录程序的实现
3. 课程设置模块的实现
4. 选课结果查询的实现
5. 学员名单报表的实现
6. 考核评定结果的实现
7. 培训统计的实现
8. 培训成绩报表的实现
9. 学员选课客户端应用程序的创建

初步设计

此为本系统的结构，从这里入手。

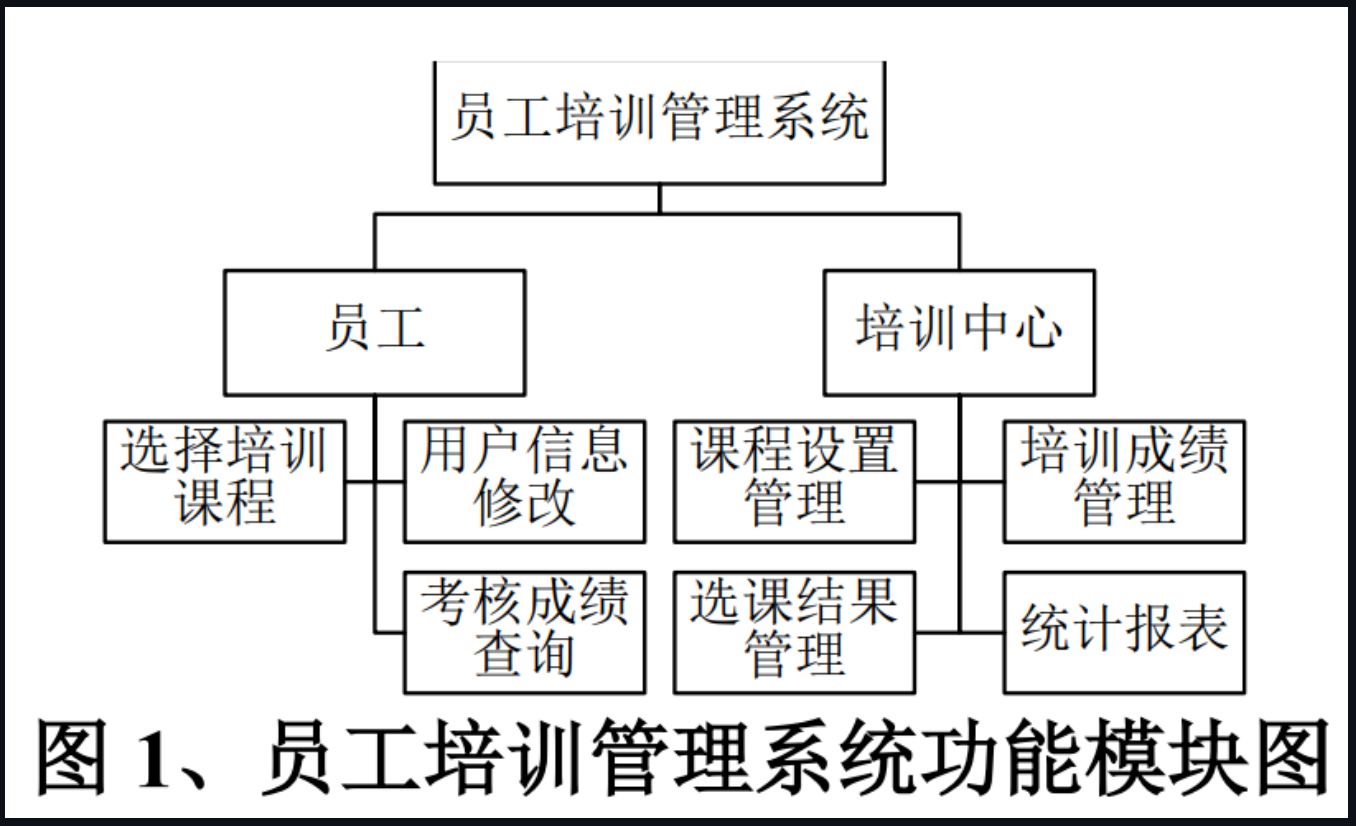


图 1、员工培训管理系统功能模块图

进入系统，首先登录。

登录系统使用数据库完成。对每个账号，设置“管理员权限”或“普通用户权限”。

如果是管理员则进入培训中心界面，用户则进入员工界面。

培训中心：

1. 添加课程

   课程设置：编号、名称、简介、所用教材、上课地点、人数、上课时间

   编号为主键，设置所有课程的表。排序按后创建在前。

2. 选课结果的查看

   选课结果：记录编号、员工、课程、考核成绩、评价、考核日期。

   记录编号，员工，课程编号为主键，表格界面添加成绩。

3. 成绩管理

   添加成绩。

4. 统计分数

做平均分，一些分析等等。

用户：

1. 选课
2. 修改信息

   人员信息：员工号、密码、权限、姓名、部门、当前状态等。

   部门设置：部门编号、名称等。
3. 成绩查询

准备工作

QT

下载链接

触发调用机制

连接函数

`connect(参数1,参数2,参数3,参数4)`

1. 参数1 信号的发送者，常常是按钮对象的指针
2. 参数2 发送的信号（函数地址），常常是点击
3. 参数3 信号的接受者，指针。
4. 参数4 处理的槽函数 （函数地址）

其 实质 应该是 监听。`参数3` 监听 `参数1` 的 `参数2` 信号，监听到了，就触发 `参数3` 的 `参数4` 方法。

信号

```
//点击
void clicked(bool checked = false)
//按下
void pressed()
//弹起
void released()
//状态变化时触发
void toggled(bool checked)
```

自定义信号

写到类定义中`signals:`下.

只需要声明不需要实现。

返回值为`void`。

可以有参数，同时也可以重载。

自定义槽函数

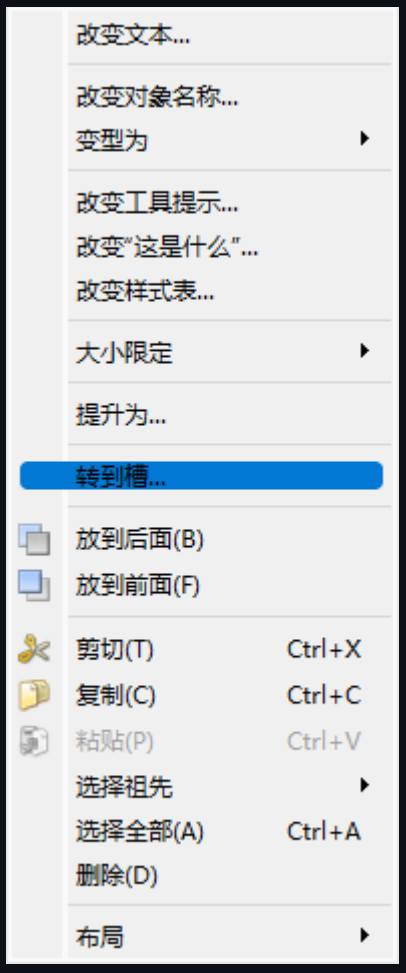
写到类定义的`public( slots):`下。

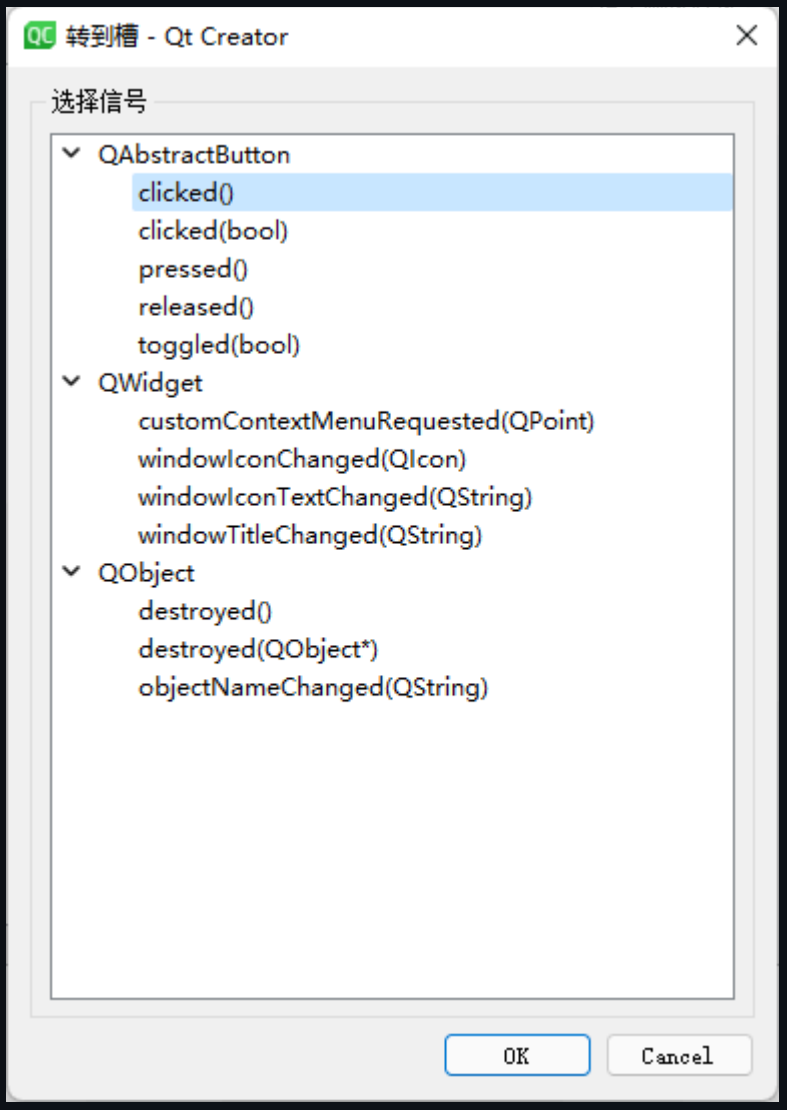
需要声明和实现。

返回`void`。

可以有参数，同时也可以重载。

可以在ui界面右键按钮，转到槽。


改变文本...
改变对象名称...
变型为
改变工具提示...
改变"这是什么"...
改变样式表...
大小限定
提升为...
转到槽...
放到后面(B)
放到前面(F)
剪切(T) Ctrl+X
复制(C) Ctrl+C
粘贴(P) Ctrl+V
选择祖先
选择全部(A) Ctrl+A
删除(D)
布局



选clicked()就好。

函数指针

函数指针的定义

```
1  //指向函数的返回类型 (*指针名)(指向函数的参数表)=指向函数名
2  //例如
3  int (*int_operator)(int, int) = int_add;
```

如果指向类里的函数

```
1  //指向函数的返回类型 (类名:: *指针名)(指向函数的参数表)=指向函数名
2  //例如
3  void (Teacher:: *Signal)(void)=&
```

注意

当自定义信号和槽函数有多个重载的时候，需要利用函数指针，明确指向函数的地址。

断开连接

如果需要断开连接的话，需要用到`disconnect`函数，参数复制`connect`即可。

pro文件介绍

```
QT       += core gui   //Qt包含的模块

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets  //大于4版本以上 包含widget模块

TARGET = MyTest  //目标   生成的.exe程序的名称
TEMPLATE = app          //模板   应用程序模板  Application  除了这个还有很多


SOURCES += \            //源文件
        main.cpp \
        mainwindow.cpp

HEADERS += \            //头文件
        mainwindow.h

FORMS += \              //ui文件界面
        mainwindow.ui
```

不需要主动管理内存，析构等。

Main 函数

```
1  QApplication a;       //应用程序对象。有且仅有一个应用程序对象。
2  myWidget w;           //窗口默认不弹出，调用 show() 方法。
3  w.show();
4  return a.exec();      //阻塞，防止窗口消失。
```

Q_OBJECT

类定义的第一行加上`Q_OBJECT`，不需要分号。

`Q_OBJECT`是Qt中的一个宏。由于Qt的语法是在c++的基础上拓展的，所以在Qt程序的编译过程中，直接用gcc这些标准编译器编译是不可行的，因为gcc不能识别这些拓展性的语法，比如信号和槽（Signal and Slot）。于是Qt引入了moc这一编译器。

`moc（Meta-Object Compiler）`，即元对象编译器，Qt 程序在交由标准编译器编译之前，会使用 moc 分析 C++ 源文件，假设它发现某个头文件中包括了 Q_OBJECT 这个宏（注意， moc 只处理头文件中标记了Q_OBJECT的类声明，不会处理 cpp 文件中的类似声明），则会生成另外一个 C++ 源文件，这个源文件里包括了 Q_OBJECT 宏的实现代码，并且文件名称将会是原文件名称前面加上 moc_ 。这个新的文件会和原本的c++源文件一起进入编译系统，最终被链接到二进制代码中完成编译工作。

在Qt中，QObject是所有Qt类的基类，是Qt对象模型的核心，只有继承了QObject类的类，才具有信号槽的能力。所以，为了使用信号槽，必须继承QObject。凡是QObject类（不管是直接子类还是间接子类），都应该在第一行代码写上Q_OBJECT。不管是不是使用信号槽，都应该添加这个宏。这个宏的展开将为我们的类提供信号槽机制、国际化机制以及 Qt 提供的不基于 C++ RTTI 的反射能力。因此，如果你觉得你的类不需要使用信号槽，就不添加这个宏，这是错误的，因为其它很多操作都会依赖于这个宏。

反正拥有Qt开发区别于C++的地方，就需要加上这行。

按钮

pro文件引入：

```
QT += widgets
```

```
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets  //大于4版本以上 包含 widget 模块
```

包含：

```
#include <QPushButton>
```

设置依赖，否则单独弹出一个窗口只有按键。

```
QPushButton *btn = new QPushButton;
btn -> setParent(this);        //设置父类、依赖
btn -> setText();          //显示文本
```

或

```
QPushButton *btn = new QPushButton("显示文本",this);
```

重置窗口/按钮大小

```
this->resize(int kuan,int gao);
```

移动按钮

```
btn->move(int x,int y);
```

窗口

```
setWindowTitle();                 //设置标题
resize(int kuan,int gao);          //设置大小
setFixedSize(int kuan,int gao);    //设置固定大小
```

菜单栏

菜单栏最多只有一个

```
QMenuBar * bar = MenuBar();
this->setMenuBar( bar )
QMenu * fileMenu = bar -> addMenu(“文件”)  //创建菜单
QAction * newAction =  fileMenu ->addAction(“新建”); //创建菜单项
fileMenu->addSeparator();      //添加分隔线
```

分割线示例：

工具栏

可以有多个

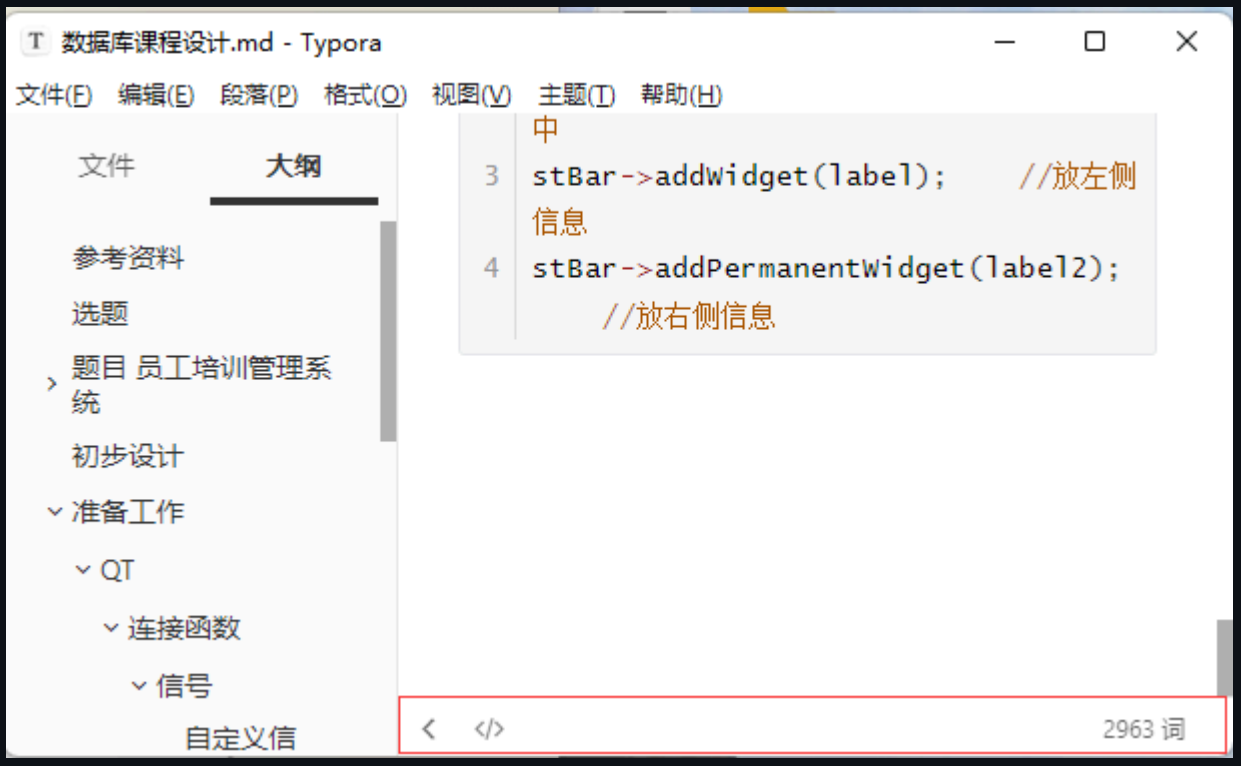
```
QToolBar * toolbar = new QToolBar(this);
addToolBar( 默认停靠区域， toolbar ); //例如:Qt::LeftToolBarArea
```

可以设置后期停靠区域，设置浮动，设置移动，添加菜单项 或者添加 小控件等。

状态栏

最多一个

```
QStatusBar * stBar = statusBar();
setStatusBar(stBar);    //设置到窗口中
stBar->addWidget(label);    //放左侧信息
stBar->addPermanentWidget(label2);    //放右侧信息
```



部件

浮动窗口 可以多个

类名：QDockWidget

方法：addDockWidget(默认停靠区域，浮动窗口指针)

设置后期停靠区域

设置核心部件

只能一个。

```
this->setCentralWidget(edit);
```

但可以多次设置。

模态对话框

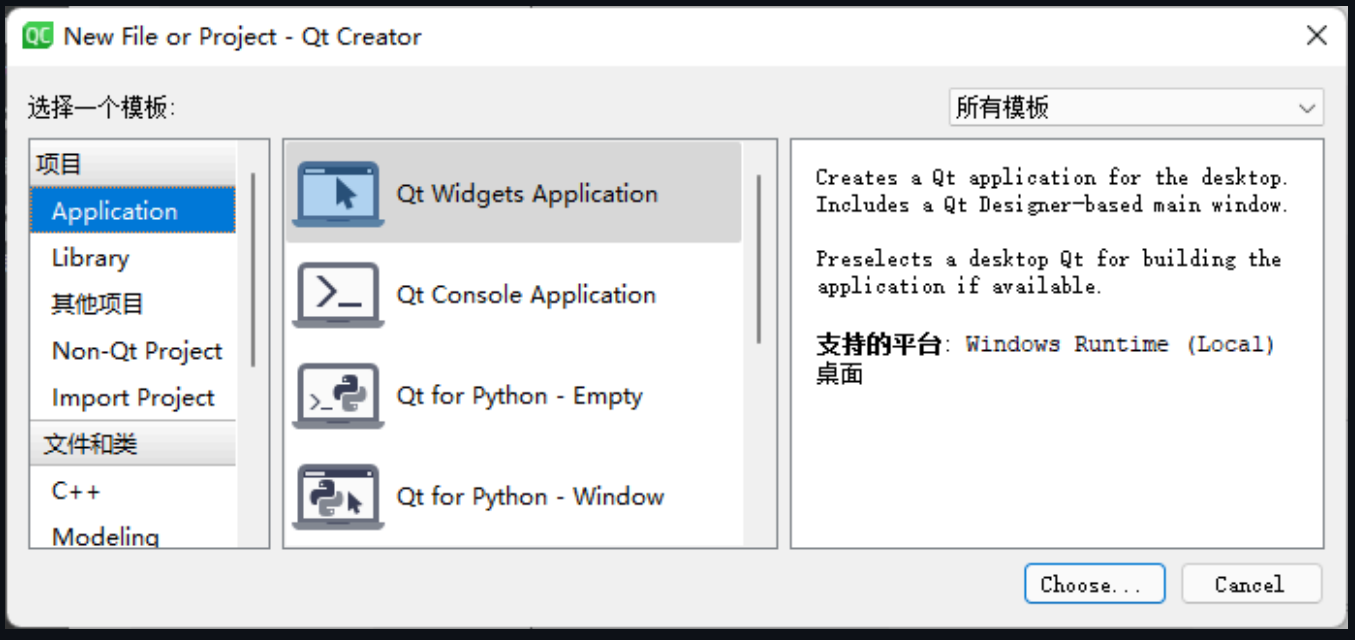
出现时不能操作其它窗口。

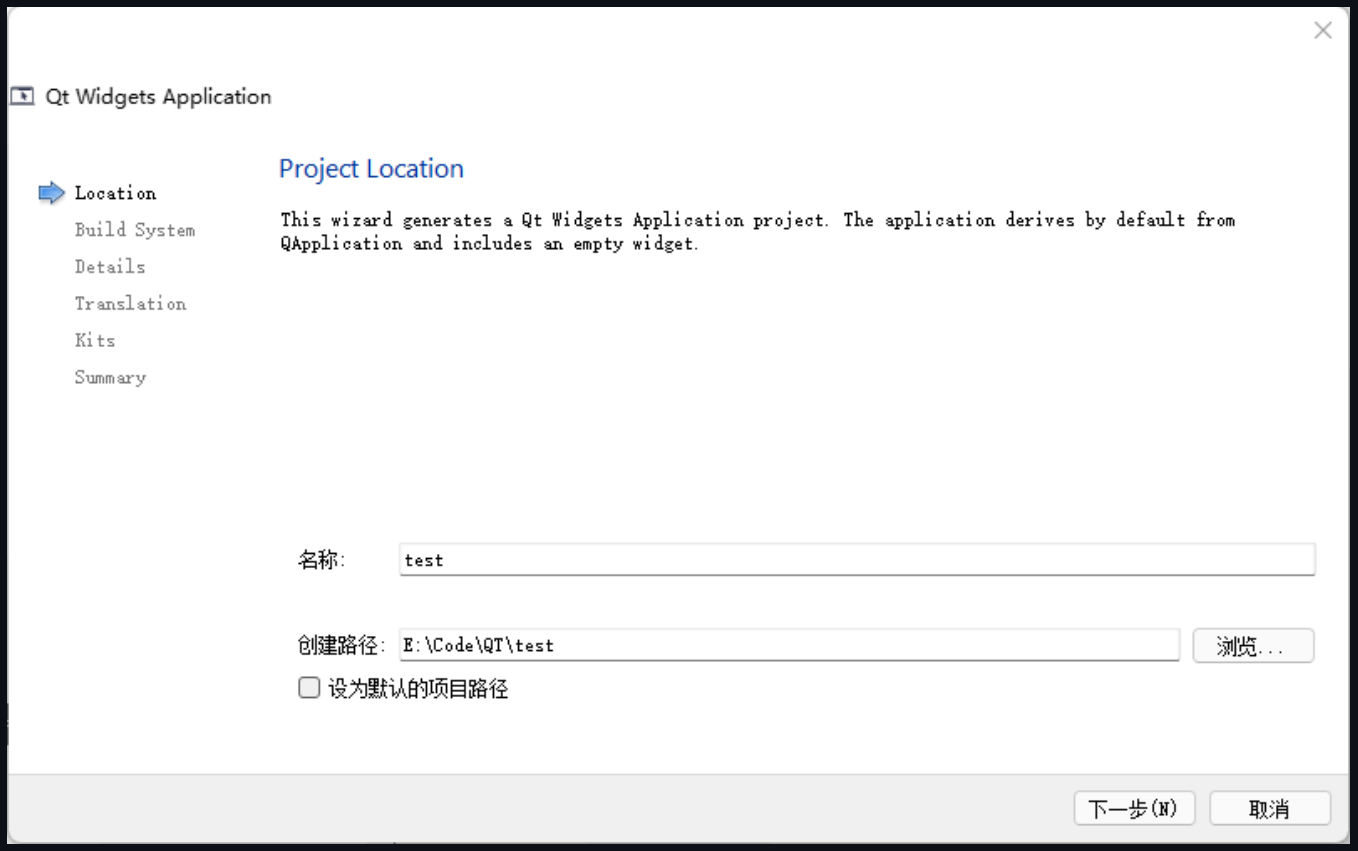
非模态对话框

可以对其他窗口进行操作。

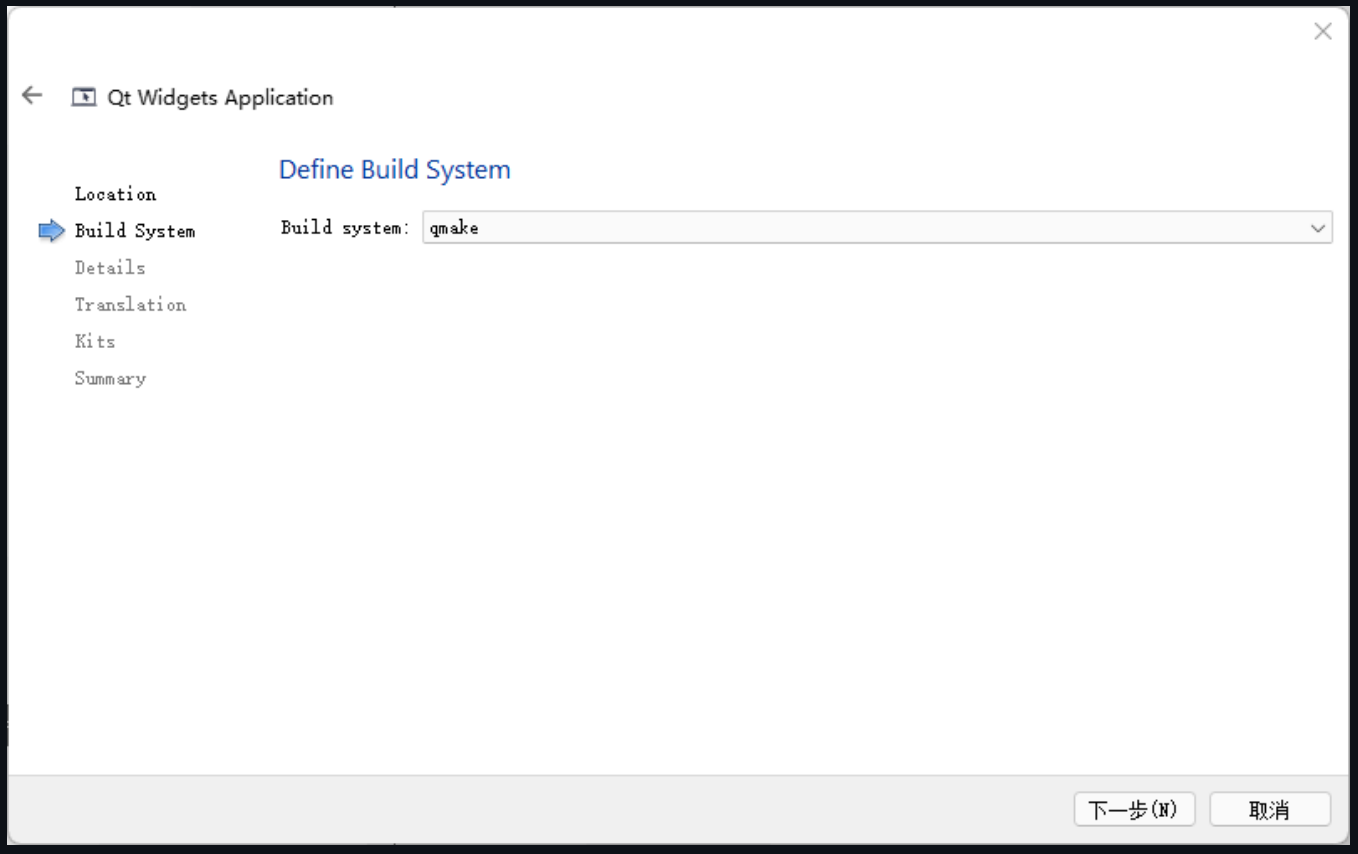
开始设计

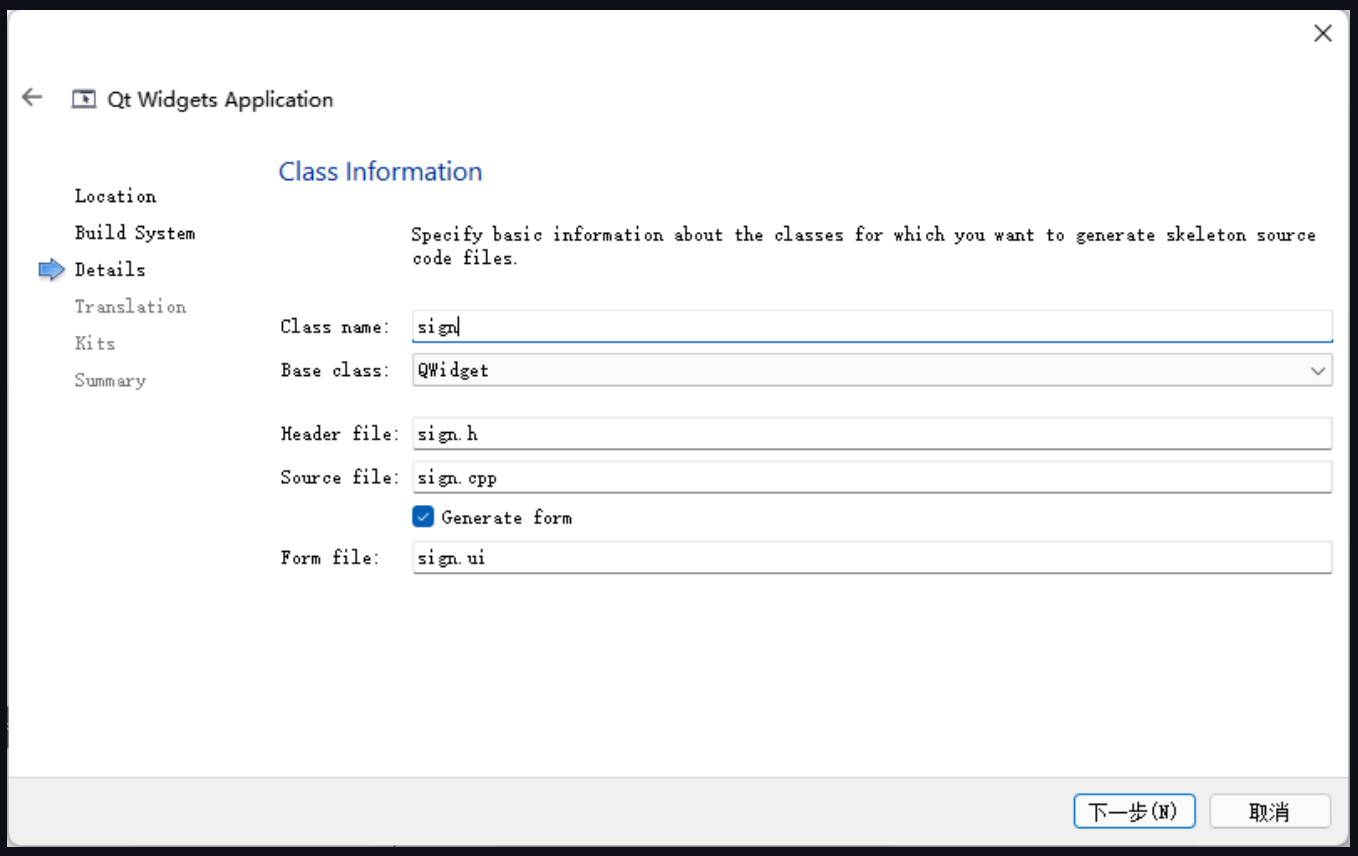
建立工程





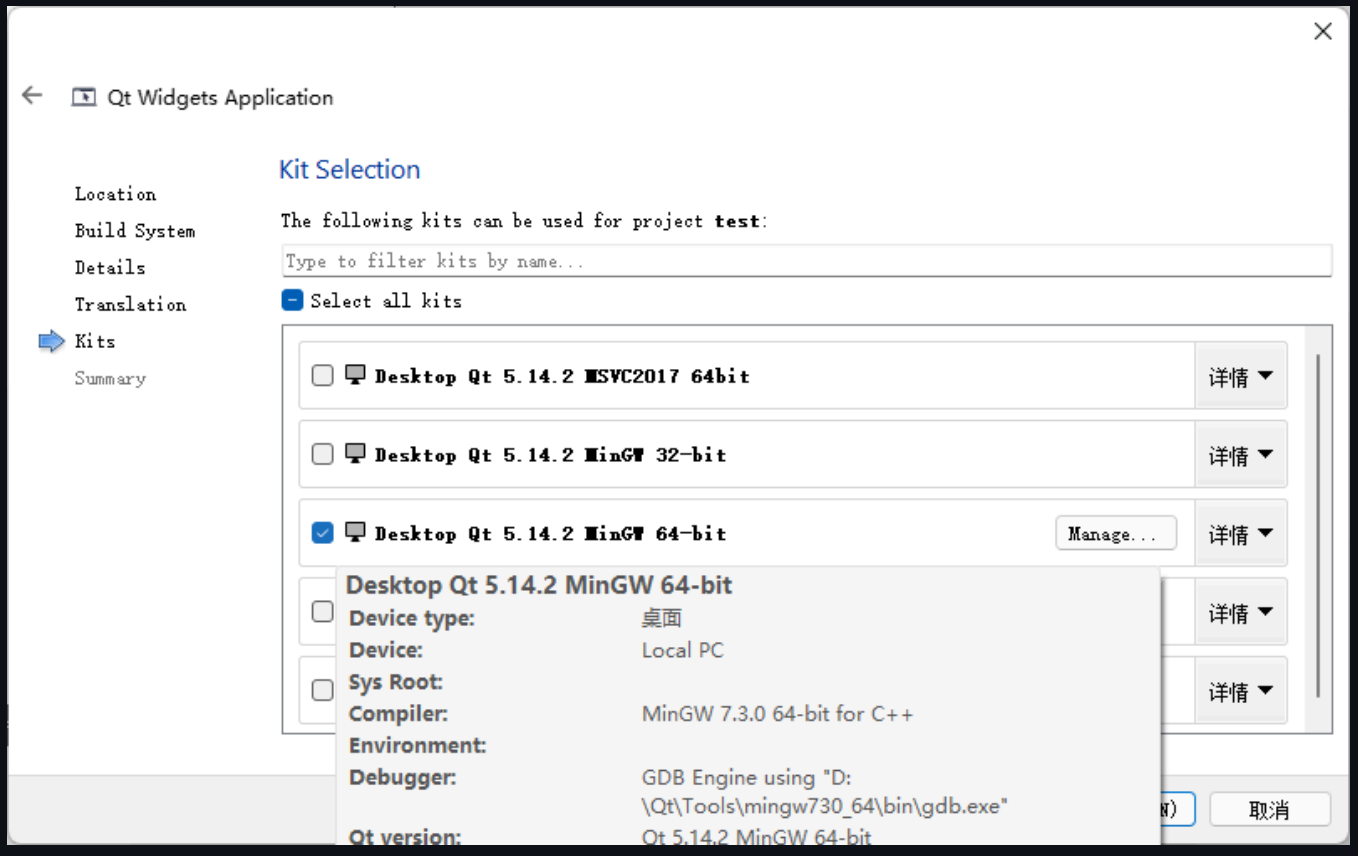
路径自选，请将test改成ETMSCPP。





这一步我们创建了一个类，继承了QWidget类，这里的Base class展开后有好几种窗口，读者可以自行搜索异同。

下一步

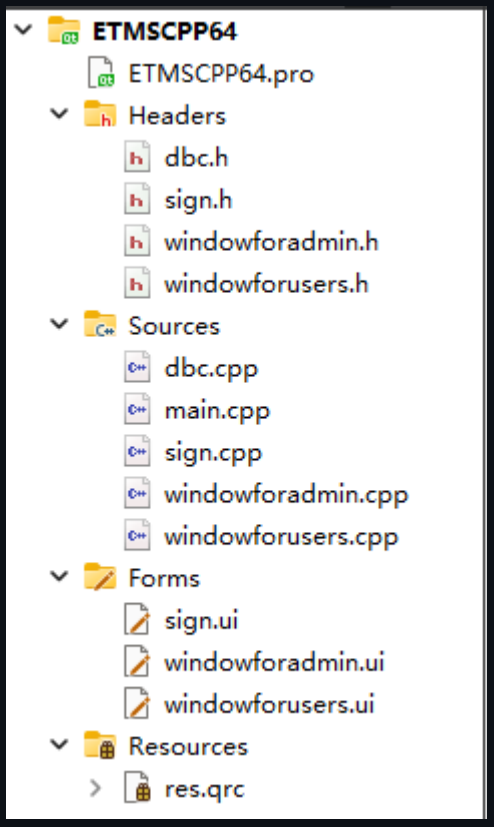


这里务必选择MinGW，选择几位？

查一下自己的数据库是多少位，例如我的MySQL是64位，因此我选择如图。选择好以后`下一步`直到`完成`就好。

创建完工程以后，会有`pro文件`，`sign.h/.cpp/.ui`，`main.cpp`。

项目结构展示

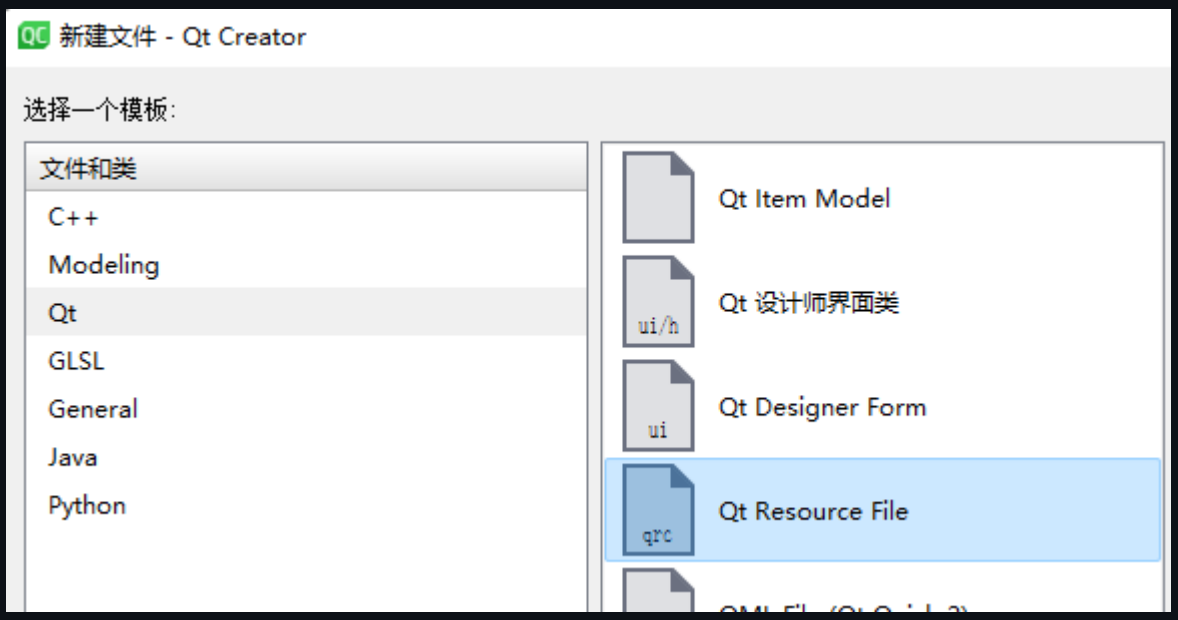


资源导入

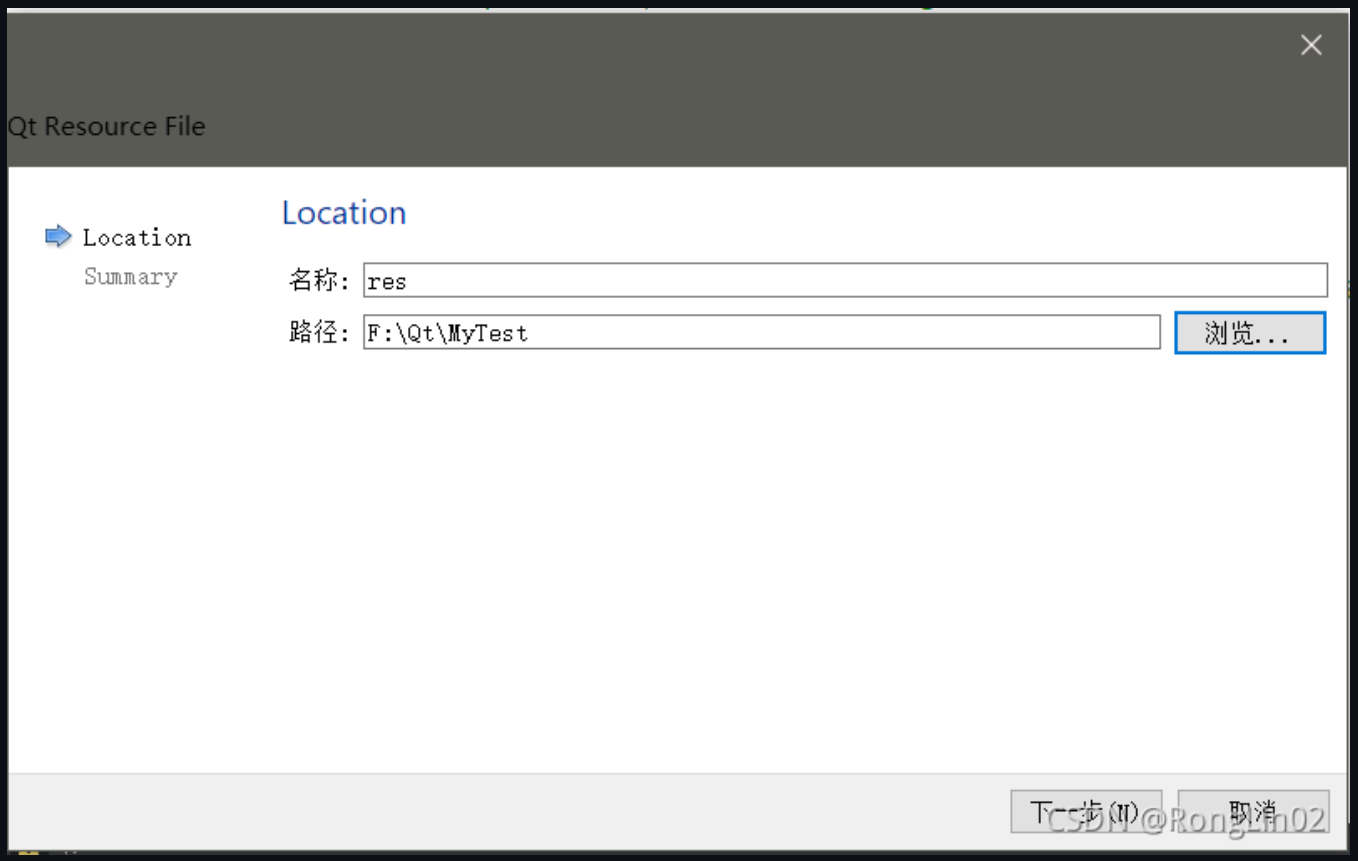
参考资料：Qt学习之Qt基础入门(中)

首先要将图片文件拷贝到项目位置下，我这里在项目根目录创建一个image文件夹，里边放入了一张图片

然后在开发工具中，右键项目->添加新文件 -> `Qt` - > `Qt recourse File`



然后起名字，叫res，然后路径也选择项目根目录



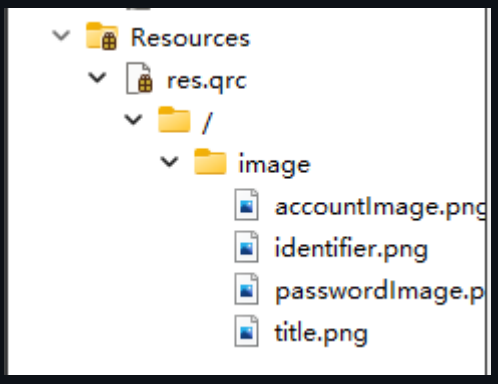
然后它会生成一个res.qrc的资源文件

然后对着`res.qrc`右键 -> `open in editor` -> 编辑资源

在右侧，添加前缀 我这里前缀起名`/`

前缀添加完毕后，再点击 添加文件，然后选择要添加的图片等资源文件，可以多选，然后再 右键项目 -> 重新构建

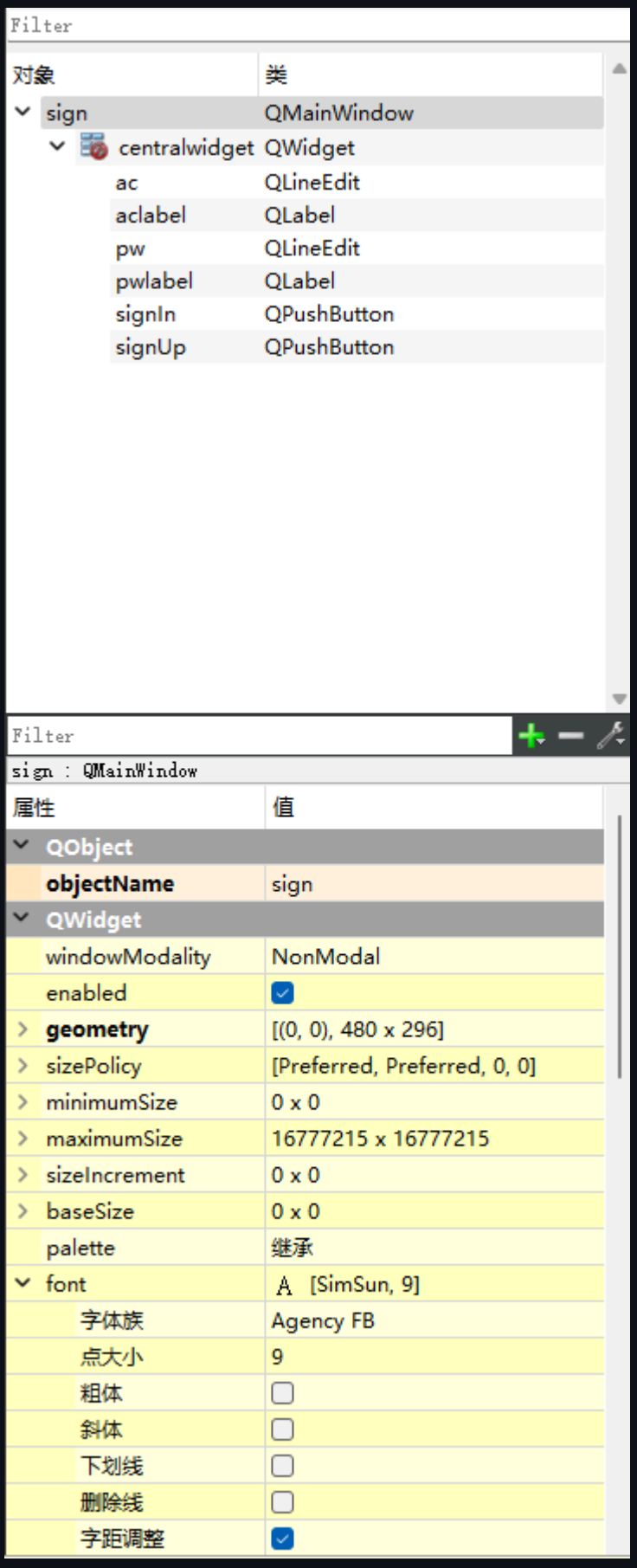
之后如果工程中出现图片，则说明导入完成。



请参照dbc类的编写，安装好ODBC，并建立数据库，再继续阅读。dbc类的代码可以不用现在深究。

sign类的设计（登录窗口）

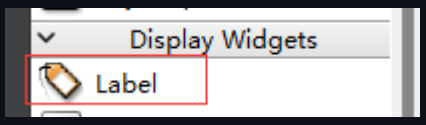
sign ui 设计如下

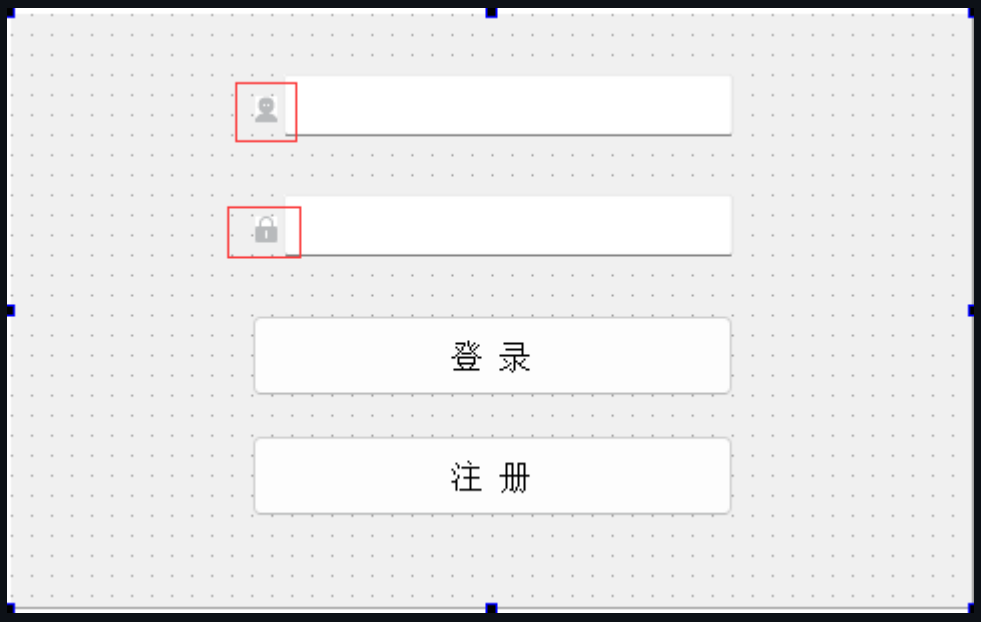


代码中设置窗口大小，在`sign.cpp`中构造函数内增加如下代码

```
1    setFixedSize(480,296);//黄金分割
2    setPalette(QPalette(Qt::white));    //设置窗口为白色背景
```

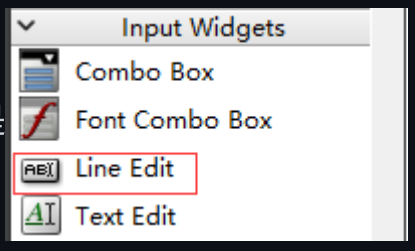
在窗口中，拖出一个 label 到某个位置以显示下面两个图标（位置自行把握）





所以，这里可以知道，一个图标的显示，是使用 label 的，其实，文字显示也是 label 。

label后面有输入框，输入框是



输入框的边框要设置成，没有上左右边框，有下边框，

我们在 sign.cpp 中构造函数增加

```
1    ui->ac->setStyleSheet("border-top:0px  solid;border-bottom:1px  solid;border-
     left:0px  solid;border-right: 0px  solid;");
2    ui->pw->setStyleSheet("border-top:0px  solid;border-bottom:1px
     solid;border-left:0px  solid;border-right: 0px  solid;");
```

两种方法设置 label 显示图片

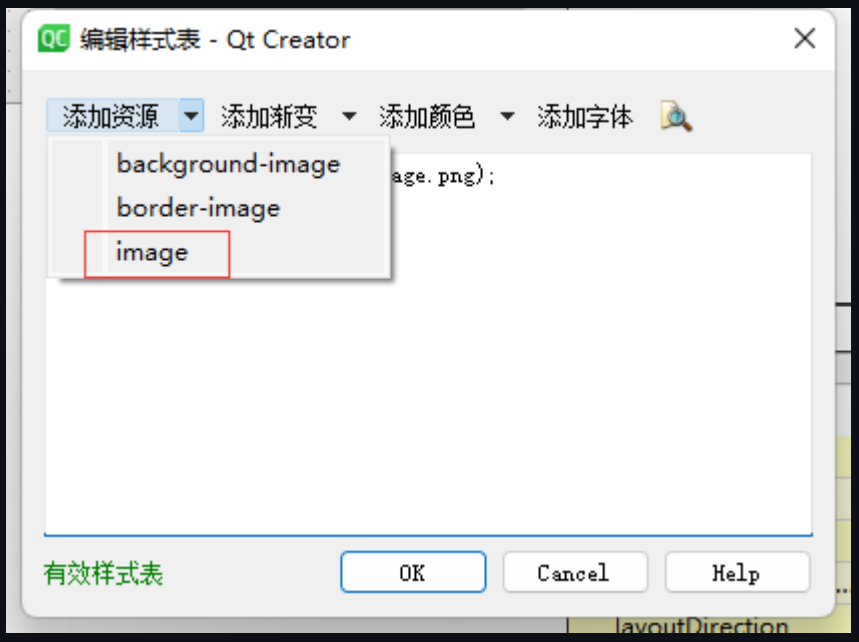
1. 代码设置

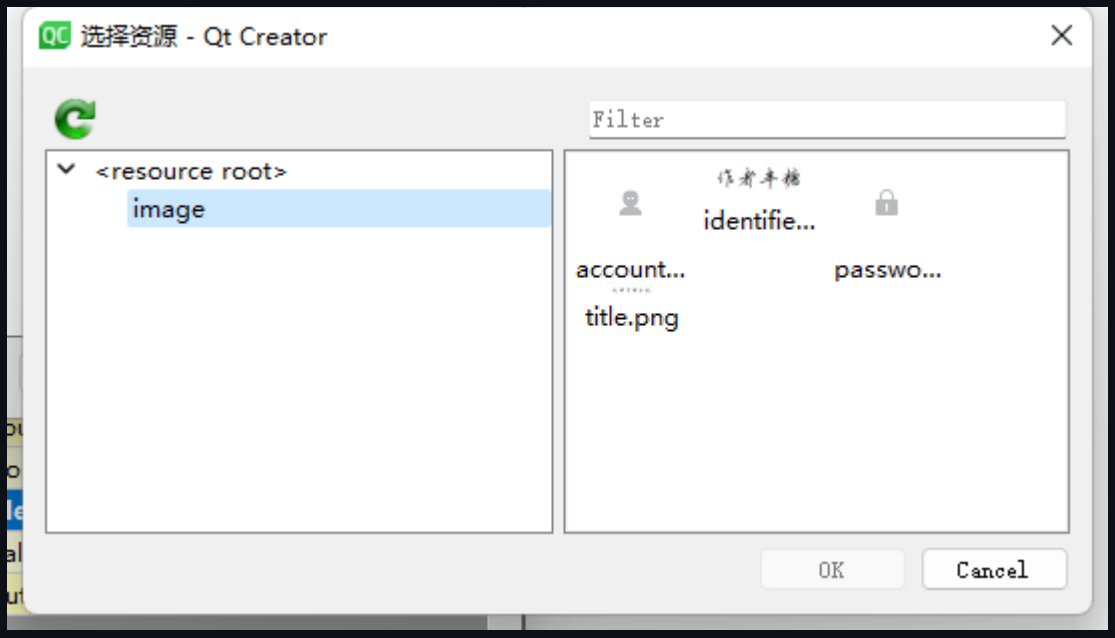
```
1  //设置图标显示,这里使用了ui设计中stylesheet来显示，因此注释掉。实际上代码中使用也可以
2      //    ui->aclabel->setPixmap(QPixmap(":/image/accountImage.png"));
3      //    ui->pwlabel->setPixmap(QPixmap(":/image/passwordImage.png"));
```

   我一开始采用这种方法，后来使用了方法2，这两行代码放到构造函数中去掉注释即可。

2. ui 设置

   点击 ui 中的 label ，右边会有其设置，找到`stylesheet`，





选择相应的图片即可。

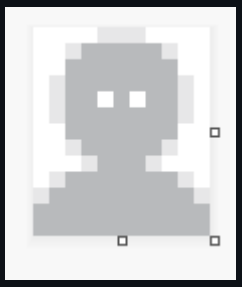
图片的设计

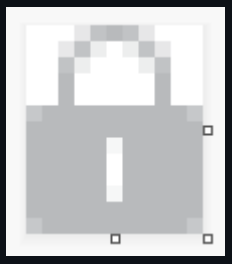
首先，密码的图标是参照QQ登录界面的

等比例画图中自绘即可（截图也随你）。

如果没记错应该是11*13大小。

账号的图标也是自绘。





按键信号和槽函数的设计(登录和注册键)

接下来在`sign.h/.cpp`中同步增加两个函数（这一步可以改动）

.h 中

```
public:
    bool signin();
    bool signup();
```

.cpp 中

```
bool sign::signin()
{
    //qDebug()<<"按了登录键";
    ac = ui->ac->text();
    pw = ui->pw->text();

    int result=dbc::signin(ac,pw);
    if(result==1)
    {
        QMessageBox::critical(this,"登录失败","该账号未注册");
        return false;
    }
    else if(result==2)
    {
```

```
15          QMessageBox::critical(this,"登录失败","密码错误");
16          return false;
17      }
18      else if(result==3)
19      {
20          //qDebug()<<"管理员成功登录";
21          emit adminSI(ac);
22          return true;
23      }
24      else
25      {
26          //qDebug()<<"普通用户成功登录";
27          emit sISuccess(ac);
28          return true;
29      }
30
31  }
32
33  bool sign::signup()
34  {
35      //qDebug()<<"按了注册";
36      ac = ui->ac->text();
37      pw = ui->pw->text();
38      if(ac.length()<=4||pw.length()<=4)
39      {
40          QMessageBox::critical(this,"注册失败","账号与密码长度均不能小于5位！");
41          return false;
42      }
43      bool ok=dbc::signup(ac,pw);
44      if(ok)
45      {
46          QMessageBox::information(this,"注册成功","注册成功！");
47          return true;
48      }
49      else
50      {
51          QMessageBox::critical(this,"注册失败","注册失败！");
52          return false;
```

```
53      }
54
55  }
56
```

函数中使用了dbc类中的静态方法，因此，我们需要创建一个类`dbc`，它是数据库的操作类，包含整个程序中所有对数据库的操作。

关于dbc类的编写，我放到后面介绍，希望大家不要重点关注数据库的操作，因为在编写代码的时候，我们应该做到，要用什么功能就加什么功能，你很难做到先把数据库类编写好，再去写其它模块。因此等看见了数据库操作的时候，再看/编写数据库代码。

在登录函数中，使用了两个信号

1. slSuccess(ac);
2. adminSl(ac);

这两个信号，在.h文件中要同步加上

```
signals:
    void slSuccess(QString ac);
    void adminSl(QString ac);
```

sign类到这里就写好了。

总览

`sign.h`

```
#ifndef SIGN_H
#define SIGN_H

#include <QMainWindow>
#include <QSqlDatabase>
```

```
6
7  QT_BEGIN_NAMESPACE
8  namespace Ui { class sign; }
9  QT_END_NAMESPACE
10
11 class sign : public QMainWindow
12 {
13     Q_OBJECT
14
15
16 signals:
17     void sISuccess(QString ac);
18     void adminSI(QString ac);
19
20 public:
21     QString ac;
22     QString pw;
23
24     sign(QWidget *parent = nullptr);
25     ~sign();
26
27     bool signin();
28     bool signup();
29
30
31
32
33 private:
34     Ui::sign *ui;
35 };
36 #endif // SIGN_H
```

`sign.cpp`

```
1  #include "sign.h"
2  #include "ui_sign.h"
3  #include<qdebug.h>
4  #include<QSqlQuery>
```

```cpp
#include<QMessageBox>
#include<QSqlError>
#include "dbc.h"


sign::sign(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::sign)
{
    ui->setupUi(this);
    setFixedSize(480,296);//黄金分割
    setPalette(QPalette(Qt::white));    //设置窗口为白色背景
    //设置下边框
    ui->ac->setStyleSheet("border-top:0px  solid;border-bottom:1px solid;border-left:0px  solid;border-right: 0px  solid;");
    ui->pw->setStyleSheet("border-top:0px  solid;border-bottom:1px solid;border-left:0px  solid;border-right: 0px  solid;");
    //设置图标显示,这里使用了ui设计中stylesheet来显示，因此注释掉。实际上代码中使用也可以
    //    ui->aclabel->setPixmap(QPixmap(":/image/accountImage.png"));
    //    ui->pwlabel->setPixmap(QPixmap(":/image/passwordImage.png"));
    connect(ui->signIn,&QPushButton::clicked,this,&sign::signin);
    connect(ui->signUp,&QPushButton::clicked,this,&sign::signup);

}

sign::~sign()
{
    delete ui;
}

bool sign::signin()
{
    //qDebug()<<"按了登录键";
    ac = ui->ac->text();
    pw = ui->pw->text();

    int result=dbc::signin(ac,pw);
```

```
40      if(result==1)
41      {
42          QMessageBox::critical(this,"登录失败","该账号未注册");
43          return false;
44      }
45      else if(result==2)
46      {
47          QMessageBox::critical(this,"登录失败","密码错误");
48          return false;
49      }
50      else if(result==3)
51      {
52          //qDebug()<<"管理员成功登录";
53          emit adminSI(ac);
54          return true;
55      }
56      else
57      {
58          //qDebug()<<"普通用户成功登录";
59          emit sISuccess(ac);
60          return true;
61      }
62
63  }
64
65  bool sign::signup()
66  {
67      //qDebug()<<"按了注册";
68      ac = ui->ac->text();
69      pw = ui->pw->text();
70      if(ac.length()<=4||pw.length()<=4)
71      {
72          QMessageBox::critical(this,"注册失败","账号与密码长度均不能小于5位！");
73          return false;
74      }
75      bool ok=dbc::signup(ac,pw);
76      if(ok)
77      {
```

```
78          QMessageBox::information(this,"注册成功","注册成功！");
79          return true;
80      }
81      else
82      {
83          QMessageBox::critical(this,"注册失败","注册失败！");
84          return false;
85      }
86
87  }
```

当登录成功后，sign类的实例会发出信号。

connect函数会监听信号，当收到相应的信号，就会触发槽函数。

不同的信号可以触发不同的槽函数，同一个信号也可以触发多个槽函数，多写几个connect就行。

监听信号的connect语句写在哪？写在main函数里面。

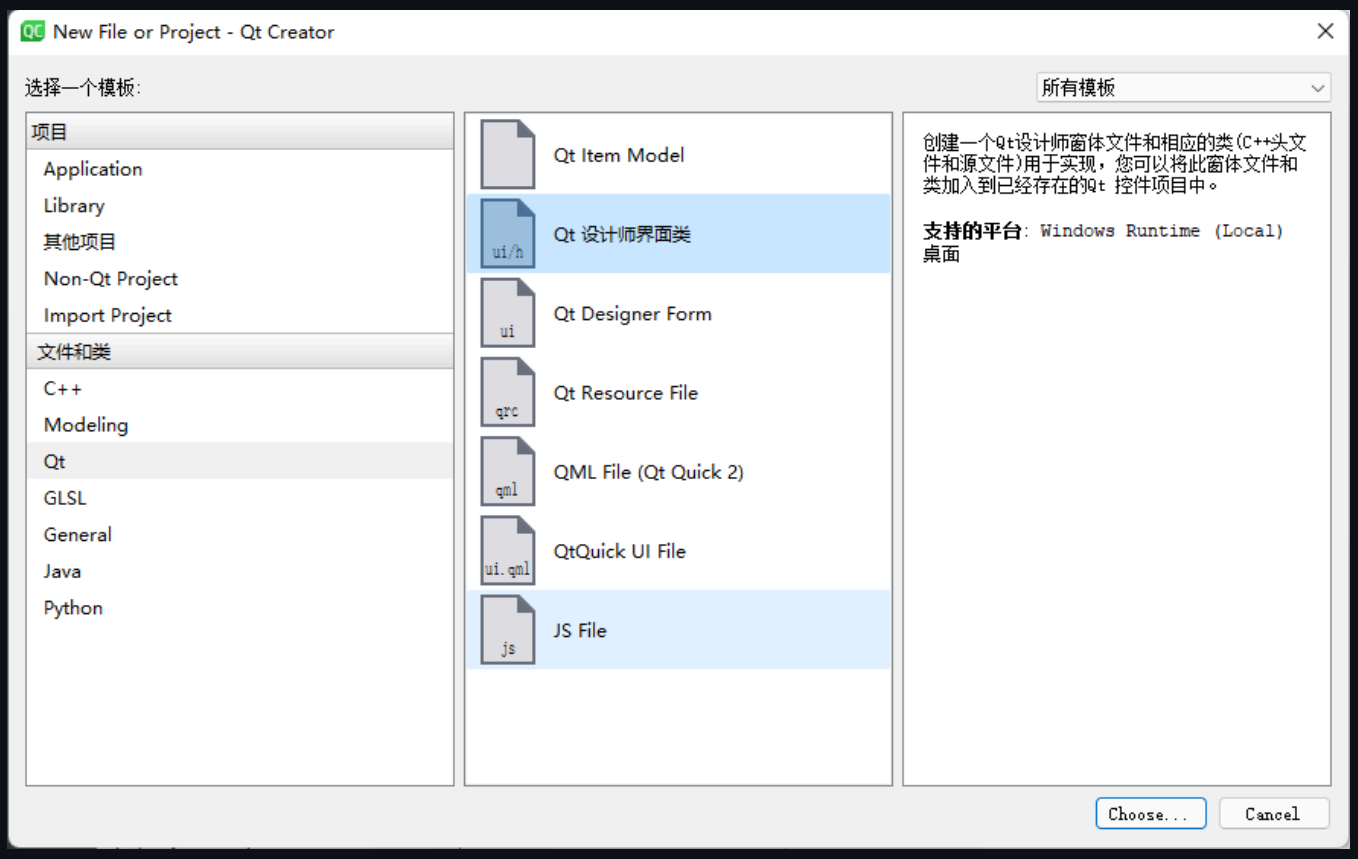
当触发了登录成功，就要进入下一个界面。根据之前的设计图，下一个界面可能是管理员界面或者用户界面。

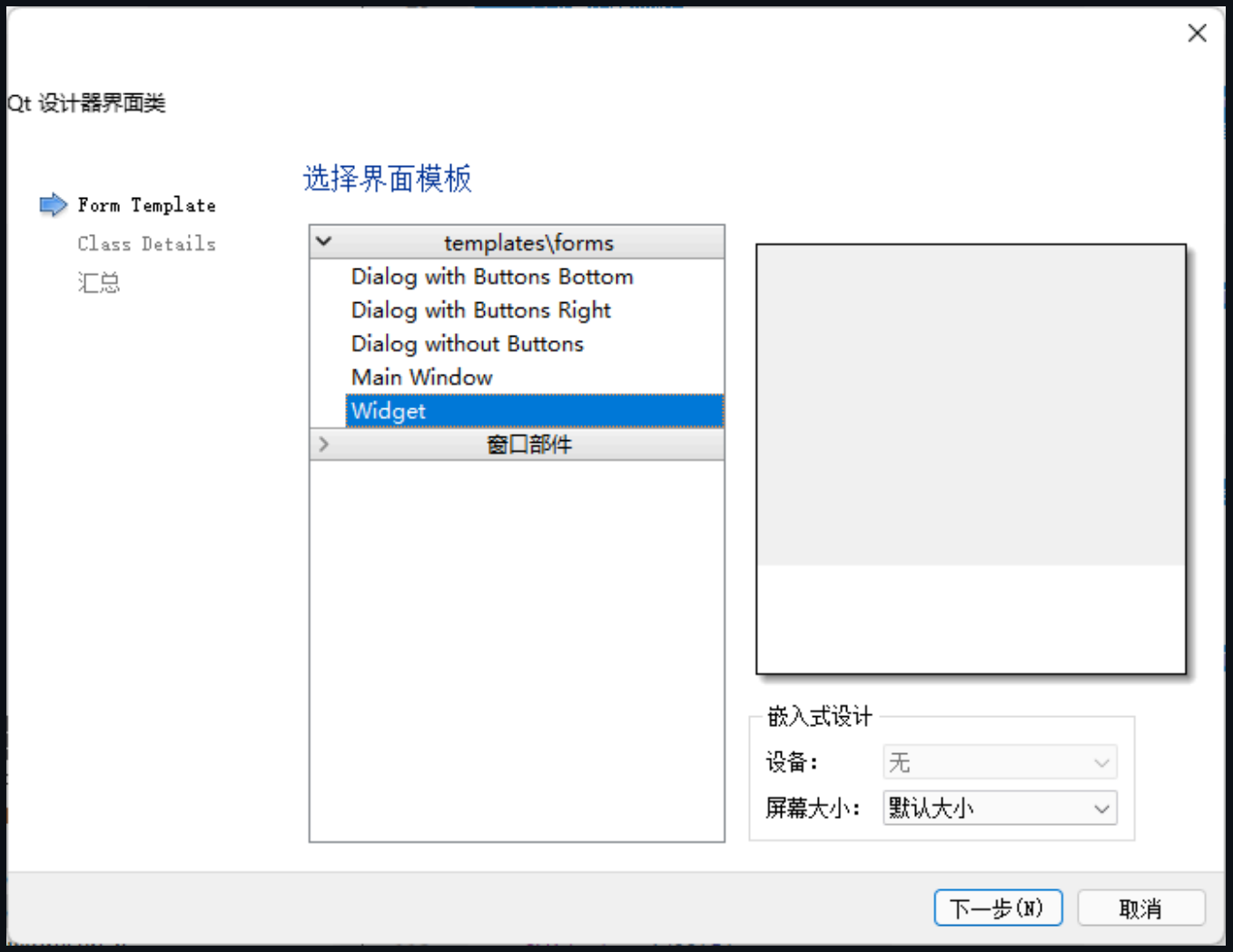
我先写的是用户界面，实际上顺序可以换。而且两个界面需要交替设计，用户修改个人信息，管理员增加课程，用户选课，管理员对选课记录给一个分数，用户查分。这些功能是交替进行的，因此需要交替编写。

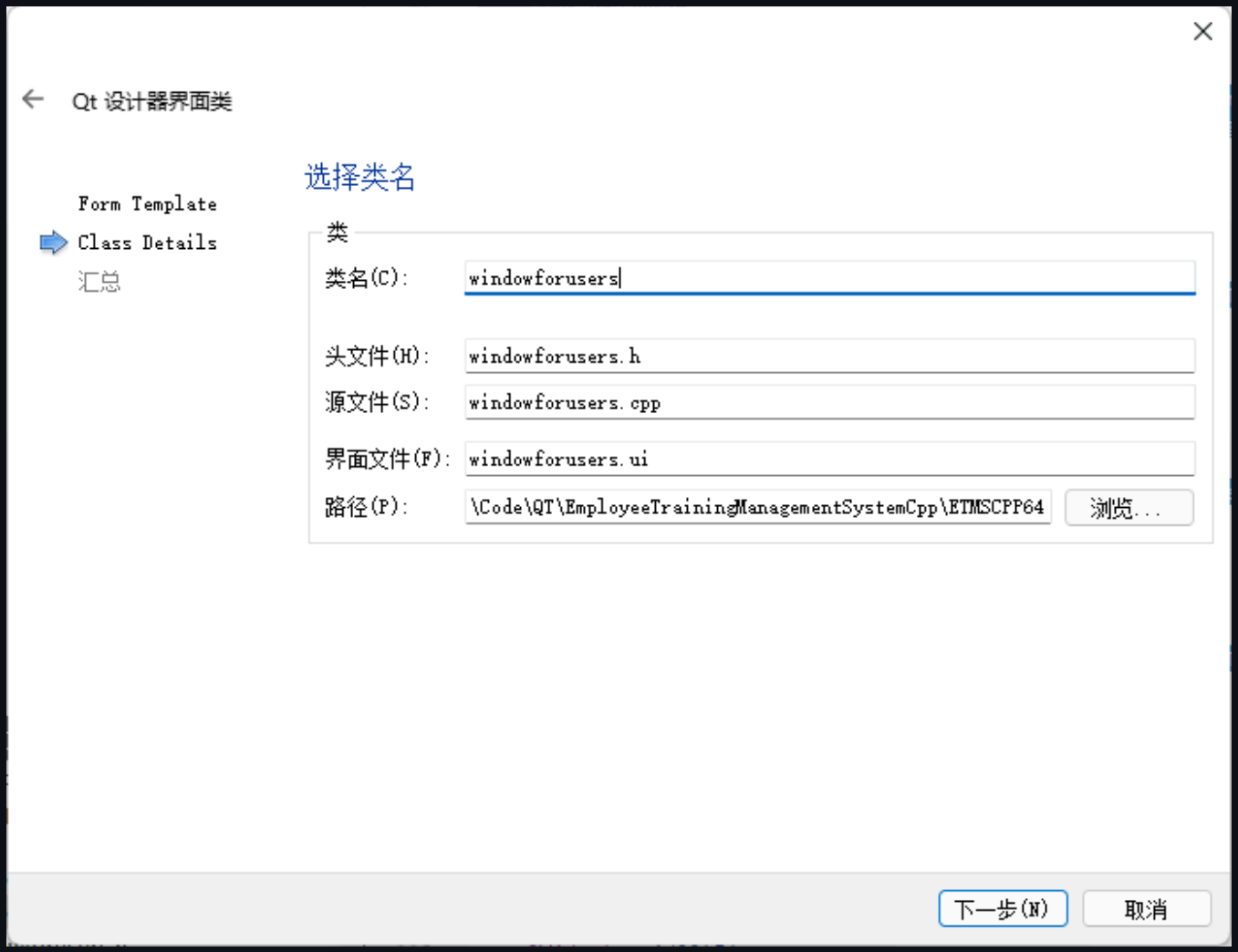
windowforusers类的设计（用户界面）

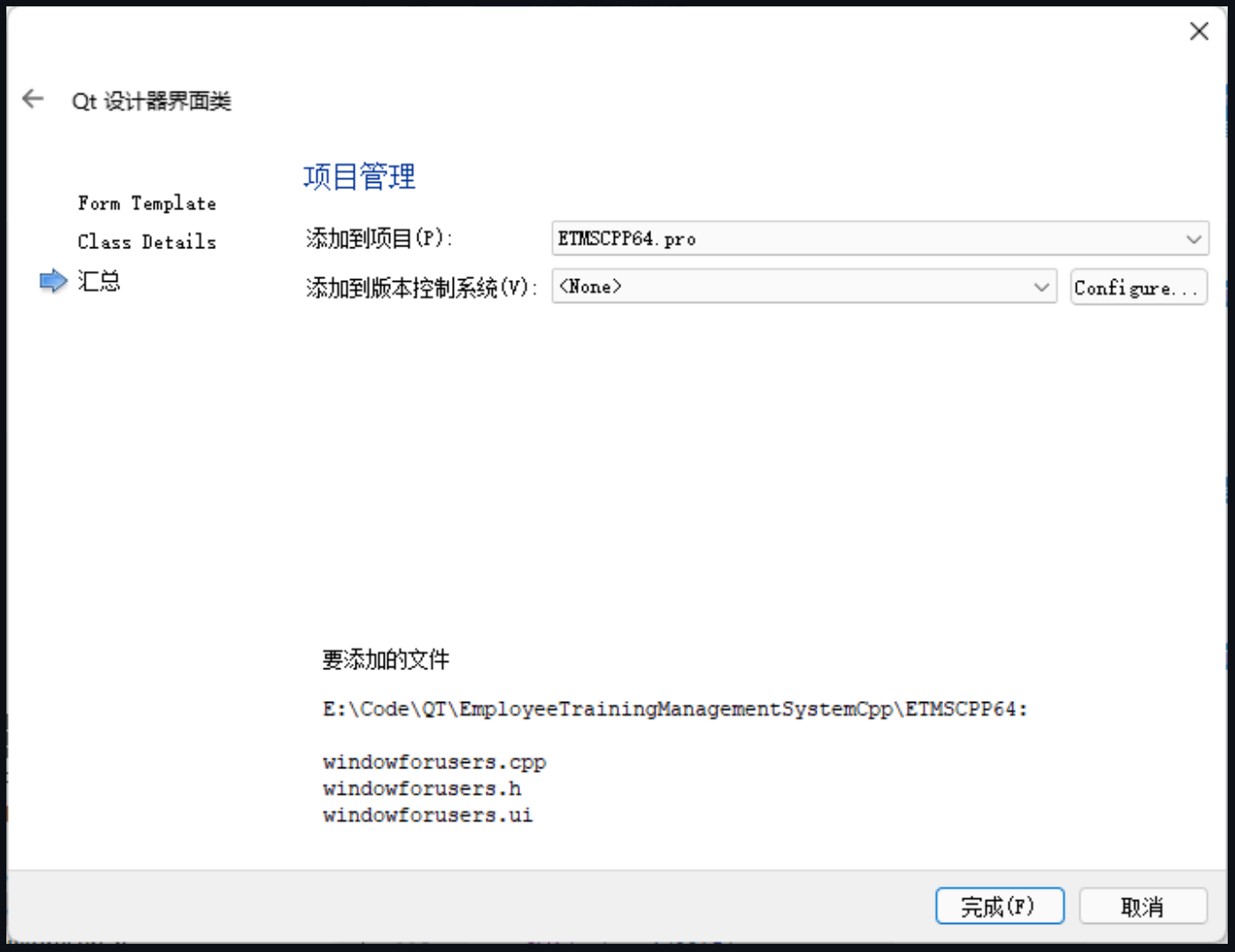
创建一个类继承QMainWindow，然后删掉状态栏和菜单栏。

（好像QWidget就行...它就没有状态栏和菜单栏，对的...我多此一举）


New File or Project - Qt Creator
选择一个模板:
所有模板
项目
Application
Library
其他项目
Non-Qt Project
Import Project
文件和类
C++
Modeling
Qt
GLSL
General
Java
Python
Qt Item Model
Qt 设计师界面类
Qt Designer Form
Qt Resource File
QML File (Qt Quick 2)
QtQuick UI File
JS File
创建一个Qt设计师窗体文件和相应的类(C++头文件和源文件)用于实现，您可以将此窗体文件和类加入到已经存在的Qt 控件项目中。
支持的平台: Windows Runtime (Local)
桌面
Choose...
Cancel


Qt 设计器界面类
选择界面模板
Form Template
Class Details
汇总
templates\forms
Dialog with Buttons Bottom
Dialog with Buttons Right
Dialog without Buttons
Main Window
Widget
窗口部件
嵌入式设计
设备:
无
屏幕大小:
默认大小
下一步(N)
取消


Qt 设计器界面类
Form Template
Class Details
汇总
选择类名
类
类名(C):
windowforusers
头文件(H):
windowforusers.h
源文件(S):
windowforusers.cpp
界面文件(F):
windowforusers.ui
路径(P):
\Code\QT\EmployeeTrainingManagementSystemCpp\ETMSCPP64
浏览...
下一步(N)
取消


Qt 设计器界面类
Form Template
Class Details
汇总
项目管理
添加到项目(P):
ETMSCPP64.pro
添加到版本控制系统(V):
<None>
Configure...
要添加的文件
E:\Code\QT\EmployeeTrainingManagementSystemCpp\ETMSCPP64:
windowforusers.cpp
windowforusers.h
windowforusers.ui
完成(F)
取消

首先看ui界面。

ui 界面的初始化：

```
WindowForUsers::WindowForUsers(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::WindowForUsers)
{
    //setupUi(this)是由.ui文件生成的类的构造函数,这个函数的作用是对界面进行初始化，它按照我们在Qt设计器里设计的样子把窗体画出来，把我们在Qt设计器里面定义的信号和槽建立起来。
    ui->setupUi(this);
    setFixedSize(800,600);
    setPalette(QPalette(Qt::white));   //设置窗口为白色背景
    ui->birT->setCalendarPopup(true);
    //注意ed，使用了 转到槽 所以取消了connect
    //connect(ui->exitB,&QPushButton::clicked,this,&WindowForUsers::close);

}
```

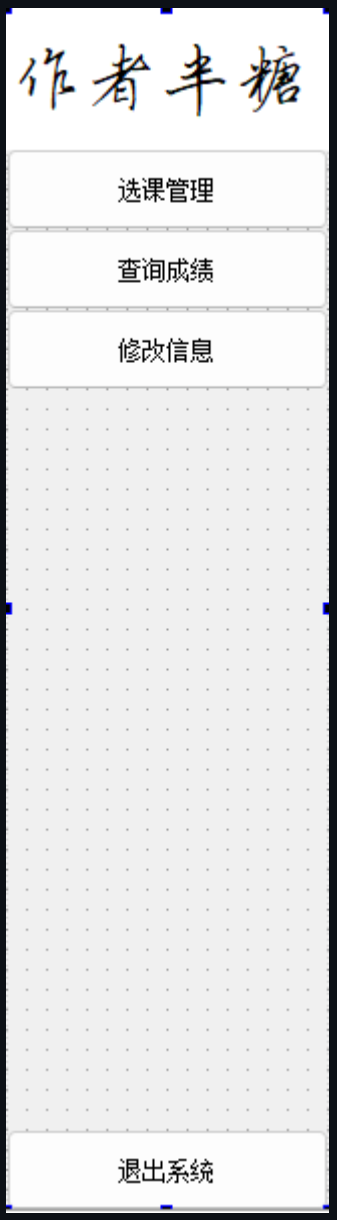
start函数（收到登录界面的信号后需要执行的函数）

```
void WindowForUsers::start(QString ac)
{

    this->ac=ac;
    this->show();
    QString welcome="欢迎您，用户"+ac;
    //记得把ui拉长，以防显示不全。
    ui->welcome->setText(welcome);
    //qDebug()<<welcome;
    QLineEdit *wid2[2];
    wid2[0]=ui->idT;
    wid2[1]=ui->nameT;
    QWidget *wid22[5];
    wid22[0]=ui->perT;
    wid22[1]=ui->sexT;
    wid22[2]=ui->staT;
    wid22[3]=ui->compT;
    wid22[4]=ui->depaT;
    for(int i=0;i<2;i++) wid2[i]->setEnabled(false);
    for(int i=0;i<5;i++) wid22[i]->setEnabled(false);
    ui->birT->setEnabled(false);

    ui->stackedWidget->setCurrentIndex(3);
    //为使按键可以检查，才能触发toggled。默认不能检查，就不能toggled。
    ui->changeInfo->setCheckable(true);
    ui->changeInfo->setText("修改信息");
    QVector<QString> info;
    info.push_back(ac);
    dbc::findinfo(info);
    dataComplete=true;
    for(auto a:info)
    {
        if(a.isEmpty())
            dataComplete=false;
    }
    ui->idT->setText(info[1]);
```

```
37      ui->nameT->setText(info[2]);
38      ui->sexT->setCurrentText(info[3]);
39      ui->birT->setDate(QDate::fromString(info[4]));
40      ui->compT->setCurrentText(info[5]);
41      ui->depaT->setCurrentText(info[6]);
42      ui->perT->setCurrentText(info[7]);
43      ui->staT->setCurrentText(info[8]);
44
45      course=new QSqlTableModel(this);
46      //设置表名
47      course->setTable("course");
48      //设置编辑策略
49      course->setEditStrategy(QSqlTableModel::OnFieldChange);
50      //查找截止日期之前的课，也就是可选的课
51      QString
    timenow=QString::number(QDateTime::currentDateTime().toTime_t());
52      course->setFilter(QString("(exptime>"+timenow+") and
    (signnum<maxnum)"));
53      //查询
54      course->select();
55      //显示
56      ui->tableView->setModel(course);
57      //设置不可编辑
58      ui->tableView->setEditTriggers(QAbstractItemView::NoEditTriggers);
59
60      //创建对象
61      signRecord=new QSqlTableModel(this);
62      //设置表名
63      signRecord->setTable("signrecord");
64      //设置编辑策略
65      signRecord->setEditStrategy(QSqlTableModel::OnFieldChange);
66      //查询
67      signRecord->setFilter("ac=\'"+ac+"\'");
68      signRecord->select();
69      ui->gradeView->setModel(signRecord);
70      //不可编辑
71      ui->gradeView->setEditTriggers(QAbstractItemView::NoEditTriggers);
72
```

```
73  }
```

左边是一个QWidget，



这个QWidget上面是一个label，用来显示图片，把该区域的分辨率调好以后，用画图制图。然后就是三个按钮，这三个按钮是我手动放的，失策，应该做一个按钮组，QButtonGroup，但当时没学到这个东西。后面有使用。读者若会使用，可以尝试改进一下。参考 QButtonGroup的使用，使用按钮组，可以设置 在按钮组中的按钮只可单选，这个功能在后面是我手写的（有点麻烦，但不难）。

退出按钮是单独的，它的槽函数也很简单，就是关闭整个程序。

```
void WindowForUsers::on_exitB_clicked()
{
    this->close();
}
```

上面三个按键的功能，主要是三方面：

1. 翻页
2. 设置按下

   设置按下这里需要说明一下，按下一个键以后，理论上需要实现的效果是，此键是按下的状态且不能再按（灰色），其它的键如果是按下状态，则弹起。其实也就是个单选功能。如果学了按钮组，何至于此！
3. 其还需要实现的功能

```
void WindowForUsers::on_selectCourse_clicked()
{
    //查找截止日期之前的课，也就是可选的课
    QString timenow=QString::number(QDateTime::currentDateTime().toTime_t());
    course->setFilter(QString("(exptime>"+timenow+") and (signnum<maxnum)"));
    //显示
    course->select();

    ui->stackedWidget->setCurrentIndex(0);
    ui->selectCourse->setEnabled(false);
    ui->checkResult->setEnabled(true);
    ui->updateInformation->setEnabled(true);
}

void WindowForUsers::on_checkResult_clicked()
{
    //查询
    signRecord->setFilter("ac=\'"+ac+"\'");
```

```
    signRecord->select();

    ui->stackedWidget->setCurrentIndex(1);
    ui->selectCourse->setEnabled(true);
    ui->checkResult->setEnabled(false);
    ui->updateInformation->setEnabled(true);
}

void WindowForUsers::on_updateInformation_clicked()
{
    ui->stackedWidget->setCurrentIndex(2);
    ui->selectCourse->setEnabled(true);
    ui->checkResult->setEnabled(true);
    ui->updateInformation->setEnabled(false);
}
```

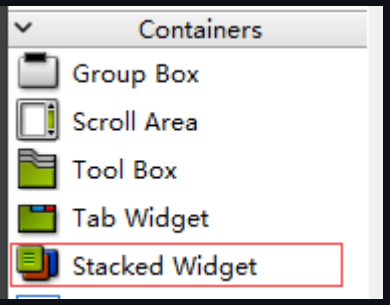
四行统一的代码就是翻页和设置单选的功能。其它代码就是这个按键独有的功能。对于选课按钮，其独有的功能是实时刷新，每次点击都相当于刷新课程。

同理，查询分数的按键，也自带刷新功能。

至于修改个人信息的按键，在初始化以后就是实时的，原因读者自己理解。

右边界面

上面的图片依然是label，下面使用了stacked widget，页面为4个，下标为0,1,2,3。



接下来介绍每个界面的详细设计。

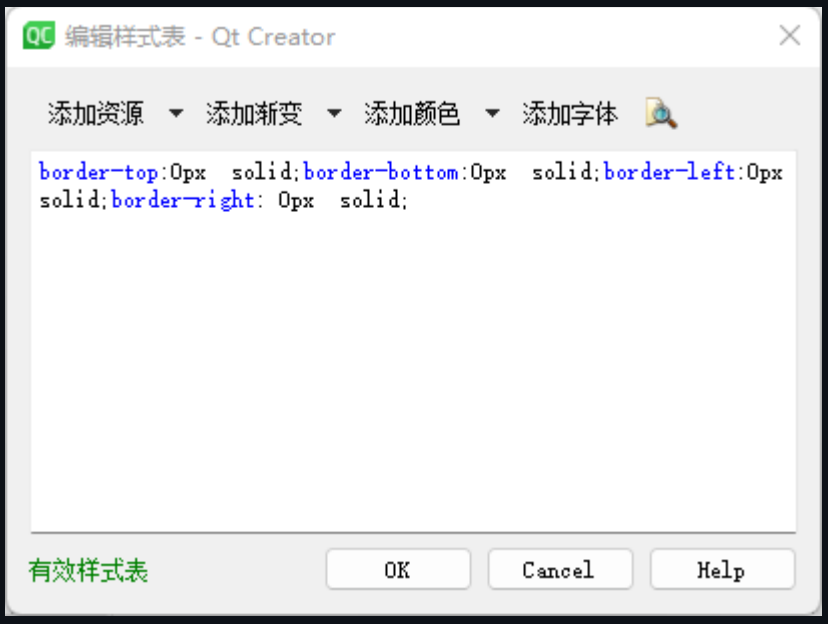
页面3

打开这个窗口，会默认显示页面3，代码在初始化函数内，

```
1  ui->stackedWidget->setCurrentIndex(3);
```

本页面非常简单，一句话（label）加一个按钮。

个人信息就是个push button。设置stylesheet如下：

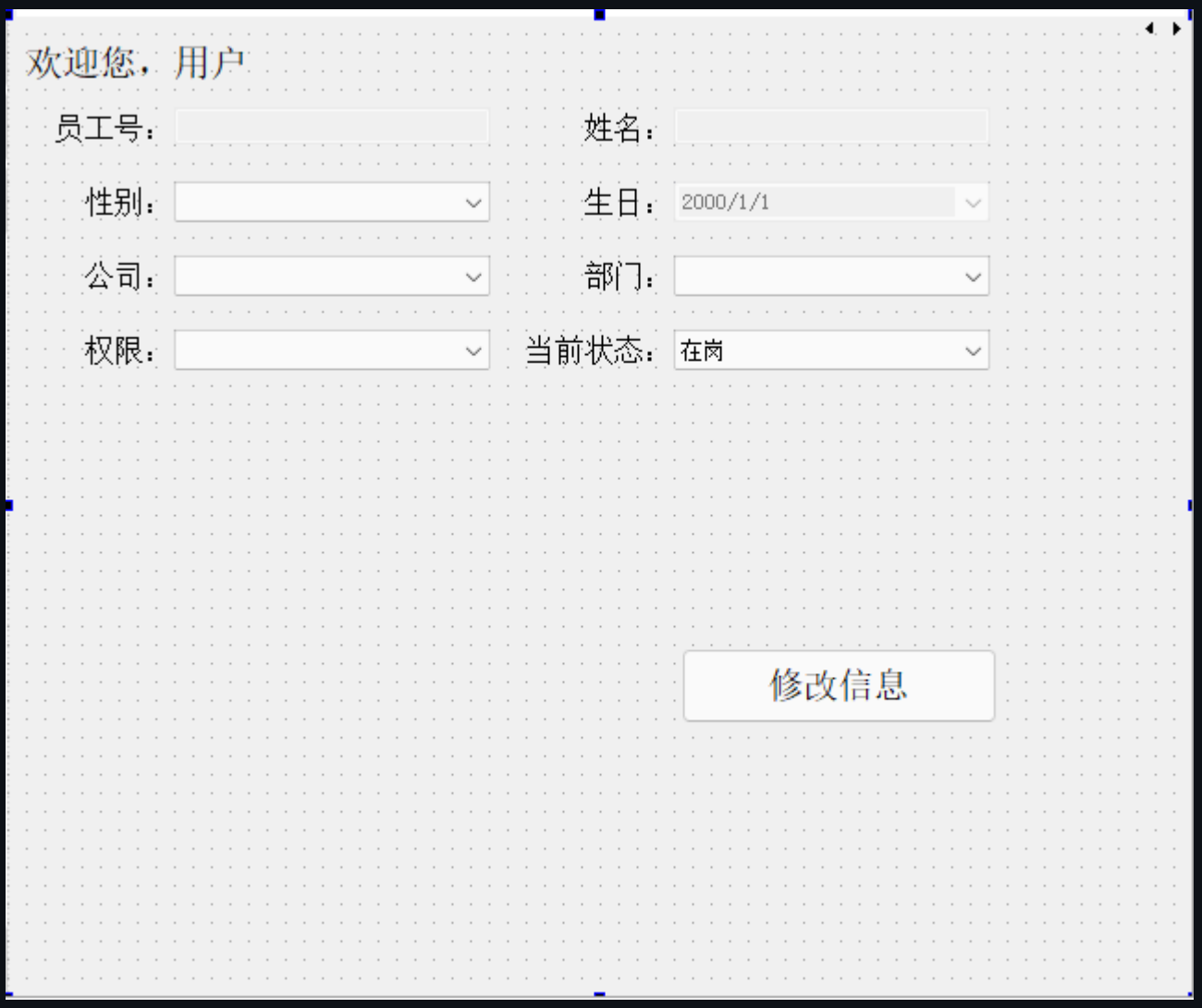



```
1 border-top:0px solid;border-bottom:0px solid;border-left:0px solid;border-right:
  0px solid;
```

其实就是取消边框。

勾上下划线。

页面2

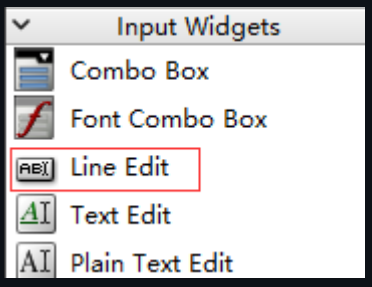


页面2是修改个人信息的，其操作在start函数中和修改信息按键中有实现。主要是设置按键的Checkable属性，代码如下

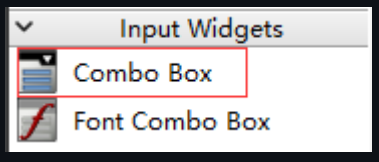
```
1    ui->changeInfo->setCheckable(true);
2    ui->changeInfo->setText("修改信息");
```

Checkable属性就是让按键有按下和抬起两种状态。类似电视机的开关那样。然后根据其按下或者抬起来设置是否在编辑状态。

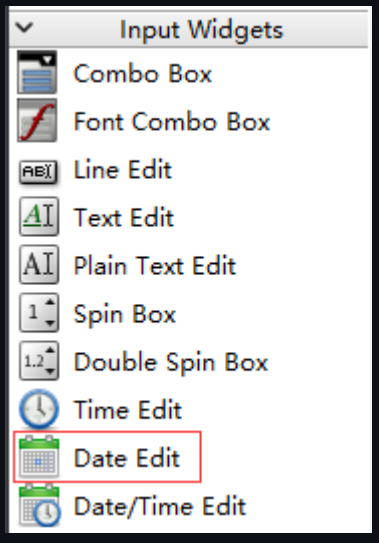
输入框是LineEdit



下拉框是Combo box



生日的日期输入框是Date Edit



页面0

这是选课管理的界面。

大白框是QTableView

这个类需要设置model，其model在start里有定义，相关代码如下。

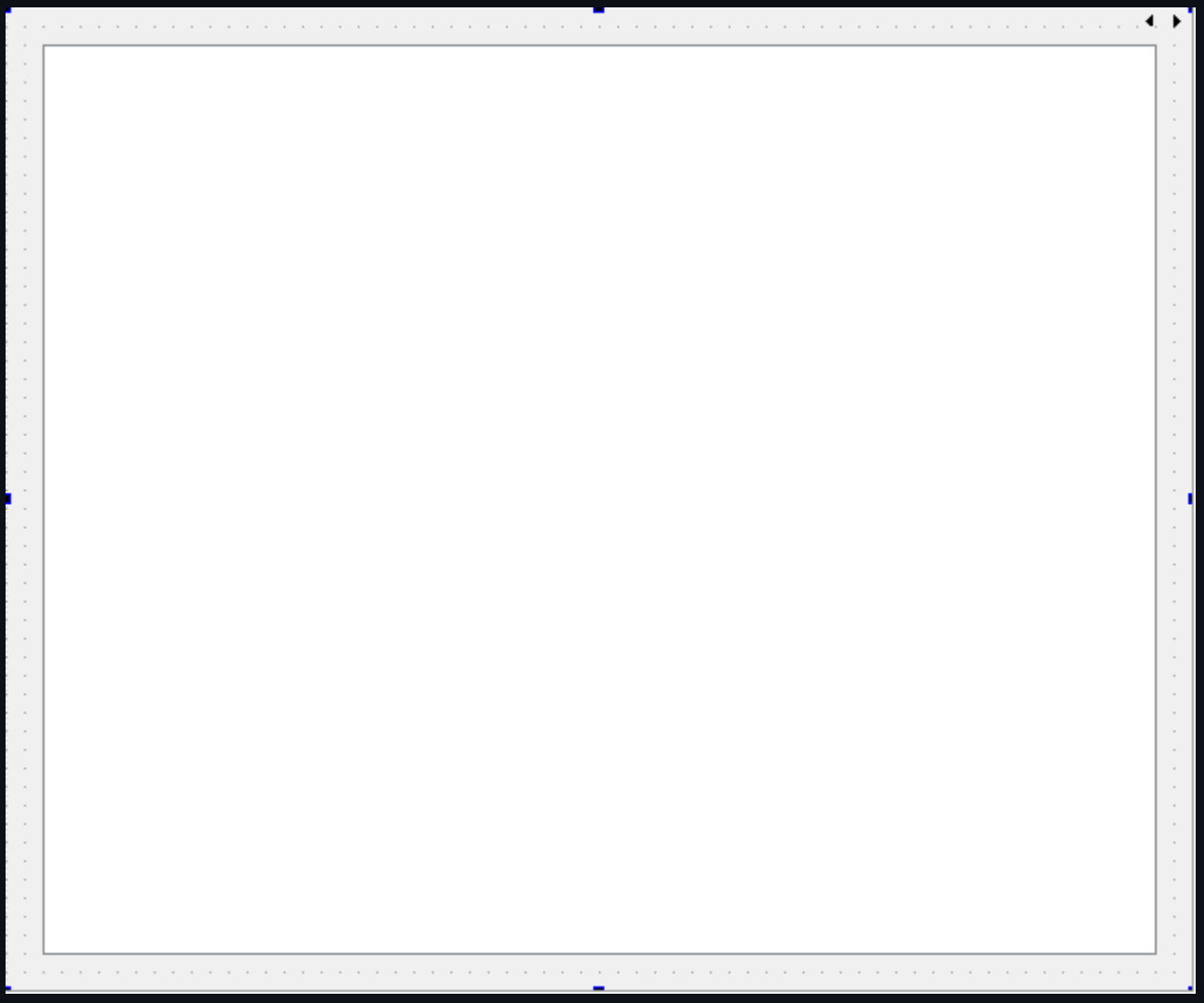
```
course=new QSqlTableModel(this);
    //设置表名
    course->setTable("course");
    //设置编辑策略
    course->setEditStrategy(QSqlTableModel::OnFieldChange);
    //查找截止日期之前的课，也就是可选的课
    QString
timenow=QString::number(QDateTime::currentDateTime().toTime_t());
```

```
8     course->setFilter(QString("(exptime>"+timenow+") and
  (signnum<maxnum)"));
9     //查询
10    course->select();
11    //显示
12    ui->tableView->setModel(course);
13    //设置不可编辑
14    ui->tableView->setEditTriggers(QAbstractItemView::NoEditTriggers);
```

注释比较完全，不多赘述。

页面1

界面



大白框依然是QTableView，相关代码如下。

```
    //创建对象
    signRecord=new QSqlTableModel(this);
    //设置表名
    signRecord->setTable("signrecord");
    //设置编辑策略
    signRecord->setEditStrategy(QSqlTableModel::OnFieldChange);
    //查询
    signRecord->setFilter("ac=\'"+ac+"\'");
    signRecord->select();
    ui->gradeView->setModel(signRecord);
    //不可编辑
    ui->gradeView->setEditTriggers(QAbstractItemView::NoEditTriggers);
```

windowforadmin类的设计

同样新建一个QWidget，步骤参考上面的类。

初始化函数如下。

```
WindowForAdmin::WindowForAdmin(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::WindowForAdmin)
{
    ui->setupUi(this);

    setFixedSize(800,600);
    setPalette(QPalette(Qt::white));   //设置窗口为白色背景
    ui->stackedWidget->setCurrentIndex(3);
    ui->exptT->setCalendarPopup(true);

    //设置一组按钮
    bg=new QButtonGroup(ui->page_2);
    bg->addButton(ui->scb1,0);
    bg->addButton(ui->scb2,1);
    bg->addButton(ui->scb3,2);
    //设置独占，即只能选中一个按钮
    bg->setExclusive(true);
```

```
    //默认选中 按用户账号查找
    ui->scb1->setChecked(true);
}
```

start函数如下。

```
void WindowForAdmin::start(QString ac)
{
    this->ac=ac;
    this->show();
    //创建对象
    signRecord=new QSqlTableModel(this);
    //设置表名
    signRecord->setTable("signrecord");
    //设置编辑策略
    signRecord->setEditStrategy(QSqlTableModel::OnFieldChange);
    //查询整张表
    signRecord->select();
    ui->tableView->setModel(signRecord);
}
```

布局展示

类似上面的类。

旁边的按钮有翻页功能。代码如下。

```
void WindowForAdmin::on_addCourse_clicked()
{
    ui->stackedWidget->setCurrentIndex(0);
    ui->addCourse->setEnabled(false);
    ui->checkCourse->setEnabled(true);
    ui->exptT->setDateTime(QDateTime::currentDateTime());
    ui->addAdmin->setEnabled(true);
}

void WindowForAdmin::on_checkCourse_clicked()
{
    ui->stackedWidget->setCurrentIndex(1);
    ui->addCourse->setEnabled(true);
    ui->checkCourse->setEnabled(false);
    ui->addAdmin->setEnabled(true);
}
```

```
17
18  void WindowForAdmin::on_addAdmin_clicked()
19  {
20      ui->stackedWidget->setCurrentIndex(2);
21      ui->addAdmin->setEnabled(false);
22      ui->addCourse->setEnabled(true);
23      ui->checkCourse->setEnabled(true);
24  }
```

页面0

按确定以后，所有信息按顺序存入数组，使用dbc类的方法存入数据库。

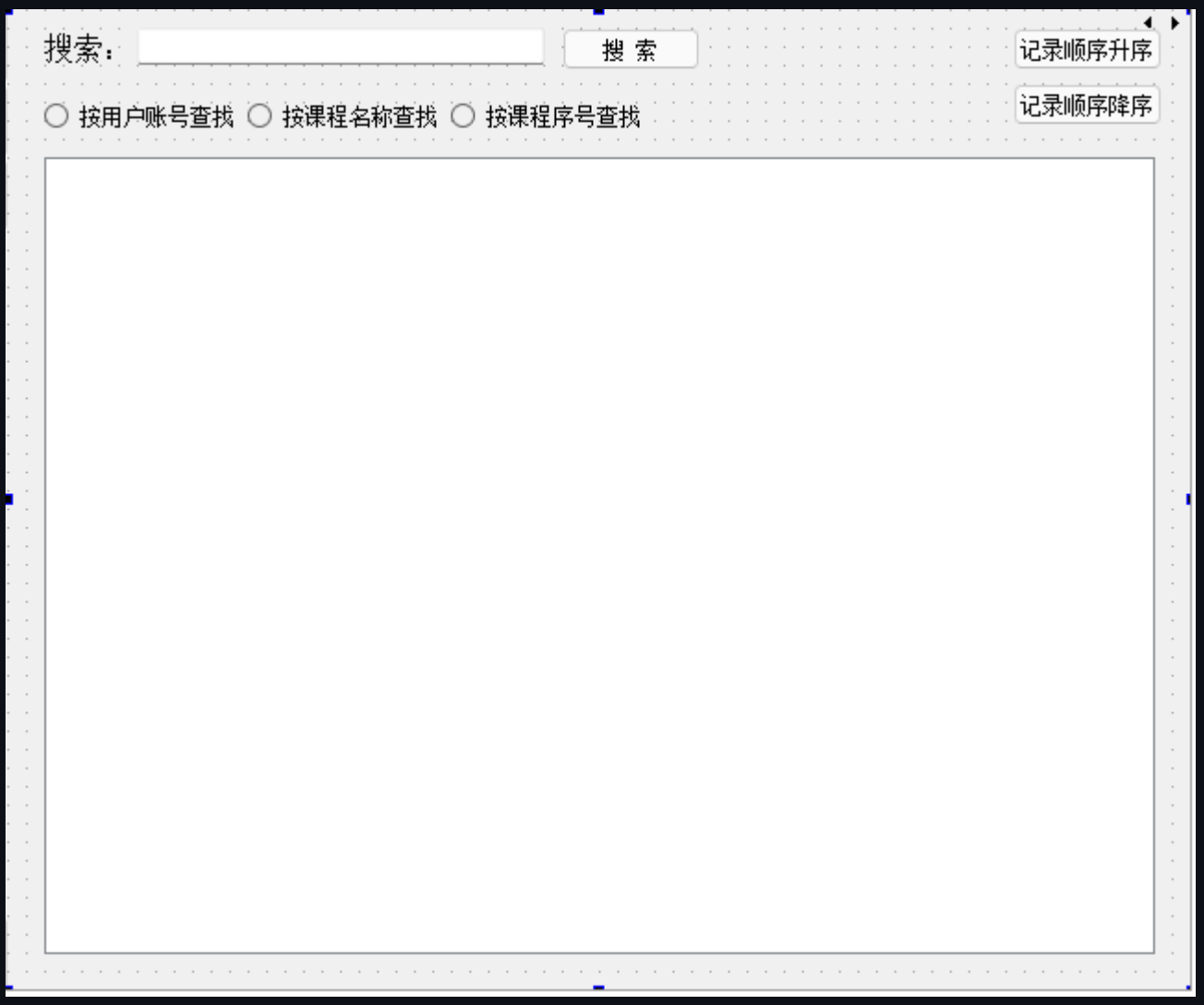
存储的代码如下。

```
1  void WindowForAdmin::on_OKB_clicked()
```

```
{
    QVector<QString> cinfo;
    cinfo.push_back(ui->nameT->text());
    cinfo.push_back(ui->intrT->document()->toPlainText());
    cinfo.push_back(ui->bookT->document()->toPlainText());
    cinfo.push_back(ui->addrT->text());
    cinfo.push_back(QString::number(ui->maxnumT->value()));
    cinfo.push_back(QString::number(ui->exptT->dateTime().toTime_t()));
    if(dbc::addC(cinfo))
    {
        QMessageBox::information(this,"课程成功增加","课程增加成功！");
        this->on_CLRB_clicked();
    }
    else
    {
        QMessageBox::critical(this,"课程添加失败","课程添加失败，您可以试试减少输入字数！");
    }

}
```

页面1

ui



大白框依然是QTableView。

查找功能改变了select语句的条件。

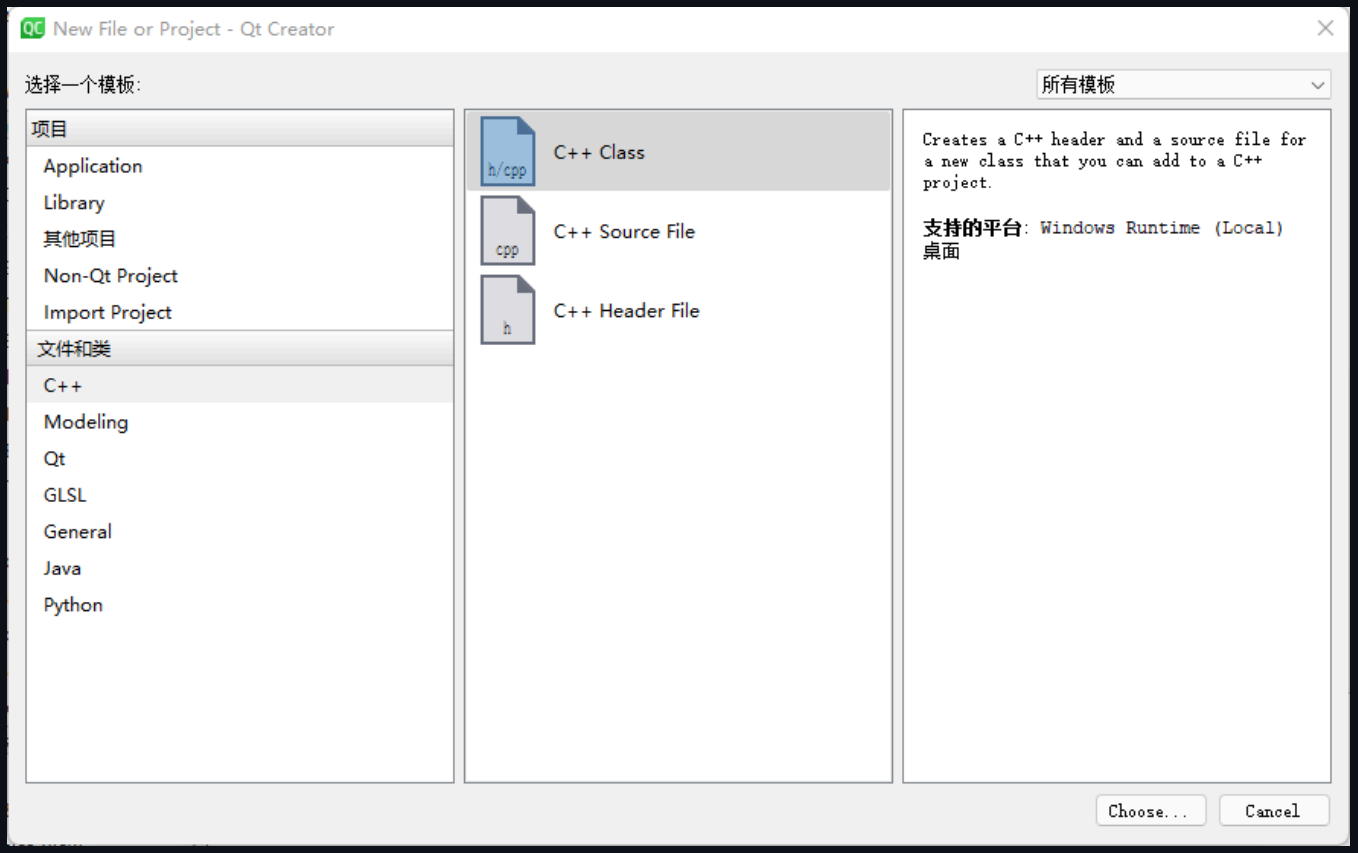
页面2

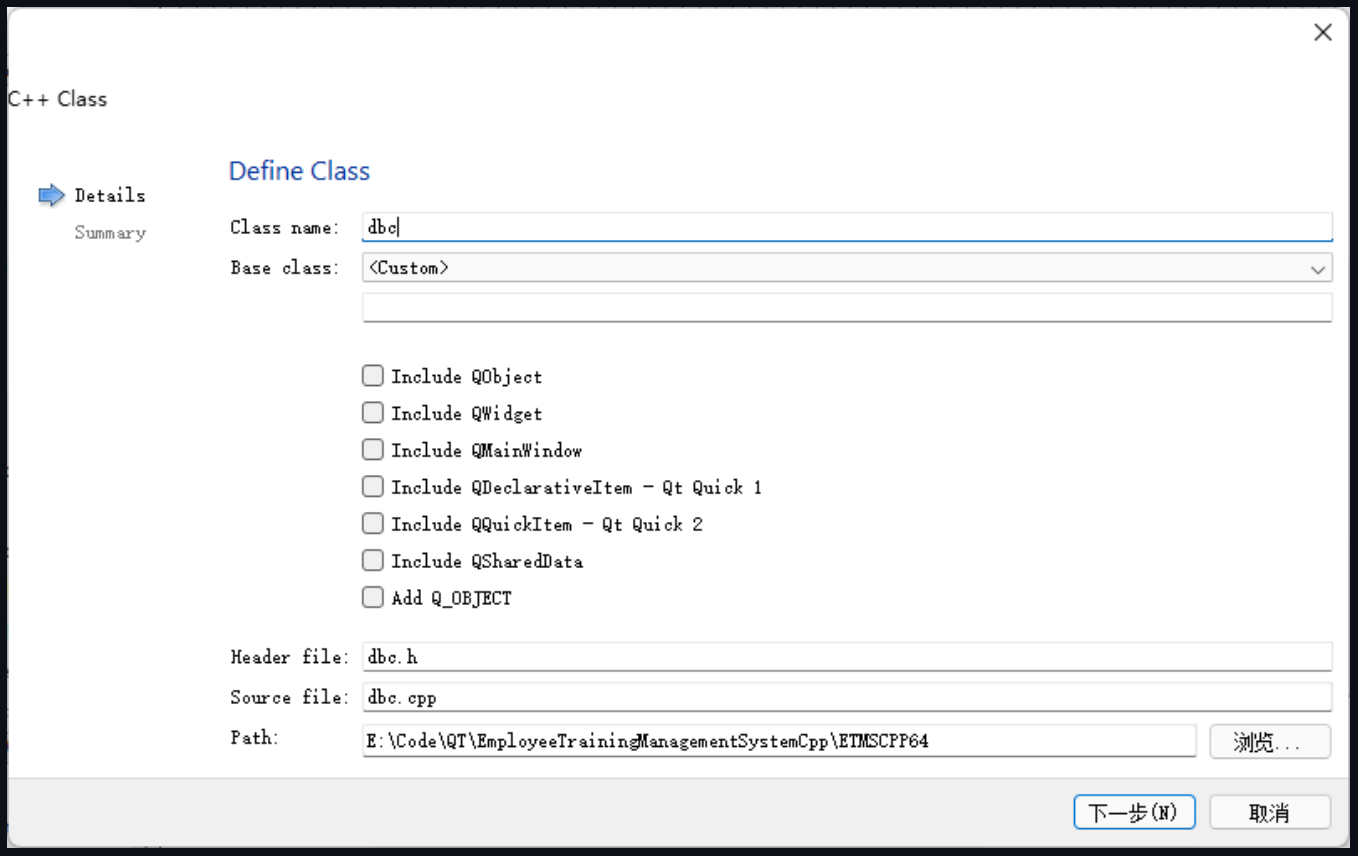
ui

这里稍微说一说账号系统的设计，账号系统设计，除了账号密码，还有一个属性“whetheradmin”。注册账号时默认是0。其它管理员可以设置普通用户为管理员。登录的时候检查这个属性，以确定展示哪个页面。

dbc类的编写

新建C++ Class


New File or Project - Qt Creator
选择一个模板:
所有模板
项目
Application
Library
其他项目
Non-Qt Project
Import Project
文件和类
C++
Modeling
Qt
GLSL
General
Java
Python
C++ Class
C++ Source File
C++ Header File
Creates a C++ header and a source file for a new class that you can add to a C++ project.
支持的平台: Windows Runtime (Local)
桌面
Choose...
Cancel


C++ Class
Details
Summary
Define Class
Class name: dbc
Base class: <Custom>
Include QObject
Include QWidget
Include QMainWindow
Include QDeclarativeItem - Qt Quick 1
Include QQuickItem - Qt Quick 2
Include QSharedData
Add Q_OBJECT
Header file: dbc.h
Source file: dbc.cpp
Path: E:\Code\QT\EmployeeTrainingManagementSystemCpp\ETMSCPP64
浏览...
下一步(N)
取消

然后 下一步 就好。

ODBC

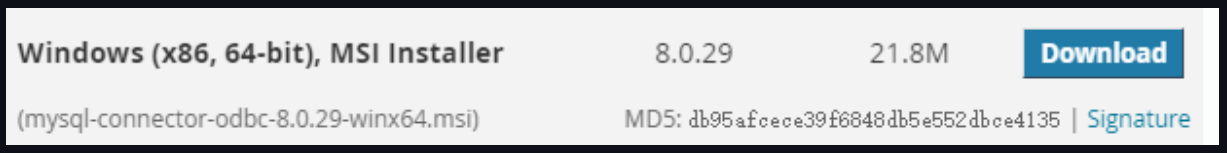
使用ODBC连接数据库，参考 Qt连接MySQL数据库最详细的教程

首先确定自己的数据库是多少位的，切记本工程与数据库位数要对应。这点在开始设计时就说明了。

因为ODBC和数据库要一样的位数，ODBC和代码也要一样的位数。

步骤

1. 打开官网
2. 选择对应版本下载，我的是64位所以选择



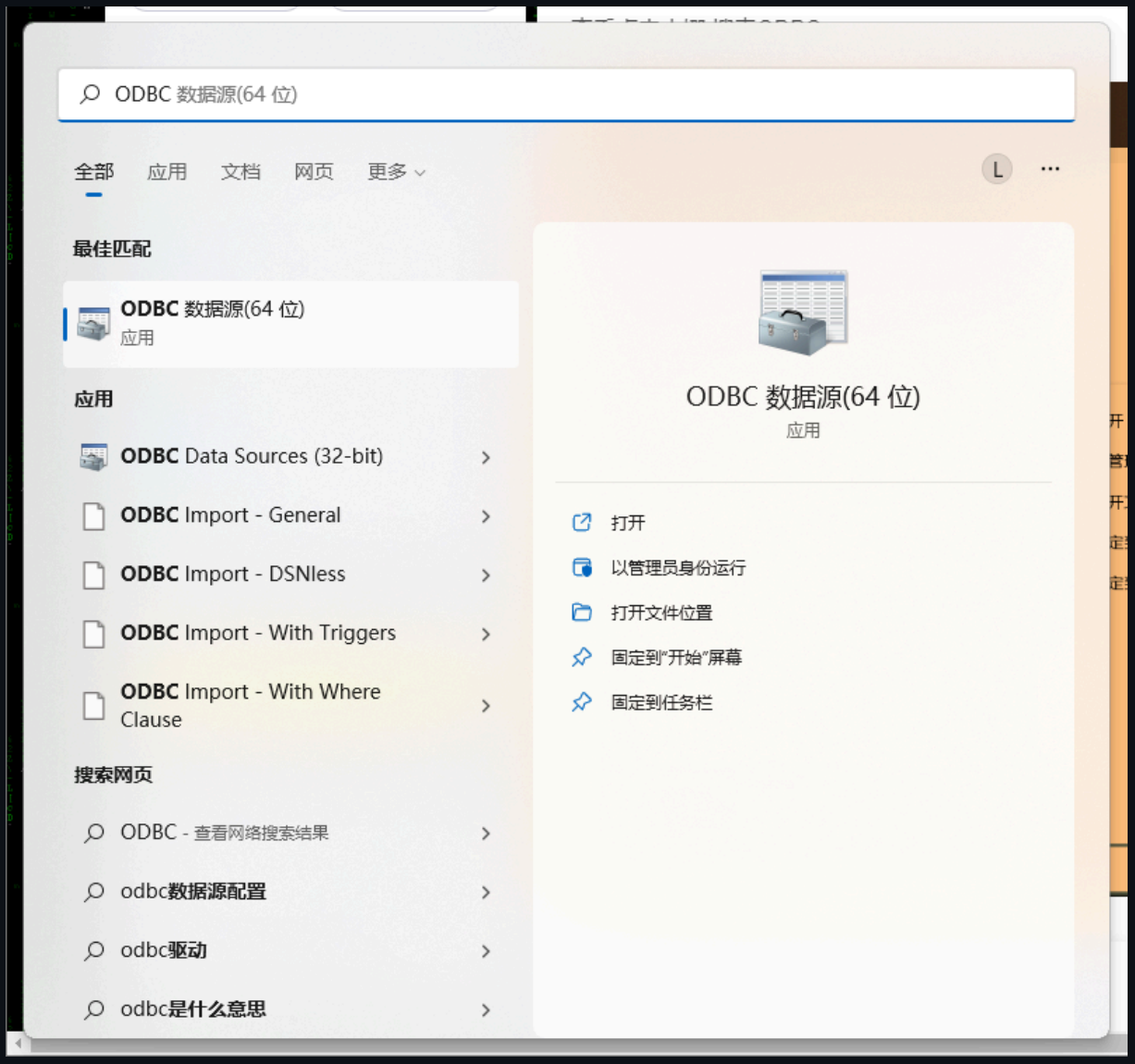
   下载链接64位

   下载链接32位

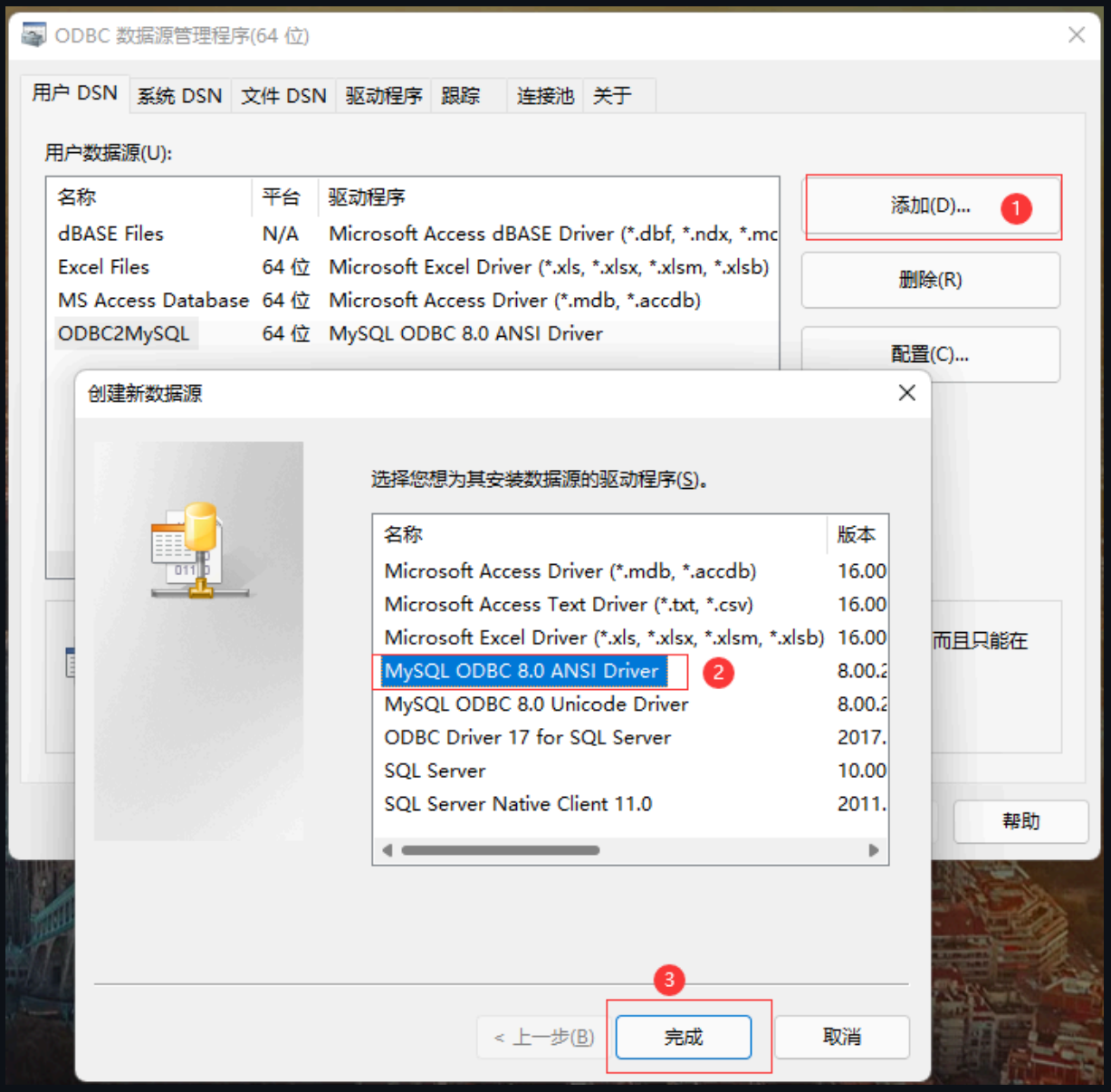
3. 安装

   一直next就行。

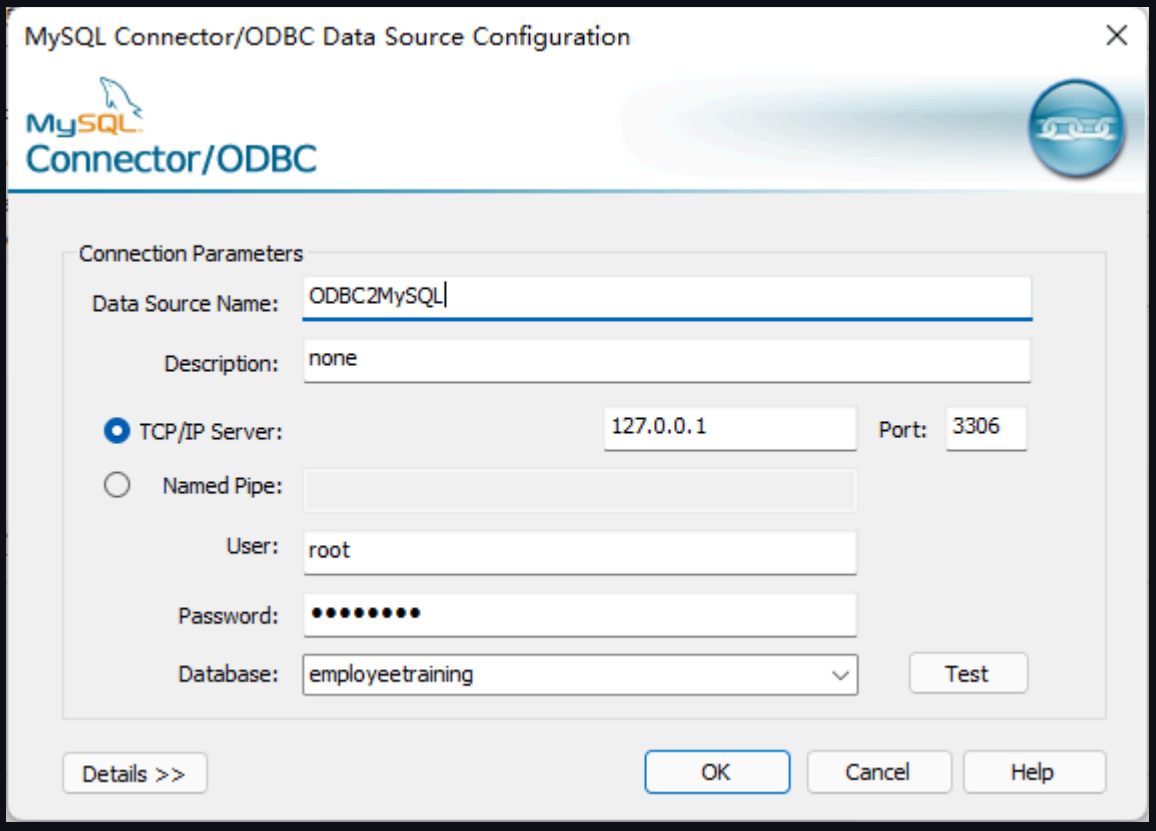
4. 左下角搜索ODBC


ODBC 数据源(64 位)
全部
应用
文档
网页
更多
L
最佳匹配
ODBC 数据源(64 位)
应用
应用
ODBC Data Sources (32-bit)
ODBC Import - General
ODBC Import - DSNless
ODBC Import - With Triggers
ODBC Import - With Where Clause
搜索网页
ODBC - 查看网络搜索结果
odbc数据源配置
odbc驱动
odbc是什么意思
ODBC 数据源(64 位)
应用
打开
以管理员身份运行
打开文件位置
固定到"开始"屏幕
固定到任务栏

5. 如图点击


ODBC 数据源管理程序(64 位)
用户 DSN
系统 DSN
文件 DSN
驱动程序
跟踪
连接池
关于
用户数据源(U):
名称
平台
驱动程序
dBASE Files
N/A
Microsoft Access dBASE Driver (*.dbf, *.ndx, *.mc
Excel Files
64 位
Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)
MS Access Database
64 位
Microsoft Access Driver (*.mdb, *.accdb)
ODBC2MySQL
64 位
MySQL ODBC 8.0 ANSI Driver
添加(D)...
1
删除(R)
配置(C)...
帮助
创建新数据源
选择您想为其安装数据源的驱动程序(S)。
名称
版本
Microsoft Access Driver (*.mdb, *.accdb)
16.00
Microsoft Access Text Driver (*.txt, *.csv)
16.00
Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)
16.00
MySQL ODBC 8.0 ANSI Driver
2
8.00.2
MySQL ODBC 8.0 Unicode Driver
8.00.2
ODBC Driver 17 for SQL Server
2017.
SQL Server
10.00
SQL Server Native Client 11.0
2011.
3
< 上一步(B)
完成
取消

6. 如图填写



password填你的mysql密码，database是本次设计使用的创建好的数据库。Data Source Name要记住，代码里要用，可以修改，记得代码里面一起改了。

数据库的设计

数据库结构

users表

Field	Type	Comment
ac	varchar(30) NOT NULL	账号
pw	varchar(20) NOT NULL	密码
whetheradmin	tinyint(1) NOT NULL	是否为管理员

userinfo表

	Field	Type	Comment
🔑	ac	varchar(30) NOT NULL	账号
	id	varchar(10) NULL	员工号
	name	varchar(20) NULL	姓名
	sex	varchar(2) NULL	性别
	bir	varchar(25) NULL	生日
	comp	varchar(30) NULL	公司
	dep	varchar(30) NULL	部门
	per	varchar(15) NULL	权限
	sta	varchar(30) NULL	状态

course表

	Field	Type	Comment
🔑	cno	int(11) NOT NULL	编号
	cname	varchar(70) NOT NULL	课程名
	intr	varchar(500) NULL	简介
	book	varchar(120) NULL	教材
	addr	varchar(30) NULL	地址
	signnum	int(11) NULL	报名人数
	maxnum	int(11) NULL	最大人数
	exptime	int(11) NOT NULL	截止日期

signrecord表

	Field	Type	Comment
🔑	sno	int(11) NOT NULL	记录序号
🔑	cno	int(11) NOT NULL	课程编号
	cname	varchar(70) NOT NULL	
🔑	ac	varchar(30) NOT NULL	账号
	name	varchar(20) NOT NULL	
	grade	int(11) NULL	成绩

代码

dbc.h代码

```
#ifndef DB_H
#define DB_H

```

```
#include<QSqlDatabase>
#include<QVector>


//数据库编码格式务必utf8，字段也要utf8 可以执行alter table `tablename` convert to character set utf8;
class dbc
{
public:
    static QSqlDatabase db;


public:
    dbc();
    //初始化
    static bool dbinit();
    //注册
    static bool signup(const QString ac,const QString pw);
    //登录
    static int signin(const QString ac,const QString pw);
    //找个人信息
    static bool findinfo(QVector<QString> &info);
    //更新个人信息数据
    static bool updinfo(QVector<QString> &info);
    //增加课程
    static bool addC(QVector<QString> &cinfo);
    //选课，传入课程号，增加数据
    static bool chooseC(int cno,QString ac);
    //添加管理员
    static bool addAdmin(QString ac);
};

#endif // DB_H

```

dbc.cpp代码如下

```
#include "dbc.h"
```

```
2
3  #include<QSqlError>
4  #include<QDebug>
5  #include<QSqlQuery>
6
7  QSqlDatabase dbc::db = QSqlDatabase::addDatabase("QODBC");
8
9  dbc::dbc()
10 {
11     //类外定义
12     //db = QSqlDatabase::addDatabase("QODBC");
13     db.setHostName("127.0.0.1");  //连接本地主机
14     db.setPort(3306);
15     db.setDatabaseName("ODBC2MySQL");
16     db.setUserName("root");
17     db.setPassword("你的密码");
18     bool ok = db.open();
19     if(!ok)
20         //#include<QSqlError>
21         //#include<QDebug>
22         qDebug()<<"error open database because"<<db.lastError().text();
23 }
24
25 bool dbc::dbinit()
26 {
27     db.setHostName("127.0.0.1");  //连接本地主机
28     db.setPort(3306);
29     db.setDatabaseName("ODBC2MySQL");
30     db.setUserName("root");
31     db.setPassword("你的密码");
32     bool ok = db.open();
33     if(!ok)
34         //#include<QSqlError>
35         //#include<QDebug>
36         qDebug()<<"error open database because"<<db.lastError().text();
37     return ok;
38 }
39
```

```cpp
bool dbc::signup(const QString ac, const QString pw)
{
    //#include<QSqlQuery>
    QSqlQuery result=db.exec();
    //往账户表插入数据
    QString query="insert into users(ac,pw,whetheradmin) values (\'"+ac+"\',\'"+pw+"\',\'"+"0\')";
    bool ok1=result.exec(query);
    //往信息表插入数据
    query="insert into userinfo(ac) values (\'"+ac+"\')";
    //qDebug()<<query;
    bool ok2=result.exec(query);
    return ok1&ok2;
}
//登录函数，返回值0是成功，1是未注册，2是密码错误,3是管理员登录成功
int dbc::signin(const QString ac, const QString pw)
{
    QString query="select * from users where ac=\'"+ac+"\'";
    //添加 #include<QSqlQuery>
    QSqlQuery result=db.exec(query);
    //注意!
    result.first();
    //没找到
    if(!result.isValid())
    {
        return 1;
    }
    if(result.value("pw").toString()!=pw)
    {
        return 2;
    }
    bool wa=result.value("whetheradmin").toBool();
    if(wa)
    {
        return 3;
    }
    else
        return 0;
```

```
77
78  }
79  //找个人信息
80  bool dbc::findinfo(QVector<QString> &info)
81  {
82      QString query="select * from userinfo where ac=\'"+info[0]+"\'";
83      //添加 #include<QSqlQuery>
84      QSqlQuery result=db.exec(query);
85      //注意!
86      result.first();
87      //info数组中已经有一个账号信息，不必再次读取写入，因此先指向结果集中的账
    号，进入while就跳过了。
88      if(!result.isValid())
89      {
90          return false;
91      }
92      for(int i=1;i<=8;i++)
93      {
94              QString name = result.value(i).toString();
95              info.push_back(name);
96      }
97      //for(auto a:info)    qDebug()<<a;
98      return true;
99  }
100 //更新个人信息
101 bool dbc::updinfo(QVector<QString> &info)
102 {
103     QString query=  "update userinfo set "
104                     "id=\'"+info[1]+"\',"
105                     "name=\'"+info[2]+"\',"
106                     "sex=\'"+info[3]+"\',"
107                     "bir=\'"+info[4]+"\',"
108                     "comp=\'"+info[5]+"\',"
109                     "dep=\'"+info[6]+"\',"
110                     "per=\'"+info[7]+"\',"
111                     "sta=\'"+info[8]+"\'"
112                     " where ac=\'"+info[0]+"\'";
113     //添加 #include<QSqlQuery>
```

```
114     //qDebug()<<query;
115     QSqlQuery result=db.exec();
116     bool ok = result.exec(query);
117     query="update signrecord set name=\'"+info[2]+"\' where
    ac=\'"+info[0]+"\'";
118     ok&=result.exec(query);
119     return ok;
120 }
121 //加课程
122 bool dbc::addC(QVector<QString> &cinfo)
123 {
124     QString query= "insert into course values(null,"
125                    "\'"
126             +cinfo[0]+"\',\'"
127             +cinfo[1]+"\',\'"
128             +cinfo[2]+"\',\'"
129             +cinfo[3]+"\',\'"
130             "0\',\'"
131             +cinfo[4]+"\',"
132             +cinfo[5]+")";
133     //qDebug()<<query;
134     QSqlQuery result=db.exec();
135     bool ok = result.exec(query);
136     return ok;
137 }
138 //选课
139 bool dbc::chooseC(int cno,QString ac)
140 {
141     QString query;
142     QSqlQuery result;
143     query="select * from signrecord where (cno="+QString::number(cno)+")
    and (ac=\'"+ac+"\')";
144     result=db.exec(query);
145     result.first();
146     //qDebug()<<query;
147     //找到了
148     if(result.isValid())
149     {
```

```
150         return false;
151     }
152
153
154     query="select cname from course where cno="+QString::number(cno);
155     //添加 #include<QSqlQuery>
156     result=db.exec(query);
157     result.first();
158     QString cname=result.value(0).toString();
159 //    qDebug()<<cname;
160     query="select name from userinfo where ac=\'"+ac+"\'";
161     //添加 #include<QSqlQuery>
162     result=db.exec(query);
163     result.first();
164     QString name=result.value(0).toString();
165 //    qDebug()<<name;
166     query= "insert into signrecord values(null,"
167                 +QString::number(cno)+","
168                  "\'"+cname+"\',"
169                  "\'"+ac+"\',"
170                  "\'"+name+"\',"
171                  "null)";
172     bool ok=result.exec(query);
173     query= "update course set signnum =signnum+1 where
    cno="+QString::number(cno);
174     ok&=result.exec(query);
175     return ok;
176 }
177 //添加管理员
178 bool dbc::addAdmin(QString ac)
179 {
180     QString query="select * from users where ac=\'"+ac+"\'";
181     QSqlQuery result;
182     result=db.exec(query);
183     result.first();
184     //qDebug()<<query;
185     if(result.isValid())
186     {
```

```
187         query="update users set whetheradmin=1 where ac=\'"+ac+"\'";
188         //]qDebug()<<query;
189         return result.exec(query);
190     }
191     else
192     {
193         return false;
194     }
195 }
```

函数功能在注释中有说明，实现方式也比较简单。不再赘述。

main函数

代码如下

```
1  #include "sign.h"
2  #include "dbc.h"
3  #include "windowforusers.h"
4  #include "windowforadmin.h"
5  #include <QApplication>
6
7  int main(int argc, char *argv[])
8  {
9      QApplication a(argc, argv);
10     dbc::dbinit();
11     sign window1;
12     WindowForUsers window2;
13     WindowForAdmin window3;
14     window1.show();
15     QObject::connect(&window1, &sign::sISuccess,
   &window2,&WindowForUsers::start);
16     QObject::connect(&window1, &sign::sISuccess, &window1,&sign::close);
17     QObject::connect(&window1, &sign::sISuccess,
   &window3,&WindowForAdmin::close);
18
```

```
    QObject::connect(&window1, &sign::adminSI, &window2,&WindowForUsers::close);
    QObject::connect(&window1, &sign::adminSI, &window1,&sign::close);
    QObject::connect(&window1, &sign::adminSI, &window3,&WindowForAdmin::start);

    return a.exec();
}

//ui不更新：
//直接找到对应的ui文件，右键，然后open command prompt with ->build environment
//然后输入uic mainwindow.ui -o ui_mainwindow.h，然后便可以更新ui_mainwindow.h，便可以更新最新的界面布局了。
//uic windowforusers.ui -o ui_windowforusers.h
//uic windowforadmin.ui -o ui_windowforadmin.h

```

总结

工程文件和数据库的建立文件我打包放在后面，有需要可以自行下载。

遇到的bug、问题，都在注释中有说明。请多看注释。

链接：https://pan.baidu.com/s/1vUst_N6eFj4iSHCXI3M8Pg
提取码：HSOL