作业管理系统文档

章敏捷, 计算机科学，3130102283

July 20, 2015

目　　录

章节 1

需求分析

以文件的形式存储数据，实现三类不同权限用户的操作

1.1 总体功能

管理员 创建、修改、删除、显示（list）教师帐号；教师帐户包括教师工号、教师姓名，教师用户以教师工号登录。

创建、修改、删除课程；绑定（包括添加、删除）课程与教师用户。课程名称以简单的中文或英文命名。

教师 对某门课程，创建或导入、修改、删除学生帐户，根据学号查找学生帐号；学生帐号的基本信息包括学号和姓名，学生使用学号登录。

发布课程信息。包括新建、编辑、删除、显示（list）课程信息等功能。

布置作业或实验。包括新建、编辑、删除、显示（list）作业或实验等功能。

查找、打印所有学生的完成作业情况。

学生 在教师添加学生账户后，学生就可以登录系统，并完成作业和实验。

基本功能：新建、编辑作业或实验功能；查询作业或实验完成情况。

章节 2

设计思路

总体采用MVC设计模式，将模型、视图用控制器连接减少耦合，方便扩展修改。

2.1 数据模型

所有数据全部存储在SQlite数据库中，以ORM方式封装所有数据库操作，采用学生、课程、实例、选课关系对象作为接口与控制器连接。

2.2 控制器

利用Qt的信号和槽作为控制器将用户视图的操作发送到模型进行处理，同时控制不同视图之间的切换。

2.3 视图

采用Qt的图形库，用QtDesigner绘制界面生成Ui类，利用不同的控件接受用户操作产生信号发送给控制器，同时接受控制器输出从模型中获取的数据。

章节 3

详细设计

3.1 数据库模型部分

数据库操作全部包括在Model.h/cpp中，每一张数据库表都对应一个实例类以及其相应的操作模型类。分别为：

- 学生
- 课程信息
- 开课信息
- 选课关系及成绩

数据库定义如下:

```
create table stu (
    [id]            integer PRIMARY KEY autoincrement,
    [stucid]        varchar(10) unique not null,
    [name]          varchar (20) not null,
    [sex]           int default 0,
    [age]           integer  not null,
    [grade]         integer  not null,
    [type]          int default 0,
    [deleteflag]    int default 0
);

create table class(
    [id]            integer PRIMARY KEY autoincrement,
    [classcid]      varchar(10) unique not null,
    [name]          varchar (20) not null,
    [intro]          TEXT
);
```

```
18
19 create table classinst (
20     [id]            integer PRIMARY KEY autoincrement,
21     [class_id]      integer ,
22     [term]          varchar(10),
23     foreign key(class_id) references class(id) on delete cascade
24  );
25
26 create table relation (
27     [id]            integer PRIMARY KEY autoincrement,
28     [stu_id]        integer ,
29     [inst_id]       integer ,
30     [score]         int,
31     foreign key(stu_id) references stu(id) on delete cascade ,
32     foreign key(inst_id) references classinst(id) on delete cascade
33 );
```

3.1.1 实例类

实例对象由模型类生成，在数据库中每个实例记录都带有自增的整形主键作为唯一识别标志。

数据库表中的每一列都对应实例类中的一个成员，同时实例类还包含一个record成员用于保存相应的数据库记录，当调用私有的getRecord()方法，就会将实例中的成员同步到数据库记录中，然后进行数据库操作。

每个实例对象都带有save()方法用于将数据写入数据库。数据库表中的外键则转化成实例对象的成员，可以直接调用，并且以递归的形式保存数据。

封装好的实例类接口不包含任何直接的数据库操作，使得与控制器部分的通信更加简单。

3.1.2 模型类

模型类主要负责数据库方面的处理，可以根据筛选条件返回单个或者多个(以list形式)实例对象；

3.2 控制器部分

控制器部分处理代码主要包含在视图类的private slots中，由视图中的控件发出信号来触发，主要方法为：

- select() 接受视图中用户的输入在数据库模型中选择相应的实例。
- setModel() 根据select()获取的实例生成表格发送给视图。

- save() 当用户触发信号时根据视图中的数据更新实例类的成员同时调用实例的save()方法写入数据库。

3.3 视图

项目中的视图可以简单分为表视图和单一视图；表视图负责输出批量查询和修改，单一视图负责新建或者编辑条目。

程序启动时显示主视图，由主视图调用其它子视图。

章节 4

其他

4.1 开发小结

4.1.1 版本控制

本轮开发采用git作为源代码管理工具，详细记录了开发的过程。虽然每次提交都有写备注但是并不十分符合标准，在正式开发中需要更加详细的描述每次提交的修改。[1]

4.1.2 重构

由于在初期开发过程中的失误导致多个变量命名不符合规范，所以使用了Visual Studio自带的代码重构工具对变量名进行了重命名。

4.1.3 内存泄漏

由于图形界面的特殊性(需要保持存在)，程序大量调用了new关键字从堆中申请内存，从代码中可以看到我并没有在析构函数中做delete处理，但是这并没有对性能造成影响，也没有产生内存泄漏，这得益于Qt的内存管理机制，所有定义了Q_OBJECT宏的对象在构造时都包含了parent指针，Qt的内存管理机制会在父对象被回收时自动回收其子对象的内存。

但是这类机制也会导致一些问题，比如当父对象早于子对象被回收，就可能导致野指针的出现。

[1]本项目git地址：https://github.com/zmj1316/Students-Score-Manage-Sys