科大讯飞 iFLYTEK

【千古江山】

概要设计说明书

文档拟稿：【范天明】

建立日期：2017 年 7 月

文档审核：

审核日期：

文档页数：

当前版本：1.0

讯飞教育

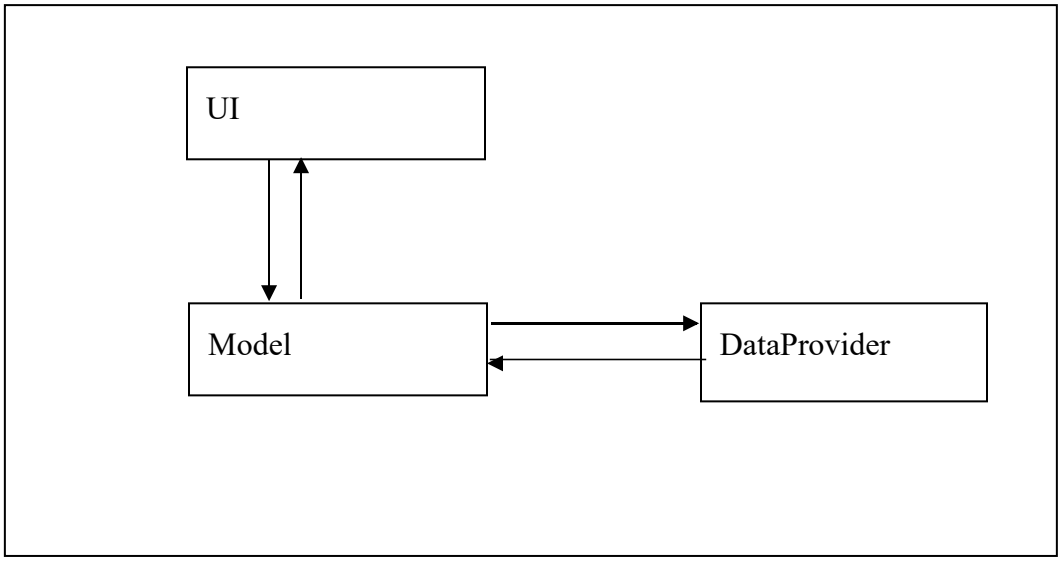
更新记录

审核记录

1. 概述

本文档用于说明千古江山项目的概要设计，旨在帮助开发人员了解系统的整体架构，关键技术和关键模块的设计，帮助开发人员了解系统的设计概况，为详细设计做铺垫。

2. 系统逻辑结构图



比如：以上图的 UI 页面命名,创建普通用户页面，命名为：FrmUser.h

用户列表页面，命名为：FrmUserList.h

用户修改页面，命名为：FrmUserModify.h

控件，如登录按钮，命名为：btnLogin

用户名输入框，命名为：txtUserName

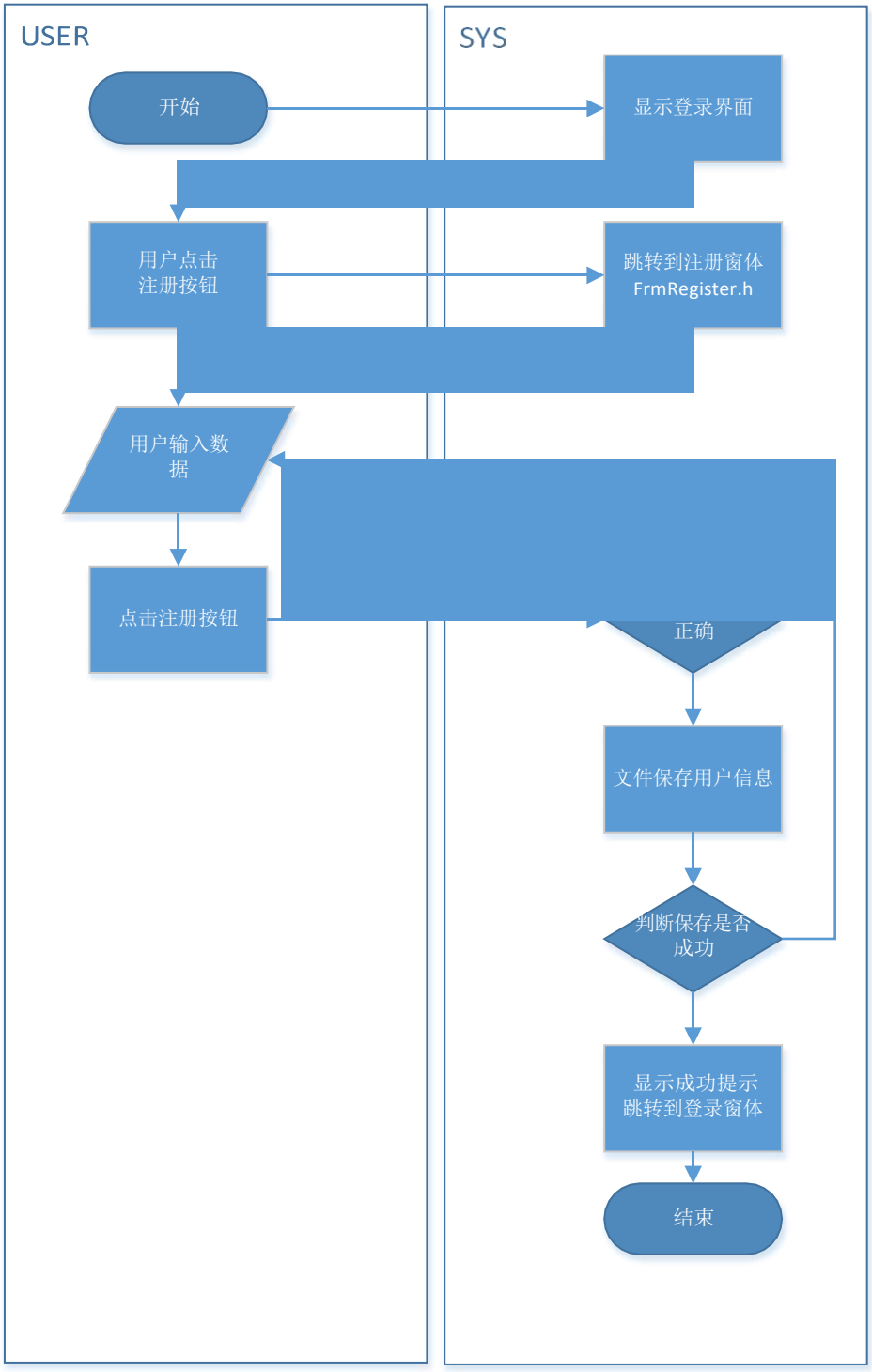
其他控件 Label，命名为：lblXXXX

两个单词的控件取首字母小写如 ListView，命名为：lvXXXX、ImageView，命名为：ivXXX

而 Model 命名，例如，用户，可命名为：User.h

3. 代码处理逻辑

本项目代码分层为 UI 层和 Model 层，UI 层负责数据的页面显示并处理数据并进行相关业务逻辑处理，然后进行页面跳转。以注册为例，整个逻辑处理流程如下图：



4. 关键技术

以下面列出一些常用关键技术的实现方式供参考。

4.1 动态创建控件

见视频

4.2 多线程

在编写大型程序时，将所有任务都放在一条线程上来完成并不是一个明智的选择，因为程序一旦卡死在某个环节或是发生了死循环，整个程序都会废掉。所以适当的用多线程有利于程序的良好运行。

QThread的常见特性：

run()是线程的入口，就像main()对于应用程序的作用。QThread中对run()的默认实现调用了exec()，从而创建一个QEventLoop对象，由其处理该线程事件队列（每一个线程都有一个属于自己的事件队列）中的事件。简单用代码描述如下：

```
01. int QThread::exec()
02. {
03. //...
04.     QEventLoop eventLoop;
05.     int returnCode = eventLoop.exec();
06. //...
07.     return returnCode;
08. }
09.
10. int QEventLoop::exec(ProcessEventsFlags flags)
11. {
12. //...
13.     while (!d->exit) {
14.         while (!posted_event_queue_is_empty) {
15.             process_next_posted_event();
16.         }
17.     }
18. //...
19. }
```

这是Qt4.7及以后版本推荐的工作方式。其主要特点就是利用Qt的事件驱动特性，将需要在次线程中处理的业务放在独立的模块（类）中，由主线程创建完该对象后，将其移交给指定的线程，且可以将多个类似的对象移交给同一个线程。在这个例子中，信号由主线程的QTimer对象发出，之后Qt会将关联的事件放到worker所属线程的事件队列。由于队列连接的作用，在不同线程间连接信号和槽是很安全的。

说说connect最后一个参数，连接类型：

1)自动连接(AutoConnection)，默认的连接方式，如果信号与槽，也就是发送者与接受者在同一线程，等同于直接连接；如果发送者与接受者处在不同线程，等同于队列连接。

2)直接连接(DirectConnection)，当信号发射时，槽函数立即直接调用。无论槽函数所属对象在哪个线程，槽函数总在发送者所在线程执行。

3)队列连接(QueuedConnection)，当控制权回到接受者所在线程的事件循环时，槽函数被调用。槽函数在接受者所在线程执行。

```cpp
#include <QtCore>
class Worker : public QObject
{
    Q_OBJECT
private slots:
    void onTimeout()
    {
        qDebug()<<"Worker::onTimeout get called from?: "<<QThread::currentThreadId();
    }
};
```

```cpp
#include "main.moc"
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    qDebug()<<"From main thread: "<<QThread::currentThreadId();

    QThread t;
    QTimer timer;
    Worker worker;

    QObject::connect(&timer, SIGNAL(timeout()), &worker, SLOT(onTimeout()));
    timer.start(1000);

    worker.moveToThread(&t);

    t.start();

    return a.exec();
}
```

4.3 MVC框架

Qt包含一组使用模型/视图结构的类，可以用来管理数据并呈现给用户。这种体系结构引入的分离使开发人员更灵活地定制项目，并且提供了一个标准模型的接口，以允许广泛范围的数据源被使用到到现有的视图中。

MVC设计模式

起源于smalltalk的一种与用户界面设计相关的设计模式。

作用：有效的分离数据和用户界面。

组成：模型model（表示数据）、视图view（表示用户界面）、控制controller（定义用户在界面上的操作）。

interView框架

Qt的MVC

区别：将视图与控制结合在一起，同时添加了代理delegate能够自定义数据条目item的显示与编辑方式。

组成：模型model（表示数据）、视图view（表示用户界面）、代理delegate（自定义数据条目item的显示与编辑方式）。

模型与视图结构：

模型与数据通信，并提供接口

视图从模型中获取数据条目索引

代理绘制数据条目

通信方式：信号&槽