个性化微博 Mac 客户端

程序设计文档

姓名：

班级：2014211302

学号：2014212070

目录

一、程序设计总体说明

1.1 程序总体说明

本次程序设计实践课，利用新浪微博开放的 API 接口，使用 MVC 的框架模式，设计了一个 Mac 风格的新浪微博客户端。在可以使用的 API 基础上，实现了一个微博客户端应有的全部功能。

程序使用 C++进行编程，使用 Qt 框架实现图形界面。

1.2 功能清单

1.2.1 登录微博
1.2.2 注销当前用户
1.2.3 浏览登录用户的微博时间线
1.2.4 浏览@登录用户的微博时间线
1.2.5 浏览登录用户收到的评论
1.2.6 浏览登录用户发出的评论
1.2.7 浏览登录用户发布的微博
1.2.8 浏览转发登录用户微博的时间线
1.2.9 查看登录用户的个人信息
1.2.10 查看登录用户的关注人列表
1.2.11 查看登录用户的粉丝列表
1.2.12 发布文字微博
1.2.13 发布带有图片的微博
1.2.14 查看某一特定微博及其评论
1.2.15 转发某一特定微博
1.2.16 回复某一特定微博
1.2.17 回复某条评论
1.2.18 删除登录用户自己发布的微博
1.2.19 删除登录用户自己发出的评论
1.2.20 清除登录用户产生的图片缓存
1.2.21 查看微博中附带的图片
1.2.22 查看微博中包含的网页链接

1.3 设计说明

在设计程序整体结构时，采用了 MVC 的设计框架。程序整体结构如下图所示。

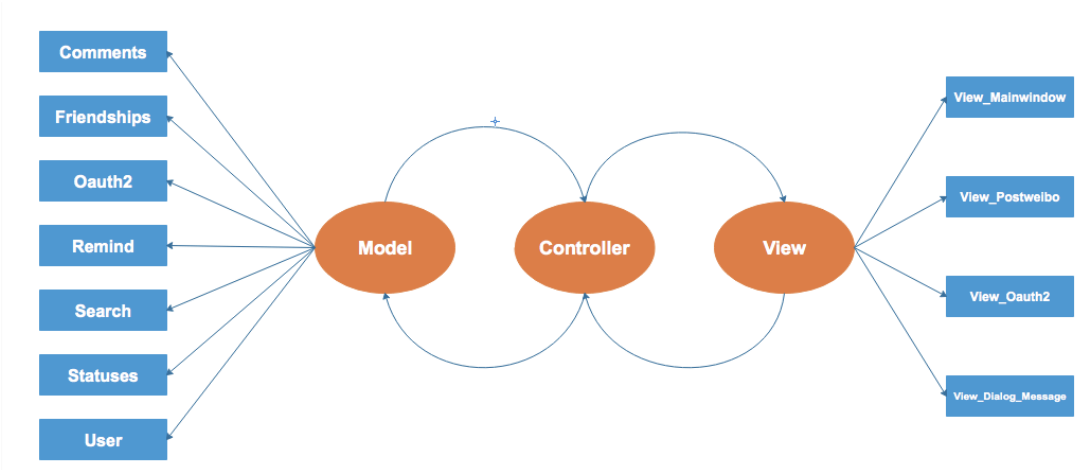

图 1 程序整体结构示意图

二、Model

2.1 Model 总体说明

在 MVC 的框架模式中，Model 指模型。在设计的微博客户端中，Model 指 weibo 类。程序使用 weibo 类完成与新浪后台的数据交互，并将数据传递给 Controller，通过 View 显示出来。

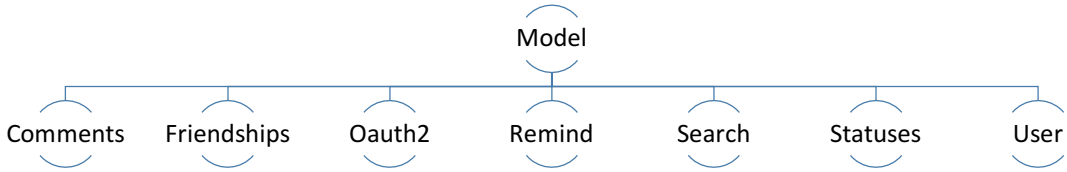

图 2 Model 模块示意图

2.2 Model 与 Controller 交互机制

Model 与 Controller 通过 Qt 内置的“信号-槽”机制实现交互。

Controller 中 New 出一个 weibo 类的对象，通过调用公有接口发出网络请求。

Model 通过发射 signal 通知 Controller 有数据返回，Controller 中 connect 信号和槽函数，获取返回的数据。

图 3 Model 与 Controller 交互示意图

2.3 Model（Weibo 类）的模块划分

2.3.1 Weibo_Oauth2

2.3.1.1 模块说明

该模块用于实现用户授权登录，获取登录用户的 access_token，使用该 access_token 访问各类接口。

2.3.1.2 Oauth2 认证机制

关于 OAuth2.0 协议的授权流程可以参考下面的流程图，其中 Client 指第三方应用，Resource Owner 指用户，Authorization Server 是新浪的授权服务器，Resource Server 是 API 服务器。

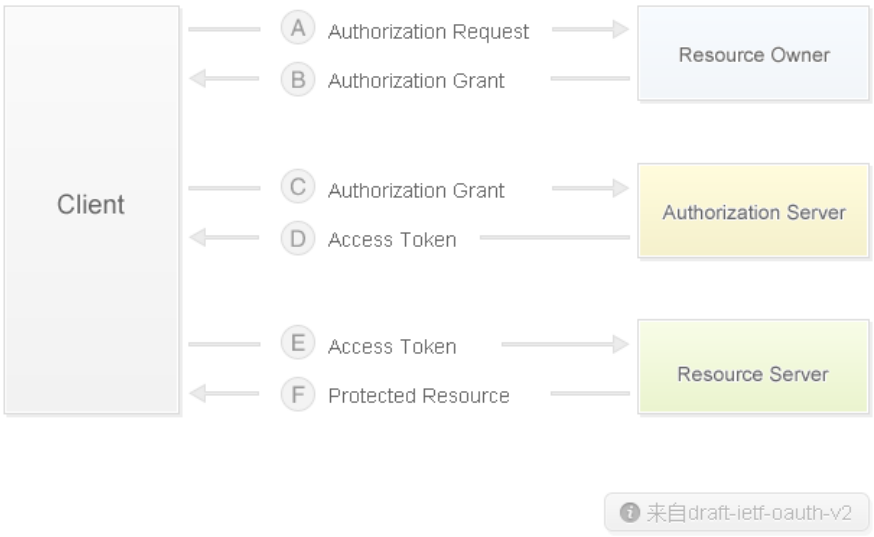

图 4 Oauth2 认证机制示意图

2.3.1.3 API 说明

（1）oauth2_authorize

接口文档：http://open.weibo.com/wiki/Oauth2/authorize

（2）oauth2_access_token

接口文档：http://open.weibo.com/wiki/OAuth2/access_token

（3）oauth2_get_token_info

接口文档：http://open.weibo.com/wiki/Oauth2/get_token_info

2.3.2 Weibo_Friendships

2.3.2.1 模块说明

该模块用于获取用户的关注人列表和粉丝列表。

2.3.2.2 API 说明

（1）friendships_friends
接口文档：http://open.weibo.com/wiki/2/friendships/friends
（2）friendships_followers
接口文档：http://open.weibo.com/wiki/2/friendships/followers

2.3.3 Weibo_Comments

2.3.3.1 模块说明

该模块用于获取用户的评论时间线，包括用户发出的评论、收到的评论、收到的@等。

2.3.3.2 API 说明

（1）comments_by_me
接口文档：http://open.weibo.com/wiki/2/comments/by_me
（2）comments_to_me
接口文档：http://open.weibo.com/wiki/2/comments/to_me
（3）comments_timeline
接口文档：http://open.weibo.com/wiki/2/comments/timeline
（4）comments_mentions
接口文档：http://open.weibo.com/wiki/2/comments/mentions

2.3.4 Weibo_Remind

2.3.4.1 模块说明

该模块用于监控是否有新的微博。若有新的微博，则通知 Controller 显示提示信息。

2.3.4.2 API 说明

（1）remind_unread_count
接口文档：http://open.weibo.com/wiki/2/remind/unread_count

2.3.5 Weibo_Search

2.3.5.1 模块说明

该模块用于搜索微博内容。由于权限原因，目前只能使用搜索用户接口，由于接口返回的数据过少，并没有在界面上显示出搜索结果。

2.3.5.2 API 说明

（1）search_suggestions_users
接口文档：http://open.weibo.com/wiki/2/search/suggestions/users

2.3.6 Weibo_Statuses

2.3.6.1 模块说明

该模块用于处理微博内容。通过该模块可以实现刷新用户关注人的时间线，刷新用户发布微博的时间线，显示某条微博及其评论，转发微博，删除微博，发布文字微博，发布带有图片的微博等功能。

2.3.6.2 API 说明

（1）statuses_friends_timeline
接口文档：http://open.weibo.com/wiki/2/statuses/friends_timeline
（2）statuses_user_timeline
接口文档：http://open.weibo.com/wiki/2/statuses/user_timeline
（3）statuses_mentions
接口文档：http://open.weibo.com/wiki/2/statuses/mentions
（4）statuses_show
接口文档：http://open.weibo.com/wiki/2/statuses/show
（5）statuses_repost
接口文档：http://open.weibo.com/wiki/2/statuses/repost

（6）statuses_destroy

接口文档：http://open.weibo.com/wiki/2/statuses/destroy

（7）statuses_upload

接口文档：http://open.weibo.com/wiki/2/statuses/upload

（8）statuses_update

接口文档：http://open.weibo.com/wiki/2/statuses/update

2.3.7 Weibo_Users

2.3.7.1 模块说明

该模块用于获取登录用户的个人信息，包括昵称、位置、性别、关注人数、粉丝数等等。

2.3.7.1 API 说明

（1）user_show

接口文档：http://open.weibo.com/wiki/2/users/show

2.4 Json 解析

微博 API 返回的数据均为 Json 格式的字节流。程序使用 Qt 自带的 QJsonParseError、QJsonDocument、QJsonObject、QJsonValue、QjsonArray 五个类实现 Json 的解析。

三、View

3.1 总体说明

在 MVC 的框架模式中，View 指视图。在设计的微博客户端中，View 包含若干个类，这些类分别用于实现不同的视图显示。

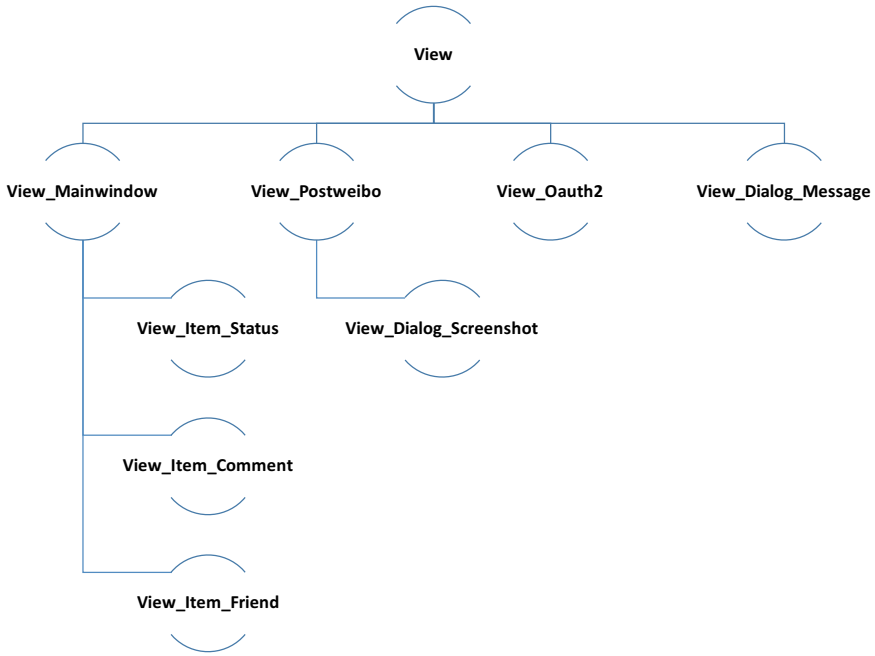

图 5 View 模块结构示意图

3.2 View 与 Controller 交互机制

View 与 Controller 通过 Qt 内置的“信号-槽”机制实现交互。

Controller 中 New 出必要的 View，通过公有的函数接口实现对 View 的控制。

View 通过 Signal 发出请求，通过在 Controller 中 connect 的槽函数实现请求的响应。

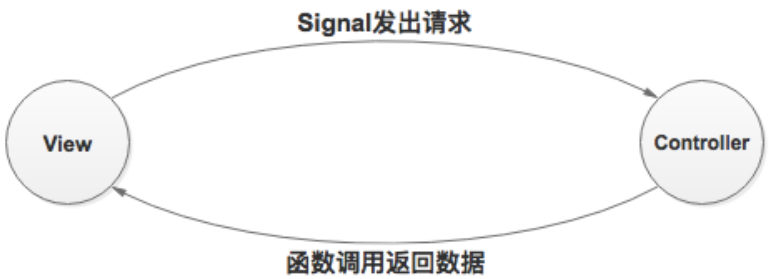

图 6 View 与 Controller 交互机制示意图

3.3各视图类说明

3.3.1 View_Mainwindow

3.3.1.1 作用说明

View_Mainwindow 类实现了微博的主界面。用户能在该类对应的视图中浏览所有的时间线，并查看任意的微博。

该类的具体内容请查看相关文件。

3.3.1.2 技术细节

（1）如何显示各种列表

该类通过 QListWidget 控件和自定义的 View_Item_Status、View_Item_Comment、View_Item_Friend 实现列表显示。每一个表项的内容由 View_Item_Status、View_Item_Comment、View_Item_Friend 封装，QListWidget 无需考虑，只需要调用接口进行插入。

（2）如何唤醒其他界面

该类通过 Signal 通知 Controller，使用槽函数显示其他界面，例如 View_Postweibo 等。

（3）如何实现刷新时列表滑动

该类通过 QEventLoop 类和 QListWidget 的 move 函数实现 List 的下滑、上滑等动画效果。

（4）如何实现 Gif 转动

该类通过 QMovie 实现 Gif 转动。

（5）如何实现界面部分区域半透明

该类通过 Qpaint 事件实现界面左侧灰色条形区域的半透明效果。

（6）如何实现淡化显示和淡化隐藏

该类通过 QPropertyAnimation 类实现淡化显示和淡化隐藏。

（7）如何实现圆形头像

该类通过 Qpixmap 类实现圆形头像，利用了遮罩层及 setMask 函数。

3.3.2 View_Oauth2

3.3.2.1 作用说明

View_Oauth2 类实现了微博登录认证的功能。在使用客户端前，用户需要通过该视图输入账号密码，与新浪后台完成认证过程。

3.3.2.2 技术细节

（1）如何载入登录网页

利用 Qt 中的 QWebView 控件实现网页载入。

（2）如何获取重定位 URL 中的 code

利用 QWebView 类中 urlChanged(QString)信号获取重定位后得到的 URL，通过分析字符串得到 code。

3.3.3 View_Postweibo

3.3.3.1 作用说明

View_Postweibo 类用于实现发布微博、评论、回复评论等功能。

3.3.3.2 技术细节

（1）如何显示剩余字数

通过文字编辑控件 QTextEdit 的 textChanged 信号，实时统计当前编辑区的字数，做减法即可获得剩余字数。

（2）如何实现选择本地图片

通过 QFileDialog 类实现本地图片的选择。

（3）如何实现无边框

使用 setWindowFlags(Qt::FramelessWindowHint);实现窗口无边框。

（4）如何实现对话框跟随鼠标拖动

通过 mousepressevent 和 mousemoveevent 事件，监视鼠标动作。当发生拖动时，将窗口的全局坐标设为鼠标的全局坐标，实现跟随鼠标。

3.3.4 View_Dialog_Message

3.3.4.1 作用说明

View_Dialog_Messgae 类用于显示提示、警告等信息。

3.3.5 View_Dialog_Screenshot

3.3.5.1 作用说明

View_Dialog_Screenshot 类用于实现截图功能。

3.3.5.2 技术细节

（1）截屏实现机制

利用 Qt 的 QRubberBand 类。当点击截图按钮时，先保存下整个桌面的图片截图，并使用 QLabel 将该截图铺满整个屏幕。之后利用 Qrubberband 类实现矩形区域选择，保存矩形区域的内容，完成截图。

3.3.6 View_Item_Status

3.3.6.1 作用说明

View_Item_Status 类用于显示一条微博的内容。包括文字内容、头像信息、相关按钮、图片、链接等等。与 View_Mainwindow 中的 QListWidget 一起使用，显示当前用户的各种时间线。

3.3.6.2 技术细节

（1）如何实现界面大小对微博内容自适应

通过设置 QLabel 为自动换行模式，并配合 adjustSize()函数实现文本内容自适应。根据文本的大小计算其他文本或图片的位置，实现自适应布局。最后控件整体调用 adjustSize()函数，实现整体大小自适应。自适应前通过 designer 锁定了控件的宽度。

（2）如何识别微博内容中的用户名和网页链接

通过手写的词法分析器实现了用户名和网页链接的识别。

具体细节请看 QString lexicalAnalysis(QString str);函数。

（3）如何实现用户名和网页链接的高亮及可点击

通过 HTML 语法实现了用户名和网页链接的高亮和可点击。

具体细节请看 QString lexicalAnalysis(QString str);函数。

（4）如何通过网页链接调用外部浏览器

通过调用函数 QDesktopServices::openUrl(QUrl(link))实现调用外部浏览器。

（5）如何实现图片可点击及浏览

通过 mousepressevent 追踪鼠标点击位置，通过坐标计算，判断是否点击到某个图片。若点击到图片，通过 QWebView 加载图片的网页链接，实现图片的显示浏览。

3.3.7 View_Item_Comment

3.3.7.1 作用说明

View_Item_Comment 类用于显示一条评论的内容。包括文字内容、头像信息、相关按钮等等。与 View_Mainwindow 中的 QListWidget 一起使用，显示评论列表。

3.3.7.2 技术细节

参考 View_Item_Status

3.3.8 View_Item_Friend

3.3.8.1 作用说明

View_Item_Friend 类用于显示一条用户的信息。包括个人简介、头像信息、相关按钮等等。与 View_Mainwindow 中的 QListWidget 一起使用，显示关注人列表和粉丝列表。

3.3.8.2 技术细节

参考 View_Item_Status

3.3.9 View_Traysystem

3.3.9.1 作用说明

View_Traysystem 属于 View_Mainwindow 类，用于实现托盘系统。

3.3.9.2 技术细节

（1）如何实现托盘系统

利用 Qt 中 QsystemTrayIcon、Qaction、Qmenu 三个类实现了托盘系统。

（2）如何实现新微博提醒

程序利用监视线程实现对未读信息的监视，一旦发现未读信息，Controller 提醒 View，托盘图标的颜色发生变化，表示此时有未读信息。

四、Controller

4.1 如何控制 View 和 Model 的交互

Controller 通过函数调用，以及“信号-槽”机制实现 View 和 Model 的交互。

当用户点击 View 的相关按钮时，View 发出 Signal，Controller 捕获 Signal 后，通过 weibo 对象调用对应的函数接口，发送出网络请求。当网络请求返回，Json 被解析后，weibo 对象发出返回数据的 Signal，同样的，Signal 被 Controller 捕获，调用 View 的相应对象的函数接口，最终在界面上显示出内容。

4.2 配置信息

为了避免每次退出程序后都需要重新登录的问题，程序保存了用户的配置信息，即用户的登录时间，access_token 值，access_token 生存周期，用户 uid 等内容。再次运行客户端时先查看配置文件信息，若 access_token 未过期，则直接登录，无需再次输入账号密码；否则需要再次通过 View_Oauth2 界面登录认证。

读写配置文件内容的代码见 controller_filemanager.cpp。

4.3 数据库

为了能在再次登录时回复所有的信息（包括所有的时间线、用户的个人信息等），Controller 通过数据库保存所有必要的内容。程序中采用 Qt 自带的数据库实现数据保存。

建立、读写数据库的代码见 controller_filemanager.cpp。

4.4 图片缓存

微博 API 返回的 Json 中，并没有图片的数据信息。为了显示包括头像在内的所有图片，需要对图片进行缓存操作。在调用 View 的函数接口前，Controller 先查看数据中的 URL，下载并缓存图片。

相关函数：

void Controller::imageDownloader(QString url, QString fileName);

QString Controller::urlChangeThumbnailToBmiddle(QString url);

void Controller::cacheImages(std::list<statusInfoT> &statusesList);

4.5 恢复时间线

用户在无需登录的情况下运行客户端，需要对时间线等信息进行恢复。恢复时间线所需要的数据记录在上文提到的数据库中。客户端开运行时，读取数据库内容，调用 View 中对应的函数，实现时间线恢复。

相关函数：

std::list<statusInfoT> Controller::recoverTimeline(QString table);

4.6 网络请求互斥

为了避免高频率网络请求导致 Json 解析出错，Controller 对所有的网络请求进行了互斥设置。Controller 通过一个名为 inHttping 的 bool 变量（私有成员变量）进行控制。当存在网络请求仍未返回数据时，Controller 不再接受任何来自 View 的网络请求。

4.7 多线程

为了实现未读消息提醒，程序另外添加了一个监视线程，该监视线程由 Controller 控制。在该监视线程中利用 QTimer 类设置了一个定时器，以一固定频率触发槽函数。槽函数则不断调用 remind_unread_count 接口，查看未读消息信息。

相关函数：

void Controller::unreadWatcher();