**程序界面设计**

**（文档内代码仅供参考）**

1. **界面设计**

该界面用户可以申请号码，然后注册自己的基本信息，若注册成功就可得到一个WECHAT号码，并根据此号码加上自己在注册基本资料时注册的密码就可以实现用户的登录，具体操作见下列各图5.1到图5.14所示：



图5.1 用户登录运行界面

具体实现：在Visual Studio.NET2005的Windows应用程序里面，首先创建一个名为“MyWECHAT用户登录”的主窗体，然后按照需要的功能拖控件，

如果用户名和密码为空:

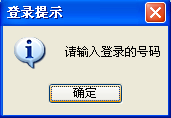


图5.2 登录提示界面 1

MessageBox.Show("请输入登录的号码", "登录提示", MessageBoxButtons.OK, MessageBoxIcon.Information);

txtLoginId.Focus();

如果密码有错：

则由图5.3所示：

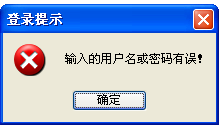


图5.3 登录提示界面 2

具体实现代码如下：

MessageBox.Show("请输入密码", "登录提示", MessageBoxButtons.OK, MessageBoxIcon.Information);

如果用户名和密码都对:

return false;

}

return true;

}



图5.4 申请号码的运行界面

创建“申请号码”主窗体，此窗体由三部分组成，具体操作同上，部分涉及数据库信息的代码如下：

// 查询星座用的sql语句

string sql = "SELECT Star FROM Star";

// 修改查询语句，查询血型

sql = "SELECT BloodType FROM BloodType";

// 获得血型的Id

bloodTypeId = GetBloodType();

sql = string.Format("INSERT INTO Users (LoginPwd, NickName, Sex, Age, Name, StarId, BloodTypeId) values ('{0}','{1}','{2}',{3},'{4}',{5},{6})",

txtLoginPwd.Text.Trim(), txtNickName.Text.Trim(), sex, int.Parse(txtAge.Text.Trim()), txtName.Text.Trim(), starId, bloodTypeId); // 获得星座的Id

starId = GetStarId();

sql = string.Format("INSERT INTO Users (LoginPwd, NickName, Sex, Age, Name, StarId) values ('{0}','{1}','{2}',{3},'{4}', {5})",

txtLoginPwd.Text.Trim(), txtNickName.Text.Trim(), sex, int.Parse(txtAge.Text.Trim()), txtName.Text.Trim(),starId);

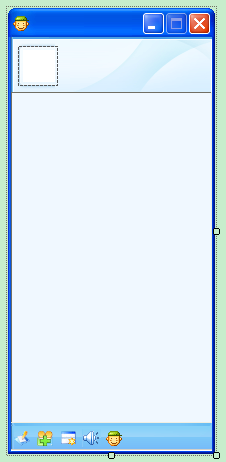


图5.5 登录后运行界面

此界面具体功能未能实现。

1. **界面具体操作**

打开用户登录，申请号码注册**：**



图5.6 用户登录

点击申请号码以后，注册自己的基本信息，包括昵称、年龄、性别、密码、真实姓名、星座和血型。



图5.7 用户注册界面

注册成功后，将获得提示。

如图5.8所示：

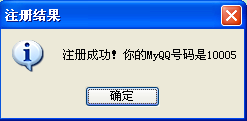


图5.8 注册成功提示

注册成功以后，用申请的号码和自己的密码进行登录。

如图5.9所示：



图5.9 用户登录界面

如果用户名和密码有错则会:

如图5.10所示：

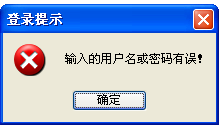


图5.10 用户登录出错

具体实现代码如下：

MessageBox.Show("请输入密码", "登录提示", MessageBoxButtons.OK, MessageBoxIcon.Information);

如果用户名和密码都对:

return false;

}

return true;

}

其效果图如图5.11所示：



图5.11 用户登录成功后的主窗体

好友列表——第三方控件运用了第三方SideBar

// 命名空间

using Aptech.UI;

// 添加组

sbFriends.AddGroup("我的好友");

sbFriends.AddGroup("陌生人");

// 添加项

SbItem item = new SbItem((string)dataReader["NickName"],

sbFriends.Groups[0].Items.Add(item);

如果要添加好友的话:

将通过三步来实现。

1. 查找 如图5.12所示
2. 添加 如图5.13所示
3. 反馈成功信息 如图5.14所示

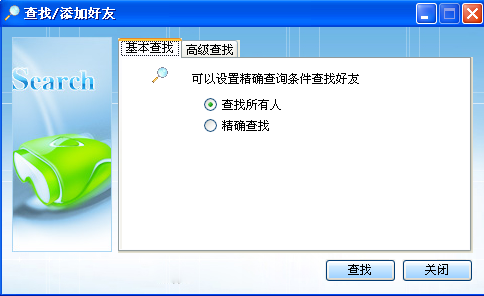


图5.12 查找好友

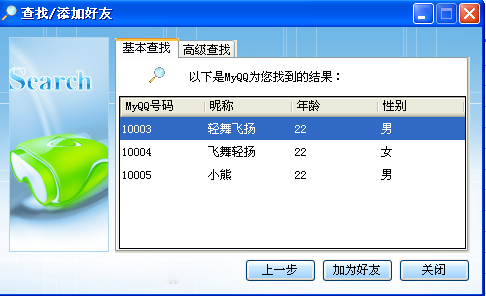


图5.13 好友列表



图5.14 加载好友已经存在

// 添加好友

private void btnAdd\_Click(object sender, EventArgs e)

{

int friendshipPolicyId = -1; // 对方的好友策略

int friendId = GetSelectedFriendId(); // 获得选中的好友的Id

// 是否有可添加的好友

if (friendId == -1)

{

MessageBox.Show("请选择一个好友！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

else if (friendId == UserHelper.loginId)

{

MessageBox.Show("不能加自己为好友！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

else if (HasAdded(friendId)) // 确认是否需要添加（是否已经是好友了）

{

MessageBox.Show("对方已在你的好友列表中！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

// 确认对方能否被加为好友

friendshipPolicyId = GetFriendshipPolicyId(friendId);

if (friendshipPolicyId == 3) // 3表示不允许任何人添加，在S2可以使用常量或枚举

{

MessageBox.Show("对方不允许任何人加他为好友！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

else if (friendshipPolicyId == 2) // 2：表示需要身份验证

{

int result = SendRequest(friendId); // 发验证消息

if (result == 1)

{

MessageBox.Show("对方需要身份验证才能被加为好友，已发出请求！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("添加失败，请稍候再试！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else if (friendshipPolicyId == 1)

// 1：表示允许任何人添加为好友

{

// 执行添加操作

int result = AddFriend(friendId);

if (result == 1)

{

MessageBox.Show("添加成功，请刷新好友列表！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("添加失败，请稍候再试！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}