WebService.asmx

using UserControl;

using DRemindControl;

using Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Services;

using System.Web.Services.Protocols;

using System.Xml.Serialization;

using EventControl;

using ListControl;

using LeavesControl;

namespace TimenoteDBwebservice

{

/// <summary>

/// WebService1 的摘要说明

/// </summary>

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

[System.ComponentModel.ToolboxItem(true)]

// 若要允许使用 ASP.NET AJAX 从脚本中调用此 Web 服务，请取消注释以下行。

//[System.Web.Script.Services.ScriptService]

public class WebService1 : System.Web.Services.WebService

{

UserManager UM = new UserManager();

DRemindManager DM = new DRemindManager();

EventManager EM = new EventManager();

ListManager LIM = new ListManager();

LeavesManager LEM = new LeavesManager();

[WebMethod(Description = "加载用户信息")]

//[XmlInclude(typeof(BeanUserInformation))]

public BeanUserInformation LoadUser(String useremail)

{

BeanUserInformation c = UM.LoadUser(useremail);

return c;

}

/\*

[WebMethod(Description = "邮箱判重验证")]

public Boolean EmailVerify(String useremail)

{

return UM.EmailVerify(useremail);

}\*/

[WebMethod(Description = "用户注册")]

public Boolean CreateUser(String useremail, String userpwd, String username)

{

Boolean a = UM.CreateUser(useremail, userpwd, username);

return a;

}

[WebMethod(Description = "用户激活")]

public void DeleteStopDate(String useremail)

{

UM.DeleteStopDate(useremail);

}

[WebMethod(Description = "用户注册信息回滚")]

public void UserRollBack(string useremail)

{

UM.UserRollBack(useremail);

}

[WebMethod(Description = "用户邮箱修改")]

public Boolean UpdateEmail(string useremail, int userid)

{

return UM.UpdateEmail(useremail, userid);

}

[WebMethod(Description = "用户昵称修改")]

public Boolean UpdateName(string username, int userid)

{

return UM.UpdateName(username, userid);

}

[WebMethod(Description = "用户密码修改")]

public Boolean UpdateUserpassward(string userpassword, int userid)

{

return UM.UpdateUserpassward(userpassword, userid);

}

[WebMethod(Description = "验证码重发并更新验证码")]

public void SendAuthCode(string useremail)

{

UM.SendAuthCode(useremail);

}

[WebMethod(Description = "上传头像")]

public Boolean SaveHead(string head, int userid)

{

return UM.SaveHead(head,userid);

}

[WebMethod(Description = "下载头像")]

public string GetHead(int userid)

{

return UM.GetHead(userid);

}

/\*[WebMethod(Description = "查找提醒表最大id")]

public int FindMaxId()

{

return DM.FindMaxId();

}\*/

[WebMethod(Description = "新建事务")]

public Boolean NewEvent(string eventname, int eventpriority, int listid, int userid, string eventdate)

{

if (eventdate.Equals("")) eventdate = "0001-01-01T00:00:00";

return EM.NewEvent(eventname, eventpriority, listid, userid, DateTime.Parse(eventdate));

}

[WebMethod(Description = "删除事务")]

public Boolean DeleteEvent(int eventid)

{

return EM.DeleteEvent(eventid);

}

[WebMethod(Description = "修改事务")]

public Boolean UpdateEvent(string eventname, int eventpriority, int listid, string eventdate, int leaveseventsign, string eventnote, int eventid)

{

if (eventdate.Equals("")) eventdate = "0001-01-01T00:00:00";

return EM.UpdateEvent(eventname, eventpriority, listid, DateTime.Parse(eventdate), leaveseventsign, eventnote, eventid);

}

[WebMethod(Description = "修改事务优先级")]

public Boolean UpdateEventPriority(int eventpriority, int eventid)

{

return EM.UpdateEventPriority(eventpriority, eventid);

}

//[XmlInclude(typeof(List<BeanEventInformation>))]

//[XmlInclude(typeof(BeanEventInformation))]

[WebMethod(Description = "按日期order返回事务")]

public List<BeanEventInformation> AllEvent(int userid)

{

return EM.AllEvent(userid);

}

[WebMethod(Description = "按日期和优先级order返回事务")]

public List<BeanEventInformation> AllEventByPro(int userid)

{

return EM.AllEventByPro(userid);

}

[WebMethod(Description = "修改事务状态至已完成")]

public Boolean EventComplete(int eventid)

{

return EM.EventComplete(eventid);

}

[WebMethod(Description = "修改事务状态至未完成")]

public Boolean EventUnComplete(int eventid)

{

return EM.EventUnComplete(eventid);

}

[WebMethod(Description = "设置事务闹钟Pid")]

public Boolean UpdatePid(int eventid, int pid)

{

return EM.UpdatePid(eventid, pid);

}

[WebMethod(Description = "新建清单")]

public Boolean NewList(string listname, int userid)

{

return LIM.NewList(listname, userid);

}

[WebMethod(Description = "载入清单")]

public List<BeanListInformation> AllList(int userid)

{

return LIM.AllList(userid);

}

[WebMethod(Description = "修改清单")]

public Boolean UpdateList(string listname, int listid)

{

return LIM.UpdateList(listname, listid);

}

[WebMethod(Description = "删除清单")]

public Boolean DeleteList(int listid)

{

return LIM.DeleteList(listid);

}

[WebMethod(Description = "新增提醒")]

public Boolean NewDRemind(int eventid, int userid)

{

return DM.NewDRemind(eventid, userid);

}

[WebMethod(Description = "载入提醒设置")]

public BeanDRemindInformation LoadDefaultSet(int userid, int eventid)

{

return DM.LoadDefaultSet(userid, eventid);

}

[WebMethod(Description = "是否振动")]

public Boolean UpdateVib(int userid, int eventid, int dremindvib)

{

return DM.UpdateVib(userid, eventid, dremindvib);

}

[WebMethod(Description = "修改叶子计时时长")]

public Boolean UpdateLeavesTime(int userid, int eventid, int leavestime)

{

return DM.UpdateLeavesTime(userid, eventid, leavestime);

}

[WebMethod(Description = "修改提醒时间")]

public Boolean UpdateDRTTime(int userid, int eventid, string defaulttime)

{

return DM.UpdateDRTTime(userid, eventid, defaulttime);

}

[WebMethod(Description = "修改铃声")]

public Boolean UpdateRing(int userid, int eventid, int dremindring)

{

return DM.UpdateRing(userid, eventid, dremindring);

}

[WebMethod(Description = "修改重复提醒")]

public Boolean UpdateRepeat(int userid, int eventid, int dremindrepeat)

{

return DM.UpdateRepeat(userid, eventid, dremindrepeat);

}

[WebMethod(Description = "修改提前时间")]

public Boolean UpdateAhead(int userid, int eventid, int aheadtime)

{

return DM.UpdateAhead(userid, eventid, aheadtime);

}

[WebMethod(Description = "载入叶子信息")]

public BeanLeavesStatistics LoadLeaves(int userid)

{

return LEM.LoadLeaves(userid);

}

[WebMethod(Description = "叶子+1")]

public Boolean LeavesAdd(int userid)

{

return LEM.LeavesAdd(userid);

}

[WebMethod(Description = "专注时长增加")]

public Boolean TimeAdd(int userid, int focustime)

{

return LEM.TimeAdd(userid, focustime);

}

}

}

DBUtils

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

namespace DBUtils

{

///一个操作数据库的类，所有对SQLServer的操作都写在这个类中，使用的时候实例化一个然后直接调用就可以

public class DBOperation : IDisposable

{

public static SqlConnection sqlCon; //用于连接数据库

//将下面的引号之间的内容换成上面记录下的属性中的连接字符串Integrated Security=True

//private String ConServerStr = @"Data Source=39.108.124.121;Initial Catalog=timenote;Persist Security Info=True;User ID=sa;Password=SE2018g01";

//private String ConServerStr = @"Data Source=39.108.124.121;Initial Catalog=timenote;integrated Security=SSPI";

private String ConServerStr = @"Data Source=39.108.124.121;Initial Catalog=timenote;Persist Security Info=True;User ID=sa;Password=SE2018g01ex";

//默认构造函数

public DBOperation()

{

if (sqlCon == null)

{

sqlCon = new SqlConnection();

sqlCon.ConnectionString = ConServerStr;

sqlCon.Open();

}

}

//关闭/销毁函数，相当于Close()

public void Dispose()

{

if (sqlCon != null)

{

sqlCon.Close();

sqlCon = null;

}

}

}

}

DRemindControl

using DBUtils;

using Model;

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Data.SqlTypes;

using System.Linq;

using System.Web;

using UserControl;

namespace DRemindControl

{

public class DRemindManager

{

public Boolean NewDRemind(int eventid,int userid)

{

DBOperation DB = new DBOperation();

try

{

string sql = "insert into dremindinformation(userid,eventid)" +

" values(" + userid + "," + eventid + ")";

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

cmd.Dispose();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

/\*public int FindMaxId()

{

DBOperation DB = new DBOperation();

int i;

try

{

string sql = "select max(dremindid) from dremindinformation";

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

SqlDataReader reader = cmd.ExecuteReader();

if (!reader.Read()) throw new BusinessException("表中没有数据");

i = (int)reader[0];

reader.Close();

cmd.Dispose();

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

return i+1;

}\*/

public BeanDRemindInformation LoadDefaultSet(int userid, int eventid)

{

DBOperation DB = new DBOperation();

int i;

try

{

string sql = "select \* from dremindinformation where userid=" + userid + " and eventid=" + eventid;

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

SqlDataReader reader = cmd.ExecuteReader();

if (!reader.Read()) return null;

BeanDRemindInformation dr = new BeanDRemindInformation();

dr.Dremindid = (int)reader["Dremindid"];

dr.Userid = (int)reader["Userid"];

dr.Dremindring = (int)reader["Dremindring"];

dr.Dremindrepeat = (int)reader["Dremindrepeat"];

dr.Aheadtime = (int)reader["Aheadtime"];

if (!reader["Pid"].ToString().Equals("")) dr.Pid = (int)reader["Pid"];

dr.Dremindvib = (Boolean)reader["Dremindvib"];

dr.Defaulttime = (string)reader["Defaulttime"];

dr.Leavestime = (int)reader["Leavestime"];

dr.Eventid = (int)reader["Eventid"];

reader.Close();

cmd.Dispose();

return dr;

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean UpdateVib(int userid, int eventid, int dremindvib)

{

DBOperation DB = new DBOperation();

try

{

string sql = "update dremindinformation set dremindvib=" + dremindvib + " where userid=" + userid + " and eventid=" + eventid;

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

cmd.Dispose();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean UpdateLeavesTime(int userid, int eventid, int leavestime)

{

DBOperation DB = new DBOperation();

try

{

string sql = "update dremindinformation set leavestime=" + leavestime + " where userid=" + userid + " and eventid=" + eventid;

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

cmd.Dispose();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean UpdateDRTTime(int userid, int eventid, string defaulttime)

{

DBOperation DB = new DBOperation();

try

{

string sql = "update dremindinformation set defaulttime='" + defaulttime + "' where userid=" + userid + " and eventid=" + eventid;

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

cmd.Dispose();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean UpdateRing(int userid, int eventid, int dremindring)

{

DBOperation DB = new DBOperation();

try

{

string sql = "update dremindinformation set dremindring=" + dremindring + " where userid=" + userid + " and eventid=" + eventid;

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

cmd.Dispose();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean UpdateRepeat(int userid, int eventid, int dremindrepeat)

{

DBOperation DB = new DBOperation();

try

{

string sql = "update dremindinformation set dremindrepeat=" + dremindrepeat + " where userid=" + userid + " and eventid=" + eventid;

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

cmd.Dispose();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean UpdateAhead(int userid, int eventid, int aheadtime)

{

DBOperation DB = new DBOperation();

try

{

string sql = "update dremindinformation set aheadtime=" + aheadtime + " where userid=" + userid + " and eventid=" + eventid;

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

cmd.Dispose();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

}

}

EventControl

using DBUtils;

using Model;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Reflection;

using System.Web;

using System.Xml.Serialization;

using UserControl;

namespace EventControl

{

public class EventManager

{

public Boolean NewEvent(string eventname, int eventpriority, int listid,int userid, DateTime eventdate)

{

DBOperation DB = new DBOperation();

try

{

string sql;

SqlCommand cmd;

sql = "select dremindid from dremindinformation where userid=" + userid;

cmd = new SqlCommand(sql, DBOperation.sqlCon);

SqlDataReader reader = cmd.ExecuteReader();

if (!reader.Read()) return false;

int dremindid = (int)reader[0];

reader.Close();

if (listid == 0)

{

sql = "select listid from listinformation where userid=" + userid + " and listpriority<=all(select listpriority from listinformation where userid=" + userid + ")";

cmd = new SqlCommand(sql, DBOperation.sqlCon);

reader = cmd.ExecuteReader();

if (!reader.Read()) return false;

listid = (int)reader[0];

reader.Close();

}

sql = "insert into eventinformation(eventname,userid,listid,eventpriority,dremindid";

if (DateTime.Compare(eventdate, Convert.ToDateTime("0001-01-01T00:00:00")) != 0) sql += ",eventdate";

sql += ") values('" + eventname + "'," + userid + "," + listid + "," + eventpriority + "," + dremindid;

if (DateTime.Compare(eventdate, Convert.ToDateTime("0001-01-01T00:00:00")) != 0) sql += ",'" + eventdate + "'";

sql += ")";

cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean UpdateEvent(string eventname, int eventpriority, int listid, DateTime eventdate, int leaveseventsign, string eventnote, int eventid)

{

DBOperation DB = new DBOperation();

SqlTransaction trans = DBOperation.sqlCon.BeginTransaction();

SqlCommand cmd = new SqlCommand();

cmd.Connection = DBOperation.sqlCon;

cmd.Transaction = trans;

try

{

string sql;

if (!eventname.Equals(""))

{

sql = "update eventinformation set eventname='" + eventname + "' where eventid=" + eventid;

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

cmd.Dispose();

}

if (listid != -1)

{

sql = "update eventinformation set listid=" + listid + " where eventid=" + eventid;

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

cmd.Dispose();

}

if (eventpriority != -1)

{

sql = "update eventinformation set eventpriority=" + eventpriority + " where eventid=" + eventid;

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

cmd.Dispose();

}

if (DateTime.Compare(eventdate, Convert.ToDateTime("0001-01-01T00:00:00")) != 0)

{

sql = "update eventinformation set eventdate='" + eventdate + "' where eventid=" + eventid;

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

cmd.Dispose();

}

if (leaveseventsign != -1)

{

sql = "update eventinformation set leaveseventsign=" + leaveseventsign + " where eventid=" + eventid;

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

cmd.Dispose();

}

if (!eventnote.Equals(""))

{

sql = "update eventinformation set eventnote='" + eventnote + "' where eventid=" + eventid;

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

cmd.Dispose();

}

trans.Commit();

return true;

}

catch (Exception ex)

{

trans.Rollback();

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean UpdateEventPriority(int eventpriority, int eventid)

{

DBOperation DB = new DBOperation();

try

{

string sql = "update eventinformation set eventpriority=" + eventpriority + " where eventid=" + eventid;

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean DeleteEvent(int eventid)

{

DBOperation DB = new DBOperation();

try

{

string sql = "delete from eventinformation where eventid=" + eventid;

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public BeanEventInformation LoadEvent(int usereid)

{

DBOperation DB = new DBOperation();

try

{

string sql = "select \* from eventinformation where useremail='" + usereid + "'";

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

SqlDataReader reader = cmd.ExecuteReader();

if (!reader.Read()) throw new BusinessException("该账号不存在");

BeanEventInformation e = new BeanEventInformation();

e.Eventid = (int)reader["Eventid"];

e.Eventname = (string)reader["Eventname"];

e.Eventnote = (string)reader["Eventnote"];

e.Eventpriority = (int)reader["Eventpriority"];

e.Userid = (int)reader["Userid"];

e.Listid = (int)reader["Listid"];

e.Dremindid = (int)reader["Dremindid"];

e.Eventdate = DateTime.Parse(reader["Eventdate"].ToString());

e.Dreminddate = DateTime.Parse(reader["Dreminddate"].ToString());

e.Leaveseventsign = (Boolean)reader["Leaveseventsign"];

e.Pid = (int)reader["Pid"];

cmd.Dispose();

reader.Close();

return e;

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

//[XmlInclude(typeof(List<BeanEventInformation>))]

//[XmlInclude(typeof(BeanEventInformation))]

public List<BeanEventInformation> AllEvent(int userid)

{

DBOperation DB = new DBOperation();

try

{

string sql = "select \* from eventinformation where userid=" + userid + "order by eventdate";

SqlDataAdapter sda = new SqlDataAdapter(sql, DBOperation.sqlCon);

DataSet ds = new DataSet();

sda.Fill(ds);

return DataSetToList(ds, 0);

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public List<BeanEventInformation> AllEventByPro(int userid)

{

DBOperation DB = new DBOperation();

try

{

string sql = "select \* from eventinformation where userid=" + userid + "order by eventeriority desc,eventdate asc";

SqlDataAdapter sda = new SqlDataAdapter(sql, DBOperation.sqlCon);

DataSet ds = new DataSet();

sda.Fill(ds);

return DataSetToList(ds, 0);

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

//[XmlInclude(typeof(List<BeanEventInformation>))]

//[XmlInclude(typeof(BeanEventInformation))]

public List<BeanEventInformation> DataSetToList(DataSet dataSet, int tableIndex)

{

if (dataSet == null || dataSet.Tables.Count <= 0 || tableIndex < 0)

return null;

DataTable dt = dataSet.Tables[tableIndex];

//BeanEventInformation e = new BeanEventInformation();

List<BeanEventInformation> result = new List<BeanEventInformation>();

for (int i = 0; i < dt.Rows.Count; i++)

{

//创建泛型对象

BeanEventInformation \_e = Activator.CreateInstance<BeanEventInformation>();

//获取对象所有属性

PropertyInfo[] propertyInfo = \_e.GetType().GetProperties();

for (int j = 0; j < dt.Columns.Count; j++)

{

foreach (PropertyInfo info in propertyInfo)

{

//属性名称和列名相同时赋值

if (dt.Columns[j].ColumnName.ToUpper().Equals(info.Name.ToUpper()))

{

if (dt.Rows[i][j] != DBNull.Value)

{

info.SetValue(\_e, dt.Rows[i][j], null);

}

else

{

info.SetValue(\_e, null, null);

}

break;

}

}

}

result.Add(\_e);

}

return result;

}

public Boolean EventComplete(int eventid)

{

DBOperation DB = new DBOperation();

try

{

string sql = "update eventinformation set complete=1 where eventid=" + eventid;

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean EventUnComplete(int eventid)

{

DBOperation DB = new DBOperation();

try

{

string sql = "update eventinformation set complete=0 where eventid=" + eventid;

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean UpdatePid(int eventid, int pid)

{

DBOperation DB = new DBOperation();

try

{

string sql = "update eventinformation set pid=" + pid + " where eventid=" + eventid;

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

}

}

Exception

using System;

using System.Runtime.Serialization;

namespace UserControl

{

[Serializable]

internal class BusinessException : Exception

{

public BusinessException()

{

}

public BusinessException(string message) : base(message)

{

}

public BusinessException(string message, Exception innerException) : base(message, innerException)

{

}

protected BusinessException(SerializationInfo info, StreamingContext context) : base(info, context)

{

}

}

}

LeavesControl

using DBUtils;

using Model;

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

namespace LeavesControl

{

public class LeavesManager

{

public BeanLeavesStatistics LoadLeaves(int userid)

{

DBOperation DB = new DBOperation();

int i;

try

{

string sql = "select \* from leavesstatistics where userid=" + userid + " and leavesdate=convert(varchar(10),getdate(),20)";

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

SqlDataReader reader = cmd.ExecuteReader();

if (!reader.Read()) return null;

BeanLeavesStatistics ls = new BeanLeavesStatistics();

ls.Leavesdate = DateTime.Parse(reader["Leavesdate"].ToString());

ls.Leavesamount = (int)reader["Leavesamount"];

ls.Focustime = (int)reader["Focustime"];

ls.Userid = (int)reader["Userid"];

reader.Close();

cmd.Dispose();

return ls;

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean LeavesAdd(int userid)

{

DBOperation DB = new DBOperation();

try

{

string sql = "update leavesstatistics set leavesamount=leavesamount+1 where userid=" + userid + " and leavesdate=convert(varchar(10),getdate(),20)";

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean TimeAdd(int userid, int focustime)

{

DBOperation DB = new DBOperation();

try

{

string sql = "update leavesstatistics set focustime=focustime+" + focustime + " where userid=" + userid + " and leavesdate=convert(varchar(10),getdate(),20)";

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

}

}

ListControl

using DBUtils;

using Model;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Reflection;

using System.Web;

namespace ListControl

{

public class ListManager

{

public Boolean NewList(string listname, int userid)

{

DBOperation DB = new DBOperation();

try

{

string sql = "select Max(listpriority) from listinformation where userid = " + userid;

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

SqlDataReader reader = cmd.ExecuteReader();

if (!reader.Read()) return false;

int listpriority = (int)reader[0]+1;

reader.Close();

sql = "insert into listinformation(listname,userid,listpriority) " +

"values('" + listname + "'," + userid + "," + listpriority + ")";

cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean UpdateList(string listname, int listid)

{

DBOperation DB = new DBOperation();

try

{

string sql = "update listinformation set listname='" + listname + "' where listid=" + listid;

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean DeleteList(int listid)

{

DBOperation DB = new DBOperation();

SqlTransaction trans = DBOperation.sqlCon.BeginTransaction();

SqlCommand cmd = new SqlCommand();

cmd.Connection = DBOperation.sqlCon;

cmd.Transaction = trans;

try

{

string sql = "select min(listid) from listinformation where userid=(select userid from listinformation where listid=" + listid + ")";

cmd.CommandText = sql;

SqlDataReader reader = cmd.ExecuteReader();

if (!reader.Read()) return false;

int listid2 = (int)reader[0];

reader.Close();

cmd.Dispose();

sql = "update eventinformation set listid=" + listid2 + " where listid=" + listid;

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

cmd.Dispose();

trans.Commit();

sql = "delete from Listinformation where listid=" + listid;

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

cmd.Dispose();

return true;

}

catch (Exception ex)

{

trans.Rollback();

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public List<BeanListInformation> AllList(int userid)

{

DBOperation DB = new DBOperation();

try

{

string sql = "select \* from ListInformation where userid=" + userid + " ";

SqlDataAdapter sda = new SqlDataAdapter(sql, DBOperation.sqlCon);

DataSet ds = new DataSet();

sda.Fill(ds);

return DataSetToList(ds, 0);

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public List<BeanListInformation> DataSetToList(DataSet dataSet, int tableIndex)

{

if (dataSet == null || dataSet.Tables.Count <= 0 || tableIndex < 0)

return null;

DataTable dt = dataSet.Tables[tableIndex];

//BeanEventInformation e = new BeanEventInformation();

List<BeanListInformation> result = new List<BeanListInformation>();

for (int i = 0; i < dt.Rows.Count; i++)

{

//创建泛型对象

BeanListInformation \_l = Activator.CreateInstance<BeanListInformation>();

//获取对象所有属性

PropertyInfo[] propertyInfo = \_l.GetType().GetProperties();

for (int j = 0; j < dt.Columns.Count; j++)

{

foreach (PropertyInfo info in propertyInfo)

{

//属性名称和列名相同时赋值

if (dt.Columns[j].ColumnName.ToUpper().Equals(info.Name.ToUpper()))

{

if (dt.Rows[i][j] != DBNull.Value)

{

info.SetValue(\_l, dt.Rows[i][j], null);

}

else

{

info.SetValue(\_l, null, null);

}

break;

}

}

}

result.Add(\_l);

}

return result;

}

}

}

UserControl

using DBUtils;

using DRemindControl;

using Model;

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Web;

using System.Xml.Serialization;

namespace UserControl

{

public class UserManager

{

[XmlInclude(typeof(BeanUserInformation))]

public BeanUserInformation LoadUser(string useremail)

{

DBOperation DB = new DBOperation();

try

{

string sql = "select \* from userinformation where useremail='" + useremail + "'";

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

SqlDataReader reader = cmd.ExecuteReader();

if (!reader.Read()) return null;

BeanUserInformation u = new BeanUserInformation();

u.Userid = (int)reader["userid"];

u.Useremail = (string)reader["useremail"];

u.Userpassword = (string)reader["userpassword"];

u.Username = (String)reader["username"];

u.Creatdate = DateTime.Parse(reader["creatdate"].ToString());

u.Leavesid = (int)reader["leavesid"];

if (!reader["stopdate"].ToString().Equals("")) u.Stopdate = DateTime.Parse(reader["stopdate"].ToString());

u.Usercalendar = (Boolean)reader["usercalendar"];

u.Usertypeface = (string)reader["usertypeface"];

u.Achievement = (int)reader["achievement"];

if (!reader["authcode"].ToString().Equals("")) u.Authcode = (string)reader["authcode"];

if (!reader["userhead"].ToString().Equals("")) u.Userhead = System.Text.Encoding.Default.GetString((byte[])reader["userhead"]);

cmd.Dispose();

reader.Close();

return u;

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

/\*

public Boolean UserActivate(String useremail)

{

DBOperation DB = new DBOperation();

try

{

string sql = "select stopdate from userinformation where useremail='" + useremail + "'";

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

SqlDataReader reader = cmd.ExecuteReader();

if (!reader.Read())

{

return true;

throw new BusinessException("该账号未激活");

}

cmd.Dispose();

reader.Close();

return false;

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean EmailVerify(String useremail)

{

DBOperation DB = new DBOperation();

try

{

string sql = "select \* from userinformation where useremail='" + useremail + "'";

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

SqlDataReader reader = cmd.ExecuteReader();

cmd.Dispose();

if (reader.Read())

{

reader.Close();

return false;

throw new BusinessException("该邮箱已注册");

}

reader.Close();

cmd.Dispose();

return true;

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}\*/

public Boolean SendAuthCode(string useremail)

{

DBOperation DB = new DBOperation();

SqlTransaction trans = DBOperation.sqlCon.BeginTransaction();

SqlCommand cmd = new SqlCommand();

cmd.Connection = DBOperation.sqlCon;

cmd.Transaction = trans;

Random rd = new Random();

string authcode = rd.Next(100000, 999999).ToString();

try

{

string sql = "update userinformation set authcode='"+authcode+"' where useremail='"+useremail+"'";

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

cmd.Dispose();

SendEmail.SendMailUse(useremail, authcode);

trans.Commit();

return true;

}

catch (Exception ex)

{

trans.Rollback();

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

int userid2;

public Boolean CreateUser(String useremail, String userpwd, String username)

{

DBOperation DB = new DBOperation();

SqlTransaction trans = DBOperation.sqlCon.BeginTransaction();

SqlCommand cmd = new SqlCommand();

cmd.Connection = DBOperation.sqlCon;

cmd.Transaction = trans;

Random rd = new Random();

string authcode = rd.Next(100000, 999999).ToString();

try

{

string sql = "select \* from userinformation where useremail='" + useremail + "'";

cmd.CommandText = sql;

SqlDataReader reader = cmd.ExecuteReader();

if (!reader.Read())

{

reader.Close();

sql = "insert into userinformation(useremail,userpassword,username,authcode) " +

"values('" + useremail + "','" + userpwd + "','" + username + "','" + authcode + "')";

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

cmd.Dispose();

sql = "select userid from userinformation where useremail='" + useremail + "'";

cmd.CommandText = sql;

reader = cmd.ExecuteReader();

if (reader.Read()) userid2 = (int)reader[0];

reader.Close();

cmd.Dispose();

sql = "insert into listinformation(listname,userid,listpriority) " +

"values('收集箱'," + userid2 + ",0)";

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

cmd.Dispose();

sql = "insert into dremindinformation(userid,eventid) values(" + userid2 + ",-99)";

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

cmd.Dispose();

SendEmail.SendMailUse(useremail, authcode);

trans.Commit(); //提交事务

return true;

}

else if (!reader[6].ToString().Equals(""))

{

reader.Close();

Random random = new Random();

string authcode1 = random.Next(100000, 999999).ToString();

sql = "update userinformation set authcode='" + authcode1 + "',stopdate = dateadd(hour,(2),getdate()),userpassword='"+userpwd+"',username='"+username+"' where useremail='" + useremail + "'";

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

cmd.Dispose();

SendEmail.SendMailUse(useremail, authcode);

trans.Commit(); //提交事务

return true;

}

else

{

reader.Close();

return false;

}

}

catch (Exception ex)

{

trans.Rollback(); //回滚事务

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public void UserRollBack(string useremail)

{

DBOperation DB = new DBOperation();

try

{

string sql = "select \* from userinformation where useremail='" + useremail + "'";

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

SqlDataReader reader = cmd.ExecuteReader();

cmd.Dispose();

if (!reader.Read())

{

reader.Close();

throw new BusinessException("该用户不存在");

}

else

{

reader.Close();

sql = "delete from userinformation where useremail='" + useremail + "'";

cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

cmd.Dispose();

}

reader.Close();

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean UpdateEmail(string useremail, int userid)

{

DBOperation DB = new DBOperation();

SqlTransaction trans = DBOperation.sqlCon.BeginTransaction();

SqlCommand cmd = new SqlCommand();

cmd.Connection = DBOperation.sqlCon;

cmd.Transaction = trans;

Random rd = new Random();

string authcode = rd.Next(100000, 999999).ToString();

try

{

string sql = "update userinformation set useremail='" + useremail + "',authcode='" + authcode + "' where userid='" + userid + "'";

cmd.CommandText = sql;

cmd.ExecuteNonQuery();

cmd.Dispose();

SendEmail.SendMailUse(useremail, authcode);

trans.Commit();

return true;

}

catch (Exception ex)

{

trans.Rollback();

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean UpdateName(string username, int userid)

{

DBOperation DB = new DBOperation();

try

{

string sql = "update userinformation set username='" + username + "' where userid=" + userid;

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

cmd.Dispose();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean UpdateUserpassward(string userpassword,int userid)

{

DBOperation DB = new DBOperation();

try

{

string sql = "update userinformation set userpassword='" + userpassword + "' where userid=" + userid;

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

cmd.Dispose();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public void DeleteStopDate(String useremail)

{

DBOperation DB = new DBOperation();

try

{

string sql = "select stopdate from userinformation where useremail='" + useremail + "'";

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

SqlDataReader reader = cmd.ExecuteReader();

cmd.Dispose();

if (!reader.Read())

{

reader.Close();

//return false;

throw new BusinessException("该用户不存在");

}

else if ((DateTime)reader[0] == null)

{

reader.Close();

//return false;

throw new BusinessException("该用户已激活");

}

reader.Close();

sql = "update userinformation set stopdate=null where useremail='"+useremail+"'";

cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.ExecuteNonQuery();

cmd.Dispose();

//return true;

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public Boolean SaveHead(string x,int userid)

{

DBOperation DB = new DBOperation();

try

{

string sql = "update userinformation set userhead=@img where userid=" + userid;

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

cmd.Parameters.AddWithValue("@img", System.Text.Encoding.Default.GetBytes(x));

//cmd.Parameters.AddWithValue("@img", x);

cmd.ExecuteNonQuery();

return true;

}

catch (Exception ex)

{

return false;

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

public static Image GetImageByBytes(byte[] bytes)

{

Image photo = null;

using (MemoryStream ms = new MemoryStream(bytes))

{

ms.Write(bytes, 0, bytes.Length);

ms.Seek(0, SeekOrigin.Begin);

ms.Read(bytes, 0, bytes.Length);

photo = Image.FromStream(ms, true);

}

return photo;

}

public string GetHead(int userid)

{

DBOperation DB = new DBOperation();

try

{

string sql = "select userhead from userinformation where userid=" + userid;

SqlCommand cmd = new SqlCommand(sql, DBOperation.sqlCon);

SqlDataReader reader = cmd.ExecuteReader();

if (!reader.Read()) return null;

if (reader[0].ToString().Equals("")) return null;

return System.Text.Encoding.Default.GetString((byte[])reader[0]);

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

finally

{

if (DB != null)

try

{

DB.Dispose();

}

catch (SqlException ex)

{

throw new Exception("数据库操作异常！");

throw new Exception(ex.Message);

}

}

}

}

}

Model

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

using System.Web;

using System.Xml;

using System.Xml.Serialization;

namespace Model

{

//[Serializable]

public class BeanUserInformation

{

private int userid;

private String username;

private String userpassword;

private String useremail;

private int leavesid;

private DateTime creatdate;

private DateTime stopdate;

private Boolean usercalendar;

private String usertypeface;

private int achievement;

private string authcode;

private string userhead;

public int Userid { get => userid; set => userid = value; }

public string Username { get => username; set => username = value; }

public string Userpassword { get => userpassword; set => userpassword = value; }

public string Useremail { get => useremail; set => useremail = value; }

public DateTime Creatdate { get => creatdate; set => creatdate = value; }

public DateTime Stopdate { get => stopdate; set => stopdate = value; }

public bool Usercalendar { get => usercalendar; set => usercalendar = value; }

public string Usertypeface { get => usertypeface; set => usertypeface = value; }

public int Achievement { get => achievement; set => achievement = value; }

public int Leavesid { get => leavesid; set => leavesid = value; }

public string Authcode { get => authcode; set => authcode = value; }

public string Userhead { get => userhead; set => userhead = value; }

}

//[Serializable]

public class BeanEventInformation

{

private int userid;

private int eventid;

private int dremindid;

private int listid;

private String eventname;

private int eventpriority;

private DateTime eventdate;

private DateTime dreminddate;

private Boolean leaveseventsign;

private String eventnote;

private int complete;

private int pid;

public int Userid { get => userid; set => userid = value; }

public int Eventid { get => eventid; set => eventid = value; }

public int Dremindid { get => dremindid; set => dremindid = value; }

public int Listid { get => listid; set => listid = value; }

public string Eventname { get => eventname; set => eventname = value; }

public int Eventpriority { get => eventpriority; set => eventpriority = value; }

public DateTime Eventdate { get => eventdate; set => eventdate = value; }

public DateTime Dreminddate { get => dreminddate; set => dreminddate = value; }

public bool Leaveseventsign { get => leaveseventsign; set => leaveseventsign = value; }

public string Eventnote { get => eventnote; set => eventnote = value; }

public int Complete { get => complete; set => complete = value; }

public int Pid { get => pid; set => pid = value; }

}

public class BeanListInformation

{

private int userid;

private int listid;

private String listname;

private DateTime createdate;

private String backgroundcolor;

private int listpriority;

public int Userid { get => userid; set => userid = value; }

public int Listid { get => listid; set => listid = value; }

public string Listname { get => listname; set => listname = value; }

public DateTime Createdate { get => createdate; set => createdate = value; }

public string Backgroundcolor { get => backgroundcolor; set => backgroundcolor = value; }

public int Listpriority { get => listpriority; set => listpriority = value; }

}

public class BeanDRemindInformation

{

private int dremindid;

private int userid;

private int dremindring;

private int dremindrepeat;

private int aheadtime;

private int eventid;

private int pid;

private string defaulttime;

private int leavestime;

private Boolean dremindvib;

public int Dremindid { get => dremindid; set => dremindid = value; }

public int Userid { get => userid; set => userid = value; }

public int Dremindring { get => dremindring; set => dremindring = value; }

public int Dremindrepeat { get => dremindrepeat; set => dremindrepeat = value; }

public int Aheadtime { get => aheadtime; set => aheadtime = value; }

public int Eventid { get => eventid; set => eventid = value; }

public int Pid { get => pid; set => pid = value; }

public string Defaulttime { get => defaulttime; set => defaulttime = value; }

public int Leavestime { get => leavestime; set => leavestime = value; }

public bool Dremindvib { get => dremindvib; set => dremindvib = value; }

}

public class BeanAppealInformation

{

private int userid;

private int appealid;

private String appealDeclare;

private String username;

private String appealemail;

private String appealphone;

private String appealstatus;

private DateTime dealtime;

public int Userid { get => userid; set => userid = value; }

public int Appealid { get => appealid; set => appealid = value; }

public string AppealDeclare { get => appealDeclare; set => appealDeclare = value; }

public string Username { get => username; set => username = value; }

public string Appealemail { get => appealemail; set => appealemail = value; }

public string Appealphone { get => appealphone; set => appealphone = value; }

public string Appealstatus { get => appealstatus; set => appealstatus = value; }

public DateTime Dealtime { get => dealtime; set => dealtime = value; }

}

public class BeanLeaveSet

{

private int leavesid;

private int userid;

private int leavesduration;

private int breakduration;

public int Leavesid { get => leavesid; set => leavesid = value; }

public int Userid { get => userid; set => userid = value; }

public int Leavesduration { get => leavesduration; set => leavesduration = value; }

public int Breakduration { get => breakduration; set => breakduration = value; }

}

public class BeanLeavesStatistics

{

private DateTime leavesdate;

private int leavesamount;

private int focustime;

private int userid;

public DateTime Leavesdate { get => leavesdate; set => leavesdate = value; }

public int Leavesamount { get => leavesamount; set => leavesamount = value; }

public int Focustime { get => focustime; set => focustime = value; }

public int Userid { get => userid; set => userid = value; }

}

}

SendEmail

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Mail;

using System.Web;

namespace UserControl

{

public class SendEmail

{

public static void SendMailUse(string useremail,string authcode)

{

string host = "smtp.mxhichina.com";// 邮件服务器smtp.163.com表示网易邮箱服务器

string userName = "sproutgod@timenote.xin";// 发送端账号

string password = "Cxw19980721";// 发送端密码(这个客户端重置后的密码)

int port = 587;

SmtpClient client = new SmtpClient();

client.DeliveryMethod = SmtpDeliveryMethod.Network;//指定电子邮件发送方式

client.Host = host;//邮件服务器

client.Port = port;//邮件服务器端口

client.UseDefaultCredentials = true;

client.EnableSsl = true;

client.Credentials = new System.Net.NetworkCredential(userName, password);//用户名、密码

//////////////////////////////////////

string strfrom = userName;

string strto = useremail;

//string strcc = "2605625733@qq.com";//抄送

string subject = "———欢迎使用TimeNote——————";//邮件的主题

string body = " 您的验证码为"+ authcode ;//发送的邮件正文

System.Net.Mail.MailMessage msg = new System.Net.Mail.MailMessage();

msg.From = new MailAddress(strfrom, "SproutGod");

msg.To.Add(strto);

//msg.CC.Add(strcc);

msg.Subject = subject;//邮件标题

msg.SubjectEncoding = System.Text.Encoding.UTF8;

msg.Body = body;//邮件内容

msg.BodyEncoding = System.Text.Encoding.UTF8;//邮件内容编码

msg.IsBodyHtml = true;//是否是HTML邮件

msg.Priority = MailPriority.High;//邮件优先级

object userState = msg;

try

{

client.SendAsync(msg,userState);

Console.WriteLine("发送成功");

}

catch (System.Net.Mail.SmtpException ex)

{

Console.WriteLine(ex.Message, "发送邮件出错");

}

}

}

}