# 公共类方法

## 电脑基本操作

### 重启

private void 立刻重启ToolStripMenuItem\_Click(object sender, EventArgs e)

{

try

{

//启动本地程序并执行命令

System.Diagnostics.Process.Start("shutdown.exe", " -r -t 0");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

### 关闭显示器

#region 调用微软API

/// 关闭显示器

/// <param name="sender"></param>

/// <param name="e"></param>

void CloseLCD(object sender, EventArgs e)

{

SendMessage(this.Handle, WM\_SYSCOMMAND, SC\_MONITORPOWER, 2); // 2 为关闭显示器， －1则打开显示器

}

#region SendMessage

public const uint WM\_SYSCOMMAND = 0x0112;

public const uint SC\_MONITORPOWER = 0xF170;

[System.Runtime.InteropServices.DllImport("user32")]

public static extern IntPtr SendMessage(IntPtr hWnd, uint wMsg, uint wParam, int lParam);

#endregion

private void 立刻关闭显示屏ToolStripMenuItem\_Click(object sender, EventArgs e)

{

SendMessage(this.Handle, WM\_SYSCOMMAND, SC\_MONITORPOWER, 2);

// 2 为关闭显示器，1则打开显示器

}

### 注销

//注销 函数 声明

[System.Runtime.InteropServices.DllImport("user32.dll", EntryPoint = "ExitWindowsEx", CharSet = System.Runtime.InteropServices.CharSet.Ansi)]

//ExitWindowsEx 函数

private static extern int ExitWindowsEx(int uFlags, int dwReserved);

private void 注销电脑ToolStripMenuItem\_Click(object sender, EventArgs e)

{

ExitWindowsEx(0, 0);

}

### 锁定计算机

[System.Runtime.InteropServices.DllImport("user32.dll")]

private static extern void LockWorkStation();//须写extern

private void 立刻锁定ToolStripMenuItem\_Click(object sender, EventArgs e)

{

/\*System.Diagnostics.Process p = new System.Diagnostics.Process();

p.StartInfo.WorkingDirectory = "c:\\windows\\system32 ";

p.StartInfo.FileName = "rundll32.exe ";

p.StartInfo.Arguments = "user32.dll,LockWorkStation ";

p.Start();\*/这是第一种方法

LockWorkStation();//这是第二种方法

}

### 睡眠

private void 立刻睡眠ToolStripMenuItem\_Click(object sender, EventArgs e)

{

Application.SetSuspendState(PowerState.Hibernate, true, true);//休眠

}

### 关机

private void 立刻关机ToolStripMenuItem\_Click(object sender, EventArgs e)

{

try

{

//启动本地程序并执行命令

System.Diagnostics.Process.Start("Shutdown.exe", " -s -t 0");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

## 取得本机串口

/// </summary>

/// <returns></returns>

public static string[] GetHostCOM()

{

string[] sAllPort = null;

try

{

sAllPort = SerialPort.GetPortNames();

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

return sAllPort;

}

## 获取本地IP地址

/// </summary>

/// <returns></returns>

public static string GetHostIPAddress()

{

string strHostIPAddr;

strHostIPAddr = "";

System.Net.IPAddress[] addressList = Dns.GetHostEntry(Dns.GetHostName()).AddressList;

strHostIPAddr += addressList[addressList.Length - 1].ToString();

return strHostIPAddr;

}

## 验证ip地址格式

public bool IPCheck(string IP)

{

string num = "(25[0-5]|2[0-4] \\d|[0-1]\\d{2}|[1-9]?\\d)";

return Regex.IsMatch(IP, ("^" + num + "\\." + num + "\\." + num + "\\." + num + "$"));

}

## 判断对象是否是整型数

/// </summary>

/// <param name="obj"></param>

/// <returns></returns>

public static bool CheckObjectIsInteger(object obj)

{

obj = obj.ToString().Trim();

if (string.IsNullOrEmpty(obj.ToString()))

{

return false;

}

else

{

return Regex.IsMatch(obj.ToString(), @"^\d{1,}$");

}

}

## 只能输入数字的验证

private void txtNewPay\_KeyPress\_1(object sender, KeyPressEventArgs e)

{

if (e.KeyChar != 8 && !char.IsDigit(e.KeyChar))

{

MessageBox.Show("只能输入数字", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);

e.Handled = true;

}

}

## 验证对象是否是字节数组

/// </summary>

/// <param name="objValue"></param>

/// <returns></returns>

public static bool CheckObjIsByteArray(object objValue)

{

if (objValue is byte[])

{

return true;

}

else

{

return false;

}

}

## 验证身份证号码的格式

/// </summary>

/// <param name="IDNo"></param>

/// <returns></returns>

public static bool IsValidIdentityNo(string IdentityNo)

{

IdentityNo = IdentityNo.Trim();

if (string.IsNullOrEmpty(IdentityNo))

{

return false;

}

else

{

return Regex.IsMatch(IdentityNo, @"^(\d{14}(\d{1}|\d{4}|\d{3}[xy]))$");

}

}

## 防注入方法

/// </summary>

/// <param name="text"></param>

/// <returns></returns>

public static string InputText(string text)

{

text = text.Trim();

if (string.IsNullOrEmpty(text))

return string.Empty;

// 替换两个或两个以上的空格为一个空格

text = Regex.Replace(text, "[\\s]{2,}", " ");

// 替换<br>为转义字符“\n”

text = Regex.Replace(text, "(<[b|B][r|R]/\*>)+|(<[p|P](.|\\n)\*?>)", "\n");

// 替换"&nbsp;"为空格

text = Regex.Replace(text, "(\\s\*&[n|N][b|B][s|S][p|P];\\s\*)+", " ");

// 替换特殊字符

text = Regex.Replace(text, "<(.|\\n)\*?>", string.Empty);

// 替换单引号

text = text.Replace("'", "''");

return text;

}

## 生成指定个数的随机数

/// </summary>

/// <param name="codeCount"></param>

/// <returns></returns>

private string CreateRandomCode(int codeCount)

{

string allChar = "0,1,2,3,4,5,6,7,8,9";

string[] allCharArray = allChar.Split(',');

string randomCode = "";

int temp = -1;

Random rand = new Random();

for (int i = 0; i < codeCount; i++)

{

if (temp != -1)

{

rand = new Random(i \* temp \* ((int)DateTime.Now.Ticks));

}

int t = rand.Next(10);

if (temp == t)

{

return CreateRandomCode(codeCount);

}

temp = t;

randomCode += allCharArray[t];

}

return randomCode;

}

## 验证数据是否是数字格式

/// <param name="number">要验证的数据</param>

public static bool IsNumber(object number)

{

try

{

int num = Convert.ToInt32(number);

return true;

}

catch

{

return false;

}

}

## 判断输入是否为日期类型

/// <param name="s">待检查数据</param>

public static bool IsDate(string s)

{

if (s == null)

{

return false;

}

else

{

try

{

DateTime d = DateTime.Parse(s);

return true;

}

catch

{

return false;

}

}

}

## 判断接收的值是否是十进制数

/// <param name="dec">值</param>

public static bool IsDecimal(string dec)

{

if (dec == null)

{ return false; }

else

{

try

{

decimal.Parse(dec);

return true;

}

catch

{ return false; }

}

}

## 正则表达式判断Email

/// <param name="str">Email</param>

public static bool EmailRegex(string str)

{

if (str.IndexOf('@') == str.Length - 1 || str.IndexOf('@') == -1)//‘@’的位置是否等于字符串的长度或者‘@’的位置不存在

{ return false; }

else

{

if (str.IndexOf('.') == str.Length - 1 || str.IndexOf('.') == -1)//‘.’的位置是否等于字符串的长度或者‘.’的位置不存在

{ return false; }

else

{

if (str.LastIndexOf('@') > str.LastIndexOf('.'))//判断最后一个‘@’是否在最后一个‘.’的前面

{ return false; }

else

{ return true; }

}

}

}

## 正则判断电话号码

/// <param name="str">电话号码</param>

public static bool PhoneRegex(string str)

{

if (str.IndexOf('-') == 4 || str.IndexOf('-') == 3)//判断‘-’的位置是否在第四获第五位上

{

if (str.IndexOf('0') == 0)//第一位是否是零

{

string[] spitStr = str.Split('-');//字符串按‘-’分组

if (spitStr.Length == 2)//判断数组长度是否为2

{

if (IsNumber(spitStr[1]))//判断第二个数组是否为数字

{

if (IsNumber(spitStr[0]))//判断第一个数组是否为数字

{ return true; }

else { return false; }

}

else { return false; }

}

else

{ return false; }

}

else { return false; }

}

else

{ return false; }

}

## 正则判断QQ

/// <param name="str">QQ</param>

public static bool QqRegex(string str)

{

if (str.IndexOf('0') == 0)//判断QQ第一位数是否为零

{ return false; }

else

{

if (str.Length > 12)//判断是否大于12位

{ return false; }

else

{

int i = 0, j = 0;

for (i = 0; i < str.Length; i++) //循环判断是否为数字

{

if (!IsNumber(str[i]))

{ j++; }

}

if (j != 0)

{ return false; }

else

{ return true; }

}

}

}

## 正则判断手机号

/// <param name="str">手机号</param>

public static bool MobileRegex(string str)

{

string Pattern = @"1[3,5,8]\d{9}";

return Regex.IsMatch(str, Pattern);

}

## 判断字符串是否是邮编

/// </summary>

/// <param name="str">字符串</param>

/// <returns></returns>

public static bool IsPost(string str)

{

if (str.Length == 6)//判断邮编是否是六位

{

if (IsNumber(str))//判断邮编是否是数字

{ return true; }

else

{ return false; }

}

else

{ return false; }

}

## 判断接收的值是否是int类型

/// <param name="obj">判断的值</param>

/// <param name="defaultValue">0</param>

public static int ToInt(object obj, int defaultValue)

{

if (obj.ToString() == "" || obj.ToString() == null)

return 0;

int result;

int.TryParse(obj.ToString(), out result);

if (result == 0)

return defaultValue;

else

return result;

}

public static bool IsInt(string str)

{

string Pattern = @"\d";

return Regex.IsMatch(str, Pattern);

}

## 判断接收的值是否是DataTime类型

/// <param name="obj">判断的值</param>

/// <param name="dafaultUalue">当前时间</param>

public static DateTime ToDateTime(object obj, DateTime dafaultUalue)

{

DateTime dt = DateTime.MinValue;

DateTime.TryParse(obj.ToString(), out dt);

if (dt == DateTime.MinValue)

return dafaultUalue;

else

return dt;

}

public static decimal ToDecimal(object obj)//入口

{ return ToDecimal(obj, 0); }

public static DateTime ToDateTime(object obj) //入口

{

return ToDateTime(obj, DateTime.Now);

}

## 判断接收的值是否是decimal类型

/// <param name="obj">接收的值</param>

/// <param name="dafaultUalue">初始值</param>

public static decimal ToDecimal(object obj, decimal dafaultUalue)

{

decimal result=decimal.MinValue;

decimal.TryParse(obj.ToString(), out result);

if (result == decimal.MinValue)

return dafaultUalue;

else

return result;

}

## 转换特殊字符

/// <param name="str">字符串</param>

public static string FunStr(string str)

{

str = str.Replace("'", "‘");

str = str.Replace(" ", "&nbsp;");

str = str.Trim();

if (str.Trim().ToString() == "")

str = "";

return str;

}

}

## 星期英文转中文

public class FormatConversion

{

public static string GetWeek(DateTime dt)

{

string weekstr = dt.DayOfWeek.ToString();

switch (weekstr)

{

case "Monday": weekstr = "星期一"; break;

case "Tuesday": weekstr = "星期二"; break;

case "Wednesday": weekstr = "星期三"; break;

case "Thursday": weekstr = "星期四"; break;

case "Friday": weekstr = "星期五"; break;

case "Saturday": weekstr = "星期六"; break;

case "Sunday": weekstr = "星期日"; break;

}

return weekstr;

}

}

## 生成20位的主键数

/// </summary>如AB200807021438250001

/// <param name="str">主键字符串的前两位</param>

/// <param name="str">例如调用接口：PKey.CreateKeyStr("RO");返回RO打头的20位字符</param>

/// <returns></returns>

public static string CreateKeyStr(string str)

{

StringBuilder strKey = new StringBuilder();

Random rd = new Random(System.DateTime.Now.Millisecond);

string strDate;

string strRd;

str = str.ToUpper().Replace(" ", "") + "AA";

str = str.Substring(0, 2);

strKey.Append(str);

strDate = DateTime.Now.ToString("yyyyMMdd") + DateTime.Now.ToString("HH:mm:ss").Replace(":", "");

strKey.Append(strDate);

strRd = "0000" + rd.Next(1, 10000);

strRd = strRd.Substring(strRd.Length - 4, 4);

strKey.Append(strRd);

return strKey.ToString();

}

## 系统重启

/// </summary>

public static void Reset()

{

Application.ExitThread();

System.Threading.Thread thtmp = new System.Threading.Thread(new System.Threading.ParameterizedThreadStart(run));

object appName = Application.ExecutablePath;

//System.Threading.Thread.Sleep(2000);

thtmp.Start(appName);

}

private static void run(Object obj)

{

System.Diagnostics.Process ps = new System.Diagnostics.Process();

ps.StartInfo.FileName = obj.ToString();

ps.Start();

}

## 导出DataGridView数据 长城修改

/// <param name="dgv">DataGridView控件</param>

public void outPutDataGridViewData(DataGridView dgv, string txtTitle)

{

SaveFileDialog save = new SaveFileDialog();

//save.Filter = "user files(" + filter + ")|" + filter";

save.Filter = "文本格式.txt|.txt|表格.xls|.xls|文档.doc|.doc|RLC格式.rlc|.rlc";

save.Title = "导出文件到";

if (save.ShowDialog() == DialogResult.OK)

{

string padStr = "";//用于填充空格

Stream myStream = save.OpenFile();

StreamWriter sw = new StreamWriter(myStream, System.Text.Encoding.GetEncoding("GB2312"));

try

{

int[] temp = new int[dgv.Columns.Count];

//for (int index = 0; index < dgv.Columns.Count; index++)

for (int index = 1; index < dgv.Columns.Count; index++)

{

temp[index] = GetMaxWidth(dgv, index);

}

//写标题

sw.WriteLine(txtTitle);

//写数据字段

string tempTitle = "";

//for (int i = 0; i < dgv.Columns.Count; i++)

for (int i = 1; i < dgv.Columns.Count; i++)

{

if (save.FilterIndex == 4)//如果是表格xls格式

{

//if (i > 0)

if (i > 1)

{

tempTitle += "\t";

}

tempTitle += dgv.Columns[i].Name;

}

else//如果是doc、txt等格式

{

//if (i == 0)

if (i == 1)

{

tempTitle += dgv.Columns[i].Name;

}

else

{

if (temp[i - 1] - Encoding.Default.GetBytes(dgv.Columns[i - 1].Name).Length < 0)

{

tempTitle += padStr.PadRight(2, ' ');

}

else

{

tempTitle += padStr.PadRight(temp[i - 1] - Encoding.Default.GetBytes(dgv.Columns[i - 1].Name).Length + 2, ' ');

}

tempTitle += dgv.Columns[i].Name;

}

}

}

sw.WriteLine(tempTitle);

//循环写内容

for (int j = 0; j < dgv.Rows.Count; j++)

{

string tempStr = "";

//for (int k = 0; k < dgv.Columns.Count; k++)

for (int k = 1; k < dgv.Columns.Count; k++)

{

if (save.FilterIndex == 4)//如果是表格xls格式

{

//if (k > 0)

if (k > 1)

{

tempStr += "\t";

}

tempStr += "'" + dgv.Rows[j].Cells[k].Value.ToString().Trim();

}

else //如果是doc、txt等格式

{

//if (k == 0)

if (k == 1)

{

tempStr += dgv.Rows[j].Cells[k].Value.ToString();

}

else

{

if (temp[k - 1] - Encoding.Default.GetBytes(dgv.Columns[k - 1].Name).Length < 0)

{

tempStr += padStr.PadRight(Encoding.Default.GetBytes(dgv.Columns[k - 1].Name).Length - Encoding.Default.GetBytes(dgv.Rows[j].Cells[k - 1].Value.ToString().Trim()).Length + 2, ' ');

}

else

{

tempStr += padStr.PadRight(temp[k - 1] - Encoding.Default.GetBytes(dgv.Rows[j].Cells[k - 1].Value.ToString().Trim()).Length + 2, ' ');

}

tempStr += dgv.Rows[j].Cells[k].Value.ToString().Trim();

}

}

}

sw.WriteLine(tempStr);

}

MessageBox.Show("导出数据成功！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);

sw.Close();

myStream.Close();

}

catch (Exception ee)

{

MessageBox.Show(ee.Message);

}

finally

{

sw.Close();

myStream.Close();

}

}

}

/// 配合上面方法得到DataGridView每列最大的宽度

/// </summary>

/// <returns></returns>

private int GetMaxWidth(DataGridView dgv, int columnIndex)

{

int max = 0;

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

{

max = max > Encoding.Default.GetBytes(dgv.Rows[i].Cells[columnIndex].Value.ToString().Trim()).Length ? max : Encoding.Default.GetBytes(dgv.Rows[i].Cells[columnIndex].Value.ToString().Trim()).Length;

}

return max;

}

}

## Winform添加图片验证码

string verification = string.Empty;//定义验证码变量，用来比较用户输入的验证码是否正确

public login()

{

InitializeComponent();

UpdateVerifyCode();//更新验证码

}

#region 动态生成验证码

// 窗体中，需要自己增加的控件：

//一个picturebox控件pbVerifyCode，

//一个按钮butUpdateVerifyCode（看不清楚，换一张图）；

private const int iVerifyCodeLength = 5;

//随机码

private String strVerifyCode = "";

//更新验证码

private void UpdateVerifyCode()

{

strVerifyCode = CreateRandomCode(iVerifyCodeLength);

CreateImage(strVerifyCode);

}

private string CreateRandomCode(int iLength)

{

int rand;

char code;

string randomCode = String.Empty;

//生成一定长度的验证码

System.Random random = new Random();

for (int i = 0; i < iLength; i++)

{

rand = random.Next();

if (rand % 3 == 0)

{

code = (char)('A' + (char)(rand % 26));

}

else

{

code = (char)('0' + (char)(rand % 10));

}

randomCode += code.ToString();

verification = randomCode;

}

return randomCode;

}

/// 创建随机码图片

private void CreateImage(string strVerifyCode)

{

try

{

int iRandAngle = 45; //随机转动角度

int iMapWidth = (int)(strVerifyCode.Length \* 21);

Bitmap map = new Bitmap(iMapWidth, 28); //创建图片背景

Graphics graph = Graphics.FromImage(map);

graph.Clear(Color.AliceBlue);//清除画面，填充背景

graph.DrawRectangle(new Pen(Color.Black, 500), 0, 0, map.Width - 1, map.Height-1);//画一个边框

graph.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;//模式

Random rand = new Random();

//背景噪点生成

Pen blackPen = new Pen(Color.LightGray, 0);

for (int i = 0; i < 50; i++)

{

int x = rand.Next(0, map.Width);

int y = rand.Next(0, map.Height);

graph.DrawRectangle(blackPen, x, y, 1, 1);

}

//验证码旋转，防止机器识别

char[] chars = strVerifyCode.ToCharArray();//拆散字符串成单字符数组

//文字距中

StringFormat format = new StringFormat(StringFormatFlags.NoClip);

format.Alignment = StringAlignment.Center;

format.LineAlignment = StringAlignment.Center;

//定义颜色

Color[] c = { Color.Black, Color.Red, Color.DarkBlue, Color.Green, Color.Orange, Color.Brown, Color.DarkCyan, Color.Purple };

//定义字体

string[] font = { "Verdana", "Microsoft Sans Serif", "Comic Sans MS", "Arial", "宋体" };

for (int i = 0; i < chars.Length; i++)

{

int cindex = rand.Next(7);

int findex = rand.Next(5);

Font f = new System.Drawing.Font(font[findex], 13, System.Drawing.FontStyle.Bold);//字体样式(参数2为字体大小)

Brush b = new System.Drawing.SolidBrush(c[cindex]);

Point dot = new Point(16, 16);

float angle = rand.Next(-iRandAngle, iRandAngle);//转动的度数

graph.TranslateTransform(dot.X, dot.Y);//移动光标到指定位置

graph.RotateTransform(angle);

graph.DrawString(chars[i].ToString(), f, b, 1, 1, format);

graph.RotateTransform(-angle);//转回去

graph.TranslateTransform(2, -dot.Y);//移动光标到指定位置

}

pbVerifyCode.Image = map;

}

catch (ArgumentException)

{

MessageBox.Show("创建图片错误。");

}

}

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

{

UpdateVerifyCode();

}

if (txt\_yanzheng.Text.Trim() != verification)

{

MessageBox.Show("验证码输入有误,请重新输入");

UpdateVerifyCode();

txt\_yanzheng.Text = "";

}

## 添加日志

/// 写日志信息

/// <param name="LogMsg">日志信息</param>

public static void LogRecord(string LogMsg)

{

StreamWriter tmpStreamWriter = null; // 定义写文件流

string strPath = GetRunningPath() + "Log";

try

{

if (!Directory.Exists(strPath))

{

Directory.CreateDirectory(strPath);//建立文件夹

}

strPath = strPath + "\\ZIBO\_" + DateTime.Now.ToString("yyyyMMdd") + ".log";

// 初始化文件流

tmpStreamWriter = new StreamWriter(strPath, true, Encoding.GetEncoding("GB2312"));//建立文件

// 写入信息

tmpStreamWriter.WriteLine("[" + DateTime.Now.ToString("HH:mm:ss:fff") + "]：" + LogMsg);

}

catch

{

}

finally

{

try

{

tmpStreamWriter.Close();

}

catch

{ }

}

}

/// 获得程序运行路径

public static string GetRunningPath()

{

string strDebugPath = System.Windows.Forms.Application.StartupPath; // 获得执行路径

if (!strDebugPath.EndsWith("\\"))

{

strDebugPath = strDebugPath + "\\";

}

return strDebugPath;

}