# 公共类方法

## 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;

}

## 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,}$");

}

}

## 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;

}

}

## 字符串特定格式的转换

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;

}

}