sntetwt

C#調用系統打印機和收銀錢箱

打印示例：

```
StringBuilder builder = new StringBuilder();
builder.AppendLine("--------------打印測試--------------");
string PrintName = PrinterHelper.GetDeaultPrinterName();
PrinterHelper.SendStringToPrinter(PrintName, builder.ToString());
```

資源文件：**資江58、80打印機驅動.rar**

```
using System;
using System.IO;
using System.Text;
using System.Runtime.InteropServices;
using System.Security;
using System.ComponentModel;

namespace Micro.Common
{
    /// <summary>
    /// 打印功能
    /// </summary>
    public class PrinterHelper
    {
        private PrinterHelper() { }
        #region API声明
        [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Auto)]
        internal struct structPrinterDefaults
        {
            [MarshalAs(UnmanagedType.LPTStr)]
            public String pDatatype;
            public IntPtr pDevMode;
            [MarshalAs(UnmanagedType.I4)]
            public int DesiredAccess;
        };

        [DllImport("winspool.Drv", EntryPoint = "OpenPrinter", SetLastError = true, CharSet =
CharSet.Unicode, ExactSpelling = false, CallingConvention = CallingConvention.StdCall),
SuppressUnmanagedCodeSecurityAttribute()]
        internal static extern bool OpenPrinter([MarshalAs(UnmanagedType.LPTStr)] string
printerName, out IntPtr phPrinter, ref structPrinterDefaults pd);
```

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

```
        [DllImport("winspool.Drv", EntryPoint = "ClosePrinter", SetLastError = true, CharSet =
CharSet.Unicode, ExactSpelling = false, CallingConvention = CallingConvention.StdCall),
SuppressUnmanagedCodeSecurityAttribute()]
        internal static extern bool ClosePrinter(IntPtr phPrinter);

        [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Auto)]
        internal struct structSize
        {
            public Int32 width;
            public Int32 height;
        }

        [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Auto)]
        internal struct structRect
        {
            public Int32 left;
            public Int32 top;
            public Int32 right;
            public Int32 bottom;
        }

        [StructLayout(LayoutKind.Explicit, CharSet = CharSet.Unicode)]
        internal struct FormInfo1
        {
            [FieldOffset(0), MarshalAs(UnmanagedType.I4)]
            public uint Flags;
            [FieldOffset(4), MarshalAs(UnmanagedType.LPWStr)]
            public String pName;
            [FieldOffset(8)]
            public structSize Size;
            [FieldOffset(16)]
            public structRect ImageableArea;
        };
```

5.C#實現判斷一遠程文件（圖片等）是否存在

6.WebApi認證刷選器獲取Post參數

7.C# 取得對象屬性類型

8.前端架構

9.WebAPI中路由參數中包含字符-點“.”

10.Web API 授權篩選器

我的標籤

httpHandlers和httpModules接口介紹(4)

httpHandlers和httpModules接口介紹(3)(1)

httpHandlers和httpModules接口介紹(4)(1)

```
[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi)]
internal struct structDevMode
{
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 32)]
    public String
     dmDeviceName;
    [MarshalAs(UnmanagedType.U2)]
    public short dmSpecVersion;
    [MarshalAs(UnmanagedType.U2)]
    public short dmDriverVersion;
    [MarshalAs(UnmanagedType.U2)]
    public short dmSize;
    [MarshalAs(UnmanagedType.U2)]
    public short dmDriverExtra;
    [MarshalAs(UnmanagedType.U4)]
    public int dmFields;
    [MarshalAs(UnmanagedType.I2)]
    public short dmOrientation;
    [MarshalAs(UnmanagedType.I2)]
    public short dmPaperSize;
    [MarshalAs(UnmanagedType.I2)]
    public short dmPaperLength;
    [MarshalAs(UnmanagedType.I2)]
    public short dmPaperWidth;
    [MarshalAs(UnmanagedType.I2)]
    public short dmScale;
    [MarshalAs(UnmanagedType.I2)]
    public short dmCopies;
    [MarshalAs(UnmanagedType.I2)]
    public short dmDefaultSource;
    [MarshalAs(UnmanagedType.I2)]
    public short dmPrintQuality;
```

httpHandlers和httpModules接口介紹(5)
(1)
ASP.NET的運行原理與運行機制(1)
WEB前端開發規範文檔(1)
高效率(1)
簡潔(1)
CSS代碼優化原則(1)
瀏覽器兼容(1)
更多

```
        [MarshalAs(UnmanagedType.I2)]
        public short dmColor;
        [MarshalAs(UnmanagedType.I2)]
        public short dmDuplex;
        [MarshalAs(UnmanagedType.I2)]
        public short dmYResolution;
        [MarshalAs(UnmanagedType.I2)]
        public short dmTTOption;
        [MarshalAs(UnmanagedType.I2)]
        public short dmCollate;
        [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 32)]
        public String dmFormName;
        [MarshalAs(UnmanagedType.U2)]
        public short dmLogPixels;
        [MarshalAs(UnmanagedType.U4)]
        public int dmBitsPerPel;
        [MarshalAs(UnmanagedType.U4)]
        public int dmPelsWidth;
        [MarshalAs(UnmanagedType.U4)]
        public int dmPelsHeight;
        [MarshalAs(UnmanagedType.U4)]
        public int dmNup;
        [MarshalAs(UnmanagedType.U4)]
        public int dmDisplayFrequency;
        [MarshalAs(UnmanagedType.U4)]
        public int dmICMMethod;
        [MarshalAs(UnmanagedType.U4)]
        public int dmICMIntent;
        [MarshalAs(UnmanagedType.U4)]
        public int dmMediaType;
        [MarshalAs(UnmanagedType.U4)]
        public int dmDitherType;
        [MarshalAs(UnmanagedType.U4)]
```

隨筆分類 (561)

.net控件(12)

ado.net(30)

asp.net(134)

C#(64)

css(17)

更多

隨筆檔案 (587)

2020年4月(1)

2020年3月(1)

2020年1月(4)

2019年11月(2)

```
            public int dmReserved1;
            [MarshalAs(UnmanagedType.U4)]
            public int dmReserved2;
        }

        [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Auto)]
        internal struct PRINTER_INFO_9
        {
            public IntPtr pDevMode;
        }

        [DllImport("winspool.Drv", EntryPoint = "AddFormW", SetLastError = true, CharSet =
CharSet.Unicode, ExactSpelling = true,
                   CallingConvention = CallingConvention.StdCall),
SuppressUnmanagedCodeSecurityAttribute()]
        internal static extern bool AddForm(IntPtr phPrinter, [MarshalAs(UnmanagedType.I4)] int
level, ref FormInfo1 form);

        [DllImport("winspool.Drv", EntryPoint = "DeleteForm", SetLastError = true, CharSet =
CharSet.Unicode, ExactSpelling = false, CallingConvention = CallingConvention.StdCall),
                SuppressUnmanagedCodeSecurityAttribute()]
        internal static extern bool DeleteForm(IntPtr phPrinter, [MarshalAs(UnmanagedType.LPTStr)]
string pName);

        [DllImport("kernel32.dll", EntryPoint = "GetLastError", SetLastError = false,
ExactSpelling = true, CallingConvention = CallingConvention.StdCall),
SuppressUnmanagedCodeSecurityAttribute()]
        internal static extern Int32 GetLastError();

        [DllImport("GDI32.dll", EntryPoint = "CreateDC", SetLastError = true, CharSet =
CharSet.Unicode, ExactSpelling = false, CallingConvention = CallingConvention.StdCall),
SuppressUnmanagedCodeSecurityAttribute()]
```

2019年10月(3)

更多

文章分類 (1)

貴源產品介紹(1)

文章檔案 (1)

2014年1月(1)

相冊 (2)

楊秀徐(2)

技術網址

在線客服聊天系統教學視頻以及程序源代碼和模板

```
        internal static extern IntPtr CreateDC([MarshalAs(UnmanagedType.LPTStr)] string pDrive,
[MarshalAs(UnmanagedType.LPTStr)] string pName, [MarshalAs(UnmanagedType.LPTStr)] string pOutput,
ref structDevMode pDevMode);

        [DllImport("GDI32.dll", EntryPoint = "ResetDC", SetLastError = true,
          CharSet = CharSet.Unicode, ExactSpelling = false, CallingConvention =
CallingConvention.StdCall), SuppressUnmanagedCodeSecurityAttribute()]
        internal static extern IntPtr ResetDC(IntPtr hDC, ref structDevMode pDevMode);

        [DllImport("GDI32.dll", EntryPoint = "DeleteDC", SetLastError = true, CharSet =
CharSet.Unicode, ExactSpelling = false,
                    CallingConvention = CallingConvention.StdCall),
SuppressUnmanagedCodeSecurityAttribute()]
        internal static extern bool DeleteDC(IntPtr hDC);

        [DllImport("winspool.Drv", EntryPoint = "SetPrinterA", SetLastError = true,
          CharSet = CharSet.Auto, ExactSpelling = true, CallingConvention =
CallingConvention.StdCall), SuppressUnmanagedCodeSecurityAttribute()]
        internal static extern bool SetPrinter(IntPtr hPrinter, [MarshalAs(UnmanagedType.I4)] int
level, IntPtr pPrinter, [MarshalAs(UnmanagedType.I4)] int command);

        [DllImport("winspool.Drv", EntryPoint = "DocumentPropertiesA", SetLastError = true,
ExactSpelling = true, CallingConvention = CallingConvention.StdCall)]
        internal static extern int DocumentProperties(IntPtr hwnd, IntPtr hPrinter,
[MarshalAs(UnmanagedType.LPStr)] string pDeviceName, IntPtr pDevModeOutput, IntPtr pDevModeInput,
int fMode);

        [DllImport("winspool.Drv", EntryPoint = "GetPrinterA", SetLastError = true, ExactSpelling
= true, CallingConvention = CallingConvention.StdCall)]
        internal static extern bool GetPrinter(IntPtr hPrinter, int dwLevel, IntPtr pPrinter, int
dwBuf, out int dwNeeded);

        [Flags]
```

```
        internal enum SendMessageTimeoutFlags : uint
        {
            SMTO_NORMAL = 0x0000,
            SMTO_BLOCK = 0x0001,
            SMTO_ABORTIFHUNG = 0x0002,
            SMTO_NOTIMEOUTIFNOTHUNG = 0x0008
        }
        const int WM_SETTINGCHANGE = 0x001A;
        const int HWND_BROADCAST = 0xffff;

        [DllImport("user32.dll", SetLastError = true, CharSet = CharSet.Auto)]
        internal static extern IntPtr SendMessageTimeout(IntPtr windowHandle, uint Msg, IntPtr
wParam, IntPtr lParam, SendMessageTimeoutFlags flags, uint timeout, out IntPtr result);

        //EnumPrinters用到的函数和结构体
        [DllImport("winspool.drv", CharSet = CharSet.Auto, SetLastError = true)]
        private static extern bool EnumPrinters(PrinterEnumFlags Flags, string Name, uint Level,
IntPtr pPrinterEnum, uint cbBuf, ref uint pcbNeeded, ref uint pcReturned);

        [StructLayout(LayoutKind.Sequential)]
        internal struct PRINTER_INFO_2
        {
            public string pServerName;
            public string pPrinterName;
            public string pShareName;
            public string pPortName;
            public string pDriverName;
            public string pComment;
            public string pLocation;
            public IntPtr pDevMode;
            public string pSepFile;
            public string pPrintProcessor;
            public string pDatatype;
```

598)

4. WPF调用图片路径，或资源图片(52889)

5. C#删除字符串最后一个字符的几种方法(45698)

评论排行榜

1. WPF调用图片路径，或资源图片(9)

2. Windows 搭建 nginx rtmp服务器(5)

3. jQuery动画animate方法使用介绍(3)

4. 各大型网站用的全局样式(3)

5. 浮点数正则表达式(2)

推荐排行榜

1. WPF调用图片路径，或资源图片(11)

```
        public string pParameters;
        public IntPtr pSecurityDescriptor;
        public uint Attributes;
        public uint Priority;
        public uint DefaultPriority;
        public uint StartTime;
        public uint UntilTime;
        public uint Status;
        public uint cJobs;
        public uint AveragePPM;
    }

    [FlagsAttribute]
    internal enum PrinterEnumFlags
    {
        PRINTER_ENUM_DEFAULT = 0x00000001,
        PRINTER_ENUM_LOCAL = 0x00000002,
        PRINTER_ENUM_CONNECTIONS = 0x00000004,
        PRINTER_ENUM_FAVORITE = 0x00000004,
        PRINTER_ENUM_NAME = 0x00000008,
        PRINTER_ENUM_REMOTE = 0x00000010,
        PRINTER_ENUM_SHARED = 0x00000020,
        PRINTER_ENUM_NETWORK = 0x00000040,
        PRINTER_ENUM_EXPAND = 0x00004000,
        PRINTER_ENUM_CONTAINER = 0x00008000,
        PRINTER_ENUM_ICONMASK = 0x00ff0000,
        PRINTER_ENUM_ICON1 = 0x00010000,
        PRINTER_ENUM_ICON2 = 0x00020000,
        PRINTER_ENUM_ICON3 = 0x00040000,
        PRINTER_ENUM_ICON4 = 0x00080000,
        PRINTER_ENUM_ICON5 = 0x00100000,
        PRINTER_ENUM_ICON6 = 0x00200000,
        PRINTER_ENUM_ICON7 = 0x00400000,
```

2. 数据库中字段类型对应的C#中的数据类型(5)

3. C# .Net计算函数执行的时间(4)

4. jQuery动画animate方法使用介绍(4)

5. web调用本地exe应用程序并传入参数(3)

最新评论

1. Re:WPF调用图片路径，或资源图片

Html文件怎么读取 我想直接获取它的路径 然后使用 System.Diagnostics.Process.Start(path); 打开

--呆呆牛

2. Re:WPF 实现窗体拖动

居然可以这样，WPF还是强大

--会长

```
        PRINTER_ENUM_ICON8 = 0x00800000,
        PRINTER_ENUM_HIDE = 0x01000000
    }

    //打印机状态
    [FlagsAttribute]
    internal enum PrinterStatus
    {
        PRINTER_STATUS_BUSY = 0x00000200,
        PRINTER_STATUS_DOOR_OPEN = 0x00400000,
        PRINTER_STATUS_ERROR = 0x00000002,
        PRINTER_STATUS_INITIALIZING = 0x00008000,
        PRINTER_STATUS_IO_ACTIVE = 0x00000100,
        PRINTER_STATUS_MANUAL_FEED = 0x00000020,
        PRINTER_STATUS_NO_TONER = 0x00040000,
        PRINTER_STATUS_NOT_AVAILABLE = 0x00001000,
        PRINTER_STATUS_OFFLINE = 0x00000080,
        PRINTER_STATUS_OUT_OF_MEMORY = 0x00200000,
        PRINTER_STATUS_OUTPUT_BIN_FULL = 0x00000800,
        PRINTER_STATUS_PAGE_PUNT = 0x00080000,
        PRINTER_STATUS_PAPER_JAM = 0x00000008,
        PRINTER_STATUS_PAPER_OUT = 0x00000010,
        PRINTER_STATUS_PAPER_PROBLEM = 0x00000040,
        PRINTER_STATUS_PAUSED = 0x00000001,
        PRINTER_STATUS_PENDING_DELETION = 0x00000004,
        PRINTER_STATUS_PRINTING = 0x00000400,
        PRINTER_STATUS_PROCESSING = 0x00004000,
        PRINTER_STATUS_TONER_LOW = 0x00020000,
        PRINTER_STATUS_USER_INTERVENTION = 0x00100000,
        PRINTER_STATUS_WAITING = 0x20000000,
        PRINTER_STATUS_WARMING_UP = 0x00010000
    }
```

3. Re:解决WPF下popup不随着window一起移动的问题

解决WPF下popup不随着window一起移动的问题

--流畅的心情

4. Re:JS中的HTML片段

<script id="weather" type="text/html"> <h1>Hello, world!</h1> </script> 这种写法如何多页面重用呢...

--胡正

5. Re:WPF加載HTML、WPF與JavaScript交互

html界面HandleMessage 報找不到的錯誤怎麼辦

--空心蘿蔔2

```
        //GetDefaultPrinter用到的API函数说明
        [DllImport("winspool.drv", CharSet = CharSet.Auto, SetLastError = true)]
        internal static extern bool GetDefaultPrinter(StringBuilder pszBuffer, ref int size);

        //SetDefaultPrinter用到的API函数声明
        [DllImport("winspool.drv", CharSet = CharSet.Auto, SetLastError = true)]
        internal static extern bool SetDefaultPrinter(string Name);

        //EnumFormsA用到的函数声明，应该和EnumPrinters类似
        [DllImport("winspool.drv", EntryPoint = "EnumForms")]
        internal static extern int EnumFormsA(IntPtr hPrinter, int Level, ref byte pForm, int
cbBuf, ref int pcbNeeded, ref int pcReturned);

        #endregion   API声明
        internal static int GetPrinterStatusInt(string PrinterName)
        {
            int intRet = 0;
            IntPtr hPrinter;
            structPrinterDefaults defaults = new structPrinterDefaults();
            if (OpenPrinter(PrinterName, out hPrinter, ref defaults))
            {
                int cbNeeded = 0;
                bool bolRet = GetPrinter(hPrinter, 2, IntPtr.Zero, 0, out cbNeeded);
                if (cbNeeded > 0)
                {
                    IntPtr pAddr = Marshal.AllocHGlobal((int)cbNeeded);
                    bolRet = GetPrinter(hPrinter, 2, pAddr, cbNeeded, out cbNeeded);
                    if (bolRet)
                    {
                        PRINTER_INFO_2 Info2 = new PRINTER_INFO_2();

                        Info2 = (PRINTER_INFO_2)Marshal.PtrToStructure(pAddr,
typeof(PRINTER_INFO_2));
```

```
                    intRet = System.Convert.ToInt32(Info2.Status);
                }
                Marshal.FreeHGlobal(pAddr);
            }
            ClosePrinter(hPrinter);
        }
        return intRet;
    }
    internal static PRINTER_INFO_2[] EnumPrintersByFlag(PrinterEnumFlags Flags)
    {
        uint cbNeeded = 0;
        uint cReturned = 0;
        bool ret = EnumPrinters(PrinterEnumFlags.PRINTER_ENUM_LOCAL, null, 2, IntPtr.Zero, 0,
ref cbNeeded, ref cReturned);
        IntPtr pAddr = Marshal.AllocHGlobal((int)cbNeeded);
        ret = EnumPrinters(PrinterEnumFlags.PRINTER_ENUM_LOCAL, null, 2, pAddr, cbNeeded, ref
cbNeeded, ref cReturned);
        if (ret)
        {
            PRINTER_INFO_2[] Info2 = new PRINTER_INFO_2[cReturned];
            int offset = pAddr.ToInt32();
            for (int i = 0; i < cReturned; i++)
            {
                Info2[i].pServerName = Marshal.PtrToStringAuto(Marshal.ReadIntPtr(new
IntPtr(offset)));
                offset += 4;
                Info2[i].pPrinterName = Marshal.PtrToStringAuto(Marshal.ReadIntPtr(new
IntPtr(offset)));
                offset += 4;
                Info2[i].pShareName = Marshal.PtrToStringAuto(Marshal.ReadIntPtr(new
IntPtr(offset)));
                offset += 4;
```

```
                    Info2[i].pPortName = Marshal.PtrToStringAuto(Marshal.ReadIntPtr(new
IntPtr(offset)));
                    offset += 4;
                    Info2[i].pDriverName = Marshal.PtrToStringAuto(Marshal.ReadIntPtr(new
IntPtr(offset)));
                    offset += 4;
                    Info2[i].pComment = Marshal.PtrToStringAuto(Marshal.ReadIntPtr(new
IntPtr(offset)));
                    offset += 4;
                    Info2[i].pLocation = Marshal.PtrToStringAuto(Marshal.ReadIntPtr(new
IntPtr(offset)));
                    offset += 4;
                    Info2[i].pDevMode = Marshal.ReadIntPtr(new IntPtr(offset));
                    offset += 4;
                    Info2[i].pSepFile = Marshal.PtrToStringAuto(Marshal.ReadIntPtr(new
IntPtr(offset)));
                    offset += 4;
                    Info2[i].pPrintProcessor = Marshal.PtrToStringAuto(Marshal.ReadIntPtr(new
IntPtr(offset)));
                    offset += 4;
                    Info2[i].pDatatype = Marshal.PtrToStringAuto(Marshal.ReadIntPtr(new
IntPtr(offset)));
                    offset += 4;
                    Info2[i].pParameters = Marshal.PtrToStringAuto(Marshal.ReadIntPtr(new
IntPtr(offset)));
                    offset += 4;
                    Info2[i].pSecurityDescriptor = Marshal.ReadIntPtr(new IntPtr(offset));
                    offset += 4;
                    Info2[i].Attributes = (uint)Marshal.ReadIntPtr(new IntPtr(offset));
                    offset += 4;
                    Info2[i].Priority = (uint)Marshal.ReadInt32(new IntPtr(offset));
                    offset += 4;
                    Info2[i].DefaultPriority = (uint)Marshal.ReadInt32(new IntPtr(offset));
```

```
                offset += 4;
                Info2[i].StartTime = (uint)Marshal.ReadInt32(new IntPtr(offset));
                offset += 4;
                Info2[i].UntilTime = (uint)Marshal.ReadInt32(new IntPtr(offset));
                offset += 4;
                Info2[i].Status = (uint)Marshal.ReadInt32(new IntPtr(offset));
                offset += 4;
                Info2[i].cJobs = (uint)Marshal.ReadInt32(new IntPtr(offset));
                offset += 4;
                Info2[i].AveragePPM = (uint)Marshal.ReadInt32(new IntPtr(offset));
                offset += 4;
            }
            Marshal.FreeHGlobal(pAddr);
            return Info2;
        }
        else
        {
            return new PRINTER_INFO_2[0];
        }
    }
    #region 获取当前指定打印机的状态
    /// </summary>
    /// 获取当前指定打印机的状态
    /// </summary>
    /// <param name="PrinterName">打印机名称</param>
    /// <returns>打印机状态描述</returns>

    public static string GetPrinterStatus(string PrinterName)
    {
        int intValue = GetPrinterStatusInt(PrinterName);
        string strRet = string.Empty;
        switch (intValue)
        {
```

```
                case 0:
                    strRet = "准备就绪(Ready)";
                    break;
                case 0x00000200:
                    strRet = "忙(Busy)";
                    break;
                case 0x00400000:
                    strRet = "门被打开(Printer Door Open)";
                    break;
                case 0x00000002:
                    strRet = "错误(Printer Error)";
                    break;
                case 0x0008000:
                    strRet = "正在初始化(Initializing)";
                    break;
                case 0x00000100:
                    strRet = "正在输入或输出(I/O Active)";
                    break;
                case 0x00000020:
                    strRet = "手工送纸(Manual Feed)";
                    break;
                case 0x00040000:
                    strRet = "无墨粉(No Toner)";
                    break;
                case 0x00001000:
                    strRet = "不可用(Not Available)";
                    break;
                case 0x00000080:
                    strRet = "脱机(Off Line)";
                    break;
                case 0x00200000:
                    strRet = "内存溢出(Out of Memory)";
                    break;
```

```
                case 0x00000800:
                    strRet = "输出口已满（Output Bin Full）";
                    break;
                case 0x00080000:
                    strRet = "当前页无法打印（Page Punt）";
                    break;
                case 0x00000008:
                    strRet = "塞纸（Paper Jam）";
                    break;
                case 0x00000010:
                    strRet = "打印纸用完（Paper Out）";
                    break;
                case 0x00000040:
                    strRet = "纸张问题（Page Problem）";
                    break;
                case 0x00000001:
                    strRet = "暂停（Paused）";
                    break;
                case 0x00000004:
                    strRet = "正在删除（Pending Deletion）";
                    break;
                case 0x00000400:
                    strRet = "正在打印（Printing）";
                    break;
                case 0x00004000:
                    strRet = "正在处理（Processing）";
                    break;
                case 0x00020000:
                    strRet = "墨粉不足（Toner Low）";
                    break;
                case 0x00100000:
                    strRet = "需要用户干预（User Intervention）";
                    break;
```

```
                case 0x20000000:
                    strRet = "等待（Waiting）";
                    break;
                case 0x00010000:
                    strRet = "正在准备（Warming Up）";
                    break;
                default:
                    strRet = "未知状态（Unknown Status）";
                    break;
            }
            return strRet;
        }
        #endregion 获取当前指定打印机的状态

        #region 删除已经存在的自定义纸张
        /**/
        /// <summary>
        /// 删除已经存在的自定义纸张
        /// </summary>
        /// <param name="PrinterName">打印机名称</param>
        /// <param name="PaperName">纸张名称</param>
        public static void DeleteCustomPaperSize(string PrinterName, string PaperName)
        {
            const int PRINTER_ACCESS_USE = 0x00000008;
            const int PRINTER_ACCESS_ADMINISTER = 0x00000004;

            structPrinterDefaults defaults = new structPrinterDefaults();
            defaults.pDatatype = null;
            defaults.pDevMode = IntPtr.Zero;
            defaults.DesiredAccess = PRINTER_ACCESS_ADMINISTER | PRINTER_ACCESS_USE;

            IntPtr hPrinter = IntPtr.Zero;
```

```
            //打开打印机
            if (OpenPrinter(PrinterName, out hPrinter, ref defaults))
            {
                try
                {
                    DeleteForm(hPrinter, PaperName);
                    ClosePrinter(hPrinter);
                }
                catch
                {

                }
            }
        }
        #endregion 删除已经存在的自定义纸张

        #region 指定的打印机设置以mm为单位的自定义纸张(Form)
        /**/
        /// <summary>
        /// 指定的打印机设置以mm为单位的自定义纸张(Form)
        /// </summary>
        /// <param name="PrinterName">打印机名称</param>
        /// <param name="PaperName">Form名称</param>
        /// <param name="WidthInMm">以mm为单位的宽度</param>
        /// <param name="HeightInMm">以mm为单位的高度</param>
        public static void AddCustomPaperSize(string PrinterName, string PaperName, float
WidthInMm, float HeightInMm)
        {
            if (PlatformID.Win32NT == Environment.OSVersion.Platform)
            {
                const int PRINTER_ACCESS_USE = 0x00000008;
                const int PRINTER_ACCESS_ADMINISTER = 0x00000004;
                structPrinterDefaults defaults = new structPrinterDefaults();
```

```
                defaults.pDatatype = null;
                defaults.pDevMode = IntPtr.Zero;
                defaults.DesiredAccess = PRINTER_ACCESS_ADMINISTER | PRINTER_ACCESS_USE;
                IntPtr hPrinter = IntPtr.Zero;
                //打开打印机
                if (OpenPrinter(PrinterName, out hPrinter, ref defaults))
                {
                    try
                    {
                        //如果Form存在删除之
                        DeleteForm(hPrinter, PaperName);
                        //创建并初始化FORM_INFO_1
                        FormInfo1 formInfo = new FormInfo1();
                        formInfo.Flags = 0;
                        formInfo.pName = PaperName;
                        formInfo.Size.width = (int)(WidthInMm * 1000.0);
                        formInfo.Size.height = (int)(HeightInMm * 1000.0);
                        formInfo.ImageableArea.left = 0;
                        formInfo.ImageableArea.right = formInfo.Size.width;
                        formInfo.ImageableArea.top = 0;
                        formInfo.ImageableArea.bottom = formInfo.Size.height;
                        if (!AddForm(hPrinter, 1, ref formInfo))
                        {
                            StringBuilder strBuilder = new StringBuilder();
                            strBuilder.AppendFormat("向打印机 {1} 添加自定义纸张 {0} 失败！错误代号：
{2}",
                             PaperName, PrinterName, GetLastError());
                            throw new ApplicationException(strBuilder.ToString());
                        }

                        //初始化
                        const int DM_OUT_BUFFER = 2;
                        const int DM_IN_BUFFER = 8;
```

```
                        structDevMode devMode = new structDevMode();
                        IntPtr hPrinterInfo, hDummy;
                        PRINTER_INFO_9 printerInfo;
                        printerInfo.pDevMode = IntPtr.Zero;
                        int iPrinterInfoSize, iDummyInt;


                        int iDevModeSize = DocumentProperties(IntPtr.Zero, hPrinter, PrinterName,
IntPtr.Zero, IntPtr.Zero, 0);

                        if (iDevModeSize < 0)
                            throw new ApplicationException("无法取得DEVMODE结构的大小！");

                        //分配缓冲
                        IntPtr hDevMode = Marshal.AllocCoTaskMem(iDevModeSize + 100);

                        //获取DEV_MODE指针
                        int iRet = DocumentProperties(IntPtr.Zero, hPrinter, PrinterName,
hDevMode, IntPtr.Zero, DM_OUT_BUFFER);

                        if (iRet < 0)
                            throw new ApplicationException("无法获得DEVMODE结构！");

                        //填充DEV_MODE
                        devMode = (structDevMode)Marshal.PtrToStructure(hDevMode,
devMode.GetType());


                        devMode.dmFields = 0x10000;

                        //FORM名称
                        devMode.dmFormName = PaperName;
```

```
                        Marshal.StructureToPtr(devMode, hDevMode, true);

                        iRet = DocumentProperties(IntPtr.Zero, hPrinter, PrinterName,
                         printerInfo.pDevMode, printerInfo.pDevMode, DM_IN_BUFFER |
DM_OUT_BUFFER);

                        if (iRet < 0)
                            throw new ApplicationException("无法为打印机设定打印方向！");

                        GetPrinter(hPrinter, 9, IntPtr.Zero, 0, out iPrinterInfoSize);
                        if (iPrinterInfoSize == 0)
                            throw new ApplicationException("调用GetPrinter方法失败！");

                        hPrinterInfo = Marshal.AllocCoTaskMem(iPrinterInfoSize + 100);

                        bool bSuccess = GetPrinter(hPrinter, 9, hPrinterInfo, iPrinterInfoSize,
out iDummyInt);

                        if (!bSuccess)
                            throw new ApplicationException("调用GetPrinter方法失败！");

                        printerInfo = (PRINTER_INFO_9)Marshal.PtrToStructure(hPrinterInfo,
printerInfo.GetType());
                        printerInfo.pDevMode = hDevMode;

                        Marshal.StructureToPtr(printerInfo, hPrinterInfo, true);

                        bSuccess = SetPrinter(hPrinter, 9, hPrinterInfo, 0);

                        if (!bSuccess)
                            throw new Win32Exception(Marshal.GetLastWin32Error(), "调用SetPrinter方
法失败，无法进行打印机设置！");
```

```
                    SendMessageTimeout(
                     new IntPtr(HWND_BROADCAST),
                     WM_SETTINGCHANGE,
                     IntPtr.Zero,
                     IntPtr.Zero,
                     PrinterHelper.SendMessageTimeoutFlags.SMTO_NORMAL,
                     1000,
                     out hDummy);
                }
                finally
                {
                    ClosePrinter(hPrinter);
                }
            }
            else
            {
                StringBuilder strBuilder = new StringBuilder();
                strBuilder.AppendFormat("无法打开打印机{0}, 错误代号: {1}",
                 PrinterName, GetLastError());
                throw new ApplicationException(strBuilder.ToString());
            }
        }
        else
        {
            structDevMode pDevMode = new structDevMode();
            IntPtr hDC = CreateDC(null, PrinterName, null, ref pDevMode);
            if (hDC != IntPtr.Zero)
            {
                const long DM_PAPERSIZE = 0x00000002L;
                const long DM_PAPERLENGTH = 0x00000004L;
                const long DM_PAPERWIDTH = 0x00000008L;
                pDevMode.dmFields = (int)(DM_PAPERSIZE | DM_PAPERWIDTH | DM_PAPERLENGTH);
                pDevMode.dmPaperSize = 256;
```

```
                pDevMode.dmPaperWidth = (short)(WidthInMm * 1000.0);
                pDevMode.dmPaperLength = (short)(HeightInMm * 1000.0);
                ResetDC(hDC, ref pDevMode);
                DeleteDC(hDC);
            }
        }
    }
    #endregion 指定的打印机设置以mm为单位的自定义纸张(Form)

    #region 获取本地打印机列表
    /**/
    /// <summary>
    /// 获取本地打印机列表
    /// 可以通过制定参数获取网络打印机
    /// </summary>
    /// <returns>打印机列表</returns>
    public static System.Collections.ArrayList GetPrinterList()
    {
        System.Collections.ArrayList alRet = new System.Collections.ArrayList();
        PRINTER_INFO_2[] Info2 = EnumPrintersByFlag(PrinterEnumFlags.PRINTER_ENUM_LOCAL);
        for (int i = 0; i < Info2.Length; i++)
        {
            alRet.Add(Info2[i].pPrinterName);
        }
        return alRet;
    }
    #endregion 获取本地打印机列表

    #region 获取本机的默认打印机名称
    /**/
    /// <summary>
    /// 获取本机的默认打印机名称
    /// </summary>
```

```
        /// <returns>默认打印机名称</returns>
        public static string GetDeaultPrinterName()
        {
            StringBuilder dp = new StringBuilder(256);
            int size = dp.Capacity;
            if (GetDefaultPrinter(dp, ref size))
            {
                return dp.ToString();
            }
            else
            {
                return string.Empty;
            }
        }
        #endregion 获取本机的默认打印机名称

        #region 设置默认打印机
        /**/
        /// <summary>
        /// 设置默认打印机
        /// </summary>
        /// <param name="PrinterName">可用的打印机名称</param>
        public static void SetPrinterToDefault(string PrinterName)
        {
            SetDefaultPrinter(PrinterName);
        }
        #endregion 设置默认打印机

        #region 判断打印机是否在系统可用的打印机列表中
        /**/
        ///// <summary>
        ///// 判断打印机是否在系统可用的打印机列表中
        ///// </summary>
```

```
        ///// <param name="PrinterName">打印机名称</param>
        ///// <returns>是：在；否：不在</returns>
        public static bool PrinterInList(string PrinterName)
        {
            bool bolRet = false;

            System.Collections.ArrayList alPrinters = GetPrinterList();

            for (int i = 0; i < alPrinters.Count; i++)
            {
                if (PrinterName == alPrinters[i].ToString())
                {
                    bolRet = true;
                    break;
                }
            }

            alPrinters.Clear();
            alPrinters = null;

            return bolRet;
        }
        #endregion 判断打印机是否在系统可用的打印机列表中

        #region 判断表单是否在指定的打印机所支持的纸张列表中
        /**/
        ///// <summary>
        ///// 判断表单是否在指定的打印机所支持的纸张列表中,表单就是我们平常所说的纸张
        ///// </summary>
        ///// <param name="PrinterName">打印机名称</param>
        ///// <param name="PaperName">纸张名称</param>
        ///// <returns>是：在；否：不在</returns>
        public static bool FormInPrinter(string PrinterName, string PaperName)
```

```
        {
            bool bolRet = false;

            System.Drawing.Printing.PrintDocument pd = new
System.Drawing.Printing.PrintDocument();

            pd.PrinterSettings.PrinterName = PrinterName;

            foreach (System.Drawing.Printing.PaperSize ps in pd.PrinterSettings.PaperSizes)
            {
                if (ps.PaperName == PaperName)
                {
                    bolRet = true;
                    break;
                }
            }

            pd.Dispose();

            return bolRet;
        }
        #endregion 判断表单是否在指定的打印机所支持的纸张列表中

        #region 判断指定纸张的宽度和高度和与打印内容指定的宽度和高度是否匹配
        /**/
        /// <summary>
        /// 判断指定纸张的宽度和高度和与打印内容指定的宽度和高度是否匹配
        /// </summary>
        /// <param name="PrinterName">打印机名称</param>
        /// <param name="FormName">表单名称</param>
        /// <param name="Width">宽度</param>
        /// <param name="Height">高度</param>
        /// <returns></returns>
```

```
        public static bool FormSameSize(string PrinterName, string FormName, decimal Width,
decimal Height)
        {
            bool bolRet = false;

            System.Drawing.Printing.PrintDocument pd = new
System.Drawing.Printing.PrintDocument();

            pd.PrinterSettings.PrinterName = PrinterName;

            foreach (System.Drawing.Printing.PaperSize ps in pd.PrinterSettings.PaperSizes)
            {
                if (ps.PaperName == FormName)
                {
                    decimal decWidth = FromInchToCM(System.Convert.ToDecimal(ps.Width));
                    decimal decHeight = FromInchToCM(System.Convert.ToDecimal(ps.Height));
                    //只要整数位相同即认为是同一纸张，毕竟inch到cm的转换并不能整除
                    if (Math.Round(decWidth, 0) == Math.Round(Width, 0) && Math.Round(decHeight,
0) == Math.Round(Height, 0))
                        bolRet = true;
                    break;
                }
            }

            pd.Dispose();

            return bolRet;
        }
        #endregion 判断指定纸张的宽度和高度和与打印内容指定的宽度和高度是否匹配

        #region 英寸到厘米的转换
        /**/
        /// <summary>
```

```
        /// 英寸到厘米的转换
        /// /* = = = = = = = = = = = = = = = = = = = *\
        /// | 換算一下计量单位，将其换算成厘米  |
        /// |    厘米      像素      英寸          |
        /// |     1         38     0.395          |
        /// |   0.026        1      0.01          |
        /// |    2.54       96        1           |
        /// \* = = = = = = = = = = = = = = = = = = = */
        /// </summary>
        /// <param name="inch">英寸数</param>
        /// <returns>厘米数，两位小数</returns>
        ///
        public static decimal FromInchToCM(decimal inch)
        {
            return Math.Round((System.Convert.ToDecimal((inch / 100)) *
System.Convert.ToDecimal(2.5400)), 2);
        }
        #endregion 英寸到厘米的转换

        #region 打印操作
        [DllImport("winspool.Drv", EntryPoint = "OpenPrinterA", SetLastError = true, CharSet =
CharSet.Ansi, ExactSpelling = true, CallingConvention = CallingConvention.StdCall)]
        public static extern bool OpenPrinter([MarshalAs(UnmanagedType.LPStr)] string szPrinter,
out IntPtr hPrinter, IntPtr pd);

        [DllImport("winspool.Drv", EntryPoint = "StartDocPrinterA", SetLastError = true, CharSet =
CharSet.Ansi, ExactSpelling = true, CallingConvention = CallingConvention.StdCall)]
        public static extern bool StartDocPrinter(IntPtr hPrinter, Int32 level, [In,
MarshalAs(UnmanagedType.LPStruct)] DOCINFOA di);
        [DllImport("winspool.Drv", EntryPoint = "EndDocPrinter", SetLastError = true,
ExactSpelling = true, CallingConvention = CallingConvention.StdCall)]
        public static extern bool EndDocPrinter(IntPtr hPrinter);
```

```
        [DllImport("winspool.Drv", EntryPoint = "StartPagePrinter", SetLastError = true,
ExactSpelling = true, CallingConvention = CallingConvention.StdCall)]
        public static extern bool StartPagePrinter(IntPtr hPrinter);
        [DllImport("winspool.Drv", EntryPoint = "EndPagePrinter", SetLastError = true,
ExactSpelling = true, CallingConvention = CallingConvention.StdCall)]
        public static extern bool EndPagePrinter(IntPtr hPrinter);
        [DllImport("winspool.Drv", EntryPoint = "WritePrinter", SetLastError = true, ExactSpelling
= true, CallingConvention = CallingConvention.StdCall)]
        public static extern bool WritePrinter(IntPtr hPrinter, IntPtr pBytes, Int32 dwCount, out
Int32 dwWritten);
        /// <summary>
        /// 该方法把非托管内存中的字节数组发送到打印机的打印队列
        /// </summary>
        /// <param name="szPrinterName">打印机名称</param>
        /// <param name="pBytes">非托管内存指针</param>
        /// <param name="dwCount">字节数</param>
        /// <returns>成功返回true，失败时为false</returns>
        public static bool SendBytesToPrinter(string szPrinterName, IntPtr pBytes, Int32 dwCount)
        {
            Int32 dwError = 0, dwWritten = 0;
            IntPtr hPrinter = new IntPtr(0);
            DOCINFOA di = new DOCINFOA();
            bool bSuccess = false;
            di.pDocName = "My C#.NET RAW Document";
            di.pDataType = "RAW";
            try
            {
                // 打开打印机
                if (OpenPrinter(szPrinterName.Normalize(), out hPrinter, IntPtr.Zero))
                {
                    // 启动文档打印
                    if (StartDocPrinter(hPrinter, 1, di))
                    {
```

```
                    // 开始打印
                    if (StartPagePrinter(hPrinter))
                    {
                        // 向打印机输出字节
                        bSuccess = WritePrinter(hPrinter, pBytes, dwCount, out dwWritten);
                        EndPagePrinter(hPrinter);
                    }
                    EndDocPrinter(hPrinter);
                }
                ClosePrinter(hPrinter);
            }
            if (bSuccess == false)
            {
                dwError = Marshal.GetLastWin32Error();
            }
        }
        catch (Win32Exception ex)
        {
            bSuccess = false;
        }
        return bSuccess;
    }
    /// <summary>
    /// 发送文件到打印机方法
    /// </summary>
    /// <param name="szPrinterName">打印机名称</param>
    /// <param name="szFileName">打印文件的路径</param>
    /// <returns></returns>
    public static bool SendFileToPrinter(string szPrinterName, string szFileName)
    {
        bool bSuccess = false;
        try
        {
```

```
            // 打开文件
            FileStream fs = new FileStream(szFileName, FileMode.Open);
            // 将文件内容读作二进制
            BinaryReader br = new BinaryReader(fs);
            // 定义字节数组
            Byte[] bytes = new Byte[fs.Length];
            // 非托管指针
            IntPtr pUnmanagedBytes = new IntPtr(0);
            int nLength;
            nLength = Convert.ToInt32(fs.Length);
            // 读取文件内容到字节数组
            bytes = br.ReadBytes(nLength);
            // 为这些字节分配一些非托管内存
            pUnmanagedBytes = Marshal.AllocCoTaskMem(nLength);
            // 将托管字节数组复制到非托管内存指针
            Marshal.Copy(bytes, 0, pUnmanagedBytes, nLength);
            // 将非托管字节发送到打印机
            bSuccess = SendBytesToPrinter(szPrinterName, pUnmanagedBytes, nLength);
            // 释放先前分配的非托管内存
            Marshal.FreeCoTaskMem(pUnmanagedBytes);
            fs.Close();
            fs.Dispose();
        }
        catch (Win32Exception ex)
        {
            bSuccess = false;
        }
        return bSuccess;
    }
    /// <summary>
    /// 将字符串发送到打印机方法
    /*
        //打印示例：
```

```
            StringBuilder builder = new StringBuilder();
            builder.AppendLine("--------------打印测试--------------");
            string PrintName = PrinterHelper.GetDeaultPrinterName();
            PrinterHelper.SendStringToPrinter(PrintName, builder.ToString());
        */
        /// </summary>
        /// <param name="szPrinterName">打印机名称</param>
        /// <param name="szString">打印的字符串</param>
        /// <returns></returns>
        public static bool SendStringToPrinter(string szPrinterName, string szString)
        {
            bool flag = false;
            try
            {
                // 读取文件内容到字节数组
                byte[] bytes = Encoding.GetEncoding("GB2312").GetBytes(szString.ToString());
                Int32 dwCount = bytes.Length;
                // 非托管指针
                IntPtr pBytes = Marshal.AllocHGlobal(dwCount);
                // 将托管字节数组复制到非托管内存指针
                Marshal.Copy(bytes, 0, pBytes, dwCount);
                // 将非托管字节发送到打印机
                flag = SendBytesToPrinter(szPrinterName, pBytes, dwCount);
                if (flag)
                {
                    StartQianXiang(szPrinterName);  //开钱箱操作
                }
                // 释放先前分配的非托管内存
                Marshal.FreeCoTaskMem(pBytes);
            }
            catch (Win32Exception ex)
            {
                flag = false;
```

```
            }
            return flag;
        }
        /// <summary>
        /// 开始弹出钱箱
        /// </summary>
        public static void StartQianXiang(string szPrinterName)
        {
            //不同的打印机需要不同的参数，这个参数应该可以在打印机的编程文档中找到
            string str = ((char)27).ToString() + ((char)112).ToString() + ((char)0).ToString() +
((char)0).ToString() + ((char)0).ToString();
            byte[] bytes = System.Text.Encoding.Default.GetBytes(str);
            Int32 dwCount = bytes.Length;
            IntPtr pBytes = Marshal.AllocHGlobal(dwCount);
            Marshal.Copy(bytes, 0, pBytes, dwCount);
            SendBytesToPrinter(szPrinterName, pBytes, dwCount);
            Marshal.FreeCoTaskMem(pBytes);
        }
    }

    [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi)]
    public class DOCINFOA
    {
        [MarshalAs(UnmanagedType.LPStr)]
        public string pDocName;
        [MarshalAs(UnmanagedType.LPStr)]
        public string pOutputFile;
        [MarshalAs(UnmanagedType.LPStr)]
        public string pDataType;
    }
    #endregion
}
```

"唯有高屋建瓴，方可水到渠成"

分類: asp.net , C#

microsoftzhcn
粉絲- 100 關注- 0

0 0

+加關注

«上一篇: WPF 實現陰影效果

»下一篇: WPF 控件之Popup

posted @ 2018-09-30 15:55 microsoftzhcn 阅读(1123) 评论(0) 编辑 收藏 举报

刷新评论 刷新页面 返回顶部