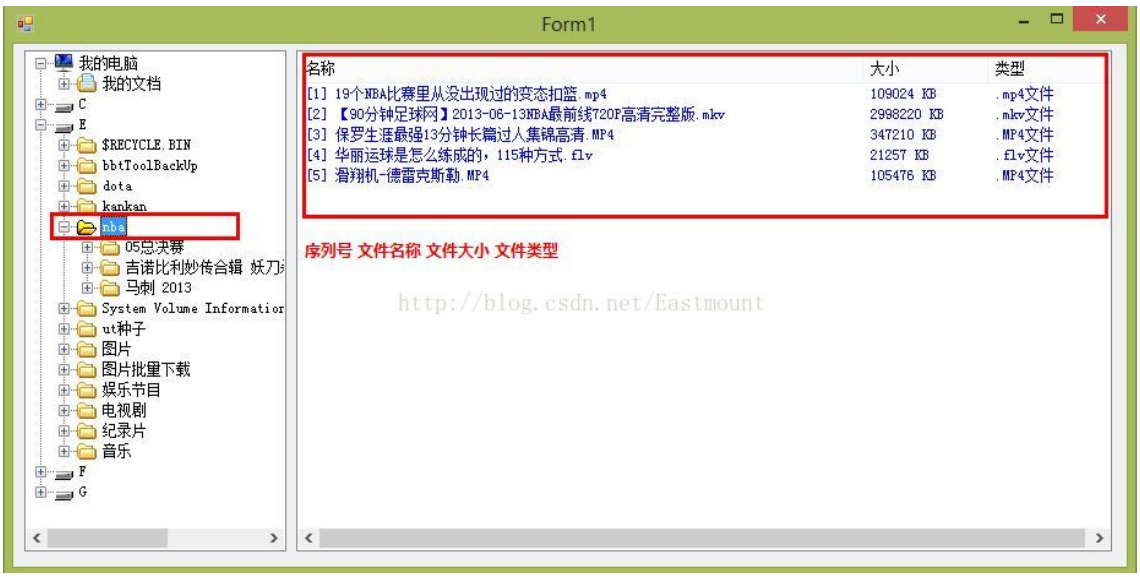
C# 系统应用之ListView控件 (二).加载选中节点文件夹下文件信息

在项目中的前面一篇文章"C# 系统应用之TreeView控件 (一).显示树状磁盘文件目录及加载图标"中我讲述了如何使用TreeView控件树状实现显示"我的电脑"所有磁盘路径下的文件夹,并加载图标如下图所示.

此篇文章我需要讲述的是当鼠标点击左边的TreeView控件中的文件夹时,如何实现在右边的ListView控件中显示相应的文件,并显示文件的名称、大小、类型属性.主要运用的知识是如何实现ListView显示信息,Subitems()函数增加子项,如何设置列表头等信息.运行结果如下图所示:



一.Load函数中加载ListView列标题头

在前篇文章基础上,把控件filesList(ListView控件)拉长,并设置Form1的属性FormBorderStyle(窗体边框和标题栏外观)为FixedSingle,此时该窗体不能拖动拉长.通过Form1_Load()函数初始加载ListView的标题头(名称、大小、类型).代码如下:

```
/// <summary>
/// 窗体加载Load事件 初始化
```

```
/// </summary> private void Form1_Load(object sender, EventArgs e)
{
    ....
    //调用SetListView()函数初始化设置ListView
    SetListView();
}
/// <summary>
/// 自定义函数设置ListView控件初始属性
/// </summary>
private void SetListView()
{
    //行和列是否显示网格线
    this.filesList.GridLines = false;
    //显示方式(注意View是Details详细显示)
    this.filesList.View = View.Details;
    //是否可编辑
    this.filesList.LabelEdit = true;
    //没有足够的空间显示时,是否添加滚动条
    this.filesList.Scrollable = true;
    //对表头进行设置
    this.filesList.HeaderStyle = ColumnHeaderStyle.Clickable;
    //是否可以选择行
    this.filesList.FullRowSelect = true;

    //设置listView列标题头 宽度为9/13 2/13 2/13
    //其中设置标题头自动适应宽度,-1根据内容设置宽度,-2根据标题设置宽度
    this.filesList.Columns.Add("名称", 9 * filesList.Width / 13);
    this.filesList.Columns.Add("大小", 2 * filesList.Width / 13);
    this.filesList.Columns.Add("类型", 2 * filesList.Width / 13);
}
```

二.AfterSelect事件实现ListView加载数据

现在需要在"解决方案"中点击directoryTree(TreeView控件,左边的树状图控件)添加AfterSelect事件,当选中某个节点后,通过该事件显示该文件夹下的所有文件信息.代码如下:

```
#region ListView显示选中文件夹中文件内容
/// <summary>
/// 获取节点的路径:递归调用产生节点对应文件夹的路径
/// </summary>
/// <param name="node"></param>
/// <returns></returns>
private string GetPathFromNode(TreeNode node)
{
    //注意:树形控件中我只赋值Tag\Name,使用Text时赋值即可使用
    if (node.Parent == null)
    {
        return node.Name;
    }
    //Path.Combine组合产生路径 如 Path.Combine("A","B")则生成"A\\B"
    return Path.Combine(GetPathFromNode(node.Parent), node.Name);
}

/// <summary>
/// 更改选定内容后发生 后去当前节点名字
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void directoryTree_AfterSelect(object sender, TreeViewEventArgs e)
```

```
{
    try
    {
        //定义变量
        long length;                        //文件大小
        string path;                        //文件路径
        TreeNode clickedNode = e.Node;      //获取当前选中结点

        //移除ListView所有项
        this.filesList.Items.Clear();

        //获取路径赋值path
        if (clickedNode.Tag.ToString() == "我的文档")
        {
            //获取计算机我的文档文件夹
            path = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
        }
        else
        {
            //通过自定义函数GetPathFromNode获取结点路径
            path = GetPathFromNode(clickedNode);
        }

        //由于"我的电脑"为空结点,无需处理,否则会出现路径获取错误或没有找到"我的电脑"路径
        if (clickedNode.Tag.ToString() != "我的电脑")
        {
            //数据更新 UI暂时挂起直到EndUpdate绘制控件,可以有效避免闪烁并大大提高加载速度
            this.filesList.BeginUpdate();
            //实例目录与子目录
            DirectoryInfo dir = new DirectoryInfo(path);
            //获取当前目录文件列表
            FileInfo[] fileInfo = dir.GetFiles();
            //循环输出获取文件信息
            for (int i = 0; i < fileInfo.Length; i++)
            {
                ListViewItem listItem = new ListViewItem();
                //listItem.SubItems[0].Text = fileInfo[i].Name;             //文件名(方法二)
                listItem.Text = "[" + (i + 1) + "] " + fileInfo[i].Name;    //显示文件名
                listItem.ForeColor = Color.Blue;                            //设置行颜色

                //length/1024转换为KB字节数整数值 Ceiling返回最小整数值 Divide除法
                length = fileInfo[i].Length;                                //获取当前文件大小
                listItem.SubItems.Add(Math.Ceiling(decimal.Divide(length, 1024)) + " KB");

                //获取文件最后访问时间
                //listItem.SubItems.Add(fileInfo[i].LastWriteTime.ToString());

                //获取文件扩展名时可用Substring除去点 否则显示".txt文件"
                listItem.SubItems.Add(fileInfo[i].Extension + "文件");
                //加载数据至filesList
                this.filesList.Items.Add(listItem);
            }
            //结束数据处理,UI界面一次性绘制 否则可能出现闪动情况
            this.filesList.EndUpdate();
        }
    }
    catch (Exception msg)  //异常处理
    {
        MessageBox.Show(msg.Message);
    }
}
#endregion
```

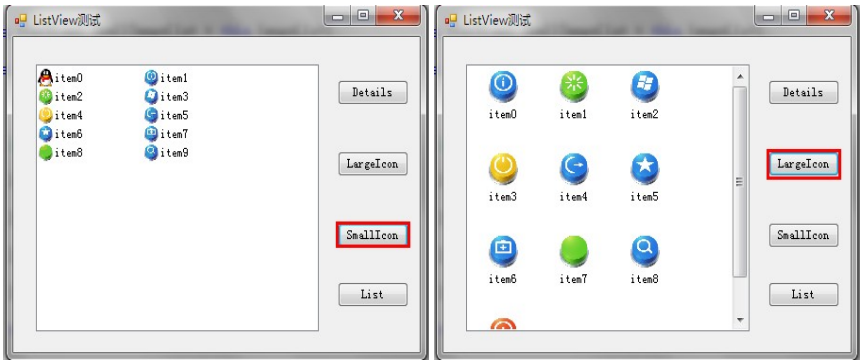
三.遇到的问题

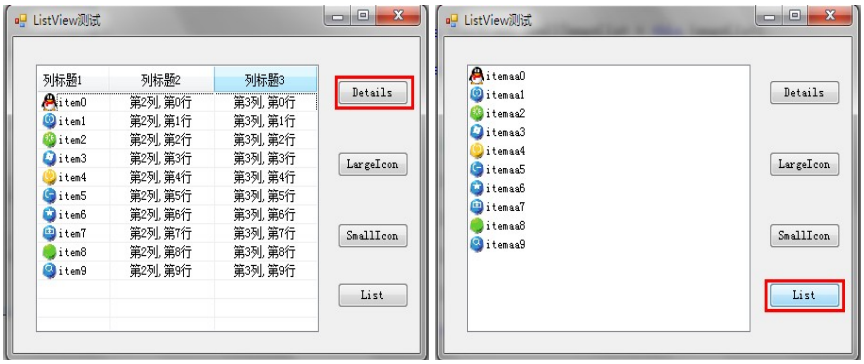
这篇文章看似简单,其实过程遇到很多问题还是值得学习的.

1.遇到的第一个问题就是为什么使用ListView控件总是只显示第一列数据呢？为什么设置了列标题Columns没有显示出来呢?

因为我在程序中初始化后又添加一段代码filesList.View = View.List;设置了View显示的视图为List.而View属性包括五种显示方式:Details(详细信息)、List(列表)、LargeIcon(大图标)、SmallIcon(小图标)、Tile.默认时LargeIcon此时显示为乱序,如果想显示详细信息一行一行的需要设置为Details.

这里推荐大家阅读夜之子的"C# ListView用法详解",真心神作!此处引用他文章中的4张图区别View不同属性显示情况.





2.遇到的第二个问题主要是如何添加数据？如何添加列标题头?

可以采用listItem.SubItems[0].Text添加,但是listItem.SubItems[1]时会报错"InvalidArgument="1"的值对于"Index"无效.参数:index".估计原因是tem没有填入123,所以我采用的方法是listItem.SubItems.Add添加.

3.使用"this.filesList.BeginUpdate()"和"this.filesList.EndUpdate()"数据更新时UI暂时挂起直到EndUpdate绘制控件,可以有效避免闪烁并大大提高加载速度.否则由于加载数据较大时会出现闪烁或空白遮挡等问题.

四.总结

最后希望该文章对大家有所帮助,感谢上面提到的文章及作者.同时如果文章中有错误或不足之处请原谅,有问题或建议者亦可提出.希望尊重作者劳动果实.接下来想实现的是使用API函数添加文件图标和实现右键打开文件和删除文件的操作.仅以此篇文章纪念自己在CSDN发表50篇博客.

文件免费下载地址:http://download.csdn.net/detail/eastmount/7041767

MFC实现该功能类似文章:http://blog.csdn.net/eastmount/article/details/19120567

(By:Eastmount 2014-3-14 下午6点 原创CSDNhttp://blog.csdn.net/eastmount/)