在GridView（列表）中设置一个超链接（LinkButton）。如何获取该超链接的文本内容？

1. 可以通过CommandArgument属性来实现。

在标签内设置属性CommandArgument。

<asp:LinkButton ID=”LinkButton1” runat=”server” CommandArgument=’<%#Eval(“Id”)%>’ onclick=”btnClick”> 签入 </asp:LinkButton>

通过事件参数e来获取该LinkButton的CommandArgument。

Int id = Convert.ToInt32(e.CommandArgument.ToString());

该事件参数e只能是对应的GridViewCommandEventArgs类型。

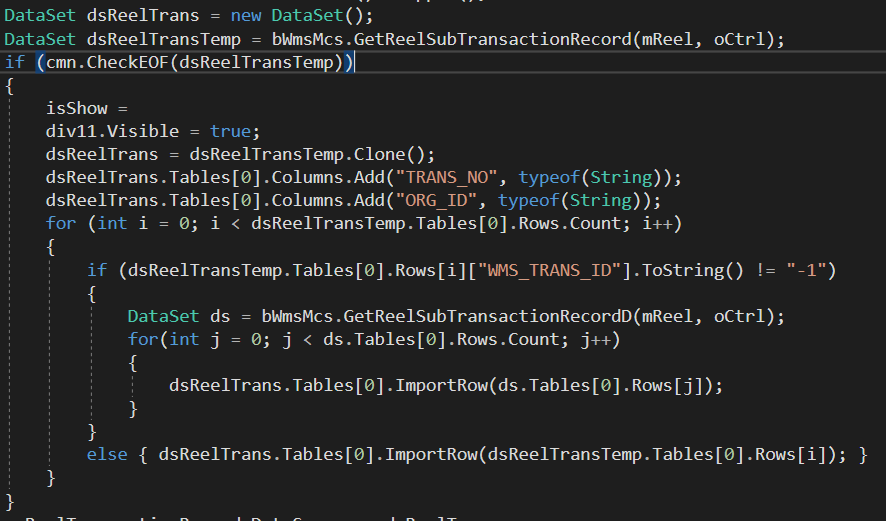
使用NamingContainer获取当前点击的列表（GridView）的行数。



通过事件对象sender来获取对应的GridViewRow，进而得到行数。

对于DataSet的操作积累：（该方法因为不实用，只需要SQL语句中使用（+）即可实现两个表的连接，所以被放弃使用了）

如何根据获取的DataSet数据集来构建自己所需要的DataSet。

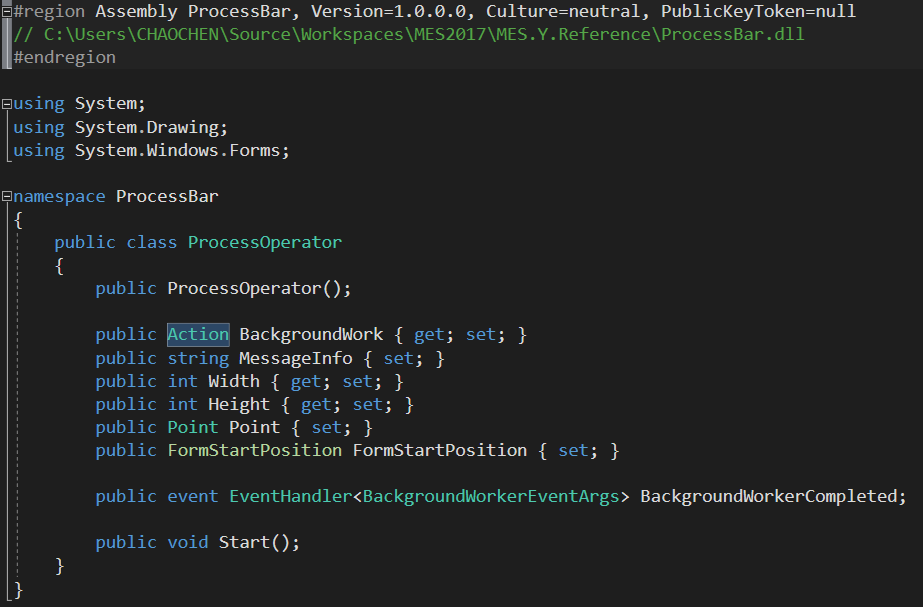


其中dsReelTrans是我们需要构建的数据集。dsReelTransTemp是我们从数据库中获取的未加工数据集。

Dataset.clone()可以根据数据集的格式复制一个没有数据的数据集。

在winform中使用OpenFileDialog來設置選取文件的對話方框。

ProcessBar：是一个进度条控件（该控件使用在PGO比对中）



一般可以指定该processBar的value：即当前进度条的值。

上图的action代表那些没有返回值和函数参数的方法。

未能加载文件或程序集“SrvCommon, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null” 问题解决

一.在写WINFORM程序的时候

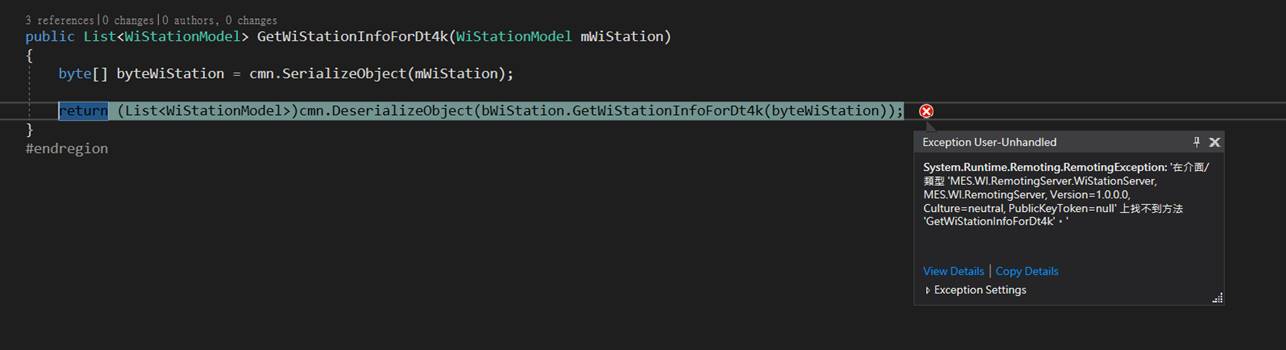
在加载DLL时，出现了如下的异常：未能加载文件或程序集“SrvCommon, Version=1.0.0.0, Culture=neutral,PublicKeyToken=null”或它的某一个依赖项。

试图加载格式不正确的程序。



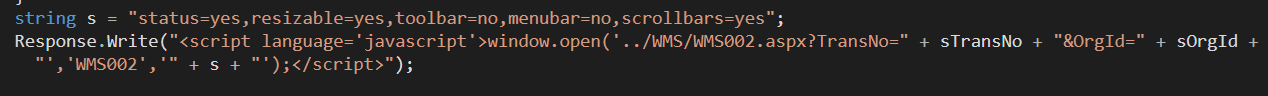
根本原因为:SrvCommon.DLL文件是X86平台生成的。而你的winform程序是Anycpu平台。

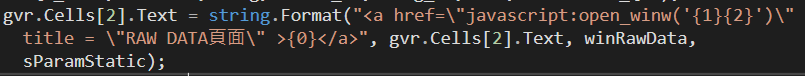
解决方法:SrvCommon.DLL平台改为Anycpu平台。  或者把winform程序改为X86平台使用即可.



如工作中这个问题是在代码中新增了方法但是没有对相应的dll文件进行重新编译，导致该dll文件中没有相应方法从而发生报错。

标签<asp:boundfield>是GridView中用于绑定从数据库中获取的字段的

可以使用winopen打开新的窗口，

还可以使用a标签

可以使用Ajax Script.Alert(“”)來報出錯誤信息。

使用用於訪問數據庫的dao文件中的方法：

通过声明一个IBLL的对象，来访问对应BLL层中的方法，再由BLL层中的方法来访问IDAL层中的数据库访问方法，最后到达DAL层中的dao文件访问对应的数据库方法。

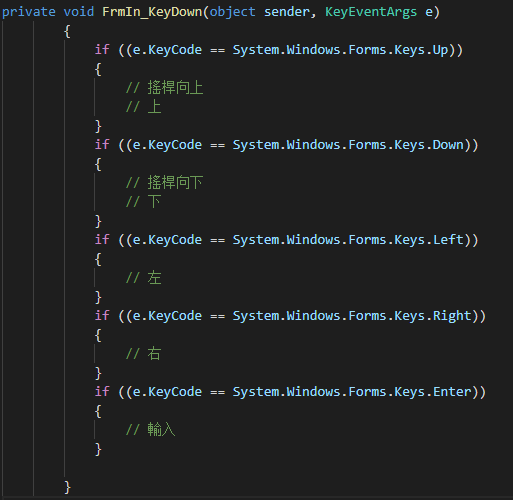
判斷數組中是否存在某一個元素：

Array.IndexOf<string>(arrTempPosition, "sss") != -1

True則存在，反之則不存在。

asp:label中的tooltip属性值为设置鼠标放上去时显示的提示文本。

有关PDA上下左右按钮代码（下部代码为系统自动生成的）



Math.ceiling(decimal var); 取得當前非整型的大於其本身的最小整數

Math.floor(decimal var); 取得當前非整形的小於其本身的最大整數

BLLFactory用於創建一個IBLL接口對象



實現遠程客戶端和服務端的交互。

Window.onbeforeunload = function (e) {

return(e|| window.event).returnValue = ‘確認離開？’;

}

Onbeforeunload事件为关闭当前网页窗口前触发。

返回的是提示的内容。

Onunload事件为退出窗口时触发。

点击浏览器退出键会触发beforeunload和unload；

点击刷新键会也会触发beforeunload和unload，之后会执行onload。（刷新也是一种关闭）

跳转也是一种关闭。

在js中获取服务器控件：

原理是在服务器端把服务器控件的客户端以js的形式输出即可。比如你有一个服务器端控件的id是test1，那你就在aspx中写

<script>

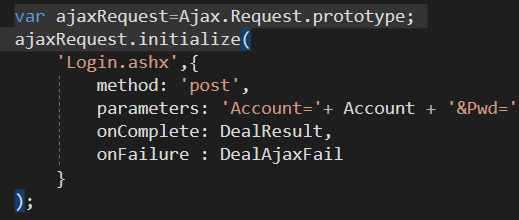
Var myControllerid = <%= test1.ClientId%>

<script>

然后在其他js中用document.getElementById(myControllerid)即可获得该服务器控件。

Cannot read property 'ajax' of undefined

对于这种ajax无法定义的问题，一般是没有导入对应的jQuery包。



使用prototype也需要导入prototype.js包。

Ashx是一个用于接收需要返回内容的请求的一个文件。多用ajax异步发送请求来访问。

一般webform中服务器控件在生成网页后就编译成了HTML中的控件，所以若要请求服务器端的内容直接通过js处理是不方便的，所以通过ashx文件来实现与服务器端的数据交互。

在给web用户设置logtype时使用过。

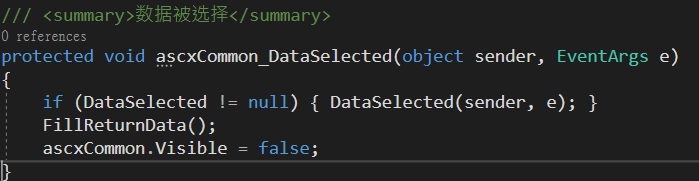
获取当前页面鼠标位置（clientX：获取当前可视窗口的位置。pageX：获取鼠标指针距离HTML的左上角距离，不会随着滚动条的改变而改变。ScreenX获取整个电脑屏幕的位置。）

1.PageX/PageX:鼠标在页面上的位置,从页面左上角开始,即是以页面为参考点,不随滑动条移动而变化  
2.clientX/clientY:鼠标在页面上可视区域的位置,从浏览器可视区域左上角开始,即是以浏览器滑动条此刻的滑动到的位置为参考点,随滑动条移动 而变化.  
可是悲剧的是,PageX只有FF特有,IE这个悲剧没有啊T\_T,所以大牛们想出了一个办法  
PageY=clientY+scrollTop-clientTop;(只讨论Y轴,X轴同理,下同)  
3.screenX/screenY：鼠标在屏幕上的位置,从屏幕左上角开始（[w3c标准](https://www.baidu.com/s?wd=w3c%E6%A0%87%E5%87%86&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)）  
4.offsetX/offsetY:IE特有,鼠标相比较于触发事件的元素的位置,以元素盒子模型的内容区域的左上角为参考点,如果有boder,可能出现负值  
5.  
layerX/layerY：FF特有,鼠标相比较于当前坐标系的位置,即如果触发元素没有设置[绝对定位](https://www.baidu.com/s?wd=%E7%BB%9D%E5%AF%B9%E5%AE%9A%E4%BD%8D&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)或[相对定位](https://www.baidu.com/s?wd=%E7%9B%B8%E5%AF%B9%E5%AE%9A%E4%BD%8D&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao),以页面为参考点,如果有,将改变参考坐标系,从触发元素盒子模型的border区域的左上角为参考点也就是当触发元素设置了相对或者[绝对定位](https://www.baidu.com/s?wd=%E7%BB%9D%E5%AF%B9%E5%AE%9A%E4%BD%8D&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)后,layerX和offsetX就幸福地生活在一起^-^,几乎相等,唯一不同就是一个从border为参考点,一个以内容为参考点  
  
chrome和safari一条龙通杀!完全支持所有属性.其中(offsetX和layerX都是以border为参考点)  
  
下面这个是获取相对于屏幕的坐标  
document.onmousemove=function(e){e=e? e:window.event;document.writeln("X:"+e.screenX+"Y:"+e.screenY);}

CommonBlock作为通用带放大镜的可搜索控件知识学习：

在CommonInfo中的进行selected选择，会触发事件gvCommon\_SelectedIndexChanging



其中DataSelected事件会触发CommonBlock中的ascxCommon\_DataSelected

只需要將ascxCommon\_DataSelected添加到DataSelected事件即可。

获取网址查询字符串中传递的值：

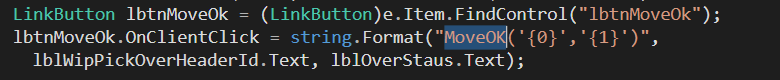




前后端数据传递思路：

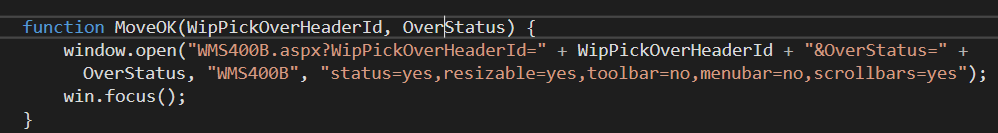
通过服务器端与js前端进行数据传递。

如WMS400中，下图所示：

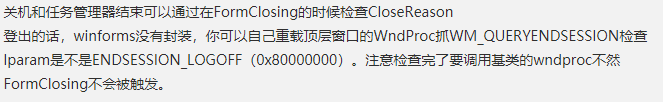


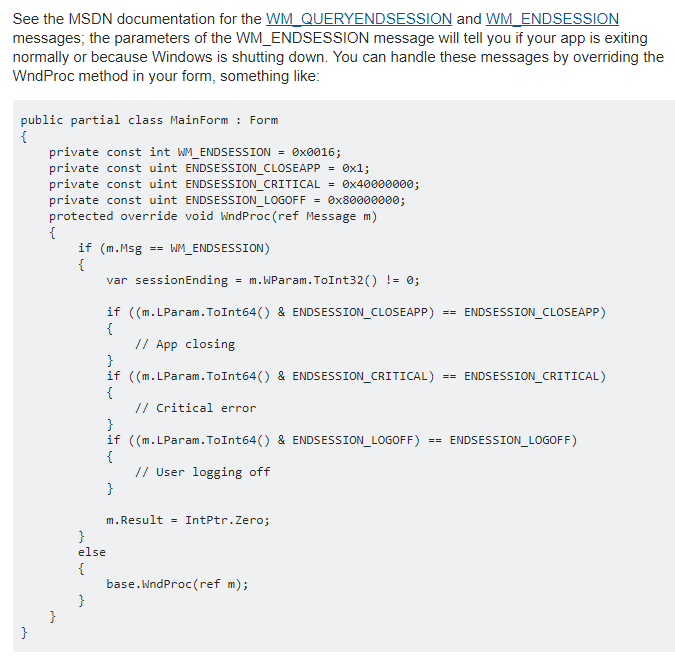
通过服务器端给前端LinkButton添加一个点击事件，并通过参数将后端数据传入前端。

然后在js中编写MoveOK函数使用数据。



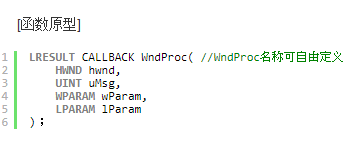
**.NET 窗体程式**捕捉Windows关机、重启，任务管理器结束进程操作所导致本应用程序的中止事件。





什么是WndProc（窗口过程：Windows process）：

WndProc是Windows操作系统向应用程序发送一系列消息，每个窗口会有一个窗口过程的回调函数，其中有4个参数。分别是窗口句柄、消息ID、WPARAM、LPARAM。（但键盘消息发出时，wParam的值为按下按键的虚拟键码，Lparam则存储按键的相关状态信息。因此，如果窗口对传入的键盘消息处理时，只需要判断wParam值即可。当鼠标消息发出时，Wparam值为鼠标按键信息，而Lparam则存储鼠标的坐标。



窗口过程：

左键按下和左键抬起，应用程序将通过GetMessage等方法最终将消息提交到窗口过程。

# System.Diagnostics

using System.Diagnostics命名空间包含了能够与系统进程 事件日志和性能计数器进行交互的类。一般用于帮助诊断和调试应用程序。

System.Diagnostics.Process.Start(); 能做什么呢？它主要有以下几个功能：

1、打开某个链接网址（弹窗）。

2、定位打开某个文件目录。

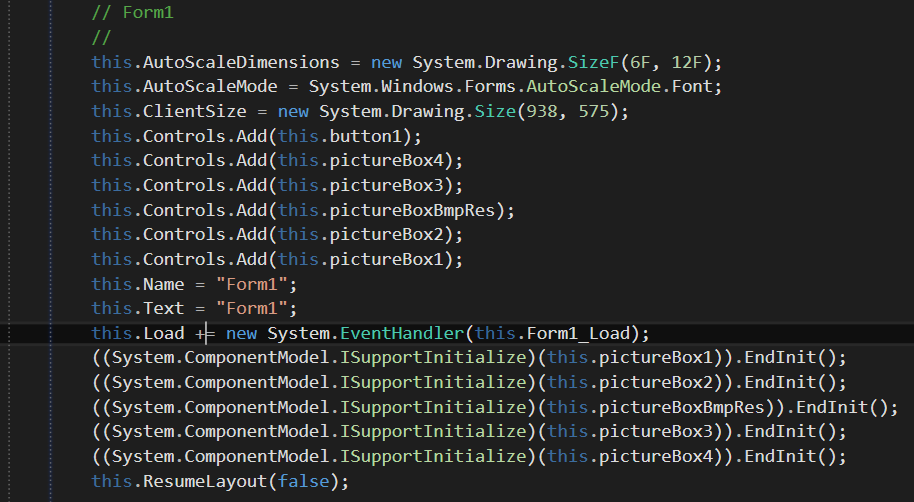
3、打开系统特殊文件夹，如“控制面板”等。

以上功能即创建一个完成功能的进程。

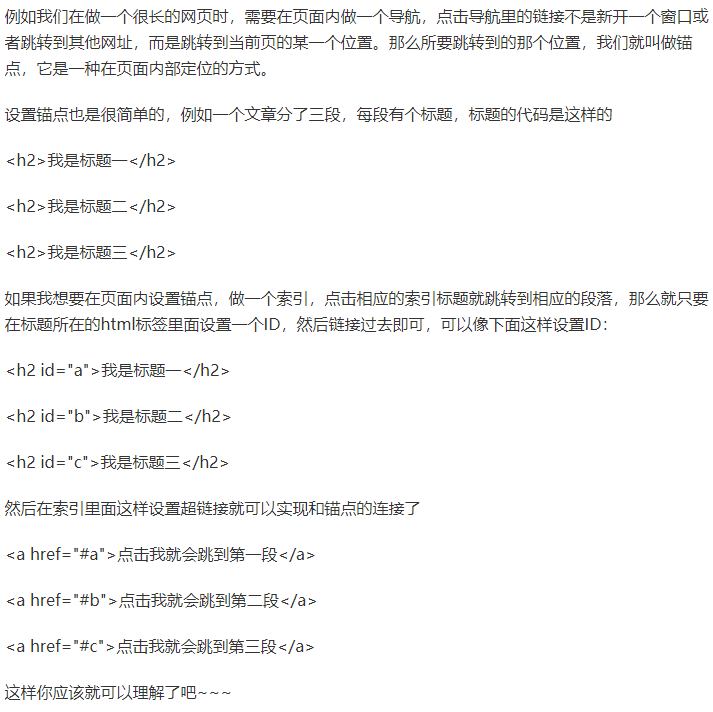
在winform中父窗口可以直接获取子窗口中的public变量和方法。

Listitem功能存在与命名空间System.Web.UI.WebControls中，但是还需要引入System.Web.dll文件才行。

Winform中对于load事件需要在对应form中进行添加



锚点（anchor）：



（复习：lock关键字可以作为Moniter类的一个替代。即lock关键字本质就是Moniter.

Moniter.Enter(this);

…

Moniter.Exit(this);

等效

Lock(this)

{} )

Moniter類中的TryEnter()方法在尝试获取一个对象上的显式锁方面和Enter()方法类似.然而,它不像Enter() 方法那样会阻塞执行.如果线程成功进入关键区域那么TryEnter()方法会返回true.

TryEnter()方法的三个重载方法中的两个以一个timeout类型值作为参数,表示按照指定事件等待锁.

当发生资源争夺而你又不想让线程睡眠一段不可预期的时间时TryEnter()是很有用的.

向ISP拨号的例子很好的解释这个.假设有两个程序A和B,它们都想使用同一个调制解调器向ISP拨号. 而一旦连接建立那么只会有一个网络连接,我们不知道需要已有的应用程序将会连接多长的时间.假设程序A首先向ISP拨号,然后程序B也向ISP拨号;毫无疑问程序B将会一直等待,因为我们不知道程序A将连接多久.在这种情况下,程序B可能使用TryEnter()来确定调制调节器是否已经被另一个应用程序锁定(就是本例中的A),而不是使用Enter()方法导致一直等待.

获取当前运行程序的路径的方法：

System.windows.form.application.startuppath

或者

public string sAppPath = Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().GetName().CodeBase);

C# 中的int?:表示可空类型,就是一种特殊的值类型,它的值可以为null