Winform是.net开发平台中对windows Form的简称，基于对.net frameword的客户端开发技术，一般使用c#编程

Windows应用程序：客户端应用，显示信息，输入，网络与远程计算机进行通信。

Windows应用程序：又名事件驱动程序

窗体上存在控件，通过键盘和鼠标，触发事件完成相应的事件。

事件驱动程序—点击窗体上的控件—触发事件—相应操作

窗体上的控件：输入的位置，点击的位置，选择的位置，显示的位置。

Windows应用程序:功能—控件，--属性和事件。

[STAThread]//特性，COM的线程模型是单线程模型，如果没有它，无法工作。

static void Main()

{

//Application,是System.Windows.Form.Application的命名空间下的一个静态类

//提供静态方法和属性来管理应用程序

//启动应用程序可视样式

Application.EnableVisualStyles();

//将某些控件上的CompatibleTextrRenderingDefault设置为false，默认值

Application.SetCompatibleTextRenderingDefault(false);

//指定应用程序中的某个窗体

//启动的窗体可见，并显示出来

Application.Run(new Form1());

}

Winform程序结构

引用： 添加引用，选择程序所需要的程序集

# 项目文件夹中

Bin：存放项目的编辑结果exe，dll，debug调试，release发布，pdb位置信息-调试

Obj文件：object编译过程中生成的中间临时文件，加快编译速度。

# 3．窗体文件

Form.cs:窗体的源代码文件(处理代码)

Form.Designer.cs:窗体布局代码，自动生成控件代码

Form1.resx:窗体的资源文件(窗体的图标啊)

# 4．Winform控件

输入或操作的对象，属性，方法和事件

属性：外观

方法：功能

事件；行为

控件特征：

控件可视化，与用户交互，属性，方法和事件供开发去使用，可发布和重用

控件开发：自我开发控件。

1. 复合控件，将各种控件组合起来，形成一个控件。
2. 扩展控件，现有控件上派生出新的控件，为原有控件增加新的功能。
3. 自定义控件，Control类派生，提供入门性能。

控件分类

一般控件，容器，菜单和工具，组件，打印

## 4.1一般控件

公共控件；lable标签

TextBox文本框

Button按钮

RadioButton单选按钮(在同一个容器，只能选择单一一个)

CheckBox复选框

ComboBox下拉框

CheckedListBox带复选框的列表项

DateTimePicker日期时间选择控件

ListBox列表框

ListView以五种不同视图显示项的集合

MaskedTextBox格式化文本框(输入规定格式)

MonthCalendar 月历

NumbericUpDown可以增减数值的显示框

PictureBox图片控件

ProgressBar 进度条

RichTextBox高级文本框

TreeView树形控件

容器

GroupBox 分组控件，可以设置标题

Panel 分组控件，可以显示滚动条，

TabControl 选项卡控件

SplitContainer 面板控件

Form：窗体控件

菜单

ContextMenuStrip 右键菜单

MenuStrip 菜单栏

工具

ToolStrip 工具栏

StatusStrip 状态栏

数据

DataGridView 数据显示

DataSet 内存缓存

组件

Timer 计时器

ImageLst 图像集合

Form简介：

可视化的图面，显示信息。通常通过添加控件和开发操作的响应。

继承与Form –System.Window.Forms.Form

From常用：

AcceptButton：将某个按钮按下回车键=单击。

AutoScroll：当指示控件内容大于它的可见区域时是否自动显示滚动条。

AutoSize：指定控件是否自动调节大小满足显示内容。

BackColor：背景颜色

BackGroundImage：背景图片

CancelButton，按下Esc就等于按下，设定好的Button。

ControlBox：为true显示窗体最上面的最大化，最小化菜单栏，false不显示。

Enabled：控件是否使用。

FormBorderStyle：设定窗体的样式(比如运行之后窗体是否可以缩放，还要将最大化禁用)

IsMdiContainer：作为MDI的容器可以显示其他窗体。

StartPosition：窗体第一次出现的位置。

窗体显示

Form.Show();//可以显示多个窗体显示。

Form.ShowDiaLog();//模式化窗体，当显示的窗体不关闭，无法使用之前的窗体。

Form.Activate();//激活窗体，就是当存在两个窗体时，你点击了应该窗体，另外一个窗体就不被聚焦，使用这个方法就可以激活窗体。

Form.Hide();隐藏窗体

事件：

Leave：当窗体不再是活动控件时发生(可以隐藏窗体)。

Activated：当窗体激活的时候发生。

Deactivate：当窗体停用的时候会发生。

**Form窗体创造控件的两种方式**

1. 就拖拽控件
2. 在Form程序中new控件的对象，第一new 一个控件，第二设定控件自动设定大小，设定控件位置，大小，第三添加事件，最后将控件添加到窗体的Controls控件集中。
3. /// <summary>
4. /// 绘制窗体前发生
5. /// </summary>
6. /// <param name="sender"></param>
7. /// <param name="e"></param>
8. private void Form1\_Load(object sender, EventArgs e)
9. {
10. //new 控件，初始化控件
11. Label lbl1=new Label();
12. lbl1.AutoSize= true;//自动显示代销
13. lbl1.Location=new Point(60,80);//设置控件的坐标
14. lbl1.Name="lbl1";
15. lbl1.Size=new Size(100,100);
16. lbl1.Text="1234567";
17. //添加事件
18. lbl1.Click+=Lbl1\_Click;
19. //将控件添加入窗体
20. this.Controls.Add(lbl1);
21. }
22. /// <summary>
23. /// 按下lbl1
24. /// </summary>
25. /// <param name="sender"></param>
26. /// <param name="e"></param>
27. private void Lbl1\_Click(object sender, EventArgs e)
28. {
29. //这里的sender就是事件的触发者
30. Label lbl2 = (Label)sender;
31. MessageBox.Show(lbl2.Text);
32. }

**标签Label，按钮Button，文本框TextBox**

Label：

属性Name：后台代码访问就是访问Name的值

Text：就是控件的文本信息

Image：设置控件的图像

ImageList：图像集控件

Size：控件的宽和高

Tag：与控件相关的自定义数据

Rnabled：控件是否可用

Location:控件位置 坐标

TextBox：多行，单行，

属性

WordWarp：是否可以自动换行

Multiline：控制编辑控件的文本是否可以跨越多行

PassWordChar：让你输入的文本框只显示为一个字符

方法：AppendText添加文本到文本框的末尾，Clear()清除文本框中的文本，Focus()获取焦点，select()选择文本

SelectAll()获取所有文本。

Button按钮，指令命令

基础与Buttonbase，buttonbase继承与Control

·DialogResult：再点击按钮之后放回一个值，**None（默认值）：** 表示没有设置特定的返回结果。

**·OK：** 表示点击按钮后执行的操作是“确定”。通常用于确认用户的输入或选择。

**·Cancel：** 表示点击按钮后执行的操作是“取消”。通常用于放弃对话框中的操作。

**·Yes：** 表示点击按钮后执行的操作是“是”。通常用于确认某种操作。

**·No：** 表示点击按钮后执行的操作是“否”。通常用于否认某种操作。

**·Abort、Retry、Ignore：** 这些选项表示中断、重试和忽略，可以用于某些特定的交互场景。

**RadioButton单选按钮**,在同一个容器中只能选择一个

属性：当AutoCheck：使单选时更改单击时自动更改状态，为true，一个容器内只能选中一个RadioButton，当为False，选中之后，另外的RadioButton不会改变、

事件 CheckedChanged ，当选中之后发生

**CheckBox复选框**

属性

CheckAlign：复选框的文本放置位置

ThreeState(true，false)：是否允许复选框存在三种状态，

事件

CheckedChanged：选中改变，CheckedStateChange选中状态改变，一般选中ComboBox，两个事件都可以触发，先触发CheckedChanged，后触发CheckedStateChanged。

**ListBox，下拉框**

属性：

DisplayMember：显示对应数据源的实际值

ValueMember：显示对应数据源实际值的属性

Item：获取listbox的现的集合。

MultiColumn：表示ListBox中的项是否水平显示

ScrollAlwysVisble：是否显示滚动条。

SelectionMode：是否可以选择多项。

Datasource：数据源，设定这个之后必须设定ValueMember，

DisplayMember。

事件

SelectedIndexChange：选择项更改时触发

方法(和List集合很多方法相识)：

Add（object item）

Addrange（object[] items）

Contains(object value)，判断是否存在某项

Insert(int index，object item)插入某项

Remove(object value)移动某项

RemoveAt（int nums）移动某项

方法在ListBox加载大量的项

需要在加载前后添加BeginUpdate()，EndUpdate()，表示开始修改，和修改结束。

**ComboBox下拉框**

属性

DropDownStyle：下拉框的样式

MaxDropDownItems：下拉框可见部分最多显示多少项

方法

ListBox一样在items中添加大量数据的时候，

开始BeginUpdate，结束EenUpdate。

当添加的项是对象集合，必须指明DisplayMember和ValueMember的值。

与ListBox一样，指定了DataSource就不能直接修改ListBox的值，而是需要通过对应其得到的对象集合比如

This.combobox1.DataSource=list;

需要修改list集合。相对应的combobox1对应的数据也会改变。

对应小量的数据可以通过，this.combobox1.items.add();一个个添加。

**CheckedListBox**：复选框列表控件(每个项之前有一个选择框)基本上和ListBox差不多。

对于CheckedListBox 勾选和选中不一样

勾选 check 选中 select

属性

CheckOnClick：当为ture时，单击选中，同时勾选，当false单击选中之后，需要再次点击才会勾选。

Itmes：添加项和ComboBox，ListBox是差不多的。

**DateTimePickel：日期**

属性

Format：确定日期和时间是用标准格式还是自定义格式

ShowCheckBox：确定显不显示复选框的，当有了复选框，勾选复选框的时候，才能选中框中时间修改，没有选中无法修改。

MonthCalender：月日期显示

属性MaxSelectCount：控制该控件选择最大天数

FristDayOfWeek：显示日期表最开始一列是星期几

方法selectstartdate：开始时间

Selectenddate：结束时间

事件：DateChange，显示

使用在选择时间范围上使用MonthCalender

**MaskedTextBox：文本框**

--掩码区分输入的正确与不正确

属性AsciiOnly：指示是否接受ASCII字符作为有效输入

BeepOnErroe：指示键入无效字符时控件是否发出提示音

Mask：设置控件允许的输入的字符串

TextMaskFormat:指示从text文本属性返回的字符串时是否包含原义字符和/或提示字符.也就是是否返回掩码中的—这类字符. ExcludePromptAndLiterals:表示只得到用户输入的文本; IncludeLiterals:返回用户输入的文本和掩码中的任意文本; IncludePromptAndLiterals: 返回用户输入的文本和掩码中的任意文本以及提示的实列啊; IncludePrompt:返回用户输入的文本以及提示的实列.

RejectInputOnFirstFailure：为true，则只要输入一个字符与掩码不符，输入文本就会被拒绝，否则文本中每个字符按按单个输入逐个处理

Password:设定输入的文本显示为你指定的字符,通过设定’\0’可以取消设定密码的字符.

事件

MaskChange:当mask文本更改时触发

MaskInputRejected:当输入的文本不符合掩码规则报错时触发.

掩码

0表示 0—9之间的数字

9表示 空格或者数字

# 数字或空格 + -

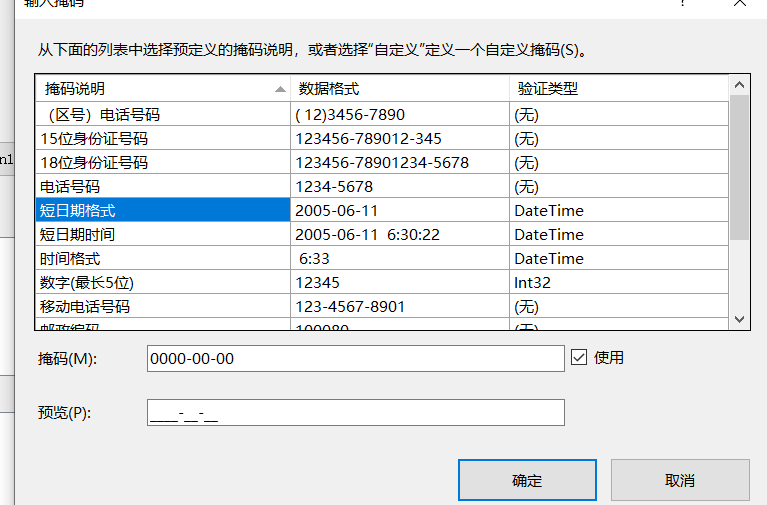
L ASCII字母

C 字符

& 字符

A字母

自定义掩码，



**NumbeicepDown:显示数字文本框**

属性

Hexadecimal:是否用十六进制来显示

Increment:指示每单击一下按钮时增加或减少的数量

IncrementArrowKeys:表示按下箭头控件数字增加或减少设定的Increment的值,默认设定为ture

Maximum;Mianimum:规定控件最大最小的值

DecimalPlaces:指定文本框可以显示的小数位数

方法

UpButton：增加对应好的Increment的值

DownButton：减少对应好的Increment的值

PictureBox：图片显示

属性

ErrorImage：当图片加载失败时显示的图片的路劲

Image：显示图片的路径,程序中 this.PictureBox1.Image=Image.FromFile(“文件路径”);

This.PictureBox1.Image=pictureBox.ImageLocation=“w文件路径”;

Sizemode：表示图片所在控件中展示的方式，其中StretchImage：拉伸(收缩)图片满足控件，Autosize：自动调整大小，CenterImage：显示图片的中间部分到控件，zoom控件自动拉伸收缩来满足图片大小。如果在代码中显示次属性this.pictureBox.ImageSize=PictureBoxSizemode.StretchImage;

ImageLocation：在加载图像的磁盘或web位置，是一个绝对路径。

事件

Click：单击事件

**RichTextBox：高级文本框**

高级的文本输入和功能编辑功能，可以放入图片，文本，超链接

属性

DetectUrls：指示是否自动将URL的格式设置为超链接

Dosk：控件在所在容器中停靠的位置

EnableAutoDragDro：为true用户输入到文本中的字符，图片和其他数据可以在空间中拖动。为false无法拖动啊。

MaxLength：表示文本框最多输入多少字符串

MultiIine:显示控件是否可以多行操作

ScrollBars：定义控件的滚动条

事件：SelectionChanged：当选中的内容更改之后触发

LinkClick：点击文本中的超链接触发

方法：拥有TextBox的方法：RichTextBox.clear();

Find()；查找文本中是否含有指定字符串字符串

加载文件 LoadFile（string path，RichTextBoxStreamType fileType）;

保存文件SaveFile（string path,RichTextStreamType fileType）;

将文本保存到指定路径的文件中去。也包含String类型的一些方法

比如，查找find（string s）,替换rapalce(string s1,string s2)

**RichTextBox使用**：

加载Text文件,暂时只知道编码格式是ASNI的不会出现乱码

OpenFileDialog ofd=new OpenFileDialog();

ofd.Filter="文本文件|\*.txt|PDF|\*.PDF";

ofd.InitialDirectory=@"F:\桌面";

ofd.Multiselect=false;//只能选择一个文件

ofd.Title="选择文件";

DialogResult result = ofd.ShowDialog();

if(result==DialogResult.Cancel)

{

return;

}

//暂时我只知道可以显示Text中ASNI的编码格式

this.richTextBox1.LoadFile(ofd.FileName,RichTextBoxStreamType.PlainText);

保存文件，只能将文件保存成ref后缀的文件

FolderBrowserDialog fbd= new FolderBrowserDialog();

fbd.Description="请选择保存路径";

fbd.SelectedPath=@"F:\";

DialogResult ds= fbd.ShowDialog();

if (ds==DialogResult.Cancel) return;

this.richTextBox1.SaveFile(fbd.SelectedPath+@"\保存文件.ref");

点击超链接，必须启动System.Diagnostics.Process.Start();

private void richTextBox1\_LinkClicked(object sender, LinkClickedEventArgs e)

{

System.Diagnostics.Process.Start(e.LinkText);

}

插入图片

在Winform中有Clipboard的类，相当于应用程序中的剪切板，可以将复制的文本存在其中，

在C# WinForms中，**Clipboard** 是一个类，用于在应用程序中实现剪贴板操作。剪贴板是一个用于在不同应用程序之间传递数据的临时存储区域。**Clipboard** 类允许你在应用程序中操作剪贴板中的数据，例如复制、剪切和粘贴文本或图像。

使用 **Clipboard** 类，你可以执行以下操作：

1. **复制（Copy）：** 将选定的文本或图像复制到剪贴板，以便稍后粘贴到其他地方。这在编辑器、文本框和图像视图等场景中很常见。
2. **剪切（Cut）：** 将选定的文本或图像剪切到剪贴板，类似于复制操作，但同时会从原始位置删除。
3. **粘贴（Paste）：** 从剪贴板中获取数据，并将其粘贴到当前焦点位置，例如文本框或图像视图。
4. **设置和获取其他数据格式：** 除了文本和图像之外，你还可以在剪贴板中设置和获取其他自定义数据格式，以满足特定需求。

//将应用程序中的ClipBoard剪切板

Clipboard.Clear();

//将图片路径得到图片粘贴到剪切板中

//Bitmap bmp = new Bitmap(ofd.FileName);

Image im = Image.FromFile(ofd.FileName);

Clipboard.SetImage(im);

this.richTextBox1.Paste();//直接粘贴剪切板上的图片

Bitmap是位图类

对于文本中的文字加粗，对齐方式

Font oldFont, newFont;

oldFont=this.richTextBox1.Font;

if (oldFont.Bold)//判断得到文本的字体样式是否加粗

{

newFont=new Font(this.richTextBox1.Font, FontStyle.Bold);

}

else//取消加粗文本

{

newFont=new Font(this.richTextBox1.Font,FontStyle.Regular);

}

//判断选择的文本的对齐方式

if (this.richTextBox1.SelectionAlignment==HorizontalAlignment.Center)

{

this.richTextBox1.SelectionAlignment = HorizontalAlignment.Left;

}

else if(this.richTextBox1.SelectionAlignment == HorizontalAlignment.Left)

{

this.richTextBox1.SelectionAlignment=HorizontalAlignment.Center;

}

**Timers 循环时间控件**（Winform提供）

每隔你设定的间隔时间就触发一次，自动触发

属性

Interval 时间间隔 单位ms

Enabled：false关闭，ture开启

事件

Tick：每过一个时间间隔就触发一次

Forms.Timer ui线程执行操作，直接修改UI元素，如果遇到耗时的操作可能会影响到UI层

**System.Timers.Timer** 基于服务 计时器，轻量的(Timers)

时间间隔Elapsed事件执行—不是ui的线程执行

耗时操作，不会使UI失去响应，不会影响下一次触发

Timer.Start()开始 Timer.Clear()关闭，Timer.Stop()停止。

是通过Action委托，之后Invoke线程在Elapsed事件中执行。

不可以直接修改ui元素，可以在其中调用Winform中的Acrion类是Delegate是委托的一个类，不带返回值和参数。之后在调用Winform中的invoke方法，是在多线程的情况下访问ui层控件信息的方法。

不能修改在winform程序ui中的控件属性

在Winform中UI是单线程

在程序中添加Timers.Timer不需要将其添加到容器control中

private void FrmTimers\_Load(object sender, EventArgs e)

{

//创建控件对象

System.Timers.Timer timer = new System.Timers.Timer();

//timer.AutoReset = false;//只执行一次，就会停止触发对应的Elapsed事件

//设定控件属性

timer.Interval = 1000;

//添加事件

timer.Elapsed+=Timer\_Elapsed;

//执行控件，这个控件不需要添加到容器中去

timer.Start();

}

private void Timer\_Elapsed(object sender, System.Timers.ElapsedEventArgs e)

{

Action action = UpTextBox2;

this.Invoke(action);

}

public void UpTextBox2()

{

this.textBox2.Text=DateTime.Now.ToLongDateString();

}

**System.Threading.Timer** 基于多线程 计时器 轻量级 精度和System.Timers.Timer

如果单次运行过程长，不会影响下次的触发。 回调方法来执行操作.

不可以直接修改UI层控件信息 ，也是通过Action委托方法，之后Invoke调用UI层现线程调用

提供指定的事件间隔来触发，这个timer是对线程池线程执行方法的机制，无法继承此类，而且不包含没有参数的构造函数。

public Timer(TimerCallback callback, object state, int dueTime, int period)

一般对此用这个构造函数

TimerCallback 一个System.Threading.TimerCallback委托，表示要执行的方法

State：一个包含回调方法中，要使用信息的对象或null，可以是回调后会返回的对象

Duetime：延时设定延迟时间量之后调用callback方法，以毫秒为单位。指定System.Threading.Timeout.Infinite以防止启动计时器。指定零(0)可立即启动计时器，表示定时器推迟多久执行。

Period：调用之间callback方法的时间间隔 callback，以毫秒为单位，指定System.Threading.Timeout.Infinite可以禁止定期终止

异常：System.ArgmentOutOfRangeException:

dueTime或period 参数为负，且不等于System.Threading.Timeout.Infinite;

T:System.ArgmentOutNullException:

Callback参数为null；

Timer.Change（int dueTime，itn period）；

//修改推迟时间和时间间隔

Period 时间间隔 0或-1 只会执行一次，然后就会停止

Change方法会让计时器重新启动。

停止period 设置为0或-1，Dispose()方法

System.Threading.Timer timer3=new System.Threading.Timer(new System.Threading.TimerCallback(objDelegate => { nums+=2;

Action<int> ac = ShowThreadTime;

this.Invoke(ac,nums);

}),null,1000,1000);

也是需要通过委托和线程来改变ui层

一般用于winform应用就用于Form.Timer

用于服务就是System.Timers.Timer

用于多线程就是System.Threading.Timer

**ProgressBar进度条控件**

属性

Enabled 表示是否启动此控件

ForeColor 此控件的前景色，也就是此文本进度条的颜色颜色

MarqueeAnimationSpeed：字幕动画的速度以毫秒为单位

Maximun：进度条的上限值

Minimum：进度条的下限值

Step：进度条增量

Style：进度条样式

Value：进度条的当前值，不能超过最大和最小值

方法 public void PerformStep()；每调用一次当前进度条就会增加Step设定的值

使用

在Timer配合下，通过PerformSetp()方法，来达到进度条的显示。一般需要选择好进度条显示，最大最小值，并且确定好Step

**ImageList ：图片集合** 用于存储图像的资源，当作某些控件的ImageList属性的对象

属性

Imags：存储图片的集合 加载到控件属性Images之后，所有图片尺寸大小一样

SizeImage：设定图片的大小

方法

Public void Add(string key,Image image);

Public void Add(Image value);

Public void Add(Icon value);

Public void Add(string key,Icon icon);

也可以使用AddRange(Image[] image);添加Image集合

可以通过contains（Image image），containskey(string key)分别是判断ImageList控件的Images属性中是否包含某个图片。

Indexof(Image/string)；返回值是int

Remove(Image);来移除指定的图片

RemoveAt(int);通过ImageList控件Images集合中的引脚移除图片

RemoveBeKey（string）:

判断输入的路劲是否存在

Directory.Exists（string path）;

返回路径下所有的文件名称

Directory.GetFile（string path）;

使用：设定ImageList控件中Images的数据

List<string> key=new List<string>();

string imageFile = @"F:\wallhaven壁纸下载地";//确定存在图片的文件夹路径

if (Directory.Exists(imageFile))//判断磁盘下是否存在对应的文件夹

{

string[] indexImage = { ".png",".bmp","ico"};

string[] fileName=Directory.GetFiles(imageFile);//得到文件夹下的所有文件名

if (fileName.Length>0)//判断文件夹下存在文件

{

foreach (string file in fileName) //得到fileNam数组的下的每一个文件名

{

if (indexImage.Contains(Path.GetExtension(file)))//判断文件名的后缀是否满足要求

{

Image im=Image.FromFile(file);

key.Add(Path.GetFileNameWithoutExtension(file));

this.imageList1.Images.Add(Path.GetFileNameWithoutExtension(file),im);

}

}

}

this.imageList1.ImageSize = new Size(200,200);

//一般无法自己显示，一般用于控件的ImageList属性，可以更具ImageList控件对应的ImageIndex和ImageKey得到ImageList控件对应的图片

}

**Listview：提供五种不同视图，是项集合控件**

属性

CheckBoxes:项的前面是否存在复选框

Items：项的集合

View：可以选择项的五种视图中的一种

Columns:只有view选择详细视图的时候details，才具有效果，可以显示多列

Alignment：指示ListView中各项的对齐方式

AllowColumnReorder：指示用户是否可以重新排列“详细信息”视图中各列的顺序

AutoA rrage：项是否自动排列，在view中Largelcon或SmallIcon视图中有效

FullRowSelect：指示当项被选中时，其所有子项是否同该项一起突出显示

Groups：ListView中的组

HeaderStyle：在详细视图中列标头的样式

Items：项的集合

LabelEdit：允许用户就地编辑项的标签

LabelWrap：标签文本是否可以换行

LargeImageList：ListView用于大图标视图中图像的ImageList控件

MultiSelect：允许选择多项

Scrollabel：当控件包含的项过多，超出工作区的范围，该控件是否显示滚动条

SmallImageList：在ListView控件中处大图标之外的所有视图中图像的ImageList控件

StartImageList：自定义状态下所使用的ImageList控件

勾选和选中

Checkindices勾选集合 CheckItems勾选项集合 Selectindices选中集合 SelectItems 选中项集合

**详细视图**：有效的属性，用的是SmallImage，右边文本，只要在详细视图下才显示colunmns列

Columns，AllowColumnReorder，HeaderStyl

**LargeIcon(大图标显示)：有效的属性 最大化图标，就是按你设定最大的图片Imagesize显示，文本显示在下方**

AutoA rrage，LargeImageLis

**SmallIcon（小图标）：有效的属性 小图标 右边显示文本**

AutoA rrage

List视图：显示小图标 右边显示文本标签，各项排列在列中，没有列标头

Tile视图：最大化视图，文本显示在右边，如果有其他子项也是显示在右边。

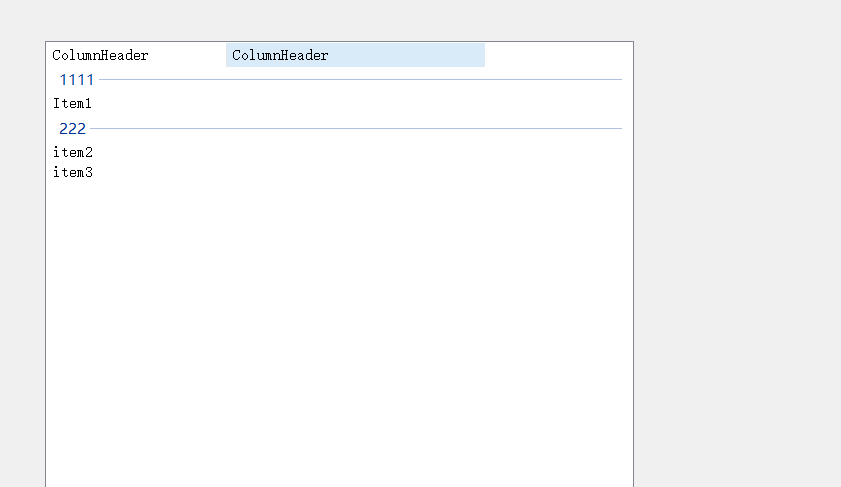
方法：

添加大量项的时候和之前的ComboBox，一样通过

BeginUpdate，EndUpdate两种方法来添加

Clear清空所有项

FindItemWithText（ string text）查找指定项的文本开头，返回值是ListViewItem



**项**

**组**

**列**

添加，访问，存在，清空，索引访问

分组显示：（List不支持分组显示）

项添加时，是没有考虑分组的，取名确包含分组

方案：判断名称，编号，将这项分到这一组

建立分组的关系，组名—编号

关系存储起来，分组时，就以关系分组

**Menustrip：菜单空间---创建菜单项的容器，打开一个页面，或者执行一个事件**

**属性**

ContextMenusStrip：这个一般用来添加ContextMenusStrip空间表示是否添加右键菜单

Dock：表示菜单栏依靠在Form空间那个边框

Enabled:是否启动空间

Layoutstyle：指定空间项的布局方向

Item：菜单项

菜单项ToolStripMenuItem，其子属性ShortcutKeys：设置快捷键，showShortcutKeys：是否显示快捷键，设置Alt+某键之后，需要在文本后面添加(&某键)；ctrl+某键

DropDownItems子菜单项：和菜单项是一样的也可也设置快捷键，这个不用在文本Text后面加什么

菜单项的图标—关联图片集合控件ImageList

菜单项响应---注册菜单的Click事件

手动添加，每个菜单项，都需要给它注册响应事件

菜单项有子集菜单，就不用给它注册响应事件

代码添加菜单项及响应

首先创建对应的菜单和子菜单

其都属于ToolStripMenuItem类，首先创建菜单项，之后创建对应的子菜单项，菜单项添加子菜单项，之后Menustrip控件添加菜单项，对于子菜单的响应，直接添加对应的Click事件，当我们需要设置Ctrl添加快捷键时，可以设置应该Menustrip控件，在其子菜单中设置一个Ctrl快捷键，将其代码复制过来

//菜单项

ToolStripMenuItem tsi=new ToolStripMenuItem();

tsi.Text="学生管理(&M)";//快捷键设置，直接在文本后面添加对应的要求就可以

tsi.Name="MiStudent";

//子菜单

ToolStripMenuItem ts=new ToolStripMenuItem();

ts.Text="学生窗体";

ts.Name="CreatStudentForm";

//添加子菜单的响应事件,就是添加事件

ts.Click +=Ts\_Click; ;

//主菜单添加子菜单之后，将主菜单添加到菜单项中

tsi.DropDownItems.Add(ts);

this.menuStrip1.Items.Add(tsi);

如果当需要鼠标放置在Menustrip第一层菜单栏的情况下就需要显示子菜单，就设定MouseHover事件，

判断第一层菜单栏中的菜单项是否有子菜单，并且判断是否为不显示，如果成立就将其显示出来。

/// <summary>

/// 当我鼠标在菜单栏第一项停留时，将自动显示子菜单项

/// </summary>

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

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

private void MeStudnet\_MouseHover(object sender, EventArgs e)

{

//将触发这个事件的的对象sender 转化为ToolStripMenuItems对象

ToolStripMenuItem item = sender as ToolStripMenuItem;

//判断是否存在子菜单，并且显示子菜单

if (item.HasDropDownItems && !item.DropDown.Visible)

{

item.ShowDropDown();//显示子菜单

}

}

**ContextMenusStrip：右键菜单，也可以和ImageList控件一样，当作某些控件的ContextMenustrip属性的对象**

只要是菜单控件，其Items的菜单栏都是ToolStripMenusItem对象。

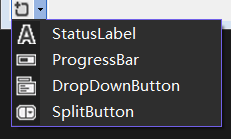
**ToolStrip：工具栏**，可以在其中添加Button按钮，Label控件，Stplitbutton下拉按钮控件，DownDrowButton子菜单，separator正线，combobox下拉框控件，textbox文本控件，ProgressBar进度条

属性

基本上就和其所拥有的控件一样的功能，没有太多的区别

**StatusStrip：状态栏，显示账户信息，角色信息，操作位**置信息，进度条，版本信息

和工具栏的效果基本上差不多



可以添加StatusLabel，ProgressBar进度条，DropDownButton，菜单栏按钮，SplitButton：下拉按钮

**GroupBox，Panel**：分组容器控件

可以呈放其他控件的容器

GroupBox：为其他控件提供可识别的分组，可以设置标题

有标题，有边框，没有滚动条，按功能分组

Panel：将其他控件集中显示在一个面板中，集中管理，没有标题，默认没有标题，有滚动条（必须属性AutoScroll为ture之后，才会显示滚动条）

共同点：分组，容器，拖动Panel GrounpBox就会使它里面的控件一起移动

**TabControl：选项卡集合**，一个模块有多个子页面，tabcontrol控件进行页面间的快速切换。

属性：管理一个TabPages集合

Alignment：确定选项卡是否显示在控件的顶部，底部，左侧或右侧

Appearance：指示选项卡是绘制成按钮还是绘制成常规选项卡

DarwMode：指示由用户还是系统来绘制标题

Multilime：选项卡是否可以多行显示

Tabpage：选项卡的项的集合，可以选择ImageIndex和Imagekey：可以选择对应ImageList的索引和文本

this.tabControl1.SelectTab(1);//指定选项卡，选择哪一项

string s = this.tabControl1.SelectedTab.Text;//获取选择的选项卡的text和name；

this.tabControl1.SelectedIndex = 1;//指定索引激活选定选项卡

**splitContainer**：

介绍：将页面拆分成两个大小可调整的区域，中间有一个拆分条，拖动拆分条来调整区域的大小。

属性：

BordStyle：设置控件的边框：用来区分两个页面的界限。

Dock：绑定容器的边框

FixedPanel：某个SplitContainer面板的大小保持不变，也就是当控件的大小发生改变时，某个面板的宽度保持不变。

IsSplitterFixed：差分条是否固定，默认为False，不固定。

Orientation:决定拆分器是水平的还是垂直的

Panel1

Panel2，都是表示两个拆分的对象

Panel1(2)Collapsed：确定panel1或者panel2是否折叠起来。

Panel1(2)Minsize：确定panel1(2)容器距离左边或者右边最小的像素宽度。

SplitterDistance：拆分条与左边或上边的像素距离

SplitterWidth：差分条的粗细

SplitterIncrement：拆分条移动时的增量

拆分器是可以嵌套的。

**DataGridView**：网格形式显示数据，行和列由用户自定义

可以在拉取此控件的时候，就和本地电脑的数据库进行连接显示表格，对此会自动在App.Config中生成连接的ConnectionStrings

<connectionStrings>

<add name="strsql" connectionString="server=.;uid=jgw;password=123456;database=学生管理系统" />

<add name="WindowsFormsApp1.Properties.Settings.学生学习系统管理DBConnectionString"

connectionString="Data Source=.;Initial Catalog=学生学习系统管理DB;User ID=jgw"

providerName="System.Data.SqlClient" />

</connectionStrings>

属性：

Columns：列的集合，可以添加列的类型，可以选择为只读/可见/冻结

AllowUserToDeleteRows：指示是否允许用户通过Delete按键删除行。

AllowUserToOrederColumns：指示用户是否可以手动调整列的位置。

AllowUserToResizeColumns:指示用户是否可以调整列的大小。

Anchor：定义控件距离容器四边的距离不变，随着容器的变化而发生改变。

AutoSizeColumnsMode:自动调整列的大小模式

AutoSizeRowsMode:自动调整行的大小模式

对于我们设置的列，对应的数据DataPropertyName，绑定的数据库对应的列名。

DataSource:数据源，一般来源于List或者DataTabel

MultiSelect：是否可以选择多行数据

逻辑删除：标识列，修改值，0表示已删除，1表示未删除

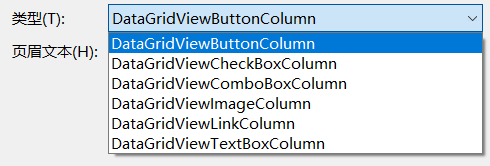
行：类名：DataGridViewRow,集合：DataGridViewRowCollection，属性：Rows

列：类名：DataGridViewColumn,集合DataGridViewColumnCollection,

属性：Columns

单元格：类名：DataGridViewCell，value获得单元格的值；Select：是否选择 RowIndex，ColumnIndex：对应行号和列号，FromattedValue：未格式化的单元值。

DataGridView，添加列时可以是



按钮，复选框，下拉框，图片，超链接，文本框

事件：CellClick：单击单元格任意部分发生

CellContentClick：单击单元格的内容部分发生

CellValueChange：单元格里面的值发生改变，失去焦点时触发 ，当前这个单元格的触发参数DataGridViewCellEventArgs e，可以通过这个e得到当前选择的DataGrideView的单元格的行和列。

CurrentCellDirtyStateChanged：当单元格的内容发生更改时触发

方法:IsCurrentCellDirty:当前单元格是否有未提交的更改

DataGridView绑定数据源

属性，数据源来源Datasource：DataTable，绑定的是数据行,则DataGridView行填充的是—DataRowView.Row—DataRow，DataGridView.Row[0].DataBoundItem—得到行对象的数据类型—为Object—转化为对应的数据类型

绑定的List，添加的就是实体对象；

通过List给定DataGridView.Datasource

1. DataTable—转化为List,行—对象，多行—List集合 ，数据量大，使用 foreach(dataRow item in datatable.rows){},将其中的每一行数据都变为List.add()的项
2. Sqldatareader –读取过程，一条数据读取—实列对象—List集合 数据量不大 while（sqldatareader.read()）{}

对于DataGridView，中存在DataGridViewComboBox下拉框的使用

下拉款—多个选择—初始化—从数据库里加载部门信息表，数据绑定到ComboBox中

和给一个ComboBox控件绑定数据是一样的操作。

CheckBox列的使用：--批量操作，选择数据

DataGridView是新增，修改，删除数据，数据源是DataTable，利用SQLDataAdapter中的Update()方法来提交到数据库

原理 DataGridView中的数据是DataRow一行数据

DataRow 属性—RowState：Added（新加），Modified（已修改），Deleted（删除），UnChanged（显示控件added，Modified，Deleted已提交过后显示）三种状态

将DataTable里所做的Insert，Update，Delete提交到数据库—一致

实例操作—所做的修改是否更新到数据库？？？--SqlcommandBuilder对象—可以将DataGridView所做的修改(DataTable)提交到数据库—小小bug？时间设置了默认值，但是没有进去。

利用SqlDataAdapter中Update（）提交到数据库，--提交的过程;配置命名的过程。

TreeView：树形控件：分层显示标签的项的集合

属性：

Nodes：根节点集合

LabelEdit：指示节点是否可以编辑

PathSeparator：由于节点FullPath返回路径字符串分隔符

ShowLines：父子节点之间的连线

ShowRootLines：根节点之间的连线

TopNode：第一个完全可见的树节点

SelectNode：选定的节点

方法：添加大量节点：BeginUpdate()禁止重绘，EndUpdate()重新启用；CollapseAll：折叠所有节点

ExpandAll：展开所有节点；

GetNodeAt（int 下，int y）：通过坐标获取节点

节点介绍 TreeNode Name –唯一标识节点 Text显示文本Nodes 子节点集合

代码添加节点

//如果一个节点下面有子节点，是应该先把这个节点添加到

//TreeView中，还是先添加好子节点，再添加到TreeView？

//都可以

//添加节点过程：创建一个节点，将这个节点添加到TreeView的Nodes集合中

//创捷节点，将这个节点添加到当前节点的Nodes集合中

//更多的应用不是这样，而采用递归的方式，加载节点及子节点

//数据来源：制作成一张表，ParentId--父节点编号

应用：权限分配，菜单

动态加载菜单表（递归）

表数据：存储的是节点与字节关系数据

1. 获取表数据
2. 创建一个方法：创建节点并且添加到TreeView 递归的原理，自己调用自己(子节点可以有多个)
3. 调用方法，创建节点

动态加载节点:节点及子节点创建好，最后将节点添加到TreeView中

分类：组合控件：在原有控件基础上，根据需要进行组合而形成一个新的控件 UserConntrol；

扩展控件：在现有基础上，对它进行扩展：继承原有控件，添加或扩展原有控件的特征

自定义控件：派生于Control基类，绘制位置全部由用户定义

组合控件：（定义好组合控件需要重新生成解决方案）

可视化外观

添加新的属性 应用窗体上，修改属性值，就可以即使看到变化

定义自定义的事件

自定义控件：Load事件；应用窗体：Load事件

--加载顺序：1应用窗体 Load事件—2用户自定义Load事件

事件调用：

* 事件是一种特殊的委托，通常在类内部声明，并使用**event**关键字标记。
* 事件通常用于在类内部触发，外部类可以订阅事件以响应事件的发生。
* 事件通常通过调用委托变量来触发，但只能在包含事件的类内部使用 **event** 关键字来引发事件。

扩展控件：继承原有控件提供的类

一般来添加控件的属性

对于事件，我们都是重写事件，也就是重载