# 无边框窗体移动

#region 窗体拖动、关闭

private Point mouseOff;//鼠标移动位置变量

private bool leftFlag;//标签是否为左键

private void FrmMain\_MouseDown(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

mouseOff = new Point(-e.X, -e.Y); //得到变量的值

leftFlag = true; //点击左键按下时标注为true;

}

}

private void FrmMain\_MouseMove(object sender, MouseEventArgs e)

{

if (leftFlag)

{

Point mouseSet = Control.MousePosition;

mouseSet.Offset(mouseOff.X, mouseOff.Y); //设置移动后的位置

Location = mouseSet;

}

}

private void FrmMain\_MouseUp(object sender, MouseEventArgs e)

{

if (leftFlag)

{

leftFlag = false;//释放鼠标后标注为false;

}

}

private void btnExitSys\_Click(object sender, EventArgs e)

{

this.Close();

}

#endregion

# OOP原则

对象职责明确原则总结

（1）原则：分离“界面代码”和“数据访问代码”。

（2）好处：不管界面是什么类型的应用程序（Winform、WebForm）当界面发生变化的时候，数据访问部分一般不需要任何变化；同时前台设计人员和后台编写人员可以很好的分离。

（3）注意：当写程序的时候，界面中不能出现任何SQL语句，数据访问后台代码中也不应该有其他业务逻辑代码。

（4）扩展：对象职责明确原则对后续“三层架构”学习打下基础。

readonly与const

# 快捷键设置

## Enter键



# 仅允许输入指定字符

e.Handle=！(e.KeyChar>=48&&e.KeyChar<=57)//允许输入数字

e.Handle=！(e.KeyChar>=‘a’&&e.KeyChar<=‘z’|| e.KeyChar>=‘A’&&e.KeyChar<=‘Z’)//允许输入大小字母

//仅允许数字字符以及删除backspace的输入，已测试

e.Handled =!((e.KeyChar >= 48 && e.KeyChar <= 57) || (e.KeyChar >='a'&& e.KeyChar <='z'|| e.KeyChar >='A'&& e.KeyChar <='Z')||e.KeyChar==8);

KeyCode对照表

<http://www.t086.com/article/4315>

# 避免重复打开窗体



# 正则表达式的使用



# SQLHepler

## DataSet



### DataView属性



# DataGridView控件

dgv.CurrentRow------当前选中行

dgv.CurrentRow.Columns[“列名”].Value------当前选中行指定列的值

dgv的属性ContexrMenuStrip绑定一个ContexrMenuStrip控件，实现右键点击