1. 创建DataTable

# private DataTable CreateData()

# {

# DataTable dt = new DataTable();

# dt.Columns.Add(new DataColumn("类型"));

# dt.Columns.Add(new DataColumn("2005-1月", typeof(decimal)));

# dt.Columns.Add(new DataColumn("2005-2月", typeof(decimal)));

# dt.Columns.Add(new DataColumn("2005-3月", typeof(decimal)));

# dt.Columns.Add(new DataColumn("2005-4月", typeof(decimal)));

# dt.Columns.Add(new DataColumn("2005-5月", typeof(decimal)));

# dt.Columns.Add(new DataColumn("2005-6月", typeof(decimal)));

# dt.Rows.Add(new object[] { "员工人数", 437, 437, 414, 397, 387, 378 });

# dt.Rows.Add(new object[] { "人均月薪", 3964, 3961, 3979, 3974, 3967, 3972 });

# dt.Rows.Add(new object[] { "成本TEU", 3104, 1339, 3595.8, 3154.5, 2499.8, 3026 });

# dt.Rows.Add(new object[] { "人均生产率", 7.1, 3.06, 8.69, 7.95, 6.46, 8.01 });

# dt.Rows.Add(new object[] { "占2005年3月人数比例", 1.06, 1.06, 1, 0.96, 0.93, 0.91 });

# return dt;

# }

1. List

# public static List<Color> colorlist = new List<Color> {Color.Red,Color.Green,Color.Yellow,Color.Blue,Color.Orange };

1. 对象赋值

# var books = new List<Book>

# {

# new Book(){Name="C# learning guide",Year=2015}, //两种写法都行

# new Book{Name="C# step by step",Year=2014}

# };

1. 可选参数

# public void Test(int id,string name="ronaldo",int age=20)

# {

# }

# //可选参数必须位于所有必需参数之后，调用如下：

# Test(1);         //同时使用了两个参数的默认值

# Test(1,age:12);     //name使用了默认值,age使用命名参数的方式设置为12

1. 数组初始化并赋值，无需指定数组长度。

# string[] fa = new string[] { "1","2"};

也可以直接作为参数传递：

# var a = myMethod("", new string[] { "1", "2" });

# ccMsg.Text += "共：" + a + "个params参数！" + Environment.NewLine;

作为方法myMethod，它的声明如下：

# int myMethod(string mmMsg,params string[] names){……}

或

# int myMethod(string mmMsg, string[] names){……}

多个string的if判断，可以这么写：

# if (!new string[] { "子模块", "自定义变量", "系统变量" }.Contains(e.Node.Text))

1. DataTable查询和筛选

# mmDT\_人员.DefaultView.RowFilter = mmSearchString;

# mmDT\_人员.DefaultView.Sort = "地区代码 asc,出生年月 desc,姓名 asc";

# dgvDataList.DataSource = mmDT\_人员.DefaultView.ToTable();

1. combox绑定List

# Common.list\_性别.Insert(0, new Item { Key = "", Value = "" });

# cb性别.DataSource = Common.list\_性别;

# cb性别.ValueMember = "Key";

# cb性别.DisplayMember = "Value";

1. list查询和遍历

# List<Item> mmList=Common.list\_地区.Where(item=>item.Key.EndsWith("0000")).ToList<Item>();

# mmList.ForEach(item =>

# {

# ccTree.Nodes.Add(new TreeNode { Name = item.Key, Text = item.Value });

# });

# foreach (TreeNode node in ccTree.Nodes)

# {

# mmList = Common.list\_地区.Where(item => !item.Key.EndsWith("0000") && item.Key.StartsWith(node.Name.Substring(0, 2))).ToList<Item>();

# mmList.ForEach(item =>

# {

# node.Nodes.Add(new TreeNode { Name = item.Key, Text = item.Value });

# });

# }

1. List增加数据

# public static List<Item> list\_地区 = new List<Item> {

# new Item{Key="130000",Value="河北省"},

# new Item{Key="130100",Value="石家庄市"},

# new Item{Key="130200",Value="唐山市"},

# new Item{Key="130300",Value="秦皇岛市"},

# new Item{Key="130400",Value="邯郸市"},

# new Item{Key="130500",Value="邢台市"},

# new Item{Key="320000",Value="江苏省"},

# new Item{Key="320100",Value="南京市"},

# new Item{Key="320200",Value="无锡市"},

# new Item{Key="320300",Value="徐州市"},

# new Item{Key="320400",Value="常州市"},

# new Item{Key="320500",Value="苏州市"}

# };