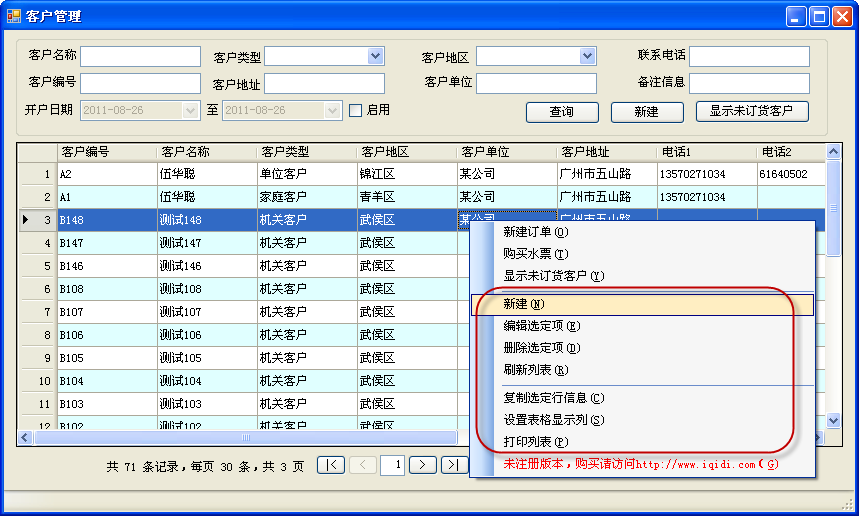
**[Winform分页控件重大更新，并实现普通版、DotNetBar、DevExpress三大版本整合更新(附各种例子源码）](http://www.cnblogs.com/wuhuacong/archive/2011/08/26/2154963.html)**

在Winform开发中，一直离不开分页处理，好的分页控件封装，能为开发节省很多时间和繁琐工作，对分页控件一直的改进和完善，也是我的兴趣之一。本次主要实现对分页控件的全面重构，优化功能及界面的处理，并统一完成普通版、基于DotNetbar界面控件版本、DevExpress界面控件版本三大版本的发布，这几个界面版本，除了界面表现效果不一样外，所有的功能展示及使用代码完全一致，保证用户的知识投资及给客户良好的使用操作。本次主要增加的功能有：可通过界面设置列表显示的列，可复制选定的行信息，可设置显示的行号、可设置显示间隔行的颜色等功能。下面分别对几个例子的截图进行总体介绍，然后在详细介绍

1）普通版（即不引用任何第三方界面控件）效果图

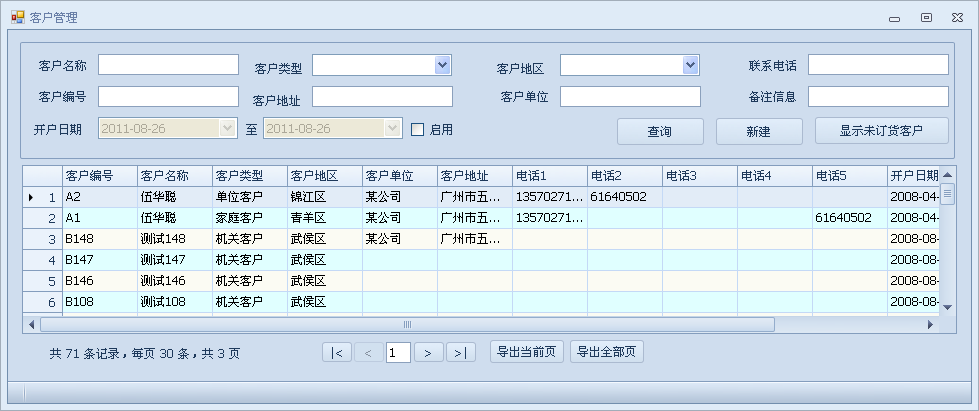


上图中，红色的菜单是分页控件内置的菜单项，其他程序附加是扩展菜单项目，其中复制选定行信息、设置表格显示列、打印列表是对表格基本的操作功能。下面两个版本的控件右键菜单均表现一致。

2） 基于DotNetbar界面控件版本



3） 基于DevExpress界面控件版本



使用分页控件的基础代码，在所有以上版本中，代码均一致，只是一些特殊的操作，需要利用到不同的控件，有所差异而已。初始化分页控件代码如下所示。

[复制代码](javascript:void(0);)

        private void FrmCustomer\_Load(object sender, EventArgs e)  
        {  
            this.winGridViewPager1.ProgressBar = this.toolStripProgressBar1.ProgressBar;  
            this.winGridViewPager1.OnPageChanged += new EventHandler(winGridViewPager1\_OnPageChanged);  
            this.winGridViewPager1.OnStartExport += new EventHandler(winGridViewPager1\_OnStartExport);  
            this.winGridViewPager1.OnEditSelected += new EventHandler(winGridViewPager1\_OnEditSelected);  
            this.winGridViewPager1.OnDeleteSelected += new EventHandler(winGridViewPager1\_OnDeleteSelected);  
            this.winGridViewPager1.OnRefresh += new EventHandler(winGridViewPager1\_OnRefresh);  
            this.winGridViewPager1.OnAddNew += new EventHandler(winGridViewPager1\_OnAddNew);  
            this.winGridViewPager1.AppendedMenu = this.contextMenuStrip1;//追加额外菜单项目  
            this.winGridViewPager1.ShowLineNumber = true;//显示行号  
            this.winGridViewPager1.PagerInfo.PageSize = 30;//页面大小  
            this.winGridViewPager1.EventRowBackColor = Color.LightCyan;//间隔颜色  
  
            BindData();  
       }

[复制代码](javascript:void(0);)

 以上是对分页控件的相关事件进行处理，如果实现了相关的事件，那么相应的菜单（如新增、编辑、删除等）就会在右键菜单中出现，否则不会出现，相应的事件就是具体对应操作的处理菜单操作。另外，分页控件的一些属性，如是否显示行号、页面显示记录大小、间隔颜色等属性也可以在这里设置。

这些初始化设置完成后，一般查询界面都会有一个条件的组合，然后通过组合条件查询数据库记录并显示，显示的时候，分页控件可以设置对应字段的中文显示名称，显示的字段列，显示顺序等。**显示字段的顺序就是列表的显示顺序，这个特性也是新版本增加的功能之一**。

[复制代码](javascript:void(0);)

       private void winGridViewPager1\_OnPageChanged(object sender, EventArgs e)  
        {  
            BindData();  
        }  
  
        /// <summary>  
        /// 根据查询条件构造查询语句  
        /// </summary>  
        /// <returns></returns>  
        private string GetSearchSql()  
        {  
            SearchCondition condition = new SearchCondition();  
            condition.AddCondition("Number", this.txtNumber.Text, SqlOperator.Like)  
                .AddCondition("Name", this.txtName.Text, SqlOperator.Like)  
                .AddCondition("Type", this.cmbType.Text, SqlOperator.Like)  
                .AddCondition("Area", this.cmbArea.Text, SqlOperator.Like)  
                .AddCondition("Address", this.txtAddress.Text, SqlOperator.Like)  
                .AddCondition("Company", this.txtCompany.Text, SqlOperator.Like)  
                .AddCondition("Note", this.txtNote.Text, SqlOperator.Like)  
                .AddCondition("Telephone1", this.txtTelephone.Text, SqlOperator.Like, true, "Telephone")  
                .AddCondition("Telephone2", this.txtTelephone.Text, SqlOperator.Like, true, "Telephone")  
                .AddCondition("Telephone3", this.txtTelephone.Text, SqlOperator.Like, true, "Telephone")  
                .AddCondition("Telephone4", this.txtTelephone.Text, SqlOperator.Like, true, "Telephone")  
                .AddCondition("Telephone5", this.txtTelephone.Text, SqlOperator.Like, true, "Telephone");  
  
            if (chkUseDate.Checked)  
            {  
                condition.AddCondition("CreateDate", dateTimePicker1.Value.ToString("yyyy-MM-dd"), SqlOperator.MoreThanOrEqual, true)  
                    .AddCondition("CreateDate", dateTimePicker2.Value.AddDays(1).ToString("yyyy-MM-dd"), SqlOperator.LessThanOrEqual, true);  
  
            }  
            string where = condition.BuildConditionSql().Replace("Where", "");  
            return where;  
        }  
  
        private void BindData()  
        {  
            #region 添加别名解析  
            //DisplayColumns与显示的字段名或者实体属性一致，大小写不敏感，顺序代表显示顺序，用逗号或者|分开  
            this.winGridViewPager1.DisplayColumns = "Number,NAME,type,Area,Company,Address,Telephone1,Telephone2,Telephone3,Telephone4,Telephone5,CreateDate,Note,LastUpdated";  
            this.winGridViewPager1.AddColumnAlias("ID", "编号");  
            this.winGridViewPager1.AddColumnAlias("Number", "客户编号");  
            this.winGridViewPager1.AddColumnAlias("Name", "客户名称");  
            this.winGridViewPager1.AddColumnAlias("Type", "客户类型");  
            this.winGridViewPager1.AddColumnAlias("Area", "客户地区");  
            this.winGridViewPager1.AddColumnAlias("Company", "客户单位");  
            this.winGridViewPager1.AddColumnAlias("Address", "客户地址");  
            this.winGridViewPager1.AddColumnAlias("Telephone1", "电话1");  
            this.winGridViewPager1.AddColumnAlias("Telephone2", "电话2");  
            this.winGridViewPager1.AddColumnAlias("Telephone3", "电话3");  
            this.winGridViewPager1.AddColumnAlias("Telephone4", "电话4");  
            this.winGridViewPager1.AddColumnAlias("Telephone5", "电话5");  
            this.winGridViewPager1.AddColumnAlias("CreateDate", "开户日期");  
            this.winGridViewPager1.AddColumnAlias("Shop\_ID", "分店ID");  
            this.winGridViewPager1.AddColumnAlias("Note", "备注");  
            this.winGridViewPager1.AddColumnAlias("LastUpdated", "更新日期");  
  
            #endregion  
  
            string where = GetSearchSql();  
            this.winGridViewPager1.DataSource = BLLFactory<Customer>.Instance.Find(where, this.winGridViewPager1.PagerInfo);  
        }  
  
        private void btnSearch\_Click(object sender, EventArgs e)  
        {  
            BindData();  
        }

[复制代码](javascript:void(0);)

很多人以前看过我介绍分页控件的用法的时候，总会觉得上面的数据绑定方式看似简单，实际操作可能绑定数据总是有问题，因为不知道Find函数里面需要实现那些操作，因为这些是基于我框架的应用逻辑。其实Find方法就是根据分页属性计算获取对应记录，然后把总记录数赋值给分页控件的属性。

下面我们来介绍一个不需要依赖我的Winform框架类，只是引用微软EnterpriseLibary数据库操作类即可完成的操作，因为最新的分页控件集成了一个基于Oracle、SqlServer、Access、Mysql数据库的分页逻辑，只需要使用PagerHelper这个类，传入相关的参数即可。

分页控件绑定数据如下代码所示。

[复制代码](javascript:void(0);)

        private void BindData()  
        {  
            #region 添加别名解析  
            //DisplayColumns与显示的字段名或者实体属性一致，大小写不敏感，顺序代表显示顺序，用逗号或者|分开  
            this.winGridViewPager1.DisplayColumns = "Number,NAME,type,Area,Company,Address,Telephone1,Telephone2,Telephone3,Telephone4,Telephone5,CreateDate,Note,LastUpdated";  
            this.winGridViewPager1.AddColumnAlias("ID", "编号");  
            this.winGridViewPager1.AddColumnAlias("Number", "客户编号");  
            this.winGridViewPager1.AddColumnAlias("Name", "客户名称");  
            this.winGridViewPager1.AddColumnAlias("Type", "客户类型");  
            this.winGridViewPager1.AddColumnAlias("Area", "客户地区");  
            this.winGridViewPager1.AddColumnAlias("Company", "客户单位");  
            this.winGridViewPager1.AddColumnAlias("Address", "客户地址");  
            this.winGridViewPager1.AddColumnAlias("Telephone1", "电话1");  
            this.winGridViewPager1.AddColumnAlias("Telephone2", "电话2");  
            this.winGridViewPager1.AddColumnAlias("Telephone3", "电话3");  
            this.winGridViewPager1.AddColumnAlias("Telephone4", "电话4");  
            this.winGridViewPager1.AddColumnAlias("Telephone5", "电话5");  
            this.winGridViewPager1.AddColumnAlias("CreateDate", "开户日期");  
            this.winGridViewPager1.AddColumnAlias("Shop\_ID", "分店ID");  
            this.winGridViewPager1.AddColumnAlias("Note", "备注");  
            this.winGridViewPager1.AddColumnAlias("LastUpdated", "更新日期");  
  
            #endregion  
  
            string where = GetSearchSql();  
            this.winGridViewPager1.DataSource = FindToDataTable(where, this.winGridViewPager1.PagerInfo);  
       }

[复制代码](javascript:void(0);)

FindToDataTable函数的相关实现代码如下所示：

[复制代码](javascript:void(0);)

        /// <summary>  
        /// 标准的记录查询函数  
        /// </summary>  
        /// <param name="where"></param>  
        /// <param name="pagerInfo"></param>  
        /// <returns></returns>  
        private DataTable FindToDataTable(string where, PagerInfo pagerInfo)  
        {  
            WHC.Pager.WinControl.PagerHelper helper = new WHC.Pager.WinControl.PagerHelper("All\_Customer", "\*", "LastUpdated", pagerInfo.PageSize, pagerInfo.CurrenetPageIndex, true, where);  
            string countSql = helper.GetPagingSql(WHC.Pager.WinControl.DatabaseType.Access, true);  
            string dataSql = helper.GetPagingSql(WHC.Pager.WinControl.DatabaseType.Access, false);  
  
            string value = SqlValueList(countSql);  
            pagerInfo.RecordCount = Convert.ToInt32(value);//为了显示具体的信息，需要设置总记录数  
            DataTable dt = SqlTable(dataSql);  
            return dt;  
        }  
  
        /// <summary>      
        /// 执行SQL查询语句，返回查询结果的所有记录的第一个字段,用逗号分隔。      
        /// </summary>      
        /// <param name="sql">SQL语句</param>      
        /// <returns>      
        /// 返回查询结果的所有记录的第一个字段,用逗号分隔。      
        /// </returns>      
        public string SqlValueList(string sql)  
        {  
            StringBuilder result = new StringBuilder();  
            Database db = DatabaseFactory.CreateDatabase();  
            DbCommand command = db.GetSqlStringCommand(sql);  
  
            using (IDataReader dr = db.ExecuteReader(command))  
            {  
                while (dr.Read())  
                {  
                    result.AppendFormat("{0},", dr[0].ToString());  
                }  
            }  
            string strResult = result.ToString().Trim(',');  
            return strResult;  
        }  
  
        /// <summary>      
        /// 执行SQL查询语句，返回所有记录的DataTable集合。      
        /// </summary>      
        /// <param name="sql">SQL查询语句</param>      
        /// <returns></returns>      
        public DataTable SqlTable(string sql)  
        {  
            Database db = DatabaseFactory.CreateDatabase();  
            DbCommand command = db.GetSqlStringCommand(sql);  
            return db.ExecuteDataSet(command).Tables[0];  
         }

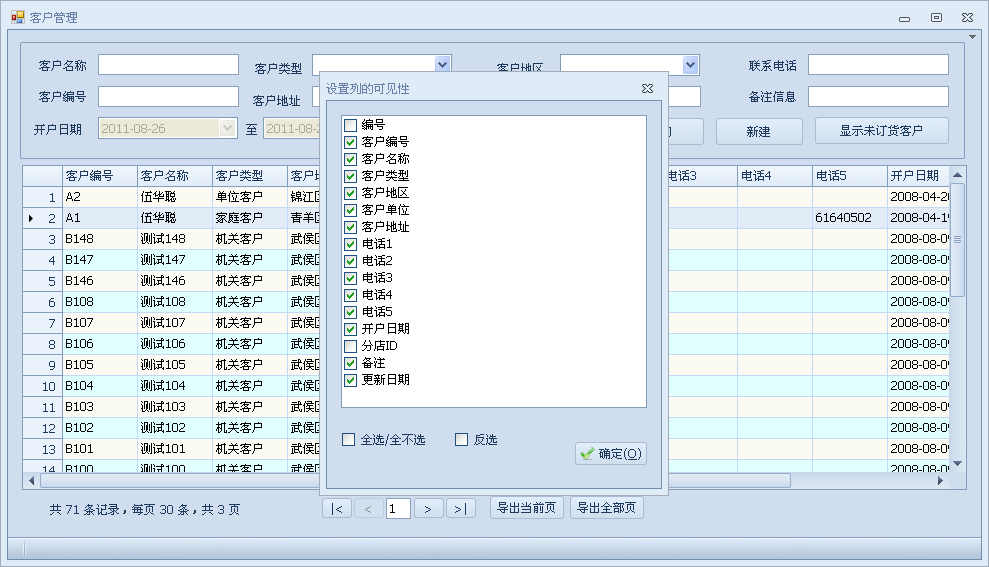
[复制代码](javascript:void(0);)

 利用控件内置的分页辅助类

WHC.Pager.WinControl.PagerHelper，你就可以不用理会各种不同数据库的分页实现逻辑，只需要按照以上的调用方式调用即可，**该类传入的第一个参数，既可以是表名，也可以是一段查询语句，非常方便各种复杂条件的查询分页**。

最新分页控件的相关界面截图如下所示。

1）设置表格列的可见性

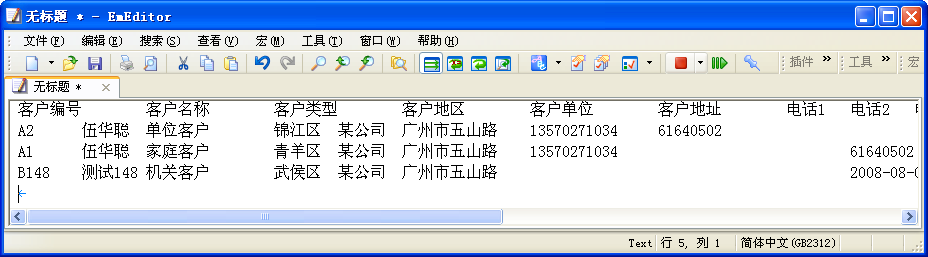
   
 2）表格数据打印预览



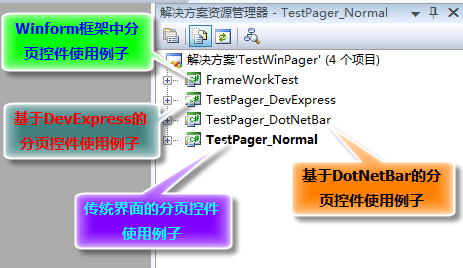
3）行数据基本信息提示



 4）复制选定行后在文本中查看



为了使得大家对我的Winform分页控件在不同情况下的深入了解和使用，我特意编写了几种情况下的使用标准例子，供大家下载学习研究，例子包括利用基于泛型模式的Winform开发框架结构的分页控件使用例子，基于传统简单数据绑定的分页控件使用例子， 基于DotNetBar界面控件及简单数据绑定的分页控件使用例子，基于DevExpress界面控件及简单数据绑定的分页控件使用例子，如下图所示。



各种例子的源码的下载地址如下：<http://www.iqidi.com/download/TestWinPager_Demo.rar>