 public class ExLookUpEdit : LookUpEdit  
    {  
        // Constructors  
        static ExLookUpEdit()  
        {  
            RepositoryItemExLookUpEdit.RegisterExLookUpEdit();  
        }

        public ExLookUpEdit()  
        {  
            this.sInputText = "";  
            this.sSql = "";  
            this.bRefreshDataSource = true;  
        }

        // Methods  
        public object GetFirstRowEditValue()  
        {  
            if (this.Properties.DataSource is DataTable)  
            {  
                DataTable table1 = this.Properties.DataSource as DataTable;  
                if (table1.Rows.Count > 0)  
                {  
                    return table1.Rows[0][this.Properties.ValueMember];  
                }  
            }  
            return null;  
        }

        protected override PopupBaseForm CreatePopupForm()  
        {  
            PopupExLookUpEditForm form1 = new PopupExLookUpEditForm(this);  
            form1.ShowQueryButtonEx = form1.Properties.ShowQueryButtonEx;  
            form1.ShowCloseButtonEx = form1.Properties.ShowCloseButtonEx;  
            form1.ShowOkButtonEx = form1.Properties.ShowOkButtonEx;  
            this.Properties.BestAllFit();  
            return form1;  
        }

        protected override void ProcessText(KeyPressHelper helper, bool canImmediatePopup)  
        {  
            this.sInputText = helper.Text;  
            if (this.sInputText == "")  
            {  
                this.Properties.DataAdapter.FilterExpression = "";  
                this.PopupForm.FilterCollectionChanged(this.Properties.DataAdapter, EventArgs.Empty);  
            }  
            base.ProcessText(helper, canImmediatePopup);  
            this.PopupForm.Filter.FilterPrefix = "";  
            if (this.Properties.DataAdapter.VisibleCount > 0)  
            {  
                this.PopupForm.SelectedIndex = 0;  
            }  
        }

        protected override void OnPopupClosed(PopupCloseMode closeMode)  
        {  
            base.OnPopupClosed(closeMode);  
            this.Properties.DataAdapter.FilterExpression = "";  
            if (this.PopupForm != null)  
            {  
                this.PopupForm.FilterCollectionChanged(this.Properties.DataAdapter, EventArgs.Empty);  
            }  
        }  
        //当焦点在这个控件上，提示文。  
        protected override void OnEnter(EventArgs e)  
        {  
            base.OnEnter(e);  
            if (this.bRefreshDataSource)  
            {  
                if (this.Properties.Tag is Label)  
                {  
                    Label label1 = this.Properties.Tag as Label;  
                    label1.Text = "按Shift+F5可以刷新" + this.Caption.Trim() + "的数据";

                }  
                if (this.Properties.Tag is BarStaticItem)  
                {  
                    BarStaticItem item1 = this.Properties.Tag as BarStaticItem;  
                    if ((this.Caption != null) && (this.Caption != ""))  
                    {  
                        item1.Caption = "按Shift+F5可以刷新" + this.Caption.Trim() + "的数据";  
                    }  
                    else if ((this.ToolTipTitle != null) && (this.ToolTipTitle != ""))  
                    {  
                        item1.Caption = "按Shift+F5可以刷新" + this.ToolTipTitle.Trim() + "的数据";  
                    }  
  
                }  
            }  
        }  
        //当焦点离开时，提示文为空。  
        protected override void OnLeave(EventArgs e)  
        {  
            base.OnLeave(e);  
            if (this.bRefreshDataSource)  
            {  
                if (this.Properties.Tag is Label)  
                {  
                    Label label1 = this.Properties.Tag as Label;  
                    label1.Text = "";  
                }  
                else  
                {  
                    if (this.Properties.Tag is BarStaticItem)  
                    {  
                        BarStaticItem item1 = this.Properties.Tag as BarStaticItem;  
                        item1.Caption = "";  
                    }  
                }  
            }  
        }

        // Properties  
        [Description("Gets the class name of the current editor.")]  
        public override string EditorTypeName  
        {  
            get  
            {  
                return "ExLookUpEdit";  
            }  
        }

        [Description("Specifies settings specific to the current editor."), DesignerSerializationVisibility(DesignerSerializationVisibility.Content), Category("Properties")]  
        public RepositoryItemExLookUpEdit Properties  
        {  
            get  
            {  
                return base.Properties as RepositoryItemExLookUpEdit;  
            }  
        }

        internal PopupExLookUpEditForm PopupForm  
        {  
            get  
            {  
                return base.PopupForm as PopupExLookUpEditForm;  
            }  
        }

        internal string InputText  
        {  
            get  
            {  
                return this.sInputText;  
            }  
        }

        public string SqlState  
        {  
            get  
            {  
                return this.sSql;  
            }  
            set  
            {  
                this.sSql = value;  
            }  
        }

        [Browsable(true)]  
        public string Caption  
        {  
            get  
            {  
                if (this.Properties.TextButton != null)  
                {  
                    return this.Properties.TextButton.Caption;  
                }  
                return "";  
            }  
            set  
            {  
                if (this.Properties.TextButton != null)  
                {  
                    this.Properties.TextButton.Caption = value;  
                }  
            }  
        }

        [Browsable(true)]  
        public Color CaptionForeColor  
        {  
            get  
            {  
                if (this.Properties.TextButton != null)  
                {  
                    return this.Properties.TextButton.Appearance.ForeColor;  
                }  
                return Color.Empty;  
            }  
            set  
            {  
                if (this.Properties.TextButton != null)  
                {  
                    this.Properties.TextButton.Appearance.ForeColor = value;  
                }  
            }  
        }

        [Browsable(true)]  
        public bool RefreshDataSource  
        {  
            get  
            {  
                return this.bRefreshDataSource;  
            }  
            set  
            {  
                if (this.bRefreshDataSource != value)  
                {  
                    this.bRefreshDataSource = value;  
                }  
            }  
        }

        // Instance Fields  
        private string sInputText;  
        private string sSql;  
        private bool bRefreshDataSource;  
    }