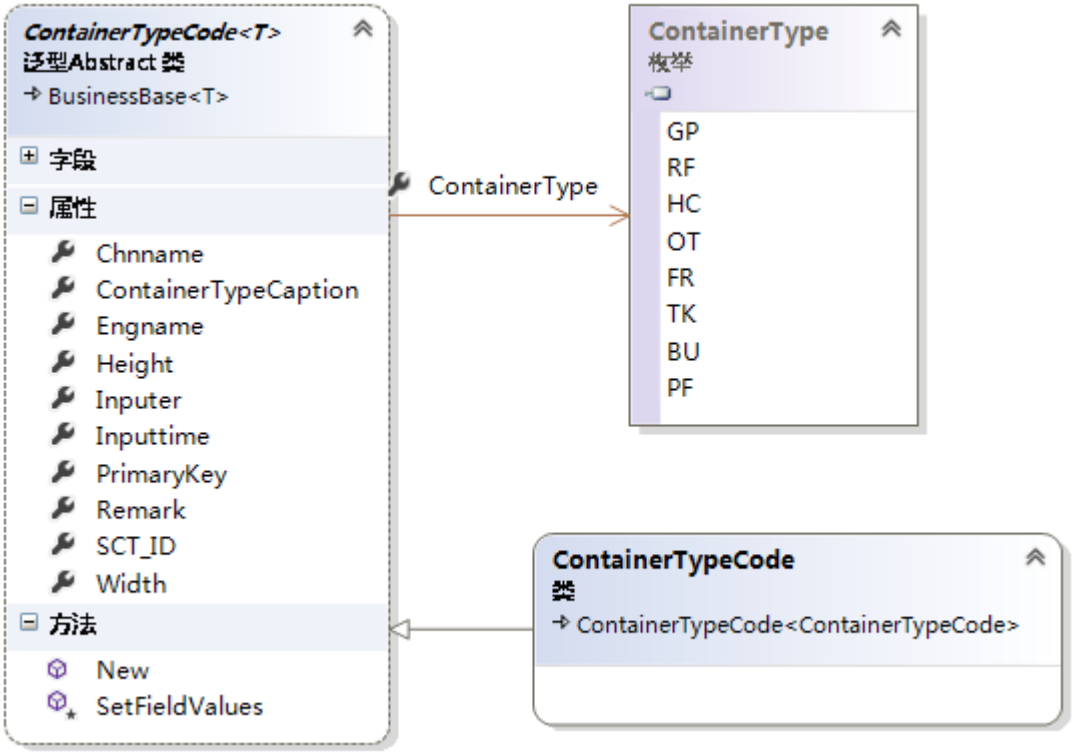
1 如何将枚举属性绑定到 LookUpEdit

DevExpress 是一个比较有名的界面控件套件,提供了一系列的界面控件套件的 DotNet 界面控件。Phenix丶在 WinForm 框架上主要以整合 DevExpress 为主，并利用了其便捷的界面设计功能，实现了快速开发目的。

本文主要介绍了业务对象或查询对象的枚举属性如何绑定到 LookUpEdit 控件上的方法，也顺便介绍了涉及到的相关开发技巧，使得整个演示过程是连贯的、可模仿的。

1.1 枚举属性

以箱型代码类为例：



1.1.1 设计业务类

箱型代码类的箱型属性：

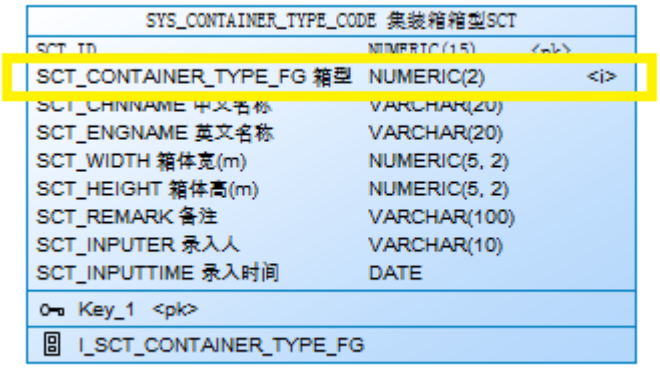
```
        /// <summary>
        /// 箱型
        /// </summary>
        public static readonly Phenix.Business.PropertyInfo<ContainerType?> ContainerTypeProperty = RegisterProperty<ContainerType?>(c => c.ContainerType);
        [Phenix.Core.Mapping.Field(FriendlyName = "箱型", Alias = "SCT_CONTAINER_TYPE_FG", TableName = "SYS_CONTAINER_TYPE_CODE", ColumnName = "SCT_CONTAINER_TYPE_FG", NeedUpdate = true)]
```

```
        private ContainerType? _containerType;
        /// <summary>
        /// 箱型
        /// </summary>
        [System.ComponentModel.DisplayName("箱型")]
        public ContainerType? ContainerType
        {
            get { return GetProperty(ContainerTypeProperty, _containerType); }
            set
            {
                SetProperty(ContainerTypeProperty, ref _containerType, value);
                _containerTypeCaption = null;
            }
           }
        [NonSerialized]
        [Csla.NotUndoable]
        private string _containerTypeCaption;
        /// <summary>
        /// 箱型标签
        /// </summary>
        [System.ComponentModel.DisplayName("箱型标签")]
        public string ContainerTypeCaption
        {
            get
            {
                if (_containerTypeCaption == null)
                    _containerTypeCaption = _containerType.HasValue ?
Phenix.Core.Rule.EnumKeyCaption.GetCaption(_containerType.Value) : String.Empty;
                return _containerTypeCaption;
            }
         }
```

这些代码是通过 Phenix丶的 Addin 工具“初始化/编辑业务类”功能，根据数据库的数据字典自动生成的。

1.1.2 设计表结构

请事先在数据库中 Create 表或视图：



1.1.3 枚举字段的设计规范

如果要在业务类中自动生成为枚举属性，且无需手工改写代码，就得满足以下几点要求（具体见“设计规范.数据库设计规范”）：

- 字段名：后缀为“_FG”；
- 字段类型：为 NUMERIC(2)；
- 枚举名与字段名对应关系：是剔去字段名前缀（“XXX_”）和后缀（“_FG”）的 Pascal 命名法来命名为枚举名；

1.1.4 设计枚举

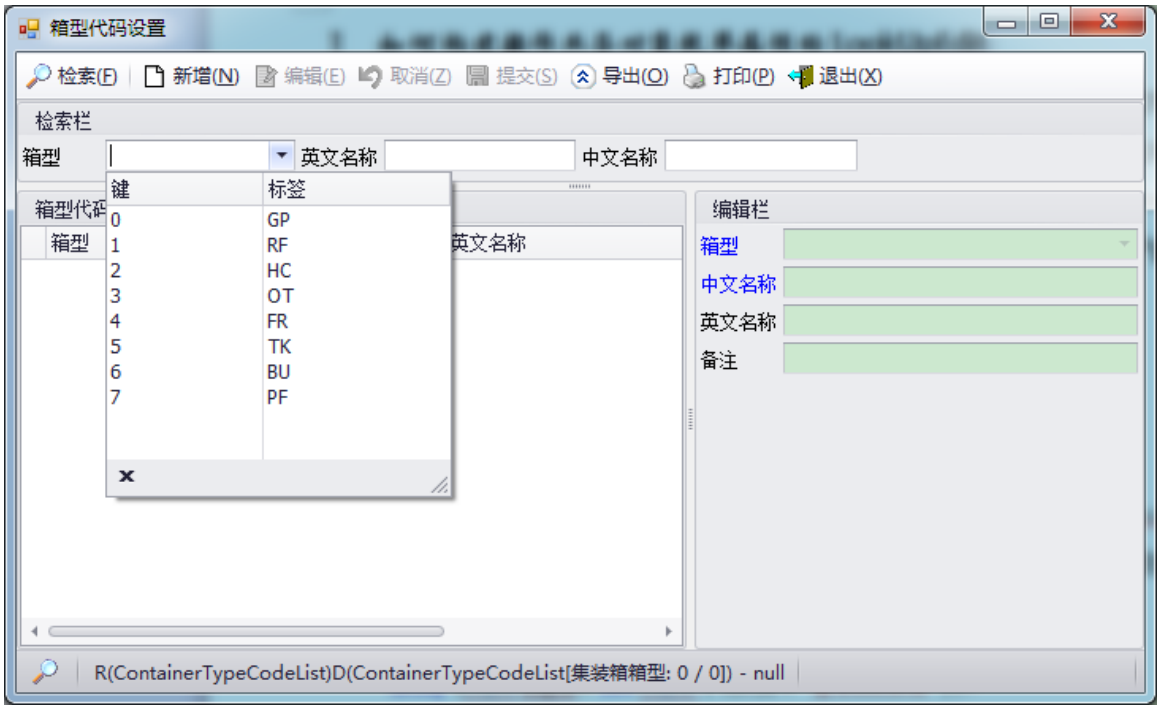
箱型枚举：

```
/// <summary>
/// 箱型
/// </summary>
[Phenix.Core.Operate.KeyCaptionAttribute(FriendlyName = "箱型"), System.SerializableAttribute()]
public enum ContainerType
{
    /// <summary>
    /// 干货箱
    /// </summary>
    [Phenix.Core.Rule.EnumCaptionAttribute("GP")]
    GP,

    /// <summary>
    /// 冷藏箱
    /// </summary>
    [Phenix.Core.Rule.EnumCaptionAttribute("RF")]
    RF,
```

```
        /// <summary>
        /// 挂衣箱
        /// </summary>
        [Phenix.Core.Rule.EnumCaptionAttribute("HC")]
        HC,

        /// <summary>
        /// 开顶箱
        /// </summary>
        [Phenix.Core.Rule.EnumCaptionAttribute("OT")]
        OT,

        /// <summary>
        /// 框架箱
        /// </summary>
        [Phenix.Core.Rule.EnumCaptionAttribute("FR")]
        FR,

        /// <summary>
        /// 罐状箱
        /// </summary>
        [Phenix.Core.Rule.EnumCaptionAttribute("TK")]
        TK,

        /// <summary>
        /// 散货箱
        /// </summary>
        [Phenix.Core.Rule.EnumCaptionAttribute("BU")]
        BU,

        /// <summary>
        /// 平板箱
        /// </summary>
        [Phenix.Core.Rule.EnumCaptionAttribute("PF")]
        PF,
    }
```

注意枚举及其属性上的标签，都是可以通过 Phenix丶的 Addin 工具“添加枚举标签”功能，根据注释内容自动生成的。

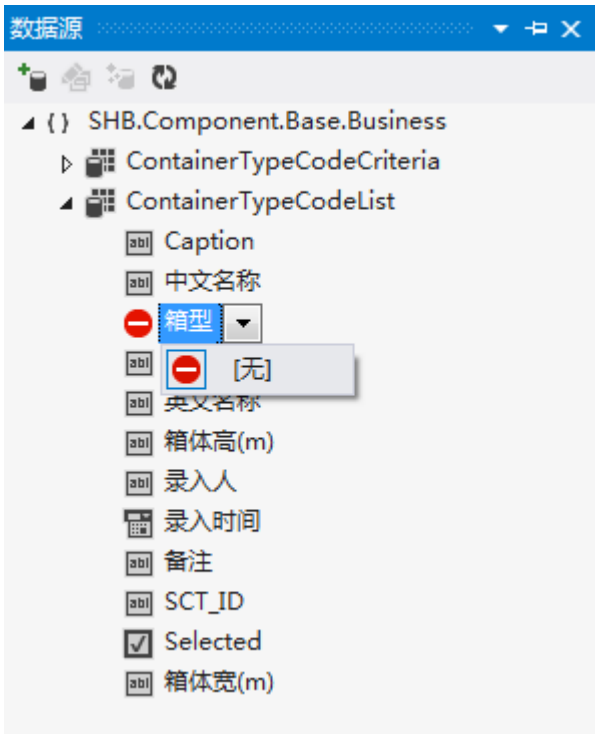
这些枚举标签的作用，是可以与 Phenix.Core.Rule.EnumKeyCaptionCollection 类一起，实现界面上的绑定、显示枚举的清单：



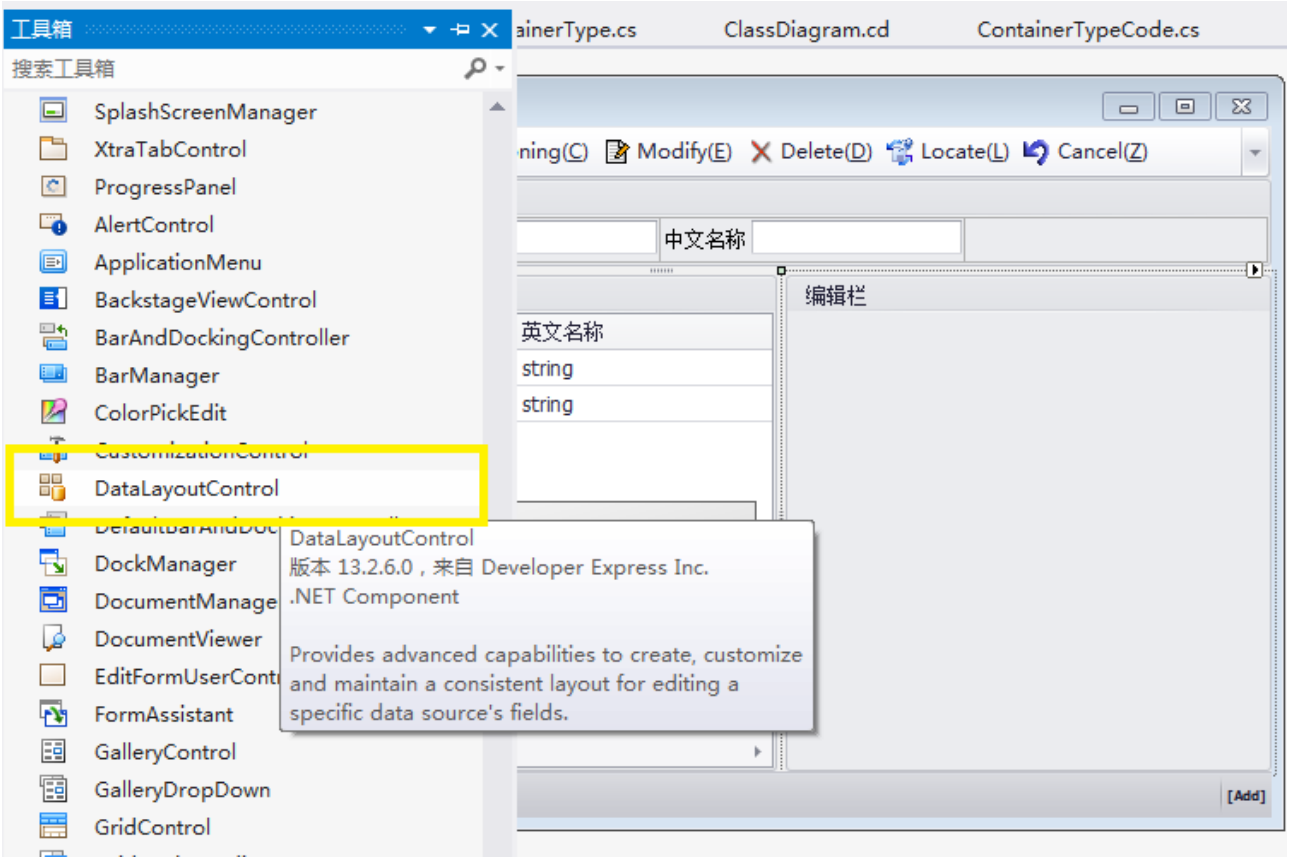
上述界面就是本示例要一步步设计实现的。

1.2 绑定 LookUpEdit 控件

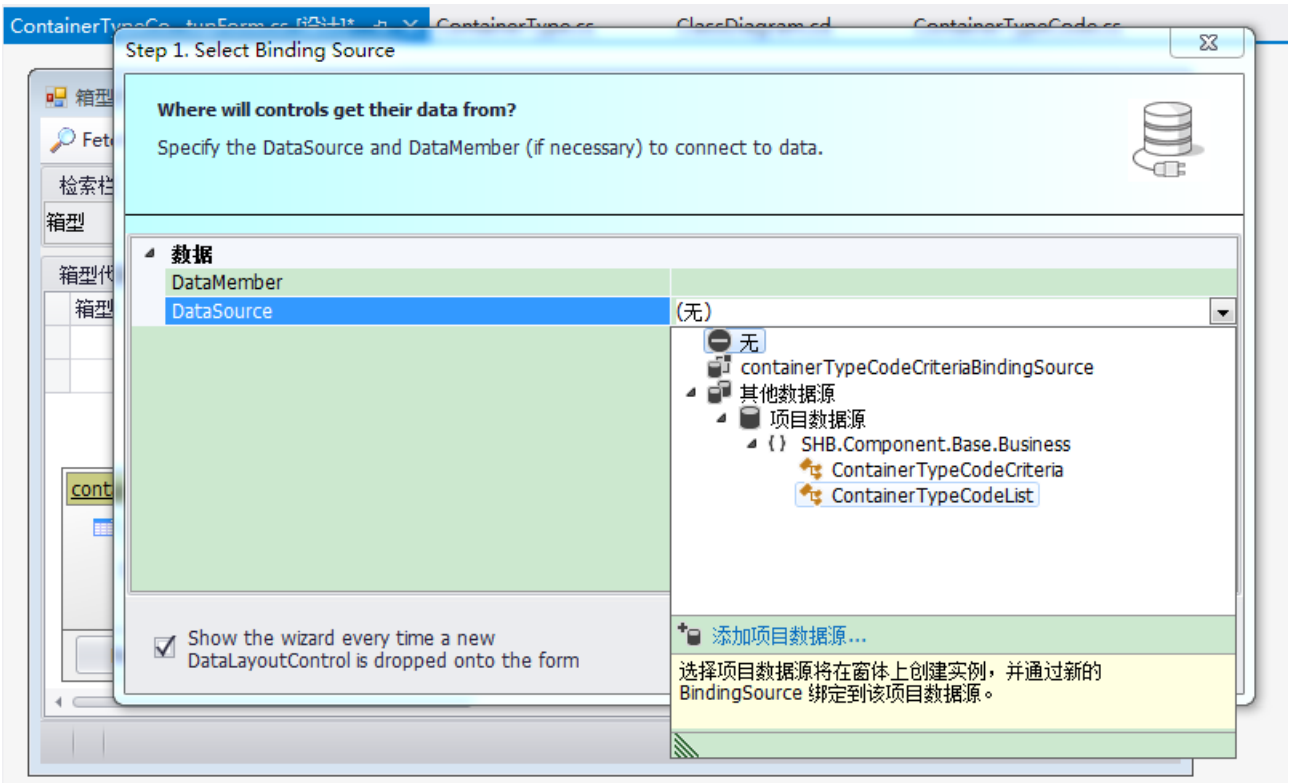
鉴于 IDE 对 Nullable<T>结构的枚举属性支持的并不好：



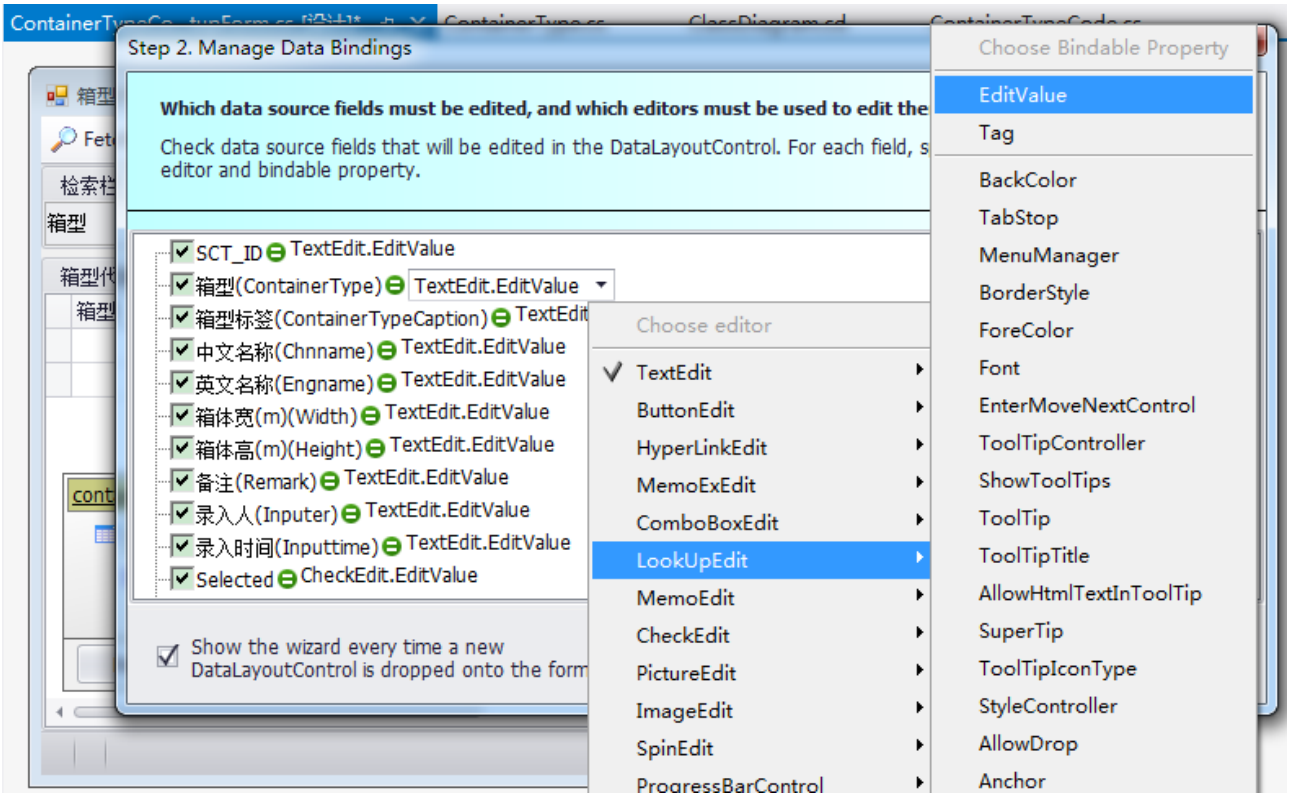
无法通过数据源单独拖放绑定的控件到界面上。我们可以通过 DevExpress 控件包的 DataLayoutControl 来实现：



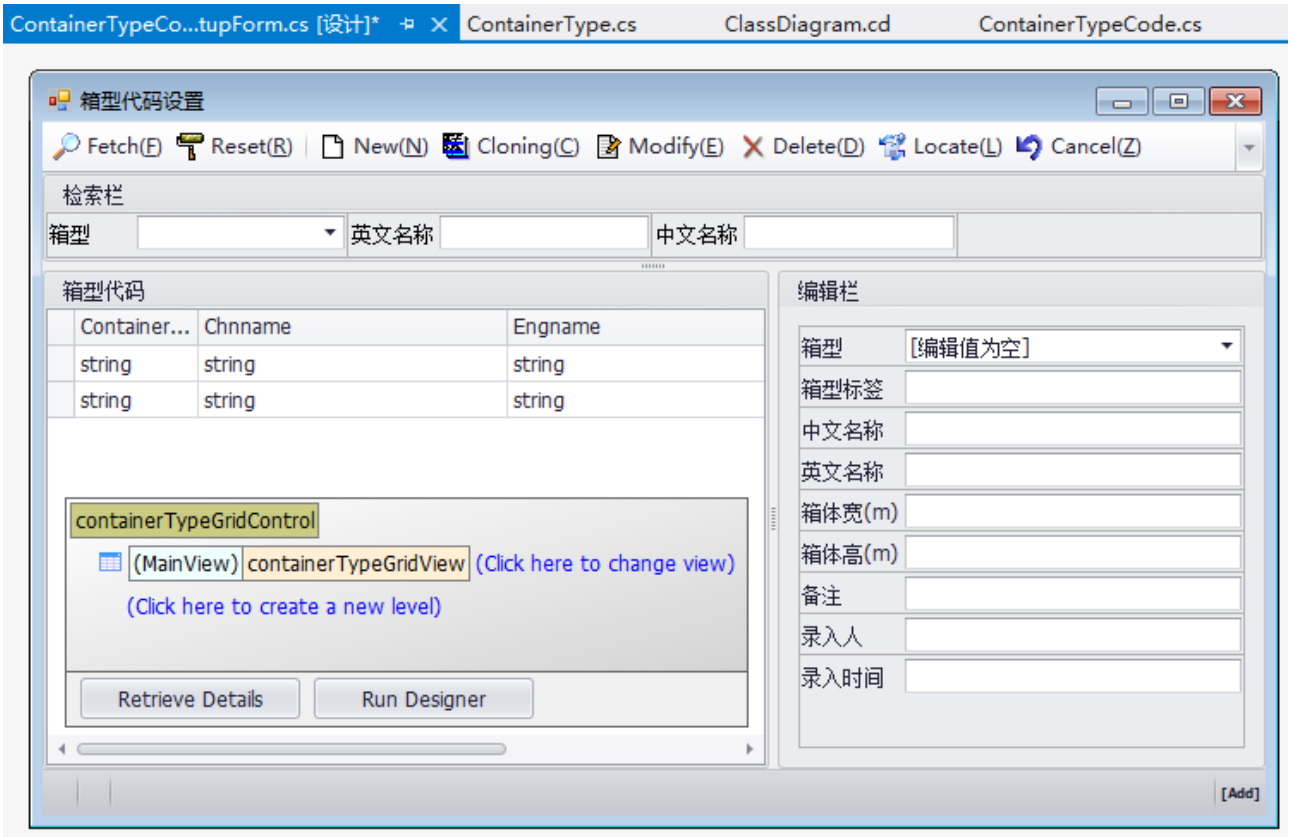
将它拖到编辑栏中时，会弹出引导对话框。首先选择绑定的数据源：



接下来，将箱型枚举属性绑定到 LookUpEdit 控件上：


Step 2. Manage Data Bindings
Which data source fields must be edited, and which editors must be used to edit the
Check data source fields that will be edited in the DataLayoutControl. For each field, s
editor and bindable property.
SCT_ID TextEdit.EditValue
箱型(ContainerType) TextEdit.EditValue
箱型标签(ContainerTypeCaption) TextEdit
中文名称(Chnname) TextEdit.EditValue
英文名称(Engname) TextEdit.EditValue
箱体宽(m)(Width) TextEdit.EditValue
箱体高(m)(Height) TextEdit.EditValue
备注(Remark) TextEdit.EditValue
录入人(Inputer) TextEdit.EditValue
录入时间(Inputtime) TextEdit.EditValue
Selected CheckEdit.EditValue
Show the wizard every time a new
DataLayoutControl is dropped onto the form
Choose editor
TextEdit
ButtonEdit
HyperLinkEdit
MemoExEdit
ComboBoxEdit
LookUpEdit
MemoEdit
CheckEdit
PictureEdit
ImageEdit
SpinEdit
ProgressBarControl
Choose Bindable Property
EditValue
Tag
BackColor
TabStop
MenuManager
BorderStyle
ForeColor
Font
EnterMoveNextControl
ToolTipController
ShowToolTips
ToolTip
ToolTipTitle
AllowHtmlTextInToolTip
SuperTip
ToolTipIconType
StyleController
AllowDrop
Anchor

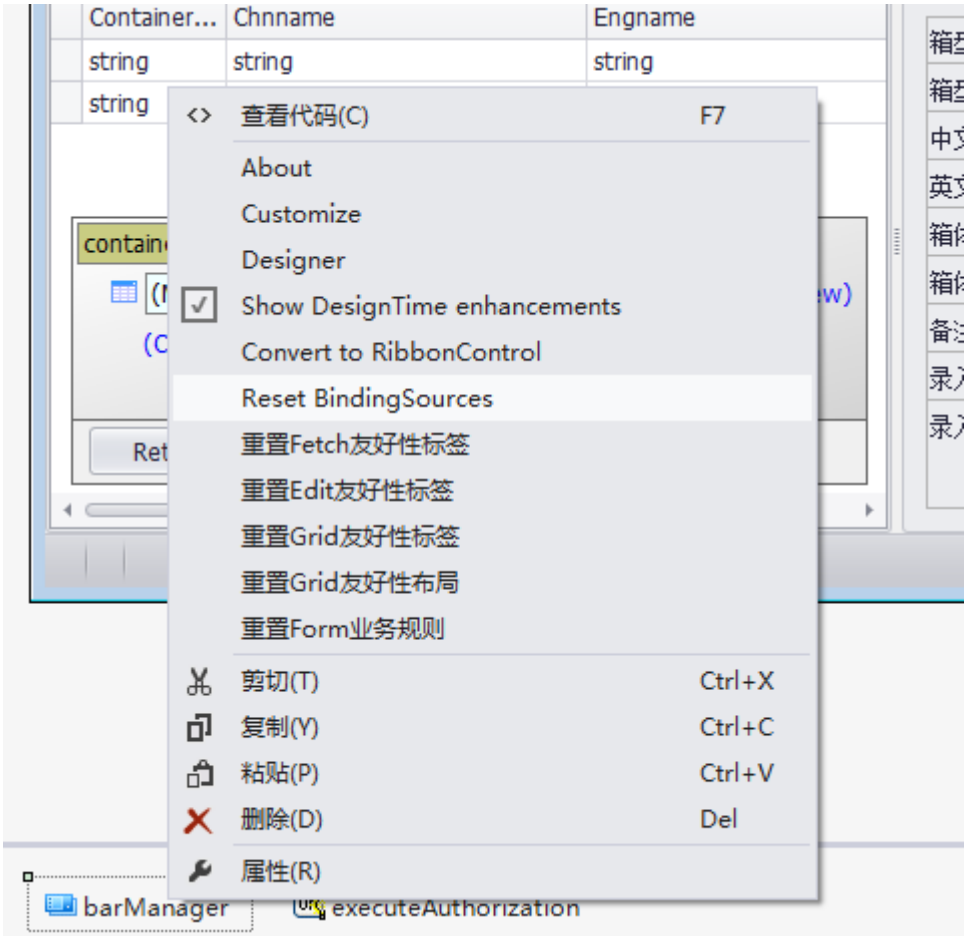
然后一步步确认，最后：


ContainerTypeCo...tupForm.cs [设计]*
ContainerType.cs
ClassDiagram.cd
ContainerTypeCode.cs
箱型代码设置
Fetch(F)
Reset(R)
New(N)
Cloning(C)
Modify(E)
Delete(D)
Locate(L)
Cancel(Z)
检索栏
箱型
英文名称
中文名称
箱型代码
Container...
Chnname
Engname
string
string
string
string
string
string
containerTypeGridControl
(MainView)
containerTypeGridView
(Click here to change view)
(Click here to create a new level)
Retrieve Details
Run Designer
编辑栏
箱型
[编辑值为空]
箱型标签
中文名称
英文名称
箱体宽(m)
箱体高(m)
备注
录入人
录入时间
[Add]

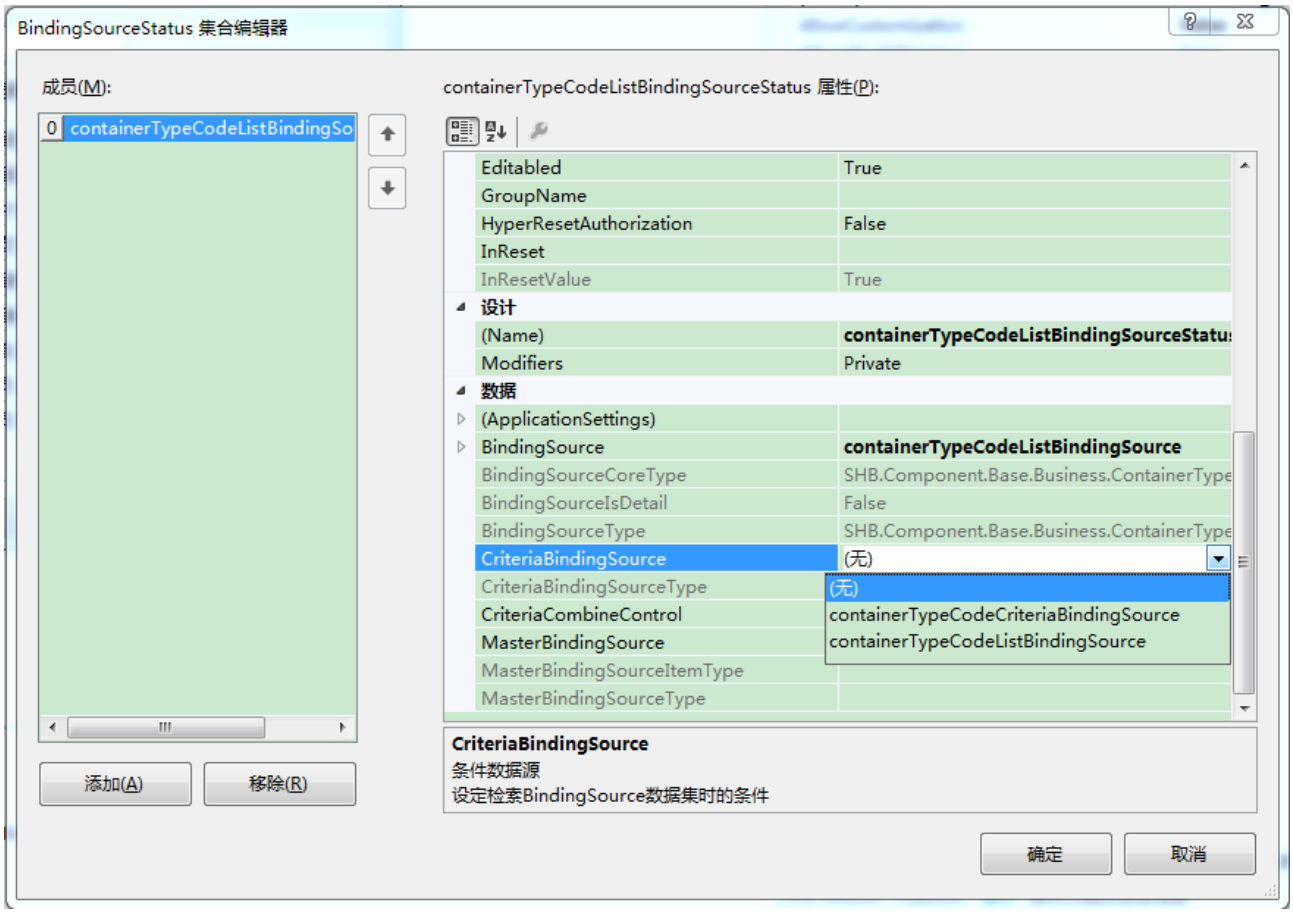
1.3 设置 LookUpEdit 下拉框属性

接下来要实现下拉框选择箱型枚举清单的功能。

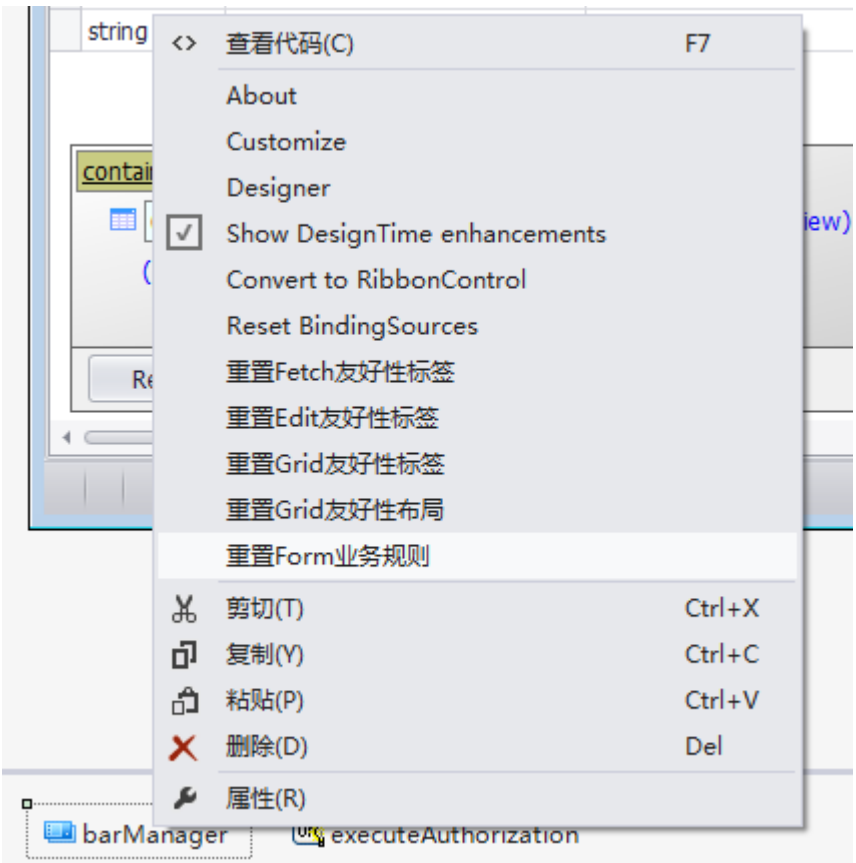
首先，将 BarManager 组件的 BindingSources 属性刷新一下：



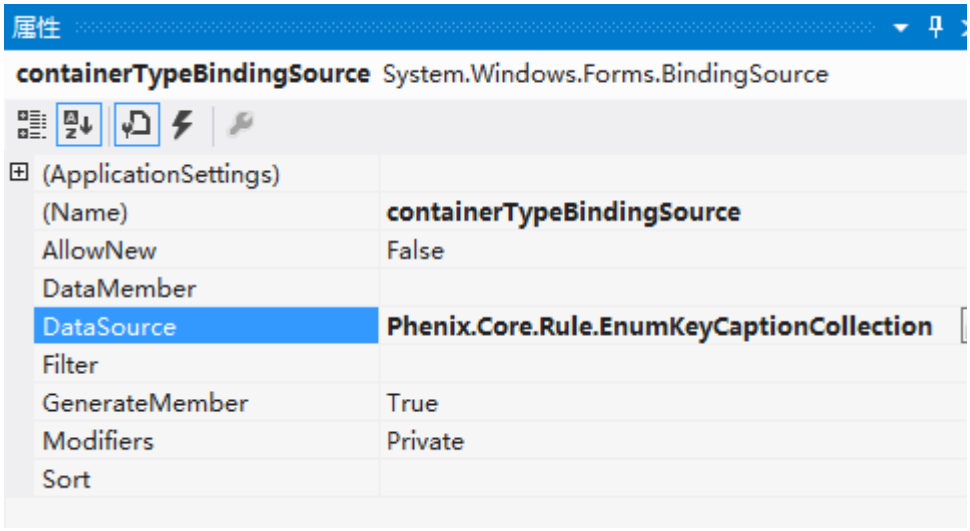
效果如下（可点开 BindingSources 属性，顺便关联上查询对象的 BindingSource）：



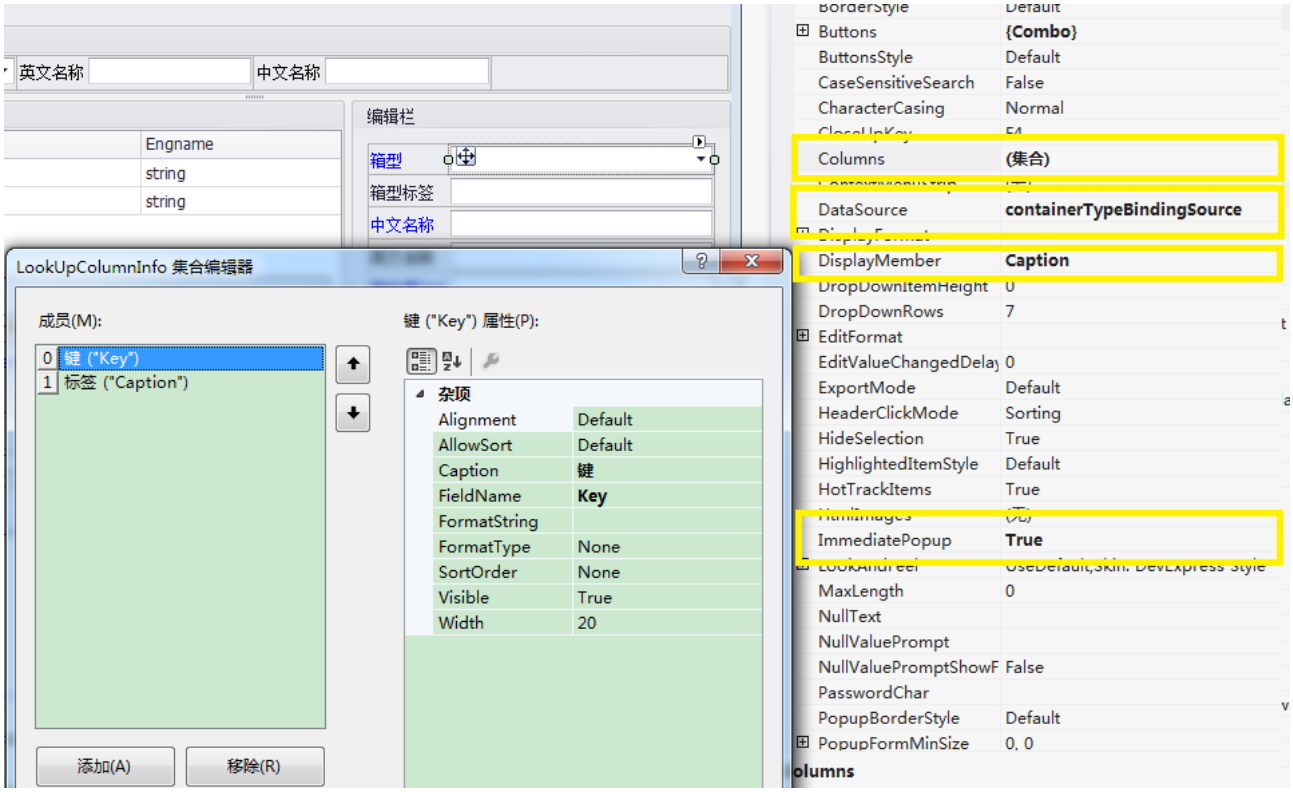
然后，重置 Form 业务规则：



此时会在组件栏里自动新增已绑定了 Phenix.Core.Rule.EnumKeyCaptionCollection 类的 BindingSource 组件：



而绑定箱型枚举属性的 LookUpEdit 控件，其下拉选择框的相关属性，也与这个 EnumKeyCaptionCollection 的 BindingSource 组件进行了关联和设置：



至此，我们完成了枚举属性的界面设计过程，期间没有手工编写一行代码，除了之前枚举类的设计过程。