7 ExecuteAuthorization 组件

所在程序集：Phenix.Services.Client

命名空间：Phenix.Services.Client.Security

组件功能：控制 WinForm 界面上组件所绑定业务对象过程的执行授权，涉及其属性：Enabled、Visible(Visibility)

7.1 前提条件

- 窗体所定义的类，需继承自 Phenix.Core.Windows.BaseForm，如果无法继承，则必须标记上[Phenix.Core.Dictionary.DataDictionary(Phenix.Core.Dictionary.AssemblyClassType.Form)]；
- 窗体所在的程序集，需通过 Phenix.Services.Host.exe 注册到配置库中，具体见“05.业务对象公共接口的授权”章节；

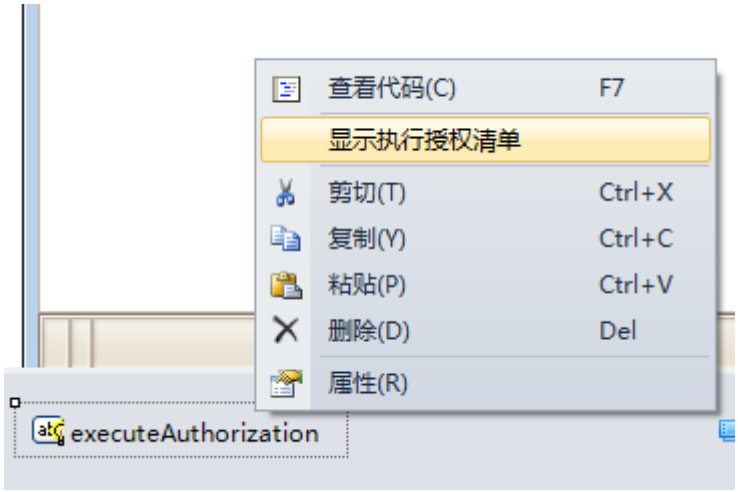
7.2 简介

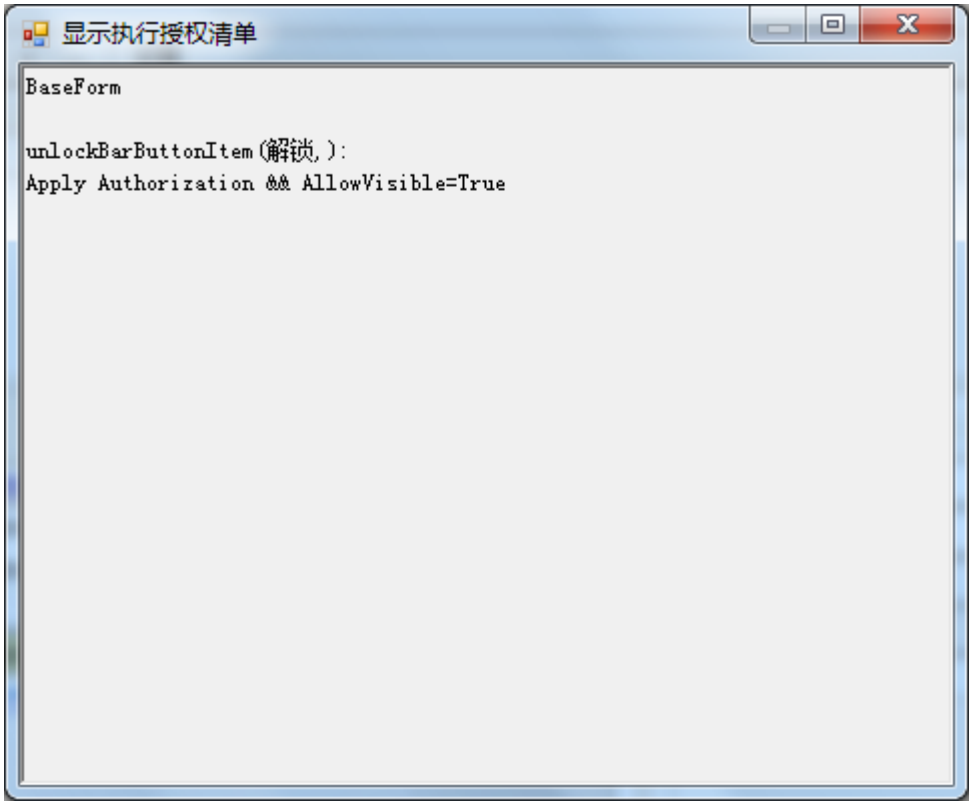
ExecuteAuthorization 组件属于界面控制层。

将业务对象过程的执行授权体现在 UI 界面上，使得用户的体验非常友好，可以有效引导用户操作，并预防了误操作。

7.3 配置

ExecuteAuthorization 组件是通过将属性扩展到容器中控件/其他组件的方法来实现配置的，如果要查看整体配置情况的话，可以：

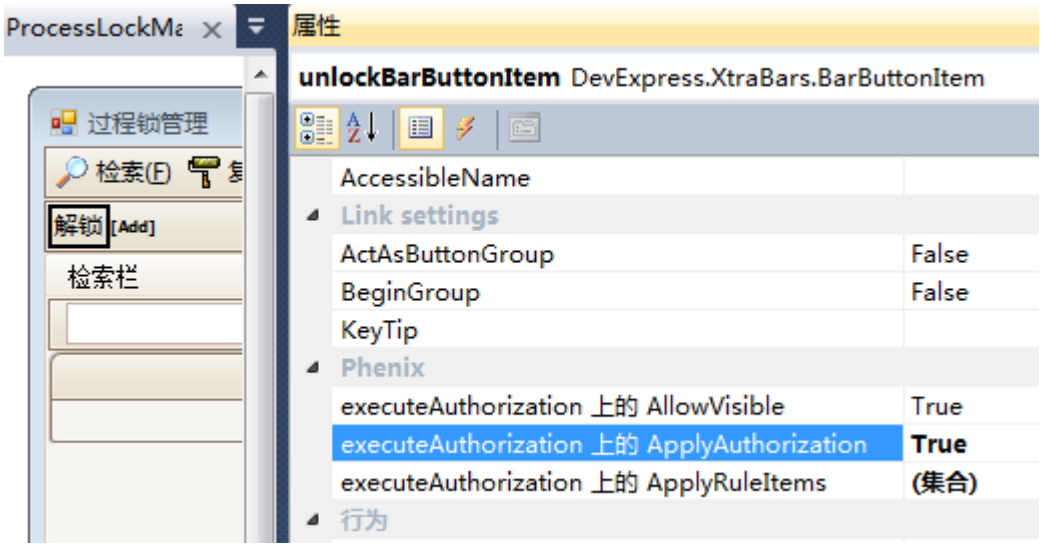




下面是供配置的属性：

属性	说明
ApplyAuthorization	是否应用授权：当为 true 时，被扩展控件将受到本组件的执行授权规则控制
AllowVisible	是否允许可见：本组件被应用到授权，当授权不可执行时是否允许可见
ApplyRuleItems	执行授权规则内容：本组件绑定到具体的执行授权规则队列

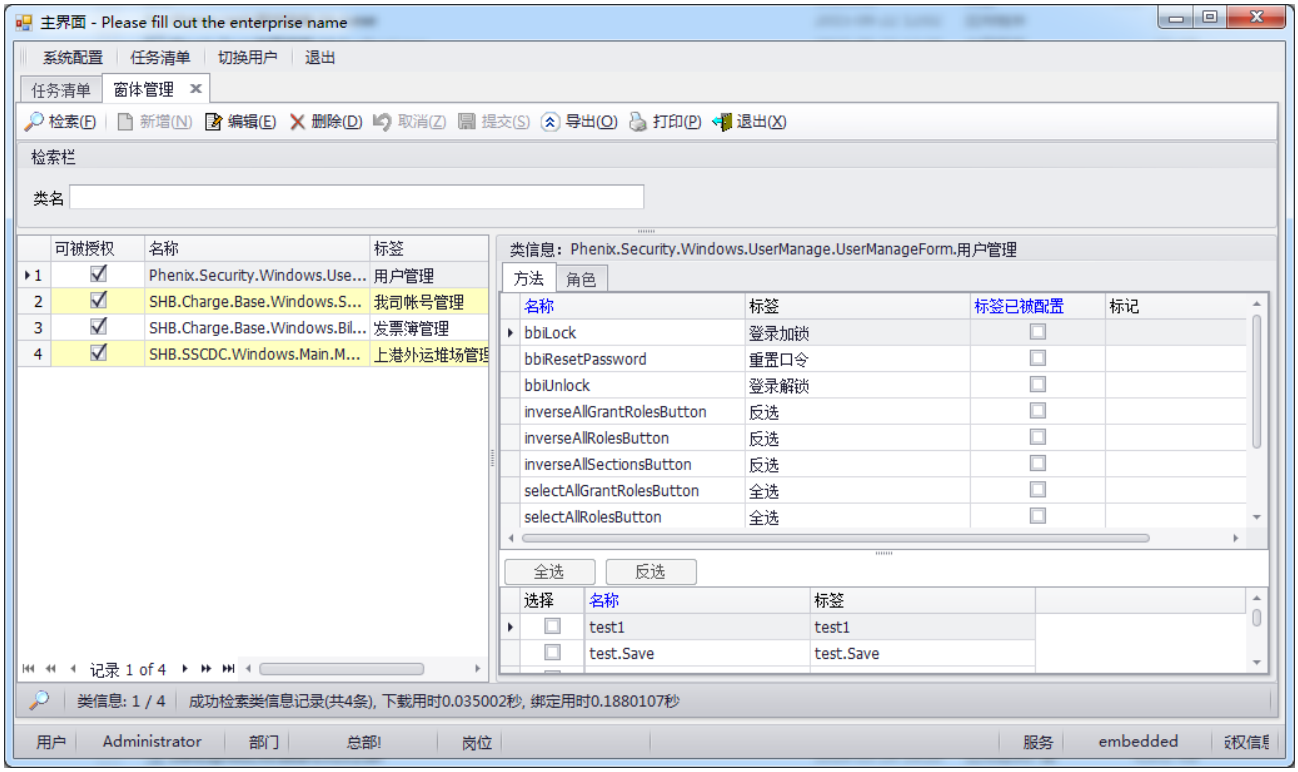
设计界面：



组件的扩展属性 ApplyAuthorization 缺省值为 false，只有当它设置为 true 后，它才会执行授权规则控制。请不要随意打开执行授权功能，否则应用系统的运行性能会有些微的下降。

缺省情况下，组件在不可使用的时候是变灰的，如果希望此时隐藏掉这个组件，则将其扩展属性 AllowVisible 缺省值设置为 true 就可以达到目的。

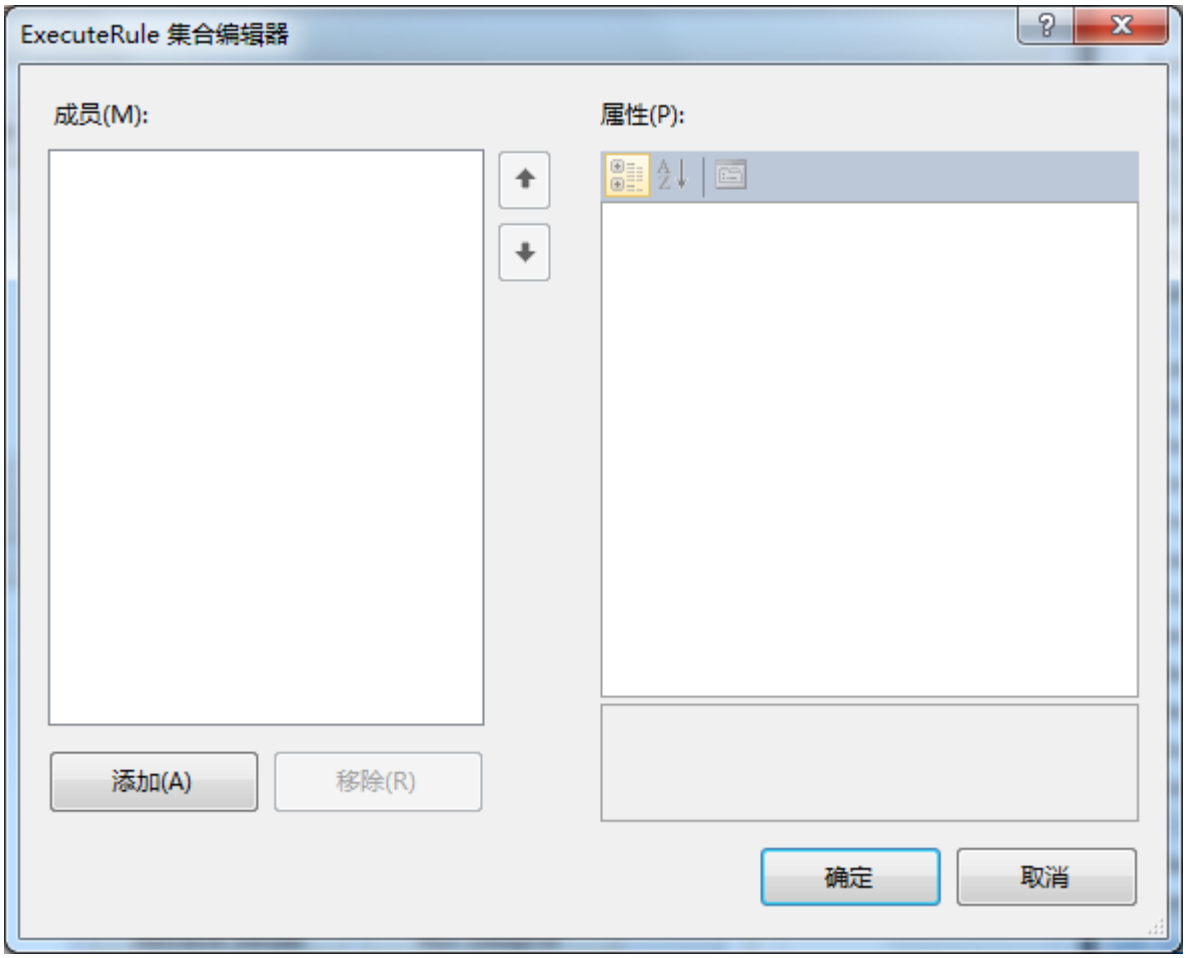
在组件扩展属性 ApplyAuthorization 等于 true 的情况下，如果希望权限管理能够直接控制这个界面上组件的话，则需要在设计好本界面组件之后，通过 Phenix.Services.Host.exe 来注册它所在的程序集，这样 Phenix.Security 权限管理功能组件包（见“Phenix.Security.Windows.FormClassManage”、“Phenix.Security.Windows.RoleManage”、“Phenix.Security.Windows.UserManage”工程）就可以将它与角色进行关联，实现用户权限的控制：



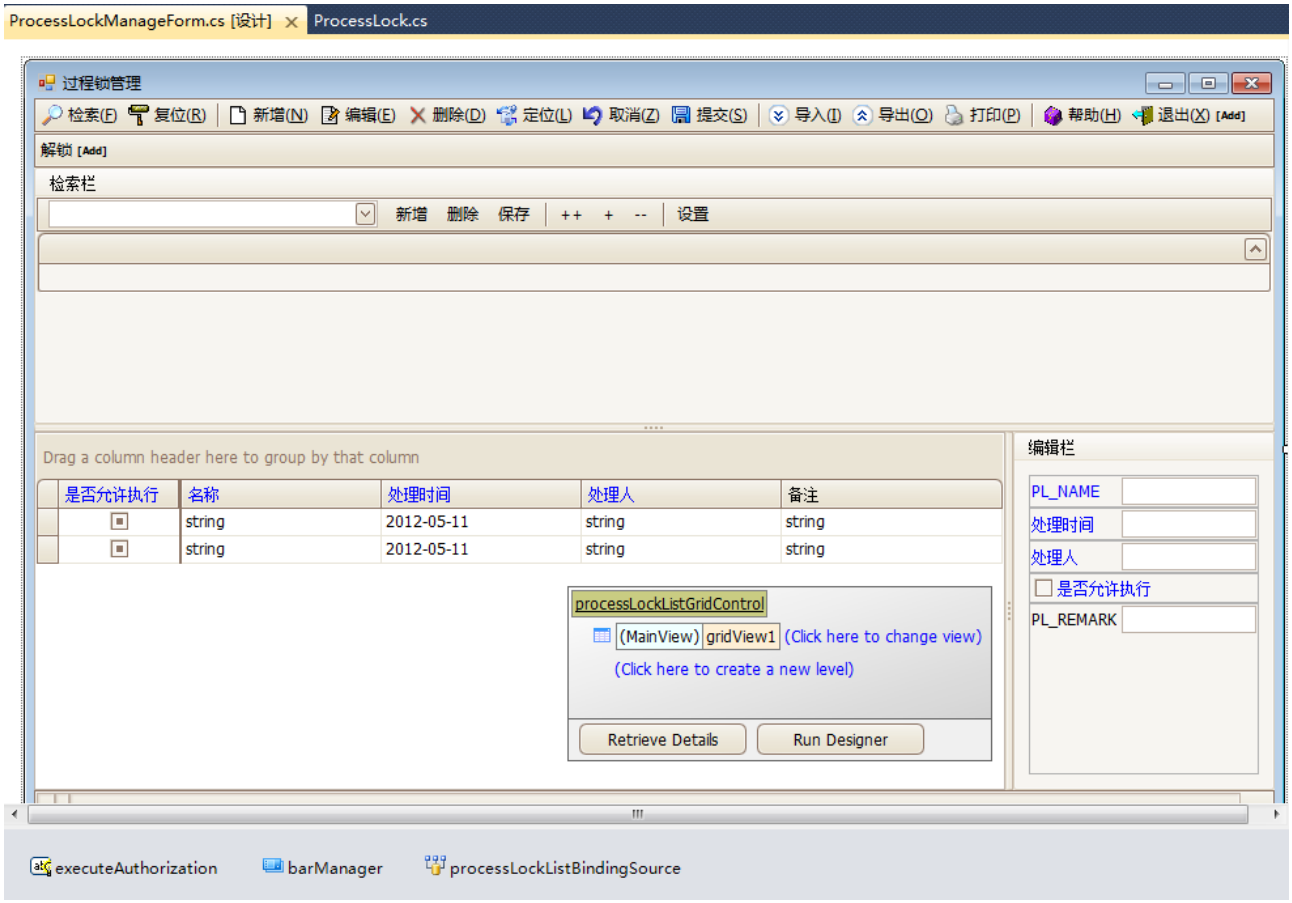
但是，如果仅仅是希望通过绑定的业务对象执行授权规则来做授权管控的话，就无需注册界面组件。否则，对系统性能来说是平添了份负担。

7.3.1 绑定业务对象过程的执行授权规则

业务对象过程的执行授权规则是通过组件扩展属性 ApplyRuleItems 进行绑定关联的：



我们以 ProcessLock 管理界面的设计为例：



界面上有一“解锁”功能按钮，执行的是当前 ProcessLock 业务对象的 Unlcok()函数：

```
private void unlockBarButtonItem_ItemClick(object sender, DevExpress.XtraBars.ItemClickEventArgs e)
{
  WorkingProcessLock.Unlock();
}
```

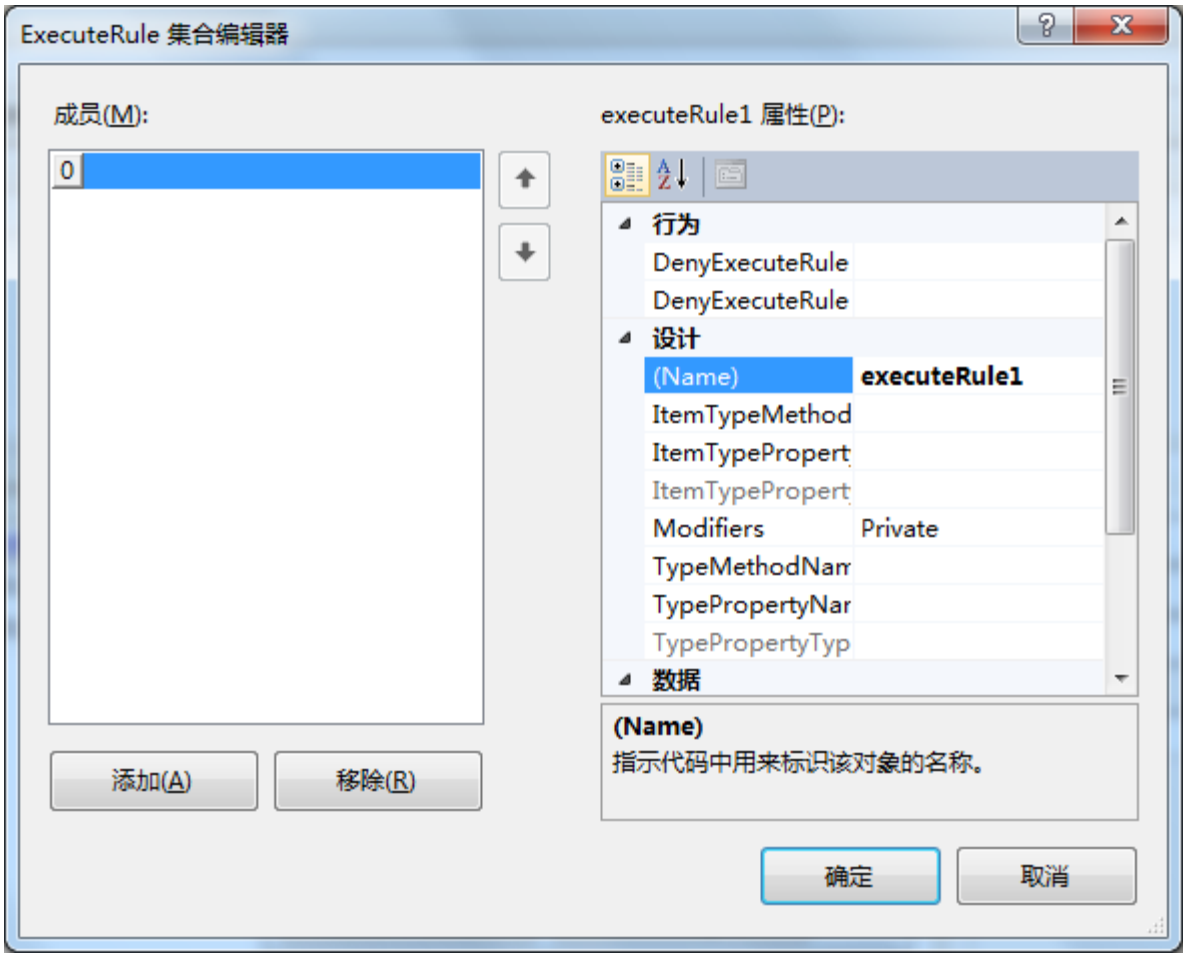
我们在《Phenix.NET 企业应用软件快速开发平台使用指南.03.Addin 工具使用方法》“构建执行过程授权规则类”章节中已讲解了如何使得 ProcessLock 业务对象的 Unlcok()函数可被授权：

```
/// <summary>
/// 解锁
/// </summary>
public static readonly Phenix.Business.MethodInfo UnlockMethod = RegisterMethod(c => c.Unlock());
/// <summary>
/// 解锁
/// </summary>
[Phenix.Core.Mapping.Method(FriendlyName = "解锁")]
public void Unlock()
{
```

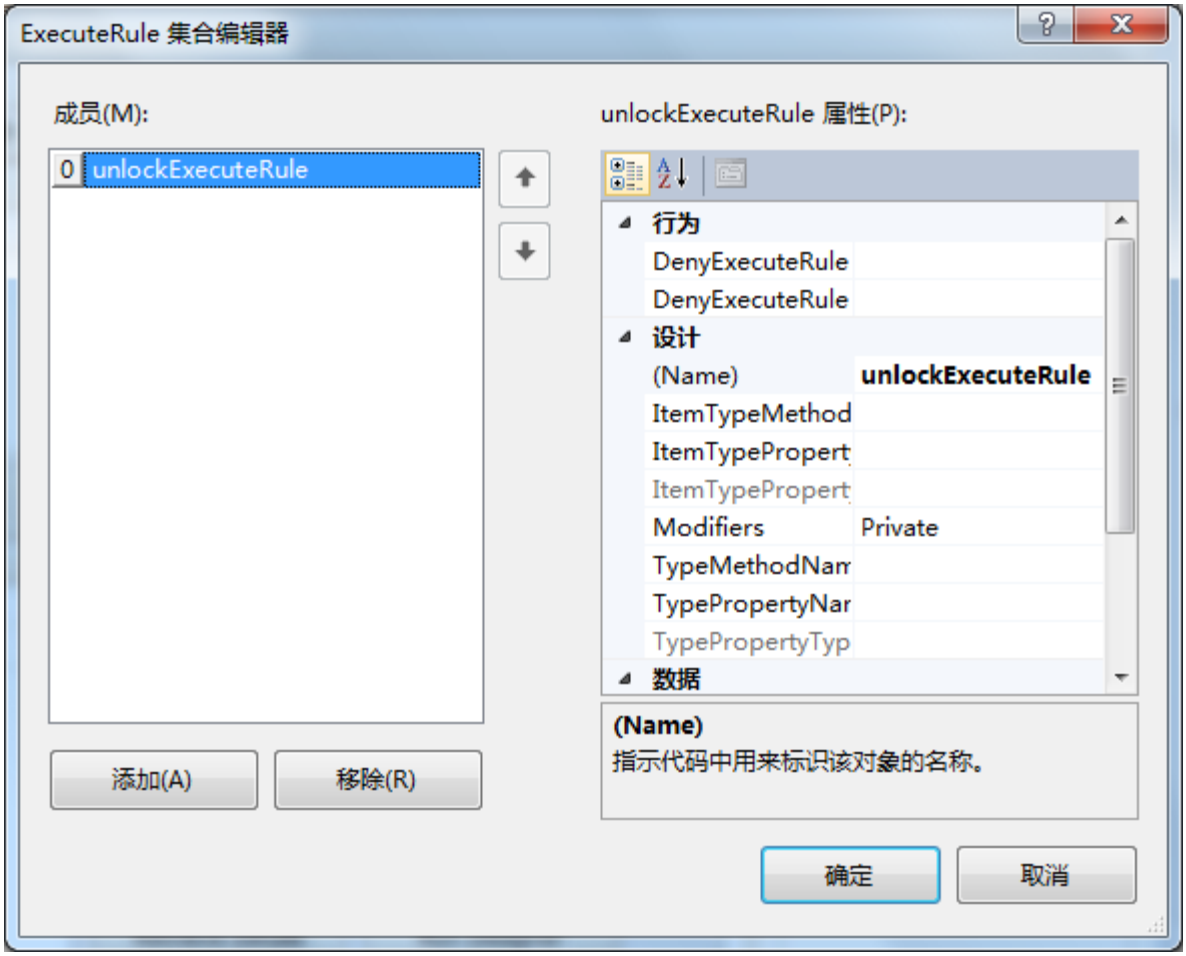
```
        CanExecuteMethod(UnlockMethod, true);

    }
```

如果仅仅设计到这一步，而不在界面设计上做绑定配置的话，应用系统就会以抛出警示对话框的方式告诉用户没有权限或不符合业务规则，这样友好性就很差。

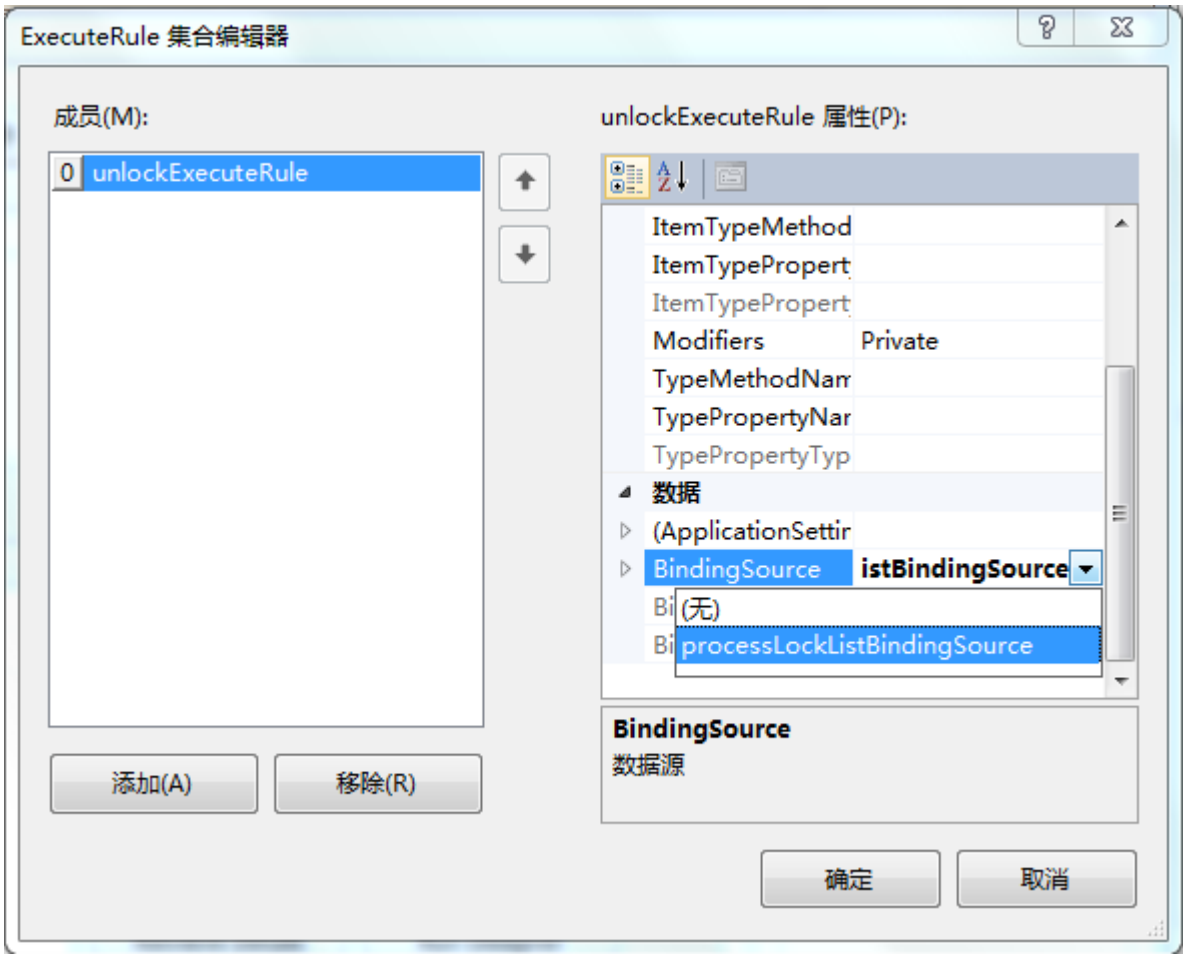
要将业务对象过程的执行授权规则绑定到本组件上，需要配置组件扩展属性 ApplyRuleItems，在 ExecuteRule 集合编辑器上点击“添加”按钮新建一个 ExecuteRule：



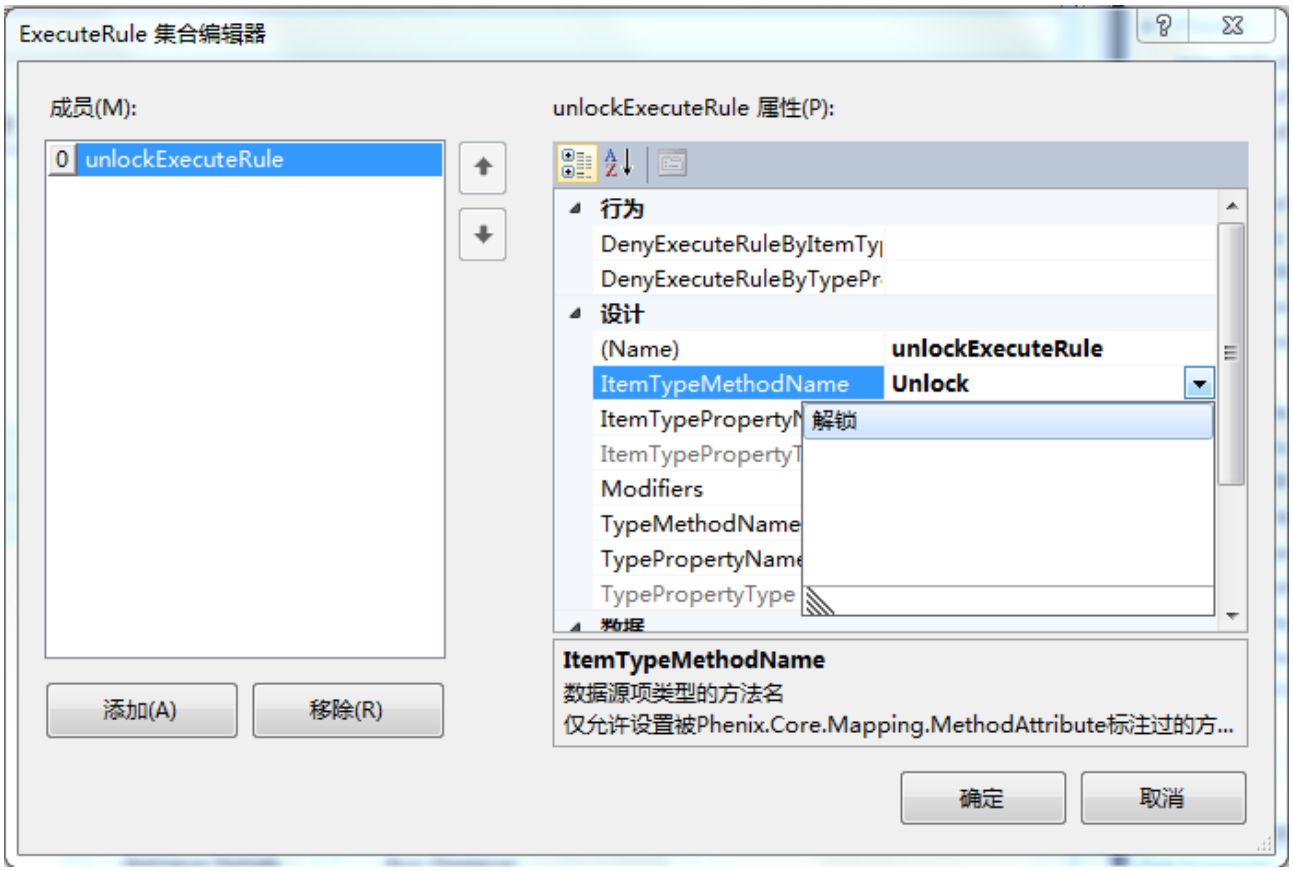
改一下名称使得本 ExecuteRule 含义清晰：



设置 BindingSource：



设置 ItemTypeMethodName 内容，下拉选择业务类上被标记过 Phenix.Core.Mapping.MethodAttribute(此标记会在 Phenix.Services.Host.exe 注册业务组件时告诉 Phenix 平台这被标注的函数将受到权限管控）的方法，完成业务对象过程的执行授权规则的绑定：



我们可以检查一下 ProcessLockManageForm.Designer 中自动生成的源代码：

```
//
// unlockExecuteRule
//
this.unlockExecuteRule.BindingSource = this.processLockListBindingSource;
this.unlockExecuteRule.ItemTypeMethodName = "Unlock";
```

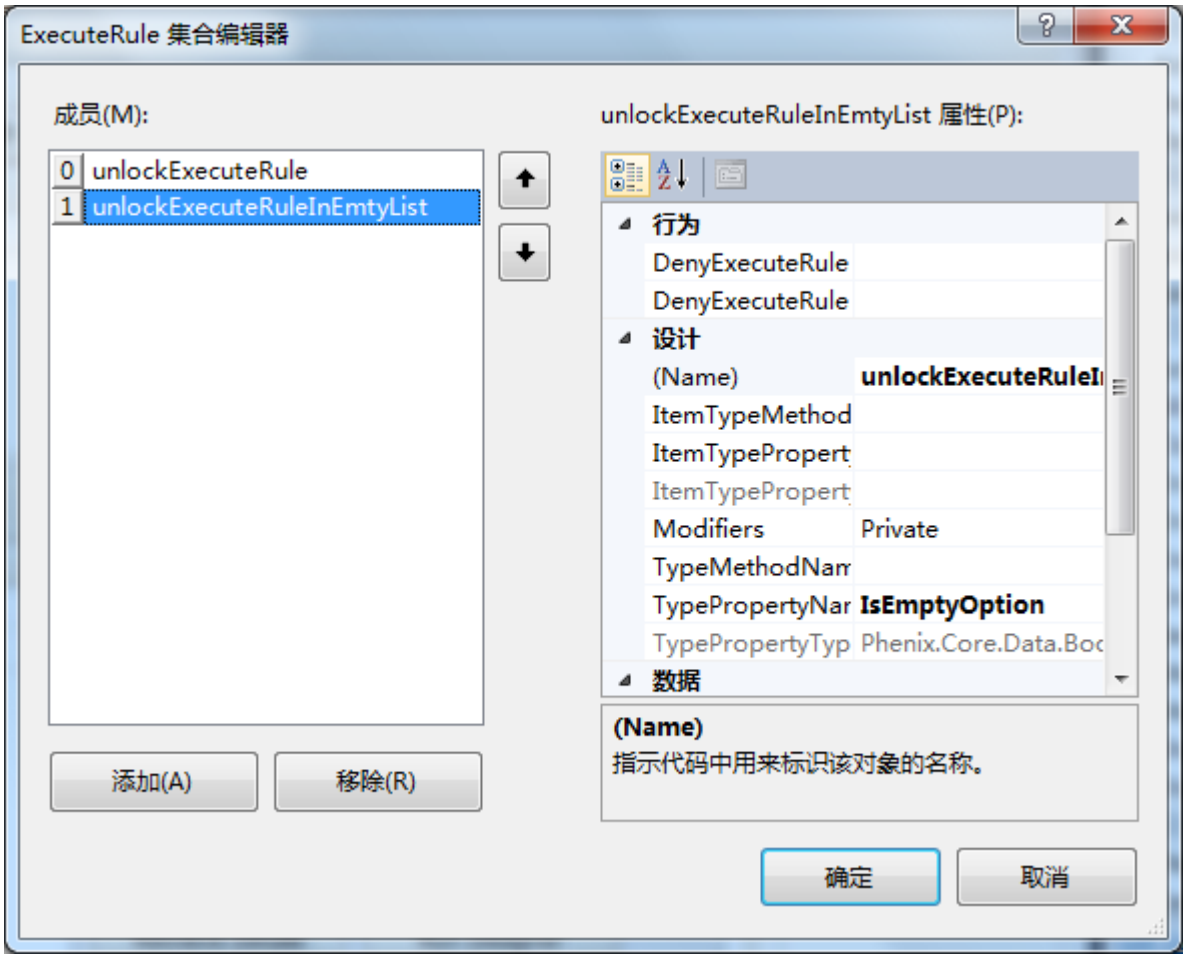
ExecuteRule 集合编辑器上可以添加多个 ExecuteRule，只要有一项返回结果是不允许执行，则这个组件就不允许被操作。

7.3.2 配置简单的执行授权规则

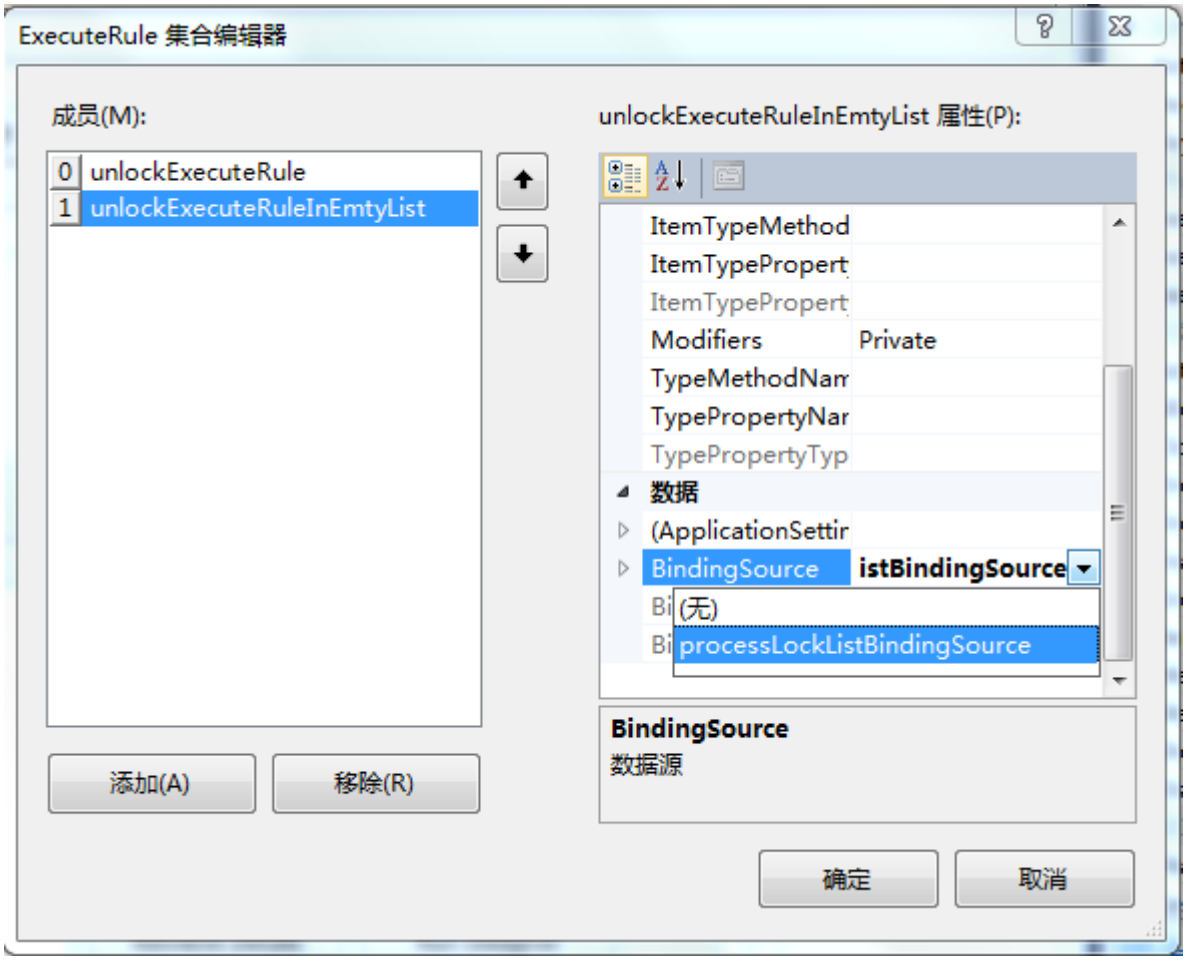
通过 ExecuteRule 集合编辑器也可以直接配置出简单的执行授权规则（注意下面讲述的是纯粹的界面控制层逻辑，因为使用本功能往往一不留神会打破系统逻辑分层原则），而无需编写一行代码。

我们仍以 ProcessLock 管理界面“解锁”功能按钮的设计为例，业务场景是:当 ProcessLock 清单为空时，不允许点击“解锁”功能按钮。

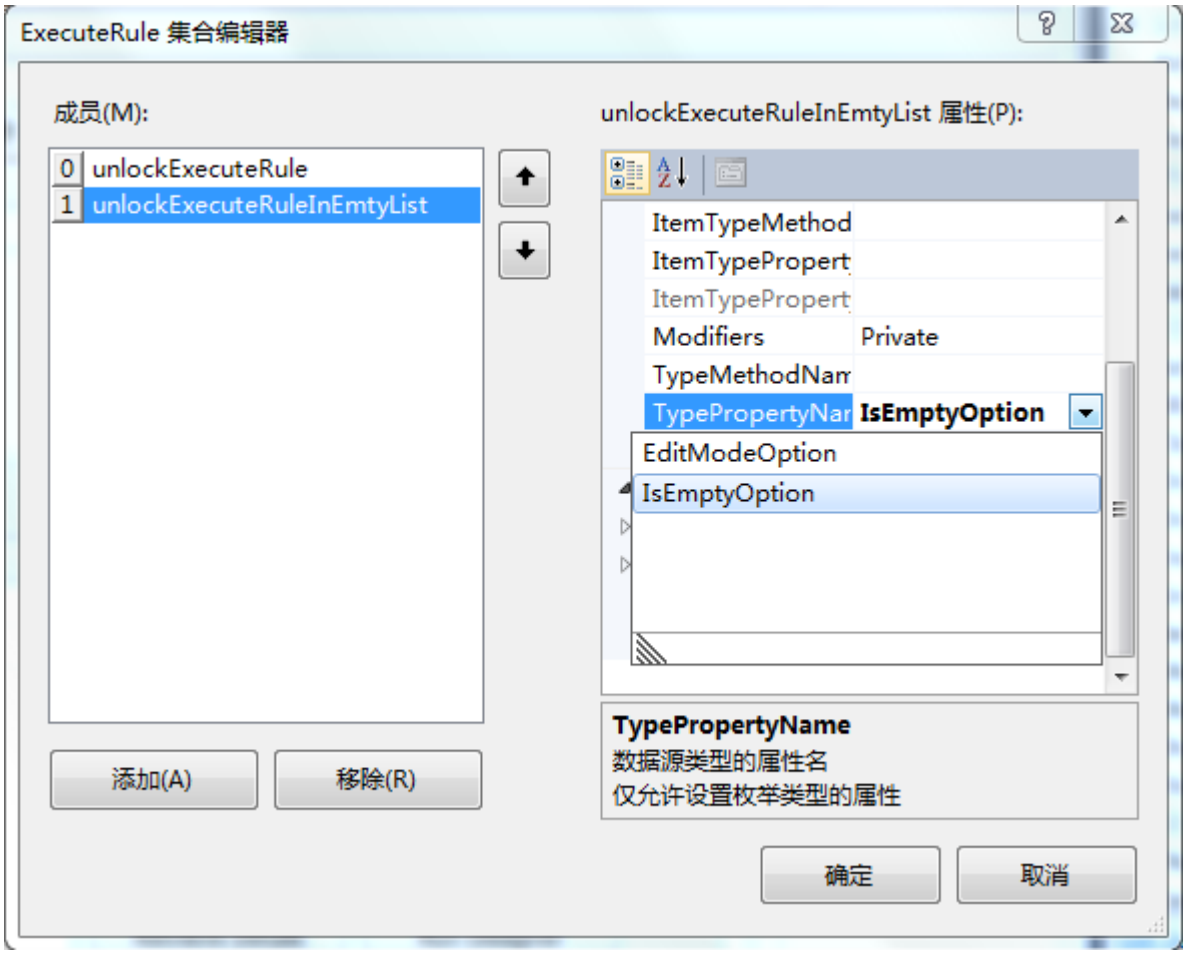
首先，我们在“解锁”功能按钮扩展属性 ApplyRuleItems 的 ExecuteRule 集合编辑器里添加一项并将其名称改得有意义：



设置 BindingSource：



设置 TypePropertyName 内容，为 IsEmptyOption：



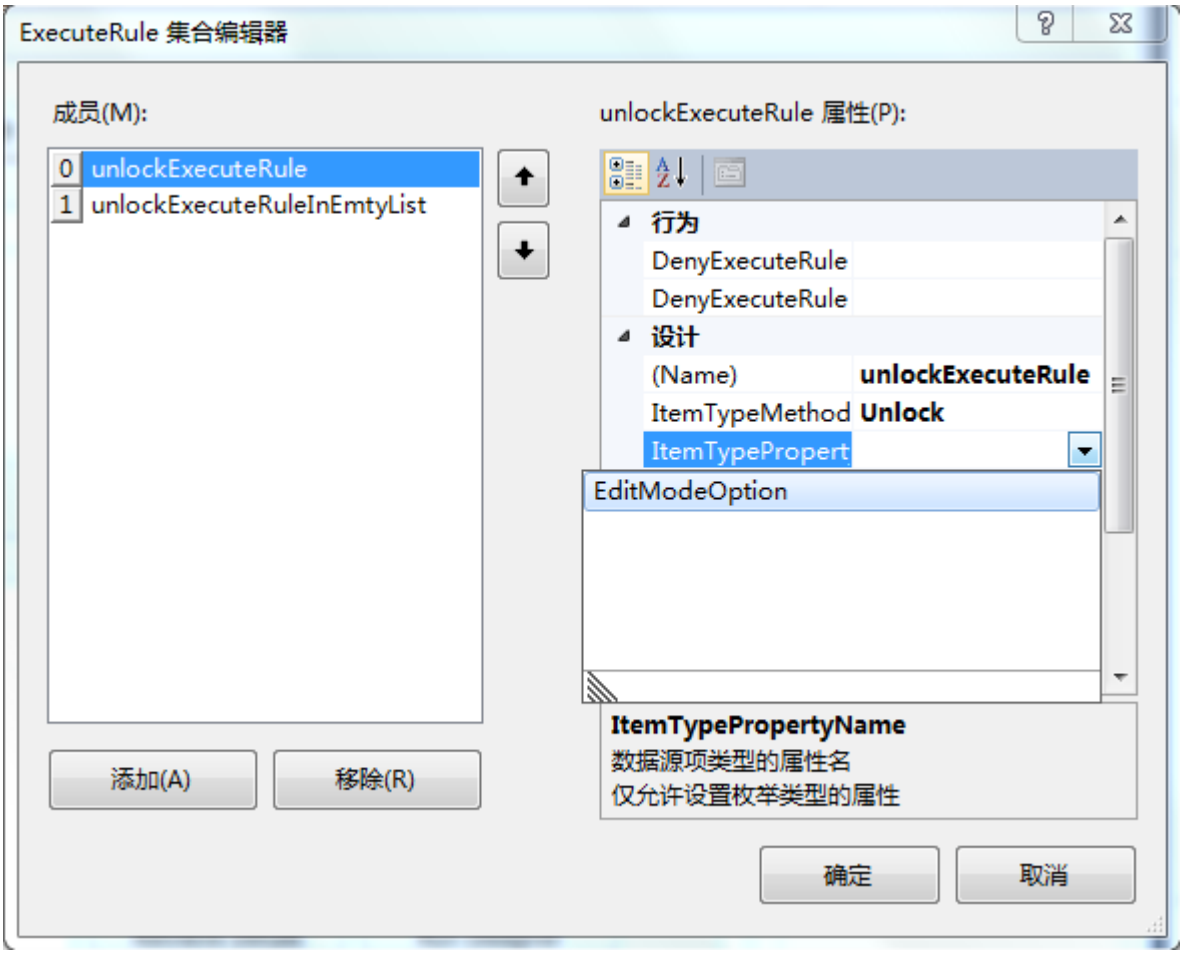
此处的 IsEmptyOption 是业务集合基类（Phenix.Business.BusinessListBase<T, TBusiness>）里供使用的两个枚举属性之一：

业务集合基类枚举属性	解释
IsEmptyOption	是否空集合
EditModeOption	在编辑状态

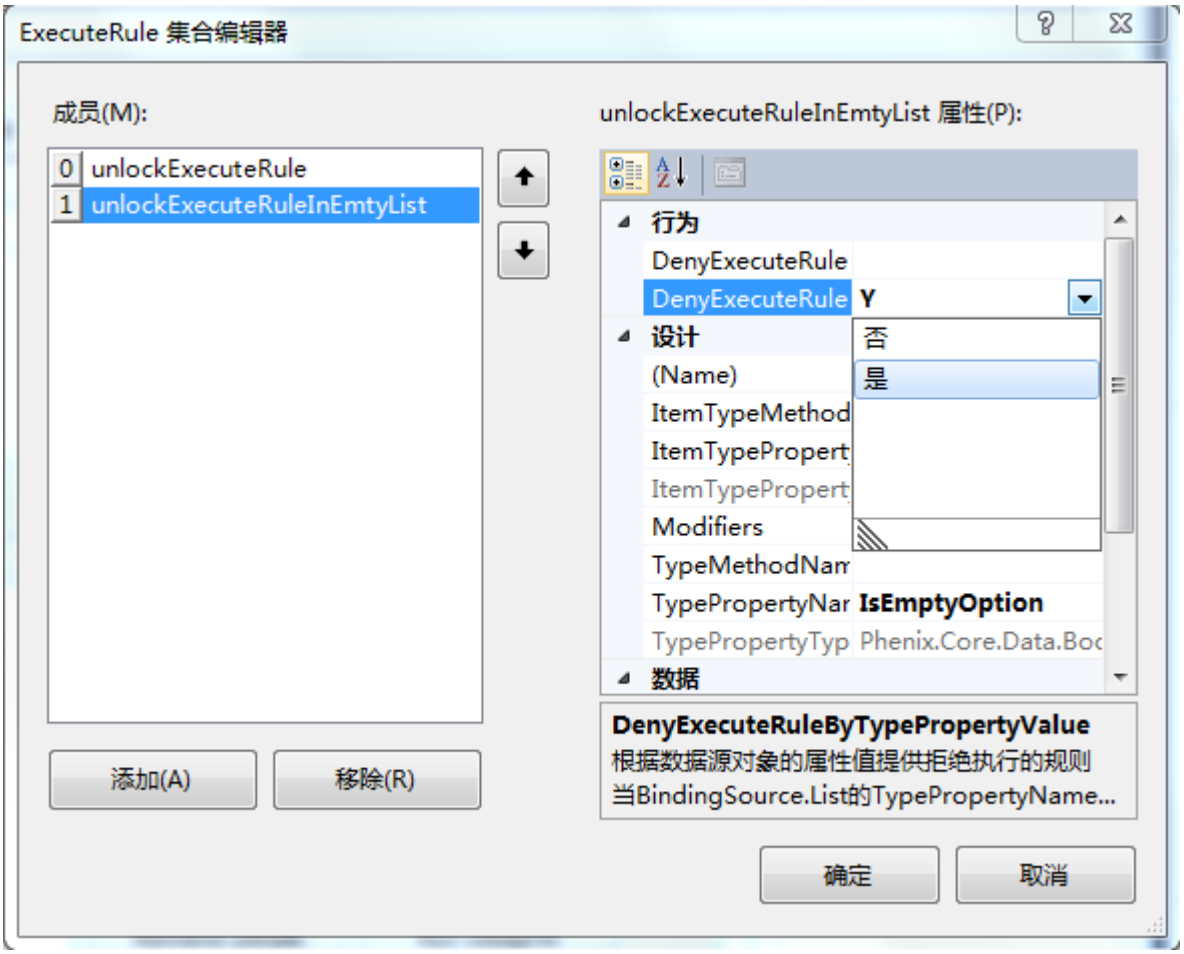
业务基类（Phenix.Business.BusinessBase<TBusiness>）也有供使用的枚举属性：

业务基类枚举属性	解释
EditModeOption	在编辑状态

可以用来设置 ItemTypePropertyName 的内容：



最后设置 DenyExecuteRuleByTypePropertyValue 的值：



我们可以检查一下 ProcessLockManageForm.Designer 中自动生成的源代码：

```
        //
        // unlockExecuteRuleInEmtyList
        //
        this.unlockExecuteRuleInEmtyList.BindingSource = this.processLockListBindingSource;
        this.unlockExecuteRuleInEmtyList.DenyExecuteRuleByTypePropertyValue =
Phenix.Core.Data.BooleanOption.Y;
        this.unlockExecuteRuleInEmtyList.TypePropertyName = "IsEmptyOption";
```

TypePropertyName、ItemTypePropertyName 绑定的是业务集合类、业务类里面的枚举属性，对于 bool 型属性，Phenix 有对应的 Phenix.Core.Data.BooleanOption 枚举类可供转换，用法可参考以下代码：

```
    /// <summary>
    /// 在编辑状态
    /// </summary>
    [System.ComponentModel.Browsable(false)]
    [System.ComponentModel.DataAnnotations.Display(AutoGenerateField = false)]
    public bool EditMode
```

```
{
  get { return base.EditLevel > 0; }
}
/// <summary>
/// 在编辑状态
/// </summary>
[System.ComponentModel.Browsable(false)]
[System.ComponentModel.DataAnnotations.Display(AutoGenerateField = false)]
public BooleanOption EditModeOption
{
  get { return EditMode ? BooleanOption.Y : BooleanOption.N; }
}
```

另外需注意的是，与 TypePropertyName、ItemTypePropertyName 对应的是它们的判断条件值 DenyExecuteRuleByTypePropertyValue、DenyExecuteRuleByItemTypePropertyValue 的定义，其含义是当定义的属性值等于判断条件值时，则将这 ExecuteRule 所属的组件调整为不可操作（变灰）。

7.4 操作

ExecuteAuthorization 组件与 Phenix.Windows.BarManager 组件会一起联动，当这两个组件在一个界面上时，Phenix.Windows.BarManager 组件会检测到 ExecuteAuthorization 组件，并根据以下界面中所操作数据的变化而自动控制 ExecuteAuthorization 组件：

- 界面初始化后；
- 切换了当前操作的 BindingSource；
- 当前操作的 BindingSource 的操作状态（浏览、编辑）发生了切换；
- 当前操作的 BindingSource 的 DataSource、List、Item、Position 发生了改变；

ExecuteAuthorization 组件自身也具备自动管控功能，比如：

- 界面初始化、显示之后会重置一遍所控制的组件；
- 继承自 Control 的组件（也就是控件）Enabled 属性会被锁定；

以下是 ExecuteAuthorization 组件的公共函数，可根据界面控制的需要来重置组件的执行授权：

```
/// <summary>
/// 重置组件的执行授权
/// </summary>
public void ResetComponentAuthorizationRules()

/// <summary>
/// 重置组件的执行授权
/// </summary>
```

```
/// <param name="component">组件</param>
public void ResetComponentAuthorizationRules(Component component)

/// <summary>
/// 是否拒绝执行
/// </summary>
public bool DenyExecute(Component component)
```

7.5 事件

ExecuteAuthorization 组件在重置组件的执行授权时，会触发以下两个事件：

事件	说明	参数
Resetting	正在 ResetComponentAuthorizationRules	ExecuteAuthorizationEventArgs
Reseted	已完成 ResetComponentAuthorizationRules	ExecuteAuthorizationEventArgs

7.6 表现

一个被 ExecuteAuthorization 控制的组件，允许设置多个执行授权规则项（通过 ExecuteRule 集合编辑器实现），只要有一项被判断出是被拒绝执行的，这个组件就要被禁止操作。

Phenix.Windows.BarManager 组件上的标准按钮执行的是标准的界面操作逻辑，并直接受到权限管控，所以一般情况下无需绑定 ExecuteAuthorization 组件，除非业务场景需要在界面逻辑上添加额外的执行授权规则，这只要对这个标准按钮的扩展属性 ApplyAuthorization 设置为 true，然后添加你的 ExecuteRule 即可。