IDynamicFormView接口使用示例代码

修订记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. No** | **日期** | **编制\修订** | **校对** | **批准** | **修改的章节号** |
| V1.0 | 20150331 | 张日照  刘泽铭  吴亮  张晋博 | 丁振华 | 赖碧云 | 初始版本 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

问题和意见

如果你对文档有任何意见、问题或想法，或者你的问题未在此文档中找到答案，请通过电子邮件联系我们

jinbo\_zhang@kingdee.com

# IDynamicFormView接口概要说明

IDynamicFormView接口，MVC架构中的视图层实现接口，负责界面展示等相关操作，对动态表单的视图进行管理，广泛应用于表单插件和列表插件。

表单插件和列表插件访问View属性取得IDynamicFormView的接口实例。

# 公共属性

## BillBusinessInfo

当前页面管理的来源单据逻辑元数据。

### 语法

**属性定义**：

|  |  |
| --- | --- |
| C# |  |
| BusinessInfo BillBusinessInfo { get; } | |

**备注**：

与BusinessInfo的区别是在列表（或在简单报表和移动单据），BillBusinessInfo表示来源单据的逻辑元数据，如列表有自身的逻辑元数据，界面上包含了一个表格，但表格中的字段又是根据来源单据的字段动态生成的，所以平台通用列表都会去访问来源单据的业务信息，这里公开来源单据信息以便用户在列表插件（或系统简单报表插件）中使用。

就目前来说，一般情况下除了在列表（或系统报表的简单报表和移动单据）视图中，它表示来源单据的逻辑元数据外，其它情况下它BusinessInfo都是表示当前对象的逻辑元数据。

### 常用属性

|  |  |
| --- | --- |
| **名称** | **描述** |
| **Elements** | 返回单据上的全部元素，包括全部字段、实体 |
| **Entrys** | 返回单据上的全部实体 |
| **MainOrgField** | 返回单据的主业务组织字段 |

### 常用方法

|  |  |
| --- | --- |
| **名称** | **描述** |
| **GetBillNoField**() | 获取单据编号字段，如果是基础资料，则返回基础资料编码 |
| **GetBillTypeField**() | 获取单据类型字段 |
| **GetBosFields**() | 获取单据全部字段 |
| **GetDynamicObjectType**([**bool** *forceRebuild* = false]) | 获取单据的ORM数据模型对象，  参数决定是否重新构建ORM数据对象模型。单据元数据改变后，需要强制要求重新构建ORM数据对象模型。 |
| **GetEntity**(**string** *key*) | 获取单据的实体元数据 |
| **GetField**(**string** *key*) | 获取单据字段元数据 |
| **GetFieldList**() | 获取单据全部字段元数据列表 |
| **GetForm**() | 获取单据整体属性元数据 |
| **GetRealField**(**string** *key*) | 获取代理字段的真实字段元数据 |
| **GetRedBlueField**() | 获取红蓝单标志字段 |
| **GetSubBusinessInfo**(**System.Collections.Generic.List<string>** *sectionKeys*) | 根据所选字段，产生节选单据元数据 |

### 案例 单据列表默认按指定字段进行排序

**代码来源**

供应链单据列表公共插件

Kingdee.K3.SCM.Business.PlugIn.SCMBillList,

Kingdee.K3.SCM.Business.PlugIn

**需求背景**

供应链所有列表在展示时，希望按编码与创建日期降序排列

**实现方案**

供应链通用插件中，在列表准备过滤条件阶段（PrepareFilterParameter事件），通过BillBusinessInfo获取编号与创建日期的字段元数据信息，通过字段元数据信息可以得到这两个字段对应的数据库字段名称。使用这些名称，修改事件的排序子句，后面加载列表数据的时候BOS平台会根据这个排序子句对列表进行排序。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.ComponentModel;  using Kingdee.BOS.Core.List.PlugIn;  using Kingdee.BOS.Core.DynamicForm;  using Kingdee.BOS.Core.List;  using Kingdee.BOS.Core.Metadata.FormElement;  using Kingdee.BOS.Core.Metadata.BarElement;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using Kingdee.BOS.Core.Metadata;  using Kingdee.K3.Core;  using Kingdee.K3.Core.SCM;  using Kingdee.BOS.Core.Bill;  using Kingdee.BOS.Core.Metadata.FieldElement;  using Kingdee.BOS.Core.Metadata.EntityElement;  using Kingdee.BOS.Core.CommonFilter;  namespace Kingdee.K3.SCM.Business.PlugIn  {  /// <summary>  /// 供应链单据模板列表公共插件  /// </summary>  [Description("供应链单据列表公共插件")]  public class SCMBillList : AbstractListPlugIn  {  public override void PrepareFilterParameter(FilterArgs e)  {  if (string.IsNullOrWhiteSpace(e.SortString))  {  Field crDateFld = this.View.BillBusinessInfo.GetFieldList().FirstOrDefault(  p => p is CreateDateField);  Field numberFld = this.View.BillBusinessInfo.GetFieldList().FirstOrDefault(  p => p is BillNoField);  //也要按分录顺序排序  StringBuilder entrySort = new StringBuilder();  foreach (FilterEntity ety in e.SelectedEntities)  {  if (ety.Selected && ety.EntityType == BOS.Core.Enums.BOSEnums.Enum\_EntityType.Entity ||  ety.Selected && ety.EntityType == BOS.Core.Enums.BOSEnums.Enum\_EntityType.SubEntity)  {  EntryEntity entryEntity = this.View.BillBusinessInfo.GetEntryEntity(ety.Key);  if (string.IsNullOrWhiteSpace(entryEntity.SeqFieldKey))  {  entrySort.AppendFormat(", {0}.{1} ASC ", entryEntity.TableAlias, entryEntity.EntryPkFieldName);  }  else  {  entrySort.AppendFormat(", {0}.{1} ASC ", entryEntity.TableAlias, entryEntity.SeqFieldKey);  }  }  }  if (crDateFld != null)  {  if (numberFld != null)  {  e.SortString = string.Format(" {0} DESC,{1} DESC {2} ", crDateFld.FieldName, numberFld.FieldName, entrySort.ToString());  }  else  {  e.SortString = string.Format(" {0} DESC {1} ", crDateFld.FieldName, entrySort.ToString());  }  }  }  }  }  } | |

## BusinessInfo

当前页面业务对象的逻辑元数据。

### 语法

**属性定义**：

|  |  |
| --- | --- |
| C# |  |
| BusinessInfo BusinessInfo { get; } | |

**备注**：

注意与BillBusinessInfo的区别和联系：

BusinessInfo属性必定为当前界面的元数据；而BillBusinessInfo，就目前来说，在列表或系统报表的简单报表和移动单据视图上，它表示来源单据的业务逻辑元数据。

更多区别参见BillBusinessInfo属性说明。

### 常用属性

参考[BillBusinessInfo](#_常用属性)

### 常用方法

参考[BillBusinessInfo](#_常用方法)

### 案例 已经下推的数据行不允许删除

**代码来源**

销售订单单据维护界面插件

Kingdee.K3.SCM.Sal.Business.PlugIn.SaleOrderEdit,

Kingdee.K3.SCM.Sal.Business.PlugIn

**需求背景**

销售订单界面，删除行数据之前要获取单据体行信息，判断行数据是否可以删除

**实现方案**

插件重写BeforeDeleteRow事件，从View中拿BusinessInfo对象，从中获取需要的单据体元数据信息,以此为参数，从模型中获取到当前行数据包，然后判断当前行数据是否能删除

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.BusinessFlow.ServiceArgs;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using Kingdee.BOS.Core.Metadata.EntityElement;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS.Util;  using System;  namespace Kingdee.K3.SCM.Sal.Business.PlugIn  {  public class SaleOrderEdit: AbstractBillPlugIn  {  public override void BeforeDeleteRow(BeforeDeleteRowEventArgs e)  {  if (e.EntityKey.EqualsIgnoreCase("FSaleOrderEntry"))  {  #region 直接订单变更时，删除明细行，只允许删除行状态为非关闭、非冻结、非终止、未关联下游数据的分录！  // 获取单据状态  string sDocumentStatus = Convert.ToString(this.View.Model.GetValue("FDocumentStatus"));  //para\_SOChangeType，变更类型 A:启用订单变更单 B：订单直接变更  // 变更类型=订单直接变更，单据状态=已审核时  if (para\_SOChangeType == "B" && sDocumentStatus == "C")  {  int currentRowIndex = e.Row;  bool bIsFreeze = Convert.ToString(this.View.Model.GetValue("FMrpFreezeStatus", currentRowIndex)).Equals("B");  bool bIsTerminate = Convert.ToString(this.View.Model.GetValue("FMrpTerminateStatus", currentRowIndex)).Equals("B");  bool bIsClose = Convert.ToString(this.View.Model.GetValue("FMrpCloseStatus", currentRowIndex)).Equals("B");  Entity entity = this.View.BusinessInfo.GetEntity("FSaleOrderEntry");  DynamicObjectCollection entityData = this.View.Model.GetEntityDataObject(entity);  DynamicObject dynamicObjectRow = entityData[currentRowIndex];  // 判断当前行是否已经下推其他单据  IsPushArgs isPushArgs = new IsPushArgs(this.View.BusinessInfo, "FSaleOrderEntry", dynamicObjectRow);  bool bRowIsPush = Kingdee.BOS.ServiceHelper.BusinessFlowDataServiceHelper.IsPush(this.Context, isPushArgs);  if (bIsClose || bIsFreeze || bIsTerminate || bRowIsPush)  {  e.Cancel = true;  this.View.ShowMessage("只允许删除行状态为非关闭、非冻结、非终止、未关联下游数据的分录！");  }  }  #endregion  }  }  }  } | |

## ClientType

访问服务器的客户端类型，如Silverlight、WPF、HTML5、移动等。

### 语法

**属性定义**：

|  |  |
| --- | --- |
| C# |  |
| ClientType ClientType { get; } | |

### 案例

暂缺

## Context

登录系统的上下文信息，包括当前登录的用户、客户端类型、连接的数据中心，当前组织等相关信息。

### 语法

**属性定义**：

|  |  |
| --- | --- |
| C# |  |
| Context Context { get; } | |

**备注**：

在插件中，这个属性对象使用比较多，也很重要。而调用服务接口的时候，此属性通常通常必不可少。

**关键字：**

上下文、数据库连接、当前组织、当前用户、语言环境，客户端类型

### 常用属性

|  |  |
| --- | --- |
| **名称** | **描述** |
| **ClientInfo** | 客户端信息 |
| **ClientType** | 客户端类型，在ClientInfo里面也有一份 |
| **CurrentOrganizationInfo** | 当前组织信息 |
| **CurrentUserTimeZone** | 当前用户时区 |
| **DatabaseType** | 数据库类型 |
| **IsMultiOrg** | 是否是多组织数据中心 |
| **UserId** | 当前用户Id |
| **UserName** | 当前用户名称 |

### 常用方法

|  |  |
| --- | --- |
| **名称** | **描述** |
| **Clone()** | 创建一个浅表副本 |

### 案例 根据数据中心的多组织属性，隐藏/显示菜单

**代码来源**

客户物料对照表——单据维护插件

Kingdee.K3.SCM.Sal.Business.PlugIn.CustMatMapping

Kingdee.K3.SCM.Sal.Business.PlugIn

**需求背景**

加载单据数据之后，如果是单组织数据中心，则需要隐藏分发下拉菜单；如果是多组织，则需要显示该菜单

**实现方案**

重写AfterBindData事件，判断数据中心的多组织类型，隐藏或显示菜单

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.Bill.PlugIn;  using System;  namespace Kingdee.K3.SCM.Sal.Business.PlugIn  {  public class CustMatMapping : AbstractBillPlugIn  {  public override void AfterBindData(EventArgs e)  {  base.AfterBindData(e);  bool bIsMultiOrg = this.View.Context.IsMultiOrg;  if (bIsMultiOrg == true)  {  this.View.GetMainBarItem("tbSplitIssue").Visible = true;  }  else  {  this.View.GetMainBarItem("tbSplitIssue").Visible = false;  }  }  }  } | |

## LayoutInfo

当前页面布局元数据，包括各个控件的大小、位置、标题等外观信息。

### 语法

**属性定义**：

|  |  |
| --- | --- |
| C# |  |
| LayoutInfo LayoutInfo { get; } | |

### 案例

暂缺

## Model

动态表单数据模型编程接口，用于视图操作数据的模型，界面数据的增删改查都需要用到。

### 语法

**属性定义**：

|  |  |
| --- | --- |
| C# |  |
| IDynamicFormModel Model { get; } | |

**备注**：

IDynamicFormModel接口介绍，请参阅《IDynamicFormModel接口调用示例代码》；

### 常用属性

参考IDynamicFormModel接口介绍

### 常用方法

参考IDynamicFormModel接口介绍

### 案例 获取字段A的值，用这个值给字段B赋值

**代码来源**

采购订单表单插件

Kingdee.K3.SCM.Purchase.Business.PlugIn.PurchaseOrderEdit

Kingdee.K3.SCM.Purchase.Business.PlugIn

**需求背景**

采购订单明细单据体，复制新行之后，需要将源分录行的数量值（FQty）带到新行的剩余收料数量（FRemainReceiveQty）、源分录行的基本单位数量值（FBaseUnitQty）带到新行的剩余入库数量（FRemainStockINQty）

**实现方案**

插件重写AfterCopyRow事件，获取源分录行的字段值，赋值给新行对应的字段

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.Bill.PlugIn;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using System;  namespace Kingdee.K3.SCM.Purchase.Business.PlugIn  {  public class PurchaseOrderEdit: AbstractBillPlugIn  {  public override void AfterCopyRow(AfterCopyRowEventArgs e)  {  decimal sRowQty = Convert.ToDecimal(this.View.Model.GetValue("FQty", e.Row));  decimal sRowBaseQty = Convert.ToDecimal(this.View.Model.GetValue("FBaseUnitQty", e.Row));  if (e.EntityKey.ToUpperInvariant() == "FPOORDERENTRY")  {  //剩余收料数量 = 源行订单数量  this.View.Model.SetValue("FRemainReceiveQty", sRowQty, e.NewRow);  //剩余入库数量 = 源行订单数量  this.View.Model.SetValue("FRemainStockINQty", sRowQty, e.NewRow);  }  }  }  } | |

## OpenParameter

界面被加载时,传入的参数。

### 语法

**属性定义：**

|  |  |
| --- | --- |
| C# |  |
| DynamicFormOpenParameter OpenParameter { get; } | |

**备 注**：

打开当前模型的相关参数，与请求加载模型时，要求的相关模型参数类似；

另外，对于不同的界面类型，如动态表单界面、单据维护界面、列表界面，本属性的类型不同；

### 常用属性

|  |  |
| --- | --- |
| **名称** | **描述** |
| **FormMetaData** | 业务对象元数据模型 |
| **PageId** | 本页面PageID |
| **ParentPageId** | 父页面PageID |

### 常用方法

|  |  |
| --- | --- |
| **名称** | **描述** |
| **GetCustomParameter (string key)**  **GetCustomParameter(string key, bool remove)** | 获取定制参数 |
| **GetCustomParameters()** | 获得所有自定义参数 |
| **SetCustomParameter (String key, object value)** | 添加定制参数 |
| **SetDynamicFormOpenParams** | 给动态表单窗体赋值参数 |

### 案例 初始化时获取个性化参数

**代码来源**

Kingdee.K3.SCM.Business.DynamicForm.PlugIn.AbstractSCMBillEdit,

Kingdee.K3.SCM.Business.DynamicForm.PlugIn

**需求背景**

供应链所有单据插件，初始化首先获取个性化参数

**实现方案**

基类插件在OnInitialize时，通过获取OpenParameter，得到打开单据时传递的各参数（标准参数与个性化参数）

**插件示例**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using Kingdee.BOS.Core;  using Kingdee.BOS.Core.Const;  using Kingdee.BOS.Core.Bill;  using Kingdee.BOS.Core.DynamicForm.PlugIn;  namespace Kingdee.K3.SCM.Business.DynamicForm.PlugIn  {  /// <summary>  /// 供应链单据插件基类 包含一些公共的地方 如果需要可继承自他 注意在子类要执行基类的方法要在事件上加上 base.XXX()  /// </summary>  public abstract class AbstractSCMBillEdit : AbstractDynamicFormPlugIn  {  //状态为新增有效 是否从上游单据下推新增  protected bool IsNewFromSrcBill = false;  public override void OnInitialize(InitializeEventArgs e)  {  BillOpenParameter para = (BillOpenParameter)this.View.OpenParameter;  if (para != null)  {  IsNewFromSrcBill = (para.GetCustomParameter(FormConst.ConvParamKey\_OneResultSessionKey) != null);  }  }  }  } | |

## PageId

页面唯一标识，用来区分同时打开的不同页面。

### 语法

**属性定义**：

|  |  |
| --- | --- |
| C# |  |
| string PageId { get; } | |

**备 注**：

浏览器中打开的每个页面，在K/3 Cloud应用服务器，都有以PageId标识与隔离的Session片段，缓存页面的数据；

可以利用[IDynamicFormView.GetView(otherPageId)](#_GetView)方法，传入其他页面的PageId，就能够获取其他页面的IDynamicFormView接口实例，从而控制其他页面；

### 案例

暂缺

## ParentFormView

调用本视图的父窗体视图模型。

### 语法

**属性定义**：

|  |  |
| --- | --- |
| C# |  |
| IDynamicFormView ParentFormView { get; set; } | |

**备 注**：

当前被打开页面的父页面：

如A页面上，点击按钮，弹出B页面，A页面是B页面的ParentFormView；

特别关注：ParentFormView可能为null，使用前要增加null值判断；

### 案例 锁定父页面字段

**代码来源**

Kingdee.K3.SCM.Credit.Business.PlugIn.CreditSysParameterEdit

Kingdee.K3.SCM.Credit.Business.PlugIn

**需求背景**

加载当前页面时，根据父页面上下文信息，初始化本单据全局变量；同时锁定父页面某字段

**实现方案**

插件重写OnInitialize方法，通过当前页面View的ParentFormView属性，获取到父页面View对象，访问父页面View对象的Context属性中IsMultiOrg属性，初始化本单据bIsMultiOrg全局对象；通过父页面View提供的GetControl方法，得到父页面的控件，对控件属性赋值，最后通过调用SendDynamicFormAction方法，把父页面作为参数传进去，将父页面的控件控制指令发送到前端

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using Kingdee.BOS.Core.DynamicForm.PlugIn;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.K3.Core.SCM;  using Kingdee.K3.Core.SCM.CRE;  namespace Kingdee.K3.SCM.Credit.Business.PlugIn  {  public class CreditSysParameterEdit : AbstractDynamicFormPlugIn  {  long \_lOrgId = 0;  bool bIsMultiOrg = false;  public override void OnInitialize(InitializeEventArgs e)  {  if (this.View.ParentFormView == null)  {  return;  }  object oOrgId = e.Paramter.GetCustomParameter("OrgId");  bIsMultiOrg = this.View.ParentFormView.Context.IsMultiOrg;  if (oOrgId != null)  {  this.\_lOrgId = Convert.ToInt64(oOrgId);  }  else  {  this.\_lOrgId = Convert.ToInt64(this.Context.CurrentOrganizationInfo.ID);  }  // 锁定父窗体的组织名称字段  ***this.View.ParentFormView.GetControl("FOrgId").Enabled = false;***  this.View.SendDynamicFormAction(this.View.ParentFormView);  }  } | |

## RuleContainer

当前视图的实体服务规则容器。

### 语法

**属性定义**：

|  |  |
| --- | --- |
| C# |  |
| BOSRuleContainer RuleContainer { get; } | |

**备 注**：

本属性为内部管理类，插件请勿调用。

### 案例

暂缺

## Session

页面数据交换区。

### 语法

**属性定义：**

|  |  |
| --- | --- |
| C# |  |
| Dictionary<string, object> Session { get; set; } | |

**备 注**：

跨页面传输复杂对象，或用于与其它视图用于数据交换的数据缓存区。

典型应用场景：A页面上，点击按钮，弹出B页面，在B页面上点击某个操作（如：关闭），将B页面的信息传递回A页面，其实质是将B页面的信息，通过访问其ParentFormView，把数据存放在A页面的数据交换区，以便在A页面，可以访问到子页面回传的数据

### 案例 与父页面进行数据交换

**代码来源**

Kingdee.K3.SCM.Credit.Business.PlugIn.CreditReCalResultEdit,

Kingdee.K3.SCM.Credit.Business.PlugIn

**需求背景**

供应链的信用重算结果页面，需要获取父页面的数据，以便初始化本页面全局变量

**实现方案**

重写OnInitialize事件，通过访问ParentFormView的Session对象，获取父页面保存在缓存区的数据，当前页面使用完后，清除父页面的这个缓存对象数据

**示例代码：**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using Kingdee.BOS.Core.DynamicForm.PlugIn;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using Kingdee.BOS.Cache;  using Kingdee.BOS.Orm.Metadata.DataEntity;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.ServiceHelper;  using Kingdee.BOS.Core;  using Kingdee.BOS.Core.DynamicForm;  using Kingdee.BOS.Core.DynamicForm.PlugIn.ControlModel;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS.Core.Metadata.EntityElement;  using Kingdee.BOS.Core.Bill;  using Kingdee.BOS.Util;  using Kingdee.BOS.Core.Metadata.FieldElement;  using Kingdee.BOS.Resource;  using Kingdee.K3.SCM.Common.BusinessEntity.Credit;  using Kingdee.BOS.Core.NetworkCtrl;  using Kingdee.BOS.Core.Permission;  using Kingdee.K3.Core.SCM.CRE;  using Kingdee.BOS.Contracts;  using System.ComponentModel;  namespace Kingdee.K3.SCM.Credit.Business.PlugIn  {  [Description("信用重算结果")]  public class CreditReCalResultEdit : AbstractDynamicFormPlugIn  {  private ReCalResult \_result; // 重算结果  private string \_recalType;  public override void OnInitialize(InitializeEventArgs e)  {  if (this.View.ParentFormView == null)  {  return;  }  if (e.Paramter.GetCustomParameter("SessionKey").IsNullOrEmpty())  {  return;  }  string sessionKey = e.Paramter.GetCustomParameter("SessionKey").ToString();  \_recalType = e.Paramter.GetCustomParameter("RecalType").ToString();  if (***this.View.ParentFormView.Session.ContainsKey(sessionKey)***)  {  \_result = this.View.ParentFormView.Session[sessionKey] as ReCalResult;  ***this.View.ParentFormView.Session.Remove(sessionKey);***  }  string resultTitle = \_recalType == "1" ? "信用初始化成功！", "信用重算成功！";  Control title = this.View.GetControl("FResultTitle");  title.Text = resultTitle;  }  } | |

## StyleManager

当前视图的控件样式管理器；

### 语法

**属性定义**：

|  |  |
| --- | --- |
| C# |  |
| IStyleManager StyleManager { get; } | |

**备 注**：

当前视图关联的控件样式管理器，主要管理控件的可用性（Enable）、可见性（Visible）、数量字段精度设置。

插件获取界面控件后，设置的控件状态，均存储与此属性对象；

本属性为内部管理类，插件请勿调用。

### 案例

暂缺

## UserParameterKey

用户参数的唯一标识。

### 语法

**属性定义**：

|  |  |
| --- | --- |
| C# |  |
| string UserParameterKey { get; } | |

**备注**：

用户参数的唯一标识。如加载的是单据，返回单据的模型ID，如加载的是列表，返回列表对应模型的ID+特殊字符串（”\_list”）

### 案例

暂缺

# 公共方法

## AddAction

添加自定义客户端指令，可带回掉函数。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void AddAction(string actionName, object param); | |

**参数说明**

actionName:客户端方法名

param:参数对象，必须可序列化为Json字符串

**返回类型**

无返回值

**关键字**

客户端指令

**重载方法**

|  |  |
| --- | --- |
| C# |  |
| void AddAction(string actionName, object param, Action<object> callback); | |

**参数说明**

actionName:客户端方法名

param:参数对象，必须可序列化为Json字符串

callback：回调函数

**返回类型**

无返回值

**关键字**

客户端指令

### 案例

暂缺

## BroadcastToOnlineUser

发送广播给用户。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void BroadcastToOnlineUser(JObject data,List<long> users); | |

**参数说明**

data：广播消息

users：广播用户，如果为空，表示所有在线用户

**返回类型**

无返回值

**关键字**

广播

### 案例

暂缺

## Close

请求关闭页面。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void Close(); | |

**参数说明**

无

**返回类型**

无返回值

**关键字**

关闭

### 案例 点击按钮，关闭页面

**代码来源**

财务插件：自动抵消-选择抵消类型

Kingdee.K3.FIN.CR.Business.PlugIn.AutoElimSelector,

Kingdee.K3.FIN.CR.Business.PlugIn

**需求背景**

点击界面的确定按钮，将数据返回给父页面，然后关闭页面

**实现方案**

重写按钮点击事件ButtonClick，打包好要返回给父页面的数据，然后使用ReturnToParentWindow方法返回数据给父页面，并关闭窗体

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| namespace Kingdee.K3.FIN.CR.Business.PlugIn  {  /// <summary>  /// 自动抵消选择  /// </summary>  [Description("自动抵消-选择抵消类型")]  public class AutoElimSelector : AbstractDynamicFormPlugIn  {  public override void ButtonClick(ButtonClickEventArgs e)  {  switch (e.Key.ToUpperInvariant())  {  case "FBTNOK":  ReturnData();  break;  }  base.ButtonClick(e);  }  private void ReturnData()  {  DynamicObject currency = this.View.Model.GetValue("FCurrencyID") as DynamicObject;  DynamicObject currencyUnit = this.View.Model.GetValue("FCurrencyUnitID") as DynamicObject;  List<int> selectedElimTypeIds = new List<int>();  DynamicObjectCollection objs = this.View.Model.GetEntityDataObject(this.View.BusinessInfo.GetEntryEntity("FEntity"));  foreach (DynamicObject obj in objs)  {  bool selected = Convert.ToBoolean(obj["Selected"]);  if (selected == true)  {  DynamicObject elimType = obj["ElimTypeID"] as DynamicObject;  selectedElimTypeIds.Add(Convert.ToInt32(elimType["ID"]));  }  }  if (selectedElimTypeIds.Count <= 0)  {  this.View.ShowErrMessage(Kingdee.BOS.Resource.ResManager.LoadKDString("未选择抵消类型", "003287000010479", Kingdee.BOS.Resource.SubSystemType.FIN));  return;  }  JSONObject ret = new JSONObject();  ret.Add("CurrencyID", currency["ID"]);  ret.Add("CurrencyUnitID", currencyUnit["ID"]);  ret.Add("CreateMode", this.Model.DataObject["CreateMode"]);  ret.Add("ElimTypeIds", selectedElimTypeIds);  this.View.ReturnToParentWindow(new FormResult(ret));  this.View.Close();  }  }  } | |

## FlexClose

请求关闭弹性域维护页面。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void FlexClose(); | |

**参数说明**

无

**返回类型**

无返回值

**关键字**

弹性域

**备注**

本方法专用于关闭弹性域维护页面，其他页面插件请使用Close()方法关闭页面；

### 案例

暂缺

## GetBarItem

获取工具栏、按钮、分录表格菜单项控件编程对象。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| BarItemControl GetBarItem(string parentKey, string barItemName);  T GetBarItem<T>(string parentKey, string barItemName) where T : BarItemControl; | |

**参数说明**

parentKey：工具栏、按钮、分录表格控件标识

barItemName：菜单项标识

**返回类型**

Kingdee.BOS.Core.DynamicForm.PlugIn.ControlModel. BarItemControl

**关键字**

设置菜单项控件的可用性 设置菜单项控件的可见性 设置菜单项控件的标题

**备注**

可以在获得工具栏按钮控件编程对象之后，设置其可用性、可见性、标题等

### 案例 设置菜单按钮不可用

**代码来源**

基础资料控制类型插件

Kingdee.BOS.Business.PlugIn.Organizations.BaseDataTypeEdit,

Kingdee.BOS.Business.PlugIn

**需求背景**

检查基础资料是否符合受控基础资料的配置要求，以此设置菜单的可用性

**实现方案**

对当前基础资料进行检查，是否受组织控制策略，并且检查是否有指定创建组织和使用组织，并且设置菜单按钮不可用

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Data;  using System.Collections.Generic;  using System.Text;  using Kingdee.BOS.Core.Bill.PlugIn;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.ServiceHelper;  using Kingdee.BOS.Core.Metadata.FormElement;  using Kingdee.BOS.Util;  public override void AfterLoadData(EventArgs e)  {  base.AfterLoadData(e);  //获取当前的控制策略类型  oldStrategyType = this.View.Model.GetValue("FStrategyType").ToString();  //单据锁定界面的通用控制还没做好，暂时先在这里处理一下了，  if (Convert.ToBoolean(this.View.Model.GetValue("FLocked")))  {  this.View.OpenParameter.Status = Core.Metadata.OperationStatus.VIEW;  }  else // add Leo 2013-06-09 20:44，恢复页面编辑状态  {  this.View.OpenParameter.Status = OperationStatus.EDIT;  }  //获取当前记录对应的Number和表单信息  curNumber = this.Model.GetValue(this.Model.BusinessInfo.GetForm().NumberFieldKey).ToString();  curFormMetadata = FormMetaDataCache.GetCachedFormMetaData(this.Context, curNumber);  Form curForm = curFormMetadata.BusinessInfo.GetForm();  //对当前基础资料进行检查，是否符合受控基础资料配置要求，如果不符合则不能进行操作。  if (!curForm.IsBaseDataTypeControl)  {  this.View.ShowWarnningMessage("该基础资料不受组织控制策略控制，不能对其进行操作，请确认。", action:  (imbr) =>  {  this.View.Close();  }  );  return;  }  if (curForm.CreateOrgFieldKey.IsNullOrEmptyOrWhiteSpace() || curForm.UseOrgFieldKey.IsNullOrEmptyOrWhiteSpace())  {  this.View.ShowWarnningMessage("该基础资料没有指定创建组织和使用组织，不能对其进行操作，请确认。" , action:  (imbr) =>  {  this.View.Close();  }  );  }  **this.View.GetBarItem("ToolBar", "tbSplitPrevious").Enabled = false;**  } | |

## GetControl

获取控件的编程对象。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| Control GetControl(string key);  T GetControl<T>(string key) where T:Control; | |

**参数说明**

Key为控件标识

**返回类型**

Kingdee.BOS.Core.DynamicForm.PlugIn.ControlModel. Control

**关键字**

控制控件状态 设置控件可见性

**备注**

获取控件信息，设置控件值、可见性等

### 案例1 数据绑定到界面之后：用于控制控件的状态

**代码来源**

查看业务流程图插件：

Kingdee.BOS.Business.DynamicForm.PlugIn.ConvertPlugIn.ViewBusinessFlowEdit，

Kingdee.BOS.Business.DynamicForm.PlugIn

**需求背景**

查看业务流程图界面，在数据绑定之后，需要根据数据设置标识为FbottomPnl的面板的可见性

**实现方案**

重写AfterBindData事件，根据标识获取控件，控制控件状态，设置控件的可见性。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.ComponentModel;  using System.Collections;  using Kingdee.BOS.Core.DynamicForm.PlugIn;  public override void AfterBindData(EventArgs e)  {  if (this.\_viewParameter == null)  return;  if (this.TotalPageCount() <= 1)  {  //隐藏  this.View.GetMainBarItem("tbSplitPrevious").Visible = false;  this.View.GetMainBarItem("tbSplitNext").Visible = false;  **this.View.GetControl("FBottomPnl").Visible = false;**  }  // 显示第一张业务流程图  this.ShowSpecPageFlowChat(0);  } | |

### 案例2 获取树形控件，构造树形控件节点数据

**代码来源**

辅助资料明细列表插件

Kingdee.BOS.Business.DynamicForm.PlugIn.Assistant. AssistantDetailList

Kingdee.BOS.Business.DynamicForm.PlugIn

**需求背景**

辅助资料明细列表界面，需要给左边的树绑定数据，显示辅助资料类别信息

**实现方案**

重写TreeLoadData事件，获取树形控件，构造树形控件节点数据，给树形控件绑定数据

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Data;  using Kingdee.BOS.Core.List.PlugIn;  using Kingdee.BOS.Core.DynamicForm.PlugIn.ControlModel;  using Kingdee.BOS.Util;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  TreeNode rootNode;  public override void TreeLoadData(TreeLoadDataArgs e)  {  **TreeView tv = this.View.GetControl<TreeView>("FGroupTreeView");**  rootNode = new TreeNode()  {  id = "0",  text = "全部",  cls = "parentnode"  };  rootNode.children.AddRange(BuildTree());  tv.SetRootNode(rootNode);  tv.SetExpanded(true);  e.Cancel = true;  }  /// <summary>  /// 构造组织树  /// </summary>  private List<TreeNode> BuildTree()  {  DataTable dtTypeInfo = GetAssistantDataType();  var query = from q in dtTypeInfo.AsEnumerable()  where q["FParentId"] == null || q["FParentId"].ToString().IsEmpty() == true  group q by new  {  id = q.Field<string>("ftopclassid"),  name = q.Field<string>("ftopclassid") == "fd1e82470267457fa433d2bf0f6ad028" ? "人力资源" : q.Field<string>("ftopclassname"),  } into grp  select new TreeNode  {  //业务领域  id = "TOP" + grp.Key.id,  text = grp.Key.name,  children = (from item in grp  select new TreeNode  {  //子节点  id = item.Field<string>("fid"),  text = item.Field<string>("fname"),  parentid = "TOP" + grp.Key.id,  children = CreateChildNode(dtTypeInfo, item)  }  ).OrderBy(p => p.text).ToList<TreeNode>()  };  List<TreeNode> nodes = query.ToList<TreeNode>();  this.View.OpenParameter.SetCustomParameter("nodes", nodes);  return nodes;  } | |

## GetEntryState

获取分录的数据显示情况：总页数和当前页数

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| IEntryState GetEntryState(string entryKey); | |

**参数说明**

entryKey：分录唯一标识

**返回类型**

分录显示的数据情况

**关键字**

分录

### 案例

暂缺

## GetFieldEditor

获取字段控件编程对象。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| FieldEditor GetFieldEditor(FieldAppearance fieldAppearance, int row);  T GetFieldEditor<T>(string key, int row) where T : FieldEditor; | |

**参数说明**

Key：为字段标识

Row：如果字段是分录中，对应分录行号

**返回类型**

Kingdee.BOS.Core.DynamicForm.PlugIn.ControlModel.FieldEditor

**关键字**

设置控件可见性 设置控件可用性 设置控件值

**备注**

获取控件，设置控件可见性、可用性和值等。对于单据头上的字段，GetControl和GetFielEditor都可以拿到控件对象，但是如果字段在单据体，则只能通过GetFieldEditor获取

### 案例 获取下拉按钮控件，给下拉按钮赋值

**代码来源**

Kingdee.BOS.Business.PlugIn.Globalization.RegionalSettingsEdit，

Kingdee.BOS.Business.PlugIn.Globalization

**需求背景**

国际化区域设置界面，需要给“本地数字格式”下拉列表字段，动态绑定枚举列表

**实现方案**

根据标识获取下拉按钮控件，绑定枚举列表

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using Kingdee.BOS.Core.DynamicForm.PlugIn;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.Util;  using Kingdee.BOS.Core;  using Kingdee.BOS.Core.DynamicForm.PlugIn.ControlModel;  using System.Globalization;  using Kingdee.BOS.Orm.DataEntity;  private void SetFNegativeNumberFormatList()  {  List<EnumItem> listSubItems = new List<EnumItem>();  string negativeSymbol = this.Model.GetValue("FNegative").ToString();  string dot = this.Model.GetValue("FDot").ToString();  listSubItems.Add(new EnumItem() { Caption = new LocaleValue("(1" + dot + "1)", this.View.Context.UserLocale.LCID), EnumId = 0.ToString(), Seq = 0, Value = 0.ToString() });  listSubItems.Add(new EnumItem() { Caption = new LocaleValue(negativeSymbol + "1" + dot + "1", this.View.Context.UserLocale.LCID), EnumId = 1.ToString(), Seq = 1, Value = 1.ToString() });  listSubItems.Add(new EnumItem() { Caption = new LocaleValue(negativeSymbol + " 1" + dot + "1", this.View.Context.UserLocale.LCID), EnumId = 2.ToString(), Seq = 2, Value = 2.ToString() });  listSubItems.Add(new EnumItem() { Caption = new LocaleValue("1" + dot + "1" + negativeSymbol, this.View.Context.UserLocale.LCID), EnumId = 3.ToString(), Seq = 3, Value = 3.ToString() });  listSubItems.Add(new EnumItem() { Caption = new LocaleValue("1" + dot + "1 " + negativeSymbol, this.View.Context.UserLocale.LCID), EnumId = 4.ToString(), Seq = 4, Value = 4.ToString() });  ComboFieldEditor combox = **this.View.GetFieldEditor("FNegativeNumberFormat", 0)** as ComboFieldEditor;  combox.SetComboItems(listSubItems);  } | |

## GetFormOperation

获取指定的表单操作，该指定操作需要已经配置在表单的操作列表中。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| IFormOperation GetFormOperation(string operationNumber);  IFormOperation GetFormOperation(FormOperationEnum operationType); | |

**参数说明**

operationNumber 为操作编码，在设计器中新增表单操作时定义

**返回类型**

Kingdee.BOS.Core.DynamicForm. IFormOperation

**关键字**

执行操作 验证操作权限

**备注**

获取操作实例对象，执行操作，验证操作权限

### 案例 获取批量修改操作对象，验证是否有批量修改权限

**代码来源**

Kingdee.BOS.Business.PlugIn.BaseData.ChangeFieldContent,

Kingdee.BOS.Business.PlugIn

**需求背景**

基础资料批改界面，需要验证当前用户对批改的基础资料是否拥有批改权

**实现方案**

根据批改的操作代码BULKEDIT，获取批量修改操作对象，验证是否有批量修改权限。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using Kingdee.BOS.Core.DynamicForm.PlugIn;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.Core.Metadata.FieldElement;  using Kingdee.BOS.Core.Metadata.ControlElement;  using Kingdee.BOS.Util;  using System.Web.UI.WebControls;  using Kingdee.BOS.Core.DynamicForm.PlugIn.ControlModel;  using Kingdee.BOS.Core.Metadata.FormElement;  using Kingdee.BOS.Core.Metadata.EntityElement;  using Kingdee.BOS.ServiceHelper;  using Kingdee.BOS.Core.Permission;  using System.Data;  using Kingdee.BOS.Core.DynamicForm;  using Kingdee.BOS.Core.List;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS.Core;  private void DoRealBatchModify(Field field, string tableName, object fieldValue, ListSelectedRow[] parentSelectRowsInfo)  {  List<string> \_notBachModifyPk = new List<string>();  IFormOperation formOperation = **this.View.ParentFormView.ParentFormView.GetFormOperation("BULKEDIT");**  //判断是否有批量修改权限  foreach (ListSelectedRow sr in parentSelectRowsInfo)  {  // IformOperation接口提供的方法验权  if (!formOperation.FuncPermissionAuth(sr.PrimaryKeyValue, false))  {  this.ResultWriteText(field, (int)MsgType.FaildBachModifyPermission, sr.Number, sr.PrimaryKeyValue);  \_notBachModifyPk.Add(sr.PrimaryKeyValue);  continue;  }  }  var hsBachModify = ((\_notBachModifyPk.Count > 0) ? (from p in parentSelectRowsInfo where \_notBachModifyPk.IndexOf(p.PrimaryKeyValue) < 0 select p.PrimaryKeyValue)  : (from p in parentSelectRowsInfo select p.PrimaryKeyValue)).Distinct().ToList();  //修改权限处理  var hsPermission = PermissionFilter(hsBachModify);  var pkids = (from p in parentSelectRowsInfo  where hsPermission.Contains(p.PrimaryKeyValue) && \_notBachModifyPk.IndexOf(p.PrimaryKeyValue) < 0  select p.PrimaryKeyValue).ToList();  //首先写入返回非权限修改的结果集  foreach (ListSelectedRow sr in parentSelectRowsInfo)  {  if ((pkids.Count() == 0 || pkids.Contains(sr.PrimaryKeyValue) == false) &&  \_notBachModifyPk.IndexOf(sr.PrimaryKeyValue) < 0)  this.ResultWriteText(field, (int)MsgType.FaildPermission, sr.Number, sr.PrimaryKeyValue);  }  if (pkids.Count() == 0)  {  msgType = (int)MsgType.FaildPermission;  return;  }  IEnumerable<string> allcatePkids = null;  …………………………..中间代码省略  BusinessDataServiceHelper.ClearCache(this.Context, \_metadata.BusinessInfo.GetDynamicObjectType(), pkids);  } | |

## GetFormTitle

获取当前界面标题

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| LocaleValue GetFormTitle(); | |

**参数说明**

无

**返回类型**

界面标题，多语言对象，可以通过ToSting()方法，取得当前语言下的内容；

**关键字**

标题

### 案例

暂缺

## GetLookupConfig

获取基础资料的模糊查找配置。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| JSONObject GetLookupConfig(FieldAppearance fieldAppearance); | |

**参数说明**

fieldAppearance：字段外观元数据

**返回类型**

配置情况

**关键字**

查找

**备注**

BOS平台内部专用方法，插件请勿调用；

### 案例

暂缺

## GetMainBarItem

获取主菜单项操作对象。

### 语法

**方法定义：**

|  |  |
| --- | --- |
| C# |  |
| BarItemControl GetMainBarItem(string barItemName); | |

**参数说明**

barItemName：菜单项标识

**返回类型**

Kingdee.BOS.Core.DynamicForm.PlugIn.ControlModel. BarItemControl

**关键字**

设置主菜单项是否可用 设置主菜单项是否可见

**备注**

获取主菜单项操作对象，设置是否可用，可见，标题等

### 案例 根据打开单据状态设置菜单项按钮是否可用

**代码来源**

附件管理单据插件

Kingdee.BOS.Business.PlugIn.Attachment.AttachmentBillPlugIn,

Kingdee.BOS.Business.PlugIn

**需求背景**

附件管理单据界面，需要根据父单据的状态，设置附件管理主菜单是否可用

**实现方案**

重写AfterBindData 事件，从this.View.OpenParameter.Status中获取当前单据是查看还是编辑状态，根据打开单据状态设置菜单项按钮是否可用。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Web;  using System.IO;  using System.Xml.Serialization;  using System.Net;  using Kingdee.BOS.Core.Attachment;  using Kingdee.BOS.Core.Metadata;  public override void AfterBindData(EventArgs e)  {  if (!CheckLocalNetIP(this.Context))  {  InitViewForm();  return;  }  AttachmentKey attachmentKey = GetAttachmentKey();  if (attachmentKey.OperationStatus == OperationStatus.VIEW)  {  this.View.OpenParameter.Status = attachmentKey.OperationStatus;  }  //编辑状态，上传控件初始值  if (this.View.OpenParameter.Status == Core.Metadata.OperationStatus.EDIT)  {  InitUploadCtrlValue();  this.View.GetControl(FileUpdate).Enabled = false;  **this.View.GetMainBarItem(tbbtnSave).Enabled = true;**  **this.View.GetMainBarItem(tbbtnSubmit).Enabled = true;**  **this.View.GetMainBarItem(tbbtnAudit).Enabled = true;**  **this.View.GetMainBarItem(tbbtnUnAudit).Enabled = true;**  }  //查看状态  else if (this.View.OpenParameter.Status == Core.Metadata.OperationStatus.VIEW)  {  InitViewForm();  }  else if (this.View.OpenParameter.Status == Core.Metadata.OperationStatus.ADDNEW)  {  this.View.GetMainBarItem(tbbtnSubmit).Enabled = false;  this.View.GetMainBarItem(tbbtnAudit).Enabled = false;  this.View.GetMainBarItem(tbbtnUnAudit).Enabled = false;  InitUploadCtrlValue();  }  EnabledBarItemByBillStatus();  this.View.GetControl(FileUpdate).SetCustomPropertyValue("ServerCustomUploading", false);  SetFileServerUrl();  } | |

## GetMainMenu

获取表单的主菜单信息，主菜单特指在使用BOS设计器设计表单时，【菜单集合】属性中配置的菜单。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| MenuControl GetMainMenu(); | |

**参数说明**

无

**返回类型**

主菜单信息

**关键字**

主菜单

### 案例

暂缺

## GetMenu

获取指定标识的菜单。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| MenuControl GetMenu(string key); | |

**参数说明**

key：菜单唯一标识

**返回类型**

菜单控件编程对象，可以通过此对象添加子菜单项；

**关键字**

菜单

### 案例

暂缺

## GetService

获取指定类型动态表单服务。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| T GetService<T>(); | |

**参数说明**

T：要获取的服务类型

**返回类型**

获取的服务

**关键字**

表单服务

### 案例

暂缺

## GetTmpfilePath

获取文件在服务器上的物理路径。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| string GetTmpfilePath(string path); | |

**参数说明**

path：文件的相对路径

**返回类型**

文件的物理路径

**关键字**

路径

### 案例

暂缺

## GetView

获取指定界面的视图实例。

### 语法

**方法定义：**

|  |  |
| --- | --- |
| C# |  |
| IDynamicFormView GetView(string pageId); | |

**参数说明**

pageId：页面唯一标识

**返回类型**

Kingdee.BOS.Core.DynamicForm.IDynamicFormView

**关键字**

获取视图实例

**备注**

根据页面pageId获取视图实例，调用视图的方法获取视图的数据、状态等。

### 案例 获取页面视图实例，关闭页面

**代码来源**

单据类型单据插件

Kingdee.BOS.Business.PlugIn.BillTypeEdit,

Kingdee.BOS.Business.PlugIn

**需求背景**

平台的单据类型编辑界面，打开通用的单据类型编辑后，加载该单据类型所对应单据业务对象上配置的【单据类型参数对象】，即平台的单据类型除通用的属性字段外，还允许用户扩展具体的单据类型参数，用于单据类型的定制化开发。

**实现方案**

AddParaSettingUI函数是用于加载或刷新单据类型参数对象视图的，首先加载单据类型参数对象的扩展视图时curOpenPageId为空，刷新时采用的是先关闭之前的视图再重新打开(ShowForm)的形式，获取之前的视图可能过GetView获取。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Linq;  using System.Collections.Generic;  using Kingdee.BOS.Util;  using System.Text;  using System.Data;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.ServiceHelper;  using Kingdee.BOS.Core.DynamicForm;  using Kingdee.BOS.Core.Bill.PlugIn;  private void AddParaSettingUI(bool canEdit = true)  {  string billTypePara = "BOS\_BILLTYPEPARAMODEL";  bool ParaVisible = true;  if (FBillFormID.IsNullOrEmptyOrWhiteSpace())  {  ParaVisible = false;  }  else  {  string billTypeParameter = ((FormMetadata)FormMetaDataCache.GetCachedFormMetaData(this.Context, FBillFormID)).BusinessInfo.GetForm().BillTypePara;  if (billTypeParameter.IsNullOrEmptyOrWhiteSpace())  {  ParaVisible = false;  }  else  {  billTypePara = billTypeParameter;  }  }  this.View.GetControl("FParaSetting").Visible = ParaVisible;  if (!curOpenPageId.IsNullOrEmpty())  {  IDynamicFormView view = this.View.GetView(curOpenPageId);  if (view == null)  {  return; //连续几次调用本函数的时候，curOpenPageId存在，但是前面一次的View还没Show出来  }  //需要先关闭界面  view.Close();  this.View.SendDynamicFormAction(view);  }  DynamicFormShowParameter param = new DynamicFormShowParameter();  param.FormId = billTypePara;  param.ParentPageId = this.View.OpenParameter.PageId;  param.OpenStyle.TagetKey = "FParaSettingPanel";  param.OpenStyle.ShowType = ShowType.InContainer;  param.PageId = curOpenPageId = Guid.NewGuid().ToString();  if (!this.Model.DataObject["BillFormID\_ID"].IsNullOrEmptyOrWhiteSpace())  {  param.CustomParams.Add("BillFormID", this.Model.DataObject["BillFormID\_ID"].ToString());  }  bool HasBillBormTypeId = false;  if (!this.Model.DataObject["ID"].IsNullOrEmptyOrWhiteSpace())  {  HasBillBormTypeId = true;  param.CustomParams.Add("BillFormTypeID", this.Model.DataObject["ID"].ToString());  param.CustomParams.Add("BillFormTypeStatus", this.Model.GetValue("FDocumentStatus").ToString());  }  var primaryKey = this.View.OpenParameter.GetCustomParameter("PrimaryKey");  if (!primaryKey.IsNullOrEmptyOrWhiteSpace() && !HasBillBormTypeId)  {  param.CustomParams.Add("BillFormTypeID", primaryKey.ToString());  }  this.View.ShowForm(param);  } | |

## InvokeFieldUpdateService

执行指定字段的表单服务。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void InvokeFieldUpdateService(string key, int row); | |

**参数说明**

key：字段唯一标识

row：字段所在的行索引

**返回类型**

无

**关键字**

表单服务

**备注**

字段”值更新事件”属性上配置的表单服务，仅用户在字段控件中，手书录入字段值时会自动触发，其他方式对字段值的修改（如插件设置字段值），则不会触发；

其他方式修改字段值，如果需要触发字段”值更新事件”上配置的表单服务，需要调用此方法，强制触发。

### 案例

暂缺

## InvokeFormOperation

调用表单操作。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| bool InvokeFormOperation(FormOperationEnum operation);  bool InvokeFormOperation(string operationNumber); | |

**参数说明**

operationNumber：操作代码

Operation：操作的枚举类型，比如FormOperationEnum.Save

**返回类型**

bool：操作是否成功

**关键字**

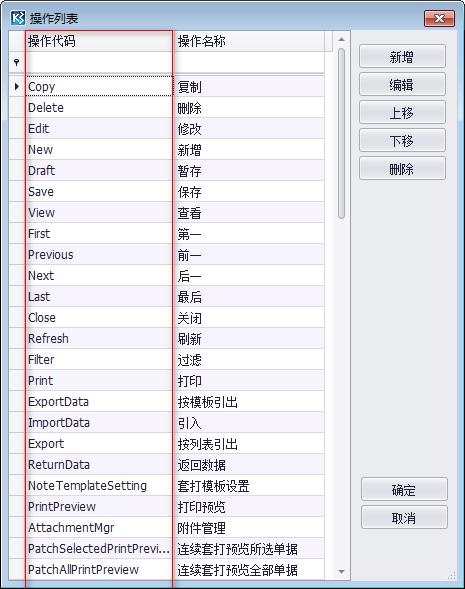
调用操作

**备注**

FormOperationEnum是预定义的表单操作标识的枚举类型，在平台中每个操作会用一个操作码来标识这个操作，这些操作一般是平台提供的，预置到用户账套中通过反射调用平台代码，二次开发也可以参考平台实现自已的操作，以提供自已的单据配置使用。

由于第一个参数是枚举类型，后面加的操作如果没有维护这个枚举，或者如果是业务或二次开发添加的操作无法维护这个枚举，就没有办法调用表单的操作，所以提供了下面的直接传入操作代码来调用表单操作的方法。还有一个原因就是这个操作代码，用户可以自定义

操作代码在用户配置操作时， 打开操作列表，第一列就是表示参数操作代码（如下图），用户点击编辑可修改自定义这个操作代码。



### 案例 调用保存操作

**代码来源**

辅助资料类别表单插件

Kingdee.BOS.Business.DynamicForm.PlugIn.Assistant.AssistantTypeEdit,

Kingdee.BOS.Business.DynamicForm.PlugIn

**需求背景**

辅助资料类别界面，需要在保存前进行校验并提示，根据提示在选择是否再调用保存操作

**实现方案**

在页面保存前，进行相关校验，取消当前保存操作e.Cancel = true，并给出消息提示和确认，确认后主动调用保存操作。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using Kingdee.BOS.Core.Bill.PlugIn;  using Kingdee.BOS.Util;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS.Core.DynamicForm;  using Kingdee.BOS.Core.Metadata;  public override void BeforeSave(Core.Bill.PlugIn.Args.BeforeSaveEventArgs e)  {  base.BeforeSave(e);  int lcid = this.View.Context.UserLocale.LCID;  object topClassId = ((DynamicObject)this.Model.GetValue("FTopClassId"))["id"];  object parentValue = this.Model.GetValue("FParentId");  object nameValue = ((LocaleValue)this.Model.GetValue("FName"))[lcid];  object dataId = this.Model.DataObject["Id"];  if (parentValue != null)  {  object parentId = ((DynamicObject)parentValue)["id"];  object parentName = ((LocaleValue)((DynamicObject)parentValue)["Name"])[lcid];  object parentTopId = ((DynamicObject)this.Model.GetValue("FParentId"))["TopClassId\_Id"];  if (parentName.Equals(nameValue) || (dataId != null && dataId.Equals(parentId)))  {  e.Cancel = true;  this.View.ShowErrMessage("上级类别不能为辅助资料类别本身！");  return;  }  if (IsChildNode(parentId))  {  e.Cancel = true;  this.View.ShowErrMessage(string.Format("【{0}】已是【{1}】的下级节点，不能作为【{1}】的上级类别！", parentName, nameValue));  return;  }  if (!topClassId.Equals(parentTopId) && !\_canSave)  {  e.Cancel = true;  this.View.ShowMessage("【业务领域】与【上级类别】所属业务领域不一致，保存时以【上级类别】所属领域为准，确认继续？", MessageBoxOptions.OKCancel,  (cfm) =>  {  if (cfm == MessageBoxResult.OK)  {  \_canSave = true;  **this.View.InvokeFormOperation(FormOperationEnum.Save);**  }  });  }  //（BT795707）假如修改上级类别，那么当前类别下的辅助资料的上级类别信息应该进行清空，提示后再进行清空处理  if (this.View.OpenParameter.Status == OperationStatus.EDIT && \_parentid != null && !parentId.Equals(\_parentid) && !\_canSave)  {  e.Cancel = true;  this.View.ShowMessage("上级类别发生改变后，若辅助资料引用了该类别对应上级资料，上级资料信息将会被清空，确认要修改？", MessageBoxOptions.OKCancel,  (cfm) =>  {  if (cfm == MessageBoxResult.OK)  {  ClearParentId(dataId);  \_canSave = true;  **this.View.InvokeFormOperation(FormOperationEnum.Save);**  }  });  }  }  } | |

## InvokeFormOperation4Wf

专用于工作流处理界面，调用表单操作。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| bool InvokeFormOperation4Wf(InvokeFormOperation4WfParam param,  CallBackWfAfterOperation callbackwf); | |

**参数说明**

param：工作流执行表单操作参数

callbackwf：工作流操作回调函数

**返回类型**

表单操作是否执行成功

**关键字**

工作流

**备注**

此方法专用于工作流任务处理，插件请勿调用；

### 案例

暂缺

## LockBill

锁定页面。

### 语法

**方法定义：**

|  |  |
| --- | --- |
| C# |  |
| void LockBill(); | |

**参数说明**

·无

**备注**

·锁定视图上所有的控件，整个页面锁定。

### 案例 锁定页面

**代码来源**

Kingdee.K3.FIN.CB.Business.PlugIn.BillForm.CBCommonBillEdit,

Kingdee.K3.FIN.CB.Business.PlugIn

**需求背景**

财务成本单据界面，初始进入的时候需要锁定页面。

**实现方案**

重写AhterBindData事件，使用LockBill方法，锁定整个页面

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| public override void AfterBindData(EventArgs e)  {  base.AfterBindData(e);  **this.LockBill();**  this.IsBDMaterialShare = CommonServiceHelper.IsBaseDataShare(this.Context, Kingdee.K3.Core.BD.FormKeyConst.BD\_MATERIAL);  } | |

## LockField

锁定或者解锁字段。

### 语法

**方法定义：**

|  |  |
| --- | --- |
| C# |  |
| void LockField(string key, bool bEnabled); | |

**参数说明**

Key：字段标识

bEnabled：是否锁定，true为锁定，false为解锁

**关键字**

锁定字段 解锁字段

**备注**

锁定单据头字段或单据体的列，等价于插件中调用this.View.StyleManager.SetEnable(field, null, bEnabled)，如果要控制单据体的某个单元格（即某行某列）的锁定性，应该调用this.View.StyleManager.SetEnable带行数据包参数的重载方法。如果执行了锁定单元格操作，受它的影响没有办法再执行锁定该列的所有单元格，只能遍历去锁定单元格。

### 案例 根据帐套类型解锁相关字段

**代码来源**

数据中心插件

Kingdee.BOS.Business.PlugIn.DataCenter.DataCenterEdit,

Kingdee.BOS.Business.PlugIn

**需求背景**

管理中心的数据中心编辑界面，由于Oracle不需要输入数据库账套保存路径和日志保存路径，所以当账套类型为SQL Server时解锁相关字段，否则锁定相关字段

**实现方案**

在AfterBindData和DataChanged事件中调用，判断数据库类型，所以当类型为SQL Server时解锁相关字段，否则锁定相关字段

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using Kingdee.BOS.Core.Bill.PlugIn;  using System.Data;  using System.Collections.Generic;  using System.Data.Sql;  using Kingdee.BOS.Util;  using System.Xml;  public override void DataChanged(DataChangedEventArgs e)  {  base.DataChanged(e);  bool needRefreshForm = false;  //修改的是数据库类型  if (e.Field.FieldName.Equals("FDatabaseType", StringComparison.InvariantCultureIgnoreCase))  {  \_selectedDatabaseType = (DatabaseType)Convert.ToInt32(e.NewValue);  needRefreshForm = true;  }  //修改数据库名称  if ((e.Field.FieldName.Equals("FDatabaseType", StringComparison.CurrentCultureIgnoreCase)  || e.Field.FieldName.Equals("FDatabaseserver", StringComparison.CurrentCultureIgnoreCase)  || e.Field.FieldName.Equals("FDatabaseUser", StringComparison.CurrentCultureIgnoreCase)  || e.Field.FieldName.Equals("FDatabasePwd", StringComparison.CurrentCultureIgnoreCase)  )  && \_bIsRegisterDataCenterForm  && !lastGetDBListServer.Equals(this.Model.GetValue("FDatabaseserver") as string, StringComparison.CurrentCultureIgnoreCase)  && \_selectedDatabaseType == DatabaseType.MS\_SQL\_Server)  {  lastGetDBListServer = this.Model.GetValue("FDatabaseserver") as string;  lastGetDBListServer = lastGetDBListServer == null ? string.Empty : lastGetDBListServer;  }  if (needRefreshForm)  RefreshForm();  }  private void RefreshForm()  {  if (\_curOperationStatus == OperationStatus.VIEW)  {  return;  }  //SQL Server  if (\_selectedDatabaseType == DatabaseType.MS\_SQL\_Server)  {  **this.View.LockField("FDatabaseName", true);**  **this.View.LockField("FDataBasePath", true);**  **this.View.LockField("FDatabaseLogPath", true);**  }  else  {  **this.View.LockField("FDatabaseName", false);**  **this.View.LockField("FDataBasePath", false);**  **this.View.LockField("FDatabaseLogPath", false);**  //数据库名称清空  this.View.Model.SetValue("FDatabaseName", "");  this.View.Model.SetValue("FDataBasePath", "");  this.View.Model.SetValue("FDatabaseLogPath", "");  }  //非创建界面，锁定路径字段  if (!\_bIsNewDataCenterForm)  {  this.View.LockField("FDataBasePath", false);  this.View.LockField("FDatabaseLogPath", false);  this.View.LockField("FAcctType", false);  }  //修改状态下锁定的字段  if (\_curOperationStatus != OperationStatus.ADDNEW)  {  this.View.LockField("FDatabaseName", false);  this.View.LockField("FDatabaseType", false);  this.View.LockField("FDatabaseserver", false);  this.View.GetControl("FAcctType").Visible = false;  }  } | |

## PopupWindow

弹窗显示定制页面。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void PopupWindow(String url, String popupMode, String name, String title, int w, int h); | |

**参数说明**

url：定制页面的Url

popupMode：弹窗模式

name：窗体名称

title：窗体标题

w：窗体宽度

h：窗体高度

**返回类型**

无

**关键字**

定制页面

### 案例

暂缺

## Print

打印。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void Print();  void Print(String formId, String layoutId, String[] pks); | |

**参数说明**

formId：打印的单据唯一标识

layoutId：打印的单据的布局Id

pks：单据内码集合

**返回类型**

无

**关键字**

打印

### 案例

暂缺

## Refresh

刷新页面。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void Refresh(); | |

**参数说明**

无

**返回类型**

无

**关键字**

刷新

### 案例

暂缺

## RefreshByFilter

刷新页面，强制重新取数；

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void RefreshByFilter(); | |

**方法描述**

**参数说明**

无

**返回类型**

无

**关键字**

刷新

**备注**

如果过滤条件是通过插件实现，此时需要调用改方法才能获取数据。

### 案例

暂缺

## RegisterPlugIn

注册插件实例。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void RegisterPlugIn(AbstractDynamicFormPlugIn plugIn); | |

**参数说明**

plugIn：插件类

**返回类型**

无

**关键字**

注册插件

### 案例

暂缺

## RemovePlugIn

移除插件，业务注入插件自己创建，自己移除。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void RemovePlugIn(AbstractDynamicFormPlugIn plugIn); | |

**参数说明**

plugIn：插件类

**返回类型**

无

**关键字**

移除插件

### 案例

暂缺

## ResetState

重新设置指定控件界面状态，用于操作后的界面刷新。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void ResetState(List<string> keys); | |

**参数说明**

keys：控件唯一标识

**返回类型**

无

**关键字**

重置状态

### 案例

暂缺

## ReturnToParentWindow

返回数据给父窗体的回调函数。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void ReturnToParentWindow(FormResult formResult);  void ReturnToParentWindow(object param); | |

**参数说明**

param：返回给父窗体的参数对象;

formResult：子窗体的操作结果

**返回类型**

无

**关键字**

返回 父页面 父窗口 数据交换 回调

**备注**

把当前页面返回给父窗口的参数放到ParentFormView.Session[“returnData”]缓存区，当前页面关闭时，把参数对象作为参数的方式传递给父窗口调用ShowForm的回调函数。

### 案例 返回工作日历中编辑的班制信息给父窗口

**代码来源**

Kingdee.K3.MFG.ENG.Business.PlugIn.Base.WorkCalCustomEdit,

Kingdee.K3.MFG.ENG.Business.PlugIn

Kingdee.K3.MFG.ENG.Business.PlugIn.Base.ShiftEdit,

Kingdee.K3.MFG.ENG.Business.PlugIn

**需求背景**

在制造的特性日历中， 点击日历中具体日期的班制编码时，打开编辑已有班制或新增班制界面，编辑完成返回更新编辑行的班制信息。

**实现方案**

点击日历中具体日期的班制编码时，打开编辑已有班制或新增班制界面，当编辑完成关闭班制编辑界面前， 调用ReturnToParentWindow方法返回班制信息结构体

WorkCalCustomEdit.T\_EditShiftParam workCalSetupParam给父窗口的ShowForm回调函数，回调函数在获取返回的结构体数据包后，给当前的班制赋值。

**示例代码**

Kingdee.K3.MFG.ENG.Business.PlugIn.Base.ShiftEdit.BeforeClosed

调用ReturnToParentWindow返回给父窗口

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.Bill.PlugIn;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using System;  namespace ComonPlugIn  {  public class WorkCalCustomEdit: AbstractBillPlugIn  {  /// 班制页面关闭前事件  /// </summary>  /// <param name="e"></param>  public override void BeforeClosed(BeforeClosedEventArgs e)  {  base.BeforeClosed(e);  if (workCalSetupParam.SourceCaller == 1)  {  long ReturnValue = 0;  long.TryParse(Convert.ToString(this.View.Model.GetPKValue()), out ReturnValue);  if (ReturnValue > 0)  {  workCalSetupParam.ShiftId = ReturnValue;  this.View.ReturnToParentWindow(workCalSetupParam);  }  }  }  }  } | |

Kingdee.K3.MFG.ENG.Business.PlugIn.Base.WorkCalCustomEdit.BeforeF7Select

父窗口在ShowForm的回调函数中，获取子窗口调ReturnToParentWindow传递过来的数据

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.Bill;  using Kingdee.BOS.Core.Bill.PlugIn;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.K3.Core.MFG.Utils;  using System;  namespace ComonPlugIn  {  public class WorkCalCustomEdit : AbstractBillPlugIn  {  public override void BeforeF7Select(BeforeF7SelectEventArgs e)  {  base.BeforeF7Select(e);  switch (e.FieldKey)  {  case Consts.WorkCalendarBill.FIELDKEY\_ENTRY\_SHIFTID:  e.Cancel = true;  this.FShiftIdPrivate(e.Row);  break;  }  }  /// <summary>  /// 表体配置私有班制信息  /// </summary>  /// <param name="RowIndex">表体行索引</param>  private void FShiftIdPrivate(int RowIndex)  {  long FShiftIDValue = this.View.Model.GetValue<long>("FSHIFTID", RowIndex, 0);  var FShift = this.Model.GetValue<DynamicObject>("FSHIFTID", RowIndex, null);  //("FSHIFTID", e.Row) == null ? "" : this.View.Model.GetValue("FSHIFTID", e.Row).ToString();  BillShowParameter billShowPara = new BillShowParameter();  billShowPara.FormId = MFGFormIdConst.SubSys\_ENG.ShiftBill;  billShowPara.ParentPageId = this.View.PageId;  billShowPara.MultiSelect = false;  billShowPara.Status = OperationStatus.ADDNEW;  T\_EditShiftParam param = new T\_EditShiftParam();  param.ShiftId = FShiftIDValue;  param.SourceCaller = 1;  param.UserOrgId = this.Model.GetValue<long>("FUseOrgId");  if (FShiftIDValue == 0 || !FShift.GetDynamicObjectItemValue("IsPrivate", false))  {  this.View.ShowForm(MFGFormIdConst.SubSys\_ENG.ShiftBill, param, result =>  {  if (result.ReturnData is T\_EditShiftParam && ((T\_EditShiftParam)result.ReturnData).ShiftId > 0)  {  this.View.Model.SetValue("FSHIFTID", ((T\_EditShiftParam)result.ReturnData).ShiftId, RowIndex);  }  });  }  else  {  billShowPara.Status = OperationStatus.EDIT;  billShowPara.FormId = MFGFormIdConst.SubSys\_ENG.ShiftBill;  if (!FShiftIDValue.ToString().Equals("0"))  {  billShowPara.PKey = FShiftIDValue.ToString();  //补上EditShiftParam：控制保存后直接审核  this.View.Session["FormInputParam"] = param;  }  this.View.ShowForm(billShowPara, result =>  {  if (FShiftIDValue != 0)  {  this.View.Model.SetValue("FSHIFTID", "", RowIndex); this.View.Model.SetValue("FSHIFTID", FShiftIDValue, RowIndex);  }  });  }  }  }  } | |

## SendAynDynamicFormAction

执行指定动态表单视图的客户端指令,在返回到客户端后再获取状态。

### 语法

**方法定义：**

|  |  |
| --- | --- |
| C# |  |
| void SendAynDynamicFormAction(IDynamicFormView view); | |

**参数说明**

view：要执行的指定动作的动态表单视图

**返回类型**

无

**关键字**

异步 刷新父窗面

**备注描述**

一般来说，服务端只会响应当前发起请求的页面动作，如果要立即响应父窗口或其它窗口的动作，需要调用该方法往客户端发送执行指定动态表单的指令，客户端收到该指令后，会重新发起请求，请求该指定的动态表单的指令集Actions。

它跟SendDynamicFormAction的最大区别是，SendDynamicFormAction的更新父表单的指令集Actions会在此次请求中连同当前表单的指令集一起返回给客户端，而SendAynDynamicFormAction会发一条指令给客户端，客户端收到指令重新发起请求，请求目标View的指令集，这个请求是异步的。

### 案例 修改父页面字段状态，执行父页面的客户端指令

**代码来源**

HR系统参数插件

Kingdee.K3.HR.HM.Business.PlugIn.SystemParamEdit,

Kingdee.K3.HR.HM.Business.PlugIn

**需求背景**

所有子系统的参数表单都内嵌到平台的系统参数框架表单的一个面板里面。所以对于各自子系统的参数表单插件，this.view就是自己的子系统参数页面，this.View.ParentView则是外面的平台系统参数框架页面。

现在HR子系统的参数设置页面，需要在数据绑定之后，根据当前的用户类型，给当前的参数设置页面上的控件设置锁定性，同时要给外面的平台参数设置页面上的组织机构字段设置可用性。

**实现方案**

绑定数据之后，先设置当前页面控件的锁定性；然后根据需要，给父页面的控件赋值并设置锁定性，

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Linq;  using System.Text;  using Kingdee.BOS.Core.Bill.PlugIn;  using Kingdee.BOS.Core.DynamicForm.PlugIn;  using Kingdee.BOS.Core.Permission;  namespace Kingdee.K3.HR.HM.Business.PlugIn  {  [Description("参数插件")]  public class SystemParamEdit : AbstractDynamicFormPlugIn  {  public override void AfterBindData(EventArgs e)  {  var view = this.View.ParentFormView;  if (view != null)  {  List<UserType> lstUserType = view.Model.UserTypes;  if (lstUserType[0] != UserType.AdminUser)  {  this.View.GetControl("FIsHRControl").Enabled = false;  }  view.Model.SetValue("FOrgId", 0);  view.GetControl("FOrgId").Enabled = false;  this.View.**SendAynDynamicFormAction**(view);  this.View.Model.DataChanged = false;  view.Model.DataChanged = false;  }  }    }  } | |

## SendDynamicFormAction

执行指定动态表单视图的客户端指令。

### 语法

**方法定义：**

|  |  |
| --- | --- |
| C# |  |
| void SendDynamicFormAction(IDynamicFormView view); | |

**参数说明**

View：要发送指令的动态表单视图

**返回类型**

无

**关键字**

立即 即时 刷新父窗面

**备注描述**

立即发送指定视图的动作到客户端执行，用于立即执行刷新当前视图外的视图控件的数据和状态，或者执行表单的动作。【请参考SendAynDynamicFormAction，查看二者的区别】

### 案例 执行指定动态表单视图的客户端指令

**代码来源**

Kingdee.BOS.Business.PlugIn.BaseData.BaseDataParameterEdit,

Kingdee.BOS.Business.PlugIn

**需求背景**

参数设置里是每个领域设置自已参数的地方，具体的参数对象，内嵌在平台的父表单中，由平台动态加载对应的参数对象表单，这里需要在子表单的参数对象表单中锁定平台父表单的字段。

**实现方案**

下面的代码是参数设置的基础资料插件中，该插件是用于放置税组合启用相关的逻辑，由于该插件表单是内嵌表单，启用多组织时，父表单上有个组织字段，但是税组合参数跟组织无关，所以这里锁定父表单上的组织字段，不让用户操作。

子表单要控制父表单的元素状态或数据，需要调用 SendAynDynamicFormAction才能立即更新。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using Kingdee.BOS.Core.DynamicForm.PlugIn;  namespace Kingdee.BOS.Business.PlugIn.BaseData  {  public partial class BaseDataParameterEdit : AbstractDynamicFormPlugIn  {  public override void BeforeBindData(EventArgs e)  {  BindTaxMixParameterData();  this.View.ParentFormView.GetControl("FOrgId").Enabled = false;  this.View.SendDynamicFormAction(this.View.ParentFormView);  base.BeforeBindData(e);  }  }  } | |

## SetClientDialogResult

设置交互界面的选择结果；

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void SetClientDialogResult(MessageBoxResult messageBoxResult); | |

**参数说明**

messageBoxResult：选择结果，包括Yes,No,Cancel等

**返回类型**

无

**关键字**

对话框结果

**备注**

交互界面打开后，通常会提供几个选项，供用户选择，包括Yes、No、Cancel等。交互界面需要根据用户选择的结果，处理如何返回值给交互界面的父窗口；

### 案例

暂缺

## SetClientReturnData

设置交互界面要返回给父窗口的数据；

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void SetClientReturnData(object retValue); | |

**参数说明**

retValue：返回值

**返回类型**

无

**关键字**

返回值

**备注**

本方法需与SetClientDialogResult配合使用， SetClientDialogResult方法，设置交互界面是否把选择结果返回给父窗口(Yes会返回，No则不返回）；而本方法，则设置交互界面需要返回给父窗口的详细数据；

### 案例

暂缺

## SetEntityFocusRow

设置指定单据体的焦点行。

### 语法

**方法定义：**

|  |  |
| --- | --- |
| C# |  |
| void SetEntityFocusRow(string entityKey, int row); | |

**参数说明**

entityKey：指定单据体标识，

row：指定聚焦的行（下标从0开始）。

**返回类型**

无

**关键字**

选中行 聚焦行 选中某行

**备注**

设置单据体选中行。同时触发Model模型的数据行聚焦以及前端的单据体行聚焦，会触发表单插件的分录行EntityRowClick单击事件。

### 案例 缺省选中单据体首行

**代码来源**

Kingdee.K3.MFG.PRD.Business.PlugIn.Bill.MOBillEdit,

Kingdee.K3.MFG.PRD.Business.PlugIn

**需求背景**

制造行业的生产订单据编辑插件中，在单据加载完数据，开始绑定用户数据时设置单据体缺省选中第一行,打开单据即缺省选中单据体首行，触发行点击事件。

**实现方案**

重写BeforeBindData事件，在事件中设置单据体的焦点行

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.Bill.PlugIn;  using System;  namespace Kingdee.K3.MFG.PRD.Business.PlugIn.Bill  {  public class MOBillEdit: AbstractBillPlugIn  {  public override void BeforeBindData(EventArgs e)  {  base.BeforeBindData(e);  SetConditionedLock();  SetFieldLock();  this.View.SetEntityFocusRow(Consts\_PRD\_MO.FKEY\_ENTITY, 0);  }  }  } | |

## SetFormTitle

设置窗体标题。

### 语法

**方法定义：**

|  |  |
| --- | --- |
| C# |  |
| void SetFormTitle(LocaleValue titleValue); | |

**参数说明**

titleValue：被设置的标题的多语言对象

**返回类型**

无

**备注**

### 案例 动态设置表单标题

**代码来源**

Kingdee.BOS.Business.PlugIn.DataCenter.DataCenterLanguageSetEdit,

Kingdee.BOS.Business.PlugIn

**需求背景**

管理中心，在给具体的数据中心设置语言列表时，需要修改标题明确标识给用户是修改的哪个数据中心。

**实现方案**

在数据中心的设置语言插件中（管理中心->数据中心列表->设置语言），通过SetFormTitle动态设置当前选中行的数据中心语言设置的标题栏，用于明确标识给用户当前设置的数据中心。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.DynamicForm.PlugIn;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using Kingdee.BOS.ServiceHelper;  namespace Kingdee.BOS.Business.PlugIn.DataCenter  {  public class DataCenterLanguageSetEdit : AbstractDynamicFormPlugIn  {  public override void OnInitialize(InitializeEventArgs e)  {  \_dataCenterID = e.Paramter.GetCustomParameter("DataCenterID");  \_dataCenterContext = DataCenterService.GetDataCenterContextByID(\_dataCenterID as string);  var dc = DataCenterService.GetBusinessDataCenterInfo(\_dataCenterID as string);  this.View.SetFormTitle(new LocaleValue(string.Format("{0}[{1}]",  this.View.BusinessInfo.GetForm().Name,  dc.DataCenterName)));  }  }  } | |

## SetInnerTitle

设置界面内嵌标题。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void SetInnerTitle(LocaleValue innerTitleValue); | |

**参数说明**

innerTitleValue：内嵌标题

**返回类型**

无

**关键字**

内嵌标题

### 案例

暂缺

## ShowErrMessage

在界面弹出错误信息，警示用户。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void ShowErrMessage(string msg, string title = "", MessageBoxType msgType = MessageBoxType.Notice); | |

**参数说明**

msg：错误提示信息

title：提示窗体标题

msgType：消息图标类型

**返回类型**

无返回值

**重载方法**

|  |  |
| --- | --- |
| C# |  |
| void ShowErrMessage(Queue<MsgInnerInfo> qLinkButtionMsgInfo, string msg, string title = "", MessageBoxType msgType = MessageBoxType.Notice); | |

**参数说明**

qLinkButtionMsgInfo：链接按钮信息，点击打开链接地址或者给链接设置点击事件

msg：错误提示信息

title：提示窗体标题

msgType：消息图标类型

**返回类型**

无返回值

**重载方法**

|  |  |
| --- | --- |
| C# |  |
| void ShowErrMessage(string msg, Action<MessageBoxResult> action, string title = "", MessageBoxType msgType = MessageBoxType.Notice); | |

**参数说明**

msg：错误提示信息

action：回调处理函数

title：提示窗体标题

msgType：消息图标类型

**返回类型**

无返回值

**关键字**

错误提示

**备注**

错误提示一般为醒目红色

### 案例 销售出库单内部交易单据不允许下推

**代码来源**

供应链插件：销售出库单表单插件

Kingdee.K3.SCM.Sal.Business.PlugIn.OutStockEdit,

Kingdee.K3.SCM.Sal.Business.PlugIn

**需求背景**

点击下推菜单按钮的时候，需要判断该销售出库单是否为内部交易产生的单据，如果是，则不允许下推，并给出错误提示

**实现方案**

重写BarItemClick事件，判断单据来源类型，如果是内部交易产生，终止菜单操作，给出错误提示

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.Bill.PlugIn;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using Kingdee.BOS.Util;  using System;  namespace Kingdee.K3.SCM.Sal.Business.PlugIn  {  public class OutStockEdit : AbstractBillPlugIn  {  public override void BarItemClick(BarItemClickEventArgs e)  {  switch (e.BarItemKey.ToUpperInvariant())  {  case "TBPUSH":  {  object isGenForIOS = this.View.Model.GetValue("FISGENFORIOS");  if (isGenForIOS.IsNullOrEmpty())  {  break;  }  bool IsIOSBill = Convert.ToBoolean(isGenForIOS);  //内部交易生成的单据  if (IsIOSBill)  {  e.Cancel = true;  this.View.ShowErrMessage("内部交易单据不允许下推！");  return;  }  }  break;  //其他case分支处理逻辑在此略去  }  base.BarItemClick(e);  }  }  } | |

## ShowForm

显示新界面。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void ShowForm(DynamicFormShowParameter param); | |

**参数说明**

param：界面显示参数对象。不同的界面类型，需要设置为不同参数类型：

|  |  |
| --- | --- |
| 界面类型 | 参数对象类型 |
| 动态表单 | Kingdee.BOS.Core.DynamicForm.DynamicFormShowParameter |
| 单据维护界面 | Kingdee.BOS.Core.Bill.BillShowParameter |
| 单据列表界面 | Kingdee.BOS.Core.List.ListShowParameter |
| 账表 | Kingdee.BOS.Core.Report.SysReportShowParameter |
| 列表过滤界面 | Kingdee.BOS.Core.CommonFilter.FilterShowParameter |
| 账表过滤界面 | Kingdee.BOS.Core.ReportFilter.SQLReportFilterShowParameter |
| 直接SQL报表 | Kingdee.BOS.Core.Report.SQLReportShowParameter |
| 其他… | 其他… |

**返回类型**

无返回值

**重载方法**

|  |  |
| --- | --- |
| C# |  |
| void ShowForm(DynamicFormShowParameter param, Action<FormResult> action); | |

**参数说明**

param：界面显示参数对象

action：回调处理函数

**返回类型**

无返回值

**关键字**

关闭

### 案例 供应链相关单据选择折扣

**代码来源**

供应链，单据通用插件

Kingdee.K3.SCM.Sal.Business.PlugIn.Common,

Kingdee.K3.SCM.Sal.Business.PlugIn

**需求背景**

供应链，销售订单、销售出库单、销售退货单、发货通知单、退货通知单在折扣上F7打开折扣明细表窗口选择折扣

**实现方案**

插件重写BeforeF7Select事件，判断如果是要选择折扣，则弹出折扣明细列表，供用户选择

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.Bill.PlugIn;  using Kingdee.BOS.Core.CommonFilter;  using Kingdee.BOS.Core.DynamicForm;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using Kingdee.BOS.Core.List;  using Kingdee.BOS.Util;  using Kingdee.K3.Core.SCM.Args;  using System;  namespace Kingdee.K3.SCM.Sal.Business.PlugIn  {  public class Common : AbstractBillPlugIn  {  public override void BeforeF7Select(BeforeF7SelectEventArgs e)  {  if (!CanF7Select(e.FieldKey, e.Row))  {  e.Cancel = true;  return;  }  string filter;  switch (e.FieldKey.ToUpperInvariant())  {  case "FDISCOUNTRATE":  Common.GetDiscountRate(this, priceDiscTaxArgs, e.Row);  break;  ///其他case分支处理逻辑略去  }  }  public static void GetDiscountRate(AbstractBillPlugIn billPlugIn, PriceDiscTaxArgs args, int row)  {  //构建一个列表显示参数对象  ListShowParameter param = new ListShowParameter();  //列表固定过滤参数对象  ListRegularFilterParameter filterparam = new ListRegularFilterParameter();  //添加过滤条件  filterparam.Filter = GetDiscountRateFilter(billPlugIn, args, row);  //需要显示的对象标识，  param.FormId = "SAL\_DiscountListDetail";  //列表的过滤信息  param.ListFilterParameter = filterparam;  param.IsLookUp = true;  //自定义参数  param.CustomParams.Add("SelectedEntryIds", GetDiscEntryId(billPlugIn, args.EntityKey, row));  //显示界面  billPlugIn.View.ShowForm(param, new Action<FormResult>((result) => DoDiscountResult(billPlugIn, args, result, row)));  }  }  } | |

## ShowK3Displayer

显示K3消息显示器。界面上包含一个表格，其列是根据显示参数动态创建的。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void ShowK3Displayer(string title, IEnumerable message,  Action<FormResult> action = null,  string formTitle = "",  string formId = FormIdConst.BOS\_K3Displayer); | |

**方法描述**

显示K3显示器，简单调用，只需要传入列名称（“|”分隔）及显示消息集合（消息内容也是“|”分隔）

**参数说明**

title：显示表格列名称（“|”分隔）

message：显示表格消息（字符串时使用“|”分隔）集合

action：回调函数

formTitle：对话框标题

formId：对话框标识

**返回类型**

无

**重载方法**

|  |  |
| --- | --- |
| C# |  |
| void ShowK3Displayer(K3DisplayerModel displayer,  Action<FormResult> action = null,  string formId = FormIdConst.BOS\_K3Displayer); | |

**方法描述**

显示K3显示器，高级调用（需要自定义显示模型）

**参数说明**

displayer：显示器模型

action：回调函数

formId：对话框标识

**返回类型**

无

**关键字**

K3显示器

### 案例

暂缺

## ShowMessage

界面弹出提示信息，提醒用户。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void ShowMessage(string msg, MessageBoxType msgType =  MessageBoxType.Notice); | |

**参数说明**

msg：提示信息

msgType：消息图标类型

**返回类型**

无返回值

**重载方法**

|  |  |
| --- | --- |
| C# |  |
| void ShowMessage(string msg, MessageBoxOptions options, string title = "", MessageBoxType msgType = MessageBoxType.Notice); | |

**参数说明**

msg：提示信息

options：消息按钮类型

title：消息窗体标题

msgType：消息图标类型

**返回类型**

无返回值

**重载方法**

|  |  |
| --- | --- |
| C# |  |
| void ShowMessage(string msg, MessageBoxOptions options,  Action<MessageBoxResult> action, string title = "", MessageBoxType msgType = MessageBoxType.Notice); | |

**参数说明**

msg：提示信息

options：消息按钮类型

action：回调函数

title：消息窗体标题

msgType：消息图标类型

**返回类型**

无

**重载方法**

|  |  |
| --- | --- |
| C# |  |
| void ShowMessage(Queue<MsgInnerInfo> qLinkButtionMsgInfo, string msg, MessageBoxType msgType = MessageBoxType.Notice); | |

**参数说明**

qLinkButtionMsgInfo：按钮链接信息

msg：提示信息

msgType：消息图标类型

**返回类型**

无

**重载方法**

|  |  |
| --- | --- |
| C# |  |
| void ShowMessage(Queue<MsgInnerInfo> qLinkButtionMsgInfo, string msg, MessageBoxOptions options, string title = "", MessageBoxType msgType = MessageBoxType.Notice); | |

**参数说明**

qLinkButtionMsgInfo：按钮链接信息

msg：提示信息

options：界面按钮类型

title：消息提示界面标题

msgType：消息图标类型

**返回类型**

无

**重载方法**

|  |  |
| --- | --- |
| C# |  |
| void ShowMessage(Queue<MsgInnerInfo> qLinkButtionMsgInfo, string msg, MessageBoxOptions options, Action<MessageBoxResult> action, string title = "", MessageBoxType msgType = MessageBoxType.Notice); | |

**参数说明**

qLinkButtionMsgInfo：按钮链接信息

msg：提示信息

options：界面按钮类型

action：回调函数

title：消息提示界面标题

msgType：消息图标类型

**返回类型**

无

**关键字**

提示

**备注**

### 案例 手工新增的销售出库单不允许选择寄售单据类型

**代码来源**

供应链：销售出库单表单插件

Kingdee.K3.SCM.Sal.Business.PlugIn. OutStockEdit,

Kingdee.K3.SCM.Sal.Business.PlugIn

**需求背景**

销售出库单选择单据类型时，不能让用户选择寄售单据类型，

**实现方案**

重写BeforeUpdateValue事件，判断如果是单据类型字段，则判断新值是否为寄售单据类型，如果是，Cancel掉，并提示用户

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.Bill.PlugIn;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS.ServiceHelper;  using Kingdee.BOS.Util;  using System;  namespace Kingdee.K3.SCM.Sal.Business.PlugIn  {  public class OutStockEdit : AbstractBillPlugIn  {  public override void BeforeUpdateValue(BeforeUpdateValueEventArgs e)  {  #region 价目表/折扣表相关字段改变，决定是否清空价目表/折扣表  switch (e.Key.ToUpperInvariant())  {  case "FBILLTYPEID":  string newbillTypeId = Convert.ToString(e.Value);  string businessType = "";  DynamicObject billTypePara = BusinessDataServiceHelper.LoadBillTypePara(this.Context, "SAL\_OSBillTypeParaSetting", newbillTypeId);  if (billTypePara != null)  {  businessType = Convert.ToString(billTypePara["BussinessType"]);  }  if (businessType.EqualsIgnoreCase(Kingdee.K3.Core.SCM.SalBusinessType.CONSIGNMENT)  && (this.View.OpenParameter.CreateFrom == CreateFrom.Default  || this.View.OpenParameter.CreateFrom == CreateFrom.Copy))  {  this.View.ShowMessage("手工新增的销售出库单不允许选择寄售单据类型！");  }  break;  //其他分支处理在此逻辑略去  }  #endregion  }  }  } | |

## ShowNotificationMessage

显示通知消息

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void ShowNotificationMessage(string msg, string title = "",  MessageBoxType msgType = MessageBoxType.Notice); | |

**参数说明**

msg：消息内容

title：消息标题

msgType：消息对话框类型

**返回类型**

无

**关键字**

通知

### 案例

暂缺

## ShowOperateResult

显示批量操作结果。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void ShowOperateResult(OperateResultCollection operateResults,  string formId = FormIdConst.BOS\_BatchTips);  void ShowOperateResult(OperateResultCollection operateResults,  Action<FormResult> action, string formId = FormIdConst.BOS\_BatchTips); | |

**参数说明**

operateResults：操作结果集

formId：可选，批量信息窗体的子类的标识

action：回调函数

**返回类型**

无

**关键字**

批量操作结果

### 案例

暂缺

## ShowStatusBarInfo

显示状态栏信息。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void ShowStatusBarInfo(string sInfo); | |

**参数说明**

sInfo：信息

**返回类型**

无

**关键字**

状态栏

### 案例

暂缺

## ShowUpload

显示上传文件对话框

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void ShowUpload(string entryKey, string fileType, string tplUrl,  Action<MessageBoxResult, string, string> action); | |

**参数说明**

entryKey：实体唯一标识

fileType：文件类型过滤串

tplUrl：文件存放路径

action：回调函数

**返回类型**

无

**关键字**

上传文件

### 案例

暂缺

## ShowWarnningMessage

显示警告消息

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void ShowWarnningMessage(string msg, string title = "", MessageBoxOptions options = MessageBoxOptions.OK, Action<MessageBoxResult> action = null,  MessageBoxType msgType = MessageBoxType.Advise); | |

**参数说明**

msg：消息

title：标题

options：选项按钮设置

action：回调函数

msgType：消息图标类型

**返回类型**

无

**关键字**

警告消息

### 案例

暂缺

## TryGetFormOperation

尝试获取表单上的操作。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| bool TryGetFormOperation(FormOperationEnum operationType,  out IFormOperation operation); | |

**参数说明**

operationType：操作类型

operation：返回操作对象，不存在时返回空

**返回类型**

是否存在指定的操作对象

**关键字**

表单操作

### 案例

暂缺

## UpdateView

刷新界面数据及状态。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void UpdateView(); | |

**参数说明**

无

**返回类型**

无

**备注**

重新绑定整个界面数据与状态；

注意在AfterBindData事件中，不能调用没有参数的UpdateView方法，否则会造成循环调用，需要传参更新指定的控件视图；

**重载方法**

|  |  |
| --- | --- |
| C# |  |
| void UpdateView(string key); | |

**参数说明**

Key：控件唯一标识

**返回类型**

无

**备注**

刷新指定控件的数据及状态

**重载方法**

|  |  |
| --- | --- |
| C# |  |
| void UpdateView(string fieldKey, int row); | |

**参数说明**

fieldKey：字段唯一标识

row：行索引，当row小于0时,更新的是表格中所有行指定列数据

**返回类型**

无

**备注**

刷新界面指定行控件的数据及状态,适用于表格

**关键字**

更新、视图

### 案例 删除分录，更新视图

**代码来源**

供应链：销售调价表客户端插件

Kingdee.K3.SCM.Sal.Business.PlugIn. SaleAdjustPriceEdit,

Kingdee.K3.SCM.Sal.Business.PlugIn

**需求背景**

价目表编码改变时，删除调价方案和调价结果表体行

**实现方案**

DataChanged中判断，如果是价目表编码改变，则删除调价方案和调价结果表体行

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.Bill.PlugIn;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  namespace Kingdee.K3.SCM.Sal.Business.PlugIn  {  public class SaleAdjustPriceEdit: AbstractBillPlugIn  {  public override void DataChanged(DataChangedEventArgs e)  {  switch (e.Field.Key.ToUpperInvariant())  {  case "FPRICELISTIDHEAD":  this.DeleteEntrys();  break;  ///其他case分支处理逻辑略去  default:  break;  }  }  /// <summary>  /// 价目表编码改变时，删除调价方案和调价结果表体行  /// </summary>  private void DeleteEntrys()  {  this.Model.DeleteEntryData("FADJUSTPRICEPLAN");  this.Model.DeleteEntryData("FADJUSTPRICERESULT");  this.View.Model.CreateNewEntryRow("FADJUSTPRICEPLAN");  this.View.UpdateView("FADJUSTPRICEPLAN");  }  }  } | |

## UpdateViewState

刷新界面状态。

### 语法

**方法定义**

|  |  |
| --- | --- |
| C# |  |
| void UpdateViewState(); | |

**参数说明**

无

**返回类型**

无

**关键字**

刷新界面状态

### 案例

暂缺