动态表单插件示例代码

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 | 齐勇慧  崔彬玉  郑泽林  黎德荣  王洪雷 | 张晋博 | 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

# 插件概要描述

动态表单插件是单据、基础资料、列表、过滤界面等插件的基类，提供了这些模型插件的统一接口，通过这些接口可以对表单编辑样式、列表样式、操作、model数据等进行控制。

请求加载单据的过程插件一般经历以下几个阶段：

1. 请求打开对应表单，服务端构建打开对应单据的网络指令
   1. PreOpenForm
2. 初始化View、Model、为字段添加默认值、将当前页面值刷新到前端控件
   1. OnInitializeService
   2. OnSetBusinessInfo
   3. OnSetLayoutInfo
   4. OnCreateDataBinder
   5. OnInitialize
   6. OnBillInitialize
   7. CreateNewData
   8. BeforeCreateNewEntryRow
   9. AfterCreateNewData
   10. AfterCreateModelData
   11. OnLoad
   12. BeforeBindData
   13. AfterBindData

除以上事件外，本插件提供菜单点击、分录点击等事件，详见以下解析

# 公共属性

## Context

当前登录系统上下文全局信息，如：当前系统语言、用户、组织、数据库类型、客户端连接类型、数据库连接信息等。

## Model

详见《IDynamicFormModel接口使用示例代码》

## View

详见《IDynamicFormView接口使用示例代码》

# 事件

## PreOpenForm

**触发时机**

加载页面之前

**应用场景**

加载页面之前，判断当前页面是否符合展开的特殊业务条件，不符合则可以给出提示信息,并取消打开；也可以指定表单视图、表单标题等动态表单打开参数信息;

**关键字**

取消页面展示、动态表单打开参数

**备注**

无

### 案例 – 根据自定义参数设置表单状态

**代码来源**

Kingdee.K3.BD.Common.Business.PlugIn.AccountTypeEdit

**需求背景**

根据用户自定义参数判断表单当前应为新增状态还是编辑状态，以实现特殊业务处理

**实现方案**

根据自定义参数关键字获取自定义对象，如果该对象已定义并且值为new,那么表单就新增;如果为edit，那么就让该表单处于编辑状态.

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.Bill.PlugIn;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args  namespace Kingdee.K3.BD.Common.Business.PlugIn  {  /// <summary>  /// 科目类别编辑插件  /// </summary>  public class AccountTypeEdit : AbstractBillPlugIn  {  public override void PreOpenForm(PreOpenFormEventArgs e)  {  object openType = e.OpenParameter.GetCustomParameter("OperationStatus");  if (openType != null && openType.ToString().Equals("New", StringComparison.OrdinalIgnoreCase))//自定义参数中存在【操作状态】参数  {  this.operationStatus = OperationStatus.ADDNEW;  e.OpenParameter.Status = this.operationStatus;  }  else if (openType != null && openType.ToString().Equals("Edit", StringComparison.OrdinalIgnoreCase))  {  this.operationStatus = OperationStatus.EDIT;  e.OpenParameter.Status = this.operationStatus;  }  }  }  } | |

## OnInitializeService

**触发时机**

请求加载页面，初始化View和Model之前

**应用场景**

每个页面即View，都持有一个服务对象集合，此对象集合里提供的服务就像工具一样，随时拿来用，如动态表单View层插件代理对象、动态表单Model层插件代理对象、Model对象、View对象、默认值计算器、表达式函数服务、页面状态变化管理器等

**示例代码**

无

## OnSetBusinessInfo

**触发时机**

在请求加载页面之后，渲染界面控件之前触发。

**应用场景**

让插件有机会在界面渲染控件之前，更改元数据，从而实现动态调整界面，增删字段。

**关键字**

修改元数据 动态 添加列 添加字段 添加控件

**备注**

本事件通常需要与OnSetLayoutInfo一起使用，一个修改单据的逻辑元模型(BusinessInfo)，另外一个修改单据的布局元模型(LayoutInfo)。

### 案例 – 动态生成尺码列

**代码来源**

服装行业产品，动态生成尺码列，技术预研插件

**需求背景**

服装行业，服装尺码，配码方案多样且支持动态变化，所含尺码也允许不断调整。

销售单据，录入物料之前，需先选择配码方案，然后在物料明细表格中，把所选的配码方案包含的尺码，动态的显示在表格中，用户直接在这些尺码下面录入数量。各个尺码的数量之和，为该物料的销售总数量；

**实现方案**

在请求加载销售订单界面之后，渲染订单界面字段之前，读取全部配码方案，所含的全部尺码，然后为每个尺码在明细表格中，动态生成一列；

界面加载完毕，用户选择配码方案后，插件控制只显示本配码方案的尺码，其他非本配码方案的尺码，全部隐藏。

在实现方案中，需通过捕获OnSetBusinessInfo事件，动态添加尺码字段的逻辑元数据(Field对象)，这些字段的外观位置，则需要通过捕获OnSetLayoutInfo事件实现。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| /// <summary>  /// 调整单据的逻辑元数据包，动态添加尺码列  /// </summary>  /// <param name="e"></param>  public override void OnSetBusinessInfo(SetBusinessInfoArgs e)  {  // 创建当前单据元数据副本，避免直接修改原始元数据，并发时串账  FormMetadata currmetadata = (FormMetadata)ObjectUtils.CreateCopy(  this.View.OpenParameter.FormMetaData);  \_currBusinessInfo = currmetadata.BusinessInfo;  \_currLayoutInfo = currmetadata.GetLayoutInfo();  // 取物料明细单据体的元数据模型  string currEntityKey = "FEntity";  var currEntity = \_currBusinessInfo.GetEntity(currEntityKey);  // 修改明细单据体采用的表格名称，指向进行了行列转换，包含了各个尺码的视图  currEntity.TableName = "v\_Cust\_JDDynamicColumnEntry";  // 取原始的数量。动态添加的尺码字段，是基于原始数量字段的一个分身  Field qtyField = \_currBusinessInfo.GetField("FQty");  FieldAppearance qtyFldApp = \_currLayoutInfo.GetFieldAppearance("FQty");  // 演示动态添加2个尺寸到单据体：FSize1, FSize2，采用数量字段模板  // 基于数量字段，复制出FSize1字段，并设置必须改动的属性  Field size1Field = (Field)ObjectUtils.CreateCopy(qtyField);  size1Field.DynamicProperty = null;  size1Field.ChildrenFields.Clear();  size1Field.EntityKey = currEntityKey;  size1Field.Entity = currEntity;  if (size1Field.UpdateActions != null)  {  size1Field.UpdateActions.Clear();  }  // 必改属性，涉及到数据的加载  size1Field.Key = "FSize1";  size1Field.FieldName = "FSize1";  size1Field.PropertyName = "FSize1";  size1Field.Name = new LocaleValue("Size1");  // 把FSize1字段，添加到单据的逻辑元数据模型(BusinessInfo)中  \_currBusinessInfo.Add(size1Field);  // 基于数量字段，复制出FSize2字段，并设置必须改动的属性  Field size2Field = (Field)ObjectUtils.CreateCopy(qtyField);  size2Field.DynamicProperty = null;  size2Field.ChildrenFields.Clear();  size2Field.EntityKey = currEntityKey;  size2Field.Entity = currEntity;  if (size2Field.UpdateActions != null)  {  size2Field.UpdateActions.Clear();  }  // 必改属性，涉及到数据的加载  size2Field.Key = "FSize2";  size2Field.FieldName = "FSize2";  size2Field.PropertyName = "FSize2";  size2Field.Name = new LocaleValue("Size2");  // 把FSize2字段，添加到单据的逻辑元数据模型(BusinessInfo)中  \_currBusinessInfo.Add(size2Field);  // 删除仅用于占位的尺码字段  \_currBusinessInfo.Remove(\_currBusinessInfo.GetElement("FSize"));  // 强制要求重新构建单据的ORM模型  \_currBusinessInfo.GetDynamicObjectType(true);  // 输出动态调整后的单据逻辑元数据模型(BusinessInfo)  e.BusinessInfo = \_currBusinessInfo;  e.BillBusinessInfo = \_currBusinessInfo;  } | |

## OnSetLayoutInfo

**触发时机**

在请求加载页面之后，渲染界面控件之前触发。

**应用场景**

让插件有机会在界面渲染控件之前，更改元数据，从而实现动态调整界面，增删字段。

**关键字**

修改元数据 动态 添加列 添加字段 添加控件

**备注**

本事件通常需要与OnSetBusinessInfo一起使用，一个修改单据的逻辑元模型(BusinessInfo)，动态添加字段，另外一个修改单据的布局元模型(LayoutInfo)，设置动态添加字段的位置。

### 案例 – 动态生成尺码列

**代码来源**

服装行业产品，动态生成尺码列，技术预研插件

**需求背景**

服装行业，服装尺码，配码方案多样且支持动态变化，所含尺码也允许不断调整。

销售单据，录入物料之前，需先选择配码方案，然后在物料明细表格中，把所选的配码方案包含的尺码，动态的显示在表格中，用户直接在这些尺码下面录入数量。各个尺码的数量之和，为该物料的销售总数量；

**实现方案**

在请求加载销售订单界面之后，渲染订单界面字段之前，读取全部配码方案，所含的全部尺码，然后为每个尺码在明细表格中，动态生成一列；

界面加载完毕，用户选择配码方案后，插件控制只显示本配码方案的尺码，其他非本配码方案的尺码，全部隐藏。

在实现方案中，捕获OnSetLayoutInfo事件，是为了给动态生成的尺码字段，产生外观位置（FieldAppreance对象）。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| /// <summary>  /// 调整单据的布局元数据，为动态添加的尺码列，产生外观布局对象  /// </summary>  /// <param name="e"></param>  public override void OnSetLayoutInfo(SetLayoutInfoArgs e)  {  // 取数量字段的外观布局对象，尺码需仿照数量外观  FieldAppearance qtyFldApp = \_currLayoutInfo.GetFieldAppearance("FQty");    // 取物料明细表格外观布局对象  EntityAppearance entryApp = \_currLayoutInfo.GetEntityAppearance(qtyFldApp.EntityKey);    // 设置字段的录入顺序  int tabIndex = qtyFldApp.Tabindex;  // 把数量之后的全部字段录入顺序，再后退2个位置，为新增加的字段流出空挡  var fldList = (from p in entryApp.Layoutinfo.Appearances  where (p is FieldAppearance) && ((FieldAppearance)p).Tabindex >= tabIndex  select (FieldAppearance)p );  foreach(var fld in fldList)  {  fld.Tabindex = fld.Tabindex + 2;  }  // 基于数量的外观，产生出FSize1的外观  FieldAppearance size1FieldApp = (FieldAppearance)ObjectUtils.CreateCopy(qtyFldApp);  size1FieldApp.Key = "FSize1";  size1FieldApp.Caption = new LocaleValue("Size1");  size1FieldApp.Field = \_currBusinessInfo.GetField("FSize1");  size1FieldApp.Tabindex = tabIndex;  tabIndex++;  // 把FSize1的外观，加入到单据布局对象中  \_currLayoutInfo.Add(size1FieldApp);  // 基于数量的外观，产生出FSize2的外观  FieldAppearance size2FieldApp = (FieldAppearance)ObjectUtils.CreateCopy(qtyFldApp);  size2FieldApp.Key = "FSize2";  size2FieldApp.Caption = new LocaleValue("Size2");  size2FieldApp.Field = \_currBusinessInfo.GetField("FSize2");  size2FieldApp.Tabindex = tabIndex;  tabIndex++;  // 把FSize2的外观，加入到单据布局对象中  \_currLayoutInfo.Add(size2FieldApp);  // 移除仅用于占位的FSize字段  \_currLayoutInfo.Remove(\_currLayoutInfo.GetAppearance("FSize"));    // 对表格中的字段，重新按照TabIndex进行排序  entryApp.Layoutinfo.Sort();  // 输出动态调整后的单据布局  e.LayoutInfo = \_currLayoutInfo;    // 要求物料明细表格，重新构建各列，以便把新加入的尺码列，也加入到表格控件中  EntryGrid grid = this.View.GetControl<EntryGrid>("FEntity");  grid.SetCustomPropertyValue("AllowLayoutSetting", false);  grid.CreateDyanmicList(\_currLayoutInfo.GetEntityAppearance("FEntity"));  this.View.SendDynamicFormAction(this.View);  } | |

## OnCreateDataBinder

**触发时机**

OnSetLayoutInfo事件之后，OnInitialize事件之前

**应用场景**

插件自定义数据绑定器，数据绑定器的意义在于将Model中的数据生成Json方式的前端控件可识别的网络指令

**代码示例**

无

## OnInitialize

**触发时机**

请求加载页面时触发。此时Model还没初始化，还是个空对象，但View已基本完成初始化

**应用场景**

请求打开一个页面时，可在此事件中根据View打开参数（DynamicFormOpenParameter）和View的信息，初始化当前插件的一些全局属性，也可修改当前页面的控件属性值，如控件大小、字段标题、表单标题等等。

**关键字**

页面初始化 字段标题 表单标题 自定义参数

**备注**

### 案例 – 动态修改字段标题

**代码来源**

参考：Kingdee.K3.SCM.Sal.Business.PlugIn.DiscountDetailEdit

**需求背景**

在打开折扣明显表时，判断折后金额是否含税，如果是含税金额，则“折扣金额”字段的标题显示为“**行折扣含税金额**”，否则标题显示为“**行折扣金额**”

**实现方案**

在页面初始化时，从自定义参数中判断折扣金额是否含税，再根据获取到的参数值动态的设置折扣金额字段的标题。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using Kingdee.BOS.Core.DynamicForm.PlugIn;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args  namespace Kingdee.K3.SCM.Sal.Business.PlugIn  {  /// <summary>  /// 折扣明细表 插件  /// </summary>  public class DiscountDetailEdit : AbstractDynamicFormPlugIn  {  public override void OnInitialize(InitializeEventArgs e)  {  //从自定义参数中获取“是否含税”（该参数在别处赋值，此处假设已有值）  bool includeTax = Convert.ToBoolean(e.Paramter.GetCustomParameter("IncludedTax"));  //如果含税则标识为FEndAmount的控件标题为“行折后含税金额”，否则标题为“行折后金额”  if (includeTax)  {  this.View.GetControl("FEndAmount").Text = "行折后含税金额";  }  else  {  this.View.GetControl("FEndAmount").Text = "行折后金额";  }  }  }  } | |

添加自定义参数方法如下：

|  |  |
| --- | --- |
| C# |  |
| //点击销售订单、销售出库单、销售退货单等表体折扣明细按钮触发打开折扣明细表  DynamicFormShowParameter param = new DynamicFormShowParameter();  //折扣明细表FormId  param.FormId = "SAL\_DISCOUNTDETAIL";  //新增“是否含税”参数  param.CustomParams.Add("IncludedTax", "true");  //打开折扣明细表  this.View.ShowForm(param); | |

## OnBillInitialize

**触发时机**

OnInitialize事件完成后，CreateNewData事件之前

**应用场景**

特别针对单据模型，在此事件中，进行一些插件的初始化操作

**备注**

列表插件无此事件

**示例代码**

无

## CreateNewData

**触发时机**

View已完成初始化，开始进入Model数据对象（Model.DataObject）初始化

**应用场景**

插件自定义Model.DataObject数据包，插件如构建此数据包，即忽略平台构建默认数据包

**关键字**

新增 默认值 动态

**备注**

本事件只有在新增表单情况下才会触发，如加载已存在的单据，不触发此事件

Ben 态表单字段默认值的目的。

### 案例1 – 展示列表操作结果信息的窗体

**代码来源**

参考：Kingdee.BOS.Business.DynamicForm.PlugIn. ShowListBatchTipsPlugIn

**需求背景**

列表操作时，当同时操作多条数据时，为方便展示对每条数据的操作结果信息，会将所有结果在新窗体中显示出来。

**实现方案**

列表操作结果显示窗体只是用于显示操作结果，所以可以在CreateNewData事件中通过从获取的操作结果中给对应的字段赋予默认值。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.DynamicForm;  using Kingdee.BOS.Core.DynamicForm.PlugIn;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.Core.Metadata.EntityElement;  using Kingdee.BOS.Core.Metadata.FieldElement;  using Kingdee.BOS.Orm.DataEntity;  namespace Kingdee.BOS.Business.DynamicForm.PlugIn  {  /// <summary>  /// 列表操作结果显示  /// </summary>  public class ShowListBatchTipsPlugIn : AbstractDynamicFormPlugIn  {  private OperateResultCollection \_results =new OperateResultCollection ();  /// <summary>  /// 创建数据包事件处理；由插件处理数据包的创建过程，界面仅展示  /// </summary>  /// <param name="e"></param>  public override void CreateNewData(BizDataEventArgs e)  {  // 创建本界面需要的数据对象  e.BizDataObject = new DynamicObject(this.View.OpenParameter.FormMetaData.BusinessInfo.GetDynamicObjectType());  BusinessInfo info = this.View.OpenParameter.FormMetaData.BusinessInfo;  // 给表格赋值  Entity resultEntity = info.GetEntity("FEntity");  Field seqField = info.GetField("FSeq");  Field nameField = info.GetField("FName");  Field statusField = info.GetField("FStatus");  Field messageField = info.GetField("FMessage");  Field typeField = info.GetField("FType");  DynamicObjectCollection resultEntityData = (DynamicObjectCollection)resultEntity.DynamicProperty.GetValue(e.BizDataObject);  int row = 0;  //\_results为操作结果集合，此处假设已有值  foreach (OperateResult rowResult in \_results)  {  // 添加新行  DynamicObject rowData = new DynamicObject(resultEntity.DynamicObjectType);  // 给行中的字段赋值  seqField.DynamicProperty.SetValue(rowData, row + 1);  nameField.DynamicProperty.SetValue(rowData, rowResult.Name);  statusField.DynamicProperty.SetValue(rowData, (rowResult.SuccessStatus ? "1" : "0"));  messageField.DynamicProperty.SetValue(rowData, rowResult.Message);  typeField.DynamicProperty.SetValue(rowData, ((int)rowResult.MessageType).ToString());  resultEntityData.Add(rowData);  row++;  }  }  }  } | |

### 案例2 – 结账，反结账动态表单界面初始化

**代码来源**

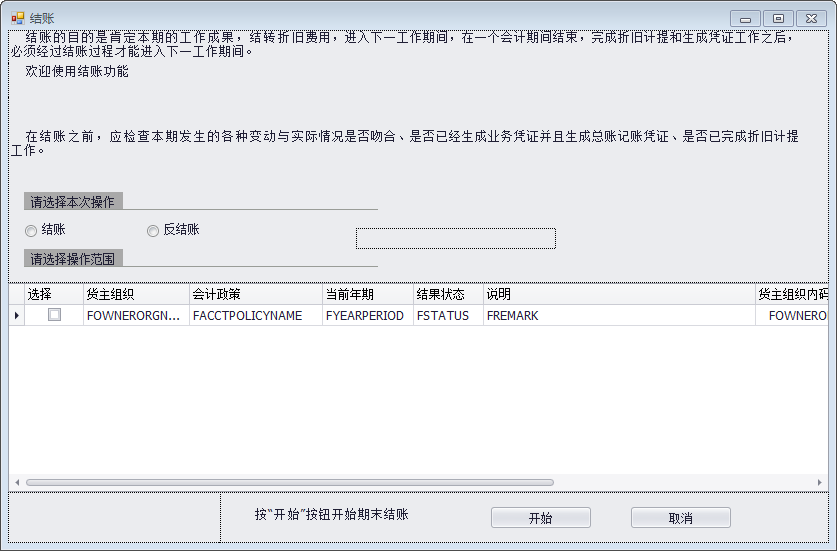
参考：Kingdee.K3.FIN.AP.Business.PlugIn.ClosingAccountEdit

**需求背景**

自定义动态表单，插件完成Model数据对象初始化

**实现方案**

通过BOS IDE配置动态表单，插件代码实现Model数据包的初始化工作



**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using Kingdee.BOS.Core;  using Kingdee.BOS.Core.DynamicForm.PlugIn;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.Core.Metadata.EntityElement;  using Kingdee.BOS.Core.SqlBuilder;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS.Orm.Metadata.DataEntity;  using Kingdee.BOS.ServiceHelper;  /// <summary>  /// 结账，反结账  /// </summary>  public class ClosingAccountEdit : AbstractDynamicFormPlugIn  {  /// <summary>  /// 表单加载时初始化数据对象  /// </summary>  /// <param name="e"></param>  public override void CreateNewData(BizDataEventArgs e)  {  //获得当前表单动态实体类型  DynamicObjectType objType = this.Model.BillBusinessInfo.GetDynamicObjectType();  //获取单据体的动态实体类型  Entity actionEntity = this.View.BusinessInfo.Entrys[1];  //准备单据体数据行动态实体类型  DynamicObjectType actionObjType = actionEntity.DynamicObjectType;  //新建动态表单当前实体对象  DynamicObject dataObject = new DynamicObject(objType);  //获得动态表单单据体实体对象  DynamicObjectCollection entryDataObject = actionEntity.DynamicProperty.GetValue<DynamicObjectCollection>(dataObject);  //读取符合条件的组织机构列表并作为单据体数据加入  List<SelectorItemInfo> selector = new List<SelectorItemInfo>();  selector.Add(new SelectorItemInfo("FORGID"));  selector.Add(new SelectorItemInfo("FName"));  selector.Add(new SelectorItemInfo("FNumber"));  selector.Add(new SelectorItemInfo("FDescription"));  QueryBuilderParemeter para = new QueryBuilderParemeter()  {  FormId = FormIdConst.ORG\_Organizations,  SelectItems = selector,  OrderByClauseWihtKey = "",  IsolationOrgList = null,  RequiresDataPermission = true  };  DynamicObjectCollection orgs = QueryServiceHelper.GetDynamicObjectCollection(this.Context, para);  int seq = 0;  List<Int64> orgList = new List<Int64>();  if (orgs != null)  {  foreach (DynamicObject obj in orgs)  {  DynamicObject settleOrg = new DynamicObject(actionObjType);  settleOrg["Check"] = true;  settleOrg["SettleOrgNo"] = obj["FNumber"].ToString();  settleOrg["SettleOrgName"] = obj["FName"].ToString();  settleOrg["SettleOrgDesc"] = obj["FDescription"].ToString();  settleOrg["SettleOrgID"] = obj["FORGID"].ToString();  settleOrg["Result"] = "";  settleOrg["RetFlag"] = false;  settleOrg["Seq"] = seq++;  entryDataObject.Add(settleOrg);  }  }  e.BizDataObject = dataObject;  }  } | |

## BeforeCreateNewEntryRow

**触发时机**

新增单据分录行数据包前

**应用场景**

插件可以取消本次单据体默认数据行的创建工作，也可以配置默认创建几条行数据，默认是创建1行，也可决定为哪个实体创建默认行数据包

**备注**

新增单据会触发

在单据分录上点新增一行时也会触发

**插件示例**

无

## AfterCreateNewEntryRow

**触发时机**

新增单据分录行数据包后

**应用场景**

新增分录行时，有些字段值可以自动填充，因此在新增分录行之后可以通过该事件根据业务需要为字段赋值。

**关键字**

新增分录 默认值

**备注**

新增单据会触发

在单据分录上点新增一行时也会触发

### 案例 – 新增员工任岗信息

**代码来源**

参考：Kingdee.K3.BD.Common.Business.PlugIn. BDEmpinfoEdit

**需求背景**

员工新增岗位时，如果该员工还没有设置主任岗位，则将新增的岗位默认设置为主任岗位。

**实现方案**

点击新增岗位，在AfterCreateNewEntryRow事件中从当前数据模型中判断该员工是否已经设置主任岗位，如果没设置，则在新增的分录行中设置“是否主任岗”是值true，并且制定刷新该控件。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.Bill.PlugIn;  using Kingdee.BOS.Orm.DataEntity;  using System;  using System.ComponentModel;  using System.Linq;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args  namespace Kingdee.K3.BD.Common.Business.PlugIn  {  [Description("基础新员工单据插件")]  public class BDEmpinfoEdit : AbstractBillPlugIn  {  public override void AfterCreateNewEntryRow(CreateNewEntryEventArgs e)  {  if (e.Entity.Key.ToUpperInvariant() == "FPOSTENTITY")  {  //判断是否已经设置主任岗，如果没有设置，则将该岗位默认设置为主任岗位  if (!((DynamicObjectCollection)this.Model.DataObject["PostEntity"]).Any(c => Convert.ToBoolean(c["IsFirstPost"])))  {  this.Model.BeginIniti();  this.Model.SetValue("FIsFirstPost", true, e.Row);  this.Model.EndIniti();  }  }  }  }  } | |

## AfterCreateNewData

**触发时机**

新增单据场景下，Model数据包创建之后，绑定数据之前触发。

**应用场景**

自定义设置字段默认值。

**关键字**

数据包创建后 默认值

**备注**

通常可以在BOSIDE中设置字段属性默认值，或通过实体服务规则设置默认值，但对于一些特殊需求，无法通过BOSIDE配置完成数值初始化，就需要通过插件实现

### 案例 – 新增单据获取当前组织邮件信息并初始化单据字段

**代码来源**

Open.kingdee.com插件示例

**需求背景**

当前单据包含邮箱信息相关字段，要求在新增单据时自动获取当前组织的邮件信息，包括：邮件类型、发送邮件服务器、SMTP端口号，并赋值给当前新增单据对应字段。

**实现方案**

通过元数据服务根据当前登入组织的ID获取组织邮件信息，再将获取到的邮件类型、发送邮件服务器地址、SMTP端口号赋值给对应的字段。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| /// <summary>  ///新增单据时根据当前组织获取邮件的缺省值，赋值到当前数据包。  /// </summary>  /// <param name="e"></param>  public override void AfterCreateNewData(EventArgs e)  {  //添加过滤条件  OQLFilter ofilter = new OQLFilter();  ofilter.Add(new OQLFilterHeadEntityItem { FilterString = string.Format(" FORGID ={0} ", this.Model.Context.CurrentOrganizationInfo.ID) });  //获取当前组织邮件信息  DynamicObject[] obj = BusinessDataServiceHelper.Load(this.View.Context, "BAS\_MAILDEFAULTSET", null, ofilter);  if (obj != null && obj.Count() > 0)  {  DynamicObject defaultSet = obj[0];  //设置当前组织邮件服务器  this.View.Model.SetValue("FMessageType", defaultSet["FMessageType"]);  this.View.Model.SetValue("FServer", defaultSet["FOutgoingMailServer"]);  this.View.Model.SetValue("FSMTPPort", defaultSet["FSMTPPort"]);  }  } | |

## AfterCreateModelData

**触发时机**

Model数据初始化完成之后

**应用场景**

新增状态下,在页面模型创建完毕，对其中某些字段模型字段设置默认值

**关键字**

给模型字段赋值、默认值

**备注**

只有在新增单据情况下，才会触发此事件

### 案例 – 促销申请单设置币别默认值

**代码来源**

Kingdee.K3.SCM.SPM.Business.Plugin.SPMPromotionApplyEdit

**需求背景**

没有设置币别的默认值，需要根据费用承担公司获取对应的币别来设置默认值.

**实现方案**

首先判断有没有设置，如果没设置就根据费用承担公司获取对应的币别来设置.

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.Bill.PlugIn;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.K3.SCM.Business;  namespace Kingdee.K3.SCM.SPM.Business.Plugin  {  [Description("促销申请单表单插件")]  /// <summary>  /// 促销申请单表单插件  /// </summary>  public class SPMPromotionApplyEdit : AbstractBillPlugIn  {  /// <summary>  /// 动态表单创建之后  /// </summary>  /// <param name="e"></param>  public override void AfterCreateModelData(EventArgs e)  {  if (this.Model.OpenParameter.Status == OperationStatus.ADDNEW)  {  DynamicObject currency = this.View.Model.GetValue("FCurrencyID") as DynamicObject;  if (currency == null || Convert.ToInt64(currency["Id"]) == 0)  {  // 设置币别的默认值  DynamicObject salOrg =  this.View.Model.GetValue("FApplyOrgID") as DynamicObject;  long salOrgID = 0;  if (salOrg != null)  {  salOrgID = Convert.ToInt64(salOrg["Id"]);  JSONObject currInfo = CommonServiceHelper.GetDefCurrencyAndExchangeTypeByBizOrgID(this.Context, salOrgID);  if (currInfo != null)  {  this.View.Model.SetValue("FCurrencyID", Convert.ToInt32(currInfo["FCyForID"]));//取帐薄  }  else  {  this.View.Model.SetValue("FCurrencyID", null);  }  }  }  }  }  }  } | |

## OnLoad

**触发时机**

完成View和Model的初始化工作，准备把Model里的数据组织成Json指令并发送给客户端控件之前

**应用场景**

处理一些特殊业务应用场景

**关键字**

页面加载 加载 触发事件

**备注**

### 案例 – 加载单据触发单据体双击事件

**代码来源**

Kingdee.K3.FIN.CR.Business.PlugIn.TreeNodeList

**需求背景**

当动态表单页面加载后，模拟双击分录事件，打开某张单据。

**实现方案**

覆写OnLoad页面加载事件，触发分录双击，打开单据界面。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using Kingdee.BOS.Core.DynamicForm.PlugIn;  using Kingdee.BOS.Core.DynamicForm.PlugIn.ControlModel;  using Kingdee.BOS.Util;  using Kingdee.BOS.Core.DynamicForm;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using Kingdee.K3.FIN.CR.Business.PlugIn;  using Kingdee.K3.FIN.CR.Common.Core;  using Kingdee.K3.FIN.CR.Common.BusinessEntity;  /// <summary>  /// 合并控制器，节点信息  /// </summary>  public class TreeNodeList : AbstractDynamicFormPlugIn  {  public override void OnLoad(EventArgs e)  {  this.View.GetControl<EntryGrid>("FEntity").SetFireDoubleClickEvent(true);  }  public override void EntityRowDoubleClick(EntityRowClickEventArgs e)  {  //双击时，展示对应表单  string nodeId = this.View.Model.GetValue("FNodeId", e.Row).ToString();  string parameter = this.View.Model.GetValue("FParameter", e.Row) as string;  string formType = this.View.Model.GetValue("FFormType", e.Row) as string;  string formId = this.View.Model.GetValue("FFormID", e.Row) as string;  if (!ObjectUtils.IsNullOrEmptyOrWhiteSpace(formId))  {  ReportManageInfo \_reportManagerInfo = this.View.OpenParameter.GetCustomParameter(CRConst.Session\_ReportManageInfo) as ReportManageInfo;  DynamicFormShowParameter param = CRCommonFunc.BuildShowFormParameter4Tree(  "CR",  nodeId,  formId,  formType,  parameter);  param.CustomComplexParams.Add(CRConst.Session\_ReportManageInfo, \_reportManagerInfo);  param.OpenStyle.ShowType = ShowType.InCurrentForm;  //如果只是合并控制用到  this.View.Close();  this.View.ShowForm(param);  }  } | |

## BeforeBindData

**触发时机**

OnLoad事件之后，正式开始准备绑定数据之前

**应用场景**

正式开始准备绑定数据前，处理控件的属性值，确保发送到客户端的控件值是根据特殊业务场景下定义的特殊值

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.Cloud;  using Kingdee.BOS.Core.DynamicForm;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using Kingdee.BOS.Core.DynamicForm.PlugIn.ControlModel;  using Kingdee.BOS.Core.Metadata.EntityElement;  using Kingdee.BOS.JSON;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.K3.SCM.Cloud.Business;  using Kingdee.K3.SCM.Core.Cloud;  using Kingdee.K3.SCM.ServiceHelper;  using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Linq;  using System.Text;  namespace Kingdee.K3.SCM.Cloud.Business.PlugIn  {  [Description("首页\_内容\_待办事宜基类Edit")]  public class WaitingDoBaseEdit : BaseEdit  {  protected const string SelectedBtnColor = "#ffffff";  public override void BeforeBindData(EventArgs e)  {  this.View.GetControl("FEntity").SetCustomPropertyValue("IsListMultiSel", false);  JSONObject selectedJson = new JSONObject();  selectedJson["color"] = SelectedBtnColor;  selectedJson["background-color"] = SelectedBtnBackColor;  this.View.GetControl(TapBtnOne).SetCustomPropertyValue("InlineStyle", selectedJson);  this.View.GetControl("FLABELSPLITE").Visible = false;  }  }  } | |

## AfterBindData

**触发时机**

控件绑定数据之后触发

**应用场景**

在控件绑定数据之后，通常需要根据其他条件设置控件的可见性、可用性、背景颜色等样式。

**关键字**

绑定数据 可见性 可用性

**备注**

### 案例 – 根据类型设置字段可见性

**代码来源**

Open.kingdee.com—〉客户服务—〉开发基础—〉单据插件—〉数据绑定

**需求背景**

动态表单打开时根据类型设置字段的可见性，如果当前打开表单是基础资料，则将隐藏字段参数页签；如果当前打开表单为单据，且该单据中包含单据类型字段，则显示字段参数页签。

**实现方案**

在初始化事件中获取当前业务对象类型，如果当前业务对象类型为基础资料则直接隐藏字段参数页签，如果为单据，则进一步判断该单据中是否包含单据类型字段。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| public override void AfterBindData(EventArgs e)  {  //基础资料  if (\_modelTypeId == ElementType.ELEMENTTYPE\_BASE.ToString())  {  this.View.StyleManager.SetVisible("FTab\_Field", null, false);  }  else if (\_modelTypeId == ElementType.ELEMENTTYPE\_BILL.ToString()) //业务单据  {  //单据不含单据类型字段时，字段参数页签屏蔽  if (this.\_metaData.GetLayoutInfo().GetFieldAppearances().Any(f => f is BillTypeFieldAppearance))  {  this.View.StyleManager.SetVisible("FTab\_Field", null, true);  }  }  } | |

## BeforeClosed

**触发时机**

页面关闭之前触发。

**应用场景**

对于多个表单交互或者嵌入表单，在关闭窗体时需要更新其他表单内容，或者窗体关闭之前自定义提示信息。

**关键字**

关闭前 刷新

**备注**

对于单个表单关闭，该插件基本不需要处理，一般是用于多表单交互或嵌入式表单（父子表单关系，即在A单上点击操作打开B单，即A单是B单是父表单）

### 案例 – 关闭时刷新父窗体

**代码来源**

Kingdee.K3.SCM.Sal.Business.PlugIn.SalReceiveBill

**需求背景**

在弹出的新窗体中选择收款单。

**实现方案**

窗体关闭之前，获取到当前窗体中收款单列表，遍历取出标记为已被选中的收款单，通过this.View.ReturnToParentWindow把子表单的数据返回给父窗体。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.DynamicForm.PlugIn;  using Kingdee.BOS.Core.Metadata.EntityElement;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  namespace Kingdee.K3.SCM.Sal.Business.PlugIn  {  //收款单列表插件  public class SalReceiveBill : AbstractDynamicFormPlugIn  {  public override void BeforeClosed(BeforeClosedEventArgs e)  {  //获取收款单列表  Entity entity = this.View.BusinessInfo.GetEntity("FSalReceive");  DynamicObjectCollection objs = this.View.Model.GetEntityDataObject(entity);  DynamicObjectCollection objReturn = new DynamicObjectCollection(entity.DynamicObjectType);  //遍历收款单列表，如果被选中则添加返回数据中  foreach (DynamicObject dy in objs)  {  if (dy["Check"].ToString().ToUpper() == "TRUE")  {  objReturn.Add(dy);  }  }  //将数据返回到父窗体中  this.View.ReturnToParentWindow(objReturn);  }  }  } | |

## ToolBarItemClick

**触发时机**

工具栏菜单单击事件

**应用场景**

工具栏菜单单击事件，判断数据完整性与合法性，执行对应逻辑。（与AfterToolBarItemClick 区别， ToolBarItemClick还没进入业务逻辑，AfterToolBarItemClick顾名思义，菜单按钮点击并完成相应业务逻辑后）

**关键字**

工具栏菜单 单击处理

**备注**

### 案例 –判断树形控件当前节点是否存在下级子节点，决定能否删除

**代码来源**

Kingdee.BOS.Business.PlugIn.PublishNeedsList

**需求背景**

实现对树形节点的删除

**实现方案**

首先判断是否做了删除的动作，如果删除并且有子节点就不想允许删除.

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| public override void ToolBarItemClick(BarItemClickEventArgs e)  {  Dictionary<string, string> dctBaritem = new Dictionary<string, string>();  dctBaritem["tbDelete"] = "tbDelete";  if (dctBaritem.ContainsKey(e.BarItemKey) == true && FindNode())  {  this.View.ShowErrMessage(“当前节点存在下级节点,不允许删除.”);  e.Cancel = true;  }  } | |

## BarItemClick

**触发时机**

主菜单单击时触发

**应用场景**

通常动态表单在设计时主菜单都会配置平台内置提供的服务，有时有些用户可能需要在调用平台内置功能前进行权限验证、数据检查等等，用户也可以在该事件中通过添加e.Cancel=true取消平台内置功能，完全自定义处理逻辑.

**关键字**

主菜单 单击 权限验证 数据检查 自定义处理

**备注**

### 案例 – 点击查看单据时进行权限验证

**代码来源**

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

**需求背景**

采购订单维护界面主菜单中有“预付查询”菜单项，对于具有付款单的查看权限的用户点击之后会自动加载付款单明细列表，对于没有查看权限的用户会提示“没有付款单的查看权限，不能查询”。

**实现方案**

在动态表单插件中重写BarItemClick方法，主菜单单击时触发该事件，根据参数中包含的当前菜单项主键判断是否点击“预付查询”，如果点击“预付查询”，则将调用权限验证服务，检查当前用户是否具有查看付款单权限，如果具有该权限，则加载对应的付款单明细列表，否则弹出提示，并取消平台内置功能。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.Bill.PlugIn;  using Kingdee.BOS.Core.Permission;  using Kingdee.BOS.ServiceHelper;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  namespace Kingdee.K3.SCM.Purchase.Business.PlugIn  {  /// <summary>  /// 采购订单 单据维护界面插件  /// </summary>  public class PurchaseOrderEdit : AbstractBillPlugIn  {  /// <summary>  /// 主菜单单击时权限判断  /// </summary>  /// <param name="e"></param>  public override void BarItemClick(BarItemClickEventArgs e)  {  switch (e.BarItemKey.ToUpperInvariant())  {  case "TBPAYSEARCH":  //获取权限验证结果  PermissionAuthResult iResult = PermissionServiceHelper.FuncPermissionAuth(this.Context, new BusinessObject() { Id = "AP\_PAYBILL" }, PermissionConst.View);  if (!iResult.Passed)  {  this.View.ShowMessage("没有付款单的查看权限，不能查询");  //插件中要取消内置操作，必须设置Cencel为true  **e.Cancel = true;**  return;  }  //展示付款单列表（具体实现省略）  this.ShowPayList();  break;  default:  break;  }  }  }  } | |

## AfterBarItemClick

**触发时机**

主菜单单击后触发

**应用场景**

通常动态表单在设计时主菜单都会配置平台内置提供的服务，在调用平台提供的服务完成特定的功能之后，用户可以根据自己的实际需求自定义后续的处理，例如：指定跳转页面、指定刷新页面、指定关闭页面等等；如果在设计时没有配置任何服务，则可以在该事件中完全自定义实现逻辑。

**关键字**

主菜单 单击后 跳转页面 刷新页面 关闭页面

**备注**

### 案例 – 序列号扫描结果返回

**代码来源**

Kingdee.K3.SCM.Sal.Business.PlugIn.QueryEstimateProfitEdit

**需求背景**

序列号扫描界面，点击确定时将扫描到的序列号返回到上级页面，并且关闭当前页面。

**实现方案**

在动态表单插件中重写AfterBarItemClick方法，主菜单单击后时触发该事件，根据参数中包含的当前菜单项主键判断如果点击“确定”，则将获取到的序列号携带到父窗体；如果点击“退出”，则直接关闭当前窗体。

**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;  using Kingdee.K3.BD.Business.SerialInput;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  namespace Kingdee.K3.BD.Business.PlugIn  {  /// <summary>  /// 序列号扫描界面插件  /// </summary>  [Description("序列号扫描界面插件")]  public class SerialScanerEditPlugIn : AbstractDynamicFormPlugIn  {  //声明一个集合用于保存序列号  private List<SimpleSerialSnap> \_serials = new List<SimpleSerialSnap>();  public override void AfterBarItemClick(AfterBarItemClickEventArgs e)  {  switch (e.BarItemKey.ToUpper())  {  case "TBOK":  //点击确定将序列号携带到父窗体（\_serials在其他方法中已被赋值，此处假设已有值）  this.View.ReturnToParentWindow(new FormResult(\_serials));  //关闭当前窗体  this.View.Close();  break;  case "TBEXIT":  this.View.Close();  break;  default:  break;  }  }  }  } | |

## ButtonClick

**触发时机**

按钮点击时触发

**应用场景**

按钮点击事件，完全是由插件决定具体实现什么业务逻辑，BOS平台不做任何干预

**关键字**

按钮 点击 数据检查

**备注**

### 案例1 –点击按钮弹出选择日期的表单

**代码来源**

参考：Kingdee.K3.SCM.Business.DynamicForm.PlugIn.DateConfirmEdit

**需求背景**

录入日期时弹出选择日期窗口，在弹出的窗口中进行日期选择并返回数据。

**实现方案**

弹出窗口，捕获当前点击的按钮主键，判断如果是点击“确认”按钮则先对日期字段进行非空判断，若不为空在将选中的日期返回到上级表单，并关闭当前窗口；如果点击“取消”按钮，则直接关闭当前窗口。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using Kingdee.BOS.Core.DynamicForm.PlugIn;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  namespace Kingdee.K3.SCM.Business.DynamicForm.PlugIn  {  /// <summary>  /// 日期选择确认窗体，可用于弹出动态表单进行日期选择并返回  /// </summary>  public class DateConfirmEdit : AbstractDynamicFormPlugIn  {  public override void ButtonClick(ButtonClickEventArgs e)  {  switch (e.Key.ToUpper())  {  case "FBTNCONFIRM":  //点击确认时判断是否选择日期  object dtObj = this.View.Model.GetValue("FDate");  if (dtObj == null)  {  this.View.ShowMessage("请选择日期");  return;  }  DateTime dt = (DateTime)dtObj;  //将选择的日期返回到父窗体中  this.View.ReturnToParentWindow(dt);  this.View.Close();  break;  case "FBTNCANCEL":  this.View.Close();  break;  default:  break;  }  }  }  } | |

### 案例2 –在公司的移动列表中，单击按钮打电话

**代码来源**

Kingdee.BOS.Mobile.FormPlugIns.html5.CompanyInfoEdit.cs

**需求背景**

客户详情的界面上，点击按钮用来拨打电话

**实现方案**

实现按钮点击事件，判断是否FimgCall这个标志，如果是的话，说明对应按钮成功，获取客户的联系信息，获取客户的手机或者移动电话号码，再去执行添加数据的操作，返回给客户端。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System.Linq;  using System.Text;  using Kingdee.BOS.Mobile.PlugIn;  using Kingdee.BOS.Core.Const;  using Kingdee.BOS.JSON;  using Kingdee.BOS.ServiceHelper;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS.Util;  namespace Kingdee.BOS.Mobile.FormPlugIns.html5  {  public class CompanyInfoEdit:AbstractMobileBillPlugin  {  public override void ButtonClick(Core.DynamicForm.PlugIn.Args.ButtonClickEventArgs e)  {  if (e.Key.EqualsIgnoreCase("FImgCall"))  {  DynamicObject dy = GetDataObject();  DynamicObject contactDy = (dy["CompanyContact"] as DynamicObjectCollection)[0];  //移动电话  string Tel\_mobile = Convert.ToString(contactDy["ContactMobile"]);  if (!Tel\_mobile.IsNullOrEmptyOrWhiteSpace())  {  JSONArray jsonArray = new JSONArray();  jsonArray.Add(Tel\_mobile);  this.View.AddAction(JSAction.SendCall, jsonArray);  }  }  }  } | |

## AfterButtonClick

**触发时机**

按钮点击后触发

**应用场景**

按钮点击之后，实现按钮点击功能之后对界面进行控制，如：设置控件可用性、可见性，设置分录焦点行等。

**关键字**

按钮点击 点击后 界面控制 设置焦点

**备注**

### 案例 – 设置按钮可用性

**代码来源**

参考：Kingdee.K3.FIN.GL.Business.PlugIn.InitBalance.InitBalExportEdit

**需求背景**

科目初始数据模板引出界面，点击“上一步”按钮，并且当前步骤为向导的第一步，需要将“下一步”按钮设置为可用状态，允许用户点击“下一步”按钮，继续完成引出数据。

**实现方案**

在AfterButtonClick事件中获取当前点击的按钮主键，如果点击是“上一步”按钮，则进一步判断当前是否为向导的第一步，如果是便将“下一步”按钮的可用性设置为true。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using Kingdee.BOS.Core.DynamicForm.PlugIn.WizardForm;  using Kingdee.BOS.Util;  using System.ComponentModel;  namespace Kingdee.K3.FIN.GL.Business.PlugIn.InitBalance  {  /// <summary>  /// 科目初始数据模板引出 界面插件  /// </summary>  [Description("科目初始数据模板引出 界面插件")]  public class InitBalExportEdit : AbstractWizardFormPlugIn  {  /// <summary>  /// 将要切换到的步骤顺序号  /// </summary>  private int \_curStep = 0;  public override void AfterButtonClick(AfterButtonClickEventArgs e)  {  //点击“上一步”按钮之后，设置“下一步”按钮可用  //\_curStep在其他事件中会被改变，此处假设其值符合条件  if (e.Key.EqualsIgnoreCase("FPrevious") && \_curStep == 0)  {  this.View.StyleManager.SetEnabled("FNext", null, true);  }  }  } | |

## AfterDeleteRow

**触发时机**

分录行删除后触发

**应用场景**

删除分录行之后，根据业务需要修改字段值或控件可见性、可用性，或级联修改其他相关信息。

**关键字**

删除行 分录行 字段值 可见性 可用性

**备注**

无

### 案例 – 删除分录后更新单据头字段

**代码来源**

参考：Kingdee.K3.FIN.SC.Business.PlugIn.ScBillEditBase

**需求背景**

资金管理单据中，单据头有字段用于记录资金笔数，新增一条资金记录或是删除，该字段及时更新。

**实现方案**

删除分录时，重新获取该分录行数量，更新到指定单据头字段中。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.Bill.PlugIn;  using System.ComponentModel;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  namespace Kingdee.K3.FIN.SC.Business.PlugIn  {  /// <summary>  /// 资金管理单据插件基类:  /// </summary>  [Description("资金管理单据插件基类")]  public abstract class ScBillEditBase : AbstractBillPlugIn  {  /// <summary>  /// 汇总分录行数至表头总笔数字段  /// </summary>  /// <param name="e"></param>  public override void AfterDeleteRow(AfterDeleteRowEventArgs e)  {  //获取分录行数量  long count = this.View.Model.GetEntryRowCount(e.EntityKey);  //设置总笔数的值  this.View.Model.SetValue("FCount", count);  this.View.UpdateView("FCount");  }  }  } | |

## EntryBarItemClick

**触发时机**

分录菜单单击

**应用场景**

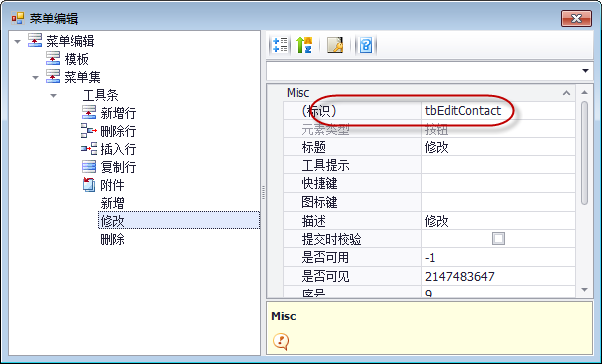
处理分录菜单单击，实现对分录做新增、修改、删除等。

**关键字**

分录菜单 单据体菜单 菜单单击 菜单

**备注**

通过菜单的标识Key进行配合



对比AfterEntryBarItemClick，EntryBarItemClick更加侧重当前需要处理的业务逻辑，AfterEntryBarItemClick偏重于后续需要处理的业务逻辑。

### 案例 – 修改客户联系人页签

**代码来源**

Kingdee.K3.BD.Common.Business.PlugIn.CustomerEdit

**需求背景**

客户联系人有几处相同的操作配置。集团客户(BD\_GroupCustomer)、客户(BD\_Customer)、客户(CRM\_MOBILE\_CUSTOMER)、我的客户(Sal\_MobileCustomer)。需要达到操作业务逻辑的一致性。

**实现方案**

对客户联系人的操作统一封装到一个插件。不同处的客户联系人配置挂接同一份插件代码，达到控制逻辑的一致性。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  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.Core.DynamicForm;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.K3.BD.ServiceHelper;  public class CustomerEdit : AbstractBillPlugIn  {  public override void EntryBarItemClick(BarItemClickEventArgs e)  {  switch (e.BarItemKey.ToUpperInvariant())  {  case "TBNEWCONTACT": // 联系人页签新增联系人  break;  case "TBEDITCONTACT": // 联系人页签修改联系人  int iRowIndex = this.View.Model.GetEntryCurrentRowIndex("FT\_BD\_CUSTLOCATION");  if (iRowIndex < 0)  {  this.View.ShowErrMessage("请选择一个分录数据！");  }  else  {  BillShowParameter BillEditParam = new BillShowParameter();  BillEditParam.FormId = Kingdee.K3.Core.SCM.SCMFormIdConst.BD\_CommonContact;  BillEditParam.Status = OperationStatus.EDIT;  BillEditParam.PKey = Convert.ToString(this.View.Model.GetValue("FContactId", iRowIndex));  BillEditParam.ParentPageId = this.View.PageId;  BillEditParam.CustomParams.Add("CompanyType", "BD\_Customer");  BillEditParam.CustomParams.Add("CustomerId", Convert.ToString(this.View.Model.GetPKValue()));  BillEditParam.CustomParams.Add("IsModify", "true");  this.View.ShowForm(BillEditParam, new Action<FormResult>((result) =>  {  if (result.ReturnData != null)  {  List<long> listContractIds = result.ReturnData as List<long>;  long lContactId = 0;  if (listContractIds != null & listContractIds.Count > 0)  {  lContactId = listContractIds[0];  }  if (lContactId > 0)  {  DynamicObjectCollection docContacts = SupplierCustomerServiceHelper.GetContactsByCompanyType(  this.Context, "BD\_Customer", new long[] { lContactId });  this.View.Model.SetValue("FCONTACT", docContacts[0]["FNAME"], iRowIndex);  this.View.Model.SetValue("FJob", docContacts[0]["FPOST"], iRowIndex);  this.View.Model.SetValue("FBIZLOCATION", docContacts[0]["FBIZLOCATION"], iRowIndex);  this.View.Model.SetValue("FOFFICEPHONE", docContacts[0]["FTEL"], iRowIndex);  this.View.Model.SetValue("FMOBILEPHONE", docContacts[0]["FMOBILE"], iRowIndex);  this.View.Model.SetValue("FFAX1", docContacts[0]["FFAX"], iRowIndex);  this.View.Model.SetValue("FContactEmail", docContacts[0]["FEMAIL"], iRowIndex);  this.View.Model.SetValue("FContactId", docContacts[0]["FCONTACTID"], iRowIndex);  this.View.UpdateView("FT\_BD\_CUSTLOCATION");  }  }  }));  }  break;  case "TBDELETECONTACT": // 联系人页签删除联系人  break;  case "TBDELLINE\_OORG": // 订货组织页签删除默认订货组织时  break;  case "TBADDLINE\_OORG":  break;  default:  break;  }  }  } | |

## AfterEntryBarItemClick

**触发时机**

分录菜单单击处理后触发。

**应用场景**

分录菜单点击后触发事件，例如对分录做删除、设置、更新等点击后。

**关键字**

分录菜单 单击后 单击

**备注**

需要和EntryBaItemClick区别，EntryBarItemClick注重单击单据体菜单的事件处理，如上下移动、删除分录等，而AfterEntryBarItemClick指事件处理后，分录关联数据的处理，如更新状态，页面跳转，提示信息等。

### 案例1 –单据体菜单删除

**代码来源**

Kingdee.BOS.Business.PlugIn.Organizations

**需求背景**

单据体菜单分录菜单点击后，把源单据体的数据同步到目标单据体中

**实现方案**

调用AfterEntryBarItemClick事件，在事件里对baritem的key做判断，如果是对应的key则执行对应的操作。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Data;  using System.Collections.Generic;  using System.Text;  using Kingdee.BOS.Util;  using System.Linq;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using Kingdee.BOS.Core.Bill.PlugIn;  /// <summary>  /// 单据体菜单事件  /// </summary>  /// <param name="e"></param>  public override void AfterEntryBarItemClick(AfterBarItemClickEventArgs e)  {  if (e.BarItemKey != "tbDelEntry") return;  if (e.ParentKey.EqualsIgnoreCase("FBizrelationEntry")) UpdateEntity(EntityRefId.RelationOrgID, EntityRefId.RelationOrgID1);  else if (e.ParentKey.EqualsIgnoreCase("FBizrelationEntry1")) UpdateEntity(EntityRefId.RelationOrgID1, EntityRefId.RelationOrgID);  } | |

## BeforeF7Select

**触发时机**

点击F8按钮或模糊查询时触发

**应用场景**

用户用过F8按钮或模糊查询选择基础资料时，希望列表或模糊查询结果中显示满足条件的基础资料，此时可以通过该事件在加载基础资料数据前添加过滤条件。

**关键字**

基础资料 选择 过滤

**备注**

### 案例1 – 选择基础资料时添加过滤

**代码来源**

Kingdee.BOS.Business.PlugIn.Implementation.BAS\_ERPImplProjectEditPlugin

**需求背景**

ERP项目实施向导中，选择组织时排除已选中的组织，选择用户时排除管理员用户。

**实现方案**

F8按钮选择基础资料时，判断如果选择的是组织，则先从已选择的组织取出组织ID，再将这些ID添加到过滤条件中，加载组织列表时就会过滤掉这些已选中的组织；如果选择的是用户，则在过滤条件中添加排除为管理员的用户。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core;  using Kingdee.BOS.Core.DynamicForm.PlugIn.WizardForm;  using Kingdee.BOS.Orm.DataEntity;  using System;  using System.Collections.Generic;  using System.ComponentModel;  using Kingdee.Core.DynamicForm.PlugIn.Args;  namespace Kingdee.BOS.Business.PlugIn.Implementation  {  [Description("ERP项目实施向导插件")]  public class BAS\_ERPImplProjectEditPlugin : AbstractWizardFormPlugIn  {  public override void BeforeF7Select(BeforeF7SelectEventArgs e)  {  switch (e.FieldKey)  {  case "FOrgId":  //过滤掉已选中的组织  string strFilter = string.Empty;  List<long> orgIds = new List<long>();  DynamicObjectCollection orgObjs = this.Model.GetEntityDataObject(this.Model.BusinessInfo.GetEntity("FChooseOrgEntry")) as DynamicObjectCollection;  foreach (DynamicObject orgObj in orgObjs)  {  long orgId = Convert.ToInt64(orgObj["OrgId\_Id"]);  if (orgId > 0) orgIds.Add(orgId);  }  if (orgIds.Count > 0)  strFilter = string.Format(" FORGID not in ({0}) ", string.Join(", ", orgIds));  e.ListFilterParameter.Filter = strFilter;  break;  case "FUserId":  //过滤出非管理员的用户  e.ListFilterParameter.Filter = " FUserID <> " + FormConst.AdministratorID;  break;  }  }  }  } | |

## BeforeUpdateValue

**触发时机**

更新字段值前触发

**应用场景**

界面字段值更新时允许用户对即将录入的值做检查

**关键字**

值更新 数据检查 改变值 值改变 更新前 字段值更新 合法性 触发值更新

**备注**

对于控件想马上触发值更新事件，BOSIDE需要选中“即时触发值更新事件”属性，否则事件延后直至触发值更新事件

### 案例 – 检查输入日期是否合法

**代码来源**

Kingdee.K3.SCM.Purchase.Business.PlugIn.PurchaseRequisitionEdit

**需求背景**

采购申请单中包含“到货日期”和“建议采购日期”，录入的“到货日期”不能小于“建议采购日期”。

**实现方案**

首先“到货日期”字段需要勾选“及时触发值更新事件”，然后在BeforeUpdateValue事件捕获到该字段新录入的值，将获取的值和“建议采购日期”的值进行比较，如果前者小于后者，则直接弹出提示，并通过e.Cacel=-true取消平台内置的修改，否则成功修改“到货日期”的值。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.Bill.PlugIn;  using Kingdee.BOS.Resource;  using System;  using Kingdee. BOS.Core.DynamicForm.PlugIn.Args;  namespace Kingdee.K3.SCM.Purchase.Business.PlugIn  {  /// <summary>  /// 采购申请单客户端插件  /// </summary>  public class PurchaseRequisitionEdit : AbstractBillPlugIn  {  public override void BeforeUpdateValue(BeforeUpdateValueEventArgs e)  {  switch (e.Key.ToUpper())  {  //更新到货日期  case "FARRIVALDATE":  DateTime arrivalDate = Convert.ToDateTime(e.Value);  DateTime suggestDate = Convert.ToDateTime(this.View.Model.GetValue("FSuggestPurDate", e.Row));  TimeSpan ts = arrivalDate - suggestDate;  if (ts.Days < 0)  {  this.View.ShowMessage("到货日期必须大于或等于建议采购日期");  //取消平台内置功能  e.Cancel = true;  }  break;  default:  break;  }  }  }  } | |

## CustomEvents

**触发时机**

客户端发送请求数据到服务端，处理当前已提供的通信机制外的其他事件

**应用场景**

客户端自定义事件，用于动态表单客户端插件和服务端插件进行通信

**关键字**

自定义 通信 客户端插件 服务端插件 交互

**备注**

### 案例 –【描述】树形加载完成事件，展示到第二级节点

**代码来源**

Kingdee.K3.FIN.CR.Business.PlugIn.CustomFetchData.FieldTreeList,

Kingdee.K3.FIN.CR.Business.PlugIn

业务对象：CR\_FieldTreeList 字段列表树

**需求背景**

自定义界面，树数据展开到指定级别。

**实现方案**

通过获取客户端发来的TreeViewDataSourceChanged 树加载完成事件，回传客户端树展开调用。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Linq;  using System.Text;  using Kingdee.BOS.Util;  using Kingdee.BOS.Core.DynamicForm.PlugIn;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using Kingdee.BOS.Core.DynamicForm.PlugIn.ControlModel;   |  | | --- | | [Description("字段列表树-动态表单插件")]  public class FieldTreeList : AbstractDynamicFormPlugIn  {  public override void CustomEvents(CustomEventsArgs e)  {  //树形加载完成事件，展示到第二级节点  if (e.EventName.EqualsIgnoreCase("TreeViewDataSourceChanged"))  {  var treeView = this.View.GetControl<TreeView>("FTreeView");  treeView.InvokeControlMethod("ExpandNode", \_rootNodeId);  }  }  } | | |

## DataChanged

**触发时机**

字段值更新之后触发

**应用场景**

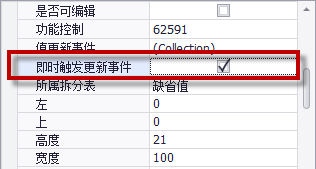
字段值更新之后可能需要级联触发修改其他字段，针对字段值改变做一些附加处理。

**关键字**

值更新 级联修改 数据改变

**备注**

一般在BOS IDE里需要勾选相应字段的“即时触发更新事件”属性



### 案例1 – 部门全称随组织关系改变

**代码来源**

Kingdee.K3.BD.Common.Business.PlugIn.DepartmentEdit

**需求背景**

部门全称格式为“隶属组织+上级部门+部门名称”组成，当部门隶属组织或上级部门发生改变时，部门全程也随之改变

**实现方案**

字段值改变会自动生成一个改变标识，在DataChanged事件可以捕获到该字段，如果捕获到的是上级部门名称，则直接重新拼接部门全程，并更新到部门全称字段上。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.Bill.PlugIn;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  namespace Kingdee.K3.BD.Common.Business.PlugIn  {  /// <summary>  /// 部门 编辑插件  /// </summary>  public class DepartmentEdit : AbstractBillPlugIn  {  /// <summary>  /// 字段修改事件函数重载  /// </summary>  public override void DataChanged(DataChangedEventArgs e)  {  switch (e.Key.ToUpper())  {  case "FNAME":  case "FORGSUBJECTSCHEMAID": //组织隶属方案  case "FPARENTDEPTID": //上级部门  //组织隶属方案和上级部门变化，重新生成部门全称  //其中BuildNewFullName为拼接全称函数，此处省略具体实现逻辑  this.Model.SetValue("FFullName", BuildNewFullName(e.Key));  this.View.UpdateView("FFullName");  break;  }  }  }  } | |

## EntityRowClick

**触发时机**

分录行单击时触发

**应用场景**

在单击分录行时设置菜单是否可用、刷新代理字段、重新计算等。

**关键字**

分录行 单击 刷新字段 重新计算、分录控件、分录行点击

**备注**

### 案例1 – 单击分录行时判断是否能够删除

**代码来源**

Kingdee.K3.FIN.HS.Business.PlugIn.OutSrcInStockFee

**需求背景**

委外入库核算费用明细表中，费用明细单据体菜单中包含“行删除”菜单项，当该费用已经钩稽，则不允许删除该费用。

**实现方案**

单击分录行时，根据行索引从数据模型中获取当前行中“是否钩稽”标识，如已钩稽则将“行删除”菜单状态设置为不可用，不允许用户删除该条记录，如果未钩稽则将“行删除”菜单状态设置为可用，允许用户删除该条记录。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.DynamicForm.PlugIn;  namespace Kingdee.K3.FIN.HS.Business.PlugIn  {  public class OutSrcInStockFee : AbstractDynamicFormPlugIn  {  /// <summary>  /// 设置删除菜单是否可用  /// </summary>  /// <param name="row"></param>  public override void EntityRowClick(EntityRowClickEventArgs e)  {  string hooked = this.View.Model.GetValue("FHooked", e.row).ToString();  if (hooked == "1")  {  //设置“行删除”菜单不可用  this.View.GetBarItem("FExpEntity", "tbDelRow").Enabled = false;  }  else  {  //设置“行删除”菜单可用  this.View.GetBarItem("FExpEntity", "tbDelRow").Enabled = true;  }  }  }  } | |

### 案例2 –点击单据体时，操作单据体分配

**代码来源**

Kingdee.BOS.Business.PlugIn.Permission

**需求背景**

单据体点击的时候操作单据体分配；

**实现方案**

调用EntityRowClick事件，在事件里对key做判断，如果是对应的key则执行对应的操作。

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| /// <summary>  /// 单据体点击事件  /// </summary>  public override void EntityRowClick(Core.DynamicForm.PlugIn.Args.EntityRowClickEventArgs e)  {  if (e.Row < 0) return;  //首先判断是否是业务对象单据体点击  //获得选择的业务对象内码  //填充分配单据体  if (e.Key.Equals(PermissionMaintainKey.FBizObjectEntity, StringComparison.OrdinalIgnoreCase))  {  this.objTypeId = this.View.Model.GetValue(PermissionMaintainKey.FObjectId, e.Row);  if (objTypeId.IsNullOrEmpty() == false)  {  BusinessObjectValueChange();  }  }  } | |

## TabItemSelectedChange

**触发时机**

页签切换后

**应用场景**

页签切换切换后相关业务处理

**关键字**

页签、页签控件、页签切换、页签切换

**备注**

### 案例1 – 计划员工作台-显示甘特图

**代码来源**

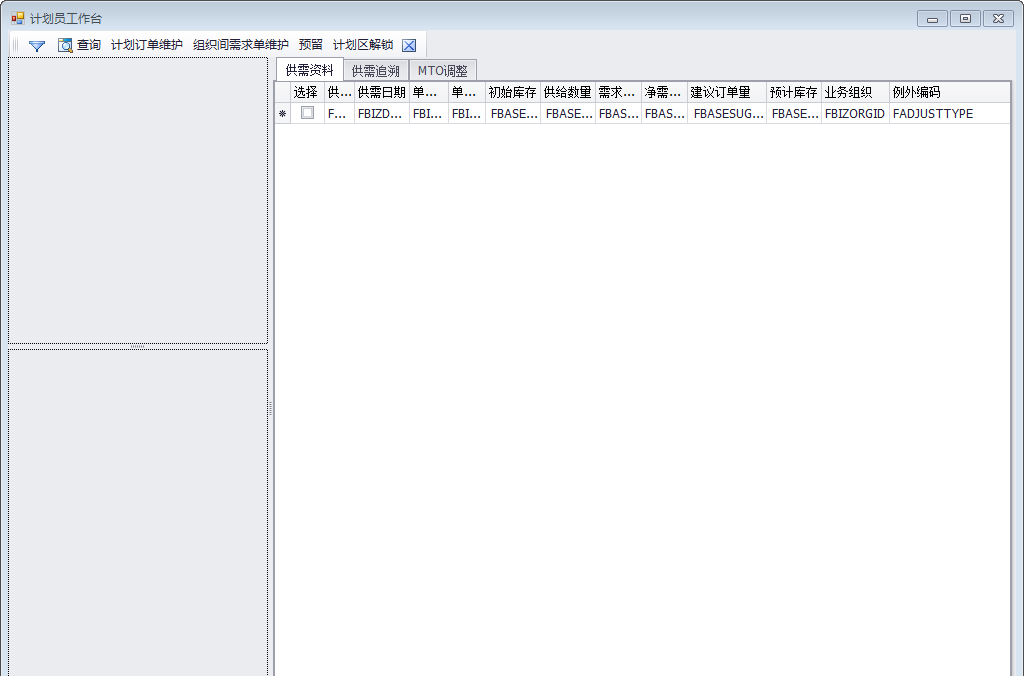
Kingdee.K3.MFG.PLN.Business.PlugIn.DynamicForm.PlanWorkBench

**需求背景**

计划员工作台是为计划员提供计划分析、调整的平台，通过该平台，能够让计划员清晰计划需求逻辑，并通过例外信息的展示和各角度的现状跟踪（单据的现状跟踪、物料的预计可用库存分析、拖期单据的跟踪），供计划员快速决策。物料针对不同的单据如销售订单，计划单在一段时间里做一个需求分析统计，为了更直观的展现物料一段时间物料需求。需要做个一个更直观的按日期分析统计。

**实现方案**

通过新增一个页签，在根据选择的物料数据行，在切换到供需追溯页签进行一个甘特图效果展示。如下图“供需资料”，切换到“供需追溯”页签时会进行效果图展现。



**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System.ComponentModel;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using Kingdee.BOS.Core.Permission;  using Kingdee.BOS.ServiceHelper;  using Kingdee.BOS.Core.DynamicForm;  using Kingdee.BOS.Util;  using Kingdee.K3.Core.MFG;  using Kingdee.K3.Core.MFG.PLN.ParamOption;  using Kingdee.K3.MFG.BusinessCommon.BillPlugIn;  /// <summary>  /// 计划员工作台插件  /// </summary>  [Description("计划员工作台插件")]  public partial class PlanWorkBench : AbstractMFGDynamicFormPlugIn  {  /// <summary>  /// 页签切换事件  /// </summary>  /// <param name="e"></param>  public override void TabItemSelectedChange(TabItemSelectedChangeEventArgs e)  {  if (e.Key.EqualsIgnoreCase(CONST\_PLN\_PLANWORKBENCH.CONST\_FControl.KEY\_FContentTab))  {  if (e.TabIndex == 1)  {  //显示甘特图  this.ShowTraceViewWithChoose();  }  }  }  /// <summary>  /// 根据选中的分录行显示预留追溯  /// </summary>  protected void ShowTraceViewWithChoose()  {  PermissionAuthResult permissionResult = PermissionServiceHelper.FuncPermissionAuth(this.Context,  new BusinessObject() { Id = MFGFormIdConst.SubSys\_PLN.ReserveTraceBack }, PermissionConst.View);  if (!permissionResult.Passed)  {  this.View.ShowWarnningMessage("当前用户没有预留追溯查询的查看权限！");  return;  }  this.ShowReserveLinkTraceView();  }  /// <summary>  /// 调用预留追溯界面  /// </summary>  /// <param name="inputParam"></param>  private void ShowReserveLinkTraceView()  {  DynamicFormShowParameter ShowPara = new DynamicFormShowParameter();  ShowPara.PageId = "PLN\_PLANWORKBENCH\_ReserveTrace";  ShowPara.ParentPageId = this.View.PageId;  ShowPara.OpenStyle.ShowType = ShowType.InContainer;  ShowPara.OpenStyle.TagetKey = CONST\_PLN\_PLANWORKBENCH.CONST\_FControl.KEY\_FTracePanel;  ShowPara.FormId = MFGFormIdConst.SubSys\_PLN.ReserveTraceBack;  this.View.ShowForm(ShowPara);  }  } | |

## EntityRowDoubleClick

**触发时机**

分录行双击

**应用场景**

双击单据体分录行。

**关键字**

分录控件，分录行双击

**备注**

### 案例1 –【描述】自动核销向导

**代码来源**

Kingdee.K3.FIN.Business.PlugIn.Match.AutoStencilWizardEdit, Kingdee.K3.FIN.Business.PlugIn

**需求背景**

选择需要核销的组织和核销方案，具体查看核销方案详细信息

**实现方案**

鼠标双击指定关键字的分录行，获取当前分录行信息用新视图打开.

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS.Core.Bill;  using Kingdee.BOS.Core.DynamicForm;  using Kingdee.BOS.Core.DynamicForm.PlugIn.Args;  using Kingdee.BOS.Core.DynamicForm.PlugIn.WizardForm;  using Kingdee.BOS.Core.Metadata;  using Kingdee.K3.FIN.Core;  [Description("自动核销向导")]  public class AutoStencilWizardEdit : AbstractWizardFormPlugIn  {  ///// <summary>  ///// 展示单据  ///// </summary>  ///// <param name="e"></param>  public override void EntityRowDoubleClick(EntityRowClickEventArgs e)  {  OpenSchemeDetail(e);  }  /// <summary>  /// 根据用户进去的业务对象来选择打开对应的核销方案  /// </summary>  /// <param name="e"></param>  private void OpenSchemeDetail(EntityRowClickEventArgs e)  {  if (!e.Key.EqualsIgnoreCase(FMatchSchemeEntry))  {  return;  }  string bussFormID = GetBusenessObjFormID();  if (!CheckPerssionViewByID(bussFormID))  {  this.View.ShowErrMessage("您没有核销方案的查看权限！");  return;  }  BillShowParameter param = new BillShowParameter();  param.FormId = GetBusenessObjFormID();  param.ParentPageId = this.View.PageId;  param.Status = OperationStatus.VIEW;  param.AllowNavigation = false;  param.OpenStyle.ShowType = ShowType.MainNewTabPage;  param.PKey = this.View.Model.GetBaseDataID("FMatchSchemeID", e.Row).ToString();  this.View.ShowForm(param);  }  /// <summary>  /// 根据用户进去的业务对象来选择formID  /// </summary>  protected virtual string GetBusenessObjFormID()  {  string thisOpenID = this.View.OpenParameter.FormId;  string result = "";  switch (thisOpenID)  {  case BusinessObjectConst.AP\_PayMatck:  result = BusinessObjectConst.AP\_MatchScheme;  break;  case BusinessObjectConst.AP\_BillingMatck:  result = BusinessObjectConst.AP\_BillingMatchScheme;  break;  case BusinessObjectConst.AR\_Matck:  result = BusinessObjectConst.AR\_MatchScheme;  break;  case BusinessObjectConst.AR\_BillingMatck:  result = BusinessObjectConst.AR\_BillingMatchScheme;  break;  case BusinessObjectConst.ER\_MatchWizard:  result = BusinessObjectConst.ER\_MatchScheme;  break;  }  return result;  }  } | |