账表（服务取数）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 | 20150415 | 杨兵  张晋博 | 刘兵 | 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](mailto:jinbo_zhang@kingdee.com)

目录

[账表（服务取数）插件示例代码 1](#_Toc416352687)

[账表（服务取数）插件概要说明 4](#_Toc416352688)

[如何创建账表服务插件 6](#_Toc416352689)

[如何启用账表服务插件 8](#_Toc416352690)

[公共属性 9](#_Toc416352691)

[BusinessInfo 9](#_Toc416352692)

[Context 10](#_Toc416352693)

[DctSpecialTempSumTable 10](#_Toc416352694)

[DynamicHeader 10](#_Toc416352695)

[ReportHeader 10](#_Toc416352696)

[ReportId 10](#_Toc416352697)

[ReportProperty 10](#_Toc416352698)

[ReportTitles 10](#_Toc416352699)

[SummarySpecialFields 11](#_Toc416352700)

[TempTableNameList 11](#_Toc416352701)

[分页账表特有属性 11](#_Toc416352702)

[CurrentListPosition 11](#_Toc416352703)

[CacheMapTempTableNameList 11](#_Toc416352704)

[CacheDataList 11](#_Toc416352705)

[树形账表特有属性 11](#_Toc416352706)

[CurrentGroupID 11](#_Toc416352707)

[事件 12](#_Toc416352708)

[Initialize 12](#_Toc416352709)

[案例 – 销售订单执行汇总表账表取数插件初始化 12](#_Toc416352710)

[GetData 17](#_Toc416352711)

[GetTableName 17](#_Toc416352712)

[BuilderReportSqlAndTempTable 17](#_Toc416352713)

[案例 – 销售订单执行汇总表账表取数 17](#_Toc416352714)

[BuilderSelectFieldSQL 19](#_Toc416352715)

[案例 – 核算错误记录账表取数 20](#_Toc416352716)

[BuilderTempTableOrderBySQL 21](#_Toc416352717)

[BuilderFromWhereSQL 21](#_Toc416352718)

[案例 – 核算错误记录账表取数 22](#_Toc416352719)

[GetIdentityFieldIndexSQL 24](#_Toc416352720)

[ExecuteBatch 25](#_Toc416352721)

[GetReportHeaders 25](#_Toc416352722)

[案例 – 应付款账龄分析表构建列头信息 25](#_Toc416352723)

[GetReportTitles 29](#_Toc416352724)

[案例 – 销售订单执行汇总表账表表头信息 30](#_Toc416352725)

[AnalyzeDspCloumn 35](#_Toc416352726)

[AfterCreateTempTable 35](#_Toc416352727)

[GetReportData 35](#_Toc416352728)

[案例 – 供应商供货汇总表取数 36](#_Toc416352729)

[GetRowsCount 37](#_Toc416352730)

[案例 – 库存台账分页账表获取当前页数据行数 38](#_Toc416352731)

[GetSummaryData 40](#_Toc416352732)

[GetSummaryColumnInfo 40](#_Toc416352733)

[案例 – 销售订单执行汇总表分组汇总字段信息 40](#_Toc416352734)

[GetSummaryColumsSQL 43](#_Toc416352735)

[案例 – 销售订单执行汇总表分组汇总SQL指令 43](#_Toc416352736)

[CloseReport 45](#_Toc416352737)

[案例 – 供货商供货汇总表关闭时，清除临时表 46](#_Toc416352738)

[GetList 47](#_Toc416352739)

[GetTreeNodes 49](#_Toc416352740)

# 账表（服务取数）插件概要说明

账表的服务端取数插件，是通过插件组装账表取数Sql指令，把取数结果返回给平台账表引擎，平台将取数结果绑定到账表页面的过程，简单说，平台在账表开发过程中，只负责把插件取到的账表数据绑定到设计好的账表模型上，取数逻辑由账表服务端插件完成

开发过程中，有几个重要模型

1. 设计账表模型
   1. 账表表头字段
   2. 账表表体字段（也说成账表列头字段）
2. 设计账表过滤模型
   1. 自定义快捷过滤面板
   2. 高级过滤条件面板
   3. 排序面板
   4. 分组汇总面板
   5. 显示隐藏列面板

账表根据使用场景，分为简单账表、树形账表、分页账表、简单Sql账表

简单Sql账表没有服务取数插件

所有账表都须实现的重要事件

1. Initialize
2. BuilderSelectFieldSQL
3. BuilderFromWhereSQL
4. BuilderReportSqlAndTempTable
5. GetReportTitles
6. GetReportHeaders
7. GetSummaryColumnInfo

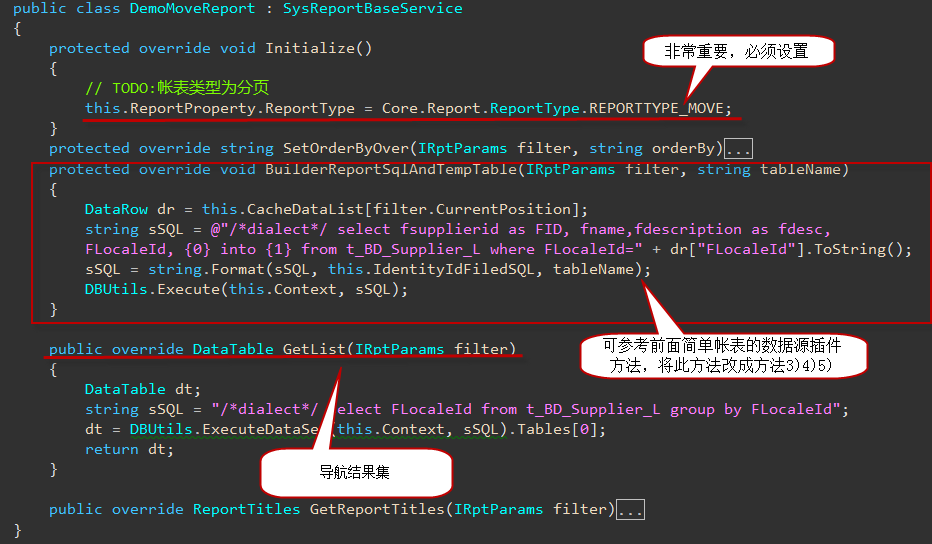
分页账表特点：

与简单账表区别在于，分页账表通过导航按钮切换帐表内容



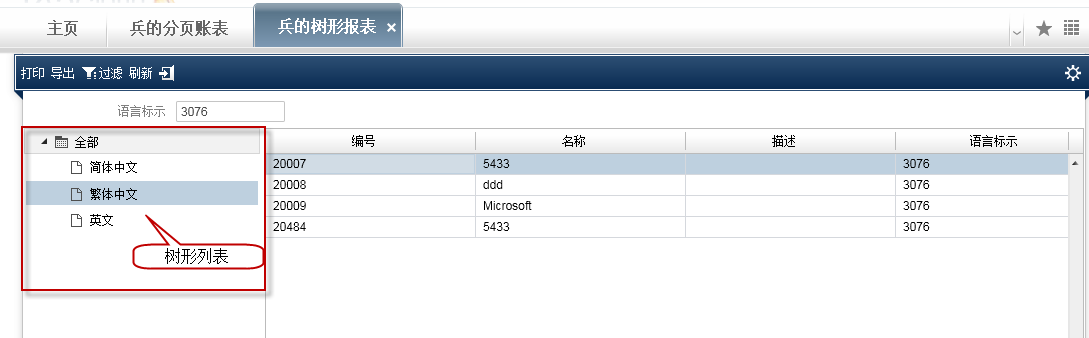
分页账表基本原理说明：

* 需要提供用于导航的结果集，如下图中以语言标识为导航结果集（实现GetList方法）
* 显示数据需要按照导航结果过滤（实现BuilderReportSqlAndTempTable方法）



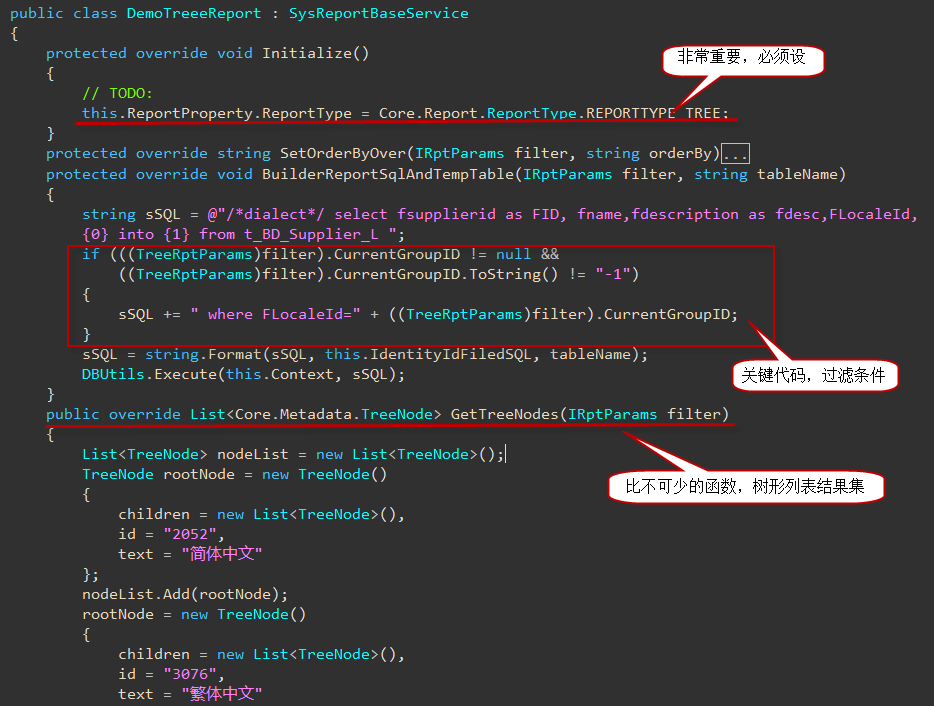
树形账表特点：

* 树形账表通过树节点切换账表内容



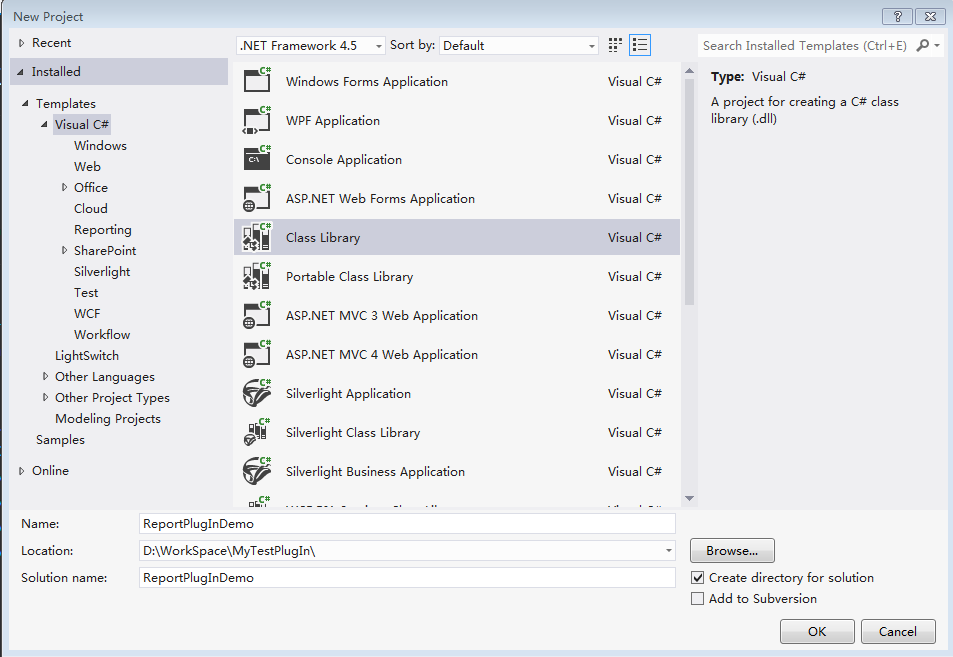
树形账表基本原理说明：

* 需要提供用于导航树形结果集（List<TreeNode>），如下图中以语言标识为树形结果集（实现GetTreeNodes方法）
* 显示数据需要按照树节点结果过滤（实现BuilderReportSqlAndTempTable方法，使用CurrentGroupID重要属性作过滤条件）

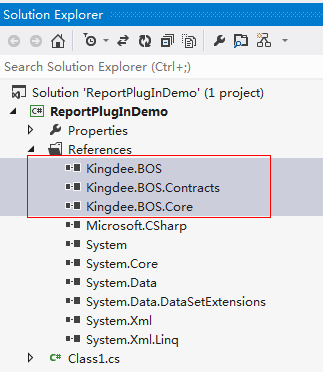


## 如何创建账表服务插件

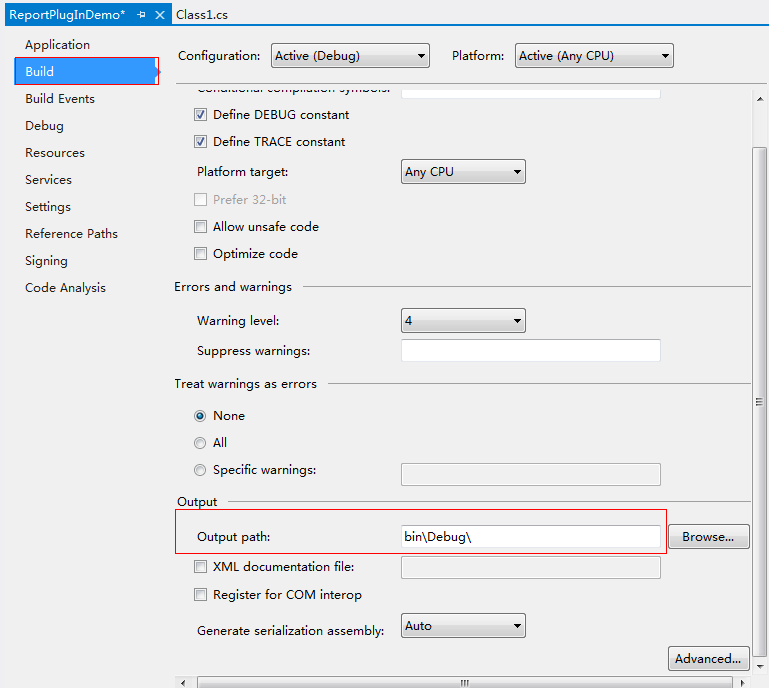
* 创建插件类库工程



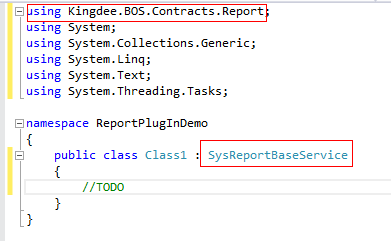
* 插件类库引用BOS平台组件（被引用组件在Cloud网站website\bin目录下）



* 修改插件类库编译输出路径为website\bin（右键插件类库，选择属性）



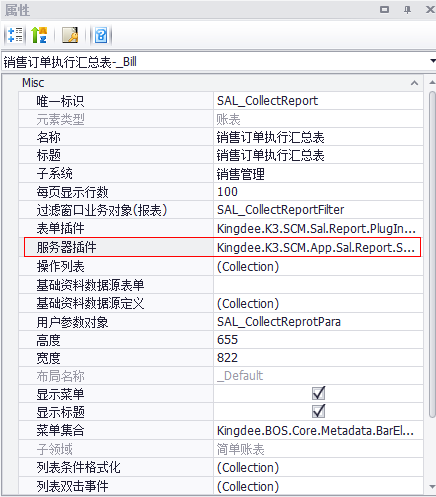
* 修改类文件继承基类，并引用基类所在命名空间

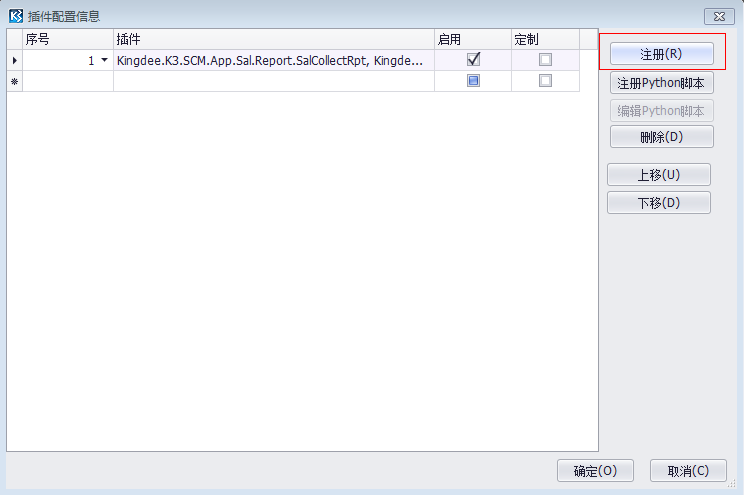


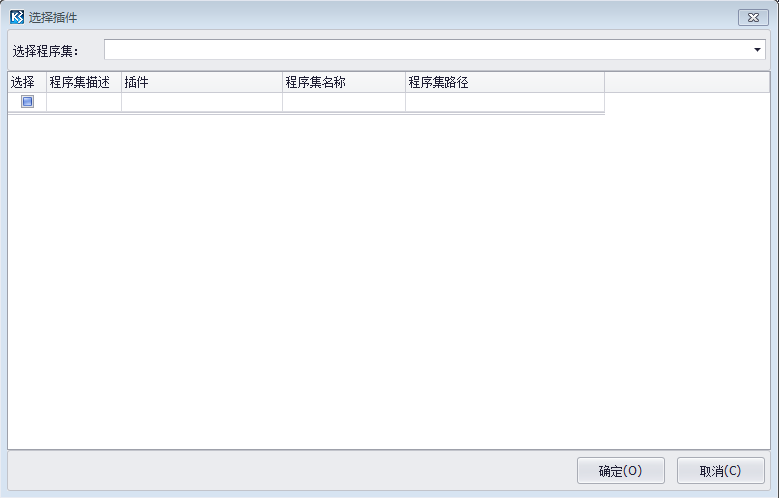
## 如何启用账表服务插件

插件编写完毕，类库编译通过后

进入BOS设计器，打开对应账表模型，配置账表服务插件：







# 公共属性

## BusinessInfo

账表模型的元数据描述信息

## Context

当前登录系统的上下文信息，如数据库连接信息，当前登录操作员信息等

## DctSpecialTempSumTable

临时汇总数据表格

## DynamicHeader

动态列头

## ReportHeader

账表列头

## ReportId

当前账表模型唯一标示

## ReportProperty

* 账表属性，包含账表开发过程中的全局信息
* 其重要属性说明：

|  |  |
| --- | --- |
| 属性名 | 描述 |
| ReportType | 账表类型（默认是简单账表、如果开发的是分页或树形帐表，请在账表插件初始化事件中，对此属性进行正确赋值，否则账表引擎会默认按简单账表处理页面） |
| IsGroupSummary | 是否支持分组汇总 |
| IsUIDesignerColumns | 账表列头是否是通过BOSIDE设计 |
| SimpleAllCols | 是否锁定账表表格列 |
| DecimalControlFieldList | 精度控制字段信息 |
| DspInsteadColumnsInfo | 列表格式化列，指示Key列被Value列内容替代 |
| GroupSummaryInfoData | 分组汇总信息 |

## ReportTitles

账表表头字段信息

## SummarySpecialFields

汇总字段信息

## TempTableNameList

临时表列表

# 分页账表特有属性

## CurrentListPosition

分页账表属性

分页账表当前页账表数据对象的位置信息，此属性与CacheMapTempTableNameList属性一起使用

## CacheMapTempTableNameList

分页账表属性

记录分页账表每页账表数据对象信息，此属性与CurrentListPosition属性一起使用

## CacheDataList

分页账表属性

根据账表当前页位置，获取DataRow对象

# 树形账表特有属性

## CurrentGroupID

树形账表属性

树形账表分组标识，点击树形账表左边树时，由平台账表引擎为其赋值，其数据来源于账表插件GetTreeNodes方法返回的树节点id属性的值，对应树形账表插件在BuilderReportSqlAndTempTable方法拼接账表取数sql时，需要以CurrentGroupID作为过滤条件拼接到Sql指令的Where部分

# 事件

## Initialize

**触发时机**

账表View初始化，开始初始化账表服务取数插件代理对象时

**应用场景**

此时账表插件基类已完成账表属性对象（ReportProperty）的创建工作

**关键字**

初始化

**备注**

一般账表插件在此事件中，对刚创建的ReportProperty做进一步初始化

### 案例 – 销售订单执行汇总表账表取数插件初始化

**代码来源**

Kingdee.K3.SCM.App.Sal.Report.SalCollectRpt

**需求背景**

销售订单执行汇总表初始化时，进一步初始化账表属性对象

1. 账表属性的报表类型、报表名称
2. 由于是汇总表，配置其明细表模型唯一标示和唯一标示一张明显表的字段信息（双击汇总表一行记录时，根据以上信息定位到明细表）
3. 账表属性的列替代显示信息（key：被替代显示的列，value：替代显示的列）
4. 字段精度控制描述信息（即哪个字段的精度被哪个字段所控制）

**实现方案**

账表服务取数插件初始化时，修改账表属性中的报表名称、明细表属性、替代显示列信息、字段精度控制信息等

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using Kingdee.BOS.Contracts.Report;  using Kingdee.BOS.Core.Report;  using Kingdee.BOS;  using Kingdee.BOS.Core.List;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS.Contracts;  using Kingdee.K3.SCM.Sal.Report.PlugIn;  using System.Collections;  using Kingdee.BOS.App.Data;  using Kingdee.BOS.Core.SqlBuilder;  using Kingdee.BOS.Core;  using Kingdee.BOS.Core.Util;  using Kingdee.BOS.Util;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.Core.Permission;  using Kingdee.BOS.App.Core.BusinessFlow;  using Kingdee.K3.Core;  using Kingdee.BOS.Core.BusinessFlow.ServiceArgs;  using Kingdee.BOS.BusinessEntity.BusinessFlow;  using System.Data;  using Kingdee.BOS.Core.Permission.Objects;  using Kingdee.BOS.Core.CommonFilter;  namespace Kingdee.K3.SCM.App.Sal.Report  {  /// <summary>  /// 销售汇总表数据源插件  /// </summary>  public class SalCollectRpt : SysReportBaseService  {  public override void Initialize()  {  base.Initialize();  this.ReportProperty.ReportType = ReportType.REPORTTYPE\_NORMAL;  this.ReportProperty.ReportName = "销售订单执行汇总表";  this.ReportProperty.DetailReportId = "SAL\_DetailReport";  this.ReportProperty.PrimaryKeyFieldName = "FBILLNO";  this.ReportProperty.IsGroupSummary = true;  this.ReportProperty.IsUIDesignerColumns = true;  //显示替代列this.ReportProperty.DspInsteadColumnsInfo.DefaultDspInsteadColumns.Add("FSALEORGID", "FSALEORGNAME"); this.ReportProperty.DspInsteadColumnsInfo.DefaultDspInsteadColumns.Add("FSALEDEPTID", "FSALEDEPT"); this.ReportProperty.DspInsteadColumnsInfo.DefaultDspInsteadColumns.Add("FSALEGROUPID", "FSALEGROUP"); this.ReportProperty.DspInsteadColumnsInfo.DefaultDspInsteadColumns.Add("FSALERID", "FSALES"); this.ReportProperty.DspInsteadColumnsInfo.DefaultDspInsteadColumns.Add("FMATERIALID", "FMATERIALNUMBER"); this.ReportProperty.DspInsteadColumnsInfo.DefaultDspInsteadColumns.Add("FCURRENCYID", "FMONEYTYPE"); this.ReportProperty.DspInsteadColumnsInfo.DefaultDspInsteadColumns.Add("FCUSTID", "FCUSTOMERNUMBER"); this.ReportProperty.DspInsteadColumnsInfo.DefaultDspInsteadColumns.Add("FMATERIALGROUP", "FMATERIALGROUPNAME"); this.ReportProperty.DspInsteadColumnsInfo.DefaultDspInsteadColumns.Add("FAUXPROPID", "FAUXPROP"); this.ReportProperty.DspInsteadColumnsInfo.DefaultDspInsteadColumns.Add("FUNITID", "FUNITNAME");  //精度控制  List<DecimalControlField> lstDcf = new List<DecimalControlField>();  DecimalControlField dcf = new DecimalControlField();  //单价  dcf.ByDecimalControlFieldName = "FPRICE";//  dcf.DecimalControlFieldName = "FPRICEDIGITS";  lstDcf.Add(dcf);  dcf = new DecimalControlField();  //订单  dcf.ByDecimalControlFieldName = "FSOQTY";  dcf.DecimalControlFieldName = "FPRECISION";  lstDcf.Add(dcf);  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FSOAMOUNT";  dcf.DecimalControlFieldName = "FAMOUNTDIGITS";  lstDcf.Add(dcf);  //发货单  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FSQTY";  dcf.DecimalControlFieldName = "FPRECISION";  lstDcf.Add(dcf);  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FSAMOUNT";  dcf.DecimalControlFieldName = "FAMOUNTDIGITS";  lstDcf.Add(dcf);  //出库  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FALREADYQTY";  dcf.DecimalControlFieldName = "FPRECISION";  lstDcf.Add(dcf);  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FALREADYAMOUNT";  dcf.DecimalControlFieldName = "FAMOUNTDIGITS";  lstDcf.Add(dcf);  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FNOTQTY";  dcf.DecimalControlFieldName = "FPRECISION";  lstDcf.Add(dcf);  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FNOTAMOUNT";  dcf.DecimalControlFieldName = "FAMOUNTDIGITS";  lstDcf.Add(dcf);  //退货  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FRETURNPQTY";  dcf.DecimalControlFieldName = "FPRECISION";  lstDcf.Add(dcf);  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FRETURNPAMOUNT";  dcf.DecimalControlFieldName = "FAMOUNTDIGITS";  lstDcf.Add(dcf);  //退库  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FRETURNSQTY";  dcf.DecimalControlFieldName = "FPRECISION";  lstDcf.Add(dcf);  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FRETURNSAMOUNT";  dcf.DecimalControlFieldName = "FAMOUNTDIGITS";  lstDcf.Add(dcf);  //开票  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FINVOECEQTY";  dcf.DecimalControlFieldName = "FPRECISION";  lstDcf.Add(dcf);  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FNVOECEAMOUNT";  dcf.DecimalControlFieldName = "FAMOUNTDIGITS";  lstDcf.Add(dcf);  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FNINVOECEQTY";  dcf.DecimalControlFieldName = "FPRECISION";  lstDcf.Add(dcf);  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FNNVOECEAMOUNT";  dcf.DecimalControlFieldName = "FAMOUNTDIGITS";  lstDcf.Add(dcf);  //收款  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FRECEIPTAMOUNT";  dcf.DecimalControlFieldName = "FAMOUNTDIGITS";  lstDcf.Add(dcf);  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FNRECEIPTAMOUNT";  dcf.DecimalControlFieldName = "FAMOUNTDIGITS";  lstDcf.Add(dcf);  //核销金额可能是尾差过小，就不按金额精度处理  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FCHARGEOFFAMOUNT";  dcf.DecimalControlFieldName = "FAMOUNTDIGITS";  lstDcf.Add(dcf);  //应收  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FRECQTY";  dcf.DecimalControlFieldName = "FPRECISION";  lstDcf.Add(dcf);  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FRECAMOUNT";  dcf.DecimalControlFieldName = "FAMOUNTDIGITS";  lstDcf.Add(dcf);  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FRECNQTY";  dcf.DecimalControlFieldName = "FPRECISION";  lstDcf.Add(dcf);  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FRECNAMOUNT";  dcf.DecimalControlFieldName = "FAMOUNTDIGITS";  lstDcf.Add(dcf);  dcf = new DecimalControlField();  dcf.ByDecimalControlFieldName = "FWRITEOFFAMOUNT";  dcf.DecimalControlFieldName = "FAMOUNTDIGITS";  lstDcf.Add(dcf);  this.ReportProperty.DecimalControlFieldList = lstDcf;  }  }  } | |

## GetData

**备注**

此方法由插件平台报表基类实现，插件不用关注

**示例代码**

无

## GetTableName

**备注**

此方法由插件平台报表基类实现，插件不用关注

用于创建一张临时表，完成插件账表数据逻辑后，用此表承载账表数据

**示例代码**

无

## BuilderReportSqlAndTempTable

**触发时机**

正式进入账表取数sql拼接并取数，把账表取数结果放到上一步创建的临时表中

**应用场景**

开始进行账表sql拼接取数，并把账表取数结果放到上一步创建的临时表中

**关键字**

账表取数

**备注**

如果此账表插件设置sql数据逻辑由插件完成（this.IsCreateTempTableByPlugin），即调用BuilderReportSqlAndTempTable接口，否则调用以下3个接口，完成账表取数逻辑的sql指令即：BuilderSelectFieldSQL、BuilderTempTableOrderBySQL、BuilderFormWhereSQL

### 案例 – 销售订单执行汇总表账表取数

**代码来源**

Kingdee.K3.SCM.App.Sal.Report.SalCollectRpt

**需求背景**

销售订单执行汇总表根据账表过滤条件取数

**实现方案**

账表插件在BuilderReportSqlAndTempTable方法中，完成账表取数并把数据填充到临时表中

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using Kingdee.BOS.Contracts.Report;  using Kingdee.BOS.Core.Report;  using Kingdee.BOS;  using Kingdee.BOS.Core.List;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS.Contracts;  using Kingdee.K3.SCM.Sal.Report.PlugIn;  using System.Collections;  using Kingdee.BOS.App.Data;  using Kingdee.BOS.Core.SqlBuilder;  using Kingdee.BOS.Core;  using Kingdee.BOS.Core.Util;  using Kingdee.BOS.Util;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.Core.Permission;  using Kingdee.BOS.App.Core.BusinessFlow;  using Kingdee.K3.Core;  using Kingdee.BOS.Core.BusinessFlow.ServiceArgs;  using Kingdee.BOS.BusinessEntity.BusinessFlow;  using System.Data;  using Kingdee.BOS.Core.Permission.Objects;  using Kingdee.BOS.Core.CommonFilter;  namespace Kingdee.K3.SCM.App.Sal.Report  {  /// <summary>  /// 销售汇总表数据源插件  /// </summary>  public class SalCollectRpt : SysReportBaseService  {  public override void BuilderReportSqlAndTempTable(IRptParams filter, string tableName)  {  ModifyShowHideColumns(ref filter);  InitTables();//初始化报表使用的临时表，使用一次即删除  moreFilter = filter.FilterParameter.FilterString.Trim();//高级过滤  this.SetFilter(filter);//设置快捷页签过滤条件  DynamicObject userParameter = filter.ParameterData;  if (userParameter != null)  {  this.onTime = Convert.ToBoolean(userParameter["onTime"]);  }  DynamicObject dyFilter = filter.FilterParameter.CustomFilter;//快捷页签过滤  if (dyFilter["SaleOrgList"] == null || string.IsNullOrWhiteSpace(dyFilter["SaleOrgList"].ToString()))  {  SetRptBosTable(tableName);  return;  }  else  {  this.GetSQLWhere(); //进行过滤快捷页签汇总  List<BaseDataTempTable> listBaseDataTempTable = filter.BaseDataTempTable; // 如果设置了基础资料数据权限，则获取基础资料数据权限隔离的临时表集合 SalRptCommon.GetFilterOrder(this.Context,rptFilterTable, !string.IsNullOrWhiteSpace(moreFilter), creatWhereFirst.ToString(), this.filterOrgList, listBaseDataTempTable, "SAL\_CollectReport");  // 添加基础资料数据权限隔离的临时表到待删除临时表集合  foreach (BaseDataTempTable item in listBaseDataTempTable)  {  deleteTables.Add(item.TempTable);  }  GetDataSource(rptFilterTable);//获取单据转换流程数据  this.CreatSQLWhereForTemp();  this.GetFinalSummaryData();//获取最终汇总数据  this.SetRptBosTable(tableName);//传到平台表里面  }  }  }  }  //说明：由于此方法很大，此示例省略xxxx行代码  //根据过滤条件、业务条件等把满足条件的账表数据取出来并赋值给临时表 | |

## BuilderSelectFieldSQL

**触发时机**

如插件在初始化时，设置this.IsCreateTempTableByPlugin = false;开始构建账表取数Sql指令时

**应用场景**

插件设置标志this.IsCreateTempTableByPlugin = false插件负责sql指令的Select部分拼接，其他取数等工作交由账表引擎处理

**关键字**

账表取数

**备注**

如果此账表插件设置sql数据逻辑由插件完成（this.IsCreateTempTableByPlugin），即调用BuilderReportSqlAndTempTable接口，否则调用以下3个接口，完成账表取数逻辑的sql指令即：BuilderSelectFieldSQL、BuilderTempTableOrderBySQL、BuilderFormWhereSQL

### 案例 – 核算错误记录账表取数

**代码来源**

Kingdee.K3.FIN.HS.App.Report.HSErrorLog

**需求背景**

核算错误记录账表取数

**实现方案**

账表插件初始化时，设置IsCreateTempTableByPlugin为false，所以在BuilderSelectFieldSQL方法中，完成账表取数Sql的Select部分

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Data;  using Kingdee.BOS.Core.Report.PlugIn;  using Kingdee.BOS.Core.Report;  using Kingdee.BOS.Contracts.Report;  using Kingdee.BOS.App.Data;  using Kingdee.BOS.Core.CommonFilter;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS;  using Kingdee.K3.FIN.App.Core;  using Kingdee.K3.FIN.Core;  using Kingdee.BOS.Resource;  using System.ComponentModel;  namespace Kingdee.K3.FIN.HS.App.Report  {  /// <summary>  /// 核算错误记录插件  /// </summary>  [Description("核算错误记录插件")]  public class HSErrorLog : SysReportBaseService  {  protected override string BuilderSelectFieldSQL(IRptParams filter)  {  return @"select seq.FMaterialID,seq.FStockOrgId,FStockID,FCargoOwnerId as FOwnerID,max(FBillCheckResult) as Result,  seq.FBillId as FSrcID ,FBillEntryId as FSrcEntryID,FAcctgSysId as FacctSysID,FAcctgOrgId as FAcctOrgID,FYear,FPeriod ,seq.FQty,process.FAcctgRangeId as FAcctRangeID ";  }  }  } | |

## BuilderTempTableOrderBySQL

**备注**

如果此账表插件设置sql数据逻辑由插件完成（this.IsCreateTempTableByPlugin），即调用BuilderReportSqlAndTempTable接口，否则调用以下3个接口，完成账表取数逻辑的sql指令即：BuilderSelectFieldSQL、BuilderTempTableOrderBySQL、BuilderFormWhereSQL

此事件应用在为账表数据临时表构建排序字段sql指令

**示例代码**

无

## BuilderFromWhereSQL

**触发时机**

如插件在初始化时，设置this.IsCreateTempTableByPlugin = false;开始构建账表取数Sql指令时

**应用场景**

插件设置标志this.IsCreateTempTableByPlugin = false插件负责sql指令的From和Where部分拼接，其他取数等工作交由账表引擎处理

**关键字**

账表取数

**备注**

如果此账表插件设置sql数据逻辑由插件完成（this.IsCreateTempTableByPlugin），即调用BuilderReportSqlAndTempTable接口，否则调用以下3个接口，完成账表取数逻辑的sql指令即：BuilderSelectFieldSQL、BuilderTempTableOrderBySQL、BuilderFormWhereSQL

### 案例 – 核算错误记录账表取数

**代码来源**

Kingdee.K3.FIN.HS.App.Report.HSErrorLog

**需求背景**

核算错误记录账表取数

**实现方案**

账表插件初始化时，设置IsCreateTempTableByPlugin为false，所以在BuilderWhereFromSQL方法中，完成账表取数Sql的From和Where部分

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Data;  using Kingdee.BOS.Core.Report.PlugIn;  using Kingdee.BOS.Core.Report;  using Kingdee.BOS.Contracts.Report;  using Kingdee.BOS.App.Data;  using Kingdee.BOS.Core.CommonFilter;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS;  using Kingdee.K3.FIN.App.Core;  using Kingdee.K3.FIN.Core;  using Kingdee.BOS.Resource;  using System.ComponentModel;  namespace Kingdee.K3.FIN.HS.App.Report  {  /// <summary>  /// 核算错误记录插件  /// </summary>  [Description("核算错误记录插件")]  public class HSErrorLog : SysReportBaseService  {  protected override string BuilderFromWhereSQL(IRptParams filter)  {  StringBuilder sb = new StringBuilder();  //快捷界面自定义参数  if (filter != null && filter.FilterParameter.CustomFilter != null)  {  DynamicObject para = filter.FilterParameter.CustomFilter;  long acctSysId = Convert.ToInt64(para["ACCTGSYSTEMID\_Id"]);  long acctOrgId = Convert.ToInt64(para["ACCTGORGID\_Id"]);  int year = Convert.ToInt32(para["Year"]);  int period = Convert.ToInt32(para["Period"]);  sb.AppendFormat(" from {0} OutAcct left join {1} Chk", "T\_HS\_OUTACCTG", T\_HS\_OUTACCTGCHECK);  sb.AppendFormat(" on Chk.FID = OutAcct.FID inner join {0} process on process.FID = OutAcct.FID", T\_HS\_AcctgProcess);  sb.AppendFormat(" inner join {0} seq on seq.FEntryId = process.FEntryId", "T\_HS\_OutInStockSeq");  sb.AppendFormat(" where OutAcct.FPeriod = {0} and OutAcct.FYear = {1}", period, year);  //从核算范围临时表的参数条件)  if (para["ACCTGRANGEID\_Id"] != null)  {  if (Convert.ToInt64(para["ACCTGRANGEID\_Id"]) != 0)  {  sb.AppendFormat(" and FACCTGRANGEID={0} ", para["ACCTGRANGEID\_Id"]);  }  }  //库存组织  if (para["STOCKORGID\_Id"] != null)  {  if (Convert.ToInt64(para["STOCKORGID\_Id"]) != 0)  {  sb.AppendFormat(" and FAcctgOrgID={0} ", para["STOCKORGID\_Id"]);  }  }  //仓库  if (para["STOCKID\_Id"] != null)  {  if (Convert.ToInt64(para["STOCKORGID\_Id"]) != 0)  {  sb.AppendFormat(" and FStockID={0} ", para["STOCKID\_Id"]);  }  }  if (para["OwnerID\_Id"] != null)  {  if (Convert.ToInt64(para["OwnerID\_Id"]) != 0)  {  sb.AppendFormat(" and FOwnerID={0} ", para["OwnerID\_Id"]);  }  }  //异常单据  if (para["OnlyWrong"] != null)  {  if (Convert.ToBoolean(para["OnlyWrong"]))  {  sb.AppendLine("and (FBillCheckItem is not null or FBillCheckItem <> '0') and (FBillCheckResult is not null or FBillCheckResult <> '0')");  }  }  //append group by  sb.AppendLine(" group by seq.FMaterialID,seq.FStockOrgId,FStockID,FCargoOwnerId ,");  sb.AppendLine(" seq.FBillId ,FBillEntryId,FAcctgSysId ,FAcctgOrgId,FYear,FPeriod ,seq.FQty,process.FAcctgRangeId ");  }  return sb.ToString();  }  }  } | |

## GetIdentityFieldIndexSQL

**备注**

此方法由插件平台报表基类已实现，插件根据索引情况可以自行决定是否重写

创建账表临时表索引sql

**示例代码**

无

## ExecuteBatch

**备注**

此方法由插件平台报表基类实现，插件不用关注

执行sql指令

**示例代码**

无

## GetReportHeaders

**应用场景**

如果账表模型在设计时，没有配置账表单据体字段信息，则通过此方法设置账表列头字段信息

**关键字**

账表列头

**备注**

无

### 案例 – 应付款账龄分析表构建列头信息

**代码来源**

Kingdee.K3.FIN.AP.App.Report.AgingAnalysisService

**需求背景**

应付款账龄分析表构建列头信息

**实现方案**

账表插件在GetReportHeaders事件中，构建ReportHead对象

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Text.RegularExpressions;  using Kingdee.BOS.Core.CommonFilter;  using Kingdee.BOS.Core.Enums;  using Kingdee.BOS.Core.Report;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS.Contracts.Report;  using Kingdee.BOS.Core.List;  using Kingdee.BOS;  using Kingdee.BOS.Core;  using Kingdee.BOS.Util;  using Kingdee.BOS.App.Data;  using Kingdee.K3.FIN.Core;  using Kingdee.BOS.Contracts;  using System.Transactions;  using SummaryField = Kingdee.BOS.Core.Report.SummaryField;  using Kingdee.K3.FIN.App.Core;  namespace Kingdee.K3.FIN.AP.App.Report  {  /// <remarks>  /// 位置：财务会计-应付款管理-应付款账龄分析表  /// </remarks>  public class AgingAnalysisService : SysReportBaseService  {  public override ReportHeader GetReportHeaders(IRptParams filter)  {  ReportHeader header = new ReportHeader();  DynamicObject filterObj = filter.FilterParameter.CustomFilter;  bool isFromFilter = false;  if (filterObj != null)  {  isFromFilter = Convert.ToBoolean(filterObj["IsFromFilter"]);  }  header.AddChild("FContactUnit", "往来单位");  header.AddChild("FCurrencyName", "币别");  if (isFromFilter)  {  header.AddChild("FSettleOrgName", "结算组织");  header.AddChild("FPayOrgName", "收付组织");  header.AddChild("FPurchaseOrgName", "采购组织");  header.AddChild("FPurchaseDeptName", "采购部门");  header.AddChild("FPurchaseGroupName", "采购组");  header.AddChild("FPurchaserName", "采购员");  }  if (filterObj == null)  {  header.AddChild("FBillTypeName", "单据类型");  header.AddChild("FBillNo", "单据编号");  ListHeader lhDate = header.AddChild("FDate", "业务日期");  lhDate.ColType = SqlStorageType.SqlSmalldatetime;  lhDate.Width = 80;  ListHeader lh = header.AddChild("FEndDate", "到期日");  lh.ColType = SqlStorageType.SqlSmalldatetime;  lh.Width = 80;  LoadBalanceColumn(header);  }  else  {  DynamicObjectCollection balanceCollection = filterObj["EntAgingGrpSetting"] as DynamicObjectCollection;  List<ColumnField> listField = filter.FilterParameter.ColumnInfo;  LoadBillColumn(header, Convert.ToBoolean(filterObj["ByBill"]));  LoadBalanceColumn(header, isFromFilter, balanceCollection, listField);  }  //设置列的索引，使其可以按照正常顺序显示  int colIndex = 0;  foreach (var child in header.GetChilds())  {  if (child.GetChildCount() == 0)  {  child.ColIndex = colIndex++;  }  else  {  child.ColIndex = colIndex++;  foreach (var childHeader in child.GetChilds())  {  childHeader.ColIndex = colIndex++;  }  }  }  return header;  }  private void LoadBalanceColumn(  ListHeader header,  bool isFromFilter = false,  DynamicObjectCollection balColumnList = null,  List<ColumnField> listField = null)  {  //原币列  ListHeader orgCurrency = header.AddChild();  orgCurrency.FieldName = "FOrgCurrency";  orgCurrency.Caption = "原币";  orgCurrency.AddChild("FBalanceAmtFor", "尚未付款金额", qlStorageType.SqlDecimal);  // 如果“隐藏列“中包含余额（原币），那么可以添加以下的列  if (listField != null && listField.Any(c => c.Key == "FBalanceAmtFor"))  {  if (balColumnList == null || balColumnList.Count == 0 ||  (balColumnList.Count == 1 && (balColumnList[0]["Section"]).IsNullOrEmptyOrWhiteSpace()))  {  orgCurrency.AddChild("FBalance4AmtFor", "90天以上", SqlStorageType.SqlDecimal);  orgCurrency.AddChild("FBalance3AmtFor", "61-90天", SqlStorageType.SqlDecimal);  orgCurrency.AddChild("FBalance2AmtFor", "31-60天", SqlStorageType.SqlDecimal);  orgCurrency.AddChild("FBalance1AmtFor", "0-30天", SqlStorageType.SqlDecimal);  return;  }  for (int rowIndex = balColumnList.Count - 1; rowIndex >= 0; rowIndex--)  {  string text = balColumnList[rowIndex]["Section"] as string;  if (text.IsNullOrEmptyOrWhiteSpace()) continue;  orgCurrency.AddChild(string.Format("FBalance{0}AmtFor", rowIndex + 1), text, SqlStorageType.SqlDecimal);  }  }  if (!isFromFilter)  {  return;  }  //本位币列  ListHeader standardCurrency = header.AddChild();  standardCurrency.FieldName = "FStandardCurrency";  standardCurrency.Caption = "本位币";  standardCurrency.AddChild("FMasterCurrencyName", "币别");  standardCurrency.AddChild("FBalanceAmt", "尚未付款金额", SqlStorageType.SqlDecimal);  // 如果“隐藏列“中不包含余额（本位币），那么不添加以下的列  if (listField != null && listField.Any(c => c.Key == "FBalanceAmt"))  {  //如果是调整此页面，此时的balColumnList上并没有值，但有默认一空行。故加载默认的四列  if (balColumnList == null || balColumnList.Count == 0 ||  (balColumnList.Count == 1 && (balColumnList[0]["Section"]).IsNullOrEmptyOrWhiteSpace()))  {  standardCurrency.AddChild("FBalance1Amt", "0-30天", SqlStorageType.SqlDecimal);  standardCurrency.AddChild("FBalance2Amt", "31-60天", SqlStorageType.SqlDecimal);  standardCurrency.AddChild("FBalance3Amt", "61-90天", SqlStorageType.SqlDecimal);  standardCurrency.AddChild("FBalance4Amt", "90天以上", SqlStorageType.SqlDecimal);  return;  }  for (int rowIndex = 0; rowIndex < balColumnList.Count; rowIndex++)  {  string text = balColumnList[rowIndex]["Section"] as string;  if (text.IsNullOrEmptyOrWhiteSpace()) continue;  standardCurrency.AddChild(string.Format("FBalance{0}Amt", rowIndex + 1), text, SqlStorageType.SqlDecimal);  }  }  }  }  } | |

## GetReportTitles

**触发时机**

此时已完成账表数据取数并赋值给临时表，完成临时表索引创建，开始准备账表表头信息

**应用场景**

账表服务端取数开发过程中，完成账表取数后，根据账表过滤方案开始准备此账表的表头信息

**关键字**

账表表头

**备注**

无

### 案例 – 销售订单执行汇总表账表表头信息

**代码来源**

Kingdee.K3.SCM.App.Sal.Report.SalCollectRpt

**需求背景**

销售订单执行汇总表根据账表过滤方案操作员配置的条件生成账表表头信息

**实现方案**

账表插件在GetReportTitles方法中，完成构建账表表头信息

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using Kingdee.BOS.Contracts.Report;  using Kingdee.BOS.Core.Report;  using Kingdee.BOS;  using Kingdee.BOS.Core.List;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS.Contracts;  using Kingdee.K3.SCM.Sal.Report.PlugIn;  using System.Collections;  using Kingdee.BOS.App.Data;  using Kingdee.BOS.Core.SqlBuilder;  using Kingdee.BOS.Core;  using Kingdee.BOS.Core.Util;  using Kingdee.BOS.Util;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.Core.Permission;  using Kingdee.BOS.App.Core.BusinessFlow;  using Kingdee.K3.Core;  using Kingdee.BOS.Core.BusinessFlow.ServiceArgs;  using Kingdee.BOS.BusinessEntity.BusinessFlow;  using System.Data;  using Kingdee.BOS.Core.Permission.Objects;  using Kingdee.BOS.Core.CommonFilter;  namespace Kingdee.K3.SCM.App.Sal.Report  {  /// <summary>  /// 销售汇总表数据源插件  /// </summary>  public class SalCollectRpt : SysReportBaseService  {  public override ReportTitles GetReportTitles(IRptParams filter)  {  DynamicObject dyFilter = filter.FilterParameter.CustomFilter;  if (dyFilter["SaleOrgList"] == null || string.IsNullOrWhiteSpace(dyFilter["SaleOrgList"].ToString()))  {  return base.GetReportTitles(filter);  }  else  {  return BuilderTitle(filter);  }  }  private ReportTitles BuilderTitle(IRptParams filter)  {  ReportTitles rptTitl = new ReportTitles();  DynamicObject dyFilter = filter.FilterParameter.CustomFilter;  if (dyFilter != null)  {  if (filterOrgList.Trim().Length > 0)  rptTitl.AddTitle("FSaleOrgListTitle", this.OrgNameList(filterOrgList));  else  rptTitl.AddTitle("FSaleOrgListTitle", "");  string blank = string.Empty;  string strstartday = (GetDataByKey(dyFilter, "SoFromDate") == string.Empty || Convert.ToDateTime(GetDataByKey(dyFilter, "SoFromDate")) == DateTime.MinValue) ? "" : FieldFormatterUtil.GetDateFormatString(this.Context, Convert.ToDateTime(GetDataByKey(dyFilter, "SoFromDate")));  string strendday = (GetDataByKey(dyFilter, "SoToDate") == string.Empty || Convert.ToDateTime(GetDataByKey(dyFilter, "SoToDate")) == DateTime.MinValue) ? "" : FieldFormatterUtil.GetDateFormatString(this.Context, Convert.ToDateTime(GetDataByKey(dyFilter, "SoToDate")));  if ((!strstartday.IsNullOrEmptyOrWhiteSpace()) || (!strendday.IsNullOrEmptyOrWhiteSpace()))  {  blank = " 至 ";  }  rptTitl.AddTitle("FDeliveryStartDay", string.Format("{0}{1}{2}", strstartday, blank, strendday));  blank = string.Empty;  //销售员范围  string salesFrom = string.Empty;  string salseTo = string.Empty;  DynamicObject saleFrom = dyFilter["SalesFrom"] as DynamicObject;  string saleFromNumber = string.Empty;  string saleFromName = string.Empty;  if (saleFrom != null)  {  saleFromNumber = GetDataByKey(saleFrom, "NUMBER");  saleFromName = GetDataByKey(saleFrom, "NAME");  }  DynamicObject saleTo = dyFilter["SalesTo"] as DynamicObject;  string saleToNumber = string.Empty;  string saleToName = string.Empty;  if (saleTo != null)  {  saleToNumber = GetDataByKey(saleTo, "NUMBER");  saleToName = GetDataByKey(saleTo, "NAME");  }  if (saleFrom != null || saleTo != null)  { blank = "--"; }  else  {  blank = "全部";  }  if (saleFrom != null)  {  salesFrom = string.Format("{0}({1})", saleFromNumber, saleFromName);  }  if (saleTo != null)  {  salseTo = string.Format("{0}({1})", saleToNumber, saleToName);  }  rptTitl.AddTitle("FSalesLimt", string.Format("{0}{1}{2}", salesFrom, blank, salseTo));  blank = string.Empty;  //客户范围  string custFromInfo = string.Empty;  string custToInfo = string.Empty;  DynamicObject custFrom = dyFilter["CustomerFrom"] as DynamicObject;  string custFromNumber = string.Empty;  string custFromName = string.Empty;  if (custFrom != null)  {  custFromNumber = GetDataByKey(custFrom, "NUMBER");  custFromName = GetDataByKey(custFrom, "NAME");  }  DynamicObject custTo = dyFilter["CustomerTo"] as DynamicObject;  string custToNumber = string.Empty;  string custToName = string.Empty;  if (custTo != null)  {  custToNumber = GetDataByKey(custTo, "NUMBER");  custToName = GetDataByKey(custTo, "NAME");  }  if (custFrom != null || custTo != null)  { blank = "--"; }  else  {  blank = "全部";  }  if (custFrom != null)  {  custFromInfo = string.Format("{0}({1})", custFromNumber, custFromName);  }  if (custTo != null)  {  custToInfo = string.Format("{0}({1})", custToNumber, custToName);  }  rptTitl.AddTitle("FCustomerLimit", string.Format("{0}{1}{2}", custFromInfo, blank, custToInfo));  blank = string.Empty;  //销售订单范围  if (GetDataByKey(dyFilter, "SaleOFrom").ToString().Trim() != string.Empty || GetDataByKey(dyFilter, "SaleOTo").Trim() != string.Empty)  { blank = "--"; }  else if (filterSFrom == string.Empty && filterSTo == string.Empty)  {  blank = "全部";  }  rptTitl.AddTitle("FIndentLimit", string.Format("{0}{1}{2}", GetDataByKey(dyFilter, "SaleOFrom"), blank, GetDataByKey(dyFilter, "SaleOTo")));  blank = string.Empty;  //物料范围  string matFrom = string.Empty;  string matTo = string.Empty;  DynamicObject materailFrom = dyFilter["MaterialFrom"] as DynamicObject;  string materailFromNumber = string.Empty;  string materailFromName = string.Empty;  if (materailFrom != null)  {  materailFromNumber = GetDataByKey(materailFrom, "NUMBER");  materailFromName = GetDataByKey(materailFrom, "NAME");  }  DynamicObject materailTo = dyFilter["MaterialTo"] as DynamicObject;  string materailToNumber = string.Empty;  string materailToName = string.Empty;  if (materailTo != null)  {  materailToNumber = GetDataByKey(materailTo, "NUMBER");  materailToName = GetDataByKey(materailTo, "NAME");  }  if (materailFrom != null || materailTo != null)  { blank = "--"; }  else  {  blank = "全部";  }  if (materailFromNumber != string.Empty)  {  matFrom = string.Format("{0}({1})", materailFromNumber, materailFromName);  }  if (materailToNumber != string.Empty)  {  matTo = string.Format("{0}({1})", materailToNumber, materailToName);  }  rptTitl.AddTitle("FMaterialLimit", string.Format("{0}{1}{2}", matFrom, blank, matTo));  blank = string.Empty;  }  return rptTitl;  }  protected string GetDataByKey(DynamicObject dy, string key)  {  if (dy != null && dy[key] != null && !string.IsNullOrWhiteSpace(dy[key].ToString()))  {  return dy[key].ToString();  }  return string.Empty;  }  }  } | |

## AnalyzeDspCloumn

**备注**

此方法由插件平台报表基类实现，插件不用关注

**示例代码**

无

## AfterCreateTempTable

**触发时机**

账表数据创建之后，给插件提供时机处理数据

**插件示例**

无

## GetReportData

**触发时机**

准备根据过滤条件获取账表数据

**应用场景**

账表插件根据账表返回数据，对数据做修改

**关键字**

账表数据

**备注**

无

### 案例 – 供应商供货汇总表取数

**代码来源**

Kingdee.K3.SCM.App.Purchase.Report.OfferGoodsSummaryRpt

**需求背景**

此账表数据返回后，更新辅助属性的值

**实现方案**

账表插件在父类GetReport方法后，循环DataTable账表数据，更新辅助属性的值

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS;  using Kingdee.BOS.App.Data;  using Kingdee.BOS.Contracts;  using Kingdee.BOS.Contracts.Report;  using Kingdee.BOS.Core.List;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.Core.Permission.Objects;  using Kingdee.BOS.Core.Report;  using Kingdee.BOS.Core.SqlBuilder;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS.Util;  using Kingdee.K3.BD.Contracts;  using Kingdee.K3.Core.SCM;  using Kingdee.K3.Core.SCM.Args;  using Kingdee.K3.SCM.Contracts;  using System;  using System.Collections.Generic;  using System.Data;  using System.Diagnostics;  using System.Text;  namespace Kingdee.K3.SCM.App.Purchase.Report  {  /// <summary>  /// 供应商供货汇总表  /// </summary>  [Description("供应商供货汇总表")]  public class OfferGoodsSummaryRpt : SysReportBaseService  {  protected override DataTable GetReportData(string tablename, IRptParams filter)  {  DataTable dtPageData = base.GetReportData(tablename, filter);  for (int i = 0; i < dtPageData.Rows.Count; i++)  {  if (dtPageData.Rows[i]["FFLEX"] != DBNull.Value  && Convert.ToInt64(dtPageData.Rows[i]["FFLEX"]) > 0)  {  dtPageData.Rows[i]["FFLEXVALUE"] = string.Join(";", GetAuxPropExtValues(this.Context, Convert.ToInt64(dtPageData.Rows[i]["FFLEX"])));  }  int temp = 0;  if (Int32.TryParse(  Convert.ToString(dtPageData.Rows[i]["FUNITID"]), out temp))  {  dtPageData.Rows[i]["FUNITNAME"] = VMIReportAppUtil.GetFormatedUnitValue(this.Context, Convert.ToInt32(dtPageData.Rows[i]["FUNITID"]));  }  }  return dtPageData;  }  }  } | |

## GetRowsCount

**触发时机**

账表取数已结束，并填充到DataTable临时表中，此时获取本页数据行数

**应用场景**

获取本页账表数据行数

**关键字**

账表数据

**备注**

普通账表一般不用关注此方法，已由账表基类实现

### 案例 – 库存台账分页账表获取当前页数据行数

**代码来源**

Kingdee.K3.SCM.App.Stock.Report.StockBookRpt

**需求背景**

库存台账分页账表获取当前页账表数据行数

**实现方案**

根据分页账表当前页所在索引数，获取分页账表临时表数据对象，通过此对象获得临时表表名，查找此表有多少行数据

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Linq;  using System.Text;  using Kingdee.BOS;  using Kingdee.BOS.App.Data;  using Kingdee.BOS.Core.List;  using Kingdee.BOS.Core.Report;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS.Resource;  using Kingdee.K3.Core.SCM;  using Kingdee.BOS.Core.Enums;  using Kingdee.BOS.Core.Metadata.FieldElement;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.Core.Report.PlugIn;  using Kingdee.BOS.Util;  using Kingdee.BOS.Contracts;  using Kingdee.K3.SCM.App.Stock.Report.StockDetail;  using Kingdee.K3.Core.BD;  using Kingdee.K3.SCM.App.Core;  using Kingdee.BOS.Orm.Metadata.DataEntity;  using Kingdee.BOS.Core.Permission.Objects;  namespace Kingdee.K3.SCM.App.Stock.Report  {  [DisplayName("库存台账")]  [Description("库存台账服务端插件")]  public class StockBookRpt : StockRptCommon  {  public override int GetRowsCount(IRptParams filter)  {  string tableName = string.Empty;  if (CacheMapTempTableNameList != null && this.CacheMapTempTableNameList.Count > this.CurrentListPosition)  {  MoveRptCurrentInfo info = this.CacheMapTempTableNameList[this.CurrentListPosition];  if (info != null)  {  tableName = info.TableName;  }  }  if (tableName.Length == 0) return 0;  string sumGroupKey = this.ReportProperty.GroupSummaryInfoData.GroupLevelFieldName;  string sumGroupingKey = this.ReportProperty.GroupSummaryInfoData.GroupingFieldName;  string sfilter = "WHERE 1=1";  //有分组汇总才进行相关的选项控制  if (!string.IsNullOrWhiteSpace(sumGroupKey) && !string.IsNullOrWhiteSpace(filter.FilterParameter.GroupbyString))  {  if (filter.IsOnlyQuerySumData)  {  sfilter += string.Format(" AND {0} > 0 ", sumGroupKey);  }  if (this.IsNoDspSubtotalRow(filter))  {  sfilter += string.Format(" AND {0} <> 0 ", sumGroupingKey);  }  if (this.IsNoDspSumtotalRow(filter))  {  sfilter += string.Format(" AND {0} <> 1 ", sumGroupingKey);  }  }  if (sfilter.Length < 10)  {  sfilter = "";  }  string sSql = string.Format("SELECT ISNULL(COUNT(1),0) as fcount FROM {0} {1} ", tableName, sfilter);  return DBUtils.ExecuteScalar<int>(this.Context, sSql, 0, null);  }  }  } | |

## GetSummaryData

**备注**

此方法由插件平台报表基类实现，插件不用关注

**示例代码**

无

## GetSummaryColumnInfo

**触发时机**

获取账表分组汇总字段信息、汇总类型

**应用场景**

获取账表分组汇总字段信息，即在账表取数的所有字段中，哪些字段需要进行分组汇总

**关键字**

分组汇总字段 汇总类型

**备注**

无

### 案例 – 销售订单执行汇总表分组汇总字段信息

**代码来源**

Kingdee.K3.SCM.App.Sal.Report.SalCollectRpt

**需求背景**

销售订单执行汇总表准备分组汇总字段信息

**实现方案**

账表插件在GetSummaryColumnInfo方法中，配置账表分组汇总字段

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using Kingdee.BOS.Contracts.Report;  using Kingdee.BOS.Core.Report;  using Kingdee.BOS;  using Kingdee.BOS.Core.List;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS.Contracts;  using Kingdee.K3.SCM.Sal.Report.PlugIn;  using System.Collections;  using Kingdee.BOS.App.Data;  using Kingdee.BOS.Core.SqlBuilder;  using Kingdee.BOS.Core;  using Kingdee.BOS.Core.Util;  using Kingdee.BOS.Util;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.Core.Permission;  using Kingdee.BOS.App.Core.BusinessFlow;  using Kingdee.K3.Core;  using Kingdee.BOS.Core.BusinessFlow.ServiceArgs;  using Kingdee.BOS.BusinessEntity.BusinessFlow;  using System.Data;  using Kingdee.BOS.Core.Permission.Objects;  using Kingdee.BOS.Core.CommonFilter;  namespace Kingdee.K3.SCM.App.Sal.Report  {  /// <summary>  /// 销售汇总表数据源插件  /// </summary>  public class SalCollectRpt : SysReportBaseService  {  public override List<SummaryField> GetSummaryColumnInfo(IRptParams filter)  {  List<BOS.Core.Report.SummaryField> list = new List<BOS.Core.Report.SummaryField>();  list.Add(new BOS.Core.Report.SummaryField("FSOAMOUNT", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FSAMOUNT", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FALREADYAMOUNT", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FNOTAMOUNT", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FRETURNPAMOUNT", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FRETURNSAMOUNT", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FNVOECEAMOUNT", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FNNVOECEAMOUNT", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FRECEIPTAMOUNT", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FNRECEIPTAMOUNT", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FCHARGEOFFAMOUNT", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FSOQTY", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FSQTY", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FALREADYQTY", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FNOTQTY", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FRETURNPQTY", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FRETURNSQTY", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FINVOECEQTY", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FNINVOECEQTY", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FRECQTY", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FRECAMOUNT", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FRECNQTY", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FRECNAMOUNT", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  list.Add(new BOS.Core.Report.SummaryField("FWRITEOFFAMOUNT", BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM));  return list;  }  }  } | |

## GetSummaryColumsSQL

**触发时机**

获取账表分组汇总sql

**应用场景**

获取账表分组汇总字段信息后，获取分组汇总sql指令

**关键字**

分组汇总sql

**备注**

无

### 案例 – 销售订单执行汇总表分组汇总SQL指令

**代码来源**

Kingdee.K3.SCM.App.Sal.Report.SalCollectRpt

**需求背景**

销售订单执行汇总表准备分组汇总Sql指令

**实现方案**

账表插件在GetSummaryColumnSQL方法中，获取分组汇总Sql指令

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using Kingdee.BOS.Contracts.Report;  using Kingdee.BOS.Core.Report;  using Kingdee.BOS;  using Kingdee.BOS.Core.List;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS.Contracts;  using Kingdee.K3.SCM.Sal.Report.PlugIn;  using System.Collections;  using Kingdee.BOS.App.Data;  using Kingdee.BOS.Core.SqlBuilder;  using Kingdee.BOS.Core;  using Kingdee.BOS.Core.Util;  using Kingdee.BOS.Util;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.Core.Permission;  using Kingdee.BOS.App.Core.BusinessFlow;  using Kingdee.K3.Core;  using Kingdee.BOS.Core.BusinessFlow.ServiceArgs;  using Kingdee.BOS.BusinessEntity.BusinessFlow;  using System.Data;  using Kingdee.BOS.Core.Permission.Objects;  using Kingdee.BOS.Core.CommonFilter;  namespace Kingdee.K3.SCM.App.Sal.Report  {  /// <summary>  /// 销售汇总表数据源插件  /// </summary>  public class SalCollectRpt : SysReportBaseService  {  protected virtual string GetSummaryColumsSQL(List<Core.Report.SummaryField> summaryFields)  {  StringBuilder sb = new StringBuilder();  foreach (SummaryField field in summaryFields)  {  switch (field.SummaryType)  {  case Kingdee.BOS.Core.Enums.BOSEnums.Enu\_SummaryType.COUNT:  sb.AppendFormat("Count(\*) AS {0}", field.Key);  sb.Append(",");  break;  case Kingdee.BOS.Core.Enums.BOSEnums.Enu\_SummaryType.SUM:  sb.AppendFormat("SUM({0}) AS {0}", field.Key);  sb.Append(",");  break;  case Kingdee.BOS.Core.Enums.BOSEnums.Enu\_SummaryType.MAX:  sb.AppendFormat("MAX({0}) AS {0}", field.Key);  sb.Append(",");  break;  case Kingdee.BOS.Core.Enums.BOSEnums.Enu\_SummaryType.MIN:  sb.AppendFormat("MIN({0}) AS {0}", field.Key);  sb.Append(",");  break;  case Kingdee.BOS.Core.Enums.BOSEnums.Enu\_SummaryType.AVERAGE:  sb.AppendFormat("ROUND(AVG({0}),10) AS {0}", field.Key);  sb.Append(",");  break;  default:  break;  }  }  sb.Remove(sb.Length - 1, 1);  return sb.ToString();  }  }  } | |

## CloseReport

**触发时机**

关闭账表页面时

**应用场景**

关闭账表页面时，清除账表插件使用到的临时表资源等业务操作

**关键字**

账表关闭

**备注**

无

### 案例 – 供货商供货汇总表关闭时，清除临时表

**代码来源**

Kingdee.K3.SCM.App.Purchase.Report.OfferGoodsSummaryRpt

**需求背景**

供货商供货汇总表关闭时，清除账表插件中使用到的自定义的临时表资源

**实现方案**

账表插件在CloseReport事件中，清除临时表资源

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using Kingdee.BOS;  using Kingdee.BOS.App.Data;  using Kingdee.BOS.Contracts;  using Kingdee.BOS.Contracts.Report;  using Kingdee.BOS.Core.List;  using Kingdee.BOS.Core.Metadata;  using Kingdee.BOS.Core.Permission.Objects;  using Kingdee.BOS.Core.Report;  using Kingdee.BOS.Core.SqlBuilder;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.BOS.Util;  using Kingdee.K3.BD.Contracts;  using Kingdee.K3.Core.SCM;  using Kingdee.K3.Core.SCM.Args;  using Kingdee.K3.SCM.Contracts;  using System;  using System.Collections.Generic;  using System.Data;  using System.Diagnostics;  using System.Text;  namespace Kingdee.K3.SCM.App.Purchase.Report  {  /// <summary>  /// 供应商供货汇总表  /// </summary>  [Description("供应商供货汇总表")]  public class OfferGoodsSummaryRpt : SysReportBaseService  {  public override void CloseReport()  {  if (tempTables.Count > 0)  {  var service = Kingdee.K3.SCM.App.ServiceHelper.GetService<ITemporaryTableService>();  service.DropTable(this.Context, new HashSet<string>(tempTables));  }  }  }  } | |

## GetList

**备注**

分页账表获取分页条件的方法

**示例代码**

|  |  |
| --- | --- |
| C# |  |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using Kingdee.BOS.Contracts.Report;  using System.ComponentModel;  using Kingdee.BOS.Core.Report;  using Kingdee.BOS.App.Data;  using Kingdee.BOS;  using System.Data;  using Kingdee.BOS.Orm.DataEntity;  using Kingdee.K3.FIN.FA.Common.Core;  using Kingdee.BOS.Core.List;  namespace Kingdee.K3.FIN.FA.App.Report  {  /// <summary>  /// 变动过程记录表服务器插件  /// </summary>  [Description("变动过程记录表服务器插件")]  public class CardChangeRecordService : SysReportBaseService  {  public override DataTable GetList(IRptParams filter)  {  // 在最先执行的方法里设置快捷过滤条件  QickFilter = new ChangeRecordFilter(filter);  string sql = GetListSql(filter);  return DBUtils.ExecuteDataSet(this.Context, sql).Tables[0];  }  /// <summary>  /// 获取分页报表列表SQL  /// </summary>  private string GetListSql(IRptParams filter)  {  StringBuilder listSql = new StringBuilder(); ;  string seniorFilter = "";  if (!string.IsNullOrWhiteSpace(filter.FilterParameter.FilterString))  {  seniorFilter = seniorFilter + string.Format(" AND {0}", filter.FilterParameter.FilterString);  }  // 发生过变动的卡片记录才显示（卡片拆分、合并不算是变动，无新卡片产生）  string changeFilter = string.Format("where FAssetCurStatus<>'{0}' AND FAssetCurStatus<'5'", ((int)FAEnums.AssetCurStatus.AddNew).ToString());  switch (QickFilter.ShowItem)  {  case ShowItems.Finance:  {  string alterTypeFilter = " and (Faltertype like '%2%') ";  listSql.AppendFormat("SELECT FAssetID, FNumber from ({0} {1} {2} {5}) listdata {3} {4} GROUP BY FAssetID, FNumber", GetFinDataSelectSql(), GetFinDataFromSql(), GetQickFilterSql(), changeFilter, seniorFilter, alterTypeFilter);  break;  }  case ShowItems.Device:  {  string alterTypeFilter = " and (Faltertype like '%4%') ";  listSql.AppendFormat("SELECT FAssetID, FNumber from ({0} {1} {2} {5}) listdata {3} {4} GROUP BY FAssetID, FNumber", GetDeviceDataSelectSql(), GetDeviceDataFromSql(), GetQickFilterSql(), changeFilter, seniorFilter, alterTypeFilter);  break;  }  case ShowItems.Distrbute:  {  string alterTypeFilter = " and (Faltertype like '%5%') ";  listSql.AppendFormat("SELECT FAssetID, FNumber from ({0} {1} {2} {5}) listdata {3} {4} GROUP BY FAssetID, FNumber", GetDistributeDataSelectSql(), GetDistributeDataFromSql(), GetQickFilterSql(), changeFilter, seniorFilter, alterTypeFilter);  break;  }  default:  {  listSql.Append("SELECT FAssetID, FNumber FROM T\_FA\_CARD WHERE FAlterID=0");  break;  }  }  return listSql.ToString();  }  }  } | |

## GetTreeNodes

**备注**

树形账表获取左边树形控件数据包的方法

**示例代码**

无