|  |
| --- |
| **插件开发应用案例介绍** |

收货单提供以下功能：

1.       增加下拉列表，显示单据头的所有字段；

2.       在分录菜单上增加库存查询(FQueryInventory)菜单项；

3.       点击库存查询时，查询分录上当前焦点所在物料的库存（STK\_InvSumQuery）；

STK\_Inventory

4.       查询库存时按组织隔离，只查询当前组织的库存；

5.       当前分录物料F8时，显示所有组织的物料；

6.       暂存时清空单据类型的值；

7.       物料基础资料增加字段有效期至（F\_MCY\_ExpiryDate）；

8.       F8时只显示有效期〉今天的物料；

9.       保存判断物料的库存，如果〉100则提示“库存〉100，是否入库？”；

10.   保存后锁定“收料部门”、“收料员”；

11.   保存后自动记录收料日志（MCY\_stk\_ReceiptLog）；

**操作步骤：**

1.       增加下拉列表，显示单据头的所有字段；

a)         新建（打开）收货单插件工程（MyDev.K3.SCM.Stock.Business.PlugIn）；

b)         重载OnInitialize方法，定义List<EnumItem>用于存储下拉列表枚举值；

c)         通过this.View.BusinessInfo.GetFieldList()方法获取所有字段；

d)         通过this.View.GetControl<ComboFieldEditor>方法获取界面上的下拉列表控件；

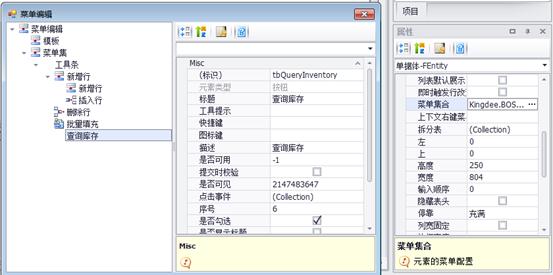
e)         SetComboItems绑定值；

f)          代码如下：

|  |  |
| --- | --- |
| **C#** |  |
| public override void OnInitialize(InitializeEventArgs e)  {      base.OnInitialize(e);      List<EnumItem> list = new List<EnumItem>();      foreach (Field field in this.View.BusinessInfo.GetFieldList())      {          EnumItem item = new EnumItem();          item.Caption = field.Name;          item.EnumId = field.Key;          item.Value = field.Key;          list.Add(item);      }      this.View.GetControl<ComboFieldEditor>("FCombo").SetComboItems(list);  } | |

2.       在分录菜单上增加库存查询(tbQueryInventory)菜单项；

a)         运行IDE，选择单据体-菜单集合，新增菜单：



b)         保存；

3.       点击库存查询时，查询分录上当前焦点所在物料的库存；

a)         打开插件工程，重载方法EntryBarItemClick

b)         判断BarItemKey==库存查询(tbQueryInventory)

c)         取当前分录行

d)         设置ListShowParameter参数，打开表单

这里介绍2种获取当前分录字段数据的方法：

TryGetEntryCurrentRow：获取单据体当前行，返回是否取到值以及行数据和行号；

另外一种方法：

先获取单据体当前行号，再取指定行数据；

2种方法没什么区别。

示例代码如下：

|  |  |
| --- | --- |
| **C#** |  |
| public override void EntryBarItemClick(BarItemClickEventArgs e)  {      base.EntryBarItemClick(e);      if (e.BarItemKey == "tbQueryInventory")      {          ShowQueryInventory();      }  }    private void ShowQueryInventory()  {      DynamicObject row;  int rowIndex;  // 直接获取当前分录行返回的是分录行对象。      if (this.Model.TryGetEntryCurrentRow("FEntity", out row, out rowIndex))      {          ListShowParameter parameter = new ListShowParameter();          parameter.FormId = "STK\_Inventory";    // 即时库存的FormId          parameter.MultiSelect = false;          parameter.ListFilterParameter.Filter = string.Format(" FMaterialId = '{0}' ", Convert.ToString(row["FBase\_Id"]));          this.View.ShowForm(parameter);      }  } | |

取单据体当前行号，再取指定行的字段数据的方法如下：

|  |  |
| --- | --- |
| **C#** |  |
| private void ShowQueryInventory()  {  // 获取当前行      int rowIndex = this.Model.GetEntryCurrentRowIndex("FEntity");      if (rowIndex > -1) // 判断当前行有数据  {      // 取指定行的物料（ide中设置key为FBase）字段数据          DynamicObject materialObj = (DynamicObject)this.Model.GetValue("FBase", rowIndex);          ListShowParameter parameter = new ListShowParameter();          parameter.FormId = "STK\_Inventory";          parameter.MultiSelect = false;          parameter.ListFilterParameter.Filter = string.Format(" FMaterialId = '{0}' ", materialObj["Id"].ToString());          this.View.ShowForm(parameter);      }  } | |

调试状态下，可以屏蔽代码parameter.ListFilterParameter.Filter看看过滤条件的效果。

注意：ListFilterParameter 的Filter属性设置的字段是用IDE中的字段标识。

4.       查询库存时按组织隔离，只查询当前组织的库存：

a)         增加过滤条件，组织=当前组织

b)         parameter.ListFilterParameter.Filter = string.Format(" FORGID ={0} ", this.Model.Context.CurrentOrganizationInfo.ID) });

5.       当前分录物料F8时，显示所有组织的物料；

a)   重载AuthPermissionBeforeF7Select方法，设置参数IsIsolationOrg = false；

b)   同样，如果需要F8时控制只显示当前组织的物料，该参数设置为true。

注意：

在BOS系统中，默认是按组织隔离的，即非共享基础资料，在F8时都是只显示当前组织的物料。

代码示例如下：

|  |  |
| --- | --- |
| **C#** |  |
| public override void AuthPermissionBeforeF7Select(AuthPermissionBeforeF7SelectEventArgs e)  {      base.AuthPermissionBeforeF7Select(e);      if (e.FieldKey == "FBase")      {          e.IsIsolationOrg = false;      }  } | |

6.       暂存时清空单据类型的值；

|  |  |
| --- | --- |
| **C#** |  |
| public override void BeforeDoOperation(BeforeDoOperationEventArgs e)  {      base.BeforeDoOperation(e);      if (e.Operation.FormOperation.Operation.Equals("DRAFT", StringComparison.OrdinalIgnoreCase))      {          this.Model.SetValue("FBillTypeID", null);      }  } | |

7.       F8时只显示审核日期〉2014-03-22的供应商；

a)         重载BeforeF7Select事件；

b)         设置列表过滤参数ListFilterParameter的属性Filter；

|  |  |
| --- | --- |
| **C#** |  |
| public override void BeforeF7Select(BeforeF7SelectEventArgs e)  {      base.BeforeF7Select(e);      if (e.FieldKey == "FSupplierId1")      {          string filter = " FCreateDate > '2014-03-20' ";          if (string.IsNullOrEmpty(e.ListFilterParameter.Filter))          {              e.ListFilterParameter.Filter = filter;          }          else          {              e.ListFilterParameter.Filter += " AND " + filter;          }      }  } | |

8.       保存判断物料的库存，如果〉100则提示“库存〉100，是否入库？”；

a)         新建收货单服务插件工程MyDev.K3.SCM.App.Stock.ServicePlugIn；

b)         定义保存服务类SaveServicePlugIn，继承自AbstractOperationServicePlugIn；

c)         重载OnAddValidators方法；

代码示例如下：

|  |  |
| --- | --- |
| **C#** |  |
| public override void OnAddValidators(AddValidatorsEventArgs e)  {      base.OnAddValidators(e);       SaveValidator saveValidator = new SaveValidator();      saveValidator.EntityKey = "FBillHead";      e.Validators.Add(saveValidator);  } | |

d)         定义保存校验类SaveValidator，继承自AbstractValidator；

e)         重载方法：Validate：

                         i.              获取单据体分录数据，取到物料Id；

                       ii.              查询物料库存；

                      iii.              检查库存是否〉100；

                      iv.              构造校验结果信息；

代码示例：

|  |  |
| --- | --- |
| **C#** |  |
| public override void Validate(ExtendedDataEntity[] dataEntities, ValidateContext validateContext, Kingdee.BOS.Context ctx)  {      if (dataEntities == null || dataEntities.Length == 0)      {          return;      }        Dictionary<long, decimal> dictErrMaterialId = new Dictionary<long, decimal>();      //取所有物料      List<long> listMaterialId = new List<long>();      foreach (ExtendedDataEntity entityObj in dataEntities)      {          DynamicObjectCollection collection = (DynamicObjectCollection)entityObj["FEntity"];          foreach (DynamicObject rowObj in collection)          {              listMaterialId.Add((long)rowObj["FBase\_Id"]);          }      }      string sql = " select a.FMATERIALID, sum(a.FBASEQTY) FQTY from  T\_STK\_INVENTORY a where exists (select 1 from TABLE(fn\_StrSplit(@FMATERIALID, ',',1)) t where t.FID = a. FMATERIALID) group by FMATERIALID ";      SqlParam param = new SqlParam("@FMATERIALID", KDDbType.udt\_inttable, listMaterialId.Distinct().ToArray());      using (IDataReader dr = DBUtils.ExecuteReader(this.Context, sql, param))      {          while (dr.Read())          {              decimal qty = Convert.ToDecimal(dr["FQTY"]);              if (qty > 100)             {                  dictErrMaterialId.Add(Convert.ToInt64(dr["FMATERIALID"]), qty);              }          }      }        foreach (ExtendedDataEntity entityObj in dataEntities)      {          DynamicObjectCollection collection = (DynamicObjectCollection)entityObj["FEntity"];          foreach (DynamicObject rowObj in collection)          {              if (dictErrMaterialId.ContainsKey((long)rowObj["FBase\_Id"]))              {                  ValidationErrorInfo errinfo = new ValidationErrorInfo("FMATERIALID", Convert.ToString(entityObj.DataEntity["Id"]), entityObj.DataEntityIndex, Convert.ToInt32(rowObj["Id"]), "SaveValidator", "库存数量大于100", "校验失败", ErrorLevel.Error);                  validateContext.AddError(entityObj, errinfo);              }          }      }  } | |

f)          重载方法：Validate：

9.       保存后锁定“收料部门”、“收料员”；

a)         锁定字段的方法：this.View.LockField；

b)         该锁定与事务无关，只要在客户端保存后事件（AfterBarItemClick）处理即可；

c)         “收料部门”、“收料员”的key可以在IDE设计器中拷贝；

代码如下：

|  |  |
| --- | --- |
| **C#** |  |
| public override void AfterBarItemClick(AfterBarItemClickEventArgs e)  {      base.AfterBarItemClick(e);      if (e.BarItemKey == "tbSave")      {          this.View.LockField("FBase1", true);          this.View.LockField("FBase2", true);      }  } | |

10.   保存后自动记录收料日志（KDV\_stk\_ReceiptLog）；

根据需求设计收料日志表：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 名称 | 类型 | 说明 |
| KDV\_ID | 日志ID | int | 自增长 |
| KDV\_UserID | 操作用户 | Int | 关联用户表ID |
| KDV\_Date | 操作时间 | Datetime | 缺省getdate |
| KDV\_Content | 日志内容 | Nvarchar(2000) |  |

保存有2种方法：

方法1：

a）  在IDE中定义收料日志基础资料；

b）  打开收货单服务插件保存服务类SaveServicePlugIn；

c）  根据收料日志基础资料的元数据定义，创建动态实体对象；

d）  设置对象属性值；

e）  调用BusinessDataService服务的保存方法保存动态实体对象；

代码如下：

|  |  |
| --- | --- |
| **C#** |  |
| public override void AfterExecuteOperationTransaction(AfterExecuteOperationTransaction e)  {      base.AfterExecuteOperationTransaction(e);        MetaDataService metaService = new MetaDataService();      FormMetadata formMetaData = (FormMetadata)metaService.Load(this.Context, "1823871d-b9cf-4d8b-93af-39c0c37011a5");      DynamicObjectType dt = formMetaData.BusinessInfo.GetDynamicObjectType();      DynamicObject obj = new DynamicObject(dt);      dt.Properties["KDV\_UserID\_Id"].SetValueFast(obj, this.Context.UserId);      dt.Properties["KDV\_Content"].SetValueFast(obj, "保存");      ISaveService saveService = ServiceHelper.GetService<ISaveService>();      saveService.Save(this.Context, new DynamicObject[] { obj });  } | |

方法2：

a）  自定义收料日志表；

b）  获取日志的自增长（序列）值；

c）  执行insert；

代码如下：

|  |  |
| --- | --- |
| **C#** |  |
| public override void AfterExecuteOperationTransaction(AfterExecuteOperationTransaction e)  {      base.AfterExecuteOperationTransaction(e);        SequenceReader sequence = new SequenceReader(this.Context);      int[] ids = (int[])sequence.GetSequence("KDV\_stk\_ReceiptLog", 1);        int id = ids[0];        string sql = " insert into KDV\_stk\_ReceiptLog(FID, KDV\_UserID, KDV\_Content) values (@FID, @KDV\_UserID, @KDV\_Content) ";      SqlParam[] sqlParams = new SqlParam[3];      sqlParams[0] = new SqlParam("@FID", KDDbType.Int64, id);      sqlParams[1] = new SqlParam("@KDV\_UserID", KDDbType.Int64, this.Context.UserId);      sqlParams[2] = new SqlParam("@KDV\_Content", KDDbType.String, "保存");        DBUtils.Execute(this.Context, sql, sqlParams);  } | |