# iDempiere 快速开发指南

|  |  |
| --- | --- |
| 上海菱威深信息技術有限公司  携手成长、共享感动 |  |

[iDempiere 快速开发指南 1](#_Toc378334612)

[前言 3](#_Toc378334613)

[关于本指南： 3](#_Toc378334614)

[文档管理者： 3](#_Toc378334615)

[版权 3](#_Toc378334616)

[iDempiere的历史 3](#_Toc378334617)

[正文 5](#_Toc378334618)

[第一章： 5](#_Toc378334619)

[章节说明： 5](#_Toc378334620)

[开发环境的准备： 5](#_Toc378334621)

[第二章： 10](#_Toc378334622)

[式样说明： 10](#_Toc378334623)

[式样-机能对照： 10](#_Toc378334624)

[第三章 Model实现 11](#_Toc378334625)

[STEP1 创建数据表 11](#_Toc378334626)

[STEP2 创建窗口，标签信息： 13](#_Toc378334627)

[STEP3 创建菜单 16](#_Toc378334628)

[STEP4 创建X,I基类 16](#_Toc378334629)

[STEP5 创建Model类 17](#_Toc378334630)

[STEP6 配置OSGI 18](#_Toc378334631)

[第4章 callout的实现 20](#_Toc378334632)

[STEP1 创建Callout的基类CalloutFactory和CalloutFromFactory 20](#_Toc378334633)

[STEP2 业务逻辑的实现CalloutCMaterial.checkNameShort 21](#_Toc378334634)

[第5章 processButtom的实现 22](#_Toc378334635)

[STEP1 创建ProcessFactory的配置文件 22](#_Toc378334636)

[STEP2 创建ProcessFactory的实现方法 22](#_Toc378334637)

[STEP3添加业务逻辑 23](#_Toc378334638)

[STEP5 添加按钮 25](#_Toc378334639)

[STEP6 完成 25](#_Toc378334640)

[第6章 processSave的实现 26](#_Toc378334641)

[STEP1 完成第5章STEP1-STEP2 26](#_Toc378334642)

[STEP2 修改MCMaterialLine.java类，添加触发器 26](#_Toc378334643)

[第7章 jasper报表 29](#_Toc378334644)

[STEP1 完成jrxml文件 29](#_Toc378334645)

[STEP2 系统中配置报表信息 30](#_Toc378334646)

[STEP3 设置报表的参数 30](#_Toc378334647)

[STEP5 创建调用报表的方法 32](#_Toc378334648)

[附录 33](#_Toc378334649)

[callout的实现方式： 33](#_Toc378334650)

[Model中的触发器 34](#_Toc378334651)

[Process中的参数 34](#_Toc378334652)

## 前言

### 关于本指南：

本指南的侧重于向程序员介绍如何利用iDempiere中提供的框架进行开发，知其然，不知其所以然。如遇问题，请自行上下求索。

本指南大部分内容都来之于<http://www.adempiere.com/>。

### 文档管理者：

* 部门：管理统括
* 氏名：于洋
* 电子邮件：yang.yu@ivision-china.cn
* 部门：技术统括
* 氏名：陈健
* 电子邮件：jian.a.chen@ivision-china.cn

### 版权

本文档著作权归属于上海菱威深信息技术有限公司（iVision Shanghai Co.,Ltd.）。在没有文档管理者书面认可的情况下，禁止全部或者部分拷贝、转载该文档的内容。

### iDempiere的历史

iDempiere起源于Compiere，在2006年左右，因为Compiere进行了商业化，分化出了iDempiere的前身ADempiere，而在2011年因为项目组管理层的原因，诞生出了iDempiere。

iDempiere不是简单的从ADempiere中分化出，而是在ADdempiere的基础上，通过OSGI进行了重构，提高了灵活性。利于商业开发。

基础要求：

至少开发过java程序的程序猿就可以了。如果想要深入了解iDempiere中使用的框架，还是需要一定的功力的。

## 正文

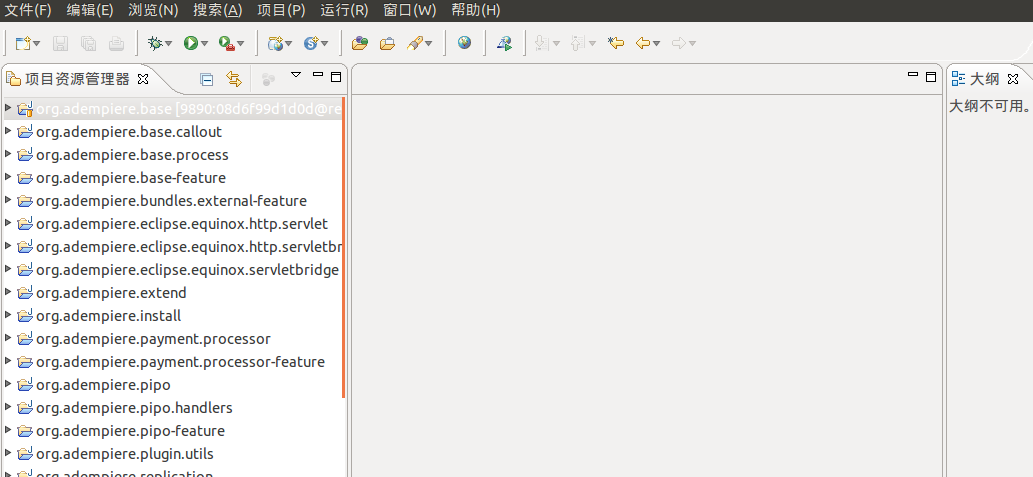
### 第一章：

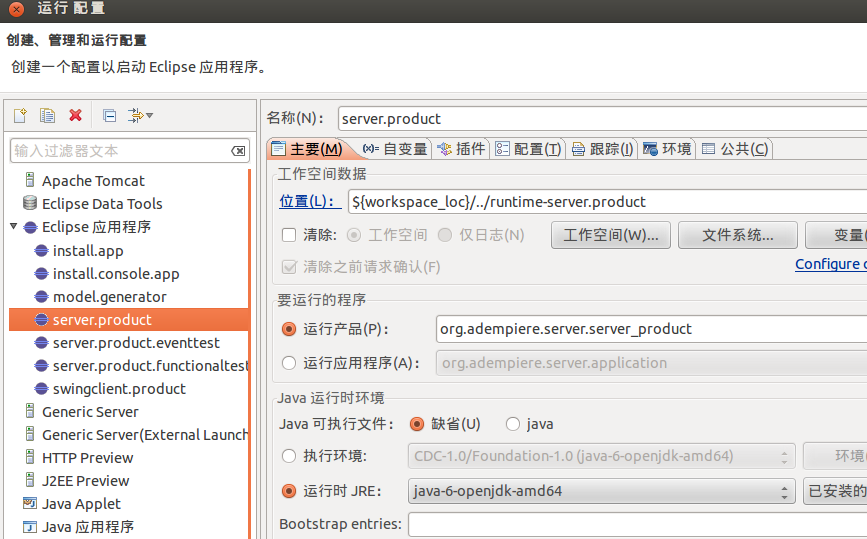
#### 章节说明：

#### 开发环境的准备：

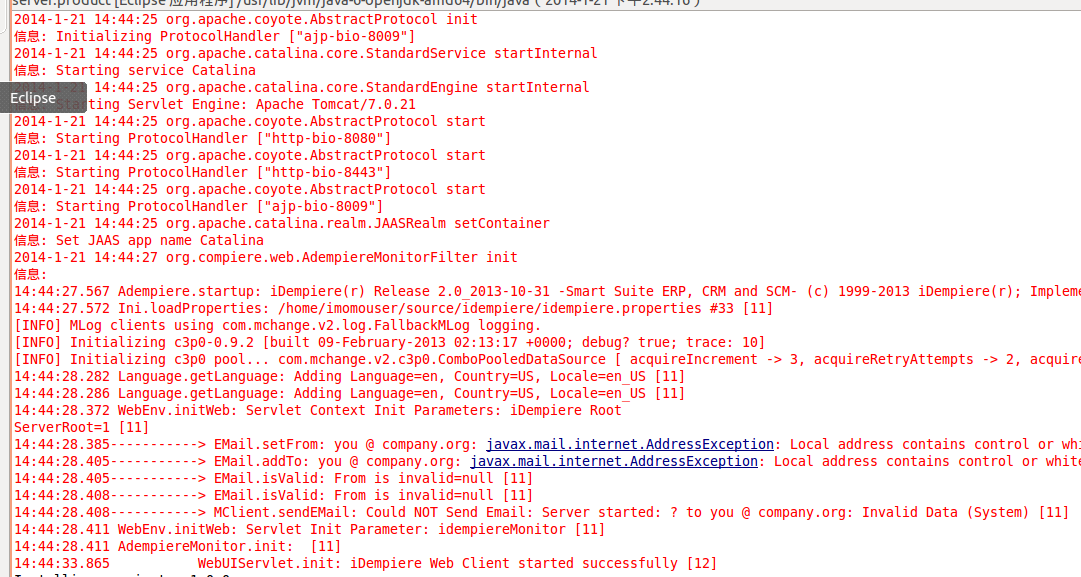
你需要有HG作为代码管理器，JDK1.7(最低版本要求1.6),postgresSQL作为数据库，Eclipse(kepler以上版本)作为开发工具。参照TCN\_开发环境搭建手册（Windows版）\_V1。

环境搭建完成后，大致开发环境看起来是这样的。



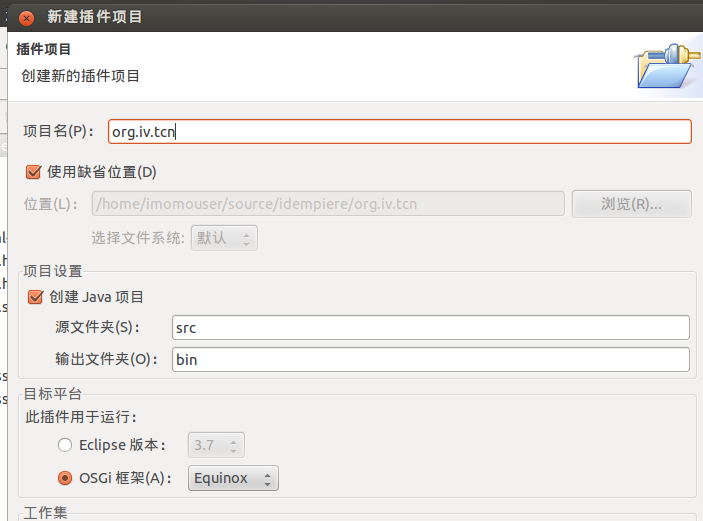
通过server.product来启动服务。

启动完成后



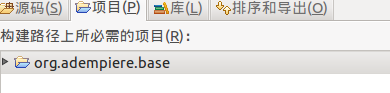
为了方便演示开发内容，这次将所有的开发内容都放置在新增的插件上。

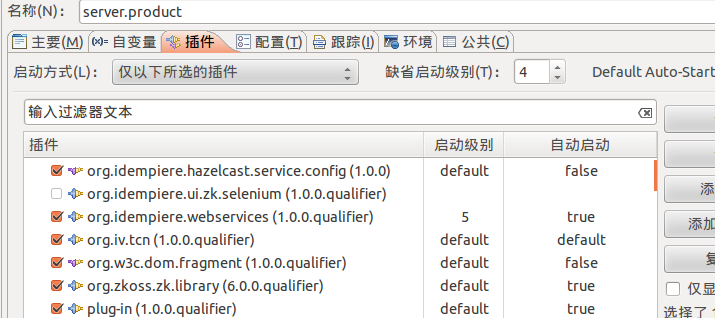
插件的新增方法：











在METE-INF目录下MANIFEST.MF



添加

*Manifest-Version: 1.0*

*Bundle-ManifestVersion: 2*

*Bundle-Name: Tcn*

*Bundle-SymbolicName: org.iv.tcn*

*Bundle-Version: 1.0.0.qualifier*

*Import-Package: org.osgi.framework;version="1.3.0",*

*org.osgi.service.event;version="1.2.0"*

*Require-Bundle: org.adempiere.base;bundle-version="1.0.0",*

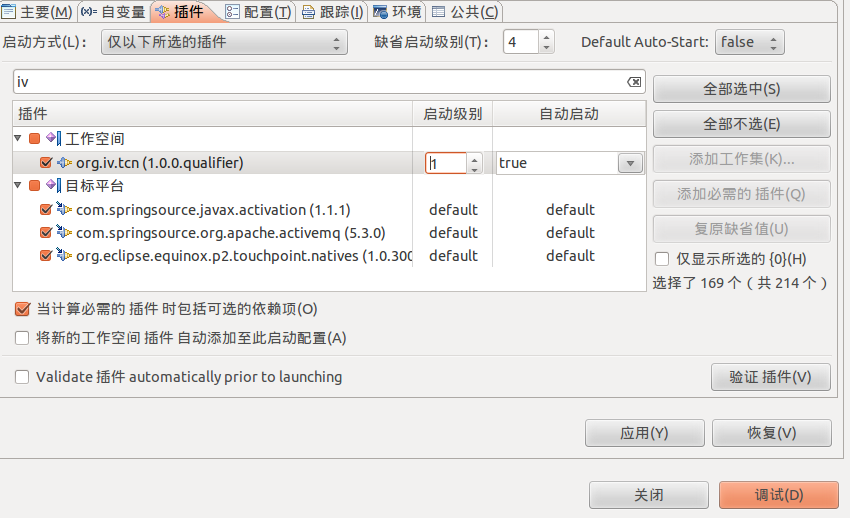
*org.adempiere.plugin.utils;bundle-version="0.0.0"*

*Bundle-Activator: org.adempiere.plugin.utils.AdempiereActivator*

*Service-Component: OSGI-INF/\*.xml*

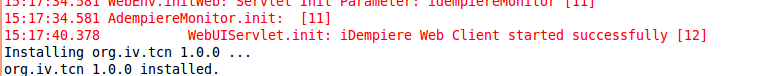
*Bundle-RequiredExecutionEnvironment: JavaSE-1.6*

修改插件启动时的参数



启动级别一般大于4即可

配置完成后，可以再次启动server.product



在iDempiere中创建EntityType。作为修改的标识符，用作版本和Patch的控制。



### 第二章：

#### 式样说明：

通过head-明细的模式，维护物料信息，每条物料head信息会有多条物料明细信息可以维护。

物料head表上有Name，nameshort,Active,total 4个业务字段。

物料明细表上有Name，nameshort，C\_Material,Active4个业务字段

式样1：物料head表中的Nameshort的值取决于Name的值，当Name的值长度小于等于4时，Nameshort的值等于Name的值，当Name值的长度大于4是，Nameshort的值等于Name值的前4个字符。

式样2：物料head表上Total的值等于这条物料信息的明细信息的行数。

式样3：物料head表上可以通过按钮来计算Total的值，也可以在保存明细表时，自动计算Head表上Total的值。

式样4：物料head表上有按钮可以打印japser报表。

#### 式样-机能对照：

Model 物料head表和物料明细表的增删改查功能。

Callout 式样1。

ProcessButtom 式样3，按钮

PorcessSave 式样3，保存

Jaser 式样4

### 第三章 Model实现

从本章开始，进入具体的开发过程。

#### STEP1 创建数据表

在数据库中创建2张数据表

Head表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *CREATE TABLE C\_Material (* | |  | |  |  |
| *AD\_CLIENT\_ID* | *NUMERIC(10) NOT NULL,* | | | |  |
| *AD\_ORG\_ID* | *NUMERIC(10) NOT NULL,* | | | |  |
| *ISACTIVE* | *CHAR(1) DEFAULT 'Y' NOT NULL,* | | | | |
| *CREATED* | *DATE DEFAULT NOW() NOT NULL,* | | | | |
| *CREATEDBY* | *NUMERIC(10) NOT NULL,* | | | |  |
| *UPDATED* | *DATE DEFAULT NOW() NOT NULL,* | | | | |
| *UPDATEDBY* | *NUMERIC(10) NOT NULL,* | | | |  |
| *C\_Material\_UU* | *VARCHAR(100),* | | |  |  |
| *C\_MATERIAL\_ID* | *NUMERIC(10,0)* | | | NOT NULL, | |
| *NAME* | *VARCHAR(100)* | | | NOT NULL, | |
| *NAMESHORT* | *VARCHAR(100)* | | | NOT NULL， | |
| *total* | *NUMERIC(10,0)* | | |  |  |
| *);* |  | |  |  |  |
| *ALTER TABLE C\_Material ADD PRIMARY KEY (C\_MATERIAL\_ID );* | | | | | |

明细表：

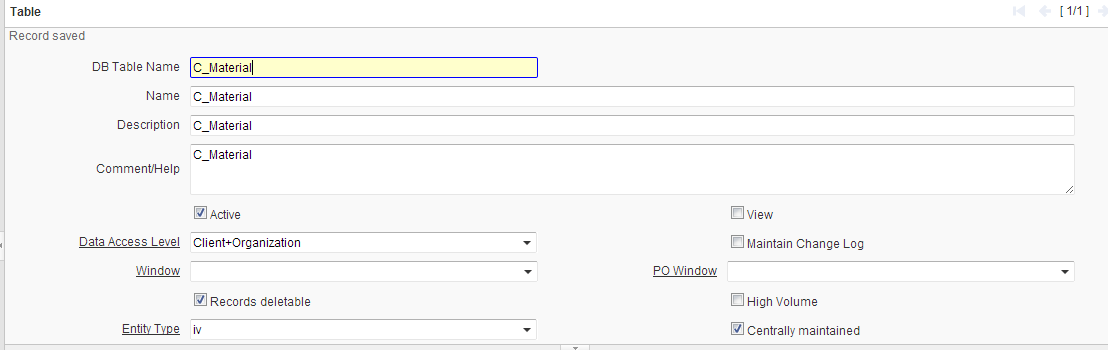
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *CREATE TABLE C\_MaterialLine (* | |  |  |  |
| *AD\_CLIENT\_ID* | *NUMERIC(10) NOT NULL,* | | |  |
| *AD\_ORG\_ID* | *NUMERIC(10) NOT NULL,* | | |  |
| *ISACTIVE* | *CHAR(1) DEFAULT 'Y' NOT NULL,* | | | |
| *CREATED* | *DATE DEFAULT NOW() NOT NULL,* | | | |
| *CREATEDBY* | *NUMERIC(10) NOT NULL,* | | |  |
| *UPDATED* | *DATE DEFAULT NOW() NOT NULL,* | | | |
| *UPDATEDBY* | *NUMERIC(10) NOT NULL,* | | |  |
| *C\_MaterialLine\_UU* | *VARCHAR(100),* | |  |  |
| *C\_MaterialLine\_ID* | *NUMERIC(10,0)* | NOT NULL, | |  |
| *NAME* | *VARCHAR(100)* | | NOT NULL, | |
| *NAMESHORT* | *VARCHAR(100)* | | NOT NULL, | |
| *C\_Material\_ID* | *NUMERIC(10,0)* | NOT NULL |  |  |
| *);* |  |  |  |  |
| *ALTER TABLE C\_MaterialLine ADD PRIMARY KEY (C\_MaterialLine\_ID )* | | | | |

其中：

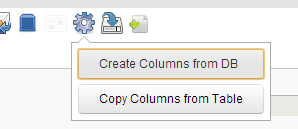
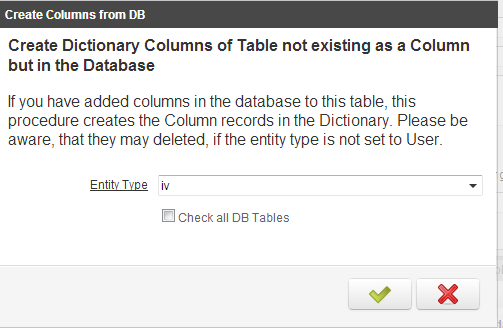
红字部分为iDempiere的必须字段，每个表都需要有。且不能更改字段名。

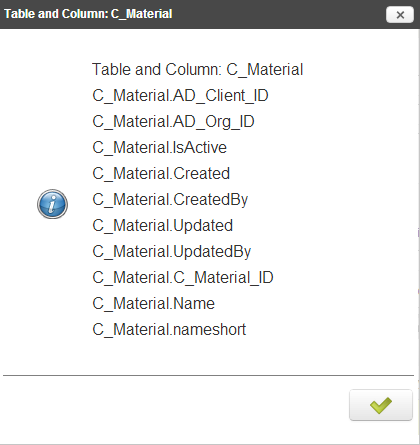
绿字部分也为iDempiere的必须字段，但是字段名必须以表名+”ID”,表名+”UU”来表示。

在iDempiere中配置这2个表和列的信息。

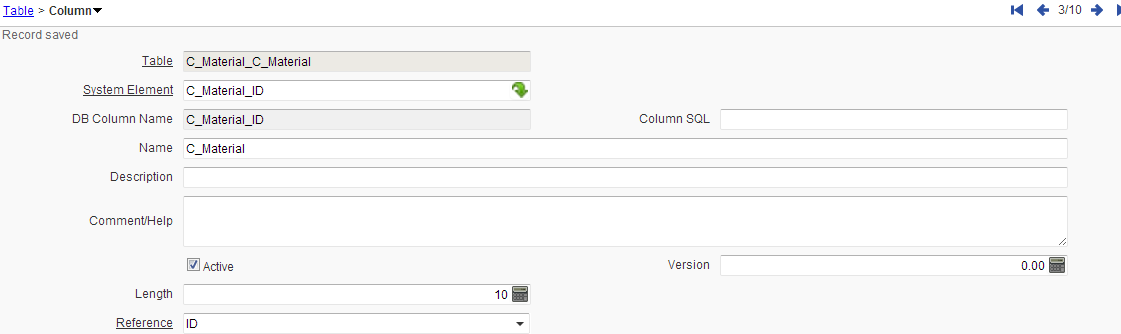


通过CreatColumnfromDB按钮来生成列

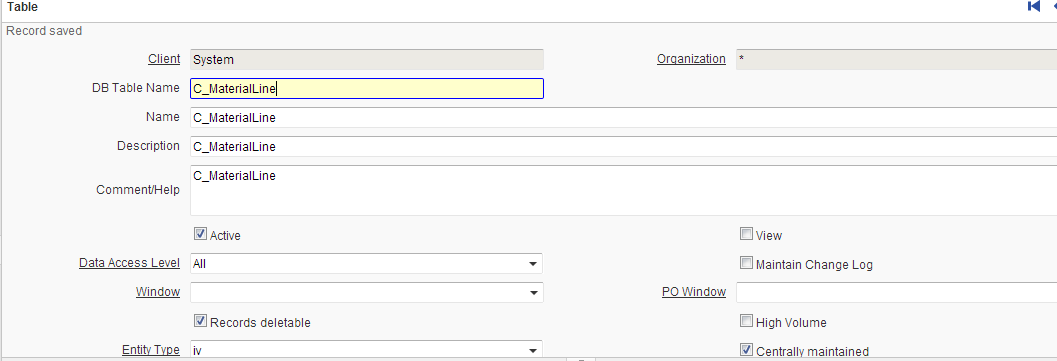
 

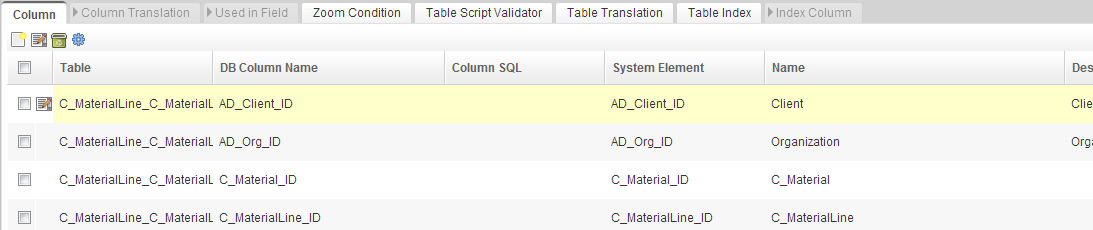


导出完成后，检查每个列的信息，注意主键的reference应该为ID



同样的，创建明细表的信息



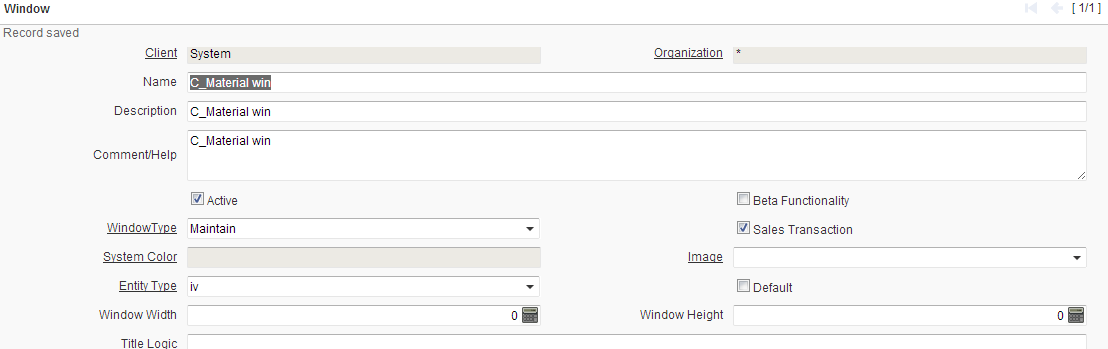


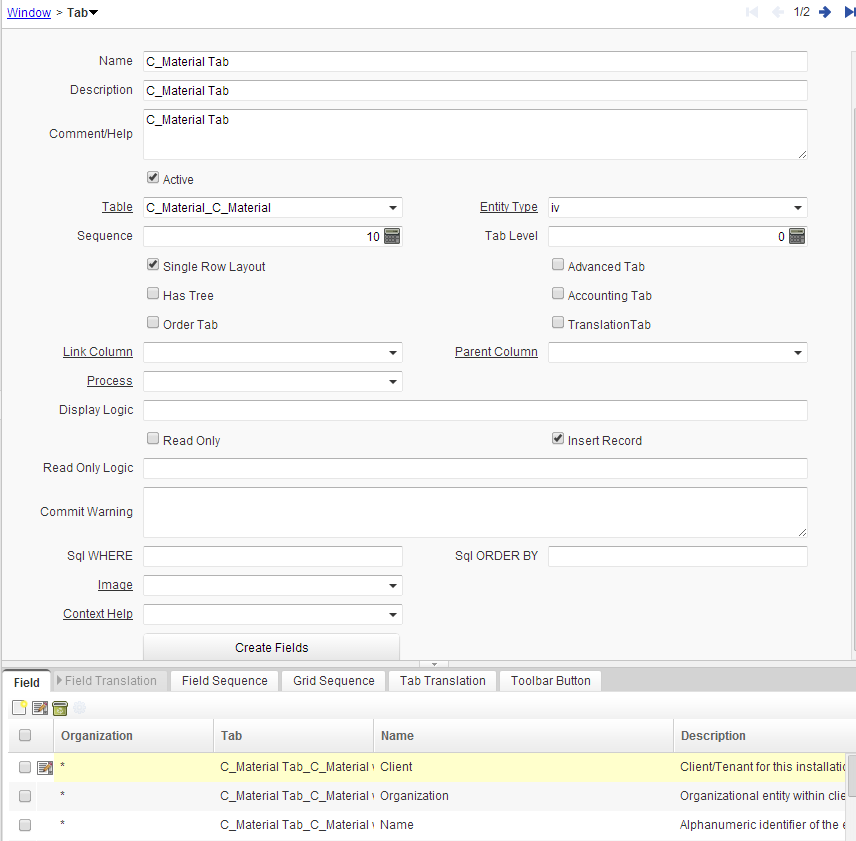
注意用于关联head表的列的reference属性应该为TableDirect

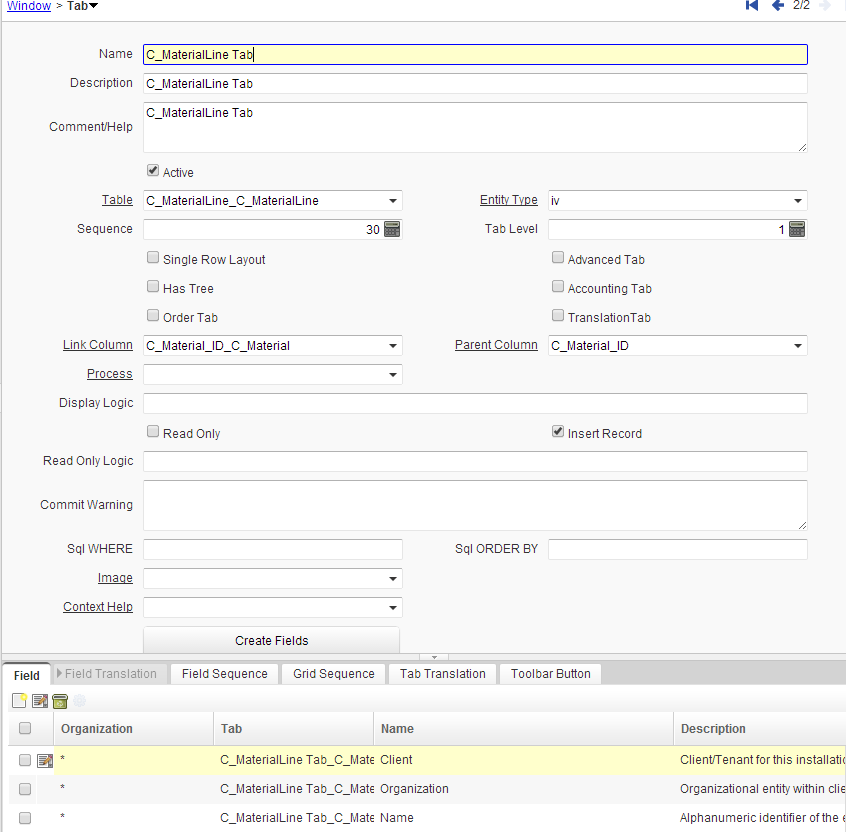


#### STEP2 创建窗口，标签信息：

窗口作为管理单位，包含有material和materialline2个页签，这2个页签对应2个表。

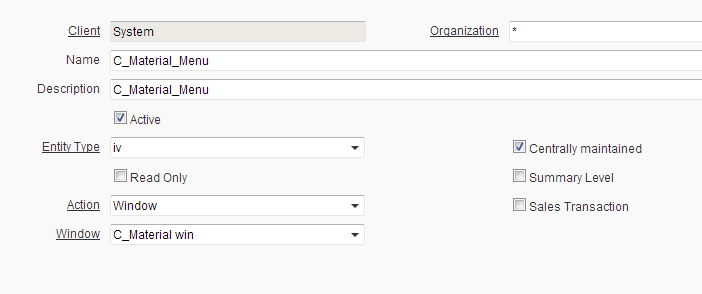




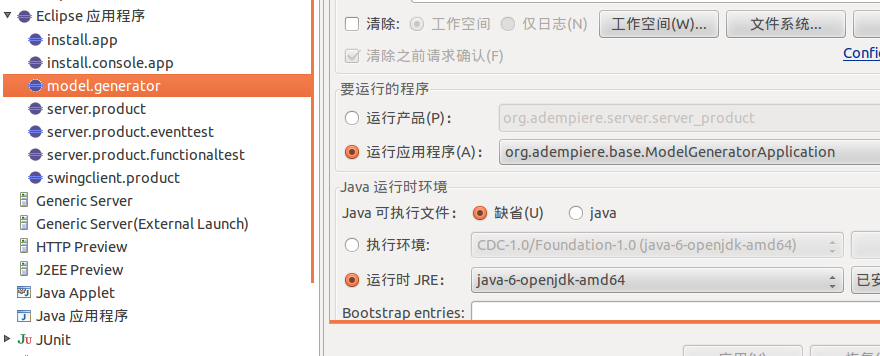


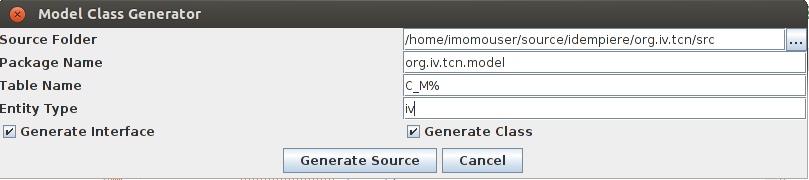
注意materialline页签中用于关联head表的Link Column和ParentColumn的配置

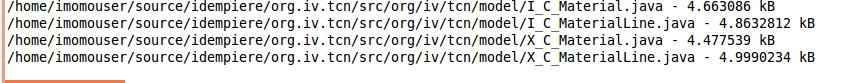
#### STEP3 创建菜单



#### STEP4 创建X,I基类







#### STEP5 创建Model类

MCMaterial：

*package org.iv.tcn.model;*

*import java.util.Properties;*

*import java.sql.PreparedStatement;*

*import java.sql.ResultSet;*

*import org.iv.tcn.model.X\_C\_Material;*

*public class MCMaterial extends X\_C\_Material {*

*/\*\* Process Message \*/*

*private static final long serialVersionUID =9879869;*

*public MCMaterial(Properties ctx, int C\_Material\_ID,*

*String trxName) {*

*super(ctx, C\_Material\_ID, trxName);*

*// TODO Auto-generated constructor stub*

*}*

*public MCMaterial(Properties ctx, ResultSet rs, String trxName) {*

*super(ctx, rs, trxName);*

*// TODO Auto-generated constructor stub*

*}*

*public static MCMaterial getMaterial(Properties ctx,int C\_Material\_ID,String trxName){*

*return new MCMaterial(ctx,C\_Material\_ID,trxName);*

*}*

*}*

MCMaterialLine：

*package org.iv.tcn.model;*

*import java.util.Properties;*

*import java.sql.PreparedStatement;*

*import java.sql.ResultSet;*

*import org.iv.tcn.model.X\_C\_MaterialLine;*

*public class MCMaterialLine extends X\_C\_MaterialLine {*

*/\*\*Process Message \*/*

*private static final long serialVersionUID =9879869;*

*public MCMaterialLine(Properties ctx, int C\_MaterialLine\_ID,*

*String trxName) {*

*super(ctx, C\_MaterialLine\_ID, trxName);*

*// TODO Auto-generated constructor stub*

*}*

*public MCMaterialLine(Properties ctx, ResultSet rs, String trxName) {*

*super(ctx, rs, trxName);*

*// TODO Auto-generated constructor stub*

*}*

*public static MCMaterialLine getMaterial(Properties ctx,int C\_MaterialLine\_ID,String trxName){*

*return new MCMaterialLine(ctx,C\_MaterialLine\_ID,trxName);*

*}*

*}*

#### STEP6 配置OSGI

在OSGI-INF目录下，创建TCN\_ModelFactory.xml文件。

文件内容如下：

*<?xml version="1.0" encoding="UTF-8"?>*

*<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0" name="org.iv.tcn.model.factory">*

*<implementation class="org.iv.tcn.model.TCN\_ModelFactory"/>*

*<property name="service.ranking" type="Integer" value="1"/>*

*<service>*

*<provide interface="org.adempiere.base.IModelFactory"/>*

*</service>*

*</scr:component>*

创建ModelFactory方法

*package org.iv.tcn.model;*

*import java.sql.ResultSet;*

*import org.adempiere.base.IModelFactory;*

*import org.compiere.model.PO;*

*import org.compiere.util.Env;*

*public class TCN\_ModelFactory implements IModelFactory {*

*@Override*

*public Class<?> getClass(String tableName) {*

*if (tableName.equals(I\_C\_Material.Table\_Name)) {*

*return MCMaterial.class;*

*} else*

*if (tableName.equals(I\_C\_MaterialLine.Table\_Name)) {*

*return MCMaterialLine.class;*

*}*

*return null;*

*}*

*@Override*

*public PO getPO(String tableName, int Record\_ID, String trxName) {*

*if (tableName.equals(I\_C\_Material.Table\_Name)) {*

*return new MCMaterial(Env.getCtx(), Record\_ID, trxName);*

*} else*

*if (tableName.equals(I\_C\_MaterialLine.Table\_Name)) {*

*return new MCMaterialLine(Env.getCtx(), Record\_ID, trxName);*

*}*

*return null;*

*}*

*@Override*

*public PO getPO(String tableName, ResultSet rs, String trxName) {*

*if (tableName.equals(I\_C\_Material.Table\_Name)) {*

*return new MCMaterial(Env.getCtx(), rs, trxName);*

*} else*

*if (tableName.equals(I\_C\_MaterialLine.Table\_Name)) {*

*return new MCMaterialLine(Env.getCtx(), rs, trxName);*

*}*

*return null;*

*}*

*}*

到这里为止，head和明细信息的简单增删改查就完成了。

### 第4章 callout的实现

#### STEP1 创建Callout的基类CalloutFactory和CalloutFromFactory

TCN\_CalloutFromFactory：

*package org.iv.tcn.callout;*

*import java.util.Properties;*

*import java.util.logging.Level;*

*import org.adempiere.base.IColumnCallout;*

*import org.compiere.model.GridField;*

*import org.compiere.model.GridTab;*

*import org.jfree.util.Log;*

*public class TCN\_CalloutFromFactory extends CalloutCMaterial implements IColumnCallout {*

*@Override*

*public String start(Properties ctx, int WindowNo, GridTab mTab,*

*GridField mField, Object value, Object oldValue) {*

*if (mField.getColumnName().equals("nameshort"))*

*return checkNameShort(ctx, WindowNo, mTab, mField,value,oldValue);*

*return null;*

*}*

*}*

TCN\_CalloutFactory：

*package org.iv.tcn.callout;*

*import java.util.ArrayList;*

*import java.util.List;*

*import org.adempiere.base.IColumnCallout;*

*import org.adempiere.base.IColumnCalloutFactory;*

*import org.iv.tcn.model.MCMaterial;*

*public class TCN\_CalloutFactory implements IColumnCalloutFactory {*

*@Override*

*public IColumnCallout[] getColumnCallouts(String tableName,*

*String columnName) {*

*List<IColumnCallout> list = new ArrayList<IColumnCallout>();*

*if (tableName.equals(MCMaterial.Table\_Name) && columnName.equals(MCMaterial.COLUMNNAME\_nameshort)) {*

*list.add(new TCN\_CalloutFromFactory());*

*}*

*return list != null ? list.toArray(new IColumnCallout[0]) : new IColumnCallout[0];*

*}*

*}*

#### STEP2 业务逻辑的实现CalloutCMaterial.checkNameShort

*package org.iv.tcn.callout;*

*import java.sql.PreparedStatement;*

*import java.sql.ResultSet;*

*import java.sql.SQLException;*

*import java.util.Properties;*

*import java.util.logging.Level;*

*import org.compiere.model.CalloutEngine;*

*import org.compiere.model.GridField;*

*import org.compiere.model.GridTab;*

*public class CalloutCMaterial extends CalloutEngine {*

*public String checkNameShort(Properties ctx, int WindowNo, GridTab mTab, GridField mField, Object value, Object oldValue)*

*{*

*String newNameShort = (String)value;*

*String oldNameShort = (String)oldValue;*

*//if (!mTab.getTableName().equals("C\_Material") || !mField.getColumnName().equals("nameshort")){*

*if(newNameShort == null || newNameShort.equals(""))*

*{*

*return "";*

*}*

*//String Name = mTab.getField("Name").getValue().toString();*

*String Name = mTab.getField("Name").getValue().toString();*

*if(Name.length()<=4){*

*mTab.setValue("nameshort", Name);*

*return "";*

*}*

*else if(newNameShort.length()<=4)*

*{*

*mTab.setValue("nameshort", Name.substring(0, 4));*

*return "";*

*}*

*else if(newNameShort.length()>=4 && newNameShort.substring(0,4).equals(Name.substring(0, 4))){*

*return "";*

*}*

*else{*

*//log.log(Level.SEVERE, "nameshort",mTab.getName());*

*mTab.setValue("nameshort", Name.substring(0, 4));*

*//return "sub!=";*

*}*

*return "";*

*}*

*}*

STEP 3 完成

无需配置，重新加载插件后，即可使用。

[附录：callout的实现方式](#_callout的实现方式：)

### 第5章 processButtom的实现

#### STEP1 创建ProcessFactory的配置文件

TCN\_ProcessFactory.xml:

*<?xml version="1.0" encoding="UTF-8"?>*

*<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0" name="org.iv.tcn.process.factory">*

*<implementation class="org.iv.tcn.process.TCN\_ProcessFactory"/>*

*<property name="service.ranking" type="Integer" value="1"/>*

*<service>*

*<provide interface="org.adempiere.base.IProcessFactory"/>*

*</service>*

*</scr:component>*

#### STEP2 创建ProcessFactory的实现方法

TCN\_ProcessFactory.java

*package org.iv.tcn.process;*

*import org.adempiere.base.IProcessFactory;*

*import org.compiere.process.ProcessCall;*

*public class TCN\_ProcessFactory implements IProcessFactory {*

*@Override*

*public ProcessCall newProcessInstance(String className) {*

*ProcessCall process = null;*

*try {*

*Class<?> clazz = getClass().getClassLoader().loadClass(className);*

*process = (ProcessCall) clazz.newInstance();*

*} catch (Exception e) {*

*}*

*return process;*

*}*

*}*

#### STEP3添加业务逻辑

UpdateTotalFromLine.java:

*package org.iv.tcn.process;*

*import org.compiere.process.ProcessInfoParameter;*

*import org.compiere.process.SvrProcess;*

*import java.sql.PreparedStatement;*

*import java.sql.ResultSet;*

*import java.sql.SQLException;*

*import java.util.logging.Level;*

*import org.compiere.util.DB;*

*import org.iv.tcn.model.MCMaterial;*

*public class UpdateTotalFromLine extends SvrProcess {*

*private int p\_Record\_ID = 0;*

*protected String doIt() throws Exception*

*{*

*int cnt = 0;*

*int CMaterialID = 0;*

*MCMaterial Mat = new MCMaterial(getCtx(), p\_Record\_ID, get\_TrxName());*

*CMaterialID = Mat.getC\_Material\_ID();*

*try*

*{*

*String sqlCnt = "Select count(\*) from C\_MaterialLine "*

*+ "where C\_Material\_ID=?";*

*PreparedStatement pstmtcnt = DB.prepareStatement(sqlCnt, get\_TrxName());*

*pstmtcnt.setInt(1, CMaterialID);*

*ResultSet rs = pstmtcnt.executeQuery();*

*while (rs.next()){*

*cnt = rs.getInt(1);*

*}*

*}*

*catch (SQLException e)*

*{*

*log.log(Level.SEVERE, "", e);*

*return "error";*

*}*

*return "total = " + cnt ;*

*}*

*@Override*

*protected void prepare() {*

*/\**

*ProcessInfoParameter[] para = getParameter();*

*for (int i = 0; i < para.length; i++)*

*{*

*String name = para[i].getParameterName();*

*if (para[i].getParameter() == null)*

*;*

*if (name.equals("C\_Material\_ID"))*

*p\_Record\_ID = para[i].getParameterAsInt();*

*else*

*log.log(Level.SEVERE, "Unknown Parameter: " + name);*

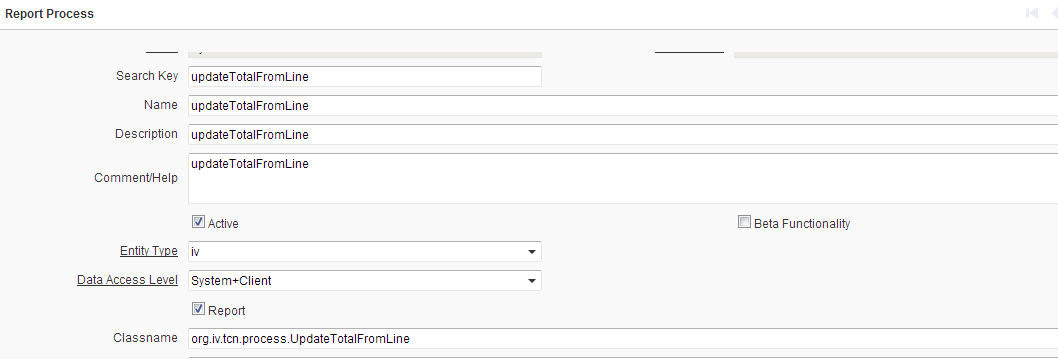
*}*

*\*/*

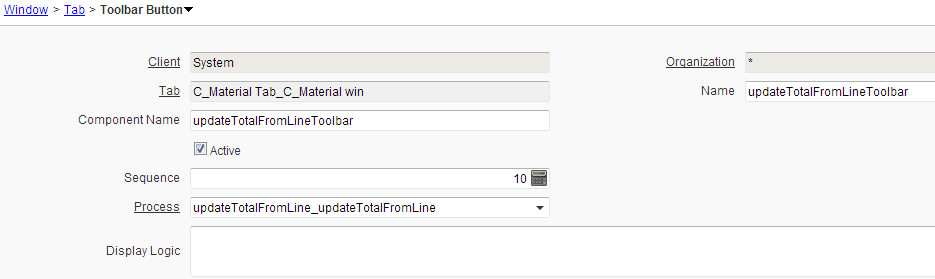
*p\_Record\_ID = getRecord\_ID();*

*}*

*}*STEP4 配置Process

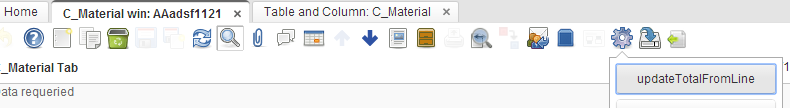


#### STEP5 添加按钮



#### STEP6 完成

刷新缓存后就完成了。



### 第6章 processSave的实现

#### STEP1 完成第5章STEP1-STEP2

#### STEP2 修改MCMaterialLine.java类，添加触发器

参照附录[model中的触发器](#_Model中的触发器)

*package org.iv.tcn.model;*

*import java.util.Properties;*

*import java.util.logging.Level;*

*import java.sql.PreparedStatement;*

*import java.sql.ResultSet;*

*import java.sql.SQLException;*

*import org.compiere.model.MAssetAddition;*

*import org.compiere.model.MInvoiceLine;*

*import org.compiere.util.DB;*

*import org.iv.tcn.model.X\_C\_MaterialLine;*

*public class MCMaterialLine extends X\_C\_MaterialLine {*

*/\*\* Process Message \*/*

*private static final long serialVersionUID =9879869;*

*public MCMaterialLine(Properties ctx, int C\_MaterialLine\_ID,*

*String trxName) {*

*super(ctx, C\_MaterialLine\_ID, trxName);*

*// TODO Auto-generated constructor stub*

*}*

*public MCMaterialLine(Properties ctx, ResultSet rs, String trxName) {*

*super(ctx, rs, trxName);*

*// TODO Auto-generated constructor stub*

*}*

*public static MCMaterialLine getMaterial(Properties ctx,int C\_MaterialLine\_ID,String trxName){*

*return new MCMaterialLine(ctx,C\_MaterialLine\_ID,trxName);*

*}*

*@Override*

*protected boolean afterDelete(boolean success) {*

*int count = 0;*

*try*

*{*

*String sqlcount = "Select count(\*) from C\_MaterialLine "*

*+" where C\_Material\_ID = ?"*

*;*

*PreparedStatement pstmtcount = DB.prepareStatement(sqlcount,get\_TrxName());*

*pstmtcount.setInt(1, getC\_Material\_ID());*

*ResultSet rs = pstmtcount.executeQuery();*

*while (rs.next()){*

*count = rs.getInt(1);*

*}*

*// update total*

*String sqlupdate = "Update C\_Material "*

*+ "Set Total = ? "*

*+ " WHERE C\_Material\_ID = ?";*

*PreparedStatement pstmtupdate = DB.prepareStatement(sqlupdate, get\_TrxName());*

*pstmtupdate.setInt(1, count);*

*pstmtupdate.setInt(2, getC\_Material\_ID());*

*int nodel = pstmtupdate.executeUpdate();*

*log.config("C\_Material total="+count);*

*pstmtupdate.close();*

*}*

*catch (SQLException e)*

*{*

*log.log(Level.SEVERE, "", e);*

*return false;*

*}*

*return true;*

*}*

*@Override*

*protected boolean beforeDelete() {*

*//updateHeaderAmount();*

*return true;*

*}*

*@Override*

*protected boolean afterSave(boolean newRecord, boolean success) {*

*int count = 0;*

*try*

*{*

*String sqlcount = "Select count(\*) from C\_MaterialLine "*

*+" where C\_Material\_ID = ?"*

*;*

*PreparedStatement pstmtcount = DB.prepareStatement(sqlcount,get\_TrxName());*

*pstmtcount.setInt(1, getC\_Material\_ID());*

*ResultSet rs = pstmtcount.executeQuery();*

*while (rs.next()){*

*count = rs.getInt(1);*

*}*

*// update total*

*String sqlupdate = "Update C\_Material "*

*+ "Set Total = ? "*

*+ " WHERE C\_Material\_ID = ?";*

*PreparedStatement pstmtupdate = DB.prepareStatement(sqlupdate, get\_TrxName());*

*pstmtupdate.setInt(1, count);*

*pstmtupdate.setInt(2, getC\_Material\_ID());*

*int nodel = pstmtupdate.executeUpdate();*

*log.config("C\_Material total="+count);*

*pstmtupdate.close();*

*}*

*catch (SQLException e)*

*{*

*log.log(Level.SEVERE, "", e);*

*return false;*

*}*

*return true;*

*}*

*@Override*

*protected boolean beforeSave(boolean newRecord) {*

*// Check LineNet Amount Should Not Over Invoice Total Amount*

*return true;*

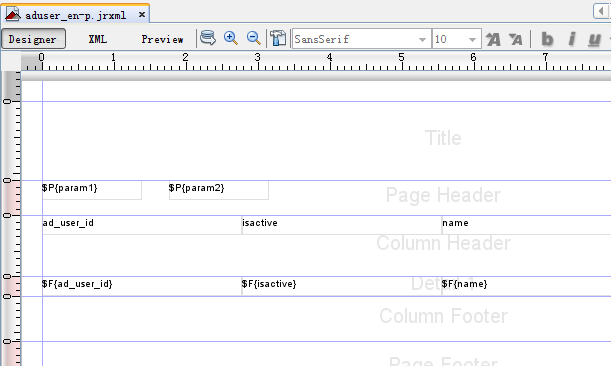
*}*

*}*

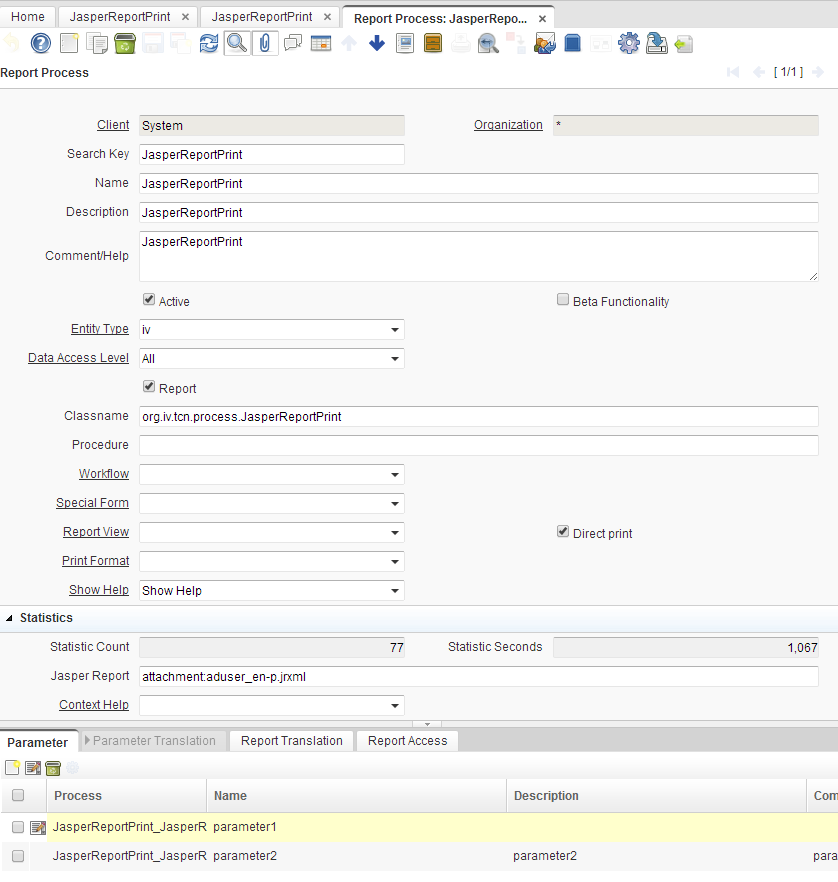
### 第7章 jasper报表

#### STEP1 完成jrxml文件

使用iReport完成jasper的模板



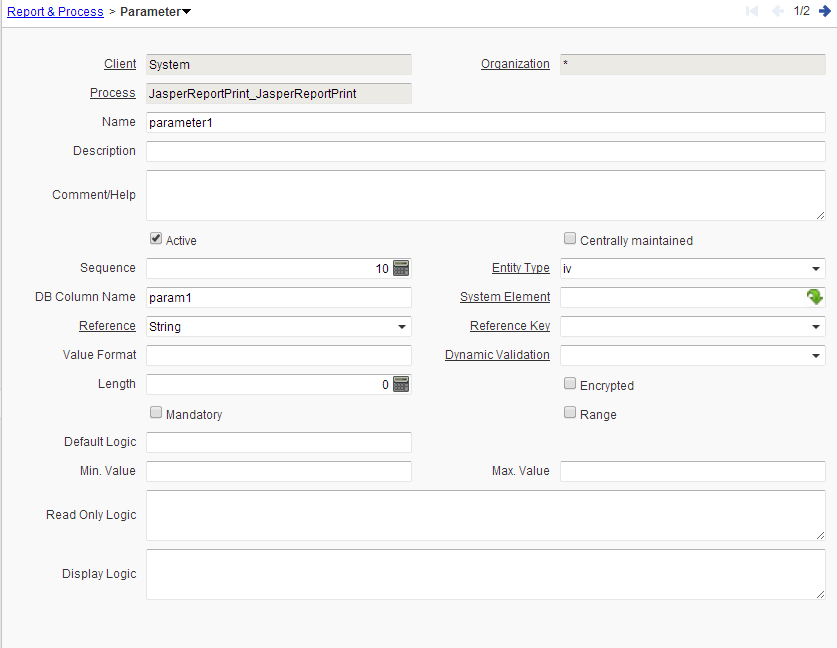
#### STEP2 系统中配置报表信息

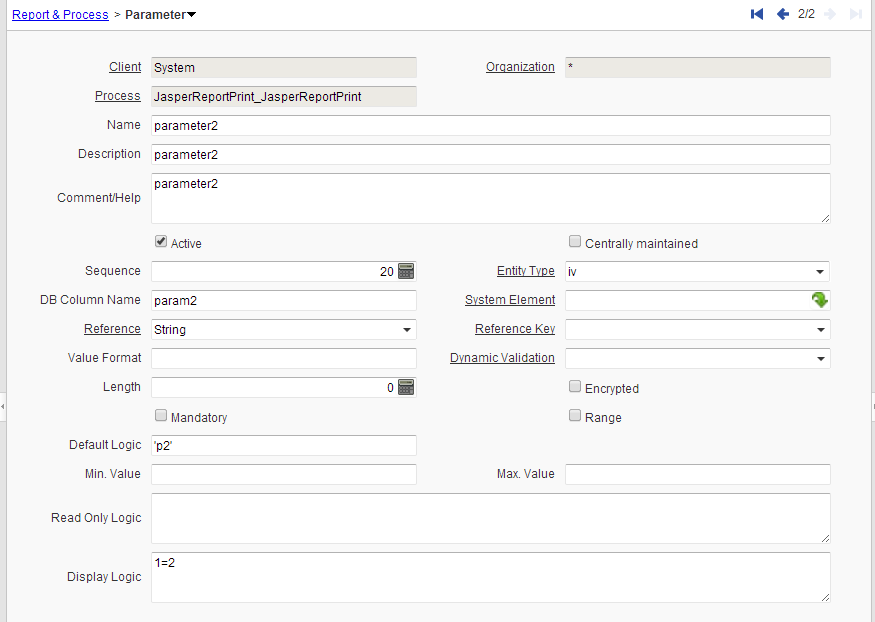
STEP2上传jrxml

在配置画面使用附件功能上传jrxml文件后，指定JasperReport文件的位置。

#### STEP3 设置报表的参数

参照附录Porcess的参数说明



STEP 4 配置按钮和菜单

配置调用该报表的按钮和菜单信息

#### STEP5 创建调用报表的方法

JasperReportPrint.java：

*package org.iv.tcn.process;*

*import java.sql.PreparedStatement;*

*import java.sql.ResultSet;*

*import java.sql.Timestamp;*

*import java.util.logging.Level;*

*import org.compiere.print.MPrintFormat;*

*import org.compiere.print.ServerReportCtl;*

*import org.compiere.process.ProcessInfo;*

*import org.compiere.process.ProcessInfoParameter;*

*import org.compiere.process.SvrProcess;*

*import org.compiere.util.AdempiereUserError;*

*import org.compiere.util.DB;*

*import org.compiere.util.Env;*

*public class JasperReportPrint extends SvrProcess {*

*private String p\_param = "";*

*private int p\_Record\_ID = 0;*

*protected void prepare()*

*{*

*ProcessInfoParameter[] para = getParameter();*

*p\_Record\_ID = getRecord\_ID();*

*for (int i = 0; i < para.length; i++)*

*{*

*String name = para[i].getParameterName();*

*String value = para[i].getParameterAsString();*

*log.log(Level.SEVERE, "Parameter List : " + name + " value :" + value);*

*if (para[i].getParameter() == null)*

*;*

*else if (name.equals("param1"))*

*{*

*p\_param = value;*

*}*

*else if(name.equals("param2"))*

*{*

*if(p\_param.equals("p1"))*

*{*

*para[i].setParameter(p\_Record\_ID);*

*}*

*}*

*else {*

*log.log(Level.SEVERE, "Unknown Parameter: " + name);*

*}*

*}*

*} // prepare*

*@Override*

*protected String doIt() throws Exception {*

*return "";*

*}*

*}*

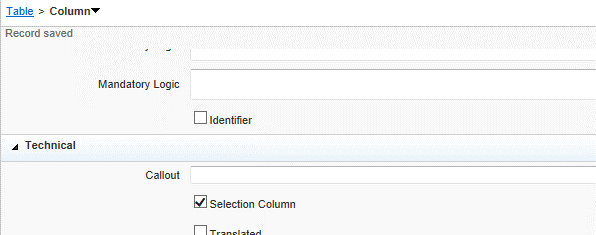
## 附录

#### callout的实现方式：

callout可以通过2种方式实现:

1 单独配置CallFactory的方法(本指南使用，不需要系统配置)

2 使用iDempiere中已实现的CallEngine方法(可参照汇率乘率和除率转换的的配置，需要配置column中的callout)。



#### Model中的触发器

在model中可以配置触发器，用于在数据保存前和保存后进行相应的业务逻辑计算。

触发器有beforeSave,afterSave,beforeDelete,afterDelete。

#### Process中的参数

在配置Report和Process的参数时需要注意一下几点。

1. 在实现业务逻辑的方法中，参数个数不能随意增加，参数个数和名称应该和配置时一致。
2. iDempiere框架在调用Process和Report时，会自动传递当前数据的主键(可以通过getRecordID()取得)。