**帮助文档**

# 页面开发概述

系统的页面开发使用xml配置的方式，页面分为两种，一种是listView ，另一种是billView。分别对应着列表页面和编辑页面。系统中列表页面的统一请求方式为 /dynamicForm/view.jhtml?id=列表页面名称，系统便会根据id参数，找到Admin.WebApp工程下的etc/admin/views/system目录下对应的xml文件。

程序在解析xml文件时，会根据每一个标签找到对应的组件转换器，转换器搭配对应的freemarker组件模板文件，渲染成html字符串。解析过程会根据xml文件dom树的深度来进行递归操作。

# listView（列表页面）开发

## 列表页面开发引入xml约束

所有的列表xml文件头下面增加如下约束，引入该约束之后，按下提示的快捷键，系统便会提供可选的标签以及标签的属性，开发起来相当方便。

|  |
| --- |
| <!DOCTYPE pageview SYSTEM "http://xd.k3cloud.kingdee.com/dtd/JFS.dtd" > |

## 列表页面一般格式

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <!DOCTYPE pageview SYSTEM "http://xd.k3cloud.kingdee.com/dtd/JFS.dtd" >  <pageview type=*"listView"* name=*"页面名称"* plugin=*"指定当前页面对应的插件 "* editview=*"指定当前页面对应的编辑页面"*>  <datasource>  <querylist>  配置当前页面的sql查询语句，注意必须要查出相应的主键并且指定为id  </querylist>  </datasource>  <view>  <toolbar id=*"toolbar"* target=*"datagrid"*>  <menubutton id=*"refresh"* name=*"刷新"* icon=*"refresh"* iservice=*"refresh"*/>  <menubutton id=*"addnew"* name=*"新增"* icon=*"add"* iservice=*"add"*/>  <menubutton id=*"delete"* needSelect=*"true"* name=*"删除"* icon=*"delete"* confirmtip=*"删除后记录不可恢复，确认删除?"*/>  </toolbar>  <listdatagrid id=*"listentry"*>  <filter id=*"filter"*>  <searcher id=*"defaultsearch"* placeholder=*"输入查询关键字"* name=*"查询"* width = *"100"*/>  </filter>  <datagrid id=*"datagrid"* showTree=*"true"*>  <tree id=*"userType"* title=*"用户分类"* rootName=*"全部"*>  <node id=*"b2b"* value=*"A"* name=*"B2B用户"*/>  <node id=*"b2c"* value=*"B"* name=*"B2C用户"*/>  </tree>  <fields>  <text id=*"number"* name=*"编号"*/>  <text id=*"userName"* name=*"用户名"*/>  <text id=*"realName"* name=*"真实姓名"*/>  <text id=*"userType"* name=*"用户类型"*/>  <checkbox id=*"status"* name=*"禁用状态"* defaultvalue=*"1"*/>  </fields>  </datagrid>  </listdatagrid>  </view>  </pageview> |

## 页面标签属性说明

**pageView根节点**：

|  |
| --- |
| 属性说明：  **必选属性（生成标签的同时系统会自动生成的属性）：**  name:导航栏显示的页面名称  type:页面类型，列表页面为listView  plugin:指定控制当前列表页面的插件类，该类需要继承ListViewPlugin  **可选属性：**  extend:指定当前页面继承的父xml，子类的xml将会得到父类的全部信息，如果子类的标签的id跟父类标签的id属性值相同，父类的标签将会被覆盖。  editview：指定当前列表页面对应的编辑页面（点击新增按钮或者是对列表中的某一行双击进行 编辑时跳转过去的页面） |

**datasource：**

|  |
| --- |
| 指定当前列表页面显示数据的sq查询语句l，配置了正确的查询sql之后系统便会调用CoreService 中的findDefault 查询数据并显示。且实现了分页/过滤等，过滤需要手动重写ListViewPlugin 的buildQueryFilter(webCtx, view, param) 方法  该方法将会在后面再作介绍 |

**View**

|  |
| --- |
| 该标签中的内容为展现给用户的所有信息 |

**toolbar**

|  |
| --- |
| 该标签用于定义列表页面中表格对应的工具栏，  属性说明  id:属性值必须唯一  target：指定工具栏所控制的表格（通过属性值与后面的datagrid的id属性值匹配）  子标签说明  menubuttongroup ：工具栏按钮组，可以通过下拉的形式显示多个menubutton  menubutton：工具栏上的某个按钮，如刷新，删除等。  子标签属性说明：  name:按钮对应的名称，会在工具栏上面显示  icon:指定按钮对应的图标，icon属性对应于iconFactory中绑定的图标内容  iservice：绑定调用后台toolbarClick 方法内部指定的事件  confirmtip：执行相关的操作时会弹出一个对话框，只有点击了确认按钮才会真正执行该操作 |

**Listdatagrid**

|  |
| --- |
| 列表根节点，里面可以包含过滤搜索框根节点和列表表格节点 |

**filter**

|  |
| --- |
| 过滤搜索框根节点,  子标签searcher  属性介绍：  id: buildQueryFilter方法中会通过该属性值获取过滤条件  placeholder：占位符，即搜索框中提示用户输入相关的信息进行搜索  name:搜索框的名称，会在页面当中显示  width:搜索框的宽度 |

**datagrid**

|  |
| --- |
| 列表表格  属性介绍：  editable：是否可编辑  showTree：是否显示树形组件 |

**tree**

|  |
| --- |
| 树形组件节点，该节点必须匹配node节点一起使用  属性介绍：  title：树形组件的名称，会在页面当中显示  rootName：树形组件根节点的名称，会显示出来 |

**node**

|  |
| --- |
| 树形组件的子节点  属性介绍：  name：节点名称  value:节点选中的值，该值最终会传到树形导航的过滤条件当中 |

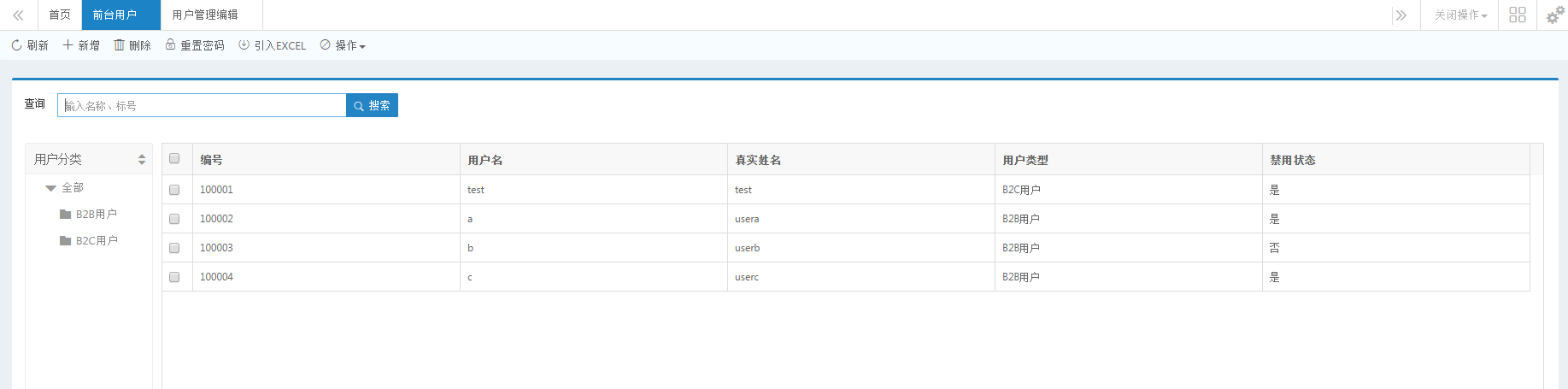
**fields**

|  |
| --- |
| 表格列的根节点  子节点介绍：  text：文本输入框  checkbox: 单选按钮  select；下拉列表 |

## xml页面配置的注意事项

1. 在编写页面对应的sql查询语句的时候，记得要查出相应的主键并且指定为id。
2. 列表表格需要显示的数据，其对应的标签的id属性值必须跟sql查询语句中的数据库字段名称一致。
3. 同一个页面当中的各个标签对应的id属性值必须唯一。

## 页面效果



# ListPlugin（列表页面控制插件）开发

系统中的每一个xml页面都会对应着一个控制插件，其中listView对应的插件为ListPlugin。用户自己编写的ListPlugin必须继承ListViewPlugin。用户对列表页面进行的刷新，删除，查询等操作都可以通过该ListPlugin进行实现。

## ListPlugin介绍

|  |
| --- |
| public class MyListPlugin extends ListViewPlugin {  @Override  public void initView(WebContext webCtx, ListView view) {  super.initView(webCtx, view);  }  /\*\*  \* 工具栏点击  \* @param webCtx  \* @param view  \* @param event  \*/  @Override  public void onToolbarClick(WebContext webCtx, ListView view,  ToolbarClientEvent event) {  // 获取操作事件对应的ID  String clickId = event.getId();  // 根据不同的clickId调用不同的方法  switch (clickId) {  // 判断为刷新操作，执行刷新  case "refresh":  view.refresh(event.getTarget());  break;  // 判断为删除操作  case "delete":  //这里执行删除相关的操作  break;  default:  break;  }  }  /\*\*  \* 组织过滤查询  \* @param webCtx  \* @param view  \* @param event  \* @return  \*/  @Override  public QueryFilter buildQueryFilter(WebContext webCtx, ListView view,IHashMap queryParam) {  QueryFilter queryFilter = new QueryFilter();  String queryStr = queryParam.getString("defaultsearch");  if(queryStr != null && !"".equals(queryStr)){  queryFilter.appendFilter("搜索过滤条件");  }  String treeFilter = queryParam.getString("treeFilter");  if(null != treeFilter && !"".equals(treeFilter)){  queryFilter.appendFilter("树形导航过滤条件");  }    return queryFilter;  }  } |

## buildQueryFilter方法介绍

1. buildQueryFilter用于拼接sql查询时的一些过滤条件
2. 所有过滤条件拼接到变量queryFilter
3. 可以对数据搜索以及左侧导航树的条件进行过滤来获取目标结果
4. 过滤条件可以通过queryParam拿到
5. 搜索条件为searcher标签指定的id，即queryParam.getString(id)
6. 树形导航过滤条件通过queryParam.getString("treeFilter")获取,值为Node的value属性

## onToolbarClick方法介绍

1. 用于操作工具栏按钮点击事件的回调
2. event.getSelections()获取当前表格选中的行
3. 对view的各种操作，会执行OperationProxy.execCommand保存相应的命令
4. view对象的顶级父类ViewProxy中的变量operationProxy会保存相应的操作

# billView（单据/编辑页面）开发

由于编辑页面和列表页面中使用的标签大部分都是相同的，这里主要讲述一下那些在编辑页面用的比较多的标签以及介绍ORM的使用。系统可以通过使用ORM来实现表头以及分录的增删改操作，不过前提是要准确的理解ORM一些配置规则。

## 编辑页面开发引入xml约束

所有的编辑页面的xml文件头下面增加如下约束，引入该约束之后，按下提示的快捷键，系统便会提供可选的标签以及标签的属性，开发起来相当方便。

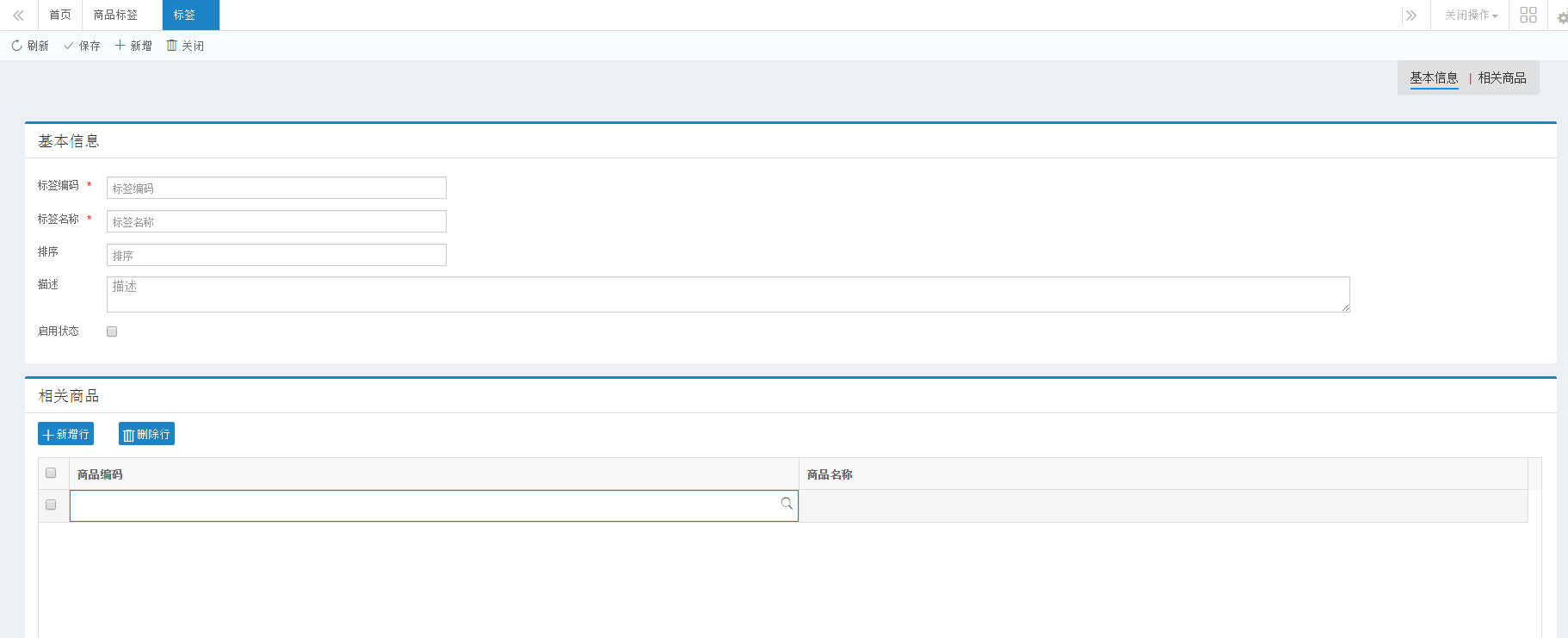
|  |
| --- |
| <!DOCTYPE pageview PUBLIC "-//kingdee.com//DTD JFS 1.0/zh-CN" "../../dtd/JFS.dtd"> |

## 编辑页面的一般格式

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE pageview PUBLIC "-//kingdee.com//DTD JFS 1.0/zh-CN" "../../dtd/JFS.dtd">  <pageview type="billView" name="标签" plugin="kingdee.k3.o2o.admin.webapp.plugin.goods.GoodsLabelEditPlugin" extend="IBillView" bo="label" >  <view>  <toolbar id="toolbar">  <menubutton id="btn\_close" name="关闭" icon="delete" />  <menubutton id="save" name="保存" icon="save" iservice="save"/>  </toolbar>  <tabgroup>  <tab name="基本信息">  <latticerowpanel>  <text id="FLABELNO" name="标签编码" placeholder="标签编码" required="true" orm="number"/>  <blank /><blank/>  <text id="FLABELNAME" name="标签名称" placeholder="标签名称" required="true" orm="name"/>  <blank /><blank/>  <text id="FORDER" name="排序" placeholder="排序" orm="ordernum"/>  <blank /><blank/>  <textarea id="FDESC" name="描述" placeholder="描述" cols="3" rows="2" orm="description" />  <checkbox id="FENABLE" name="启用状态" orm="enable"/>  </latticerowpanel>  </tab>  <tab name="相关商品">  <datagrid id="datagrid" editable="true" orm="goodsDetail">  <toolbar id="toolbarIn" >  <menubutton id="entry1" name="新增行" icon="add" class="blue-bg" iservice="addEntry"/>  <menubutton id="entry2" name="删除行" icon="delete" class="blue-bg" iservice="delEntry"/>  </toolbar>  <fields>  <selectdata id="FGOODS" name="商品编码" col="1" superselect="GoodsNumber" editable="true" linkdatamap="FNAME" orm="goodsId" linkorm="goodsId.number"/>  <text id="FNAME" name="商品名称" linkorm="goodsId.goodsName"/>  </fields>  </datagrid>  </tab>  </tabgroup>  </view>  </pageview> |

注:上面中的绿色部分为ORM的相关配置，黄色部分为F8弹出框的相关配置，后面会逐一加以说明。

## 页面效果



## 页面标签属性说明

tabgroup

|  |
| --- |
| 单据根节点，下面可以包含多个单据标签页tab |

tab

|  |
| --- |
| 单据标签页，当页面有多个tab的时候，可以通过点击页面右上角的导航标签进行导航。用法：可以通过一个tab编辑表头的信息，另外再用一个或者多个tab来编辑分录的信息，其中表头对应于数据库当中的主表，分录对应于数据库当中的从表。 |

blank

|  |
| --- |
| 换行的符号，两个blank组合在一起达到换行的效果 |

## ORM使用说明

开发者可以使用系统中的ORM来进行增删改操作，不过前提是先要在元数据管理中正确的配置业务对象以及在页面编辑的xml文件中正确使用ORM的相关标签。其中xml文件中的相关配置就如同4.2的编辑页面当中绿色部分所示。下面详细描述ORM使用上的一些相关的配置。

### 元数据管理中的业务对象编辑

进入后台主页，把鼠标光标移动到左侧导航栏的测试模块中，然后在元数据管理菜单组中点击业务对象，进入到业务对象列表页面（如下图所示）。系统中原来已经建好的业务对象都会在这里显示。我们可以通过搜索框进行搜索或者点击页面右下方的翻页按钮进行查找。这里主要讲述一下按照ORM的规则，我们应该怎样建立一个元数据。

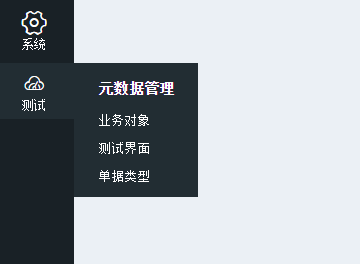


图 4‑1进入元数据管理菜单组

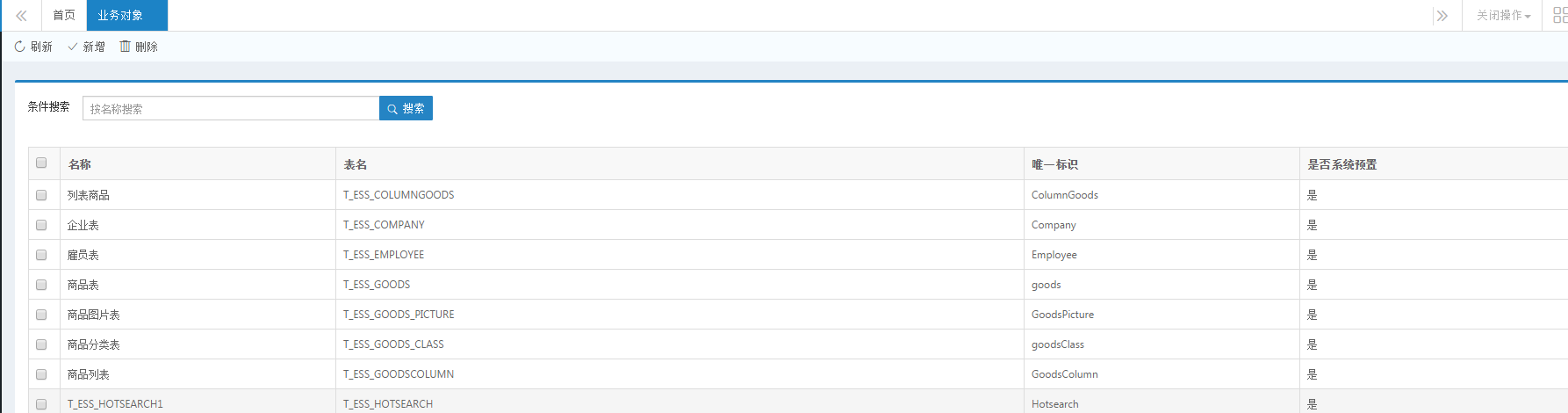


图 4‑2 业务对象列表页面

点击业务对象列表页面中的新增按钮（位于左上角），进入到业务对象的编辑页面。编辑页面的效果如下图所示。

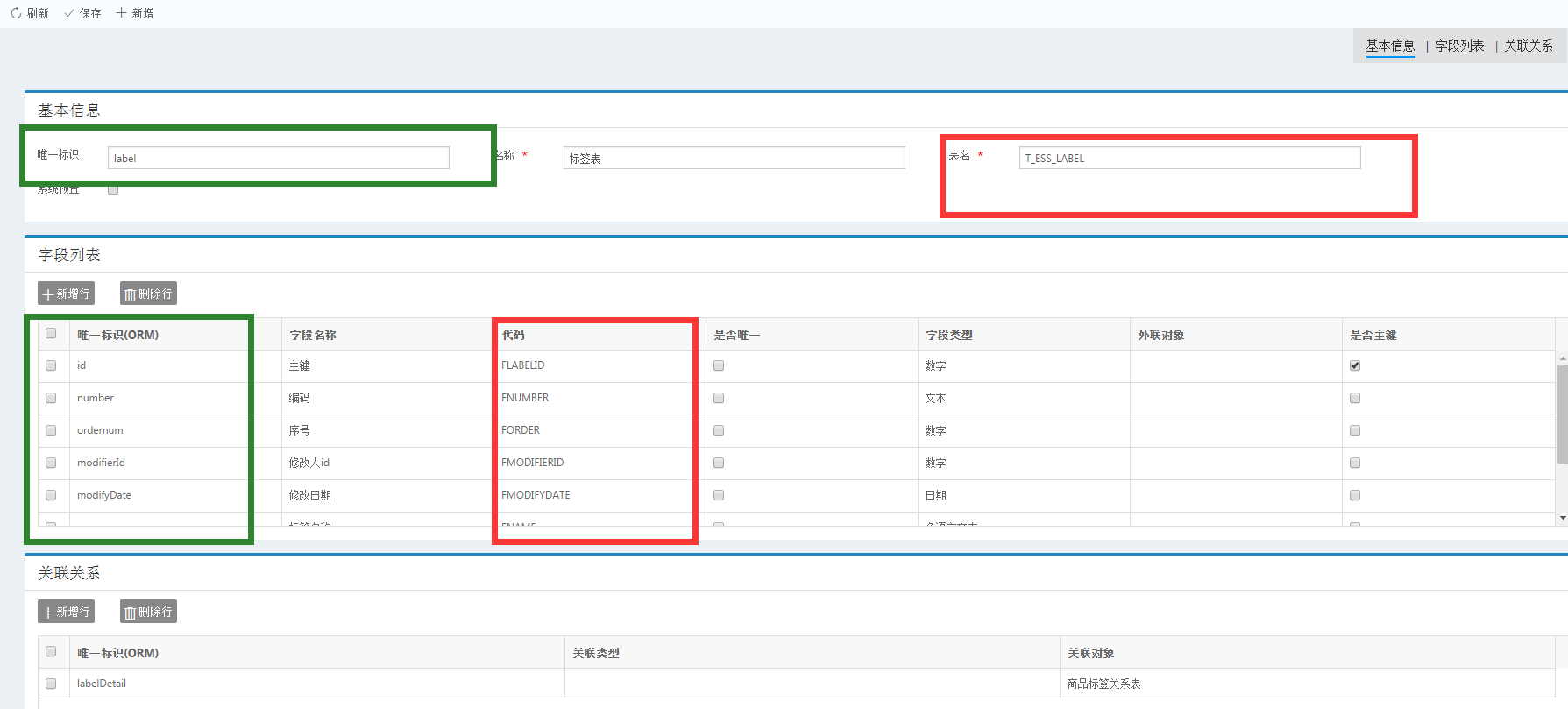


图 4‑3 业务对象编辑页面

该业务对象（标签label）所对应的数据库当中的表T\_ESS\_LABEL

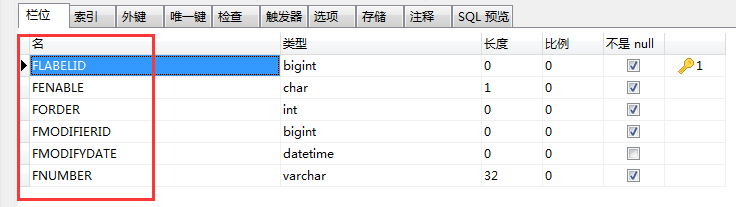


图 4‑4 数据库中的表T\_ESS\_LABE结构

下面对照着4.2当中的编辑页面，结合上面的业务对象编辑页面还有就是数据库当中的表结构详细描述出ORM的配置规则。

**业务对象编辑页面的基本信息：**

**唯一标识**：填入一个与数据库表名对应的标识，要求与编辑页面中的pageviw标签的bo属性值一致。对照着页面中的绿色部分和图 4 3 业务对象编辑页面中的绿色部分来看就很清楚了。

**名称**：这里根据实际情况取一个名字即可。

**表名**：这里填入的表名必须跟数据库中的表名一致。

**业务对象编辑页面的字段列表：**

**唯一标识（ORM）：**这里其实对应的是数据库表中的字段，要求跟编辑页面对应的标签中的orm属性值一致。注意：如果当前对应的字段是主键，那么唯一标识只能够定义为id。

**字段名称：**这里根据实际情况填写即可。

**代码**：这里跟数据库中对应的表的字段保持一致。

**字段类型**：如果对应的字段属于多语言表中的字段，这里应该选择“多语言文本”，（使用orm的时候，多语言表没有对应的业务对象，其对应的字段都加在主表当中，通过选择“多语言文本”进行关联）。如果对应的属于外键，这里就应该选择“外联数据”。

**外联对象**：如果在前面的字段类型当中选择了“外联数据”，这里应该选择其关联的业务对象。在弹出的F8控件中选中外联对象后，点击“返回数据”按钮即可。

**是否主键**：这里根据实际情况进行勾选。

**业务对象编辑页面的关联关系：**

这一栏主要用于主表关联从表（表头关联分录）

**唯一标识（ORM）**:根据实际填写关联的业务对象的唯一标识。

**关联对象：**在弹出的F8控件中选中关联的业务对象后，点击返回数据按钮即可形成关联。

### 编辑页面中的ORM相关配置

如4.2中的编辑页面中的绿色部分所示，首先在pageview标签当中引入一个bo属性，其属性值跟相关业务对象的唯一标识一致。另外就是在那些跟业务对象的每个字段相对应的标签当中引入一个orm属性，其属性值跟业务对象中相应的字段的唯一标识保持一致。

### F8控件的配置

所谓的F8控件其实就是在填写一些页面信息的时候，不需要手工输入，只需要通过点击输入框，然后系统会自动弹出一个选择框，用户只需要在选择框当中选中对应的行然后点击“返回数据”按钮，就可以完成数据的输入，主要应用于一些系统已有数据的填写。具体形式如下图所示。



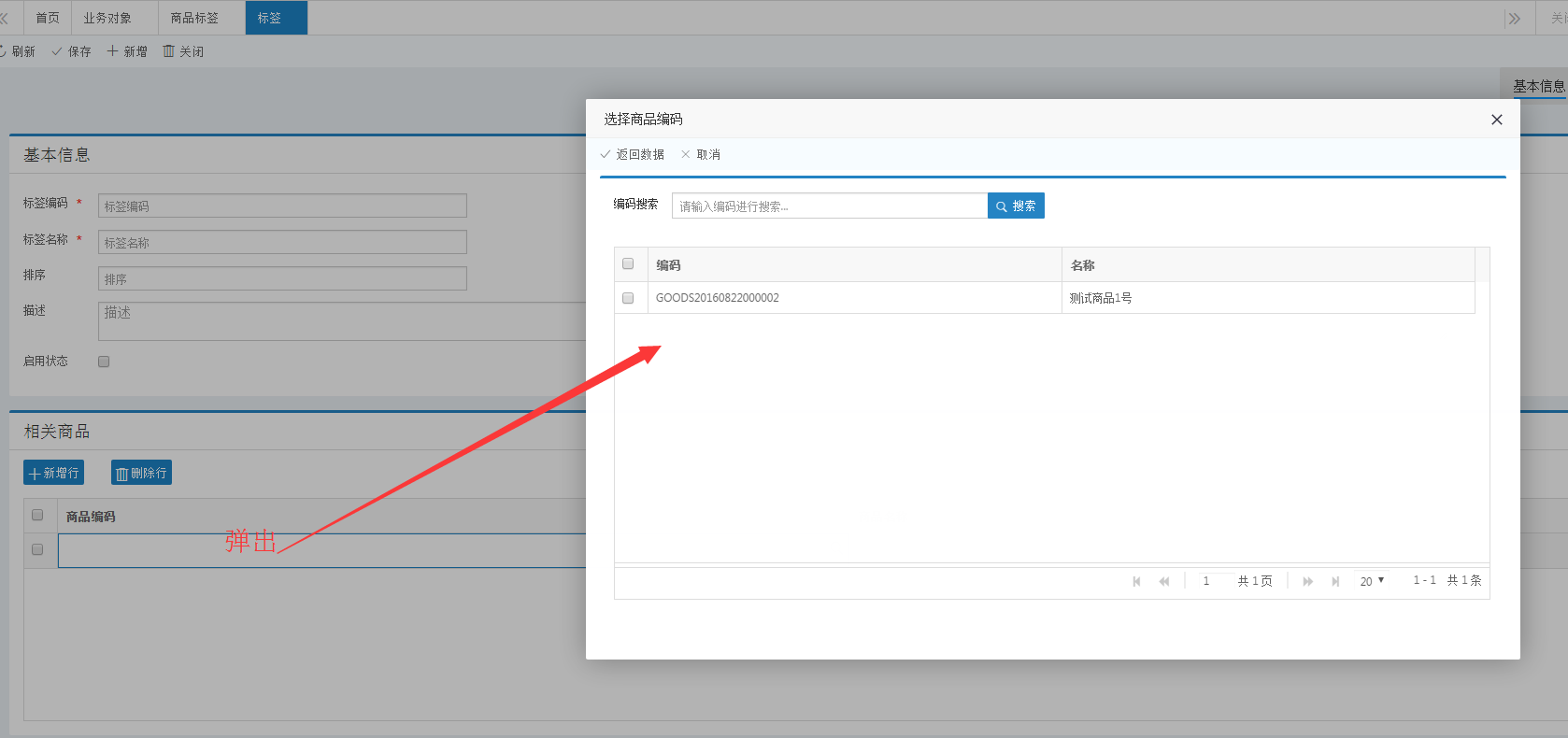


图 4‑5 F8控件

要应用F8控件，主要有两个步骤。首先要在系统中的SuperSelectDefine.xml文件中定义一个SuperSelect，然后在编辑页面的xml文件中引入该SuperSelect。

* SuperSelectDefine.xml文件中定义一个SuperSelect

|  |
| --- |
| <SuperSelect id=*"GoodsNumber"* keyfield=*"id"* valuefield=*"FNUMBER"* multiselect = *"true"*>  <Sql select=*" "* from=*" "* where=*" "* />  <Filters>  <Filter id=*"num\_search"* name=*"编码搜索"* type=*"searcher"* placeholder=*"请输入编码进行搜索..."* searchfield=*"gs.FNUMBER"*/>  </Filters>  <Fields>  <Field id=*"FNUMBER"* name=*"编码"* type=*"text"* />  <Field id=*"FNAME"* name=*"名称"* type=*"text"* />  </Fields >  </SuperSelect> |

标签属性配置说明：

superSelect 标签中id的属性值用来标识当前的控件（必须唯一），提供给外部引用

keyfield：调用控件之后返回的数据（这里一般都是主键）

valuefield: 调用控件之后返回的数据(这里一般是编辑页面需要显示的数据)

multiselect：是否多选,值为true或者false

sql:根据实际需求编写查询sql

Filter:弹出框中的过滤查询，通过searchfield属性指定过滤查询的字段

Fields：弹出框当中列表的根节点

Field：指定列表中的一列

* 编辑页面的xml文件中引入该SuperSelect

|  |
| --- |
| <fields>  <selectdata id="FGOODS" name="商品编码" col="1" superselect="GoodsNumber" editable="true" linkdatamap="FNAME" orm="goodsId" linkorm="goodsId.number"/>  <text id="FNAME" name="商品名称" linkorm="goodsId.goodsName"/>  </fields> |

标签属性配置说明：

selectdata： 里面可以引入superselect

id：用来绑定数据

superselect:通过属性值来指定引用SuperSelectDefine.xml文件中定义的某一个控件

editable:是否可编辑，true/false

linkdatamap:带数据出来供下面的text标签中使用

orm:关联中间表的数据

linkorm:通过中间表再关联另外一张表的字段

# EditPlugin（编辑页面控制插件）开发

系统中的每一个编辑页面的xml文件都会对应着一个控制插件，其中billView对应的插件为EditPlugin。用户自己编写的EditPlugin需要继承BillViewPlugin。例如用户对页面进行的一些保存更新等操作都可以通过该EditPlugin进行实现。不过由于系统采用了ORM来实现增删改操作，EditPlugin插件的开发任务就变得简单了很多。

# 把开发好的页面挂到所属菜单组下面

系统列表页面的统一请求方式为 /dynamicForm/view.jhtml?id=列表页面名称。下面主要讲述一下如何把开发好的列表页面文件挂到其所属的菜单组下面。

（1）打开数据库当中的T\_ESS\_MENU\_GROUP\_L表，在FNAME列中找到相应的菜单组的名称，记住相应的FID的值。

（2）在表T\_ESS\_MENU中新建一条记录，在FGROUPID中填入步骤（1）中记住的FID,在FURL列中填入页面的请求方式：/dynamicForm/view.jhtml?id=xxxListView。然后在菜单多语言表T\_ESS\_MENU\_L中新建一条对应的记录，在FNAME列中填入页面的中文名称。