苍穹开发必备 100 个小知识

本文的苍穹开发小知识结合了常见的需求小场景进行讲解，通过一问一答的形式，可以帮助大家快速找到对应场景的实现方式，所涉及的知识是苍穹开发很常用的，值得大家认真阅读及掌握。文档内容是在苍穹开发初级学习路径的课程内容及课后练习需求上总结及扩展的，适合已学完苍穹开发初级学习路径的用户。此文档还会持续根据社区及项目常见问题进行迭代更新。

版本	编写部门	编写人	时间
V1.0	苍穹开发者赋能部	黄春媛、陈来珍	2022.09

使用方法：在 word 视图下开启导航窗格，然后阅读全文，需要时再自行查找相关内容。

目录

一、页面 4

1. 表单页面取值 4

(1) 获取简单字段的值 4

(2) 获取基础资料的属性值 5

(3) 获取单据体某个字段或行对象 6

(4) 获取子单据体某个字段或行 7

(5) 获取富文本控件的值 7

(6) 获取多选基础资料的值 8

2. 表单页面赋值 8

(1) 给简单字段赋值 8

(2) 给基础资料字段赋值 9

(3) 给单据体的字段赋值 9

(4) 给单据体新建行 10

(5) 给子单据体新建一行 10

(6) 给富文本控件赋值 11

(7) 给多选基础资料赋值 11

3. 列表页面赋值 12

(1) 单据列表 12

4. 数据复制 12

(1) 如何复制单据数据 12

5. 页面参数 14

(1) 如何判断界面是否新增界面？ 14
6. 视图模型 VIEW 14
(1) 取消打开页面 14
(2) 设置控件的锁定性 15
第一种：锁定单据头字段 15
第二种：锁定单据体或单据体字段 15
(3) 设置控件可见性 16
(4) 设置字段必录性 16
(5) 设置单据体选中行 16
(6) 执行页面的按钮点击事件 17
(7) 执行操作 17
(8) 刷新数据 17
(9) 页面关闭 18
(10) 页面弹出提示信息 19
(11) 修改控件名称 19
(12) 页面缓存 20
(13) 获取当前页面的元数据信息 20
二、前后端交互 21
1. 用户输入 21
(1) 数据联动 21
(2) 通过代码赋值后不想触发值更新事件 21
2. 用户点击 22
(1) 文本字段/按钮点击 23
(2) 工具栏按钮点击 24
(3) 基础资料点击 25
(4) 单据体点击 25
①单据体单元格点击 25
②单据体行点击 26
③用代码选中单据体行 27
④ 获取单据体当前选中行 27
(5) 单据列表超链接点击 27
(6) 列表获取当前选中行 28
3. 父子页面交互 28
(1) 代码弹出表单页面 28
(2) 如何修改父页面的数据 29
4. 基础资料选择界面（F7） 31

(1) F7 选择界面显示哪些字段 31
(2) F7 选择界面显示未审核的数据 31
(3) F7 选择界面展示已选择的数据，开启多选等 32
三、后端数据操作 32
1. 如何新建数据对象并赋值 32
(1) 新建空数据对象 32
(2) 给对象赋简单字段的值 33
(3) 给对象新增单据体并赋值 33
(4) 给对象新增子单据体并赋值 33
2. 如何保存数据对象 34
(1) SaveServiceHelper 34
(2) OperationServiceHelper 34
3. 如何查询数据对象 35
(1) 查询简单字段&使用简单字段做过滤条件 35
(2) 查询复杂字段&使用复杂字段做过滤条件 35
(3) 查询单据体字段&使用单据体字段做过滤条件 36
①使用 BusinessDataServiceHelper 36
②使用 BusinessDataServiceHelper 36
(4) 查询子单据体字段&使用单据体字段做过滤条件 37
4. 如何删除数据对象 38
（1）直接从数据库删除数据，不触发数据校验 38
（2）删除数据时，触发数据校验 38
四、操作 38
1. 代码触发操作 38
(1) 界面插件触发操作 38
(2) 非界面插件触发操作 39
(3) 代码触发操作如何忽略验权 39
2. 操作后如何刷新字段 39
(1) 配置实现 39
(2) 代码实现 40
3. 操作后如何提示 41
(1) 配置提示 41
(2) 代码修改操作后的提示 41
4. 操作插件跟界面插件如何传递参数 42
(1) 从界面传递参数给操作插件 42
(2) 从操作插件传递参数给界面 43

一、页面

1. 表单页面取值

(1) 获取简单字段的值

问题 1：获取办公用品登记单的备注字段（文本字段）的值？

参考答案：

数据模型 IDataModel 中存储了当前页面的数据包，对于文本类型、整数、时间字段等简单类型字段，可以直接通过 IDataModel 的 get 方法直接获取字段值，值的返回类型有 String、Date、BigDecimal 等。

```
IDataModel model = this.getModel();
String remark = (String)model .getValue("文本字段标识");
```

问题 2：获取单据体数量字段的值？

参考答案：

取单据体中的字段值需要指定行数，下标从 0 开始，如获取第一行数量字段的值：

```
BigDecimal qty=this.getModel().getValue("数量字段标识",0)
```

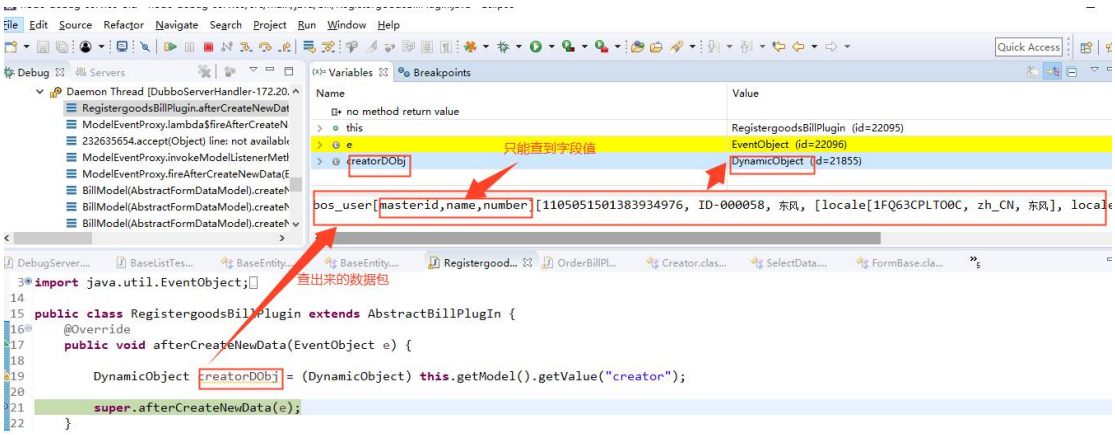
(2) 获取基础资料的属性值

问题：如何获取创建人的生日？

参考答案：

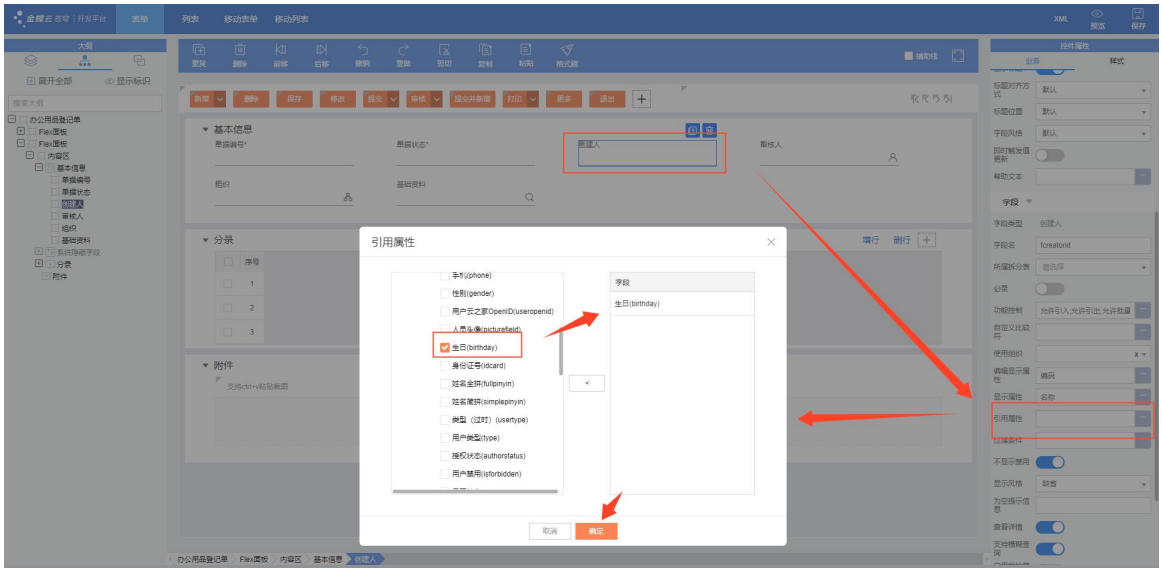
创建人是基础资料字段，像创建人、修改人、用户字段本质都是基础资料字段，只是关联的基础资料类型已预置人员【bos_user】。对于基础资料， 通过 this.getModel().getValue("基础资料字段标识") 拿到是一个对象 DynamicObject ，并且默认取出来的 DynamicObject 数据中只包含基础资料的 id、name、number 属性值。

DynamicObject creatorDObj = (DynamicObject) this.getModel().getValue("creator");



所以直接 this.getModel().getValue（ ）默认无法获取到基础资料除 id,name、number 外的属性值，需要通过以下两种方式获取。

方式 1：在表单设计器中引用人员的“生日”属性，然后再通过以下代码可以取到基础资料某个属性。



DynamicObject creatorDObj = (DynamicObject) this.getModel().getValue("creator");

String birthday = creatorDObj.getString("birthday");

方式 2：先拿到基础资料的 id，再通过 orm 接口去查基础资料的某个属性值。

```
//获取当前页面创建人字段的数据对象
DynamicObject creatorDObj = (DynamicObject) this.getModel().getValue("creator");
Object pkValue = creatorDObj.getPkValue();//拿主键即 id
QFilter idQFilter = new QFilter("id", QCP.equals, pkValue);//构造过滤条件
DynamicObject queryRuselt =
QueryServiceHelper.queryOne("bos_user", "birthday",new QFilter[] {idQFilter} );//查询数据库
String birthday = queryRuselt.getString("birthday");//拿属性的值
```

(3) 获取单据体某个字段或行对象

问题 1：如何获取单据体某行某个字段的值？

答：**this.getModel().getValue("kded_textfield", 0);**//0 代表第一行

问题 2：如何获取单据体中各行的数据？

参考答案：

单据体中的数据是多行数据的集合，每行数据都是一个 DynamicObject 数据包，每行数据又包含多个字段（简单值、复杂值）值。

方式 1：

```
//获取当前页面的数据包
DynamicObject dataEntity = this.getModel().getDataEntity(true);
//获取单据体数据的集合
DynamicObjectCollection goodsEntities=dataEntity.getDynamicObjectCollection("单据体标识");
for (DynamicObject entryObj : goodsEntities) {
    //获取某行数据的 id
    Object pk=entryObj.getPkValue();
    //获取某行的某个字段的值
    Object object = entryObj.get("字段标识");
}
```

方式 2：

```
DynamicObjectCollection entryRows = this.getModel().getEntryEntity("单据体标识");
```

(4) 获取子单据体某个字段或行

问题 1：如何获取子单据体某行某个字段的值？

参考答案：

```
this.getModel().getValue("kded_textfield", 0,0);
//第一个整数代表子单据体行号，第二个整数代表父单据体行号
```

问题 2：如何获取子单据体中各行的数据？

参考答案：

子单据体中的数据也是是多行数据的集合，并且是挂在单据体行里的，所以需要先得到单据体的行数据对象，再从单据体行数据对象获取子单据数据行集合。

```
//获取当前页面的数据包
DynamicObject dataEntity = this.getModel().getDataEntity(true);
//先获取单据体数据的集合
DynamicObjectCollection goodsEntities=dataEntity.getDynamicObjectCollection("单据体标识");
for (DynamicObject entryObj : goodsEntities) {
    //再获取子单据体行某行数据的 id
    DynamicObjectCollection cols=entryObj .getDynamicObjectCollection("子单据体标识");
    for (DynamicObject col: cols) {
        col.get("字段标识");//获取子单据体字段的值
    }
}
```

(5) 获取富文本控件的值

问题：如何获取富文本控件的值？

参考答案：富文本是通用控件，通用控件不能存储数据，所以不能通过 model 来取值，富文本的控件模型提供了相应的接口。

```
RichTextEditor edit = this.getView().getControl("富文本控件标识");
String text = edit.getText();
```

因此需要保存时把富文本的数据保存到一个大文本字段上，打开保存后的单据时，将该大文本的值绑定到富文本控件上。

```
//在保存单据时，将富文本控件的内容保存到大文本字段
public void beforeDoOperation(BeforeDoOperationEventArgs e) {
```

```
        FormOperate operate =   (FormOperate)e.getSource();
        String key = operate.getOperateKey();
         if(key.equalsIgnoreCase("save")) {
              RichTextEditor edit = this.getView().getControl("富文本控件标识");
            String text = edit.getText();
            this.getModel().setValue("大文本字段标识",text );//赋值到大文本字段上
         }
    //在打开单据时，将大文本字段的值显示到富文本控件
    public void afterBindData(EventObject e) {
        super.afterBindData(e);
        RichTextEditor edit = this.getView().getControl("富文本控件标识");
        String text = edit.setText((String) this.getModel().getValue("大文本字段标识"));
    }
```

注意事项：能够通过 model 去获取值的字段，必须在表有对应的表字段，比如富文本、基础资料属性这类字段，是没有表字段的，自然不能通过 model 去取值及赋值。

(6) 获取多选基础资料的值

多选基础资料字段配置的时候是要写表名的 ,通过 model 拿到的就是该表某些数据的集合，遍历集合才能拿到实际选择的基础资料数据对象。

```
    DynamicObjectCollection multiCols = (DynamicObjectCollection)
    this.getModel().getValue("kded_mulbasedatafield");
        for(DynamicObject col:multiCols){
            //获取选择的基础资料数据对象
            DynamicObject baseData = (DynamicObject) col.get("fbasedataid");
        }
```

2. 表单页面赋值

(1) 给简单字段赋值

问题：新增单据时，如何自动给办公用品登记单备注字段（文本）设置默认值？

参考答案：简单字段的赋值，直接通过当前页面的数据模型 IDataModel 赋值给字段赋值。

```
@Override
public void afterCreateNewData(EventObject e) {
    IDataModel model = this.getModel();
    model.setValue("kded_remark", "我的备注数据");
}
```

(2) 给基础资料字段赋值

问题：如何给办公用品登记单的登记人字段赋值？

参考答案：因为登记人的是“用户”类型字段，而用户字段又是特殊的基础资料字段。所以给基础资料赋值必须用基础资料的 id 或者基础资料的 DynamicObject。

```
@Override
public void afterCreateNewData(EventObject e) {
    long currentUserId = UserServiceHelper.getCurrentUserId();
    //方式 1：通过 id 赋值-推荐
    this.getModel().setValue("kded_registrant", currentUserId );
    //方式 2：通过对象赋值
    DynamicObject user = BusinessDataServiceHelper.loadSingle(currentUserId,
    "bos_user");
    this.getModel().setValue("kded_registrant", user);
}
```

注意：不能使用 QueryServiceHelper.queryOne 查到的平铺对象赋值给基础资料字段。

(3) 给单据体的字段赋值

问题：如何给单据体某个字段赋值？

参考答案：

方式 1：如果是知道行号 index，且只需要给这行的字段赋值，则参考：

```
this.getModel().setValue("字段标识","字段值",index);
```

方式 2：需要给每行赋值，则需遍历单据体进行赋值：

```
DynamicObjectCollection entryRows = this.getModel().getEntryEntity("单据体标识");
for (DynamicObject entryObj : entryRows) {
    //修改某行的某个字段的值
    entryObj.set("kded_textfield", "测试");}
     this.getView().updateView("entryentity");
```

```
}
```

注意点：方式 2 如果在 afterCreateNewData 事件则不需要 updateView,否则需要代码刷新前台才会显示。

(4) 给单据体新建行

问题：如何给办公用品登记单的单据体添加一行数据？

方式 1：单据体虽然是集合，但是数据模型 IModel 提供了直接增加行的接口，需要知道要增加的行数 row。

```
int[] newEntryRow = this.getModel().batchCreateNewEntryRow("单据体标识", row);
for (int index : newEntryRow) {
    this.getModel().setValue("单据体某个字段标识", 值, index);
}
```

方式 2：给单据体创建行对象，再调用 IModel 增加行的接口。

```
DynamicObject entry =
ORM.create().newDynamicObject(this.getModel().getEntryEntity("单据体标识
").getDynamicObjectType());
entry.set("kded_textfield1","44");
entry.set("kded_amountfield","10");
int rowindex = this.getModel().createNewEntryRow("单据体标识", entry);
```

注意：如果单据体字段有设置默认值，此方式默认值会覆盖代码设置的值。

方式 3：通过数据包给单据体直接新增行对象。

```
DynamicObjectCollection entryentityCols =
this.getModel().getDataEntity(true).getDynamicObjectCollection("entryentity");
    DynamicObject entryCol = entryentityCols.addNew();
    entryCol.set("kded_textfield", "方式 4");
    this.getView().updateView("entryentity");
```

获取单据体数据包，在数据包新增行对象，是需要代码调用 updateView 才会将新内容显示到界面上。

(5) 给子单据体新建一行

问题：如何给子单据体添加一行数据？

参考答案：

```
//先设置父单据体选中行
this.getModel().setEntryCurrentRowIndex("entryentity",0);
//创建子单据体行-方式 1
int[] ints = this.getModel().batchCreateNewEntryRow("kded_subentryentity", 1);
for (int index : ints) {
    this.getModel().setValue("kded_textfield2", "文本值", index);
}

//创建子单据体行-方式 2
DynamicObject subEntry =
ORM.create().newDynamicObject(this.getModel().getEntryEntity("kded_subentryentit
y").getDynamicObjectType());
subEntry.set("kded_textfield2","行对象");
int rowindex = this.getModel().createNewEntryRow("kded_subentryentity",
subEntry);
```

(6) 给富文本控件赋值

问题：如何给富文本控件赋值？

参考答案：富文本是通用控件，不能像实体字段一样直接用数据模型 this.getmodel 赋值，需要通过富文本控件模型 RichTextEditor 提供的方法 setText 设置值。

```
RichTextEditor edit = this.getView().getControl("richtexteditorap");
edit.setText("要设置的内容");
```

(7) 给多选基础资料赋值

参考答案：

【https://vip.kingdee.com/article/198799132897594368?productLineId=29】

3. 列表页面赋值

(1) 单据列表

问题：如何在列表中动态修改或更新当前流程处理人字段？
参考答案：列表是用来显示单据存储在数据库中的数据的，如果要在列表显示的时候，修改列表数据，可以在列表插件中重写 beforeCreateListDataProvider 方法，构建自定义的列表取数器，实现自主取数，具体参考：
【https://vip.kingdee.com/article/306390194805169920?productLineId=29】

4. 数据复制

(1) 如何复制单据数据

问题：办公物品登记单添加添加一个备份功能，保存时，如果数据库中已存在该表单数据，则生成该条单据的副本数据，要求业务数据和原来保持一致(id 这种唯一性数据不做要求)，并把备注字段赋值："备份数据"。
参考答案：首先，要找到在哪个事件中去备份数据。要求保存时，要先查询数据库中是否有数据，那就是该次保存还未正式保存到数据库中的时候去处理。即在操作插件 beginOperationTransaction 事件中同步备份数据。

```
public class SrSubmitOpPlugin extends AbstractOperationServicePlugIn {
    @Override
    public void onPreparePropertys(PreparePropertysEventArgs e) {
        super.onPreparePropertys(e);
        e.getFieldKeys().add("kded_srstatus");
    }
    @Override
    public void beginOperationTransaction(BeginOperationTransactionArgs e) {
        super.beginOperationTransaction(e);
        DynamicObject[] dataEntities = e.getDataEntities();
        for(DynamicObject data : dataEntities) {
            //单据标识
            String entityNumber = data.getDataEntityType().getName();
```

```
                //data 是保存前且校验通过的数据包，此处备份数据
                Object pkValue = obj.getPkValue()
                boolean exists = QueryServiceHelper.exists(entityNumber , pkValue);
                if(exists){
                    //如下复制数据方案 1 或者方案 2
                )else{
                }
            }
        }
    }
```

数据复制有两种方案：

方案 1：创建空数据包，并逐一赋值-不推荐，后续表单增加字段，需要不断迭代代码。

```
    //创建空的数据包
    DynamicObject newDynamicObject =
    BusinessDataServiceHelper.newDynamicObject(entityNumber);
    newDynamicObject.set("单据头字段标识 a", data.get("单据头字段标识 a"));
    newDynamicObject.set("备注字段标识", "备份数据");
    //单据体数据行
    DynamicObjectCollection newCol =
    newDynamicObject.getDynamicObjectCollection("单据体标识");
    DynamicObjectCollection oldCol = data.getDynamicObjectCollection("单据体标    识
    ");
    //复制当前分录数据到新增数据包的新增分录行上
    for (DynamicObject oldEntry : oldCol) {
        //创建一个空的分录行
        DynamicObject newEntry = new
        DynamicObject(newCol.getDynamicObjectType());
        newEntry.set("单据体字段标识 x", oldEntry.get("单据体字段标识 x"));
        //新增数据行添加到单据体上
        newCol.add(newEntry);
    }
    //保存数据
    SaveServiceHelper.saveOperate(entityNumber, new DynamicObject[]
    {newDynamicObject})
```

方案 2：直接克隆整个数据包-增加字段无需迭代代码，推荐。

```
    DynamicObject newData = (DynamicObject) new   CloneUtils(false,true).clone(data );
```

```
newData.set("备注字段标识", "备份数据");
//保存数据
SaveServiceHelper.saveOperate(entityNumber, new DynamicObject[] {newData })
```

5. 页面参数

(1) 如何判断界面是否新增界面？

问题：办公用品登记单打开时，如果是新增页面，则弹出提示“您正在新增办公用品登记单”；如果是编辑页面，则提示”您正在编辑一个办公用品登记单“。

参考答案：

①首先确认要重写的事件，要在“新增”和“编辑”状态下都会执行。而弹出提示是 view 的行为，所以一般建议在 afterBindData 事件中处理。

②每个界面视图中，都保存了相关界面的参数，包括但不局限于：页面 pageid，表单标识、表单运行插件、父页面传过来的自定义参数等，例如当前页面是新增状态还是编辑状态，可以根据单据页面的界面参数 FormShowParameter 中的 getStatus 方法获取到。

```
BillShowParameter bsp=(BillShowParameter)this.getView().getFormShowParameter();
if(bsp.getStatus()==OperationStatus.ADDNEW ){
    this.getView().showTipNotification("您正在新增办公用品登记单");
}
if(bsp.getStatus()==OperationStatus.EDIT){
    this.getView().showTipNotification("您正在编辑一个办公用品登记单");
}
```

6. 视图模型 VIEW

(1) 取消打开页面

问题：如何控制办公用品登记单在周日的的时候不能新增单据？

参考答案：在表单显示前判断 new Date()判断一下如果是周日，然后在 preOpenForm 事件中取消页面的打开。

```
@Override
public void preOpenForm(PreOpenFormEventArgs e) {
    e.setCancel(true);
```

```
        e.setCancelMessage("对不起，周日不能制单！");
    }
```

(2) 设置控件的锁定性

第一种：锁定单据头字段

问题 1：办公用品登记单的总金额自动计算得出的，所以始终是锁定状态。

参考答案：

方式 1：表单设计器上设置”新增锁定、修改锁定、提交锁定、审核锁定"。

方式 2：通过如下代码设置锁定性。

```
    @Override
    public void afterBindData(EventObject e) {
        super.afterBindData(e);
        this.getView().setEnable(false, "字段标识");
    }
```

说明:其他场景下，影响锁定性的因素还有单据类型、界面规则、布局等。

第二种：锁定单据体或单据体字段

问题 1：如何锁定单据体某几行？

参考答案：

```
    //锁定单据体某几行，所有字段都锁定
        EntryGrid entryGrid = this.getView().getControl("entryentity");
        entryGrid.setRowLock(true,new int[]{0,1});
```

问题 2：如何锁定单据体某一行某一列？

参考答案：

```
    //锁定单据体某行某列
        this.getView().setEnable(false,2,"kded_dealdesc");
```

问题 3：如何锁定某一列？

参考答案：没有锁定整列的接口，需遍历单据体行锁定

```
    int rowCount = this.getModel().getEntryRowCount("entryentity");
    for(int i = 0,i<rowCount ,i++){
        this.getView().setEnable(false,i,"kded_dealdesc");
    }
```

问题 4：如何锁定整个单据体？

参考答案：

```
this.getView().setEnable(false,"entryentity");
```

(3) 设置控件可见性

问题：如何设置单据在审核状态下审核人字段可见？

参考答案：

方式 1：字段的可见性配置勾选“审核可见”。

方式 2：同样在 afterBindData 事件中先判断单据状态是审核状态时，再设置审核人字段可见：**this.getView().setVisible(true, "字段标识");**

补充:其他场景下，影响可见性的因素还包括界面规则、单据类型、布局等。

(4) 设置字段必录性

问题 1：如何通过代码设置单据头字段必录？

参考答案：

```
TextEdit textField = (TextEdit)this.getView().getControl("kded_textfield1");
    textField.setMustInput(true);
```

问题 2：如何通过代码设置单据体字段必录？

参考答案：

```
//设置单据体字段必录，kded_dealdesc 是单据体一个文本字段的标识
    TextEdit textField1 = (TextEdit)this.getView().getControl("kded_dealdesc");
    textField1.setMustInput(true);
    TextProp it2 =(TextProp) textField1.getProperty();
    it2.setMustInput(true);
```

(5) 设置单据体选中行

问题：单据体有数据时，如何设置单据体默认选中第一行数据？

参考答案：苍穹平台的每个控件都有定义好的控件模型，通过控件模型的接口可以间接改变界面显示的内容；设置单据体默认选中行，首先想到这是单据体控件的独有控件属性，所以先获取单据体控件模型 EntryGrid ,查找 EntryGrid 有没有提供对应的方法 ,显然 EntryGrid 有设置选中行的设置方法 selectRows。

```
//数据模型中获取单据体的数据行数
```

```
int entryRowCount = this.getModel().getEntryRowCount("sunp_entryentity");
if (entryRowCount!=0) {
    //获取单据体控件模型
    EntryGrid eg = this.getView().getControl("sunp_entryentity");
    //通过单据体控件模型的方法实现选中行功能
    eg.selectRows(0);
}
```

(6) 执行页面的按钮点击事件

问题：如何通过代码触发工具栏或按钮的点击逻辑？

参考答案：先获取对应的控件模型，控件模型提供了点击方法。

```
//tbmain 是工具栏标识，bar_save 是工具栏项标识，save 是工具栏绑定的操作
Toolbar toolbar = this.getView().getControl("tbmain");
toolbar.itemClick("bar_save", "save");
//触发按钮，kded_btsave 是按钮标识
Button button = this.getView().getControl("kded_btsave");
button.click();
```

(7) 执行操作

问题：如何通过代码触发某个操作？

参考答案：界面模型提供了相关接口。

```
this.getView().invokeOperation("save");//save 是操作标识
```

(8) 刷新数据

问题 1：通过插件给单据头的字段赋值，发现值没显示出来时，应该如何刷新字段的值？

参考答案：

```
this.getView().updateView("字段标识");
//或者
this.getView().updateView();
```

问题 2：通过插件给单据体的字段赋值，发现值没显示出来时，应该如何刷新单据体字段的值？

参考答案：

```
this.getView().updateView("单据体字段", 所在行号);
//如果刷新很多单据体字段，可以选择刷新整个单据体
this.getView().updateView("单据体标识");
//或者
this.getView().updateView();
```

问题 3：在用户打开查看单据的某条数据时，并在插件中通过 BusinessDataServiceHelper.loadSingle 查数据库中对应的数据进行修改,并通过 SaveServiceHelper.update 更新数据到数据库，那么用户此时该如何刷新修改的数据到当前页面

参考答案：

```
//或者调用表单的“刷新”操作
this.getView().invokeOperation("refresh");
```

推荐 :由于刷新是有性能消耗的 ,updateView 建议指定刷新字段或刷新单据体的标识，尽量不要进行全局刷新-**this.getView().updateView()**。

(9) 页面关闭

问题 1：在修改了数据之后，没有保存就退出，会提示如下图。如何取消该校验？

参考答案：提示是在页面关闭时的校验，那么可以在页面关闭前事件 beforeClosed 中，通过修改事件参数取消校验。

```
@Override
```

```
public void beforeClosed(BeforeClosedEvent e) {
    super.beforeClosed(e);
    e.setCheckDataChange(false);
}
```

问题 2：如何通过代码关闭页面。

参考答案：

```
this.getView().close();
    //或者
    this.getView().invokeOperation("close");
```

(10) 页面弹出提示信息

问题 1：如何通过代码弹出提示信息？

参考答案：

```
this.getView.showMessage("提示信息");
```

更多提示方式见：

https://vip.kingdee.com/school/238714716992919296?topicId=239383452913417728&stageId=23938367157087385 6&pathId=23941065777356569 6&productLineId=29

(11) 修改控件名称

问题 1：如何通过代码修改字段控件的名称？

参考答案：TextEdit 的父类 FieldEdit 提供 setCaption 接口修改控件名称。如：

```
TextEdit textField = (TextEdit)this.getView().getControl("kded_textfield1");
    textField.setCaption(new LocaleString("sss"));
```

注意：当通过该方式修改字段标题后，修改该字段退出界面会提示已有 XXX 字段修改，是否继续退出，这里还是显示修改前的字段名称，需要通过以下代码修改主数据模型对应属性的标题：

```
//对应修改元数据对应属性的标题
Map<String, IDataEntityProperty> allFields
=this.getModel().getDataEntityType().getAllFields();
IDataEntityProperty kded_textfield1 = allFields.get("kded_textfield1");
kded_textfield1.getDisplayName().setLocaleValue("sss");
```

问题 2：如何通过代码修改非字段类型的控件名称，比如按钮、高级面板等？

参考答案：使用动态更新元数据属性的接口 updateControlMetadata。

```
Map<String, Object> text = new HashMap<>();
Map<String, Object> text1 = new HashMap<>();
text.put("zh_CN", "test");
text1.put("text", text);
this.getView().updateControlMetadata("控件标识", text1);
```

(12) 页面缓存

问题：如何使用页面缓存？

参考答案：

```
this.getView().getPageCache().put("isLeaf","true");
    String isLeaf = this.getView().getPageCache().get("isLeaf");
```

(13) 获取当前页面的元数据信息

问题：如何获取运行期表单及表单所有控件信息？

参考答案：

元数据相关的 servicehelper：MetadataServiceHelper、AppMetaServiceHelper、ConvertMetaServiceHelper、LocaleMetadataServiceHelper 等，优先使用这些。（注意不要用到非平台的）

```
//根据表单编码获取表单 id
String id = MetadataDao.getIdByNumber("kdec_cehsi", MetaCategory.Form);
//获取表单元数据
FormMetadata formMeta = (FormMetadata) MetadataDao.readRuntimeMeta(id, Meta
Category.Form);
//遍历获取所有控件集合
for (ControlAp<?> item : formMeta.getItems()) {
    //可见性
    String visible = item.getVisible();
    //控件名称
    String name = item.getName().getLocaleValue();
     //控件编码
     String key = item.getKey();
```

```
}
```

二、前后端交互

1. 用户输入

(1) 数据联动

问题：办公用品登记单选择物品后，如何自动给物品分类赋值？

参考答案：

方式 1：使用基础资料属性字段，然后绑定物品字段，显示属性选择物品分类名称。

方式 2：通过业务规则配置【计算定义公式的值并填写到指定列 】服务实现。

方式 3：在值更新事件 propertyChanged 中获取变更的物品值，并给“物品分类”字段赋值。

```
@Override
public void propertyChanged(PropertyChangedArgs e) {
    if (e.getProperty().getName().equals("物品字段标识")) {
        ChangeData changeData = e.getChangeSet()[0];
        //变更后的数据
        DynamicObject newValue = (DynamicObject) changeData.getNewValue();
        DynamicObject
        goodObj=BusinessDataServiceHelper.loadSingle(newValue.getPkValue(), "物品的标识", "物品里分类字段标识");
        DynamicObject groupObj = goodObj.getDynamicObject("物品里分类字段标识");
        this.getModel().setValue("物品分类字段标识", groupObj);
    }
}
```

(2) 通过代码赋值后不想触发值更新事件

方式 1：在 afterCreatedNewData 事件赋值，是不会触发值更新事件的。

方式 2：在其他事件赋值时，使用以下方式赋值。

```
this.getModel().beginInit();
this.getModel().setValue("字段标识","字段值");
this.getModel().endInit();
```

2. 用户点击

苍穹平台表单页面的点击及触发原理也是基于 Java 委托事件处理机制，关键对象包括如下：

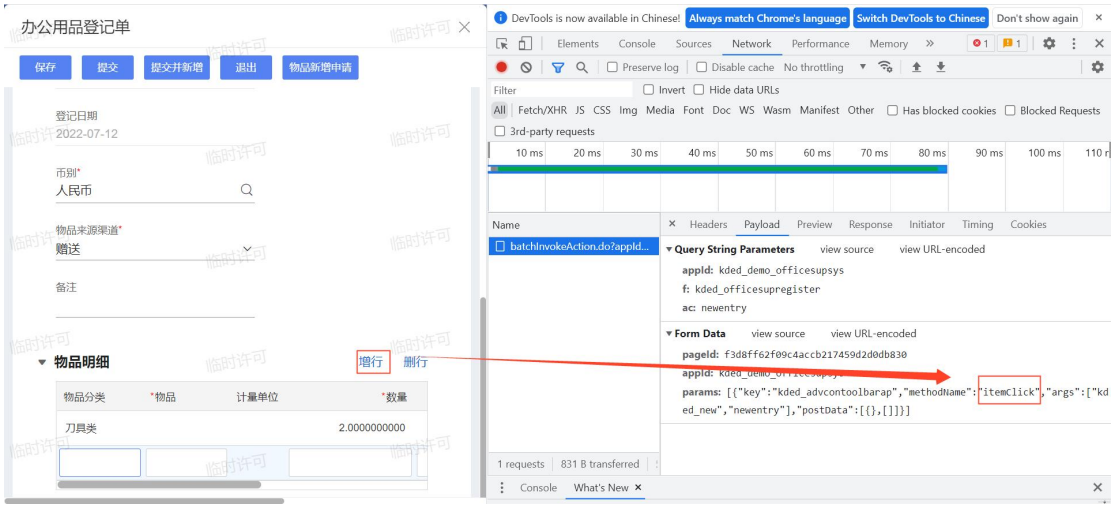
Event Source(事件源)：事件发生的场所，通常就是各个组件，例如按钮、窗口、菜单、工具栏（在苍穹上对应 Toolbar 控件）等。

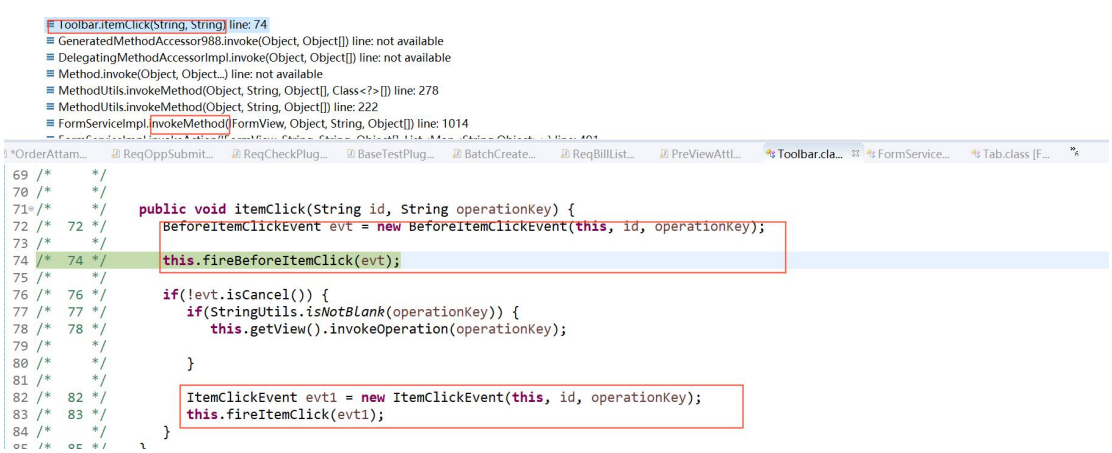
Event(事件)：事件封装了 GUI 组件上发生的特定事情（用户动作、行为）。

Event Listener(事件监听器)：注册于某个事件源上，用于响应特定的事件。

在苍穹平台中，所有标准控件的事件源（Button.class,Toolbal.class,EntryGrid.class 等）已经定义好了，用户点击苍穹表单页面某个地方时，是否会触发特定的响应方法（click、itemClick、billListHyperLinkClick 等），需要满足以下两个条件。

①事件的产生：以用户点击工具栏新增按钮为例，表单微服务根据前端发送的指令最终调用事件源（Toolbar 控件模型）创建了两个事件（BeforeItemClickEvent、ItemClickEvent）。





② 在表单插件的 registerListener 方法中注册 ToolBar 事件源的事件监听器（ClickListener、ItemClickListener)并重写监听器对应的响应事件方法 click、itemClick。

```
@Override
public void registerListener(EventObject e) {
    //添加单据体工具栏的监听
    Toolbar tb = this.getView().getControl("kded_advcontoolbarap");
    tb.addClickListener(this);
    tb.addItemClickListener(this);
    super.registerListener(e);
}
```

由于只产生了 BeforeItemClickEvent、ItemClickEvent 这两个事件，没有 ClickEvent 事件，所以只会响应 itemClick 方法。

```
119    public void beforeClick(BeforeClickEvent evt) {
120        // TODO Auto-generated method stub
121        super.beforeClick(evt);
122    }
123
124    @Override
125    public void click(EventObject evt) {
126        super.click(evt);
127    }
128    @Override
129    public void itemClick(ItemClickEvent evt) {
130        super.itemClick(evt);
131    }
132    @Override
133    public void beforeItemClick(BeforeItemClickEvent evt) {
```

以上举例介绍了下事件的处理机制，下面列举几种常见的干预用户点击事件的处理方法。

(1) 文本字段/按钮点击

问题：如何在点击办公用品登记单的备注（文本字段）时，提示“备注信息不允许修改”？
参考答案：文本字段设置属性“编辑风格”为”按钮点击编辑“时，在 register 事件中注册监听。

```
@Override
public void registerListener(EventObject e) {
    // TODO Auto-generated method stub
    super.registerListener(e);
    TextEdit tEdit = this.getView().getControl("备注字段标识");
    tEdit.addClickListener(this);
}
```

并重写 Click 方法。

```
@Override
public void click(EventObject evt) {
    // TODO Auto-generated method stub
```

```
        super.click(evt);
        Control source = (Control)evt.getSource();
         if ("备注字段标识".equals(source.getKey())){
            // TODO  在此添加业务逻辑
             }
        }
```

(2) 工具栏按钮点击

问题：点击办公用品登记单的提交按钮时，弹出确认提示框是否确认提交。

参考答案：确认是否提交，所以必须在提交按钮执行前处理，点击确认提交后再继续执行提交操作。

```
    @Override
    public void registerListener(EventObject e) {
        super.registerListener(e);
        //监听工具栏按钮点击事件
        this.addItemClickListeners(KEY_TOOlBAR);
    }
    @Override
    public void beforeItemClick(BeforeItemClickEvent evt) {
        //工具栏上的所有按钮的点击都会激活 itemClick 和 beforeItemClick 方法， 需
        //要开发人员实现不同按钮的逻辑
        if (evt.getItemKey().equals("bar_submit")){
            evt.setCancel(true);
            ConfirmCallBackListener confirmCallBackListener = new
            ConfirmCallBackListener("submitconfirm", this);
            //设置页面确认框，参数为：标题，选项框类型，回调监听
            this.getView().showConfirm("您确认提交该办公用品登记单吗？",
            MessageBoxOptions.YesNoCancel, confirmCallBackListener);
        }
        super.beforeItemClick(evt);
    }
    @Override
    public void confirmCallBack(MessageBoxClosedEvent messageBoxClosedEvent) {
        super.confirmCallBack(messageBoxClosedEvent);
```

```
        //判断是否是对应确认框的点击回调事件
    if (("submitconfirm").equals(messageBoxClosedEvent.getCallBackId())) {
        if (MessageBoxResult.Yes.equals(messageBoxClosedEvent.getResult())) {
            //如果点击确认按钮，则调用提交操作
            this.getView().invokeOperation("submit");
        } else if (MessageBoxResult.No.equals(messageBoxClosedEvent.getResult())) {

        } else if
        (MessageBoxResult.Cancel.equals(messageBoxClosedEvent.getResult())) {
            // 点击取消的相关处理逻辑。。。。
        }
    }
```

(3) 基础资料点击

问题：办公用品登记单先选择物品分类，然后在选择物品时，只显示已选物品分类的物品。
参考答案：在选择物品前，添加过滤条件。需要监听物品基础资料的选择前事件。

```
@Override
public void registerListener(EventObject e) {
    super.registerListener(e);
    BasedataEdit bde = this.getView().getControl("物品字段字段标识");
    bde.addBeforeF7SelectListener(this)
}
```

并实现 BeforeF7SelectListener 接口及重写 beforeF7Select 方法。最后在 beforeF7Select 方法中设置过滤条件，参考开发案例
【https://vip.kingdee.com/article/198077267325441280?productLineId=29】。

(4) 单据体点击

①单据体单元格点击

问题 1：单据体上添加了个文本字段，数据存储了某个附件 url 的下载地址，锁定状态；点击某个 url 时，下载该文件。
参考答案：先在 registerListener 事件注册单据体单元格监听器。

```
@Override
public void registerListener(EventObject e) {
    super.registerListener(e);
    EntryGrid eg = this.getView().getControl("sunp_entryentity");
    eg.addCellClickListener(this);
}
```

然后实现 CellClickListener 接口及 cellClick 方法

```
@Override
public void cellClick(CellClickEvent arg0) {
    //获取点击单元格的字段及值
    String fieldKey = arg0.getFieldKey();
    if (fieldKey.equals("字段标识")) {
        String url = (String) this.getModel().getValue(fieldKey, arg0.getRow());
        this.getView().openUrl(url);
        //或者 this.getView().download(url);
    }
    System.out.println("xx");
}
```

②单据体行点击

问题 2：在办公用品登记单的单据体的工具栏上添加一个查看库存按钮，查看选择行的物品的库存，在选择行时，校验是否物品为空。

参考答案：单据体选择行不需要考虑点击的字段，主要鼠标点击时在单据体某行上时即可。所以注册行点击监听器。

```
@Override
public void registerListener(EventObject e) {
    super.registerListener(e);
    EntryGrid eg = this.getView().getControl("sunp_entryentity");
    eg.addRowClickListener(this);
}
```

并实现接口 RowClickEventListener 及实现 entryRowClick 方法，在 entryRowClick 方法中获取物品值是否为空，为空则取消选择行。

③用代码选中单据体行

```
this.getModel().setEntryCurrentRowIndex("单据体标识",rowIndex);
```

④ 获取单据体当前选中行

```
//方式 1-获取当前选中行
this.getModel().getEntryCurrentRowIndex("单据体标识");//只能获取当前选中行
//方式 2-获取所有选中行
//通过单据体控件获取选中行
    EntryGrid entryGrid = this.getView().getControl("sunp_entryentity");
    int[] selectRows = entryGrid.getSelectRows();
```

(5) 单据列表超链接点击

问题：办公用品登记单的单据列表点击某个超链接时，取消原有的弹窗逻辑，自定义弹出该表单。

参考答案：

```
/**
* 用户点击超链接单元格时，触发此事件
*/
@Override
public void billListHyperLinkClick(HyperLinkClickArgs args) {
    if (StringUtils.equals("超链接所在列的字段标识",
    args.getHyperLinkClickEvent().getFieldName())){
        // 取消系统自动打开本单的处理
        args.setCancel(true);
        int rowIndex = args.getRowIndex();//点击的行下标
        BillList bl = this.getView().getControl("billlistap");
        ListSelectedRowCollection currentListAllRowCollection =
        bl.getCurrentListAllRowCollection();
        ListSelectedRow listSelectedRow =
        currentListAllRowCollection.get(rowIndex);
        //点击行数据
```

```
            Object primaryKeyValue = listSelectedRow.getPrimaryKeyValue();//主键值
            String formID = listSelectedRow.getFormID();
            BillShowParameter bsp = new BillShowParameter();
            bsp.setFormId(formID);
            bsp.setPkId(primaryKeyValue);
            bsp.getOpenStyle().setShowType(ShowType.Modal);
            bsp.setStatus(OperationStatus.EDIT);
            this.getView().showForm(showParameter);
        }
    }
```

点击单据列表时，同时会触发执行 listRowClick 方法。大家可能好奇为什么单据列表中点击为什么不用注册监听器。这是因为标准的单据列表视图模型对象中注册了相关监听器：kd.bos.mvc.list.AbstractListView.registerListener()。

(6) 列表获取当前选中行

方式 1：通过列表控件模型获取

```
BillList billlistap = this.getView().getControl("billlistap");
ListSelectedRowCollection selectedRows = billlistap.getSelectedRows();
```

方式 2： 在列表插件实例自带接口

```
ListSelectedRowCollection selectedRows1 = this.getSelectedRows();
```

3. 父子页面交互

比如办公用品登记单点击物品新增申请 ,在新增申请表单点击确定后把物品名称给办公物品登记单的备注字段”。

(1) 代码弹出表单页面

第一步：办公用品登记单插件的 itemClick 方法中弹出物品新增申请单。

```
    @Override
    public void itemClick(ItemClickEvent evt) {
        FormShowParameter fsp = new FormShowParameter();
        fsp.setFormId("物品新增申请单");
        fsp.getOpenStyle().setShowType(ShowType.Modal);
```

```
        fsp.setCloseCallBack(new CloseCallBack(this, "show-kded_supaddnew"));//
        this.getView().showForm(fsp);
        super.itemClick(evt);
    }
```

(2) 如何修改父页面的数据

方案 1：直接在子页面的表单插件中拿到父页面的数据模型 model 进行赋值。

```
    @Override
    public void click(EventObject evt) {
        if (evt.getSource() instanceof Button) {
            Button bt=(Button) evt.getSource();
            String key = bt.getKey();
            if (key.equals("btnok")) {
                Object goodName = this.getModel().getValue("物品名称字段标识");
                IFormView parentView = this.getView().getParentView();//父页面
                IDataModel parentModel = parentView.getModel();
                parentModel.setValue("sunp_textfield",goodName ) );//修改父页面数据
                parentView .updateView();//刷新
                //调用了其他表单的控制方法时，需调用以下方法将目标表单的控制 指
                //令发给前端
                this.getView().sendFormAction(this.getView().getParentView());
                this.getView().close();
                }
            }
            super.click(evt);
        }
```

方案 2：将子页面的字段值传给父页面，然后在父页面的表单插件的 closedCallBack 方法中修改备注字段值。

新增物品申请单表单插件：

```
    @Override
    public void click(EventObject evt) {
        if (evt.getSource() instanceof Button) {
            Button bt=(Button) evt.getSource();
            String key = bt.getKey();
```

```
            if (key.equals("btnok")) {//确定按钮或者是邮件通知按钮
                Object goodName = this.getModel().getValue("物品名称字段标识");/
                this.getView().returnDataToParent(goodName);//返回数据给父页面
                this.getView().close();
            }
            }
        }
```

办公用品登记单插件：

```
    @Override
    public void closedCallBack(ClosedCallBackEvent closedCallBackEvent) {
        if (closedCallBackEvent.getActionId().equals("show-kded_supaddnew")) {
            Object returnData = closedCallBackEvent.getReturnData();
            this.getModel().setValue("sunp_textfield",returnData);
        }
        super.closedCallBack(closedCallBackEvent);
    }
```

方案 3：将子页面的值放到父页面缓存，由父页面插件进行赋值处理。

新增物品申请单表单插件：

```
    @Override
    public void click(EventObject evt) {
        if (evt.getSource() instanceof Button) {
            Button bt=(Button) evt.getSource();
            String key = bt.getKey();
            if (key.equals("btnok")) {//确定按钮或者是邮件通知按钮
                Object goodName = this.getModel().getValue("物品名称字段标识");/
                //通过界面缓存传递值
                this.getView().getParentView().getPageCache().put("name", goodName );
                this.getView().close();
                }
            }
        }
    }
```

办公用品登记单插件：

```
    @Override
    public void closedCallBack(ClosedCallBackEvent closedCallBackEvent) {
        if (closedCallBackEvent.getActionId().equals("show-kded_supaddnew")) {
```

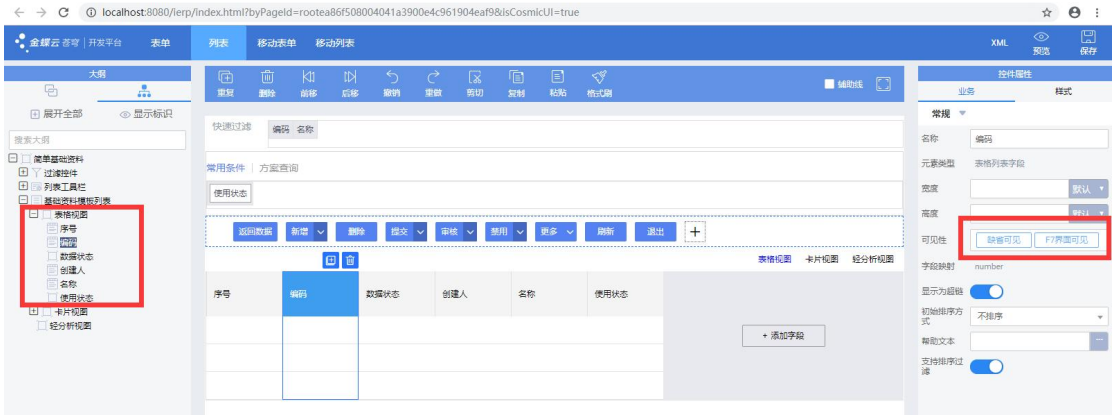
```
            this.getModel().setValue("sunp_textfield",this.getView().getPageCache().
            get("name"));
        }
        super.closedCallBack(closedCallBackEvent);
    }
```

4. 基础资料选择界面（F7）

(1) F7 选择界面显示哪些字段

基本上 F7 选择界面都是统一的界面，普通基础资料用的是 bos_listf7，分组基础资料和树形基础资料用的是 bos_templatetreelistf7 界面，部分基础资料，比如人员、组织、客户、供应商等基础资料有自己的 f7 界面。

F7 选择界面的字段由基础资料列表的字段决定，给列表增加字段，并且【可见性】选上 F7 界面可见。



(2) F7 选择界面显示未审核的数据

```
@Override
    public void beforeF7Select(BeforeF7SelectEvent beforeF7SelectEvent) {

if("kded_basedatafield".equals(beforeF7SelectEvent.getProperty().getName())){
            ListShowParameter listShowParameter = (ListShowParameter)
            beforeF7SelectEvent.getFormShowParameter();
            listShowParameter.setShowApproved(false);
        }
```

```
}
```

(3) F7 选择界面展示已选择的数据，开启多选等

```
public void beforeF7Select(BeforeF7SelectEvent beforeF7SelectEvent) {

if("kded_basedatafield".equals(beforeF7SelectEvent.getProperty().getName())){
            ListShowParameter listShowParameter = (ListShowParameter)

        beforeF7SelectEvent.getFormShowParameter();
            listShowParameter.setShowApproved(false);
            DynamicObject baseData = (DynamicObject)
        this.getModel().getValue("kded_basedatafield");
            if(baseData != null) {
              //设置单个已选
                listShowParameter.setSelectedRow(baseData.getPkValue());
            listShowParameter.setSelectedRows(new
            Object[]{baseData.getPkValue()});//设置多个已选
                listShowParameter.setMultiSelect(true);//开启多选
             }
    }
```

三、后端数据操作

1. 如何新建数据对象并赋值

(1) 新建空数据对象

可以使用 DynamicObject 的构造函数 ,也可以使用 BusinessDataServiceHelper 或 ORM 提供的接口。

方式 1：

```
DynamicObject data
=BusinessDataServiceHelper.newDynamicObject("kded_simplebill");
```

方式 2：

```
DynamicObject kded_simplebill =
ORM.create().newDynamicObject("kded_simplebill");
```

方式 3

```
MainEntityType type =
EntityMetadataCache.getDataEntityType("kded_simplebill");
DynamicObject data1 = new DynamicObject((DynamicObjectType) type);
```

(2) 给对象赋简单字段的值

```
data.set("billno","测试 0002");
data.set("billstatus","A");
```

(3) 给对象新增单据体并赋值

```
DynamicObjectCollection entryentity =
data.getDynamicObjectCollection("entryentity");
//新增单据体方式一：
DynamicObject dynamicObject = entryentity.addNew();
dynamicObject.set("kded_textfield","代码新增单据体方式 1");

//新增单据体方式二：
DynamicObject dynamicObject1 =
new DynamicObject(entryentity.getDynamicObjectType());
dynamicObject1.set("kded_textfield","代码新增单据体方式 2");
entryentity.add(dynamicObject1);
```

(4) 给对象新增子单据体并赋值

```
DynamicObjectCollection entryentity = data.getDynamicObjectCollection("entryentity
");
DynamicObject entry = new DynamicObject(entryentity.getDynamicObjectType());
entry.set("kdec_textfield","hello1");
entry.set("kdec_integerfield", 95271);
entryentity.add(entry);
```

```
//单据体数据
DynamicObjectCollection subentryentity = entry.getDynamicObjectCollection("subentryentity");
//单据体的子单据体
DynamicObject subentry = new DynamicObject(subentryentity.getDynamicObjectType());
subentry.set("kdec_textfield1","world");
subentry.set("kdec_integerfield1", 12138); subentryentity.add(subentry);
```

2. 如何保存数据对象

(1) SaveServiceHelper

```
//方式 1-走保存操作的校验
    OperationResult operationResult1 =
SaveServiceHelper.saveOperate("kded_simplebill", new DynamicObject[]{data},
OperateOption.create());
//方式 2-直接存库
    SaveServiceHelper.save(new DynamicObject[]{data});
```

(2) OperationServiceHelper

该接口可以执行单据的操作，如执行提交操作：

```
OperationResult operationResult =
OperationServiceHelper.executeOperate("submit","kded_simplebill",new
DynamicObject[]{data},OperateOption.create());
if(!operationResult.getSuccessPkIds().isEmpty()){
    //执行成功
}
```

3. 如何查询数据对象

(1) 查询简单字段&使用简单字段做过滤条件

```
//方式 1-根据单据 id 查询-可以查询所有字段
    Object billId = "1515996182680175616";
    DynamicObject simpleBill = BusinessDataServiceHelper.loadSingle(billId,
"kded_simplebill");
    //方式 2-根据单据 id 查询-查询指定字段，查询结果是平铺的
    QFilter filter = new QFilter("id", QCP.equals, billId);
    DynamicObject simpleBillObject =
QueryServiceHelper.queryOne("kded_simplebill",  "id,billno,createtime",  new
QFilter[]{filter});
```

(2) 查询复杂字段&使用复杂字段做过滤条件

```
//方式 1-根据创建人的名称构造过滤条件-BusinessDataServiceHelper
    QFilter filter2 = new QFilter("creator.name", QCP.equals, "XXX");
    DynamicObject simpleBill2 =
BusinessDataServiceHelper.loadSingle("kded_simplebill","id,billno,creator,creato
r.id,createtime",new QFilter[]{filter2});
    DynamicObject creator = simpleBill2.getDynamicObject("creator");
    long creatorId = simpleBill2.getLong("creator.id");
    //方式 2-根据创建人的名称构造过滤条件-QueryServiceHelper
    DynamicObject simpleBillObject3 =
QueryServiceHelper.queryOne("kded_simplebill",
"id,billno,creator,creator.id,createtime", new QFilter[]{filter2});
     DynamicObject creator1 = simpleBill2.getDynamicObject("creator");
```

(3) 查询单据体字段&使用单据体字段做过滤条件

①使用 BusinessDataServiceHelper

```
//根据单据体的字段构造过滤条件
    QFilter filter3 = new QFilter("entryentity.kded_dealuser.name", QCP.equals, "金小
蝶");
    DynamicObject simpleBill3 =
BusinessDataServiceHelper.loadSingle("kded_simplebill","id,billno,kded_dealuse
r,kded_dealdesc",new QFilter[]{filter3});
    DynamicObjectCollection cols =
simpleBill3.getDynamicObjectCollection("entryentity");
    for(DynamicObject col:cols){
        DynamicObject user = col.getDynamicObject("kded_dealuser");
        String name = user.getString("name");
        String dec = col.getString("kded_dealdesc");
    }
```

注意点：

1.过滤条件使用单据体字段做过滤条件需要加上单据体标识，如 entryentity.kded_dealuser.name；

2.查询字段 selectfields 查询单据体字段可不用加单据体标识，如 DynamicObject user = col.getDynamicObject("kded_dealuser");

3.user 默认有 id，number、name 字段，如需其他字段，需要单据给该基础资料字段配置引用属性。

②使用 BusinessDataServiceHelper

```
QFilter filter5 = new QFilter("entryentity.kded_dealuser.name", QCP.equals, "金小
蝶");
    DynamicObjectCollection simpleBills =
QueryServiceHelper.query("kded_simplebill","id,billno,entryentity.kded_dealuser.
name,entryentity.kded_dealdesc",new QFilter[]{filter5});
    for (DynamicObject dynamicObject : simpleBills) {
```

```
            dynamicObject.get("entryentity.kded_dealuser.name");
        }
```

注意点：过滤条件和查询字段使用单据体字段都需要用单据体标识 entryentity， 指定什么字段则查出什么字段。

(4) 查询子单据体字段&使用单据体字段做过滤条件

```
    //根据单据体的字段构造过滤条件查询子单据体字段
        QFilter filter4 = new QFilter("entryentity.kded_dealuser.name", QCP.equals, "金小蝶      ");
        DynamicObject simpleBill4 = BusinessDataServiceHelper.loadSingle("kded_simplebill","id,billno,kded_dealuser, kded_dealdesc,kded_subdealuser,kded_subdec",new QFilter[]{filter4});
        DynamicObjectCollection cols1 = simpleBill4.getDynamicObjectCollection("entryentity");
        for(DynamicObject col:cols1){
            //获取单据体字段
             DynamicObject user = col.getDynamicObject("kded_dealuser");
            String name = user.getString("name");
            String dec = col.getString("kded_dealdesc");
            DynamicObjectCollection subCols = col.getDynamicObjectCollection("kded_subentryentity");
            for (DynamicObject subCol:subCols) {
                //获取子单据体字段
                DynamicObject subDealUser = subCol.getDynamicObject("kded_subdealuser");
                String subDec = subCol.getString("kded_subdec");
            }
    }
```

4. 如何删除数据对象

（1）直接从数据库删除数据，不触发数据校验

```
DynamicObject dObj = null;
//IDataEntityType 为实体类型基类
IDataEntityType dataEntityType = dObj.getDataEntityType();
//pks 为数据主键值
DeleteServiceHelper.delete(IDataEntityType type, Object[] pks)
```

（2）删除数据时，触发数据校验

```
DeleteServiceHelper deleteServiceHelper = new DeleteServiceHelper();
deleteServiceHelper.deleteOperate("删除的操作代码", "单据标识", pks,
OperateOption.create());
//或者直接调用通用操作接口
OperationServiceHelper.executeOperate("删除的操作代码", "单据标识", pks,
OperateOption.create())
```

四、操作

1. 代码触发操作

(1) 界面插件触发操作

界面模型 View 提供了触发操作的接口。

```
this.getView().invokeOperation("submit");
```

(2) 非界面插件触发操作

非界面插件没有界面模型 View，需要通过操作服务工具类触发（也可以界面插件使用）。

```
OperationResult operationResult =
OperationServiceHelper.executeOperate("submit","kded_simplebill",new
DynamicObject[]{data},OperateOption.create());
if(!operationResult.getSuccessPkIds().isEmpty()){
    //todo;
}
```

(3) 代码触发操作如何忽略验权

通过 OperateOption 传递对应参数，可以绕过权限校验。

```
OperateOption option = OperateOption.create();
option.setVariableValue(OperateOptionConst.ISHASRIGHT, "true");OperationResult
```

2. 操作后如何刷新字段

(1) 配置实现

单据状态、日期这两种类型的字段可以在操作配置刷新字段实现。

(2) 代码实现

```
@Override
    public void afterDoOperation(AfterDoOperationEventArgs
afterDoOperationEventArgs) {
        super.afterDoOperation(afterDoOperationEventArgs);
        FormOperate formoperate = (FormOperate)
    afterDoOperationEventArgs.getSource();
        if("操作代码".equals(formoperate.getOperateKey())) {
            if(!afterDoOperationEventArgs.getOperationResult().getSuccessPkIds().
        isEmpty()) {
                this.getView().updateView("字段标识");
            }
         }
```

3. 操作后如何提示

(1) 配置提示

固定的提示语可以直接在操作进行配置，如图：

(2) 代码修改操作后的提示

首先得在操作配置操作成功后提示，通过代码修改提示内容才会弹出显示。

方式 1：在操作插件实现。

```
@Override
public void afterExecuteOperationTransaction(AfterOperationArgs e) {
    if("save".equals(e.getOperationKey())&&!this.getOperationResult().getSuccessPkI
    ds().isEmpty()) {
```

```
            this.getOperationResult().setMessage("保存成功啦-操作插件");
        }
    }
```

方式 2：在界面插件实现。

```
    @Override
        public void afterDoOperation(AfterDoOperationEventArgs
    afterDoOperationEventArgs) {
            super.afterDoOperation(afterDoOperationEventArgs);
            FormOperate formoperate = (FormOperate)
        afterDoOperationEventArgs.getSource();
            if("save".equals(formoperate.getOperateKey())&&!afterDoOperationEventA
            rgs.getOperationResult().getSuccessPkIds().isEmpty()) {
                afterDoOperationEventArgs.getOperationResult().setMessage("保存成
            功！-界面插件");
                }
    }
```

4. 操作插件跟界面插件如何传递参数

(1) 从界面传递参数给操作插件

首先在界面插件给操作塞参数

```
    @Override
        public void beforeDoOperation(BeforeDoOperationEventArgs args) {
          super.beforeDoOperation(args);
            FormOperate formoperate = (FormOperate) args.getSource();
            if("save".equals(formoperate.getOperateKey())) {
                formoperate.getOption().setVariableValue("customItem",this.getView().
            getPageCache().get("customItem"));
            }
    }
```

然后在操作插件取参数：

```
    @Override
        public void beginOperationTransaction(BeginOperationTransactionArgs e) {
            super.beginOperationTransaction(e);
```

```
        if ("save".equals(e.getOperationKey())) {
            //取出界面传过来的参数
           String customItem = this.getOption().getVariableValue("customItem");
           }
    }
```

(2) 从操作插件传递参数给界面

首先在操作插件给操作 Option 塞参数

```
    @Override
        public void afterExecuteOperationTransaction(AfterOperationArgs e) {
        if("save".equals(e.getOperationKey())&&!this.getOperationResult().getSuccessPkI
        ds(). isEmpty()) {
                this.getOption().setVariableValue("allBillNo","");
            }
        }
```

然后在界面插件取参数

```
    @Override
        public void afterDoOperation(AfterDoOperationEventArgs
    afterDoOperationEventArgs) {
            super.afterDoOperation(afterDoOperationEventArgs);
            FormOperate formoperate = (FormOperate)
        afterDoOperationEventArgs.getSource();
            if("save".equals(formoperate.getOperateKey())&&!afterDoOperationEventA
    rgs.g       etOperationResult().getSuccessPkIds().isEmpty()) {
                String                                  allBillNo                                =
    formoperate.getOption().getVariableValue("allBillNo");
           }
        }
```