**Activiti工作流开发手册**

**XXX公司版权所有**

修改记录

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **制定日期** | **描述** | **制定 /修订内容摘要** | **页数** | **版本** | **拟稿** | **审查** | **批准** |
| 2016-09-5 |  | 创建 |  | V0.1 | 董飞 |  |  |
| 2016-09-28 |  | 修改 |  | V0.2 | 董飞 |  |  |
| 2016-09-29 |  | 修改 |  | V0.3 | 董飞 |  |  |
| 2016-09-30 |  | 修改 |  | V0.4 | 琚军军 |  |  |
| 2016-10-24 |  | 修改 |  | V0.5 | 董飞 |  |  |
| 2016-11-23 | 新增流转历史 | 修改 |  | V0.6 | 董飞 |  |  |
| 2017-06-26 | 新增设计流程-可以跳过节点设置说明 | 修改 |  | V0.8 | 董飞 |  |  |
| 2017-12-01 | 新增决议类型 | 修改 |  | V1.0 | 董飞 |  |  |

目录

[一、 新建模型 4](#_Toc465097585)

[1、登录后台管理系统，选择新建模型 4](#_Toc465097586)

[2、选择模型和流程所属系统（流程分类），同一模型只能一个系统使用。点击保存进入流程图设计界面 4](#_Toc465097587)

[二、 设计流程图 5](#_Toc465097588)

[1、在线画图界面 5](#_Toc465097589)

[2、设置流程的Key和流程名称 5](#_Toc465097590)

[3、流程的几个关键节点主键命名要求 6](#_Toc465097591)

[4、设置开始节点 6](#_Toc465097592)

[5、用户任务属性设置(区域经理2) 6](#_Toc465097593)

[6、设置箭头走向的名称和判断条件 7](#_Toc465097594)

[7、会签节点的属性设置 8](#_Toc465097595)

[8、设置会签通过条件 8](#_Toc465097596)

[9、重新提交节点设置 9](#_Toc465097597)

[10、保存设计的流程图 10](#_Toc465097598)

[三、 模型管理 11](#_Toc465097599)

[1、选择一条记录导出，生成的XML 11](#_Toc465097600)

[2、选择一条记录部署模型 11](#_Toc465097601)

[四、 流程管理 11](#_Toc465097602)

[五、 对外提供的几个接口 11](#_Toc465097603)

[1、启动流程 11](#_Toc465097604)

[2、处理任务 12](#_Toc465097605)

[3、获取待办列表 13](#_Toc465097606)

[4、获取已办任务 13](#_Toc465097607)

[5、获取流转历史列表 13](#_Toc465097608)

[六、 业务系统服务层展现层代码 14](#_Toc465097609)

[1、业务申请启动流程 14](#_Toc465097610)

[2、业务审批各节点处理实例 14](#_Toc465097611)

[3、业务系统展现层处理示例 16](#_Toc465097612)

[七、系统配置 17](#_Toc465097613)

[1、工作流地址配置 17](#_Toc465097614)

[八、工作流流程图任务节点设置表单 18](#_Toc465097615)

[1、灵活控制节点访问Controller 18](#_Toc465097616)

[九、跟踪任务流程 18](#_Toc465097617)

[1、跟踪任务流程可以访问工作流Controller 18](#_Toc465097618)

[十、工作流任务执行案例 19](#_Toc465097619)

[1、新建任务 19](#_Toc465097620)

[2、查看代办任务 21](#_Toc465097621)

[3、任务办理 21](#_Toc465097622)

## 新建模型

### 1、登录后台管理系统，选择新建模型



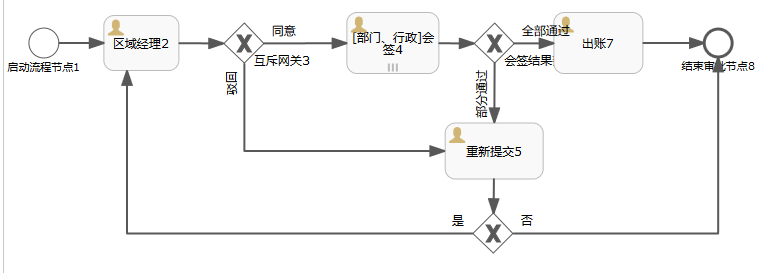
### 2、选择模型和流程所属系统（流程分类），同一模型只能一个系统使用。点击保存进入流程图设计界面

## 设计流程图

### 1、在线画图界面



以会签流程为例：



### 2、设置流程的Key和流程名称



### 3、流程的几个关键节点主键命名要求

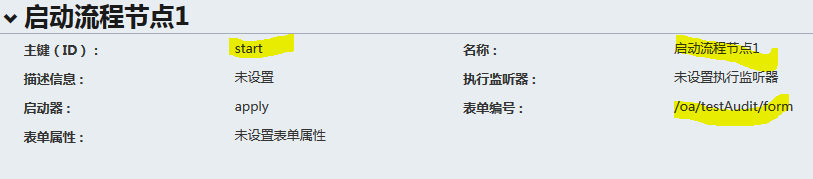
开始节点：start

结束节点：end

修改节点：modify

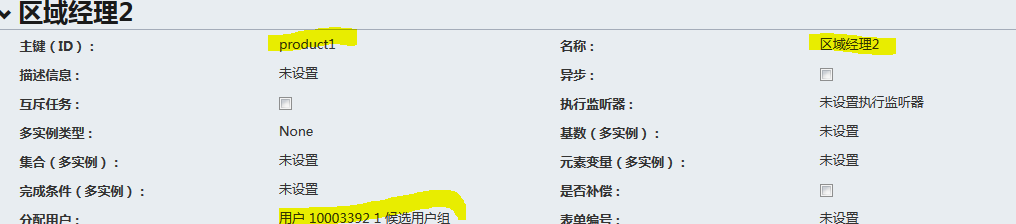
会签节点：以countersign开头的任务名称

### 4、设置开始节点



需要设置表单编号：任务办理时的访问地址（Controller）

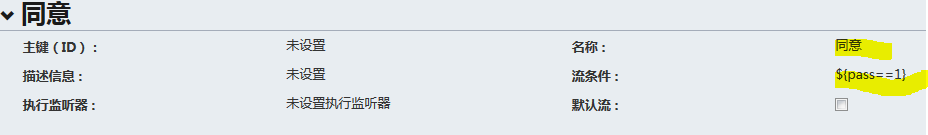
### 5、用户任务属性设置(区域经理2)

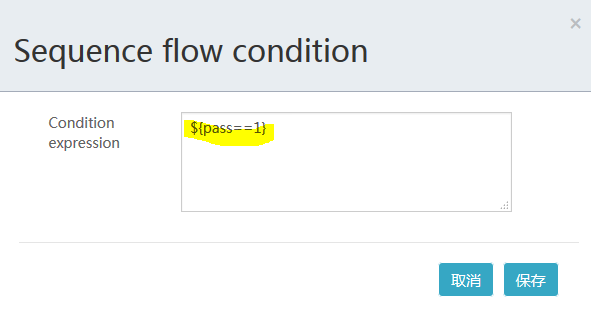


给用户任务分配用户，可以直接设置用户或者候选组



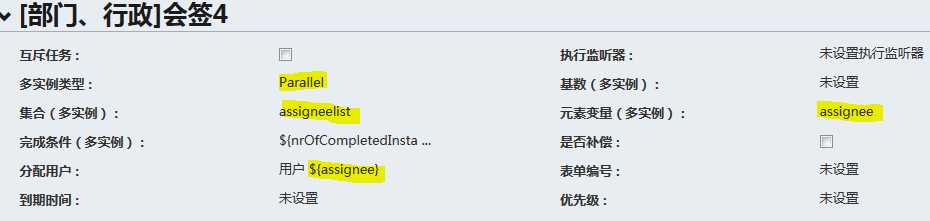
### 6、设置箭头走向的名称和判断条件





### 7、会签节点(多实例节点)

#### 7.1属性设置



设置多实例类型：并行还是顺序执行。

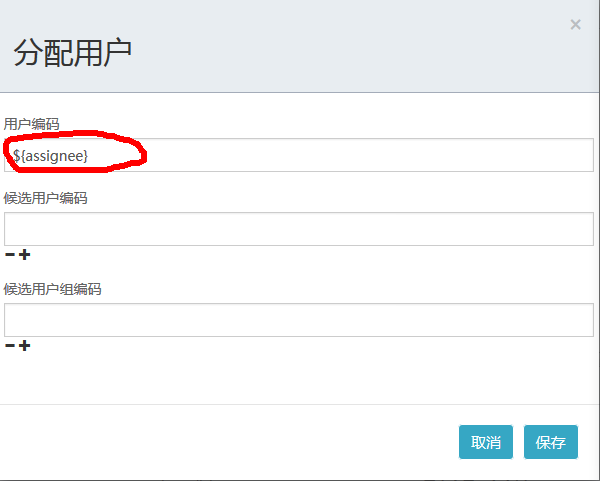
##### 1、集合为会签人员

节点key：countersign\_sub

集合：assigneelist

元素变量：assignee

分配用户：${assignee}



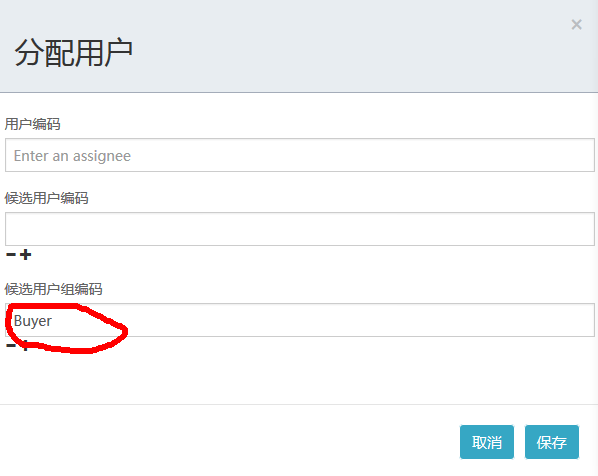
##### 2、集合为其他，需要设置执行任务的角色。

节点key：countersign\_sub

集合：assigneelist

元素变量：assignee

分配：Buyer



##### 3设置流程变量”subkey\_”的值用于子流程、多实例用户权限过滤

调用工作流产生会签任务前需要设置：

//设置会签人员、设置启动子流程、多实例任务集合

**if**(StringUtils.*isNotEmpty*(testAudit.getAssigneelist())){

String[] str=testAudit.getAssigneelist().split(",");

List<String> sublist=**new** ArrayList();

//设置子流程权限过滤KEY

Map subvars =**new** HashMap();

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

//设置生成子流程集合

sublist.add(str[i]);

//如果Subkey不为空，分别设置子流程、多实例用户权限过滤流程变量

**if**(StringUtils.*isNotEmpty*(testAudit.getSubkey())){

Map map =**new** HashMap();

map.put(testAudit.getSubkey(), str[i]);

//可以设置多个

//map.put("department", \*\*\*);

//map.put("branchcom", \*\*\*);

//map.put("productline", \*\*\*);

subvars.put(str[i], map);

}

}

vars.put(Constant.Field.***ASSIGNEELIST***, sublist);

**if**(StringUtils.*isNotEmpty*(testAudit.getSubkey())){

vars.put(“subkey\_”, subvars);

}

}

##### 4获取任务ID和子流程、多实例用户权限过滤的流程变量对应关系

Map varMap=ActTaskService .getVariables(String procInId);

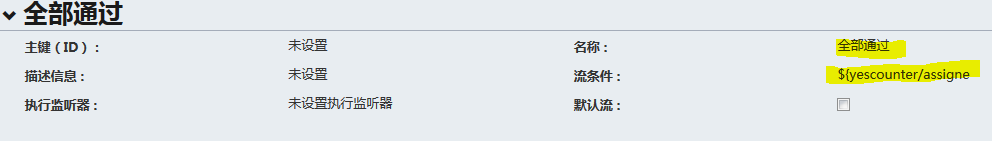
//子流程任务ID和流程变量对照

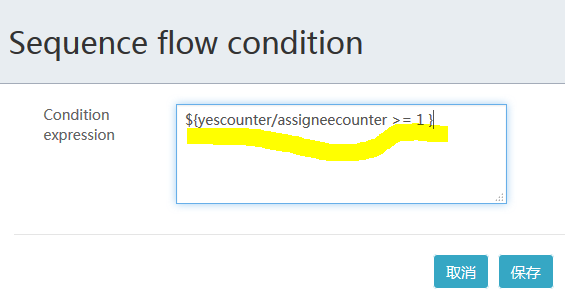
Map subtaskMap=(Map) varMap.get(“subtask\_”);

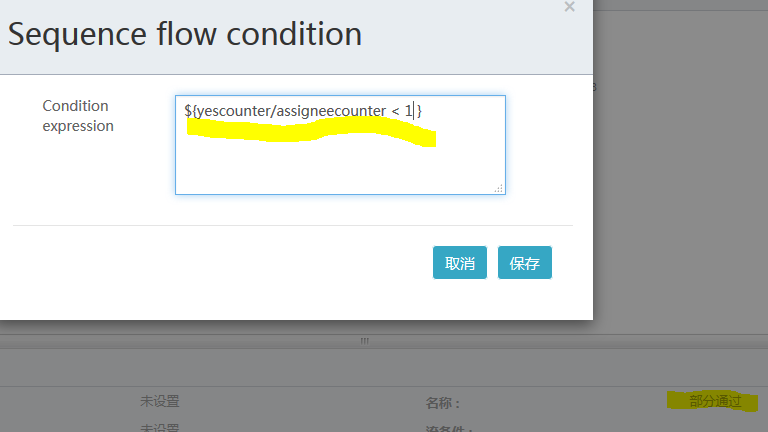
#### 7.2设置会签通过条件

会签同意人数：yescounter\_countersign\_sub

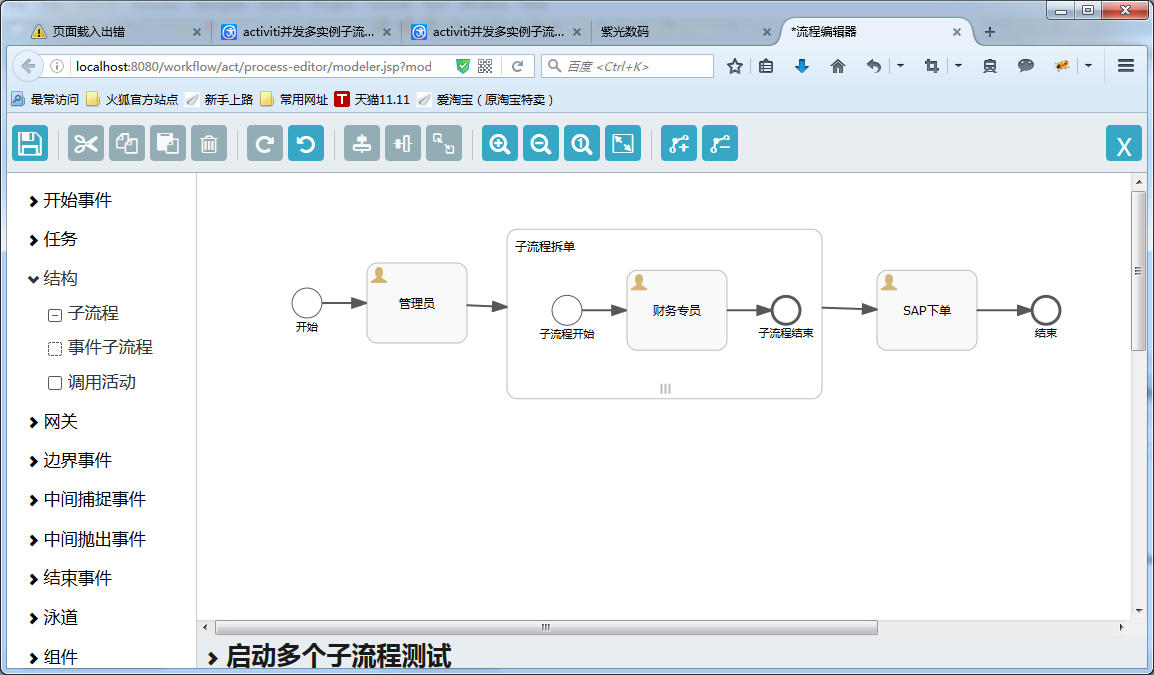
参与会签总人数：assigneecounter\_countersign\_sub



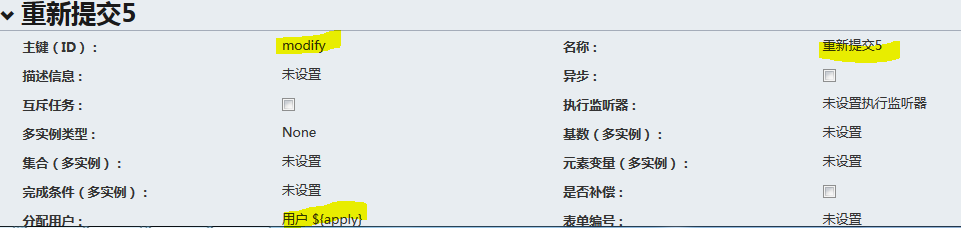




### 8、子流程



### 9、重新提交节点设置





分配用户为：${apply}

### 10、可跳过节点设置

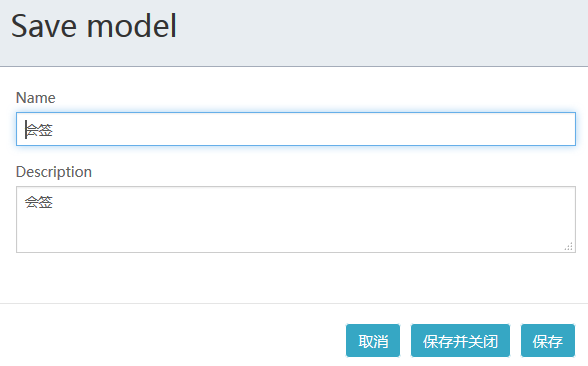
如果同一用户在多个节点都需要他审批，该用户在第一个负责审批的节点审批后，后面需要该用户审批的节点可以跳过。

A、选中可以跳过的节点，设置描述信息为：\_autotask





### 11、保存设计的流程图



## 模型管理



### 1、选择一条记录导出，生成的XML



### 2、选择一条记录部署模型

## 流程管理



## 对外提供的几个接口

/loan-workflow/src/main/java/com/lhhs/workflow/bs/impl/ActTaskServiceImpl.java

### 1、启动流程

/\*\*

\* 启动流程

\*

\* @param parm 启动流程参数

\* @return 流程实例ID

\*/

public String startProcess(ActVo parm);

/\*\* 流程启动需要的参数 \*\*/

/\*\* 启动流程的人员ID登录名 必输 \*\*/

**private** String userId;

/\*\* 系统标识 必输 \*\*/

**private** String sys;

/\*\* 渠道标识 必输 \*\*/

**private** String channel;

/\*\* 流程变量 可以为空 \*\*/

Map<String, Object> vars;

/\*\* 流程启动需要的参数 \*\*/

/\*\* 流程定义KEY 必输 \*\*/

**private** String procDefKey;

/\*\* 业务编号 必输 \*\*/

**private** String businessId;

/\*\* 启动任务标题 必输 \*\*/

**private** String title;

/\*\* 业务表 可以为空 \*\*/

**private** String businessTable;

### 2、处理任务

/\*\*

\* 提交任务, 并保存意见

\*

\* @param parm 提交任务参数

\*/

public String complete(ActVo parm);

/\*\*任务处理需要的参数 \*\*/

/\*\* 启动流程的人员ID登录名 必输 \*\*/

**private** String userId;

/\*\* 系统标识 必输 \*\*/

**private** String sys;

/\*\* 渠道标识 必输 \*\*/

**private** String channel;

/\*\* 流程变量 可以为空 \*\*/

Map<String, Object> vars;

/\*\* 通过类型 必输 \*\*/

String pass;

/\*\* 任务编号 必输 \*\*/

**private** String taskId;

/\*\* 流程实例ID ，如果为空，则不保存任务提交意见 \*\*/

**private** String procInsId;

/\*\* 任务提交意见的内容 \*\*/

**private** String comment;

### 3、获取待办列表

/\*\*

\* 获取待办列表

\* 分页查询

\* **@param** act 工作流实体对象

\* **@return** List<Act>工作流列表

\*/

**public** ResultEasyuiVO todoList(TaskVo act);

### 4、获取已办任务

/\*\*

\* 获取已办任务

\*

\* **@param** page

\* **@param** procDefKey

\* 流程定义标识

\* **@return**

\*/

**public** ResultEasyuiVO historicList(TaskVo act);

### 5、获取流转历史列表

/\*\*

\* 获取流转历史列表

\*

\* @param procInsId

\* 流程实例

\* @param startAct

\* 开始活动节点名称

\* @param endAct

\* 结束活动节点名称

\*/

public List<Act> histoicFlowList(String procInsId, String startAct, String endAct);

**6、决议枚举类**

***YES***("同意", "yes","1"),

***AGAIN***("重走审批", "again","11"),

***PATCHOVER***("补件完成", "patchover","14"),

***NO***("不同意","no", "0"),

***REJECTALL***("驳回修改","rejectall", "-1"),

***REJECT***("修改","reject", "-2"),

***BACK***("返回驳回节点","back", "-3"),

***OVER***("结束任务", "over","90"),

***OVERMAIN***("撤销", "overmain","91"),//主动撤回结束

***REFUSE***("拒贷", "refuse","92"),//被审核人拒绝结束

***CANCEL***("结束当前任务", "cancel","80"),

***PATCH***("补件", "patch","4"),

***SIGN***("会签", "sign","5");

/loan-workflow/src/main/java/com/lhhs/workflow/common/enumeration/Resolution.java

## 业务系统服务层展现层代码

### 1、业务申请启动流程

/\*\*

\* 业务申请启动流程

\* **@param** testAudit

\*/

@Transactional(readOnly = **false**)

**public** **void** save(TestAudit testAudit) {

// 申请发起

……

//设置启动流程参数

ActVo parm =new ActVo(procDefKey, businessId, userId, title, vars, sys, channel);

// 启动流程

String PROC\_INS\_ID =actTaskService.startProcess(parm);

//设置流程实例ID

testAudit.setProcInsId(PROC\_INS\_ID);

//更新业务表

……

}

### 2、业务审批各节点处理

/\*\*

\* 审核审批保存

\* **@param** testAudit

\*/

@Transactional(readOnly = **false**)

**public** **void** auditSave(TestAudit testAudit) {

String userId = UserUtils.*getUser*().getLoginName();

//枚举类型获取

String comment=Resolution.*getName*("", testAudit.getAct().getFlag());

// 设置意见

**if**(comment!=**null**&&!"".equals(comment)){

testAudit.getAct().setComment(comment+testAudit.getAct().getComment());

}

testAudit.preUpdate();

// 对不同环节的业务逻辑进行操作

String taskDefKey = testAudit.getAct().getTaskDefKey();

// 审核环节

**if** ("audit".equals(taskDefKey)){

……

}

**else** **if** ("audit2".equals(taskDefKey)){

……

}

**else** **if** ("audit3".equals(taskDefKey)){

……

}

**else** **if** ("audit4".equals(taskDefKey)){

……

}

**else** **if** ("apply\_end".equals(taskDefKey)){

//结束节点处理

}

// 提交流程任务

Map<String, Object> vars = Maps.*newHashMap*();

String pass=Resolution.*getId*(testAudit.getAct().getFlag(), "");

vars.put("pass", pass);

//设置会签人员、设置启动子流程、多实例任务集合 **if**(StringUtils.*isNotEmpty*(testAudit.getAssigneelist())){

String[] str=testAudit.getAssigneelist().split(",");

List<String> sublist=**new** ArrayList();

//设置子流程权限过滤KEY

Map subvars =**new** HashMap();

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

//设置生成子流程集合

sublist.add(str[i]);

//如果Subkey不为空，分别设置子流程、多实例用户权限过滤流程变量

**if**(StringUtils.*isNotEmpty*(testAudit.getSubkey())){

Map map =**new** HashMap();

map.put(testAudit.getSubkey(), str[i]);

//可以设置多个

//map.put("department", \*\*\*);

//map.put("branchcom", \*\*\*);

//map.put("productline", \*\*\*);

subvars.put(str[i], map);

}

}

vars.put(Constant.Field.***ASSIGNEELIST***, sublist);

**if**(StringUtils.*isNotEmpty*(testAudit.getSubkey())){

vars.put(Constant.Field.***SUBKEY***, subvars);

}

}

//设置启动流程参数

ActVo parm =new ActVo(taskId, procInsId, userId, comment, pass, vars, channel);

//处理审核

actTaskService.complete(parm);

}

### 3、业务系统展现层处理

#### 3.1通过待办或者已办跳转到业务系统FORM传递的参数通过ActVo获取具体属性说明

/\*\*业务编号 必输 \*\*/

**private** String id;

/\*\* 流程定义KEY 必输 \*\*/

**private** String procDefKey;

/\*\* 业务编号 \*\*/

**private** String businessId;

/\*\* 任务编号 必输 \*\*/

**private** String taskId;

/\*\* 流程实例ID \*\*/

**private** String procInsId;

**private** String taskName; // 任务名称

**private** String taskDefKey; // 任务定义Key（任务环节标识）

/\*\*流程定义ID\*/

**private** String procDefId;

**private** String status; // 任务状态（待处理：todo/待签收：claim/已完成：finish）

#### 3.2代码事例

@RequestMapping(value = "form")

**public** String form(ActVo actVo, Model model) {

String view = "testAuditForm";

// 查看审批申请单

**if** (StringUtils.*isNotBlank*(actVo.getId())){

//业务编号

String businessId = actVo.getBusinessId();

// 环节编号

String taskDefKey = actVo.getTaskDefKey();

// 查看工单

**if**(actVo.isFinishTask()){

view = "testAuditView";

}

// 修改环节

**else** **if** ("modify".equals(taskDefKey)){

view = "testAuditForm";

}

// 审核环节

**else** **if** (taskDefKey!=**null**&&(taskDefKey.indexOf("audit")>=0||"countersign".equals(taskDefKey))){

view = "testAuditAudit";

// String formKey = "/oa/testAudit";

// return "redirect:" + ActUtils.getFormUrl(formKey, testAudit.getAct());

}

// 兑现环节

**else** **if** ("apply\_end".equals(temp)){

view = "testAuditAudit";

}

}

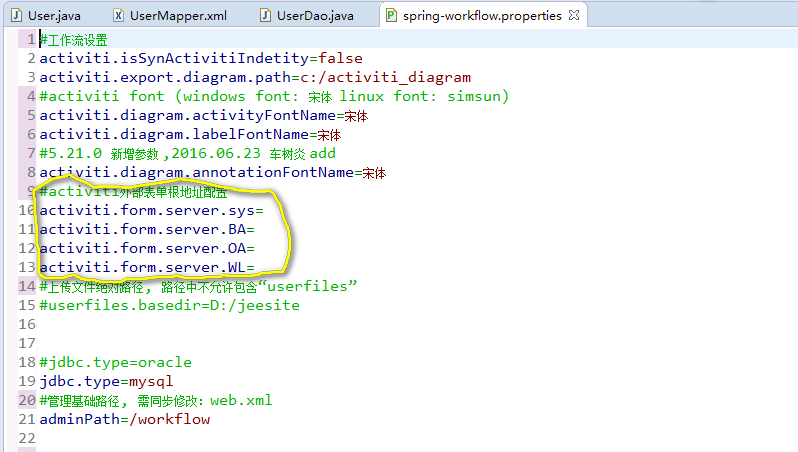
**return** "url/test/" + view;

}

## 七、系统配置

### 1、工作流地址配置

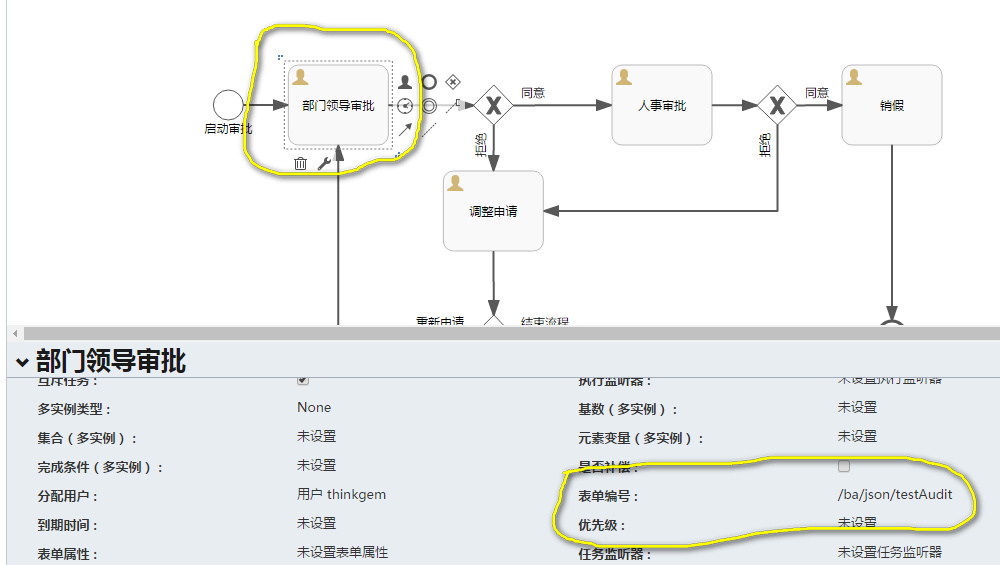
各系统使用工作流需要在工作流的配置文件spring-workflow.properties中配置地址。各系统需要配置好域名，例如<https://IP>：port



## 八、工作流流程图任务节点设置表单

### 1、灵活控制节点访问Controller

可以在任务节点中设置表单编号，办理当前节点的任务是可以跳到当前表单



## 九、跟踪任务流程

## 1、跟踪任务流程可以访问工作流Controller

Controller访问路径：

域名+workflow/url/acttask/tracePhoto?procDefId="+row.procDefId+"

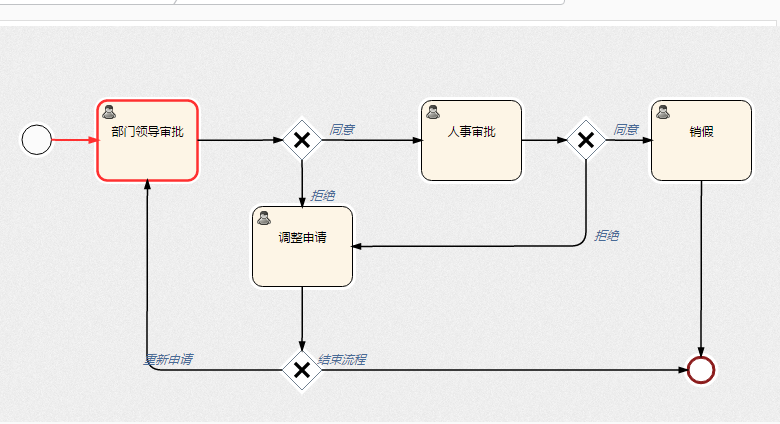
&procInsId="+row.procInsId"，

需要传入两个参数procDefId，procInsId。域名为<https://IP>:端口

示例图：



示例图：



## 十、工作流任务执行案例

### 1、新建任务

选择一个流程



点击启动流程



输入相关任务信息，点击提交申请，任务创建完成



代码参考路径：

/loan-service/src/main/java/com/lhhs/loan/service/scheduler/impl/OrderInfoWorkFlowSchedulerImpl.java

/workflow/src/main/java/com/thunis/workflow/controller/json/TestAuditJsonController.java

对应的方法名:**public** String save(TestAudit testAudit, Model model, RedirectAttributes redirectAttributes,HttpServletRequest request)

### 2、查看代办任务

点击代办任务，可以看到刚才创建的任务



代码参考路径：

/workflow/src/main/java/com/thunis/workflow/controller/json/ActTaskJsonController.java

对应的方法名：**public** Map<String, Object> todoList(ModelMap model,TaskVo act, @RequestParam("page") **int** nowpage, @RequestParam("rows") **int** pageSize,@RequestParam("sort") String sort, @RequestParam("order") String order)

### 3、任务办理

点击任务办理



办理任务，输入调整原因，点击同意就可以审批完成



代码参考路径：

/workflow/src/main/java/com/thunis/workflow/controller/json/ActTaskJsonController.java

对应的方法名：Map<String, Object> historicList(ModelMap model,TaskVo act, @RequestParam("page") int nowpage, @RequestParam("rows") int pageSize,@RequestParam("sort") String sort, @RequestParam("order") String order)

## 十一、流转历史

1. 审批、详情页面引入

/loan-web/src/main/resources/templates/auditing/financeLoanAuditing.html

<script src=*"/js/jsxd/workflow/actTaskHistoricFlow.js"* type=*"text/javascript"* charset=*"utf-8"*></script>

<div id=*"histoicFlow"*></div>

/loan-workflow/src/main/resources/templates/workflow/\_actTaskHistoricFlow.html

2、进审批、详情页面时需要在上下文中设置流程实例ID

## /loan-web/src/main/java/com/lhhs/loan/web/controller/AuditProcessController.java

## 十二、相关命名规则和角色对应

### 部分命名规则

1流程分类：BA,OA

1. 模型名称：分类+名称（BA\_转储流程）
2. 流程标识：分类+流程标识（BA\_transfer\_store）
3. 节点标识：流程标识+act+数字（BA\_transfer\_store\_act1）
4. 流程代码中传参数：事业部（productdpt）产品线（productline）平台（branchcom）办公室（salesffice）部门（department）备注（activitiremark）

### 2.角色的编码和名称对应(loan\_quarters)

|  |  |  |
| --- | --- | --- |
| 编号 | 岗位名称 | 岗位编码 |
| 1 | 客户经理 | CustomerManager |
| 2 | 事业部总经理 | BusinessUnitGM |
| 3 | 风控实勘经理 | RiskRealManager |
| 4 | 区域风控经理 | AreaRiskManager |
| 5 | 风控总监 | RiskControlDirector |
| 6 | 资金部总监 | FundDepartmentDirector |
| 7 | 分公司总经理 | BranchOfficeGM |
| 8 | 区域总经理 | AreaManager |
| 9 | 审贷会 | Approval |
| 10 | 权证专员 | WarrantCommissioner |
| 11 | 集团执行总裁 | GroupCEO |
| 12 | 结算经理 | SettlementManager |
| 13 | 结算专员 | SettlementSpecialist |
| 14 | 权证总监 | WarrantDirector |
| 15 | 出纳 | Cashier |
| 16 | 风控实勘专员 | RiskRealAssistant |
| 17 | 会计 | Accountant |
| 18 | 财务经理 | FinancialManager |
| 19 | 财务专员 | FinancialCommissioner |
| 20 | 风控专员 | RiskControlSpecialist |
| 21 | 集团风控经理 | GroupRiskControlManager |
| 22 | 团队总监 | TeamDirector |
| 23 | 执行董事 | ExecutiveDirector |
| 24 | 风控总经理 | RiskUnitManager |
| 25 | 风控经理 | RiskManager |
| 26 | 执行副总裁 | ExecutiveVicePresident |
| 27 | 资金部专员 | FundDepCommissioner |

### 3、流程变量公共设置KEY命名常量内部类com.thunis.workflow.common.utils.Constant.Field

事例：业务编号KEY常量：Constant.Field.BUSINESSID

***具体如下：***

**public** **class** Field{

/\*\*系统标识\*\*/

**public** **static** **final** String ***SYS***="sys";

/\*\*渠道标识\*\*/

**public** **static** **final** String ***CHANNEL***="channel";

/\*\*任务申请人\*\*/

**public** **static** **final** String ***APPLY***="apply";

/\*\*多实例计数器\*\*/

**public** **static** **final** String ***YESCOUNTER***="yescounter";

/\*\*多实例用户集合\*\*/

**public** **static** **final** String ***ASSIGNEELIST***="assigneelist";

/\*\*多实例用户集合数\*\*/

**public** **static** **final** String ***ASSIGNEECOUNTER***="assigneecounter";

/\*\*实例总数\*\*/

**public** **static** **final** String ***NROFINSTANCES***="nrOfInstances";

/\*\*任务标题\*\*/

**public** **static** **final** String ***TITLE***="title";

/\*\*流程实例ID\*\*/

**public** **static** **final** String ***PROCINSID***="procInsId";

/\*\*处理结果返回编码：成功true，失败false\*\*/

**public** **static** **final** String ***CODE***="code";

/\*\*处理结果错误提示信息\*\*/

**public** **static** **final** String ***MSG***="msg";

/\*\*业务流水号\*\*/

**public** **static** **final** String ***BUSINESSID***="businessId";

/\*\*子流程建立集合\*\*/

**public** **static** **final** String ***SUBLIST***="sublist";

/\*\*子流程建立集合变量\*\*/

**public** **static** **final** String ***SUBPARM***="subparm";

/\*\*子流程建立流程变量KEY\*\*/

**public** **static** **final** String ***SUBKEY***="subkey\_";

/\*\*子流程建立任务和流程流程变量对应关系\*\*/

**public** **static** **final** String ***SUBTASK***="subtask\_";

/\*\*多实例子任务流程变量KEY\*\*/

**public** **static** **final** String ***SUBTEMP***="\_sub\_temp";

/\*\*备注\*\*/

**public** **static** **final** String ***REMARK***="remark";

/\*\*合作伙伴\*\*/

**public** **static** **final** String ***PARTNER***="partner";

/\*\*金额\*\*/

**public** **static** **final** String ***AMOUNT***="amount";

/\*\*订单编号\*\*/

**public** **static** **final** String ***ORDERNO***="orderNo";

/\*\*部门\*\*/

**public** **static** **final** String ***DEPARTMENT***="department";

/\*\*部门名称\*\*/

**public** **static** **final** String ***DEPARTMENTNAME***="departmentName";

/\*\*平台\*\*/

**public** **static** **final** String ***BRANCHCOM***="branchcom";

/\*\*销售办公室\*\*/

**public** **static** **final** String ***SALESOFFICE***="salesoffice";

/\*\*事业部\*\*/

**public** **static** **final** String ***PRODUCTDPT***="productdpt";

/\*\*产品线\*\*/

**public** **static** **final** String ***PRODUCTLINE***="productline";

/\*\*预留字段\*\*/

**public** **static** **final** String ***FIELD1***="field1";

/\*\*预留字段\*\*/

**public** **static** **final** String ***FIELD2***="field2";

/\*\*预留字段\*\*/

**public** **static** **final** String ***FIELD3***="field3";

/\*\*预留字段\*\*/

**public** **static** **final** String ***FIELD4***="field4";

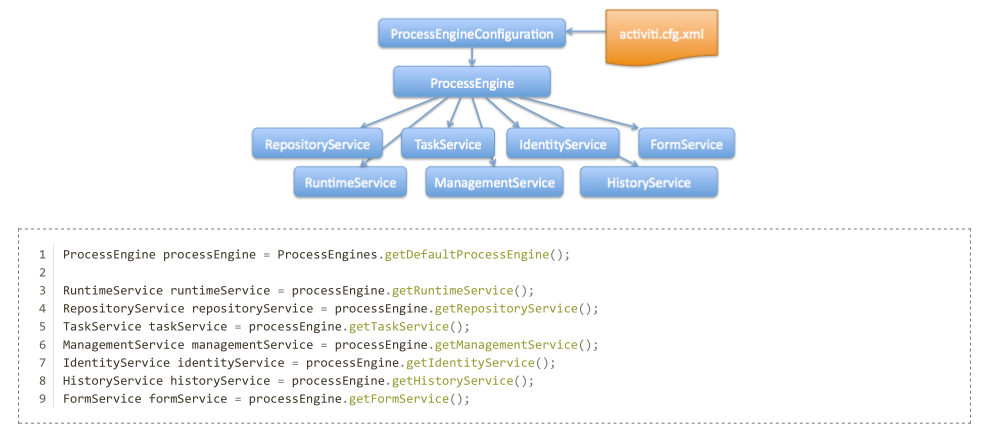
}

## 十三、流程引擎API与服务

引擎API是与Activiti交互的最常用手段。中心入口是ProcessEngine，可以使用多种方式创建。使用

ProcessEngine，可以获得包含工作流/BPM方法的多种服务。ProcessEngine与服务对象都是线程安全的，因此可以在整个服务器中保

存一份引用。



ProcessEngines.getDefaultProcessEngine()在第一次被调用时将初始化并构建流程引擎，在之后的调用都会返回相同的流程引

擎。流程引擎的创建通过ProcessEngines.init()实现，关闭由ProcessEngines.destroy()实现。

ProcessEngines会扫描所有activiti.cfg.xml与activiti‐context.xml文件。对于所有的activiti.cfg.xml文件，流程引擎会

以标准Activiti方式构

建：ProcessEngineConfiguration.createProcessEngineConfigurationFromInputStream(inputStream).buildProcessEngine()。

对于所有的activiti‐context.xml文件，流程引擎会以Spring的方式构建：首先构建Spring应用上下文，然后从该上下文中获取流程

引擎。

所有的服务都是无状态的。这意味着你可以很容易的在集群环境的多个节点上运行Activiti，使用同一个数据库，而不用担心上一次调用

实际在哪台机器上执行。不论在哪里执行，对任何服务的任何调用都是幂等（idempotent）的。

RepositoryService很可能是使用Activiti引擎要用的第一个服务。这个服务提供了管理与控制deployments（部署）与process

definitions（流程定义）的操作。在这里简单说明一下，流程定义是BPMN 2.0流程的Java等价副本，展现流程中每一步的结构与行

为。deployment是 Activiti引擎中的包装单元，一个部署中可以包含多个BPMN 2.0 xml文件，以及其他资源。开发者可以决定在一个部

署中包含的内容，可以是单各流程的BPMN 2.0 xml文件，也可以包含多个流程及其相关资源（如’hr­processes’部署可以包含所有与人

力资源流程相关的的东西）。RepositoryService可用于deploy（部署）这样的包。部署意味着将它上传至引擎，引擎将在储存至数

据库之前检查与分析所有的流程。从这里开始，系统知道了这个部署，部署中包含的所有流程都可以启动。

此外，这个服务还可以：

查询引擎已知的部署与流程定义。

暂停或激活部署中的某些流程，或整个部署。暂停意味着不能再对它进行操作，激活是其反操作。

读取各种资源，比如部署中保存的文件，或者引擎自动生成的流程图。

读取POJO版本的流程定义。使用它可以用Java而不是xml的方式检查流程。

RepositoryService提供的是静态信息（也就是不会改变，至少不会经常改变的信息），而RuntimeService就完全相反。它可以启动

流程定义的新流程实例。前面介绍过，process definition（流程定义）定义了流程中不同步骤的结构与行为。流程实例则是流程定

义的实际执行。同一时刻，一个流程定义通常有多个运行中的实例。RuntimeService也用于读取与存储process variables（流程

变量）。流程变量是给定流程持有的数据，可以在流程的许多构造中使用（例如排他网关exclusive gateway 经常使用流程变量决定流程

下一步要选择的路径）。RuntimeService还可以用于查询流程实例与执行（execution）。执行代表了BPMN 2.0中的 'token' 概

念。通常执行是指向流程实例当前位置的指针。最后，RuntimeService还可以在流程实例等待外部触发时使用，以便流程可以继续运

服务提供了许多操作用于“通知”流程实例，告知已经接收到外部触发，

行。流程有许多wait states（暂停状态），RuntimeService服务提供了许多操作用于“通知”流程实例，告知已经接收到外部触发，

使流程实例可以继续运行。

对于像Activiti这样的BPM引擎来说，核心是需要人类用户实际操作的任务。所有任务相关的东西都组织在TaskService中，例如

查询分派给用户或组的任务

创建standalone（独立运行）任务。这是一种没有关联到流程实例的任务。

决定任务的执行用户（assignee），或者将用户通过某种方式与任务关联。

认领（claim）与完成（complete）任务。认领是指某人决定成为任务的执行用户，也即他将会完成这个任务。完成任务是指“做这个

任务要求的工作”，通常是填写某种表单。

IdentityService很简单。它用于管理（创建，更新，删除，查询……）组与用户。请重点注 意，Activiti实际上在运行时并不做任何用户

检查。例如任务可以分派给任何用户，而引擎并不会验证系统中是否存在该用户。这是因为Activiti 有时要与LDAP、Active Directory等

服务结合使用。

FormService是可选服务。也就是说Activiti没有它也能很好地运行，而不必牺牲任何功能。这个服务引入了start form（开始表单）与

task form（任务表单）的概念。 开始表单是在流程实例启动前显示的表单，而任务表单是用户完成任务时显示的表单。Activiti可以在

BPMN 2.0流程定义中定义这些表单。表单服务通过简单的方式暴露这些数据。再次重申，表单不一定要嵌入流程定义，因此这个服务

是可选的。

HistoryService暴露所有Activiti引擎收集的历史数据。当执行流程时，引擎会保存许多数据（可以配置），例如流程实例启动时间，谁

在执行哪个任务，完成任务花费的事件，每个流程实例的执行路径，等等。这个服务主要提供查询这些数据的能力。

ManagementService通常在用Activiti编写用户应用时不需要使用。它可以用于读取数据库 表与表原始数据的信息，也提供了对作业

（job）的查询与管理操作。Activiti中很多地方都使用作业，例如定时器（timer），异步操作 （asynchronous continuation），延时暂

停/激活（delayed suspension/activation）等等。

## 十四、activiti数据库表结构说明

1.1逻辑结构

Activiti使用到的表都是ACT\_开头的。

ACT\_RE\_\*:

’RE’表示repository(存储)，RepositoryService接口所操作的表。带此前缀的表包含的是静态信息，如，流程定义，流程的资源（图片，规则等）。

ACT\_RU\_\*:

‘RU’表示runtime，运行时表-RuntimeService。这是运行时的表存储着流程变量，用户任务，变量，职责（job）等运行时的数据。Activiti只存储实例执行期间的运行时数据，当流程实例结束时，将删除这些记录。这就保证了这些运行时的表小且快。

ACT\_ID\_\*:

’ID’表示identity (组织机构)，IdentityService接口所操作的表。用户记录，流程中使用到的用户和组。这些表包含标识的信息，如用户，用户组，等等。

ACT\_HI\_\*:’HI’表示history，历史数据表，HistoryService。就是这些表包含着流程执行的历史相关数据，如结束的流程实例，变量，任务，等等

ACT\_GE\_\*:

全局通用数据及设置(general)，各种情况都使用的数据。

1.2、   表的含义

|  |  |  |
| --- | --- | --- |
| 序号 | 表名 | 说明 |
| 1 | act\_ge\_bytearray | 二进制数据表 |
| 2 | act\_ge\_property | 属性数据表存储整个流程引擎级别的数据,初始化表结构时，会默认插入三条记录， |
| 3 | act\_hi\_actinst | 历史节点表 |
| 4 | act\_hi\_attachment | 历史附件表 |
| 5 | act\_hi\_comment | 历史意见表 |
| 6 | act\_hi\_identitylink | 历史流程人员表 |
| 7 | act\_hi\_detail | 历史详情表，提供历史变量的查询 |
| 8 | act\_hi\_procinst | 历史流程实例表 |
| 9 | act\_hi\_taskinst | 历史任务实例表 |
| 10 | act\_hi\_varinst | 历史变量表 |
| 11 | act\_id\_group | 用户组信息表 |
| 12 | act\_id\_info | 用户扩展信息表 |
| 13 | act\_id\_membership | 用户与用户组对应信息表 |
| 14 | act\_id\_user | 用户信息表 |
| 15. | act\_re\_deployment | 部署信息表 |
| 16. | act\_re\_model | 流程设计模型部署表 |
| 17 | act\_re\_procdef | 流程定义数据表 |
| 18 | act\_ru\_event\_subscr | ？ |
| 19 | act\_ru\_execution | 运行时流程执行实例表 |
| 20 | act\_ru\_identitylink | 运行时流程人员表，主要存储任务节点与参与者的相关信息 |
| 21 | act\_ru\_job | 运行时定时任务数据表 |
| 22 | act\_ru\_task | 运行时任务节点表 |
| 23 | act\_ru\_variable | 运行时流程变量数据表 |

工作流全局监听

/loan-workflow/src/main/java/com/lhhs/workflow/interceptor/ActivitiGlobalEventListener.java

启动流程、处理流程、代办、已办，任务签收

/loan-workflow/src/main/java/com/lhhs/workflow/bs/impl/ActTaskServiceImpl.java

/loan-workflow/src/main/java/com/lhhs/workflow/common/enumeration/Resolution.java

/loan-workflow/src/main/java/com/lhhs/workflow/sys/utils/bs/UserUtils.java

权限对接Mapper配置文件：

/loan-workflow/src/main/java/com/lhhs/workflow/dao/mapper/UserPermissionMapper.xml

审批意见Mapper配置文件：

/loan-workflow/src/main/java/com/lhhs/workflow/dao/mapper/ActCommentMapper.xml

小贷审核实例：

/loan-web/src/main/java/com/lhhs/loan/web/controller/AuditProcessController.java

/loan-service/src/main/java/com/lhhs/loan/service/audit/impl/AuditProcessServiceImpl.java