# ECMP平台工作流使用说明

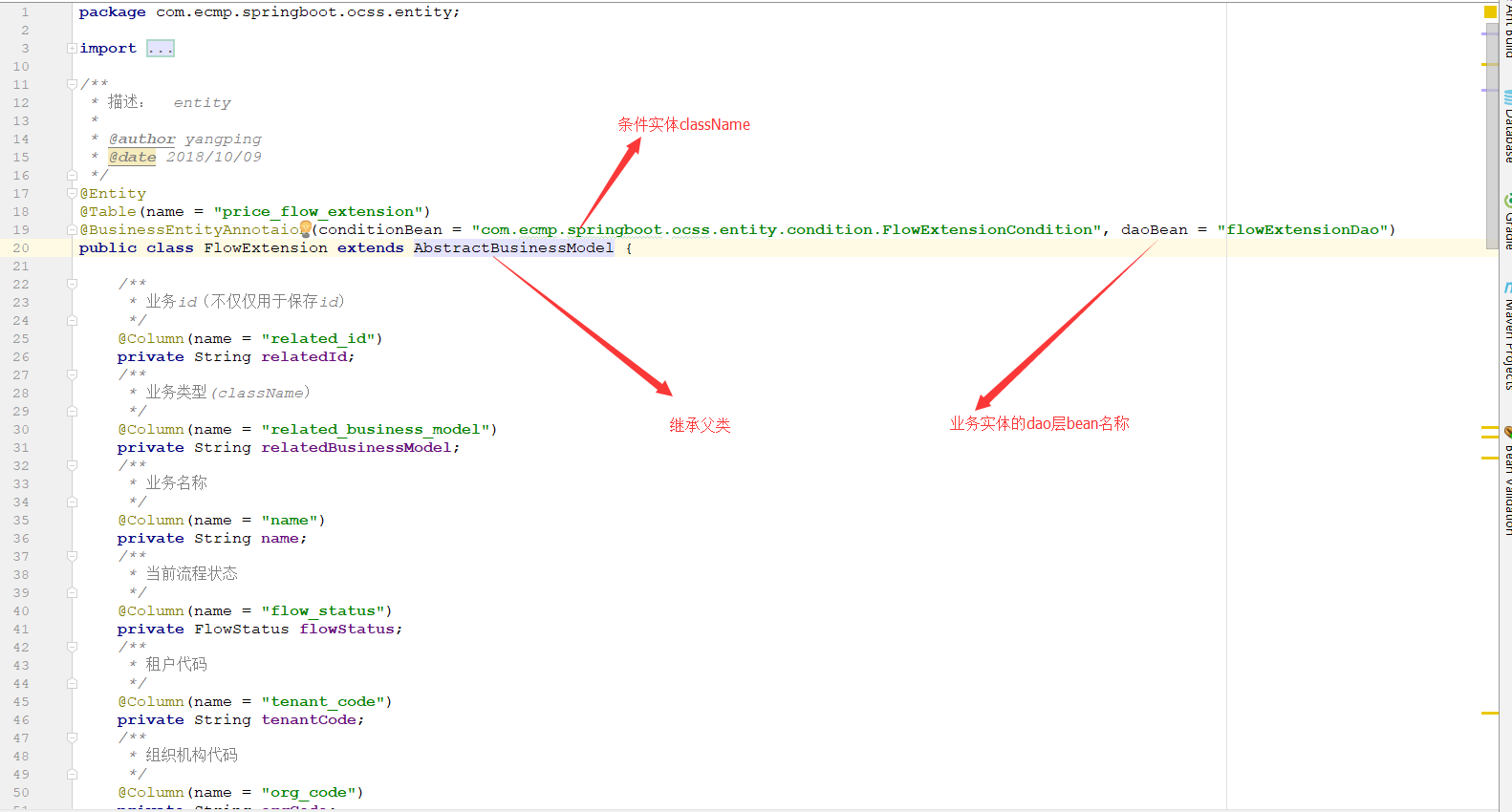
## 后端代码实现

### 依赖JAR包

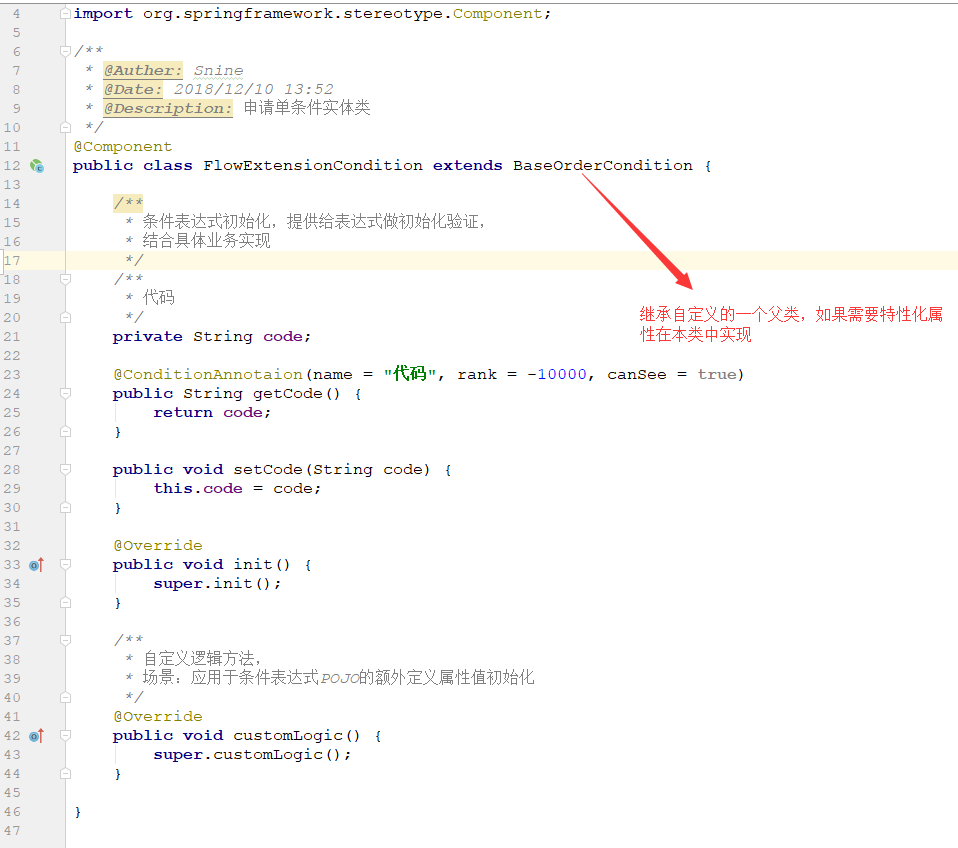
compile(**"com.ecmp:flow-service-client:**3.+**"**)

### 业务实体

中泰营销平台工作流业务实体只有一个即FlowExtension，通过该类关联真正的业务实体，示例：



### 条件实体

条件实体主要是提供与流程业务实体相关的一些属性，可以用作控制流程走向或者其他

### 提供接口

需要提供的接口有两类：1、通用接口，如条件实体属性获取；2、专有接口，如各节点的回调事件。

#### 通用接口

通用接口直接继承父类CommonConditionService，调用父类方法即可。

|  |  |  |
| --- | --- | --- |
| 方法 | 方法名 | 备注 |
| properties | 通过业务实体代码获取条件POJO属性说明 | 获取条件实体的字段以及注解中的中文名称 |
| initPropertiesAndValues | 通过业务实体代码获取条件POJO属性初始化值键值对 | 调用条件实体中的初始化方法并返回各属性值 |
| propertiesAndValues | 通过业务实体代码,业务ID获取条件POJO属性键值对 | 克隆业务实体属性值到条件实体并返回条件实体的属性值 |
| resetState | 通过业务实体代码及单据ID重置业务单据流程状态 | 启动流程、流程结束、流程废弃状态的修改 |
| businessPropertiesAndValues | 通过业务实体代码,业务ID获取POJO属性键值对 | formPropertiesAndValues  获取业务实体中有 Remark注解的属性值 |

#### 专有接口

专有接口一般都是工作流节点配置的事件

|  |  |  |
| --- | --- | --- |
| 入参 | 返回值 | 备注 |
| FlowInvokeParams | FlowOperateResult |  |

### 消费接口

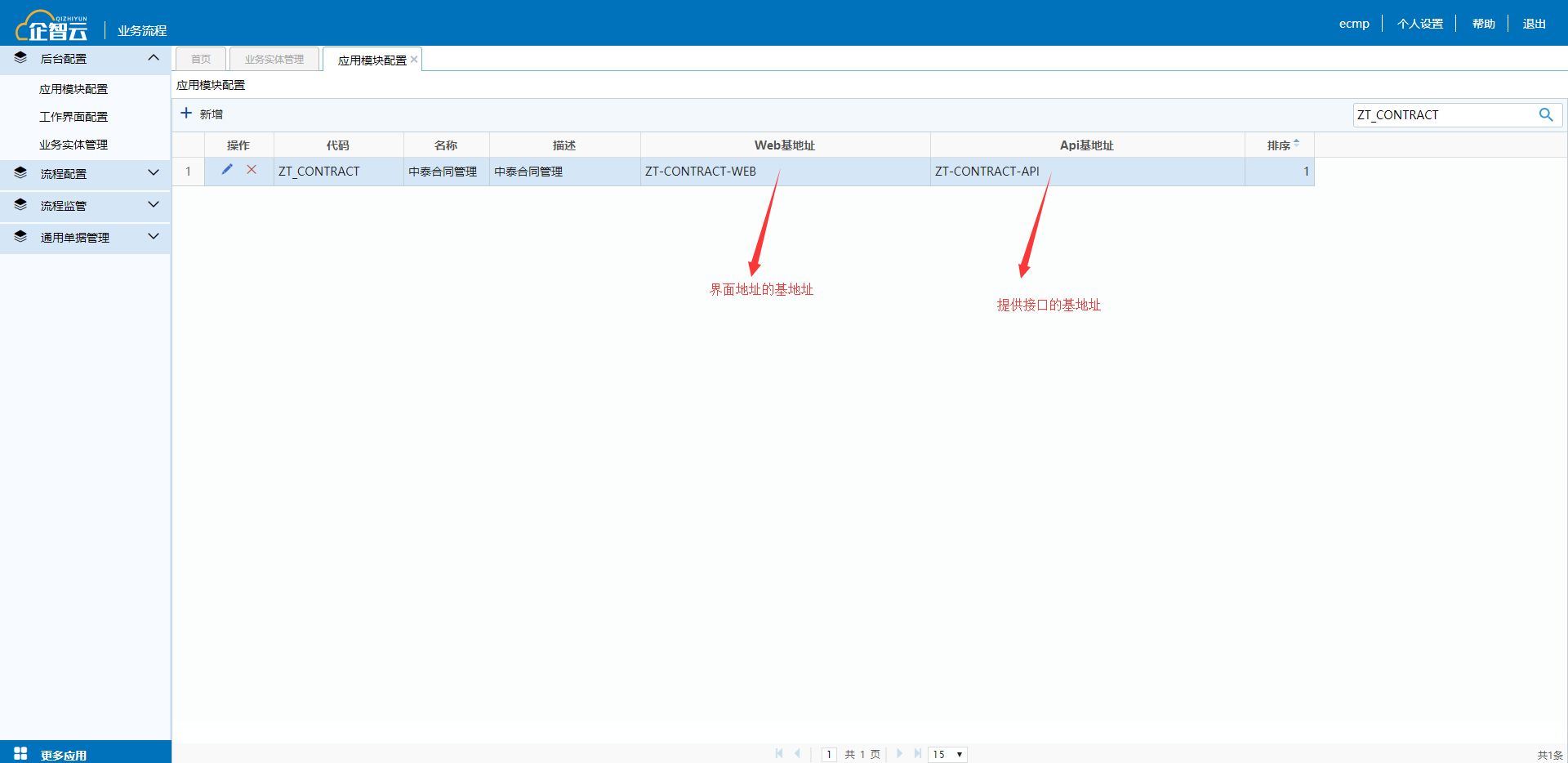
主动调用工作流的接口IDefaultFlowBaseService，如启动流程、审批任务等。

|  |  |  |
| --- | --- | --- |
| 方法 | 方法名 | 备注 |
| startFlow | 通过流程定义key启动流程 |  |
| claimTask | 签收任务 |  |
| completeTask | 完成任务 |  |
| rollBackTo | 回退（撤销）任务 |  |
| rejectTask | 任务驳回 |  |
| nextNodesInfo | 获取当前审批任务的决策信息 |  |
| getSelectedNodesInfo | 获取下一步的节点信息任务 |  |
| nextNodesInfoWithUser | 获取下一步的节点信息任务 |  |
| getApprovalHeaderInfo | 获取任务抬头信息信息任务 |  |
| findTasksByBusinessId | 通过业务单据Id获取待办任务 |  |

## ECMP配置

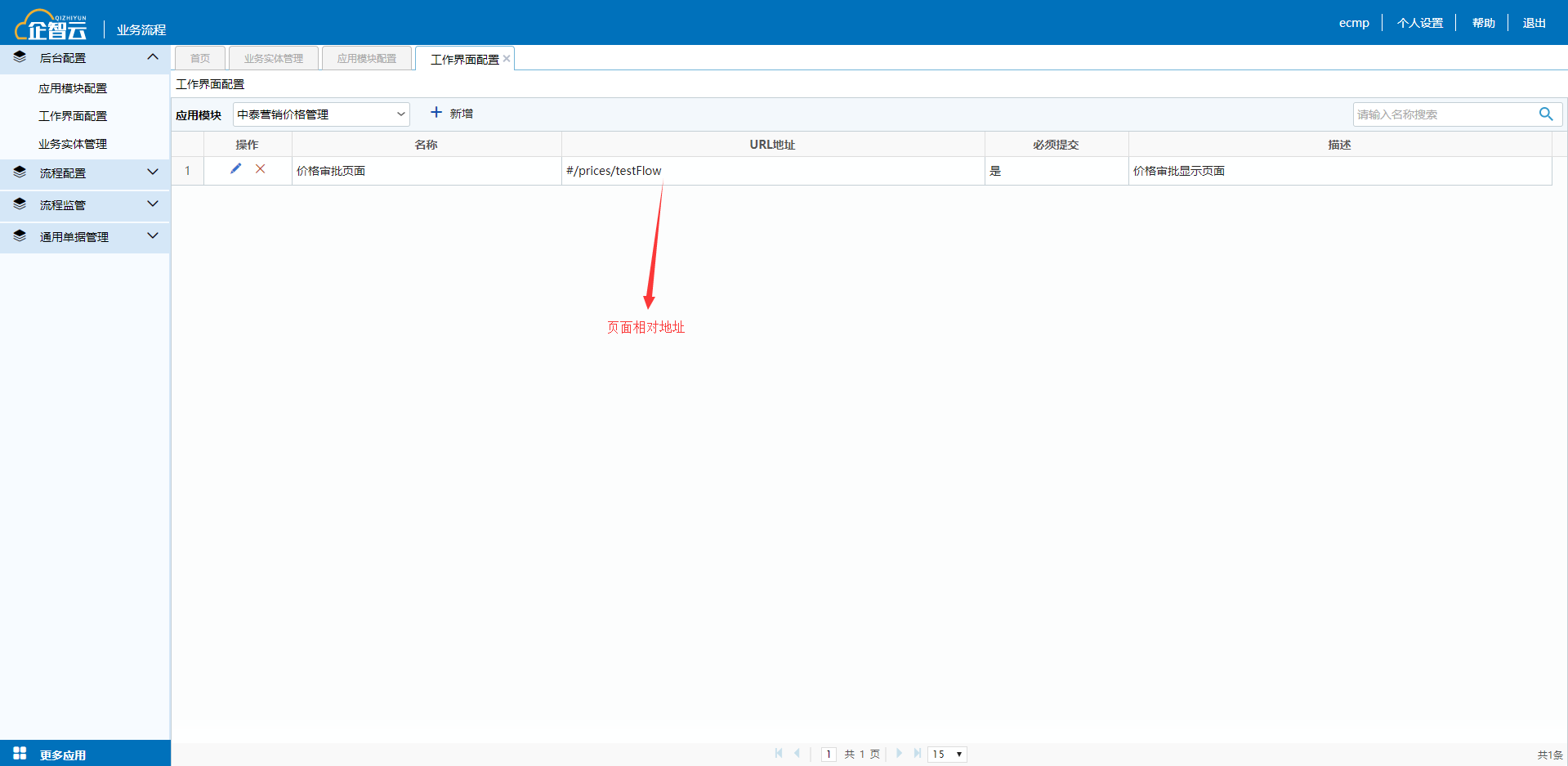
### 后台配置

#### 应用模块配置



#### 工作界面配置

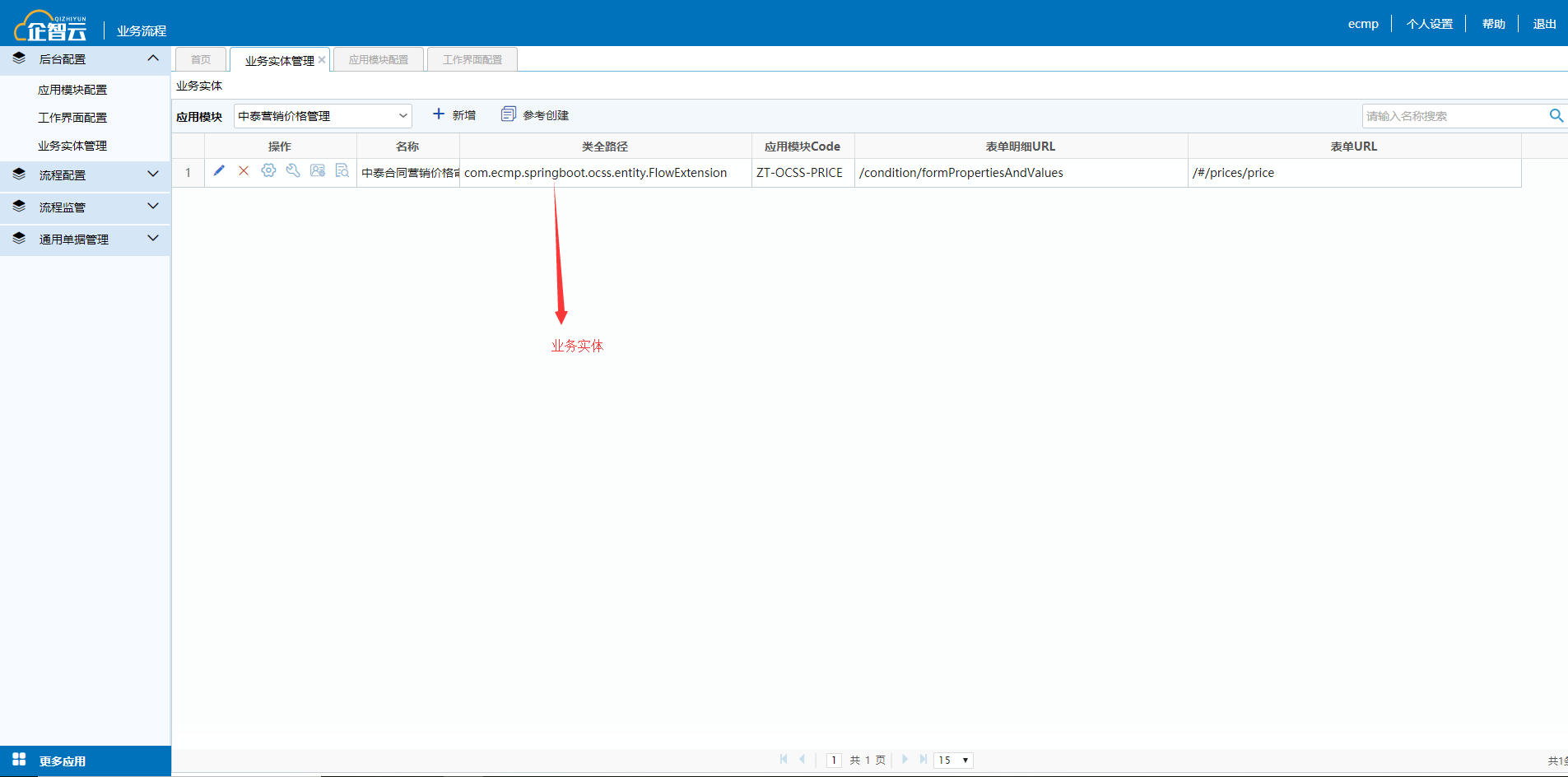
**注意：工作界面选择的应用模块来自【业务流程】->【后台配置】->【应用模块配置】**

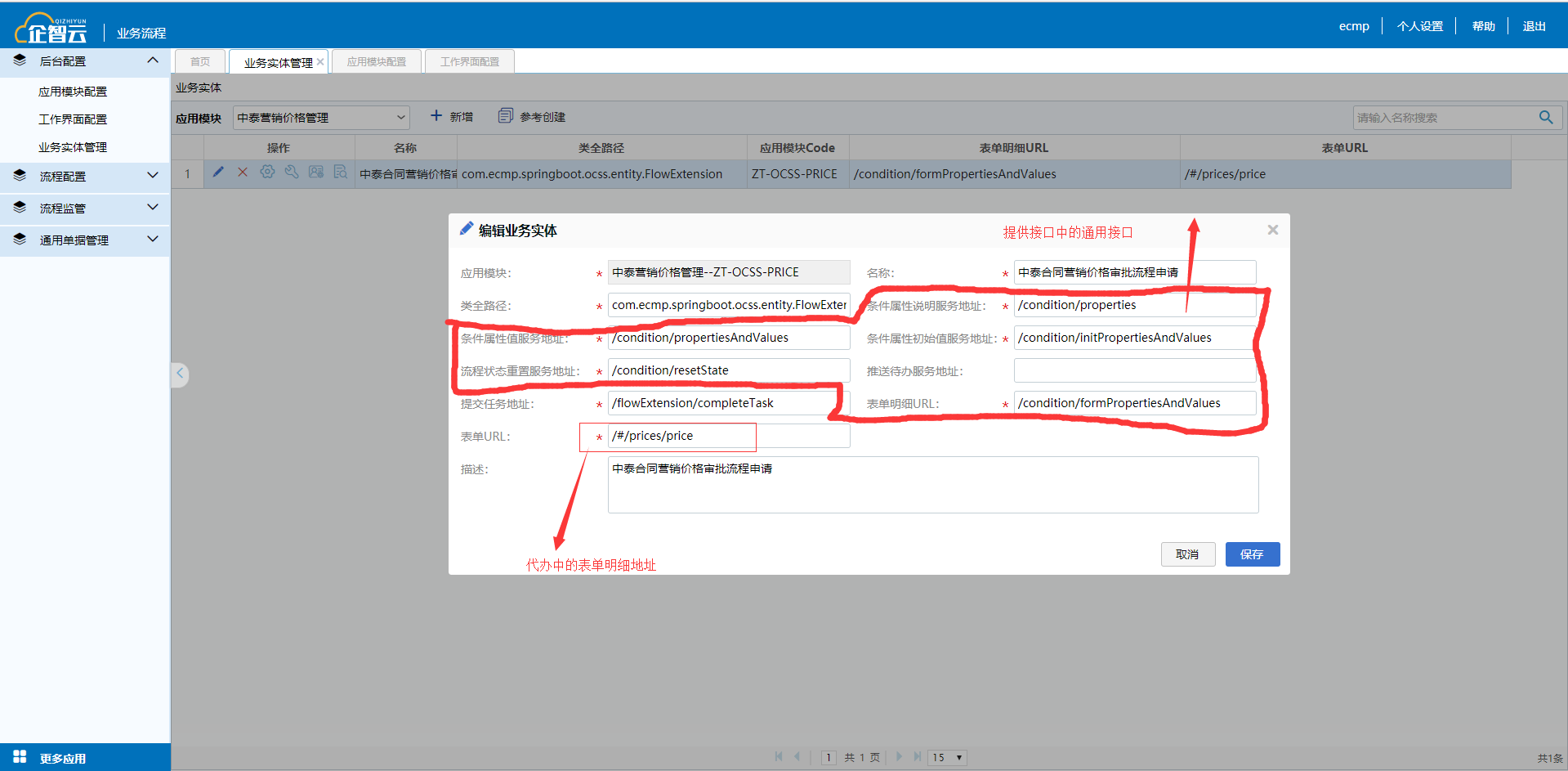
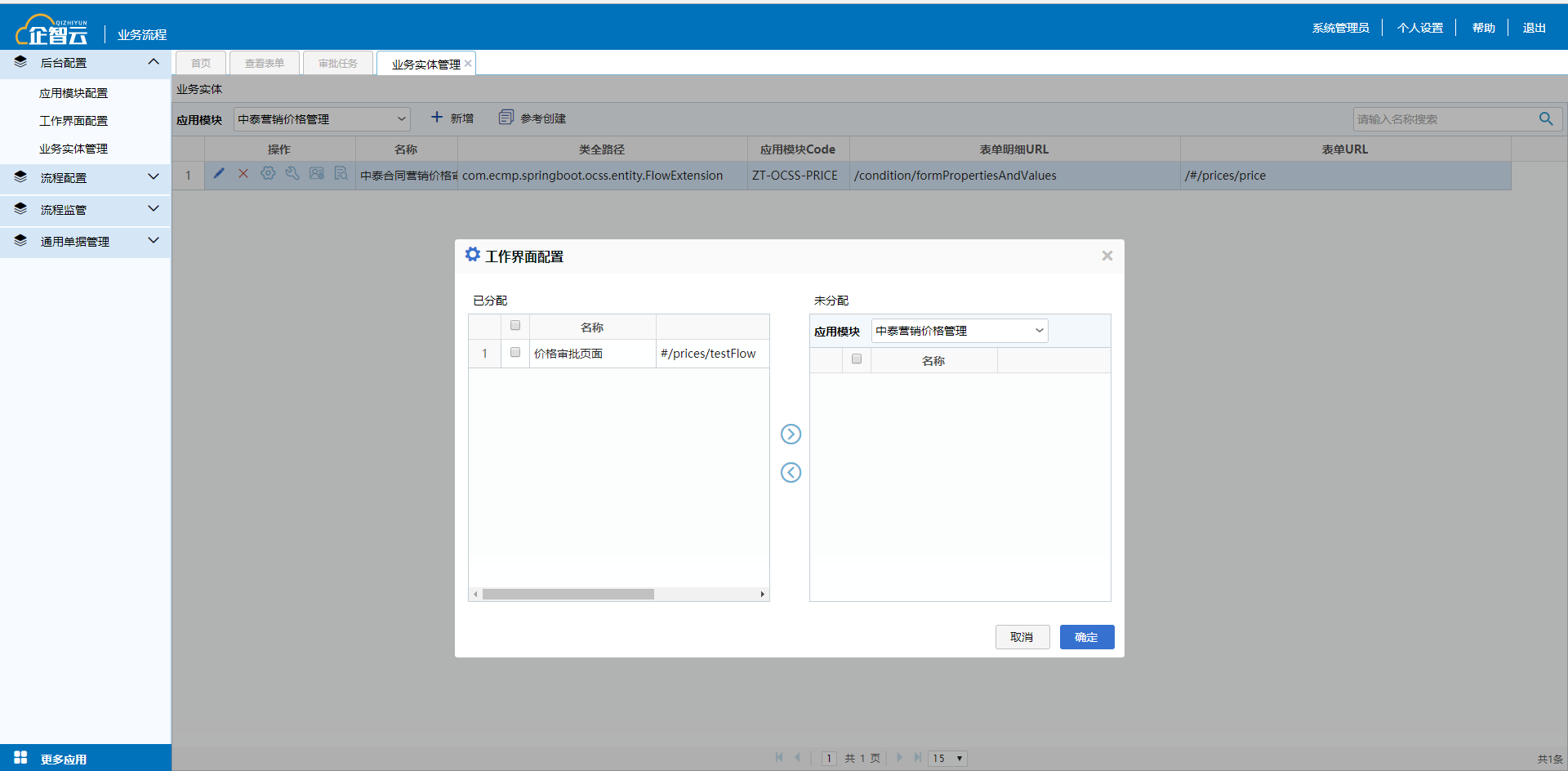
****

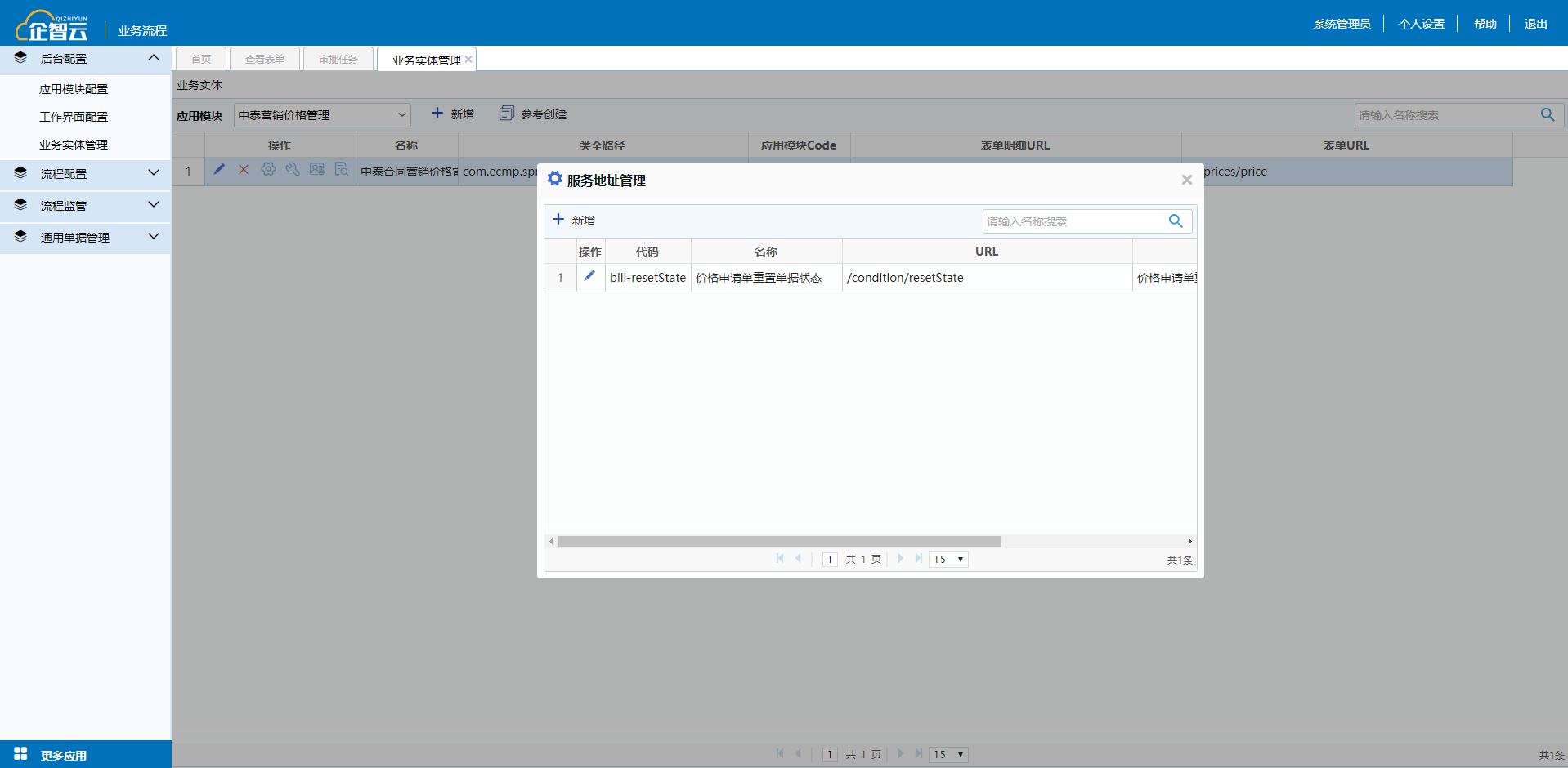
#### 业务实体管理

**注意：业务实体管理选择的应用模块来自【基础应用】->【后台配置】->【业务模块配置】**

**两个地方的应用模块保持一致；基础应用中创建了对应的业务模块后还要在租户中给对应的租户分配权限。**

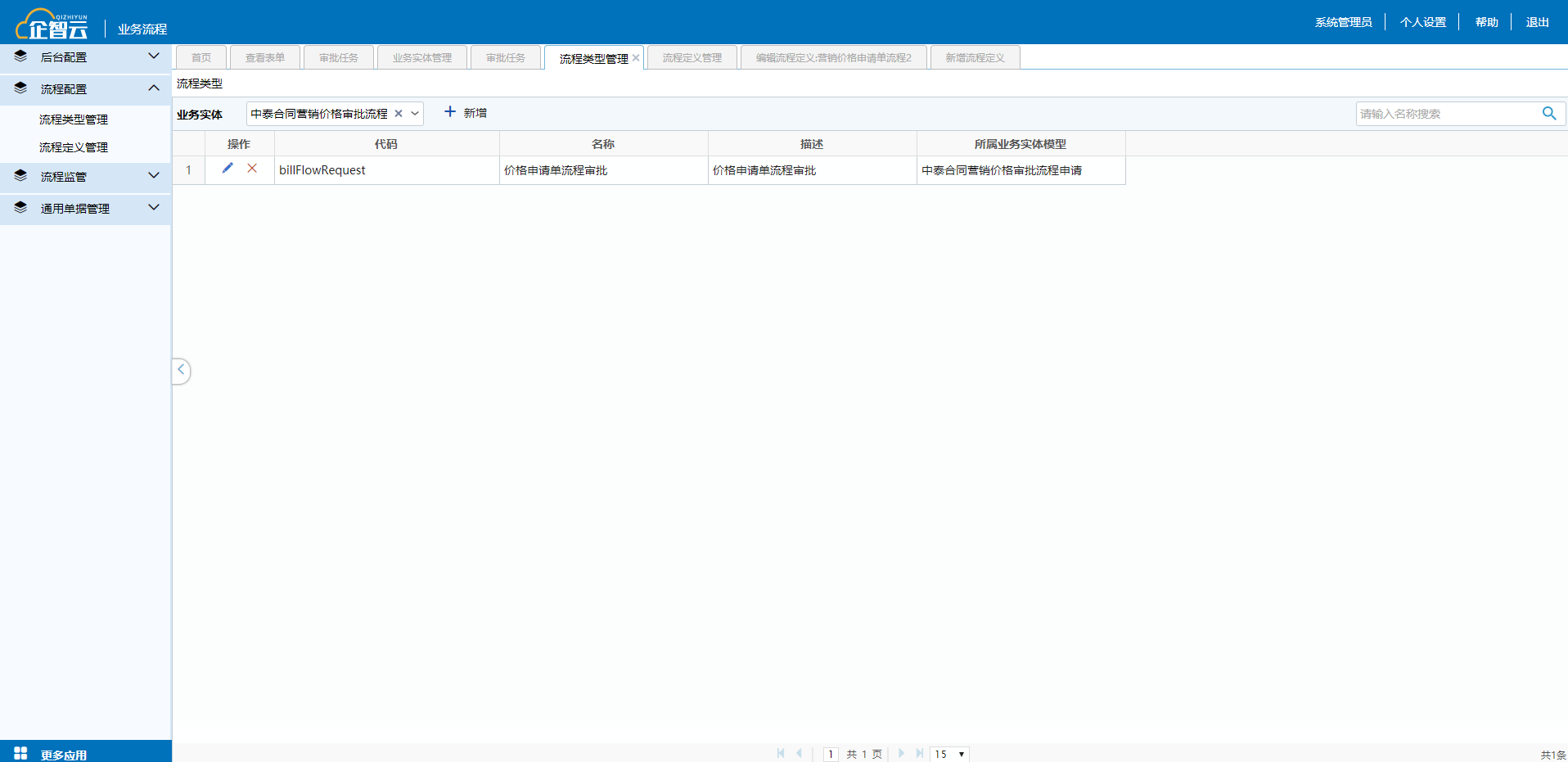
包含：编辑、页面、事件回调的服务器地址



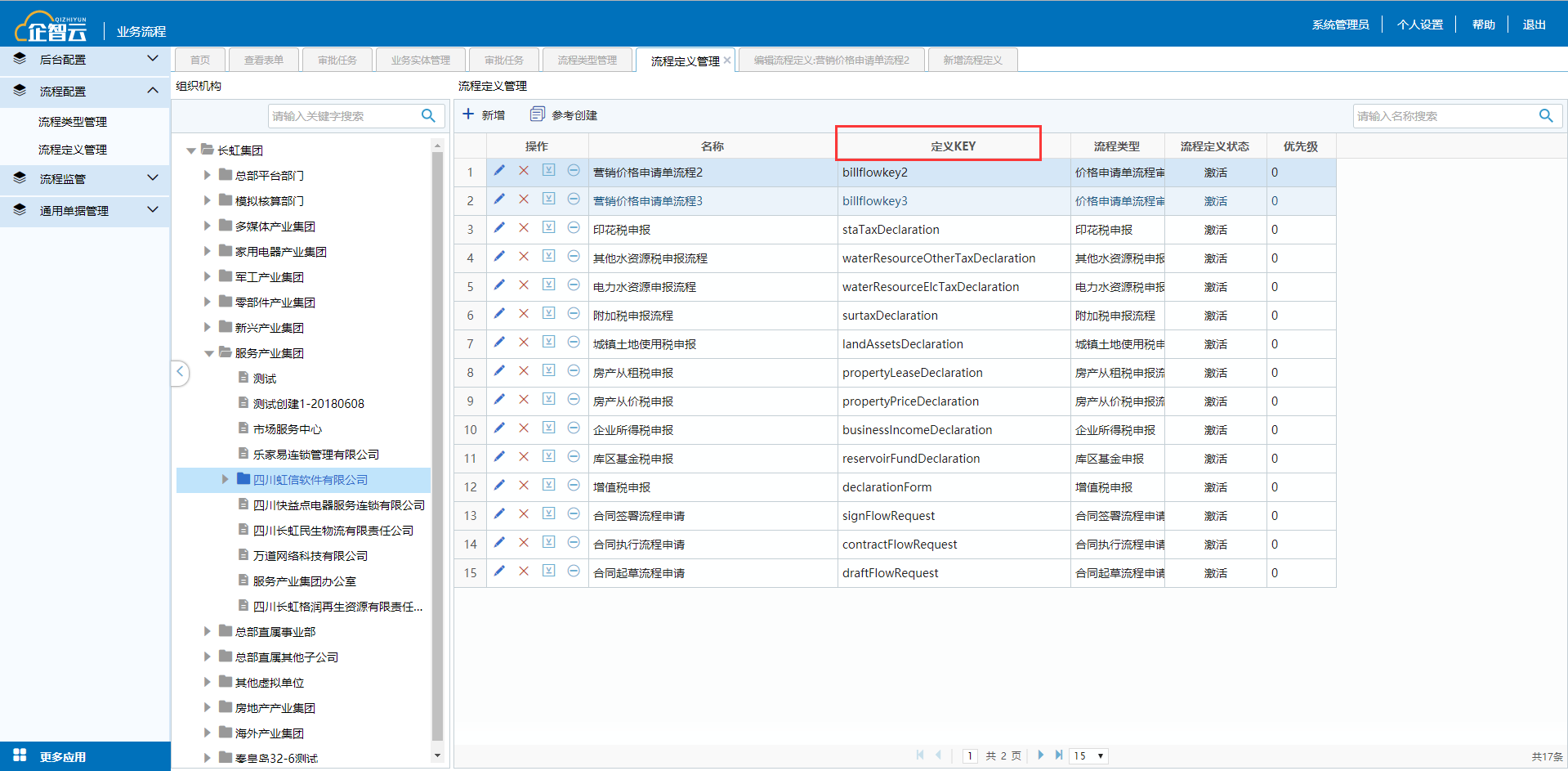


### 流程配置

#### 流程类型管理

流程的定义-》**流程类型**-》流程实体

#### 流程定义管理

流程定义即流程图，一个流程实体可以对应多个流程定义，启动的时候通过指定key

## 任务处理

### 待办列表

## 调试方法

### 本地与Flow项目联调

Flow项目中的application.properties 文件修改：

1. 应用和zk地址修改为成都开发部的flow项目

**ECMP\_APP\_ID**=**80F02DA4-0C39-6769-4954-ADCDF6AD6CFC**

**ECMP\_CONFIG\_CENTER**=**10.4.208.83:2181**

1. 指定端口号

**server.port**=**80**

1. 指定flow-api地址

**FLOW\_API**=**http://localhost:80**

业务项目中的application.properties文件修改：

1、添加flow-api地址配置

**FLOW\_API**=**http://localhost:80**

### 测试用例

**private static** String *businessModelCode* =OrderFlowExtension.**class**.getName();  
**private static** String *businessKey* = **"397B1FE0-1002-11E9-B442-B88198088734"**;  
**private static** String *typeId* = **"7638F8F3-0FD0-11E9-A9DF-0242C0A84415"**;  
**private static** String *flowDefKey* = **"orderFlowKey2"**;  
**private static** String *orgId* = **"33D858F3-F3A5-11E8-93E6-0242C0A8440A"**;

*/\*\*  
 \* 创建单据  
 \*/*@Test  
**public void** insert() **throws** NoSuchMethodException {  
 OrderFlowExtension flowExtension = **new** OrderFlowExtension();  
 flowExtension.setRelatedId(UUID.*randomUUID*().toString());  
 flowExtension.setRelatedBusinessModel(So.**class**.getName());  
 flowExtension.setName(**"营销订单流程测试14"**);  
 flowExtension.setWorkCaption(**"这仅仅是一个流程测试！"**);  
 flowExtension.setOrgId(*orgId*);  
 flowExtension.setBusinessCode(**"O2019010405"**);  
  
 **orderFlowExtensionDao**.save(flowExtension);  
 System.***out***.println(**"flowExtension.getId() = "** + flowExtension.getId());  
}  
  
*/\*\*  
 \* 启动流程  
 \*  
 \** ***@throws*** *NoSuchMethodException  
 \*/*@Test  
**public void** startFlow() **throws** NoSuchMethodException {  
 *//流程启动前获取流程节点信息* IDefaultFlowBaseService service = ApiClient.*createProxy*(IDefaultFlowBaseService.**class**);  
 ResponseData responseData = service.startFlow(*businessModelCode*,  
 *businessKey*,  
 **null**,  
 *typeId*,  
 **null**,  
 **null**,  
 **null**);  
 System.***out***.println(**"responseData = "** + JSON.*toJSONString*(responseData));  
  
 *//获取taskList* JSONArray array = **new** JSONArray();  
 JSONObject jsonObject = JSON.*parseObject*(JSON.*toJSONString*(responseData.getData()));  
 JSONArray taskArray = jsonObject.getJSONArray(**"nodeInfoList"**);  
 JSONObject taskJson = JSON.*parseObject*(JSON.*toJSONString*(taskArray.get(0)));  
 taskJson.put(**"nodeId"**,taskJson.get(**"id"**));  
 taskJson.put(**"instancyStatus"**,**false**);  
 taskJson.remove(**"executorSet"**);  
 taskJson.put(**"userIds"**, ContextUtil.*getUserId*());  
 array.add(taskJson);  
 System.***out***.println(**"array = "** + array.toJSONString());  
  
 *//启动流程参数* String taskList = array.toJSONString();  
 String opinion = **""**;  
 String anonymousNodeId = **null**;  
 ResponseData result = service.startFlow(*businessModelCode*,  
 *businessKey*,  
 opinion,  
 *typeId*,  
 *flowDefKey*,  
 taskList,  
 anonymousNodeId);  
 System.***out***.println(**"responseData = "** + JSON.*toJSONString*(result));  
  
}  
  
*/\*\*  
 \* 处理任务  
 \*  
 \** ***@throws*** *Exception  
 \*/*@Test  
**public void** completeTask() **throws** Exception {  
 *//审批意见* String opinion = **"同意"**;  
 *//审批结果* String approved = **"true"**;  
  
 IDefaultFlowBaseService service = ApiClient.*createProxy*(IDefaultFlowBaseService.**class**);  
  
 ResponseData taskInfo = service.findTasksByBusinessId(*businessKey*);  
 System.***out***.println(**"taskInfo = "** + JSON.*toJSONString*(taskInfo));  
 JSONArray taskInfoArray = JSON.*parseArray*(JSON.*toJSONString*(taskInfo.getData()));  
 JSONObject task = taskInfoArray.getJSONObject(0);  
 String taskId = task.getString(**"id"**);  
 System.***out***.println(**"taskId = "** + taskId);  
 *//单据信息* ResponseData approvalHeaderInfo = service.getApprovalHeaderInfo(taskId);  
 System.***out***.println(**"JSON.toJSONString(approvalHeaderInfo) = "** + JSON.*toJSONString*(approvalHeaderInfo));  
  
 *//当前节点信息* ResponseData nextNodesInfo = service.nextNodesInfo(taskId);  
 System.***out***.println(**"JSON.toJSONString(nextNodesInfo) = "** + JSON.*toJSONString*(nextNodesInfo));  
  
 *//获取下一个节点信息以及处理人* ResponseData selectedNodesInfo = service.getSelectedNodesInfo(taskId, approved, **""**);  
 System.***out***.println(**"JSON.toJSONString(selectedNodesInfo) = "** + JSON.*toJSONString*(selectedNodesInfo));  
  
 *//获取taskList* JSONArray array = **new** JSONArray();  
 **if** (!**"EndEvent"**.equals(selectedNodesInfo.getData())) {  
 JSONArray taskArray = JSONArray.*parseArray*(JSON.*toJSONString*(selectedNodesInfo.getData()));  
 JSONObject taskJson = JSON.*parseObject*(JSON.*toJSONString*(taskArray.get(0)));  
 taskJson.put(**"nodeId"**,taskJson.get(**"id"**));  
 taskJson.put(**"instancyStatus"**,**false**);  
 taskJson.remove(**"executorSet"**);  
 taskJson.put(**"userIds"**, ContextUtil.*getUserId*());  
 array.add(taskJson);  
 }  
 System.***out***.println(**"array = "** + array.toJSONString());  
  
 String taskList = JSONArray.*toJSONString*(array);  
 ResponseData result = service.completeTask(taskId,  
 *businessKey*,  
 opinion,  
 taskList,  
 **""**,  
 **false**,  
 approved,  
 **null**);  
  
 System.***out***.println(JSON.*toJSONString*(result));  
  
}  
  
*/\*\*  
 \* 获取自定义执行人  
 \*/*@Test  
**public void** getExecutors(){  
 *//通过组织id获取员工集合* IEmployeeService employeeService = ApiClient.*createProxy*(IEmployeeService.**class**);  
 List<Employee> employeeList = employeeService.findByOrganizationId(*orgId*);  
  
 *//员工转执行人对象* **if**(CollectionUtils.*isNotEmpty*(employeeList)){  
 List<String> idList = employeeList.stream().map(BaseEntity::getId).collect(Collectors.*toList*());  
 **if**(CollectionUtils.*isNotEmpty*(idList)){  
 List<Executor> result = employeeService.getExecutorsByEmployeeIds(idList);  
 System.***out***.println(**"result = "** + JSON.*toJSONString*(result));  
 }  
 }  
}

### 注意事项

1、启动流程是orgId是必须存在的，并且与当前登录人存在联系，因为组织接口有做租户过滤。