* 1. 工作流模块
     1. 增加工作流模型接口

**a.接口url:**

pub/admin/system/workflow/addWorkflow.do

**b.接口说明：**

创建一个工作流模型

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| addWorkflow | | Result | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| wfname | String | | 工作流名称 | |
| wftext | String | | 工作流描述 | |
| classid | String | | 栏目Id | |
| orgId | String | | 机构Id | |

**d.返回值示例与说明**

{

"result": { //结果集;

"flag": "0", //标志，0为成功，1为操作失败

"msg": "添加模型成功" //返回的信息；

}

}

**e.功能实现说明**

1. 产生一个Activiti的model对象。
2. 把wfname设置到对象name中，wfname,wftest, classid，orgId以json字符串的形式存到对象中的MetaInfo变量中，执行repositoryService接口的saveModel方法，传入model对象参数，就把model对象存入数据库
3. 创建一个json对象editorNode，editorNode对象放id，resourceId和stencilset这3个变量，id的值为canvas，resourceId的值为canvas，stencilset是一个json对象，里面放namespace变量，值为http://b3mn.org/stencilset/bpmn2.0#。在执行repositoryService接口的addModelEditorSource方法，传入Model对象的id和editorNode对象的字符化两个参数。
4. 创建Resultd对象，设置msg为添加模型成功，flag为0，并返回这个Result对象。
   * 1. 修改工作流模型基本信息接口

**a.接口url:**

pub/admin/system/workflow/updateWorkflow.do

**b.接口说明：**

修改已经创建的工作流模型的流程

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| updateWorkflow | | Result | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| wfname | String | | 工作流名称 | |
| wftext | String | | 工作流描述 | |
| classid | String | | 系统模型Id | |
| modelId | String | | 工作流模型Id | |
| orgId | String | | 机构Id | |

**d.返回值示例与说明**

{

"result": { //结果集;

"flag": "0", //标志，0为成功，1为操作失败

"msg": "修改模型成功" //返回的信息；

}

}

**e.功能实现说明**

1. 把参数wftext，classid，orgId放到一个json对象中。在把这个json对象字符化赋值到MetaInfo变量中。
2. 把wfname，modelId，MetaInfo这四个变量放到一个map中。
3. 执行对应dao层的接口的修改方法，参入map参数。
4. 创建Resultd对象，设置msg为修改模型成功，flag为0，并返回这个Result对象。
   * 1. 查询工作流模基本信息程接口

**a.接口url:**

pub/admin/system/workflow/ searchWorkflowinfo.do

**b.接口说明：**

查找工作流模型前查找该模型的数据

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| searchWorkflowinfo | | Map<String,Object> | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| modelId | String | | 工作流模型Id | |

**d.返回值示例与说明**

{

" sysModelId ": "HI9385938 ", //配置工作流模型Id

"description": "aa", //工作流描述

"wfname": "aaaa", //工作流名称

"myorder": "34", //顺序

“orgId”：“” //机构Id

}

**e.功能实现说明**

1. 根据modelId查到这个model对象。
2. 把model中的name,description, classid,，orgId数据放到map中，其中description, classid，orgId这三个数据存在model中的MetaInfo字段中。
3. 返回这个map。
   * 1. 查找工作流模型列表接口

**a.接口url:**

pub/admin/system/workflow/searchWorkflow.do

**b.接口说明：**

查询工作流模型列表，如果传modelName或orgId参数，用这个orgId 或modelName来模糊查找，如果不传，是对所有模型的搜索。

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| searchWorkflow | | infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| modelName | String | | 用模型名作为条件搜索 | |
| orgId | String | | 机构Id | |
| pageSize | Integer | | 每页显示的尺寸 | |
| nowPage | Integer | | 当前页 | |

**d.返回值示例与说明**

{

"page": { //分页信息;

"page": 2, //当前页;

"pages": 10, //总页数;

"lines": 100, //信息总条数;

"pageSize": 10, //每页显示信息数;

"pageStart": 1 //起始页;

},

"total": 8, //当前页信息条数;

"infos": [

{

"sysModelId ": "837423", //系统模型Id

"modelid": "15001", //模型的id

"addtime": "Fri Aug 05 21:15:06 CST 2016",//模型增加的时间

"wfname": "aaaa", //工作流模型的名称

“orgId”：“” //机构Id

},

{}

]

}

**e.功能实现说明**

1. 根据参数拼接sql语句，把得到的list数据放到infor对象中的infos字段。把得到数据的长度放到infor对象中的total字段中。
2. 根据参数拼接sql语句，把得到的counts数据放到infor对象中的page对象的lines变量中，在给page对象设置page,pages,pageSize,pageStart字段。
3. 最后返回infor对象。
   * 1. 删除工作流模型

**a.接口url:**

pub/admin/system/workflow/deleteWorkflow.do

**b.接口说明：**

把已经存在的工作流模型删除掉

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| deleteWorkflow | | Result | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| modelId | String | | 模型Id | |

**d.返回值示例与说明**

{

"result": { //结果集;

"flag": "0", //标志，0为成功，1为操作失败

"msg": "删除模型成功" //返回的信息；

}

}

**e.功能实现说明**

1. 执行repositoryService接口的deleteMode方法，传入modelId参数，就可以删除这个模型。
2. 创建Resultd对象，设置msg为删除模型成功，flag为0，并返回这个Result对象。
   * 1. 部署流程接口

**a.接口url:**

pub/admin/system/workflow/deploy.do

**b.接口说明：**

部署工作流模型

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| deploy | | Result | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| modelId | String | | 模型Id | |

**d.返回值示例与说明**

{

"result": { //结果集;

"flag": "0", //标志，0为成功，1为操作失败

"msg": "部署工作流模型成功" //返回的信息；

}

}

**e.功能实现说明**

1. 执行repositoryService接口的getMode方法，传入modelId参数，就可以得到这个模型。
2. 读这个模型的EditorSource文件放到ObjectNode中去，在把Object转化为BpmnModel，在从BpmnModel中得到byte[]数据。
3. 在把byte[]数据放到字符串bpmnString中去，从工作流模型得到模型名modelName，流程名processName设置为modelName加上‘.bpmn20.xml’的组合。
4. 执行repositoryService接口的.createDeployment() . name(modelName). addString(processName ,bpmnString) .deploy()方法.就可以部署好这个流程pi。
5. 创建Resultd对象，设置msg为部署流程成功，flag为0，并返回这个Result对象。
   * 1. 启动流程接口

**a.接口url:**

pub/admin/system/workFlow/ startProcess.do

**b.接口说明：**

申请流程-创建一个流程实例

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| startProcess | | Result | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| processkey | String | | 流程id | |
| aName | String | | 申请人 | |

**d.返回值示例与说明**

{

"result": { //结果集;

"flag": "0", //标志，0为成功，1为操作失败

"msg": "申请成功" //返回的信息；

}

}

**e.功能实现说明**

1. 执行runtimeService().startProcessInstanceByKey(processkey)方法，启动一个流程实例。
2. 修改实例的NAME\_字段，将对应系统的信息id和aNme放入（信息id主动生成，平台信息中还没有这条信息）。
3. 同时创建第一个任务（流程初始任务），返回任务id。
4. 创建Resultd对象，设置msg为申请成功，flag为0，并返回这个Result对象。
   * 1. 执行（完成）任务接口

**a.接口url:**

pub/admin/system/workFlow/completeTask.do

**b.接口说明：**

执行（完成）一个流程任务

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| completeTask | | Result | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| taskId | String | | 任务id | |
| jsonInfo | String | | 上传的Json数据 | |
| userid | String | | 完成人姓名 | |
| username | String | | 完成人id | |
| completeUserIdOrRoleId | String | | 下个任务的处理权限 | |

**d.返回值示例与说明**

{

"result": { //结果集;

"flag": "0", //标志，0为成功，1为操作失败

"msg": "完成任务成功" //返回的信息；

}

}

**f.功能实现说明**

1. 根据任务id查到任务对象，通过getDescription()获取任务对象的Description（当前任务节点的判断关键字）。
2. 根据任务id查到该任务对应流程实例id，根据实例id查到实例对象，获取实例对象的Name字段。
3. 拆分例对象的Name字段，获取实例对象的信息id。
4. 执行infoUtil.operateInfoWay方法（该函数会返回信息id），传入jsonInfo，任务id, 人员userid, 人员姓名username。修改实例对应的平台信息，同步生成威客信息，获取威客信息id。
5. 根据1）中获取的判断关键字来从jsonInfo中取数据，再把变量名当成key值，取得数据当成value值。放到一个processVariables中。
6. 更新设置任务的威客id和对应处理人。
7. 根据任务id和processVariables完成任务。
8. 设置下一个任务的执行人completeUserIdOrRoleId。
9. 将当前任务的威客id放在下一个任务的From-key 用于页面显示。
10. 创建Resultd对象，设置msg为执行任务成功，flag为0，并返回这个Result对象。
    * 1. 查询某人未完成的任务

**a.接口url:**

pub/admin/system/workFlow/findUserWillTask.do

**b.接口说明：**

查询一个人未完成的任务

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| findUserWillTask | | infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| userid | String | | 人员id | |
| pageSize | String | | 每页尺寸 | |
| nowPage | String | | 当前页 | |
| orderType | String | | 排序类型 | |
| searchValue | String[] | | 搜索方法的关键字数组 | |
| searchTime | String | |  | |
| startTime | String | | 开始时间 | |
| endTime | String | | 结束时间 | |

**d.返回值示例与说明**

{

"page": { //分页信息;

"page": 2, //当前页;

"pages": 10, //总页数;

"lines": 100, //信息总条数;

"pageSize": 10, //每页显示信息数;

"pageStart": 1 //起始页;

},

"total": 8, //当前页信息条数;

"infos": [

{

"id": "", //任务Id

"assignee": "", //执行人

"createTime": "", //创建时间

"name": "", //姓名

"processDefinitionId": "", //流程定义id

"processInstanceId": "" , //流程实例id

"description": "", //描述

“inforId”:”” //信息Id

},

{}

]

}

**f.功能实现说明**

1. 通过用户id查询平台用户权限，再去工作流act\_ru\_task（当前待执行任务表），查询对应数据。act\_ru\_task（当前待执行任务表）的权限字段ASSIGNEE\_。
2. 返回这个infor对象。
   * 1. 查询某人已完成的任务

**a.接口url:**

pub/admin/system/workFlow/findUserDoneTask.do

**b.接口说明：**

查询一个人未完成的任务

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| findUserDoneTask | | infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| userid | String | | 人员id | |
| pageSize | String | | 每页尺寸 | |
| nowPage | String | | 当前页 | |
| orderType | String | | 排序类型 | |
| searchValue | String[] | | 搜索方法的关键字数组 | |
| searchTime | String | |  | |
| startTime | String | | 开始时间 | |
| endTime | String | | 结束时间 | |

**d.返回值示例与说明**

{

"page": { //分页信息;

"page": 2, //当前页;

"pages": 10, //总页数;

"lines": 100, //信息总条数;

"pageSize": 10, //每页显示信息数;

"pageStart": 1 //起始页;

},

"total": 8, //当前页信息条数;

"infos": [

{

"id": "", //任务Id

"assignee": "", //执行人

"createTime": "", //创建时间

"name": "", //姓名

"processDefinitionId": "", //流程定义id

"processInstanceId": "" , //流程实例id

"description": "", //描述

“inforId”:”” //信息Id

},

{}

]

}

**f.功能实现说明**

1. 通过用户id查询平台用户权限，再去工作流act\_hi\_taskinst（所有任务列表），查询对应数据。act\_hi\_taskinst（所有任务列表）的权限字段OWNER \_。
2. 返回这个infor对象。
   * 1. 查询流程实例信息

**a.接口url:**

pub/admin/system/workFlow/sezrch HistoricProcessInstance.do

**b.接口说明：**

查询单个流程的基本信息

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| sezrchHistoricProcess Instance | | Map<String,Object> | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| processInstanceId | String | | 流程实例Id | |

**d.返回值示例与说明**

{

}

**f.功能实现说明**

1. 执行historyService接口的createHistoricProcessInstanceQuery () . processInstanceId(businessKey). singleResult();
2. 得到HistoricProcessInstance的数据。
3. 定义一个map，把HistoricProcessInstance的一些参数放到这个map中
4. 返回map。
   * 1. 查询流程实例的过程（任务列表）

**a.接口url:**

pub/admin/system/workFlow/searchTaskHis.do

**b.接口说明：**

查询一个流程的执行了哪些任务

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| searchTaskHis | | List<Map<String, Object>> | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| processInstanceId | String | | 流程实例Id | |

**d.返回值示例与说明**

[

{

"id": "", //任务Id

"assignee": "", //执行人

"createTime": "", //创建时间

"name": "", //姓名

"processDefinitionId": "", //流程定义id

"processInstanceId": "" , //流程实例id

"description": "", //描述

"endTime": "", //结束时间

}

]

**f.功能实现说明**

1. 执行historyService接口的createHistoricTaskInstanceQuery() . processInstanceId(processInstanceId). list();
2. 得到List<HistoricTaskInstance>的数据。
3. 再把List<HistoricTaskInstance>返回到页面
   * 1. 查询流程定义列表

**a.接口url:**

pub/admin/system/workFlow/sezrch ProcessDefinitions.do

**b.接口说明：**

查询已经部署的流程有哪些

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| sezrch ProcessDefinitions | | Infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| pageSize | Integer | | 每页尺寸 | |
| nowPage | Integer | | 当前页 | |

**d.返回值示例与说明**

{

"page": { //分页信息;

"page": 2, //当前页;

"pages": 10, //总页数;

"lines": 100, //信息总条数;

"pageSize": 10, //每页显示信息数;

"pageStart": 1 //起始页;

},

"total": 8, //当前页信息条数;

"infos": [

{

"DeploymentId": "1", //部署Id

"id": "leaveProcess:1:4", //流程定义Id

"Key": "leaveProcess", //流程定义的Key

"Name": "leaveMyProcess" //流程名

} ,

{}

]

}

**f.功能实现说明**

1. 执行repositoryService接口的createProcessDefinitionQuery () .listPage(nowPage, pageSize);
2. 往方法中传入nowPage和pageSize这两个参数，把得到的数据放到infor对象中的infos字段。把得到数据的长度放到infor对象中的total字段中。
3. 执行repositoryService接口的createHistoricProcessInstanceQuery().count()方法，把得到的数据放到infor对象中的page对象的lines变量中，在给page对象设置page,pages,pageSize,pageStart字段。
4. 返回这个infor对象。
   * 1. 查询流程实例列表

**a.接口url:**

pub/admin/system/workFlow/sezrch HistoricProcessInstances.do

**b.接口说明：**

求一个流程产生了多少条流程实例

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| sezrchHistoricProcess Instances | | Infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| processDefinitionId | String | | 流程定义id | |
| pageSize | Integer | | 每页尺寸 | |
| nowPage | Integer | | 当前页 | |

**d.返回值示例与说明**

{

"page": { //分页信息;

"page": 2, //当前页;

"pages": 10, //总页数;

"lines": 100, //信息总条数;

"pageSize": 10, //每页显示信息数;

"pageStart": 1 //起始页;

},

"total": 8, //当前页信息条数;

"infos": [

{

"processDefinitionId": "leaveProcess:1:4", //流程定义Id

"name": null, //流程的名称

"description": null, //描述

"startTime": "Wed Sep 14 11:02:51 CST 2016", //流程的开始时间

"endTime": "Wed Sep 14 11:49:32 CST 2016", //流程的结束时间

"id": "2501" //流程的id

},

{}

]

}

**f.功能实现说明**

1. 执行HistoryService(). createHistoricProcessInstanceQuery(). processDefinitionId(processDefinitionId) .listPage(nowPage, pageSize)。
2. 往方法中传入processDefinitionId,nowPage和pageSize这三个参数，把得到的数据放到infor对象中的infos字段。把得到数据的长度放到infor对象中的total字段中。
3. 执行HistoryService接口的createHistoricProcessInstanceQuery(). processDefinitionId(processDefinitionId. count()方法，把得到的数据放到infor对象中的page对象的lines变量中，在给page对象设置page,pages,pageSize,pageStart字段。
4. 返回这个infor对象。
   * 1. 查询某个节点下待完成任务列表

**a.接口url:**

pub/admin/system/workFlow/searchNodeWillTaskByKey.do

**b.接口说明：**

通过条件key 查询某个节点下待完成任务列表

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| searchNodeWillTaskByKey | | Infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| nodeId | String | | 节点id | |
| key | String[] | | 搜索方法的关键字数组 | |
| pageSize | Integer | | 每页尺寸 | |
| nowPage | Integer | | 当前页 | |

**d.返回值示例与说明**

{

"page": { //分页信息;

"page": 2, //当前页;

"pages": 10, //总页数;

"lines": 100, //信息总条数;

"pageSize": 10, //每页显示信息数;

"pageStart": 1 //起始页;

},

"total": 8, //当前页信息条数;

"infos": [

{

"processDefinitionId": "leaveProcess:1:4", //流程定义Id

"name": null, //流程的名称

"description": null, //描述

"startTime": "Wed Sep 14 11:02:51 CST 2016", //流程的开始时间

"endTime": "Wed Sep 14 11:49:32 CST 2016", //流程的结束时间

"id": "2501" //流程的id

},

{}

]

}

**f.功能实现说明**

* + 1. 查询某个节点下已完成任务列表

**a.接口url:**

pub/admin/system/workFlow/searchNodeWillTaskByKey.do

**b.接口说明：**

通过条件key 查询某个节点下已完成任务列表

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| searchNodeWillTaskByKey | | Infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| nodeId | String | | 节点id | |
| key | String[] | | 搜索方法的关键字数组 | |
| pageSize | Integer | | 每页尺寸 | |
| nowPage | Integer | | 当前页 | |

**d.返回值示例与说明**

{

"page": { //分页信息;

"page": 2, //当前页;

"pages": 10, //总页数;

"lines": 100, //信息总条数;

"pageSize": 10, //每页显示信息数;

"pageStart": 1 //起始页;

},

"total": 8, //当前页信息条数;

"infos": [

{

"processDefinitionId": "leaveProcess:1:4", //流程定义Id

"name": null, //流程的名称

"description": null, //描述

"startTime": "Wed Sep 14 11:02:51 CST 2016", //流程的开始时间

"endTime": "Wed Sep 14 11:49:32 CST 2016", //流程的结束时间

"id": "2501" //流程的id

},

{}

]

}

**f.功能实现说明**

* + 1. 设置 more approve

**a.接口url:**

pub/admin/system/workFlow/setMoreApprove.do

**b.接口说明：**

设置 more approve

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| setMoreApprove | | Infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| auserid | String | | 授权人id | |
| buserids | String[] | | 被授权人id集合 | |
| taskid | String | | 任务id | |

**d.返回值示例与说明**

{

},

{}

]

}

**f.功能实现说明**

* + 1. 取消 more approve

**a.接口url:**

pub/admin/system/workFlow/cancelMoreApprove.do

**b.接口说明：**

可以在每次完成任务时调用,后台会判断是否被 more approve 过

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| cancelMoreApprove | | Infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| userid | String | | 人员id | |
| taskid | String | | 任务id | |

**d.返回值示例与说明**

{

{}

]

}

**f.功能实现说明**

* + 1. 流程工作授权

**a.接口url:**

pub/admin/system/workFlow/setWorkJurisdiction.do

**b.接口说明：**

流程工作授权

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| setWorkJurisdiction | | Infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| auserid | String | | 创建人 id | |
| buserid | String | | 授权人 id | |
| cuserids | String[] | | 被授权人 id 数组 | |
| startTime | Integer | | 授权开始时间(时间戳 秒) | |
| endTime | Integer | | 授权结束时间(时间戳 秒) | |
| application | String | | 应用 | |
| flow | String | | 流程 | |
| remark | String | | 备注 | |

**d.返回值示例与说明**

{

{}

]

}

**f.功能实现说明**

* + 1. 我申请的流程 最新信息列表

**a.接口url:**

pub/admin/system/workFlow/searchTaskForMyApply.do

**b.接口说明：**

通过userid 查询申请的流程 最新信息列表

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| searchTaskForMyApply | | Infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| userid | String | | 人员id | |
| pageSize | Integer | | 每页尺寸 | |
| nowPage | Integer | | 当前页 | |

**d.返回值示例与说明**

{

},

{}

]

}

**f.功能实现说明**

* + 1. 我参与的流程 最新信息列表

**a.接口url:**

pub/admin/system/workFlow/searchTaskForMyTake.do

**b.接口说明：**

我参与的流程 最新信息列表

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| searchTaskForMyTake | | Infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| userid | String | | 人员id | |
| pageSize | Integer | | 每页尺寸 | |
| nowPage | Integer | | 当前页 | |

**d.返回值示例与说明**

{

},

{}

]

}

**f.功能实现说明**

* 1. 旧接口
     1. 根据任务id查询任务对应实例所属javaplat平台的信息详情

**a.接口url:**

pub/admin/system/workFlow/findInfoByTaskid.do

**b.接口说明：**

查询一个人未完成的任务

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| findInfoByTaskid | | infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| taskid | String | | 任务id | |
| classid | String | | 栏目id | |

**d.返回值示例与说明**

{

"page": { //分页信息;

"page": 2, //当前页;

"pages": 10, //总页数;

"lines": 100, //信息总条数;

"pageSize": 10, //每页显示信息数;

"pageStart": 1 //起始页;

},

"total": 8, //当前页信息条数;

"infos": [

{

"id": "", //任务Id

"assignee": "", //执行人

"createTime": "", //创建时间

"name": "", //姓名

"processDefinitionId": "", //流程定义id

"processInstanceId": "" , //流程实例id

"description": "", //描述

“inforId”:”” //信息Id

},

{}

]

}

**f.功能实现说明**

1. 通过用户id查询平台用户权限，再去工作流act\_hi\_taskinst（所有任务列表），查询对应数据。act\_hi\_taskinst（所有任务列表）的权限字段OWNER \_。
2. 返回这个infor对象。
   * 1. 查询某人已完成或未完成的任务

**a.接口url:**

pub/admin/system/workFlow/decisionMakerTask.do

**b.接口说明：**

查询一个人已经或未完成的任务（根据节点id来查任务，根据决策人姓名来查任务或根据决策人姓名和流程定义id来查任务）

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| decisionMakerTask | | infor | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| nodeId | String | | 节点id | |
| processDefinitionId | String | | 流程定义id | |
| decisionMaker | String | | 决策人姓名 | |
| complete | Integer | | 0表示未完成的任务，1表示已经完成的任务 | |
| pageSize | Integer | | 每页尺寸 | |
| nowPage | Integer | | 当前页 | |

**d.返回值示例与说明**

{

"page": { //分页信息;

"page": 2, //当前页;

"pages": 10, //总页数;

"lines": 100, //信息总条数;

"pageSize": 10, //每页显示信息数;

"pageStart": 1 //起始页;

},

"total": 8, //当前页信息条数;

"infos": [

{

"id": "", //任务Id

"assignee": "", //执行人

"createTime": "", //创建时间

"name": "", //姓名

"processDefinitionId": "", //流程定义id

"processInstanceId": "" , //流程实例id

"description": "", //描述

“inforId”:”” //信息Id

},

{}

]

}

**f.功能实现说明**

1. 当complete参数为0的时候。
   1. 如果nodeId不为空且decisionMake不为空，执行TaskService接口的createTaskQuery. taskDefinitionKey(nodeId). taskCandidateOrAssigned (decisionMaker) . listPage(nowPage, pageSize)方法
   2. 如果nodeId不为空，执行TaskService接口的createTaskQuery. taskDefinitionKey(nodeId) .listPage(nowPage, pageSize)方法.（包括一个流程下节点下所有的任务，比如申请的话，所有申请人任务都在）
   3. 如果decisionMake不为空且processDefinitionId不为空，执行TaskService接口的createTaskQuery. processDefinitionId(processDefinitionId) . taskCandidateOrAssigned(decisionMaker).listPage(nowPage,pageSize)方法.（一个流程下，一个人的所有任务，可能有其他路径）
   4. 如果decisionMake不为空，执行TaskService接口的createTaskQuery. taskCandidateOrAssigned (tdecisionMaker) .listPage(nowPage, pageSize)方法.（一个人所有的任务，可能多个流程）
   5. 往方法中传入tdecisionMaker,nowPage和pageSize这三个参数，把得到的数据放到infor对象中的infos字段。把得到数据的长度放到infor对象中的total字段中。
   6. 查找每一个任务的上一个任务绑定的信息id变量，放到infor对象的list变量的一个对象中。
   7. 执行相应的查询数量方法，把得到的数据放到infor对象中的page对象的lines变量中，在给page对象设置page,pages,pageSize,pageStart字段。
   8. 返回这个infor对象。
2. 当complete参数为1的时候。
   1. 如果nodeId不为空且decisionMake不为空,执 行historyService(). taskDefinitionKey(nodeId) . taskAssignee(decisionMaker). Finished ( ).listPage(nowPage\*pageSize-pageSize, nowPage\*pageSize);
   2. 如果nodeId不为空,执 行historyService(). taskDefinitionKey(nodeId) .finished().listPage(nowPage\*pageSize-pageSize, nowPage\*pageSize);
   3. 如果decisionMake不为空且processDefinitionId不为空执行historyService()..createHistoricTaskInstanceQuery().processDefinitionId(processDefinitionId).taskAssignee(decisionMaker).finished().listPage(nowPage\*pageSize-pageSize, nowPage\*pageSize);
   4. 如果decisionMake不为空,执 行historyService()..createHistoricTaskInstanceQuery().taskAssignee(decisionMaker).finished().listPage(nowPage\*pageSize-pageSize, nowPage\*pageSize);
   5. 往方法中传入tdecisionMaker,（nowPage\*pageSize-pageSize）和pageSize这三个参数，把得到的数据放到infor对象中的infos字段。把得到数据的长度放到infor对象中的total字段中。
   6. 执行historyService接口的相应的count()方法，把得到的数据放到infor对象中的page对象的lines变量中，在给page对象设置page,pages,pageSize,pageStart字段。
   7. 返回这个infor对象。
      1. 查找节点候选人名单接口

**a.接口url:**

pub/admin/system/workFlow/findCandidater.do

**b.接口说明：**

查询此节点候选人名单

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| findCandidater | | List<Map<String,String>> | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| taskId | String | | 任务id | |

**d.返回值示例与说明**

[

{

"personName": "" //候选人姓名

},

{

"personName": "" //候选人姓名

}

]

**e.功能实现说明**

1. 执行taskService接口的getIdentityLinksForTask(taskId)方法。
2. 传入参数taskId，会得到IdentityLink对象的list，遍历list，把IdentityLink对象的UserId字段值赋值给personName变量，把personName变量放到一个map中，这个map中的数据放到list中
3. 返回list。
   * 1. 拾取任务接口

**a.接口url:**

pub/admin/system/workFlow/claimTask.do

**b.接口说明：**

拾取任务

**c. 接口定义：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口定义 | | | | |
| 接口名 | | 对象名 | | 返回值类型 |
| claimTask | | Result | | Json |
| 参数说明： | | | | |
| 参数名称 | 参数类型 | | 备注 | |
| taskId | String | | 任务id | |
| userId | String | | 拾取人名称 | |

**d.返回值示例与说明**

{

"result": { //结果集;

"flag": "0", //标志，0为成功，1为操作失败

"msg": "拾取成功" //返回的信息；

}

}

**e.功能实现说明**

1. TaskService接口执行claim方法，传入taskId和userId两个变量。
2. 创建Resultd对象，设置msg为拾取成功，flag为0，并返回这个Result对象。