Activiti第二天 流程实例、个人任务

课程安排：

Api：

流程实例：

流程实例是什么？

启动流程实例的方法（掌握）

两种方法，第二种方法在企业开发常用（涉及和业务系统整合）

查询流程实例（掌握）

查询结果如何和业务整合。

流程实例的暂停和激活（了解）

个人任务：

个人任务分配方法。（掌握）

三种分配方法，**以第二种（UEL表达式）和第三种方法（监听器设置）**在企业开发常用 。

查询个人任务。（掌握）

查询结果如何和业务整合。

完成个人任务（办理任务）

案例：

完成activiti实现采购流程管理。

创建采购单

提交采购单

审核采购单

学习目标：

1. 重点学习activiti开发方法
2. 学习springmvc和mybatis开发方法

# 复习

什么是工作流？

利用计算机技术对业务流程进行自动化管理，实现多个参与者按照预先定义的流程自动执行业务流程。

什么是activiti？

Activiti是实现工作流的一个引擎（框架，组件），通常activiti是要嵌入到业务系统实现业务流程管理。

1. 利用bpmn2.0标准进行流程定义
2. 将流程定义的文件（.bpmn和.png，.bpmn是必须的）部署到activiti中

此时，activiti就可以去管理部署的流程定义

1. 启动一个流程实例
2. 查询待办任务
3. 办理任务
4. 流程结束

三个环境：

环境1： 没有加入activiti的采购系统

和加入activiti的采购系统进行功能开发对比

Procject:



数据库：（包括采购系统的表）



环境2：用于activiti的api测试

Project:



数据库：(包括activiti数据库表，包括23张)



环境3：用于activiti和业务系统整合开发，加入activiti的采购系统

Project:



数据库：（包括activiti数据库表和采购系统的表）



Activiti的架构：

activiti.cfg.xml：spring配置文件，配置ProcessEngine对象和各各Service

ProcessEngineConfiguration：

在上边的activiti.cfg.xml进行配置，通过此对象创建ProcessEngine

**(重点)SpringProcessEngineConfiguration：用于spring和activiti的整合**

Service：

|  |  |
| --- | --- |
| RepositoryService | activiti的资源管理类 |
| RuntimeService | activiti的流程运行管理类 |
| TaskService | activiti的任务管理类 |
| HistoryService | activiti的历史管理类 |
| IdentityService | activiti的用户身份管理类 |
| FormService | activiti的表单管理类 |
| ManagerService | activiti的引擎管理类 |

注：红色标注为常用service。

什么是流程定义？

如果要让activiti去管理业务流程，需要按照bpmn2.0标准进行业务流程定义（.bpmn文件，png文件），流程定义的过程在线下操作，定义生成 的文件是静态的。

什么是流程定义部署？

在线下定义的bpmn文件，需要通过activiti的api部署到activiti的数据库中，这样activiti才能去按照流程定义的内容管理业务流程。

# 流程实例

## 什么是流程实例？

参与者（可以是用户也可以是程序）按照流程定义内容发起一个流程，发起的这个流程就是一个流程实例。是动态的。

流程定义和流程实例图解：

流程定义（.bpmn和png文件）

静态的

比如：定义采购流程

XXX创建一个采购单（采购单号：XXX），这时启动一个流程实例

动态的：流程的执行

李四创建一个采购单（采购单号：002），这时启动一个流程实例

动态的：流程的执行

张三创建一个采购单（采购单号：001），这时启动一个流程实例

动态的：流程的执行

总结：

流程定义是静态的，流程实例是动态的，流程实例的启动是需要参与者来操作的，流程实例按照流程定义来启动，不同的流程实例之间互不影响 。

使用流程实例记录不同流程的执行过程。

## 启动流程实例

必须先启动一个流程实例，activiti才开始去管理控制这个业务流程（流程实例）。

根据流程定义key来启动一个实例

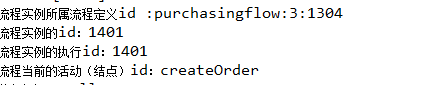
使用RuntimeService 启动一个流程实例，启动流程实例依赖于流程定义，启动实例之前 需要将流程定义部署到activit中。

RuntimeService：用于管理当前正在运行流程信息。

代码如下：

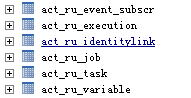


测试结果：



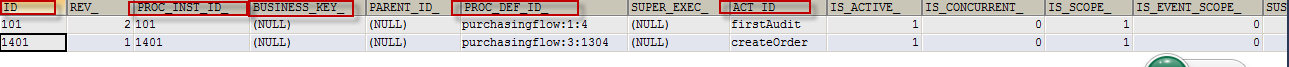
数据库表

记录流程当前 运行的信息：



SELECT \* FROM act\_ru\_execution #流程实例执行表

记录了流程实例的当前执行信息，如果该流程实例结束，流程实例的执行信息从该表删除。



Id\_：流程实例执行id

Proc\_inst\_id\_：流程实例id

流程实例执行id和流程实例id区别：

如果流程实例运行当前有多个分支，每个分支执行id和流程实例id不一样，总会有一条记录执行id和流程实例id是一样的。

记录数：**当前** 执行分支数+1 (每个分支执行id和流程实例id不一样)

Business\_key\_：业务标识

Proc\_def\_id\_：流程定义id

Act\_id\_：流程实例当前结点id（包括任务、事件等类型的结点）

SELECT \* FROM act\_hi\_procinst #流程实例历史 表

记录流程实例的历史 信息



Start\_time\_：流程实例开始运行时间，就是流程实例启动的时间

End\_time\_：流程实例结束运行时间，如果 为空表示该流程未结束

Duration：流程实例运行时长.

## businessKey(业务标识 )

分析

需求：

查询系统中当前正在运行的流程有哪些？比如：要查询采购流程，包括信息：流程实例id、当前 运行的结点id、采购单号、采购单名称 、采购金额。

分析：

流程实例id、当前 运行的结点id 通过activiti的api 从activiti数据库查询。

采购单号、采购单名称 、采购金额这个字段在activiti数据库中是不存在的，应该从业务系统中查询。

思路 ：

在启动每个流程实例时（比如采购流程），指定一个采购单id，将此采购单id记录到activiti的数据库中。

流程定义（.bpmn和png文件）

静态的

比如：定义采购流程

XXX创建一个采购单（采购单号：XXX），这时启动一个流程实例

动态的：流程的执行

启动时指定采购 单id(XXX)

一个流程实例

李四创建一个采购单（采购单号：002），这时启动一个流程实例

动态的：流程的执行

启动时指定采购 单id(002)

一个流程实例

张三创建一个采购单（采购单号：001），这时启动一个流程实例

动态的：流程的执行

启动时指定采购 单id(001)

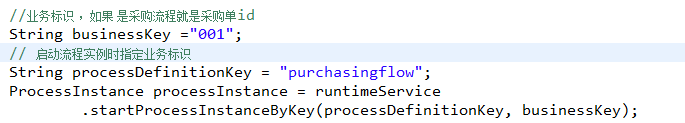
采购单id和每个流程实例一一对应，activiti中为了满足需求：在查询activiti的信息时关联查询业务信息。专门设置businessKey(业务标识 )

businessKey业务标识概念

业务标识：查询activiti的信息时关联查询业务信息要用到的一个标识，标识通常是业务表的一个主键。比如：如果启动一个采购流程，业务标识等于采购单id，如果启动一个请假流程，业务标识等于请假信息的id。

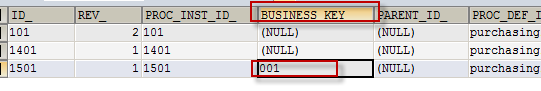
设置businessKey方法：

在启动一个流程实例时设置该流程实例对应的businessKey。

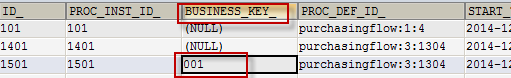


数据库表：

SELECT \* FROM act\_ru\_execution #流程实例执行表



SELECT \* FROM act\_hi\_procinst #流程实例历史 表



小结：

企业开发中，通常使用此方法启动一个流程实例，启动时指定流程实例对应业务标识 。

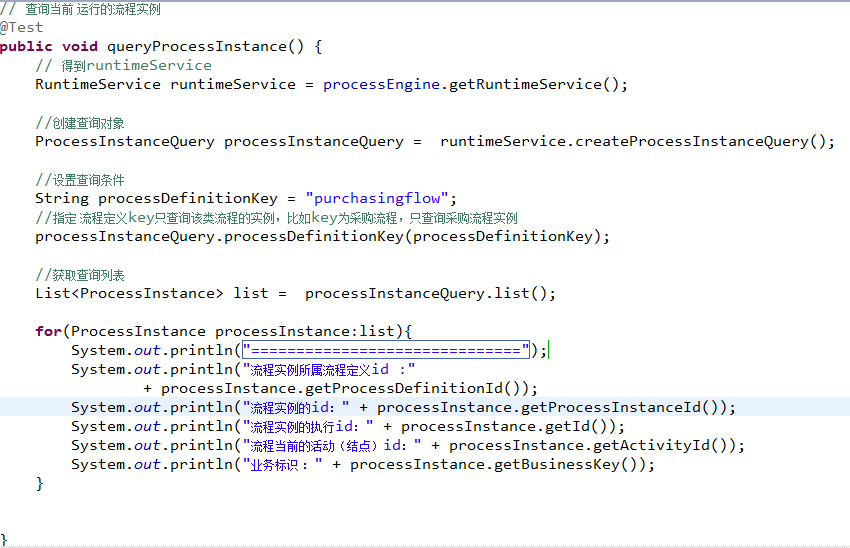
## 查询流程实例

需求：

查询系统中**当前正在运行**的流程有哪些？比如：要查询采购流程，包括信息：流程实例id、当前 运行的结点id、采购单号、采购单名称 、采购金额。

实现方法

使用RuntimeService查询系统中**当前正在运行**的流程。

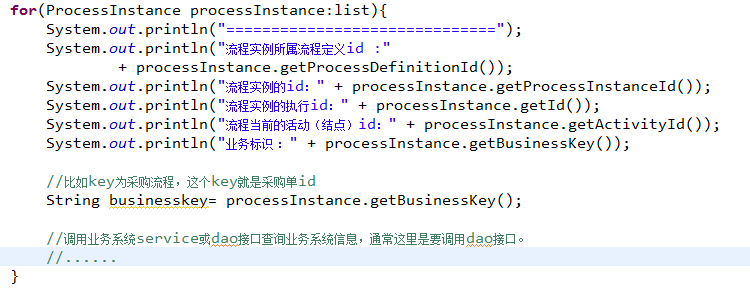


如果关联查询业务信息

思路：

在查询流程实例时，从processInstance中获取businesskey

根据businessKey查询业务系统数据库



关联查询业务信息的方法

流程实例id、当前 运行的结点id、采购单号、采购单名称 、采购金额（标记红的为业务信息）。

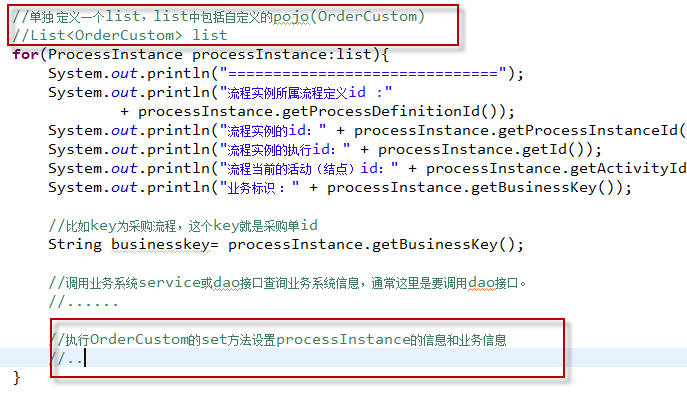
分析：

Activiti的processInstance对象不包括业务信息，需要单独创建一个pojo（包括业务信息和processInstance的信息）

这样作的意义：

**通过service将activiti和业务系统控制层进行隔离。**

代码思路：



## 挂起、激活流程实例(了解)

需求：

如果要删除流程实例：

方法1：通过级联删除流程定义将该流程定义下所有相关信息（包括流程实例）全部删除

注意：谨慎使用，通常是给超级管理使用。

方法2：不删除，暂停流程实例，好处：暂停后可以再恢复。

通常使用方法2。

根据流程定义挂起、激活

暂停或激活时指定流程定义id，该流程定义下所有流程实例全部暂停或激活。

**如果流程定义暂停或激活，不能再启动该流程定义的流程实例了!!!**

代码：

@Test

**public** **void** suspendOrActivateProcessDefinition() {

// 流程定义id

String processDefinitionId = "";

RepositoryService repositoryService = processEngine

.getRepositoryService();

// 获得流程定义

ProcessDefinition processDefinition = repositoryService

.createProcessDefinitionQuery()

.processDefinitionId(processDefinitionId).singleResult();

//是否暂停

**boolean** suspend = processDefinition.isSuspended();

**if**(suspend){

//如果暂停则激活，这里将流程定义下的所有流程实例全部激活

repositoryService.activateProcessDefinitionById(processDefinitionId, **true**, **null**);

System.*out*.println("流程定义："+processDefinitionId+"激活");

}**else**{

//如果激活则挂起，这里将流程定义下的所有流程实例全部挂起

repositoryService.suspendProcessDefinitionById(processDefinitionId, **true**, **null**);

System.*out*.println("流程定义："+processDefinitionId+"挂起");

}

}

根据流程实例挂起、激活

暂停或激活时指定流程实例id，对该流程实例进行暂停或激活。

**如果该流程实例id暂停了，流程实例下的任务无法办理。**

@Test

**public** **void** suspendOrActiveProcessInstance() {

// 流程实例id

String processInstanceId = "";

// 获取RunTimeService

RuntimeService runtimeService = processEngine.getRuntimeService();

//根据流程实例id查询流程实例

ProcessInstance processInstance = runtimeService.createProcessInstanceQuery()

.processInstanceId(processInstanceId).singleResult();

**boolean** suspend = processInstance.isSuspended();

**if**(suspend){

//如果暂停则激活

runtimeService.activateProcessInstanceById(processInstanceId);

System.*out*.println("流程实例："+processInstanceId+"激活");

}**else**{

//如果激活则挂起

runtimeService.suspendProcessInstanceById(processInstanceId);

System.*out*.println("流程实例："+processInstanceId+"挂起");

}

}

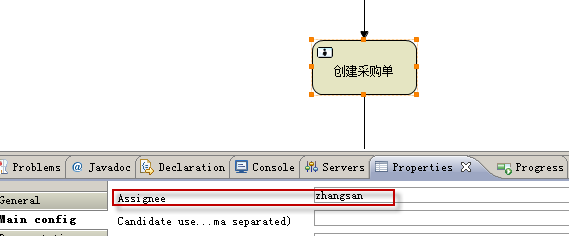
# 个人任务

个人任务，指定单个办理人(任务分配) 去办理任务。

## 任务分配

固定分配

设置该任务task的assignee（任务办理人）。



缺点：

将任务办理人硬编码在.bpmn文件中，如果该办理人变更需要修改bpmn文件，重新部署流程定义，过程麻烦，系统可以扩展性差。

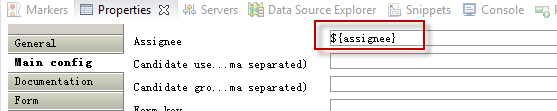
使用表达式(常用)

使用UEL表达式设置任务办理人。

Activiti使用UEL表达式，UEL是java EE6规范的一部分，UEL（Unified Expression Language）即统一表达式语言，activiti支持两个UEL表达式：UEL-value和UEL-method。

* UEL-value

UEL-value定义如下：

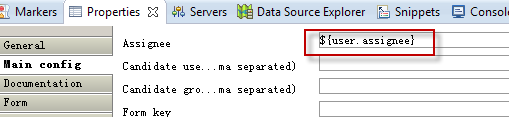


Assignee是一个流程变量，assignee是流程变量名称。

${assignee}获取流程变量的值 ，作为任务的办理人。

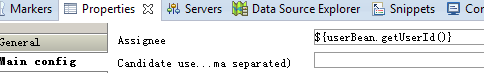
注意：如果 assignee流程变量不存在，activiti就会抛出异常。

从流程变量(pojo)中获取值：



user也是activiti的一个流程变量，是一个pojo，user.assignee表示通过调用user的getter方法获取值。

* UEL-method方法



userBean是spring容器中的一个bean，表示调用该bean的getUserId()方法。

通过调用 userBean的getUserId()方法，获取一个值，将这个值作为任务办理人。

* UEL-method与UEL-value结合

再比如：

${ ldapService.findManagerForEmployee(emp) }

**ldapService是spring容器的一个bean**，findManagerForEmployee是该bean的一个方法

**emp是activiti流程变量**，emp作为参数传到ldapService.findManagerForEmployee方法中。

* 其它

表达式支持解析基础类型、bean、list、array和map，也可作为条件判断。

如下：

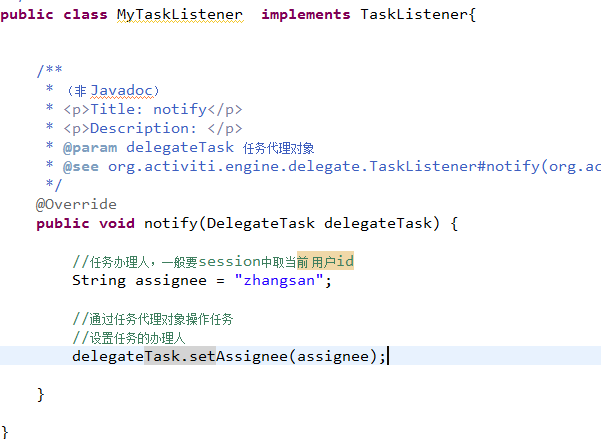
${order.price > 100 && order.price < 250}

使用监听器

定义步骤：

1. 自定义一个任务监听器，实现org.activiti.engine.delegate.TaskListener。

在监听器中要设置任务办理

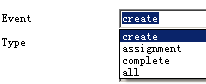


1. 在流程定义文件中设置任务监听器

方法1：修改.bpmn文件

方法2：通过activiti的流程设计 器设置监听器

确定 监听器执行时机：



Create：在任务创建时执行（在创建任务分配任务办理人）

Assignment：在任务分配时执行

Complete：在任务完成时执行

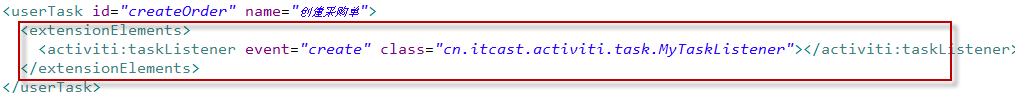
All：以上边全部情况下执行

选择监听器类：



查看.bpmn文件：

监听器配置如下：



注意：只要bpmn文件变更就需要重新部署流程定义文件。

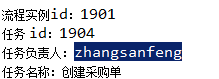
运行流程：

启动一个流程实例：

自动创建第一个任务

在第一个任务上设置了监听器（执行时机是在创建时执行），监听器设置该任务的办理人为”zhangsanfeng”

通过测试，监听器设置任务的办理人：



## 执行任务操作数据表

任务创建后会向当前任务表插入一条记录：

SELECT \* FROM act\_ru\_task #当前运行任务表

记录当前任务信息，如果任务完成了，从此表删除任务记录。



Id\_：任务id(主键) (重点)

对任务的操作根据此id执行

Execution\_id\_：对应流程实例的执行id

proc\_inst\_id\_：对应流程实例的id(重点)

proc\_def\_id\_：对应流程定义的id

name\_：任务名称(重点)

task\_def\_key\_：任务标识(重点)

assignee\_：任务办理人(重点)

SELECT \* FROM act\_ru\_identitylink #流程当前参与者(人、组)信息

记录流程当前参与者信息，记录了流程执行过程中所有参与者

SELECT \* FROM act\_hi\_taskinst #历史任务表(用于查询历史 信息)(重点)

记录任务的历史信息，在任务创建时向历史任务表插入一条记录

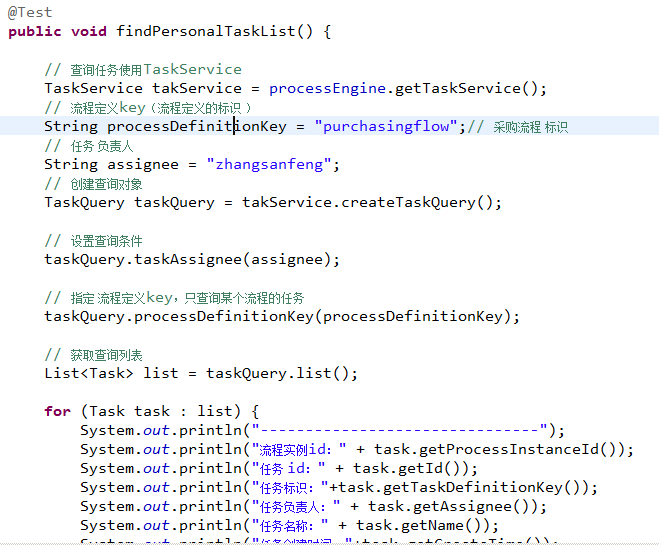
SELECT \* FROM act\_hi\_actinst #历史活动表(用于查询历史 信息)

记录流程执行过程中所有结点

## 任务查询

当前用户查询当前 待办任务：

使用TaskService查询当前 任务。



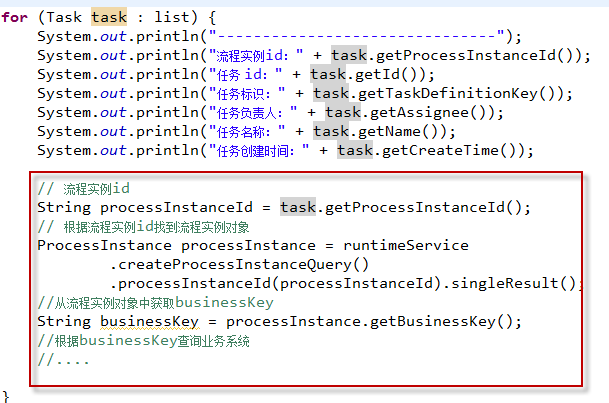
关联查询业务信息

需求：

查询用户当前待办任务，查询当前待处理采购单信息：

任务id、任务办理人、任务名称、采购单号、采购单名称、采购金额。。

红色：业务信息，从业务系统数据库查，根据businesskey查询



小结

查询待办任务时必须要指定任务办理人assignee。



查询信息中如果包括业务信息，通过businesskey（通过流程实例获取）查询业务系统。

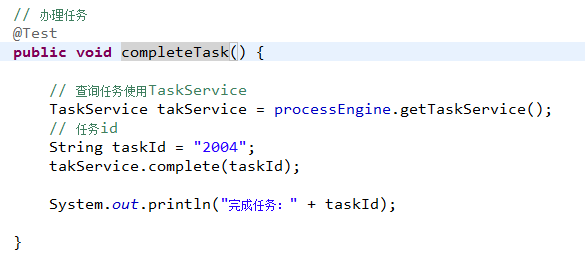
## 任务办理

任务办理：当前该任务的办理人去完成任务。

任务办理时指定：

任务id

任务办理人



任务办理人权限问题

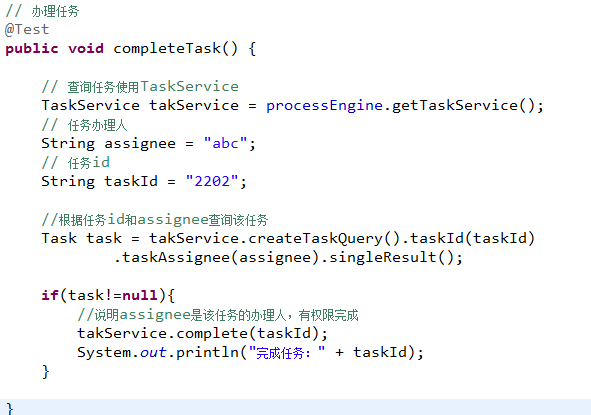
调用taskService.complete方法完成任务，activiti不要求传入任务办理人，问题：不管当前用户是哪位，只要指定任务id都能完成任务。

解决：

在调用taskService.complete方法完成任务之前校验任务办理是否有该任务的完成权限。

校验思路：

根据任务id和assignee查询该任务，如果查询到说明assignee就是该任务id的办理人。



# 案例分析

## 案例学习目标

1. 学习activiti在业务系统中开发方法
2. 学习springmvc+mybatis的开发方法

## 功能需求

业务需求不受activiti的影响 ，需求如下：



需求阶段，由需求分析人员进行用户需求分析，编写“**用户需求文档**”给开发人员和用户看的，最后编写“**系统需求规格说明书**”，此文档给开发人员看的。

上边业务流程要在需求文档和需求规格说明书中描述。

先分析出用户角色，再分析不同的角色之间完成的业务流程。

用户角色：

员工

部门经理

总经理

财务

注意：在需求阶段不用考虑工作流。

## 设计阶段

在设计阶段由设计人员依据“系统需求规格说明书”进行系统设计，编写“系统概要设计说明书”和“系统详细设计说明书”。

讨论确定哪些业务流程使用工作流，哪些业务流程由activiti去管理。

针对业务流程简单且需求变更不大的业务流程，建议不使用工作流，因为使用工作流增加开发成本 。

针对业务流程复杂且需求变更较大的业务流程，建议使用工作流去管理。

准备使用activiti去管理如下业务流程：



流程定义

对于activiti管理的业务流程，需要按照bpmn2.0标准进行流程定义，依据的原型就是需求分析阶段的业务流程图。

流程定义：

第一步：需要考虑哪个功能和流程启动操作对应的，即哪个功能执行后流程即将 启动

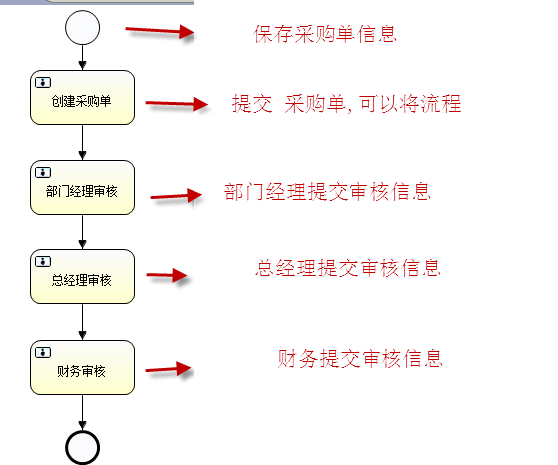
员工创建了一个新的采购单表示一个新采购流程开始了。

系统功能：创建采购单 --------> activiti：启动流程

确定创建采购单哪一步(用户保存采购单)表示启动流程。

第二步：考虑哪些功能将业务流程向后推进一步。

如下图：



功能设计

**创建采购单**功能：

**操作步骤**：

员工登陆系统

点击“创建采购单”，打开创建采购单页面

填写采购单信息（这里创建采购单可能会很复杂，比如分为三步创建）

填写采购单第一步：填写采购单基本信息

填写采购单第二步：填写采购明细信息(确定采购物品类型。。。)

保存采购单（确定启动activiti流程）

**操作后置条件**（大多数是数据库操作）：

操作1 向采购单表中插入一条记录

操作2 操作activiti的api启动流程(activiti和业务系统整合功能)

上边操作1和操作2在一个事务。

提交采购单功能：

提交采购单表示采购单不再修改了。

操作步骤：

员工登陆系统

查询待提交采购单

提交采购单

后置条件：

这里不用采用原始系统的方法更新采购单表的状态。

将流程推进功能交给activiti来完成。

调用 activiti的api将流程向后推进一步.

**调用taskService.complete(taskid) 将流程向后推进一步**

部门经理审核采购单提交：

操作步骤：

进入待审核列表

进入审核页面

填写审核信息

提交审核信息

后置条件：

向采购单审核表pur\_bus\_order\_audit插入一条记录

**调用taskService.complete(taskid) 将流程向后推进一步**

小结

对于activiti管理的业务流程，进行功能设计 时需要和activiti整合问题。

1、需要考虑哪个功能和流程启动操作对应的，即哪个功能执行后流程即将 启动

2、考虑哪些功能将业务流程向后推进一步

# 创建采购单

## 需求

功能需求：

员工在企业进行采购，需要首先创建一个采购单，员工创建完成采购单要经过三级审批。

操作步骤：

1、员工登陆系统

2、点击“创建采购单”，打开创建采购单页面

3、填写采购单信息

4、保存采购单

后置条件：

向采购单表中插入一条记录

操作activiti的api启动一个流程实例

## 设计实现

dao

向采购单表中插入一条记录

调用mybatis的mapper向采购单表pur\_bus\_order插入一条记录

service

接口功能：保存采购单，添加一个新的采购单

接口参数：采购单信息（OrderCustom扩展对象）,userId

接口实现：

向采购单表中插入一条记录

操作activiti的api启动一个流程实例

获取采购单id当成businessKey，使用businessKey启动一个流程实例

Public void saveOrder(String userId,OrderCustom orderCustom){

//调用mapper向采购单表中插入一条记录

//流程定义的key从配置文件获取

String processDefinitionKey = “”;

//获取采购单id

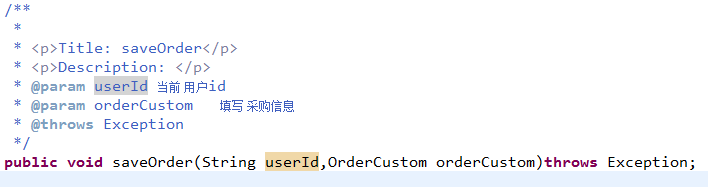
String businessKey = “采购单id”;

//调用activiti的api启动一个流程实例

runtimeService

.startProcessInstanceByKey(processDefinitionKey, businessKey);

}



在applicationContext-service.xml配置：

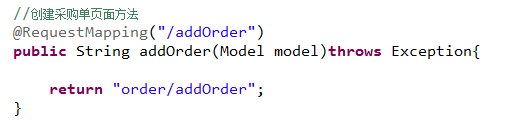


action

关注：用户的请求和响应。

1. 创建采购单页面方法

响应一个jsp页面（填写采购单信息）

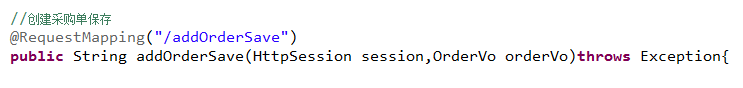


1. 保存填写采购单信息方法（参数：使用包括对象（包括orderCustom））

调用service：createOrder(OrderCustom orderCustom)

请求采购单信息

响应到采购单处理列表



页面

采购单信息页面：



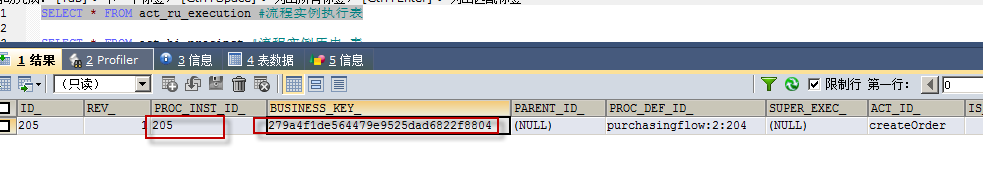
测试

创建采购单测试目标：

1. 检查业务系统采购单表是否添加了新记录



1. 检查activiti中是否启动一个新流程实例



测试步骤：

1. 部署流程定义
2. 创建采购单

# 查询待提交采购单

## 需求

员工创建采购单完成，在确定采购单不再修改，执行提交采购单，员工首先查询出待提交的采购单。

查询内容：任务id、任务名称、任务负责人、采购单名称、采购金额。

## 实现

dao

查询数据库采购单表，得到待提交 的采购单，由于原来采购单状态字段为空，无法查询出待提交的采购单。

不用实现dao，通过activiti的提供api查询出待提交的采购单。

service

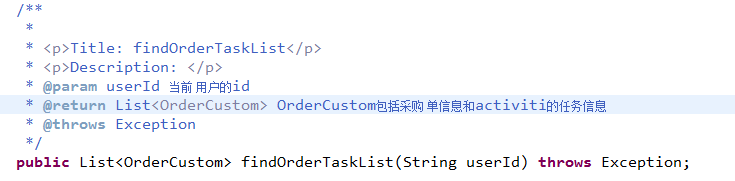
通过activiti的提供api查询出待提交的采购单。

接口功能：采购单待处理任务列表

接口参数：任务负责人

接口内容：

使用TaskService查询当前用户待办任务(待提交的采购单)



注意：

Service创建一个自定义的对象，使用对象将activiti和业务系统控制层进行隔离，防止将来修改工作流引擎，系统控制层是不需要修改代码。

即使service只返回activiti的task对象就可以满足，也要在中间加一个自定义的对象将activiti和业务系统控制层进行隔离。

@Override

**public** List<OrderCustom> findOrderTaskList(String userId) **throws** Exception {

// 流程定义key（流程定义的标识 ）

String processDefinitionKey = ResourcesUtil.*getValue*(

"diagram.purchasingflow", "purchasingProcessDefinitionKey");

// 任务 负责人

String assignee = userId;

// 创建查询对象

TaskQuery taskQuery = taskService.createTaskQuery();

// 设置查询条件

taskQuery.taskAssignee(assignee);

// 指定 流程定义key，只查询某个流程的任务

taskQuery.processDefinitionKey(processDefinitionKey);

//设置排序 字段，根据任务创建时间降序

taskQuery.orderByTaskCreateTime().desc();

// 获取查询列表

List<Task> list = taskQuery.list();

//需要返回的内容：任务id、任务标识 、任务名称、任务负责人、采购单名称、采购金额

List<OrderCustom> orderList = **new** ArrayList<OrderCustom>();

**for** (Task task : list) {

OrderCustom orderCustom =**new** OrderCustom();

// 流程实例id

String processInstanceId = task.getProcessInstanceId();

// 根据流程实例id找到流程实例对象

ProcessInstance processInstance = runtimeService

.createProcessInstanceQuery()

.processInstanceId(processInstanceId).singleResult();

// 从流程实例对象中获取businessKey

String businessKey = processInstance.getBusinessKey();

// 根据businessKey查询业务系统

//采购单id

String orderId = businessKey;

PurBusOrder purBusOrder = purBusOrderMapper.selectByPrimaryKey(orderId);

//获取采购单名称、采购金额等采购单信息

//将purBusOrder内容拷贝到orderCustom

BeanUtils.*copyProperties*(purBusOrder, orderCustom);

//下边向orderCustom开始设置任务信息

//任务id、任务标识 、任务名称、任务负责人

//任务id

orderCustom.setTaskId(task.getId());

//任务标识

orderCustom.setTaskDefinitionKey(task.getTaskDefinitionKey());

//任务名称

orderCustom.setTaskName(task.getName());

//任务负责人

orderCustom.setAssignee(task.getAssignee());

orderList.add(orderCustom);

}

**return** orderList;

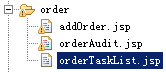
}

action

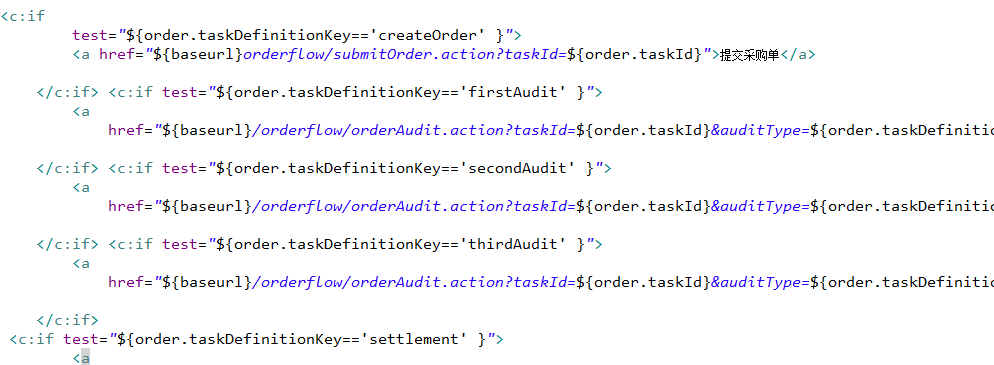
采购单处理列表：

查询当前用户采购单待处理任务列表。

页面



根据任务标识 ，显示不同的链接名称 。



# 提交采购单

## 需求

员工创建采购单完成，在确定采购单不再修改，执行提交采购单，提交后由经理审核。

## 实现

提交采购单目标将流程向后推进一步。

dao

不用实现.

service

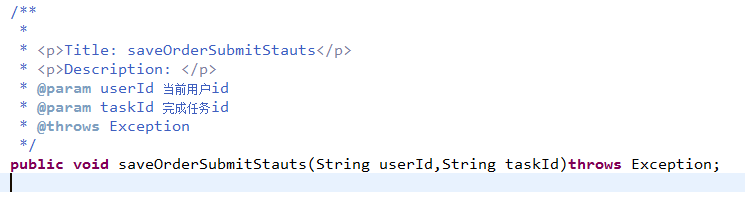
接口功能：提交采购单

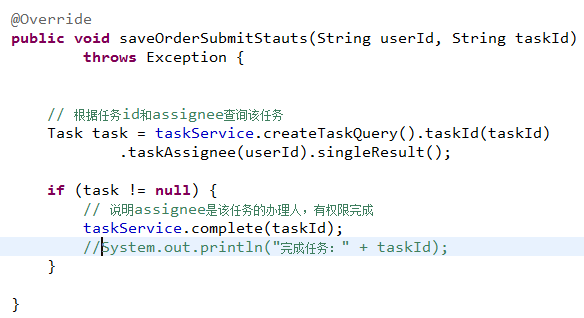
接口参数：任务id、任务负责人

接口内容:

调用acti

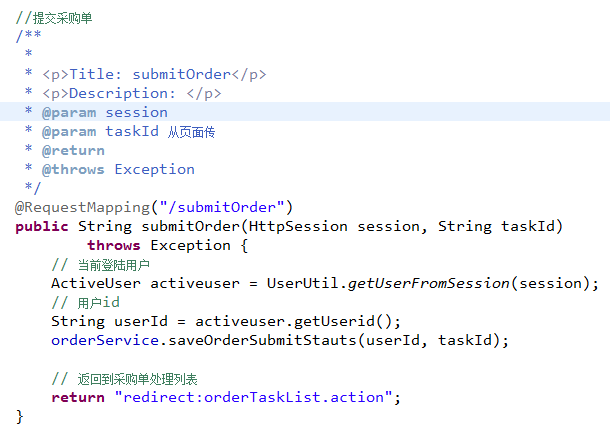
service执行任务完成。





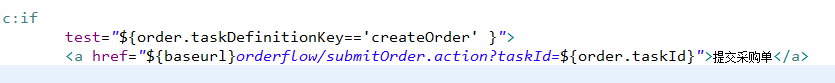
action

提交采购单：



页面

修改orderTaskList.jsp页面，添加提交采购单链接，将任务id传给action方法。



测试

测试步骤：

以zhangsan登陆，提交采购单，查询采购单处理列表，任务没有了

以lisi登陆，查询采购单处理列表，有一个部门经理审核的任务。

# 采购单审核

正式开发时为了系统扩展方便，需要将部门经理审核、总经理审核、财务审核定义不同的action、service。采用共用代码方式进行重用。

为了教学方便，定义一个action方法，一个service完成上边三级审核。

## 需求

员工提交采购单后，由部门经理审核、部门经理审核通过后由总经理审核，总经理审核通过由财务审核。

操作步骤：

进入审核页面

填写审核信息

提交审核

后置条件：

提交审核向pur\_bus\_order\_audit表插入一条记录

调用activiti的api执行任务完成操作，将流程向后推进一步

## 实现

dao

提交审核向pur\_bus\_order\_audit表插入一条记录

service

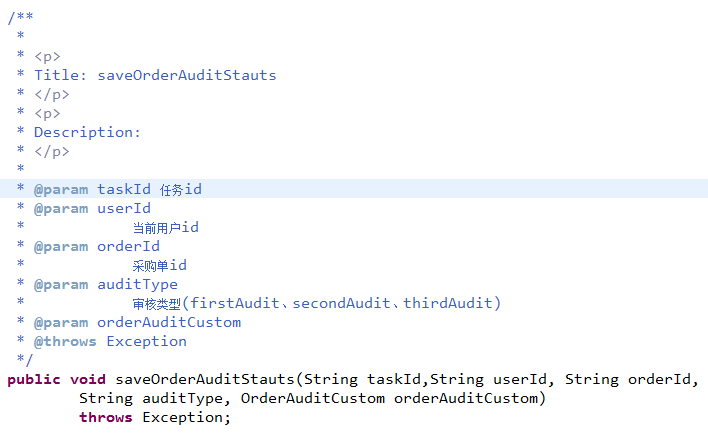
接口功能：采购单审核

接口参数：userId当前用户id, orderId采购单id、审核类型auditType、审核信息pojo(审核意见 和审核结果1：通过，0：不通过)、任务id

接口内容：

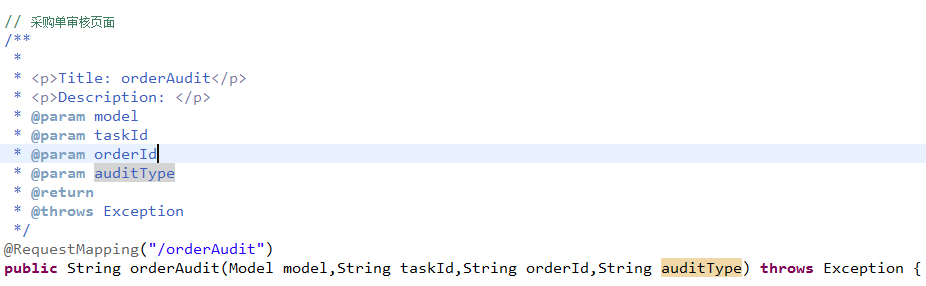
提交审核向pur\_bus\_order\_audit表插入一条记录

调用activiti的api执行任务完成操作，将流程向后推进一步

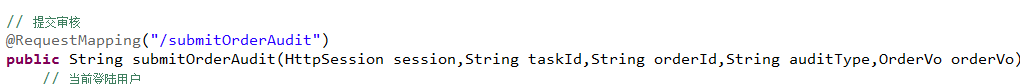


action

1. 进入审核页面方法



1. 提交审核方法

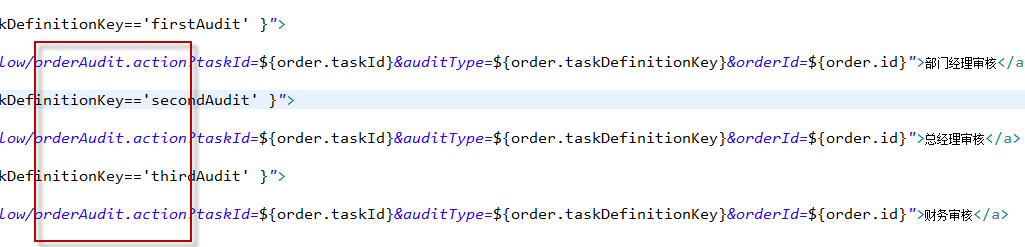


页面

修改采购单处理列表页面：

添加链接：部门经理审核、总经理审核、财务审核

点击审核链接，向action方法传入 参数：



编写采购单审核页面：



测试

分别以部门经理、总经理、财务登陆系统进行审核。

修改流程，业务系统代码不需要修改的。

## Activiti开发中遵循原则

1. 业务流程的管理由activiti负责，业务功能由业务系统负责
2. activiti和业务系统整合时，数据共享 问题，让activiti找到业务系统数据(通过businessKey)，让业务系统找到activiti的数据(在业务系统表中添加流程实例的id)。

3、需要自定义对象将activiti和业务系统控制层进行隔离。