**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. RepositoryService repositoryService = engine.getRepositoryService();
3. //查询流程定义，可以排序，查询数量，分页等
4. List<ProcessDefinition> list = repositoryService.createProcessDefinitionQuery()
5. .orderByDeploymentId().asc().list();
6. ProcessDefinition processDefinition = **null**;
7. **if**(list!=**null** && list.size()>0){
8. processDefinition = list.get(0);
9. System.out.println("流程组织机构： "+processDefinition.getCategory());
10. System.out.println("流程部署ID： "+processDefinition.getDeploymentId());
11. System.out.println("流程描述：       "+processDefinition.getDescription());
12. System.out.println("流程图片文件： "+processDefinition.getDiagramResourceName());
13. System.out.println("流程定义ID： "+processDefinition.getId());
14. System.out.println("流程定义key "+processDefinition.getKey());
15. System.out.println("流程设计名称： "+processDefinition.getName());
16. System.out.println("流程定义文件： "+processDefinition.getResourceName());
17. System.out.println("流程所有人ID：    "+processDefinition.getTenantId());
18. System.out.println("流程版本：       "+processDefinition.getVersion());
19. }

（2）查询最新版本：

a) 之前版本查询最新版本时可以使用

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. RepositoryService repositoryService = engine.getRepositoryService();
3. //将流程按照版本的升序排列得到一个所有流程的集合
4. List<ProcessDefinition> list = repositoryService.createProcessDefinitionQuery().orderByProcessDefinitionVersion().asc().list();
6. //使用map集合存储得到的集合，同时将所有低版本的过滤
7. Map<String, ProcessDefinition> map = **new** LinkedHashMap<String, ProcessDefinition>();
8. **if**(list!=**null** && list.size()>0){
9. **for** (ProcessDefinition pd : list)
10. {
11. //新版本数据将会替代就版本数据
12. map.put(pd.getKey(), pd);
13. }
14. }

17. //循环遍历输出
18. **for** (ProcessDefinition processDefinition : map.values())
19. {
20. System.out.println("流程组织机构： "+processDefinition.getCategory());
21. System.out.println("流程部署ID： "+processDefinition.getDeploymentId());
22. System.out.println("流程描述：       "+processDefinition.getDescription());
23. System.out.println("流程图片文件： "+processDefinition.getDiagramResourceName());
24. System.out.println("流程定义ID： "+processDefinition.getId());
25. System.out.println("流程定义key "+processDefinition.getKey());
26. System.out.println("流程设计名称： "+processDefinition.getName());
27. System.out.println("流程定义文件： "+processDefinition.getResourceName());
28. System.out.println("流程所有人ID：    "+processDefinition.getTenantId());
29. System.out.println("流程版本：       "+processDefinition.getVersion());
30. System.out.println("#############################");
31. }

b) 当前版本提供了一个便捷的API，可以直接查询最新版本

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. repositoryService.createProcessDefinitionQuery().latestVersion().singleResult();

（3）删除流程定义：

删除流程定义需要使用流程部署id删除部署的流程，流程定义便会相应删除。

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. RepositoryService repositoryService = engine.getRepositoryService();
2. List<ProcessDefinition> list = repositoryService.createProcessDefinitionQuery()
3. .orderByDeploymentId().asc().list();
5. ProcessDefinition processDefinition = **null**;
6. **if**(list!=**null** && list.size()>0){
7. processDefinition = list.get(0);
8. String id = processDefinition.getDeploymentId();
9. // 级联删除流程，不论是否启动，都会删除
10. repositoryService.deleteDeployment(id, **true**);
11. }

（4）挂起和激活流程：

当前API提供了一个挂起流程的操作，如果当前流程不使用的话可以将之挂起，使之不能启动。

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. RepositoryService repositoryService = engine.getRepositoryService();
2. //挂起流程定义，挂起后不能启动
3. //repositoryService.suspendProcessDefinitionByKey("SF");
5. //激活流程定义，激活后可以启动
6. repositoryService.activateProcessDefinitionByKey("SF");

**2.流程实例**

在流程启动之后会生成相应的流程实例记录，当前流程实例表示此流程正在运行，如果流程结束，流程实例相应删除，同时历史记录会更新。

（1）启动流程

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. //流程运行服务
2. RuntimeService runtimeService = engine.getRuntimeService();

5. //1.使用流程定义的id启动流程实例，返回值为流程实例对象
6. //ProcessInstance processInstance = runtimeService.startProcessInstanceById(id);
8. //2.使用流程定义的key启动流程实例,推荐使用
9. //同一个流程key相同，不同的是版本，使用key启动可以默认启动最新版本的流程
10. ProcessInstance processInstance = runtimeService.startProcessInstanceById(key);
12. //流程实例中包含的信息
13. System.out.println("当前活动节点  "+processInstance.getActivityId());
14. System.out.println("关联业务键：  "+processInstance.getBusinessKey());
15. System.out.println("流程部署id： "+processInstance.getDeploymentId());
16. System.out.println("流程描述：       "+processInstance.getDescription());
17. System.out.println("流程实例id： "+processInstance.getId());
18. System.out.println("流程实例名称： "+processInstance.getName());
19. System.out.println("父流程id：      "+processInstance.getParentId());
20. System.out.println("流程定义id： "+processInstance.getProcessDefinitionId());
21. System.out.println("流程定义key：    "+processInstance.getProcessDefinitionKey());
22. System.out.println("流程定义名称： "+processInstance.getProcessDefinitionName());
23. System.out.println("流程实例id： "+processInstance.getProcessInstanceId());
24. System.out.println("流程所属人id：    "+processInstance.getTenantId());
25. System.out.println("流程定义版本： "+processInstance.getProcessDefinitionVersion());
26. System.out.println("流程变量：       "+processInstance.getProcessVariables());
27. System.out.println("是否结束：       "+processInstance.isEnded());
28. System.out.println("是否暂停：       "+processInstance.isSuspended());
29. System.out.println("################################");

（2）删除流程实例

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. RuntimeService runtimeService = engine.getRuntimeService();
2. //会清空当前执行的流程表和当前任务表中的与当前流程实例对应的数据
3. //同时更新历史任务表中的数据
4. runtimeService.deleteProcessInstance("5001", "deleteReason");

（3）业务关联信息更新

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. RuntimeService runtimeService = engine.getRuntimeService();
3. //修改业务关联信息
4. runtimeService.updateBusinessKey("7501", "BusinessKey");

（4）暂停启动当前流程

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. RuntimeService runtimeService = engine.getRuntimeService();
2. //暂停
3. //runtimeService.suspendProcessInstanceById("7501");
4. //启用
5. runtimeService.activateProcessInstanceById("7501");

（5）查询流程定义

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. RuntimeService runtimeService = engine.getRuntimeService();
2. List<ProcessInstance> list = runtimeService.createProcessInstanceQuery()
3. .orderByProcessDefinitionId().asc().listPage(0, 10);
5. **if**(list!=**null** && list.size()>0){
6. **for** (ProcessInstance processInstance : list)
7. {
8. System.out.println("当前活动节点  "+processInstance.getActivityId());
9. System.out.println("关联业务键：  "+processInstance.getBusinessKey());
10. System.out.println("流程部署id： "+processInstance.getDeploymentId());
11. System.out.println("流程描述：       "+processInstance.getDescription());
12. System.out.println("流程实例id： "+processInstance.getId());
13. System.out.println("流程实例名称： "+processInstance.getName());
14. System.out.println("父流程id：      "+processInstance.getParentId());
15. System.out.println("流程定义id： "+processInstance.getProcessDefinitionId());
16. System.out.println("流程定义key：    "+processInstance.getProcessDefinitionKey());
17. System.out.println("流程定义名称： "+processInstance.getProcessDefinitionName());
18. System.out.println("流程实例id： "+processInstance.getProcessInstanceId());
19. System.out.println("流程所属人id：    "+processInstance.getTenantId());
20. System.out.println("流程定义版本： "+processInstance.getProcessDefinitionVersion());
21. System.out.println("流程变量：       "+processInstance.getProcessVariables());
22. System.out.println("是否结束：       "+processInstance.isEnded());
23. System.out.println("是否暂停：       "+processInstance.isSuspended());
24. System.out.println("################################");
25. }
26. }

（6）流程是否结束

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. String processInstanceId = "7501";
2. ProcessInstance pi = engine.getRuntimeService().createProcessInstanceQuery()
3. .processInstanceId(processInstanceId).singleResult();
4. **if**(pi==**null**){
5. System.out.println("流程已经结束");
6. }
7. **else**{
8. System.out.println("流程没有结束");
9. }

**3.流程执行**

流程启动之后会有一个执行对象

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. ExecutionQuery executionQuery = engine.getRuntimeService().createExecutionQuery();
3. //查询流程执行信息，可以根据条件查询
4. List<Execution> list = executionQuery.list();
5. **if**(list!=**null** && list.size()>0){
6. **for** (Execution e : list)
7. {
8. System.out.println("流程当前活动节点："+e.getActivityId());
9. System.out.println("流程描述：       "+e.getDescription());
10. System.out.println("流程ID：       "+e.getId());
11. System.out.println("流程名称：       "+e.getName());
12. System.out.println("父流程ID：      "+e.getParentId());
13. System.out.println("流程定义ID： "+e.getProcessInstanceId());
14. System.out.println("流程所有人ID：    "+e.getTenantId());
15. System.out.println("#######################");
16. }
17. }

**4.任务**

任务是在流程执行过程中产生的，对应的任务有个人任务和组任务，组任务最后也需要拾取为个人任务，任务从属于某一个流程实例，流程结束任务同时消失。

（1）任务批注

a) 添加删除批注

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. /\*\*
2. \* 添加，删除任务评论 act\_hi\_comment--历史评论表
3. \*/
4. @Test
5. **public** **void** testaddComment() {
7. /\*\*
8. \* 注意：添加批注的时候，由于Activiti底层代码是使用：
9. \*      String userId = Authentication.getAuthenticatedUserId();
10. \*      CommentEntity comment = new CommentEntity();
11. \*      comment.setUserId(userId);
12. \*  所有需要从Session中获取当前登录人，作为该任务的办理人（审核人），对应act\_hi\_comment表中的
13. \*  User\_ID的字段，不过不添加审核人，该字段为null
14. \*  所以要求，添加配置执行使用Authentication.setAuthenticatedUserId();添加当前任务的审核人
15. \* \*/
16. Authentication.setAuthenticatedUserId("user");
17. // 添加
18. // taskService.addComment("7504", "7501", "this is a test");
19. // 删除
20. taskService.deleteComments("7504", "7501");
21. }

b) 查看批注信息

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. HistoryService historyService = engine.getHistoryService();
2. List<HistoricProcessInstance> hplist = historyService.createHistoricProcessInstanceQuery()
3. .processInstanceId(piid).list();
4. **if** (hplist != **null** && hplist.size() > 0) {
5. **for** (HistoricProcessInstance hp : hplist) {
6. String htaskid = hp.getId();
7. List<Comment> lc = taskService.getTaskComments(htaskid);
8. list.addAll(lc);
9. }
10. }

以上使用的是流程实例查询，获取当前流程实例相关的批注信息，下面的是使用任务查询

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. String taskId = "7504";
2. List<Comment> list = **new** ArrayList<Comment>();
3. Task task = taskService.createTaskQuery().taskId(taskId).singleResult();
4. String piid = task.getProcessInstanceId();
5. list = taskService.getProcessInstanceComments(piid);
6. System.out.println(list);

（2）任务附件

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. /\*\*
2. \* 添加，删除附件 act\_ge\_bytearray--资源表中添加记录 act\_hi\_attachment--历史附件添加记录
3. \*/
4. @Test
5. **public** **void** testcreateAttachment() {
7. // Attachment attachment = taskService.createAttachment("1", "7504",
8. // "7501", "a", "aaa", new ByteArrayInputStream("haha".getBytes()));
10. // System.out.println("附件ID： "+attachment.getId());
12. taskService.deleteAttachment("12501");
13. }

（3）委派任务

委派任务是将任务的办理人设置为所委派的人

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. /\*\*
2. \* 委派任务 将任务的办理人设置为所委派的人
3. \*/
4. @Test
5. **public** **void** testdelegateTask() {
6. taskService.delegateTask("7504", "u1");
7. }

（4）回退任务

回退任务，将任务回退给委派之前的办理人,任务节点不变，需先有委派操作，然后进行回退

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. /\*\*
2. \* 回退任务，将任务回退给之前的办理人,任务节点不变
3. \*/
4. @Test
5. **public** **void** testresolveTask() {
6. taskService.resolveTask("7504");
7. }

（5）完成任务

使用任务id完成任务

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. /\*\*
2. \* 完成任务
3. \*/
4. @Test
5. **public** **void** testcomplete() {
6. taskService.complete("22502");
7. }

（6）设置办理人

直接为任务设置一个办理人

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. /\*\*
2. \* 设置任务办理人
3. \*/
4. @Test
5. **public** **void** testsetAssignee() {
6. taskService.setAssignee("22502", "u2");
7. }

（7）删除任务

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. /\*\*
2. \* 删除任务
3. \*/
4. @Test
5. **public** **void** testdeleteTask() {
6. taskService.saveTask(**new** TaskEntity("1234"));
8. // taskService.deleteTask(taskId);
9. taskService.deleteTask("1234", **true**);
10. // taskService.deleteTask(taskId, deleteReason);
11. // taskService.deleteTasks(taskIds);
12. }

（8）认领任务

在组任务中，需要将任务认领为个人任务才可以办理

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. /\*\*
2. \* 认领任务,从组任务中认领为个人任务
3. \*/
4. @Test
5. **public** **void** testCTask() {
6. taskService.claim("22502", "u1");
7. }

（9）查询任务

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. /\*\*
2. \* 查询个人任务
3. \*/
4. @Test
5. **public** **void** testTaskAssignee() {
6. TaskQuery taskQuery = taskService.createTaskQuery();
8. Task task = taskQuery.taskAssignee("u2").singleResult();
10. System.out.println("当前任务办理人：    " + task.getAssignee());
11. System.out.println("任务类型：       " + task.getCategory());
12. System.out.println("任务描述：       " + task.getDescription());
13. System.out.println("任务执行ID： " + task.getExecutionId());
14. System.out.println("表单key：      " + task.getFormKey());
15. System.out.println("任务ID：       " + task.getId());
16. System.out.println("任务名称：       " + task.getName());
17. System.out.println("任务所有者：  " + task.getOwner());
18. System.out.println("任务父ID：      " + task.getParentTaskId());
19. System.out.println("任务优先级：  " + task.getPriority());
20. System.out.println("流程定义的id：    " + task.getProcessDefinitionId());
21. System.out.println("流程实例的id：    " + task.getProcessInstanceId());
22. System.out.println("任务定义的key：   " + task.getTaskDefinitionKey());
23. System.out.println("所有人ID：      " + task.getTenantId());
24. System.out.println("任务创建时间： " + task.getCreateTime());
25. System.out.println("任务委派状态： " + task.getDelegationState());
26. System.out.println("持续时间：       " + task.getDueDate());
27. System.out.println("任务流程变量： " + task.getProcessVariables());
28. System.out.println("##############################");
29. }

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. /\*\*
2. \* 任务查询，可以使用各种条件查询，查讯数量等
3. \*/
4. @Test
5. **public** **void** testTaskQuery() {
6. List<Task> list = taskService.createTaskQuery().orderByTaskCreateTime().asc().list();
8. **if** (list != **null** && list.size() > 0) {
9. **for** (Task task : list) {
10. System.out.println("当前任务办理人：    " + task.getAssignee());
11. System.out.println("任务类型：       " + task.getCategory());
12. System.out.println("任务描述：       " + task.getDescription());
13. System.out.println("任务执行ID： " + task.getExecutionId());
14. System.out.println("表单key：      " + task.getFormKey());
15. System.out.println("任务ID：       " + task.getId());
16. System.out.println("任务名称：       " + task.getName());
17. System.out.println("任务所有者：  " + task.getOwner());
18. System.out.println("任务父ID：      " + task.getParentTaskId());
19. System.out.println("任务优先级：  " + task.getPriority());
20. System.out.println("流程定义的id：    " + task.getProcessDefinitionId());
21. System.out.println("流程实例的id：    " + task.getProcessInstanceId());
22. System.out.println("任务定义的key：   " + task.getTaskDefinitionKey());
23. System.out.println("所有人ID：      " + task.getTenantId());
24. System.out.println("任务创建时间： " + task.getCreateTime());
25. System.out.println("任务委派状态： " + task.getDelegationState());
26. System.out.println("持续时间：       " + task.getDueDate());
27. System.out.println("任务流程变量： " + task.getProcessVariables());
28. System.out.println("##############################");
29. }
30. }
31. }

**5.流程变量**

（1）启动时设置流程变量，同时还可以使用businessKey关联业务数据

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. /\*\*
2. \* 设置流程变量
3. \* 1.启动流程时设置,同时还可以使用businessKey关联业务数据
4. \* act\_hi\_varinst--历史流程变量表
5. \* act\_ru\_variable--运行时流程变量表
6. \*/
7. @Test
8. **public** **void** teststartProcessInstanceByKey(){
9. //engine.getRepositoryService().createDeployment().addClasspathResource("process/Var.bpmn").addClasspathResource("process/Var.png").name("var").deploy();
11. Map<String, Object> variables = **new** HashMap<String, Object>();
12. variables.put("u1", "u1");//启动流程同时设置任务办理人
13. ProcessInstance processInstance = engine.getRuntimeService()
14. .startProcessInstanceByKey("Var", "BusinessKey",variables);
16. //流程实例中包含的信息
17. System.out.println("当前活动节点  "+processInstance.getActivityId());
18. System.out.println("关联业务键：  "+processInstance.getBusinessKey());
19. System.out.println("流程部署id： "+processInstance.getDeploymentId());
20. System.out.println("流程描述：       "+processInstance.getDescription());
21. System.out.println("流程实例id： "+processInstance.getId());
22. System.out.println("流程实例名称： "+processInstance.getName());
23. System.out.println("父流程id：      "+processInstance.getParentId());
24. System.out.println("流程定义id： "+processInstance.getProcessDefinitionId());
25. System.out.println("流程定义key：    "+processInstance.getProcessDefinitionKey());
26. System.out.println("流程定义名称： "+processInstance.getProcessDefinitionName());
27. System.out.println("流程实例id： "+processInstance.getProcessInstanceId());
28. System.out.println("流程所属人id：    "+processInstance.getTenantId());
29. System.out.println("流程定义版本： "+processInstance.getProcessDefinitionVersion());
30. System.out.println("流程变量：       "+processInstance.getProcessVariables());
31. System.out.println("是否结束：       "+processInstance.isEnded());
32. System.out.println("是否暂停：       "+processInstance.isSuspended());
33. System.out.println("################################");
35. }

（2）为任务设置流程变量

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. /\*\*
2. \* 2.为任务设置流程变量
3. \*/
4. @Test
5. **public** **void** testsetVariable(){
6. TaskService taskService = engine.getTaskService();
7. //流程变量的值可以为任意类型
8. //taskService.setVariable("5005", "hello", new Date());
9. //taskService.setVariables(taskId, variables);
11. //与当前任务ID绑定的流程变量，当前任务结束后变量消失
12. taskService.setVariableLocal("5005", "bd", "aaa");
13. //任务完成，流程变量失效
14. //taskService.complete("5005");
15. }

（3）设置流程变量

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. RuntimeService runtimeService = engine.getRuntimeService();
2. TaskService taskService = engine.getTaskService();
4. /\*\*设置流程变量\*/
5. //      表示使用执行对象ID，和流程变量的名称，设置流程变量的值（一次只能设置一个值）
6. //      runtimeService.setVariable(executionId, variableName, value)
8. //      表示使用执行对象ID，和Map集合设置流程变量，map集合的key就是流程变量的名称，map集合的value就是流程变量的值（一次设置多个值）
9. //      runtimeService.setVariables(executionId, variables);
11. //      表示使用任务ID，和流程变量的名称，设置流程变量的值（一次只能设置一个值）
12. //      taskService.setVariable(taskId, variableName, value)
14. //      表示使用任务ID，和Map集合设置流程变量，map集合的key就是流程变量的名称，map集合的value就是流程变量的值（一次设置多个值）
15. //      taskService.setVariables(taskId, variables)
17. //      启动流程实例的同时，可以设置流程变量，用Map集合
18. //      runtimeService.startProcessInstanceByKey(processDefinitionKey, variables);
19. //      完成任务的同时，设置流程变量，用Map集合
20. //      taskService.complete(taskId, variables)

（4）获取流程变量

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. /\*\*获取流程变量\*/
2. //使用执行对象ID和流程变量的名称，获取流程变量的值
3. //      runtimeService.getVariable(executionId, variableName);
5. //使用执行对象ID，获取所有的流程变量，将流程变量放置到Map集合中，map集合的key就是流程变量的名称，map集合的value就是流程变量的值
6. //      runtimeService.getVariables(executionId);
8. //使用执行对象ID，获取流程变量的值，通过设置流程变量的名称存放到集合中，获取指定流程变量名称的流程变量的值，值存放到Map集合中
9. //      runtimeService.getVariables(executionId, variableNames);
10. //
11. //使用任务ID和流程变量的名称，获取流程变量的值
12. //      taskService.getVariable(taskId, variableName);
14. //使用任务ID，获取所有的流程变量，将流程变量放置到Map集合中，map集合的key就是流程变量的名称，map集合的value就是流程变量的值
15. //      taskService.getVariables(taskId);
17. //使用任务ID，获取流程变量的值，通过设置流程变量的名称存放到集合中，获取指定流程变量名称的流程变量的值，值存放到Map集合中
18. //      taskService.getVariables(taskId, variableNames);

**6.流程连线**

（1）流程变量影响流程走向

设计流程时，为流程走向的连线设置条件，使用流程变量将变量的值输入，从而根据条件判断流程走向

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. /\*\*
2. \* 启动流程
3. \* 流程为两分支，to\_u1走向u1节点，to\_u2走向u2节点
4. \*/
5. @Test
6. **public** **void** teststartProcessInstanceByKey(){
7. //部署流程
8. //engine.getRepositoryService().createDeployment().addClasspathResource("process/SequenceFlow.bpmn").addClasspathResource("process/SequenceFlow.png").name("sf").deploy();
10. Map<String, Object> variables = **new** HashMap<String, Object>();
11. variables.put("condition", "to\_u1");//启动流程同时设置流程变量，确定流程走向u1
12. ProcessInstance processInstance = engine.getRuntimeService()
13. .startProcessInstanceByKey("SF", "BusinessKey",variables);
15. //流程实例中包含的信息
16. System.out.println("当前活动节点  "+processInstance.getActivityId());
17. System.out.println("关联业务键：  "+processInstance.getBusinessKey());
18. System.out.println("流程部署id： "+processInstance.getDeploymentId());
19. System.out.println("流程描述：       "+processInstance.getDescription());
20. System.out.println("流程实例id： "+processInstance.getId());
21. System.out.println("流程实例名称： "+processInstance.getName());
22. System.out.println("父流程id：      "+processInstance.getParentId());
23. System.out.println("流程定义id： "+processInstance.getProcessDefinitionId());
24. System.out.println("流程定义key：    "+processInstance.getProcessDefinitionKey());
25. System.out.println("流程定义名称： "+processInstance.getProcessDefinitionName());
26. System.out.println("流程实例id： "+processInstance.getProcessInstanceId());
27. System.out.println("流程所属人id：    "+processInstance.getTenantId());
28. System.out.println("流程定义版本： "+processInstance.getProcessDefinitionVersion());
29. System.out.println("流程变量：       "+processInstance.getProcessVariables());
30. System.out.println("是否结束：       "+processInstance.isEnded());
31. System.out.println("是否暂停：       "+processInstance.isSuspended());
32. System.out.println("################################");
34. }

（2）获得当前任务的出口连线

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. /\*\*
2. \* 通过任务id
3. \* 查询连线路径信息
4. \*/
5. @Test
6. **public** **void** testSequenceFlow(){
7. //存放连线的名称集合
8. List<String> list = **new** ArrayList<String>();
9. //1:使用任务ID，查询任务对象
10. Task task = engine.getTaskService().createTaskQuery()
11. .taskId("17505")//使用任务ID查询
12. .singleResult();
13. //2：获取流程定义ID
14. String processDefinitionId = task.getProcessDefinitionId();
16. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*1.使用流程定义id得到流程定义对象\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
17. //3：查询ProcessDefinitionEntiy对象
18. ProcessDefinitionEntity processDefinitionEntity = (ProcessDefinitionEntity) engine.getRepositoryService()
19. .getProcessDefinition(processDefinitionId);
21. //使用任务对象Task获取流程实例ID
22. String processInstanceId = task.getProcessInstanceId();
24. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*2.使用流程实例得到当前活动节点\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
25. //使用流程实例ID，查询正在执行的执行对象表，返回流程实例对象
26. ProcessInstance pi = engine.getRuntimeService().createProcessInstanceQuery()//
27. .processInstanceId(processInstanceId)//使用流程实例ID查询
28. .singleResult();
29. //获取当前活动的id
30. String activityId = pi.getActivityId();
32. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*3.使用流程定义对象查询活动\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
33. //4：获取当前的活动
34. ActivityImpl activityImpl = processDefinitionEntity.findActivity(activityId);
36. //5：获取当前活动完成之后连线的名称
37. List<PvmTransition> pvmList = activityImpl.getOutgoingTransitions();//出连线
38. **if**(pvmList!=**null** && pvmList.size()>0){
39. **for**(PvmTransition pvm:pvmList){
40. String name = (String) pvm.getProperty("name");
41. **if**(StringUtils.isNotBlank(name)){
42. list.add(name);
43. }
44. **else**{
45. //list.add("默认");
46. }
47. }
48. }
49. System.out.println(list);
51. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
52. List<String> list2 = **new** ArrayList<String>();
53. //获得当前所有活动节点id
54. List<String> alist = engine.getRuntimeService().getActiveActivityIds(task.getExecutionId());
55. **for** (String str : alist)
56. {
57. ActivityImpl activity = processDefinitionEntity.findActivity(str);
58. List<PvmTransition> pvmList2 = activity.getIncomingTransitions();//入连线
59. **for** (PvmTransition pvm : pvmList2)
60. {
61. String name = (String) pvm.getProperty("name");
62. **if**(StringUtils.isNotBlank(name)){
63. list2.add(name);
64. }
65. **else**{
66. //list.add("默认");
67. }
68. }
69. }
70. System.out.println(list2);
71. }

7.**流程历史**

流程历史需要使用historyService服务进行。

（1）历史活动查询

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. /\*\*
2. \* 查询流程历史活动实例记录
3. \* act\_hi\_actinst--活动记录表
4. \*/
5. @Test
6. **public** **void** testactivityInstanceQuery(){
7. HistoricActivityInstanceQuery activityInstanceQuery = historyService.createHistoricActivityInstanceQuery();
8. //根据id查询
9. //activityInstanceQuery.activityId(activityId).singleResult();
11. //根据名称查询
12. //activityInstanceQuery.activityName(activityName).list();
14. //条件查询
15. List<HistoricActivityInstance> list = activityInstanceQuery.orderByActivityId().asc()
16. .orderByProcessDefinitionId().desc().listPage(0, 10);
18. **if**(list!=**null** && list.size()>0){
19. **for** (HistoricActivityInstance hai : list)
20. {
21. System.out.println("活动id：       "+hai.getActivityId());
22. System.out.println("活动名称：       "+hai.getActivityName());
23. System.out.println("活动类型：       "+hai.getActivityType());
24. System.out.println("任务办理人：  "+hai.getAssignee());
25. System.out.println("调用子流程实例id:"+hai.getCalledProcessInstanceId());
26. System.out.println("流程执行id： "+hai.getExecutionId());
27. System.out.println("记录id：       "+hai.getId());
28. System.out.println("流程定义id： "+hai.getProcessDefinitionId());
29. System.out.println("流程实例id： "+hai.getProcessInstanceId());
30. System.out.println("任务id：       "+hai.getTaskId());
31. System.out.println("历时时长：       "+hai.getDurationInMillis());
32. System.out.println("流程结束时间： "+hai.getEndTime());//未结束流程该值为空
33. System.out.println("流程开始时间： "+hai.getStartTime());
34. System.out.println("创建时间：       "+hai.getTime());
35. System.out.println("###########################");
36. }
37. }
38. }

（2）历史流程定义查询

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. /\*\*
2. \* 历史流程定义查询
3. \* act\_hi\_procinst--历史流程定义表
4. \*/
5. @Test
6. **public** **void** testhistoricProcessInstanceQuery(){
7. HistoricProcessInstanceQuery historicProcessInstanceQuery = historyService
8. .createHistoricProcessInstanceQuery();
10. //使用条件查询并按ProcessDefinitionId进行升序排列
11. List<HistoricProcessInstance> list = historicProcessInstanceQuery.processInstanceId("5001").or()
12. .processDefinitionId("Var:1:2504").endOr().orderByProcessDefinitionId().asc().list();
14. **if**(list!=**null** && list.size()>0){
15. **for** (HistoricProcessInstance historicProcessInstance : list)
16. {
17. System.out.println("历时流程实例："+historicProcessInstance);
18. }
19. }
20. }

（3）历史任务实例查询

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. /\*\*
2. \* 历史任务实例查询
3. \* act\_hi\_taskinst--历史任务实例表
4. \*/
5. @Test
6. **public** **void** testhistoricTaskInstanceQuery(){
7. HistoricTaskInstanceQuery historicTaskInstanceQuery = historyService
8. .createHistoricTaskInstanceQuery();
10. List<HistoricTaskInstance> list = historicTaskInstanceQuery.list();
12. **if**(list!=**null** && list.size()>0){
14. **for** (HistoricTaskInstance historicTaskInstance : list)
15. {
16. System.out.println("历史任务办理人：        "+historicTaskInstance.getAssignee());
17. System.out.println("历史任务描述：     "+historicTaskInstance.getDescription());
18. System.out.println("历史任务实例id：       "+historicTaskInstance.getId());
19. System.out.println("历史执行id：     "+historicTaskInstance.getExecutionId());
20. System.out.println("历史任务优先级：        "+historicTaskInstance.getPriority());
21. System.out.println("历史流程定义id：       "+historicTaskInstance.getProcessDefinitionId());
22. System.out.println("历史流程实例id：       "+historicTaskInstance.getProcessInstanceId());
23. System.out.println("历史任务定义key：      "+historicTaskInstance.getTaskDefinitionKey());
24. System.out.println("创建时间：           "+historicTaskInstance.getCreateTime());
25. System.out.println("历史流程变量：     "+historicTaskInstance.getProcessVariables());
26. System.out.println("###############################");
27. }
28. }
30. }

（4）其他

**[java]** [view plain](http://blog.csdn.net/cs1075341779/article/details/52817725) [copy](http://blog.csdn.net/cs1075341779/article/details/52817725)

1. //历史流程变量查询
2. historyService.createHistoricVariableInstanceQuery();
3. //历史详情表
4. historyService.createHistoricDetailQuery();
5. //历史流程时日志查询
6. //historyService.createProcessInstanceHistoryLogQuery(processInstanceId);