**[工作流学习——Activiti流程定义](http://blog.csdn.net/zwk626542417/article/details/46602419)**

标签： [工作流](http://www.csdn.net/tag/%e5%b7%a5%e4%bd%9c%e6%b5%81)[activiti 流程定义 crud](http://www.csdn.net/tag/activiti%20%e6%b5%81%e7%a8%8b%e5%ae%9a%e4%b9%89%20crud)

2015-06-23 09:51 16132人阅读 [评论](http://blog.csdn.net/zwk626542417/article/details/46602419#comments)(5) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/zwk626542417/article/details/46602419#report)

本文章已收录于：

[**Java 知识库**](http://lib.csdn.net/base/13?source=blogtop)

 分类：

java学习（46）

目录[(?)](http://blog.csdn.net/zwk626542417/article/details/46602419" \o "系统根据文章中H1到H6标签自动生成文章目录)[[+]](http://blog.csdn.net/zwk626542417/article/details/46602419)

**一、前言**

    在上一篇文章我们通过一个小demo对Activiti进行了宏观的介绍，让大家对Activiti有了整体的认识，这篇文章我们来学习具体的流程定义管理的CRUD.

**二、正文**

**流程定义是什么**

    ProcessDefinition（流程定义）就是一个流程的步骤说明，比如我们接下来要说的这个流程，申请人王三发起提交申请，李四作为部门经理进行审批，审批完成后，此申请到达下一级总经理王五，进行审批。就这么整个流程说明其实就是流程定义，不过在Activiti中整个流程定义是以helloworld.bpmn与helloworld.png格式存在的。

    在上一篇文章中我们只是稍微提了下，关于helloworld.bpmn是在流程设计器中拖拖拽拽形成的，其实还可以通过在配置文件中进行配置，具体的图形我已经放到了上一篇文章中了，现在我就将我配置好的helloworld.bpmn配置文件展示给大家：

**[html]** [view plain](http://blog.csdn.net/zwk626542417/article/details/46602419) [copy](http://blog.csdn.net/zwk626542417/article/details/46602419)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<definitions** xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:activiti="http://activiti.org/bpmn" xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI" xmlns:omgdc="http://www.omg.org/spec/DD/20100524/DC" xmlns:omgdi="http://www.omg.org/spec/DD/20100524/DI" typeLanguage="http://www.w3.org/2001/XMLSchema" expressionLanguage="http://www.w3.org/1999/XPath" targetNamespace="http://www.activiti.org/test"**>**
3. **<process** id="HelloWorld" name="HelloWorldProcess" isExecutable="true"**>**
4. **<startEvent** id="startevent1" name="Start"**></startEvent>**
5. **<endEvent** id="endevent1" name="End"**></endEvent>**
6. **<userTask** id="usertask1" name="提交申请" activiti:assignee="张三"**></userTask>**
7. **<userTask** id="usertask2" name="审批【部门经理】" activiti:assignee="李四"**></userTask>**
8. **<userTask** id="usertask3" name="审批【总经理】" activiti:assignee="王五"**></userTask>**
9. **<sequenceFlow** id="flow1" sourceRef="startevent1" targetRef="usertask1"**></sequenceFlow>**
10. **<sequenceFlow** id="flow2" sourceRef="usertask1" targetRef="usertask2"**></sequenceFlow>**
11. **<sequenceFlow** id="flow3" sourceRef="usertask2" targetRef="usertask3"**></sequenceFlow>**
12. **<sequenceFlow** id="flow4" sourceRef="usertask3" targetRef="endevent1"**></sequenceFlow>**
13. **</process>**
14. **<bpmndi:BPMNDiagram** id="BPMNDiagram\_HelloWorld"**>**
15. **<bpmndi:BPMNPlane** bpmnElement="HelloWorld" id="BPMNPlane\_HelloWorld"**>**
16. **<bpmndi:BPMNShape** bpmnElement="startevent1" id="BPMNShape\_startevent1"**>**
17. **<omgdc:Bounds** height="35.0" width="35.0" x="320.0" y="50.0"**></omgdc:Bounds>**
18. **</bpmndi:BPMNShape>**
19. **<bpmndi:BPMNShape** bpmnElement="endevent1" id="BPMNShape\_endevent1"**>**
20. **<omgdc:Bounds** height="35.0" width="35.0" x="320.0" y="430.0"**></omgdc:Bounds>**
21. **</bpmndi:BPMNShape>**
22. **<bpmndi:BPMNShape** bpmnElement="usertask1" id="BPMNShape\_usertask1"**>**
23. **<omgdc:Bounds** height="55.0" width="105.0" x="285.0" y="120.0"**></omgdc:Bounds>**
24. **</bpmndi:BPMNShape>**
25. **<bpmndi:BPMNShape** bpmnElement="usertask2" id="BPMNShape\_usertask2"**>**
26. **<omgdc:Bounds** height="55.0" width="105.0" x="285.0" y="240.0"**></omgdc:Bounds>**
27. **</bpmndi:BPMNShape>**
28. **<bpmndi:BPMNShape** bpmnElement="usertask3" id="BPMNShape\_usertask3"**>**
29. **<omgdc:Bounds** height="55.0" width="105.0" x="285.0" y="350.0"**></omgdc:Bounds>**
30. **</bpmndi:BPMNShape>**
31. **<bpmndi:BPMNEdge** bpmnElement="flow1" id="BPMNEdge\_flow1"**>**
32. **<omgdi:waypoint** x="337.0" y="85.0"**></omgdi:waypoint>**
33. **<omgdi:waypoint** x="337.0" y="120.0"**></omgdi:waypoint>**
34. **</bpmndi:BPMNEdge>**
35. **<bpmndi:BPMNEdge** bpmnElement="flow2" id="BPMNEdge\_flow2"**>**
36. **<omgdi:waypoint** x="337.0" y="175.0"**></omgdi:waypoint>**
37. **<omgdi:waypoint** x="337.0" y="240.0"**></omgdi:waypoint>**
38. **</bpmndi:BPMNEdge>**
39. **<bpmndi:BPMNEdge** bpmnElement="flow3" id="BPMNEdge\_flow3"**>**
40. **<omgdi:waypoint** x="337.0" y="295.0"**></omgdi:waypoint>**
41. **<omgdi:waypoint** x="337.0" y="350.0"**></omgdi:waypoint>**
42. **</bpmndi:BPMNEdge>**
43. **<bpmndi:BPMNEdge** bpmnElement="flow4" id="BPMNEdge\_flow4"**>**
44. **<omgdi:waypoint** x="337.0" y="405.0"**></omgdi:waypoint>**
45. **<omgdi:waypoint** x="337.0" y="430.0"**></omgdi:waypoint>**
46. **</bpmndi:BPMNEdge>**
47. **</bpmndi:BPMNPlane>**
48. **</bpmndi:BPMNDiagram>**
49. **</definitions>**

    通过流程设计器或者通过配置文件直接书写都是可以的。

**流程定义的CRUD**

**部署流程定义**

    在进行流程定义的操作之前，先要将流程定义进行部署，部署流程定义的方式有两种：

    1.部署流程定义的helloworld.bpmn与helloworld.png两个文件

**[java]** [view plain](http://blog.csdn.net/zwk626542417/article/details/46602419) [copy](http://blog.csdn.net/zwk626542417/article/details/46602419)

1. /\*\*
2. \* 部署流程定义 类路径从classpath
3. \*/
4. @Test
5. **public** **void** deoploymentProcessDefinition\_classpath() {
6. Deployment deployment = processEngine.getRepositoryService() // 与流程定义和部署对象相关的service
7. .createDeployment()// 创建一个部署对象
8. .name("流程定义")// 添加部署的名称
9. .addClasspathResource("diagrams/helloworld.bpmn")// 从classpath的资源中加载，一次只能加载一个文件
10. .addClasspathResource("diagrams/helloworld.png")// 从classpath的资源中加载，一次只能加载一个文件
11. .deploy();// 完成部署
12. System.out.println("部署ID：" + deployment.getId());
13. System.out.println("部署名称:" + deployment.getName());
14. }

运行结果：

    部署ID：301

    部署名称:流程定义

    2.将helloworld.bpmn与helloworld.png压缩成zip进行部署

**[java]** [view plain](http://blog.csdn.net/zwk626542417/article/details/46602419) [copy](http://blog.csdn.net/zwk626542417/article/details/46602419)

1. /\*\*
2. \* 部署流程定义 zip
3. \*/
4. @Test
5. **public** **void** deploymentProcessDefinition\_zip() {
6. InputStream in = **this**.getClass().getClassLoader()
7. .getResourceAsStream("diagrams/helloworld.zip");
8. ZipInputStream zipInputStream = **new** ZipInputStream(in);
9. Deployment deployment = processEngine.getRepositoryService()// 与流程定义和部署对象相关的service
10. .createDeployment()// 创建一个部署对象
11. .name("流程定义")// 添加部署
12. .addZipInputStream(zipInputStream)// 指定zip格式的文件完成部署
13. .deploy();// 完成部署
14. System.out.println("部署ID：" + deployment.getId());
15. System.out.println("部署名称:" + deployment.getName());
17. }

运行结果：

    部署ID：401

    部署名称:流程定义

我们将上面部署的过程进行下解释：

    1)先获取流程引擎对象：在创建时会自动加载classpath下的activiti.cfg.xml

    2)通过获取的流程引擎对象，通过流程引擎对象获取一个RepositoryService对象（仓库对象）

    3)由仓库的服务对象产生一个部署对象配置对象，用来封装部署操作的相关配置

    4)这是一个链式编程，在部署配置对象中设置显示名字，上传流程定义规则文件

    5)向**[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "_blank)**表中存放流程定义的规则信息

这些表都是跟部署对象和流程定义相关的表：

    act\_re\_deployment存放流程定义的显示名和部署时间，每部署一次增加一条记录；

    act\_re\_procdef（存放流程定义的属性信息，部署每个新的流程定义都会在这张表中增加一条记录，需要注意一下的当流程定义的key相同的情况下，使用的是版本升级；

    act\_ge\_bytearray存储流程定义相关的部署信息。即流程定义文档的存放地。每部署一次就会增加两条记录，一条是关于bpmn规则文件的，一条是图片的（如果部署时只指定了bpmn一个文件，activiti会在部署时解析bpmn文件内容自动生成流程图）。两个文件不是很大，都是以二进制形式存储在数据库中。

**流程定义的查询**

    关于流程定义在上面我们已经部署完毕了，在这里我们进行流程定义的查询，查询分成两个，一个是查询所有的流程定义还有一个查询最新版本的流程定义

查看所有的流程定义

**[java]** [view plain](http://blog.csdn.net/zwk626542417/article/details/46602419) [copy](http://blog.csdn.net/zwk626542417/article/details/46602419)

1. /\*\*
2. \* 查询所有的流程定义
3. \*/
4. @Test
5. **public** **void** findProcessDefinition() {
6. List<ProcessDefinition> list = processEngine.getRepositoryService()// 与流程定义和部署对象先相关的service
7. .createProcessDefinitionQuery()// 创建一个流程定义的查询
8. /\*\* 指定查询条件，where条件 \*/
9. // .deploymentId(deploymentId) //使用部署对象ID查询
10. // .processDefinitionId(processDefinitionId)//使用流程定义ID查询
11. // .processDefinitionNameLike(processDefinitionNameLike)//使用流程定义的名称模糊查询
13. /\* 排序 \*/
14. .orderByProcessDefinitionVersion().asc()
15. // .orderByProcessDefinitionVersion().desc()
17. /\* 返回的结果集 \*/
18. .list();// 返回一个集合列表，封装流程定义
19. // .singleResult();//返回惟一结果集
20. // .count();//返回结果集数量
21. // .listPage(firstResult, maxResults);//分页查询
23. **if** (list != **null** && list.size() > 0) {
24. **for** (ProcessDefinition pd : list) {
25. System.out.println("流程定义ID:" + pd.getId());// 流程定义的key+版本+随机生成数
26. System.out.println("流程定义的名称:" + pd.getName());// 对应helloworld.bpmn文件中的name属性值
27. System.out.println("流程定义的key:" + pd.getKey());// 对应helloworld.bpmn文件中的id属性值
28. System.out.println("流程定义的版本:" + pd.getVersion());// 当流程定义的key值相同的相同下，版本升级，默认1
29. System.out.println("资源名称bpmn文件:" + pd.getResourceName());
30. System.out.println("资源名称png文件:" + pd.getDiagramResourceName());
31. System.out.println("部署对象ID：" + pd.getDeploymentId());
32. System.out.println("#########################################################");
33. }
34. }
35. }

运行结果：

    流程定义ID:HelloWorld:1:304

    流程定义的名称:HelloWorldProcess

    流程定义的key:HelloWorld

    流程定义的版本:1

    资源名称bpmn文件:diagrams/helloworld.bpmn

    资源名称png文件:diagrams/helloworld.png

    部署对象ID：301

    #########################################################

    流程定义ID:HelloWorld:2:404

    流程定义的名称:HelloWorldProcess

    流程定义的key:HelloWorld

    流程定义的版本:2

    资源名称bpmn文件:helloworld.bpmn

    资源名称png文件:helloworld.png

    部署对象ID：401

    #########################################################

    从上面我们可以看出，流程定义key值相同的情况下，版本是从1开始逐次升级的，流程定义的id是【key：版本：生成ID】；

我们对上面代码进行下说明：

    1)流程定义和部署对象相关的Service都是RepositoryService。

    2)创建流程定义查询对象，可以在ProcessDefinitionQuery上设置查询的相关参数

    3)调用ProcessDefinitionQuery对象的list方法，执行查询，获得符合条件的流程定义列表

    4)由运行结果可以看出：Key和Name的值为：bpmn配置文件process节点的id和name的属性值

    5)key属性被用来区别不同的流程定义。

    6)带有特定key的流程定义第一次部署时，version为1。之后每次部署都会在当前最高版本号上加1

    7)Id的值的生成规则为:{processDefinitionKey}:{processDefinitionVersion}:{generated-id},这里的generated-id是一个自动生成的唯一的数字

    8)重复部署一次，deploymentId的值以一定的形式变化规则act\_ge\_property表生成

    查看最新版本的流程定义：

**[java]** [view plain](http://blog.csdn.net/zwk626542417/article/details/46602419) [copy](http://blog.csdn.net/zwk626542417/article/details/46602419)

1. 查看最新版本的流程定义：
2. /\*\*
3. \* 附加功能，查询最新版本的流程定义
4. \*/
5. @Test
6. **public** **void** findLastVersionProcessDefinition() {
7. List<ProcessDefinition> list = processEngine.getRepositoryService()
8. .createProcessDefinitionQuery()
9. .orderByProcessDefinitionVersion().asc() // 使用流程定义的版本升序排列
10. .list();
12. /\*\*
13. \* Map<String,ProcessDefinition> map集合的key：流程定义的key map集合的value：流程定义的对象
14. \* map集合的特点：当map集合key值相同的情况下，后一次的值将替换前一次的值
15. \*/
16. Map<String, ProcessDefinition> map = **new** LinkedHashMap<String, ProcessDefinition>();
17. **if** (list != **null** && list.size() > 0) {
18. **for** (ProcessDefinition pd : list) {
19. map.put(pd.getKey(), pd);
20. }
21. }
23. List<ProcessDefinition> pdList = **new** ArrayList<ProcessDefinition>(
24. map.values());
25. **if** (pdList != **null** && pdList.size() > 0) {
26. **for** (ProcessDefinition pd : pdList) {
27. System.out.println("流程定义ID:" + pd.getId());// 流程定义的key+版本+随机生成数
28. System.out.println("流程定义的名称:" + pd.getName());// 对应helloworld.bpmn文件中的name属性值
29. System.out.println("流程定义的key:" + pd.getKey());// 对应helloworld.bpmn文件中的id属性值
30. System.out.println("流程定义的版本:" + pd.getVersion());// 当流程定义的key值相同的相同下，版本升级，默认1
31. System.out.println("资源名称bpmn文件:" + pd.getResourceName());
32. System.out.println("资源名称png文件:" + pd.getDiagramResourceName());
33. System.out.println("部署对象ID：" + pd.getDeploymentId());
34. System.out
35. .println("#########################################################");
36. }
37. }
39. }

运行结果：

    流程定义ID:HelloWorld:2:404

    流程定义的名称:HelloWorldProcess

    流程定义的key:HelloWorld

    流程定义的版本:2

    资源名称bpmn文件:helloworld.bpmn

    资源名称png文件:helloworld.png

    部署对象ID：401

    #########################################################

    运行结果可看到我们可以查出最新版本的流程定义，查询与上面的全部查询是一样的，只不过多了一个过滤版本的功能，是用map来做代码很好理解。

**获取流程定义的文件资源**

    我们将流程定义部署完毕后，还可以查看流程定义的图片。

**[java]** [view plain](http://blog.csdn.net/zwk626542417/article/details/46602419) [copy](http://blog.csdn.net/zwk626542417/article/details/46602419)

1. /\*\*
2. \* 查看流程图
3. \*/
4. @Test
5. **public** **void** viewPic() **throws** IOException {
6. // 将生产的图片放到文件夹下
7. String deploymentId = "401";// TODO
8. // 获取图片资源名称
9. List<String> list = processEngine.getRepositoryService()
10. .getDeploymentResourceNames(deploymentId);
12. // 定义图片资源名称
13. String resourceName = "";
14. **if** (list != **null** && list.size() > 0) {
15. **for** (String name : list) {
16. **if** (name.indexOf(".png") >= 0) {
17. resourceName = name;
18. }
19. }
20. }
22. // 获取图片的输入流
23. InputStream in = processEngine.getRepositoryService()
24. .getResourceAsStream(deploymentId, resourceName);
26. File file = **new** File("D:/" + resourceName);
27. // 将输入流的图片写到D盘下
28. FileUtils.copyInputStreamToFile(in, file);
29. }

说明：

    1)deploymentId为流程部署ID

    2)resourceName为act\_ge\_bytearray表中NAME\_列的值

    3)使用repositoryService的getDeploymentResourceNames方法可以获取指定部署下得所有文件的名称

    4)使用repositoryService的getResourceAsStream方法传入部署ID和资源图片名称可以获取部署下指定名称文件的输入流

    5)最后的有关IO流的操作，使用FileUtils工具的copyInputStreamToFile方法完成流程流程到文件的拷贝，将资源文件以流的形式输出到指定文件夹下

**流程定义的删除**

    流程定义的删除，因为流程定义可以启动，所以涉及到一个普通删除和级联删除的情况，如果该流程定义下没有正在运行的流程，则可以用普通删除。如果是有关联的信息，用级联删除。关于删除我们既可以通过部署对象的id删除也可以通过流程定义的key删除，不同是使用id删除的只是一条记录，而使用key删除的是将key相同的所有版本的流程定义全部删除。

**[java]** [view plain](http://blog.csdn.net/zwk626542417/article/details/46602419) [copy](http://blog.csdn.net/zwk626542417/article/details/46602419)

1. /\*\*
2. \* 删除流程定义(删除key相同的所有不同版本的流程定义)
3. \*/
4. @Test
5. **public** **void** delteProcessDefinitionByKey() {
6. // 流程定义的Key
7. String processDefinitionKey = "HelloWorld";
8. // 先使用流程定义的key查询流程定义，查询出所有的版本
9. List<ProcessDefinition> list = processEngine.getRepositoryService()
10. .createProcessDefinitionQuery()
11. .processDefinitionKey(processDefinitionKey)// 使用流程定义的key查询
12. .list();
13. // 遍历，获取每个流程定义的部署ID
14. **if** (list != **null** && list.size() > 0) {
15. **for** (ProcessDefinition pd : list) {
16. // 获取部署ID
17. String deploymentId = pd.getDeploymentId();
18. //      /\*
19. //       \* 不带级联的删除， 只能删除没有启动的流程，如果流程启动，就会抛出异常
20. //       \*/
21. //       processEngine.getRepositoryService().deleteDeployment(deploymentId);
23. /\*\*
24. \* 级联删除 不管流程是否启动，都可以删除
25. \*/
26. processEngine.getRepositoryService().deleteDeployment(
27. deploymentId, **true**);
29. }
31. }
32. }

说明：

    1)因为删除的是流程定义，而流程定义的部署是属于仓库服务的，所以应该先得到RepositoryService

    2)根据流程定义的key先查询出key值相同的所有版本的流程定义，然后获取每个流程定义的部署对象id

    3)利用部署对象id，进行级联删除

    到这里我们就将流程定义的部署、查询、删除介绍完了，关于流程定义的修改其实就是在key值相同的情况下再次部署，让流程定义的版本进行升级，不影响以前的就版本的流程，对于新的流程就会默认使用最新版本的流程定义。

**三、总结**

    我们这篇文章主要讲解了流程定义的概念，然后详细的讲解了不同方式的流程定义部署，还讲解了流程定义的查询、流程定义的文档资源的获取、流程定义的删除等这些内容。