**activiti包容网关（Inclusive Gateway）**

转载[非·池](https://me.csdn.net/weixin_42068560" \t "_blank) 发布于2018-05-05 13:18:38 阅读数 3784  收藏

展开

**转载自：**<http://www.cnblogs.com/dengjiahai/p/8894339.html>

**1.什么是包容网关？**

包容网关（Inclusive Gateway）就是并行网关（Parallel Gateway）和排他网关（Exclusive Gateway）的组合。可以在出口顺序流上定义条件，包容网关会计算它们。然而主要的区别是，包容网关与并行网关一样，可以选择多于一条（出口）顺序流

**2.包容网关、并行网关和排他网关的异同：**

**同：**

1．都有出口顺序流和入口顺序流。

2．至少有一个分支

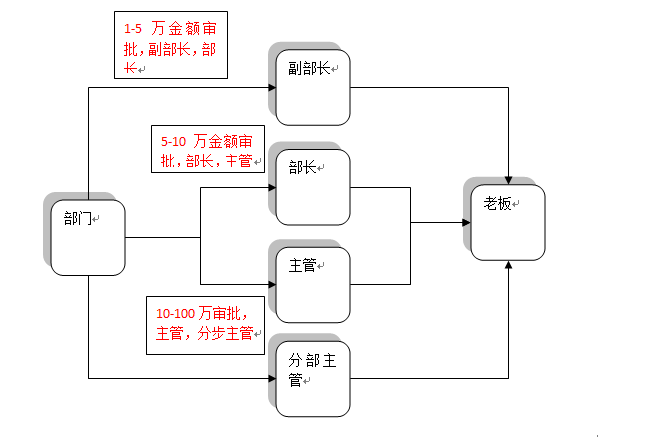
**异：**

1．排他网关只有一条分支被执行，如果有多条符合条件的分支，流程会默认走第一条。并行网关至少有一条分支被执行，而且所有的分支都会被执行。包容网关有多条或者一条分支会被执行。

2．包容网关包括了并行网关和排他网关的所有功能。

**3.使用场合：**

不确定分支的情形下面。例如：审批部门领导根据不同的条件确定。审批的会签根据不同的条件确定。



**4.包容网关两个重要的特性**

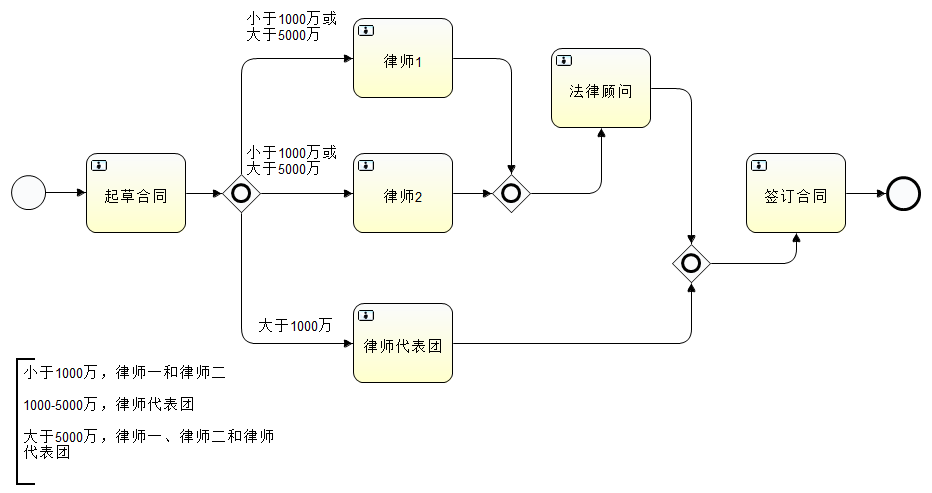
* 分支（fork）：所有出口顺序流都会被计算，对于计算为true的分支都会被执行。
* 聚合（join）：所有到达包容网关的并行执行，都会在网关处等待，直到每一条具有流程标志的入口顺序流，都有一个执行到达。这是与并行网关的重要区别。换句话说，包容网关只会等待将会被执行的入口顺序流。在合并后，流程穿过合并并行网关继续

**5.注意点：**

包容网关基于出口顺序流和入口顺序流。

包容网关，用内部带有’圆圈’图标的网关（菱形）表示。

**6.流程设计图：**

****

**8.运行流程：**

**1. package inclusiveGateway;**

**2.**

**3. import java.util.HashMap;**

**4. import java.util.List;**

**5. import java.util.Map;**

**6.**

**7. import org.activiti.engine.ProcessEngine;**

**8. import org.activiti.engine.ProcessEngines;**

**9. import org.activiti.engine.RepositoryService;**

**10. import org.activiti.engine.TaskService;**

**11. import org.activiti.engine.runtime.ProcessInstance;**

**12. import org.activiti.engine.task.Task;**

**13.**

**14. public class InclusiveGatewayTest01 {**

**15. /\*\* 完成我的任务 \*/**

**16. public static void main(String[] args) throws InterruptedException {**

**17. ProcessEngine processEngine = ProcessEngines.getDefaultProcessEngine();**

**18. RepositoryService repositoryService = processEngine.getRepositoryService();**

**19. repositoryService.createDeployment()**

**20. .addClasspathResource("diagrams/InclusiveGateway/InclusiveGatewayTest01.bpmn")**

**21. .deploy();**

**22. System.out.println("流程部署成功!");// 流程定义ID**

**23. // 流程定义的key**

**24. String processDefinitionKey = "InclusiveGatewayTest01";// 绘制的流程图ID**

**25. ProcessInstance pi = processEngine.getRuntimeService()// 与正在执行的流程实例和执行对象相关的Service**

**26. .startProcessInstanceByKey(processDefinitionKey);// 使用流程定义的key启动流程实例，key对应helloworld.bpmn文件中id的属性值。使用key值启动好处：默认是按照最新版本的流程定义启动**

**27. String procesInstanceId = pi.getId();**

**28. System.out.println("启动了一个ID为" + procesInstanceId + "的流程实例" );**

**29.**

**30. TaskService taskService = processEngine.getTaskService();**

**31. while(processEngine.getRuntimeService()**

**32. .createProcessInstanceQuery()**

**33. .processInstanceId(procesInstanceId)**

**34. .singleResult()!=null){//如果流程实例没有结束就一直查询当前实例的用户任务**

**35. // 查询当前到达的任务（多线）**

**36. List<Task> tasks = taskService.createTaskQuery()**

**37. .processInstanceId(procesInstanceId)**

**38. .list();**

**39. if(tasks!=null && tasks.size()>0){**

**40. for (Task t : tasks) {**

**41. // 设置变量**

**42. Map<String, Object> variables = new HashMap<String, Object>();**

**43. // variables.put("money", 200);**

**44. // variables.put("money", 1200);**

**45. variables.put("money", 5200);**

**46.**

**47. // 完成当前运行节点任务**

**48. taskService.complete(t.getId(),variables);**

**49. System.out.println("任务 " + t.getName() + " 执行完成!");**

**50. System.out.println("===================================");**

**51. }**

**52. }**

**53. }**

**54. System.out.println("流程实例： " + procesInstanceId + "执行完成!");**

**55. }**

**56. }**

**9.运行结果：**

流程部署成功!

启动了一个ID为177505的流程实例!

任务 起草合同 执行完成!

*===================================*

任务 律师1 执行完成!

*===================================*

任务 律师2 执行完成!

*===================================*

任务 律师代表团 执行完成!

*===================================*

任务 法律顾问 执行完成!

*===================================*

任务 签订合同 执行完成!

*===================================*

ID为 177505的流程实例执行完成!