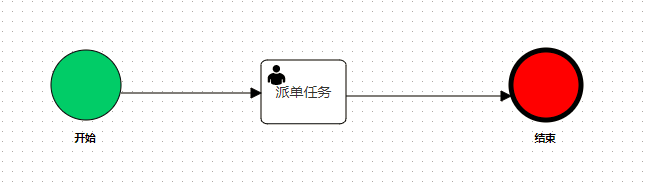
## 流程设计

### 最简化的流程

开始节点---》用户任务---》结束节点，如下图



用户任务中只需要配置处理人即可

### 任务

支持两种任务：用户任务和服务任务，图例如下



用户任务是需要处理人执行“完成”操作；服务任务是自动执行的，在它的“开始表达式”或“结束表达式”可以配置它需要执行的动作。

### 定时器事件定义

定时器事件是根据指定的时间触发的事件。可以用于 [开始事件](http://www.mossle.com/docs/activiti/index.html#bpmnTimerStartEvent)， [中间事件](http://www.mossle.com/docs/activiti/index.html#bpmnIntermediateCatchingEvent) 或 [边界事件](http://www.mossle.com/docs/activiti/index.html#bpmnTimerBoundaryEvent)

定时器定义必须下面介绍的一个元素：

* **timeDate**。使用 [ISO 8601](http://en.wikipedia.org/wiki/ISO_8601#Dates) 格式指定一个确定的时间，触发事件的时间。示例：

<timerEventDefinition>  
    <timeDate>2011-03-11T12:13:14</timeDate>  
</timerEventDefinition>

* **timeDuration**。指定定时器之前要等待多长时间， *timeDuration*可以设置为*timerEventDefinition*的子元素。 使用[ISO 8601](http://en.wikipedia.org/wiki/ISO_8601#Durations)规定的格式 （由BPMN 2.0规定）。示例（等待10天）。

<timerEventDefinition>  
    <timeDuration>P10D</timeDuration>  
</timerEventDefinition>

* **timeCycle**。指定重复执行的间隔， 可以用来定期启动流程实例，或为超时时间发送多个提醒。 timeCycle元素可以使用两种格式。第一种是 [ISO 8601](http://en.wikipedia.org/wiki/ISO_8601#Repeating_intervals) 标准的格式。示例（重复3次，每次间隔10小时）：

<timerEventDefinition>  
    <timeCycle>R3/PT10H</timeCycle>  
</timerEventDefinition>

另外，你可以使用cron表达式指定timeCycle，下面的例子是从整点开始，每5分钟执行一次：

0 0/5 \* \* \* ?

请参考[教程](http://www.quartz-scheduler.org/docs/tutorials/crontrigger.html" \t "_top) 来了解如何使用cron表达式。

**注意：** 第一个数字表示秒，而不是像通常Unix cron中那样表示分钟。

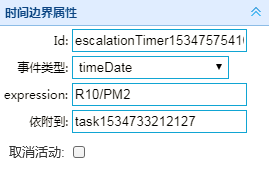
重复的时间周期能更好的处理相对时间，它可以计算一些特定的时间点 （比如，用户任务的开始时间），而cron表达式可以处理绝对时间 - 这对[定时启动事件](http://www.mossle.com/docs/activiti/index.html" \l "timerStartEventDescription" \o "描述)特别有用。

你可以在定时器事件定义中使用表达式，这样你就可以通过流程变量来影响那个定时器定义。 流程定义必须包含ISO 8601（或cron）格式的字符串，以匹配对应的时间类型。

  <boundaryEvent id="escalationTimer" cancelActivity="true" attachedToRef="firstLineSupport">  
     **<timerEventDefinition>  
      <timeDuration>${duration}</timeDuration>  
    </timerEventDefinition>**  
  </boundaryEvent>

**注意：** 只有启用job执行器之后，定时器才会被触发。 （activiti.cfg.xml中的*jobExecutorActivate*需要设置为true， 不过，默认job执行器是关闭的）。

#### 界面



事件类型：timeDate、timeDuration、timeCycle

依附到：用户任务 或 服务任务的Id

#### timeDuration使用实践

在指定时间到达后且关联用户任务时，需要设置cancelActivity=”true”，这样原有的分支会被撤消；

在指定时间到达后且关联服务任务，仅用于提醒时，需要设置cancelActivity=”false”

#### timeCycle使用实践

R/PT10H：表示每隔10小时，无限次循环

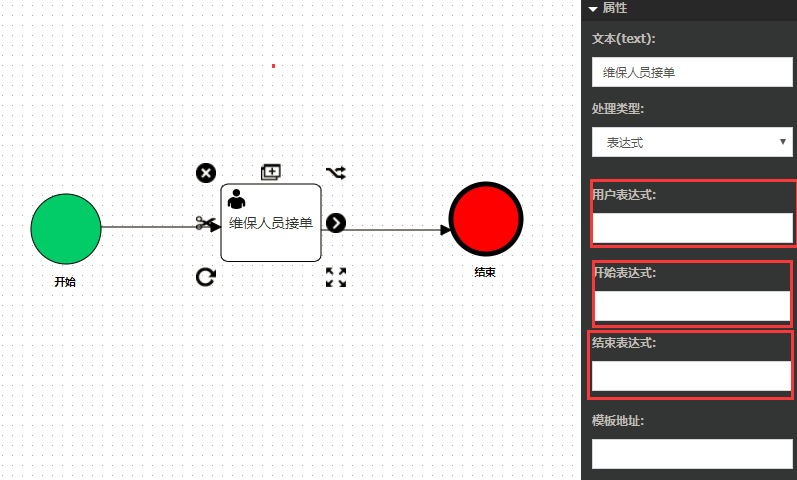
R3/PT3M：时间间隔3分钟，循环3次

### 表达式

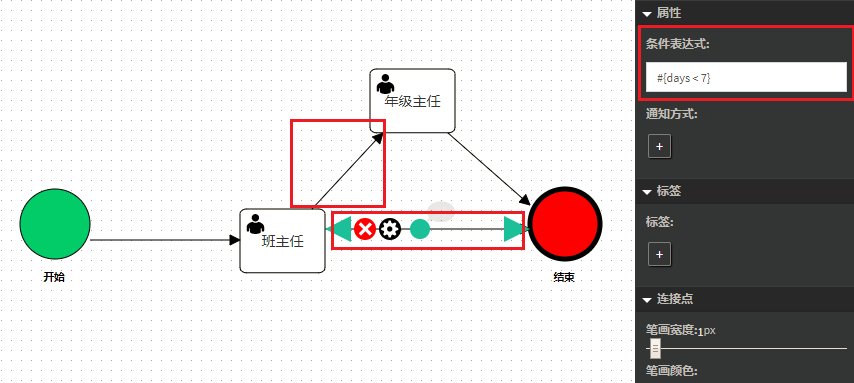
#### 支持的表达式种类

支持四种类型的表达式，如下图：

1. 开始表达式和结束表达式：在开始节点、结束节点、用户任务和服务任务中可以配置；
2. 用户表达式：只在用户任务中可以配置，在处理类型中选择“表达式”



1. 条件达表式：是配置在节点间的连线上，通常是配置在分支节点后面的连线中



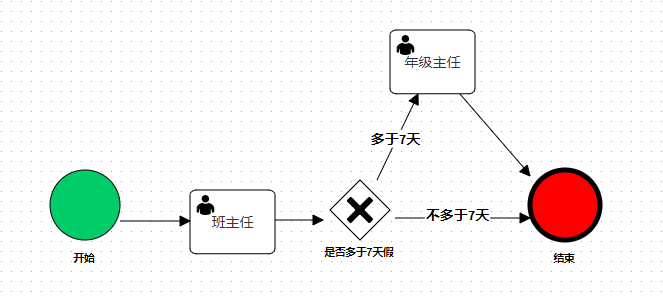
#### 表达式语法

工单发布时有默认可使用的变量：ticketNo、deviceId，faultId

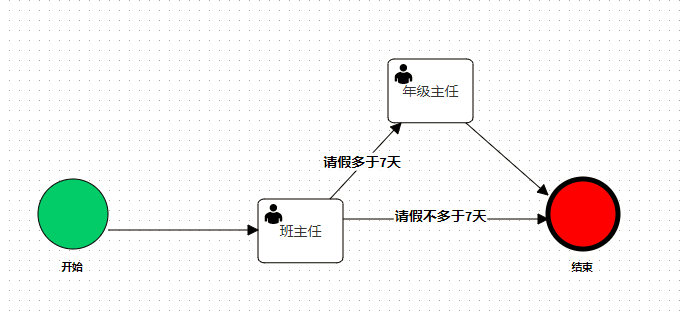
1. 用户表达式、开始表达式和结束表达式
   1. 它可以使用在执行路径之前节点的开始表达式和结束表达式中出现过的变量
   2. 可以调用提供给WEB使用的接口
   3. 赋值“:=”、变量引用“#”、接口调用“@”、语句结束“;”、对象属性的引用使用“象对.属性名”如：“ticket:=@ticketTaskService.getTicket(#ticketNo);”。
   4. 其它语法遵从Spring表达式，可百度搜索“Spring表达式”
   5. 字符串支持方法参照文档：<https://docs.oracle.com/javase/8/docs/api/java/lang/String.html>
   6. 其它java已定的类，参照文档：https://docs.oracle.com/javase/8/docs/api/
2. 条件表达式
   1. 它可以使用在执行路径之前节点的开始表达式和结束表达式中出现过的变量
   2. 可以使用执行路径前一个节点中定义的动态属性
   3. 条件表达式放在“#{…}”之间，如：#{isUnderWarranty == true}

### 分支节点及流程

分支节点用于处理路径分成多个分支执行，且只有一个分支可以执行，分条在分支节点后连线中设置，如下图

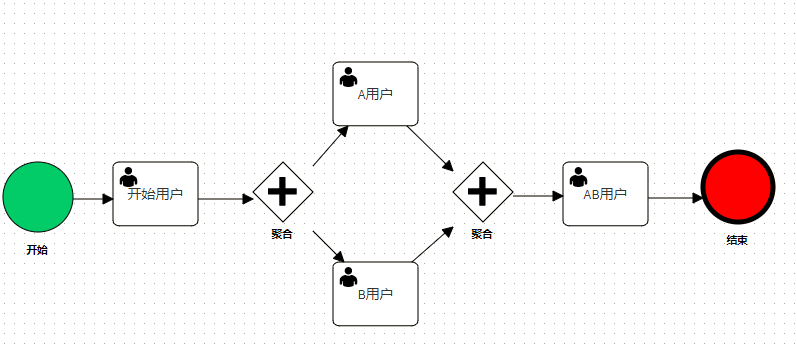


该图也可以简化成如下



### 并行节点及流程

当一条执行路径需要分成多个路径并行执行时，需要使用聚合节点。聚合节点要成对使用，聚合节点前面和后面的连线不需要设置条件表达式



### 支持js表达式

支持的范围：开始表达式、结束表达式和用户表达式

1. 安装目录psiot/app/conf/config.properties中的config.properties中的expression.type中的值修改js
2. 如果表达式中有需要在流程引擎及工单任务中使用的变量，需要存放在workflow对象中，且表达式的最后一行写：workflow;

示例：

var device = core.callService("resourceUIService","getResourceById",[params.get("deviceId")]);

print("device: " + device);

workflow.put("device",device);

workflow;