

Activiti学习笔记

工作流

- 概述
- 适用行业
- 应用
- 实现方式

Activiti7

- 概述
 - 官网
 - BPM
 - BPMN
- 使用步骤

Activiti部署

- 数据库支持
 - 创建数据库
 - 导入maven依赖
 - 修改核心配置文件
 - 添加activiti配置文件
 - 在activiti.cfg.xml中进行配置
 - 编写程序生成表
 - 运行查看结果
 - 查看创建的表
- 表结构
 - 表的命名规则和作用
 - 数据表介绍

Activiti类关系图

- 类关系图
- activiti.cfg.xml
- 流程引擎配置类
 - StandaloneProcessEngineConfiguration
 - SpringProcessEngineConfiguration
 - 创建processEngineConfiguration
- 工作流引擎创建
 - 默认创建方式
 - 其它创建方式
- Service服务接口
 - Service创建方式
- Service
 - RepositoryService
 - RuntimeService
 - TaskService
 - HistoryService
 - ManagementService

Activiti入门

- 流程
- 流程符号
 - 事件 Event
 - 活动 Activity
 - 网关 GateWay
 - 排他网关

并行网关	
包容网关	
事件网关	
流向 Flow	
idea插件	
Activiti BPMN visualizer使用示例	
Camunda Modeler	
下载	
解压	
运行	
BPMN设计界面	
流程操作	
流程定义	
流程定义部署	
操作的数据表	
启动流程实例	
操作的数据表	
任务查询	
流程任务处理	
流程定义信息查询	
流程删除	
流程资源下载	
流程历史信息的查看	
流程实例	
概述	
查询流程实例	
关联BusinessKey	
挂起、激活流程实例	
全部流程实例挂起	
单个流程实例挂起	
个人任务	
分配任务负责人	
固定分配	
表达式分配	
UEL 表达式	
UEL-value	
UEL-method	
UEL-method与UEL-value结合	
其它	
编写代码配置负责人	
监听器分配	
查询任务	
办理任务	
流程变量	
概述	
流程变量类型	
流程变量作用域	
global变量	
local变量	
使用方法	
在属性上使用UEL表达式	
在连线上使用UEL表达式	
使用Global变量控制流程	
需求	
流程定义	

- 创建实体类Evection
- 启动流程时设置变量
- 张三提交任务
- 任务办理时设置变量
- 通过当前流程实例设置
- 通过当前任务设置
- 注意事项
- 设置local流程变量
- 任务办理时设置
- 通过当前任务设置

组任务

- 概述
- 设置任务候选人
- 查询组任务
- 拾取组任务
- 查询个人待办任务
- 办理个人任务
- 归还组任务
- 任务交接

网关

- 排他网关
 - 概述
 - 流程定义
- 并行网关
 - 概述
 - 流程定义
- 包含网关
 - 概述
 - 流程定义
- 事件网关
 - 概述
 - 流程定义

Activiti整合SpringBoot

- pom依赖
- application.yml配置
- 安全框架整合配置
 - SecurityUtil类
 - DemoApplicationConfig类
- 创建Bpmn文件
- 测试

workflow

概述

workflow (Workflow), 就是通过计算机对业务流程自动化执行管理。它主要解决的是“使在多个参与者之间按照某种预定义的规则自动进行传递文档、信息或任务的过程, 从而实现某个预期的业务目标, 或者促使此目标的实现”。

workflow 概念起源于生产组织和办公自动化领域, 是针对日常工作中具有固定程序活动而提出的一个概念, 目的是通过将工作分解成定义良好的任务或角色, 按照一定的规则和过程来执行这些任务并对其进行了监控, 达到提高工作效率、更好的控制过程、增强对客户的服务、有效管理业务流程等目的。尽管 workflow 已经取得了相当的成就, 但对 workflow 的定义还没有能够统一和明确。

一个软件系统中具有 workflow 的功能, 我们把它称为 workflow 系统, 一个系统中 workflow 的功能是什么? 就是对系统的业务流程进行自动化管理, 所以 workflow 是建立在业务流程的基础上, 所以一个软件的系统核心根本上还是系统的业务流程, workflow 只是协助进行业务流程管理。即使没有 workflow 业务系统也可以开发运行, 只不过有了 workflow 可以更好的管理业务流程, 提高系统的可扩展性。

适用行业

消费品行业, 制造业, 电信服务业, 银证险等金融服务业, 物流服务业, 物业服务业, 物业管理, 大中型进出口贸易公司, 政府事业机构, 研究院所及教育服务业等, 特别是大的跨国企业和集团公司。

应用

- 关键业务流程: 订单、报价处理、合同审核、客户电话处理、供应链管理等
- 行政管理类: 出差申请、加班申请、请假申请、用车申请、各种办公用品申请、购买申请、日报周报等凡是原来手工流转处理的行政表单
- 人事管理类: 员工培训安排、绩效考评、职位变动处理、员工档案信息管理等
- 财务相关类: 付款请求、应收款处理、日常报销处理、出差报销、预算和计划申请等
- 客户服务类: 客户信息管理、客户投诉、请求处理、售后服务管理等
- 特殊服务类: ISO 系列对应流程、质量管理对应流程、产品数据信息管理、贸易公司报关处理、物流公司货物跟踪处理等各种通过表单逐步手工流转完成的任务均可应用 workflow 软件自动规范地实施

实现方式

在没有专门的 workflow 引擎之前, 我们之前为了实现流程控制, 通常的做法就是采用状态字段的值来跟踪流程的变化情况。这样不用角色的用户, 通过状态字段的取值来决定记录是否显示。

针对有权限可以查看的记录, 当前用户根据自己的角色来决定审批是否合格的操作。如果合格将状态字段设置一个值, 来代表合格; 当然如果不合格也需要设置一个值来代表不合格的情况。

这是一种最为原始的方式。通过状态字段虽然做到了流程控制，但是当我们的流程发生变更的时候，这种方式所编写的代码也要进行调整。

工作流可以做到业务流程变化之后，我们的程序可以不用改变，如果可以实现这样的效果，那么我们的业务系统的适应能力就得到了极大提升。

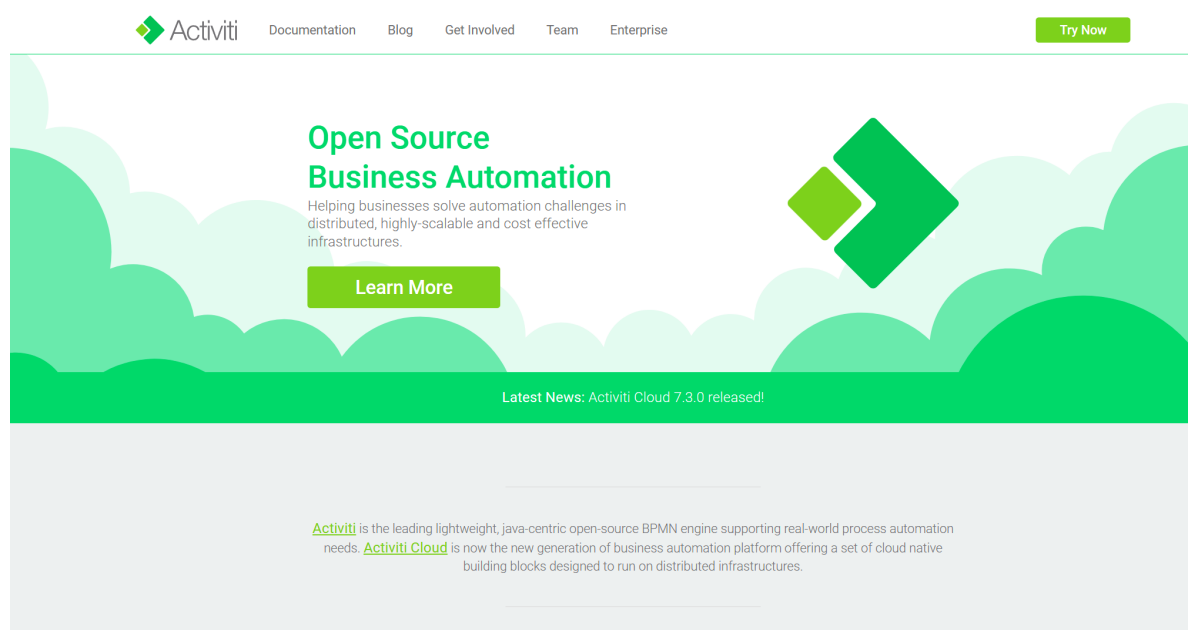
Activiti7

概述

Activiti是一个工作流引擎，activiti可以将业务系统中复杂的业务流程抽取出来，使用专门的建模语言BPMN2.0进行定义，业务流程按照预先定义的流程进行执行，实现了系统的流程由activiti进行管理，减少业务系统由于流程变更进行系统升级改造的工作量，从而提高系统的健壮性，同时也减少了系统开发维护成本。

官网

<https://www.activiti.org/>



BPM

BPM（Business Process Management），即业务流程管理，是一种规范化的构造端到端的业务流程，以持续的提高组织业务效率。常见商业管理教育如EMBA、MBA等都将BPM包含在内。

BPM软件就是根据企业中业务环境的变化，推进人与人之间、人与系统之间以及系统与系统之间的整合及调整的经营方法与解决方案的IT工具。

通过BPM软件对企业内部及外部的业务流程的整个生命周期进行建模、自动化、管理监控和优化，使企业成本降低，利润得以大幅提升。

BPM软件在企业中应用领域广泛，凡是有业务流程的地方都可以BPM软件进行管理，比如企业人事办公管理、采购流程管理、公文审批流程管理、财务管理等。

BPMN

BPMN (Business Process Model And Notation) - 业务流程模型和符号，是由BPMI (Business Process Management Initiative) 开发的一套标准的业务流程建模符号，使用BPMN提供的符号可以创建业务流程。

BPMN 是目前被各 BPM 厂商广泛接受的 BPM 标准。Activiti 就是使用 BPMN 2.0 进行流程建模、流程执行管理，它包括很多的建模符号，比如：

- Event用一个圆圈表示，它是流程中运行过程中发生的事情
- 活动用圆角矩形表示，一个流程由一个活动或多个活动组成

BPMN图形其实是通过xml表示业务流程

```
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="myProcess" name="My process" isExecutable="true">
4     <startEvent id="startevent1" name="Start"></startEvent>
5     <userTask id="usertask1" name="创建请假单"></userTask>
6     <sequenceFlow id="flow1" sourceRef="startevent1" targetRef="usertask1">
7       <sequenceFlow>
8         <userTask id="usertask2" name="部门经理审核"></userTask>
9         <sequenceFlow id="flow2" sourceRef="usertask1" targetRef="usertask2">
10          <sequenceFlow>
11            <userTask id="usertask3" name="人事复核"></userTask>
12            <sequenceFlow id="flow3" sourceRef="usertask2" targetRef="usertask3">
13              <sequenceFlow>
14                <endEvent id="endevent1" name="End"></endEvent>
15                <sequenceFlow id="flow4" sourceRef="usertask3" targetRef="endevent1">
16                  <sequenceFlow>
17                    </process>
18    <bpmndi:BPMNDiagram id="BPMNDiagram_myProcess">
19      <bpmndi:BPMNPlane bpmnElement="myProcess" id="BPMNPlane_myProcess">
20        <bpmndi:BPMNShape bpmnElement="startevent1"
21          id="BPMNShape_startevent1">
```

```

17         <omgdc:Bounds height="35.0" width="35.0" x="130.0" y="160.0">
18     </omgdc:Bounds>
19     </bpmndi:BPMNShape>
20     <bpmndi:BPMNShape bpmnElement="usertask1" id="BPMNShape_usertask1">
21         <omgdc:Bounds height="55.0" width="105.0" x="210.0" y="150.0">
22     </omgdc:Bounds>
23     </bpmndi:BPMNShape>
24     <bpmndi:BPMNShape bpmnElement="usertask2" id="BPMNShape_usertask2">
25         <omgdc:Bounds height="55.0" width="105.0" x="360.0" y="150.0">
26     </omgdc:Bounds>
27     </bpmndi:BPMNShape>
28     <bpmndi:BPMNShape bpmnElement="usertask3" id="BPMNShape_usertask3">
29         <omgdc:Bounds height="55.0" width="105.0" x="510.0" y="150.0">
30     </omgdc:Bounds>
31     </bpmndi:BPMNShape>
32     <bpmndi:BPMNShape bpmnElement="endevent1" id="BPMNShape_endevent1">
33         <omgdc:Bounds height="35.0" width="35.0" x="660.0" y="160.0">
34     </omgdc:Bounds>
35     </bpmndi:BPMNShape>
36     <bpmndi:BPMNEdge bpmnElement="flow1" id="BPMNEdge_flow1">
37         <omgdi:waypoint x="165.0" y="177.0"></omgdi:waypoint>
38         <omgdi:waypoint x="210.0" y="177.0"></omgdi:waypoint>
39     </bpmndi:BPMNEdge>
40     <bpmndi:BPMNEdge bpmnElement="flow2" id="BPMNEdge_flow2">
41         <omgdi:waypoint x="315.0" y="177.0"></omgdi:waypoint>
42         <omgdi:waypoint x="360.0" y="177.0"></omgdi:waypoint>
43     </bpmndi:BPMNEdge>
44     <bpmndi:BPMNEdge bpmnElement="flow3" id="BPMNEdge_flow3">
45         <omgdi:waypoint x="465.0" y="177.0"></omgdi:waypoint>
46         <omgdi:waypoint x="510.0" y="177.0"></omgdi:waypoint>
47     </bpmndi:BPMNEdge>
48     <bpmndi:BPMNEdge bpmnElement="flow4" id="BPMNEdge_flow4">
49         <omgdi:waypoint x="615.0" y="177.0"></omgdi:waypoint>
50         <omgdi:waypoint x="660.0" y="177.0"></omgdi:waypoint>
51     </bpmndi:BPMNEdge>
52 </bpmndi:BPMNPlane>
53 </bpmndi:BPMNDiagram>
54 </definitions>

```

使用步骤

1. **部署activiti**: Activiti是一个工作流引擎（其实就是一堆jar包API），业务系统访问(操作)activiti的接口，就可以方便的操作流程相关数据，这样就可以把工作流环境与业务系统的环境集成在一起
2. **流程定义**: 使用activiti流程建模工具(activity-designer)定义业务流程(.bpmn文件)
3. **流程定义部署**: 使用activiti提供的api把流程定义内容存储起来，在Activiti执行过程中可以查询定义的内容，Activiti执行把流程定义内容存储在数据库中

4. **启动一个流程实例**：启动一个流程实例表示开始一次业务流程的运行，在员工请假流程定义部署完成后，如果张三要请假就可以启动一个流程实例，如果李四要请假也启动一个流程实例，两个流程的执行互相不影响
5. **用户查询待办任务**：因为现在系统的业务流程已经交给activiti管理，通过activiti就可以查询当前流程执行到哪了，当前用户需要办理什么任务了，这些activiti帮我们管理了，而不需要开发人员自己编写在sql语句查询
6. **用户办理任务**：用户查询待办任务后，就可以办理某个任务，如果这个任务办理完成还需要其它用户办理，比如采购单创建后由部门经理审核，这个过程也是由activiti帮我们完成了
7. **流程结束**：当任务办理完成没有下一个任务结点了，这个流程实例就完成了

Activiti部署

数据库支持

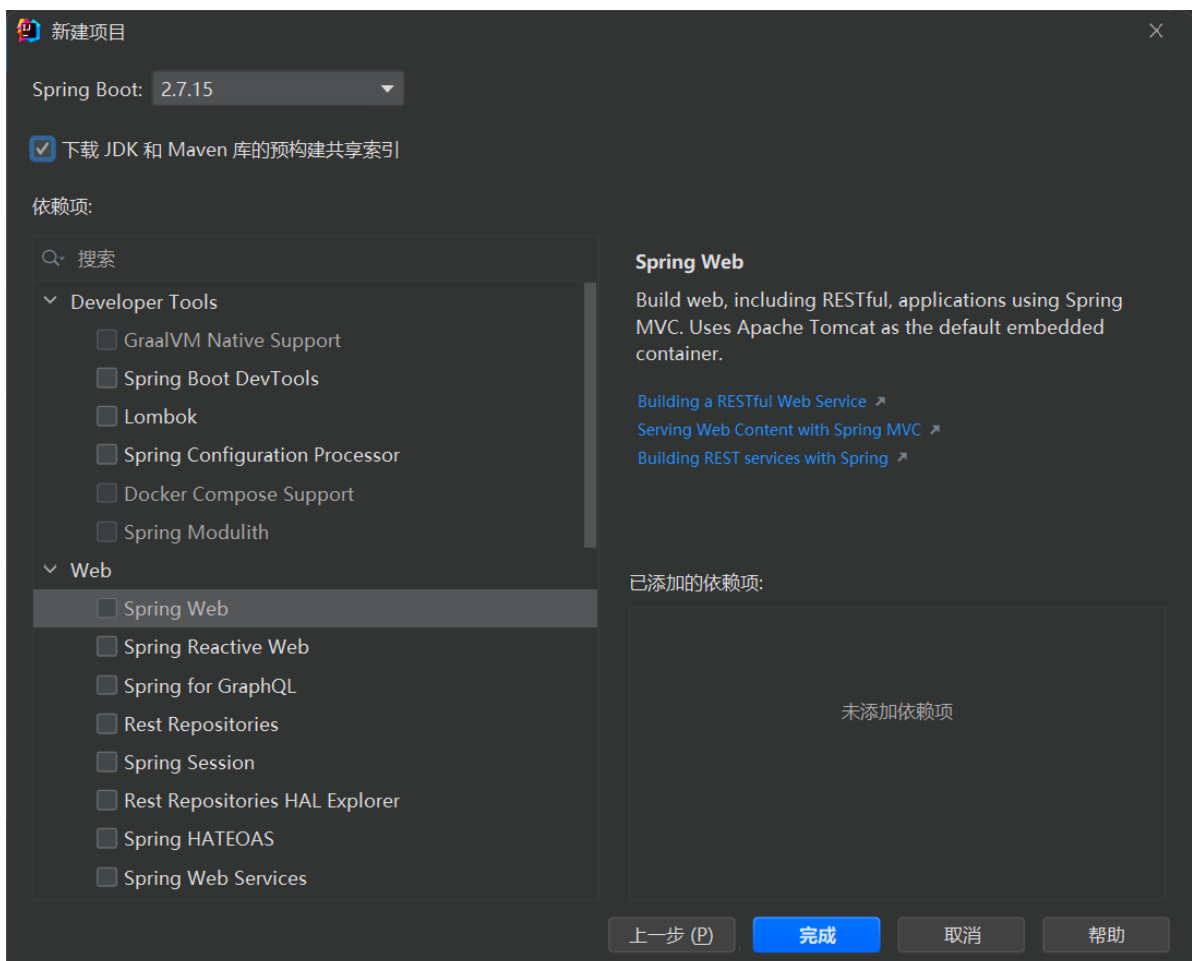
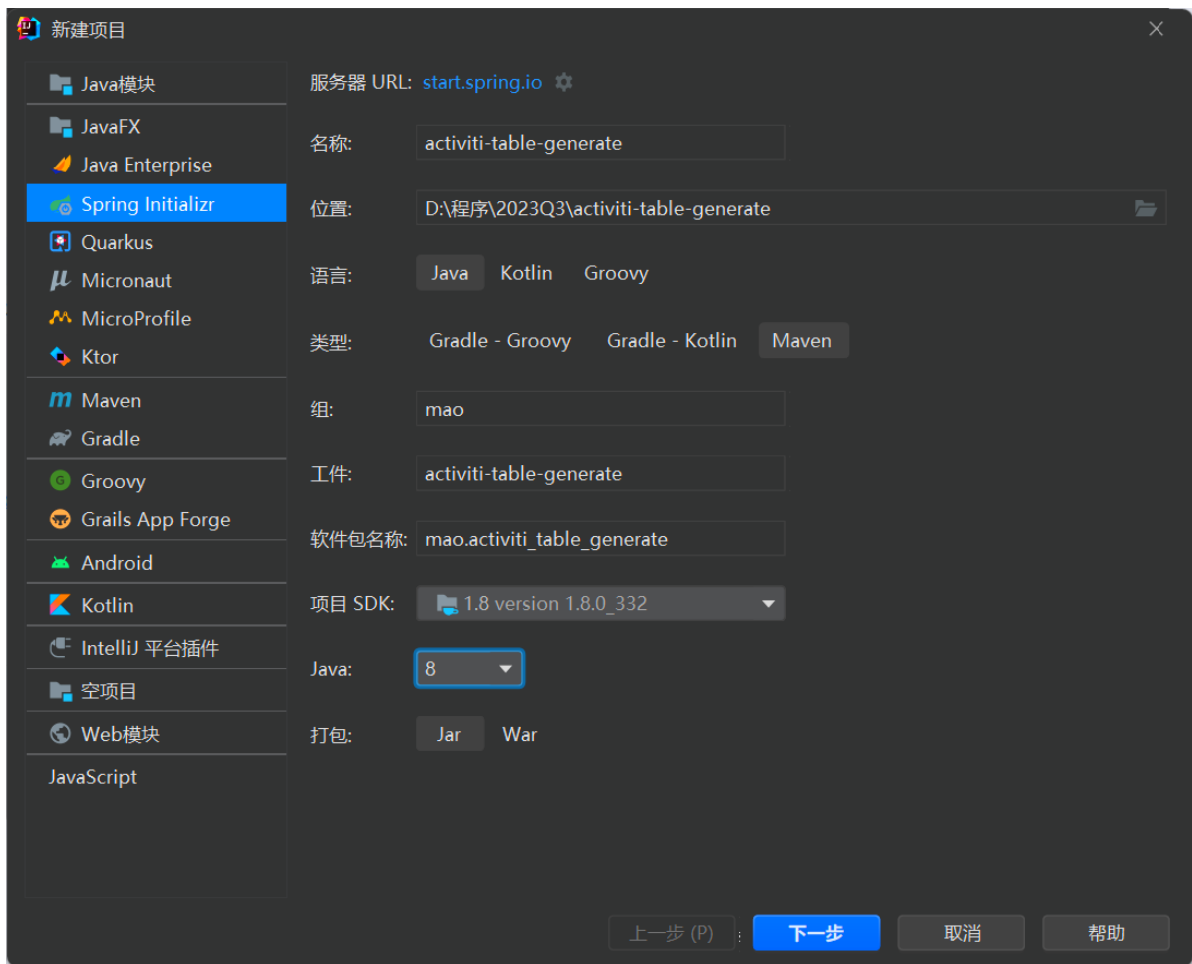
Activiti 在运行时需要数据库的支持，使用25张表，把流程定义节点内容读取到数据库表中，以供后续使用

创建数据库

创建 mysql 数据库 activiti

```
1 | CREATE DATABASE activiti DEFAULT CHARACTER SET utf8;
```

创建 java项目，目的是使用java生成数据库表



导入maven依赖

导入ProcessEngine所需要的 jar 包

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5     https://maven.apache.org/xsd/maven-4.0.0.xsd">
6     <modelVersion>4.0.0</modelVersion>
7     <parent>
8         <groupId>org.springframework.boot</groupId>
9         <artifactId>spring-boot-starter-parent</artifactId>
10        <version>2.7.1</version>
11        <relativePath/> <!-- lookup parent from repository -->
12    </parent>
13    <groupId>mao</groupId>
14    <artifactId>activiti-table-generate</artifactId>
15    <version>0.0.1-SNAPSHOT</version>
16    <name>activiti-table-generate</name>
17    <description>activiti-table-generate</description>
18    <properties>
19        <java.version>1.8</java.version>
20        <activiti.version>7.0.0.Beta1</activiti.version>
21    </properties>
22    <dependencies>
23        <dependency>
24            <groupId>org.springframework.boot</groupId>
25            <artifactId>spring-boot-starter</artifactId>
26        </dependency>
27
28        <dependency>
29            <groupId>org.springframework.boot</groupId>
30            <artifactId>spring-boot-starter-test</artifactId>
31            <scope>test</scope>
32        </dependency>
33
34        <dependency>
35            <groupId>org.activiti</groupId>
36            <artifactId>activiti-engine</artifactId>
37            <version>${activiti.version}</version>
38        </dependency>
39
40        <dependency>
41            <groupId>org.activiti</groupId>
42            <artifactId>activiti-spring</artifactId>
43            <version>${activiti.version}</version>
44        </dependency>
45        <!-- bpmn 模型处理 -->
46        <dependency>
47            <groupId>org.activiti</groupId>
48            <artifactId>activiti-bpmn-model</artifactId>
```

```

46         <version>${activiti.version}</version>
47     </dependency>
48     <!-- bpmn 转换 -->
49     <dependency>
50         <groupId>org.activiti</groupId>
51         <artifactId>activiti-bpmn-converter</artifactId>
52         <version>${activiti.version}</version>
53     </dependency>
54     <!-- bpmn json数据转换 -->
55     <dependency>
56         <groupId>org.activiti</groupId>
57         <artifactId>activiti-json-converter</artifactId>
58         <version>${activiti.version}</version>
59     </dependency>
60     <!-- bpmn 布局 -->
61     <dependency>
62         <groupId>org.activiti</groupId>
63         <artifactId>activiti-bpmn-layout</artifactId>
64         <version>${activiti.version}</version>
65     </dependency>
66     <!-- activiti 云支持 -->
67     <dependency>
68         <groupId>org.activiti.cloud</groupId>
69         <artifactId>activiti-cloud-services-api</artifactId>
70         <version>${activiti.version}</version>
71     </dependency>
72     <!--mysql依赖 spring-boot-->
73     <dependency>
74         <groupId>mysql</groupId>
75         <artifactId>mysql-connector-java</artifactId>
76         <scope>runtime</scope>
77     </dependency>
78     <!--mybatis依赖-->
79     <dependency>
80         <groupId>org.mybatis</groupId>
81         <artifactId>mybatis</artifactId>
82         <version>3.5.9</version>
83     </dependency>
84     <!-- 链接池 -->
85     <dependency>
86         <groupId>commons-dbcp</groupId>
87         <artifactId>commons-dbcp</artifactId>
88         <version>1.4</version>
89     </dependency>
90 </dependencies>
91
92 <build>
93     <plugins>
94         <plugin>
95             <groupId>org.springframework.boot</groupId>
96             <artifactId>spring-boot-maven-plugin</artifactId>
97         </plugin>
98     </plugins>
99 </build>
100

```

修改核心配置文件

application.yml:

```

1  # 设置日志级别，root表示根节点，即整体应用日志级别
2  logging:
3    # 日志输出到文件的文件名
4    file:
5      name: server.log
6    # 字符集
7    charset:
8      file: UTF-8
9    # 分文件
10   logback:
11     rollingpolicy:
12       #最大文件大小
13       max-file-size: 64KB
14       # 文件格式
15       file-name-pattern: logs/server_log/%d{yyyy/MM月/dd日/}%i.log
16   # 设置日志组
17   group:
18     # 自定义组名，设置当前组中所包含的包
19     mao_pro: mao
20   level:
21     root: info
22     # 为对应组设置日志级别
23     mao_pro: debug
24     # 日志输出格式
25   # pattern:
26   # console: "%d %clr(%p) --- [%16t] %clr(%-40.40c){cyan} : %m %n"

```

添加activiti配置文件

我们使用activiti提供的默认方式来创建mysql的表

默认方式的要求是在 resources 下创建 activiti.cfg.xml 文件，注意：默认方式目录和文件名不能修改，因为activiti的源码中已经设置，到固定的目录读取固定文件名的文件。

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xmlns:tx="http://www.springframework.org/schema/tx"
6       xsi:schemaLocation="http://www.springframework.org/schema/beans
7                           http://www.springframework.org/schema/beans/spring-
8 beans.xsd
9 http://www.springframework.org/schema/context
10 http://www.springframework.org/schema/context/spring-context.xsd
11 http://www.springframework.org/schema/tx
12 http://www.springframework.org/schema/tx/spring-tx.xsd">
13 </beans>

```

在activiti.cfg.xml中进行配置

默认方式要在在activiti.cfg.xml中bean的名字叫processEngineConfiguration，名字不可修改

在这里有两种配置方式：

- 单独配置数据源
- 不单独配置数据源

processEngineConfiguration 用来创建 ProcessEngine，在创建 ProcessEngine 时会执行数据库的操作

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xmlns:tx="http://www.springframework.org/schema/tx"
6       xsi:schemaLocation="http://www.springframework.org/schema/beans
7                           http://www.springframework.org/schema/beans/spring-
8 beans.xsd
9 http://www.springframework.org/schema/context
10 http://www.springframework.org/schema/context/spring-context.xsd
11 http://www.springframework.org/schema/tx
12 http://www.springframework.org/schema/tx/spring-tx.xsd">
13     <!-- 默认id对应的值 为processEngineConfiguration -->
14     <!-- processEngine Activiti的流程引擎 -->
15     <bean id="processEngineConfiguration"
16           class="org.activiti.engine.impl.cfg.StandaloneProcessEngineConfiguration">
17         <property name="jdbcDriver" value="com.mysql.jdbc.Driver"/>
18         <property name="jdbcUrl" value="jdbc:mysql:///activiti"/>
19         <property name="jdbcUsername" value="root"/>
20         <property name="jdbcPassword" value="123456"/>
21         <!-- activiti数据库表处理策略 -->
22         <property name="databaseSchemaUpdate" value="true"/>
23     </bean>

```

或者:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xmlns:context="http://www.springframework.org/schema/context"
5     xmlns:tx="http://www.springframework.org/schema/tx"
6     xsi:schemaLocation="http://www.springframework.org/schema/beans
7         http://www.springframework.org/schema/beans/spring-
8         beans.xsd
9         http://www.springframework.org/schema/context
10        http://www.springframework.org/schema/context/spring-context.xsd
11        http://www.springframework.org/schema/tx
12        http://www.springframework.org/schema/tx/spring-tx.xsd">
13
14     <!-- 这里可以使用 链接池 dbcp-->
15     <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
16         <property name="driverClassName" value="com.mysql.jdbc.Driver" />
17         <property name="url" value="jdbc:mysql:///activiti" />
18         <property name="username" value="root" />
19         <property name="password" value="123456" />
20         <property name="maxActive" value="3" />
21         <property name="maxIdle" value="1" />
22     </bean>
23
24     <bean id="processEngineConfiguration"
25         class="org.activiti.engine.impl.cfg.StandaloneProcessEngineConfiguration">
26         <!-- 引用数据源 上面已经设置好了-->
27         <property name="dataSource" ref="dataSource" />
28         <!-- activiti数据库表处理策略 -->
29         <property name="databaseSchemaUpdate" value="true"/>
30     </bean>
31 </beans>

```

这些xml配置也可以在@Configuration配置里编写

```

1 package mao.activiti_table_generate.config;
2
3 import org.activiti.engine.impl.cfg.StandaloneProcessEngineConfiguration;
4 import org.apache.commons.dbcp.BasicDataSource;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.context.annotation.Bean;
7 import org.springframework.context.annotation.Configuration;
8
9 import javax.sql.DataSource;
10

```

```

11  /**
12   * Project name(项目名称): activiti-table-generate
13   * Package(包名): mao.activiti_table_generate.config
14   * Class(类名): ActivitiConfig
15   * Author(作者): mao
16   * Author QQ: 1296193245
17   * GitHub: https://github.com/maomao124/
18   * Date(创建日期): 2023/9/9
19   * Time(创建时间): 20:52
20   * Version(版本): 1.0
21   * Description(描述): 无
22   */
23
24  @Configuration
25  public class ActivitiConfig
26  {
27
28      /**
29       * Data source data source.
30       *
31       * @return the data source
32       */
33      @Bean
34      public DataSource dataSource()
35      {
36          BasicDataSource basicDataSource = new BasicDataSource();
37          basicDataSource.setDriverClassName("com.mysql.jdbc.Driver");
38          basicDataSource.setUrl("jdbc:mysql:///activiti");
39          basicDataSource.setUsername("root");
40          basicDataSource.setPassword("123456");
41          return basicDataSource;
42      }
43
44
45      /**
46       * Activiti配置
47       * @param dataSource 数据源
48       * @return {@link StandaloneProcessEngineConfiguration}
49       */
50      @Bean
51      public StandaloneProcessEngineConfiguration
52      processEngineConfiguration(@Autowired DataSource dataSource)
53      {
54          StandaloneProcessEngineConfiguration
55          standaloneProcessEngineConfiguration = new
56          StandaloneProcessEngineConfiguration();
57          standaloneProcessEngineConfiguration.setDataSource(dataSource);
58          return standaloneProcessEngineConfiguration;
59      }
60  }

```

编写程序生成表

调用activiti的工具类，生成acitviti需要的数据库表

直接使用activiti提供的工具类ProcessEngines，会默认读取classpath下的activiti.cfg.xml文件，读取其中的数据库配置，创建 ProcessEngine，在创建ProcessEngine 时会自动创建表，默认的不使用配置类的形式导入

```

1 package mao.activiti_table_generate;
2
3 import org.activiti.engine.ProcessEngine;
4 import org.activiti.engine.ProcessEngines;
5 import org.slf4j.Logger;
6 import org.slf4j.LoggerFactory;
7 import org.springframework.boot.SpringApplication;
8 import org.springframework.boot.autoconfigure.SpringBootApplication;
9
10 @SpringBootApplication
11 public class ActivitiTableGenerateApplication
12 {
13     private static final Logger log =
14         LoggerFactory.getLogger(ActivitiTableGenerateApplication.class);
15
16     public static void main(String[] args)
17     {
18         SpringApplication.run(ActivitiTableGenerateApplication.class, args);
19         log.info("开始生成");
20         //使用classpath下的activiti.cfg.xml中的配置创建processEngine
21         ProcessEngine processEngine =
22             ProcessEngines.getDefaultProcessEngine();
23         log.debug(processEngine.toString());
24         log.info("生成结束");
25     }
26 }

```

运行查看结果

[illegible]


```

42 2023-09-09 21:11:03.349 INFO 25912 --- [          main]
    o.activiti.engine.impl.db.DbSqlSession : Found MySQL: majorVersion=8
    minorVersion=0
43 2023-09-09 21:11:04.658 INFO 25912 --- [          main]
    o.activiti.engine.impl.db.DbSqlSession : performing create on history with
    resource org/activiti/db/create/activiti.mysql.create.history.sql
44 2023-09-09 21:11:04.659 INFO 25912 --- [          main]
    o.activiti.engine.impl.db.DbSqlSession : Found MySQL: majorVersion=8
    minorVersion=0
45 2023-09-09 21:11:04.905 INFO 25912 --- [          main]
    o.a.engine.impl.ProcessEngineImpl : ProcessEngine default created
46 2023-09-09 21:11:04.915 INFO 25912 --- [          main]
    org.activiti.engine.ProcessEngines : initialised process engine
    default
47 2023-09-09 21:11:04.915 DEBUG 25912 --- [          main]
    m.a.ActivitiTableGenerateApplication :
    org.activiti.engine.impl.ProcessEngineImpl@5b275174
48 2023-09-09 21:11:04.915 INFO 25912 --- [          main]
    m.a.ActivitiTableGenerateApplication : 生成结束

```

查看创建的表

```

1  PS C:\Users\mao\Desktop> mysql -u root -p
2  Enter password: *****
3  welcome to the MySQL monitor.  Commands end with ; or \g.
4  Your MySQL connection id is 19
5  Server version: 8.0.32 MySQL Community Server - GPL
6
7  Copyright (c) 2000, 2023, Oracle and/or its affiliates.
8
9  Oracle is a registered trademark of Oracle Corporation and/or its
10 affiliates. Other names may be trademarks of their respective
11 owners.
12
13 Type 'help;' or '\h' for help. Type '\c' to clear the current input
    statement.
14
15 mysql> use activiti;
16 Database changed
17 mysql> show tables;
18 +-----+
19 | Tables_in_activiti |
20 +-----+
21 | act_evt_log         |
22 | act_ge_bytearray    |
23 | act_ge_property     |
24 | act_hi_actinst      |
25 | act_hi_attachment   |
26 | act_hi_comment      |

```

```

27 | act_hi_detail |
28 | act_hi_identitylink |
29 | act_hi_proclist |
30 | act_hi_taskinst |
31 | act_hi_varinst |
32 | act_procdef_info |
33 | act_re_deployment |
34 | act_re_model |
35 | act_re_procdef |
36 | act_ru_deadletter_job |
37 | act_ru_event_subscr |
38 | act_ru_execution |
39 | act_ru_identitylink |
40 | act_ru_integration |
41 | act_ru_job |
42 | act_ru_suspended_job |
43 | act_ru_task |
44 | act_ru_timer_job |
45 | act_ru_variable |
46 +-----+
47 25 rows in set (0.00 sec)
48
49 mysql> show create table act_evt_log;
50 +-----+
51 | Table | Create Table
52
53
54
55
56 |
57 +-----+
58 | act_evt_log | CREATE TABLE `act_evt_log` (
59   `LOG_NR` bigint NOT NULL AUTO_INCREMENT,
60   `TYPE` varchar(64) COLLATE utf8mb3_bin DEFAULT NULL,
61   `PROC_DEF_ID` varchar(64) COLLATE utf8mb3_bin DEFAULT NULL,
62   `PROC_INST_ID` varchar(64) COLLATE utf8mb3_bin DEFAULT NULL,
63   `EXECUTION_ID` varchar(64) COLLATE utf8mb3_bin DEFAULT NULL,

```

```

64     `TASK_ID_` varchar(64) COLLATE utf8mb3_bin DEFAULT NULL,
65     `TIME_STAMP_` timestamp(3) NOT NULL,
66     `USER_ID_` varchar(255) COLLATE utf8mb3_bin DEFAULT NULL,
67     `DATA_` longblob,
68     `LOCK_OWNER_` varchar(255) COLLATE utf8mb3_bin DEFAULT NULL,
69     `LOCK_TIME_` timestamp(3) NULL DEFAULT NULL,
70     `IS_PROCESSED_` tinyint DEFAULT '0',
71     PRIMARY KEY (`LOG_NR_`)
72 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 COLLATE=utf8mb3_bin |
73 +-----+-----+
74 1 row in set (0.00 sec)
75
76 mysql>

```

表已经在数据库里创建完成

表结构

表的命名规则和作用

Activiti 的表都以 act_ 开头

第二部分是表示表的用途的两个字母标识。用途也和服务的 API 对应

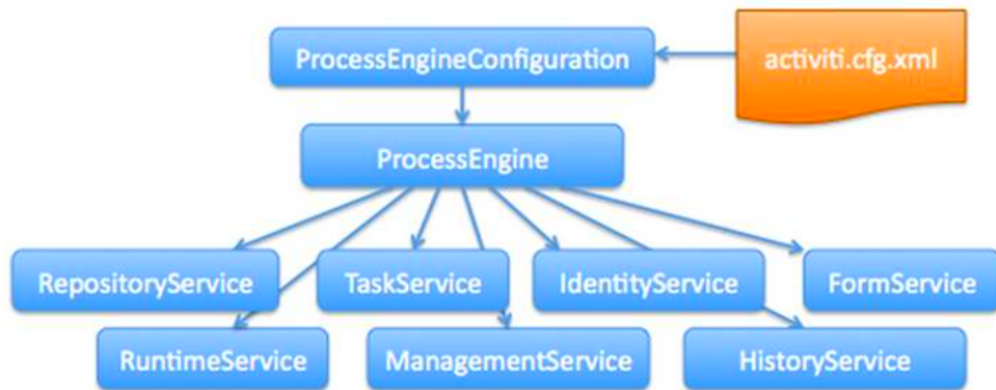
- **ACT_RE**：'RE'表示 repository。这个前缀的表包含了流程定义和流程静态资源（图片，规则，等等）。
- **ACT_RU**：'RU'表示 runtime。这些运行时的表，包含流程实例，任务，变量，异步任务，等运行中的数据。Activiti 只在流程实例执行过程中保存这些数据，在流程结束时就会删除这些记录。这样运行时表可以一直很小速度很快。
- **ACT_HI**：'HI'表示 history。这些表包含历史数据，比如历史流程实例，变量，任务等等。
- **ACT_GE**：GE 表示 general。通用数据，用于不同场景下

数据表介绍

表分类	表名	解释
一般数据		
	[ACT_GE_BYTEARRAY]	通用的流程定义和流程资源
	[ACT_GE_PROPERTY]	系统相关属性
流程历史记录		
	[ACT_HI_ACTINST]	历史的流程实例
	[ACT_HI_ATTACHMENT]	历史的流程附件
	[ACT_HI_COMMENT]	历史的说明性信息
	[ACT_HI_DETAIL]	历史的流程运行中的细节信息
	[ACT_HI_IDENTITYLINK]	历史的流程运行过程中用户关系
	[ACT_HI_PROCINST]	历史的流程实例
	[ACT_HI_TASKINST]	历史的任务实例
	[ACT_HI_VARINST]	历史的流程运行中的变量信息
流程定义表		
	[ACT_RE_DEPLOYMENT]	部署单元信息
	[ACT_RE_MODEL]	模型信息
	[ACT_RE_PROCDEF]	已部署的流程定义
运行实例表		
	[ACT_RU_EVENT_SUBSCR]	运行时事件
	[ACT_RU_EXECUTION]	运行时流程执行实例
	[ACT_RU_IDENTITYLINK]	运行时用户关系信息，存储任务节点与参与者的相关信息
	[ACT_RU_JOB]	运行时作业
	[ACT_RU_TASK]	运行时任务
	[ACT_RU_VARIABLE]	运行时变量表

Activiti类关系图

类关系图



IdentityService, FormService两个Service都已经删除了，不是继承关系

```
package org.activiti.engine;

import java.io.InputStream;
import java.util.HashMap;
import java.util.Map;
import javax.sql.DataSource;
import org.activiti.engine.cfg.MailServerInfo;
import org.activiti.engine.impl.asyncexecutor.AsyncExecutor;
import org.activiti.engine.impl.cfg.BeansConfigurationHelper;
import org.activiti.engine.impl.cfg.StandaloneInMemProcessEngineConfiguration;
import org.activiti.engine.impl.cfg.StandaloneProcessEngineConfiguration;
import org.activiti.engine.impl.history.HistoryLevel;
import org.activiti.engine.impl.persistence.entity.integration.IntegrationContextService;
import org.activiti.engine.integration.IntegrationContextService;
import org.activiti.engine.runtime.Clock;
import org.activiti.runtime.api.identity.UserGroupManager;

public abstract class ProcessEngineConfiguration {
    public static final String DB_SCHEMA_UPDATE_FALSE = "false";
    public static final String DB_SCHEMA_UPDATE_CREATE_DROP = "create-drop";
    public static final String DB_SCHEMA_UPDATE_TRUE = "true";
```

activiti.cfg.xml

activiti的引擎配置文件，包括：ProcessEngineConfiguration的定义、数据源定义、事务管理等，此文件其实就是一个spring配置文件

流程引擎配置类

流程引擎的配置类（ProcessEngineConfiguration），通过ProcessEngineConfiguration可以创建工作流引擎ProceccEngine

常用的实现类如下：

- SpringProcessEngineConfiguration
- StandaloneProcessEngineConfiguration

```
import org.activiti.runtime.api.identity.UserGroupManager;

public abstract class ProcessEngineConfiguration {
    public static final String DB_SCHEMA_UPDATE_FALSE = "false";
    public static final String DB_SCHEMA_UPDATE_CREATE_DROP = "create-drop";
    public static final String DB_SCHEMA_UPDATE_TRUE = "true";
    public static final String NO_TENANT_ID = "";
    protected String processEngineName = "default";
    protected int idBlockSize = 2500;
    protected String history;
    protected boolean asyncExecutorActivate;
    protected String mailServerHost;
```

```
选择ProcessEngineConfiguration的子类
JtaProcessEngineConfiguration (org.activiti.engine.impl.cfg)
MultiSchemaMultiTenantProcessEngineConfiguration (org.activiti.engine.impl.cfg.multitenant)
ProcessEngineConfigurationImpl (org.activiti.engine.impl.cfg)
SpringProcessEngineConfiguration (org.activiti.spring)
StandaloneInMemProcessEngineConfiguration (org.activiti.engine.impl.cfg)
StandaloneProcessEngineConfiguration (org.activiti.engine.impl.cfg)
```

```
package org.activiti.spring;

import ...

public class SpringProcessEngineConfiguration extends ProcessEngineConfigurationImpl implements Ap
    protected PlatformTransactionManager transactionManager;
    protected String deploymentName = "SpringAutoDeployment";
    protected Resource[] deploymentResources = new Resource[0];
    protected String deploymentMode = "default";
    protected ApplicationContext applicationContext;
    protected Integer transactionSynchronizationAdapterOrder = null;
    private Collection<AutoDeploymentStrategy> deploymentStrategies = new ArrayList();
```

```
import org.activiti.engine.impl.interceptor.CommandInterceptor;

public class StandaloneProcessEngineConfiguration extends ProcessEngineConfigurationImpl {
    public StandaloneProcessEngineConfiguration() {
    }

    public CommandInterceptor createTransactionInterceptor() { return null; }
}
```

StandaloneProcessEngineConfiguration

使用StandaloneProcessEngineConfigurationActiviti可以单独运行，来创建ProcessEngine，Activiti会自己处理事务

通常在activiti.cfg.xml配置文件中定义一个id为 `processEngineConfiguration` 的bean

SpringProcessEngineConfiguration

通过 `org.activiti.spring.SpringProcessEngineConfiguration` 与Spring整合

xml配置示例：

```
1 <beans xmlns="http://www.springframework.org/schema/beans"
2     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:mvc="http://www.springframework.org/schema/mvc"
3     xmlns:context="http://www.springframework.org/schema/context"
4     xmlns:aop="http://www.springframework.org/schema/aop"
   xmlns:tx="http://www.springframework.org/schema/tx"
5     xsi:schemaLocation="http://www.springframework.org/schema/beans
6         http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
7         http://www.springframework.org/schema/mvc
8         http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd
9         http://www.springframework.org/schema/context
10        http://www.springframework.org/schema/context/spring-context-3.1.xsd
11        http://www.springframework.org/schema/aop
12        http://www.springframework.org/schema/aop/spring-aop-3.1.xsd
13        http://www.springframework.org/schema/tx
14        http://www.springframework.org/schema/tx/spring-tx-3.1.xsd ">
15     <!-- 工作流引擎配置bean -->
16     <bean id="processEngineConfiguration"
   class="org.activiti.spring.SpringProcessEngineConfiguration">
17         <!-- 数据源 -->
18         <property name="dataSource" ref="dataSource" />
19         <!-- 使用spring事务管理器 -->
20         <property name="transactionManager" ref="transactionManager" />
21         <!-- 数据库策略 -->
```



```

22     <property name="databaseSchemaUpdate" value="drop-create" />
23     <!-- activiti的定时任务关闭 -->
24     <property name="jobExecutorActivate" value="false" />
25 </bean>
26 <!-- 流程引擎 -->
27 <bean id="processEngine"
class="org.activiti.spring.ProcessEngineFactoryBean">
28     <property name="processEngineConfiguration"
ref="processEngineConfiguration" />
29 </bean>
30 <!-- 资源服务service -->
31 <bean id="repositoryService" factory-bean="processEngine"
32     factory-method="getRepositoryService" />
33 <!-- 流程运行service -->
34 <bean id="runtimeService" factory-bean="processEngine"
35     factory-method="getRuntimeService" />
36 <!-- 任务管理服务 -->
37 <bean id="taskService" factory-bean="processEngine"
38     factory-method="getTaskService" />
39 <!-- 历史管理服务 -->
40 <bean id="historyService" factory-bean="processEngine" factory-
method="getHistoryService" />
41 <!-- 用户管理服务 -->
42 <bean id="identityService" factory-bean="processEngine" factory-
method="getIdentityService" />
43 <!-- 引擎管理服务 -->
44 <bean id="managementService" factory-bean="processEngine" factory-
method="getManagementService" />
45 <!-- 数据源 -->
46 <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
47     <property name="driverClassName" value="com.mysql.jdbc.Driver" />
48     <property name="url" value="jdbc:mysql://localhost:3306/activiti" />
49     <property name="username" value="root" />
50     <property name="password" value="mysql" />
51     <property name="maxActive" value="3" />
52     <property name="maxIdle" value="1" />
53 </bean>
54 <!-- 事务管理器 -->
55 <bean id="transactionManager"
56
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
57     <property name="dataSource" ref="dataSource" />
58 </bean>
59 <!-- 通知 -->
60 <tx:advice id="txAdvice" transaction-manager="transactionManager">
61     <tx:attributes></tx:attributes>
62     <!-- 传播行为 -->
63     <tx:method name="save*" propagation="REQUIRED" />
64     <tx:method name="insert*" propagation="REQUIRED" />
65     <tx:method name="delete*" propagation="REQUIRED" />
66     <tx:method name="update*" propagation="REQUIRED" />
67     <tx:method name="find*" propagation="SUPPORTS" read-only="true"
/>
68     <tx:method name="get*" propagation="SUPPORTS" read-only="true" />
69 </tx:attributes>

```

```

70     </tx:advice>
71     <!-- 切面，根据具体项目修改切点配置 -->
72     <aop:config proxy-target-class="true">
73         <aop:advisor advice-ref="txAdvice" pointcut="execution(* 业务包
名.service.impl.*(..))*" />
74     </aop:config>
75 </beans>

```

创建processEngineConfiguration

方式一：

```

1 ProcessEngineConfiguration configuration =
  ProcessEngineConfiguration.createProcessEngineConfigurationFromResource("acti
viti.cfg.xml")

```

上边的代码要求activiti.cfg.xml中必须有一个processEngineConfiguration的bean

方式二：

```

1 ProcessEngineConfiguration.createProcessEngineConfigurationFromResource(String
resource, String beanName);

```

工作流引擎创建

工作流引擎（ProcessEngine），相当于一个门面接口，通过ProcessEngineConfiguration创建processEngine，然后通过ProcessEngine创建各个service接口

默认创建方式

将activiti.cfg.xml文件名及路径固定，且activiti.cfg.xml文件中有 processEngineConfiguration的配置

```

1 //直接使用工具类 ProcessEngines，使用classpath下的activiti.cfg.xml中的配置创建
processEngine
2 ProcessEngine processEngine = ProcessEngines.getDefaultProcessEngine();
3 System.out.println(processEngine);

```

其它创建方式

```
1 //先构建ProcessEngineConfiguration
2 ProcessEngineConfiguration configuration =
  ProcessEngineConfiguration.createProcessEngineConfigurationFromResource("acti
  viti.cfg.xml");
3 //通过ProcessEngineConfiguration创建ProcessEngine, 此时会创建数据库
4 ProcessEngine processEngine = configuration.buildProcessEngine();
```

Service服务接口

Service是工作流引擎提供用于进行工作流部署、执行、管理的服务接口，我们使用这些接口可以就是操作服务对应的数据表

Service创建方式

通过ProcessEngine创建Service

```
1 RuntimeService runtimeService = processEngine.getRuntimeService();
2 RepositoryService repositoryService = processEngine.getRepositoryService();
3 TaskService taskService = processEngine.getTaskService();
```

Service

有如下的Service

service名称	service作用
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
ManagerService	activiti的引擎管理类

RepositoryService

是activiti的资源管理类，提供了管理和控制流程发布包和流程定义的操作。使用工作流建模工具设计的业务流程图需要使用此service将流程定义文件的内容部署到计算机

除了部署流程定义以外还可以查询引擎中的发布包和流程定义

暂停或激活发布包，对应全部和特定流程定义。暂停意味着它们不能再执行任何操作了，激活是对应的反向操作。获得多种资源，像是包含在发布包里的文件，或引擎自动生成的流程图。

RuntimeService

Activiti的流程运行管理类。可以从这个服务类中获取很多关于流程执行相关的信息

TaskService

Activiti的任务管理类。可以从这个类中获取任务的信息

HistoryService

Activiti的历史管理类，可以查询历史信息，执行流程时，引擎会保存很多数据（根据配置），比如流程实例启动时间，任务的参与者，完成任务的时间，每个流程实例的执行路径，等等。这个服务主要通过查询功能来获得这些数据

ManagementService

Activiti的引擎管理类，提供了对Activiti流程引擎的管理和维护功能，这些功能不在工作流驱动的应用程序中使用，主要用于Activiti系统的日常维护

Activiti入门

流程

1. 定义流程，按照BPMN的规范，使用流程定义工具，用**流程符号**把整个流程描述出来
2. 部署流程，把画好的流程定义文件，加载到数据库中，生成表的数据
3. 启动流程，使用java代码来操作数据库表中的内容

流程符号

事件 Event



活动 Activity

活动是工作或任务的一个通用术语。一个活动可以是一个任务，还可以是一个当前流程的子处理流程



网关 GateWay



排他网关

只有一条路径会被选择。流程执行到该网关时，按照输出流的顺序逐个计算，当条件的计算结果为true时，继续执行当前网关的输出流

如果多条线路计算结果都是 true，则会执行第一个值为 true 的线路。如果所有网关计算结果没有true，则引擎会抛出异常

排他网关需要和条件顺序流结合使用，default 属性指定默认顺序流，当所有的条件不满足时会执行默认顺序流

并行网关

所有路径会被同时选择

并行执行所有输出顺序流，为每一条顺序流创建一个并行执行线路

所有从并行网关拆分并执行完成的线路均在此等候，直到所有的线路都执行完成才继续向下执行

包容网关

可以同时执行多条线路，也可以在网关上设置条件

计算每条线路上的表达式，当表达式计算结果为true时，创建一个并行线路并继续执行

所有从并行网关拆分并执行完成的线路均在此等候，直到所有的线路都执行完成才继续向下执行

事件网关

专门为中间捕获事件设置的，允许设置多个输出流指向多个不同的中间捕获事件。当流程执行到事件网关后，流程处于等待状态，需要等待抛出事件才能将等待状态转换为活动状态

流向 Flow

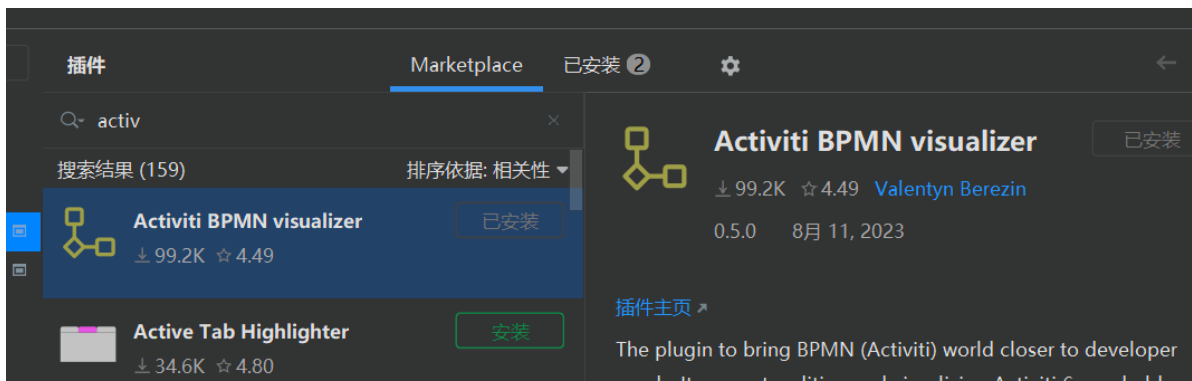
流是连接两个流程节点的连线



idea插件

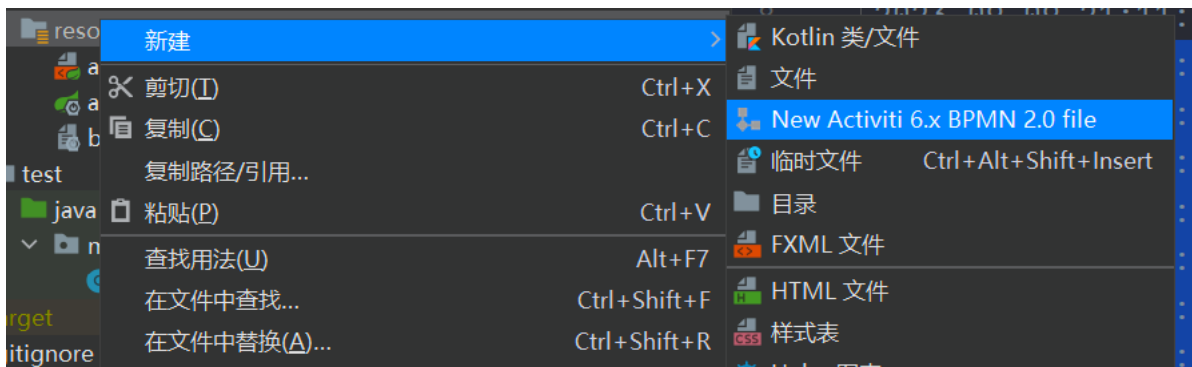
Activiti插件actiBPM在新版的idea 2020及以上版本中已经不支持

Activiti BPMN visualizer是一款支持编辑和游览 workflow 设计图的 idea 插件，但是它对 workflow 设计中的网关设计支持并不太友好



Activiti BPMN visualizer使用示例

在资源路径下创建文件



示例如下：

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
5   xmlns:activiti="http://activiti.org/bpmn"
6   xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
7   xmlns:omgdc="http://www.omg.org/spec/DD/20100524/DC"
8   xmlns:omgdi="http://www.omg.org/spec/DD/20100524/DI"
9   typeLanguage="http://www.w3.org/2001/XMLSchema"
10  expressionLanguage="http://www.w3.org/1999/XPath"
11  targetNamespace="http://www.activiti.org/processdef">
  <process id="test" name="test" isExecutable="true">
    </process>
    <bpmndi:BPMNDiagram id="BPMNDiagram_test">
      <bpmndi:BPMNPlane bpmnElement="test" id="BPMNPlane_test">
```

```
12         </bpmndi:BPMNPlane>
13     </bpmndi:BPMNDiagram>
14 </definitions>
15
```

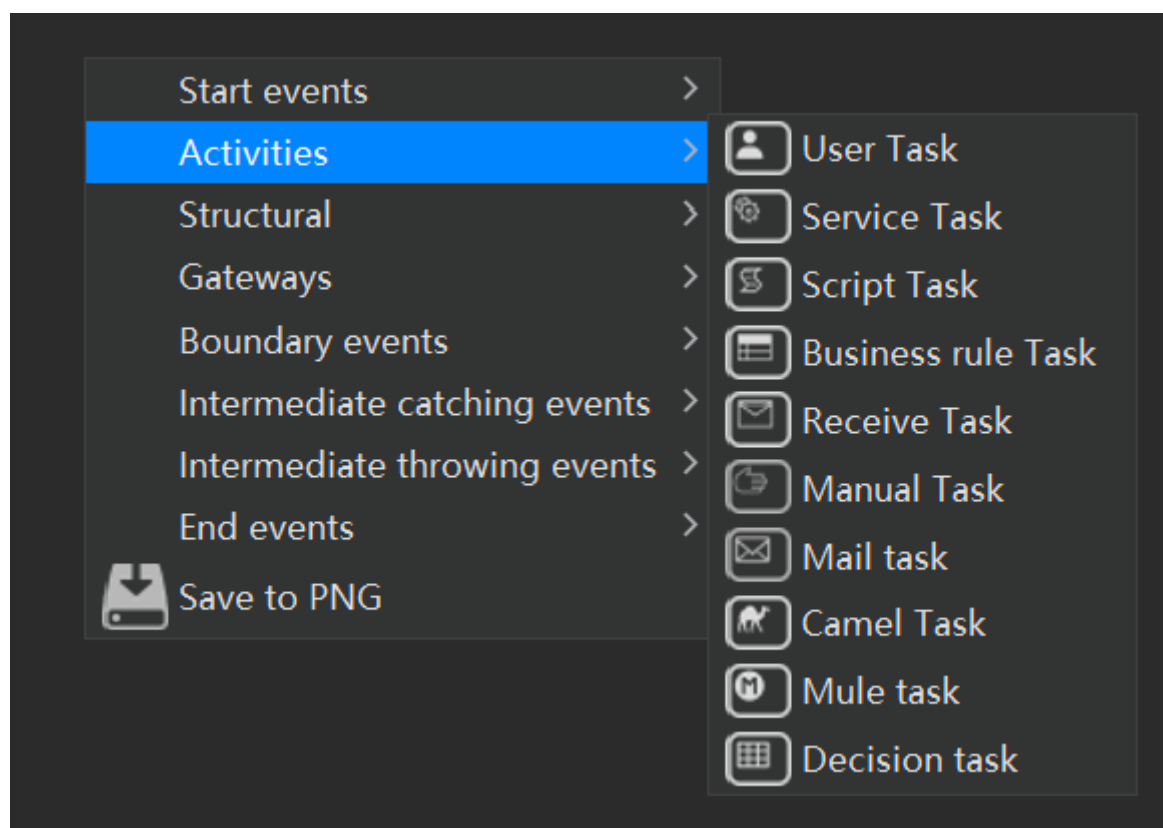
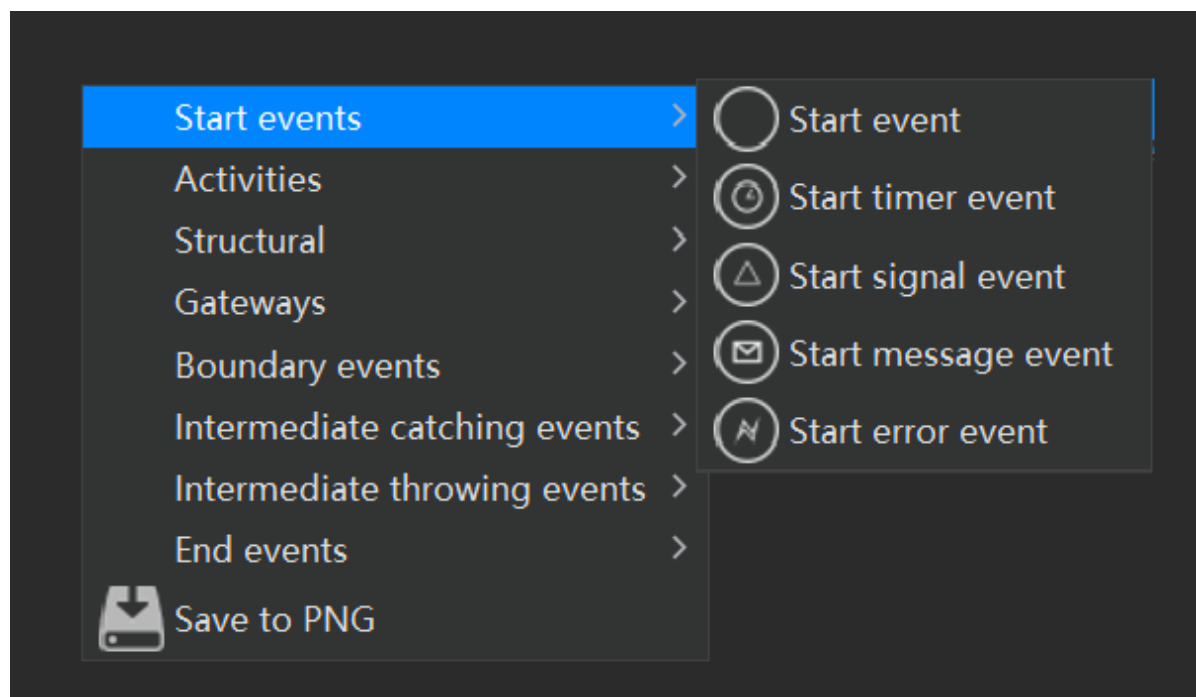
右键点击

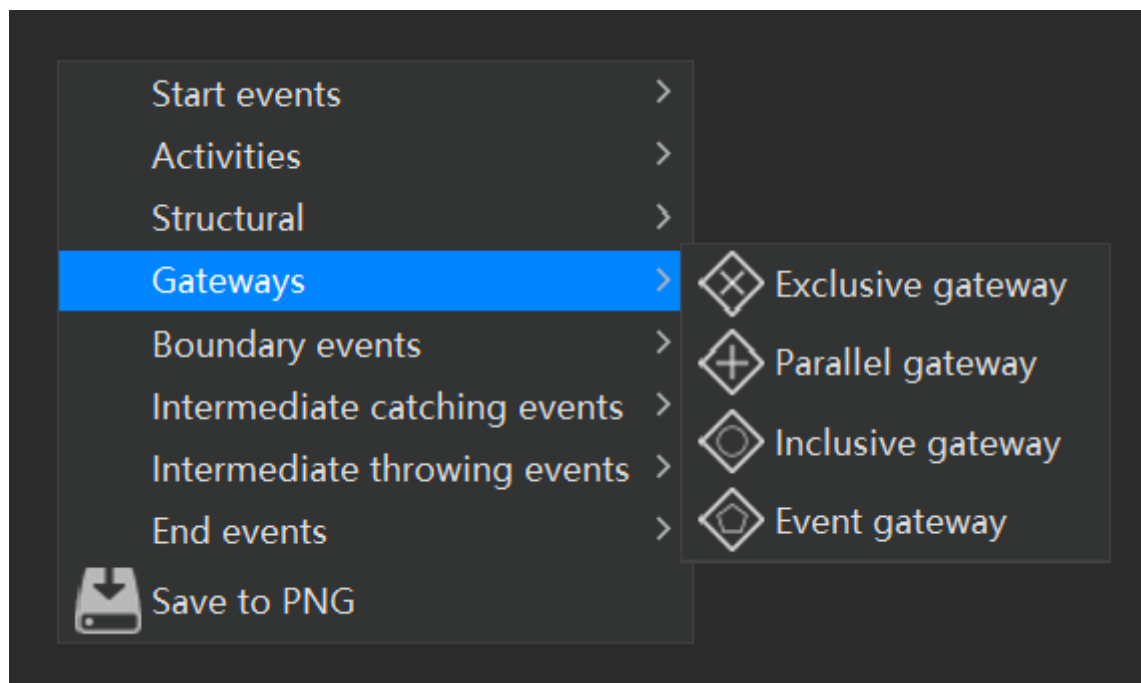
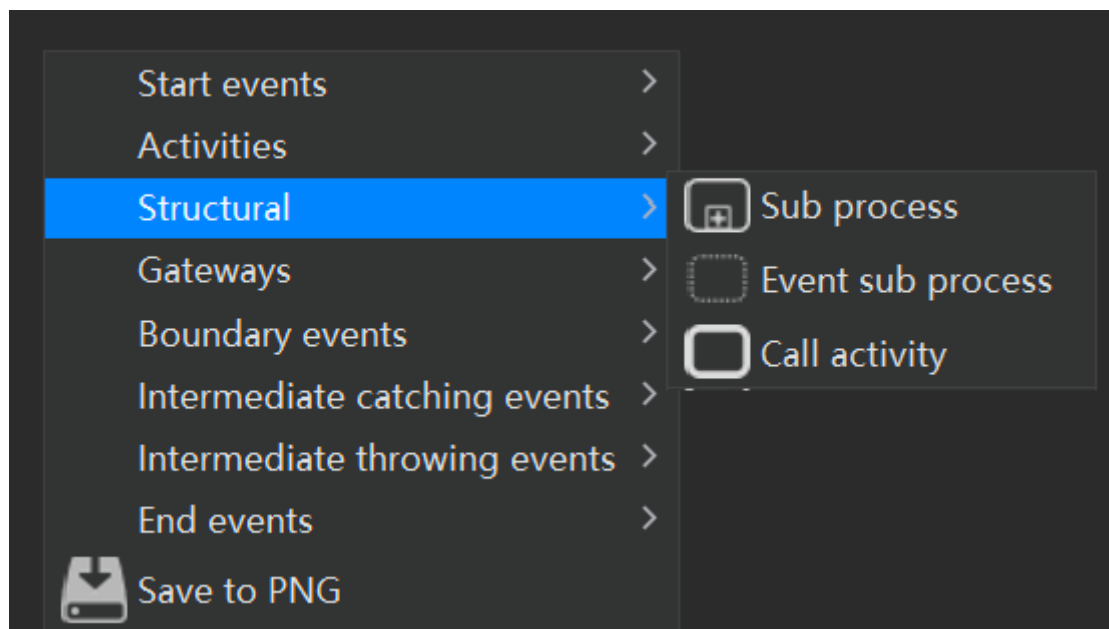

```
bpmnElement="test" id="BPMNPlane_test">
```

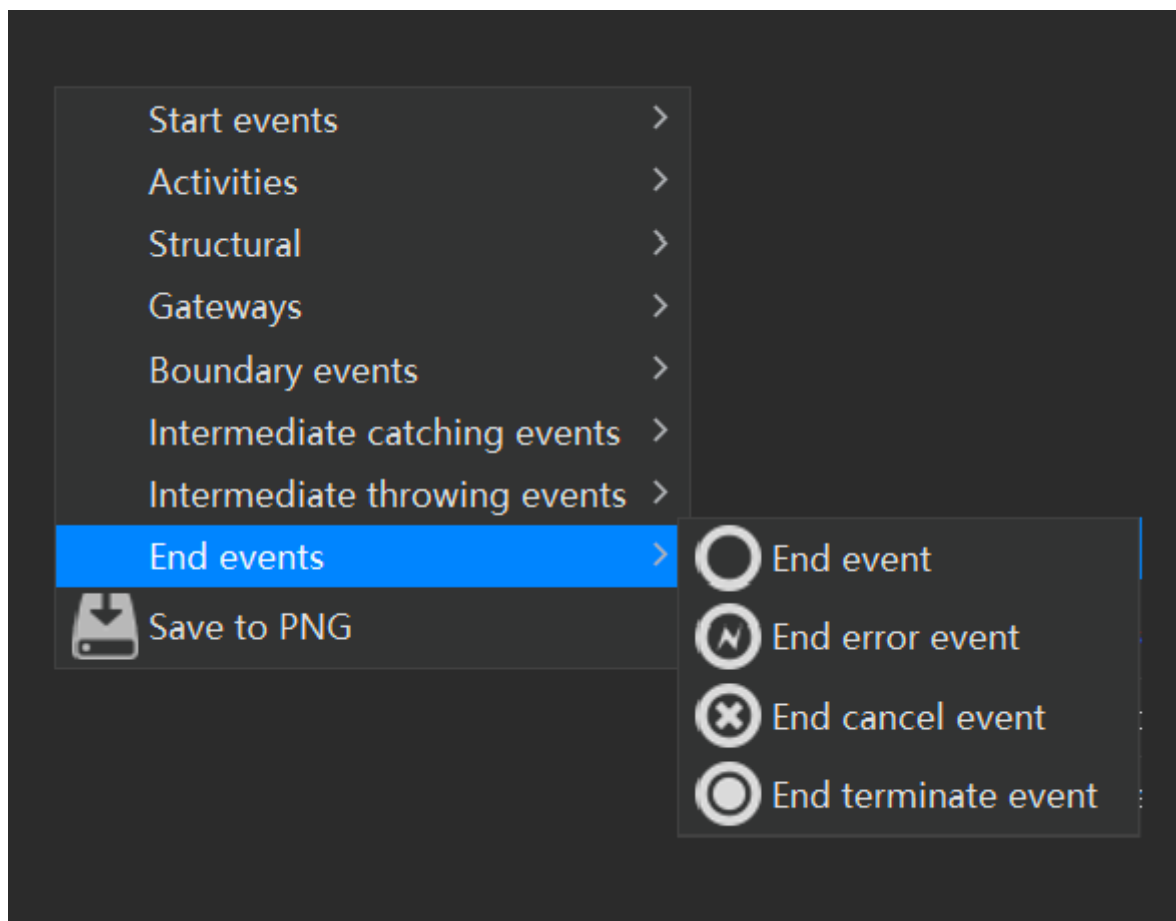
	显示上下文操作	Alt+Enter
	粘贴(P)	Ctrl+V
	复制/粘贴特殊	>
	列选择模式(M)	Alt+Shift+Insert
	查找用法(U)	Alt+F7
	重构(R)	>
	折叠	>
	分析(Z)	>
	转到	>
	生成...	Alt+Insert
	Run Maven	>
	Debug Maven	>
	Open Terminal at the Current Maven Module Path	
	打开于	>
	验证(V)	
	本地历史记录(H)	>
	Git(G)	>
	与剪贴板比较(B)	
	从 XML 文件生成 DTD(X)	
	从 XML 文件生成 XSD 架构...	
	Create Gist...	
	创建 Gist...	
	View BPMN (Activiti) Diagram	
	评估 XPath...	Ctrl+Alt+X, E
	显示唯一 XPath	Ctrl+Alt+X, P

BPMN-Activiti-Diagram test.bpmn20.xml

ID	test
Name	test
Documentation	







Camunda Modeler

Camunda Modeler是Camunda官方提供的建模器

是一个流程图的绘图工具。支持三种协议类型流程文件： BPMN diagram、 DMN diagram、 Form。

下载

<https://camunda.com/download/modeler/>

Open Source Desktop Modeler

Supports: BPMN, DMN, Forms

Version: 5.15.0

Release Date: September 12, 2023

Platform: Camunda 8 and below

License: Camunda Modeler is licensed under the MIT license. Third-party libraries included are distributed under their respective licenses ([view third-party notices](#)). No warranties: Camunda Modeler comes without any guarantees.

Download:

Mac OS .zip

Windows 64bit

Linux 64bit




















Mac OS .dmg

Windows 32bit

解压

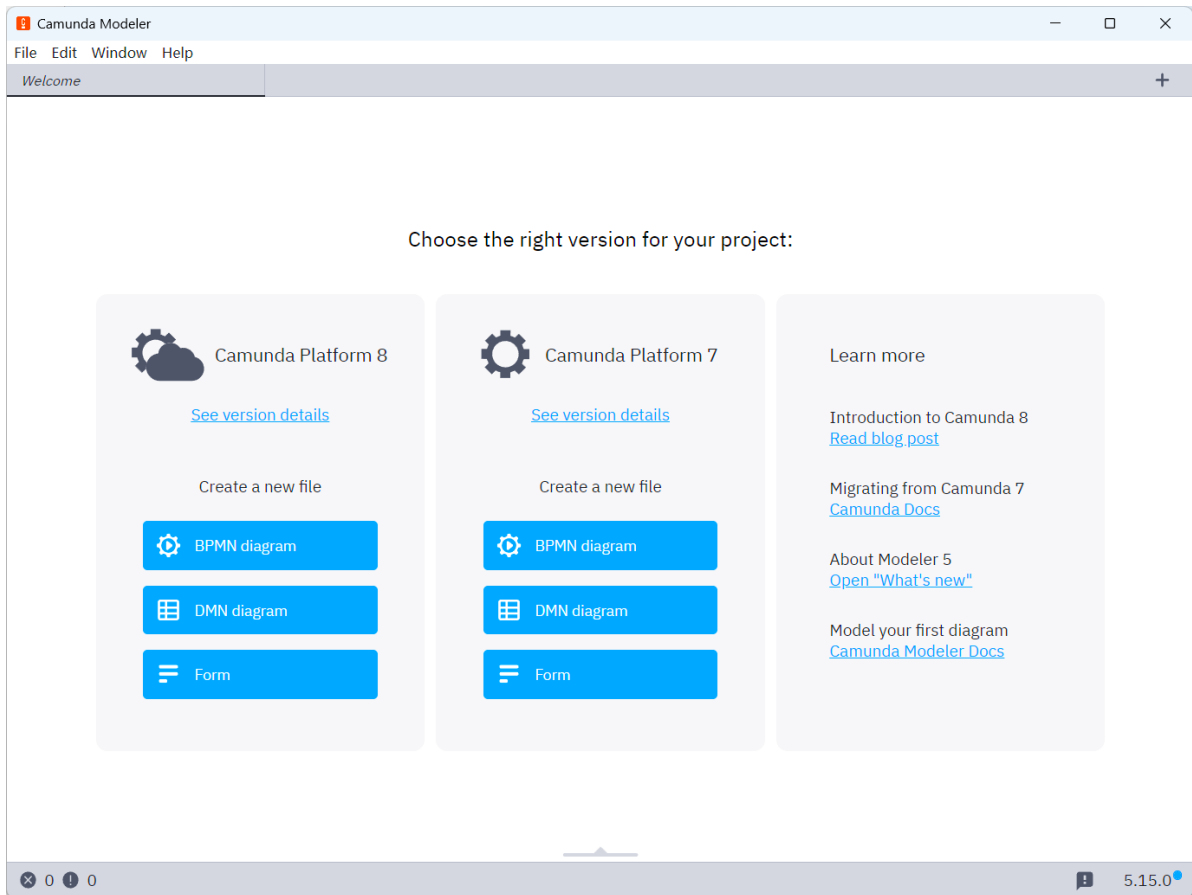
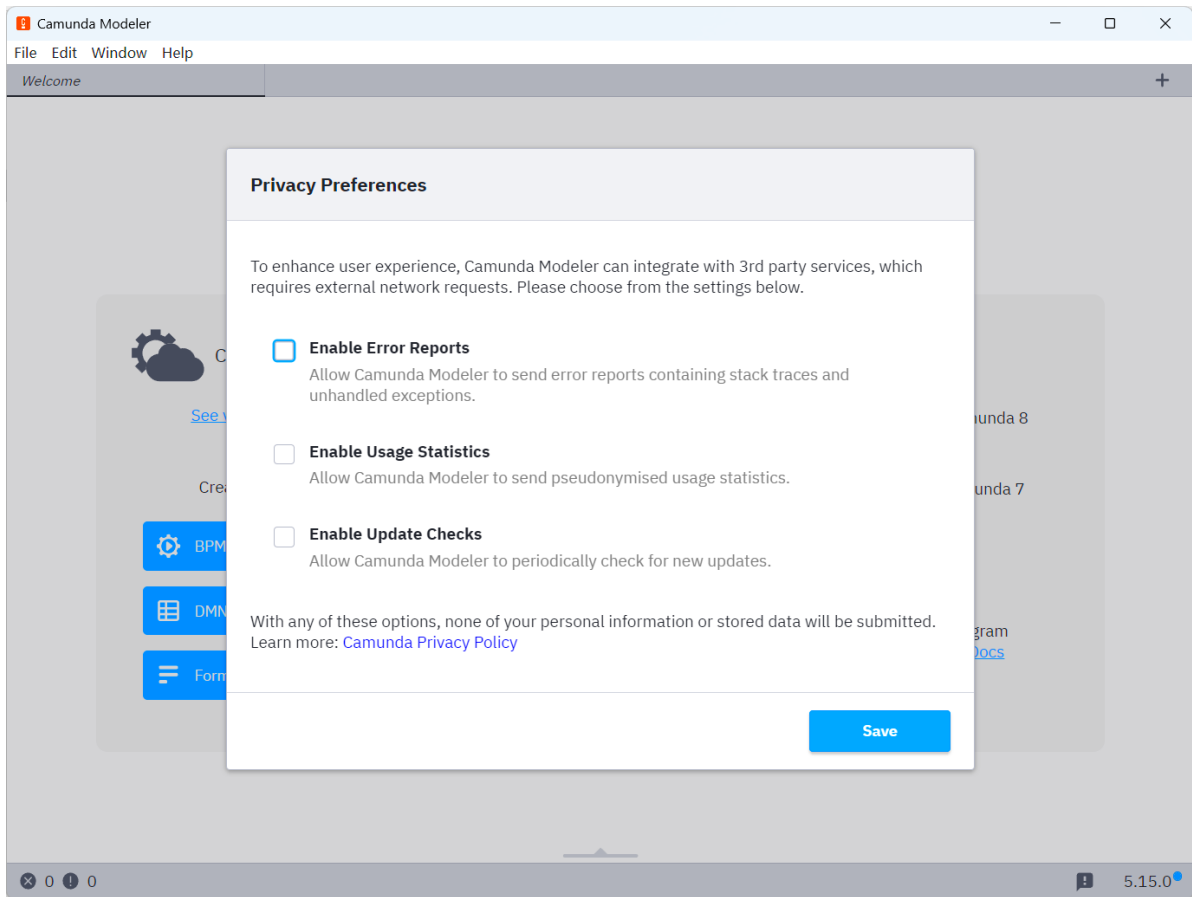
解压后安装路径必须是非中文路径

解压后的目录如下：

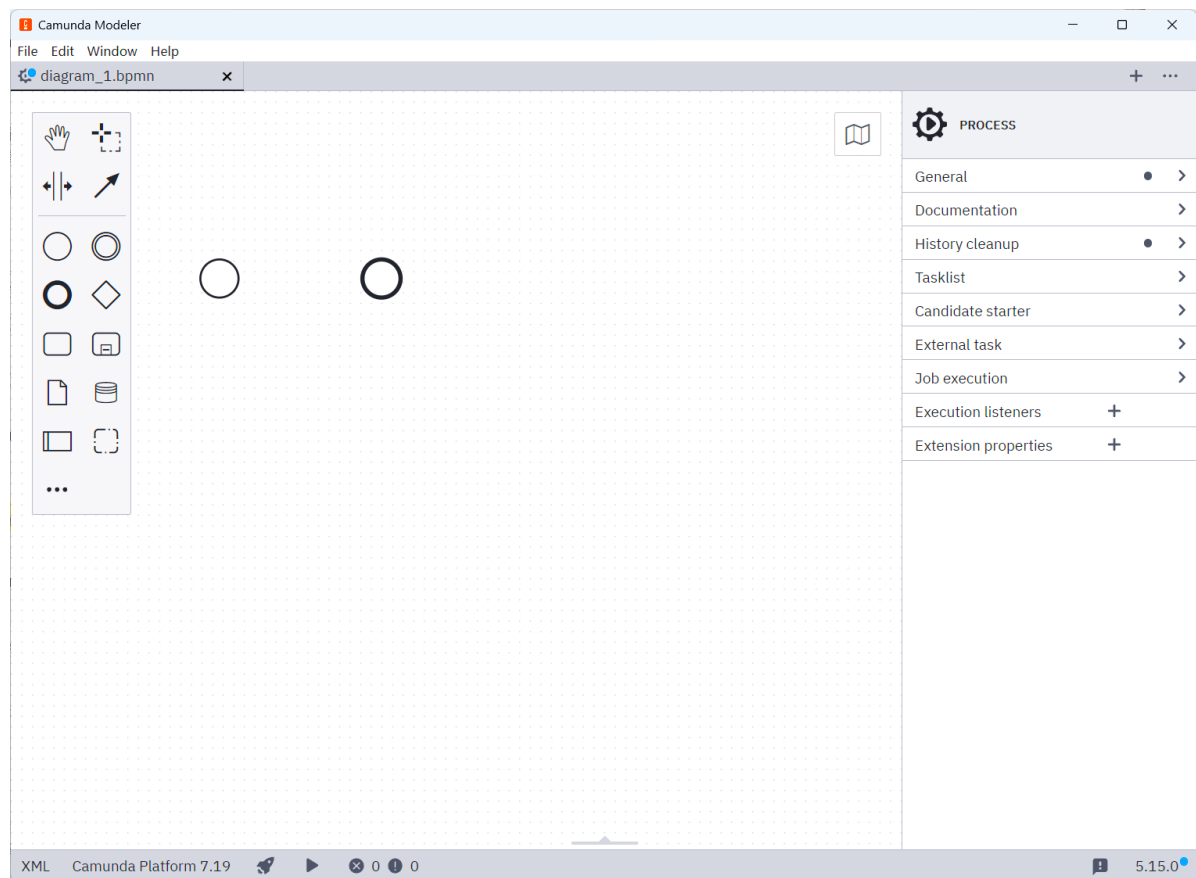
名称	修改日期	类型	大小
locales	2023/9/11 12:11	文件夹	
resources	2023/9/11 12:12	文件夹	
support	2023/9/11 12:12	文件夹	
 Camunda Modeler.exe	2023/9/11 12:12	应用程序	162,094 KB
 chrome_100_percent.pak	2023/9/11 12:11	PAK 文件	133 KB
 chrome_200_percent.pak	2023/9/11 12:11	PAK 文件	192 KB
 d3dcompiler_47.dll	2023/9/11 12:11	应用程序扩展	4,802 KB
 ffmpeg.dll	2023/9/11 12:11	应用程序扩展	2,813 KB
 icudtl.dat	2023/9/11 12:11	DAT 文件	10,383 KB
 libEGL.dll	2023/9/11 12:11	应用程序扩展	470 KB
 libGLESv2.dll	2023/9/11 12:11	应用程序扩展	7,312 KB
 LICENSE.camunda-modeler.txt	2023/9/11 12:05	文本文档	2 KB
 LICENSE.electron.txt	2023/9/11 12:11	文本文档	2 KB
 LICENSES.chromium.html	2023/9/11 12:11	Microsoft Edge H...	8,643 KB
 resources.pak	2023/9/11 12:11	PAK 文件	5,270 KB
 snapshot_blob.bin	2023/9/11 12:11	BIN 文件	263 KB
 THIRD_PARTY_NOTICES.camunda-mo...	2023/9/11 12:05	文本文档	419 KB
 v8_context_snapshot.bin	2023/9/11 12:11	BIN 文件	582 KB
 VERSION	2023/9/11 12:12	文件	1 KB
 vk_swiftshader.dll	2023/9/11 12:11	应用程序扩展	5,007 KB
 vk_swiftshader_icd.json	2023/9/11 12:11	JSON File	1 KB
 vulkan-1.dll	2023/9/11 12:11	应用程序扩展	918 KB

运行

双击执行 `Camunda Modeler.exe`



BPMN设计界面

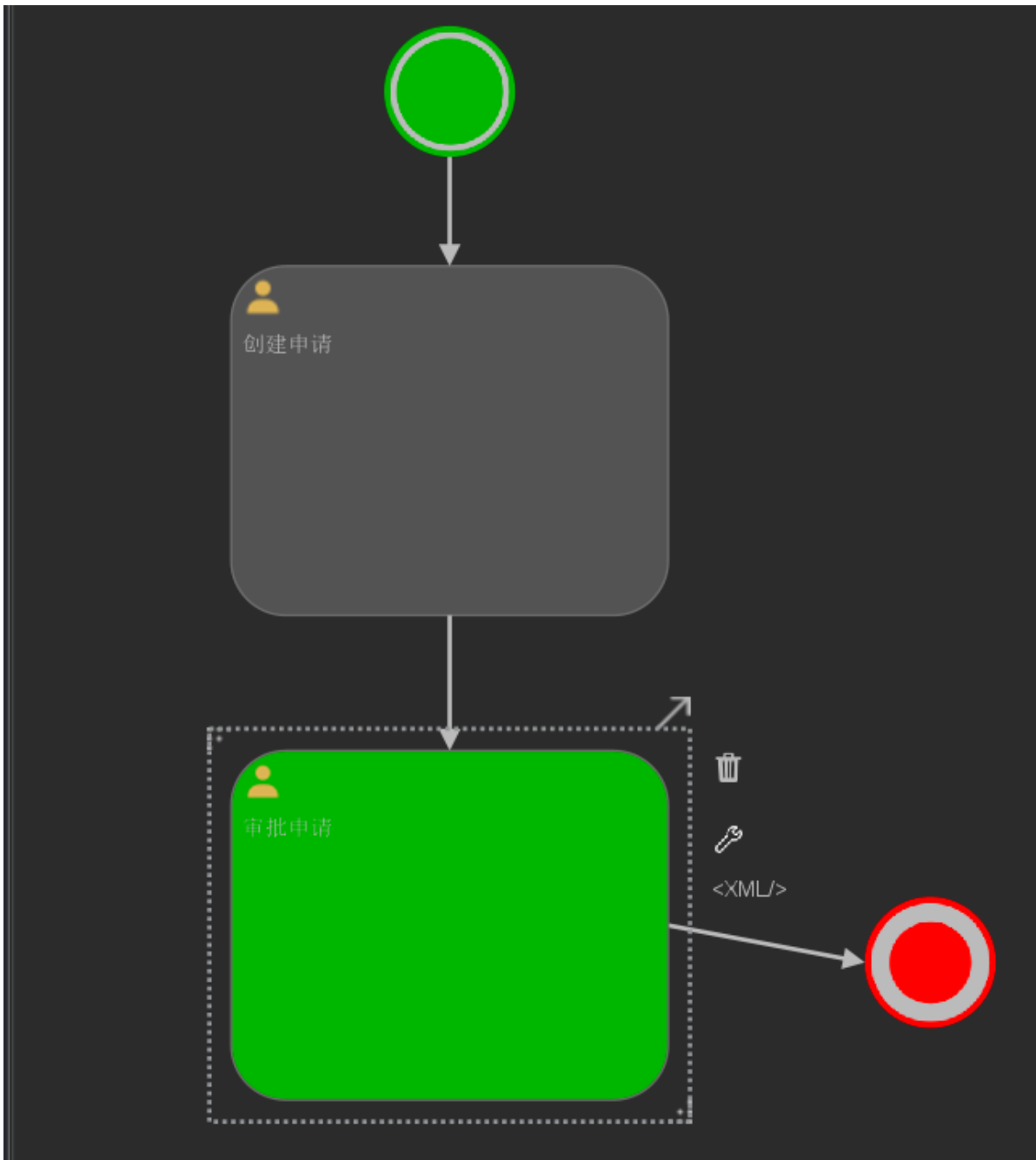


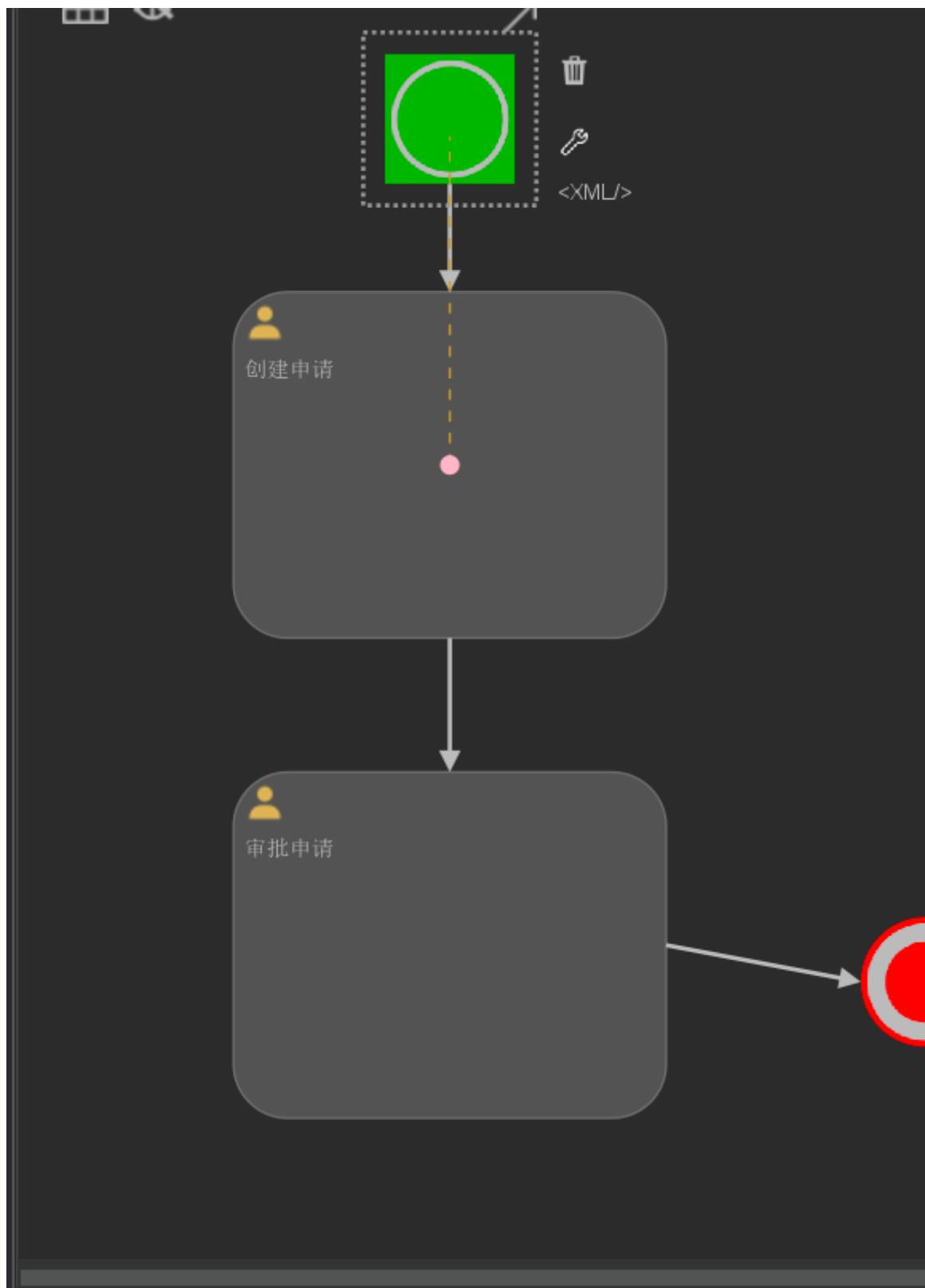
流程操作

流程定义

流程定义是线下按照bpmn2.0标准去描述业务流程

创建一个比较简单的请假流程



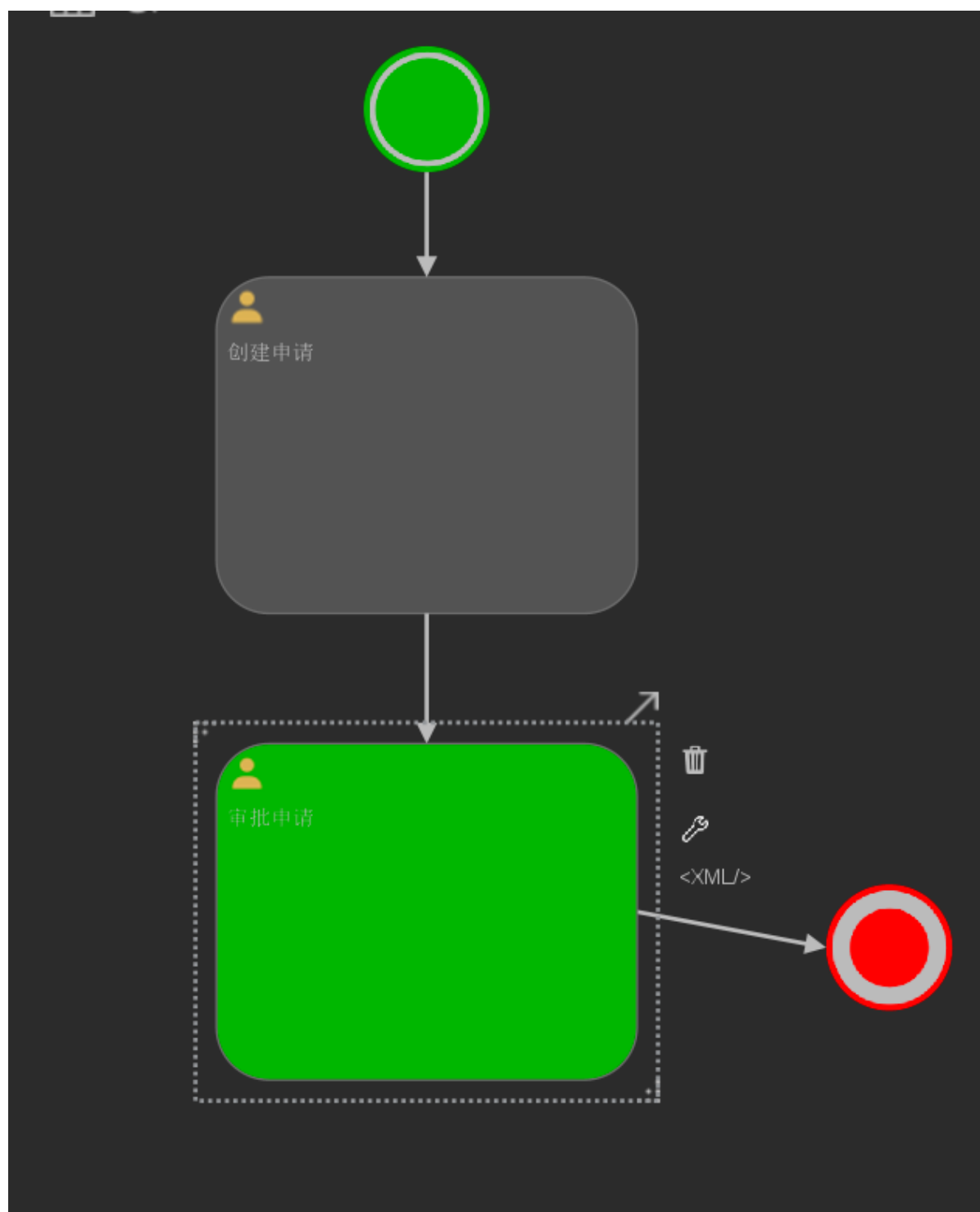


ID	sid-b602704c-cae7-4346-bab2-07f7831ba0e8
Name	审批申请
Documentation	
Is for compensation	<input type="checkbox"/>
Asynchronous	<input type="checkbox"/>
Assignee	
Candidate Users	
Candidate Groups	

Candidate groups	
Due date	
Form key	
Priority	

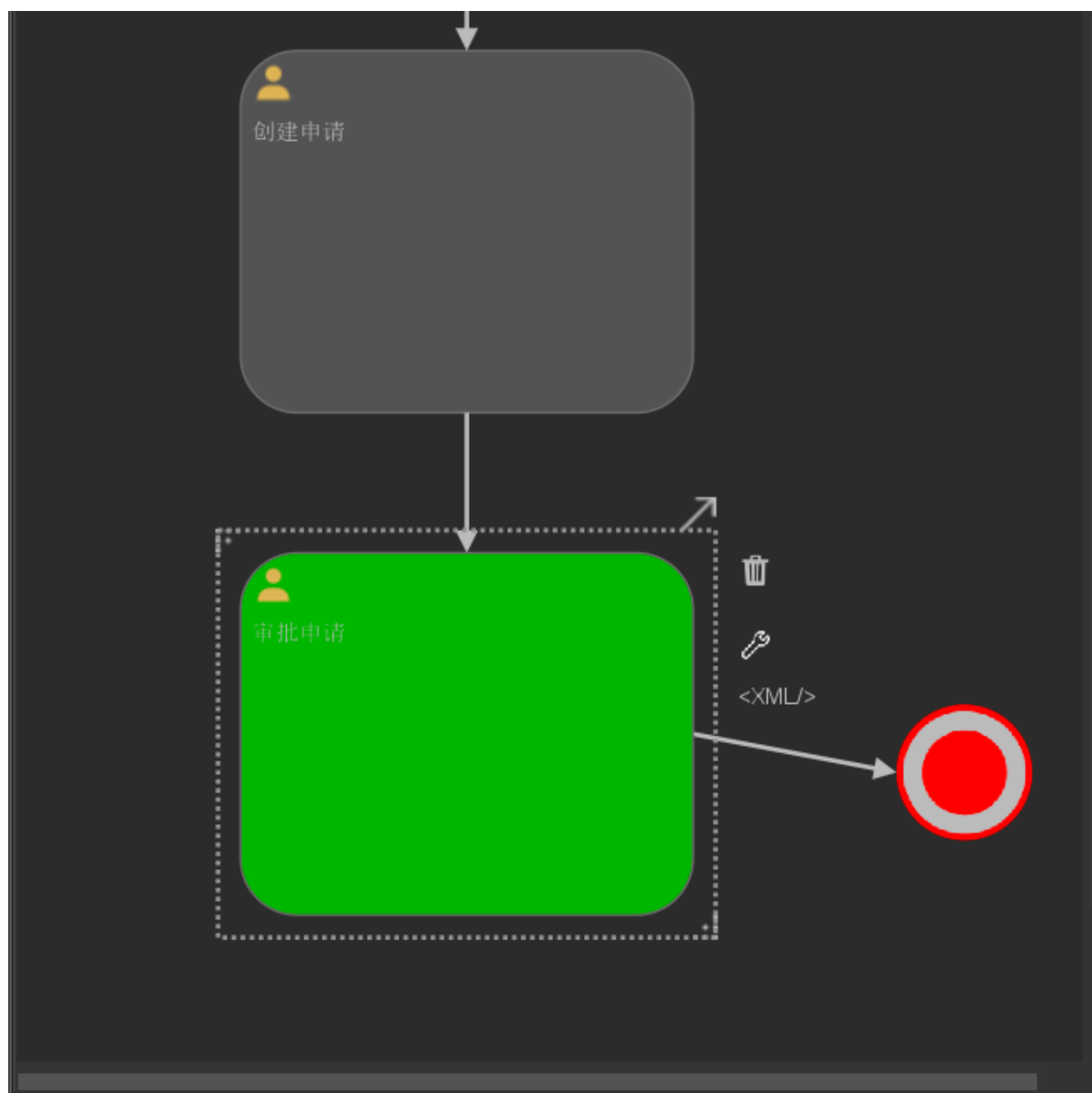
分别指定两个活动的**名称与负责人**。一个流程模板每个工作节点的负责人当然不可能是固定的，所以，在这里我们采用 Activiti 所支持的 **UEL** 表达式，将负责人定义为动态参数

assignee0 和 assignee1



ID	sid-b602704c-cae7-4346-bab2-07f7831ba0e8
Name	审批申请
Documentation	
Is for compensation	<input type="checkbox"/>
Asynchronous	<input type="checkbox"/>
Assignee	
Candidate Users	
Candidate Groups	
Due date	
Form key	

Form key	
Priority	



ID	sid-b602704c-cae7-4346-bab2-07f7831ba0e8
Name	审批申请
Documentation	
Is for compensation	<input type="checkbox"/>
Asynchronous	<input type="checkbox"/>
Assignee	\${assignee1}
Candidate Users	
Candidate Groups	
Due date	
Form key	
Priority	

xml文件:

```
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/processdef">
3   <process id="test" name="test" isExecutable="true">
4     <startEvent id="sid-c3b15a5f-fe70-4fe1-af9a-edf7b415c0ee" name="请假流程定
      义"/>
5     <userTask id="sid-32a2b9d4-6749-4d6b-aedb-2fac3815170c" name="创建申请"
      activiti:assignee="{assignee0}"/>
6     <userTask id="sid-b602704c-cae7-4346-bab2-07f7831ba0e8" name="审批申请"
      activiti:assignee="{assignee1}"/>
7     <sequenceFlow id="sid-a6d8056e-3823-4e0b-beb4-1dde11a497c8"
      sourceRef="sid-32a2b9d4-6749-4d6b-aedb-2fac3815170c" targetRef="sid-
      b602704c-cae7-4346-bab2-07f7831ba0e8"/>
8     <endEvent id="sid-c82edc5e-3028-4d2e-bcbc-d780ece87d6f" name="结束"/>
9     <sequenceFlow id="sid-fac53945-3036-4db7-9c73-da1631eb6cf9"
      sourceRef="sid-b602704c-cae7-4346-bab2-07f7831ba0e8" targetRef="sid-
      c82edc5e-3028-4d2e-bcbc-d780ece87d6f"/>
10    <sequenceFlow id="sid-8930ce5c-ba7d-4f48-a4a0-113ff80fca96"
      sourceRef="sid-c3b15a5f-fe70-4fe1-af9a-edf7b415c0ee" targetRef="sid-
      32a2b9d4-6749-4d6b-aedb-2fac3815170c"/>
11  </process>
12  <bpmndi:BPMNDiagram id="BPMNDiagram_test">
13    <bpmndi:BPMNPlane bpmnElement="test" id="BPMNPlane_test">
14      <bpmndi:BPMNShape id="shape-643471ce-5aaf-4ce9-9654-0f9b552705b4"
      bpmnElement="sid-c3b15a5f-fe70-4fe1-af9a-edf7b415c0ee">
15        <omgdc:Bounds x="-1823.05" y="-3102.7" width="30.0" height="30.0"/>
16      </bpmndi:BPMNShape>
17      <bpmndi:BPMNShape id="shape-659cf6cb-a545-4b63-925d-4cad5523a1dc"
      bpmnElement="sid-32a2b9d4-6749-4d6b-aedb-2fac3815170c">
18        <omgdc:Bounds x="-1858.05" y="-3047.7" width="100.0" height="80.0"/>
19      </bpmndi:BPMNShape>
20      <bpmndi:BPMNShape id="shape-279309f5-d8fa-4951-88e1-20c98857ea5b"
      bpmnElement="sid-b602704c-cae7-4346-bab2-07f7831ba0e8">
21        <omgdc:Bounds x="-1858.05" y="-2936.0425" width="100.0"
      height="80.0"/>
22      </bpmndi:BPMNShape>
23      <bpmndi:BPMNEdge id="edge-f434e240-3d5e-4c2f-9f6c-2544d81f831e"
      bpmnElement="sid-a6d8056e-3823-4e0b-beb4-1dde11a497c8">
24        <omgdi:waypoint x="-1808.05" y="-2967.7"/>
25        <omgdi:waypoint x="-1808.05" y="-2936.0425"/>
26      </bpmndi:BPMNEdge>
```

```

27     <bpmndi:BPMNShape id="shape-f0874fe4-9331-4915-9b96-c436412231dc"
bpmnElement="sid-c82edc5e-3028-4d2e-bcbc-d780ece87d6f">
28         <omgdc:Bounds x="-1713.05" y="-2903.2" width="30.0" height="30.0"/>
29     </bpmndi:BPMNShape>
30     <bpmndi:BPMNEdge id="edge-b8629684-33cd-4007-b782-b6403dc5034b"
bpmnElement="sid-fac53945-3036-4db7-9c73-da1631eb6cf9">
31         <omgdi:waypoint x="-1758.05" y="-2896.0425"/>
32         <omgdi:waypoint x="-1713.05" y="-2888.2"/>
33     </bpmndi:BPMNEdge>
34     <bpmndi:BPMNEdge id="edge-3144cad3-fff8-4390-a9d6-f1ffdce61cde"
bpmnElement="sid-8930ce5c-ba7d-4f48-a4a0-113ff80fca96">
35         <omgdi:waypoint x="-1808.05" y="-3072.7"/>
36         <omgdi:waypoint x="-1808.05" y="-3047.7"/>
37     </bpmndi:BPMNEdge>
38 </bpmndi:BPMNPlane>
39 </bpmndi:BPMNDiagram>
40 </definitions>
41

```

流程定义部署

将上面在设计器中定义的流程部署到activiti数据库中，就是流程定义部署

```

1  package mao.activiti_process_deploy;
2
3  import org.activiti.engine.ProcessEngine;
4  import org.activiti.engine.ProcessEngines;
5  import org.activiti.engine.RepositoryService;
6  import org.activiti.engine.repository.Deployment;
7  import org.slf4j.Logger;
8  import org.slf4j.LoggerFactory;
9  import org.springframework.boot.SpringApplication;
10 import org.springframework.boot.autoconfigure.SpringBootApplication;
11
12
13 @SpringBootApplication
14 public class ActivitiProcessDeployApplication
15 {
16     private static final Logger log =
LoggerFactory.getLogger(ActivitiProcessDeployApplication.class);
17
18     public static void main(String[] args)
19     {
20         SpringApplication.run(ActivitiProcessDeployApplication.class, args);
21         ProcessEngine processEngine =
ProcessEngines.getDefaultProcessEngine();

```



```

22     RepositoryService repositoryService =
processEngine.getRepositoryService();
23     Deployment deployment = repositoryService.createDeployment()
24         .name("请假流程定义")
25         .addClasspathResource("test.bpmn20.xml")
26         .deploy();
27     log.info("流程id: " + deployment.getId());
28     log.info("流程名称: " + deployment.getName());
29 }
30
31 }

```

也可以直接加载压缩包

```

1 // 定义zip输入流
2     InputStream inputStream = this
3         .getClass()
4         .getClassLoader()
5         .getResourceAsStream(
6             "bpmn/evection.zip");
7     ZipInputStream zipInputStream = new ZipInputStream(inputStream);
8     // 获取repositoryService
9     RepositoryService repositoryService = processEngine
10         .getRepositoryService();
11     // 流程部署
12     Deployment deployment = repositoryService.createDeployment()
13         .addZipInputStream(zipInputStream)
14         .deploy();

```

运行程序：

```

1 2023-09-19 11:03:50.027 INFO 13952 --- [          main]
   aultActiviti5CompatibilityHandlerFactory : Activiti 5 compatibility handler
   implementation not found or error during instantiation :
   org.activiti.compatibility.DefaultActiviti5CompatibilityHandler. Activiti 5
   backwards compatibility disabled.
2 2023-09-19 11:03:50.050 INFO 13952 --- [          main]
   o.a.engine.impl.ProcessEngineImpl      : ProcessEngine default created
3 2023-09-19 11:03:50.052 INFO 13952 --- [          main]
   org.activiti.engine.ProcessEngines     : initialised process engine default
4 2023-09-19 11:03:50.142 INFO 13952 --- [          main]
   m.a.ActivitiProcessDeployApplication    : 流程id: 2501
5 2023-09-19 11:03:50.142 INFO 13952 --- [          main]
   m.a.ActivitiProcessDeployApplication    : 流程名称: 请假流程定义

```

操作的数据表

流程定义部署后操作Activiti的3张表如下：

- **act_re_deployment**：流程定义部署表，每部署一次增加一条记录
- **act_re_procdef**：流程定义表，部署每个新的流程定义都会在这张表中增加一条记录
- **act_ge_bytearray**：流程资源表

看看写入了什么数据：

```
1 | select * from act_re_deployment;
```

```
1 | select * from act_re_procdef;
```

```
1 | select * from act_ge_bytearray;
```

```
1 | mysql> use activiti;
2 | Database changed
3 | mysql> select * from act_re_deployment;
4 | +-----+-----+-----+-----+-----+-----+
5 | | ID_ | NAME_ | CATEGORY_ | KEY_ | TENANT_ID_ | DEPLOY_TIME_
6 | | ENGINE_VERSION_ |
7 | | 2501 | 请假流程定义 | NULL | NULL | | 2023-09-19
8 | 11:03:50.054 | NULL |
9 | 1 row in set (0.00 sec)
10 |
11 | mysql> select * from act_re_procdef;
12 | +-----+-----+-----+-----+-----+-----+
13 | | ID_ | REV_ | CATEGORY_ | NAME_ | KEY_ |
14 | VERSION_ | DEPLOYMENT_ID_ | RESOURCE_NAME_ | DGRM_RESOURCE_NAME_ |
15 | DESCRIPTION_ | HAS_START_FORM_KEY_ | HAS_GRAPHICAL_NOTATION_ |
16 | SUSPENSION_STATE_ | TENANT_ID_ | ENGINE_VERSION_ |
```

```

14  +-----+-----+-----+-----+-----+-----+
    | test:1:2503 | 1 | http://www.activiti.org/processdef | test | test | |
    | 1 | 2501 | | test.bpmn20.xml | NULL | NULL |
    | | 0 | | 1 | | 1 |
    | NULL | |
16  +-----+-----+-----+-----+-----+-----+
    | | | | | | | | | |
    +-----+-----+-----+-----+-----+-----+
17  1 row in set (0.00 sec)
18
19  mysql> select * from act_ge_bytearray;

```

```
1 mysql> select * from act_ge_bytearray\G;
2 ***** 1. row *****
3         ID_: 2502
4         REV_: 1
5         NAME_: test.bpmn20.xml
6        DEPLOYMENT_ID_: 2501
```

0x3C3F786D6C2076657273696F6E3D22312E302220656E636F64696E673D225554462D38223F3
E0D0A3C646566696E6974696F6E7320786D6C6E733D22687474703A2F2F7777772E6F6D672E6F
72672F737065632F42504D4E2F32303130303532342F4D4F44454C2220786D6C6E733A7873693
D22687474703A2F2F7777772E77332E6F72672F323030312F584D4C536368656D612D696E7374
616E63652220786D6C6E733A7873643D22687474703A2F2F7777772E77332E6F72672F3230303
12F584D4C536368656D612220786D6C6E733A61637469766974693D22687474703A2F2F616374
69766974692E6F72672F62706D6E2220786D6C6E733A62706D6E64693D22687474703A2F2F777
7772E6F6D672E6F72672F737065632F42504D4E2F32303130303532342F44492220786D6C6E73
3A6F6D6764633D22687474703A2F2F7777772E6F6D672E6F72672F737065632F44442F3230313
0303532342F44432220786D6C6E733A6F6D6764693D22687474703A2F2F7777772E6F6D672E6F
72672F737065632F44442F32303130303532342F44492220747970654C616E67756167653D226
87474703A2F2F7777772E77332E6F72672F323030312F584D4C536368656D6122206578707265
7373696F6E4C616E67756167653D22687474703A2F2F7777772E77332E6F72672F313939392F5
85061746822207461726765744E616D6573706163653D22687474703A2F2F7777772E61637469
766974692E6F72672F70726F63657373646566223E0D0A20203C70726F636573732069643D227
465737422206E616D653D22746573742220697345786563757461626C653D2274727565223E0D
0A202020203C73746172744576656E742069643D227369642D63336231356135662D666537302
D346665312D616639612D65646637623431356330656522206E616D653D22E8AFB7E58187E6B5
81E7A88BE5AE9AE4B989222F3E0D0A202020203C757365725461736B2069643D227369642D333
26132623964342D363734392D346436622D616564622D32666163333831353137306322206E61
6D653D22E5889BE5BBBAE794B3E8AFB7222061637469766974693A61737369676E65653D22247
B61737369676E6565307D222F3E0D0A202020203C757365725461736B2069643D227369642D62
363032373034632D636165372D343334362D626162322D30376637383331626130653822206E6
16D653D22E5AEA1E689B9E794B3E8AFB7222061637469766974693A61737369676E65653D2224
7B61737369676E6565317D222F3E0D0A202020203C73657175656E6365466C6F772069643D227
369642D61366438303536652D333832332D346530622D626562342D3164646531316134393763
382220736F757263655265663D227369642D33326132623964342D363734392D346436622D616
564622D32666163333831353137306322207461726765745265663D227369642D623630323730
34632D636165372D343334362D626162322D303766373833316261306538222F3E0D0A2020202
03C656E644576656E742069643D227369642D63383265646335652D333032382D346432652D62
6362632D64373830656365383764366622206E616D653D22E7BB93E69D9F222F3E0D0A2020202
03C73657175656E6365466C6F772069643D227369642D66616335333934352D333033362D3464
62372D396337332D6461313633316562366366392220736F757263655265663D227369642D623
63032373034632D636165372D343334362D626162322D30376637383331626130653822207461
726765745265663D227369642D63383265646335652D333032382D346432652D626362632D643
738306563653837643666222F3E0D0A202020203C73657175656E6365466C6F772069643D2273
69642D38393330636535632D626137642D346634382D613461302D31313366663830666361393
62220736F757263655265663D227369642D6336231356135662D666537302D346665312D6166
39612D65646637623431356330656522207461726765745265663D227369642D3332613262396
4342D363734392D346436622D616564622D326661633338313531373063222F3E0D0A20203C2F
70726F636573733E0D0A20203C62706D6E64693A42504D4E4469616772616D2069643D2242504
D4E4469616772616D5F74657374223E0D0A202020203C62706D6E64693A42504D4E506C616E65
2062706D6E456C656D656E743D2274657374222069643D2242504D4E506C616E655F746573742
23E0D0A2020202020203C62706D6E64693A42504D4E53686170652069643D2273686170652D36
343334373163652D356161662D346365392D393635342D3066396235353237303562342220627
06D6E456C656D656E743D227369642D63336231356135662D666537302D346665312D61663961
2D656466376234313563306565223E0D0A20202020202020203C6F6D6764633A426F756E64732
0783D222D313832332E30352220793D222D333130322E37222077696474683D2233302E302220
6865696768743D2233302E30222F3E0D0A2020202020203C2F62706D6E64693A42504D4E53686
170653E0D0A2020202020203C62706D6E64693A42504D4E53686170652069643D227368617065
2D36353963663663622D613534352D346236332D393235642D346361643535323361316463222
062706D6E456C656D656E743D227369642D33326132623964342D363734392D346436622D6165
64622D326661633338313531373063223E0D0A20202020202020203C6F6D6764633A426F756E6
47320783D222D313835382E30352220793D222D333034372E37222077696474683D223130302E

3022206865696768743D2238302E30222F3E0D0A2020202020203C2F62706D6E64693A42504D4
E53686170653E0D0A2020202020203C62706D6E64693A42504D4E53686170652069643D227368
6170652D32373933303966352D643866612D343935312D383865312D323063393838353765613
562222062706D6E456C656D656E743D227369642D62363032373034632D636165372D34333436
2D626162322D303766373833316261306538223E0D0A20202020202020203C6F6D6764633A426
F756E647320783D222D313835382E30352220793D222D323933362E3034323522207769647468
3D223130302E3022206865696768743D2238302E30222F3E0D0A2020202020203C2F62706D6E6
4693A42504D4E53686170653E0D0A2020202020203C62706D6E64693A42504D4E456467652069
643D22656467652D66343334653234302D336435652D346332662D396636632D3235343464383
16638333165222062706D6E456C656D656E743D227369642D61366438303536652D333832332D
346530622D626562342D316464653131613439376338223E0D0A20202020202020203C6F6D676
4693A776179706F696E7420783D222D313830382E30352220793D222D323936372E37222F3E0D
0A20202020202020203C6F6D6764693A776179706F696E7420783D222D313830382E303522207
93D222D323933362E30343235222F3E0D0A2020202020203C2F62706D6E64693A42504D4E4564
67653E0D0A2020202020203C62706D6E64693A42504D4E53686170652069643D2273686170652
D66303837346665342D393333312D343931352D396239362D6334333634313232333164632220
62706D6E456C656D656E743D227369642D63383265646335652D333032382D346432652D62636
2632D643738306563653837643666223E0D0A20202020202020203C6F6D6764633A426F756E64
7320783D222D313731332E30352220793D222D323930332E32222077696474683D2233302E302
2206865696768743D2233302E30222F3E0D0A2020202020203C2F62706D6E64693A42504D4E53
686170653E0D0A2020202020203C62706D6E64693A42504D4E456467652069643D22656467652
D62383632393638342D333363642D343030372D623738322D6236343033646335303334622220
62706D6E456C656D656E743D227369642D66616335333934352D333033362D346462372D39633
7332D646131363331656236636639223E0D0A20202020202020203C6F6D6764693A776179706F
696E7420783D222D313735382E30352220793D222D323839362E30343235222F3E0D0A2020202
0202020203C6F6D6764693A776179706F696E7420783D222D313731332E30352220793D222D32
3838382E32222F3E0D0A2020202020203C2F62706D6E64693A42504D4E456467653E0D0A20202
02020203C62706D6E64693A42504D4E456467652069643D22656467652D33313434636164332D
666666382D343339302D613964362D663166666463653631636465222062706D6E456C656D656
E743D227369642D38393330636535632D626137642D346634382D613461302D31313366663830
6663613936223E0D0A20202020202020203C6F6D6764693A776179706F696E7420783D222D313
830382E30352220793D222D333037322E37222F3E0D0A20202020202020203C6F6D6764693A77
6179706F696E7420783D222D313830382E30352220793D222D333034372E37222F3E0D0A20202
02020203C2F62706D6E64693A42504D4E456467653E0D0A202020203C2F62706D6E64693A4250
4D4E506C616E653E0D0A20203C2F62706D6E64693A42504D4E4469616772616D3E0D0A3C2F646
56696E6974696F6E733E0D0A

8 GENERATED_: 0
9 1 row in set (0.00 sec)

信息	结果 1	结果 2	结果 3	剖析	状态	
ID_	NAME_	CATEGORY_	KEY_	TENANT_ID_	DEPLOY_TIME_	ENGINE_VERSION_
▶ 2501	请假流程定义	(Null)	(Null)		2023-09-19 11:03:50.054	(Null)

信息														结果 1	结果 2	结果 3	剖析	状态
ID_	REV_	CATEGORY_	NAME_	KEY_	VERSION_	DEPLOYMENT_ID_	RESOURCE_NAME_	DGRM_	RESOURCE_DESCRIPTION_	HAS_START_FORM_	HAS_GRAPHICAL_N	SUSPENSION_STAT	TENANT_ID_	ENGINE_VERSION_				
▶test:1:2503		1	http://www.activiti..test	test		1	2501	test.bpmn20.xml	(Null)	(Null)	0	1	1	(Null)				

信息	结果 1	结果 2	结果 3	剖析	状态
ID_	REV_	NAME_	DEPLOYMENT_ID_	BYTES_	GENERATED_
▶ 2502	1	test.bpmn20.xml	2501	(BLOB) 3.36 KB	0

act_re_deployment 和 act_re_procdef 一对多关系，一次部署在流程部署表生成一条记录，但一次部署可以部署多个流程定义，每个流程定义在流程定义表生成一条记录

启动流程实例

流程定义部署在activiti后就可以通过工作流管理业务流程了

针对该流程，启动一个流程表示发起一个新的请假申请单，这就相当于java类与java对象的关系，类定义好后需要new创建一个对象使用

```

1  package mao.activiti_process_deploy;
2
3  import org.activiti.engine.ProcessEngine;
4  import org.activiti.engine.ProcessEngines;
5  import org.activiti.engine.RuntimeService;
6  import org.activiti.engine.runtime.ProcessInstance;
7  import org.junit.jupiter.api.Test;
8  import org.slf4j.Logger;
9  import org.slf4j.LoggerFactory;
10 import org.springframework.boot.test.context.SpringBootTest;
11
12 import java.util.HashMap;
13 import java.util.Map;
14
15 @SpringBootTest
16 class ActivitiProcessDeployApplicationTests
17 {
18     private static final Logger log =
19         LoggerFactory.getLogger(ActivitiProcessDeployApplicationTests.class);
20
21     @Test
22     void testStartProcess()
23     {
24         ProcessEngine processEngine =
25             ProcessEngines.getDefaultProcessEngine();
26         RuntimeService runtimeService = processEngine.getRuntimeService();
27         Map<String, Object> map = new HashMap<>();
28         map.put("assignee0", "张三");

```

```

27         map.put("assignee1", "王经理");
28         ProcessInstance instance =
runtimeService.startProcessInstanceByKey("test", map);
29         log.info("流程定义id:" + instance.getProcessDefinitionId());
30         log.info("流程实例 id:" + instance.getId());
31         log.info("当前活动的id:" + instance.getActivityId());
32     }
33
34 }
35

```

启动流程时，分别指定各个任务的**负责人**，上述代码表示该请假流程是由**张三**发起的，审批人是**王经理**其中参数 **key** 表示指定所启动的流程定义，而 **map** 则是为其启动之后的流程实例赋值

key位于xml文件里

```

<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL bpmn20.xml">
  <process id="test" name="test" isExecutable="true">
    <startEvent id="sid-c3b15a5f-fe70-4fe1-af9a-edf7b415c0ee" name="请假流程定义"/>
    <userTask id="sid-32a2b9d4-6749-4d6b-aedb-2fac3815170c" name="创建申请" activiti:assignee="张三"/>
    <userTask id="sid-b602704c-cae7-4346-bab2-07f7831ba0e8" name="审批申请" activiti:assignee="王经理"/>
    <sequenceFlow id="sid-a6d8056e-3823-4e0b-beb4-1dde11a497c8" sourceRef="sid-32a2b9d4-6749-4d6b-aedb-2fac3815170c" targetRef="sid-b602704c-cae7-4346-bab2-07f7831ba0e8"/>
    <endEvent id="sid-c82edc5e-3028-4d2e-bcbc-d780ece87d6f" name="结束"/>
    <sequenceFlow id="sid-fac53945-3036-4db7-9c73-da1631eb6cf9" sourceRef="sid-b602704c-cae7-4346-bab2-07f7831ba0e8" targetRef="sid-c82edc5e-3028-4d2e-bcbc-d780ece87d6f"/>
    <sequenceFlow id="sid-8930ce5c-ba7d-4f48-a4a0-113ff80fca96" sourceRef="sid-c3b15a5f-fe70-4fe1-af9a-edf7b415c0ee" targetRef="sid-b602704c-cae7-4346-bab2-07f7831ba0e8"/>
  </process>
</definitions>

```

运行：

```

1 2023-09-19 11:23:29.648 INFO 27400 --- [main]
.a.ActivitiProcessDeployApplicationTests : 流程定义id:test:1:2503
2 2023-09-19 11:23:29.648 INFO 27400 --- [main]
.a.ActivitiProcessDeployApplicationTests : 流程实例 id:5001
3 2023-09-19 11:23:29.648 INFO 27400 --- [main]
.a.ActivitiProcessDeployApplicationTests : 当前活动的id:null

```

操作的数据表

启动流程实例操作的数据表如下：

- **act_hi_actinst**：流程实例执行历史
- **act_hi_identitylink**：流程的参与用户历史信息
- **act_hi_procinst**：流程实例历史信息

- **act_hi_taskinst**: 流程任务历史信息
- **act_ru_execution**: 流程执行信息
- **act_ru_identitylink**: 流程的参与用户信息
- **act_ru_task**: 任务信息

```

1 select * from act_hi_actinst;
2 select * from act_hi_identitylink;
3 select * from act_hi_procinstd;
4 select * from act_hi_taskinst;
5 select * from act_ru_execution;
6 select * from act_ru_identitylink;
7 select * from act_ru_task;

```

```

1 mysql> select * from act_hi_actinst;
2 +-----+-----+-----+-----+-----+-----+-----+
3 | ID_ | PROC_DEF_ID_ | PROC_INST_ID_ | EXECUTION_ID_ | ACT_ID_ |
4 | | | | | | |
5 | ID_ | PROC_DEF_ID_ | PROC_INST_ID_ | EXECUTION_ID_ | ACT_ID_ |
6 | | | | | | |
7 | ID_ | PROC_DEF_ID_ | PROC_INST_ID_ | EXECUTION_ID_ | ACT_ID_ |
8 | | | | | | |
9 | ID_ | PROC_DEF_ID_ | PROC_INST_ID_ | EXECUTION_ID_ | ACT_ID_ |
10 | | | | | | |
11 | ID_ | PROC_DEF_ID_ | PROC_INST_ID_ | EXECUTION_ID_ | ACT_ID_ |
12 | | | | | | |
13 | ID_ | PROC_DEF_ID_ | PROC_INST_ID_ | EXECUTION_ID_ | ACT_ID_ |
14 | | | | | | |
15 | ID_ | PROC_DEF_ID_ | PROC_INST_ID_ | EXECUTION_ID_ | ACT_ID_ |
16 | | | | | | |
17 | ID_ | PROC_DEF_ID_ | PROC_INST_ID_ | EXECUTION_ID_ | ACT_ID_ |
18 | | | | | | |

```

ID_	PROC_DEF_ID_	PROC_INST_ID_	EXECUTION_ID_	ACT_ID_	TASK_ID_	CALL_PROC_INST_ID_	ACT_NAME_	ACT_TYPE_	ASSIGNEE_	START_TIME_	END_TIME_	DURATION_	DELETE_REASON_	TENANT_ID_
5005	test:1:2503	5001	5004	sid-c3b15a5f-fe70-4fe1-af9a-edf7b415c0ee	NULL	NULL	请假流程定义	1	张三	2023-09-19 11:23:29.585	2023-09-19 11:23:29.586	1	NULL	NULL
5006	test:1:2503	5001	5004	sid-32a2b9d4-6749-4d6b-aedb-2fac3815170c	5007	NULL	创建申请	1	张三	2023-09-19 11:23:29.586	NULL	NULL	NULL	NULL

```

2 rows in set (0.00 sec)

mysql> select * from act_hi_identitylink;
+-----+-----+-----+-----+-----+-----+-----+
| ID_ | GROUP_ID_ | TYPE_ | USER_ID_ | TASK_ID_ | PROC_INST_ID_ |
+-----+-----+-----+-----+-----+-----+-----+
| 5008 | NULL | participant | 张三 | NULL | 5001 |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from act_hi_procinstd;

```

```

19 +-----+-----+-----+-----+-----+-----+
   | ID_ | PROC_INST_ID_ | BUSINESS_KEY_ | PROC_DEF_ID_ | START_TIME_
   | END_TIME_ | DURATION_ | START_USER_ID_ | START_ACT_ID_
   | END_ACT_ID_ | SUPER_PROCESS_INSTANCE_ID_ | DELETE_REASON_ |
   | TENANT_ID_ | NAME_ |
20 +-----+-----+-----+-----+-----+-----+
   | 5001 | 5001 | NULL | test:1:2503 | 2023-09-19
   | 11:23:29.567 | NULL | NULL | NULL | sid-c3b15a5f-fe70-
   | 4fe1-af9a-edf7b415c0ee | NULL | NULL | NULL
   | | NULL |
21 +-----+-----+-----+-----+-----+-----+
   | 5001 | 5001 | NULL | test:1:2503 | 2023-09-19
   | 11:23:29.567 | NULL | NULL | NULL | sid-c3b15a5f-fe70-
   | 4fe1-af9a-edf7b415c0ee | NULL | NULL | NULL
   | | NULL |
22 +-----+-----+-----+-----+-----+-----+
   | 5001 | 5001 | NULL | test:1:2503 | 2023-09-19
   | 11:23:29.567 | NULL | NULL | NULL | sid-c3b15a5f-fe70-
   | 4fe1-af9a-edf7b415c0ee | NULL | NULL | NULL
   | | NULL |
23 +-----+-----+-----+-----+-----+-----+
   | 5001 | 5001 | NULL | test:1:2503 | 2023-09-19
   | 11:23:29.567 | NULL | NULL | NULL | sid-c3b15a5f-fe70-
   | 4fe1-af9a-edf7b415c0ee | NULL | NULL | NULL
   | | NULL |
24 1 row in set (0.00 sec)
25
26 mysql> select * from act_hi_taskinst;
27 +-----+-----+-----+-----+-----+-----+
   | ID_ | PROC_DEF_ID_ | TASK_DEF_KEY_ | | | |
   | PROC_INST_ID_ | EXECUTION_ID_ | NAME_ | PARENT_TASK_ID_ | DESCRIPTION_ |
   | OWNER_ | ASSIGNEE_ | START_TIME_ | CLAIM_TIME_ | END_TIME_ |
   | DURATION_ | DELETE_REASON_ | PRIORITY_ | DUE_DATE_ | FORM_KEY_ | CATEGORY_ |
   | TENANT_ID_ |
28 +-----+-----+-----+-----+-----+-----+
   | 5007 | test:1:2503 | sid-32a2b9d4-6749-4d6b-aedb-2fac3815170c | 5001
   | 5004 | 创建申请 | NULL | NULL | NULL | 张
   | 2023-09-19 11:23:29.592 | NULL | NULL | NULL |
   | NULL | 50 | NULL | NULL | NULL |
   |
29 +-----+-----+-----+-----+-----+-----+
   | 5007 | test:1:2503 | sid-32a2b9d4-6749-4d6b-aedb-2fac3815170c | 5001
   | 5004 | 创建申请 | NULL | NULL | NULL | 张
   | 2023-09-19 11:23:29.592 | NULL | NULL | NULL |
   | NULL | 50 | NULL | NULL | NULL |
   |
30 +-----+-----+-----+-----+-----+-----+
   | 5007 | test:1:2503 | sid-32a2b9d4-6749-4d6b-aedb-2fac3815170c | 5001
   | 5004 | 创建申请 | NULL | NULL | NULL | 张
   | 2023-09-19 11:23:29.592 | NULL | NULL | NULL |
   | NULL | 50 | NULL | NULL | NULL |
   |
31 +-----+-----+-----+-----+-----+-----+
   | 5007 | test:1:2503 | sid-32a2b9d4-6749-4d6b-aedb-2fac3815170c | 5001
   | 5004 | 创建申请 | NULL | NULL | NULL | 张
   | 2023-09-19 11:23:29.592 | NULL | NULL | NULL |
   | NULL | 50 | NULL | NULL | NULL |
   |
32 1 row in set (0.00 sec)
33
34 mysql> select * from act_ru_execution;

```



```
51 1 row in set (0.00 sec)
52
53 mysql> select * from act_ru_task;
54 +-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+
55 | ID_ | REV_ | EXECUTION_ID_ | PROC_INST_ID_ | PROC_DEF_ID_ | NAME_ |
  PARENT_TASK_ID_ | DESCRIPTION_ | TASK_DEF_KEY_
56 | OWNER_ | ASSIGNEE_ | DELEGATION_ | PRIORITY_ | CREATE_TIME_
  | DUE_DATE_ | CATEGORY_ | SUSPENSION_STATE_ | TENANT_ID_ | FORM_KEY_ |
  CLAIM_TIME_ |
57 +-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+
58 | 5007 | 1 | 5004 | 5001 | test:1:2503 | 创建申请 |
  NULL | NULL | sid-32a2b9d4-6749-4d6b-aedb-2fac3815170c |
  NULL | 张三 | NULL | 50 | 2023-09-19 11:23:29.586 | NULL
  | NULL | 1 | NULL | NULL
  |
59 +-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+
60 1 row in set (0.00 sec)
61
62 mysql>
```

ID_	PROC_DEF_ID_	PROC_INST_ID_	EXECUTION_ID_	ACT_ID_	TASK_ID_	CALL_PROC_INST_ID	ACT_NAME	ACT_TYPE	ASSIGNEE	START_TIME	END_TIME	DURATION	DELETE_REASON	TENANT_ID
5005	test:1:2503	5001	5004	sid-c3b15a5f-fe70-(Null)	(Null)	(Null)	请做流程定义	startEvent	(Null)	2023-09-19 11:23:21	2023-09-19 11:23:21		1 (Null)	
5006	test:1:2503	5001	5004	sid-32a2b9d4-6749-5007	(Null)	(Null)	创建申请	userTask	张三	2023-09-19 11:23:21	(Null)		(Null) (Null)	

信息	结果 1	结果 2	结果 3	结果 4	结果 5	结果 6	结果 7	剖析	状态
ID_	GROUP_ID_	TYPE_	USER_ID_	TASK_ID_	PROC_INST_ID_				
5008	(Null)	participant	张三	(Null)	5001				

信息	结果 1	结果 2	结果 3	结果 4	结果 5	结果 6	结果 7	剖析	状态				
ID_	PROC_INST_ID_	BUSINESS_KEY_	PROC_DEF_ID_	START_TIME	END_TIME	DURATION	START_USER_ID_	START_ACT_ID_	END_ACT_ID_	SUPER_PROCESS_ID	DELETE_REASON_	TENANT_ID_	NAME_
5001	5001	(Null)	test:1.2503	2023-09-19 11:23:21	(Null)	(Null)	(Null)	sid-c3b15a5f-fe70-	(Null)	(Null)	(Null)	(Null)	(Null)

信息															结果 1	结果 2	结果 3	结果 4	结果 5	结果 6	结果 7	剖析	状态
ID	PROC_DEF_ID	TASK_DEF_KEY	PROC_INST_ID	EXECUTION_ID	NAME	PARENT_TASK_ID	DESCRIPTION	OWNER	ASSIGNEE	START_TIME	CLAIM_TIME	END_TIME	DURATION	DELETE_REASON									
5007	test:1:2503	sid-32a2b9d4-6749-5001		5004	创建申请	(Null)	(Null)	(Null)	张三	2023-09-19 11:23:21	(Null)	(Null)		(Null) (Null)									

ID_	REV_	PROC_INST_ID_	BUSINESS_KEY_	PARENT_ID_	PROC_DEF_ID_	SUPER_EXEC_	ROOT_PROC_INST_ID_	ACT_ID_	IS_ACTIVE_	IS_CONCURRENT_	IS_SCOPE_	IS_EVENT_SCOPE_	IS_MI_ROOT_	SUSPENDED
5001	1	5001	(Null)	(Null)	test:1:2503	(Null)	5001	(Null)	1	0	1	0	0	0
5004	1	5001	(Null)	5001	test:1:2503	(Null)	5001	sid-32a2b9d4-6749-	1	0	0	0	0	0

信息	结果 1	结果 2	结果 3	结果 4	结果 5	结果 6	结果 7	剖析	状态
ID_	REV_	GROUP_ID_	TYPE_	USER_ID_	TASK_ID_	PROC_INST_ID_	PROC_DEF_ID_		
▶5008		1 (Null)	participant	张三	(Null)	5001	(Null)		

信息	结果 1	结果 2	结果 3	结果 4	结果 5	结果 6	结果 7	剖析	状态						
ID_	REV_	EXECUTION_ID_	PROC_INST_ID_	PROC_DEF_ID_	NAME_	PARENT_TASK_ID_	DESCRIPTION_	TASK_DEF_KEY_	OWNER_	ASSIGNEE_	DELEGATION_	PRIORITY_	CREATE_TIME_	DUE_DATE_	C
5007	1	5004	5001	test:1:2503	创建申请	(Null)	(Null)	sid-32a2b9d4-6749-	(Null)	张三	(Null)	50	2023-09-19 11:23:21	(Null)	(

任务查询

流程启动后，任务的负责人就可以查询自己当前需要处理的任务，查询出来的任务都是该用户的待办任务

```
1 @Test
2     void testFindPersonalTaskList()
3     {
4         //任务负责人
5         String assignee = "张三";
6         ProcessEngine processEngine =
7             ProcessEngines.getDefaultProcessEngine();
8         TaskService taskService = processEngine.getTaskService();
9         List<Task> taskList = taskService.createTaskQuery()
10             .processDefinitionKey("test")
11             .taskAssignee(assignee)//只查询该任务负责人的任务
12             .list();
13         for (Task task : taskList)
14         {
15             log.info("流程定义id:" + task.getProcessDefinitionId());
16             log.info("流程实例 id:" + task.getId());
17             log.info("任务负责人: " + task.getAssignee());
18             log.info("任务名称: " + task.getName());
19         }
20     }
```

```
19  
20 }
```

```
1 2023-09-24 11:33:07.491 INFO 27788 --- [main]  
  .a.ActivitiProcessDeployApplicationTests : 流程定义id:test:1:2503  
2 2023-09-24 11:33:07.492 INFO 27788 --- [main]  
  .a.ActivitiProcessDeployApplicationTests : 流程实例 id:5007  
3 2023-09-24 11:33:07.492 INFO 27788 --- [main]  
  .a.ActivitiProcessDeployApplicationTests : 任务负责人: 张三  
4 2023-09-24 11:33:07.492 INFO 27788 --- [main]  
  .a.ActivitiProcessDeployApplicationTests : 任务名称: 创建申请  
5 2023-09-24 11:33:07.492 INFO 27788 --- [main]  
  .a.ActivitiProcessDeployApplicationTests : 流程定义id:test:1:2503  
6 2023-09-24 11:33:07.492 INFO 27788 --- [main]  
  .a.ActivitiProcessDeployApplicationTests : 流程实例 id:7507  
7 2023-09-24 11:33:07.492 INFO 27788 --- [main]  
  .a.ActivitiProcessDeployApplicationTests : 任务负责人: 张三  
8 2023-09-24 11:33:07.492 INFO 27788 --- [main]  
  .a.ActivitiProcessDeployApplicationTests : 任务名称: 创建申请
```

流程任务处理

任务负责人查询待办任务，选择任务进行处理，完成任务

```
1  /**  
2     * 完成任务  
3     */  
4     @Test  
5     void completeTask()  
6     {  
7         ProcessEngine processEngine =  
ProcessEngines.getDefaultProcessEngine();  
8         TaskService taskService = processEngine.getTaskService();  
9         Task task = taskService.createTaskQuery()  
10             .processDefinitionKey("test")  
11             .taskAssignee("张三")  
12             .list().get(0);  
13         //完成任务  
14         taskService.complete(task.getId());  
15     }
```

执行完成后，查询

```

1  @Test
2      void testFindPersonalTaskList2()
3      {
4          //任务负责人
5          String assignee = "王经理";
6          ProcessEngine processEngine =
ProcessEngines.getDefaultProcessEngine();
7          TaskService taskService = processEngine.getTaskService();
8          List<Task> taskList = taskService.createTaskQuery()
9              .processDefinitionKey("test")
10             .taskAssignee(assignee)//只查询该任务负责人的任务
11             .list();
12         for (Task task : taskList)
13         {
14             log.info("流程定义id:" + task.getProcessDefinitionId());
15             log.info("流程实例 id:" + task.getId());
16             log.info("任务负责人: " + task.getAssignee());
17             log.info("任务名称: " + task.getName());
18         }
19     }
20 }

```

```

1  2023-09-24 12:17:57.135 INFO 28432 --- [          main]
.a.ActivitiProcessDeployApplicationTests : 流程定义id:test:1:2503
2  2023-09-24 12:17:57.135 INFO 28432 --- [          main]
.a.ActivitiProcessDeployApplicationTests : 流程实例 id:10002
3  2023-09-24 12:17:57.135 INFO 28432 --- [          main]
.a.ActivitiProcessDeployApplicationTests : 任务负责人: 王经理
4  2023-09-24 12:17:57.135 INFO 28432 --- [          main]
.a.ActivitiProcessDeployApplicationTests : 任务名称: 审批申请

```

流程定义信息查询

查询流程相关信息，包含流程定义，流程部署，流程定义版本

```

1  /**
2      * 查询流程定义
3      */
4      @Test
5      public void queryProcessDefinition()
6      {
7          ProcessEngine processEngine =
ProcessEngines.getDefaultProcessEngine();
8          TaskService taskService = processEngine.getTaskService();

```

```

9      RepositoryService repositoryService =
processEngine.getRepositoryService();
10      ProcessDefinitionQuery processDefinitionQuery =
repositoryService.createProcessDefinitionQuery();
11      //查询出当前所有的流程定义
12      List<ProcessDefinition> processDefinitionList =
processDefinitionQuery.processDefinitionKey("test")
13          .orderByProcessDefinitionVersion()
14          .desc()
15          .list();
16      //输出
17      processDefinitionList.forEach(processDefinition ->
18      {
19          log.info("流程定义 id=" + processDefinition.getId());
20          log.info("流程定义 name=" + processDefinition.getName());
21          log.info("流程定义 key=" + processDefinition.getKey());
22          log.info("流程定义 Version=" + processDefinition.getVersion());
23          log.info("流程部署ID =" + processDefinition.getDeploymentId());
24      });
25  }
26

```

```

1  2023-09-24 12:26:10.022 INFO 36460 --- [          main]
.a.ActivitiProcessDeployApplicationTests : 流程定义 id=test:1:2503
2  2023-09-24 12:26:10.022 INFO 36460 --- [          main]
.a.ActivitiProcessDeployApplicationTests : 流程定义 name=test
3  2023-09-24 12:26:10.022 INFO 36460 --- [          main]
.a.ActivitiProcessDeployApplicationTests : 流程定义 key=test
4  2023-09-24 12:26:10.022 INFO 36460 --- [          main]
.a.ActivitiProcessDeployApplicationTests : 流程定义 Version=1
5  2023-09-24 12:26:10.022 INFO 36460 --- [          main]
.a.ActivitiProcessDeployApplicationTests : 流程部署ID =2501

```

流程删除


```

1  /**
2      * 流程删除
3      */
4      @Test
5      void deleteDeployment()
6      {
7          ProcessEngine processEngine =
ProcessEngines.getDefaultProcessEngine();
8          RepositoryService repositoryService =
processEngine.getRepositoryService();
9          //设置true 级联删除流程定义，即使该流程有流程实例启动也可以删除，设置为false非级
别删除方式
10         repositoryService.deleteDeployment("2503", false);
11     }

```

- 使用repositoryService删除流程定义，历史表信息不会被删除
- 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
- 如果该流程定义下存在已经运行的流程，使用普通删除报错，可用级联删除方法将流程及相关记录全部删除
- 先删除没有完成流程节点，最后就可以完全删除流程定义信息
- 项目开发中级联删除操作一般只开放给超级管理员使用

流程资源下载

下载之前添加的资源：

```

1  Deployment deployment = repositoryService.createDeployment()
2      .addClasspathResource("资源位置") // 添加资源

```

使用commons-io.jar 解决IO的操作

引入commons-io依赖包

```

1  <dependency>
2      <groupId>commons-io</groupId>
3      <artifactId>commons-io</artifactId>
4      <version>2.6</version>
5  </dependency>

```

```

1  /**
2      * 下载文件

```

```

3  *
4  * @throws IOException 异常
5  */
6  @Test
7  void queryBpmnFile() throws IOException
8  {
9      ProcessEngine processEngine = ProcessEngines.getDefaultProcessEngine();
10     RepositoryService repositoryService =
processEngine.getRepositoryService();
11     ProcessDefinition processDefinition =
repositoryService.createProcessDefinitionQuery()
12         .processDefinitionKey("test")
13         .singleResult();
14     //得到部署ID
15     String deploymentId = processDefinition.getDeploymentId();
16     InputStream inputStream =
repositoryService.getResourceAsStream(deploymentId,
processDefinition.getResourceName());
17     File file = new File("./t.bpmn");
18     FileOutputStream fileOutputStream = new FileOutputStream(file);
19     FileCopyUtils.copy(inputStream, fileOutputStream);
20     fileOutputStream.close();
21     inputStream.close();
22 }

```

- deploymentId为流程部署ID
- 使用repositoryService的getDeploymentResourceNames方法可以获取指定部署下所有文件的名称
- 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

流程历史信息的查看

流程执行的历史信息保存在activiti的act_hi_*相关的表中，可以通过HistoryService来查看相关的历史记录。

```

1  /**
2   * 查看历史
3   */
4  @Test
5  void findHistoryInfo()
6  {
7      ProcessEngine processEngine = ProcessEngines.getDefaultProcessEngine();
8      HistoryService historyService = processEngine.getHistoryService();
9      List<HistoricActivityInstance> instanceList =
historyService.createHistoricActivityInstanceQuery()

```

```

10         .processInstanceId("2503")
11         .orderByHistoricActivityInstanceStartTime()
12         .asc()
13         .list();
14     for (HistoricActivityInstance historicActivityInstance : instanceList)
15     {
16         log.info(historicActivityInstance.getActivityId());
17         log.info(historicActivityInstance.getActivityName());
18         log.info(historicActivityInstance.getProcessDefinitionId());
19         log.info(historicActivityInstance.getProcessInstanceId());
20     }
21 }

```

```

1 2023-09-26 15:28:08.213 INFO 7888 --- [main]
   .a.ActivitiProcessDeployApplicationTests : 流程定义id:test:1:2503
2 2023-09-26 15:28:08.213 INFO 7888 --- [main]
   .a.ActivitiProcessDeployApplicationTests : 流程实例 id:7507
3 2023-09-26 15:28:08.213 INFO 7888 --- [main]
   .a.ActivitiProcessDeployApplicationTests : 任务负责人: 张三
4 2023-09-26 15:28:08.213 INFO 7888 --- [main]
   .a.ActivitiProcessDeployApplicationTests : 任务名称: 创建申请

```

流程实例

概述

流程实例代表流程定义的执行实例，一个流程实例包括了所有的运行节点，我们可以利用这个对象来了解当前流程实例的进度等信息

用户或程序按照流程定义内容发起一个流程，这就是一个流程实例

流程定义部署在activiti后，就可以在系统中通过activiti去管理该流程的执行，执行流程表示流程的一次执行

比如部署系统出差流程后，如果某用户要申请出差这时就需要执行这个流程，如果另外一个用户也要申请出差则也需要执行该流程，每个执行互不影响，每个执行是单独的流程实例

启动流程实例时，可以指定的businesskey，businesskey为业务标识，通常为业务表的主键，业务标识和流程实例一一对应

查询流程实例

流程在运行过程中可以查询流程实例的状态，当前运行结点等信息

```
1 List<ProcessInstance> list = runtimeService
2     .createProcessInstanceQuery()
3     .processDefinitionKey(processDefinitionKey)
4     .list();
```

关联BusinessKey

在activiti实际应用时，查询流程实例列表时可能要显示出业务系统的一些相关信息

在查询流程实例时，通过businessKey（业务标识）关联查询业务系统的出差单表，查询出差天数等信息

获取BusinessKey:

```
1 String businessKey = processInstance.getBusinessKey();
```

设置BusinessKey:

```
1 //30为BusinessKey
2 ProcessInstance processInstance = runtimeService.
3     startProcessInstanceByKey("test", "30");
```

挂起、激活流程实例

某些情况可能由于流程变更需要将当前运行的流程暂停而不是直接删除，流程暂停后将不会继续执行

全部流程实例挂起

操作流程定义为挂起状态，该流程定义下边所有的流程实例全部暂停

流程定义为挂起状态该流程定义将不允许启动新的流程实例，同时该流程定义下所有的流程实例将全部挂起暂停执行

```
1 //得到当前流程定义的实例是否都为暂停状态
2 boolean suspended = processInstance.isSuspended();
```

```
1 //暂停
2 repositoryService.suspendProcessDefinitionById(processDefinitionId,true,null)
;
```

```
1 //激活
2 repositoryService.activateProcessDefinitionById(processDefinitionId,true,null)
);
```

单个流程实例挂起

操作流程实例对象，针对单个流程执行挂起操作，某个流程实例挂起则此流程不再继续执行，完成该流程实例的当前任务将报异常

```
1 //暂停
2 runtimeService.suspendProcessInstanceById(processDefinitionId);
```

```
1 //激活
2 runtimeService.activateProcessInstanceById(processDefinitionId);
```

个人任务

分配任务负责人

固定分配

在进行业务流程建模时指定固定的任务负责人

ID	sid-32a2b9d4-6749-4d6b-aedb-2fac3815170c
Name	创建申请
Documentation	
Is for compensation	<input type="checkbox"/>
Asynchronous	<input type="checkbox"/>
Assignee	张三
Candidate Users	
Candidate Groups	

表达式分配

由于固定分配方式，任务只管一步一步执行任务，执行到每一个任务将按照 bpmn 的配置去分配任务负责人

UEL 表达式

activiti 支持两个 UEL 表达式：UEL-value 和 UEL-method

UEL-value

ID	sid-32a2b9d4-6749-4d6b-aedb-2fac3815170c
Name	创建申请
Documentation	
Is for compensation	<input type="checkbox"/>
Asynchronous	<input type="checkbox"/>
Assignee	\${assignee0}
Candidate Users	
Candidate Groups	

assignee 这个变量是 activiti 的一个流程变量

	sid-32a2b9d4-6749-4d6b-aedb-2fac3815170c
	创建申请
ion	<input type="checkbox"/>
	<input type="checkbox"/>
	<code>\${user.assignee0}</code>

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

UEL-method

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

ID	sid-32a2b9d4-6749-4d6b-aedb-2fac3815170c
Name	创建申请
Documentation	
Is for compensation	<input type="checkbox"/>
Asynchronous	<input type="checkbox"/>
Assignee	<code>\${userBean.getUserId()}</code>
Candidate Users	
Candidate Groups	

UEL-method与UEL-value结合

示例:

```
${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

示例：

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

编写代码配置负责人

```
1 //设置assignee的取值，用户可以在界面上设置流程的执行
2 Map<String, Object> assigneeMap = new HashMap<>();
3 assigneeMap.put("assignee0", "张三");
4 assigneeMap.put("assignee1", "李经理");
5 runtimeService.startProcessInstanceByKey("myEvection1", assigneeMap);
```

监听器分配

可以使用监听器来完成很多Activiti流程的业务，流程设计时就不需要指定assignee

事件：

- Create：任务创建后触发
- Assignment：任务分配后触发
- Delete：任务完成后触发
- All：所有事件发生都触发

定义任务监听类，且类必须实现 org.activiti.engine.delegate.TaskListener 接口

```
1 public class MyTaskListener implements TaskListener {
2     @Override
3     public void notify(DelegateTask delegateTask) {
4         if(delegateTask.getName().equals("创建出差申请")&&
5             delegateTask.getEventName().equals("create")){
6             //这里指定任务负责人
7             delegateTask.setAssignee("张三");
8         }
9     }
10 }
```


查询任务

```
1 List<Task> taskList = taskService.createTaskQuery()
2     .processDefinitionKey(processDefinitionKey)
3     .includeProcessVariables()
4     .taskAssignee(assignee)
5     .list();
```

办理任务

在实际应用中，完成任务前需要校验任务的负责人是否具有该任务的办理权限

```
1 //根据任务id和任务负责人查询当前任务，如果查到该用户有权限，就完成
2 Task task = taskService.createTaskQuery()
3     .taskId(taskId)
4     .taskAssignee(assignee)
5     .singleResult();
6 if(task != null){
7     taskService.complete(taskId);
8     System.out.println("完成任务");
9 }
```

流程变量

概述

流程变量在 activiti 中是一个非常重要的角色，流程运转有时需要靠流程变量，业务系统和 activiti 结合时少不了流程变量，流程变量就是 activiti 在管理工作流时根据管理需要而设置的变量。

在出差申请流程流转时如果出差天数大于 3 天则由总经理审核，否则由人事直接审核，出差天数就可以设置为流程变量，在流程流转时使用

虽然流程变量中可以存储业务数据可以通过activiti的api查询流程变量从而实现 查询业务数据，但是不建议这样使用，因为业务数据查询由业务系统负责， activiti设置流程变量是为了流程执行需要而创建

流程变量类型

如果将 pojo 存储到流程变量中，必须实现序列化接口 serializable，为了防止由于新增字段无法反序列化，需要生成 serialVersionUID

类型如下：

- string
- integer
- short
- long
- double
- boolean
- date
- binary
- serializable

流程变量作用域

流程变量的作用域可以是一个流程实例(processInstance)，或一个任务(task)，或一个执行实例(execution)

global变量

流程变量的默认作用域是流程实例。当一个流程变量的作用域为流程实例时，可以称为 global 变量
global 变量中变量名不允许重复，设置相同名称的变量，后设置的值会覆盖前设置的变量值

local变量

任务和执行实例仅仅是针对一个任务和一个执行实例范围，范围没有流程实例大，称为 local 变量

Local 变量由于在不同的任务或不同的执行实例中，作用域互不影响，变量名可以相同没有影响。Local 变量名也可以和 global 变量名相同，没有影响

使用方法

在属性上使用UEL表达式

可以在 assignee 处设置 UEL 表达式，表达式的值为任务的负责人，比如：\${assignee}，assignee 就是一个流程变量名称

在连线上使用UEL表达式

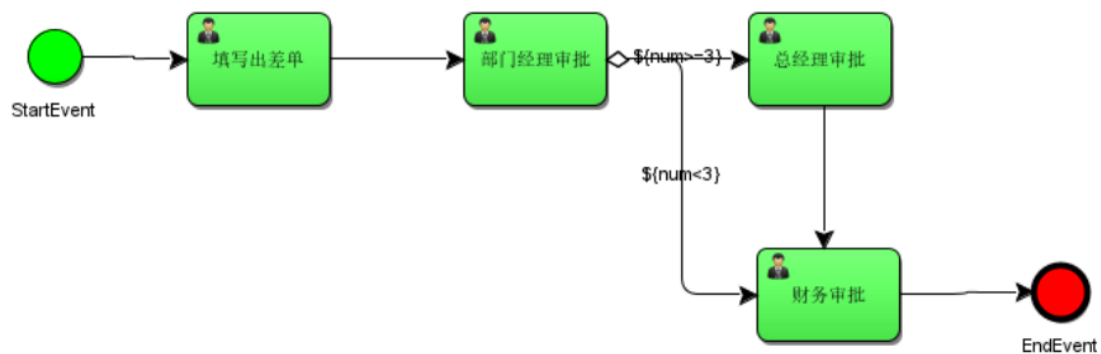
可以在连线上设置UEL表达式，决定流程走向

比如：\${price<10000}。price就是一个流程变量名称，uel表达式结果类型为布尔类型

使用Global变量控制流程

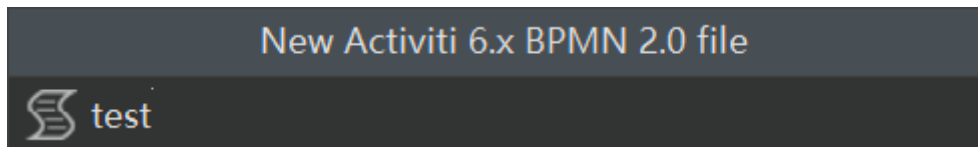
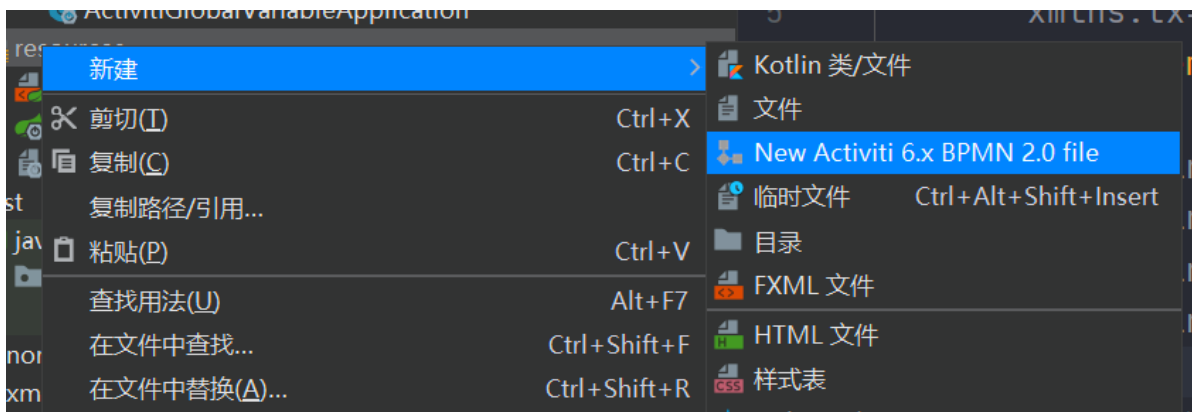
需求

员工创建出差申请单，由部门经理审核，部门经理审核通过后出差3天及以下由人财务直接审批，3天以上先由总经理审核，总经理审核通过再由财务审批

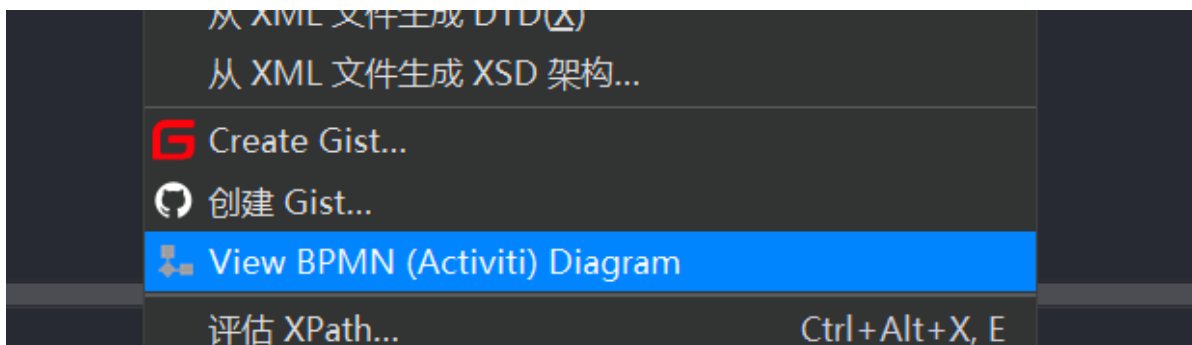


流程定义

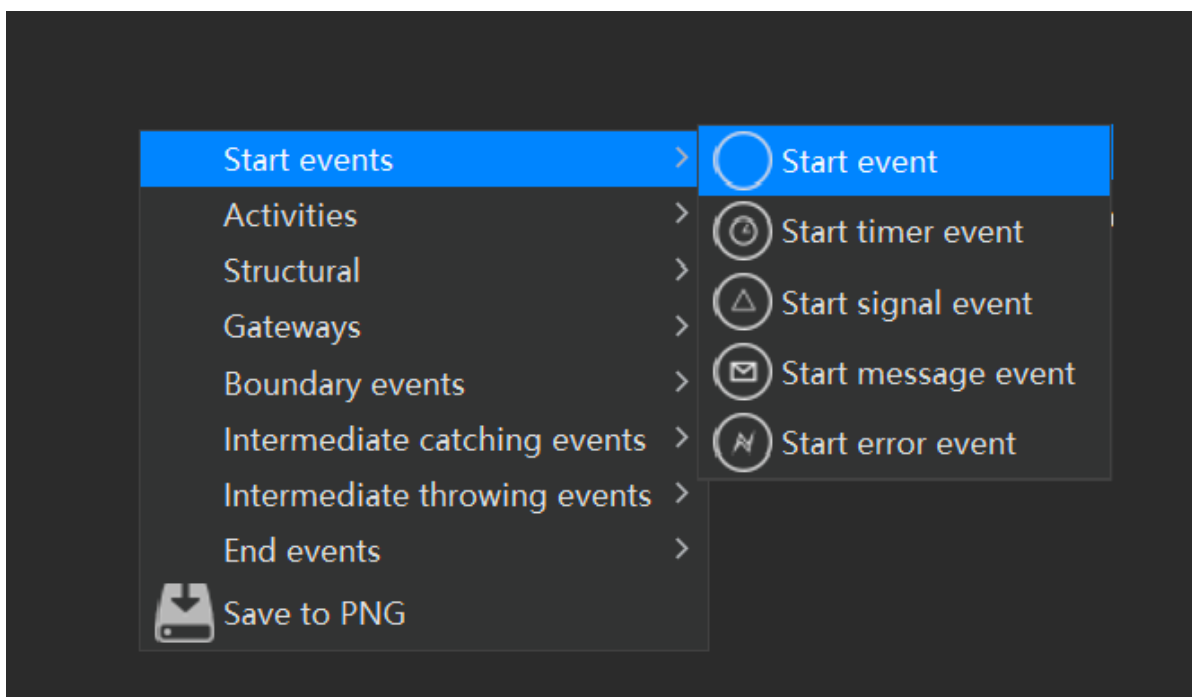
创建：



右键点击设计

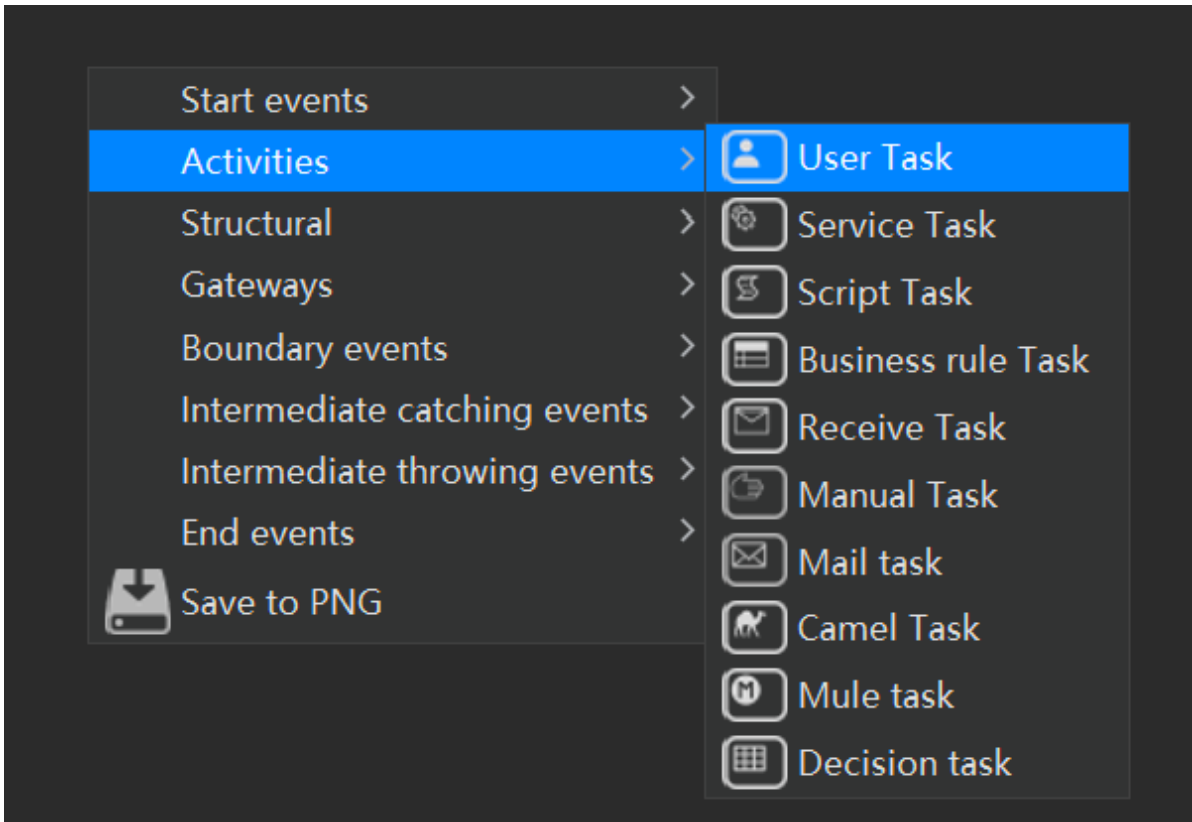


添加开始事件





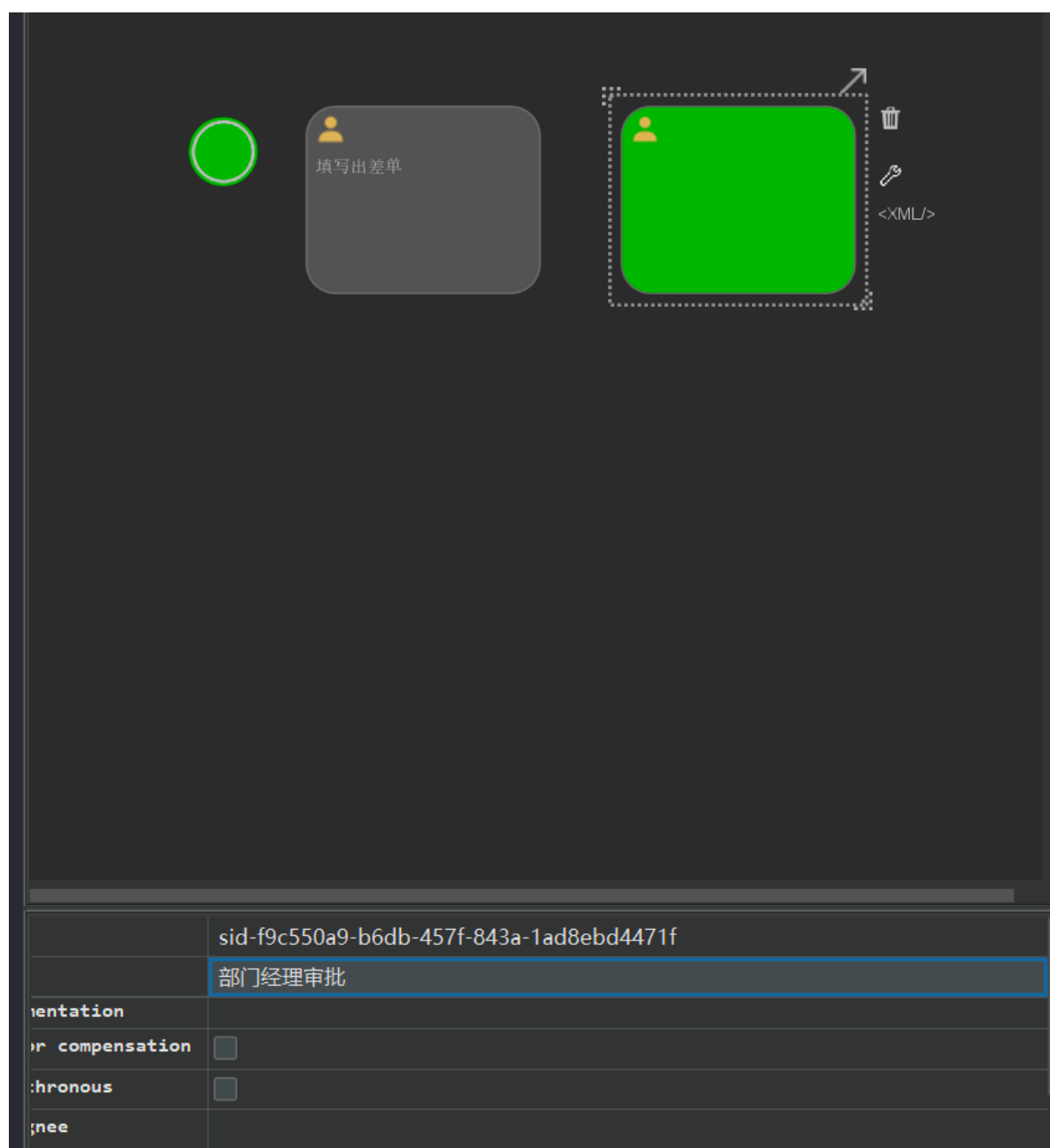
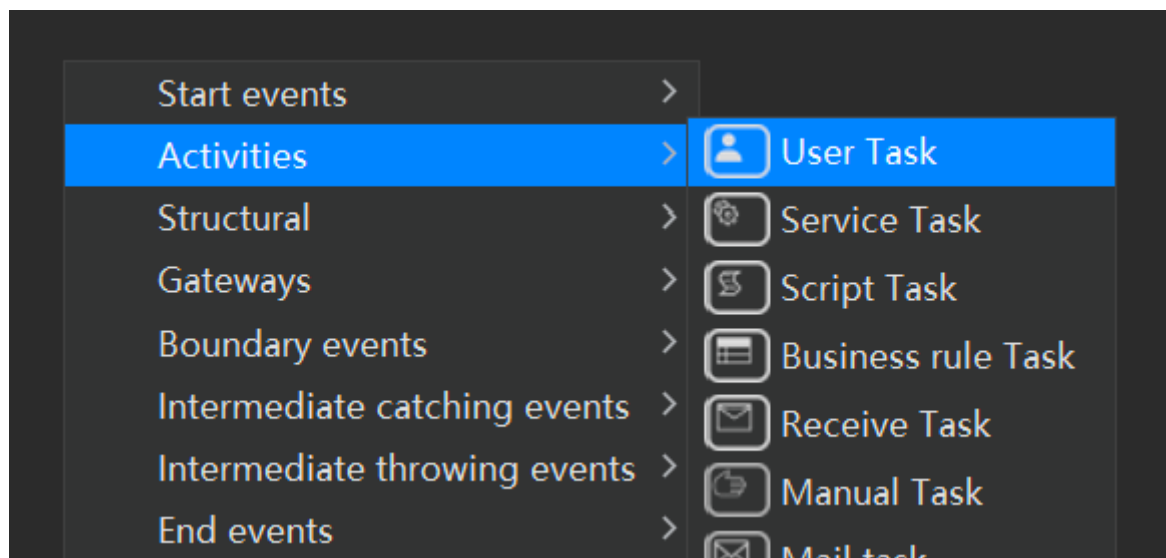
添加填写出差单



填写名称

ID	sid-63f05c99-05c9-4735-81d0-0318f3672601
Name	填写出差单
Documentation	
Is for compensation	<input type="checkbox"/>
Asynchronous	<input type="checkbox"/>

添加部门经理审批



添加总经理审批



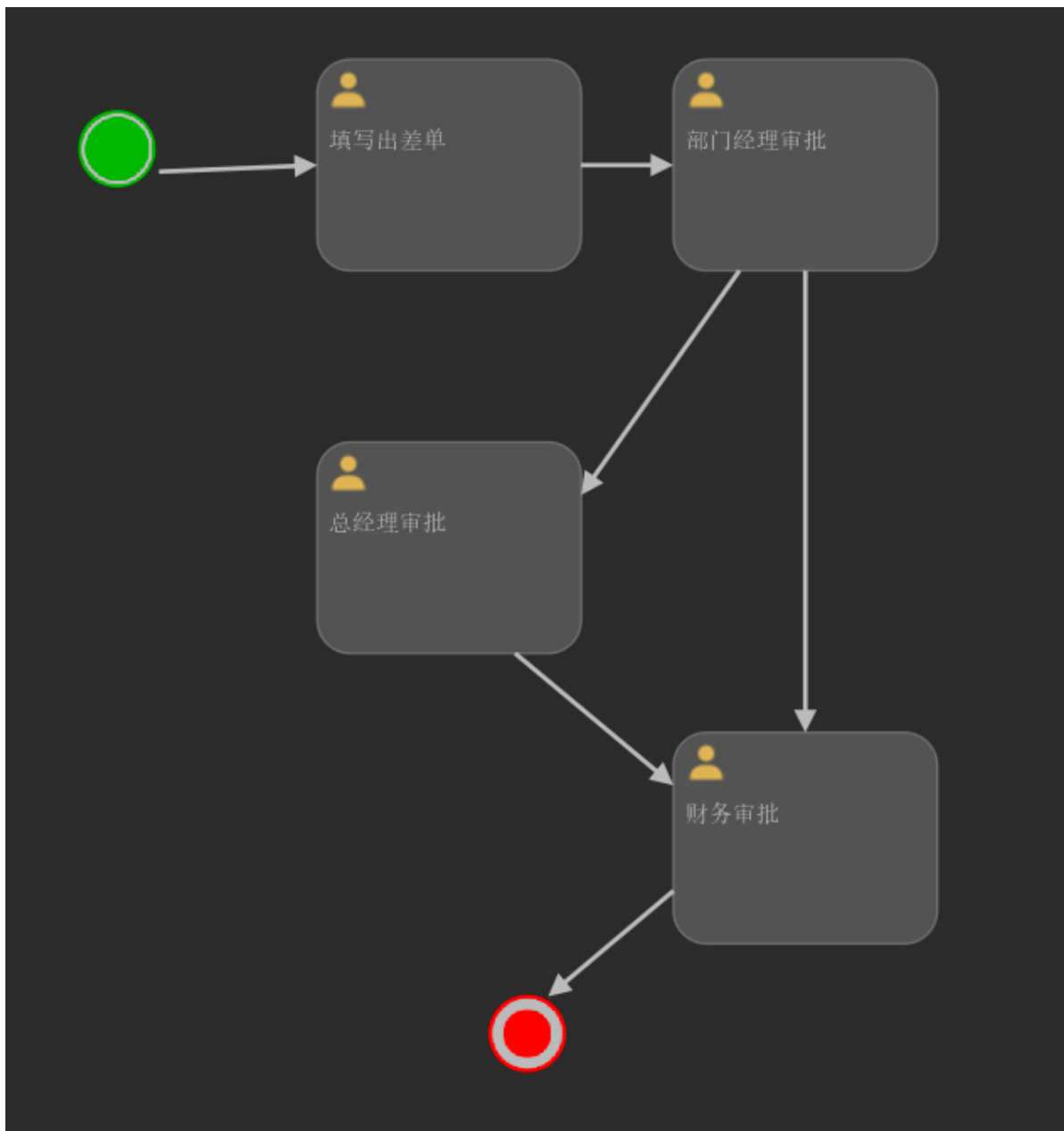
添加财务审批



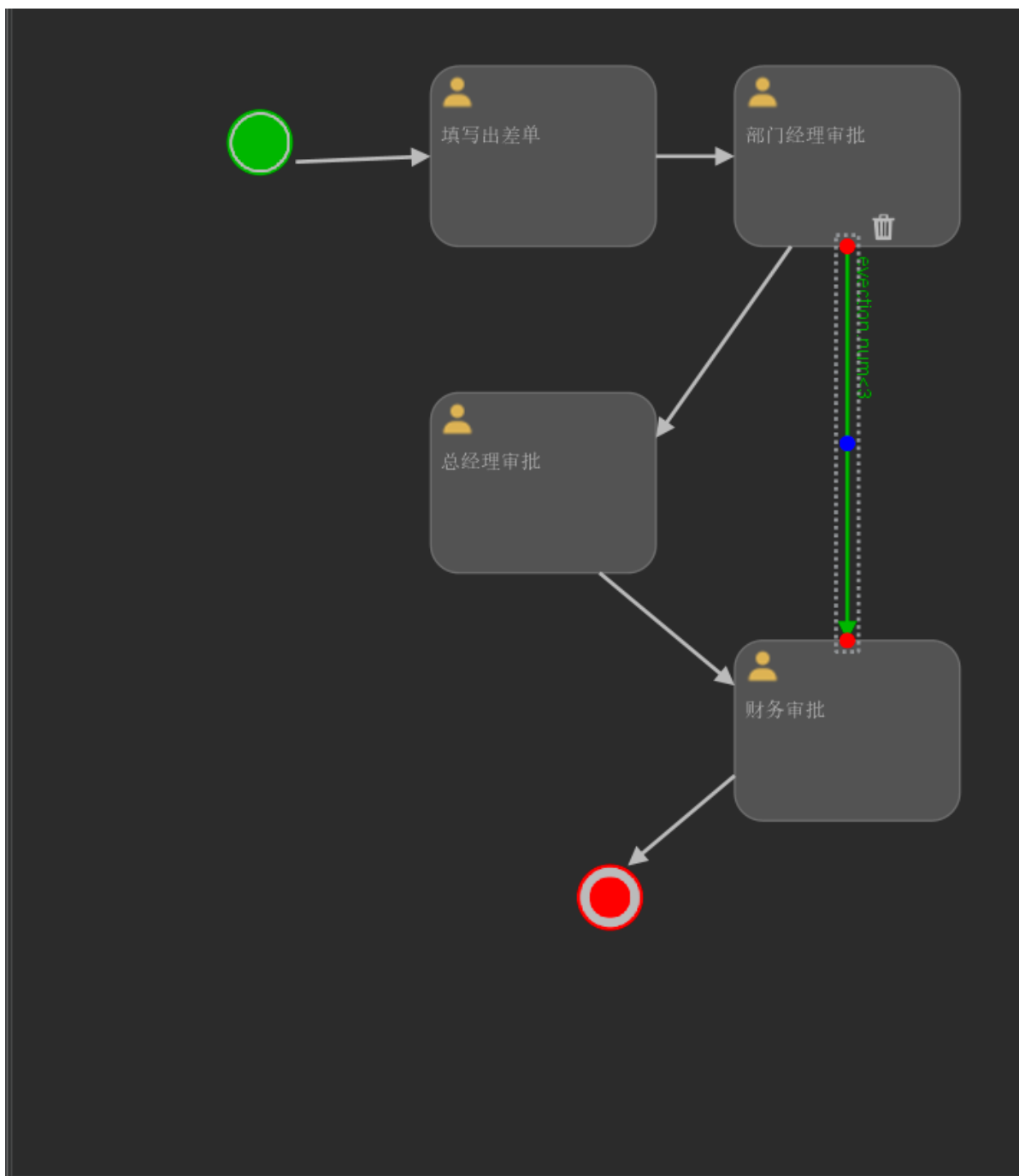
结束



连线



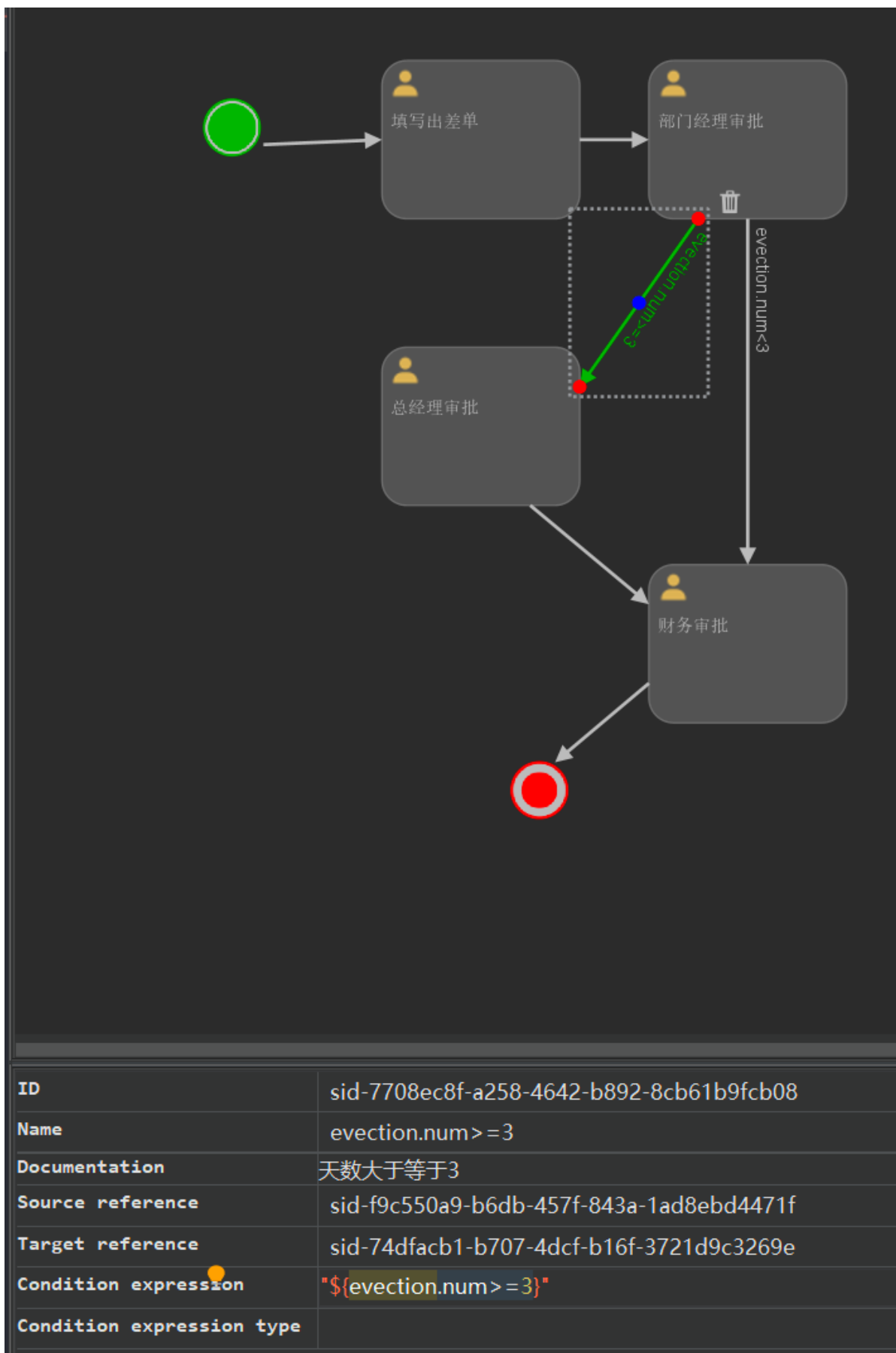
填写出差天数小于3连线条件



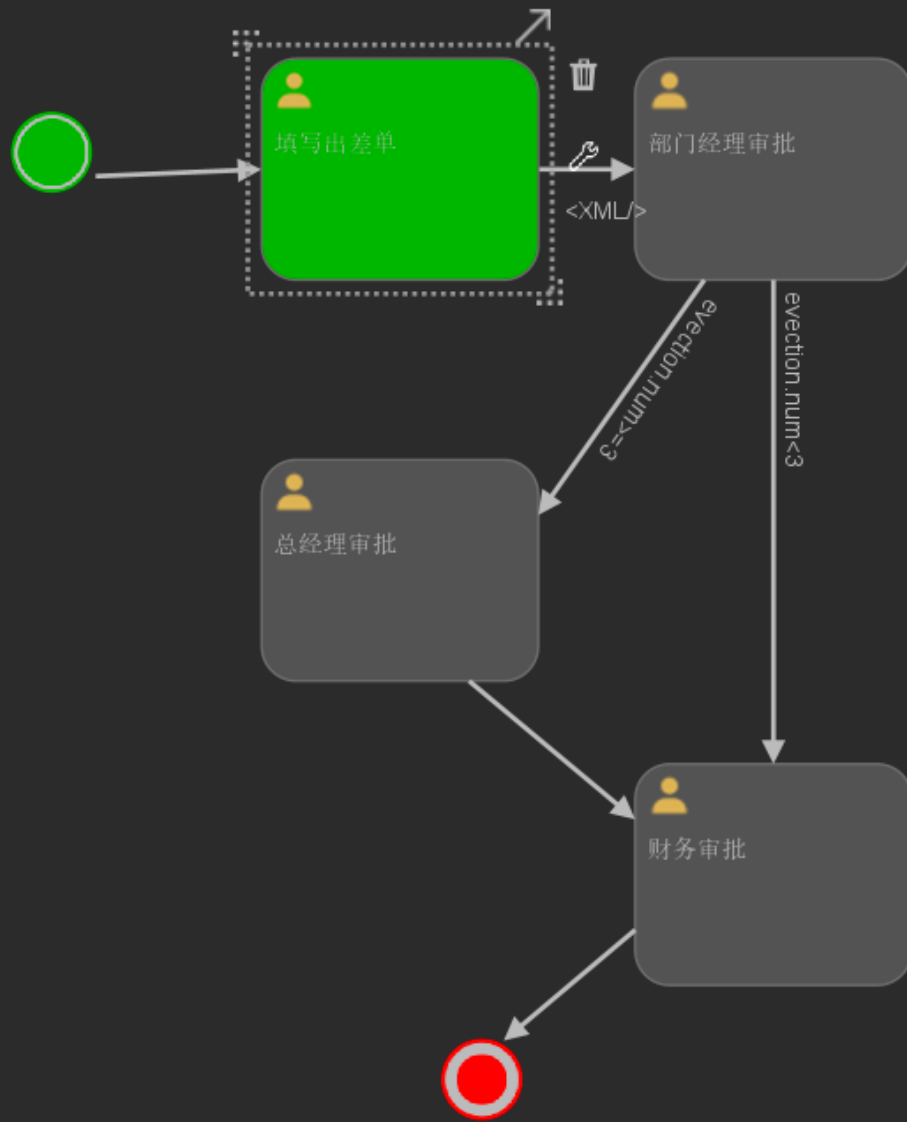
ID	sid-c64e0ebc-3803-418f-8e6f-be662d0d2586
Name	evection.num<3
Documentation	天数小于3天
Source reference	sid-f9c550a9-b6db-457f-843a-1ad8ebd4471f
Target reference	sid-c4d7ddf1-79c5-4890-a9bc-daae67dd603a
Condition expression	"\${evection.num<3}"
Condition expression type	

evection为一个对象，num为evection对象里的num属性

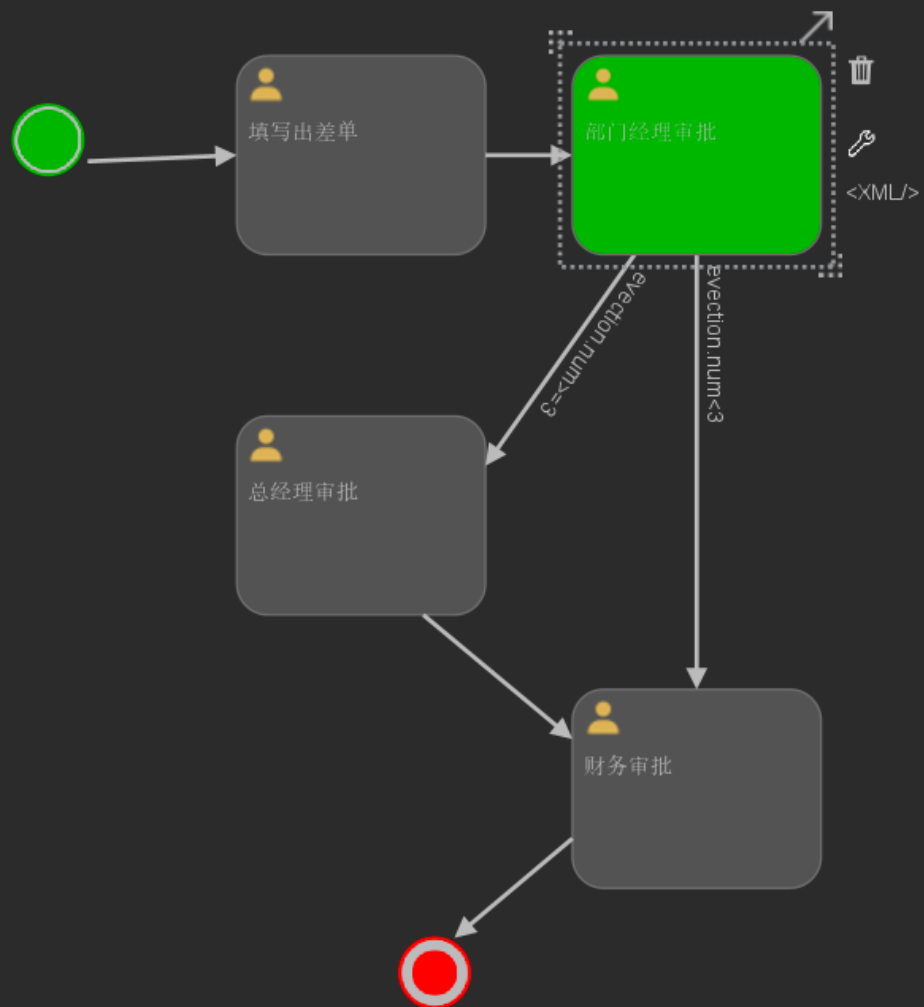
填写出差天数大于等于3连线条件



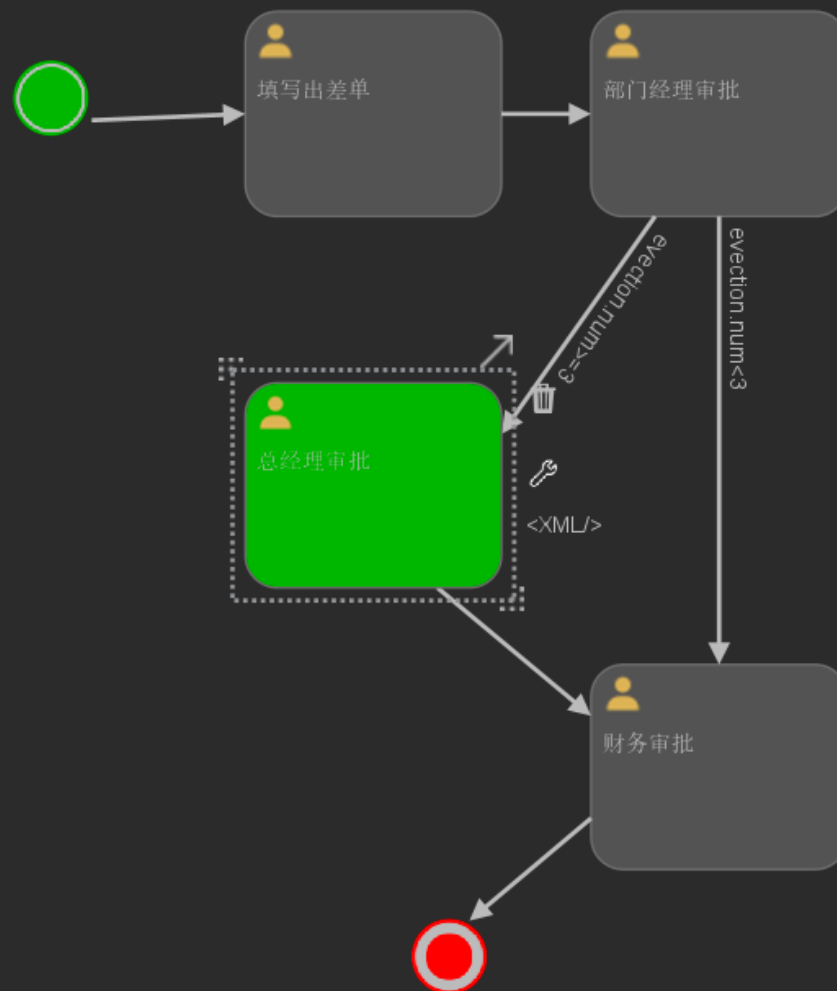
填写对应的assignee



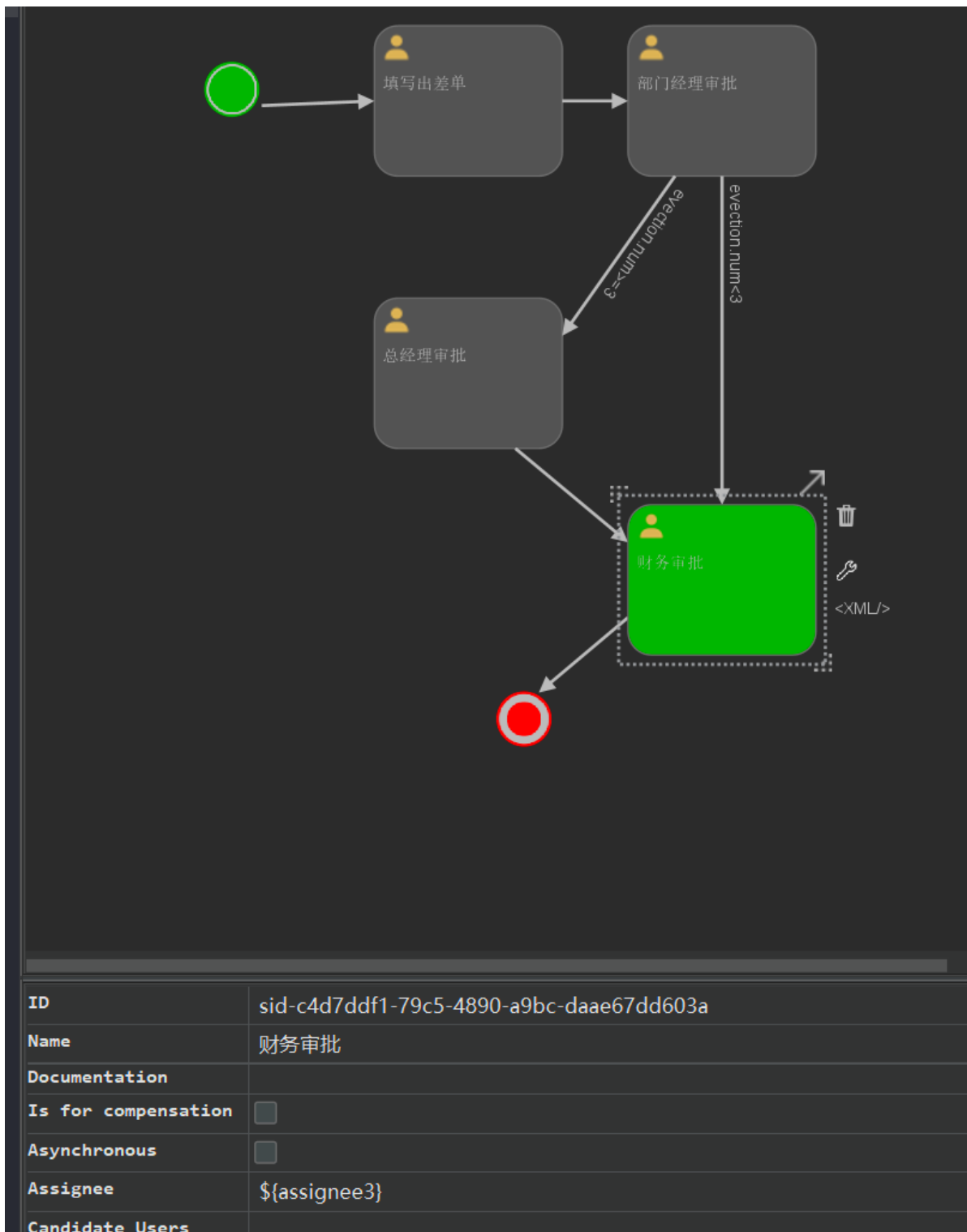
id	sid-63f05c99-05c9-4735-81d0-0318f3672601
name	填写出差单
documentation	
for compensation	<input type="checkbox"/>
asynchronous	<input type="checkbox"/>
assignee	\${assignee0}
candidate Users	



ID	sid-f9c550a9-b6db-457f-843a-1ad8ebd4471f
Name	部门经理审批
Documentation	
Is for compensation	<input type="checkbox"/>
Asynchronous	<input type="checkbox"/>
Assignee	\${assignee1}
Candidate Users	



ID	sid-74dfacb1-b707-4dcf-b16f-3721d9c3269e
Name	总经理审批
Documentation	
Is for compensation	<input type="checkbox"/>
Asynchronous	<input type="checkbox"/>
Assignee	\${assignee2}
Candidate Users	



xml文件如下:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
5   xmlns:activiti="http://activiti.org/bpmn"
6   xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
   xmlns:omgdc="http://www.omg.org/spec/DD/20100524/DC"

```



```

7         xmlns:omgdi="http://www.omg.org/spec/DD/20100524/DI"
8         typeLanguage="http://www.w3.org/2001/XMLSchema"
9         expressionLanguage="http://www.w3.org/1999/XPath"
10        targetNamespace="http://www.activiti.org/processdef">
11        <process id="test" name="test" isExecutable="true">
12            <startEvent id="sid-a3f22bc2-2624-4c15-b2b9-05bc4b491ec9" name="填写
出差单"/>
13            <userTask id="sid-63f05c99-05c9-4735-81d0-0318f3672601" name="填写出
出差单" activiti:assignee="${assignee0}"/>
14            <userTask id="sid-f9c550a9-b6db-457f-843a-1ad8ebd4471f" name="部门经
理审批" activiti:assignee="${assignee1}"/>
15            <userTask id="sid-74dfacb1-b707-4dcf-b16f-3721d9c3269e" name="总经理
审批" activiti:assignee="${assignee2}"/>
16            <userTask id="sid-c4d7ddf1-79c5-4890-a9bc-daae67dd603a" name="财务审
批" activiti:assignee="${assignee3}"/>
17            <endEvent id="sid-62c7a357-5389-4abf-92d9-37bf4daebe07"/>
18            <sequenceFlow id="sid-58662137-3afe-4833-a23c-dc7d7640e4ce"
sourceRef="sid-a3f22bc2-2624-4c15-b2b9-05bc4b491ec9"
19                targetRef="sid-63f05c99-05c9-4735-81d0-0318f3672601"/>
20            <sequenceFlow id="sid-483a3633-566c-4d48-9450-1e792d0e398b"
sourceRef="sid-c4d7ddf1-79c5-4890-a9bc-daae67dd603a"
21                targetRef="sid-62c7a357-5389-4abf-92d9-37bf4daebe07"/>
22            <sequenceFlow id="sid-df9f0725-0328-4029-a5a5-55ff96e86a11"
sourceRef="sid-74dfacb1-b707-4dcf-b16f-3721d9c3269e"
23                targetRef="sid-c4d7ddf1-79c5-4890-a9bc-daae67dd603a"/>
24            <sequenceFlow id="sid-0093c9ff-43cf-4dbb-967c-a3d01b945b6b"
sourceRef="sid-63f05c99-05c9-4735-81d0-0318f3672601"
25                targetRef="sid-f9c550a9-b6db-457f-843a-1ad8ebd4471f"/>
26            <sequenceFlow id="sid-7708ec8f-a258-4642-b892-8cb61b9fcb08"
sourceRef="sid-f9c550a9-b6db-457f-843a-1ad8ebd4471f"
27                targetRef="sid-74dfacb1-b707-4dcf-b16f-3721d9c3269e"
name="evection.num>=3">
28                <documentation>天数大于等于3</documentation>
29                <conditionExpression>${evection.num>=3}</conditionExpression>
30            </sequenceFlow>
31            <sequenceFlow id="sid-c64e0ebc-3803-418f-8e6f-be662d0d2586"
sourceRef="sid-f9c550a9-b6db-457f-843a-1ad8ebd4471f"
32                targetRef="sid-c4d7ddf1-79c5-4890-a9bc-daae67dd603a"
name="evection.num<3">
33                <documentation>天数小于3天</documentation>
34                <conditionExpression>${evection.num<3}</conditionExpression>
35            </sequenceFlow>
36        </process>
37        <bpmndi:BPMNDiagram id="BPMNDiagram_test">
38            <bpmndi:BPMNPlane bpmnElement="test" id="BPMNPlane_test">
39                <bpmndi:BPMNShape id="shape-9ad45617-e012-402b-94a6-
d845aa07d2e6"
40                    bpmnElement="sid-a3f22bc2-2624-4c15-b2b9-
05bc4b491ec9">
41                    <omgdc:Bounds x="-175.0" y="-85.0" width="30.0"
height="30.0"/>
42                </bpmndi:BPMNShape>
43                <bpmndi:BPMNShape id="shape-842e04c7-ca37-4e6c-843c-
a41d844c1d28"

```

```

44         bpmnElement="sid-63f05c99-05c9-4735-81d0-
0318f3672601">
45         <omgdc:Bounds x="-85.0" y="-105.0" width="100.0"
height="80.0"/>
46     </bpmndi:BPMNShape>
47     <bpmndi:BPMNShape id="shape-c03cede2-a022-45c5-9474-
31117a97b616"
48         bpmnElement="sid-f9c550a9-b6db-457f-843a-
1ad8ebd4471f">
49         <omgdc:Bounds x="50.0" y="-105.0" width="100.0"
height="80.0"/>
50     </bpmndi:BPMNShape>
51     <bpmndi:BPMNShape id="shape-c0458a26-653b-4502-ba86-
a29dc6e18bd2"
52         bpmnElement="sid-74dfacb1-b707-4dcf-b16f-
3721d9c3269e">
53         <omgdc:Bounds x="-85.0" y="40.0" width="100.0"
height="80.0"/>
54     </bpmndi:BPMNShape>
55     <bpmndi:BPMNShape id="shape-c2df426e-2d55-44c8-9b9e-
4d2a3f1cd772"
56         bpmnElement="sid-c4d7ddf1-79c5-4890-a9bc-
daae67dd603a">
57         <omgdc:Bounds x="50.0" y="150.0" width="100.0"
height="80.0"/>
58     </bpmndi:BPMNShape>
59     <bpmndi:BPMNShape id="shape-294292d1-05bc-400a-8657-
849239dcaf7a"
60         bpmnElement="sid-62c7a357-5389-4abf-92d9-
37bf4daebe07">
61         <omgdc:Bounds x="-20.0" y="250.0" width="30.0"
height="30.0"/>
62     </bpmndi:BPMNShape>
63     <bpmndi:BPMNEdge id="edge-4eef764e-e601-4274-be70-7a2983007bbf"
64         bpmnElement="sid-58662137-3afe-4833-a23c-
dc7d7640e4ce">
65         <omgdi:waypoint x="-145.0" y="-62.5"/>
66         <omgdi:waypoint x="-85.0" y="-65.0"/>
67     </bpmndi:BPMNEdge>
68     <bpmndi:BPMNEdge id="edge-9df9418d-bc3b-425d-baa0-374a759fe404"
69         bpmnElement="sid-483a3633-566c-4d48-9450-
1e792d0e398b">
70         <omgdi:waypoint x="50.0" y="210.0"/>
71         <omgdi:waypoint x="2.5" y="250.0"/>
72     </bpmndi:BPMNEdge>
73     <bpmndi:BPMNEdge id="edge-40b8582a-e79e-4a75-8029-67d6edeb368d"
74         bpmnElement="sid-df9f0725-0328-4029-a5a5-
55ff96e86a11">
75         <omgdi:waypoint x="-10.0" y="120.0"/>
76         <omgdi:waypoint x="50.0" y="170.0"/>
77     </bpmndi:BPMNEdge>
78     <bpmndi:BPMNEdge id="edge-b5366cd9-6f9d-4fbd-95ca-9ca6e4218ba4"
79         bpmnElement="sid-0093c9ff-43cf-4dbb-967c-
a3d01b945b6b">
80         <omgdi:waypoint x="15.0" y="-65.0"/>

```

```

81         <omgdi:waypoint x="50.0" y="-65.0"/>
82     </bpmndi:BPMNEdge>
83     <bpmndi:BPMNEdge id="edge-a1292cd8-69ea-4370-90df-dd9fa3baa829"
84         bpmnElement="sid-7708ec8f-a258-4642-b892-
8cb61b9fcb08">
85         <omgdi:waypoint x="75.0" y="-25.0"/>
86         <omgdi:waypoint x="15.0" y="60.0"/>
87     </bpmndi:BPMNEdge>
88     <bpmndi:BPMNEdge id="edge-ccd02057-64b3-447b-adc2-197a25653b2e"
89         bpmnElement="sid-c64e0ebc-3803-418f-8e6f-
be662d0d2586">
90         <omgdi:waypoint x="100.0" y="-25.0"/>
91         <omgdi:waypoint x="100.0" y="150.0"/>
92     </bpmndi:BPMNEdge>
93 </bpmndi:BPMNPlane>
94 </bpmndi:BPMNDiagram>
95 </definitions>
96

```

创建实体类Evection

```

1  package mao.activiti_global_variable.entity;
2
3  import java.io.Serializable;
4  import java.util.Date;
5  import java.util.StringJoiner;
6
7  /**
8   * Project name(项目名称): activiti-global-variable
9   * Package(包名): mao.activiti_global_variable.entity
10  * Class(类名): Evection
11  * Author(作者): mao
12  * Author QQ: 1296193245
13  * GitHub: https://github.com/maomao124/
14  * Date(创建日期): 2023/10/7
15  * Time(创建时间): 16:01
16  * Version(版本): 1.0
17  * Description(描述): 无
18  */
19
20  public class Evection implements Serializable
21  {
22      /**
23       * 主键id
24       */
25      private Long id;
26      /**
27       * 出差申请单名称
28       */

```

```

29     private String evectionName;
30     /**
31      * 出差天数
32      */
33     private Double num;
34     /**
35      * 预计开始时间
36      */
37     private Date beginDate;
38     /**
39      * 预计结束时间
40      */
41     private Date endDate;
42     /**
43      * 目的地
44      */
45     private String destination;
46     /**
47      * 出差事由
48      */
49     private String reson;
50
51     public Long getId()
52     {
53         return id;
54     }
55
56     public void setId(Long id)
57     {
58         this.id = id;
59     }
60
61     public String getEvectionName()
62     {
63         return evectionName;
64     }
65
66     public void setEvectionName(String evectionName)
67     {
68         this.evectionName = evectionName;
69     }
70
71     public Date getBeginDate()
72     {
73         return beginDate;
74     }
75
76     public void setBeginDate(Date beginDate)
77     {
78         this.beginDate = beginDate;
79     }
80
81     public Date getEndDate()
82     {
83         return endDate;

```

```

84     }
85
86     public void setEndDate(Date endDate)
87     {
88         this.endDate = endDate;
89     }
90
91     public String getDestination()
92     {
93         return destination;
94     }
95
96     public void setDestination(String destination)
97     {
98         this.destination = destination;
99     }
100
101     public String getReson()
102     {
103         return reson;
104     }
105
106     public void setReson(String reson)
107     {
108         this.reson = reson;
109     }
110
111     public Double getNum()
112     {
113         return num;
114     }
115
116     public void setNum(Double num)
117     {
118         this.num = num;
119     }
120
121     @Override
122     public String toString()
123     {
124         return new StringJoiner(", ", Evection.class.getSimpleName() + "[",
125 "]"")
126             .add("id=" + id)
127             .add("evectionName=" + evectionName + "'")
128             .add("num=" + num)
129             .add("beginDate=" + beginDate)
130             .add("endDate=" + endDate)
131             .add("destination=" + destination + "'")
132             .add("reson=" + reson + "'")
133             .toString();
134     }
135 }

```

启动流程时设置变量

在启动流程时设置流程变量，变量的作用域是整个流程实例

通过Map<key,value>设置流程变量，map中可以设置多个变量，这个key就是流程变量的名字

```
1  @Test
2      void startProcess()
3      {
4          ProcessEngine processEngine =
5      ProcessEngines.getDefaultProcessEngine();
6          RuntimeService runtimeService = processEngine.getRuntimeService();
7          //变量集合
8          Map<String, Object> map = new HashMap<>();
9          //创建出差实体类
10         Evection evection = new EvECTION();
11         evection.setNum(2d);
12         //实体类
13         map.put("evection", evection);
14         //人名参数
15         map.put("assignee0", "张三");
16         map.put("assignee1", "李经理");
17         map.put("assignee2", "王总经理");
18         map.put("assignee3", "赵财务");
19         //启动流程实例
20         ProcessInstance processInstance =
21     runtimeService.startProcessInstanceByKey("test", map);
22         log.info("id:" + processInstance.getId());
23         log.info("名称: " + processInstance.getName());
24     }
```

张三提交任务

```
1  /**
2      * 张三提交任务
3      */
4      @Test
5      void completeTask1()
6      {
7          String assignee = "张三";
8          ProcessEngine processEngine =
9      ProcessEngines.getDefaultProcessEngine();
10         TaskService taskService = processEngine.getTaskService();
11         Task task = taskService.createTaskQuery()
```

```

11         .processDefinitionKey("test")
12         .taskAssignee(assinee)
13         .orderByTaskCreateTime()
14         .desc()
15         .list()
16         .get(0);
17     if (task == null)
18     {
19         log.info("无权操作");
20     }
21     else
22     {
23         taskService.complete(task.getId());
24         log.info("任务执行完成");
25     }
26 }

```

任务办理时设置变量

在完成任务时设置流程变量，该流程变量只有在该任务完成后其它结点才可使用该变量，它的作用域是整个流程实例，如果设置的流程变量的key在流程实例中已存在相同的名字则后设置的变量替换前边设置的变量

```

1  /**
2   * 张三提交任务，任务办理时设置变量
3   */
4  @Test
5  void completeTask1_1()
6  {
7      String assinee = "张三";
8      ProcessEngine processEngine =
9      ProcessEngines.getDefaultProcessEngine();
10     TaskService taskService = processEngine.getTaskService();
11     Task task = taskService.createTaskQuery()
12         .processDefinitionKey("test")
13         .taskAssignee(assinee)
14         .orderByTaskCreateTime()
15         .desc()
16         .list()
17         .get(0);
18     if (task == null)
19     {
20         log.info("无权操作");
21     }
22     else
23     {
24         Map<String, Object> map = new HashMap<>();
25         //创建出差实体类
26         Evection evection = new Evection();

```

```

26         evection.setNum(2d);
27         //实体类
28         map.put("evection", evection);
29         taskService.complete(task.getId(), map);
30         log.info("任务执行完成");
31     }
32 }

```

通过当前任务设置流程变量，需要指定当前任务id，如果当前执行的任务id不存在则抛出异常

任务办理时也是通过map<key,value>设置流程变量，一次可以设置多个变量

通过当前流程实例设置

通过流程实例id设置全局变量，该流程实例必须未执行完成

```

1  /**
2      * 通过当前流程实例设置
3      */
4  @Test
5  void setGlobalVariableByExecutionId()
6  {
7      ProcessEngine processEngine =
8      ProcessEngines.getDefaultProcessEngine();
9      RuntimeService runtimeService = processEngine.getRuntimeService();
10     Evection evection = new Evection();
11     evection.setNum(3d);
12     runtimeService.setVariable("7501", "evection", evection);

```

executionId必须当前未结束流程实例的执行id，通常此id设置流程实例的id。也可以通过runtimeService.getVariable()获取流程变量

通过当前任务设置


```

1  @Test
2      void setGlobalVariableByTaskId()
3      {
4          ProcessEngine processEngine =
ProcessEngines.getDefaultProcessEngine();
5          TaskService taskService = processEngine.getTaskService();
6          Evection evection = new Eviction();
7          evection.setNum(3d);
8          taskService.setVariable("2501", "evection", evection);
9      }

```

任务id必须是当前待办任务id, act_ru_task中存在

也可以通过taskService.getVariable()获取流程变量

注意事项

- 如果UEL表达式中流程变量名不存在则报错
- 如果UEL表达式中流程变量值为空NULL, 流程不按UEL表达式去执行, 而流程结束
- 如果UEL表达式都不符合条件, 流程结束
- 如果连线不设置条件, 会走flow序号小的那条线

设置local流程变量

任务办理时设置

任务办理时设置local流程变量, 当前运行的流程实例只能在该任务结束前使用, 任务结束该变量无法在当前流程实例使用, 可以通过查询历史任务查询

```

1  /**
2      * 任务办理时设置local流程变量
3      */
4      @Test
5      void completeTask()
6      {
7          ProcessEngine processEngine =
ProcessEngines.getDefaultProcessEngine();
8          TaskService taskService = processEngine.getTaskService();
9          Map<String, Object> map = new HashMap<>();
10         Evection evection = new EvECTION();
11         evection.setNum(3d);
12         taskService.setVariablesLocal("2501", map);
13     }

```

设置作用域为任务的local变量，每个任务可以设置同名的变量，互不影响

通过当前任务设置

```

1  /**
2      * 通过当前任务设置local流程变量
3      */
4      @Test
5      void setLocalVariableByTaskId()
6      {
7          ProcessEngine processEngine =
ProcessEngines.getDefaultProcessEngine();
8          TaskService taskService = processEngine.getTaskService();
9          EvECTION evection = new EvECTION ();
10         evection.setNum(3d);
11         taskService.setVariableLocal("2501", "evection", evection);
12     }

```

组任务

概述

在流程定义中在任务结点的 assignee 固定设置任务负责人，在流程定义时将参与者固定设置在.bpmn 文件中，如果临时任务负责人变更则需要修改流程定义，系统可扩展性差

针对这种情况可以给任务设置多个候选人，可以从候选人中选择参与者来完成任务

设置任务候选人

在流程图中任务节点的配置中设置 candidate-users(候选人)，多个候选人之间用逗号分开

ID	sid-74dfacb1-b707-4dcf-b16f-3721d9c3269e
Name	总经理审批
Documentation	
Is for compensation	<input type="checkbox"/>
Asynchronous	<input type="checkbox"/>
Assignee	\${assignee2}
Candidate Users	user1,user2,user3
Candidate Groups	

```
1 <userTask id="sid-74dfacb1-b707-4dcf-b16f-3721d9c3269e" name="总经理审批"
  activiti:assignee="${assignee2}"
  activiti:candidateUsers="user1,user2,user3"/>
```

查询组任务

根据候选人查询组任务

```
1 @Test
2 void findGroupTaskList()
3 {
4     ProcessEngine processEngine =
5     ProcessEngines.getDefaultProcessEngine();
6     TaskService taskService = processEngine.getTaskService();
7     List<Task> list = taskService.createTaskQuery()
8         .processDefinitionKey("test")
9         .taskCandidateUser("user2")//根据候选人查询
10        .list();
```

```

10         for (Task task : list)
11         {
12             log.info("流程实例id: " + task.getProcessInstanceId());
13             log.info("任务id: " + task.getId());
14             log.info("任务负责人: " + task.getAssignee());
15             log.info("任务名称: " + task.getName());
16         }
17     }

```

拾取组任务

候选人员拾取组任务后该任务变为自己的个人任务

即使该用户不是候选人也能拾取，建议拾取时校验是否有资格

组任务拾取后，该任务已有负责人，通过候选人将查询不到该任务

```

1  @Test
2  void claimTask()
3  {
4      ProcessEngine processEngine =
5      ProcessEngines.getDefaultProcessEngine();
6      TaskService taskService = processEngine.getTaskService();
7      //校验该用户有没有拾取任务的资格
8      Task task = taskService.createTaskQuery()
9          .taskId("7501")
10         .taskCandidateUser("user2")//根据候选人查询
11         .singleResult();
12     if (task==null)
13     {
14         log.info("user2无资格");
15     }
16     else
17     {
18         taskService.claim("7501", "user2");
19         log.info("user2 任务拾取成功");
20     }
21 }

```

查询个人待办任务

查询方式同个人任务查询

```

1  @Test
2  void findPersonalTaskList()
3  {

```

```

4      ProcessEngine processEngine =
ProcessEngines.getDefaultProcessEngine();
5      TaskService taskService = processEngine.getTaskService();
6      List<Task> list = taskService.createTaskQuery()
7          .processDefinitionKey("test")
8          .taskAssignee("user2")
9          .list();
10     for (Task task : list)
11     {
12         log.info("流程实例id: " + task.getProcessInstanceId());
13         log.info("任务id: " + task.getId());
14         log.info("任务负责人: " + task.getAssignee());
15         log.info("任务名称: " + task.getName());
16     }
17 }

```

办理个人任务

```

1  @Test
2      void completeTask()
3      {
4          ProcessEngine processEngine =
ProcessEngines.getDefaultProcessEngine();
5          processEngine.getTaskService()
6              .complete("7501");
7      }

```

归还组任务

如果个人不想办理该组任务，可以归还组任务，归还后该用户不再是该任务的负责人

```

1  @Test
2      void setAssigneeToGroupTask()
3      {
4          ProcessEngine processEngine =
ProcessEngines.getDefaultProcessEngine();
5          TaskService taskService = processEngine.getTaskService();
6          Task task = taskService
7              .createTaskQuery()
8              .taskId("7501")
9              .taskAssignee("user2")
10             .singleResult();
11         if (task != null)
12         {
13             // 如果设置为null，归还组任务，该任务没有负责人

```

```
14         taskService.setAssignee("7501", null);
15     }
16 }
```

任务交接

任务负责人将任务交给其它候选人办理该任务

```
1  @Test
2      void setAssigneeToCandidateUser()
3      {
4          ProcessEngine processEngine =
ProcessEngines.getDefaultProcessEngine();
5          TaskService taskService = processEngine.getTaskService();
6          Task task = taskService
7              .createTaskQuery()
8              .taskId("7501")
9              .taskAssignee("user2")
10             .singleResult();
11         if (task!=null)
12         {
13             taskService.setAssignee("7501", "user3");
14         }
15     }
```

网关

排他网关

概述

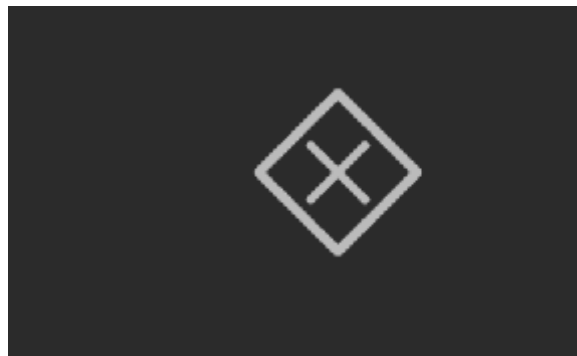
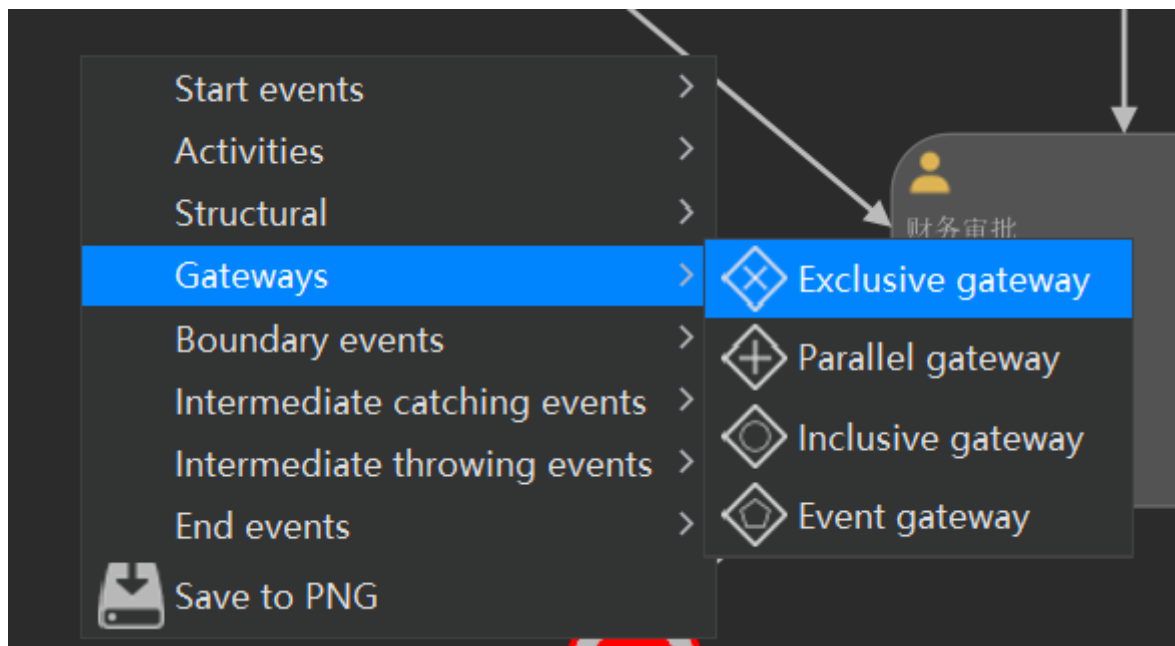
ExclusiveGateway(排他网关)，用来在流程中实现决策。当流程执行到这个网关，所有分支都会判断条件是否为true，如果为true则执行该分支

排他网关只会选择一个为true的分支执行。如果有两个分支条件都为true，排他网关会选择id值较小的一条分支去执行

不用排他网关也可以实现分支，如：在连线的condition条件上设置分支条件。

在连线设置condition条件的缺点：如果条件都不满足，流程就结束了(是异常结束)。

流程定义



并行网关

概述

parallelGateway(并行网关)，并行网关允许将流程分成多条分支，也可以把多条分支汇聚到一起，并行网关的功能是基于进入和外出顺序流的

- fork分支：并行后的所有外出顺序流，为每个顺序流都创建一个并发分支
- join汇聚：所有到达并行网关，在此等待的进入分支，直到所有进入顺序流的分支都到达以后，流程就会通过汇聚网关

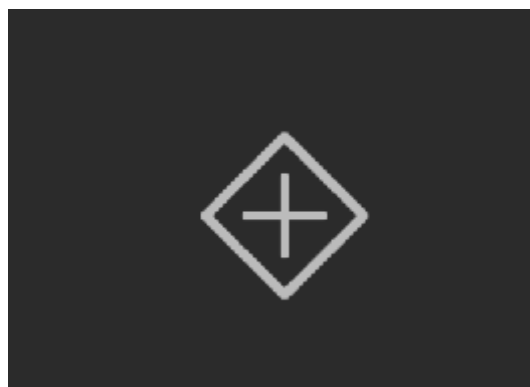
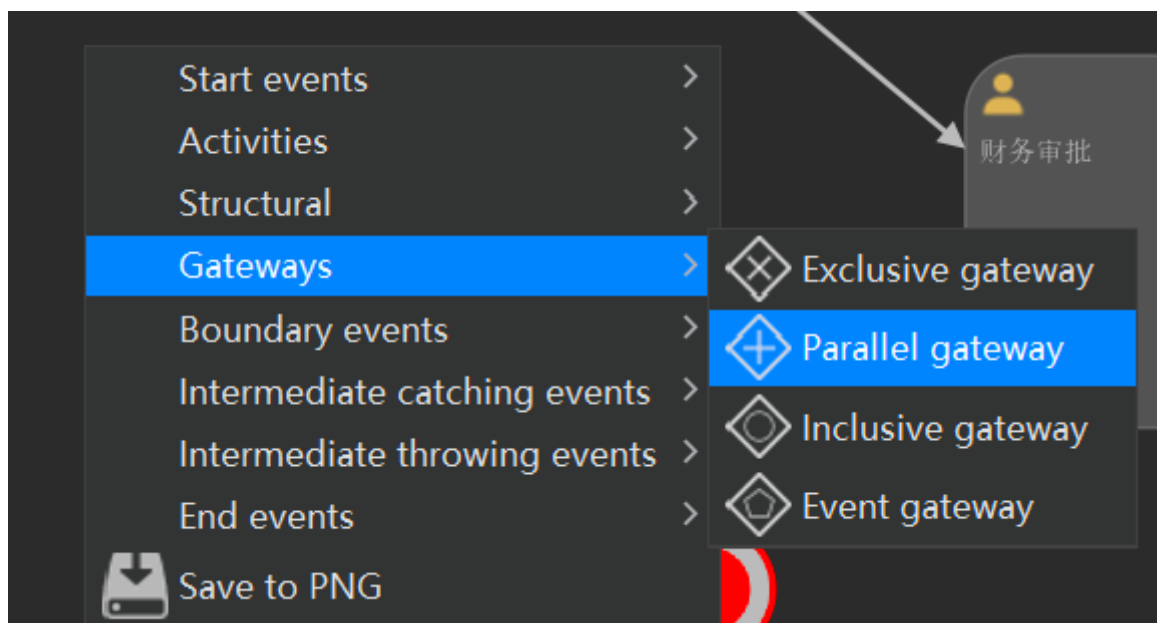
如果同一个并行网关有多个进入和多个外出顺序流，它就同时具有分支和汇聚功能。这时，网关会先汇聚所有进入的顺序流，然后再切分成多个并行分支

并行网关不会解析条件。即使顺序流中定义了条件，也会被忽略。

任务全部完成，在汇聚点汇聚，才通过parallelGateway并行网关

并行网关在业务应用中常用于会签任务，会签任务即多个参与者共同办理的任务

流程定义



包含网关

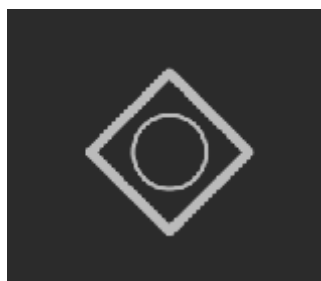
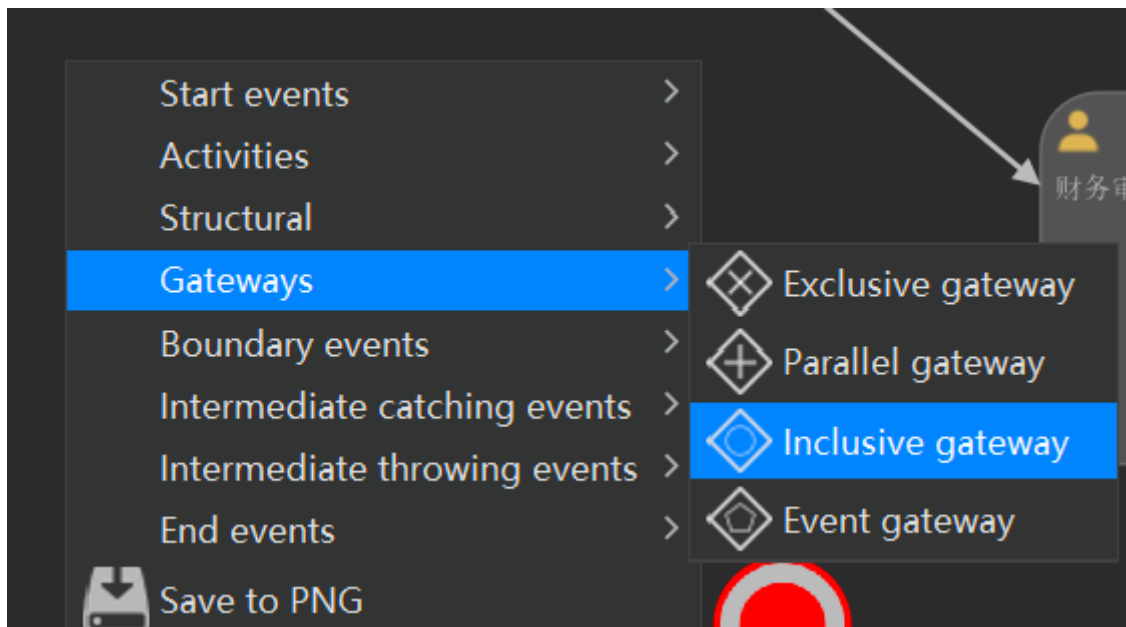
概述

InclusiveGateway(包含网关)，包含网关可以看做是排他网关和并行网关的结合体。和排他网关一样，你可以在外出顺序流上定义条件，包含网关会解析它们。但是主要的区别是包含网关可以选择多于一条顺序流，这和并行网关一样。

包含网关的功能是基于进入和外出顺序流的

- 分支：所有外出顺序流的条件都会被解析，结果为true的顺序流会以并行方式继续执行，会为每个顺序流创建一个分支
- 汇聚：所有并行分支到达包含网关，会进入等待状态，直到每个包含流程token的进入顺序流的分支都到达。这是与并行网关的最大不同。换句话说，包含网关只会等待被选中执行了的进入顺序流。在汇聚之后，流程会穿过包含网关继续执行

流程定义



事件网关

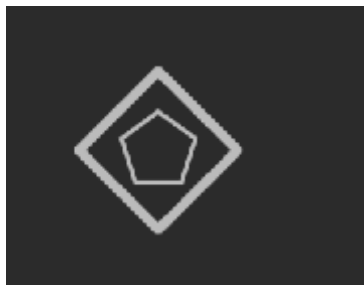
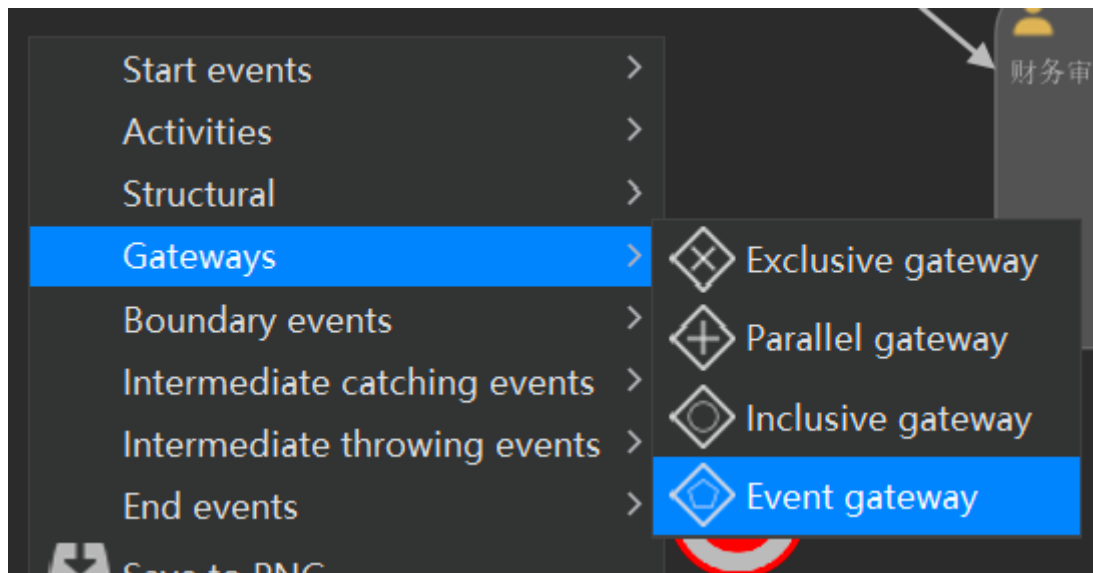
概述

EventGateway(事件网关)，事件网关允许根据事件判断流向。网关的每个外出顺序流都要连接到一个中间捕获事件。当流程到达一个基于事件网关，网关会进入等待状态：会暂停执行。与此同时，会为每个外出顺序流创建相对的事件订阅。

事件网关的外出顺序流和普通顺序流不同，这些顺序流不会真的"执行"，相反它们让流程引擎去决定执行到事件网关的流程需要订阅哪些事件

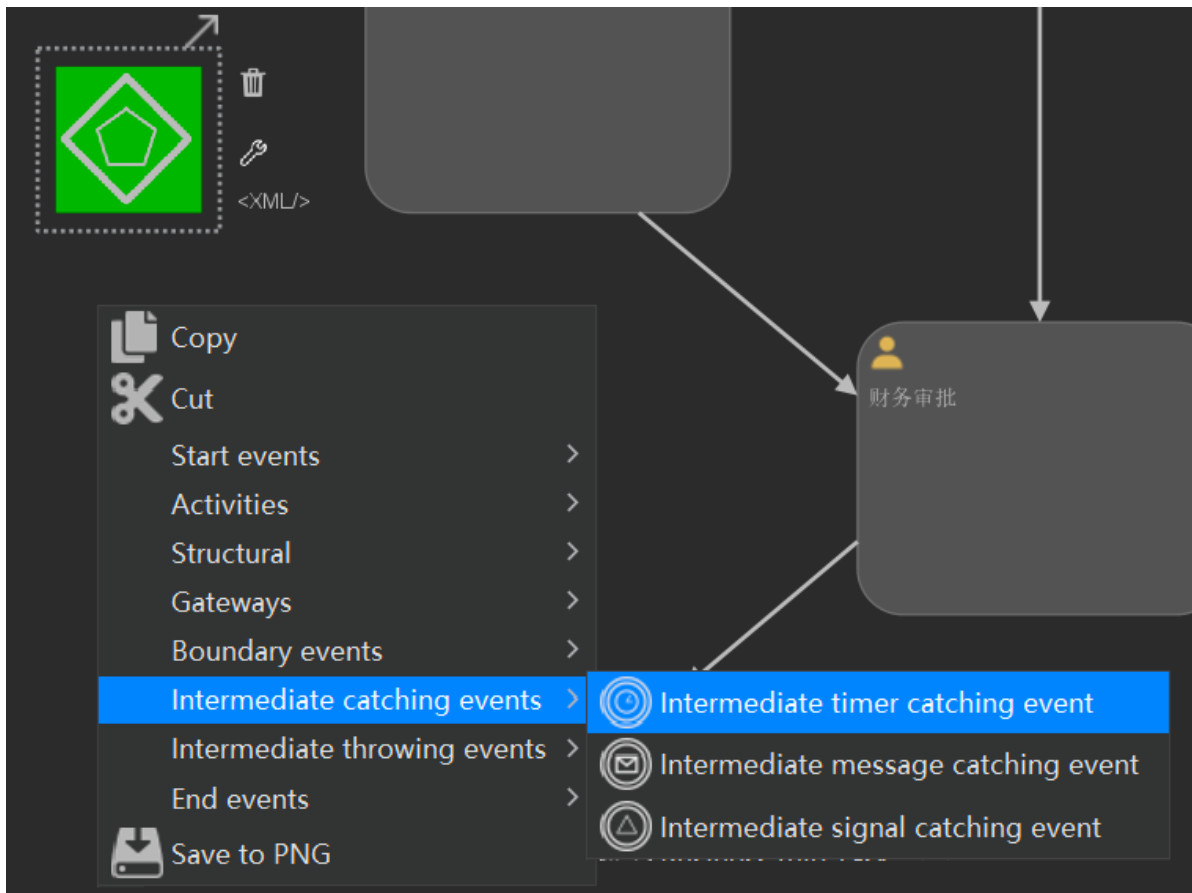
- 事件网关必须有两条或以上外出顺序流
- 事件网关后，只能使用intermediateCatchEvent类型（activiti不支持基于事件网关后连接ReceiveTask）
- 连接到事件网关的中间捕获事件必须只有一个入口顺序流

流程定义



intermediateCatchEvent支持的事件类型：

- Message Event：消息事件
- Singal Event：信号事件
- Timer Event：定时事件



Activiti整合SpringBoot

pom依赖

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5       http://maven.apache.org/xsd/maven-4.0.0.xsd">
6   <modelVersion>4.0.0</modelVersion>
7   <parent>
8     <groupId>org.springframework.boot</groupId>
9     <artifactId>spring-boot-starter-parent</artifactId>
10    <version>2.7.2</version>
11    <relativePath/> <!-- lookup parent from repository -->
```

```

10     </parent>
11     <groupId>mao</groupId>
12     <artifactId>Activiti-SpringBoot-demo</artifactId>
13     <version>0.0.1-SNAPSHOT</version>
14     <name>Activiti-SpringBoot-demo</name>
15     <description>Activiti-SpringBoot-demo</description>
16     <properties>
17         <java.version>8</java.version>
18     </properties>
19     <dependencies>
20         <dependency>
21             <groupId>org.springframework.boot</groupId>
22             <artifactId>spring-boot-starter-web</artifactId>
23         </dependency>
24
25         <dependency>
26             <groupId>org.springframework.boot</groupId>
27             <artifactId>spring-boot-starter-test</artifactId>
28             <scope>test</scope>
29         </dependency>
30         <dependency>
31             <groupId>org.springframework.boot</groupId>
32             <artifactId>spring-boot-starter-jdbc</artifactId>
33         </dependency>
34
35         <!--spring boot activiti 依赖-->
36         <dependency>
37             <groupId>org.activiti</groupId>
38             <artifactId>activiti-spring-boot-starter</artifactId>
39             <version>7.0.0.SR1</version>
40         </dependency>
41
42         <!--mysql 依赖 spring-boot-->
43         <dependency>
44             <groupId>mysql</groupId>
45             <artifactId>mysql-connector-java</artifactId>
46             <scope>runtime</scope>
47         </dependency>
48     </dependencies>
49
50     <build>
51         <plugins>
52             <plugin>
53                 <groupId>org.springframework.boot</groupId>
54                 <artifactId>spring-boot-maven-plugin</artifactId>
55             </plugin>
56         </plugins>
57     </build>
58
59 </project>
60

```

application.yml配置

```
1 spring:
2   datasource:
3     url: jdbc:mysql://activiti?
4     useUnicode=true&characterEncoding=utf8&serverTimezone=GMT
5     username: root
6     password: 123456
7     driver-class-name: com.mysql.cj.jdbc.Driver
8   activiti:
9     #1.flase: 默认值。activiti在启动时，对比数据库表中保存的版本，如果没有表或者版本不匹
10    配，将抛出异常
11    #2.true: activiti会对数据库中所有表进行更新操作。如果表不存在，则自动创建
12    #3.create_drop: 在activiti启动时创建表，在关闭时删除表（必须手动关闭引擎，才能删
13    除表）
14    #4.drop-create: 在activiti启动时删除原来的旧表，然后在创建新表（不需要手动关闭引
15    擎）
16    database-schema-update: true
17    #检测历史表是否存在 activiti7默认没有开启数据库历史记录 启动数据库历史记录
18    db-history-used: true
19    #记录历史等级 可配置的历史级别有none, activity, audit, full
20    #none: 不保存任何的历史数据，因此，在流程执行过程中，这是最高效的。
21    #activity: 级别高于none，保存流程实例与流程行为，其他数据不保存。
22    #audit: 除activity级别会保存的数据外，还会保存全部的流程任务及其属性。audit为
23    history的默认值。
24    #full: 保存历史数据的最高级别，除了会保存audit级别的数据外，还会保存其他全部流程相关
25    的细节数据，包括一些流程参数等。
26    history-level: full
27    #校验流程文件，默认校验resources下的processes文件夹里的流程文件
28    check-process-definitions: false
29
30 server:
31   port: 9090
```

安全框架整合配置

Activiti7与SpringBoot整合后，默认情况下，集成了SpringSecurity安全框架，这样我们就要去准备SpringSecurity整合进来的相关用户权限配置信息

SecurityUtil类

```
1 package mao.activiti_springboot_demo.utils;
2
3 import org.slf4j.Logger;
4 import org.slf4j.LoggerFactory;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.beans.factory.annotation.Qualifier;
```

```

7  import org.springframework.security.core.Authentication;
8  import org.springframework.security.core.GrantedAuthority;
9  import org.springframework.security.core.context.SecurityContextHolder;
10 import org.springframework.security.core.context.SecurityContextImpl;
11 import org.springframework.security.core.userdetails.UserDetails;
12 import org.springframework.security.core.userdetails.UserDetailsService;
13 import org.springframework.stereotype.Component;
14
15 import java.util.Collection;
16
17 /**
18  * Project name(项目名称): Activiti-SpringBoot-demo
19  * Package(包名): mao.activiti_springboot_demo.utils
20  * Class(类名): SecurityUtils
21  * Author(作者): mao
22  * Author QQ: 1296193245
23  * GitHub: https://github.com/maomao124/
24  * Date(创建日期): 2023/10/19
25  * Time(创建时间): 15:53
26  * Version(版本): 1.0
27  * Description(描述): 无
28  */
29
30 @Component
31 public class SecurityUtils
32 {
33
34     private static final Logger log =
35     LoggerFactory.getLogger(SecurityUtils.class);
36
37     @Autowired
38     @Qualifier("myUserDetailsService")
39     private UserDetailsService userDetailsService;
40
41     public void loginAs(String username)
42     {
43         UserDetails user = userDetailsService.loadUserByUsername(username);
44         if (user == null)
45         {
46             throw new IllegalStateException("User " + username + " doesn't
47             exist, please provide a valid user");
48         }
49         log.info("> Logged in as: " + username);
50
51         SecurityContextHolder.setContext(
52             new SecurityContextImpl(
53                 new Authentication()
54                 {
55                     @Override
56                     public Collection<? extends GrantedAuthority>
57                     getAuthorities()
58                     {
59                         return user.getAuthorities();
60                     }
61                 }
62             )
63         );
64     }
65 }

```

```

59         @Override
60         public Object getCredentials()
61         {
62             return user.getPassword();
63         }
64
65         @Override
66         public Object getDetails()
67         {
68             return user;
69         }
70
71         @Override
72         public Object getPrincipal()
73         {
74             return user;
75         }
76
77         @Override
78         public boolean isAuthenticated()
79         {
80             return true;
81         }
82
83         @Override
84         public void setAuthenticated(boolean
isAuthenticated) throws IllegalArgumentException
85         {
86         }
87
88         @Override
89         public String getName()
90         {
91             return user.getUsername();
92         }
93     });
94
95     org.activiti.engine.impl.identity.Authentication.setAuthenticatedUserId(use
rname);
96 }

```

DemoApplicationConfig类

它的作用是为了实现SpringSecurity框架的用户权限的配置，这样我们就可以在系统中使用用户权限信息。

后面处理流程时用到的任务负责人，需要添加在这里

```

1 package mao.activiti_springboot_demo.config;
2

```

```

3  import org.slf4j.Logger;
4  import org.slf4j.LoggerFactory;
5  import org.springframework.context.annotation.Bean;
6  import org.springframework.context.annotation.Configuration;
7  import org.springframework.security.core.authority.SimpleGrantedAuthority;
8  import org.springframework.security.core.userdetails.User;
9  import org.springframework.security.core.userdetails.UserDetailsService;
10 import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
11 import org.springframework.security.crypto.password.PasswordEncoder;
12 import org.springframework.security.provisioning.InMemoryUserDetailsManager;
13
14 import java.util.Arrays;
15 import java.util.List;
16 import java.util.stream.Collectors;
17
18 /**
19  * Project name(项目名称): Activiti-SpringBoot-demo
20  * Package(包名): mao.activiti_springboot_demo.config
21  * Class(类名): DemoApplicationConfiguration
22  * Author(作者): mao
23  * Author QQ: 1296193245
24  * GitHub: https://github.com/maomao124/
25  * Date(创建日期): 2023/10/19
26  * Time(创建时间): 15:56
27  * Version(版本): 1.0
28  * Description(描述): 无
29  */
30
31 @Configuration
32 public class DemoApplicationConfiguration
33 {
34     private final Logger log =
35         LoggerFactory.getLogger(DemoApplicationConfiguration.class);
36
37     @Bean
38     public UserDetailsService myUserDetailsService()
39     {
40         InMemoryUserDetailsManager inMemoryUserDetailsManager = new
41             InMemoryUserDetailsManager();
42         //这里添加用户，后面处理流程时用到的任务负责人，需要添加在这里
43         String[][] usersGroupsAndRoles = {
44             {"jack", "password", "ROLE_ACTIVITI_USER",
45             "GROUP_activitiTeam"},
46             {"rose", "password", "ROLE_ACTIVITI_USER",
47             "GROUP_activitiTeam"},
48             {"tom", "password", "ROLE_ACTIVITI_USER",
49             "GROUP_activitiTeam"},
50             {"other", "password", "ROLE_ACTIVITI_USER",
51             "GROUP_otherTeam"},
52             {"system", "password", "ROLE_ACTIVITI_USER"},
53             {"admin", "password", "ROLE_ACTIVITI_ADMIN"}
54         };
55
56         for (String[] user : usersGroupsAndRoles)
57         {

```



```

52         List<String> authoritiesStrings =
Arrays.asList(Arrays.copyOfRange(user, 2, user.length));
53         log.info("> Registering new user: " + user[0] + " with the
following Authorities[" + authoritiesStrings + "]");
54         inMemoryUserDetailsManager.createUser(new User(user[0],
passwordEncoder().encode(user[1]),
55
        authoritiesStrings.stream().map(SimpleGrantedAuthority::new).collect(Collectors.toList())));
56     }
57
58     return inMemoryUserDetailsManager;
59 }
60
61 @Bean
62 public PasswordEncoder passwordEncoder()
63 {
64     return new BCryptPasswordEncoder();
65 }
66 }

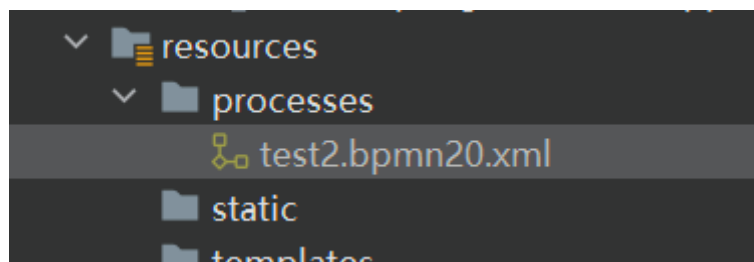
```

创建Bpmn文件

Activiti7可以自动部署流程，前提是在resources目录下，创建一个新的目录processes，用来放置bpmn文件

Candidate Groups中的内容与上面DemoApplicationConfiguration类中出现的用户组名称保持一致，可以填写：activitiTeam 或者 otherTeam

这样填写的好处：当不确定到底由谁来负责当前任务的时候，只要是Groups内的用户都可以拾取这个任务



测试

```

1  package mao.activiti_springboot_demo;
2
3  import mao.activiti_springboot_demo.utils.SecurityUtils;
4  import org.activiti.api.process.model.ProcessDefinition;
5  import org.activiti.api.process.model.ProcessInstance;
6  import org.activiti.api.process.model.builders.ProcessPayloadBuilder;
7  import org.activiti.api.process.runtime.ProcessRuntime;
8  import org.activiti.api.runtime.shared.query.Page;
9  import org.activiti.api.runtime.shared.query.Pageable;
10 import org.activiti.api.task.runtime.TaskRuntime;
11 import org.apache.catalina.security.SecurityUtil;
12 import org.junit.jupiter.api.Test;
13 import org.springframework.beans.factory.annotation.Autowired;
14 import org.springframework.boot.test.context.SpringBootTest;
15
16 @SpringBootTest
17 class ActivitiSpringBootTestApplicationTests
18 {
19
20     @Autowired
21     private ProcessRuntime processRuntime;
22
23     @Autowired
24     private TaskRuntime taskRuntime;
25
26     @Autowired
27     private SecurityUtils securityUtils;
28
29     @Test
30     void contextLoads()
31     {
32         Page<ProcessDefinition> processDefinitionPage =
33             processRuntime.processDefinitions(Pageable.of(0, 10));
34         System.out.println("可用的流程定义数量: " +
processDefinitionPage.getTotalItems());
35         for (org.activiti.api.process.model.ProcessDefinition pd :
processDefinitionPage.getContent())
36         {
37             System.out.println("流程定义: " + pd);
38         }
39     }
40
41 }

```
