Activti工作流实战开发

课前默写

1. 请写出Activiti的服务
2. 请写出Activiti任务的启动方式
3. 请写出activiti用户和用户组的设置

课程回顾

1. Activiti的服务
2. Activiti的任务
3. Activiti的用户和用户组
4. Activiti的表单

今日内容

```
1.ServiceTask自动执行任务
2.MailTask邮件任务
3.ManualTask人工任务
4.ReceiveTask接收任务
14.指定任务组办理人
15.Activiti网关设置
```

教学目标

```
1.掌握ServiceTask自动执行任务
2.掌握MailTask邮件任务
3.掌握Activiti网关设置
4.整合Spring
```

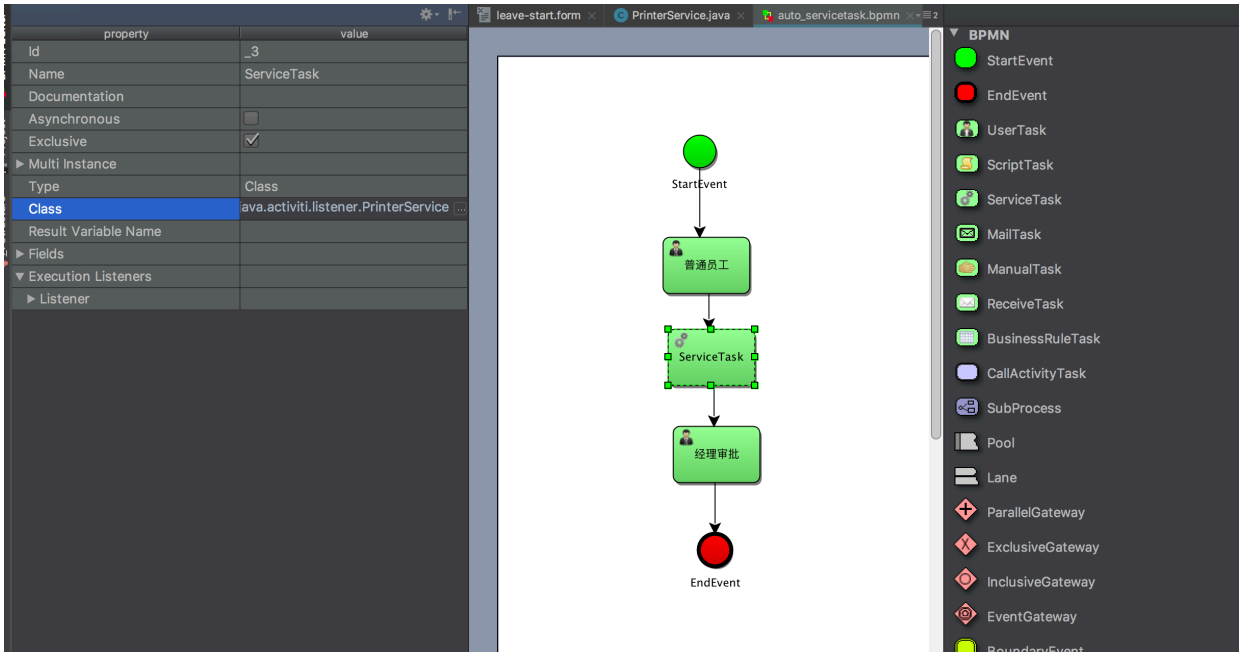
第十二章 ServiceTask自动执行任务

在Activiti执行待办任务中，有一些任务是自动执行的操作，在这个节点上不需要任何的人工干预，从而实现自动化操作。

当前一个办理人办理完节点之后，然后下一个节点就自动马上执行。ServiceTask能实现这个功能。

12.1、使用监听类实现java的服务任务

BPMN图



定义自动执行的类

```java
package com.sudojava.activiti.listener;

import org.activiti.engine.delegate.DelegateExecution;
import org.activiti.engine.delegate.JavaDelegate;

import java.util.Map;

public class PrinterService implements JavaDelegate {
    @Override
    public void execute(DelegateExecution execution) throws Exception {
        Map<String, Object> var = execution.getVariables();
        System.out.println("--hello-:>>" + var.get("name"));
    }
}
```

执行流程

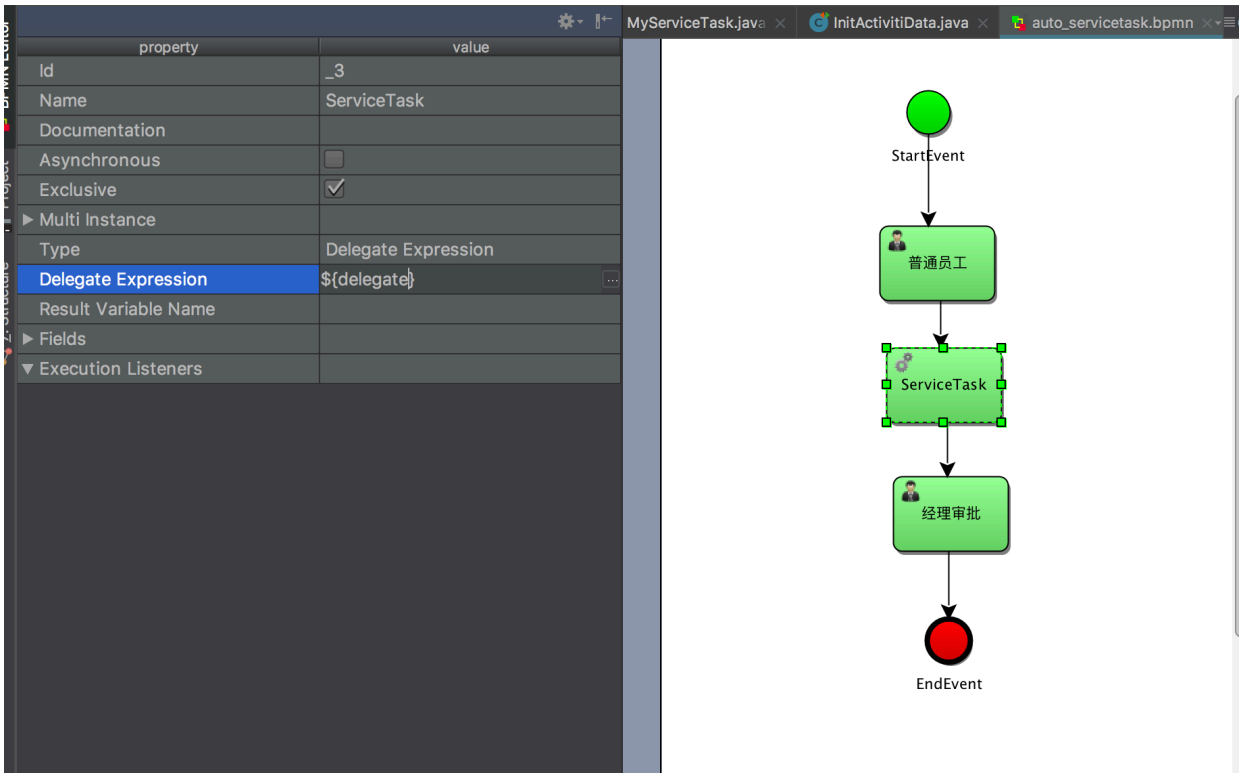
```java
package com.sudojava.activiti.servicetask;


import com.sudojava.activiti.initdata.ProjectInitManager;
import com.sudojava.activiti.task.ExecutorMyTask;
import org.activiti.engine.RepositoryService;
import org.activiti.engine.RuntimeService;
import org.activiti.engine.TaskService;
import org.activiti.engine.repository.Deployment;
import org.activiti.engine.runtime.ProcessInstance;
import org.activiti.engine.task.Task;

import java.io.InputStream;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.function.Consumer;

public class MyServiceTask {

    private ProjectInitManager manager;

    public MyServiceTask() {
        manager = new ProjectInitManager();
    }

    /**
     * 发布流程
     *
```

```
     * @param fileName
     */
    public void deloyProcess(String fileName) {
        RepositoryService repositoryService =
manager.getRepositoryService();
        InputStream inputStream =
this.getClass().getClassLoader().getResourceAsStream(fileName);
        Deployment deployment =
repositoryService.createDeployment().addInputStream(fileName,
inputStream).deploy();
        System.out.println("------>>" + deployment.getId());
        System.out.println("------>>" + deployment.getName());
    }

    public void startProcessInstance(String processName) {
        RuntimeService runtimeService = manager.getRuntimeService();
        Map<String, Object> var = new HashMap<>();
        var.put("name", "jack");
        ProcessInstance processInstance =
runtimeService.startProcessInstanceByKey(processName, var);
        System.out.println("---->>" + processInstance.getId());
        System.out.println("---->>" +
processInstance.getProcessDefinitionName());
        System.out.println("---->>" +
processInstance.getProcessDefinitionId());
        System.out.println("---->>" + processInstance.getBusinessKey());
    }

    public void findPersonalTask() {
        TaskService taskService = manager.getTaskService();
        List<Task> list =
taskService.createTaskQuery().taskAssignee("zhangsan")
                .processDefinitionKey("myProcess_1").list();
        list.forEach(new Consumer<Task>() {
            @Override
            public void accept(Task task) {
                System.out.println("------>>"+task.getId());
                System.out.println("------>>"+task.getAssignee());
            }
        });
    }
    public void completePersonalTask(String taskID){
        manager.getTaskService().complete(taskID);
    }
    public static void main(String[] args) {


         // new MyServiceTask().deloyProcess("auto_servicetask.bpmn");
         // new MyServiceTask().startProcessInstance("myProcess_1");
```

```
        // new MyServiceTask().startProcessInstance("myProcess_1");
        // new MyServiceTask().findPersonalTask();
         new MyServiceTask().completePersonalTask("2505");
    }
}
```

12.2、使用Delegate expression来实现java的服务任务

BPMN流程图



${delegate}表示存放流程中的变量，该变量是一个java对象，来实现对应的服务。

辅助自动执行类

```java
package com.sudojava.activiti.listener;

import org.activiti.engine.delegate.DelegateExecution;
import org.activiti.engine.delegate.JavaDelegate;

import java.io.Serializable;
import java.util.Map;

public class PrinterService implements Serializable, JavaDelegate {
    @Override
    public void execute(DelegateExecution execution) throws Exception {
        Map<String, Object> var = execution.getVariables();
        System.out.println("--hello-:>>" + var.get("name"));
    }
}
```

执行流程

```java
package com.sudojava.activiti.servicetask;


import com.sudojava.activiti.initdata.ProjectInitManager;
import com.sudojava.activiti.listener.PrinterService;
import com.sudojava.activiti.task.ExecutorMyTask;
import org.activiti.engine.RepositoryService;
import org.activiti.engine.RuntimeService;
import org.activiti.engine.TaskService;
import org.activiti.engine.repository.Deployment;
import org.activiti.engine.runtime.ProcessInstance;
import org.activiti.engine.task.Task;

import java.io.InputStream;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.function.Consumer;

public class MyServiceTask {

    private ProjectInitManager manager;

    public MyServiceTask() {
        manager = new ProjectInitManager();
    }

    /**
```

```
     * 发布流程
     *
     * @param fileName
     */
    public void deloyProcess(String fileName) {
        RepositoryService repositoryService =
manager.getRepositoryService();
        InputStream inputStream =
this.getClass().getClassLoader().getResourceAsStream(fileName);
        Deployment deployment =
repositoryService.createDeployment().addInputStream(fileName,
inputStream).deploy();
        System.out.println("------>>" + deployment.getId());
        System.out.println("------>>" + deployment.getName());
    }

    public void startProcessInstance(String processName) {
        RuntimeService runtimeService = manager.getRuntimeService();
        Map<String, Object> var = new HashMap<>();
        var.put("name", "jack");
        var.put("delegate",new PrinterService());
        ProcessInstance processInstance =
runtimeService.startProcessInstanceByKey(processName, var);
        System.out.println("---->>" + processInstance.getId());
        System.out.println("---->>" +
processInstance.getProcessDefinitionName());
        System.out.println("---->>" +
processInstance.getProcessDefinitionId());
        System.out.println("---->>" + processInstance.getBusinessKey());
    }

    public void findPersonalTask() {
        TaskService taskService = manager.getTaskService();
        List<Task> list =
taskService.createTaskQuery().taskAssignee("zhangsan")
                .processDefinitionKey("myProcess_1").list();
        list.forEach(new Consumer<Task>() {
            @Override
            public void accept(Task task) {
                System.out.println("------>>"+task.getId());
                System.out.println("------>>"+task.getAssignee());
            }
        });
    }
    public void completePersonalTask(String taskID){
        manager.getTaskService().complete(taskID);
    }
    public static void main(String[] args) {
```

```
          // new MyServiceTask().deloyProcess("auto_servicetask.bpmn");
          // new MyServiceTask().startProcessInstance("myProcess_1");
        // new MyServiceTask().startProcessInstance("myProcess_1");
        // new MyServiceTask().findPersonalTask();
         new MyServiceTask().completePersonalTask("5008");
    }
}
```

第十三章 MailTask邮件任务

Activiti允许通过一个或者多个收件人发送电子邮件的自动邮件服务任务来增强业务流程，包括CC、BCC、简单的HTML内容等，值得注意的是邮件任务不是BPMN2.0官方任务规范，因此，在Activiti邮件任务作为一个专门的服务任务来实现的。并且该节点是自动执行的。

邮件服务器的配置一

activiti引擎要通过支持SMTP功能的外部邮件服务器发送邮件(gevhfepsbjgwbjfc)。 为了实际发送邮件，引擎突知道如何访问邮件服务器。 下面的配置可以设置到activiti.cfg.xml配置文件中：

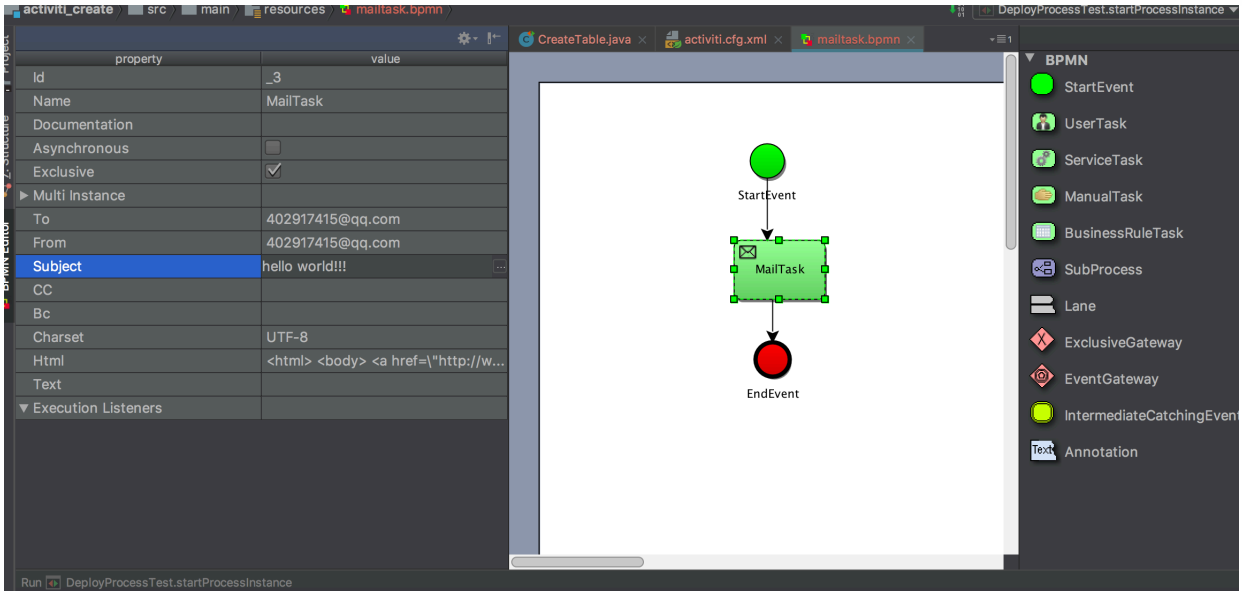
Property	Required?	Description
mailServerHost	no	The hostname of your mail server (e.g. mail.mycorp.com). Default is `localhost`
mailServerPort	yes, if not on the default port	The port for SMTP traffic on the mail server. The default is *25*
mailServerDefaultFrom	no	The default e-mail address of the sender of e-mails, when none is provided by the user. By default this is *activiti@activiti.org*
mailServerUsername	if applicable for your server	Some mail servers require credentials for sending e-mail. By default not set.
mailServerPassword	if applicable for your server	Some mail servers require credentials for sending e-mail. By default not set.
mailServerUseSSL	if applicable for your server	Some mail servers require ssl communication. By default set to false.
mailServerUseTLS	if applicable for your server	Some mail servers (for instance gmail) require TLS communication. By default set to false.

邮件服务器配置二

Property	Required?	Description
to	yes	The recipients if the e-mail. Multiple recipients are defined in a comma-separated list
from	no	The sender e-mail address. If not provided, the default configured from address is used.
subject	no	The subject of the e-mail.
cc	no	The cc's of the e-mail. Multiple recipients are defined in a comma-separated list

bcc	no	The bcc's of the e-mail. Multiple recipients are defined in a comma-separated list
charset	no	Allows to change the charset of the email, which is necessary for many non-English languages.
html	no	A piece of HTML that is the content of the e-mail.
text	no	The content of the e-mail, in case one needs to send plain none-rich e-mails. Can be used in combination with *html*, for e-mail clients that don't support rich content. The client will then fall back to this text-only alternative.
htmlVar	no	The name of a process variable that holds the HTML that is the content of the e-mail. The key difference between this and html is that this content will have expressions replaced before being sent by the mail task.
textVar	no	The name of a process variable that holds the plain text content of the e-mail. The key difference between this and html is that this content will have expressions replaced before being sent by the mail task.
ignoreException	no	Whether an failure when handling the e-mail throws an ActivitiException. By default this is set to false.
exceptionVariableName	no	When email handling does not throw an exception since *ignoreException* = *true* a variable with the given name is used to hold a failure message

配置BPMN流程图



在activiti.cfg.xml文件中需要配置指定的邮件服务器

```
<bean id="processEngineConfiguration"
        class="org.activiti.spring.SpringProcessEngineConfiguration">
      <property name="dataSource" ref="dataSource"/>
      <property name="transactionManager" ref="transactionManager"/>
      <property name="databaseSchemaUpdate" value="true"/>
      <!--配置邮件服务器-->
      <property name="mailServerHost" value="smtp.qq.com"/>
      <property name="mailServerPort" value="465"/>
      <property name="mailServerDefaultFrom" value="402917415@qq.com"/>
      <property name="mailServerUsername" value="402917415@qq.com"/>
      <property name="mailServerPassword" value="@@@@@@@@@@"/>
      <property name="mailServerUseSSL" value="true"/>
   </bean>
```

备注：其中mailServerPassword密码是通过QQ邮箱设置POP3服务，QQ邮箱默认是不开启的，通过手机发送短信到指定的QQ收信地址如图所示：

核心代码展示

```
private ProcessEngine engine = ProcessEngines.getDefaultProcessEngine();


    @Test
    public void deployProcess() {
        RepositoryService repositoryService =
                engine.getRepositoryService();
        InputStream inputStream =
                this.getClass().getClassLoader()
                        .getResourceAsStream("mailtask.bpmn");
        DeploymentBuilder deployment = repositoryService.createDeployment()
                .addInputStream("mailtask.bpmn", inputStream);
        Deployment deployment1 = deployment.deploy();
        System.out.println(deployment1.getDeploymentTime());
        System.out.println(deployment1.getId());

    }
```

启动流程并自动发送邮件

```
  @Test
    public void startProcessInstance() {

        RuntimeService runtimeService = engine.getRuntimeService();
        ProcessInstance processInstance =
runtimeService.startProcessInstanceByKey("myProcess_1");
        System.out.println("----->>"+processInstance.getId());
        System.out.println("----->>"+processInstance.getName());

    }
```

第十四章 指定任务组办理人

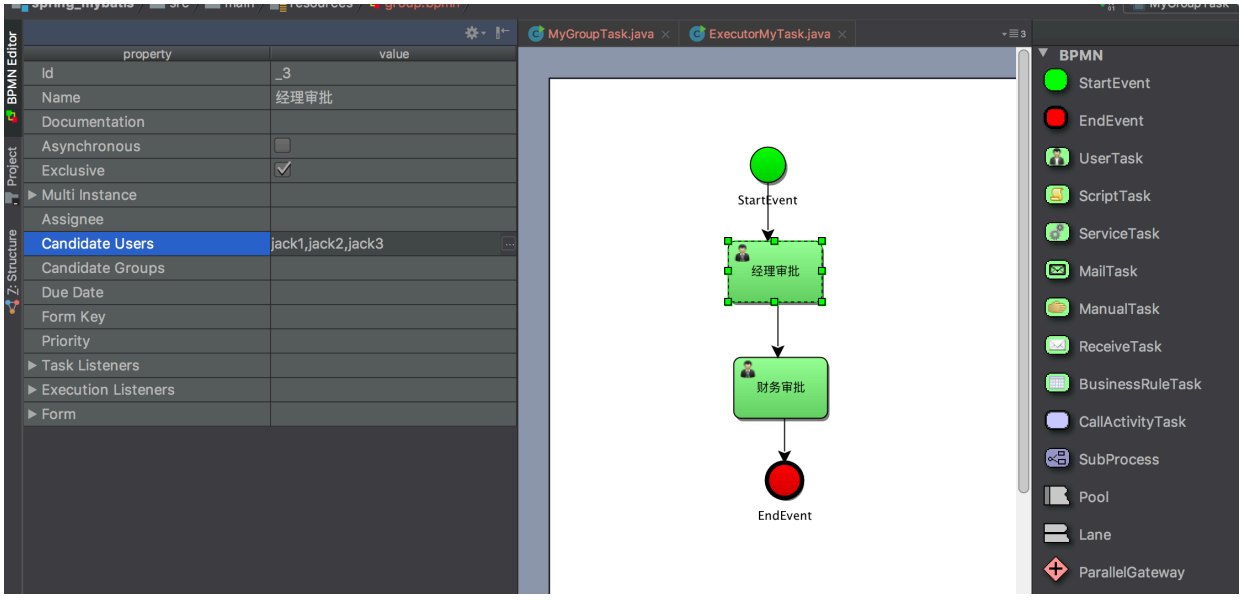
candidate-user是给某一个组的成员分配任务，candidate-user参数要求值的格式是多个字符串，中间以逗号分隔开。

当某一个流程传到该task对象时候，Activiti同样会创建一条task记录，但是与assignee不同的是，它并不属于某一个用户的task，而是属于在**candidate-user**属性中定义的所有参与者共同的任务，因此，在代码中需要执行taskService.createTaskQuery()查找到组的办理任务，只需要指定组中的任意一个成员办理了该任务，Activiti都会直接删除该记录，其他组员再也查不到该task记录了。

具体步骤如下：

1. 给任务设置候选人(多个候选人，中间使用逗号分隔开)，这里需要注意的是候选人是无法办理任务的。
2. 候选人都可以查询到任务组。
3. 候选人拾取(claim)组任务。
4. 候选人拾取任务之后，该候选人变成任务的执行人。
5. 执行完毕之后，Activiti直接删除该记录，其他组员查询不到该记录了。

BMPN流程图



核心代码展示

```java
package com.sudojava.activiti.grouptask;

import com.sudojava.activiti.initdata.ProjectInitManager;
import com.sudojava.activiti.task.ExecutorMyTask;
import org.activiti.engine.RepositoryService;
import org.activiti.engine.RuntimeService;
import org.activiti.engine.repository.Deployment;
import org.activiti.engine.runtime.ProcessInstance;
import org.activiti.engine.task.Task;
import org.activiti.engine.task.TaskQuery;

import java.io.InputStream;
import java.util.List;
import java.util.function.Consumer;

public class MyGroupTask {
    private ProjectInitManager manager;

    public MyGroupTask(){
        manager = new ProjectInitManager();
    }
    public void deployProcess(String fileName){
        RepositoryService repositoryService =
                manager.getRepositoryService();

        InputStream inputStream =
this.getClass().getClassLoader().getResourceAsStream(fileName);
        Deployment deployment = repositoryService.createDeployment()
                .addInputStream(fileName,inputStream).name("用户组授权")
                .category("分类").deploy();
        System.out.println("--->>" + deployment.getId());
        System.out.println("--->>" + deployment.getName());
    }

    /**
     *
     * @param processName
     */
    public void startProcessInstance(String processName) {
        RuntimeService service = manager.getRuntimeService();
        ProcessInstance instance =
service.startProcessInstanceByKey(processName);
        System.out.println("---->>" + instance.getId());
        System.out.println("---->>" + instance.getProcessDefinitionName());
        System.out.println("---->>" + instance.getProcessDefinitionId());
        System.out.println("---->>" + instance.getBusinessKey());
    }

    /**
```

```
     *
     */
    public void findPersonalTask() {
        String userID = "jack2";
      TaskQuery taskQuery =  manager.getTaskService().createTaskQuery();
      taskQuery.taskCandidateUser(userID);
      taskQuery.processDefinitionKey("myProcess_1");

        List<Task> list = taskQuery.list();
        list.forEach(new Consumer<Task>() {
            @Override
            public void accept(Task task) {
                System.out.println("----->>"+task.getAssignee());
                System.out.println("----->>"+task.getId());
            }
        });
      //claim与setAssignee区别在于claim领取之后别人不可以再领取不然会报错而
setAssignee则不然
        //认领流程
       // manager.getTaskService().claim("2504","jack");
        //撤销办理
      //  manager.getTaskService().resolveTask("2504");
    }

    /**
     *
     * @param taskID
     */
    public void completeMyPersonalTask(String taskID){
        manager.getTaskService().complete(taskID);
    }

    public static void main(String[] args) {

        MyGroupTask groupTask = new MyGroupTask();
  //      groupTask.deployProcess("group.bpmn");
     //   groupTask.startProcessInstance("myProcess_1");
        groupTask.findPersonalTask();

    }

}
```

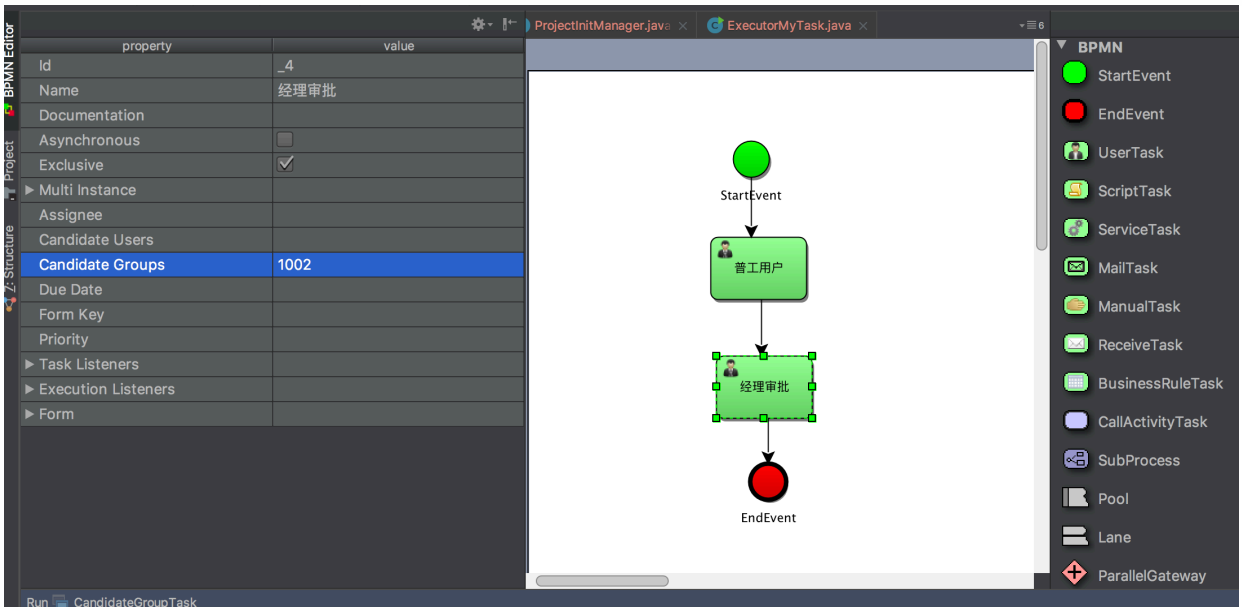
设置用户组办理任务

指定用户组，并添加该用户组下的对应用户

```
package com.sudojava.activiti.grouptask;

import com.sudojava.activiti.initdata.ProjectInitManager;
import org.activiti.engine.IdentityService;
import org.activiti.engine.identity.User;
import org.activiti.engine.impl.persistence.entity.GroupEntity;
import org.activiti.engine.impl.persistence.entity.UserEntity;

public class CandidateGroupTask {

    private ProjectInitManager manager;

    public CandidateGroupTask(){
        manager = new ProjectInitManager();
    }

    public void setUserGroup(){
        //设置用户组合用户信息
        IdentityService identityService = manager.getIdentityService();
        GroupEntity commonUser = new GroupEntity();
        commonUser.setId("1001");
        commonUser.setName("普通用户");
        identityService.saveGroup(commonUser);
        //设置该组对应的员工


        UserEntity zhangsan = new UserEntity();
        zhangsan.setId("zhangsan");
        zhangsan.setFirstName("zhangsan");
        identityService.saveUser(zhangsan);

        identityService.deleteMembership("zhangsan","1001");
        identityService.createMembership("zhangsan","1001");


        UserEntity lisi = new UserEntity();
        lisi.setId("lisi");
        lisi.setFirstName("lisi");
        identityService.saveUser(lisi);

        identityService.deleteMembership("lisi","1001");
```

```
        identityService.createMembership("lisi","1001");


        GroupEntity manager = new GroupEntity();
        manager.setId("1002");
        manager.setName("项目经理");
        identityService.saveGroup(manager);
        //设置该组对应的员工


        UserEntity jack = new UserEntity();
        jack.setId("jack");
        jack.setFirstName("jack");
        identityService.saveUser(jack);

        identityService.deleteMembership("jack","1002");
        identityService.createMembership("jack","1002");


        UserEntity rose = new UserEntity();
        rose.setId("rose");
        rose.setFirstName("rose");
        identityService.saveUser(rose);

        identityService.deleteMembership("rose","1002");
        identityService.createMembership("rose","1002");

    }

    public static void main(String[] args){
         CandidateGroupTask groupTask = new CandidateGroupTask();
         groupTask.setUserGroup();
    }

}
```

BPMN流程图



测试代码展示

分别使用不同用户组的人进行查询，结果如下：

```
package com.sudojava.activiti.grouptask;

import com.sudojava.activiti.initdata.ProjectInitManager;
import org.activiti.engine.IdentityService;
import org.activiti.engine.RepositoryService;
import org.activiti.engine.RuntimeService;
import org.activiti.engine.identity.User;
import org.activiti.engine.impl.persistence.entity.GroupEntity;
import org.activiti.engine.impl.persistence.entity.UserEntity;
import org.activiti.engine.repository.Deployment;
import org.activiti.engine.runtime.ProcessInstance;
import org.activiti.engine.task.Task;
import org.activiti.engine.task.TaskQuery;

import java.io.InputStream;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.function.Consumer;

public class CandidateGroupTask {

    private ProjectInitManager manager;

    public CandidateGroupTask(){
        manager = new ProjectInitManager();
    }

    public void setUserGroup(){
        //设置用户组合用户信息
```

```
IdentityService identityService = manager.getIdentityService();
GroupEntity commonUser = new GroupEntity();
commonUser.setId("1001");
commonUser.setName("普通用户");
identityService.saveGroup(commonUser);
//设置该组对应的员工


UserEntity zhangsan = new UserEntity();
zhangsan.setId("zhangsan");
zhangsan.setFirstName("zhangsan");
identityService.saveUser(zhangsan);

identityService.deleteMembership("zhangsan","1001");
identityService.createMembership("zhangsan","1001");


UserEntity lisi = new UserEntity();
lisi.setId("lisi");
lisi.setFirstName("lisi");
identityService.saveUser(lisi);

identityService.deleteMembership("lisi","1001");
identityService.createMembership("lisi","1001");


GroupEntity manager = new GroupEntity();
manager.setId("1002");
manager.setName("项目经理");
identityService.saveGroup(manager);
//设置该组对应的员工


UserEntity jack = new UserEntity();
jack.setId("jack");
jack.setFirstName("jack");
identityService.saveUser(jack);

identityService.deleteMembership("jack","1002");
identityService.createMembership("jack","1002");


UserEntity rose = new UserEntity();
rose.setId("rose");
rose.setFirstName("rose");
identityService.saveUser(rose);

identityService.deleteMembership("rose","1002");
identityService.createMembership("rose","1002");
```

```
    }

    /**
     * 发布流程
     * @param fileName
     */
    public void deployProcess(String fileName){

        RepositoryService repositoryService =
                manager.getRepositoryService();

        InputStream inputStream =
this.getClass().getClassLoader().getResourceAsStream(fileName);
        Deployment deployment = repositoryService.createDeployment()
                .addInputStream(fileName,inputStream).name("用户组授权")
                .category("按照用户组授权").deploy();
        System.out.println("--->>" + deployment.getId());
        System.out.println("--->>" + deployment.getName());
    }

    /**
     * 启动流程
     * @param processName
     */
    public void startProcessInstance(String processName) {
        RuntimeService runtimeService = manager.getRuntimeService();
        ProcessInstance processInstance =
runtimeService.startProcessInstanceByKey(processName);
        System.out.println("---->>" + processInstance.getId());
        System.out.println("---->>" +
processInstance.getProcessDefinitionName());
        System.out.println("---->>" +
processInstance.getProcessDefinitionId());
        System.out.println("---->>" + processInstance.getBusinessKey());
    }

    /**
     *
     */
    public void findPersonalTask() {
        String userID = "jack";
        TaskQuery taskQuery =  manager.getTaskService().createTaskQuery();
        taskQuery.taskCandidateUser(userID);
        taskQuery.processDefinitionKey("myProcess_1");

        List<Task> list = taskQuery.list();
        list.forEach(new Consumer<Task>() {
            @Override
```

```
            public void accept(Task task) {
                System.out.println("----->>"+task.getAssignee());
                System.out.println("----->>"+task.getId());
            }
        });
        //认领流程
         //manager.getTaskService().claim("15004","lisi");
        //撤销办理
        //  manager.getTaskService().resolveTask("2504");
    }

    /**
     *
     * @param taskID
     */
    public void completeMyPersonalTask(String taskID){
        manager.getTaskService().complete(taskID);
    }

    public static void main(String[] args){
         CandidateGroupTask groupTask = new CandidateGroupTask();
         //groupTask.setUserGroup();
        //groupTask.deployProcess("group.bpmn");
        //groupTask.startProcessInstance("myProcess_1");
        groupTask.findPersonalTask();
       // groupTask.completeMyPersonalTask("15004");
    }

}
```

第十五章 Activiti的网关(GateWay)设置

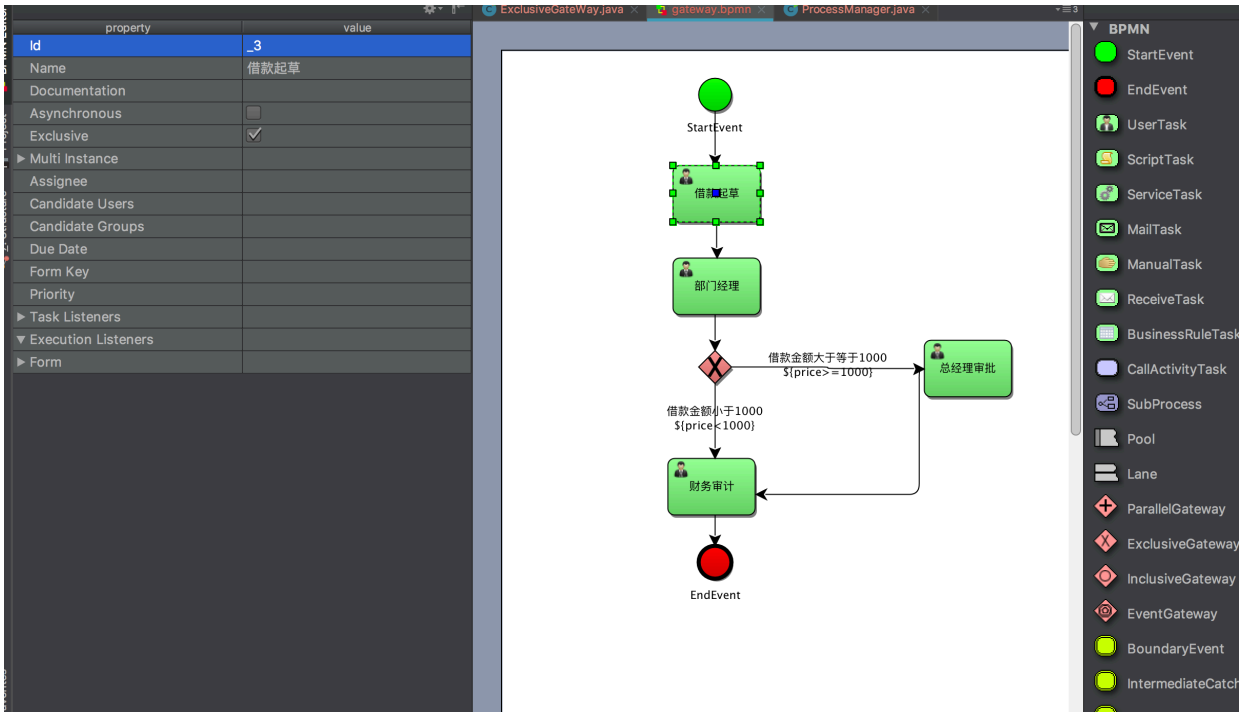
15.1、排他网关

排他网关，主要是在流程实例中实现决策，当流程执行到这个网关的时候，分支上需要设置condition条件，所有分支都会判断条件是否为true，如果条件为true，则执行该分支， **注意：排他网关只会选择一个为true的分支执行。**

如果分支的条件都不满足，没有经过排他网关，Activiti会抛出异常。

排他网关有点像Java中的if ... else if... 每一个分支上可以指定一个条件。

BPMN流程图



测试代码展示

```
package com.sudojava.activiti.gateway;

import com.sudojava.activiti.initdata.ProjectInitManager;
import org.activiti.engine.RepositoryService;
import org.activiti.engine.RuntimeService;
import org.activiti.engine.repository.Deployment;
import org.activiti.engine.runtime.ProcessInstance;
import org.activiti.engine.task.Task;
import org.junit.Test;

import java.io.InputStream;
import java.util.HashMap;
import java.util.Map;
import java.util.function.Consumer;

public class ExclusiveGateWay {
```

```
    private ProjectInitManager manager;

    public ExclusiveGateWay() {
        manager = new ProjectInitManager();
    }

    /**
     * 发布流程
     */
    @Test
    public void deployProcess() {

        RepositoryService repositoryService =
                manager.getRepositoryService();

        InputStream inputStream =
this.getClass().getClassLoader().getResourceAsStream("gateway.bpmn");
        Deployment deployment = repositoryService.createDeployment()
                .addInputStream("gateway.bpmn", inputStream).name("排他网关")
                .category("排他网关").deploy();
        System.out.println("--->>" + deployment.getId());
        System.out.println("--->>" + deployment.getName());

    }

    @Test
    public void startProcessInstance() {
        RuntimeService runtimeService = manager.getRuntimeService();
      //设置流程变量，指定借款的金额为1001
        Map<String, Object> variable = new HashMap<>();
        variable.put("price", "1001");
        ProcessInstance processInstance =
                runtimeService.startProcessInstanceByKey("myProcess_1",
variable);
        System.out.println("---->>" + processInstance.getId());
        System.out.println("---->>" +
processInstance.getProcessDefinitionName());
        System.out.println("---->>" +
processInstance.getProcessDefinitionId());
        System.out.println("---->>" + processInstance.getBusinessKey());
    }

    /**
     * 查看我的待办事宜
     */
    @Test
    public void findPersonalTask() {

        String userID = "wangwu";
```

```
 manager.getTaskService().createTaskQuery().taskAssignee(userID).list().for
Each(new Consumer<Task>() {
            public void accept(Task task) {
                System.out.println("id=" + task.getId());
                System.out.println("name=" + task.getName());
                System.out.println("assinee=" + task.getAssignee());
                System.out.println("createTime=" + task.getCreateTime());
                System.out.println("executionId=" + task.getExecutionId());
            }
        });
    }

    /**
     * 结束我的待办事宜
     */
    @Test
    public void completePersonalTask() {
        String taskID = "20002";
        manager.getTaskService().complete(taskID);
    }
}
```

15.2、并行网关

并行网关，包括分支和汇聚两个节点，他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的顺序流：

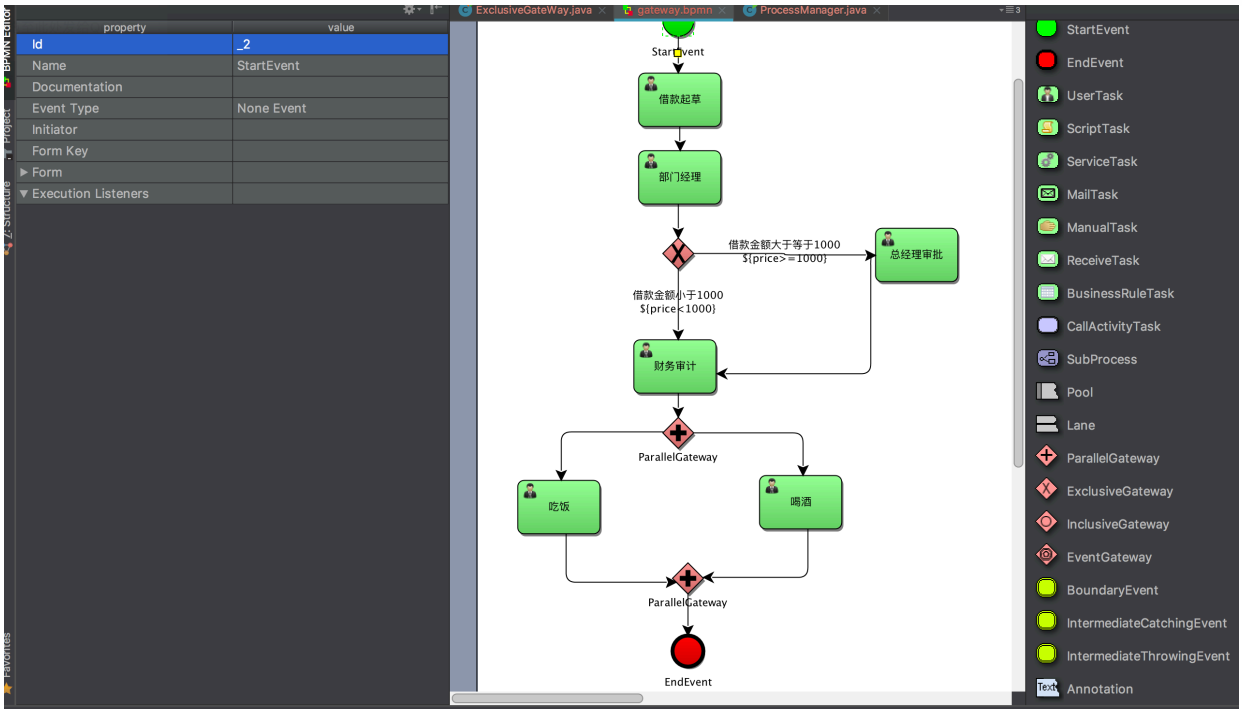
- 分支(Fock)：并行后所有外出顺序流，为每一个顺序流创建一个并发分支
- 汇聚(Join)：所有到达并行网关，在此等待进入分支，直到所有进入顺序流的分支到达以后，流程就会通过汇聚网关。

注意：如果同一个并行的网关有多个进入和多个外出顺序流，他就同时具有分支和汇聚功能，这个时候，网关会先汇聚所有进入的顺序流，然后在切分成多个并行的分支。

需要强调的是：

并行网关一定是成对出现的，有分支必然有汇聚，一个分支执行完毕后，需要等待其他分支全部执行完流程才会走到下一个节点。

BPMN流程图



代码展示

```
package com.sudojava.activiti.gateway;

import com.sudojava.activiti.initdata.ProjectInitManager;
import org.activiti.engine.RepositoryService;
import org.activiti.engine.RuntimeService;
import org.activiti.engine.repository.Deployment;
import org.activiti.engine.runtime.ProcessInstance;
import org.activiti.engine.task.Task;
import org.junit.Test;

import java.io.InputStream;
import java.util.HashMap;
import java.util.Map;
import java.util.function.Consumer;

public class FockAndJoinGate {

    private ProjectInitManager manager;

    public FockAndJoinGate(){
        manager = new ProjectInitManager();
    }

    @Test
    public void deployProcess(){
        RepositoryService repositoryService =
                manager.getRepositoryService();
        InputStream inputStream =
this.getClass().getClassLoader().getResourceAsStream("gateway.bpmn");
```

```
        Deployment deployment =
                repositoryService.createDeployment()
                        .addInputStream("gateway.bpmn",inputStream)
                        .name("并行网关")
                .category("并行网关").deploy();
        System.out.println("--->>" + deployment.getId());
        System.out.println("--->>" + deployment.getName());
    }

    @Test
    public void startProcessInstance() {
        RuntimeService runtimeService = manager.getRuntimeService();
        Map<String, Object> variable = new HashMap<>();
        variable.put("price", "1001");
        ProcessInstance processInstance =
                runtimeService.startProcessInstanceByKey("myProcess_1",
variable);
        System.out.println("---->>" + processInstance.getId());
        System.out.println("---->>" +
processInstance.getProcessDefinitionName());
        System.out.println("---->>" +
processInstance.getProcessDefinitionId());
        System.out.println("---->>" + processInstance.getBusinessKey());
    }

    /**
     * 查看我的待办事宜
     */
    @Test
    public void findPersonalTask() {

        String userID = "shuaige";

 manager.getTaskService().createTaskQuery().taskAssignee(userID).list().for
Each(new Consumer<Task>() {
            public void accept(Task task) {
                System.out.println("id=" + task.getId());
                System.out.println("name=" + task.getName());
                System.out.println("assinee=" + task.getAssignee());
                System.out.println("createTime=" + task.getCreateTime());
                System.out.println("executionId=" + task.getExecutionId());
            }
        });
    }


    /**
     * 结束我的待办事宜
     */
```

```
    @Test
    public void completePersonalTask() {
        String taskID = "37508";
        manager.getTaskService().complete(taskID);
    }


}
```

15.3、包含网关

包含网关是看做排他网关和并行网关的结合体，和排他网关一样，我们能够在外出顺序流上定义条件，包含网关会解析他们，和排他网关的区别是能够选择多于一条的顺序流，这个和并行网关一样。

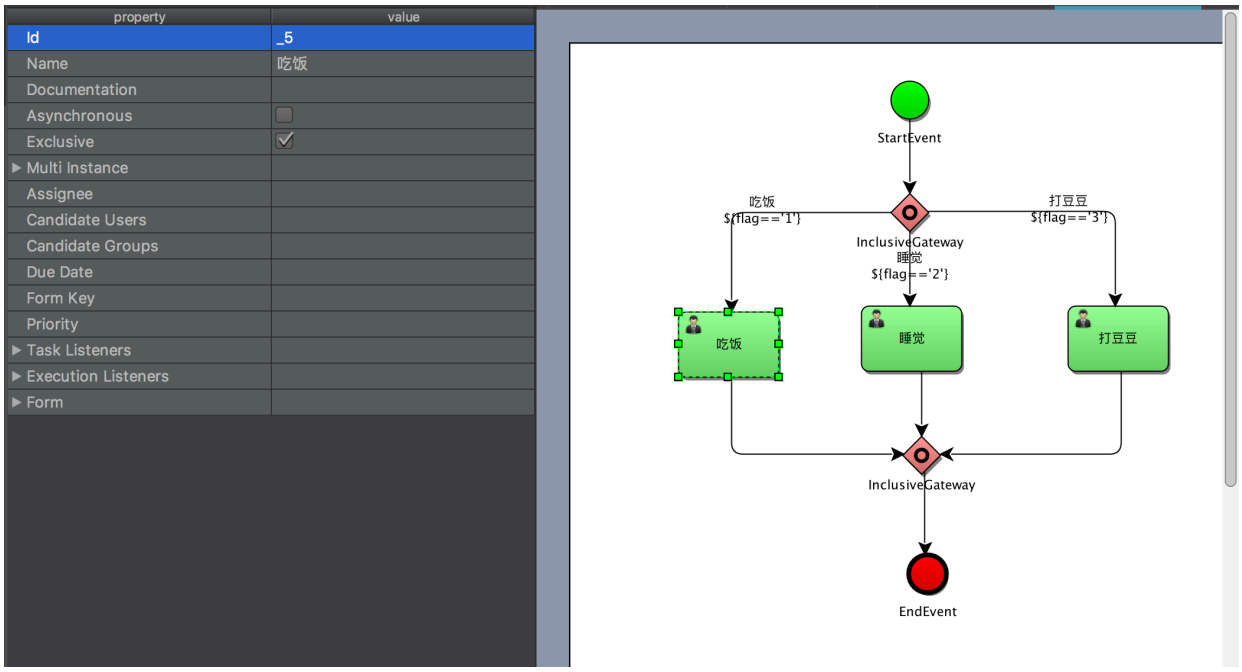
包含网关的功能是基于进入和外出顺序流的：

- **分支：** 全部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运行。 会为每一个顺序流创建一个分支。
- **汇聚：** 全部并行分支到达包括网关。会进入等待章台， 直到每一个包括流程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，包括网关仅仅会等待被选中运行了的进入顺序流。 在汇聚之后，流程会穿过包括网关继续运行。

注意，假设同一个包括节点拥有多个进入和外出顺序流。 它就会同一时候**含有分支和汇聚功能**。包含网关是判断条件的。

这时，网关会先汇聚全部拥有流程token的进入顺序流。 再依据条件推断结果为true的外出顺序流，为它们生成多条并行分支。

BPMN流程图



测试代码展示

```
package com.sudojava.activiti.gateway;

import com.sudojava.activiti.initdata.ProjectInitManager;
import org.activiti.engine.RepositoryService;
import org.activiti.engine.RuntimeService;
import org.activiti.engine.repository.Deployment;
import org.activiti.engine.runtime.ProcessInstance;
import org.activiti.engine.task.Task;
import org.junit.Test;

import java.io.InputStream;
import java.util.HashMap;
import java.util.Map;
import java.util.function.Consumer;

public class InclusiveGateWay {
    private ProjectInitManager manager;

    public InclusiveGateWay(){
        manager = new ProjectInitManager();
    }

    @Test
    public void deployProcess(){
        RepositoryService repositoryService =
                manager.getRepositoryService();
        InputStream inputStream =
this.getClass().getClassLoader().getResourceAsStream("inclusive.bpmn");
        Deployment deployment =
                repositoryService.createDeployment()
```

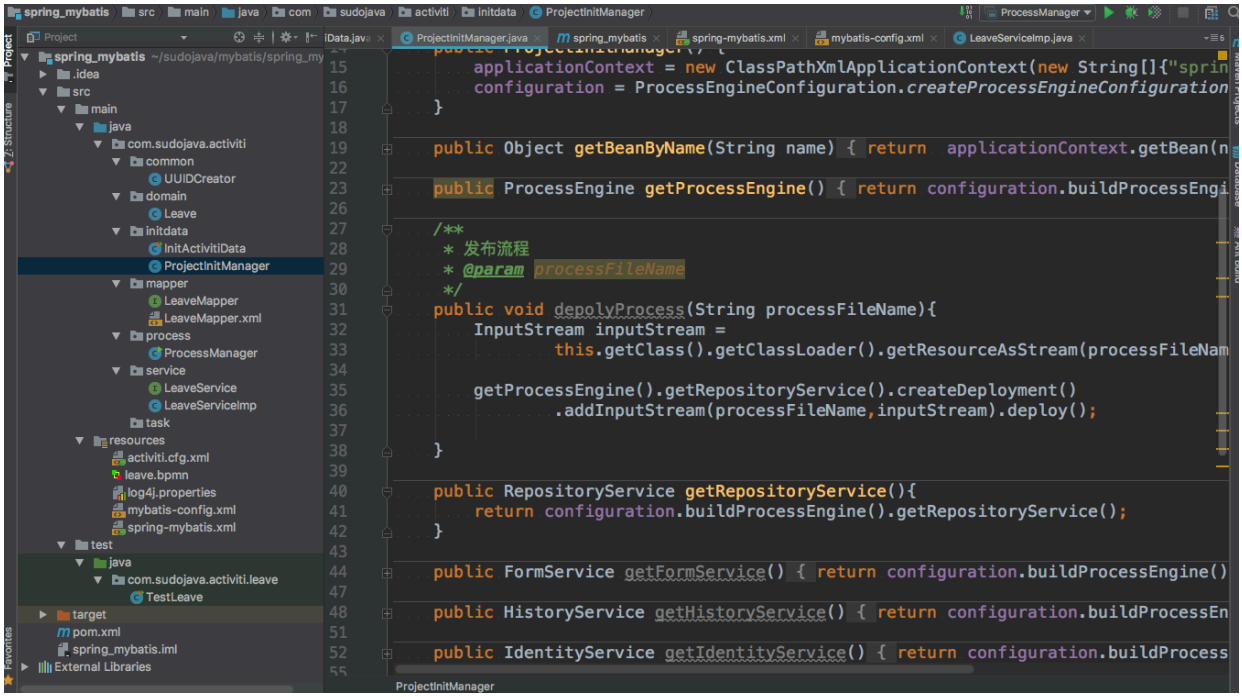
```
                        .addInputStream("inclusive.bpmn",inputStream)
                        .name("包含网关")
                        .category("包含网关").deploy();
        System.out.println("--->>" + deployment.getId());
        System.out.println("--->>" + deployment.getName());
    }

    @Test
    public void startProcessInstance() {
        RuntimeService runtimeService = manager.getRuntimeService();
        Map<String, Object> variable = new HashMap<>();
        variable.put("flag", "2");
        ProcessInstance processInstance =
                runtimeService.startProcessInstanceByKey("myProcess_1",
variable);
        System.out.println("---->>" + processInstance.getId());
        System.out.println("---->>" +
processInstance.getProcessDefinitionName());
        System.out.println("---->>" +
processInstance.getProcessDefinitionId());
        System.out.println("---->>" + processInstance.getBusinessKey());
    }

    /**
     * 查看我的待办事宜
     */
    @Test
    public void findPersonalTask() {

        String userID = "lisi";

 manager.getTaskService().createTaskQuery().taskAssignee(userID).processDef
initionKey("myProcess_1").list().forEach(new Consumer<Task>() {
            public void accept(Task task) {
                System.out.println("id=" + task.getId());
                System.out.println("name=" + task.getName());
                System.out.println("assinee=" + task.getAssignee());
                System.out.println("createTime=" + task.getCreateTime());
                System.out.println("executionId=" + task.getExecutionId());
            }
        });
    }

    /**
     * 结束我的待办事宜
     */
    @Test
    public void completePersonalTask() {
        String taskID = "10006";
```

```
        manager.getTaskService().complete(taskID);
    }
}
```

第十六章：Activiti集成Spring

Mybatis实现自定义表单

工程结构图：



16.1 配置pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.sudojava.springmybatis</groupId>
    <artifactId>spring_mybatis</artifactId>
    <version>1.0-SNAPSHOT</version>
    <build>
        <plugins>
```

```
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
       <!--maven加载src下的xml文件-->
        <resources>
            <resource>
                <directory>src/main/java/</directory>
                <includes>
                    <include>**/*.xml</include>
                </includes>
            </resource>
        </resources>
    </build>

    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>4.3.6.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>4.3.6.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-test</artifactId>
            <version>4.3.6.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-tx</artifactId>
            <version>4.3.6.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.activiti</groupId>
            <artifactId>activiti-spring</artifactId>
            <version>5.18.0</version>
        </dependency>

        <dependency>
            <groupId>org.activiti</groupId>
```

```
            <artifactId>activiti-engine</artifactId>
            <version>5.18.0</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
            <version>4.3.6.RELEASE</version>
        </dependency>

        <!--
https://mvnrepository.com/artifact/org.mybatis.generator/mybatis-generator-
core -->
        <dependency>
            <groupId>org.mybatis.generator</groupId>
            <artifactId>mybatis-generator-core</artifactId>
            <version>1.3.3</version>
        </dependency>
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis-spring</artifactId>
            <version>1.3.1</version>
        </dependency>
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>3.4.4</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.44</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>1.7.10</version>
        </dependency>
        <dependency>
            <groupId>commons-dbcp</groupId>
            <artifactId>commons-dbcp</artifactId>
            <version>1.4</version>
        </dependency>

        <dependency>
```

```
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>1.7.10</version>
        </dependency>
    </dependencies>
</project>
```

16.2、配置Activiti工作的引擎和数据库连接

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
        <property name="maxActive" value="3"/>
        <property name="maxIdle" value="1"/>
        <property name="username" value="root"/>
        <property name="password" value="root"/>
        <property name="url" value="jdbc:mysql://localhost:3306/activiti2?
useUnicode=true&amp;characterEncoding=UTF-8"/>
        <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
    </bean>
    <bean id="processEngineConfiguration"
          class="org.activiti.spring.SpringProcessEngineConfiguration">
        <property name="dataSource" ref="dataSource"/>
        <property name="transactionManager" ref="transactionManager"/>
        <property name="databaseSchemaUpdate" value="true"/>
    </bean>
    <!-- (事务管理)transaction manager, use JtaTransactionManager for global
tx -->
    <bean id="transactionManager"

 class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource" ref="dataSource"/>
    </bean>
</beans>
```

16.3、spring集成mybatis环境配置

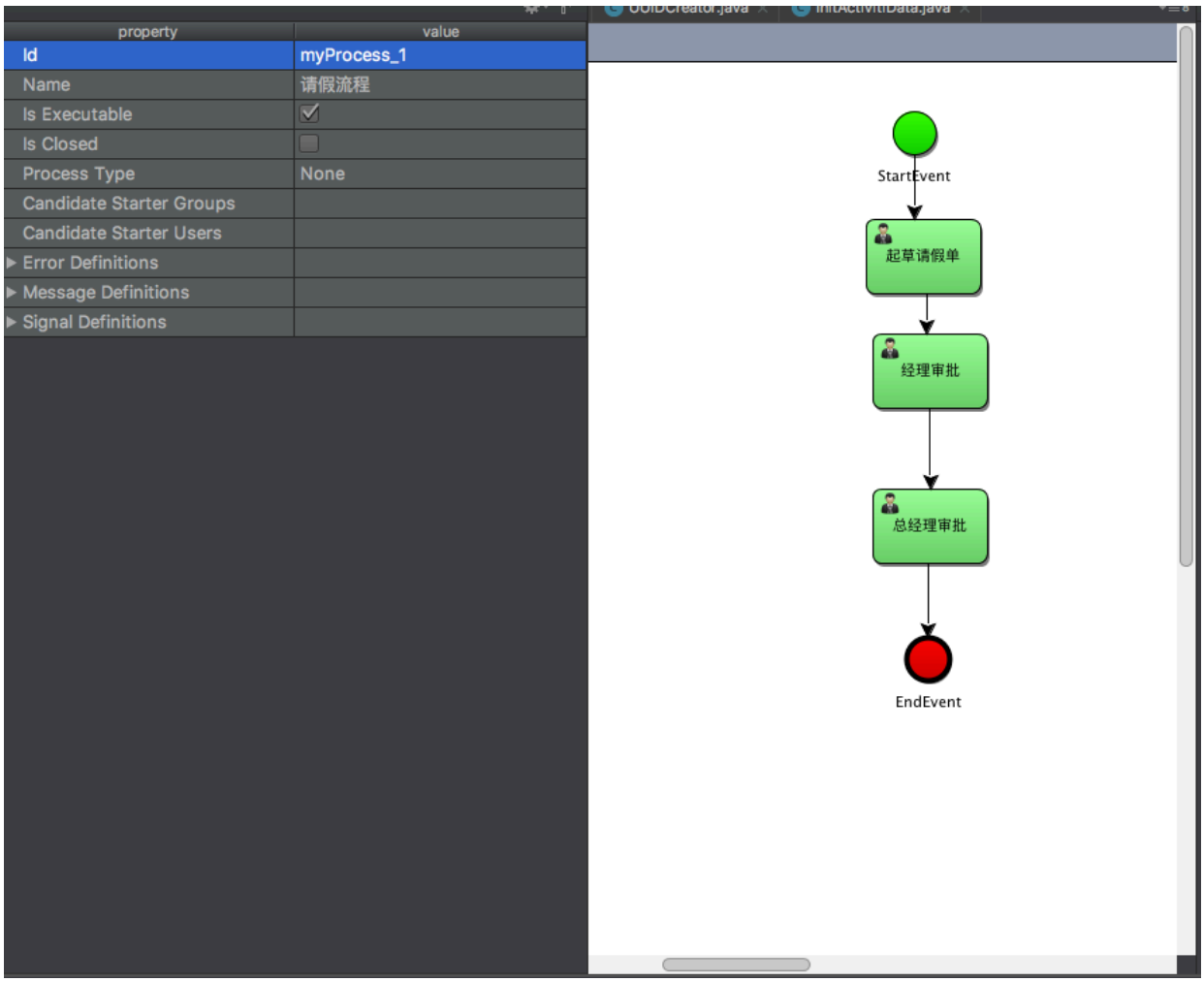
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:p="http://www.springframework.org/schema/p"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:jee="http://www.springframework.org/schema/jee"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="
        http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-4.0.xsd
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
```

```
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.0.xsd
        http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-4.0.xsd
        http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.0.xsd">


<context:component-scan base-package="com.sudojava.activiti"/>


    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
        <property name="maxActive" value="3"/>
        <property name="maxIdle" value="1"/>
        <property name="username" value="root"/>
        <property name="password" value="root"/>
        <property name="url" value="jdbc:mysql://localhost:3306/activiti2?
useUnicode=true&amp;characterEncoding=UTF-8"/>
        <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
    </bean>

    <!-- spring和MyBatis完美整合，不需要mybatis的配置映射文件 -->
    <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <!-- 自动扫描mapping.xml文件，**表示迭代查找 -->
        <!--<property name="mapperLocations"
value="classpath:com/sudojava/activiti/**/*.xml" />-->
        <property name="configLocation" value="classpath:mybatis-
config.xml"/>
    </bean>
    <!-- DAO接口所在包名，Spring会自动查找其下的类 ,包下的类需要使用@MapperScan注解,
否则容器注入会失败 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.sudojava.activiti.mapper"
/>
        <property name="sqlSessionFactoryBeanName"
value="sqlSessionFactory" />
    </bean>
    <!-- (事务管理)transaction manager, use JtaTransactionManager for global
tx -->
    <bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource" ref="dataSource" />
    </bean>
</beans>
```

16.4、配置mybatis-config.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
        PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <!-- 别名 -->
    <typeAliases>
        <package name="com.sudojava.activiti.domain"/>
    </typeAliases>
</configuration>
```

16.5、绘制请假流程单



16.6、初始化Activiti数据库操作

```
package com.sudojava.activiti.initdata;

import org.activiti.engine.ProcessEngine;
import org.activiti.engine.ProcessEngineConfiguration;
import org.junit.Test;

public class InitActivitiData {

    @Test
    public void initData(){
        ProcessEngineConfiguration configuration =

 ProcessEngineConfiguration.createProcessEngineConfigurationFromResource("a
ctiviti.cfg.xml");
        ProcessEngine processEngine =configuration.buildProcessEngine();
        System.out.println(processEngine);
    }

}
```

16.7、具体业务逻辑管理类

```
package com.sudojava.activiti.initdata;

import org.activiti.engine.*;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.io.InputStream;

public class ProjectInitManager {

    private ApplicationContext applicationContext;
    private ProcessEngineConfiguration configuration;

    public ProjectInitManager() {
        applicationContext = new ClassPathXmlApplicationContext(new
String[]{"spring-mybatis.xml"});
        configuration =
ProcessEngineConfiguration.createProcessEngineConfigurationFromResource("ac
tiviti.cfg.xml");
    }

    public Object getBeanByName(String name) {
        return  applicationContext.getBean(name);
    }
```

```
    public ProcessEngine getProcessEngine(){
        return configuration.buildProcessEngine();
    }

    /**
     * 发布流程
     * @param processFileName
     */
    public void depolyProcess(String processFileName){
        InputStream inputStream =

this.getClass().getClassLoader().getResourceAsStream(processFileName);

        getProcessEngine().getRepositoryService().createDeployment()
                .addInputStream(processFileName,inputStream).deploy();

    }

    public RepositoryService getRepositoryService(){
        return configuration.buildProcessEngine().getRepositoryService();
    }

    public FormService getFormService(){
        return configuration.buildProcessEngine().getFormService();
    }

    public HistoryService getHistoryService(){
        return configuration.buildProcessEngine().getHistoryService();
    }

    public IdentityService getIdentityService(){
        return configuration.buildProcessEngine().getIdentityService();
    }

    public ManagementService getManagementService(){
        return configuration.buildProcessEngine().getManagementService();
    }

    public RuntimeService getRuntimeService(){
        return configuration.buildProcessEngine().getRuntimeService();
    }

    public TaskService getTaskService(){
        return configuration.buildProcessEngine().getTaskService();
    }

    /**
     * 初始化数据库
```

```
     */
    public void initActivitiData(){
        configuration.buildProcessEngine();
    }

}
```

16.8 编写请假单的实体类和service接口实现类

Leave请假单：

```
package com.sudojava.activiti.domain;

public class Leave {
    private String id;
    private String leaveName;
    private String leaveReason;
    private Integer leaveDay;
    private String advise;

    public Leave(String id, String leaveName, String leaveReason, Integer
leaveDay, String advise) {
        this.id = id;
        this.leaveName = leaveName;
        this.leaveReason = leaveReason;
        this.leaveDay = leaveDay;
        this.advise = advise;
    }

    public Leave() {

    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
```

```
    }

    public String getLeaveName() {
        return leaveName;
    }

    public void setLeaveName(String leaveName) {
        this.leaveName = leaveName;
    }

    public String getLeaveReason() {
        return leaveReason;
    }

    public void setLeaveReason(String leaveReason) {
        this.leaveReason = leaveReason;
    }

    public Integer getLeaveDay() {
        return leaveDay;
    }

    public void setLeaveDay(Integer leaveDay) {
        this.leaveDay = leaveDay;
    }

    public String getAdvise() {
        return advise;
    }

    public void setAdvise(String advise) {
        this.advise = advise;
    }

    @Override
    public String toString() {
        return "Leave{" +
                ", id=" + id +
                ", leaveName='" + leaveName + '\'' +
                ", leaveReason='" + leaveReason + '\'' +
                ", leaveDay=" + leaveDay +
                ", advise='" + advise + '\'' +
                '}';
    }

}
```

LeaveService类

主要是实现对请假单的CRUD功能

```
package com.sudojava.activiti.service;

import com.sudojava.activiti.domain.Leave;

public interface LeaveService {

    public void saveLeave(Leave leave);

    public Leave findLeaveByID(String id);

    public void updateLeave(Leave leave);

    public void deleteLeaveByID(String id);

}
```

LeaveServiceImp实现类

```java
package com.sudojava.activiti.service;

import com.sudojava.activiti.domain.Leave;
import com.sudojava.activiti.mapper.LeaveMapper;
import org.springframework.stereotype.Service;

import javax.annotation.Resource;


@Service("leaveService")
public class LeaveServiceImp implements LeaveService {

    @Resource
    private LeaveMapper leaveMapper;

    @Override
    public void saveLeave(Leave leave) {

        leaveMapper.saveLeave(leave);

    }
    @Override
    public Leave findLeaveByID(String id) {
        return leaveMapper.findLeaveByID(id);
    }

    @Override
    public void updateLeave(Leave leave) {
        leaveMapper.updateLeave(leave);
    }

    @Override
    public void deleteLeaveByID(String id) {
         leaveMapper.deleteLeaveByID(id);
    }
}
```

16.9、mybatis接口以及mapper文件

MyBatis接口定义

```java
package com.sudojava.activiti.mapper;

import com.sudojava.activiti.domain.Leave;


public interface LeaveMapper {

    public void saveLeave(Leave leave);

    public Leave findLeaveByID(String id);

    public void updateLeave(Leave leave);

    public void deleteLeaveByID(String id);


}
```

Mapper XML文件定义

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.sudojava.activiti.mapper.LeaveMapper">
    <insert id="saveLeave"
parameterType="com.sudojava.activiti.domain.Leave">
        INSERT INTO leave1
        (id,leaveName,leaveReason,leaveDay,advise)
        VALUES
        (#{id}, #{leaveName},#{leaveReason},#{leaveDay},#{advise})
    </insert>

    <select id="findLeaveByID"
resultType="com.sudojava.activiti.domain.Leave" parameterType="string">
        select * from leave1 where id = #{id}
    </select>

    <update id="updateLeave"
parameterType="com.sudojava.activiti.domain.Leave">
        update leave1 set advise=#{advise} where id=#{id}
    </update>

    <delete id="deleteLeaveByID" parameterType="string">
        DELETE * from leave1 where id = #{id}
    </delete>

</mapper>
```

16.10、主键生成策略和核心测试类

UUID生成策略

```
package com.sudojava.activiti.common;

import java.util.UUID;

public class UUIDCreator {

    public static String createUUID(){

        return UUID.randomUUID().toString().replaceAll("-","")
                .substring(1,25);
    }

}
```

核心测试类ProcessManager

```
package com.sudojava.activiti.process;

import com.sudojava.activiti.common.UUIDCreator;
import com.sudojava.activiti.domain.Leave;
import com.sudojava.activiti.initdata.ProjectInitManager;
import com.sudojava.activiti.service.LeaveService;
import org.activiti.engine.RepositoryService;
import org.activiti.engine.RuntimeService;
import org.activiti.engine.TaskService;
import org.activiti.engine.runtime.ProcessInstance;
import org.activiti.engine.runtime.ProcessInstanceQuery;
import org.activiti.engine.task.Task;
import org.activiti.engine.task.TaskQuery;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.function.Consumer;
import java.util.zip.ZipInputStream;

public class ProcessManager {

    private ProjectInitManager manager;

    public ProcessManager() {
        manager = new ProjectInitManager();
    }
```

```
    public void initData() {
        manager.initActivitiData();
    }

    /**
     * 发布流程
     *
     * @param processFileName
     */
    public void deployProcess(String processFileName) {
        InputStream inputStream =

 this.getClass().getClassLoader().getResourceAsStream(processFileName);
        RepositoryService repositoryService =
manager.getRepositoryService();

 repositoryService.createDeployment().addInputStream(processFileName,
inputStream).deploy();
    }

    /**
     * @param processFileName
     * @throws IOException
     */
    public void deployProcessByZip(String processFileName) throws
IOException {
        ZipInputStream zipInputStream = new ZipInputStream(new
FileInputStream(processFileName));
        RepositoryService repositoryService =
manager.getRepositoryService();

 repositoryService.createDeployment().addZipInputStream(zipInputStream).dep
loy();
    }

    /**
     * @param id
     */
    public void deleteProcessByID(String id) {
        RepositoryService repositoryService =
manager.getRepositoryService();
        repositoryService.deleteDeployment(id, true);
    }

    public void startProcessInstance() {
        RuntimeService runtimeService = manager.getRuntimeService();
```

```
        //起草请假流程时候顺便填写请假单，直接调用自定义的service接口，把请假的信息写入
数据库中
        LeaveService leaveService = (LeaveService)
manager.getBeanByName("leaveService");
        Map<String, Object> variables = new HashMap<String, Object>();
        variables.put("userID", "zhang");
        String businessKey = UUIDCreator.createUUID();
        ProcessInstance processInstance = runtimeService.
                startProcessInstanceByKey("myProcess_1", businessKey,
variables);

        System.out.println("--->>" + processInstance.isSuspended());//无效
        System.out.println("--->>" + processInstance.isEnded());//无效

        Leave leave = new Leave();
        leave.setId(businessKey);
        leave.setLeaveName("请假条");
        leave.setLeaveDay(2);
        leave.setAdvise("同意");
        leave.setLeaveReason("肚子疼");
        leaveService.saveLeave(leave);

    }

    /**
     * 查询流程实例的相关信息
     */
    public void queryProcessInstance(){
        RuntimeService runtimeService = manager.getRuntimeService();
        ProcessInstanceQuery processInstanceQuery =
runtimeService.createProcessInstanceQuery();
        String processDefinitionKey = "myProcess_1";

 processInstanceQuery.processDefinitionKey(processDefinitionKey).list().for
Each(new Consumer<ProcessInstance>() {
            @Override
            public void accept(ProcessInstance processInstance) {
                System.out.println("--流程实例的ID--
>>"+processInstance.getId());
                System.out.println("--流程业务的KEY--
>>"+processInstance.getBusinessKey());
                System.out.println("--流程实例的名称--
>>"+processInstance.getName());
                System.out.println("--流程定义的ID--
>>"+processInstance.getProcessDefinitionId());
                System.out.println("--流程当前活动节点的ID--
>>"+processInstance.getActivityId());
                System.out.println("----
>>"+processInstance.getProcessDefinitionKey());
```

```
                System.out.println("----
>>"+processInstance.getProcessDefinitionName());
                System.out.println("---->>"+processInstance.getTenantId());
                System.out.println("---->>"+processInstance.getParentId());
                System.out.println("----
>>"+processInstance.getProcessDefinitionVersion());
                System.out.println("----------------------------------------
------");
            }
        });
    }
    /**
     * 结束我的待办事宜
     * @param taskID
     * @param variables
     */
    public void completePersonalTask(String taskID,Map<String, Object>
variables){
        manager.getTaskService().complete(taskID,variables);
    }

    public void findPersonalTaskList(String taskMan, String processKey) {
        TaskService taskService = manager.getTaskService();
        TaskQuery taskQuery = taskService.createTaskQuery();

        //获取运行时状态
        RuntimeService runtimeService = manager.getRuntimeService();
        //如何获取业务流程中的bussinessKey，可以通过个人待办任务获取流程实例的ID，最终
来获取bussinessKey
        taskQuery.taskAssignee(taskMan);
        taskQuery.processDefinitionKey(processKey);

        LeaveService leaveService = (LeaveService)
manager.getBeanByName("leaveService");
//
//        Leave leave = leaveService.findLeaveByID(bussinessKey);
//        System.out.println(leave);

        List<Task> list = taskQuery.list();
        for (Task task : list) {
            System.out.println(task.getId());
            System.out.println(task.getAssignee());
            System.out.println(task.getName());
            System.out.println(task.getOwner());
            System.out.println(task.getCategory());
            System.out.println(task.getCreateTime());
            System.out.println(task.getExecutionId());
            System.out.println(task.getTaskDefinitionKey());
            System.out.println(task.getProcessDefinitionId());
```

```
            String bussinessKey =
runtimeService.createProcessInstanceQuery()

.processInstanceId(task.getProcessInstanceId()).singleResult().getBusinessK
ey();
            Leave leave = leaveService.findLeaveByID(bussinessKey);
            System.out.println("----------------------" + leave + "---------
---------------");
        }
    }

    public static void main(String[] args) {
        ProcessManager manager = new ProcessManager();
        //manager.initData();
        //manager.deployProcess("leave.bpmn");
        //manager.startProcessInstance();
    //    manager.findPersonalTaskList("luo", "myProcess_1");
//        Map<String,Object> variables = new HashMap<>();
//        variables.put("userID","luo");
//        manager.completePersonalTask("5005", variables);
        manager.queryProcessInstance();
    }

}
```

总结

作业

1. 使用Activiti工作流完成OA中的请假审批流程

面试题