SaaS-IHRM 项目-Activiti7

审批中心模块

传智播客-研究院

第1章 SaaS-IHRM 项目介绍

1.1 项目介绍

SaaS-IHRM 是基于 SaaS 模式的人力资源管理系统。他不同于传统的人力资源软件应用，使用者只需打开浏览器即可管理上百人的薪酬、绩效、社保、入职离职。

1.2 审批中心模块介绍

业务流是项目中最为核心的部分，为了能够将业务流程进一步规范化出来，在我们的 SaaS-IHRM 项目中也可以将日常业务规范化处理，这样我们就可以使用所学的工作流引擎来完成操作。我们的审批中心模块：

请假流程
公告流程
调薪流程
考勤流程
调岗流程

本次项目，我们先以请假流程来进行分析和讲解。

1.2.1 请假流程业务

请假的业务，是我们最为熟悉的一套业务了，主要包括请假当事人，以及他的直属领导，或更高层的领导来逐个进行审批，最后完成一个请假的流程。每个公司的请假流程细节上可能存在差异，但总体流程都差不多。下面我们一起来看一个简版的请假流程，如下：

1）当事人发起请假申请

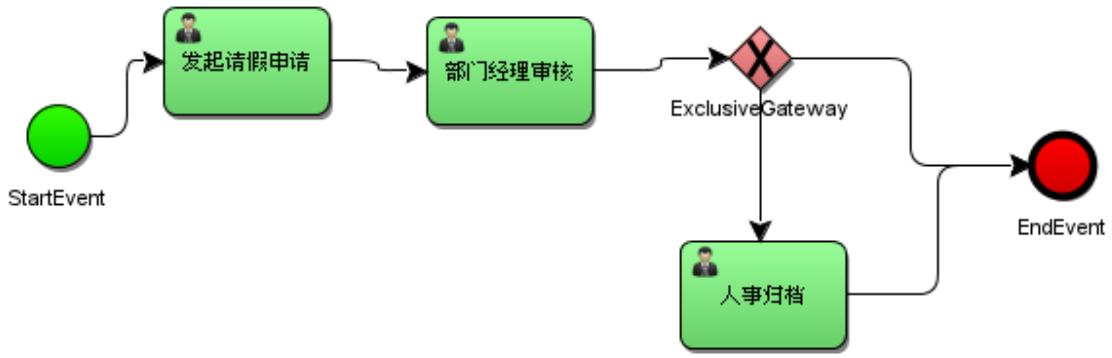
2）直属领导根据实际情况进行审核，如果没有问题就通过，流程继续向后执行。
如果有问题就可以不通过，流程就终止。

3）当直属领导审核通过后，就可以由人事归档，该归档成功后继续后面流程。

4）归档成功，请假流程就结束了。

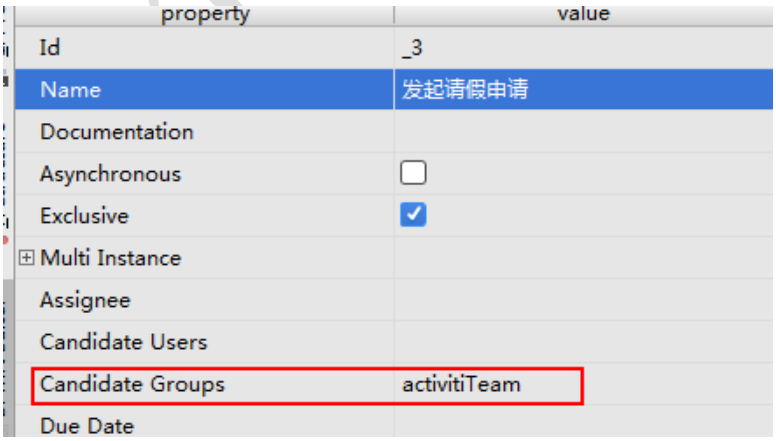
1.2.2 请假流程的流程定义

请假流程，在 IDEA 工具中实现具体的 BPMN 文件如下：



1.3 设置任务执行者

请假流程中，我们将流程的任务执行者，任务执行者可以是单个用户（Assignee），候选用户组（Candidate Users），候选组（Candidate Groups）。本次操作我们打算使用候选组来设置流程的执行者。



1.4 设置任务分支条件

请假当事人发起请求时，首先经过直属领导的审核，此时直属领导可以根据情况决定是否通过，此时流程就会根据情况进入不同的走向，我们在此设置 agree 的流程变量来保存，agree 取值为 1 时，代表同意；agree 取值为 0 时，代表不同意。

property	value
Name	
Condition	${agree==0}
Documentation	
⊞ Execution Listeners	

第2章 Activiti7 工作流系统

2.1 流程部署

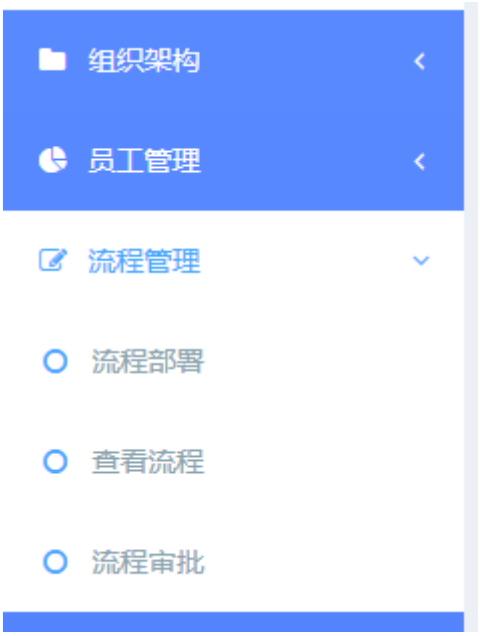
通常情况下，一些流程是提前规定好的（比如我们的请假流程），此时我们可以提前在我们的流程设计器中将流程先设计出来，再将已设计好的流程通过文件上传方式，先上传到服务器，同时再将流程定义文件部署到 activti 流程引擎中。

2.1.1 流程相关菜单制作

首先，我们需要一起来改造我们的菜单，使得左侧菜单支持我们的流程管理。

```
    <li class="treeview active">
          <a href="#">
              <i class="fa fa-edit"></i> <span>流程管理</span>
              <span class="pull-right-container">
  <i class="fa fa-angle-left pull-right"></i>
</span>
          </a>
          <ul class="treeview-menu">
              <li><a href="all-admin-blank"><i class="fa fa-circle-o"></i> 流程部署
</a></li>
              <li><a href="/pages/search-Process"><i class="fa fa-circle-o"></i> 查看
流程</a></li>
              <li><a href="../forms/editors.html"><i class="fa fa-circle-o"></i> 流程
审批</a></li>
          </ul>
      </li>
```

改造完成后的页面效果如下：



2.1.2 流程部署页面

将已设置好的请假流程部署到 activiti 系统 中。
首先添加流程部署页面，因为要实现请假流程的文件上传，所以采用 post 方式提交表单。

```
    <form          th:action="@{/pages/deployment}"                    method="post"
enctype="multipart/form-data">
    <div>
        <ul class="formInfo">
            <li>
                <label>  上  传  BPMN  :  </label><input  type="file"  name="bpmn"
class="input-mini">
            </li>
        </ul>
        <p class="infoAlign">
            <button type="/submit" class="el-button el-button--primary el-button--mini"
title="设置">
                <!----><span>提交</span></button>
            <button type="button" class="el-button el-button--primary el-button--mini"
title="设置">
                <!----><span>取消</span></button>
        </p>
    </div>
</form>
```

页面运行效果如下：

2.2 Thymeleaf 模板技术

Thymeleaf 是一个跟 Velocity、FreeMarker 类似的模板引擎，它可以完全替代 JSP。

a) 为什么要使用 Thymeleaf 模板技术

Activiti7 强耦合 SpringSecurtiy，这样就会导致表单在进行 POST 提交时，容易产生 csrf 过滤器被拦截的问题，而普通的 get 请求是不会产生问题的。为什么会这样?

b) CsrfFilter 过滤器与 POST 请求的矛盾

我们在进行文件上传时，都会采用 POST 提交数据。而 Activiti7 强耦合 Spring Security，默认启用 csrf 后，所有 http 请求都被会 CsrfFilter 拦截，而 CsrfFilter 中有一个私有类 DefaultRequiresCsrfMatcher。我们一起查看源码，如下：

```
    private static final class DefaultRequiresCsrfMatcher implements RequestMatcher {
          private              Pattern              allowedMethods              =
Pattern.compile("^(GET|HEAD|TRACE|OPTIONS)$");

          public boolean matches(HttpServletRequest request) {
             return !allowedMethods.matcher(request.getMethod()).matches();
          }
       }
```

从源码中发现，CsrfFilter 过滤器的确默认情况下，将 POST 请求拦截了，没有放行。这就导致了我们的表单提交时出现了 403 状态码（禁止访问）。

也就是说只有 GET|HEAD|TRACE|OPTIONS 这 4 类方法会被放行，其它 Method 的 http 请求，都要验证_csrf 的 token 是否正确，而通常 post 方式调用 rest 服务时，又没有_csrf 的 token，所以校验失败。

c) 如何解决

我们可以使用 Thymeleaf 模板技术，让服务器端给我们渲染 CsrfFilter 过滤器相关的 token 信息

并回传到客户端，从而让客户端带有 token 信息，所以再次发送 post 请求时就会认为是合法用户，从而解决了 POST 提交的非法访问的问题。

d) ThymeLeaf 模板技术的使用步骤

1) 引入 SpringBoot 与 ThymeLeaf 整合的坐标

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

2) ThymeLeaf 的配置参数

```
#thymeleaf start
spring.thymeleaf.mode=HTML5
spring.thymeleaf.encoding=UTF-8
spring.thymeleaf.content-type=text/html
#开发时关闭缓存,不然没法看到实时页面
spring.thymeleaf.cache=false
#thymeleaf end
```

配置经验：

你可能会发现在默认配置下，thymeleaf 对.html 的内容要求很严格，比如<meta charset="UTF-8" />，如果少最后的标签封闭符号/，就会报错而转到错误页。也比如你在使用 Vue.js 这样的库，然后有<div v-cloak> 这样的 html 代码，
也会被 thymeleaf 认为不符合要求而抛出错误。

因此，建议增加下面这段：

```
spring.thymeleaf.mode = LEGACYHTML5
```

spring.thymeleaf.mode 的默认值是 HTML5，其实是一个很严格的检查，改为 LEGACYHTML5 可以得到一个可能更友好亲切的格式要求。

需要注意的是，LEGACYHTML5 需要搭配一个额外的库 NekoHTML 才可用。

```
<dependency>
      <groupId>net.sourceforge.nekohtml</groupId>
      <artifactId>nekohtml</artifactId>
      <version>1.9.22</version>
</dependency>
```

最后重启项目就可以感受到不那么严格的 thymeleaf 了。

3) 编写控制器

```
@Controller
public class Test2Controller {
```

```
    @RequestMapping("/all-admin-blank")
    public String allAdminBlank() {
        return "ablank";
    }

    @RequestMapping("/all-admin-index")
    public String allAdminIndex() {
        return "aindex";
    }

}
```

4) Thymeleaf 编写的页面要点

```
    <!DOCTYPE HTML>
    <html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>hello</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    </head>
    <body>
        <p th:text="'Hello! , ' + ${name} + '!'">itcast</p>
    </body>
    </html>
```

引入 th 标签：
<html xmlns:th="http://www.thymeleaf.org">

通过${…}进行取值，这点和 ONGL 表达式语法一致！
 itcast

用来配合 link src href 使用的语法，类似的标签有:th:href 和 th:src

条件 if-else
<a th:href="@{/login}" th:unless=${session.user != null}>Login
循环

```
    <tr th:each="emp : ${empList}">
            <td th:text="${emp.id}">1</td>
            <td th:text="${emp.name}">海</td>
            <td th:text="${emp.age}">18</td>
        </tr>
```

2.3 实现流程部署

a) 编写流程部署的 Controller

编写 ProcessController 类，并同时注入相关的 ProcessRuntime 和 TaskRuntime 接口，RepositoryService

接口。代码如下：

```
@Controller
@RequestMapping("/pages")
public class ProcessController {
    @Autowired
    private ProcessRuntime processRuntime;
    @Autowired
    private TaskRuntime taskRuntime;
    @Autowired
    private RepositoryService repositoryService;

    @PostMapping("/deployment")
    public String deployment(@RequestParam("bpmn") MultipartFile file) throws Exception
{
        String fileName = file.getOriginalFilename();
        Deployment deploy = repositoryService.createDeployment()
                .addBytes(fileName, file.getBytes()).deploy();
        System.out.println("部署 ID:" + deploy.getId());

        return "ablank";
    }
```

通过上述代码，我们要实现 bpmn 文件的上传，可以使用 SpringMVC 的 MultipartFile 来实现文件保存，当文件上传成功后，就可以使用 repositoryService 来实现文件部署。

b) 查看部署后的数据库

当部署成功后，我们可以查看 activiti 数据库的 act_ge_bytearray 表

ID_	REV_	NAME_	DEPLOYMENT_ID_	BYTES_
c2bfb535-6732-11e9-8e6f-507b9d24484e	1	调研-itcast001.bpmn	c2bfb534-6732-11e9-8e6f-507b9d24484	<?xml version="1.0" encod... 1 Kb...
d9fb7fcc-6a1b-11e9-aac1-507b9d24484e	1	C:\Users\Administrator\	d9fb7fcb-6a1b-11e9-aac1-507b9d24484	<?xml version="1.0" encod... 3 Kb...

2.4 查看所有流程定义

a) 编写查询流程定义的方法

查询所有的流程部署信息，并保存到集合。代码如下：

```
@RequestMapping("/search-Process")
public String searchProcess(Model model) {
    //查看流程定义信息
    ProcessDefinitionQuery                    processDefinitionQuery                    =
repositoryService.createProcessDefinitionQuery();
    List<ProcessDefinition> list = processDefinitionQuery.list();
```

```
    model.addAttribute("list", list);
    System.out.println("==============" + list.size());
    return "searchProcess";
}
```

通过上述代码，首先得到 ProcessDefinitionQuery 对象，再查询出所有的流程定义。

b)展示所有的流程定义信息

searchProcess.html 页面，用于展示流程定义信息列表展示的代码如下：

```
<table class="table table-striped">
    <thead>
    <tr>
        <td>流程部署 id</td>
        <td>流程定义 id</td>
        <td>流程定义名称</td>
        <td>流程定义 key</td>
        <td>流程定义版本</td>
        <td>bpmn</td>
        <td>删除流程</td>
    </tr>
    </thead>
    <tbody>
    <tr th:each="processDefinition:${list}">
        <td th:text="${processDefinition.deploymentId}"></td>
        <td th:text="${processDefinition.id}"></td>
        <td th:text="${processDefinition.name}"></td>
        <td th:text="${processDefinition.key}"></td>
        <td th:text="${processDefinition.version}"></td>
        <td ><a
th:href="@{/pages/viewBpmn(processDefinitionId=${processDefinition.id},resourceType
='bpmn')}">查看 bpmn</a></td>
        <td ><a
th:href="|javascript:deleteDeployment('${processDefinition.deploymentId}')|">删除流程
</a></td>
    </tr>
    </tbody>
</table>
```

该部分使用了 ThymeLeaf 模板实现了流程定义列表的遍历（th:each 来实现），为了实现流程定义信息的输出，使用了 th:text=”${对象名.属性名}”的方式来实现流程定义对象的各个属性输出。
使用 th:href=”@{}”实现超链接的编写
使用 th:href=”|javascript:函数名(参数)|”实现函数调用

页面展示效果如下：

2.5 查看流程定义 BPMN 文件

a) 编写 BPMN 文件的查看方法

步骤如下：

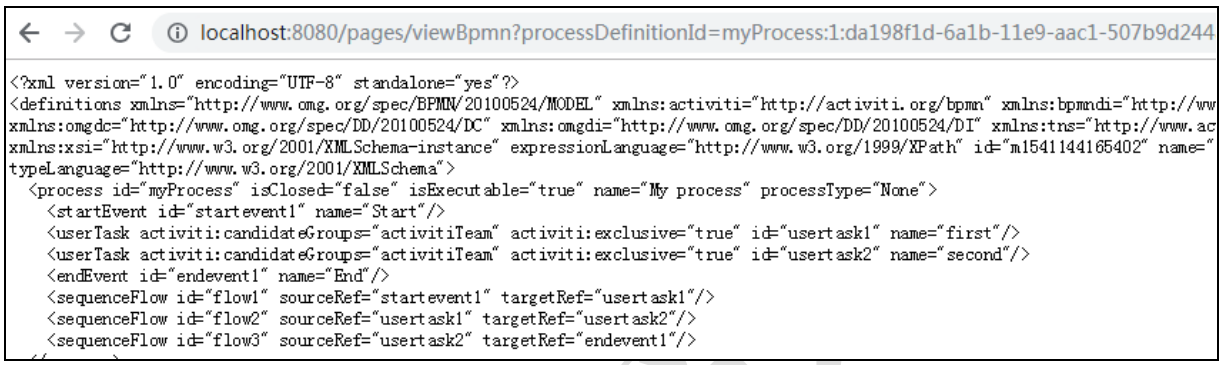
1.得到 ProcessDefinitionQuery 对象
2.根据指定的流程定义 id，获取流程流程定义对象
3.通过流程定义对象，得到流程部署 id 和流程资源文件名
4.通过 repositoryService 对象，得到 bpmn 文件的输入流对象
5.使用 commons-io 包中的工具方法，实现文件复制
6.关闭流对象

实现代码如下：

```
    //查看流程 bpmn 文件
@RequestMapping("/viewBpmn")
public void viewBpmn(String processDefinitionId, String resourceType, Model model,
HttpServletResponse response) throws Exception {
    ProcessDefinitionQuery                    processDefinitionQuery                    =
repositoryService.createProcessDefinitionQuery();
    processDefinitionQuery.processDefinitionId(processDefinitionId);

    ProcessDefinition processDefinition = processDefinitionQuery.singleResult();

    String deploymentId = processDefinition.getDeploymentId();

    InputStream is = null;
    String resourceName =null;
    if ("bpmn".equals(resourceType)) {
        resourceName = processDefinition.getResourceName();
        is = repositoryService.getResourceAsStream(deploymentId, resourceName);
    }
```

```
    ServletOutputStream outputStream = response.getOutputStream();
    IOUtils.copy(is,outputStream);

    outputStream.close();
    is.close();
}
```

b)展示 pbmn 文件查看



2.6 删除流程

a) 编写流程定义删除方法

步骤如下：

1.得到流程定义的 id

2.调用 repositoryService 来实现流程部署信息的级联删除

实现代码如下：

```
@RequestMapping("/deleteDeployment")
public String deleteDeployment(String deploymentId){
    System.out.println(deploymentId);
    repositoryService.deleteDeployment(deploymentId,true);
    return "forward:search-Process";
}
```

注意：为了能够实现流程定义文件删除时，相关的流程信息都实现级联删除，可以调用 repositoryService.deleteDeployment(deploymentId,true)方法，其中第二个参数 true 代表要进行级联删除。

第3章 业务系统-请假流程

3.1 请假表创建

为了能够保存当事人的请假信息，并进行逐步审核，所以我们要有请假基本信息表，审核表。

3.1.1 请假单表

```
create table Holiday_C(
  holiday_id varchar(40) primary key,
  employee_name varchar(40),
  reason varchar(50),
  holiday_num numeric(8,2),
  createtime datetime,
  endtime datetime,
  status varchar(10),
  processinstanceId varchar(100),
  details varchar(400)
);
```

3.1.2 审核意见表

```
create table HolidayAudit_C(
   holidayAudit_id varchar(40),
   employee_name varchar(40),
   audit_name varchar(40),
   auditInfo  varchar(100),
   audit_type varchar(100),
   status   varchar(20),
   createtime datetime
);
```

3.2 请假相关的实体类封装