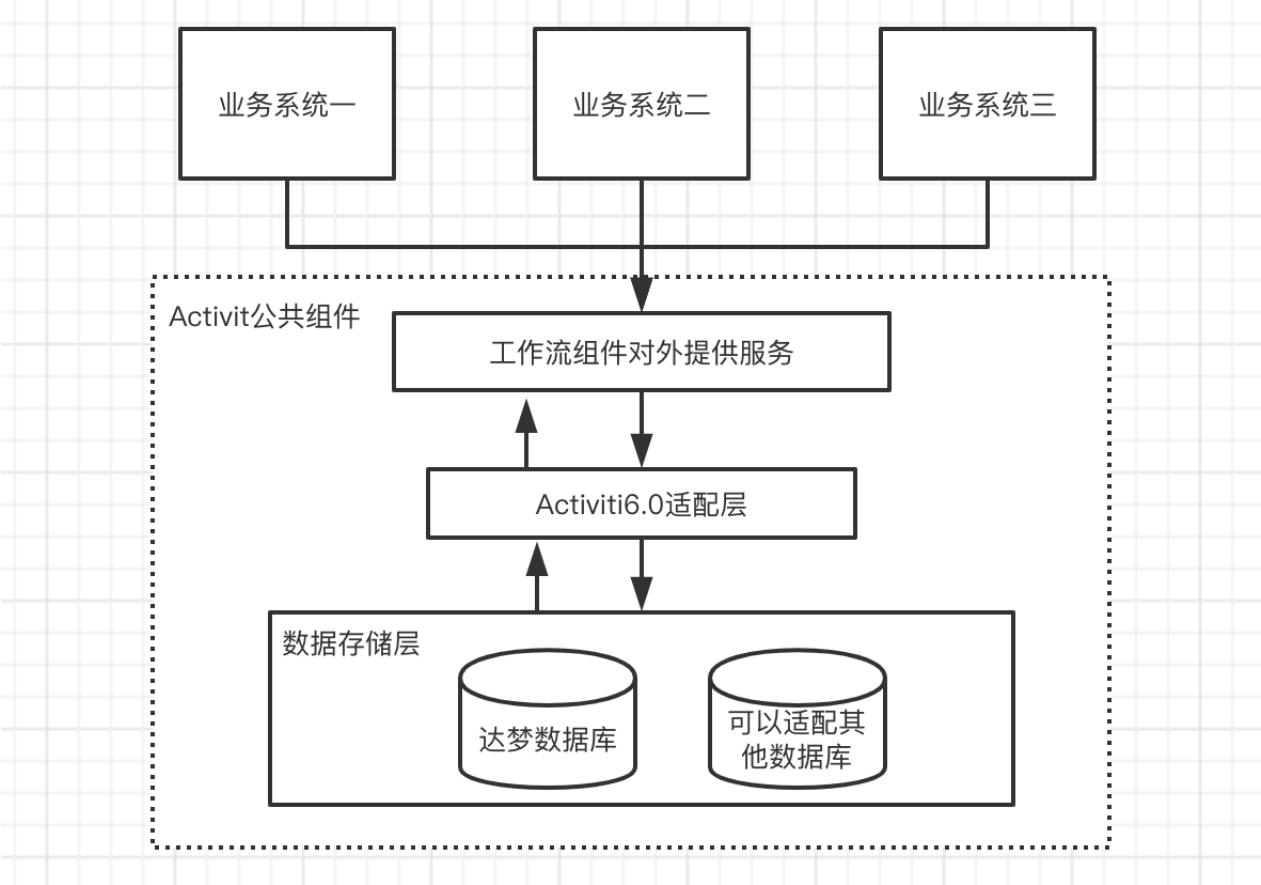
1.Activiti公共组件介绍

Activiti公共组件依赖于Activiti6.0，是一项新的基于Apache许可的开源BPM平台，从基础开始构建，旨在提供支持新的BPMN 2.0标准，包括支持对象管理组（OMG），可以定义流程、执行流程并以不同方式对其实现运行。

简而言之就是一个工作审批流，在我们日程生活中审批很常见，类似于请假审批流程，比如我们发起请假申请，依据请假的天数或者其他条件，可能会经过不同的部门进行审批，比如我们发出的申请会经过SM审核，SM审核通过后会自动发给人事归档，人事审核确认没有问题就通过，整个流程结束。

架构图如下：



2.组件清单

2.1前端组件清单

前端封装好的组件放在公共前端项目的 `src/components/activiti` 目录下。

序号	组件名	组件说明	参数说明

1	`FlowSaveButton`	工作流暂存前端组件	{
			`processDefKey` ： 流程实例,
			`processInstanceId` :流程实例id,
			`afterSuccess` ： 暂存成功后回调
			}

2	`FlowSubmitButton`	工作流流程提交前端组件	{
			`processDefKey` ： 流程实例,
			`processInstanceId` :流程实例id,
			`afterSuccess` ： 提交成功后回调
			}

3	`FlowRejectButton`	工作流流程驳回前端组件	{
			`processInstanceId` :流程实例id,
			`formData` ： 提交到后端数据
			}

4	`FlowHistoryList`	流程历史信息展示前端组件	{
			`processInstanceId` :流程实例id
			}

5	`FlowPicModal`	流程图信息展示Modal组件，依赖于 `WorkFlow` 组件	{
			`processInstanceId` :流程实例id,
			`showFlowModal` ： 是否展示modal框
			}

6	`WorkFlow`	流程图组件	

2.2 后端接口清单

后端接口 `ActivitiCommonService` 放在 `com.zdww.eemp.common.workflow.internal.service` 下

序号	方法名	方法说明	参数说明
1	`String startUp(String processDefKey, String assign, ActivitiEventListener<String> bizTaskInterface);`	启动流程	`processDefKey`：流程定义key `assign`：流程发起人（当前登录用户） `bizTaskInterface`：监听
2	`List<String> getTodoInstances(String assign);`	获取当前登录人代办的流程	`assign`：当前登录用户
3	`List<String> getDoneInstances(String assign);`	获取当前登录人已办的流程	`assign`：当前登录人
4	`InputStream getImgInputStream(String processId);`	获得默认流程图	`processId`：流程实例id
5	`InputStream getFlowPicByInstanceId(String instanceId);`	获取流程图流	`instanceId`：流程实例id
6	`List<FlowHistoryDTO> getFlowHistoryByInstanceId(String instanceId);`	根据流程id获取办理历史记录	`instanceId`：流程实例id
7	`List<ProDefVersion> getProcessDefinitionListByKey(String key);`	根据流程定义key查询所有版本	`key`：流程定义key
8	`TaskDto getNextUserTask(TaskDto taskDto);`	获取下一个任务的节点	`taskDto`：数据传输类
9	`void submitTask(TaskDto taskSubmitDto, ActivitiEventListener<String> activitiEventListener);`	提交任务	`taskSubmitDto`：数据传输类 `activitiEventListener`：事件监听
10	`String rollBack(RejectDTO rejectDTO);`	驳回到指定节点	`rejectDTO`：数据传输类
11	`WfNodeDto getFlowPicData(String processInstanceId);`	获取流程图节点信息，匹配前端 `WorkFlow` 组件	`processInstanceId`：流程实例id

3.组件使用

3.1 引入依赖

```
<dependency>
  <groupId>com.zdww.eemp.common</groupId>
  <artifactId>eemp-common-workflow-starter</artifactId>
  <version>1.0.0-SNAPSHOT</version>
</dependency>
```

3.2 添加配置信息

```
spring:
  # 达梦数据库
  datasource:
    driver-class-name: dm.jdbc.driver.DmDriver
    type: org.apache.commons.dbcp.BasicDataSource
    url: jdbc:dm://10.18.101.81:5236/DB_GSSC_ZT
    username: DB_GSSC_ZT
    password: eemp@V1.0#ZT
  # 工作流
  activiti:
    check-process-definitions: false
    database-schema-update: true
```

```
  #jpa
  jpa:
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
    show-sql: true


# 自定义配置 - 适配dm数据库
activiti:
  database-type: oracle
```

- ### 3.3 注入 `ActivitiCommonService` 类

在需要的时候注入公共service类就可以调用里面的方法

```
@Resource
private ActivitiCommonService activitiCommonService;
```

4. 组件使用示例

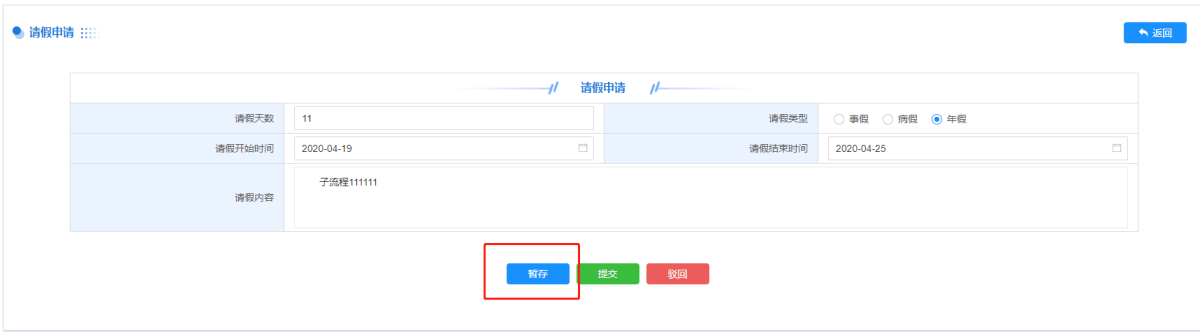
拿发起请假申请示例，我们定义好模型，画好流程图，然后部署好流程。

1. 发起请假申请，在页面加入组件

```
<FlowSaveButton :afterSuccess="handelSubmitSuccess" processDefKey="qxcode3"
:processInstanceId="params.processInstanceId"/>

<script>
import FlowSaveButton from '@/compoents/activity/FlowSaveButton'
export default {
  name: 'applyAdd',
  components: {
    FlowSaveButton
  },
  data() {
    return {
      params: {
        processInstanceId: '' // 流程实例id
      }
    }
  },
  methods: {
    handelSubmitSuccess(processDefKey, processInstanceId, processState) {
      const _this = this
      save({ ...this.params, processInstanceId: processInstanceId, processState:
processState }).then(function(data) {
        _this.$router.go(-1)
      }).catch(error => {
        console.log(error)
      })
    }
  },
  mounted() {
    this.init()
  }
}
```

2. 页面效果



3. 点击暂存调用后端接口 `startUp` 发起流程
4. 后端接口调用示例
    1. `String startUp(String processDefKey, String assign, ActivitiEventListener<String> bizTaskInterface);` 启动流程
    2. 方法入参

参数名	参数类型	备注
`processDefKey`	`String`	流程定义key
`assign`	`String`	流程发起人
`bizTaskInterface`	`ActivitiEventListener`	自定义监听类

    3. 调用示例

```
@Override
public JsonResponse<Map<String, Object>> startUp(String processDefKey) {
  Map<String, Object> retMap = new HashMap<>();
  // 使用TokenUtil.getCurrentUserId()获取当前的登录人
  JsonResponse.ok(activitiCommonService
                  .startUp(processDefKey, TokenUtil.getCurrentUserId(), new
AbstractActivitiEvenet<String>() {
                    @Override
                    public void onSuccess(String t) {
                      retMap.put("processState", "暂存");
                      retMap.put("processInstanceId", t);
                    }
                  }));
  return JsonResponse.ok(retMap);
}
```

    4. 返回值

        无

5. 说明

    当前示例只展示流程发起过程，此外还有流程提交以及流程驳回。

5. 附录