

山东大学

SHANDONG UNIVERSITY



通知管理模块开发文档

课程名称：项目实训

姓名：_____

学号：_____

专业班级：21软件工程1班

授课教师：张亚峰

2024 年 7 月 11 日

目录

- 一、概述
- 二、功能描述
- 三、关系图
- 四、代码结构
 - (一) NoticeController
 - (二) 模板部分
- 五、脚本部分
- 六、样式部分
- 七、API接口
- 八、主要功能实现
 - (一) 获取通知列表
 - (二) 新增通知
 - (三) 编辑通知
 - (四) 删除通知
 - (五) 批量删除
 - (六) 提交表单
- 九、注意事项
- 十、总结

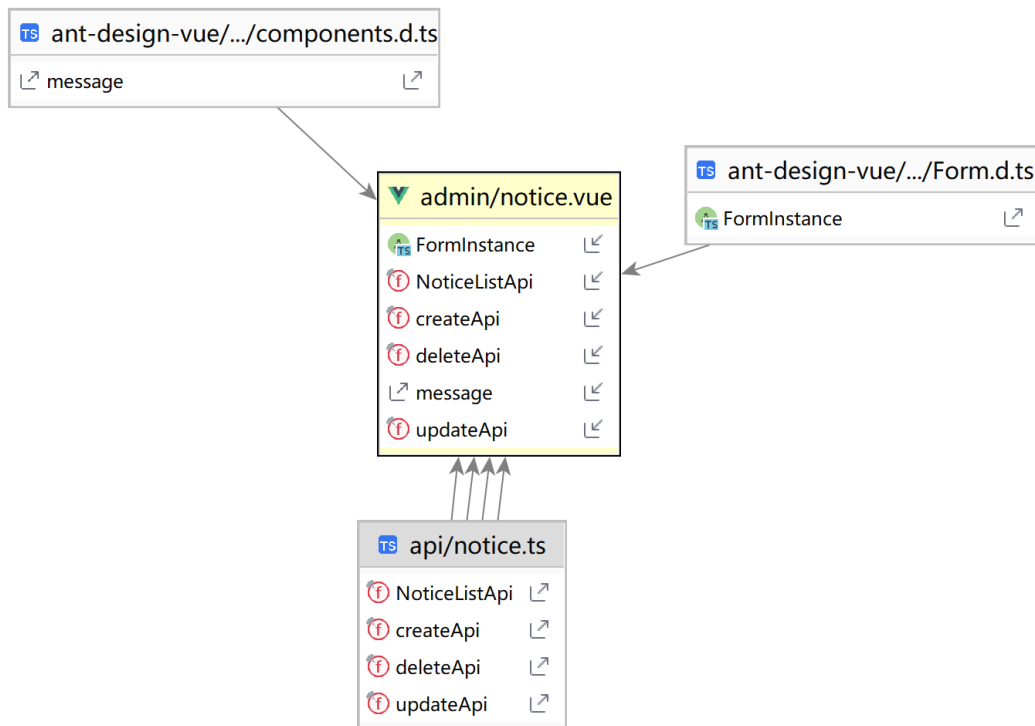
一、概述

通知管理模块是一个用于管理系统通知的功能模块。该模块支持通知的新增、编辑、删除和批量删除等操作，提供了用户友好的界面和高效的操作方式。本模块使用了Vue.js框架，并结合ant-design-vue组件库，确保界面的美观性和一致性。

二、功能描述

- 通知信息展示：**使用a-table组件展示通知列表，包括通知的标题、内容和操作按钮。支持分页显示，每页显示10条通知信息。
- 搜索功能：**可以通过关键词搜索通知信息，并实时更新搜索结果。
- 新增通知：**通过点击“新增”按钮，弹出表单输入通知的标题和内容。支持表单验证，确保必要字段的完整性。
- 编辑通知：**通过点击表格中的“编辑”按钮，弹出表单编辑现有通知的信息。
- 删除通知：**支持单个删除和批量删除操作，确保通知信息的及时更新。

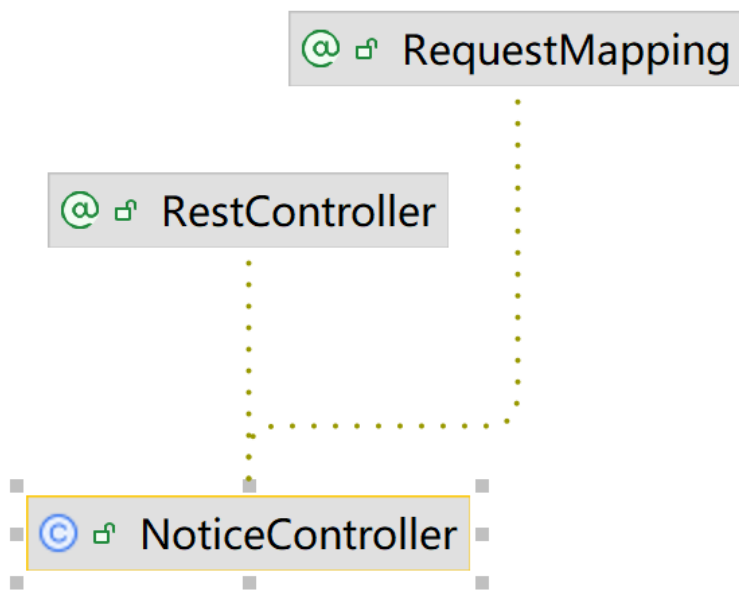
三、关系图



四、代码结构

(一) NoticeController

1. 关系图



2. 功能接口

查询通知列表

URL: `/notice/list`

请求方式: GET

描述: 获取所有通知的列表。

返回值:

- `APIResponse` 对象, 包含查询结果和状态码。

创建通知

URL: `/notice/create`

请求方式: POST

权限要求: 管理员

事务管理: `@Transactional`

描述：创建一个新的通知。

参数：

- **Notice** 对象，包含通知的详细信息。

返回值：

- **APIResponse** 对象，包含创建结果和状态码。

删除通知

URL: `/notice/delete`

请求方式: **POST**

权限要求: 管理员

描述：删除指定 ID 的通知。

参数：

- **String** 类型的通知 ID 列表，逗号分隔。

返回值：

- **APIResponse** 对象，包含删除结果和状态码。

更新通知

URL: `/notice/update`

请求方式: **POST**

权限要求: 管理员

事务管理: **@Transactional**

描述：更新通知的详细信息。

参数：

- **Notice** 对象，包含通知的详细信息。

返回值：

- **APIResponse** 对象，包含更新结果和状态码。

3. 代码

```
1 package com.gk.study.controller;
2
3 import com.gk.study.common.APIResponse;
4 import com.gk.study.common.ResponseCode;
5 import com.gk.study.entity.Notice;
6 import com.gk.study.permission.Access;
7 import com.gk.study.permission.AccessLevel;
8 import com.gk.study.service.NoticeService;
9 import org.slf4j.Logger;
10 import org.slf4j.LoggerFactory;
11 import org.springframework.beans.factory.annotation.Autowired;
12 import org.springframework.transaction.annotation.Transactional;
13 import org.springframework.web.bind.annotation.RequestMapping;
14 import org.springframework.web.bind.annotation.RequestMethod;
15 import org.springframework.web.bind.annotation.RestController;
16
17 import java.io.IOException;
18 import java.util.List;
19
20 @RestController
21 @RequestMapping("/notice")
22 public class NoticeController {
23
24     private final static Logger logger =
25         LoggerFactory.getLogger(NoticeController.class);
26
27     @Autowired
28     NoticeService service;
29
30     @RequestMapping(value = "/list", method = RequestMethod.GET)
31     public APIResponse list() {
32         List<Notice> list = service.getNoticeList();
33         return new APIResponse(ResponseCode.SUCCESS, "查询成功",
34             list);
35     }
36
37     @Access(level = AccessLevel.ADMIN)
38     @RequestMapping(value = "/create", method =
39         RequestMethod.POST)
40     @Transactional
41     public APIResponse create(Notice notice) throws IOException {
42         service.createNotice(notice);
43         return new APIResponse(ResponseCode.SUCCESS, "创建成功");
44     }
45
46     @Access(level = AccessLevel.ADMIN)
```

```

44     @RequestMapping(value = "/delete", method =
RequestMapping.POST)
45     public APIResponse delete(String ids) {
46         System.out.println("ids==" + ids);
47         // 批量删除
48         String[] arr = ids.split(",");
49         for (String id : arr) {
50             service.deleteNotice(id);
51         }
52         return new APIResponse(ResponseCode.SUCCESS, "删除成功");
53     }
54
55     @Access(level = AccessLevel.ADMIN)
56     @RequestMapping(value = "/update", method =
RequestMapping.POST)
57     @Transactional
58     public APIResponse update(Notice notice) throws IOException {
59         service.updateNotice(notice);
60         return new APIResponse(ResponseCode.SUCCESS, "更新成功");
61     }
62
63 }

```

4. 其他略

(二) 模板部分

```

1 <template>
2     <div>
3         <!-- 页面区域 -->
4         <div class="page-view">
5             <div class="table-operations">
6                 <a-space>
7                     <a-button type="primary" @click="handleAdd">新增</a-
button>
8                     <a-button @click="handleBatchDelete">批量删除</a-button>
9                 </a-space>
10            </div>
11            <a-table
12                size="middle"
13                rowKey="id"
14                :loading="data.loading"
15                :columns="columns"
16                :data-source="data.noticeList"
17                :scroll="{ x: 'max-content' }"
18                :row-selection="rowSelection"
19                :pagination="{
20                    size: 'default',

```



```

21         current: data.page,
22         pageSize: data.pageSize,
23         onChange: (current) => (data.page = current),
24         showSizeChanger: false,
25         showTotal: (total) => 共${total}条数据,
26     }"
27 >
28 <template #bodyCell="{ text, record, index, column }">
29     <template v-if="column.key === 'operation'">
30         <span>
31             <a @click="handleEdit(record)">编辑</a>
32             <a-divider type="vertical" />
33             <a-popconfirm title="确定删除?" ok-text="是" cancel-
text="否" @confirm="confirmDelete(record)">
34                 <a href="#">删除</a>
35             </a-popconfirm>
36         </span>
37     </template>
38 </template>
39 </a-table>
40 </div>
41
42 <!-- 弹窗区域 -->
43 <div>
44     <a-modal
45         :visible="modal.visible"
46         :forceRender="true"
47         :title="modal.title"
48         ok-text="确认"
49         cancel-text="取消"
50         @cancel="handleCancel"
51         @ok="handleOk"
52     >
53         <div>
54             <a-form ref="myform" :label-col="{ style: { width:
'80px' } }" :model="modal.form" :rules="modal.rules">
55                 <a-row :gutter="24">
56                     <a-col span="24">
57                         <a-form-item label="标题" name="title">
58                             <a-input placeholder="请输入标题" v-
model:value="modal.form.title"></a-input>
59                         </a-form-item>
60                     </a-col>
61                     <a-col span="24">
62                         <a-form-item label="通知内容" name="content">
63                             <a-textarea placeholder="请输入内容" :rows="4"
v-model:value="modal.form.content"></a-textarea>
64                         </a-form-item>

```

```
65         </a-col>
66     </a-row>
67 </a-form>
68 </div>
69 </a-modal>
70 </div>
71 </div>
72 </template>
```

五、脚本部分

```
1 <script setup lang="ts">
2 import { FormInstance, message } from 'ant-design-vue';
3 import { createApi, NoticeListApi, updateApi, deleteApi } from
  '@api/notice';
4
5 const columns = reactive([
6   {
7     title: '序号',
8     dataIndex: 'index',
9     key: 'index',
10    align: 'center'
11  },
12  {
13    title: '标题',
14    dataIndex: 'title',
15    key: 'title',
16    align: 'center'
17  },
18  {
19    title: '内容',
20    dataIndex: 'content',
21    key: 'content',
22    align: 'center',
23    customRender: ({ text, record, index, column }) =>
      text?.substring(0, 20) + '...',
24  },
25  {
26    title: '操作',
27    dataIndex: 'action',
28    key: 'operation',
29    align: 'center',
30    fixed: 'right',
31    width: 140,
32  },
33 ]);
```

```
34
35 // 页面数据
36 const data = reactive({
37   noticeList: [],
38   loading: false,
39   keyword: '',
40   selectedRowKeys: [] as any[],
41   pageSize: 10,
42   page: 1,
43 });
44
45 // 弹窗数据源
46 const modal = reactive({
47   visile: false,
48   editFlag: false,
49   title: '',
50   form: {
51     id: undefined,
52     title: undefined,
53   },
54   rules: {
55     title: [{ required: true, message: '请输入', trigger:
'change' }],
56   },
57 });
58
59 const myform = ref<FormInstance>();
60
61 onMounted(() => {
62   getDataList();
63 });
64
65 const getDataList = () => {
66   data.loading = true;
67   NoticeListApi({
68     keyword: data.keyword,
69   })
70   .then((res) => {
71     data.loading = false;
72     res.data.forEach((item: any, index: any) => {
73       item.index = index + 1;
74     });
75     data.noticeList = res.data;
76     data.loading = false;
77   })
78   .catch((err) => {
79     data.loading = false;
80     console.log(err);
```

```

81     });
82 };
83
84 const rowSelection = ref({
85   onChange: (selectedRowKeys: (string | number)[], selectedRows:
86     DataItem[]) => {
87     data.selectedRowKeys = selectedRowKeys;
88   },
89 });
90
91 const handleAdd = () => {
92   resetModal();
93   modal.visible = true;
94   modal.editFlag = false;
95   modal.title = '新增';
96   // 重置
97   for (const key in modal.form) {
98     modal.form[key] = undefined;
99   }
100 };
101
102 const handleEdit = (record: any) => {
103   resetModal();
104   modal.visible = true;
105   modal.editFlag = true;
106   modal.title = '编辑';
107   // 重置
108   for (const key in modal.form) {
109     modal.form[key] = undefined;
110   }
111   for (const key in record) {
112     modal.form[key] = record[key];
113   }
114 };
115
116 const confirmDelete = (record: any) => {
117   deleteApi({ ids: record.id })
118     .then((res) => {
119       getDataList();
120     })
121     .catch((err) => {
122       message.error(err.msg || '操作失败');
123     });
124 };
125
126 const handleBatchDelete = () => {
127   if (data.selectedRowKeys.length <= 0) {
128     message.warn('请勾选删除项');
129   }
130 };

```

```
128     return;
129 }
130 deleteApi({ ids: data.selectedRowKeys.join(',') })
131   .then((res) => {
132     message.success('删除成功');
133     data.selectedRowKeys = [];
134     getDataList();
135   })
136   .catch((err) => {
137     message.error(err.msg || '操作失败');
138   });
139 };
140
141 const handleOk = () => {
142   myform.value
143     ?.validate()
144     .then(() => {
145       if (modal.editFlag) {
146         updateApi(modal.form)
147           .then((res) => {
148             hideModal();
149             getDataList();
150           })
151           .catch((err) => {
152             message.error(err.msg || '操作失败');
153           });
154       } else {
155         createApi(modal.form)
156           .then((res) => {
157             hideModal();
158             getDataList();
159           })
160           .catch((err) => {
161             message.error(err.msg || '操作失败');
162           });
163       }
164     })
165     .catch((err) => {
166       console.log('不能为空');
167     });
168 };
169
170 const handleCancel = () => {
171   hideModal();
172 };
173
174 // 恢复表单初始状态
175 const resetModal = () => {
```

```
176   myform.value?.resetFields();
177 };
178
179 // 关闭弹窗
180 const hideModal = () => {
181   modal.visile = false;
182 };
183 </script>
```

六、样式部分

```
1 <style scoped lang="less">
2 .page-view {
3   min-height: 100%;
4   background: #fff;
5   padding: 24px;
6   display: flex;
7   flex-direction: column;
8 }
9
10 .table-operations {
11   margin-bottom: 16px;
12   text-align: right;
13 }
14
15 .table-operations > button {
16   margin-right: 8px;
17 }
18 </style>
```

七、API接口

- 获取通知列表: NoticeListApi({ keyword: data.keyword })
- 新增通知: createApi(modal.form)
- 更新通知: updateApi(modal.form)
- 删除通知: deleteApi({ ids: record.id })

八、主要功能实现

（一）获取通知列表

```
1  const getDataList = () => {
2    data.loading = true;
3    NoticeListApi({
4      keyword: data.keyword,
5    })
6    .then((res) => {
7      data.loading = false;
8      res.data.forEach((item: any, index: any) => {
9        item.index = index + 1;
10      });
11      data.noticeList = res.data;
12      data.loading = false;
13    })
14    .catch((err) => {
15      data.loading = false;
16      console.log(err);
17    });
18  };
```

（二）新增通知

```
1  const handleAdd = () => {
2    resetModal();
3    modal.visile = true;
4    modal.editFlag = false;
5    modal.title = '新增';
6    // 重置
7    for (const key in modal.form) {
8      modal.form[key] = undefined;
9    }
10  };
```

（三）编辑通知

```
1  const handleEdit = (record: any) => {
2    resetModal();
3    modal.visible = true;
4    modal.editFlag = true;
5    modal.title = '编辑';
6    // 重置
7    for (const key in modal.form) {
8      modal.form[key] = undefined;
9    }
10   for (const key in record) {
11     modal.form[key] = record[key];
12   }
13 };
```

(四) 删除通知

```
1  const confirmDelete = (record: any) => {
2    deleteApi({ ids: record.id })
3      .then((res) => {
4        getDataList();
5      })
6      .catch((err) => {
7        message.error(err.msg || '操作失败');
8      });
9  };
```

(五) 批量删除

```
1  const handleBatchDelete = () => {
2    if (data.selectedRowKeys.length <= 0) {
3      message.warn('请勾选删除项');
4      return;
5    }
6    deleteApi({ ids: data.selectedRowKeys.join(',') })
7      .then((res) => {
8        message.success('删除成功');
9        data.selectedRowKeys = [];
10       getDataList();
11     })
12     .catch((err) => {
13       message.error(err.msg || '操作失败');
14     });
15  };
```

（六）提交表单

```
1  const handleOk = () => {
2    myform.value
3    ?.validate()
4    .then(() => {
5      if (modal.editFlag) {
6        updateApi(modal.form)
7          .then((res) => {
8            hideModal();
9            getDataList();
10           })
11          .catch((err) => {
12            message.error(err.msg || '操作失败');
13          });
14      } else {
15        createApi(modal.form)
16          .then((res) => {
17            hideModal();
18            getDataList();
19          })
20          .catch((err) => {
21            message.error(err.msg || '操作失败');
22          });
23      }
24    })
25    .catch((err) => {
26      console.log('不能为空');
27    });
28  };
```

九、注意事项

- 数据格式：确保后端API返回的数据格式与前端组件预期一致，避免数据解析错误。
- 组件库依赖：确保项目中安装并正确配置了ant-design-vue组件库，以支持表格、表单、弹窗等UI组件的使用。

十、总结

通知管理模块为系统提供了一个高效的 notification 管理工具，通过直观的界面和便捷的操作方式，管理员可以轻松管理通知信息，提升工作效率。该模块的设计和实现充分考虑了用户体验，使用现代化的前端技术栈，确保系统的可维护性和可扩展性。开发者可以根据具体需求对模块进行定制和扩展，满足不同场景下的通知管理需求。