



# 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

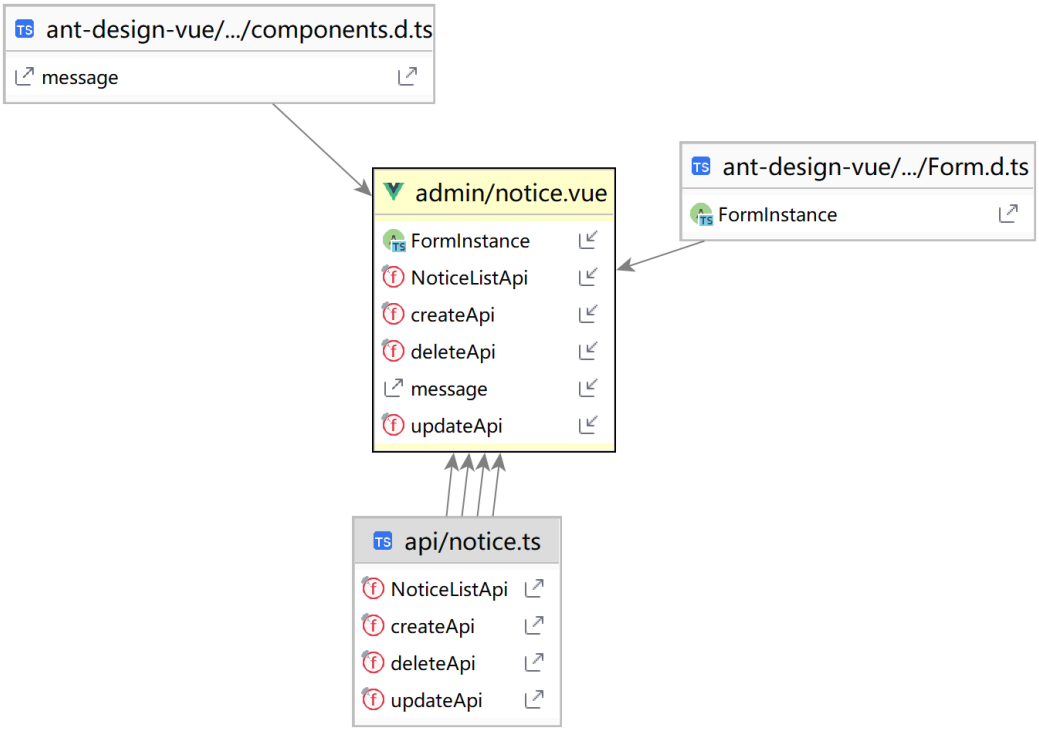
# 一、概述

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。

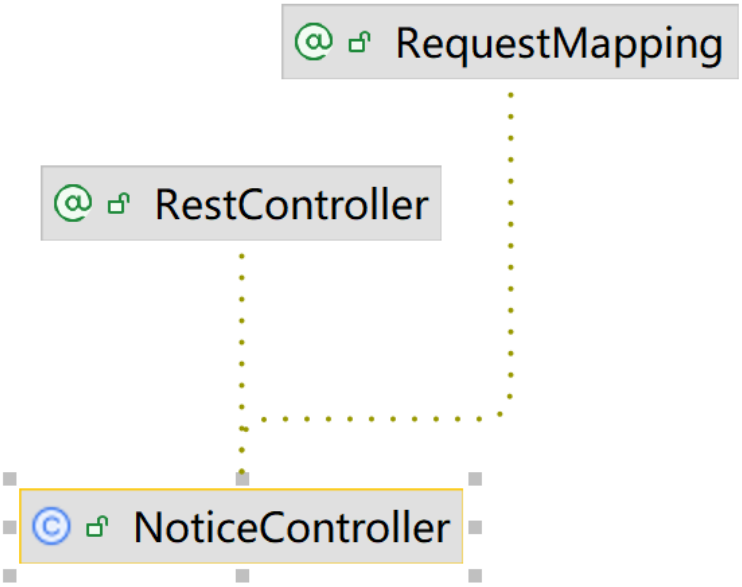
# 三、关系图




# 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.ResponeCode;
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 =
   LoggerFactory.getLogger(NoticeController.class);
25
26     @Autowired
27     NoticeService service;
28
29     @RequestMapping(value = "/list", method = RequestMethod.GET)
30     public APIResponse list() {
31         List<Notice> list = service.getNoticeList();
32         return new APIResponse(ResponeCode.SUCCESS, "查询成功",
   list);
33     }
34
35     @Access(level = AccessLevel.ADMIN)
36     @RequestMapping(value = "/create", method =
   RequestMethod.POST)
37     @Transactional
38     public APIResponse create(Notice notice) throws IOException {
39         service.createNotice(notice);
40         return new APIResponse(ResponeCode.SUCCESS, "创建成功");
41     }
42
43     @Access(level = AccessLevel.ADMIN)
```

```
    @RequestMapping(value = "/delete", method = RequestMethod.POST)
    public APIResponse delete(String ids) {
        System.out.println("ids===" + ids);
        // 批量删除
        String[] arr = ids.split(",");
        for (String id : arr) {
            service.deleteNotice(id);
        }
        return new APIResponse(ResponeCode.SUCCESS, "删除成功");
    }

    @Access(level = AccessLevel.ADMIN)
    @RequestMapping(value = "/update", method = RequestMethod.POST)
    @Transactional
    public APIResponse update(Notice notice) throws IOException {
        service.updateNotice(notice);
        return new APIResponse(ResponeCode.SUCCESS, "更新成功");
    }

}
```

4. 其他略

## （二）模板部分

```
<template>
  <div>
    <!--页面区域-->
    <div class="page-view">
      <div class="table-operations">
        <a-space>
          <a-button type="primary" @click="handleAdd">新增</a-button>
          <a-button @click="handleBatchDelete">批量删除</a-button>
        </a-space>
      </div>
      <a-table
        size="middle"
        rowKey="id"
        :loading="data.loading"
        :columns="columns"
        :data-source="data.noticeList"
        :scroll="{ x: 'max-content' }"
        :row-selection="rowSelection"
        :pagination="{
          size: 'default',
```

```
          current: data.page,
          pageSize: data.pageSize,
          onChange: (current) => (data.page = current),
          showSizeChanger: false,
          showTotal: (total) => 共${total}条数据,
        }"
      >
        <template #bodyCell="{ text, record, index, column }">
          <template v-if="column.key === 'operation'">
            <span>
              <a @click="handleEdit(record)">编辑</a>
              <a-divider type="vertical" />
              <a-popconfirm title="确定删除?" ok-text="是" cancel-
text="否" @confirm="confirmDelete(record)">
                <a href="#">删除</a>
              </a-popconfirm>
            </span>
          </template>
        </template>
      </a-table>
    </div>

    <!--弹窗区域-->
    <div>
      <a-modal
        :visible="modal.visile"
        :forceRender="true"
        :title="modal.title"
        ok-text="确认"
        cancel-text="取消"
        @cancel="handleCancel"
        @ok="handleOk"
      >
        <div>
          <a-form ref="myform" :label-col="{ style: { width:
'80px' } }" :model="modal.form" :rules="modal.rules">
            <a-row :gutter="24">
              <a-col span="24">
                <a-form-item label="标题" name="title">
                  <a-input placeholder="请输入标题" v-
model:value="modal.form.title"></a-input>
                </a-form-item>
              </a-col>
              <a-col span="24">
                <a-form-item label="通知内容" name="content">
                  <a-textarea placeholder="请输入内容" :rows="4"
v-model:value="modal.form.content"></a-textarea>
                </a-form-item>
```

```
            </a-col>
          </a-row>
        </a-form>
      </div>
    </a-modal>
  </div>
 </div>
</template>
```

## 五、脚本部分

```
<script setup lang="ts">
import { FormInstance, message } from 'ant-design-vue';
import { createApi, NoticeListApi, updateApi, deleteApi } from
'/@/api/notice';

const columns = reactive([
  {
    title: '序号',
    dataIndex: 'index',
    key: 'index',
    align: 'center'
  },
  {
    title: '标题',
    dataIndex: 'title',
    key: 'title',
    align: 'center'
  },
  {
    title: '内容',
    dataIndex: 'content',
    key: 'content',
    align: 'center',
    customRender: ({ text, record, index, column }) =>
text?.substring(0, 20) + '...',
  },
  {
    title: '操作',
    dataIndex: 'action',
    key: 'operation',
    align: 'center',
    fixed: 'right',
    width: 140,
  },
]);
```

```
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);
```

```
    });
};

const rowSelection = ref({
  onChange: (selectedRowKeys: (string | number)[], selectedRows:
DataItem[]) => {
    data.selectedRowKeys = selectedRowKeys;
  },
});

const handleAdd = () => {
  resetModal();
  modal.visile = true;
  modal.editFlag = false;
  modal.title = '新增';
  // 重置
  for (const key in modal.form) {
    modal.form[key] = undefined;
  }
};

const handleEdit = (record: any) => {
  resetModal();
  modal.visile = true;
  modal.editFlag = true;
  modal.title = '编辑';
  // 重置
  for (const key in modal.form) {
    modal.form[key] = undefined;
  }
  for (const key in record) {
    modal.form[key] = record[key];
  }
};

const confirmDelete = (record: any) => {
  deleteApi({ ids: record.id })
    .then((res) => {
      getDataList();
    })
    .catch((err) => {
      message.error(err.msg || '操作失败');
    });
};

const handleBatchDelete = () => {
  if (data.selectedRowKeys.length <= 0) {
    message.warn('请勾选删除项');
```

```
    return;
  }
  deleteApi({ ids: data.selectedRowKeys.join(',') })
    .then((res) => {
      message.success('删除成功');
      data.selectedRowKeys = [];
      getDataList();
    })
    .catch((err) => {
      message.error(err.msg || '操作失败');
    });
};

const handleOk = () => {
  myform.value
    ?.validate()
    .then(() => {
      if (modal.editFlag) {
        updateApi(modal.form)
          .then((res) => {
            hideModal();
            getDataList();
          })
          .catch((err) => {
            message.error(err.msg || '操作失败');
          });
      } else {
        createApi(modal.form)
          .then((res) => {
            hideModal();
            getDataList();
          })
          .catch((err) => {
            message.error(err.msg || '操作失败');
          });
      }
    })
    .catch((err) => {
      console.log('不能为空');
    });
};

const handleCancel = () => {
  hideModal();
};

// 恢复表单初始状态
const resetModal = () => {
```

```
  myform.value?.resetFields();
};

// 关闭弹窗
const hideModal = () => {
  modal.visile = false;
};
</script>
```

## 六、样式部分

```
<style scoped lang="less">
.page-view {
  min-height: 100%;
  background: #fff;
  padding: 24px;
  display: flex;
  flex-direction: column;
}

.table-operations {
  margin-bottom: 16px;
  text-align: right;
}

.table-operations > button {
  margin-right: 8px;
}
</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

```
  const getDataList = () => {
  data.loading = true;
  NoticeListApi({
    keyword: data.keyword,
  })
    .then((res) => {
      data.loading = false;
      res.data.forEach((item: any, index: any) => {
        item.index = index + 1;
      });
      data.noticeList = res.data;
      data.loading = false;
    })
    .catch((err) => {
      data.loading = false;
      console.log(err);
    });
};
```

## （二）新增通知

```
  const handleAdd = () => {
  resetModal();
  modal.visile = true;
  modal.editFlag = false;
  modal.title = '新增';
  // 重置
  for (const key in modal.form) {
    modal.form[key] = undefined;
  }
};
```

## （三）编辑通知

```
  const handleEdit = (record: any) => {
  resetModal();
  modal.visile = true;
  modal.editFlag = true;
  modal.title = '编辑';
  // 重置
  for (const key in modal.form) {
    modal.form[key] = undefined;
  }
  for (const key in record) {
    modal.form[key] = record[key];
  }
};
```

## （四）删除通知

```
  const confirmDelete = (record: any) => {
  deleteApi({ ids: record.id })
    .then((res) => {
      getDataList();
    })
    .catch((err) => {
      message.error(err.msg || '操作失败');
    });
};
```

## （五）批量删除

```
  const handleBatchDelete = () => {
  if (data.selectedRowKeys.length <= 0) {
    message.warn('请勾选删除项');
    return;
  }
  deleteApi({ ids: data.selectedRowKeys.join(',') })
    .then((res) => {
      message.success('删除成功');
      data.selectedRowKeys = [];
      getDataList();
    })
    .catch((err) => {
      message.error(err.msg || '操作失败');
    });
};
```

### （六）提交表单

```
const handleOk = () => {
myform.value
  ?.validate()
  .then(() => {
    if (modal.editFlag) {
      updateApi(modal.form)
        .then((res) => {
          hideModal();
          getDataList();
        })
        .catch((err) => {
          message.error(err.msg || '操作失败');
        });
    } else {
      createApi(modal.form)
        .then((res) => {
          hideModal();
          getDataList();
        })
        .catch((err) => {
          message.error(err.msg || '操作失败');
        });
    }
  })
  .catch((err) => {
    console.log('不能为空');
  });
};
```

## 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通知管理工具，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。