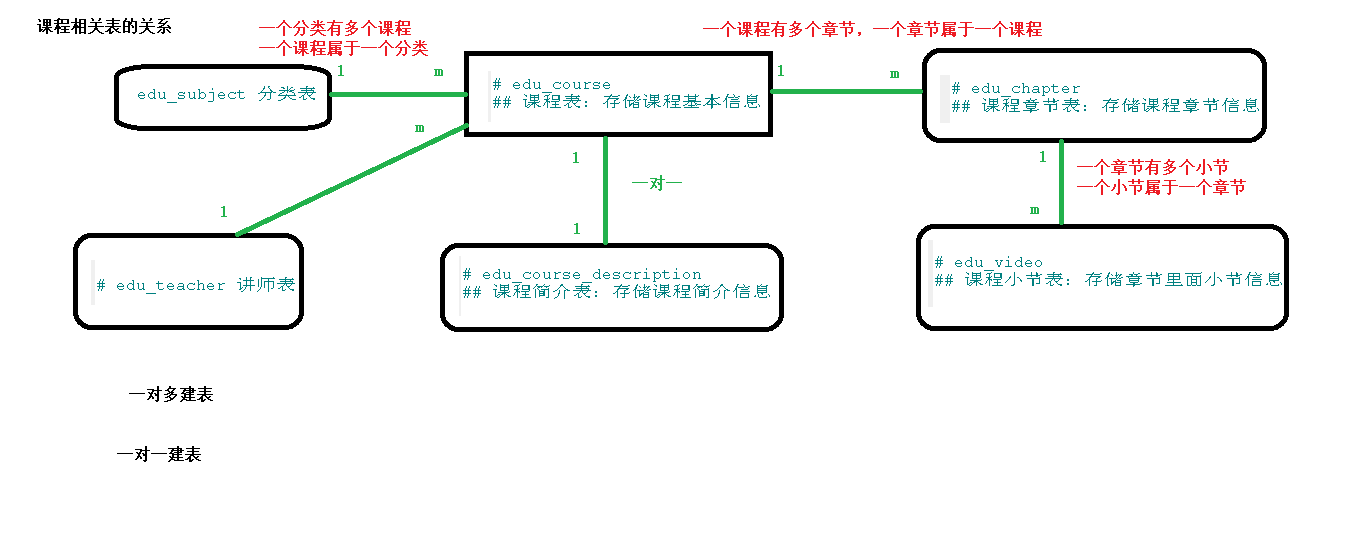


# 数据库



# 课程发布表单-步骤导航

## 一、需求







## 二、配置路由

### 1、添加路由

// 课程管理

   {

        path: '/edu/course',

        component: Layout,

        redirect: '/edu/course/list',

        name: 'Course',

        meta: { title: '课程管理', icon: 'form' },

        children: [

          {

            path: 'list',

            name: 'EduCourseList',

            component: () *=>* import('@/views/edu/course/list'),

            meta: { title: '课程列表' }

          },

          {

            path: 'info',

            name: 'EduCourseInfo',

            component: () *=>* import('@/views/edu/course/info'),

            meta: { title: '发布课程' }

          },

          {

            path: 'info/:id',

            name: 'EduCourseInfoEdit',

            component: () *=>* import('@/views/edu/course/info'),

            meta: { title: '编辑课程基本信息', noCache: true },

            hidden: true

          },

          {

            path: 'chapter/:id',

            name: 'EduCourseChapterEdit',

            component: () *=>* import('@/views/edu/course/chapter'),

            meta: { title: '编辑课程大纲', noCache: true },

            hidden: true

          },

          {

            path: 'publish/:id',

            name: 'EduCoursePublishEdit',

            component: () *=>* import('@/views/edu/course/publish'),

            meta: { title: '发布课程', noCache: true },

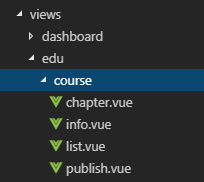
            hidden: true

          }

        ]

      },

### 2、添加vue组件



## 三、整合步骤条组件

参考 <http://element-cn.eleme.io/#/zh-CN/component/steps>

### 1、课程信息页面

info.vue

<template>

  <div class="app-container">

    <h2 style="text-align: center;">发布新课程</h2>

    <el-steps :active="1" process-status="wait" align-center style="margin-bottom: 40px;">

      <el-step title="填写课程基本信息"/>

      <el-step title="创建课程大纲"/>

      <el-step title="最终发布"/>

    </el-steps>

    <el-form label-width="120px">

      <el-form-item>

        <el-button :disabled="saveBtnDisabled" type="primary" @click="next">保存并下一步</el-button>

      </el-form-item>

    </el-form>

  </div>

</template>

<script>

export default {

  data() {

    return {

      saveBtnDisabled: false // 保存按钮是否禁用

    }

  },

  created() {

    console.log('info created')

  },

  methods: {

    next() {

      console.log('next')

      this.$router.push({ path: '/edu/course/chapter/1' })

    }

  }

}

</script>

### 2、课程大纲页面

chapter.vue

<template>

  <div class="app-container">

    <h2 style="text-align: center;">发布新课程</h2>

    <el-steps :active="2" process-status="wait" align-center style="margin-bottom: 40px;">

      <el-step title="填写课程基本信息"/>

      <el-step title="创建课程大纲"/>

      <el-step title="提交审核"/>

    </el-steps>

    <el-form label-width="120px">

      <el-form-item>

        <el-button @click="previous">上一步</el-button>

        <el-button :disabled="saveBtnDisabled" type="primary" @click="next">下一步</el-button>

      </el-form-item>

    </el-form>

  </div>

</template>

<script>

export default {

  data() {

    return {

      saveBtnDisabled: false // 保存按钮是否禁用

    }

  },

  created() {

    console.log('chapter created')

  },

  methods: {

    previous() {

      console.log('previous')

      this.$router.push({ path: '/edu/course/info/1' })

    },

    next() {

      console.log('next')

      this.$router.push({ path: '/edu/course/publish/1' })

    }

  }

}

</script>

### 3、课程发布页面

publish.vue

<template>

<div class="app-container">

<h2 style="text-align: center;">发布新课程</h2>

<el-steps :active="3" process-status="wait" align-center style="margin-bottom: 40px;">

<el-step title="填写课程基本信息"/>

<el-step title="创建课程大纲"/>

<el-step title="提交审核"/>

</el-steps>

<el-form label-width="120px">

<el-form-item>

<el-button @click="previous">返回修改</el-button>

<el-button :disabled="saveBtnDisabled" type="primary" @click="publish">发布课程</el-button>

</el-form-item>

</el-form>

</div>

</template>

<script>

export default {

data() {

return {

saveBtnDisabled: false // 保存按钮是否禁用

}

},

created() {

console.log('publish created')

},

methods: {

previous() {

console.log('previous')

this.$router.push({ path: '/edu/course/chapter/1' })

},

publish() {

console.log('publish')

this.$router.push({ path: '/edu/course/list' })

}

}

}

</script>

StepNo1添加课程-----------------

# 编辑课程基本信息

## 后台api



### 1、定义form表单对象

CourseInfoForm.java

@ApiModel*(*value = "课程基本信息", description = "编辑课程基本信息的表单对象"*)*

@Data  
public class CourseInfoForm implements Serializable *{* private static final long *serialVersionUID* = 1L;  
 @ApiModelProperty*(*value = "课程ID"*)* private String id;  
 @ApiModelProperty*(*value = "课程讲师ID"*)* private String teacherId;  
 @ApiModelProperty*(*value = "课程专业ID"*)* private String subjectId;  
 @ApiModelProperty*(*value = "课程标题"*)* private String title;  
 @ApiModelProperty*(*value = "课程销售价格，设置为0则可免费观看"*)* private BigDecimal price;  
 @ApiModelProperty*(*value = "总课时"*)* private Integer lessonNum;  
 @ApiModelProperty*(*value = "课程封面图片路径"*)* private String cover;  
 @ApiModelProperty*(*value = "课程简介"*)* private String description;  
*}*

### 2、修改CourseDescription主键生成策略

@ApiModelProperty*(*value = "课程ID"*)*

@TableId*(*value = "id", type = IdType.*INPUT)*private String id;

### 3、定义常量

实体类Course.Java中定义

public static final String COURSE\_DRAFT = "Draft";

public static final String COURSE\_NORMAL = "Normal";

### 4、定义控制层接口

CourseAdminController.java

package com.guli.edu.controller.admin;

@Api*(*description = "课程管理"*)*@CrossOrigin //跨域  
@RestController  
@RequestMapping*(*"/admin/edu/course"*)*public class CourseAdminController *{* @Autowired  
 private CourseService courseService;  
  
 @ApiOperation*(*value = "新增课程"*)* @PostMapping*(*"save-course-info"*)* public R saveCourseInfo*(* @ApiParam*(*name = "CourseInfoForm", value = "课程基本信息", required = true*)* @RequestBody CourseInfoForm courseInfoForm*) {* String courseId = courseService.saveCourseInfo*(*courseInfoForm*)*;  
 if *(*!StringUtils.isEmpty*(*courseId*)) {* return R.ok().data("courseId", courseId);  
 } else {  
 return R.error().message("保存失败");  
 }  
 }  
}

### 5、定义业务层方法

接口：CourseService.java

*/\*\**

*\* 保存课程和课程详情信息  
 \*  
 \** ***@param*** *courseInfoForm  
 \** ***@return*** *新生成的课程id  
 \*/*String saveCourseInfo(CourseInfoForm courseInfoForm);

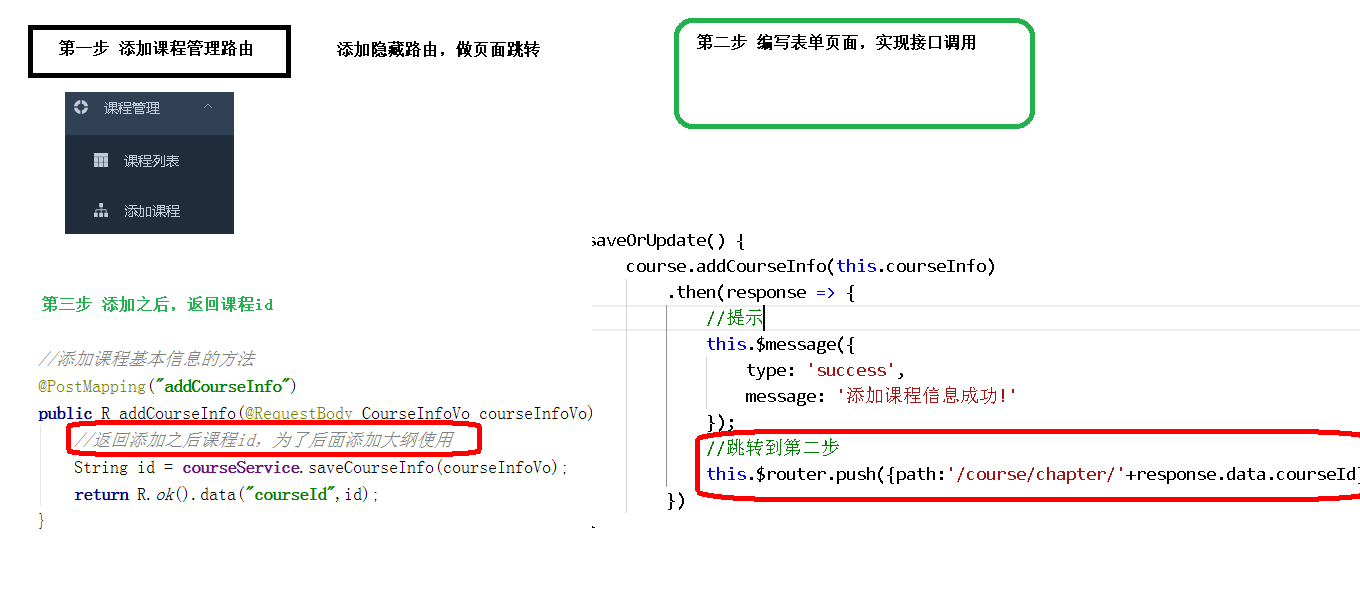
实现：CourseServiceImpl.java

@Autowired

private CourseDescriptionService courseDescriptionService;  
  
@Override  
public String saveCourseInfo*(*CourseInfoForm courseInfoForm*) {* //保存课程基本信息  
 Course course = new Course*()*;  
 course.setStatus*(*Course.COURSE\_DRAFT*)*;  
 BeanUtils.copyProperties*(*courseInfoForm, course);  
 boolean resultCourseInfo = this.save*(*course*)*;  
 if *(*!resultCourseInfo*) {* throw new GuliException*(*20001, "课程信息保存失败"*)*;  
 *}* //保存课程详情信息  
 CourseDescription courseDescription = new CourseDescription*()*;  
 courseDescription.setDescription*(*courseInfoForm.getDescription*())*;  
 courseDescription.setId*(*course.getId*())*;  
 boolean resultDescription = courseDescriptionService.save*(*courseDescription*)*;  
 if *(*!resultDescription*) {* throw new GuliException*(*20001, "课程详情信息保存失败"*)*;  
 *}* return course.getId*()*;  
}

### 6、Swagger测试

## 前端实现



### 1、定义api

import request from '@/utils/request'

*const* api\_name = '/admin/edu/course'

export default {

  saveCourseInfo(*courseInfo*) {

    return request({

      url: `${api\_name}/save-course-info`,

      method: 'post',

      data: *courseInfo*

    })

  }

}

### 2、组件模板

<el-form label-width="120px">

  <el-form-item label="课程标题">

    <el-input v-model="courseInfo.title" placeholder=" 示例：机器学习项目课：从基础到搭建项目视频课程。专业名称注意大小写"/>

  </el-form-item>

  <!-- 所属分类 TODO -->

  <!-- 课程讲师 TODO -->

  <el-form-item label="总课时">

    <el-input-number :min="0" v-model="courseInfo.lessonNum" controls-position="right" placeholder="请填写课程的总课时数"/>

  </el-form-item>

  <!-- 课程简介 TODO -->

  <!-- 课程封面 TODO -->

  <el-form-item label="课程价格">

    <el-input-number :min="0" v-model="courseInfo.price" controls-position="right" placeholder="免费课程请设置为0元"/> 元

  </el-form-item>

  <el-form-item>

    <el-button :disabled="saveBtnDisabled" type="primary" @click="next">保存并下一步</el-button>

  </el-form-item>

</el-form>

### 3、组件js

<script>

import course from '@/api/edu/course'

*const* defaultForm = {

  title: '',

  subjectId: '',

  teacherId: '',

  lessonNum: 0,

  description: '',

  cover: '',

  price: 0

}

export default {

  data() {

    return {

      courseInfo: defaultForm,

      saveBtnDisabled: false // 保存按钮是否禁用

    }

  },

  watch: {

    $route(*to*, *from*) {

      console.log('watch $route')

      this.init()

    }

  },

  created() {

    console.log('info created')

    this.init()

  },

  methods: {

    init() {

      if (this.$route.params && this.$route.params.id) {

*const* id = this.$route.params.id //获取路由参数

        console.log(id)

      } else {

        this.courseInfo = { ...defaultForm }

      }

    },

    next() {

      console.log('next')

      this.saveBtnDisabled = true

      if (!this.courseInfo.id) {

        this.saveData()

      } else {

        this.updateData()

      }

    },

    // 保存

    saveData() {

      course.saveCourseInfo(this.courseInfo).then(*response* *=>* {

        this.$message({

          type: 'success',

          message: '保存成功!'

        })

        return *response*// 将响应结果传递给then

      }).then(*response* *=>* {

        this.$router.push({ path: '/edu/course/chapter/' + *response*.data.courseId })

      }).catch((*response*) *=>* {

        this.$message({

          type: 'error',

          message: *response*.message

        })

      })

    },

    updateData() {

      this.$router.push({ path: '/edu/course/chapter/1' })

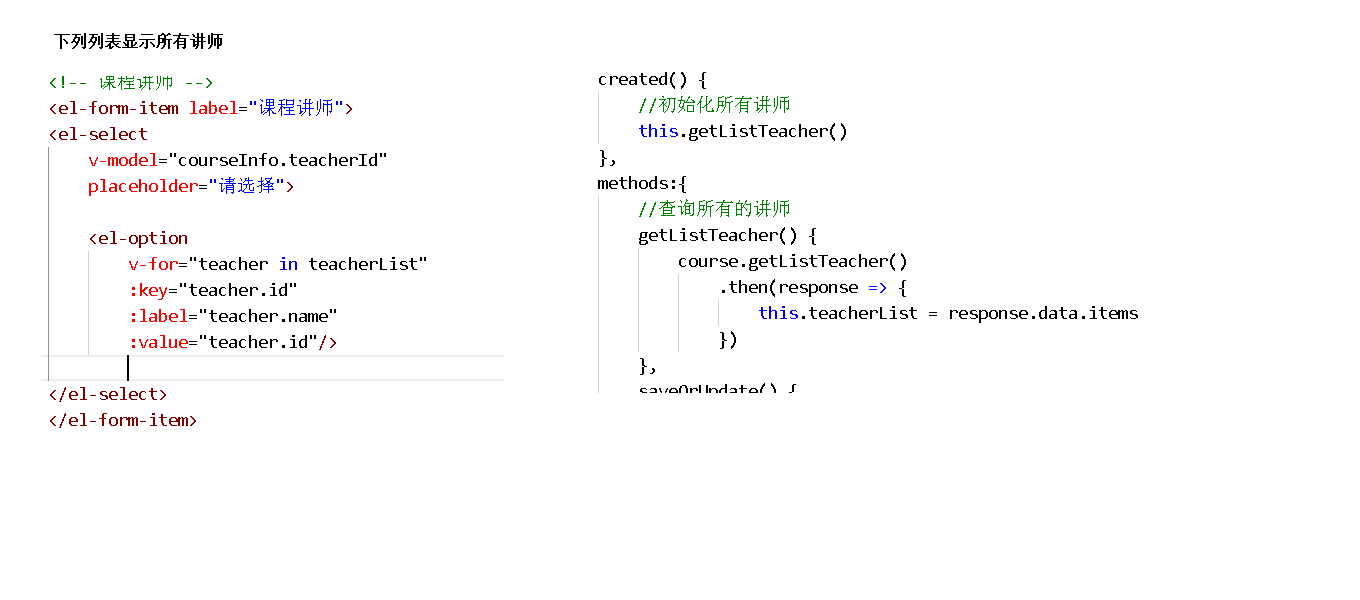
    }

  }

}

</script>

# 讲师下拉列表



## 一、前端实现

### 1、组件模板

<!-- 课程讲师 -->

<el-form-item label="课程讲师">

  <el-select

    v-model="courseInfo.teacherId"

    placeholder="请选择">

    <el-option

      v-for="teacher in teacherList"

      :key="teacher.id"

      :label="teacher.name"

      :value="teacher.id"/>

  </el-select>

</el-form-item>

### 2、定义api

api/edu/teacher.js

    /\*\*

     \* 查询所有讲师

     \* *@returns*

     \*/

    getAll(){

        return request({

            url: `/eduservice/teacher/findAll`,

            method: 'get'

        })

    }

组件中引入teacher api

import teacher from '@/api/edu/teacher'

### 3、组件脚本

定义data

teacherList: [] // 讲师列表

表单初始化时获取讲师列表

init() {

......

// 获取讲师列表

this.initTeacherList()

},

initTeacherList() {

teacher.getList().then(response => {

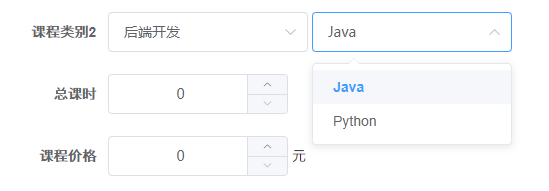
this.teacherList = response.data.items

})

},

# 课程分类多级联动的实现

## 一、需求



## 二、获取一级分类

### 1、组件数据定义

定义在data中

subjectNestedList: [],//一级分类列表

subSubjectList: []//二级分类列表

### 2、组件模板

<!-- 一级分类 -->

<el-form-item label="课程类别">

    <el-select

      v-model="courseInfo.subjectParentId"

      placeholder="请选择">

      <el-option

        v-for="subject in subjectNestedList"

        :key="subject.id"

        :label="subject.title"

        :value="subject.id"/>

    </el-select>

  </el-form-item>

### 3、组件脚本

表单初始化时获取一级分类嵌套列表，引入subject api

import subject from '@/api/edu/subject'

定义方法

init() {

......

// 初始化分类列表

this.initSubjectList()

},

initSubjectList() {

subject.getNestedTreeList().then(response => {

this.subjectNestedList = response.data.items

})

},

## 三、级联显示二级分类

### 1、组件模板

      <!-- 所属分类：级联下拉列表 -->

<!-- 一级分类 -->

<el-form-item label="课程类别">

    <el-select

      v-model="courseInfo.subjectParentId"

      placeholder="请选择">

      <el-option

        v-for="subject in subjectNestedList"

        :key="subject.id"

        :label="subject.title"

        :value="subject.id"/>

    </el-select>

    <!-- 二级分类 -->

    <el-select v-model="courseInfo.subjectId" placeholder="请选择">

        <el-option

        v-for="subject in subSubjectList"

        :key="subject.value"

        :label="subject.title"

        :value="subject.id"/>

    </el-select>

  </el-form-item>

### 2、注册change事件

在一级分类的<el-select>组件中注册change事件

<el-select @change="subjectLevelOneChanged" ......

### 3、定义change事件方法

      subjectLevelOneChanged(*value*) {

        console.log(*value*)

        for (*let* i = 0; i < this.subjectNestedList.length; i++) {

          if (this.subjectNestedList[i].id === *value*) {

            this.subSubjectList = this.subjectNestedList[i].children

            this.courseInfo.subjectId = ''

          }

        }

      },

# 课程封面

## 一、整合上传组件

参考 <http://element-cn.eleme.io/#/zh-CN/component/upload> 用户头像上传

### 1、上传默认封面

创建文件夹cover，上传默认的课程封面



### 2、定义默认封面

 不指定默认封面界面不显示

const defaultForm = {

......,

cover: process.env.OSS\_PATH + '/cover/default.gif',

......

}

或者static目录下放默认图片

cover: "/static/default\_course\_avoter.jpg" //课程封面

### 3、定义data数据

BASE\_API: process.env.BASE\_API // 接口API地址

### 4、组件模板

在info.vue中添加上传组件模板

<!-- 课程封面-->

<el-form-item label="课程封面">

<el-upload

:show-file-list="false"

:on-success="handleAvatarSuccess"

:before-upload="beforeAvatarUpload"

:action="BASE\_API+'/admin/oss/file/upload?host=cover'"

class="avatar-uploader">

<img :src="courseInfo.cover">

</el-upload>

</el-form-item>

### 5、结果回调

      handleAvatarSuccess(*res*, *file*) {

        console.log(*res*) // 上传响应

        console.log(URL.createObjectURL(*file*.raw)) // base64编码

        this.courseInfo.cover = *res*.data.url

      },

      beforeAvatarUpload(*file*) {

*const* isJPG = *file*.type === 'image/jpeg'

*const* isLt2M = *file*.size / 1024 / 1024 < 2

        if (!isJPG) {

          this.$message.error('上传头像图片只能是 JPG 格式!')

        }

        if (!isLt2M) {

          this.$message.error('上传头像图片大小不能超过 2MB!')

        }

        return isJPG && isLt2M

      },

## 二、修改后端api

### 1、修改上传controller

添加host可选参数

/\*\*

\* 文件上传

\*

\* @param file

\*/

@ApiOperation(value = "文件上传")

@PostMapping("upload")

public R upload(

@ApiParam(name = "file", value = "文件", required = true)

@RequestParam("file") MultipartFile file,

@ApiParam(name = "host", value = "文件上传路径", required = false)) {

String uploadUrl = fileService.upload(file);

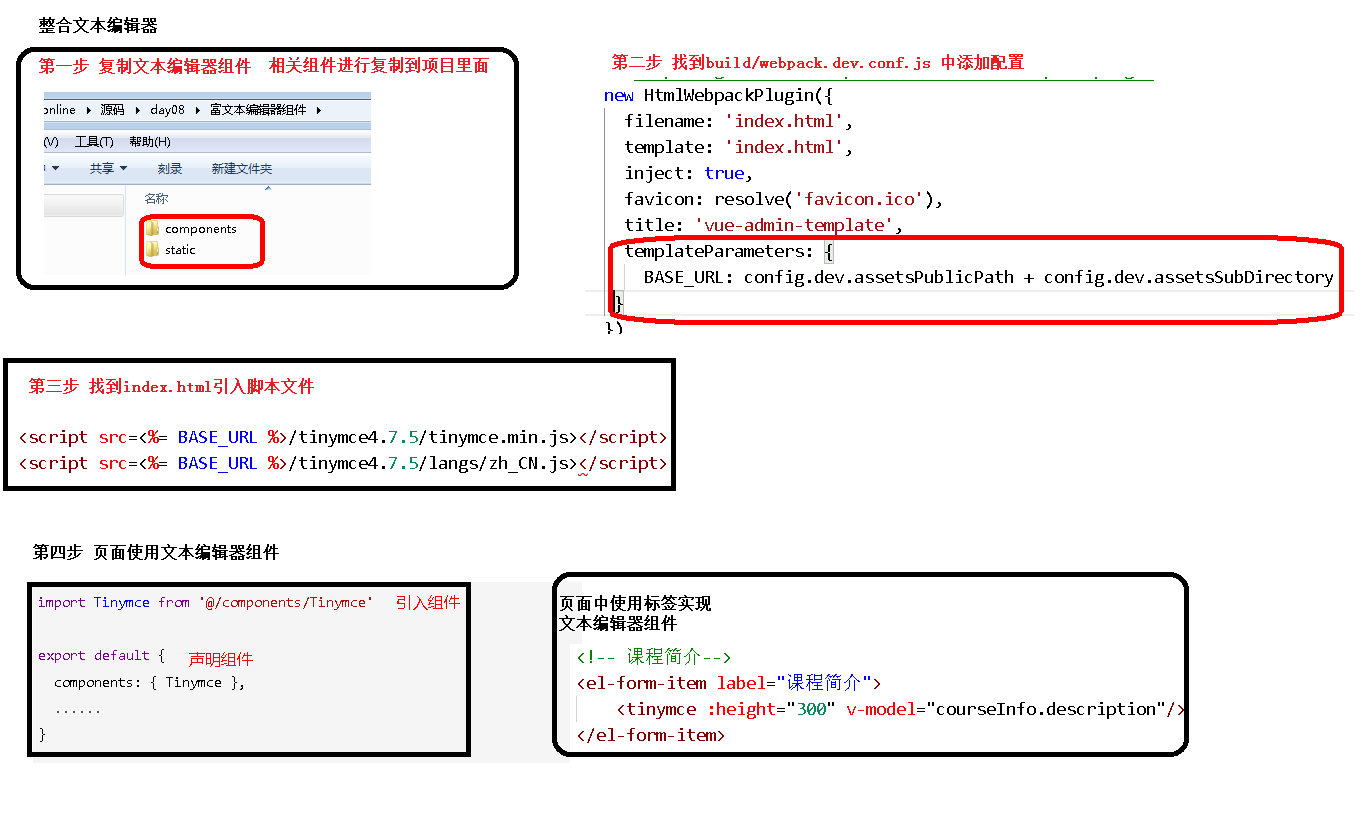
//返回r对象

return R.ok().message("文件上传成功").data("url", uploadUrl);

}

### 2、综合测试

# 1-5 富文本编辑器Tinymce



## 一、Tinymce可视化编辑器

参考

<https://panjiachen.gitee.io/vue-element-admin/#/components/tinymce>

<https://panjiachen.gitee.io/vue-element-admin/#/example/create>

## 二、组件初始化

Tinymce是一个传统javascript插件，默认不能用于Vue.js因此需要做一些特殊的整合步骤

### 1、复制脚本库

将脚本库复制到项目的static目录下（在vue-element-admin-master的static路径下）

### 2、配置html变量

在 guli-admin/build/webpack.dev.conf.js 中添加配置

使在html页面中可是使用这里定义的BASE\_URL变量

new HtmlWebpackPlugin({

......,

templateParameters: {

BASE\_URL: config.dev.assetsPublicPath + config.dev.assetsSubDirectory

}

})

### 3、引入js脚本

在guli-admin/index.html 中引入js脚本

    <script src=<%= BASE\_URL %>/tinymce4.7.5/tinymce.min.js></script>

    <script src=<%= BASE\_URL %>/tinymce4.7.5/langs/zh\_CN.js></script>

## 三、组件引入

为了让Tinymce能用于Vue.js项目，vue-element-admin-master对Tinymce进行了封装，下面我们将它引入到我们的课程信息页面

### 1、复制组件

src/components/Tinymce

### 2、引入组件

课程信息组件中引入 Tinymce

import Tinymce from '@/components/Tinymce'

export default {

    components: { Tinymce },

    ……

  }

### 3、组件模板

      <!-- 课程简介 TODO -->

      <el-form-item label="课程简介">

        <tinymce :height="300" v-model="courseInfo.description"/>

    </el-form-item>

### 4、组件样式

在info.vue文件的最后添加如下代码，调整上传图片按钮的高度

<style scoped>

    .tinymce-container {

*line-height*: 29px;

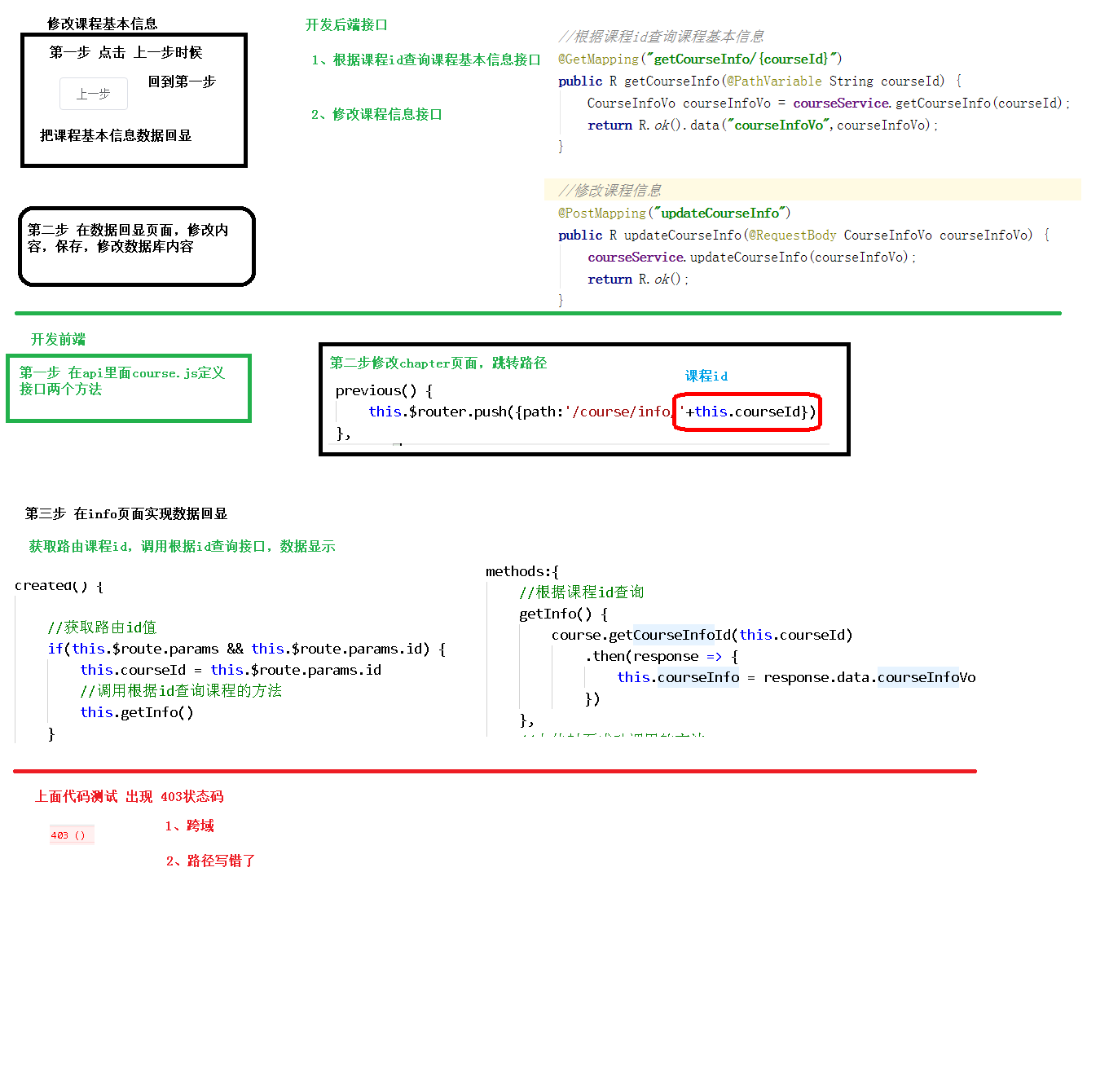
    }

</style>

### 5、图片的base64编码

Tinymce中的图片上传功能直接存储的是图片的base64编码，因此无需图片服务器

# 课程信息回显



## 一、后端实现

### 1、业务层

接口：CourseService.java

CourseInfoForm getCourseInfoFormById(String id);

实现：CourseServiceImpl.java

@Override

public CourseInfoForm getCourseInfoFormById(String id) {

Course course = this.getById(id);

if(course == null){

throw new GuliException(20001, "数据不存在");

}

CourseInfoForm courseInfoForm = new CourseInfoForm();

BeanUtils.copyProperties(course, courseInfoForm);

CourseDescription courseDescription = courseDescriptionService.getById(id);

if(course != null){

courseInfoForm.setDescription(courseDescription.getDescription());

}

return courseInfoForm;

}

### 2、web层

@ApiOperation(value = "根据ID查询课程")

@GetMapping("course-info/{id}")

public R getById(

@ApiParam(name = "id", value = "课程ID", required = true)

@PathVariable String id){

CourseInfoForm courseInfoForm = courseService.getCourseInfoFormById(id);

return R.ok().data("item", courseInfoForm);

}

### 3、Swagger中测试

## 二、前端实现

### 1、定义api

api/edu/course.js

getCourseInfoById(id) {

return request({

url: `${api\_name}/course-info/${id}`,

method: 'get'

})

}

### 2、组件js

init() {

if (this.$route.params && this.$route.params.id) {

const id = this.$route.params.id

//根据id获取课程基本信息

this.fetchCourseInfoById(id)

}

......

},

fetchCourseInfoById(id) {

course.getCourseInfoById(id).then(response => {

this.courseInfo = response.data.item

}).catch((response) => {

this.$message({

type: 'error',

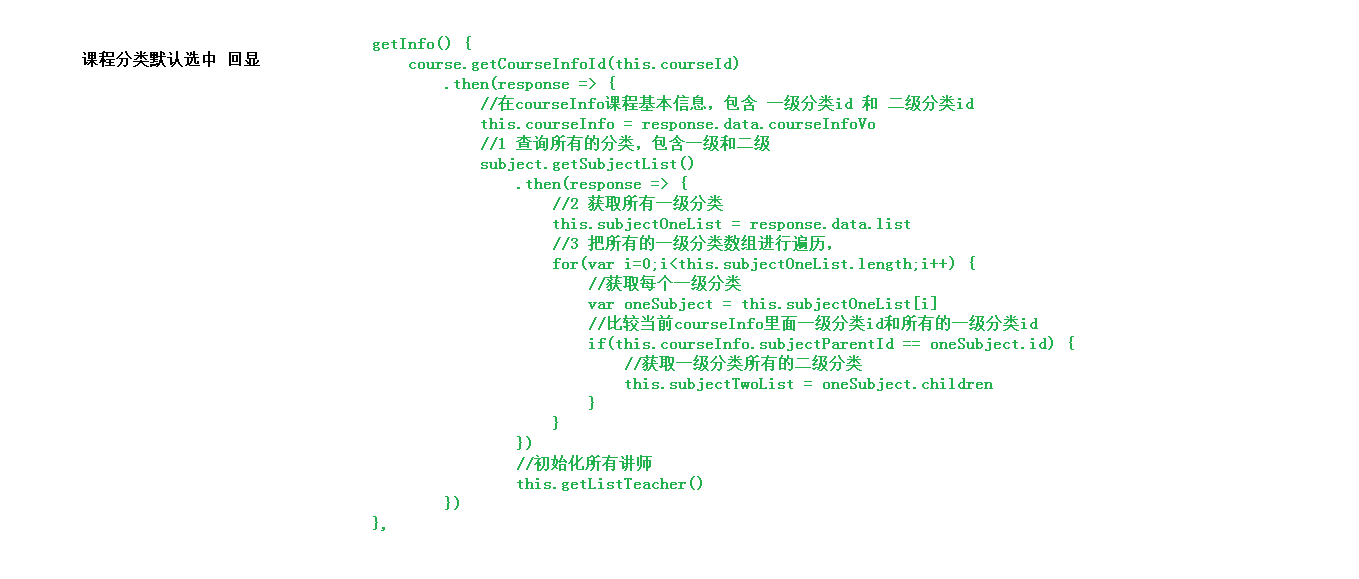
message: response.message

})

})

},

## 解决级联下拉菜单回显问题（初始化数据时获取二级联动数据）



### 1、数据库中增加冗余列

subject\_parent\_id 课程专业父级ID

### 2、pojo中增加属性

entity.Course.java

form.CourseInfo.java

@ApiModelProperty(value = "课程专业父级ID")

private String subjectParentId;

### 3、vue组件中绑定数据

edu/course/infoinfo.vue

<el-select v-model="courseInfo.subjectParentId" ......

### 4、修改init方法

将 this.initSubjectList() 和 this.initTeacherList()移至else

init() {

    if (this.$route.params && this.$route.params.id) {

*const* id = this.$route.params.id

        // 根据id获取课程基本信息

        this.fetchCourseInfoById(id)

    } else {

        this.courseInfo = { ...defaultForm }

        // 初始化分类列表

        this.initSubjectList()

        // 获取讲师列表

        this.initTeacherList()

    }

},

### 5、\*\*修改fetchCourseInfoById方法

在获取回显信息的回调里面 调用接口初始化二级分类数据，

fetchCourseInfoById(*id*) {

    course.getCourseInfoById(*id*).then((*responseCourse*) *=>* {

        this.courseInfo = *responseCourse*.data.item;

        // 初始化分类列表

        subject.getNestedTreeList().then((*responseSubject*) *=>* {

            this.subjectNestedList = *responseSubject*.data.items;

            for (*let* i = 0; i < this.subjectNestedList.length; i++) {

                if (this.subjectNestedList[i].id === this.courseInfo.subjectParentId) {

                    this.subSubjectList = this.subjectNestedList[i].children;

                }

            }

        });

        // 获取讲师列表

        this.initTeacherList();

    }).catch((*response*) *=>* {

        this.$message({

            type: "error",

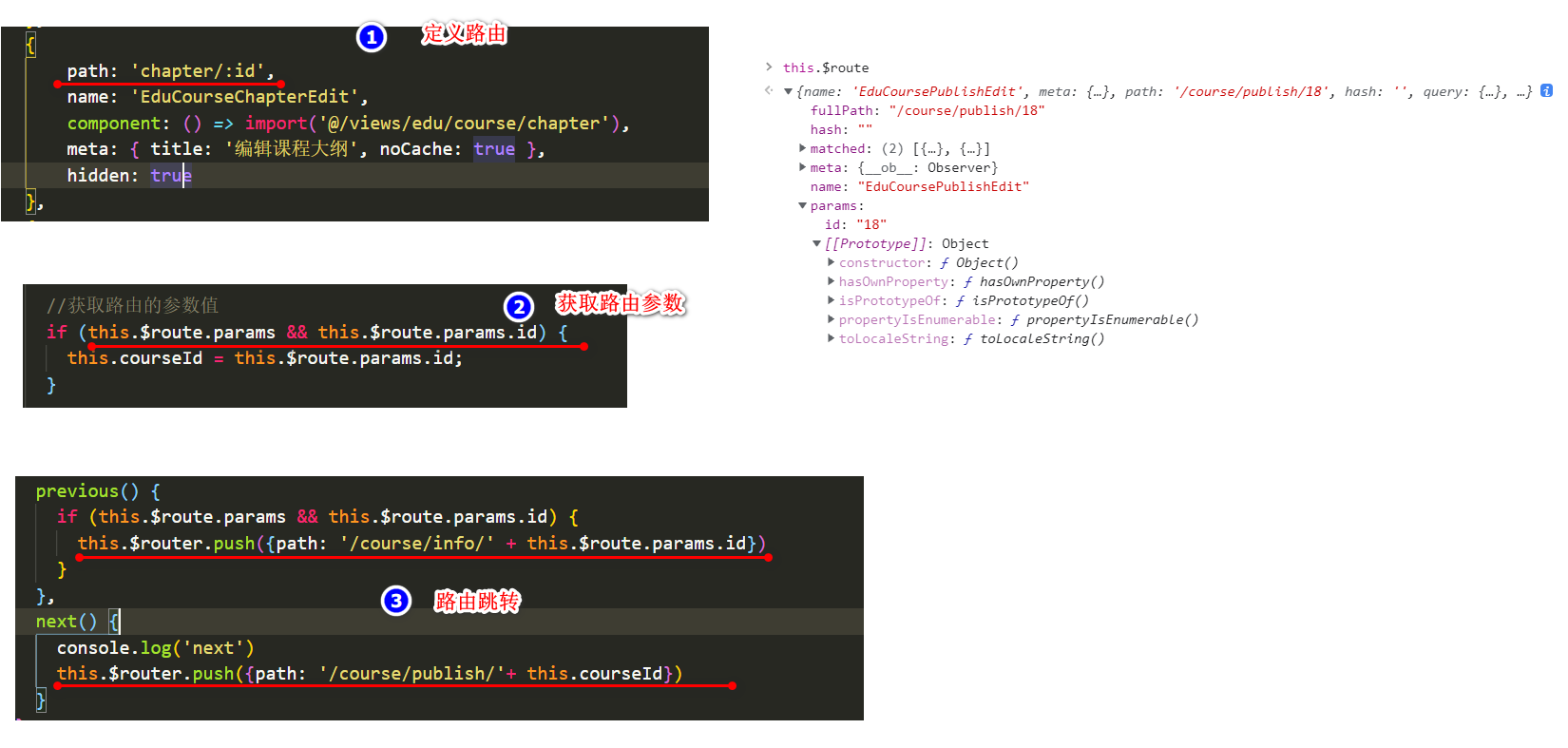
            message: *response*.message,

        });

    });

},

### 6、路由常用总结



# 1-7 更新课程信息

## 一、后端实现

### 1、业务层

接口：CourseService.java

void updateCourseInfoById(CourseInfoForm courseInfoForm);

实现：CourseServiceImpl.java

@Override

public void updateCourseInfoById(CourseInfoForm courseInfoForm) {

//保存课程基本信息

Course course = new Course();

BeanUtils.copyProperties(courseInfoForm, course);

boolean resultCourseInfo = this.updateById(course);

if(!resultCourseInfo){

throw new GuliException(20001, "课程信息保存失败");

}

//保存课程详情信息

CourseDescription courseDescription = new CourseDescription();

courseDescription.setDescription(courseInfoForm.getDescription());

courseDescription.setId(course.getId());

boolean resultDescription = courseDescriptionService.updateById(courseDescription);

if(!resultDescription){

throw new GuliException(20001, "课程详情信息保存失败");

}

}

### 2、web层

@ApiOperation(value = "更新课程")

@PutMapping("update-course-info/{id}")

public R updateCourseInfoById(

@ApiParam(name = "CourseInfoForm", value = "课程基本信息", required = true)

@RequestBody CourseInfoForm courseInfoForm,

@ApiParam(name = "id", value = "课程ID", required = true)

@PathVariable String id){

courseService.updateCourseInfoById(courseInfoForm);

return R.ok();

}

## 二、前端实现

### 1、定义api

course.js

updateCourseInfoById(courseInfo) {

return request({

url: `${api\_name}/update-course-info/${courseInfo.id}`,

method: 'put',

data: courseInfo

})

}

### 2、组件js

info.vue

updateData() {

this.saveBtnDisabled = true

course.updateCourseInfoById(this.courseInfo).then(response => {

this.$message({

type: 'success',

message: '修改成功!'

})

return response// 将响应结果传递给then

}).then(response => {

this.$router.push({ path: '/edu/course/chapter/' + response.data.courseId })

}).catch((response) => {

// console.log(response)

this.$message({

type: 'error',

message: '保存失败'

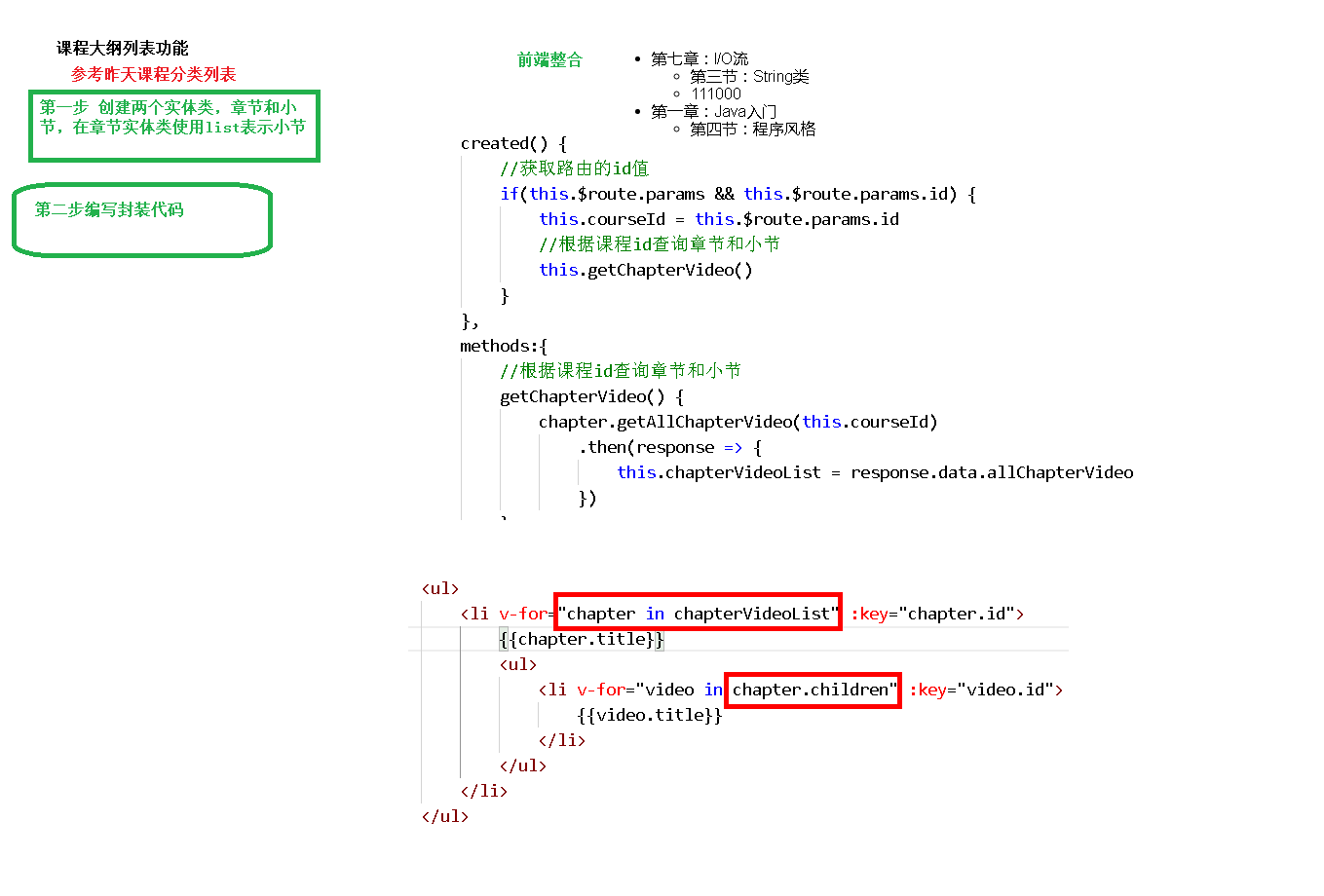
})

})

},

StepNo2课程大纲开发-------------------

# 2-1 课程大纲列表显示



## 一、后端实现

### 1、定义vo

ChapterVo

@ApiModel*(*value = "章节信息"*)*

@Data  
public class ChapterVo implements Serializable *{* private static final long *serialVersionUID* = 1L;  
 private String id;  
 private String title;  
 private List*<*VideoVo*>* children = new ArrayList*<>()*;  
*}*

VideoVo

@ApiModel*(*value = "课时信息"*)*

@Data  
public class VideoVo implements Serializable *{* private static final long *serialVersionUID* = 1L;  
 private String id;  
 private String title;  
 private Boolean free;  
*}*

### 2、服务层

接口

package com.guli.edu.service;

public interface ChapterService extends IService<Chapter> {

List<ChapterVo> nestedList(String courseId);

}

实现

package com.guli.edu.service.impl;

@Service

public class ChapterServiceImpl extends ServiceImpl<ChapterMapper, Chapter> implements ChapterService {

@Autowired

private VideoService videoService;

@Override

public List<ChapterVo> nestedList(String courseId) {

//最终要的到的数据列表

ArrayList<ChapterVo> chapterVoArrayList = new ArrayList<>();

//获取章节信息

QueryWrapper<Chapter> queryWrapper1 = new QueryWrapper<>();

queryWrapper1.eq("course\_id", courseId);

queryWrapper1.orderByAsc("sort", "id");

List<Chapter> chapters = baseMapper.selectList(queryWrapper1);

//获取课时信息

QueryWrapper<Video> queryWrapper2 = new QueryWrapper<>();

queryWrapper2.eq("course\_id", courseId);

queryWrapper2.orderByAsc("sort", "id");

List<Video> videos = videoService.list(queryWrapper2);

//填充章节vo数据

int count1 = chapters.size();

for (int i = 0; i < count1; i++) {

Chapter chapter = chapters.get(i);

//创建章节vo对象

ChapterVo chapterVo = new ChapterVo();

BeanUtils.copyProperties(chapter, chapterVo);

chapterVoArrayList.add(chapterVo);

//填充课时vo数据

ArrayList<VideoVo> videoVoArrayList = new ArrayList<>();

int count2 = videos.size();

for (int j = 0; j < count2; j++) {

Video video = videos.get(j);

if(chapter.getId().equals(video.getChapterId())){

//创建课时vo对象

VideoVo videoVo = new VideoVo();

BeanUtils.copyProperties(video, videoVo);

videoVoArrayList.add(videoVo);

}

}

chapterVo.setChildren(videoVoArrayList);

}

return chapterVoArrayList;

}

}

### 3、web层

package com.guli.edu.controller.admin;

@Api(description="课程章节管理")

@CrossOrigin //跨域

@RestController

@RequestMapping("/admin/edu/chapter")

public class ChapterAdminController {

@Autowired

private ChapterService chapterService;

@ApiOperation(value = "嵌套章节数据列表")

@GetMapping("nested-list/{courseId}")

public R nestedListByCourseId(

@ApiParam(name = "courseId", value = "课程ID", required = true)

@PathVariable String courseId){

List<ChapterVo> chapterVoList = chapterService.nestedList(courseId);

return R.ok().data("items", chapterVoList);

}

}

### 4、Swagger测试

## 二、前端实现

### 1、定义api

chapter.js

import request from '@/utils/request'

const api\_name = '/admin/edu/chapter'

export default {

getNestedTreeList(courseId) {

return request({

url: `${api\_name}/nested-list/${courseId}`,

method: 'get'

})

}

}

### 2、定义组件脚本

定义data

courseId: '', // 所属课程

chapterNestedList: [] // 章节嵌套课时列表

created中调用init方法

created() {

console.log('chapter created')

this.init()

},

定义相关methods获取章节和课时列表

init() {

if (this.$route.params && this.$route.params.id) {

this.courseId = this.$route.params.id

// 根据id获取课程基本信息

this.fetchChapterNestedListByCourseId()

}

},

fetchChapterNestedListByCourseId() {

chapter.getNestedTreeList(this.courseId).then(response => {

this.chapterNestedList = response.data.items

})

},

### 3、定义组件模板

    <el-button type="text">添加章节</el-button>

    <!-- 章节 -->

    <ul class="chanpterList">

        <li

            v-for="chapter in chapterNestedList"

            :key="chapter.id">

            <p>

                {{ chapter.title }}

                <span class="acts">

                    <el-button type="text">添加课时</el-button>

                    <el-button style="" type="text">编辑</el-button>

                    <el-button type="text">删除</el-button>

                </span>

            </p>

            <!-- 视频 -->

            <ul class="chanpterList videoList">

                <li

                    v-for="video in chapter.children"

                    :key="video.id">

                    <p>{{ video.title }}

                        <span class="acts">

                            <el-button type="text">编辑</el-button>

                            <el-button type="text">删除</el-button>

                        </span>

                    </p>

                </li>

            </ul>

        </li>

    </ul>

    <div>

        <el-button @click="previous">上一步</el-button>

        <el-button :disabled="saveBtnDisabled" type="primary" @click="next">下一步</el-button>

    </div>

### 4、定义样式

将样式的定义放在页面的最后

scope表示这里定义的样式只在当前页面范围内生效，不会污染到其他的页面

<style scoped>

  .chanpterList {

*position*: relative;

*list-style*: none;

*margin*: 0;

*padding*: 0;

  }

  .chanpterList li {

*position*: relative;

  }

  .chanpterList p {

*float*: left;

*font-size*: 20px;

*margin*: 10px 0;

*padding*: 10px;

*height*: 70px;

*line-height*: 50px;

*width*: 100%;

*border*: 1px solid #DDD;

  }

  .chanpterList .acts {

*float*: right;

*font-size*: 14px;

  }

  .videoList {

*padding-left*: 50px;

  }

  .videoList p {

*float*: left;

*font-size*: 14px;

*margin*: 10px 0;

*padding*: 10px;

*height*: 50px;

*line-height*: 30px;

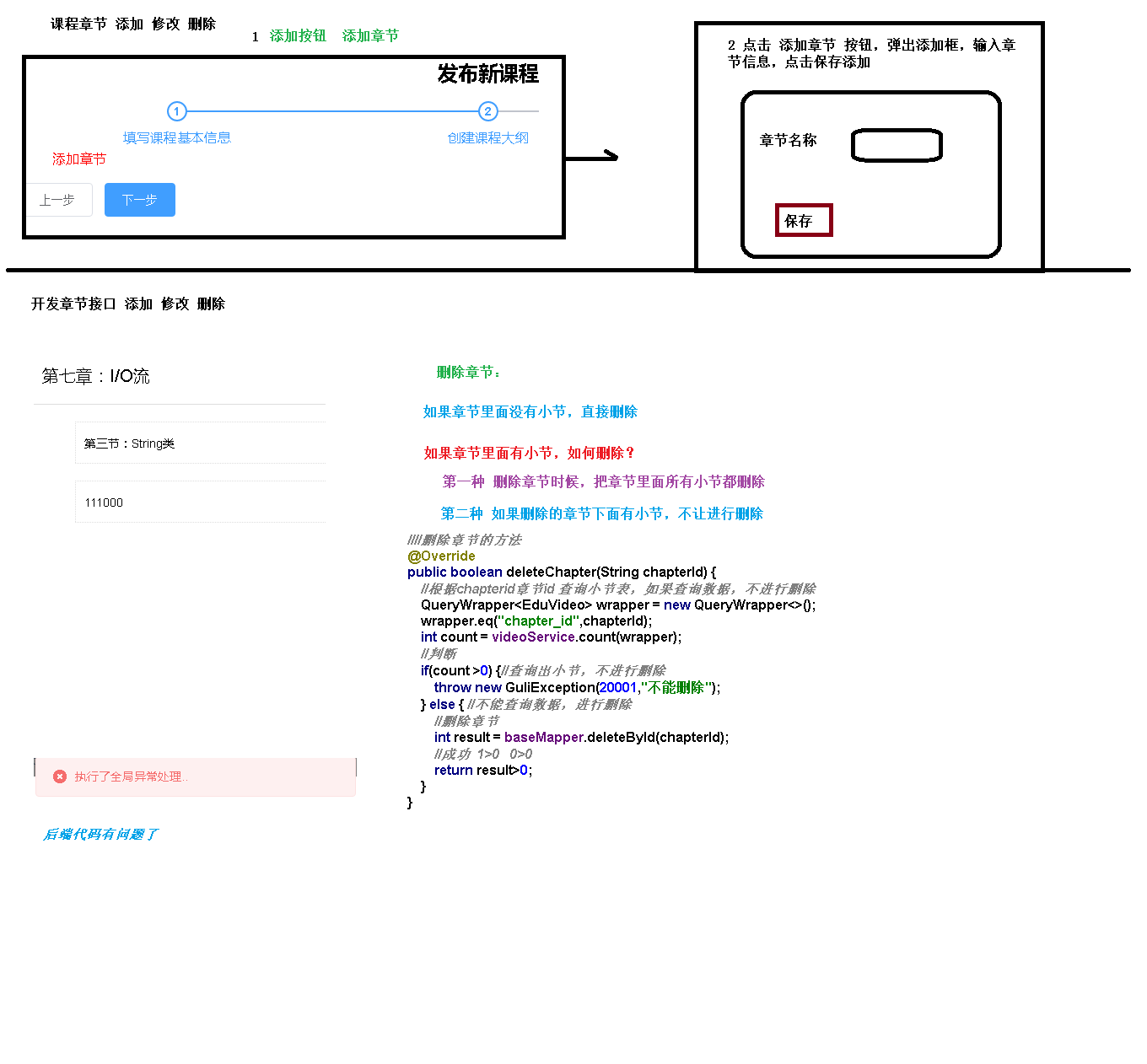
*width*: 100%;

*border*: 1px dotted #DDD;

  }

</style>

# 2-2 章节管理后端接口开发



## 一、新增章节

web层

@ApiOperation(value = "新增章节")

@PostMapping

public R save(

@ApiParam(name = "chapterVo", value = "章节对象", required = true)

@RequestBody Chapter chapter){

chapterService.save(chapter);

return R.ok();

}

## 二、根据id查询

web层

@ApiOperation(value = "根据ID查询章节")

@GetMapping("{id}")

public R getById(

@ApiParam(name = "id", value = "章节ID", required = true)

@PathVariable String id){

Chapter chapter = chapterService.getById(id);

return R.ok().data("item", chapter);

}

## 三、更新

web层

@ApiOperation(value = "根据ID修改章节")

@PutMapping("{id}")

public R updateById(

@ApiParam(name = "id", value = "章节ID", required = true)

@PathVariable String id,

@ApiParam(name = "chapter", value = "章节对象", required = true)

@RequestBody Chapter chapter){

chapter.setId(id);

chapterService.updateById(chapter);

return R.ok();

}

## 四、删除

### 1、web层

@ApiOperation(value = "根据ID删除章节")

@DeleteMapping("{id}")

public R removeById(

@ApiParam(name = "id", value = "章节ID", required = true)

@PathVariable String id){

boolean result = chapterService.removeChapterById(id);

if(result){

return R.ok();

}else{

return R.error().message("删除失败");

}

}

### 2、Service

ChapterService层：接口

boolean removeChapterById(String id);

ChapterService层：实现

@Override

public boolean removeChapterById(String id) {

//根据id查询是否存在视频，如果有则提示用户尚有子节点

if(videoService.getCountByChapterId(id)){

throw new GuliException(20001,"该分章节下存在视频课程，请先删除视频课程");

}

Integer result = baseMapper.deleteById(id);

return null != result && result > 0;

}

VideoService：接口

boolean getCountByChapterId(String chapterId);

VideoService：实现

@Override

public boolean getCountByChapterId(String chapterId) {

QueryWrapper<Video> queryWrapper = new QueryWrapper<>();

queryWrapper.eq("chapter\_id", chapterId);

Integer count = baseMapper.selectCount(queryWrapper);

return null != count && count > 0;

}

## 五、Swagger测试

# 2-3 章节管理前端页面实现

## 一、定义api

removeById(id) {

return request({

url: `${api\_name}/${id}`,

method: 'delete'

})

},

save(chapter) {

return request({

url: api\_name,

method: 'post',

data: chapter

})

},

getById(id) {

return request({

url: `${api\_name}/${id}`,

method: 'get'

})

},

updateById(chapter) {

return request({

url: `${api\_name}/${chapter.id}`,

method: 'put',

data: chapter

})

}

## 二、新增章节页面功能

### 1、定义data数据

dialogChapterFormVisible: false, //是否显示章节表单

chapter: {// 章节对象

title: '',

sort: 0

}

### 2、添加章节按钮

<el-button type="text" @click="dialogChapterFormVisible = true">添加章节</el-button>

### 3、章节表单dialog

<!-- 添加和修改章节表单 -->

<el-dialog :visible.sync="dialogChapterFormVisible" title="添加章节">

<el-form :model="chapter" label-width="120px">

<el-form-item label="章节标题">

<el-input v-model="chapter.title"/>

</el-form-item>

<el-form-item label="章节排序">

<el-input-number v-model="chapter.sort" :min="0" controls-position="right"/>

</el-form-item>

</el-form>

<div slot="footer" class="dialog-footer">

<el-button @click="dialogChapterFormVisible = false">取 消</el-button>

<el-button type="primary" @click="saveOrUpdate">确 定</el-button>

</div>

</el-dialog>

### 4、添加章节methods

saveOrUpdate() {

this.saveBtnDisabled = true

if (!this.chapter.id) {

this.saveData()

} else {

this.updateData()

}

},

saveData() {

this.chapter.courseId = this.courseId

chapter.save(this.chapter).then(response => {

this.$message({

type: 'success',

message: '保存成功!'

})

this.helpSave()

}).catch((response) => {

this.$message({

type: 'error',

message: response.message

})

})

},

updateData() {

},

helpSave(){

this.dialogChapterFormVisible = false// 如果保存成功则关闭对话框

this.fetchChapterNestedListByCourseId()// 刷新列表

this.chapter.title = ''// 重置章节标题

this.chapter.sort = 0// 重置章节标题

this.saveBtnDisabled = false

},

## 三、修改章节信息

### 1、编辑章节按钮

<el-button type="text" @click="editChapter(chapter.id)">编辑</el-button>

### 2、定义编辑方法

editChapter(chapterId) {

this.dialogChapterFormVisible = true

chapter.getById(chapterId).then(response => {

this.chapter = response.data.item

})

},

### 3、定义更新方法

updateData() {

chapter.updateById(this.chapter).then(response => {

this.$message({

type: 'success',

message: '修改成功!'

})

this.helpSave()

}).catch((response) => {

// console.log(response)

this.$message({

type: 'error',

message: response.message

})

})

},

## 四、删除章节

### 1、按钮

<el-button type="text" @click="removeChapter(chapter.id)">删除</el-button>

### 2、定义删除方法

removeChapter(chapterId) {

this.$confirm('此操作将永久删除该记录, 是否继续?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'warning'

}).then(() => {

return chapter.removeById(chapterId)

}).then(() => {

this.fetchChapterNestedListByCourseId()// 刷新列表

this.$message({

type: 'success',

message: '删除成功!'

})

}).catch((response) => { // 失败

if (response === 'cancel') {

this.$message({

type: 'info',

message: '已取消删除'

})

} else {

this.$message({

type: 'error',

message: response.message

})

}

})

},

# 2-4 小节课时管理后端开发

## 一、定义Form表单对象

VideoInfoForm.java

package com.guli.edu.form;

import io.swagger.annotations.ApiModel;

import io.swagger.annotations.ApiModelProperty;

import lombok.Data;

/\*\*

\* @author helen

\* @since 2019/3/5

\*/

@ApiModel(value = "课时基本信息", description = "编辑课时基本信息的表单对象")

@Data

public class VideoInfoForm {

@ApiModelProperty(value = "视频ID")

private String id;

@ApiModelProperty(value = "节点名称")

private String title;

@ApiModelProperty(value = "课程ID")

private String courseId;

@ApiModelProperty(value = "章节ID")

private String chapterId;

@ApiModelProperty(value = "视频资源")

private String videoSourceId;

@ApiModelProperty(value = "显示排序")

private Integer sort;

@ApiModelProperty(value = "是否可以试听：0默认 1免费")

private Boolean free;

}

## 二、课时保存

### 1、web层接口的定义

VideoAdminController.java

package com.guli.edu.controller.admin;

@Api(description="课时管理")

@CrossOrigin //跨域

@RestController

@RequestMapping("/admin/edu/video")

public class VideoAdminController {

@Autowired

private VideoService videoService;

@ApiOperation(value = "新增课时")

@PostMapping("save-video-info")

public R save(

@ApiParam(name = "videoForm", value = "课时对象", required = true)

@RequestBody VideoInfoForm videoInfoForm){

videoService.saveVideoInfo(videoInfoForm);

return R.ok();

}

}

### 2、业务层

VideoService.java

void saveVideoInfo(VideoInfoForm videoInfoForm);

VideoServiceImpl.java

@Override

public void saveVideoInfo(VideoInfoForm videoInfoForm) {

Video video = new Video();

BeanUtils.copyProperties(videoInfoForm, video);

boolean result = this.save(video);

if(!result){

throw new GuliException(20001, "课时信息保存失败");

}

}

## 三、课时的修改

### 1、web层接口的定义

VideoAdminController.java

@ApiOperation(value = "根据ID查询课时")

@GetMapping("video-info/{id}")

public R getVideInfoById(

@ApiParam(name = "id", value = "课时ID", required = true)

@PathVariable String id){

VideoInfoForm videoInfoForm = videoService.getVideoInfoFormById(id);

return R.ok().data("item", videoInfoForm);

}

@ApiOperation(value = "更新课时")

@PutMapping("update-video-info/{id}")

public R updateCourseInfoById(

@ApiParam(name = "VideoInfoForm", value = "课时基本信息", required = true)

@RequestBody VideoInfoForm videoInfoForm,

@ApiParam(name = "id", value = "课时ID", required = true)

@PathVariable String id){

videoService.updateVideoInfoById(videoInfoForm);

return R.ok();

}

### 2、业务层

VideoService.java

VideoInfoForm getVideoInfoFormById(String id);

void updateVideoInfoById(VideoInfoForm videoInfoForm);

VideoServiceImpl.java

@Override

public VideoInfoForm getVideoInfoFormById(String id) {

//从video表中取数据

Video video = this.getById(id);

if(video == null){

throw new GuliException(20001, "数据不存在");

}

//创建videoInfoForm对象

VideoInfoForm videoInfoForm = new VideoInfoForm();

BeanUtils.copyProperties(video, videoInfoForm);

return videoInfoForm;

}

@Override

public void updateVideoInfoById(VideoInfoForm videoInfoForm) {

//保存课时基本信息

Video video = new Video();

BeanUtils.copyProperties(videoInfoForm, video);

boolean result = this.updateById(video);

if(!result){

throw new GuliException(20001, "课时信息保存失败");

}

}

## 四、课时的删除

### 1、web层接口的定义

VideoAdminController.java

@ApiOperation(value = "根据ID删除课时")

@DeleteMapping("{id}")

public R removeById(

@ApiParam(name = "id", value = "课时ID", required = true)

@PathVariable String id){

boolean result = videoService.removeVideoById(id);

if(result){

return R.ok();

}else{

return R.error().message("删除失败");

}

}

### 2、业务层

VideoService.java

boolean removeVideoById(String id);

VideoServiceImpl.java

@Override

public boolean removeVideoById(String id) {

//删除视频资源 TODO

Integer result = baseMapper.deleteById(id);

return null != result && result > 0;

}

# 2-5 小节课时管理前端开发

## 一、定义api

创建video.js

参考course.js

import request from '@/utils/request'

const api\_name = '/admin/edu/video'

export default {

saveVideoInfo(videoInfo) {

return request({

url: `${api\_name}/save-video-info`,

method: 'post',

data: videoInfo

})

},

getVideoInfoById(id) {

return request({

url: `${api\_name}/video-info/${id}`,

method: 'get'

})

},

updateVideoInfoById(videoInfo) {

return request({

url: `${api\_name}/update-video-info/${videoInfo.id}`,

method: 'put',

data: videoInfo

})

},

removeById(id) {

return request({

url: `${api\_name}/${id}`,

method: 'delete'

})

}

}

## 二、新增课时页面功能

### 1、定义data数据

saveVideoBtnDisabled: false, // 课时按钮是否禁用

dialogVideoFormVisible: false, // 是否显示课时表单

chapterId: '', // 课时所在的章节id

video: {// 课时对象

title: '',

sort: 0,

free: 0,

videoSourceId: ''

},

### 2、添加课时按钮

<el-button type="text" @click="dialogVideoFormVisible = true; chapterId = chapter.id">添加课时</el-button>

### 3、课时表单dialog

    <!-- 添加和修改课时表单 -->

    <el-dialog :visible.sync="dialogVideoFormVisible" title="添加课时">

      <el-form :model="video" label-width="120px">

        <el-form-item label="课时标题">

          <el-input v-model="video.title" />

        </el-form-item>

        <el-form-item label="课时排序">

          <el-input-number v-model="video.sort" :min="0" controls-position="right" />

        </el-form-item>

        <el-form-item label="是否免费">

          <el-radio-group v-model="video.free">

            <el-radio :label="true">免费</el-radio>

            <el-radio :label="false">默认</el-radio>

          </el-radio-group>

        </el-form-item>

        <el-form-item label="上传视频">

          <!-- TODO -->

        </el-form-item>

      </el-form>

      <div slot="footer" class="dialog-footer">

        <el-button @click="dialogVideoFormVisible = false">取 消</el-button>

        <el-button :disabled="saveVideoBtnDisabled" type="primary" @click="saveOrUpdateVideo">确 定</el-button>

      </div>

    </el-dialog>

### 4、添加课时methods

引入video模块

import video from '@/api/edu/video'

方法的定义

saveOrUpdateVideo() {

this.saveVideoBtnDisabled = true

if (!this.video.id) {

this.saveDataVideo()

} else {

this.updateDataVideo()

}

},

saveDataVideo() {

this.video.courseId = this.courseId

this.video.chapterId = this.chapterId

video.saveVideoInfo(this.video).then(response => {

this.$message({

type: 'success',

message: '保存成功!'

})

this.helpSaveVideo()

})

},

updateDataVideo() {

},

helpSaveVideo() {

this.dialogVideoFormVisible = false// 如果保存成功则关闭对话框

this.fetchChapterNestedListByCourseId()// 刷新列表

this.video.title = ''// 重置章节标题

this.video.sort = 0// 重置章节标题

this.video.videoSourceId = ''// 重置视频资源id

this.saveVideoBtnDisabled = false

},

## 三、修改课时信息

### 1、编辑课时按钮

<el-button type="text" @click="editVideo(video.id)">编辑</el-button>

### 2、定义编辑方法

editVideo(videoId) {

this.dialogVideoFormVisible = true

video.getVideoInfoById(videoId).then(response => {

this.video = response.data.item

})

},

### 3、定义更新方法

updateDataVideo() {

video.updateVideoInfoById(this.video).then(response => {

this.$message({

type: 'success',

message: '修改成功!'

})

this.helpSaveVideo()

})

},

## 四、删除课时

### 1、按钮

<el-button type="text" @click="removeVideo(video.id)">删除</el-button>

### 2、定义删除方法

removeVideo(videoId) {

this.$confirm('此操作将永久删除该记录, 是否继续?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'warning'

}).then(() => {

return video.removeById(videoId)

}).then(() => {

this.fetchChapterNestedListByCourseId()// 刷新列表

this.$message({

type: 'success',

message: '删除成功!'

})

}).catch((response) => { // 失败

if (response === 'cancel') {

this.$message({

type: 'info',

message: '已取消删除'

})

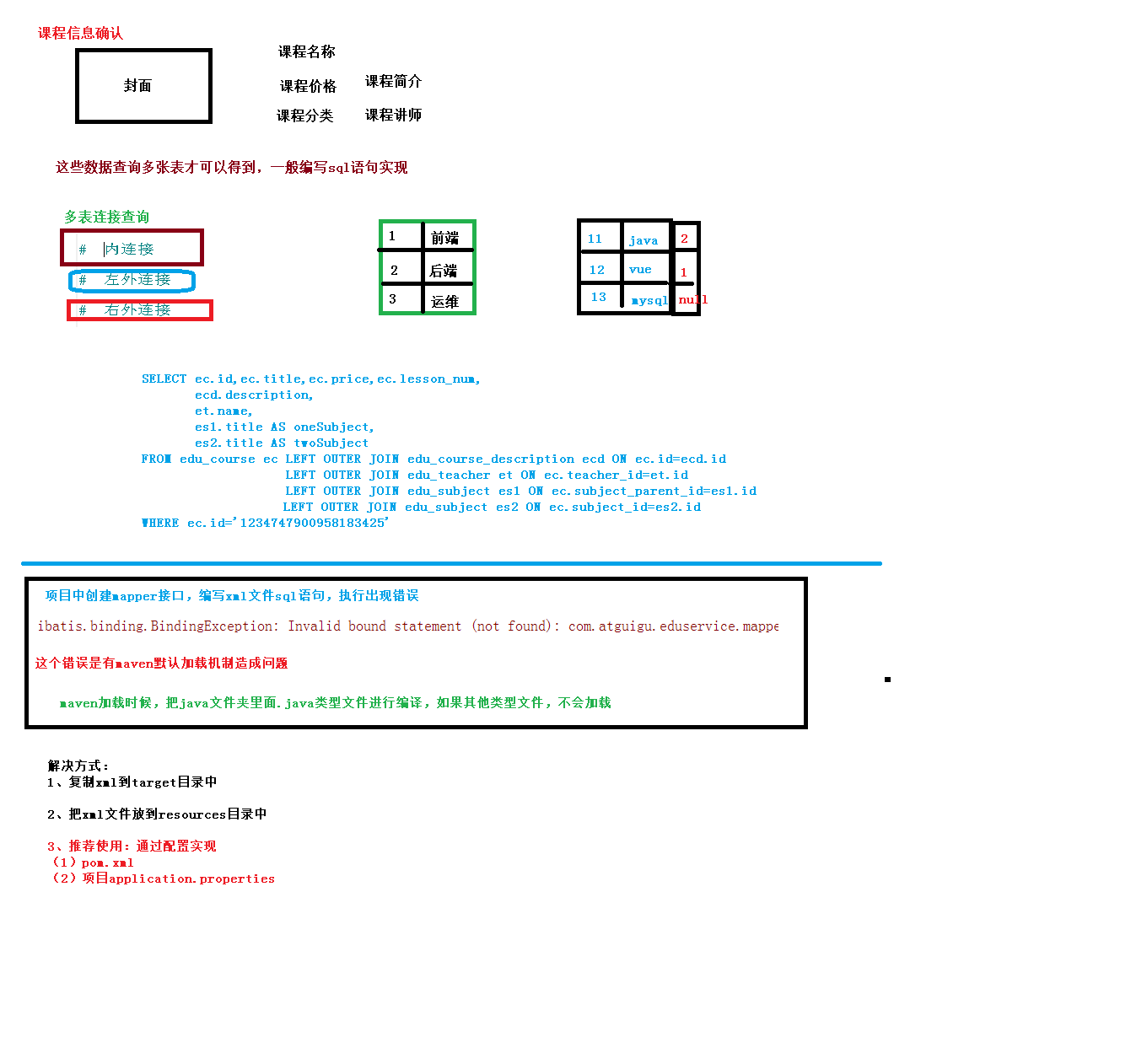
}

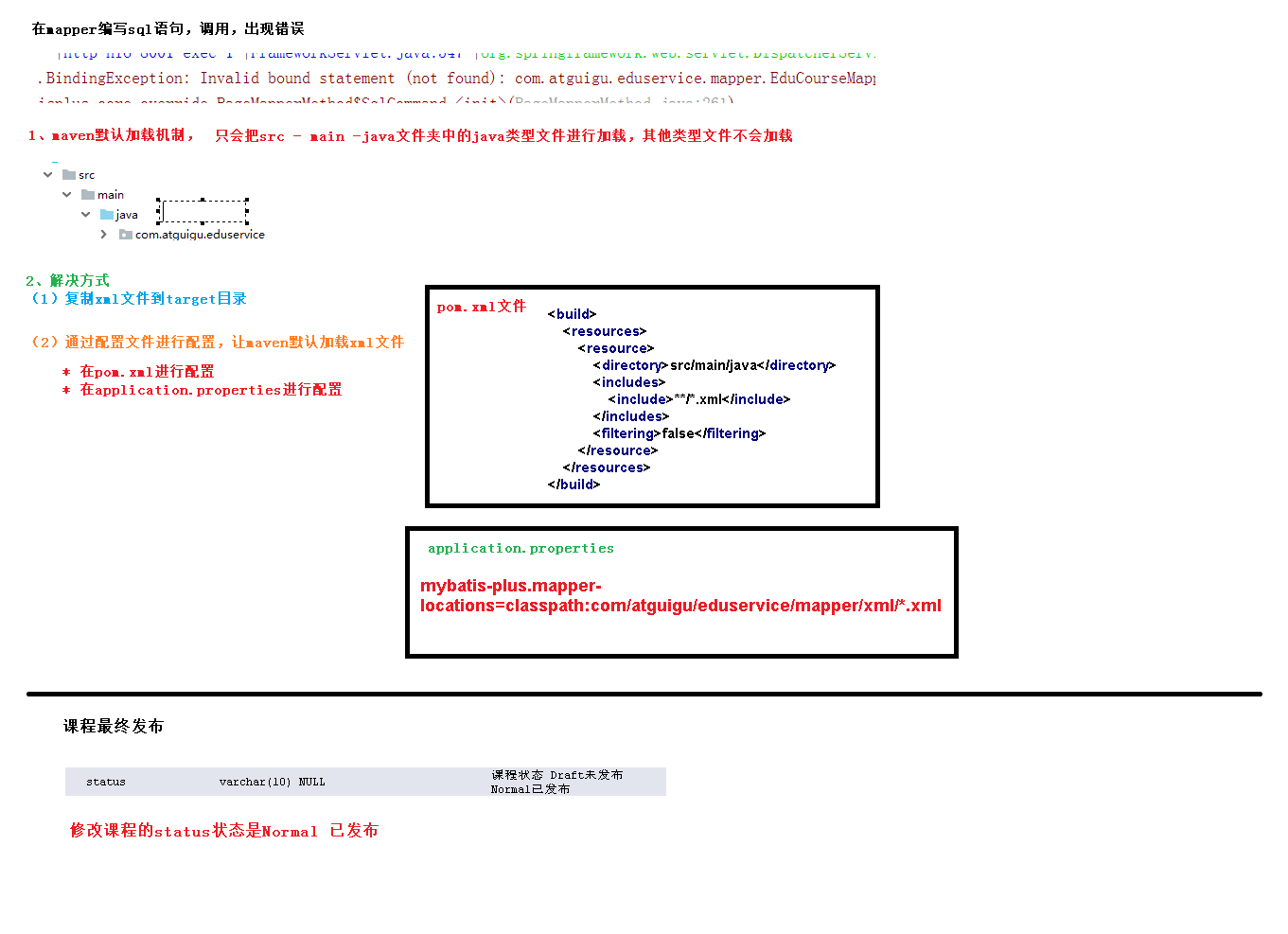
})

}

Step3 发布页面

# 3-1 课程最终发布后端





## 一、根据id查询课程发布信息

方式一：业务层组装多个表多次的查询结果

方式二：数据访问层进行关联查询

我们使用第二种方式实现

### 1、定义vo

package com.guli.edu.vo;

@ApiModel*(*value = "课程发布信息"*)*

@Data  
public class CoursePublishVo implements Serializable *{* private static final long *serialVersionUID* = 1L;  
 private String title;  
 private String cover;  
 private Integer lessonNum;  
 private String subjectLevelOne;  
 private String subjectLevelTwo;  
 private String teacherName;  
 private String price;//只用于显示  
*}*

### 2、数据访问层

接口：CourseMapper.java

package com.guli.edu.mapper;

public interface CourseMapper extends BaseMapper<Course> {

CoursePublishVo selectCoursePublishVoById(String id);

}

实现：CourseMapper.xml

*<*select id="getCoursePublishVoById" resultType="com.guli.edu.vo.CoursePublishVo"*>*

SELECT c.title,  
 c.cover,  
 c.lesson\_num AS lessonNum,  
 CONVERT(c.price, DECIMAL (8, 2)) AS price,  
 s1.title AS subjectLevelOne,  
 s2.title AS subjectLevelTwo,  
 t.name AS teacherName  
 FROM edu\_course c  
 LEFT JOIN edu\_teacher t ON c.teacher\_id = t.id  
 LEFT JOIN edu\_subject s1 ON c.subject\_parent\_id = s1.id  
 LEFT JOIN edu\_subject s2 ON c.subject\_id = s2.id  
 WHERE c.id = #{id}  
*</*select*>*

### 3、业务层

接口：CourseService.java

CoursePublishVo getCoursePublishVoById(String id);

实现：CourseServiceImpl.java

@Override

public CoursePublishVo getCoursePublishVoById(String id) {

return baseMapper.getCoursePublishVoById(id);

}

### 4、web层

@ApiOperation(value = "根据ID获取课程发布信息")

@GetMapping("course-publish-info/{id}")

public R getCoursePublishVoById(

@ApiParam(name = "id", value = "课程ID", required = true)

@PathVariable String id){

CoursePublishVo courseInfoForm = courseService.getCoursePublishVoById(id);

return R.ok().data("item", courseInfoForm);

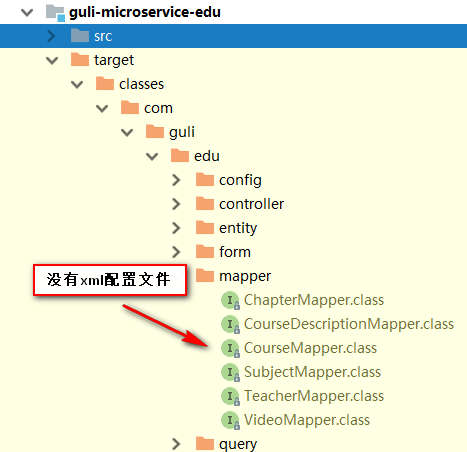
}

### 测试：报告异常 mapper xml绑定异常

AbstractHandlerExceptionResolver.java:194 |org.springframework.web.servlet.mvc.method.annotation.ExceptionHandlerExceptionResolver |Resolved exception caused by handler execution: org.apache.ibatis.binding.BindingException: Invalid bound statement (not found): com.guli.edu.mapper.CourseMapper.getCoursePublishVoById

**问题分析：**

dao层编译后只有class文件，没有mapper.xml，因为maven工程在默认情况下src/main/java目录下的所有资源文件是不发布到target目录下的，



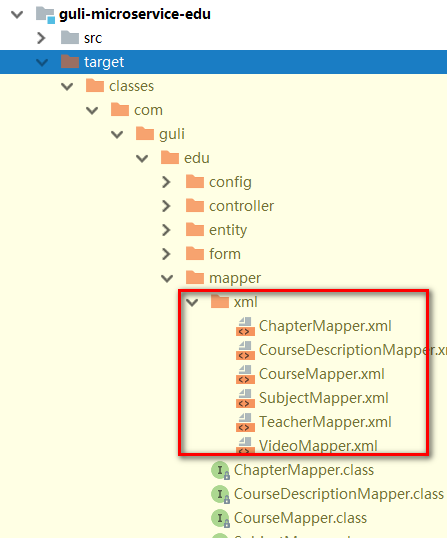
**解决方案：**

1、在guli\_edu的pom中配置如下节点

<!-- 项目打包时会将java目录中的\*.xml文件也进行打包 -->

*<*build*>  
 <*resources*>  
 <*resource*>  
 <*directory*>*src/main/java*</*directory*>  
 <*includes*>  
 <*include*>*\*\*/\*.xml*</*include*>  
 </*includes*>  
 <*filtering*>*false*</*filtering*>  
 </*resource*>  
 </*resources*>  
</*build*>*

重新打包项目会发现target目录下出现了xml文件夹



2、在Spring Boot配置文件中添加配置

#配置mapper xml文件的路径

mybatis-plus.mapper-locations=classpath:com/guli/edu/mapper/xml/\*.xml

## 二、根据id发布课程

### 1、web层

@ApiOperation(value = "根据id发布课程")

@PutMapping("publish-course/{id}")

public R publishCourseById(

@ApiParam(name = "id", value = "课程ID", required = true)

@PathVariable String id){

courseService.publishCourseById(id);

return R.ok();

}

### 2、service层

接口

void publishCourseById(String id);

实现

@Override

public boolean publishCourseById(String id) {

Course course = new Course();

course.setId(id);

course.setStatus(Course.COURSE\_NORMAL);

Integer count = baseMapper.updateById(course);

return null != count && count > 0;

}

# 3-2 课程最终发布前端

## 一、前端代码

### 1、定义api

分析这个页面一共有两个远程方法：一个是根基课程id获取课程基本预览信息，第二个是发布课程

getCoursePublishInfoById(id) {

return request({

url: `${api\_name}/course-publish-info/${id}`,

method: 'get'

})

},

publishCourse(id) {

return request({

url: `${api\_name}/publish-course/${id}`,

method: 'put'

})

}

### 2、定义数据模型

data() {

return {

saveBtnDisabled: false, // 保存按钮是否禁用

courseId: '', // 所属课程

coursePublish: {}

}

},

### 3、完善步骤导航

edu/course/chapter.js

previous() {

console.log('previous')

this.$router.push({ path: '/edu/course/info/' + this.courseId })

},

next() {

console.log('next')

this.$router.push({ path: '/edu/course/publish/' + this.courseId })

}

edu/course/pubish.js

<div>

<el-button @click="previous">返回修改</el-button>

<el-button :disabled="saveBtnDisabled" type="primary" @click="publish">发布课程</el-button>

</div>

previous() {

console.log('previous')

this.$router.push({ path: '/edu/course/chapter/' + this.courseId })

},

publish() {

console.log('publish')

course.publishCourse(this.courseId).then(response => {

this.$router.push({ path: '/edu/course/list' })

})

}

### 4、组件方法定义

import

import course from '@/api/edu/course'

created

created() {

console.log('chapter created')

this.init()

},

获取数据的方法

init() {

if (this.$route.params && this.$route.params.id) {

this.courseId = this.$route.params.id

// 根据id获取课程基本信息

this.fetchCoursePublishInfoById()

}

},

fetchCoursePublishInfoById() {

course.getCoursePublishInfoById(this.courseId).then(response => {

this.coursePublish = response.data.item

})

},

### 5、组件模板

<template>

  <div class="app-container">

    <h2 style="text-align: center;">发布新课程</h2>

    <el-steps :active="3" process-status="wait" align-center style="margin-bottom: 40px;">

      <el-step title="填写课程基本信息"/>

      <el-step title="创建课程大纲"/>

      <el-step title="发布课程"/>

    </el-steps>

    <div class="ccInfo">

      <img :src="coursePublish.cover">

      <div class="main">

        <h2>{{ coursePublish.title }}</h2>

        <p class="gray"><span>共{{ coursePublish.lessonNum }}课时</span></p>

        <p><span>所属分类：{{ coursePublish.subjectLevelOne }} — {{ coursePublish.subjectLevelTwo }}</span></p>

        <p>课程讲师：{{ coursePublish.teacherName }}</p>

        <h3 class="red">￥{{ coursePublish.price }}</h3>

      </div>

    </div>

    <div>

      <el-button @click="previous">返回修改</el-button>

      <el-button :disabled="saveBtnDisabled" type="primary" @click="publish">发布课程</el-button>

    </div>

  </div>

</template>

### 6、css样式

<style scoped>

  .ccInfo {

*background*: #f5f5f5;

*padding*: 20px;

*overflow*: hidden;

*border*: 1px dashed #DDD;

*margin-bottom*: 40px;

*position*: relative;

  }

  .ccInfo img {

*background*: #d6d6d6;

*width*: 500px;

*height*: 278px;

*display*: block;

*float*: left;

*border*: none;

  }

  .ccInfo .main {

*margin-left*: 520px;

  }

  .ccInfo .main h2 {

*font-size*: 28px;

*margin-bottom*: 30px;

*line-height*: 1;

*font-weight*: normal;

  }

  .ccInfo .main p {

*margin-bottom*: 10px;

*word-wrap*: break-word;

*line-height*: 24px;

*max-height*: 48px;

*overflow*: hidden;

  }

  .ccInfo .main p {

*margin-bottom*: 10px;

*word-wrap*: break-word;

*line-height*: 24px;

*max-height*: 48px;

*overflow*: hidden;

  }

  .ccInfo .main h3 {

*left*: 540px;

*bottom*: 20px;

*line-height*: 1;

*font-size*: 28px;

*color*: #d32f24;

*font-weight*: normal;

*position*: absolute;

  }

  </style>