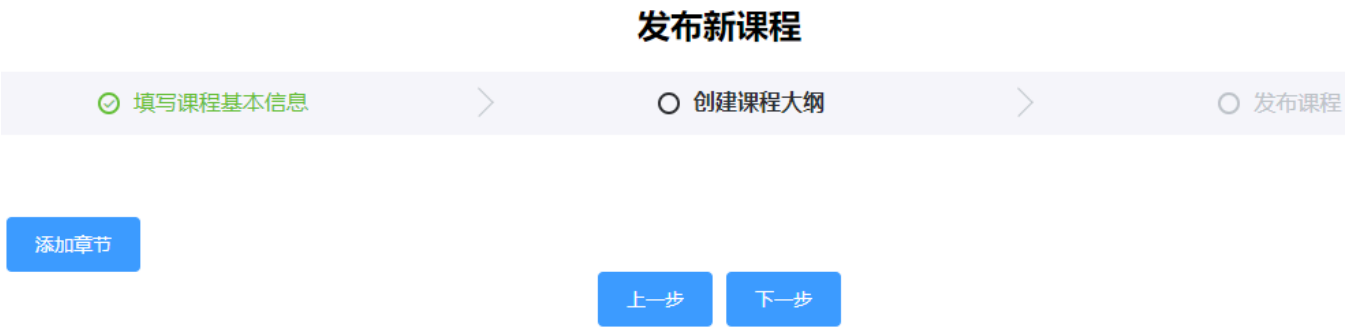
硅谷课堂第八天-点播管理模块（二）

一、发布课程-创建课程大纲



1、课程章节接口

实现课程章节的列表、添加、修改和删除功能

1.1、编写章节 Controller

```java
@RestController
@RequestMapping(value="/admin/vod/chapter")
@CrossOrigin
public class ChapterController {

    @Autowired
    private ChapterService chapterService;

    //获取章节小结列表
    @ApiOperation("嵌套章节数据列表")
    @GetMapping("getNestedTreeList/{courseId}")
    public Result getNestedTreeList(
            @ApiParam(value = "课程ID", required = true)
            @PathVariable Long courseId){

        List<ChapterVo> chapterVoList = chapterService.getNestedTreeList(courseId);
        return Result.ok(chapterVoList);
    }

    //2 添加章节
    @PostMapping("save")
    public Result save(@RequestBody Chapter chapter) {
        chapterService.save(chapter);
        return Result.ok(null);
    }

    //3 修改-根据id查询
    @GetMapping("get/{id}")
    public Result get(@PathVariable Long id) {
        Chapter chapter = chapterService.getById(id);
        return Result.ok(chapter);
    }

    //4 修改-最终实现
    @PostMapping("update")
    public Result update(@RequestBody Chapter chapter) {
        chapterService.updateById(chapter);
```

```
        return Result.ok(null);
    }

    //5 删除章节
    @DeleteMapping("remove/{id}")
    public Result remove(@PathVariable Long id) {
        chapterService.removeById(id);
        return Result.ok(null);
    }
}
```

1.2、编写章节 Service

（1）ChapterService

```
public interface ChapterService extends IService<Chapter> {
    //章节小结列表
    List<ChapterVo> getNestedTreeList(Long courseId);
}
```

（2）ChapterServiceImpl

```java
@Service
public class ChapterServiceImpl extends ServiceImpl<ChapterMapper, Chapter> implements ChapterServ

    @Autowired
    private VideoService videoService;

    //章节小结列表封装
    @Override
    public List<ChapterVo> getNestedTreeList(Long courseId) {
        List<ChapterVo> chapterVoList = new ArrayList<>();

        //获取章信息
        LambdaQueryWrapper<Chapter> queryWrapperChapter = new LambdaQueryWrapper<>();
        queryWrapperChapter.eq(Chapter::getCourseId, courseId);
        queryWrapperChapter.orderByAsc(Chapter::getSort, Chapter::getId);
        List<Chapter> chapterList = baseMapper.selectList(queryWrapperChapter);

        //获取课时信息
        LambdaQueryWrapper<Video> queryWrapperVideo = new LambdaQueryWrapper<>();
        queryWrapperVideo.eq(Video::getCourseId, courseId);
        queryWrapperVideo.orderByAsc(Video::getSort, Video::getId);
        List<Video> videoList = videoService.list(queryWrapperVideo);

        //填充列表数据：Chapter列表
        for (int i = 0; i < chapterList.size(); i++) {
            Chapter chapter = chapterList.get(i);

            //创建ChapterVo对象
            ChapterVo chapterVo = new ChapterVo();
            BeanUtils.copyProperties(chapter, chapterVo);
            chapterVoList.add(chapterVo);

            //填充列表数据：Video列表
            List<VideoVo> videoVoList = new ArrayList<>();
            for (int j = 0; j < videoList.size(); j++) {
                Video video = videoList.get(j);
                if(chapter.getId().equals(video.getChapterId())){
```

```
                    VideoVo videoVo = new VideoVo();
                    BeanUtils.copyProperties(video, videoVo);
                    videoVoList.add(videoVo);
                }
            }
            chapterVo.setChildren(videoVoList);
        }
        return chapterVoList;
    }
}
```

2、课程小节接口

2.1、编写 VideoController

```java
@Api(tags = "课程小结（课时）")
@RestController
@RequestMapping(value="/admin/vod/video")
@CrossOrigin
public class VideoController {

    @Autowired
    private VideoService videoService;

    @ApiOperation(value = "获取")
    @GetMapping("get/{id}")
    public Result get(@PathVariable Long id) {
        Video video = videoService.getById(id);
        return Result.ok(video);
    }

    @ApiOperation(value = "新增")
    @PostMapping("save")
    public Result save(@RequestBody Video video) {
        videoService.save(video);
        return Result.ok(null);
    }

    @ApiOperation(value = "修改")
    @PutMapping("update")
    public Result updateById(@RequestBody Video video) {
        videoService.updateById(video);
        return Result.ok(null);
    }

    @ApiOperation(value = "删除")
    @DeleteMapping("remove/{id}")
    public Result remove(@PathVariable Long id) {
        videoService.removeById(id);
        return Result.ok(null);
    }
}
```

3、课程大纲前端

3.1、定义接口

（1）chapter.js

```
import request from "@/utils/request";

const api_name = "/admin/vod/chapter";

export default {
  getNestedTreeList(courseId) {
    return request({
      url: `${api_name}/getNestedTreeList/${courseId}`,
      method: "get",
    });
  },

  removeById(id) {
    return request({
      url: `${api_name}/remove/${id}`,
      method: "delete",
    });
  },

  save(chapter) {
    return request({
      url: `${api_name}/save`,
      method: "post",
      data: chapter,
    });
  },

  getById(id) {
    return request({
      url: `${api_name}/get/${id}`,
      method: "get",
    });
  },

  updateById(chapter) {
    return request({
      url: `${api_name}/update`,
```

```
      method: "put",
      data: chapter,
    });
  },
};
```

（2）创建 video.js

```
import request from "@/utils/request";

const api_name = "/admin/vod/video";

export default {
  save(video) {
    return request({
      url: `${api_name}/save`,
      method: "post",
      data: video,
    });
  },

  getById(id) {
    return request({
      url: `${api_name}/get/${id}`,
      method: "get",
    });
  },

  updateById(video) {
    return request({
      url: `${api_name}/update`,
      method: "put",
      data: video,
    });
  },

  removeById(id) {
    return request({
      url: `${api_name}/remove/${id}`,
      method: "delete",
    });
  },
};
```

3.2、编写章节页面

（1）Chapter/index.vue

```
<template>
  <div class="app-container">
    <!-- 添加章节按钮 -->
    <div>
      <el-button type="primary" @click="addChapter()">添加章节</el-button>
    </div>

    <!-- 章节列表 -->
    <ul class="chapterList">
      <li v-for="chapter in chapterList" :key="chapter.id">
        <p>
          {{ chapter.title }}
          <span class="acts">
            <el-button type="text" @click="addVideo(chapter.id)"
              >添加课时</el-button
            >
            <el-button type="text" @click="editChapter(chapter.id)"
              >编辑</el-button
            >
            <el-button type="text" @click="removeChapterById(chapter.id)"
              >删除</el-button
            >
          </span>
        </p>
        <!-- 视频 -->
        <ul class="chapterList videoList">
          <li v-for="video in chapter.children" :key="video.id">
            <p>
              {{ video.title }}
              <el-tag v-if="!video.videoSourceId" size="mini" type="danger">
                {{ "尚未上传视频" }}
              </el-tag>
              <span class="acts">
                <el-tag v-if="video.isFree" size="mini" type="success"
                  >{{ "免费观看" }}</el-tag
                >
                <el-button type="text" @click="editVideo(chapter.id, video.id)"
```

```
                >编辑</el-button
              >
              <el-button type="text" @click="removeVideoById(video.id)"
                >删除</el-button
              >
            </span>
          </p>
        </li>
      </ul>
    </li>
  </ul>
  <!-- 章节表单对话框 -->
  <chapter-form ref="chapterForm" />
  <!-- 课时表单对话框 -->
  <video-form ref="videoForm" />
  <div style="text-align:center">
    <el-button type="primary" @click="prev()">上一步</el-button>
    <el-button type="primary" @click="next()">下一步</el-button>
  </div>
</div>
</template>
<script>
  import chapterApi from "@/api/vod/chapter";
  import videoApi from "@/api/vod/video";

  // 引入组件
  import ChapterForm from "@/views/vod/course/components/Chapter/Form";
  import VideoForm from "@/views/vod/course/components/Video/Form";

  export default {
    // 注册组件
    components: { ChapterForm, VideoForm },
    data() {
      return {
        chapterList: [], // 章节嵌套列表
      };
    },
```

```
created() {
  this.fetchNodeList();
},
methods: {
  // 获取章节小节数据
  fetchNodeList() {
    chapterApi.getNestedTreeList(this.$parent.courseId).then((response) => {
      this.chapterList = response.data;
    });
  },
  //删除章节
  removeChapterById(chapterId) {
    this.$confirm("此操作将永久删除该章节，是否继续?", "提示", {
      confirmButtonText: "确定",
      cancelButtonText: "取消",
      type: "warning",
    })
      .then(() => {
        return chapterApi.removeById(chapterId);
      })
      .then((response) => {
        this.fetchNodeList();
        this.$message.success(response.message);
      })
      .catch((response) => {
        if (response === "cancel") {
          this.$message.info("取消删除");
        }
      });
  },
  // 添加章节
  addChapter() {
    this.$refs.chapterForm.open();
  },
  // 编辑章节
  editChapter(chapterId) {
    this.$refs.chapterForm.open(chapterId);
```

```
  },
  // 添加课时
  addVideo(chapterId) {
    this.$refs.videoForm.open(chapterId);
  },
  // 编辑课时
  editVideo(chapterId, videoId) {
    this.$refs.videoForm.open(chapterId, videoId);
  },
  // 删除课时
  removeVideoById(videoId) {
    this.$confirm("此操作将永久删除该课时，是否继续?", "提示", {
      confirmButtonText: "确定",
      cancelButtonText: "取消",
      type: "warning",
    })
      .then(() => {
        return videoApi.removeById(videoId);
      })
      .then((response) => {
        this.fetchNodeList();
        this.$message.success(response.message);
      })
      .catch((response) => {
        if (response === "cancel") {
          this.$message.info("取消删除");
        }
      });
  },
  // 上一步
  prev() {
    this.$parent.active = 0;
  },
  // 下一步
  next() {
    this.$parent.active = 2;
  },
```

```
    },
  };
</script>
```

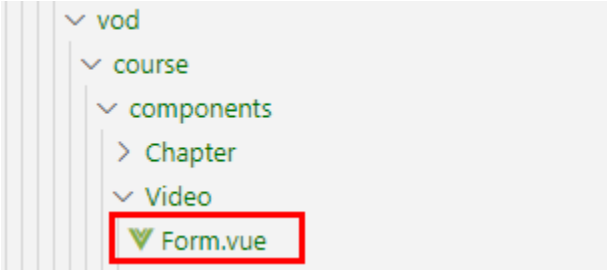
（2）Chapter/Form.vue

```
<template>
  <!-- 添加和修改章节表单 -->
  <el-dialog :visible="dialogVisible" title="添加章节" @close="close()">
    <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" />
      </el-form-item>
    </el-form>
    <div slot="footer" class="dialog-footer">
      <el-button @click="close()">取 消</el-button>
      <el-button type="primary" @click="saveOrUpdate()">确 定</el-button>
    </div>
  </el-dialog>
</template>

<script>
  import chapterApi from "@/api/vod/chapter";
  export default {
    data() {
      return {
        dialogVisible: false,
        chapter: {
          sort: 0,
        },
      };
    },
    methods: {
      open(chapterId) {
        this.dialogVisible = true;
        if (chapterId) {
          chapterApi.getById(chapterId).then((response) => {
            this.chapter = response.data.item;
          });
        }
```

```
  },

  close() {
    this.dialogVisible = false;
    // 重置表单
    this.resetForm();
  },

  resetForm() {
    this.chapter = {
      sort: 0,
    };
  },

  saveOrUpdate() {
    if (!this.chapter.id) {
      this.save();
    } else {
      this.update();
    }
  },

  save() {
    this.chapter.courseId = this.$parent.$parent.courseId;
    chapterApi.save(this.chapter).then((response) => {
      this.$message.success(response.message);
      // 关闭组件
      this.close();
      // 刷新列表
      this.$parent.fetchNodeList();
    });
  },

  update() {
    chapterApi.updateById(this.chapter).then((response) => {
      this.$message.success(response.message);
      // 关闭组件
```

```
          this.close();
          // 刷新列表
          this.$parent.fetchNodeList();
        });
      },
    },
  };
</script>
```

3.3、编写小节（课时）页面



（1）Video/Form.vue

```
<template>
  <!-- 添加和修改课时表单 -->
  <el-dialog :visible="dialogVisible" title="添加课时" @close="close()">
    <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" />
      </el-form-item>
      <el-form-item label="是否免费">
        <el-radio-group v-model="video.isFree">
          <el-radio :label="0">免费</el-radio>
          <el-radio :label="1">默认</el-radio>
        </el-radio-group>
      </el-form-item>

      <!-- 上传视频 -->
      <el-form-item label="上传视频">
        <el-upload
          ref="upload"
          :auto-upload="false"
          :on-success="handleUploadSuccess"
          :on-error="handleUploadError"
          :on-exceed="handleUploadExceed"
          :file-list="fileList"
          :limit="1"
          :before-remove="handleBeforeRemove"
          :on-remove="handleOnRemove"
          :action="BASE_API + '/admin/vod/upload'"
        >
          <el-button slot="trigger" size="small" type="primary"
            >选择视频</el-button
          >
          <el-button
            :disabled="uploadBtnDisabled"
            style="margin-left: 10px;"
```

```
            size="small"
            type="success"
            @click="submitUpload()"
            >上传</el-button
          >
        </el-upload>
      </el-form-item>
    </el-form>
    <div slot="footer" class="dialog-footer">
      <el-button @click="close()">取 消</el-button>
      <el-button type="primary" @click="saveOrUpdate()">确 定</el-button>
    </div>
  </el-dialog>
</template>

<script>
  import videoApi from "@/api/vod/video";
  //import vodApi from '@/api/vod/vod'
  export default {
    data() {
      return {
        BASE_API: "http://localhost:8301",
        dialogVisible: false,
        video: {
          sort: 0,
          free: false,
        },
        fileList: [], // 上传文件列表
        uploadBtnDisabled: false,
      };
    },

    methods: {
      open(chapterId, videoId) {
        this.dialogVisible = true;
        this.video.chapterId = chapterId;
        if (videoId) {
```

```
      videoApi.getById(videoId).then((response) => {
        this.video = response.data;
        // 回显
        if (this.video.videoOriginalName) {
          this.fileList = [{ name: this.video.videoOriginalName }];
        }
      });
    }
  },

  close() {
    this.dialogVisible = false;
    // 重置表单
    this.resetForm();
  },

  resetForm() {
    this.video = {
      sort: 0,
      free: false,
    };

    this.fileList = []; // 重置视频上传列表
  },

  saveOrUpdate() {
    if (!this.video.id) {
      this.save();
    } else {
      this.update();
    }
  },

  save() {
    this.video.courseId = this.$parent.$parent.courseId;
    videoApi.save(this.video).then((response) => {
      this.$message.success(response.message);
```

```
        // 关闭组件
        this.close();
        // 刷新列表
        this.$parent.fetchNodeList();
      });
    },

    update() {
      videoApi.updateById(this.video).then((response) => {
        this.$message.success(response.message);
        // 关闭组件
        this.close();
        // 刷新列表
        this.$parent.fetchNodeList();
      });
    },

    // 上传多于一个视频
    handleUploadExceed(files, fileList) {
      this.$message.warning("想要重新上传视频，请先删除已上传的视频");
    },

    // 上传
    submitUpload() {
      this.uploadBtnDisabled = true;
      this.$refs.upload.submit(); // 提交上传请求
    },

    // 视频上传成功的回调
    handleUploadSuccess(response, file, fileList) {
      this.uploadBtnDisabled = false;
      this.video.videoSourceId = response.data;
      this.video.videoOriginalName = file.name;
    },

    // 失败回调
    handleUploadError() {
```

```
        this.uploadBtnDisabled = false;
        this.$message.error("上传失败2");
      },

      // 删除视频文件确认
      handleBeforeRemove(file, fileList) {
        return this.$confirm(`确定移除 ${file.name}? `);
      },

      // 执行视频文件的删除
      handleOnRemove(file, fileList) {
        if (!this.video.videoSourceId) {
          return;
        }
      },
    },
  };
</script>
```

二、发布课程-课程最终发布

1、课程最终发布接口

1.1、编写 CourseController

添加方法

```
@ApiOperation("根据id获取课程发布信息")
@GetMapping("getCoursePublishVo/{id}")
public Result getCoursePublishVoById(
        @ApiParam(value = "课程ID", required = true)
        @PathVariable Long id){

    CoursePublishVo coursePublishVo = courseService.getCoursePublishVo(id);
    return Result.ok(coursePublishVo);
}

@ApiOperation("根据id发布课程")
@PutMapping("publishCourseById/{id}")
public Result publishCourseById(
        @ApiParam(value = "课程ID", required = true)
        @PathVariable Long id){

    boolean result = courseService.publishCourseById(id);
    return Result.ok();
}
```

1.2、编写 CourseService

```
//根据id获取课程发布信息
CoursePublishVo getCoursePublishVo(Long id);

//根据id发布课程
boolean publishCourseById(Long id);
```

1.3、编写 CourseServiceImpl

```
//根据id获取课程发布信息
@Override
public CoursePublishVo getCoursePublishVo(Long id) {
  return courseMapper.selectCoursePublishVoById(id);
}

//根据id发布课程
@Override
public boolean publishCourseById(Long id) {
  Course course = new Course();
  course.setId(id);
  course.setPublishTime(new Date());
  course.setStatus(1);
  return this.updateById(course);
}
```

1.4、编写 CourseMapper

```
public interface CourseMapper extends BaseMapper<Course> {
  //根据id获取课程发布信息
    CoursePublishVo selectCoursePublishVoById(Long id);
}
```

1.5、编写 CourseMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-map
<mapper namespace="com.atguigu.ggkt.vod.mapper.CourseMapper">

    <select id="selectCoursePublishVoById" resultType="com.atguigu.ggkt.vo.vod.CoursePublishVo">
        SELECT
                c.id,
                c.title,
                c.cover,
                c.lesson_num AS lessonNum,
                c.price,
                t.name AS teacherName,
                s1.title AS subjectParentTitle,
                s2.title AS subjectTitle
        FROM course c
             LEFT OUTER JOIN teacher t ON c.teacher_id=t.id
             LEFT OUTER JOIN `subject` s1 ON c.subject_parent_id=s1.id
             LEFT OUTER JOIN `subject` s2 ON c.subject_id=s2.id
        WHERE c.id=#{id}
    </select>
</mapper>
```

1.6、添加配置

（1）application.properties 添加

```
mybatis-plus.mapper-locations=classpath:com/atguigu/ggkt/vod/mapper/xml/*.xml
```

（2）service 模块 pom.xml 添加

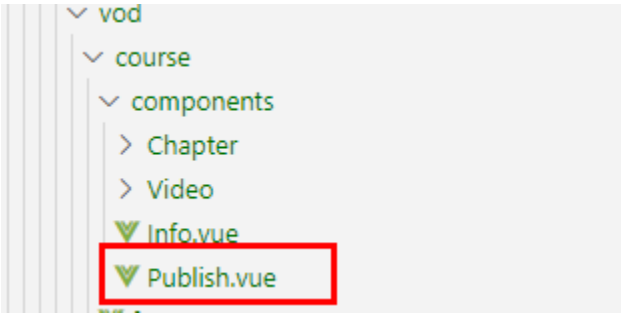
```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
    <resources>
        <resource>
            <directory>src/main/java</directory>
            <includes>
                <include>**/*.yml</include>
                <include>**/*.properties</include>
                <include>**/*.xml</include>
            </includes>
            <filtering>false</filtering>
        </resource>
        <resource>
            <directory>src/main/resources</directory>
            <includes> <include>**/*.yml</include>
                <include>**/*.properties</include>
                <include>**/*.xml</include>
            </includes>
            <filtering>false</filtering>
        </resource>
    </resources>
</build>
```

2、课程最终发布前端

2.1、course.js 定义接口

```
//获取发布课程信息
getCoursePublishById(id) {
    return request({
      url: `${api_name}/getCoursePublishVo/${id}`,
      method: 'get'
    })
},
//发布课程
publishCourseById(id) {
    return request({
      url: `${api_name}/publishCourseById/${id}`,
      method: 'put'
    })
},
```

2.2、编写 Publish.vue



```
<template>
  <div class="app-container">
    <!--课程预览-->
    <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.subjectParentTitle }} — {{
            coursePublish.subjectTitle }}</span
          >
        </p>
        <p>课程讲师：{{ coursePublish.teacherName }}</p>
        <h3 class="red">￥{{ coursePublish.price }}</h3>
      </div>
    </div>
    <div style="text-align:center">
      <el-button type="primary" @click="prev()">上一步</el-button>
      <el-button
        :disabled="publishBtnDisabled"
        type="primary"
        @click="publish()"
        >发布课程</el-button
      >
    </div>
  </div>
</template>

<script>
  import courseApi from "@/api/vod/course";

  export default {
    data() {
```

```
      return {
        publishBtnDisabled: false, // 按钮是否禁用
        coursePublish: {},
      };
    },
    created() {
      if (this.$parent.courseId) {
        this.fetchCoursePublishById(this.$parent.courseId);
      }
    },
    methods: {
      // 获取课程发布信息
      fetchCoursePublishById(id) {
        courseApi.getCoursePublishById(id).then((response) => {
          this.coursePublish = response.data;
        });
      },
      // 上一步
      prev() {
        this.$parent.active = 1;
      },
      // 下一步
      publish() {
        this.publishBtnDisabled = true;
        courseApi.publishCourseById(this.$parent.courseId).then((response) => {
          this.$parent.active = 3;
          this.$message.success(response.message);
          this.$router.push({ path: "/vodcourse/course/list" });
        });
      },
    },
  };
</script>
```

三、功能实现-课程删除

1、课程删除接口

1.1、编写课程 Controller

```
@ApiOperation(value = "删除课程")
@DeleteMapping("remove/{id}")
public Result remove(@PathVariable Long id) {
    courseService.removeCourseById(id);
    return Result.ok();
}
```

1.2、编写课程 Service

```
//删除课程
@Override
public void removeCourseById(Long id) {
    //根据课程id删除小节
    videoService.removeVideoByCourseId(id);
    //根据课程id删除章节
    chapterService.removeChapterByCourseId(id);
    //根据课程id删除描述
    descriptionService.removeById(id);
    //根据课程id删除课程
    baseMapper.deleteById(id);
}
```

1.3、编写 VideoService

```
@Service
public class VideoServiceImpl extends ServiceImpl<VideoMapper, Video> implements VideoService {

    //根据课程id删除小节
    @Override
    public void removeVideoByCourseId(Long id) {
        QueryWrapper<Video> wrapper = new QueryWrapper<>();
        wrapper.eq("course_id",id);
        baseMapper.delete(wrapper);
    }
}
```

1.4、编写 ChapterService

```
//根据课程id删除章节
@Override
public void removeChapterByCourseId(Long id) {
    QueryWrapper<Chapter> wrapper = new QueryWrapper<>();
    wrapper.eq("course_id",id);
    baseMapper.delete(wrapper);
}
```

2、课程删除前端

2.1、course.js 定义接口

```
  removeById(id) {
    return request({
      url: `${api_name}/remove/${id}`,
      method: 'delete'
    })
  },
```

2.2、course/list.vue 添加方法

```
methods: {
  // 根据id删除数据
  removeById(id) {
    this.$confirm('此操作将永久删除该课程，以及该课程下的章节和视频，是否继续?', '提示', {
      confirmButtonText: '确定',
      cancelButtonText: '取消',
      type: 'warning'
    }).then(() => {
      return courseApi.removeById(id)
    }).then(response => {
      this.fetchData()
      this.$message.success(response.message)
    }).catch((response) => { // 失败
      if (response === 'cancel') {
        this.$message.info('取消删除')
      }
    })
  }
}
```