

硅谷课堂第六天-整合腾讯云对象存储和课程分类管理

硅谷课堂第六天-整合腾讯云对象存储和课程分类管理

一、讲师管理模块整合腾讯云对象存储

1、腾讯云对象存储介绍

- 1.1、开通“对象存储 COS”服务
- 1.2、创建 Bucket
- 1.3、创建 API 秘钥
- 1.4、快速入门

2、整合腾讯云对象存储

- 2.1、service_vod 模块引入依赖
- 2.2、配置 application.properties
- 3.3、创建工具类
- 3.4、创建 Service
- 3.5、创建 Controller

3、添加讲师前端完善

- 3.1、添加上传组件
- 3.2、添加上传方法

二、后台管理系统-课程分类管理模块

- 1、课程分类管理模块需求
- 2、课程分类数据库设计

3、功能实现-课程分类列表

- 3.1、接口实现分析
- 3.2、编写 SubjectController
- 3.3、编写 SubjectService
- 3.4、编写 SubjectServiceImpl
- 3.5、开发课程分类列表前端

4、技术点-EasyExcel

- 4.1、EasyExcel 介绍
- 4.2、EasyExcel 特点
- 4.3、EasyExcel 写操作

- 4.4、EasyExcel 读操作
- 5、功能实现-课程分类导出
 - 5.1、查看 model 实体类
 - 5.2、编写 SubjectService 和实现
 - 5.3、添加 Controller 方法
 - 5.4、数据字典导出前端
- 6、功能实现-课程分类导入
 - 6.1、创建读取监听器
 - 6.2、添加 controller 方法
 - 6.3、添加 service 方法
 - 6.4、数据字典导入前端

一、讲师管理模块整合腾讯云对象存储

1、腾讯云对象存储介绍

对象存储 COS

对象存储（Cloud Object Storage，COS）是由腾讯云推出的无目录层次结构、无数据格式限制，可容纳海量数据且支持 HTTP/HTTPS 协议访问的分布式存储服务。腾讯云 COS 的存储桶空间无容量上限，无需分区管理，适用于 CDN 数据分发、数据万象处理或大数据计算与分析的数据湖等多种场景。

售前在线咨询

产品文档

购买资源包

立即使用 >

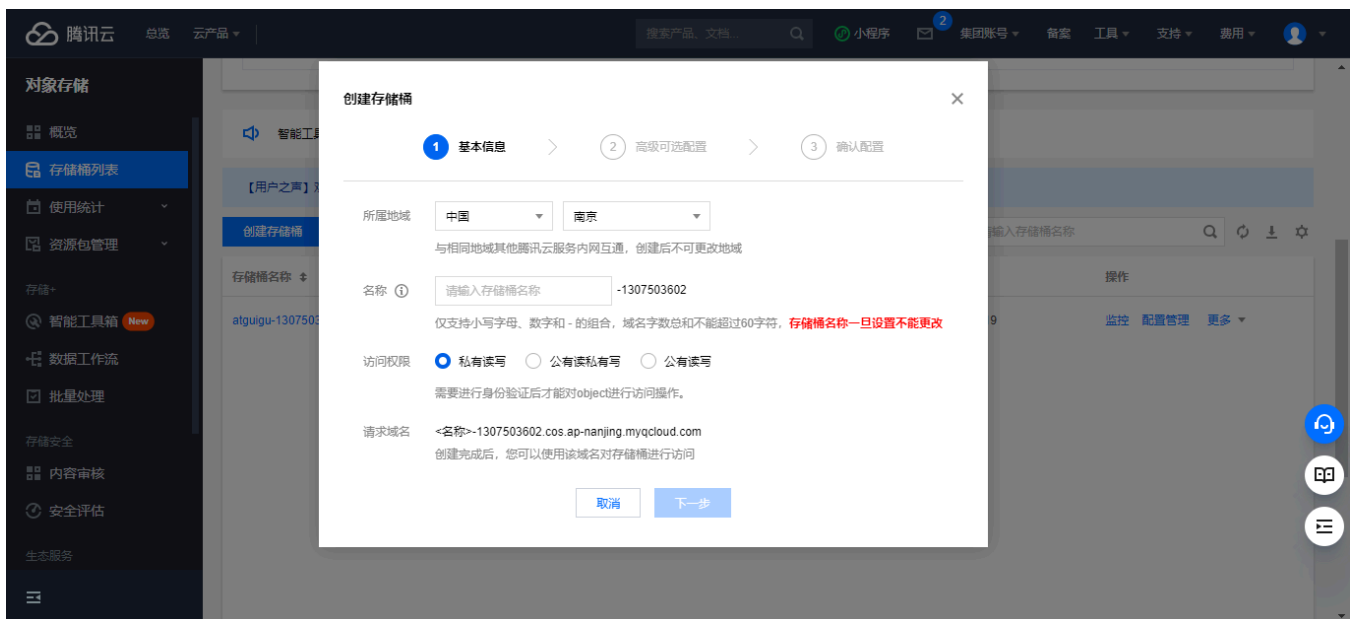
1.1、开通“对象存储 COS”服务

- (1) 申请腾讯云账号：<https://cloud.tencent.com/>
- (2) 实名认证
- (3) 开通“对象存储 COS”服务
- (4) 进入管理控制台

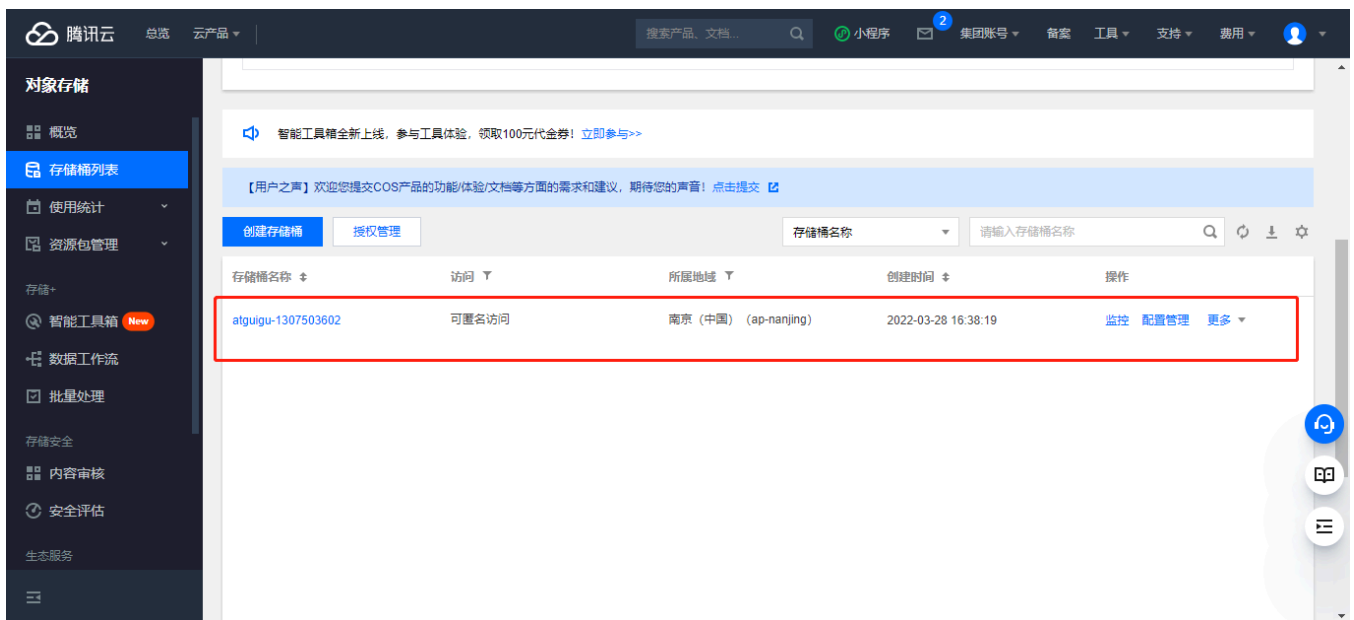


1.2、创建 Bucket

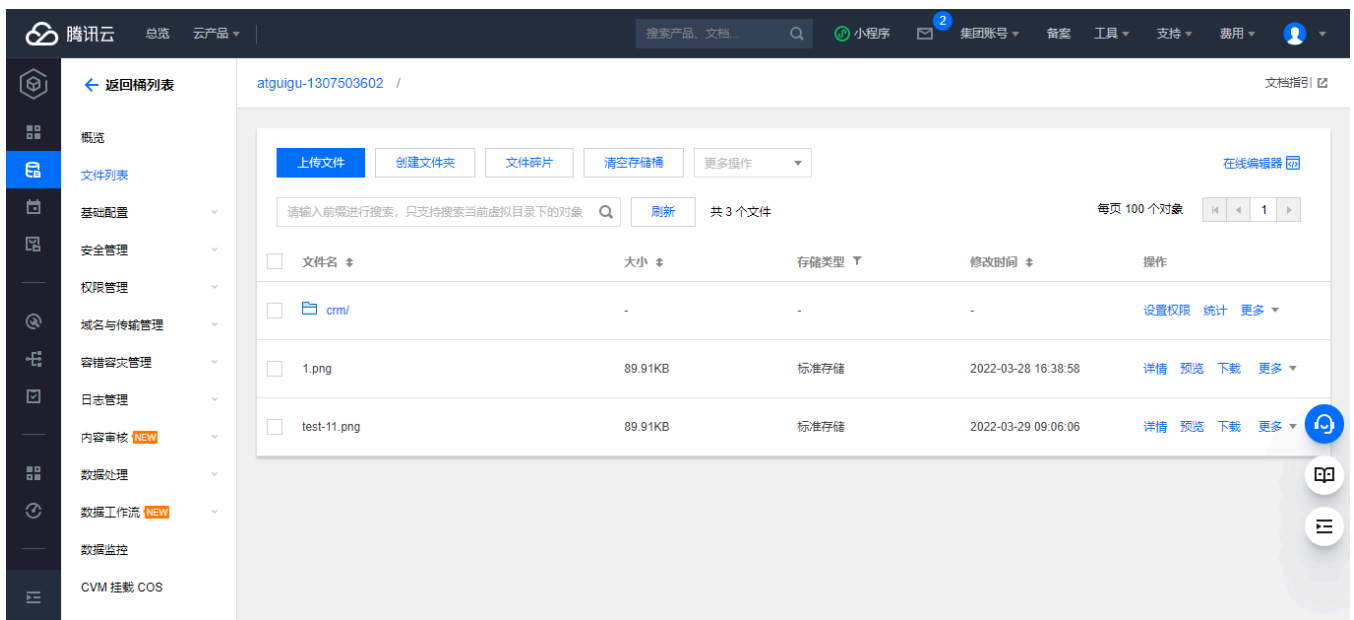
进入管理控制台，找到存储桶列表，创建存储桶



输入桶名称，选择：公有读取，其他默认

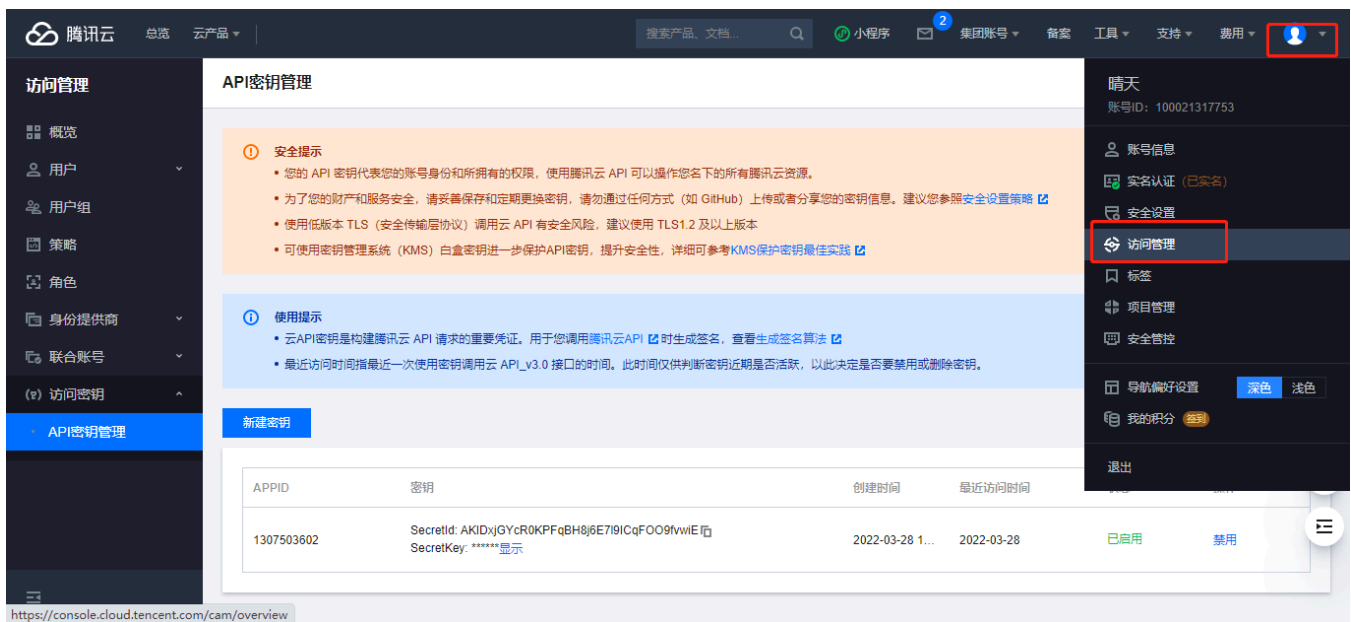


点击桶名称，进入详情页，可测试上传文件

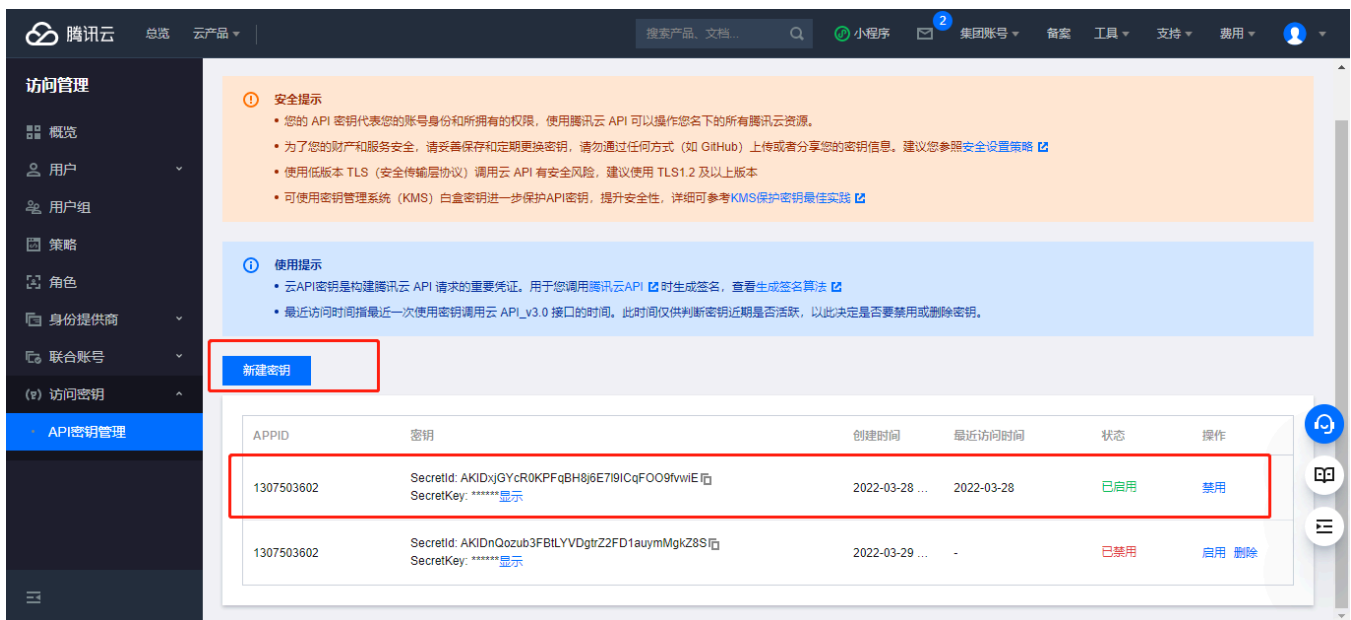


1.3、创建 API 秘钥

进入 API 秘钥管理



新建密钥



1.4、快速入门

参考文档：<https://cloud.tencent.com/document/product/436/10199>

引入依赖

```
<dependency>
  <groupId>com.qcloud</groupId>
  <artifactId>cos_api</artifactId>
  <version>5.6.54</version>
</dependency>
```

测试上传

```

import com.alibaba.fastjson.JSON;
import com.qcloud.cos.COSClient;
import com.qcloud.cos.ClientConfig;
import com.qcloud.cos.auth.BasicCOSCredentials;
import com.qcloud.cos.auth.COSCredentials;
import com.qcloud.cos.exception.CosClientException;
import com.qcloud.cos.exception.CosServiceException;
import com.qcloud.cos.http.HttpProtocol;
import com.qcloud.cos.model.*;
import com.qcloud.cos.region.Region;

import java.io.File;

public class FileTest {

    public static void main(String[] args) {
        // 1 初始化用户身份信息 (secretId, secretKey) 。
        // SECRETID和SECRETKEY请登录访问管理控制台 https://console.cloud.tencent.com/cam/capi 进行查
        String secretId = "你的secretId";
        String secretKey = "你的secretKey";
        COSCredentials cred = new BasicCOSCredentials(secretId, secretKey);
        // 2 设置 bucket 的地域, COS 地域的简称请参照 https://cloud.tencent.com/document/product/436
        // clientConfig 中包含了设置 region, https(默认 http), 超时, 代理等 set 方法, 使用可参见源码或
        Region region = new Region("ap-nanjing");
        ClientConfig clientConfig = new ClientConfig(region);
        // 这里建议设置使用 https 协议
        // 从 5.6.54 版本开始, 默认使用了 https
        clientConfig.setHttpProtocol(HttpProtocol.https);
        // 3 生成 cos 客户端。
        COSClient cosClient = new COSClient(cred, clientConfig);

        try{
            // 指定要上传的文件
            File localFile = new File("D:\\glkt_work\\glkt\\11.png");
            // 指定文件将要存放的存储桶
            String bucketName = "你的bucketName";
            // 指定文件上传到 COS 上的路径, 即对象键。例如对象键为folder/picture.jpg, 则表示将文件 pict

```

```

        String key = "test-11.png";
        PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, key, localFile);
        PutObjectResult putObjectResult = cosClient.putObject(putObjectRequest);
        System.out.println(JSON.toJSONString(putObjectResult));
    } catch (Exception clientException) {
        clientException.printStackTrace();
    }
}
}

```

2、整合腾讯云对象存储

2.1、service_vod 模块引入依赖

```

<dependencies>
    <!-- 腾讯云COS依赖 -->
    <dependency>
        <groupId>com.qcloud</groupId>
        <artifactId>cos_api</artifactId>
        <version>5.6.54</version>
    </dependency>
    <!-- 日期工具栏依赖 -->
    <dependency>
        <groupId>joda-time</groupId>
        <artifactId>joda-time</artifactId>
    </dependency>
</dependencies>

```

2.2、配置 application.properties

添加如下内容：


```
spring.servlet.multipart.max-file-size=1024MB  
spring.servlet.multipart.max-request-size=1024MB
```

不同的服务器，地址不同

```
tencent.cos.file.region=ap-beijing
```

```
tencent.cos.file.secretid=你的id
```

```
tencent.cos.file.secretkey=你的key
```

bucket可以在控制台创建，也可以使用java代码创建

```
tencent.cos.file.bucketname=你的bucketName
```

3.3、创建工具类

```
/**
 * 常量类，读取配置文件application.properties中的配置
 */
@Component
public class ConstantPropertiesUtil implements InitializingBean {

    @Value("${tencent.cos.file.region}")
    private String region;

    @Value("${tencent.cos.file.secretid}")
    private String secretId;

    @Value("${tencent.cos.file.secretkey}")
    private String secretKey;

    @Value("${tencent.cos.file.bucketname}")
    private String bucketName;

    public static String END_POINT;
    public static String ACCESS_KEY_ID;
    public static String ACCESS_KEY_SECRET;
    public static String BUCKET_NAME;

    @Override
    public void afterPropertiesSet() throws Exception {
        END_POINT = region;
        ACCESS_KEY_ID = secretId;
        ACCESS_KEY_SECRET = secretKey;
        BUCKET_NAME = bucketName;
    }
}
```

3.4、创建 Service

创建 Interface : FileService.java

```
public interface FileService {  
    //文件上传  
    String upload(MultipartFile file);  
}
```

实现：FileServiceImpl.java

[文档中心](#) [入门中心](#) [API 中心](#) [SDK 中心](#) [我的文档中心](#)

对象存储

Go SDK

iOS SDK

Java SDK

快速入门

存储桶操作

对象操作

上传对象

下载对象

复制与移动对象

列出对象

删除对象

判断对象是否存在

查询对象元数据

修改对象元数据

versionId	当存储桶开启了版本控制功能，返回对象的版本号 Id
crc64Ecma	服务端根据对象内容计算出来的 CRC64

上传流类型

上传的源是一个 InputStream 类型（和其子类型）的流实例。

方法原型

```
// 上传对象  
public Upload upload(final PutObjectRequest putObjectRequest)  
    throws CosServiceException, CosClientException;
```

请求示例

```
// 使用高级接口必须先保证本进程存在一个 TransferManager 实例，如果没有则创  
// 详细代码参见本页：高级接口 -> 创建 TransferManager  
TransferManager transferManager = createTransferManager();
```

```

@Service
public class FileServiceImpl implements FileService {
    @Override
    public String upload(MultipartFile file) {
        // Endpoint以杭州为例，其它Region请按实际情况填写。
        String endpoint = ConstantPropertiesUtil.END_POINT;

        String bucketName = ConstantPropertiesUtil.BUCKET_NAME;
        // 1 初始化用户身份信息 (secretId, secretKey) 。
        String secretId = ConstantPropertiesUtil.ACCESS_KEY_ID;
        String secretKey = ConstantPropertiesUtil.ACCESS_KEY_SECRET;
        COSCredentials cred = new BasicCOSCredentials(secretId, secretKey);

        // 2 设置 bucket 的地域
        // clientConfig 中包含了设置 region, https(默认 http),超时,代理等 set 方法
        Region region = new Region(ConstantPropertiesUtil.END_POINT);
        ClientConfig clientConfig = new ClientConfig(region);
        // 这里建议设置使用 https 协议
        // 从 5.6.54 版本开始，默认使用了 https
        clientConfig.setHttpProtocol(HttpProtocol.https);
        // 3 生成 cos 客户端。
        COSClient cosClient = new COSClient(cred, clientConfig);

        try{
            // 指定要上传的文件
            InputStream inputStream = file.getInputStream();
            // 指定文件将要存放的存储桶
            // 指定文件上传到 COS 上的路径，即对象键。例如对象键为folder/picture.jpg，则表示将文件 pict
            String key = UUID.randomUUID().toString().replaceAll("-", "") +
                file.getOriginalFilename();
            String dateUrl = new DateTime().toString("yyyy/MM/dd");
            key = dateUrl+"/"+key;

            ObjectMetadata objectMetadata = new ObjectMetadata();
            PutObjectRequest putObjectRequest =
                new PutObjectRequest(bucketName, key, inputStream, objectMetadata);
            PutObjectResult putObjectResult = cosClient.putObject(putObjectRequest);
        }
    }
}

```

```

        System.out.println(JSON.toJSONString(putObjectResult));
        //https://ggkt-atguigu-1310644373.cos.ap-beijing.myqcloud.com/01.jpg
        String url = "https://" + bucketName + "." + "cos" + "." + endpoint + ".myqcloud.com" + "/" + key;
        return url;
    } catch (Exception clientException) {
        clientException.printStackTrace();
        return null;
    }
}
}
}

```

3.5、创建 Controller

FileUploadController.java

```

@Api(tags = "文件上传接口")
@RestController
@RequestMapping("/admin/vod/file")
public class FileUploadController {
    @Autowired
    private FileService fileService;
    /**
     * 文件上传
     */
    @ApiOperation(value = "文件上传")
    @PostMapping("upload")
    public Result upload(
        @ApiParam(name = "file", value = "文件", required = true)
        @RequestParam("file") MultipartFile file) {
        String uploadUrl = fileService.upload(file);
        return Result.ok(uploadUrl).message("文件上传成功");
    }
}
}

```

3、添加讲师前端完善

3.1、添加上传组件

操作 teacher 目录下的 form.vue 页面

```
<!-- 讲师头像 -->
<el-form-item label="讲师头像">
  <el-upload
    :show-file-list="false"
    :on-success="handleAvatarSuccess"
    :before-upload="beforeAvatarUpload"
    :on-error="handleAvatarError"
    :action="BASE_API+'/admin/vod/file/upload?module=avatar'"
    class="avatar-uploader"
  >
    
    <i v-else class="el-icon-plus avatar-uploader-icon" />
  </el-upload>
</el-form-item>
```

3.2、添加上传方法

初始化访问路径

```
BASE_API: 'http://localhost:8301',
```

添加上传操作方法

```
// 上传成功回调
handleAvatarSuccess(res, file) {
  // console.log(res)
  if (res.code==200) {
    // console.log(res)
    this.teacher.avatar = res.data
    // 强制重新渲染
    this.$forceUpdate()
  } else {
    this.$message.error('上传失败 (非0) ')
  }
},

// 错误处理
handleAvatarError() {
  console.log('error')
  this.$message.error('上传失败 (http失败) ')
},


// 上传校验
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、后台管理系统-课程分类管理模块

1、课程分类管理模块需求

(1) 课程分类列表功能

 Dashboard / 课程分类管理 / 课程分类列表

名称	创建时间
√ 后端开发	2019-09-29 15:47:25
Java	2019-09-29 15:47:25
√ 前端开发	2019-09-29 15:47:25
JavaScript	2019-09-29 15:47:25
> 数据库	2019-09-29 15:47:25
> 大数据	2019-09-29 15:47:25

(2) 课程分类导入功能

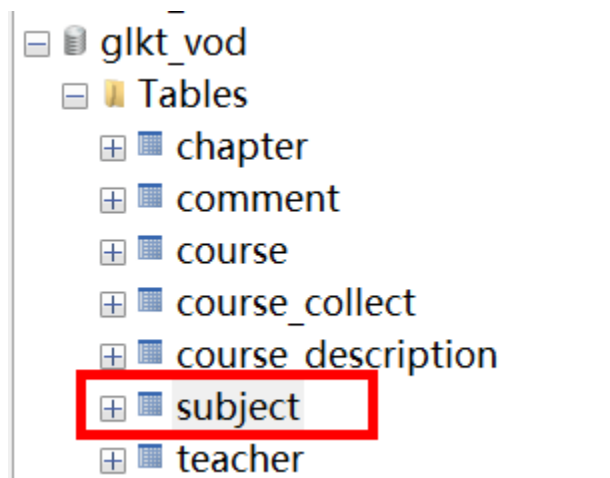
导出 导入

(3) 课程分类导出功能



2、课程分类数据库设计

(1) 创建课程分类表 subject



(2) 课程分类表结构分析




Table: subject

Calculate Optimal Datatypes

Find the optimal datatypes for this table by reading existing data. [Read more](#)

Column Information

Field	Type	Comment
 id	bigint(20) unsigned NOT NULL	主键
title	varchar(10) NOT NULL	类别名称
parent_id	bigint(20) NOT NULL	父ID
sort	int(10) unsigned NOT NULL	排序字段
create_time	timestamp NOT NULL	创建时间
update_time	timestamp NOT NULL	
is_deleted	tinyint(3) NOT NULL	

3、功能实现-课程分类列表

3.1、接口实现分析

课程分类采用树形展示，我们使用“树形数据与懒加载”的方式展现数据列表，因此需要提供的接口如下：根据上级 id 获取下级数据，参考 element-ui 文档：<https://element.eleme.cn/#/zh-CN/component/table>，页面搜索：树形数据与懒加载

√ 2016-05-01	王小虎	上海市普陀区金沙江路 1519 弄
2016-05-01	王小虎	上海市普陀区金沙江路 1519 弄
2016-05-01	王小虎	上海市普陀区金沙江路 1519 弄
2016-05-03	王小虎	上海市普陀区金沙江路 1516 弄

支持树类型的数据的显示。当 row 中包含 children 字段时，被视为树形数据。渲染树形数据时，必须要指定 row-key。支持子节点数据异步加载。设置 Table 的 lazy 属性为 true 与加载函数 load。通过指定 row 中的 hasChildren 字段来指定哪些行是包含子节点。children 与 hasChildren 都可以通过 tree-props 配置。

3.2、编写 SubjectController

```
@Api(tags = "课程分类管理")
@RestController
@RequestMapping(value="/admin/vod/subject")
//@CrossOrigin
public class SubjectController {

    @Autowired
    private SubjectService subjectService;

    //查询下一层课程分类
    //根据parent_id
    @ApiOperation("查询下一层的课程分类")
    @GetMapping("getChildSubject/{id}")
    public Result getChildSubject(@PathVariable Long id) {
        List<Subject> list = subjectService.selectList(id);
        return Result.ok(list);
    }
}
```

3.3、编写 SubjectService

```
public interface SubjectService extends IService<Subject> {

    //查询下一层课程分类
    List<Subject> findChildSubject(Long id);
}
```

3.4、编写 SubjectServiceImpl

```
@Service
public class SubjectServiceImpl extends ServiceImpl<SubjectMapper, Subject> implements SubjectService {

    //查询下一层课程分类
    @Override
    public List<Subject> selectList(Long id) {
        QueryWrapper<Subject> wrapper = new QueryWrapper<>();
        wrapper.eq("parent_id",id);
        List<Subject> subjectList = baseMapper.selectList(wrapper);
        //向list集合每个Subject对象中设置hasChildren
        for (Subject subject:subjectList) {
            Long subjectId = subject.getId();
            boolean isChild = this.isChildren(subjectId);
            subject.setHasChildren(isChild);
        }
        return subjectList;
    }
    //判断id下面是否有子节点
    private boolean isChildren(Long id) {
        QueryWrapper<Subject> wrapper = new QueryWrapper<>();
        wrapper.eq("parent_id",id);
        Integer count = baseMapper.selectCount(wrapper);
        return count>0;
    }
}
```

3.5、开发课程分类列表前端

(1) 添加数据字典路由

修改 router/index.js 文件

```

{
  path: '/subject',
  component: Layout,
  redirect: '/subject/list',
  name: '课程分类管理',
  alwaysShow: true,
  meta: { title: '课程分类管理', icon: 'example' },
  children: [
    {
      path: 'list',
      name: '课程分类列表',
      component: () => import('@/views/vod/subject/list'),
      meta: { title: '课程分类列表', icon: 'table' }
    }
  ]
},

```

(2) 定义数据字典列表接口

创建文件 `src/api/vod/subject.js`

```

import request from "@/utils/request";

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

export default {
  getChildList(id) {
    return request({
      url: `${api_name}/getChildSubject/${id}`,
      method: "get",
    });
  },
};

```

(3) 编写 `subject/list.vue`

```
<template>
  <div class="app-container">
    <el-table
      :data="list"
      style="width: 100%"
      row-key="id"
      border
      lazy
      :load="load"
      :tree-props="{ children: 'children', hasChildren: 'hasChildren' }"
    >
      <el-table-column prop="title" label="名称" width="150"> </el-table-column>
      <el-table-column prop="createTime" label="创建时间"> </el-table-column>
    </el-table>
  </div>
</template>
```

```
<script>
import subjectApi from "@api/vod/subject";
export default {
  data() {
    return {
      list: [], //数据字典列表数组
    };
  },
  created() {
    this.getSubList(0);
  },
  methods: {
    //数据字典列表
    getSubList(id) {
      subjectApi.getChildList(id).then((response) => {
        this.list = response.data;
      });
    },
    load(tree, treeNode, resolve) {
      subjectApi.getChildList(tree.id).then((response) => {
```

```
        resolve(response.data);
    });
},
},
};
</script>
```

4、技术点-EasyExcel

4.1、EasyExcel 介绍

EasyExcel 是阿里巴巴开源的一个 excel 处理框架，**以使用简单、节省内存著称**。EasyExcel 能大大减少占用内存的主要原因是在解析 Excel 时没有将文件数据一次性全部加载到内存中，而是从磁盘上一行行读取数据，逐个解析。

4.2、EasyExcel 特点

- Java 领域解析、生成 Excel 比较有名的框架有 Apache poi、jxl 等。但他们都存在一个严重的问题就是非常的耗内存。如果系统并发量不大的话可能还行，但是一旦并发上来后一定会 OOM 或者 JVM 频繁的 Full GC。
- **EasyExcel 采用一行一行的解析模式**，并将一行的解析结果以观察者的模式通知处理（AnalysisEventListener）。
- EasyExcel 是一个基于 Java 的简单、省内存的读写 Excel 的开源项目。在尽可能节约内存的情况下支持读写百 MB 的 Excel。

4.3、EasyExcel 写操作

(1) pom 中引入 xml 相关依赖

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/com.alibaba/easyexcel -->
    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>easyexcel</artifactId>
        <version>2.1.1</version>
    </dependency>
</dependencies>
```

(2) 创建实体类

设置表头和添加的数据字段

```
@Data
public class Stu {
    //设置表头名称
    @ExcelProperty("学生编号")
    private int sno;
    //设置表头名称
    @ExcelProperty("学生姓名")
    private String sname;
}
```

(3) 实现写操作

创建方法循环设置要添加到 Excel 的数据

```
//循环设置要添加的数据，最终封装到list集合中
private static List<Stu> data() {
    List<Stu> list = new ArrayList<Stu>();
    for (int i = 0; i < 10; i++) {
        Stu data = new Stu();
        data.setSno(i);
        data.setSname("张三"+i);
        list.add(data);
    }
    return list;
}
```

实现最终的添加操作


```

public static void main(String[] args) throws Exception {
    // 写法1
    String fileName = "F:\\11.xlsx";
    // 这里 需要指定写用哪个class去写, 然后写到第一个sheet, 名字为模板 然后文件流会自动关闭
    // 如果这里想使用03 则 传入excelType参数即可
    EasyExcel.write(fileName, DemoData.class).sheet("写入方法").doWrite(data());
}

```

4.4、EasyExcel 读操作

(1) 创建实体类

```

@Data
public class Stu {
    //设置表头名称
    //设置列对应的属性
    @ExcelProperty(value = "学生编号", index = 0)
    private int sno;
    //设置表头名称
    //设置列对应的属性
    @ExcelProperty(value = "学生姓名", index = 1)
    private String sname;
}

```

(2) 创建读取操作的监听器

```

public class Excellistener extends AnalysisEventListener<Stu> {
    //创建list集合封装最终的数据
    List<Stu> list = new ArrayList<Stu>();
    //一行一行去读取excle内容
    @Override
    public void invoke(Stu user, AnalysisContext analysisContext) {
        System.out.println("****"+user);
        list.add(user);
    }
    //读取excel表头信息
    @Override
    public void invokeHeadMap(Map<Integer, String> headMap, AnalysisContext context) {
        System.out.println("表头信息: "+headMap);
    }
    //读取完成后执行
    @Override
    public void doAfterAllAnalysed(AnalysisContext analysisContext) {
    }
}

```

(3) 调用实现最终的读取

```

public static void main(String[] args) throws Exception {
    String fileName = "F:\\11.xlsx";
    // 这里 需要指定读用哪个class去读, 然后读取第一个sheet 文件流会自动关闭
    EasyExcel.read(fileName, ReadData.class, new Excellistener()).sheet().doRead();
}

```

5、功能实现-课程分类导出

5.1、查看 model 实体类

在 model 模块查看实体：com.atguigu.ggkt.vo.vod.SubjectEeVo

```

@Data
public class SubjectEeVo {

    @ExcelProperty(value = "id" ,index = 0)
    private Long id;

    @ExcelProperty(value = "课程分类名称" ,index = 1)
    private String title;

    @ExcelProperty(value = "上级id" ,index = 2)
    private Long parentId;

    @ExcelProperty(value = "排序" ,index = 3)
    private Integer sort;
}

```

5.2、编写 SubjectService 和实现

SubjectService

```

public interface SubjectService extends IService<Subject> {

    //查询下一层课程分类
    List<Subject> selectList(Long id);

    /**
     * 导出
     * @param response
     */
    void exportData(HttpServletResponse response);
}

```

SubjectServiceImpl

```

//课程分类导出
@Override
public void exportData(HttpServletResponse response) {
    try {
        response.setContentType("application/vnd.ms-excel");
        response.setCharacterEncoding("utf-8");
        // 这里URLCoder.encode可以防止中文乱码 当然和easyexcel没有关系
        String fileName = URLEncoder.encode("课程分类", "UTF-8");
        response.setHeader("Content-disposition", "attachment;filename="+ fileName + ".xlsx");
        List<Subject> dictList = baseMapper.selectList(null);
        List<SubjectEeVo> dictVoList = new ArrayList<>(dictList.size());
        for(Subject dict : dictList) {
            SubjectEeVo dictVo = new SubjectEeVo();
            BeanUtils.copyProperties(dict,dictVo);
            dictVoList.add(dictVo);
        }
        EasyExcel.write(response.getOutputStream(), SubjectEeVo.class).sheet("课程分类").doWrite(d
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```

5.3、添加 Controller 方法

```

@ApiOperation(value="导出")
@GetMapping(value = "/exportData")
public void exportData(HttpServletResponse response) {
    subjectService.exportData(response);
}

```

5.4、数据字典导出前端

(1) list.vue 页面添加导出按钮

```

<div class="el-toolbar">
  <div class="el-toolbar-body" style="justify-content: flex-start;">
    <el-button type="text" @click="exportData">
      <i class="fa fa-plus" /> 导出</el-button>
    </div>
  </div>
</div>

```

(2) 编写调用方法

```

exportData() {
  window.open("http://localhost:8301/admin/vod/subject/exportData")
},

```

6、功能实现-课程分类导入

6.1、创建读取监听器

```

@Component
public class SubjectListener extends AnalysisEventListener<SubjectEeVo> {
    @Autowired
    private SubjectMapper dictMapper;
    //一行一行读取
    @Override
    public void invoke(SubjectEeVo subjectEeVo, AnalysisContext analysisContext) {
        //调用方法添加数据库
        Subject subject = new Subject();
        BeanUtils.copyProperties(subjectEeVo,subject);
        dictMapper.insert(subject);
    }
    @Override
    public void doAfterAllAnalysed(AnalysisContext analysisContext) {
    }
}

```

6.2、添加 controller 方法

```
@ApiOperation(value = "导入")
@PostMapping("importData")
public Result importData(MultipartFile file) {
    subjectService.importDictData(file);
    return Result.ok();
}
```

6.3、添加 service 方法

```
@Autowired
private SubjectListener subjectListener;

//导入
@Override
public void importData(MultipartFile file) {
    try {
        EasyExcel.read(file.getInputStream(),
            SubjectEeVo.class, subjectListener).sheet().doRead();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

6.4、数据字典导入前端

(1) 在 list.vue 页面添加导入按钮

```
<el-button type="text" @click="importData">
  <i class="fa fa-plus" /> 导入</el-button>
</el-button>
```

(2) 添加导入弹出层

```

<el-dialog title="导入" :visible.sync="dialogImportVisible" width="480px">
  <el-form label-position="right" label-width="170px">
    <el-form-item label="文件">
      <el-upload
        :multiple="false"
        :on-success="onUploadSuccess"
        :action="'http://localhost:8333/admin/vod/subject/importData'"
        class="upload-demo"
      >
        <el-button size="small" type="primary">点击上传</el-button>
        <div slot="tip" class="el-upload__tip">
          只能上传xls文件, 且不超过500kb
        </div>
      </el-upload>
    </el-form-item>
  </el-form>
  <div slot="footer" class="dialog-footer">
    <el-button @click="dialogImportVisible = false">取消</el-button>
  </div>
</el-dialog>

```

(3) 添加导入弹出层属性

```

data() {
  return {
    dialogImportVisible: false,
    list:[] //数据字典列表数组
  }
},

```

(4) 添加导入方法

```
importData() {  
    this.dialogImportVisible = true  
},  
onUploadSuccess(response, file) {  
    this.$message.info('上传成功')  
    this.dialogImportVisible = false  
    this.getSubList(0)  
},
```