

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 高级程序设计 |
| 姓 名： | 童芳慧 |
| 学 院： | 计算机科学与技术学院 |
| 系： | 计算机 |
| 专 业： | 计算机科学与技术 |
| 学 号： | 3210102034 |
| 指导教师： | 耿晨歌 |

2023年 1月 5日

**浙江大学实验报告**

课程名称： 高级程序设计 实验类型： 综合

实验项目名称： 作业3 Springboot练习

学生姓名：童芳慧 专业： 计算机科学与技术 学号： 3210102034

同组学生姓名： 指导老师： 耿晨歌

实验地点： 实验日期： 2023 年 1 月 5 日

1. 设计内容

设计能够实现文档上传下载的文档管理系统。

1. 代码解析

**前端部分：**

登录界面：

输入用户名，密码进行登录。

登录完成后，用户信息保存在浏览器中以备后续使用。

<template>  
 <div class="loginBody">  
 <div class="loginDiv">  
 <div class="login-content">  
 <h1 class="login-title">用户登录</h1>  
 <el-form :model="loginForm" label-width="100px"  
 :rules="rules" ref="loginForm">  
 <el-form-item label="账号" prop="no">  
 <el-input style="width: 200px" type="text" v-model="loginForm.no"  
 autocomplete="off" size="small"></el-input>  
 </el-form-item>  
 <el-form-item label="密码" prop="password">  
 <el-input style="width: 200px" type="password" v-model="loginForm.password"  
 show-password autocomplete="off" size="small" @keyup.enter.native="confirm"></el-input>  
 </el-form-item>  
 <el-form-item>  
 <el-button type="primary" @click="confirm" :disabled="confirm\_disabled">确 定</el-button>  
 </el-form-item>  
 </el-form>  
 </div>  
 </div>  
 </div>  
</template>  
  
  
<script>  
 export default {  
 name: "Login",  
 data(){  
 return{  
 confirm\_disabled:false,  
 loginForm:{  
 no:'',  
 password:''  
 },  
 rules:{  
 no: [  
 { required: true, message: '请输入账号', trigger: 'blur' }  
 ],  
 password: [  
 { required: true, message: '请输密码', trigger: 'blur' }  
 ],  
 }  
 }  
 },  
 methods:{  
  
 confirm(){  
 ***console***.log("confirm")  
 this.confirm\_disabled=true;  
 this.$refs.loginForm.validate((valid) => {  
 if (valid) { //valid成功为true，失败为false  
 //去后台验证用户名密码  
 this.$axios.post(this.$httpUrl+'/user/login',this.loginForm).then(res=>res.data).then(res=>{  
 ***console***.log(res)  
 if(res.code==1){  
 //存储  
 ***sessionStorage***.setItem("CurUser",***JSON***.stringify(res.data.user))  
  
 ***console***.log(res.data.menu)  
 this.$store.commit("setMenu",res.data.menu)  
 //跳转到主页  
 this.$router.replace('/Index');  
 }else{  
 this.confirm\_disabled=false;  
 alert('校验失败，用户名或密码错误！');  
 return false;  
 }  
 });  
 } else {  
 this.confirm\_disabled=false;  
 ***console***.log('校验失败');  
 return false;  
 }  
 });  
  
 }  
 }  
 }  
</script>

注册界面

输入账号，名字，密码进行注册。

前端会在后台对账号和名字的重复性进行验证。

<template>  
 <div class="loginBody">  
 <div class="loginDiv">  
 <div class="login-content">  
 <h1 class="login-title">用户注册</h1>  
 <el-form :model="registerForm" label-width="100px"  
 :rules="rules" ref="registerForm">  
 <el-form-item label="账号" prop="no">  
 <el-input style="width: 200px" type="text" v-model="registerForm.no"  
 autocomplete="off" size="small"></el-input>  
 </el-form-item>  
 <el-form-item label="名字" prop="name">  
 <el-input style="width: 200px" type="text" v-model="registerForm.name"  
 autocomplete="off" size="small"></el-input>  
 </el-form-item>  
 <el-form-item label="密码" prop="password">  
 <el-input style="width: 200px" type="password" v-model="registerForm.password"  
 show-password autocomplete="off" size="small" ></el-input>  
 </el-form-item>  
  
 <el-form-item>  
 <el-button type="primary" @click="cancel" :disabled="confirm\_disabled">取 消</el-button>  
 <el-button type="primary" @click="register" :disabled="confirm\_disabled">注 册</el-button>  
 </el-form-item>  
 </el-form>  
 </div>  
 </div>  
 </div>  
</template>  
  
  
<script>  
export default {  
 name: "register",  
 data(){  
 let checkDuplicateNo =(rule,value,callback)=>{  
 this.$axios.get(this.$httpUrl+"/user/findByNo?no="+this.registerForm.no).then(res=>res.data).then(res=>{  
 if(res.code!=1){  
 callback()  
 }else{  
 callback(new Error('账号已经存在'));  
 }  
 })  
 };  
 let checkDuplicateName =(rule,value,callback)=>{  
 this.$axios.get(this.$httpUrl+"/user/findByName?name="+this.registerForm.name).then(res=>res.data).then(res=>{  
 if(res.code!=1){  
 callback()  
 }else{  
 callback(new Error('名字已经存在'));  
 }  
 })  
 };  
 return{  
 confirm\_disabled:false,  
 registerForm:{  
 no:'',  
 name:'',  
 password:''  
 },  
 rules:{  
 no: [  
 { required: true, message: '请输入账号', trigger: 'blur' },  
 {validator:checkDuplicateNo, trigger: 'blur'}  
 ],  
 name: [  
 { required: true, message: '请输入名字', trigger: 'blur' },  
 {validator:checkDuplicateName, trigger: 'blur'}  
 ],  
 password: [  
 { required: true, message: '请输密码', trigger: 'blur' },  
 {min: 6, max: 20, message: '长度在 6 到 20 个字符', trigger: 'blur'}  
 ]  
 }  
 }  
 },  
 methods:{  
  
 register(){  
 this.confirm\_disabled=true;  
 this.$refs.registerForm.validate((valid) => {  
 if (valid) { //valid成功为true，失败为false  
 //去后台验证用户名密码  
 this.$axios.post(this.$httpUrl+'/user/register',this.registerForm).then(res=>res.data).then(res=>{  
 ***console***.log(res)  
 if(res.code==1){  
 //跳转到登录  
 this.$message({message: '注册成功！', type: 'success'});  
 this.$router.replace('/');  
 }else{  
 this.confirm\_disabled=false;  
 alert('注册失败！请重试');  
 return false;  
 }  
 });  
 } else {  
 this.confirm\_disabled=false;  
 ***console***.log('注册失败');  
 return false;  
 }  
 });  
  
 },  
 cancel(){  
 this.$router.replace('/');  
 }  
 }  
}  
</script>

将界面分为边栏，标题和窗体内容：

窗体主体：

首页文档列表：

动态根据文件名，标签，查询用户拥有的文档，并展示文档文件名，路径，标签。对文档有下载，分享，删除的操作。

下载功能：调用浏览器下载功能下载。分享功能：在输入框中搜索其他用户的用户名选中后即分享成功。

删除：删除该文件。

<template>  
 <div>  
 <div style="margin-bottom: 5px;">  
 <el-input v-model="fileName" placeholder="请输入文件名" suffix-icon="el-icon-search" style="width: 200px;"  
 @keyup.enter.native="loadPost"></el-input>  
  
 <el-select v-model="selectedTags" multiple placeholder="请选择文件标签" style="margin-left: 5px;">  
 <el-option v-for="tag in tags" :key="tag" :label="tag" :value="tag"></el-option>  
 </el-select>  
  
 <el-button type="primary" style="margin-left: 5px;" @click="loadPost">查询</el-button>  
 <el-button type="success" @click="resetParam">重置</el-button>  
 </div>  
 <el-table :data="tableData"  
 :header-cell-style="{ background: '#f2f5fc', color: '#555555' }"  
 border  
 >  
 <!-- <el-table-column prop="id" label="ID" width="60">-->  
 <!-- </el-table-column>-->  
 <el-table-column prop="filename" label="文件名" width="300">  
 </el-table-column>  
 <el-table-column prop="fileurl" label="路径">  
 </el-table-column>  
 <el-table-column prop="label" label="标签">  
 </el-table-column>  
  
 <el-table-column prop="operate" label="操作" width="210">  
 <template slot-scope="scope">  
 <el-button size="small" type="success" @click="downloadResult(scope.row)">下载</el-button>  
 <el-button size="small" type="success" @click="showShareDialog(scope.row)">分享</el-button>  
 <el-popconfirm  
 title="确定删除吗？"  
 @confirm="del(scope.row.id)"  
 style="margin-left: 5px;"  
 >  
 <el-button slot="reference" size="small" type="danger">删除</el-button>  
 </el-popconfirm>  
 </template>  
 </el-table-column>  
 </el-table>  
 <el-pagination  
 @size-change="handleSizeChange"  
 @current-change="handleCurrentChange"  
 :current-page="pageNum"  
 :page-sizes="[5, 10, 20,30]"  
 :page-size="pageSize"  
 layout="total, sizes, prev, pager, next, jumper"  
 :total="total">  
 </el-pagination>  
 <el-dialog :visible.sync="dialogVisible" title="分享" width="20%">  
 <span slot="footer" class="dialog-footer">  
 <el-button @click="dialogVisible = false">取消</el-button>  
 <el-button type="primary" @click="share">确定</el-button>  
 </span>  
 <div>  
 <p>请输入要分享的用户：</p>  
 <el-autocomplete  
 v-model="searchValue"  
 :fetch-suggestions="fetchUserSuggestions"  
 placeholder="请输入用户名"  
 :trigger-on-focus="false"  
 ></el-autocomplete>  
 </div>  
 </el-dialog>  
 </div>  
</template>

<script>  
import fileDownload from "js-file-download";  
import ***axios*** from "axios";  
  
function request(config) {  
 // 创建axios的实例  
 const instance = ***axios***.create({  
 baseURL: 'http://localhost:8090', // 这个是后端的ip地址  
 });  
 return instance(config);  
}  
  
export function getResult(resultfile) {  
 return request({  
 url: 'comp/getResult',  
 method: 'post',  
 responseType: 'arraybuffer',  
 data: {  
 resultfile: resultfile,  
 userId: ***JSON***.parse(***sessionStorage***.getItem('CurUser')).id,  
 }  
 });  
}  
  
export default {  
 name: "Home",  
 data() {  
 return {  
 shareFile:'',  
 selectedTags: [],  
 tags: ["游戏", "学习", "学术", "娱乐", "工作"],  
 searchValue: '',  
 user: ***JSON***.parse(***sessionStorage***.getItem('CurUser')),  
 fileName: '',  
 dialogVisible:false,  
 shareUrl: '',  
 tableData: [],  
 pageNum: 1,  
 pageSize: 10,  
 total: 0,  
 }  
 },  
 methods: {  
 fetchUserSuggestions(queryString, callback) {  
 // 从后台获取用户名列表  
 // 请根据你的实际情况修改下面的请求地址和参数  
 const url = this.$httpUrl + '/user/getUserNameTop10';  
 ***axios*** .get(url, { params: { userName: queryString } })  
 .then(response => {  
 const users = response.data.map(user => ({ value: user, label: user }));  
 callback(users);  
 })  
 .catch(error => {  
 ***console***.error(error);  
 callback([]);  
 });  
 },  
 resetParam() {  
 this.fileName = ''  
 this.selectedTags =[]  
 },  
 downloadResult(row) {  
 ***console***.log("准备下载")  
 ***console***.log(row)  
 getResult(row.filename).then(res => {  
 fileDownload(res.data, row.filename); // 使用js-file-download来接收二进制流然后下载  
 }).catch(err => {  
 this.$message.error(err);  
 });  
 },  
 del(id) {  
 this.$axios.get(this.$httpUrl + '/comp/del?id=' + id).then(res => res.data).then(res => {  
 ***console***.log(res)  
 if (res.code == 1) {  
  
 this.$message({  
 message: '操作成功！',  
 type: 'success'  
 });  
 this.loadPost()  
 } else {  
 this.$message({  
 message: '操作失败！',  
 type: 'error'  
 });  
 }  
  
 })  
 },  
 showShareDialog(row) {  
 this.shareFile = row  
 this.searchValue = ''  
 this.dialogVisible = true  
 },  
 share() {  
 // 在这里执行分享操作  
 ***console***.log(this.shareFile)  
 this.shareFile.id = null  
 this.$set(this.shareFile, 'userName', this.searchValue)  
 this.$axios.post(this.$httpUrl + '/comp/saveShare', this.shareFile).then(res => res.data).then(res => {  
 ***console***.log(res)  
 this.$message.success("分享成功")  
 this.searchValue = ''  
 this.shareFile = ''  
 })  
 this.dialogVisible = false;  
 },  
 searchUsers() {  
 if (this.shareUserName.trim()) {  
 this.$axios.get(`${this.$httpUrl}/user/search?name=${this.shareUserName.trim()}`).then(res => res.data).then(res => {  
 if (res && res.length > 0) {  
 const userList = res;  
 this.$autocomplete('请选择要分享的用户', userList, {  
 confirmButtonText: '确定',  
 cancelButtonText: '取消',  
 }).then(({ value }) => {  
 // 用户选择后更新选中的用户ID  
 this.selectedUserId = value;  
 }).catch(() => {  
 this.selectedUserId = null; // 用户取消选择时清空选中的用户ID  
 });  
 } else {  
 this.selectedUserId = null; // 未找到匹配用户时清空选中的用户ID  
 // this.$message.warning('未找到匹配的用户');  
 }  
 });  
 }  
 },  
 loadPost() {  
 this.$axios.post(this.$httpUrl + '/comp/listPageC1', {  
 pageSize: this.pageSize,  
 pageNum: this.pageNum,  
 param: {  
 userId: this.user.id,  
 fileName: this.fileName,  
 label: this.selectedTags  
 }  
 }).then(res => res.data).then(res => {  
 ***console***.log(res)  
 if (res.code == 1) {  
 this.tableData = res.data  
 ***console***.log(this.tableData)  
 this.total = res.total  
 } else {  
 alert('获取数据失败')  
 }  
  
 })  
 },  
 handleSizeChange(val) {  
 ***console***.log(`每页 ${val} 条`);  
 this.pageNum = 1  
 this.pageSize = val  
 this.loadPost()  
 },  
 handleCurrentChange(val) {  
 ***console***.log(`当前页: ${val}`);  
 this.pageNum = val  
 this.loadPost()  
 },  
 },  
 mounted() {  
 this.loadPost()  
 }  
  
}  
</script>

上传界面：

一次上传一个文件，支持拖拽或在文件管理器中选择上传。

<script>  
import ***axios*** from 'axios'  
  
  
  
export function uploadResult(){ // 上传文件的方法  
 return ***axios***.create({  
 withCredentials: true,  
 });  
}  
  
export default {  
 name: "File",  
 data() {  
  
 return {  
 dialogVisible: false,  
 tags: ["游戏", "学习", "学术", "娱乐", "工作"], // 可选择的标签  
 selectedTags: [],  
 user : ***JSON***.parse(***sessionStorage***.getItem('CurUser')),  
 fileList: [],  
 }  
 },  
 computed:{  
  
 },  
 methods:{  
 showDialog() {  
 // 检验用户是否未选取文件就上传  
 if(this.fileList.length === 0){  
 this.$message.warning('请添加文件');  
 return  
 }  
 this.selectedTags = []  
 this.dialogVisible = true;  
 },  
 confirmSelection() {  
 ***console***.log("已选择的标签:", this.selectedTags);  
 this.dialogVisible = false;  
 this.submitUpload()  
 },  
 fileChange(file, fileList) {  
 // 因为是限制为只能上传一个文件，所以前端选取文件时，如果选取第二个文件时，要把第一个文件给覆盖掉。  
 this.fileList = [fileList[fileList.length - 1]];  
 // 也可以这样写：  
 // this.fileList = [file];  
 },  
 reset(){  
 // 当对话框关闭时，清空已上传的文件列表，同时fileList重新设置为空数组  
 this.$refs.upload.clearFiles(); // 这个是el-upload自带的清空文件列表方法  
 this.fileList = [];  
 },  
 uploadHttpRequest(){  
 const formData = new FormData();  
 formData.append('file', this.fileList[0].raw); // 要上传的文件  
 formData.append('userId', this.user.id); // 用户id  
 const label = this.selectedTags.join(',');  
 formData.append('label', label);  
 // ${server.url}/comp/uploadResult 这个是后端接收的接口  
 // withCredentials: true一定要设置，否则可能会上传不成功，这是个坑  
 ***console***.log("准备上传")  
 uploadResult().post(this.$httpUrl+'/comp/uploadResult', formData).then(res => res.data) .then(res =>{  
 ***console***.log(res)  
 if(res.code === 1){  
 this.$message.success('上传成功');  
 this.fileList = []; // 上传成功就初始化文件列表  
 }else{  
 this.$message.error(); // 上传失败就输出失败信息  
 }  
 }).catch(err =>{  
 this.$message.error(err);  
 });  
 },  
 // 自定义上传功能  
 submitUpload(){  
 this.$refs.upload.submit(); // 这个会触发upload的http-request，手动上传文件  
 },  
  
  
  
 }  
}  
</script>  
  
<template>  
 <div style="text-align: center; margin-top:60px">  
<!-- <el-dialog width="400px" top="8vh" title="上传文件" @close="closeUpload" :visible.sync="upload\_visible">-->  
 <el-upload :file-list="fileList"  
 :auto-upload="false"  
 :http-request="uploadHttpRequest"  
 class="upload-demo"  
 :on-change="fileChange"  
 ref="upload"  
 drag  
 action="#">  
 <i class="el-icon-upload"/>  
 <div class="el-upload\_\_text">将文件拖到此处，或<em>点击选取文件</em></div>  
 <div slot="tip" class="el-upload\_\_tip">一次只能上传一个文件</div>  
 </el-upload>  
 <el-button type="danger" @click="reset" style="margin-left: 30px; margin-top: 20px">清空列表</el-button>  
 <el-button type="success" @click="showDialog" style="margin-left: 30px; margin-top: 20px">上传</el-button>  
<!-- </el-dialog>-->  
  
 <el-dialog :visible.sync="dialogVisible" title="请选择文件标签">  
 <el-checkbox-group v-model="selectedTags">  
 <el-checkbox v-for="tag in tags" :label="tag" :key="tag">{{ tag }}</el-checkbox>  
 </el-checkbox-group>  
 <span slot="footer" class="dialog-footer">  
 <el-button @click="dialogVisible = false">取消</el-button>  
 <el-button type="primary" @click="confirmSelection">确定</el-button>  
 </span>  
 </el-dialog>  
 </div>  
</template>

好友分享界面：

可以看到好友分享的文档，可以搜索、下载、删除。

<template>  
 <div>  
 <div style="margin-bottom: 5px;">  
 <el-input v-model="fileName" placeholder="请输入文件名" suffix-icon="el-icon-search" style="width: 200px;"  
 @keyup.enter.native="loadPost"></el-input>  
  
 <el-select v-model="selectedTags" multiple placeholder="请选择文件标签" style="margin-left: 5px;">  
 <el-option v-for="tag in tags" :key="tag" :label="tag" :value="tag"></el-option>  
 </el-select>  
  
 <el-button type="primary" style="margin-left: 5px;" @click="loadPost">查询</el-button>  
 <el-button type="success" @click="resetParam">重置</el-button>  
 </div>  
 <el-table :data="tableData"  
 :header-cell-style="{ background: '#f2f5fc', color: '#555555' }"  
 border  
 >  
 <!-- <el-table-column prop="id" label="ID" width="60">-->  
 <!-- </el-table-column>-->  
 <el-table-column prop="filename" label="文件名" width="300">  
 </el-table-column>  
 <el-table-column prop="fileurl" label="路径">  
 </el-table-column>  
 <el-table-column prop="label" label="标签">  
 </el-table-column>  
  
 <el-table-column prop="operate" label="操作" width="210">  
 <template slot-scope="scope">  
 <el-button size="small" type="success" @click="downloadResult(scope.row)">下载</el-button>  
 <el-popconfirm  
 title="确定删除吗？"  
 @confirm="del(scope.row.id)"  
 style="margin-left: 5px;"  
 >  
 <el-button slot="reference" size="small" type="danger">删除</el-button>  
 </el-popconfirm>  
 </template>  
 </el-table-column>  
 </el-table>  
 <el-pagination  
 @size-change="handleSizeChange"  
 @current-change="handleCurrentChange"  
 :current-page="pageNum"  
 :page-sizes="[5, 10, 20,30]"  
 :page-size="pageSize"  
 layout="total, sizes, prev, pager, next, jumper"  
 :total="total">  
 </el-pagination>  
 <el-dialog :visible.sync="dialogVisible" title="分享" width="20%">  
 <span slot="footer" class="dialog-footer">  
 <el-button @click="dialogVisible = false">取消</el-button>  
 <el-button type="primary" @click="share">确定</el-button>  
 </span>  
 <div>  
 <p>请输入要分享的用户：</p>  
 <el-autocomplete  
 v-model="searchValue"  
 :fetch-suggestions="fetchUserSuggestions"  
 placeholder="请输入用户名"  
 :trigger-on-focus="false"  
 ></el-autocomplete>  
 </div>  
 </el-dialog>  
 </div>  
</template>  
  
<script>  
import fileDownload from "js-file-download";  
import ***axios*** from "axios";  
  
function request(config) {  
 // 创建axios的实例  
 const instance = ***axios***.create({  
 baseURL: 'http://localhost:8090', // 这个是后端的ip地址  
 });  
 return instance(config);  
}  
  
export function getResult(row) {  
 return request({  
 url: 'comp/getResult',  
 method: 'post',  
 responseType: 'arraybuffer',  
 data: {  
 resultfile: row.filename,  
 userId: row.shareFrom  
 }  
 });  
}  
  
export default {  
 name: "Home",  
 data() {  
 return {  
 shareFile:'',  
 selectedTags: [],  
 tags: ["游戏", "学习", "学术", "娱乐", "工作"],  
 searchValue: '',  
 user: ***JSON***.parse(***sessionStorage***.getItem('CurUser')),  
 fileName: '',  
 dialogVisible:false,  
 shareUrl: '',  
 tableData: [],  
 pageNum: 1,  
 pageSize: 10,  
 total: 0,  
 }  
 },  
 methods: {  
 fetchUserSuggestions(queryString, callback) {  
 // 从后台获取用户名列表  
 // 请根据你的实际情况修改下面的请求地址和参数  
 const url = this.$httpUrl + '/user/getUserNameTop10';  
 ***axios*** .get(url, { params: { userName: queryString } })  
 .then(response => {  
 const users = response.data.map(user => ({ value: user, label: user }));  
 callback(users);  
 })  
 .catch(error => {  
 ***console***.error(error);  
 callback([]);  
 });  
 },  
 resetParam() {  
 this.fileName = ''  
 this.selectedTags =[]  
 },  
 downloadResult(row) {  
 ***console***.log("准备下载")  
 ***console***.log(row)  
 getResult(row).then(res => {  
 fileDownload(res.data, row.filename); // 使用js-file-download来接收二进制流然后下载  
 }).catch(err => {  
 this.$message.error(err);  
 });  
 },  
 del(id) {  
 this.$axios.get(this.$httpUrl + '/comp/del1?id=' + id).then(res => res.data).then(res => {  
 ***console***.log(res)  
 if (res.code == 1) {  
  
 this.$message({  
 message: '操作成功！',  
 type: 'success'  
 });  
 this.loadPost()  
 } else {  
 this.$message({  
 message: '操作失败！',  
 type: 'error'  
 });  
 }  
  
 })  
 },  
 showShareDialog(row) {  
 this.shareFile = row  
 this.searchValue = ''  
 this.dialogVisible = true  
 },  
 share() {  
 // 在这里执行分享操作  
 ***console***.log(this.shareFile)  
 this.shareFile.id = null  
 this.$set(this.shareFile, 'userName', this.searchValue)  
 this.$axios.post(this.$httpUrl + '/comp/saveShare', this.shareFile).then(res => res.data).then(res => {  
 ***console***.log(res)  
 this.$message.success("分享成功")  
 this.searchValue = ''  
 this.shareFile = ''  
 })  
 this.dialogVisible = false;  
 },  
 searchUsers() {  
 if (this.shareUserName.trim()) {  
 this.$axios.get(`${this.$httpUrl}/user/search?name=${this.shareUserName.trim()}`).then(res => res.data).then(res => {  
 if (res && res.length > 0) {  
 const userList = res;  
 this.$autocomplete('请选择要分享的用户', userList, {  
 confirmButtonText: '确定',  
 cancelButtonText: '取消',  
 }).then(({ value }) => {  
 // 用户选择后更新选中的用户ID  
 this.selectedUserId = value;  
 }).catch(() => {  
 this.selectedUserId = null; // 用户取消选择时清空选中的用户ID  
 });  
 } else {  
 this.selectedUserId = null; // 未找到匹配用户时清空选中的用户ID  
 // this.$message.warning('未找到匹配的用户');  
 }  
 });  
 }  
 },  
 loadPost() {  
 this.$axios.post(this.$httpUrl + '/comp/listPageC2', {  
 pageSize: this.pageSize,  
 pageNum: this.pageNum,  
 param: {  
 userId: this.user.id,  
 fileName: this.fileName,  
 label: this.selectedTags  
 }  
 }).then(res => res.data).then(res => {  
 ***console***.log(res)  
 if (res.code == 1) {  
 this.tableData = res.data  
 ***console***.log(this.tableData)  
 this.total = res.total  
 } else {  
 alert('获取数据失败')  
 }  
  
 })  
 },  
 handleSizeChange(val) {  
 ***console***.log(`每页 ${val} 条`);  
 this.pageNum = 1  
 this.pageSize = val  
 this.loadPost()  
 },  
 handleCurrentChange(val) {  
 ***console***.log(`当前页: ${val}`);  
 this.pageNum = val  
 this.loadPost()  
 },  
 },  
 mounted() {  
 this.loadPost()  
 }  
  
}

边栏：

从后台menu数据表中查询菜单栏的信息动态调整。

<template>  
 <el-menu  
 background-color="#545c64"  
 text-color="#fff"  
 active-text-color="#ffd04b"  
 style="height: 100%;"  
 default-active="/Home"  
 :collapse="isCollapse"  
 :collapse-transition="false"  
 router  
 >  
 <el-menu-item index="/Home">  
 <i class="el-icon-s-home"></i>  
 <span slot="title">首页</span>  
 </el-menu-item>  
  
 <el-menu-item :index="'/'+item.menuclick" v-for="(item,i) in menu" :key="i">  
 <i :class="item.menuicon"></i>  
 <span slot="title">{{item.menuname}}</span>  
 </el-menu-item>  
  
  
 </el-menu>  
</template>  
  
<script>  
 export default {  
 name: "Aside",  
 data(){  
 return {  
 //isCollapse:false  
  
 /\* menu:[  
 {  
 menuClick:'Admin',  
 menuName:'管路员管理',  
 menuIcon:'el-icon-s-custom'  
 },{  
 menuClick:'User',  
 menuName:'用户管理',  
 menuIcon:'el-icon-user-solid'  
 }  
 ]\*/  
 }  
 },  
 computed:{  
 "menu":{  
 get(){  
 return this.$store.state.menu  
 }  
 }  
 },  
 props:{  
 isCollapse:Boolean  
 }  
 }  
</script>

标题：

<template>  
 <div style="display: flex;line-height: 60px;">  
 <div style="margin-top: 8px;">  
 <i :class="icon" style="font-size: 20px;cursor: pointer;" @click="collapse"></i>  
 </div>  
 <div style="flex: 1;text-align: center;font-size: 34px;">  
 <span>欢迎来到文档管理系统</span>  
 </div>  
 <el-dropdown>  
 <span>{{user.name}}</span><i class="el-icon-arrow-down" style="margin-left: 5px;"></i>  
 <el-dropdown-menu slot="dropdown">  
 <el-dropdown-item @click.native="logout">退出登录</el-dropdown-item>  
 </el-dropdown-menu>  
 </el-dropdown>  
  
 </div>  
</template>  
  
<script>  
 export default {  
 name: "Header",  
 data(){  
 return {  
 user : ***JSON***.parse(***sessionStorage***.getItem('CurUser'))  
 }  
 },  
 props:{  
 icon:String  
 },  
 methods:{  
 toUser(){  
 ***console***.log('to\_user')  
  
 this.$router.push("/Home")  
 },  
 logout(){  
 ***console***.log('logout')  
  
 this.$confirm('您确定要退出登录吗?', '提示', {  
 confirmButtonText: '确定', //确认按钮的文字显示  
 type: 'warning',  
 center: true, //文字居中显示  
  
 })  
 .then(() => {  
 this.$message({  
 type:'success',  
 message:'退出登录成功'  
 })  
  
 this.$router.push("/")  
 ***sessionStorage***.clear()  
 })  
 .catch(() => {  
 this.$message({  
 type:'info',  
 message:'已取消退出登录'  
 })  
 })  
  
 },  
 collapse(){  
 this.$emit('doCollapse')  
 }  
  
 },  
 created(){  
 this.$router.push("/Home")  
 }  
  
 }  
</script>

**后端部分：**

后端使用controller, entity, mapper, service的结构对MySQL中的数据表进行组织。

user：

controller层：

重要函数：

//登录  
@PostMapping("/login")  
public Result login(@RequestBody User user){  
 List list = userService.lambdaQuery()  
 .eq(User::getNo,user.getNo())  
 .eq(User::getPassword,user.getPassword()).list();  
  
  
 if(list.size()>0){  
 User user1 = (User)list.get(0);  
 List menuList = menuService.lambdaQuery().like(Menu::getMenuright,user1.getRoleId()).list();  
 HashMap res = new HashMap();  
 res.put("user",user1);  
 res.put("menu",menuList);  
 return Result.*suc*(res);  
 }  
 return Result.*fail*();  
}

service：

分享时显示前十个用户名。

@Override  
public List<String> getUserNameTop10(String userName) {  
 QueryWrapper<User> wrapper = new QueryWrapper<>();  
 wrapper.like("name", userName)  
 .last("limit 10");  
 return userMapper.selectList(wrapper).stream().map(User::getName).collect(Collectors.*toList*());  
}

Myfile：

controller：

分享：

收到分享消息后，会在shareFile数据库中增加一条数据记录分享来源。

@PostMapping("/saveShare")  
public void saveShare(@RequestBody ShareFileDto shareFileDto){  
 Sharefile shareFile = new Sharefile();  
 shareFileDto.setShareFrom(shareFileDto.getUserid());  
 LambdaQueryWrapper<User> lambdaQueryWrapper = new LambdaQueryWrapper();  
 lambdaQueryWrapper.eq(User::getName,shareFileDto.getUserName());  
 int userId = userService.list(lambdaQueryWrapper).stream().map(User::getId).collect(Collectors.*toList*()).get(0);  
 shareFileDto.setUserid(userId);  
 BeanUtils.*copyProperties*(shareFileDto,shareFile);  
 sharefileService.save(shareFile);  
}

上传：

上传后，文件保存在以用户id为名字的文件夹中。

//上传  
@RequestMapping("/uploadResult")  
public Result upload(@RequestParam("file") MultipartFile file, @RequestParam("userId") Integer userId,@RequestParam("label") String label){  
 System.*out*.println("上传文件");  
 String fileName = file.getOriginalFilename();  
 // 获取文件名  
 File userDirectory = new File(uploadPath + userId);  
 if(!userDirectory.exists()){  
 userDirectory.mkdir();  
 }  
 File dest = new File(uploadPath +userId+"\\"+fileName);  
 try{  
 file.transferTo(dest);  
 System.*out*.println("上传成功");  
 save(new Myfile(userId,fileName, dest.getAbsolutePath(),label));  
 }catch(Exception e){  
 return Result.*fail*(); // 上传失败就返回前端上传失败的信息  
 }  
 return Result.*suc*(); // 上传成功就返回前端上传成功的信息。  
}

下载：

从路径下载文件

//下载  
@PostMapping("getResult")  
public void getCompResult(@RequestBody Map<String, String> map, HttpServletResponse response) throws Exception {  
 System.*out*.println("下载文件");  
 String owner = map.get("userId");  
 if(map.get("shareFrom")!=null && !map.get("shareFrom").equals("0")){  
 owner = map.get("shareFrom");  
 }  
 String path = uploadPath +owner+"\\"+map.get("resultfile"); // 这里的path是文件存储的路径  
 InputStream inputStream = new FileInputStream(path);  
 response.reset();  
 response.setContentType("application/octet-stream");  
 response.setHeader("Access-Control-Allow-Origin", "\*");  
 response.addHeader("Content-Disposition", "attachment; filename="+ URLEncoder.*encode*(map.get("resultfile"), "UTF-8"));  
 ServletOutputStream outputStream = response.getOutputStream();  
 byte[] b = new byte[1024];  
 int len;  
 while((len = inputStream.read(b)) > 0){  
 outputStream.write(b, 0, len);  
 }  
 System.*out*.println("下载成功");  
 inputStream.close();  
}

删除：

删除记录和文件。

//删除  
@GetMapping("/del")  
public Result del(@RequestParam String id){  
 File dest = new File(myfileService.getById(id).getFileurl());  
 if(dest.exists()){  
 // 上传的文件被保存了  
 dest.delete();  
 myfileService.removeById(id);  
 return Result.*suc*();  
 }  
  
 return Result.*fail*();  
}

列表：

根据查询条件列表。

@PostMapping("/listPageC1")  
public Result listPageC1(@RequestBody QueryPageParam query){  
 HashMap param = query.getParam();  
 String fileName = (String)param.get("fileName");  
 Integer userId = (Integer)param.get("userId");  
 List<String> list = (List) param.get("label");  
 Page<Myfile> page = new Page();  
 page.setCurrent(query.getPageNum());  
 page.setSize(query.getPageSize());  
  
 LambdaQueryWrapper<Myfile> lambdaQueryWrapper = new LambdaQueryWrapper();  
 for(String label:list){  
 lambdaQueryWrapper.or().like(Myfile::getLabel,label);  
 }  
 if(StringUtils.*isNotBlank*(fileName) && !"null".equals(fileName)){  
 lambdaQueryWrapper.like(Myfile::getFilename,fileName);  
 }  
 lambdaQueryWrapper.eq(Myfile::getUserid,userId);  
  
 //IPage result = userService.pageC(page);  
 IPage result = myfileService.pageCC(page,lambdaQueryWrapper);  
  
 System.*out*.println("total=="+result.getTotal());  
  
 return Result.*suc*(result.getRecords(),result.getTotal());  
}

数据结构：

user：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 描述 | 备注 |
| id | int(11) | 主键 | 主键，非空，自增 |
| no | varchar(20) | 账号 | 非空，unique |
| name | varchar(100) | 名字 | 非空，unique |
| password | varchar(20) | 密码 | 非空 |
| roleId | int(11) | 角色0超级管理员，1管理员，2普通账号 | 在本实验中，没有进行区分，权限管理功能可以进一步扩展。 |
| isValid | varchar(4) | 是否有效，Y有效，其他无效 |  |

myfile&sharefile：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 描述 | 备注 |
| id | int(11) | 主键 | 主键，非空，自增 |
| userId | int(11) | 用户id | 非空 |
| filename | varchar(100) | 文件名 | 非空 |
| fileUrl | varchar(200) | 文件地址 | 非空 |
| shareFrom | int(11) | 分享人 | 非空 |
| label | varchar(11) | 标签 |  |

menu：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 描述 | 备注 |
| id | int(11) | 主键 | 主键，非空，自增 |
| menuCode | varchar(8) | 菜单编码 | 非空 |
| menuName | varchar(16) | 菜单名字 | 非空 |
| menuLevel | varchar(2) | 菜单级别 | 非空 |
| menuParentCode | varchar(8) | 菜单的父code |  |
| menuClick | varchar(16) | 点击触发的函数 | 非空 |
| menuRight | varchar(8) | 权限，0超级管理员，1表示管理员，2表示普通用户，可以用逗号组合使用 |  |
| menuComponent | varchar(200) | 菜单页面对应组件 | 非空 |
| menuIcon | varchar(100) | 菜单图标 |  |

1. 实验结果分析

登录界面：



注册界面：



首页：



文件上传页



好友分享页：

