|  |  |
| --- | --- |
| **项目编号** | **Course\_1** |
| **文档编号** | **YWPHH** |
| **密级** |  |

**课程资料自主学习系统编码清单**

**V1.0**

**广西民族大学**

评审日期： 2021年6月24日

目录

[1.后端项目 3](#_Toc75505130)

[1.1 Common：放置dto数据模型、全局异常捕获、结果封装 3](#_Toc75505131)

[1.2 Dto:表现层与控制层之间的数据传输对象，去除entity中表现层不需要的属性 3](#_Toc75505132)

[1.3 Exception：进行项目的全局异常捕获 4](#_Toc75505133)

[1.4 Lang：放置控制层返回表现层的结果封装 6](#_Toc75505134)

[1.5 Config:对第三方引入的包进行配置 7](#_Toc75505135)

[1.6 entity:实体类层，封装数据模型 11](#_Toc75505136)

[1.7 Mapper：dao层，对应接口中由mybaties-plus映射生成，接口无内容 13](#_Toc75505137)

[1.8 Service：业务逻辑层，对应接口中调用mybaties-plus中的wrapper，接口无内容 14](#_Toc75505138)

[1.9 Shiro：实现了项目的身份验证 14](#_Toc75505139)

[1.10 Controller：控制层，实现表现层和控制层的数据传输 18](#_Toc75505140)

[2.前端项目 24](#_Toc75505141)

[2.1 Admin：存放管理员对应界面 24](#_Toc75505142)

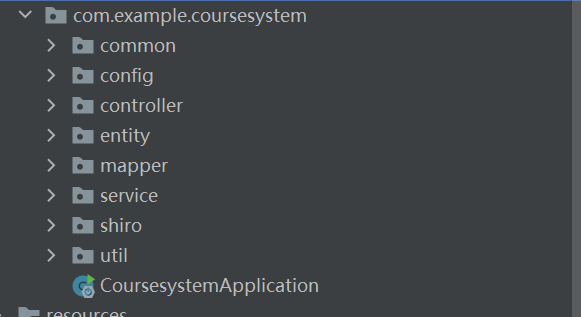
[2.2 Inc放置各角色的菜单，绑定到各个角色的路由 54](#_Toc75505143)

[2.3 News：放置新闻内容相关页面 58](#_Toc75505144)

[2.4 Student:放置显示学生的页面 60](#_Toc75505145)

[2.5 Teacher：放置教师页面 62](#_Toc75505146)

1.后端项目



1.1 Common：放置dto数据模型、全局异常捕获、结果封装



1.2 Dto:表现层与控制层之间的数据传输对象，去除entity中表现层不需要的属性

FileDto.java

@Data

**public** **class** FileDto **implements** Serializable {

String fileName;

}

CourseDto.java

@Data

**public** **class** CourseDto **implements** Serializable {

**private** String cno;

**private** String cname;

}

LoginDto.java

@Data

**public** **class** LoginDto **implements** Serializable {

@NotBlank(message = "昵称不能为空")

**private** String username;

@NotBlank(message = "密码不能为空")

**private** String password;

@NotBlank(message = "角色不能为空")

**private** String role;

}

1.3 Exception：进行项目的全局异常捕获

@Slf4j

@RestControllerAdvice

**public** **class** GlobalExceptionHandler {

@ResponseStatus(HttpStatus.UNAUTHORIZED)

@ExceptionHandler(value = ShiroException.**class**)

**public** Result handler(ShiroException e) {

log.error("运行时异常：----------------{}", e);

**return** Result.fail(401, e.getMessage(), **null**);

}

@ResponseStatus(HttpStatus.BAD\_REQUEST)

@ExceptionHandler(value = MethodArgumentNotValidException.**class**)

**public** Result handler(MethodArgumentNotValidException e) {

log.error("实体校验异常：----------------{}", e);

BindingResult bindingResult = e.getBindingResult();

ObjectError objectError = bindingResult.getAllErrors().stream().findFirst().get();

**return** Result.fail(objectError.getDefaultMessage());

}

@ResponseStatus(HttpStatus.BAD\_REQUEST)

@ExceptionHandler(value = IllegalArgumentException.**class**)

**public** Result handler(IllegalArgumentException e) {

log.error("Assert异常：----------------{}", e);

**return** Result.fail(e.getMessage());

}

@ResponseStatus(HttpStatus.BAD\_REQUEST)

@ExceptionHandler(value = RuntimeException.**class**)

**public** Result handler(RuntimeException e) {

log.error("运行时异常：----------------{}", e);

**return** Result.fail(e.getMessage());

}

}

1.4 Lang：放置控制层返回表现层的结果封装

@Data

**public** **class** Result **implements** Serializable {

**private** **int** code; *// 200是正常，非200表示异常*

**private** String msg;

**private** Object data;

**public** **static** Result success(Object data) {

**return** success(200, "操作成功", data);

}

**public** **static** Result success(**int** code, String msg, Object data) {

Result r = **new** Result();

r.setCode(code);

r.setMsg(msg);

r.setData(data);

**return** r;

}

**public** **static** Result fail(String msg) {

**return** fail(400, msg, **null**);

}

**public** **static** Result fail(String msg, Object data) {

**return** fail(400, msg, data);

}

**public** **static** Result fail(**int** code, String msg, Object data) {

Result r = **new** Result();

r.setCode(code);

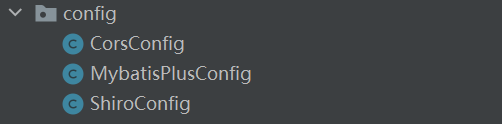
r.setMsg(msg);

r.setData(data);

**return** r;

}

}



1.5 Config:对第三方引入的包进行配置

***/\*\****

***\* 解决跨域问题***

***\*/***

@Configuration

**public** **class** CorsConfig **implements** WebMvcConfigurer {

@Override

**public** **void** addCorsMappings(CorsRegistry registry) {

registry.addMapping("/\*\*")

.allowedOriginPatterns("\*")

.allowedMethods("GET", "HEAD", "POST", "PUT", "DELETE", "OPTIONS")

.allowCredentials(**true**)

.maxAge(3600)

.allowedHeaders("\*");

}

}

@Configuration

*//开启基于注解的事务管理，跟@Transactional注解配套使用*

@EnableTransactionManagement

@MapperScan("com.example.coursesystem.mapper")

**public** **class** MybatisPlusConfig {

@Bean

**public** PaginationInterceptor paginationInterceptor() {

PaginationInterceptor paginationInterceptor = **new** PaginationInterceptor();

**return** paginationInterceptor;

}

}

@Configuration

**public** **class** ShiroConfig {

@Autowired

JwtFilter jwtFilter;

@Autowired

RedisSessionDAO redisSessionDAO;

@Autowired

RedisCacheManager redisCacheManager;

@Bean

**public** SessionManager sessionManager(RedisSessionDAO redisSessionDAO) {

DefaultWebSessionManager sessionManager = **new** DefaultWebSessionManager();

*// inject redisSessionDAO*

sessionManager.setSessionDAO(redisSessionDAO);

**return** sessionManager;

}

@Bean

**public** DefaultWebSecurityManager securityManager(AccountRealm accountRealm,

SessionManager sessionManager,

RedisCacheManager redisCacheManager) {

DefaultWebSecurityManager securityManager = **new** DefaultWebSecurityManager(accountRealm);

*//inject sessionManager*

securityManager.setSessionManager(sessionManager);

*// inject redisCacheManager*

securityManager.setCacheManager(redisCacheManager);

**return** securityManager;

}

@Bean

**public** ShiroFilterChainDefinition shiroFilterChainDefinition() {

DefaultShiroFilterChainDefinition chainDefinition = **new** DefaultShiroFilterChainDefinition();

Map<String, String> filterMap = **new** LinkedHashMap<>();

filterMap.put("/\*\*", "jwt");

chainDefinition.addPathDefinitions(filterMap);

**return** chainDefinition;

}

@Bean("shiroFilterFactoryBean")

**public** ShiroFilterFactoryBean shiroFilterFactoryBean(SecurityManager securityManager,

ShiroFilterChainDefinition shiroFilterChainDefinition) {

ShiroFilterFactoryBean shiroFilter = **new** ShiroFilterFactoryBean();

shiroFilter.setSecurityManager(securityManager);

Map<String, Filter> filters = **new** HashMap<>();

filters.put("jwt", jwtFilter);

shiroFilter.setFilters(filters);

Map<String, String> filterMap = shiroFilterChainDefinition.getFilterChainMap();

*// Map<String, String> filterMap = new LinkedHashMap<String, String>();*

filterMap.put("/login", "anon");

*//设置需要过滤的链接*

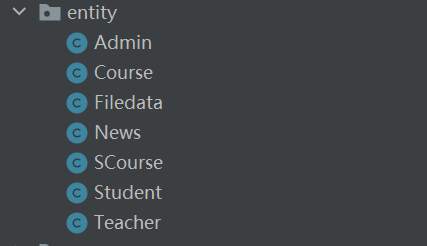
filterMap.put("/admin/\*\*","authc");

shiroFilter.setFilterChainDefinitionMap(filterMap);

**return** shiroFilter;

}

}



1.6 entity:实体类层，封装数据模型

@Data

@EqualsAndHashCode(callSuper = **false**)

@Accessors(chain = **true**)

**public** **class** Admin **implements** Serializable {

**private** **static** **final** **long** serialVersionUID = 1L;

@TableId(value = "id", type = IdType.AUTO)

**private** Integer id;

**private** String ano;

**private** String pwd;

}

@Data

@EqualsAndHashCode(callSuper = **false**)

@Accessors(chain = **true**)

**public** **class** Course **implements** Serializable {

**private** **static** **final** **long** serialVersionUID = 1L;

@TableId(value = "id", type = IdType.AUTO)

**private** Integer id;

**private** String cno;

**private** String cname;

**private** String tno;

}

@Data

@EqualsAndHashCode(callSuper = **false**)

@Accessors(chain = **true**)

**public** **class** Filedata **implements** Serializable {

**private** **static** **final** **long** serialVersionUID = 1L;

@TableId(value = "id", type = IdType.AUTO)

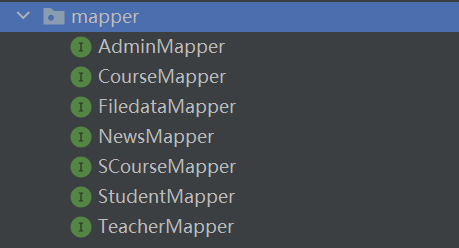
**private** Integer id;

**private** String filename;

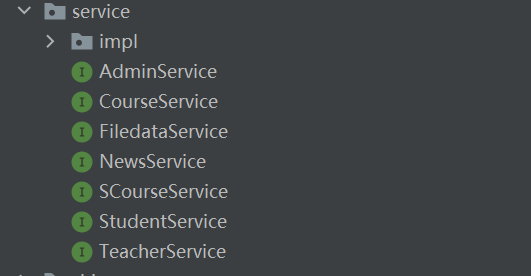
**private** String filepath;

**private** String cno;

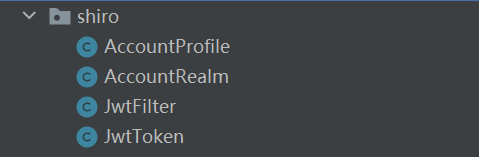
}



1.7 Mapper：dao层，对应接口中由mybaties-plus映射生成，接口无内容



1.8 Service：业务逻辑层，对应接口中调用mybaties-plus中的wrapper，接口无内容



1.9 Shiro：实现了项目的身份验证

@Data

**public** **class** AccountProfile **implements** Serializable {

**private** Long id;

**private** String role;

}

@Component

**public** **class** JwtFilter **extends** AuthenticatingFilter {

@Autowired

JwtUtils jwtUtils;

@Override

**protected** AuthenticationToken createToken(ServletRequest servletRequest, ServletResponse servletResponse) **throws** Exception {

HttpServletRequest request = (HttpServletRequest) servletRequest;

String jwt = request.getHeader("Authorization");

**if** (StringUtils.isEmpty(jwt)){

**return** **null**;

}

**return** **new** com.example.coursesystem.shiro.JwtToken(jwt);

}

@Override

**protected** **boolean** onAccessDenied(ServletRequest servletRequest, ServletResponse servletResponse) **throws** Exception {

HttpServletRequest request = (HttpServletRequest) servletRequest;

String token = request.getHeader("Authorization");

**if** (StringUtils.isEmpty(token)){ *//无jwt时交给注解进行拦截*

**return** **true**;

}**else** {

*//校验jwt*

Claims claims = jwtUtils.getClaimByToken(token);

**if** (claims==**null**||jwtUtils.isTokenExpired(claims.getExpiration())){

**throw** **new** ExpiredCredentialsException("token已失效，请重新登录");

}

**return** executeLogin(servletRequest,servletResponse);

}

}

@Override

**protected** **boolean** onLoginFailure(AuthenticationToken token, AuthenticationException e, ServletRequest request, ServletResponse response) {

HttpServletResponse httpServletResponse = (HttpServletResponse) response;

Throwable throwable = e.getCause()==**null**?e:e.getCause();

Result result = Result.fail(throwable.getMessage());

String json = JSONUtil.toJsonStr(result);

**try** {

httpServletResponse.getWriter().print(json);

} **catch** (IOException ioException) {

ioException.printStackTrace();

}

**return** **false**;

}

***/\*\****

***\* 对跨域提供支持***

***\*/***

@Override

**protected** **boolean** preHandle(ServletRequest request, ServletResponse response) **throws** Exception {

HttpServletRequest httpServletRequest = WebUtils.toHttp(request);

HttpServletResponse httpServletResponse = WebUtils.toHttp(response);

httpServletResponse.setHeader("Access-control-Allow-Origin", httpServletRequest.getHeader("Origin"));

httpServletResponse.setHeader("Access-Control-Allow-Methods", "GET,POST,OPTIONS,PUT,DELETE");

httpServletResponse.setHeader("Access-Control-Allow-Headers", httpServletRequest.getHeader("Access-Control-Request-Headers"));

*// 跨域时会首先发送一个OPTIONS请求，这里我们给OPTIONS请求直接返回正常状态*

**if** (httpServletRequest.getMethod().equals(RequestMethod.OPTIONS.name())) {

httpServletResponse.setStatus(org.springframework.http.HttpStatus.OK.value());

**return** **false**;

}

**return** **super**.preHandle(request, response);

}

}

**public** **class** JwtToken **implements** AuthenticationToken, Serializable {

**private** String token;

**public** JwtToken(String jwt) {

**this**.token = jwt;

}

@Override

**public** Object getPrincipal() {

**return** token;

}

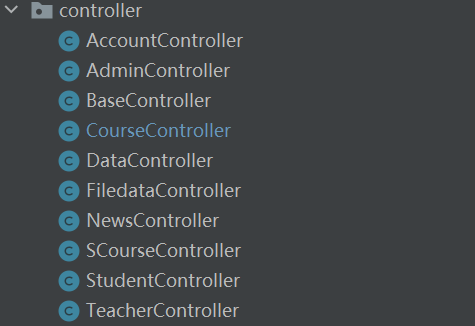
@Override

**public** Object getCredentials() {

**return** token;

}

}

‘

1.10 Controller：控制层，实现表现层和控制层的数据传输

@RestController

**public** **class** AccountController {

@Autowired

JwtUtils jwtUtils;

@Autowired

TeacherService teacherService;

@Autowired

StudentService studentService;

@Autowired

AdminService adminService;

@CrossOrigin

@PostMapping("/login")

**public** Result login(@Validated @RequestBody LoginDto loginDto, HttpServletResponse response) {

String roleType = loginDto.getRole();

String jwt;

JwtToken jwtToken;

**switch** (roleType){

**case** "Student":

Student student = studentService.getOne(**new** QueryWrapper<Student>().eq("sno", loginDto.getUsername()));

**Assert**.notNull(student, "用户不存在");

**if**(!student.getPwd().equals(loginDto.getPassword())) {

**return** Result.fail("密码错误！");

}

jwt = jwtUtils.generateToken(student.getSno(),"Student");

JwtToken token = **new** JwtToken(jwt);

response.setHeader("Authorization", jwt);

response.setHeader("Access-control-Expose-Headers", "Authorization");

*// 用户可以另一个接口*

**return** Result.success(MapUtil.builder()

.put("id",student.getId())

.put("sno", student.getSno())

.put("sex", student.getSex())

.put("sname", student.getSname())

.put("role","Student")

.map()

);

**case** "Teacher":

Teacher teacher = teacherService.getOne(**new** QueryWrapper<Teacher>().eq("tno", loginDto.getUsername()));

**Assert**.notNull(teacher, "用户不存在");

**if**(!teacher.getPwd().equals(loginDto.getPassword())) {

**return** Result.fail("密码错误！");

}

jwt = jwtUtils.generateToken(teacher.getTno(),"Teacher");

response.setHeader("Authorization", jwt);

response.setHeader("Access-Control-Expose-Headers", "Authorization");

*// 用户可以另一个接口*

**return** Result.success(MapUtil.builder()

.put("id",teacher.getId())

.put("sno", teacher.getTno())

.put("sex", teacher.getSex())

.put("sname", teacher.getTname())

.put("role","Teacher")

.map()

);

**case** "Admin":

Admin admin = adminService.getOne(**new** QueryWrapper<Admin>().eq("ano", loginDto.getUsername()));

**Assert**.notNull(admin, "用户不存在");

**if**(!admin.getPwd().equals(loginDto.getPassword())) {

**return** Result.fail("密码错误！");

}

jwt = jwtUtils.generateToken(admin.getAno(),"Admin");

response.setHeader("Authorization", jwt);

response.setHeader("Access-Control-Expose-Headers", "Authorization");

*// 用户可以另一个接口*

**return** Result.success(MapUtil.builder()

.put("id",admin.getId())

.put("sno", admin.getAno())

.put("role","Admin")

.map()

);

}

**return** Result.fail("未选择角色");

}

@RequiresAuthentication

@GetMapping("/logout")

**public** Result logout() {

SecurityUtils.getSubject().logout();

**return** Result.success(**null**);

}

}

@RestController

@RequestMapping("/admin")

**public** **class** AdminController {

@Autowired

StudentService studentService;

@Autowired

TeacherService teacherService;

@Autowired

NewsService newsService;

@GetMapping("/news")

**public** Result news(Integer currentPage) {

**if**(currentPage == **null** || currentPage < 1) currentPage = 1;

Page page = **new** Page(currentPage, 5);

IPage pageData = newsService.page(page, **new** QueryWrapper<News>().orderByDesc("created"));

**return** Result.success(pageData);

}

@GetMapping("/news/{id}")

**public** Result detail(@PathVariable(name = "id") String id) {

News news = newsService.getById(id);

**Assert**.notNull(news, "该新闻已删除！");

**return** Result.success(news);

}

@PostMapping("/news/edit")

**public** Result edit(@Validated @RequestBody News news) {

System.out.println(news.toString());

News temp = **null**;

**if**(news.getId() != **null**) {

temp = newsService.getById(news.getId());

} **else** {

temp = **new** News();

temp.setCreated(LocalDateTime.now());

}

BeanUtil.copyProperties(news, temp, "id", "created");

newsService.saveOrUpdate(temp);

**return** Result.success("操作成功");

}

}

**public** **class** BaseController {

@Autowired

HttpServletRequest req;

***/\*\****

***\* 获取页面***

***\* @return***

***\*/***

**public** Page getPage() {

**int** current = ServletRequestUtils.getIntParameter(req, "cuurent", 1);

**int** size = ServletRequestUtils.getIntParameter(req, "size", 10);

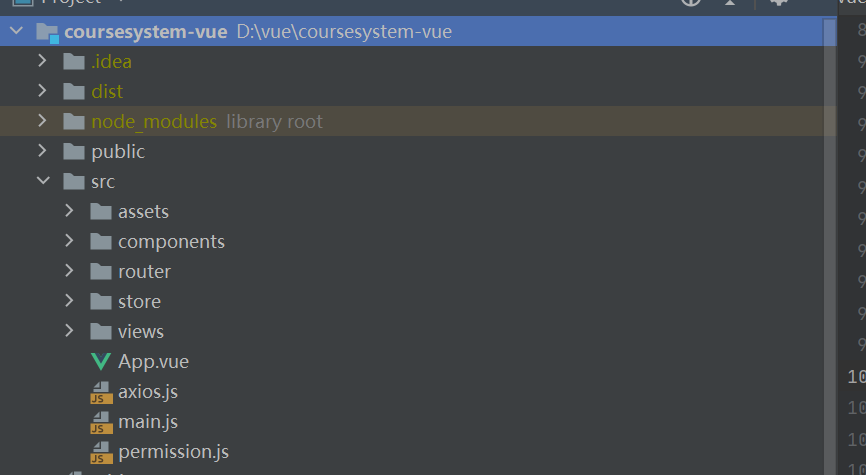
**return** **new** Page(current, size);

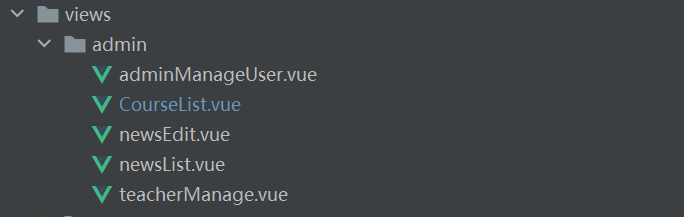
}

}

2.前端项目

前端项目由vue.js编写





2.1 Admin：存放管理员对应界面

<template>

<**div**>

<el-form :inline="true">

<el-form-item>

<el-input

v-model="searchForm.sno"

placeholder="用户名"

clearable

>

</el-input>

</el-form-item>

<el-form-item>

<el-button @click="getUserList">搜索</el-button>

</el-form-item>

<el-form-item>

<el-button type="primary" @click="dialogVisible = true">新增</el-button>

</el-form-item>

<el-form-item>

<el-popconfirm title="确定批量删除吗？" @confirm="delHandle(null)">

<el-button type="danger" slot="reference" :disabled="delBtlStatu">批量删除</el-button>

</el-popconfirm>

</el-form-item>

</el-form>

<el-table

ref="multipleTable"

:data="tableData"

tooltip-effect="dark"

style="width: 100%"

border

stripe

@selection-change="handleSelectionChange">

<el-table-column

type="selection"

width="55">

</el-table-column>

<el-table-column

prop="id"

label="ID"

width="120">

</el-table-column>

<el-table-column

prop="sname"

label="姓名">

</el-table-column>

<el-table-column

prop="sno"

label="学号">

</el-table-column>

<el-table-column

prop="sex"

label="性别">

</el-table-column>

<el-table-column

prop="pwd"

label="密码">

</el-table-column>

<el-table-column

prop="icon"

width="260px"

label="操作">

<template slot-scope="scope">

<el-button type="text" @click="editHandle(scope.row.id)">编辑</el-button>

<el-divider direction="vertical"></el-divider>

<template>

<el-popconfirm title="确定删除吗？" @confirm="delHandle(scope.row.id)">

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

</el-popconfirm>

</template>

</template>

</el-table-column>

</el-table>

<el-pagination

@size-change="handleSizeChange"

@current-change="handleCurrentChange"

layout="total, sizes, prev, pager, next, jumper"

:page-sizes="[10, 20, 50, 100]"

:current-page="current"

:page-size="size"

:total="total">

</el-pagination>

*<!--新增对话框-->*

<el-dialog

title="提示"

:visible.sync="dialogVisible"

width="600px"

:before-close="handleClose">

<el-form :model="editForm" :rules="editFormRules" ref="editForm">

<el-form-item label="ID" prop="id" label-width="100px">

<el-input v-model="editForm.id" autocomplete="off" disabled></el-input>

</el-form-item>

<el-form-item label="学号" prop="sno" label-width="100px">

<el-input v-model="editForm.sno" autocomplete="off" ></el-input>

</el-form-item>

<el-form-item label="姓名" prop="sname" label-width="100px">

<el-input v-model="editForm.sname" autocomplete="off"></el-input>

*<!-- <el-alert-->*

*<!-- title="初始密码为888888"-->*

*<!-- :closable="false"-->*

*<!-- type="info"-->*

*<!-- style="line-height: 12px;"-->*

*<!-- ></el-alert>-->*

</el-form-item>

<el-form-item label="性别" prop="sex" label-width="100px">

<el-radio-group v-model="editForm.sex">

<el-radio :label="0">男</el-radio>

<el-radio :label="1">女</el-radio>

</el-radio-group>

</el-form-item>

<el-form-item label="密码" prop="pwd" label-width="100px">

<el-input v-model="editForm.pwd" autocomplete="off"></el-input>

</el-form-item>

</el-form>

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

<el-button @click="resetForm('editForm')">取 消</el-button>

<el-button type="primary" @click="submitForm('editForm')">确 定</el-button>

</**div**>

</el-dialog>

</**div**>

</template>

<**script**>

export default {

name: "adminManageUser",

data() {

return {

searchForm: {},

delBtlStatu: true,

total: 0,

size: 10,

current: 1,

dialogVisible: false,

editForm: {

id:"",

sno:"",

sname:"",

sex:0,

pwd:""

},

tableData: [],

editFormRules: {

sname: [

{required: true, message: '请输入姓名', trigger: 'blur'}

],

sex: [

{required: true, message: '请输入性别', trigger: 'blur'}

],

pwd: [

{required: true, message: '请输入密码', trigger: 'blur'}

]

},

multipleSelection: [],

roleDialogFormVisible: false,

defaultProps: {

children: 'children',

label: 'name'

},

roleForm: {},

roleTreeData: [],

treeCheckedKeys: [],

checkStrictly: true

}

},

created() {

this.getUserList()

this.$axios.get("/student/list").then(res => {

this.roleTreeData = res.data.data.records

})

},

methods: {

toggleSelection(rows) {

if (rows) {

rows.forEach(row => {

this.$refs.multipleTable.toggleRowSelection(row);

});

} else {

this.$refs.multipleTable.clearSelection();

}

},

handleSelectionChange(val) {

console.log("勾选")

console.log(val)

this.multipleSelection = val;

this.delBtlStatu = val.length == 0

},

handleSizeChange(val) {

console.log(`每页 ${val} 条`);

this.size = val

this.getUserList()

},

handleCurrentChange(val) {

console.log(`当前页: ${val}`);

this.current = val

this.getUserList()

},

resetForm(formName) {

this.$refs[formName].resetFields();

this.dialogVisible = false

this.editForm = {}

},

handleClose() {

this.resetForm('editForm')

},

getUserList() {

this.$axios.get("/student/list/", {

params: {

sno: this.searchForm.sno,

current: this.current,

size: this.size

}

}).then(res => {

this.tableData = res.data.data.records

this.size = res.data.data.size

this.current = res.data.data.current

this.total = res.data.data.total

})

},

submitForm(formName) {

this.$refs[formName].validate((valid) => {

if (valid) {

console.log(this.editForm)

this.$axios.post('/student' + (this.editForm.id?'/update' : '/save'), this.editForm)

.then(res => {

this.$message({

showClose: true,

message: '恭喜你，操作成功',

type: 'success',

onClose:() => {

this.getUserList()

}

});

this.dialogVisible = false

})

} else {

console.log('error submit!!');

return false;

}

});

},

editHandle(id) {

console.log(id)

this.$axios.get('/student/Info/' + id).then(res => {

this.editForm = res.data.data

if (res.data.data.sex=="男"){

this.editForm.sex = 0

}

else{

this.editForm.sex = 1

}

this.dialogVisible = true

})

},

delHandle(id) {

var ids = []

if (id) {

ids.push(id)

} else {

this.multipleSelection.forEach(row => {

ids.push(row.id)

})

}

console.log(ids)

this.$axios.post("/student/delete", ids).then(res => {

this.$message({

showClose: true,

message: '恭喜你，操作成功',

type: 'success',

onClose:() => {

this.getUserList()

}

});

})

},

}

}

</**script**>

<**style** scoped>

.el-pagination {

float: right;

margin-top: 22px;

}

</**style**>

<template>

<**div**>

<el-form :inline="true">

<el-form-item>

<el-input

v-model="searchForm.cno"

placeholder="课程名或课程号"

clearable

>

</el-input>

</el-form-item>

<el-form-item>

<el-button @click="getCourseList">搜索</el-button>

</el-form-item>

<el-form-item>

<el-button type="primary" @click="dialogVisible = true">新增</el-button>

</el-form-item>

<el-form-item>

<el-popconfirm title="确定批量删除吗？" @confirm="delHandle(null)">

<el-button type="danger" slot="reference" :disabled="delBtlStatu">批量删除</el-button>

</el-popconfirm>

</el-form-item>

</el-form>

<el-table

ref="multipleTable"

:data="tableData"

tooltip-effect="dark"

style="width: 100%"

border

stripe

@selection-change="handleSelectionChange">

<el-table-column

type="selection"

width="55">

</el-table-column>

<el-table-column

prop="id"

label="ID"

width="120">

</el-table-column>

<el-table-column

prop="cno"

label="课程号">

</el-table-column>

<el-table-column

prop="cname"

label="课程名">

</el-table-column>

<el-table-column

prop="tno"

label="教师编号">

</el-table-column>

<el-table-column

prop="icon"

width="260px"

label="操作">

<template slot-scope="scope">

<el-button type="text" @click="editHandle(scope.row.id)">编辑</el-button>

<el-button type="text" @click="liststudentHandle(scope.row.cno)">查看学生</el-button>

<el-button type="text" @click="addStudentHandle(scope.row.cno)">添加学生</el-button>

<el-divider direction="vertical"></el-divider>

<template>

<el-popconfirm title="确定删除吗？" @confirm="delHandle(scope.row.id)">

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

</el-popconfirm>

</template>

</template>

</el-table-column>

</el-table>

<el-pagination

@size-change="handleSizeChange"

@current-change="handleCurrentChange"

layout="total, sizes, prev, pager, next, jumper"

:page-sizes="[10, 20, 50, 100]"

:current-page="current"

:page-size="size"

:total="total">

</el-pagination>

*<!--新增对话框-->*

<el-dialog

title="课程"

:visible.sync="dialogVisible"

width="600px"

:before-close="handleClose">

<el-form :model="editForm" :rules="editFormRules" ref="editForm">

<el-form-item label="ID" prop="id" label-width="100px">

<el-input v-model="editForm.id" autocomplete="off" disabled></el-input>

</el-form-item>

<el-form-item label="课程号" prop="cno" label-width="100px">

<el-input v-model="editForm.cno" autocomplete="off" ></el-input>

</el-form-item>

<el-form-item label="课程名" prop="sname" label-width="100px">

<el-input v-model="editForm.cname" autocomplete="off"></el-input>

</el-form-item>

<el-form-item label="教师编号" prop="tno" label-width="100px">

<el-select v-model="editForm.tno" placeholder="请选择">

<el-option

v-for="item in tnoList"

:value="item" :key="item">

</el-option>

</el-select>

</el-form-item>

</el-form>

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

<el-button @click="resetForm('editForm')">取 消</el-button>

<el-button type="primary" @click="submitForm('editForm')">确 定</el-button>

</**div**>

</el-dialog>

*<!--查看课程学生对话框-->*

<el-dialog

title="学生列表"

:visible.sync="dialogVisible1"

width="600px"

:before-close="handleClose1">

<el-row>

<el-col :span="8" v-for="item in editForm1" :value="item" :key="item.cno">

<el-card :body-style="{ padding: '0px' }">

<**div** style="padding: 14px;">

<**span**>{{item.sno}}</**span**>

<**span**>{{item.sname}}</**span**>

</**div**>

</el-card>

</el-col>

</el-row>

</el-dialog>

*<!--添加学生对话框-->*

<el-dialog

title="添加学生"

:visible.sync="dialogVisible2"

width="600px"

:before-close="handleClose2">

<el-form :model="editForm2" ref="editForm2">

<el-form-item label="课程号" prop="cno" label-width="100px">

<el-input v-model="editForm2.cno" autocomplete="off" disabled></el-input>

</el-form-item>

<el-form-item label="学生编号" prop="sno" label-width="100px">

<el-select v-model="editForm2.sno" placeholder="请选择">

<el-option

v-for="item1 in snoList"

:value="item1" :key="item1">

</el-option>

</el-select>

</el-form-item>

</el-form>

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

<el-button @click="resetForm('editForm2')">取 消</el-button>

<el-button type="primary" @click="addStudent(editForm2.cno,editForm2.sno)">确 定</el-button>

</**div**>

</el-dialog>

</**div**>

</template>

<**script**>

export default {

name: "CourseList",

data() {

return {

searchForm: {},

delBtlStatu: true,

total: 0,

size: 10,

current: 1,

dialogVisible: false,

dialogVisible1: false,

dialogVisible2: false,

editForm: {

id:"",

cno:"",

cname:"",

tno:"",

},

editForm1: {

id:"",

sno:"",

sname:"",

sex:"",

pwd:""

},

editForm2: {

sno:"",

cno:""

},

tableData: [],

editFormRules: {

cname: [

{required: true, message: '请输入课程名', trigger: 'blur'}

],

tno: [

{required: true, message: '请绑定教师编号', trigger: 'blur'}

]

},

editFormRules1: {

sno: [

{required: true, message: '请绑定学生编号', trigger: 'blur'}

]

},

multipleSelection: [],

roleDialogFormVisible: false,

defaultProps: {

children: 'children',

label: 'name'

},

roleForm: {},

roleTreeData: [],

treeCheckedKeys: [],

checkStrictly: true,

tnoList:[],

snoList:[]

}

},

created() {

this.getCourseList()

this.$axios.get("/course/list").then(res => {

this.roleTreeData = res.data.data.records

})

this.$axios.get("/course/tno/list").then(res => {

this.tnoList = res.data.data

console.log(res.data.data);

})

},

methods: {

toggleSelection(rows) {

if (rows) {

rows.forEach(row => {

this.$refs.multipleTable.toggleRowSelection(row);

});

} else {

this.$refs.multipleTable.clearSelection();

}

},

handleSelectionChange(val) {

console.log("勾选")

console.log(val)

this.multipleSelection = val;

this.delBtlStatu = val.length == 0

},

handleSizeChange(val) {

console.log(`每页 ${val} 条`);

this.size = val

this.getCourseList()

},

handleCurrentChange(val) {

console.log(`当前页: ${val}`);

this.current = val

this.getCourseList()

},

resetForm(formName) {

this.$refs[formName].resetFields();

this.dialogVisible = false

this.editForm = {}

},

resetForm1(formName) {

this.dialogVisible1 = false

this.editForm1 = {}

},

resetForm2(formName) {

this.dialogVisible2 = false

this.editForm2 = {}

},

handleClose() {

this.resetForm('editForm')

},

handleClose1() {

this.resetForm1('editForm1')

},

handleClose2() {

this.resetForm2('editForm2')

},

getCourseList() {

this.$axios.get("/course/list/", {

params: {

cno: this.searchForm.cno,

current: this.current,

size: this.size

}

}).then(res => {

this.tableData = res.data.data.records

this.size = res.data.data.size

this.current = res.data.data.current

this.total = res.data.data.total

})

},

submitForm(formName) {

this.$refs[formName].validate((valid) => {

if (valid) {

console.log(this.editForm)

this.$axios.post('/course' + (this.editForm.id?'/update' : '/save'), this.editForm)

.then(res => {

this.$message({

showClose: true,

message: '恭喜你，操作成功',

type: 'success',

onClose:() => {

this.getCourseList()

}

});

this.dialogVisible = false

})

} else {

console.log('error submit!!');

return false;

}

});

},

editHandle(id) {

console.log(id)

this.$axios.get('/course/Info/' + id).then(res => {

this.editForm = res.data.data

this.dialogVisible = true

})

},

delHandle(id) {

var ids = []

if (id) {

ids.push(id)

} else {

this.multipleSelection.forEach(row => {

ids.push(row.id)

})

}

console.log(ids)

this.$axios.post("/course/delete", ids).then(res => {

this.$message({

showClose: true,

message: '恭喜你，操作成功',

type: 'success',

onClose:() => {

this.getCourseList()

}

});

})

},

liststudentHandle(cno){

this.$axios.get("/course/listStudent?cno="+cno).then(res => {

this.editForm1 = res.data.data

this.dialogVisible1 = true

})

},

addStudentHandle(cno){

this.$axios.get("/student/getsno").then(res => {

this.snoList = res.data.data

console.log(this.snoList)

this.editForm2.cno = cno

this.dialogVisible2 = true

})

},

addStudent(cno,sno){

this.$axios.post("/scourse/addStudent",this.editForm2).then(res => {

console.log(res.data.data)

this.dialogVisible2 = false

})

}

}

}

</**script**>

<template>

<**div** class="m-container">

<**div** class="m-content">

<el-form ref="editForm" status-icon :model="editForm" :rules="rules" label-width="80px">

<el-form-item label="标题" prop="title">

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

</el-form-item>

<el-form-item label="摘要" prop="description">

<el-input type="textarea" v-model="editForm.description"></el-input>

</el-form-item>

<el-form-item label="内容" prop="content">

<mavon-editor v-model="editForm.content"/>

</el-form-item>

<el-form-item>

<el-button type="primary" @click="submitForm()">立即创建</el-button>

<el-button>取消</el-button>

</el-form-item>

</el-form>

</**div**>

</**div**>

</template>

<**script**>

export default {

name: "newsEdit",

data() {

return {

editForm: {

id: null,

title: '',

description: '',

content: ''

},

rules: {

title: [

{required: true, message: '请输入标题', trigger: 'blur'},

{min: 3, max: 50, message: '长度在 3 到 50 个字符', trigger: 'blur'}

],

description: [

{required: true, message: '请输入摘要', trigger: 'blur'}

]

}

}

},

created() {

const newsId = this.$route.params.newsId

const \_this = this

if(newsId) {

this.$axios.get('/admin/news/' + newsId).then((res) => {

console.log(res)

const news = res.data.data

\_this.editForm.id = news.id

\_this.editForm.title = news.title

\_this.editForm.description = news.description

\_this.editForm.content = news.content

});

}

},

methods: {

submitForm() {

const \_this = this

this.$refs.editForm.validate((valid) => {

if (valid) {

this.$axios.post('/admin/news/edit', this.editForm, {

headers: {

"Authorization": localStorage.getItem("token")

}

}).then((res) => {

\_this.$alert('操作成功', '提示', {

confirmButtonText: '确定',

callback: action => {

\_this.$router.push("/admin/newsList")

}

});

});

} else {

console.log('error submit!!');

return false;

}

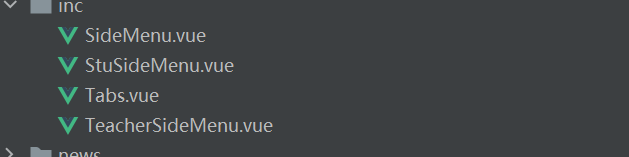
})

}

}

}

</**script**>



2.2 Inc放置各角色的菜单，绑定到各个角色的路由

<template>

<el-menu

class="el-menu-vertical-demo"

background-color="#545c64"

text-color="#fff"

active-text-color="#ffd04b"

>

<el-submenu index="1">

<template slot="title">

<**i** class="el-icon-s-operation"></**i**>

<**span**>用户管理</**span**>

</template>

<router-link to="/admin/stuManage">

<el-menu-item index="1-1">

<template slot="title">

<**i** class="el-icon-s-custom"></**i**>

<**span** slot="title">学生管理</**span**>

</template>

</el-menu-item>

</router-link>

<router-link to="/admin/teacherManage">

<el-menu-item index="1-2">

<template slot="title">

<**i** class="el-icon-s-custom"></**i**>

<**span** slot="title">教师管理</**span**>

</template>

</el-menu-item>

</router-link>

</el-submenu>

<el-submenu index="2">

<template slot="title">

<**i** class="el-icon-news"></**i**>

<**span**>新闻管理</**span**>

</template>

<router-link to="/admin/newsList">

<el-menu-item index="2-1">

<template slot="title">

<**i** class="el-icon-tickets"></**i**>

<**span** slot="title">新闻列表</**span**>

</template>

</el-menu-item>

</router-link>

<router-link to="/admin/newsEdit">

<el-menu-item index="2-2">

<template slot="title">

<**i** class="el-icon-add-location"></**i**>

<**span** slot="title">新闻添加</**span**>

</template>

</el-menu-item>

</router-link>

</el-submenu>

<router-link to="/admin/courseList">

<el-menu-item index="3">

<template slot="title">

<**i** class="el-icon-reading"></**i**>

<**span** slot="title">课程管理</**span**>

</template>

</el-menu-item>

</router-link>

</el-menu>

</template>

<**script**>

export default {

name: "SideMenu",

data() {

}

}

</**script**>

<**style** scoped>

.el-menu-vertical-demo {

height: 100%;

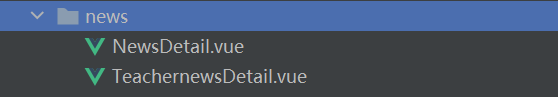
}

a{

text-decoration:none;

}

</**style**>



2.3 News：放置新闻内容相关页面

<template>

<**div** class="m-container">

<**div** class="mblog">

<**h2**>{{ news.title }}</**h2**>

<el-link icon="el-icon-edit" v-if="ownnews"><router-link :to="{name: 'newsEdit', params: {newsId: news.id}}">编辑</router-link></el-link>

<el-divider></el-divider>

<**div** class="content markdown-body" v-html="news.content"></**div**>

</**div**>

</**div**>

</template>

<**script**>

import 'github-markdown-css/github-markdown.css' // 然后添加样式markdown-body

export default {

name: "NewsDetail",

data() {

return {

news: {

title: "",

description: "",

content: ""

},

ownnews: false

}

},

methods: {

getnews() {

const newsId = this.$route.params.newsId

const \_this = this

this.$axios.get('/admin/news/' + newsId).then((res) => {

console.log(res)

console.log(res.data.data)

\_this.news = res.data.data

var MarkdownIt = require('markdown-it'),

md = new MarkdownIt();

var result = md.render(\_this.news.content);

\_this.news.content = result

// 判断是否是自己的文章，能否编辑

\_this.ownnews = ("Admin" === this.$store.getters.getUser.role)

});

}

},

created() {

this.getnews()

}

}

</**script**>

<**style**>

.m-container{

margin-top: 50px;

}

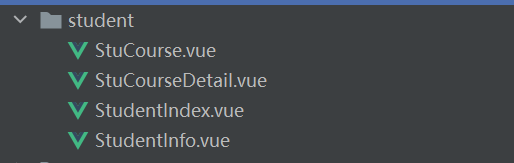
.mblog{

margin: 0 auto;

text-align: center;

}

</**style**>



2.4 Student:放置显示学生的页面

<template>

<**div** class="m-container">

<el-row>

<el-col :span="8" v-for="item in cname" :value="item" :key="item.cno">

<el-card :body-style="{ padding: '0px' }">

<**img** src="https://www.gxun.edu.cn/images/gx\_03.png" class="image">

<**div** style="padding: 14px;">

<**span**>{{item.cname}}</**span**>

<**div** class="bottom clearfix">

<el-button type="text" class="button" >

<router-link :to="{name: 'StuCourseDetail', params: {cno: item.cno}}">详细信息

</router-link>

</el-button>

</**div**>

</**div**>

</el-card>

</el-col>

</el-row>

</**div**>

</template>

<**script**>

export default {

name: "StuCourseDetail",

data() {

return {

cname:[]

}

},

created() {

const \_this = this

const sno = this.$store.getters.getUser.sno

if(sno) {

this.$axios.get('/course/getStuCourse/' + sno).then((res) => {

console.log(res)

this.cname = res.data.data

});

}

},

methods: {

},

}

</**script**>

<**style** scoped>

.m-container{

margin-top: 50px;

}

</**style**>



2.5 Teacher：放置教师页面

<template>

<**div**>

<el-upload class="upload-demo"

:action="uploadUrl"

:before-upload="handleBeforeUpload"

:on-error="handleUploadError"

:before-remove="beforeRemove"

multiple

:limit="5"

:on-exceed="handleExceed"

:file-list="fileList">

<el-button size="small" type="primary">点击上传</el-button>

</el-upload>

*<!-- 下载-->*

<el-table

ref="multipleTable"

:data="tableData"

tooltip-effect="dark"

style="width: 100%"

border

stripe

>

<el-table-column

prop="filename"

label="文件名">

</el-table-column>

<el-table-column

prop="icon"

width="260px"

label="操作">

<template slot-scope="scope">

*<!-- <el-button type="button" @click="download(scope.row.filename)"> 下载附件</el-button>-->*

<**a** :href="'http://localhost:8081/teacher/file/download?fileName='+scope.row.filename">下载附件</**a**>

<el-divider direction="vertical"></el-divider>

*<!-- <template>-->*

*<!-- <el-popconfirm title="确定删除吗？" @confirm="delHandle(scope.row.id)">-->*

*<!-- <el-button type="text" slot="reference">删除</el-button>-->*

*<!-- </el-popconfirm>-->*

*<!-- </template>-->*

</template>

</el-table-column>

</el-table>

</**div**>

</template>

<**script**>

export default {

name: "CourseDeatil",

data () {

return {

uploadUrl: 'http://localhost:8081/teacher/file/upload',

fileList: [],

downloadlist:[],

tableData: [],

}

},

created() {

this.getFileList()

},

methods: {

handleExceed(files, fileList) {

this.$message.warning(`当前限制选择 5 个文件，本次选择了 ${files.length} 个文件，共选择了 ${files.length + fileList.length} 个文件`);

},

beforeRemove(file, fileList) {

return this.$confirm(`确定移除 ${ file.name }？`);

},

handleUploadError(error, file) {

console.log("文件上传出错："+error)

// this.$notify.error({

// title: 'error',

// message: '上传出错:' + error,

// type: 'error',

// position: 'bottom-right'

// })

},

//测试上传文件(注意web的上下文)

handleBeforeUpload(file){

this.uploadUrl = 'http://localhost:8081/teacher/file/upload'

console.log("开始上传，上传的文件为："+file)

let formData = new FormData();

formData.append("multipartFiles", file);

this.$axios.post('/teacher/file/upload',formData,{

headers: {

'Content-Type': 'multipart/form-data',

},

params:{

'cno':this.$route.params,

}

}).then((res) => {

console.log("文件上传返回："+res)

this.$router.go(0);

}).catch(error => {

console.log("文件上传异常:"+error)

})

},

//获取所有该课程文件

getFileList(){

this.$axios.get('/teacher/file/getFileList',{

params:{

'cno':this.$route.params,

}

}).then((res) => {

console.log(res.data.data.records)

this.tableData = res.data.data.records

})

},

//文件下载

download(fileName){

this.$axios.get('/teacher/file/download?fileName='+fileName,{timeout: 1000})

}

}

}

</**script**>

<**style** scoped>

</**style**>

|  |  |  |  |
| --- | --- | --- | --- |
| 单位名称 | 广西民族大学 | 单位名称 | 18软件2班开5人学习小组 |
| 签署人姓名 | 姚展 | 签署人姓名 | 潘建辉 |
| 签署日期 | 2021.06.24 | 签署日期 | 2021.06.24 |