Contents

[初始化配置，Dependency 2](#_Toc68126589)

[Spring Boot MVC 大概结构 2](#_Toc68126590)

[方便的Spring Boot 方法 3](#_Toc68126591)

[ComponentScan 4](#_Toc68126592)

[Validation （验证参数合法性） 4](#_Toc68126593)

[Vue Cli 4](#_Toc68126594)

[Axios 5](#_Toc68126595)

[Vue 表格 及渲染 6](#_Toc68126596)

[Spring 拦截器和过滤器 6](#_Toc68126597)

[树形结构的UI 组件 8](#_Toc68126598)

[Menu 树型结构 8](#_Toc68126599)

[Cascader 8](#_Toc68126600)

[Table 树形结构 9](#_Toc68126601)

[解决树形结构Bug <a-tree-select> 9](#_Toc68126602)

[用户登录校验 9](#_Toc68126603)

[后端登录Token 管理 10](#_Toc68126604)

[前端token管理 10](#_Toc68126605)

[Spring Boot 几个注解 11](#_Toc68126606)

[Spring Boot 定时Jobs 11](#_Toc68126607)

[Spring Boot 异步化注解 11](#_Toc68126608)

[Spring Boot 事务 12](#_Toc68126609)

[前端设计相关 12](#_Toc68126610)

[SVG 12](#_Toc68126611)

[CSS 13](#_Toc68126612)

[FrontAwesome 13](#_Toc68126613)

[Pseudo Element 13](#_Toc68126614)

[业务逻辑 13](#_Toc68126615)

[设计理念 13](#_Toc68126616)

[Todos 14](#_Toc68126617)

# 初始化配置，Dependency

1. Git 配置：VCS => Enable Version Control Integration

关联远程git 仓库

1. MyBatis 2.13
2. MySQL 8.0 (Dev)
3. Ant Design Vue 2.0.1
4. Axios: 0.21.0
5. Vue 3.0.0
6. I18n 9.0.0(International Locale Plugin)
7. wangEditor 4.6.9 (Rich Text Editor)

# Spring Boot MVC 大概结构

Controller 层：目的是通过视图函数映射地址到方法

@RestController  
public class HelloWord {  
  
 @Resource  
 private TestService testService;  
  
 @GetMapping("/test")  
 public List<TestAcc> getAccList(){  
 return testService.list();  
 }

标识符为@RestController

这里调用了service层的资源

Service 层：目的是完成后端主要逻辑

@Service  
public class TestService {  
  
 @Resource  
 private Testmapper testmapper;  
  
 public List<TestAcc> list(){  
 return testmapper.list();  
 }

这里又调用了Mapper层的资源 testmapper

Mapper 层：Mybatis 集成，目的完成与数据库交互

public interface Testmapper {  
  
 public List<TestAcc> list();  
}

定义一个接口一般叫XXXmapper，并在xml里定义SQL 语句映射接口下的方法，调用此方法就相当于调用了映射关系中的SQL 语句并返回为domain里的相关的实体类（TestAcc）。

**Auto generator 代码自动生成器：要配置数据库信息，改环境时可能会用到。**

**Mybatis xml默认每次只执行一个SQL 语句，可以在application properties 里的数据库连接添加？allowMultiQueries=true**

# 方便的Spring Boot 方法

1. BeansUtils.copy(source,target) 复制两个object的属性。

2. 使用Post 请求， 并将paramaters 单独作为一个参数传递

@PostMapping("/save")  
public commonResp saveEbook(@RequestBody UpdateReq req){

3.使用delete 请求，id作为url 的一部分拼接

@DeleteMapping("/delete/{id}")  
public commonResp deleteEbook(@PathVariable Long id)

# ComponentScan

@ComponentScan(excludeFilters = { @Filter(type = FilterType.CUSTOM, classes = TypeExcludeFilter.class),  
 @Filter(type = FilterType.CUSTOM, classes = AutoConfigurationExcludeFilter.class) })

 会扫描main方法下的类找到postmapping, requestmapping 等

可以手动添加。

# Validation （验证参数合法性）

1. 集成组件 spring-boot-starter-validation, 添加依赖

2. 在变量上附加constraint (@Max(value, Max))

3. 写Handler 函数加上注解，处理某类Exception, 是BindException 就给value赋值。

此函数就会自动处理BindException.

@ExceptionHandler(value = BindException.class)  
@ResponseBody  
public CommonResp validExceptionHandler(BindException e) {

4. 在视图函数参数中加入@Valid

5. 如果不满足constraint 会返回400 错误

# Vue Cli

1. Lazy Load: 只有当 route被visited 的时候才会加载component.
2. 安装Vue cli: npm install -g @vue/cli
3. 登录网址之后会自动导向 router-view
4. Footer 和 Header 做成了固定的Component
5. 三个部分：Template(HTML)相关， Script(Js相关)，Style(CSS相关)
6. 如果需要自定义CSS 文件在style里 @import（file）。
7. Vue 中 <div :label=“”> 属于dynamic property 相当于v-bind: 绑定一个变量和一个Object.
8. Vue 页面之间的参数传递：把to 变成响应式变量，然后
9. “‘string部分用单引号’+动态变量”

<router-link :to="'/admin/doc?ebookId=' + record.id">

然后在target的页面 导入useRoute得到路由的各种信息

1. 在一个Vue function里可能不同语句会异步执行

# Axios

创建HTTP 请求从后端获取数据

Npm install axios –save

export default defineComponent({  
 name: 'Home',  
 setup(){  
 ***console***.log("---setup---")  
 ***axios***.get("http://localhost:8088/ebook/search?name=Spring").then((response)=>{  
 ***console***.log(response)  
 })  
 }  
});

Error: Access-Control-Allow-Origin: （跨域问题）因前后端地址不一致导致前端请求被后端拒绝。做好crossConfig.

前端先发送option请求，检验route是否存在，如果存在就正式发送GET 请求。

响应式数据：

Setup 函数：集成了vue2 method, data 等

Const v=ref(); 会随着程序运行而实时更新改变。如果定义成一个String，就不会随着程序加载而改变。**注意赋值的时候用v.value 赋值**

**注意：axios.get 是一个异步函数，不同的axios.get 相当于会开启新的线程。存在线程安全性**

# Vue 表格 及渲染

<a-table ：columns=”columns”></a-table> 表示一个表格, 会自定义一些关于表格的变量。

HTML:

<template #name=”{text}”></template>

TS:

slots: { title: 'customTitle', customRender: 'name' } title 是column name 的渲染， customRender 是值的渲染，‘name’就对应#name

**v-show**: 是将整个元素隐藏

**v-if** 是将整个元素删除，增加

Button:

<a-button type="primary" @click="edit(record)">  
 编辑  
</a-button>

点击这个button就会触发edit这个函数并传入参数record。

# Vue 组件Component

## 定义组件：

在一个文件中直接定义比如叫navigation.vue 并在script 中export default defineComponent({

name: navigation

})

在要使用的页面中导入

## 自定义组件变量及方法

在组件文件中定义

props:["title"],

<li><a href="#" @click="$emit('warn')

>{{title}}</a></li>

点击{{title}} 会触发warn 事件

在父文件也就是调用组件的文件中定义

这里会监听子文件是否有warn 事件，如果有就调用自己定义的prompt 方法

<Navigation title="Sakura" @warn="prompt"/>

# Spring 拦截器和过滤器

Filter 是servlet 容器里的 范围更大

Interceptor 是集成spring web 应用的，所以范围更小

开发interceptor 方法：

实现HandlerInterceptor 接口，重写preHandle 和postHandle 方法。

并标注Component

@Component  
public class LogInterceptor implements HandlerInterceptor {  
  
 private static final Logger *LOG* = LoggerFactory.*getLogger*(LogInterceptor.class);  
 @Override  
 public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {  
 // 打印请求信息  
  
 long startTime = System.*currentTimeMillis*();  
 request.setAttribute("requestStartTime", startTime);  
 return true;  
 }  
  
 @Override  
 public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) throws Exception {  
 long startTime = (Long) request.getAttribute("requestStartTime");  
 *LOG*.info("------------- LogInterceptor 结束 耗时：{} ms -------------", System.*currentTimeMillis*() - startTime);  
 }  
}

并在设置中 注册：

@Configuration  
public class SpringMVCConfig implements WebMvcConfigurer {  
 @Resource  
 LogInterceptor Interceptor;  
  
 public void addInterceptors(InterceptorRegistry registry) {  
 registry.addInterceptor(Interceptor)  
 .addPathPatterns("/\*\*");  
  
 }  
}

# 树形结构的UI 组件

## Menu 树型结构

需要自定义component sub-menu, 详见 views/submenu.vue

必须要选中leaf

Home.vue

<a-menu  
 mode="inline"  
 style="height: 100%"  
 @click="handleClick"  
>  
 <a-menu-item :key="0">  
 {{ $t('sider.allcategory') }}  
 </a-menu-item>  
 <template v-for="item in level1" :key="item.id">  
 <template v-if="!item.children">  
 <a-menu-item :key="item.id">  
 <span>{{ item.name }}</span>  
 </a-menu-item>  
 </template>  
 <template v-else>  
 <sub-menu :menu="item" :key="item.id" />  
 </template>  
 </template>

## Cascader

必须要选中leaf，必须绑定value为 list

<a-cascader v-model:value="arr"  
 :options="level1"  
 placeholder="Please select"  
 :field-names="{ label: 'name', value: 'id'}"  
/>

const arr=ref<string[]>([]);

## Table 树形结构

可以自动支持, 只要data-source 符合结构：

<a-table  
 v-if="level1.length>0"  
 :columns="columns"  
 :row-key="record => record.id"  
 :data-source="level1"  
 :loading="loading"  
 :pagination="false"  
 :defaultExpandAllRows="true"  
>

## 解决树形结构Bug <a-tree-select>

A-tree-select 可以选中任意层级

1. 修改完一个文档分类后，再点击修改会出现没有更改之前的内容， 修改方法：在post之后重新query一次。

2. 不能选择自己为父ID或者自己的子节点为父ID，不然branch会消失。

3. 新增不能新增父节点，parent=0. （填上 0）

# 用户登录校验

统一登录系统，单点登录系统。

X：用户管理，登录，退出登录

JWT(Json web Tokens)：加密业务信息，有意义，加密的，不需要存储到Redis。

1. 前端输入用户名密码
2. 校验用户名密码
3. 生成登录标识token
4. 前端保存token
5. 后端保存token（Redis）

校验：

1. 前端请求时，带上token （放在header）
2. 登录拦截器，校验token（Redis 获取token）
3. 校验失败，返回到登录页面

## 后端登录Token 管理

后端login成功之后 => 用雪花算法生成token=>

loginResp.setToken(token.toString());  
redisTemplate.opsForValue().set(token, JSONObject.*toJSONString*(loginResp),3600, TimeUnit.*SECONDS*);

LoginResp 设置token，token放入Redis=>

返回LoginResp 到前端，包含 token, account, name(昵称)

## 前端token管理

1. 由于login和header是不同页面，所以可以新建一个文件status在status里定义全局变量userstatus。前端拿到LoginResp， 显示登陆成功，userstatus 赋予返回值，同时header监听userstatus account值。

2. vuex ：在index.ts 里定义并导出：

const ***store***=createStore({  
 state: {  
 userstatus:{}  
 },  
 mutations: {  
 setUser(state,user){  
 state.userstatus=user;  
 }  
},

export default store

前端拿到LoginResp， 显示登陆成功，执行commit 方法

***store***.commit("setUser",data.content)

在需要用userstatus 全局变量的地方定义相应变量并用computed监听：

const user=computed(()=> ***store***.state.userstatus);

3. 定义统一前端Axios 拦截器，在所有发送的请求header中附带token信息。

***axios***.interceptors.request.use(function (config) {  
 ***console***.log('Request Parameters：', config);  
 const token=***store***.state.userstatus.token;  
 if(Tool.*isNotEmpty*(token)){  
 config.headers.token=token;  
 }  
 return config;  
}, error => {  
 return ***Promise***.reject(error);  
});

4. 前端登录校验隐藏菜单，防止手敲前端路由，退出登录后自动redirect。

# Spring Boot 几个注解

## Spring Boot 定时Jobs

注意所有定时器是统一线程。

下一个任务会等待上一个job 执行完成，如果错过就不再执行了

Cron 格式

1.首先在启动类上加注解

@EnableScheduling  
public class WikiApplication {…}

2. 定义job，并打上注解：

@Component  
public class DocJob {

@Scheduled(cron=””)

public void cron() {

}

## Spring Boot 异步化注解

@EnableAsync  
public class WikiApplication {  
}

新建一个类：然后去调用里面的异步方法

@Service

public class WsService{  
 @Async  
 public void sendInfo{}  
}

注意同一个类里，调用注解的方法不会生效

缺点：如果这个异步任务耗时很长，就会一直开启新线程占用资源。

## Spring Boot 事务

涉及一项事务失败回滚，要么全部成功要么全部失败

方法上加上@Transactional

注意同一个类里，调用事务注解方法不会生效

# 前端设计相关

## SVG

SVG 层级不能用index只能按照标签先后。详见login

以下方法可使背景星空全可见，且表单可填写

Back 100vw,100vh

|\_banner 100%，100%， absolute

| |\_\_表单wrapper

|\_stars wrapper 100%,100%, absolute

|\_\_svg 画布

SVG的旋转有两次一是按照左上角整体旋转或者指定旋转点（1300 0） 二是自身的旋转（cx,cy）同样以左上角为坐标原点。

<g transform="rotate(45 1300 0)">  
 <ellipse class="comet-f" fill="url(#comet-gradient)" cx="1300px" cy="0px" rx="150" ry="2"></ellipse>  
</g>

**Tips: 注意从photoshop 直接拷贝的SVG代码可能会存在Filter ID 重复比如第一次拷贝会有Filter\_0-Filter\_20 第二次拷贝you会有Filter\_0-Filter\_20.破坏ID 唯一性导致污染。必须要重新定义ID。**

## CSS

调节背景图片适配division缩放，而不会截断：用background-size:100%,100%;

## FrontAwesome

### Pseudo Element

在Index 里导入search 依赖JS

<script data-search-pseudo-elements defer src="https://kit.fontawesome.com/2e75c5cb4f.js"></script>

在CSS 里使用

#team .title:before{  
 font-family: "Font Awesome 5 Free";  
 content: "\f024";  
}

#： 选中ID

. : 选中class

什么都没有是选中tag

# 业务逻辑

1. 用户点赞或者浏览，则点赞数，浏览量+1，直接实时数据库更新 Doc DB

2. 每天执行一次，将今日的DOC最新数据更新到对应的Ebook，updateEbookInfo

3. 每天执行一次，将今日的

# 设计理念

1. 任何表中的ID 都不应该允许人为赋值，用雪花算法进行赋值。
2. 当列表中有一列 数据量特别大时考虑纵向分表，避免在进入时一起加载，增加数据表服务器负荷。
3. updateByPrimaryKeySelective 只会更新非空值，可以用此方法选择性更新表内容。
4. 实现 vote\_count++：
5. update doc set vote\_count=vote\_count+1 where id= #{id}
6. DB replica 专门负责query 任务 分担主库压力。

# 项目部署

## 安装Nginx

在Amazon EC2 中安装 Nginx:

Create File: /etc/yum.repos.d/nginx.repo

[nginx]

name=nginx repo

baseurl=https://nginx.org/packages/centos/7 /$basearch/

gpgcheck=0

enabled=1

如果是centos 7 系统

然后sudo yum update 获得nginx repository

然后 sudo yum install nginx

Sudo service nginx start/stop.

# 整合AE, bodymovin 实现AE 动画转换网页动画

1. 安装bodymovin 后在AE 窗口>扩展程序 选择生成路径json 文件

2. 安装 vue-lotties, npm install –save lottie-vuejs

3. 在需要的vue文件中导入 组件，并在Script 里返回

import LottieAnimation from "lottie-vuejs/src/LottieAnimation.vue";

4. 最后使用组件

注意path 是从public folder开始

1. <lottie-animation
2. path="path/to/your/lottie-animation.json"
3. :loop="false"
4. :autoPlay="true"
5. :loopDelayMin="2.5"
6. :loopDelayMax="5"
7. :speed="1"
8. :width="256"
9. :height="256"
10. @AnimControl="setAnimController"
11. />

# Todos

1. 上线之后要更改Redis 服务
2. ~~上线之后更改MySQL 数据库安全组~~
3. ~~上线测试，点赞功能每天一次~~
4. 注意定时任务每天零点执行一次
5. ~~上线后的路径问题，css~~