



小哥主页地址:<https://space.bilibili.com/471822092>

喜欢的可以三连，谢谢！！！

Java全栈交流群: 1135453115

# 新冠-物资管理系统

## 项目简介及核心技术

### 新冠-物资管理系统简介

SpringBoot+Vue+Element-UI (新冠-物资管理系统) 纯前后端分离项目，抗疫相关物料的管理系统，领用、派发记录，库存查询统计。项目预览地址：<https://www.zykheme.club>



新冠-物资管理系统

III

系统管理 欢迎

用户信息

获取源码 用户中心

| 用户账号   | 用户名称       | 所属部门  | 用户角色  |
|--------|------------|-------|-------|
| xiaoge | NieChangan | 物资管理部 | 超级管理员 |

日历

添加日程

2020年8月

上个月 今天 下个月

| 一     | 二     | 三     | 四     | 五     | 六     | 日     |
|-------|-------|-------|-------|-------|-------|-------|
| 07-27 | 07-28 | 07-29 | 07-30 | 07-31 | 08-01 | 08-02 |
| 08-03 | 08-04 | 08-05 | 08-06 | 08-07 | 08-08 | 08-09 |
| 08-10 | 08-11 | 08-12 | 08-13 | 08-14 | 08-15 | 08-16 |
| 08-17 | 08-18 | 08-19 | 08-20 | 08-21 | 08-22 | 08-23 |
| 08-24 | 08-25 | 08-26 | 08-27 | 08-28 | 08-29 | 08-30 |

公告管理

物资资料 物资库存 物资入库 物资发放

用户登入统计

全部 我

08-14 08-15 08-16

| Date  | 全部 | 我 |
|-------|----|---|
| 08-14 | 1  | 0 |
| 08-15 | 4  | 2 |
| 08-16 | 9  | 5 |

业务管理

健康设备 其他管理 日志管理

The screenshot shows the 'Role Management' interface. On the left, there's a sidebar with various system management options like 'Welcome Page', 'User Management', 'Attachment Management', etc. The main area displays a table of roles with columns for ID, Role Name, and a permission tree. A modal window is open over the table, titled 'Assign Menu Permissions', showing the detailed permission assignments for the selected role.

| ID  | 角色名    | 权限树  | 操作  |
|-----|--------|--|---|
| 0   | admin3 | <ul style="list-style-type: none"><li><input checked="" type="checkbox"/> 系统管理</li><li><input checked="" type="checkbox"/> 欢迎页面</li><li><input checked="" type="checkbox"/> 用户管理</li><li><input type="checkbox"/> 附件管理</li></ul> | <a href="#">由授权</a> <a href="#">编辑</a> <a href="#">删除</a> |
| 77  | Redis监 | <ul style="list-style-type: none"><li><input type="checkbox"/> 菜单权限</li><li><input type="checkbox"/> 角色管理</li><li><input type="checkbox"/> 部门管理</li><li><input checked="" type="checkbox"/> 公告管理</li></ul>                       | <a href="#">由授权</a> <a href="#">编辑</a> <a href="#">删除</a> |
| 78  | 系统监    | <ul style="list-style-type: none"><li><input type="checkbox"/> 业务管理</li><li><input type="checkbox"/> 健康报备</li><li><input type="checkbox"/> 其他管理</li><li><input type="checkbox"/> 日志管理</li></ul>                                  | <a href="#">由授权</a> <a href="#">编辑</a> <a href="#">删除</a> |
| 79  | 跑批人    | <ul style="list-style-type: none"><li><input type="checkbox"/> 业务管理</li><li><input type="checkbox"/> 健康报备</li><li><input type="checkbox"/> 其他管理</li><li><input type="checkbox"/> 日志管理</li></ul>                                  | <a href="#">由授权</a> <a href="#">编辑</a> <a href="#">删除</a> |
| 80  | 开发人    | <ul style="list-style-type: none"><li><input type="checkbox"/> 业务管理</li><li><input type="checkbox"/> 健康报备</li><li><input type="checkbox"/> 其他管理</li><li><input type="checkbox"/> 日志管理</li></ul>                                  | <a href="#">由授权</a> <a href="#">编辑</a> <a href="#">删除</a> |
| 125 | 测试用    |  | <a href="#">由授权</a> <a href="#">编辑</a> <a href="#">删除</a> |
| 126 | 蔡徐坤    |  | <a href="#">由授权</a> <a href="#">编辑</a> <a href="#">删除</a> |

分配菜单权限

取消 [授权](#)

The screenshot shows the Swagger UI interface for the COVID-19 Material Management System. The left sidebar contains navigation links for System Management, Business Management, Health Reporting, Other Management, Monitoring Management, Personal Blogs, Swagger Documentation, Icon Management, and Log Management. The main content area has a green header bar with the title "swagger" and a dropdown menu for "default (/v2/api-docs)". On the right side of the header is a "Explore" button. Below the header, the title "新冠物资管理系统 API 文档" is displayed, followed by the subtitle "talk is cheap , show me the code~". The main content lists various API endpoints categorized by controller:

| 接口描述                                 | Show/Hide | List Operations | Expand Operations |
|--------------------------------------|-----------|-----------------|-------------------|
| 菜单权限接口 : Menu Controller             | Show/Hide | List Operations | Expand Operations |
| 登入日志接口 : Login Log Controller        | Show/Hide | List Operations | Expand Operations |
| 健康上报接口 : Health Controller           | Show/Hide | List Operations | Expand Operations |
| 文件上传接口 : File Upload Controller      | Show/Hide | List Operations | Expand Operations |
| 物资出库接口 : Out Stock Controller        | Show/Hide | List Operations | Expand Operations |
| 物资来源接口 : Supplier Controller         | Show/Hide | List Operations | Expand Operations |
| 物资类别接口 : Product Category Controller | Show/Hide | List Operations | Expand Operations |
| 物资去向接口 : Consumer Controller         | Show/Hide | List Operations | Expand Operations |
| 物资入库接口 : In Stock Controller         | Show/Hide | List Operations | Expand Operations |
| 物资资料接口 : Product Controller          | Show/Hide | List Operations | Expand Operations |

新冠-物资管理系统

系统管理 / 菜单管理

输入关键字进行过滤

菜单权限树

|        | 权限               | 操作           |
|--------|------------------|--------------|
| + 系统管理 | 菜单               | 编辑 + 增加 白 删除 |
| + 欢迎页面 | 菜单               | 编辑 + 增加 白 删除 |
| + 用户管理 | 菜单               | 编辑 + 增加 白 删除 |
| + 用户添加 | 权限 [user.add]    | 编辑 + 增加 白 删除 |
| + 用户删除 | 权限 [user.delete] | 编辑 + 增加 白 删除 |
| + 用户编辑 | 权限 [user.edit]   | 编辑 + 增加 白 删除 |
| + 禁用用户 | 权限 [user.status] | 编辑 + 增加 白 删除 |
| + 用户更新 | 权限 [user.update] | 编辑 + 增加 白 删除 |
| + 导出表格 | 权限 [user.export] | 编辑 + 增加 白 删除 |
| + 分配角色 | 权限 [user.assign] | 编辑 + 增加 白 删除 |
| + 附件管理 | 菜单               | 编辑 + 增加 白 删除 |
| + 菜单权限 | 菜单               | 编辑 + 增加 白 删除 |
| + 角色管理 | 菜单               | 编辑 + 增加 白 删除 |
| + 部门管理 | 菜单               | 编辑 + 增加 白 删除 |
| + 公告管理 | 菜单               | 编辑 + 增加 白 删除 |
| + 业务管理 | 菜单               | 编辑 + 增加 白 删除 |
| + 健康报备 | 菜单               | 编辑 + 增加 白 删除 |
| + 其他管理 | 菜单               | 编辑 + 增加 白 删除 |
| + 日志管理 | 菜单               | 编辑 + 增加 白 删除 |

新冠-物资管理系统

系统管理 / 业务管理

添加

| # | 分类名称 | 排序 | 创建时间                | 修改时间                | 备注          | 层级   | 操作 |
|---|------|----|---------------------|---------------------|-------------|------|----|
| 1 | 交通运输 | 1  | 2020-03-17 00:00:00 | 2020-05-16 17:43:39 | 交通运输, . . . | 一级分类 |    |
| 2 | 医疗物资 | 2  | 2020-03-17 00:00:00 | 2020-03-22 23:00:38 | 11111122    | 一级分类 |    |
| 3 | 生活用品 | 3  | 2020-03-17 00:00:00 | 2020-03-22 23:00:43 | 生活用品        | 一级分类 |    |
|   | 衣服用品 | 2  | 2020-03-17 11:38:50 | 2020-03-17 11:38:50 | 衣服用品        | 二级分类 |    |
|   | 衣服   | 1  | 2020-03-18 01:50:33 | 2020-03-18 01:50:33 | 衣服          | 三级分类 |    |
|   | 粮油类  | 2  | 2020-03-17 11:51:29 | 2020-03-17 11:51:29 | 粮油类         | 二级分类 |    |
|   | 大米   | 2  | 2020-03-18 01:50:49 | 2020-03-18 01:50:49 | 大米          | 三级分类 |    |
|   | 食用油  | 2  | 2020-03-25 11:13:39 | 2020-03-25 11:13:39 | 食用油         | 三级分类 |    |
|   | 洗化用品 | 3  | 2020-03-17 11:51:50 | 2020-03-17 11:51:50 | 洗化用品        | 二级分类 |    |
| 4 | test | 4  | 2020-03-23 10:46:29 | 2020-03-23 10:46:29 | test        | 一级分类 |    |

共 4 条 5条/页 < > 前往 1 页

新冠-物资管理系统

系统管理 / 其他管理 / 个人博客

Nie Changan

生如蝼蚁,当有鸿鹄之志

主页  
个人简历  
Linux  
Java

所有文章/ 友链/ 关于我

SpringCloud Alibaba

SpringCloud Alibaba

作者:Nie Changan 转载请注明出处

Spring Cloud Alibaba 致力于提供微服务开发的一站式解决方案。此项目包含开发分布式应用微服务的必需组件，方便开发者通过 Spring Cloud 编程模型轻松使用这些组件来开发分布式应用服务。

依托 Spring Cloud Alibaba，您只需要添加一些注解和少量配置，就可以将 Spring Cloud 应用接入阿里分布式应用解决方案，通过阿里中间件来迅速搭建分布式应用系统。

Spring Cloud Netflix

目前市场上主流的 第一套微服务架构解决方案：Spring Boot + Spring Cloud Netflix

Spring Cloud 为开发人员提供了快速构建分布式系统中一些常见模式的工具（例如配置管理，服务发现，断路器，智能路由，微代理，控制总线）。分布式系统的协调导致了样板模式，使用 Spring Cloud 开发人员可以快速地支持实现这些模式的服务和应用程序。他们将在任何分布式环境中运行良好，包括

## 技术交流群

- QQ群: 1135453115
- 微信号: 17670753912 加微信【备注: 新冠-物资管理系统】联系作者
- 源码, 工具可以通过群下载

## 在线体验

### 预览地址

<https://www.zykhome.club>

接口文档 <https://www.zykhome.club:8081/swagger-ui.html>

后台项目地址 <https://github.com/zykzhangyukang/Xinguang>

前端项目地址 <https://github.com/zykzhangyukang/Vue-Xinguang>

## 核心技术

- 核心框架: Spring Boot2.x
- 权限框架: Spring Security5.x
- 持久层框架: MyBatis-Plus、 MyBatis
- 数据库连接池: Alibaba Druid
- 缓存框架: Redis3.x
- 日志管理: LogBack
- 工具类: Apache Commons、 HuTools、 joda-time
- 视图框架: Spring MVC
- 定时器: Quartz
- 数据库连接池: Druid
- 日志管理: Logback
- 前端框架: Vue
- 前端组件库: Element-UI
- 短信服务: 阿里云短信服务
- 分布式文件系统: 阿里对象存储OSS
- Excel导入导出: 阿里EasyExcel
- API文档: Swagger2
- 数据库: MySQL8
- 版本控制: Git
- 持续集成: Jenkins(暂定)
- 可视化图表: Apache Echarts

## 系统结构

```
1 | xinguan-parent //聚合支付
2 |   —— xinguan-base-common //通用工具类、配置类
3 |   |
4 |   —— xinguan-base-web //基础API服务
```

## 学习收获

- 掌握Vue Element 开发后台页面的能力，从而深入理解Vue在后台管理系统中的开发流程；
- 掌握运用Spring Boot开发后台接口的能力；
- 掌握Spring Security开发权限管理的能力；
- 掌握Redis缓存在开发中的运用能力；
- 最终学会用Vue Element Spring Boot 从零开始搭建前后端分离项目的能力，从而更深入的理解系统中整个数据的流向，从哪里来，到哪里去

## 前端项目工具及环境搭建

```
1 | 丑话说在前面，因为本人是后台开发人员，对前端比较好用的工具不太熟练，例如 vscode、webstorm
2 | 所以此项目前端都是使用IDEA进行开发的，不过对于前端开发人员还是推荐使用vscode，因为提示功能强大，编码效率也随之提高不少
```

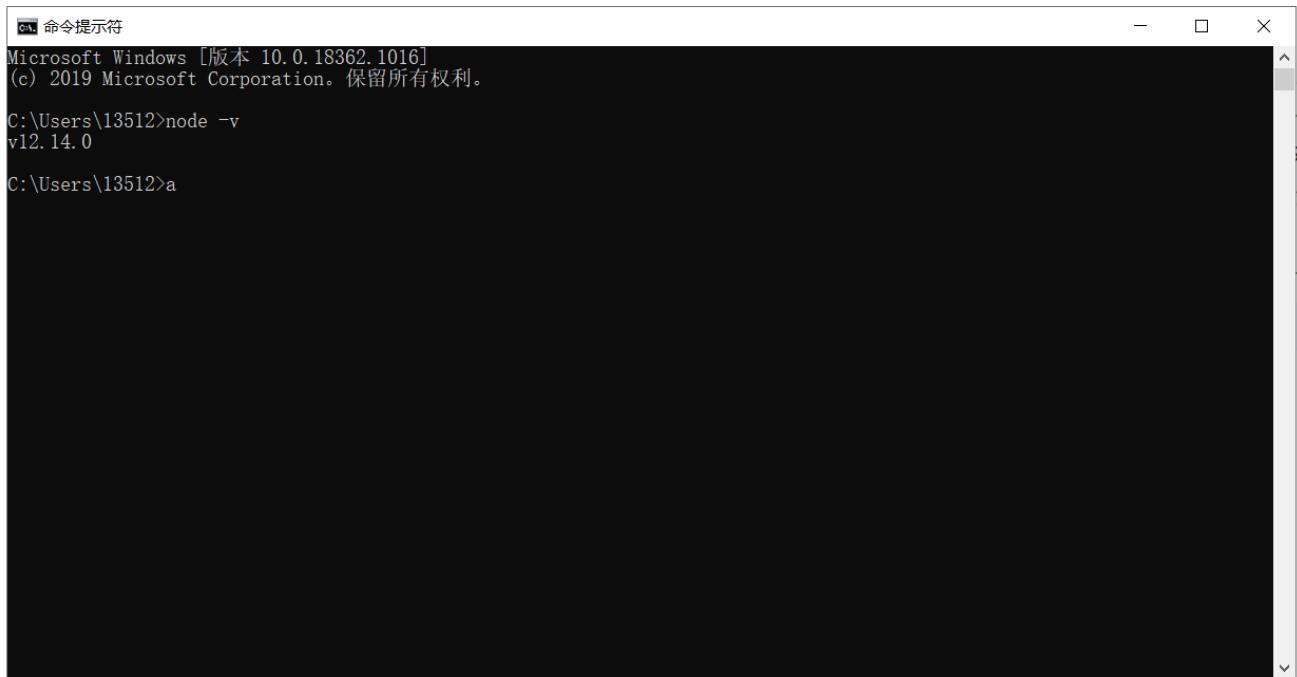
## 安装前端工具

[visual studio code 官网下载地址](#)

[Node.js中文官网下载地址](#)

安装过程这里就省略了...

**查看是否安装成功**



```
命令提示符
Microsoft Windows [版本 10.0.18362.1016]
(c) 2019 Microsoft Corporation. 保留所有权利。

C:\Users\13512>node -v
v12.14.0

C:\Users\13512>a
```

## 安装Vue-Cli脚手架

[Vue CLI脚手架中文官网](#): Vue.js 开发的标准工具

1 | 关于旧版本

2 | vue CLI 的包名称由 vue-cli 改成了 @vue/cli。如果你已经全局安装了旧版本的 vue-cli (1.x 或 2.x)，你需要先通过 npm uninstall vue-cli -g 或 yarn global remove vue-cli 卸载它。

### Node 版本要求

Vue CLI 需要 [Node.js](#) 8.9 或更高版本 (推荐 8.11.0+)。你可以使用 [nvm](#) 或 [nvm-windows](#) 在同一台电脑中管理多个 Node 版本。

可以使用下列任一命令安装这个新的包：

```
1 | npm install -g @vue/cli
2 | # OR
3 | yarn global add @vue/cli
```

安装之后，你就可以在命令行中访问 `vue` 命令。你可以通过简单运行 `vue`，看看是否展示出了一份所有可用命令的帮助信息，来验证它是否安装成功。

你还可以用这个命令来检查其版本是否正确：

```
1 | vue --version
```

可以使用`vue -V`查看版本

```
C:\Users\13512>vue -V
@vue/cli 4.4.6
C:\Users\13512>
```

## 创建前端项目

启动vue ui控制台

```
选择npm
Microsoft Windows [版本 10.0.18362.1016]
(c) 2019 Microsoft Corporation。保留所有权利。
C:\Users\13512>vue ui
□□□ Starting GUI..
□□□ Ready on http://localhost:800
```

启动之后会自动跳转到浏览器,能看到如下界面:

欢迎来到新项目！

这里是项目仪表盘，你可以点击右上方的“自定义”按钮来添加部件。你的改动将会自动保存。

左侧是各个管理页面。在「插件」页面可以添加新的 Vue CLI 插件，「依赖」页面用于管理项目的依赖包，「配置」页面用于配置各种工具，「任务」页面用于运行各个脚本（比如 webpack 打包）。

点击左上方的下拉菜单或状态栏上的小房子按钮来返回到项目管理器。

然后创建项目

创建新项目

项目文件夹  
xinguang-demo

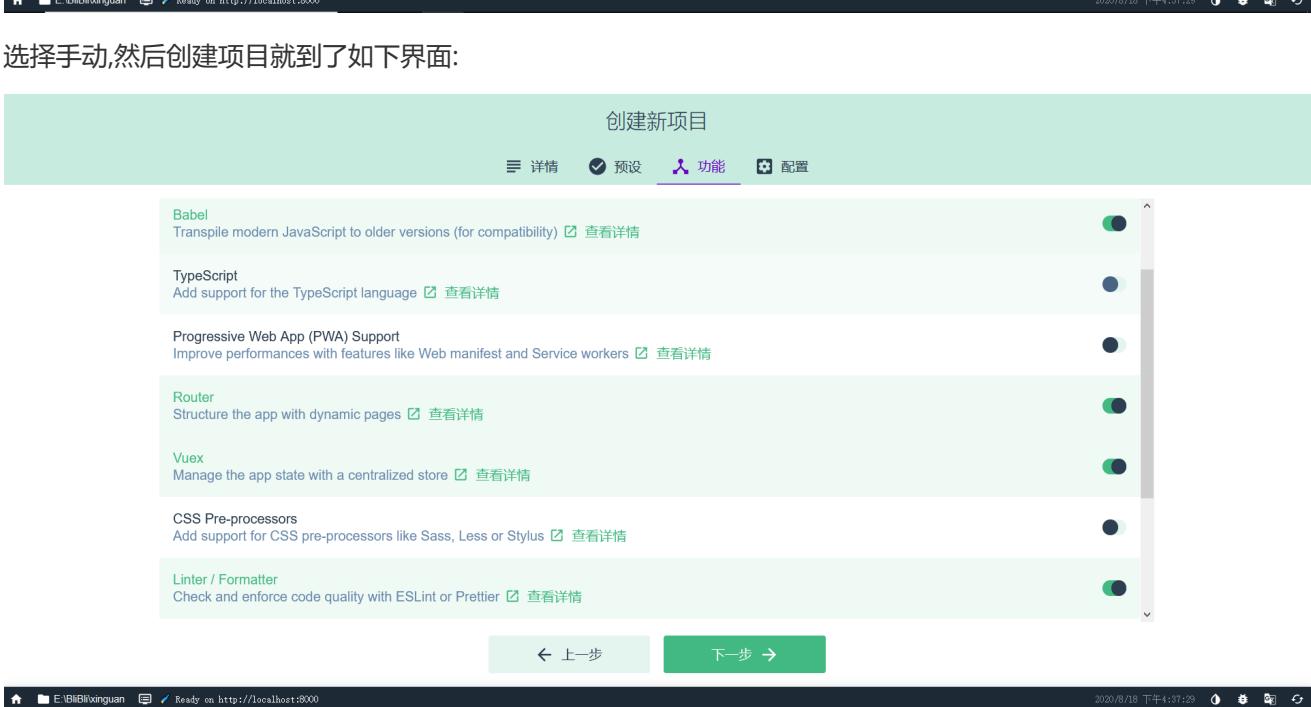
E:\BiliBili\xinguang\xinguang-demo

包管理器  
npm

更多选项  
若目标文件夹已存在则将其覆盖  
无新手指引的脚手架项目

Git  
初始化 git 仓库 (建议)  
初始化新冠疫情管理系统前端部分

取消 下一步 →





同学们选的和我一样的就行了,还有一个ESLint,你们可以自由选择,反正我是不喜欢那个代码提示,太烦了,我先选上,后面我会取消的,这里我给你们解释一下这些选项是什么意思。

## 选项说明

- 1 Babel: 将ES6编译成ES5
- 2
- 3 TypeScript: 使用TypeScript
- 4
- 5 Router和Vuex: 路由和状态管理
- 6
- 7 Linter/ Formatter: 代码检查工具
- 8
- 9 CSS Pre-processors: css预编译

然后继续下一步,看到如下页面:





创建项目可能需要点时间,请耐心等待...

在等待的过程中出现了意外,创建项目很久,你们可以回到vue ui小黑板进行手动加速(自己发现的)

```
选择npm
Microsoft Windows [版本 10.0.18362.1016]
(c) 2019 Microsoft Corporation。保留所有权利。
C:\Users\13512>vue ui
[####] Starting GUI..
[####] Ready on http://localhost:8000
[#####.....] \ refresh-package-json: tweetnacl: sill refresh-package-json E:\BliBli\xinguan\xinguan-vue\node_mod
```

在这里可以多敲几次回车,好像会快很多...

项目创建成功之后会跳转到这个界面:



如果能看到如下界面,那么恭喜你,快跟上我的步伐了



## Welcome to Your Vue.js App

For a guide and recipes on how to configure / customize this project,  
check out the [vue-cli documentation](#).

### Installed CLI Plugins

[babel](#) [router](#) [vuex](#) [eslint](#)

### Essential Links

[Core Docs](#) [Forum](#) [Community Chat](#) [Twitter](#) [News](#)

### Ecosystem

[vue-router](#) [vuex](#) [vue-devtools](#) [vue-loader](#) [awesome-vue](#)

ESLint是个很烦的东西,不喜欢的可以把它停掉

The screenshot shows the Vuetify configuration interface for a project named 'xinguan-vue'. On the left sidebar, the '配置' (Configuration) option is selected. In the main panel, there are two sections: 'Vue CLI' and 'ESLint configuration'. The 'ESLint configuration' section is highlighted with a red box and a red number '2'. It contains a sub-section for '保存时检查' (Check on save) which is also highlighted with a red box and a red number '3'. A dropdown menu next to it shows '必要(默认)' (Necessary (Default)). At the bottom right of the main panel, there are '取消修改' (Cancel changes) and '保存' (Save) buttons, with the '保存' button also having a red box and a red number '4' over it. The status bar at the bottom shows the path 'E:\Bilibili\xinguan\xinguan-vue' and the message 'INFO Task E:\Bilibili\xinguan\xinguan-vue:serve was terminated'.

将项目导入到IDEA中

Project: xinguan-vue [E:\BiliBili\xinguan\xinguan-vue] - ...\\src\\store\\index.js [xinguan-vue] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

router\\index.js store\\index.js

ESLint: Type Error: 'options' is undefined (no-undef)

```

1 import Vue from 'vue'
2 import Vuex from 'vuex'
3
4 Vue.use(Vuex)
5
6 export default new Vuex.Store(options: {
7   state: {
8     },
9     mutations: {
10    },
11    actions: {
12      },
13    modules: {
14      }
15  })
16

```

Event Log

## 安装Element UI

[Element UI中文官网](#)

赞助商

更新日志

Element React

Element Angular

开发指南

安装 2

快速上手

国际化

自定义主题

内置过渡动画

组件

正在传输来自 static.codepen.io 的数据...

安装

npm 安装

推荐使用 npm 的方式安装，它能更好地和 webpack 打包工具配合使用。

```
npm i element-ui -S
```

CDN

目前可以通过 [unpkg.com/element-ui](https://unpkg.com/element-ui) 获取到最新版本的资源，在页面上引入 js 和 css 文件即可开始使用。

```
<!-- 引入样式 -->
<link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-chalk/index.css">
<!-- 引入组件库 -->
<script src="https://unpkg.com/element-ui/lib/index.js"></script>
```

我们建议使用 CDN 引入 Element 的用户在链接地址上锁定版本，以免将来 Element 升级时受到非兼容性更新的影响。锁定版本的方法请查看 [unpkg.com](https://unpkg.com)。

## npm 安装

推荐使用 npm 的方式安装，它能更好地和 [webpack](#) 打包工具配合使用。

```

1 #npm安装组件可能比较慢,建议配置国内加速器(这里以淘宝为例)
2 npm config set registry https://registry.npm.taobao.org/
3 npm i element-ui -S

```

## 引入 Element

你可以引入整个 Element，或是根据需要仅引入部分组件。我们先介绍如何引入完整的 Element。

## 完整引入

在 main.js 中写入以下内容：

```
1 import Vue from 'vue';
2 import ElementUI from 'element-ui';
3 import 'element-ui/lib/theme-chalk/index.css';
4 import App from './App.vue';
5
6 Vue.use(ElementUI);
7
8 new Vue({
9   el: '#app',
10  render: h => h(App)
11});
```

以上代码便完成了 Element 的引入。需要注意的是，样式文件需要单独引入。

当然也可以按需导入，具体参考官网

以下是我的配置：

### main.js

```
1 import Vue from 'vue'
2 import App from './App.vue'
3 import router from './router'
4 import ElementUI from 'element-ui';
5 import 'element-ui/lib/theme-chalk/index.css';
6 import store from './store'
7
8 Vue.use(ElementUI);
9 Vue.config.productionTip = false
10
11 new Vue({
12   router,
13   store,
14   render: h => h(App)
15 }).$mount('#app')
```

## About.vue

```
1 <template>
2   <div class="about">
3     <h1>This is an about page</h1>
4     <el-button type="primary">点击有美女</el-button>
5   </div>
6 </template>
```

启动测试：

## This is an about page

点击有美女

文档由小哥(安哥)原创,禁止商用,有问题可以添加交流群: 1135453115

作为一个靓仔,我已经喜欢了孤独....

## 登录页面布局

**设置盒子的高度100% ,内边距、外边距为 0**

在assets下新建css的文件夹，文件夹中新建一个global.css文件

**global.css**

```
1 /*全局样式表*/
2 html, body, #app{
3     height: 100%;
4     margin: 0;
5     padding: 0;
6 }
```

views下新建登录组件Login.vue

**Login.vue**

```
1 <template>
2
3 </template>
4
5 <script>
6   export default {
7     name: 'Login'
8   }
9 </script>
10
11 <style scoped>
12
13 </style>
```

## 安装less-loader

项目中想使用less进行进行样式编辑

```
1 <style lang="less" scoped>
2
3 </style>
```

直接使用会报错:Can't resolve 'less-loader'

打开项目文件夹，终端命令：

```
1 | npm install --save-dev less-loader less
```

如果不想用less，可以将`lang='less'`删除

## 登录页面完整代码

写页面需要同学们有足够的基本功，除了会Vue之外，还要熟练使用Css、Html等基本操作，如果不是很熟练的话可以拷贝我的代码,粘贴即用

### Login.vue

```
1 <template>
2   <!--登录表单的容器-->
3   <div class="login_container">
4     <!--登录区域-->
5     <div class="login_box">
6       <!--头像-->
7       <div class="avatar_box">
8         
9       </div>
```

```
10     <!--表单-->
11     <el-form :model="loginForm" :rules="loginRules" ref="loginForm" label-
12     width="0px" class="login_form">
13         <el-form-item prop="username">
14             <el-input v-model="loginForm.username" placeholder="请输入用户名" prefix-
15             icon="el-icon-user-solid"></el-input>
16         </el-form-item>
17         <el-form-item prop="password">
18             <el-input v-model="loginForm.password" placeholder="请输入登录密码" prefix-
19             icon="el-icon-lock"></el-input>
20         </el-form-item>
21         <el-form-item prop="verifyCode">
22             <div class="verifyCode_box">
23                 <el-input v-model="loginForm.verifyCode" placeholder="请输入手机验证码" prefix-
24                 icon="el-icon-mobile" class="verifyCode"></el-input>
25                 
26             </div>
27         </el-form-item>
28     </el-form>
29     </div>
30 </div>
31 </template>
32
33 <script>
34     export default {
35         name: 'Login',
36         data() {
37             return {
38                 loginForm: {
39                     username: '',
40                     password: '',
41                     verifyCode: ''
42                 },
43                 loginRules: {
44                     username: [
45                         { required: true, message: '请输入用户名', trigger: 'blur' },
46                         { min: 3, max: 16, message: '长度在 3 到 16 个字符', trigger: 'blur' }
47                     ],
48                     password: [
49                         { required: true, message: '请输入登录密码', trigger: 'blur' },
50                         { min: 3, max: 16, message: '长度在 3 到 16 个字符', trigger: 'blur' }
51                     ],
52                     verifyCode: [
53                         { required: true, message: '请输入验证码', trigger: 'blur' }
54                     ]
55                 }
56             };
57         },
58     };
59 
```

```
58 |     methods: {
59 |       submitForm(formName) {
60 |         this.$refs[formName].validate((valid) => {
61 |           if (valid) {
62 |             alert('submit!');
63 |           } else {
64 |             console.log('error submit!!');
65 |             return false;
66 |           }
67 |         });
68 |       },
69 |       resetForm(formName) {
70 |         this.$refs[formName].resetFields();
71 |       }
72 |     }
73 |   }
74 </script>
75
76 <style lang="less" scoped>
77   .login_container {
78     height: 100%;
79     background-color: aquamarine;
80   }
81
82   .login_box {
83     width: 450px;
84     height: 380px;
85     background-color: #FFFFFF;
86     border-radius: 3px;
87     position: absolute;
88     left: 50%;
89     top: 50%;
90     transform: translate(-50%, -50%);
91
92   .avatar_box{
93     width: 130px;
94     height: 130px;
95     border: 1px solid #EEEEEE;
96     border-radius: 50%;
97     padding: 10px;
98     box-shadow: 0 2px 12px 0 rgba(0, 0, 0, 0.1);
99     margin: -65px auto;
100    background-color: #FFFFFF;
101
102    img{
103      width: 100%;
104      height: 100%;
105      border-radius: 50%;
106      background-color: #EEEEEE;
107    }
108  }
109
110  .login_form{
```

```
111     position: absolute;
112     bottom: 0px;
113     width: 100%;
114     padding: 0px 20px;
115     box-sizing: border-box;
116
117     .login_btn{
118         display: flex;
119         justify-content: flex-end;
120     }
121
122     .verifyCode_box{
123         display: flex;
124         .verifyCode{
125             width: 70%;
126             justify-content: left;
127         }
128
129         .verifyCode_img{
130             width: 30%;
131             height: 45px;
132             justify-content: flex-end;
133         }
134     }
135 }
136 }
137 </style>
```

## 修改页面布局

项目中 `Home.vue` 和 `About.vue` 是多余的可以删除

✗  **xinguang-vue** E:\BliBli\xinguang\xinguang-vue

>  node\_modules library root

>  public

✗  src

>  assets

>  components

✗  router

 index.js

>  store

✗  views

 About.vue

 Home.vue

 Login.vue

 Main.vue

 App.vue

 main.js

 .browserslistrc

 .editorconfig

 .eslintrc.js

 .gitignore

 babel.config.js

 package.json

 package-lock.json

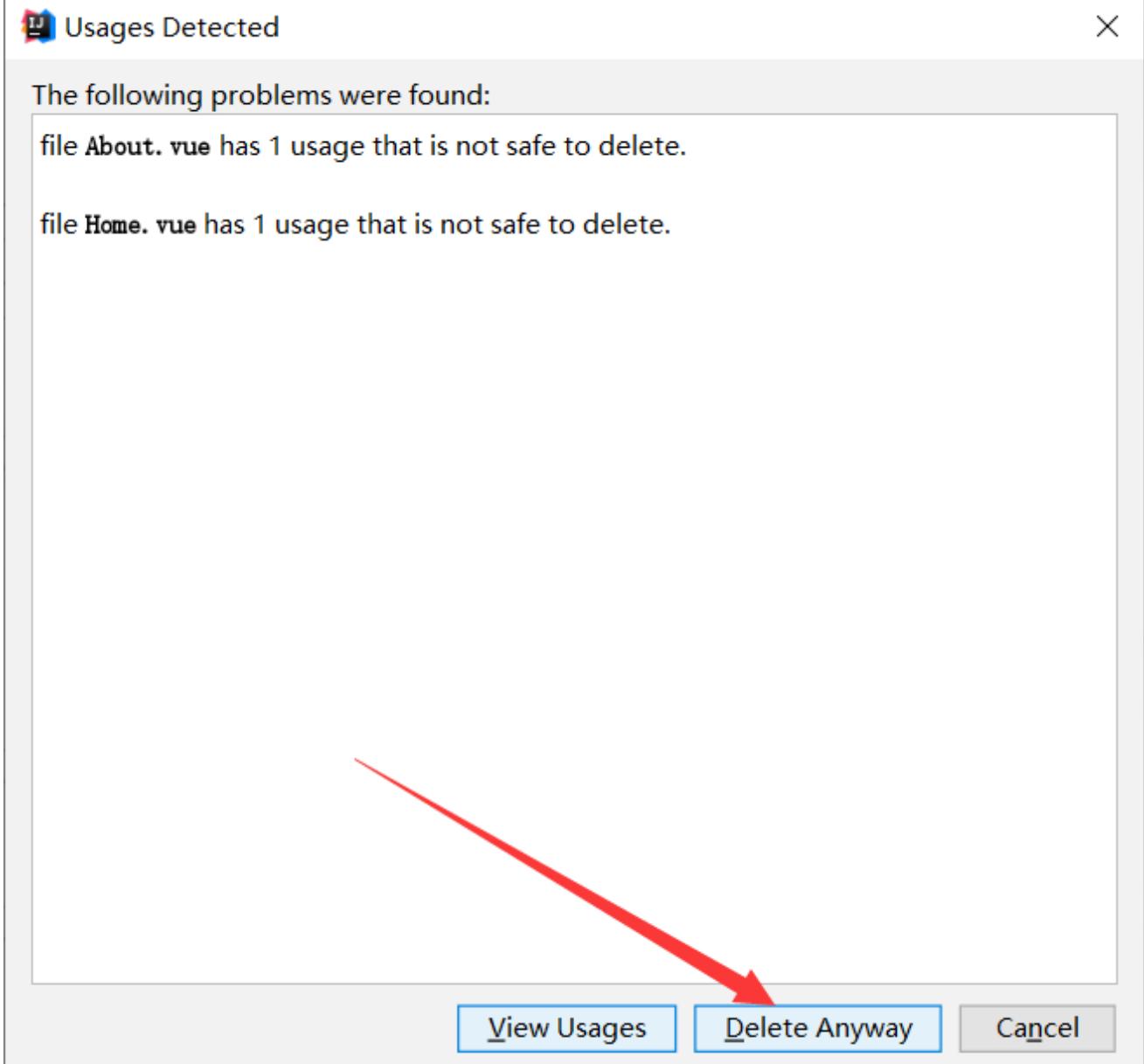
 README.md

 vue.config.js

 External Libraries

>  Scratches and Consoles

删除这两个文件

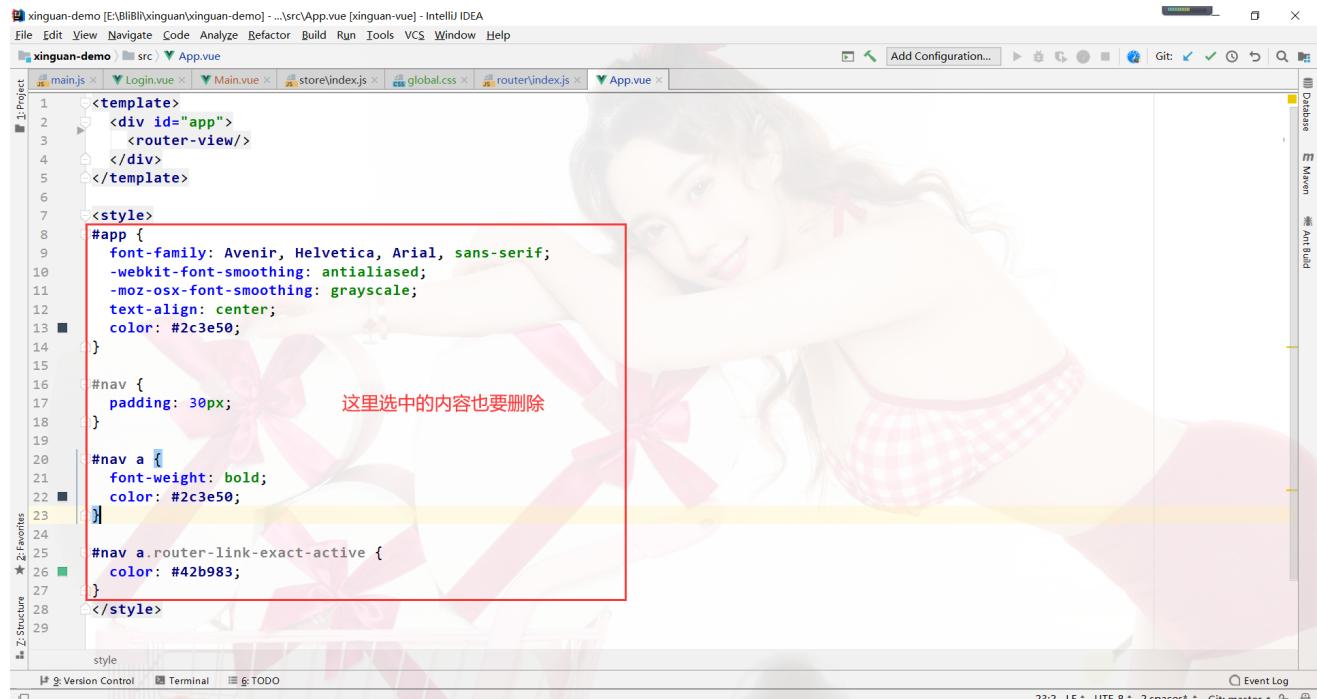


删除 `App.vue` 中 id 为 nav 的 div 标签

```
<template>
  <div id="app">
    <div id="nav">
      <router-link to="/">Home</router-link> | 
      <router-link to="/about">About</router-link> | 
      <router-link to="/login">Login</router-link>
    </div>
    <router-view/>
  </div>
</template>
```

删除选中部分

```
<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
```



```
xinguang-demo [E:\Bilibili\xinguang\xinguang-demo] - ...src\App.vue [xinguang-vue] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
xinguang-demo src App.vue
main.js Login.vue Main.vue store\index.js global.css router\index.js App.vue
1 <template>
2   <div id="app">
3     <router-view/>
4   </div>
5 </template>
6
7 <style>
8 #app {
9   font-family: Avenir, Helvetica, Arial, sans-serif;
10  -webkit-font-smoothing: antialiased;
11  -moz-osx-font-smoothing: grayscale;
12  text-align: center;
13  color: #2c3e50;
14 }
15
16 #nav {
17   padding: 30px;          这里选中的内容也要删除
18 }
19
20 #nav a {
21   font-weight: bold;
22   color: #2c3e50;
23 }
24
25 #nav a.router-link-exact-active {
26   color: #42b983;
27 }
28 </style>
29
style
Event Log
```

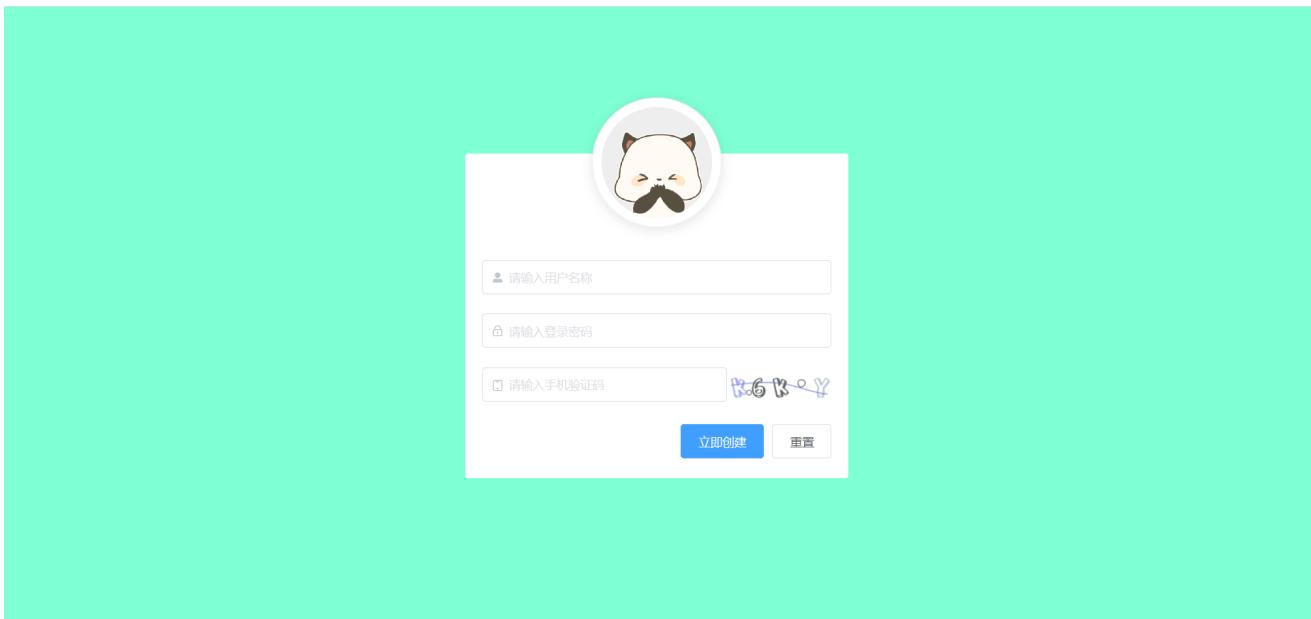
App.vue 最终代码

```
1 <template>
2   <div id="app">
3     <router-view/>
4   </div>
5 </template>
6
7 <style>
8
9 </style>
```

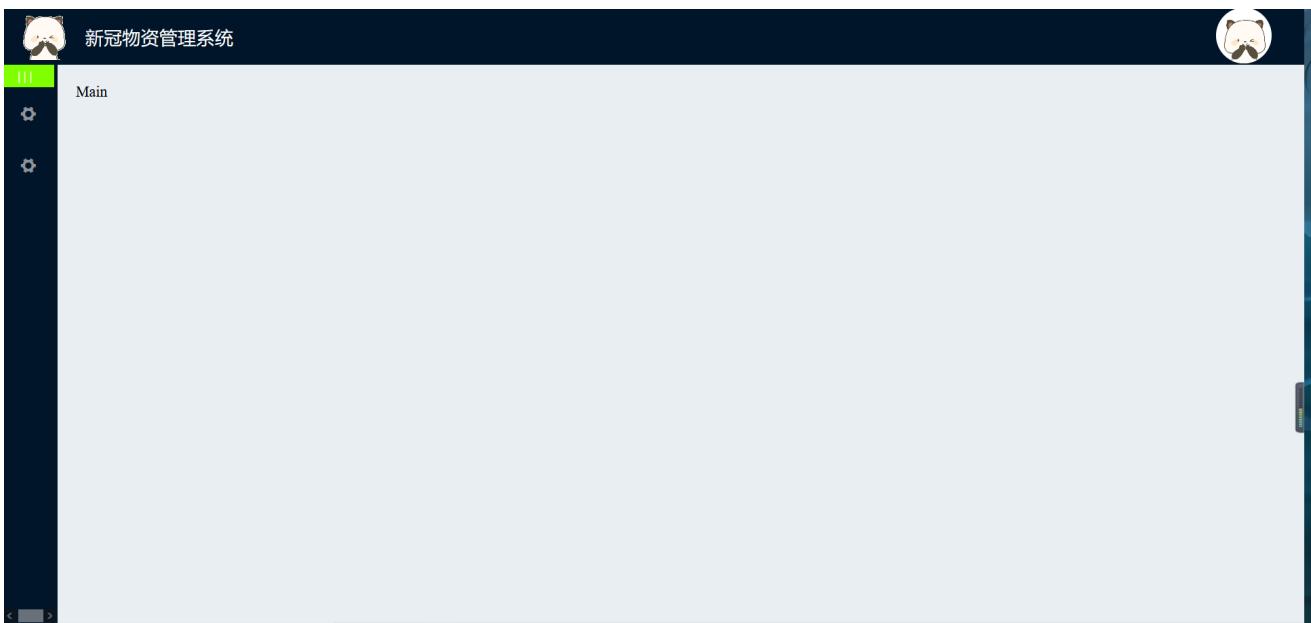
编辑 `router/index.js` 文件,整体代码如下

```
1 import Vue from 'vue'
2 import VueRouter from 'vue-router'
3
4 Vue.use(VueRouter)
5
6 const routes = [
7   {
8     path: '/',
9     redirect: '/login', //重定向到/login
10    },
11    {
12      path: '/login',
13      name: 'Login',
14      component: () => import('../views/Login.vue')
15    }
16  ]
17
18 const router = new VueRouter({
19   routes
20 })
21
22 export default router
```

再次启动项目会发现点击链接直接跳转到 `Login.vue`



## 主界面布局



初步代码如下:

```
1 <template>
2   <el-container class="home-container">
3     <!-- 导航-->
4     <el-header>
5       <div>
6         <span style="margin-left:20px;">
7           
15     </span>
16   </div>
17   <el-dropdown>
18     <div class="block">
19       <el-avatar :size="50" :src="this.userInfo.avatar" style="cursor: pointer;"></el-avatar>
20     </div>
21     <el-dropdown-menu slot="dropdown" trigger="click">
22       <el-dropdown-item>
23         <span type="danger" @click="towelcome"><span class="el-icon-house">
24           &nbsp;系统首页</span>
25         </el-dropdown-item>
26
27       <el-dropdown-item>
28         <span type="danger" @click="getContact"><span class="el-icon-ship">
29           &nbsp;交流讨论</span>
30       </el-dropdown-item>
31
32       <el-dropdown-item>
33         <span type="danger" @click="logout"><span class="el-icon-switch-button"></span> &nbsp;退出登入</span>
34
35       </el-dropdown-item>
36     </el-dropdown-menu>
37   </el-dropdown>
38 </el-header>
39 <!--主体-->
40 <el-container style="height: 500px;">
41   <!--菜单-->
42   <el-aside :width="isOpen==true?'64px':'200px'">
43     <div class="toggle-btn" @click="toggleMenu">|||</div>
44     <el-menu
45       class="el-menu-vertical-demo"
46       :collapse="isOpen"
47       :router="true"
48       :default-active="activePath"
49       background-color="#272c33"
50       :collapse-transition="false"
51       text-color="rgba(255,255,255,0.7)"
52       unique-opened
53     >
54       <MenuTree :menuList="this.menuList"></MenuTree>
55     </el-menu>
56   </el-aside>
57 <!--右边主体-->
```

```
61      <el-main v-loading="loading">
62
63          <router-view></router-view>
64
65      </el-main>
66  </el-container>
67  </el-container>
68</template>
69
70<script>
71 import MenuTree from "./MenuTree.vue"; //引进菜单模板
72
73export default {
74    data() {
75        return {
76            loading: true,
77            activePath: "", //激活的路径
78            isOpen: false,
79            menuList: {},
80            userInfo: {},
81
82        };
83    },
84    components: {
85        MenuTree
86    },
87    methods: {
88        /**
89         *
90         * 退出登入
91         */
92        async logout() {
93            var res = await this.$confirm("此操作将退出系统, 是否继续?", "提示", {
94                confirmButtonText: "确定",
95                cancelButtonText: "取消",
96                type: "warning"
97            }).catch(() => {
98                this.$message({
99                    type: "info",
100                   message: "已取消退出登入"
101                });
102            });
103            if (res == "confirm") {
104                window.localStorage.clear();
105                this.$router.push("/login");
106            }
107        },
108        /**
109         * 去系统首页
110         */
111        towelcome(){
112            this.$router.push("/welcome");
113        },
114    }
115}
```

```
114  /**
115   * 加载菜单数据
116   */
117  async getMenuList() {
118      const { data: res } = await this.$http.get("user/findMenu");
119      if (res.code !== 200)
120          return this.$message.error("获取菜单失败：" + res.msg);
121      this.menuList = res.data;
122  },
123  /**
124   * 获取用户信息
125   */
126  async getUserInfo() {
127      const { data: res } = await this.$http.get("user/info");
128      if (res.code !== 200) {
129          return this.$message.error("获取用户信息失败：" + res.msg);
130      } else {
131          this.userInfo = res.data;
132          //保存用户
133          this.$store.commit("setUserInfo", res.data);
134      }
135  },
136  /**
137   * 菜单伸缩
138   */
139  toggleMenu() {
140      this.isOpen = !this.isOpen;
141  },
142
143  /**
144   * 点击交流
145   */
146  getContact(){
147      const w = window.open('about:blank');
148      w.location.href = 'https://www.zykcoderman.xyz/';
149  }
150
151 },
152 mounted() {
153     this.getUserInfo();
154 },
155 created() {
156     this.getMenuList();
157     this.activePath = window.sessionStorage.getItem("activePath");
158     // if(window.sessionStorage.getItem("activePath")==null){
159     //     this.activePath='/welcome';
160     // }
161     setTimeout(() => {
162         this.loading = false;
163     }, 500);
164 }
165 };
166 </script>
```

```
167 <style>
168 /* 为对应的路由跳转时设置动画效果 */
169
170 .el-header {
171   background-color: #272c33;
172   display: flex;
173   justify-content: space-between;
174   align-items: center;
175   color: #fff;
176   font-size: 19px;
177
178   padding-left: 0px;
179 }
180
181 .el-aside {
182   background-color:#272c33
183 }
184
185 .el-main {
186   background-color: #eaedf1;
187 }
188
189 .home-container {
190   width: 100%;
191   height: 100% !important;
192 }
193
194 .toggle-btn {
195   background-color: #2d3a4b !important;
196   font-size: 10px;
197   line-height: 24px;
198   color: #fff;
199   text-align: center;
200   letter-spacing: 0.2em;
201   cursor: pointer;
202 }
203
204 </style>
```

## 侧边栏布局优化

虽然看上去整体的页面布局没太大问题了,但是我们想一想,菜单栏难道就只有用户列表那两个吗,肯定会有许多数据对不对,而且我们的菜单栏数据肯定是比较庞大的,把所有的内容放一个页面重总感觉不是特别好,所以我们可以将菜单栏部分抽取成一个组件.

创建一个**MenuTree**(菜单树)的组件,将**el-menu**节点下的数据放在**MenuTree**中,具体代码如下

### MenuTree

```
1 <template>
2   <div>
3     <template>
4       <el-submenu index="1">
```

```

5      <template slot="title">
6          <i class="el-icon-location"></i>
7          <span>用户管理</span>
8      </template>
9      <el-menu-item index="1-1">
10         <i class="el-icon-location"></i>
11         <span slot="title">用户管理</span>
12     </el-menu-item>
13 </el-submenu>
14
15     <el-submenu index="2">
16         <template slot="title">
17             <i class="el-icon-location"></i>
18             <span>用户管理</span>
19         </template>
20         <el-menu-item index="2-1">
21             <i class="el-icon-location"></i>
22             <span slot="title">用户管理</span>
23         </el-menu-item>
24     </el-submenu>
25 </template>
26 </div>
27 </template>
28
29 <script>
30     export default {
31         name: 'MenuTree'
32     }
33 </script>
34
35 <style scoped>
36
37 </style>

```

`Main.vue`下el-menu的代码使用**MenuTree**替代,具体操作如下

- 在script标签中引入**MenuTree**组件
- 将**MenuTree**组件添加到当前组件中
- 使用**MenuTree**代替刚刚移出的代码

```

1 <script>
2     /*1.这里引入MenuTree组件*/
3     import MenuTree from '../components/MenuTree'
4
5     export default {
6         name: 'Main',
7         data(){
8             return{
9                 isCollapse: true,
10            }
11        },
12        /*2.将MenuTree组件添加到当前组件中*/
13        components: {

```

```

14     MenuTree
15   },
16   methods:{
17     toggleCollapse(){
18       this.iscollapse = !this.iscollapse
19     },
20     handleOpen(key, keyPath) {
21       console.log(key, keyPath);
22     },
23     handleClose(key, keyPath) {
24       console.log(key, keyPath);
25     }
26   }
27 }
28 </script>

```

```

1 <el-container>
2   <!--侧边栏-->
3   <el-aside :width="iscollapse?'60px':'200px'">
4     <!--展开/收起-->
5     <div class="toggle_box" @click="toggleCollapse">|||</div>
6     <!--菜单栏-->
7     <el-menu
8       default-active="2"
9       class="el-menu-vertical-demo"
10      @open="handleOpen"
11      @close="handleClose"
12      background-color="#001529"
13      text-color="#fff"
14      active-text-color="#ffd04b"
15      :collapse="iscollapse"
16      :collapse-transition="false">
17       <!--这里使用菜单树代替刚刚的代码-->
18       <MenuTree></MenuTree>
19     </el-menu>
20   </el-aside>
21   <el-main>Main</el-main>
22 </el-container>

```

## 侧边栏菜单展示

后台返回的菜单栏数据肯定是通过登录的用户获取菜单的,数据肯定不是写死的,前端请求后台接口返回Json数据,前端对JSON数据进行解析,并且绑定数据,这里为了方便,我们暂且将菜单数据写死,先显示所有的菜单项,复制我下面的代码作为菜单列表

```

1 [
2   {
3     "id": 1,
4     "parentId": 0,

```

```
5     "menuName": "系统管理",
6     "url": "",
7     "icon": "el-icon-setting",
8     "orderNum": 1,
9     "open": 1,
10    "disabled": false,
11    "perms": null,
12    "type": 0,
13    "children": [
14      {
15        "id": 253,
16        "parentId": 1,
17        "menuName": "欢迎页面",
18        "url": "/welcome",
19        "icon": "el-icon-star-off",
20        "orderNum": 1,
21        "open": 0,
22        "disabled": false,
23        "perms": "welcome:view",
24        "type": 0,
25        "children": []
26      },
27      {
28        "id": 226,
29        "parentId": 1,
30        "menuName": "用户管理",
31        "url": "/users",
32        "icon": "el-icon-user",
33        "orderNum": 2,
34        "open": 0,
35        "disabled": false,
36        "perms": "users",
37        "type": 0,
38        "children": []
39      },
40      {
41        "id": 321,
42        "parentId": 1,
43        "menuName": "附件管理",
44        "url": "/attachments",
45        "icon": "el-icon-picture-outline",
46        "orderNum": 2,
47        "open": 1,
48        "disabled": false,
49        "perms": "",
50        "type": 0,
51        "children": []
52      },
53      {
54        "id": 4,
55        "parentId": 1,
56        "menuName": "菜单权限",
57        "url": "/menus",
```

```
58     "icon": "el-icon-help",
59     "orderNum": 3,
60     "open": 0,
61     "disabled": false,
62     "perms": null,
63     "type": 0,
64     "children": []
65   },
66   {
67     "id": 235,
68     "parentId": 1,
69     "menuName": "角色管理",
70     "url": "/roles",
71     "icon": "el-icon-postcard",
72     "orderNum": 3,
73     "open": 0,
74     "disabled": false,
75     "perms": "",
76     "type": 0,
77     "children": []
78   },
79   {
80     "id": 261,
81     "parentId": 1,
82     "menuName": "部门管理",
83     "url": "/departments",
84     "icon": "el-icon-s-home",
85     "orderNum": 3,
86     "open": 0,
87     "disabled": false,
88     "perms": "",
89     "type": 0,
90     "children": []
91   },
92   {
93     "id": 319,
94     "parentId": 1,
95     "menuName": "公告管理",
96     "url": "/notices",
97     "icon": "el-icon-s-flag",
98     "orderNum": 4,
99     "open": 0,
100    "disabled": true,
101    "perms": "",
102    "type": 0,
103    "children": []
104  }
105 ]
106 },
107 {
108   "id": 312,
109   "parentId": 0,
110   "menuName": "业务管理",
```

```
111     "url": null,
112     "icon": "el-icon-s-goods",
113     "orderNum": 2,
114     "open": 0,
115     "disabled": false,
116     "perms": null,
117     "type": 0,
118     "children": [
119       {
120         "id": 229,
121         "parentId": 312,
122         "menuName": "物资管理",
123         "url": "",
124         "icon": "el-icon-date",
125         "orderNum": 1,
126         "open": 1,
127         "disabled": false,
128         "perms": "el-icon-date",
129         "type": 0,
130         "children": [
131           {
132             "id": 230,
133             "parentId": 229,
134             "menuName": "物资入库",
135             "url": "/instocks",
136             "icon": "el-icon-date",
137             "orderNum": 1,
138             "open": 1,
139             "disabled": false,
140             "perms": "el-icon-date",
141             "type": 0,
142             "children": []
143           },
144           {
145             "id": 267,
146             "parentId": 229,
147             "menuName": "物资资料",
148             "url": "/products",
149             "icon": "el-icon-goods",
150             "orderNum": 2,
151             "open": 0,
152             "disabled": false,
153             "perms": "",
154             "type": 0,
155             "children": []
156           },
157           {
158             "id": 268,
159             "parentId": 229,
160             "menuName": "物资类别",
161             "url": "/productCategorys",
162             "icon": "el-icon-star-off",
163             "orderNum": 2,
```

```
164     "open": 0,
165     "disabled": false,
166     "perms": "",
167     "type": 0,
168     "children": []
169   },
170   {
171     "id": 270,
172     "parentId": 229,
173     "menuName": "物资发放",
174     "url": "/outStocks",
175     "icon": "el-icon-goods",
176     "orderNum": 5,
177     "open": 1,
178     "disabled": false,
179     "perms": "",
180     "type": 0,
181     "children": []
182   },
183   {
184     "id": 316,
185     "parentId": 229,
186     "menuName": "物资库存",
187     "url": "/stocks",
188     "icon": "el-icon-tickets",
189     "orderNum": 5,
190     "open": 0,
191     "disabled": false,
192     "perms": "",
193     "type": 0,
194     "children": []
195   }
196 ]
197 },
198 {
199   "id": 311,
200   "parentId": 312,
201   "menuName": "物资流向",
202   "url": null,
203   "icon": "el-icon-edit",
204   "orderNum": 3,
205   "open": 0,
206   "disabled": false,
207   "perms": null,
208   "type": 0,
209   "children": [
210     {
211       "id": 310,
212       "parentId": 311,
213       "menuName": "物资去处",
214       "url": "/consumers",
215       "icon": "el-icon-edit",
216       "orderNum": 1,
```

```
217     "open": 0,
218     "disabled": false,
219     "perms": "",
220     "type": 0,
221     "children": []
222   },
223   {
224     "id": 269,
225     "parentId": 311,
226     "menuName": "物资来源",
227     "url": "/suppliers",
228     "icon": "el-icon-coordinate",
229     "orderNum": 5,
230     "open": 0,
231     "disabled": false,
232     "perms": "",
233     "type": 0,
234     "children": []
235   }
236 ]
237 }
238 ]
239 },
240 {
241   "id": 303,
242   "parentId": 0,
243   "menuName": "健康报备",
244   "url": "",
245   "icon": "el-icon-platform-eleme",
246   "orderNum": 3,
247   "open": 0,
248   "disabled": false,
249   "perms": "",
250   "type": 0,
251   "children": [
252     {
253       "id": 273,
254       "parentId": 303,
255       "menuName": "全国疫情",
256       "url": "/map",
257       "icon": "el-icon-s-opportunity",
258       "orderNum": 1,
259       "open": 1,
260       "disabled": false,
261       "perms": "map:view",
262       "type": 0,
263       "children": []
264     },
265     {
266       "id": 304,
267       "parentId": 303,
268       "menuName": "健康打卡",
269       "url": "/health",
```

```
270     "icon": "el-icon-s-cooperation",
271     "orderNum": 1,
272     "open": 0,
273     "disabled": false,
274     "perms": "",
275     "type": 0,
276     "children": []
277   },
278   {
279     "id": 305,
280     "parentId": 303,
281     "menuName": "查看情况",
282     "url": null,
283     "icon": "el-icon-c-scale-to-original",
284     "orderNum": 2,
285     "open": 1,
286     "disabled": false,
287     "perms": null,
288     "type": 0,
289     "children": []
290   },
291   {
292     "id": 272,
293     "parentId": 303,
294     "menuName": "疫情辟谣",
295     "url": "/rumors",
296     "icon": "el-icon-help",
297     "orderNum": 5,
298     "open": 0,
299     "disabled": false,
300     "perms": null,
301     "type": 0,
302     "children": []
303   }
304 ]
305 },
306 {
307   "id": 295,
308   "parentId": 0,
309   "menuName": "其他管理",
310   "url": "",
311   "icon": "el-icon-s-marketing",
312   "orderNum": 5,
313   "open": 0,
314   "disabled": false,
315   "perms": "",
316   "type": 0,
317   "children": [
318     {
319       "id": 297,
320       "parentId": 295,
321       "menuName": "监控管理",
322       "url": "",
```

```
323     "icon": "el-icon-warning",
324     "orderNum": 1,
325     "open": 0,
326     "disabled": false,
327     "perms": "",
328     "type": 0,
329     "children": [
330       {
331         "id": 298,
332         "parentId": 297,
333         "menuName": "SQL监控",
334         "url": "/druid",
335         "icon": "el-icon-view",
336         "orderNum": 1,
337         "open": 0,
338         "disabled": false,
339         "perms": null,
340         "type": 0,
341         "children": []
342       }
343     ]
344   },
345   {
346     "id": 341,
347     "parentId": 295,
348     "menuName": "个人博客",
349     "url": "/blog",
350     "icon": "el-icon-view",
351     "orderNum": 1,
352     "open": 0,
353     "disabled": false,
354     "perms": "",
355     "type": 0,
356     "children": []
357   },
358   {
359     "id": 296,
360     "parentId": 295,
361     "menuName": "swagger文档",
362     "url": "/swagger",
363     "icon": "el-icon-document",
364     "orderNum": 2,
365     "open": 0,
366     "disabled": false,
367     "perms": null,
368     "type": 0,
369     "children": []
370   },
371   {
372     "id": 318,
373     "parentId": 295,
374     "menuName": "图标管理",
375     "url": "/icons",
```

```
376     "icon": "el-icon-star-off",
377     "orderNum": 2,
378     "open": 1,
379     "disabled": false,
380     "perms": "",
381     "type": 0,
382     "children": []
383   }
384 ]
385 },
386 {
387   "id": 5,
388   "parentId": 0,
389   "menuName": "日志管理",
390   "url": "/logs",
391   "icon": "el-icon-camera",
392   "orderNum": 6,
393   "open": 0,
394   "disabled": false,
395   "perms": null,
396   "type": 0,
397   "children": [
398     {
399       "id": 271,
400       "parentId": 5,
401       "menuName": "登入日志",
402       "url": "/loginLog",
403       "icon": "el-icon-date",
404       "orderNum": 1,
405       "open": 0,
406       "disabled": false,
407       "perms": "login:log",
408       "type": 0,
409       "children": []
410     },
411     {
412       "id": 307,
413       "parentId": 5,
414       "menuName": "操作日志",
415       "url": "/logs",
416       "icon": "el-icon-edit",
417       "orderNum": 1,
418       "open": 1,
419       "disabled": false,
420       "perms": "",
421       "type": 0,
422       "children": []
423     }
424   ]
425 }
426 ]
```

可能有些字段不是特别理解,这里做一下解释

```
1 `id` '菜单/按钮ID',
2 `parentId` '上级菜单ID',
3 `menuName` '菜单/按钮名称',
4 `url` '菜单URL',
5 `perms` '权限标识',
6 `icon` '图标',
7 `type` '类型 0菜单 1按钮',
8 `orderNum` '排序',
9 `disabled` '0: 不可用, 1: 可用',
10 `open` '0:不展开, 1: 展开',
11 `children` '子菜单',
```

接下来对菜单数据进行循环绑定,具体操作如下

- 在Main.vue中定义一个MenuList的属性,属性值为上面的那个JSON数据
  - 将Main.vue中的MenuList数据传递到MenuTree组件中去
  - 在MenuTree中接收传递过来的参数
  - 将传递过来的MenuList数据循环遍历,绑定到菜单栏上,并且添加CSS样式

```
/*1.在Main.vue中定义一个MenuList的属性，属性值为上面的那个JSON数据*/
data() {
    return {
        menuList: [这里存放上面的JSON数据],
    };
},
```

```
1 <!--2.将Main.vue中的MenuList数据传递到MenuTree组件中去-->
2 <MenuTree :menuList="this.menuList"></MenuTree>
```

```
1 <script>
2     export default {
3         name: "MenuTree", //模板名称
4         data() {
5             return {
6                 };
7             },
8             /*3.在MenuTree中接收传递过来的参数*/
9             props: ["menuList", "tagList"],
10            }
11        }
12    </script>
```

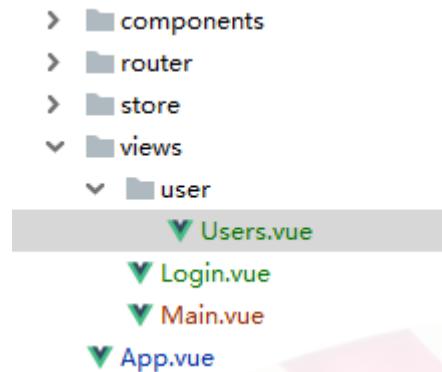
```
1 <!--4.将传递过来的MenuList数据循环遍历,绑定到菜单栏上,并且添加CSS样式-->
2 <template>
3   <div>
4     <template v-for="item in this.menuList">
5       <el-submenu :disabled="item.disabled" :index="item.id+''" v-
6 if="item.children.length>0" :key="item.id+''">
7         <template slot="title" style="padding-left:30px">
8           <i :class="item.icon"></i>
```

```
8     <span slot="title">{{ item.menuName }}</span>
9     </template>
10    <MenuTree :menuList="item.children"></MenuTree>
11    </el-submenu>
12    <el-menu-item
13      v-else
14      :disabled="item.disabled"
15      :index="item.url+''"
16      :route="item.url"
17      @click="savePath(item.url)"
18      :key="item.id+'"
19      style="padding-left: 50px;">
20    >
21      <i :class="item.icon"></i>
22      <span>{{item.menuName}}</span>
23    </el-menu-item>
24  </template>
25 </div>
26 </template>
27 <style>
28 .el-menu--collapse span,
29 .el-menu--collapse i.el-submenu__icon-arrow {
30   height: 0;
31   width: 0;
32   overflow: hidden;
33   visibility: hidden;
34   display: inline-block;
35 }
36 </style>
37 <script>
38 export default {
39   name: "MenuTree", //模板名称
40   data() {
41     return {
42       };
43     },
44   },
45   beforeMount() {},
46   props: ["menuList"],
47   methods: {
48     //保存激活路径
49     savePath(path) {
50       window.sessionStorage.setItem("activePath", path);
51       this.activePath = path;
52     },
53   },
54   created(){
55   }
56 };
57 </script>
58
```

## 用户管理布局

在views下新建一个文件夹user

user文件夹下新建一个Users.vue文件



修改router文件夹下的index.js文件,给main节点增加一个子节点children

```
1  {
2      path: '/main',
3      name: 'Main',
4      component: () => import('../views/Main.vue'),
5      children: [
6          {
7              path: '/users',
8              component: () => import('../views/user/Users.vue'),
9              meta:{title: '用户管理'},
10         }
11     ]
12 }
```

修改Main.vue, 在el-main中添加一个标签

```
1 <el-main>
2     <!--路由跳转的位置-->
3     <router-view></router-view>
4 </el-main>
```

移除.el-main的样式修改为

```

1  /*内容主体*/
2  .el-main {
3      background-color: #E9EEF3;
4      color: #333;
5      text-align: center;
6      line-height: 160px;
7  }
8
9  改成
10 /*内容主体*/
11 .el-main {
12     background-color: #E9EEF3;
13 }

```

`Users.vue`下的布局：

```

1 <template>
2   <div>
3     <el-breadcrumb separator="/" style="padding-left:10px;padding-
bottom:10px;font-size:12px;">
4       <el-breadcrumb-item :to="{ path: '/main' }">首页</el-breadcrumb-item>
5       <el-breadcrumb-item>系统管理</el-breadcrumb-item>
6       <el-breadcrumb-item>用户管理</el-breadcrumb-item>
7     </el-breadcrumb>
8     <!-- 用户列表卡片区 -->
9     <el-card>
10    <el-form :inline="true" :model="formInline">
11      <el-form-item label="部门" label-width="70px">
12        <el-select clearable v-model="formInline.user" placeholder="请选择" >
13          <el-option
14            v-for="item in cities"
15            :key="item.value"
16            :label="item.label"
17            :value="item.value">
18            <span style="float: left">{{ item.label }}</span>
19            <span style="float: right; color: #8492a6; font-size: 13px">{{
item.value }}</span>
20          </el-option>
21        </el-select>
22      </el-form-item>
23      <el-form-item label="用户名" label-width="70px">
24        <el-select clearable v-model="formInline.region" placeholder="活动区域">
25          <el-option label="区域一" value="shanghai"></el-option>
26          <el-option label="区域二" value="beijing"></el-option>
27        </el-select>
28      </el-form-item>
29      <el-form-item label="邮箱" label-width="70px">
30        <el-input clearable v-model="formInline.name"></el-input>
31      </el-form-item>
32      <el-form-item label="性别" label-width="70px">
33        <el-radio v-model="radio" label="1">备选项</el-radio>
34        <el-radio v-model="radio" label="2">备选项</el-radio>

```

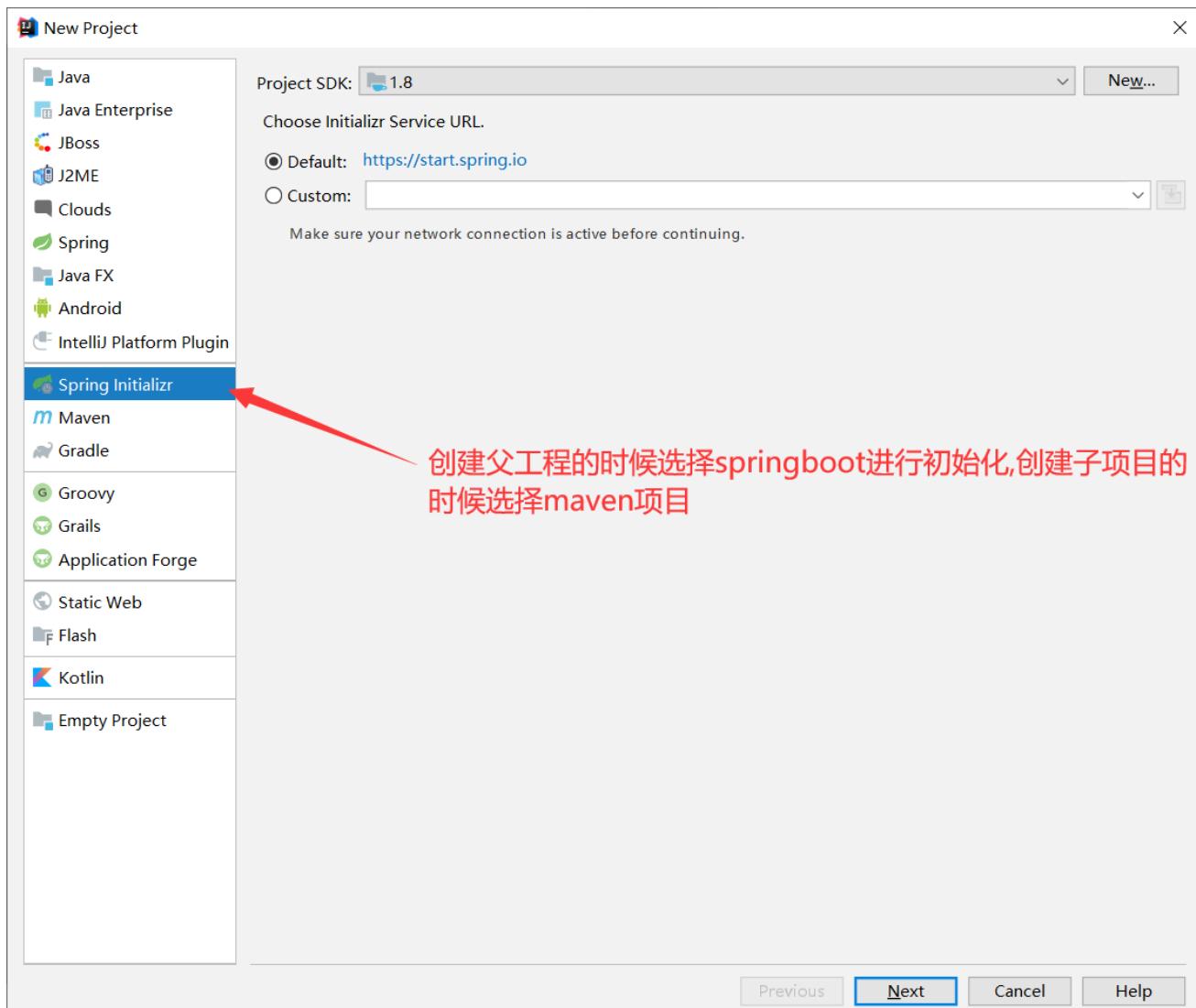
```
35         <el-radio v-model="radio" label="2">备选项</el-radio>
36     </el-form-item>
37     <el-form-item label="昵称" label-width="70px">
38         <el-input clearable v-model="formInline.name"></el-input>
39     </el-form-item>
40     <el-form-item style="margin-left:50px;">
41         <el-button @click="" icon="el-icon-refresh">重置</el-button>
42         <el-button type="primary" @click="" icon="el-icon-search">查询</el-
button>
43         <el-button
44             type="success"
45             icon="el-icon-plus"
46             @click=""
47             >添加</el-button>
48         <el-button @click="downExcel" icon="el-icon-download">导出</el-button>
49     </el-form-item>
50 </el-form>
51 <!--表格-->
52 <el-table
53     :data="tableData"
54     border
55     style="width: 100%;height: 390px">
56     <el-table-column
57         prop="date"
58         label="日期"
59         width="180">
60     </el-table-column>
61     <el-table-column
62         prop="name"
63         label="姓名"
64         width="180">
65     </el-table-column>
66     <el-table-column
67         prop="address"
68         label="地址">
69     </el-table-column>
70 </el-table>
71 <el-pagination
72     style="margin-top:10px;" 
73     @size-change="handleSizeChange"
74     @current-change="handleCurrentChange"
75     :current-page="currentPage4"
76     :page-sizes="[100, 200, 300, 400]"
77     :page-size="100"
78     layout="total, sizes, prev, pager, next, jumper"
79     :total="500">
80     </el-pagination>
81 </el-card>
82 </div>
83 </template>
84
85 <script>
86     export default {
```

```
87 |     name: 'Users',
88 |     data () {
89 |       return {
90 |         radio: '1',
91 |         formInline: {
92 |           user: '',
93 |           region: '',
94 |           name: ''
95 |         },
96 |         cities: [{{
97 |           value: 'Beijing',
98 |           label: '北京'
99 |         }, {
100 |           value: 'Shanghai',
101 |           label: '上海'
102 |         }, {
103 |           value: 'Nanjing',
104 |           label: '南京'
105 |         }, {
106 |           value: 'Chengdu',
107 |           label: '成都'
108 |         }, {
109 |           value: 'Shenzhen',
110 |           label: '深圳'
111 |         }, {
112 |           value: 'Guangzhou',
113 |           label: '广州'
114 |         }],
115 |         tableData: [{{
116 |           date: '2016-05-02',
117 |           name: '王小虎',
118 |           address: '上海市普陀区金沙江路 1518 弄'
119 |         }, {
120 |           date: '2016-05-04',
121 |           name: '王小虎',
122 |           address: '上海市普陀区金沙江路 1517 弄'
123 |         }, {
124 |           date: '2016-05-01',
125 |           name: '王小虎',
126 |           address: '上海市普陀区金沙江路 1519 弄'
127 |         }, {
128 |           date: '2016-05-03',
129 |           name: '王小虎',
130 |           address: '上海市普陀区金沙江路 1516 弄'
131 |         }],
132 |         currentPage4: 4
133 |       }
134 |     },
135 |     methods: {
136 |       onSubmit () {
137 |         console.log('submit!')
138 |       },
139 |       handleSizeChange(val) {
```

```
140     console.log(`每页 ${val} 条`);
141   },
142   handleCurrentChange(val) {
143     console.log(`当前页: ${val}`);
144   }
145 }
146 </script>
147 <style scoped>
148
149 </style>
150
151
```

## 搭建后台工程

### 父工程 xinguang-parent

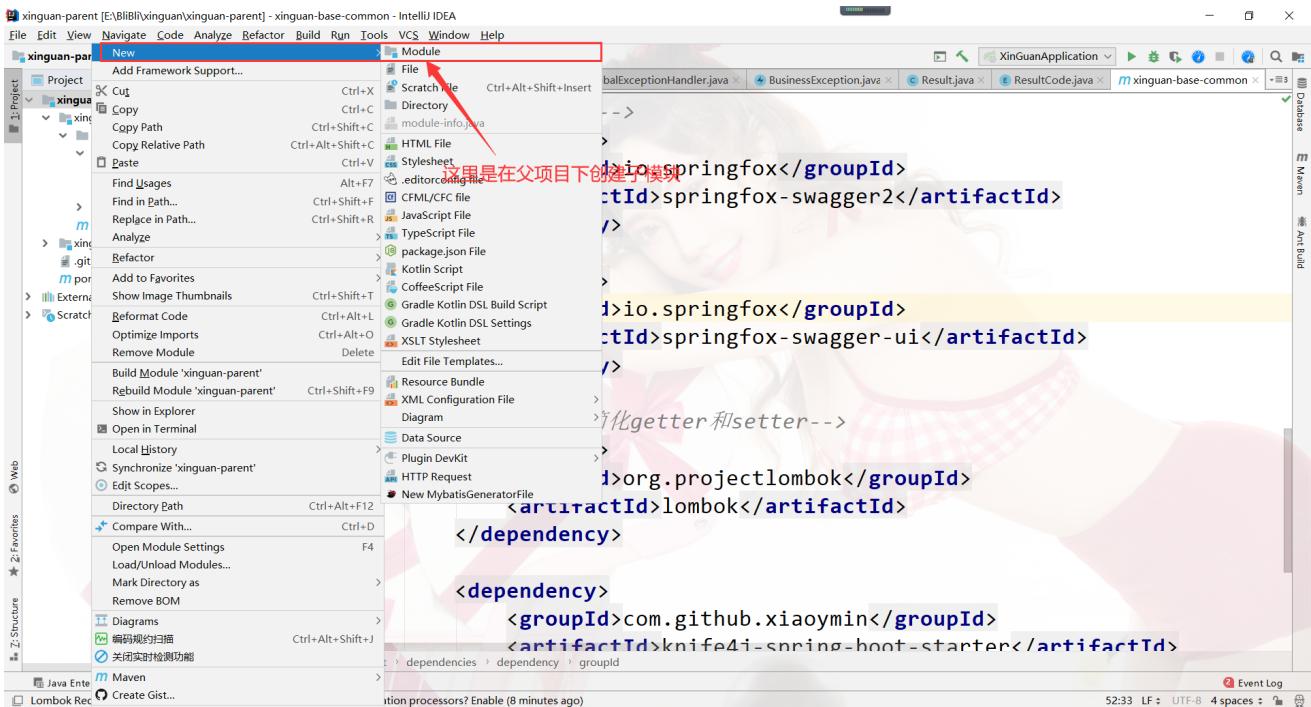


### pom.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5       https://maven.apache.org/xsd/maven-4.0.0.xsd">
6   <modelVersion>4.0.0</modelVersion>
7   <packaging>pom</packaging>
8   <modules>
9     <module>xinguang-base-common</module>
10    <module>xinguang-base-web</module>
11  </modules>
12  <parent>
13    <groupId>org.springframework.boot</groupId>
14    <artifactId>spring-boot-starter-parent</artifactId>
15    <version>2.3.3.RELEASE</version>
16    <relativePath/> <!-- lookup parent from repository --&gt;
17  &lt;/parent&gt;
18  &lt;groupId&gt;com.xiaoage&lt;/groupId&gt;
19  &lt;artifactId&gt;xinguang-parent&lt;/artifactId&gt;
20  &lt;version&gt;0.0.1-SNAPSHOT&lt;/version&gt;
21  &lt;name&gt;xinguang-parent&lt;/name&gt;
22  &lt;description&gt;Demo project for Spring Boot&lt;/description&gt;
23
24  &lt;properties&gt;
25    &lt;java.version&gt;1.8&lt;/java.version&gt;
26    &lt!--mybatis-plus--&gt;
27    &lt;mp.version&gt;3.4.0&lt;/mp.version&gt;
28    &lt!--swagger--&gt;
29    &lt;swagger.version&gt;2.7.0&lt;/swagger.version&gt;
30    &lt!--velocity模板引擎--&gt;
31    &lt;velocity.version&gt;2.2&lt;/velocity.version&gt;
32    &lt!--java连接mysql--&gt;
33    &lt;mysql-java.sersion&gt;8.0.13&lt;/mysql-java.sersion&gt;
34    &lt!--knife4j的版本--&gt;
35    &lt;knife4j.version&gt;2.0.4&lt;/knife4j.version&gt;
36  &lt;/properties&gt;
37
38  &lt;dependencyManagement&gt;
39    &lt;dependencies&gt;
40      &lt;dependency&gt;
41        &lt;groupId&gt;com.baomidou&lt;/groupId&gt;
42        &lt;artifactId&gt;mybatis-plus-boot-starter&lt;/artifactId&gt;
43        &lt;version&gt;${mp.version}&lt;/version&gt;
44      &lt;/dependency&gt;
45      &lt;dependency&gt;
46        &lt;groupId&gt;org.apache.velocity&lt;/groupId&gt;
47        &lt;artifactId&gt;velocity-engine-core&lt;/artifactId&gt;
48        &lt;version&gt;${velocity.version}&lt;/version&gt;
49      &lt;/dependency&gt;
50      &lt;dependency&gt;
51        &lt;groupId&gt;io.springfox&lt;/groupId&gt;</pre>
```

```
52     <artifactId>springfox-swagger-ui</artifactId>
53     <version>${swagger.version}</version>
54   </dependency>
55
56   <dependency>
57     <groupId>io.springfox</groupId>
58     <artifactId>springfox-swagger2</artifactId>
59     <version>${swagger.version}</version>
60   </dependency>
61
62   <dependency>
63     <groupId>mysql</groupId>
64     <artifactId>mysql-connector-java</artifactId>
65     <version>${mysql.java.sersion}</version>
66   </dependency>
67
68   <!--mybatis-plus自动代码生成-->
69   <dependency>
70     <groupId>com.baomidou</groupId>
71     <artifactId>mybatis-plus-generator</artifactId>
72     <version>${mp.version}</version>
73   </dependency>
74
75   <dependency>
76     <groupId>com.github.xiaoymin</groupId>
77     <artifactId>knife4j-spring-boot-starter</artifactId>
78     <!--在引用时请在maven中央仓库搜索最新版本号-->
79     <version>${knife4j.version}</version>
80   </dependency>
81 </dependencies>
82 </dependencyManagement>
83
84 <build>
85   <plugins>
86     <plugin>
87       <groupId>org.springframework.boot</groupId>
88       <artifactId>spring-boot-maven-plugin</artifactId>
89     </plugin>
90   </plugins>
91 </build>
92 </project>
```

*xinguang-base-common*



## pom.xml

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5   http://maven.apache.org/xsd/maven-4.0.0.xsd">
6   <parent>
7     <artifactId>xinguang-parent</artifactId>
8     <groupId>com.xiaooge</groupId>
9     <version>0.0.1-SNAPSHOT</version>
10    </parent>
11    <modelVersion>4.0.0</modelVersion>
12    <artifactId>xinguang-base-common</artifactId>
13
14    <dependencies>
15      <!--controller层-->
16      <dependency>
17        <groupId>org.springframework.boot</groupId>
18        <artifactId>spring-boot-starter-web</artifactId>
19      </dependency>
20
21      <!--数据访问层-->
22      <dependency>
23        <groupId>com.baomidou</groupId>
24        <artifactId>mybatis-plus-boot-starter</artifactId>
25      </dependency>
26
27      <!--mybatis自动代码生成-->
28      <dependency>
29        <groupId>com.baomidou</groupId>
30        <artifactId>mybatis-plus-generator</artifactId>

```

```

31     </dependency>
32
33     <!--代码生成的模板引擎-->
34     <dependency>
35         <groupId>org.apache.velocity</groupId>
36         <artifactId>velocity-engine-core</artifactId>
37     </dependency>
38
39     <!--java连接mysql-->
40     <dependency>
41         <groupId>mysql</groupId>
42         <artifactId>mysql-connector-java</artifactId>
43     </dependency>
44
45     <!--swagger-->
46     <dependency>
47         <groupId>io.springfox</groupId>
48         <artifactId>springfox-swagger2</artifactId>
49     </dependency>
50
51     <dependency>
52         <groupId>io.springfox</groupId>
53         <artifactId>springfox-swagger-ui</artifactId>
54     </dependency>
55
56     <!--lombok简化getter和setter-->
57     <dependency>
58         <groupId>org.projectlombok</groupId>
59         <artifactId>lombok</artifactId>
60     </dependency>
61
62     <dependency>
63         <groupId>com.github.xiaoymin</groupId>
64         <artifactId>knife4j-spring-boot-starter</artifactId>
65     </dependency>
66
67     </dependencies>
68 </project>

```

## xinguan-base-web

这个模块是跟前端交互的模块,创建方式和xinguan-base-common是一样的,也是子模块

### pom.xml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4           xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5           http://maven.apache.org/xsd/maven-4.0.0.xsd">

```

```

5   <parent>
6     <artifactId>xinguang-parent</artifactId>
7     <groupId>com.xiaooge</groupId>
8     <version>0.0.1-SNAPSHOT</version>
9   </parent>
10  <modelVersion>4.0.0</modelVersion>
11
12  <artifactId>xinguang-base-web</artifactId>
13
14  <dependencies>
15    <dependency>
16      <groupId>com.xiaooge</groupId>
17      <artifactId>xinguang-base-common</artifactId>
18      <version>0.0.1-SNAPSHOT</version>
19    </dependency>
20  </dependencies>
21 </project>

```

## 主配置文件application.yml

```

1 spring:
2   application:
3     name: xinguang-base-web
4
5   datasource:
6     driver-class-name: com.mysql.cj.jdbc.Driver
7     url: jdbc:mysql://localhost:3306/xinguang?useUnicode=true&characterEncoding=UTF-
8&serverTimezone=GMT%2B8
8     username: root
9     password: root
10  jackson:
11    date-format: yyyy-MM-dd HH:mm:ss
12    time-zone: GMT+8
13  server:
14    port: 8081
15
16  mybatis-plus:
17    configuration:
18      log-impl: org.apache.ibatis.logging.stdout.StdoutImpl
19    global-config:
20      db-config:
21        logic-delete-field: deleted # 全局逻辑删除的实体字段名(since 3.3.0,配置后可以忽略不
配置步骤2)
22        logic-delete-value: 1 # 逻辑已删除值(默认为 1)
23        logic-not-delete-value: 0 # 逻辑未删除值(默认为 0)
24    mapper-locations: classpath*:mapper/*.xml

```

## 主启动类XinGuanApplication

```

1 package com.xiaooge;
2
3 import org.mybatis.spring.annotation.MapperScan;

```

```

4 import org.springframework.boot.SpringApplication;
5 import org.springframework.boot.autoconfigure.SpringBootApplication;
6 import springfox.documentation.swagger2.annotations.EnableSwagger2;
7
8 @SpringBootApplication
9 @MapperScan("com.xiaoge.system.mapper")
10 //开启SwaggerUI
11 @EnableSwagger2
12 public class XinGuanApplication {
13     public static void main(String[] args) {
14         SpringApplication.run(XinGuanApplication.class,args);
15     }
16 }
```

代码自动生成工具类**CodeGenerator**

```

1 package com.xiaoge.generator;
2
3 import com.baomidou.mybatisplus.annotation.DbType;
4 import com.baomidou.mybatisplus.annotation.IdType;
5 import com.baomidou.mybatisplus.core.exceptions.MybatisPlusException;
6 import com.baomidou.mybatisplus.generator.AutoGenerator;
7 import com.baomidou.mybatisplus.generator.config.*;
8 import com.baomidou.mybatisplus.generator.config.rules.DateType;
9 import com.baomidou.mybatisplus.generator.config.rules.NamingStrategy;
10 import org.apache.commons.lang3.StringUtils;
11
12 import java.util.Scanner;
13
14 public class CodeGenerator {
15
16     /**
17      * <p>
18      * 读取控制台内容
19      * </p>
20      */
21     public static String scanner(String tip) {
22         Scanner scanner = new Scanner(System.in);
23         StringBuilder help = new StringBuilder();
24         help.append("请输入" + tip + ": ");
25         System.out.println(help.toString());
26         if (scanner.hasNext()) {
27             String ipt = scanner.next();
28             if (StringUtils.isNotEmpty(ipt)) {
29                 return ipt;
30             }
31         }
32         throw new MybatisPlusException("请输入正确的" + tip + "！");
33     }
34
35     public static void main(String[] args) {
36         // 创建代码生成器对象
37         AutoGenerator mpg = new AutoGenerator();
```

```
38 // 全局配置
39 GlobalConfig gc = new GlobalConfig();
40 gc.setOutputDir(scanner("请输入你的项目路径") + "/src/main/java");
41 gc.setAuthor("xiaoge");
42 //生成之后是否打开资源管理器
43 gc.setOpen(false);
44 //重新生成时是否覆盖文件
45 gc.setFileOverride(false);
46 // %s 为占位符
47 //mp生成service层代码，默认接口名称第一个字母是有I
48 gc.setServiceName("%sService");
49 //设置主键生成策略 自动增长
50 gc.setIdType(IdType.AUTO);
51 //设置Date的类型 只使用 java.util.date 代替
52 gc.setDateType(DateType.ONLY_DATE);
53 //开启实体属性 Swagger2 注解
54 gc.setSwagger2(true);
55 mpg.setGlobalConfig(gc);
56
57 // 数据源配置
58 DataSourceConfig dsc = new DataSourceConfig();
59 dsc.setUrl("jdbc:mysql://localhost:3306/xinguang?useUnicode=true&characterEncoding=UTF-8&serverTimezone=GMT%2B8");
60
61 dsc.setDriverName("com.mysql.cj.jdbc.Driver");
62 dsc.setUsername("root");
63 dsc.setPassword("root");
64 //使用mysql数据库
65 dsc.setDbType(DbType.MYSQL);
66 mpg.setDataSource(dsc);
67
68 // 包配置
69 PackageConfig pc = new PackageConfig();
70 pc.setModuleName(scanner("请输入模块名"));
71 pc.setParent("com.xiaoge");
72 pc.setController("controller");
73 pc.setService("service");
74 pc.setServiceImpl("service.impl");
75 pc.setMapper("mapper");
76 pc.setEntity("entity");
77 pc.setXml("mapper");
78 mpg.setPackageInfo(pc);
79
80 // 策略配置
81 StrategyConfig strategy = new StrategyConfig();
82 //设置哪些表需要自动生成
83 strategy.setInclude(scanner("表名，多个英文逗号分割").split(","));
84
85 //实体类名称驼峰命名
86 strategy.setNaming(NamingStrategy.underline_to_camel);
87 //列名名称驼峰命名
88 strategy.setColumnNaming(NamingStrategy.underline_to_camel);
89 //使用简化getter和setter
```

```
90     strategy.setEntityLombokModel(true);
91     //设置controller的api风格 使用RestController
92     strategy.setRestControllerStyle(true);
93     //驼峰转连字符
94     strategy.setControllerMappingHyphenStyle(true);
95     mpg.setStrategy(strategy);
96     mpg.execute();
97 }
98 }
```

## 引入Swagger

使用代码生成器生成一些测试代码,改写TbUserController类如下:

```
1 package com.xiaoge.system.controller;
2
3 import com.xiaoge.handler.BusinessException;
4 import com.xiaoge.response.Result;
5 import com.xiaoge.response.ResultCode;
6 import com.xiaoge.system.entity.TbUser;
7 import com.xiaoge.system.service.TbUserService;
8 import io.swagger.annotations.Api;
9 import io.swagger.annotations.ApiOperation;
10 import org.springframework.beans.factory.annotation.Autowired;
11 import org.springframework.web.bind.annotation.GetMapping;
12 import org.springframework.web.bind.annotation.PathVariable;
13 import org.springframework.web.bind.annotation.RequestMapping;
14
15 import org.springframework.web.bind.annotation.RestController;
16
17 import java.util.List;
18
19 /**
20 * <p>
21 * 用户表 前端控制器
22 * </p>
23 *
24 * @author xiaoge
25 * @since 2020-08-28
26 */
27 @RestController
28 @RequestMapping("/system/tb-user")
29 @Api(value = "系统用户模块",tags = "系统用户接口")
30 public class TbUserController {
31
32     @Autowired
33     private TbUserService tbuserService;
34
35     @GetMapping
36     @ApiOperation(value = "用户列表",notes = "查询所有用户信息")
```

```

37     public List<TbUser> findUsers(){
38         List<TbUser> list = tbUserService.list();
39         return list;
40     }
41 }

```

## API详细说明

注释汇总

| 作用范围      | API                | 使用位置                      |
|-----------|--------------------|---------------------------|
| 对象属性      | @ApiModelProperty  | 用在出入参数对象的字段上              |
| 协议集描述     | @Api               | 用于controller类上            |
| 协议描述      | @ApiOperation      | 用在controller的方法上          |
| Response集 | @ApiResponses      | 用在controller的方法上          |
| Response  | @ApiResponse       | 用在 @ApiResponses里边        |
| 非对象参数集    | @ApiImplicitParams | 用在controller的方法上          |
| 非对象参数描述   | @ApiImplicitParam  | 用在@ApiImplicitParams的方法里边 |
| 描述返回对象的意义 | @ApiModel          | 用在返回对象类上                  |

详细介绍参考:<https://www.jianshu.com/p/12f4394462d5>

启动项目进行测试，浏览器输入:<http://localhost:8081/swagger-ui.html>

展示如下：

或许这个文档有那么点草率,我们来美化一些

# knife4j美化Swagger

[knife4j官方地址](#)

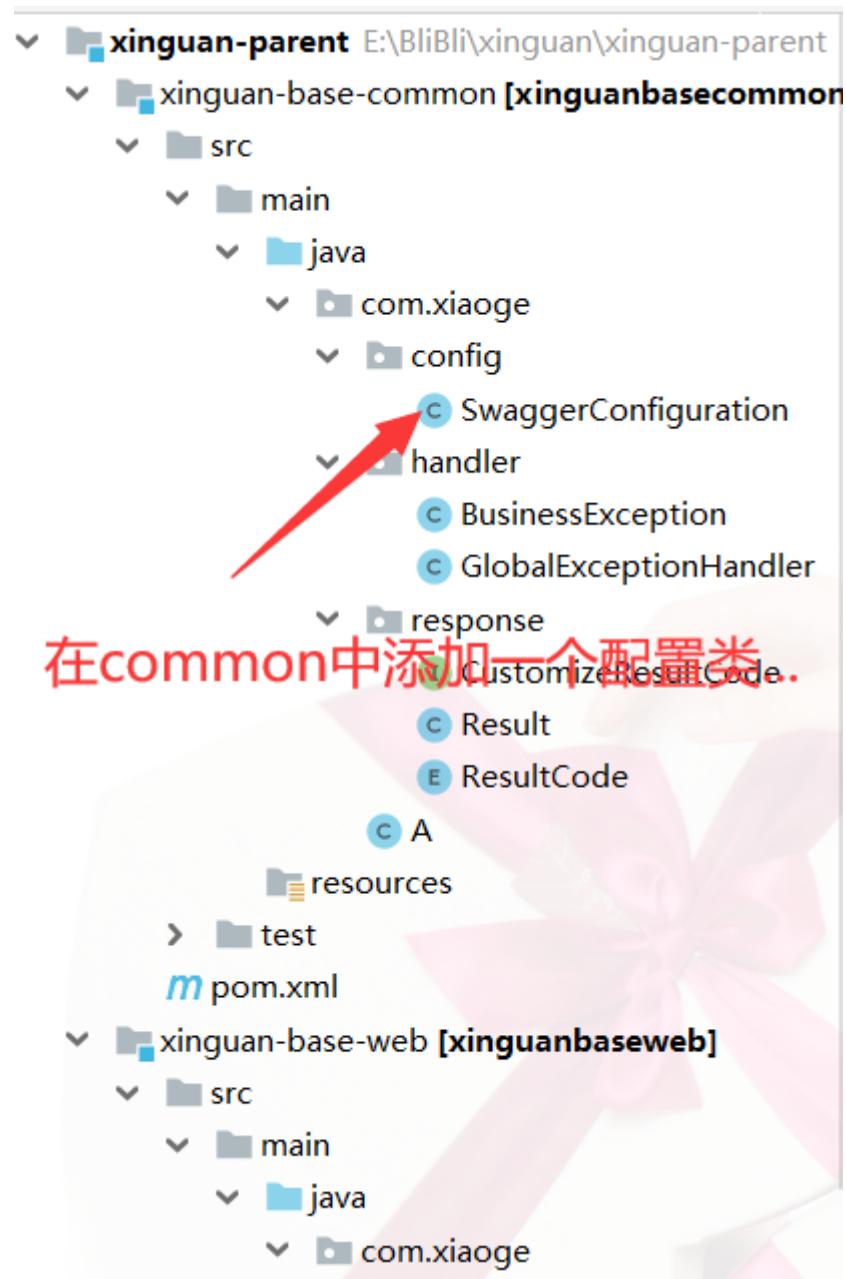
The screenshot shows the official website for knife4j. At the top is a navigation bar with links for '指南' (Guide), 'knife4j', 'Knife4jCloud', '解决方案' (Solution), 'Gitter', '更新日志' (Update Log), 'FAQ', '源码分析' (Code Analysis), '代码托管' (Code Hosting), '赞助' (Sponsor), and '选择语言' (Select Language). Below the navigation is a large green button labeled '快速开始 →'. To the left of this button is a section titled '简洁' (Simple) with a brief description. To the right are sections for '个性化配置' (Personalized Configuration) and '增强' (Enhanced), each with their own descriptions.

点击快速开始,可以看到

The screenshot shows the 'Quick Start' page. On the left is a sidebar with a tree-like menu under the '开始' (Start) category. A red arrow points from the text '这里可以快速入门' (Here you can start quickly) to the '快速开始' (Quick Start) item in the sidebar. The main content area has a title '快速开始' (Quick Start) and a sub-section 'Java开发' (Java Development). It contains instructions for Java developers and code snippets for Maven dependencies. A red arrow also points from the text '主要添加了一个坐标' (Mainly added one coordinate) to the dependency code in the sidebar.

主要添加了一个坐标

```
1 <dependency>
2   <groupId>com.github.xiaoymin</groupId>
3   <artifactId>knife4j-spring-boot-starter</artifactId>
4   <!--在引用时请在maven中央仓库搜索最新版本号-->
5   <version>${knife4j.version}</version>
6 </dependency>
```



### Swagger配置类SwaggerConfiguration

```
1 package com.xiaoge.config;  
2  
3 import net.sf.jsqlparser.expression.operators.arithmetic.Concat;  
4 import org.springframework.context.annotation.Bean;  
5 import org.springframework.context.annotation.Configuration;  
6 import springfox.documentation.builders.ApiInfoBuilder;  
7 import springfox.documentation.builders.PathSelectors;  
8 import springfox.documentation.builders.RequestHandlerSelectors;  
9 import springfox.documentation.service.ApiInfo;  
10 import springfox.documentation.service.Contact;  
11 import springfox.documentation.spi.DocumentationType;  
12 import springfox.documentation.spring.web.plugins.Docket;  
13 import springfox.documentation.swagger2.annotations.EnableSwagger2;  
14  
15 /**
```

```

16 * @author NieChangan
17 */
18 @Configuration
19 @EnableSwagger2
20 public class SwaggerConfiguration {
21
22     @Bean
23     public Docket createRestApi() {
24         return new Docket(DocumentationType.SWAGGER_2)
25             .apiInfo(apiInfo())
26             .select()
27             //这里一定要标注你控制器的位置
28
29         .apis(RequestHandlerSelectors.basePackage("com.xiaoge.system.controller"))
30             .paths(PathSelectors.any())
31             .build();
32     }
33
34     private ApiInfo apiInfo() {
35         return new ApiInfoBuilder()
36             .title("新冠物资管理系统API文档")
37             .description("新冠物资管理系统API文档")
38             .termsOfServiceUrl("https://angegit.gitee.io/myblog/")
39             .contact(new
40             Contact("xiaoge","https://angegit.gitee.io/myblog/","1351261434@qq.com"))
41             .version("1.0")
42             .build();
43     }
44 }
```

再次重启项目访问: <http://localhost:8081/doc.html>

The screenshot shows the Knife4j API documentation interface. On the left, there's a sidebar with navigation links: '主页', 'Swagger Models', '文档管理', '系统用户接口', '用户列表' (which is highlighted in blue), and '查询单个用户'. The main content area has a title '新冠物资管理系统API文档'. Below it, there's a navigation bar with tabs: '主页', '用户列表 X', '查询单个用户 X', '文档' (which is also highlighted in blue), and '调试'. The '用户列表' section is currently active. It shows a 'GET' method for the endpoint '/system/tb-user'. The '请求数据类型' (Request Data Type) is set to 'application/x-www-form-urlencoded', and the '响应数据类型' (Response Data Type) is '[ \*/\*' ]. The '接口描述' (Interface Description) section says '查询所有用户信息'. The '请求参数' (Request Parameters) section is empty, indicated by 'No Data'. The '响应状态' (Response Status) section is also empty. At the bottom, there's a footer with the text 'Apache License 2.0 | Copyright © 2019-Knife4j'.

程序员不仅要会写代码,还要追求美,例如我就喜欢美女...😊

The screenshot shows the Knife4j API documentation interface. On the left, there's a sidebar with a dark theme containing links like '主页', 'Swagger Models', '文档管理', '系统用户接口', '用户列表' (which is highlighted in blue), and '查询单个用户'. The main content area has a light background and displays the '新冠物资管理系统API文档'. It shows a '请求方式' (Request Method) section for a 'GET /system/tb-user' endpoint. The '调试' (Debug) button in the sidebar is highlighted with a red arrow, and the text '其实用法类似Postman' is written below it. Other buttons in the request method section include '请求头部' (Request Headers), '请求参数' (Request Parameters), and radio buttons for 'x-www-form-urlencoded', 'form-data', and 'raw'. At the bottom right of the main content area, there's a '发送' (Send) button and a note about Apache License 2.0.

## 统一返回类处理

用户返回User列表

部门返回Dept列表

...

如果不对返回的结果进行统一的格式返回,那么前端将如何对后台返回的结果进行解析呢?

那接下来我们封装自己的一个统一结果处理类

**接口CustomizeResultCode**

```
1 package com.xiaoge.response;
2
3 /**
4 * @author NieChangan
5 */
6 public interface CustomizeResultCode {
7
8     /**
9      * 获取错误状态码
10     * @return 错误状态码
11     */
12     Integer getCode();
13
14     /**
15      * 获取错误信息
16     * @return 错误信息
17     */
18     String getMessage();
19 }
```

## 枚举类返回结果ResultCode

```
1 package com.xiaoge.response;
2
3 /**
4  * @author NieChangan
5 */
6 public enum ResultCode implements CustomizeresultCode{
7     /**
8      * 20000:"成功"
9      */
10    SUCCESS(20000, "成功"),
11    /**
12     * 20001:"失败"
13     */
14    ERROR(20001, "失败"),
15    /**
16     * 3005:"密码不正确!"
17     */
18    PASS_NOT_CORRECT(3005, "密码不正确!请重新尝试!"),
19
20    /**
21     * 3006:"算数异常"
22     */
23    ARITHMETIC_EXCEPTION(3006, "算数异常"),
24
25    /**
26     * 3007:"用户不存在"
27     */
28    USER_NOT_FOUND_EXCEPTION(3007, "用户不存在"),
29    /**
30     * 3006:"尚未登录!"
31     */
32    NOT_LOGIN(3006, "尚未登录!"),
33    /**
34     * 2005:"没有找到这一条历史信息!有人侵入数据库强制删除了!"
35     */
36    INTRODUCTION_NOT_FOUND(2005, "没有找到这一条历史信息!有人侵入数据库强制删除了!"),
37    /**
38     * 404:没有找到对应的请求路径
39     */
40    PAGE_NOT_FOUND(404, "你要请求的页面好像暂时飘走了...要不试试请求其它页面?"),
41    /**
42     * 500:服务端异常
43     */
44    INTERNAL_SERVER_ERROR(500, "服务器冒烟了...要不等它降降温再来访问?"),
45    /**
46     * 2001:未知异常
47     */
48    UNKNOW_SERVER_ERROR(2001, "未知异常,请联系管理员!");
49
50    private Integer code;
51}
```

```
52     private String message;
53
54     ResultCode(Integer code, String message){
55         this.code = code;
56         this.message = message;
57     }
58
59     @Override
60     public Integer getCode() {
61         return code;
62     }
63
64     @Override
65     public String getMessage() {
66         return message;
67     }
68 }
```

## 公共返回结果Result

```
1 package com.xiaoge.response;
2
3 import io.swagger.annotations.ApiModelProperty;
4 import io.swagger.models.auth.In;
5 import lombok.Data;
6
7 import java.util.HashMap;
8 import java.util.Map;
9
10 /**
11 * 公共返回结果
12 * @author NieChangan
13 */
14 @Data
15 public class Result {
16
17     @ApiModelProperty(value = "是否成功")
18     private Boolean success;
19
20     @ApiModelProperty(value = "返回码")
21     private Integer code;
22
23     @ApiModelProperty(value = "返回消息")
24     private String message;
25
26     @ApiModelProperty(value = "返回数据")
27     private Map<String, Object> data = new HashMap<>();
28
29 /**
30 * 构造方法私有化，里面的方法都是静态方法
31 * 达到保护属性的作用
32 */
33     private Result(){
```

```
34     }
35
36     /**
37      * 这里是使用链式编程
38      */
39
40     public static Result ok(){
41         Result result = new Result();
42         result.setSuccess(true);
43         result.setCode(ResultCode.SUCCESS.getCode());
44         result.setMessage(ResultCode.SUCCESS.getMessage());
45         return result;
46     }
47
48     public static Result error(){
49         Result result = new Result();
50         result.setSuccess(false);
51         result.setCode(ResultCode.ERROR.getCode());
52         result.setMessage(ResultCode.ERROR.getMessage());
53         return result;
54     }
55
56     /**
57      * 自定义返回成功与否
58      * @param success
59      * @return
60      */
61     public Result success(Boolean success){
62         this.setSuccess(success);
63         return this;
64     }
65
66     public Result message(String message){
67         this.setMessage(message);
68         return this;
69     }
70
71     public Result code(Integer code){
72         this.setCode(code);
73         return this;
74     }
75
76     public Result data(String key, Object value){
77         this.data.put(key,value);
78         return this;
79     }
80
81     public Result data(Map<String, Object> map){
82         this.setData(map);
83         return this;
84     }
85 }
```

修改**TbUserController**如下：

```

50     throw new
51     BusinessException(ResultCode.USER_NOT_FOUND_EXCEPTION.getCode(),ResultCode.USER_NOT
52         _FOUND_EXCEPTION.getMessage());
53 }

```

在Swagger中进行测试

这样的数据就是统一的,不管什么类型的数据返回都是这样的

## 统一异常处理

```

@RestController
@RequestMapping("/system/tb-user")
@Api(value = "系统用户模块",tags = "系统用户接口")
public class TbUserController {
    @Autowired
    private TbUserService tbUserService;

    @GetMapping
    @ApiOperation(value = "用户列表",notes = "查询所有用户信息")
    public Result findUsers(){
        int i = 1/0; //添加这样一句测试下,会发现报错了,其实那样并不是用户想看到的结果,肯定我们需要对未知的异常进行处理
        List<TbUser> list = tbUserService.list();
        return Result.ok().data("users",list);
    }

    @GetMapping("/{id}")
    @ApiOperation(value = "查询单个用户",notes = "通过ID查询对应的用户信息")
    public Result getUserById(@PathVariable("id") Long id){
        TbUser user = tbUserService.getById(id);
        if(user!=null){
            return Result.ok().data("user",user);
        }else{
            throw new BusinessException(ResultCode.USER_NOT_FOUND_EXCEPTION.getCode(),ResultCode.USER_NOT_FOUND_EXCEPTION.getMessage());
        }
    }
}

```

定义BusinessException类

```

1 package com.xiaoge.handler;
2

```

```
3 import io.swagger.annotations.ApiModelProperty;
4 import lombok.AllArgsConstructor;
5 import lombok.Data;
6 import lombok.NoArgsConstructor;
7
8 /**
9  * @author NieChangan
10 */
11 @Data
12@AllArgsConstructor
13@NoArgsConstructor
14 public class BusinessException extends RuntimeException{
15
16     @ApiModelProperty(value = "状态码")
17     private Integer code;
18
19     @ApiModelProperty(value = "错误信息")
20     private String errMsg;
21 }
```

## 定义GlobalExceptionHandler类

```
1 package com.xiaoge.handler;
2
3 import com.xiaoge.response.Result;
4 import com.xiaoge.response.ResultCode;
5 import lombok.extern.slf4j.Slf4j;
6 import org.springframework.web.bind.annotation.ControllerAdvice;
7 import org.springframework.web.bind.annotation.ExceptionHandler;
8 import org.springframework.web.bind.annotation.RequestMapping;
9 import org.springframework.web.bind.annotation.ResponseBody;
10
11 /**
12  * @author NieChangan
13  */
14 @ControllerAdvice
15 @Slf4j
16 public class GlobalExceptionHandler {
17
18     /**
19      * 全局异常处理,没有指定异常的类型,不管什么异常都是可以捕获的
20      * @param e
21      * @return
22      */
23     @ExceptionHandler(Exception.class)
24     @ResponseBody
25     public Result error(Exception e){
26         //e.printStackTrace();
27         log.error(e.getMessage());
28         return Result.error();
29     }
30
31     /**
```

```

32     * 处理特定异常类型,可以定义多个,这里只以ArithmeticException为例
33     * @param e
34     * @return
35     */
36     @ExceptionHandler(ArithmeticException.class)
37     @ResponseBody
38     public Result error(ArithmeticException e){
39         //e.printStackTrace();
40         log.error(e.getMessage());
41         return Result.error().code(ResultCode.ARITHMETIC_EXCEPTION.getCode())
42             .message(ResultCode.ARITHMETIC_EXCEPTION.getMessage());
43     }
44
45     /**
46      * 处理业务异常,我们自己定义的异常
47      * @param e
48      * @return
49      */
50     @ExceptionHandler(BusinessException.class)
51     @ResponseBody
52     public Result error(BusinessException e){
53         //e.printStackTrace();
54         log.error(e.getErrMsg());
55         return Result.error().code(e.getCode())
56             .message(e.getErrMsg());
57     }
58 }
```

当我们的程序出现异常的时候也能正确的处理

The screenshot shows the Knife4j API documentation interface. On the left, there's a sidebar with navigation links: 'default', '主页' (Home), 'Swagger Models', '文档管理' (Document Management), '系统用户接口' (System User Interface), and '用户列表' (User List) which is currently selected. The main area is titled '新冠物资管理系统API文档' (COVID-19 Material Management System API Documentation). It shows a search bar and several filter buttons: '主页' (Home), '用户列表 X', '查询单个用户 X', '全局参数设置(default) X', '离线文档(default) X', and '个性化设置 X'. Below these is a 'GET /system/tb-user' endpoint. The '请求头部' (Request Headers) section has a radio button selected for 'x-www-form-urlencoded'. The '请求参数' (Request Parameters) section is empty. Under the '响应内容' (Response Content) tab, the JSON response is displayed:

```

1  {
2     "success": false,
3     "code": 3006,
4     "message": "查教异常",
5     "data": {}
6 }
```

A red annotation '特定异常处理' (Specific Exception Handling) is placed over the 'message' field in the JSON response. To the right of the response, there are four small buttons with Chinese labels: '是否成功' (Is Success), '返回码' (Return Code), '返回消息' (Return Message), and '返回数据' (Return Data). At the bottom right of the main area, there are buttons for '显示说明' (Show Description), '响应码: 200 耗时: 253ms 大小: 60 b' (Response Code: 200, Duration: 253ms, Size: 60 b), and '发送' (Send). The footer of the page says 'Apache License 2.0 | Copyright © 2019-Knife4j'.

The screenshot shows the Swagger UI interface for a COVID-19 material management system API. On the left, there's a sidebar with navigation links like '主页', 'Swagger Models', '文档管理', '系统用户接口', '用户列表', and '查询单个用户'. The main area has tabs for '主页', '用户列表', '查询单个用户', '全局参数设置(default)', '离线文档(default)', and '个性化设置'. A search bar at the top right contains the text '输入文档关键字搜索'. Below the tabs, there's a 'GET /system/tb-user/{id}' section. Under '请求参数', the 'x-www-form-urlencoded' option is selected, with '参数名称' (name) set to 'id' and '参数值' (value) set to '10000'. There are tabs for '响应内容' (Response Content), 'Raw', 'Headers', and 'Curl'. The 'Raw' tab shows a JSON response: { "success": false, "code": 3007, "message": "用户不存在", "data": {} }. To the right of the response, there are four small labels: '是否成功' (Success), '返回码' (Return Code), '返回消息' (Return Message), and '返回数据' (Return Data). At the bottom right of the main area, there are buttons for '显示说明' (Show Description), '响应码: 200 耗时: 1.0s 大小: 62 b' (Response Code: 200, Duration: 1.0s, Size: 62 b), and '删除' (Delete). The footer of the page says 'Apache License 2.0 | Copyright © 2019-Knife4j'.

## 统一日志处理

这里的话,本来是想用AOP进行异常收集的,但是录视频和备课的时候没考虑到,暂时先用简单的方式实现

### 日志级别

日志记录器 (Logger) 的行为是分等级的。如下表所示：

分为：OFF、FATAL、ERROR、WARN、INFO、DEBUG、ALL

默认情况下，spring boot从控制台打印出来的日志级别只有INFO及以上级别，可以配置日志级别

```
1 # 设置日志级别
2 Logging.level.root=WARN
```

这种方式只能将日志打印在控制台上

### Logback日志

spring boot内部使用Logback作为日志实现的框架。

Logback和log4j非常相似，如果你对log4j很熟悉，那对logback很快就会得心应手。

logback相对于log4j的一些优点：[https://blog.csdn.net/caisini\\_vc/article/details/48551287](https://blog.csdn.net/caisini_vc/article/details/48551287)

### 配置logback日志

删除application.properties中的日志配置

安装idea彩色日志插件：grep-console

resources 中创建 [logback-spring.xml](#)

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2
3 <!-- 级别从高到低 OFF 、 FATAL 、 ERROR 、 WARN 、 INFO 、 DEBUG 、 TRACE 、 ALL -->
```

```
4  <!-- 日志输出规则 根据当前ROOT 级别，日志输出时，级别高于root默认的级别时 会输出 -->
5  <!-- 以下 每个配置的 filter 是过滤掉输出文件里面，会出现高级别文件，依然出现低级别的日志信息，通过filter 过滤只记录本级别的日志 -->
6  <!-- scan 当此属性设置为true时，配置文件如果发生改变，将会被重新加载，默认值为true。 -->
7  <!-- scanPeriod 设置监测配置文件是否有修改的时间间隔，如果没有给出时间单位，默认单位是毫秒。当
8  scan为true时，此属性生效。默认的时间间隔为1分钟。 -->
9  <!-- debug 当此属性设置为true时，将打印出logback内部日志信息，实时查看logback运行状态。默认值
10 为false。 -->
11  <configuration scan="true" scanPeriod="60 seconds" debug="false">
12
13
14  <!--
15  ****
16  <!--自定义项 开始-->
17  <!--
18  ****
19  <!-- 定义日志文件 输出位置 -->
20  <property name="log.home_dir" value="E:/Blibli/xinguan/logs"/>
21  <property name="log.app_name" value="http-demo"/>
22  <!-- 日志最大的历史 30天 -->
23  <property name="log.maxHistory" value="30"/>
24  <!-- 日志界别 -->
25  <property name="log.level" value="info"/>
26  <!-- 打印sql语句 需要指定dao/mapper层包的位置 -->
27  <property name="mapper.package" value="com.xiaoge.system.mapper"/>
28
29  <!-- 彩色日志 -->
30  <!-- 配置格式变量：CONSOLE_LOG_PATTERN 彩色日志格式 -->
31  <!-- magenta:洋红 -->
32  <!-- boldMagenta:粗红-->
33  <!-- cyan:青色 -->
34  <!-- white:白色 -->
35  <!-- magenta:洋红 -->
36  <property name="CONSOLE_LOG_PATTERN"
37      value="%yellow(%date{yyyy-MM-dd HH:mm:ss}) |%highlight(%-5level)
|%blue(%thread) |%blue(%file:%line) |%green(%logger) |%cyan(%msg%n)"/>
38
39  <!--
40  ****
41  <!--自定义项 结束-->
42  <!--
43  ****
44  <!-- ConsoleAppender 控制台输出日志 -->
45  <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
46      <encoder>
47          <pattern>
48              <!-- 设置日志输出格式 -->
49                  ${CONSOLE_LOG_PATTERN}
```

```
49         </pattern>
50     </encoder>
51 </appender>
52
53     <!-- ERROR级别日志 -->
54     <!-- 滚动记录文件，先将日志记录到指定文件，当符合某个条件时，将日志记录到其他文件
55 RollingFileAppender -->
56     <appender name="ERROR"
57         class="ch.qos.logback.core.rolling.RollingFileAppender">
58         <!-- 过滤器，只记录WARN级别的日志 -->
59         <!-- 果日志级别等于配置级别，过滤器会根据onMatch 和 onMismatch接收或拒绝日志。 -->
60         <filter class="ch.qos.logback.classic.filter.LevelFilter">
61             <!-- 设置过滤级别 -->
62             <level>ERROR</level>
63             <!-- 用于配置符合过滤条件的操作 -->
64             <onMatch>ACCEPT</onMatch>
65             <!-- 用于配置不符合过滤条件的操作 -->
66             <onMismatch>DENY</onMismatch>
67         </filter>
68         <!-- 最常用的滚动策略，它根据时间来制定滚动策略.既负责滚动也负责触发滚动 -->
69         <rollingPolicy
70             class="ch.qos.logback.core.rolling.sizeAndTimeBasedRollingPolicy">
71             <!--日志输出位置 可相对、和绝对路径 -->
72             <fileNamePattern>
73                 ${log.home_dir}/error/%d{yyyy-MM-dd}/${log.app_name}-%i.log
74             </fileNamePattern>
75             <!-- 可选节点，控制保留的归档文件的最大数量，超出数量就删除旧文件，假设设置每个月滚
76             动，且<maxHistory>是6，  

77             则只保存最近6个月的文件，删除之前的旧文件。注意，删除旧文件是，那些为了归档而创建的
78             目录也会被删除 -->
79             <maxHistory>${log.maxHistory}</maxHistory>
80             <!--日志文件最大的大小-->
81             <MaxFileSize>${log.maxSize}</MaxFileSize>
82         </rollingPolicy>
83         <encoder>
84             <pattern>
85                 <!-- 设置日志输出格式 -->
86                 %msg% %date% [%thread] %-5level %logger{50} -
87             </pattern>
88         </encoder>
89     </appender>
90
91     <!-- WARN级别日志 appender -->
92     <appender name="WARN" class="ch.qos.logback.core.rolling.RollingFileAppender">
93         <!-- 过滤器，只记录WARN级别的日志 -->
94         <!-- 果日志级别等于配置级别，过滤器会根据onMatch 和 onMismatch接收或拒绝日志。 -->
95         <filter class="ch.qos.logback.classic.filter.LevelFilter">
96             <!-- 设置过滤级别 -->
97             <level>WARN</level>
98             <!-- 用于配置符合过滤条件的操作 -->
99             <onMatch>ACCEPT</onMatch>
```

```
96         <!-- 用于配置不符合过滤条件的操作 -->
97         <onMismatch>DENY</onMismatch>
98     </filter>
99     <rollingPolicy
100    class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
101        <!--日志输出位置 可相对、和绝对路径 -->
102        <fileNamePattern>${log.home_dir}/warn/%d{yyyy-MM-dd}/${log.app_name}-
103          %i.log</fileNamePattern>
104        <maxHistory>${log.maxHistory}</maxHistory>
105        <!--当天的日志大小 超过MaxFileSize时,压缩日志并保存-->
106        <MaxFileSize>${log.maxSize}</MaxFileSize>
107    </rollingPolicy>
108    <encoder>
109        <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} -
110          %msg%n</pattern>
111    </encoder>
112  </appender>
113
114
115  <!-- INFO级别日志 appender -->
116  <appender name="INFO" class="ch.qos.logback.core.rolling.RollingFileAppender">
117      <filter class="ch.qos.logback.classic.filter.LevelFilter">
118          <level>INFO</level>
119          <onMatch>ACCEPT</onMatch>
120          <onMismatch>DENY</onMismatch>
121      </filter>
122      <rollingPolicy
123        class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
124            <fileNamePattern>${log.home_dir}/info/%d{yyyy-MM-dd}/${log.app_name}-
125              %i.log</fileNamePattern>
126            <maxHistory>${log.maxHistory}</maxHistory>
127            <MaxFileSize>${log.maxSize}</MaxFileSize>
128        </rollingPolicy>
129        <encoder>
130            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%-5level] %logger -
131              %msg%n</pattern>
132        </encoder>
133    </appender>
134
135
136  <!-- DEBUG级别日志 appender -->
137  <appender name="DEBUG"
138    class="ch.qos.logback.core.rolling.RollingFileAppender">
139      <filter class="ch.qos.logback.classic.filter.LevelFilter">
140          <level>DEBUG</level>
141          <onMatch>ACCEPT</onMatch>
142          <onMismatch>DENY</onMismatch>
143      </filter>
144      <rollingPolicy
145        class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
146            <fileNamePattern>${log.home_dir}/debug/%d{yyyy-MM-dd}/${log.app_name}-
147              %i.log</fileNamePattern>
148            <maxHistory>${log.maxHistory}</maxHistory>
```

```
140         <MaxFileSize>${log.maxsize}</MaxFileSize>
141     </rollingPolicy>
142     <encoder>
143         <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%-5level] %logger -
144         %msg%n</pattern>
145     </encoder>
146   </appender>
147
148   <!-- TRACE级别日志 appender -->
149   <appender name="TRACE"
150     class="ch.qos.logback.core.rolling.RollingFileAppender">
151     <filter class="ch.qos.logback.classic.filter.LevelFilter">
152       <level>TRACE</level>
153       <onMatch>ACCEPT</onMatch>
154       <onMismatch>DENY</onMismatch>
155     </filter>
156     <rollingPolicy
157       class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
158       <fileNamePattern>${log.home_dir}/trace/%d{yyyy-MM-dd}/${log.app_name}-
159       %i.log</fileNamePattern>
160       <maxHistory>${log.maxHistory}</maxHistory>
161       <MaxFileSize>${log.maxSize}</MaxFileSize>
162     </rollingPolicy>
163     <encoder>
164         <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%-5level] %logger -
165         %msg%n</pattern>
166     </encoder>
167   </appender>
168
169   <!--设置一个向上传递的appender,所有级别的日志都会输出-->
170   <appender name="app" class="ch.qos.logback.core.rolling.RollingFileAppender">
171     <rollingPolicy
172       class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
173       <fileNamePattern>${log.home_dir}/app/%d{yyyy-MM-dd}/${log.app_name}-
174       %i.log</fileNamePattern>
175       <maxHistory>${log.maxHistory}</maxHistory>
176       <MaxFileSize>${log.maxSize}</MaxFileSize>
177     </rollingPolicy>
178     <encoder>
179         <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%-5level] %logger -
180         %msg%n</pattern>
181     </encoder>
182   </appender>
183
184   <!--org.springframework.web包下的类的日志输出-->
185   <logger name="org.springframework.web" additivity="false" level="WARN">
186     <appender-ref ref="WARN"/>
187   </logger>
188   <!--dao层包下的类的日志输出-->
189   <logger name="${mapper.package}" additivity="false" level="DEBUG">
```

```

185     <appender-ref ref="app"/>
186     <appender-ref ref="ERROR"/>
187     <!--打印控制台-->
188     <appender-ref ref="CONSOLE"/>
189   </logger>
190
191
192     <!-- root级别    DEBUG -->
193   <root>
194     <!-- 打印debug级别日志及以上级别日志 -->
195     <level value="${log.level}" />
196     <!-- 控制台输出 -->
197     <appender-ref ref="CONSOLE"/>
198     <!-- 不管什么包下的日志都输出文件 -->
199     <appender-ref ref="ERROR"/>
200     <appender-ref ref="INFO"/>
201     <appender-ref ref="WARN"/>
202     <appender-ref ref="DEBUG"/>
203     <appender-ref ref="TRACE"/>
204   </root>
205
206 </configuration>

```

## 将错误日志输出到文件

### 修改GlobalExceptionHandler类

```

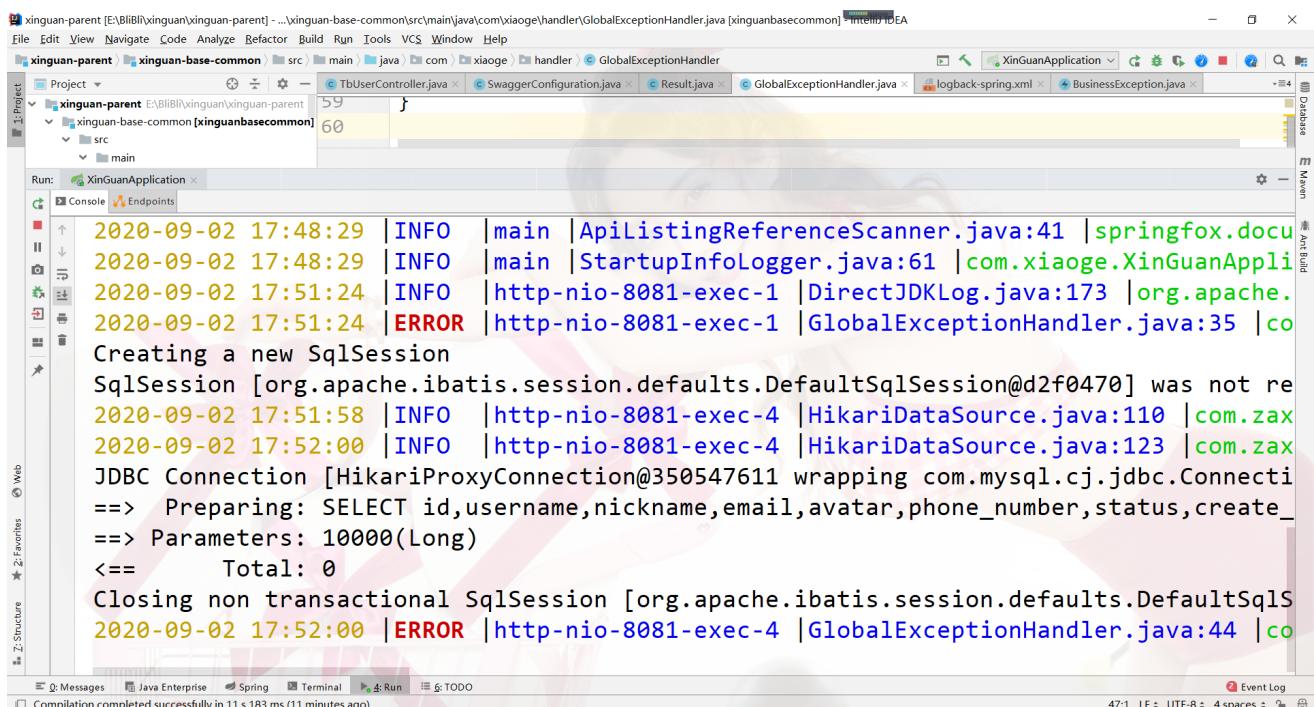
1 package com.xiaoge.handler;
2
3 import com.xiaoge.response.Result;
4 import com.xiaoge.response.ResultCode;
5 import lombok.extern.slf4j.Slf4j;
6 import org.springframework.web.bind.annotation.ControllerAdvice;
7 import org.springframework.web.bind.annotation.ExceptionHandler;
8 import org.springframework.web.bind.annotation.RequestMapping;
9 import org.springframework.web.bind.annotation.ResponseBody;
10
11 /**
12  * @author NieChangan
13  */
14 @ControllerAdvice
15 @Slf4j
16 public class GlobalExceptionHandler {
17
18     /**
19      * 全局异常处理,没有指定异常的类型,不管什么异常都是可以捕获的
20      * @param e
21      * @return
22      */
23     @ExceptionHandler(Exception.class)
24     @ResponseBody
25     public Result error(Exception e){
26         //e.printStackTrace();

```

```

27     log.error(e.getMessage());
28     return Result.error();
29 }
30
31 /**
32 * 处理特定异常类型,可以定义多个,这里只以ArithmeticeException为例
33 * @param e
34 * @return
35 */
36 @ExceptionHandler(ArithmeticeException.class)
37 @ResponseBody
38 public Result error(ArithmeticeException e){
39     //e.printStackTrace();
40     log.error(e.getMessage());
41     return Result.error().code(ResultCode.ARITHMETIC_EXCEPTION.getCode())
42         .message(ResultCode.ARITHMETIC_EXCEPTION.getMessage());
43 }
44
45 /**
46 * 处理业务异常,我们自己定义的异常
47 * @param e
48 * @return
49 */
50 @ExceptionHandler(BusinessException.class)
51 @ResponseBody
52 public Result error(BusinessException e){
53     //e.printStackTrace();
54     log.error(e.getErrMsg());
55     return Result.error().code(e.getCode())
56         .message(e.getErrMsg());
57 }
58 }

```



## 发现打印的日志颜色不一样了,本地也有日志输出



```
1 2020-09-02 12:18:25.423 [main] ERROR o.s.b.diagnostics.LoggingFailureAnalysisReporter -  
2  
3 ****  
4 APPLICATION FAILED TO START  
5 ****  
6  
7 Description:  
8  
9 Web server failed to start. Port 8081 was already in use.  
10  
11 Action:  
12  
13 Identify and stop the process that's listening on port 8081 or configure this application to listen on another port.  
14  
15 2020-09-02 12:19:14.307 [http-nio-8081-exec-2] ERROR com.xiaoge.handler.GlobalExceptionHandler - 用户不存在  
16 2020-09-02 17:51:24.052 [http-nio-8081-exec-1] ERROR com.xiaoge.handler.GlobalExceptionHandler - / by zero  
17 2020-09-02 17:52:00.161 [http-nio-8081-exec-4] ERROR com.xiaoge.handler.GlobalExceptionHandler - 用户不存在  
18
```

刚测试的用户不存在也被收集了