广大Java网友好，最近锋哥要录制一套权限系统实战课程，定位是应届生国内高级Java就业实战课程，希望大家喜欢，本套基于SpringBoot+SpringSecurity后端的实战基本版免费，开源，包括文档，感谢广大Java粉丝多年来的支持。本课程 B站 java1234 连载更新中...敬请期待...

本套课程需要一定的Java基础，如果基础薄弱的，建议大家先把基础学习下，锋哥给下Java学习路线图，基础都免费的哦。

http://www.java1234.vip/article/1

前言

课程简介

本课程采用主流的技术栈实现，Mysql数据库，SpringBoot2+Mybatis Plus后端，redis缓存，安全框架SpringSecurity，Vue3.2+Element Plus实现后台管理。基于JWT技术实现前后端分离。项目开发同时采用MybatisX插件生成代码，提高开发效率。

基于SpringSecurity实现了 登录验证鉴权功能，用户管理，角色管理，权限管理。

高清视频+源码+配套文档获取

关注 **java1234** 公众号，回复 **权限**

本套课程的基本版高清视频+源码+配套文档

进本课程专属微信交流群

加上面锋哥微信，**java9579**　备用 **java8822**，备注暗号：**小爱爱**

即可进入微信小程序电商实战课程的专属技术交流群。

作者简介

我叫曹锋，网名：java1234_小锋 江苏南通人，12年毕业于江苏师范大学，计算机与科学技术专业，10年Java老兵，资深Java讲师，Java技术自媒体人，南通小锋网络科技有限公司光杆司令员，司令部：www.java1234.vip 还有一个人网站：www.java1234.com

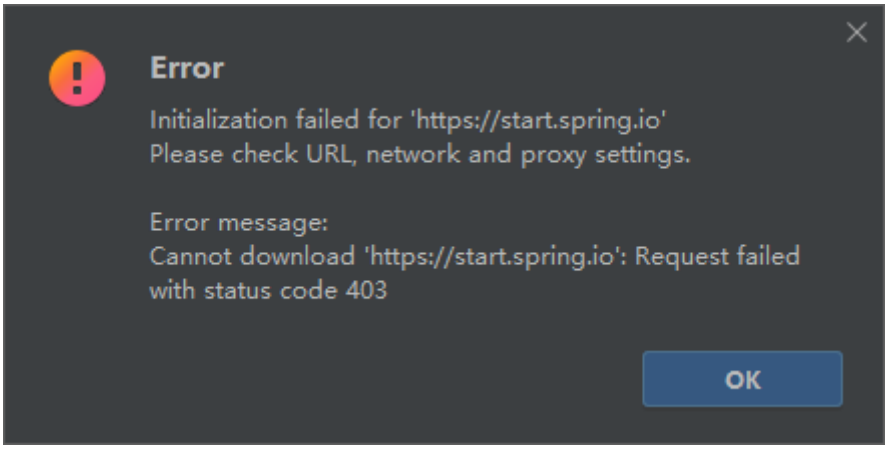
爆丑照，美颜后，依然很丑。

架构搭建

后端架构搭建

新建**java1234-admin3**项目

如果遇到报错



换成 https://start.aliyun.com

pom.xml修改：

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.3.2.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.java1234</groupId>
    <artifactId>java1234-admin</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>java1234-admin</name>
    <description>java1234-admin</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>

        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
```

```
            <scope>runtime</scope>
        </dependency>

        <!-- 连接池 -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>
            <version>1.1.10</version>
        </dependency>
        <!-- mybatis-plus -->
        <dependency>
            <groupId>com.baomidou</groupId>
            <artifactId>mybatis-plus-boot-starter</artifactId>
            <version>3.3.2</version>
        </dependency>
        <dependency>
            <groupId>com.baomidou</groupId>
            <artifactId>mybatis-plus-boot-starter</artifactId>
            <version>3.3.2</version>
        </dependency>

        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.2.40</version>
        </dependency>

        <!-- JWT -->
        <dependency>
            <groupId>com.auth0</groupId>
            <artifactId>java-jwt</artifactId>
            <version>3.2.0</version>
        </dependency>
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt</artifactId>
            <version>0.9.1</version>
        </dependency>

        <dependency>
            <groupId>commons-io</groupId>
            <artifactId>commons-io</artifactId>
            <version>2.5</version>
        </dependency>

        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>

        <!-- spring boot redis 缓存引入 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-redis</artifactId>
        </dependency>
        <!-- lettuce pool 缓存连接池 -->
        <dependency>
```

```
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-pool2</artifactId>
        </dependency>

        <!-- hutool工具类-->
        <dependency>
            <groupId>cn.hutool</groupId>
            <artifactId>hutool-all</artifactId>
            <version>5.3.3</version>
        </dependency>



    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <version>2.3.2.RELEASE</version>
            </plugin>
        </plugins>
    </build>


</project>
```

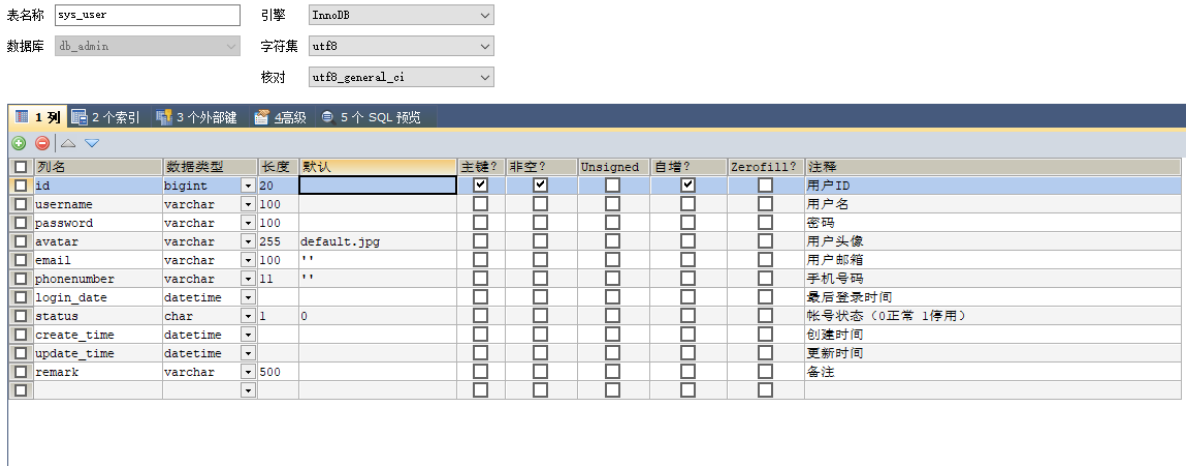
新建application.yml

```
server:
  port: 80
  servlet:
    context-path: /

spring:
  datasource:
    type: com.alibaba.druid.pool.DruidDataSource
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/db_admin3?serverTimezone=Asia/Shanghai
    username: root
    password: 123456
  redis: # redis配置
    host: 127.0.0.1 # IP
    port: 6379  # 端口
    password:  # 密码
    connect-timeout: 10s  # 连接超时时间
    lettuce: # lettuce redis客户端配置
      pool: # 连接池配置
        max-active: 8  # 连接池最大连接数（使用负值表示没有限制） 默认 8
        max-wait: 200s  # 连接池最大阻塞等待时间（使用负值表示没有限制） 默认 -1
        max-idle: 8 # 连接池中的最大空闲连接 默认 8
        min-idle: 0 # 连接池中的最小空闲连接 默认 0
```

```yaml
mybatis-plus:
  global-config:
    db-config:
      id-type: auto
  configuration:
    map-underscore-to-camel-case: true
    auto-mapping-behavior: full
    log-impl: org.apache.ibatis.logging.stdout.StdOutImpl
  mapper-locations: classpath:mapper/*.xml
```

新建数据库**db_admin3**

新建用户表**sys_user**

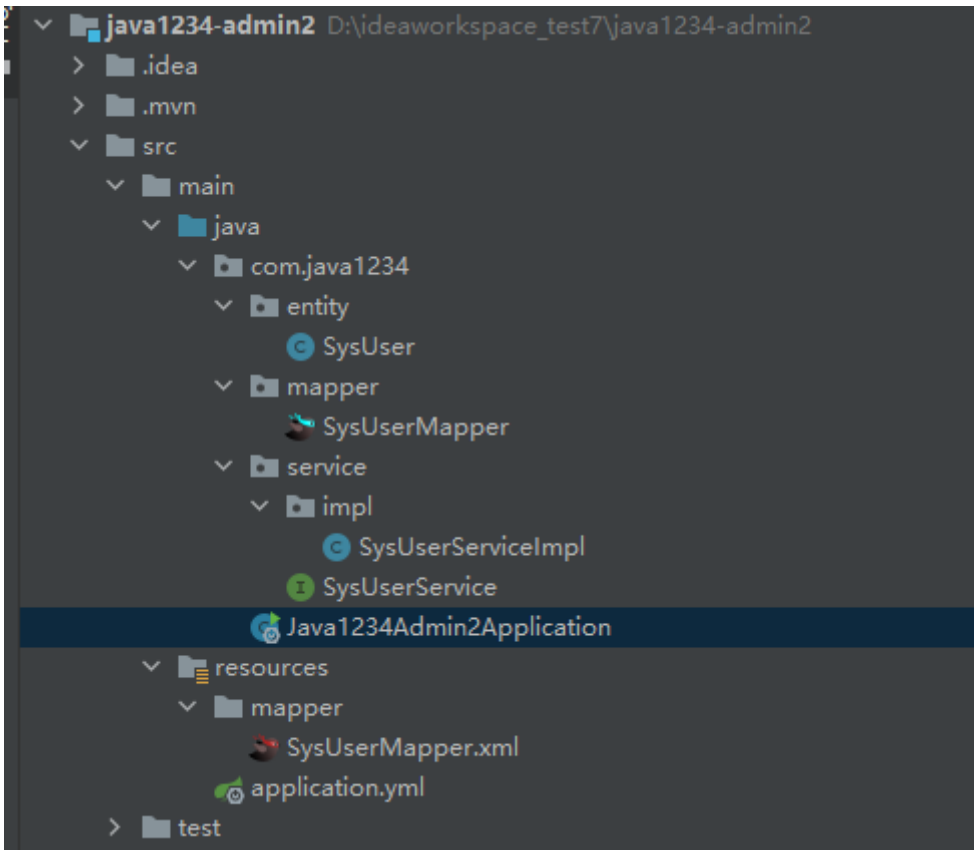


列名	数据类型	长度	默认	主键?	非空?	Unsigned	自增?	Zerofill?	注释
id	bigint	20		☑	☑	☐	☑	☐	用户ID
username	varchar	100		☐	☐	☐	☐	☐	用户名
password	varchar	100		☐	☐	☐	☐	☐	密码
avatar	varchar	255	default.jpg	☐	☐	☐	☐	☐	用户头像
email	varchar	100	''	☐	☐	☐	☐	☐	用户邮箱
phonenumber	varchar	11	''	☐	☐	☐	☐	☐	手机号码
login_date	datetime			☐	☐	☐	☐	☐	最后登录时间
status	char	1	0	☐	☐	☐	☐	☐	帐号状态（0正常 1停用）
create_time	datetime			☐	☐	☐	☐	☐	创建时间
update_time	datetime			☐	☐	☐	☐	☐	更新时间
remark	varchar	500		☐	☐	☐	☐	☐	备注

```sql
CREATE TABLE `sys_user` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '用户ID',
  `username` varchar(100) DEFAULT NULL COMMENT '用户名',
  `password` varchar(100) DEFAULT NULL COMMENT '密码',
  `avatar` varchar(255) DEFAULT 'default.jpg' COMMENT '用户头像',
  `email` varchar(100) DEFAULT '' COMMENT '用户邮箱',
  `phonenumber` varchar(11) DEFAULT '' COMMENT '手机号码',
  `login_date` datetime DEFAULT NULL COMMENT '最后登录时间',
  `status` char(1) DEFAULT '0' COMMENT '帐号状态（0正常 1停用）',
  `create_time` datetime DEFAULT NULL COMMENT '创建时间',
  `update_time` datetime DEFAULT NULL COMMENT '更新时间',
  `remark` varchar(500) DEFAULT NULL COMMENT '备注',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=31 DEFAULT CHARSET=utf8;

/*Data for the table `sys_user` */
```

```
insert  into
`sys_user`(`id`,`username`,`password`,`avatar`,`email`,`phonenumber`,`login_date
`,`status`,`create_time`,`update_time`,`remark`) values
(1,'java1234','$2a$10$kib4zuVhTzg3I1CoqJfdOunuY9G9ysI7cfbhyT3fi7k7Z/4pr3bGW','20
220727112556000000325.jpg','caofeng4017@126.com','18862857417','2022-08-29
22:10:52','0','2022-06-09 08:47:52','2022-06-22 08:47:54','备注'),
(2,'common','$2a$10$tiArwmOGxChyEP5kOJGzsOuzyY15IKA.ZTl8S2aj3haYlKAfpwfl.','222.
jpg','','','2022-08-22 21:34:39','0',NULL,NULL,NULL),
(3,'test','$2a$10$tiArwmOGxChyEP5kOJGzsOuzyY15IKA.ZTl8S2aj3haYlKAfpwfl.','333.jp
g','','','2022-07-24 17:36:07','0',NULL,NULL,NULL),
(4,'1','$2a$10$lDOFx7oMsFFmX9hVkmYy7eJteH8pBaXXro1X9DEMP5sbM.Z6Co55m','default.j
pg','','',NULL,'1',NULL,NULL,NULL),
(5,'2',NULL,'default.jpg','','',NULL,'1',NULL,NULL,NULL),
(15,'fdsfs','$2a$10$AQVcp4hQ7REc5o7ztVnI7eX.sJdcYy3d1x2jm5CfrcCoMZMPacfpi','defa
ult.jpg','fdfa4@qq.com','18862851414','2022-08-02 02:22:45','1','2022-08-02
02:21:24','2022-08-01 18:23:16','fdfds4'),
(28,'sdfss2','$2a$10$7aNJxwVmefIOXAk64vrzYuOqeeImYJUQnoBrtKP9pLTGTWO2CXQ/y','def
ault.jpg','dfds3@qq.com','18862857413',NULL,'1','2022-08-07 00:42:46','2022-08-
06 16:43:04','ddd33'),
(29,'ccc','$2a$10$7cbWeVwDWO9Hh3qbJrvTHOnOE/DLYXxnIZpxZeiOjY4ChfQbJuhi.','202208
29080150000000341.jpg','3242@qq.com','18862584120','2022-08-29
19:52:27','0','2022-08-29 17:04:58',NULL,'xxx'),
(30,'ccc666','$2a$10$Tmw5VCM/K2vb837AZDYHQOqE3gPiRZKevxLsh/ozndpTSjdwABqaK','202
208291004540000007 71.jpg','fdafds@qq.com','18865259845','2022-08-29
22:05:18','0','2022-08-29 22:00:39',NULL,'ccc');
```

使用MybatisX插件，生成代码



启动类修改

```java
package com.java1234;

import org.mybatis.spring.annotation.MapperScan;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
@MapperScan("com.java1234.mapper")
public class Java1234AdminApplication {

    public static void main(String[] args) { SpringApplication.run(Java1234AdminApplication.class, args); }

}
```

```
@MapperScan("com.java1234.mapper")
```

写一个测试Controller类

```java
@RestController
@RequestMapping("/test")
public class TestController {

    @Autowired
    private SysUserService sysUserService;

    @GetMapping("/user/list")
    public R userList(){
        Map<String,Object> resutMap=new HashMap<>();
        List<SysUser> userList = sysUserService.list();
        resutMap.put("userList",userList);
        return R.ok(resutMap);
    }

}
```

浏览器输入：http://localhost/test/user/list


{obj} JSON 2
Num code : 200
[arr] userList : 9
{obj} 0 11
Num id : 1
"str" username : java1234
"str" password : $2a$10$Kib4zuVhTzg3I1CoqJfd0unuY9G9ysI7cfbhyT3fi7k7Z/4pr3bGW
"str" avatar : 20220727112556000000325.jpg
"str" email : caofeng4017@126.com
"str" phonenumber : 18862857417
"str" loginDate : 2022-08-29T14:10:52.000+00:00
"str" status : 0
"str" createTime : 2022-06-09T00:47:52.000+00:00
"str" updateTime : 2022-06-22T00:47:54.000+00:00
"str" remark : 备注
{obj} 1 11
Num id : 2
"str" username : common
"str" password : $2a$10$tiArwm0GxChyEP5k0JGzsOuzyY15IKA.ZTl8S2aj3haYlKAfpwfl.
"str" avatar : 222.jpg
"str" email :
"str" phonenumber :
"str" loginDate : 2022-08-22T13:34:39.000+00:00
"str" status : 0
Null createTime : null
Null updateTime : null
Null remark : null
{obj} 2 11
Num id : 3
"str" username : test
"str" password : $2a$10$tiArwm0GxChyEP5k0JGzsOuzyY15IKA.ZTl8S2aj3haYlKAfpwfl.
"str" avatar : 333.jpg
"str" email :
"str" phonenumber :
"str" loginDate : 2022-07-24T09:36:07.000+00:00
"str" status : 0
Null createTime : null
Null updateTime : null
Null remark : null
{obj} 3 11
Num id : 4
"str" username : 1
"str" password : $2a$10$lD0Fx7oMsFFmX9hVkmYy7eJteH8pBaXXro1X9DEMP5sbM.Z6Co55m
"str" avatar : default.jpg
"str" email :
"str" phonenumber :
Null loginDate : null
"str" status : 1
Null createTime : null
Null updateTime : null
Null remark : null
{obj} 4 11

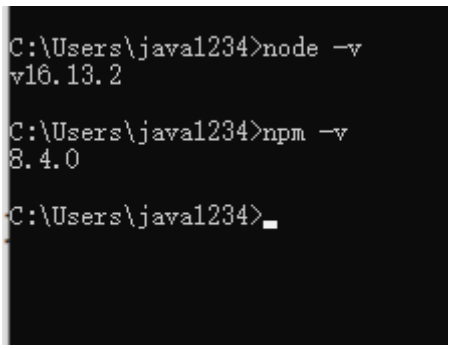
说明OK。

前端架构搭建

我们用vue ui来搭建vue项目；

vue ui是一个可视化图形界面，方便你去创建、更新和管理vue项目，包括下载router，vuex，axios，elementui等插件，配置好一些属性以及依赖关系，方便我们使用，我个人第一次接触它就感觉非常非常非常智能和强大。

1，安装node



2，安装Vue Cli

vue-cli 是一个官方发布 vue.js 项目脚手架，使用 vue-cli 可以快速创建 vue 项目。

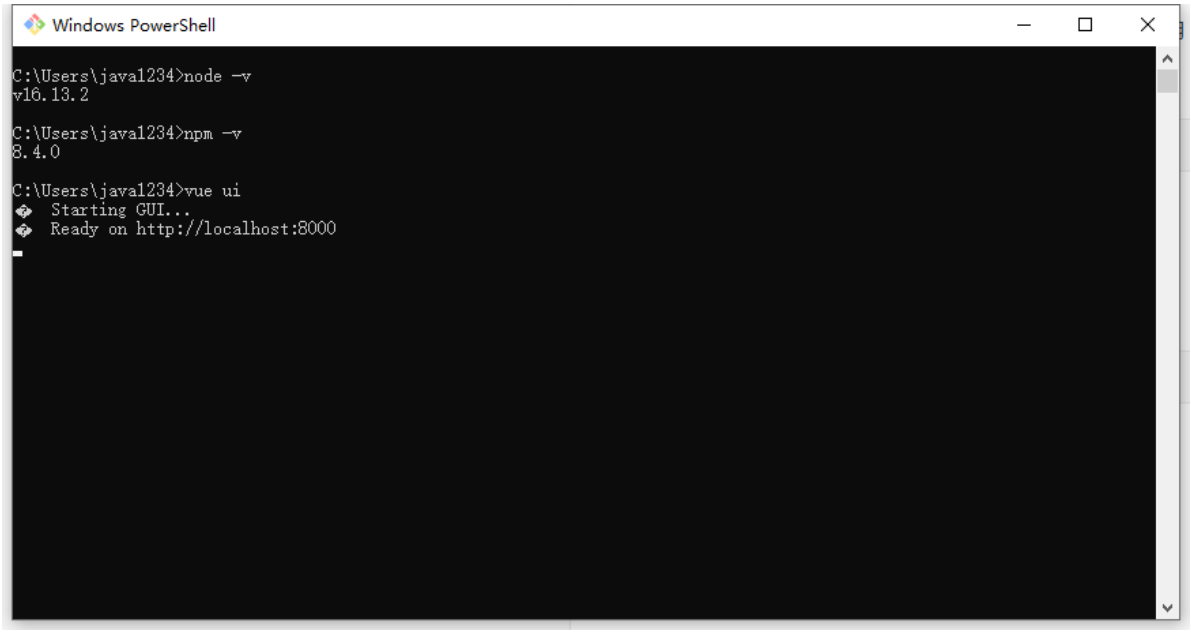
因为vue ui是在Vue CLI基础上封装的

```
npm install -g @vue/cli
```

3，vue ui搭建vue项目

命令行窗口执行：

```
vue ui
```



4，安装axios element-plus

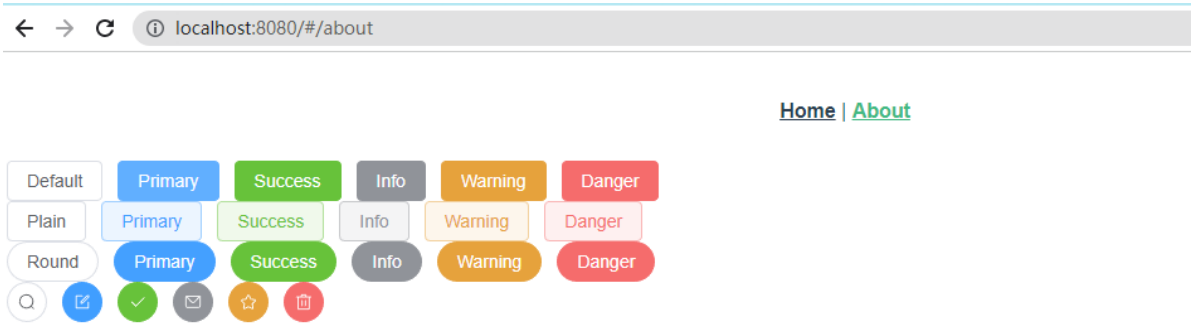
element-plus官网：

https://element-plus.gitee.io/

5，项目测试

导入elementplus以及样式

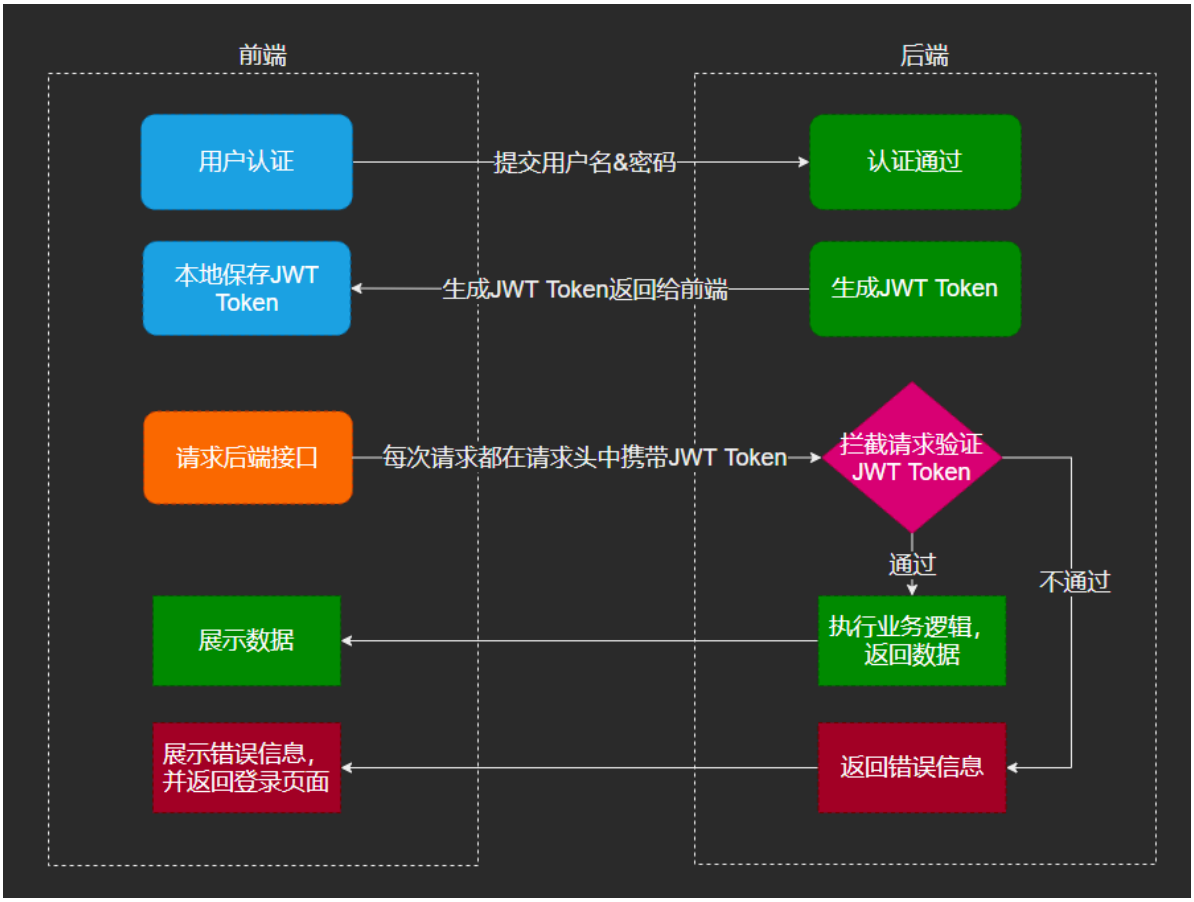
```
import ElementPlus from 'element-plus'
import 'element-plus/dist/index.css'

createApp(App).use(store).use(router).use(ElementPlus).mount('#app')
```



引入JWT前后端交互

Json web token (JWT), 是为了在网络应用环境间传递声明而执行的一种基于JSON的开放标准（(RFC 7519);

JWT就是一段字符串，用来进行用户身份认证的凭证，该字符串分成三段【头部、载荷、签证】



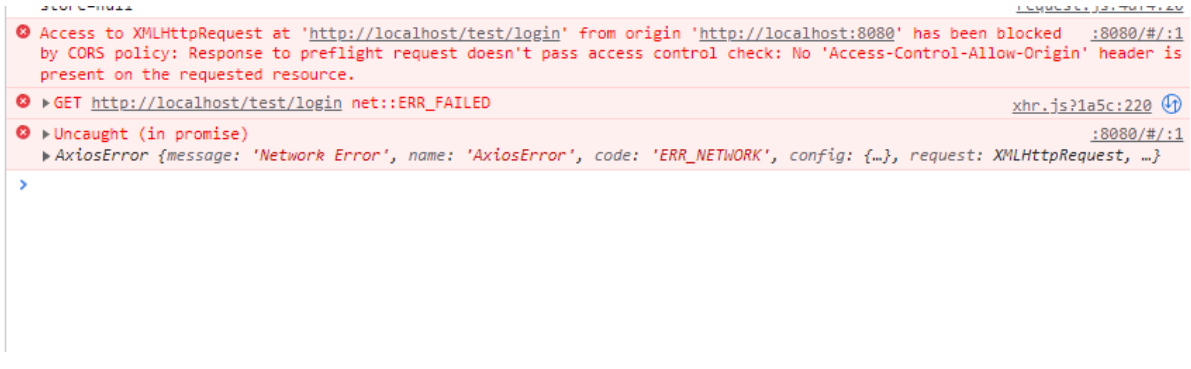
后端接口测试：

```
@RestController
@RequestMapping("/test")
public class TestController {

    @Autowired
    private SysUserService sysUserService;

    @GetMapping("/user/list")
    public R userList(@RequestHeader(required = false) String token){
        if(StringUtil.isNotEmpty(token)){
            Map<String,Object> resutMap=new HashMap<>();
            List<SysUser> userList = sysUserService.list();
            resutMap.put("userList",userList);
            return R.ok(resutMap);
        }else{
            return R.error(401,"没有权限访问");
        }


    }

    @GetMapping("/login")
    public R login(){
        String token= JwtUtils.genJwtToken("java1234");
        return R.ok().put("token",token);
    }

}
```

前端测试：

```
<template>

  <el-button type="primary" @click="handleLogin">测试登录</el-button>
  <el-button type="danger" @click="handleUserList">测试获取用户列表信息</el-button>

</template>
<script setup>
import requestUtil from '@/util/request'
import store from '@/store'

const handleLogin=async()=>{
  let result=await requestUtil.get("test/login");
  let data=result.data;
  if(data.code==200){
    const token=data.token;
    console.log("登录成功：token="+token);
    store.commit('SET_TOKEN',token);
  }else{
    console.log("登录出错！")
  }
}

const handleUserList=async ()=>{
  let result=await requestUtil.get("test/user/list");
  let data=result.data;
  if(data.code==200){
    const userList=data.userList;
    console.log("用户列表信息：userList="+userList);
  }else{
    console.log("出错！")
  }
}

</script>
<style>

</style>
```

跨越问题：

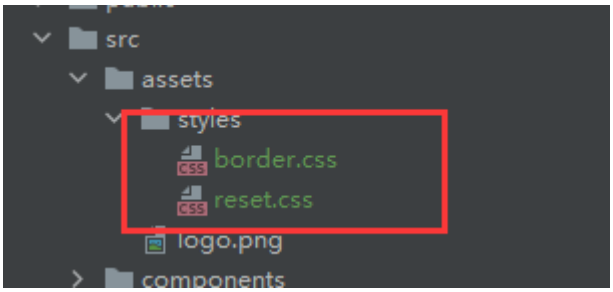


```
@Configuration
public class WebAppConfigurer implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**")
                .allowedOrigins("*")
                .allowCredentials(true)
                .allowedMethods("GET", "HEAD", "POST", "PUT",
"DELETE","OPTIONS")
                .maxAge(3600);
    }


}
```

登录功能实现

前端静态登录页面实现

引入全局样式：



main.js导入样式文件：

```
import '@/assets/styles/border.css'
import '@/assets/styles/reset.css'
```

加路由：

```
const routes = [

  {
    path: '/login',
    name: 'login',
    component: () => import('../views/Login.vue')
  }
]
```

App.vue

```
<template>
  <router-view/>
</template>
```

安装sass和sass-loader依赖

Login.vue

```
<template>
<div class="login">

    <el-form ref="loginRef" :model="loginForm" :rules="loginRules" class="login-
form">
      <h3 class="title">Java1234 Vue3 后台管理系统</h3>

      <el-form-item prop="username">

        <el-input

            type="text"
            size="large"
            auto-complete="off"
            placeholder="账号"
        >

        </el-input>
      </el-form-item>
      <el-form-item prop="password">
```

```
        <el-input

            type="password"
            size="large"
            auto-complete="off"
            placeholder="密码"
            @keyup.enter="handleLogin"
        >

        </el-input>
      </el-form-item>


      <el-checkbox  style="margin:0px 0px 25px 0px;">记住密码</el-checkbox>
      <el-form-item style="width:100%;">
        <el-button
            size="large"
            type="primary"
            style="width:100%;"
            @click.prevent="handleLogin"
        >
          <span>登 录</span>

        </el-button>

      </el-form-item>
    </el-form>
    <!--  底部  -->
    <div class="el-login-footer">
      <span>Copyright © 2013-2022 <a href="http://www.java1234.vip"
target="_blank">java1234.vip</a> 版权所有.</span>
    </div>
  </div>
</template>

<script setup>

</script>

<style lang="scss" scoped>
a{
  color:white
}
.login {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100%;
  background-image: url("../assets/images/login-background.jpg");
  background-size: cover;
}
.title {
  margin: 0px auto 30px auto;
  text-align: center;
  color: #707070;
}

.login-form {
```

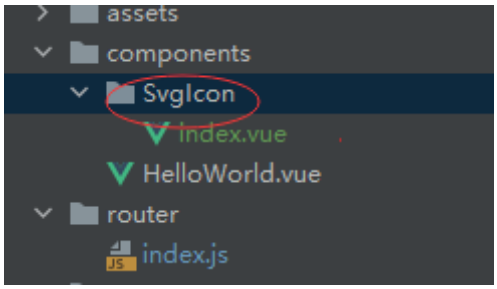
```
  border-radius: 6px;
  background: #ffffff;
  width: 400px;
  padding: 25px 25px 5px 25px;

  .el-input {
    height: 40px;



    input {
      display: inline-block;
      height: 40px;
    }
  }
  .input-icon {
    height: 39px;
    width: 14px;
    margin-left: 0px;
  }

}
.login-tip {
  font-size: 13px;
  text-align: center;
  color: #bfbfbf;
}
.login-code {
  width: 33%;
  height: 40px;
  float: right;
  img {
    cursor: pointer;
    vertical-align: middle;
  }
}
.el-login-footer {
  height: 40px;
  line-height: 40px;
  position: fixed;
  bottom: 0;
  width: 100%;
  text-align: center;
  color: #fff;
  font-family: Arial;
  font-size: 12px;
  letter-spacing: 1px;
}
.login-code-img {
  height: 40px;
  padding-left: 12px;
}
</style>
```

App.vue设置下全局样式：

```
<style>
html,body,#app{
  height: 100%;
}

.app-container{
  padding:20px
}
</style>
```

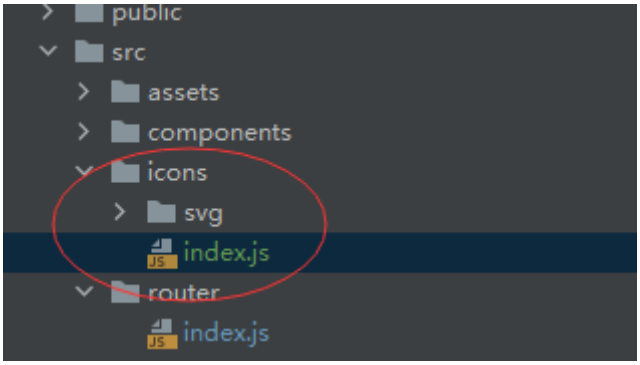
自定义icon实现

component下新建SvgIcon目录，再新建index.vue 定义svg-icon组件




```
<template>
  <svg class="svg-icon" aria-hidden="true">
    <use :xlink:href="iconName"></use>
  </svg>
</template>

<script setup>
import { defineProps, computed } from 'vue'
const props = defineProps({
  icon: {
    type: String,
    required: true
  }
})

const iconName = computed(() => {
  return `#icon-${props.icon}`
})
</script>

<style lang="scss" scoped>
.svg-icon {
  width: 1em;
  height: 1em;
  vertical-align: -0.15em;
  fill: currentColor;
  overflow: hidden;
}
</style>
```

新建icons文件夹，放一个svg文件的文件夹目录，以及新建index.js，全局定义组件




```
import SvgIcon from '@/components/SvgIcon'

const svgRequired = require.context('./svg', false, /\.svg$/)
svgRequired.keys().forEach((item) => svgRequired(item))

export default (app) => {
  app.component('svg-icon', SvgIcon)
}
```

安装依赖 webpack

安装依赖 svg-sprite-loader

vue.config.js

```
const webpack = require('webpack');

const path = require('path')
```

```
function resolve(dir) {
  return path.join(__dirname, dir)
}

module.exports = {
  lintOnSave: false,

  chainWebpack(config) {
    // 设置 svg-sprite-loader
    // config 为 webpack 配置对象
    // config.module 表示创建一个具名规则，以后用来修改规则
    config.module
        // 规则
        .rule('svg')
        // 忽略
        .exclude.add(resolve('src/icons'))
        // 结束
        .end()
    // config.module 表示创建一个具名规则，以后用来修改规则
    config.module
        // 规则
        .rule('icons')
        // 正则，解析 .svg 格式文件
        .test(/\.svg$/)
        // 解析的文件
        .include.add(resolve('src/icons'))
        // 结束
        .end()
        // 新增了一个解析的loader
        .use('svg-sprite-loader')
        // 具体的loader
        .loader('svg-sprite-loader')
        // loader 的配置
        .options({
          symbolId: 'icon-[name]'
        })
        // 结束
        .end()
    config
        .plugin('ignore')
        .use(
            new webpack.ContextReplacementPlugin(/moment[/\\]locale$/, /zh-cn$/)
        )
    config.module
        .rule('icons')
        .test(/\.svg$/)
        .include.add(resolve('src/icons'))
        .end()
        .use('svg-sprite-loader')
        .loader('svg-sprite-loader')
        .options({
          symbolId: 'icon-[name]'
        })
        .end()
  }
}
```

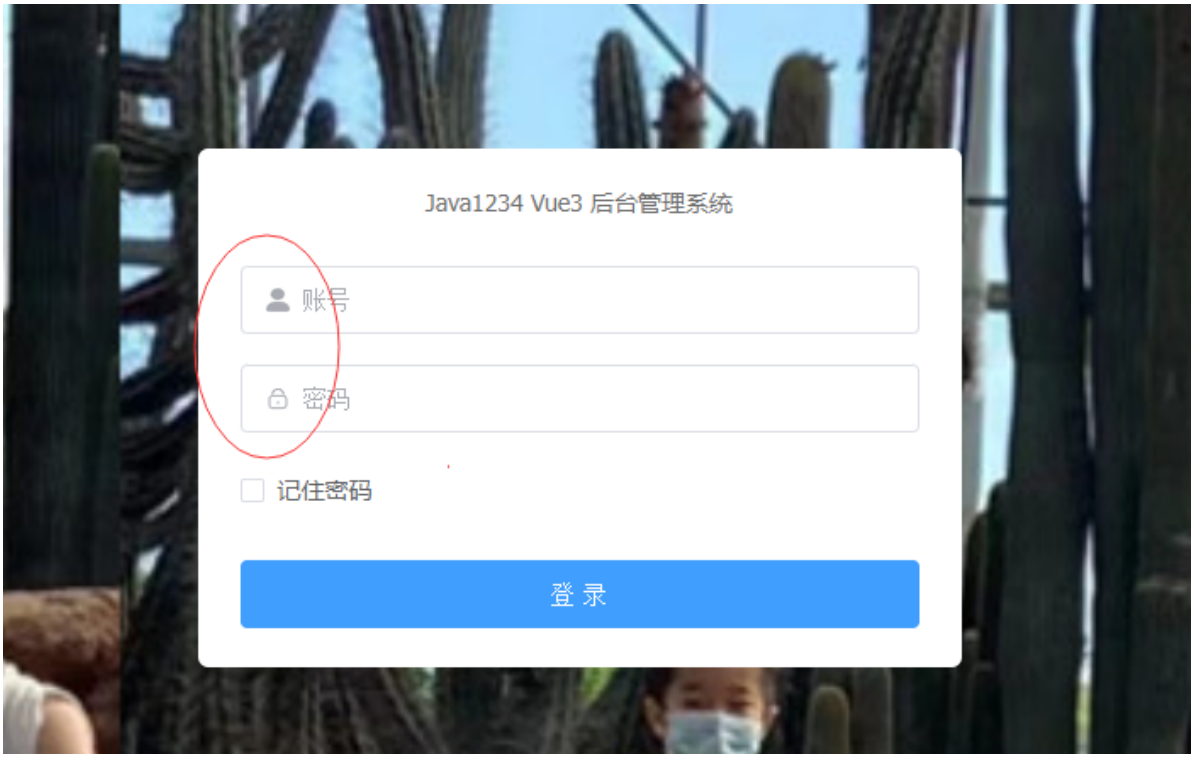
main.js修改：

```
import SvgIcon from '@/icons'

const app=createApp(App)
SvgIcon(app);

app.use(store)

app.use(router)

app.use(ElementPlus)
app.mount('#app')
```

使用语法：

```
<template #prefix><svg-icon icon="user" /></template>
```

```
<el-form-item prop="username">

  <el-input

      type="text"
      size="large"
      auto-complete="off"
      placeholder="账号"
  >
  <template #prefix><svg-icon icon="user" /></template>
  </el-input>
</el-form-item>
```



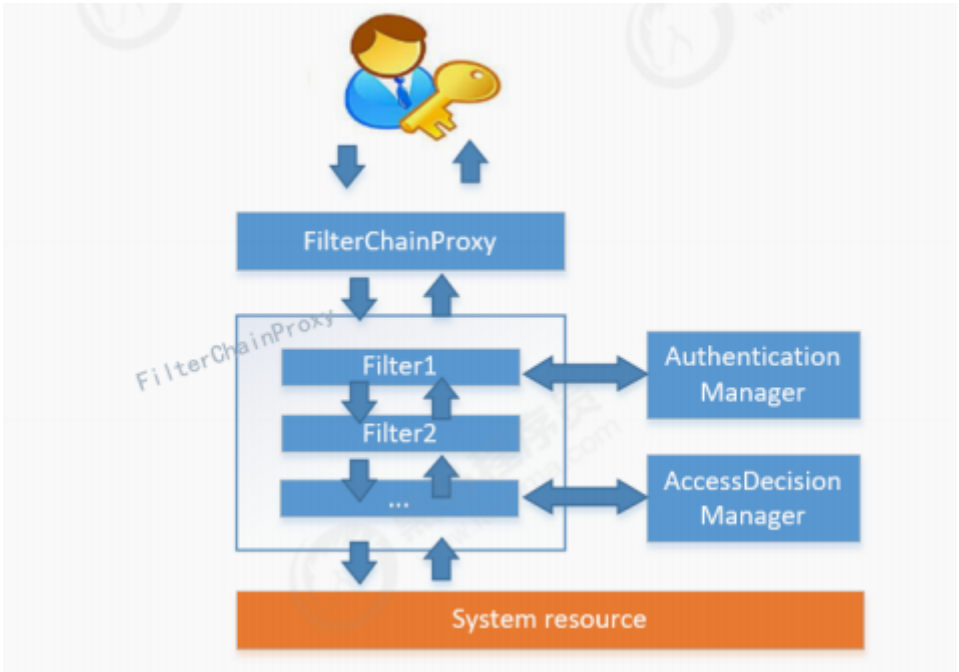
SpringSecurity执行原理概述

spring security的简单原理：
SpringSecurity有很多很多的拦截器，在执行流程里面主要有两个核心的拦截器

1，登陆验证拦截器**AuthenticationProcessingFilter**

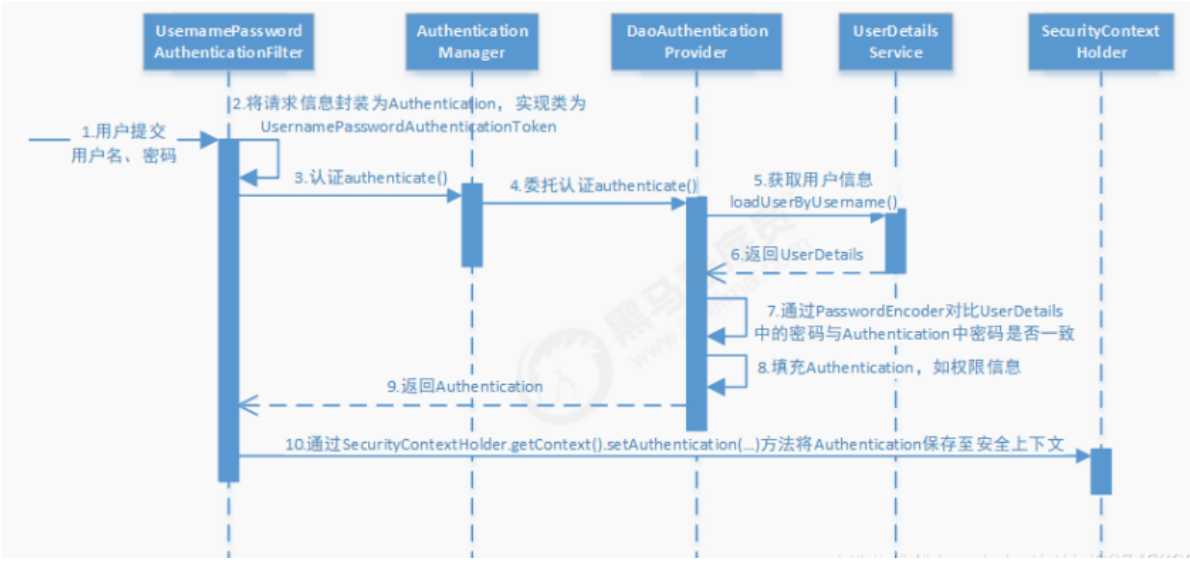
2，资源管理拦截器**AbstractSecurityInterceptor**

但拦截器里面的实现需要一些组件来实现，所以就有了**AuthenticationManager认证管理器、accessDecisionManager决策管理器**等组件来支撑。



FilterChainProxy是一个代理，真正起作用的是各个Filter，这些Filter作为Bean被Spring管理，是Spring Security核心，各有各的职责，不直接处理认证和授权，交由认证管理器和决策管理器处理！

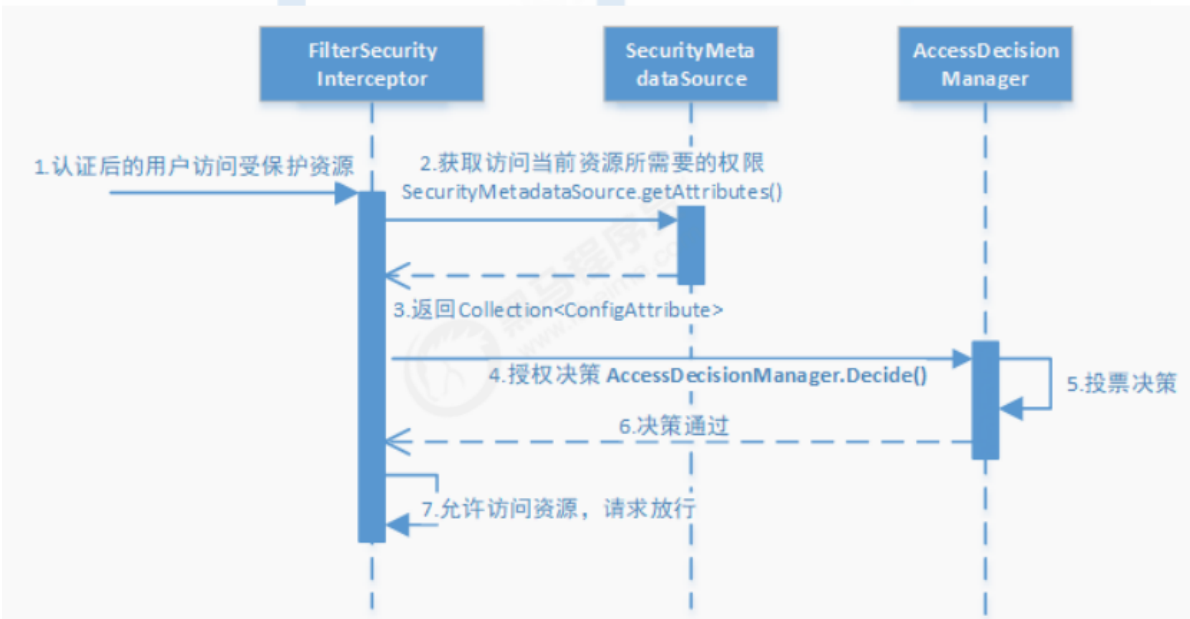
大概流程



认证管理

流程图解读：
1、用户提交用户名、密码被SecurityFilterChain中的 UsernamePasswordAuthenticationFilter 过滤器获取到， 封装为请求Authentication，通常情况下是UsernamePasswordAuthenticationToken这个实现类。
2、然后过滤器将Authentication提交至认证管理器（AuthenticationManager）进行认证 。
3、认证成功后， AuthenticationManager 身份管理器返回一个被填充满了信息的（包括上面提到的权限信息， 身份信息，细节信息，但密码通常会被移除） Authentication 实例。
4、SecurityContextHolder 安全上下文容器将第3步填充了信息的 Authentication ，通过 SecurityContextHolder.getContext().setAuthentication(...)方法，设置到其中。 可以看出 AuthenticationManager接口（认证管理器）是认证相关的核心接口，也是发起认证的出发点，它 的实现类为ProviderManager。而Spring Security支持多种认证方式，因此ProviderManager维护着一个 List 列表，存放多种认证方式，最终实际的认证工作是由 AuthenticationProvider完成的。咱们知道 web表单的对应的AuthenticationProvider实现类为 DaoAuthenticationProvider，它的内部又维护着一个UserDetailsService负责UserDetails的获取。最终 AuthenticationProvider将UserDetails填充至 Authentication。

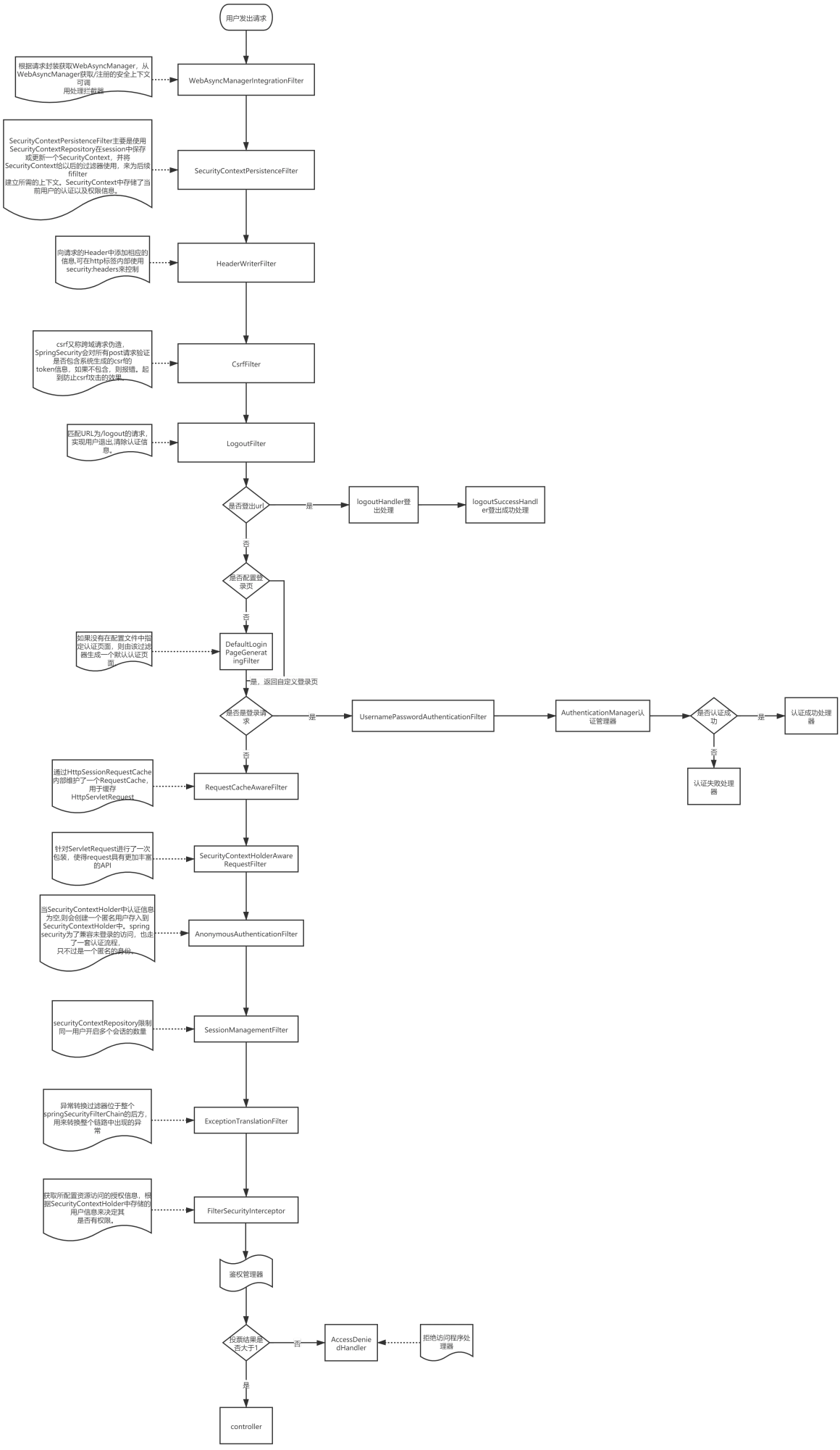
授权管理



访问资源（即授权管理），访问url时，会通过FilterSecurityInterceptor拦截器拦截，其中会调用SecurityMetadataSource的方法来获取被拦截url所需的全部权限，再调用授权管理器AccessDecisionManager，这个授权管理器会通过spring的全局缓存SecurityContextHolder获取用户的权限信息，还会获取被拦截的url和被拦截url所需的全部权限，然后根据所配的投票策略（有：一票决定，一票否定，少数服从多数等），如果权限足够，则决策通过，返回访问资源，请求放行，否则跳转到403页面、自定义页面。

转载自：https://blog.csdn.net/weixin_51542566/article/details/119705963

详细执行流程


用户发出请求
根据请求封装获取WebAsyncManager，从WebAsyncManager获取/注册的安全上下文可调
用处理拦截器
WebAsyncManagerIntegrationFilter
SecurityContextPersistenceFilter主要是使用SecurityContextRepository在session中保存或更新一个SecurityContext，并将SecurityContext给以后的过滤器使用，来为后续fifilter
建立所需的上下文。SecurityContext中存储了当前用户的认证以及权限信息。
SecurityContextPersistenceFilter
向请求的Header中添加相应的信息,可在http标签内部使用security:headers来控制
HeaderWriterFilter
csrf又称跨域请求伪造，SpringSecurity会对所有post请求验证是否包含系统生成的csrf的token信息，如果不包含，则报错。起到防止csrf攻击的效果。
CsrfFilter
匹配URL为/logout的请求，实现用户退出,清除认证信息。
LogoutFilter
是否登出url
是
logoutHandler登出处理
logoutSuccessHandler登出成功处理
否
是否配置登录页
否
如果没有在配置文件中指定认证页面，则由该过滤器生成一个默认认证页面
DefaultLoginPageGeneratingFilter
是，返回自定义登录页
是否是登录请求
是
UsernamePasswordAuthenticationFilter
AuthenticationManager认证管理器
是否认证成功
是
认证成功处理器
否
认证失败处理器
否
通过HttpSessionRequestCache内部维护了一个RequestCache，用于缓存HttpServletRequest
RequestCacheAwareFilter
针对ServletRequest进行了一次包装，使得request具有更加丰富的API
SecurityContextHolderAwareRequestFilter
当SecurityContextHolder中认证信息为空,则会创建一个匿名用户存入到SecurityContextHolder中。spring security为了兼容未登录的访问，也走了一套认证流程，
只不过是一个匿名的身份。
AnonymousAuthenticationFilter
securityContextRepository限制同一用户开启多个会话的数量
SessionManagementFilter
异常转换过滤器位于整个springSecurityFilterChain的后方，用来转换整个链路中出现的异常
ExceptionTranslationFilter
获取所配置资源访问的授权信息，根据SecurityContextHolder中存储的用户信息来决定其
是否有权限。
FilterSecurityInterceptor
鉴权管理器
投票结果是否大于1
否
AccessDeniedHandler
拒绝访问程序处理器
是
controller

项目整合SpringSecurity

pom.xml加下springsecurity依赖

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

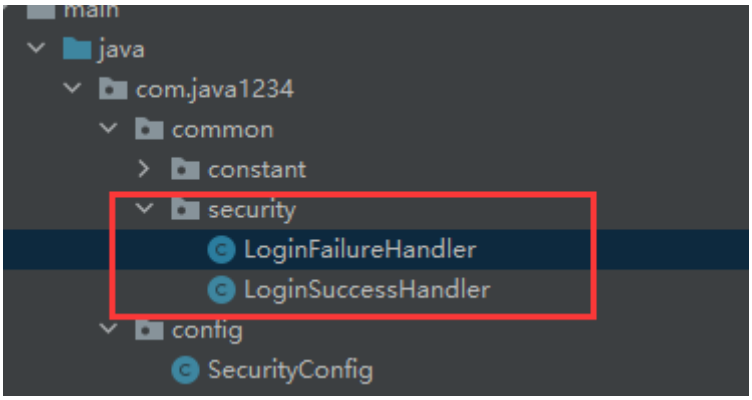
SecurityConfig配置文件：

```
package com.java1234.config;

import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.Authentic
ationManagerBuilder;
import
org.springframework.security.config.annotation.method.configuration.EnableGlobal
MethodSecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecuri
ty;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConf
igurerAdapter;
import org.springframework.security.config.http.SessionCreationPolicy;

/**
 * spring security配置
 * @author java1234_小锋 （公众号：java1234）
 * @site www.java1234.vip
 * @company 南通小锋网络科技有限公司
 */
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    private static final String URL_WHITELIST[] ={
            "/login",
            "/logout",
            "/captcha",
            "/password",
            "/image/**",
            "/test/**"
    } ;

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        // 开启跨域 和 csrf攻击 关闭
        http
            .cors()
            .and()
```

```
            .csrf()
            .disable()

        // 登录配置
        .formLogin()
//                .successHandler()
//                .failureHandler()
//        .and()
//                .logout()
//                .logoutSuccessHandler()

        // session禁用配置
        .and()
            .sessionManagement()
            .sessionCreationPolicy(SessionCreationPolicy.STATELESS)

        // 拦截规则配置
                .and()
                .authorizeRequests()
                .antMatchers(URL_WHITELIST).permitAll()
                .anyRequest().authenticated();


        // 异常处理器配置

        // 自定义过滤器配置
    }


    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception
{
        super.configure(auth);
    }
}
```

重写登录成功和登录失败处理器

common下新建security包，再新建两个类，**LoginSuccessHandler和LoginFailureHandler**




```
@Component
public class LoginSuccessHandler implements AuthenticationSuccessHandler {
```

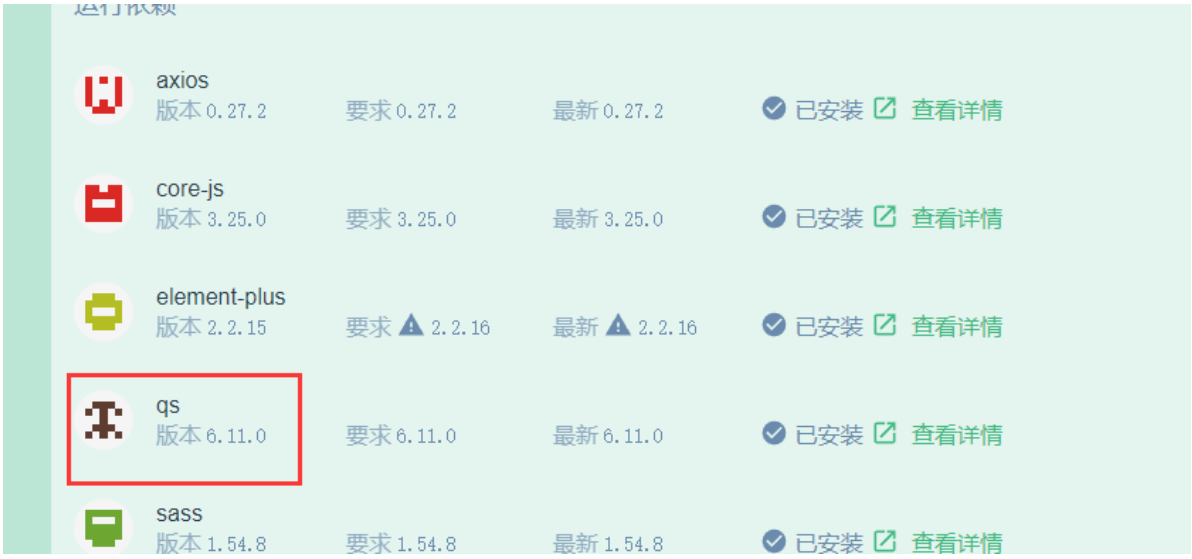
```
    @Override
    public void onAuthenticationSuccess(HttpServletRequest httpServletRequest,
HttpServletResponse httpServletResponse, Authentication authentication) throws
IOException, ServletException {
        httpServletResponse.setContentType("application/json;charset=UTF-8");
        ServletOutputStream outputStream =
httpServletResponse.getOutputStream();

        String username="user";
        String token = JwtUtils.genJwtToken(username);

        outputStream.write(JSONUtil.toJsonStr(R.ok("登录成
功").put("authorization",token)).getBytes());
        outputStream.flush();
        outputStream.close();

    }
}
```

```
@Component
public class LoginFailureHandler implements AuthenticationFailureHandler {

    @Override
    public void onAuthenticationFailure(HttpServletRequest httpServletRequest,
HttpServletResponse httpServletResponse, AuthenticationException e) throws
IOException, ServletException {
        httpServletResponse.setContentType("application/json;charset=UTF-8");
        ServletOutputStream outputStream =
httpServletResponse.getOutputStream();

        String message=e.getMessage();
        if(e instanceof BadCredentialsException){
            message="用户名或者密码错误！";
        }

        outputStream.write(JSONUtil.toJsonStr(R.error(message)).getBytes("UTF-
8"));

        outputStream.flush();
        outputStream.close();
    }
}
```

SecurityConfig注入这两个类对象，然后设置到自定义配置里面去

```java
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private LoginSuccessHandler loginSuccessHandler;

    @Autowired
    private LoginFailureHandler loginFailureHandler;

    private static final String URL_WHITELIST[] ={
            "/login",
            "/logout",
            "/captcha",
            "/password",
            "/image/**",
            "/test/**"
    } ;

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        // 开启跨域 和 csrf攻击 关闭
        http
            .cors() CorsConfigurer<HttpSecurity>
            .and() HttpSecurity
            .csrf() CsrfConfigurer<HttpSecurity>
            .disable() HttpSecurity

        // 登录配置
        .formLogin() FormLoginConfigurer<HttpSecurity>
                .successHandler(loginSuccessHandler)
                .failureHandler(loginFailureHandler)
//          .and()
//                .logout()
//                .logoutSuccessHandler()
```

前端安装qs依赖，用于url参数，把对象转成url字符串



```
    <el-form ref="loginRef" :model="loginForm" :rules="loginRules" class="login-form">
      <h3 class="title">Java1234 Vue3 后台管理系统</h3>

      <el-form-item prop="username">

        <el-input
            v-model="loginForm.username"
            type="text"
            size="large"
            auto-complete="off"
            placeholder="账号"
        >
        <template #prefix><svg-icon icon="user" /></template>
        </el-input>
      </el-form-item>
      <el-form-item prop="password">
        <el-input
            v-model="loginForm.password"
            type="password"
            size="large"
            auto-complete="off"
            placeholder="密码"
            @keyup.enter="handleLogin"
        >
  <template #prefix><svg-icon icon="password" /></template>
        </el-input>
      </el-form-item>
```

```
<script setup>

  import {ref} from "vue"
  import requestUtil from '@/util/request'
  import store from '@/store'
  import {ElMessage} from 'element-plus'
  import qs from 'qs'

  const loginRef=ref(null);

  const loginForm=ref({
    username:"",
    password:""
  })

  const loginRules = {
    username: [{ required: true, trigger: "blur", message: "请输入您的账号" }],
    password: [{ required: true, trigger: "blur", message: "请输入您的密码" }]
  };

  const handleLogin=()=>{
    loginRef.value.validate(async (valid)=>{
      if(valid){
        try{
            let result=await
requestUtil.post("login?"+qs.stringify(loginForm.value))
            let data=result.data;
            if(data.code==200){
              const token = data.authorization;
```

```
              store.commit('SET_TOKEN',token);
            }else{
              ElMessage.error(data.msg)
            }
        }catch(err){
          console.log("error:"+error);
          ElMessage.error("服务器出错，请联系管理员！")
        }
      }else{
        console.log("验证失败")
      }
    })
  }


</script>
```

用户登录SpringSecurity查库实现

security包下新建MyUserDetailServiceImpl

```
@Service
public class MyUserDetailServiceImpl implements UserDetailsService {

    @Autowired
    SysUserService sysUserService;

    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        SysUser sysUser = sysUserService.getByUsername(username);
        if(sysUser==null){
            throw new UsernameNotFoundException("用户名或者密码错误！");
        }else if("1".equals(sysUser.getStatus())){
            throw new UserCountLockException("该用户账号已被封禁，具体联系管理员！");
        }
        return new
User(sysUser.getUsername(),sysUser.getPassword(),getUserAuthority(sysUser.getId(
)));
    }

    private List<GrantedAuthority> getUserAuthority(Long id) {
        return new ArrayList<>();
    }
}
```

getByUsername实现：

```
@Service
public class SysUserServiceImpl extends ServiceImpl<SysUserMapper, SysUser>
    implements SysUserService{


    @Override
    public SysUser getByUsername(String username) {
        return getOne(new QueryWrapper<SysUser>().eq("username",username));
    }
}
```

SecurityConfig配置类里面配置下MyUserDetailServiceImpl

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private LoginSuccessHandler loginSuccessHandler;

    @Autowired
    private LoginFailureHandler loginFailureHandler;

    @Autowired
    private MyUserDetailServiceImpl myUserDetailService;

    @Bean
```

```
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(myUserDetailService);
    }
}
```

配置下默认加密bean

```
@Bean
BCryptPasswordEncoder bCryptPasswordEncoder(){
    return new BCryptPasswordEncoder();
}
```

自定义异常UserCountLockException

```
public class UserCountLockException extends AuthenticationException {

    public UserCountLockException(String msg, Throwable t) {
        super(msg, t);
    }

    public UserCountLockException(String msg) {
        super(msg);
    }
}
```

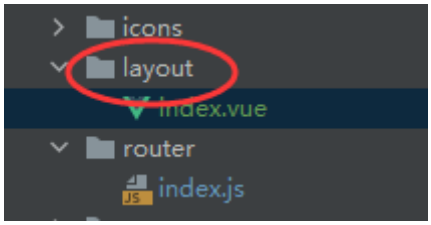
全局异常GlobalExceptionHandler：

```
@Slf4j
@RestControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler(value = RuntimeException.class)
    public R handler(RuntimeException e){
        log.error("运行时异常：----------------{}", e.getMessage());
        System.out.println("运行时异常：");
        return R.error(e.getMessage());
    }

}
```

实现JWT认证过滤器

router配置，加下首页路由

```
{
  path: '/',
  name: '首页',
  component: () => import('../layout')
},
```



测试按钮

```
<template>
<el-button type="danger" @click="testHandler">测试接口</el-button>
</template>

<script setup>
import requestUtil from '@/util/request'
```

```
const testHandler=async ()=>{
  let result=await requestUtil.get("test/user/list");

}
</script>

<style scoped>

</style>
```

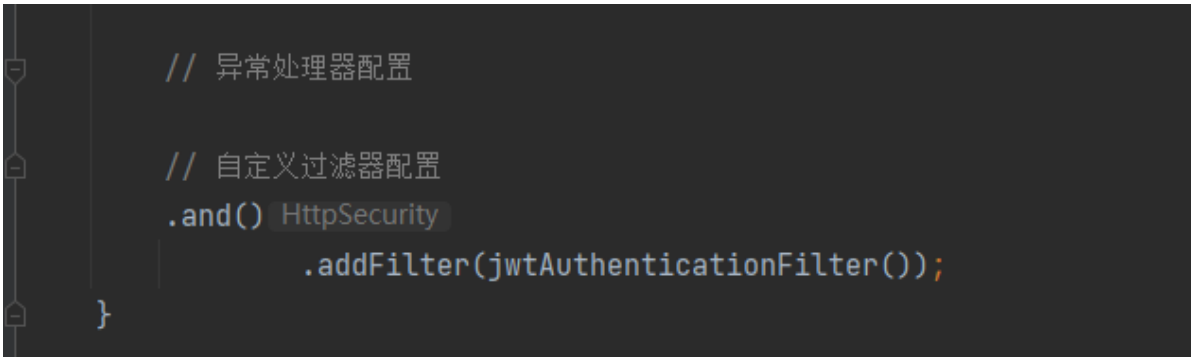
由于做了前后端分离配置，通过jwt生成token，所以我们要搞一个jwt自定义认证过滤器，来实现jwt token认证；

```
/**
 * jwt认证自定义过滤器
 * @author java1234_小锋 （公众号：java1234）
 * @site www.java1234.vip
 * @company 南通小锋网络科技有限公司
 */
public class JwtAuthenticationFilter extends BasicAuthenticationFilter {

    @Autowired
    private SysUserService sysUserService;

    @Autowired
    private MyUserDetailServiceImpl myUserDetailService;

    private static final String URL_WHITELIST[] ={
            "/login",
            "/logout",
            "/captcha",
            "/password",
            "/image/**"
    } ;

    public JwtAuthenticationFilter(AuthenticationManager authenticationManager)
{
        super(authenticationManager);
    }

    @Override
    protected void doFilterInternal(HttpServletRequest request,
HttpServletResponse response, FilterChain chain) throws IOException,
ServletException {
        String token=request.getHeader("token");
        System.out.println("请求url:"+request.getRequestURI());
        // 如果token是空或者url在白名单里 则放行 让后面的springsecurity认证过滤器去认证
        if(StringUtil.isEmpty(token)|| new ArrayList<String>
(Arrays.asList(URL_WHITELIST)).contains(request.getRequestURI())){
            chain.doFilter(request,response);
            return;
        }
        CheckResult checkResult = JwtUtils.validateJWT(token);
        if(!checkResult.isSuccess()){
```

```
            switch (checkResult.getErrCode()){
                case JwtConstant.JWT_ERRCODE_NULL: throw new JwtException("Token
不存在");
                case JwtConstant.JWT_ERRCODE_FAIL: throw new JwtException("Token
验证不通过");
                case JwtConstant.JWT_ERRCODE_EXPIRE: throw new
JwtException("Token过期");
            }
        }

        Claims claims=JwtUtils.parseJWT(token);
        String username=claims.getSubject();
        SysUser sysUser = sysUserService.getByUsername(username);
        UsernamePasswordAuthenticationToken
usernamePasswordAuthenticationToken=new
UsernamePasswordAuthenticationToken(username,null,myUserDetailService.getUserAut
hority(sysUser.getId()));

 SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthentica
tionToken);
        chain.doFilter(request,response);
    }
}
```

SecurityConfig配置

```
@Bean
JwtAuthenticationFilter jwtAuthenticationFilter() throws Exception {
    JwtAuthenticationFilter jwtAuthenticationFilter=new
JwtAuthenticationFilter(authenticationManager());
    return jwtAuthenticationFilter;
}
```



实现JWT认证异常处理器

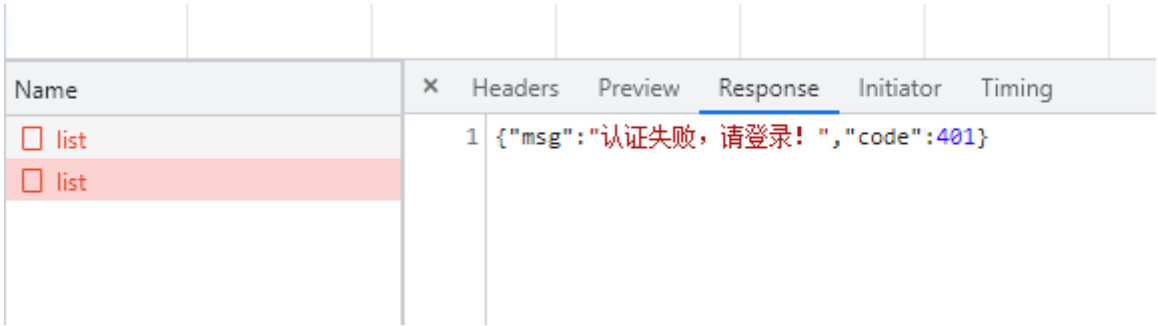
新建JwtAuthenticationEntryPoint

```
/**
 * jwt认证异常处理
 * @author java1234_小锋 （公众号：java1234）
 * @site www.java1234.vip
```

```
 * @company 南通小锋网络科技有限公司
 */
@Component
public class JwtAuthenticationEntryPoint implements AuthenticationEntryPoint {
    @Override
    public void commence(HttpServletRequest httpServletRequest,
HttpServletResponse httpServletResponse, AuthenticationException e) throws
IOException, ServletException {
        httpServletResponse.setContentType("application/json;charset=UTF-8");
        ServletOutputStream outputStream =
httpServletResponse.getOutputStream();


 outputStream.write(JSONUtil.toJsonStr(R.error(HttpServletResponse.SC_UNAUTHORIZ
ED,"认证失败，请登录！")).getBytes());
        outputStream.flush();
        outputStream.close();
    }
}
```

```
@Autowired
private JwtAuthenticationEntryPoint jwtAuthenticationEntryPoint;
```

```
    // 异常处理器配置
            .and() HttpSecurity
            .exceptionHandling() ExceptionHandlingConfigurer<HttpSecurity>
            .authenticationEntryPoint(jwtAuthenticationEntryPoint)
```



实现自定义logout处理

默认logout请求实现是有状态的，返回到login请求页面；我们现在是前后端分离处理，所以需要自定义实现logout

新建JwtLogoutSuccessHandler

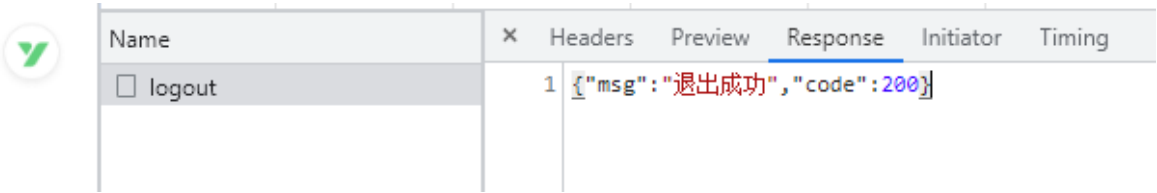
```
/**
 * 自定义Logout处理
 * @author java1234_小锋 （公众号：java1234）
```

```
 * @site www.java1234.vip
 * @company 南通小锋网络科技有限公司
 */
@Component
public class JwtLogoutSuccessHandler implements LogoutSuccessHandler {
    @Override
    public void onLogoutSuccess(HttpServletRequest httpServletRequest,
HttpServletResponse httpServletResponse, Authentication authentication) throws
IOException, ServletException {
        httpServletResponse.setContentType("application/json;charset=UTF-8");
        ServletOutputStream outputStream =
httpServletResponse.getOutputStream();

        outputStream.write(JSONUtil.toJsonStr(R.ok("退出成功")).getBytes("UTF-
8"));
        outputStream.flush();
        outputStream.close();
    }
}
```

SecurityConfig配置

```
@Autowired
private JwtLogoutSuccessHandler jwtLogoutSuccessHandler;
```

```
// 登出配置
.and() HttpSecurity
        .logout() LogoutConfigurer<HttpSecurity>
        .logoutSuccessHandler(jwtLogoutSuccessHandler)
```



获取用户角色权限信息实现

springsecurity鉴权需要获取用户的角色权限系统，包括前端也需要这些信息；

首先我们新建角色表sys_role，菜单权限表sys_menu，用户角色关联表sys_user_role，角色菜单权限关联表sys_role_menu

角色表sys_role


```
CREATE TABLE `sys_role` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '角色主键ID',
  `name` varchar(30) DEFAULT NULL COMMENT '角色名称',
  `code` varchar(100) DEFAULT NULL COMMENT '角色权限字符串',
  `create_time` datetime DEFAULT NULL COMMENT '创建时间',
  `update_time` datetime DEFAULT NULL COMMENT '更新时间',
  `remark` varchar(500) DEFAULT NULL COMMENT '备注',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=22 DEFAULT CHARSET=utf8;

/*Data for the table `sys_role` */

insert  into `sys_role`(`id`,`name`,`code`,`create_time`,`update_time`,`remark`)
values (1,'超级管理员','admin','2022-07-04 14:40:44','2022-07-04 14:40:47','拥有系统
最高权限'),(2,'普通角色','common','2022-07-04 14:41:56','2022-07-04 14:41:58','普通
角色'),(3,'测试角色','test','2022-07-04 14:42:24','2022-07-04 14:42:27','测试角色'),
(4,'2',NULL,NULL,NULL,NULL),(5,'3',NULL,NULL,NULL,NULL),
(6,'4',NULL,NULL,NULL,NULL),(7,'5',NULL,NULL,NULL,NULL),
(14,'6',NULL,NULL,NULL,NULL),(16,'8',NULL,NULL,NULL,NULL),
(17,'0',NULL,NULL,NULL,NULL),(19,'测2','cc2','2022-08-13 21:06:21','2022-08-13
13:06:27','eewew2'),(20,'ccc测试','test2','2022-08-29 17:10:33',NULL,'xxx'),
(21,'今天测试角色','todytest','2022-08-29 22:01:11',NULL,'ccc');
```

菜单权限表sys_menu


```
CREATE TABLE `sys_menu` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '菜单主键ID',
  `name` varchar(50) DEFAULT NULL COMMENT '菜单名称',
  `icon` varchar(100) DEFAULT '#' COMMENT '菜单图标',
  `parent_id` bigint(20) DEFAULT NULL COMMENT '父菜单ID',
  `order_num` int(11) DEFAULT '0' COMMENT '显示顺序',
  `path` varchar(200) DEFAULT '' COMMENT '路由地址',
  `component` varchar(255) DEFAULT NULL COMMENT '组件路径',
  `menu_type` char(1) DEFAULT '' COMMENT '菜单类型（M目录 C菜单 F按钮）',
  `perms` varchar(100) DEFAULT '' COMMENT '权限标识',
  `create_time` datetime DEFAULT NULL COMMENT '创建时间',
  `update_time` datetime DEFAULT NULL COMMENT '更新时间',
  `remark` varchar(500) DEFAULT NULL COMMENT '备注',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=34 DEFAULT CHARSET=utf8;

/*Data for the table `sys_menu` */
```

```
insert  into
`sys_menu`(`id`,`name`,`icon`,`parent_id`,`order_num`,`path`,`component`,`menu_t
ype`,`perms`,`create_time`,`update_time`,`remark`) values (1,'系统管
理','system',0,1,'/sys','','M','','2022-07-04 14:56:29','2022-07-04 14:56:31','系
统管理目录'),(2,'业务管理','monitor',0,2,'/bsns','','M','','2022-07-04
14:59:43','2022-07-04 14:59:45','业务管理目录'),(3,'用户管
理','user',1,1,'/sys/user','sys/user/index','C','system:user:list','2022-07-04
15:20:51','2022-07-04 15:20:53','用户管理菜单'),(4,'角色管
理','peoples',1,2,'/sys/role','sys/role/index','C','system:role:list','2022-07-04
15:23:35','2022-07-04 15:23:39','角色管理菜单'),(5,'菜单管理','tree-
table',1,3,'/sys/menu','sys/menu/index','C','system:menu:list','2022-07-04
15:23:41','2022-07-04 15:23:43','菜单管理菜单'),(6,'部门管
理','tree',2,1,'/bsns/department','bsns/Department','C','','2022-07-04
15:24:40','2022-07-04 15:24:44','部门管理菜单'),(7,'岗位管
理','post',2,2,'/bsns/post','bsns/Post','C','','2022-07-04 15:24:42','2022-07-04
15:24:46','岗位管理菜单'),(8,'用户新增','#',3,2,'','','F','system:user:add','2022-
07-04 15:24:42','2022-07-04 15:24:46','添加用户按钮'),(9,'用户修
改','#',3,3,'','','F','system:user:edit','2022-07-04 15:24:42','2022-07-04
15:24:46','修改用户按钮'),(10,'用户删
除','#',3,4,'','','F','system:user:delete','2022-07-04 15:24:42','2022-07-04
15:24:46','删除用户按钮'),(11,'分配角色','#',3,5,'','','F','system:user:role','2022-
07-04 15:24:42','2022-07-04 15:24:46','分配角色按钮'),(12,'重置密
码','#',3,6,'','','F','system:user:resetPwd','2022-07-04 15:24:42','2022-07-04
15:24:46','重置密码按钮'),(13,'角色新增','#',4,2,'','','F','system:role:add','2022-
07-04 15:24:42','2022-07-04 15:24:46','添加用户按钮'),(14,'角色修
改','#',4,3,'','','F','system:role:edit','2022-07-04 15:24:42','2022-07-04
15:24:46','修改用户按钮'),(15,'角色删
除','#',4,4,'',NULL,'F','system:role:delete','2022-07-04 15:24:42','2022-07-04
15:24:46','删除用户按钮'),(16,'分配权限','#',4,5,'','','F','system:role:menu','2022-
07-04 15:24:42','2022-07-04 15:24:46','分配权限按钮'),(17,'菜单新
增','#',5,2,'',NULL,'F','system:menu:add','2022-07-04 15:24:42','2022-07-04
15:24:46','添加菜单按钮'),(18,'菜单修
改','#',5,3,'',NULL,'F','system:menu:edit','2022-07-04 15:24:42','2022-07-04
15:24:46','修改菜单按钮'),(19,'菜单删
除','#',5,4,'',NULL,'F','system:menu:delete','2022-07-04 15:24:42','2022-07-04
15:24:46','删除菜单按钮'),(20,'用户查
询','#',3,1,'',NULL,'F','system:user:query','2022-07-04 15:24:42','2022-07-04
15:24:46','用户查询按钮'),(21,'角色查
询','#',4,1,'',NULL,'F','system:role:query','2022-07-04 15:24:42','2022-07-04
15:24:46','角色查询按钮'),(22,'菜单查
询','#',5,1,'',NULL,'F','system:menu:query','2022-07-04 15:24:42','2022-07-04
15:24:46','菜单查询按钮'),(33,'测速22','122',3,3,'','34','M','33','2022-08-19
03:11:20','2022-08-18 19:11:33',NULL);
```

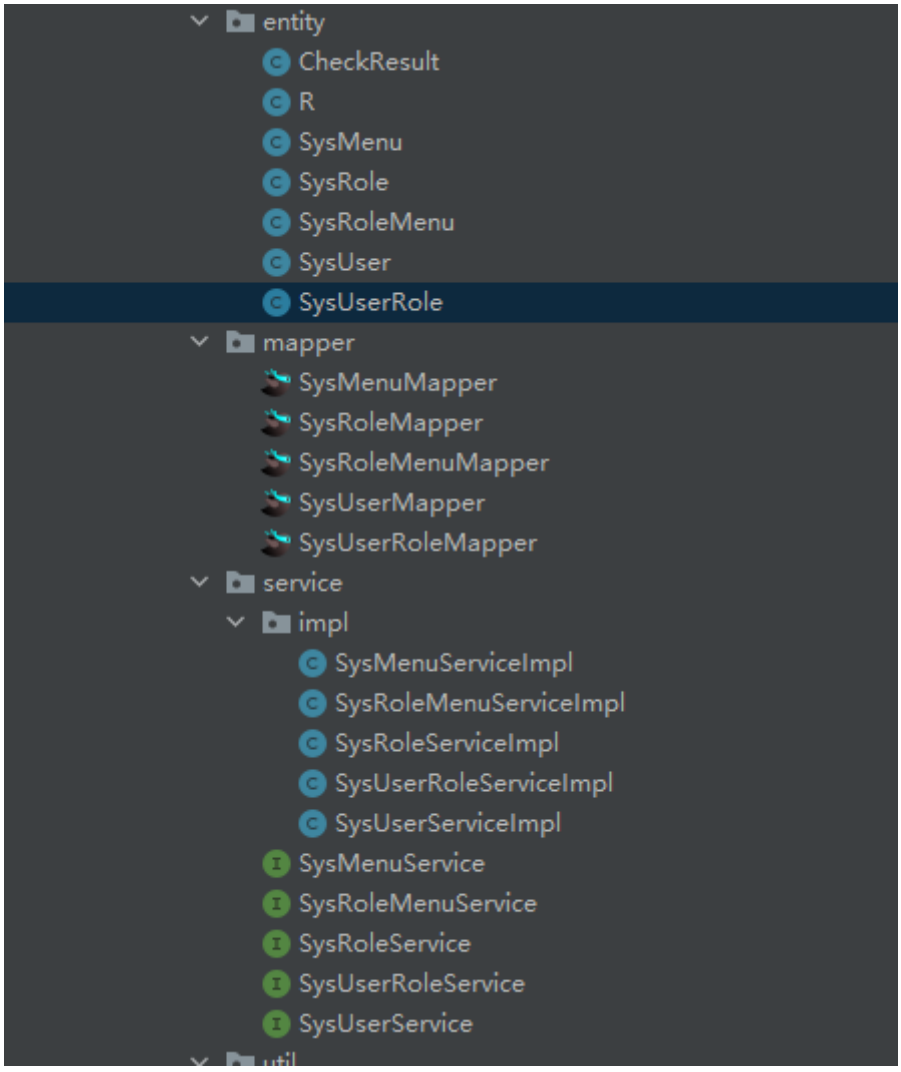
用户角色关联表sys_user_role


```
CREATE TABLE `sys_user_role` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '用户角色主键ID',
  `user_id` bigint(20) DEFAULT NULL COMMENT '用户ID',
  `role_id` bigint(20) DEFAULT NULL COMMENT '角色ID',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=23 DEFAULT CHARSET=utf8;

/*Data for the table `sys_user_role` */

insert  into `sys_user_role`(`id`,`user_id`,`role_id`) values (1,1,1),(2,2,2),
(4,1,2),(6,3,3),(7,3,2),(9,4,3),(10,5,3),(11,15,3),(16,28,2),(17,28,3),
(20,29,20),(21,30,17),(22,30,21);
```

角色菜单权限关联表sys_role_menu

```sql
CREATE TABLE `sys_role_menu` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '角色菜单主键ID',
  `role_id` bigint(20) DEFAULT NULL COMMENT '角色ID',
  `menu_id` bigint(20) DEFAULT NULL COMMENT '菜单ID',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=239 DEFAULT CHARSET=utf8;

/*Data for the table `sys_role_menu` */

insert  into `sys_role_menu`(`id`,`role_id`,`menu_id`) values (8,2,1),(9,2,2),
(10,2,3),(11,2,4),(12,2,5),(13,2,6),(14,2,7),(15,3,2),(16,3,6),(17,3,7),
(21,7,1),(22,7,2),(23,7,6),(24,7,7),(25,6,1),(26,6,3),(27,6,9),(28,6,10),
(29,19,1),(30,19,3),(31,19,2),(32,19,6),(33,1,1),(34,1,3),(35,1,20),(36,1,8),
(37,1,9),(38,1,10),(39,1,11),(40,1,12),(41,1,4),(42,1,21),(43,1,13),(44,1,14),
(45,1,15),(46,1,16),(47,1,23),(48,1,5),(49,1,22),(50,1,17),(51,1,18),(52,1,19),
(53,1,2),(54,1,6),(55,1,7),(208,20,1),(209,20,3),(210,20,20),(211,20,8),
(212,20,9),(213,20,33),(214,20,10),(215,20,11),(216,20,4),(217,20,21),
(218,20,13),(219,20,5),(220,20,22),(221,20,17),(222,20,18),(223,20,2),
(224,20,6),(225,20,7),(232,21,1),(233,21,9),(234,21,4),(235,21,21),(236,21,2),
(237,21,6),(238,21,7);
```

我们通过MybatisX生成代码：



因为每个实体都有几个通用属性，id，createTime，updateTime，remark，所以我们搞一个通用基础实体类，让其他类继承下；

方便维护；

新建BaseEntity

```
/**
 * 公共基础实体类
 * @author java1234_小锋 （公众号：java1234）
 * @site www.java1234.vip
 * @company 南通小锋网络科技有限公司
 */
@Data
public class BaseEntity implements Serializable {

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

    /**
     * 创建日期
     */
    @JsonSerialize(using=CustomDateTimeSerializer.class)
    @JsonFormat(pattern ="yyyy-MM-dd HH:mm:ss")
    private Date createTime;

    /**
     * 更新日期
     */
    @JsonSerialize(using=CustomDateTimeSerializer.class)
    @JsonFormat(pattern ="yyyy-MM-dd HH:mm:ss")
    private Date updateTime;

    /**
     * 备注
     */
    @TableField(value = "remark")
    private String remark;

}
```

`MyUserDetailServiceImpl``的getUserAuthority`实现：

```
public List<GrantedAuthority> getUserAuthority(Long userId) {
    // 格式
ROLE_admin,ROLE_common,system:user:resetPwd,system:role:delete,system:user:list,
system:menu:query,system:menu:list,system:menu:add,system:user:delete,system:rol
e:list,system:role:menu,system:user:edit,system:user:query,system:role:edit,syst
em:user:add,system:user:role,system:menu:delete,system:role:add,system:role:quer
y,system:menu:edit
    String authority=sysUserService.getUserAuthorityInfo(userId);
    System.out.println("authority="+authority);
    return AuthorityUtils.commaSeparatedStringToAuthorityList(authority);
}
```

`SysUserServiceImpl``的实现方法getUserAuthorityInfo`

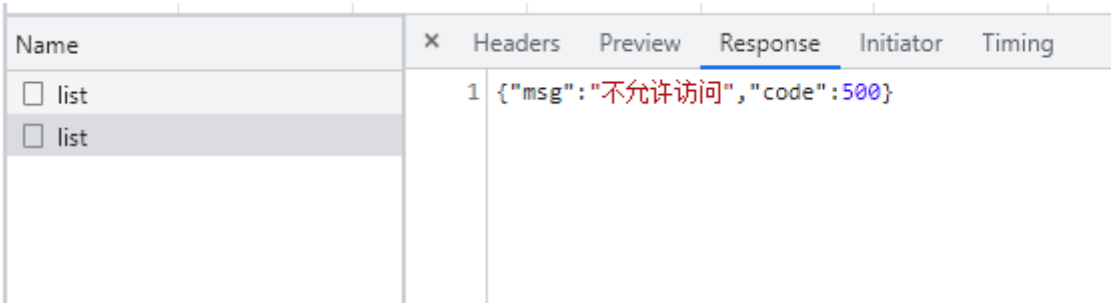
```
public String getUserAuthorityInfo(Long userId) {
    StringBuffer authority=new StringBuffer();
    // 根据用户id获取所有的角色
    List<SysRole> roleList = sysRoleMapper.selectList(new QueryWrapper<SysRole>
().inSql("id", "SELECT role_id FROM sys_user_role WHERE user_id=" + userId));
    if(roleList.size()>0){
        String roleCodeStrs = roleList.stream().map(r -> "ROLE_" +
r.getCode()).collect(Collectors.joining(","));
        authority.append(roleCodeStrs);
    }
    // 遍历角色，获取所有菜单权限
    Set<String> menuCodeSet=new HashSet<String>();
    for(SysRole sysRole:roleList){
        List<SysMenu> sysMenuList = sysMenuMapper.selectList(new
QueryWrapper<SysMenu>().inSql("id", "SELECT menu_id FROM sys_role_menu WHERE
role_id=" + sysRole.getId()));
        for(SysMenu sysMenu:sysMenuList){
            String perms=sysMenu.getPerms();
            if(StringUtil.isNotEmpty(perms)){
                menuCodeSet.add(perms);
            }
        }
    }
    if(menuCodeSet.size()>0){
        authority.append(",");
        String menuCodeStrs =
menuCodeSet.stream().collect(Collectors.joining(","));
        authority.append(menuCodeStrs);
    }
    System.out.println("authority:"+authority.toString());
    return authority.toString();
}
```

参考代码：根绝用户id查询所有用户角色信息

SELECT * FROM sys_role WHERE id IN (SELECT role_id FROM sys_user_role WHERE user_id=1)

接口权限测试：

```
    @GetMapping("/user/list")
    // @PreAuthorize("hasAnyAuthority('system2:user:list')")
    @PreAuthorize("hasRole('ROLE2_common')")
    public R userList(@RequestHeader(required = false) String token){
        if(StringUtil.isNotEmpty(token)){
            Map<String,Object> resutMap=new HashMap<>();
            List<SysUser> userList = sysUserService.list();
            resutMap.put("userList",userList);
            return R.ok(resutMap);
        }else{
            return R.error(401,"没有权限访问");
        }
    }
```



记住密码功能实现

记住密码，我们通过cookie来实现，先安装依赖 'js-cookie'

存储用户密码，为了安全需要加密，获取密码解密。所以我们安装依赖'jsencrypt'

util下新建jsencrypt.js

```
import JSEncrypt from 'jsencrypt/bin/jsencrypt.min'

// 密钥对生成 http://web.chacuo.net/netrsakeypair

const publicKey =
'MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAKoR8mXOrGKLqzcWmOzbfj64K8ZIgOdH\n' +
  'nzkXSOVOZbFu/TJhZ7rFAN+eaGkl3C4buccQd/EjEsj9ir7ijT7h96MCAwEAAQ=='

const privateKey =
'MIIBVAIBADANBgkqhkiG9w0BAQEFAASCAT4wggE6AgEAAkEAqhHyZfSsYourNxaY\n' +
  '7Nt+PrgrxkiA5OefORdI5U5lsW79MmFnusUA355oaSXcLhu5xxB38SMSyP2KvuKN\n' +
  'PuH3owIDAQABAkAfoiLyL+Z4lf4Myxk6xUDgLaWGximj2OCUf+5BKKnlrK+Ed8gA\n' +
  'kMOHqoTt2UZwA5E2MzS4EI2gjfQhz5X28uqxAiEA3wNFxfrCZlSZHbOgn2zDpWow\n' +
  'cSxQAgiCstxGUoOqlW8CIQDDOerGKH5OmCJ4Z21v+F25WaHYPxCFMvwxpcw99Ecv\n' +
  'DQIgIdhDTIqD2jfYjPTY8Jj3EDGPbH2HHuffvflECt3Ek6OCIQCFRlCkHpi7hthh\n' +
  'YhovyloRYsM+IS9h/OBzlEAuOOktMQIgSPT3aFAgJYwKpqRYKlLDVcflZFCKY7u3\n' +
  'UP8iWi1QwOY='

// 加密
export function encrypt(txt) {
  const encryptor = new JSEncrypt()
  encryptor.setPublicKey(publicKey) // 设置公钥
  return encryptor.encrypt(txt) // 对数据进行加密
```

```
}

// 解密
export function decrypt(txt) {
  const encryptor = new JSEncrypt()
  encryptor.setPrivateKey(privateKey) // 设置私钥
  return encryptor.decrypt(txt) // 对数据进行解密
}
```

```
</el-form-item>

<el-checkbox v-model="loginForm.rememberMe" style="margin:0px 0px 25px 0px;">记住密码</el-checkbox>
<el-form-item style="width:100%;">
  <el-button
      size="large"
      type="primary"
      style="width:100%;"
      @click.prevent="handleLogin"
  >
```

```
import router from '@/router'
import Cookies from "js-cookie";

const loginRef=ref(null);

const loginForm=ref({
  username:"",
  password:"",
  rememberMe:false
})
```

```
import Cookies from "js-cookie";
import { encrypt, decrypt } from "@/util/jsencrypt";
```

```
// 勾选了需要记住密码设置在 cookie 中设置记住用户名和密码
if (loginForm.value.rememberMe) {
  Cookies.set("username", loginForm.value.username, { expires: 30 });
  Cookies.set("password", encrypt(loginForm.value.password), { expires: 30 });
  Cookies.set("rememberMe", loginForm.value.rememberMe, { expires: 30 });
} else {
  // 否则移除
  Cookies.remove("username");
  Cookies.remove("password");
  Cookies.remove("rememberMe");
}
```

```
function getCookie() {
  const username = Cookies.get("username");
  const password = Cookies.get("password");
  const rememberMe = Cookies.get("rememberMe");
  loginForm.value = {
    username: username === undefined ? loginForm.value.username : username,
    password: password === undefined ? loginForm.value.password :
decrypt(password),
    rememberMe: rememberMe === undefined ? false : Boolean(rememberMe)
  };
}

getCookie();
```

主页面功能实现

主页面布局实现

```
<template>
 <div class="app-wrapper">
    <el-container>
      <el-aside width="200px" class="sidebar-container"><Menu/></el-aside>
      <el-container>
        <el-header><Header/></el-header>
        <el-main><Tabs/></el-main>
        <el-footer><Footer/></el-footer>
      </el-container>
    </el-container>
  </div>
</template>

<script setup>
import Menu from '@/layout/menu'
import Header from '@/layout/header'
import Footer from '@/layout/footer'
import Tabs from '@/layout/tabs'

</script>

<style lang="scss" scoped>

.app-wrapper {
  position: relative;
  width: 100%;
  height: 100%;
}

.sidebar-container {
  background-color: #2d3a4b;
  height: 100%;
}

.el-container{
```

```
  height:100%
}

.el-header{
  padding-left: 0px;
  padding-right: 0px;
}

:deep(ul.el-menu){
  border-right-width: 0px
}

</style>
```

footer index.vue

```
<template>
  <div class="footer">
    Copyright © 2012-2022 Java知识分享网 版权所有  <a
href="http://www.java1234.vip" target="_blank">www.java1234.vip</a>
  </div>
</template>

<script setup>

</script>

<style lang="scss" scoped>
.footer{
  padding: 20px;
  display: flex;
  align-items: center;
}
</style>
```

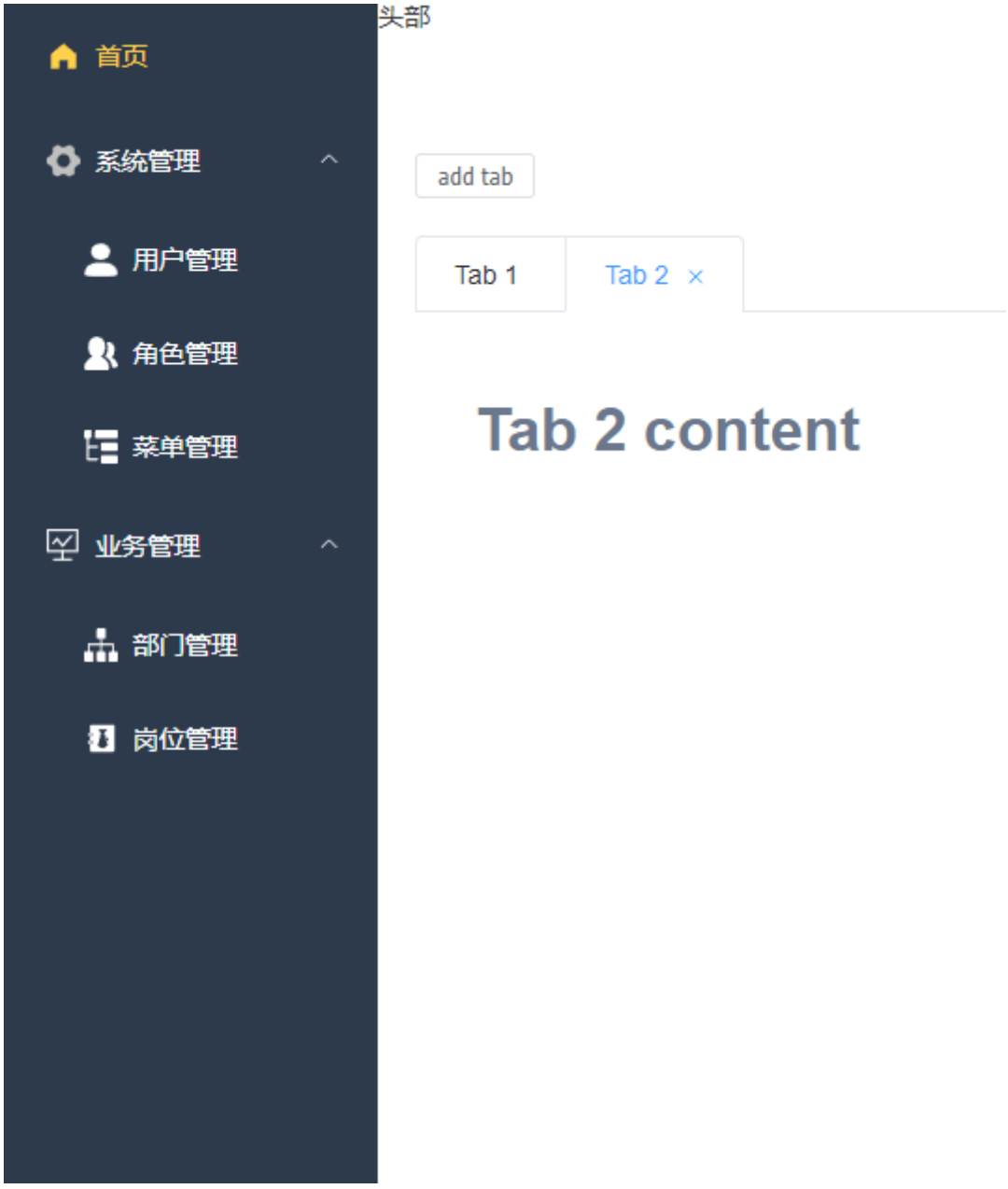
tabs index.vue

```
<template>
  <div style="margin-bottom: 20px">
    <el-button size="small" @click="addTab(editableTabsValue)">
      add tab
    </el-button>
  </div>
  <el-tabs
    v-model="editableTabsValue"
    type="card"
    class="demo-tabs"
    closable
    @tab-remove="removeTab"
  >
    <el-tab-pane
      v-for="item in editableTabs"
      :key="item.name"
```

```
      :label="item.title"
      :name="item.name"
    >
      {{ item.content }}
    </el-tab-pane>
  </el-tabs>
</template>
<script setup>
import { ref } from 'vue'

let tabIndex = 2
const editableTabsValue = ref('2')
const editableTabs = ref([
  {
    title: 'Tab 1',
    name: '1',
    content: 'Tab 1 content',
  },
  {
    title: 'Tab 2',
    name: '2',
    content: 'Tab 2 content',
  },
])

const addTab = (targetName) => {
  const newTabName = `${++tabIndex}`
  editableTabs.value.push({
    title: 'New Tab',
    name: newTabName,
    content: 'New Tab content',
  })
  editableTabsValue.value = newTabName
}
const removeTab = (targetName) => {
  const tabs = editableTabs.value
  let activeName = editableTabsValue.value
  if (activeName === targetName) {
    tabs.forEach((tab, index) => {
      if (tab.name === targetName) {
        const nextTab = tabs[index + 1] || tabs[index - 1]
        if (nextTab) {
          activeName = nextTab.name
        }
      }
    })
  }

  editableTabsValue.value = activeName
  editableTabs.value = tabs.filter((tab) => tab.name !== targetName)
}
</script>
<style>
.demo-tabs > .el-tabs__content {
  padding: 32px;
  color: #6b778c;
  font-size: 32px;
  font-weight: 600;
```

```
}
</style>
```

左侧动态权限菜单实现



LoginSuccessHandler 修改：

```
@Component
public class LoginSuccessHandler implements AuthenticationSuccessHandler {

    @Autowired
    private SysUserService sysUserService;

    @Autowired
    private SysRoleService sysRoleService;

    @Autowired
    private SysMenuService sysMenuService;
```

```
    @Override
    public void onAuthenticationSuccess(HttpServletRequest httpServletRequest,
HttpServletResponse httpServletResponse, Authentication authentication) throws
IOException, ServletException {
        httpServletResponse.setContentType("application/json;charset=UTF-8");
        ServletOutputStream outputStream =
httpServletResponse.getOutputStream();

        String username=authentication.getName(); // 获取用户名
        // 更新用户最后登录记录
        sysUserService.update(new UpdateWrapper<SysUser>().set("login_date",new
Date()).eq("username",username));

        SysUser currentUser = sysUserService.getByUsername(username);

        String token = JwtUtils.genJwtToken(username);

        // 根据用户id获取所有的角色
        List<SysRole> roleList = sysRoleService.list(new QueryWrapper<SysRole>
().inSql("id", "SELECT role_id FROM sys_user_role WHERE user_id=" +
currentUser.getId()));

        // 遍历角色，获取所有菜单权限
        Set<SysMenu> menuSet=new HashSet<SysMenu>();
        for(SysRole sysRole:roleList){
            List<SysMenu> sysMenuList = sysMenuService.list(new
QueryWrapper<SysMenu>().inSql("id", "SELECT menu_id FROM sys_role_menu WHERE
role_id=" + sysRole.getId()));
            for(SysMenu sysMenu:sysMenuList){
                menuSet.add(sysMenu);
            }
        }

        List<SysMenu> sysMenuList=new ArrayList<>(menuSet);

        // 排序
        sysMenuList.sort(Comparator.comparing(SysMenu::getOrderNum));

        List<SysMenu> menuList=sysMenuService.buildTreeMenu(sysMenuList);


        outputStream.write(JSONUtil.toJsonStr(R.ok("登录成
功").put("authorization",token).put("currentUser",currentUser).put("menuList",men
uList)).getBytes());
        outputStream.flush();
        outputStream.close();

    }
}
```

SysMenuServiceImpl buildTreeMenu

```
@Override
```

```
public List<SysMenu> buildTreeMenu(List<SysMenu> sysMenuList) {
    List<SysMenu> resultMenuList=new ArrayList<>();

    for(SysMenu sysMenu:sysMenuList){
        // 寻找子节点
        for(SysMenu e:sysMenuList){
            if(e.getParentId()==sysMenu.getId()){
                sysMenu.getChildren().add(e);
            }
        }

        // 判断父节点，添加到集合
        if(sysMenu.getParentId()==0L){
            resultMenuList.add(sysMenu);
        }
    }

    return resultMenuList;
}
```

SysMenu加一个字段

```
@TableField(exist = false)
private List<SysMenu> children=new ArrayList<>();
```

store修改：

```
export default createStore( options: {
  state: {
  },
  getters: {
    GET_TOKEN: state => {
      return sessionStorage.getItem( key: "token")
    },
    GET_MENULIST:state => {
      return JSON.parse(sessionStorage.getItem( key: "menuList"));
    }
  },
  mutations: {
    SET_TOKEN: (state, token) => {
      sessionStorage.setItem("token", token)
    },
    SET_MENULIST: (state, menuList) => {
      sessionStorage.setItem("menuList", JSON.stringify(menuList))
    }

  },
  actions: {
  },
  modules: {
  }
})
```

Login.vue

```
      Cookies.remove("password");
      Cookies.remove("rememberMe");
    }

    try{
        let result=await requestUtil.post("login?"+qs.stringify(loginForm.value))
        let data=result.data;
        if(data.code==200){
          const token = data.authorization;
          const menuList=data.menuList;
          console.log("menuList:"+menuList);
          store.commit("SET_MENULIST",menuList);
          store.commit('SET_TOKEN',token);
          router.replace("/")
        }else{
          ElMessage.error(data.msg)
        }
```

menu/index.vue

```
<template>
```

```
  <el-menu
      active-text-color="#ffd04b"
      background-color="#2d3a4b"
      class="el-menu-vertical-demo"
      text-color="#fff"
      router
      :default-active="'/index'"
  >
      <el-menu-item index="/index">
        <el-icon><home-filled /></el-icon>
        <span>首页</span>
      </el-menu-item>

       <el-sub-menu :index="menu.path" v-for="menu in menuList">
            <template #title>
              <el-icon><svg-icon :icon="menu.icon"/></el-icon>
              <span>{{menu.name}}</span>
            </template>

            <el-menu-item :index="item.path" v-for="item in menu.children">
              <el-icon><svg-icon :icon="item.icon"/></el-icon>
              <span>{{item.name}}</span>
            </el-menu-item>

       </el-sub-menu>
  </el-menu>
</template>

<script setup>
import
{HomeFilled,User,Tickets,Goods,DocumentAdd,Management,Setting,Edit,SwitchButton,
Promotion} from '@element-plus/icons-vue'
import {ref} from 'vue'
import store from '@/store'

const menuList=ref(store.getters.GET_MENULIST)
console.log("menuList2="+menuList)
</script>

<style lang="scss" scoped>

</style>
```

右上角用户头像显示实现

这里有个用户头像，虚拟路径映射配置下：

WebAppConfigurer类：

```
@Override
public void addResourceHandlers(ResourceHandlerRegistry registry) {

 registry.addResourceHandler("/image/userAvatar/**").addResourceLocations("file:
D:\\java1234-admin2\\userAvatar\\");
}
```

store/index.js

```
},
getters: {
  GET_TOKEN: state => {
    return sessionStorage.getItem( key: "token")
  },
  GET_MENULIST:state => {
    return JSON.parse(sessionStorage.getItem( key: "menuList"));
  },
  GET_USERINFO:state => {
    return JSON.parse(sessionStorage.getItem( key: "userInfo"));
  }
},
mutations: {
  SET_TOKEN: (state, token) => {
    sessionStorage.setItem("token", token)
  },
  SET_MENULIST: (state, menuList) => {
    sessionStorage.setItem("menuList", JSON.stringify(menuList))
  },
  SET_USERINFO: (state, userInfo) => {
    sessionStorage.setItem("userInfo", JSON.stringify(userInfo))
  },
```

Login.vue

```
      Cookies.remove("rememberMe");
    }

    try{
        let result=await requestUtil.post("login?"+qs.stringify(loginForm.value))
        let data=result.data;
        if(data.code==200){
          const token = data.authorization;
          const menuList=data.menuList;
          const currentUser=data.currentUser;
          console.log("menuList:"+menuList);
          store.commit("SET_MENULIST",menuList);
          store.commit('SET_TOKEN',token);
          store.commit('SET_USERINFO',currentUser)
          router.replace("/")
```

header/index.vue

```
<template>
  <div class="navbar">
    <Breadcrumb/>
    <div class="navbar-right">
      <Avatar/>
    </div>
  </div>
</template>
```

```
<script setup>
import Breadcrumb from './components/breadcrumb.vue'
import Avatar from './components/avatar.vue'
</script>

<style lang="scss" scoped>
.navbar {
  width: 100%;
  height: 60px;
  overflow: hidden;
  background-color: #F5F5F5;
  box-shadow: 0 1px 4px rgba(0, 21, 41, 0.08);
  padding: 0 16px;
  display: flex;
  align-items: center;
  box-sizing: border-box;
  position: relative;
  .navbar-right {
    flex: 1;
    display: flex;
    align-items: center;
    justify-content: flex-end;
    :deep(.navbar-item) {
      display: inline-block;
      margin-left: 18px;
      font-size: 22px;
      color: #5a5e66;
      box-sizing: border-box;
      cursor: pointer;
    }
  }
}
</style>
```

breadcrumb.vue

```
<template>
面包屑
</template>

<script setup>

</script>

<style lang="scss" scoped>

</style>
```

avatar.vue

```
<template>
 <el-dropdown>
    <span class="el-dropdown-link">
```

```
      <el-avatar shape="square" :size="40" :src="squareUrl" />
        {{ currentUser.username }}
      <el-icon class="el-icon--right">
        <arrow-down />
      </el-icon>
    </span>
    <template #dropdown>
      <el-dropdown-menu>
        <el-dropdown-item>个人中心</el-dropdown-item>
        <el-dropdown-item @click="logout">安全退出</el-dropdown-item>
      </el-dropdown-menu>
    </template>
  </el-dropdown>
</template>

<script setup>
import { ArrowDown } from '@element-plus/icons-vue'
import {ref} from 'vue'
import store from '@/store'
import requestUtil,{getServerUrl} from '@/util/request'

const currentUser=ref(store.getters.GET_USERINFO);

const squareUrl=ref(getServerUrl()+'image/userAvatar/'+currentUser.value.avatar)

const logout=async ()=>{
  let result=await requestUtil.get("/logout");

  if(result.data.code===200){
    store.dispatch('logout')
  }
}

</script>

<style lang="scss" scoped>
.el-dropdown-link {
  cursor: pointer;
  color: var(--el-color-primary);
  display: flex;
  align-items: center;
}
</style>
```

安全退出

store

路由守卫功能实现

前端如果没有登录过，也就没有token，则自动跳转到登录页面，这个就是路由守卫。

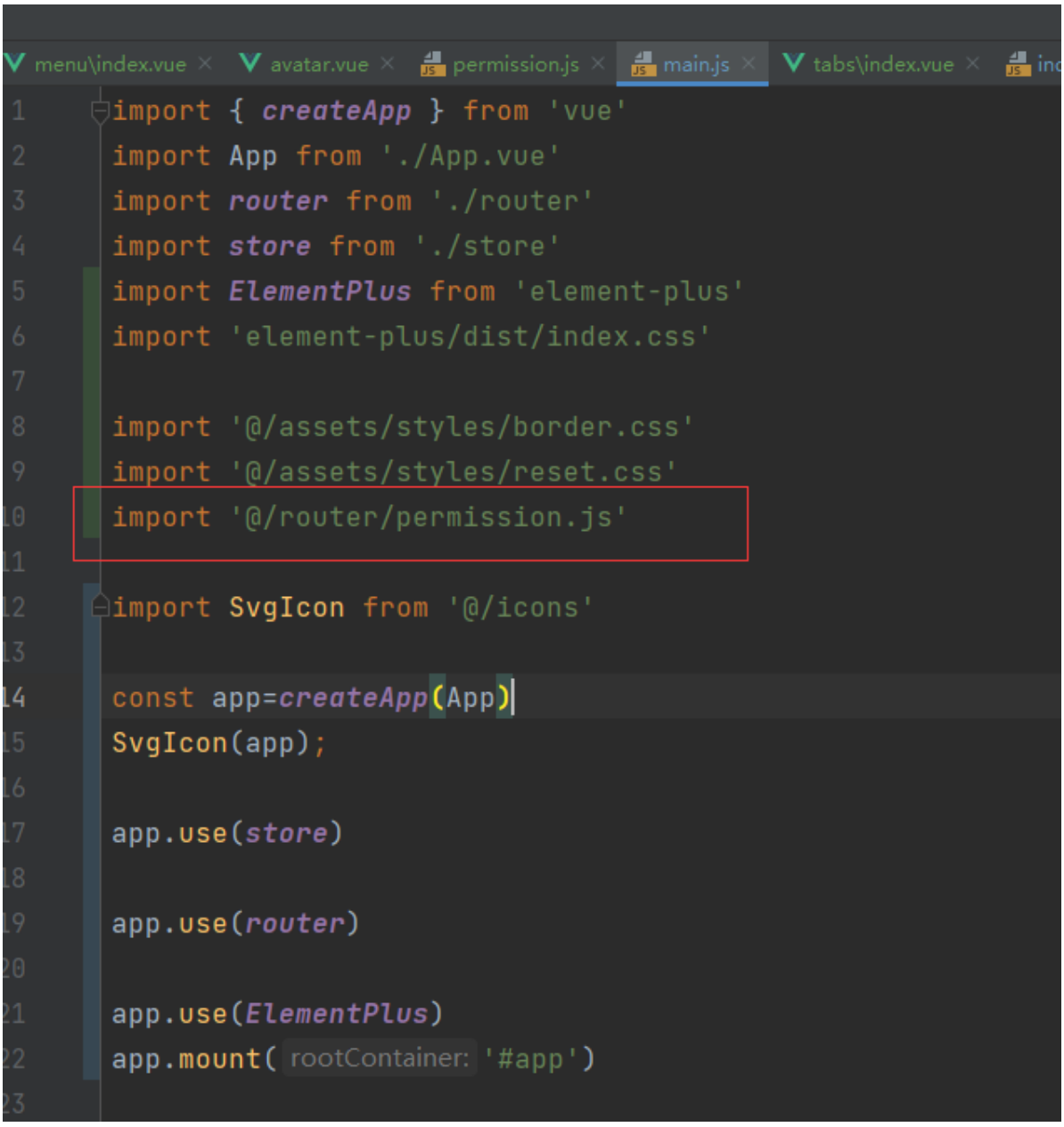
我们通过 `router.beforeEach((to, from, next)=>{})` 实现

router目录下新建permission.js

```
import router from "@/router/index"
import store from "@/store"

router.beforeEach((to,from,next)=>{
```

```
    const whiteList=['/login'] // 白名单
    let token=store.getters.GET_TOKEN;
    if(token){
        next();
    }else{
        if(whiteList.includes(to.path)){
            next();
        }else{
            next("/login");
        }
    }
})
```

main.js里面引入permission.js



动态路由实现

我们vue路由信息，需要通过后端查询的menuList，动态设置到router里面去；

layout index.vue 加下

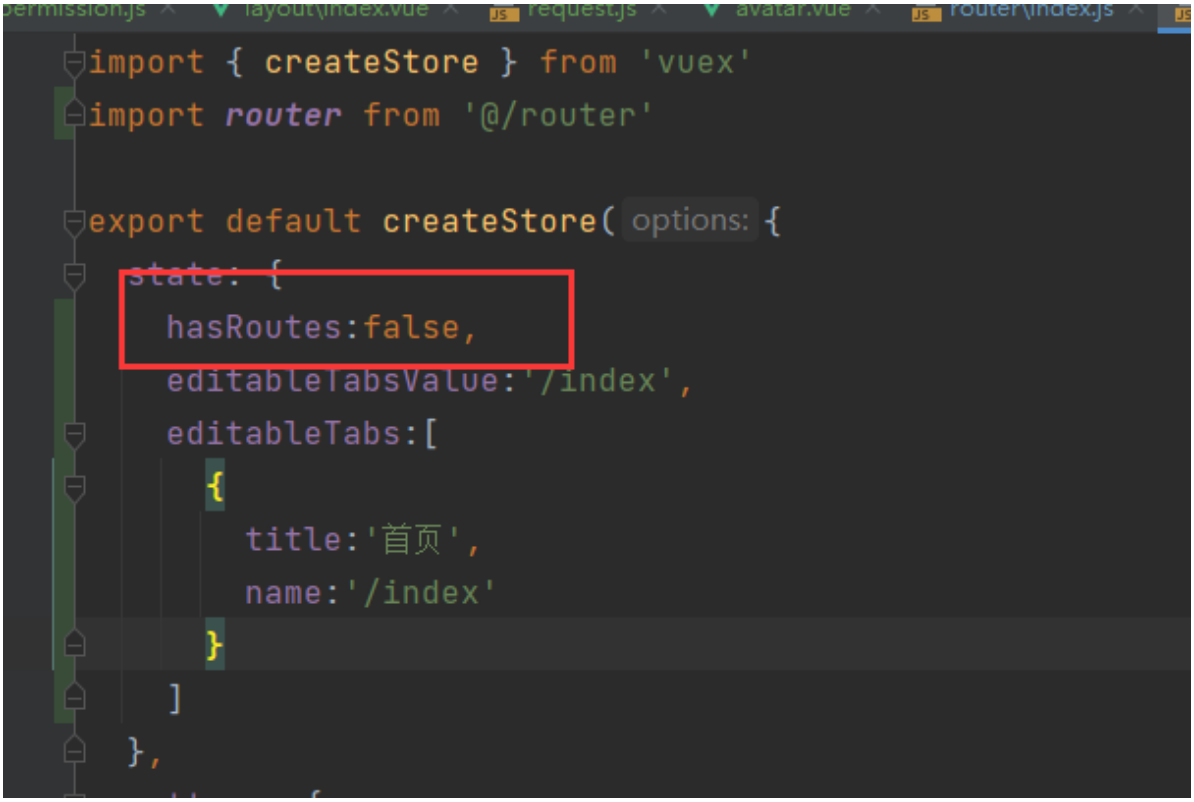
```
<template>
 <div class="app-wrapper">
    <el-container>
      <el-aside width="200px" class="sidebar-container"><Menu/></el-aside>
      <el-container>
        <el-header><Header/></el-header>
        <el-main>
          <Tabs/>
          <router-view/>
        </el-main>
        <el-footer><Footer/></el-footer>
      </el-container>
    </el-container>
 </div>
</template>
```

router要搞个children

```
const routes = [
  {
    path: '/',
    name: '首页',
    component: () => import('../layout'),
    redirect:'/index',
    children:[
      {
        path: '/index',
        name: '首页',
        component: () => import('../views/index/index')
      },{
        path: '/userCenter',
        name: '个人中心',
        component: () => import('../views/userCenter/index')
      }
```

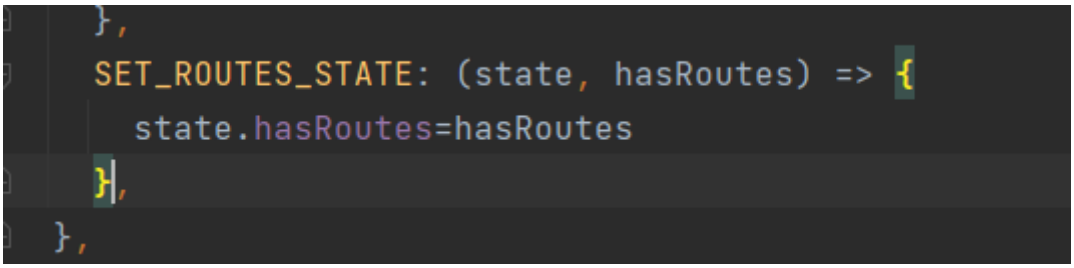
我们在路由守卫代码里面动态判断，绑定一次动态路由

我们搞一个标识变量 hasRoutes



```
import { createStore } from 'vuex'
import router from '@/router'

export default createStore( options: {
  state: {
    hasRoutes:false,
    editableTabsValue:'/index',
    editableTabs:[
      {
        title:'首页',
        name:'/index'
      }
    ]
  },
```

搞一个设置方法



```
  },
  SET_ROUTES_STATE: (state, hasRoutes) => {
    state.hasRoutes=hasRoutes
  },
},
```

permission.js

```
import router from "@/router/index"
import store from "@/store"

router.beforeEach((to,from,next)=>{
    const whiteList=['/login'] // 白名单
    let token=store.getters.GET_TOKEN;
    let hasRoutes=store.state.hasRoutes;
    let menuList=store.getters.GET_MENULIST;
    if(token){
        if(!hasRoutes){
            bindRoute(menuList);
            store.commit("SET_ROUTES_STATE",true);
        }
        next();
    }else{
        if(whiteList.includes(to.path)){
            next();
        }else{
            next("/login");
        }
    }
```

```
})

// 动态绑定路由
const bindRoute=(menuList)=>{
    console.log(menuList)
    let newRoutes=router.options.routes;
    menuList.forEach(menu=>{
        if(menu.children){
            menu.children.forEach(m=>{
                // 菜单转成路由
                let route=menuToRoute(m,menu.name);
                if(route){
                    newRoutes[0].children.push(route); // 添加到路由管理
                }
            })
        }
    })
    // 重新添加到路由
    newRoutes.forEach(route=>{
        router.addRoute(route);
    })
}

// 菜单转成路由
const menuToRoute = (menu,parentName) => {

    if (!menu.component) {
        return null
    }else{
        let route = {
            name: menu.name,
            path: menu.path,
            meta:{
                parentName:parentName
            }
        }
        route.component = () => import('@/views/' + menu.component +'.vue')

        return route
    }
}
```

安全退出代码里面，需要设置下hasRoutes状态：

```
const squareUrl=ref(getServerUrl()+'image/userAvatar/'+cu

const logout=async ()=>{
  let result=await requestUtil.get("/logout");

  if(result.data.code===200){
    store.commit("SET_ROUTES_STATE",false);
    store.dispatch('logout')
  }
}
```

动态标签页实现

store添加默认tabs数组，以及默认选中的tab值，以及添加和充值tab

```
export default createStore( options: {
  state: {
    hasRoutes:false,
    editableTabsValue:'/index',
    editableTabs:[
      {
        title:'首页',
        name:'/index'
      }
    ]
  },
```

```
  },
  ADD_TABS: (state, tab) => {
    if(state.editableTabs.findIndex(e=>e.name===tab.path)===-1){
      state.editableTabs.push({
        title:tab.name,
        name:tab.path
      })
    }
    state.editableTabsValue=tab.path;
  },
  RESET_TABS: (state) => {
    state.editableTabsValue='/index';
    state.editableTabs=[
      {
        title:'首页',
        name:'/index'
      }
    ]
  },
```

安全退出的时候，要重置下tabs

```
const logout=async ()=>{
  let result=await requestUtil.get("/logout");

  if(result.data.code===200){
    store.commit("SET_ROUTES_STATE",false);
    store.commit("RESET_TABS");
    store.dispatch('logout')
  }
}
```

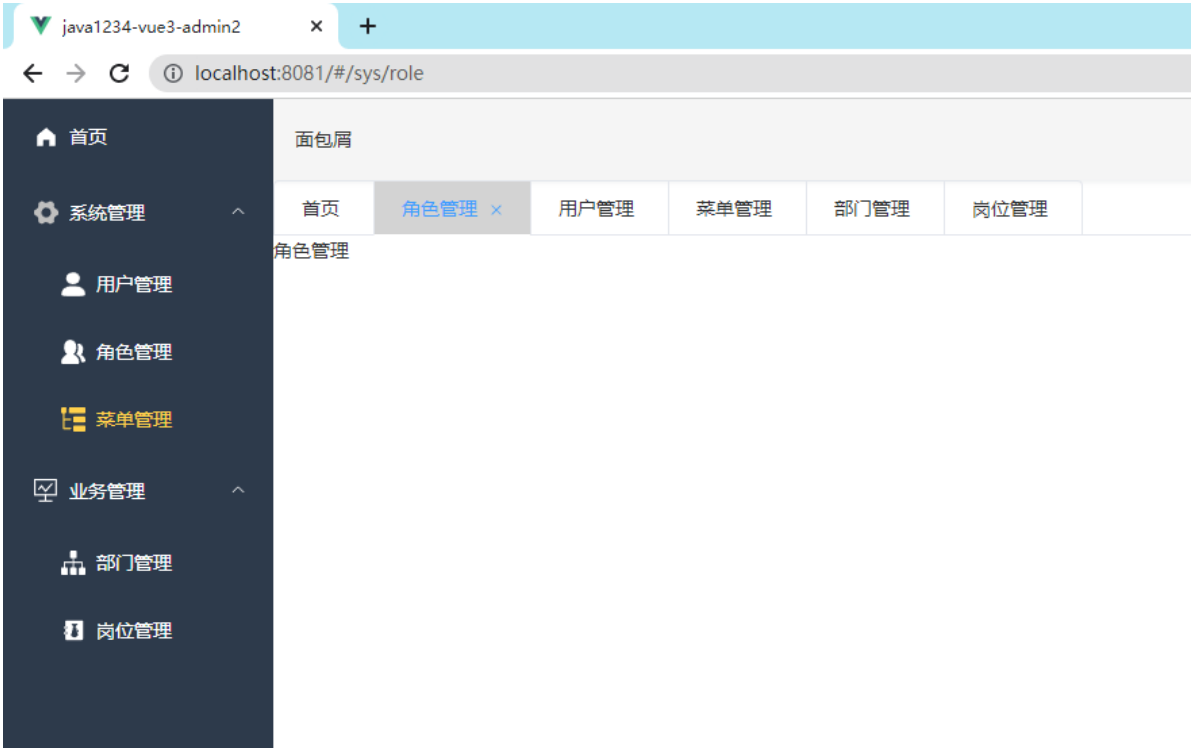
menu index.vue

```
          <el-menu-item :index="item.path" v-for="item in menu.children" @click="openTab(item)">
            <el-icon><svg-icon :icon="item.icon"/></el-icon>
            <span>{{item.name}}</span>
          </el-menu-item>

      </el-sub-menu>
  </el-menu>
</template>

<script setup>
import ...

const menuList=ref(store.getters.GET_MENULIST)
console.log("menuList2="+menuList)

const openTab=(item)=>{
  store.commit('ADD_TABS',item);
}
```

tabs index.vue

```
<template>
  <el-tabs
    v-model="editableTabsValue"
    type="card"
    class="demo-tabs"
    closable
    @tab-remove="removeTab"
    @tab-click="clickTab"
  >
    <el-tab-pane
      v-for="item in editableTabs"
      :key="item.name"
      :label="item.title"
      :name="item.name"
    >

    </el-tab-pane>
  </el-tabs>
</template>
<script setup>
import { ref ,watch} from 'vue'
import store from "@/store";
import {useRouter} from 'vue-router'
const router=useRouter();


const editableTabsValue = ref(store.state.editableTabsValue)
const editableTabs = ref(store.state.editableTabs)


const removeTab = (targetName) => {
  const tabs = editableTabs.value
  let activeName = editableTabsValue.value

  if(activeName==='/index'){
    return
  }

  if (activeName === targetName) {
    tabs.forEach((tab, index) => {
      if (tab.name === targetName) {
        const nextTab = tabs[index + 1] || tabs[index - 1]
        if (nextTab) {
          activeName = nextTab.name
        }
      }
    })
  }

  editableTabsValue.value = activeName
  editableTabs.value = tabs.filter((tab) => tab.name !== targetName)

  store.state.editableTabsValue=editableTabsValue.value
```

```
  store.state.editableTabs=editableTabs.value

  router.push({path:activeName})
}

const refreshTabs=()=>{
  editableTabsValue.value=store.state.editableTabsValue;
  editableTabs.value=store.state.editableTabs;
}

const clickTab=(target)=>{
  console.log('target.props.label='+target.props.label)
  router.push({name:target.props.label})
}

watch(store.state,()=>{
  refreshTabs();
},{deep:true,immediate:true})

</script>
<style>
.demo-tabs > .el-tabs__content {
  padding: 32px;
  color: #6b778c;
  font-size: 32px;
  font-weight: 600;
}

.el-main{
  padding:0px;
}


.el-tabs--card>.el-tabs__header .el-tabs__item.is-active{
  background-color: lightgray;
}

.el-tabs{
  height:45px
}
</style>
```



动态面包屑实现

breadcrumb.vue

```
<template>
  <el-icon><HomeFilled /></el-icon>
    <el-breadcrumb separator="/">

      <el-breadcrumb-item v-for="(item,index) in breadcrumbList" :key="index">
        <span class="root" v-if="parentName && index>0">
{{parentName}}  /  </span>
        <span class="leaf" v-if="index==breadcrumbList.length-1">{{item.name}}
</span>
        <span class="root" v-else>{{item.name}}</span>
      </el-breadcrumb-item>

    </el-breadcrumb>
</template>

<script setup>
import { ref,watch } from 'vue'
import {HomeFilled} from '@element-plus/icons-vue'
import {useRoute} from 'vue-router'
import store from "@/store";

const route=useRoute();
const breadcrumbList=ref([]);
const parentName=ref("")

const initBreadcrumbList=()=>{
  breadcrumbList.value=route.matched;
  parentName.value=route.meta.parentName;
}
```

```
watch(route,()=>{
  initBreadcrumbList();
},{deep:true,immediate:true})


</script>

<style lang="scss" scoped>
.leaf{
  cursor:text;
}
.root{
  color:#666;
  font-weight:600;
}
</style>
```

index index.vue

```
<template>
  <div class="home">
    欢迎使用，Java1234 通用权限系统 ！
  </div>

</template>

<script>
export default {
  name: "index"
};
</script>

<style lang="scss" scoped>
```

```
.home{
  padding: 40px;
  font-size: 30px;
  font-weight: bold;
}
</style>
```

个人中心功能实现

路由与导航动态绑定实现

App.vue加上下面这个 监控route,动态添加标签

```
import { ref ,watch} from 'vue'
import { useRoute,useRouter } from 'vue-router'
const route=useRoute();
const router=useRouter();
const whitePath=['/login','/index','/']


watch(route,(to,from)=>{
  console.log("to"+to.name)
  console.log(to.path)

  if (whitePath.indexOf(to.path)===-1) {
    console.log("to.path="+to.path)
    let obj = {
      name: to.name,
      path: to.path
    }

    store.commit("ADD_TABS", obj)
  }

},{deep:true,immediate:true})
```

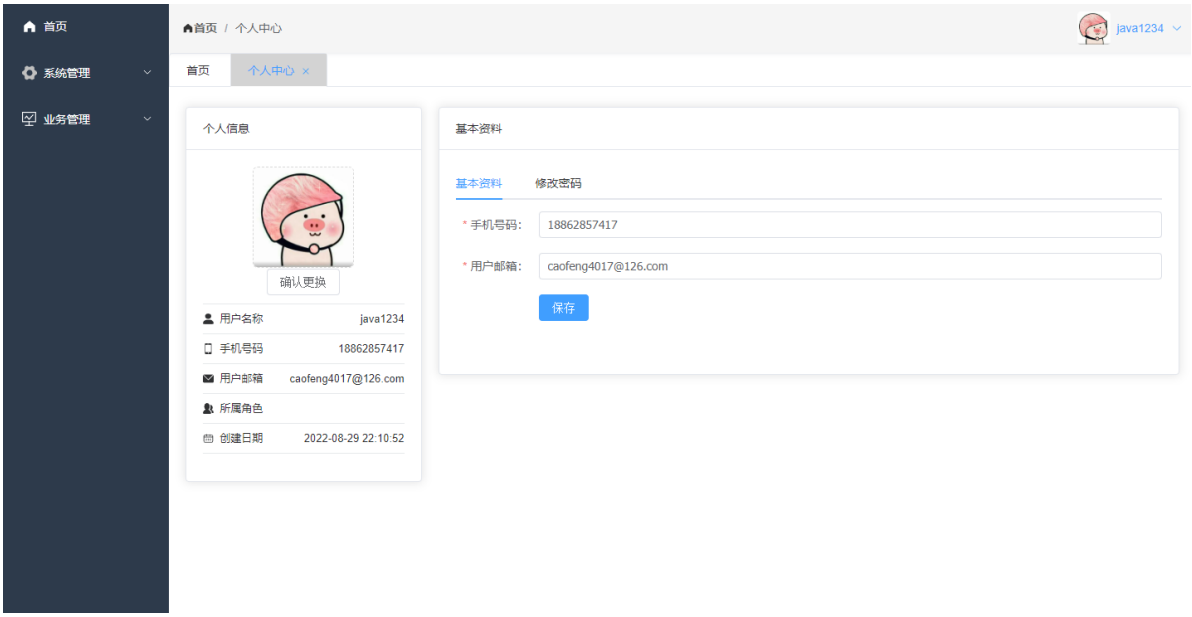
```
<template>
    <el-menu
        active-text-color="#ffd04b"
        background-color="#2d3a4b"
        class="el-menu-vertical-demo"
        text-color="#fff"
        router
        :default-active="activeIndex"
      >
      <el-menu-item index="/index">
        <el-icon><home-filled /></el-icon>
        <span>首页</span>
```

```
const activeIndex=ref("/index")

watch(store.state,()=>{
  console.log("editableTabsValue="+store.state.editableTabsValue)
  activeIndex.value=store.state.editableTabsValue
},{deep:true,immediate:true})
```

avatar.vue加下router-link

```
index.js | avatar.vue | AboutView.vue | App.vue | tabs\index.vue | breadcrumb.vue
4         <el-avatar shape="square" :size="40" :src="squareUrl" />
5           {{ currentUser.username }}
6         <el-icon class="el-icon--right">
7           <arrow-down />
8         </el-icon>
9       </span>
10      <template #dropdown>
11        <el-dropdown-menu>
12          <el-dropdown-item>
13            <router-link :to="{name:'个人中心'}">个人中心</router-link>
14          </el-dropdown-item>
15          <el-dropdown-item @click="logout">安全退出</el-dropdown-item>
16        </el-dropdown-menu>
```

个人中心页面构建实现



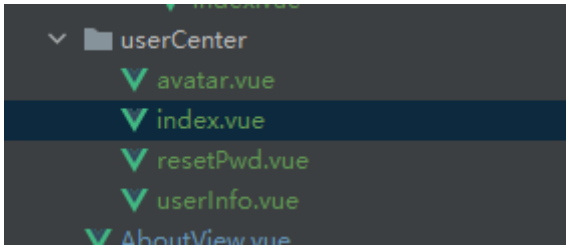

```
<template>
  <div class="app-container">
      <el-row :gutter="20">
        <el-col :span="6">
          <el-card class="box-card">
               <template v-slot:header>
                 <div class="clearfix">
                   <span>个人信息</span>
                 </div>
               </template>
               <div>
```

```
                    <div class="text-center">
                       修改头像
                    </div>
                    <ul class="list-group list-group-striped">
                       <li class="list-group-item">
                          <svg-icon icon="user" />  用户名称
                          <div class="pull-right">{{currentUser.username}}</div>
                       </li>
                       <li class="list-group-item">
                          <svg-icon icon="phone" />  手机号码
                          <div class="pull-right">{{currentUser.phonenumber}}
</div>
                       </li>
                       <li class="list-group-item">
                          <svg-icon icon="email" />  用户邮箱
                          <div class="pull-right">{{currentUser.email}}</div>
                       </li>
                       <li class="list-group-item">
                          <svg-icon icon="peoples" />  所属角色
                          <div class="pull-right">{{currentUser.roles}}</div>
                       </li>
                       <li class="list-group-item">
                          <svg-icon icon="date" />  创建日期
                          <div class="pull-right">
{{formatDate(currentUser.loginDate)}}</div>
                       </li>
                    </ul>
                 </div>
              </el-card>
           </el-col>
           <el-col :span="18">
             <el-card>
                  <template v-slot:header>
                    <div class="clearfix">
                      <span>基本资料</span>
                    </div>
                  </template>
                  <el-tabs v-model="activeTab">
                     <el-tab-pane label="基本资料" name="userinfo">
                         基本资料
                     </el-tab-pane>
                     <el-tab-pane label="修改密码" name="resetPwd">
                         修改密码
                     </el-tab-pane>
                  </el-tabs>
              </el-card>
           </el-col>
        </el-row>
     </div>
</template>

<script setup>
import {ref} from 'vue'
import store from '@/store'
import { formatDate } from '@/util/formatDate.js'

const currentUser = ref(store.getters.GET_USERINFO);
```

```
const activeTab = ref("userinfo");


</script>

<style lang="scss" scoped>

.list-group-striped>.list-group-item {
  border-left: 0;
  border-right: 0;
  border-radius: 0;
  padding-left: 0;
  padding-right: 0;
}

.list-group-item {
  border-bottom: 1px solid #e7eaec;
  border-top: 1px solid #e7eaec;
  margin-bottom: -1px;
  padding: 11px 0;
  font-size: 13px;
}

.pull-right{
  float: right!important;
}

::v-deep .el-card__body{
  height:230px;
}

::v-deep .box-card{
  height:450px;
}
</style>
```

个人中心页面数据显示

新建 avatar.vue resetPwd.vue userInfo.vue




```
<script setup>
import avatar from './avatar'
import userInfo from './userInfo'
import resetPwd from './resetPwd'
```

```
            <div class="text-center">
              <avatar/>
            </div>
            <ul class="list-group list-group-striped">
              <li class="list-group-item">
                <svg-icon icon="user" /> 用户名称
                <div class="pull-right">{{currentUser.username}}</div>
              </li>
              <li class="list-group-item">
                <svg-icon icon="phone" /> 手机号码
                <div class="pull-right">{{currentUser.phonenumber}}</d
              </li>
              <li class="list-group-item">
                <svg-icon icon="email" /> 用户邮箱
                <div class="pull-right">{{currentUser.email}}</div>
              </li>
              <li class="list-group-item">
                <svg-icon icon="peoples" /> 所属角色
                <div class="pull-right">{{currentUser.roles}}</div>
              </li>
              <li class="list-group-item">
                <svg-icon icon="date" /> 创建日期
                <div class="pull-right">{{formatDate(currentUser.loginI
              </li>
            </ul>
          </div>
        </el-card>
      </el-col>
      <el-col :span="18">
        <el-card>
          <template v-slot:header>
            <div class="clearfix">
              <span>基本资料</span>
            </div>
          </template>
          <el-tabs v-model="activeTab">
            <el-tab-pane label="基本资料" name="userinfo">
              <userInfo/>
            </el-tab-pane>
            <el-tab-pane label="修改密码" name="resetPwd">
              <resetPwd/>
            </el-tab-pane>
          </el-tabs>
```

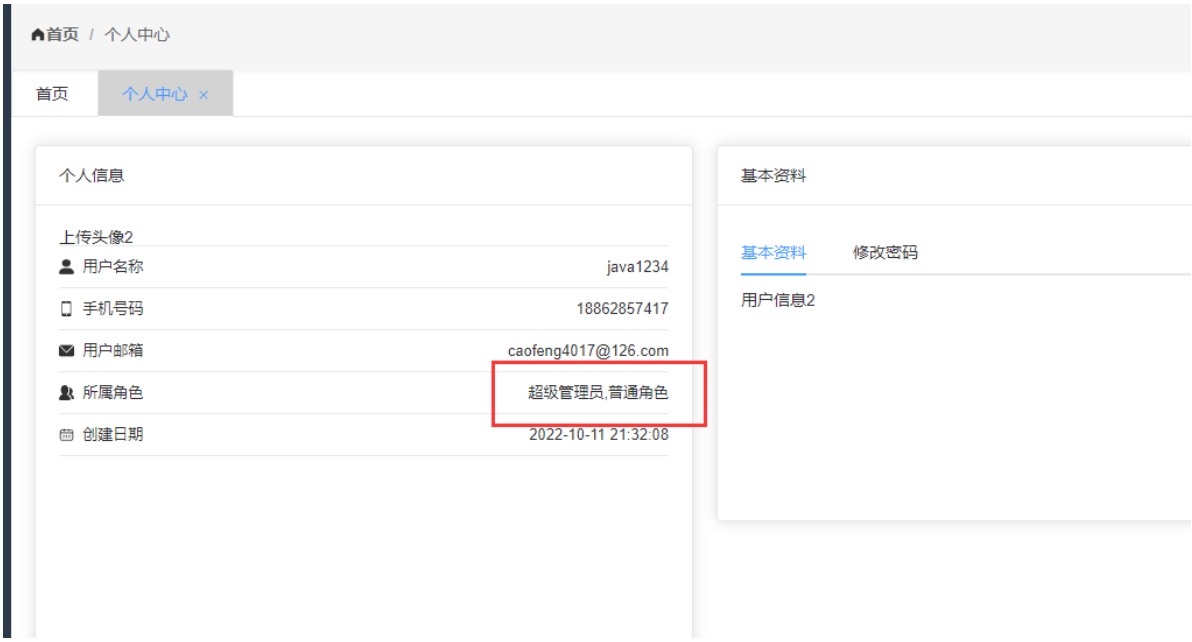
SysUser添加role属性

```
/**
 * 所属角色
 */
@TableField(exist = false)
private String roles;
```

LoginSuccessHandler.java

加上

```
currentUser.setRoles(roleList.stream().map(SysRole::getName).collect(Collectors.
joining(",")));
```



基本资料修改功能实现

userInfo.vue

```
<template>
  <el-form ref="userRef" :model="form" :rules="rules" label-width="100px" >
      <el-form-item label="手机号码：" prop="phonenumber">
         <el-input v-model="form.phonenumber" maxlength="11" />
      </el-form-item>
      <el-form-item label="用户邮箱：" prop="email">
         <el-input v-model="form.email" maxlength="50" />
      </el-form-item>
      <el-form-item>
      <el-button type="primary" @click="handleSubmit">保存</el-button>

      </el-form-item>
   </el-form>
</template>

<script setup>
```

```
import {defineProps, ref} from "vue";
import requestUtil from "@/util/request";
import { ElMessage } from 'element-plus'
import store from "@/store";

const props=defineProps(
    {
      user:{
        type:Object,
        default:()=>{},
        required:true
      }
    }
)

const form=ref({
  id:-1,
  phonenumber:'',
  email:''
})

const userRef=ref(null)

const rules = ref({
  email: [{ required: true, message: "邮箱地址不能为空", trigger: "blur" }, { type:
"email", message: "请输入正确的邮箱地址", trigger: ["blur", "change"] }],
  phonenumber: [{ required: true, message: "手机号码不能为空", trigger: "blur" }, {
pattern: /^1[3|4|5|6|7|8|9][0-9]\d{8}$/, message: "请输入正确的手机号码", trigger:
"blur" }],
});

form.value=props.user;

const handleSubmit=()=>{

  userRef.value.validate(async (valid)=>{
    if(valid) {
      let result = await requestUtil.post("sys/user/save", form.value);
      let data = result.data;
      if (data.code == 200) {
        ElMessage.success("执行成功！")
        store.commit("SET_USERINFO", form.value)
      }
    }
  })
}



</script>

<style lang="scss" scoped>

</style>
```

父页面传参

```
</template>
<el-tabs v-model="activeTab">
    <el-tab-pane label="基本资料" name="userinfo">
        <userInfo :user="currentUser"/>
    </el-tab-pane>
    <el-tab-pane label="修改密码" name="resetPwd">
        <resetPwd/>
```

新建SysUserController

```
/**
 * 用户Controller控制器
 * @author java1234_小锋 （公众号：java1234）
 * @site www.java1234.vip
 * @company 南通小锋网络科技有限公司
 */
@RestController
@RequestMapping("/sys/user")
public class SysUserController {

    @Autowired
    private SysUserService sysUserService;

    /**
     * 添加或者修改
     * @param sysUser
     * @return
     */
    @PostMapping("/save")

 @PreAuthorize("hasAuthority('system:user:add')"+"||"+"hasAuthority('system:user
:edit')")
    public R save(@RequestBody SysUser sysUser){
        if(sysUser.getId()==null || sysUser.getId()==-1){

        }else{
            sysUser.setUpdateTime(new Date());
            sysUserService.updateById(sysUser);
        }
        return R.ok();
    }
}
```

修改密码功能实现

resetPwd.vue

```
<template>
  <el-form ref="pwdRef" :model="form" :rules="rules" label-width="80px">
        <el-form-item label="旧密码" prop="oldPassword">
```

```
            <el-input v-model="form.oldPassword" placeholder="请输入旧密码"
type="password" show-password />
        </el-form-item>
        <el-form-item label="新密码" prop="newPassword">
            <el-input v-model="form.newPassword" placeholder="请输入新密码"
type="password" show-password />
        </el-form-item>
        <el-form-item label="确认密码" prop="confirmPassword">
            <el-input v-model="form.confirmPassword" placeholder="请确认密码"
type="password" show-password/>
        </el-form-item>
        <el-form-item>
        <el-button type="primary" @click="handleSubmit">保存</el-button>

        </el-form-item>
   </el-form>
</template>

<script setup>
import {defineProps, ref} from "vue";
import requestUtil from "@/util/request";
import { ElMessage } from 'element-plus'
import store from "@/store";


const props=defineProps(
    {
      user:{
        type:Object,
        default:()=>{},
        required:true
      }
    }
)

const form=ref({
  id:-1,
  oldPassword:'',
  newPassword:'',
  confirmPassword:''
})

const pwdRef=ref(null)

form.value=props.user;



const equalToPassword = (rule, value, callback) => {
  if (form.value.newPassword !== value) {
    callback(new Error("两次输入的密码不一致"));
  } else {
    callback();
  }
};

const rules = ref({
  oldPassword: [{ required: true, message: "旧密码不能为空", trigger: "blur" }],
```

```
  newPassword: [{ required: true, message: "新密码不能为空", trigger: "blur" }, {
min: 6, max: 20, message: "长度在 6 到 20 个字符", trigger: "blur" }],
  confirmPassword: [{ required: true, message: "确认密码不能为空", trigger: "blur"
}, { required: true, validator: equalToPassword, trigger: "blur" }]
});


const handleSubmit=()=>{

  pwdRef.value.validate(async (valid)=>{
    if(valid) {
      let result = await requestUtil.post("sys/user/updateUserPwd", form.value);
      let data = result.data;
      if (data.code == 200) {
        ElMessage.success("密码修改成功，下一次登录生效！")
        store.commit("SET_USERINFO", form.value)
      }else{
        ElMessage.error(data.msg)
      }
    }
  })
}


</script>

<style lang="scss" scoped>

</style>
```

```
@Autowired
BCryptPasswordEncoder bCryptPasswordEncoder;
```

SysUser

```
/**
 * 确认新密码
 */
@TableField(exist = false)
private String newPassword;

/**
 * 旧密码（前端传来的）
 */
@TableField(exist = false)
private String oldPassword;
```

后端SysUserController.java

```
/**
```

```
 * 修改密码
 * @param sysUser
 * @return
 */
@PostMapping("/updateUserPwd")
@PreAuthorize("hasAuthority('system:user:edit')")
public R updateUserPwd(@RequestBody SysUser sysUser){
    SysUser currentUser = sysUserService.getById(sysUser.getId());

 if(bCryptPasswordEncoder.matches(sysUser.getOldPassword(),currentUser.getPasswo
rd())){

 currentUser.setPassword(bCryptPasswordEncoder.encode(sysUser.getNewPassword()))
;
        currentUser.setUpdateTime(new Date());
        sysUserService.updateById(currentUser);
    }else{
        return R.error("输入旧密码错误！");
    }
    return R.ok();
}
```

头像更换功能实现

avatar.vue

```
<template>


    <el-form
        ref="formRef"
        :model="form"
        label-width="100px"
        style="text-align: center;padding-bottom:10px"
    >

      <el-upload
          :headers="headers"
          class="avatar-uploader"
          :action="getServerUrl()+'sys/user/uploadImage'"
          :show-file-list="false"
          :on-success="handleAvatarSuccess"
          :before-upload="beforeAvatarUpload"
      >
        <img v-if="imageUrl" :src="imageUrl" class="avatar" />
        <el-icon v-else class="avatar-uploader-icon"><Plus /></el-icon>
      </el-upload>

      <el-button @click="handleConfirm" >确认更换</el-button>

    </el-form>


</template>
```

```
<script setup>

import {defineProps, ref} from "vue";
import requestUtil,{getServerUrl} from "@/util/request";
import { ElMessage } from 'element-plus'
import { Plus } from '@element-plus/icons-vue'
import store from "@/store";



const props=defineProps(
    {
      user:{
        type:Object,
        default:()=>{},
        required:true
      }
    }
)

const headers=ref({
  token:store.getters.GET_TOKEN
})

const form=ref({
  id:-1,
  avatar:''
})

const formRef=ref(null)

const imageUrl=ref("")

form.value=props.user;
imageUrl.value=getServerUrl()+'image/userAvatar/'+form.value.avatar

const handleAvatarSuccess=(res)=>{
  imageUrl.value=getServerUrl()+res.data.src
  form.value.avatar=res.data.title;
}


const beforeAvatarUpload = (file) => {
  const isJPG = file.type === 'image/jpeg'
  const isLt2M = file.size / 1024 / 1024 < 2

  if (!isJPG) {
    ElMessage.error('图片必须是jpg格式')
  }
  if (!isLt2M) {
    ElMessage.error('图片大小不能超过2M!')
  }
  return isJPG && isLt2M
}

  const handleConfirm=async()=>{

          let result=await requestUtil.post("sys/user/updateAvatar",form.value);
```

```
            let data=result.data;
            if(data.code==200){
              ElMessage.success("执行成功！")
              store.commit("SET_USERINFO",form.value)
            }else{
              ElMessage.error(data.msg);
            }

  }

</script>

<style>

.avatar-uploader .el-upload {
  border: 1px dashed #d9d9d9;
  border-radius: 6px;
  cursor: pointer;
  position: relative;
  overflow: hidden;
}
.avatar-uploader .el-upload:hover {
  border-color: #409eff;
}
.el-icon.avatar-uploader-icon {
  font-size: 28px;
  color: #8c939d;
  width: 178px;
  height: 178px;
  text-align: center;
}
.avatar {
  width: 120px;
  height: 120px;
  display: block;
}



</style>
```

```
avatarImagesFilePath: D://java1234-admin2/userAvatar/
```

```
@Value("${avatarImagesFilePath}")
private String avatarImagesFilePath;
```

```
/**
 * 上传用户头像图片
 * @param file
 * @return
 * @throws Exception
 */
@RequestMapping("/uploadImage")
@PreAuthorize("hasAuthority('system:user:edit')")
public Map<String,Object> uploadImage(MultipartFile file)throws Exception{
```

```
    Map<String,Object> resultMap=new HashMap<>();
    if(!file.isEmpty()){
        // 获取文件名
        String originalFilename = file.getOriginalFilename();
        String
suffixName=originalFilename.substring(originalFilename.lastIndexOf("."));
        String newFileName= DateUtil.getCurrentDateStr()+suffixName;
        FileUtils.copyInputStreamToFile(file.getInputStream(),new
File(avatarImagesFilePath+newFileName));
        resultMap.put("code",0);
        resultMap.put("msg","上传成功");
        Map<String,Object> dataMap=new HashMap<>();
        dataMap.put("title",newFileName);
        dataMap.put("src","image/userAvatar/"+newFileName);
        resultMap.put("data",dataMap);
    }
    return resultMap;
}
```

```
/**
 * 修改用户头像
 * @param sysUser
 * @return
 */
@RequestMapping("/updateAvatar")
@PreAuthorize("hasAuthority('system:user:edit')")
public R updateAvatar(@RequestBody SysUser sysUser){
    SysUser currentUser = sysUserService.getById(sysUser.getId());
    currentUser.setUpdateTime(new Date());
    currentUser.setAvatar(sysUser.getAvatar());
    sysUserService.updateById(currentUser);
    return R.ok();
}
```

用户管理实现

用户管理-列表分页显示实现

分页实体PageBean：

```
package com.java1234.entity;

/**
 * 分页Model类
 * @author java1234_小锋
 * @site www.java1234.com
 * @company Java知识分享网
 * @create 2020-02-16 上午 11:05
 */
public class PageBean {


    private int pageNum; // 第几页
    private int pageSize; // 每页记录数
```

```
    private int start;  // 起始页
    private String query; // 查询参数

    public PageBean() {
    }

    public PageBean(int pageNum, int pageSize, String query) {
        this.pageNum = pageNum;
        this.pageSize = pageSize;
        this.query = query;
    }

    public PageBean(int pageNum, int pageSize) {
        super();
        this.pageNum = pageNum;
        this.pageSize = pageSize;
    }

    public int getPageNum() {
        return pageNum;
    }

    public void setPageNum(int pageNum) {
        this.pageNum = pageNum;
    }

    public int getPageSize() {
        return pageSize;
    }

    public void setPageSize(int pageSize) {
        this.pageSize = pageSize;
    }

    public int getStart() {
        return (pageNum-1)*pageSize;
    }

    public String getQuery() {
        return query;
    }

    public void setQuery(String query) {
        this.query = query;
    }
}
```

MybatisPlus开启分页功能：

```
package com.java1234.config;

import com.baomidou.mybatisplus.extension.plugins.PaginationInterceptor;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
```

```
/**
 * MybatisPlus配置类
 * @author java1234_小锋
 * @site www.java1234.com
 * @company 南通小锋网络科技有限公司
 * @create 2022-01-30 8:10
 */
@Configuration
public class MybatisPlusConfig {

    @Bean
    public PaginationInterceptor paginationInterceptor(){
        return new PaginationInterceptor();
    }
}
```

SysUserController

```
/**
 * 根据条件分页查询用户信息
 * @param pageBean
 * @return
 */
@PostMapping("/list")
@PreAuthorize("hasAuthority('system:user:query')")
public R list(@RequestBody PageBean pageBean){
    Page<SysUser> pageResult = sysUserService.page(new Page<>
(pageBean.getPageNum(), pageBean.getPageSize()));
    List<SysUser> userList = pageResult.getRecords();
    Map<String,Object> resultMap=new HashMap<>();
    resultMap.put("userList",userList);
    resultMap.put("total",pageResult.getTotal());
    return R.ok(resultMap);
}
```

前端user/index.vue

```
<template>
  <div class="app-container">

      <el-table :data="tableData" stripe style="width: 100%">
        <el-table-column prop="username" label="用户名" width="180" />

      </el-table>
      <el-pagination
          v-model:currentPage="queryForm.pageNum"
          v-model:page-size="queryForm.pageSize"
          :page-sizes="[10, 20, 30, 40]"
          layout="total, sizes, prev, pager, next, jumper"
          :total="total"
          @size-change="handleSizeChange"
          @current-change="handleCurrentChange"
      />
```

```
  </div>
</template>

<script setup>
import requestUtil,{getServerUrl} from "@/util/request";
import { ref } from 'vue'

const tableData=ref([]);

const total=ref(0)

const queryForm=ref({
  query:'',
  pageNum:1,
  pageSize:10
})

const initUserList=async()=>{
  const res=await requestUtil.post("sys/user/list",queryForm.value);
  tableData.value=res.data.userList;
  total.value=res.data.total;
}

initUserList();

const handleSizeChange=(pageSize)=>{
  queryForm.value.pageNum=1;
  queryForm.value.pageSize=pageSize;
  initUserList()
}

const handleCurrentChange=(pageNum)=>{
  queryForm.value.pageNum=pageNum;
  initUserList()
}

</script>

<style lang="scss" scoped>

.header{
  padding-bottom: 16px;
  box-sizing: border-box;
}

.el-pagination{
  float: right;
  padding: 20px;
  box-sizing: border-box;
}

::v-deep th.el-table__cell{
  word-break: break-word;
  background-color: #f8f8f9 !important;
  color: #515a6e;
  height: 40px;
  font-size: 13px;
```

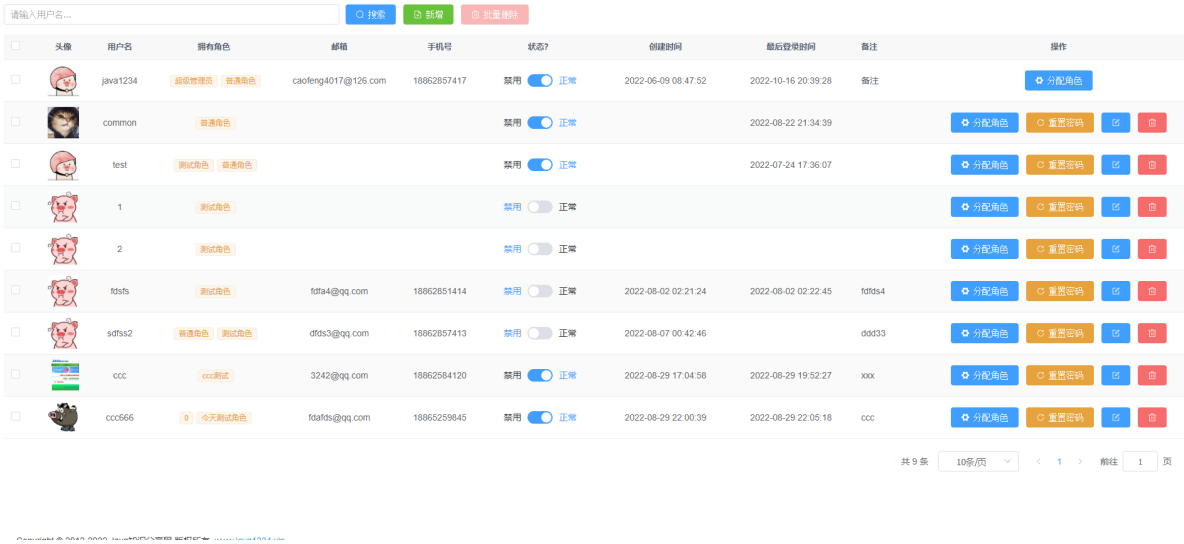
```
}

.el-tag--small {
  margin-left: 5px;
}
</style>
```

elementplus国际化中文

```
// 国际化中文
import zhCn from 'element-plus/es/locale/lang/zh-cn'
```

```
app.use(ElementPlus, {
    locale: zhCn,
})
```

用户管理-搜索功能实现



前端需要显示角色信息

SysUser实体加字段

```
/**
 * 所有角色集合
 */
@TableField(exist = false)
private List<SysRole> sysRoleList;
```

```
/**
 * 根据条件分页查询用户信息
 * @param pageBean
 * @return
 */
@PostMapping("/list")
```

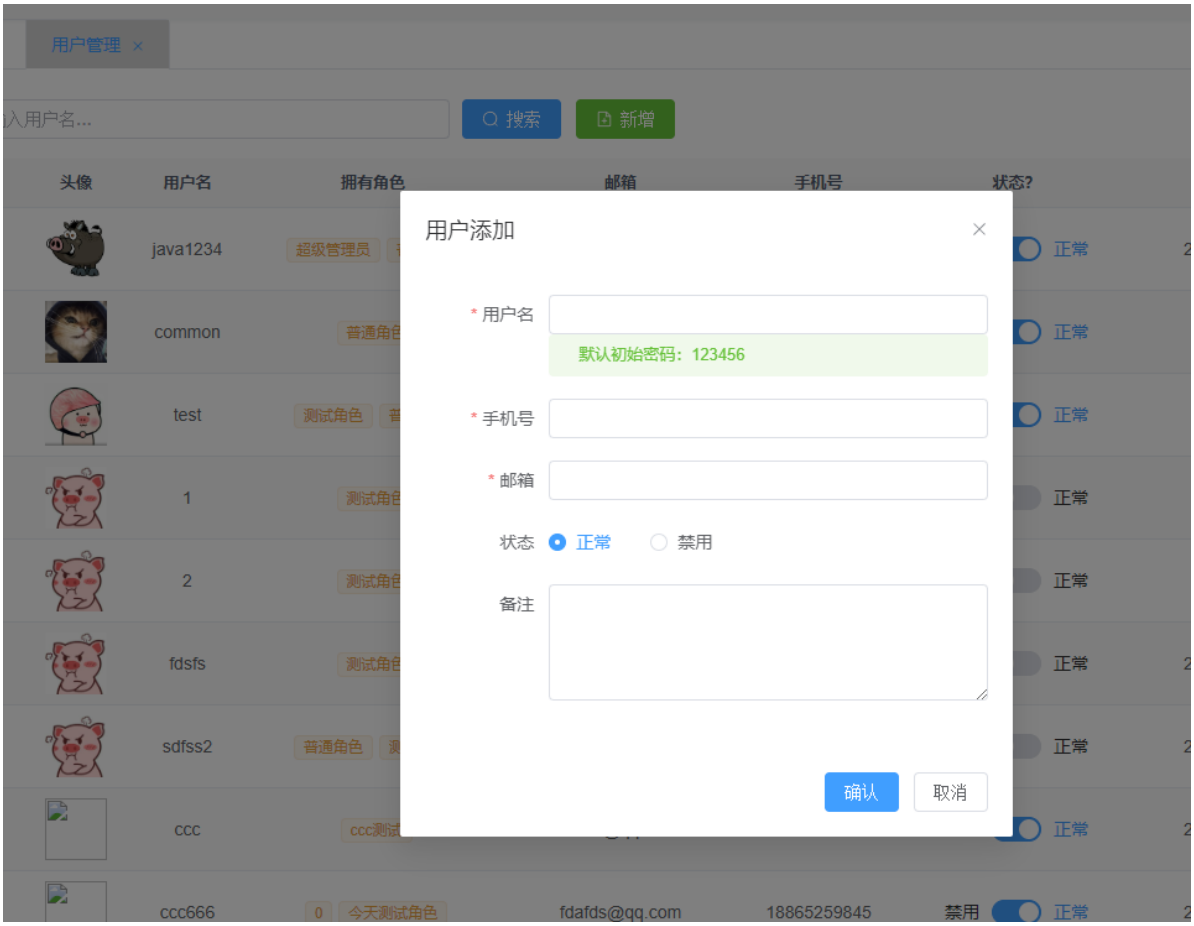
```
@PreAuthorize("hasAuthority('system:user:query')")
public R list(@RequestBody PageBean pageBean){
    String query=pageBean.getQuery().trim();
    Page<SysUser> pageResult = sysUserService.page(new Page<>
(pageBean.getPageNum(), pageBean.getPageSize()),new QueryWrapper<SysUser>
().like(StringUtil.isNotEmpty(query),"username",query));
    List<SysUser> userList = pageResult.getRecords();
    for(SysUser user:userList){
        List<SysRole> roleList = sysRoleService.list(new QueryWrapper<SysRole>
().inSql("id", "select role_id from sys_user_role where user_id=" +
user.getId()));
        user.setSysRoleList(roleList);
    }
    Map<String,Object> resultMap=new HashMap<>();
    resultMap.put("userList",userList);
    resultMap.put("total",pageResult.getTotal());
    return R.ok(resultMap);
}
```

```
<template>
  <div class="app-container">

    <el-row :gutter="20" class="header">
      <el-col :span="7">
        <el-input placeholder="请输入用户名..." v-model="queryForm.query" clearable
></el-input>
      </el-col>
      <el-button type="primary" :icon="Search" @click="initUserList">搜索</el-
button>
    </el-row>

      <el-table :data="tableData" stripe style="width: 100%">
        <el-table-column type="selection" width="55" />
        <el-table-column prop="avatar" label="头像" width="80" align="center">
          <template v-slot="scope">
            <img :src="getServerUrl()+'image/userAvatar/'+scope.row.avatar"
width="50" height="50"/>
          </template>
        </el-table-column>
        <el-table-column prop="username" label="用户名"  width="100"
align="center"/>
        <el-table-column prop="roles" label="拥有角色" width="200" align="center">
          <template v-slot="scope">
            <el-tag size="small" type="warning" v-for="item in
scope.row.sysRoleList">   {{item.name}}</el-tag>
          </template>
        </el-table-column>
        <el-table-column prop="email" label="邮箱" width="200" align="center"/>
        <el-table-column prop="phonenumber" label="手机号" width="120"
align="center"/>
        <el-table-column prop="status" label="状态？" width="200" align="center"
>
          <template v-slot="{row}" >
            <el-switch  v-model="row.status" @change="statusChangeHandle(row)"
active-text="正常"
```

```
                        inactive-text="禁用" active-value="0" inactive-value="1">
</el-switch>
          </template>
        </el-table-column>
        <el-table-column prop="createTime" label="创建时间" width="200"
align="center"/>
        <el-table-column prop="loginDate" label="最后登录时间" width="200"
align="center"/>
        <el-table-column prop="remark" label="备注"  />
        <el-table-column prop="action" label="操作" width="400" fixed="right"
align="center">
        <template v-slot="scope" >
          <el-button  type="primary" :icon="Tools" >分配角色</el-button>



        </template>
      </el-table-column>
      </el-table>
      <el-pagination
          v-model:currentPage="queryForm.pageNum"
          v-model:page-size="queryForm.pageSize"
          :page-sizes="[10, 20, 30, 40]"
          layout="total, sizes, prev, pager, next, jumper"
          :total="total"
          @size-change="handleSizeChange"
          @current-change="handleCurrentChange"
      />
  </div>
</template>

<script setup>
import { Search ,Delete,DocumentAdd ,Edit, Tools, RefreshRight} from '@element-
plus/icons-vue'
import requestUtil,{getServerUrl} from "@/util/request";
import { ref } from 'vue'

const tableData=ref([]);

const total=ref(0)

const queryForm=ref({
  query:'',
  pageNum:1,
  pageSize:10
})

const initUserList=async()=>{
  const res=await requestUtil.post("sys/user/list",queryForm.value);
  tableData.value=res.data.userList;
  total.value=res.data.total;
}

initUserList();

const handleSizeChange=(pageSize)=>{
  queryForm.value.pageNum=1;
  queryForm.value.pageSize=pageSize;
```

```
  initUserList()
}

const handleCurrentChange=(pageNum)=>{
  queryForm.value.pageNum=pageNum;
  initUserList()
}

</script>

<style lang="scss" scoped>

.header{
  padding-bottom: 16px;
  box-sizing: border-box;
}

.el-pagination{
  float: right;
  padding: 20px;
  box-sizing: border-box;
}

::v-deep th.el-table__cell{
  word-break: break-word;
  background-color: #f8f8f9 !important;
  color: #515a6e;
  height: 40px;
  font-size: 13px;

}

.el-tag--small {
  margin-left: 5px;
}
</style>
```

用户管理-添加修改功能实现



后端提供一个API

添加或者修改：

```
/**
```

```
 * 添加或者修改
 * @param sysUser
 * @return
 */
@PostMapping("/save")
@PreAuthorize("hasAuthority('system:user:add')"+"||"+"hasAuthority('system:user:
edit')")
public R save(@RequestBody SysUser sysUser){
    if(sysUser.getId()==null || sysUser.getId()==-1){
        sysUser.setCreateTime(new Date());

 sysUser.setPassword(bCryptPasswordEncoder.encode(sysUser.getPassword()));
        sysUserService.save(sysUser);
    }else{
        sysUser.setUpdateTime(new Date());
        sysUserService.updateById(sysUser);
    }
    return R.ok();
}
```

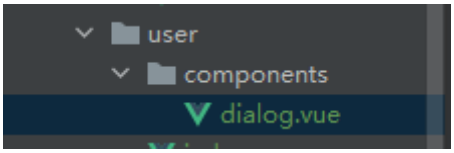
修改功能，需要根据id查询数据

```
/**
 * 根据id查询
 * @param id
 * @return
 */
@GetMapping("/{id}")
@PreAuthorize("hasAuthority('system:user:query')")
public R findById(@PathVariable(value = "id")Integer id){
    SysUser sysUser = sysUserService.getById(id);
    Map<String,Object> map=new HashMap<>();
    map.put("sysUser",sysUser);
    return R.ok(map);
}
```

添加用户，需要验证用户名是否重复

```
/**
 * 验证用户名
 * @param sysUser
 * @return
 */
@PostMapping("/checkUserName")
@PreAuthorize("hasAuthority('system:user:query')")
public R checkUserName(@RequestBody SysUser sysUser){
    if(sysUserService.getByUsername(sysUser.getUsername())==null){
        return R.ok();
    }else{
        return R.error();
    }
}
```

前端，新建一个 dialog.vue 作为用户添加和修改页面组件




```
<template>

  <el-dialog
    v-model="dialogVisible"
    :title="dialogTitle"
    width="30%"
    @close="handleClose"
  >
       <el-form
        ref="formRef"
        :model="form"
        :rules="rules"
        label-width="100px"
    >
      <el-form-item label="用户名" prop="username">
        <el-input v-model="form.username" :disabled="form.id==-1?
false:'disabled'"  />
        <el-alert
            v-if="form.id==-1"
            title="默认初始密码：123456"
            :closable="false"
            style="line-height: 10px;"
            type="success" >
        </el-alert>
      </el-form-item>

      <el-form-item label="手机号" prop="phonenumber">
        <el-input v-model="form.phonenumber" />
      </el-form-item>

      <el-form-item label="邮箱" prop="email">
        <el-input v-model="form.email" />
      </el-form-item>

      <el-form-item label="状态" prop="status">
        <el-radio-group v-model="form.status">
          <el-radio :label="'0'">正常</el-radio>
          <el-radio :label="'1'">禁用</el-radio>
        </el-radio-group>
      </el-form-item>


      <el-form-item label="备注" prop="remark">
        <el-input v-model="form.remark" type="textarea" :rows="4"/>
      </el-form-item>


    </el-form>
    <template #footer>
```

```
      <span class="dialog-footer">
        <el-button type="primary" @click="handleConfirm">确认</el-button>
        <el-button @click="handleClose"
          >取消</el-button
        >
      </span>
    </template>
  </el-dialog>

</template>

<script setup>
import {defineEmits, defineProps,ref,watch } from "vue"
import requestUtil,{getServerUrl} from "@/util/request";
import { ElMessage } from 'element-plus'

const props=defineProps(
    {
      id:{
        type:Number,
        default:-1,
        required:true
      },
      dialogTitle:{
        type:String,
        default:'',
        required:true
      },
      dialogVisible:{
        type:Boolean,
        default:false,
        required:true
      }
    }
)

const form=ref({
  id:-1,
  username:"",
  password:"123456",
  status:"0",
  phonenumber:"",
  email:"",
  remark:""
})

const checkUsername = async (rule, value, callback) => {
  if(form.value.id==-1){
    const res=await requestUtil.post("sys/user/checkUserName",
{username:form.value.username});
    if (res.data.code==500) {
      callback(new Error("用户名已存在！"));
    } else {
      callback();
    }
  }else{
    callback();
  }
```

```
}


const rules=ref({
  username:[
    { required: true, message: '请输入用户名'},
    { required: true, validator: checkUsername, trigger: "blur" }
  ],
  email: [{ required: true, message: "邮箱地址不能为空", trigger: "blur" }, { type:
"email", message: "请输入正确的邮箱地址", trigger: ["blur", "change"] }],
  phonenumber: [{ required: true, message: "手机号码不能为空", trigger: "blur" }, {
pattern: /^1[3|4|5|6|7|8|9][0-9]\d{8}$/, message: "请输入正确的手机号码", trigger:
"blur" }],
})

const formRef=ref(null)

const initFormData=async(id)=>{
  const res=await requestUtil.get("sys/user/"+id);
  form.value=res.data.sysUser;
}

watch(
    ()=>props.dialogVisible,
    ()=>{
      let id=props.id;
      console.log("id="+id)
      if(id!=-1){
        initFormData(id);
      }else{
        form.value={
          id:-1,
          username:"",
          password:"123456",
          status:"0",
          phonenumber:"",
          email:"",
          remark:""
        }
      }
    }
)

const emits=defineEmits(['update:modelValue','initUserList'])

const handleClose=()=>{
  emits('update:modelValue',false)
}

const handleConfirm=()=>{
  formRef.value.validate(async(valid)=>{
    if(valid){
      let result=await requestUtil.post("sys/user/save",form.value);
      let data=result.data;
      if(data.code==200){
        ElMessage.success("执行成功！")
        formRef.value.resetFields();
```

```
        emits("initUserList")
        handleClose();
      }else{
        ElMessage.error(data.msg);
      }
    }else{
      console.log("fail")
    }
  })
}

</script>

<style lang="scss" scoped>

</style>
```

index.vue页面

```
<el-button type="success" :icon="DocumentAdd" @click="handleDialogValue()">新增
</el-button>
```

```
<el-row :gutter="20" class="header">
  <el-col :span="7">
    <el-input placeholder="请输入用户名..." v-model="queryForm.query" clearable ></el-input>
  </el-col>
  <el-button type="primary" :icon="Search" @click="initUserList">搜索</el-button>
  <el-button type="success" :icon="DocumentAdd" @click="handleDialogValue()">新增</el-button>
</el-row>
```

```
<el-button v-if="scope.row.username!='java1234'" type="primary" :icon="Edit"
@click="handleDialogValue(scope.row.id)" />
```

```
  <el-table-column prop="remark" label="备注"  />
  <el-table-column prop="action" label="操作" width="400" fixed="right" align="center">
  <template v-slot="scope" >
    <el-button  type="primary" :icon="Tools" >分配角色</el-button>
    <el-button v-if="scope.row.username!='java1234'" type="primary" :icon="Edit" @click="handleDialogValue(scope.row.id)" />


  </template>
</el-table-column>
```

添加和修改按钮

```
import Dialog from './components/dialog'
```

```
<Dialog v-model="dialogVisible" :dialogVisible="dialogVisible" :id="id"
:dialogTitle="dialogTitle" @initUserList="initUserList"/>
```

```
    />
  </div>
  <Dialog v-model="dialogVisible" :dialogVisible="dialogVisible" :id="id" :dialogTitle="dialogTitle" @initUserList="initUserList"/>
</template>

<script setup>
import { Search ,Delete,DocumentAdd ,Edit, Tools, RefreshRight} from '@element-plus/icons-vue'
import requestUtil,{getServerUrl} from "@/util/request";
import { ref } from 'vue'
import Dialog from './components/dialog'
```

```
const dialogVisible=ref(false)

const dialogTitle=ref("")

const id=ref(-1)
```

```
const handleDialogValue=(userId)=>{
  if(userId){
    id.value=userId;
    dialogTitle.value="用户修改"
  }else{
    id.value=-1;
    dialogTitle.value="用户添加"
  }
  dialogVisible.value=true
}
```

用户管理-删除和批量删除功能实现

后端:

```
/**
 * 删除
 * @param ids
 * @return
 */
@Transactional
@PostMapping("/delete")
@PreAuthorize("hasAuthority('system:user:delete')")
public R delete(@RequestBody Long[] ids){
    sysUserService.removeByIds(Arrays.asList(ids));
    sysUserRoleService.remove(new QueryWrapper<SysUserRole>
().in("user_id",ids));
    return R.ok();
}
```

前端:

```
<el-popconfirm title="您确定批量删除这些记录吗？" @confirm="handleDelete(null)">
  <template #reference>
   <el-button type="danger" :disabled="delBtnStatus" :icon="Delete" >批量删除</el-
button>
  </template>
 </el-popconfirm>
```

```
const delBtnStatus=ref(true)
```

表格复选框选中事件：

```
@selection-change="handleSelectionChange"
```

```
<el-table :data="tableData" stripe style="width: 100%" @selection-change="handleSelectionChange">
```

定义 选中的行

```
const multipleSelection=ref([])
```

```
const handleSelectionChange=(selection)=>{
  console.log("勾选了")
  console.log(selection)
  multipleSelection.value=selection;
  delBtnStatus.value=selection.length==0;
}
```

行上面的删除按钮

```
<el-popconfirm  v-if="scope.row.username!='java1234'" title="您确定要删除这条记录
吗？" @confirm="handleDelete(scope.row.id)">
  <template #reference>
    <el-button  type="danger" :icon="Delete" />
  </template>
</el-popconfirm>
```

```
import { ElMessage, ElMessageBox } from 'element-plus'
```

```
const handleDelete=async (id)=>{
  var ids = []
```

```
  if(id){
    ids.push(id)
  }else{
    multipleSelection.value.forEach(row=>{
      ids.push(row.id)
    })
  }
  const res=await requestUtil.post("sys/user/delete",ids)
  if(res.data.code==200){
    ElMessage({
      type: 'success',
      message: '执行成功!'
    })
    initUserList();
  }else{
    ElMessage({
      type: 'error',
      message: res.data.msg,
    })
  }
}
```

用户管理-用户信息重置密码和状态更新功能实现

后端 重置密码:

```
/**
 * 重置密码
 * @param id
 * @return
 */
@GetMapping("/resetPassword/{id}")
@PreAuthorize("hasAuthority('system:user:edit')")
public R resetPassword(@PathVariable(value = "id")Integer id){
    SysUser sysUser = sysUserService.getById(id);

 sysUser.setPassword(bCryptPasswordEncoder.encode(Constant.DEFAULT_PASSWORD));
    sysUser.setUpdateTime(new Date());
    sysUserService.updateById(sysUser);
    return R.ok();
}
```

```java
package com.java1234.common.constant;

/**
 * 通用常量信息
 * @author java1234_小锋 （公众号：java1234）
 * @site www.java1234.vip
 * @company 南通小锋网络科技有限公司
 */
public class Constant {

    public final static String DEFAULT_PASSWORD="123456";
}
```

```java
/**
 * 更新status状态
 * @param id
 * @param status
 * @return
 */
@GetMapping("/updateStatus/{id}/status/{status}")
@PreAuthorize("hasAuthority('system:user:edit')")
public R updateStatus(@PathVariable(value = "id")Integer id,@PathVariable(value
= "status")String status){
    SysUser sysUser = sysUserService.getById(id);
    sysUser.setStatus(status);
    sysUserService.saveOrUpdate(sysUser);
    return R.ok();
}
```

前端：

```vue
<el-popconfirm  v-if="scope.row.username!='java1234'" title="您确定要对这个用户重置密
码吗？" @confirm="handleResetPassword(scope.row.id)">
  <template #reference>
    <el-button  type="warning" :icon="RefreshRight" >重置密码</el-button>
  </template>
</el-popconfirm>
```

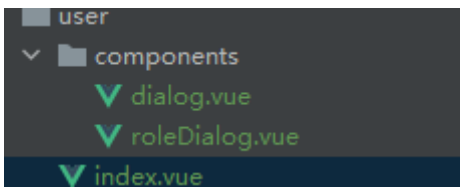
```vue
const handleResetPassword=async (id)=>{
  const res=await requestUtil.get("sys/user/resetPassword/"+id)
  if(res.data.code==200){
    ElMessage({
      type: 'success',
      message: '执行成功!'
    })
    initUserList();
  }else{
    ElMessage({
      type: 'error',
```

```
      message: res.data.msg,
    })
  }
}
```

```
const statusChangeHandle=async (row)=>{
  let res=await
requestUtil.get("sys/user/updateStatus/"+row.id+"/status/"+row.status);
  if(res.data.code==200){
    ElMessage({
      type: 'success',
      message: '执行成功!'
    })
  }else{
    ElMessage({
      type: 'error',
      message: res.data.msg,
    })
    initUserList();
  }
}
```

用户管理-用户分配角色功能实现

新建 roleDialog.vue




```
import RoleDialog from './components/roleDialog'
```

```
<RoleDialog v-model="roleDialogVisible" :sysRoleList="sysRoleList"
:roleDialogVisible="roleDialogVisible" :id="id" @initUserList="initUserList">
</RoleDialog>
```

```
const sysRoleList=ref([])

const roleDialogVisible=ref(false)
```

```
<el-button  type="primary" :icon="Tools"
@click="handleRoleDialogValue(scope.row.id,scope.row.sysRoleList)">分配角色</el-
button>
```

```
const handleRoleDialogValue=(userId,roleList)=>{
  console.log("userId="+userId)
  id.value=userId;
  sysRoleList.value=roleList;
  roleDialogVisible.value=true
}
```

```
<template>
  <el-dialog
    model-value="roleDialogVisible"
    title="分配角色"
    width="30%"
    @close="handleClose"
  >

    <el-form
        ref="formRef"
        :model="form"
        label-width="100px"
    >

      <el-checkbox-group v-model="form.checkedRoles">
        <el-checkbox v-for="role in form.roleList" :id="role.id" :key="role.id"
:label="role.id"  name="checkedRoles" >{{role.name}}</el-checkbox>
      </el-checkbox-group>

    </el-form>

    <template #footer>
      <span class="dialog-footer">
        <el-button type="primary" @click="handleConfirm">确认</el-button>
        <el-button  @click="handleClose">取消</el-button>
      </span>
    </template>
```

```
  </el-dialog>
</template>

<script setup>

import {defineEmits, defineProps, ref, watch} from "vue";
import requestUtil,{getServerUrl} from "@/util/request";
import { ElMessage } from 'element-plus'


  const props=defineProps(
      {
        id:{
          type:Number,
          default:-1,
          required:true
        },
        roleDialogVisible:{
          type:Boolean,
          default:false,
          required:true
        },
        sysRoleList:{
          type:Array,
          default:[],
          required:true
        }
      }
  )


const form=ref({
  id:-1,
  roleList:[],
  checkedRoles:[]
})


const formRef=ref(null)

const initFormData=async(id)=>{
  const res=await requestUtil.get("sys/role/listAll");
  form.value.roleList=res.data.roleList;
  form.value.id=id;
}


  watch(
      ()=>props.roleDialogVisible,
      ()=>{
        let id=props.id;
        console.log("id="+id)
        if(id!=-1){
          form.value.checkedRoles=[]
          props.sysRoleList.forEach(item=>{
            form.value.checkedRoles.push(item.id);
          })
          initFormData(id)
```

```
        }
      }
  )


  const emits=defineEmits(['update:modelValue','initUserList'])

  const handleClose=()=>{
    emits('update:modelValue',false)
  }

  const handleConfirm=()=>{
    formRef.value.validate(async(valid)=>{
      if(valid){
          let result=await
requestUtil.post("sys/user/grantRole/"+form.value.id,form.value.checkedRoles);
          let data=result.data;
          if(data.code==200){
            ElMessage.success("执行成功！")

            emits("initUserList")
            handleClose();
          }else{
            ElMessage.error(data.msg);
          }
      }else{
        console.log("fail")
      }
    })
  }

</script>

<style scoped>

</style>
```

```
/**
 * 系统角色Controller控制器
 * @author java1234_小锋 （公众号：java1234）
 * @site www.java1234.vip
 * @company 南通小锋网络科技有限公司
 */
@RestController
@RequestMapping("/sys/role")
public class SysRoleController {

    @Autowired
    private SysRoleService sysRoleService;

    @GetMapping("/listAll")
    @PreAuthorize("hasAuthority('system:role:query')")
    public R listAll(){
        Map<String,Object> resultMap=new HashMap<>();
        List<SysRole> roleList = sysRoleService.list();
```

```
        resultMap.put("roleList",roleList);
        return R.ok(resultMap);
    }
}
```

```
/**
 * 用户角色授权
 * @param userId
 * @param roleIds
 * @return
 */
@Transactional
@PostMapping("/grantRole/{userId}")
@PreAuthorize("hasAuthority('system:user:role')")
public R grantRole(@PathVariable("userId") Long userId,@RequestBody Long[]
roleIds){
    List<SysUserRole> userRoleList=new ArrayList<>();
    Arrays.stream(roleIds).forEach(r -> {
        SysUserRole sysUserRole = new SysUserRole();
        sysUserRole.setRoleId(r);
        sysUserRole.setUserId(userId);
        userRoleList.add(sysUserRole);
    });
    sysUserRoleService.remove(new QueryWrapper<SysUserRole>
().eq("user_id",userId));
    sysUserRoleService.saveBatch(userRoleList);
    return R.ok();
}
```

角色管理实现

角色管理-根据条件列表分页显示实现

后端

```
/**
 * 根据条件分页查询角色信息
 * @param pageBean
 * @return
 */
@PostMapping("/list")
@PreAuthorize("hasAuthority('system:role:query')")
public R list(@RequestBody PageBean pageBean){
    String query=pageBean.getQuery().trim();
    Page<SysRole> pageResult = sysRoleService.page(new Page<>
(pageBean.getPageNum(), pageBean.getPageSize()),new QueryWrapper<SysRole>
().like(StringUtil.isNotEmpty(query),"name",query));
    List<SysRole> userList = pageResult.getRecords();
    Map<String,Object> resultMap=new HashMap<>();
    resultMap.put("roleList",userList);
    resultMap.put("total",pageResult.getTotal());
```

```
    return R.ok(resultMap);
}
```

前端 从user index.vue复制一份改改 user改成role User改成Role

```
<el-table :data="tableData" stripe style="width: 100%" @selection-
change="handleSelectionChange">
  <el-table-column type="selection" width="55" />

  <el-table-column prop="name" label="角色名"  width="100" align="center"/>

  <el-table-column prop="code" label="权限字符" width="200" align="center"/>

  <el-table-column prop="createTime" label="创建时间" width="200" align="center"/>

  <el-table-column prop="remark" label="备注"  />
  <el-table-column prop="action" label="操作" width="400" fixed="right"
align="center">
  <template v-slot="scope" >
    <el-button  type="primary" :icon="Tools"
@click="handleRoleDialogValue(scope.row.id,scope.row.sysRoleList)">分配权限</el-
button>

    <el-button v-if="scope.row.code!='admin'" type="primary" :icon="Edit"
@click="handleDialogValue(scope.row.id)" />
    <el-popconfirm  v-if="scope.row.code!='admin'" title="您确定要删除这条记录吗？"
@confirm="handleDelete(scope.row.id)">
      <template #reference>
        <el-button  type="danger" :icon="Delete" />
      </template>
    </el-popconfirm>
  </template>
</el-table-column>
</el-table>
```

角色管理-添加修改功能实现

后端:

```
/**
 * 添加或者修改
 * @param sysRole
 * @return
 */
@PostMapping("/save")
@PreAuthorize("hasAuthority('system:role:add')"+"||"+"hasAuthority('system:role:
edit')")
public R save(@RequestBody SysRole sysRole){
    if(sysRole.getId()==null || sysRole.getId()==-1){
        sysRole.setCreateTime(new Date());
        sysRoleService.save(sysRole);
    }else{
        sysRole.setUpdateTime(new Date());
```

```
        sysRoleService.updateById(sysRole);
    }
    return R.ok();
}
```

```
/**
 * 根据id查询
 * @param id
 * @return
 */
@GetMapping("/{id}")
@PreAuthorize("hasAuthority('system:role:query')")
public R findById(@PathVariable(value = "id")Integer id){
    SysRole sysRole = sysRoleService.getById(id);
    Map<String,Object> map=new HashMap<>();
    map.put("sysRole",sysRole);
    return R.ok(map);
}
```

前端:

dialog.vue

```
<template>

  <el-dialog
    v-model="dialogVisible"
    :title="dialogTitle"
    width="30%"
    @close="handleClose"
  >
       <el-form
        ref="formRef"
        :model="form"
        :rules="rules"
        label-width="100px"
    >
      <el-form-item label="角色名称" prop="name">
        <el-input v-model="form.name" />
      </el-form-item>

      <el-form-item label="权限字符" prop="code">
        <el-input v-model="form.code" />
      </el-form-item>


      <el-form-item label="备注" prop="remark">
        <el-input v-model="form.remark" type="textarea" :rows="4"/>
      </el-form-item>


    </el-form>
    <template #footer>
```

```
      <span class="dialog-footer">
        <el-button type="primary" @click="handleConfirm">确认</el-button>
        <el-button @click="handleClose"
          >取消</el-button
        >
      </span>
    </template>
  </el-dialog>

</template>

<script setup>
import {defineEmits, defineProps,ref,watch } from "vue"
import requestUtil,{getServerUrl} from "@/util/request";
import { ElMessage } from 'element-plus'

const props=defineProps(
    {
      id:{
        type:Number,
        default:-1,
        required:true
      },
      dialogTitle:{
        type:String,
        default:'',
        required:true
      },
      dialogVisible:{
        type:Boolean,
        default:false,
        required:true
      }
    }
)

const form=ref({
  id:-1,
  name:"",
  code:"",
  remark:""
})



const rules=ref({
  name:[
    { required: true, message: '请输入角色名称'}
  ],
  code:[
    { required: true, message: '请输入权限字符'}
  ]
})

const formRef=ref(null)

const initFormData=async(id)=>{
  const res=await requestUtil.get("sys/role/"+id);
```

```
  form.value=res.data.sysRole;
}

watch(
    ()=>props.dialogVisible,
    ()=>{
      let id=props.id;
      console.log("id="+id)
      if(id!=-1){
        initFormData(id);
      }else{
        form.value={
          id:-1,
          name:"",
          code:"",
          remark:""
        }
      }
    }
)

const emits=defineEmits(['update:modelValue','initRoleList'])

const handleClose=()=>{
  emits('update:modelValue',false)
}

const handleConfirm=()=>{
  formRef.value.validate(async(valid)=>{
    if(valid){
      let result=await requestUtil.post("sys/role/save",form.value);
      let data=result.data;
      if(data.code==200){
        ElMessage.success("执行成功！")
        formRef.value.resetFields();
        emits("initRoleList")
        handleClose();
      }else{
        ElMessage.error(data.msg);
      }
    }else{
      console.log("fail")
    }
  })
}

</script>

<style lang="scss" scoped>

</style>
```

角色管理-删除和批量删除功能实现

后端

```
/**
 * 删除
 * @param ids
 * @return
 */
@Transactional
@PostMapping("/delete")
@PreAuthorize("hasAuthority('system:role:delete')")
public R delete(@RequestBody Long[] ids){
    sysRoleService.removeByIds(Arrays.asList(ids));
    sysUserRoleService.remove(new QueryWrapper<SysUserRole>
().in("role_id",ids));
    return R.ok();
}
```

角色管理-显示权限菜单树

```
/**
 * 系统菜单Controller控制器
 * @author java1234_小锋 （公众号：java1234）
 * @site www.java1234.vip
 * @company 南通小锋网络科技有限公司
 */
@RestController
@RequestMapping("/sys/menu")
public class SysMenuController {

    @Autowired
    private SysMenuService sysMenuService;

    /**
     * 查询所有菜单树信息
     * @return
     */
    @RequestMapping("/list")
    @PreAuthorize("hasAuthority('system:menu:query')")
    public R list(){
        List<SysMenu> menuList = sysMenuService.list(new QueryWrapper<SysMenu>
().orderByAsc("order_num"));
        return R.ok().put("treeMenu",sysMenuService.buildTreeMenu(menuList));
    }
}
```

menuDialog

```
<template>
  <el-dialog
    model-value="menuDialogVisible"
    title="分配权限"
    width="30%"
```

```
    @close="handleClose"
  >

    <el-form
        ref="formRef"
        :model="form"
        label-width="100px"
    >

      <el-tree
          ref="treeRef"
          :data="treeData"
          :props="defaultProps"
          show-checkbox
          :default-expand-all=true
          node-key="id"
          :check-strictly=true
      />


    </el-form>

    <template #footer>
      <span class="dialog-footer">
        <el-button type="primary" @click="handleConfirm">确认</el-button>
        <el-button  @click="handleClose">取消</el-button>
      </span>
    </template>
  </el-dialog>
</template>

<script setup>

import {defineEmits, defineProps, ref, watch} from "vue";
import requestUtil,{getServerUrl} from "@/util/request";
import { ElMessage } from 'element-plus'

const defaultProps = {
  children: 'children',
  label: 'name'
}

  const props=defineProps(
      {
        id:{
          type:Number,
          default:-1,
          required:true
        },
        menuDialogVisible:{
          type:Boolean,
          default:false,
          required:true
        }
      }
  )
```

```
const form=ref({
  id:-1
})

const treeData=ref([])

const formRef=ref(null)

const initFormData=async(id)=>{
  const res=await requestUtil.get("sys/menu/list");
  treeData.value=res.data.treeMenu;
  form.value.id=id;
}


  watch(
      ()=>props.menuDialogVisible,
      ()=>{
        let id=props.id;
        console.log("id="+id)
        if(id!=-1){
          initFormData(id)
        }
      }
  )


  const emits=defineEmits(['update:modelValue','initRoleList'])

  const handleClose=()=>{
    emits('update:modelValue',false)
  }

  const handleConfirm=()=>{
    formRef.value.validate(async(valid)=>{
      if(valid){
          let result=await
requestUtil.post("sys/user/grantRole/"+form.value.id,form.value.checkedRoles);
          let data=result.data;
          if(data.code==200){
            ElMessage.success("执行成功！")

            emits("initUserList")
            handleClose();
          }else{
            ElMessage.error(data.msg);
          }
      }else{
        console.log("fail")
      }
    })
  }

</script>

<style scoped>

</style>
```

index.vue

```
prop="action" label="操作" width="400" fixed="right" align="center">
"scope" >
e="primary" :icon="Tools" @click="handleMenuDialogValue(scope.row.id)">分配权限</el-button>

="scope.row.code!='admin'" type="primary" :icon="Edit" @click="handleDialogValue(scope.row.id)" />
 v-if="scope.row.code!='admin'" title="您确定要删除这条记录吗？" @confirm="handleDelete(scope.row.id)">
```

```
<MenuDialog v-model="menuDialogVisible"  :menuDialogVisible="menuDialogVisible"
:id="id" @initRoleList="initRoleList"></MenuDialog>
```

```
import MenuDialog from './components/menuDialog'
```

```
const menuDialogVisible=ref(false);
```

```
const handleMenuDialogValue=(roleId)=>{
  if(roleId){
    id.value=roleId;
  }
  menuDialogVisible.value=true
}
```

角色管理-分配功能实现

后端:

```
/**
 * 获取当前角色的权限菜单ID集合
 * @param id
 * @return
 */
@GetMapping("/menus/{id}")
@PreAuthorize("hasAuthority('system:role:menu')")
public R menus(@PathVariable(value = "id")Integer id){
    List<SysRoleMenu> roleMenuList = sysRoleMenuService.list(new
QueryWrapper<SysRoleMenu>().eq("role_id", id));
    List<Long> menuIdList = roleMenuList.stream().map(p ->
p.getMenuId()).collect(Collectors.toList());
    return R.ok().put("menuIdList",menuIdList);
}

/**
 * 更新角色权限信息
 * @param id
 * @param menuIds
 * @return
 */
@Transactional
@PostMapping("/updateMenus/{id}")
```

```
@PreAuthorize("hasAuthority('system:role:menu')")
public R updateMenus(@PathVariable(value = "id")Long id,@RequestBody Long[]
menuIds){
    sysRoleMenuService.remove(new QueryWrapper<SysRoleMenu>().eq("role_id",
id));
    List<SysRoleMenu> sysRoleMenuList=new ArrayList<>();
    Arrays.stream(menuIds).forEach(menuId->{
        SysRoleMenu roleMenu=new SysRoleMenu();
        roleMenu.setRoleId(id);
        roleMenu.setMenuId(menuId);
        sysRoleMenuList.add(roleMenu);
    });
    sysRoleMenuService.saveBatch(sysRoleMenuList);
    return R.ok();
}
```

前端:

```
<template>
  <el-dialog
    model-value="menuDialogVisible"
    title="分配权限"
    width="30%"
    @close="handleClose"
  >

    <el-form
        ref="formRef"
        :model="form"
        label-width="100px"
    >

      <el-tree
          ref="treeRef"
          :data="treeData"
          :props="defaultProps"
          show-checkbox
          :default-expand-all=true
          node-key="id"
          :check-strictly=true
      />


    </el-form>

    <template #footer>
      <span class="dialog-footer">
        <el-button type="primary" @click="handleConfirm">确认</el-button>
        <el-button  @click="handleClose">取消</el-button>
      </span>
    </template>
  </el-dialog>
</template>

<script setup>
```

```
import {defineEmits, defineProps, ref, watch} from "vue";
import requestUtil,{getServerUrl} from "@/util/request";
import { ElMessage } from 'element-plus'

const defaultProps = {
  children: 'children',
  label: 'name'
}

  const props=defineProps(
      {
        id:{
          type:Number,
          default:-1,
          required:true
        },
        menuDialogVisible:{
          type:Boolean,
          default:false,
          required:true
        }
      }
  )


const form=ref({
  id:-1
})

const treeData=ref([])

const formRef=ref(null)

const treeRef=ref(null)

const initFormData=async(id)=>{
  const res=await requestUtil.get("sys/menu/list");
  treeData.value=res.data.treeMenu;

  form.value.id=id;

  const res2=await requestUtil.get("sys/role/menus/"+id);
  treeRef.value.setCheckedKeys(res2.data.menuIdList);

}


  watch(
      ()=>props.menuDialogVisible,
      ()=>{
        let id=props.id;
        console.log("id="+id)
        if(id!=-1){
          initFormData(id)
        }
      }
  )
```

```
  const emits=defineEmits(['update:modelValue','initRoleList'])

  const handleClose=()=>{
    emits('update:modelValue',false)
  }

  const handleConfirm=()=>{
    formRef.value.validate(async(valid)=>{
      if(valid){
          var menuIds=treeRef.value.getCheckedKeys();
          let result=await
requestUtil.post("sys/role/updateMenus/"+form.value.id,menuIds);
          let data=result.data;
          if(data.code==200){
            ElMessage.success("执行成功！")

            emits("initRoleList")
            handleClose();
          }else{
            ElMessage.error(data.msg);
          }
      }else{
        console.log("fail")
      }
    })
  }

</script>

<style scoped>

</style>
```

菜单管理实现

菜单管理-树形表格菜单信息显示

```
<template>
  <div class="app-container">

    <el-row :gutter="20" class="header">

      <el-button type="success" :icon="DocumentAdd"
@click="handleDialogValue()">新增</el-button>

    </el-row>

      <el-table
          :data="tableData"
          row-key="id"
          stripe
```

```
          style="width: 100%; margin-bottom: 20px"
          border
          default-expand-all
          :tree-props="{ children: 'children', hasChildren: 'hasChildren' }"
      >

        <el-table-column prop="name" label="菜单名称"  width="200"/>
        <el-table-column prop="icon" label="图标"  width="70" align="center">
          <template v-slot="scope">
            <el-icon><svg-icon :icon="scope.row.icon" /></el-icon>
          </template>
        </el-table-column>
        <el-table-column prop="orderNum" label="排序"  width="70"
align="center"/>
        <el-table-column prop="perms" label="权限标识"  width="200" />
        <el-table-column prop="path" label="组件路径"  width="180" />
        <el-table-column prop="menuType" label="菜单类型"  width="120"
align="center">
          <template v-slot="scope">
            <el-tag size="small" v-if="scope.row.menuType === 'M'" type="danger"
effect="dark">目录</el-tag>
            <el-tag size="small" v-else-if="scope.row.menuType === 'C'"
type="success" effect="dark">菜单</el-tag>
            <el-tag size="small" v-else-if="scope.row.menuType === 'F'"
type="warning" effect="dark">按钮</el-tag>
          </template>
        </el-table-column>
        <el-table-column prop="createTime" label="创建时间" align="center"/>
        <el-table-column prop="action" label="操作" width="400" fixed="right"
align="center">
        <template v-slot="scope" >
          <el-button  type="primary" :icon="Edit"
@click="handleDialogValue(scope.row.id)" />
          <el-popconfirm  title="您确定要删除这条记录吗？"
@confirm="handleDelete(scope.row.id)">
            <template #reference>
              <el-button  type="danger" :icon="Delete" />
            </template>
          </el-popconfirm>
        </template>
      </el-table-column>
      </el-table>

  </div>
  <Dialog v-model="dialogVisible" :dialogVisible="dialogVisible" :id="id"
:dialogTitle="dialogTitle" @initUserList="initUserList"/>


</template>

<script setup>
import { Search ,Delete,DocumentAdd ,Edit, Tools, RefreshRight} from '@element-
plus/icons-vue'
import requestUtil,{getServerUrl} from "@/util/request";
import { ref } from 'vue'
import Dialog from './components/dialog'
import { ElMessage, ElMessageBox } from 'element-plus'
```

```
const tableData=ref([]);

const total=ref(0)

const dialogVisible=ref(false)

const dialogTitle=ref("")

const id=ref(-1)



const initMenuList=async()=>{
  const res=await requestUtil.post("sys/menu/list");
  tableData.value=res.data.treeMenu;
  total.value=res.data.total;
}

initMenuList();

const handleDialogValue=(userId)=>{
  if(userId){
    id.value=userId;
    dialogTitle.value="用户修改"
  }else{
    id.value=-1;
    dialogTitle.value="用户添加"
  }
  dialogVisible.value=true
}


const handleDelete=async (id)=>{
  var ids = []
  if(id){
    ids.push(id)
  }else{
    multipleSelection.value.forEach(row=>{
      ids.push(row.id)
    })
  }
  const res=await requestUtil.post("sys/user/delete",ids)
  if(res.data.code==200){
    ElMessage({
      type: 'success',
      message: '执行成功!'
    })
    initUserList();
  }else{
    ElMessage({
      type: 'error',
      message: res.data.msg,
    })
  }
}
```

```
</script>

<style lang="scss" scoped>

.header{
  padding-bottom: 16px;
  box-sizing: border-box;
}

.el-pagination{
  float: right;
  padding: 20px;
  box-sizing: border-box;
}

::v-deep th.el-table__cell{
  word-break: break-word;
  background-color: #f8f8f9 !important;
  color: #515a6e;
  height: 40px;
  font-size: 13px;

}

.el-tag--small {
  margin-left: 5px;
}
</style>
```

菜单管理-添加修改功能实现

后端:

```
/**
 * 添加或者修改
 * @param sysMenu
 * @return
 */
@PostMapping("/save")
@PreAuthorize("hasAuthority('system:menu:add')"+"||"+"hasAuthority('system:menu:edit')")
public R save(@RequestBody SysMenu sysMenu){
    if(sysMenu.getId()==null || sysMenu.getId()==-1){
        sysMenu.setCreateTime(new Date());
        sysMenuService.save(sysMenu);
    }else{
        sysMenu.setUpdateTime(new Date());
        sysMenuService.updateById(sysMenu);
    }
    return R.ok();
}
```

```java
/**
 * 根据id查询
 * @param id
 * @return
 */
@GetMapping("/{id}")
@PreAuthorize("hasAuthority('system:menu:query')")
public R findById(@PathVariable(value = "id")Long id){
    SysMenu sysMenu = sysMenuService.getById(id);
    Map<String,Object> map=new HashMap<>();
    map.put("sysMenu",sysMenu);
    return R.ok(map);
}
```

前端dialog

```vue
<template>
  <el-dialog
    model-value="dialogVisible"
    :title="dialogTitle"
    width="30%"
  @close="handleClose"
  >

    <el-form
        ref="formRef"
        :model="form"
        :rules="rules"
        label-width="100px"
    >

      <el-form-item label="上级菜单" prop="parentId">
        <el-select v-model="form.parentId" placeholder="请选择上级菜单">
          <template v-for="item in tableData">
            <el-option :label="item.name" :value="item.id"></el-option>
            <template v-for="child in item.children">
              <el-option :label="child.name" :value="child.id">
                <span>{{ "    -- " + child.name }}</span>
              </el-option>
            </template>
          </template>
        </el-select>
      </el-form-item>

      <el-form-item label="菜单类型" prop="menuType" label-width="100px">
     <el-radio-group v-model="form.menuType">
      <el-radio :label="'M'">目录</el-radio>
      <el-radio :label="'C'">菜单</el-radio>
      <el-radio :label="'F'">按钮</el-radio>
     </el-radio-group>
    </el-form-item>

      <el-form-item label="菜单图标" prop="icon">
        <el-input v-model="form.icon" />
```

```
      </el-form-item>

      <el-form-item label="菜单名称" prop="name">
        <el-input v-model="form.name" />
      </el-form-item>

      <el-form-item label="权限标识" prop="perms">
        <el-input v-model="form.perms" />
      </el-form-item>

      <el-form-item label="组件路径" prop="component">
        <el-input v-model="form.component" />
      </el-form-item>

      <el-form-item label="显示顺序" prop="orderNum" >
     <el-input-number v-model="form.orderNum" :min="1" label="显示顺序"></el-
input-number>
    </el-form-item>



    </el-form>
    <template #footer>
      <span class="dialog-footer">
        <el-button type="primary" @click="handleConfirm">确认</el-button>
        <el-button  @click="handleClose">取消</el-button>
      </span>
    </template>
  </el-dialog>
</template>

<script setup>

import {defineEmits, defineProps, ref, watch} from "vue";
import requestUtil,{getServerUrl} from "@/util/request";
import { ElMessage } from 'element-plus'

  const tableData=ref([])


  const props=defineProps(
      {
        id:{
          type:Number,
          default:-1,
          required:true
        },
        dialogTitle:{
          type:String,
          default:'',
          required:true
        },
        dialogVisible:{
          type:Boolean,
          default:false,
          required:true
        },
        tableData:{
```

```
          type:Array,
          default:[],
          required:true
        }
      }
  )


const form=ref({
  id:-1,
  parentId:'',
  menuType:"M",
  icon:'',
  name:'',
  perms:'',
  component:'',
  orderNum:1
})


const rules=ref({
  parentId:[
    { required: true, message: '请选择上级菜单'}
  ],
  name: [{ required: true, message: "菜单名称不能为空", trigger: "blur" }]
})

const formRef=ref(null)

const initFormData=async(id)=>{
 const res=await requestUtil.get("sys/menu/"+id);
 form.value=res.data.sysMenu;
}



  watch(
      ()=>props.dialogVisible,
      ()=>{
        let id=props.id;
        tableData.value=props.tableData;
        if(id!=-1){
          initFormData(id)
        }else{
          form.value={
            id:-1,
            parentId:'',
            menuType:"M",
            icon:'',
            name:'',
            perms:'',
            component:'',
            orderNum:1
          }
        }
      }
  )
```

```
  const emits=defineEmits(['update:modelValue','initMenuList'])

  const handleClose=()=>{
    emits('update:modelValue',false)
  }

  const handleConfirm=()=>{
    formRef.value.validate(async(valid)=>{
      if(valid){
          let result=await requestUtil.post("sys/menu/save",form.value);
          let data=result.data;
          if(data.code==200){
            ElMessage.success("执行成功！")
            formRef.value.resetFields();
            emits("initMenuList")
            handleClose();
          }else{
            ElMessage.error(data.msg);
          }
      }else{
        console.log("fail")
      }
    })
  }

</script>

<style scoped>

</style>
```

```
<Dialog v-model="dialogVisible" :tableData="tableData"
:dialogVisible="dialogVisible" :id="id" :dialogTitle="dialogTitle"
@initMenuList="initMenuList"/>
```

菜单管理-删除功能实现

后端:

```
/**
 * 删除
 * @param id
 * @return
 */
@GetMapping("/delete/{id}")
@PreAuthorize("hasAuthority('system:menu:delete')")
public R delete(@PathVariable(value = "id")Long id){
    int count = sysMenuService.count(new QueryWrapper<SysMenu>().eq("parent_id",
id));
    if(count>0){
```

```
            return R.error("请先删除子菜单！");
        }
        sysMenuService.removeById(id);
        return R.ok();
    }
```

前端：

```
const handleDelete=async (id)=>{

  const res=await requestUtil.get("sys/menu/delete/"+id)
  if(res.data.code==200){
    ElMessage({
      type: 'success',
      message: '执行成功!'
    })
    initMenuList();
  }else{
    ElMessage({
      type: 'error',
      message: res.data.msg,
    })
  }
}
```

进阶优化功能实现（VIP）

验证码功能实现

权限加载实现redis缓存

动态控制页面操作按钮显示和隐藏

后端字段验证实现