SpringBoot + vue + shiro登录、权限认证

一、需求

- 登录：不同的用户具有不同的角色，访问不同的资源
- 权限：不同的用户左边展示的菜单栏不太一样，动态路由进行非法资源拦截

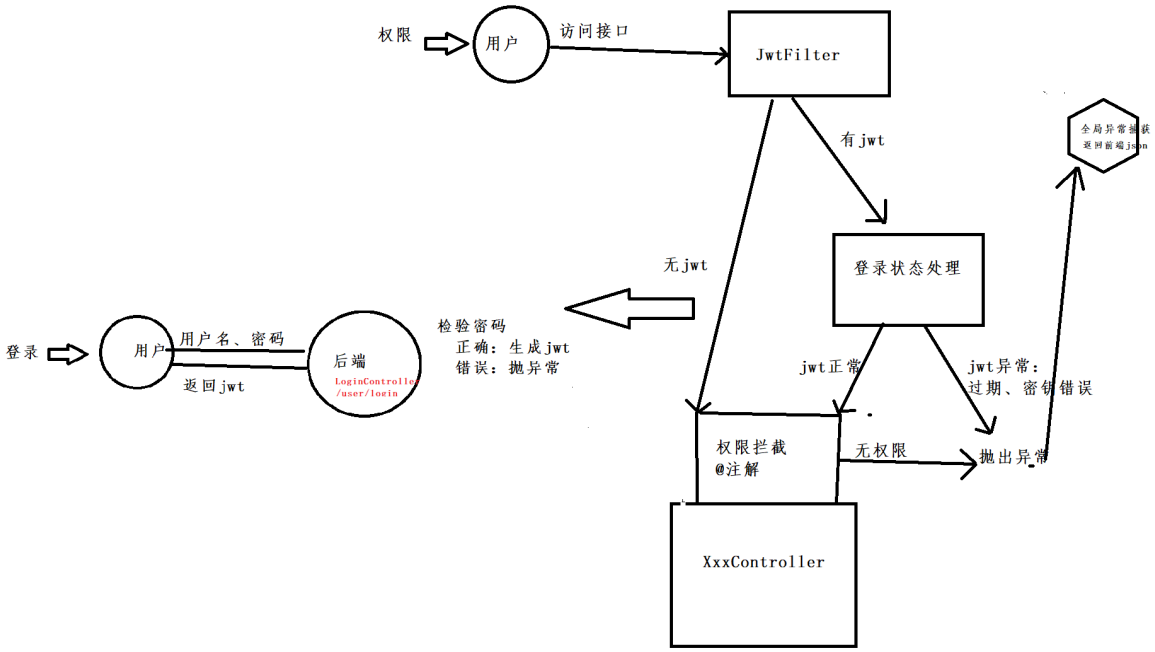
二、分析

- 登录：输入账号、密码请求后端验证，验证通过，服务端返回一个token,拿到token,存储到cookie中，前端会根据token再去拉取一个 **user_info** 的接口来获取用户的详细信息（如用户权限，用户名等等信息）。
- 权限：后端返回用户的信息，前端通过存储在cookie中的token和role信息进行页面显示和权限控制
    - 前端：动态根据用户的role算出显示的页面
    - 前端：动态根据用户的role算出其对应权限的路由，通过 **router.addRoutes** 动态挂载这些路由。
    - 后端：shiro注解权限控制

三、实现

3.1 后端登录实现

SpingBoot + Vue + Shiro登录认证、权限授予



一、shiro相关实现

- `ShiroConfig`配置类：
  - 配置`ShiroFilter`---》实现所有资源认证、授权拦截过滤
  - 配置`SessionManager`---》使用redis存储Session信息
  - 配置`SessionsSecurityManager`---》关联Realm、SessionManager、CacheManager组件
  - 配置`ShiroFilterChainDefinition`---》哪些链接需要经过哪些过滤器
  - 配置`ShiroFilterFactoryBean`---》装配自己的拦截器，设置认证、权限拦截的资源链接
- `ShiroFilter`过滤器类
  - 重写`preHandle`方法---》跨域处理，因为访问XxxController之前需要经过过滤器，放行Options的请求
  - 重写`onAccessDenied`方法---》该方法在用户未登录的状态下会自动执行
    - 第一步：第一次登陆访问`LoginController`，直接放行到`LoginController`，在那里获得`token`；
    - 第二步：接着前端会携带`token`访问i`nfo接口`拉取用户信息，此时需要校验token有效性，有效的话接着执行`shiro`的`executeLogin`登录方法
    - 第三步：`executeLogin`方法会
      - 调用我们需要重写的 createToken 方法，该方法会从Header中获取token
      - 调用UserRealm中的认证方法，失败的话，调用我们需要重写的`onLoginFailure`
- `UserRealm`认证、授权逻辑类
  - doGetAuthenticationInfo（）认证方法
  - doGetAuthenticationInfo（）授权方法

二、LoginController相关实现

- `/user/login`接口：未存在token可以被shiro放行直接访问，主要根据JwtUtil根据userNumber生成token，存放到Header中，返回到Response中，然后前端取出来，设置到每次访问的request中
- `/user/info`接口：拉取用户信息的接口，获取用户角色、头像、学院等相关信息，根据token转化为userNumber查询数据库
- `/user/logout`接口:注销登录操作，直接使用shiro的`subject.logout()`即可

3.2前端登录实现

一、取出来访问后端Controller得到token

2021/1/28 18:43:24
1. auth.js

```
import Cookies from 'js-cookie'

const TokenKey = 'token'

export function getToken() {
  return Cookies.get(TokenKey)
}

export function setToken(token) {
  return Cookies.set(TokenKey, token)
}

export function removeToken() {
  return Cookies.remove(TokenKey)
}
```

2021/1/28 18:49:34
3. user.js
将Controller返回的response中的token取出来setToken()　　到Cookie中

```
const actions = {
  // user login
  login({ commit }, userInfo) {
    const { userNumber, password } = userInfo
    return new Promise((resolve, reject) => {
      login({ userNumber: userNumber.trim(), password: password }).then(response => {
        const { data } = response
        commit('SET_TOKEN', data.token)
        setToken(data.token)
        resolve()
      }).catch(error => {
        reject(error)
      })
    })
  },
```

二、请求拦截器将token取出来加到每此的请求中

2021/1/28 18:44:49

2. request拦截器

将每次请求的token获取出来，设置到header

```
// request interceptor
service.interceptors.request.use(
  config => {
    // do something before request is sent

    if (store.getters.token) {
      // let each request carry token
      // ['X-Token'] is a custom headers key
      // please modify it according to the actual situation
      config.headers['token'] = getToken()
    }
    return config
  },
  error => {
    // do something with request error
    console.log(error) // for debug
    return Promise.reject(error)
  }
)
```

3.3后端授权实现

一、主要是shiro的UserRealm认证、授权类来实现

- 认证的时候，将当前用户存入`new SimpleAuthenticationInfo(currentUser, shiroToken.getCredentials(),getName());`
- 授权的时候，将当前用户取出来，查询数据库对应的role字段，加入对应的角色权限，然后Controller使用注解拦截即可

```
    SimpleAuthorizationInfo info = new SimpleAuthorizationInfo();

        //拿到当前登录的对象
        Subject subject = SecurityUtils.getSubject();
        User currentUser = (User)subject.getPrincipal();


        //拿出来user表中Role字段为当前用户添加角色
        info.addRole(currentUser.getRole());
        return info;
```

3.4前端授权实现

主要是两方面

- 前端获取用户的role信息实现不同用户资源展示
- 前端动态挂载路由拦截用户非法请求

一、不同用户role显示不同的菜单资源

二、主要是根据当前用户的role属性来动态挂载路由