Vue.js前端权限控制方案

本方案的实现方式是通过获取当前用户的权限去比对路由表，生成当前用户权限可访问的路由表，通过router.addRoutes动态挂载到router上。不同的权限对应着不同的路由，异步生成。

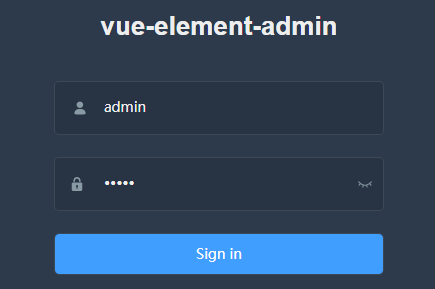
要实现这个方案，主要分为两部分：登录和权限验证。实现的大概思路如下。

**登录**：当用户填写完账号和密码后向服务端验证是否正确，验证通过之后，服务端会返回一个 token（该token的是一个能唯一标示用户身份的一个key），之后我们将token存储在本地cookie之中，这样下次打开页面或者刷新页面的时候就能记住用户的登录状态，不用再去登录页面重新登录了。

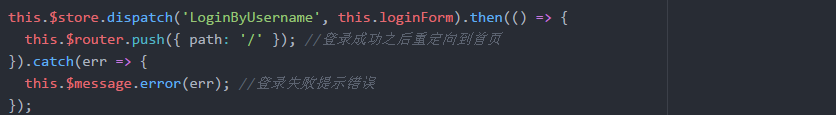
**权限验证**：用户登录成功之后，我们会在全局钩子router.beforeEach中拦截路由，判断是否已获得token。通过token获取用户对应的 role，动态根据用户的 role 算出其对应有权限的路由，通过 router.addRoutes 动态挂载这些路由。

**登录**

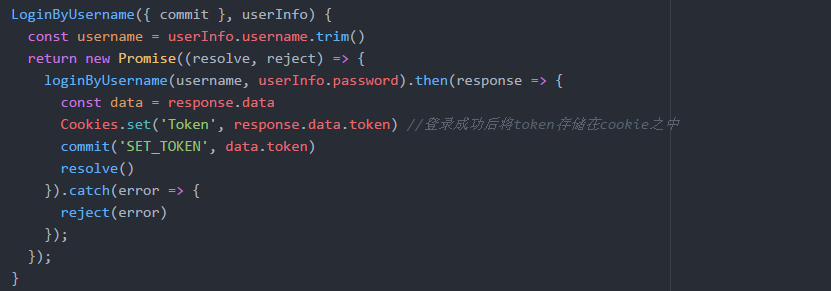
以vue-admin-template为例。



点击登录按钮触发操作：

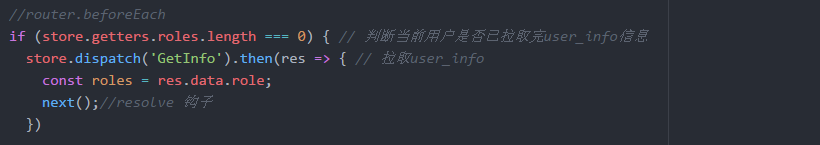


LoginByUsername action:



登录成功后，服务端会返回一个 token（该token的是一个能唯一标示用户身份的一个key），之后我们将token存储在本地cookie之中，这样下次打开页面或者刷新页面的时候能记住用户的登录状态，不用再去登录页面重新登录了。

用户登录成功之后，我们会在全局钩子router.beforeEach中拦截路由，判断是否已获得token，在获得token之后我们就要去获取用户的基本信息了



就像前面所说的，我们只在本地存储了一个用户的token，并没有存储别的用户信息（如用户权限，用户名，用户头像等）。没有保存其它的用户信息主要出于如下的考虑：

假设我们把用户权限和用户名也存在了本地，但我这时候用另一台电脑登录修改了自己的用户名，之后再用这台存有之前用户信息的电脑登录，它默认会去读取本地 cookie 中的名字，并不会去拉去新的用户信息。

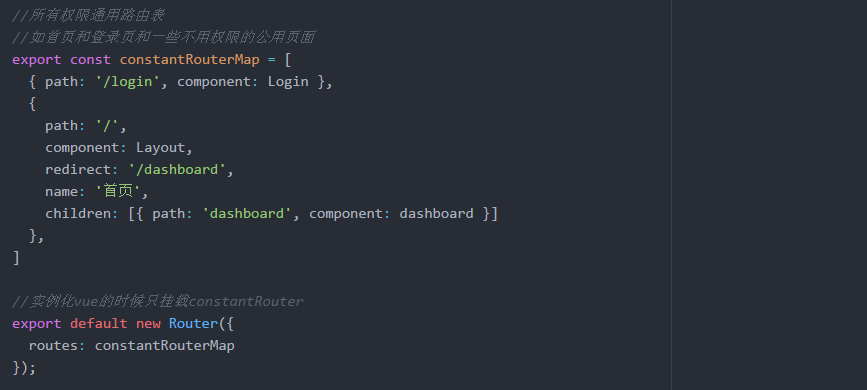
所以现在的策略是：页面会先从 cookie 中查看是否存有 token，没有，就走一遍上一部分的流程重新登录，如果有token,就会把这个 token 返给后端去拉取user\_info，保证用户信息是最新的。

**权限验证**

权限控制的主体思路是，前端会有一份路由表，它表示了每一个路由可访问的权限。当用户登录之后，通过 token 获取用户的 role ，动态根据用户的 role 算出其对应有权限的路由，再通过router.addRoutes动态挂载路由。具体步骤如下：

1. 创建vue实例的时候将vue-router挂载，但这个时候vue-router挂载一些登录或者不用权限的公用的页面。
2. 当用户登录后，获取用role，将role和路由表每个页面的需要的权限作比较，生成最终用户可访问的路由表。
3. 调用router.addRoutes(store.getters.addRouters)添加用户可访问的路由。

在router.js路由表中的具体操作：

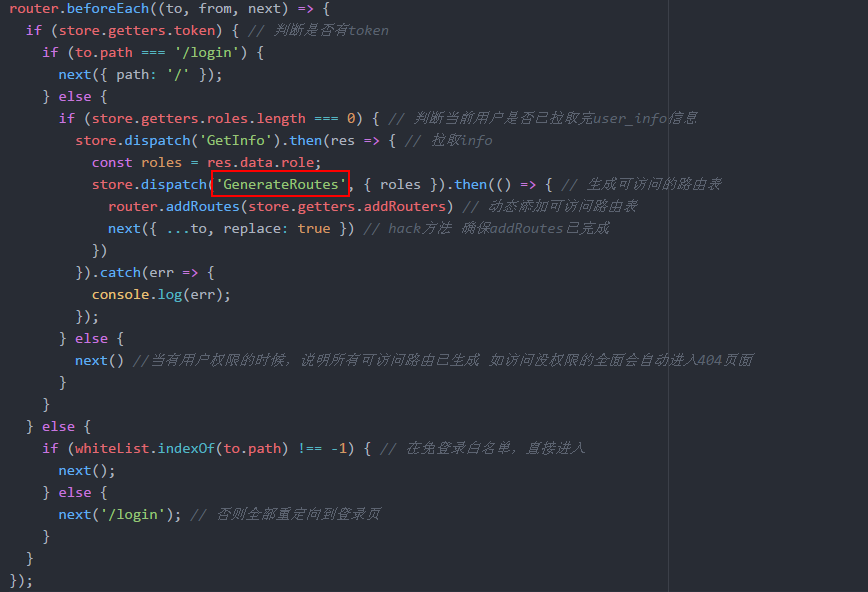




这里我们根据 vue-router官方推荐的方法通过meta标签来标示改页面能访问的权限有哪些。如meta: { role: ['admin','super\_editor'] }表示该页面只有admin和超级编辑才能有资格进入。

**注意事项**：这里有一个需要非常注意的地方就是 404 页面一定要最后加载，如果放在constantRouterMap一同声明了404，后面的所以页面都会被拦截到404。

在main.js中的具体操作：



先判断是否有token，如果没有就只能进入免登陆的白名单页面，如登录页面。如果有token且目标不是登录页面，则判断用户信息是否已经取到，取到后通过GenerateRoutes action去生成可访问的路由表，然后动态添加到router中。

GenerateRoutes action中的具体操作：



在GenerateRoutes action中就干了一件事，通过用户的权限和之前在router.js里面asyncRouterMap的每一个页面所需要的权限做匹配，最后返回该用户能够访问路由有哪些。