Vue权限控制

在Web系统中, 权限很久以来一直都只是后端程序所控制的.为什么呢? 因为Web系统的本质围绕的是数据, 而和数据库最紧密接触的是后端程序.所以在很长的一段时间内, 权限一直都只是后端程序需要考虑的话题.但是随着前后端分离架构的流行, 越来越多的项目也在前端进行权限控制.

1.权限相关概念

1.1.权限的分类

- 后端权限

  从根本上讲前端仅仅只是视图层的展示, 权限的核心是在于服务器中的数据变化, 所以后端才是权限的关键, 后端权限可以控制某个用户是否能够查询数据, 是否能够修改数据等操作

  - 后端如何知道该请求是哪个用户发过来的

    ```
    cookie
    session
    token
    ```

  - 后端的权限设计RBAC

    ```
    用户
    角色
    权限
    ```

- 前端权限

  前端权限的控制本质上来说, 就是控制前端的 **视图层的展示**和前端所发送的**请求**. 但是只有前端权限控制没有后端权限控制是万万不可的. 前端权限控制只可以说是达到锦上添花的效果.

1.2.前端权限的意义

如果仅从能够修改服务器中数据库中的数据层面上讲， 确实只在后端做控制就足够了, 那为什么越来越多的项目也进行了前端权限的控制, 主要有这几方面的好处

- 降低非法操作的可能性

  不怕贼偷就怕贼惦记, 在页面中展示出一个 就算点击了也最终会失败 的按钮, 势必会增加有心者非法操作的可能性

- 尽可能排除不必要请求,减轻服务器压力

  没必要的请求, 操作失败的请求, 不具备权限的请求, 应该压根就不需要发送, 请求少了, 自然也会减轻服务器的压力

- 提高用户体验

根据用户具备的权限为该用户展现自己权限范围内的内容，避免在界面上给用户带来困扰, 让用户专注于分内之事

2.前端权限控制思路

2.1.菜单的控制

在登录请求中, 会得到权限数据, 当然, 这个需要后端返回数据的支持. 前端根据权限数据, 展示对应的菜单.点击菜单,才能查看相关的界面.

2.2.界面的控制

如果用户没有登录,手动在地址栏敲入管理界面的地址, 则需要跳转到登录界面

如果用户已经登录, 可是手动敲入非权限内的地址, 则需要跳转404界面

2.3.按钮的控制

在某个菜单的界面中, 还得根据权限数据, 展示出可进行操作的按钮, 比如删除,修改,增加

2.4.请求和响应的控制

如果用户通过非常规操作, 比如通过浏览器调试工具将某些禁用的按钮变成启用状态, 此时发的请求, 也应该被前端所拦截

3. Vue的权限控制实现

3.1.菜单的控制

- 查看登录之后获取到的数据


```
{
  "data": {
    "id": 500,
    "rid": 0,
    "username": "admin",
    "mobile": "13999999999",
    "email": "123999@qq.com",
    "token": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOjUwMCwicmlkIjowLCJpYXQiOjE1M
TI1NDQyOTksImV4cCI6MTUxMjYzMDY5OX0.eGrsrvwHm-
tPsO9r_pxHIQ5i5L1kX9RX444uwnRGaIM"
  },
  "rights": [{
    "id": 125,
    "authName": "用户管理",
    "icon": "icon-user",
    "children": [{
      "id": 110,
      "authName": "用户列表",
      "path": "users",
```

```
      "rights": ["view", "edit", "add", "delete"]
    }]
  }, {
    "id": 103,
    "authName": "角色管理",
    "icon": "icon-tijikongjian",
    "children": [{
      "id": 111,
      "authName": "角色列表",
      "path": "roles",
      "rights": ["view", "edit", "add", "delete"]
    }]
  }, {
    "id": 101,
    "authName": "商品管理",
    "icon": "icon-shangpin",
    "children": [{
      "id": 104,
      "authName": "商品列表",
      "path": "goods",
      "rights": ["view", "edit", "add", "delete"]
    }, {
      "id": 121,
      "authName": "商品分类",
      "path": "categories",
      "rights": ["view", "edit", "add", "delete"]
    }]
  }],
  "meta": {
    "msg": "登录成功",
    "status": 200
  }
}
```


在这部分数据中, 除了该用户的基本信息之外, 还有两个字段很关键

- token,用于前端用户的状态保持
- rights:该用户具备的权限数据,一级权限就对应一级菜单,二级权限就对应二级菜单

- 根据rights中的数据, 动态渲染左侧菜单栏, 数据在Login.vue得到, 但是在Home.vue才使用, 所以可以把数据用vuex进行维护
  - vuex中的代码

```
export default new Vuex.Store({
  state: {
    rightList:[]
  },
  mutations: {
    setRightList(state, data) {
      state.rightList = data
```

```
    }
  },
  actions: {
  },
  getters: {
  }
})
```

- Login.vue的代码

```
login() {
    this.$refs.loginFormRef.validate(async valid => {
        ......
        this.$store.commit('setRightList', res.rights)
        this.$message.success('登录成功')
        this.$router.push('/home')
    })
}
```

- Home.vue的代码

```
import { mapState } from 'vuex'
computed: {
    ...mapState(['rightList'])
}
created() {
    this.activePath = window.sessionStorage.getItem('activePath')
    this.menulist = this.rightList
},
```

- 刷新界面菜单消失
  - 原因分析

```
因为菜单数据是登录之后才获取到的，获取菜单数据之后,就存放在Vuex中
一旦刷新界面，Vuex中的数据会重新初始化，所以会变成空的数组
因此，需要将权限数据存储在sessionStorage中，并让其和Vuex中的数据保持同步
```

  - 代码解决

```
export default new Vuex.Store({
  state: {
    rightList:JSON.parse(sessionStorage.getItem('rightList')||'[]')
  },
  mutations: {
    setRightList(state, data) {
      state.rightList = data
      sessionStorage.setItem('rightList',JSON.stringify(data))
```

```
    }
  },
  actions: {
  },
  getters: {
  }
})
```

- 标识用户名, 方便查看当前用户
  - vuex的代码

```
export default new Vuex.Store({
  state: {
    rightList:JSON.parse(sessionStorage.getItem('rightList')||'[]'),
    username: sessionStorage.getItem('username')
  },
  mutations: {
    setRightList(state, data) {
      state.rightList = data
      sessionStorage.setItem('rightList',JSON.stringify(data))
    },
    setUsername(state, data) {
      state.username = data
      sessionStorage.setItem('username',data)
    }
  },
  actions: {
  },
  getters: {
  }
})
```

  - Login.vue的代码

```
login() {
      this.$refs.loginFormRef.validate(async valid => {
        ......
        this.$store.commit('setRightList', res.rights)
        this.$store.commit('setUsername', res.data.username)
        this.$message.success('登录成功')
        this.$router.push('/home')
      })
    }
```

  - Home.vue的代码

```
computed: {
    ...mapState(['rightList','username'])
}
<el-button type="info" @click="logout">{{username}}退出</el-button>
```

- 退出按钮的逻辑

```
logout() {
    sessionStorage.clear()
    this.$router.push('/login')
    window.location.reload()
},
```

3.2.界面的控制

1.正常的逻辑是通过登录界面, 登录成功之后跳转到管理平台界面, 但是如果用户直接敲入管理平台的地址, 也是可以跳过登录的步骤.所以应该在某个时机判断用户是否登录

- 如何判断是否登录

```
sessionStorage.setItem('token', res.data.token)
```

- 什么时机
  - 路由导航守卫

```
router.beforeEach((to, from, next) => {
  if (to.path === '/login') {
    next()
  } else {
    const token = sessionStorage.getItem('token')
    if(!token) {
      next('/login')
    } else {
      next()
    }
  }
})
```

2.虽然菜单项已经被控制住了, 但是路由信息还是完整的存在于浏览器,正比如zhangsan这个用户并不具备角色这个菜单, 但是他如果自己在地址栏中敲入/roles的地址, 依然也可以访问角色界面

- 路由导航守卫

  路由导航守卫固然可以在每次路由地址发生变化的时候, 从vuex中取出rightList判断用户将要访问的界面, 这个用户到底有没有权限.不过从另外一个角度来说,这个用户不具备权限的路由, 是否也应该压根就不存在呢?

- 动态路由

- 登录成功之后动态添加
- App.vue中添加
- 代码如下:
  - router.js

```
import Vue from 'vue'
import Router from 'vue-router'
import Login from '@/components/Login.vue'
import Home from '@/components/Home.vue'
import Welcome from '@/components/Welcome.vue'
import Users from '@/components/user/Users.vue'
import Roles from '@/components/role/Roles.vue'
import GoodsCate from '@/components/goods/GoodsCate.vue'
import GoodsList from '@/components/goods/GoodsList.vue'
import NotFound from '@/components/NotFound.vue'
import store from '@/store'

Vue.use(Router)

const userRule = { path: '/users', component: Users }
const roleRule = { path: '/roles', component: Roles }
const goodsRule = { path: '/goods', component: GoodsList }
const categoryRule = { path: '/categories', component: GoodsCate }

const ruleMapping = {
  'users': userRule,
  'roles': roleRule,
  'goods': goodsRule,
  'categories': categoryRule
}

const router = new Router({
  routes: [
    {
      path: '/',
      redirect: '/home'
    },
    {
      path: '/login',
      component: Login
    },
    {
      path: '/home',
      component: Home,
      redirect: '/welcome',
      children: [
        { path: '/welcome', component: Welcome },
        // { path: '/users', component: Users },
```

```
        // { path: '/roles', component: Roles },
        // { path: '/goods', component: GoodsList },
        // { path: '/categories', component: GoodsCate }
      ]
    },
    {
      path: '*',
      component: NotFound
    }
  ]
})

router.beforeEach((to, from, next) => {
  if (to.path === '/login') {
    next()
  } else {
    const token = sessionStorage.getItem('token')
    if(!token) {
      next('/login')
    } else {
      next()
    }
  }
})

export function initDynamicRoutes() {
  const currentRoutes = router.options.routes
  const rightList = store.state.rightList
  rightList.forEach(item => {
    item.children.forEach(item => {
      currentRoutes[2].children.push(ruleMapping[item.path])
    })
  })
  router.addRoutes(currentRoutes)
}

export default router
```

- Login.vue

```
import { initDynamicRoutes } from '@/router.js'
login() {
      this.$refs.loginFormRef.validate(async valid => {
        if (!valid) return
        const { data: res } = await this.$http.post('login',
this.loginForm)
        if (res.meta.status !== 200) return
this.$message.error('登录失败! ')
```

```
          this.$store.commit('setRightList', res.rights)
          this.$store.commit('setUsername', res.data.username)
          sessionStorage.setItem('token', res.data.token)
          initDynamicRoutes()
          this.$message.success('登录成功')
          this.$router.push('/home')
        })
      }
```

- App.vue

```
import { initDynamicRoutes } from '@/router.js'
export default {
  name: 'app',
  created() {
    initDynamicRoutes()
  }
}
```

3.3.按钮的控制

按钮控制

虽然用户可以看到某些界面了, 但是这个界面的一些按钮,该用户可能是没有权限的.因此, 我们需要对组件中的一些按钮进行控制. 用户不具备权限的按钮就隐藏或者禁用, 而在这块中, 可以把该逻辑放到自定义指令中

- permission.js

```
import Vue from 'vue'
import router from '@/router.js'
Vue.directive('permission', {
  inserted: function(el, binding){
    const action = binding.value.action
    const currentRight = router.currentRoute.meta
    if(currentRight) {
      if(currentRight.indexOf(action) == -1) {
        // 不具备权限
        const type = binding.value.effect
        if(type === 'disabled') {
          el.disabled = true
          el.classList.add('is-disabled')
        } else {
          el.parentNode.removeChild(el)
        }
      }
    }
  }
})
```

- main.js

```
import './utils/permission.js'
```

- router.js

```
export function initDynamicRoutes() {
  const currentRoutes = router.options.routes
  const rightList = store.state.rightList
  rightList.forEach(item => {
    item.children.forEach(item => {
      const itemRule = ruleMapping[item.path]
      itemRule.meta = item.rights
      currentRoutes[2].children.push(itemRule)
    })
  })
  router.addRoutes(currentRoutes)
}
```

- 使用指令

```
v-permission="{action:'add'}"
v-permission="{action:'delete', effect:'disabled'}"
```

3.4.请求和响应的控制

请求控制

- 除了登录请求都得要带上token, 这样服务器才可以鉴别你的身份

```
axios.interceptors.request.use(function(req){
  const currentUrl = req.url
  if(currentUrl !== 'login') {
    req.headers.Authorization = sessionStorage.getItem('token')
  }
  return req
})
```

- 如果发出了非权限内的请求, 应该直接在前端访问内组织,虽然这个请求发到服务器也会被拒绝

```
import axios from 'axios'
import Vue from 'vue'
import router from '../router'
// 配置请求的跟路径, 目前用mock模拟数据, 所以暂时把这一项注释起来
// axios.defaults.baseURL = 'http://127.0.0.1:8888/api/private/v1/'
const actionMapping = {
  get: 'view',
  post: 'add',
```

```
  put: 'edit',
  delete: 'delete'
}
axios.interceptors.request.use(function(req){
  const currentUrl = req.url
  if(currentUrl !== 'login') {
    req.headers.Authorization = sessionStorage.getItem('token')
    // 当前模块中具备的权限
    // 查看  get请求
    // 增加  post请求
    // 修改  put请求
    // 删除  delete请求
    const method = req.method
    // 根据请求, 得到是哪种操作
    const action = actionMapping[method]
    // 判断action是否存在当前路由的权限中
    const rights = router.currentRoute.meta
    if(rights && rights.indexOf(action) == -1) {
      // 没有权限
      alert('没有权限')
      return Promise.reject(new Error('没有权限'))
    }
  }
  return req
})
axios.interceptors.response.use(function(res){
  return res
})
Vue.prototype.$http = axios
```

响应控制

- 得到了服务器返回的状态码401, 代表token超时或者被篡改了, 此时应该强制跳转到登录界面

```
axios.interceptors.response.use(function(res){
  if (res.data.meta.status === 401) {
    router.push('/login')
    sessionStorage.clear()
    window.location.reload()
  }
  return res
})
```

4.小结

前端权限的实现必须要后端提供数据支持, 否则无法实现.

返回的权限数据的结构,前后端需要沟通协商, 怎样的数据使用起来才最方便.

4.1.菜单控制

- 权限的数据需要在多组件之间共享, 因此采用vuex
- 防止刷新界面,权限数据丢失, 所以需要存储在sessionStorage, 并且要保证两者的同步

4.2.界面控制

- 路由的导航守卫可以防止跳过登录界面
- 动态路由可以让不具备权限的界面的路由规则压根就不存在

4.3.按钮控制

- 路由规则中可以增加路由元数据meta
- 通过路由对象可以得到当前的路由规则,以及存储在此规则中的meta数据
- 自定义指令可以很方便的实现按钮控制

4.4.请求和响应控制

- 请求拦截器和响应拦截器的使用
- 请求方式的约定restful