Vue.js项目架构最佳实践

资源

- Vue-CLI 3.0

知识点

vue-cli 3.x项目配置

vue-cli项目基于 webpack 构建，并带有合理的默认配置；可以通过项目内的配置文件进行配置；可以通过插件进行扩展。

- webpack相关配置，请在根目录创建vue.config.js

```
// vue.config.js
const port = 7070;
const title = "vue项目最佳实践";

module.exports = {
  publicPath: '/best-practice', // 部署应用包时的基本 URL
  devServer: {
    port: port,
  },
  configureWebpack: {
    // 向index.html注入标题
    name: title
  }
};


// index.html
<title><%= webpackConfig.name %></title>
```

> 注意：以上配置要求vue-cli-service版本>=3.5
>
> 通过命令查看配置结果：
>
> - vue inspect 全部配置
> - vue inspect --rules 查看全部规则
> - vue inspect --rule vue 查看指定规则
> - vue inspect --plugins 查看全部插件
> - vue inspect --plugin vue-plugin 查看指定插件

> - vue inspect --mode development 指定模式
>
> 通过ui查看配置结果：
>
> - vue ui

- 高级链式操作：演示webpack规则配置

使用icon前先安装依赖：svg-sprite-loader

```
npm i svg-sprite-loader -D
```

下载图标，存入src/icons/svg中

修改规则和新增规则，vue.config.js

```
// resolve定义一个绝对路径获取函数
const path = require('path')

function resolve(dir) {
  return path.join(__dirname, dir)
}
//...
chainWebpack(config) {
    // 配置svg规则排除icons目录中svg文件处理
    config.module
        .rule("svg")
        .exclude.add(resolve("src/icons"))
        .end();
    // 新增icons规则，设置svg-sprite-loader处理icons目录中的svg
    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();
}
```

图标自动导入

```
// icons/index.js
const req = require.context('./svg', false, /\.svg$/)
req.keys().map(req);

// main.js
import './icons'
```

创建SvgIcon组件，./components/SvgIcon.vue

```
<template>
  <svg :class="svgClass" aria-hidden="true" v-on="$listeners">
    <use :xlink:href="iconName" />
  </svg>
</template>

<script>
export default {
  name: 'SvgIcon',
  props: {
    iconClass: {
      type: String,
      required: true
    },
    className: {
      type: String,
      default: ''
    }
  },
  computed: {
    iconName() {
      return `#icon-${this.iconClass}`
    },
    svgClass() {
      if (this.className) {
        return 'svg-icon ' + this.className
      } else {
        return 'svg-icon'
      }
    }
  }
}
</script>

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

注册，icons/index.js

```
import Vue from 'vue'
import SvgIcon from '@/components/SvgIcon.vue'

Vue.component('svg-icon', SvgIcon)
```

使用，App.vue

```
<svg-icon icon-class="qq"></svg-icon>
```

权限控制+动态路由

定义路由，router/index.js

```
import Vue from "vue";
import Router from "vue-router";
import Layout from '@/layout'; // 布局页

Vue.use(Router);

// 通用页面
export const constRoutes = [
  {
    path: "/login",
    component: () => import("@/views/Login"),
    hidden: true // 导航菜单忽略该项
  },
  {
    path: "/",
    component: Layout,// 应用布局
    redirect: "/home",
    children: [
      {
        path: "home",
        component: () =>
          import(/* webpackChunkName: "home" */ "@/views/Home.vue"),
        name: "home",
        meta: {
            title: "Home", // 导航菜单项标题
            icon: "qq" // 导航菜单项图标
        }
      }
    ]
  }
];

export default new Router({
  mode: "history",
  base: process.env.BASE_URL,
  routes: constRoutes
});
```

- 创建布局页面，layout/index.vue

```
<template>
  <div class="app-wrapper">
    <!-- <sidebar class="sidebar-container" /> -->
    <div class="main-container">
      <router-view />
    </div>
  </div>
</template>
```

- 创建用户登录页面，views/Login.vue

```
<template>
  <div>
    <h2>用户登录</h2>
    <div>
      <input type="text" v-model="username">
      <button @click="login">登录</button>
    </div>
  </div>
</template>
<script>
export default {
  data() {
    return {
      username: "admin"
    };
  },
  methods: {
    login() {
      this.$store
        .dispatch("user/login", { username: this.username })
        .then(() => {
          this.$router.push({
            path: this.$route.query.redirect || "/"
          });
        })
        .catch(error => {
            alert(error)
        });
    }
  }
};
</script>
```

添加动态路由

asyncRoutes中定义的路由需要在用户登录后获取其角色并过滤出有权访问的部分，最后动态添加至router

- 添加动态路由定义，router/index.js

```
// 权限页面
export const asyncRoutes = [
  {
    path: "/about",
    component: Layout,
    redirect: "/about/index",
    children: [
      {
        path: "index",
        component: () =>
          import(/* webpackChunkName: "home" */ "@/views/About.vue"),
        name: "about",
        meta: {
            title: "About",
            icon: "qq",
            roles: ['admin', 'editor']
        },
      }
    ]
  }
];
```

- 路由守卫，创建./src/permission.js，并在main.js中引入

```
import router from './router'
import store from './store'
import { Message } from 'element-ui'
import { getToken } from '@/utils/auth' // 从cookie获取令牌

const whiteList = ['/login'] // 无需令牌白名单

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

  // 获取令牌判断用户是否登录
  const hasToken = getToken()

  if (hasToken) {
    if (to.path === '/login') {
      // 若已登录重定向至首页
      next({ path: '/' })
    } else {
      // 若用户角色已附加则说明动态路由已添加
      const hasRoles = store.getters.roles && store.getters.roles.length > 0
      if (hasRoles) {
        next() // 继续即可
      } else {
        try {
          // 先请求获取用户信息
          const { roles } = await store.dispatch('user/getInfo')
```

```
          // 根据当前用户角色动态生成路由
          const accessRoutes = await store.dispatch('permission/generateRoutes', roles)

          // 添加这些路由至路由器
          router.addRoutes(accessRoutes)

          // 继续路由切换，确保addRoutes完成
          next({ ...to, replace: true })
        } catch (error) {
          // 出错需重置令牌并重新登录（令牌过期、网络错误等原因）
          await store.dispatch('user/resetToken')
          Message.error(error || 'Has Error')
          next(`/login?redirect=${to.path}`)
        }
      }
    }
  } else {
    // 用户无令牌
    if (whiteList.indexOf(to.path) !== -1) {
      // 白名单路由放过
      next()
    } else {
      // 重定向至登录页
      next(`/login?redirect=${to.path}`)
    }
  }
})
```

- 添加缺少组件：

1. element-ui：`vue add element`
2. js-cookie：`npm i js-cookie -S`
3. 添加utils/auth.js

```
import Cookies from "js-cookie";

const Token = "token";

export function getToken() {
  return Cookies.get(Token);
}

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

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

- vuex相关模块实现，创建store/index.js

```
import Vue from 'vue'
import Vuex from 'vuex'
import permission from './modules/permission'
import user from './modules/user'

Vue.use(Vuex)

const store = new Vuex.Store({
  modules: {permission, user}
})

export default store
```

- user模块：用户数据、用户登录等，store/modules/user.js

```
import { getToken, setToken, removeToken } from "@/utils/auth";

const state = {
  token: getToken(),
  roles: []
  // 其他用户信息
};

const mutations = {
  SET_TOKEN: (state, token) => {
    state.token = token;
  },
  SET_ROLES: (state, roles) => {
    state.roles = roles;
  }
};

const actions = {
  // user login
  login({ commit }, userInfo) {
    const { username } = userInfo;
    return new Promise((resolve, reject) => {
      setTimeout(() => {
        if (username === "admin" || username === "jerry") {
          commit("SET_TOKEN", username);
          setToken(username);
          resolve();
        } else {
          reject("用户名、密码错误");
        }
      }, 1000);
    });
  },

  // get user info
```

```
  getInfo({ commit, state }) {
    return new Promise((resolve) => {
      setTimeout(() => {
        const roles = state.token === 'admin' ? ['admin'] : ['editor']
        commit("SET_ROLES", roles);
        resolve({roles});
      }, 1000);
    });
  },

  // remove token
  resetToken({ commit }) {
    return new Promise(resolve => {
      commit("SET_TOKEN", "");
      commit("SET_ROLES", []);
      removeToken();
      resolve();
    });
  }
};

export default {
  namespaced: true,
  state,
  mutations,
  actions
};
```

- permission模块：路由配置信息、路由生成逻辑, store/modules/permission.js

```
import { asyncRoutes, constRoutes } from "@/router";

/**
 * 根据路由meta.role确定是否当前用户拥有访问权限
 * @roles 用户拥有角色
 * @route 待判定路由
 */
function hasPermission(roles, route) {
  // 如果当前路由有roles字段则需判断用户访问权限
  if (route.meta && route.meta.roles) {
    // 若用户拥有的角色中有被包含在待判定路由角色表中的则拥有访问权
    return roles.some(role => route.meta.roles.includes(role));
  } else {
    // 没有设置roles则无需判定即可访问
    return true;
  }
}

/**
 * 递归过滤AsyncRoutes路由表
 * @routes 待过滤路由表，首次传入的就是AsyncRoutes
 * @roles 用户拥有角色
```

```
 */
export function filterAsyncRoutes(routes, roles) {
  const res = [];

  routes.forEach(route => {
    // 复制一份
    const tmp = { ...route };
    // 如果用户有访问权则加入结果路由表
    if (hasPermission(roles, tmp)) {
      // 如果存在子路由则递归过滤之
      if (tmp.children) {
        tmp.children = filterAsyncRoutes(tmp.children, roles);
      }
      res.push(tmp);
    }
  });

  return res;
}

const state = {
  routes: [], // 完整路由表
  addRoutes: [] // 用户可访问路由表
};

const mutations = {
  SET_ROUTES: (state, routes) => {
    state.addRoutes = routes;
    state.routes = constRoutes.concat(routes);
  }
};

const actions = {
  // 路由生成：在得到用户角色后会第一时间调用
  generateRoutes({ commit }, roles) {
    return new Promise(resolve => {
      let accessedRoutes;
      //   用户是管理员则拥有完整访问权限
      if (roles.includes("admin")) {
        accessedRoutes = asyncRoutes || [];
      } else {
        //   否则需要根据角色做过滤处理
        accessedRoutes = filterAsyncRoutes(asyncRoutes, roles);
      }
      commit("SET_ROUTES", accessedRoutes);
      resolve(accessedRoutes);
    });
  }
};

export default {
  namespaced: true,
  state,
```

```
  mutations,
  actions
};
```

- getters编写，store/index.js

```
const store = new Vuex.Store({
  modules: { permission, user },
  // 全局定义getters便于访问
  getters: {
    roles: state => state.user.roles,
  }
});
```