Vue中的ajax库

axios库

通用的 Ajax 请求库, 官方推荐，使用广

官网：https://axios-http.com/zh/docs/intro

安装

使用npm

```
npm install axios
```

或者使用yarn

```
yarn add axios
```

基本使用方法

axios发起请求基本上就是 axios(配置对象)的方式，完整的请求配置参考这里：https://axios-http.com/zh/docs/req_config

例如：

```
import axios from 'axios'; //导入库
axios({
  method: 'post',           //可以为 get,post,put,delete等http请求方法
  url: '/user/12345',
  data: {                   // 请求体数据
    firstName: 'Fred',
    lastName: 'Flintstone'
  }
}).then(function (response) {
    console.log(respones.data) // 状态码200时，处理响应数据
  }).catch(error){
    // 非200时，当作异常响应处理
}
```

封装自定义方法案例：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>axios-demo</title>
</head>
<body>
  <div id="root">
```

```
    <button @click="login">点我发送</button>
    <h3>响应信息:{{respMsg}}</h3>
  </div>
  <script src="../vue.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
  <script>
    Vue.createApp({
      setup(){
        let respMsg=Vue.ref('')
        function login(){
          axios({
            method: 'post',
            url: 'http://120.27.146.185:8076/api/login/',
            data: {
              username: 'haiwen',
              password: '123123'
            }
          }).then(function(response){
              respMsg.value=response.data
          }).catch(function(error){
            respMsg.value=error.response.data
          })
        }
        return{
          respMsg,
          login,
        }
      },
    }).mount("#root")
  </script>
</body>
</html>
```


请求方式别名

为了方便起见，已经为所有支持的请求方法提供了别名。

axios.request(config)

axios.get(url[, config])

axios.delete(url[, config])

axios.head(url[, config])

axios.options(url[, config])

axios.post(url[, data[, config]])

axios.put(url[, data[, config]])

axios.patch(url[, data[, config]])

注意

在使用别名方法时， `url`、`method`、`data` 这些属性都不必在配置中指定。

封装常用方法

以登录和登出为例, 将业务逻辑和请求方法统一封装到函数中

```
import axios from 'axios'; //导入库
import { ElMessage } from "element-plus";
import router from '@/router'
//默认配置axios.defaults.xxxx
axios.defaults.validateStatus=(status)=>{
  return status >= 200 && status < 400; //  根据状态码判断响应走正常还是异常逻辑
}

const host = 'http://120.27.146.185:8076'
function login(username,password){
  axios.post(`${host}/api/login/`,{
    username,
    password,
  }).then((resp) => {
    console.log('正常响应')
    router.push("/");
    localStorage.setItem('islogin','yes')
    ElMessage.success(resp.data.msg);
  }).catch((err) => {
    console.log('异常响应')
    ElMessage.error(err.response.data.error);
  })
}

function logout(){
  axios.get(`${host}/api/logout`).then(()=>{
    router.push("/login")
    localStorage.setItem('islogin','no')
    ElMessage.info('退出登录');
  })
}

export {
  login,
  logout
}
```

Login.vue这里再调用相关逻辑

```
import { ref, reactive } from "vue";
import background from '../components/Background'
import {logout,login} from '../httplib'
export default {
  components:{
    background
  },
  setup() {
    const form = reactive({
      account: "",
      psw: "",
    });
    const loginForm = ref(null);
```

```
    function submit() {
      login(form.account,form.psw) //调用登录
      form.account = "";
      form.psw = "";
    }

    function reset() {
      loginForm.value.resetFields()
    }
    const rules = {
      account: [
        {
          required: true,
          message: "请输入用户名",
          trigger: "blur",
        },
      ],
      pass: [
        {
          required: true,
          message: "请输入密码",
          trigger: "blur",
        },
      ],
    };
    return {
      form,
      loginForm,
      submit,
      reset,
      rules,
    };
  },
};
```

主页这里测试下登出逻辑

```
<template>
  <div class="home">
    <h1>这里是主页</h1>
    <button @click="quit">点我退出</button>
  </div>
</template>
<script>
import {logout} from '@/httplib'
export default {
  setup(props) {
    function quit(){
      logout()
    }
    return{
      quit
    }
  }
}
</script>
```

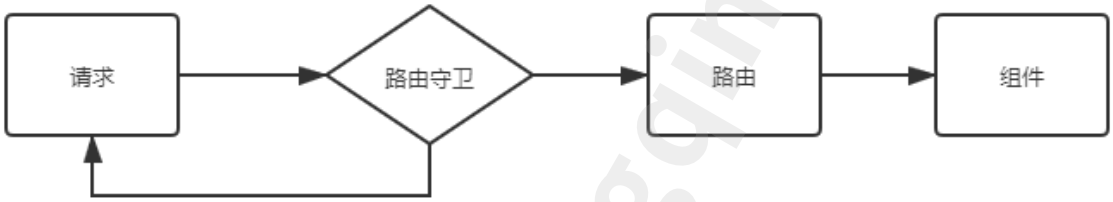
```
<style scoped>
  .home{
    height: 100vh;
    background-color: antiquewhite;
  }
</style>
```

vue路由守卫

路由守卫作用

目前为止登录和退出功能，可以正常跳转对应的页面。但是我们发现，直接访问url，不通过登录也能访问主页。

这个在现实项目中是不被允许的，所以要有个检查机制，只有通过校验的请求才能访问对应的路由，这个就叫做路由守卫。



路由守卫常用阶段

全局前置守卫

```
//router/index.js
...
//全局前置守卫,守卫接收两个参数,to,from
router.beforeEach((to,from)=>{
  console.log('to',to.path)      // 即将要访问的路由
  console.log('from',from.path)  // 当前导航正要离开的路由
})
```

可以的返回值有: false 或路由地址

```
//全局前置守卫
router.beforeEach((to,from)=>{
  console.log('to',to.path)
  console.log('from',from.path)
  if(to.path==='/login'){
    // return false       系统访问/login就取消导航
    return '/haiwen'      //系统访问/login 重定向到/haiwen
  }
})
```

路由守卫实际应用

根据用户是否登录来判断访问的路由是否该放行还是重定向到登录,可以接收第三个参数next，next表示放行的意思，会继续访问to指向的路径，但是要注意，这仍然会触发全局守卫

```
//router/index.js
...
//全局前置守卫
router.beforeEach((to,from,next)=>{
  if(to.name !=='login' && localStorage.getItem('islogin')!=='yes'){
    console.log('未登录')
    next({name:'login'}) //如果访问的不是login且没有登录，就重定向到login
  }else{
    if(to.name !=='login'){
      console.log('已登录')
    }else{
      console.log('未登录')
    }
    next()
  }
})
...
```