

【前端面试】你封装过Axios请求吗？

前言

axios 是目前最优秀的 HTTP 请求库之一，虽然 axios 已经封装的非常好了，我们可以直接拿来用。但是在实际的项目中，我们可能还需要对 axios 在封装一下，以便我们更好的管理项目和各个借口。

但是，目前网上有特别多的针对于 axios 在项目中的封装。不得不说，很多大佬封装得非常全面，方方面面都考虑到了。但是我们的每个真的都需要那些封装吗？显然不是的，网上的很多封装其实都显得**有点过度封装**了！

本篇文章实现最简单 Axios 封装，让小伙伴们扩展起来容易一些。

1.封装目的

此次进行简单的封装，所以暂时没有考虑**取消重复请求、重复发送请求、请求缓存**等情况！这里主要实现以下目的：

1. 实现请求拦截
2. 实现响应拦截
3. 常见错误信息处理
4. 请求头设置
5. api 集中式管理

2.初始化 axios 实例

虽然 axios 可以调用 get、post 等方法发起请求，但是我们为了更好的全局控制所有请求的相关配置，所以我们使用 axios.create()创建实例的方法来进行相关配置，这也是封装 axios 的精髓所在。

示例代码：

```
1 // 创建 axios 请求实例
2 const serviceAxios = axios.create({
3   baseURL: "", // 基础请求地址
4   timeout: 10000, // 请求超时设置
5   withCredentials: false, // 跨域请求是否需要携带 cookie
6 });
```

通过 create 方法我们得到了一个 axios 的实例，该实例上有很多方法，比如拦截器等等。我们创建实例的时候可以配置一些基础设置，比如基础请求地址，请求超时等等。

3.设置请求拦截

我们在发送请求的时候可能需要携带一些信息在请求头上，比如 token 等，所以说我们就需要将请求拦截下来，处理一些我们的业务逻辑。

示例代码：

```
1 // 创建请求拦截
2 serviceAxios.interceptors.request.use(
3   (config) => {
4     // 如果开启 token 认证
5     if (serverConfig.useTokenAuthorization) {
6       config.headers["Authorization"] = localStorage.getItem("token"); // 请
7     }
8     // 设置请求头
9     if(!config.headers["content-type"]) { // 如果没有设置请求头
10      if(config.method === 'post') {
11        config.headers["content-type"] = "application/x-www-form-urlencoded"
12        config.data = qs.stringify(config.data); // 序列化,比如表单数据
13      } else {
14        config.headers["content-type"] = "application/json"; // 默认类型
15      }
16    }
17    console.log("请求配置", config);
18    return config;
19  },
20  (error) => {
21    Promise.reject(error);
22  }
23 );
```

我们通过调用 axios 的实例方法来进行请求拦截。

4.设置响应拦截

axios 请求的返回结果里面包含了很多东西，我们的业务层面通常只需要后端返回的数据即可，所以我们需要设置相应拦截，在响应结果返回给业务层之前做一些操作。

示例代码：

```
1 // 创建响应拦截
2 serviceAxios.interceptors.response.use(
3   (res) => {
4     let data = res.data;
5     // 处理自己的业务逻辑,比如判断 token 是否过期等等
6     // 代码块
7     return data;
```

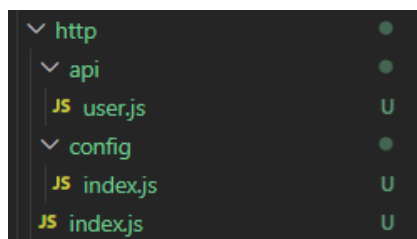
```
8   },
9   (error) => {
10     let message = "";
11     if (error && error.response) {
12       switch (error.response.status) {
13         case 302:
14           message = "接口重定向了！";
15           break;
16         case 400:
17           message = "参数不正确！";
18           break;
19         case 401:
20           message = "您未登录，或者登录已经超时，请先登录！";
21           break;
22         case 403:
23           message = "您没有权限操作！";
24           break;
25         case 404:
26           message = `请求地址出错：${error.response.config.url}`;
27           break;
28         case 408:
29           message = "请求超时！";
30           break;
31         case 409:
32           message = "系统已存在相同数据！";
33           break;
34         case 500:
35           message = "服务器内部错误！";
36           break;
37         case 501:
38           message = "服务未实现！";
39           break;
40         case 502:
41           message = "网关错误！";
42           break;
43         case 503:
44           message = "服务不可用！";
45           break;
46         case 504:
47           message = "服务暂时无法访问，请稍后再试！";
48           break;
49         case 505:
50           message = "HTTP 版本不受支持！";
51           break;
52         default:
53           message = "异常问题，请联系管理员！";
54           break;
55       }
56     }
57     return Promise.reject(message);
58   }
```

5.完整示例

上面直接简单介绍了拦截等相关代码，接下来我们在实际项目中来演练一下，以 Vue 项目为例。

在 src 下面新建 http 文件夹，用来存储关于 axios 请求的一些文件，然后在 http 文件夹下新建 index.js 文件，用于封装我们的 axios，然后在新建 config 文件夹，主要用来创建配置文件，最后新建一个 api 文件夹，用于集中管理我们的接口。

文件夹目录如下：



index.js 代码：

```
1 import axios from "axios";
2 import serverConfig from "../config";
3 import qs from "qs";
4
5 // 创建 axios 请求实例
6 const serviceAxios = axios.create({
7   baseURL: serverConfig.baseURL, // 基础请求地址
8   timeout: 10000, // 请求超时设置
9   withCredentials: false, // 跨域请求是否需要携带 cookie
10 });
11
12 // 创建请求拦截
13 serviceAxios.interceptors.request.use(
14   (config) => {
15     // 如果开启 token 认证
16     if (serverConfig.useTokenAuthorization) {
17       config.headers["Authorization"] = localStorage.getItem("token"); // 请
18     }
19     // 设置请求头
20     if (!config.headers["content-type"]) { // 如果没有设置请求头
21       if (config.method === 'post') {
22         config.headers["content-type"] = "application/x-www-form-urlencoded"
23         config.data = qs.stringify(config.data); // 序列化,比如表单数据
24       } else {
25         config.headers["content-type"] = "application/json"; // 默认类型
26       }
27     }
28     console.log("请求配置", config);
29     return config;
```

```
30   },
31   (error) => {
32     Promise.reject(error);
33   }
34 );
35
36 // 创建响应拦截
37 serviceAxios.interceptors.response.use(
38   (res) => {
39     let data = res.data;
40     // 处理自己的业务逻辑，比如判断 token 是否过期等等
41     // 代码块
42     return data;
43   },
44   (error) => {
45     let message = "";
46     if (error && error.response) {
47       switch (error.response.status) {
48         case 302:
49           message = "接口重定向了！";
50           break;
51         case 400:
52           message = "参数不正确！";
53           break;
54         case 401:
55           message = "您未登录，或者登录已经超时，请先登录！";
56           break;
57         case 403:
58           message = "您没有权限操作！";
59           break;
60         case 404:
61           message = `请求地址出错：${error.response.config.url}`;
62           break;
63         case 408:
64           message = "请求超时！";
65           break;
66         case 409:
67           message = "系统已存在相同数据！";
68           break;
69         case 500:
70           message = "服务器内部错误！";
71           break;
72         case 501:
73           message = "服务未实现！";
74           break;
75         case 502:
76           message = "网关错误！";
77           break;
78         case 503:
79           message = "服务不可用！";
80           break;
```

```

81         case 504:
82             message = "服务暂时无法访问，请稍后再试！";
83             break;
84         case 505:
85             message = "HTTP 版本不受支持！";
86             break;
87         default:
88             message = "异常问题，请联系管理员！";
89             break;
90     }
91 }
92 return Promise.reject(message);
93 }
94 );
95 export default serviceAxios;

```

config/index.js 代码：

```

1  const serverConfig = {
2      baseURL: "https://smallpig.site", // 请求基础地址,可根据环境自定义
3      useTokenAuthorization: true, // 是否开启 token 认证
4  };
5  export default serverConfig;

```

api/user.js 接口调用示例代码：

```

1  import serviceAxios from "../index";
2
3  export const getUserInfo = (params) => {
4      return serviceAxios({
5          url: "/api/website/queryMenuWebsite",
6          method: "post",
7          params,
8      });
9  };
10 export const login = (data) => {
11     return serviceAxios({
12         url: "/api/user/login",
13         method: "post",
14         data,
15     });
16 };

```

注意：get 请求需要传 params，post 请求需要传 data。

Vue 文件中调用示例：

```

1  import { login } from "@http/api/user"
2  async loginAsync() {
3      let params = {
4          email: "123",
5          password: "12321"
6      }

```

```
7   let data = await login(params);  
8   console.log(data);  
9 }
```

6.总结

我们这里只做了最最基础的 axios 封装，但是可扩展性较高。相较于其它文章的过度封装，这里的封装形式其实可以满足大部分应用场景了。

我们在此基础上可以根据业务场景做一些以下扩展建议：

- 请求拦截里面针对 token 进行处理
- 响应拦截里面判断 token 是否过期等等
- 在 config/index.js 里面动态更改 baseUrl
- 在请求拦截里面根据业务场景修改请求头
- 在拦截里面设置全局请求进度条等等