河南中医药大学信息技术学院（智能医疗行业学院）产教协同课程建设成果
智能医学工程专业《医疗信息系统开发》课程

第07章：调用接口进行开发

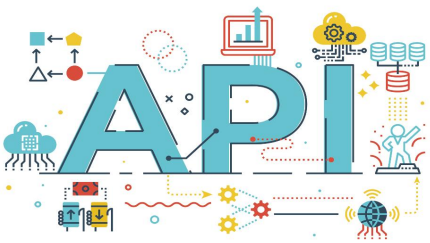
黄子杰
河南方和信息科技有限公司

河南中医药大学信息技术学院（智能医疗行业学院）智能医疗教研室

https://aitcm.hactcm.edu.cn
2025/10/23

本章概要

- 核心概念解析
  - 前后端分离架构回顾
  - 接口联调的核心要素
  - 软件功能设计
- 若依框架中的请求封装
  - Axios简介
  - 核心配置
  - API集中管理
- 系统案例
  - 分析接口文档
  - 编写模板并渲染数据
  - 编写script部分并调用API
  - 编写API函数
  - 常见问题排查



核心概念解析

前后端分离架构回顾

- 前端：负责视图渲染、用户交互。 -> 若依(Vue)
- 后端：负责业务逻辑、数据存储。 -> 提供API接口
- 桥梁：HTTP API （JSON作为数据交换格式）

核心概念解析

接口联调的核心要素

- 接口文档（API Documentation）：前后端沟通的“契约”。

  包含内容：请求URL、请求方法（GET/POST/PUT/DELETE）、请求参数（Query、Body）、响应数据结构。

- HTTP状态码

  200（成功）、400（客户端错误）、401（未授权）、404（未找到）、500（服务器错误）。

- 开发者工具（DevTools）：使用浏览器Network面板查看请求和响应，浏览器自带的实用工具。

若依框架中的请求封装

Axios

- 若依框架中的网络请求架构

  Vue 组件层(pages, components) 调用简洁的API函数

  API 管理层(api/ 目录) 接口集中管理

  Axios 实例封装层(utils/request.js) 核心封装

  后端 API 接口(Spring Boot)

- 若依中 Axios 的封装理念

```
axios.post('/api/user/list', params, {
  headers: {
    'Authorization': 'Bearer ' + getToken(),
    'Content-Type': 'application/json'
  }
}).then(response => {
  if (response.data.code === 401) {
    // 处理token过期
    alert('请重新登录')
    router.push('/login')
  } else if (response.data.code === 200) {
    return response.data
  } else {
    // 处理其他错误...
  }
})
```

```
// 在 Vue 组件中
methods: {
  getUserList() {
    // 调用 API 管理层的函数
    listUser(this.queryParams).then(data => {
      this.userList = data.rows
    })
    // 不需要处理错误，统一的错误处理已生效
  }
}
```

```
// api/system/user.js
export function listUser(query) {
  return request({
    url: '/system/user/list',
    method: 'get',
    params: query
  })
}
```


若依框架中的请求封装

核心配置

- request.js

```
// 创建axios实例
const service = axios.create({
  // axios中请求配置有baseURL选项，表示请求URL公共部分
  baseURL: import.meta.env.VITE_APP_BASE_API,
  // 超时
  timeout: 10000
})
// request拦截器
service.interceptors.request.use( onFulfilled: config => {
  // 是否需要设置 token
  const isToken = (config.headers || {}).isToken === false
  // 是否需要防止数据重复提交
  const isRepeatSubmit = (config.headers || {}).repeatSubmit === false
  if (getToken() && !isToken) {
    // 每个请求携带自定义token 可根据实际情况自行修改
    config.headers['Authorization'] = 'Bearer ' + getToken()
  }
  // get请求映射params参数
  if (config.method === 'get' && config.params) {
    let url = config.url + '?' + tansParams(config.params)
    url = url.slice(0, -1)
    config.params = {}
    config.url = url
  }
```

```
//防重复提交机制
if (!isRepeatSubmit && (config.method === 'post' || config.method === 'put')) {
  const requestObj = {
    url: config.url,
    data: typeof config.data === 'object' ? JSON.stringify(config.data) : config.data,
    time: new Date().getTime()
  }
  const requestSize = Object.keys(JSON.stringify(requestObj)).length // 请求数据大小
  const limitSize = 5 * 1024 * 1024 // 限制存放数据5M
  if (requestSize >= limitSize) {
    console.warn( message: `[${config.url}]: ` + '请求数据大小超出允许的5M限制，无法进行防重复提交验证。')
    return config
  }
  const sessionObj = cache.session.getJSON( key: 'sessionObj')
  if (sessionObj === undefined || sessionObj === null || sessionObj === '') {
    cache.session.setJSON( key: 'sessionObj', requestObj)
  } else {
    const s_url = sessionObj.url                // 请求地址
    const s_data = sessionObj.data              // 请求数据
    const s_time = sessionObj.time              // 请求时间
    const interval = 1000                       // 间隔时间(ms)，小于此时间视为重复提交
    if (s_data === requestObj.data && requestObj.time - s_time < interval && s_url === requestObj.url) {
      const message = '数据正在处理，请勿重复提交'
      console.warn( message: `[${s_url}]: ` + message)
      return Promise.reject(new Error(message))
    } else {
      cache.session.setJSON( key: 'sessionObj', requestObj)
    }
```

若依框架中的请求封装

API集中管理

- 所有接口函数集中管理：利于维护和复用。

```
// 将整个Axios封装层引入到当前文件中
// 实际导入的是一个配置好的Axios实例
import request from '@/utils/request'
// 等价于
//import request from '/src/utils/request.js'

// 查询患者列表
export function listPatient(query) {
    return request({
        url: '/patient/list',
        method: 'post',
        data: query
    })
}

// 获取患者详细信息
export function getPatient(data) {
    return request({
        url: '/patient/detail',
        method: 'post',
        data: data
    })
}
```

```
// 修改患者
export function updatePatient(data) {
    return request({
        url: '/patient/update',
        method: 'post',
        data: data
    })
}

// 删除患者
export function delPatient(patientId) {
    return request({
        url: '/patient/' + patientId,
        method: 'delete'
    })
}
```


系统案例

- 分析接口文档

  接口地址：/system/user/list

  请求方法：GET

  请求参数：{ userName: “张” }

  响应数据：{ code: 200, msg: “查询成功”, rows: [...], total: 100 }

- 编写API函数

```
export function listUser(query) {
  return request({
    url: '/system/user/list',
    method: 'get',
    params: query // 注意：GET请求使用 `params`
  })
}
```

系统案例

- 分析接口文档

  接口地址：/system/user/list
  请求方法：GET
  请求参数：{ userName: “张” }
  响应数据：{ code: 200, msg: “查询成功”, rows: [...], total: 100 }

- 创建模板并在模板中渲染数据

```
<el-table :data="userList">
  <el-table-column prop="userId" label="用户ID"></el-table-column>
  <el-table-column prop="userName" label="用户姓名"></el-table-column>
  <el-table-column prop="email" label="邮箱"></el-table-column>
</el-table>
```


系统案例

- 编写vue 页面的 <script> 部分

```
import { listUser } from “@/api/system/user”; // 导入API

export default {
  name: ‘UserManagement’,
  data() {
    return {
      // 查询参数
      queryParams: {
        pageNum: 1,
        pageSize: 10,
        userName: “”
      },
      // 表格数据
      userList: [],
      total: 0
    };
  created() {
    this.getList(); // 页面加载时自动调用
  },
  methods: {
    /** 查询用户列表 */
    getList() {
      listUser(this.queryParams).then(response => {
        // response 已经是拦截器处理后的 res.data
        this.userList = response.rows;
        this.total = response.total;
        // 若依框架可能会在成功时自动提示，此处根据情况决定是否手动 message
      }).catch(error => {
        // 错误信息已被响应拦截器统一处理，这里通常用于处理特殊逻辑
        console.log(“请求失败: ”, error);
      });
    },
    /** 搜索按钮事件 */
    handleQuery() {
      this.queryParams.pageNum = 1;
      this.getList();
    },
    /** 重置按钮事件 */
    resetQuery() {
      // ... 重置 queryParams ...
      this.handleQuery();
    }
  }
};
```

系统案例

- 编写API函数

```
// 获取用户列表
export function listUser(query) {
  return request({
    url: '/system/user/list',
    method: 'get',
    params: query // 注意：GET请求使用 `params`
  })
}
```


系统案例

常见问题排查

- 404 Not Found：检查请求URL是否正确，baseURL 是否配置。
- 401 Unauthorized：检查token是否有效、是否已正确携带。
- 400 Bad Request：检查请求参数格式（Query/Body）是否正确，特别是JSON格式。
- 500 Internal Server Error：首先查看后端日志，通常是后端业务代码bug。
- 跨域问题（CORS）：同源策略(协议、域名、端口一致)，浏览器的一种安全机制，禁止一个域的网页访问另一个域的资源。若依通过 vue.config.js的proxy代理解决，生产环境由Nginx等解决。
- Network面板的使用：可以查看请求与响应。

本章作业

- 使用若依框架(前后端分离)完成患者健康档案管理功能

  要求:

  1.使用本地数据库并创建患者表。

  2.具有患者建档管理功能。

  3.后端使用RESTful接口。

河南中医药大学信息技术学院（智能医疗行业学院）智能医疗教研室
河南中医药大学医疗健康信息工程技术研究所