

# 第13章 Spring Boot+Vue 前后端分离项目实战

## 13.1 选项式 API 项目实战

本项目使用原生的 Vue 开发，复制项目 `test_vue_router` 并重命名为 `hrsys_options`。虽然利用 Bootstrap 修饰，实际上还是使用 Vue 原生代码与 HTML、CSS 结合的方式进行设计和开发工作。

### 13.1.1 引入技术依赖

在本项目中需要引入 axios 依赖和 Bootstrap 依赖，首先引入 axios 依赖，打开 package.json

找到 dependencies 属性，在其对应的对象中加入 `"axios": "^1.4.0"`，代码如下。

在项目中引入 Bootstrap 依赖，这与引入 axios 依赖的方式一样。在 package.json 的 dependencies 中先加入 jQuery 依赖，因为 Bootstrap 依赖于 jQuery，所以要引入 jQuery 依赖，代码如下。

@rollup/plugin-inject 的作用是扫描系统的全局变量，并进行 jQuery 的 \$ 关键字替换。

```
{
  "name": "hrsys_options",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vue-tsc && vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "vue": "^3.2.47",
    "vue-router": "^4.2.1",
    "axios": "^1.4.0",
    "bootstrap": "^3.4.0",
    "jquery": "^3.3.1",
    "@rollup/plugin-inject": "^5.0.3"
  },
  "devDependencies": {
    "@vitejs/plugin-vue": "^4.1.0",
    "typescript": "^5.0.2",
    "vite": "^4.3.9",
    "vue-tsc": "^1.4.2"
  }
}
```

在 src 下新建 util 目录，在该目录中新建 `axiosInstance.ts`，并在其中配置后端服务器路径。

```
import axiosAPI from 'axios'

//使用create({config})方法创建axios实例
const axios = axiosAPI.create({
  baseURL: 'http://localhost:8090', //请求后端数据的地址
  timeout: 2000 //请求超时设置，单位ms
})

export default axios
```

在项目根目录下的 vite.config.ts 中先配置 jQuery，添加以下代码。

```
import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
import inject from '@rollup/plugin-inject'
// https://vitejs.dev/config/
export default defineConfig({
  plugins: [vue(),
    inject({
      $: "jquery", // 这里会自动载入 node_modules 中的 jquery
      jquery: "jquery",
      "windows.jquery": "jquery"
    })],
})
```

然后在 src/main.ts 中引入 Bootstrap 依赖。

```
import { createApp } from 'vue'
import './style.css'
import App from './App.vue'
import router from './router'
import "jquery"
import "bootstrap"
import "bootstrap/dist/css/bootstrap.css"
createApp(App).use(router).mount('#app')
```

在 Vue 的项目中，当 Bootstrap 依赖建立 < Table > 标签时不能自动生成高级表格，故需要使用高级表格的形式。

### 13.1.2 员工管理模块

因为后端程序已经开发完毕，在 IDEA 中启动第 10 章的 hrsys\_restful 项目，记住它的端口号，以本项目为例是 8090，页面通过 axios 获取数据就是从该后端项目中获取。

在 components 下新建 employee 文件夹，并在其中新建 Show.vue 文件

```
<script lang="ts">
import axios from "../../util/axiosInstance"

export default {
  name: "Show",
  data() {
    return {
      selectedId: -1,
      form: {
```

```

        number: null,
        name: null,
        gender: "",
        age: null,
        'dep.id': ""
    },
    list: [],
    depList: []
}
},
methods: {
    search: function () {
        axios.get('/emp', {params: this.form})
            .then(function (res) {
                this.list = res.data;
            }).bind(this))
    },
    searchDep: function () {
        axios.get('/dep')
            .then(function (res) {
                this.depList = res.data;
            }).bind(this))
    },
    selectTr: function (id) {
        this.selectedId = id;
    },
    showAdd: function () {
        this.$router.push({name: "EmpAdd"})
    },
    showUpdate: function () {
        if (this.selectedId > -1) {
            this.$router.push({name: "EmpUpdate", query: {id: this.selectedId}})
        } else {
            alert("请选中数据");
        }
    },
    deleteData: function () {
        if (this.selectedId > -1) {
            axios.delete('/emp/' + this.selectedId)
                .then(function (res) {
                    if (res.data == true) {
                        this.search();
                    } else {
                        alert("系统错误");
                    }
                }).bind(this))
        } else {
            alert("请选中数据");
        }
    },
    mounted() {
        this.search();
        this.searchDep();
    }
}
}
</script>

```

```

<template>
  <div id="container">
    <div class="form-horizontal">
      <form class="form-group" @submit.prevent="search()">
        <div class="col-sm-2">
          <input type="text" class="form-control"
            placeholder="编号" v-model="form.number">
        </div>
        <div class="col-sm-2">
          <input type="text" class="form-control"
            placeholder="姓名" v-model="form.name">
        </div>
        <div class="col-sm-2">
          <select name="gender" class="form-control" v-
model="form.gender">
            <option value="">性别</option>
            <option value="男">男</option>
            <option value="女">女</option>
          </select>
        </div>
        <div class="col-sm-2">
          <input type="text" class="form-control" placeholder="年龄" v-
model="form.age">
        </div>
        <div class="col-sm-2">
          <select class="form-control" v-model="form['dep.id']">
            <option value="">部门</option>
            <option v-for="dep in depList"
              v-text="dep.name" v-bind:key="dep.id" v-
bind:value="dep.id">
            </option>
          </select>
        </div>
        <div class="col-sm-2">
          <button class="btn btn-primary">搜索</button>
        </div>
      </form>
    </div>
    <table class="table table-striped table-bordered table-responsive table-
hover">
      <thead>
        <tr>
          <th>编号</th>
          <th>名字</th>
          <th>性别</th>
          <th>年龄</th>
          <th>部门</th>
        </tr>
      </thead>
      <tbody class="scrollable-tbody">
        <tr class="data" v-for="emp in list" v-bind:key="emp.id" v-
bind:class="{selected:emp.id==selectedId}"
          @click="selectTr(emp.id)">
          <td v-text="emp.number"></td>
          <td v-text="emp.name"></td>
          <td v-text="emp.gender"></td>
          <td v-text="emp.age"></td>

```



根据上一步，分别创建 Add.vue 和 Update.vue 组件

Add.vue 如下：

```
<script lang="ts">
import axios from "../../util/axiosInstance"
export default {
  name: "Add",
  data() {
    return {
      form: {
        number: null,
        name: null,
        gender: null,
        age: null,
        dep: {
          id: null
        }
      },
      depList: []
    }
  },
  methods: {
    add: function () {
      axios.post('/emp', this.form)
        .then(function (res) {
          if (res.data == true) {
            this.$router.push({name: "EmpShow"})
          } else {
            alert("系统错误");
          }
        })
        .bind(this)
    },
    searchDep: function () {
      axios.get('/dep')
        .then(function (res) {
          this.depList = res.data;
          this.form.dep.id = this.depList[0].id
        })
        .bind(this)
    },
  },
  mounted() {
    this.searchDep();
  }
}
</script>

<template>
<div id="container">
  <form class="form-horizontal" @submit.prevent="add">
    <div class="form-group">
      <label class="col-sm-2 control-label">编号</label>
      <div class="col-sm-10">
        <input type="text" class="form-control" placeholder="请输入编号" v-model="form.number">
      </div>
    </div>
  </form>
</div>
</template>
```

```

    </div>
    <div class="form-group">
      <label class="col-sm-2 control-label">名字</label>
      <div class="col-sm-10">
        <input type="text" class="form-control" placeholder="请输入名字"
          v-model="form.name">
      </div>
    </div>
    <div class="form-group">
      <label class="col-sm-2 control-label">性别</label>
      <div class="col-sm-10">
        <input type="radio" value="男" v-model="form.gender"/>男
        <input type="radio"
          value="女" v-model="form.gender"/>女
      </div>
    </div>
    <div class="form-group">
      <label class="col-sm-2 control-label">年龄</label>
      <div class="col-sm-10">
        <input type="text" class="form-control" placeholder="请输入年龄"
          v-model="form.age">
      </div>
    </div>
    <div class="form-group">
      <label class="col-sm-2 control-label">部门</label>
      <div class="col-sm-10">
        <select class="form-control" v-model="form.dep.id">
          <option v-for="dep in depList"
            v-text="dep.name" v-bind:key="dep.id" v-
            bind:value="dep.id">
          </option>
        </select>
      </div>
    </div>
    <div class="form-group">
      <div class="col-sm-offset-2 col-sm-10">
        <button type="submit" class="btn btn-primary">保存</button>
      </div>
    </div>
  </form>
</div>
</template>

<style scoped>
  #container {
    width: 600px;
    margin: 10px auto;
  }
</style>

```

Update.vue 代码如下:

```
<script lang="ts">
```

```

import API from "../../util/axiosInstance"
import { useRoute } from 'vue-router';

export default {
  name: "Update",
  data() {
    return {
      form: {
        id: null,
        number: null,
        name: null,
        gender: null,
        age: null,
        dep: {
          id: null
        }
      },
      depList: []
    }
  },
  methods: {
    update: function () {
      API.post('/emp', this.form)
        .then(function (res) {
          if (res.data == true) {
            this.$router.push({name: "EmpShow"})
          } else {
            alert("系统错误");
          }
        }).bind(this))
    },
    searchById: function () {
      API.get('/emp/' + this.form.id)
        .then(function (res) {
          this.form = res.data;
          if (this.form.dep == null) {
            this.form.dep = {id: this.depList[0].id};
          }
        }).bind(this))
    },
    searchDep: function () {
      API.get('/dep')
        .then(function (res) {
          this.depList = res.data;
        }).bind(this))
    },
    mounted() {
      this.form.id = this.$route.query.id;
      console.log(this.form.id)
      this.searchDep();
      this.searchById();
    }
  }
}
</script>
<template>

```



```

<div id="container">
  <form class="form-horizontal" @submit.prevent="update">
    <div class="form-group">
      <label class="col-sm-2 control-label">编号</label>
      <div class="col-sm-10">
        <input type="text" class="form-control" placeholder="请输入编号" v-model="form.number">
      </div>
    </div>
    <div class="form-group">
      <label class="col-sm-2 control-label">名字</label>
      <div class="col-sm-10">
        <input type="text" class="form-control" placeholder="请输入名字" v-model="form.name">
      </div>
    </div>
    <div class="form-group">
      <label class="col-sm-2 control-label">性别</label>
      <div class="col-sm-10">
        <input type="radio" value="男" v-model="form.gender"/>男
        <input type="radio" value="女" v-model="form.gender"/>女
      </div>
    </div>
    <div class="form-group">
      <label class="col-sm-2 control-label">年龄</label>
      <div class="col-sm-10">
        <input type="text" class="form-control" placeholder="请输入年龄" v-model="form.age">
      </div>
    </div>
    <div class="form-group">
      <label class="col-sm-2 control-label">部门</label>
      <div class="col-sm-10">
        <select class="form-control" v-model="form.dep.id">
          <option v-for="dep in depList" v-text="dep.name" v-bind:key="dep.id" v-bind:value="dep.id">
          </option>
        </select>
      </div>
    </div>
    <div class="form-group">
      <div class="col-sm-offset-2 col-sm-10">
        <button type="submit" class="btn btn-primary">保存</button>
      </div>
    </div>
  </form>
</div>
</template>

<style scoped>
  #container {
    width: 600px;
    margin: 10px auto;
  }

```

```
}  
</style>
```

在 router/index.ts 中创建三个组件对应的路由：

下面将路由重新规划，所有模块的路由都应该是 Index 组件对应路由的子路由。

```
// history模式  
import {  
  createRouter, createWebHistory  
} from 'vue-router'  
  
const routes = [  
  {  
    path: "/",  
    name: "Index",  
    component: () =>  
      import( "../components/Index.vue"),  
    children: [  
      {  
        path: "/emp/show",  
        name: "EmpShow",  
        component: () =>  
          import( "../components/employee/Show.vue")  
      },  
      {  
        path: "/emp/add",  
        name: "EmpAdd",  
        component: () =>  
          import( "../components/employee/Add.vue")  
      },  
      {  
        path: "/emp/update",  
        name: "EmpUpdate",  
        component: () =>  
          import( "../components/employee/Update.vue")  
      }  
    ]  
  }  
];  
  
// 创建路由对象  
const router = createRouter({  
  history: createWebHistory(),  
  routes  
})  
export default router;
```

这里定义的是/emp/update 路径，是为了避免和 Department、SysUser 等模块产生命名 冲突。

需要注意的是，在 Vue 项目下，对于由 v-for 指令生成的下拉框、表格行等必须加上

v-bind: key，其值一般由对应数据的 id 来填充。

路由控制页面跳转和 axios 控制从后端服务器接口获取数据。

### 13.1.3 解决跨域问题

此时，从浏览器访问项目会发现界面可以正常显示，但数据无法显示，通过浏览器控制台可以看到报“No 'Access-Control-Allow-Origin' header”的错误。引发这个错误的原因是浏览器从前端服务器，即端口号为 5173 的 URL 中得到页面，根据浏览器的同域原则，只有协议名、主机名、端口号三者一致才被认为浏览器属于同一个域，而页面中 axios 访问的是 8090 端口对应的 hrsys\_restful 项目，故基于同域的安全原则，浏览器不允许程序通过 AJAX 方式访问。

本章会使用 CROS 方式解决跨域问题，只需要在 hrsys\_restful 项目所要访问的 Controller 层的类上加上 @CrossOrigin 注解即可。

```
@CrossOrigin
@RestController
@RequestMapping("emp")
public class EmployeeController {
    //略
}
```

此时，项目数据可以正常显示。

### 13.1.4 Index 组件

在 components 下新建 index.vue 组件，并使用 <router-view> 标签替代原来的 iframe，使用 <router-link> 标签替换原来的超链接。

```
<script lang="ts">
export default {
  name: "Index",
  mounted() {
    this.$router.push("/emp/show");
  }
}
</script>
<template>
  <div id="container">
    <div id="top">
      <div id="logo">Alan人事管理系统</div>
    </div>
    <div id="main">
      <div id="left">
        <div class="yi">员工管理</div>
        <ul class="er">
          <li>
            <router-link to="/emp/show">员工管理</router-link>
          </li>
          <li>
            <router-link to="/emp/add">员工添加</router-link>
          </li>
        </ul>
        <div class="yi">部门管理</div>
        <ul class="er">
          <li><router-link to="/dep/show">部门管理</router-link></li>
          <li><router-link to="/dep/add">部门添加</router-link></li>
        </ul>
        <div class="yi">权限管理</div>
        <ul class="er">
```

```

                <li><router-link to="/sysUser/show">用户管理</router-link>
</li>
                <li><router-link to="/sysRole/show">角色管理</router-link>
</li>
                <li><router-link to="/sysPermission/show">权限管理</router-
link></li>
            </ul>
        </div>
        <div id="right">
            <router-view/>
        </div>
    </div>
    <div id="bottom"></div>
</div>
</template>

```

```

<style scoped>
    * {
        margin: 0;
        padding: 0;
    }

    #top, #bottom {
        clear: both;
        height: 80px;
        background: #337ab7;
        height: 80px;
    }

    #top #logo {
        color: #fff;
        font-size: 30px;
        font-weight: bold;
        padding: 15px 0 0 40px;
    }

    #main {
        margin-top: 10px;
        overflow: hidden;
    }

    a {
        color: #fff;
        text-decoration: none;
    }

    #left {
        width: 240px;
        float: left;
        height: 460px;
        padding-left: 20px;
    }

    #right {
        width: 800px;
        float: left;
        height: 460px;
    }

```

```
}

.yi {
  width: 160px;
  height: 40px;
  background: #337ab7;
  color: #fff;
  margin-top: 5px;
  border-radius: 3px;
  text-align: center;
  line-height: 40px;
}

.er {
  list-style: none;
  width: 160px;
}

.er li {
  margin-top: 2px;
  text-align: center;
  background: #337aae;
  color: #fff;
  height: 30px;
  line-height: 30px;
  font-size: 14px;
}
</style>
```