

基于Element的前端管理端与Java后端的联调(上)

1、前端管理端的题目pagination的实现：

The screenshot shows a management interface titled "Quiz后台管理". On the left, there is a sidebar with "管理选项" and two menu items: "用户管理" (highlighted in purple) and "题目管理" (highlighted in blue). The main area has a search bar with placeholder "请输入用户名" and buttons for "查询" (blue) and "添加用户" (green). Below the search bar is a table with columns: 日期 (Date), 姓名 (Name), and 操作 (Operations). The table contains four rows, each with a date (2016-05-02, 04, 01, 03) and name (王小虎). For each row, there are "编辑" (Edit) and "删除" (Delete) buttons. At the bottom of the table is a pagination bar with buttons for <, 1, 2, 3, 4, 5, 6, ..., 100, and >.

The screenshot shows another management interface titled "Quiz后台管理". The sidebar has "管理选项" and "题目管理" (highlighted in blue). The main area has a search bar with placeholder "请输入题目关键词" and buttons for "查询" (blue) and "添加题目" (green). Below the search bar is a table with columns: 序号 (Index), 题目 (Question), 选项 (Options), 答案 (Answer), and 操作 (Operations). The table contains four rows, each with a question about the capital of France. For each row, there are "编辑" (Edit) and "删除" (Delete) buttons. At the bottom of the table is a pagination bar with buttons for <, 1, 2, 3, 4, 5, 6, ..., 100, and >.

Vue异步向后端加载数据：

1、安装axios：

```
1 | npm install axios
```

安装后，重启前端Vue项目；

2、当前使用axios的页面，导入axios包：

```
1 | import axios from 'axios';
```

在Vue对象的钩子方法，mounted中请求数据，以我们的QuestionView.Vue页面为例：

```

1 //scripts中的mounted方法中，添加axios访问服务器api;
2 <script>
3     mounted(){
4         axios.get("/questions").then((response) => {
5             // 假设返回的数据是一个数组
6             console.log(response.data.data.qsBeanList);
7             this.tableData = response.data.data.qsBeanList; // 更新表格数据
8         }).catch((error) => {
9             console.error("Error fetching questions:", error);
10        });
11    }
12 </script>

```

注意：这里的localhost:8080是我们Java后台的接口地址，

与Vue前端的接口8080冲突，

所以我们修改下前端的端口，比如改成8081；具体在vue.config.js中修改端口号：

```

1 const { defineConfig } = require('@vue/cli-service')
2 module.exports = defineConfig({
3     transpileDependencies: true,
4     devServer: {
5         port: 8081
6     }
7 })

```

3、跨域问题：

因为前后端的端口不一致，因此访问时，会发生跨域问题，具体的错误信息如下：

```

1 :8081/#/question:1 Access to XMLHttpRequest at
'http://localhost:8080/getQuestion' from origin 'http://localhost:8081' has
been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is
present on the requested resource.

```

在开发环境中，前端代码通过配置代理的方式解决：

- 配置代理：在 [vue.config.js](#) 文件中设置代理，将所有以 / 开头的请求转发到 <http://localhost:8080>。

```

1 const { defineConfig } = require('@vue/cli-service')
2 module.exports = defineConfig({
3     transpileDependencies: true,
4     devServer: {
5         port: 8081,
6         proxy: {
7             '/': {
8                 target: 'http://localhost:8080',
9                 changeOrigin: true,
10                pathRewrite: {
11                    '^/': ''
12                }
13            }
14        }
15    }
}

```

2. 修改API请求地址：在 `QuestionView.vue` 文件中，将 `axios` 请求的URL从绝对路径 `http://localhost:8080/getQuestion` 修改为相对路径 `/getQuestion`，以便让代理生效。

```

1  axios.get("/questions").then((response) => {
2      // 假设返回的数据是一个数组
3      console.log(response.data.data.qsBeanList);
4      this.tableData = response.data.data.qsBeanList; // 更新表格数据
5  }).catch((error) => {
6      console.error("Error fetching questions:", error);
7  });

```

同时，还需要修改 `QuestionView.vue` 中列表中的元素：

```

1      <el-table :data="tableData" style="width: 80%">
2          <el-table-column label="序号" width="50">
3              <template slot-scope="scope">
4                  <!-- <i class="el-icon-time"></i> -->
5                  <span style="margin-left: 10px">{{ scope.row.id }}</span>
6                  <div slot="reference" class="name-wrapper">
7                      <span style="margin-left: 10px">{{ scope.row.questionText }}</span>
8                  <span style="margin-left: 10px">{{ scope.row.options }}</span>
9              </template>
10             <span style="margin-left: 10px">{{ scope.row.answer }}</span>
11         </el-table-column>
12     </el-table>

```

上述的四个元素：

- `scope.row.id`,
- `scope.row.questionText`, //此处原来的question要对应地修改为questionText;
- `scope.row.options`,
- `scope.row.answer`,

最终的测试结果：

序号	题目	选项	答案	操作
1	What is the capital of France?	["Paris", "London", "Berlin", "Madrid"]	Paris	<button>编辑</button> <button>删除</button>
2	What is the capital of France?	["Paris", "London", "Berlin", "Madrid"]	Paris	<button>编辑</button> <button>删除</button>
3	which of the following is not a programming language?	["Java", "Apple", "Python", "JavaScript"]	Apple	<button>编辑</button> <button>删除</button>
4	which of the following is not a programming language?	["Java", "Apple", "Python", "JavaScript"]	Apple	<button>编辑</button> <button>删除</button>
5	which of the following is not a programming language?	["Java", "Apple", "Python", "JavaScript"]	Apple	<button>编辑</button> <button>删除</button>

< 1 2 3 4 5 ... 100 >

当前页面存在的两个问题：

- 序号还不能显示；

- 页码不对；一共7个问题，每页显示5个，应该只有两页数据；

主要的原因：

- Java后端返回的数据中，没有将id返回；
- 前端的分页组件没有接收参数；

前端分页组件的修改：

```

1 //定义pageSize和total两个变量:
2 return {
3     dialogFormVisible: false,
4     pageSize: 5, // 每页显示的条数
5     total: 0, // 总条数
6     ...
7 }
8
9 //组件中绑定变量:
10 <el-pagination
11   background
12   layout="prev, pager, next"
13   :page-size="pageSize"
14   :total="total"
15 >
16 </el-pagination>
17
18 //mounted函数中，设置total值:
19 mounted() {
20   axios.get("/questions").then((response) => {
21     const data = response.data.data.qsBeanList;
22     this.tableData = data; // 更新表格数据
23     this.total = response.data.data.total; // 动态设置总条数

```

题目id的显示：

检查qsBeanList对象，不存在id属性；

```

1 public class QSBeanOutManage {
2     private String questionText;
3     private List<String> options;
4     private String answer;
5 }

```

1、在QSBeanOutManage中，增加id属性：

```

1 public class QSBeanOutManage {
2     private Integer id; //新增;
3     private String questionText;
4     private List<String> options;
5     private String answer;
6 }

```

2、在Tool类的convertToQSBeanManageList方法中，添加对应的get/set方法；

```
1 | bean.setQuestionText(question.getQuestionText());
2 | //设置id;
3 | bean.setId(question.getId());
```

做完上述修改以后：

序号	题目	选项	答案	操作
1	What is the capital of France?	["Paris", "London", "Berlin", "Madrid"]	Paris	<button>编辑</button> <button>删除</button>
2	What is the capital of France?	["Paris", "London", "Berlin", "Madrid"]	Paris	<button>编辑</button> <button>删除</button>
3	which of the following is not a programming language?	["Java", "Apple", "Python", "JavaScript"]	Apple	<button>编辑</button> <button>删除</button>
4	which of the following is not a programming language?	["Java", "Apple", "Python", "JavaScript"]	Apple	<button>编辑</button> <button>删除</button>
5	which of the following is not a programming language?	["Java", "Apple", "Python", "JavaScript"]	Apple	<button>编辑</button> <button>删除</button>

pagination 中的页面请求：

当前点击不同的页面，没有反应，所以要给不同的页码按钮增加点击事件，加载不同页码对应的题目数据。

1、pagination中增加变量和方法：

```
1 | <el-pagination
2 |   background
3 |   layout="prev, pager, next"
4 |   :page-size="pageSize"
5 |   :total="total"
6 |   //新增变量以及绑定方法;
7 |   :current-page="currentPage"
8 |   @current-change="handlePageChange"
9 | />
```

2、在js脚本中的修改：

```
1 | export default {
2 |   data() {
3 |     return {
4 |       currentPage: 1,
5 |       pageSize: 5,
6 |       total: 0,
7 |       tableData: [],
8 |     };
9 |   },
10 |   methods: {
11 |     handlePageChange(page) {
12 |       this.currentPage = page;
13 |       axios
```

```

14     .get(`/questions?
15     page=${this.currentPage}&pageSize=${this.pageSize}`)
16     .then((response) => {
17       this.tableData = response.data.data.qsBeanList;
18       this.total = response.data.data.total;
19     })
20     .catch((error) => {
21       console.error("Error fetching questions:", error);
22     });
23   },
24   mounted() {
25     // 初始化第一页数据
26     this.handlePageChange(1);
27   },

```

经过上述的修改后，可以点击页面来实现页面的切换。

2、前端管理端的题目查询



上述输入框中，输入题目的关键词，再调用/findQuestion接口获取相关的题目；

a、在输入框组件中，设定keyword:

```

1 //组件:
2 <el-form-item label="题目">
3   <el-input
4     v-model="formInline.keyword" //改为keyword
5     placeholder="请输入题目关键词"
6   ></el-input>
7 </el-form-item>
8
9   //绑定onSearch()方法;
10 <el-form-item>
11   <el-button type="primary" @click="onSearch">查询</el-button>
12 </el-form-item>
13
14 //脚本中;
15   formInline: {
16     keyword: '',
17     // region: '',
18   },
19
20 onSearch() {
21   // 这里可以添加查询逻辑
22   console.log("Searching for:", this.formInline.keyword);
23   // 例如，调用API获取数据
24   axios
25     .get(`/findQuestion?keyword=${this.formInline.keyword}`)
26     .then((response) => {

```

```

27         this.tableData = response.data.data;
28         console.log("Search results:", response.data);
29         // this.total = response.data.data.total;
30     })
31     .catch((error) => {
32       console.error("Error searching questions:", error);
33     });
34   },

```

结果：

The screenshot shows a search interface at the top with a '题目' input field containing 'France', a '查询' button, and a '添加题目' button. Below is a table with columns: 序号 (Index), 题目 (Question), 选项 (Options), 答案 (Answer), and 操作 (Operations). The table contains three rows of data, each with a red box around the '题目' column. The first row's question is 'What is the capital of France?'. The second row's question is also 'What is the capital of France?'. The third row's question is 'What is the capital of France2?'. The '操作' column for each row has two buttons: '编辑' (Edit) and '删除' (Delete).

序号	题目	选项	答案	操作
1	What is the capital of France?	["Paris", "London", "Berlin", "Madrid"]	Paris	<button>编辑</button> <button>删除</button>
2	What is the capital of France?	["Paris", "London", "Berlin", "Madrid"]	Paris	<button>编辑</button> <button>删除</button>
6	What is the capital of France2?	["Paris", "London", "Berlin", "Madrid"]	Paris	<button>编辑</button> <button>删除</button>

练习作业：

- 1、关键词查询，启用分页；小于5条，只有1页；大于5条，只显示5条；
- 2、当前查询后，input这边不能自动清空，建议可以增加查询后清空的功能。

3、前端管理端的题目添加

- 1、在组件中，添加onSubmitQuestion事件绑定；

```

1 | <el-button type="primary" @click="onAddNewQuestion">确 定</el-button>

```

2、js脚本中实现事件处理函数：

```

1 | export default {
2 |   data() {
3 |     return {
4 |       dialogFormVisible: false,
5 |       form: {
6 |         question: '',
7 |         optiona: '',
8 |         optionb: '',
9 |         optionc: '',
10 |        optiond: '',
11 |        answer: '',
12 |      },
13 |    };
14 |  },
15 |  methods: {
16 |    onAddNewQuestion() {
17 |      console.log("Submitting question:", this.form);
18 |      axios
19 |        .post("/addQuestion", this.form)

```

```
20     .then((response) => {
21       console.log("Question added successfully:", response.data);
22       this.dialogFormVisible = false;
23       this.handlePageChange(this.currentPage);
24     })
25     .catch((error) => {
26       console.error("Error adding question:", error);
27     });
28   },
29 },
30 );
31 </script>
```

出现的问题：

前端的确有数据发送到后端；

```
1  onSubmitQuestion() {
2    console.log("Submitting question:", this.form);
3    axios
4      .post("/addQuestion", this.form)
```

但是后端没有正确接收到数据；

```
1  @PostMapping("/addQuestion")
2  public Result addQuestion(QSBean question) {
3    System.out.println(question);
```

可能的原因：前端发送的数据格式与后端的接收方式不一致。

如何解决：

前后端统一数据格式，均采用json格式：前端按json格式发送数据，后端按json格式接收及解析数据。

```
1  .post("/addQuestion", this.form, {
2    headers: {
3      "Content-Type": "application/json",
4    },
5  })
```

```
1  @PostMapping("/addQuestion")
2  public Result addQuestion(@RequestBody QSBean question) {
3    System.out.println(question);
4    return Result.success("Question added successfully");
5  }
```

经过上述的修改，可以正常地插入数据(插入到最后)

作业和练习：

前端根据后端的返回，弹出一个对话框，告知用户，添加新问题成功。

4、前端管理端的题目删除

1、删除按钮绑定事件函数；

```
1      <el-button
2          size="mini"
3          type="danger"
4          @click="handleDelete(scope.$index, scope.row)"
5          >删除</el-button
6      >
```

2、在js脚本中，书写删除命令对应的函数：

```
1 handleDelete(index, row) {
2     const id = row.id; // 获取当前条目的 id
3     this.$confirm("此操作将永久删除该题目，是否继续?", "提示", {
4         confirmButtonText: "确定",
5         cancelButtonText: "取消",
6         type: "warning",
7     })
8     .then(() => {
9         // 调用删除接口（GET 请求）
10        axios
11            .get(`/delQuestion?id=${id}`) // 使用 GET 请求传递 id 参数
12            .then((response) => {
13                console.log(response.data);
14                // 删除成功后刷新当前页数据
15                this.handlePageChange(this.currentPage);
16            })
17            .catch((error) => {
18                console.error("Error deleting question:", error);
19                this.$message({
20                    type: "error",
21                    message: "删除失败，请稍后重试!",
22                });
23            });
24    })
```

```
25     .catch(() => {
26       this.$message({
27         type: "info",
28         message: "已取消删除",
29       });
30     });
31   },
```

处理函数中需要注意的：

- axios.get(`/delQuestion?id=\${id}`) // 使用 GET 请求传递 id 参数

作业和练习：

1	What is the capital of France?	["Paris", "London", "Berlin", "Madrid"]	Paris	<button>编辑</button>	<button>删除</button>
2	What is the capital of France?	["Paris", "London", "Berlin", "Madrid"]	Paris	<button>编辑</button>	<button>删除</button>
3	which of the following is not a programming language?	["Java", "Apple", "Python", "JavaScript"]	Apple	<button>编辑</button>	<button>删除</button>

实现"编辑"按钮的功能，点击后弹出对话框(类似于添加用户按钮)，

可以对questionText，各个optionText和answer的内容进行修改更新。