2.1 案例——学习计划表

2.1.1 准备工作

在开发“学习计划表”案例之前，需要先完成一些准备工作，具体步骤如下。

① 打开命令提示符，切换到 D:\vue\chapter02 目录，在该目录下执行如下命令，创建项目。

```
yarn create vite learning_schedule --template vue
```

项目创建完成后，执行如下命令进入项目目录，启动项目。

```
cd learning_schedule
yarn
yarn dev
```

项目启动后，可以使用 URL 地址 http://127.0.0.1:5173/进行访问。

② 使用 VS Code 编辑器打开 D:\vue\chapter02\learning_schedule 目录。

③ 将 src\style.css 文件中的样式代码全部删除，从而避免项目创建时自带的样式代码影响本案例的实现效果。

④ 打开 D:\vue\chapter02\ learning_schedule 目录，找到 index.html 文件。在文件中引入 BootStrap 样式文件，具体代码如下。

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    ……（原有代码）
5    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
6  </head>
7  </html>
```

上述代码中，第 5 行代码引入 Bootstrap 的 CSS 文件，引用后即可使用 Bootstrap 快速开发响应式网页，使用全局 CSS 样式美化标签。

⑤ 修改 src\App.vue 文件中的默认内容，具体代码如下。

```
1  <template>
2    学习计划表
3  </template>
```

至此，“学习计划表”案例准备工作已完成。

2.1.2 渲染表格区域数据

接下来正式进入“学习计划表”案例的开发。在 App 组件中编写表格区域的页面结构和样式，并在页面上渲染表格数据，具体实现步骤如下。

① 在<script setup>标签中定义渲染表格所需的数据，具体代码如下。

```
<script setup>
import { ref } from 'vue'
const list = ref([
  {
    id: '1',
    subject: 'Vue.js 前端实战开发',
    content: '学习指令，例如 v-if、v-for、v-model 等',
    place: '自习室',
    status: false,
  },
])
</script>
```

上述代码中，list 数组表示渲染表格区域所需的数据，id 属性表示序号，subject 属性表示学习科目，content 属性表示学习内容，place 属性表示学习地点，status 属性表示学习计划的完成状态，若属性值为 false，表示“未完成”，若属性值为 true 表示“已完成”。

② 根据 Bootstrap 文档找到 Tables，根据实际需要合适的样式，本案例中表格的结构样式具体如下。

```
<template>
  <table class="table table-striped table-hover table-bordered">
    <thead>
      <tr>
        <th scope="col">序号</th>
        <th scope="col">学习科目</th>
        <th scope="col">学习内容</th>
        <th scope="col">学习地点</th>
        <th scope="col">完成状态</th>
        <th scope="col">操作</th>
      </tr>
    </thead>
    <tbody>
      <tr>
```

```
15        <td>序号</td>
16        <td>学习科目</td>
17        <td>学习内容</td>
18        <td>学习地点</td>
19        <td>完成状态</td>
20        <td>操作</td>
21      </tr>
22    </tbody>
23  </table>
24 </template>
```

保存上述代码后，在浏览器中打开 http://127.0.0.1:5173/，初始页面如图 2-1 所示。

图2-1　初始页面

③　接下来通过 v-for 指令循环渲染表格行的数据，修改<tbody>标签中的代码，具体如下。

```
1  <tbody>
2    <tr v-for="item in list" :key="item.id">
3      <td>{{ item.id }}</td>
4      <td>{{ item.subject }}</td>
5      <td>{{ item.content }}</td>
6      <td>{{ item.place }}</td>
7      <td>
8        <div class="form-check form-switch">
9          <input class="form-check-input" type="checkbox" role="switch"
id="flexSwitchCheckDefault" v-model="item.status" />
10         <label class="form-check-label" for="flexSwitchCheckDefault" v-
if="item.status">已完成</label>
11         <label class="form-check-label" for="flexSwitchCheckDefault" v-
else>未完成</label>
12       </div>
13     </td>
14     <td>
15       <a href="javascript:;">删除</a>
```

```
16      </td>
17    </tr>
18 </tbody>
```

上述代码中，第 2~17 行代码通过 v-for 指令渲染表格行，将 list 数组中的数据渲染到页面上，每项必须提供一个唯一 key 值；第 3~6 行代码将列表项中序号、学习科目、学习内容、学习地点通过 Mustache 语法渲染到页面上；第 7~13 行代码将表格行渲染为 switch 开关效果，其中，第 8 行代码将<input>标签渲染为开关的效果，通过 v-model 指令绑定绑定 data 中的 list 数组中每个对象的 status 属性，第 10~11 行代码通过 v-if、v-else 条件渲染指令根据 status 属性的属性值渲染“已完成”或“未完成”的<label>标签；第 15 行代码定义了<a>标签，通过单击删除字段，即可将该行学习计划删除。

保存上述代码后，在浏览器中打开 http://127.0.0.1:5173/，渲染表格数据的页面效果如图 2-2 所示。

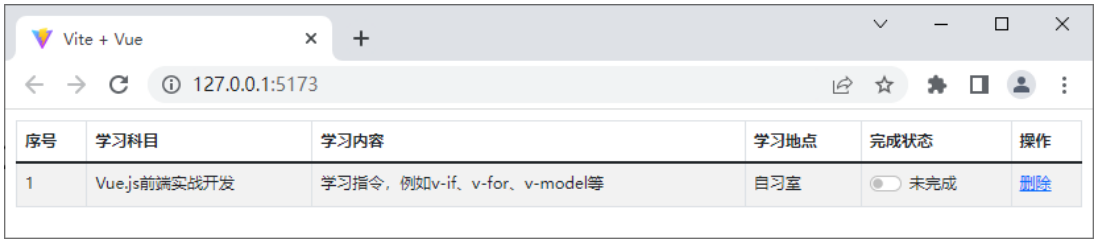

图2-2　渲染表格数据的页面效果

从图 2-2 中可以看出，学习计划列表中的数据被成功渲染。

至此，学习计划表渲染表格区域数据已实现。

2.1.3　实现学习计划的删除功能

在单击表格行最后一列“删除”时，可以对整行的学习计划进行删除操作。在实现该功能时，在单击“删除”时，传递该行数据所对应的 id 属性，通过调用数组中的 filter()方法实现数据的过滤。在删除学习计划时，如果完成状态为“未完成”时禁止删除，完成状态为“已删除”时该学习计划可以进行删除操作。实现学习计划删除功能的具体步骤如下。

①　修改<a>标签，添加单击事件，具体代码如下。

```
<a href="javascript:;" @click="remove(item.id)">删除</a>
```

上述代码中，加粗代码为新增代码，为<a>标签绑定单击事件，并传递参数 id。

②　在<script setup>标签中编写 remove()方法，实现学习计划的删除，具体代码如下。

```
1  let remove = id => {
2    list.value = list.value.filter(item => item.id !== id)
3  }
```

上述代码中，第 2 行代码通过调用 filter()方法创建一个新数组，新数组为通过检查指定 list 数组中符合条件的所有元素。

③ 修改<a>标签，在单击“删除”时传递该学习计划的完成状态，具体代码如下。

```
<a href="javascript:;" @click="remove(item.id, item.status)">删除</a>
```

④ 修改 remove()方法，实现对学习计划完成状态的判断，具体代码如下。

```
1 let remove = (id, status) => {
2   if (status) {
3     list.value = list.value.filter(item => item.id !== id)
4   } else {
5     alert('请完成该学习计划后再进行删除操作！')
6   }
7 }
```

上述代码中，通过 if 进行判断，若 status 为 true，则执行第 3 行代码进行删除操作，否则执行第 5 行代码，弹出警告。

保存上述代码后，在浏览器中打开 http://127.0.0.1:5173/进行测试。

至此，学习计划表的删除功能已实现。

2.1.4 实现学习计划的添加功能

接下来实现添加学习计划的功能，学生可以自主填写学习科目、学习内容和学习地点。该功能通过表单来进行实现，在单击“添加”按钮时实现表单中所有信息的提交，进行添加学习计划的操作。实现学习计划的添加功能具体实现步骤如下。

① 首先在<script setup>标签中定义页面所需的数据，具体代码如下。

```
1 let subject = ref('')
2 let content = ref('')
3 let nextId = ref('')
4 let selectedOption = ref('自习室')
5 let options = ref([
6   { placeCode: 0, place: '自习室', },
7   { placeCode: 1, place: '图书馆', },
8   { placeCode: 2, place: '宿舍', },
9 ])
```

上述代码中，第 1 行代码定义了 subject，表示学习科目；第 2 行代码定义了 content，表示学习内容；第 3 行代码定义了 nextId，表示新添加数据的 id；第 4 行代码定义了默认选中的学习地点；第 5~9 行代码定义了 options 数组，表示学习地点，其中 placeCode 属性表示地点编号，place 属性表示地点名称。

② 卡片区域的整体页面结构，具体代码如下。

```
1  <template>
2    <div class="card">
3      <!-- 标题区域 -->
4      <div class="card-header">学习计划表</div>
5      <!-- 提交区域 -->
6      <div class="card-body">
7        <form>
8          <div class="row g-4">
9            <!-- 学习科目 -->
10           <div class="col-auto"></div>
11           <!-- 学习任务 -->
12           <div class="col-auto"></div>
13           <!-- 学习地点 -->
14           <div class="col-auto"></div>
15           <!-- 添加按钮 -->
16           <div class="col-auto">
17             <button type="submit" class="btn btn-primary">添加</button>
18           </div>
19         </div>
20       </form>
21     </div>
22   </div>
23   <table class="table table-striped table-hover table-bordered">
24     ……（原有代码）
25   </table>
26 </template>
```

上述代码中，第 2~22 行代码为新增代码，表示卡片区域。第 4 行代码定义了标题区域，名称为“学习计划表”，第 7~20 行代码定义了表单区域，实现学习计划的添加功能。第 10、12、14 行代码分别定义了学习科目、学习任务、学习地点区域，这 3 个区域的实现在后续步骤中进行讲解。第 16~18 行代码中，定义了“添加”按钮，type 类型为 submit，当单击该按钮时会触发<form>标签的 submit 事件提交表单信息。

③ 添加卡片区域中学习科目区域的页面结构，具体代码如下。

```
1  <div class="col-auto">
2    <div class="input-group mb-3">
3      <span class="input-group-text" id="basic-addon1">学习科目</span>
4      <input type="text" class="form-control" placeholder="请输入学习科目"
v-model.trim="subject" />
5    </div>
```

```
6  </div>
```

上述代码中，第 2~5 行代码定义了学习科目区域，通过<input>标签定义的文本框可以输入学习科目，通过 v-model 指令，将<input>标签与 subject 实现数据的双向绑定，即当更改文本框中的值时 subject 的值更改。同时为 v-model 指令添加了 trim 修饰符，用于自动过滤用户输入的首尾空白字符。

④ 添加卡片区域中学习任务区域的页面结构，具体代码如下。

```
1  <div class="col-auto">
2    <div class="input-group mb-3">
3      <span class="input-group-text" id="basic-addon1">学习内容</span>
4      <textarea class="form-control" v-model.trim="content" placeholder="
请输入学习内容" :style="{ height: '32px' }"></textarea>
5    </div>
6  </div>
```

上述代码中，第 2~5 行代码定义了学习内容区域，通过<textarea>标签定义的多行文本框可以输入学习内容，通过 v-model 指令与 content 绑定，实现视图与数据的双向绑定。同时为 v-model 指令添加了 trim 修饰符，用于自动过滤用户输入的首尾空白字符。

⑤ 添加卡片区域中学习地点区域的页面结构，具体代码如下。

```
1  <div class="col-auto">
2    <div class="input-group mb-3">
3      <span class="input-group-text" id="basic-addon1">学习地点</span>
4      <select class="form-select form-select-sm" v-model="selectedOption">
5        <option v-for="option in
options" :value="option.place" :key="option.placeCode">
6          {{ option.place }}
7        </option>
8      </select>
9    </div>
10 </div>
```

上述代码中，第 2~9 行代码定义了学习地点区域，其中，第 4~8 行代码通过<select>标签定义了下拉列表，通过 v-model 指令与 selectedOption 绑定，实现数据的双向绑定。如果 v-model 指令的初始值不匹配任何一个选项，<select>标签会渲染成未选择的状态，所以 selectedOption 属性值为“自习室”，表示<select>标签的初始值为“自习室”，自习室为下拉列表中定义的一个值。第 5~7 行代码通过 v-for 条件渲染指令实现学习地点的渲染，当下拉列表中选项改变时，selectedOption 属性更改。

⑥ 为<form>标签添加 submit 事件，实现单击“添加”按钮时添加信息，具体代码如下。

```
    <form @submit.prevent="add">
      ……（原有代码）
```

```
</form>
```

上述代码中，黑色加粗部分为新增代码，添加事件处理函数为 add()，同时为 submit 事件添加事件修饰符 prevent，用来阻止表单的默认提交行为。实现在单击按钮后提交表单，执行 add()方法。

⑦ 在<script setup>标签中编写 add()方法，实现学习计划的添加，具体代码如下。

```
let add = () => {
  if (subject.value === '') {
    alert('学习科目为必填项！')
    return
  }
  nextId.value = Math.max(...list.value.map(item => item.id)) + 1
  const obj = {
    id: nextId.value,
    subject: subject.value,
    content: content.value,
    place: selectedOption.value,
    status: false,
  }
  list.value.push(obj)
  subject.value = ''
  content.value = ''
  selectedOption.value = '自习室'
}
```

上述代码中，第 2~5 行代码通过 if 进行判断，若学习科目字段 subject 为空，则弹出提示，且不执行接下来的添加操作；第 6 行代码，通过调用 max()方法，找到 list 数组中 id 中最大值，新添加数据的 id 为最大值加 1；第 7~13 行代码为需要新添加的数据，包括 id、subject、content、place 等，在默认情况下 status 为 false，表示未完成该学习计划；第 14 行代码调用 push()方法实现将 obj 对象添加到 list 数组的末尾；第 15~16 行代码将 subject、content 中的数据清空；第 17 行代码定义下拉列表的默认值。

保存上述代码后，在浏览器中打开 http://127.0.0.1:5173/，查看页面效果。

至此，学习计划表的添加功能已实现。

2.1.5 实现状态的切换功能

在实现学习计划的添加功能后，可以添加多条学习计划。在之前遗留如下问题：在表格中更改添加后的几条学习计划的完成状态时，单击<input>标签生成的开关按钮可以实现

“未完成”与“已完成”状态之间的切换，但是单击<label>标签“未完成”与“已完成”不能实现该表格行的完成状态的切换，只会切换表格第 1 行的完成状态。

若想实现单击文字也可实现某条学习计划完成状态的切换，则可以通过动态生成<input>标签的 id 属性，修改完成状态列的代码，具体如下。

```
1  <div class="form-check form-switch">
2    <input class="form-check-input" type="checkbox" role="switch" :id="'cb' + item.id" v-model="item.status" />
3    <label class="form-check-label" :for="'cb' + item.id" v-if="item.status">已完成</label>
4    <label class="form-check-label" :for="'cb' + item.id" v-else>未完成</label>
5  </div>
```

上述代码中，黑体加粗部分代码为修改代码，<label>标签中的 for 属性规定<label>标签与哪个表单元素绑定，所以在第 3~3 行代码中通过<label>标签中 for 属性与<input>标签绑定，for 属性的属性值为<input>标签的 id 属性值，实现单击<label>标签中的文本时，浏览器自动将焦点转到和<label>标签的相关控件<input>控件中，实现单击文本完成某条学习计划的完成状态的改变。

保存上述代码后，在浏览器中打开 http://127.0.0.1:5173/进行测试。

至此，“学习计划表”案例开发完成。