讲师：正心

b站账号：正心全栈编程

视频地址：https://www.bilibili.com/video/BV1Jr4y1771i

源码文档：加微信 zhengxinonly 免费获取

从零开始用 flask + vue 做一个前后端分离的案例

本个案例，使用 flask、vue、elements-plus、axios 制作一个基于 restful api 的前后端分离的图书信息管理案例

后端部分

flask: https://flask.palletsprojects.com/en/2.1.x/

flask-sqlalchemy: https://flask-sqlalchemy.palletsprojects.com/en/2.x/

flask-cors: https://flask-cors.readthedocs.io/en/latest/

flask 快速上手

```
from flask import Flask, request


app = Flask(__name__)


@app.route('/')
def hello_world():  # put application's code here
    return 'Welcome Books!'
```

数据库部分

```
# -*- coding: utf-8 -*-
from extension import db


class Book(db.Model):
    __tablename__ = 'book'
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    book_number = db.Column(db.String(255), nullable=False)
    book_name = db.Column(db.String(255), nullable=False)
    book_type = db.Column(db.String(255), nullable=False)
    book_prize = db.Column(db.Float, nullable=False)
    author = db.Column(db.String(255))
    book_publisher = db.Column(db.String(255))
```

```
    @staticmethod
    def init_db():
        rets = [
            (1, '001', '活着', '小说', 39.9, '余华', '某某出版社'),
            (2, '002', '三体', '科幻', 99.8, '刘慈欣', '重庆出版社')
        ]
        for ret in rets:
            book = Book()
            book.id = ret[0]
            book.book_number = ret[1]
            book.book_name = ret[2]
            book.book_type = ret[3]
            book.book_prize = ret[4]
            book.author = ret[5]
            book.book_publisher = ret[6]
            db.session.add(book)
        db.session.commit()
```

使用之前需要 flask create 初始化一下数据

接口部分

RESTful API 最佳实践(阮一峰) : https://www.ruanyifeng.com/blog/2018/10/restful-api-best-practices.html

Method Views for APIs: https://flask.palletsprojects.com/en/2.1.x/views/#method-views-for-apis

```
from flask import Flask, request
from flask_cors import CORS
from flask.views import MethodView

from extension import db
from models import Book

app = Flask(__name__)
CORS().init_app(app)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///books.sqlite'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db.init_app(app)


@app.cli.command()
def create():
    db.drop_all()
    db.create_all()
    Book.init_db()


@app.route('/')
def hello_world():  # put application's code here
    return 'Welcome Books!'
```

```python
class BookApi(MethodView):
    def get(self, book_id):
        if not book_id:
            books: [Book] = Book.query.all()
            results = [
                {
                    'id': book.id,
                    'book_name': book.book_name,
                    'book_type': book.book_type,
                    'book_prize': book.book_prize,
                    'book_number': book.book_number,
                    'book_publisher': book.book_publisher,
                    'author': book.author,
                } for book in books
            ]
            return {
                'status': 'success',
                'message': '数据查询成功',
                'results': results
            }
        book: Book = Book.query.get(book_id)
        return {
            'status': 'success',
            'message': '数据查询成功',
            'result': {
                'id': book.id,
                'book_name': book.book_name,
                'book_type': book.book_type,
                'book_prize': book.book_prize,
                'book_number': book.book_number,
                'book_publisher': book.book_publisher,
                'author': book.author,
            }
        }

    def post(self):
        form = request.json
        book = Book()
        book.book_number = form.get('book_number')
        book.book_name = form.get('book_name')
        book.book_type = form.get('book_type')
        book.book_prize = form.get('book_prize')
        book.author = form.get('author')
        book.book_publisher = form.get('book_publisher')
        db.session.add(book)
        db.session.commit()
        # id, book_number, book_name, book_type, book_prize, author,
book_publisher

        return {
            'status': 'success',
            'message': '数据添加成功'
        }

    def delete(self, book_id):
```

```
        book = Book.query.get(book_id)
        db.session.delete(book)
        db.session.commit()
        return {
            'status': 'success',
            'message': '数据删除成功'
        }

    def put(self, book_id):
        book: Book = Book.query.get(book_id)
        book.book_type = request.json.get('book_type')
        book.book_name = request.json.get('book_name')
        book.book_prize = request.json.get('book_prize')
        book.book_number = request.json.get('book_number')
        book.book_publisher = request.json.get('book_type')
        book.author = request.json.get('book_type')
        db.session.commit()
        return {
            'status': 'success',
            'message': '数据修改成功'
        }


book_api = BookApi.as_view('book_api')
app.add_url_rule('/books', view_func=book_api, methods=['GET', ], defaults=
{'book_id': None})
app.add_url_rule('/books', view_func=book_api, methods=['POST', ])
app.add_url_rule('/books/<int:book_id>', view_func=book_api, methods=['GET',
'PUT', 'DELETE'])
```

前端部分

vite: https://vitejs.cn/

vue3: https://v3.cn.vuejs.org/

Element Plus: https://element-plus.gitee.io/zh-CN/

axios: https://axios-http.com/docs/intro

项目创建

```
C:\Users\xxp\Desktop>npm init vite@latest
√ Project name: ... book-fontend
√ Select a framework: » vue
√ Select a variant: » vue

Scaffolding project in C:\Users\xxp\Desktop\book-fontend...

Done. Now run:

  cd book-fontend
  npm install
  npm run dev
```

项目初始化

```
npm install element-plus
npm install axios
```

初始化 element-plus

```
import {createApp} from 'vue'
import App from './App.vue'
import ElementPlus from 'element-plus'
import 'element-plus/dist/index.css'


const app = createApp(App)
app.use(ElementPlus)
app.mount('#app')
```

页面创建

表单数据显示

```
<template>
  <div style="margin: 0 auto;width: 50%;">
    <h1 style="text-align: center">图书管理系统</h1>
    <!--  添加图书按钮 -->
    <el-button type="primary" @click="add_dialog_visible = true" size="small">添加图书</el-button>
    <!-- 数据表格 -->
    <el-table :data="books" style="margin: 20px auto;">
      <el-table-column label="编号" prop="book_number"/>
      <el-table-column label="书名" prop="book_name"/>
      <el-table-column label="类型" prop="book_type"/>
      <el-table-column label="价格" prop="book_prize"/>
      <el-table-column label="作者" prop="author"/>
```

```
      <el-table-column label="出版社" prop="book_publisher"/>
      <el-table-column align="right" label="操作" width="200px">
        <template #default="scope">
          <el-button size="small" @click="handleEdit(scope.$index, scope.row)">
            编辑
          </el-button>
          <el-button
              size="small"
              type="danger"
              @click="handleDelete(scope.$index, scope.row)"
          >
            删除
          </el-button
          >
        </template>
      </el-table-column>
    </el-table>
  </div>
</template>
```

```
<script setup>
import axios from 'axios'
import {reactive, ref, onMounted} from "vue";
import {ElMessageBox} from 'element-plus'


const books = reactive([])
const getStudents = () => {
  axios.get("http://localhost:5000/books",).then(res => {
    books.splice(0, books.length)
    books.push(...res.data.results)
    console.log('更新数据')
  })
}
// 页面渲染之后添加数据
onMounted(() => {
  getStudents()
})

// 删除数据
const handleDelete = (index, scope) => {
  axios.delete(`http://localhost:5000/books/${scope.id}`).then(() => {
    getStudents()
  })
}
</script>
```

添加数据

html表单

```
  <!-- 添加图书页面 -->
  <el-dialog
      title="添加图书"
```

```
      v-model="add_dialog_visible"
      width="30%"
      :before-close="handleClose"
  >
    <el-form
        ref="ruleFormRef"
        :model="book_form"
        status-icon
        label-width="120px"
        class="demo-ruleForm"
    >
      <el-form-item label="编号" prop="book_number">
        <el-input v-model="book_form.book_number" autocomplete="off"/>
      </el-form-item>
      <el-form-item label="书名" prop="book_name">
        <el-input v-model="book_form.book_name" autocomplete="off"/>
      </el-form-item>
      <el-form-item label="类型" prop="book_type">
        <el-input v-model="book_form.book_type" autocomplete="off"/>
      </el-form-item>
      <el-form-item label="价格" prop="book_prize">
        <el-input v-model.number="book_form.book_prize" autocomplete="off"/>
      </el-form-item>
      <el-form-item label="作者" prop="author">
        <el-input v-model="book_form.author" autocomplete="off"/>
      </el-form-item>
      <el-form-item label="出版社" prop="book_publisher">
        <el-input v-model="book_form.book_publisher" autocomplete="off"/>
      </el-form-item>
      <el-form-item>
        <el-button type="primary" @click="submitForm(ruleFormRef)">提交</el-
button>
        <el-button @click="resetForm(ruleFormRef)">重置</el-button>
      </el-form-item>
    </el-form>
  </el-dialog>
```

JavaScript

```
/*表单添加*/
const add_dialog_visible = ref(false)
const ruleFormRef = ref()
const book_form = reactive({
  book_number: "",
  book_name: "",
  book_type: "",
  book_prize: "",
  author: "",
  book_publisher: "",
  id: "",
})
// 表单提交事件
const submitForm = (formEl) => {
  axios.post('http://localhost:5000/books', book_form).then(() => {
    add_dialog_visible.value = false
```

```
    formEl.resetFields()
    getStudents()
  })
}
// 重置表单
const resetForm = (formEl) => {
  formEl.resetFields()
}

// 关闭弹窗前确认
const handleClose = (done) => {
  ElMessageBox.confirm('确认关闭？')
      .then(() => {
        done();
      })
      .catch(() => {
      });
}
```

编辑数据

html表单

```
<!-- 编辑图书页面 -->
<el-dialog
    title="编辑图书"
    v-model="edit_dialog_visible"
    width="30%"
    :before-close="handleClose"
>
  <el-form
      ref="editFormRef"
      :model="book_form"
      status-icon
      label-width="120px"
      class="demo-ruleForm"
  >
    <el-form-item label="编号" prop="book_number">
      <el-input v-model="book_form.book_number" autocomplete="off"/>
    </el-form-item>
    <el-form-item label="书名" prop="book_name">
      <el-input v-model="book_form.book_name" autocomplete="off"/>
    </el-form-item>
    <el-form-item label="类型" prop="book_type">
      <el-input v-model="book_form.book_type" autocomplete="off"/>
    </el-form-item>
    <el-form-item label="价格" prop="book_prize">
      <el-input v-model.number="book_form.book_prize" autocomplete="off"/>
    </el-form-item>
    <el-form-item label="作者" prop="author">
      <el-input v-model="book_form.author" autocomplete="off"/>
    </el-form-item>
    <el-form-item label="出版社" prop="book_publisher">
      <el-input v-model="book_form.book_publisher" autocomplete="off"/>
```

```
      </el-form-item>
      <el-form-item>
        <el-button type="primary" @click="submitEditForm(editFormRef)">提交</el-
button>
        <el-button @click="resetForm(editFormRef)">重置</el-button>
      </el-form-item>
    </el-form>
  </el-dialog>
```

JavaScript代码

```
/*编辑表单*/
const editFormRef = ref()
const edit_dialog_visible = ref(false)
const handleEdit = (index, scope) => {
  for (let key in scope) {
    book_form[key] = scope[key]
  }
  edit_dialog_visible.value = true
}
// 编辑提交按钮
const submitEditForm = (formEl) => {
  axios.put(`http://localhost:5000/books/${book_form.id}`, book_form).then((res)
=> {
    formEl.resetFields()
    edit_dialog_visible.value = false
    getStudents()
  })
}
```