link: null
title: 珠峰架构师成长计划
description: src/index.ts
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=60 sentences=565, words=3075

1.项目初始化

```
mkdir server
cd server
cnpm init -y
```

```
cnpm i express mongoose body-parser bcryptjs jsonwebtoken morgan cors validator helmet dotenv multer -S
cnpm i typescript  @types/node @types/express @types/mongoose @types/bcryptjs @types/jsonwebtoken  @types/morgan @types/cors @types/validator ts-node-dev
nodemon  @types/helmet @types/multer -D
```

模块名 用途

加载到环境变量

```
npx tsconfig.json
```

```
+  "scripts": {
+    "build": "tsc",
+    "start": "cross-env PORT=8000  ts-node-dev --respawn src/index.ts",
+    "dev": "cross-env PORT=8000 nodemon --exec ts-node --files src/index.ts"
+  }
```

```
node_modules
src/public/upload/
.env
```

```
JWT_SECRET_KEY=zhufeng
MONGODB_URL=mongodb:
```

2.用户管理

src/index.ts

```
import express, { Express, Request, Response, NextFunction } from 'express';
import mongoose from 'mongoose';
import HttpException from './exceptions/HttpException';
import cors from 'cors';
import morgan from 'morgan';
import helmet from 'helmet';
import errorMiddleware from './middlewares/errorMiddleware';
import *  as userController from './controller/user';
import "dotenv/config";
import multer from 'multer';
import path from 'path';
const storage = multer.diskStorage({
    destination: path.join(__dirname, 'public', 'uploads'),
    filename(_req: Request, file: Express.Multer.File, cb) {
        cb(null, Date.now() + path.extname(file.originalname));
    }
});
const upload = multer({ storage });
const app: Express = express();
app.use(morgan("dev"));
app.use(cors());
app.use(helmet());
app.use(express.static(path.resolve(__dirname, 'public')));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.get('/', (_req: Request, res: Response) => {
    res.json({ success: true, message: 'hello world' });
});
app.get('/user/validate', userController.validate);
app.post('/user/register', userController.register);
app.post('/user/login', userController.login);
app.post('/user/uploadAvatar', upload.single('avatar'), userController.uploadAvatar);
app.use((_req: Request, _res: Response, next: NextFunction) => {
    const error: HttpException = new HttpException(404, '尚未为此路径分配路由');
    next(error);
});
app.use(errorMiddleware);
const PORT: number = (process.env.PORT && parseInt(process.env.PORT)) || 8000;
(async function () {
    mongoose.set('useNewUrlParser', true);
    mongoose.set('useUnifiedTopology', true);
    await mongoose.connect('mongodb://localhost/zhufengketang');
    app.listen(PORT, () => {
        console.log(`Running on http://localhost:${PORT}`);
    });
})();
```

src/exceptions/HttpException.ts

```
class HttpException extends Error {
    constructor(public status: number, public message: string, public errors?: any) {
        super(message);
    }
}
export default HttpException;
```

src/middlewares/errorMiddleware.ts

```
import { Request, Response, NextFunction } from 'express';
import HttpException from '../exceptions/HttpException';
import { INTERNAL_SERVER_ERROR } from 'http-status-codes';
const errorMiddleware = (err: HttpException, _req: Request, res: Response, _next: NextFunction) => {
    let result: any = {
        success: false,
        message: err.message
    };
    if (err.errors && Object.keys(err.errors).length > 0) {
        result.errors = err.errors;
    }
    res.status(err.status || INTERNAL_SERVER_ERROR)
        .json(result);
}
export default errorMiddleware;
```

src/utils/validator.ts

```
import validator from 'validator';
import { UserDocument } from '../models/user';

export interface RegisterInput extends Partial {
    confirmPassword?: string;
}

export interface RegisterInputValidateResult {
    errors: RegisterInput,
    valid: boolean
}

export const validateRegisterInput = (
    username: string,
    password: string,
    confirmPassword: string,
    email: string
): RegisterInputValidateResult => {
    let errors: RegisterInput = {};
    if (username == undefined || validator.isEmpty(username)) {
        errors.username = '用户名不能为空';
    }
    if (password == undefined || validator.isEmpty(password)) {
        errors.password = '密码不能为空';
    }
    if (confirmPassword == undefined || validator.isEmpty(confirmPassword)) {
        errors.password = '确认密码不能为空';
    }
    if (!validator.equals(password, confirmPassword)) {
        errors.confirmPassword = '确认密码和密码不相等';
    }
    if (email == undefined || validator.isEmpty(password)) {
        errors.email = '邮箱不能为空';
    }
    if (validator.isEmail(password)) {
        errors.email = '邮箱格式必须合法';
    }
    return { errors, valid: Object.keys(errors).length == 0 };
}
```

src/typings/jwt.ts

```
import { UserDocument } from "../models/user";

export interface UserPayload {
    id: UserDocument['_id']
}
```

src\models\index.ts

```
export * from './user';
```

src/models/user.ts

```ts
import mongoose, { Schema, Model, Document, HookNextFunction } from 'mongoose';
import validator from 'validator';
import jwt from 'jsonwebtoken';
import { UserPayload } from '../typings/jwt';
import bcrypt from 'bcryptjs';
export interface UserDocument extends Document {
    username: string,
    password: string,
    email: string;
    avatar: string;
    generateToken: () => string,
    _doc: UserDocument
}
const UserSchema: Schema = new Schema({
    username: {
        type: String,
        required: [true, '用户名不能为空'],
        minlength: [6, '最小长度不能少于6位'],
        maxlength: [12, '最大长度不能大于12位']
    },
    password: String,
    avatar: String,
    email: {
        type: String,
        validate: {
            validator: validator.isEmail
        },
        trim: true,
    }
}, { timestamps: true ,    toJSON: {
        transform: function (_doc: any, result: any) {
            result.id = result._id;
            delete result._id;
            delete result.__v;
            delete result.password;
            delete result.createdAt;
            delete result.updatedAt;
            return result;
        }
    }});

UserSchema.methods.generateToken = function (): string {
    let payload: UserPayload = ({ id: this.id });
    return jwt.sign(payload, process.env.JWT_SECRET_KEY!, { expiresIn: '1h' });
}
UserSchema.pre('save', async function (next: HookNextFunction) {
    if (!this.isModified('password')) {
        return next();
    }
    try {
        this.password = await bcrypt.hash(this.password, 10);
        next();
    } catch (error) {
        next(error);
    }
});
UserSchema.static('login', async function (this: any, username: string, password: string): Promise<UserDocument | null> {
    let user: UserDocument | null = await this.model('User').findOne({ username });
    if (user) {
        const matched = await bcrypt.compare(password, user.password);
        if (matched) {
            return user;
        } else {
            return null;
        }
    }
    return user;
});
interface IUserModel extends Model {
    login: (username: string, password: string) => UserDocument | null
}
export const User: IUserModel = mongoose.model<IUserModel>('User', UserSchema);
```

src\controller\user.ts

```
import { Request, Response, NextFunction } from 'express';
import { validateRegisterInput } from '../utils/validator';
import HttpException from '../exceptions/HttpException';
import { UNPROCESSABLE_ENTITY, UNAUTHORIZED } from 'http-status-codes';
import { UserDocument, User } from '../models/user';
import { UserPayload } from '../typings/jwt';
import jwt from 'jsonwebtoken';

export const validate = async (req: Request, res: Response, next: NextFunction) => {
    const authorization = req.headers.authorization;
    if (authorization) {
        const access_token = authorization.split(' ')[1];
        if (access_token) {
            try {
                const userPayload: UserPayload = jwt.verify(access_token, process.env.JWT_SECRET_KEY || 'zhufeng') as UserPayload;
                const user: UserDocument | null = await User.findById(userPayload.id);
                if (user) {
                    res.json({
                        success: true,
                        data: user.toJSON()
                    })
                } else {
                    next(new HttpException(UNAUTHORIZED, '用户未找到'));
                }
            } catch (error) {
                next(new HttpException(UNAUTHORIZED, 'access_token不正确'));
            }
        } else {
            next(new HttpException(UNAUTHORIZED, 'access_token未提供'));
        }
    } else {
        next(new HttpException(UNAUTHORIZED, 'authorization未提供'));
    }
}
export const register = async (req: Request, res: Response, next: NextFunction) => {
    try {
        let { username, password, confirmPassword, email, addresses } = req.body;
        const { valid, errors } = validateRegisterInput(username, password, confirmPassword, email);
        if (!valid) {
            throw new HttpException(UNPROCESSABLE_ENTITY, `参数验证失败!`, errors);
        }
        let user: UserDocument = new User({
            username,
            email,
            password,
            addresses
        });
        let oldUser: UserDocument | null = await User.findOne({ username: user.username });
        if (oldUser) {
            throw new HttpException(UNPROCESSABLE_ENTITY, `用户名重复!`);
        }
        await user.save();
        let token = user.generateToken();
        res.json({
            success: true,
            data: { token }
        });
    } catch (error) {
        next(error);
    }
}

export const login = async (req: Request, res: Response, next: NextFunction) => {
    try {
        let { username, password } = req.body;
        let user = await User.login(username, password);
        if (user) {
            let token = user.generateToken();
            res.json({
                success: true,
                data: {
                    token
                }
            });
        } else {
            throw new HttpException(UNAUTHORIZED, `登录失败`);
        }
    } catch (error) {
        next(error);
    }
}
export const uploadAvatar = async (req: Request, res: Response, next: NextFunction) => {
    let { userId } = req.body;
    let avatar = `${req.protocol}://${req.headers.host}/uploads/${req.file.filename}`;
    await User.updateOne({ _id: userId }, { avatar });
    res.send({ success: true, data: avatar });
}
```

src\typings\express.d.ts

```
import { UserDocument } from "../models/user";
declare global {
    namespace Express {
        export interface Request {
            currentUser?: UserDocument | null;
            file: Multer.File
        }
    }
}
```

3.轮播图

src\index.ts

```
import express, { Express, Request, Response, NextFunction } from 'express';
import mongoose from 'mongoose';
import HttpException from './exceptions/HttpException';
import cors from 'cors';
import morgan from 'morgan';
import helmet from 'helmet';
import errorMiddleware from './middlewares/errorMiddleware';
import *  as userController from './controller/user';
+import *  as sliderController from './controller/slider';
import "dotenv/config";
import multer from 'multer';
import path from 'path';
+import { Slider } from './models';
const storage = multer.diskStorage({
    destination: path.join(__dirname, 'public', 'uploads'),
    filename(_req: Request, file: Express.Multer.File, cb) {
        cb(null, Date.now() + path.extname(file.originalname));
    }
});
const upload = multer({ storage });
const app: Express = express();
app.use(morgan("dev"));
app.use(cors());
app.use(helmet());
app.use(express.static(path.resolve(__dirname, 'public')));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.get('/', (_req: Request, res: Response) => {
    res.json({ success: true, message: 'hello world' });
});
app.get('/user/validate', userController.validate);
app.post('/user/register', userController.register);
app.post('/user/login', userController.login);
app.post('/user/uploadAvatar', upload.single('avatar'), userController.uploadAvatar);
+app.get('/slider/list', sliderController.list);
app.use((_req: Request, _res: Response, next: NextFunction) => {
    const error: HttpException = new HttpException(404, 'Route not found');
    next(error);
});
app.use(errorMiddleware);
const PORT: number = (process.env.PORT && parseInt(process.env.PORT)) || 8000;
(async function () {
    mongoose.set('useNewUrlParser', true);
    mongoose.set('useUnifiedTopology', true);
    await mongoose.connect(process.env.MONGODB_URL!);
+    await createSliders();
    app.listen(PORT, () => {
        console.log(`Running on http://localhost:${PORT}`);
    });
})();

+async function createSliders() {
+    const sliders = await Slider.find();
+    if (sliders.length == 0) {
+        const sliders = [
+            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/reactnative.png' },
+            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png' },
+            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png' },
+            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/wechat.png' },
+            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/architect.jpg' }
+        ];
+        Slider.create(sliders);
+    }
+}
```

src\controller\slider.ts

```
import { Request, Response } from 'express';
import { ISliderDocument, Slider } from '../models';
export const list = async (_req: Request, res: Response) => {
    let sliders: ISliderDocument[] = await Slider.find();
    res.json({ success: true, data: sliders });
}
```

src\models\slider.ts

```
import mongoose, { Schema, Document } from 'mongoose';
export interface ISliderDocument extends Document {
    url: string,
    _doc: ISliderDocument
}
const SliderSchema: Schema = new Schema({
    url: String,
}, { timestamps: true });

export const Slider = mongoose.model('Slider', SliderSchema);
```

src\models\index.ts

```
export * from './user';
+export * from './slider';
```

4.课程管理

src\index.ts

```
import express, { Express, Request, Response, NextFunction } from 'express';
import mongoose from 'mongoose';
import HttpException from './exceptions/HttpException';
import cors from 'cors';
import morgan from 'morgan';
import helmet from 'helmet';
import errorMiddleware from './middlewares/errorMiddleware';
import *  as userController from './controller/user';
import *  as sliderController from './controller/slider';
```

```
+import *  as lessonController from './controller/lesson';
import "dotenv/config";
import multer from 'multer';
import path from 'path';
+import { Slider, Lesson } from './models';
const storage = multer.diskStorage({
    destination: path.join(__dirname, 'public', 'uploads'),
    filename(_req: Request, file: Express.Multer.File, cb) {
        cb(null, Date.now() + path.extname(file.originalname));
    }
});
const upload = multer({ storage });
const app: Express = express();
app.use(morgan("dev"));
app.use(cors());
app.use(helmet());
app.use(express.static(path.resolve(__dirname, 'public')));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.get('/', (_req: Request, res: Response) => {
    res.json({ success: true, message: 'hello world' });
});
app.get('/user/validate', userController.validate);
app.post('/user/register', userController.register);
app.post('/user/login', userController.login);
app.post('/user/uploadAvatar', upload.single('avatar'), userController.uploadAvatar);
app.get('/slider/list', sliderController.list);
+app.get('/lesson/list', lessonController.list);
+app.get('/lesson/:id', lessonController.get);
app.use((_req: Request, _res: Response, next: NextFunction) => {
    const error: HttpException = new HttpException(404, 'Route not found');
    next(error);
});
app.use(errorMiddleware);
const PORT: number = (process.env.PORT && parseInt(process.env.PORT)) || 8000;
(async function () {
    mongoose.set('useNewUrlParser', true);
    mongoose.set('useUnifiedTopology', true);
    await mongoose.connect(process.env.MONGODB_URL!);
    await createSliders();
+    await createLessons();
    app.listen(PORT, () => {
        console.log(`Running on http://localhost:${PORT}`);
    });
})();

async function createSliders() {
    const sliders = await Slider.find();
    if (sliders.length == 0) {
        const sliders = [
            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/reactnative.png' },
            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png' },
            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png' },
            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/wechat.png' },
            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/architect.jpg' }
        ];
        Slider.create(sliders);
    }
}

+async function createLessons() {
+    const lessons = await Lesson.find();
+    if (lessons.length == 0) {
        const lessons = [
            {
                order: 1,
                title: '1.React全栈架构',
                video: "http://img.zhufengpeixun.cn/gee2.mp4",
                poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
                url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
                price: '¥100.00元',
                category: 'react'
            },
            {
                order: 2,
                title: '2.React全栈架构',
                video: "http://img.zhufengpeixun.cn/gee2.mp4",
                poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
                url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
                price: '¥200.00元',
                category: 'react'
            },
            {
                order: 3,
                title: '3.React全栈架构',
                video: "http://img.zhufengpeixun.cn/gee2.mp4",
                poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
                url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
                price: '¥300.00元',
                category: 'react'
            },
            {
                order: 4,
                title: '4.React全栈架构',
                video: "http://img.zhufengpeixun.cn/gee2.mp4",
                poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
                url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
                price: '¥400.00元',
                category: 'react'
            },
            {
                order: 5,
                title: '5.React全栈架构',
                video: "http://img.zhufengpeixun.cn/gee2.mp4",
```

```
    poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
    price: '¥500.00元',
    category: 'react'
},
{
    order: 6,
    title: '6.Vue从入门到项目实战',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
    price: '¥100.00元',
    category: 'vue'
},
{
    order: 7,
    title: '7.Vue从入门到项目实战',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
    price: '¥200.00元',
    category: 'vue'
},
{
    order: 8,
    title: '8.Vue从入门到项目实战',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
    price: '¥300.00元',
    category: 'vue'
},
{
    order: 9,
    title: '9.Vue从入门到项目实战',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
    price: '¥400.00元',
    category: 'vue'
},
{
    order: 10,
    title: '10.Vue从入门到项目实战',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
    price: '¥500.00元',
    category: 'vue'
},
{
    order: 11,
    title: '11.React全栈架构',
    "video": "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
    price: '¥600.00元',
    category: 'react'
},
{
    order: 12,
    title: '12.React全栈架构',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
    price: '¥700.00元',
    category: 'react'
},
{
    order: 13,
    title: '13.React全栈架构',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
    price: '¥800.00元',
    category: 'react'
},
{
    order: 14,
    title: '14.React全栈架构',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
    price: '¥900.00元',
    category: 'react'
},
{
    order: 15,
    title: '15.React全栈架构',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
    price: '¥1000.00元',
    category: 'react'
},
{
    order: 16,
    title: '16.Vue从入门到项目实战',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
    price: '¥600.00元',
    category: 'vue'
},
```

```
            {
                order: 17,
                title: '17.Vue从入门到项目实战',
                video: "http://img.zhufengpeixun.cn/gee2.mp4",
                poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
                url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
                price: '¥700.00元',
                category: 'vue'
            },
            {
                order: 18,
                title: '18.Vue从入门到项目实战',
                video: "http://img.zhufengpeixun.cn/gee2.mp4",
                poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
                url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
                price: '¥800.00元',
                category: 'vue'
            },
            {
                order: 19,
                title: '19.Vue从入门到项目实战',
                video: "http://img.zhufengpeixun.cn/gee2.mp4",
                poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
                url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
                price: '¥900.00元',
                category: 'vue'
            },
            {
                order: 20,
                title: '20.Vue从入门到项目实战',
                video: "http://img.zhufengpeixun.cn/gee2.mp4",
                poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
                url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
                price: '¥1000.00元',
                category: 'vue'
            }
        ];
+       Lesson.create(lessons);
    }
}
```

src\models\index.ts

```
export * from './user';
export * from './slider';
+export * from './lesson';
```

src\controller\lesson.ts

```
import { Request, Response } from 'express';
import { ILessonDocument, Lesson } from '../models';
export const list = async (req: Request, res: Response) => {
    let { offset, limit, category } = req.query;
    offset = isNaN(offset) ? 0 : parseInt(offset);
    limit = isNaN(limit) ? 5 : parseInt(limit);
    let query: Partial = {} as ILessonDocument;
    if (category && category != 'all')
        query.category = category;
    let total = await Lesson.count(query);
    let list = await Lesson.find(query).sort({ order: 1 }).skip(offset).limit(limit);
    setTimeout(function () {
        res.json({ code: 0, data: { list, hasMore: total > offset + limit } });
    }, 1000);
}
export const get = async (req: Request, res: Response) => {
    let id = req.params.id;
    let lesson = await Lesson.findById(id);
    res.json({ success: true, data: lesson });
}
```

src\models\lesson.ts

```
import mongoose, { Schema, Document } from 'mongoose';
export interface ILessonDocument extends Document {
    order: number,
    title: string,
    video: string,
    poster: string,
    url: string,
    price: string,
    category: string,
    _doc: ILessonDocument
}
const LessonSchema: Schema = new Schema({
    order: Number,
    title: String,
    video: String,
    poster: String,
    url: String,
    price: String,
    category: String,
}, { timestamps: true });

export const Lesson = mongoose.model('Lesson', LessonSchema);
```