link: null
title: 珠峰架构师成长计划
description: package.json
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=81 sentences=220, words=1275

1.搭建开发环境

```
mkdir zhufengexpress
cd zhufengexpress
cnpm init -y
npx tsconfig.json
```

1.1 安装依赖

```
cnpm i express sequelize mysql2 morgan http-errors http-status-codes  -S
cnpm i cross-env typescript @types/express @types/morgan @types/http-errors ts-node-dev nodemon ts-node nyc mocha @types/mocha chai @types/chai chai-http  -D
```

1.2 package.json

package.json

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

1.3 bin\www.ts

src\binwww.ts

```
import app from '../app';
import http from 'http';

const port = process.env.PORT || 8000;

const server = http.createServer(app);

server.listen(port);
server.on('error', onError);
server.on('listening', onListening);
function onError(error: any) {
  console.error(error);
}
function onListening() {
  console.log('Listening on ' + port);
}
```

1.4 app.ts

src\app.ts

```
import createError from 'http-errors';
import express, { Request, Response } from 'express';
import logger from 'morgan';
var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');
var app = express();
app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));

app.use('/', indexRouter);
app.use('/users', usersRouter);

app.use(function (_req, _res, next) {
    next(createError(404));
});

app.use(function (error: any, _req: Request, res: Response, _next: NextFunction) {
    res.status(error.status || INTERNAL_SERVER_ERROR);
    res.json({
        success: false,
        error
    });
});

export default app;
```

1.5 routes\index.ts

src\routes\index.ts

```
import express, { Request, Response } from 'express';
var router = express.Router();

router.get('/', function (_req: Request, res: Response) {
    res.json({
        success: true,
        data: 'home'
    })
});

module.exports = router;
```

1.6 routes\users.ts

src\routes\users.ts

```
import express, { Request, Response } from 'express';
var router = express.Router();

router.get('/', function (_req: Request, res: Response) {
    res.json({
        success: true,
        data: 'users'
    })
});

module.exports = router;
```

2. 使用sequelize

- Sequelize 是一个基于 promise 的 Node.js ORM, 目前支持 Postgresql, MySQL, SQLite 和 Microsoft SQL Server. 它具有强大的事务支持, 关联关系, 预读和延迟加载,读取复制等功能.

2.1 安装

```
cnpm i sequelize -S
```

2.2 model\sequelize.ts

src\model\sequelize.ts

```
import { Sequelize } from 'sequelize';
const sequelize = new Sequelize('restful', 'root', 'root', {
    host: 'localhost',
    dialect: 'mysql',
    logging: false
})
export {
    sequelize
};
```

2.3 src\model\user.ts

src\model\user.ts

```
import { Model, DataTypes } from 'sequelize';
import { sequelize } from './';
class User extends Model { }

User.init({
    username: DataTypes.STRING,
    password: DataTypes.STRING,
    email: DataTypes.STRING
}, { sequelize, modelName: 'user' });

export {
    User
};
```

2.4 model\index.ts

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

3. restful

- REST就是用 `URI`表示资源，用HTTP方法(GET, POST, PUT, DELETE)表示对这些资源做什么操作

方法 路径 名称 GET /users 查看用户列表 GET /users/:id 查看单个用户 POST /users 添加用户 PUT /users/:id 修改单个用户 DELETE /users/:id 删除单个用户

3.1 routes\users.ts

```
import express, { Request, Response, NextFunction } from 'express';
import createError from 'http-errors';
import { User } from '../model';
import { INTERNAL_SERVER_ERROR } from 'http-status-codes';
var router = express.Router();

router.get('/', async function (_req: Request, res: Response, next: NextFunction) {
    try {
        let users = await User.findAll();
        res.json({
            success: true,
            data: users
        })
    } catch (error) {
        next(createError(INTERNAL_SERVER_ERROR));
    }
});
router.get('/:id', async function (req: Request, res: Response, next: NextFunction) {
    try {
        let user = await User.findByPk(req.params.id);
        if (!user) {
            return next(createError(INTERNAL_SERVER_ERROR));
        }
        res.json({
            success: true,
            data: user
        })
    } catch (error) {
        next(createError(INTERNAL_SERVER_ERROR));
    }

});
router.post('/', async function (req: Request, res: Response, next: NextFunction) {
    try {
        let user = req.body;
        user = await User.create(user);
        res.json({
            success: true,
            data: user
        })
    } catch (error) {
        next(createError(INTERNAL_SERVER_ERROR));
    }

});
router.put('/:id', async function (req: Request, res: Response, next: NextFunction) {
    try {
        let id = req.params.id;
        let update = req.body;
        let user = await User.findByPk(id);
        if (!user) {
            return next(createError(INTERNAL_SERVER_ERROR));
        }
        user = await user.update(update);
        res.json({
            success: true,
            data: user
        })
    } catch (error) {
        next(createError(INTERNAL_SERVER_ERROR));
    }

});
router.delete('/:id', async function (req: Request, res: Response, next: NextFunction) {
    try {
        let id = req.params.id;
        let user = await User.findByPk(id);
        if (!user) {
            return next(createError(INTERNAL_SERVER_ERROR));
        }
        user = await user.destroy();
        res.json({
            success: true,
            data: user
        })
    } catch (error) {
        next(createError(INTERNAL_SERVER_ERROR));
    }
});

module.exports = router;
```

4. 单元测试

4.1 安装

- mochajs (https://mochajs.org/)
- chaijs (https://www.chaijs.com/)
- chai-http (https://www.chaijs.com/plugins/chai-http/)
- growl (https://www.npmjs.com/package/growl)

```
cnpm i mocha @types/mocha chai @types/chai chai-http -D
```

4.2 src\tests\helper.spec.ts

```
import chai from 'chai';
import chaiHttp from 'chai-http';
import { sequelize, User } from '../model';
chai.use(chaiHttp);

before(async () => {
    await sequelize.sync();
});

beforeEach(async () => {
    await User.truncate();
});
afterEach(async () => {
    await User.truncate();
});
```

4.3 tests\index.spec.ts

```
import app from '../app';
import chai, { expect } from 'chai';
describe('index', () => {
    it('GET / 访问首页', (done) => {
        chai
            .request(app)
            .get('/')
            .end(function (err, res) {
                expect(err).to.be.null;
                expect(res).to.have.status(200);
                expect(res.body.success).to.equal(true);
                expect(res.body).to.have.property('data');
                done();
            });
    });
});
```

4.4 src\tests\user.spec.ts

src\tests\user.spec.ts

```
import app from '../app';
import chai, { expect } from 'chai';
describe('users', () => {
    it('POST /users 添加用户', async () => {
        let result = await chai
            .request(app)
            .post('/users')
            .set('Content-Type', 'application/json')
            .send({ username: 'zhangsan', password: '123456' })
        expect(result).to.have.status(200);
        expect(result.body.success).to.equal(true);
        expect(result.body).to.have.property('data');
    });

    it('GET /users 查看用户列表', async () => {
        await chai
            .request(app)
            .post('/users')
            .set('Content-Type', 'application/json')
            .send({ username: 'zhangsan', password: '123456' })

        let result = await chai.request(app).get('/users');
        expect(result.body.data).to.have.lengthOf(1);
    });

    it('PUT /users/1 更新', async () => {
        let result: any = await chai
            .request(app)
            .post('/users')
            .set('Content-Type', 'application/json')
            .send({ username: 'zhangsan', password: '123456' })

        let update = await chai
            .request(app)
            .put(`/users/${result.body.data.id}`)
            .set('Content-Type', 'application/json')
            .send({ password: '111111' })
        expect(update.body.data.password).to.equal('111111');
    });
    it('PUT /users/1 删除用户', async () => {
        let addResult: any = await chai
            .request(app)
            .post('/users')
            .set('Content-Type', 'application/json')
            .send({ username: 'zhangsan', password: '123456' });
        let findRequest = await chai.request(app).get('/users');
        expect(findRequest.body.data).to.have.lengthOf(1);
        await chai
            .request(app)
            .delete(`/users/${addResult.body.data.id}`);
        let findRequest2 = await chai.request(app).get('/users');
        expect(findRequest2.body.data).to.have.lengthOf(0);
    });
});
```

5. docker布署

5.1. 准备工作

- 建议从阿里云 (https://dc.console.aliyun.com/next/index)购买域名
- 建议从阿里云 (https://ecs.console.aliyun.com)购买ECS服务器
- 建议从阿里云 (https://bsn.console.aliyun.com)进行备案

5.2. 安装系统

- 选择最新的 CentOS 7.6

5.3. 安装docker

```
yum install -y yum-utils   device-mapper-persistent-data   lvm2
yum-config-manager \
    --add-repo \
    https:
yum install -y docker-ce docker-ce-cli containerd.io
```

5.4 阿里云加速

```
mkdir -p /etc/docker
tee /etc/docker/daemon.json <
```

5.5 安装git

```
yum install git -y
```

5.6 安装node

- [nodejs (https://nodejs.org/en/download/)](https://nodejs.org/en/download/)

5.6.1 nvm

- [nvm (https://github.com/nvm-sh/nvm)](https://github.com/nvm-sh/nvm)

```
wget -qO- https:
source /root/.bashrc
nvm ls-remote
nvm install v12.16.0
```

5.6.2 源码安装

```

wget http:
tar -xvf node-v11.0.0.tar.gz
cd node-v11.0.0
yum install gcc gcc-c++ -y
./configure
make
make install
node -v
```

5.6.3 xz

```
wget http://img.zhufengpeixun.cn/node-v12.16.0-linux-x64.tar.xz
xz -d node-v12.16.0-linux-x64.tar.xz
tar -xf node-v12.16.0-linux-x64.tar
ln -s ~/node-v12.16.0-linux-x64/bin/node /usr/bin/node
ln -s ~/node-v12.16.0-linux-x64/bin/npm /usr/bin/npm
ln -s ~/node-v12.16.0-linux-x64/bin/npm /usr/bin/npx
```

5.7 Dockerfile

```
FROM node
COPY . /api
WORKDIR /api
RUN npm install
EXPOSE 8000
CMD npm start
```

5.8 .dockerignore

```
.git
node_modules
package-lock.json
Dockerfile
.dockerignore
```

5.8 启动

```
docker build -t rest .
docker container run -p 8000:8000 -d  rest
```

6. docker-compose

6.1 安装docker-compose

```
curl -L https:
chmod +x /usr/local/bin/docker-compose
```

6.2 docker-compose.yml

```
version: '2'
services:
 node:
  build:
    context: ./ts_express
    dockerfile: Dockerfile
  ports:
    - "8000:8000"
  depends_on:
   - db
 db:
  image: mariadb
  environment:
   MYSQL_ROOT_PASSWORD: "root"
   MYSQL_DATABASE: "restful"
  volumes:
    - db:/var/lib/mysql
volumes:
 db:
  driver: local
```

6.2 启动docker-compose

```
docker-compose build
docker-compose up
docker-compose up -d
```