link: null
title: 珠峰架构师成长计划
description: config/config.default.js
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=122 sentences=547, words=2590

1.初始化项目

```
$ npm i egg-init -g
$ egg-init cms-api --type=simple
$ cd cms-api
$ npm i
$ npm run dev
```

2.使用MYSQL

```
CREATE TABLE user (
 id int(11) PRIMARY KEY AUTO_INCREMENT,
 username varchar(255) NULL,
 password varchar(255) NULL,
 email varchar(255) NULL,
 phone varchar(255) NULL,
 gender tinyint(255) NULL,
 birthday datetime NULL,
 address varchar(255) NULL
);

CREATE TABLE role (
 id int(11) PRIMARY KEY AUTO_INCREMENT,
 name varchar(255) NULL
);

CREATE TABLE role_user (
role_id int(11) NOT NULL,
user_id int(11) NOT NULL,
PRIMARY KEY (user_id, role_id)
);

CREATE TABLE resource (
id int(11) PRIMARY KEY AUTO_INCREMENT,
name varchar(255) NOT NULL
);

CREATE TABLE role_resource (
role_id int(11) NOT NULL,
resource_id int(255) NOT NULL,
PRIMARY KEY (role_id, resource_id)
);
```

```
cnpm i egg-mysql -S
```

config/config.default.js

```
config.security = {
    csrf:false
}
config.mysql = {

    client: {

      host: 'localhost',

      port: '3306',

      user: 'root',

      password: 'root',

      database: 'cms',
}
```

config/plugin.js

```
exports.mysql = {
    enable: true,
    package: 'egg-mysql'
};
```

3. 用户管理

```
module.exports = app => {
  const { router, controller } = app;
  router.get('/',controller.home.index);
  router.resources('user', '/api/user', controller.user);
};
```

app/controller/user.js

```
const {Controller} = require('egg');
class UserController extends Controller {
  async index() {
    let {ctx,service}=this;
    let users=await service.user.select();
    ctx.body = users;
  }

  async create() {
    let {ctx,service}=this;
    let user=ctx.request.body;
    await service.user.create(user);
    ctx.body={
      code: 0,
      data:'success!'
    }
  }

  async update() {
    let {ctx,service}=this;
    let user=ctx.request.body;
    user.id=ctx.params.id;
    let savedUser=await service.user.update(user);
    ctx.body={
      code: 0,
      data:'success!'
    }
  }
  async destroy() {
    let {ctx,service}=this;
    let id=ctx.params.id;
    await service.user.delete(id);
    ctx.body={
      code: 0,
      data:'success!'
    }
  }
}

module.exports = UserController;
```

app/service/user.js

```
const {Service}=require('egg');
class UserService extends Service{
    async select() {
        return await this.app.mysql.select('user');
    }
    async create(entity) {
        return await this.app.mysql.insert('user',entity);
    }
    async update(entity) {
        return await this.app.mysql.update('user',entity);
    }
    async delete(id) {
        return await this.app.mysql.delete('user',{id});
    }
}
module.exports=UserService;
```

4. 提取基类

app/controller/api.js

```
const BaseController = require('./base');
class ApiController extends BaseController {
    async index() {
    let {ctx,service}=this;
    let list=await service[this.entity].select();
    ctx.body = list;
  }

  async create() {
    let {ctx,service}=this;
    let user=ctx.request.body;
    await service[this.entity].create(user);
    ctx.body={
      code: 0,
      data:'success!'
    }
  }

  async update() {
    let {ctx,service}=this;
    let user=ctx.request.body;
    user.id=ctx.params.id;
    await service[this.entity].update(user);
    ctx.body={
      code: 0,
      data:'success!'
    }
  }
  async destroy() {
    let {ctx,service}=this;
    let id=ctx.params.id;
    await service[this.entity].delete(id);
    ctx.body={
      code: 0,
      data:'success!'
    }
  }
}
module.exports=ApiController;
```

app/controller/base.js

```
const Controller = require('egg').Controller;
class BaseController extends Controller {
    success(data) {
        let {ctx}=this;
        ctx.body={
            code: 0,
            data
        }
    }
    error(error) {
        let {ctx}=this;
        ctx.status=404;
        ctx.body={
            code: 1,
            error
        }
    }
}
module.exports=BaseController;
```

app/service/base.js

```
const {Service}=require('egg');
class BaseService extends Service{
    async select() {
        return await this.app.mysql.select(this.entity);
    }
    async create(entity) {
        return await this.app.mysql.insert(this.entity,entity);
    }
    async update(entity) {
        return await this.app.mysql.update(this.entity,entity);
    }
    async delete(id) {
        return await this.app.mysql.delete(this.entity,{id});
    }
}
module.exports=BaseService;
```

app/controller/user.js

```
const ApiController = require('./api');
class UserController extends ApiController {
  constructor(...args) {
    super(...args);
    this.entity='user';
  }
}

module.exports = UserController;
```

app/service/user.js

```
const BaseService = require('./base');
class UserService extends BaseService{
    constructor(...args) {
        super(...args);
        this.entity='user';
    }
}
module.exports=UserService;
```

5. 角色

app/router.js

```
+ router.resources('role', '/api/role', controller.role);
```

app/controller/role.js

```
const ApiController = require('./api');
class RoleController extends ApiController {
  constructor(...args) {
    super(...args);
    this.entity='role';
  }
}

module.exports = RoleController;
```

app/service/role.js

```
const BaseService = require('./base');
class RoleService extends BaseService{
    constructor(...args) {
        super(...args);
        this.entity='role';
    }
}
module.exports=RoleService;
```

6. 其它功能

- 实现分页功能
- 根据数据库操作返回值来判断操作是成功还是失败

```
const BaseController = require('./base');
class ApiController extends BaseController {
    async index() {
    const {ctx,service}=this;
    const {pageNum,pageSize,...where}=ctx.query;
    let result=await service[this.entity].list(isNaN(pageNum)?1:parseInt(pageNum),isNaN(pageSize)?this.config.PAGE_SIZE:parseInt(pageSize),where);
    this.success(result);
  }

  async create() {
    let {ctx,service}=this;
    let user=ctx.request.body;
    let result = await service[this.entity].create(user);
    result>0? this.success('添加成功'):this.error('添加失败');
  }

  async update() {
    let {ctx,service}=this;
    let user=ctx.request.body;
    user.id=ctx.params.id;
    let result = await service[this.entity].update(user);
    result>0? this.success('更新成功'):this.error('更新失败');
  }
  async destroy() {
    let {ctx,service}=this;
    let id=ctx.params.id;
    let ids=ctx.request.body;
    if (!ids) {ids=[id]}
    let result = await service[this.entity].delete(ids);
    result>0? this.success('删除成功'):this.error('删除失败');
  }
}
module.exports=ApiController;
```

app/controller/base.js

```
const Controller = require('egg').Controller;
class BaseController extends Controller {
    success(data) {
        let {ctx}=this;
        ctx.body={
            code: 0,
            data
        }
    }
    error(error) {
        let {ctx}=this;
        ctx.status=500;
        ctx.body={
            code: 1,
            error
        }
    }
}
module.exports=BaseController;
```

app/router.js

```
module.exports = app => {
  const { router, controller } = app;
  router.get('/',controller.home.index);
  router.resources('user','/api/user',controller.user);
  router.resources('role','/api/role',controller.role);
  router.resources('resource', '/api/resource', controller.resource);
  router.resources('roleUser', '/api/roleUser', controller.roleUser);
  router.resources('roleResource', '/api/roleResource', controller.roleResource);
};
```

app/service/base.js

```
const {Service}=require('egg');
class BaseService extends Service{
    async list(pageNum,pageSize,where) {
        const {app} = this;
        const list=await app.mysql.select(this.entity,{
            where,
            order: [['id','desc']],
            offset: (pageNum-1)*pageSize,
            limit :pageSize
        });
        const total=await app.mysql.count(this.entity,where);

        return {list,total};
    }
    async create(entity) {
        const {app}=this;
        let result=await app.mysql.insert(this.entity,entity);
        const affectedRows=result.affectedRows;
        return affectedRows;
    }
    async update(entity) {
        const {app}=this;
        let result = await app.mysql.update(this.entity,entity);
        const affectedRows=result.affectedRows;
        return affectedRows;
    }
    async delete(ids) {
        const {app}=this;
        let result = await app.mysql.delete(this.entity,{id:ids});
        const affectedRows=result.affectedRows;
        return affectedRows;
    }
}
module.exports=BaseService;
```

app/controller/resource.js

```
const ApiController = require('./api');
class ResourceController extends ApiController {
  constructor(...args) {
    super(...args);
    this.entity='resource';
  }
}
module.exports = ResourceController;
```

app/controller/roleResource.js

```
const ApiController = require('./api');
class RoleResourceController extends ApiController {
  constructor(...args) {
    super(...args);
    this.entity='roleResource';
  }
}
module.exports = RoleResourceController;
```

app/controller/roleUser.js

```
const ApiController = require('./api');
class RoleUserController extends ApiController {
  constructor(...args) {
    super(...args);
    this.entity='roleUser';
  }
}

module.exports = RoleUserController;
```

service/resource.js

```
const BaseService = require('./base');
class ResourceService extends BaseService{
    constructor(...args) {
        super(...args);
        this.entity='resource';
    }
}
module.exports=ResourceService;
```

app/service/roleResource.js

```
const BaseService = require('./base');
class roleResourceService extends BaseService{
    constructor(...args) {
        super(...args);
        this.entity='role_resource';
    }
}
module.exports=roleResourceService;
```

app/service/roleUser.js

```
const BaseService = require('./base');
class roleUserService extends BaseService{
    constructor(...args) {
        super(...args);
        this.entity='role_user';
    }
}
module.exports=roleUserService;
```

7. 权限管理

app/controller/role.js

```
const ApiController = require('./api');
class RoleController extends ApiController {
  constructor(...args) {
    super(...args);
    this.entity='role';
  }

  async getResource() {
    const { app, ctx, service } = this;
    const result = await service[this.entity].getResource();
    ctx.body = result;
  }
  async setResource() {
    const { app, ctx, service } = this;
    let body = ctx.request.body;
    const result = await service[this.entity].setResource(body);
    ctx.body = result;
  }
  async getUser() {
    const { app, ctx, service } = this;
    const result = await service[this.entity].getUser();
    ctx.body = result;
  }
  async setUser() {
    const { app, ctx, service } = this;
    let body = ctx.request.body;
    const result = await service[this.entity].setUser(body);
    ctx.body = result;
  }
}

module.exports = RoleController;
```

app/service/role.js

```
const BaseService = require('./base');
class RoleService extends BaseService{
    constructor(...args) {
        super(...args);
        this.entity='role';
    }
    async list(pageNum, pageSize, where) {
        const { app } = this;
        const list = await app.mysql.select(this.entity, {
          where,
          orders: [['id', 'desc']],
          offset: (pageNum - 1) * pageSize,
          limit: pageSize,
        });

        for (let i = 0; i < list.length; i++) {
          let rows = await app.mysql.select('role_resource', {
            where: { role_id: list[i].id }
          });
          list[i].resourceIds = rows.map(item => item.resource_id);

          rows = await app.mysql.select('role_user', {
            where: { role_id: list[i].id }
          });
          list[i].userIds = rows.map(item => item.user_id);
        }

        const total = await app.mysql.count(this.entity, where);
        return { list, total };
    }
    async getResource() {
        const { app } = this;
        const list = await app.mysql.select('resource');
        let rootMenus = [];
        let map = {};
        list.forEach(item => {
          item.children = [];
          map[item.id] = item;
          if (item.parent_id == 0) {
              rootMenus.push(item);
          } else {
               map[item.parent_id].children.push(item)
          }
        });
        return rootMenus;
      }
    async setResource(values) {
      const {app} = this;
      let {roleId,resourceIds} = values;
      const conn = await app.mysql.beginTransaction();
      try {

          await conn.query(`DELETE FROM role_resource WHERE role_id = ?`,[roleId]);

          for(let i=0;ilet resourceId = resourceIds[i];
            await conn.insert('role_resource',{role_id:roleId,resource_id:resourceId});
          }
        await conn.commit();
      } catch (err) {

        await conn.rollback();
        throw err;
      }
      return '修改权限成功!'
  }
    async getUser() {
        const { app } = this;
        const list = await app.mysql.select('user');
        return list;
      }
    async setUser(values){
      const {app} = this;
      let {roleId,userIds} = values;
      const conn = await app.mysql.beginTransaction();
      try {

          await conn.query(`DELETE FROM role_user WHERE role_id=?`,[roleId]);

          for(let i=0;ilet userId = userIds[i];
            await conn.insert('role_user',{role_id:roleId,user_id:userId});
          }
        await conn.commit();
      } catch (err) {

        await conn.rollback();
        throw err;
      }

      return '给角色分配用户成功!'
    }

}
module.exports=RoleService;
```

app/router.js

```
router.post('/role/setUser', controller.role.setUser);
router.get('/role/getUser', controller.role.getUser);
router.post('/role/setResource', controller.role.setResource);
router.get('/role/getResource', controller.role.getResource);
```

8. 验证码

- svg-captcha (https://www.npmjs.com/package/svg-captcha)**8.1 app/router.js**app/router.js

```
router.get('/captcha',controller.index.captcha);
```

8.2 controller/index.jsapp/controller/index.js

```
const BaseController=require('./base');
const svgCaptcha = require('svg-captcha');
class IndexController extends BaseController {
  async captcha() {
  let {ctx}=this;
  var captcha=svgCaptcha.create({});
  ctx.session.captcha=captcha.text;
  ctx.set('Content-Type','image/svg+xml');
  ctx.body=captcha.data;
}
}
module.exports=IndexController;
```

9. 跨域

- 跨域传 cookie的时候要求主域要一致，不能从 localhost跨到 127.0.0.1
- egg-cors (https://www.npmjs.com/package/egg-cors)

config/config.default.js

```
config.security = {
    csrf: false,
    domainWhiteList: [ 'http://127.0.0.1:8000' ]
}
```

config/plugin.js

```
exports.cors = {
    enable: true,
    package: 'egg-cors',
};
```

** 10. 注册登录 **

app/controller/user.js

```
const ApiController = require('./api');
const {sign} = require('jsonwebtoken');
class UserController extends ApiController {
  constructor(...args) {
    super(...args);
    this.entity = 'user';
  }

   async signin() {
    let {ctx,app} = this;
    let body = ctx.request.body;
    const result = await app.mysql.select('user', {where: {username: body.username,password: body.password},
      limit: 1,
      offset: 0
    });
    if (result && result.length > 0) {
      let user = JSON.parse(JSON.stringify(result[0]));
      let list = await app.mysql.query(`SELECT resource.* FROM role_user,role_resource,resource where role_user.role_id = role_resource.role_id AND
role_resource.resource_id = resource.id AND role_user.user_id = ? ORDER BY resource.id ASC`,[user.id]);
      let resources = [];
      let map = {};
      list.forEach(item => {
          item.children = [];
          map[item.id] = item;
          if (item.parent_id == 0) {
            resources.push(item);
          } else {
            map[item.parent_id].children.push(item);
          }
      });
      user.resources=resources;
      this.success(sign(user, this.config.jwtSecret));
    } else {
      this.error('登录失败');
    }
  }
  async signup() {
    let {ctx,app} = this;
    const { agreement, prefix, phone, address, repassword, captcha, ...user } = ctx.request.body;
    if (!agreement) {
      return this.error('请同意协议再注册!');
    }
    if (user.password !== repassword) {
      return this.error('密码和确认密码不一致!');
    }
    if (!captcha || !ctx.session.captcha || captcha.toLowerCase() !== ctx.session.captcha.toLowerCase()) {

    }
    user.phone = prefix + '-' + phone;
    user.address = address.join('-');
    const result = await app.mysql.insert('user', user);
    if (result.affectedRows > 0) {
      this.success({
        id: result.insertId,
      });
    } else {
      this.error('注册失败');
    }
  }
}

module.exports = UserController;
```

config/config.default.js

```
config.security = {
    csrf: false,
    domainWhiteList: [ 'http://localhost:8000' ]
}
config.jwtSecret="zfpx";
config.cors = {
    credentials: true
}
```

app/middleware/auth.js

```
let {verify}=require('jsonwebtoken');
function verifyToken(token,jwtSecret) {
    return new Promise(function (resolve,reject) {
        verify(token,jwtSecret,async (err,data) => {
            if (err) {
                reject(err);
            } else {
                resolve(data);
            }
        });
    });
}
module.exports=(options,app) => {
    return async function (ctx,next) {
        const token=ctx.get('authorization');
        if (token) {
            try {
                let user = await verifyToken(token,app.config.jwtSecret);
                ctx.session.user=user;
                await next();
            } catch (err) {
                ctx.status=401;
                ctx.body={
                    code: 1,
                    error:'token验证失败'
                }
            }
        } else {
            ctx.status=401;
                ctx.body={
                    code: 1,
                    error:'请提供token'
                }
        }
    }
}
```

app/router.js

```
  const auth=app.middleware.auth({},app);
  router.post('/api/signin',controller.user.signin);
  router.post('/api/signup',controller.user.signup);
  router.get('/captcha',controller.index.captcha);
  router.get('/',controller.home.index);
  router.post('/role/setUser', controller.role.setUser);
  router.get('/role/getUser', controller.role.getUser);
  router.post('/role/setResource', controller.role.setResource);
  router.get('/role/getResource', controller.role.getResource);
  router.resources('user','/api/user',auth,controller.user);
  router.resources('role','/api/role',auth,controller.role);
  router.resources('resource', '/api/resource', auth,controller.resource);
  router.resources('roleUser', '/api/roleUser', auth,controller.roleUser);
  router.resources('roleResource', '/api/roleResource', auth,controller.roleResource);
```

** 11.使用 VSCode 进行调试 **

- 使用-egg-bin-调试 (https://eggjs.org/zh-cn/core/development.html#%E4%BD%BF%E7%94%A8-egg-bin-%E8%B0%83%E8%AF%95)
- 方式一：开启 VSCode 配置 Debug: Toggle Auto Attach，然后在 Terminal 执行 npm run debug 即可。
- 方式二：配置 VSCode 的 .vscode/launch.json，然后 F5 一键启动即可。（注意，需要关闭方式一中的配置）

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Launch Egg",
      "type": "node",
      "request": "launch",
      "cwd": "${workspaceRoot}",
      "runtimeExecutable": "npm",
      "windows": { "runtimeExecutable": "npm.cmd" },
      "runtimeArgs": [ "run", "debug" ],
      "console": "integratedTerminal",
      "protocol": "auto",
      "restart": true,
      "port": 9229,
      "autoAttachChildProcesses": true
    }
  ]
}
```

** 12. 参考 **