eggjs是基于koa的nodejs框架，由alibaba集团开发，免费使用。

新建项目

npm init egg –type=simple

npm i

# 启动项目

npm run dev

目录结构

egg-project

|── package.json

|── app.js (可选)

|── agent.js (可选)

|── app

| |── router.js(路由配置)

| |── controller(controller层，app接口)

| | |——....js

| |── service (service层，供controller层调用，处理数据)

| | |——....js

| |── middleware (可选)(中间件，先于controller调用，处理数据)

| | |——....js

| |── schedule (可选)

| | |——my\_task.js

| |── public (可选，静态资源)

| | |——reset.css

| |── view (可选)

| | |——home.tpl

| |——extend (可选)

| |── helper.js (可选)

| |── request.js (可选)

| |── response.js (可选)

| |── context.js (可选)

| |── application.js (可选)

| |——agent.js (可选)

|── config

| |── plugin.js

| |── config.default.js

| |── config.prod.js

| |── config.test.js (可选)

| |── config.local.js (可选)

| |—— config.unittest.js (可选)

|———— test

|── middleware

| |—— response\_time.test.js

|———controller

|—— home.test.js

http访问->router->controller->service

## middleware中间件

创建中间件：项目的src目录下创建文件夹middleware用于存放中间件。

module.exports = options => {

return async function middlewarreone(ctx, next) {

//...前逻辑

await next();

//...返回后逻辑

}

};

全局使用中间件

//全局中间件配置

exports.middlewarreone = {

enable:true,//是否启用中间件，默认true

match:['/'], //中间件起作用的路径，默认全路径

ignore:[], //中间件不起作用的路径，ignore配置和match配置不能一起配置

//以下是组件内部参数配置【提醒：enable,match,ignore关键字不能作为组件内部参数】

name:'tomacat'

};

config.middleware = ['middlewarreone'];//全局中间件必须使用字符串格式的

router使用中间件

全局中间件最先执行，最后结束，所以router用的中间件如果在全局中使用的话，可能导致返回结果被覆盖。

'use strict';

/\*\*

\* @param {Egg.Application} app - egg application

\*/

module.exports = app => {

const { router, controller } = app;

router.get('/', controller.home.index);

const **middlewarreone** = **app.middleware.middlewarreone({name:'jack'})**;//配置中间件

router.get('/hello',**middlewarreone**,controller.hello.index);//引入使用

};

## router路由

在项目跟路径下的router文件夹里新建router.js

API

router.~('path-match', app.controller.action);

router.~('router-name', 'path-match', app.controller.action);

router.~('path-match', middleware1, ..., middlewareN, app.controller.action);

router.~('router-name','path-match',middleware1,...,middlewareN, app.controller.action);

~:请求方式，如get,post,delete,put,match

router-name:路由别名

path-match:映射地址

app.controller.action:controller类路径

middleware1:中间件

例

'use strict';

module.exports = app => {

const { router, controller } = app;

router.get('/', controller.home.index);

router.get('/hello',controller.hello.index); //get请求

//get请求，通过url传参，this.ctx.params.name获取参数

router.get('/hello/:id',controller.hello.index);

router.post('/hello2',controller.hello.index);//post请求

router.put('/hello3',controller.hello.index); //put请求

router.delete('/hello4',controller.hello.index);//delete请求

router.del('/hello5',controller.hello.index);//delete请求（别名）

router.redirect('/','/persons');//路由转发，内部重定向

};

router支持get，post，put，delete,head请求方式

### 太多router？模块化！

src下新建router文件夹

src/router.js内容如下

module.exports = app => {

require('./router/news')(app);

require('./router/admin')(app);

};

src/router/news.js内容如下

module.exports = app => {

app.router.get('/news/list', app.controller.news.list);

app.router.get('/news/detail', app.controller.news.detail);

};

src/router/admin.js内容如下

module.exports = app => {

app.router.get('/admin/user', app.controller.admin.user);

app.router.get('/admin/log', app.controller.admin.log);

};

### egg-router-plus使用

1. npm安装插件：npm i egg-router-plus --save
2. src/config/plugin.js中添加

routerPlus:{

enable: true,

package: 'egg-router-plus',

};

## **controller接口层**

const Controller = require('egg').Controller;

class Hello extends Controller {

async index(){

let {service} = this.ctx;

…

this.ctx.body = …;

}

}

module.exports = Hello;

## **service服务层**

const Service = require('egg').Service

class helloService extends Service {

async hello(){

return `hello service`;

}

}

module.exports = helloService;