Egg.js基础

成钞公司印钞管理部 李宾

2018-01-09


node
JS

下载

下载 8.9.4 LTS　文档

中文版文档

> 下载完毕后安装即可使用。

```dos
node -v
V8.7.0
```

NPM

> node.js中自带的包管理工具。

官网　文档

```dos
cd yourAppDir
e:\yourAppDir > npm install --save echarts
e:\yourAppDir > npm install --save-dev webpack
e:\yourAppDir > npm uninstall --save echarts
c:\ > npm install -g vue-cli
```

cnpm

由于一些众所周知的原因，npm在国内安装软件包的时候非常慢，因而淘宝启动了一个叫cnpm的镜象，墙内也能很方便地使用了。其用法与cnpm一样，支持 npm 除了 publish 之外的所有命令。

```dos
c:\> npm install -g cnpm --registry=https://registry.npm.taobao.org
```

开发前的准备

点击下载 Visual Studio Code

nodejs 官网例子

建立文件app.js，写一个简单的http服务端:

```js
// app.js
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

启动服务

```dos

c:\> node app.js
```

Egg.js

阿里在2016年南京JsConf 大会上正式开源的NodeJS框架，

当时的分享PPT在此 。在介绍Egg.js之前有必要简单了解一下Express和KOA

Express

express 是目前node.js上最流行的web应用框架，由于在node.js 7.5以后正式支持async异步调用，所以目前有部分开发者已经转移到KOA等框架。

```js
var express = require('express');
var app = express();

// 这里的function称为回调函数，callback function。
app.get('/', function (req, res) {
  res.send('Hello World!');
});

var server = app.listen(3000, function () {
  var host = server.address().address;
  var port = server.address().port;

  console.log('Example app listening at http://%s:%s', host, port);
```

回调函数可以简单理解为函数接收多个参数，其中最后一个参数是一个函数，里面包含了对传入数据的处理方式，这在js的编程中是一个较普遍的概念。

Koa -- 基于 Node.js 平台的下一代 web 开发框架

简介

koa 是由 Express 原班人马打造的，致力于成为一个更小、更富有表现力、更健壮的 Web 框架。 使用 koa 编写 web 应用，通过组合不同的 generator，可以免除重复繁琐的回调函数嵌套， 并极大地提升错误处理的效率。koa 不在内核方法中绑定任何中间件， 它仅仅提供了一个轻量优雅的函数库，使得编写 Web 应用变得得心应手。

hello world

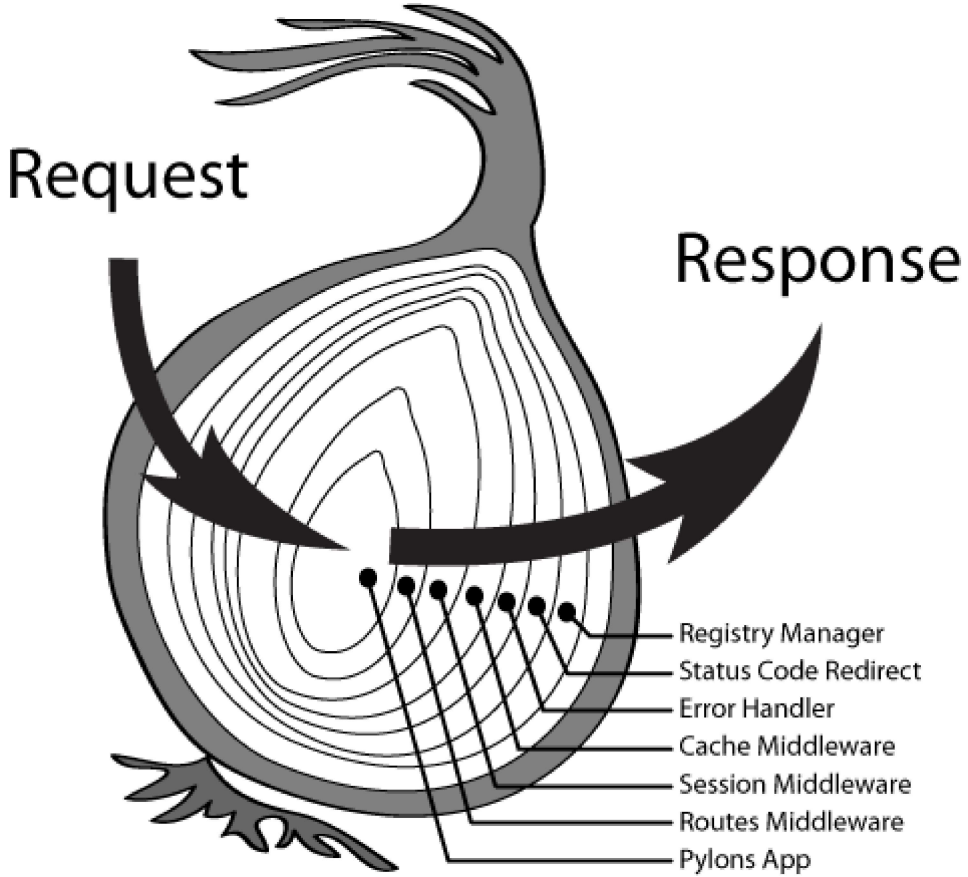
```js
const Koa = require('koa');
const app = new Koa();

app.use(async ctx => {
  ctx.body = 'Hello World';
});

app.listen(3000);
```

Middleware

koa的洋葱圈模型:



```js
const Koa = require('koa');
const app = new Koa();

// x-response-time
app.use(async (ctx, next) => {
  const start = Date.now();
  await next();
  const ms = Date.now() - start;
  ctx.set('X-Response-Time', `${ms}ms`);
});

// logger
app.use(async (ctx, next) => {
  const start = Date.now();
  await next();
  const ms = Date.now() - start;
  console.log(`${ctx.method} ${ctx.url} - ${ms}`);
});

// response
app.use(async ctx => {
  ctx.body = 'Hello World';
});

app.listen(3000);
```

app.context

app.context 是 ctx 的原型。

```js
const Koa = require('koa');
const app = new Koa();
// ...
app.context.db = db();

app.use(async ctx => {
  console.log(ctx.db);
});
```

快速初始化

```dos
$ npm i egg-init -g
$ egg-init egg-example --type=simple
$ cd egg-example
$ npm i
```

跟着教程一步步学习

添加控制器

```js
// ./app/controller/api.js
"use strict";

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

class ApiController extends Controller {
    async index() {
        this.ctx.body = {
            status: 200,
            data: [{
                name: "zhangsan",
                age: 21
            }]
        };
    }
```

```js
// ./app/router.js
"use strict";
module.exports = app => {
    const { router, controller } = app;
    router.get("/", controller.home.index);
    router.get("/api", controller.api.index);
```

快速入门

连接mysql数据库

egg-mysql

```dos
$ npm i egg-mysql --save
```

在${app_root}/config/plugin.js启用插件:

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

在${app_root}/config/config.default.js添加mysql配置信息:

```js
config.mysql = {
        // database configuration
        client: {
            // host
            host: "localhost",
            // port
            port: "3306",
            // username
            user: "root",
            // password
            password: "root",
            // database
            database: "api"
        },
        // load into app, default is open
        app: true,
        // load into agent, default is close
        agent: false
    };
```

query data

controller:

```js
async apiList() {
        const mysql = this.ctx.app.mysql;
        const data = await mysql.query("select * from sys_api");

        this.ctx.body = {
            status: 200,
            data
        };
    }
```

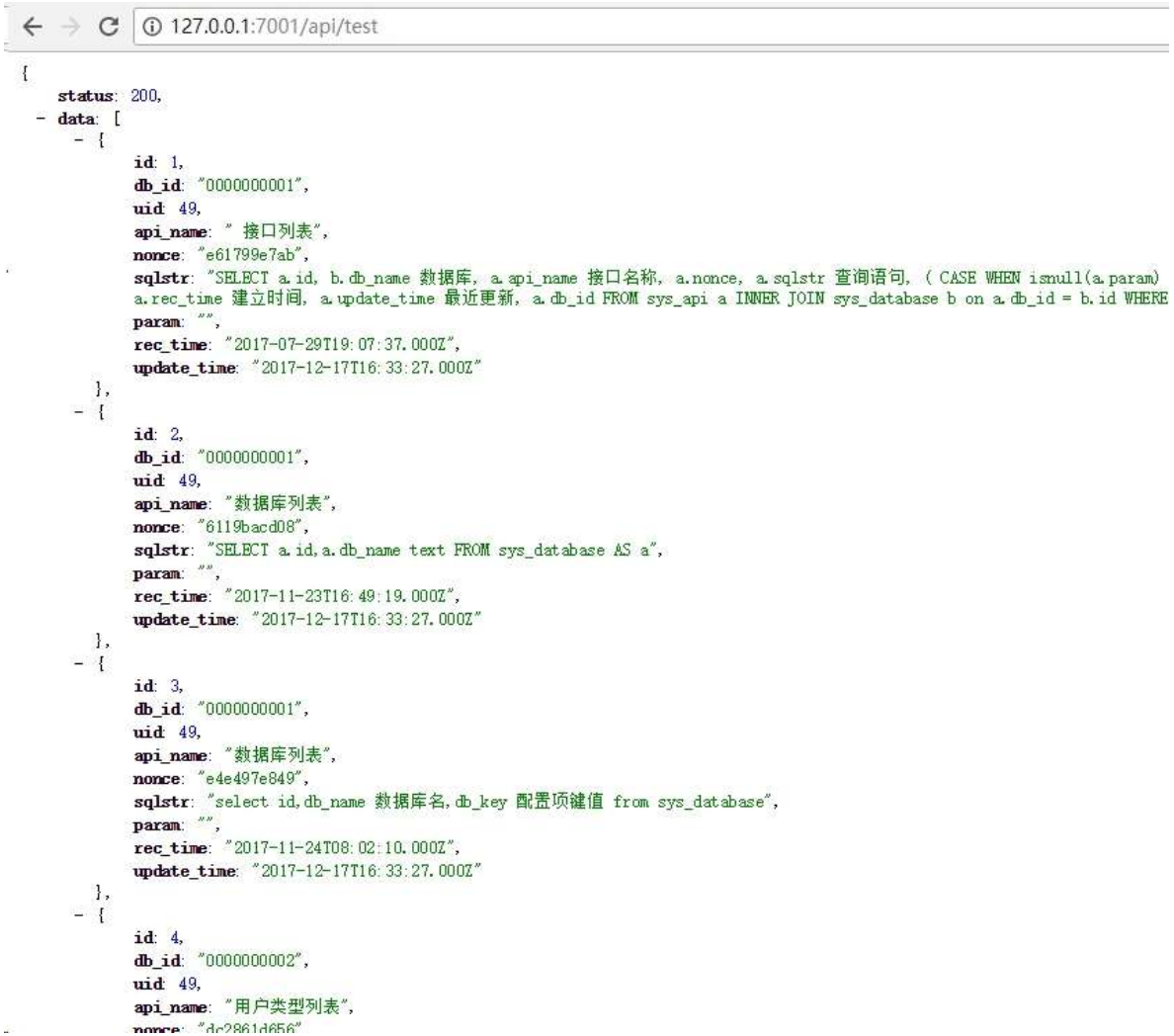
router:

```js
router.get("/api/list", controller.api.apiList);
```

自定义SQL语句联接

> const results = await app.mysql.query('update posts set hits = (hits + ?) where id = ?', [1, postId]);

测试结果


127.0.0.1:7001/api/test
{
status: 200,
data: [
{
id: 1,
db_id: "0000000001",
uid: 49,
api_name: " 接口列表",
nonce: "e61799e7ab",
sqlstr: "SELECT a.id, b.db_name 数据库, a.api_name 接口名称, a.nonce, a.sqlstr 查询语句, (CASE WHEN ismull(a.param)
a.rec_time 建立时间, a.update_time 最近更新, a.db_id FROM sys_api a INNER JOIN sys_database b on a.db_id = b.id WHERE
param: "",
rec_time: "2017-07-29T19:07:37.000Z",
update_time: "2017-12-17T16:33:27.000Z"
},
{
id: 2,
db_id: "0000000001",
uid: 49,
api_name: "数据库列表",
nonce: "6119bacd08",
sqlstr: "SELECT a.id,a.db_name text FROM sys_database AS a",
param: "",
rec_time: "2017-11-23T16:49:19.000Z",
update_time: "2017-12-17T16:33:27.000Z"
},
{
id: 3,
db_id: "0000000001",
uid: 49,
api_name: "数据库列表",
nonce: "e4e497e849",
sqlstr: "select id,db_name 数据库名,db_key 配置项键值 from sys_database",
param: "",
rec_time: "2017-11-24T08:02:10.000Z",
update_time: "2017-12-17T16:33:27.000Z"
},
{
id: 4,
db_id: "0000000002",
uid: 49,
api_name: "用户类型列表",

CRUD

遵循 **官网说明** 编写相应代码即可。

添加更多的功能

待更新