

Egg.js基础(上)

成钞公司印钞管理部 李宾



2018-01-11



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。

```
1 node -v
2 v8.7.0
```

dos

NPM

node.js中自带的包管理工具。

[官网](#) [文档](#)

```
1 cd yourAppDir
2 e:\yourAppDir > npm install --save echarts
3 e:\yourAppDir > npm install --save-dev webpack
4 e:\yourAppDir > npm uninstall --save echarts
5 c:\ > npm install -g vue-cli
```

dos

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的镜像，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。

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

dos

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:

```
1 // app.js
2 const http = require('http');
3
4 const hostname = '127.0.0.1';
5 const port = 3000;
6
7 const server = http.createServer((req, res) => {
8   res.statusCode = 200;
9   res.setHeader('Content-Type', 'text/plain');
10  res.end('Hello World\n');
11 });
12
13 server.listen(port, hostname, () => {
14   console.log(`Server running at http://${hostname}:${port}/`);
15 });
```

启动服务

```
1 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 7.6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。

```
1 var express = require('express');
2 var app = express();
3
4 // 这里的function称为回调函数, callback function.
5 app.get('/', function (req, res) {
6   res.send('Hello World!');
7 });
8
9 var server = app.listen(3000, function () {
10   var host = server.address().address;
11   var port = server.address().port;
12
13   console.log('Example app listening at http://%s:%s', host, port);
14 });
```

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

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。

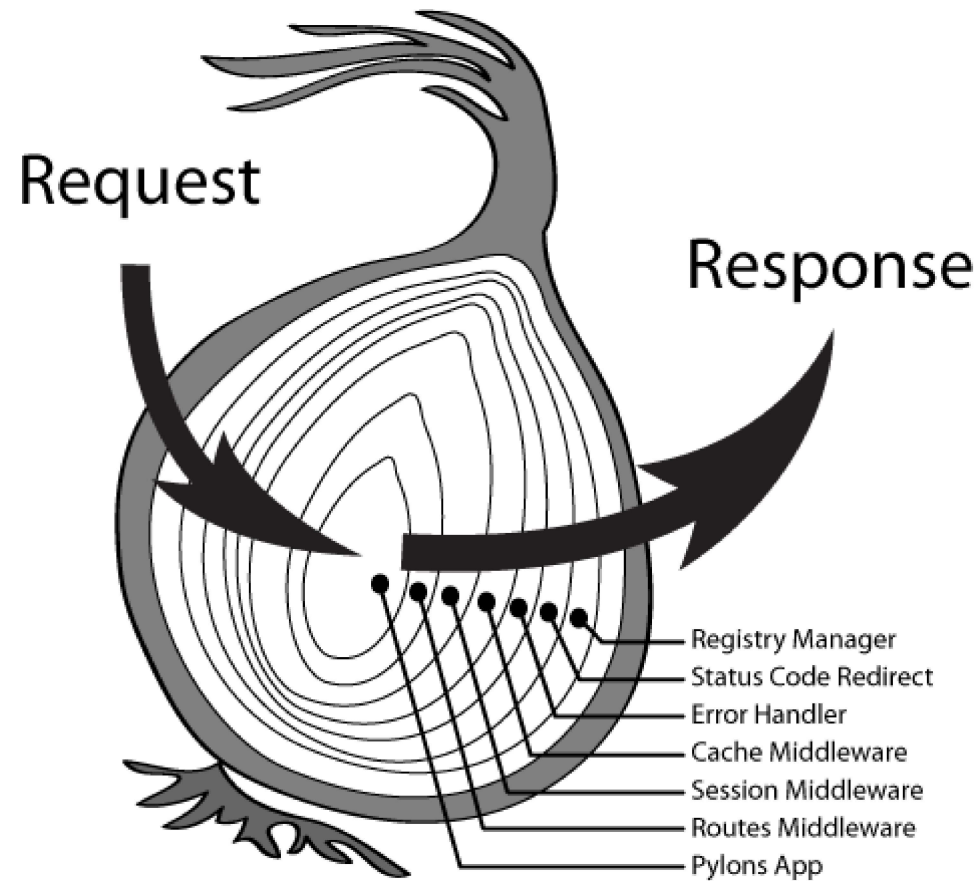
hello world

```
1  const Koa = require('koa');  
2  const app = new Koa();  
3  
4  app.use(async ctx => {  
5    ctx.body = 'Hello World';  
6  });  
7  
8  app.listen(3000);
```

js

Middleware

koa的洋葱圈模型:



```
1  const Koa = require('koa');
2  const app = new Koa();
3
4  // x-response-time
5  app.use(async (ctx, next) => {
6    const start = Date.now();
7    await next();
8    const ms = Date.now() - start;
9    ctx.set('X-Response-Time', `${ms}ms`);
10 });
11
12 // logger
13 app.use(async (ctx, next) => {
14   const start = Date.now();
15   await next();
16   const ms = Date.now() - start;
17   console.log(`${ctx.method} ${ctx.url} - ${ms}`);
18 });
19
20 // response
21 app.use(async ctx => {
22   ctx.body = 'Hello World';
23 });
24
25 app.listen(3000);
```

app.context

app.context 是 ctx 的原型。

```
1  const Koa = require('koa');  
2  const app = new Koa();  
3  // ...  
4  app.context.db = db();  
5  
6  app.use(async ctx => {  
7    console.log(ctx.db);  
8  });
```

js

快速初始化

```
1 $ npm i egg-init -g
2 $ egg-init egg-example --type=simple
3 $ cd egg-example
4 $ npm i
```

dos

跟着教程一步步学习

添加控制器

```
1 // ./app/controller/api.js
2 "use strict";
3
4 const Controller = require("egg").Controller;
5
6 class ApiController extends Controller {
7   async index() {
8     this.ctx.body = {
9       status: 200,
10      data: [{
11        name: "zhangsan",
12        age: 21
13      }]
14    };
15  }
```

```
1 // ./app/router.js
2 "use strict";
3 module.exports = app => {
4   const { router, controller } = app;
5   router.get("/", controller.home.index);
6   router.get("/api", controller.api.index);
7 }
```

快速入门

连接mysql数据库

egg-mysql

```
1 $ npm i egg-mysql --save
```

dos

在`${app_root}/config/plugin.js`启用插件:

```
1 exports.mysql = {  
2   enable: true,  
3   package: 'egg-mysql',  
4 };
```

js

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

```
1 config.mysql = {  
2     // database configuration  
3     client: {  
4         // host  
5         host: "localhost",  
6         // port  
7         port: "3306",  
8         // username  
9         user: "root",  
10        // password  
11        password: "root",  
12        // database  
13        database: "api"  
14    },  
15    // load into app, default is open  
16    app: true,  
17    // load into agent, default is close  
18    agent: false  
19 };
```

js

query data

controller:

```
1  async apiList() {  
2      const mysql = this.ctx.app.mysql;  
3      const data = await mysql.query("select * from sys_api");  
4  
5      this.ctx.body = {  
6          status: 200,  
7          data  
8      };  
9  }
```

router:

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

拼接 sql 语句

!

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

测试结果

```
< → C 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 isnull(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",
      nonce: "dc2861d666"
```

CRUD

遵循 **官网说明** 编写相应代码即可。

连接MS SQL

node-mssql 文档

```
1 // ./app/lib/mssql.config.js
2 const mssql = {
3   user: "sa",
4   password: "123",
5   server: "localhost", // You can use 'localhost\\instance' to connect to named
6   database: "api",
7   options: {
8     encrypt: false // Use this if you're on Windows Azure
9   }
10 };
11
12 module.exports = mssql;
```

封装查询函数

```
1 // ./app/lib/mssql.js
2 const config = require("./mssql.config");
3 const sql = require("mssql");
4
5 const query = async sqlStr => {
6     const pool = await sql.connect(config);
7     const result = await pool.request().query(sqlStr);
8     await sql.close();
9     return {
10         rows: result.rowsAffected[0],
11         header: result.recordset.length === 0 ? [] : Object.keys(result.recordset),
12         data: result.recordset
13     };
14 };
15
16 module.exports = { query };
```


使用

```
1 const mssql = require("../lib/mssql");
2 async apiMSSql() {
3     const data = await mssql.query("select * from tblApi");
4     this.ctx.body = data;
5 }
```

js

结果

```
← → ↻ ⓘ 127.0.0.1:7001/api/test

{
  rows: 16,
  - header: [
    "ID",
    "ApiID",
    "ApiName",
    "AuthorName",
    "Token",
    "ApiDesc",
    "strSQL",
    "Params",
    "DBID",
    "URL",
    "CreateDate"
  ],
  - data: [
    - {
      ID: 1,
      ApiID: 0,
      ApiName: "数据接口列表 ",
      AuthorName: "develop ",
      Token: "79d84495ca776ccb523114a2120e273ca80b315b ",
      ApiDesc: " [功能说明] <span>本接口主要用于 <i>凹印离线设备每天得分汇总</i>&nbsp;信息的查询.</span> [主要参数] <ol> <li> TimeStar
      strSQL: "UOVMRUNUIGEuSUQgYXMGsUQsYS5BcG1JRCBhcyDluo/1j7csYS5BdXR0b3J0YVW1LlIGFzIOaJg0acieiAhSxiLkRCTmFtZSBhcyDmlbDmja7lupMsYS5BcG
      Params: "author",
      DBID: 1,
      URL: "http://localhost/DataInterface/Api?Token=79d84495ca776ccb523114a2120e273ca80b315b&ID=2&M=3",
      CreateDate: null
    },
    - {
      ID: 2,
      ApiID: 1,
      ApiName: "选择列表自定义设置 ",
      AuthorName: "develop ",
      Token: "79d84495ca776ccb523114a2120e273ca80b315b ",
      ApiDesc: "<p>[功能说明]</p> <p style='text-indent: 2em;'>本接口主要用于 <i>XX</i> 信息的查询.</p> <p>[主要参数]</p> <ol> <li> ts
      strSQL: "c2VsZWNOIFZhbHV1LE5hbWUsYXBpVjJMLHRibE1EIGZyb20gdGJsU2V0dGluZ3NfU2VsZWNOO0xpc3Q=",
      Params: "",
      DBID: 1,
      URL: "http://localhost/DataInterface/Api?Token=79d84495ca776ccb523114a2120e273ca80b315b&ID=34&M=3",
      CreateDate: null
    },
  ],
}
```

改进

- 基于快速实现功能的目的将mssql的连接和查询封装为了独立的模块，此处可考虑将其封装为egg的插件，方便直接 **注入 this.app 中**。
- 此处连接数据库-->查询-->关闭数据库，可考虑合理使用连接池， **将连接态全局保存**；
- 封装更多的功能

oracle

文档

后续的数据库连接操作自行根据官网文档编写业务代码。

Q & A