

19 | 全栈项目搭建：如何搭建Vue.js的前后台全栈项目？

2023-01-06 杨文坚 来自北京

天下无鱼
<https://shikey.com/>

《Vue 3 企业级项目实战课》

课程介绍 >



讲述：杨文坚

时长 11:06 大小 10.14M



你好，我是杨文坚。

前几节课中，我们基于 Node.js 的 Koa.js 搭建了 Web 服务，设计了 Node.js 项目的前后端分离，学习了如何基于 Node.js 在服务端渲染 Vue.js 页面，以及前后端项目如何联动实现 Vue.js 的同构渲染。数一数你掌握的项目能力，Web 服务开发、Node.js 服务端分层，Vue.js 前端渲染和服务端渲染，看起来前后端能力都有了，应该就是全栈项目吧？

如果从广义上理解，确实已经算是全栈项目了，但是从实际项目角度上分析，还不满足我们课程中运营搭建平台的“全栈”功能需要，比如，还缺少数据存储层的设计、前台和后台的服务隔离设计（这里的前台指的是面向客户的 Web 服务，后台指的是面向管理员的 Web 服务）。

这么多新概念，不知道你有没有觉得有点云里雾里的？不用困惑，今天我们会逐一分析，最终搭建出自己的全栈项目。

首先老规矩，先分析一下我们的课程项目，完整的全栈系统设计需要做什么准备？

全栈系统设计需要准备什么？



我们最终的项目是实现一个运营搭建平台，需要考虑项目的用户人群。运营搭建平台是给企业员工用的，快速搭建网页，然后把网页输出给客户使用。

那就要考虑两个用户维度：员工、客户。在大厂的项目中，这类场景**需要把平台拆分成两个独立的服务**。拆分主要从“安全”、“稳定”和“便于维护”三个因素考虑的。

- **安全**，指的是平台功能、数据等安全因素。

设想一下，如果员工和客户的功能都部署在同一个 **Web** 网站服务里，企业员工可以搭建和发布页面，客户可以浏览页面，如果出现“水平越权”的漏洞，也就是普通客户可以在同一个网站里搭建和修改页面，那将会是很恐怖的事情。

- **稳定**，指的是功能使用的稳定。

如果都部署在同一个 **Web** 网站服务里，员工使用的某个功能导致服务崩溃了，这时候，员工操作不了页面，客户也浏览不了页面，将会出现整体平台崩溃。如果员工和客户使用的平台是两套服务系统，其中一个奔溃了，不会影响另外一个的使用。

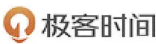
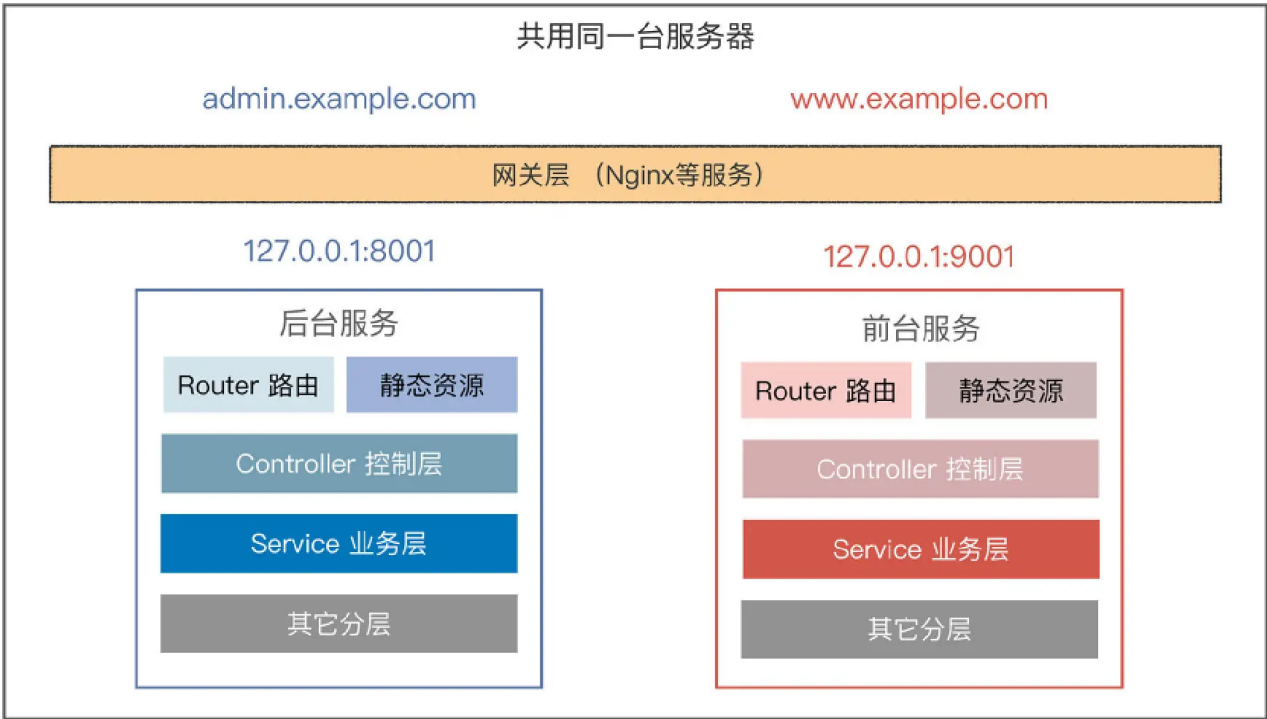
- **便于维护**，指的是便于后续升级或者改造。

员工和客户两个独立 **Web** 服务，当客户使用量增加，服务器扛不住压力了，可以只针对客户的独立服务做扩容，不需要关注员工服务的影响。

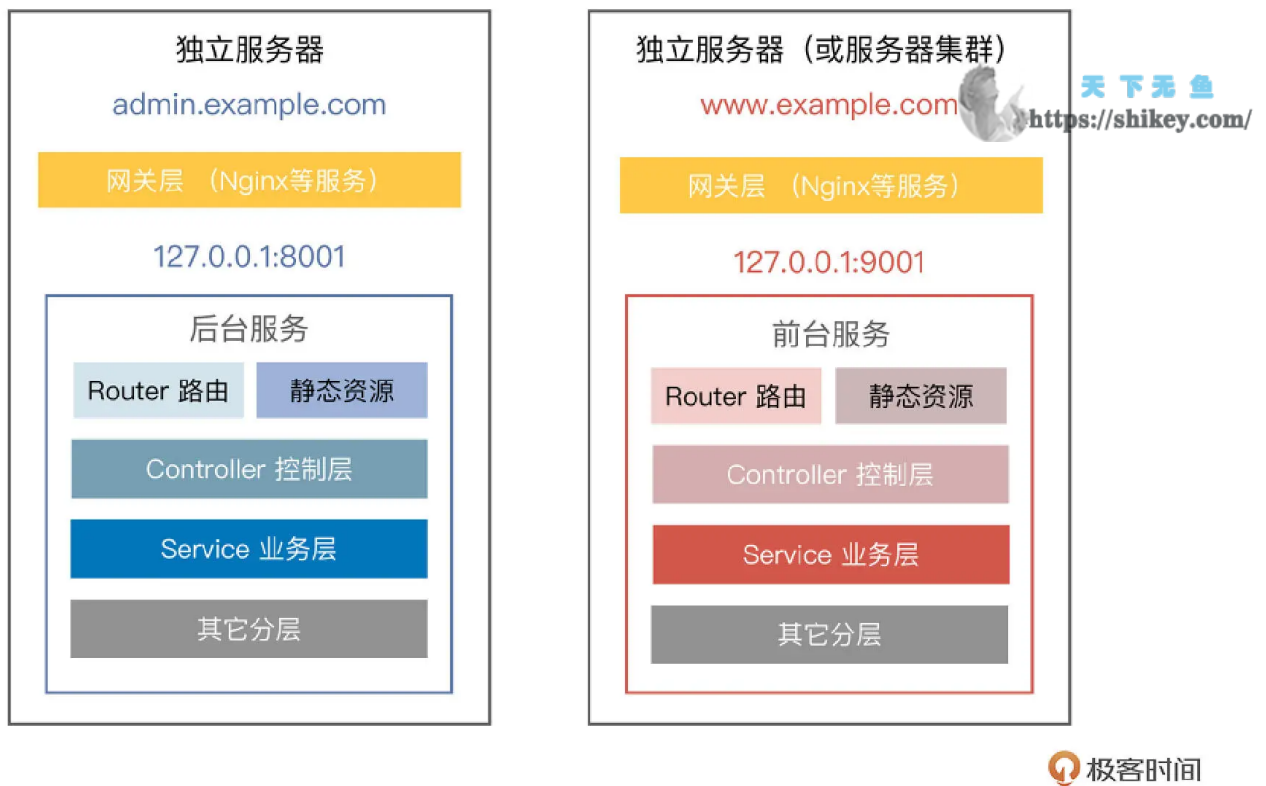
既然要前台后台服务分离，如何设计呢？我整理了三步。

- 第一步：**Web** 服务的拆分和隔离
- 第二步：前台和后台 **Web** 服务的系统设计
- 第三步：前后台的数据和资源的共享设计

第一步，Web 服务的拆分和隔离。如果业务体量不大的话，前台和后台两个 Web 服务可以独立部署，在同一台服务器上，各自使用独立的服务端口。同时，还需要使用类似 Nginx 的网关服务，进行同一机器上不同端口服务的域名端口转发管理。就像这样：



如果业务体量变大了，例如搭建的页面访问用户量增大，服务器压力扛不住了，可以把面向用户的前台 Web 服务独立出来，用集群来部署，以应付前台大量的用户请求。就像这样：

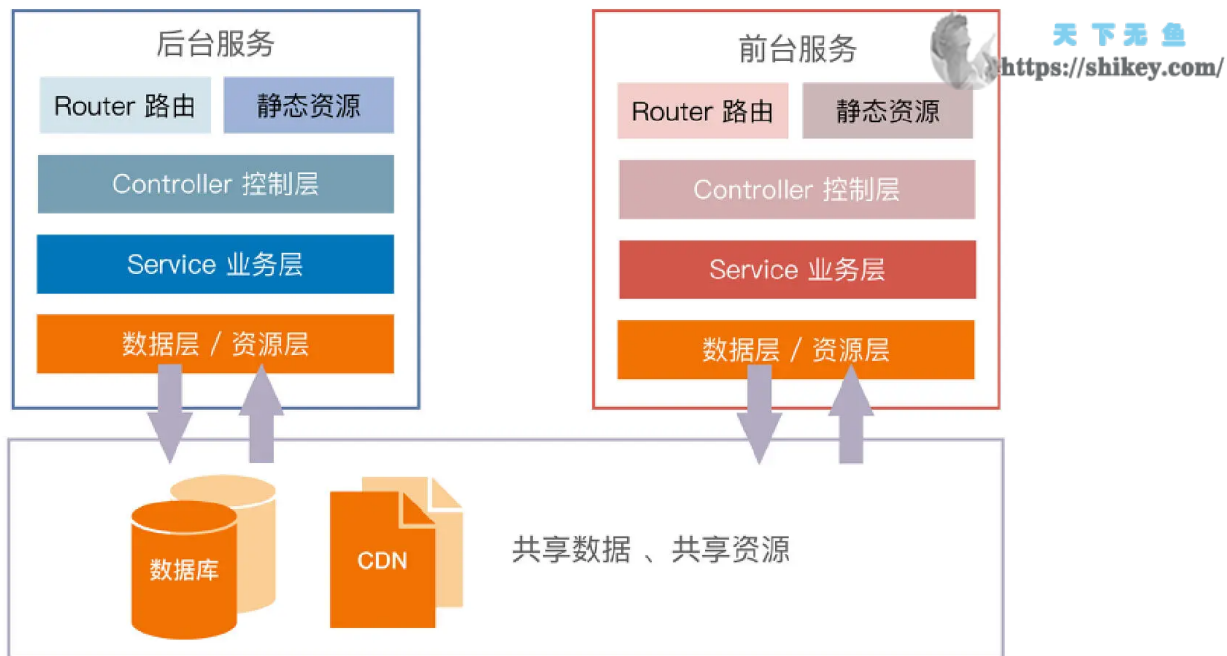


第二步，前台和后台 Web 服务的系统设计。

两个 Web 服务，都按照我们之前讲过的“前后端项目分离”的形式，服务系统的设计都是一样，只不过后续要实现不一样的操作功能。

- 后台服务，面向企业内部员工，服务核心是提供搭建页面能力，那么最重要的就是员工操作权限控制，页面搭建和发布的流程管理。
- 前台服务，面向外部客户，服务核心是提供能力渲染“已搭建好”的页面。这里的渲染能力需要支持 SSR，也就是服务端渲染，能支持 SEO，也就是浏览器引擎优化。

第三步，就是前后台的资源和数据的共享设计，大致是这样：



极客时间

我们可以看到系统分层，最底层是操作共享的数据和资源。其中，共享的资源主要是搭建页面用到的物料静态资源（JavaScript、CSS 文件等），以及搭建成功后生成的页面静态资源、用户浏览页面的静态资源。共享的数据是搭建搭建页面的配置数据、操作记录、发布记录等。

对于共享的静态资源操作，一般企业里，做法是将静态资源文件存放到 CDN 云服务上。

CDN，全称 Content Delivery Network，中文意思就是“内容分发网络”。CDN 支持存放多种静态资源，并且将资源分布到多个地区的服务器上，用户请求资源时可以就近获取服务器上的内容。CDN 是需要向云服务厂商购买的服务，不同云服务厂商提供的 CDN 使用方式都有差异。

为了方便演示，在课程源码的 monorepo 项目中，我就用一个子项目来代替共享的 CDN 资源，也就是把 JavaScript、CSS 等文件用子项目形式来共享使用。如果你以后想实践在企业项目中的话，替换成企业购买的 CDN 服务就行。

对于共享数据的操作，就需要用到数据库。一般在企业里，数据库有两种使用方式，自建数据库服务、向服务厂商买数据库服务。如果是自建数据库，需要付出精力学习和做好数据库运维工作，例如数据库扩容、数据库读写分离、数据库备份等等。如果是购买厂商的数据库服务，数据库的运维可以靠购买增值服务解决。

课程里，我们就用自建数据库的方式来实现功能，至于数据库的运维等操作，已经超出普通前端程序员和普通后端程序员的职能范畴了，不在课程内涉及。



好，接下来我们就自建数据库操作，开始前照例我们需要准备数据库环境。

如何准备数据库服务环境？

首先，我们要选择数据库。数据库选择很多，MySQL、MongoDB 等等。那要怎么选呢？

不卖关子，这里我先放出答案，我们会用 MySQL 这一关系型数据库来作为项目数据库，主要是考虑“**关系型数据库**”和“**国内企业使用情况**”这两个因素。

第一个因素，MySQL 是关系型数据库，而且是被很多企业广泛使用的数据库。

关系型数据库，你可以这么理解，就是将不同维度的数据，例如用户数据、页面数据等数据，用独立的“数据表”来承载，通过“数据表”里的每个数据的 ID，来进行表之间的数据关联。

换个更简单的角度，你可以类比成用 Excel 来管理数据。把数据依照不同数据的差异，归纳放在不同 Excel 表格里来管理。这样，同一维度的数据维护比较清晰，不同维度数据，也能通过 Excel 的行列关系来进行关联，使得数据的联动能灵活组合。

另一个因素“国内企业使用情况”。国内大部分企业，基本都是用 MySQL 或者“类 MySQL”的数据进行系统开发。首要原因是“MySQL 社区版”是免费的，而且技术生态和资料都很丰富，对于企业来讲，可以节省很大资金成本和学习成本。

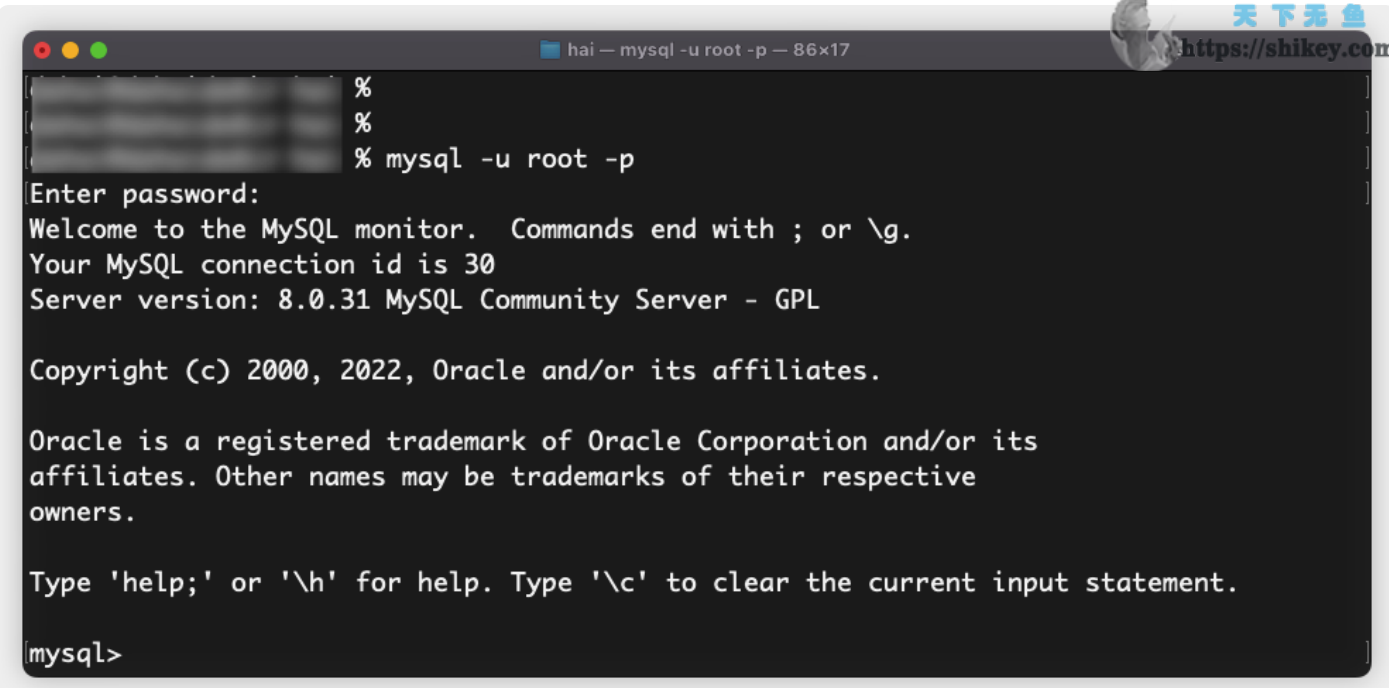
由于国内的 MySQL 有庞大的使用群体，很多大厂自研数据库的设计，都使用或者兼容 MySQL 的操作语法，例如阿里巴巴内部自研的数据库、阿里云的提供数据库服务等，基本都兼容 MySQL 语法。

选好数据库，接下来就是如何搭建数据库环境了。

首先从官网下载 MySQL 的社区版服务安装包（

<https://dev.mysql.com/downloads/mysql/>），你可以根据自己电脑系统情况选择安装。安装过程很简单，就是根据提醒设置好账号和密码。为了方便讲解，案例源码我就直接使用 root 账号。

安装成功后，用 root 账号登录一下，看看是否能访问默认的数据库内容：

A terminal window titled 'hai - mysql -u root -p - 86x17' with a URL 'https://shikey.com/' in the top right corner. The terminal shows the command 'mysql -u root -p' being entered. It prompts for a password, then displays the MySQL welcome message: 'Welcome to the MySQL monitor. Commands end with ; or \g. Your MySQL connection id is 30 Server version: 8.0.31 MySQL Community Server - GPL'. It also shows the copyright notice for Oracle and the instruction to type 'help;' or '\h' for help. The prompt 'mysql>' is visible at the bottom.

```
hai - mysql -u root -p - 86x17 https://shikey.com/
%
%
% mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 30
Server version: 8.0.31 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

接下来，我们就用 Node.js 来操作数据库，这里要注意，刚刚我们安装的是 MySQL 的“服务端”，现在需要用“客户端”来进行服务端操作。Node.js 的生态里有 MySQL 的客户端 npm 模块 mysql，我们就使用它来进行 MySQL 的数据库操作。

我先演示一下数据库“建库”的操作代码。

 复制代码

```
1  /* eslint-disable no-console */
2  import mysql from 'mysql';
3
4  // 需要创建的数据库名称
5  const database = 'hello_vue_project';
6  // 数据库连接配置
7  const config = {
8    host: '127.0.0.1',
9    port: 3306,
10   user: 'root',
11   password: 'xxxxxx' // 这里需要填写自己电脑本地MySQL数据库的密码
12 };
13
14 // 创建一个数据库连接池
15 // 用来建库
16 const pool = mysql.createPool(config);
17
18 // 封装连接池执行方法
19 function querySQLByPool(sql: string) {
20   return new Promise((resolve, reject) => {
```

```

21     pool.query(sql, (err, results, fields) => {
22         if (err) {
23             pool.end();
24             reject(err);
25         } else {
26             pool.end();
27             resolve({ results, fields });
28         }
29     });
30 });
31 }
32
33 async function init() {
34     // 建库 SQL 语句
35     const sqlDB = `CREATE DATABASE IF NOT EXISTS ${database}`;
36     // 执行建库操作
37     await querySQLByPool(sqlDB);
38     console.log(`运营搭建平台 - 数据库 ${database} 建库成功!`);
39 }
40
41 // 开始数据库初始化
42 init();
43

```



上述代码中，我在 Node.js 环境里，用 SQL 语法实现了一个数据库“hello_vue_project”的创建，代码里面的账号密码都是本地 MySQL 数据库的。我这里用 root 主要是方便调试，**实际企业项目中，必须创建一个新的账户来操作数据库，同时，密码也不能像这里那么简单。**

还有个小提醒，如果你使用最新版 MySQL 操作数据库，遇到如下报错。

复制代码

```
1 Error: ER_NOT_SUPPORTED_AUTH_MODE: Client does not support authentication proto
```

```

my-demo - zsh - 82x11
> init:db
> vite-node ./packages/init-database/src/index.ts

node:internal/process/promises:288
    triggerUncaughtException(err, true /* fromPromise */);
    ^

Error: ER_NOT_SUPPORTED_AUTH_MODE: Client does not support authentication protocol
requested by server; consider upgrading MySQL client
    at Sequence._packetToError (/node_modules/.../node_modules/mysql/lib/protocol/sequences/Sequence.js:47:14)

```


这个问题是由于 MySQL 最新版本的加密方式和 Node.js 的 mysql 客户端不一致产生的，通过以下 MySQL 脚本，在 MySQL 命令框里修改就能解决问题。



复制代码

```
1 # 更新root账号密码，用其他加密形式处理
2 # 里面的 'password' 为实际的密码
3 ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'password'
4
5 # 刷新配置
6 FLUSH PRIVILEGES;
```

接下来就进入数据库建表，我来演示一下数据库的“建表”操作代码。

复制代码

```
1 /* eslint-disable no-console */
2 import mysql from 'mysql';
3
4 // 需要创建的数据库名称
5 const database = 'hello_vue_project';
6 // 数据库连接配置
7 const config = {
8   host: '127.0.0.1',
9   port: 3306,
10  user: 'root',
11  password: '1234abcd'
12 };
13
14 // 创建一个数据库连接池
15 // 用来建表
16 const poolDatabase = mysql.createPool({ ...config, ...{ database } });
17
18 // 封装连接池执行方法
19 function queryDatabaseSQLByPool(sql: string) {
20   return new Promise((resolve, reject) => {
21     poolDatabase.query(sql, (err, results, fields) => {
22       if (err) {
23         poolDatabase.end();
24         reject(err);
25       } else {
26         poolDatabase.end();
27         resolve({ results, fields });
28       }
29     });
30   });
31 }
32
33 async function init() {
```

```

34 // 建库 SQL 语句
35 const sqlDB = `
36 CREATE TABLE IF NOT EXISTS \`user_info\` (
37   \`id\` int(11) NOT NULL AUTO_INCREMENT,
38   \`uuid\` varchar(128) NOT NULL UNIQUE COMMENT '员工用户UUID',
39   \`username\` varchar(64) NOT NULL UNIQUE COMMENT '员工用户名称',
40   \`password\` varchar(64) NOT NULL COMMENT '员工用户密码',
41   \`info\` json COMMENT '扩展描述（JSON数据格式）',
42   \`create_time\` datetime DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
43   \`modify_time\` datetime DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
44   PRIMARY KEY (\`id\`)
45 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
46 `;
47 // 执行建表操作
48 await queryDatabaseSQLByPool(sqlDB);
49 console.log('运营搭建平台 - 数据表 user_info 创建成功!');
50 }
51
52 // 开始数据库初始化
53 init();
54

```



天下无鱼

<https://shikey.com/>

上述代码中，用 SQL 语法，在数据库“hello_vue_project”，实现了一个表“user_info”的创建，用来存储用户数据信息。

再接着，用 Node.js 在表里查询数据的操作，代码如下所示。

复制代码

```

1  /* eslint-disable no-console */
2  import mysql from 'mysql';
3  import type { OkPacket } from 'mysql';
4
5  // 数据库名称
6  const database = 'hello_vue_project';
7  // 数据库连接配置
8  const config = {
9    host: '127.0.0.1',
10   port: 3306,
11   user: 'root',
12   password: 'xxxxx'
13 };
14
15 // 封装连接执行方法
16 function queryDatabaseSQL(sql: string, values: (string | number)[]) {
17   const conn = mysql.createConnection({ ...config, ...{ database } });
18   conn.connect();
19   return new Promise<OkPacket>((resolve, reject) => {
20     conn.query(sql, values, (err, rows: OkPacket) => {

```

```

21     if (err) {
22         conn.end();
23         reject(err);
24     } else {
25         conn.end();
26         resolve(rows);
27     }
28 });
29 });
30 }
31
32 async function init() {
33     // 插入数据 SQL 语句
34     const sql = `
35         INSERT INTO user_info
36         ( uuid, username, password, info, create_time, modify_time)
37         VALUES
38         ('00000000-aaaa-bbbb-cccc-ddddeeee0001','hello-vue-004', '1f22f6ce6e58a7326
39         ('00000000-aaaa-bbbb-cccc-ddddeeee0002','hello-vue-005', '1f22f6ce6e58a7326
40         ('00000000-aaaa-bbbb-cccc-ddddeeee0003','hello-vue-006', '1f22f6ce6e58a7326
41     `;
42     // 执行插入数据操作
43     const data: OkPacket = await queryDatabaseSQL(sql, []);
44     console.log(
45         `运营搭建平台 - 数据表 user_info 成功插入${data?.affectedRows}条数据`
46     );
47 }
48
49 // 开始执行数据库操作
50 init();

```



天下无鱼

<https://shikey.com/>

上述代码中，执行了 SQL 的 INSERT 数据操作，将 3 条数据插入到“user_info”数据库表中。

这里，我演示一下在表里查询数据的操作。

 复制代码

```

1  /* eslint-disable no-console */
2  import mysql from 'mysql';
3  import type { OkPacket } from 'mysql';
4
5  // 数据库名称
6  const database = 'hello_vue_project';
7  // 数据库连接配置
8  const config = {
9      host: '127.0.0.1',
10     port: 3306,
11     user: 'root',
12     password: 'xxxx'

```

```

13 };
14
15 // 封装连接执行方法
16 function queryDatabaseSQL(sql: string, values: (string | number)[]) {
17     const conn = mysql.createConnection({ ...config, ...{ database } });
18     conn.connect();
19     return new Promise<OkPacket | unknown[]>((resolve, reject) => {
20         conn.query(sql, values, (err, rows: OkPacket) => {
21             if (err) {
22                 conn.end();
23                 reject(err);
24             } else {
25                 conn.end();
26                 resolve(rows);
27             }
28         });
29     });
30 }
31
32 async function init() {
33     // 查询数据 SQL 语句
34     const sql = `
35         SELECT username, create_time FROM user_info WHERE id = ?
36     `;
37     // 执行插入数据操作
38     const data: unknown[] = (await queryDatabaseSQL(sql, [1])) as unknown[];
39     console.log(`运营搭建平台 - 数据表 user_info 成功查询${data?.length}条数据`);
40     console.log(data);
41 }
42
43 // 开始执行数据库操作
44 init();
45

```

上述代码，是用 SQL 的 SELECT 语句进行查询数据操作，查找了 id 为 1 的用户数据。结果返回是一个对象数组。

最后演示一下表的更新数据操作。

 复制代码

```

1 /* eslint-disable no-console */
2 import mysql from 'mysql';
3 import type { OkPacket } from 'mysql';
4
5 // 数据库名称
6 const database = 'hello_vue_project';
7 // 数据库连接配置
8 const config = {

```

```

9     host: '127.0.0.1',
10    port: 3306,
11    user: 'root',
12    password: '1234abcd'
13 };
14
15 // 封装连接执行方法
16 function queryDatabaseSQL(sql: string, values: (string | number)[]) {
17     const conn = mysql.createConnection({ ...config, ...{ database } });
18     conn.connect();
19     return new Promise<OkPacket | unknown[]>((resolve, reject) => {
20         conn.query(sql, values, (err, rows: OkPacket) => {
21             if (err) {
22                 conn.end();
23                 reject(err);
24             } else {
25                 conn.end();
26                 resolve(rows);
27             }
28         });
29     });
30 }
31
32 async function init() {
33     // 更新数据 SQL 语句
34     const sql = `
35     UPDATE user_info SET username=? WHERE id=?;
36     `;
37
38     // 执行更新数据操作
39     const data: OkPacket = (await queryDatabaseSQL(sql, [
40         'hi_vue_001',
41         1
42     ])) as OkPacket;
43     console.log(
44         `运营搭建平台 - 数据表 user_info 成功更新${data?.affectedRows}条数据`
45     );
46     console.log(data);
47 }
48
49 // 开始执行数据库操作
50 init();
51

```



天下无鱼

<https://shikey.com/>

上述代码是用 SQL 的 UPDATE 语句进行查询数据操作，将 id 为 1 的的用户名称更新。

这里，我们展示了 MySQL 在 Node.js 环境里的“建库”、“建表”和数据的“增改查”的操作实现。不知道你有没有发现，缺少了“删除”数据的操作。

MySQL 本身有提供删除数据的操作，但是实际项目中，我们不会真的删除数据，只会用“假删除”的方式进行删除数据，例如给表加个字段标注是否删除。这样做，是为了避免误删数据导致故障或者损失。




好，在 Node.js 环境里操作 MySQL 的基本内容我们就了解得差不多了，那么我们如何在课程项目中做数据库的配置呢？

如何搭建全栈项目的数据库配置？


数据库的配置首先要区分环境，比如本地开发环境、测试环境和生产环境，你可以通过 Node.js 进程的环境变量进行控制。我推荐用 `dotenv` 这个 npm 模块，用来基于环境变量管理 MySQL 的配置，看看如何操作。

先在项目的根目录建一个 `“.env”` 文件，并且将环境变量配置到该文件里。

 复制代码

```
1 # .env 文件
2 MYSQL_HOST = "127.0.0.1"
3
4 MYSQL_PORT = "3306"
5
6 MYSQL_DATABASE = "my_vue_project_20"
7
8 MYSQL_USER = "root"
9
10 MYSQL_PASSWORD = "xxxx"
```

在项目服务端代码中，可以这么来使用环境变量。

 复制代码

```
1 import mysql from 'mysql';
2 import dotenv from 'dotenv';
3
4 dotenv.config();
5
6 // 需要创建的数据库名称
7 const database = process.env.MYSQL_DATABASE;
8 // 数据库连接配置
9 const config = {
10   host: process.env.MYSQL_HOST,
11   port: parseInt(process.env.MYSQL_PORT),
12   user: process.env.MYSQL_USER,
```



```
13   password: process.env.MYSQL_PASSWORD
14 };
15
16 const pool = mysql.createPool(config);
17
```



如果在开发过程中，出现 **TypeScript** 类型声明报错，可以重新定义环境变量的类型声明。

 复制代码

```
1 declare module 'process' {
2   global {
3     // eslint-disable-next-line no-var
4     var process: NodeJS.Process;
5     namespace NodeJS {
6       interface ProcessEnv extends Dict<string> {
7         NODE_ENV: 'development' | 'production';
8         MYSQL_HOST: string;
9         MYSQL_PORT: string;
10        MYSQL_USER: string;
11        MYSQL_PASSWORD: string;
12        MYSQL_DATABASE: string;
13      }
14    }
15  }
16 }
```

如果你在企业有用 **Docker**，或者购买其它云服务厂商的 **MySQL** 服务，可以把数据库的账号密码设置在 **Docker** 等容器的环境变量里。

这里要再次说明一下，生产环境中数据库的账号不能用 **root** 权限的账号，容易带来权限安全问题，最好用 **MySQL** 创建一个只有某个数据库的“增改查数据”权限的账号。

总结

通过今天的学习，相信你对平台的全栈项目搭建有了新的认知，特别是 **Node.js** 的服务拆分和 **MySQL** 数据库的选择和准备。全栈项目的搭建是没有标准答案的，都是根据实际项目情况因地制宜做方案设计的。

一个全栈项目的设计主要有 **4** 个要点。

- 根据用户的差异，运营搭建平台拆分成面向的员工后台服务，以及面向客户的前台服务。
- 后台 Web 服务主要提供管理能力，给企业内部员工搭建页面。
- 前台 Web 服务主要提供渲染能力，支持 CSR 和 SSR 给用户可以预览页面，以及搜索引擎抓取页面实现 SEO。
- 前后台服务的联动主要是共享静态资源和共享数据库。



平台拆分成前台和后台两个服务，主要考虑到“安全”、“稳定”和“便于维护”的因素。“安全”是指服务各自独立，不容易产生权限问题；“稳定”是指服务独立部署，服务崩溃不会互相干扰；而“便于维护”是指前台后台代码都可以独立升级，升级不会互相影响。

思考题

为什么不能用读写静态文件的方式来代替数据库使用？

期待在留言区看到你的思考。除了掌握今天课程中全栈项目设计思路外，也希望你能举一反三设计出更优雅的全栈项目。我们下节课再见。

[完整的代码在这里](#)

分享给需要的人，Ta购买本课程，你将得 18 元

生成海报并分享

赞 2 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 18 | Node.js服务端渲染页面：客户端渲染和服务端渲染有何区别？

下一篇 20 | 数据库方案设计：如何设计运营搭建平台的数据库？

精选留言

写留言

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。

