# 项目准备开发

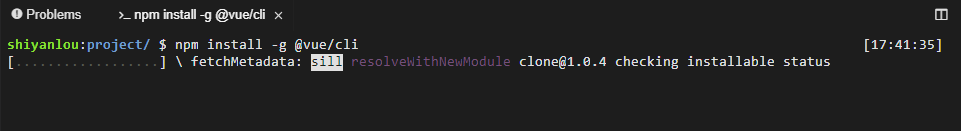
## Vue CLI 创建项目

使用 Vue CLI 创建项目，首先需要安装 @vue/cli，在终端中输入如下命令：

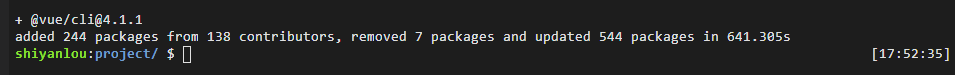
# 安装 `@vue/cli`

npm install -g @vue/cli

安装的时间较长，大约有 10 分钟左右，请耐心等候。



当出现下面这样的情况时，就完成了 @vue/cli 的全局安装。



然后，我们可以通过以下命令来检查安装的版本是否正确：

vue --version

这是在写本课程时所查询的版本：

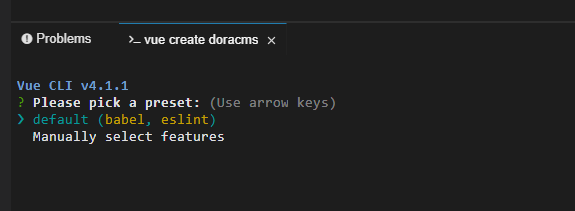


接下来，就可以使用 @vue/cli 搭建 Vue 项目了。运行 vue create 命令即可创建一个新的项目，本次项目名为：doracms。

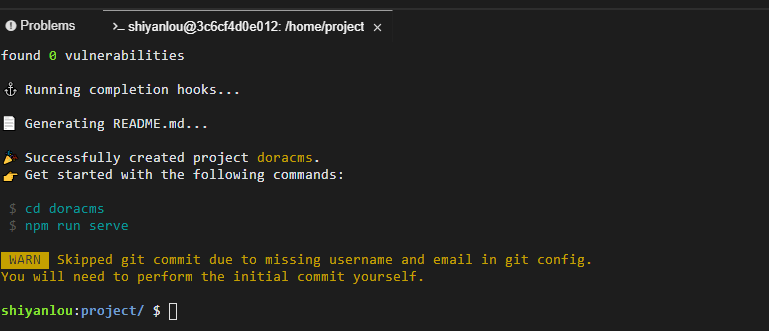
# 创建名为 doracms 的项目

vue create doracms

你会被提示选择一个 preset，可以选择默认的 default(babel, eslint)，直接回车就好。当然，后续熟悉 Vue 使用之后，你也可以自己手动进行配置。



选择 preset 之后，@vue/cli 就会自动开始创建一个名为 doracms 的 Vue 项目了。这里创建的时间较长，大约 15 分钟左右，请耐心等待。

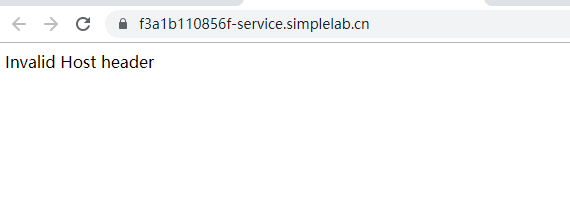


此时，我们的项目 doracms 就已经是创建成功了

## 运行 Vue 项目

项目创建之后，通过 cd doracms 进入项目目录中，再输入 npm run serve 即可运行项目。当加载完成之后，点击实验楼环境右侧工具栏中的「Web 服务」按钮，即可在浏览器中打开项目。

打开项目之后大家会出现如下界面：



出现这样的界面是因为实验楼的环境比较特殊，需要修改默认配置后才能通过外网访问项目。接下来，在根目录 /doracms 下创建 vue.config.js 配置文件，并写入以下代码。

// 获取用户环境中的 HOST

const HOST = process.env.HOST;

module.exports = {

// 部署应用包时的基本 URL

publicPath: './',

// 不为生产环境构建生成 source map

productionSourceMap: false,

// 设置开发服务器选项

devServer: {

// 指定要使用的主机

host: HOST || '0.0.0.0',

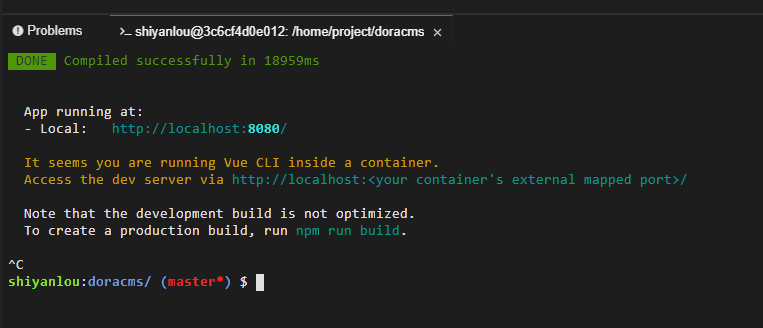
// 设置为 true 绕过主机检测

disableHostCheck: true,

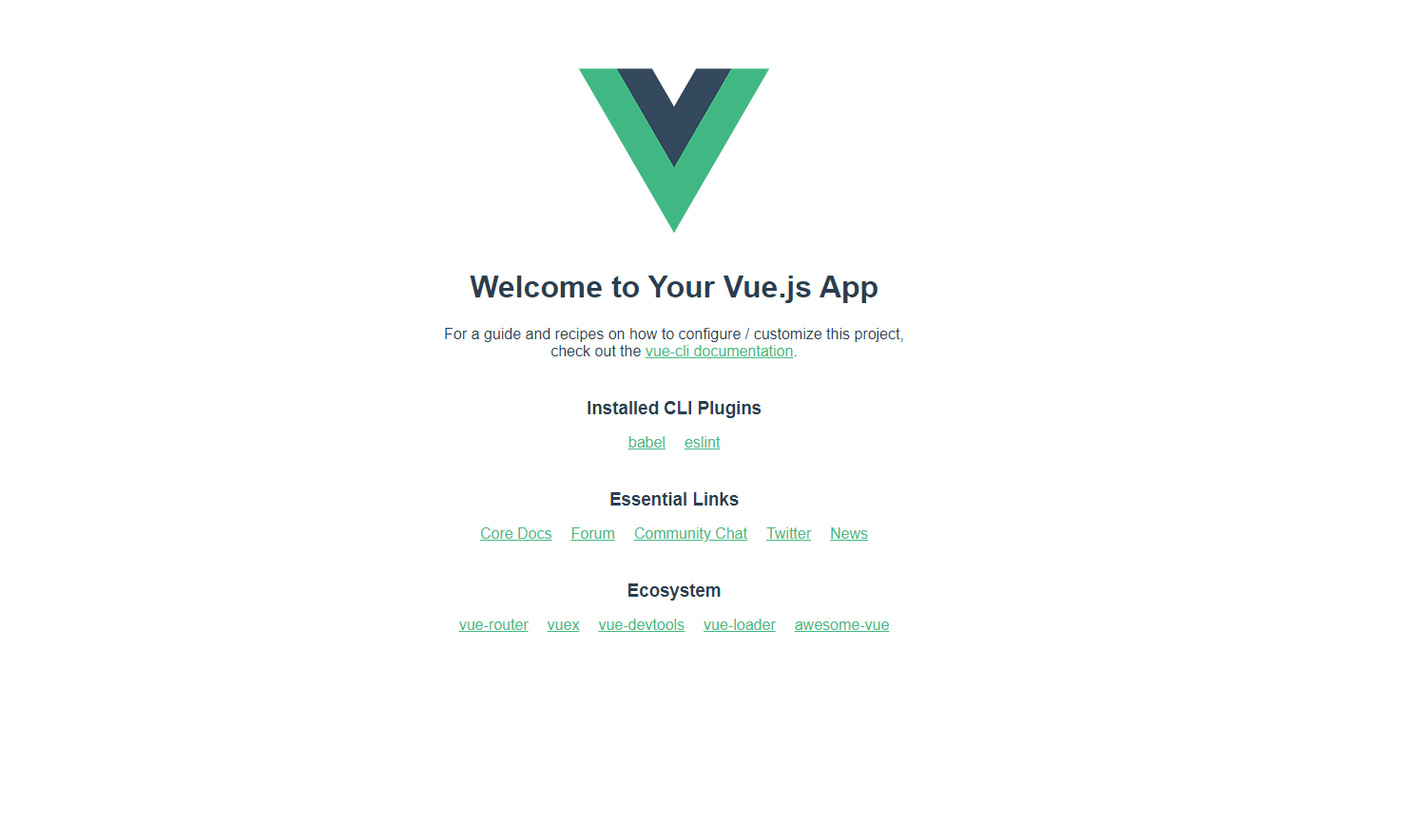
},

};

实验楼的环境是包裹在容器当中的，所以我们需要让我们的环境与外部浏览器进行接通。而上述代码就是为了跳过 host 检测，从而与外部浏览器进行正常连接。配置好 vue.config.js 文件之后，我们还需要通过在终端中使用 Ctrl + C 来中断当前项目的启动状态。



之后，再次在终端中输入 npm run serve 并回车。等到加载完成之后，重新点击 Web 服务，你应该可以成功看到初始化的页面。



至此，我们就已经学会了使用 Vue CLI 创建一个新项目。上面的操作步骤稍微有些复杂，如果你运行项目失败，需要再仔细阅读一遍内容，避免遗漏步骤。

## 项目插件安装

本次项目要想实现所需功能，就需要安装好各种插件，Vue 的插件配置默认在 package.json 文件下方。接下来，我们在终端中使用 Ctrl + C 来中断当前项目的启动状态，再开始对插件进行安装。

大家可以手动将原来生成的 package.json 文件删除之后，在终端写入以下代码下载新的 package.json 文件。

# 下载新的 package.json 文件

wget https://labfile.oss.aliyuncs.com/courses/1473/package.json

# 安装项目插件

npm install

安装项目插件的时间较长，大约 5 分钟左右，请耐心等待。

当然，也可以手动下载插件进行安装，下面是本次实验中所需要使用的插件：

* npm install --save element-ui：安装 ElementUI 库，用于对项目进行 UI 设计。
* npm install --save vue-router：安装 Vue 的路由插件，用于路由之间的跳转。
* npm install --save mongoose：安装 MongoDB 数据库，用于存放项目所需数据。
* npm install --save axios：安装 axios 库，用于发送数据请求，进行前后端交互。
* npm install express：安装 express 中间件，为 Nodejs 和 Web 之间提供辅助。
* npm install bcrypt：安装 bcrypt 加密，用于对用户密码进行加密处理。
* npm install jsonwebtoken：安装 jsonwebtoken 插件，用于生成 token。
* npm install nodemon：安装 nodemon 插件，用于实现 Node.js 热加载。

由于上方已经通过 package.json 安装了插件，实验不必手动重复安装。至此，我们已经完成了本课程所涉及项目的全部初始化过程。

## 项目整体认知

本次项目主要是使用 Vue、ElementUI、Node.js、以及 MongoDB 搭建文档内容管理系统的前台部分。主要功能有展示前台数据，以及管理员与普通用户登录时不同的处理过程。

在正式开始构建项目之前，我们希望你对项目有一个整体认识，熟悉其文件结构。这将对后续的实际开发过程很有帮助。

#### 项目文件结构

首先，我们下载完整的项目源码。

# 删除当前环境中已有的文件夹

rm -rf doracms

# 下载完整的项目源码压缩包

wget https://labfile.oss.aliyuncs.com/courses/1473/doracms.zip

# 解压资源

unzip doracms.zip

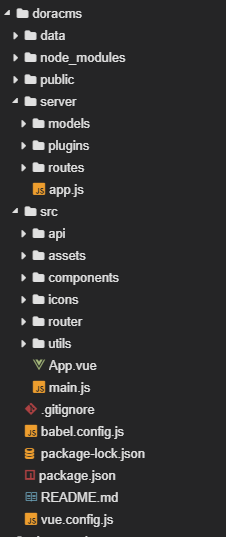
# 进入项目目录

cd doracms

# 安装项目依赖，时间较长，大约 5-10 分钟，请耐心等候。

npm install

安装过程中，你可以通过 WebIDE 左侧栏看到完整项目的结构。



其中，在 src 目录中的为前端代码，在 server 目录中的则是服务端代码。在 src 目录下创建 api、router、utils 三个文件。api 文件夹用于前后端交互的路径配置，router 文件夹用于 Vue 项目中各页面的路由配置，utils 文件夹用于存放公共 js 代码。在 src/components 目录下创建 common 以及 page 两个文件夹，分别用于存放公共组件以及页面组件。

为了让完整项目能够在实验楼线上环境中启动，与上文一样，我们需要对 vue.config.js 配置文件进行简单修改。请将 vue.config.js 配置文件中的数据替换为：

// 获取用户环境中的 HOST

const HOST = process.env.HOST;

// 导入路由

const path = require('path');

function resolve(dir) {

return path.join(\_\_dirname, dir);

}

module.exports = {

// 部署应用包时的基本 URL

publicPath: './',

// 生成的生产环境构建文件的目录

outputDir: 'dist',

// 在开发环境下通过 eslint-loader 在每次保存时 lint 代码

lintOnSave: true,

// 不为生产环境构建生成 source map

productionSourceMap: false,

// 设置开发服务器选项

devServer: {

// 指定要使用的主机

host: HOST || '0.0.0.0',

// 自动打开默认浏览器

open: true,

// 设置为 true 绕过主机检测

disableHostCheck: true,

},

};

等待上方 npm install 安装命令执行完成之后，就可以尝试运行项目了。完整的项目运行需要使用到三个终端。首先，选中 doracms 文件夹，然后点击右键选择 open in Terminal 即可开启一个新终端，同时会自动进入 home/project/doracms 目录下。

接着分别在 3 个终端中键入以下命令：

# 第 1 个终端，开启数据库连接

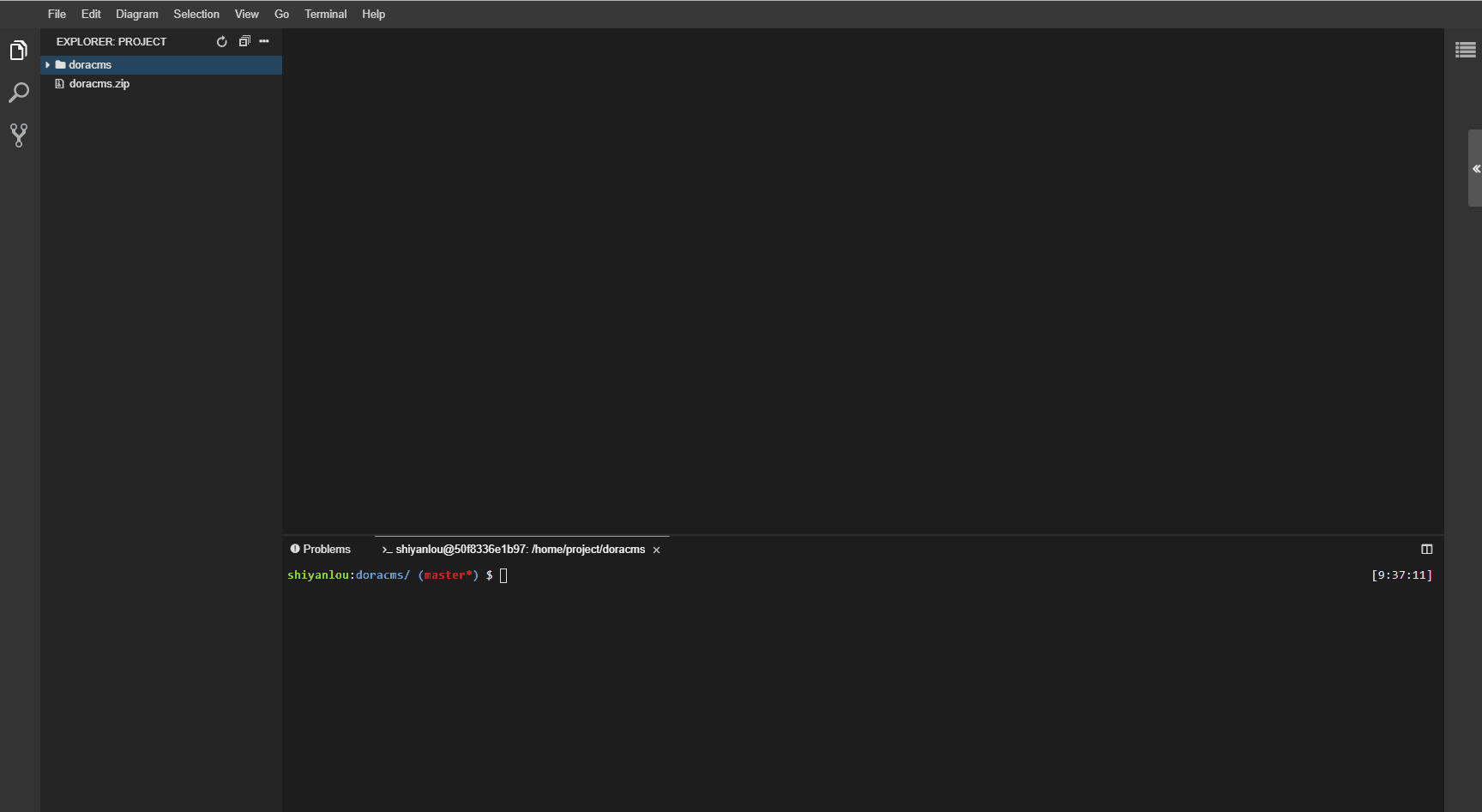
sudo mongod --dbpath /home/project/doracms/data/db --logpath /home/project/doracms/data/log/mongodb/mongod.log

# 第 2 个终端，运行服务器

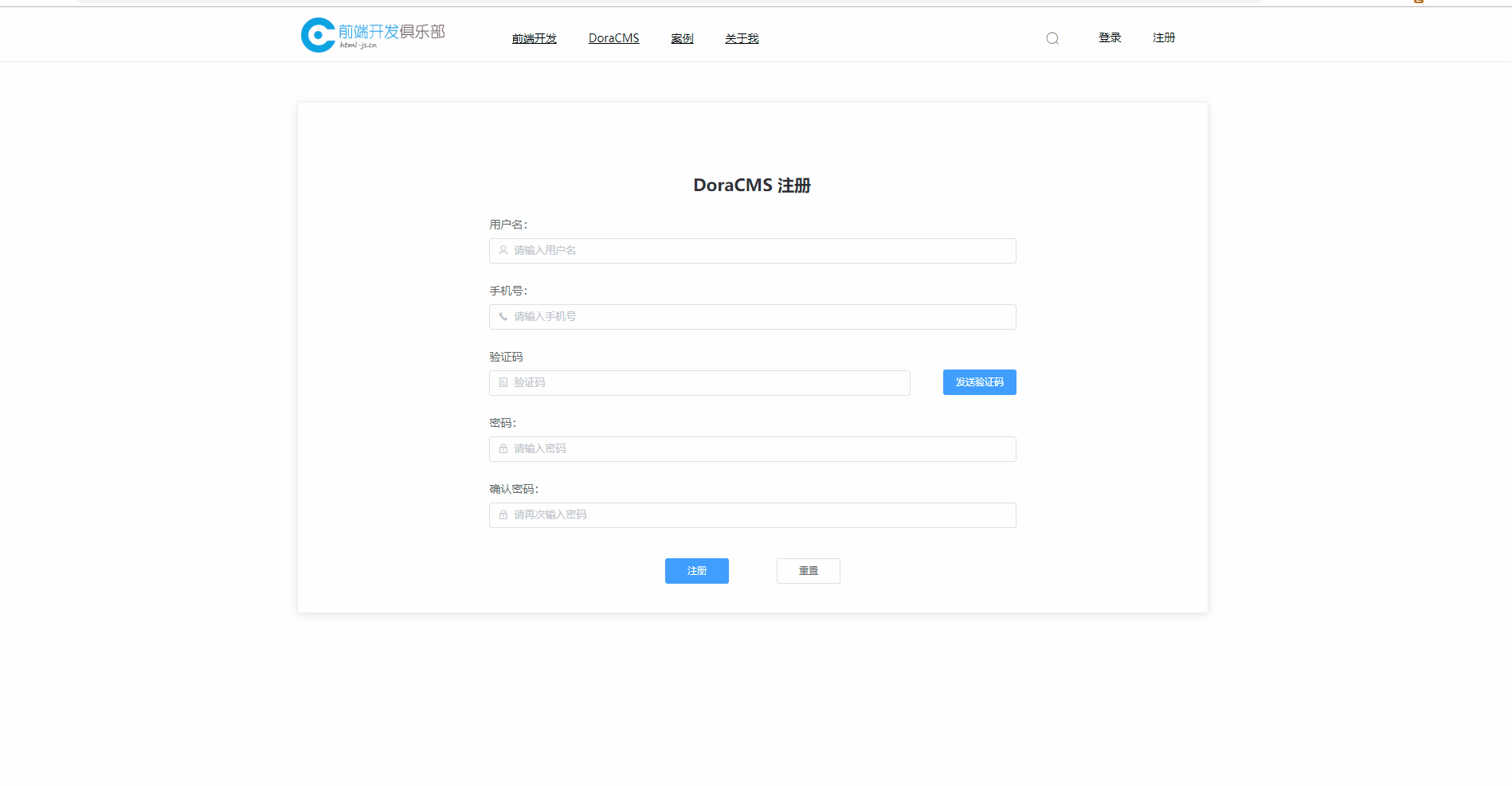
npm run dev

# 第 3 个终端，运行 Vue 项目

npm run serve



操作完成之后，点击在线环境右侧「Web 服务」，即可看到完整的项目页面。



你可以随意点击项目中的选项，简单熟悉项目，这将对后续的开发过程很有帮助。

# 服务端搭建与配置

## 实验介绍

本课程是基于 Vue.js 和 Node.js 进行开发的，所以我们需要完成后端服务器的搭建工作。在本次实验中，我们将对本次项目的服务端进行搭建，并且完成一些项目中所需要的配置。

#### 知识点

* Node.js 搭建服务器
* 客户端与服务端跨域连接
* 配置服务端

## 恢复环境

为了避免每次开始实验之前重复的安装过程，我们提供了上一个实验的源码。你可以通过以下步骤，快速恢复课程开发所需环境：

# 下载资源

wget https://labfile.oss.aliyuncs.com/courses/1473/doracms1.zip

# 解压资源

unzip doracms1.zip

# 进入项目目录

cd doracms

# 安装项目依赖

npm install

由于项目安装时间较长，你也可以使用会员支持的「保存环境」功能，直接保存上次实验结束时的环境避免重新安装。

## Node.js 搭建服务器

Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境，能够在服务端运行 JavaScript。所以，我们常常使用它来作为我们前端开发人员涉足后台的利器，我们可以使用 Node.js 来搭建一个简单的服务器。

#### 使用 HTTP 模块创建服务器

在上一次实验中，我们已经搭建好了一个 Vue 项目，接下来，我们便开始搭建一个服务端。我们在项目根目录 /doracms 下创建 server 文件夹（如果存在则不用重复创建），在其中创建一个 app.js 文件，并写入以下代码。

// 加载 http 核心模块

var http = require('http');

// 使用 http.createServer() 方法创建一个 web 服务器，返回一个 Server 实例

var server = http.createServer();

// 使用服务器对数据提供服务：发送请求，接受请求，处理请求 ，请求反馈（发送响应）

// 注册 request 请求事件，当客户端请求过来，就会自动触发服务器的 request 请求事件

// 然后执行第二个参数：回调处理函数

server.on('request', function () {

console.log('收到客户端的请求了');

});

// 绑定端口号，启动服务器

server.listen(8081, function () {

console.log('服务器启动成功了，可以通过 http://localhost:3000/ 来访问');

});

在 Node.js 中提供了一个核心模块：http，这个模块的功能就是帮助你创建一个服务器。大家可以 cd server 进入 server 目录，在终端中执行命令 node app.js，等待加载完毕就可以在终端中看见服务启动成功的提示，这就表示我们的服务器已经创建成功了。

## 使用 Express 创建服务器

我们可以使用 HTTP 核心模块来进行服务器的搭建，但是想要项目更好地运行，就需要对项目进行改进。我们引入了 Express，Express 是一个高度包容、快速而又极其简单的 Node.js Web 框架，它为 Web 和移动应用程序提供了一组强大的功能。所以我们还是选择使用 Express 创建服务器。

在终端中使用 Ctrl + C，并将 app.js 当中的内容进行注释或删除，我们开始使用 Express 来进行服务器的创建。

// 1. 引入 express 框架

var express = require('express');

// 2. 创建一个 express 实例

var app = express();

// 3. 服务器对客户端传来的请求进行响应

app.get('/', function (req, res) {

res.send('Hello World!');

});

// 4. 绑定端口号，启动服务器

app.listen(8081, function () {

console.log('Server is running in http://localhost:8081!');

});

同样的 cd server 进入 server 目录，在终端中执行命令 node app.js，等待加载完毕就可以在终端中看见「Server is running in ……」的提示。

## 客户端与服务端进行跨域连接

服务器已经搭建好了，那我们便开始进行前后端的交互工作，实验楼的默认输出端口是 8080。也就是说，当我们开始运行 Vue 项目时，它只能通过 8080 端口进行访问。而服务端默认的端口是 8081，这就会造成一个不同域名之间请求的问题，也就是跨域问题。

#### Vue 解决跨域

Vue 项目中有一个配置文件 vue.config.js，在这个文件中有一个 webpack 选项 devServer，修改该选项即可解决跨域问题。在 vue.config.js 文件中修改如下：

devServer: {

// 指定要使用的主机

host: HOST || '0.0.0.0',

// 自动打开默认浏览器

open: true,

// 绕过主机检测

disableHostCheck:true,

proxy: {

// 将请求到 /api 的请求理到 http://localhost:8081

'/api': {

// 接口的域名

target: 'http://localhost:8081',

// 允许代理 websockets

ws: true,

// 允许接口跨域

changeOrigin: true,

// 重写地址将前缀转为 '/'

pathRewrite : {

'^/api' : ''

}

}

}

}

对比上一个实验中的代码可以发现，我们添加了一个 Proxy 属性，这个属性就可以用来配置我们的跨域问题。在这里，我们将 http://localhost:8081，代理为 /api。使我们可以通过访问不同的 URL 的方式来响应服务器，同时设置 changeOrigin 为 true 来允许跨域解决。这也是我们在 Vue 项目中推荐的使用方式。

## 前后端连接

解决了跨域问题，接下来便需要开始进行连接操作。我们在根目录下的 src 目录下创建 api 文件夹，并在文件夹中创建 index.js，在其中写入以下代码。

// 1. 引入 axios 库

import axios from 'axios';

// 2. 创建自定义 axios 实例

const http = axios.create({

baseURL: '/api',

});

// 3. 导出 axios 实例

export default http;

axios 是我们用来进行前后端通信的插件工具，在这里我们创建了一个 axios 实例，并设置其对应的请求路径。

接下来在 main.js 中添加以下代码。

// 4. 导入 axios 实例。

import http from './api';

// 5. 将 axios 实例导入 Vue 实例的原型当中

// 这一步需要在 import Vue from 'vue' 以及 import http from './api' 代码下面书写

Vue.prototype.$http = http;

在入口文件 main.js 中进行全局注册之后，我们就可以在不同组件中使用 axios 来进行请求数据了。

最后在需要的组件中进行使用：

// 6. 发送请求

this.$http.get('url', data).then();

this.$http.post('url', data).then();

这就是我们 axios 的使用方法，我们现在还没有开始编写组件所以不着急使用。这里只是让大家了解 axios 的使用方法。

## 配置服务端

接下来，我们需要对服务端进行一定的配置。在 server 目录下创建 routes 文件夹。用于存放路由配置。我们对路由文件进行配置。在 routes 文件夹中，创建 index.js 文件，后续的对于各个请求的处理以及逻辑功能的实现我们都会在这个文件中进行设置。填入以下代码：

// 将路由配置文件当做一个函数进行导出

// 在 main.js 中添加代码：require('./routes')(app) 进行导入使用

module.exports = (app) => {

// 导入 express 框架

const express = require('express');

// 对客户端发起的请求进行响应处理

app.post('url', async (req, res) => {

// 处理逻辑

});

};

在 app.js 中添加代码，在服务端的入口文件中进行调用。以下代码需要写在原文件中 var app = express() 行下方：

// 对 post 传递的数据进行处理

app.use(express.json());

// 引入路由文件

require('./routes')(app);

接下来，我们还需要配置服务端的启动方式，使用 nodemon 开启。修改 package.json 文件的 scripts 属性为：

"scripts":{

"serve": "vue-cli-service serve",

// 添加服务器的开启方法，使用 nodemon 开启，添加入文件之后，本注释请删除

"dev": "nodemon server/app.js",

"build": "vue-cli-service build",

"lint": "vue-cli-service lint"

}

至此，我们服务端的基本配置就已经完成了。我们设置好了服务端的了路由处理文件，还完成了前后端连接的 axios 的配置，对于跨域问题也进行了解决。