

**Vue-Element 前端应用**  
**环境安装和产品部署**  
**说明书**  
V2.0



# 目 录

<b>1.</b>	<b>引言 .....</b>	<b>3</b>
1.1.	背景.....	3
1.2.	编写目的 .....	3
1.3.	参考资料 .....	3
1.4.	术语与缩写.....	3
<b>2.</b>	<b>基础知识介绍 .....</b>	<b>4</b>
2.1.	VUE 框架简介 .....	4
2.2.	ELEMENT 介绍.....	5
<b>3.</b>	<b>前端开发所需的软件环境.....</b>	<b>5</b>
3.1.	VS CODE 的安装.....	6
3.2.	安装 NODE 开发环境.....	8
3.3.	VUE 脚手架的安装.....	9
3.4.	VUE DEVTOOL CHROME 插件的安装.....	9
3.5.	开发环境的配置使用.....	10
3.6.	编译运行及发布项目代码 .....	14
<b>4.</b>	<b>后端开发框架开发所需环境.....</b>	<b>15</b>
4.1.	常规开发工具的安装.....	15
4.1.1.	Redis 的安装使用 .....	16
4.2.	MYSQL 数据库及管理工具.....	19
4.3.	WINFORM 前端开发组件 .....	21
<b>5.</b>	<b>产品部署说明 .....</b>	<b>21</b>
5.1.	部署基于.NETCORE 的开发框架后台 API 服务端 .....	21

5.1.1.	安装.net core 环境 .....	21
5.1.2.	发布.net core 项目 .....	23
5.1.3.	在服务器中设置 IIS .....	25
5.2.	使用 NGINX 部署 VUE+ELEMENT 前端应用.....	28
5.3.	利用云服务提供商的免费证书, 在服务器上发布 HTTPS 前端应用和 WEBAPI 的应用.....	32
5.3.1.	申请免费证书.....	32
5.3.2.	发布基于 IIS 的 Web API 的 https 应用接口.....	35
5.3.3.	发布 Nginx 的前端应用.....	38
5.4.	开发框架使用 MySQL 数据库 .....	40
5.4.1.	使用 MySql 数据库开发框架后端的代码修改.....	41
5.4.2.	基于 SQLServer 创建 MySql 数据库的架构和数据.....	42

## 1. 引言

### 1.1. 背景

Vue + Element 的前端开发方式, 和以前的开发方式有很大的不同, 不再依赖于 VS (Visual Studio) 开发 IDE, 而是采用微软另外一套轻型的开发 IDE--VS Code, VS Code 是跨平台的应用, 在 Mac、Windows 平台均可使用, 这样开发前端不用依赖于 Windows 环境了。VS Code 虽然是轻型开发工具, 不过功能也是非常强大的, 而且开发环境可以在 Windows 系统, 也可以在 Mac 系统等, 实现了多平台的开发环境。

Vue + Element 的前端开发方式, 主要就是利用 node.js 的开发环境, 使用各种前端框架或者组件, 实现对前端视图页面和 JS 代码的统一整合。这里完全没有 C# 代码, 也没有 Java 代码, 而是通过 JS 前端框架和后端 API 平台实现数据交互/或者于本地 Mock 服务交互。

### 1.2. 编写目的

本篇内容主要介绍如何准备 Vue + Element 的前端开发环境, 以及项目开发完成后进行产品部署, 所需要的发布产品步骤。

### 1.3. 参考资料

序号	名称	版本/日期	来源
1	《伍华聪的博客》( <a href="http://wuhuacong.cnblogs.com">http://wuhuacong.cnblogs.com</a> )		博客园
1	《》		内部

### 1.4. 术语与缩写

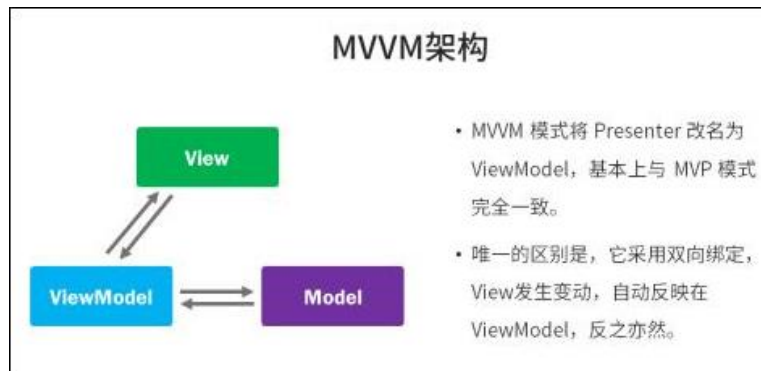
- 1 在本文件中出现的“系统”、“框架”一词, 除非特别说明, 均适用于《SqlSugar 开发框架》。

2 当前基于.netcore的版本为6.0,如无特殊说明,均指该版本。

## 2. 基础知识介绍

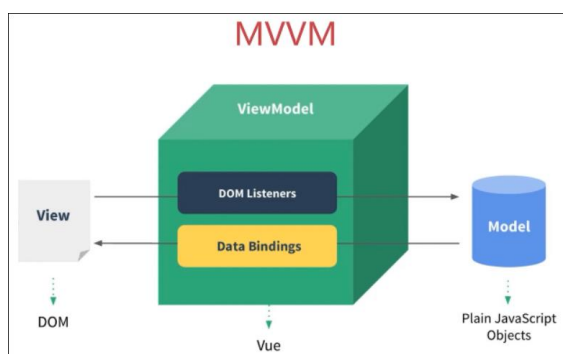
### 2.1. Vue 框架简介

Vue 是一套构建用户界面的框架, 开发只需要关注视图层, 它不仅易于上手, 还便于与第三方库或既有项目的整合; 另一方面, 当与现代化的工具链以及各种支持类库结合使用时, Vue 也完全能够为复杂的单页应用提供驱动。是基于 MVVM (Model-View-ViewModel) 设计思想。提供 MVVM 数据双向绑定的库, 专注于 UI 层面。



View 就是 DOM 层, ViewModel 就是通过 new Vue() 的实例对象, Model 是原生 js。开发者修改了 DOM, ViewModel 对修改的行为进行监听, 监听到了后去更改 Model 层的数据, 然后再通过 ViewModel 去改变 View, 从而达到自动同步。

Vue 核心思想, 包括数据驱动、组件化等方面。



#### 1、数据驱动

数据驱动（数据双向绑定），在Vue中，Directives对view进行了封装，当model中的数据发生变化时，Vue就会通过Directives指令去修改DOM，同时也通过DOM Listener

实现对视图 view 的监听，当 DOM 改变时，就会被监听到，实现 model 的改变，从而实现数据的双向绑定。

## 2、组件化

组件化就是实现了扩展 HTML 元素，封装可用的代码。

- 1) 页面上每个独立的可视/可交互区域视为一个组件。
- 2) 页面不过是组件的容器，组件可以嵌套自由组合形成完整的页面

本框架 Vue 前端基于 Vue3+TypeScript 的技术路线，并主要采用 Vue3.2 语法进行编写 Vue 页面内容，学习 Vue-Element 前端框架前，需要逐步了解 Vue3 的一些基本知识。详细可以参考官网 <https://cn.vuejs.org/> 进行专题学习。

## 2.2. Element 介绍

Element，是饿了么公司开发的一套 UI 组件，一套为开发者、设计师和产品经理准备的基于 Vue 前端组件库。提供了配套设计资源，帮助你的开发快速成型。由饿了么公司前端团队开源。

Element 前端界面套件，提供了几乎所有常见的 Web 组件封装，并扩展了很多功能丰富的 UI 组件，使用这些界面组件可以极大提高 Web 界面的开发效率，同时也能够把这些基础组件封装层更高级、功能更丰富的自定义组件。

本框架 Vue-Element 前端采用当前最新的基于 Vue3.2 语法的 ElementPlus 组件库进行开发，主要特点是针对 Vue3 语法（选项式语法，TypeScript）进行了 Element 的升级，对旧版本 Element 部分组件的属性进行了调整升级，并增加了一些新的组件。

## 3. 前端开发所需的软件环境

有别于之前的 Asp.net 的开发，纯前端的开发，一般不会再采用笨重的 VS 进行前端的开发，而改用 VS Code 或者 WebStorm 等其他轻型的开发工具来进行前端代码的开发和维护，虽然是轻型开发工具，不过功能也是非常强大的，而且开发环境可以在 Windows 系统，也可以在 Mac 系统等，实现多平台的开发环境。

### 3.1. VS code 的安装

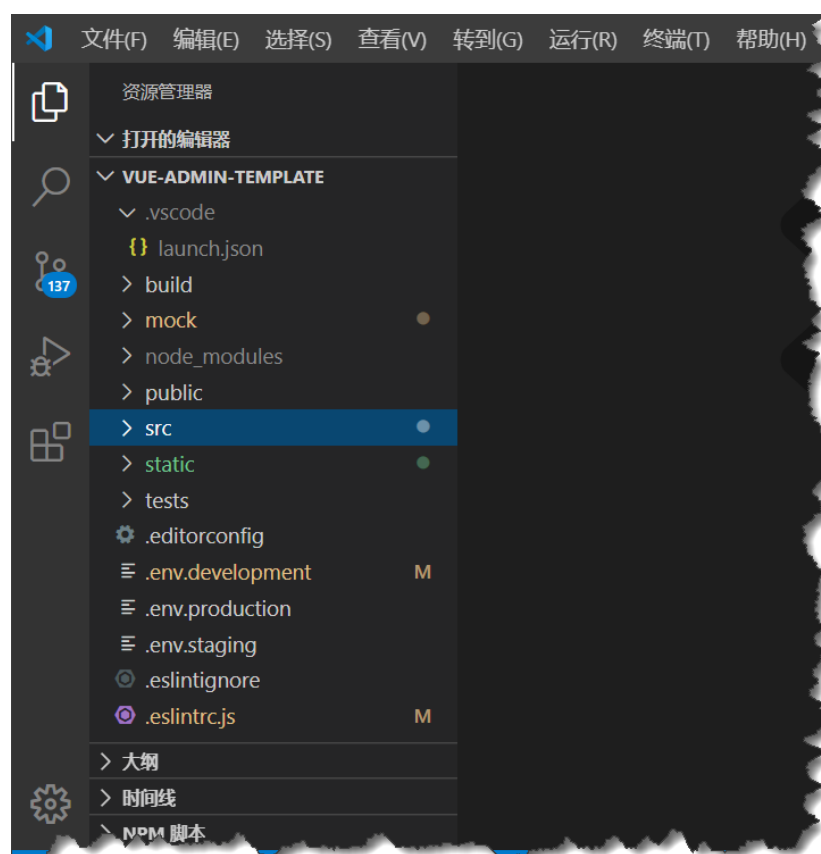
VS Code (Visual Studio Code) 是由微软研发的一款免费、开源的跨平台文本 (代码) 编辑器。几乎完美的编辑器。

官网: <https://code.visualstudio.com>

文档: <https://code.visualstudio.com/docs>

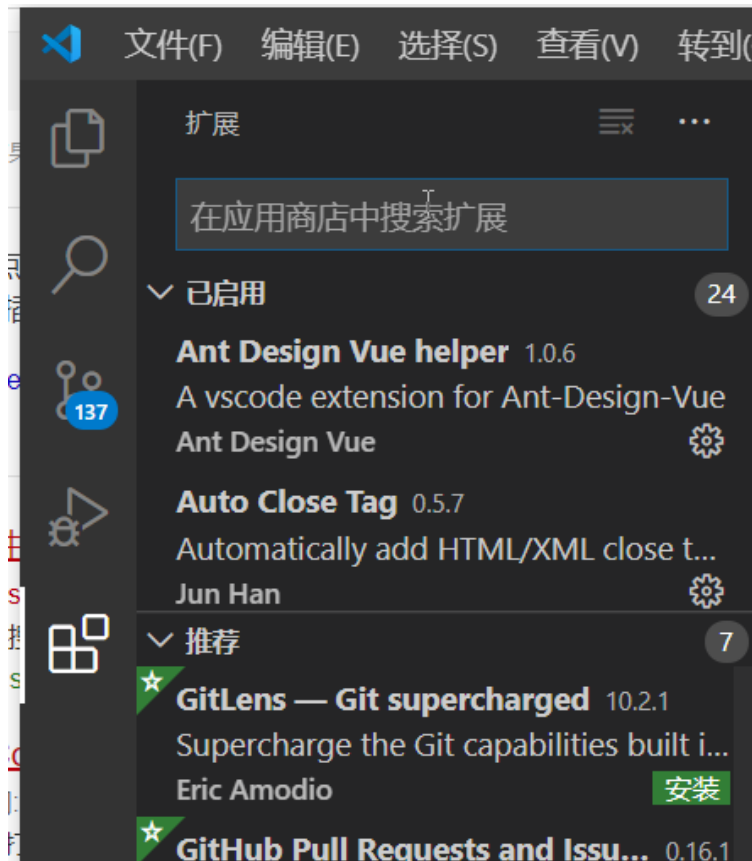
源码: <https://github.com/Microsoft/vscode>

VS Code 的界面大概如下所示, 一般安装后, 如果为英文界面, 则安装它的中文包即可。



VS Code 安装后, 我们一般还需要搜索安装一些所需要的插件辅助开发。安装插件很简单, 在搜索面板中查找到后, 直接安装即可。





一般我们需要安装这些 vs code 插件:

## Vue.volar

volar 是一个针对 vue 的 vscode 插件, 是 vetur 的升级版本, Vue 多功能集成插件, 包括: 语法高亮, 智能提示, 错误提示, 格式化, 自动补全, debugger。vscode 官方钦定 Vue 插件, Vue 开发者必备。

## ESLint

ESLint 是一个语法规则和代码风格的检查工具, 可以用来保证写出语法正确、风格统一的代码。而 VSCode 中的 ESLint 插件就直接将 ESLint 的功能集成好, 安装后即可使用, 对于代码格式与规范的细节还可以自定义, 并且一个团队可以共享同一个配置文件, 这样一个团队所有人写出的代码就可以使用同一个代码规范, 在代码签入前每个人可以完成自己的代码规范检查。

## VS Code - Debugger for Chrome 结合 Chrome 进行调试的插件

此工具简直就是前端开发必备, 将大大地改变你的开发与调试模式。

以往的前端调试, 主要是 JavaScript 调试, 你需要在 Chrome 的控制台中找到对应代码的部分, 添加上断点, 然后在 Chrome 的控制台中单步调试并在其中查看值的变化。

而在使用了 Debugger for Chrome 后, 当代码在 Chrome 中运行后, 你可以直接在 VSCode 中加上断点, 点击运行后, Chrome 中的页面继续运行, 执行到你在 VSCode 中添加的断点后, 你可以直接在 VSCode 中进行单步调试。

## Beautify

Beautify 插件可以快速格式化你的代码格式, 让你在编写代码时杂乱的代码结构瞬间变得非常规整, 代码强迫症必备, 较好的代码格式在后期维护以及他人阅读时都会有很多的便利。

## 3.2. 安装 node 开发环境

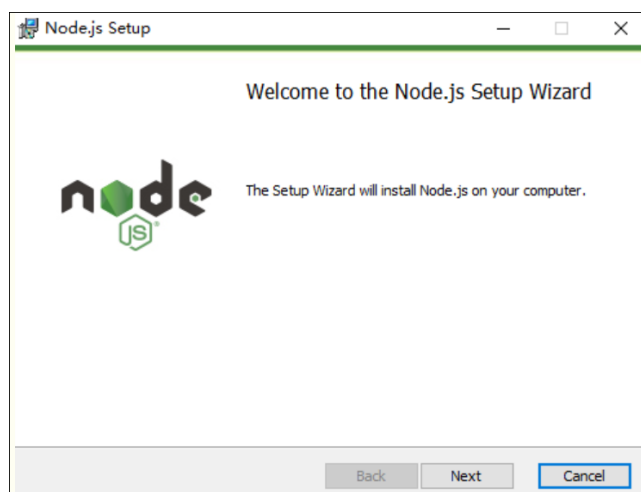
利用 VS Code 开发, 我们很多时候, 需要使用命令行 npm 进行相关模块的安装, 这些需要 node 环境的支持, 安装好 node 后, npm 也就一起安装好了。

Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境。

Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型, 使其轻量又高效。

Node.js 的包管理器 npm, 是全球最大的开源库生态系统。

node 下载: <https://nodejs.org/en/>



安装后, 我们可以通过命令行或者 VS Code 里面的 Shell 进行查看 node 和 npm 的版本号了

node -v

npm -v

### 3.3. vue 脚手架的安装

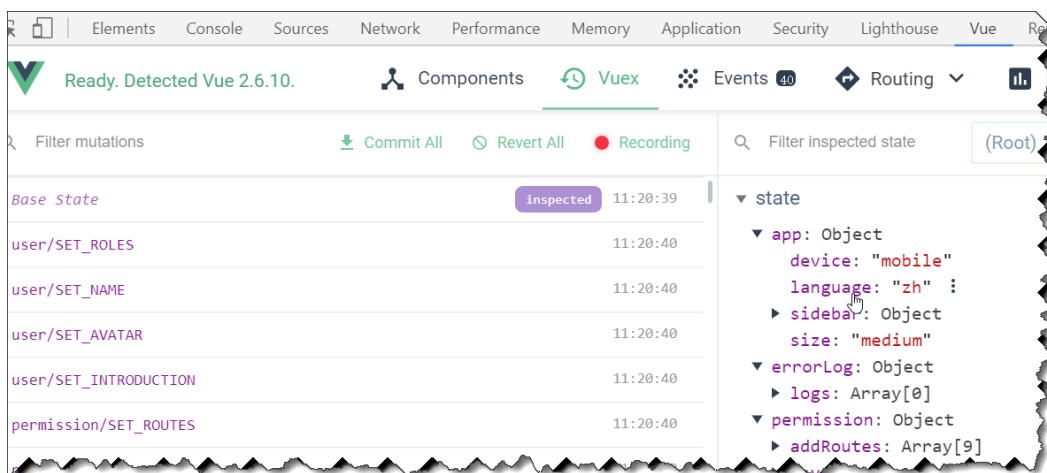
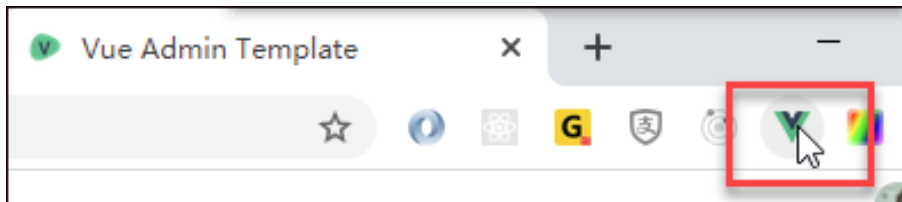
Vue (读音 /vju:/, 类似于 view) 是一套用于构建用户界面的渐进式框架。

全局安装: `npm install vue-cli -g` (全局卸载: `npm uninstall vue-cli -g`)

### 3.4. Vue DevTool Chrome 插件的安装

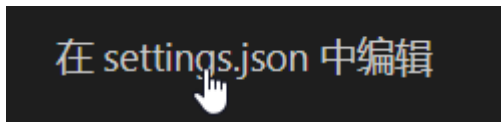
这个插件也是开发 Vue 必备的 Chrome 插件, 一般没有外网, 不能直接在 Chrome 的插件官网上进行安装, 而通过 GitHub 下载进行编译在安装又显得太过麻烦, 后来在一个网站上下载安装成功。

<https://chrome.zzzmh.cn/info?token=nhdogjmejiglipccpnnnanhbledajbpd>



### 3.5. 开发环境的配置使用

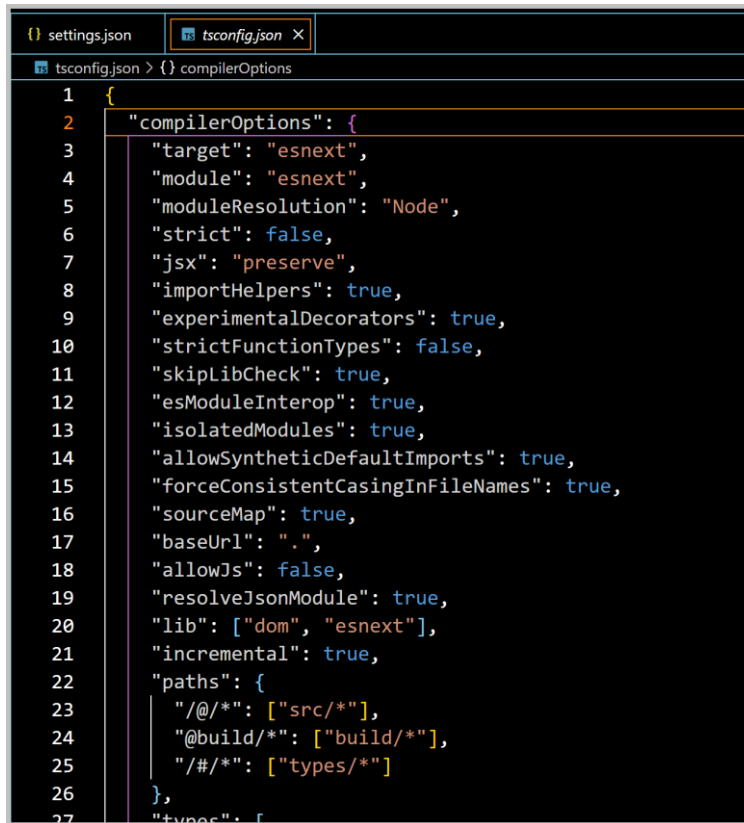
默认框架代码已经根据需要自动安装相关的插件，以及配置好相关的参数属性，如果需要自定义设置一些信息，可以在【文件】【首选项】【设置】中，然后单击 settings.json 进行编辑即可。 settings.json 是设置扩展程序或 vscode 编辑器的一些属性的地方。



以下是相关设置的代码视图：

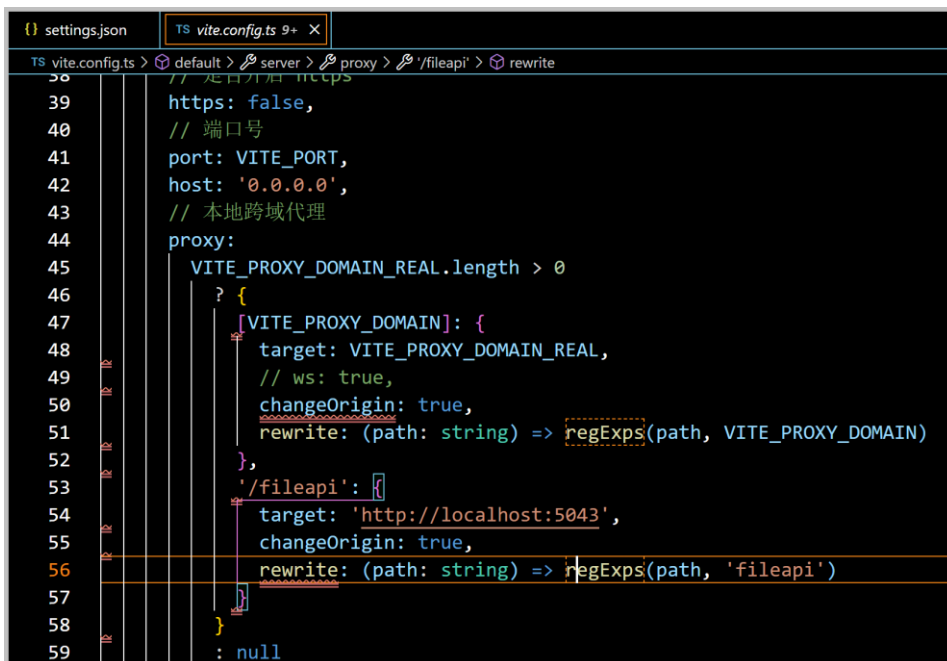
```
1  {
2    "editor.formatOnType": true,
3    "editor.formatOnSave": true,
4    "[vue]": {
5      "editor.defaultFormatter": "esbenp.prettier-vscode",
6    },
7    "editor.tabSize": 2,
8    "editor.formatOnPaste": true,
9    "editor.guides.bracketPairs": "active",
10   "files.autoSave": "afterDelay",
11   "git.confirmSync": false,
12   "workbench.startupEditor": "newUntitledFile",
13   "editor.suggestSelection": "first",
14   "editor.acceptSuggestionOnCommitCharacter": false,
15   "css.lint.propertyIgnoredDueToDisplay": "ignore",
16   "editor.quickSuggestions": {
17     "other": true,
18     "comments": true,
19     "strings": true
20   },
21   "files.associations": {
22     "editor.snippetSuggestions": "top"
23   }
24 }
```

如需修改 TypeScript 的相关设置，可以打开 tsconfig.json 内容进行修改即可，如下代码所示。



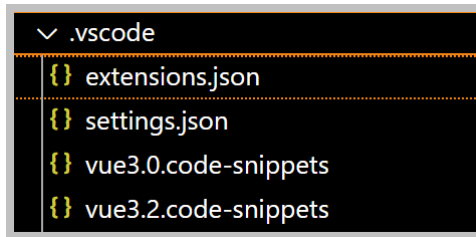
```
tsconfig.json > {} compilerOptions
1 {
2   "compilerOptions": {
3     "target": "esnext",
4     "module": "esnext",
5     "moduleResolution": "Node",
6     "strict": false,
7     "jsx": "preserve",
8     "importHelpers": true,
9     "experimentalDecorators": true,
10    "strictFunctionTypes": false,
11    "skipLibCheck": true,
12    "esModuleInterop": true,
13    "isolatedModules": true,
14    "allowSyntheticDefaultImports": true,
15    "forceConsistentCasingInFileNames": true,
16    "sourceMap": true,
17    "baseUrl": ".",
18    "allowJs": false,
19    "resolveJsonModule": true,
20    "lib": ["dom", "esnext"],
21    "incremental": true,
22    "paths": {
23      "@//*": ["src/*"],
24      "@build/*": ["build/*"],
25      "@/#/*": ["types/*"]
26    },
27    "types": [
```

如需修改 Vite 编译器的项目相关设置, 可以修改 vite.config.ts 文件内容, 如下所示。

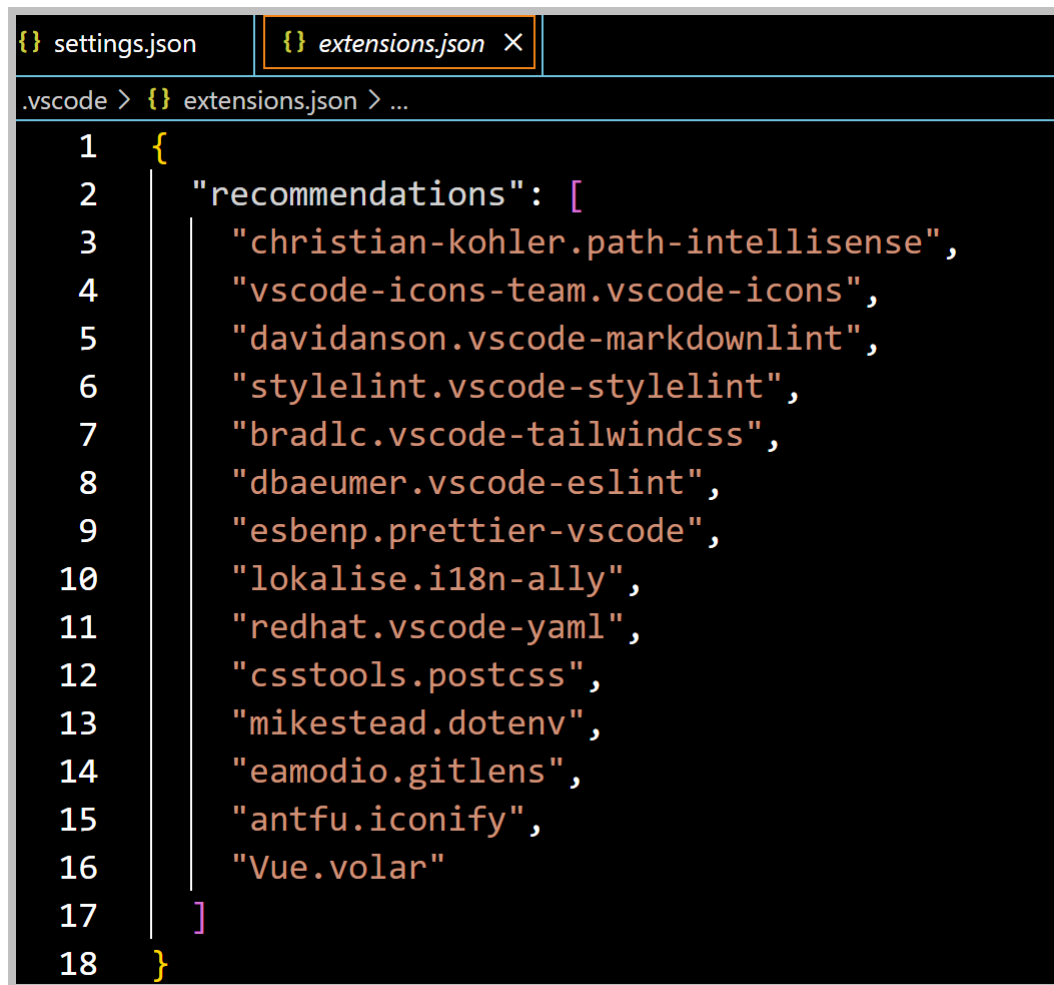


```
TS vite.config.ts > default > server > proxy > /fileapi > rewrite
39 https: false,
40 // 端口号
41 port: VITE_PORT,
42 host: '0.0.0.0',
43 // 本地跨域代理
44 proxy:
45   VITE_PROXY_DOMAIN_REAL.length > 0
46   ? {
47     [VITE_PROXY_DOMAIN]: {
48       target: VITE_PROXY_DOMAIN_REAL,
49       // ws: true,
50       changeOrigin: true,
51       rewrite: (path: string) => regExps(path, VITE_PROXY_DOMAIN)
52     },
53     '/fileapi': {
54       target: 'http://localhost:5043',
55       changeOrigin: true,
56       rewrite: (path: string) => regExps(path, 'fileapi')
57     }
58   }
59 : null
```

Vue 前端项目所需的扩展组件定义在 extensions.json 文件中



打开后可以查看相关的第三方 VSCode 扩展工具, 项目开始使用的时候, 使用 `pnpm i` 会一键安装平台推荐的 `vscode` 插件, VSCode 自动安装所需的插件。



另外, 相关的 `*.code-snippets` 的文件是定义的一些快捷模板代码, 可以通过在 `Vue` 文件中输入快捷命令进行生成相关的模板代码块的。

**注:** 为了提高开发效率, 很多时候会在 `vscode` 中配置好一些常用的代码片段, 在日常编程, 特别是在工作中写内容相似的业务代码时, 利用 `Snippet` 功能, 可以极大加快编程效率。

例如，我们在相应的文件定义中，可以修改自己的代码块生成，如下所示。

```
{ } vue3.0.code-snippets
{ } vue3.2.code-snippets
{ } vue3m.code-snippets
```

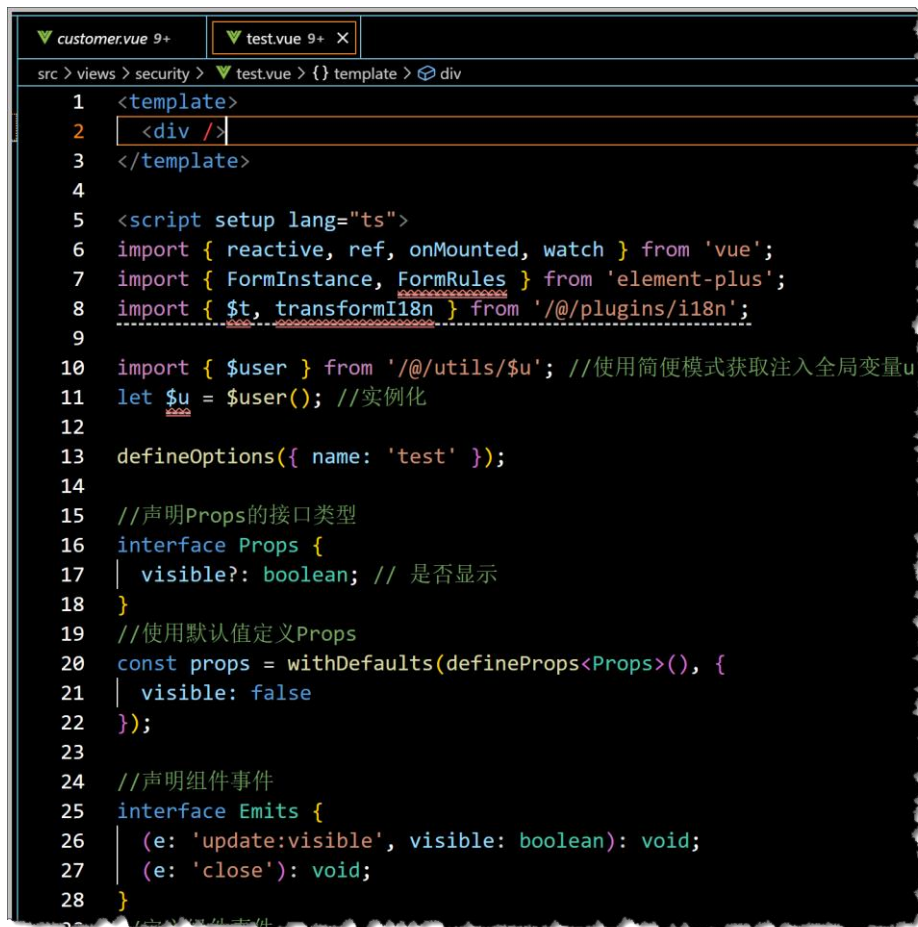
其中 vue3.0.code-snippets 、 vue3.2.code-snippets 是一些常规的代码块命令内容，而 vue3m.code-snippets 这个是本前端框架自定义的代码块命令内容，如下所示。

```
settings.json  vue3m.code-snippets X
.vscode > { } vue3m.code-snippets > ...
1 {
2   "Vue3.2+常规页面自定义快速生成模板": {
3     "prefix": "vue3m",
4     "body": [
5       "<template>",
6       "\t<div>\n",
7       "\t</div>",
8       "</template>\n",
9       "<script setup lang='ts'>",
10      "import { reactive, ref, onMounted, watch } from \"vue\";",
11      "import { FormInstance, FormRules } from \"element-plus\";",
12      "import { $$t, transformI18n } from \"/@/plugins/i18n\";\n",
13      "import { $$user } from './@/utils/$$u'; //使用简便模式获取注入全局变量$u",
14      "let $$u = $$user(); //实例化\n",
15      "defineOptions({ name: \"test\" });\n",
16      "//声明Props的接口类型",
17      "interface Props {",
18      "\tvisible?: boolean; // 是否显示",
19      "}",
20      "//使用默认值定义Props",
21      "const props = withDefaults(defineProps<Props>(), {",
22      "\tvisible: false",
```

我们只需要在 vue 文件中，输入 vue3m 的快捷命令，就会生成一套标准的代码，方便我们在其中修改、调整代码，极大提高效率。

```
customer.vue 9+  test.vue X
src > views > security > test.vue
1 vue3
2 [ ] vue3m      Vue3.2+常规页面自定义快速生成模板
   [ ] Vue3.0    Vue3.0快速生成模板
   [ ] Vue3.2+  Vue3.2+快速生成模板
```

选择对应的快捷命令，就可以生成对应的代码块了。



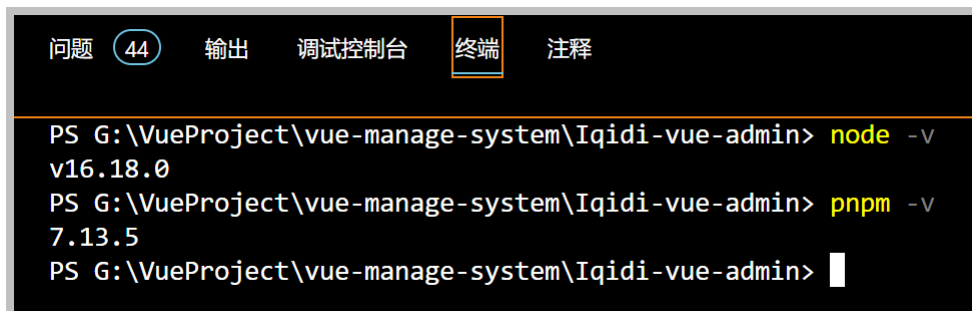
```
1 <template>
2   <div />
3 </template>
4
5 <script setup lang="ts">
6   import { reactive, ref, onMounted, watch } from 'vue';
7   import { FormInstance, FormRules } from 'element-plus';
8   import { $t, transformI18n } from '@/plugins/i18n';
9
10  import { $user } from '@/utils/$u'; //使用简便模式获取注入全局变量u
11  let $u = $user(); //实例化
12
13  defineOptions({ name: 'test' });
14
15  //声明Props的接口类型
16  interface Props {
17    | visible?: boolean; // 是否显示
18  }
19  //使用默认值定义Props
20  const props = withDefaults(defineProps<Props>(), {
21    | visible: false
22  });
23
24  //声明组件事件
25  interface Emits {
26    | (e: 'update:visible', visible: boolean): void;
27    | (e: 'close'): void;
28  }
```

### 3.6. 编译运行及发布项目代码

当前前端项目开发环境的 node 版本应不小于 16 ， pnpm 版本应不小于 6。如果您还没安装 pnpm，请执行下面命令进行安装（mac 用户遇到安装报错请在命令前加上 sudo）。

```
npm install -g pnpm
```

安装 node 环境和 pnpm 后，可以查看 node -v 和 pnpm -v 版本情况，如下所示。



```
问题 44 输出 调试控制台 终端 注释
PS G:\VueProject\vue-manage-system\Iqidi-vue-admin> node -v
v16.18.0
PS G:\VueProject\vue-manage-system\Iqidi-vue-admin> pnpm -v
7.13.5
PS G:\VueProject\vue-manage-system\Iqidi-vue-admin> |
```

前端项目代码获取后，需要在本地安装所需的项目依赖插件环境，因此可以通过一下命令进行初始化安装依赖。



**pnpm install**

然后接着在 VSCode 的项目终端窗口上启动项目即可, **启动项目**命令如下:

**pnpm dev**

安装包和移除包, 可以通过 **add** 命令和 **remove** 命令进行处理即可。

**pnpm add 包名**

**pnpm remove 包名**

如果需要带上最新版本, 加上@latest 即可。

**pnpm add axios@latest -S**

-S 后, 安装包会在 package 中的 dependencies 对象中。简称 dep。**dep** 是在生产环境中要用到的。

-D 后, 安装包会在 package 中的 devDependencies 对象中。简称 dev。**dev** 是在开发环境中要用到的。

## 4. 后端开发框架开发所需环境

后端框架是基于 C# /.NET Core6.0 开发的, 基础类库为了兼容 .net Framework 应用, 一般采用 .net Standard2.0, 而 Asp.net core Web API 部分则采用 .net core6.0 框架, 且后端框架也是使用最新版本的基础组件模块。

### 4.1. 常规开发工具的安装

常规的框架后端 Web API 是基于 .NET Core6.0 的框架应用, 开发工具一般采用 VS Studio2022 或以上版本; 数据库默认采用 SQLServer 2014, 而登录部分也采用了 Redis 缓存的数据库。

VS 开发工具可以在微软官方下载即可: <https://visualstudio.microsoft.com/zh-hans/vs/>, 默认下载社区版本即可。

SQLServer 2014 则自行获取地址进行下载。

框架后端登录部分采用了 Redis 缓存的数据库, 因此也需要 Redis 的环境支持, Redis 数据库下载地址: <https://github.com/MSOpenTech/redis/releases>, Redis 管理工具下载地址: <http://redisdesktop.com/download>。

### 4.1.1. Redis 的安装使用

Redis 是一个开源的使用 ANSI C 语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value 数据库,和 Memcached 类似,它支持存储的 value 类型相对更多,包括 string(字符串)、list(链表)、set(集合)、zset(sorted set --有序集合)和 hash (哈希类型)。在此基础上,redis 支持各种不同方式的排序。与 memcached 一样,为了保证效率,数据都是缓存在内存中。区别的是 Redis 会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件,并且在此基础上实现了 master-slave(主从)同步。

Redis 的代码遵循 ANSI-C 编写,可以在所有 POSIX 系统(如 Linux, \*BSD, Mac OS X, Solaris 等)上安装运行。而且 Redis 并不依赖任何非标准库,也没有编译参数必需添加。

#### 1) Redis 支持两种持久化方式:

(1) :snapshotting(快照)也是默认方式.(把数据做一个备份,将数据存储到文件)

(2) Append-only file(缩写 aof)的方式

快照是默认的持久化方式,这种方式是将内存中数据以快照的方式写到二进制文件中,默认的文件名称为 dump.rdb.可以通过配置设置自动做快照持久化的方式。我们可以配置 redis 在 n 秒内如果超过 m 个 key 键修改就自动做快照。

aof 方式:由于快照方式是在一定间隔时间做一次的,所以如果 Redis 意外 down 掉的话,就会丢失最后一次快照后的所有修改。aof 比快照方式有更好的持久化性,是由于在使用 aof 时,redis 会将每一个收到的写命令都通过 write 函数追加到文件中,当 redis 重启时会通过重新执行文件中保存的写命令来在内存中重建整个数据库的内容。

#### 2) Redis 数据结构

Redis 的作者 antirez 曾称其为一个数据结构服务器 (**data structures server**),这是一个非常准确的表述,Redis 的所有功能就是将数据以其固有的几种结构保存,并提供给用户操作这几种结构的接口。我们可以想象我们在各种语言中的那些固有数据类型及其操作。

Redis 目前提供四种数据类型: **string**,**list**,**set** 及 **zset**(sorted set)和 **Hash**。

- **string** 是最简单的类型,你可以理解成与 Memcached 一模一个的类型,一个 key 对应一个 value,其上支持的操作与 Memcached 的操作类似。但它的功能更丰富。

- **list** 是一个链表结构, 主要功能是 push、pop、获取一个范围的所有值等等。操作中 key 理解为链表的名字。
- **set** 是集合, 和我们数学中的集合概念相似, 对集合的操作有添加删除元素, 有对多个集合求交并差等操作。操作中 key 理解为集合的名字。
- **zset** 是 set 的一个升级版, 他在 set 的基础上增加了一个顺序属性, 这一属性在添加修改元素的时候可以指定, 每次指定后, zset 会自动重新按新的值调整顺序。可以理解为有两列的 mysql 表, 一列存 value, 一列存顺序。操作中 key 理解为 zset 的名字。
- **Hash** 数据类型允许用户用 Redis 存储对象类型, Hash 数据类型的一个重要优点是, 当你存储的数据对象只有很少几个 key 值时, 数据存储的内存消耗会很小. 更多关于 Hash 数据类型的说明请见: <http://code.google.com/p/redis/wiki/Hashes>

### 3) Redis 数据存储

Redis 的存储分为内存存储、磁盘存储和 log 文件三部分, 配置文件中有三个参数对其进行配置。

**save seconds updates**, **save** 配置, 指出在多长时间, 有多少次更新操作, 就将数据同步到数据文件。这个可以多个条件配合, 比如默认配置文件中的设置, 就设置了三个条件。

**appendonly yes/no**, **appendonly** 配置, 指出是否在每次更新操作后进行日志记录, 如果不开启, 可能会在断电时导致一段时间内的数据丢失。因为 redis 本身同步数据文件是按上面的 save 条件来同步的, 所以有的数据会在一段时间内只存在于内存中。

**appendfsync no/always/everysec**, **appendfsync** 配置, **no** 表示等操作系统进行数据缓存同步到磁盘, **always** 表示每次更新操作后手动调用 **fsync()** 将数据写到磁盘, **everysec** 表示每秒同步一次。

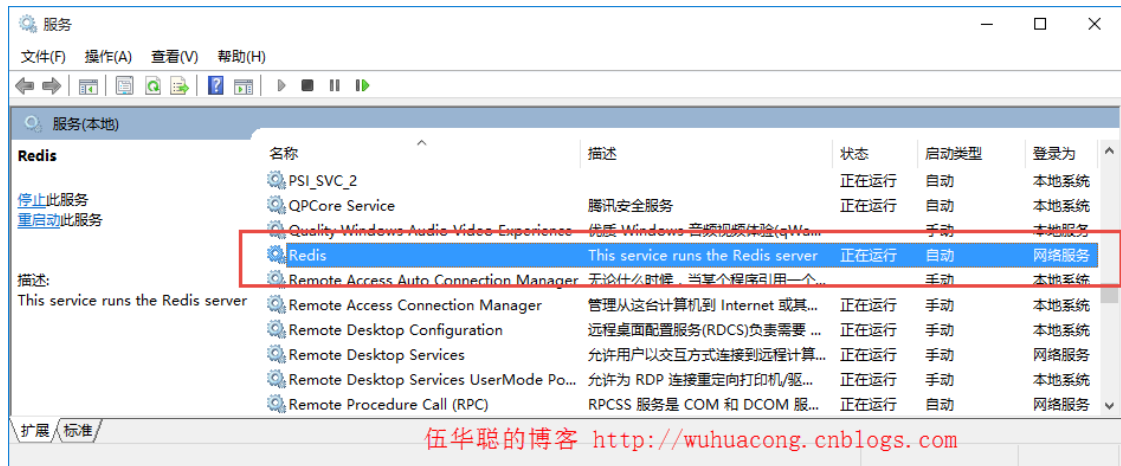
### 4) Redis 的安装

Redis 可以在不同的平台运行, 不过我主要基于 Windows 进行开发工作, 所以下面主要是基于 Windows 平台进行介绍。

Redis 可以安装以 DOS 窗口启动的, 也可以安装为 Windows 服务的, 一般为了方便, 我们更愿意把它安装为 Windows 服务, 这样可以比较方便管理。下载地址:

<https://github.com/MSOpenTech/redis/releases> 下载, 安装为 Windows 服务即可。

当前可以下载到最新的 Windows 安装版本为 3.0, 安装后作为 Windows 服务运行, 安装后可以在系统的服务里面看到 Redis 的服务在运行了, 如下图所示。



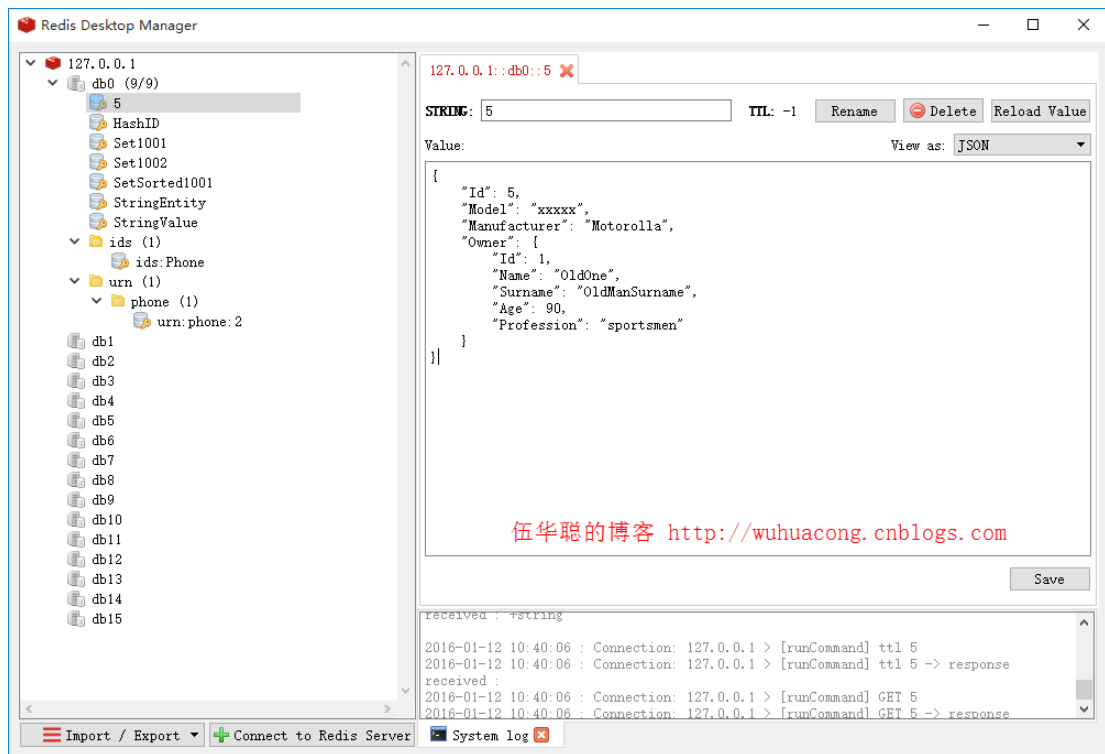
安装好 Redis 后, 还有一个 Redis 伴侣 Redis Desktop Manager 需要安装, 这样可以实时查看 Redis 缓存里面有哪些数据, 具体地址如下: <http://redisdesktop.com/download> 下载属于自己平台的版本即可

## Download Redis Desktop Manager version 0.8.3

[Release notes](#) [All releases](#)

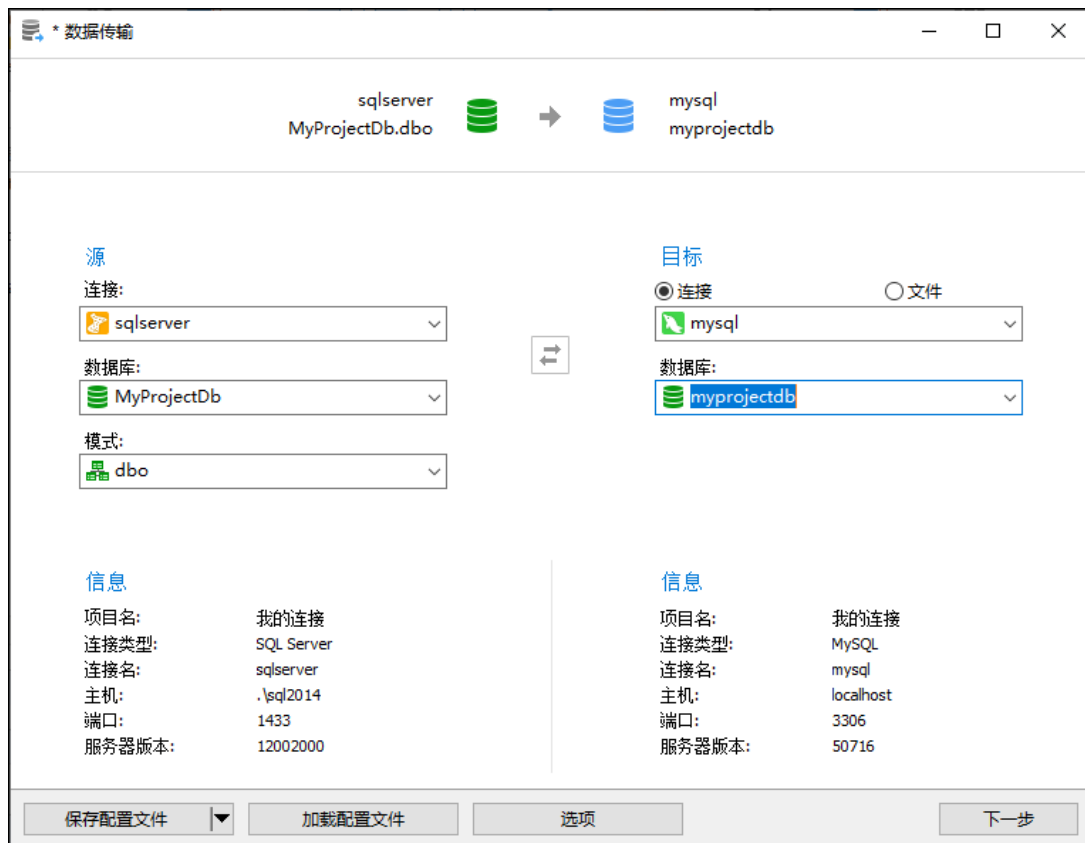
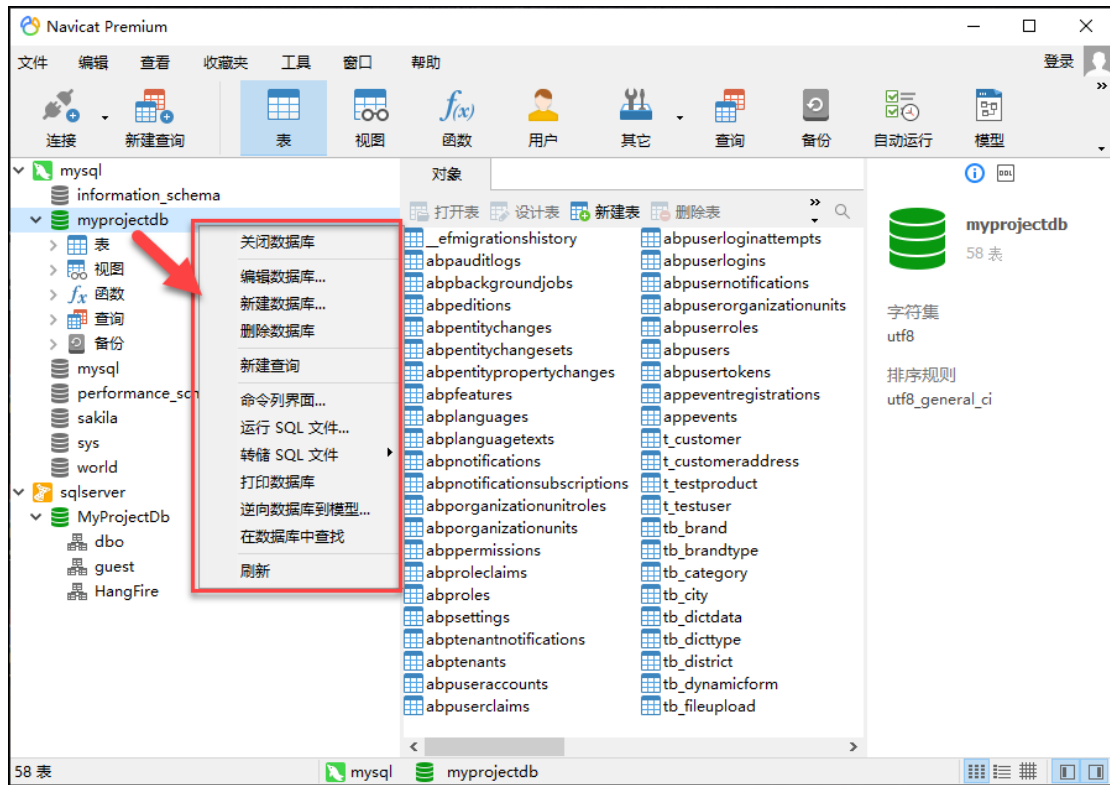
 Supported Mac OS X versions: 10.10+ <a href="#">DOWNLOAD INSTALLER</a>	 Supported Windows versions: 7+ <a href="#">DOWNLOAD INSTALLER</a>	 Supported Ubuntu versions: 14+ <a href="#">DOWNLOAD PACKAGE</a>	All Source code available on GitHub <a href="#">VIEW</a> <a href="#">BUILD FROM SOURCE</a>
---	--	--	---

下载安装后, 打开运行界面, 如果我们往里面添加键值的数据, 那么可以看到里面的数据了。



## 4.2. MySQL 数据库及管理工具

开发框架后端可以切换不同的数据库，如切换为 MySQL 数据库，那么你需要准备好 MySQL 的开发环境，一般需要 MySQL 5.7 或以上版本，而 MySQL 管理工具则推荐使用 Navicat Premium15 或以上版本，Navicat Premium 是一个非常方便、强大的 MySQL 数据库管理工具，可以生成/执行 SQL 脚本，从不同类型的数据库中导入数据库等。



## 4.3. Winform 前端开发组件

开发框架其中之一的前端是基于 Winform 界面的，而 Winform 则使用应用比较官方的界面套件 DevExpress，根据需要下载对应版本 DevExpress，获取官方授权码注册或者使用补丁文件处理即可。

## 5. 产品部署说明

开发框架的后端是基于 .net core6 的 Asp.net core 应用，因此和常规的 Asp.net core 应用部署一样；而 Vue+Element 前端应用则是基于 nodejs 的应用，部署方式又有所不同，这里介绍基于 Nginx 的部署。

### 5.1. 部署基于.netcore 的开发框架后台 Api 服务端

#### 5.1.1. 安装.net core 环境

在部署 asp.net core 服务前，需要在服务器中安装必须的环境。

由于当前框架后端的 Web API 是基于 .net core 6 的，因此，我们打开 .net core 5 的页面：

<https://dotnet.microsoft.com/en-us/download/dotnet/6.0>

The screenshot shows the .NET 6.0.11 download page. It is divided into two main sections: 'Build apps - SDK' and 'Run apps - Runtime'. The 'Build apps - SDK' section features a table for 'SDK 6.0.403' with columns for OS, Installers, and Binaries. The 'Run apps - Runtime' section includes a table for 'ASP.NET Core Runtime 6.0.11' and another for '.NET Desktop Runtime 6.0.11'. A red box highlights the 'Hosting Bundle' option in the ASP.NET Core Runtime table for Windows.

OS	Installers	Binaries
Linux	<a href="#">Package manager instructions</a>	<a href="#">Arm32</a>   <a href="#">Arm32 Alpine</a>   <a href="#">Arm64</a>   <a href="#">Arm64 Alpine</a>   <a href="#">x64</a>   <a href="#">x64 Alpine</a>
macOS	<a href="#">Arm64</a>   <a href="#">x64</a>	<a href="#">Arm64</a>   <a href="#">x64</a>
Windows	<a href="#">Arm64</a>   <a href="#">x64</a>   <a href="#">x86</a>   <a href="#">winget instructions</a>	<a href="#">Arm64</a>   <a href="#">x64</a>   <a href="#">x86</a>
All	<a href="#">dotnet-install scripts</a>	

OS	Installers	Binaries
Linux	<a href="#">Package manager instructions</a>	<a href="#">Arm32</a>   <a href="#">Arm32 Alpine</a>   <a href="#">Arm64</a>   <a href="#">Arm64 Alpine</a>   <a href="#">x64</a>   <a href="#">x64 Alpine</a>
macOS		<a href="#">Arm64</a>   <a href="#">x64</a>
Windows	<a href="#">Hosting Bundle</a>   <a href="#">x64</a>   <a href="#">x86</a>   <a href="#">winget instructions</a>	<a href="#">Arm64</a>   <a href="#">x64</a>   <a href="#">x86</a>

OS	Installers	Binaries
Windows	<a href="#">Arm64</a>   <a href="#">x64</a>   <a href="#">x86</a>   <a href="#">winget instructions</a>	

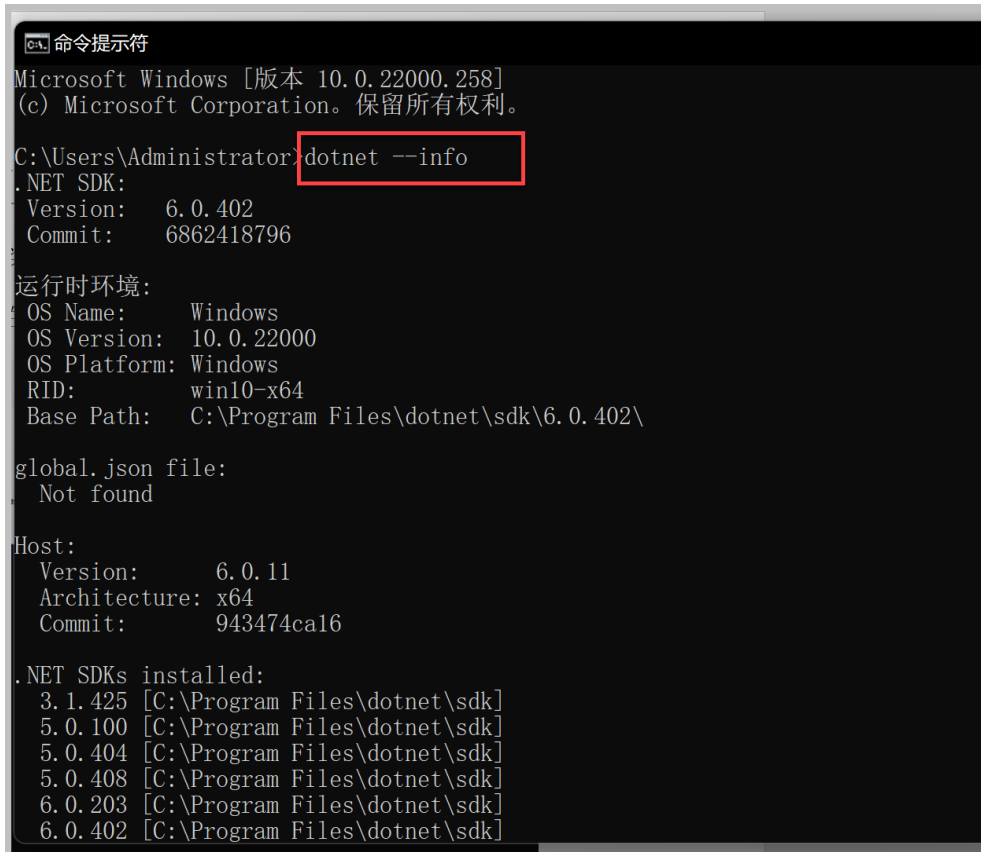
在这里选择下载 Hosting Bundle。因为 Hosting Bundle 包括了 .Net Core 运行时和 IIS 支

持。下载完成以后，双击 exe 文件即可进行安装即可。

安装完成以后我们在命令行里面输入下面的命令，检查是否安装成功：

```
dotnet --info
```

如下图所示：



```
命令提示符
Microsoft Windows [版本 10.0.22000.258]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\Administrator>dotnet --info

.NET SDK:
  Version: 6.0.402
  Commit: 6862418796

运行时环境:
  OS Name: Windows
  OS Version: 10.0.22000
  OS Platform: Windows
  RID: win10-x64
  Base Path: C:\Program Files\dotnet\sdk\6.0.402\

global.json file:
  Not found

Host:
  Version: 6.0.11
  Architecture: x64
  Commit: 943474ca16

.NET SDKs installed:
  3.1.425 [C:\Program Files\dotnet\sdk]
  5.0.100 [C:\Program Files\dotnet\sdk]
  5.0.404 [C:\Program Files\dotnet\sdk]
  5.0.408 [C:\Program Files\dotnet\sdk]
  6.0.203 [C:\Program Files\dotnet\sdk]
  6.0.402 [C:\Program Files\dotnet\sdk]
```

可以看到提示我们已经安装了 .NET Core runtimes 环境等所需软件。

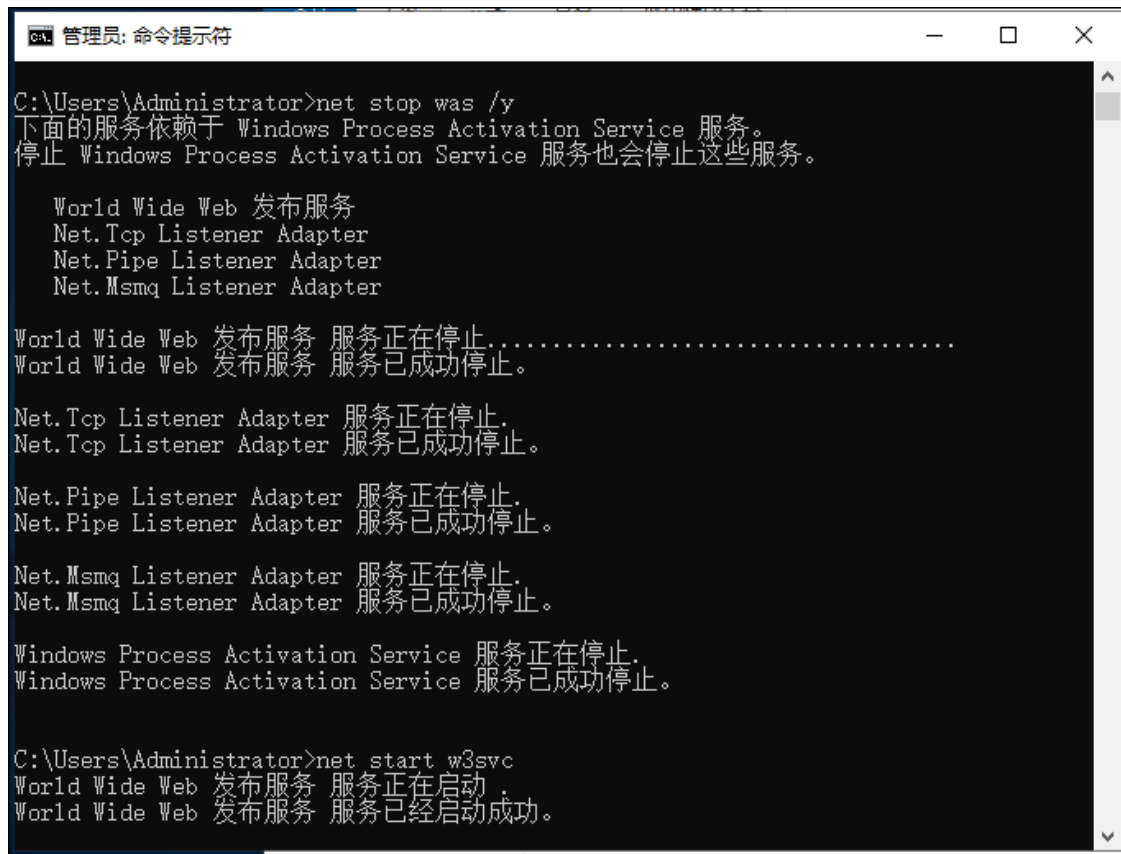
由于我们需要部署到 IIS 上面，所以安装完需使用下面的命令行重启 IIS 服务：

```
net stop was /y
```

```
net start w3svc
```

如下图所示：





```
C:\Users\Administrator>net stop was /y
下面的服务依赖于 Windows Process Activation Service 服务。
停止 Windows Process Activation Service 服务也会停止这些服务。

World Wide Web 发布服务
Net.Tcp Listener Adapter
Net.Pipe Listener Adapter
Net.Msmq Listener Adapter

World Wide Web 发布服务 服务正在停止.....
World Wide Web 发布服务 服务已成功停止。

Net.Tcp Listener Adapter 服务正在停止。
Net.Tcp Listener Adapter 服务已成功停止。

Net.Pipe Listener Adapter 服务正在停止。
Net.Pipe Listener Adapter 服务已成功停止。

Net.Msmq Listener Adapter 服务正在停止。
Net.Msmq Listener Adapter 服务已成功停止。

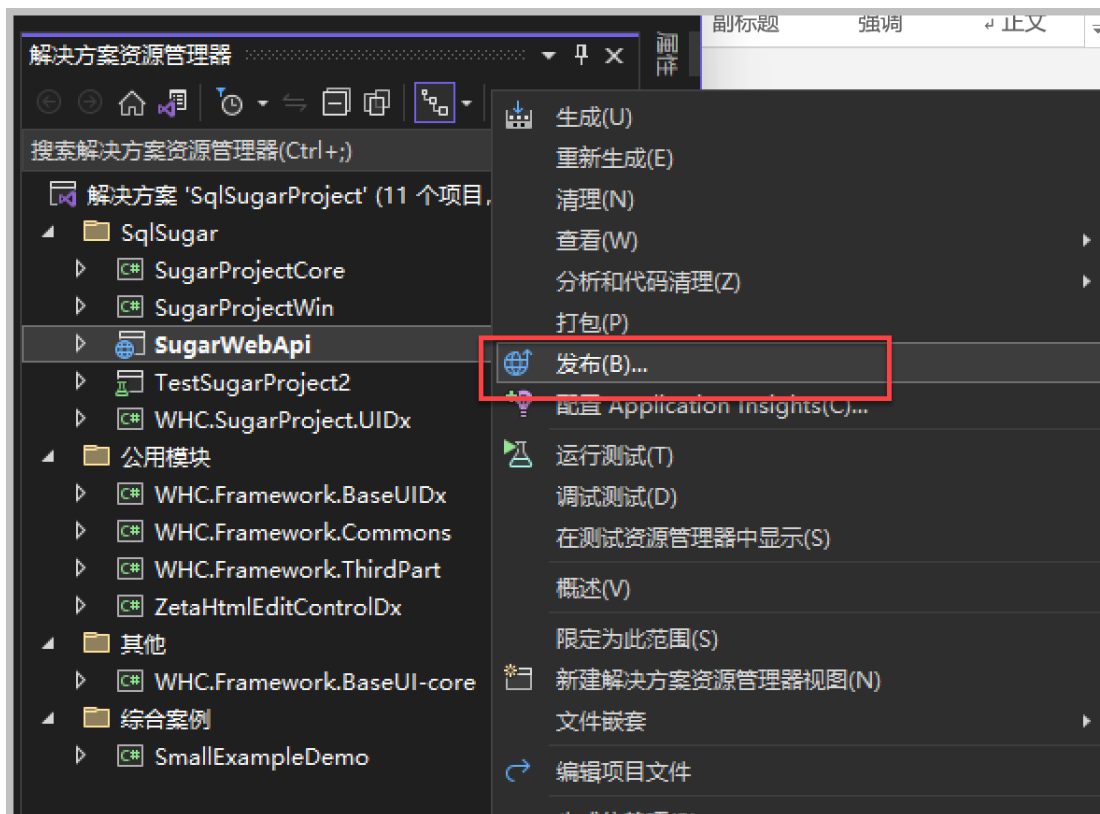
Windows Process Activation Service 服务正在停止。
Windows Process Activation Service 服务已成功停止。

C:\Users\Administrator>net start w3svc
World Wide Web 发布服务 服务正在启动。
World Wide Web 发布服务 服务已经启动成功。
```

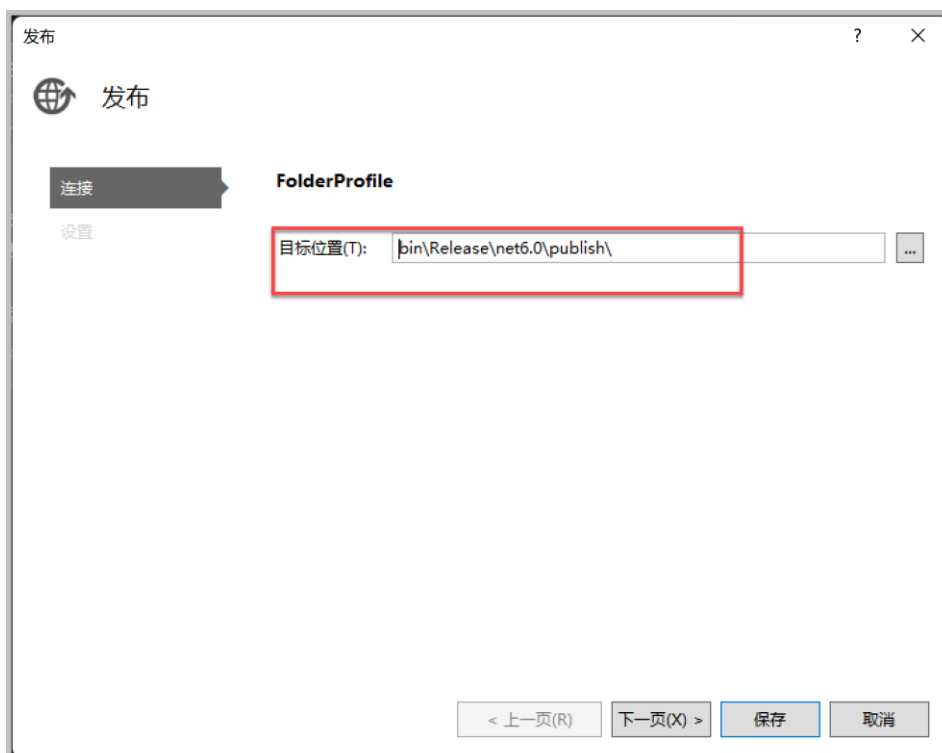
也可以在管理服务器里面重新启动 IIS。

### 5.1.2. 发布.net core 项目

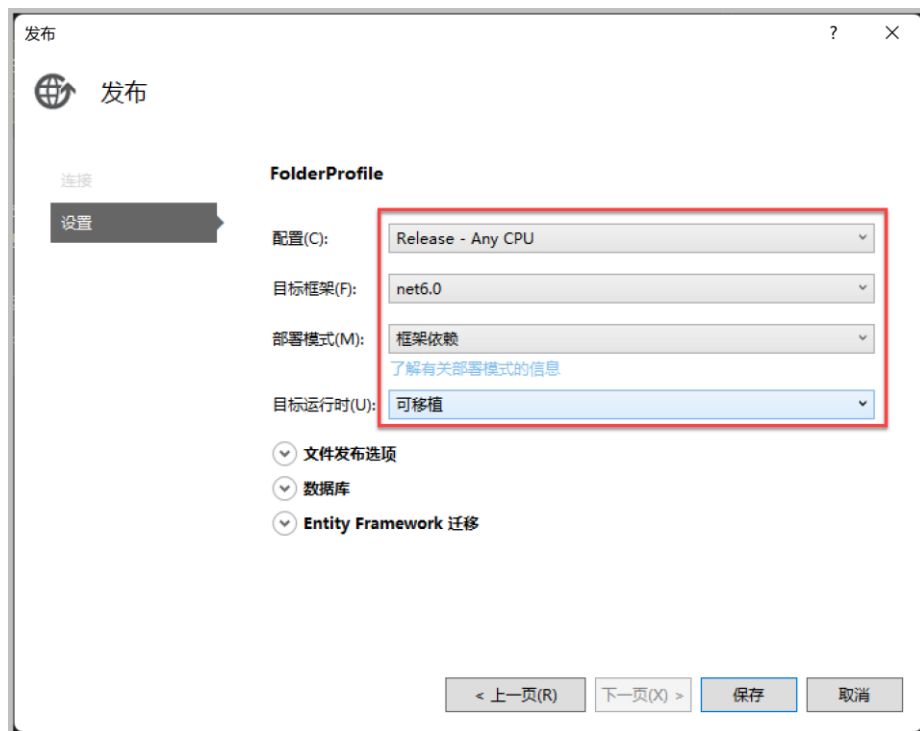
服务器.net core 环境弄好后, 下一步就是准备好发布包, 我们在开发框架的 Host 项目进行发布。



发布选择文件发布，如下所示。



然后调整设置如下所示。



最后我们生成发布包，在对应的目录下，如下所示。

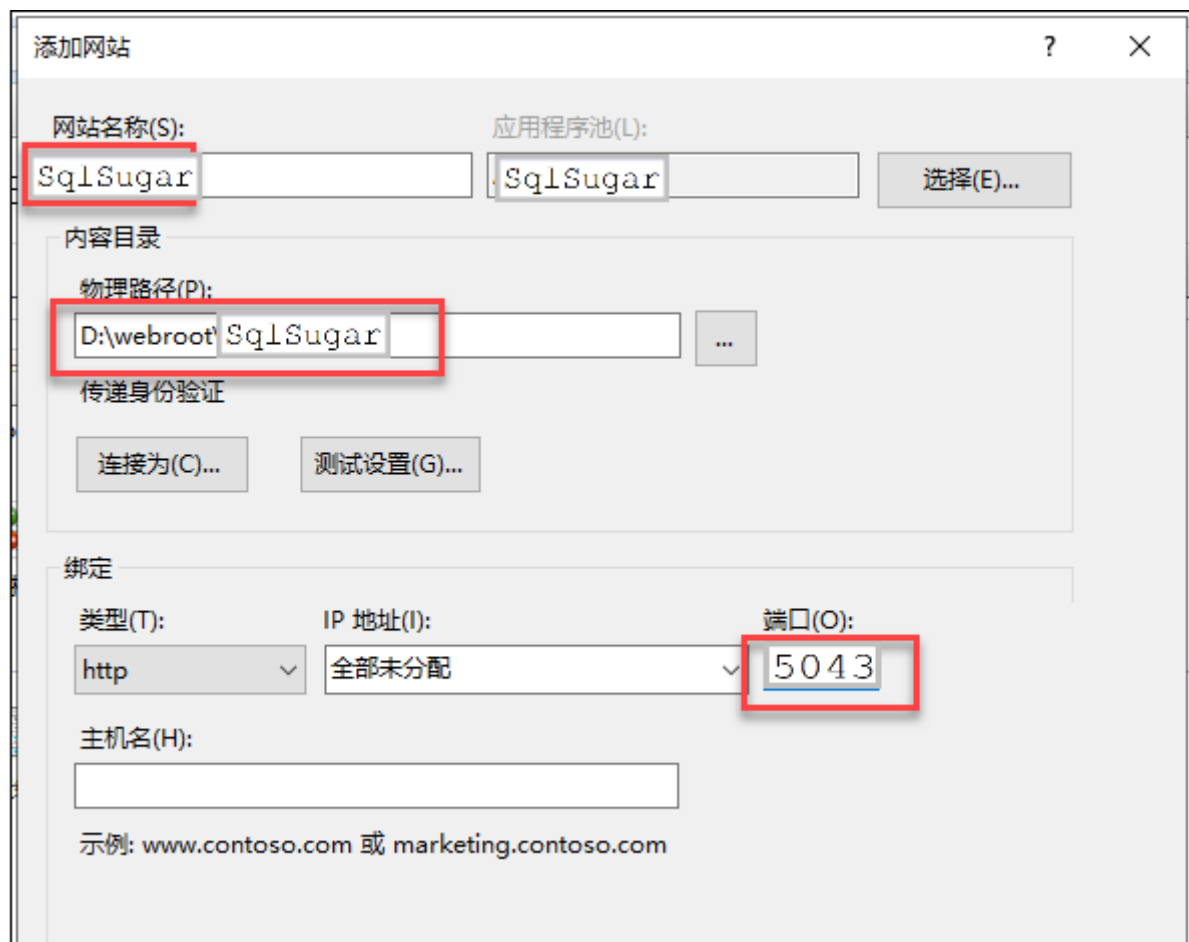
**G:\\*\*\*\Web.Host\bin\Release\net6.0\publish\**

完成后就可以把这些文件上传到服务器进行发布设置了。

### 5.1.3. 在服务器中设置 IIS

把文件上传到服务器上，然后就是准备设置好 IIS 了。

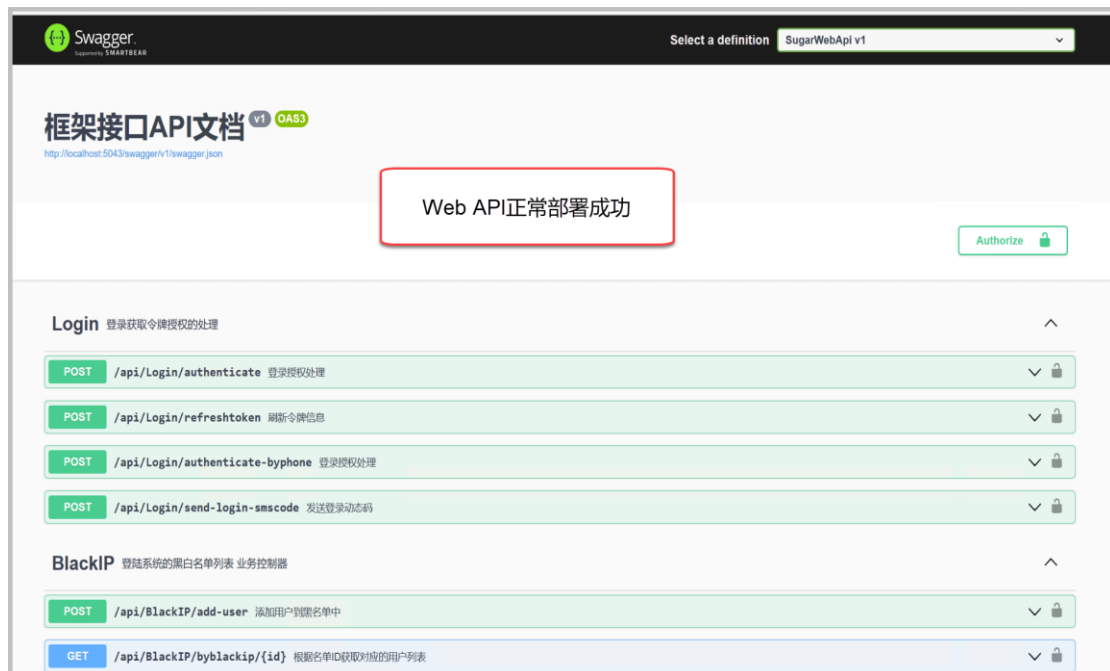
先在 IIS 服务器上创建一个网站，指定对应目录和端口等信息，如下所示。



然后找到对应的应用程序池,找到刚才创建的对应的应用程序池。设置.net clr 版本为【无

托管代码】即可。





顺利弄完 asp.net core 的后端 API 服务，那么下面就需要同时把 Vue+Element 的前端部署在服务端了。

## 5.2. 使用 Nginx 部署 Vue+Element 前端应用

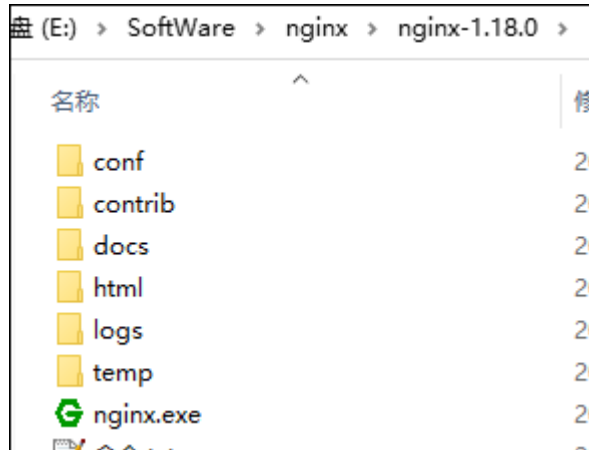
首先使用 `pnpm run build` / `npm run build` 进行 Vue+Element 的项目进行编译，编译成功后在文件项目目录的 `dist` 目录下。

部署 Vue+Element 的前端应用，建议使用 Nginx 服务，这个对于 Vue 里面的 URL 代理转向设置比较方便些。

Nginx 是一个高性能的 HTTP 和反向代理 web 服务器。

首先到 [nginx 服务网站](http://nginx.org/en/download.html) 下载对应的程序包进行安装：

<http://nginx.org/en/download.html>，建议下载稳定版本进行安装。



nginx 的 DOS 操作命令有几个，比较简单

```
start nginx 启动
```

```
nginx -s reload 刷新配置文件
```

```
tasklist /fi "imagename eq nginx.exe" 查看所有的 nginx 进程
```

```
taskkill /fi "imagename eq nginx.exe" /f 停止所有 nginx 进程
```

定位到解压的目录，然后在 DOS 窗口中执行 `start nginx` 启动 nginx 服务。

在使用前，我们需要检查 nginx 是否启动成功，直接在浏览器地址栏输入网址 `http://localhost:80`，回车，出现以下页面说明启动成功。



也可以在 cmd 命令窗口输入命令 `tasklist /fi "imagename eq nginx.exe"`，出现如下结果说明启动成功

```
管理员: 命令提示符

C:\Users\Administrator>tasklist /fi "imagename eq nginx.exe"

映像名称          PID 会话名          会话#  内存使用
-----
nginx.exe         37372 RDP-Tcp#81      2      7,520 K
nginx.exe         15856 RDP-Tcp#81      2      7,980 K

C:\Users\Administrator>
```

nginx 的配置文件是 conf 目录下的 nginx.conf，默认配置的 nginx 监听的端口为 80，如果 80 端口被占用可以修改为未被占用的端口即可。

在处理网站的 URL 代理设置前，我们先回到我们 Vue+Element 项目里面，我们在 vite.config.js 里面一般都有为跨域处理实现的代理设置，如下图所示。

```
TS vite.config.ts 9+ X
default > server > proxy
// 服务端渲染
server: {
  // 是否开启 https
  https: false,
  // 端口号
  port: VITE_PORT,
  host: '0.0.0.0',
  // 本地跨域代理
  proxy: {
    VITE_PROXY_DOMAIN_REAL.length > 0
    ? [
      {
        [VITE_PROXY_DOMAIN]: {
          target: VITE_PROXY_DOMAIN_REAL,
          // ws: true,
          changeOrigin: true,
          rewrite: (path: string) => regExps(path, VITE_PROXY_DOMAIN)
        },
        '/fileapi': {
          target: 'http://localhost:5043',
          changeOrigin: true,
          rewrite: (path: string) => regExps(path, 'fileapi')
        }
      }
    ]
    : null
  },
  plugins: getPluginsList(command, VITE_LEGACY)
},
```

```
testvue 9+ .env.production X
.env.production
1 # 线上环境项目打包路径
2 VITE_PUBLIC_PATH = /
3
4 # 线上环境路由由历史模式
5 VITE_ROUTER_HISTORY = "hash"
6
7 # 线上环境后端地址
8 VITE_PROXY_DOMAIN_REAL = "http://www.iqidi.com:5043/api"
9
10 # 是否为打包后的文件提供传统浏览器兼容性支持 支持 true 不支持 false
11 VITE_LEGACY = false
12
```

API

```
const VITE_PROXY_DOMAIN: string
```

其中的变量 VITE\_PROXY\_DOMAIN、VITE\_PROXY\_DOMAIN\_REAL 是定义在 .env.development 和 .env.production 的变量，这里配置设置了对应的反向代理。

而发布应用到服务器的时候，我们需要配置它的反向代理设置。使用 Nginx 部署



Vue+Element 前端应用的时候, 我们可以利用它的反向代理设置配置即可。

在 nginx 下的 conf/nginx.conf 中修改 nginx 的配置文件, 配置修改。

根据我在 Vue 前端项目上的 devServer 的配置, 我们在 nginx 的反向代理设置如下所示。

```
server {
    listen      8848;
    server_name localhost;

    location / {
        root    html/Iqidi-vue-admin;
        index  index.html index.htm;
        try_files $uri $uri/ /index.html =404;
    }

    location /api {
        proxy_set_header Host          $host:5043; #图片等资源需带端口获取
        proxy_set_header x-forwarded-for $remote_addr;
        proxy_set_header X-Real-IP     $remote_addr;
        proxy_pass                      http://localhost:5043/api;
    }
}
```

完整设置信息如下所示:

```
server {

    listen      8848;

    server_name localhost;

    location / {

        root    html/Iqidi-vue-admin;

        index  index.html index.htm;

        try_files $uri $uri/ /index.html =404;

    }

    location /api {

        proxy_set_header Host          $host:5043; #图片等资源需带端口获取

        proxy_set_header x-forwarded-for $remote_addr;

        proxy_set_header X-Real-IP     $remote_addr;

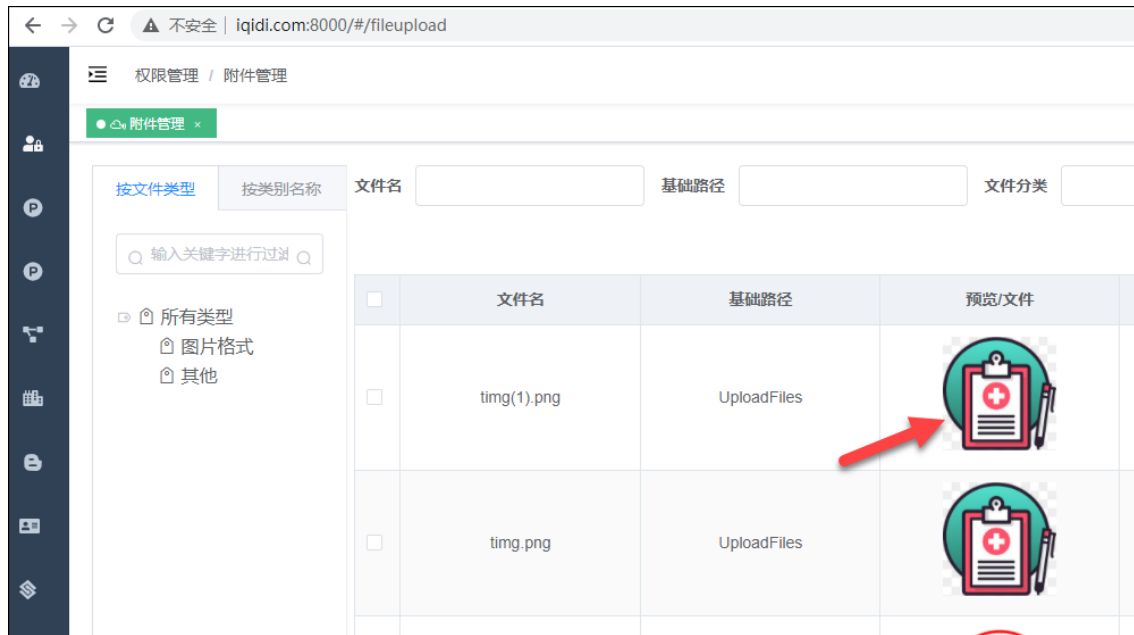
        proxy_pass                      http://localhost:5043/api;

    }

}
```

以上设置处理后, 前端使用到的相对路径, 如登录窗口, 以及调用 Web API 端的文件,

反向代理也会带上对应的端口号, 实现图片等上传 API 目录下的资源的正常访问了。



### 5.3. 利用云服务提供商的免费证书, 在服务器上发布 https 前端应用和 WebAPI 的应用

我们如果要在服务器上发布 https 前端应用和 WebAPI 的应用, 那么我们就需要用到 https 证书了。我们一般发布的应用的云服务器上, 都会提供一定量的相关的免费证书 (一般为 20 个) 供我们使用, 每个一年期限, 到期再续即可, 一般情况下基本上满足要求了, 该小节介绍如何基于云服务提供商的免费证书, 在服务器上发布 Nginx 的前端应用和基于 IIS 的 Web API 接口的 https 应用处理。

#### 5.3.1. 申请免费证书

如阿里云和腾讯云, 他们云服务器管理控制台上, 都可以找到对应免费 https 的 SSL 证书申请的入口, 如下所示。



在申请界面上，填入所需的域名，以及相关信息就可以发起申请了，申请后等待一点时间就会成功了，如阿里云的申请界面如下。

### 证书申请

**1 填写申请**

**i** 申请证书需要将您提供的个人/公司信息提交给CA机构，请知悉。注意：非国产证书，申请证书时只能在国密浏览器下显示可信。

\* 证书绑定域名   
**i** 默认包含iqidi.com域名  
**w** 请输入完整的单个域名，域名格式例如：www.aliyun.com，IP证书

\* 域名验证方式

\* 联系人

\* 所在地

\* 密钥算法  RSA

\* CSR生成方式  系统生成  手动填写  选择已有的CSR  
**i** 为保障您的证书顺利申请，建议您使用默认生成CSR的方式，手动上传建议您使用系统创建的CSR，避免因内容不正确而导致的审核失败。使用已创建的CSR申请证书，请不要在证书签发完成前删除CSR。

而腾讯云上的申请入口也是类似，如下界面所示。

### 免费证书申请

**1 提交证书申请** > **2 验证域名**

证书签发机构  亚洲诚信 TRUSTAsia

证书有效期 1年

证书绑定域名 \*    
tencent.com只赠送www.tencent.com，不包含ssl.tencent.com，需单独申请；如需绑定泛域名（例如 \*.tencent.com）或者绑定IP，请购买付费证书。[前往购买](#)

证书签发后，无法修改域名。

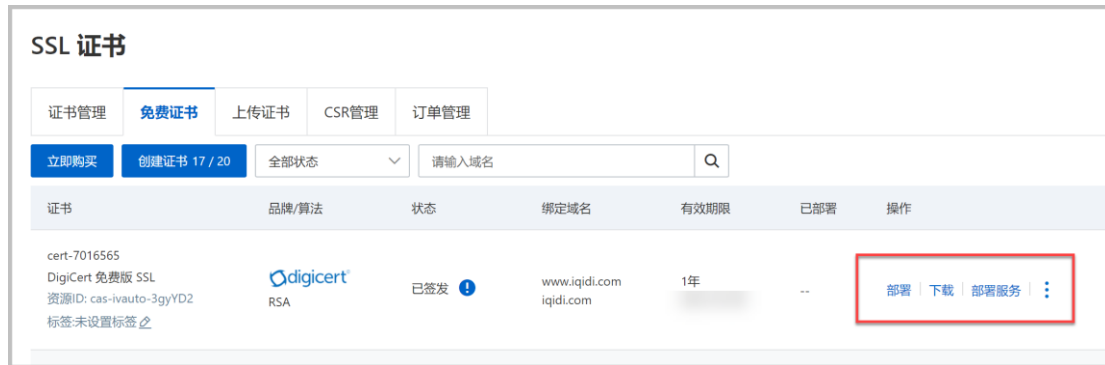
域名验证方式 **i** \*    
您需要手动为域名添加一条解析记录，操作简单，验证较快。

申请邮箱 \*  

算法选择  RSA算法 **推荐**  ECC算法  
RSA对浏览器和客户端的兼容性更好，但对网站服务器的性能开销更大；ECC加密效率更

[更多](#)

申请成功后, 在列表中就可以看到下载 SSL 证书的信息了。如下所示。



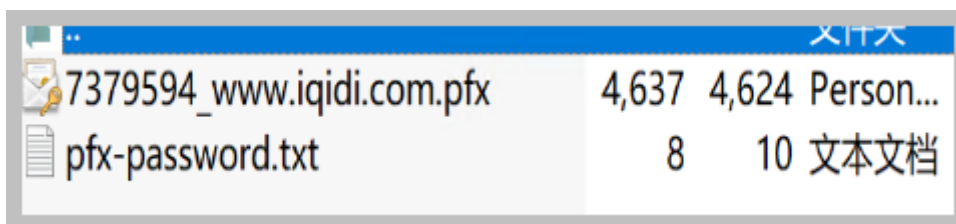
在下载界面上, 我们可以看到不同部署服务器上的不同证书下载入口, 选择我们具体的 (如这里用到了 Nginx 和 IIS 的 SSL 证书文件)



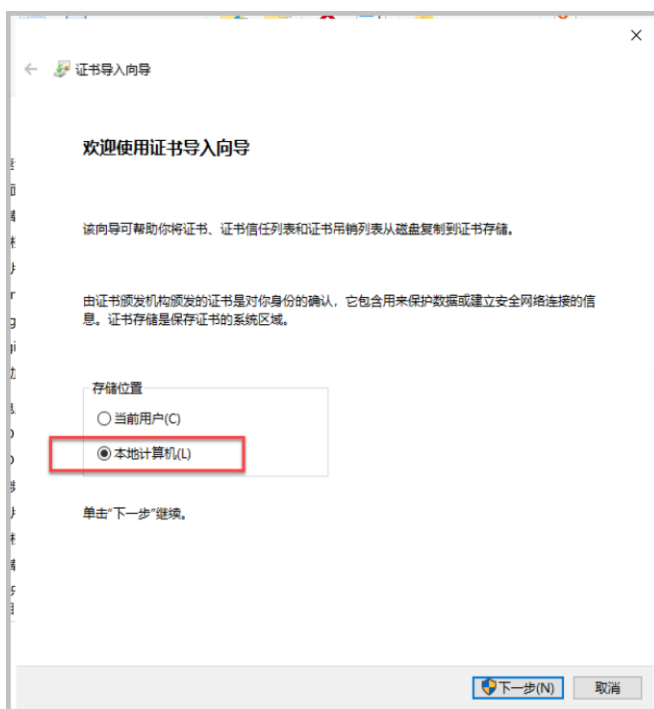
我们选择所需的证书文件下载下来备用即可。下面会继续介绍 IIS 证书的安装和使用, 以及 Nginx 的证书文件处理实现 https 的应用和接口服务。

### 5.3.2. 发布基于 IIS 的 Web API 的 https 应用接口

如我们先下载 IIS 的证书文件, 我们可以看到除了证书文件, 还有一个附带的文本文件, 是证书的密码信息。



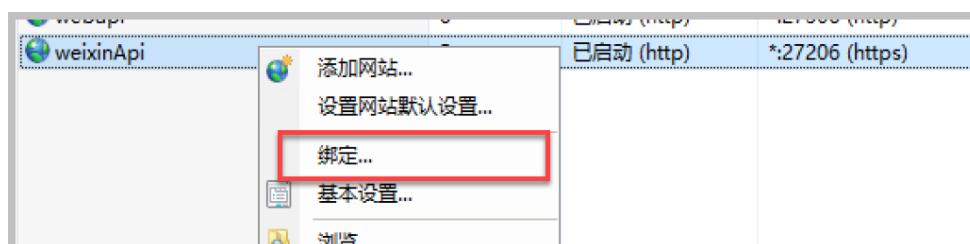
我们双击进行证书的安装，选择本地计算机的存储位置即可。



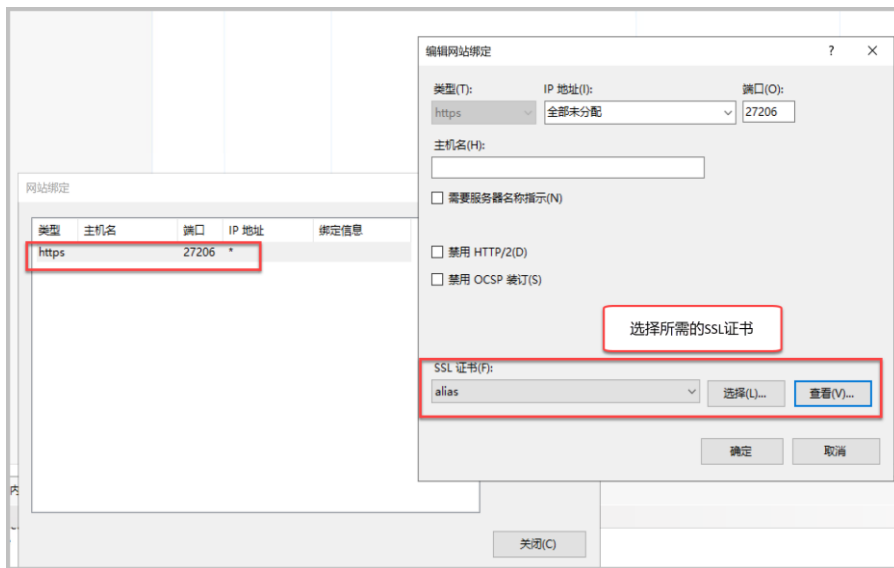
然后输入所需的证书密码，完成安装就可以了。



发布一个 IIS 的 Web API 应用, 然后在右键进行端口的绑定处理, 设置绑定的为 https, 指定端口, 并指定具体的 SSL 证书就是了, 如下所示。



绑定的界面如下所示。



这样 IIS 的服务器端的 Web API 就可以使用 https 的协议了。

### 5.3.3. 发布 Nginx 的前端应用

我们的前端是基于 Vue 的应用的, 因此应用发布后, 使用 Nginx 发布前端应用更为方便, 因此这里介绍使用 SSL 免费证书在服务器上发布 Nginx 的前端应用, 以便使用 https 协议访问。

前面我们提到了在申请完免费的 SSL 证书后, 下载对应的 Nginx 的 SSL 证书文件。



基于 Nginx 的 SSL 证书设置, 有两种方式, 一种是创建一个 ssl.conf 文件, 设置 ssl.conf 的方式指定对应的证书信息, 如下所示。



#ssl.conf文件内容

```
server {
    listen 8080 ssl http2;
    server_name localhost;
    ssl_certificate C:/WebRoot/nginx/conf/ssl/www.iqidi.com_bundle.crt;
    ssl_certificate_key C:/WebRoot/nginx/conf/ssl/www.iqidi.com.key;
    #先配置签名证书, 再配置加密证书, 签名加密证书私钥 key 为同一个!
    ssl_session_timeout 5m;
    ssl_protocols TLSv1.2;
    ssl_ciphers SM2-WITH-SMS4-SM3:ECDH:AESGCM:HIGH:MEDIUM:!RC4:!DH:!MD5:!aNULL:!eNULL;
    ssl_prefer_server_ciphers on;
    location / {
        root    html/CollectDataApp;
        index  index.html index.htm;
    }
}
```

这样我们在 conf/nginx.conf 文件中设置端口侦听, 就可以了。

```
server {
    listen      8080 ssl;
    server_name localhost;

    #charset koi8-r;
    #access_log logs/host.access.log main;

    location / {
        root    html/CollectDataApp;
        index  index.html index.htm;

        try_files $uri $uri/ /index.html =404;
    }
}
```

如果是不想独立分开两个配置文件, 也可以把 SSL 证书位置信息写在 conf/nginx.conf 文件中, 也是可以的, 如下所示。

```
server {
    listen          9002 ssl;
    server_name     localhost;
    ssl_certificate C:/WebRoot/nginx/conf/ssl/www.iqidi.com_bundle.crt;
    ssl_certificate_key C:/WebRoot/nginx/conf/ssl/www.iqidi.com.key;
    ssl_session_cache    shared:SSL:1m;
    ssl_session_timeout  5m;
    ssl_ciphers    HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers    on;

    #charset koi8-r;
    #access_log  logs/host.access.log  main;

    location / {
        root    html/AssetCheckApp;
        index  index.html index.htm;

        try_files $uri $uri/ /index.html =404;
    }
}
```

这样就合并了 SSL 设置和端口侦听的文件在一起，测试后正常使用了。

以上就是关于利用云服务提供商的免费证书，在服务器上发布 https 前端应用和 WebAPI 的应用的整个过程，证书解决了，根据不同的应用服务器，设置好对应的方式就可以实现 https 应用了。

一旦我们完成了免费证书的申请、下载，那么在服务器上不同端口的应用，都可以使用这个证书作为 SSL 证书，从而实现多个不同应用端口上公用一个 SSL 证书了，因为证书对应的是一个相同域名的，因此可以正常使用。

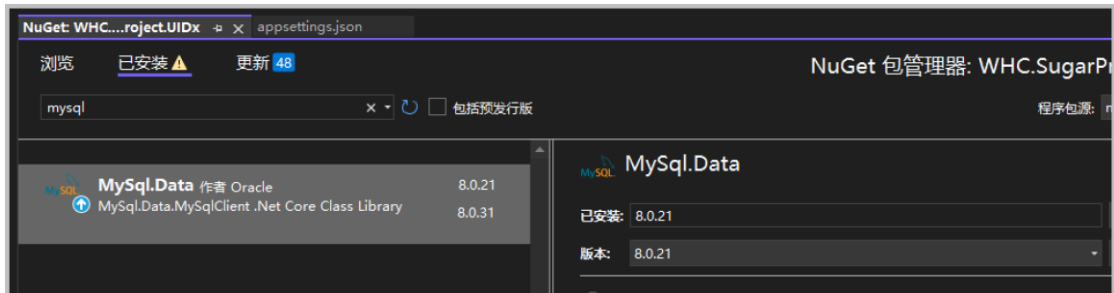
## 5.4. 开发框架使用 MySql 数据库

框架默认的数据库是 SQLServer，不过开发框架底层是基于 SqlSugar 的 ORM 框架，因此也是很容易支持其他类型的数据库的。本小节介绍在开发框架使用 Mysql 数据库，以及基于 SQLServer 创建 MySql 数据库的架构和数据的处理。

### 5.4.1. 使用 MySQL 数据库开发框架后端的代码修改

如果需要其他方式数据库的支持,那么需要引入相应的支持程序集,这里介绍如何修改增加一个 MySQL 数据库的支持。

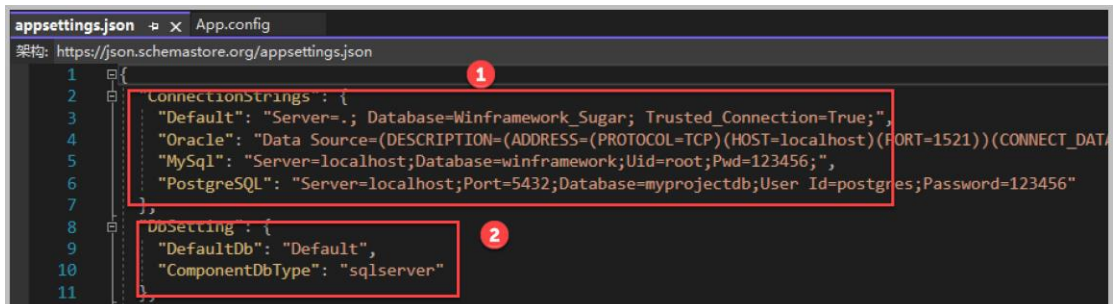
在 Nuget 中安装 MySQL.Data 的相关包,对应这些版本的驱动如下图所示。



开发框架中,如果是 Winform 前端系统,则是修改界面项目(如 WHC.SugarProject.UIDx)里面的 App.config 文件的数据库配置:



如果是 Vue 的前端框架,这修改后端 Web API 的 appsettings.json 文件中的数据库配置即可。



如果是 MySQL 数据库,那么修改 DefaultDb 指向配置节点名称: MySQL,而 ComponentDbType 是对应数据库类型,对应配置为 mysql 即可。

不同数据库的详细配置信息,可以参考《开发框架-系统数据库还原操作说明.pdf》进行了解即可。

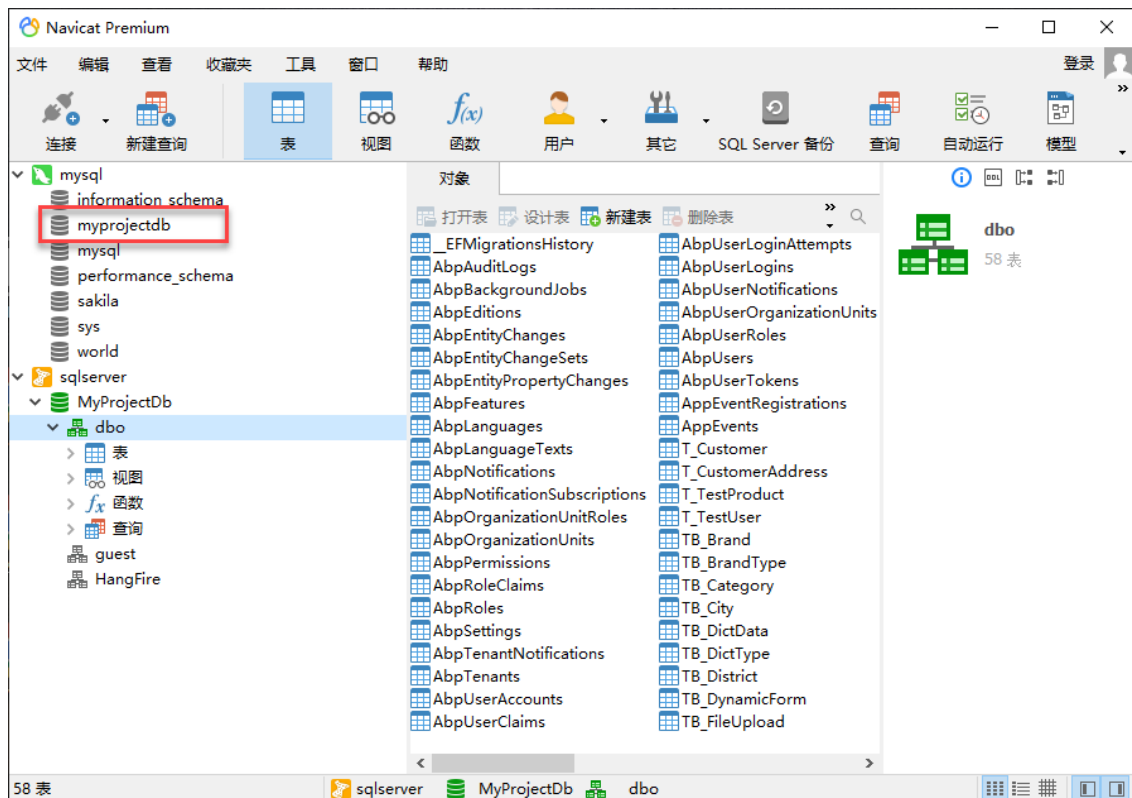
## 5.4.2. 基于 SQLServer 创建 MySQL 数据库的架构和数据

上面小节介绍了切换到其他数据库 MySQL 数据库的时候,配置参数上进行了调整修改,但是我们还需要把已有 SQLServer 上运行的数据库迁移到 MySQL 上去,如何处理呢。

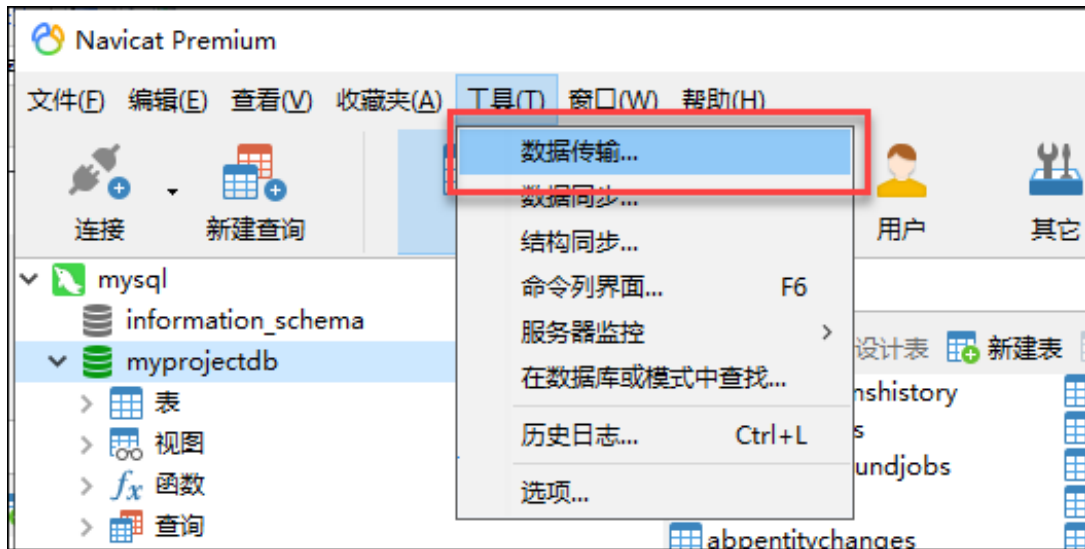
我这里通过工具的方式,把 SQLServer 数据库导入到 MySQL 数据库里面,然后在导出脚本进行一定的调整就可以完成。

我这里说到的工具就是 Navicat Premium,由于我的数据库用的是 SQLServer2014,因此建议使用 Navicat Premium15 或更高版本来处理数据库的迁移,我测试过 Navicat Premium 11 是不行的。

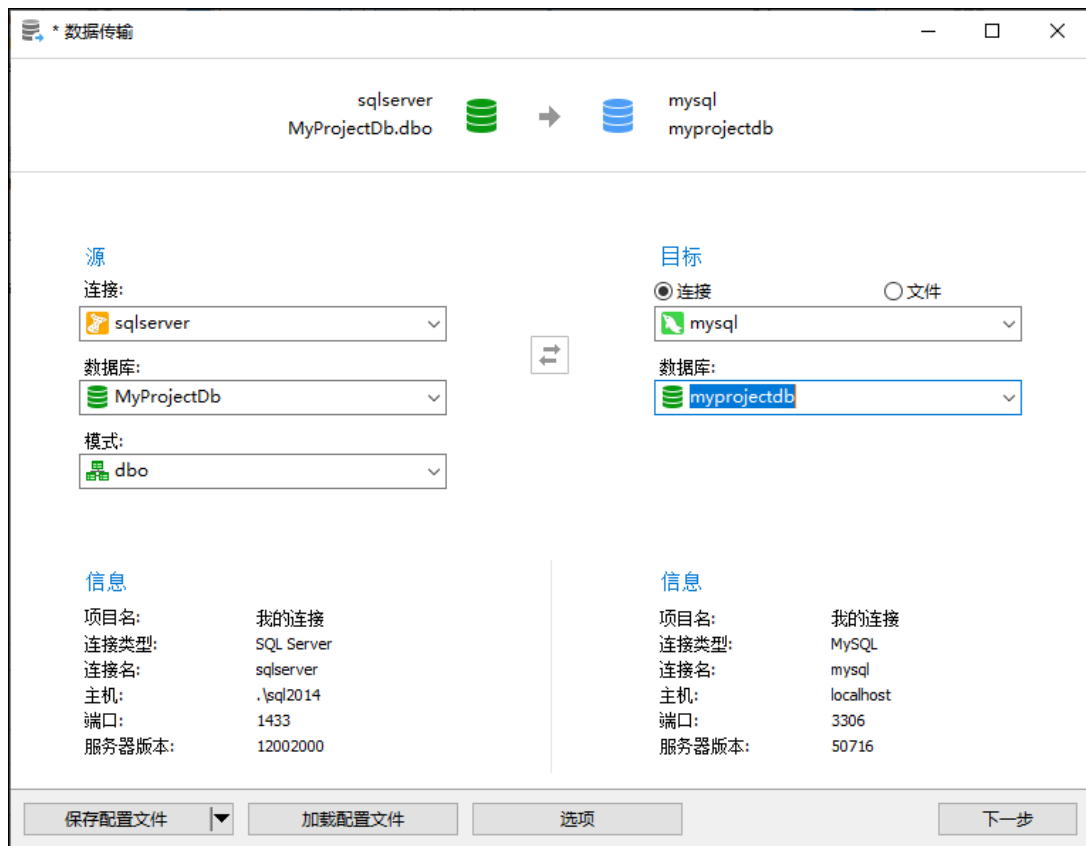
在 Navicat Premium 工具上连接好 MySQL 数据库和 SQLServer 数据库,然后在 MySQL 里面创建一个同名的数据库,以 Utf-8 编码创建数据库即可,如下所示。



然后在工具上选择数据库传输



在弹出的界面中选择源数据库和目标数据库，如下所示。

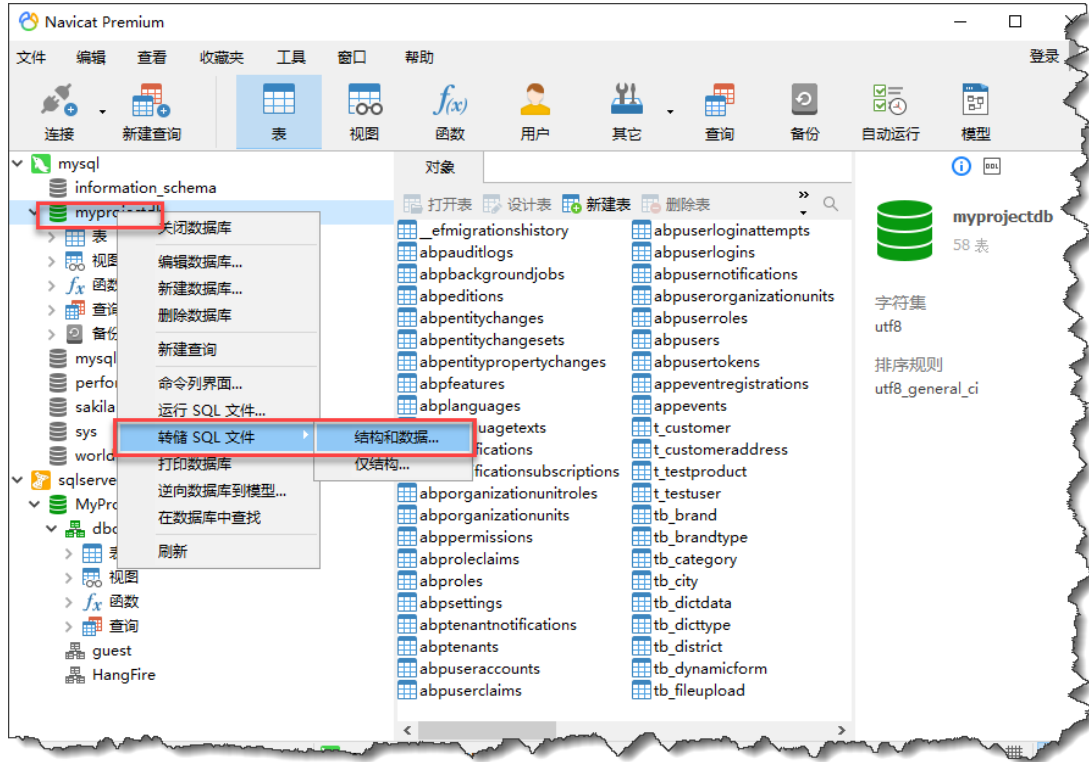


然后一步步处理即可完成数据库结构和数据的导入，这样 Mysql 数据库里面就有对应的数据库结构和数据了。

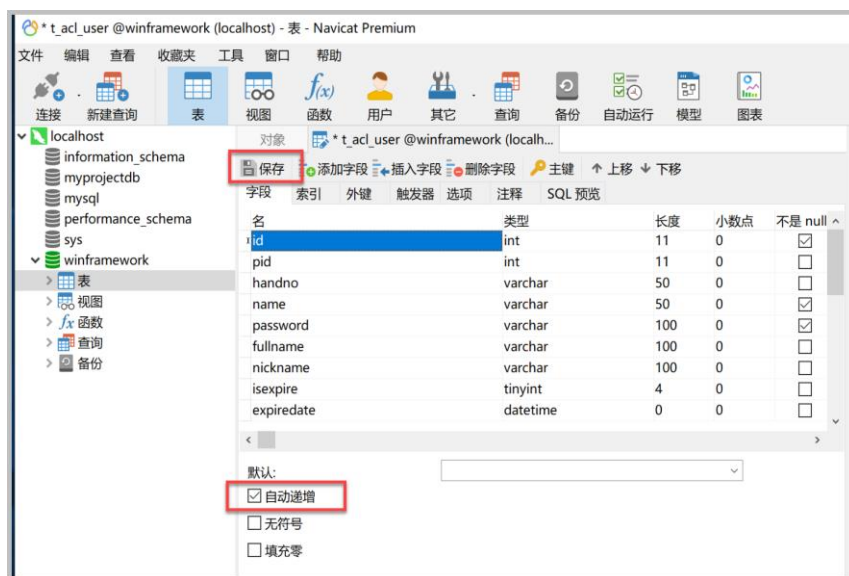
不过完成这部还是不够十分完美，因为从 SQLServer 钟导入到 Mysql 数据库里面，数据库的自增长列全部被取消了，如果我们在管理界面里面修改，则无法修改带有约束关系的主键为

自增长。

因此我们需要再次调整一下，也就是把 Mysql 数据库导出为 SQL 脚本，然后在脚本上进行一定的调整即可。



用 Notepad++ 打开导出的数据库脚本，对自增长的表字段进行添加一个关键字 AUTO\_INCREMENT，标注为自增长即可。也可以在 MySQL 数据库编辑器中进行自增长的设置，然后保存。



不过对于有外键引用的，这样保存可能不成功，因此建议导出的 MySQL 数据库脚本中

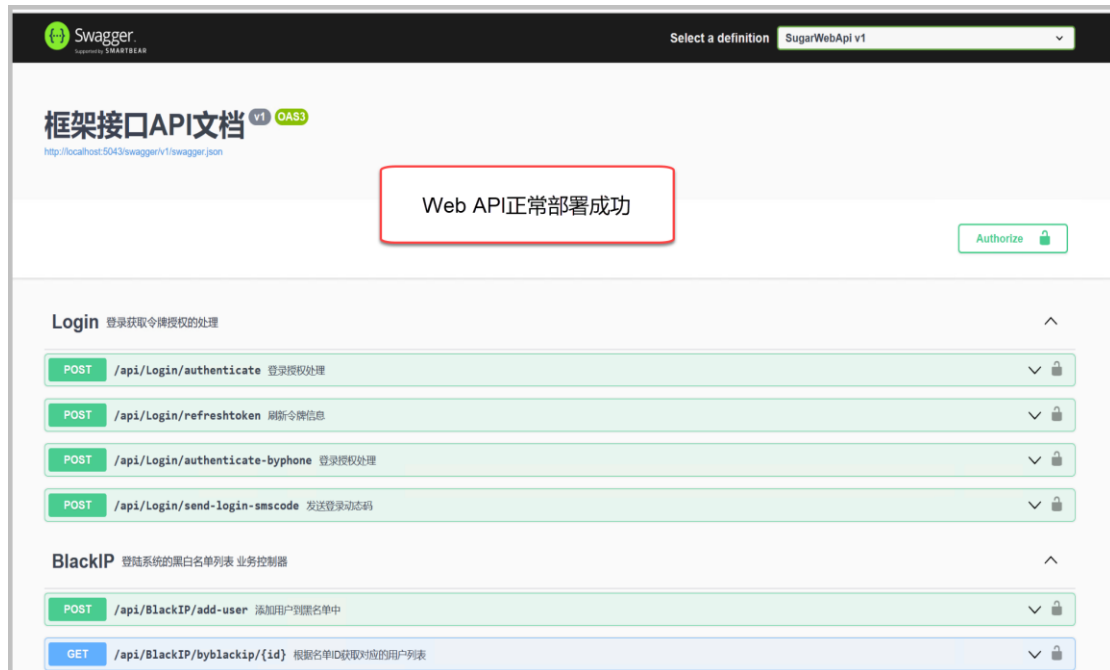
进行修改即可，如下界面所示。

```
187 -----
188 -- Table structure for abpfeatures
189 -----
190 DROP TABLE IF EXISTS `abpfeatures`;
191 CREATE TABLE `abpfeatures` (
192   `id` bigint(20) NOT NULL AUTO_INCREMENT,
193   `creationtime` datetime(0) NOT NULL,
194   `creatoruserid` bigint(20) NULL DEFAULT NULL,
195   `discriminator` longtext CHARACTER SET utf8mb4 COLLATE utf8mb4_bin NOT NULL,
196   `name` varchar(128) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin NOT NULL,
197   `value` text CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,
198   `editionid` int(11) NULL DEFAULT NULL,
199   `tenantid` int(11) NULL DEFAULT NULL,
200   PRIMARY KEY (`id`) USING BTREE,
201   INDEX `ix_abpfeatures_editionid_name` (`editionid`, `name`) USING BTREE,
202   INDEX `ix_abpfeatures_tenantid_name` (`tenantid`, `name`) USING BTREE,
203   CONSTRAINT `fk_abpfeatures_abpeditions_editionid` FOREIGN KEY (`editionid`) REFERENCES `abpeditions` (`id`) ON DELETE CASCADE,
204 ) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8_general_ci ROW_FORMAT = Dynamic;
205 -----
```

手工增加自增长的处理代码

然后删除已有的 MySQL 数据库，然后重新创建对应名称的数据库，重新导入 MySQL 脚本进行运行生成新的表和数据即可完成。

运行框架后台项目，启动 Swagger 界面则表示服务器端 Web API 正常启动，如下所示。



然后启动 Vue+Element 的客户端进行测试 MySQL 数据库的运行情况，前端正常。



