Python 开发框架之

Vue-Element 前端应用

环境安装和产品部署

说明书

V1.0

序号	修改人	修改日期	修改后版本	修改说明
1	伍华聪	2025-1-14	V1.0	初稿

目　录

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，不过功能却是非常强大的，而且开发环境可以在 Windows 系统，也可以在 Mac 系统或者 Linux 系统上运行，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，没有 Java 代码，也没有利用 Python 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。

《Python 开发框架》使用的是前后端分离模式，因此除了基于常规的跨平台桌面前端（**WxPython 前端框架**）外，还可以接入其他前端，本文档介绍就是基于 FastAPI 的统一数据接口 Web API 的 BS 终端（**Vue + Element 的前端**）的相关说明。

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》(http://wuhuacong.cnblogs.com)		博客园
1	《》		内部

1.4. 术语与缩写

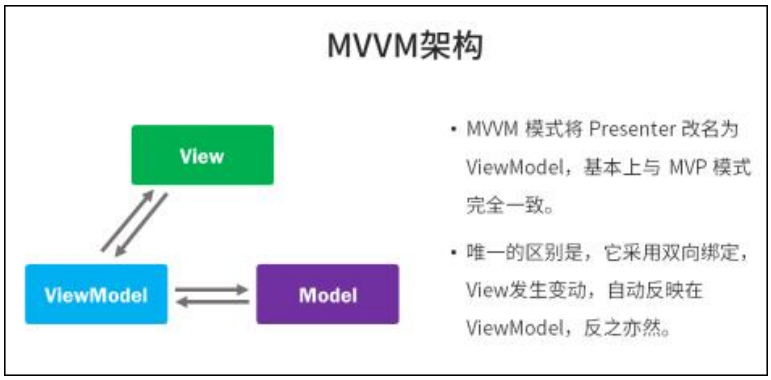
1 在本文件中出现的“系统”、“框架”一词，除非特别说明,均适用于《Python开发框架》。

2 当前后端Web API基于Python的FastAPI,如无特殊说明，均指该后端。

2. 基础知识介绍

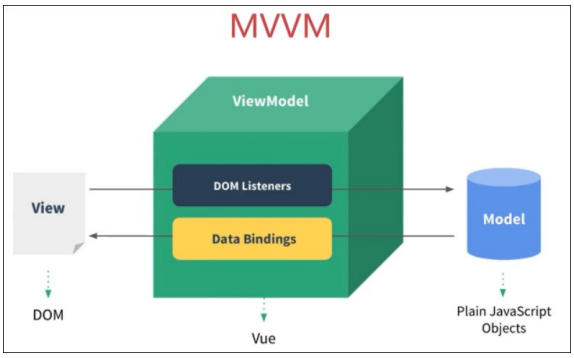
2.1. Vue 框架简介

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+TypeScrip+ElementPlus 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. ElementPlus 介绍

Element(新版本为 ElementPlus)，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

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

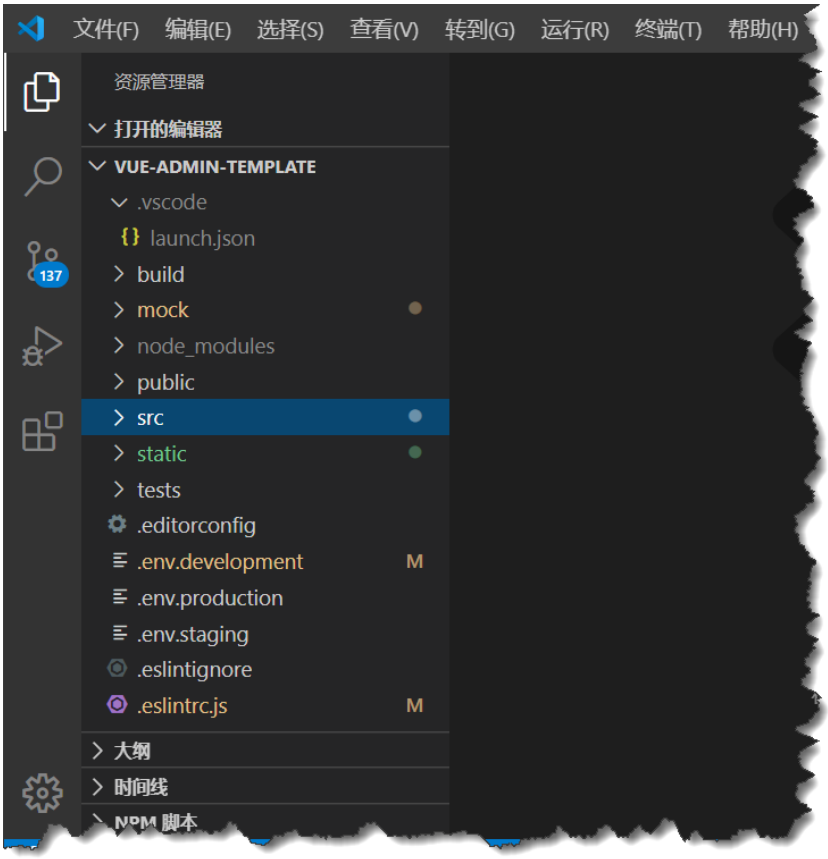
VS Code（Visual Studio Code）是由微软研发的一款免费、开源的跨平台文本（代码）编辑器。这是一个轻量级但功能强大的代码编辑器，支持丰富的扩展。你可以从 Visual Studio Code 官方网站 下载。

官网：https://code.visualstudio.com

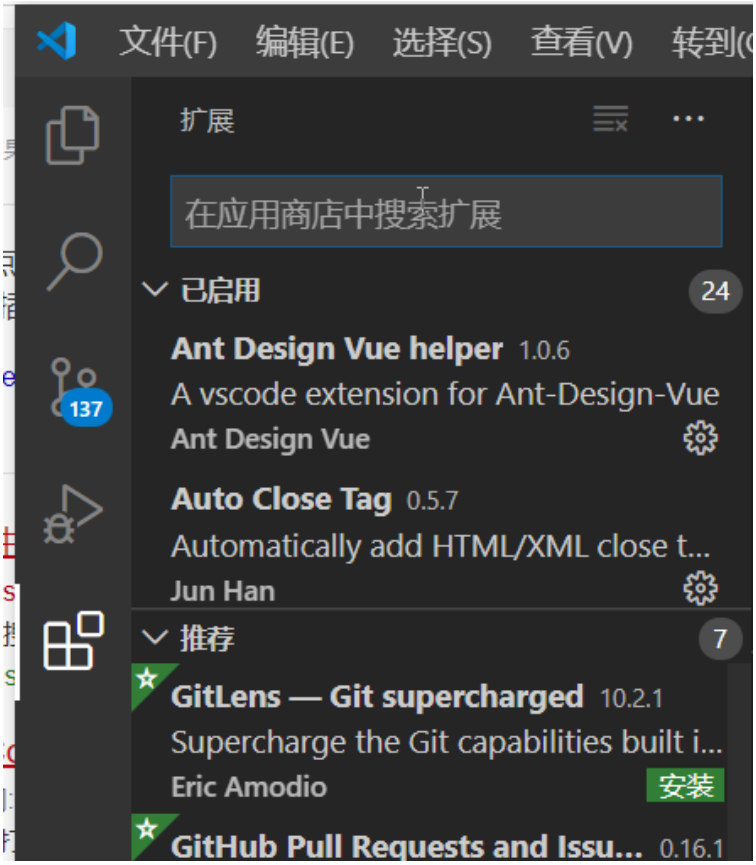
文档：https://code.visualstudio.com/docs

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：

AI 编程助手，下面任选其一即可：

1、Fitten Code：

它是由非十大模型驱动的 AI 编程助手，它可以自动生成代码，提升开发效率，帮您调试 Bug，节省您的时间，另外还可以对话聊天，解决您编程碰到的问题。

Fitten Code 是由非十大模型驱动的 AI 编程助手，它可以自动生成代码，提升开发效率，帮您调试 Bug，节省您的时间。还可以对话聊天，解决您编程碰到的问题。免费且支持 80 多种语言：Python、C++、Javascript、Typescript、Java 等。

强烈推荐使用，自动补齐代码功能，可以节省很多手工键入代码的时间，减少错误。

2、GitHub Copilot：

GitHub Copilot 主要作为 Visual Studio Code（VS Code）的插件提供。你可以从 VS Code 的插件市场下载并安装它。最新的 Copilot 已经可以免费试用。

GitHub Copilot 是一种由 GitHub 与 OpenAI 合作开发的人工智能编程助手。它使用 GPT-3 或更高版本的语言模型，旨在帮助开发人员编写代码、提供代码片段、自动生成注释和文档，甚至根据上下文和已有代码提供建议。它集成在多个开发环境中，尤其是在 Visual Studio Code 中，作为插件使用。

主要功能：

1. **自动补全代码：** GitHub Copilot 会根据你正在编写的代码提供实时建议，自动完成代码行或函数。它可以根据函数名称、变量类型或你前面的代码片段来推测并生成合理的后续代码。
2. **生成代码片段和函数：** 你可以输入一个简短的注释或描述（例如："函数计算斐波那契数列"），然后 Copilot 会生成一个完整的代码实现。
3. **智能提示和补充：** Copilot 会根据上下文理解你的意图，提供相关的代码建议。它不仅能在你编写代码时给出补全建议，还能在你不确定如何开始时帮助你生成框架代码。
4. **跨语言支持：** GitHub Copilot 支持多种编程语言，包括 Python、JavaScript、TypeScript、Go、C++ 等。它能够根据你正在使用的编程语言提供相应的建议。
5. **重构代码和优化：** 在一些情况下，Copilot 能够识别出代码中的重复部分，提供优化或重构建议，帮助你改进代码质量。

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。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

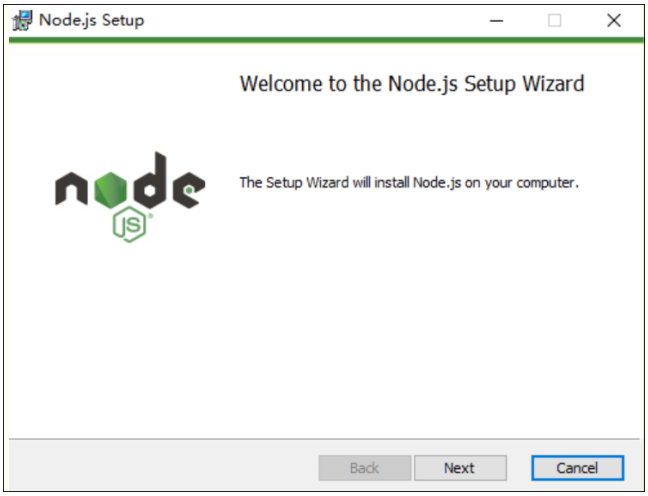
利用 VS Code 开发，我们很多时候，需要使用命令行 npm 进行相关模块的安装，这些需要 node 环境的支持，安装好 node 后，npm 也就一起安装好了。

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

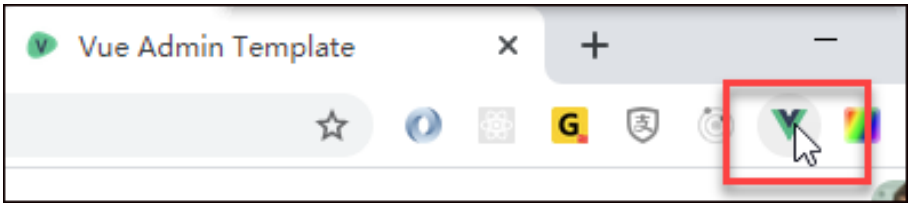
Vue (读音 /vju:/，类似于 view) 是一套用于构建用户界面的渐进式框架。

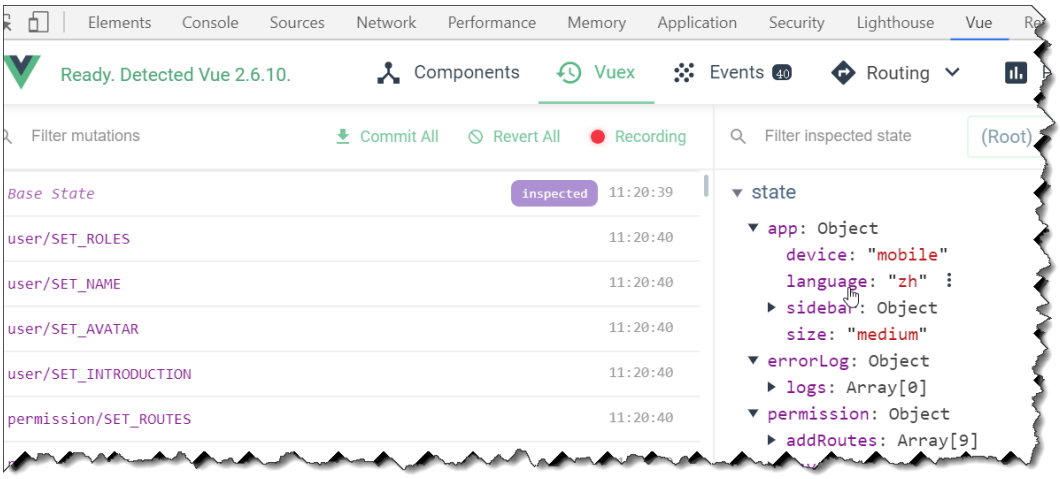
全局安装：npm install vue-cli -g （全局卸载：npm uninstall vue-cli -g）

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

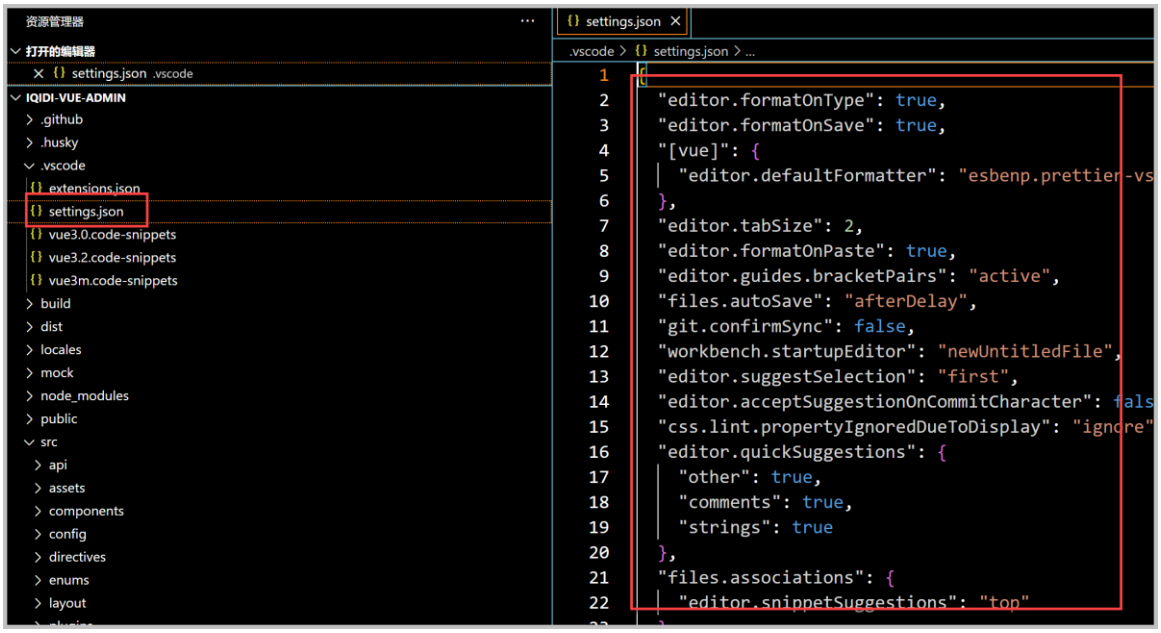




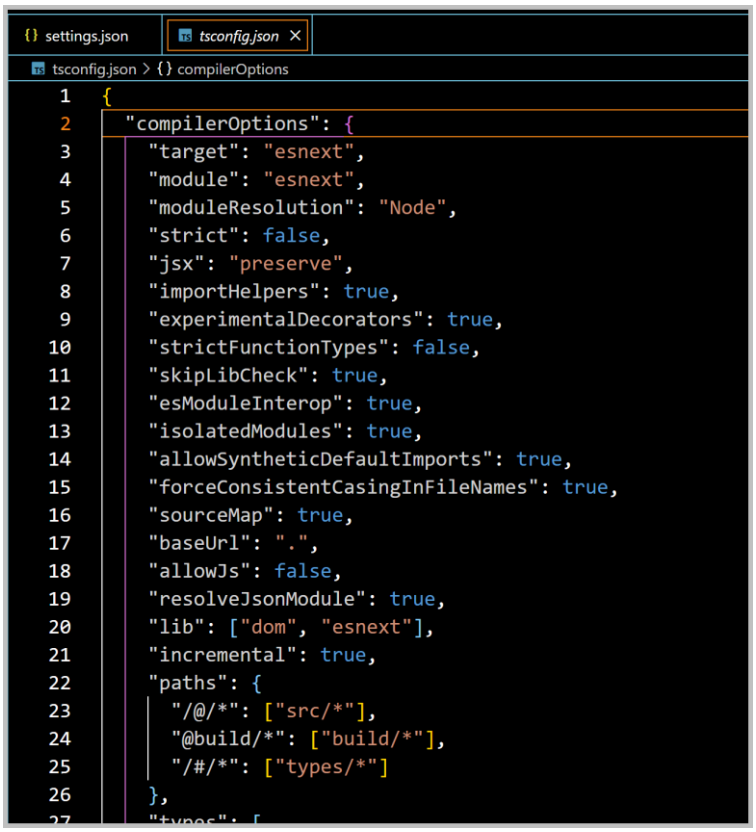
3.5. 开发环境的配置使用

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。

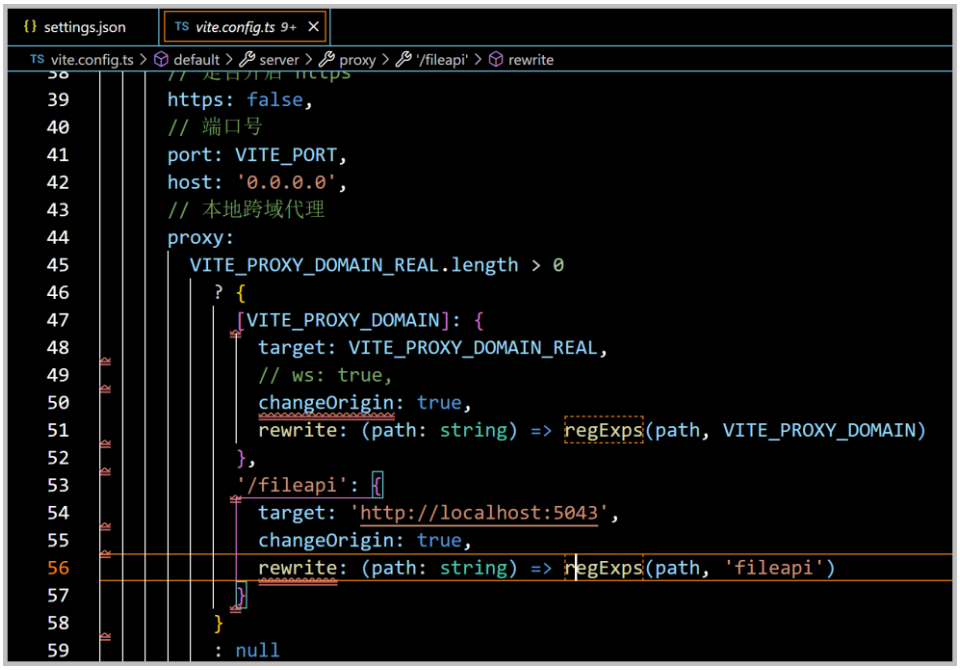
如下是相关设置的代码视图：



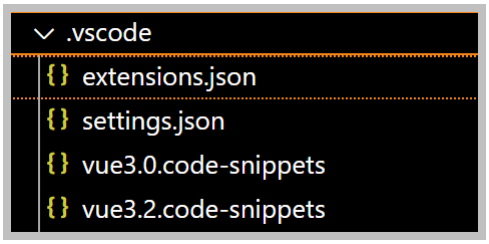
如需修改 TypeScript 的相关设置，可以打开 tsconfig.json 内容进行修改即可，如下代码所示。



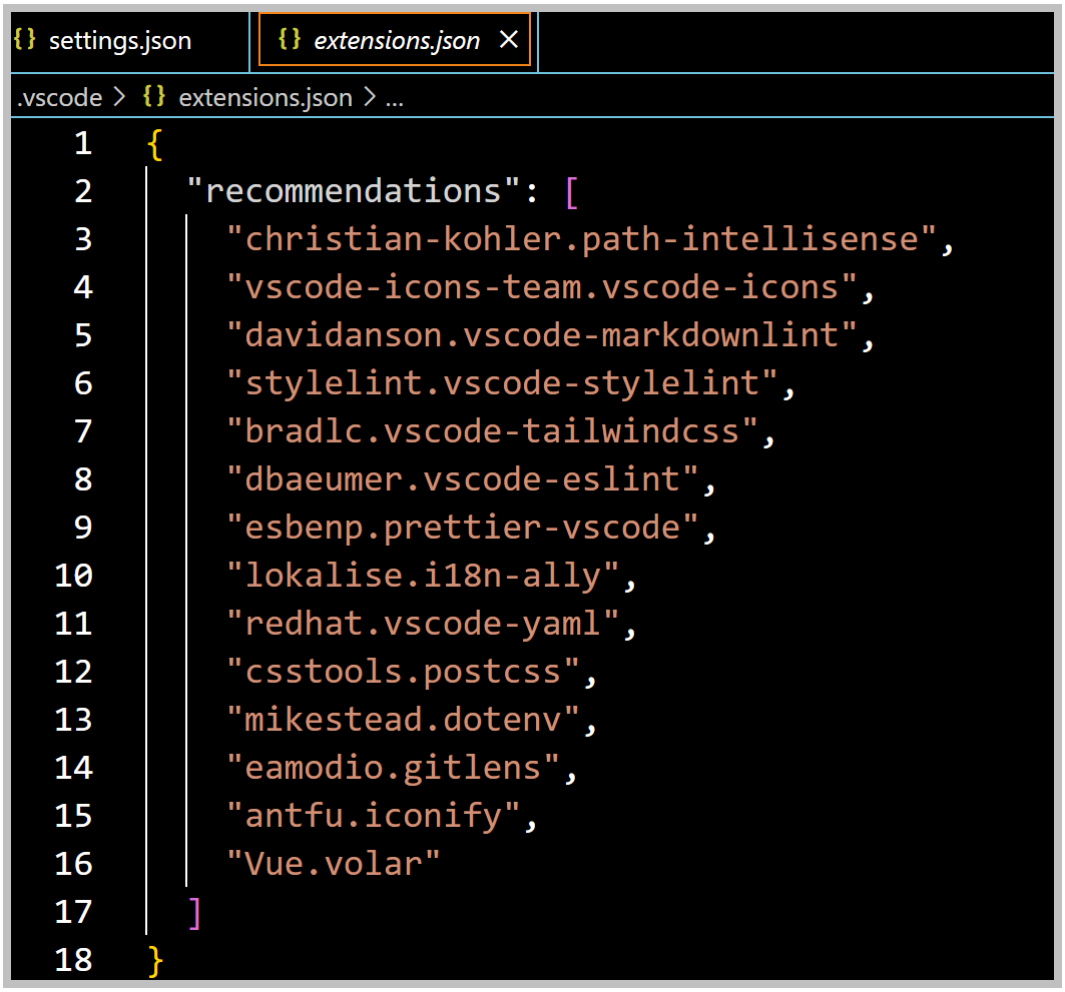
如需修改 Vite 编译器的项目相关设置，可以修改 vite.config.ts 文件内容，如下所示。



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



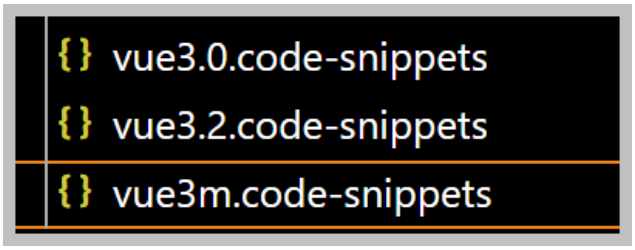
打开后可以进行查看相关的第三方 VSCode 扩展工具，项目在开始使用的时候，使用 pnpm i 会一键安装平台推荐的 vscode 插件,VSCode 自动安装所需的插件。



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。

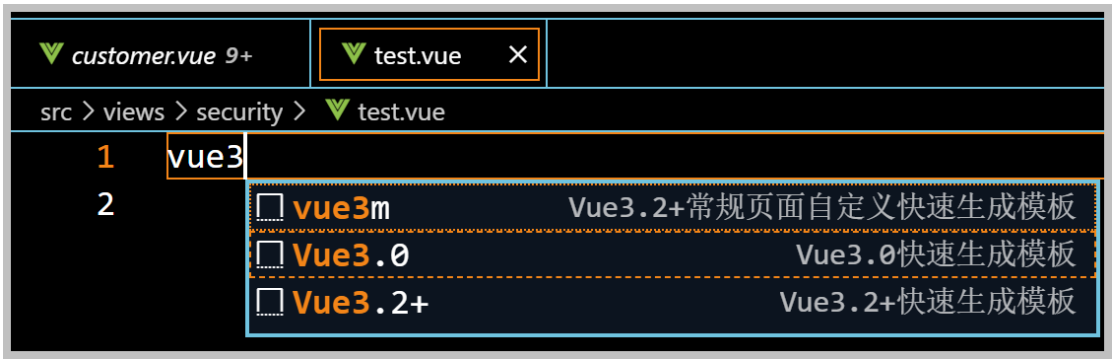
注：为了提高开发效率，很多时候会在 vscode 中配置好一些常用的代码片段，在日常编程，特别是在工作中写内容相似的业务代码时，利用 Snippet 功能，可以极大加快编程效率。

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。



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。

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。



选择对应的快捷命令，就可以生成对应的代码块了。

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。

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

《Python 开发框架》的后端是基于 Python+FastApi+SqlAlchemy + Pydantic+Redis 的技术路线。

FastAPI 是一个现代、快速（高性能）的 Web 框架，用于构建 API。它基于 Python 3.7+ 的类型提示，并且依赖于 Starlette（用于 web 服务器和路由）和 Pydantic（用于数据验证和序列化）。FastAPI 的设计目标是提供与 Flask 和 Django 类似的开发体验，但在性能、类型安全和开发者友好性方面做出更大的提升。GitHub 地址：https://github.com/fastapi/fastapi

FastAPI 的主要特性

- **极高的性能**: 基于 ASGI 的异步支持，使得 FastAPI 在性能上接近 Node.js 和 Go 的水平，适合处理高并发。
- **自动生成 API 文档**: 使用 OpenAPI 和 JSON Schema 自动生成交互式的 API 文档（如 Swagger UI 和 ReDoc）。
- **基于类型提示的自动验证**: 利用 Python 的类型提示和 Pydantic，自动进行数据验证和解析。
- **异步支持**: 原生支持 async 和 await，能够处理异步任务，适合与数据库、第三方 API、

WebSocket 等交互。

- **内置依赖注入系统**：使得依赖的声明和管理变得简洁而强大，便于模块化设计。
- **开发者友好**：提供了详细的错误信息和文档，支持自动补全，极大提升了开发效率。

当你运行 FastAPI 应用时，它会自动生成交互式文档：

- **Swagger UI**：访问 http://127.0.0.1:8000/docs
- **ReDoc**：访问 http://127.0.0.1:8000/redoc

这两个文档界面可以让你查看 API 的结构，甚至可以直接在界面中进行 API 调用。

FastAPI 是一个非常现代化和高效的框架，非常适合用于构建高性能的 API。其自动文档生成、数据验证和依赖注入等特性，使得开发者能够更快、更安全地编写代码，并提供出色的用户体验。

4.1. 常规开发工具的安装

《Python 开发框架》的后端 Web API 是基于 FastAPI 的框架应用，开发工具一般采用 VSCode；数据库默认采用 MySql，而登录部分也采用了 Redis 缓存的数据库。

开发框架支持多种数据库，SqlServer、MySql、Oracle、SQLite、PostgreSQL、达梦数据库都可以支持，可以再配置文件中修改指定即可。

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。

MySQL 或其他数据库则自行获取地址进行下载，数据库表管理推荐使用 Navicat Premium 17 或以上版本进行维护。

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 的一个升级版本，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。可以理解了有两列的 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，指出在多长时间内，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。

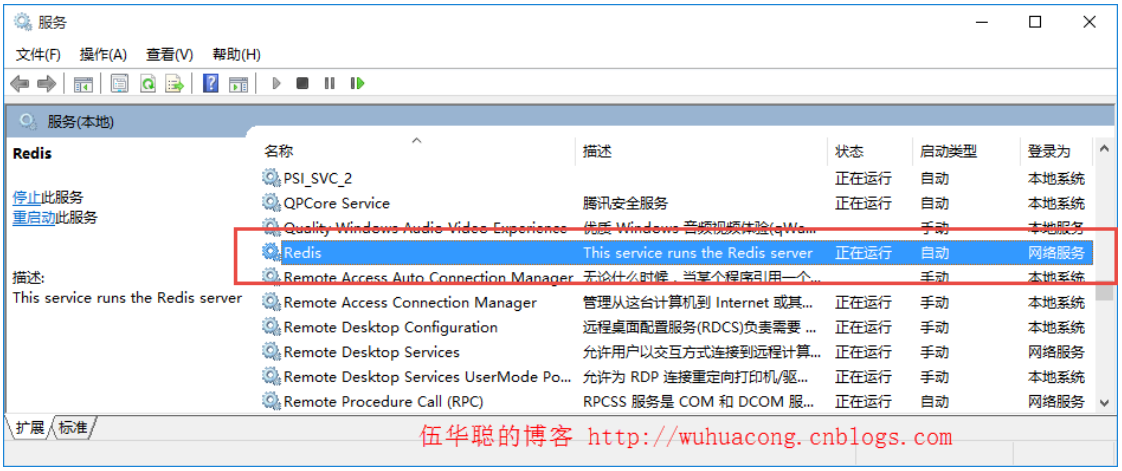
appendfsync no/always/everysec ，appendfsync 配置，no 表示等操作系统进行数据缓存同步到磁盘，always 表示每次更新操作后手动调用 fsync()将数据写到磁盘，everysec 表示每秒同步一次。

4）Redis 的安装

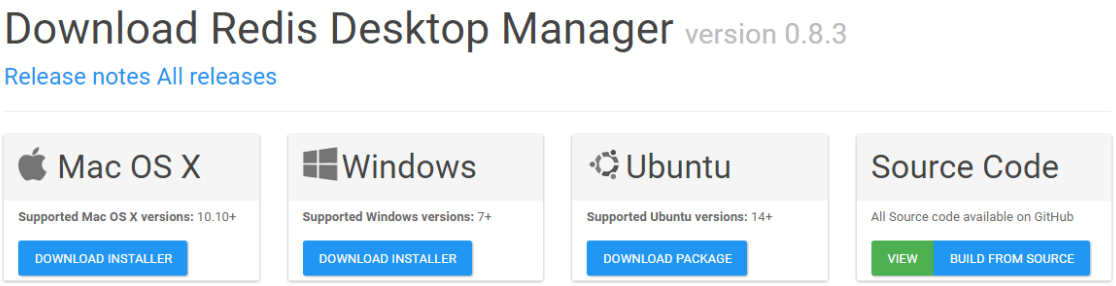
Redis 可以在不同的平台运行，不过我主要基于 Windows 进行开发工作，所以下面主要是基于 Windows 平台进行介绍。

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。

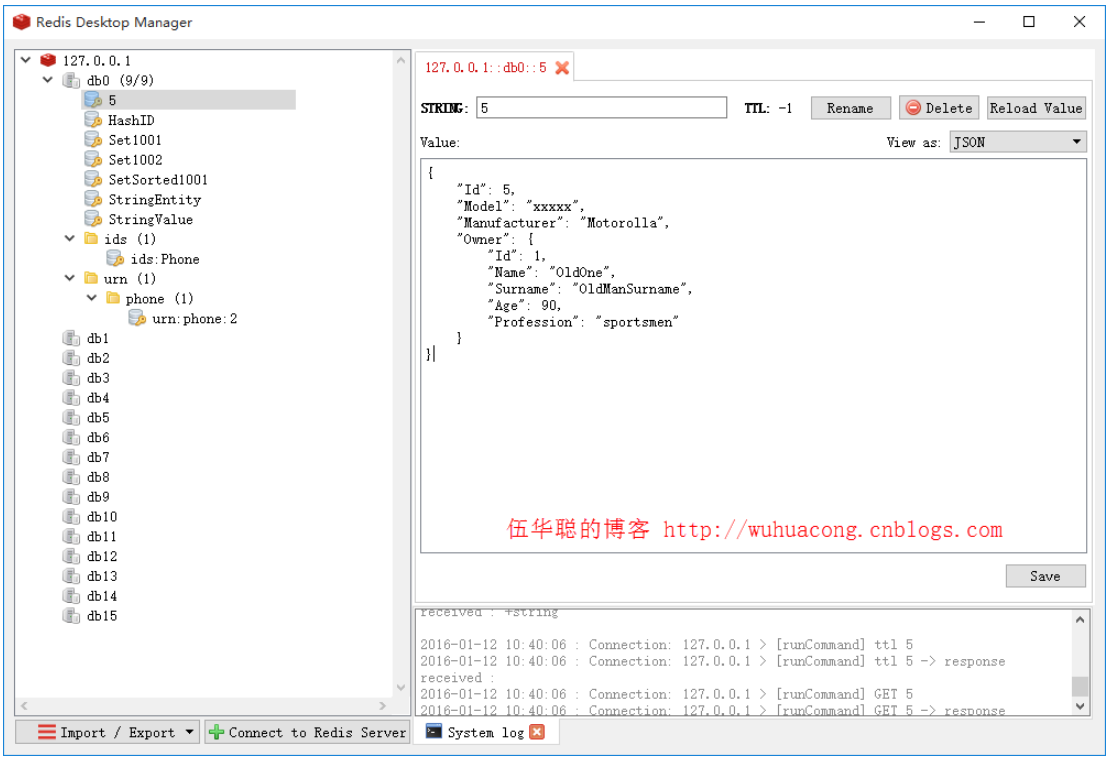
当前可以下载到最新的 Windows 安装版本为 3.0，安装后作为 Windows 服务运行，安装后可以在系统的服务里面看到 Redis 的服务在运行了，如下图所示。



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

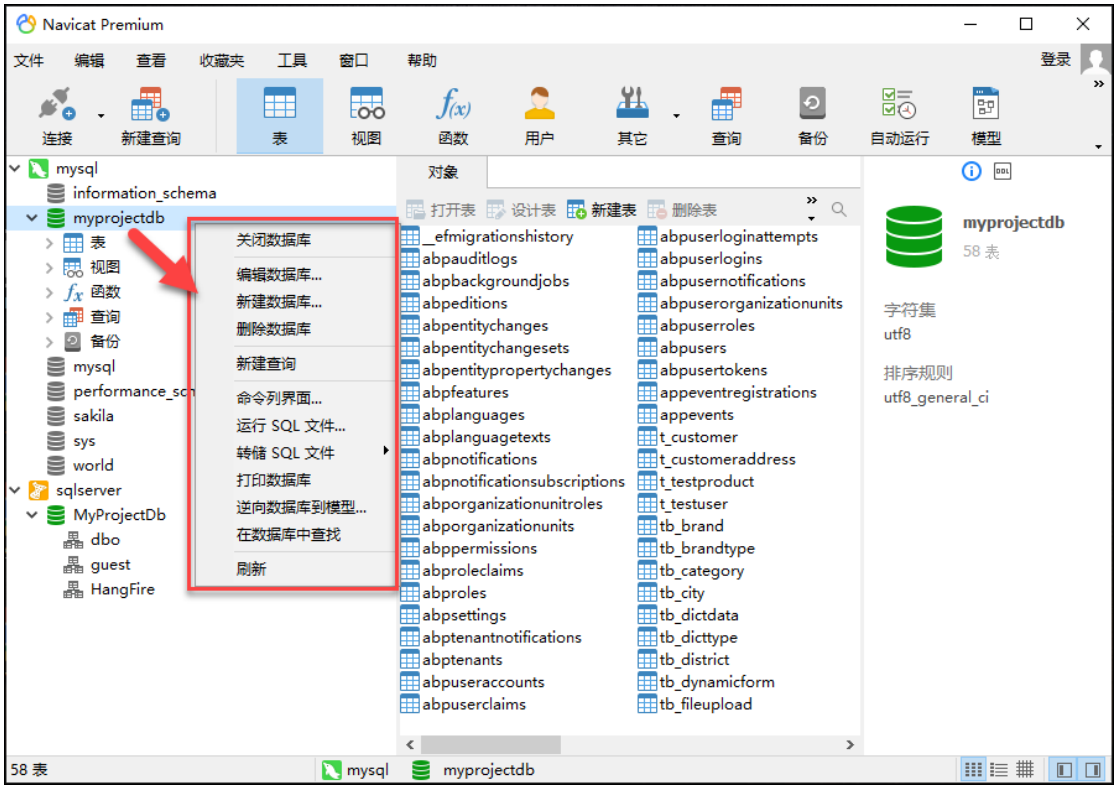


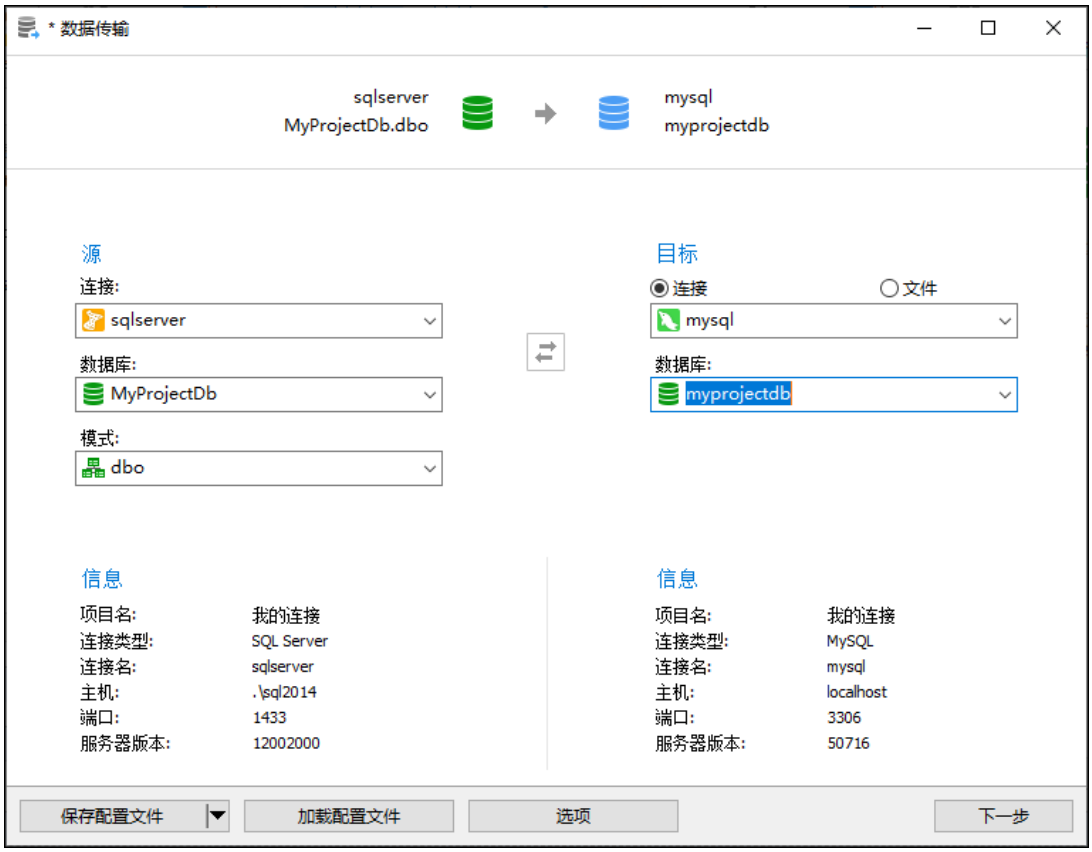
下载安装后，打开运行界面，如果我们往里面添加键值的数据，那么可以看到里面的数据了。



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。


Navigacat Premium
文件 编辑 查看 收藏夹 工具 窗口 帮助
登录
连接 新建查询 表 视图 函数 用户 其它 查询 备份 自动运行 模型
mysql
information_schema
myprojectdb
表
视图
函数
查询
备份
mysql
performance_sch
sakila
sys
world
sqlserver
MyProjectDb
dbo
guest
HangFire
关闭数据库
编辑数据库...
新建数据库...
删除数据库
新建查询
命令列界面...
运行 SQL 文件...
转储 SQL 文件
打印数据库
逆向数据库到模型...
在数据库中查找
刷新
对象
打开表 设计表 新建表 删除表
_efmigrationshistory
abpauditlogs
abpbackgroundjobs
abpeditions
abpentitychanges
abpentitychangesets
abpentitypropertychanges
abpfeatures
abplanguages
abplanguagetexts
abpnotifications
abpnotificationsubscriptions
abporganizationunitroles
abporganizationunits
abppermissions
abproleclaims
abproles
abpsettings
abptenantnotifications
abptenants
abpuseraccounts
abpuserclaims
abpuserloginattempts
abpuserlogins
abpusernotifications
abpuserorganizationunits
abpuserroles
abpusers
abpusertokens
appeventregistrations
appevents
t_customer
t_customeraddress
t_testproduct
t_testuser
tb_brand
tb_brandtype
tb_category
tb_city
tb_dictdata
tb_dicttype
tb_district
tb_dynamicform
tb_fileupload
myprojectdb
58 表
字符集
utf8
排序规则
utf8_general_ci
58 表
mysql
myprojectdb


* 数据传输
sqlserver
MyProjectDb.dbo
mysql
myprojectdb
源
连接:
sqlserver
数据库:
MyProjectDb
模式:
dbo
目标
连接
文件
mysql
数据库:
myprojectdb
信息
项目名: 我的连接
连接类型: SQL Server
连接名: sqlserver
主机: .\sql2014
端口: 1433
服务器版本: 12002000
信息
项目名: 我的连接
连接类型: MySQL
连接名: mysql
主机: localhost
端口: 3306
服务器版本: 50716
保存配置文件
加载配置文件
选项
下一步

5. 产品部署说明

开发框架的后端是基于 Python 的 FastAPI 的后端，遵循 Restful 的 Web API 标准，部署方式直接复制项目在服务上运行即可，运行前需要再服务器安装好 Python 运行环境，以及项目的依赖引用即可；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 FastApi 的 WebAPI 服务端

Web API 的后端内容，采用 基于 Python+FastApi +SqlAlchemy+Pydantic+Redis 的后端框架。Python 项目具有跨平台运行的特性，可以在 Windows、MacOS、Ubuntu 等 Linux 系统上运行。为了正常运行 Python 开发框架的后端 Web API 项目，我们需要在服务器安装好 Python 运行环境，以及项目的依赖引用。

5.1.1. 下载 Python 3 安装包并安装。

Window 平台安装 Python: https://www.python.org/downloads/windows/

MacOS 平台安装 Python: https://www.python.org/downloads/mac-osx/

下载对应版本的 Python 安装程序，根据您的系统是 32 位还是 64 位，选择对应的安装包，一般服务器上使用 64 位的 Python 安装程序，安装的时候记得**勾选 “Add Python to PATH”**（非常重要，避免后续手动配置环境变量）。

对于大多数 Linux 发行版，您可以通过包管理器直接安装 Python。例如，在 Ubuntu 上，您可以使用以下命令：

sudo apt update

sudo apt install python3

MacOS 通常也会自动配置好环境变量。如果需要手动配置，可以编辑 ~/.bash_profile、~/.zshrc 或其他 shell 配置文件，添加 Python 的安装路径到 PATH 变量中。

安装完成后，务必验证 Python 是否成功安装，并检查版本信息是否正确，在命令行中验证 Python 版本是否正常安装：

python --version

在安装库之前，建议将 pip 更新到最新版本：

python -m pip install --upgrade pip

5.1.2. 上传后端项目到服务器中并安装虚拟环境

在 Python 3 中，创建和使用虚拟环境可以帮助你管理项目的依赖，避免系统级别的包和项目之间的冲突。以下是如何在 Python 3 中创建和使用虚拟环境的步骤。

1. **安装 venv 模块（如果尚未安装）**

从 Python 3.3 起，venv 模块已经被包含在 Python 的标准库中。你不需要额外安装任何东西，只需要确保你的 Python 版本是 3.3 或更高。

你可以使用以下命令来检查 Python 版本：

python3 --version

2. **创建虚拟环境**

使用 venv 模块来创建虚拟环境。首先，进入你想要放置项目的目录，然后运行以下命令：

python3 -m venv myenv

其中，myenv 是你想要创建的虚拟环境的名字，你可以根据需要自定义。此命令会在当前目录下创建一个名为 myenv 的子目录，里面包含了虚拟环境的所有文件。

3. **激活虚拟环境**

虚拟环境创建完成后，你需要激活它。

在 Windows 上：

myenv\Scripts\activate

在 macOS 和 Linux 上：

source myenv/bin/activate

当虚拟环境激活时，你会看到命令行提示符前面有 (myenv)，表示当前正在使用 myenv 虚拟环境。

4. **安装依赖**

激活虚拟环境后，你可以使用 pip 来安装项目所需的库和依赖。在项目中，我们通常

会使用 requirements.txt 文件来记录所有依赖包，这样其他人可以通过 pip 安装所有依赖。

pip install -r requirements.txt

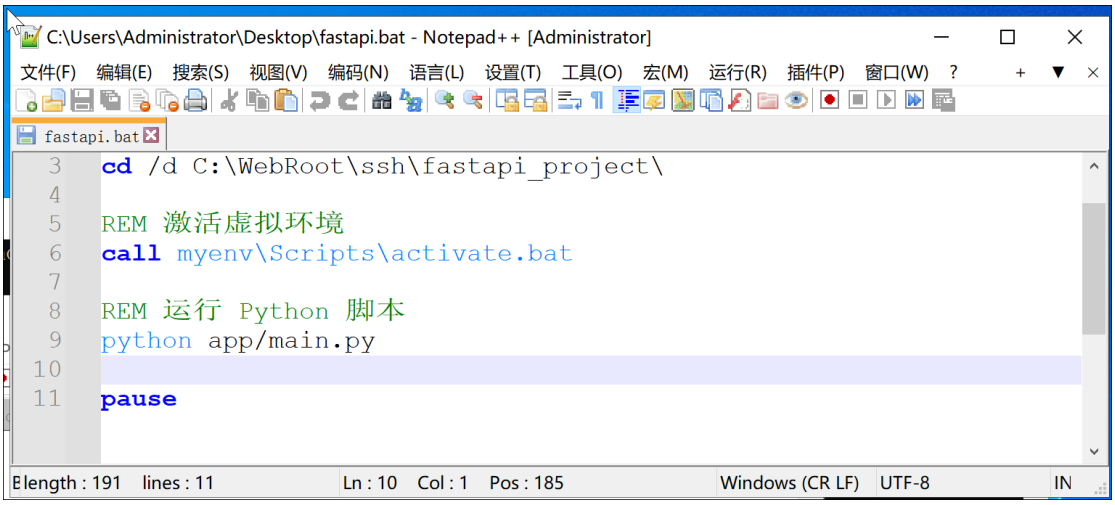
如果顺利安装，那么所有项目的依赖将在虚拟环境上安装完成。

5. 启动项目

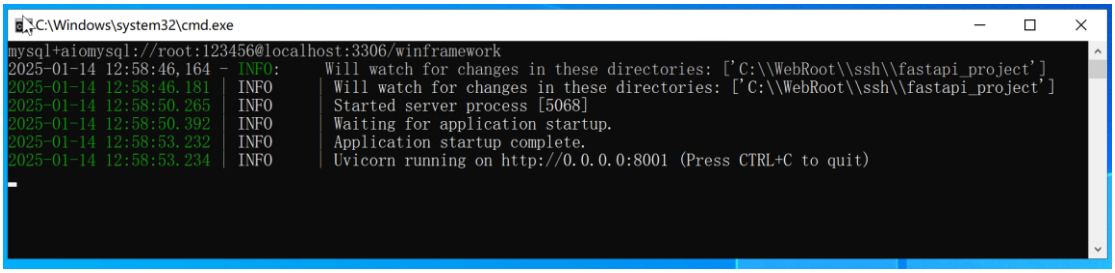
通过上面的步骤完成了虚拟环境和项目引用环境的安装后，就可以通过 Python 命令启动项目的 main.py 入口文件了。

python app/main.py

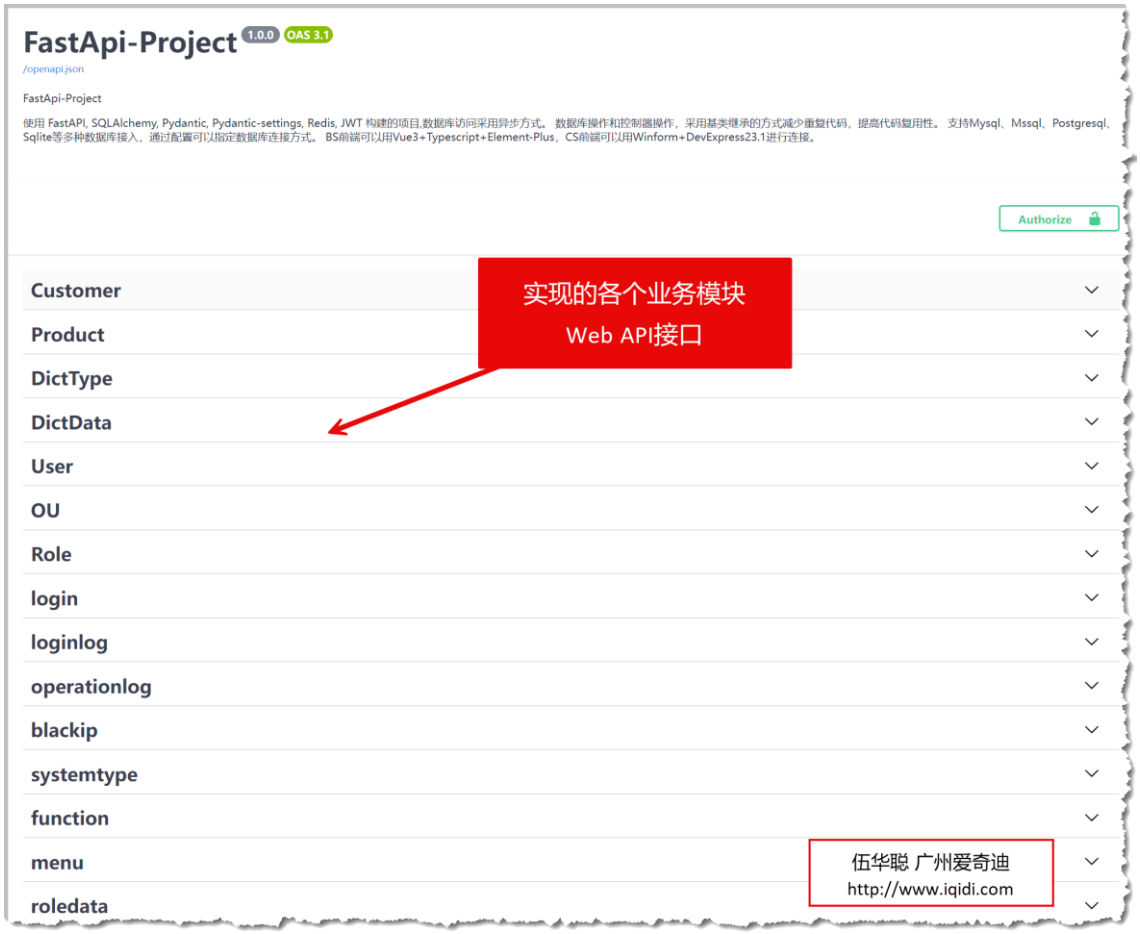
为了方便，在 Window 服务器中我们往往通过编写一个 bat 文件（Window 下批处理文件）来快速启动后端 Web API 项目。



启动项目过程中，可以看到命令行的提示结果。



正常启动项目后，可以看到 WebAPI 主页中有详细的 Swagger 文档介绍，非常方便参考使用。



有了统一 Web API 的后端，我们可以根据需要扩展实现自己的系统业务终端了。

5.1.3. 如何查看和终止正在运行的 Python 进程或端口

如何查看和终止正在运行的 Python 进程：

要查看和终止正在运行的 Python 进程，你可以根据不同的操作系统使用不同的命令和方法。以下是具体步骤：

输入以下命令并执行,查看正在运行的 Python 进程：

Mac/Linux 下命令：

ps -ef | grep python

其中，PID 是进程号，kill 命令用于终止进程。

kill -9 [PID]

windows 下命令：

查看所有 Python 进程：

tasklist | findstr python

结束所有 python 进程，可以使用 taskkill 命令：

taskkill /f /im python.exe

如何查看和终止正在运行的端口

要查看和终止正在运行的端口，你可以使用不同的命令来查找哪些进程占用了哪些端口，并根据需要终止它们。以下是根据不同操作系统的具体方法。

Windows 下命令：

查看正在运行的端口

netstat -ano | findstr :8000

终止正在运行的端口：

taskkill /PID <PID> /F

示例：

taskkill /PID 12345 /F

Mac 下命令：

查看正在运行的端口：

lsof -i:8000

终止正在运行的端口：

kill -9 [进程号]

5.2. 使用 Nginx 部署 Vue+Element 前端应用

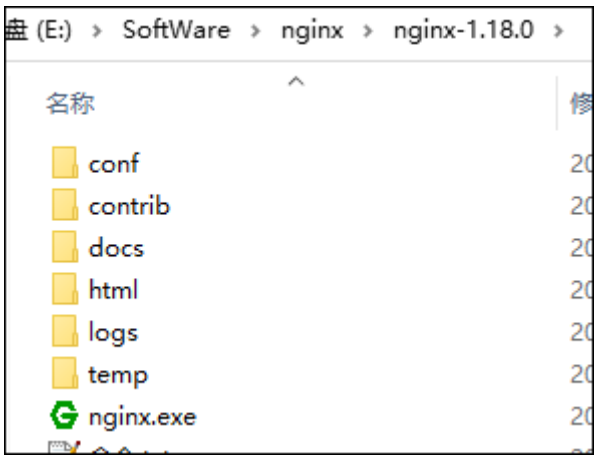
首先使用 **pnpm run build** / **npm run build** 进行 Vue+Element 的项目进行编译，编译成功后在文件项目目录的 dist 目录下。

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
```

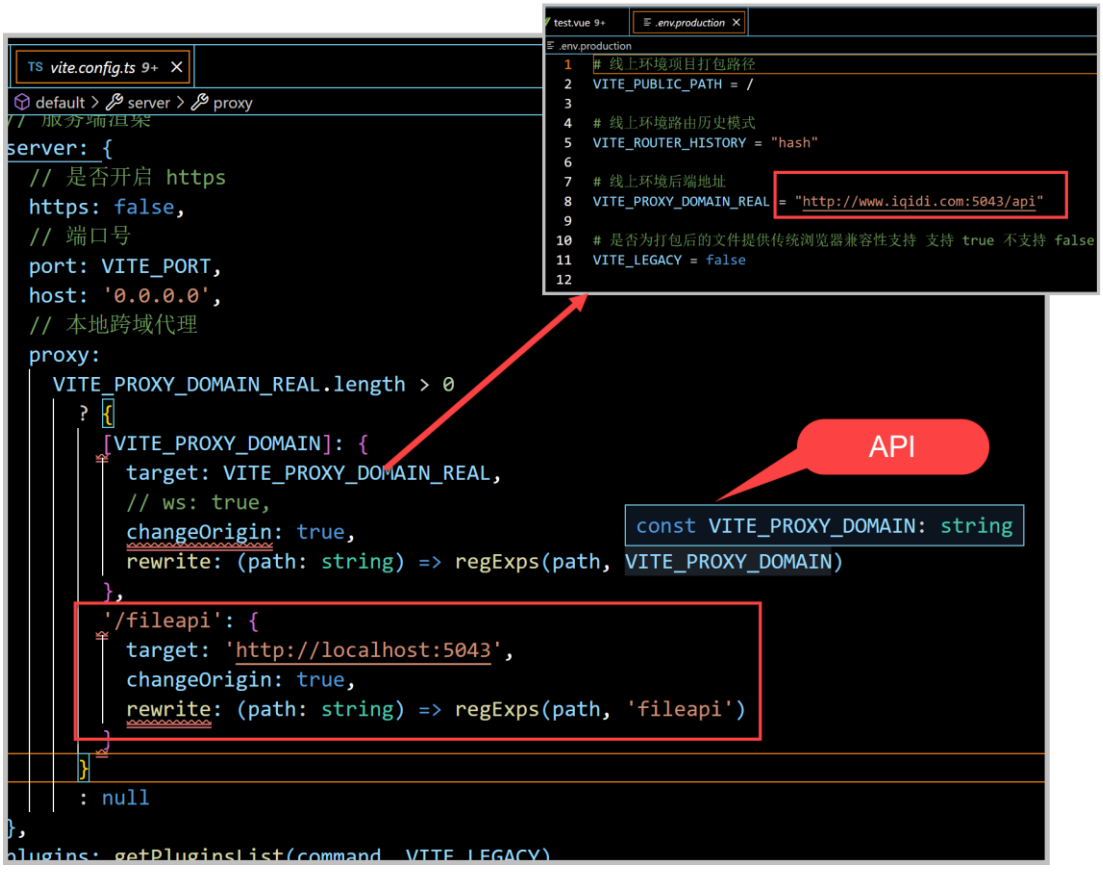
定位到解压的目录，然后在 DOS 窗口中执行 start nginx 启动 nginx 服务。

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

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。



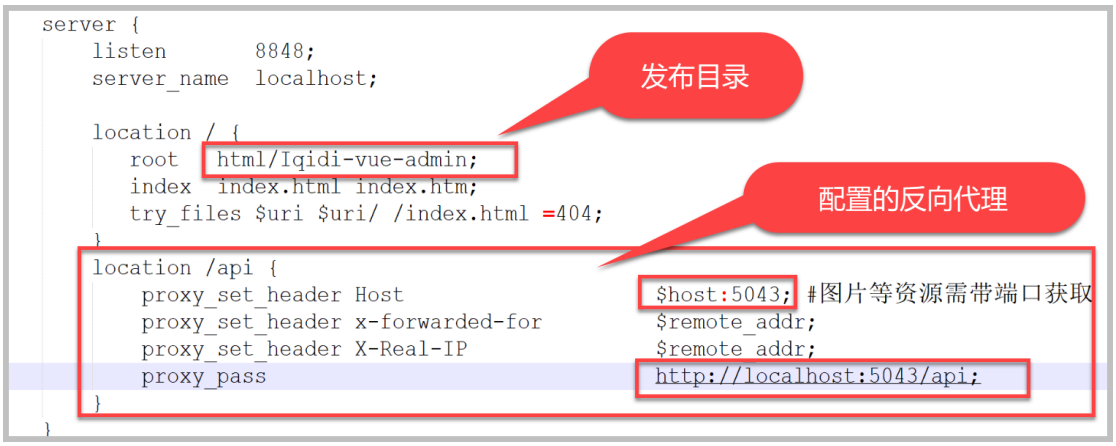
其中的变量 VITE_PROXY_DOMAIN、VITE_PROXY_DOMAIN_REAL 是定义在.env.development 和 .env.production 的变量，这里配置设置了对应的反向代理。

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。

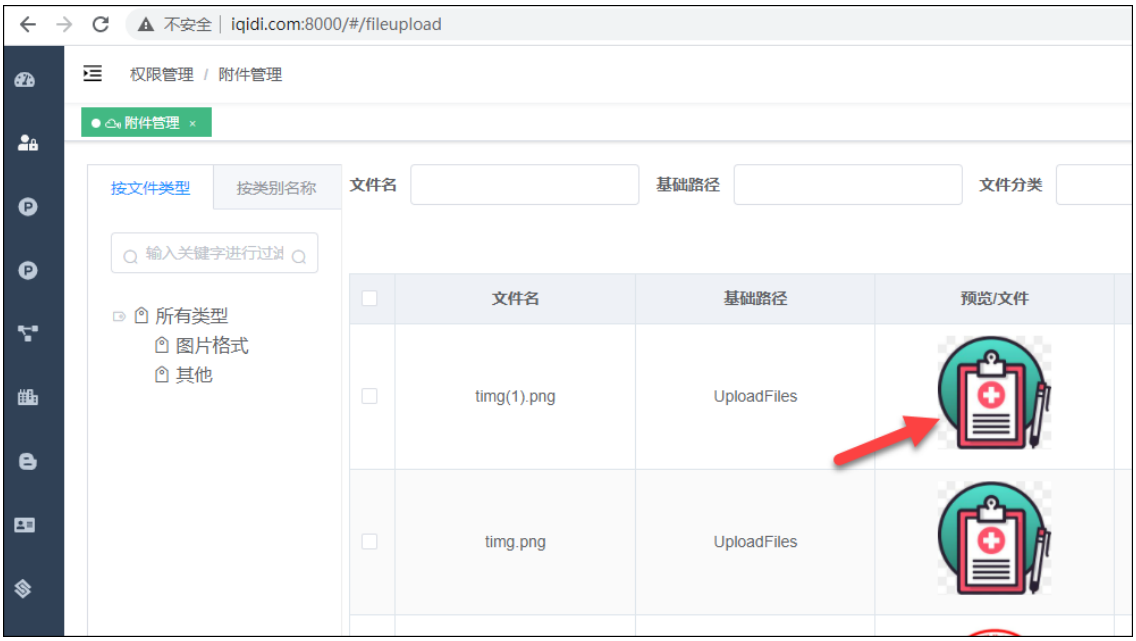


完整设置信息如下所示：

```
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

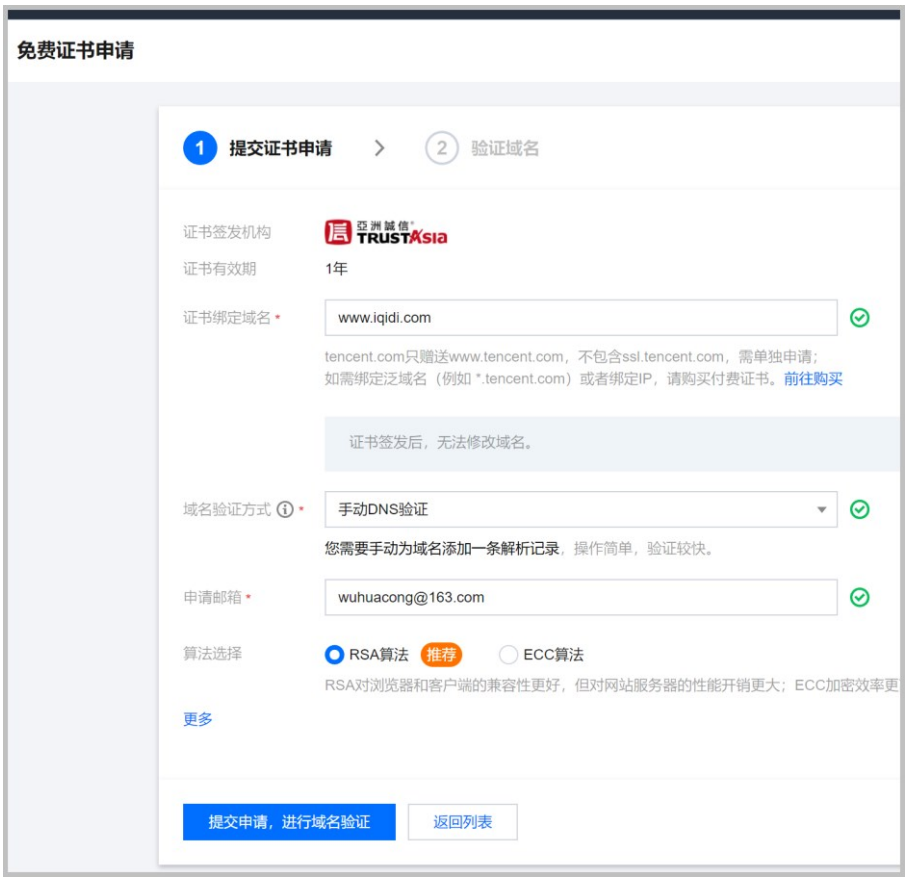
我们如果要在服务器上发布 https 前端应用和 WebAPI 的应用，那么我们就需要用到 https 证书了。我们一般发布的应用的云服务器上，都会提供一定量的相关的免费证书（一般为 20 个）供我们使用，每个一年期限，到期再续即可，一般情况下基本上满足要求了，该小节介绍如何基于云服务提供商的免费证书，在服务器上发布 Nginx 的前端应用和基于 IIS 的 Web API 接口的 https 应用处理。

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。

而腾讯云上的申请入口也是类似，如下界面所示。



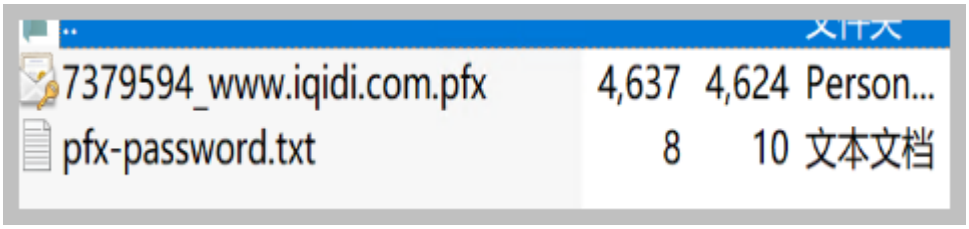
申请成功后，在列表中就可以看到下载 SSL 证书的信息了。如下所示。

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

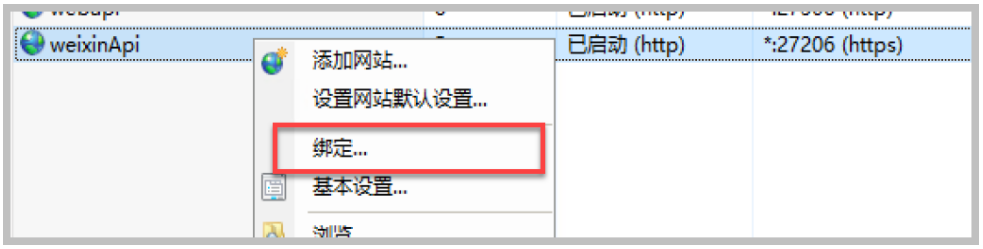
如我们先下载 IIS 的证书文件，我们可以看到除了证书文件，还有一个附带的文本文件，是证书的密码信息。



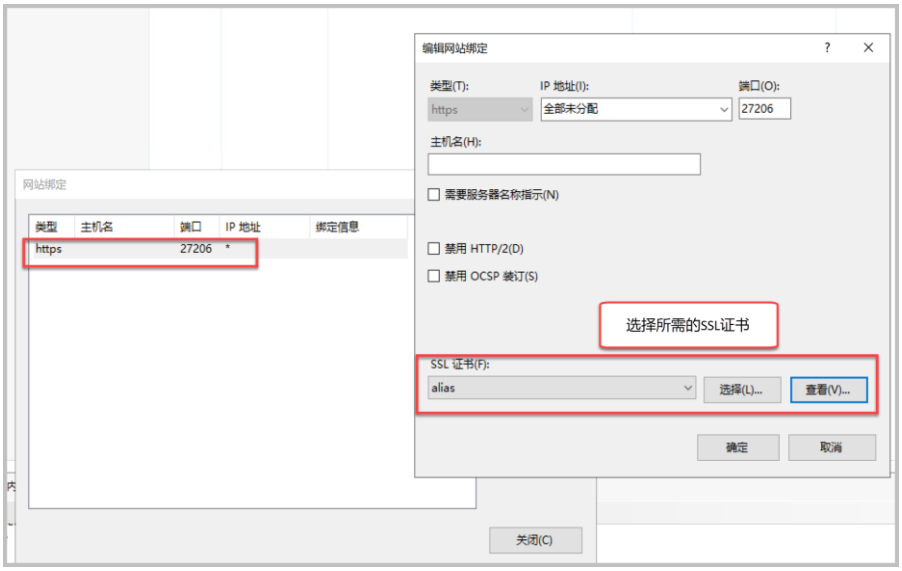
我们双击进行证书的安装，选择本地计算机的存储位置即可。

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。