# 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 | 2020-12-14 | 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 | 《ABP 快速开发框架介绍.pptx》 | | 内部 |
| 2 | 《伍华聪的博客》(http://wuhuacong.cnblogs.com ) | | 博客园 |
| 3 | 《循序渐进 VUE-Element 前端应用开发说明书.doc》 | | 内部 |
| 4 | | | 内部 |

## 1.4. 术语与缩写

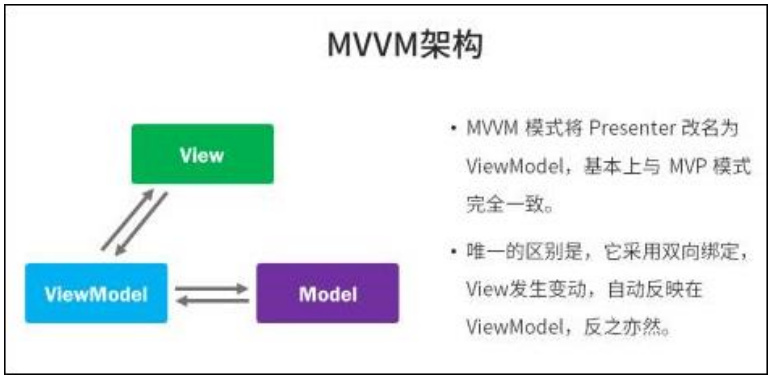
1 在本文件中出现的“系统”、“框架”一词，除非特别说明，均适用于《ABP开发框架》。

2 当前基于.netcore的版本为5.0，如无特殊说明，均指该版本。

# 2. 基础知识介绍

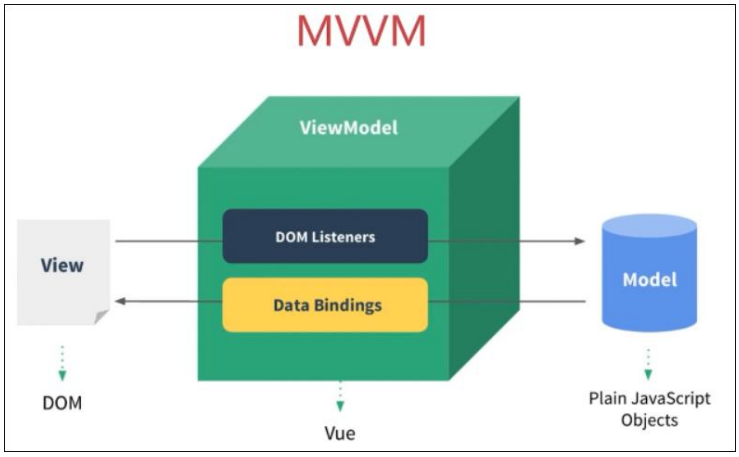
## 2.1. Vue 框架简介

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

## 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 2.0 的桌面端组件库。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。

# 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

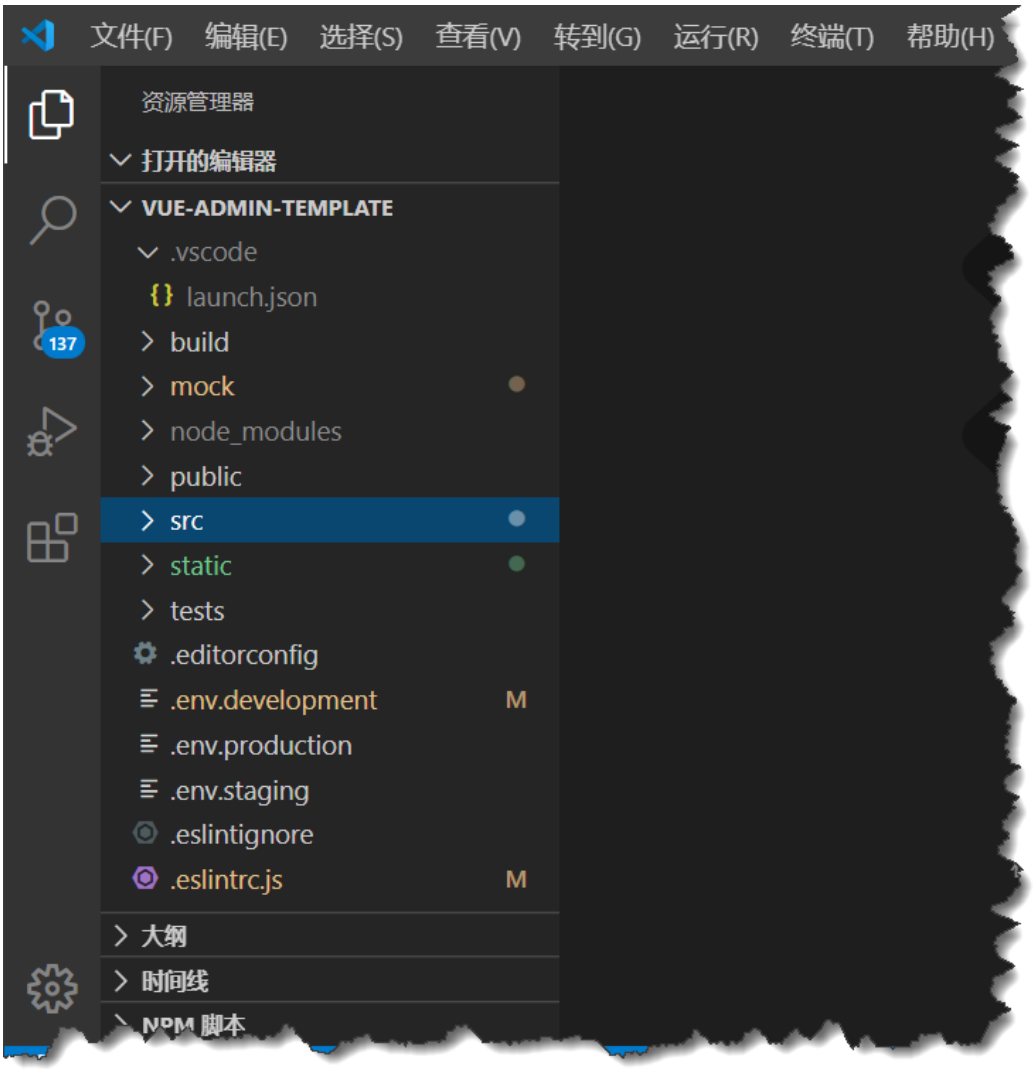
VS Code（Visual Studio Code）是由微软研发的一款免费、开源的跨平台文本（代码）编辑器。几乎完美的编辑器。

官网：https://code.visualstudio.com

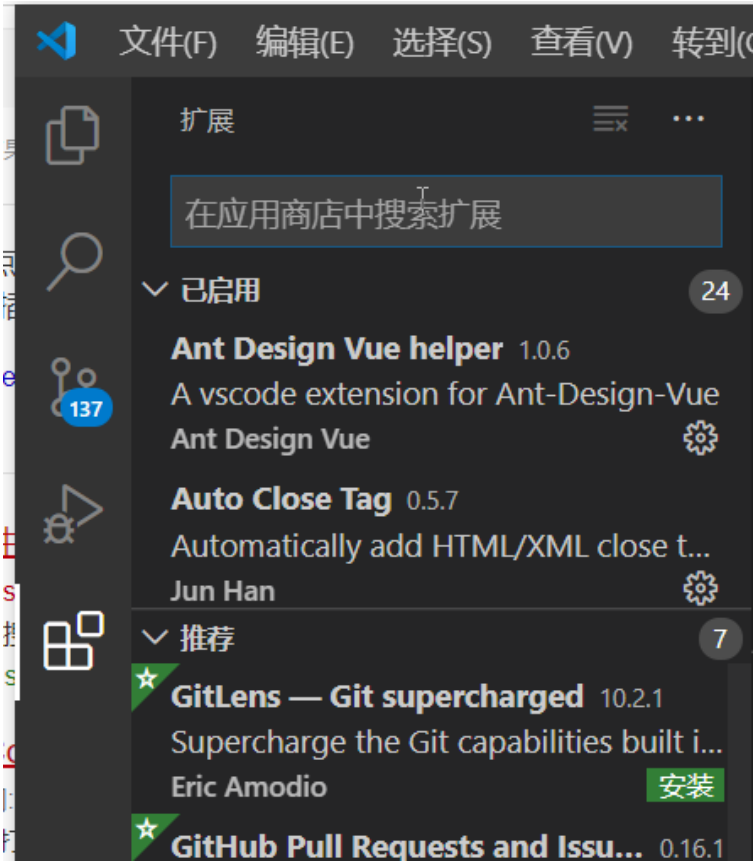
文档：https://code.visualstudio.com/docs

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:

**Vetur**

Vue 多功能集成插件，包括：语法高亮，智能提示，emmet，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

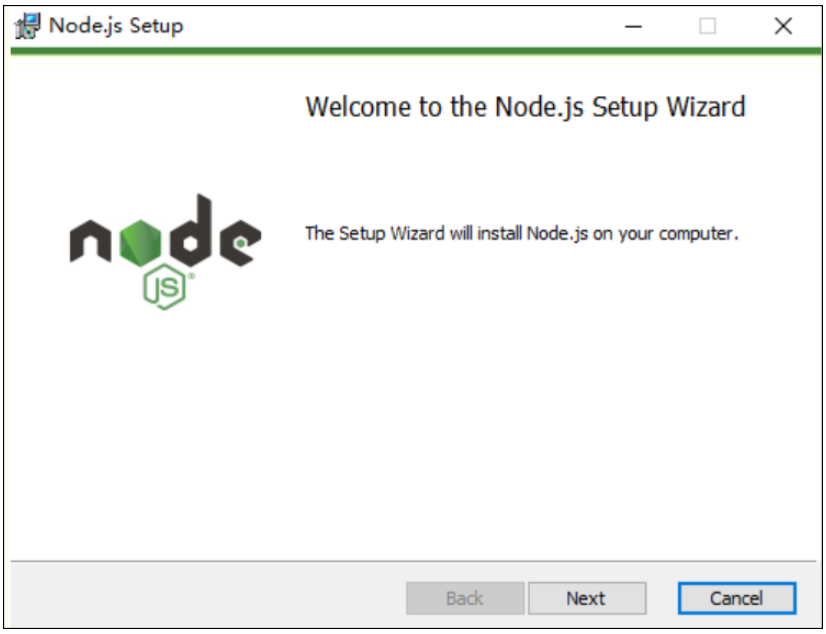
利用 VS Code 开发，我们很多时候，需要使用命令行 npm 进行相关模块的安装，这些需要 node 环境的支持，安装好 node 后，npm 也就一起安装好了。

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

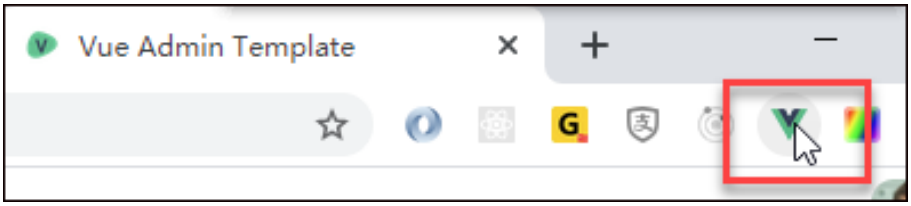
Vue (读音 /vju:/，类似于 view) 是一套用于构建用户界面的渐进式框架。

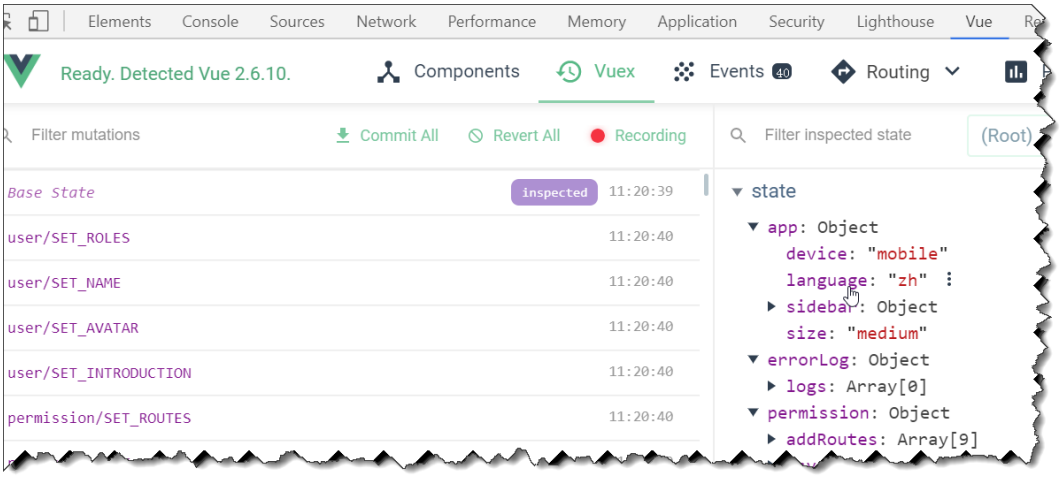
全局安装：npm install vue-cli -g （全局卸载：npm uninstall vue-cli -g）

## 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

对于 **Vetur** 等代码自动修正处理，我们需要在 VS Code 里面进行设置好，在【文件】【首选项】【设置】中，然后单击 Settings.json 进行编辑即可。

设置 { } settings.json ×

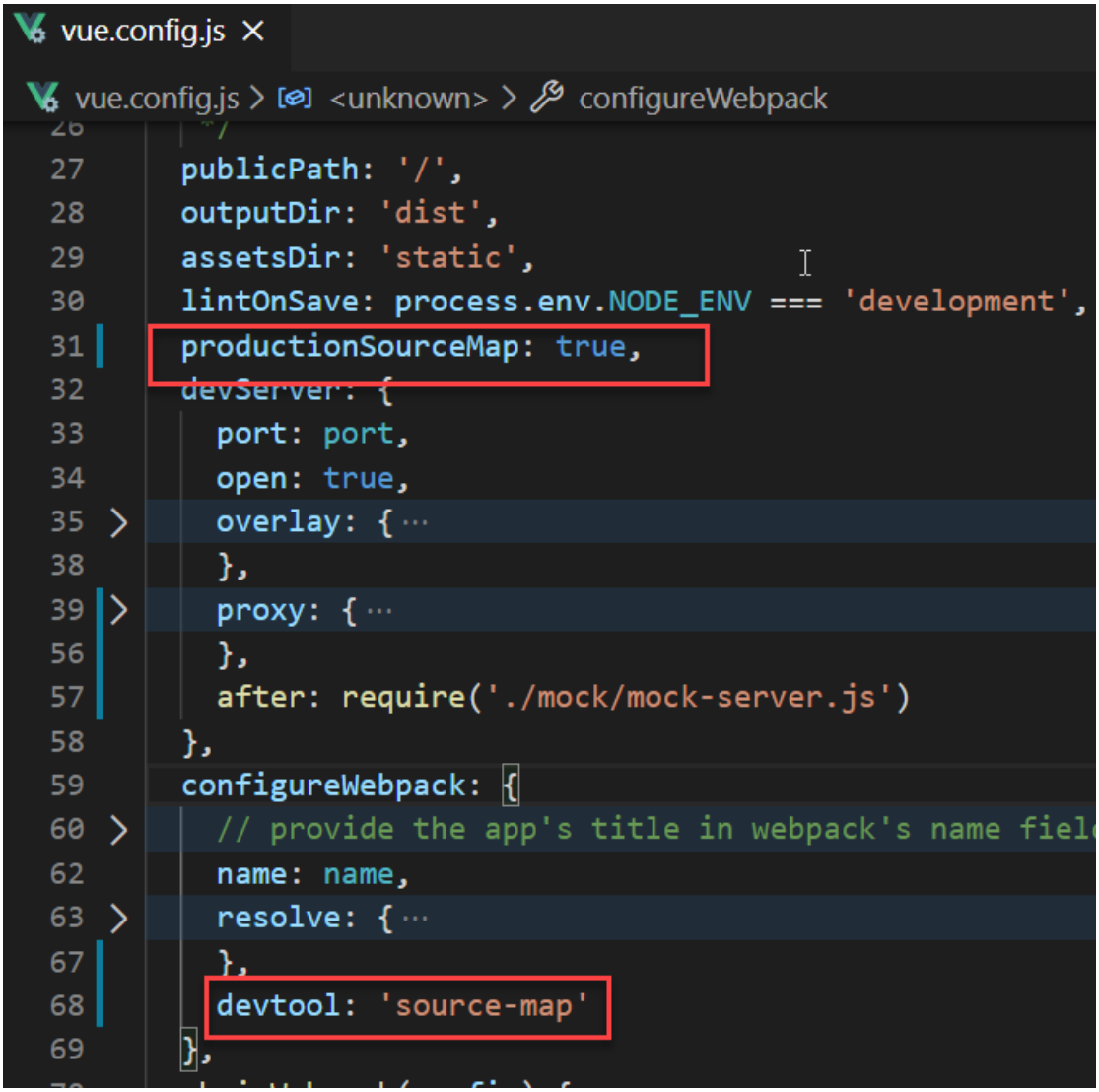
C: > Users > Administrator > AppData > Roaming > Code > User > { } settings.json > { } vetu

```
    // 强制单引号
    "prettier.singleQuote": true,
    // 尽可能控制尾随逗号的打印
    "prettier.trailingComma": "all",
    // 开启 eslint 支持
    "prettier.eslintIntegration": true,
    // 使用插件格式化 html
    "vetur.format.defaultFormatter.html": "js-beautify-html",
    // 格式化插件的配置
    "vetur.format.defaultFormatterOptions": {
        "js-beautify-html": {
            // 属性强制折行对齐
            "wrap_attributes": "force-aligned",
        }
    },
    "editor.codeActionsOnSave": {
        "source.fixAll.eslint": true,
    },
    "editor.quickSuggestions": {
        "strings": true
    },
```

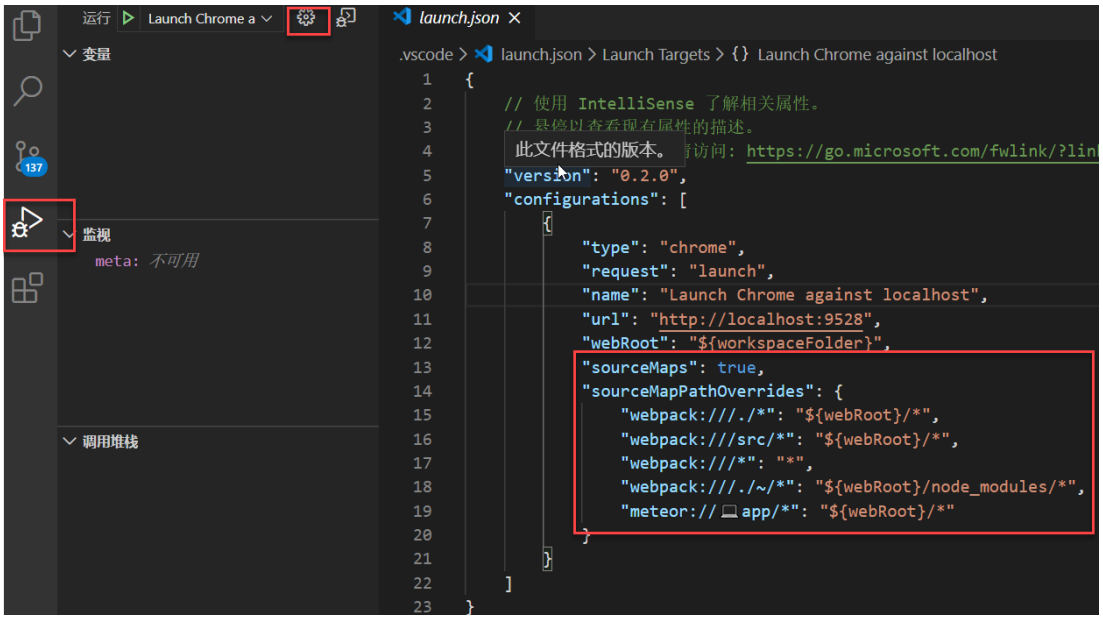
我这里主要设置保存代码后能够对代码进行缩进排版的常规的处理

调试环境的处理，为了结合 Chrome 调试 VScode，我们需要安装插件 Debugger for Chrome ，然后进行 Vue 项目代码的设置处理即可。

打开项目根目录的 Vue.Config.js 文件，在合适的位置，加入 productionSourceMap: true 以及 devtool: 'source-map' 如下所示

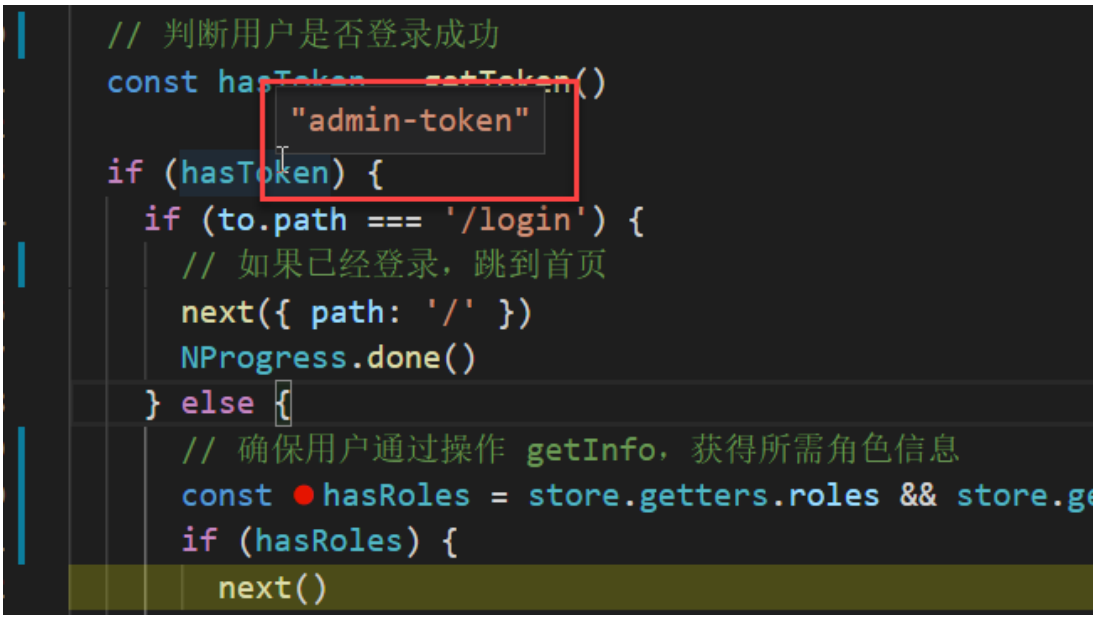



然后再在运行面板里面，进行调试参数设置的处理，如下所示




指定这些设置后，我们就可以以调试模式进行调试 VS Code 里面的代码了，代码只需要设置对应的断点即可跟踪对象的数据。

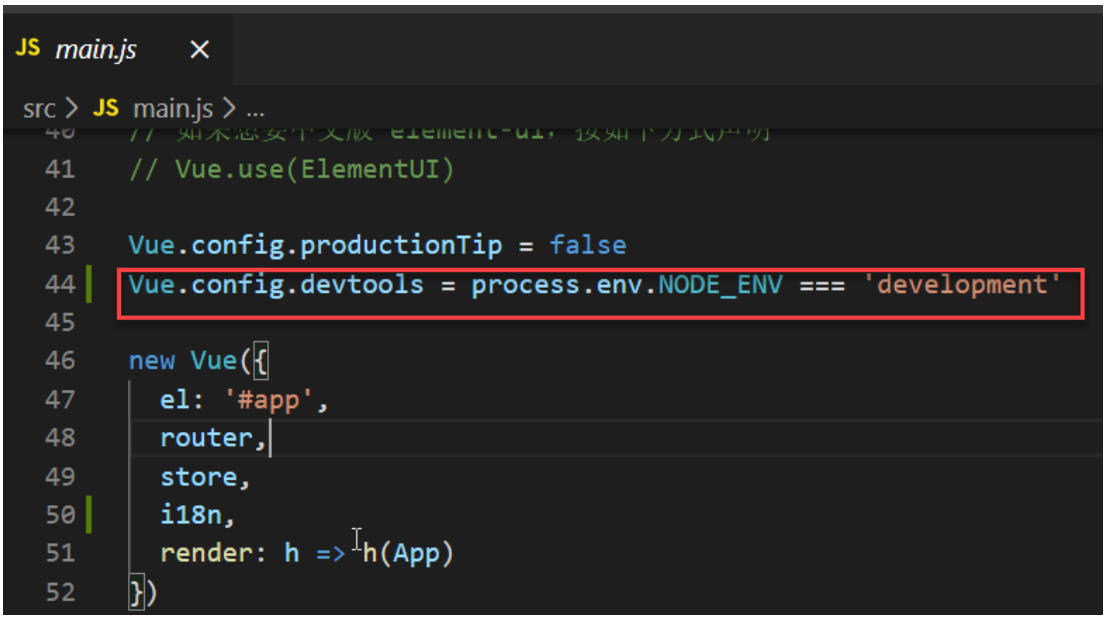
调试前，记得先使用 npm run dev 启动项目，项目完全启动后会在 Chrome 浏览器打开项目地址，再使用 F5 进行项目代码的调试。

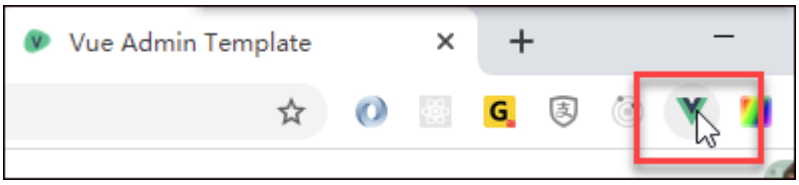


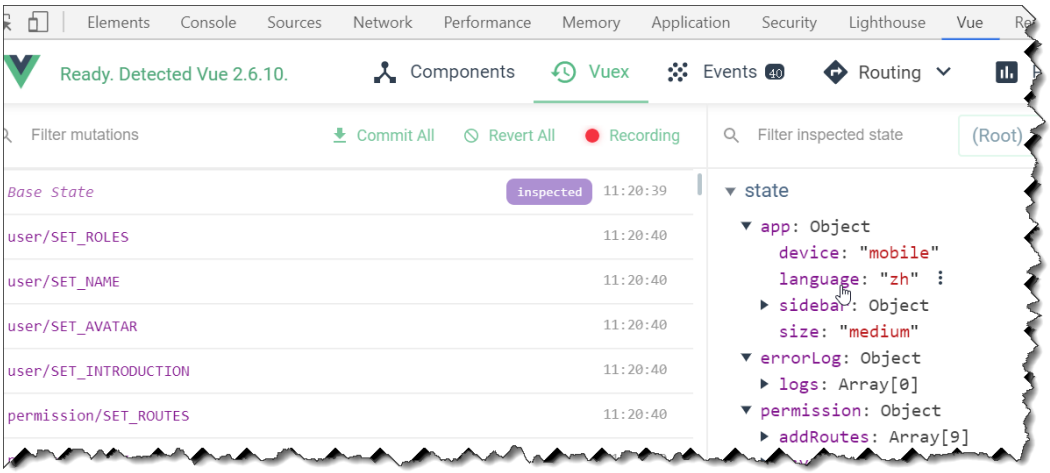

Vue DevTools 也是用来跟踪 Vue 项目路由、状态等信息的，可以信息很好的跟踪处理。为了点亮 Chrome 浏览器上面 Vue DevTools 图标，我们可以在 Vue 项目的 main.js 里面加入一行代码。

Vue.config.devtools = process.env.NODE_ENV === 'development'

如下界面所示




## 4. 后端 ABP 框架开发所需环境

后端 ABP 框架是基于 C# 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 部分则采用.net core5.0 框架，且 ABP 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 ABP 框架后端 Web API 是基于.NET Core5.0 的框架应用，开发工具一般采用 VS Studio2019 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/ ，默认下载社区版本即可。

SQLServer 2014 则自行获取地址进行下载。

ABP 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。

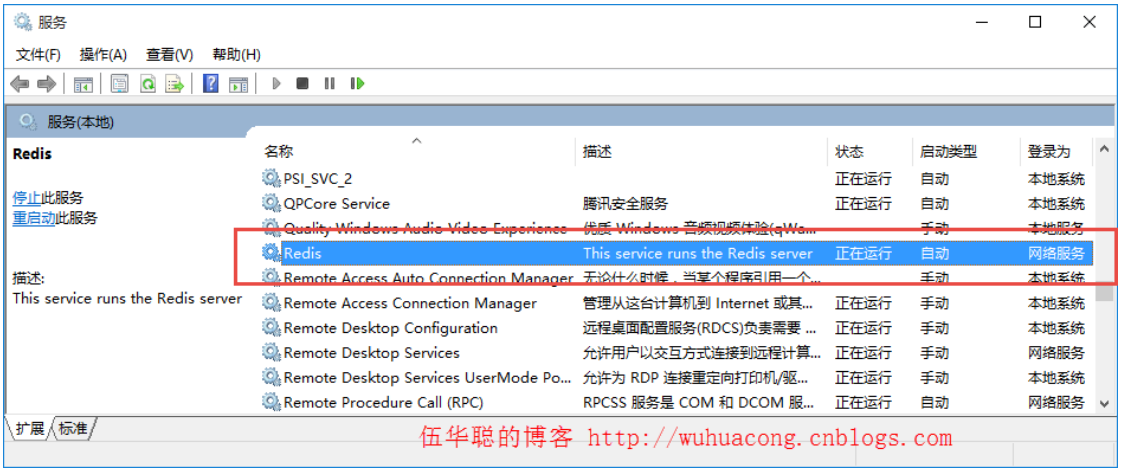
appendfsync no/always/everysec ，appendfsync 配置，no 表示等操作系统进行数据缓存同步到磁盘，always 表示每次更新操作后手动调用 fsync()将数据写到磁盘，everysec 表示每秒同步一次。

### 4）Redis 的安装

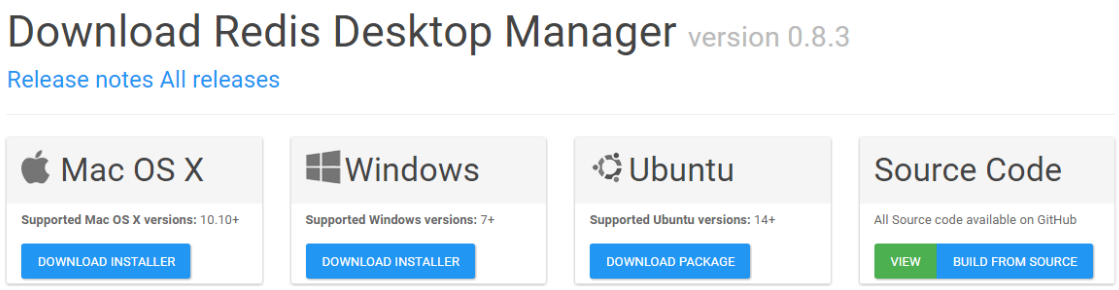
Redis 可以在不同的平台运行，不过我主要基于 Windows 进行开发工作，所以下面主要是基于 Windows 平台进行介绍。

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。

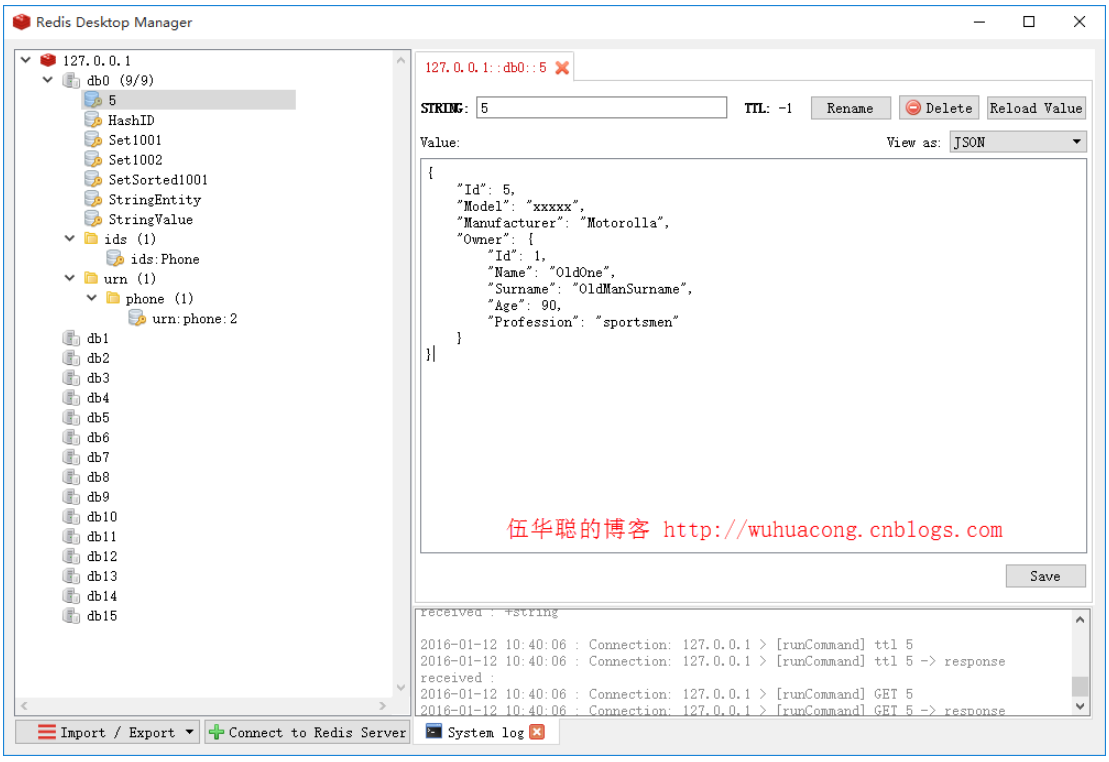
当前可以下载到最新的 Windows 安装版本为 3.0，安装后作为 Windows 服务运行，安装后可以在系统的服务里面看到 Redis 的服务在运行了，如下图所示。




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

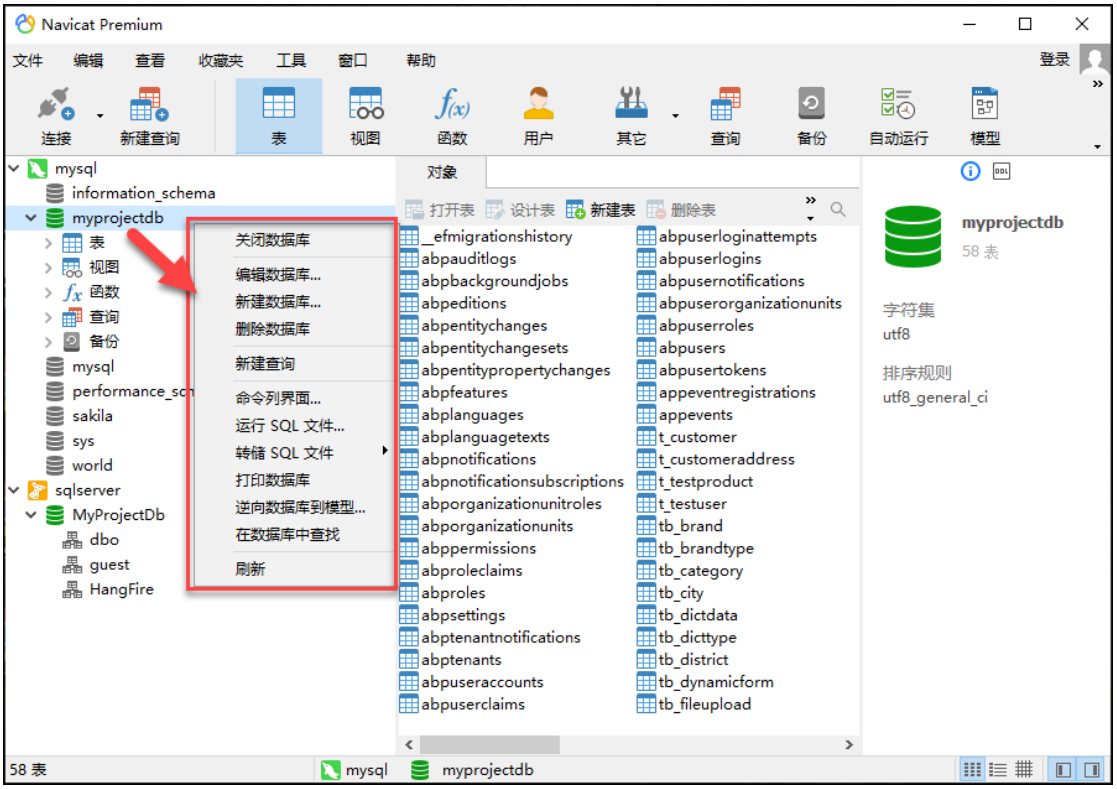


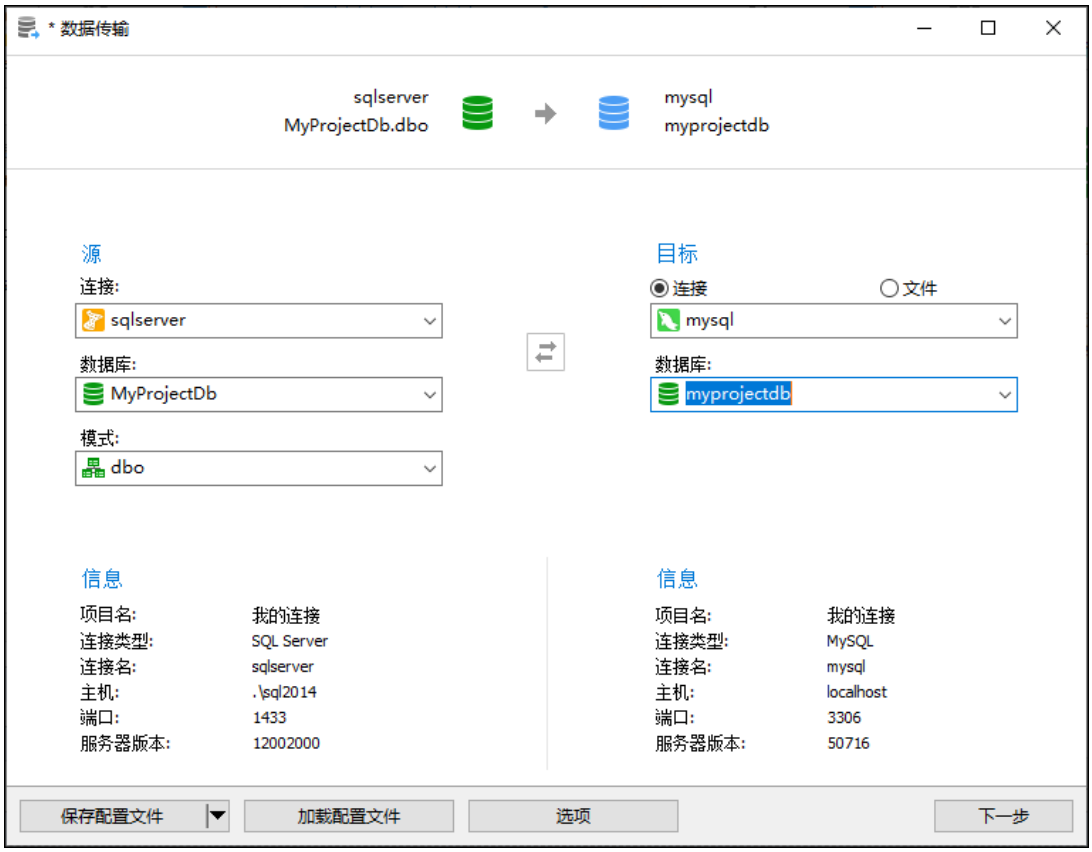

下载安装后，打开运行界面，如果我们往里面添加键值的数据，那么可以看到里面的数据了。




## 4.2. MySQL 数据库及管理工具

ABP 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。


Navigator 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

## 4.3. Winform 前端开发组件

ABP 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 以及补丁文件处理即可。

# 5. 产品部署说明

ABP 框架的后端是基于.net core5 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。

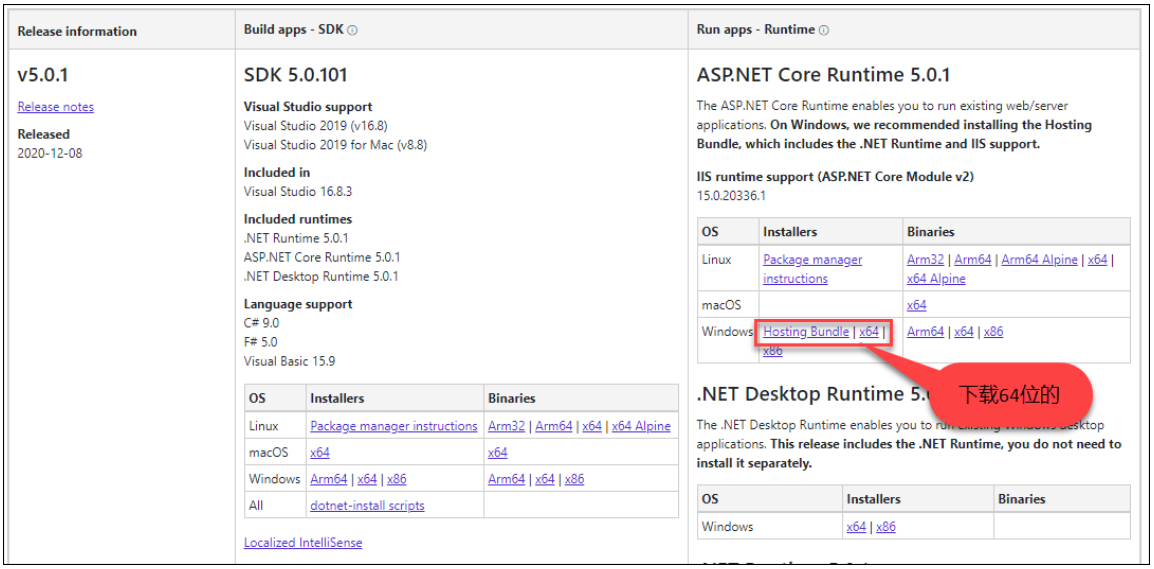
## 5.1. 部署基于.netcore 的 ABP 框架后台 Api 服务端

### 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 ABP 的 Web API 是基于.net core 5 的，因此，我们打开.net core 5 的页面：

https://dotnet.microsoft.com/download/dotnet/5.0

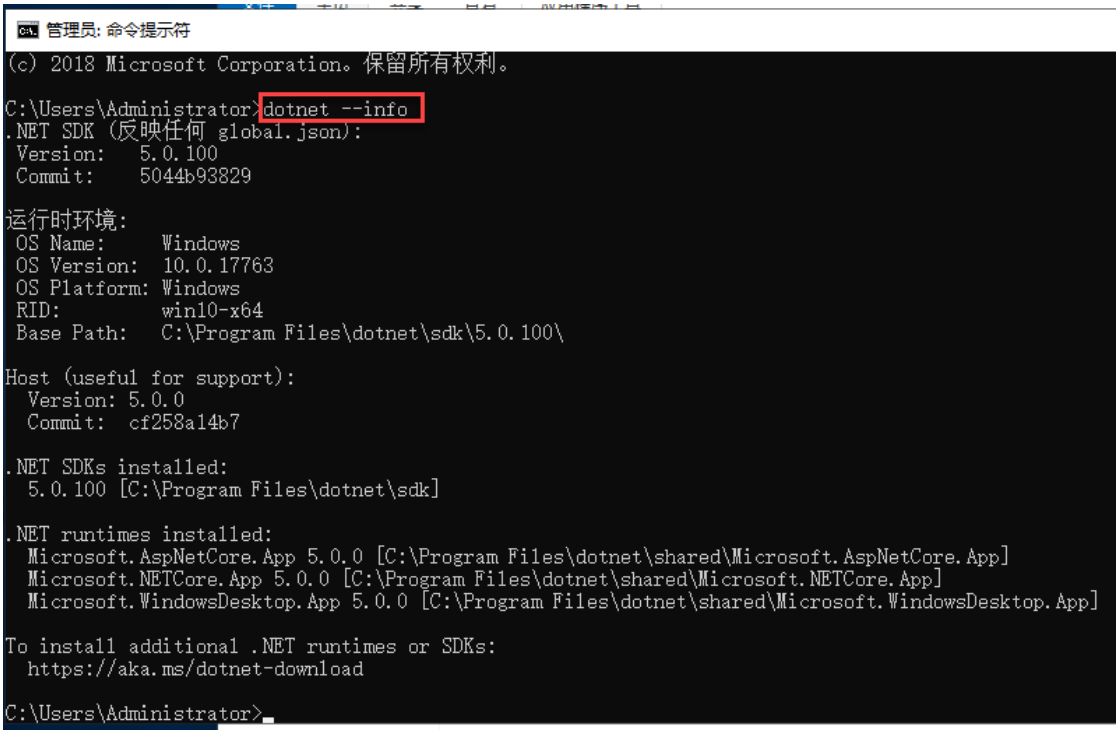



在这里选择下载 Hosting Bundle。因为 Hosting Bundle 包括了.Net Core 运行时和 IIS 支持。下载完成以后，双击 exe 文件即可进行安装即可。


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：

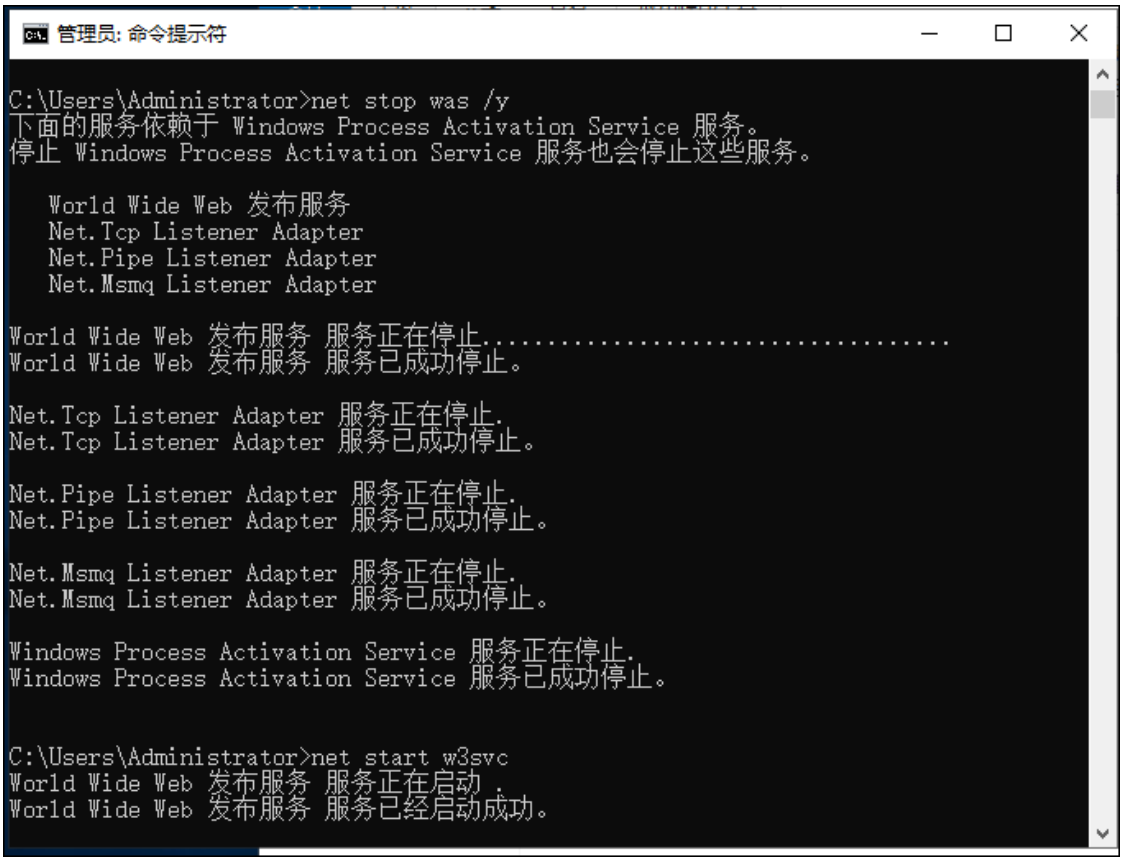



可以看到提示我们已经安装了.NET Core runtimes 环境等所需软件。

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
```

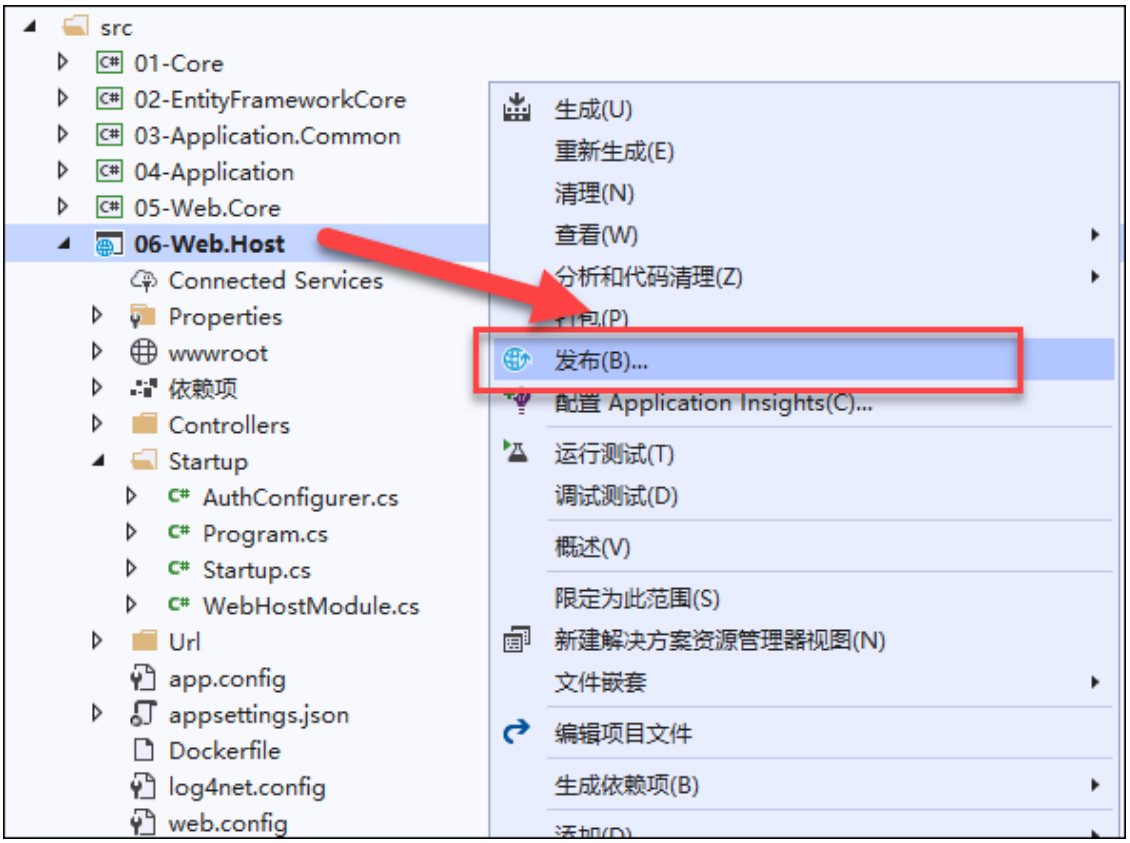
如下图所示：



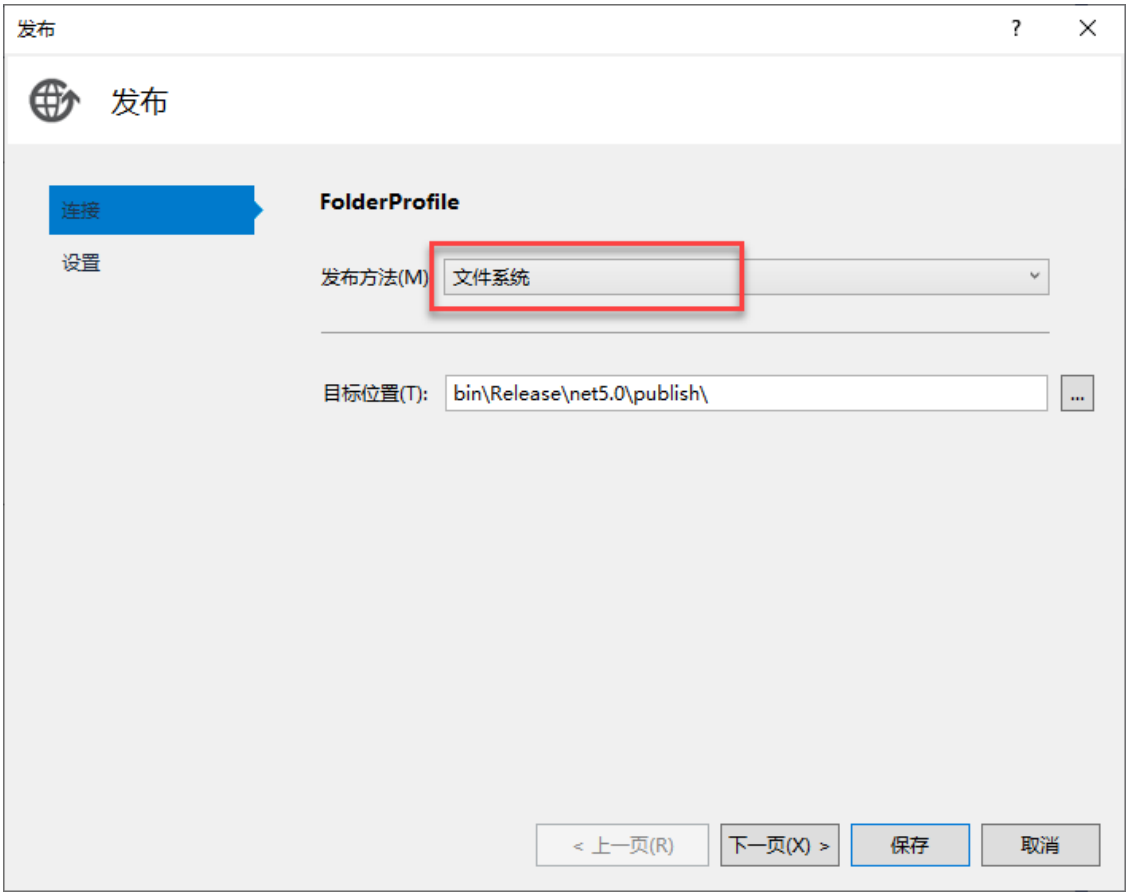

也可以在管理服务器里面重启启动 IIS。

### 5.1.2. 发布.net core 项目

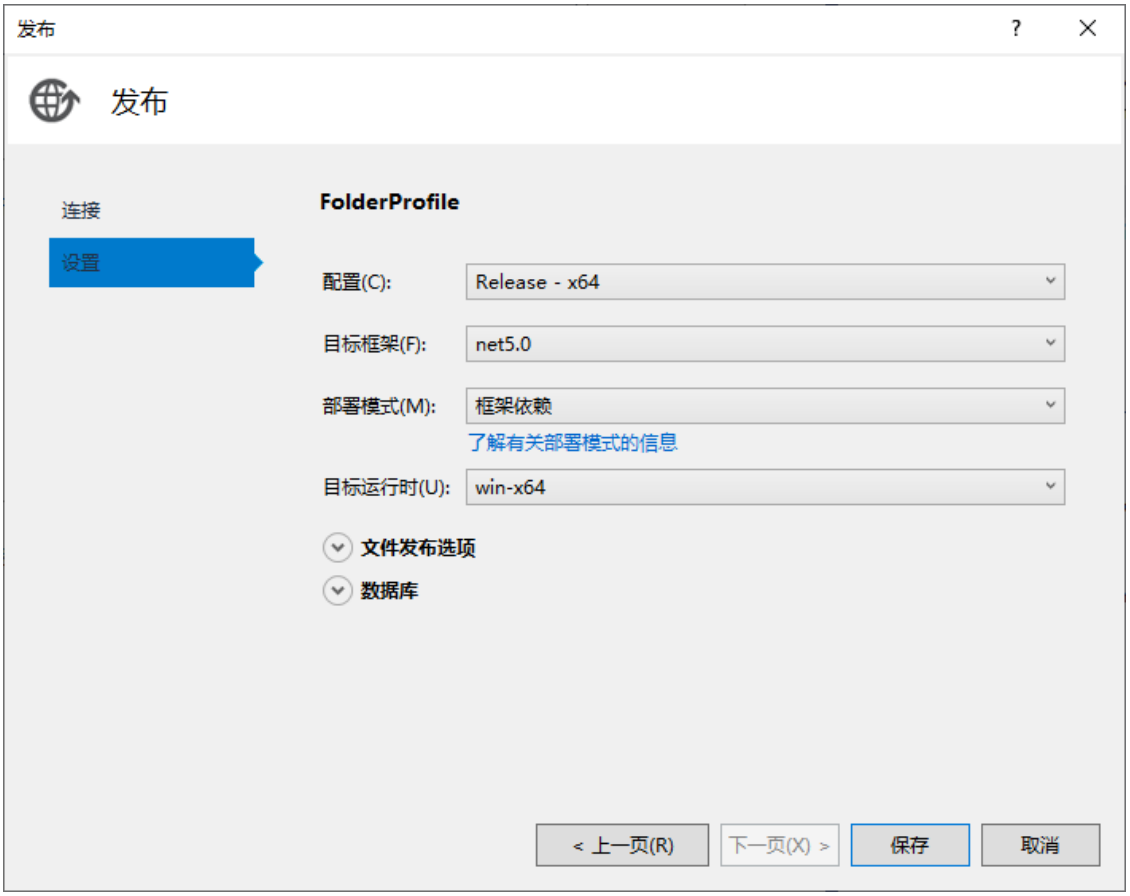
服务器.net core 环境弄好后，下一步就是准备好发布包，我们在 ABP 框架的 Host 项目进行发布。




发布选择文件发布，如下所示。


发布
发布
连接
设置
FolderProfile
发布方法(M)
文件系统
目标位置(T):
bin\Release\net5.0\publish\
...
< 上一页(R)
下一页(X) >
保存
取消

然后调整设置如下所示。



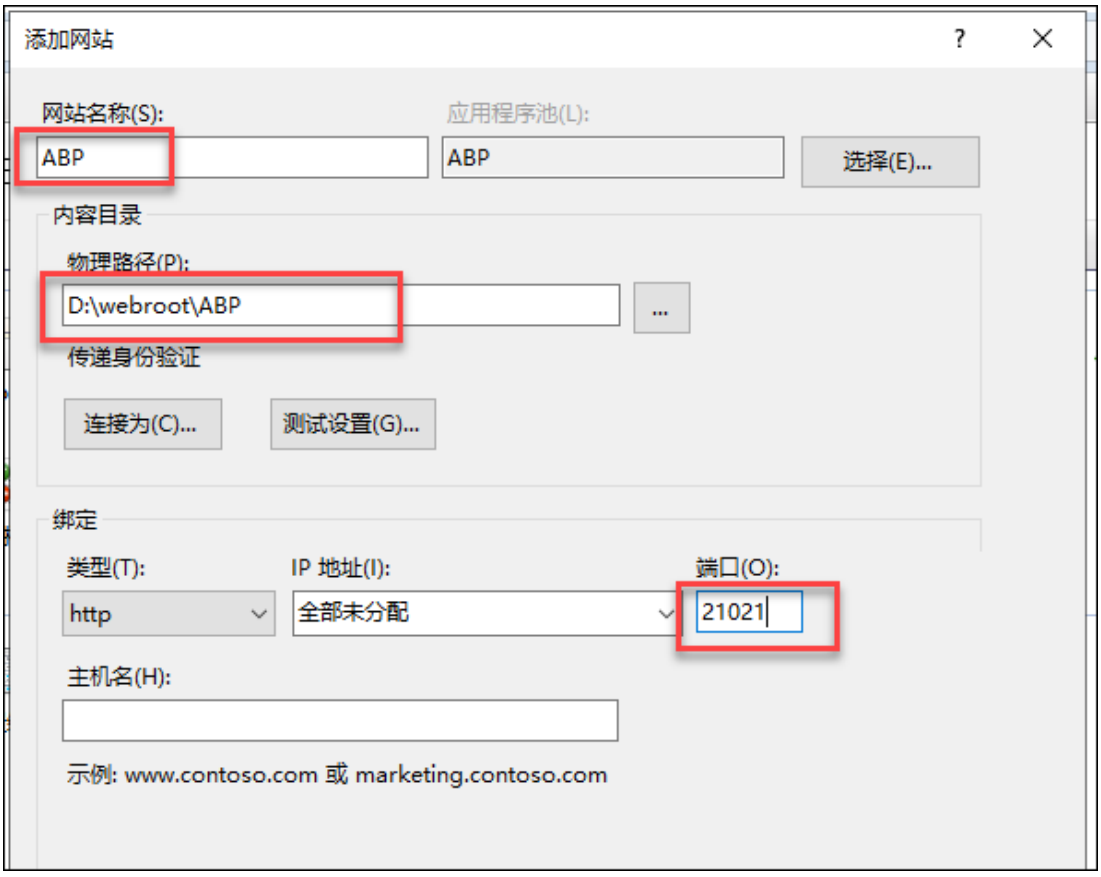

最后我们生成发布包，在对应的目录下，如下所示。

**G:***\Web.Host\bin\Release\net5.0\publish**

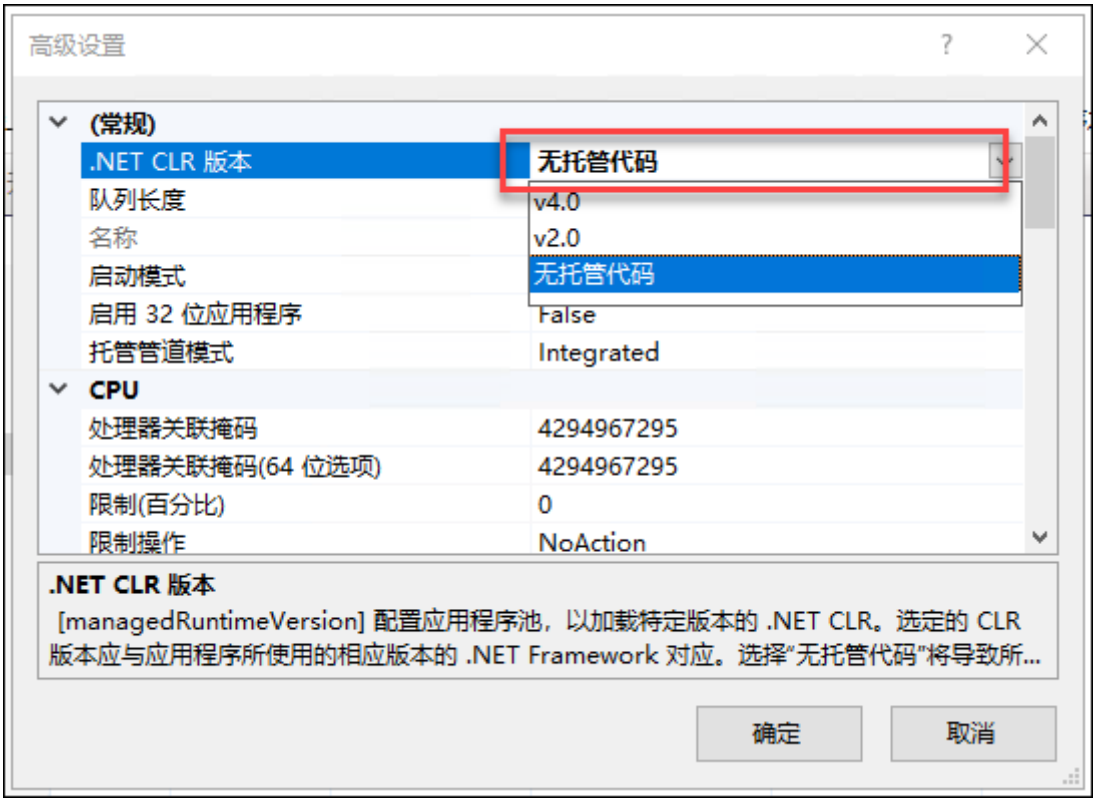
### 5.1.3. 在服务器中设置 IIS

把文件上传到服务器上，然后就是准备设置好 IIS 了。

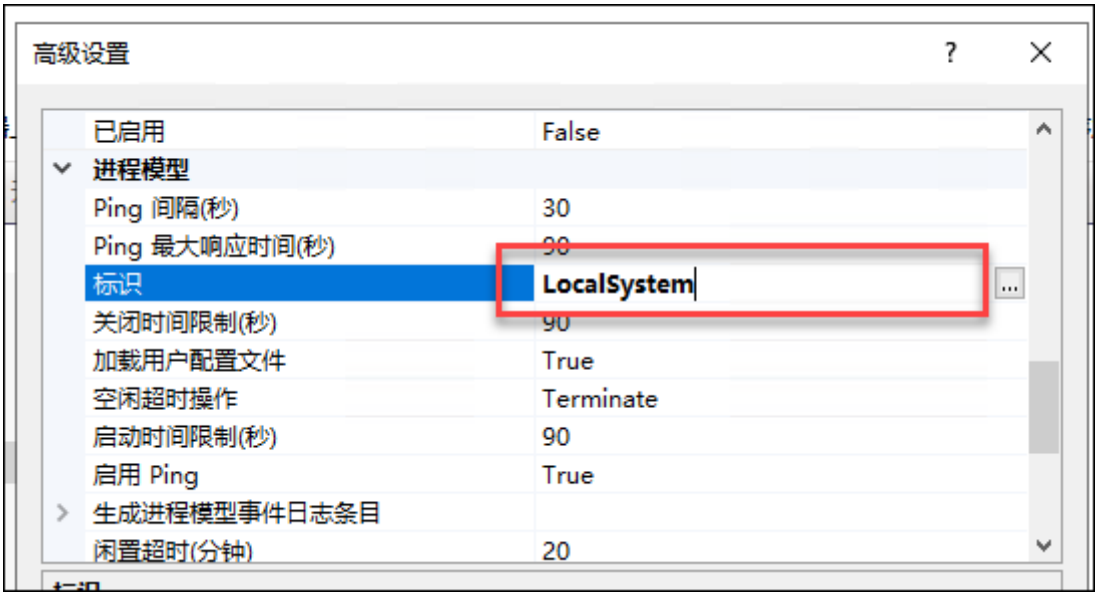
先在 IIS 服务器上创建一个网站，指定对应目录和端口等信息，如下所示。




然后找到对应的应用程序池，找到刚才创建的 ABP 应用程序池。设置.net clr 版本为**【无托管代码】**




在其中把标识由 **ApplicationPoolIdentity** 修改为 **LocalSystem**，以提供应用权限可以访问数据库。




最后点击【确定】按钮，网站及部署完成。我们就可以在浏览器里面进行浏览了。

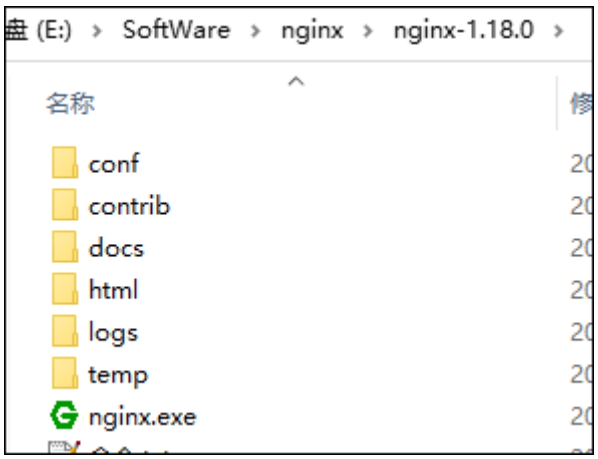

顺利弄完 asp.net core 的后端 API 服务，那么下面就需要同时把 Vue+Element 的前端部署在服务端了。

## 5.2. 使用 Nginx 部署 Vue+Element 前端应用

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 (engine x) 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。

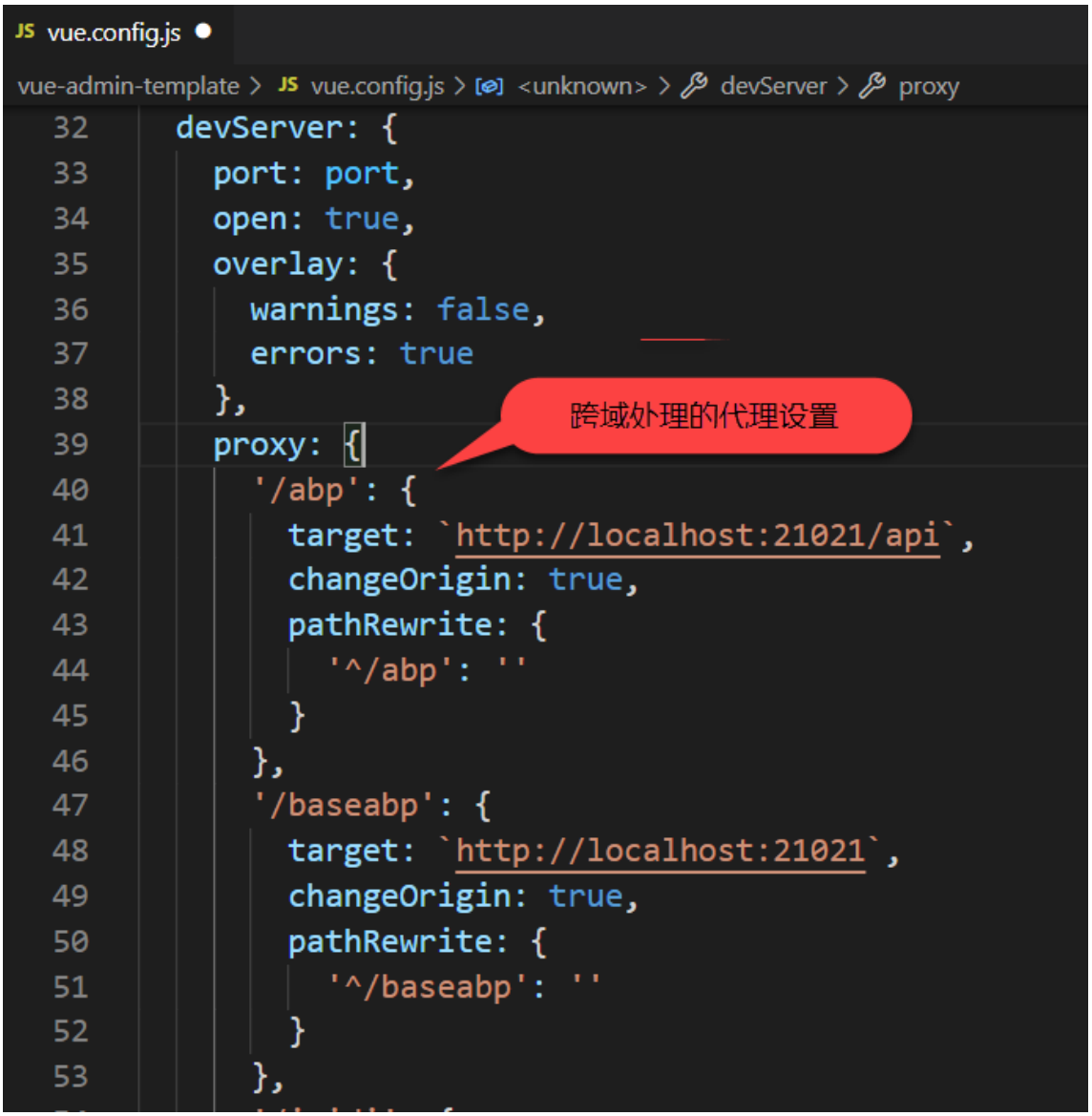
在使用前，我们需要检查 nginx 是否启动成功，直接在浏览器地址栏输入网址 http://localhost:80，回车，出现以下页面说明启动成功。


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

vue.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。

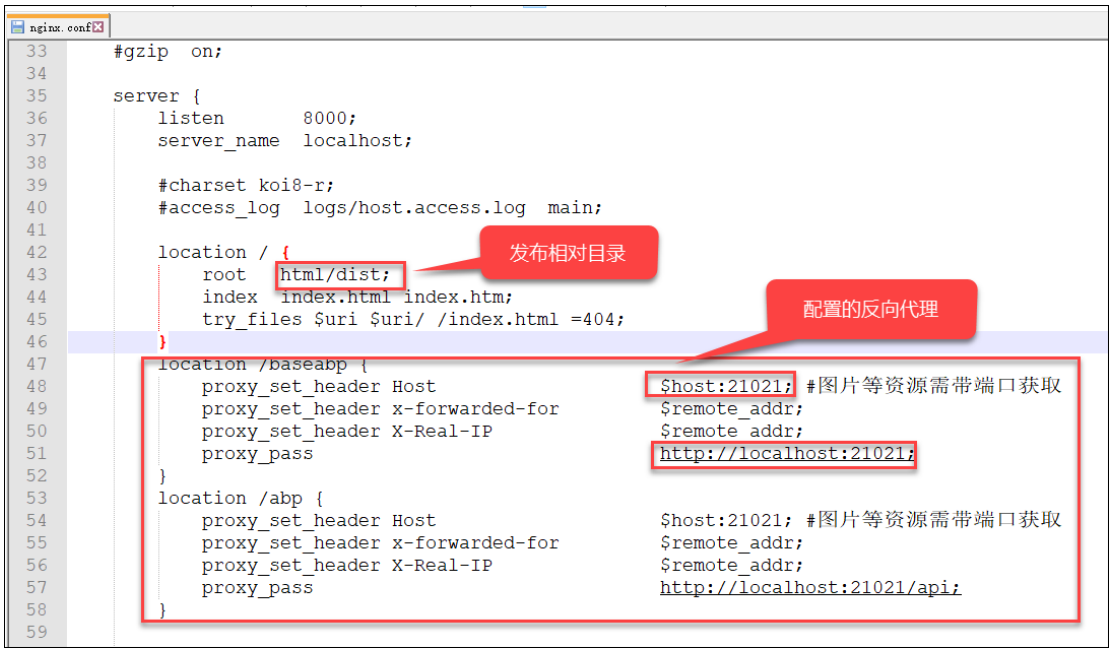



而发布应用到服务器的时候，我们需要配置它的反向代理设置。

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。

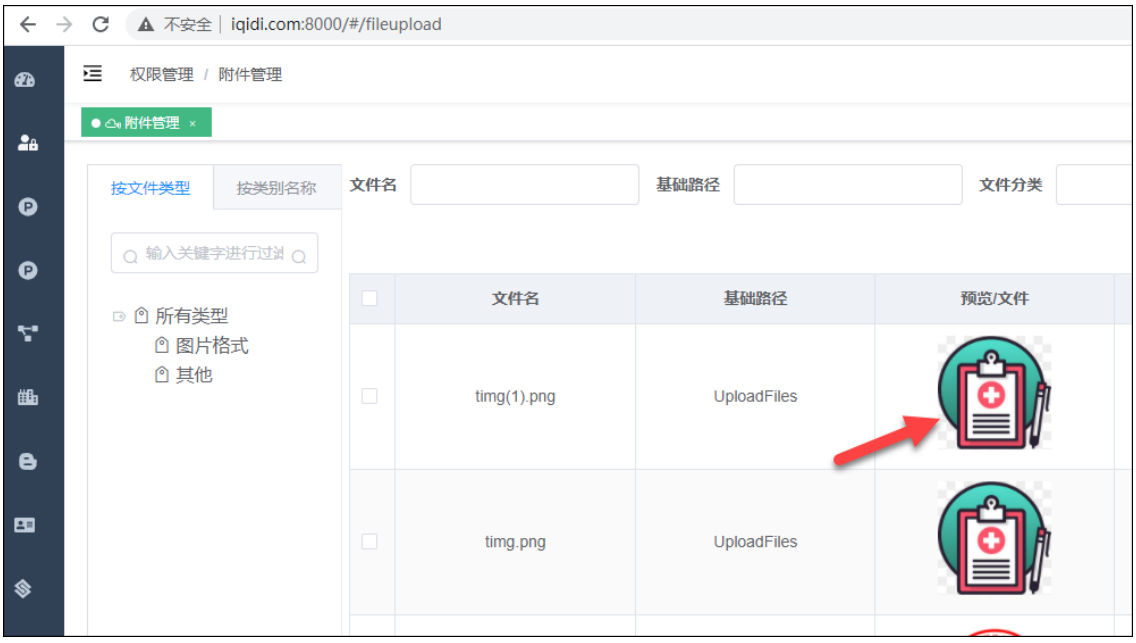



完整设置信息如下所示：

```
    server {
        listen       8000;
        server_name  localhost;

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

        location / {
            root   html/dist;
            index  index.html index.htm;
            try_files $uri $uri/ /index.html =404;
        }
        location /baseabp {
            proxy_set_header Host                    $host:21021; #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:21021;
        }
        location /abp {
            proxy_set_header Host                       $host:21021;#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:21021/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. ABP 框架使用 Mysql 数据库

ABP 默认的数据库是 SQLServer，不过 ABP 框架底层是 EF 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 ABP 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。

### 5.3.1. 使用 Mysql 数据库 ABP 后端的代码修改

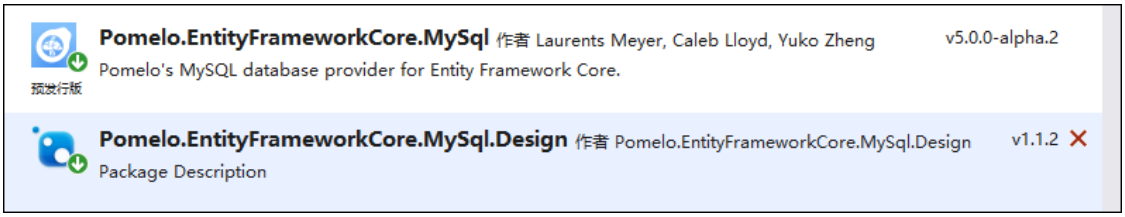
如果需要其他方式数据库的支持，那么需要引入相应的支持程序集，这里介绍如何修改增加一个 Mysql 数据库的支持。

在 Nuget 中安装 MySql 相关包，Oracle 公司提供的 MySql 包试了很多次，都还是用起来有问题，所以现在都是用的第三方的 MySql 包，这个包也是 ABP 官方文档中推荐使用的。

pomelo.entityframeworkcore.mysql

pomelo.entityframeworkcore.mysql.design

目前使用的是基于正式版的.NETCore5.0，那么 Microsoft.EntityFrameworkCore.Relational 对应版本是 5.0.0。因此，对应这些版本的驱动如下图所示。




在 EntityFrameworkCore 层中找到 XXDbContextConfigurer，修改 DbContext 中数据库配置，默认使用的 SqlServer，修改为 mysql。

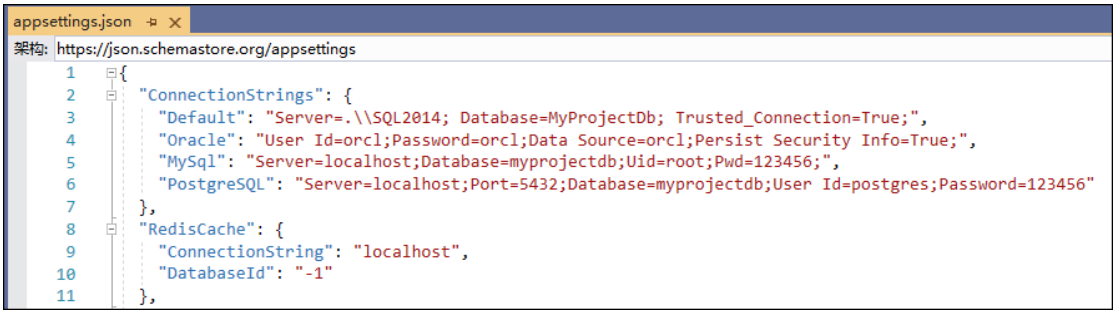
```
namespace MyProject.EntityFrameworkCore
{
    /// <summary>
    /// 配置使用特定数据库的处理
    /// </summary>
    public static class MyProjectDbContextConfigurer
    {
        public static void Configure(DbContextOptionsBuilder<MyProjectDbContext> builder, string connectionString)
        {
            //builder.UseSqlServer(connectionString);//默认SQlServer
            builder.UseMySql(connectionString, new MySqlServerVersion(new Version(5, 7)));//MySQL

            //builder.UseNpgsql(connectionString);//PostgreSQL
            //builder.UseOracle(connectionString);//Oracle
        }

        public static void Configure(DbContextOptionsBuilder<MyProjectDbContext> builder, DbConnection connection)
        {
            //builder.UseSqlServer(connection);//默认SQlServer
            builder.UseMySql(connection, new MySqlServerVersion(new Version(5, 7)));//MySQL

            //builder.UseNpgsql(connection);//PostgreSQL
            //builder.UseOracle(connection);//Oracle
        }
    }
}
```

其中 UseMySql 需要用到第二个版本参数，我 Mysql 用的是 Mysql5.7，因此使用代码构

建版本参数。

```
new MySqlServerVersion(new Version(5, 7)
```

我们在 Host 启动项目里面的 appsettings.json 里面定义了不同数据库的连接信息，如下所示。




而我们系统不同的数据库连接就就是来自 ConnectionStrings 里面的键值对象，我们 ABP 框架里面，使用引用常数来配置对应的数据库连接。

```
namespace MyProject.EntityFrameworkCore
{
    /* This class is needed to run "dotnet ef ..." commands from command line on development. Not used anywhere else */
    public class MyProjectDbContextFactory : IDesignTimeDbContextFactory<MyProjectDbContext>
    {
        public MyProjectDbContext CreateDbContext(string[] args)
        {
            var builder = new DbContextOptionsBuilder<MyProjectDbContext>();

            //获取配置文件信息
            var configuration = AppConfigurations.Get(WebContentDirectoryFinder.CalculateContentRootFolder());

            //通过连接字符串切换对应的数据库, ConnectionStringName默认为Default
            MyProjectDbContextConfigurer.Configure(builder, configuration.GetConnectionString(Constants.ConnectionStringName));

            return new MyProjectDbContext(builder.Options);
        }
    }
}
```

那么我们修改其中对应的常数定义，切换为我们所需要的 Mysql 数据库即可，如下所示。

```
namespace MyProject
{
    /// <summary>
    /// 系统定义常量
    /// </summary>
    public class Constants
    {
        public const string LocalizationSourceName = "EventCloud";

        /// <summary>
        /// appsettings.json定义的连接字符串名称
        /// </summary>
        public const string ConnectionStringName = "MySql";//Default/MySql

        /// <summary>
        /// 是否为多租户模式
        /// </summary>
        public const bool MultiTenancyEnabled = true;
    }
```

至此，代码上修改就完成了，那么我们现有运行的 SQLServer 数据库，如何迁移到 Mysql

环境中去呢？

## 5.3.2. 基于 SQLServer 创建 Mysql 数据库的架构和数据

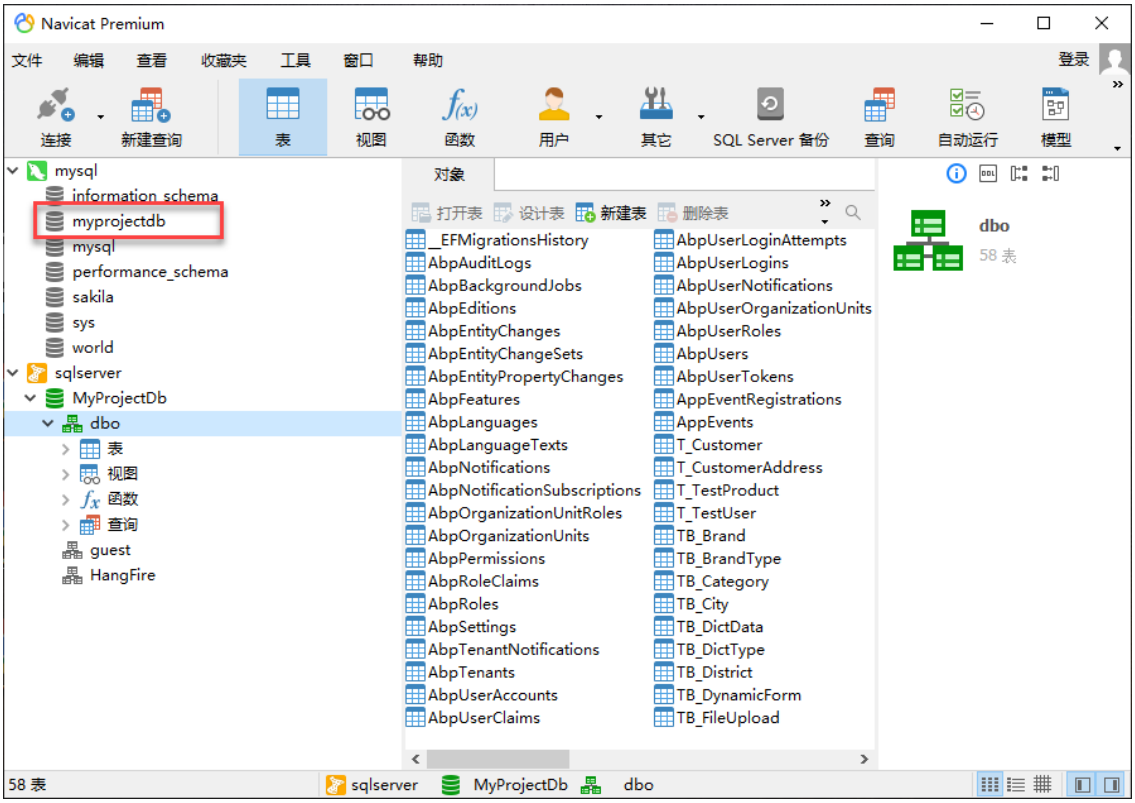
上面小节介绍了切换到其他数据库 Mysql 数据库的时候，代码上的调整修改，但是我们还需要把已有 SQLServer 上运行的数据库迁移到 Mysql 上去，如何处理呢。

有些人希望通过 ABP 带有的 Migrator 项目进行项目数据库的重构，不过我不建议使用这个，因为我们实际开发的时候，变动了很多数据库，而变动了再映射回到代码里面，比较麻烦，不如在数据库基础上进行迁移来的快捷完整。

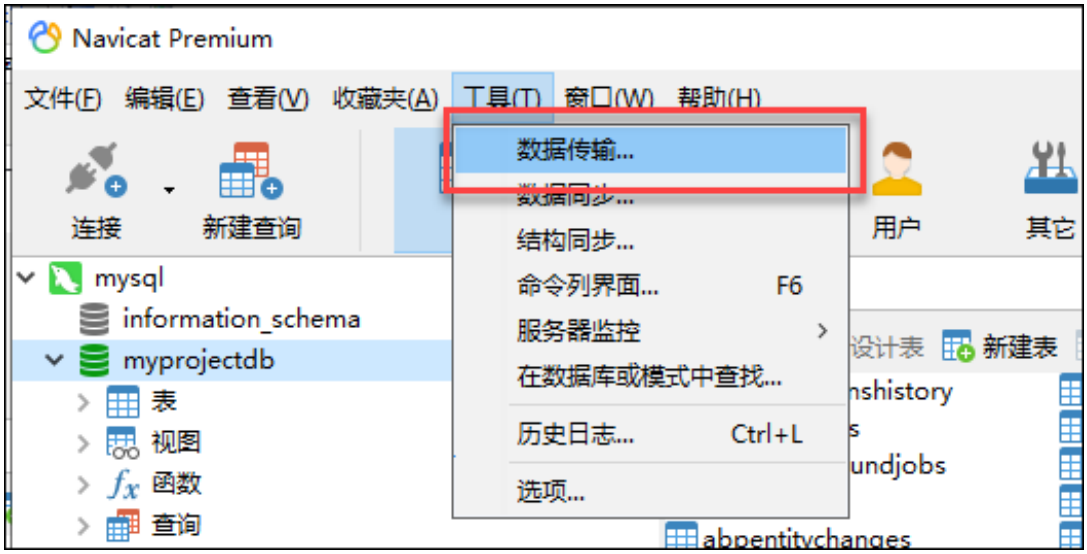
我这里通过工具的方式，把 SQLServer 数据库导入到 Mysql 数据库里面，然后在导出脚本进行一定的调整就可以完成。

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。

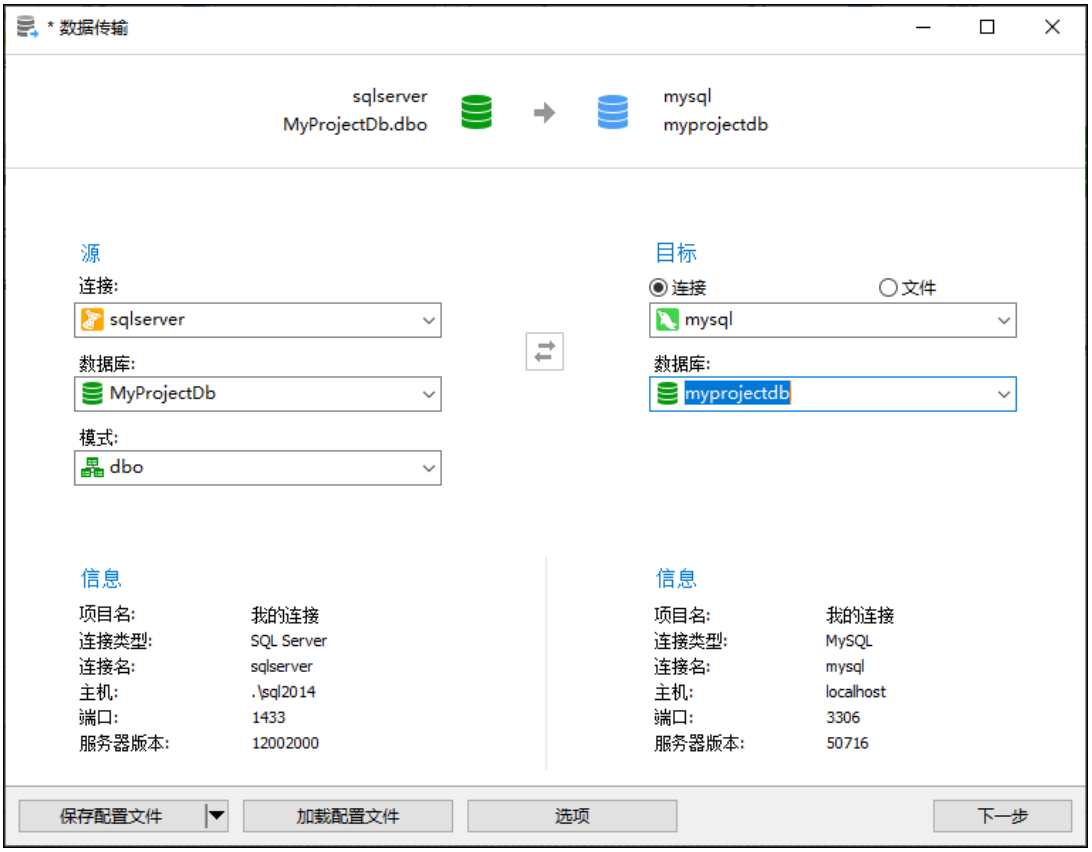
在 Navicat Premium 工具上连接好 Mysql 数据库和 SQLServer 数据库，然后在 Mysql 里面创建一个同名的数据库，以 Utf-8 编码创建数据库即可，如下所示。




然后在工具上选择数据库传输



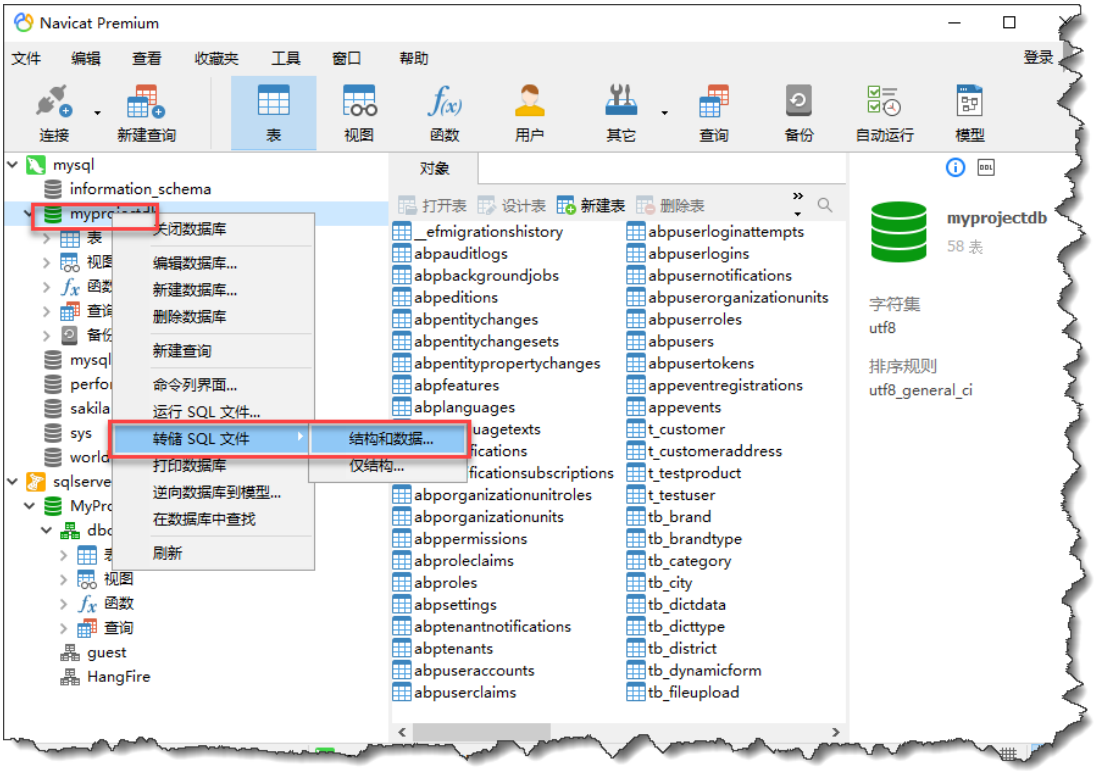

在弹出的界面中选择源数据库和目标数据库，如下所示。



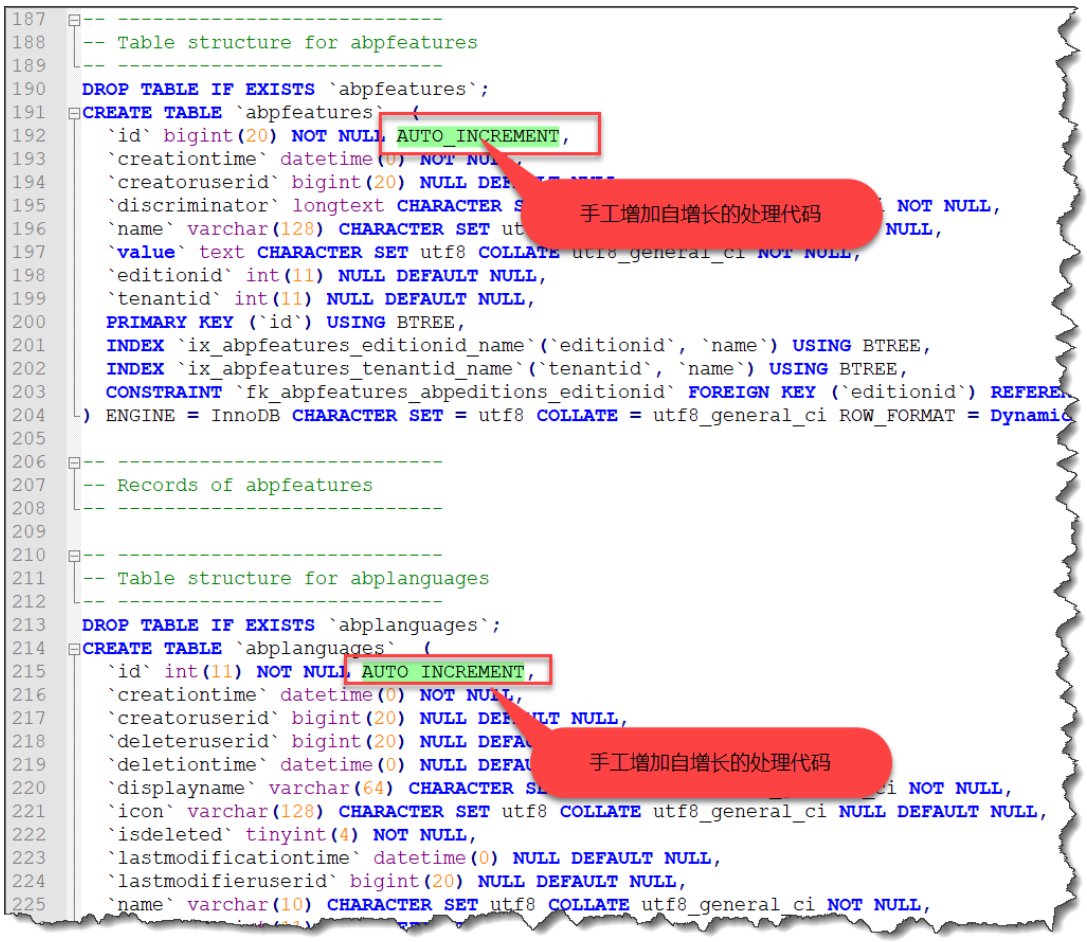

然后一步步处理即可完成数据库结构和数据的导入，这样 Mysql 数据库里面就有对应的数据库结构和数据了。

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。

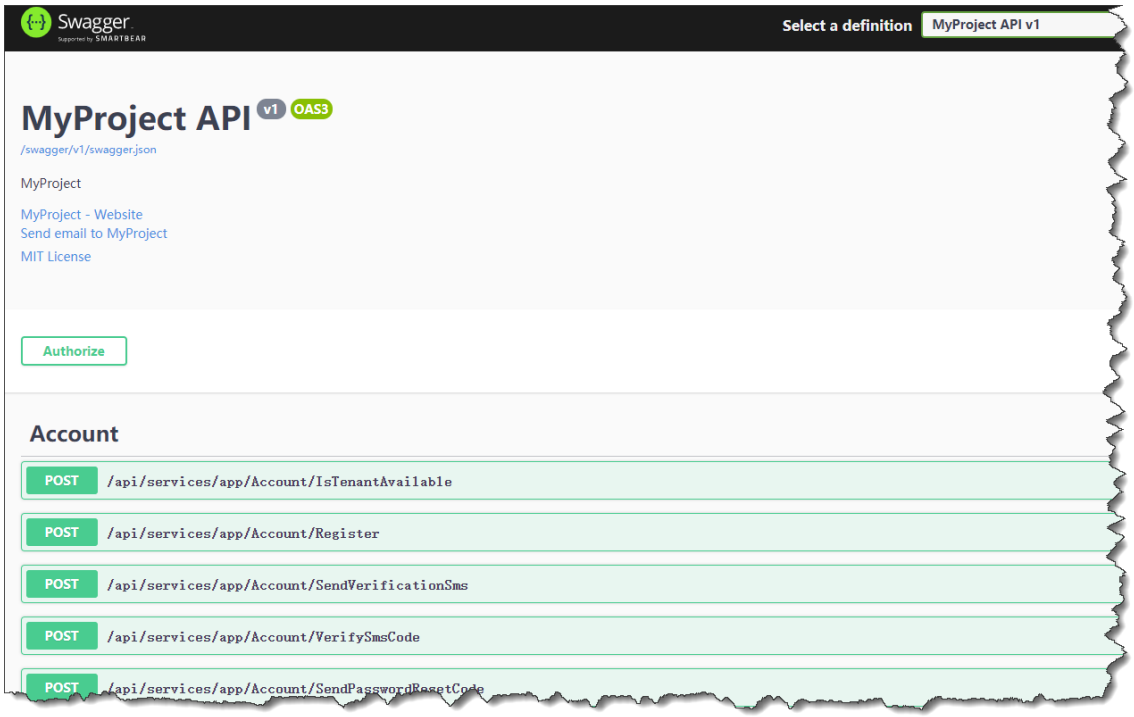



用 Notepad++打开导出的数据库脚本，对自增长的表字段进行添加一个关键字 AUTO_INCREMENT，标注为自增长即可，如下所示。




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 ABP 后台项目，启动 Swagger 界面，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。