项目结构说明

您下载的源码文件名为DTcms_80_src.7z的压缩包，通过解压后，我们将得到以下四个目录

> **DTcms.Admin**
>
> > 后台管理项目，这是一个标准的Vue3.0 SPA单页面项目
>
> **DTcms.App（授权版本才有）**
>
> > 手机应用项目，这是一个标准的UniApp项目，它的的内核其实也是Vue3.0项目，它可以生成适用于H5网站、小程序、手机App应用。
>
> **DTcms.Core**
>
> > 后台端API项目，它是整个系统的核心，它负责存储和提供数据给前端应用，后端语言采用.Net 8.0 + EFCore，是一个跨平台应用。
>
> **DTcms.Web**
>
> > 电脑网站项目，采用Nuxt3.0框架，其实它的内核也是Vue3.0+Vite，兼容大部分的Vue3.0+Vite，使用服务端渲染(SSR)解决SEO搜索引擎抓取问题。

DTcms.Admin项目结构

```
node_modules     npm 加载的项目依赖模块
src     这里是我们要开发的目录，基本上要做的事情都在这个目录里。
 |- assets    静态文件目录，放置一些图片、css、字体等文件。
 |- components    vue组件文件的存放目录,也是主要的工作目录。
 |- pages    页面的主要目录，所有的Vue页面都要放在这里面。
 |- request    HTTP请求的封装方法目录。
 |- router    Vue路由的配置文件目录。
 |- stores    Vue State持久性存储封装的方法。
 |- utils    一些封装好的帮助方法。
 |- App.vue     项目入口文件。
 |-main.js     项目的核心文件。
public    这里面的文件发布后的文件原样带出。
index.html    首页入口文件，你可以添加一些 meta 信息或统计代码啥的。
package.json    项目配置文件。
vite.config.js    Vite的配置文件
```

DTcms.Core项目结构

```
DTcms.Core.API    RESTful API接口项目，引用Model、IServices 、Services 、DBFactory 、Common类库。
DTcms.Core.Common    公共组件方法类库，独立项目，不能引用任何类库。
DTcms.Core.DBFactory    数据库工厂类库，负责数据库上下文，引用Model、Common类库。
DTcms.Core.IServices    数据访问层接口类库，仓储模式对数据库的增删改查定义接口，以便依赖注入，引用Model、Common类库。
DTcms.Core.Model    数据库表实体类库，Models目录主要负责定义实体与数据库表的映射、关系，ViewModels目录主要负责业务需求的DTO实体定义。
DTcms.Core.Services    数据访问层接口实现类库，引用Model、IServices 、DBFactory 、Common类库。
```

DTcms.Web项目结构

```
assets    静态文件目录，放置一些图片、css、字体等文件。
components    用于存放非页面级的组件，即可以在多个页面中复用的组件。
composables    Vue组合式函数，自动导入到你的应用程序中。
layouts   应用的布局组件，可以定义页面的结构，相当于母版页
   |- account.vue    会员中心布局
   |- default.vue    默认布局
   |- home.vue 网站首页布局
middleware    中间件，在导航到特定路由之前运行代码
   |- route.global.js    全局路由中间件
node_modules    npm 加载的项目依赖模块
pages    存放页面组件，Nuxt3.0 采用文件系统路由
plugins   用于存放插件，如 vue-i18n 等，这些插件会在应用启动时自动加载。
public    这里面的文件发布后的文件原样带出。
server    应用程序中注册API和服务器处理程序，目前用不到。
utils    帮助的工具类方法
   |- common.js    自定义的一些JS封装方法
app.vue   项目入口文件
error.vue    项目自定义的错误页面
nuxt.config.ts    Nuxt配置文件
package.json    通过npm安装的组件都会自动登记在这里
```

项目开发

1. 准备工作

开发人员要求

1. 前端开发者要求

1. 熟悉并掌握`Vue 3.0`的语法，如果不熟悉，请点击 这里 学习相关教程；不需要TypeScript，我们用的是JavaScript；
2. 熟悉JavaScript语法、HTML、CSS样式，至少要掌握JavaScript、scss前端这些是最基本的；
3. 熟悉Vue Rotur、Pina，这些教程在跸哩跸哩网站上面都是可以找到的；

2. 后端开发者要求

1. 熟悉并掌握 .Net Core RESTful接口开发、EFCore框架，当然也意味着你必须要熟悉C#语法；
2. 至少要掌握熟悉一款数据库，如SQL Server 2012以上、Mysql 8.0.2以上，其中一些主流的数据库（包括国产数据库）都是可以支持的；

软件要求

1. 后端开发工具

> `Visual Studio 2022 社区版`（免费的）下载地址 或者`Visual Studio Code`，两者可以二选一，选择你自己喜欢的；
>
> 数据库方面可以选择主流的，如SQL Server、Mysql、Oracle、达梦数据库（国产）等等，只要有相关的NuGet包支持就可以。
>
> 我们标准的开发数据库是使用SQL Server 2012版本，你可以安装最新，性能会好点；

2.前端开发工具

> 前端的后台开发我们统一使用了`HBuilderX`国产开发工具，它的优点就是小巧和启动速度快，[下载地址]
>
> 当然除了`HBuilderX`，你也可以选择`Visual Studio Code`，具体要根据你的习惯来决定；

2. API项目开发

1.数据库迁移

由于我们API项目使用的是Code First模式，所下载的源码中并未包含数据库，所以在运行项目时，我们必须要先生成数据库，也称之为`数据迁移`，下面是Code First模式的一些解析：

> TIP
>
> Code First模式我们称之为“代码优先”模式，从某种角度来看，其实“Code First”和“Model First”区别并不是太明显，只是它不借助于实体数据模型设计器，而是直接通过编码方式设计实体模型（这也是为什么最开始“Code First”被叫做“Code Only”的原因）。但是对于EF它的处理过程有所差别，例如我们使用Code First就不再需要EDM文件，所有的映射通过“数据注释”和“fluent API”进行映射和配置。

1. 在Window下数据库迁移

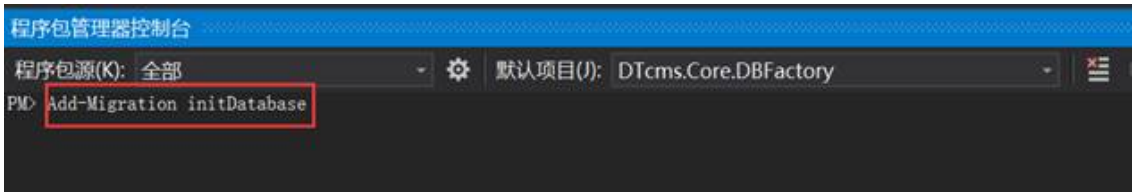
1. 确保你的Windows系统上已经安装了最新版本的Visual Studio 2022，并且包含了.NET 8.0的开发工具。在源码包里面打开DTcms.Core/DTcms.Core.API目录，找到“`DTcms.Core.sln`”双击打开；
2. 首先我们要在Visual Studio 2022 解决方案资源管理器里展开DTcms.Core.API，找到appsettings.json文件，双击打开，修改你的数据库连接字符串。

```
"ConnectionStrings": {
  "DBType": "SqlServer", //MySql,SqlServer,Sqlite,Oracle,PostgreSQL
  "WriteConnection":
"server=.;uid=sa;pwd=123;database=DTcms8db;TrustServerCertificate=true;",
  "ReadConnectionList": [
    "server=.;uid=sa;pwd=123;database=DTcms8db;TrustServerCertificate=true;"
  ],
  "Strategy": "Random"
}
```

注意：ReadConnectionList至少要保留一条记录，将uid和pwd改更成你的数据库账号密码即可。

3. 首先我们要将`DTcms.Core.API`设置为启动项目，可以在解决方案资源管理器中选中`DTcms.Core.API`项目，右键在弹出的菜单中选择`设为启动项目`，这步基本忽略，因为`DTcms.Core.API`是唯一的Web项目，默认是设为启动项目的。
4. 在工具栏上面的`视图`>`其它窗口`>`程序包管理器控制台`，打开`程序包管理控制台`，如图所示：

5. 在程序包管理控制台里面将DTcms.Core.DBFactory设置为默认项目，如图所示：



6. 输入以下命令，按回车键执行：

```
Add-Migration initDatabase
```

> TIIP
>
> 这里值得注意的是`initDatabase`是我们自定义迁移文件的名称，且不能重复，也就是说，下次你再进行迁移的时候就不能于重复使用这个名称，需要另外起名。

7. 等待数据库迁移文件生成后，我们再输入以下命令按回车键执行，当我们看到执行到Done时，证明已经成功生成了数据库：

```
Update-Database
```

8. 做完以上工作，我们的数据库就会自动创建好了，不需要我们手动创建数据库。

2. 下次进行数据库迁移

在我们对Model的类进行修改时，如果涉及到增删改字段，需要同步到数据库里面，这时候你就需要再一次迁移，依次的操作和上面是一样的，执行命令顺序如：

```
Add-Migration 自定义名称
Update-Database
```

3. 删除数据库和迁移文件

当我们手动调整了数据库结构或更改了类名或属性后，造成迁移文件和数据库不同步时，数据库再次迁移就会失败，此时我们打算重做数据库，这里，我们就需要先删除数据库，再执行删除迁移文件，具体命令如下：

```
Drop-Database //删除数据库
Remove-Migration //删除迁移文件，一次删除一条最新的数据迁移文件
```

当我们删除完迁移文件后，再依次像上面首次迁移数据库迁移的命令即可。

在苹果系统Mac OS数据迁移

在 macOS 上使用 Visual Studio 2022 进行 Entity Framework Core (EF Core) 的数据库迁移，首先我们需要将 `DTcms.Core.API` 项目设置为默认项目，具体步骤如下：

1. 设置开发环境

确保已经在 macOS 上安装了以下工具：

- Visual Studio 2022 for Mac
- .NET SDK

2. 安装 EF Core CLI 工具

在终端中运行以下命令，确保安装了 EF Core 命令行工具：

```
dotnet tool install --global dotnet-ef
```

验证安装是否成功：

```
dotnet ef --version
```

3. 添加和应用迁移

1. **添加迁移**

   在终端中，导航到 `DTcms.Core.API` 项目目录，并运行以下命令以创建初始迁移：

   ```
   dotnet ef migrations add InitialCreate
   ```

   这将生成一个新的迁移文件夹，包含数据库架构的初始快照。

2. **应用迁移**

   运行以下命令以将迁移应用到数据库：

   ```
   dotnet ef database update
   ```

4. 更新模型和迁移

如果需要更新模型（例如，添加新的属性或实体），请按以下步骤操作：

1. **更新实体类**

   更新实体类，例如，添加一个新的属性：

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
    public int Stock { get; set; }  // 新属性
}
```

2. **添加新迁移**

   创建新的迁移以反映模型的更改：

   ```
   dotnet ef migrations add AddStockToProduct
   ```

3. **应用迁移**

   将新迁移应用到数据库：

   ```
   dotnet ef database update
   ```

结论

通过上述步骤，你可以在 macOS 上使用 Visual Studio 2022 进行 EF Core 的数据库迁移。这包括设置项目、配置 EF Core、创建和应用迁移，以及在模型发生更改时更新迁移。确保每次更改模型后都生成新的迁移并将其应用到数据库，以保持数据库架构与模型一致。

2. 项目启动

API站点做为主要的驱动，它将为后台、前台Web、手机APP提供数据，所以在开发前端时，我们需要启动API项目以便给前端提供数据访问，如果你是团队合作，你可以在局域网启动一个API站点。当然，这个地址必须要在局域网中能访问。

一般情况下，我们不需要部署到API，.Net为我们提供了命令就可以直接启动一个站点。

通过命令启动一个站点

1. 首先我们先用Visual Studio 2022打开DTcms.Core项目，在DTcms.Core.API项目上面右健，在弹出的菜单中选择`生成`，这样就会自动编译程序集到DTcms.Core.API项目的`bin目录`里面。
2. 我们直接打开文件夹，例如你的项目在F盘的DTcms.Core里面，此时我们可以进入F:\DTcms.Core\DTcms.Core.API\bin\Debug\net8.0，在地址栏上面输入cmd，按回车。
3. 此时弹出了DOC窗体，输入以下命令回车即可启动API站点

   ```
   dotnet DTcms.Core.API.dll --urls="http://*:5200"
   ```

4. 因为需要调试，所以不要关闭DOC窗口，关闭即代表网站不在运行。

   > 注意：编译项目的时候，必须要关闭DOC窗口，如果你的项目还在运行，占用进程，Visual Studio 2022无法编译的。

> 除了以上的方式，你当然也可以直接从Visual Studio 2022直接启动调试。
>
> 或者部署到服务器或本地的IIS里面去运行。

3. 项目发布

在Visual Studio 2022里面选中 `DTcms.Core.API` 项目，右健弹出的菜单中选择 发布，这时会自动编译。

稍等片刻后，在下面的`输出`窗口中，按住`Ctrl健`+`鼠标左健`单击，直接打开发布文件目录，里面就是我们编译过后的发布文件，将里面的文件上传到你的站点目录下即可。

3. 后台项目开发

开发环境准备

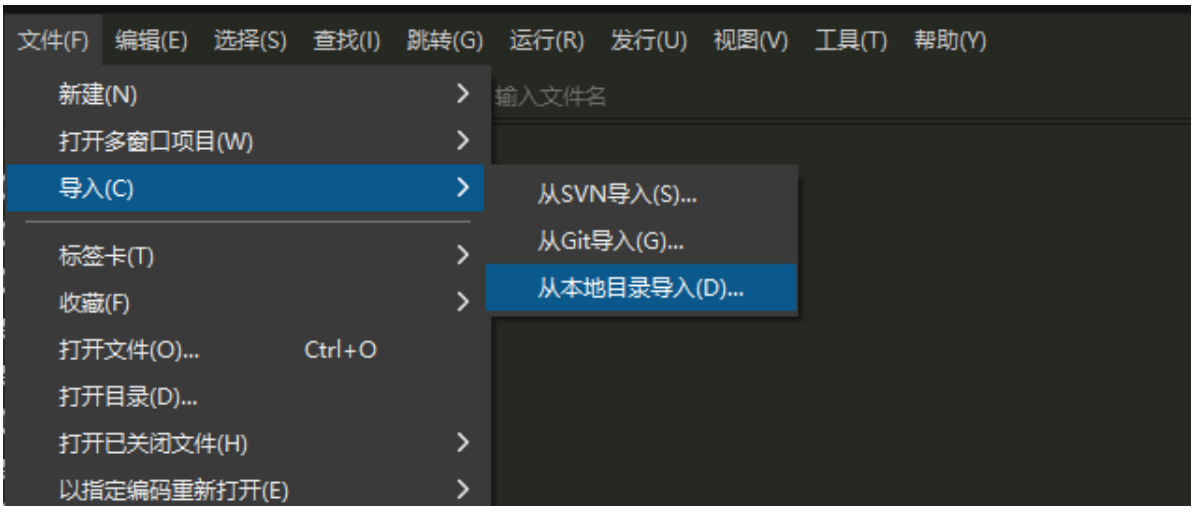
1. 后台管理项目的文件夹名为`DTcms.Admin`，它是一个标准的Vue 3.0 项目，关于它的目录结构，可以查看`项目结构说明`。
2. 开发工具我们统一使用 HBuilderX ，我们需要进入他们的官网下载安装，下载正式版进行本地安装。
3. 接下来我们还需要安装Nodejs，那是因为我们Vue是基于Nodejs环境的，点击进行他们的官网下载最新版的LTS稳定版。

Tip：如果你的Nodejs有多个版本需要管理，推荐你使用nvm进行对npm版本的管理以及切换，这样你的旧项目需要老版本的npm时，可以进行切换，nvm 安装说明

项目运行

1. 项目导入

打开HBuilderX开发工具，从左上角的 文件 > 导入 > 从本地目录导入，选择DTcms.Admin目录，导入即可，如图：



2. API地址修改

在HBuilderX开发工具里，依次展开DTcms.Admin > public ，我们将会看到一个config.js，把里面的地址修改成你正在运行或后端人员提供给你的API地址。

```
export default {
    //请求API地址
    baseApi: 'http://localhost:5200'
}
```

注意：API网址后台不要带/结尾，很多人都是因为带了/结尾，导致请求不到数据。

3. 安装依赖项

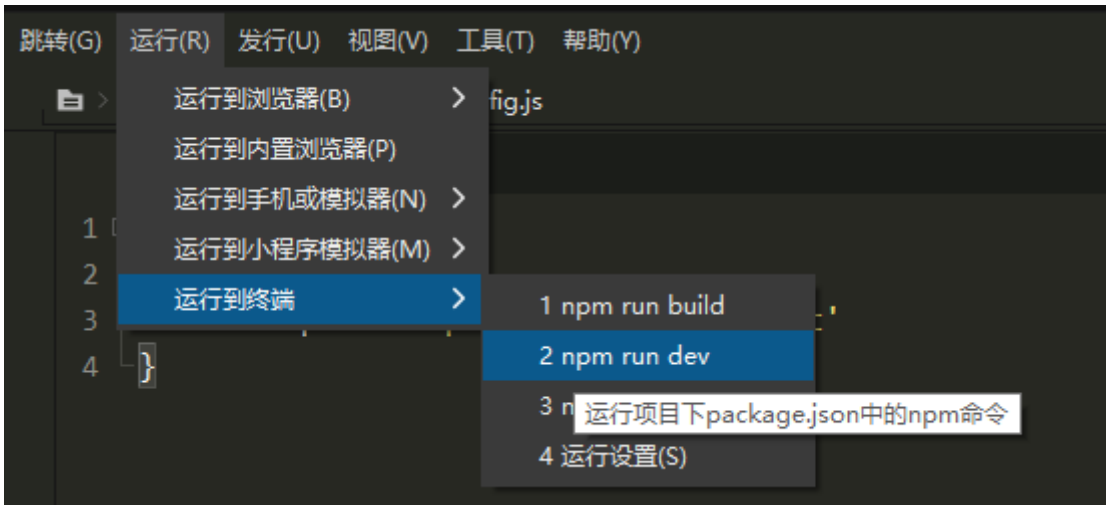
在项目里的 `node_modules` 文件夹存在的是我们在开发时用到的组件，由于太大，我们在提供源代码是把里面的文件删除了。所以我们需要通过npm安装，在确认安装Nodejs后，执行以下命令即可：

```
npm install
```

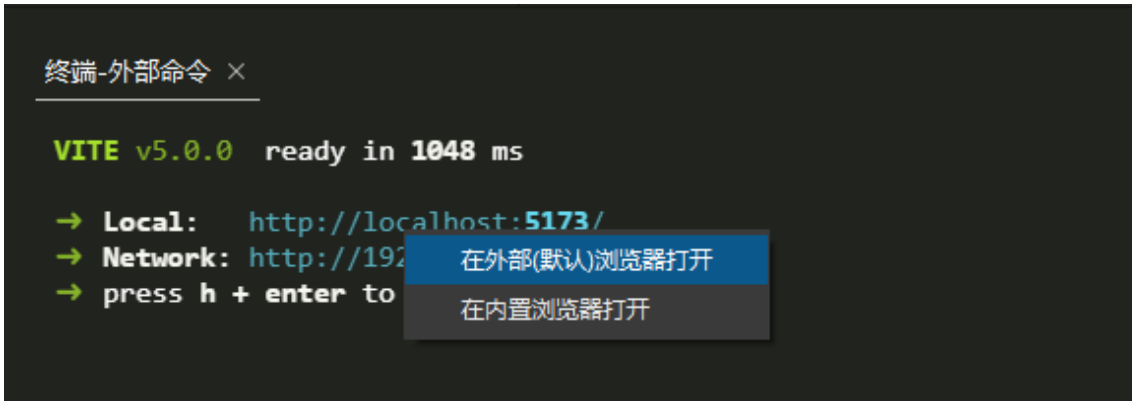
> 在安装的过程序中，可能会遇到下载慢，安装过程中有错误导致中断下载，可以把`node_modules`文件夹里面的文件全部删除后再执行以上命令。
>
> 如果你需要更快的速度，可以设置国内镜像，如 淘宝镜像

4. 项目运行预览

1. 首先双击打开一个项目的文件，让HBuilderX知道你要运行哪个项目。
2. 在主菜单栏中选择 `运行` > `运行到终端` > `npm run dev`，第一次运行的时候可以会弹出让你选择内置还是在外部运行，我们一般选择内置，外部的话他们打开一个CMD窗口，实际都一样。



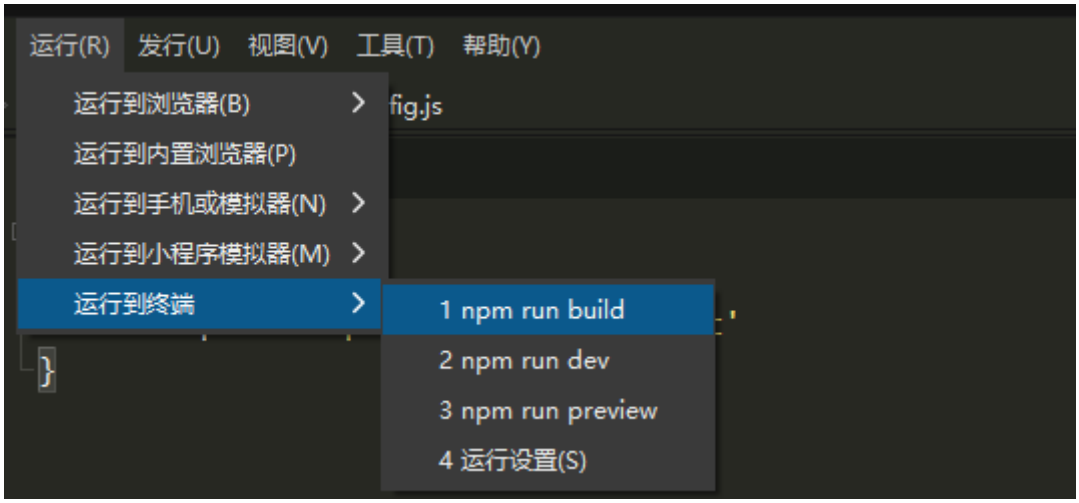
3. 执行后下面就会有两个预览地址，一个是本地预览地址，另一个是局域网都可以访问的地址，单击它，选择默认浏览器中打开。



这样，我们的项目就可以实时预览了。

5. 项目发布

1. 依然要双击打开一个项目的文件，让HBuilderX知道你要发布哪个项目。
2. 在主菜单栏中选择 `运行` > `运行到终端` > `npm run build`，如图所示：



4. 命令执行完毕后，在项目里面就多出了一个`dist`目录，这是一个纯HTML网站文件，将这些文件上传到你的空间下面即可，服务器上面不用安装任何软件即可运行。

4. Web项目开发

开发环境准备

1. 前端Web项目的文件夹名为`DTcms.Web`，它是一个标准的Nuxt 3.0 项目，关于它的目录结构，可以查看`项目结构说明`。
2. 开发工具我们统一使用 HBuilderX，我们需要进入他们的官网下载安装，下载正式版进行本地安装。

   > 如果你已经安装，跳过些步骤

3. 接下来我们还需要安装Nodejs，那是因为我们Vue是基于Nodejs环境的，点击进入Nodejs的官网下载最新版的LTS稳定版。

   > 如果你已经安装，跳过些步骤

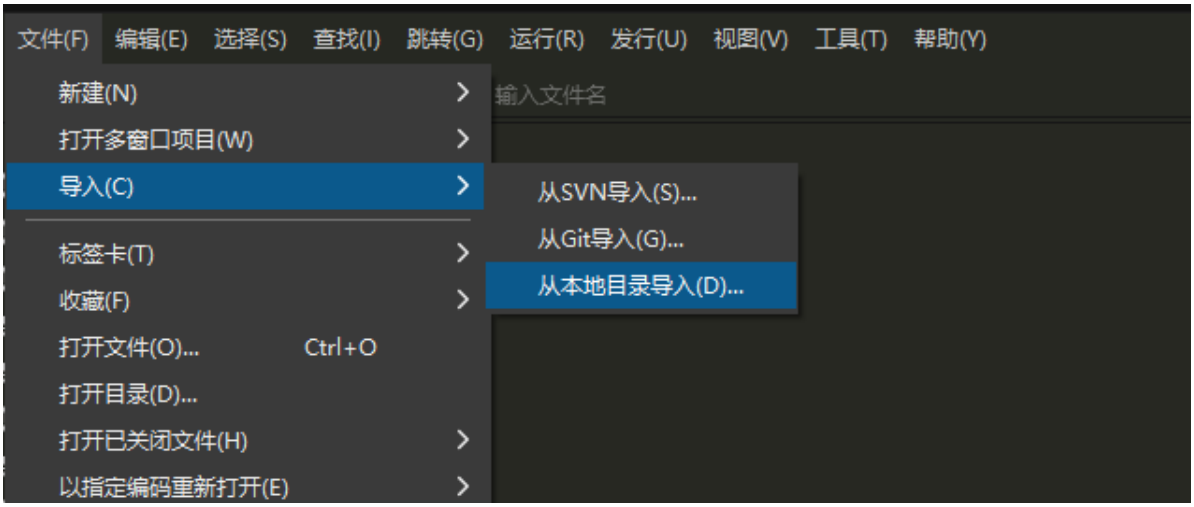
4. 在开发之前，你需要对Nuxt 3.0框架要有所熟悉，推荐套简单上手的视频，可以点击这里查看 视频教程，以便您能更快上手。

Tip：如果你的Nodejs有多个版本需要管理，推荐你使用nvm进行对npm版本的管理以及切换，这样你的旧项目需要老版本的npm时，可以进行切换，nvm安装说明

项目运行

1. 项目导入

打开HBuilderX开发工具，从左上角的 文件 > 导入 > 从本地目录导入，选择DTcms.Web目录，导入即可，如图：



2. API地址修改

在HBuilderX开发工具里，展开DTcms.Web项目，我们将会看到一个nuxt.config.ts，把里面的地址修改成你正在运行或后端人员提供给你的API地址。

```
export default defineNuxtConfig({
    runtimeConfig: {
        isServer: true,
        //客户端配置的API地址，在这里修改
        public: {
            baseURL: 'http://localhost:5200',
            timeout: 20000
        }
    },
    vite: {
        build: { chunkSizeWarningLimit: 1600 },
    },
    devtools: { enabled: true },
    modules: [
        '@element-plus/nuxt',
    ],
    css: [
        'element-plus/dist/index.css',
        'element-plus/theme-chalk/display.css',
        'animate.css/animate.min.css',
        '@wangeditor/editor/dist/css/style.css',
        '~/assets/fonts/iconfont.css',
        '~/assets/scss/style.scss'
    ],
})
```

注意：API网址后台不要带/结尾，很多人都是因为带了/结尾，导致请求不到数据。

3. 安装依赖项

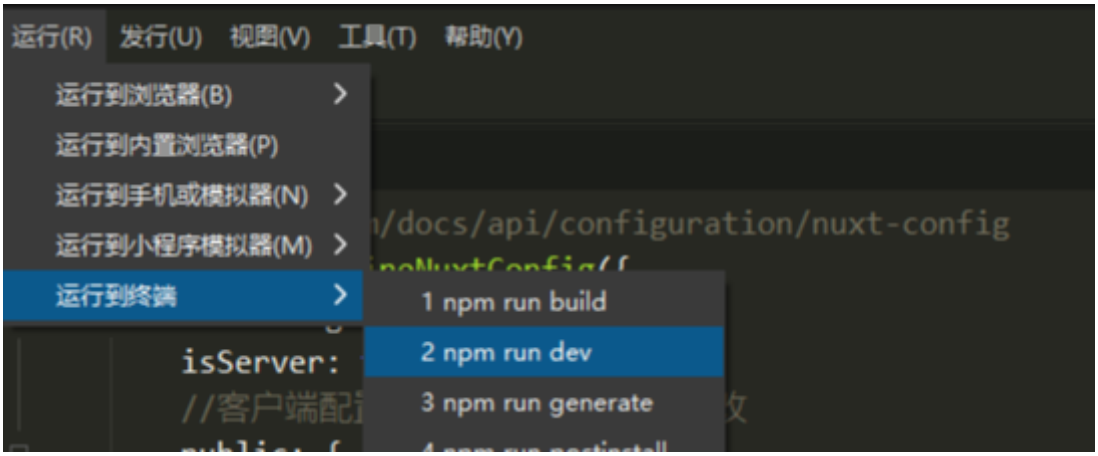
在项目里的 `node_modules` 文件夹存在的是我们在开发时用到的组件，由于太大，我们在提供源代码是把里面的文件删除了。所以我们需要通过npm安装，在确认安装Nodejs后，执行以下命令即可：

```
npm install
```

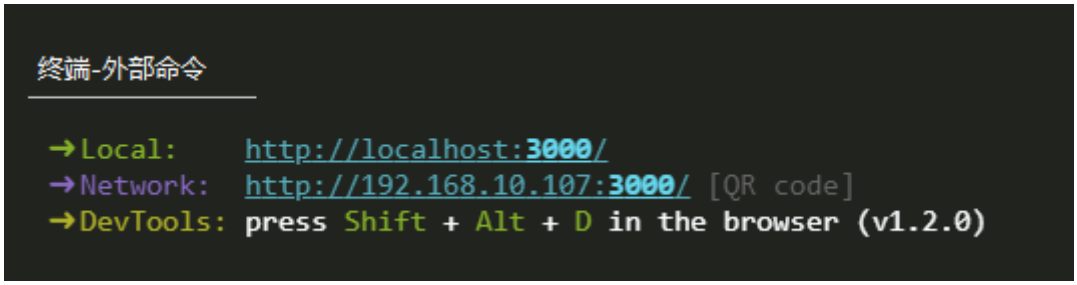
> 在安装的过程序中，可能会遇到下载慢，安装过程中有错误导致中断下载，可以把`node_modules`文件夹里面的文件全部删除后再执行以上命令。
>
> 如果你需要更快的速度，可以设置国内镜像，如 淘宝镜像

4. 项目运行预览

1. 首先双击打开一个项目的文件，让HBuilderX知道你要运行哪个项目。
2. 在主菜单栏中选择 `运行` > `运行到终端` > `npm run dev`，第一次运行的时候可以会弹出让你选择内置还是在外部运行，我们一般选择内置，外部的话他们打开一个CMD窗口，实际都一样。



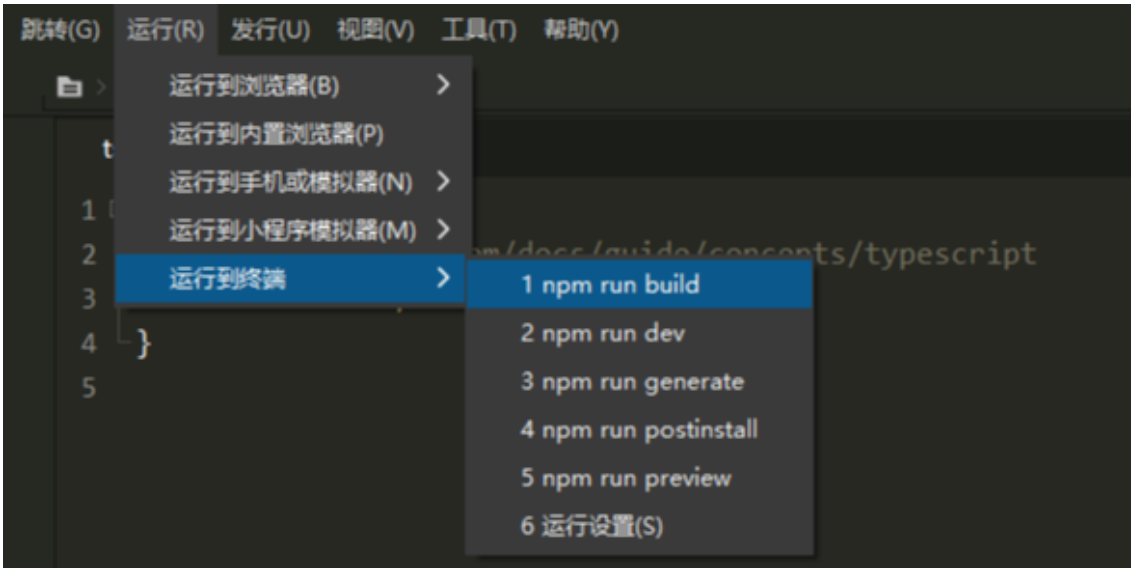
3. 执行后可能有点慢，因为要运行Nuxt的服务端和客户端，你的电脑配置要好点，要不然会很慢的哦！等待片刻后，就会出现两个预览地址，单击它，选择默认浏览器中打开。



这样，我们的项目就可以实时预览了。

5. 项目发布

1. 依然要双击打开一个项目的文件，让HBuilderX知道你要发布哪个项目。
2. 在主菜单栏中选择 `运行` > `运行到终端` > `npm run build`，如图所示：



3. 命令执行完毕后，在项目里面就多出了一个 `.output` 目录，要注意的是，要将 `.output` 目录上传到网站空间下，而不是 `.output` 目录里面的文件 。

> 例如我的网站路径是：`F:\wwwroot`
>
> 那么上传后的目录是这样的：`F:\wwwroot\.output\...`

项目部署

API网站部署

默认网站是不启用Redis缓存，如果需要部署Redis，用户可以在服务器里面安装Redis服务端，通过修改DTcms.Core/DTcms.Core.API项目里的appsettings.json以支持Redis，如下面这样修改。

```
"CacheSettings": {
  "Enabled": true,
  "Provider": "Redis", //可选项:"Redis"或"InMemory"
  "RedisServer": "localhost:6379",
  "InstanceName": "DTcms"
}
```

> TIP
>
> Enabled　是否启用缓存，`true`代表启用，`false`代表不启用
>
> Provider　它有两个可选值，分别是`Redis`和`InMemory`

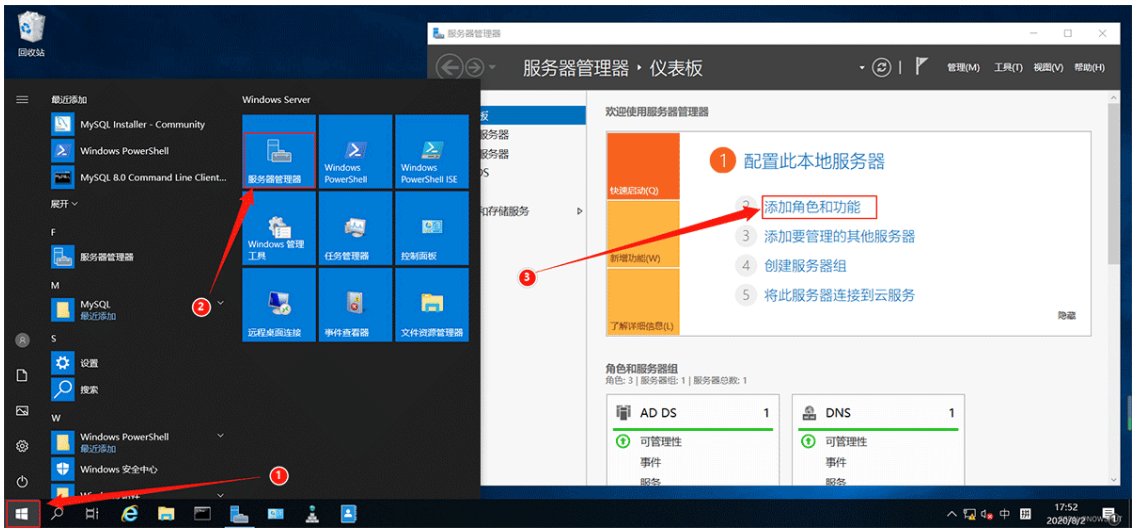
Windows系统部署

请确保已经在Visual Studio 2022里面发布了网站文件，并且上传到了服务器站点目录下。

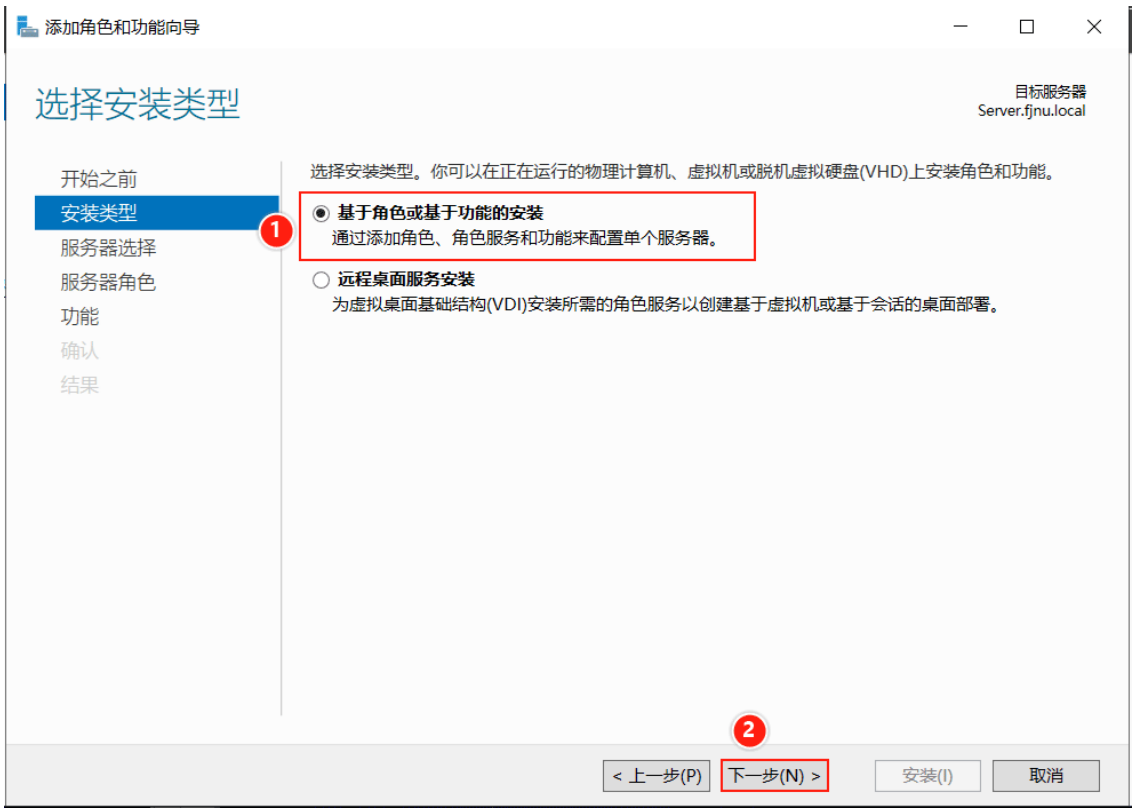
1. 安装 IIS 服务器

选择一台服务器作为WEB-IIS服务器，在Windows Server系统中，IIS角色是可选组件，默认情况下是没有安装的。

1. 打开【服务器管理器】，单击【添加角色和功能】。

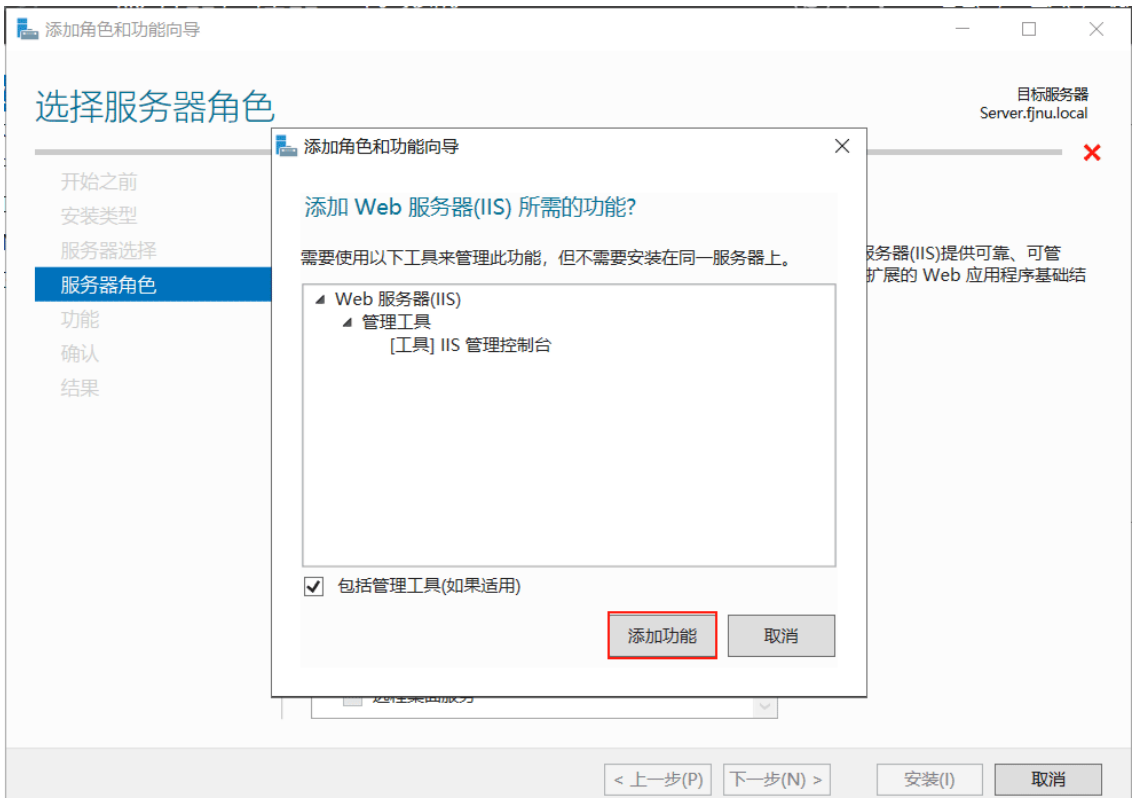


2. 默认选择【基于角色或基于功能的安装】，点击【下一步】。

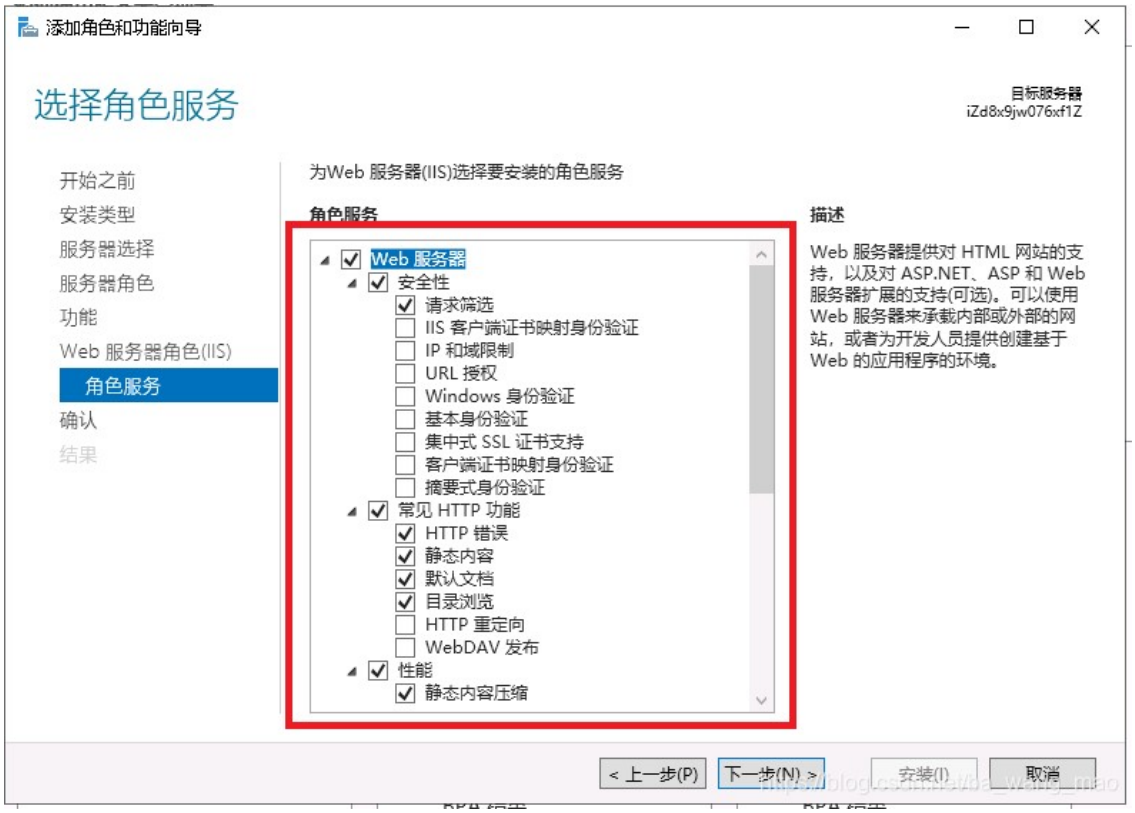


3. 默认选项，继续下一步。

4. 进入【服务器角色】页面，点击Web服务器（IIS），在弹出的对话框点击【添加功能】。

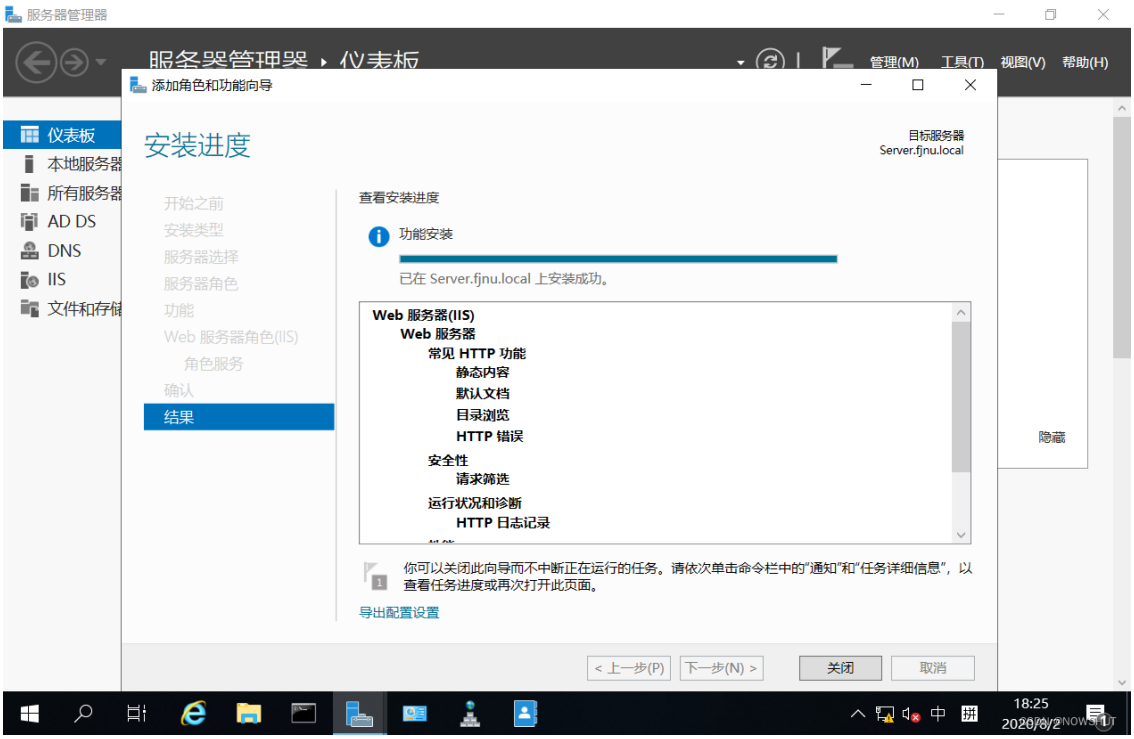


5. 默认选项，点击【下一步】，直到角色服务为止。角色服务中有很多选项没有选择，我们暂时不需要用到这些选项，继续点击【下一步】。



6. 进入【确认】页面，点击【安装】。

7. 进入【结果】界面， 安装过程需要等待一段时间， 安装完成后， 会在进度条下面显示【安装成功】。



2. 安装ASP.NET Core运行时

ASP.NET 核心运行时使你能够运行现有的 Web/服务器应用程序。在 Windows 上，我们建议安装托管捆绑包，其中包括 .NET 运行时和 IIS 支持。

1. 在Web服务器上面访问微软官网，下载ASP.NET Core运行时，必须是.Net 8.0的，点击这里直达下载，如下图所示：

运行应用 - 运行时

ASP.NET Core 运行时 8.0.6

ASP.NET 核心运行时使你能够运行现有的 Web/服务器应用程序。**在 Windows 上，我们建议安装托管捆绑包，其中包括 .NET 运行时和 IIS 支持。**

IIS 运行时支持 (ASP.NET Core Module v2)
18.0.24141.6

OS	安装程序	二进制文件
Linux	包管理器说明	Arm32 | Arm32 Alpine | Arm64 | Arm64 Alpine | x64 | x64 Alpine
macOS		Arm64 | x64
Windows	Hosting Bundle | x64 | x86 | winget 指令	Arm64 | x64 | x86

2. 服务器上安装下载的EXE文件，安装成功后，打开IIS的模块，就能看到ASP.NET Core Module v2的模块了。

3. 数据库连接字符串

1. 选择你喜欢的数据库进行安装，在这里就不再赘述了，可以查询对应的数据库教程。
2. 安装好IIS、ASP.NET Core运行时、数据库之后，我们就可以更改相关的数据库连接字符串了。
3. 找到我们API网站目录，找到appsetting.json，记事本打开，修改以下的数据库连接字符串，如图所示：

```
"ConnectionStrings": {
    "DBType": "SqlServer", //MySql,SqlServer,Sqlite,Oracle,PostgreSQL
    "WriteConnection":
"server=.;uid=sa;pwd=123;database=DTcms8db;TrustServerCertificate=true;",
    "ReadConnectionList": [

"server=.;uid=sa;pwd=123;database=DTcms8db;TrustServerCertificate=true;"
    ],
    "Strategy": "Random"
  },
```

参数名	说明
DBType	数据库类型
WriteConnection	写的数据库的连接字符串
ReadConnectionList	读的数据库连接字符串，这是一个数组，至少要保留一条连接字符串，可以和写的数据库连接字符串相同
Strategy	读的数据库策略，它有两个可选值：`Random` 随机策略，`Polling` 轮循策略

4. 修改好后，我们重启一下网站才能生效哦！

Linux系统部署

ASP.NET Core是跨平台项目，它支持部署到不同的平台，Linux系统也是比较常见的部署方案，DTcms 8.0的部署和普通的ASP.NET Core部署没有任何区别。

要将 .NET 8.0 API 项目部署到 Linux 环境，可以按照以下步骤操作。此流程假设你已经在 Windows 环境中开发并测试了你的 .NET 8.0 API 项目。

1. 准备工作

确保已经在 Linux 服务器上安装了 .NET SDK 和运行时。你可以参考 微软的官方文档 获取详细的安装步骤。

2. 在 Windows 上发布应用程序

1. 打开命令提示符或 PowerShell，导航到你的项目文件夹。
2. 使用以下命令发布你的应用程序：

   ```
   dotnet publish -c Release -r linux-x64 --self-contained
   ```

   这个命令会生成一个针对 Linux x64 的自包含（self-contained）发布包，意味着你的应用程序将包含所有的 .NET 运行时和依赖项。
3. 发布完成后，生成的文件将位于项目目录下的 `bin/Release/net8.0/linux-x64/publish` 文件夹中。

3. 将文件传输到 Linux 服务器

1. 使用 SCP、SFTP 或其他文件传输工具将发布包传输到 Linux 服务器。例如，使用 SCP 命令：

   ```
   scp -r bin/Release/net8.0/linux-x64/publish
   username@yourserver:/path/to/your/app
   ```

4. 配置和运行应用程序

1. 连接到你的 Linux 服务器。
2. 导航到你传输文件的目录：

   ```
   cd /path/to/your/app
   ```

3. 确保主程序文件（例如，`myapp`）具有可执行权限：

   ```
   chmod +x myapp
   ```

4. 运行你的应用程序：

   ```
   ./myapp
   ```

5. 使用 Systemd 管理服务

为了确保你的应用程序在系统重启后自动启动，可以使用 `systemd` 创建一个服务。

1. 创建一个新的服务文件，例如：

```
sudo nano /etc/systemd/system/myapp.service
```

2. 在文件中添加以下内容：

```
[Unit]
Description=My .NET 8.0 API Service
After=network.target

[Service]
WorkingDirectory=/path/to/your/app
ExecStart=/path/to/your/app/myapp
Restart=always
RestartSec=10
SyslogIdentifier=myapp
User=youruser
Environment=ASPNETCORE_ENVIRONMENT=Production

[Install]
WantedBy=multi-user.target
```

3. 保存并关闭文件。
4. 重新加载 systemd 配置：

```
sudo systemctl daemon-reload
```

5. 启动服务：

```
sudo systemctl start myapp
```

6. 如果你希望服务在系统启动时自动启动：

```
sudo systemctl enable myapp
```

6. 配置反向代理（可选）

为了更好地管理流量和安全性，你可以配置 Nginx 或 Apache 作为反向代理。以下是一个简单的 Nginx 配置示例：

1. 安装 Nginx：

```
sudo apt-get update
sudo apt-get install nginx
```

2. 配置 Nginx：

```
sudo nano /etc/nginx/sites-available/default
```

3. 添加以下内容到配置文件中：

```
server {
    listen 80;
    server_name your_domain_or_ip;

    location / {
        proxy_pass http://localhost:5000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection keep-alive;
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

4. 测试 Nginx 配置：

```
sudo nginx -t
```

5. 重新加载 Nginx：

```
sudo systemctl reload nginx
```

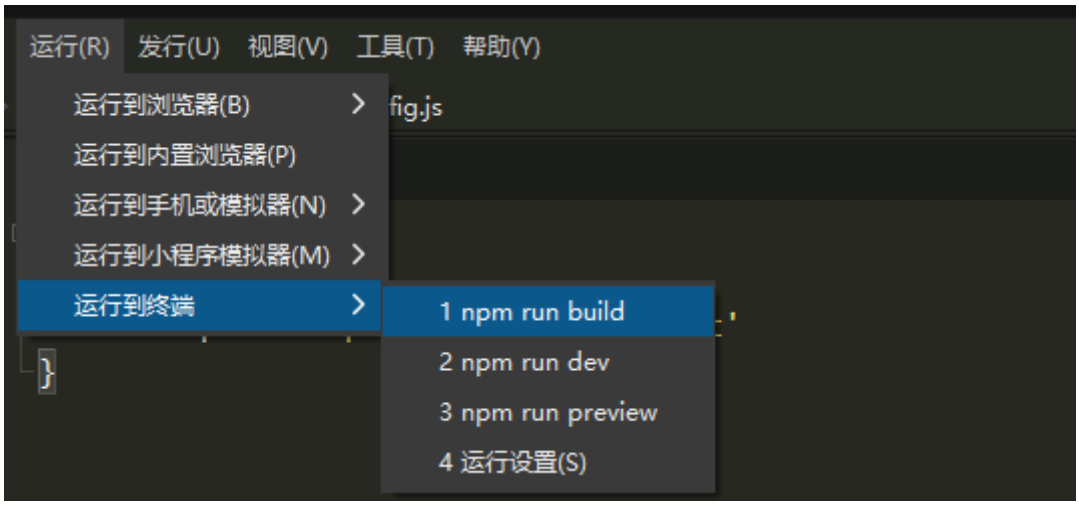
结论

通过上述步骤，你可以将 .NET 8.0 API 项目部署到 Linux 服务器，并配置服务以便自动启动和管理。如果有任何问题，可以参考 微软的官方文档 获取更多信息。

后台网站部署

项目发布

1. 在HBuilderX开发工具中打开DTcms.Admin项目，随便打开一个项目的文件，让HBuilderX知道你要发布哪个项目。
2. 在主菜单栏中选择 `运行` > `运行到终端` > `npm run build`，如图所示：



3. 命令执行完毕后，在项目里面就多出了一个`dist`目录，这是一个纯HTML网站文件，将这些文件上传到你的空间下面即可，服务器上面不用安装任何软件即可运行。

服务器部署

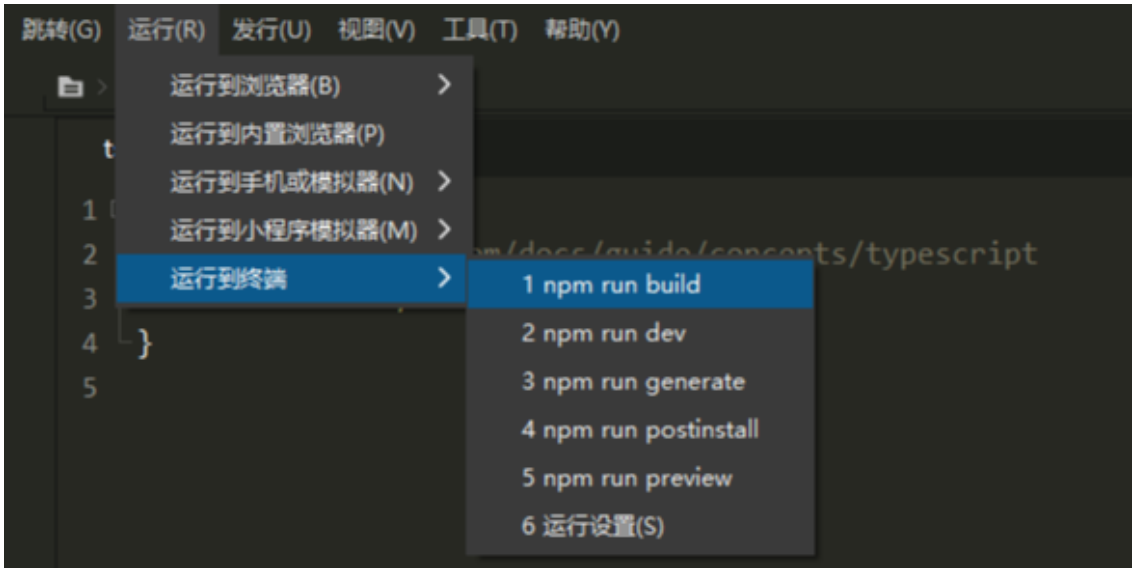
由于DTcms.Admin项目发布后，生成的是HTML网站，所以在服务器里面只要能支持HTML的站点都能顺利运行，值得一提的是发布后可以通过修改DTcms.Admin /public/config.js文件修改你的API的地址，不过需要清空一下你的浏览器缓存才能生效。

Web网站部署

项目发布

用`HBuilderX`打开`DTcms.Web`项目，双击打开一个项目的文件，让`HBuilderX`知道你要发布哪个项目。

1. 在主菜单栏中选择`运行` > `运行到终端` > `npm run build`，如图所示：



2. 命令执行完毕后，在项目里面就多出了一个`.output`目录，要注意的是，要将`.output`目录上传到网站空间下，而不是`.output`目录里面的文件 。

> 例如我的网站路径是：`F:\wwwroot`
>
> 那么上传后的目录是这样的：`F:\wwwroot\.output\...`

Windows服务器部署

DTcms.Web是Nuxt 3服务端渲染（SSR）项目，它包含前端和后端运行程序，所以相对于其它项目来说，是复杂点，但一切都是值得的。

1. 安装Nodejs

点击进入Nodejs的官网下载最新版的LTS稳定版，它包含了npm。MSI安装包下载，下载后直接运行安装即可。

2.安装pm2

> TIP
>
> 此步骤是在服务器上操作, 前提是安装好了Nodejs环境。

```
npm install pm2@latest -g
```

3.配置pm2

我们需要在网站的根目录下，创建一个`ecosystem.config.js`文件，内容如下：

```
module.exports = {
  apps: [
    {
      name: 'DTcmsWeb',
      port: '3000',
      exec_mode: 'cluster',
      instances: 'max',
      script: './.output/server/index.mjs'
    }
  ]
}
```

4. 启动Nuxt3服务端

到项目目录启动，执行以下命令：

```
pm2 start ecosystem.config.js
```

顺利的话，我们就成功启动了一个`http://localhost:3000`的站点，这是Nuxt3服务端，但是如果服务器重启的话，Nuxt3服务端就停止了，所以我们还需要执行以下命令：

1. 全局安装`pm2-windows-startup`

   ```
   npm install pm2-windows-startup -g
   ```

2. 保存当前进程

   ```
   pm2 save
   ```

3. 开机自启动(记得回车健)

   ```
   pm2-startup install
   ```

   > TIP
   >
   > 如果网站文件更新后，需要重新启动进程，在项目目录下执行CMD命令如下：

   ```
     pm2 restart all
   ```

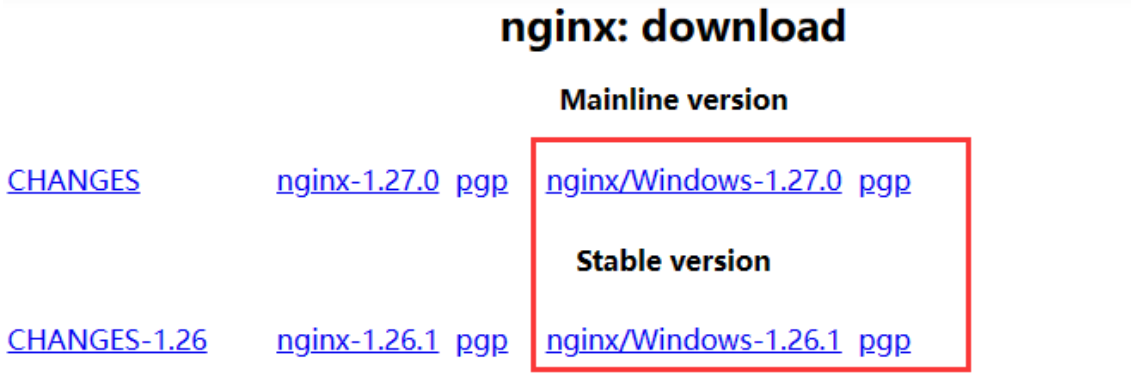
5. 反向代理(IIS或Nginx)

Nuxt3服务端虽然配置好了，但是用户想要访问网站，我们还需要绑定域名的是吧。这时候你就要考虑使用Nginx还是IIS了？

当然他们俩是可以共存的，但是不能同时监听80和443端口，如果你想使用Nginx，那么不要让IIS监听80和443端口就可以了，可以在IIS里面使用其它端口创建网站，让Nginx反向代理交回给IIS里面的网站。下面我们只介绍两种方法，二选一即可，我建议使用Nginx。

Nginx反向代理（推荐）

1. 访问Nginx官方网站，下载最新的Windows版本的Nginx。



2. 将下载的Nginx压缩包解压到你希望安装Nginx的位置。
3. 运行Nginx，打开命令提示符（CMD）或者PowerShell。导航到Nginx的安装目录下的nginx文件夹。运行以下命令启动Nginx

```
start nginx
```

4. 若要重新加载配置文件（例如，在修改了`nginx.conf`之后），运行：

```
nginx -s reload
```

5. 修改nginx.conf

   找到Nginx安装目录，记事本打开nginx.conf文件，修改以下内容：

```
server {
    listen 80;
    server_name 你的域名(不带http);

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

server {
    listen 443 ssl;
    server_name 你的域名(不带https);

    ssl_certificate /path/to/your/cert.pem;
    ssl_certificate_key /path/to/your/key.pem;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
```

```
            proxy_set_header Connection 'upgrade';
            proxy_set_header Host $host;
            proxy_cache_bypass $http_upgrade;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }
    }
```

经过以上的步骤，你们可以尝试一下输入你的域名是否可以访问了。

6. **为其他站点配置不同的端口或使用反向代理**

   确保 IIS 站点使用不同的端口，或者配置 Nginx 将其他域名的请求转发到 IIS。例如：

```nginx
server {
    listen 80;
    server_name othersite.com;

    location / {
        proxy_pass http://localhost:8080; # IIS 上的其他站点
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

URL Rewrite反向代理(不推荐)

> TIP
>
> 前提不使用Nginx，用URL Rewrite做为反向代理

1. 安装IIS

> TIP
>
> 安装IIS可以看回`API网站部署`，在这里不赘述了。

2. 安装URL Rewrite

可以点击这里下载安装，下载地址

3. 安装Application Request Routing

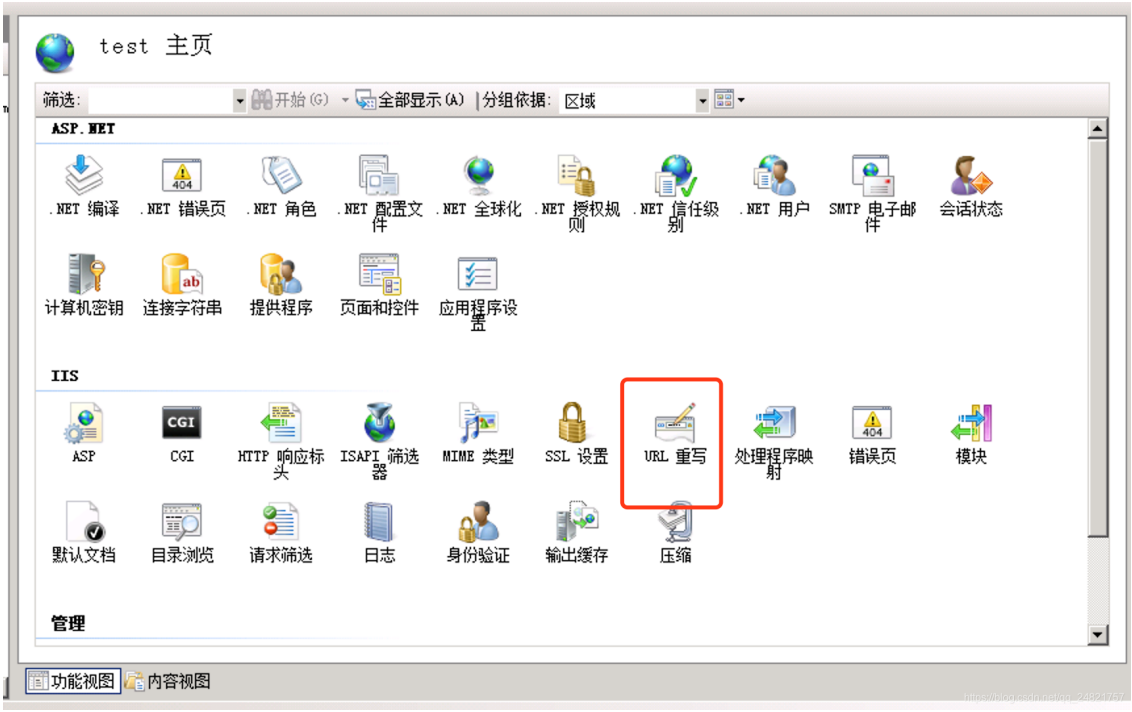
可以点击这里下载安装，下载地址

4.在IIS里面创建站点

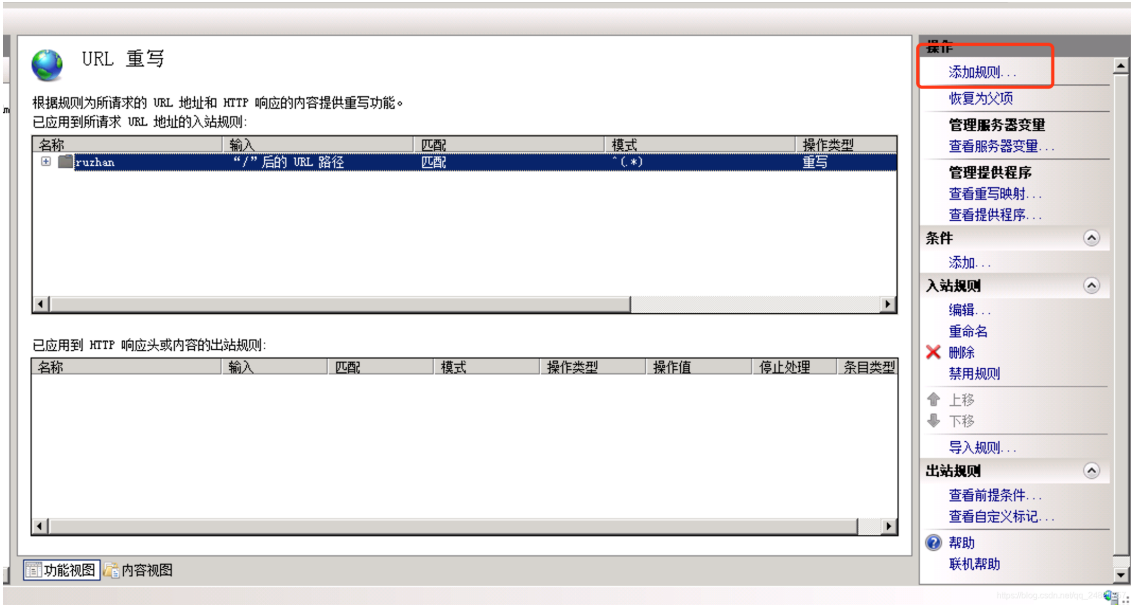
打开IIS后，我们创建一个空的站点，域名可以绑定你需要对外的域名，网站目录指定我们发布的DTcms.Web项目文件目录

5.配置URL重写

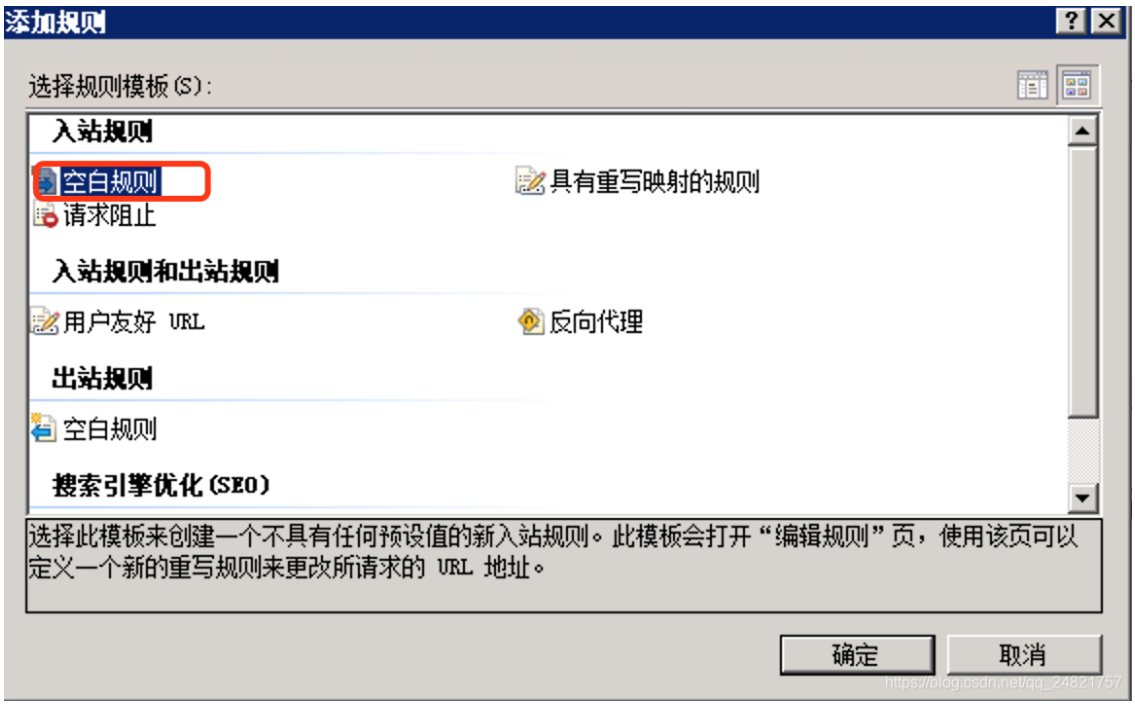
1. 选中刚才创建的站点，在IIS右边有个`URL 重写`，双击进入，如图：



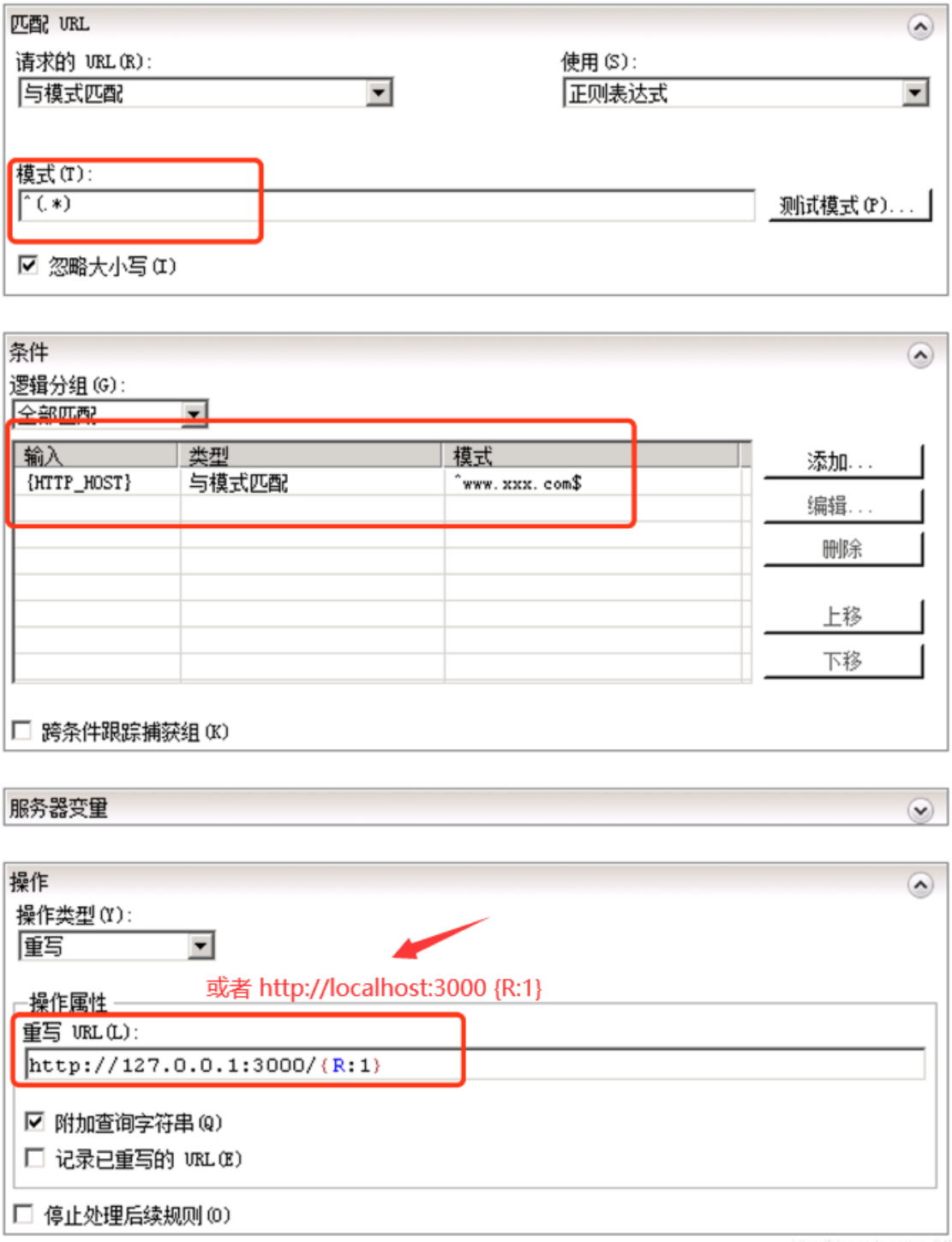
2. 在右边操作里选择`添加规则`，如图：



3. 选择反向代理或空白规则，如图：



4. 输入你对外的域名以及代正则



Linux服务器部署

1. 安装Nodejs

点击进入Nodejs的官网下载最新版的LTS稳定版，它包含了npm。[

```
# installs fnm (Fast Node Manager)
winget install Schniz.fnm

# download and install Node.js
fnm use --install-if-missing 20

# verifies the right Node.js version is in the environment
node -v # should print `v20.15.0`

# verifies the right NPM version is in the environment
npm -v # should print `10.7.0`
```

2. 安装pm2

> TIP
>
> 此步骤是在服务器上操作, 前提是安装好了Nodejs环境。

```
npm install pm2@latest -g
# or
yarn global add pm2
```

3. 配置pm2

我们需要在网站的根目录下，创建一个`ecosystem.config.js`文件，内容如下：

```
touch ecosystem.config.js
```

写入下面配置项

```
module.exports = {
  apps: [
    {
      name: 'DTcmsWeb',
      port: '3000',
      exec_mode: 'cluster',
      instances: 'max',
      script: './.output/server/index.mjs'
    }
  ]
}
```

- exec_mode:应用程序启动模式，这里设置的是cluster_mode（集群），默认是fork
- instances：启用多少个实例，可用于负载均衡。如果-i 0或者-i max，则根据当前机器核数确定实例数目。

4. 启动Nuxt3服务端

到项目目录启动，执行以下命令：

```
pm2 start ecosystem.config.js
```

顺利的话，我们就成功启动了一个`http://localhost:3000`的站点，这是Nuxt3服务端，但是如果服务器重启的话，Nuxt3服务端就停止了，所以我们还需要执行以下两个命令：

1. 保存当前进程

```
pm2 save
```

2. 开机自启动

```
pm2 startup
```

完成这两步之后，服务器重启也会自动运行Nuxt3服务端了

> TIP
>
> 此步骤是在服务器上操作, 前提是安装好了Nodejs环境。

```
pm2 restart all
```

5. 安装Nginx

有了上面这些步骤后，我们就可以安装Nginx做为反向代理，将你的域名请求交给http://localhost:3000来处理了。

我们通过nginx配置, 将以下配置代码添加到nginx.conf中

```
server {
    listen 80;
    server_name 你的用户端访问域名;

    location / {
        proxy_pass http://服务器IP:3000/;
    }
}
```