前后端分离设计

前后端分离指的是前端采用 HTML 页面，通过 Ajax 来调用 http 请求调用后端的 RESTful API，前端只需关注页面的样式与动态数据的解析&渲染，而后端专注于具体业务逻辑。前后端分离会为以后的大型分布式架构、弹性计算架构、微服务架构和多端化服务（例如：浏览器，安卓，IOS 等）打下坚实的基础。前后端分离框架图如图 14-1 所示：

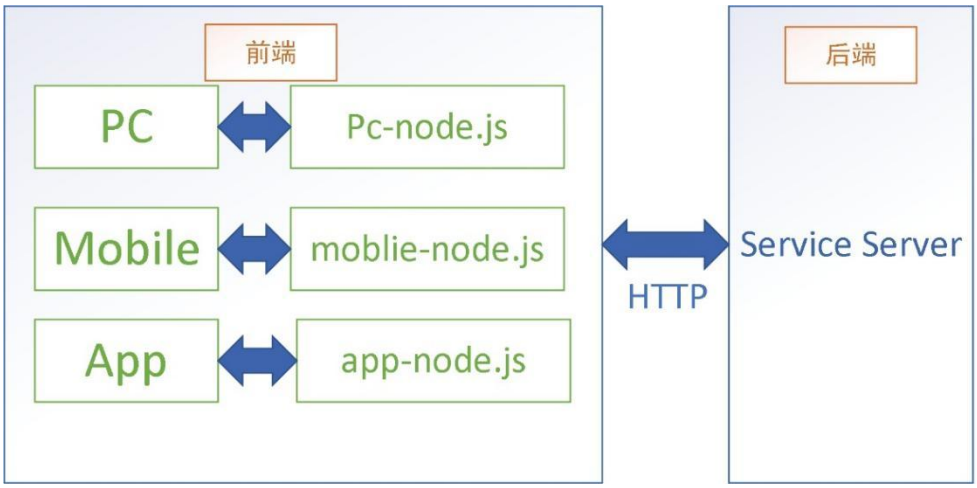

图 14-1　前后端分离框架图

14.1　Django 的前后端分离

Django 的普通项目是基于 MVT 模式（Model View Template）开发，而 Django 的前后端分离项目则是基于 MVVM 模式（Model View ViewModel）开发。MVVM 模式比普通 MVC 模式多一个 ViewModel 对象，它可以将数据和逻辑处理部分从 Controller 控制器中抽离出来，是 Model 和 Controller 之间的一座桥梁。MVVC 基本架构图如图 14-2 所示：

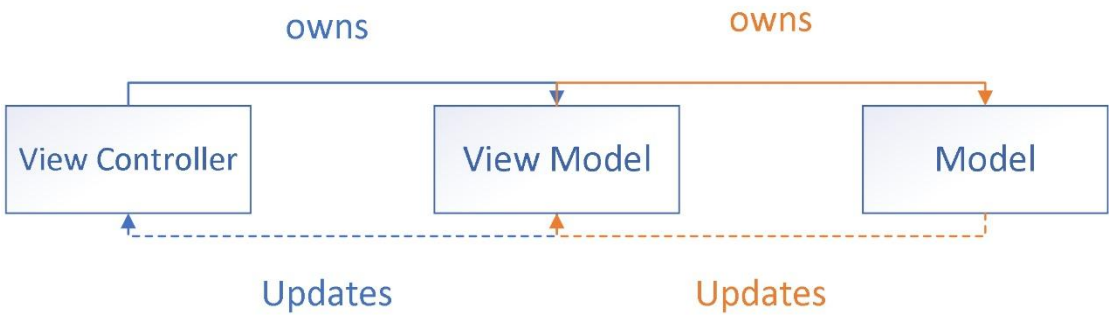

图 14-2　MVVC 基本框架图

Django 前后端分离项目原理：后端遵循 restful 规范开发 API，与前端进行数据交互，实现多端应用。

14.1.1　Django REST framework 简介

Django REST framework 框架是用于构建 Web API 的强大且灵活的工具包，通常简称为 DRF 框架或 REST framework。Django REST framework 的 10 个常用组件如下：权限组件；

认证组件；访问频率限制组件；序列化组件；路由组件；视图组件；分页组件；解析器组件；渲染器组件和版本组件。

Django REST framework 官方文档的地址是 https://www.django-rest-framework.org/。

14.1.2 API 介绍

API（Application Programming Interface）是一些预先定义的接口（如函数、HTTP 接口），或指软件系统不同组成部分衔接的约定。用来提供应用程序与开发人员基于某软件或硬件得以访问的一组例程，而又无需访问源码，或理解内部工作机制的细节。

例如，用户可以通过中国气象数据网提供的 API 接口，传入账号密码、时间和地点等参数来调用某时刻某站的中国地面气象站逐小时观测资料的温度要素。气象 API 服务以 Restful WebServices 的方式提供，对于要素型数据以 Json 格式直接返回，对于文件型数据以 Json 格式返回文件清单列表。

14.1.3 RESTful API

RESTful API 是使用 REST 原理的应用程序接口。REST API 定义了一组功能，开发人员可以通过 HTTP 协议（例如 GET 和 POST）执行请求并接收响应。

（1）HTTP 动词

对于资源的具体操作类型，由 HTTP 动词表示，常用的 HTTP 动词有下面五个。

```
GET ：从服务器去除资源
POST ：在服务器新建一个资源
PUT：在服务器更新资源（客户端提供改变后的完整资源）
PATCH：在服务器更新资源（客户端提供改变的属性）
DELETE：从服务器删除资源
```

（2）路径

在 RESTful 架构中，每一个网址代表一种资源，所以网址中只能用名词，且所用名词与数据库表格名对应。

```
http://webdesign.hyit.online/students
```

（3）版本

URL 中嵌入版本编号，可以在引入新版本 API 的同时确保旧版本 API 仍然可用。

```
http://webdesign.hyit.online/v1/students
```

（4）域名

将 API 部署在专用域名之下

```
http://api/webdesign.hyit.online.com
```

若 API 简单，无进一步扩展，可以放在主域名下。

```
http://webdesign.hyit.online/api/v1/students
```

（5）HTTPS 协议

谷歌浏览器对所有不是 HTTPS 请求的链接全都会提示用户此链接为不安全链接，腾讯等平台也对小程序等产品强制要求使用 HTTPS 协议。

```
https://webdesign.hyit.online/api/v1/students
```

（6）状态码（Status Codes）

服务器向用户返回的状态码和提示信息，常见的有以下一些（方括号中是该状态码对应的 HTTP 动词）。

```
200 OK - [GET]：服务器成功返回用户请求的数据，该操作是幂等的（Idempotent）。
201 CREATED - [POST/PUT/PATCH]：用户新建或修改数据成功。
```

202 Accepted - [*]：表示一个请求已经进入后台排队（异步任务）

204 NO CONTENT - [DELETE]：用户删除数据成功。

400 INVALID REQUEST - [POST/PUT/PATCH]：用户发出的请求有错误，服务器没有进行新建或修改数据的操作，该操作是幂等的。

401 Unauthorized - [*]：表示用户没有权限（令牌、用户名、密码错误）。

403 Forbidden - [*] 表示用户得到授权（与 401 错误相对），但是访问是被禁止的。

404 NOT FOUND - [*]：用户发出的请求针对的是不存在的记录，服务器没有进行操作，该操作是幂等的。

406 Not Acceptable - [GET]：用户请求的格式不可得（比如用户请求 JSON 格式，但是只有 XML 格式）。

410 Gone -[GET]：用户请求的资源被永久删除，且不会再得到的。

422 Unprocesable entity - [POST/PUT/PATCH] 当创建一个对象时，发生一个验证错误。

500 INTERNAL SERVER ERROR - [*]：服务器发生错误，用户将无法判断发出的请求是否成功。

（7）过滤信息（Filtering）

API 的参数中加入筛选条件参数，实现资源过滤，下面是一些常见的参数。

?limit=10：指定返回记录的数量

?offset=10：指定返回记录的开始位置。

?page=2&per_page=100：指定第几页，以及每页的记录数。

?sortby=id&order=asc：指定返回结果按照 id 排序，以及顺序排序。

?students_id=1：指定筛选条件

（8）返回结果

针对不同操作，服务器向用户返回的结果应该符合以下规范。

GET /collection：返回资源对象的列表（数组）

GET /collection/resource：返回单个资源对象

POST /collection：返回新生成的资源对象

PUT /collection/resource：返回完整的资源对象

PATCH /collection/resource：返回完整的资源对象

DELETE /collection/resource：返回一个空文档

请求方法	请求地址	后端操作
GET	https://webdesign.hyit.online/api/v2/students	获取所有学生信息
POST	https://webdesign.hyit.online/api/v2/students	增加学生信息
GET	https://webdesign.hyit.online/api/v2/students/1201317130	获取学号为 1201317130 的学生信息
PUT	https://webdesign.hyit.online/api/v2/students/1201317130	修改学号为 1201317130 的学生信息
DELETE	https://webdesign.hyit.online/api/v2/students/1201317130	删除学号为 1201317130 的学生信息

（9）错误处理

返回的详细信息中加入错误代号 code，并以字典的形式放在 data 中。

```
ret {
```

```
code:1000，
data:{
    {'id':1,'title':Python 程序设计'
      'detail':https://webdesign.hyit.online/}
    }
}
```

14.2 准备项目开发环境

本项目的开发环境以及安装方法如下：

（1） 安装 Python

本项目的开发环境需使用 Python，可以从其官网中下载并安装。下载地址：https://www.python.org/downloads/windows/，选择下载稳定 Python 3.9.2 Windows installer(64-bit)如图 14-3 所示。

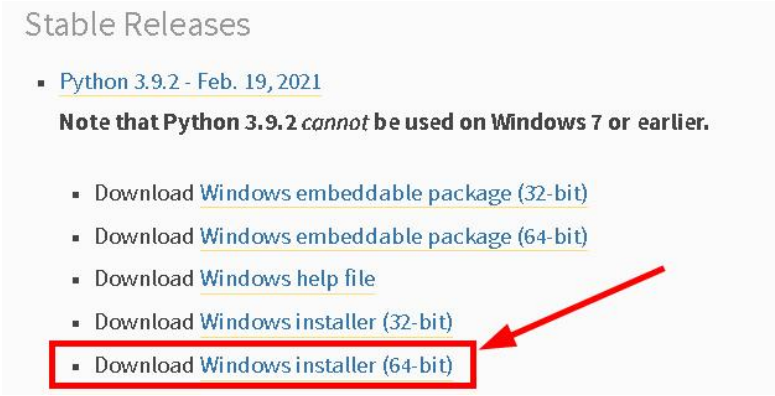

图 14-3 本项目开发使用的 Python 安装包

下载完成后，Python 的 Windows 安装包以 python-3.9.2-amd64.exe 方式命名。直接运行该程序可以进入 Python 的安装界面，如图 14-4 所示。

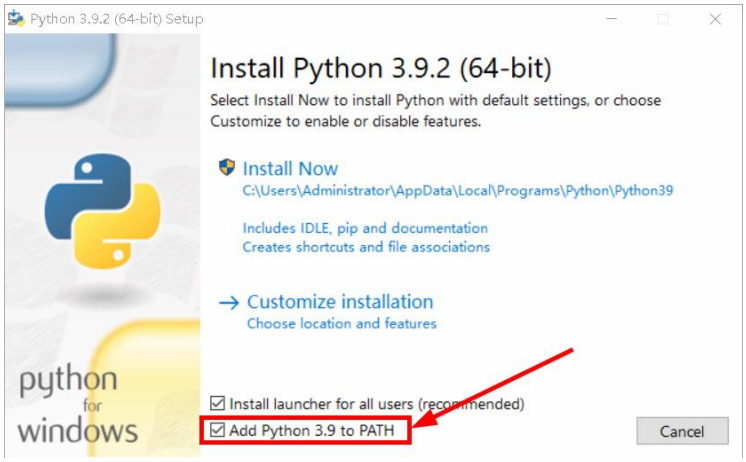

图 14-4 Python 在 Windows 中的安装界面

勾选对话框下方的“Add Python 3.9 to PATH”复选框，然后单击“Install Now”按钮继续安装，Python 在 Windows 中的安装过程如图 14-5 所示。

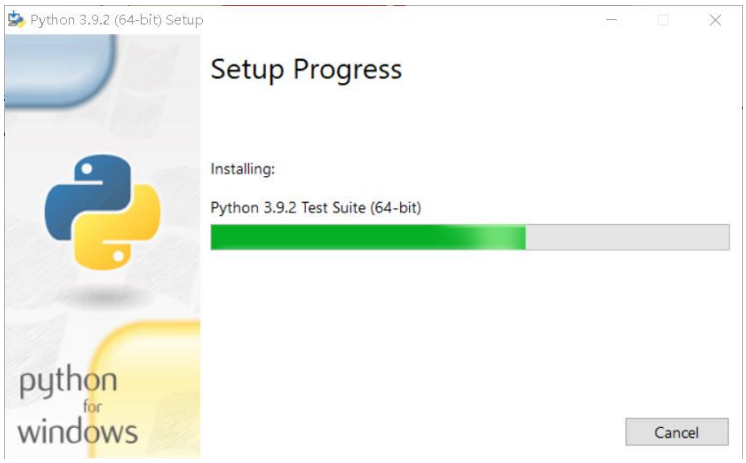

图 14-5　Python 在 Windows 中的安装过程

安装完成后，Python3.9 的可执行程序路径就会被自动加入 Windows 环境变量中。可以通过环境变量对话框查看确认，具体查找路径为：【计算机】|【属性】|【高级系统设置】|【用户变量】对话框中的【Path】变量，如图 14-6 所示。

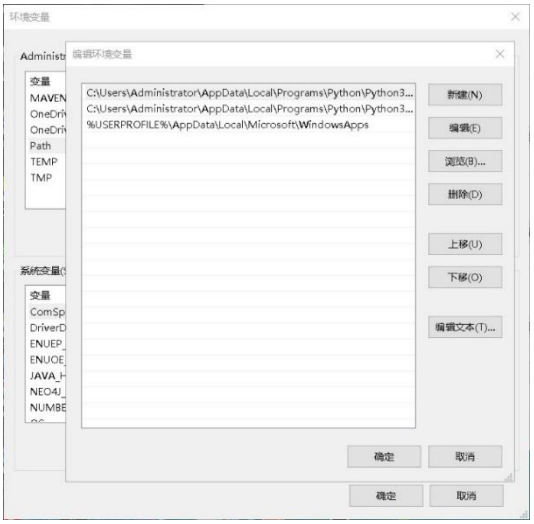

图 14-6　在 Windows 中设置 PATH 环境变量

在 Path 环境变量中添加 Python 的路径后，打开 Windows 命令行工具，执行 python 命令，如果输出如图 14-7 所示的内容，说明 Python 已经安装成功。

```
Microsoft Windows [版本 10.0.19041.804]
(c) 2020 Microsoft Corporation. 保留所有权利。

C:\Users\Administrator>python
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

图 14-7　测试 Python 运行环境是否安装成功

（2）　安装 PyCharm

本项目的开发平台使用 PyCharm 编辑器。PyCharm 分为社区版和专业版，我们选择下载专业版。

进入 PyCharm 下载页面，下载专业版安装包，如图 14-8 所示。

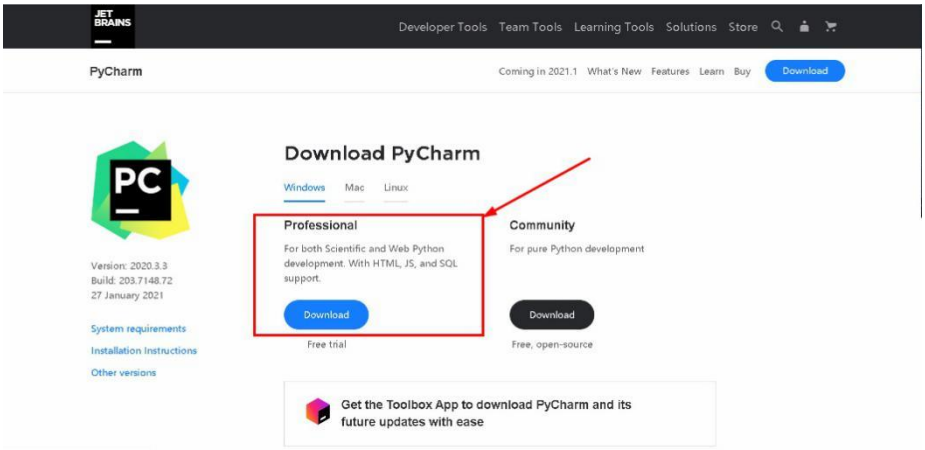

图 14-8　PyCharm 专业版下载

下载好安装包，双击进行安装，选择安装路径，关联.py 文件，单击“Next”按钮安装，如图 14-9 所示。

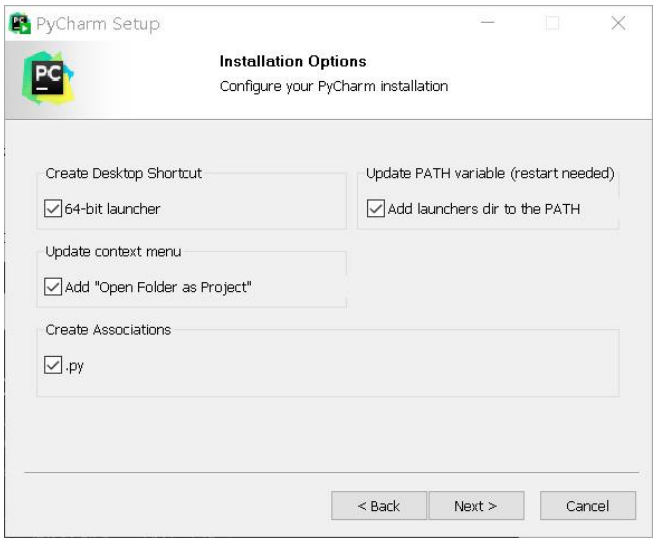

图 14-9　PyCharm 安装选项

本项目开发使用的 PyCharm 平台 Version 信息如图 14-10 所示。

图 14-10 本项目开发使用的 PyCharm 平台 Version 信息

（3）　安装 Django

Django 是 Python 语言的一个类库，可以通过 pip 工具安装。进入 python 安装目录下的 Scripts 目录，打开 Windows 命令行工具，以管理员身份运行命令：

```
pip install django
```

系统将自动安装 PyPi 提供的最新版本。成功安装 Django 之后，进入 Python 交互式环境，按下面所示查看安装版本。

```
>>> import django
>>> django.get_version()
3.1.2
```

Django 安装完成后，在 Python 解释器目录下的 Scripts 文件夹中可找到一个 django-admin.exe 文件，这是 Django 的核心管理程序，需将它加入操作系统的环境变量中，方便调用。

在 Path 变量中添加 Scripts 目录，如图 14-11 所示。

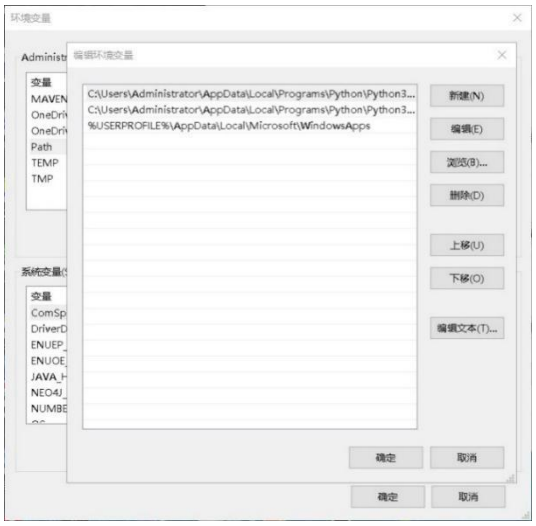

图 14-11　设置 Django 环境变量

环境变量设置完成后，回到命令行工具界面，运行命令 django-admin help，如果输出如下内容，说明 Django 已经安装成功。

```
D:\webdesign-hyit\Python38\Scripts>django-admin help
Type 'django-admin help <subcommand>' for help on a specific subcommand.
Available subcommands:
[django]
    check
    compilemessages
    createcachetable
    dbshell
...以下省略
```

（4）　安装 Django REST framework

打开 windows 命令行工具，执行如下命令。

```
pip install djangorestframework
```

（5）　安装 Virtualenv

Virtualenv 可用于创建独立的 Python 环境，在这些环境里面可以选择不同的 Python 版本或者不同的 Packages，并且可以在没有 root 权限的情况下在环境里安装新套件，互相不会产生任何的影响。

打开 windows 命令行工具，执行如下命令。

```
pip install virtualenv
```

（6） 安装 HbuilderX

后续的实验案例中需使用 HbuilderX 作为网页开发工具，因此需从官网（https://www.dcloud.io/hbuilderx.html）下载其安装包。进入 HbuilderX 下载页面，下载标准版压缩包，如图 14-12 所示。

图 14-12 HbuilderX 下载页面

下载完成后，直接解压 HbuilderX 压缩包完成安装。

14.3 项目创建

14.3.1 项目创建

打开 Pycharm，依次点击“file”-“new project”，选择左边的“Django”，出现对话框如图 14-13 所示。

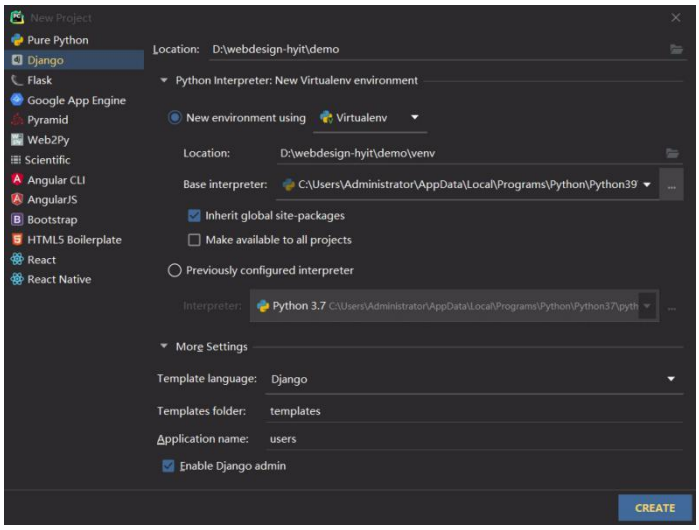

图 14-13 新建 Django 项目

在“Location”处选择工程目录位置。

在“New Virtualenv environment”处选择“Virtualenv”，虚拟环境会以 venv 的名字，自动在工程目录下生成。

在“Base interpreter”处，选择要使用的 Python 解释器。

下方两个单选框，勾选第一个“Inhert global site-packages”单选框，不然无法使用系统库。

如果想使用现成的解释器或者虚拟环境，请选择“Previously configured interpreter”。

在“More Settings”里选择使用的模板语言，默认 Django。

14.3.2 项目目录结构

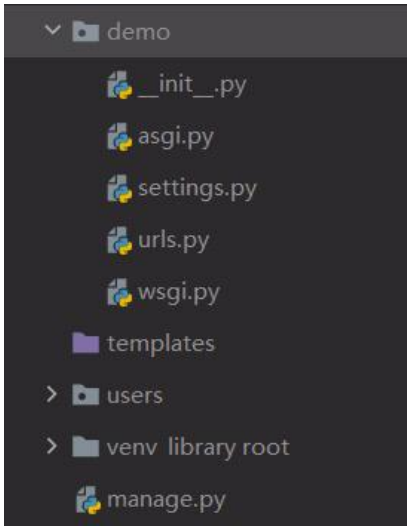

图 14-14　项目目录结构

asgi.py：项目与 ASGI 兼容的 Web 服务器入口；

settings.py：项目的整体配置文件；

urls.py：项目的 URL 配置文件；

wsgi.py：项目与 WSGI 兼容的 Web 服务器入口；

manage.py ：项目管理文件，通过它管理项目。

14.3.3 运行开发服务器

在开发阶段，为了能够快速预览到开发的效果，Django 提供了一个 Python 编写的轻量级 web 服务器，仅在开发阶段使用。

运行服务器命令如下：

```
python manage.py runserver ip:端口
或
python manage.py runserver
```

可不写 IP 和端口，默认 IP 是 127.0.0.1，默认端口为 8000，启动后如图 14-15 所示。

```
Terminal:  Local  +
System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
January 29, 2021 - 12:58:08
Django version 3.1.5, using settings 'djangoProject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

图 14-15　启动项目

在浏览器中输入地址 http://localhost:8080/，或者单击该地址在浏览器中预览项目运行效果，如图 14-16 所示。

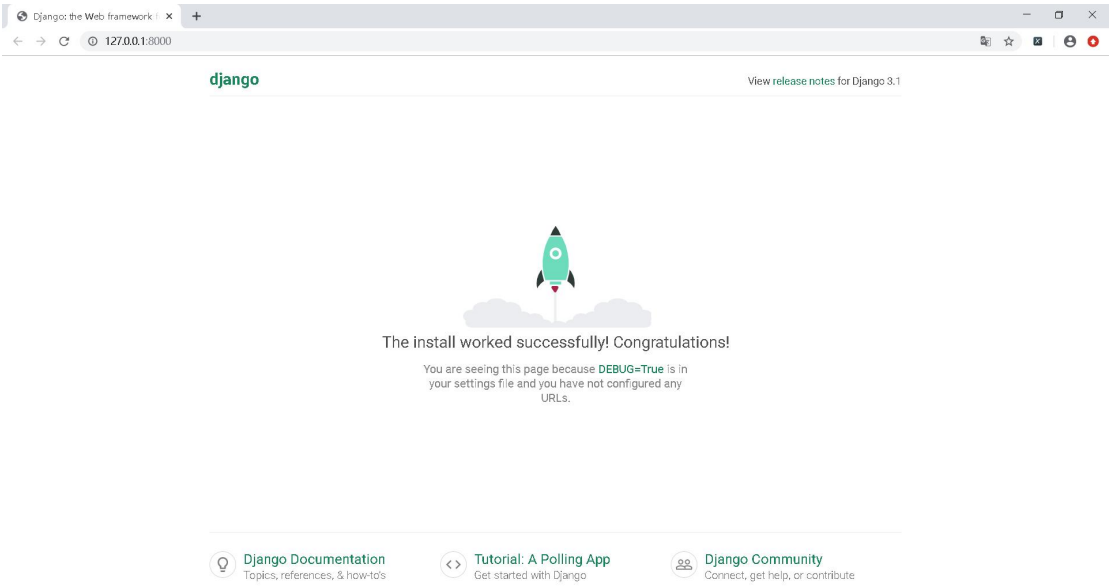

图 14-16 项目运行成功

14.4 Vue 实现 API 应用

创建 API 项目：打开 HBuilderX 应用程序，点击左上角文件，依次选择“新建”-“项目”，出现如图 14-17 所示对话框。

图 14-17 新建项目

项目类型选择“普通项目”，模板类型选择“基于 HTML 项目”，点击“创建”按钮新建项目。项目创建成功之后，应用程序左侧会出现如图 14-18 所示目录。

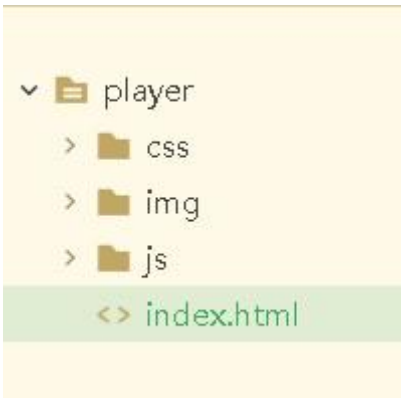

图 14-18　项目目录

选择 js 目录，右击新建 main.js 文件，出现如图 14-19 所示对话框。

图 14-19　创建 main. js 文件

点击“创建”按钮创建 js 文件。

本项目共使用 3 个 js 文件和 1 个 css 文件，剩下的文件详情如图 14-20 所示。

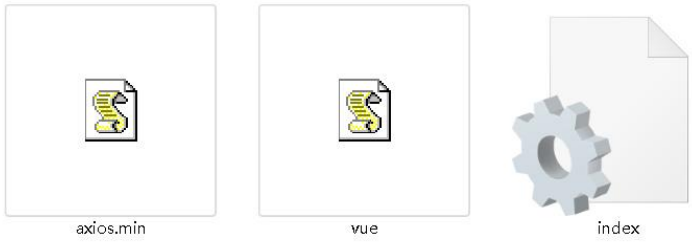

图 14-20　文件素材

将上述 js 和 css 文件分别复制到 js 和 css 文件夹下。

本项目共使用 8 张图片，详情如图 14-21 所示。

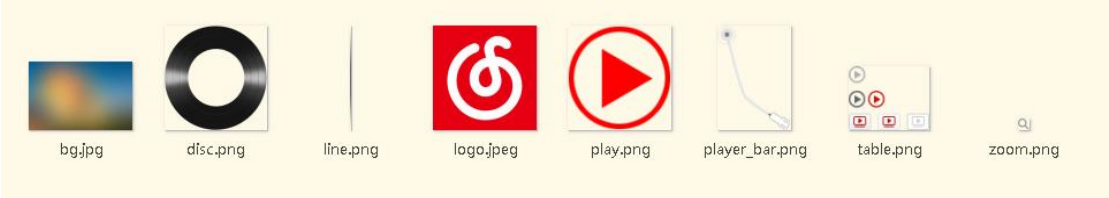

图 14-21　图片素材

将上述图片复制到 img 文件夹下。

14.5　项目页面设计

播放器由搜索歌曲、歌曲列表、歌曲信息、评论容器、播放器、MV 六个部分组成，播放器静态界面如图 14-22 所示。

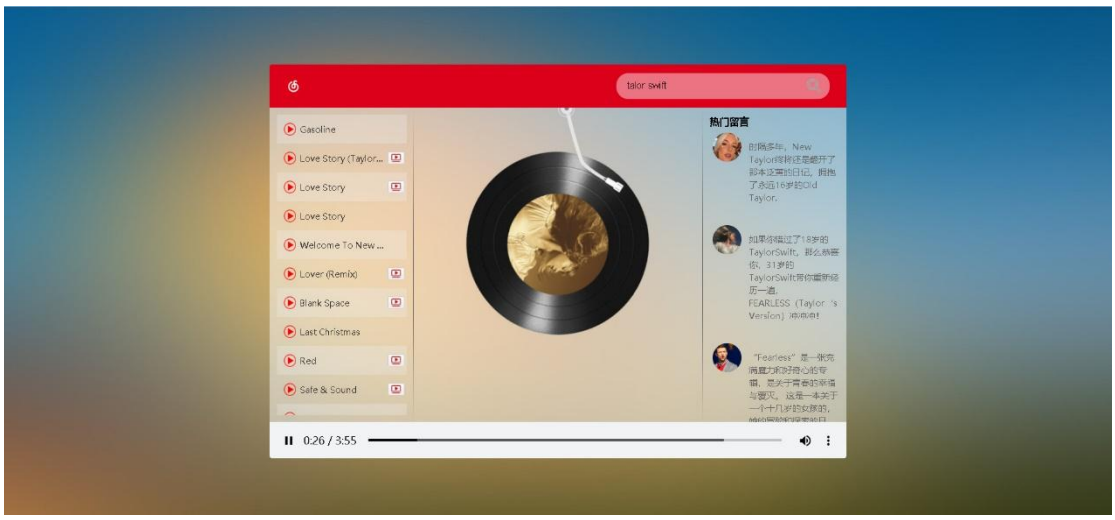

图 14-22　播放器静态界面

点击打开目录下的 index.html，开始设计页面。初始的 html 代码里面由<html>、<head>、<meta>、<title>和<body>五个标签。<head>标签用于定义文档的头部，它是所有头部元素的容器，文档的头部描述了文档的各种属性和信息，包括文档的标题、在 Web 中的位置以及和其他文档的关系等，绝大多数文档头部包含的数据都不会真正作为内容显示给读者。<title> 元素可定义文档的标题。<body>元素包含文档的所有内容（比如文本、超链接、图像、表格和列表等等）。

1. 设计头内容

在<title>元素里设置网页标题为“player”；使用<link> 标签链接新创建的 index.css 文件，其中 rel="stylesheet"表示调用的是样式，href="./css/index.css"表示外部样式文件的路径。

```
    <title>player</title>
    <link rel="stylesheet" href="./css/index.css">
```

2. 引入 js 文件

在<body>标签下定义 3 个<script>标签，src 表示 js 文件存放路径。

```
    <script src="./js/vue.js"></script>
    <script src="./js/axios.min.js"></script>
    <script src="./js/main.js"></script>
```

3. 页面整体布局

页面可分为背景墙和播放器主体区域，播放器里面分为三块，由上而下分别为搜索栏、歌曲详情栏和播放器控制栏。中间部分为三栏，从左到右分别为歌曲列表、歌曲信息和评论。

先使用 标签设置整个背景墙，其中 class="wrap"表示调用 css 样式表中的 wrap 类。

```
<div class="wrap">
<!-- 播放器主体区域 -->
    <div class="play_wrap" id="player">
    </div>
</div>
```

在背景墙 中定义 标签，标签里的内容为播放器主体和 MV，其中 class="play_wrap"表示调用 css 样式表中的 play_wrap 类，id="player"用于元素定位。

```
<div class="wrap">
    <div class="play_wrap" id="player">
      <!-- 搜索栏 -→
      <div class="search_bar">
      </div>
      <!-- 歌曲详情栏 -→
      <div class='center_con'>
      </div>
      <!-- 播放器控制栏 -→
      <div class=" audio_con">
      </div>
      <!-- MV -→
      <div class=" video_con">
      </div>
    </div>
</div>
```

4. 搜索栏

定义 标签，其中 class="search_bar"表示调用 css 样式表中的 search_bar 类。在 标签中定义<img>和<input>标签，<img>标签中 src 表示图片存放路径；<input>标签中 type="text"表示输入为文本，autocomplete="off"表示不启用自动完成功能，v-model 实现标签数据的双向绑定，@keyup.enter 监听事件实现歌曲搜索。

```
<div class="search_bar">
        <img src="img/logo.jpeg" style="width:1.25rem;" alt="" />
        <input type="text" autocomplete="off" v-model="query"
        @keyup.enter="searchMusic" />
</div>
```

5. 歌曲详情栏

定义 标签，其中 class='song_wrapper'表示调用 css 样式表中的 song_wrapper 类。在 标签中定义<ul>和<img>标签，<ul>标签中 class="song_list"表示调用 css 样式表中的 song_list 类，<img>标签中 src 表示图片存放路径，class="switch_btn"表示调用 css 样式表中的 switch_btn 类；在<ul>标签中定义<li>列表标签，列表渲染指令 v-for 用于获取列表信息；在<li>标签中定义<a>、<b>和标签，<a>标签中 href="javascript:;"表示在触发默认动作时，不执行 JavaScript 代码，@click="playMusic(item.id)监听事件实现播放歌曲；<b>标签中 item.name 用于获取歌曲名字；标签中 v-if="item.mvid!=0 实现元素的重建或销毁，@click="playMV(item.mvid)监听事件实现播放 MV。

```
<div class='song_wrapper'>
        <ul class="song_list">
          <li v-for="item in musicList">
            <a href="javascript:;" @click="playMusic(item.id)"></a>
            <b>{{ item.name }}</b>
            <span v-if="item.mvid!=0" @click="playMV(item.mvid)"><i></i></span>
          </li>
```

```
            </ul>
            <img src="img/line.png" class="switch_btn" alt="">
    </div>
```

定义 标签，其中 class=' player_con '表示调用 css 样式表中的 player_con 类，:class="{playing:isPlaying}用于判断 playing 是否作用于该 div 元素上，数据由 isPlaying 决定；在 标签中定义 3 个<img>标签，src 表示图片存放路径，class="play_bar"、class="disc autoRotate"和 class="cover autoRotate"分别表示调用 css 样式表中的 play_bar、disc autoRotate 和 cover autoRotate 类，:src="musicCover"用于动态绑定歌曲封面 url。

```
    <div class="player_con" :class="{playing:isPlaying}">
            <img src="img/player_bar.png" class="play_bar" />
            <img src="img/disc.png" class="disc autoRotate" />
            <img :src="musicCover" class="cover autoRotate" />
    </div>
```

定义 标签，其中 class=' comment_wrapper '表示调用 css 样式表中的 player_con 类。在 标签中定义<h5>、 和<img>标签，<h5>标签中 class='title'表示调用 css 样式表中的 title 类；<h5>标签中 class=' comment_list '表示调用 css 样式表中的 comment_list 类；<img>标签中 class=' right_line '表示调用 css 样式表中的 right_line 类，src 表示图片存放路径。使用列表渲染指令 v-for 获取评论信息，其中包括头像、网名和评论详情。在二级 标签中定义<dl>标签，其中列表渲染指令 v-for 用于获取列表信息。在<dl>标签中定义<dt>和<dd>标签，<dt>标签中定义<img>标签，其中:src="item.user.avatarUrl"用于动态绑定用户图片 url，<dd>标签中的 class="name"和 class="detail"分别表示调用 css 样式表中的 name 和 detail 类，item.nickname 和 item.nickname 分别用于获取用户名字和评论详情。

```
     <div class="comment_wrapper">
            <h5 class='title'>热门留言</h5>
            <div class='comment_list'>
               <dl v-for="item in hotComments">
                  <dt><img :src="item.user.avatarUrl" alt=""></dt>
                  <dd class="name">{{ item.nickname}}</dd>
                  <dd class="detail">
                     {{ item.content }}
                  </dd>
               </dl>
            </div>
            <img src="img/line.png" class="right_line">
    </div>
```

6. 播放器

定义 标签，其中 class=' audio_con '表示调用 css 样式表中的 audio_con 类；在 标签中定义<audio>音频标签，其中 class="myaudio"表示调用 css 样式表中的 myaudio 类，@play="play" 和@pause="pause"监听事件分别实现播放和暂停，:src="musicUrl"用于动态绑定歌曲 url。

```
    <div class="audio_con">
        <audio class="myaudio" ref='audio' @play="play" @pause="pause" :src="musicUrl"
           controls autoplay loop >
        </audio>
```

```
</div>
```

7. 播放 MV

定义 标签，其中 class="video_con"表示调用 css 样式表中的 video_con 类，v-show 用于控制元素显隐。在 标签中定义<video>和 标签，<video>标签中 v-show 用于控制元素显隐，:src="mvUrl"用于动态绑定 MVurl； 标签中 class="mask"表示调用 css 样式表中的 mask 类，@click="hide"监听事件实现 MV 界面的退出。

```
<div class="video_con" v-show="isShow" style="display: none;">
        <video :src="mvUrl" controls="controls"></video>
        <div class="mask" @click="hide"></div>
</div>
```

14.6 项目功能模块

点击打开文件目录下的 main.js，开始实现功能模块。

14.6.1 歌曲搜索

（1）歌曲搜索接口

请求地址:https://autumnfish.cn/search

请求方法:get

请求参数:keywords(查询关键字)

响应内容:歌曲搜索结果

（2）功能实现

通过 el 设置挂载元素，元素我们通过审查元素的方法找到，具体元素定位如图 14-23 所示。

图 14-23 挂载元素定位

在 data 属性里添加 query 和 musicList 字段，内容分别为字符串和数组。

在 methods 方法里定义 searchMusic 方法，通过 axios.get(地址+参数)的方法来获取歌曲信息；添加两个回调函数，在第一个回调函数中打印网易云服务器返回的歌曲信息。

```
var app = new Vue({
  el: "#player",
  data: {
    query: "",
    musicList: [],
  },
  methods: {
    searchMusic: function() {
```

```
            axios.get("https://autumnfish.cn/search?keywords=" + this.query).then(
                function(response) {
                    console.log(response);
                },
                function(err) {}
            );
        },
    });
```

运行项目服务器，在搜索框输入歌手名字，网易云服务器返回的信息如图 14-24 所示。

图 14-24 网易云服务器返回的歌曲信息

从图中可以看出歌曲信息都放在“data”-“result”-“songs”，重新修改第一个回调函数，另外添加新的变量来保存 this 的值。

```
            var that = this;
            axios.get("https://autumnfish.cn/search?keywords=" + this.query).then(
                function(response) {
                    that.musicList = response.data.result.songs;
                },
```

14.6.2 歌曲播放

（1）歌曲 url 获取接口

请求地址:https://autumnfish.cn/song/url

请求方法:get

请求参数:id(歌曲 id)

响应内容:歌曲 url 地址

（2）功能实现

在 data 属性里添加 musicUrl 字段，内容为字符串。

在 methods 方法里定义 playMusic 方法，设置行参 musicId，通过 axios.get(地址+参数)的方法来获取歌曲信息；添加两个回调函数，在第一个回调函数中打印服务器返回的信息。

```
    playMusic: function(musicId) {
            var that = this;
```

```
        axios.get("https://autumnfish.cn/song/url?id=" + musicId).then(
            function(response) {
                console.log(response);
            },
            function(err) {}
        );
```

运行项目服务器，播放歌曲，网易云服务器返回的信息如图 14-25 所示。

图 14-25 网易云服务器返回的歌曲 url 地址

从图中可以看出歌曲 url 地址放在“data”-“data[0]”-“url”，需要修改第一个回调函数，另外添加新变量来保存 this 的值。

```
var that = this;
axios.get("https://autumnfish.cn/song/url?id=" + musicId).then(
    function(response) {
        that.musicUrl = response.data.data[0].url;
},
```

14.6.3 歌曲封面

（1）歌曲详情获取

请求地址:https://autumnfish.cn/song/detail

请求方法:get

请求参数:ids(歌曲 id)

响应内容:歌曲详情(包括封面信息)

（2）功能实现

在 data 属性里添加 musicCover 字段，内容为字符串。

歌曲详情和播放歌曲同时进行，因此在 playMusic 方法里添加 axios.get(地址+参数)的方法来获取歌曲详情信息；设置两个回调函数，在第一个回调函数中打印服务器返回的信息。

```
axios.get("https://autumnfish.cn/song/detail?ids=" + musicId).then(
    function(response) {
        console.log(response);
        },
    function(err) {}
 );
```

运行项目服务器，播放歌曲，网易云服务器返回的信息如图 14-26 所示。

图 14-26 网易云服务器返回的封面信息 url 地址

从图中可以看出歌曲详情信息地址放在“data”-“songs[0]”-“al”-“picUrl”，需要修改第一个回调函数。

```
that.musicCover = response.data.songs[0].al.picUrl;
```

下面实现歌曲封面动画，在 data 属性里定义 isPlaying 字段，默认为 false；新建两个方法，分别为 play 和 pause，用来修改 isPlaying 的值 。

```
play: function() {
        this.isPlaying = true;
    },
    pause: function() {
        this.isPlaying = false;
    },
```

14.6.4 歌曲评论

（1）热门评论获取

请求地址:https://autumnfish.cn/comment/hot?type=0

请求方法:get

请求参数:id(歌曲 id,地址中的 type 固定为 0)

响应内容:歌曲的热门评论

（2）功能实现

在 data 属性里添加 hotComments 字段，内容为数组。

歌曲评论和播放歌曲应同时进行，因此在 playMusic 方法里添加 axios.get(地址+参数)的方法来获取歌曲评论信息；设置两个回调函数，在第一个回调函数中打印服务器返回的信息。

```
axios.get("https://autumnfish.cn/comment/hot?type=0&id=" + musicId).then(
            function(response) {
                console.log(response);
            },
            function(err) {}
        );
```

运行项目服务器，播放歌曲，网易云服务器返回的信息如图 14-27 所示。

图 14-27 网易云服务器返回的评论信息

从图中可以看出歌曲详情信息地址放在“data”-“hotComments”，需要修改第一个回调函数。

```
that.hotComments = response.data.hotComments;
```

14.6.5 播放 MV

（1）MV 地址获取

请求地址:https://autumnfish.cn/mv/url

请求方法:get

请求参数:id(mvid,为 0 表示没有 mv)

响应内容:MV 地址

（2）功能实现

在 data 属性里添加 mvUrl 字段，内容为字符串。

在 methods 方法里定义 playMV 方法，设置行参 mvId，通过 axios.get(地址+参数)的方法来获取歌曲 MV 地址；添加两个回调函数，在第一个回调函数中打印服务器返回的信息。

```
playMV: function(mvid) {
        axios.get("https://autumnfish.cn/mv/url?id=" + mvid).then(
          function(response) {
              console.log(response);
          },
          function(err) {}
        );
      },
```

运行项目服务器，播放 MV，网易云服务器返回的信息如图 14-28 所示。

图 14-28 网易云服务器返回的 MVurl 地址

从图中可以看出歌曲详情信息地址放在“data”-“data”-“url”，需要修改第一个回调函数，另外添加新变量来保存 this 的值。

```
var that = this;
```

```
axios.get("https://autumnfish.cn/mv/url?id=" + mvid).then(
    function(response) {
        that.isShow = true;
        that.mvUrl = response.data.data.url;
},
```

下面实现 MV 播放界面的隐显。在 data 属性里定义 isShow 字段，默认为 false；当播放 MV 时，需将 isShow 设置为 ture，因此在回调函数中修改 isShow 的值；另外新建一个 hide 方法，用来隐藏 MV 播放界面。

```
hide: function() {
        this.isShow = false;
    }
```

14.7 项目代码

index.html 部分代码如下：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>player</title>
    <link rel="stylesheet" href="./css/index.css">
</head>
<body>
    <div class="wrap">
        <!-- 播放器主体区域 -->
        <div class="play_wrap" id="player">
            <div class="search_bar">
                <img src="img/logo.jpeg" style="width:1.25rem;" alt="" />
                <!-- 搜索歌曲 -->
                    <input type="text" placeholder="输入歌手" autocomplete="off"
                    v-model="query" @keyup.enter="searchMusic" />
            </div>
            <div class="center_con">
                <!-- 搜索歌曲列表 -->
                <div class='song_wrapper'>
                    <ul class="song_list">
                        <li v-for="item in musicList">
                            <a href="javascript:;" @click="playMusic(item.id)"></a>
                            <b>{{ item.name }}</b>
                            <span v-if="item.mvid!=0" @click="playMV(item.mvid)"><i></i></span>
                        </li>
                    </ul>
                    <img src="img/line.png" class="switch_btn" alt="">
                </div>
                <!-- 歌曲信息容器 -->
```

```
                <div class="player_con" :class="{playing:isPlaying}">
                    <img src="img/player_bar.png" class="play_bar" />
                    <img src="img/disc.png" class="disc autoRotate" />
                    <img :src="musicCover" class="cover autoRotate" />
                </div>
                <!-- 评论容器 -->
                <div class="comment_wrapper">
                    <h5 class='title'>热门留言</h5>
                    <div class='comment_list'>
                        <dl v-for="item in hotComments">
                            <dt><img :src="item.user.avatarUrl" alt=""></dt>
                            <dd class="name">{{ item.nickname}}</dd>
                            <dd class="detail">
                                {{ item.content }}
                            </dd>
                        </dl>
                    </div>
                    <img src="img/line.png" class="right_line">
                </div>
            </div>
            <div class="audio_con">
                <audio ref='audio' @play="play" @pause="pause" :src="musicUrl" controls
                    autoplay loop class="myaudio"></audio>
            </div>
            <div class="<video_con></video_con>" v-show="isShow" style="display: none;">
                <video :src="mvUrl" controls="controls"></video>
                <div class="mask" @click="hide"></div>
            </div>
        </div>
    </div>
    <script src="./js/vue.js"></script>
    <script src="./js/axios.min.js"></script>
    <script src="./js/main.js"></script>
</body>
</html>
```

main.js 部分代码如下：

```
var app = new Vue({
    el: "#player",
    data: {
        query: "",
        musicList: [],
        musicUrl: "",
        musicCover: "",
```

```
        hotComments: [],
        isPlaying: false,
        isShow: false,
        mvUrl: ""
    },
    methods: {
        searchMusic: function() {
            var that = this;
            axios.get("https://autumnfish.cn/search?keywords=" + this.query).then(
                function(response) {
                    //console.log(response);
                    that.musicList = response.data.result.songs;
                },
                function(err) {}
            );
        },
        playMusic: function(musicId) {
            var that = this;
            axios.get("https://autumnfish.cn/song/url?id=" + musicId).then(
                function(response) {
                    //console.log(response);
                    that.musicUrl = response.data.data[0].url;
                },
                function(err) {}
            );
            axios.get("https://autumnfish.cn/song/detail?ids=" + musicId).then(
                function(response) {
                    //console.log(response);
                    that.musicCover = response.data.songs[0].al.picUrl;
                },
                function(err) {}
            );
            axios.get("https://autumnfish.cn/comment/hot?type=0&id=" + musicId).then(
                function(response) {
                    //console.log(response);
                    that.hotComments = response.data.hotComments;
                },
                function(err) {}
            );
        },
        play: function() {
            this.isPlaying = true;
        },
        pause: function() {
```

```
            this.isPlaying = false;
        },
        playMV: function(mvid) {
            var that = this;
            axios.get("https://autumnfish.cn/mv/url?id=" + mvid).then(
                function(response) {
                    //console.log(response);
                    that.isShow = true;
                    that.mvUrl = response.data.data.url;
                },
                function(err) {}
            );
        },
        hide: function() {
            this.isShow = false;
        }
    }
});
```