Spring Boot + Vue 如此强大？竟然可以开发基于 C/S 架构的应用

前言

虽然 B/S 是目前开发的主流，但是 C/S 仍然有很大的市场需求。受限于浏览器的沙盒限制，网页应用无法满足某些场景下的使用需求，而桌面应用可以读写本地文件、调用更多系统资源，再加上 Web 开发的低成本、高效率的优势，这种跨平台方式越来越受到开发者的喜爱。

Electron 是一个基于 Chromium 和 Node.js，使用 HTML、CSS 和 JavaScript 来构建跨平台应用的跨平台开发框架，兼容 Mac、Windows 和 Linux。目前，Electron 已经创建了包括 VScode 和 Atom 在内的大量应用。

环境搭建

创建 Electron 跨平台应用之前，需要先安装一些常用的工具，如 Node、vue 和 Electron 等。

安装 Node

进入 Node 官网下载页 http://nodejs.cn/download/，然后下载对应的版本即可，下载时建议下载稳定版本。如果安装 Node 使用 Homebrew 方式，建议安装时将 npm 仓库镜像改为淘宝镜像，如下所示。

```
npm config set registry http://registry.npm.taobao.org/
```

或者

```
npm install -g cnpm --registry=https://registry.npm.taobao.org
```

安装/升级 vue-cli

先执行以下命令，确认下本地安装的 vue-cli 版本。

```
vue -V
```

如果没有安装或者不是最新版，可以执行以下命令安装/升级。

```
npm install @vue/cli -g
```

安装 Electron

使用如下命令安装 Electron 插件。

```
npm install -g electron
```

或者

```
cnpm install -g electron
```

为了验证是否安装成功，可以使用如下的命令。

```
electron --version
```

创建运行项目

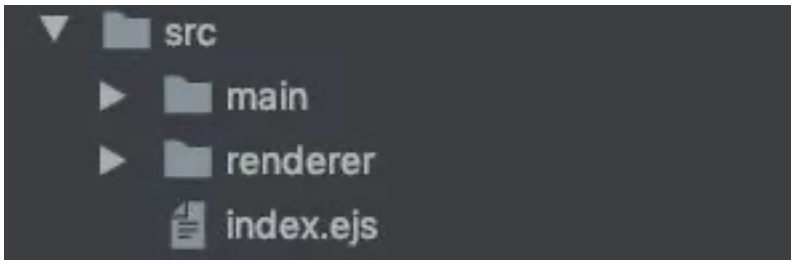
Electron 官方提供了一个简单的项目，可以执行以下命令将项目克隆到本地。

```
git clone https://github.com/electron/electron-quick-start
```

然后在项目中执行如下命令即可启动项目。

```
cd electron-quick-start
npm install
npm start
```

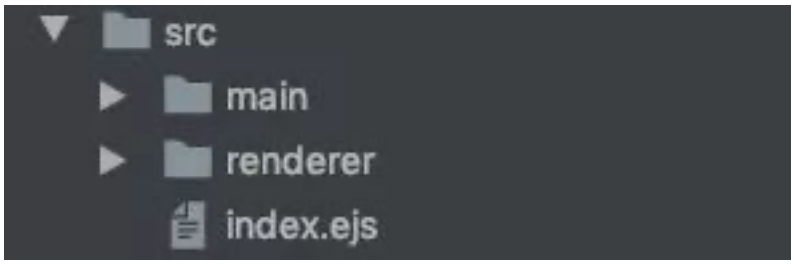
启动后项目的效果如下图。



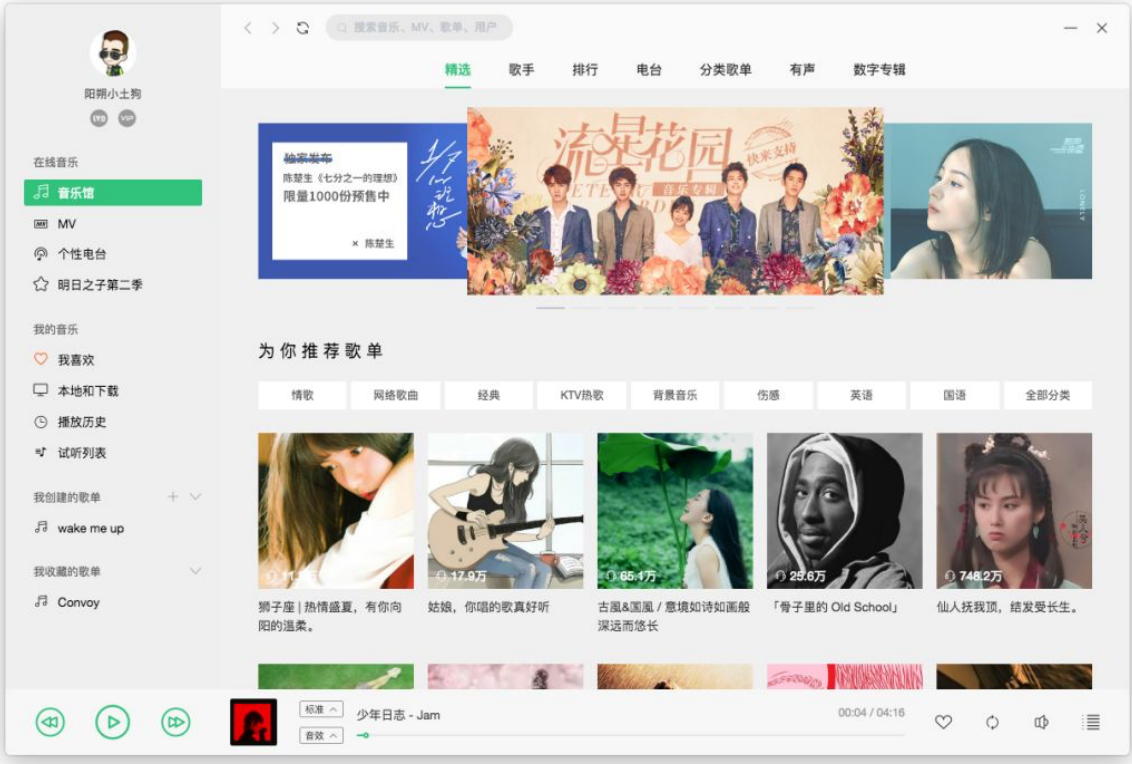
除此之外，我们可以使用 vue-cli 脚手架工具来创建项目。

```
vue init simulatedgreg/electron-vue
```

然后根据下面的提示一步步选中选项即可创建项目，如下所示。



然后，使用 npm install 命令安装项目所需要的依赖包，安装完成之后，可以使用 npm run dev 或 npm run build 命令运行 electron-vue 模版应用程序，运行效果如下图所示。



Electron 源码目录

Electron 的源代码主要依据 Chromium 的拆分约定被拆成了许多部分。为了更好地理解源代码，您可能需要了解一下 Chromium 的多进程架构。

Electron 源码目录结构和含义具体如下：

```
Electron
├──atom - Electron 的源代码
|   ├── app - 系统入口代码
|   ├── browser - 包含了主窗口、UI 和其他所有与主进程有关的东西，它会告诉渲染进程如何管理页面
|   |   ├── lib - 主进程初始化代码中 JavaScript 部分的代码
|   |   ├── ui - 不同平台上 UI 部分的实现
|   |   |   ├── cocoa - Cocoa 部分的源代码
|   |   |   ├── gtk - GTK+ 部分的源代码
|   |   |   └── win - Windows GUI 部分的源代码
|   |   ├── default_app - 在没有指定 app 的情况下 Electron 启动时默认显示的页面
|   |   ├── api - 主进程 API 的实现
|   |   |   └── lib - API 实现中 Javascript 部分的代码
|   |   ├── net - 网络相关的代码
|   |   ├── mac - 与 Mac 有关的 Objective-C 代码
```

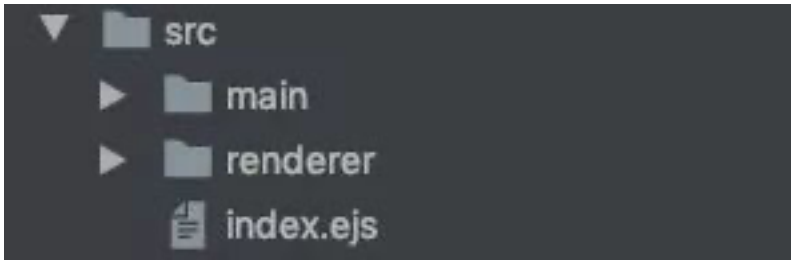
```
|   |   └── resources - 图标，平台相关的文件等
|   ├── renderer - 运行在渲染进程中的代码
|   |   ├── lib - 渲染进程初始化代码中 JavaScript 部分的代码
|   |   └── api - 渲染进程 API 的实现
|   |       └── lib - API 实现中 Javascript 部分的代码
|   └── common - 同时被主进程和渲染进程用到的代码，包括了一些用来将 node 的事件循环
|       |          整合到 Chromium 的事件循环中时用到的工具函数和代码
|       ├── lib - 同时被主进程和渲染进程使用到的 Javascript 初始化代码
|       └── api - 同时被主进程和渲染进程使用到的 API 的实现以及 Electron 内置模块的
基础设施
|           └── lib - API 实现中 Javascript 部分的代码
├── chromium_src - 从 Chromium 项目中拷贝来的代码
├── docs - 英语版本的文档
├── docs-translations - 各种语言版本的文档翻译
├── spec - 自动化测试
├── atom.gyp - Electron 的构建规则
└── common.gypi - 为诸如 `node` 和 `breakpad` 等其他组件准备的编译设置和构建规则
```

平时开发时，需要重点关注的就是 src、package.json 和 appveyor.yml 目录。除此之外，其他需要注意的目录如下：

- script - 用于诸如构建、打包、测试等开发用途的脚本
- tools - 在 gyp 文件中用到的工具脚本，但与 script 目录不同， 该目录中的脚本不应该被用户直接调用
- vendor - 第三方依赖项的源代码，为了防止人们将它与 Chromium 源码中的同名目录相混淆， 在这里我们不使用 third_party 作为目录名
- node_modules - 在构建中用到的第三方 node 模块
- out - ninja 的临时输出目录
- dist - 由脚本 script/create-dist.py 创建的临时发布目录
- external_binaries - 下载的不支持通过 gyp 构建的预编译第三方框架

应用工程目录

使用 electron-vue 模版创建的 Electron 工程结构如下图。

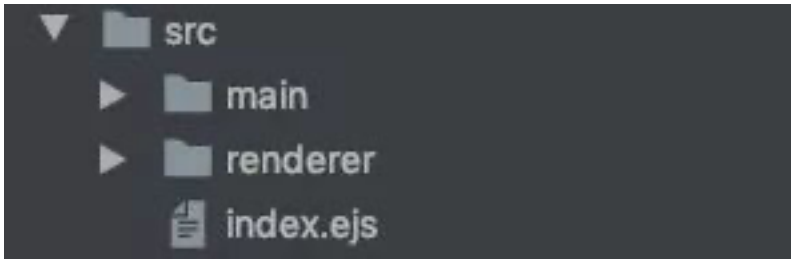


和前端工程的项目结构类似，Electron 项目的目录结构如下所示：

- electron-vue：Electron 模版配置。

- build：文件夹，用来存放项目构建脚本。
- config：中存放项目的一些基本配置信息，最常用的就是端口转发。
- node_modules：这个目录存放的是项目的所有依赖，即 npm install 命令下载下来的文件。
- src：这个目录下存放项目的源码，即开发者写的代码放在这里。
- static：用来存放静态资源。
- index.html：则是项目的首页、入口页，也是整个项目唯一的 HTML 页面。
- package.json：中定义了项目的所有依赖，包括开发时依赖和发布时依赖。

对于开发者来说， 90% 的工作都是在 src 中完成，src 中的文件目录如下。



Electron 应用程序分成三个基础模块：主进程、进程间通信和渲染进程。

1、主进程

Electron 运行 package.json 的 main 脚本（background.js）的进程被称为主进程。在主进程中运行的脚本通过创建 web 页面来展示用户界面。一个 Electron 应用总是有且只有一个主进程。

2、渲染进程

由于 Electron 使用了 Chromium 来展示 Web 页面，所以 Chromium 的多进程架构也被使用到。每个 Electron 中的 Web 页面运行在它自己的渲染进程中。在普通的浏览器中，Web 页面通常在一个沙盒环境中运行，不被允许去接触原生的资源。然而 Electron 允许用户在 Node.js 的 API 支持下可以在页面中和操作系统进行一些底层交互。

3、主进程与渲染进程通信

主进程使用 BrowserWindow 实例创建页面。每个 BrowserWindow 实例都在自己的渲染进程里运行页面。当一个 BrowserWindow 实例被销毁后，相应的渲染进程也会被终止。主进程管理所有的 Web 页面和它们对应的渲染进程。每个渲染进程都是独立的，它只关心它所运行的 Web 页面。

src 目录结构

在 Electron 目录中，src 会包包含 main 和 renderer 两个目录。

main 目录

main 目录会包含 index.js 和 index.dev.js 两个文件。

- index.js：应用程序的主文件，electron 也从这里启动的，它也被用作 webpack 产品构建的入口文件，所有的 main 进程工作都应该从这里开始。
- index.dev.js：此文件专门用于开发阶段，因为它会安装 electron-debug 和 vue-devtools。一般不需要修改此文件，但它可以扩展开发的需求。

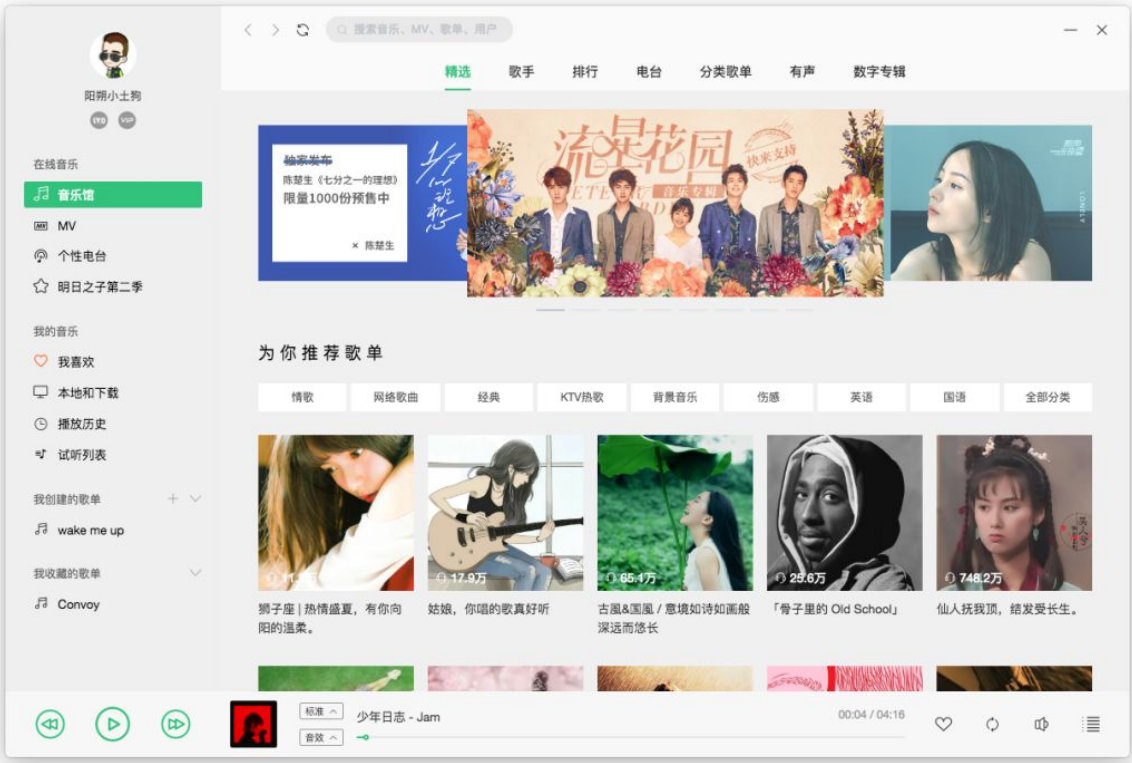
渲染进程

renderer 是渲染进程目录，平时项目开发源码的存放目录，包含 assets、components、router、store、App.vue 和 main.js。

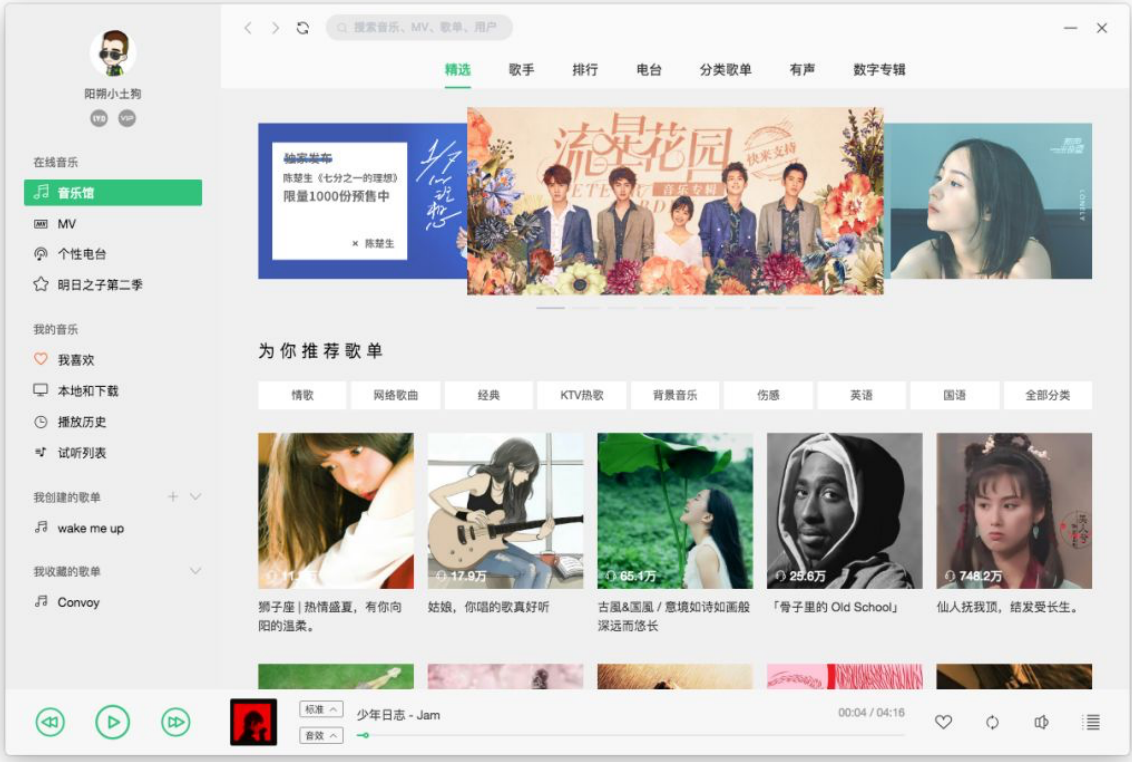
assets：assets 下的文件如（js、css）都会在 dist 文件夹下面的项目目录分别合并到一个文件里面去。components：此文件用于存放应用开发的组件，可以是自定义的组件。router：如果你了解 vue-router，那么 Electron 项目的路由的使用方式和 vue-router 的使用方式类似。modules：electron-vue 利用 vuex 的模块结构创建多个数据存储，并保存在 src/renderer/store/modules 中。

相关案例

- https://github.com/xiaozhu188/electron-vue-cloud-music



- https://github.com/SmallRuralDog/electron-vue-music


阳明小土狗
在线音乐
音乐馆
MV
个性电台
明日之子第二季
我的音乐
我喜欢
本地和下载
播放历史
试听列表
我创建的歌单
wake me up
我收藏的歌单
Convoy
搜索音乐、MV、歌单、用户
精选
歌手
排行
电台
分类歌单
有声
数字专辑
独家发布
陈楚生《七分之一的理想》
限量1000份预售中
× 陈楚生
流星花园
为你推荐歌单
情歌
网络歌曲
经典
KTV热歌
背景音乐
伤感
英语
国语
全部分类
17.9万
65.1万
29.6万
748.2万
狮子座 | 热情盛夏，有你向阳的温柔。
姑娘，你唱的歌真好听
古風&国風 / 意境如诗如画般深远而悠长
「骨子里的 Old School」
仙人抚我顶，结发受长生。
标准
音效
少年日志 - Jam
00:04 / 04:16