
黑马程序员™
www.itheima.com
传智播客旗下
高端IT教育品牌
Vue全家桶-项目优化上线
GO

目录 Contents

1. 项目优化

1.1 项目优化策略

1. 生成打包报告
2. 第三方库启用 CDN
3. Element-UI 组件按需加载
4. 路由懒加载
5. 首页内容定制

1. 项目优化

1.1 项目优化策略

1. 生成打包报告

打包时，为了直观地发现项目中存在的问题，可以在打包时生成报告。生成报告的方式有两种：

① 通过命令行参数的形式生成报告

```
// 通过 vue-cli 的命令选项可以生成打包报告
// --report 选项可以生成 report.html 以帮助分析包内容
vue-cli-service build --report
```

② 通过可视化的UI面板直接查看报告（推荐）

在可视化的UI面板中，通过控制台和分析面板，可以方便地看到项目中所存在的问题。

1. 项目优化

1.1 项目优化策略

2. 通过 vue.config.js 修改 webpack 的默认配置

通过 vue-cli 3.0 工具生成的项目，默认隐藏了所有 webpack 的配置项，目的是为了屏蔽项目的配置过程，让程序员把工作的重心，放到具体功能和业务逻辑的实现上。

如果程序员有修改 webpack 默认配置的需求，可以在项目根目录中，按需创建 vue.config.js 这个配置文件，从而对项目的打包发布过程做自定义的配置（具体配置参考 https://cli.vuejs.org/zh/config/#vue-config-js ）。

```
// vue.config.js
// 这个文件中，应该导出一个包含了自定义配置选项的对象
module.exports = {
  // 选项...
}
```

1. 项目优化

1.1 项目优化策略

3. 为开发模式与发布模式指定不同的打包入口

默认情况下，Vue项目的开发模式与发布模式，共用同一个打包的入口文件（即 src/main.js ）。为了将项目的开发过程与发布过程分离，我们可以为两种模式，各自指定打包的入口文件，即：

① 开发模式的入口文件为 src/main-dev.js

② 发布模式的入口文件为 src/main-prod.js

1. 项目优化

1.1 项目优化策略

4. configureWebpack 和 chainWebpack

在 vue.config.js 导出的配置对象中，新增 configureWebpack 或 chainWebpack 节点，来自定义 webpack 的打包配置。

在这里， configureWebpack 和 chainWebpack 的作用相同，唯一的区别就是它们修改 webpack 配置的方式不同：

① chainWebpack 通过链式编程的形式，来修改默认的 webpack 配置

② configureWebpack 通过操作对象的形式，来修改默认的 webpack 配置

两者具体的使用差异，可参考如下网址：

https://cli.vuejs.org/zh/guide/webpack.html#webpack-%E7%9B%B8%E5%85%B3

1. 项目优化

1.1 项目优化策略

5. 通过 chainWebpack 自定义打包入口

代码示例如下：

```
module.exports = {
  chainWebpack: config => {
    config.when(process.env.NODE_ENV === 'production', config => {
      config.entry('app').clear().add('./src/main-prod.js')
    })
    config.when(process.env.NODE_ENV === 'development', config => {
      config.entry('app').clear().add('./src/main-dev.js')
    })
  }
}
```

1. 项目优化

1.1 项目优化策略

6. 通过 externals 加载外部 CDN 资源

默认情况下，通过 import 语法导入的第三方依赖包，最终会被打包合并到同一个文件中，从而导致打包成功后，单文件体积过大的问题。

为了解决上述问题，可以通过 webpack 的 externals 节点，来配置并加载外部的 CDN 资源。凡是声明在 externals 中的第三方依赖包，都不会被打包。

1. 项目优化

1.1 项目优化策略

6. 通过 externals 加载外部 CDN 资源

具体配置代码如下：

```
config.set('externals', {
  vue: 'Vue',
  'vue-router': 'VueRouter',
  axios: 'axios',
  lodash: '_',
  echarts: 'echarts',
  nprogress: 'NProgress',
  'vue-quill-editor': 'VueQuillEditor'
})
```

1. 项目优化

1.1 项目优化策略

6. 通过 externals 加载外部 CDN 资源

同时，需要在 public/index.html 文件的头部，添加如下的 CDN 资源引用：

```
<!-- nprogress 的样式表文件 -->
<link rel="stylesheet" href="https://cdn.staticfile.org/nprogress/0.2.0/nprogress.min.css" />
<!-- 富文本编辑器 的样式表文件 -->
<link rel="stylesheet" href="https://cdn.staticfile.org/quill/1.3.4/quill.core.min.css" />
<link rel="stylesheet" href="https://cdn.staticfile.org/quill/1.3.4/quill.snow.min.css" />
<link rel="stylesheet" href="https://cdn.staticfile.org/quill/1.3.4/quill.bubble.min.css" />
```

1. 项目优化

1.1 项目优化策略

6. 通过 externals 加载外部 CDN 资源

同时，需要在 public/index.html 文件的头部，添加如下的 CDN 资源引用：

```
<script src="https://cdn.staticfile.org/vue/2.5.22/vue.min.js"></script>
<script src="https://cdn.staticfile.org/vue-router/3.0.1/vue-router.min.js"></script>
<script src="https://cdn.staticfile.org/axios/0.18.0/axios.min.js"></script>
<script src="https://cdn.staticfile.org/lodash.js/4.17.11/lodash.min.js"></script>
<script src="https://cdn.staticfile.org/echarts/4.1.0/echarts.min.js"></script>
<script src="https://cdn.staticfile.org/nprogress/0.2.0/nprogress.min.js"></script>
<!-- 富文本编辑器的 js 文件 -->
<script src="https://cdn.staticfile.org/quill/1.3.4/quill.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/vue-quill-editor@3.0.4/dist/vue-quill-editor.js"></script>
```

1. 项目优化

1.1 项目优化策略

7. 通过 CDN 优化 ElementUI 的打包

虽然在开发阶段，我们启用了 element-ui 组件的按需加载，尽可能的减少了打包的体积，但是那些被按需加载的组件，还是占用了较大的文件体积。此时，我们可以将 element-ui 中的组件，也通过 CDN 的形式来加载，这样能够进一步减小打包后的文件体积。

具体操作流程如下：

① 在 main-prod.js 中，注释掉 element-ui 按需加载的代码

② 在 index.html 的头部区域中，通过 CDN 加载 element-ui 的 js 和 css 样式

```
<!-- element-ui 的样式表文件 -->
<link rel="stylesheet" href="https://cdn.staticfile.org/element-ui/2.8.2/theme-chalk/index.css" />
<!-- element-ui 的 js 文件 -->
<script src="https://cdn.staticfile.org/element-ui/2.8.2/index.js"></script>
```

1. 项目优化

1.1 项目优化策略

8. 首页内容定制

不同的打包环境下，首页内容可能会有所不同。我们可以通过插件的方式进行定制，插件配置如下：

```
chainWebpack: config => {
  config.when(process.env.NODE_ENV === 'production', config => {
    config.plugin('html').tap(args => {
      args[0].isProd = true
      return args
    })
  })
  config.when(process.env.NODE_ENV === 'development', config => {
    config.plugin('html').tap(args => {
      args[0].isProd = false
      return args
    })
  })
}
```

1. 项目优化

1.1 项目优化策略

8. 首页内容定制

在 public/index.html 首页中，可以根据 isProd 的值，来决定如何渲染页面结构：

```
<!- 按需渲染页面的标题 -->
<title><%= htmlWebpackPlugin.options.isProd ? '' : 'dev - ' %>电商后台管理系统</title>

<!- 按需加载外部的 CDN 资源 -->
<% if(htmlWebpackPlugin.options.isProd) { %>
<!—通过 externals 加载的外部 CDN 资源-->
<% } %>
```

1. 项目优化

1.1 项目优化策略

9. 路由懒加载

当打包构建项目时，JavaScript 包会变得非常大，影响页面加载。如果我们能把不同路由对应的组件分割成不同的代码块，然后当路由被访问的时候才加载对应组件，这样就更加高效了。

具体需要 3 步：

① 安装 @babel/plugin-syntax-dynamic-import 包。

② 在 babel.config.js 配置文件中声明该插件。

③ 将路由改为按需加载的形式，示例代码如下：

```
const Foo = () => import(/* webpackChunkName: "group-foo" */ './Foo.vue')
const Bar = () => import(/* webpackChunkName: "group-foo" */ './Bar.vue')
const Baz = () => import(/* webpackChunkName: "group-boo" */ './Baz.vue')
```

关于路由懒加载的详细文档，可参考如下链接：

https://router.vuejs.org/zh/guide/advanced/lazy-loading.html

目录 Contents

2. 项目上线

2.1 项目上线相关配置

1. 通过 node 创建 web 服务器。
2. 开启 gzip 配置。
3. 配置 https 服务。
4. 使用 pm2 管理应用。

2. 项目上线

2.1 项目上线相关配置

1. 通过 node 创建 web 服务器

创建 node 项目，并安装 express，通过 express 快速创建 web 服务器，将 vue 打包生成的 dist 文件夹，托管为静态资源即可，关键代码如下：

```
const express = require('express')
// 创建 web 服务器
const app = express()

// 托管静态资源
app.use(express.static('./dist'))

// 启动 web 服务器
app.listen(80, () => {
  console.log('web server running at http://127.0.0.1')
})
```

2. 项目上线

2.1 项目上线相关配置

2. 开启 gzip 配置

使用 gzip 可以减小文件体积，使传输速度更快。

② 可以通过服务器端使用 Express 做 gzip 压缩。其配置如下：

```
// 安装相应包
npm install compression -S
// 导入包
const compression = require('compression');
// 启用中间件
app.use(compression());
```

2.1 项目上线相关配置

3. 配置 HTTPS 服务

为什么要启用 HTTPS 服务？

- 传统的 HTTP 协议传输的数据都是明文，不安全
- 采用 HTTPS 协议对传输的数据进行了加密处理，可以防止数据被中间人窃取，使用更安全

2.1 项目上线相关配置

3. 配置 HTTPS 服务

申请 SSL 证书（https://freessl.org）

① 进入 https://freessl.cn/ 官网，输入要申请的域名并选择品牌。
② 输入自己的邮箱并选择相关选项。
③ 验证 DNS（在域名管理后台添加 TXT 记录）。
④ 验证通过之后，下载 SSL 证书（ full_chain.pem 公钥；private.key 私钥）。

2. 项目上线

2.1 项目上线相关配置

3. 配置 HTTPS 服务

在后台项目中导入证书

```
const https = require('https');
const fs = require('fs');
const options = {
    cert: fs.readFileSync('./full_chain.pem'),
    key: fs.readFileSync('./private.key')
}
https.createServer(options, app).listen(443);
```

2. 项目上线

2.1 项目上线相关配置

4. 使用 pm2 管理应用

① 在服务器中安装 pm2：npm i pm2 -g
② 启动项目：pm2 start 脚本 --name 自定义名称
③ 查看运行项目：pm2 ls
④ 重启项目：pm2 restart 自定义名称
⑤ 停止项目：pm2 stop 自定义名称
⑥ 删除项目：pm2 delete 自定义名称

传智播客旗下高端IT教育品牌