13 Webpack 中打包 HTML 和多页面配置

更新时间：2019-06-24 09:26:46

> 辛苦是获得一切的定律。
>
> ——牛顿

在项目中我们除了需要 JavaScript、CSS 和图片等静态资源，还需要页面来承载这些内容和页面结构，怎么在 Webpack 中处理 HTML。并且我们项目也不仅仅是单页应用（Single-Page Application，SPA），也可能是多页应用，所以我们还需要使用 Webpack 来给多页应用做打包。本小节将讲解这俩问题。

使用 HTML 插件来做页面展现

有了 JavaScript 文件，还缺 HTML 页面，要让 Webpack 处理 HTML 页面需要只需要使用 html-webpack-plugin 插件即可，首先安装它：

```
npm i html-webpack-plugin --save-dev
```

然后我们修改对应的 webpack.config.js 内容：

```
const HtmlWebPackPlugin = require('html-webpack-plugin');

module.exports = {
   mode: 'development',
   entry: {
      main: './src/index.js'
   },
   plugins: [new HtmlWebPackPlugin()]
};
```

只需要简单配置，执行 webpack 打包之后，发现 log 中显示，在 dist 文件夹中生成一个 index.html 的文件：

```
Hash: b1e026363f3dd3c57a8f
Version: webpack 4.29.6
Time: 334ms
Built at: 2019-04-06 12:30:18
     Asset       Size  Chunks             Chunk Names
index.html  225 bytes          [emitted]
   main.js    3.8 KiB    main  [emitted]  main
Entrypoint main = main.js
[./src/index.js] 29 bytes {main} [built]
```

打开后发现 HTML 的内容如下：

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Webpack App</title>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
  </head>

  <body>
    <script src="main.js"></script>
  </body>
</html>
```

除了我们 HTML 外，我们的 entry 也被主动插入到了页面中，这样我们打开 index.html 就直接加载了 main.js 了。

如果要修改 HTML 的 title 和名称，可以如下配置：

```
const HtmlWebPackPlugin = require('html-webpack-plugin');

module.exports = {
  mode: 'development',
  entry: {
    main: './src/index.js'
  },
  plugins: [new HtmlWebPackPlugin({title: 'hello', filename: 'foo.html'})]
};
```

Template

虽然我们可以简单的修改 Title 这里自定义内容，但是对于我们日常项目来说，这远远不够。我们希望 HTML 页面需要根据我们的意愿来生成，也就是说内容是我们来定的，甚至根据打包的 entry 最后结果来定，这时候我们就需要使用 html-webpack-plugin 的 template 功能了。

比如我在 index.js 中，给 id="app" 的节点添加内容，这时候 HTML 的内容就需要我们自定义了，至少应该包含一个含有 id="app" 的 DIV 元素：

```
<div id="app"></div>
```

我们可以创建一个自己想要的 HTML 文件，比如 index.html，在里面写上我们想要的内容：

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Webpack</title>
  </head>
  <body>
    <h1>hello world</h1>
    <div id="app"></div>
  </body>
</html>
```

把 webpack.config.js 更改如下：

```
const HtmlWebPackPlugin = require('html-webpack-plugin');

module.exports = {
  mode: 'development',
  entry: {
    main: './src/index.js'
  },
  plugins: [
    new HtmlWebPackPlugin({
      template: './src/index.html'
    })
  ]
};
```

这时候，打包之后的 HTML 内容就变成了：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Webpack</title>
</head>
<body>
  <h1>hello world</h1>
  <div id="app"></div>
<script src="main.js"></script></body>
</html>
```

也是添加上了 main.js 内容。

使用 JavaScript 模板引擎

HTML 毕竟还是有限，这时候还可以使用 JavaScript 模板引擎来创建 html-webpack-plugin 的 Template 文件。下面我以pug模板引擎为例，来说明下怎么使用模板引擎文件的 Template。

首先创建个 index.pug 文件，内容如下：

```
doctype html
html(lang="en")
 head
  title="Hello Pug"
  script(type='text/javascript').
    console.log('pug inline js')
 body
  h1 Pug - node template engine
  #app
  include includes/footer.pug
```

如果不理解 Pug 模板的语法，可以简单看下文档，我这里简单解释下，首先在头部加了 title 和一个 script 标签，然后在 body 中内容为 h1、 id="app" 的 div 和引入（include）了一个 footer.pug 的文件：

```
footer#footer
  p Copyright @Copyright 2019
```

这时候我们需要修改 webpack.config.js 内容：

```
const HtmlWebPackPlugin = require('html-webpack-plugin');

module.exports = {
   mode: 'development',
   entry: {
      main: './src/index.js'
   },
   plugins: [
      new HtmlWebPackPlugin({
         template: './src/index.pug'
      })
   ]
};
```

但是只修改 template='src/index.pug' 是不够的，因为 .pug 这样的文件 Webpack 是不会解析的，所以我们需要加上 Pug 的 loader：pug-html-loader，除了这个插件还需要安装html-loader。首先通过 npm i -D pug-html-loader html-loader 安装它们，然后修改 webpack.config.js 内容，添加 rule：

```
const HtmlWebPackPlugin = require('html-webpack-plugin');

module.exports = {
   mode: 'development',
   entry: {
      main: './src/index.js'
   },
   plugins: [
      new HtmlWebPackPlugin({
         template: './src/index.pug'
      })
   ],
   module: {
      rules: [{test: /\.pug$/, loader: ['html-loader', 'pug-html-loader']}]
   }
};
```

最后，我们得到的 index.html 内容如下：

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Hello Pug</title>
      <script type="text/javascript">
         console.log('pug inline js');
      </script>
   </head>
   <body>
      <h1>Pug - node template engine</h1>
      <div id="app"></div>
      <footer id="footer"><p>Copyright @Copyright 2019</p></footer>
      <script src="main.js"></script>
   </body>
</html>
```

Pug 引擎被转换成 HTML代码，里面包含了： main.js 和 footer.pug 的内容。

关于html-webpack-plugin的参数这里就不展开了，可以查阅它的 README 文档。

> Tips：使用 JavaScript 模板引擎，还可以定义一些变量，通过 html-webpack-plugin 传入进去。

多页项目配置

要做一个多页项目的配置，那么需要考虑以下几个问题：

1. 多页应用，顾名思义最后我们打包生成的页面也是多个，即 HTML 是多个；
2. 多页应用不仅仅是页面多个，入口文件也是多个；
3. 多页应用可能页面之间页面结构是不同的，比如一个网站项目，典型的三个页面是：首页、列表页和详情页，肯定每个页面都不一样。

下面我们来一个一个的问题解决：

多页面问题

多页面就是指的多个 HTML 页面，这时候可以直接借助 html-webpack-plugin 插件来实现，我们只需要多次实例化一个 html-webpack-plugin 的实例即可，例如：

下面是同一个 template，那么可以只修改 filename 输出不同名的 HTML 即可：

```
const HtmlWebPackPlugin = require('html-webpack-plugin');

const indexPage = new HtmlWebPackPlugin({
    template: './src/index.html',
    filename: 'index.html'
});
const listPage = new HtmlWebPackPlugin({
    template: './src/index.html',
    filename: 'list.html'
});
module.exports = {
    mode: 'development',
    entry: {
        main: './src/index.js'
    },
    plugins: [indexPage, listPage]
};
```

对于页面结构不同的 HTML 页面的配置，使用不同的 template 即可。

```
const HtmlWebPackPlugin = require('html-webpack-plugin');

const indexPage = new HtmlWebPackPlugin({
    template: './src/index.html',
    filename: 'index.html'
});
const listPage = new HtmlWebPackPlugin({
    template: './src/list.html',
    filename: 'list.html'
});
module.exports = {
    mode: 'development',
    entry: {
        main: './src/index.js'
    },
    plugins: [indexPage, listPage]
};
```

多入口问题

上面的多页面解决是多次实例化 html-webpack-plugin，根据传入的参数不同（主要是 filename 不同），打包出两个文件，但是这两个文件的特点是引入的 JavaScript 文件都是一样的，即都是 main.js 。

对于多入口，并且入口需要区分的情况，那么需要怎么处理呢？

这时候就需要借助 html-webpack-plugin 的两个参数了： chunks 和 excludeChunks 。 chunks 是当前页面包含的 chunk 有哪些，可以直接用 entry 的 key 来命名， excludeChunks 则是排除某些 chunks。

例如，现在有两个 entry，分别是 index.js 和 list.js ，我们希望 index.html 跟 index.js 是一组， list.html 跟 list.js 是一组，那么 webpack.config.js 需要修改为：

```
const HtmlWebPackPlugin = require('html-webpack-plugin');

module.exports = {
   mode: 'development',
   entry: {
      index: './src/index.js',
      list: './src/list.js'
   },
   plugins: [
      new HtmlWebPackPlugin({template: './src/index.html', filename: 'index.html', chunks: ['index']}),
      new HtmlWebPackPlugin({template: './src/list.html', filename: 'list.html', chunks: ['list']})
   ]
};
```

最佳实践

现在说下我在项目中的一般做法，个人认为这是多页应用的最佳实践。

首先，需要规定下目录结构规范，一般我们项目会有下面的目录规范：

```
├── package.json
├── webpack.config.json
├── src
│   ├── libs
│   └── pages
│       ├── detail.js
│       ├── index.js
│       └── list.js
├── template
│   ├── detail.html
│   ├── index.html
│   └── list.html
```

保证 template 和实际的 entry 是固定的目录，并且名字都是对应的。

这时候我们可以写个 Node.js 代码遍历对应的路径，然后生成 webpack.config.js 的 entry 和 html-webpack-plugin 内容。

这里我使用了globby这个 NPM 模块，先写了读取 src/pages/*.js 的内容，然后生成 entry ：

```
const path = require('path');
const globby = require('globby');

const getEntry = (exports.getEntry = () => {
  // 异步方式获取所有的路径
  const paths = globby.sync('./pages/*.js', {
    cwd: path.join(__dirname, './src')
  });
  const rs = {};
  paths.forEach(v => {
    // 计算 filename
    const name = path.basename(v, '.js');
    let p = path.join('./src', v);
    if (!p.startsWith('.')) {
      // 转成相对地址
      p = './' + p;
    }

    rs[name] = p;
  });
  return rs;
});

// 输出内容
console.log(getEntry());
```

下一步就是遍历 entry 对象，然后生成 html-webpack-plugins 的数组了：

```
const HtmlWebPackPlugin = require('html-webpack-plugin');

exports.getHtmlWebpackPlugins = () => {
  const entries = getEntry();
  return Object.keys(entries).reduce((plugins, filename) => {
    plugins.push(
      new HtmlWebPackPlugin({
        template: entries[filename],
        filename: `${filename}.html`,
        chunks: [filename]
      })
    );
    return plugins;
  }, []);
};
```

我们在 webpack.config.js 用的时候，直接 require 引入刚刚写的这个文件，然后：

```
const {getEntry, getHtmlWebpackPlugins} = require('./scripts/utils');

module.exports = {
  mode: 'development',
  getEntry(),
  plugins: [
    //...
    ...getHtmlWebpackPlugins()
  ]
};
```

小结

我们写的代码最终还是需要页面来承载展现，本小节主要介绍 Webpack 的 html-webpack-plugin 插件的使用方法。通过 html-webpack-plugin 我们可以生成包含 Webpack 打包后资源的 HTML 页面。针对 Webpack 中多页应用的打包，我们可以配置多个 html-webpack-plugin 插件实例。

我们还可以按照文章介绍的多页应用最佳实践的方案，通过约定目录规范来通过 Node.js 代码来自动生成 Webpack 的多页应用配置。html-webpack-plugin 是 Webpack 中很重要的一个插件，基于这个插件的 API 我们可以做很多跟页面相关的优化项目，比如预取资源、实现 modern 打包等，后面的实战章节会继续介绍。

> 本小节 Webpack 相关面试题：
>
> 1. 怎么配置 Webpack 的多页面开发？
> 2. 你们项目中 Webpack 的多页面开发有什么最佳实践吗？

12 使用 Webpack 管理项目中的静态资源

14 Webpack Dev Server 本地开发服务

精选留言 2

欢迎在这里发表留言，作者筛选后可公开显示

BskyRui

多页应用打包给后端node或者Python用的时候, 页面公共部分怎么处理?

0 回复 2019-06-14

作者 **三水清** 回复 **BskyRui**

node 可以直接用模板引擎啊，比如 ejs 的 include 这类语法，python 应该也有类似的模板引擎吧，没做过 Python 项目。。。

回复 2019-07-03 15:37:58

躁动的胸大肌

学到了很多，谢谢老师

2 回复 2019-06-13