16 Webpack 环境相关配置及配置文件拆分

更新时间：2019-06-24 09:30:10

> 最聪明的人是最不愿浪费时间的人。
>
> ——但丁

在实际开发实践中，为了方便开发调试和上线，项目中我们一般配置两个 Webpack 配置，一个是**开发环境**一个是**生产环境**，开发环境帮助我们快速开发测试联调，生产环境保证上线环境的打包流程是最优化体验，这就是配置文件根据环境进行拆分。

配置文件拆分最重要的收益是我们可以提取不同目标之间的共性，将经常改动的配置跟公共配置分开。并且还可以识别要组合的较小配置部件，这些配置不仅可以推送到自己的软件包以跨项目使用。另外，配置拆分还可以将配置作为依赖项进行管理，而不是在多个项目中复制类似的配置。

开发环境和生产环境需要注意的区别

- 生产环境可能需要分离 CSS 成单独的文件，以便多个页面共享同一个 CSS 文件；
- 生产环境需要压缩 HTML/CSS/JS 代码；
- 生产环境需要压缩图片；
- 开发环境需要生成 SourceMap 文件；
- 开发环境需要打印 debug 信息；
- 开发环境需要 HMR、devServer 等功能…

按环境划分 Webpack 配置文件

- `webpack.config.js`：所有环境的默认入口配置文件；
- `webpack.base.js`：基础部分，即多个文件中共享的配置；
- `webpack.development.js`：开发环境使用的配置；
- `webpack.production.js`：生产环境使用的配置。

```
// webpack.config.js
const baseWebpackConfig = require('./webpack.base.js');
const devWebpackConfig = require('./webpack.development.js');
const merge = require('webpack-merge');
module.exports = merge(baseWebpackConfig, devWebpackConfig);
```

webpack development config

`webpack.development.js` 文件里面的配置多数跟开发环境相关的配置，例如 `devServer` 和一些开发环境中需要的 plugin，比如 hmr、devServer、devtool 等。

webpack production config

这里的配置主要包括了跟生产环境相关的配置项，例如 `optimization`、`devtool` 等。

webpack base config

`webpack.base.js` 是公共部分的配置，除了完全一致的内容之外，还可能会涉及到一些根据不同环境做的事情，比如 `style-loader` 在开发环境用，而开发环境是用 `mini-css-extract-plugin` 的 loader，这时候需要使用环境变量来判断。

有两种方式来判断当前的环境变量：

1. 使用环境变量，例如 cross-env + `NODE_ENV`；
2. 使用 Webpack 配置文件的 `function` 方式。

> Tips：这是因为我们所有的入口文件都是 `webpack.config.js`。当然如果我们使用 Webpack 的时候，就已经在 `npm scripts` 里面区分了配置文件（`webpack --config webpack.production.js`），就不在讨论范围之内了。这里只是顺着之前的逻辑继续分析下去。

cross-env

首先安装 cross-env：

```
npm i -D cross-env
```

然后修改 `npm scripts` 内容：

```
// package.json
{
    "scripts": {
        "build": "cross-env NODE_ENV=production webpack --config webpack.config.js"
    }
}
```

最后在 `webpack.base.js` 中使用环境变量：

```
// webpack.base.js
const isProduction = process.env.NODE_ENV === 'production';
//....
module.exports = {
    // ...
    devtool: isProduction ? null : 'source-map'
};
```

`function` 配置

根据前面的章节我们知道，Webpack 的配置可以是对象，也可以是函数，如果是 function 则接受一个`mode`参数，即开发环境打包还是生产环境打包。利用这一点我们可以做下面的配置。

首先是修改`npm scripts`添加`--mode`选项：

```
// package.json
{
    "scripts": {
        "build": "webpack --mode production --config webpack.config.js"
    }
}
```

然后我们可以将配置文件改成 function 类型的配置：

```
// 以webpack.config.js为例
module.exports = mode => {
    if (mode === 'production') {
        // 生产环境
    } else {
        // 开发环境
    }
};
```

将零件配置进行拆分

除了按照开发环境拆分出`development`和`production`之外，我们还可以将公共的配置按照 loader、devServer 等相关配置拆到`webpack.parts.js`文件中，在其他项目直接组装想用的内容即可。

例如下面的配置：

```
// webpack.parts.js
const MiniCssExtractPlugin = require('mini-css-extract-plugin');

// 获取 css-loader 配置
exports.getCssLoader = ({mode, test = /\.css$/, include, exclude, uses = []} = {}) => ({
    test,
    include,
    exclude,
    use: [
        {
            loader: mode === 'production' ? MiniCssExtractPlugin.loader : 'style-loader'
        },
        {
            loader: 'css-loader'
        }
    ].concat(uses)
});
// 获取 devServer 配置
exports.getDevServerConfig = ({host = '0.0.0.0', port = 8888} = {}) => ({
    stats: 'errors-only',
    host,
    port,
    open: true,
    overlay: true
});
// 获取 url-loader 配置
exports.getUrlLoader = ({test, largeAssetSize = 1000, assetsDir, dir} = {}) => ({
    test,
    use: {
        loader: 'url-loader',
        options: {
            limit: largeAssetSize,
            name: getAssetPath(assetsDir, `${dir}/[name]${isProduction ? '.[hash:8]' : ''}.[ext]`)
        }
    }
});
// .....
```

使用的时候，直接调用对应的方法即可：

```
const partsConfig = require('./webpack.parts.js');

module.exports = {
    mode: 'production',
    devtool: 'source-map',
    //...
    devServer: partsConfig.getDevServer(),
    modules: {
        rules: [
            partsConfig.getCssLoader(),
            partsConfig.getUrlLoader({test: /\.(png|jpe?g|gif|webp|svg)(\?.*)?$/, dir: 'img'})
            // ....
        ]
    }
};
```

小结

在一个 Webpack 项目中，根据不同的使用环境合理划分 Webpack 的配置文件是很有必要的。本文介绍了如何按照开发环境和生产环境划分 Webpack 配置文件，并且介绍了公共配置部分拆分规则，可以使用 cross-env 模块来传入不同的环境变量，这样在公共配置文件内可以根据不同的环境变量进行配置。我们甚至可以将 Webpack 的配置拆成一个个的零件进行自由的搭配。

> 本小节 Webpack 相关面试题：

1. 你们项目是如何管理 Webpack 的配置文件的？

15 Webpack 中配置React和Vue 开发环境

17 Webpack 优化之体积优化