# webpack4——SplitChunksPlugin使用指南

2018年03月06日 15:04:42

阅读数：7106

## **写在前面**

      前面写了[一篇](http://blog.csdn.net/qq_26733915/article/details/79446460" \t "https://blog.csdn.net/qq_26733915/article/details/_blank)有关webpack4的不完全升级指南以及在webpack3.x迁移的时候遇到的问题，有兴许可以看一下。

### **0. 参数介绍**

先对参数有一个大概的认识，虽然撸了很多遍官方的更新文档，但是还是去参看了一下新的wbepack源码，下面是各种参数及含义：

* chunks: 表示显示块的范围，有三个可选值：initial(初始块)、async(按需加载块)、all(全部块)，默认为all;
* minSize: 表示在压缩前的最小模块大小，默认为0；
* minChunks: 表示被引用次数，默认为1；
* maxAsyncRequests: 最大的按需(异步)加载次数，默认为1；
* maxInitialRequests: 最大的初始化加载次数，默认为1；
* name: 拆分出来块的名字(Chunk Names)，默认由块名和hash值自动生成；
* cacheGroups: 缓存组。

对于缓存组是一个对象，处了可以有上面的chunks、minSize、minChunks、maxAsyncRequests、maxInitialRequests、name外，还有其他的一些参数：

如果不在缓存组中重新赋值，缓存组默认继承上面的选项，但是还有一些参数是必须在缓存组进行配置的。

* priority: 表示缓存的优先级；
* test: 缓存组的规则，表示符合条件的的放入当前缓存组，值可以是function、boolean、string、RegExp，默认为空；
* reuseExistingChunk: 表示可以使用已经存在的块，即如果满足条件的块已经存在就使用已有的，不再创建一个新的块。

### **1. 基本使用**

首先，在新版本的webpack会默认对代码进行拆分，拆分的规则是：

* 模块被重复引用或者来自node\_modules中的模块
* 在压缩前最小为30kb
* 在按需加载时，请求数量小于等于5
* 在初始化加载时，请求数量小于等于3

小于30kb的模块不值得再单独发送一次请求，在很小的模块的前提下，相比与多次打包，减少请求次数成本要低。

当然也可以不使用默认的配置，比如这样：

new webpack.optimize.SplitChunksPlugin({

chunks: "all",

minSize: 20000,

minChunks: 1,

maxAsyncRequests: 5,

maxInitialRequests: 3,

name: true

)}

上面的代码就表示，在所有代码中，引用模块大小最小为20kb，引用次数最少为1次，按需加载最大请求次数为5，初始化加载最大请求次数为3的所有模块就行拆分到一个单独的代码块中，name表示代码的名字，设置为true则表示根据模块和缓存组秘钥自动生成。

### **2. 使用缓存组(Cache Groups)**

如果想继续细分代码，可以使用缓存组(Cache Groups)。同样的，缓存组也有默认的配置；缓存组默认将node\_modules中的模块拆分带一个叫做vendors的代码块中，将最少重复引用两次的模块放入default中。

这是一段[官方](https://gist.github.com/sokra/1522d586b8e5c0f5072d7565c2bee693" \t "https://blog.csdn.net/qq_26733915/article/details/_blank)里面的代码：

splitChunks: {

chunks: "async",

minSize: 30000,

minChunks: 1,

maxAsyncRequests: 5,

maxInitialRequests: 3,

name: true,

cacheGroups: {

default: {

minChunks: 2,

priority: -20

reuseExistingChunk: true,

},

vendors: {

test: /[\\/]node\_modules[\\/]/,

priority: -10

}

}

}

上面是缓存组的默认配置，可以通过default:false禁用默认的缓存组，然后就可以自定义缓存组，将初始化加载时被重复引用的模块进行拆分，就像这样：

cacheGroups: {

commons: {

name: "commons",

chunks: "initial",

minChunks: 2

}

}

之后就随心所欲，可以根据具体的需求，创建多个缓存组：

cacheGroups: {

a: {

// ...

},

b: {

// ...

}

// ...

* }

## **写在后面**

       webpack更新已经过了很多天了，最近算是把更新的基本都过了一遍，在此记录以便日后查看，理解有问题的地方还请不吝赐教。

随笔 - 12  文章 - 0  评论 - 28

# **[webpack4 splitChunksPlugin && runtimeChunkPlugin 配置杂记](https://www.cnblogs.com/ufex/p/8758792.html)**

webpack2 还没研究好，就发现升级到4了，你咋这么快~

最近要做项目脚手架，项目构建准备重新做，因为之前写的太烂了...然后发现webpack大版本已经升到4了(又去看了一眼，4.5了)，这么快的节奏，不适应啊...既然是新的，就用新的来吧，先跳过version3

webpack4 相对于之前的版本，有不少变化，其中包括一些不向下兼容的变化。具体哪些变化这里就不啰嗦了（[4.0 changelog](https://github.com/webpack/webpack/releases/tag/v4.0.0)， [译文](https://zhuanlan.zhihu.com/p/34028750)），今天主要记录一下 废除 commonChunksPlugin 之后，怎么使用 splitChunksPlugin && runtimeChunkPlugin

只记录到自己了解到的部分，不定期更新吧

### **splitChunksPlugin**

splitChunksPlugin 默认配置项 ：

{

"chunks": "all",

"minSize": 0,

"misChunks": 1,

"maxAsyncRequests": 1,

"maxInitialRequests": 1,

"name": undefined,

"automaticNameDelimiter": "~",

"filename": undefined,

"cacheGroups": {}

}

稍微来解释下各项配置：

* **chunks**: all, async, initial 三选一, 插件作用的chunks范围
* **minSize**： 最小尺寸
* **misChunks**：最小chunks
* **maxAsyncRequests**： 最大异步请求chunks
* **maxInitialRequests**: 最大初始化chunks
* **name**： split 的 chunks name
* **automaticNameDelimiter**： 如果不指定name，自动生成name的分隔符（‘runtime~[name]’）
* **filename**: ''
* **cacheGroups**: 字面意思缓存组，主要配置在这里

cacheGroups配置项：

[key]: {

"priority": "缓存优先级权重",

"name": "split 出来的 chunk 的名字",

"chunks": "应该用范围",

"enforce": "未知",

"minSize": "最小尺寸",

"minChunks": "最小chunks",

"maxAsyncRequests": "",

"maxInitialRequests": "",

"filename": "最后output的文件名",

"reuseExistingChunk": "未知"

}

### **runtimeChunkPlugin**

"runtimeChunk": {

"name": "manifest"

}

### **直接上例子了：**

用不着的属性就不配置了，需要关联到 output 和 ExtractTextPlugin 配置

webpackConf: {

...,

output: {

path: path.join(process.cwd(), 'dist'),

publicPath,

filename: '[name].js',

chunkFilename: '[name].js'

},

optimization: {

splitChunks: {

chunks: 'initial', // 只对入口文件处理

cacheGroups: {

vendor: { // split `node\_modules`目录下被打包的代码到 `page/vendor.js && .css` 没找到可打包文件的话，则没有。css需要依赖 `ExtractTextPlugin`

test: /node\_modules\//,

name: 'page/vendor',

priority: 10,

enforce: true

},

commons: { // split `common`和`components`目录下被打包的代码到`page/commons.js && .css`

test: /common\/|components\//,

name: 'page/commons',

priority: 10,

enforce: true

}

}

},

runtimeChunk: {

name: 'page/manifest'

}

},

plugins: [

new ExtractTextPlugin({

filename: '[name].css',

ignoreOrder: true

}),

...

]

...

}

如果没有css文件，说明对应目录下没有相关css资源引用，或者未使用ExtractTextPlugin

dist  
└── page  
├── [commons.css]  
├── commons.js  
├── [manifest.css]  
├── manifest.js  
├── [vendor.css]  
├── vendor.js  
└── demo // entry 中 配置  
├── index.css  
└── index.js  
```

### **写在最后**

上边的例子是用的 optimization 配置项形式，你也可以用 plugin 形式，配置参数应该一样的。

之前搜索相关的配置，介绍的文章较少，有一些简单的demo，但总觉得不受用。  
参考了不少文章，然后又大概读了 webpack 的 splitChunksPlugin 的源码，去了解了下都有哪些参数，看源码挺好