# [html-webpack-plugin详解](http://www.cnblogs.com/wonyun/p/6030090.html)

## 引言

最近在react项目中初次用到了[html-webapck-plugin](https://github.com/ampedandwired/html-webpack-plugin)插件，用到该插件的两个主要作用：

* 为html文件中引入的外部资源如script、link动态添加每次compile后的hash，防止引用缓存的外部文件问题
* 可以生成创建html入口文件，比如单页面可以生成一个html文件入口，配置**N**个html-webpack-plugin可以生成**N**个页面入口

有了这种插件，那么在项目中遇到类似上面的问题都可以轻松的解决。

在本人项目中使用html-webpack-plugin，由于对该插件不太熟悉，开发过程中遇到这样或者那样的问题，下面就来说说这个插件。

## html-webpack-plugin

插件的基本作用就是生成html文件。原理很简单：

将 webpack中`entry`配置的相关入口thunk 和 `extract-text-webpack-plugin`抽取的css样式 插入到该插件提供的`template`或者`templateContent`配置项指定的内容基础上生成一个html文件，具体插入方式是将样式`link`插入到`head`元素中，`script`插入到`head`或者`body`中。

实例化该插件时可以不配置任何参数，例如下面这样：

var HtmlWebpackPlugin = require('html-webpack-plugin')

webpackconfig = {

...

plugins: [

new HtmlWebpackPlugin()

]

}

不配置任何选项的html-webpack-plugin插件，他会默认将webpack中的entry配置所有入口thunk和extract-text-webpack-plugin抽取的css样式都插入到文件指定的位置。例如上面生成的html文件内容如下：

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Webpack App</title>

<link href="index-af150e90583a89775c77.css" rel="stylesheet"></head>

<body>

<script type="text/javascript" src="common-26a14e7d42a7c7bbc4c2.js"></script>

<script type="text/javascript" src="index-af150e90583a89775c77.js"></script></body>

</html>

当然可以使用具体的配置项来定制化一些特殊的需求，那么插件有哪些配置项呢？

## html-webpack-plugin配置项

插件提供的配置项比较多，通过源码可以看出具体的配置项如下：

this.options = \_.extend({

template: path.join(\_\_dirname, 'default\_index.ejs'),

filename: 'index.html',

hash: false,

inject: true,

compile: true,

favicon: false,

minify: false,

cache: true,

showErrors: true,

chunks: 'all',

excludeChunks: [],

title: 'Webpack App',

xhtml: false

}, options);

* **title**: 生成的html文档的标题。配置该项，它并不会替换指定模板文件中的title元素的内容，除非html模板文件中使用了模板引擎语法来获取该配置项值，如下ejs模板语法形式：

<title>{%= o.htmlWebpackPlugin.options.title %}</title>

* **filename**：输出文件的文件名称，默认为**index.html**，不配置就是该文件名；此外，还可以为输出文件指定目录位置（例如'html/index.html'）

**关于filename补充两点：**

1、filename配置的html文件目录是相对于webpackConfig.output.path路径而言的，不是相对于当前项目目录结构的。  
2、指定生成的html文件内容中的link和script路径是相对于生成目录下的，写路径的时候请写生成目录下的相对路径。

* **template**: 本地模板文件的位置，支持加载器(如handlebars、ejs、undersore、html等)，如比如 handlebars!src/index.hbs；

**关于template补充几点：**

1、template配置项在html文件使用file-loader时，其所指定的位置找不到，导致生成的html文件内容不是期望的内容。  
2、为template指定的模板文件没有指定**任何loader的话**，默认使用ejs-loader。如template: './index.html'，若没有为.html指定任何loader就使用ejs-loader

* **templateContent**: string|function，可以指定模板的内容，**不能与template共存**。配置值为function时，可以直接返回html字符串，也可以异步调用返回html字符串。
* **inject**：向template或者templateContent中注入所有静态资源，不同的配置值注入的位置不经相同。

1、**true或者body**：所有**JavaScript**资源插入到body元素的底部  
2、**head**: 所有**JavaScript**资源插入到head元素中  
3、**false**： 所有静态资源css和JavaScript都不会注入到模板文件中

* **favicon**: 添加特定favicon路径到输出的html文档中，这个同title配置项，需要在模板中动态获取其路径值
* **hash**：true|false，是否为所有注入的静态资源添加webpack每次编译产生的唯一hash值，添加hash形式如下所示：  
  html <script type="text/javascript" src="common.js?a3e1396b501cdd9041be"></script>
* **chunks**：允许插入到模板中的一些chunk，不配置此项默认会将entry中所有的thunk注入到模板中。在配置多个页面时，每个页面注入的thunk应该是不相同的，需要通过该配置为不同页面注入不同的thunk；
* **excludeChunks**: 这个与chunks配置项正好相反，用来配置不允许注入的thunk。
* **chunksSortMode**: none | auto| function，默认auto； 允许指定的thunk在插入到html文档前进行排序。  
  >**function**值可以指定具体排序规则；**auto**基于thunk的id进行排序； **none**就是不排序
* **xhtml**: true|fasle, 默认false；是否渲染link为自闭合的标签，true则为自闭合标签
* **cache**: true|fasle, 默认true； 如果为true表示在对应的thunk文件修改后就会emit文件
* **showErrors**: true|false，默认true；是否将错误信息输出到html页面中。这个很有用，在生成html文件的过程中有错误信息，输出到页面就能看到错误相关信息便于调试。
* **minify**: {....}|false；传递 html-minifier 选项给 minify 输出，false就是不使用html压缩。

下面的是一个用于配置这些属性的一个例子：

new HtmlWebpackPlugin({

title:'rd平台',

template: 'entries/index.html', // 源模板文件

filename: './index.html', // 输出文件【注意：这里的根路径是module.exports.output.path】

showErrors: true,

inject: 'body',

chunks: ["common",'index']

})

## 配置多个html页面

html-webpack-plugin的一个实例生成一个html文件，如果单页应用中需要多个页面入口，或者多页应用时配置多个html时，那么就需要实例化该插件多次；

即有几个页面就需要在webpack的plugins数组中配置几个该插件实例：

...

plugins: [

new HtmlWebpackPlugin({

template: 'src/html/index.html',

excludeChunks: ['list', 'detail']

}),

new HtmlWebpackPlugin({

filename: 'list.html',

template: 'src/html/list.html',

thunks: ['common', 'list']

}),

new HtmlWebpackPlugin({

filename: 'detail.html',

template: 'src/html/detail.html',

thunks: ['common', 'detail']

})

]

...

如上例应用中配置了三个入口页面：index.html、list.html、detail.html；并且每个页面注入的thunk不尽相同；类似如果多页面应用，就需要为每个页面配置一个；

## 配置自定义的模板

不带参数的html-webpack-plugin默认生成的html文件只是将thunk和css样式插入到文档中，可能不能满足我们的需求；

另外，如上面所述，三个页面指定了三个不同html模板文件；在项目中，可能所有页面的模板文件可以共用一个，因为html-webpack-plugin插件支持不同的模板loader，所以结合模板引擎来共用一个模板文件有了可能。

所以，配置自定义模板就派上用场了。具体的做法，借助于模板引擎来实现，例如插件没有配置loader时默认支持的ejs模板引擎，下面就以ejs模板引擎为例来说明；

例如项目中有2个入口html页面，它们可以共用一个模板文件，利用ejs模板的语法来动态插入各自页面的thunk和css样式，代码可以这样：

<!DOCTYPE html>

<html style="font-size:20px">

<head>

<meta charset="utf-8">

<title><%= htmlWebpackPlugin.options.title %></title>

<% for (var css in htmlWebpackPlugin.files.css) { %>

<link href="<%=htmlWebpackPlugin.files.css[css] %>" rel="stylesheet">

<% } %>

</head>

<body>

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

<% for (var chunk in htmlWebpackPlugin.files.chunks) { %>

<script type="text/javascript" src="<%=htmlWebpackPlugin.files.chunks[chunk].entry %>"></script>

<% } %>

</body>

</html>

你可能会对代码中的上下文htmlWebpackPlugin数据感到迷惑，这是啥东东？其实这是html-webpack-plugin插件在生成html文件过程中产生的数据，这些数据对html模板文件是可用的。

## 自定义模板上下文数据

html-webpack-plugin在生成html文件的过程中，插件会根据配置生成一个对当前模板可用的特定数据，模板语法可以根据这些数据来动态生成html文件的内容。

那么，插件生成的特殊数据格式是什么，生成的哪些数据呢？从源码或者其官网都给出了答案。从源码中可以看出模板引擎具体可以访问的数据如下:

var templateParams = {

compilation: compilation,

webpack: compilation.getStats().toJson(),

webpackConfig: compilation.options,

htmlWebpackPlugin:

files: assets,

options: self.options

}

};

从中可以看出，有四个主要的对像数据。其中compilation为所有webpack插件提供的都可以访问的一个编译对象，此处就不太做介绍，具体可以自己查资料。下面就对剩下的三个对象数据进行说明。

### webpack

webpack的stats对象；注意一点：

这个可以访问的stats对象是htm文件生成时所对应的stats对象，而不是webpack运行完成后所对应的整个stats对象。

### webpackConfig

webpack的配置项；通过这个属性可以获取webpack的相关配置项，如通过webpackConfig.output.publicPath来获取publicPath配置。当然还可以获取其他配置内容。

### htmlWebpackPlugin

html-webpack-plugin插件对应的数据。它包括两部分：

* htmlWebpackPlugin.files: 此次html-webpack-plugin插件配置的chunk和抽取的css样式。该files值其实是webpack的stats对象的assetsByChunkName属性代表的值，该值是插件配置的chunk块对应的按照webpackConfig.output.filename映射的值。例如对应上面配置插件各个属性配置项例子中生成的数据格式如下：

"htmlWebpackPlugin": {

"files": {

"css": [ "inex.css" ],

"js": [ "common.js", "index.js"],

"chunks": {

"common": {

"entry": "common.js",

"css": [ "index.css" ]

},

"index": {

"entry": "index.js",

"css": ["index.css"]

}

}

}

}

这样，就可以是用如下模板引擎来动态输出script脚本

<% for (var chunk in htmlWebpackPlugin.files.chunks) { %>

<script type="text/javascript" src="<%=htmlWebpackPlugin.files.chunks[chunk].entry %>"></script>

<% } %>

* htmlWebpackPlugin.options: 传递给插件的配置项，具体的配置项如上面插件配置项小节所描述的。

## 插件事件

不知道你发现没有，html-webpack-plugin插件在插入静态资源时存在一些问题：

* **在插入js资源只能插入head或者body元素中，不能一些插入head中，另一些插入body中**
* **不支持在html中文件内联**\*，例如在文件的某个地方用<script src="xxx.js?\_\_inline"></script>来内联外部脚本

为此，有人专门给插件作者提问了[这个问题](https://github.com/ampedandwired/html-webpack-plugin/issues/157)；对此插件作者提供了插件事件，允许**其他插件**来改变html文件内容。具体的事件如下：

Async（异步事件）:

\* html-webpack-plugin-before-html-generation

\* html-webpack-plugin-before-html-processing

\* html-webpack-plugin-alter-asset-tags

\* html-webpack-plugin-after-html-processing

\* html-webpack-plugin-after-emit

Sync（同步事件）:

\* html-webpack-plugin-alter-chunks

这些事件是提供给其他插件使用的，用于改变html的内容。因此，要用这些事件需要提供一个webpack插件。例如下面定义的MyPlugin插件。

function MyPlugin(options) {

// Configure your plugin with options...

}

MyPlugin.prototype.apply = function(compiler) {

// ...

compiler.plugin('compilation', function(compilation) {

console.log('The compiler is starting a new compilation...');

compilation.plugin('html-webpack-plugin-before-html-processing', function(htmlPluginData, callback) {

htmlPluginData.html += 'The magic footer';

callback(null, htmlPluginData);

});

});

};

module.exports = MyPlugin;

然后，在webpack.config.js文件中配置Myplugin信息：

plugins: [

new MyPlugin({options: ''})

]