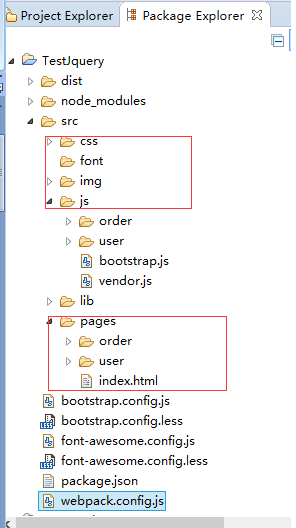
本篇文章主要介绍了"webpack集成bootstrap进行多页面开发"，主要涉及到方面的内容，对于web前端感兴趣的同学可以参考一下： webpack还是很有搞头的，再搞搞gulp都不用了。。。最近以jquery、bootstrap为例，搞了个多页面，多入口开发结构，mark一下。开发人员主要集.

webpack还是很有搞头的，再搞搞gulp都不用了。。。

最近以jquery、bootstrap为例，搞了个多页面，多入口开发结构，mark一下。

开发人员主要集中在以上三个目录，整理js/css/img/page

最终通过webpack打包到dist目录，打出目录结构



因为使用extract-text-webpack-plugin插件，提取css文件名，这个坑爹有点，后续通过改造这个插件，可以单独打出来css文件夹，实践过，又忘了。。。。

ok，上webpack.config.js

var path = require('path');

var glob = require('glob');

var webpack = require('webpack');

var ExtractTextPlugin = require("extract-text-webpack-plugin");

var CommonsChunkPlugin = require("webpack/lib/optimize/CommonsChunkPlugin");

// CSS浏览器前缀问题

var autoprefixer = require('autoprefixer');

var precss = require('precss');

// 多入口文件

function getEntrys() {

var entrys = {};

var src = new RegExp(\_\_dirname.replace(/\\/g, "/") + "/src/");

glob.sync(\_\_dirname + '/src/js/\*\*/\*.js').forEach(function(name) {

// 前缀

var entry = name.replace(src, "");

// 后缀

entry = entry.replace(/\.js$/, "");

entrys[entry] = name;

});

return entrys;

};

module.exports = {

context : \_\_dirname + '/src',

entry : getEntrys(),

output : {

path : path.resolve(\_\_dirname, './dist'),

publicPath : '/dist/',

filename : '[name].js'

},

resolveLoader : {

root : path.join(\_\_dirname, 'node\_modules')

},

resolve : {

root : path.join(\_\_dirname, 'node\_modules'),

extensions : [ '', '.js' ],

alias : {}

},

plugins : [

// 全局依赖jQuery

new webpack.ProvidePlugin({

$ : "jquery",

jQuery : "jquery",

"window.jQuery" : "jquery"

}),

// 提取依赖的jQuery通用插件

new CommonsChunkPlugin({

name : "js/vendor",

minChunks : Infinity

}),

// 提取CSS文件

new ExtractTextPlugin("[name].css"),

// 查找相等或近似的模块，避免在最终生成的文件中出现重复的模块

new webpack.optimize.DedupePlugin() ],

module : {

loaders : [

{

test : /\.js$/,

loader : 'babel',

exclude : /node\_modules/

},

{

test : /\.jsx$/,

loader : 'babel',

exclude : /node\_modules/

},

{

test : /\.json$/,

loader : 'json'

},

{

test : /\.(png|jpg|gif)$/,

loader : 'url',

query : {

limit : 10000,

// CSS图片目录

name : '[path][name]\_[hash].[ext]'

}

},

{

test : /\.less$/,

loader : ExtractTextPlugin.extract("style-loader",

"css-loader!postcss-loader")

},

{

test : /\.css$/,

loader : 'style!css',

loader : ExtractTextPlugin.extract("style-loader",

"css-loader!postcss-loader")

}, {// bootstrap font-awesome

test : /\.(woff|woff2)(\?v=\d+\.\d+\.\d+)?$/,

loader : 'url',

query : {

limit : 10000,

mimetype : 'application/font-woff',

// 字体文件放置目录

name : 'font/bootstrap/[name]\_[hash].[ext]'

}

}, {// bootstrap

test : /\.ttf(\?v=\d+\.\d+\.\d+)?$/,

loader : 'url',

query : {

limit : 10000,

mimetype : 'application/octet-stream',

// 字体文件放置目录

name : 'font/bootstrap/[name]\_[hash].[ext]'

}

}, {// bootstrap

test : /\.eot(\?v=\d+\.\d+\.\d+)?$/,

loader : 'file',

query : {

limit : 10000,

// 字体文件放置目录

name : 'font/bootstrap/[name]\_[hash].[ext]'

}

}, {// bootstrap

test : /\.svg(\?v=\d+\.\d+\.\d+)?$/,

loader : 'url',

query : {

limit : 10000,

mimetype : 'application/image/svg+xml',

// 字体文件放置目录

name : 'font/bootstrap/[name]\_[hash].[ext]'

}

}, {// font-awesome

test : /\.(ttf|eot|svg)(\?v=[0-9]\.[0-9]\.[0-9])?$/,

loader : "file",

query : {

limit : 10000,

// 字体文件放置目录

name : 'font/bootstrap/[name]\_[hash].[ext]'

}

}, {// 如果要加载jQuery插件,解析路径&参数

test : "/lib/jquery/\*\*/\*.js$",

loader : "'imports?jQuery=jquery,$=jquery,this=>window"

} ]

},

postcss : function() {

return [ autoprefixer({

browsers : [ 'not ie <= 8' ]

}), precss ];

},

devServer : {

historyApiFallback : true,

noInfo : true,

// 其实很简单的，只要配置这个参数就可以了

proxy : {

'/v1/\*' : {

target : 'http://localhost:3000/',

secure : false

}

}

},

devtool : 'eval-source-map'

}

if (process.env.NODE\_ENV === 'production') {

module.exports.devtool = 'source-map'

// http://vuejs.github.io/vue-loader/workflow/production.html

module.exports.plugins = (module.exports.plugins || []).concat([

new webpack.DefinePlugin({

'process.env' : {

NODE\_ENV : '"production"'

}

}), new webpack.optimize.UglifyJsPlugin({

compress : {

warnings : false

}

}), new webpack.optimize.OccurenceOrderPlugin() ])

}

 关于vendor.js、bootstrap.js的内容，以往记录过，就是个集合而已package.json也有，额外加了glob依赖。

现在面临如下几个问题：

1.css文件打包到dist目录没有形成单独的文件夹，受限于css提取插件，可改造实现

2.现在只有bootstrap、font-awesome字体文件，放到bootstrap文件夹了，其他字体文件呢，其实缺乏识别bootstrap、font-awesome的有效标识

3.dist目录下，除了js、css文件，其他文件都有hash值，还缺乏最后一步，将js、css文件hash处理，然后替换掉pages目录下的页面引用，这一步可以考虑gulp实现，在开发阶段，是不能直接hash js、css文件的，做了hash开发人员怎么办。。。

以上就介绍了webpack集成bootstrap进行多页面开发，包括了方面的内容，希望对web前端有兴趣的朋友有所帮助。