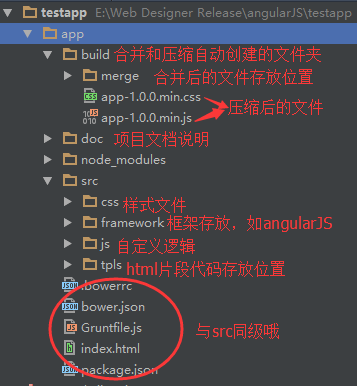
**本项目采用grunt+bower管理**

文件目录：



首先创建项目文件名为app，使用命令提示符cmd调出DOS界面，操作如下：



效果如下：



接下来输入命令npm init根据提示配置，最后生成package.json文件。这个package.json文件其实是node.js用来描述一个项目文件的，JSON格式。

生成的package.json文件如下：

{  
 "name": "app",  
 "version": "1.0.0",  
 "description": "一个例子",  
 "main": "index.js",  
 "scripts": {  
 "test": "echo \"Error: no test specified\" && exit 1"  
 },  
 "author": "lsdt",  
 "license": "MIT"  
}

**Grunt操作**

首先安装grunt，输入命令npm install –save-dev grunt，表示通过npm安装了grunt到当前项目，同时加上了—save-dev参数，表示会把刚安装的东西添加到package.json文件中，这时package.json文件中会出现如下情况：

"devDependencies": {  
 "grunt": "^0.4.5"  
}

1、创建Grunt配置文件

在app目录下手动创建Gruntfile.js文件，内容如下：

module.exports = **function**(grunt) {  
 grunt.initConfig({  
 });  
};

2、安装load-grunt-tasks插件

load-grunt-tasks插件会自动读取并加载项目package.json文件中devDependencies配置下以grunt-\*开头的依赖库，效果如下：

require('load-grunt-tasks')(grunt);  
// 就代替了以下全部  
grunt.loadNpmTasks('grunt-contrib-uglify');  
grunt.loadNpmTasks('grunt-contrib-jshint');  
grunt.loadNpmTasks('grunt-contrib-qunit');  
grunt.loadNpmTasks('grunt-contrib-watch');  
grunt.loadNpmTasks('grunt-contrib-concat');

在Gruntfile.js引入：

module.exports = **function**(grunt) {  
 require("load-grunt-tasks")(grunt);  
 grunt.initConfig({  
 });  
};

3、安装grunt-contrib-concat插件

concat插件用来将多个文件合并成一个文件。安装该插件的命令是npm install –save-dev grunt-contrib-concat，安装完成后可以看到package.json里多了它。

在Gruntfile.js对该插件进行配置并测试

module.exports = **function**(grunt) {  
 //加载package.json中devDependencies里的所有依赖模块  
 require("load-grunt-tasks")(grunt);  
 grunt.initConfig({  
 //读取package.json文件  
 pkg: grunt.file.readJSON('package.json'),  
 //concat用来合并js文件  
 concat: {  
 options: {  
 // 定义一个用于插入合并输出文件之间的字符  
 separator: ';'  
 },  
 dist: {  
 // 将要被合并的文件  
 src: ['src/js/\*\*/\*.js'],  
 // 合并后的JS文件的存放位置  
 dest: 'build/merge/<%= pkg.name %>.js'  
 }  
 }  
 });  
  
 */\*\*\*启动任务,grunt start,start这个名字可以是其它自定义名字\*\*\*/* grunt.registerTask("start","启动",**function** () {  
 grunt.task.run([  
 "concat"  
 ]);  
 });  
};

在app/src/js目录下手动创建两个JS文件，xx.js和oo.js，然后输入命令grunt start，可以看到在app/build/merge/目录下有一个app.js，这时xx和oo合并的结果。

4、安装grunt-contrib-uglify插件

uglify插件是用来压缩JS文件的，安装该插件的命令为：

npm install --save-dev grunt-contrib-uglify。

安装完后在Gruntfile.js中进行配置并测试：

module.exports = **function**(grunt) {  
 //加载package.json中devDependencies里的所有依赖模块  
 require("load-grunt-tasks")(grunt);  
 grunt.initConfig({  
 //读取package.json文件  
 pkg: grunt.file.readJSON('package.json'),  
 //concat用来合并JS文件  
 concat: {  
 options: {  
 separator: ";"//定义一个用于插入合并输出文件之间的字符  
 },  
 js: {  
 src: ["src/js/\*\*/\*.js"],//将要被合并的文件  
 dest: "build/merge/<%= pkg.name %>.js"//合并后的JS文件所存放的路径  
 }  
 },  
  
 //uglify用来压缩JS文件  
 uglify: {  
 options: {  
 // 此处定义的banner注释(插入名字+版本和插入时间)将插入到输出文件的顶部  
 banner: "/\*! <%= pkg.name %>-<%= pkg.version %> <%= grunt.template.today('dd-mm-yyyy') %> \*/\n"  
 },  
 js: {  
 //uglify会自动压缩concat任务中生成的文件  
 expand: **true**,//启用下面的选项  
 cwd: "build/merge/",//待压缩文件目录  
 src: "\*\*/\*.js",//指定压缩的文件名后缀  
 dest: "build",//压缩后文件存放路径  
 ext: "-<%= pkg.version %>.min.js"//压缩成的文件使用版本+.min.js替换原有扩展名。  
 }  
 }  
 });  
  
 */\*\*\*启动任务,grunt start,start这个名字可以是其它自定义名字\*\*\*/* grunt.registerTask("start","启动",**function** () {  
 grunt.task.run([  
 "concat",  
 "uglify"  
 ]);  
 });  
};

5、安装grunt-contrib-jshint插件

jshint插件是用来对JS的代码规范进行检查的，安装该插件的命令为：

npm install --save-dev grunt-contrib-jshint。

安装完后在Gruntfile.js中进行配置并测试：

//jshint用来检查js代码规范  
jshint: {  
 files: ["Gruntfile.js","src/js/\*\*/\*.js"],//要进行js检查的文件  
 //这里是覆盖jshint默认配置的选项  
 globals: {  
 jQuery: **true**,  
 console: **true**,  
 module: **true**,  
 document: **true** }  
},

当输入grunt start，jshint便会对文件进行检查，并在控制台输出检查结果。

6、安装grunt-contrib-cssmin插件

cssmin插件是用来压缩CSS文件的，安装该插件的命令为：

npm install --save-dev grunt-contrib-cssmin。

安装完后首先在app/src/css/目录下创建app.css，然后在Gruntfile.js中进行配置并测试：

//cssmin用来压缩CSS文件  
cssmin: {  
 options: {  
 // 此处定义的banner注释(插入名字+版本和插入时间)将插入到输出文件的顶部  
 banner: "/\*! <%= pkg.name %>-<%= pkg.version %> <%= grunt.template.today('dd-mm-yyyy') %> \*/\n"  
 },  
 css: {  
 expand: **true**,//启用下面的选项  
 cwd: "src/css/",//待压缩文件目录  
 src: "\*\*/\*.css",//指定压缩的文件名后缀  
 dest: "build",//压缩后文件存放路径  
 ext: "-<%= pkg.version %>.min.css"//压缩成的文件使用版本+.min.js替换原有扩展名。  
 }  
},

7、安装grunt-contrib-watch插件

watch插件是用来监听文件的，当文件发生变化时会执行tasks中指定的任务，安装该插件的命令为：

npm install --save-dev grunt-contrib-watch。

//watch用来监听文件，当文件发生变化时会执行tasks中指定的任务  
watch: {  
 options: {  
 livereload: **true** },  
 // '\*\*' 表示包含所有的子目录  
 // '\*' 表示包含所有的文件  
 files: ["<%= jshint.files %>", "\*.html","src/tpls/\*.html","src/css/\*"],//监听的文件  
 tasks: ["jshint","cssmin:css"]//文件发生改变时要做的事情  
},

在命令行输入grunt start之后，会发现watch启动，一直在监听文件的变化，当监听的文件发生变化时，会执行jshint命令来进行检查，cssmin命令来压缩css文件。

8、安装grunt-contrib-connect插件

connect插件可以让前端项目开启connect服务，安装该插件的命令为：

npm install --save-dev grunt-contrib-connect。

//通过connect任务，创建一个静态服务器  
connect: {  
 options: {  
 port: 9000,// 服务器端口号  
 hostname: "localhost",//服务器地址(可以使用主机名localhost，也能使用IP)  
 protocol: "http",//超文本传输协议,如http,https  
 open: **true**,//表示静态服务启动之后是否以默认浏览器打开首页,即base.options.index指定的页面  
 base: {  
 path: "./",//配置站点的根目录，这里把根目录配置成了当前的项目文件夹(./)  
 options: {  
 index: "index.html"  
 }  
 },  
 livereload: **true** },  
 **default**: {}  
}

通过grunt start启动之后，会自动打开浏览器，打开app/index.html。

9、安装grunt-wiredep插件

wiredep插件用来自动将bower\_components中的包加入到index.html中，安装该插件的命令为：

npm install --save-dev grunt-wiredep。

安装完成后，为了使用bower.json文件中的依赖库，需要在index.html中注入：

JS：

<!-- bower:js -->  
<!-- endbower -->

CSS:

<!-- bower:css -->  
<!-- endbower -->

注入后如下：

<!DOCTYPE html>  
<html>  
<head lang="en">  
 <meta charset="UTF-8">  
 <title>grunt+angular</title>  
 <!-- bower:css -->  
 <!-- endbower -->  
</head>  
<body>  
  
  
<!-- bower:js -->  
<!-- endbower -->  
</body>  
</html>

wiredep插件在Gruntfile.js中的配置

//wiredep自动将bower\_components中的包加入到index.html中  
wiredep: {  
 task: {  
 src: ["index.html"]  
 }  
},

对于wiredep插件的测试，在使用bower安装angularJS和boostrap时进行。

**Bower操作**

1、设置bower所下载的包(如：angular、bootstrap等)放到我们指定的位置，这里我把他们放在了app/src/framework/目录下，首先操作如下：

在app/目录下创建1.txt，然后在命令行输入rename 1.txt .bowerrc，这时我们可以在项目中我们创建的1.txt变成了 .bowerrc文件，它是自定义bower下载的代码包的目录。然后在 .bowerrc文件内写入：

{  
 "directory" : "src/framework"  
}

这样的话，用bower下载angular、bootstrap等就会把他们放在app/src/ framework目录下。

为了让wiredep插件能够自动将bower安装的插件注入到index.html中，我们先生成bower的配置文件，在命令行输入bower init，会提示你输入一些信息，根据提示输入或按回车跳过即可，最后生成bower.json，它是用来保存本项目的配置的。

{  
 "name": "app",  
 "description": "app",  
 "main": "Gruntfile.js",  
 "authors": [  
 "lsdt"  
 ],  
 "license": "MIT",  
 "homepage": "",  
 "private": **true**,  
 "ignore": [  
 "\*\*/.\*",  
 "node\_modules",  
 "bower\_components",  
 "src/framework",  
 "test",  
 "tests"  
 ]  
}

2、用bower安装项目所需的包

A、安装angular，命令为bower install angular –save grunt，安装完成后可以看到在bower.json中读到了grunt和angularJS：

"dependencies": {  
 "grunt": "^1.0.1",  
 "angular": "^1.6.3"  
}

然后在命令行中输入grunt start，运行后可以看到在index.html中wiredep帮我们加入了angular.js。

顺带说一下，如果想让wiredep帮我们注入的是 .min.js，而不是 .js的话，我们需要在bower.json加入如下配置：

"overrides": {  
 "angular": {  
 "main": [  
 "angular.min.js"  
 ]  
 }  
}

wiredep帮我们在index.html中注入angularJS后的效果：

<!DOCTYPE html>  
<html>  
<head lang="en">  
 <meta charset="UTF-8">  
 <title>grunt+angular</title>  
 <!-- bower:css -->  
 <!-- endbower -->  
</head>  
<body>  
Web内容  
<!-- bower:js -->  
<script src="src/framework/angular/angular.min.js"></script>  
<!-- endbower -->  
</body>  
</html>

B、安装bootstrap，命令为bower install bootstrap –save，这时你会看到bower在下载bootstrap时，还下载了bootstrap所要依赖的jQuery。

为了让wiredep帮我们在index.html中注入的是 .min.js或 .min.css，我们在bower.json中需要添加如下设置：

"overrides": {  
 "angular": {  
 "main": [  
 "angular.min.js"  
 ]  
 },  
 "jquery": {  
 "main": [  
 "dist/jquery.min.js"  
 ]  
 },  
 "bootstrap": {  
 "main": [  
 "dist/js/bootstrap.min.js",  
 "dist/css/bootstrap.min.css",  
 "less/bootstrap.less"  
 ]  
 }  
}

很显然这跟angular的设置是一样的。

接下来在命令行中输入grunt start，可以看到：

<!DOCTYPE html>  
<html>  
<head lang="en">  
 <meta charset="UTF-8">  
 <title>grunt+angular</title>  
 <!-- bower:css -->  
 <link rel="stylesheet" href="src/framework/bootstrap/dist/css/bootstrap.min.css" />  
 <!-- endbower -->  
</head>  
<body>  
  
Web内容  
  
<!-- bower:js -->  
<script src="src/framework/angular/angular.min.js"></script>  
<script src="src/framework/jquery/dist/jquery.min.js"></script>  
<script src="src/framework/bootstrap/dist/js/bootstrap.min.js"></script>  
<!-- endbower -->  
</body>  
</html>

**最后**

**1、完整的package.json**

{  
 "name": "app",  
 "version": "1.0.0",  
 "description": "app",  
 "main": "Gruntfile.js",  
 "scripts": {  
 "test": "echo \"Error: no test specified\" && exit 1"  
 },  
 "author": "lsdt",  
 "license": "MIT",  
 "devDependencies": {  
 "connect-livereload": "^0.6.0",  
 "grunt": "^1.0.1",  
 "grunt-contrib-concat": "^1.0.1",  
 "grunt-contrib-connect": "^1.0.2",  
 "grunt-contrib-cssmin": "^2.0.0",  
 "grunt-contrib-jshint": "^1.1.0",  
 "grunt-contrib-uglify": "^2.2.0",  
 "grunt-contrib-watch": "^1.0.0",  
 "grunt-wiredep": "^3.0.1",  
 "load-grunt-tasks": "^3.5.2"  
 }  
}

**2、完整的bower.json**

{  
 "name": "app",  
 "description": "app",  
 "main": "Gruntfile.js",  
 "authors": [  
 "lsdt"  
 ],  
 "license": "MIT",  
 "homepage": "",  
 "private": **true**,  
 "ignore": [  
 "\*\*/.\*",  
 "node\_modules",  
 "bower\_components",  
 "src/framework",  
 "test",  
 "tests"  
 ],  
 "overrides": {  
 "angular": {  
 "main": [  
 "angular.min.js"  
 ]  
 },  
 "jquery": {  
 "main": [  
 "dist/jquery.min.js"  
 ]  
 },  
 "bootstrap": {  
 "main": [  
 "dist/js/bootstrap.min.js",  
 "dist/css/bootstrap.min.css",  
 "less/bootstrap.less"  
 ]  
 }  
 },  
 "dependencies": {  
 "grunt": "^1.0.1",  
 "angular": "^1.6.3",  
 "bootstrap": "^3.3.7"  
 }  
}

**3、完整的Gruntfile.js**

module.exports = **function** (grunt) {  
  
 */\*\*\*加载package.json中devDependencies里的所有依赖模块\*\*\*/* require("load-grunt-tasks")(grunt);  
  
 */\*\*\*任务配置,所有插件的配置信息\*\*\*/* grunt.initConfig({  
 //读取package.json文件  
 pkg: grunt.file.readJSON("package.json"),  
  
 //concat用来合并JS文件  
 concat: {  
 options: {  
 separator: ";"//定义一个用于插入合并输出文件之间的字符  
 },  
 js: {  
 src: ["src/js/\*\*/\*.js"],//将要被合并的文件  
 dest: "build/merge/<%= pkg.name %>.js"//合并后的JS文件所存放的路径  
 }  
 },  
 //uglify用来压缩JS文件  
 uglify: {  
 options: {  
 // 此处定义的banner注释(插入名字+版本和插入时间)将插入到输出文件的顶部  
 banner: "/\*! <%= pkg.name %>-<%= pkg.version %> <%= grunt.template.today('dd-mm-yyyy') %> \*/\n"  
 },  
 js: {  
 //uglify会自动压缩concat任务中生成的文件  
 expand: **true**,//启用下面的选项  
 cwd: "build/merge/",//待压缩文件目录  
 src: "\*\*/\*.js",//指定压缩的文件名后缀  
 dest: "build",//压缩后文件存放路径  
 ext: "-<%= pkg.version %>.min.js"//压缩成的文件使用版本+.min.js替换原有扩展名。  
 }  
 },  
 //jshint用来检查js代码规范  
 jshint: {  
 files: ["Gruntfile.js","src/js/\*\*/\*.js"],//要进行js检查的文件  
 //这里是覆盖jshint默认配置的选项  
 globals: {  
 jQuery: **true**,  
 console: **true**,  
 module: **true**,  
 document: **true** }  
 },  
 //cssmin用来压缩CSS文件  
 cssmin: {  
 options: {  
 // 此处定义的banner注释(插入名字+版本和插入时间)将插入到输出文件的顶部  
 banner: "/\*! <%= pkg.name %>-<%= pkg.version %> <%= grunt.template.today('dd-mm-yyyy') %> \*/\n"  
 },  
 css: {  
 expand: **true**,//启用下面的选项  
 cwd: "src/css/",//待压缩文件目录  
 src: "\*\*/\*.css",//指定压缩的文件名后缀  
 dest: "build",//压缩后文件存放路径  
 ext: "-<%= pkg.version %>.min.css"//压缩成的文件使用版本+.min.js替换原有扩展名。  
 }  
 },  
  
 //watch用来监听文件，当文件发生变化时会执行tasks中指定的任务  
 watch: {  
 options: {  
 livereload: **true** },  
 // '\*\*' 表示包含所有的子目录  
 // '\*' 表示包含所有的文件  
 files: ["<%= jshint.files %>", "\*.html","src/tpls/\*.html","src/css/\*"],//监听的文件  
 tasks: ["jshint","cssmin:css"]//文件发生改变时要做的事情  
 },  
  
 //wiredep自动将bower\_components中的包加入到index.html中  
 wiredep: {  
 task: {  
 src: ["index.html"]  
 }  
 },  
  
 //通过connect任务，创建一个静态服务器  
 connect: {  
 options: {  
 port: 9000,// 服务器端口号  
 hostname: "localhost",//服务器地址(可以使用主机名localhost，也能使用IP)  
 protocol: "http",//超文本传输协议,如http,https  
 open: **true**,//表示静态服务启动之后是否以默认浏览器打开首页,即base.options.index指定的页面  
 base: {  
 path: "./",//配置站点的根目录，这里把根目录配置成了当前的项目文件夹(./)  
 options: {  
 index: "index.html"  
 }  
 },  
 livereload: **true** },  
 **default**: {}  
 }  
 });  
  
 */\*\*\*启动任务,grunt start,start这个名字可以是其它自定义名字\*\*\*/* grunt.registerTask("start","启动",**function** () {  
 grunt.task.run([  
 "concat",  
 "uglify",  
 "jshint",  
 "cssmin",  
 "wiredep",  
 "connect:default",  
 "watch"  
 ]);  
 });  
};

**文献参考**

1、Grunt搭建AngularJS项目

http://blog.csdn.net/ch717828/article/details/50339087

2、npm package.json属性详解

http://www.cnblogs.com/tzyy/p/5193811.html#\_h1\_10

3、编写可维护的Gruntfile.js

http://www.tuicool.com/articles/yABV73

4、bower简明入门教程

https://segmentfault.com/a/1190000002971135

5、利用grunt-contrib-connect和grunt-connect-proxy搭建前后端分离的开发环境

<http://www.mamicode.com/info-detail-1222744.html>

6、自动化之Bower+grunt-wiredep依赖库自动注入介绍

http://div.io/topic/1332

7、<https://www.npmjs.com/package/connect-livereload>

8、<https://www.npmjs.com/package/grunt-contrib-watch>

9、https://www.npmjs.com/package/grunt-contrib-connect