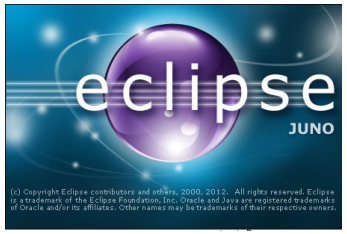
前端自动化构建之旅

—— 初探Grunt 构建工具

为什么是Grunt？

Why Grunt?

传统的开发流程

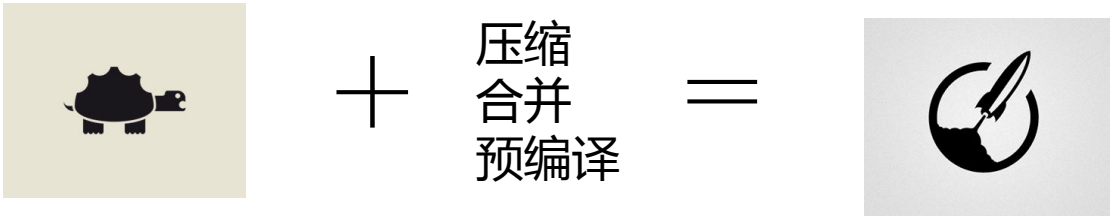


编码　　调试　　上线

追求更好的用户体验

1. 运用新技术改善交互：

2. 优化代码提升性能：



当传统流程遇见新技术

编码

手动 Uglify 对JavaScript、CSS压缩混淆

手动 HTMLMin 对HTML压缩

手动 imagemin 批量压缩图片

手动 合并静态文件，减少页面请求

手动修改文件引用路径

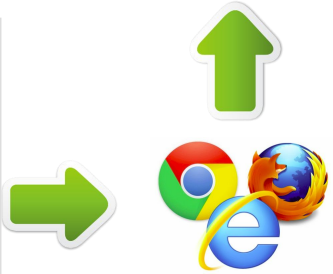
上线

手动 SASS/LESS/CoffeeScript 编译

手动 Handlebars 模版预编译

手动 JSHint 代码质量检测

手动 Jasmine/QUnit 单元测试



调试

每一步，都要手动执行！

难道创造好的用户体验真的就那么难？

Grunt简介

简单说，Grunt是一个自动任务运行器，会按照预先设定的顺序自动运行一系列的任务。这可以简化工作流程，减轻重复性工作带来的负担。

Grunt的安装与运行

前提：

Grunt需要Nodejs环境，这里假设你已经安装好了Nodejs和NPM，因为下面的代码需要在命令行中运行，所以推荐安装一个好用的Shell工具。

安装：

```
1. npm install -g grunt-cli        // 使用NPM安装 grunt-cli
```

运行：

```
1. cd ~/workspace/project/WebApp   // 进入已配置Grunt模块的项目目录

2. grunt build                     // 运行Grunt，执行相应任务
```

小贴士：Windows下推荐使用Gow, MinGW MSYS, Cygwin等工具替代cmd.exe

Grunt 项目的结构

注意：

Grunt的核心模块并非全局安装，而是在项目中单独存在，保持项目使用Grunt插件与Grunt核心模块版本一致，避免由于版本不同造成冲突。同时，Grunt是非侵入式的，对项目原有结构影响较小。

```
WebApp/
├─ grunt/                          // 推荐将Grunt配置在前端WebApp的根目录
│  ├─ Gruntfile.js                 // Grunt的任务配置文件
│  ├─ package.json                 // 项目信息，模块依赖声明
│  └─ node_modules/                // 存放相关模块
│     ├─ grunt/                    // Grunt的核心模块
│     ├─ grunt-contrib-sass/       // Grunt的第三方插件
│     └─ ...
├─ res/                            // 项目中的其他文件
│  ├─ theme/
│  ├─ common/
│  └─ ...
├─ index.html
└─ ...
```

Grunt的构成

统一管理Grunt，
模版化配置

核心配置文件：Gruntfile.js

➢Gruntfile.js是Grunt的配置文件，不同于常见的XML格式，它采用的是JavaScript的代码进行配置。整个Gruntfile.js就是一个符合Node.js标准的JavaScript模块。

Gruntfile主要有三个方法：

```
1. grunt.initConfig          // 定义模块的参数配置
2. grunt.loadNpmTasks        // 引用声明，完成任务所需的模块加载
3. grunt.registerTask        // 定义具体的任务
```

小贴士：和其他的NodeJS模块一样，Gruntfile.js遵循CommonJS模块化规范

核心配置文件：Gruntfile.js


```
module.exports = function(grunt) {

  // Project configuration.                                   任务配置代码
  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json'),
    uglify: {
      options: {
        banner: '/*! <%= pkg.name %> <%= grunt.template.today("yyyy-mm-dd") %>
      },
      build: {
        src: 'src/<%= pkg.name %>.js',
        dest: 'build/<%= pkg.name %>.min.js'
      }
    }
  });

  // Load the plugin that provides the "uglify" task.         插件加载声明
  grunt.loadNpmTasks('grunt-contrib-uglify');

  // Default task(s).                                         定义任务组合
  grunt.registerTask('default', ['uglify']);

};
```

Gruntfile.js是模块化的JavaScript文件，非常灵活，可以自行扩展。

✓基于任务配置
✓第三方模块调用
✓提供API实现扩展

配置详解 - grunt.initConfig

➢grunt.initConfig方法用于模块配置，它接受一个对象作为参数。对象中每一个成员项对应一个同名模块。这里我们用uglify模块的配置作为演示：

配置文件中的uglify属性指向一个对象，该对象又包含多个成员。除了一些系统设定的成员（比如options），其他自定义的成员称为目标（target）。一个模块可以有多个目标。

uglify模块有一个目标："minify",它用于压缩指定路径的JS文件，其中配置了需要压缩的文件路径、文件类型和过滤条件。

```
grunt.initConfig({
     uglify:{
          options:{        // 配置uglify的参数
               mangle: { except: ['jQuery'] },       // 防止混淆变量名时对jQuery产生影响
               banner: '/*\n Minified by Uglify <%=grunt.template.today("yyyy-MM-dd-
HH:mm:ss")%>*/\n',
               footer: '\n/* Powered by AILK UED */' // 在头部和尾部增加声明
          },
          minify:{         // 配置uglify的执行目标
               files:[
                    {
                         expand: true,
                         cwd: 'WebApp/res/',          // 待压缩目录
                         src: ['**/*.js','!**/*.min.js','!**/full.js'], // 过滤带压缩文件
                         dest: 'WebApp/res/',         // 压缩后目录
                         ext: '.min.js'               // 对压缩后的文件使用.min.js的后缀
                    }
               ]
          }
     }
});

grunt.loadNpmTasks('grunt-contrib-uglify');          // 对uglify插件进行加载声明
```

配置详解 - grunt.loadNpmTasks

➢使用grunt.loadNpmTasks方法声明需要载入的模块文件。

```
grunt.loadNpmTasks('grunt-contrib-uglify');
```

需要使用几个模块，这里就要写几条grunt.loadNpmTasks语句，将各个模块逐一加载。

如果加载模块很多，这部分会非常冗长。这里有一个解决办法，就是安装load-grunt-tasks模块，然后在Gruntfile.js文件中，用下面的语句替代所有的grunt.loadNpmTasks语句。Grunt便可以自动分析package.json文件，自动加载所找到的grunt模块。

```
require('load-grunt-tasks')(grunt);
```

配置详解 - grunt.registerTask

➢使用grunt.registerTask方法定义具体的任务。

```
grunt.registerTask('init',['uglify','sass','watch']);
```

第一个参数为任务名，第二个参数是一个数组，表示该任务需要依次使用的模块。如上述代码中，init表示任务名，在该任务中，会依次执行uglify、sass和watch三个模块。

需要执行该任务，在命令行输入以下命令：

```
$ grunt init ↵
```

Grunt实例讲解 - 项目描述

下面借助一个实例，进行讲解。

- 当前项目中应用了SASS技术，并且在较少重构已有代码的前提下，希望可以进一步提升的页面加载速度。
- 项目的目录结构：

```
项目目录
├─ WebApp/                  // 前端目录
│   ├─ res/
│   │   ├─ common/          // 存放具有通用性js文件的目录，需要压缩合并
│   │   │   └─ **.js
│   │   ├─ theme/           // 存放样式的目录
│   │   │   └─ **.scss
│   │   └─ ...
│   ├─ index.html           // HTML页面
│   └─ ...
└─...
```

- 那么，我们可以总结出以下几个需求：

1. 转换theme/文件夹下的.scss文件为同名.css文件
2. 压缩common/文件夹中的.js文件到.min.js结尾的同名文件
3. 在需要的时候将已压缩的JavaScript文件合并到full.js中

Grunt实例讲解 - 配置

- 将包含所需插件node_modules文件夹和package.json复制到项目的根目录
- 配置Gruntfile.js，内容如下所示：

```
module.exports = function(grunt) {
grunt.initConfig({
     uglify:{
          options:{/* 相关配置请参见之前的页面 */},
          minify:{/* 相关配置请参见之前的页面 */},
          dynamic:{
               expand: true,ext: '.min.js',src:''
          }},
     sass:{
          compile: {
               files: [{
                    expand:true, src: ['*.scss'],
                    cwd: 'WebApp/res/theme/',
                    dest: 'WebApp/res/theme/',
                    ext: '.css'}]},
          dynamic: {
               expand:true,ext:'.css', src:''}},
     concat:{
          options:{separator:';'},
          target:{
               src:['WebApp/res/**/*.min.js'],
               dest:'WebApp/res/full.js'}},
     watch:{
          js:{
               files:['WebApp/res/**/*.js',
'!**/*.min.js','!**/full.js'],
               tasks:['uglify:dynamic'],
               options:{spawn:false}},
          sass: {
               files:['WebApp/res/**/*.scss'],
               tasks:['sass:dynamic],
               options:{spawn:false}}}});

// 对插件进行加载声明
grunt.loadNpmTasks('grunt-contrib-uglify');
grunt.loadNpmTasks('grunt-contrib-sass');
grunt.loadNpmTasks('grunt-contrib-concat');
grunt.loadNpmTasks('grunt-contrib-watch');

// 定义任务
grunt.registerTask('init',['uglify:minify','sass:comp
ile','watch']);
grunt.registerTask('combo',['concat']);

// 我们利用GruntAPI进行了扩展
// 检测到文件变更时 取出该文件路径
grunt.event.on('watch',
     function(action,filepath){
          grunt.log.
            writeln('---Found: '
               + filepath +
                  ' has changed, processing...');
          grunt.config('uglify.dynamic.src',filepath);
          grunt.config('sass.dynamic.src',filepath);
});
};
```

Grunt实例讲解 - 运行init任务

➢刚才，我们在Gruntfile.js中配置了两个Task：init和combo。

在命令行输入grunt init，执行init任务：

```
Fei@FEI-PC ~/Dropbox/Codes/workspace/EasyTools
$ grunt init
Running "uglify:processAll" (uglify) task
File "WebApp/res/common/a.min.js" created.
File "WebApp/res/common/common.min.js" created.
File "WebApp/res/common/test.min.js" created.

Running "sass:compile" (sass) task
File WebApp/res/theme/ui.css created.

Running "watch" task
Waiting...---Found: WebApp\res\common\common.js changed
OK
>> File "WebApp\res\common\common.js" changed.

Running "uglify:needed" (uglify) task
File "WebApp/res/common/common.min.js" created.

Running "watch" task
 - Waiting...
```

可以看到，任务启动首先将所有.js文件压缩，并将.scss文件编译。

随后在后台监听，如有.js或者.scss文件内容变更，将会自动对该文件进行压缩或编译。

Grunt实例讲解 - 运行combo任务

- 在命令行输入grunt combo，执行合并任务：

```
Fei@FEI-PC ~/Dropbox/Codes/workspace/EasyTools
$ grunt combo
Running "concat:target" (concat) task
File "WebApp/res/common/full.js" created.

Done, without errors.
```

运行combo任务后，Grunt自动将common文件夹下已压缩的.js文件合并到了full.js：

```
$ ls -lh WebApp/res/common/
total 10
-rw-r--r--    1 Fei      Administ       84 Apr  2 11:43 a.js
-rw-r--r--    1 Fei      Administ       83 Apr  2 11:08 test.js
-rw-r--r--    1 Fei      Administ     9.9k Apr  2 14:43 common.js
-rw-r--r--    1 Fei      Administ      252 Apr  2 14:43 a.min.js
-rw-r--r--    1 Fei      Administ      253 Apr  2 14:43 test.min.js
-rw-r--r--    1 Fei      Administ     2.4k Apr  2 14:43 common.min.js
-rw-r--r--    1 Fei      Administ     2.9k Apr  2 14:55 full.js
```

Grunt实例讲解 - 运行结果

- 经过以上两个任务，可以看出我们之前的需求都得到解决：

1. 自动转换theme/文件夹下的.scss文件为同名.css文件 ✓
2. 自动压缩common/文件夹中的.js文件到.min.js结尾的同名文件 ✓
3. 在需要的时候将已压缩的JavaScript文件合并到full.js中 ✓

这只是一个简单的例子，运用Grunt可实现非常多的功能。

Grunt的第三方插件

Grunt的魅力很大程度上来自其庞大的开源生态系统，目前发布在NPM上的Grunt插件已经超过2000个，且还在快速增加。同时，任何人都可以方便的发布自己的插件到NPM上供其他人使用。

常用插件：

grunt-contrib-sass：编译SASS代码编译为CSS。
grunt-contrib-ugligy： 压缩JS源文件。
grunt-contrib-concat：合并文件，减少HTTP请求。
grunt-contrib-imagemin：对PNG, JPEG和GIF等格式的图像进行批量压缩。
grunt-contrib-cssmin：压缩以及合并CSS文件。
grunt-contrib-handlebars: 预编译Handlebars模板文件。
grunt-contrib-watch: 监视文件变更，自动运行一系列指定任务，例如编译、压缩等。
grunt-contrib-jshint: 检查JavaScript代码质量，类似JSLint。

Grunt的API扩展

某些特殊情况下，插件无法满足我们使用需求时，可以通过自定义任务进行扩展。Grunt提供了许多API接口帮助实现特殊需求。

常用API：

grunt.config: 读取和管理Gruntfile.js中的配置信息。
grunt.event: 自定义事件触发。
grunt.log: Grunt自有的log功能。
grunt.task: 用于注册自定义任务和加载外部任务。
grunt.fail: 用于异常处理时发出警告或强制终止任务。
grunt.option: 用于从命令行中读取参数。
grunt.file: 用于磁盘文件操作，例如read, write, copy, delete, mkdir 等。
grunt.template: 处理Gruntfile中的模板变量，以及提供了常用日期模板辅助函数。
grunt.util: 各种工具函数，以及集成了各种外部库，包括Lo-Dash，Async，Hook等。

Grunt的优缺点

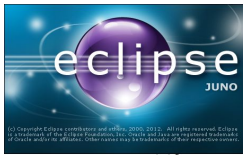
优点：

1. 简化工作流程，一劳永逸的实现前端的自动化构建
2. 符合前端发展趋势，推动其他新技术的应用
3. 配置文件灵活，可利用第三方插件，也可以自行扩展

缺点：

1. 依赖NodeJS环境，在Windows上体验稍差
2. Eclipse等IDE暂无直接支持，需手动在命令行运行

这是新的开发流程？没错!



启动Grunt　　编码　　调试　　上线

Just code with Grunt.

感谢聆听！

CMC CRM SRD New Business Dept - 齐飞

qifei3@asiainfo-linkage.com