GRUNT

jsCafe vol.13

Grunt

Grunt は NodeJS で動く自動化ツールです。
プラグインとして配布されているタスクを
インストールする事でさまざまな作業を自動化す
る事が出来ます。
SASS、CoffeeScript、ファイル操作
画像操作、ファイル監視、などをファイル形式に
とらわれず、まとめて自動化できる事が最大の利点
です。

今日お話する事

- First Step
  - Grunt Install
  - タスク設定
  - タスク実行パターン
- Next Step
  - Grunt API
- grunt-init
  - grunt-init install
  - grunt-init を使う
- Tasks
  - grunt-contrib-*
  - その他のプラグイン
- Grunt を使ってみて

First Step

Grunt Install

Grunt は grunt-cli と grunt の本体両方が必要

The Grunt command line interface.

```
$ npm install -g grunt-cli
```

タスク設定

1. npm パッケージ定義ファイルの準備

2. Grunt 本体とタスクをインストール

3. タスクを管理する定義ファイルの準備

タスク設定

1. npm パッケージ定義ファイルの準備

package.json

2. Grunt 本体とタスクをインストール

3. タスクを管理する定義ファイルの準備

タスク設定

1. npm パッケージ定義ファイルの準備

package.json

2. Grunt 本体とタスクをインストール

npm install

3. タスクを管理する定義ファイルの準備

タスク設定

1. npm パッケージ定義ファイルの準備

package.json

2. Grunt 本体とタスクをインストール

npm install

3. タスクを管理する定義ファイルの準備

Gruntfile.js

1. npm パッケージ定義ファイルの準備

package.json を用意します。

```
{
  "name": "My-Project-Name",
  "version": "0.0.0",
  "description": "jsCafe GruntJs Startup."
}
```

Grunt を使う上でほぼ必須。
とりあえず使うなら **name**、**version**、**description** の
3つくらいあれば良いと思います。

2. Grunt 本体とタスクをインストール

```
# Grunt 本体 のインストール
$ npm install grunt --save-dev [-D]

# Grunt Plugin のインストール
$ npm install <PackageName> --save-dev [-D]
```

`--save-dev` もしくは `-D` オプションでインストールし
package.json に依存ファイルとして一緒に定義します。

3. タスクを管理する Gruntfile.js の準備

Gruntfile.js もしくは Gruntfile.coffee を用意します。
これも 3 Step で編集します。

1. grunt.**initConfig**(<Configs>)
   タスク毎の設定。

2. grunt.**loadNpmTasks**(<GruntTackName>)
   プラグインの読み込み。

3. grunt.**registerTask**(<TaskName>, [<Task>])
   タスクの順番を定義

Gruntfile.js - 1 / 2

- **親タスク**：Gruntプラグインを指定する。
- **子タスク**：小分けにタスクを設定できる。

```
module.exports = function(grunt){
    grunt.initConfig({
        concat:{
            dist:{
                src: ['foo.js' , 'bar.js'],
                dest: 'main.js'
            }
        }
    });
    // ... 続く
```

Gruntfile.js - 2 / 2

```
  // ... 続き...

  // タスクプラグインを読み込む
  grunt.loadNpmTasks('grunt-contrib-concat');

  // Default のタスクを定義します
  grunt.registerTask('default', ['concat']);

  // 他にもタスクは定義できる。
  grunt.registerTask('foo', ['concat:dist']);
};
```

タスク実行パターン

タスク実行の 3 パターン指定方法がある。

```
# default タスク
$ grunt

# concat タスク (親タスク)
$ grunt concat

# concat => dist タスク (子タスク)
$ grunt concat:dist
```

Next Step

Grunt API

Grunt API を利用すると
テンプレートエンジン (include Lo-Dash)や
JSON, YAML などのデータ・フォーマット
などもGruntのみで扱える。
他にもタスクに別名をつけるとか便利なAPIがある

http://gruntjs.com/api/grunt

Grunt API で動的バナーを追加する - 1 / 3

```
grunt.initConfig({

    // pkg プロパティにpackage.jsonの内容をインポートする
    pkg: grunt.file.readJSON('package.json'),

    // banner プロパティにテンプレートを埋め込む
    banner: '/**\n'+
    '* Name: <%= pkg.name %>\n'+
    '* Version: <%= pkg.version %>\n'+
    '* Description: <%= pkg.description %>\n'+
    '* Date: <%= grunt.template.today("yyyy-mm-dd") %>\n'+
    '*/\n',

    // 続く...
```

Grunt API で動的バナーを追加する - 2 / 3

```
// ... 続き ...

concat: {
    options: {
        // banner プロパティの読み込み
        banner: '<%= banner %>\n'
    },
    dist: {
        src: ['foo.js', 'bar.js'],
        dest: 'main.js'
    }
}
});
```

Grunt API で動的バナーを追加する - 3 / 3

タスクが実行される度に日付やバージョンなど更新される。

```
# 出力結果
/**
* Name: My-Project-Name
* Version: 0.0.0
* Description: jsCafe Gruntjs Startup.
* Date: 2013-08-25
*/

// foo.js

// bar.js
```

Grunt API

Config のオブジェクトに値を渡す事が出来て
テンプレートを使って値を使える。

package.json や他言語の コンフィグファイル
compass の config.yaml などの設定データも
Gruntfile に使える。
そして、Gruntfile 自体も再利用できる。

Watch

以前、grunt-contrib-livereload を紹介しているブログなどがありました。

が、LiveReload 機能が grunt-contrib-watch に統合され、grunt-contrib-livereload は非推奨になりました。
では基本的な watch 設定方法を見てみましょう。

grunt-contrib-watch - 1 / 2

ポイントは子タスクを指定するところです。

```
less: {
   dist: {
      files: { "css/*.css": "css/*.less" }
   }
},
watch: {
   options: {
      livereload: true // リロード有効になります

   },
   less_files: {
      files: 'css/*.less',
      tasks: ['less:dist'] //lessの子タスクを指定

   }
```

grunt-contrib-watch - 2 / 2

Gruntfileの設定が終われば

ブラウザ用の LiveReload プラグイン

もしくは、

HTMLにスクリプトを埋め込みます。

```
<script src="http://localhost:35729/livereload.js"></script>
```

grunt-init

Scaffolding

grunt-init Install

Gruntプロジェクトのひな形 ユーティリティ

Gruntfile や Grunt プラグインのひな形を管理

http://gruntjs.com/project-scaffolding

```
# grunt-init
$ npm install -g grunt-init
```

grunt-init を使う

grunt-init で使うテンプレートをgitからインストールします

- **grunt-init-gruntfile** : Gruntfile の基本的なひな形
- **grunt-init-gruntplugin** : Grunt Plugin を作るひな形
- **grunt-init-jquery** : jQuery Plugin のひな形

テンプレートは `~/.grunt-init` ディレクリから読み込む

他にも、指定した Path からも読み込む事が出来ます。

grunt-init を使う

```
# git clone でテンプレートをインストール
$ git clone git@github.com:gruntjs/grunt-init-jquery.git ~/.grunt-init/jquery

# テンプレート名から作成
$ grunt-init jquery
$ grunt-init <TemplateName>

# 任意のPath を指定して作成
$ grunt-init <Path>
```

Tasks

Grunt タスクあれこれ

grunt-contrib-*

Grunt 謹製 タスク

https://github.com/gruntjs/grunt-contrib

ファイルユーティリティ

- **grunt-contrib-clean**
  指定した不要なファイル・ディレクトリの削除
  ※ キャッシュファイルの削除とか。
- **grunt-contrib-compress**
  指定したディレクトリなどを圧縮ファイルとしてアーカイブする
- **grunt-contrib-concat**
  ファイルを結合
- **grunt-contrib-copy**
  ファイル・ディレクトリのコピー
  ※ altJSからコンパイルした後に指定ディレクトリにコピーするとか

JavaScript ユーティリティ、altJS コンパイラ

- **grunt-contrib-coffee**
  CoffeeScript ファイルの結合・コンパイル
- **grunt-contrib-jshint**
  JSHint
- **grunt-contrib-requirejs**
  RequireJS プロジェクトのファイル群を r.js で最適化してくれます。
  ※ 1ファイルにまとめたり、uglify とか
- **grunt-contrib-uglify**
  minify してついでに多少難読化できます。
- **grunt-contrib-yuidoc**
  YUIDoc で JavaScript のドキュメントを自動生成します。

スタイルシートユーティリティ、プリプロセッサ

- **grunt-contrib-compass**
  SASSのフレームワーク Compass
- **grunt-contrib-cssmin**
  CSS ファイルを minify したり、gzip化 します。
- **grunt-contrib-csslint**
  CSSLint
- **grunt-contrib-less**
  LESS のコンパイル
- **grunt-contrib-sass**
  SASS のコンパイル
- **grunt-contrib-stylus**
  Stylus のコンパイル

HTMLユーティリティ、テンプレートエンジン

- **grunt-contrib-htmlmin**
  Minify HTML
- **grunt-contrib-jade**
  Compile Jade files to HTML.
- **grunt-contrib-handlebars**
  Precompile Handlebars templates to JST file.
  プリコンパイルファイルの作成
- **grunt-contrib-jst**
  Precompile Underscore templates to JST file.
  プリコンパイルファイルの作成

テストフレームワーク

- **grunt-contrib-nodeunit**
  Nodeunit を実行してくれる。
- **grunt-contrib-jasmine**
  PhantomJS を対象とした jasmine テストを実行してくれる。
- **grunt-contrib-qunit**
  PhantomJS を対象とした QUnit のブラウザテストを実行してくれる。

その他

- **grunt-contrib-imagemin**
  PNG や JPEG の画像を、OptiPNG, pngquant, jpegtran などで最適化します。

- **grunt-contrib-connect**
  > Node パッケージ connect で Webサーバー を実行します。

- **grunt-contrib-watch**
  指定したファイルの変更を監視して、タスク実行を自動化します。
  LiveReload 機能も grunt-contrib-watch のオプションに統合されました。それに従い、grunt-contrib-livereload は **deprecated** (非推奨) になりました。

More Tasks

Grunt Plugins. - 1 / 3

- **grunt-typescript**
  TypeScript のコンパイル

- **grunt-haxe**
  Haxe から JavaScript へのコンパイル

- **grunt-karma**
  テストフレームワーク karma の実行

- **grunt-csso**
  CSSO CSS Optimizer タスク

- **grunt-csscomb**
  CSSComb CSSファイルの プロパティ順などを正規化してくれたり色々

Grunt Plugins. - 2 / 3

- **grunt-imageoptim**
  非常に便利な Macアプリ ImageOptim と ImageAlpha のなどの画像最適化 タスク

- **grunt-data-uri**
  CSSの中の画像を dataURI に変更して embed するタスク

- **grunt-jekyll**
  Ruby 製の Static サイト、Blog ジェネレータ jekyll ビルド タスク

- **grunt-shell**
  シェルコマンド実行

- **grunt-include-replace**
  HTMLをインクルードしたり、パラメータ埋め込んだり

Grunt Plugins. - 3 / 3

- **grunt-connect-proxy**
  grunt-contrib-connect をベースに、proxy（ http-proxy ）機能を追加。

- **grunt-express-server**
  Express サーバーをタスクとして立ち上げる。
  grunt-express-watch と合わせて使い、app.js などに変更があった時にサーバーを再起動できる。

- **grunt-text-replace**
  テキストファイル中を正規表現置換

- **grunt-open**
  指定したURLをブラウザで開く

More...

Grunt を使ってみて...

静的サイト とか

Webサイトで大量の静的ファイルをさばく事が多いですが
jekyll などのHTMLジェネレーター + Grunt + MAMP で
快適環境出来ます。

良く使っている Grunt Plugins

- grunt-jekyll
- grunt-contrib-coffee
- grunt-contrib-copy
- grunt-contrib-compass
- grunt-contrib-watch
- grunt-contrib-concat

Grunt タスクが遅い

最初はファイル数が少ないのでタスク終了まで時間はかからないが、ファイル数が多く実行されるタスクの種類も多くなると割ともたつく。

解析系やテスト系、コンパイル系のタスクは Grunt が呼び出してから起動するものが多いので起動時間を減らす。

結論、子タスク は沢山あった方がいい。

Grunt タスクを使いこなすポイント

子タスクを種類別に分ける。

ライブラリ系の全体で使われる様な開発時は頻繁に改変されるCSS、JS、画像などの子タスク

ページ毎に必要なCSS、JS、画像などの子タスク

この2種類を分けるだけで Watch した時のパフォーマンスは大幅に変わる

タスクは分散しましょう

Lessファイルの種類に応じてビルドを分ける

```
less: {
   libs: { /* configs */ },
   pages: { /* configs */ }
},
watch: {
   lib_files: {
      tasks: ['less:libs', ...]
   },
   page_files: {
      tasks: ['less:pages', ...]
   }
}
```

他にも

HTML や CSS や JavaScript や 画像 にかぎらず
RubyでもPHPでも他の言語ファイルでも
Watch でまとめれば一石二鳥。

PHPUnit を実行させるとか
PhantomJS でサイト巡回させるとか
Vagrant のスターターとか

Grunt は

ルーティング可能なタスクマネージャです。

面倒でも計画的に利用しましょう。

おわり

@sakunyo