

nando vieira[Go to English Blog](#)

09 de Dezembro de 2013

Usando Grunt.js

Leia em 5 minutos

O Node tornou possível a criação de uma série de ferramentas. Este é o caso do Grunt, que permite automatizar tarefas no terminal, como concatenar arquivos de JavaScript e CSS, executar o JSHint, dentre muitas outras coisas.



A grande vantagem de se usar o Grunt.js em vez de outras alternativas é a grande quantidade de plugins disponíveis. A lista de plugins oficiais é bastante grande, mas também existem centenas de plugins criados pela comunidade.

Neste artigo você verá como o Grunt funciona e como configurar o seu projeto com as mais diversas tarefas.

Instalando

Para instalar o Grunt você vai precisar instalar o pacote `grunt-cli` através do NPM (Node Package Manager). Ele é apenas uma interface sobre os plugins disponíveis em seu projeto.

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

Isso irá disponibilizar o comando `grunt` em seu `$PATH`. Toda vez que você executa este comando, o Grunt irá procurar as tarefas disponíveis e executará a tarefa que você pediu.

No entanto, ainda temos algumas coisas para fazer. Para que o Grunt funcione, é preciso criar o arquivo de configuração do seu projeto, que definirá as dependências do NPM. É o arquivo `package.json`.

JAVASCRIPT

```
{  
  "name": "simplesideias",  
  "version": "0.0.0",  
  "private": true  
}
```

Se você não está criando um projeto que será disponibilizado publicamente, adicione a opção `private`; isso garantirá que você não publique o seu projeto no NPM por engano.

Agora, toda a parte de configuração das tarefas é feita em um outro arquivo chamado `Gruntfile.js`.

Criando o arquivo Gruntfile.js

O arquivo `Gruntfile.js` define como suas tarefas irão se comportar. Para isso, você deve instalar os plugins que precisa. Isso vai depender muito do seu objetivo, mas existem plugins para quase tudo o que você precisar fazer.

Vamos começar com um plugin oficial de concatenação de arquivos, mantido pelo time do Grunt. Da raíz do seu projeto, execute o comando abaixo.

```
$ npm install grunt-contrib-concat --save-dev
```

Isso irá instalar o plugin `grunt-contrib-concat`, além de atualizar o arquivo `package.json` com esta dependência.

JAVASCRIPT

```
{  
  "name": "grunt-samples",  
  "version": "0.0.0",  
  "private": true,  
  "devDependencies": {  
    "grunt": "~0.4.2",  
    "grunt-contrib-concat": "~0.3.0"  
  }  
}
```

Para ver uma lista completa dos plugins disponíveis, [acesse o site do projeto](#).

Agora você vai precisar configurar o arquivo `Gruntfile.js`. A

configuração segue uma estrutura mais ou menos comum, mas vai depender muito dos plugins instalados.

JAVASCRIPT

```
module.exports = function(grunt) {  
  grunt.loadNpmTasks("grunt-contrib-concat");  
  
  grunt.initConfig({  
    concat: {  
      dist: {  
        src: ["src/a.js", "src/b.js"]  
        , dest: "dist/app.js"  
      }  
    }  
  });  
};
```

A primeira coisa que você deve notar é que as tarefas, mesmo sendo instaladas e listadas no arquivo `package.json`, precisam ser carregadas no arquivo `Gruntfile.js`. Quando você possui muitas tarefas isso pode se tornar um pouco chato.

A chamada `grunt.initConfig(config)` é responsável por definir as configurações de cada tarefa. Esse objeto de configuração pode se tornar muito complexo, mas veremos algumas alternativas para facilitar a manutenção mais à frente.

Quase todas as tarefas trabalham com arquivos existentes, onde serão executadas tarefas específicas. No nosso caso, criamos um namespace `concat.dist`, que define a concatenação dos arquivos `a.js` e `b.js`, nesta ordem. O arquivo final será gerado em `dist/app.js`.

Para executar esta tarefa, execute o comando `grunt concat`.

```
$ grunt concat
Running "concat:dist" (concat) task
File "dist/app.js" created.
```

Done, without errors.

Você também pode executar um namespace específico; basta definir passar o nome da tarefa como em `grunt concat:dist`.

O Grunt permite que você use diferentes tipos de sintaxe na definição dos caminhos. Por isso, é importante entender quais são elas.

Definindo os caminhos

Quase todas as tarefas trabalham com input (arquivos que serão processados) e output (saída gerada pela tarefa). Para definir estes caminhos, o Grunt permite que você defina algumas sintaxes diferentes.

O primeiro modo, mostrado no exemplo acima, define o input e output com as chaves `src` e `dest`. Você pode passar uma string ou array como input.

JAVASCRIPT

```
grunt.initConfig({
  concat: {
    dist: {
      src: "src/*.js"
```

```
        , dest: "dist/full.js"
      }
    }
  });

  grunt.initConfig({
    concat: {
      dist: {
        src: ["src/a.js", "b.js"]
        , dest: "dist/full.js"
      }
    }
  });
```

É possível ignorar arquivos com uma exclamação no início do caminho, como em `!src/c.js`.

JAVASCRIPT

```
grunt.initConfig({
  concat: {
    dist: {
      src: ["src/*.js", "!src/c.js"]
      , dest: "dist/full.js"
    }
  }
});
```

Também é possível usar objetos. Neste caso, a chave/propriedade irá definir o destino.

JAVASCRIPT

```
grunt.initConfig({
  concat: {
    files: {
      "dist/full.js": ["src/*.js", "!src/c.js"]
    }
  }
});
```

```
    }  
  });  
}
```

Um outro modo é passar um array de objetos.

JAVASCRIPT

```
grunt.initConfig({  
  concat: {  
    dist: {  
      files: [  
        {src: ["a/*.js"], dest: "dist/a.js", nonull: true}  
        , {src: ["b/*.js"], dest: "dist/b.js", filter: "isFile"  
      ]  
    }  
  }  
});
```



A sintaxe acima aceita muitas outras opções. Para saber mais sobre os caminhos, acesse a [documentação do projeto](#).

Registrando novas tarefas

Você pode registrar novas tarefas com a função `grunt.registerTask`. É muito comum, por exemplo, registrar a tarefa `default`, executada quando nenhum nome de tarefa é passado.

```
$ grunt  
Warning: Task "default" not found. Use --force to continue.  
  
Aborted due to warnings.
```

Para registrar a tarefa padrão, você pode especificar os nomes de outras tarefas. Este tipo de definição é chamado de *alias*.

JAVASCRIPT


```
grunt.registerTask("default", ["jshint", "qunit", "concat", "ug  
grunt.registerTask("dist", ["concat:dist", "uglify:dist"]);
```



Você também pode registrar a sua própria tarefa, separadas em dois tipos. A tarefa *básica* irá executar a função atribuída sem buscar configurações no objeto de configuração.

JAVASCRIPT

```
grunt.registerTask("hello", "Say hello to the given name", func  
    console.log("Hi there,", name || "strange");  
});
```



Perceba que esta tarefa espera um argumento opcional. Para executar esta tarefa sem passar o argumento, basta executar o comando `grunt hello`.

```
$ grunt hello  
Running "hello" task  
Hi there, strange  
  
Done, without errors.
```

Se quiser especificar o argumento, use o comando `grunt task: [arg1]:[arg2]:[argN]`; no nosso caso, pode ser algo como `grunt hello:John`.


```
$ grunt hello:John  
Running "hello:John" (hello) task  
Hi there, John
```

Done, without errors.

Você pode executar outras tarefas usando a função
`grunt.tasks.run`.

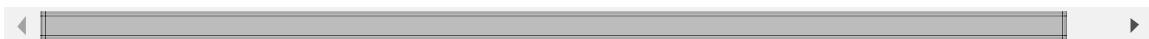
JAVASCRIPT

```
grunt.registerTask("build", function(name){  
  grunt.tasks.run(["concat", "uglify"]);  
});
```

Também é possível dar feedback ao usuário através do log.

JAVASCRIPT

```
grunt.log.writeln("Processing...");  
grunt.log.ok("Yay!");  
grunt.log.warn("Watch out!");  
grunt.log.error("WTF?");  
grunt.log.header("Result");  
grunt.log.subhead("Here is the response:");  
grunt.log.writeln(grunt.log.wordlist(["this", "this", "that"]))
```



```
$ grunt
Processing...
>> Yay!
>> Watch out!
>> WTF?
```

Result

Here is the response:
this, this, that

A *multitarefa* permite receber as configurações definidas no objeto de inicialização do Grunt.

JAVASCRIPT

```
var fs = require("fs");

module.exports = function(grunt) {
  grunt.registerMultiTask("symlink", function(){
    this.files.forEach(function(fileset){
      fileset.src.forEach(function(path){
        grunt.log.writeln("Symlinking " + path + " to " + fileset.dest);
      });
    });

    grunt.log.ok();
  });

  grunt.initConfig({
    symlink: {
      dist: {
        src: ["src/*.js", "dist/app.js", "Gruntfile.js"],
        dest: "symlinked/"
      }
    }
  });
};
```

```
};
```

Limpando seu Gruntfile.js

É muito comum na comunidade criar arquivos `Gruntfile.js` com o objeto de configuração extremamente grande, definindo na chamada da função `grunt.initConfig` e que usei como exemplo até agora. Acredito que este tipo de definição dificulta muito a manutenção do arquivo, tornando praticamente ilegível.

Ultimamente ando usando um plugin que criei para facilitar um pouco esse problema. Para instalá-lo, use o comando `npm install grunt-settings --save-dev`.

A primeira vantagem de usar este plugin é que todas as tarefas começando com `grunt-` serão carregadas automaticamente. Você não precisa mais chamar a função `grunt.loadNpmTasks` manualmente.

Outra vantagem é que ficou simples definir as configurações por namespace; basta chamar a função `config.set` e o namespace será criado para você.

JAVASCRIPT

```
var config = require("grunt-settings");

module.exports = function(grunt) {
  // Initialize the configuration block.
  config.init(grunt);

  // Set the concat configuration.
```

```
// The target's name is going to be "dist".
config.set("concat.dist", {
  src: "src/*.js"
  , dest: "dist/full.js"
});

// Set the uglify configuration.
config.set("uglify.dist", {
  options: {report: "gzip"}
  , src: "dist/full.js"
  , dest: "dist/full.min.js"
});

// Register the default task.
config.registerTask("default", ["concat", "uglify"]);
};
```

Se você precisar registrar tarefas personalizadas, pode continuar usando a função `grunt.registerTask` e `grunt.registerMultiTask`.

Por fim, caso sua configuração ainda continue muito grande, separe as configurações por arquivos.

Finalizando

Para quem usa o Rails e possui o Asset Pipeline, usar o Grunt pode não ser tão útil assim (embora você possa configurar coisas como JSHint e qualidade do código), mas isso não é tão verdade em outras linguagens/frameworks. Para esses casos, o Grunt é uma boa alternativa e conta com uma muitos plugins disponíveis que, com certeza, irão facilitar a sua vida.

Compartilhe:**7 Comentários****Nando Vieira****1 Entrar** ▾

Recommend

Compartilhar

Ordenar por Mais antigo ▾



Participe da discussão...

**lucascaton** • 2 anos atrás

Pra quem é do mundo Ruby, uma boa forma de explicar o que é o Grunt, é pensando em um "rake" para Javascript :-)

Ótimo artigo Nando! Estou usando o GruntJS há um tempo e gosto bastante.

| • Responder • Compartilhar >

**Nando Vieira** autor ➔ lucascaton • 2 anos atrás

Eu não acho que Rake seja uma boa analogia; esse seria mais o caso do Jake. Mas concordo que você possa usar para o mesmo objetivo. ;)

| • Responder • Compartilhar >

**Palmer** □ • 2 anos atrás

Por mais que tenha já alguns artigos sobre Grunt, o Nando sempre vai mais a fundo. Ótimo artigo!

1 | • Responder • Compartilhar >

**Nando Vieira** autor ➔ Palmer □ • 2 anos atrás

Valeu! ;)

| • Responder • Compartilhar >

**rizidoro** • 2 anos atrás

Parabéns pelo artigo Nando, como sempre com muita qualidade. Venho utilizando Grunt há algum tempo em meus projetos, e quanto a carregar as Tasks, utilizo este pacote NPM <https://github.com/sindresorhu...>, é bem parecido com o seu também, exceto que ele não cria namespaces...

| • Responder • Compartilhar >

**Nando Vieira** autor ➔ rizidoro • 2 anos atrás

Eu usava esse plugin, mas decidi implementar algo ainda mais simples, sem precisar nem carregar essa biblioteca.

| • Responder • Compartilhar >



Michel Ribeiro • 2 anos atrás

Olá, sei que a explicação aqui não é de IDE e sim das Tasks criadas. Só que estou tendo um problema e gostaria de saber se alguém teve, ou ainda não tentou. Estou usando o WebStorm e criando o projeto pelo terminal do mesmo, adiciono ao meu projeto o grunt-contrib-jade para usar o .jade como linguagem (vira html no final p quem não sabe), só que não estou conseguindo fazer a configuração das tasks corretamente. Só para exemplificar: Coloco a config da Task no watch, na configuração depois do watch como padrao e faço as chamadas da task no serve e build. Dá erro no concat. (Meu arquivo q é gerado na criação do projeto é um index.html e tiro todo taqueamento e insiro do formato iade. E iá vi e

Textos escritos por Nando Vieira — exceto quando informado — e com algumas restrições de uso. Aqui está o RSS.