02

Versionamento

Pegando carona no cache do navegador

Vimos no capítulo sobre merge (concatenação) que reduzir o número de requisições melhora consideravelmente o carregamento de nossas páginas e por isso aprendemos a concatenar automaticamente nossos arquivos.

Outro ponto que pode ajudar a diminuir ainda mais essas requisições é fazer com que esses arquivos fiquem no cache do navegador durante um bom tempo. Isso é interessante, porque no lugar de baixá-los novamente, o navegador utilizará uma versão já baixada. Essas configurações da data de expiração dos arquivos no cache são feitas no lado do servidor.

Porém, como o navegador saberá que deve carregar um novo arquivo quando ele for alterado no servidor? Ele terá que esperar o cache expirar para então baixá-lo? Abrir a página de configuração de seu navegador e limpar o cache? É aí que entra a técnica de versionamento.

A técnica de versionamento

Queremos que nossos arquivos fiquem no cache quase que eternamente, mas queremos que arquivos alterados sejam baixados novamente pelo navegador. Uma maneira de fazermos isso é alterando o nome do arquivo criando uma nova versão. Por exemplo:

```
<img src="img/logo.png">
```

Renomeamos o arquivo para:

```
<img src="2babcead.logo.png">
```

Repare o prefixo do nome do arquivo. Poderia ter sido qualquer coisa como um número incremental, mas no exemplo acima usamos um hash de 8 dígitos calculado a partir do arquivo. Se por acaso ele mudar, basta calcularmos esse hash com alguma ferramenta alterando logo em seguida o nome do arquivo e sua referência em nossas páginas.

A pergunta que não quer calar: já imaginou fazer isso manualmente? Não, com certeza você deve ter pensando em fazer isso através do Grunt e é exatamente o que faremos.

Versionamento através do Grunt

Você já deve ter percebido que além de termos que adicionar automaticamente um hash de 8 dígitos do arquivo como prefixo precisaremos alterar nosso HTML para que aponte para o novo arquivo. Já temos a task 'usemin' que altera a importação de arquivos JavaScript e CSS de nossa aplicação. Ela é ainda mais esperta, e alterará o caminho dos arquivos que tiverem sua versão de revisão no disco. Fantástico!

Primeiro instalaremos o plugin grunt-rev (https://github.com/cbas/grunt-rev). Ele será responsável em renomear os arquivos adicionando seu hash como prefixo:

```
npm install grunt-rev --save-dev
```

Registrando o plugin em nosso GruntFile.js

```
grunt.loadNpmTasks('grunt-rev');
```

O nome da task é `rev`. Ela conterá dois targets: um que aponta para nossas imagens e outro para nossos scripts, porém apenas os minificados!

```
rev: {
      imagens: {
        src: ['dist/img/**/*.{png,jpg,gif}']
      },

      minificados: {
        src: ['dist/js/**/*.min.js', 'dist/css/**/*.min.css']
      }
}
```

Nossa task está quase pronta, ainda falta indicar qual estratégia de versionamento utilizaremos para gerar o hash e quantos dígitos ele utilizará. Fazemos isso adicionando um targets especial, que parece target mais não é. Quase todos os plugins do grunt aceitam o parâmetro 'options':

```
rev: {
      options: {
        encoding: 'utf8',
        algorithm: 'md5',
        length: 8
      },

      imagens: {
        src: ['dist/img/**/*.{png,jpg,gif}']
      },

      minificados: {
        src: ['dist/js/**/*.min.js', 'dist/css/**/*.min.css']
      }
}
```

A configuração `options` recebe um objeto com três parâmetros: o encoding do arquivo, o algoritmo utilizado e o tamanho do prefixo gerado. Usaremos UTF-8, MD5 e o tamanho 8 respectivamente.

Por fim, precisamos rodar as dois targets antes da task 'usemin':

```
grunt.registerTask('minifica', ['useminPrepare',
                                'concat', 'uglify', 'cssmin', 'rev:imagens','rev:minificados'
```

Nosso script final ficará assim:

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

```
 grunt.initConfig({
    /* Copia os arquivos para o diretório 'dist' */
    copy: {
       public: {
         expand: true,
         cwd: 'public',
         src: '**',
         dest: 'dist'
       }
    },

    clean: {
         dist: {
             src: 'dist'
         }
    },

    useminPrepare: {
      html: 'dist/**/*.html'
    },

    usemin: {
      html: 'dist/**/*.html'
    },

    imagemin: {
     public: {
       expand: true,
       cwd: 'dist/img',
       src: '**/*.{png,jpg,gif}',
       dest: 'dist/img'
     }
   },

   rev: {
     options: {
       encoding: 'utf8',
       algorithm: 'md5',
       length: 8
     },

     imagens: {
       src: ['dist/img/**/*.{png,jpg,gif}']
     },
     minificados: {
       src: ['dist/js/**/*.min.js', 'dist/css/**/*.min.css']
     }
   }

});

//registrando task para minificação

grunt.registerTask('dist', ['clean', 'copy']);

grunt.registerTask('minifica', ['useminPrepare',
                                'concat', 'uglify', 'cssmin', 'rev:imagens','rev:minificados'
```

```
  // registrando tasks
  grunt.registerTask('default', ['dist', 'minifica', ]);

  // carregando tasks
  grunt.loadNpmTasks('grunt-contrib-copy');
  grunt.loadNpmTasks('grunt-contrib-clean');
  grunt.loadNpmTasks('grunt-contrib-concat');
  grunt.loadNpmTasks('grunt-contrib-uglify');
  grunt.loadNpmTasks('grunt-contrib-cssmin');
  grunt.loadNpmTasks('grunt-usemin');
  grunt.loadNpmTasks('grunt-contrib-imagemin');
  grunt.loadNpmTasks('grunt-rev');
}
```

Rodando a tasks padrão:

```
grunt
```

Se abrirmos o arquivo dist/index.html veremos que as tag's de recurso passaram a apontar para o arquivo da revisão encontrado em disco:

```
<link rel="stylesheet" href="css/ebb6096a.index.min.css"/>
...
<img src="img/2babcead.grunt-banner.png">
...
<script src="js/3ee7db69.index.min.js"></script>
```

Se você rodar novamente o script, os arquivo continuaram com o mesmo nome e só mudarão caso o arquivo original tenha sido modificado.