02

Minificação e concatenação

Minificação e concatenação

Toda vez que solicitamos uma página, além da requisição feita para obtê-la de nosso servidor, realizamos outras para baixar recursos externos como css's, scripts e imagens.

Sendo assim, nossa página index.html realizará um total de 5 requisições:

```
index.html <!-- uma requisição -->
...
<link rel="stylesheet" href="css/base.css"> <!-- uma requisição -->
<link rel="stylesheet" href="css/index.css"> <!-- uma requisição -->
...
<img src="img/grunt-banner.png"> <!-- uma requisição -->
...
<script src="js/index.js"></script> <!-- uma requisição -->
```

Imagine uma página mais complexa, em que o número de recursos externos seja muito maior, podendo chegar a trinta requisições.

O problema das múltiplas requisições

Mas qual o problema de realizarmos várias requisições? Cada navegador possui um número máximo de requisições simultâneas para um mesmo domínio, aliás, bem reduzido em smartphones. Se chegarmos a este limite, o próximo recurso a ser baixado terá que esperar até que uma das requisições acabe.

Outro ponto é a latência, que geralmente afeta bastante redes de dispositivos móveis. A latência é o tempo entre você realizar uma requisição ao servidor e ela começar a ser respondida.

Não podemos contar que nossos usuários tenham a rede com a menor latência do mundo ou um navegador que faça zilhões de requisições simultâneas para um mesmo domínio. É por isso que precisamos otimizar nossas páginas, diminuindo sempre que possível o número de requisições.

A técnica merge (concatenação)

Para reduzir o número de requisições feitas pelo navegador, podemos juntar arquivos de um mesmo tipo num só arquivo. Na nossa página, por exemplo, poderíamos juntar o base.css e o index.css. Na hora de o navegador interpretar o CSS, não importa se são dois arquivos ou um só.

Essa técnica de juntar arquivos de tipos semelhantes em um único arquivo é chamada de **merge** ou **concatenação**.

Vamos lembrar da nossa página index.html:

```
<link rel="stylesheet" href="css/base.css">
<link rel="stylesheet" href="css/index.css">

<script src="js/index.js"></script>
```

Juntando index.css e base.css pouparemos uma requisição. Pode parecer pouco, mas caso nossa página tivesse 10 css's evitaríamos de realizar 9 requisições, o que com certeza aceleraria bastante o carregamento da página.

O problema com o tamanho dos arquivos

Outra questão é a largura de banda. Cada recurso externo tem um peso e, quanto maior a largura de banda, mais bytes enviaremos e receberemos por segundo. Nem todos possuem Internet rápida e ainda há redes móveis que, depois de uma quantidade de bytes transferidos, reduzem sua largura de banda.

A técnica de minificação

Uma das técnicas de otimização que visam atacar o problema da largura de banda é a minificação. O processo de minificação consiste em diminuir o tamanho de arquivos css e JavaScript removendo espaços, pulo de linha, colocando todo arquivo inline, inclusive trocando o nome de parâmetros, funções e variáveis para nomes menores, diminuindo assim o arquivo final. Veja uma exemplo:

```
function exibe(frase) {
    alert(frase);
}
exibe();
```

O arquivo minificado ficaria:

```
function y(a){alert(a)}y();
```

Podemos ter uma redução de 10 a 15 por cento no tamanho do arquivo. Inclusive podemos fazer isso em nossos arquivos já concatenados.

O problema da não-automação

O problema é que tanto a concatenação como a minificação tornam a manutenção de nosso código algo complicado e até mesmo impossível. É por isso que ambos devem ser feitos apenas na versão de distribuição, e não nos arquivos originais do projeto.

Tudo bem, mas imagina realizar todo esse processo nos arquivos de distribuição toda vez que os originais mudarem? Impraticável, não? É por isso que essa tarefa deve ser automatizada e aprenderemos a fazer isso através do Grunt.

Automatizando merge e minificação com Grunt

Minificação e merge de arquivos `.js` e `.css` com certeza é uma das tarefas mais executadas no Grunt. Porém, não são tarefas que estão prontas por padrão. Precisamos instalar alguns *plugins*.

Os plugins envolvidos nestas tarefas são o **grunt-contrib-concat**, o **grunt-contrib-uglify** e o **grunt-contrib-cssmin**. Como todo plugin, precisamos instalar cada um deles através do `npm` e carregar suas tasks no Gruntfile.js com a função grunt.loadNpmTasks:

No terminal:

```
npm install grunt-contrib-concat --save-dev
npm install grunt-contrib-uglify --save-dev
npm install grunt-contrib-cssmin --save-dev
```

Em **Gruntfile.js**:

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

O problema é que seremos responsáveis pela configuração das tasks de cada um deles. Para resolver isso, utilizaremos o **grunt-usemin** para gerar as configurações dos plugins citados.

grunt-usemin

O **grunt-usemin** é um plugin que **facilita incrivelmente** a configuração das tasks envolvidas no processo de merge e minificação. Só precisamos configurar sua task para que ela gere automaticamente os parâmetros de configuração para os três plugins que vimos anteriormente.

Para instalá-lo, basta executar no terminal o comando:

```
npm install grunt-usemin --save-dev
```

O `grunt-usemin` funciona da seguinte maneira. Em nossas páginas, envolvemos os blocos de css e de script que desejamos realizar o merge e minificação utilizando um comentário especial. Por exemplo, em nossa página `index.html`:

```
<!-- build:css css/index.min.css -->
<link rel="stylesheet" href="css/base.css">
<link rel="stylesheet" href="css/index.css">
<!-- endbuild -->

<!-- build:js js/index.min.js -->
<script src="js/index.js"></script>
<!-- endbuild -->
```

Repare que usamos `build:css` e `build:js` repectivamente para merge e minificação de CSS e de JavaScript. Cada um deles recebe o caminho e o nome do arquivo resultante relativo ao arquivo HTML. Isso é importante porque, no final, além da criação dos arquivos, o HTML também será modificado para:

```
<!-- os dois arquivos foram concatenados e minificados -->
<link rel="stylesheet" href="css/index.min.css">
...
<!-- o arquivo foi minificado -->
<script src="js/index.min.js"></script>
```

Para que esta mágica aconteça, é necessário registrar o `grunt-usemin` no Gruntfile.js e configurar duas tasks distintas.

A primeira **useminPrepare** gerará configurações dinâmicas para `grunt-contrib-concat`, `grunt-contrib-uglify`, `grunt-contrib-cssmin`. O primeiro plugin concatenará os arquivos e os dois últimos "minificarão" JavaScript e CSS respectivamente.

A segunda task **usemin** alterará nossos arquivos HTML fazendo com que eles apontem para os arquivos concatenados e minificados definidos nos comentários especiais, pois foram criados antes pela task useminPrepare.

Não se preocupe, as duas tasks são tão simples quanto as que vimos anteriormente:

No Gruntfile.js:

```
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'
      }
    });


    //registrando task para minificação

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

    grunt.registerTask('minifica', ['useminPrepare',
                                    'concat', 'uglify', 'cssmin', 'usemin']);

    // 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');
}
```

Para testar executamos no terminal:

Perceba que registramos na task 'minifica' a chamada do `useminPrepare` para gerar as configurações das tasks subsequentes `concat`, `uglify` e `cssmin`. No final, a task `usemin` é chamada para modificar o HTML fazendo com que ele aponte para os arquivos criados.

Testando nossa task no terminal:

```
grunt
```

Nossa task default será executada aplicando a concatenação e minificação de arquivos na pasta 'dist'.