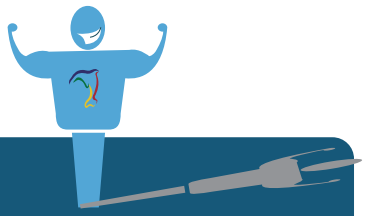
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

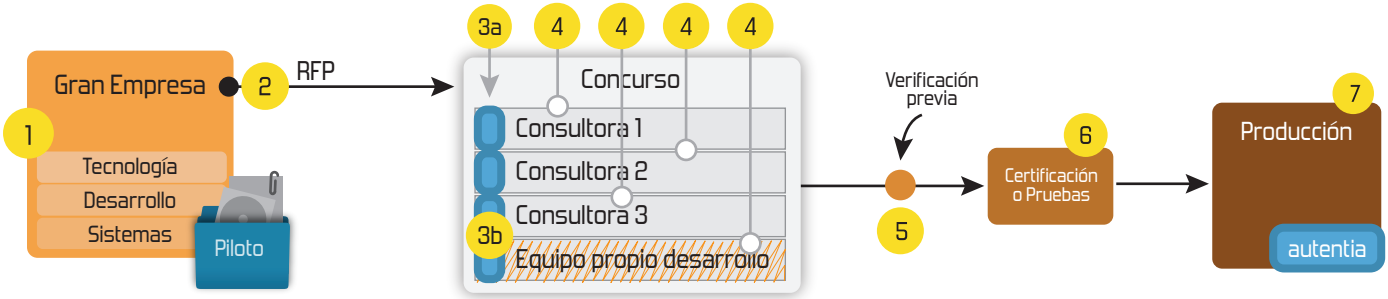
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

- Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery
- Gestor portales (Liferay)
- Gestor de contenidos (Alfresco)
- Aplicaciones híbridas
- Control de autenticación y acceso (Spring Security)
- UDDI
- Web Services
- Rest Services
- Social SSO
- SSO (Cas)
- JPA-Hibernate, MyBatis
- Motor de búsqueda empresarial (Solr)
- ETL (Talend)
- Dirección de Proyectos Informáticos.
- Metodologías ágiles
- Patrones de diseño
- TDD
- Tareas programadas (Quartz)
- Gestor documental (Alfresco)
- Inversión de control (Spring)
- BPM (jBPM o Bonita)
- Generación de informes (JasperReport)
- ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Daniel Diaz Suarez

Desarrollador Web en Autentia

Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/JEE

Ver todos los tutoriales del autor

Fecha de publicación del tutorial: 2013-09-26 Tutorial visitado 2 veces Descargar en PDF

Grunt, el TaskRunner de Javascript

0. Índice de contenidos.

1. Entorno

Este tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil MacBook Pro 17' (2.8 GHz Intel Core Duo, 8GB DDR3 SDRAM)
- Sistema Operativo: Mac OS X Lion 10.7.5

2. Introducción

Grunt es una herramienta que nos permite simplificar el proceso de construcción(build) de proyectos en Javascript, al igual que Maven nos permite automatizar una serie de pasos / procesos a la hora de compilar nuestro código Java, Grunt nos permite hacerlo con nuestro código Javascript.

3. ¿ Por qué usar Grunt ?

Hoy en día nuestras aplicaciones Javascript se componen de cada vez más ficheros, a medida que las bases de código también se van complicando, se están empezando a aplicar algunas de las buenas prácticas que ya se utilizaban en la parte servidora, como pueda ser TDD o maneras de organizar el código (el uso de clases / objetos, una clase por fichero.. etc).

El problema de estas técnicas es que es inviable usarlas directamente en un entorno de producción, el mandar 20 archivos Javascript al cliente es un proceso costoso y largo, ya que cada archivo supone una conexión HTTP, y con el crecimiento de las tecnologías móviles se busca minimizar las conexiones.

Para ello se usan técnicas como la concatenación de varios archivos Javascript en un único archivo optimizado, de manera que solo se le mandará un archivo al cliente, ocultando así también la disposición de los archivos en el entorno de desarrollo.

Para facilitar y automatizar estas tareas, existen herramientas como Grunt.

4. ¿ Que ventajas nos ofrece frente a Maven ?

Apache Maven es una herramienta diseñada para código Java, por lo que las necesidades que se tienen a la hora de construir un proyecto Java son bastante distintas a las necesarias a la hora de compilar código Javascript, Grunt es un Gestor de Tareas específico para Javascript, y nos permite realizar las tareas más comunes en nuestro código Javascript, como pueden ser el control de calidad(JSLint), la ofuscación, minimización y concatenación de archivos, también nos puede servir para automatizar el pase de una batería de test antes de realizar estos pasos.

5. Instalar Grunt

5.1. Requisitos:

Grunt usa el entorno de Node.js para funcionar, además del sistema de paquetes que proporciona Node (npm), por lo que es multiplataforma (siempre que la plataforma esté soportada por Node.js).

El primer paso para instalar Grunt es instalar Node.js, si no lo tenemos ya instalado en nuestro equipo, para ello basta

con irse a la página oficial de Node y descargar e instalar el ejecutable de nuestra plataforma.

Una vez hayamos instalado Node comprobamos que esté bien instalado escribiendo en la consola:

```
node --version
```

El siguiente paso será instalar Grunt en nuestro sistema, para ello usaremos el comando:

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

El "flag" -g indica a npm que queremos instalar Grunt de manera global, lo que nos permitirá invocarlo desde cualquier parte dentro de la línea de comandos.

Con esto ya tendríamos Grunt instalado en nuestro sistema, el siguiente paso será establecer las dependencias que tendrá nuestro proyecto y crear un GruntFile, parecido al pom.xml en Maven.

5.2. Declarando las dependencias:

Las dependencias de Grunt se declaran en el archivo package.json en nuestro proyecto, podemos añadir manualmente los paquetes que queramos añadir como dependencias a nuestro proyecto, o, cuando descarguemos nuevos paquetes, añadirlos automáticamente al fichero poniendo el flag --save-dev.

El primer paso es crear nuestro fichero package.json en la raíz de nuestro proyecto, podemos usar la siguiente plantilla:

```
{
  "name": "my-project-name",
  "version": "0.1.0",
  "devDependencies": {
    "grunt": "~0.4.1",
  }
}
```

De momento vamos a descargar dos paquetes de ejemplo para familiarizarnos con el uso, estos paquetes o plugins añaden funcionalidad a Grunt, una lista se puede encontrar aquí, los plugins a instalar son los siguientes:

- grunt-contrib-uglify : Este plugin nos permite ofuscar ligeramente nuestros archivos Javascript, de manera que nuestro código estará un poco mas protegido ante ojos ajenos, además reduce el tamaño de nuestros archivos Javascript, eliminando espacios innecesarios, y reduciendo el tamaño de los nombres de las variables.
- grunt-contrib-concat : Este plugin nos permitirá concatenar varios ficheros en uno, así podremos tener un fichero por clase por ejemplo limitando las peticiones HTTP.

Para instalar los plugin tan solo tenemos que escribir en la consola:

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

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

Si todo ha ido bien podremos ver como se habrán agregado las 3 dependencias a nuestro archivo Package.json, de manera que si otro compañero quiere instalar las dependencias de Grunt en nuestro proyecto tan solo tendría que usar el comando:

```
npm install
```

```
{
  "name": "my-project-name",
  "version": "0.1.0",
  "devDependencies": {
    "grunt": "~0.4.1",
    "grunt-contrib-jshint": "~0.6.3",
    "grunt-contrib-nodeunit": "~0.2.0",
    "grunt-contrib-uglify": "~0.2.2",
    "grunt-contrib-copy": "~0.4.1",
    "grunt-contrib-concat": "~0.3.0"
  }
}
```

6. Declarando nuestras tareas en el Gruntfile.js

El Gruntfile.js es para Grunt lo que es el pom.xml para Maven, aquí es donde declararemos todas las tareas que más adelante podremos invocar, a diferencia de Maven, Grunt nos ofrece mucha más flexibilidad lo que viene perfecto para un lenguaje tan dinámico como Javascript.

Los ficheros Gruntfile hacen uso de la notación JSON por lo que son fáciles de leer, con el siguiente código vamos a conseguir que cada vez que ejecutemos Grunt, todos los archivos Javascript que se encuentren en el fichero src (y las subcarpetas) se junten en un único archivo en la carpeta /js y luego se ofuscara ese archivo.

```
module.exports = function(grunt) {  // Toda la configuración se encontará dentro de es

  grunt.initConfig({    // Iniciamos la configuración de Grunt

    pkg: grunt.file.readJSON('package.json'),   // Leemos el archivo packageJSON, lo que

    concat: {   // Iniciamos la configuración del plugin concat

      options: {

        separator: ';'  // Este es el caracter que separará unos ficheros de otros cuand

      },
      dist: {

        src: ['src/**/*.js'],   // Este es un patrón para seleccionar los archivos que c
        dest: 'js/<%= pkg.name %>.js'   // Este es el archivo de salida y la carpeta dor

      }
    },
    uglify: {
```

Últimos Tutoriales del Autor

» Spring Container y la Inyección de Dependencias

» Hola Mundo con Spring 3 MVC

» Integración de Selenium Grid con Jenkins

» TDD, BDD & Test de aceptación

» Haciendo BDD con Cucumber

Últimas ofertas de empleo

2011-09-08
Comercial - Ventas - MADRID.

2011-09-03
Comercial - Ventas - VALENCIA.

2011-08-19
Comercial - Compras - ALICANTE.

2011-07-12
Otras Sin catalogar - MADRID.

2011-07-06
Otras Sin catalogar - LUGO.

```
      options: {

        banner: '/*! <%= pkg.name %> <%= grunt.template.today("dd-mm-yyyy") %> */\n'

      },
      dist: {

        files: {

          'js/<%= pkg.name %>.min.js': ['<%= concat.dist.dest %>']  // en este caso, co

        }
      }
    }
  });

  grunt.loadNpmTasks('grunt-contrib-uglify');   // Aquí tenemos que añadir los módulos
  grunt.loadNpmTasks('grunt-contrib-concat');

  grunt.registerTask('default', ['concat','uglify']); // Estas son las distintas task q
  grunt.registerTask('onlyConcat', ['concat']); // Este task se ejecutaría escribiendo

};
```

Con esto ya tendríamos lo básico para poder organizar nuestro proyecto sin las limitaciones que nos da el luego tener que distribuir estos archivos al cliente.

Ejecutando el comando grunt en el fichero del proyecto podemos ver como ejecuta el task por defecto:

```
MacBook-Pro-de-autentia:grunt_taskmanager_proyecto autentia$ grunt
Running "concat:dist" (concat) task
File "js/tutorial_gruntjs.js" created.

Running "uglify:dist" (uglify) task
File "js/tutorial_gruntjs.min.js" created.
```

7. Controlando la calidad del código Javascript con Grunt

Otra función interesante de Grunt sería la de usarlo para controlar la calidad del código Javascript de manera automática durante nuestro desarrollo, para ello podemos usar herramientas como pueden ser JSLint o JSHint que nos avisaran de posibles fallos en nuestro código.

También podemos usar Grunt para que corra nuestros test automáticos antes de hacer una build final usando Jasmine o Mocha usando los plugin correspondientes.

Un ejemplo de la configuración que usaríamos para los test sería:

```
jasmine : {
  src : 'src/**/*.js',          // El directorio donde se encuentran el código que tendrá
  options : {
    specs : 'spec/**/*.js'      // El directorio donde se encuentran los tests
  }
},
jshint: {
  all: [                        // Todos los archivos que pasaran a través del jsHint
    'Gruntfile.js',             // Pasaremos también el propio  Gruntfile
    'src/**/*.js',
    'spec/**/*.js'              // Y los archivos de test.
  ],
  options: {
    jshintrc: '.jshintrc'       // Indicamos donde se encuentra el archivo de opciones de
  }
}
```

Por ultimo solo tendríamos que registrar el Task para poder ejecutarla solo tendríamos que escribir Grunt test

```
grunt.registerTask('test', ['jshint', 'jasmine']);
```

También podríamos hacer una tarea que incluyese todo:

```
grunt.registerTask('build', ['jshint','jasmine','concat','uglify']);
```

8. Integrando Grunt con otras herramientas.

Grunt, al ser un programa de linea de comandos es fácilmente integrable en otras herramientas como pueden ser Jenkins o Maven, variando la complejidad de dicha integración, lo cual nos ofrece la posibilidad de controlar la calidad del código en el principal lenguaje de nuestra aplicación como también la calidad del código Javascript (contando que el código Javascript no es el principal en nuestra aplicación).

Por último, tambien tenemos la opción de integrar Grunt en nuestro IDE de manera que podamos pasar un Task de Grunt antes de hacer la build normal (si no usásemos Maven para ello)

9. Conclusiones

En los lenguajes dinámicos como Javascript debido a la falta de un compilador que nos avise de los posibles fallos, los test unitarios y las herramientas como JSHint cobran vital importancia a la hora de asegurar la calidad de nuestro código, herramientas como Grunt nos ayudan a realizar estas tareas y nos permiten automatizarlas, haciendo así mas fácil que no se pierdan las buenas costumbres a lo largo del ciclo de vida del proyecto.

El hecho de que Grunt esté escrito en Javascript nos otorga esa flexibilidad a la hora de manipular nuestros Task, lo cual le da una ventaja importante ante otras herramientas como Ant o Maven del mundo Java.

Links de Interes:

- Docs de Grunt
- Descarga Node
- Página de Plugins de Grunt

- Integrando Maven con Grunt

A continuación puedes evaluarlo:

Regístrate para evaluarlo

Por favor, vota +1 o compártelo si te pareció interesante

Share | 0

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

» Registrate y accede a esta y otras ventajas **«**