# Tema 6

Grunt y procesos automáticos

# ¿Qué es Grunt?

- Grunt en una palabra es **automatización**. Grunt permite automatizar tareas repetitivas de forma que el proceso de compilación y testeo sea rápido y automático.

- Es una herramienta muy utilizada en el desarrollo de aplicaciones web, ya que existen muchas tareas predefinidas y repetitivas que un desarrollador debe llevar a cabo para lograr optimizar el software y convertirlo en una pieza de calidad que siga los estándares de la web con el menor esfuerzo posible.

# ¿Qué es Grunt?

- Grunt puede ejecutar casi cualquier tarea que puedas definir de manera programada, incluso gran parte de las tareas comunes conforman ahora la comunidad de plugins de grunt.

- Entre ellos podrás encontrar minificadores y compresores de código, convertidores de un lenguaje a otro (como SASS a CSS), ejecución de pruebas, validadores, rutinas de conexión a servidores, instalación de dependencias especiales, y mucho más.

- Sin duda lo más común (o incluso lo menos esperado) ya está desarrollado por algún miembro de la comunidad; sin embargo si ninguno cumple con una necesidad particular que puedas tener siempre puedes desarrollar tu propia tarea y hasta la puedes publicar en npm para que los demás hagan uso de ella

# ¿Quién usa Grunt?


- Grunt es usado por importantes empresas que utilizan JavaScript, como pueden ser:
  - **jQuery**: jQuery, jQuery UI, QUnit
  - **Twitter**: Tweetdeck, Typeahead
  - **Mozilla**: Firefox Accounts projects
  - **Adobe**: Brackets, CSS FilterLab
  - **Wordpress**: WordPress Build Process, bbPress, BuddyPress
  - **Microsoft**: Visual Studio 2015 tiene soporte nativo para Grunt

# Instalación

- Para instalar Grunt, de forma global en nuestros proyectos, lo haremos con `npm` como estamos acostumbrados:

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

- Realmente esto no instala el ejecutor de tareas como tal (ya que éste se instala como un módulo más al incluirlo como una dependencia en el `package.json` del proyecto), sino una interfaz que permite ejecutar la versión correcta de Grunt dependiendo del proyecto en el que esté.

# Instalación

- Cada vez que `grunt` se lanza se comprueba si está instalado en el proyecto usando el sistema de node de `require()` es por esto por lo que se puede ejecutar grunt desde cualquier subcarpeta del proyecto.

- Si se encuentra localmente Grunt, CLI (Command Line Interface) carga la instalación de la librería de Grunt y añade la configuración al Gruntfile, y ejecuta la tarea que se ha intentado ejecutar.

# Instalación

- Una vez instalado grunt-cli para integrarlo en nuestro proyecto necesitamos crear dos ficheros:
  - **package.json**: Este es el fichero utilizado habitualmente por pm para guardar información de proyectos con módulos `npm`. Los plugins de Grunt deberán ir en `devDependencies` como veremos más adelante.
  - **Gruntfile**: Este fichero podrá ser Gruntfile.js o Gruntfile.coffe según el lenguaje en el que se escriba y se usa para configurar o definir tareas y cargar plugins de Grunt

# Instalación

- **Package.json:**
  - Está en el directorio raíz, en el mismo lugar que Gruntfile, debe ser confirmado si el proyecto utiliza control de versiones y contiene una lista de dependencias que se instalarán con el comando `npm install`
  - La mayoría de plantillas automáticamente crean un fichero package.json
  - El generador automático de Express genera ese mismo fichero también.

# Instalación

- **Package.json:**
  - Un ejemplo de package.json con grunt sería:

```
{
  "name": "my-project-name",
  "version": "0.1.0",
  "devDependencies": {
    "grunt": "~0.4.5",
    "grunt-contrib-jshint": "~0.10.0",
    "grunt-contrib-nodeunit": "~0.4.1",
    "grunt-contrib-uglify": "~0.5.0"
  }
}
```

# Instalación

- La forma más sencilla de añadir plugins de Grunt es a través de npm:

```
$ npm install <module> --save-dev
```

- Este comando no sólo instalará el módulo sino que lo guardará en nuestra lista de dependencias, en el apartado de dependencias para el desarrollo (devDependencies)

- Estos son algunos ejemplos:

```
npm install grunt –save-dev
npm install grunt-contrib-jshint --save-dev
```

# Gruntfile

- El fichero Gruntfile es un fichero JavaScript o CoffeeScript que está contenido en el directorio raíz del proyecto.

- Un fichero Gruntfile está compuesto de estas partes:
  - La función contenedora
  - Proyecto y configuraciones de las tareas
  - Carga de plugins y tareas
  - Tareas por defecto

# Gruntfile

- Un ejemplo de Gruntfile sería:

```
//Función contenedora
module.exports = function(grunt) {

  // Configuración del proyecto
  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json'),
    uglify: {
      options: {
        banner: '/*! <%= pkg.name %> <%= grunt.template.today("yyyy-mm-dd") %> */\n'
      },
      build: {
        src: 'src/<%= pkg.name %>.js',
        dest: 'build/<%= pkg.name %>.min.js'
      }
    }
  });

  // Carga el plugin "uglify".
  grunt.loadNpmTasks('grunt-contrib-uglify');

  // Tareas predeterminadas.
  grunt.registerTask('default', ['uglify']);

};
```

- Ahora veremos cada una de las partes por separado

# Gruntfile

- **La función contenedora:**
  - Es únicamente una función que se exporta como módulo y que debe contener todo lo relacionado con Grunt en su interior.
  - En el caso anterior se correspondería con esto:

```
module.exports = function(grunt) {
  // Todo lo relacionado con Grunt
};
```

# Gruntfile

- **Proyecto y configuración de tareas:**
  - La mayoría de las tareas Grunt utilizan los datos de configuración de definidos como un objeto que se obtiene con el método `grunt.initConfig`.
  - En este ejemplo `grunt.file.readjSON('package.json')` importa la información del `JSON` almacenada en `package.json` al fichero de configuración de Grunt.
  - La notación `<%  %>` hace referencia a propiedades de la configuración como rutas de directorios o lista de ficheros.
  - Se puede almacenar cualquier dato dentro del objeto de configuración siempre y cuando no cause conflictos con los datos de las tareas, en ese caso serán ignorados.

# Gruntfile

- **Proyecto y configuración de tareas:**
  - Como la mayoría de tareas, la tarea uglify del plugin `grunt-contrib-uglify` espera que su configuración esté especificada en una propiedad con el mismo nombre.
  - Dentro de el se especifica la opción `banner` y un nombre y destino para `build`:

```
grunt.initConfig({
  pkg: grunt.file.readJSON('package.json'),
  uglify: {
    options: {
      banner: '/*! <%= pkg.name %> <%= grunt.template.today("yyyy-mm-dd") %> */\n'
    },
    build: {
      src: 'src/<%= pkg.name %>.js',
      dest: 'build/<%= pkg.name %>.min.js'
    }
  }
});
```

# Gruntfile

- **Carga de plugins:**
  - Las tareas más comunes como concatenación o minificación están disponibles como plugins de grunt.
  - Al igual que un plugin está especificado en `package.json` como una dependencia y se instala con `pm install`, debe estar habilitado en el Gruntfile con este comando:

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

# Gruntfile

- **Tareas por defecto:**
  - Se pueden configurar una o varias tareas que se ejecutarán por defecto cuando se ejecute el comando grunt sin especificar una tarea concreta:
  - Si se ejecuta `grunt <plugin>` unicamente se ejecutará el plugin indicador, si se ejecuta `grunt` o `grunt default` se ejecutará uglify en este caso:

```
grunt.registerTask('default', ['uglify']);
```

# contrib-clean

- Se encarga de limpiar directorios y archivos.
- Se instala con: `npm install grunt-contrib-clean --save-dev`
- Se carga en Gruntfile con: `grunt.loadNpmTasks('grunt-contrib-clean');`
- Ejemplos:

```
clean: ["path/to/dir/one", "path/to/dir/two"]
clean: {
  build: ["path/to/dir/one", "path/to/dir/two"],
  release: ["path/to/another/dir/one", "path/to/another/dir/two"]
}
clean: {
  build: {
    src: ["path/to/dir/one", "path/to/dir/two"]
  }
}
```

# contrib-uglify

- Elimina espacios innecesarios reduciendo el tamaño de los archivos.
- Se instala con: `npm install grunt-contrib-uglify --save-dev`
- Se carga en Gruntfile con: `grunt.loadNpmTasks('grunt-contrib-uglify');`
- Contiene multiples opciones de compresión.
- Ejemplo:

```
grunt.initConfig({
  uglify: {
    my_target: {
      files: {
        'dest/output.min.js': ['src/input1.js', 'src/input2.js']
      }
    }
  }
});
```

# contrib-copy

- Copia archivos y carpetas
- Se instala con: `npm install grunt-contrib-copy --save-dev`
- Se carga en Gruntfile con: `grunt.loadNpmTasks('grunt-contrib-copy');`
- Ejemplo:

```
copy: {
  main: {
    files: [
      {expand: true, src: ['path/*'], dest: 'dest/', filter: 'isFile'},
      {expand: true, src: ['path/**'], dest: 'dest/'},
      {expand: true, cwd: 'path/', src: ['**'], dest: 'dest/'},
      {expand: true, flatten: true, src: ['path/**'], dest: 'dest/', filter: 'isFile'},
    ],
  },
},
```

# contrib-less

- Compila los archivos LESS a CSS
- Se instala con: `npm install grunt-contrib-less --save-dev`
- Se carga en Gruntfile con: `grunt.loadNpmTasks('grunt-contrib-less');`
- Ejemplo:

```
less: {
  development: {
    options: { paths: ["assets/css"] },
    files: { "path/to/result.css": "path/to/source.less" }
  },
  production: {
    options: {    paths: ["assets/css"],
                  plugins: [ new require('less-plugin-autoprefix')({browsers: ["last 2 versions"]}),
                         new require('less-plugin-clean-css')(cleanCssOptions) ],
      modifyVars: { imgPath: '"http://mycdn.com/path/to/images"', bgColor: 'red' }
    },
    files: {"path/to/result.css": "path/to/source.less"}
  }
}
```

# contrib-coffe

- Compila los archivos CoffeeScript aJavaScript
- Se instala con: `npm install grunt-contrib-coffee --save-dev`
- Se carga en Gruntfile con: `grunt.loadNpmTasks('grunt-contrib-coffee');`
- Ejemplo:

```
coffee: {
  compile: {
    files: {
      'path/to/result.js': 'path/to/source.coffee',
      'path/to/another.js': ['path/to/sources/*.coffee', 'path/to/more/*.coffee']
    }
  },
...
```