PLUGINS (PRÁCTICA)

Tiempo estimado: 20min

El objeto de esta práctica es afianzar y asentar los conocimientos teóricos presentados en la lección.

Al finalizar la práctica, el estudiante:

- Habrá creado un *plugin*.

Creación del proyecto

Para comenzar, vamos a crear el directorio de un proyecto para el desarrollo de un *plugin* de Justo.js:

1. Abrir una consola de PowerShell.
2. Instalar globalmente el generador de *plugins*:

   ```
   npm install -g justo-generator-plugin
   ```
3. Mostrar los parámetros del generador:

   ```
   > justo -g plugin help
   ```
4. Crear el directorio de la práctica e ir a él.
5. Crear la estructura de directorios del *plugin*:

   ```
   > justo -g plugin
   ```
6. Listar el contenido del directorio:

   ```
   > dir
   ```
7. Editar el archivo package.json y completar su contenido.
8. Guardar cambios.
9. Instalar las dependencias:

   ```
   > npm install
   ```

Preparación del entorno

El *plugin* lo vamos a desarrollar con JavaScript, concretamente mediante la especificación ES2015. Por lo que tenemos que compilarlo para que pueda ser utilizado en Node.js. Para ello, vamos a utilizar varios *plugins* oficiales: justo-plugin-babel para compilar; justo-plugin-jshint para comprobar buenas prácticas y errores gramaticales; justo-plugin-fs para borrar directorios y crear el paquete del *plugin* a publicar. Si fuésemos a publicar el paquete en NPM, podríamos utilizar justo-plugin-npm.

Por otra parte, vamos a utilizar el paquete justo-fs que contiene una API síncrona para trabajar con archivos. Se puede consultar la API en su repositorio GitHub, github.com/justojs/justo-fs.

Así pues, tenemos que añadir justo-fs y los *plugins* a las dependencias del proyecto:

1. Ir a la consola de PowerShell.
2. Abrir el proyecto, por ejemplo, con Atom:

   ```
   > atom .
   ```
3. Editar el archivo package.json.
4. Añadir justo-fs a la propiedad dependencies:

   ```
   "dependencies": {
       "justo-fs": "*"
     }
   ```

5. Añadir los *plugins* a la propiedad devDependencies:

```
"devDependencies": {
    "justo": "*",
    "justo-assert": "*",
    "justo-plugin-babel": "*",
    "justo-plugin-fs": "*",
    "justo-plugin-jshint": "*"
  }
```

6. Guardar cambios.
7. Ir a la consola de PowerShell.
8. Instalar dependencias no instaladas todavía:

```
> npm install
```

9. Si no está instalado Babel, instalarlo globalmente:

```
> npm install -g babel-cli
```

Para comprobar si Babel está instalado, consultar su versión:

```
> babel -V
6.4.5 (babel-core 6.4.5)
>
```

Debe ser 6.0 o superior.

Desarrollo del plugin

El objetivo del *plugin* es crear una tarea simple reutilizable para copiar archivos. Esta tarea ya está implementada por el *plugin* justo-plugin-fs, pero necesitamos un punto de partida fácil con el que aprender a desarrollar *plugins*.

1. Editar el archivo lib/op.js.
2. Implementar la operación para que copie un archivo.

   La función recibirá, en el *array* params, el archivo a copiar y con qué nombre copiarlo. La función de tarea la invocaremos, por ejemplo, como sigue:

   ```
   copy("Copiar a.txt como b.txt", "a.txt", "b.txt");
   ```

   Para la copia, usaremos el paquete justo-fs que viene con funciones síncronas específicas para trabajar con archivos y directorios.

   A continuación, se muestra un ejemplo del archivo:

   ```
   //imports
   import {copy} from "justo-fs";

   /**
    * Task operation.
    */
   export default function op(params) {
     if (!copy(params[0], params[1])) {
       throw new Error(`Error durante la copia.`);
     }
   }
   ```

3. Guardar cambios.
4. Editar el archivo index.js.
5. Exportar la tarea como predeterminada del paquete y darle un nombre descriptivo:

   ```
   //imports
   import {simple} from "justo";

   //api
   module.exports = simple({ns: "mi.punto.com", name: "copia"}, require("./lib/op").default);
   ```

6. Guardar cambios.

Una vez creado el *plugin*, sólo falta la parte de pruebas de unidad, pero eso se ve detenidamente en un curso

posterior de Justo.js. Recordemos que Justo.js viene con su propia suite de pruebas.

Creación del paquete

Vamos a automatizar algunas tareas del desarrollo mediante Justo.js.

1. Editar el archivo Justo.js.
2. Crear un flujo de trabajo que construya el paquete publicable:

```
catalog.workflow({name: "build", desc: "Construye el paquete publicable."}, function() {
  clean("Borrar el directorio build", {
    dirs: ["build/es5"]
  });

  jshint("Buenas prácticas y gramática", {
    output: true,
    src: [
      "index.js",
      "lib/op.js"
    ]
  });

  babel("Compilar", {
    comments: false,
    retainLines: true,
    preset: "es2015",
    files: [
      {src: "index.js", dst: "build/es5/"},
      {src: "lib/", dst: "build/es5/lib/"}
    ]
  });

  clean("Borrar directorio dist", {
    dirs: ["dist/es5"]
  });

  copy(
    "Crear paquete",
    {
      src: "build/es5/index.js",
      dst: "dist/es5/nodejs/justo-uplugin-copy/"
    },
    {
      src: "build/es5/lib/",
      dst: "dist/es5/nodejs/justo-uplugin-copy/lib"
    },
    {
      src: ["package.json", "README.md"],
      dst: "dist/es5/nodejs/justo-uplugin-copy"
    }
  );
});
```

   No hay que olvidar, añadir los objetos reutilizados en la sección de importaciones:

```
//imports
const justo = require("justo");
const catalog = justo.catalog;
const babel = require("justo-plugin-babel");
const clean = require("justo-plugin-fs").clean;
const copy = require("justo-plugin-fs").copy;
const jshint = require("justo-plugin-jshint");
```

3. Guardar cambios.
4. Ir a la consola de PowerShell.
5. Listar el catálogo de tareas del proyecto:

```
> justo -l
```

```
Name     Description
build    Construye el paquete publicable.
default  Default task.
>
```

6. Construir el paquete:

```
> justo build

  build
  build
    V Borrar el directorio build (16 ms)
    V Buenas prácticas y gramática (453 ms)
    V Compilar (2594 ms)
    V Borrar directorio dist (0 ms)
    V Crear paquete (15 ms)

  OK 5 | Failed 0 | Ignored 0 | Total 5

>
```

El paquete publicable se encuentra en dist/es5/nodejs/justo-uplugin-copy. Para su uso, bastará con publicarlo e instalarlo, o simplemente instalarlo directamente desde este directorio.