Tareas simples (práctica)

Tiempo estimado: 15min

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

Al finalizarla, el estudiante:

- Habrá creado el directorio de un proyecto Justo.
- Habrá creado tareas simples.
- Habrá registrado tareas en el catálogo de un proyecto.
- Habrá listado las tareas registradas de un proyecto.
- Habrá invocado tareas del catálogo de un proyecto.

Creación del proyecto

Para comenzar, vamos a crear el directorio de un proyecto Justo:

1. Abrir una consola.
2. Crear el directorio de la práctica e ir a él.
3. Crear los archivos Justo.json, Justo.js y package.json, para lo que usaremos el generador instalado globalmente en la práctica anterior:

   ```
   > justo -g justo
   ```
   Seleccionar standalone como tipo de proyecto.

   Cuando se selecciona standalone como tipo de proyecto, si el directorio actual no está vacío, propagará un error, Destination dir is not empty., y no generará ningún archivo. Esto es así para evitar errores si, por cualquier razón, nos hemos equivocado de directorio de proyecto.
4. Instalar las dependencias del proyecto mediante npm:

   ```
   > npm install
   ```
5. Mostrar el contenido de los archivos anteriores:

   ```
   > cat package.json
   > cat Justo.json
   > cat Justo.js
   ```

Registro de tareas simples síncronas

A continuación, vamos a crear una tarea simple síncrona, que comprueba si un archivo, pasado como argumento, existe. Para ello, usaremos la función fs.accessSync() de Node.js que, recordemos, propaga un error cuando el archivo no pasa la comprobación.

1. Ir a la consola.
2. Editar el archivo Justo.js.
3. Registrar una tarea simple con el nombre comprobar que realice lo deseado. El archivo podría quedar como sigue:

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

   //catalog
   catalog.workflow({name: "build", desc: "Build the package."}, function() {
   ```



```
  jshint("Best practices and grammar", {
    output: true,
    src: ["index.js", "Justo.js"]
  });
});

catalog.simple(
  {name: "comprobar", desc: "Comprobar que un archivo existe."},
  function(params) {
    fs.accessSync(params[0]);
  }
);

catalog.macro({name: "default", desc: "Default task."}, ["build"]);
```

4. Guardar cambios.
5. Ir a la consola.
6. Listar las tareas registradas en el catálogo del proyecto:

```
> justo -c
Name       Description
build      Build the package.
comprobar  Comprobar que un archivo existe.
default    Default task.
>
```

7. Ejecutar la tarea comprobar para que compruebe si existe el archivo Justo.json:

```
> justo comprobar:Justo.json

  comprobar
  [ OK ] comprobar (0 ms)

  OK 1 | Failed 0 | Ignored 0 | Total 1

>
```

Observar que al finalizar la ejecución, Justo muestra un resumen de las tareas ejecutadas.

8. Ejecutar la tarea comprobar pasando como argumento desconocido.txt:

```
> justo comprobar:desconocido.txt

  comprobar
  [ ER ] comprobar (1 ms)
    Error: ENOENT: no such file or directory, access 'desconocido.txt'

  OK 0 | Failed 1 | Ignored 0 | Total 1

>
```

Observe que si la operación de tarea propaga un error, el motor considerará que la invocación ha acabado en fallo y así lo mostrará en el informe resumen.

Tareas simples asíncronas

Ahora, vamos a hacer lo mismo que antes, pero mediante una tarea asíncrona. En este caso, usaremos la función fs.access().

1. Editar el archivo Justo.js.
2. Añadir una tarea simple asíncrona con el nombre comprobar2 que realice lo deseado:

```
catalog.simple(
  {name: "comprobar2", desc: "Comprobar que un archivo existe asíncronamente."},
  function(params, done) {
    fs.access(params[0], done);
  }
);
```

En este caso, la función fs.access() es asíncrona. Como primer parámetro, espera el archivo a comprobar; como segundo, la función que debe invocar cuando haya finalizado su operación completamente. Esta función la invocará sin argumentos, si todo ha ido bien, o pasándole el

error como primer argumento. De ahí que podamos pasarle nuestro parámetro done sin problemas.

Otra manera de hacer lo mismo sería como sigue:

```
catalog.simple(
  {name: "comprobar2", desc: "Comprobar que un archivo existe asíncronamente."},
  function(params, done) {
    fs.access(params[0], function(error) {
      if (error) done(error);
      else done();
    });
  }
);
```

Recordemos que las tareas utilizan el parámetro función done() para indicarle al motor de Justo que han terminado. Si no se invoca en una tarea asíncrona, el motor se quedará esperando.

3. Guardar cambios.
4. Ir a la consola.
5. Listar las tareas registradas:

```
> justo -c
Name        Description
build       Build the package.
comprobar   Comprobar que un archivo existe.
comprobar2  Comprobar que un archivo existe asíncronamente.
default     Default task.
>
```

6. Comprobar que el archivo Justo.json existe asíncronamente:

```
> justo comprobar2:Justo.json

  comprobar2
  [ OK ] comprobar2 (5 ms)

  OK 1 | Failed 0 | Ignored 0 | Total 1

>
```

Tarea predeterminada

Finalmente, vamos a configurar la tarea predeterminada para que compruebe que los archivos Justo.js y Justo.json existen:

1. Editar el archivo Justo.js.
2. Modificar la definición de la tarea predeterminada para que invoque la tarea del catálogo comprobar dos veces, una para cada archivo:

```
catalog.macro(
  {name: "default", desc: "Default task."},
  [
    {task: "comprobar", params: ["Justo.js"]},
    {task: "comprobar", params: ["Justo.json"]}
  ]
);
```

En una lección posterior, se expone detalladamente las macros.

3. Guardar cambios.
4. Ir a la consola.
5. Ejecutar la tarea predeterminada:

```
> justo

  default
  default
    [ OK ] comprobar (1 ms)
    [ OK ] comprobar (0 ms)
```



```
  OK 2 | Failed 0 | Ignored 0 | Total 2

>
```

La tarea predeterminada se puede invocar explícitamente mediante `justo default`:

```
> justo default

  default
  default
    [ OK ] comprobar (1 ms)
    [ OK ] comprobar (0 ms)

  OK 2 | Failed 0 | Ignored 0 | Total 2

>
```

6. Cerrar la consola.

