Módulos

Tiempo estimado: 15min

Acabamos de ver los *plugins*, un medio a través del cual extender la funcionalidad de Justo, proporcionando tareas reutilizables. Ahora, vamos a presentar formalmente los módulos, paquetes de Node que definen, principalmente, procesos que podemos utilizar en cualquier momento, por ejemplo, para implementar procesos de instalación o configuración.

La lección comienza introduciendo el concepto de módulo y algunos posibles usos. A continuación, se ilustra cómo ejecutar la funcionalidad definida en los módulos. Finalizamos describiendo el desarrollo de nuestros propios módulos.

Al finalizar la lección, el estudiante sabrá:

- Qué es un módulo.
- Cómo ejecutar módulos.
- Cómo listar los módulos instalados.
- Cómo implementar módulos.

Introducción

Un módulo (*module*) es un paquete de Node autocontenido que realiza una o más actividades, procesos, trabajos o tareas relacionadas entre sí como, por ejemplo, la instalación, la desinstalación o la configuración de un determinado producto. Es autónomo y se dedica exclusivamente a hacer algo determinado mediante Justo.

Cada módulo presenta su propio catálogo de tareas. Generalmente, relacionadas con algo en concreto. Lo más habitual es empaquetar un conjunto de tareas o procesos relacionados con un determinado producto.

Uso de módulos

Los módulos se utilizan principalmente para implementar procesos normalizados como, por ejemplo, el despliegue de productos y la configuración de sistemas. Son muchas las organizaciones que redactan sus propios estándares de instalación para que todas ellas se instalen de igual manera en todas las máquinas. Estos estándares se pueden implementar mediante módulos de Justo, permitiendo así que las instalaciones se realicen bajo el estándar de la organización.

Las principales ventajas de implementar los estándares mediante módulos de Justo son:

- Aumento de la productividad.
- Aumento de la calidad.
- Aumento de la precisión o exactitud.
- Reducción de costes.
- Reducción de errores.
- Reducción de tareas manuales.
- Reducción de tiempos.

Y las principales funciones de los módulos son:

- La implementación de procesos de instalación.
- La configuración de productos.
- El empaquetado de tareas ejecutables desde línea de comandos.

Ejecución de módulos

Una vez implementado el módulo, algo que veremos en breve, lo más habitual es publicarlo, por ejemplo, en el repositorio de NPM. Así, cuando tengamos que utilizar su funcionalidad, bastará con instalarlo en nuestra máquina mediante npm y, a continuación, invocarlo con justo, concretamente, con la opción -m o --module:

```
justo -m módulo
justo -m módulo tarea tarea tarea…
```

A continuación, se muestra el proceso de instalación del sistema de gestión de bases de datos RethinkDB mediante Justo:

```
$ npm install -g install-rethinkdb-on-ubuntu
/home/me/.npm/lib
└── install-rethinkdb-on-ubuntu@0.1.0

$ sudo -E justo -m install-rethinkdb-on-ubuntu
[sudo] password for me:

  default
    default
      install
      [ OK ] Check whether RethinkDB installed (2134 ms)
      [ OK ] Check whether wget installed (409 ms)
      [ OK ] Check whether /etc/apt/sources.list.d/rethinkdb.list exists (1 ms)
      [ OK ] Create /etc/apt/sources.list.d/rethinkdb.list (12 ms)
      [ OK ] Add APT key (4848 ms)
      [ OK ] Update APT index (16836 ms)
      [ OK ] Check whether rethinkdb package available (381 ms)
      [ OK ] Install RethinkDB (28675 ms)
      [ OK ] Check rethinkdb command (31 ms)

  OK 9 | Failed 0 | Ignored 0 | Total 9

$
```

Puede consultar el módulo en su repositorio de GitHub, github.com/justojsm/install-rethinkdb-on-ubuntu.

Puede encontrar módulos oficiales de Justo en justojs.org.

Listado de módulos instalados

Para conocer qué módulos hay instalados, es decir, qué paquetes globales tienen justo-module entre sus palabras claves, se puede utilizar la opción --modules. Ejemplo:

```
$ justo --modules
Name                          Version Description
install-arangodb-on-ubuntu    0.1.0   Justo module for installing ArangoDB on Ubuntu.
install-cassandra-on-debian 0.1.1   Justo module for installing Cassandra on Debian distros.
install-rethinkdb-on-ubuntu 0.1.0   Justo module for installing RethinkDB on Ubuntu.
$
```

Creación de un módulo

Ahora, vamos a ver cómo desarrollar nuestros propios módulos. Para ello, la mejor manera de arrancar el proyecto es usar el generador justo-generator-justo con el comando module, el cual creará la estructura del proyecto rápidamente:

```
justo -g justo module
```

La estructura del directorio del proyecto es muy similar a la de cualquier otro proyecto de Node. Aunque hay que tener algunas cosas en cuenta:

- La propiedad keywords del archivo package.json debe de contener justo-module, para facilitar su búsqueda.

  Sólo los paquetes instalados que tengan justo-module en su propiedad keywords serán listados por el comando justo --modules.

- El punto de entrada del módulo debe ser index.js. Y el archivo index.js debe importar el módulo Justo.js.

Recordemos que el punto de entrada de un paquete NPM se indica mediante la propiedad main del archivo package.json.

- El archivo Justo.js contiene el catálogo de tareas asociado a los procesos automatizados por el módulo como, por ejemplo, la instalación, la configuración, etc. En definitiva, las tareas que son invocables mediante justo -m una vez instalado el módulo.
- El archivo Justo.dev.js contiene el catálogo de tareas específicas de desarrollo del módulo como, por ejemplo, su compilación, la creación del paquete, la publicación en NPM, etc.
- Los módulos tienen el paquete justo entre sus dependencias principales en el archivo package.json, esto es, en la propiedad dependencies.

  Mientras que cuando integramos Justo en otro proyecto, generalmente, justo es una dependencia de desarrollo, devDependencies.

Archivo Justo.dev.js

Hasta el momento, hemos visto que el archivo Justo.js contiene el catálogo de tareas automatizadas del proyecto. Existe un segundo catálogo definido mediante el archivo Justo.dev.js. Su comportamiento es similar, pero se invoca mediante la opción -d o --dev de justo. Cuando no se indica esta opción, se usará siempre el catálogo definido por el archivo Justo.js. En cambio, si se usa la opción -d, se usará Justo.dev.js, en vez de Justo.js.

Este archivo se suele utilizar únicamente en los módulos de Justo. Contiene el catálogo de tareas automatizadas relacionadas con el desarrollo del módulo. Dejando el archivo Justo.js para contener las tareas automatizadas propias del módulo como, por ejemplo, aquellas que realizan la instalación o configuración de un determinado producto.

Por ejemplo, en el módulo de instalación de RethinkDB, el archivo Justo.js es como sigue:

```
//imports
const justo = require("justo");
const catalog = justo.catalog;
const apt = require("justo-plugin-apt");
const cli = require("justo-plugin-cli");
const fs = require("justo-plugin-fs");
const sync = require("justo-sync");
const getos = require("getos");

//private data
const PKG = "rethinkdb";

//catalog
catalog.workflow({name: "install", desc: "Install RethinkDB."}, function() {
  var os;

  //(1) get OS info
  os = sync((done) => getos(done));

  if (os.os != "linux" || !/Ubuntu/.test(os.dist))  {
    throw new Error("Distribution not supported by this module.");
  }

  //(2) install
  if (!apt.installed("Check whether RethinkDB installed", {name: PKG})) {
    //(2.1) install dependencies for installing
    if (!apt.installed("Check whether wget installed", {name: "wget"})) {
      apt.install("Install wget package", {name: "wget"});
    }

    //(2.2) add package source if needed
    if (!fs.exists("Check whether /etc/apt/sources.list.d/rethinkdb.list exists", {src:
"/etc/apt/sources.list.d/rethinkdb.list"})) {
      cli("Create /etc/apt/sources.list.d/rethinkdb.list", {
        cmd: "bash",
        args: ["-c", "source /etc/lsb-release && echo \"deb
http://download.rethinkdb.com/apt $DISTRIB_CODENAME main\" | tee
```



```
/etc/apt/sources.list.d/rethinkdb.list"],
      });
    }

    cli("Add APT key", {
      cmd: "bash",
      args: ["-c", "wget -qO- https://download.rethinkdb.com/apt/pubkey.gpg | apt-key add
-"]
    });

    apt.update("Update APT index");
    if (!apt.available(`Check whether ${PKG} package available`, {name: PKG})) return;

    //(2.3) install RethinkDB
    apt.install("Install RethinkDB", {
      name: PKG
    });
  }

  //(3) post
  cli("Check rethinkdb command", {
    cmd: "bash",
    args: ["-c", "rethinkdb --version"]
  });
});

catalog.macro({name: "default", desc: "Install RethinkDB and its dependencies."},
["install"]);
```

Mientras que Justo.dev.js contiene:

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

//catalog
const jslint = catalog.simple({
  name: "jslint",
  desc: "Parse best practices and grammar (JavaScript).",
  task: jslinter,
  params: {
    output: true,
    src: [
      "index.js",
      "Justo.js",
      "Justo.dev.js",
      "lib/",
    ]
  }
});

catalog.workflow({name: "build", desc: "Build the package."}, function(params) {
  var newDist = false;

  //(1) params
  for (let param of params) {
    if (param == "new") newDist = true;
  }

  //(2) tasks
  jslint("Best practices and grammar (JavaScript)");

  clean("Remove build directory", {
    dirs: ["build/es5"]
  });
```



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

  if (newDist) {
    clean("Remove dist directory", {
      dirs: ["dist/es5"]
    });
  }

  copy(
    "Create package",
    {
      src: "build/es5/index.js",
      dst: "dist/es5/nodejs/install-rethinkdb-on-ubuntu/"
    },
    {
      src: "build/es5/lib/",
      dst: "dist/es5/nodejs/install-rethinkdb-on-ubuntu/lib",
      force: true
    },
    {
      src: ["package.json", "README.md", "Justo.js"],
      dst: "dist/es5/nodejs/install-rethinkdb-on-ubuntu/"
    }
  );
});

catalog.simple({
  name: "publish",
  desc: "NPM publish.",
  task: npm.publish,
  params: {
    who: "justojs",
    src: "dist/es5/nodejs/install-rethinkdb-on-ubuntu"
  }
});

catalog.simple({
  name: "install",
  desc: "Install the module globally for testing.",
  task: npm.install,
  params: {
    pkg: "./dist/es5/nodejs/install-rethinkdb-on-ubuntu",
    global: true,
    output: false
  }
});

catalog.macro({name: "default", desc: "Build the Justo module."}, ["build"]);
```

Así pues, para invocar la tarea build del catálogo de desarrollo, es decir, la del archivo Justo.dev.js, tendremos que usar:

```
justo -d build
```

Y para consultar este catálogo:

```
justo -d -c
justo -dc
```

Dependencias del módulo

Como con cualquier otro módulo o paquete de NPM, las dependencias se registran en las propiedades

dependencies y devDependencies del archivo package.json. Cuando se instala un módulo de Justo mediante npm, sólo se instala el archivo Justo.js, *no* así Justo.dev.js. Por lo tanto, no olvide añadir a la propiedad dependencies del archivo package.json todos los paquetes utilizados en el catálogo del archivo Justo.js.

En el caso de nuestro módulo de ejemplo, éstas son las dependencias:

```
"dependencies": {
  "getos": "*",
  "justo": "*",
  "justo-sync": "*",
  "justo-plugin-apt": "*",
  "justo-plugin-cli": "*",
  "justo-plugin-fs": "*",
  "justo-plugin-npm": "*"
},
"devDependencies": {
  "babel-preset-es2015": "*",
  "justo-plugin-eslint": "*",
  "justo-plugin-babel": "*"
}
```

