Introducción

Tiempo estimado: 10min

Ésta es la primera lección del curso Acceso a bases de datos SQLite con Node.js. Tiene como objetivo describir cómo trabajar con SQLite en la plataforma Node. Asume del estudiante conocimientos previos de JavaScript, Node.js y SQLite.

Comenzamos la lección introduciendo el concepto de *driver* de bases de datos y los tipos existentes. A continuación, presentamos el *driver* sqlite3 disponible en el repositorio oficial de NPM. Finalmente, presentamos el programa del curso.

Al finalizar la lección, el estudiante sabrá:

- Qué es un *driver*.
- Cuáles son los tipos de *drivers*.
- Cuál es el *driver* que utilizaremos.
- Cuál es el programa del curso.

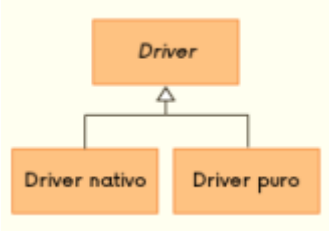
Introducción

Un driver es un componente de software que permite, a una aplicación, el acceso a una instancia de base de datos. Actualmente, Node proporciona *drivers* para muchos sistemas de gestión de bases de datos, entre otros, Cassandra, CouchDB, MariaDB, MongoDB, PotsgreSQL, Redis, RethinkDB, SQL Server y SQLite.

El objeto de este curso es presentar detenidamente el *driver* por excelencia para el acceso a bases de datos SQLite.

Tipos de drivers

Atendiendo a su implementación, se distingue básicamente dos tipos de *drivers*, los nativos y los puros.



Para algunos sistemas de gestión de bases de datos, se dispone de *drivers* de ambos tipos como es el caso de PostgreSQL. En cambio para SQLite, el *driver* es nativo.

Drivers nativos

Un driver nativo (*native driver*) es aquel que está implementado en JavaScript y C/C++.

Suelen ser multiplataforma, pero no siempre tiene que ser así, debido a su dependencia de C/C++.. Cada vez que se instala, hay que instalar también la biblioteca nativa correspondiente. Su principal ventaja es que suelen ser más rápidos que los puros.

Drivers puros

Un driver puro (*pure driver*) es aquel que está implementado íntegramente en JavaScript y, por lo tanto, es 100% portable de una plataforma a otra, ya que el acceso a la instancia de bases de datos se hace a través de conexiones de red. No depende de ninguna biblioteca nativa extra.

La principal desventaja de este tipo de *drivers*, con respecto a los nativos, es que suelen ser más lentos,

pero, como ventaja, al encontrarse implementados totalmente en JavaScript suelen funcionar en cualquier plataforma para la que está disponible Node.

Driver de SQLite

SQLite dispone de varios *drivers* para Node.js. El más utilizado es sqlite3, con más de 200.000 descargas mensuales y actualizaciones periódicas para resolver *bugs* y añadir nueva funcionalidad. Este *driver* es nativo, porque tiene una parte implementada en JavaScript y otra implementada en C++, la cual además utiliza la biblioteca nativa de SQLite para el acceso a la instancia de base de datos.

Es multiplataforma, funcionando en Android, Linux, OS X y Windows.

Instalación del driver

La recomendación de los desarrolladores de SQLite es importar cualquier *driver* de acceso a SQLite localmente al paquete, no se recomienda la instalación global. Para ello, registraremos una entrada en la propiedad dependencies del archivo package.json del paquete:

```
"dependencies": {
  "sqlite3": "*"
}
```

Cuando se instala el *driver*, el paquete comprueba si la máquina dispone de la biblioteca SQLite instalada. Si no lo está, utilizará una copia propia. Por otra parte, comprobará si dispone de una copia compilada de la parte nativa del *driver* para la plataforma. Si es así, la usará. En otro caso, realizará una compilación durante la instalación para lo cual es necesario tener un compilador de C/C++.

Importación del driver

Por convenio y buenas prácticas, el paquete se importa como sqlite:

```
//si sentencia import no soportada
const sqlite = require("sqlite3");

//si sentencia import soportada
import sqlite from "sqlite3";
```

Para validar que funciona, bastará con comprobar que se puede importar:

Modo verboso

En algunas ocasiones, se puede desear que el *driver* muestre información de depuración, para ayudar a resolver problemas. En estos casos, se puede utilizar el modo verboso (*verbose mode*), que proporciona más información que el predeterminado. Pare este fin, se utiliza la función verbose() del *driver*:

```
const sqlite = require("sqlite3").verbose();
```

Arquitectura

El *driver* sqlite3 tiene básicamente dos componentes, la clase Database y la clase Statement.

La clase Database representa una conexión al archivo principal de una base de datos SQLite. Mientras que la clase Statement representa una consulta preparada.

Información del curso

Este minicurso está dedicado al paquete sqlite3 de Node, disponible en el repositorio oficial de NPM.

Tiene como objetivo presentar cómo acceder a bases de datos SQLite.

Al finalizarlo, el estudiante sabrá:

- Qué es un *driver*.
- Qué tipo de *driver* es sqlite3.
- Cómo acceder a bases de datos SQLite desde Node.
- Cómo ejecutar transacciones.

- Cómo trabajar con conexiones cacheadas.
- Cómo crear bases de datos en memoria.

Conocimientos previos

El estudiante debe tener conocimientos de JavaScript, Node.js y SQLite.

Plan del curso

El curso tiene una duración aproximada de 2 horas. Se divide en 3 lecciones, cada una de ellas con una parte de teoría y generalmente una de práctica Al finalizar el curso, puede realizar un examen de tipo test, con el que evaluar los conocimientos adquiridos.

El enfoque a seguir es muy sencillo: ir lección a lección; primero hay que leer la teoría y, después, realizar la práctica. Se recomienda encarecidamente que el estudiante realice cada lección, tanto teoría como práctica, en el mismo día, con el menor número de interrupciones a lo largo de su estudio. Al finalizar el curso, se recomienda realizar el examen.

A continuación, se enumera las distintas lecciones y el tiempo estimado para su estudio:

Lección	Teoría	Práctica	Descripción
1 Introducción	10min	-	Esta lección.
2 Consultas	25min	15min	Cómo conectar a una base de datos SQLite y ejecutar consultas y transacciones.
3 Conceptos avanzados	10min	15min	Cómo usar la caché de conexiones, cómo utilizar bases de datos en memoria y los modos de ejecución y cómo cargar extensiones.
Examen	30min		Evaluación de los conocimientos adquiridos.

Información de publicación

Título Acceso a bases de datos SQLite con Node.js

Autor Raúl G. González - raulggonzalez@nodemy.com

Primera edición Noviembre de 2016

Versión actual 1.0.0

Versión de sqlite3 3.1

Contacto hola@nodemy.com

