Clase 17: Conexión y consultas a base de datos

Sitio:	Centro de E-Learning - UTN.BA	Imprimido por:	Nelson Brian Avila Solano
Curso:	Curso de Backend Developer - Turno Noche	Día:	Tuesday, 30 de December de 2025, 20:00
Libro:	Clase 17: Conexión y consultas a base de datos		

Descripción

Objetivos de la clase

▪ Configurar una conexión a una base de datos relacional usando phpMyAdmin y XAMPP.

▪Realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) en la base de datos relacional usando Node.js y Express.

▪Visualizar información en HBS

▪Hacer uso de pool de conexiones

Tabla de contenidos

1. Introducción

El uso de bases de datos en aplicaciones web es fundamental para almacenar y gestionar información de manera eficiente. En Node.js, la combinación de Express con MySQL permite desarrollar aplicaciones dinámicas que interactúan con bases de datos relacionales. A través de herramientas como XAMPP y phpMyAdmin, es posible configurar un entorno de desarrollo local que facilita la creación y administración de bases de datos. Además, mediante consultas SQL, se pueden realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar), permitiendo una gestión completa de los datos dentro de la aplicación.

Para optimizar el rendimiento y mejorar la escalabilidad, se recomienda el uso de un pool de conexiones, que mantiene un conjunto de conexiones abiertas y reutilizables en lugar de establecer una nueva conexión para cada consulta. Esto reduce la sobrecarga del servidor y mejora la eficiencia del sistema, especialmente en aplicaciones con alto tráfico. Además, mediante la integración de Handlebars como motor de plantillas, es posible visualizar los datos de manera estructurada en el frontend, logrando así una experiencia de usuario más clara y organizada.

2. Conexión y consultas a base de datos

1. Configuración de XAMPP y phpMyAdmin

- Instalar y configurar XAMPP en la máquina.
- Iniciar los servicios Apache y MySQL.
- Acceder a phpMyAdmin en el navegador web y crear una base de datos y una tabla.

Crear estructura de base de datos

- Crear base de datos
- Crear Tabla

CRUD

- Create
- Read
- Update
- Delete

2. Instalar dependencias

1. Express (marco de node.js)
2. MySQL (controlador mysql para node.js)
3. HBS (motor de plantilla)

Conexión

https://www.npmjs.com/package/mysql2

3. Proyecto

```
npm init -y

npm install --save express mysql2 hbs
```

4. Estructura del proyecto

Conexión a base de datos

```
// Conexión a base de datos

const conn = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password:'',
    database: 'crud-node'
})

conn.connect((err) =>{
    if(err) throw err;
    console.log("CONEXIÓN ESTABLECIDA");
})
```

Creación de las rutas para las operaciones CRUD

- Crear las rutas para la visualización de registros utilizando el método GET.
- Crear rutas para la creación de registros utilizando el método POST.
- Crear rutas para la actualización de registros utilizando el método PUT.
- Crear las rutas para la eliminación de registros utilizando el método DELETE.

Select

```
// SELECT router.get('/', (req, res) => {    let
sql = "SELECT * FROM producto";    let query =
conn.query(sql, (err, results) => {        if
(err) throw err;
res.render('../views/productos', {
results: results
        });
    });
});
```

Insert

```
// Insertar router.post('/save', (req, res) => {    let
data = { producto_nombre: req.body.producto_nombre,
producto_precio: req.body.producto_precio };    let sql =
"INSERT INTO producto SET ?";    let query =
conn.query(sql, data, (err, results) => {        if (err)
throw err;        res.redirect('/productos');
   });
});
```

Actualizar

```
conexion.query('UPDATE estudiantes SET nombre =
"Severus", apellido= "Snape" WHERE
id_estudiante=6', function (error, results){    if
(error) throw error;    console.log('¡Registro
Actualizado', results);
});
```

Delete

```
conexion.query ('DELETE FROM estudiantes
WHERE id_estudiante=203',
function(error, results){    if(error)
throw error; console.log('¡Registro
Eliminado!', results)

})
```

3. Uso de pool de conexiones

Uso de pool de conexiones

Pool de conexiones

Un pool de conexiones es una técnica que mejora la eficiencia en la gestión de conexiones a la base de datos. En lugar de abrir y cerrar una nueva conexión cada vez que se realiza una consulta, se crea un grupo (pool) de conexiones reutilizables.

Funcionamiento del pool de conexiones

Se establece un número limitado de conexiones que están abiertas en todo momento.

Las conexiones se reutilizan cuando se realizan consultas.

Cuando una conexión no se está utilizando, vuelve al pool para que otras solicitudes puedan ejecutarse.

Beneficios de utilizar un pool de conexiones

1. **Rendimiento mejorado:**

   Evita el coste de abrir y cerrar conexiones para cada consulta.

   Las conexiones se gestionan eficientemente, lo que reduce el tiempo de espera de las consultas.

2. **Escalabilidad:**

   El pool limita el número de conexiones abiertas simultáneamente, evitando la saturación de la base de datos.

3. **Reducción de la sobrecarga del servidor:**

   Menos conexiones abiertas y cerradas reducen la carga en el servidor de base de datos.

Parámetros importantes del pool de conexiones

- connectionLimit: Especifica el número máximo de conexiones activas que se permiten en el pool al mismo tiempo. Establecer un límite es importante para evitar sobrecargar la base de datos.
- waitForConnections: Determina si las solicitudes que llegan cuando todas las conexiones del pool están ocupadas deben esperar a que una conexión quede libre (true) o fallar inmediatamente (false).
- queueLimit: Define cuántas solicitudes pueden estar en cola esperando una conexión. Si la cola se llena, las solicitudes adicionales son rechazadas.

Casos de uso recomendados para pools de conexiones

- Aplicaciones web con alto volumen de tráfico que necesitan consultar bases de datos constantemente.
- Sistemas en producción donde la eficiencia y la velocidad son cruciales.

▪ Aplicaciones donde es importante limitar el número de conexiones simultáneas para no sobrecargar el servidor.

Desventajas del pool de conexiones

▪ Si se configura mal, un número muy bajo de conexiones en el pool puede hacer que las solicitudes se queden esperando, lo que afecta el rendimiento.

▪ Un número muy alto de conexiones en el pool puede saturar el servidor de base de datos, afectando a otras aplicaciones que lo utilicen.

Comparación entre conexión tradicional y pool de conexiones en Mysql

Características	Conexión tradicional	Pool de conexiones
Manejo de conexiones	Abre y cierra una conexión por cada consulta, lo que puede ser costoso en tiempo.	Mantiene un conjunto de conexiones abiertas que se reutilizan para múltiples consultas.
Rendimiento	Menor rendimiento debido a la sobrecarga de establecer conexiones repetidamente.	Mejor rendimiento al reducir la latencia al reutilizar conexiones ya establecidas.
Configuración de las conexiones	Cada conexión se establece independientemente, aumentando la latencia.	Las conexiones son gestionadas por el pool, mejorando la latencia y optimizando el rendimiento.

4. ¡A practicar!

¡A practicar!

Establecé una conexión a base de datos desde tu servidor node y emplea las siguientes operaciones:

- SELECT
- INSERT
- UPDATE
- DELETE