


Open · Opened 2 days ago by  [Rubén Montero](#)

Conectarse y desconectarse una sola vez

Resumen

- Crearemos una nueva clase en la que trabajaremos en las siguientes tareas, `MoviesConnector.java`
- Tendrá un atributo privado de tipo `Connection` para almacenar la conexión a la base de datos
- Establecerá dicha conexión a la base de datos en su método constructor
- Tendrá un método `closeConnection` para cerrar la conexión

Descripción

Nuestro `MoviesDataProvider` tiene diversos métodos que establecen conexiones a la base de datos `movies.db`, lanzan consultas, y cierran la conexión.

Abrir una conexión es una operación relativamente costosa. No queremos abrir una nueva conexión para cada consulta. Lo **ideal** sería aprovechar la misma conexión para lanzar todas las consultas necesarias.

¿Cómo hacer eso?

Recordemos que la conexión a la base de datos se representa mediante un objeto de tipo `Connection`. Podemos **almacenar** dicho objeto como un **atributo** de la clase.

Pongámoslo en práctica

Creemos una **nueva clase** `MoviesConnector.java`, con un atributo privado como el que acabamos de comentar.

```
import java.sql.Connection;

public class MoviesConnector {
    private Connection connection;

    // ...
}
```

Ahora, en el método **constructor**, como ya sabemos hacer, podemos **establecer una conexión**.

```
import java.sql.*;

public class MoviesConnector {
    private Connection connection;

    public MoviesConnector() {
        String connectionStr = "jdbc:sqlite:db/sqlite3/movies.db";
        try {
            // Esta vez la conexión se guarda
            // como atributo.
            // Cada instancia de MoviesConnector
            // tendrá su conexión. ¡Hurra!
            this.connection = DriverManager.getConnection(connectionStr);
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }
}
```

¿Y esto para qué?

Recordemos que queremos abrir la conexión **una sola vez**.

Luego, la aprovecharemos para lanzar múltiples consultas. La idea es que haya métodos que se sirvan de `this.connection` para ello, aunque no implementaremos ninguno de momento.

```
public class MoviesConnector {
    private Connection connection;

    public MoviesConnector() { /* ... */ }

    public String lanzarUnaConsulta() {
        // Se usa this.connection
        /* ... */
    }

    public ArrayList<String> lanzarOtraConsulta() {
        // Se usa this.connection
        /* ... */
    }
}
```

¡Un momento!

¿Y cuándo se cierra la conexión?

¡Vaya!

Es verdad. De momento, no hemos tenido en cuenta que la conexión **debe cerrarse**.

Una idea sencilla

Añadamos un nuevo método que se encargue de ello:

```
public class MoviesConnector {
    private Connection connection;

    /* ... */

    public void closeConnection() {
        try {
            this.connection.close();
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }
}
```

¡Listo!

Por último

Verifica que el test funciona correctamente.

Haz **commit** y **push** para subir los cambios al repositorio.



Rubén Montero @ruben.montero changed milestone to [%Sprint 2](#) 2 days ago