


Open · Opened 2 days ago by  **Rubén Montero**

CRUD

Resumen

- Entenderemos el concepto de CRUD
- Escribiremos un método en nuestro **MoviesConnector** que **inserta** (crea) una nueva fila en la tabla de la base de datos

Descripción

Un concepto esencial que aparece en distintas disciplinas de la informática es el de **CRUD**. Son siglas en inglés para:

- **C**reate
- **R**ead
- **U**pdate
- **D**elete

...y definen las operaciones básicas de acceso a datos.

En una base de datos SQL, dichas operaciones se corresponden con sentencias:

- **INSERT**
- **SELECT**
- **UPDATE**
- **DELETE**

SQL **INSERT**

Es la sentencia destinada a grabar un nuevo registro (fila) en una tabla.

Su sintaxis es:

```
INSERT INTO Tabla VALUES (valor1, valor2, valor3)
```

Si queremos realizar una inserción indicando **sólo algunos** campos, podemos hacerlo indicándolos con unos paréntesis:

```
INSERT INTO Tabla (columna2, columna3) VALUES (valor2, valor3)
```

Esto es útil, por ejemplo, si no queremos especificar un ID porque el *sistema gestor de base de datos* lo genera de manera automática al ser *autoincremental*.

Insertar la película *Inside Out*

La película *Inside Out* fue un éxito de Disney del año 2015.

Vamos a insertarla en la base de datos ejecutando la siguiente sentencia:

```
INSERT INTO TMovies (title, year, duration, countryIso3166, genre, synopsis) VALUES ('Inside Out', 2015, 94, 'us', 'animation')
```

Un nuevo método en **MoviesConnector**

Lo llamaremos `insertExampleFilm`, y su primera misión será crear un objeto `Statement` aprovechando `this.connection`

```
public void insertExampleFilm() {  
    try {  
        Statement statement = this.connection.createStatement();  
        // ...  
    } catch (SQLException e) {  
        throw new RuntimeException(e);  
    }  
}
```

Filas afectadas

Cuando invocamos `.executeQuery`, se nos *devuelve* un objeto tipo `ResultSet`. Contiene los resultados de la consulta.

¿Y si queremos hacer un `INSERT`?

En ese caso, el resultado **no** consiste en un conjunto de filas.

El resultado es un simple y sencillo `int`. Viene a ser **el número de filas afectadas**.

Cuando *insertamos* datos, siempre será `1`. (**Una** fila afectada, es decir, **insertada**).

Si estamos *eliminando* (`DELETE`) o *actualizando* (`UPDATE`) filas, puede haber *más de una* afectada. Lo veremos más adelante.

`.executeUpdate` en vez de `.executeQuery`

Al lanzar un `SELECT`, invocamos `.executeQuery` y se devuelve un `ResultSet`.

Para ejecutar un `INSERT`, `UPDATE` o `DELETE`, debemos invocar `.executeUpdate`, que devuelve un tipo `int`:

```
public void insertExampleFilm() {
    try {
        Statement statement = this.connection.createStatement();
        int affectedRows = statement.executeUpdate(/* Debe ser INSERT, DELETE o UPDATE. De lo contrario, saltará una exce
        // ...

    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

¿Puede fallar la inserción?

Sí. Si insertamos una fila que viola una restricción como NOT NULL o UNIQUE. También si la sentencia INSERT está mal formada o contiene tipos de datos incorrectos.

Por ello, es adecuado controlar esa posibilidad. **De momento, sólo haremos un *print*:**

```
public void insertExampleFilm() {
    try {
        Statement statement = this.connection.createStatement();
        int affectedRows = statement.executeUpdate(/* Debe ser INSERT, DELETE o UPDATE. De lo contrario, saltará una exce
        if (affectedRows == 1) {
            System.out.println("Se ha insertado una nueva fila");
        } else {
            System.out.println("Parece que ha habido un problema");
        }
        // ...
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

No es predecible que esta sentencia falle, pues los datos son correctos.

¿Algo más?

Sí, un pequeño detalle. Hay que invocar `.close()` sobre la instancia de `Statement`. Esto liberará recursos.

En casos anteriores no lo hemos hecho, puesto que al cerrar la conexión de la base de datos, todos los recursos se liberan. Como ahora hacemos las cosas ligeramente distintas, debemos liberar el `Statement` lo antes posible:

```
public void insertExampleFilm() {
    try {
        Statement statement = this.connection.createStatement();
        int affectedRows = statement.executeUpdate(/* Debe ser INSERT, DELETE o UPDATE. De lo contrario, saltará una exce
        if (affectedRows == 1) {
            System.out.println("Se ha insertado una nueva fila");
        } else {
            System.out.println("Parece que ha habido un problema");
        }
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

```
    }  
    statement.close();  
} catch (SQLException e) {  
    throw new RuntimeException(e);  
}  
}
```

¿Y ahora?

¡Enhorabuena! Has lanzado con éxito tu primera inserción a una base de datos empleando un conector Java.

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