Introducción a JDBC (Java DataBase Connectivity)

Java Database Connectivity (JDBC) es una API fundamental en el desarrollo de aplicaciones Java, ya que habilita la interacción con bases de datos relacionales. Esta interfaz proporciona una plataforma estándar para establecer conexiones, ejecutar consultas y administrar datos almacenados en una amplia gama de sistemas de gestión de bases de datos (SGBD), como MySQL, PostgreSQL, Oracle y SQL Server. JDBC desempeña un papel esencial al permitir que las aplicaciones Java accedan y manipulen datos en estas bases de datos de manera eficiente y segura. A continuación se muestran las características, comandos comunes y ejemplos de uso de JDBC para comprender cómo esta API facilita la interacción entre aplicaciones Java y bases de datos relacionales.

Características de JDBC

1. Conectividad

JDBC proporciona una interfaz para conectarse a diferentes bases de datos, lo que permite a las aplicaciones Java acceder a una amplia variedad de sistemas de gestión de bases de datos (SGBD) como MySQL, PostgreSQL, Oracle, SQL Server, entre otros.

2. Independencia de la base de datos

Una de las características más importantes de JDBC es su capacidad para ofrecer independencia de la base de datos. Esto significa que se puede escribir código Java que sea compatible con múltiples SGBD sin cambiarlo significativamente.

3. Acceso a datos

JDBC permite a las aplicaciones Java realizar operaciones CRUD (Crear, Leer, Actualizar y Borrar) en la base de datos. Puedes ejecutar consultas SQL y recuperar resultados en forma de conjuntos de datos (result sets).

4. Transacciones

JDBC es compatible con transacciones, lo que permite la ejecución de múltiples operaciones de base de datos como una unidad atómica. Puedes confirmar o revertir una serie de operaciones como una sola transacción.

Comandos Comunes de JDBC

1. DriverManager

El objeto **DriverManager** se utiliza para gestionar controladores de base de datos y para establecer la conexión con la base de datos. Te permite especificar la URL de conexión, nombre de usuario y contraseña.

2. Connection

La interfaz **Connection** representa la conexión activa con la base de datos. A través de ella, se pueden crear objetos **`Statement`** y **`PreparedStatement`**, declaraciones, ejecutar consultas y controlar transacciones a través de este objeto.

3. Statement

El objeto **Statement** se utiliza para ejecutar sentencias SQL. Hay dos tipos principales de objetos de este tipo: **Statement** estándar para sentencias estáticas y **PreparedStatement** para sentencias parametrizadas.

```
//  Statement estándar
Statement statement = connection.createStatement();
statement.executeUpdate("INSERT INTO tabla (columna) VALUES (valor)");
```

Ejemplo Statement

4. ResultSet

Cuando ejecutas una consulta, obtienes un objeto **ResultSet** que contiene los resultados. Se puede recorrer y procesar los datos en este objeto. Es importante cerrar el **ResultSet** cuando hayas terminado.

5. SQLException

SQLException es una excepción que se utiliza para manejar errores relacionados con la base de datos. Se deben capturar y gestionar estas excepciones para manejar errores de manera efectiva y obtener información detallada sobre los mismos.

Ejemplo de Uso de JDBC

A continuación, se muestra un ejemplo sencillo de cómo utilizar JDBC para conectarse a una base de datos y ejecutar una consulta

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class EjemploJDBC {
    public static void main(String[] args) {
        // Datos de conexión
        String url = "jdbc:mysql://localhost:3306/mi_base_de_datos";
        String usuario = "usuario";
        String contraseña = "contraseña";
        try {
            // Conexión a la base de datos
            Connection conexion = DriverManager.getConnection(url,
usuario, contraseña);

            // Crear una declaración
            Statement statement = conexion.createStatement();

            // Ejecutar una consulta
            String consulta = "SELECT * FROM tabla_ejemplo";
            ResultSet resultado = statement.executeQuery(consulta);

            // Procesar los resultados
            while (resultado.next()) {
                String dato = resultado.getString("nombre_columna");
                System.out.println("Dato: " + dato);
            }
         // Cerrar la conexión
            conexion.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

El código establece una conexión, crea una declaración, ejecuta una consulta para seleccionar registros de una tabla específica y luego procesa los resultados en un bucle, extrayendo datos de una columna y mostrándolos en la consola. Finalmente, se cierra la conexión a la base de datos. Este código es un ejemplo básico de cómo interactuar con bases de datos relacionales mediante JDBC en Java.