Sesión 04

JAVA DATABASE CONECTIVITY JDBC

Curso Virtual

Fundamentos de POO, Funcional y Reactivo

Ing. Aristedes Novoa

anovoa@galaxy.edu.pe

GALAXY TRAINING

ÍNDICE

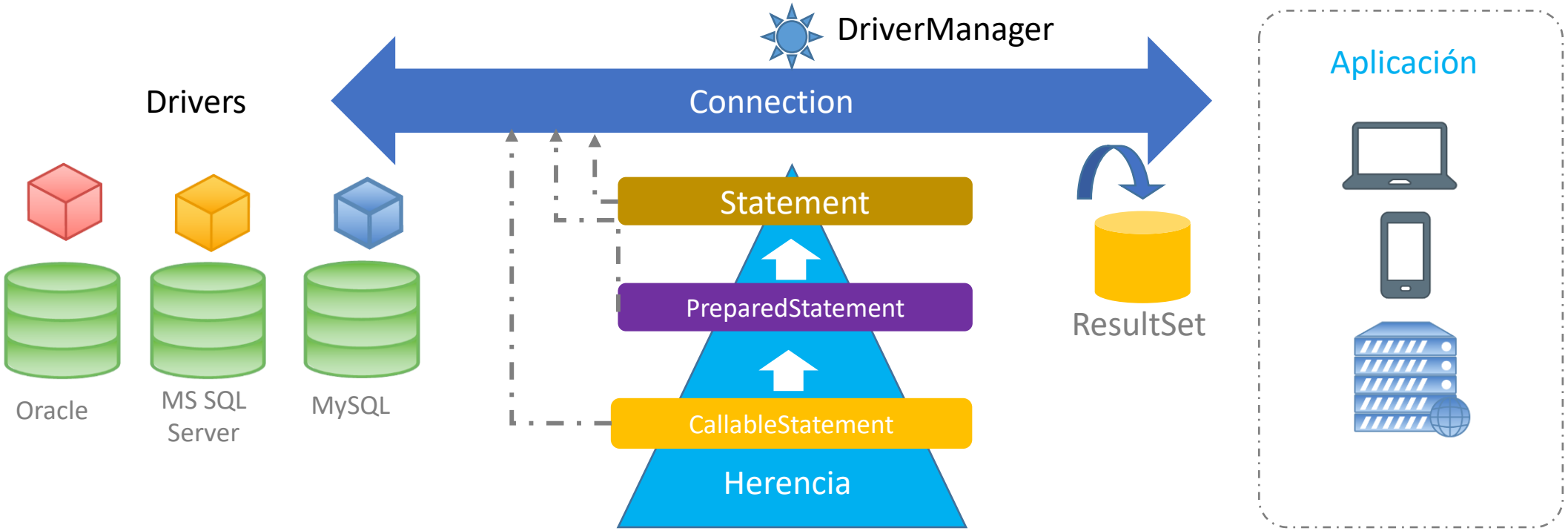
Para conectarnos a una base de datos y realizar operaciones sobre ella mediante el lenguaje SQL, Java nos brinda la API JDBC (Java Database Connectivity).

Esta API presenta una colección de clases e interfaces Java para las operaciones sobre la base de datos. Para ello, necesita un Driver JDBC (JAR) particular por cada base de datos (Oracle, MS SQL Server, MySQL, PostgreSQL, MariaDB).

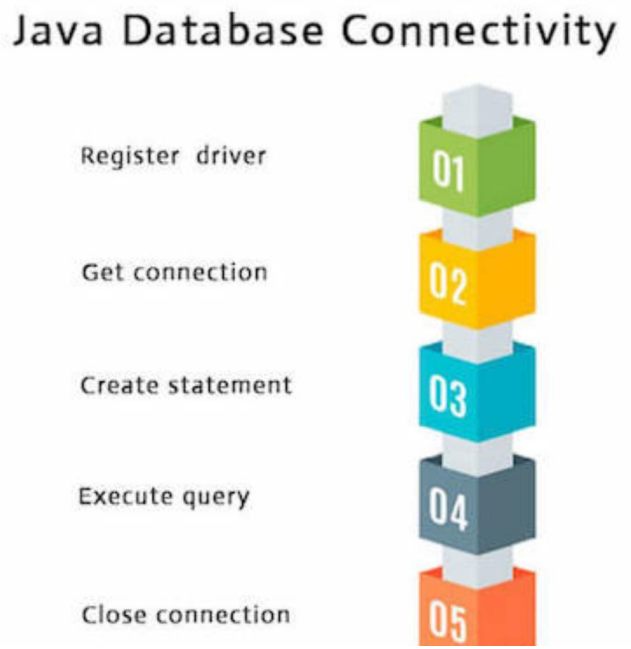
La API JDBC tiene las siguientes clases principales del paquete java.sql:

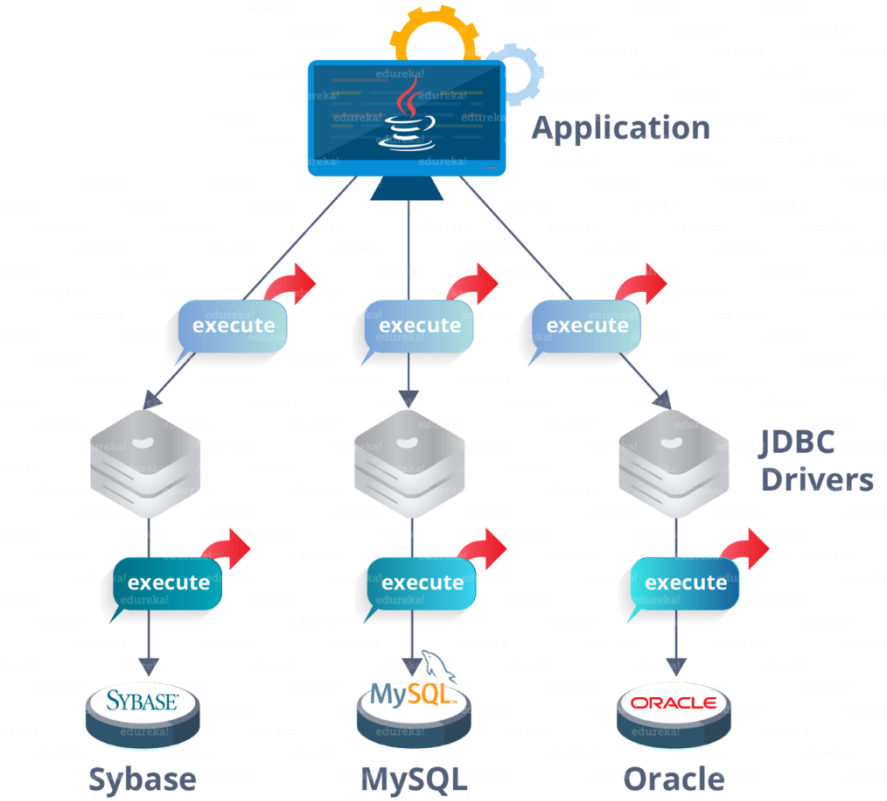
Clase	Descripción
DriverManager	Para cargar el Driver JDBC.
Connection	Para establecer la conexión con la base de datos.
Statement	Para ejecutar sentencias SQL sobre la base de datos.
PreparedStatement	Amplia la funcionalidad del Statement.
ResultSet	Para almacenar el resultado de la consulta.

JDBC- Principales clases e interfaces



Diseño conceptual - Conectividad





Diseño conceptual - Conectividad

Para conectarnos a una base de datos, necesitamos cargar el Driver y obtener la conexión.

Driver para Oracle DB

ojdbc6.jar

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

public class ConexionBD {

    public static void main(String[] args) {

        String url = "jdbc:oracle:thin:@localhost:1521:XE";
        String driver = "oracle.jdbc.OracleDriver";
        try {

            //1. Cargar el Driver
            Class.forName(driver);

            //2. Obtener la conexión a la BD
            Connection cn = DriverManager.getConnection(url, "user", "pass");

            System.out.println(cn);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

1. Se obtiene la conexión en la clase Connection.
2. Preparamos la consulta con el método **preparedStatement(sql)** de la conexión.
3. Ejecutamos la sentencia con el método **executeQuery**() del PreparedStatement.
4. Preguntamos si existe fila que recorrer con el método **next**() de la clase ResultSet.
5. Obtenemos el valor de cada columna (por nombre o índice) utilizando los métodos getXxx() por tipo de dato.

```
String sql = "SELECT ID, TITULO, RESUMEN,NROPAGINAS FROM LIBRO";

List<Libro> libros = new ArrayList<Libro>();

try {

    Connection cn = ConexionBD.getConnection(); 1

    PreparedStatement ps = cn.prepareStatement(sql); 2

    ResultSet rs = ps.executeQuery(); 3

    while (rs.next()) { 4

        Libro libro = new Libro();

        libro.setId(rs.getInt("id")); 5
        libro.setTitulo(rs.getString("titulo"));
        libro.setResumen(rs.getString("resumen"));
        libro.setNroPaginas(rs.getInt("nroPaginas"));

        libros.add(libro);

    }

} catch (Exception e) {
    System.err.println("Error al listar libros" + e.getMessage());

}
```

Consultar en la Base de datos

1. Se obtiene la conexión en la clase Connection.
2. Preparamos el insert con el método **preparedStatement(sql)** de la conexión.
3. Establecemos el valor de cada columna (por índice) utilizando los métodos setXxx() por tipo de dato.
4. Ejecutamos la sentencia con el método **executeUpdate**() del PreparedStatement.

```
String sql = "insert into libro (id, titulo, resumen, nroPaginas) values(?, ?, ?, ?)";

try {

    Connection cn = ConexionBD.getConnection();  1

    PreparedStatement ps = cn.prepareStatement(sql);  2

    ps.setInt(1, libro.getId());
    ps.setString(2, libro.getTitulo());
    ps.setString(3, libro.getResumen());
    ps.setInt(4, libro.getNroPaginas());  3

    ps.executeUpdate();  4

    return true;

} catch (SQLException e) {
    e.printStackTrace();
    return false;
}
```

Insertar en la Base de datos

1. Se obtiene la conexión en la clase Connection.
2. Preparamos el update con el método **preparedStatement(sql)** de la conexión.
3. Establecemos el valor de cada columna (por índice) utilizando los métodos setXxx() por tipo de dato.
4. Ejecutamos la sentencia con el método **executeUpdate**() del PreparedStatement.

```
String sql = "update libro set titulo = ?, resumen = ?, nropaginas = ? where id = ? ";

try {

    Connection cn = ConexionBD.getConnection();  1

    PreparedStatement ps = cn.prepareStatement(sql);  2

    ps.setString(1, libro.getTitulo());
    ps.setString(2, libro.getResumen());
    ps.setInt(3, libro.getNroPaginas());
    ps.setInt(4, libro.getId());  3

    ps.executeUpdate();  4

    return true;
} catch (Exception e) {
    System.err.println("Error al actualizar libro" + e.getMessage());
    return false;
}
```

Actualizar en la Base de datos

1. Se obtiene la conexión en la clase Connection.
2. Preparamos el update con el método **preparedStatement(sql)** de la conexión.
3. Establecemos el valor de cada columna (por índice) utilizando los métodos setXxx() por tipo de dato.
4. Ejecutamos la sentencia con el método **executeUpdate**() del PreparedStatement.


```
String sql = "delete libro where id = ? ";

try {

    Connection cn = ConexionBD.getConnection();  1

    PreparedStatement ps = cn.prepareStatement(sql);  2

    ps.setInt(1, libro.getId());  3

    ps.executeUpdate();  4

    return true;
} catch (Exception e) {
    System.err.println("Error al eliminar libro" + e.getMessage());
    return false;
}
```

Eliminar en la Base de datos

GALAXY

TRAINING