

50
1972 - 2022
AÑOS
Transformando vidas



UNIVERSIDAD DE
MANIZALES®



Acreditación Institucional
de Alta Calidad

Resolución 4792 del 15 de mayo de 2019
Sede Manizales por vigencia de 6 años (2019-2025)



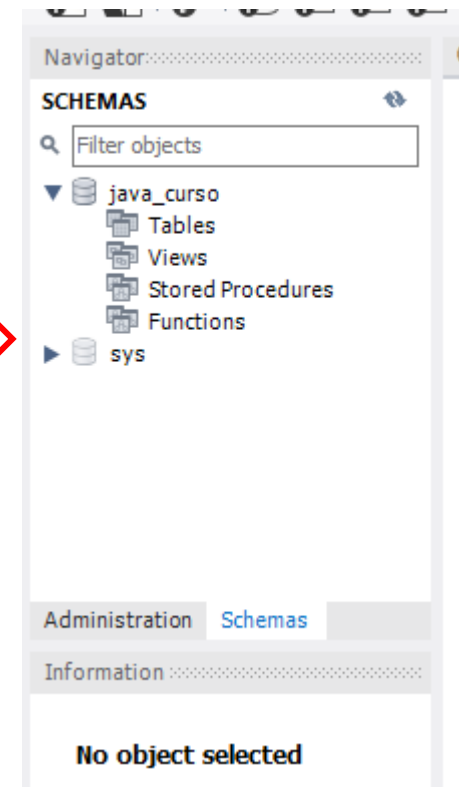
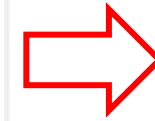
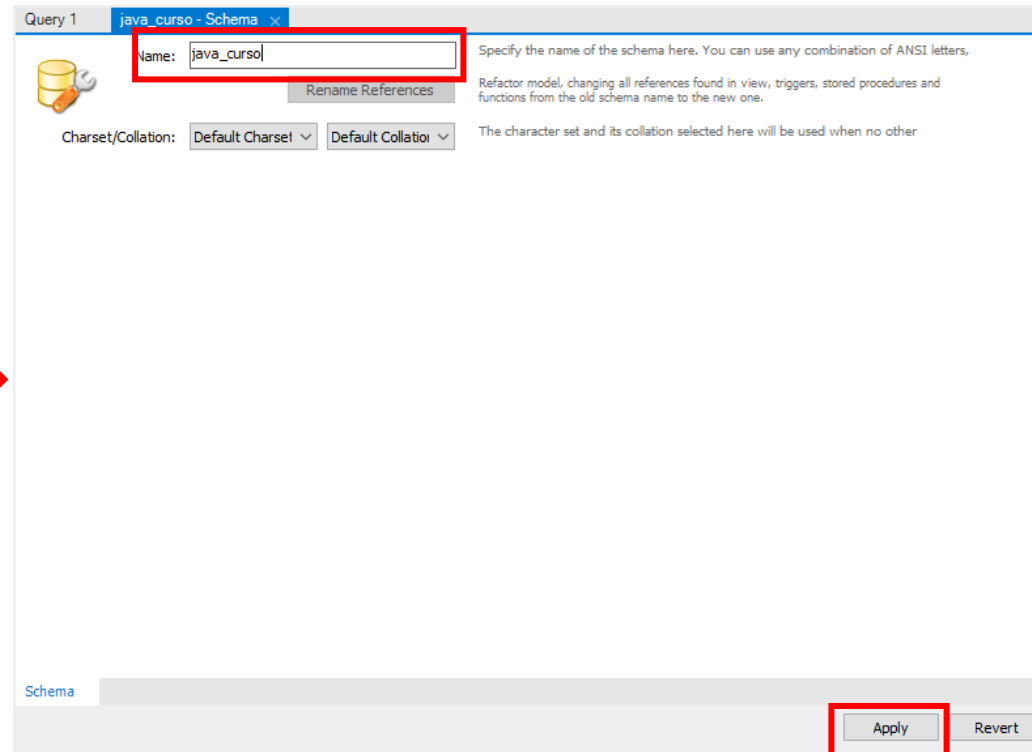
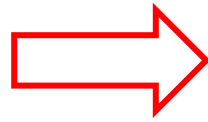
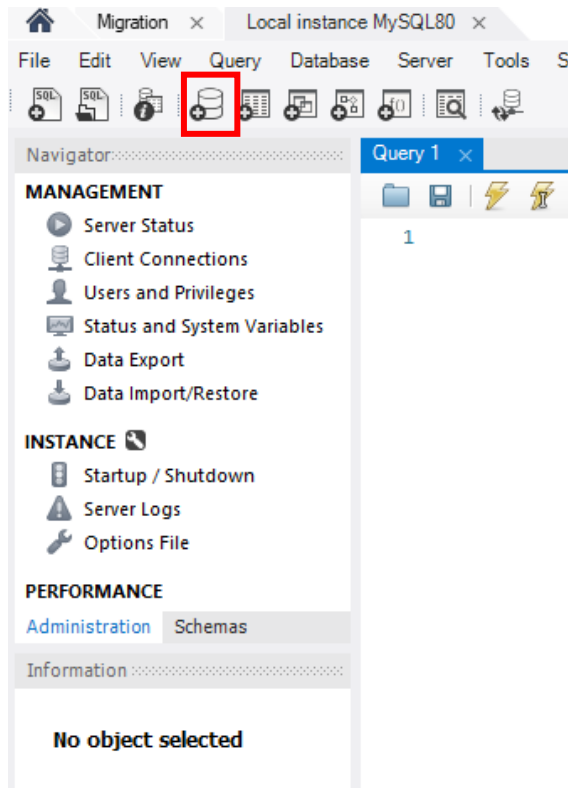
Diplomado

Programación Web Java JEE

Modalidad: Virtual

Duración: 80 horas

Bases de Datos



Bases de Datos

MySQL Workbench

Migration x Local instance MySQL80 x

File Edit View Query Database Server Tools

Navigator

SCHEMAS

Filter objects

java_curso

Load Spatial Data

Set as Default Schema

Filter to This Schema

Schema Inspector

Table Data Import Wizard

Copy to Clipboard

Send to SQL Editor

Create Schema...

Alter Schema...

Drop Schema...

Search Table Data...

Refresh All

Schema:

Query 1

java_c

Charset/Collation

Table Name: productos Schema: java_curso

Charset/Collation: Default Charset Default Collation Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
nombre	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
precio	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
fecha_registro	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name: Data Type:

Charset/Collation: Default Charset Default Collation

Default:

Storage: ☐ Virtual ☐ Stored

☐ Primary Key ☐ Not Null ☐ Unique

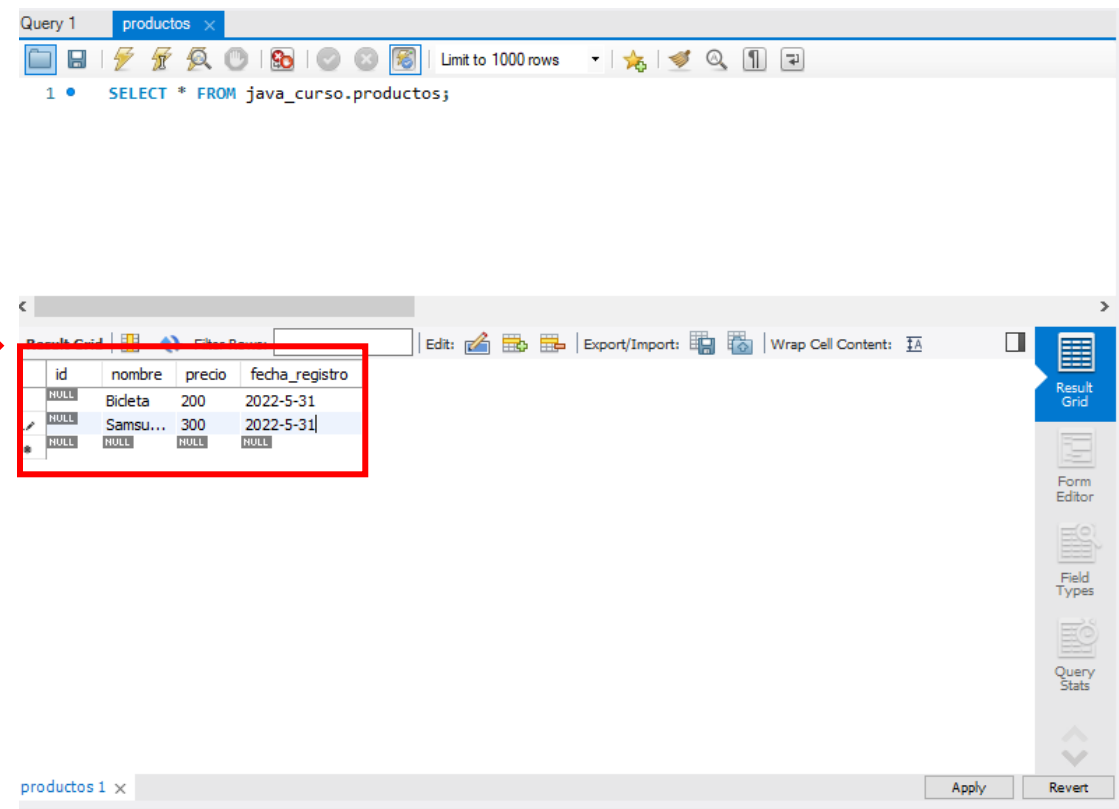
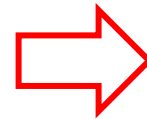
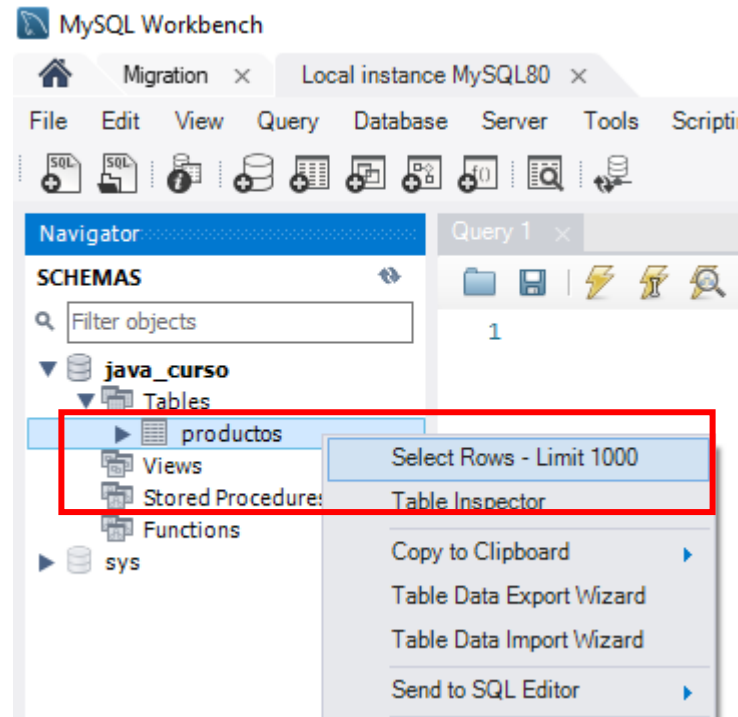
☐ Binary ☐ Unsigned ☐ Zero Fill

☐ Auto Increment ☐ Generated

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert

Bases de Datos



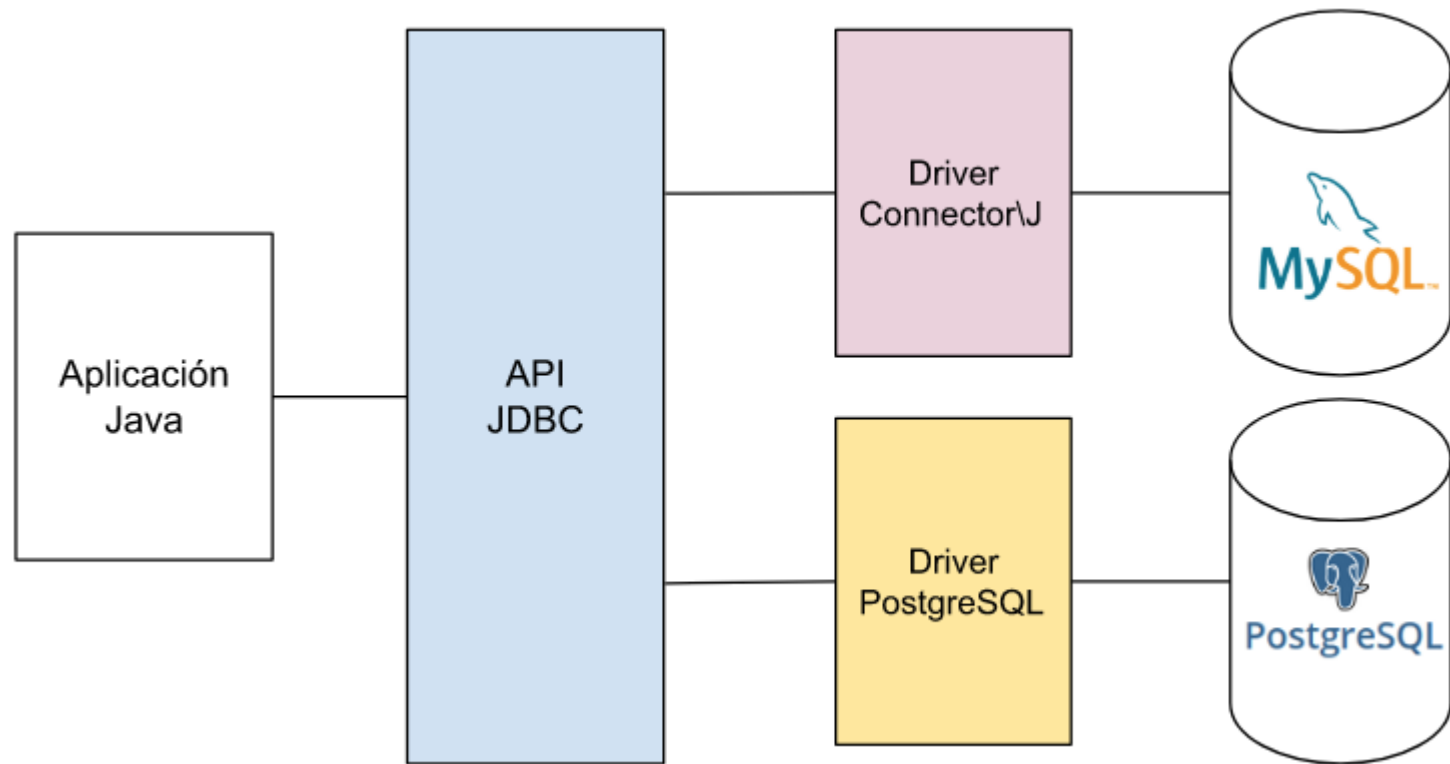


JDBC

JDBC stands for Java™ EE Database Connectivity. In Java EE development, this is a well known and commonly used technology for implementing database interaction. JDBC is a call-level API, meaning that SQL statements are passed as strings to the API, which then takes care of executing them on the RDMS. Consequently, the value of these strings can be changed at runtime, making JDBC dynamic.

<https://www.ibm.com/docs/en/developer-for-zos/9.5.1?topic=support-what-is-jdbc>

JDBC



JDBC

Clase/Interface	Descripción
Driver	Permite conectarse a una base de datos: cada gestor de base de datos requiere un driver distinto
DriverManager	Permite gestionar todos los drivers instalados en el sistema
Connection	Representa una conexión con una base de datos. Una aplicación puede tener más de una conexión a más de una base de datos
DatabaseMetadata	Proporciona información acerca de una Base de Datos, como las tablas que contiene, etc.
Statement	Permite ejecutar sentencias SQL sin parámetros
PreparedStatement	Permite ejecutar sentencias SQL con parámetros de entrada
CallableStatement	Permite ejecutar sentencias SQL con parámetros de entrada y salida, típicamente procedimientos almacenados
ResultSet	Contiene las filas o registros obtenidos al ejecutar una sentencia SELECT
ResultSetMetadata	Permite obtener información sobre un ResultSet, como el número de columnas, sus nombres, etc.

JDBC

```
String url = "jdbc:mysql://localhost:3306/nombre_db";
String username = "root";
String password = "sasa";

Connection conn = DriverManager.getConnection(url, username, password);
```

```
Statement stmt = conn.createStatement();
ResultSet resultado = stmt.executeQuery("select * from productos");
while( resultado.next() ){
    System.out.print(resultado.getInt("id"));
    System.out.print(" | ");
    System.out.print(resultado.getString("nombre"));
    System.out.print(" | ");
    System.out.print(resultado.getDouble("precio"));
    System.out.print(" | ");
    System.out.println(resultado.getDate("fecha"));
}
resultado.close();
stmt.close();
conn.close();
```

```
Statement stmt = conn.createStatement();
ResultSet resultado = stmt.executeQuery("select * from productos");
while( resultado.next() ){
    System.out.print(resultado.getInt(1));
    System.out.print(" | ");
    System.out.print(resultado.getString(2));
    System.out.print(" | ");
    System.out.print(resultado.getDouble(3));
    System.out.print(" | ");
    System.out.println(resultado.getDate(4));
}
resultado.close();
stmt.close();
conn.close();
```

```
PreparedStatement stmt = conn.prepareStatement("select * from productos WHERE id=?");
stmt.setInt(1, 4);

ResultSet resultado = stmt.executeQuery();
if( resultado.next() ){
    System.out.print(resultado.getInt("id"));
    System.out.print(" | ");
    System.out.print(resultado.getString("nombre"));
    System.out.print(" | ");
    System.out.print(resultado.getDouble("precio"));
    System.out.print(" | ");
    System.out.println(resultado.getDate("fecha"));
}
resultado.close();
stmt.close();
conn.close();
```

JDBC

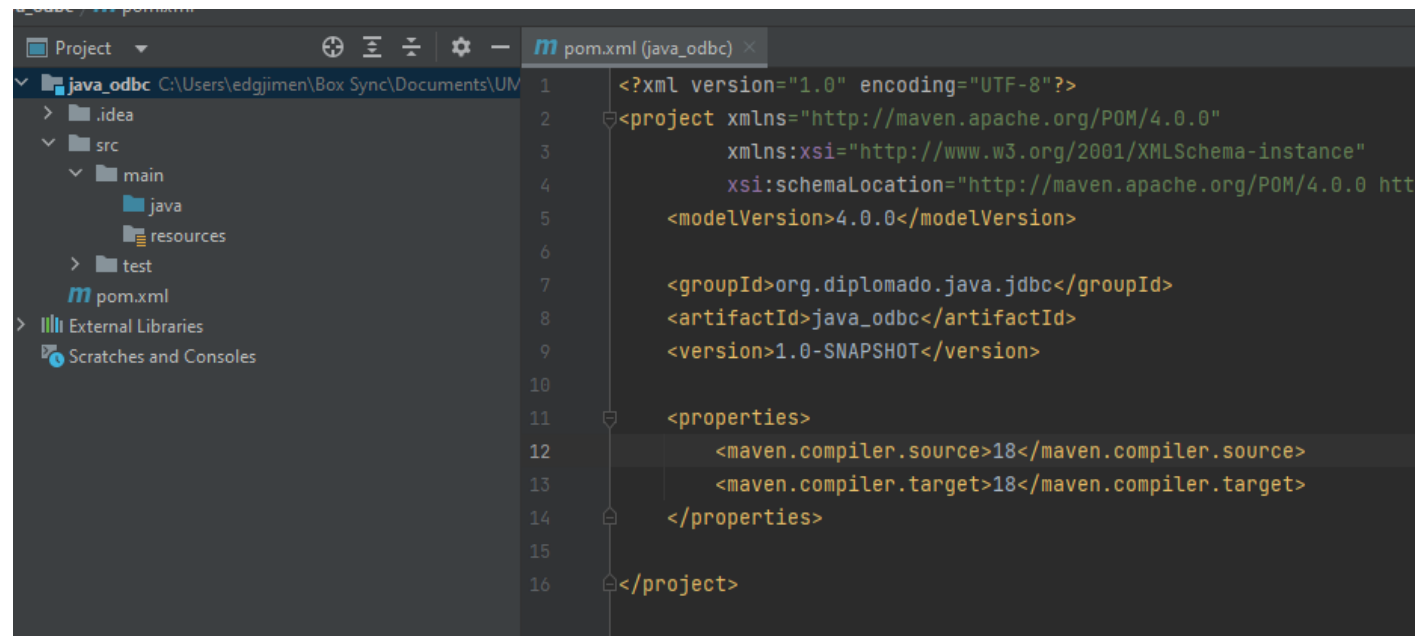
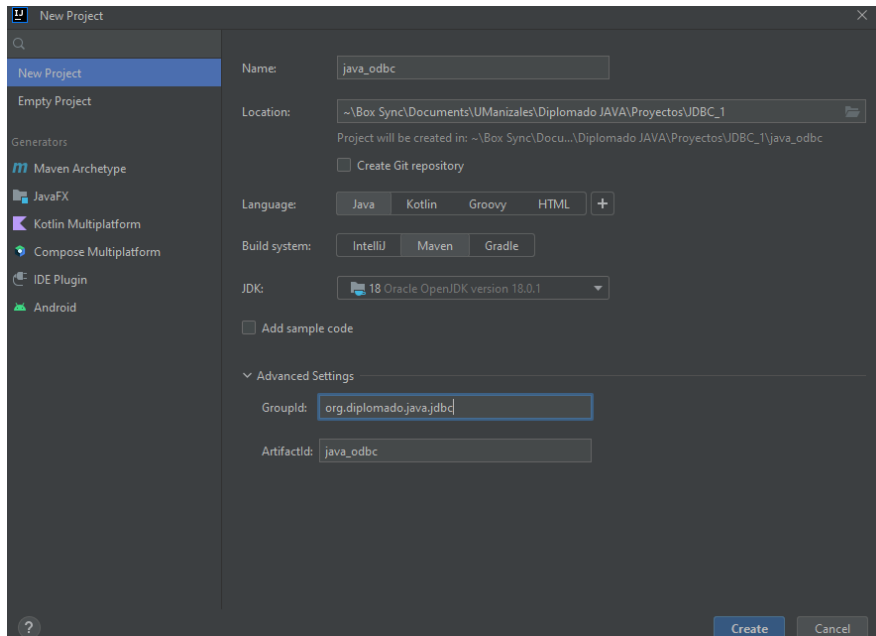
```
Statement stmt = conn.createStatement();  
  
int resultado = stmt.executeUpdate("INSERT INTO productos (nombre, precio, fecha) VALUES ('Bicicleta', 50, NOW())");  
  
int resultado = stmt.executeUpdate("UPDATE productos SET precio=59, nombre='Mountain Bike' WHERE id=7");  
  
int resultado = stmt.executeUpdate("DELETE FROM productos WHERE id=5");
```

```
PreparedStatement insertStmt = conn.prepareStatement("INSERT INTO productos (nombre, precio, fecha) VALUES (?, ?, ?)");  
insertStmt.setString(1, "Bicicleta");  
insertStmt.setDouble(2, 59.99);  
insertStmt.setDate(3, new Date(System.currentTimeMillis()));  
  
insertStmt.executeUpdate();
```

```
PreparedStatement updateStmt = conn.prepareStatement("UPDATE productos SET precio=?, nombre=? WHERE id=?");  
updateStmt.setDouble(1, 99.99);  
updateStmt.setString(2, "Mountain Bike");  
updateStmt.setInt(3, 7);  
  
updateStmt.executeUpdate();
```

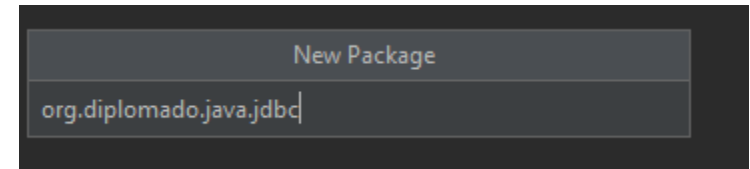
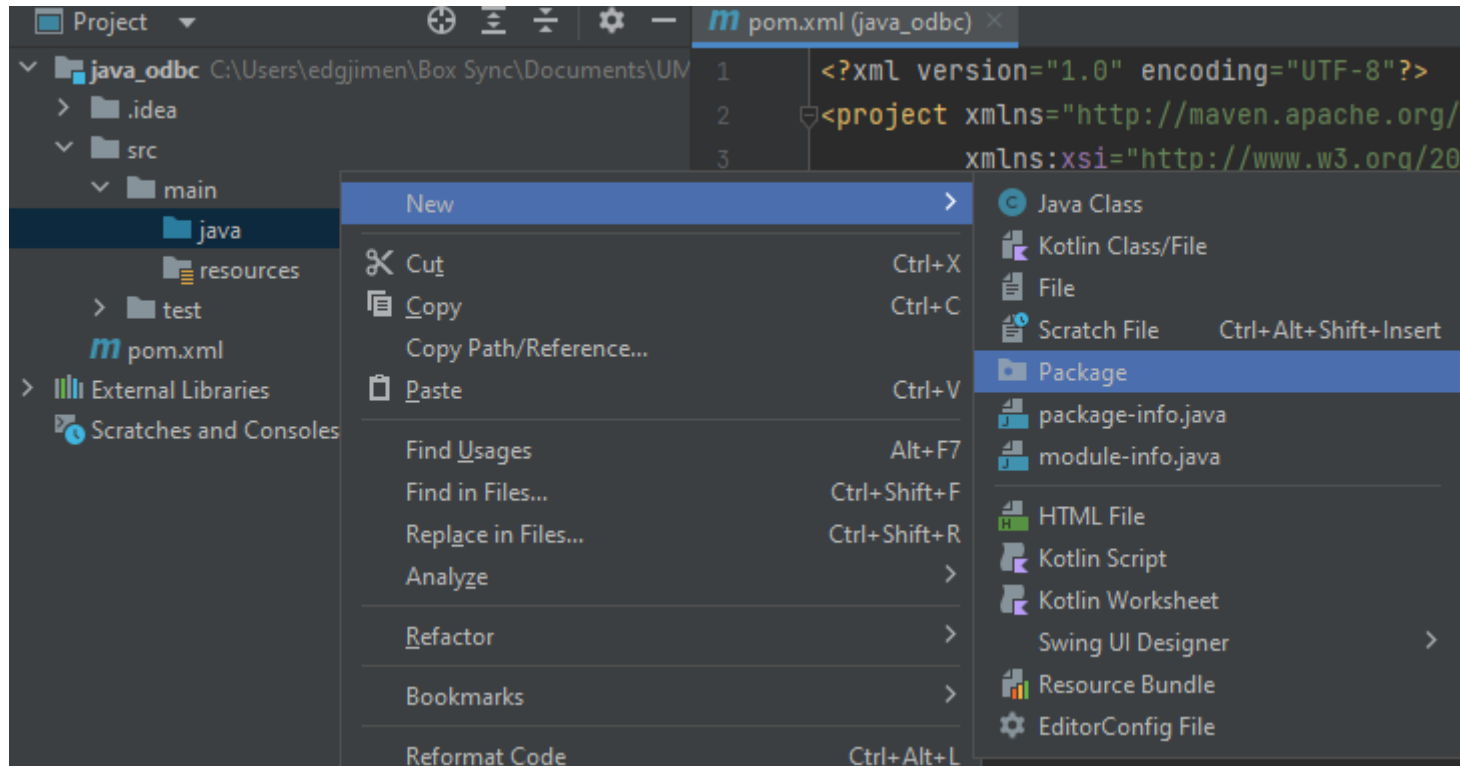
JDBC – Primer Proyecto

1. Crear un nuevo proyecto de tipo Maven para permitir agregar las librerías MySql



JDBC – Primer Proyecto

2. Creamos nuestro package



JDBC – Primer Proyecto

3. Agregamos la dependencia a las librerías de MySQL - <https://mvnrepository.com/artifact/mysql/mysql-connector-java>

← → ↺ <https://mvnrepository.com/artifact/mysql/mysql-connector-java/8.0.29>

MVNREPOSITORY Search for groups, artifacts, categories

Indexed Artifacts (28.2M)

Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities
- Dependency Injection

Home » mysql » mysql-connector-java » 8.0.29

MySQL Connector/J » 8.0.29
JDBC Type 4 driver for MySQL

License GPL 2.0

Categories MySQL Drivers

Organization Oracle Corporation

HomePage <http://dev.mysql.com/doc/connector-j/en/>

Date (Apr 25, 2022)

Files pom (2 KB) jar (2.4 MB) View All

Repositories Central

Used By 6,257 artifacts

Maven Gradle Gradle (Short) Gradle (Kotlin) SBT Ivy Grape Leiningen Buildr

```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.29</version>
</dependency>
```

☒ Include comment with link to declaration

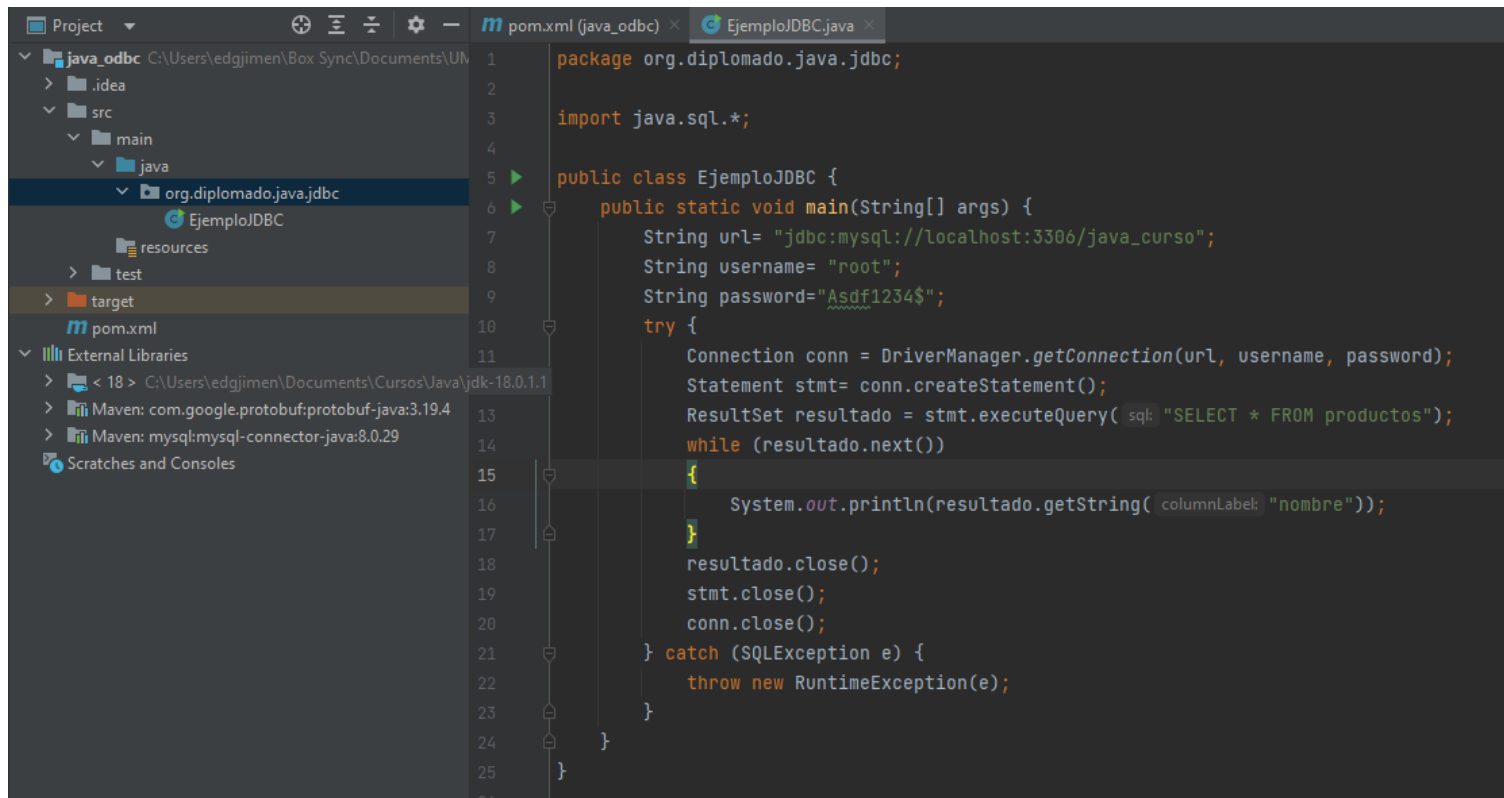
```
m pom.xml (java_odbc)
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
5   <modelVersion>4.0.0</modelVersion>
6
7   <groupId>org.diplomado.java.jdbc</groupId>
8   <artifactId>java_odbc</artifactId>
9   <version>1.0-SNAPSHOT</version>
10
11   <properties>
12     <maven.compiler.source>18</maven.compiler.source>
13     <maven.compiler.target>18</maven.compiler.target>
14   </properties>
15
16   <dependencies>
17     <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
18     <dependency>
19       <groupId>mysql</groupId>
20       <artifactId>mysql-connector-java</artifactId>
21       <version>8.0.29</version>
22     </dependency>
23   </dependencies>
```

java_odbc C:\Users\edgjimen\Box Sync\Documents\UN

- > .idea
- src
 - main
 - java
 - org.diplomado.java.jdbc
 - resources
 - test
- m pom.xml
- External Libraries
 - > < 18 > C:\Users\edgjimen\Documents\Cursos\Java\j
 - > Maven: com.google.protobuf:protobuf-java:3.19.4
 - > Maven: mysql:mysql-connector-java:8.0.29
- Scratches and Consoles

JDBC – Primer Proyecto

4. Agregamos clase MAIN.



The screenshot shows an IDE with the following components:

- Project View (Left):** Shows the project structure for 'java_odbc'. The 'src/main/java' directory contains the package 'org.diplomado.java.jdbc', which includes the 'EjemploJDBC' class. The 'pom.xml' file is also visible in the project structure.
- Code Editor (Right):** Displays the source code for 'EjemploJDBC.java'. The code is as follows:

```
1 package org.diplomado.java.jdbc;
2
3 import java.sql.*;
4
5 public class EjemploJDBC {
6     public static void main(String[] args) {
7         String url= "jdbc:mysql://localhost:3306/java_curso";
8         String username= "root";
9         String password="Asdf1234$";
10        try {
11            Connection conn = DriverManager.getConnection(url, username, password);
12            Statement stmt= conn.createStatement();
13            ResultSet resultado = stmt.executeQuery( sql: "SELECT * FROM productos");
14            while (resultado.next())
15            {
16                System.out.println(resultado.getString( columnLabel: "nombre"));
17            }
18            resultado.close();
19            stmt.close();
20            conn.close();
21        } catch (SQLException e) {
22            throw new RuntimeException(e);
23        }
24    }
25 }
```

JDBC – Primer Proyecto

5. Manejo de errores

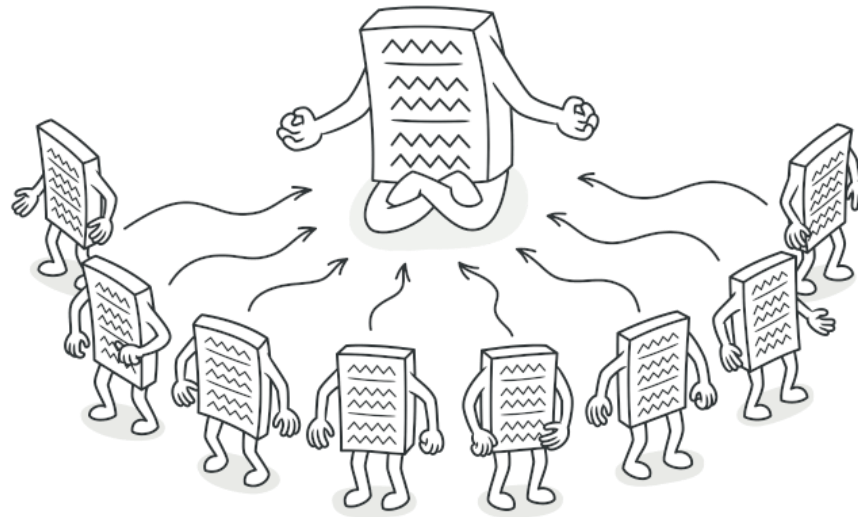
```
public class EjemploJDBC {  
    public static void main(String[] args) {  
        String url= "jdbc:mysql://localhost:3306/java_curso";  
        String username= "root";  
        String password="Asdf1234$";  
        try {  
            Connection conn = DriverManager.getConnection(url, username, password);  
            Statement stmt= conn.createStatement();  
            ResultSet resultado = stmt.executeQuery( sql: "SELECT * FROM productos2");  
            while (resultado.next())  
            {  
                System.out.println(resultado.getString( columnLabel: "nombre"));  
            }  
            resultado.close();  
            stmt.close();  
            conn.close();  
        } catch (SQLException e) {  
            throw new RuntimeException(e);  
        }  
    }  
}
```

Recursos
abiertos

```
public class EjemploJDBC {  
    public static void main(String[] args) {  
        String url= "jdbc:mysql://localhost:3306/java_curso";  
        String username= "root";  
        String password="Asdf1234$";  
        try {  
            Connection conn = DriverManager.getConnection(url, username, password);  
            Statement stmt= conn.createStatement();  
            ResultSet resultado = stmt.executeQuery( sql: "SELECT * FROM productos");  
            while (resultado.next())  
            {  
                System.out.print(resultado.getInt( columnLabel: "id"));  
                System.out.print(" | ");  
                System.out.print(resultado.getString( columnLabel: "nombre"));  
                System.out.print(" | ");  
                System.out.print(resultado.getInt( columnLabel: "precio"));  
                System.out.print(" | ");  
                System.out.println(resultado.getDate( columnLabel: "fecha_registro"));  
            }  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

JDBC – Primer Proyecto

5.Desacoplando conexión – Patrón Singleton

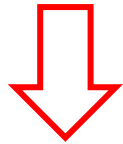


<https://refactoring.guru/design-patterns/singleton>

JDBC – Primer Proyecto

5.Desacoplando conexión – Patrón Singleton

New Package
org.diplomado.java.jdbc.util



New Java Class
ConexionBaseDatos
Class
Interface
Record
Enum
Annotation

```
package org.diplomado.java.jdbc.util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

2 usages
public class ConexionBaseDatos {
    1 usage
    private static String url = "jdbc:mysql://localhost:3306/java_curso?serverTimezone=America/Santiago";
    1 usage
    private static String username = "root";
    1 usage
    private static String password = "Asdf1234$";
    3 usages
    private static Connection connection;

    1 usage
    public static Connection getInstance() throws SQLException {
        if (connection == null) {
            connection = DriverManager.getConnection(url, username, password);
        }
        return connection;
    }
}
```

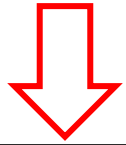
```
public class EjemploJDBC {
    public static void main(String[] args) {

        //Try con recurso para cerrar las conexiones automáticamente
        try (Connection conn = ConexionBaseDatos.getInstance();
            Statement stmt= conn.createStatement();
            ResultSet resultado = stmt.executeQuery("sql: \"SELECT * FROM productos\") {
            while (resultado.next())
            {
                System.out.print(resultado.getInt( columnLabel: "id"));
                System.out.print(" | ");
                System.out.print(resultado.getString( columnLabel: "nombre"));
                System.out.print(" | ");
                System.out.print(resultado.getInt( columnLabel: "precio"));
                System.out.print(" | ");
                System.out.println(resultado.getDate( columnLabel: "fecha_registro"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

JDBC – Primer Proyecto

6. Agregando Repositorio y Modelo (DAO)

New Package
org.diplomado.java.jdbc.modelo



New Java Class
Producto
Class
Interface
Record
Enum
Annotation

```
import java.util.Date;

public class Producto {
    4 usages
    private Long id;
    4 usages
    private String nombre;
    4 usages
    private Integer precio;
    4 usages
    private Date fechaRegistro;

    public Producto() {
    }

    public Producto(Long id, String nombre, Integer precio, Date fechaRegistro) {
        this.id = id;
        this.nombre = nombre;
        this.precio = precio;
        this.fechaRegistro = fechaRegistro;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
}
```

Alt+insert para
agregar set-get

```
public void setNombre(String nombre) {
    this.nombre = nombre;
}

public Integer getPrecio() {
    return precio;
}

public void setPrecio(Integer precio) {
    this.precio = precio;
}

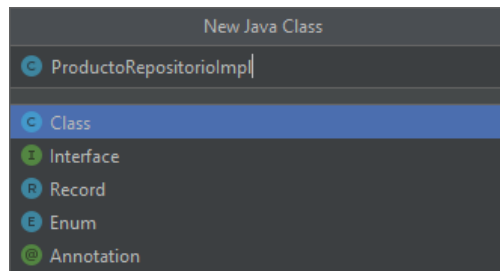
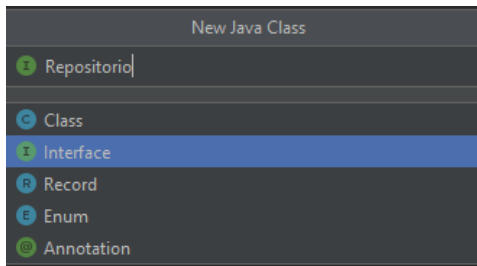
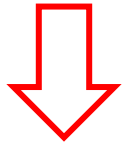
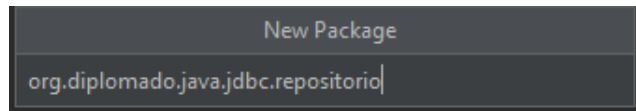
public Date getFechaRegistro() {
    return fechaRegistro;
}

public void setFechaRegistro(Date fechaRegistro) {
    this.fechaRegistro = fechaRegistro;
}

@Override
public String toString() {
    return id +
        " | " +
        nombre +
        " | " +
        precio +
        " | " +
        fechaRegistro;
}
}
```

JDBC – Primer Proyecto

6. Agregando Repositorio y Modelo (DAO)



```
package org.diplomado.java.jdbc.repositorio;

import java.util.List;

public interface Repositorio<T> {
    List<T> listar();
    T porId(Long id);
    void guardar(T t);
    void eliminar(Long id);
}
```

JDBC – Primer Proyecto

6. Agregando Repositorio y Modelo (DAO)

```
1 package org.diplomado.java.jdbc.repositorio;
2
3 import org.diplomado.java.jdbc.modelo.Producto;
4 import org.diplomado.java.jdbc.util.ConexionBaseDatos;
5 import java.sql.*;
6 import java.util.ArrayList;
7 import java.util.List;
8
9 public class ProductoRepositorioImpl implements Repositorio<Producto>{
10     4 usages
11     private Connection getConnection() throws SQLException {
12         return ConexionBaseDatos.getInstance();
13     }
14
15     @Override
16     public List<Producto> listar() {
17         List<Producto> productos = new ArrayList<>();
18
19         try (Statement stmt = getConnection().createStatement();
20              ResultSet rs = stmt.executeQuery( sql: "SELECT * FROM productos")) {
21             while (rs.next()) {
22                 Producto p = crearProducto(rs);
23                 productos.add(p);
24             }
25         } catch (SQLException e) {
26             e.printStackTrace();
27         }
28         return productos;
29     }
```

```
30
31     @Override
32     public Producto porId(Long id) {
33         Producto producto = null;
34
35         try (PreparedStatement stmt = getConnection().
36              prepareStatement( sql: "SELECT * FROM productos WHERE id = ?")) {
37             stmt.setLong( parameterIndex: 1, id);
38             try (ResultSet rs = stmt.executeQuery()) {
39                 if (rs.next()) {
40                     producto = crearProducto(rs);
41                 }
42             }
43         } catch (SQLException e) {
44             e.printStackTrace();
45         }
46         return producto;
47     }
48
49     @Override
50     public void guardar(Producto producto) {
51         String sql;
52         if (producto.getId() != null && producto.getId() > 0) {
53             sql = "UPDATE productos SET nombre=?, precio=? WHERE id=?";
54         } else {
55             sql = "INSERT INTO productos(nombre, precio, fecha_registro) VALUES(?,?,?)";
56         }
57         try (PreparedStatement stmt = getConnection().prepareStatement(sql)) {
58             stmt.setString( parameterIndex: 1, producto.getNombre());
59             stmt.setLong( parameterIndex: 2, producto.getPrecio());
60
61             if (producto.getId() != null && producto.getId() > 0) {
62                 stmt.setLong( parameterIndex: 3, producto.getId());
63             } else {
64                 stmt.setDate( parameterIndex: 3, new Date(producto.getFechaRegistro().getTime()));
65             }
66             stmt.executeUpdate();
67         } catch (SQLException throwables) {
68             throwables.printStackTrace();
69         }
70     }
```

```
71
72     @Override
73     public void eliminar(Long id) {
74         try (PreparedStatement stmt = getConnection().prepareStatement( sql: "DELETE FROM productos WHERE id=?")) {
75             stmt.setLong( parameterIndex: 1, id);
76             stmt.executeUpdate();
77         } catch (SQLException throwables) {
78             throwables.printStackTrace();
79         }
80
81     2 usages
82     @ private Producto crearProducto(ResultSet rs) throws SQLException {
83         Producto p = new Producto();
84         p.setId(rs.getLong( columnLabel: "id"));
85         p.setNombre(rs.getString( columnLabel: "nombre"));
86         p.setPrecio(rs.getInt( columnLabel: "precio"));
87         p.setFechaRegistro(rs.getDate( columnLabel: "fecha_registro"));
88         return p;
89     }
```

JDBC – Primer Proyecto

6. Agregando Repositorio y Modelo (DAO)

```
package org.diplomado.java.jdbc;

import org.diplomado.java.jdbc.modelo.Producto;
import org.diplomado.java.jdbc.repositorio.ProductoRepositorioImpl;
import org.diplomado.java.jdbc.repositorio.Repositorio;
import org.diplomado.java.jdbc.util.ConexionBaseDatos;

import java.sql.*;
import java.util.Date;

public class EjemploJDBC {
    public static void main(String[] args) {
        try (Connection conn = ConexionBaseDatos.getInstance()) {

            Repositorio<Producto> repositorio = new ProductoRepositorioImpl();
            System.out.println("===== listar =====");
            repositorio.listar().forEach(System.out::println);

            System.out.println("===== obtener por id =====");
            System.out.println(repositorio.findById(1L));

            System.out.println("===== insertar nuevo producto =====");
            Producto producto = new Producto();
            producto.setNombre("Teclado mecánico");
            producto.setPrecio(500);
            producto.setFechaRegistro(new Date());
            repositorio.guardar(producto);
            System.out.println("Producto guardado con éxito");
            repositorio.listar().forEach(System.out::println);

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

```
package org.diplomado.java.jdbc;

import org.diplomado.java.jdbc.modelo.Producto;
import org.diplomado.java.jdbc.repositorio.ProductoRepositorioImpl;
import org.diplomado.java.jdbc.repositorio.Repositorio;
import org.diplomado.java.jdbc.util.ConexionBaseDatos;

import java.sql.Connection;
import java.sql.SQLException;

public class EjemploJdbcUpdate {
    public static void main(String[] args) {
        try (Connection conn = ConexionBaseDatos.getInstance()) {

            Repositorio<Producto> repositorio = new ProductoRepositorioImpl();
            System.out.println("===== listar =====");
            repositorio.listar().forEach(System.out::println);

            System.out.println("===== obtener por id =====");
            System.out.println(repositorio.findById(1L));

            System.out.println("===== editar producto =====");
            Producto producto = new Producto();
            producto.setId(3L);
            producto.setNombre("Teclado Razer mecánico");
            producto.setPrecio(700);
            repositorio.guardar(producto);
            System.out.println("Producto editado con éxito");
            repositorio.listar().forEach(System.out::println);

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

```
package org.diplomado.java.jdbc;

import org.diplomado.java.jdbc.modelo.Producto;
import org.diplomado.java.jdbc.repositorio.ProductoRepositorioImpl;
import org.diplomado.java.jdbc.repositorio.Repositorio;
import org.diplomado.java.jdbc.util.ConexionBaseDatos;

import java.sql.Connection;
import java.sql.SQLException;

public class EjemploJdbcDelete {
    public static void main(String[] args) {
        try (Connection conn = ConexionBaseDatos.getInstance()) {

            Repositorio<Producto> repositorio = new ProductoRepositorioImpl();
            System.out.println("===== listar =====");
            repositorio.listar().forEach(System.out::println);

            System.out.println("===== obtener por id =====");
            System.out.println(repositorio.findById(1L));

            System.out.println("===== Eliminar producto =====");
            repositorio.eliminar(3L);
            System.out.println("Producto eliminado con éxito");
            repositorio.listar().forEach(System.out::println);

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

JDBC – Primer Proyecto

7. Relación entre tablas

Table Name: Schema: **java_curso**

Charset/Collation: Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
nombre	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name: Data Type:

Charset/Collation: Expression:

Comments:

Storage: ☐ Virtual ☐ Stored

☐ Primary Key ☐ Not Null ☐ Unique

☐ Binary ☐ Unsigned ☐ Zero Fill

☐ Auto Increment ☐ Generated

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert

Table Name: Schema: **java_curso**

Charset/Collation: Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
nombre	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
precio	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
fecha_registro	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
categoria_id	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name: Data Type:

Charset/Collation: Expression:

Comments:

Storage: ☐ Virtual ☐ Stored

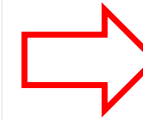
☐ Primary Key ☐ Not Null ☐ Unique

☐ Binary ☐ Unsigned ☐ Zero Fill

☐ Auto Increment ☐ Generated

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert



Query 1 productos - Table

Table Name: Schema: **java_curso**

Charset/Collation: Engine:

Comments:

Foreign Key Name	Referenced Table	Column	Referenced Column	Foreign Key Options
productos_categoria	java_curso.'categorias'	<input checked="" type="checkbox"/> categoria_id	<input checked="" type="checkbox"/> id	On Update: <input type="text" value="NO ACTION"/> On Delete: <input type="text" value="NO ACTION"/> <input type="checkbox"/> Skip in SQL generation

Foreign Key Comment:

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert

JDBC – Primer Proyecto

7. Relación entre tablas

```
public class Categoria {  
    2 usages  
    private Long id;  
    2 usages  
    private String nombre;  
  
    public Long getId() {  
        return id;  
    }  
  
    public void setId(Long id) {  
        this.id = id;  
    }  
  
    public String getNombre() {  
        return nombre;  
    }  
  
    public void setNombre(String nombre) {  
        this.nombre = nombre;  
    }  
}
```

```
public class Producto {  
    4 usages  
    private Long id;  
    4 usages  
    private String nombre;  
    4 usages  
    private Integer precio;  
    4 usages  
    private Date fechaRegistro;  
    2 usages  
    private Categoria categoria;  
  
    public Categoria getCategoria() {  
        return categoria;  
    }  
  
    public void setCategoria(Categoria categoria) {  
        this.categoria = categoria;  
    }  
}
```

```
@Override  
public String toString() {  
    return id +  
        " | " +  
        nombre +  
        " | " +  
        precio +  
        " | " +  
        fechaRegistro + " | "  
        + categoria.getNombre();  
}
```

JDBC – Primer Proyecto

7. Relación entre tablas

```
3 import org.diplomado.java.jdbc.modelo.Categoria;
4 import org.diplomado.java.jdbc.modelo.Producto;
5 import org.diplomado.java.jdbc.util.ConexionBaseDatos;
6 import java.sql.*;
7 import java.util.ArrayList;
8 import java.util.List;
9
10 6 usages
11 public class ProductoRepositorioImpl implements Repositorio<Producto>{
12
13 4 usages
14     private Connection getConnection() throws SQLException {
15         return ConexionBaseDatos.getInstance();
16     }
17
18 6 usages
19     @Override
20     public List<Producto> listar() {
21         List<Producto> productos = new ArrayList<>();
22
23         try (Statement stmt = getConnection().createStatement();
24              ResultSet rs = stmt.executeQuery( sql: "SELECT p.*, c.nombre as categoria FROM productos as p " +
25              "inner join categorias as c ON (p.categoria_id = c.id)")) {
26             while (rs.next()) {
27                 Producto p = crearProducto(rs);
28                 productos.add(p);
29             }
30         } catch (SQLException e) {
31             e.printStackTrace();
32         }
33         return productos;
34     }
35 }
```

```
33 @Override
34 public Producto porId(Long id) {
35     Producto producto = null;
36
37     try (PreparedStatement stmt = getConnection().
38         prepareStatement( sql: "SELECT p.*, c.nombre as categoria FROM productos as p " +
39         "inner join categorias as c ON (p.categoria_id = c.id) WHERE p.id = ?")) {
40         stmt.setLong( parameterIndex: 1, id);
41         try (ResultSet rs = stmt.executeQuery()) {
42             if (rs.next()) {
43                 producto = crearProducto(rs);
44             }
45         } catch (SQLException e) {
46             e.printStackTrace();
47         }
48         return producto;
49     }
50
51 2 usages
52 @Override
53 public void guardar(Producto producto) {
54     String sql;
55     if (producto.getId() != null && producto.getId() > 0) {
56         sql = "UPDATE productos SET nombre=?, precio=?, categoria_id=? WHERE id=?";
57     } else {
58         sql = "INSERT INTO productos(nombre, precio, categoria_id, fecha_registro) VALUES(?, ?, ?, ?)";
59     }
60     try (PreparedStatement stmt = getConnection().prepareStatement(sql)) {
61         stmt.setString( parameterIndex: 1, producto.getNombre());
62         stmt.setLong( parameterIndex: 2, producto.getPrecio());
63         stmt.setLong( parameterIndex: 3, producto.getCategoria().getId());
64
65         if (producto.getId() != null && producto.getId() > 0) {
66             stmt.setLong( parameterIndex: 4, producto.getId());
67         } else {
68             stmt.setLong( parameterIndex: 4, producto.getId());
69         }
70     }
71 }
```

```
67     } else {
68         stmt.setDate( parameterIndex: 4, new Date(producto.getFechaRegistro().getTime()));
69     }
70
71     stmt.executeUpdate();
72 } catch (SQLException throwables) {
73     throwables.printStackTrace();
74 }
75
76 }
77
78 1 usage
79 @Override
80 public void eliminar(Long id) {
81     try (PreparedStatement stmt = getConnection().prepareStatement( sql: "DELETE FROM productos WHERE id=?")) {
82         stmt.setLong( parameterIndex: 1, id);
83         stmt.executeUpdate();
84     } catch (SQLException throwables) {
85         throwables.printStackTrace();
86     }
87
88 2 usages
89 private Producto crearProducto(ResultSet rs) throws SQLException {
90     Producto p = new Producto();
91     p.setId(rs.getLong( columnLabel: "id"));
92     p.setNombre(rs.getString( columnLabel: "nombre"));
93     p.setPrecio(rs.getInt( columnLabel: "precio"));
94     p.setFechaRegistro(rs.getDate( columnLabel: "fecha_registro"));
95     Categoria categoria = new Categoria();
96     categoria.setId(rs.getLong( columnLabel: "categoria_id"));
97     categoria.setNombre(rs.getString( columnLabel: "categoria"));
98     p.setCategoria(categoria);
99     return p;
100 }
```


JDBC – Primer Proyecto

7. Relación entre tablas

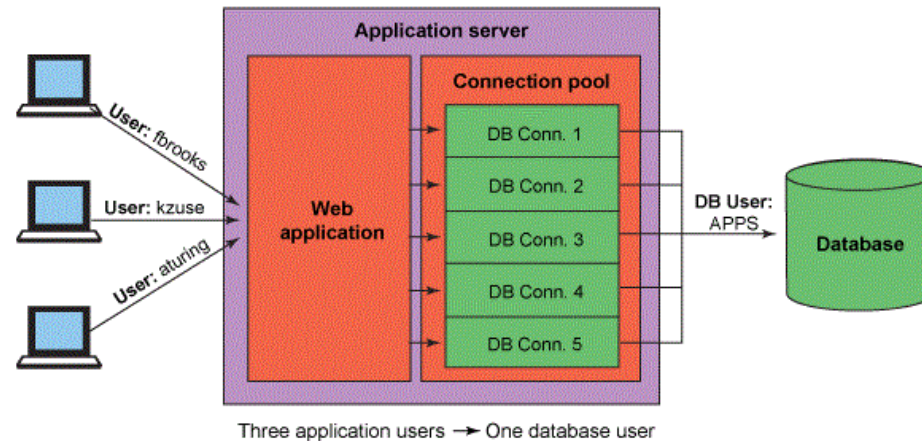
```
EjemploJDBC.java x
16 Repositorio<Producto> repositorio = new ProductoRepositorioImpl();
17 System.out.println("===== listar =====");
18 repositorio.listar().forEach(System.out::println);
19
20 System.out.println("===== obtener por id =====");
21 System.out.println(repositorio.porId(1L));
22
23 System.out.println("===== insertar nuevo producto =====");
24 Producto producto = new Producto();
25 producto.setNombre("Teclado Razer mecánico");
26 producto.setPrecio(550);
27 producto.setFechaRegistro(new Date());
28 Categoria categoria = new Categoria();
29 categoria.setId(3L);
30 producto.setCategoria(categoria);
31 repositorio.guardar(producto);
32 System.out.println("Producto guardado con éxito");
33 repositorio.listar().forEach(System.out::println);
```

```
EjemploJdbcUpdate.java x
16 Repositorio<Producto> repositorio = new ProductoRepositorioImpl();
17 System.out.println("===== listar =====");
18 repositorio.listar().forEach(System.out::println);
19
20 System.out.println("===== obtener por id =====");
21 System.out.println(repositorio.porId(1L));
22
23 System.out.println("===== editar producto =====");
24 Producto producto = new Producto();
25 producto.setId(5L);
26 producto.setNombre("Teclado Cosair k95 mecánico");
27 producto.setPrecio(700);
28 Categoria categoria = new Categoria();
29 categoria.setId(2L);
30 producto.setCategoria(categoria);
31 repositorio.guardar(producto);
32 System.out.println("Producto editado con éxito");
33 repositorio.listar().forEach(System.out::println);
34
```

JDBC – Primer Proyecto

8. Pool de conexiones

Database Connection Pool for Java web app



<https://javarevisited.blogspot.com/2018/07/how-to-setup-jndi-database-connection-pool-tomcat-spring-example-tutorial.html#axzz7V4T4OISs>

JDBC – Primer Proyecto

8. Pool de conexiones

https://mvnrepository.com/artifact/org.apache.commons/commons-dbcp2/2.9.0

Search for groups, artifacts, categories

(28.2M)

Home » org.apache.commons » commons-dbcp2 » 2.9.0

Apache Commons DBCP » 2.9.0

Apache Commons DBCP software implements Database Connection Pooling

License	Apache 2.0
Categories	JDBC Pools
HomePage	https://commons.apache.org/dbcp/
Date	(Aug 04, 2021)
Files	pom (19 KB) jar (206 KB) View All
Repositories	Central
Used By	827 artifacts
Vulnerabilities	Vulnerabilities from dependencies: CVE-2022-23221 CVE-2021-42392 CVE-2021-23463

Maven Gradle Gradle (Short) Gradle (Kotlin) SBT Ivy Grape Leiningen Buildr

```
<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-dbcp2 -->
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-dbcp2</artifactId>
  <version>2.9.0</version>
</dependency>
```

```
m pom.xml (java_odbc) x
13      <maven.compiler.target>18</maven.compiler.target>
14    </properties>
15
16    <dependencies>
17      <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
18      <dependency>
19        <groupId>mysql</groupId>
20        <artifactId>mysql-connector-java</artifactId>
21        <version>8.0.29</version>
22      </dependency>
23      <!-- https://mvnrepository.com/artifact/org.apache.commons/commons-dbcp2 -->
24      <dependency>
25        <groupId>org.apache.commons</groupId>
26        <artifactId>commons-dbcp2</artifactId>
27        <version>2.9.0</version>
28      </dependency>
29    </dependencies>
30
31  </project>
```

JDBC – Primer Proyecto

8. Pool de conexiones

```
public class ConexionBaseDatos {  
    1 usage  
    private static String url = "jdbc:mysql://localhost:3306/java_curso";  
    1 usage  
    private static String username = "root";  
    1 usage  
    private static String password = "Asdf1234$";  
    10 usages  
    private static BasicDataSource pool;  
  
    1 usage  
    public static BasicDataSource getInstance() throws SQLException {  
        if (pool == null) {  
            pool = new BasicDataSource();  
            pool.setUrl(url);  
            pool.setUsername(username);  
            pool.setPassword(password);  
            pool.setInitialSize(3);  
            pool.setMinIdle(3);  
            pool.setMaxIdle(8);  
            pool.setMaxTotal(8);  
        }  
        return pool;  
    }  
  
    1 usage  
    public static Connection getConnection() throws SQLException {  
        return getInstance().getConnection();  
    }  
}
```

```
public class ProductoRepositorioImpl implements Repositorio<Producto>{  
    4 usages  
    private Connection getConnection() throws SQLException {  
        return ConexionBaseDatos.getConnection();  
    }  
}
```



JDBC – Primer Proyecto

9. Transacciones

Una transacción es un conjunto de operaciones sobre una base de datos que se deben ejecutar como una unidad.

<http://puntocomnoesunlenguaje.blogspot.com/2017/11/java-jdbc-transacciones.html>

JDBC – Primer Proyecto

9. Transacciones

Table Name: Schema: **java_curso**

Charset/Collation: Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
nombre	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
precio	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
fecha_registro	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
categoria_id	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
sku	VARCHAR(10)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name: Data Type:

Charset/Collation:

Default:

Comments:

Storage: ☐ Virtual ☐ Stored

☐ Primary Key ☐ Not Null ☒ Unique

☐ Binary ☐ Unsigned ☐ Zero Fill

☐ Auto Increment ☐ Generated

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert

```
26 usages
public class Producto {
    4 usages
    private Long id;
    4 usages
    private String nombre;
    4 usages
    private Integer precio;
    4 usages
    private Date fechaRegistro;
    3 usages
    private Categoria categoria;
    2 usages
    private String sku;

    1 usage
    public String getSKU() {
        return sku;
    }

    1 usage
    public void setSKU(String sku) {
        this.sku = sku;
    }
}
```

```
@Override
public String toString() {
    return id +
        " | " +
        nombre +
        " | " +
        precio +
        " | " +
        fechaRegistro + " | " +
        categoria.getNombre() +
        " | " + sku;
}
```

JDBC – Primer Proyecto

9. Transacciones

```
package org.diplomado.java.jdbc.repositorio;

import java.sql.SQLException;
import java.util.List;

7 usages 2 implementations
public interface Repositorio<T> {
    2 usages 2 implementations
    List<T> listar() throws SQLException;

    1 usage 2 implementations
    T porId(Long id) throws SQLException;

    2 usages 2 implementations
    T guardar(T t) throws SQLException;

    2 implementations
    void eliminar(Long id) throws SQLException;
}
```

JDBC – Primer Proyecto

9. Transacciones

```
1 package org.diplomado.java.jdbc.repositorio;
2
3 import org.diplomado.java.jdbc.modelo.Categoria;
4 import org.diplomado.java.jdbc.modelo.Producto;
5
6 import java.sql.*;
7 import java.util.ArrayList;
8 import java.util.List;
9
10 4 usages
11 public class ProductoRepositorioImpl implements Repositorio<Producto>{
12
13     5 usages
14     private Connection conn;
15
16     1 usage
17     public ProductoRepositorioImpl(Connection conn) {
18         this.conn = conn;
19     }
20
21     2 usages
22     @Override
23     public List<Producto> listar() throws SQLException {
24         List<Producto> productos = new ArrayList<>();
25
26         try (Statement stmt = conn.createStatement()) {
27             ResultSet rs = stmt.executeQuery( "SELECT p.*, c.nombre as categoria FROM productos as p " +
28                 "inner join categorias as c ON (p.categoria_id = c.id) WHERE p.id = ?" );
29             while (rs.next()) {
30                 Producto p = crearProducto(rs);
31                 productos.add(p);
32             }
33         }
34
35         return productos;
36     }
37 }
```

```
38 @Override
39 public Producto porId(Long id) throws SQLException {
40     Producto producto = null;
41
42     try (PreparedStatement stmt = conn.prepareStatement( "SELECT p.*, c.nombre as categoria FROM productos as p " +
43         "inner join categorias as c ON (p.categoria_id = c.id) WHERE p.id = ?" )) {
44         stmt.setLong( parameterIndex: 1, id );
45         ResultSet rs = stmt.executeQuery();
46         if (rs.next()) {
47             producto = crearProducto(rs);
48         }
49     }
50
51     return producto;
52 }
53
54 2 usages
55 @Override
56 public Producto guardar(Producto producto) throws SQLException {
57     String sql;
58     if (producto.getId() != null && producto.getId() > 0) {
59         sql = "UPDATE productos SET nombre=?, precio=?, categoria_id=?, sku=? WHERE id=?";
60     } else {
61         sql = "INSERT INTO productos(nombre, precio, categoria_id, sku, fecha_registro) VALUES(?,?,?,?,?)";
62     }
63
64     try (PreparedStatement stmt = conn.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS)) {
65         stmt.setString( parameterIndex: 1, producto.getNombre());
66         stmt.setLong( parameterIndex: 2, producto.getPrecio());
67         stmt.setLong( parameterIndex: 3, producto.getCategoria().getId());
68         stmt.setString( parameterIndex: 4, producto.getSku());
69
70         if (producto.getId() != null && producto.getId() > 0) {
71             stmt.setLong( parameterIndex: 5, producto.getId());
72         } else {
73             stmt.setDate( parameterIndex: 5, new Date(producto.getFechaRegistro().getTime()));
74         }
75
76         stmt.executeUpdate();
77     }
78 }
```

```
79
80 if (producto.getId() == null) {
81     try (ResultSet rs = stmt.getGeneratedKeys()) {
82         if (rs.next()) {
83             producto.setId(rs.getLong( columnIndex: 1));
84         }
85     }
86
87     return producto;
88 }
89
90 @Override
91 public void eliminar(Long id) throws SQLException {
92     try (PreparedStatement stmt = conn.prepareStatement( "DELETE FROM productos WHERE id=?" )) {
93         stmt.setLong( parameterIndex: 1, id );
94         stmt.executeUpdate();
95     }
96 }
97
98 2 usages
99 private Producto crearProducto(ResultSet rs) throws SQLException {
100     Producto p = new Producto();
101     p.setId(rs.getLong( columnIndex: "id"));
102     p.setNombre(rs.getString( columnIndex: "nombre"));
103     p.setPrecio(rs.getInt( columnIndex: "precio"));
104     p.setFechaRegistro(rs.getDate( columnIndex: "fecha_registro"));
105     p.setSku(rs.getString( columnIndex: "sku"));
106     Categoria categoria = new Categoria();
107     categoria.setId(rs.getLong( columnIndex: "categoria_id"));
108     categoria.setNombre(rs.getString( columnIndex: "categoria"));
109     p.setCategoria(categoria);
110     return p;
111 }
112 }
```


JDBC – Primer Proyecto

9. Transacciones

```
1 package org.diplomado.java.jdbc.repositorio;
2
3 import org.diplomado.java.jdbc.modelo.Categoria;
4
5 import java.sql.*;
6 import java.util.ArrayList;
7 import java.util.List;
8
9 2 usages
10 public class CategoriaRepositorioImpl implements Repositorio<Categoria> {
11     5 usages
12     private Connection conn;
13
14     1 usage
15     public CategoriaRepositorioImpl(Connection conn) {
16         this.conn = conn;
17     }
18
19     6 usages
20     @Override
21     public List<Categoria> listar() throws SQLException {
22         List<Categoria> categorias = new ArrayList<>();
23         try (Statement stmt = conn.createStatement();
24              ResultSet rs = stmt.executeQuery("SELECT * FROM categorias")) {
25             while (rs.next()) {
26                 categorias.add(crearCategoria(rs));
27             }
28         }
29         return categorias;
30     }
31 }
```

```
28 @Override
29 public Categoria porId(Long id) throws SQLException {
30     Categoria categoria = null;
31     try (PreparedStatement stmt = conn.prepareStatement("SELECT * FROM categorias as c WHERE c.id=?")) {
32         stmt.setLong(1, id);
33         try (ResultSet rs = stmt.executeQuery()) {
34             if (rs.next()) {
35                 categoria = crearCategoria(rs);
36             }
37         }
38     }
39     return categoria;
40 }
41
42 3 usages
43 @Override
44 public Categoria guardar(Categoria categoria) throws SQLException {
45     String sql = null;
46     if (categoria.getId() != null && categoria.getId() > 0) {
47         sql = "UPDATE categorias SET nombre=? WHERE id=?";
48     } else {
49         sql = "INSERT INTO categorias(nombre) VALUES(?)";
50     }
51     try (PreparedStatement stmt = conn.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS)) {
52         stmt.setString(1, categoria.getNombre());
53         if (categoria.getId() != null && categoria.getId() > 0) {
54             stmt.setLong(2, categoria.getId());
55         }
56         stmt.executeUpdate();
57         if (categoria.getId() == null) {
58             try (ResultSet rs = stmt.getGeneratedKeys()) {
59                 if (rs.next()) {
60                     categoria.setId(rs.getLong(1));
61                 }
62             }
63         }
64     }
65     return categoria;
66 }
```

```
65
66 1 usage
67 @Override
68 public void eliminar(Long id) throws SQLException {
69     try (PreparedStatement stmt = conn.prepareStatement("DELETE FROM categorias WHERE id=?")) {
70         stmt.setLong(1, id);
71         stmt.executeUpdate();
72     }
73 }
74
75 2 usages
76 private Categoria crearCategoria(ResultSet rs) throws SQLException {
77     Categoria c = new Categoria();
78     c.setId(rs.getLong("id"));
79     c.setNombre(rs.getString("nombre"));
80     return c;
81 }
```

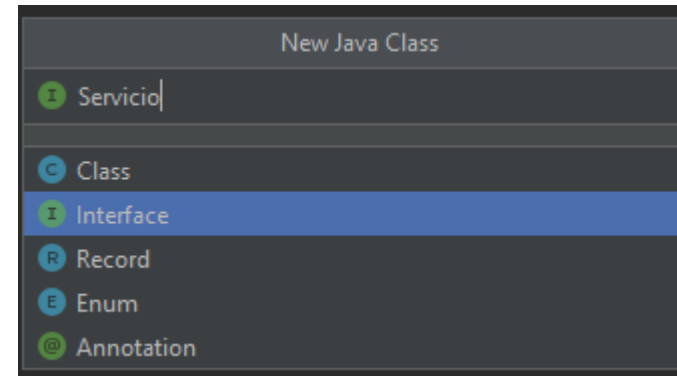
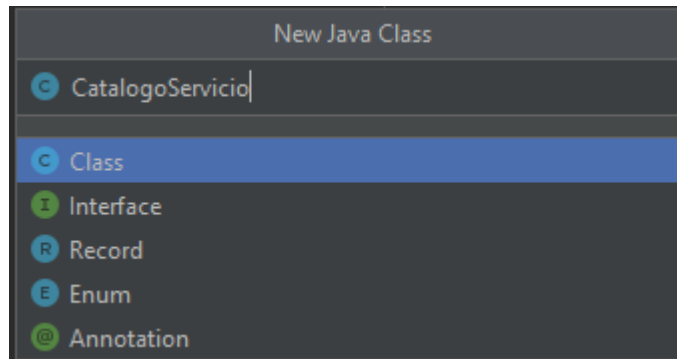
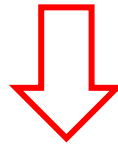
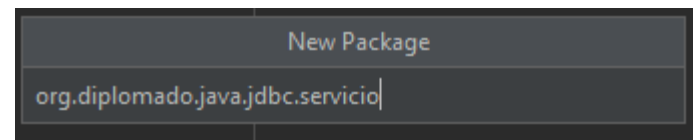
JDBC – Primer Proyecto

9. Transacciones

```
13 public class EjemploJDBC {
14     public static void main(String[] args) throws SQLException {
15         try(Connection conn = ConexionBaseDatos.getConnection()) {
16             if (conn.getAutoCommit()) {
17                 conn.setAutoCommit(false);
18             }
19             try {
20                 Repositorio<Categoria> repositorioCategoria = new CategoriaRepositorioImpl(conn);
21                 System.out.println("===== Insertar nueva categoria =====");
22                 Categoria categoria = new Categoria();
23                 categoria.setNombre("Electrohogar");
24                 Categoria nuevaCategoria = repositorioCategoria.guardar(categoria);
25                 System.out.println("Categoria guardada con éxito: " + nuevaCategoria.getId());
26
27                 Repositorio<Producto> repositorio = new ProductoRepositorioImpl(conn);
28                 System.out.println("===== listar =====");
29                 repositorio.listar().forEach(System.out::println);
30
31                 System.out.println("===== obtener por id =====");
32                 System.out.println(repositorio.findById(1L));
33
34                 System.out.println("===== insertar nuevo producto =====");
35                 Producto producto = new Producto();
36                 producto.setNombre("Refrigerador Samsung");
37                 producto.setPrecio(9900);
38                 producto.setFechaRegistro(new Date());
39                 producto.setSku("abcdefg123");
40
41                 producto.setCategoria(nuevaCategoria);
42                 repositorio.guardar(producto);
43                 System.out.println("Producto guardado con éxito: " + producto.getId());
44                 repositorio.listar().forEach(System.out::println);
45
46                 conn.commit();
47             } catch (SQLException e) {
48                 conn.rollback();
49                 e.printStackTrace();
50             }
51         }
52     }
53 }
```

JDBC – Primer Proyecto

10. Desacoplar lógica de negocio



JDBC – Primer Proyecto

10. Desacoplar lógica de negocio

```
import java.sql.Connection;
import java.sql.SQLException;
import java.util.List;

10 usages 2 implementations
public interface Repositorio<T> {

    10 usages 2 implementations
    void setConn(Connection conn);

    4 usages 2 implementations
    List<T> listar() throws SQLException;

    3 usages 2 implementations
```

```
6 usages
public class ProductoRepositorioImpl implements Repositorio<Producto>{

    6 usages
    private Connection conn;

    1 usage
    public ProductoRepositorioImpl(Connection conn) { this.conn = conn; }

    1 usage
    public ProductoRepositorioImpl() {

    10 usages
    public void setConn(Connection conn) {
        this.conn = conn;
    }
}
```

```
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

4 usages
public class CategoriaRepositorioImpl implements Repositorio<Categoria> {

    6 usages
    private Connection conn;

    1 usage
    public CategoriaRepositorioImpl(Connection conn) { this.conn = conn; }

    1 usage
    public CategoriaRepositorioImpl() {

    10 usages
    public void setConn(Connection conn) {
        this.conn = conn;
    }
}
```

JDBC – Primer Proyecto

10. Desacoplar lógica de negocio

```
package org.diplomado.java.jdbc.servicio;

import org.diplomado.java.jdbc.modelo.Categoria;
import org.diplomado.java.jdbc.modelo.Producto;

import java.sql.SQLException;
import java.util.List;

public interface Servicio {
    List<Producto> listar() throws SQLException;

    Producto porId(Long id) throws SQLException;

    Producto guardar(Producto producto) throws SQLException;

    void eliminar(Long id) throws SQLException;

    List<Categoria> listarCategoria() throws SQLException;

    Categoria porIdCategoria(Long id) throws SQLException;

    Categoria guardarCategoria(Categoria categoria) throws SQLException;

    void eliminarCategoria(Long id) throws SQLException;

    void guardarProductoConCategoria(Producto producto, Categoria categoria) throws SQLException;
}
```

JDBC – Primer Proyecto

10. Desacoplar lógica de negocio

```
1 import org.diplomado.java.jdbc.modelo.Categoria;
2 import org.diplomado.java.jdbc.modelo.Producto;
3 import org.diplomado.java.jdbc.repositorio.CategoriaRepositorioImpl;
4 import org.diplomado.java.jdbc.repositorio.ProductoRepositorioImpl;
5 import org.diplomado.java.jdbc.repositorio.Repositorio;
6 import org.diplomado.java.jdbc.util.ConexionBaseDatos;
7
8 import java.sql.Connection;
9 import java.sql.SQLException;
10 import java.util.List;
11
12 public class CatalogoServicio implements Servicio {
13     11 usages
14     private Repositorio<Producto> productoRepositorio;
15     11 usages
16     private Repositorio<Categoria> categoriaRepositorio;
17
18     public CatalogoServicio() {
19         this.productoRepositorio = new ProductoRepositorioImpl();
20         this.categoriaRepositorio = new CategoriaRepositorioImpl();
21     }
22
23     @Override
24     public List<Producto> listar() throws SQLException {
25         try (Connection conn = ConexionBaseDatos.getConnection()) {
26             productoRepositorio.setConn(conn);
27             return productoRepositorio.listar();
28         }
29     }
30
31     @Override
32     public Producto porId(Long id) throws SQLException {
33         try (Connection conn = ConexionBaseDatos.getConnection()) {
34             productoRepositorio.setConn(conn);
35             return productoRepositorio.porId(id);
36         }
37     }
38 }
```

```
39
40 @Override
41 public Producto guardar(Producto producto) throws SQLException {
42     try (Connection conn = ConexionBaseDatos.getConnection()) {
43         productoRepositorio.setConn(conn);
44         if (conn.getAutoCommit()) {
45             conn.setAutoCommit(false);
46         }
47         Producto nuevoProducto = null;
48         try {
49             nuevoProducto = productoRepositorio.guardar(producto);
50             conn.commit();
51         } catch (SQLException e) {
52             conn.rollback();
53             e.printStackTrace();
54         }
55         return nuevoProducto;
56     }
57 }
58
59 @Override
60 public void eliminar(Long id) throws SQLException {
61     try (Connection conn = ConexionBaseDatos.getConnection()) {
62         productoRepositorio.setConn(conn);
63         if (conn.getAutoCommit()) {
64             conn.setAutoCommit(false);
65         }
66         try {
67             productoRepositorio.eliminar(id);
68             conn.commit();
69         } catch (SQLException e) {
70             conn.rollback();
71             e.printStackTrace();
72         }
73     }
74 }
```

```
75
76 @Override
77 public List<Categoria> listarCategoria() throws SQLException {
78     try (Connection conn = ConexionBaseDatos.getConnection()) {
79         categoriaRepositorio.setConn(conn);
80         return categoriaRepositorio.listar();
81     }
82 }
83
84 @Override
85 public Categoria porIdCategoria(Long id) throws SQLException {
86     try (Connection conn = ConexionBaseDatos.getConnection()) {
87         categoriaRepositorio.setConn(conn);
88         return categoriaRepositorio.porId(id);
89     }
90 }
91
92 @Override
93 public Categoria guardarCategoria(Categoria categoria) throws SQLException {
94     try (Connection conn = ConexionBaseDatos.getConnection()) {
95         categoriaRepositorio.setConn(conn);
96
97         if (conn.getAutoCommit()) {
98             conn.setAutoCommit(false);
99         }
100         Categoria nuevaCategoria = null;
101         try {
102             nuevaCategoria = categoriaRepositorio.guardar(categoria);
103             conn.commit();
104         } catch (SQLException e) {
105             conn.rollback();
106             e.printStackTrace();
107         }
108         return nuevaCategoria;
109     }
110 }
```

```
111
112 @Override
113 public void eliminarCategoria(Long id) throws SQLException {
114     try (Connection conn = ConexionBaseDatos.getConnection()) {
115         categoriaRepositorio.setConn(conn);
116
117         if (conn.getAutoCommit()) {
118             conn.setAutoCommit(false);
119         }
120         try {
121             categoriaRepositorio.eliminar(id);
122             conn.commit();
123         } catch (SQLException e) {
124             conn.rollback();
125             e.printStackTrace();
126         }
127     }
128 }
129
130 @Override
131 public void guardarProductoConCategoria(Producto producto, Categoria categoria) throws SQLException {
132     try (Connection conn = ConexionBaseDatos.getConnection()) {
133         productoRepositorio.setConn(conn);
134         categoriaRepositorio.setConn(conn);
135
136         if (conn.getAutoCommit()) {
137             conn.setAutoCommit(false);
138         }
139         try {
140             Categoria nuevaCategoria = categoriaRepositorio.guardar(categoria);
141             producto.setCategoria(nuevaCategoria);
142             productoRepositorio.guardar(producto);
143             conn.commit();
144         } catch (SQLException e) {
145             conn.rollback();
146             e.printStackTrace();
147         }
148     }
149 }
```

JDBC – Primer Proyecto

10. Desacoplar lógica de negocio

```
import org.diplomado.java.jdbc.modelo.Producto;
import org.diplomado.java.jdbc.modelo.Categoria;
import org.diplomado.java.jdbc.servicio.CatalogoServicio;
import org.diplomado.java.jdbc.servicio.Servicio;

import java.sql.*;
import java.util.Date;

public class EjemploJDBC {
    public static void main(String[] args) throws SQLException {
        Servicio servicio = new CatalogoServicio();
        System.out.println("===== listar =====");
        servicio.listar().forEach(System.out::println);
        Categoria categoria = new Categoria();
        categoria.setNombre("Iluminación");

        Producto producto = new Producto();
        producto.setNombre("Lámpara led escritorio");
        producto.setPrecio(990);
        producto.setFechaRegistro(new Date());
        producto.setSku("abcdefgh12");
        servicio.guardarProductoConCategoria(producto, categoria);
        System.out.println("Producto guardado con éxito: " + producto.getId());
        servicio.listar().forEach(System.out::println);
    }
}
```