Machine Translated by Google

GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

ceedcv CENTRE ESPECÍFIC D'EDUCACIÓ A DISTÀNCIA DE LA COMUNITAT VALENCIANA

# Unidad 2. ACCESO A BASES DE DATOS Parte 1. Trabajar con Bases de Datos Relacionales

## Acceso a Datos (ADA) (a distancia en inglés)

CFGS Desarrollo de Aplicaciones Multiplataforma (DAM)

Abelardo Martínez

Año 2023-2024

Machine Translated by Google

# Créditos

•Apuntes realizados por Abelardo Martínez. •Basado y modificado de Sergio Badal (www.sergiobadal.com). •Las imágenes e iconos utilizados están protegidos por la LGPL . licencia y haber sid de:

- •https://commons.wikimedia.org/wiki/Crystal_Clear
- •https://www.openclipart.org

Machine Translated by Google

# Progreso de la unidad

Machine Translated by Google

# Contenido

Machine Translated by Google

# 1. CONECTORES

Machine Translated by Google

# Conectores

- Los sistemas de gestión de bases de datos (DBMS) de diferentes tipos tienen sus propios lenguajes especializados para tratar los datos que almacenan. •

Los programas de aplicación están escritos en lenguajes de programación de propósito general, como Java.

- Para que estos programas funcionen con DBMS, se necesitan mecanismos que permitan que los programas de aplicación se comuniquen con las bases de datos en estos lenguajes.
- Estos se implementan en una API y se denominan conectores.

Machine Translated by Google

# Conectores para RDBMS

- Para trabajar con sistemas de gestión de bases de datos relacionales (RDBMS), se utiliza el lenguaje SQL .
- Cada RDBMS utiliza su propia versión de SQL, con sus particularidades, por lo que debemos utilizar su propia estructuras de bajo nivel.
- El uso de drivers permite desarrollar una arquitectura genérica en la que el conector tiene una interfaz común tanto para las aplicaciones como para las bases de datos, y los drivers cuidan las particularidades de las diferentes bases de datos.

Machine Translated by Google

# Conectores para RDBMS

• De esta manera, el conector no es sólo una API, sino una arquitectura, porque especifica interfaces que los diferentes drivers deben implementar para acceder a las bases de datos. • Existen diferentes arquitecturas en este sentido:

- •ODBC
- •OLE-DB
- •ADO
- •JDBC

• ODBC y JDBC son los predominantes hoy en día, ya que casi todos los DBMS los implementan y permiten trabajar con ellos. • Los beneficios que ofrecen los conectores basados en controladores tienen el costo de una mayor complejidad de programación.

Machine Translated by Google

# ODBC y JDBC

ODBC (Conectividad abierta de bases de datos) • Define

una API que las aplicaciones pueden usar para abrir una conexión a la base de datos, enviar consultas, actualizaciones, etc. • Las aplicaciones pueden

usar esta API para conectarse a cualquier servidor de base de datos compatible. • Está escrito en lenguaje C.

JDBC (Java Database Connectivity) • Esta es la API

por excelencia para conectar bases de datos a aplicaciones Java y es en la que nos centraremos.

en esta unidad.

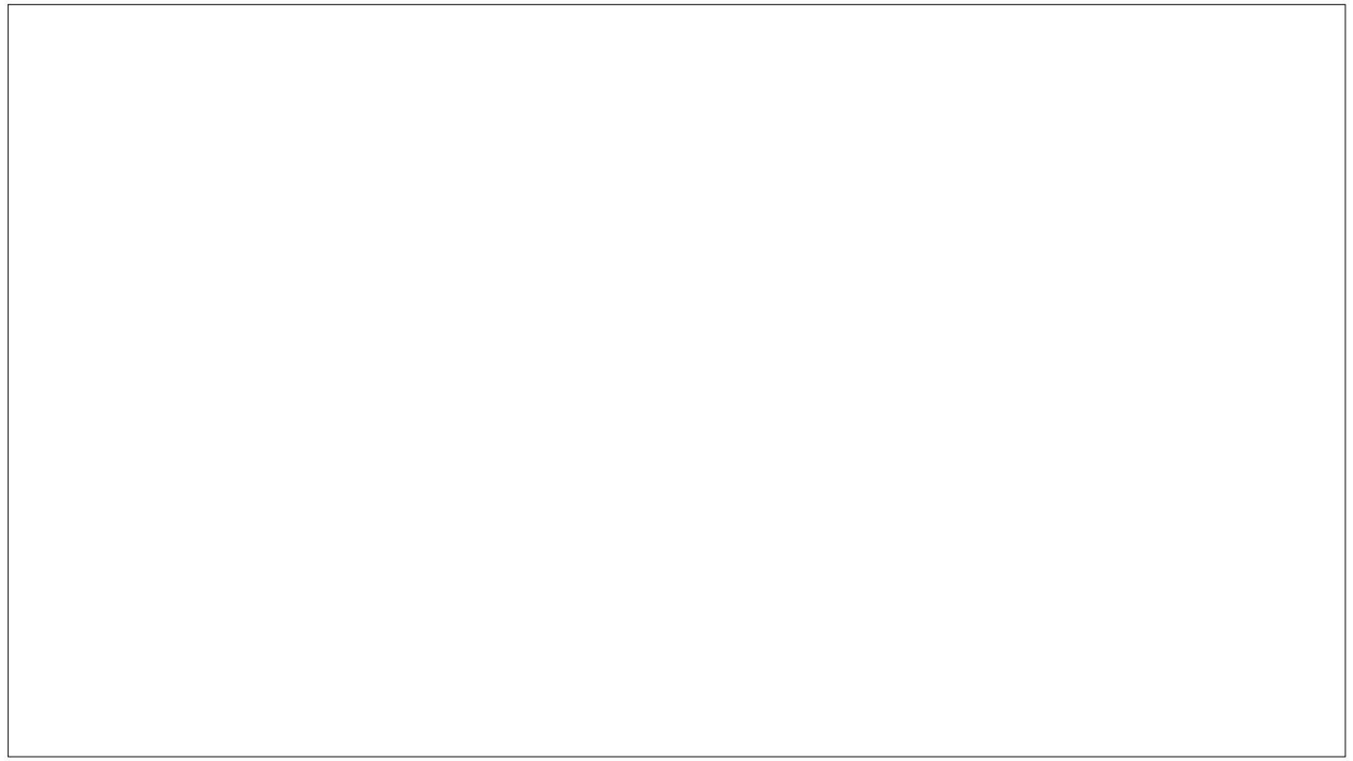
Machine Translated by Google

# 2. ¿QUÉ ES JDBC?

Machine Translated by Google

# ¿Qué es JDBC?

• JDBC proporciona una API para acceder a fuentes de datos orientadas a bases de datos relacionales SQL. •
JDBC tiene una interfaz diferente para cada DBMS, esto se llama Controlador. •
Proporciona una arquitectura que permite a los proveedores crear controladores que permiten que las aplicaciones Java accedan a los datos. • Esto permite que las llamadas a métodos Java se correspondan con la API de la base de datos.

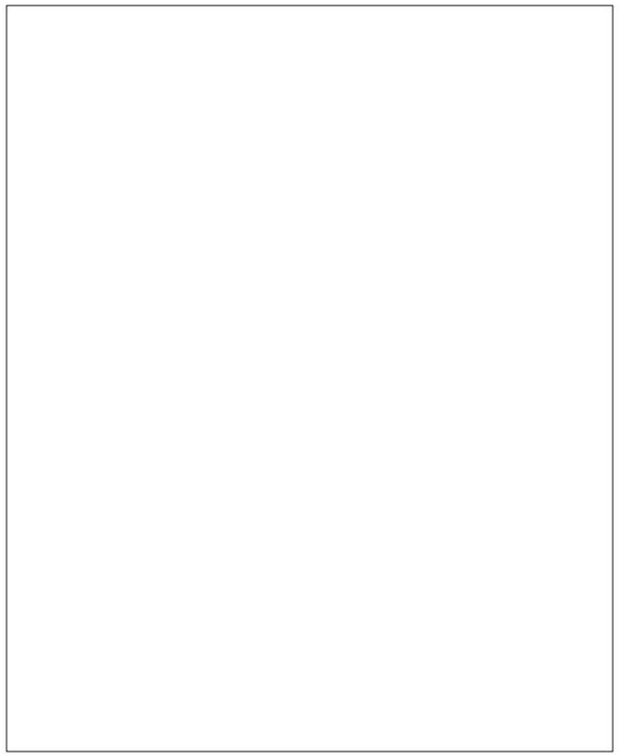


Machine Translated by Google

# Acceso a la base de datos a través de JDBC

JDBC consta de un conjunto de interfaces y clases que nos permiten

escriba aplicaciones Java para gestionar las siguientes tareas con una base de datos relacional:



- Conectarse a una base de datos.
- Enviar consultas y actualizar instrucciones a una base de datos.
- Recuperar y procesar los resultados recibidos de la base de datos en respuesta a esas consultas.

Machine Translated by Google

# JDBC. Modelos de acceso

La API JDBC admite dos modelos de acceso:

1) Modelo de dos capas → La aplicación Java se comunica directamente con la base de datos. Esto requiere un controlador JDBC que resida en el mismo espacio que la aplicación.

- Las sentencias SQL se envían desde el programa Java al DBMS para procesamiento y el DBMS envía los resultados de vuelta al programa.
- El DBMS puede estar ubicado en la misma máquina o en otra y el Las solicitudes se realizan a través de la red.
- El conductor es el encargado de gestionar las conexiones de forma transparente al programador.

2) Modelo de tres capas → Los comandos se envían a una capa intermedia que se encarga de enviar los comandos SQL a la base de datos y recopilar los resultados.

Machine Translated by Google

# JDBC. tipos de conductores

| Tipo | Descripción |
|---|---|
| Puente JDBC-ODBC Permite el acceso JDBC DBMS a través del protocolo ODBC. | |
| Nativo | Controlador escrito parcialmente en Java y en código de base de datos nativo. Traduce llamadas a la API de Java en llamadas DBMS. |
| Red | Controlador Java puro que utiliza un protocolo de red para comunicarse con el servidor de la base de datos. |
| Delgado | Controlador Java puro con protocolo nativo. Traduce llamadas API a llamadas nativas del protocolo de red utilizado por el DBMS. |

Machine Translated by Google

# JDBC. Clases e interfaces

JDBC define clases e interfaces en el paquete java.sql. En la siguiente tabla se muestran los más relevantes:

| Clase o interfaz | Descripción |
|---|---|
| Conductor | Permite conectarse a una base de datos. |
| Administrador de conductores | Le permite administrar los controladores instalados en el sistema. |
| Información de propiedad del controlador | Proporciona información del conductor. |
| Conexión | Representa una conexión a la base de datos. |
| Base de datosMetadatos | Proporciona información de la base de datos. |
| Declaración | Permite ejecutar sentencias SQL sin parámetros. |
| Declaración preparada | Permite ejecutar sentencias SQL con parámetros. |
| Declaración invocable | Permite ejecutar sentencias SQL con parámetros de entrada y salida. |
| Conjunto resultante | Contiene el resultado de las consultas. |
| Conjunto de resultadosMetadatos | Permite obtener información de un ResultSet. |

Machine Translated by Google

# JDBC. Pasos de operación

El funcionamiento de un programa JDBC sigue los siguientes pasos:

• Importar las clases necesarias •

Configurar el archivo POM para cargar el controlador

JDBC • Identificar la fuente

de datos • Crear un objeto

Conexión • Crear un objeto

Declaración • Ejecutar la consulta con el objeto

Declaración • Recuperar el resultado en un ResultSet

• Liberar el objeto ResultSet •

Liberar el objeto Declaración •

Liberar el objeto Conexión

Machine Translated by Google

# 3. ¿QUÉ ES MAVEN?

Machine Translated by Google

# ¿Qué es Maven?

Maven ha sido un proyecto de código abierto bajo Apache desde 2003. Maven es muy estable y rico en funciones, y proporciona numerosos complementos que pueden hacer cualquier cosa, desde generar versiones en PDF de la documentación de su proyecto hasta generar una lista de cambios recientes desde su SCM. Y todo lo que se necesita para agregar esta funcionalidad es una pequeña cantidad de XML adicional o un parámetro de línea de comando adicional.

¿Tienes muchas dependencias? Ningún problema. Maven se conecta a repositorios remotos (o puede configurar sus propios repositorios locales) y descarga automáticamente todas las dependencias necesarias para construir su proyecto.

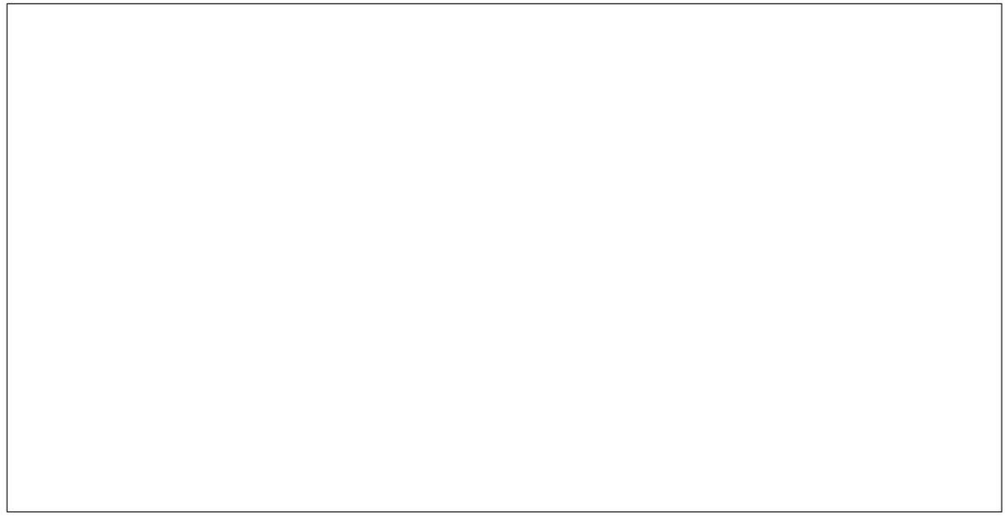
Fuente: https://stackabuse.com/search/?q=maven

Machine Translated by Google

# ¿Cómo funciona Maven?

- Se basa en un archivo central, pom.xml -el Modelo de Objetos del Proyecto (POM) escrito en XML- donde defines todo lo que tu proyecto necesita. Maven gestiona las dependencias del proyecto, compila, empaqueta y ejecuta las pruebas.

• A través de complementos, permite hacer mucho más, como generar mapas de Hibernate a partir de una base de datos, implementar la aplicación, etc.
• Antes de Maven, tenías que descargar manualmente el archivo jar que necesitabas en tu proyecto y copiarlo manualmente en el classpath. •
Con Maven sólo necesitas definir en tu pom.xml las dependencias y Maven las descarga y las agrega al classpath.



Machine Translated by Google

# Maven. Ejemplo de archivo POM

- En este ejemplo definimos el archivo POM para acceder a una base de datos MySQL.

Fuente: https://stackabuse.com/search/?q=maven

Machine Translated by Google

# 4. CONECTARSE A LA BASE DE DATOS

Machine Translated by Google

# Paso 1. Crea la base de datos

Veamos un ejemplo de conexión y consulta de una base de datos MySQL.

1. Instale MySQL: https://dev.mysql.com/doc/mysql-installation-excerpt/8.0/en/ 2. Utilice el código proporcionado a continuación para crear una nueva base de datos llamada testDB y un usuario llamado mavenUser con todos los privilegios (para hacerlo más fácil) y contraseña maven1234X.
3. Usaremos el siguiente esquema de base de datos.
   - Nombre de la base de datos → Empresa
   - Nombre de la tabla → Empleados
   - Campos:
     - Identificación fiscal Varchar(9) PK
     - Nombre Varchar(100)
     - Apellido Varchar(100)
     - Salario decimal (9,2)

4. Agregue cinco personas aleatorias a esa base de datos/tabla.

Machine Translated by Google

# Paso 1. Crea la base de datos

## Este podría ser un buen ejemplo de cómo lograr esos pasos:

- Crear base de datos https://matomo.org/faq/how-to-install/faq_23484/ CREAR BASE DE
DATOS DBCompany;

-- Cree un usuario si está utilizando MySQL 5.7 o MySQL 8 o posterior CREAR USUARIO
'mavenuser'@'localhost' IDENTIFICADO CON mysql_native_password POR 'ada0486'; -- O si está utilizando una versión anterior como
MySQL 5.1, 5.5, 5.6: -- CREAR USUARIO 'mavenuser'@'localhost' IDENTIFICADO POR
'ada0486'; CONCEDA TODOS LOS PRIVILEGIOS EN DBCompany.* A 'mavenuser'@'localhost';

-- Seleccione la base de datos a utilizar
UTILICE DBCompany;

- Crear la tabla del empleado.
CREAR TABLA Empleado (
Identificación del impuesto VARCHAR(9),
Nombre VARCHAR(100), Apellido
VARCHAR(100), Salario DECIMAL(9,2),
CONSTRAINT emp_tid_pk
PRIMARY KEY (TaxID)); -- Insertar empleados aleatorios INSERT

INTO Empleado (NIF, Nombre,
Apellido, Salario) VALUES ('11111111A', 'José', 'Salcedo López', 1279.90); INSERT INTO Empleado (NIF, Nombre, Apellido, Salario) VALUES ('22222222B', 'Juan',
'De la Fuente Arqueros', 1100.73); INSERT INTO Empleado (NIF, Nombre, Apellido, Salario) VALUES ('33333333C', 'Antonio', 'Bosch Jericó', 1051.45); INSERT INTO Empleado
(NIF, Nombre, Apellido, Salario) VALUES ('44444444D', 'Ana', 'Sanchís Torres', 1300.02); INSERT INTO Empleado (NIF, Nombre, Apellido, Salario) VALUES
('55555555E', 'Isabel', 'Martí Navarro', 1051.45);

Machine Translated by Google

# Paso 2. Crea el proyecto dentro del IDE

El primer paso es abrir Eclipse, que viene con el entorno Maven integrado.

- Vaya al menú Archivo, opción Nuevo → Proyecto.
- Seleccione la opción Proyecto Maven .
- Seguimos el resto de pasos explicados en las notas ampliadas.
- Finalmente, seleccionamos Proyecto → Limpiar en nuestro proyecto para que las bibliotecas y archivos necesarios se hayan descargado correctamente.

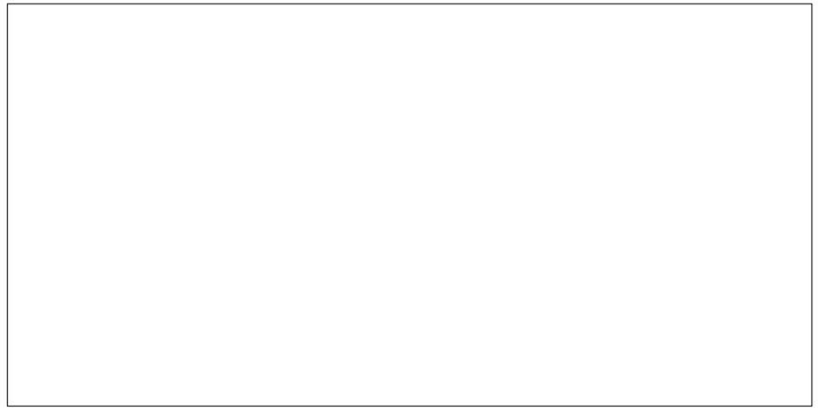
Machine Translated by Google

# Paso 3. Agregue la dependencia al archivo POM

```
<?xml versión="1.0" codificación="UTF-8"?>
<proyecto xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3 .org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"> <modelVersion>4.0.0</
   modelVersion> <groupId>com.simplilearn</
   groupId> <artifactId>U2JDBCExample</
   artifactId> <versión>0.0.1-SNAPSHOT</version>
   <nombre>U2JDBCExample</nombre> <!
   -- FIXME cámbielo al sitio web del
   proyecto --> <url>http://www.example.com</url> <properties>
   <project.build.sourceEncoding>UTF-8</

   project.build.sourceEncoding> < maven.compiler.source>1.7</maven.compiler.source>
      <maven.compiler.target>1.7</maven.compiler.target>

   </properties>
   <dependencias>
      <dependencia>
         <groupId>junit</groupId>
         <artifactId>junit</artifactId>
         <versión>4.11</versión>
         <scope>test</scope> </
      dependency>
      <!-- https://mvnrepository.com/artifact/com.mysql/mysql-connector-j --> <dependencia>
      <groupId>com.mysql</groupId>
         <artifactId>mysql-connector-j</artifactId> <versión>
         8.0.33</versión> </dependencia>
   </dependencias>
```



Machine Translated by Google

# Paso 4. Crea una clase y configura la cadena de conexión.

- Creamos una clase llamada DBMySQL.
- Tenemos que construir nuestra cadena de conexión para MYSQL (donde DBNAME corresponde al nombre de la base de datos):

```
URL="
jdbc:mysql://localhost:3306/DBNAME ?
useSSL=false
&useTimezone=verdadero
&serverTimezone=UTC
&allowPublicKeyRetrieval=true ";
```

Machine Translated by Google

# Paso 5. Conéctese a la base de datos

Establecer la conexión

La conexión a la BD se establece mediante la clase DriverManager y el método getConnection pasando como parámetros URL, USUARIO y CONTRASEÑA de la BD.

```
// Constantes DB
static final String DBUSER = "mavenuser"; //usuario static
final String DBPASSWORD = "ada0486"; //contraseña //cadena para
conectarse a MySQL static final
String URL =
  "jdbc:mysql://localhost:3306/DBCompany?useSSL=false&useTimezone=true&serverTimezone=UTC&allowPublicKeyRetrieval=true";
```

```
//establece la conexión con DBCompany
Connection cnDBCompany = DriverManager.getConnection(URL, DBUSER, DBPASSWORD);
```

Machine Translated by Google

# 5. CONSULTAS DQL

Machine Translated by Google

# Ejecutar declaraciones DQL

El lenguaje de consulta de datos es el sublenguaje responsable de leer o consultar datos de una base de datos. En SQL, corresponde al SELECT.

Podemos realizar consultas a través de la interfaz Statement .

1. Para obtener un objeto Declaración, se llama al método createStatement() de un objeto Conexión válido.
2. El objeto Statement tiene el método ejecutarQuery() que ejecuta una consulta en la base de datos. Este método recibe como parámetro un String que debe contener una sentencia SQL.
3. El resultado se recopila en un objeto ResultSet, este objeto es una especie de matriz o lista que incluye todos los datos devueltos por el DBMS.

```
Declaración staSQLquery = cnDBCompany.createStatement();
//seleccionamos todos los registros
String stSQLSelect = "SELECCIONAR * DEL Empleado";
//recupera datos y los muestra en pantalla. sentencia de selección SQL
ResultSet rsEmployee = staSQLquery.executeQuery(stSQLSelect);
```

Machine Translated by Google

# Desplácese por los resultados

Una vez que hemos obtenido los resultados de la consulta y los hemos almacenado en nuestro ResultSet solo nos queda recorrerlos y tratarlos como mejor nos parezca.

a) ResultSet ha implementado un puntero que apunta al primer elemento de la lista.

b) Mediante el método next() el puntero avanza al siguiente elemento.

- El método next() además de avanzar al siguiente registro devuelve verdadero si hay más registros y falso si ha llegado al final.
- Los métodos getString() y getDouble() se utilizan para obtener los valores de las diferentes columnas. Estos métodos reciben el nombre de la columna como parámetro.

```
mientras (rsEmpleado.siguiente()) {
      System.out.println("TaxID:" + rsEmployee.getString("TaxID"));
      System.out.println(" Nombre: + rsEmployee.getString("Nombre"));
       System.out.println("Apellido : + rsEmployee.getString("Apellido"));
       System.out.println("Salario: " + rsEmployee.getBigDecimal("Salario"));
}
```

Machine Translated by Google

# Ejemplo completo

```
importar java.sql.*; /**

 * ================================================== ==================
 *Ejemplo de acceso a una base de datos relacional MySQL con
  JDBC * @autor Abelardo Martínez
 * ================================================== ==================
 */
public class DBMySQL { //
        Constantes DB
        static final String DBUSER = "mavenuser"; //usuario static final
        String DBPASSWORD = "ada0486"; //contraseña //cadena para conectarse
        a MySQL static final String URL =
        "jdbc:mysql://localhost:3306/DBCompany?useSSL=false&useTimezone=true&serverTimezone=UTC&allowPublicKeyRetrieval=true";

         principal vacío estático público (String[] stArgs )
    {
                try { //
                    establecer la conexión con DBCompany Connection
                      cnDBCompany = DriverManager.getConnection(URL, DBUSER, DBPASSWORD); Declaración staSQLquery
                      = cnDBCompany.createStatement(); //selecciona todos los registros String
                      stSQLSelect = "SELECT * FROM
                      Employee"; //recupera datos y los muestra en pantalla.
                      Declaración de selección SQL ResultSet rsEmployee =
                      staSQLquery.executeQuery(stSQLSelect); while (rsEmployee.next())
                      { System.out.println("TaxID:" +
                            rsEmployee.getString("TaxID")); System.out.println(" Nombre: +
                            rsEmployee.getString("Nombre")); System.out.println("Apellido : + rsEmployee.getString("Apellido"));
                            System.out.println("Salario: " + rsEmployee.getBigDecimal("Salario"));

            }
                      //liberar recursos
                      rsEmployee.close(); //cerrar el conjunto de resultados
                      staSQLquery.close(); //cerrar la declaración
                      cnDBCompany.close(); //cierramos la conexión a la BD
                } catch (SQLException sqle)
                      { sqle.printStackTrace(System.out);
        }
    }
}
```

Machine Translated by Google

# 6. CONSULTAS DML

Machine Translated by Google

# Ejecutar sentencias DML

En SQL las sentencias DML son aquellas que nos permiten agregar, editar o eliminar datos en una base de datos.

• INSERTAR. Para insertar datos en una tabla.

• ACTUALIZAR. Modificar datos existentes dentro de una

tabla. • ELIMINAR. Para eliminar todos los registros de la tabla; no elimina los espacios asignados a los registros.

Para ejecutar cualquiera de estas sentencias debemos utilizar objetos:

• Declaración → Le permite ejecutar declaraciones SQL sin parámetros. • PreparedStatement

→ Ejecutar sentencias SQL con parámetros.

Machine Translated by Google

# Ejemplo

Insertar y eliminar oraciones. Definición:

```
//insertar empleados
String stSQLInsert = "INSERTAR EN Empleado (ID fiscal, nombre, apellido, salario) VALORES "
            + "('11111111A', 'José', 'Salcedo López', 1279.90),"
            + "('22222222B', 'Juan', 'De la Fuente Arqueros', 1100.73),"
            + "('33333333C', 'Antonio', 'Bosch Jericó', 1051.45),"
            + "('44444444D', 'Ana', 'Sanchís Torres', 1300.02),"
            + "('55555555E', 'Isabel', 'Martí Navarro', 1051.45)";
//eliminar el primer empleado
String stSQLDeleteFirst = "ELIMINAR DEL Empleado DONDE TaxID='11111111A'";
```

Insertar y eliminar oraciones. Ejecución:

```
staSQLquery.executeUpdate(stSQLInsert);
System.out.println(" Tabla poblada con varios registros en una base de datos determinada...");
staSQLquery.executeUpdate(stSQLDeleteFirst);
System.out.println("Se eliminó la primera tabla de registros en una base de datos determinada...");
```

Machine Translated by Google

# 7. CONSULTAS DDL

Machine Translated by Google

# Ejecutar sentencias DDL

DDL ( Lenguaje de definición de datos ) es el sublenguaje encargado de definir la forma en que se estructuran los datos en una base de datos.

Aunque el grueso de las operaciones que se realizan desde una aplicación cliente contra una base de datos son operaciones de manipulación de datos, pueden darse casos en los que nos veamos obligados a realizar operaciones de definición:

- Acabamos de instalar una aplicación en un ordenador nuevo y necesitamos crear, de forma automatizada, una base de datos local para su funcionamiento (el caso más común).
- Después de una actualización de software, la estructura de la base de datos ha cambiado y necesitamos actualizarla desde nuestra aplicación Java.
- No conocemos la estructura de la BD y queremos realizar consultas dinámicas sobre la misma.

Las declaraciones DDL tradicionales siguen siendo declaraciones SQL y, por lo tanto, se pueden ejecutar de la forma que ya hemos estudiado a través de la interfaz Statement/PreparedStatement .

Machine Translated by Google

# Ejemplo

Podemos hacer uso del modificador IF NOT EXISTS dentro de nuestro código SQL para asegurarnos de que la base de datos se cree antes de iniciar los procesos CRUD de nuestra aplicación.

```
//soltar tabla Empleado si existe
  String stSQLDrop = "DROP TABLE SI EXISTE Empleado";
//crear tabla Empleado
  Cadena stSQLCreate = "CREAR TABLA Empleado ("
                + "ID de impuesto VARCHAR(9), "
                + "Nombre              VARCHAR(100), "
                + "Apellido            VARCHAR(100), "
                + "Salario             DECIMALES(9,2), "
                + "RESTRICCIÓN emp_tid_pk CLAVE PRIMARIA (ID de impuesto))";
```

Oraciones DDL. Ejecución:

```
//declaraciones y operaciones SQL
staSQLquery.executeUpdate(stSQLDrop);
System.out.println(" Tabla eliminada (si existe) en la base de datos dada...");
staSQLquery.executeUpdate(stSQLCreate);
System.out.println(" Tabla creada en una base de datos determinada...");
```

Machine Translated by Google

# 8. CAMBIAR RDBMS

Machine Translated by Google

# ¿Qué pasa con el uso de otro RDBMS?

Es tan fácil como:

1) Agregue sus dependencias al archivo POM

2) Cambiar la cadena de conexión

¡ESO ES TODO! No hay JAR que descargar, ni ruta que cambiar... ¡nada que configurar! Maven lo hace todo.

Machine Translated by Google

# Cambiar a SQLite

• Instalar SQLite. https://www.sqlite.org/download.html •

Pero, si desea hacerlo más sencillo, no necesita instalar SQLite. Simplemente descargue un ejemplo de base de datos desde aquí: https://www.sqlitetutorial.net/sqlite-sample-database/ •

Coloque el archivo .db dentro de la carpeta de su proyecto.

Machine Translated by Google

# Agregar dependencias al archivo POM

• Vaya al repositorio de Maven: https://mvnrepository.com/artifact/org.xerial/sqlite-jdbc •

Verifique la versión de SQLite que tiene instalada. https://database.guide/check-sqlite-version/ •

Copie y pegue el código de dependencia y agréguelo a su archivo pom.xml dentro de un nuevo nodo llamado <dependencias>, configurando la versión adecuada dentro del nodo <versión>. • Su archivo POM debería ser así:

```
<dependencias>
    <dependencia>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <versión>4.11</versión>
        <scope>test</scope> </
    dependency>
    <!-- https:// mvnrepository.com/artifact/com.mysql/mysql-connector-j --> <dependencia>

    <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <versión>8.0.33< /versión> </
    dependencia>
    <!-- https://mvnrepository.com/artifact/org.xerial/sqlite-jdbc --> <dependencia>

    <groupId>org.xerial</groupId>
        <artifactId>sqlite-jdbc </artifactId>
        <versión>3.31.1</versión> </
    dependencia> </
dependencias>
```

Machine Translated by Google

# Cadena de conexión

- Aprovechamos la clase creada previamente.
  - Cree un nuevo paquete y clase llamado DBSQLite.
  - Copie y pegue el código de la clase DBMySQL.
- Finalmente, cambiamos la cadena de conexión. • Por ejemplo:

```
//Constantes de
base de datos //cadena para conectarse
a MySQL static final String URL = "jdbc:sqlite:bd_sqlite\\chinook.db";

/**
  * ------------------
  * PROGRAMA PRINCIPAL
  * ------------------
  */
principal vacío estático público (cadena [] stArgs) {

        intentar
            { //establecer la conexión con chinook.db
                Conexión cnDBChinook = DriverManager.getConnection(URL);
```

Machine Translated by Google

# 9. ACTIVIDADES PARA LA PRÓXIMA SEMANA

Machine Translated by Google

# Actividades propuestas

Consulta las sugerencias de ejercicios que encontrarás en el “Aula Virtual”. Estas actividades son opcionales y no evaluables, pero comprenderlas es esencial para resolver la tarea evaluable que tenemos por delante.

En breve encontrará las soluciones propuestas.

Machine Translated by Google

# 10. BIBLIOGRAFÍA

Machine Translated by Google

# Recursos


- Josep Cañellas Bornas, Isidre Guixà Miranda. Acceso a datos. Desarrollo de aplicaciones multiplataforma. Creative Commons. Departamento de Enseñanza, Institut Obert de Catalunya. Depósito legal: B. 29430-2013. https://ioc.xtec.cat/educacio/recursos
- Alberto Oliva Molina. Acceso a datos. UD 2. Manejo de conectores. IES Tubalcaín. Tarazona (Zaragoza, España).

Machine Translated by Google

// EDUCACIÓN A DISTANCIA // EDUCACIÓN PÚBLICA Y GRATUITA // EDUCACIÓN DE CALIDAD // EDUCACIÓN AUTONÓMICA // EDUCACIÓN A DISTANCIA // EDUCACIÓN PÚBLICA Y GRATUITA // EDUCACIÓN DE CALIDAD // EDUCACIÓN AUTONÓMICA // EDUCACIÓN A DISTANCIA // EDU

GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

ceedcv CENTRE ESPECÍFIC D'EDUCACIÓ A DISTÀNCIA DE LA COMUNITAT VALENCIANA

CC BY NC ND

BLICA Y GRATUITA // EDUCACIÓN DE CALIDAD // EDUCACIÓN AUTONÓMICA // EDUCACIÓN A DISTANCIA // EDUCACIÓN PÚBLICA Y GRATUITA // EDUCACIÓN DE CALIDAD // EDUCACIÓN AUTONÓMICA // EDUCACIÓN A DISTANCIA // EDUCACIÓN PÚBLICA Y GRATUIT A // EDUCACIÓN DE CALIDAD // EDUCACIÓN AUTONÓMICA // EDUCACIÓN A DISTANCIA // EDU