NEREA ZAPATERO JARA
DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Nerea Zapatero Jara

2 DAM

Contenido

Hibernate

Para usar Hibernate, se ha de tener en cuenta lo siguiente se necesita la aplicación de MySql Workbench y tener instaladas las JBox tool en Eclipse.

> **¿Qué es Hibernate?**
>
> Hibernate es una herramienta de mapeo objeto-relacional.

Paso a paso

Lo primero a realizar será crear el nuevo usuario y la *database* (o esquema o base de datos) dentro del MySQL Workbench.

Una vez creado el usuario y la base de datos, iremos al inicio para realizar una nueva conexión con dicho usuario.

Pondremos en la casilla: "*Esquema por defecto*" el nombre de la base de datos que hemos creado anteriormente.

Después, le añadiremos los datos.

Pasamos a Eclipse.

En Eclipse lo que haremos será crear un nuevo proyecto Maven, en el archivo POM será donde meteremos las siguientes dependencias:

- ***MySQL Connector/J***
  - Versión que usar: 8.3.0
- ***Hibernate Core Relocation***
  - Versión que usar: 5.6.6.Final

Configuración de Hibernate(Sin imágenes)

Para realizar la configuración se ha de crear y configurar los siguientes archivos:

- *Configure Hibernate*
- *Consola*
- *Reveng*

En el primer archivo que haremos será el de configuración y la consola.

La ruta donde crearemos este archivo será en la raíz del proyecto, es decir, en "src/main/java".

Lo siguiente será seleccionar la versión que hayamos puesto en el POM (es decir la 5.6), lenguaje de la base de datos será MySql.

RECORDATORIO

Algunas cosas hay que modificarlas según lo que uno tenga.

Los campos con el color: **a modificar**

Controlador: *com.mysql.jdbc.Driver*

URL de la conexión: *jdbc:mysql://localhost/database*

Nombre del usuario y su contraseña, también pondremos el nombre del esquema/base de datos por defecto.

Antes de finalizar, le daremos a la casilla de crear consola a la vez que la configuración de Hibernate. La consola se dejará tal cual, el nombre se puede sustituir con otro, si no es el caso, tomara por defecto el nombre del proyecto.

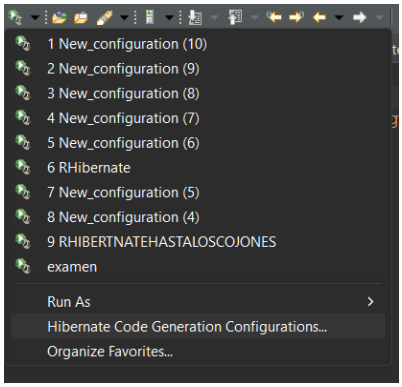
También se puede crear a parte la consola, pero sería el mismo proceso.

Segundo archivo que crear *Reveng.xml*

La ruta será la misma que se uso en el anterior archivo. Después seleccionaremos la consola que hemos creado y le daremos al botón de refrescar. Hay que expandir la base que queramos añadir, y hay que añadir uno a uno las tabas.

Por último, generamos las clases de Hibernate.

Para eso iremos al botón de "Hibernate" al desplegable y le daremos a "Hibernate Code Generation Configurations···"



- Main:
  - Selecciona la consola
  - El directorio de salida
  - Clica en la casilla del "Reverse engineer from JDBC Connection"
  - Package: **Nombre del lugar donde se va a crear dichas clases**
  - Reveng.xml: **Ruta del archivo**
- Exporters:
  - Use Java 5 syntax
  - Domain Code (.java)
  - Hibernate XML Mapping (.hbm.xml)
  - Hibernate XML Configuration (.cfg.xml)

Después iremos a la consola y una vez hay le daremos a mapeo diagrama. Si sale el diagrama es que esta todo bien, si no sale algo hay mal.

La conexión con Hibernate = Java la realizaremos con la clase HibernateUtil y colocaremos lo siguiente dentro del código:

```
SessionFactory sf = HibernateUtil.getSessionFactory();

Session s = sf.openSession();
```

Conectores

Acceso a datos mediante JDBC

Se unas una API estándar de Java para la conexión con bases de datos, principalmente relacionales[1], que utilizando SQL.

JDBC nos permite realizar lo siguiente:

- **Conexión a la base de datos.**
- **Ejecutar las sentencias SQL:** Con la interfaz **Statament**, se pueden crear y ejecutar consultas SQL
- **Manejar resultados:** Los resultados de las consultas se manejan con la interfaz **ResultSet,** permite recorrer y obtener los datos de cada fila.
- **Liberar recursos.**

Guía Conectores

Me da pereza hacer de nuevo entera la guía de cada conector así que:
ENLACE A LA GUIA

Detalles sobre Statament y ResultSet

Son componentes fundamentales de la API JDBC, que permite a las aplicaciones Java interactuar con las bases de datos relacionales.

Se usan en conjunto para ejecutar consultas SQL y procesar los resultados de dichas consultas.

Relación entre ambas interfaces

- **Statament:** Ejecutar consultas SQL
  - Métodos principales: executeQuery, executeUpdate y execute
  - Crea y devuelve un ***ResultSet*** para consultas '*Select*'
- **ResulSet:** devuelve la ejecución de la consulta '*Select*' a través del ***Statament***
  - Métodos principales: next, get…
  - Permite iterar y acceder a los datos de cada fila en el conjunto de resultados.

[1] Sistema de almacenamiento de datos que organiza la información en tablas con filas y columnas.

Statament

Es una interfaz de la API JDBC que permite ejecutar consultas SQL contra una base de datos.

Su uso principal es para enviar consultas SQL estáticas.

- Es una interfaz en el paquete **java.sql**
- Permite ejecutar consultas SQL y comandos DML [2](Select, Insert, Update, Delete).

ResulSet

Es una interfaz de la API JDBC que representa el conjunto de resultados devueltos por una consulta SQL.

Actúa como un cursor que se puede mover a través de los resultados fila por fila, permite acceder y manipular los datos devueltos por la consulta.

- Es una interfaz en el paquete **java.sql**
- Métodos para navegar y obtener datos de un conjunto de resultados.

Ejemplo:

```
public class Ej_Clase1 {
	public static void main(String[] args) {
		try {
				Connection conexion =
DriverManager.getConnection("jdbc:mysql://localhost/basededatos?useSSL=true
&serverTimezone=UTC", "usuario", "contraseña");
				Statement s = conexion.createStatement();
				String sql = "SELECT * FROM departamentos";
				ResultSet r = s.executeQuery(sql);
				while (r.next()) {
	System.out.printf("%d, %s, %s, %n",
r.getInt(1), r.getString(2), r.getString(3));
				}
		} catch (SQLException ex) {
	System.out.println("BASE DE DATOS NO ENCONTRADA");
ex.printStackTrace();
		}
	}
}
```

Lo que hace el ejemplo es que selecciona toda la tabla de departamentos y la muestra.

[2] Lenguaje de manipulación de datos; permite manipular los datos almacenados en las tablas de una base de datos.

El *ResultSet* puede mostrar o bien por la posición(como en el ejemplo) o por el nombre de la columna.

Ejecutar sentencias de descripción de datos (DatabaseMetaData)

Cuando no conocemos la estructura de las tablas de una base de datos, se puede obtener a través de los *metaobjetos*, que son objetos que proporcionan información sobre la base de datos.

Para eso debemos de usar la interfaz **DatabaseMetaData** que es la que nos proporciona información acerca de la base de datos.

En este enlace se muestra el ejemplo de la obtención de las claves privadas y ajenas.

Ejecutar sentencias de descripción de datos (ResultSetMetaData)

Proporciona metadatos sore las columnas de un ojebto ReusltSet. Se utiliza para obtener información detalla acerca de las columnas resultamtes de una consulta SQL

Sentencias preparadas

Se ha de usar la interfaz PreparedStatament la cual nos deja construir una cadena de SQL con marcadores de posición (no es lo mismo que los cursores).

Se representan mediante el símbolo de interrogación.

Por ejemplo:

String sql = “INSERT INTO departamento VALUES (?, ?, ?)”;

Procedimiento y Funciones

Son bloques de códigos SQL que se guardan y ejecutan en el servidor de la base de datos. Facilitan tareas respetivas y complejas directamente en la base de datos.

Se pueden definirse con parámetros de entrada (IN), salida(OUT) o ambos(INOUT). Si no se coloca ninguno se pone el de por defecto, entrada (IN).

- **Procedimientos:** Realiza una tarea especifica y no necesariamente devuelven un valor.
- **Funciones:** Devuelven un valor (y su nombre)

Hay que tener en cuenta de que hay cuatro formas de declarar las llamadas a los procedimientos y funciones.

- Procedimientos:
  - {Call pro}
  - {Call pro (?, ?.....)}
- Funciones:
  - {? = Call fun}
  - {? = Call fun (?, ?.....)}
  -

Procedimientos

```
DELIMITER $$
          CREATE PROCEDURE subida_sal(d INT, subida INT)
          BEGIN UPDATE empleados SET salario = salario + subida WHERE
dept_no = d;
          COMMIT;
          END$$ DELIMITER ;
```

Se define la llamada del procedimiento, los parámetros indica que son de entrar (no se definió por lo tanto son IN).

El bloque 'Begin···End': lo que hace es que contiene las instrucciones que se deben de ejecutar.

Lo que hace es actualizar la tabla de empleados, incrementan el salario de todos los empleados del departamento especificado.

Con el Commit se confirma la transacción, asegurándose de que se hayan realizado las instrucciones.

```
public class SubidaProc {

	public static void main(String[] args) {
		try {
			Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost/basededatos?useSSL=true&serverTimezone=UTC", "usuario", "contraseña");                        String dep = args[0];
			String subida = args[0];
			String sql = "{call subida_sal (?, ?)}";
			//Se esta usando un procedimiento.s
			CallableStatement l = conexion.prepareCall(sql);
			l.setInt(1, Integer.parseInt(dep));
			l.setFloat(2, Float.parseFloat(subida));
			l.executeUpdate();
			System.out.println("Subida realizada.... ");
		} catch (SQLException e) {
			System.out
					.println("ERROR EN LA BASE DE DATOS, COLUMNA O TABLA NO ENCONTRADA. INTENTELO DE NUEVO POR FAVOR.");
			System.out.println("DESCRIPCION DEL ERROR: \n");
			e.printStackTrace();
			System.exit(0);
		}

	}

}
```

Funciones

```
DELIMITER $$

CREATE FUNCTION nombre_dep (d INT, OUT locali VARCHAR(255)) RETURNS VARCHAR(15)
BEGIN
    DECLARE nom VARCHAR(15);

    SELECT dnombre, loc INTO nom, locali FROM departamentos WHERE dept_no = d;

    IF nom IS NULL THEN
        SET nom := 'NO EXISTE';
    END IF;

    RETURN nom;
END$$DELIMITER ;
```

- ***Manejo de Ficheros***: El documento trata sobre el manejo de ficheros en Java, incluyendo clases para operaciones de gestión, flujos de datos y excepciones.
- ***Operaciones Básicas***: Se describen operaciones sobre ficheros como creación, apertura, cierre, lectura y escritura, tanto para acceso secuencial como aleatorio.
- ***Flujos y Streams***: Se explica el uso de flujos de bytes y caracteres para la entrada/salida de datos en ficheros, con ejemplos de clases como FileInputStream y FileWriter.
- ***Acceso Aleatorio***: Se detalla cómo utilizar la clase RandomAccessFile para acceder y modificar ficheros de forma no secuencial, permitiendo un manejo más eficiente de los datos.

Nerea Zapatero Jara
2 DAM

Algunos comandos de MySQL Workbench:

RECORDATORIO

Algunas cosas hay que modificarlas según lo que uno tenga.

Los campos con el color: **a modificar**

- Borrar usuario:
  - **drop user 'usuario'@'localhost';**
- Crear usuario:
  - **Create user 'usuario'@'localhost' identified by 'contraseña';**
- Privilegios de los usuarios (Hacerlo super Usuario):
  - **GRANT ALL PRIVILEGES ON *.* TO 'usuario'@'localhost' WITH GRANT OPTION;**
  - **FLUSH PRIVILEGES;**
- Crear *database*/base de datos/esquema:
  - **CREATE DATABASE nombre;**
- Borrar *database*/base de datos/esquema:
  - **DROP DATABASE nombre;**