Sumario

Notas Examen todo con prepareStatement
DAO un método que maneje el rs
Metadatos en una clase a parte
Generar DAO estándar y POJO de una tabla

API JDBC

Para utilizar una base de datos a través de JDBC tenemos que :

1. Añadir el driver a nuestro proyecto
2. Registrar el driver
3. Establecer la conexión
4. Realizar operaciones sobre la base de datos
5. Cerrar la conexión.

Properti

Para añadir un fichero de propiedades que tiene la conexión. Se usa con DataSource

Statement

Para enviar sentencias SQL contra la base de datos. Es un método de la Connection

```
stmt = conexion.createStatement();
stmt.executeUpdate(sql);
```

Método	Uso recomendado
executeQuery()	Con sentencias SELECT - Devuelve un ResultSet
executeUpdate()	Con sentencias INSERT, UPDATE y DELETE, o DDL SQL.
execute()	Cualquier sentencia DDL, DML o comando especifico de la BD

PreparedStatement

Permite crear objetos de sentencias SQL precompiladas con parámetros de entrada (?)

```
conn = DriverManager.getConnection(jdbcUrl,"root","");
pstmt = conn.prepareStatement("update facturas set serie=? where id=?");
```

Se especifica el valor antes de ejecutar el pstmt con métodos setXXX()

```
pstmt.setString(1, "B");
pstmt.setLong(2, 1);
pstmt.executeUpdate();
```

- **clearParameters()** borrados los parámetros.
- **executeUpdate()** ejecuta los cambios en la base de datos.

Cierre de conexiones

Primero el Statement, después la Connection, dentro de un bloque finally
Evitar cualquier excepción en el cierre. Una forma es capsular cada cierre en un try-catch independiente

ResultSet

Mantiene un cursor apuntando a la fila actual de datos. Inicialmente está antes de la primera fila y es válido hasta que el **ResultSet** o su padre **Statement** se cierra.

- **next()** cada vez que se llama se mueve haca abajo el cursor

TIPOS DE ResultSet

- **createStatement()** Si especificamos el tipo ResultSet es obligatorio indicar si va a ser de solo lectura o no.

```
//Uso del método executeQuery
stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_READ_ONLY);
String sql = "select * from equipo";
rs = stmt.executeQuery(sql);
```

Primer parámetro indica el tipo de objeto ResultSet

- **TYPE_FORWARD_ONLY**: Con movimiento únicamente hacia delante. **(por defecto)**.
- **TYPE_SCROLL_INSENSITIVE**: Permite todo tipo de movimientos. Mientras está abierto no mostrará los cambios que se realicen sobre los datos.
- **TYPE_SCROLL_SENSITIVE**: Permite todo tipo de movimientos y ver los cambios que se realizan sobre los datos que contiene.

Segundo parámetro (tipo de concurrencia) si el ResultSet es solo de escritura o si permite modificaciones.

- **CONCUR_READ_ONLY**: indica que es solo de lectura. **(por defecto)**.
- **CONCUR_UPDATABLE**: permite realizar modificaciones sobre los datos.

OBTENER DATOS EN EL ResultSet

getXXX**()** para recuperar valores de columnas
Para indicar una columna podemos utilizar su nombre o bien su número empezando por la columna 1

```
String valor = rs.getString(2);
String valor = rs.getString("titulo");
```

findColumn() Devuelve el numero de columna dándole el nombre.

DESPLAZAMIENTO POR EL ResultSet

Métodos de desplazamiento del cursor.

- **boolean absolute(int registro)**: Cursor al registro indicado. Si es negativo empieza por el final. Devuelve false si se manda después del último o antes del primer registro.
- **void afterLast()**: Cursor después del último registro.
- **void beforeFirst()**: Cursor antes del primer registro.
- **boolean first()**: Cursor al primer registro. false si registro no valido o si no tiene registros. Si no true.
- **boolean last()**: Cursor al último registro. false si registro no valido o si no tiene registros. Si no true.
- **void moveToCurrentRow()**: Tras una inserción, mueve cursor a la posición anterior.
- **boolean previous()**: Cursor al registro anterior. contrario al **next()**. True si registro o fila válidos, false en caso contrario.
- **boolean relative(int registros)**: Cursor se mueve un número de registros. Si número negativo hacia el principio. Si positivo hacia el final.

Métodos de información del cursor.

- **boolean isAfterLast()**: indica si está después del último registro. Solo en scrollable.
- **boolean isBeforeFirst()**: indica si está antes del primer registro. Solo en scrollable.
- **boolean isFirst()**: indica si está en el primer registro. Solo en scrollable.
- **boolean isLast()**: indica si está en el último registro. Solo en scrollable.
- **int getRow()**: devuelve el número de registro actual. Devolverá cero si no hay registro actual.

MODIFICAR DATOS EN EL ResultSet

UpdateXXX()
El ResultSet debe ser Updatable

1. Nos situamos sobre el registro a modificar
2. Lanzamos métodos **updateXXX()** adecuados, con los nuevos valores.
3. Lanzamos el método **update Row()** para que los cambios tengan efecto **sobre la base de datos**.

```
rs.last();
rs.updateString("direccion", "C/ PepeCiges, 3");
rs.updateRow();
```

Si nos desplazamos antes de lanzar el updateRow(), se perderán las modificaciones.

- **cancelRowUpdates()** para cancelar las modificaciones en todos los campos del registro. No tiene efecto si se ha invocado antes al updateRow()

INSERTAR UNA FILA

1. **MoveToInsertRow()**
2. **updateXXX()** Los campos a los que no demos valor tendrán valor NULL
3. **insertRow()**
4. moveToCurrentRow() **(Opcional)** volver a la antigua posición. MoveToCurrentRow() solo se puede utilizar en combinación con el método moveToInsertRow().

BORRAR UNA FILA

1. movernos al registro
2. deleteRow()

```
rs.last();
rs.deleteRow();
```

TRANSACCIONES

Para ejecutar varias sentencias como un todo. Se ejecuta sobre la Conexión.

- **setAutoCommit()** en Connection, a false, la sentencia no se ejecuta automáticamente.

```
conexion.setAutoCommit(false);
String sql = "UPDATE Cuentas SET cantidad = ? WHERE NCuenta= ?”
```

- **commit()** modificará la base de datos.

```
String sql = "UPDATE Cuentas SET cantidad = ? WHERE NCuenta= ?”
PreparedStatement sentenciaResta=conexion.prepare Statement();
sentenciaResta.executeUpdate(sql);
Statements entenciaSuma=conexion.createStatement();
sentenciaSuma.executeUpdate("UPDATE Cuentas SET
cantidad=cantidad+5000 WHERE NCuenta="+cuentaDestino);
conexion.commit();
conexion.setAutoCommit(true);
```

- **rollback()** En caso de producirse algún error, se aborta la transacción mediante el método del interfaz Connection

```
}catch(SQLException ex){
conexion.rollback();
System.out.println("La transacción a fallado.");
conexion.setAutoCommit(true);
```

ERRORES

- SQLException → Casi todos los métodos de JDBC
- SQLWarning → Connection, Statement y ResultSet
- DataTruncation → Statement.setMaxFieldSize()

DatabaseMetadata

Ofrece Información de la BD. "%" cadena de 0 o más caracteres. "_" un solo carácter.

- **getMetaData()** sobre el objeto **Connection**

```
Connection conn = DriverManager.getConnection(jdbcUrl,"root","");
DatabaseMetadatad bmd = conn.getMetaData();
```

Métodos que devuelven valores String o int

```
//Nombre del SGBD
String producto = dbmd.getDatabaseProductName();
//Versióndel SGBD
String version = dbmd.getDatabaseProductVersion();
//Usuario conectado
String nombreUsr = dbmd.getUserName();
//Url de la base de datos
String url = dbmd.getURL();
-----------
String driver = dbmd.getDriverName();
String driverVersion = dbmd.getDriverVersion();
-----------
String funcionesCadenas = dbmd.getStringFunctions();
String funcionesSistema = dbmd.getSystemFunctions();
String funcionesTiempo = dbmd.getTimeDateFunctions();
String funcionesNumericas = dbmd.getNumericFunctions();
```

Métodos que devuelven información sobre las tablas de la base de datos en un ResultSet

- **getTables()** Los tres primeros parámetros los pondremos a null, el último indica el tipo de tabla que queremos obtener. **los tipos de tabla son:**
  - *TABLE, VIEW, SYSTEM TABLE, GLOBAL TEMPORARY, LOCAL TEMPORARY, ALIAS y SYNONYM*

El ResultSet devuelto tiene un formato determinado con los siguientes nombres de columnas:
TABLE_CAT: catálogo de la tabla.
TABLE_SCHEM: esquema de la tabla.
TABLE_NAME: nombre de la tabla.
TABLE_TYPE: tipo de la tabla.
REMARKS: comentarios acerca de la tabla

```
ResultSetrsTablas = dbmd.getTables(null,null,null,
new String[]{"TABLE","SYSTEM_TABLE"});
System.out.println();
System.out.println("------ TABLAS-------");
while(rsTablas.next()){
System.out.println(rsTablas.getString("TABLE_CAT"));
System.out.println(rsTablas.getString("TABLE_SCHEM"));
System.out.println(rsTablas.getString("TABLE_NAME"));
System.out.println(rsTablas.getString("TABLE_TYPE"));
System.out.println(rsTablas.getString("REMARKS"));
System.out.println("------------");
```

- **getProcedures()** mismos parámetros que getTables(), a excepción del ultimo, el array de String.

El ResultSet devuelto tiene un formato determinado con los siguientes nombres de columnas:

PROCEDURE_CAT: catálogo del procedimiento.
PROCEDURE_SCHEM: esquema del procedimiento.
PROCEDURE_NAME: nombre del procedimiento.
REMARKS: comentarios acerca del procedimiento.
PROCEDURE_TYPE: tipo de procedimiento.

consultar la tabla ***puerto*** de la base de datos *ciclismo* y mostrar información de cada una de las columnas como el nombre de la columna, el tipo y otra información relevante.

ResultSetMetadata

Un objeto ResultSetMetaData lo obtendremos lanzando el método getMetaData() sobre el ResultSet.

- **getTableName()** Devuelve String con el nombre de la tabla
- **getColumnCount()** Devuelve int con el número de columnas
- **getColumnName(int n)** Devuelve String con el nombre de la columna en la posición n
- **getColumnType(int n)** Devuelve el tipo JDBC de la columna en la posición n
- **getColumnTypeName(int n)** Devuelve el nombre del tipo JDBC que se corresponde con el tipo de la columna en la posición n.

```
Statement stm = con.createStatement();
ResultSet rs = stm.executeQuery("SELECT * FROM puerto");
// DatabaseMetadata
ResultSetMetaData rsmd = rs.getMetaData();
int numColumnas = rsmd.getColumnCount();
for(int i = 1; i<=numColumnas ;i++){
System.out.println("Columna " + i);
System.out.println("Nombre: " + rsmd.getColumnName(i));
System.out.println("Tipo: " + rsmd.getColumnType(i));
System.out.println("Tipo: " + rsmd.getColumnTypeName(i));
System.out.println("Display Size: " + rsmd.getColumnDisplaySize(i));
System.out.println("ClassName: " + rsmd.getColumnClassName(i));
System.out.println("Label: " + rsmd.getColumnLabel(i));
System.out.println("Precision: " + rsmd.getPrecision(i));
System.out.println("Escala: " + rsmd.getScale(i));
System.out.println("Nullable: " + rsmd.isNullable(i));
System.out.println("Auto increm: " + rsmd.isAutoIncrement(i));
System.out.println("-----------");
}
…
```