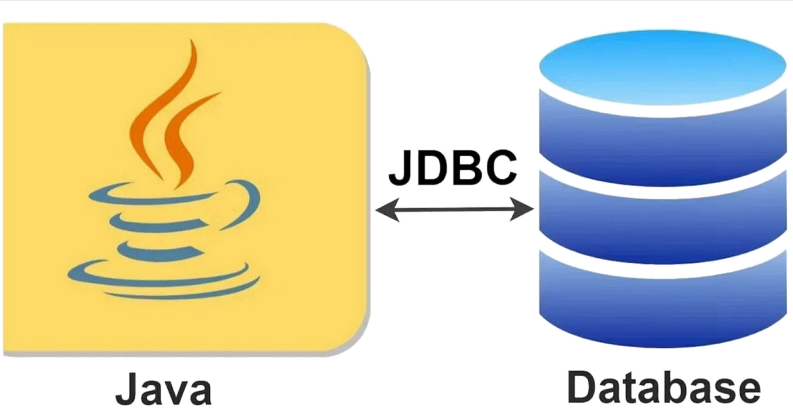
PAU CASESNOVES
Centre Integrat de Formació Professional

Desenvolupament d'aplicacions multiplataforma



Accés a dades

UT 5.2. ORDB i JDBC

Índex de continguts

JDBC i ORDB

A les darreres versions JDBC incorpora mètodes i classes que ens permeten utilitzar els nous tipus de dades.

Seguirem utilitzant *PostgreSQL* per practicar-ho

Connexió

La gràcia que té JDBC és que has de canviar molt poques coses quan canvies de Sistema Gestor de Bases de Dades.

Una de les coses que hem de modificar és la url. En aquest cas, canviam el protocol *jdbc:mysql* que utilitzàvem fins ara per *jdbc:postgresql* i segurament haurem de canviar el port, del 3306 de *MySQL* a 5432 de *PostgreSQL*

```
jdbc:postgresql://daw.paucasesnovescifp.cat:5432/usuaridb?
        user=usuari&password=seCret_25
```

A la resta de codi, si no hem d'utilitzar cap de les característiques que hem vist en aquest tema no farà falta canviar pràcticament res.

Boolean

Recuperar una columna boolean

L'objecte *Resultset* té el mètode *getBoolean* que torna un valor del tipus primitiu *boolean*.

```
...
try (Connection con = ...;
Statement smt = con.createStatement();
ResultSet rs = stmt.executeQuery("select * from proves.\"Clients\"");) {
        while (rs.next()) {
```

```
            boolean recomanaria = rs.getBoolean("recomanaria");
...
```

Assignar o modificar una columna boolean

A l'executeUpdate podem posar qualsevol dels valors vàlids.

L'objecte PreparedStatement té el mètode setBoolean(int index, boolean valor);

```
try (Connection con = ...;
  PreparedStatement stmt = con.prepareStatement("update
                proves.\"Clients\" set recomanari=? where id=?");){
        stmt.setBoolean(1, true);
        stmt.setInt(2, 5);
        stmt.executeUpdate();
} catch (SQLException e)
...
```

Arrays

No podem treballar directament amb els arrays a la base de dades, haurem d'utilitzar la classe *java.sql.Array* com a intermediari i haurem de fer la conversió de l'array Java a un objecte d'aquest tipus, i al revés.

Recuperar un array

L'objecte *ResultSet* té el mètode *getArrays* que ens **torna una referència** a un objecte del tipus *java.sql.Array*.

```
...
ResultSet rs = stmt.excuteQuery();
while(rs.next){
  Array ref=rs.getArray("emails");
...
```

Important: Ens torna una referència a l'array dins de la base de dades, encara no l'hem recuperat.

Per obtenir les dades que conté l'array i passar-los a objectes Java, hauríem d'obtenir l'array i fer un càsting

```
...
ResultSet rs = stmt.excuteQuery();
while(rs.next){
  Array ref=rs.getArray("emails");
  String[] emails=(String[]) ref.getArray();
...
```

En executar-se l'anterior instrucció es recupera l'array de la base de dades, es fa el càsting i es guarda dins la variable emails.

Convé alliberar els recursos assignats a l'objecte *java.sql.Array*

```
...
ResultSet rs = stmt.excuteQuery();
while(rs.next){
  Array ref=rs.getArray("emails");
  String[] emails=(String[]) ref.getArray();
  ref.free();
...
}
```

Insertar una fila amb una columna de tipus array

Si ho feim amb l'objecte *Statement* es tracta de generar una cadena amb la instrucció SQL. L'array es representa com una cadena SQL amb ' ', tancat dins claus, i cada element és del tipus de l'array SQL, en aquest cas VARCHAR tancats amb " ":

```
insert into proves."Clients" values(5, 'Armando Adistancia',
            '{"ad@mail.net","ad@mail.org"}',  false,  null,    null,    null);
```

Com sempre, per a més seguretat i eficiència, podem utilitzar el *PreparedStatement*:

- Cream un objecte del tipus *java.sql.Array* a partir de l'objecte *Connection*. Aquest mètode té com a primer paràmetre el tipus SQL dels elements de l'array i com a segon paràmetre l'array Java que volem guardar a la base de dades:

```
String[] temp = {"ad@mail.net", "ad@mail.org"};
java.sql.Array emails = con.createArrayOf("VARCHAR", temp);
```

- Cream el *PreparedStatement* a partir de la connexió:

```
PreparedStatement st = con.prepareStatement("insert into proves."Clients" values(?, ?, ?, ?, ?, ?, ?)")
```

- Utilitzam el mètode *setArray* per passar-li l'array:

```
st.setArray(3, emails);
```

- Executam l'statement

```
st.executeUpdate();
```

Tipus definits per l'usuari

Com hem vist abans, PostgreSQL ens permet definir tipus nous que es poden utilitzar per definir columnes de les taules.

Per recuperar aquestes columnes necessitam crear un POJO que representi aquest tipus. Quan volguem recuperar aquestes columnes recuperarem un objecte de la classe del POJO i quan volguem insertar o modificar el valor d'aquestes columnes també utilitzarem un objecte d'aquesta classe.

Recuperar un objecte de la base de dades

JDBC defineix el mètode *getObject()* de la classe *ResultSet* que ens permet recuperar objectes de la base de dades. Quan ens connectam a *PostgreSQL* aquest mètode torna un objecte de tipus *PGobject*.

L'ideal seria que ens tornàs un objecte d'una classe que representàs aquest tipus. El procés és el següent:

1 Crear un POJO amb atributs per els camps del tipus definit a la base de dades:

1.1 Aquesta classe ha d'implementar la interfície PGobject. D'aquesta manera, per polimorfisme, el mètode getObject podrà tornar objectes d'aquesta classe.

1.2 Ha d'implementar els mètodes getValue() i setValue(String) definits per PGobject. (Veure més endavant)

1.3 No és imprescindible, però simplifica les coses tenir un atribut de classe que contengui el nom del tipus definit a la base de dades, incloent l'esquema i les cometes:

```
public static final String sqlType="\"proves\".\"t_telefon\"";
```

1.4 Els constructors del POJO han d'inicialitzar l'atribut *type* de *PGobject* amb el tipus *sql*, per exemple utilitzant l'atribut del punt anterior.

2 Configurar la connexió a la base de dades de manera que sàpiga quina classe Java correspon al tipus definit a la base de dades. D'aquesta manera el resultSet crearà un objecte d'aquest tipus per recollir el valor de la base de dades.

2.1 Per això hem d'utilitzar la connexió del driver de PostgreSQL en lloc de la de JDBC:

```
PGConnection conPG = (PGConnection) con;
```

2.2 I registrar el mapatge entre el tipus de la base de dades i la classe Java:

```
conPG.addDataType(Telefon.sqlType, Telefon.class);
```

3 En recórrer el *ResultSet*, per recuperar l'objecte utilitzam el mètode *getObject*(). Tornarà un objecte de la nostra classe, però per polimorfisme, serà amb l'aspecte d'un *PGobject*. Per tant, per recuperar l'objecte de la nostra classe haurem de fer un càsting explícit.

```
Telefon telefon = (Telefon) rs.getObject("telefon");
```

Per exemple, per recuperar les dades d'un client, inclòs el telèfon, podríem utilitzar el següent codi:

```
try (Connection con =
 DriverManager.getConnection("jdbc:postgresql://192.168.56.103:5432/
            demo?user=postgres&password=seCret_24");
  PreparedStatement stmt = con.prepareStatement("select telefon from
                    proves.\"Clients\" where id = ?");
        ) {
            // Registram el mapatge del tipus. Si tenim l'atribut type assignat
            // correctament no faria falta.
            PGConnection conPG = (PGConnection) con;
            conPG.addDataType(Telefon.sqlType, Telefon.class);
            //Assignam valor al paràmetre del PreparedStatement
            stmt.setInt(1, 5);
            //Executam la consulta
            try (ResultSet rs = stmt.executeQuery();) {
                    while (rs.next()) {
                            //Recuperam l'objecte guardat dins la taula
                            Telefon telefon = (Telefon)  rs.getObject(1);
                            if (telefon != null) {
                                    System.out.println("telefon = " + telefon);
                            }
                    }
            }
 } catch (SQLException e) {
            System.out.println("e.getMessage() = " + e.getMessage());
}
```

Mètode setValue()

L'objecte que hem guardat a la base de dades arriba al driver com una cadena de text. El driver crea un objecte de la classe que hem mapat,*Telefon* a l'exemple, i crida el seu mètode *setValue* passant-li per paràmetre aquesta cadena de test.

setValue s'hauria d'aprofitar per parsejar aquesta cadena de text i inicialitzar els atributs de l'objecte amb els valors inclosos a aquesta cadena de text. D'aquesta manera, *getObject* ja tornarà l'objecte inicialitzat.

El format d'aquesta cadena és el següent:

- Està tancada entre parèntesis ()
- Els valors estan separats per comes ,
- Les cadenes de text estan tancades entre cometes dobles " "

El tractament d'aquesta cadena sempre és igual, per tant es pot fer a una classe d'utilitat a part i es podrà reutilitzar a totes les classes que ho necessitin.

Per exemple:

```
public abstract class TractaValors {
    public static String[] parseValue(String value) {
        //Si el darrer atribut és nul tendrem ,) ho substituim per ,null)
        value = value.replaceAll(",\\)", ",null)");
        //si el primer atribut és nul tendrem (, ho substituim per (null,
        value = value.replaceAll("\\(,", "(null,");
        //Si un altre camp és null tendrem ,, ho substituim per ,null,
        value = value.replaceAll(",,", ",null,");
        //Llevam parèntesis
        String valor = value.substring(1, value.length() - 1);
        //si hi ha atributs amb comes dins del text ens complica el procés,
            //substituirem aquestes comes per un caràcter no imprimible.
        valor = tractaComes(valor);
        //Separam els diferents valors per ,
        String[] atributs = valor.split(",");
        //Eliminar " inicial i final si n'hi ha. Assignam nuls i recuperam les
            // comes.
        for (int i = 0; i < atributs.length; i++) {
            if ("null".equalsIgnoreCase(atributs[i])) {
                atributs[i] = null;
            } else if (atributs[i].charAt(0) == '"' &&
                                atributs[i].charAt(atributs[i].length() - 1) == '"') {
                atributs[i] = atributs[i].substring(1, atributs[i].length() - 1);
                atributs[i] = recuperaComes(atributs[i]);
            }
        }
        return atributs;
    }
```

Aquests mètode torna els valors de la cadena dins d'un array de cadenes. Al mètode *setValue* haurem d'assignar aquestes cadenes als atributs i si és necessari fer el tractament necessari per convertir-los a d'altres tipus, per exemple de String a int.

```
@Override
    public void setValue(@Nullable String value) throws SQLException {
        super.setValue(value);
        String[] atributs= TractaValors.parseValue(value);
        //Assignam valors als atributs.
        tipus=atributs[0];
        numero=atributs[1];
    }
```

El mètode tracta comes:

```
private static String tractaComes(String s) {
        boolean obertes = false;
        //Converteix la cadena a array de caràcters.
        char[] caracters = s.toCharArray();
        //Totes les , que estiguin entre " " les substitueix per el caracter 7
        for (int i = 0; i < caracters.length; i++) {
            if (caracters[i] == "") {
                obertes = !obertes;
            } else if (caracters[i] == ',' && obertes) {
                caracters[i] = (char) 7;
            }
        }
        //Torna una cadena creada a partir de l'array de caràcters.
        return new String(caracters);
}
```

Inserir o modificar un objecte a la base de dades

Per inserir o modificar una fila que tengui un camp de tipus objecte el més còmode és utilitzar un *PreparedStatement*.

Els passos a seguir són:

1. Crear l'objecte Java.
2. Passar-lo al *PreparedStatement* amb el mètode *setObject*. Aquest mètode utilitzarà el mètode *getValue()* per recuperar les dades de l'objecte en forma de cadena i passar-les al driver.

Per exemple per modificar el telèfon d'un client de la base de dades podríem fer:

```
try (Connection con =
   DriverManager.getConnection("jdbc:postgresql://192.168.56.103:5432/
  demo?user=postgres&password=seCret_25");
  PreparedStatement stmt = con.prepareStatement("update
            proves.\"Clients\" set telefon=? where id=?");
      ) {
          Telefon telefon = new Telefon("974456528", "Casa platja");

          PGConnection conPG = (PGConnection) con;
          conPG.addDataType(Telefon.sqlType, Telefon.class);

          stmt.setObject(1, telefon);
          stmt.setInt(2, 5);
          stmt.executeUpdate();
  } catch (SQLException e) {
          System.out.println("Error: "+e.getMessage());
  }
```

Mètode getValue()

Aquest mètode ha de tornar els valors dels atributs de l'objecte Java en el format que utilitza el driver explicat abans:

```
@Override
    public @Nullable String getValue() {
        String resultat = "(\""+ tipus + "\",";
        resultat += +numero + ')';
        return resultat;
    }
```

Tancam els atributs de tipus String entre " ". Els atributs d'altres tipus no n'han de dur.