A JDBC Statement típusok

Statement

A Statement interfész tartalmazza az alap metódusokat az SQL utasítások végrehajtásához és feldolgozásához. Ilyen típusú objektumhoz a Connection objektum createStatement metódusának meghívásával juthatunk. Ennek a metódusnak több fajtája van melyekkel optimalizálhatjuk az adatbázishoz való hozzáférést úgy, hogy megadjuk a ResultSet interfészben definiált konstansok közül valamelyiket:

- resultSetType:
    - TYPE_FORWARD_ONLY – Az eredménytábla kurzora csak elrefele mozgatható.
    - TYPE_SCROLL_INSENSITIVE – Az eredménytábla kurzora mindkét irányba mozgatható de a mások által eszközölt módosításokat nem képes észlelni. Elfordulhat ugyanis, hogy míg mi az eredménykészletet feldolgozzuk, annak egy vagy több sorát egy másik program módosítja.
    - TYPE_SCROLL_SENSITIVE – Az eredménytábla kurzora mindkét irányba mozgatható és képes észlelni a mások által eszközölt módosításokat.
- resultSetConcurrency:
    - CONCUR_READ_ONLY – Az eredménytábla nem változtatható meg.
    - CONCUR_UPDATABLE – Az eredménytábla megváltoztatható az update<típus> kezdet metódusokkal.
- resultSetHoldability – Bizonyos adatbáziskezelk esetén a ResultSet objektum bezárulhat, ha meghívjuk a Connection.commit metódust. Az alábbi konstansokkal beállíthatjuk ezt a viselkedést.
    - HOLD_CURSORS_OVER_COMMIT – nem zárul be a ResultSet objektum
    - CLOSE_CURSORS_AT_COMMIT – bezárulhat a ResultSet objektum

A Statement futtatása

A Statement objektumot többféleképpen lehet futtatni, mindegyik metódus paraméteréül magát az SQL utasítást kell megadni. Ezek mindegyikének meghívásakor automatikusan lezárulnak az esetlegesen korábban létrejött, ugyanezen Statement objektumhoz kötd eredménytáblák, ezért ügyelni kell, hogy új SQL utasítás végrahajtása eltt a korábbi ereménytáblák feldolgozásra kerüljenek.

- execute – ha az adatbázis által visszaadott értékek els komponense eredménykészlet akkor visszatérési értéke igaz egyébként hamis. Akkor érdemes használni, ha nem tudjuk, hogy a paraméteréül átadott SQL utasítás milyen jelleg (select, insert, update, stb), ekkor az execute metódus visszatérési értékének függvényében elágaztathatjuk programunkat a megfelel feldolgozás érdekében:
    - a getResultSet metódussal juthatunk hozzá az eredménykészlethez
    - a getUpdateCount metódussal juthatunk hozzá a megváltoztatott sorok számához
    - a getMoreResults metódussal lekérhetjük az adatbázis által visszaadott következ komponenst, ha ez eredménykészlet akkor a visszatérési értéke igaz egyébként hamis.
- executeUpdate – visszatérési értéke egy egész szám mely a megváltoztatott sorok számát jelzi. Akkor érdemes használni, ha tudjuk, hogy a végrehajtandó SQL utasítás adatmanipulációkat fog elkövetni (insert, update, delete). Abban az esetben, ha adatdefiníciós utasítást hajtunk végre (create, drop) a visszatérési érték mindig nulla.
- executeQuery – visszatérési értéke egy ResultSet objektum mely tartalmazza magát az eredménykészletet. Biztosított, hogy ez az érték soha nem null, még akkor sem, ha a lekérdezés eredménye egyetlen egy sort sem tartalmaz. Ekkor egy üres ResultSet objektumot kapunk.

A Statement bezárása

A Statement objektumot ugyanezen objektum close metódusának megívásával zárhatjuk be melyrl minden esetben nekünk kell gondoskodni, különben egy id után azt fogjuk tapasztalni, hogy elfogyott a memóriánk. A Statement objektum bezárásakor automatikusan bezáródnak az esetlegesen létez ResultSet objektumok is, így azokat nekünk külön nem kell bezárnunk, bár lehetségünk van rá a ResultSet objektum close metódusának meghívásával.

PreparedStatement

A PreparedStatement interfész a Statement interfészt terjeszti ki újabb tulajdonságokkal melyek lehetvé teszik az elfordított utasítások használatát adatbázisunkban. Míg a Statement interfész alkalmazása esetén minden egyes újabb utasítás végrehajtásának alkalmával át kell adni az adatbáziskezelnek az utasítás teljes szövegét, illetve az adatbáziskezelnek fel kell dolgoznia azt, addig a PreparedStatement interfész alkalmazása esetén elegend ezt egy alkalommal megtenni, majd a késbbiekben – az újbóli használat alkalmával – már csak a paramétereket kell átadni az adatbázisban lév elfordított utasításnak. Ennek a technológiának az elnyét nyilván akkor tudjuk leginkább kamatoztatni, ha ugyanazt az utasítást ciklikusan többször hajtjuk végre egymás után más-más paraméterekkel, mivel az utasításunk elfordított (csre töltött) állapotban figyel az adatbáziskezelben, melynek csak a paramétereit kell megadni és már futtatható is.

PreparedStatement létrehozása

Ezért az SQL utasítás szövegét már a PreparedStatement objektumot létrehozó metódus paramétereként meg kell adni, melyben a paraméterek helyét '?' karakterrel kell kijelölnünk. Ez a paraméter átadódik a PreparedStatement interfészt implementáló osztály konstruktorának, mely továbbítja azt az adatbázis-kezel felé, amely létrehozza az elkészített utasítást. PreparedStatement objektumhoz a Connection objektum prepareStatement metódus meghívásával juthatunk, melynek – mint ahogy a Statement interfész bemutatásánál láthattuk – szintén többféle típusa van melyekkel speciális, a ResultSet objektummal kapcsolatos tulajdonságokat állíthatunk be.

PreparedStatement futtatása

A PreparedStatement végrehajtása eltt minden kijelölt paramétert inicializálni kell, melyeket a set<típus> kezdet metódusokkal tehetünk meg úgy, hogy a metódus els paramétereként egy egész számmal hivatkozunk az SQL szövegében kijelölt paraméterre, a metódus második paraméterében pedig megadjuk magát az értéket. A balesetek elkerülése végett, üdvös dolog ha ezt a tevékenységet megelzi egy clearParameters metódus meghívása, amely törli az összes, korábban beállított értéket. Mindezek után végrehajthatjuk SQL utasításunkat a PreparedStatement objektum execute, executeQuery vagy executeUpdate metódusának megívásával melyeknek különbözségeirl fentebb írtam.

Egy konkrét példa a PreparedStatement interfész alkalmazására:

```
PreparedStatement pstmt = con.prepareStatement("INSERT INTO PICTURE (KEP, KOZEPES_KEP, KEP_IDX, SORREND, ALBUM,
VISIBLE) VALUES (?, ?, ?, ?, ?, ?)");
pstmt.clearParameters();
pstmt.setBlob(1, sbKep);
pstmt.setBlob(2, sbKozepesKep);
pstmt.setBlob(3, sbKepIndex);
pstmt.setInt(4, sorrend);
pstmt.setString(5, album);
pstmt.setBoolean(6, visible);
pstmt.execute();
pstmt.close();
```

PreparedStatement bezárása

A PreparedStatement objektum bezárására ugyanazok a szabályok érvényesek mint a Statement objektum bezárására.