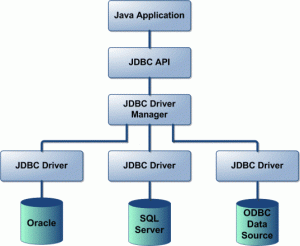
KẾT NỐI CƠ SỞ DỮ LIỆU BẰNG JDBC

1: Trình điều khiển JDBC:

* Hiện nay có rất nhiều hệ quản trị cơ sở dữ liệu khác nhau như là SQL server, MySQL, Oracle, MS Access, FoxPro,...
* Để truy cập các DBMS khác nhau từ chương trình viết bằng Java thì cần có các JDBC driver tương ứng.
* Hãng Sun đã đưa ra 4 loại JDBC driver.
* Tương tác căn bản nhất của JDBC:
* Mở 1 kết nối đến csdl (Open connection)
* Thực thi truy vấn SQL( Execute SQL).
* Xử lý dữ liệu (Process results).
* Đóng kết nối (close connection)
* Kiến trúc của JDBC gồm 2 tầng: tầng đâù tiên là JDBC API, có nhiệm vụ chuyển các câu lệnh SQL cho bộ quản lý trình điều khiển JDBC; tầng thứ 2 là các JDBC Driver API,thực hiện nhiệm vụ liên hệ với các trình điều khiển của hệ quản trị CSDL cụ thể như SQL-Server, MS-Access, MySQL, IBM-DB2, Oracle,…



* Các trình điều khiển có nhiệm vụ là yêu cầu hệ quản trị CSDL thực hiện các câu lệnh SQL.Các trình điều khiển trong JBDC là các đoạn chương trình do nhà sản xuất hệ quản trị CSDL cung cấp hoặc do các nhà cung cấp ở bên thứ 3 đưa ra.Các trình điều khiển JDBC được chia ra làm 4 loại:
* Cầu nối JDBC-ODBC (Bridge)
* Trình điều khiển thuần túy Java (Native-API).
* Trình điều khiển JDBC-Net, Pure Java.
* Trình điều khiển Native-Protocol, Pure Java.

2: các bước lập trình, truy vấn dữ liệu:

* Nạp trình điều khiển
* Thiết lập kết nối
* Tạo đối tượng Statement
* Thực hiện vấn tin
* Xử lý kết quả trả về
* Đóng kết nối

3: Nạp trình điều khiển:

* Cách dùng:

Try{

Class.forName(“Database driver name”);

}Catch(ClassNotFoundException){

System.out.println(“Driver not found: ” + e.getMessage());

}Catch(SQLException e){

System.out.println(“SQL Exception: ” + e.getMessage());

}

* Ex: Trình điều khiển của MySQL:

**Class.forName(“org.gjf.mm.mysql.Driver”);**

4: Thiết lập kết nối:

* Để thiết lập kết nối ta gọi phương thức tĩnh getConnection() của lớp DriverManager, khi đó trả về 1 thể hiện của lớp Connection, theo dạng như sau:

String username = “sa”;

String password = “secret”;

Connection conn = DriverManager.getConnection(dbUrl, username, password);

Trong đó:

* dbUrl: là chuỗi kết nối đến cơ sở dữ liệu
* username: tên người dùng đăng nhập
* password: mật khẩu đăng nhập

5: Tạo đối tượng Connection để tạo đối tượng Statement.

* Sử dụng đối tượng Connection để tạo đối tượng Statement

Statement s = conn.createStatement();

* Đối tượng này có nhiệm vụ gửi các câu lệnh sql đến CSDL
* Cùng một đối tượng Statement có thể sử dụng cho nhiều câu lệnh SQL khác nhau.
* Có 3 phương thức thực thi:

+ executeQuery(String Sql)

+ executeUpdate(String Sql)

+ execute(String Sql)

6: Nhập dữ liệu qua JDBC

* Sử dụng phương thức excuteUpdate() của Statement để cập nhật dữ liệu qua JDBC
* excuteUpdate() trả về số dòng bị ảnh hưởng sau khi câu lệnh được thực thi.

7: Đọc dữ liệu từ CSDL:

* Sử dụng phương thức executeQuery() của Statement để đọc dữ liệu từ cơ sở dữ liệu.
* Dữ liệu đọc được sẽ được lưu trong ResultSet.
* Có thể di chuyển tới lui trong ResultSet để lấy dữ liệu ra.

8: Các phương thức của ResultSet:

* next(): di chuyển con trỏ đến dòng kế, trả về true nếu có dòng kế tiếp, false nếu đến cuối ResultSet.
* previous(): di chuyển con trỏ đến dòng trước
* first(): di chuyển con trỏ đến dòng đầu tiên
* last(): di chuyển con trỏ đến dòng cuối cùng
* beforeFirst(): di chuyển con trỏ đến vị trí trước dòng đầu tiên
* afterLast(): di chuyển con trỏ đến sau dòng cuối cùng
* relative(int rows): di chuyển con trỏ tương đối với vị trí hiện tại của nó với số dòng là rows
* absolute(int row): di chuyển con trỏ đến dòng thứ row

9: Lấy dữ liệu từ ResultSet:

* dùng phương thức getXXX(String colname)

+ XXX là kiểu dữ liệu được trả về

+ colname là tên của cột cần lấy dữ liệu ra

* Ví dụ:

+ String name = rs.getString(“NAME”);

+ double val = rs.getDouble(“PRICE”);

10: Sử dụng PrepareStatement

* Dùng trong trường hợp ta cần thực hiện nhiều câu lệnh SQL có cấu trúc tương tự nhau, chỉ có giá trị là thay đổi
* PrepareStatement có thể sử dụng để soạn trước câu lệnh có sẵn cấu trúc cần thiết
* Giá trị sẽ được đưa vào như những đối số khi câu lệnh thực thi.
* Phòng chống lỗi SQL Injection

11: sử dụng Transaction:

* Mặc định, sau mỗi câu lệnh SQL được thực thi qua JBDC, dữ liệu sẽ được cập nhật ngay vào CSDL
* Có những trường hợp, ta muốn dữ liệu chỉ được cập nhật vào CSDL sau khi một số câu lệnh SQL được thực hiện.
* Ví dụ: đối với trang ứng dụng bán hàng qua mạng, để CSDL được thống nhất, ta chỉ muốn lưu các dữ liệu liên quan tới một đơn đặt hàng cùng một lúc
* Một nhóm các câu lệnh như thế được gọi là một giao dịch

12: Cài đặt Transaction:

* Trước tiên, không dùng chế độ commit tự động từ Connection object

conn.setAutoCommit(false);

* Thực hiện các câu lệnh trong 1 giao dịch
* Thực hiện COMMIT(lưu) CSDL

conn.commit();

* Nếu k cần dùng ở chế độ giao dịch nữa, ta nên trả lại chế độ COMMIT tự động

conn.setAutoCommit(true);

13: Hủy 1 Transaction

* Trong quá trình thực hiện 1 giao dịch, nếu có sai sót nào xảy ra, ta có thể hủy giao dịch đang được thực hiện nửa chừng bằng cách sử dụng rollback
* conn.rollback sẽ hủy dử liệu đến thời điểm nó được commit gần nhất

+ tùy thuộc vào hệ quản trị cơ sở dữ liệu mà cách rollback() có thể khác nhau

+ Dữ liệu sau khi được commit sẽ k hủy được bằng rollback();