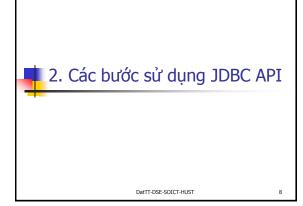




Database URL

- Được sử dụng để tạo một kết nối tới database
 - Có thể chứa server, port, protocol, etc.
- Cú pháp:
 - jdbc:subprotocol_name:driver_dependant_databasename
- Ví du
 - Oracle thin driver
 - jdbc:oracle:thin:@machinename:1521:dbname
 - Derhy
 - jdbc:derby://localhost:1527/sample
 - Pointhase
 - jdbc:pointbase:server://localhost/sample

DatTT-DSF-SOICT-HUST





Các bước sử dụng JDBC

- B1. Load JDBC driver cho từng loại CSDL
- B2. Lấy đối tương Connection
- B3. Lấy đối tương Statement
- B4. Thực hiện câu truy vấn, câu lênh update
- B5. Đoc kết quả trả về
- B6. Đoc các Meta-data (tùy chon)
- B7. Đóng đối tượng Statement và đối tượng Connection

DatTT-DSE-SOICT-HUST



B1. Load JDBC driver cho từng loai CSDL

- Để load về driver cho CSDL và đăng ký nó với DriverManager, cần load class tương ứng
 - Class.forName(<database-driver>)

```
try {
    // This loads an instance of the Pointbase DB Driver.
    // The driver has to be in the classpath.
    Class.forName("org.apache.derby.jdbc.ClientDriver");
}catch (ClassNotFoundException cnfe) {
    System.out.println("" + cnfe);
}
```

TT-DSE-SOICT-HUST



B2. Lấy ra đối tượng Connection

- Lớp DriverManager chịu trách nhiệm tạo kết nối tới CSDL
 - Sử dụng DataSource là cách hay dùng hơn khi muốn lấy ra một đối tượng connection (trình bày ở phần sau)
- Ví dụ tạo kết nối tới CSDL như sau:

DriverManager & Connection

- java.sql.DriverManager
 - getConnection(String url, String user, String password) throws SQLException
- java.sql.Connection
 - Statement createStatement() throws SQLException
 - void close() throws SQLException
 - void setAutoCommit(boolean b) throws SQLException
 - void commit() throws SQLException
 - void rollback() throws SQLException



B3. Lấy ra đối tượng Statement

- Tạo một đối tượng Statement từ đối tượng Connection
 - java.sql.Statement
 - ResultSet executeQuery(string sql)
 - int executeUpdate(String sql)
 - Ví du:
 - Statement statement = connection.createStatement();
- Cùng đối tượng Statement có thể được dùng cho nhiều queries không liên quan tới nhau

DatTT-DSE-SOICT-HUST



B4. Thực thi các câu truy vấn/các lệnh

- Từ đối tượng Statement, 2 lệnh được sử dụng nhiều nhất là
 - (a) QUERY (SELECT)
 - ResultSet rs = statement.executeQuery("select * from customer_tbl");
 - (b) ACTION COMMAND (UPDATE/DELETE)
 - int iReturnValue = statement.executeUpdate("update manufacture_tbl set name = 'IBM' where mfr_num = 19985678");

DatTT-DSE-SOICT-HUST



B5. Xử lý kết quả nhận về

- Duyệt trên ResultSet để xử lý thông tin
 - java.sql.ResultSet
 - boolean next()
 - xxx getXxx(int columnNumber)
 - xxx getXxx(String columnName)
 - void close()
- Đầu tiên, con trỏ lặp nằm ở trước hàng đầu tiên
 - LTV cần gọi phương thức next() để chuyển con trỏ đến hàng đầu tiên

DatTT-DSE-SOICT-HUST



B5. Xử lý kết quả nhận về (2)

- Khi đã có ResultSet, LTV dễ dàng xử lý dữ liêu
- Lưu ý: Chỉ số của ResultSet bắt đầu từ 1

```
// Wrong this will generate an error
String value0 = rs.getString(0);

// Correct!
String value1 = rs.getString(1);
int value2 = rs.getInt(2);
int value3 = rs.getInt("ADDR_LN1");
```

DatTT-DSE-SOICT-HUST



B5. Xử lý kết quả nhận về (3)

- Muốn lấy dữ liệu từ ResultSet, sử dụng phương thức getXXX() cho phù hợp
 - getString()
 - getInt()
 - getDouble()
 - getObject()
- Mỗi kiểu dữ liệu trong java.sql.Types, đều có phương thức getXXX tương ứng

DatTT-DSF-SOICT-HUST



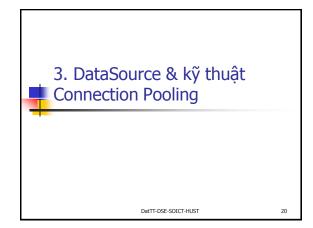
B6. Đọc metadata của ResultSet và metadata của CSDL (tùy chọn)

- Khi đã có đối tượng ResultSet hoặc Connection, LTV có thể lấy về metadata của CSDL hoặc của câu truy vấn
- → Đem lại thông tin hữu ích về dữ liệu lấy về, hoặc về CSDL đang sử dụng
 - ResultSetMetaData rsMeta = rs.getMetaData();
 - DatabaseMetaData dbmetadata = connection.getMetaData();
 - Có khoảng 150 phương thức trong lớp DatabaseMetaData

DatTT-DSE-SOICT-HUST

10

ResultSetMetaData meta = rs.getMetaData(); //Return the column count int iColumnCount = meta.getColumnCount(); for (int i = 1; i <= iColumnCount ; i++){ System.out.println("Column Name: " + meta.getColumnName(i)); System.out.println("Column Type" + meta.getColumnType(i)); System.out.println("Display Size: " + meta.getColumnDisplaySize(i)); System.out.println("Precision: " + meta.getPrecision(i)); System.out.println("Scale: " + meta.getScale(i)); }





3. DataSource & kỹ thuật Connection Pooling

- 3.1. Giao diện DataSource và đối tượng DataSource
- 3.2. Các thuộc tính của đối tương DataSource
- 3.3. Đăng ký JNDI của đối tượng DataSource
- 3.4. Connection Pooling
- 3.5. Lấy về đối tượng DataSource

DatTT-DSE-SOICT-HUST



3.1. Giao diện javax.sql.DataSource và đối tương DataSource

- Từng nhà cung cấp Driver sẽ thực thi cài đặt interface
- Đối tượng DataSource dùng để tạo các kết nối CSDL (database connections)

SE-SOICT-HUST



Giao diện javax.sql.DataSource và đối tương DataSource

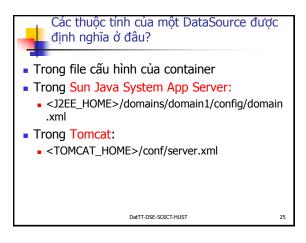
- Có 3 kiểu cài đăt interface
 - Basic implementation: cung cấp đối tượng Connection chuẩn tắc
 - Connection pooling implementation: cung cấp đổi tượng Connection tự động nằm trong connection pooling
 - Distributed transaction implementation: cung cấp đối tượng Connection có thể dùng được cho các giao dịch (transactions) phân tán, hầu hết nằm trong connection pooling

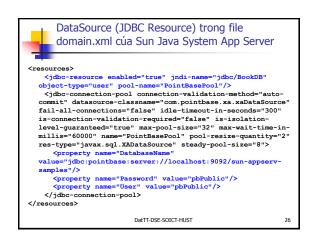
DatTT-DSE-SOICT-HUST



3.2. Các thuộc tính của đối tượng DataSource

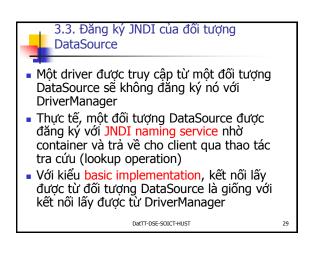
- Một đối tượng DataSource có các thuộc tính có thể sửa đổi khi cần thiết-được định nghĩa trong file cấu hình của container
 - Địa chỉ của database server
 - Tên của database
 - Network protocol được sử dụng để giao tiếp với server
- Lợi ích: vì thay đổi được thuộc tính của DataSource, các đoạn code truy cập tới DataSource đó không cần thay đổi
- Trong Sun Java System Application Server (và GlassFish V2), một data source được gọi là một JDBC resource

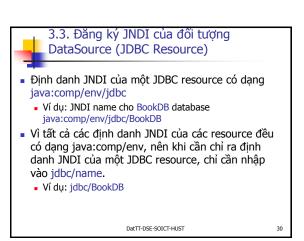














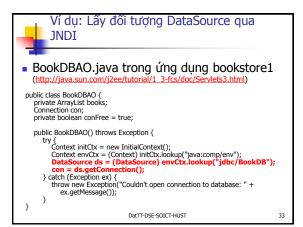
3.4. Tại sao cần Connection Pooling?

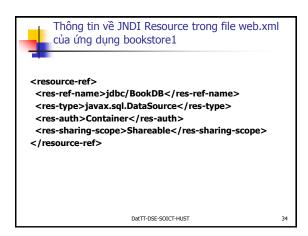
- Kết nối CSDL là tài nguyên có chi phí cao, và bi han chế
 - Sử dụng connection pooling, chỉ cần 1 số nhỏ các connections có thể được dùng chung cho 1 số lớn clients
- Tạo và hủy các kết nối CSDL là thao tác có chi phí cao
 - Sử dụng connection pooling, có sẵn một tập các kết nối sẵn dùng, hạn chế tạo mới/hủy các connection

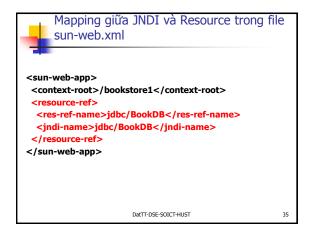
DatTT-DSE-SOICT-HUST

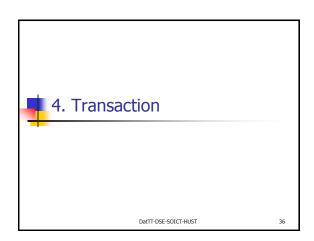


- Úng dụng thực hiện thao tác tra cứu JNDI (JNDI lookup) để lấy về đối tượng DataSource
- Đối tượng DataSource sau đó dùng để lấy ra đối tương Connection
- Trong file web.xml của ứng dụng, cần chỉ ra thông tin về external resource, & về đối tượng DataSource
- Trong Sun Java System App server, c\u00e4n \u00e4n \u00e4
 - Tăng tính flexibility











Transaction

- Việc commit từng câu lệnh một ngay khi được yêu cầu tiêu tốn nhiều thời gian
- Khi thiết lập AutoCommit là false, LTV có thể cập nhật CSDL nhiều lần, sau đó commit toàn bộ trong một transaction
- Ngoài ra, nếu các lệnh phụ thuộc lẫn nhau, toàn bộ transaction có thể được rolled back



Các phương thức trong JDBC Transaction

- setAutoCommit()
 - Nếu thiết lập là true, tất cả các lệnh thực thi (executed statement) sẽ được commit ngay lập tức
- commit()
 - Chỉ hợp lệ khi đã thiết lập setAutoCommit(false)
 - Commit tất cả các thao tác được thực hiện, từ lúc mở một Connection hoặc từ lời gọi commit() hoặc rollback() gần nhất
- rollback()
 - Chỉ hợp lệ khi đã thiết lập setAutoCommit(false)
 - Hủy tất cả các thao tác vừa thực hiện

DatTT-DSE-SOICT-HUST



Ví du về Transactions

Connection connection = null; try {

connection =

DriverManager.getConnection("jdbc:oracle:thin:@machinename:1521:db name","username","password");

connection.setAutoCommit(false);

PreparedStatement updateQty = connection.prepareStatement("UPDATE STORE_SALES SET QTY = ? WHERE ITEM CODE = ? ");



Ví du về Transactions (2)

int [][] arrValueToUpdate = { {123, 500} , {124, 250}, {125, 10},

int iRecordsUpdate = 0;

{126, 350} };

for (int items=0 ; items < arrValueToUpdate.length ; items++) { int itemCode = arrValueToUpdate[items][0]; int qty = arrValueToUpdate[items][1];

Ví du về Transactions (3)

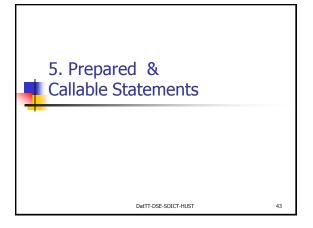
```
updateQty.setInt(1,qty);
  updateQty.setInt(2,itemCode);
  iRecordsUpdate += updateQty.executeUpdate();
connection.commit();
System.out.println(iRecordsUpdate +
```

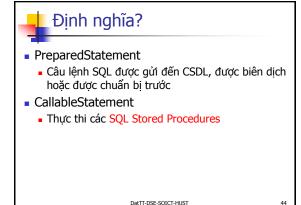
" record(s) have been updated");

} catch(SQLException sqle) { System.out.println("" + sqle);

Ví du về Transactions (4)

```
connection.rollback();
} catch(SOLException sqleRollback) {
       System.out.println("" + sqleRollback);
finally {
      try {
           connection.close();
        catch(SQLException sqleClose) {
           System.out.println("" + sqleClose);
```







PreparedStatement

- Đôi khi, nhiều câu lệnh có cấu trúc tương tự nhau, chỉ có giá tri là thay đổi
- PreparedStatement có thể được sử dụng để soạn trước câu lênh có cấu trúc cần thiết
- PreparedStatement có thể nhận các tham số VÀO, hoạt động tương tự các đối số cho một phương thức
- PreparedStatements deal with data conversions that can be error prone in straight ahead, built on the fly SQL
 - handling quotes and dates in a manner transparent to the developer

tTT-DSE-SOICT-HUST



Các bước làm việc với PreparedStatement

- B1: Tao DB connection như bình thường
- B2: Tạo đối tượng prepared statement từ connection

```
PreparedStatement updateSales = con.prepareStatement("UPDATE OFFER TBL SET QUANTITY = ? WHERE ORDER NUM = ? ");

// "?" are referred to as Parameter Markers

// Parameter Markers are referred to by number,

// starting from 1, in left to right order.

// PreparedStatement's setXXX() methods are used to set

// the IN parameters, which remain set until changed.

DatTH-DSE-SOICT-HUST
```

Các bước làm việc với PreparedStatement (2)

- B3. Truyền vào các đối số theo đúng vị trí updateSales.setInt(1, 75);
 updateSales.setInt(2, 10398001);

patTT-DSF-SOICT-HUST



Các bước làm việc với PreparedStatement (3)

 Nếu thuộc tính AutoCommit ilà true, khi thực thi câu lệnh, thay đổi sẽ được commit. Từ đó về sau, có thể sử dụng lại đối tượng Prepared Statement này.

```
updateSales.setInt(1, 150);
updateSales.setInt(2,10398002);
```



PreparedStatement

 Nếu đối tượng prepared statement có kiểu là câu lệnh select, sau khi thực thi và lấy về kết quả, tiến hành duyêt trên ResultSet như phần trước

```
PreparedStatement itemsSold =
    con.prepareStatement("select o.order_num,
    o.customer_num, c.name, o.quantity from order_tbl
    o, customer_tbl c where o.customer_num =
    c.customer_num and o.customer_num = ?;");
    itemsSold.setInt(1,10398001);
    ResultSet rsItemsSold = itemsSold.executeQuery();
    while (rsItemsSold.next()){
        System.out.println( rsItemsSold.getString("NAME")
        + " sold "+ rsItemsSold.getString("QUANTITY") + "
        unit(s)");
        DatTI-DSE-SOLCT-HUST 49
```



CallableStatement

- Là giao diện được sử dụng để thực thi các SQL stored procedures
- Một stored procedure là một nhóm các câu lệnh SQL thực hiện một công việc nào đó
- Stored procedures được sử dụng để đóng gói một tập các thao tác hoặc truy vấn trên một database server.

DatTT-DSE-SOICT-HUST



CallableStatement (2)

- Một đối tượng CallableStatement chứa lời gọi đến một stored procedure; nó không chứa chính stored procedure này.
- Ví dụ: gọi đến stored procedure SHOW_SUPPLIERS sử dụng connection con và trả về ResultSet

CallableStatement cs = con.prepareCall("{call SHOW_SUPPLIERS}"); ResultSet rs = cs.executeQuery();

T-DSE-SOICT-HUST



Ví du CallableStatement

 Ví dụ sử dụng các tham số IN, OUT và INOUT

```
// set int IN parameter
cstmt.setInt( 1, 333 );
// register int OUT parameter
cstmt.registerOutParameter( 2, Types.INTEGER );
// set int INOUT parameter
cstmt.setInt( 3, 666 );
// register int INOUT parameter
cstmt.registerOutParameter( 3, Types.INTEGER );
//You then execute the statement with no return value
cstmt.execute(); // could use executeUpdate()
// get int OUT and INOUT
int iOUT = cstmt.getInt( 2 );
int iINOUT = cstmt.getInt( 3 );
```

Ví du Stored Procedure

DatTT-DSF-SOICT-HUST

Ví dụ với Oracle DB

```
try {
    Connection connection = DriverManager.getConnection("");
    CallableStatement queryreport = connection.prepareCall("{
    ? = call SRO21208_PKG.QUEUE_REPORT ( ? , ? , ? , ? , ? , ? , ? , ? , ? ) }");
    queryreport.registerCutParameter(1,OracleTypes.CURSOR);
    queryreport.setInt(2,10);
    queryreport.setString(3, "000004357");
    queryreport.setString(4, "01/07/2003");
    queryreport.setString(5, "N");
    queryreport.setString(6, "N");
    queryreport.setString(7, "N");
    queryreport.setString(7, "N");
    queryreport.setInt(8, 2);
```

#