Java DataBase Connectivity (JDBC)

Vittayasak Rujivorakul

Lecture 13

Java DataBase Connectivity (JDBC)

Vittayasak Rujivorakul

Lecture 13

Java & Databases

- เพื่อตอบสนองการทำงานกับ Web
- นอกจากทำงานในลักษณะ application และ applet ได้ดีแล้ว Java ยังเป็นภาษาที่สามารถใช้ในการพัฒนา DB applications ทั่วไปได้
- ผู้ผลิต DBMS (OODB, RDB) สนับสนุนการเชื่อมโยงกับ DB โดย Java
- ODMG กำลังพัฒนา Java language binding (http:/ / www.odmg.org/odmg93/updates/chap7.html) (SQLJ)

JDBC

- เป็น Java API สำหรับทำงานกับ SQL statements
- เป็นมาตรฐานของ Javasoft สำหรับ RDB API, โดยอาศัยพื้นฐานจาก Microsoft ODBC และได้รับการสนับสนุนจาก IBM, Informix, O2, ODI, Oracle, Borland etc.
- กำหนด Java classes เพื่อใช้ทำการเชื่อมโยงกับ RDB, ทำการ run SQL statements และคืนผลลัพธ์, ให้ schema information
- ทำให้สามารถ embedded SQL ใน Java programs ได้
- JDBC Driver Products : http: / / www.javasoft.com/products/jdbc

ขั้นตอนการทำงานกับ JDBC

1. Importing Packages :
import java.sql. * / / JDBC
2. ลงทะเบียน หรือ load JDBC Driver :
DriverManager.registerDriver (new oracle.jdbc.driver.OracleDriver ( )) ;
OR Class.forName (“oracle.jdbc.driver.OracleDriver”) ;
3. เปิดการเชื่อมโยงกับ Database : สำหรับ JDBC Thin Client :
Connection conn = DriverManager.getConnection
(“jdbc:oracle:thin: @myhost:1521:orcl”, “scott”, “tiger”) ;
สำหรับ JDBC OCI Driver :
Connection conn - DriverManager.getConnection
(“jdbc : oracle : oci8 : scott / tiger@) ;
OR Connection conn = DriverManager.getConnection
(“jdbc:oracle:oci8:@”, “scott”, “tiger”) ;

ขั้นตอนการทำงานกับ JDBC (ต่อ)

4. สร้าง JDBC Statement Object :

```
Statement  stmt  =  conn.createStatement ( ) ;
```

5. ทำการ run Query และรับผลลัพธ์กลับคืนใน Result Set Object

```
ResultSet rset  =  stmt.executeQuery ("SELECT ename
        FROM  emp") ;
```

6. จัดการกับผลลัพธ์ :

```
while  (rset.next ( ) )  {
        System.out.println  (rset.getString (1) ) ;

}
```

7. ทำการปิด Result Set และ Statement Objects :

```
rset.close ( )
stmt.close ( ) ;
```

8. ทำการปิดการเชื่อมโยงกับ Database :

```
conn.close ( ) ;
```


Type Mapping : JDBC, Java Native & Oracle

Standard JDBC Datatype	Java Native Datatype	SQL Datatype	Oracle Extensions -- SQL Database
java.sql.Type.CHAR	java.lang.String	CHAR	oracle.sql.CHAR
java.sql.Type.VARCHAR	java.lang.String	VARCHAR2	oracle.sql.CHAR
java.sql.Type.LONGVARCHAR	java.lang.String	LONG	oracle.sql.CHAR
java.sql.Type.NUMERIC	java.math.BigDecimal	NUMBER	oracle.sql.NUMBER
java.sql.Type.DECIMAL	java.math.BigDecimal	NUMBER	oracle.sql.NUMBER
java.sql.Type.BIT	boolean	NUMBER	oracle.sql.NUMBER
java.sql.Type.TINYINT	byte	NUMBER	oracle.sql.NUMBER
java.sql.Type.SMALLINT	short	NUMBER	oracle.sql.NUMBER
java.sql.Type.INTEGER	int	NUMBER	oracle.sql.NUMBER
java.sql.Type.BIGINT	long	NUMBER	oracle.sql.NUMBER
java.sql.Type.REAL	float	NUMBER	oracle.sql.NUMBER
java.sql.Type.FLOAT	double	NUMBER	oracle.sql.NUMBER
java.sql.Type.DOUBLE	double	NUMBER	oracle.sql.NUMBER
java.sql.Type.BINARY	byte[]	NUMBER	oracle.sql.NUMBER
java.sql.Type.VARBINARY	byte[]	RAW	oracle.sql.RAW
java.sql.Type.LONGVARBINARY	byte[]	LONGRAW	oracle.sql.NUMBER
java.sql.Type.DATE	java.sql.Date	DATE	oracle.sql.DATE
java.sql.Type.TIME	java.sql.Time	DATE	oracle.sql.DATE
java.sql.Type.TIMESTAMP	java.sql.Timestamp	DATE	oracle.sql.DATE

การ execute statements

- การสั่งให้ run staement นั้นมี 2 ลักษณะ
  - stmt.**executeUpdate** (Strint SQL) ใช้กับคำสั่งเพื่อ**สร้าง**ตารางหรือ**แก้ไข**โดยคำสั่ง Insert, Update, Delete
  - stmt.**executeQuery** (String SQL) ใช้กับการ **Select** ซึ่งจะให้คำตอบเป็น ResultSet

ตัวอย่าง Execute Statement

```
java.sql.Statement  stmt  =  conn.createStatement  ( )
ResultSet  r = stmt.executeQuery ("Select a, b, c FROM Table1") ;
while  (r.next  ( ) )  {
    / / print the values for the current row.
    Int  I  =  r.getInt  ("a") ;
    String  s  =  r.getStrint ("b") ;
    float  f  =  r.getFloat  ("c") ;
    System.out.println ("ROW  =  "  +  I  +  "  "  +  s  +  "  "  +  f) ;
    }

java.sql.  Statement stmt  =  conn.createStatement ( ) ;
stmt.executeUpdate ("CREATE TABLE CUST "  +
                    "(NAME VARCHAR(30), ADDRESS VARCHAR(30) ;"
                    "AGE INTEGER)"  ) ;
```

การใช้ PreparedStatement

- PrepareStatement Object ใช้เพื่อเตรียม SQL stmt ซึ่งรู้ล่วงหน้า
- PrepareStatement จะส่งคำสั่งไปยัง DBMS เพื่อ pre-compile จึงทำงานในทันทีที่สั่งให้ execute
- สามารถรับ parameters โดยใช้ **?** เพื่อเพิ่มความยืดหยุ่นให้กับ stmt.

```
Java.aql.PreparedStatement stmt = conn.prepareStatement (
    "UPDATE SALETABLE SET AMONT  =  ?  Where CUSTID = ?") ;
stmt.setInt (1, 1200) ;  stmt.setString (2, "AB1230");
int rows  =  stmt.executeUpdate ( ) ;
```


Transaction

- เพื่อรับประกันว่า กลุ่มของ statement จะทำงานเป็น transaction โดยใช้ commit

```
conn.setAutoCommit (false) ;
Statement  stmt  =  new conn.createStatement ( ) ;
stmt.executeUpdate ("INSERT INTO CUST " +
    "VALUES ("NidNoi", "1518 Piboolsongkram RD." , 20) ;
stmt.executeUpdate ("INSERT INTO ORDER "  +
    "VALUES ("NidNoi", "Bicycle", "1" 3000) ;
conn.commit ( ) ;
conn.setAutoCommit (true) ;
```


การใช้ MetaData

- สามารถที่จะหารายละเอียดเกี่ยวกับ **metadata** หรือ **schema ของ ResultSet** ได้จาก **ResultSetMetaData object**

```
ResultSet  rs = stmt.excuteQuery ("SELECT * FROM CUST") ;
ResultSetMetaData  rm  =  rs.getMetaData ( ) ;
```

- methods ใน ResultSetMetaData
  - getColumnCount ( ) คืนจำนวน columns ใน resultset
  - getColumnLabel (int) คืนชื่อที่แนะนำสำหรับการพิมพ์ column นั้น
  - getColumnName (int) คืนชื่อของ column นั้น
  - getColumnTypeName (int) คืนชื่อของชนิดของ column นั้น