Lecture 10: Database Connectivity - JDBC

Wendy Liu
CSC309F – Fall 2007

Outline
- Persistence via Database
- JDBC (Java Database Connectivity)
- JDBC API

Persistence via Database

Java Persistence
- JDBC (Java Database Connectivity)
- Object relational mapping
- Java Data Object (JDO)
- Enterprise JavaBean (EJB)
- Relational Database Management System (RDBMS)
- Object-oriented Database Management System (OODBMS)

Three Tier Architecture

- Database
  - A way of saving and accessing structured data on persistent (disk) storage

Database Advantages
- Data Safety
  - data is immune to program crashes
- Concurrent Access
  - atomic updates via transactions
- Fault Tolerance
  - replicated DBs for instant fail-over on machine/disk crashes
- Data Integrity
  - aids to keep data meaningful
- Scalability
  - can handle small/large quantities of data in a uniform manner
- Reporting
  - easy to write SQL programs to generate arbitrary reports
Relational Database

- First published by Edgar F. Codd in 1970
- A relational database consists of a collection of tables
- A table consists of rows and columns
- Each row represents a record
- Each column represents an attribute of the records contained in the table

Movie Database Example

<table>
<thead>
<tr>
<th>attendance</th>
<th>sdate</th>
<th>stime</th>
<th>available</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3/20/2005</td>
<td>20:00</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>3/20/2005</td>
<td>22:00</td>
<td>90</td>
</tr>
</tbody>
</table>

RDBMS Technology

- Client/Server Databases
  - Derby, Oracle, Sybase, MySQL, PointBase, SQLServer
- Embedded Databases
  - Derby, PointBase
- Personal Databases
  - Access

JDBC

(Java DataBase Connectivity)

JDBC Background

- Common SQL database access interface
- Allow Java programs to issue SQL statements and process the results
- Database independence
  - Can update underlying database with minimal code impact
- Represent database constructs as objects
  - Database connections, SQL statements, result sets, and database metadata

JDBC Architecture

Components of JDBC Architecture - 1
- Java application
  - In need to access database
  - Uses the API
- JDBC API
  - Provides DB independent abstraction to
    - Establish a connection with a database
    - Send SQL statements
    - Process the results

Components of JDBC Architecture - 2
- JDBC Driver
  - Translates API calls to requests made against the specific database
  - Specific driver required for the chosen database
  - Installed on the client. Usually a set of class files placed in the class path
  - All large databases are now supported

Components of JDBC Architecture - 3
- DBMS
  - The actual database engine
  - Derby, MySQL, Oracle, SQL Server, MS Access, PointBase, Postgresql

JDBC API
v3.0
v4.0
(May incur different class names in the Derby drivers)

API Highlights
- https://java.sun.com/javase/6/docs/api/
  - java.sql
    - DataSource
      - getConnection()
  - java.util
- Java - Read the full method signatures from this API carefully before using.
- Connection
  - commit(), rollback(), setAutoCommit()
  - Two statements
  - statement(), prepareStatement()
  - For SQL statements
- Statement
  - executeQuery(), executeUpdate()
- PreparedStatement
  - Parameter index starts from 1
  - Parameter index starts from 1
- ResultSet
  - next()
  - getString(), getInt()

Establishing a Connection
- javax.sql
  - DataSource
    - public Connection getConnection()
    - Attempts to establish a connection with the data source that this DataSource object represents
    - public Connection getConnection(String username, String password)
    - Attempts to establish a connection with the data source that this DataSource object represents using the given username and password
Derby Example 1: Connection

```java
import org.apache.derby.jdbc.EmbeddedDataSource;
import javax.sql.DataSource;
import java.sql.*;

// Driver code
EmbeddedDataSource eds = new EmbeddedDataSource();
eds.setDatabaseName(dbname);
eds.setCreateDatabase("create");
...
// JDBC code
Connection con = eds.getConnection();
```

Executing a Query

- `java.sql`
- `PreparedStatement` - Precompiled SQL statement; more efficient for multiple executions
  - `executeQuery()`, `executeUpdate()`, `setInt()`, `setString()`
  - Parameter index starts from 1
- `Statement` - `executeQuery()`, `executeUpdate()`
- `ResultSet` - `next()`
  - Accessing next row
  - `getString()`, `getInt()`
  - Retrieving attribute values

Example: PreparedStatement

```java
String insertStmt=""INSERT INTO ACCOUNT 
(NAME, AMOUNT) VALUES (?, ?);";
PreparedStatement ps = con.prepareStatement(insertStmt);

// Fill in the first and second args
ps.setString(1, "Charlie Smith");
ps.setDouble(2, 23.45);
int rowsAffected = ps.executeUpdate();

// Replace the first and second args
ps.setString(1, "Arnold Jones");
ps.setDouble(2, 102.23);
rowsAffected = ps.executeUpdate();
```

Example: Executing Query

```java
Statement stmt = con.createStatement();
// Send the query to the DB, get back a ResultSet
ResultSet rs = stmt.executeQuery("SELECT * FROM PART;");
// Go through all rows returned by the query
while(rs.next()){
  // Pull out individual columns from the current row
  int pno = rs.getInt("PNO");
  String pname = rs.getString("PNAME");
  // Print out the values
  System.out.println(pno + "\t" + pname);
}
rs.close();
```

Executing Update

```java
int rowsAffected = 
  stmt.executeUpdate(  
    "DELETE * FROM ACCOUNTS;" );
```
- Executes SQL INSERT, UPDATE, or DELETE statements
- Returns the number of rows affected

Announcements
**Midterm**
- Topics covered
  - Lectures 1-9 inclusive
  - From XHTML to Java Servlets and JSP
  - Use lecture notes as a guideline
- You can bring only 1 page cheat-sheet
  - Letter size
  - Double sided
- Format similar to previous years
  - Short conceptual questions ~40%
  - Programming questions ~60%

**Important Announcement (Repeat)**
- Tuesday Oct 16, 2007
  - 2-hr tutorial given in a CDF lab, BA3185
- Thursday Oct 18, 2007
  - Midterm, BA2195
- Starting Tuesday Oct 23, 2007
  - The official lecture room will be BA2185 on each Tuesday until the end of term
  - No change to Thursdays' classes (BA1240)