*Advanced Java Programming

Java Database Connectivity

Eriq Muhammad Adams J
eriq.adams@ub.ac.id | http://eriq.lecture.ub.ac.id

Agenda

* Connecting into Database
* Executing SQL Statemens
* Result Sets
* Batching SQL Statements & Transaction
* Prepared Statements
* Demo

Connecting into Database

* Get appropriate JDBC driver (depends on database vendor)
* Load JDBC driver

```
Class.forName("com.mckoi.JDBCDriver");
```

* Get JDBC URL String

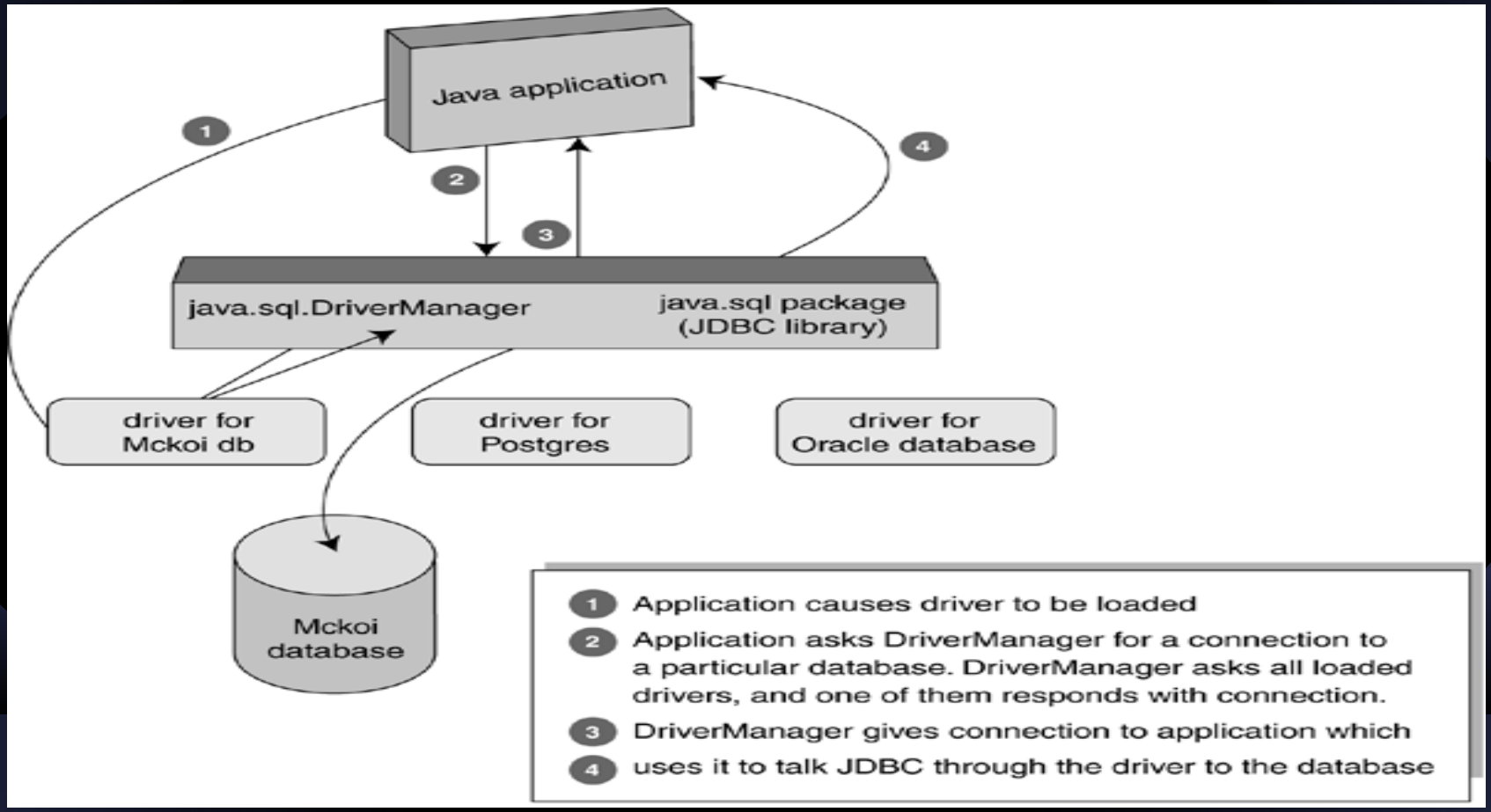
```
String url = "jdbc:mckoi:local://ExampleDB.conf?create=true";
```

* Get JDBC connection instance

```
connection = java.sql.DriverManager.getConnection(url, user,
passwd);
```

Connecting into Database (cont.)

How JDBC establishes a connection between your code and a database



Executing SQL Statements

* java.sql.Connection ‘s Methods

Method	Purpose
`Statement createStatement()`	Returns a statement object that is used to send SQL to the database.
`PreparedStatement prepareStatement (String sql)`	Returns an object that can be used for sending parameterized SQL statements.
`CallableStatement prepareCall (String sql)`	Returns an object that can be used for calling stored procedures.
`DataBaseMetaData getMetaData()`	Gets an object that supplies database configuration information.
`boolean isClosed()`	Reports whether the database is currently open or not.
`void setReadonly(boolean yn)`	Restores/removes read-only mode, allowing certain database optimizations.
`void commit()`	Makes all changes permanent since the previous commit/rollback.
`void rollback()`	Undoes and discards all changes done since the previous commit/rollback.
`void setAutoCommit(boolean yn)`	Restores/removes auto-commit mode, which does an automatic commit after each statement.
`void close()`	Closes the connection and releases the JDBC resources for it.

Executing SQL Statements (cont.)

* Call createStatement() first before executing SQL statements

```
Statement myStmt = connection.createStatement();

ResultSet myResult= myStmt.executeQuery( "SELECT *      FROM
Person;" );
```

java.sql.Statement's Methods

SQL statement	JDBC statement to use	Type of its return value	Comment
SELECT	`executeQuery(String sql)`	`ResultSet`	The return value will hold the data extracted from the database.
INSERT, UPDATE, DELETE, CREATE, DROP	`executeUpdate(String sql)`	`int`	The return value will give the count of the number of rows changed (for insert, update, or delete statements), or zero otherwise.
Stored procedure with multiple results	`execute(String sql)`	`boolean`	The return value is true if the first result is a ResultSet, false otherwise. You get the actual results by calling another method of the statement class.

Result Sets

* ResultSet to get query results

```
ResultSet  result = statement.executeQuery( " SELECT
Person.name, Person.age " + "FROM Person " + "WHERE
Person.age = 24 " );
while (result.next()) {
    String p = result.getString(1);
    int a = result.getInt(2);
    System.out.println( p + " is " + a + " years");
    }
```

Batching SQL Statements & Transaction

* To reduce the overhead of network latency, many vendors support a way to batch several SQL statements together and send them to the database as a group. In JDBC you can use JDBC Batch and transaction for data integrity.

```
Statement myStmt = conn.createStatement();
myStmt.addBatch( myNonSelectSQL0 );
myStmt.addBatch( myNonSelectSQL1 );
myStmt.addBatch( myNonSelectSQL2 );
try{
        connection.setAutoCommit(false);
        int [] res = myStmt.executeBatch();
        connection.commit(); // commit the changes
}catch(SQLException sqle){
        connection.rollback();
}
```

Prepared Statements

* Another way to boost performance is to precompile the SQL statement using what is termed a "prepared statement."

```
PreparedStatement pstmt = conn.prepareStatement( "UPDATE EMPLOYEES SET SALARY = ? WHERE ID = ?");
pstmt.setBigDecimal(1, 150000.00);
pstmt.setInt(2, linden4303);
pstmt.executeUpdate(); // other code goes here
pstmt.setBigDecimal(1, 85000.00);
pstmt.setInt(2, jenkins2705);
pstmt.executeUpdate();
```

Demo

* Available in JDBCDemo.zip

References

* Just Java 6th edition, Peter Van Der Linden, Addison Wesley