JDBC – Types of Statements

INTRODUCTION

JDBC – Types of Statements

LEARNING OBJECTIVES

At the end of this lesson, you will be able to:

- List the types of Statements in JDBC
- Understand PreparedStatement and it's usage
- Methods of Connection interface used to create Statements
- Methods of Statement interface used to execute queries
- Understand CallableStatement and it's usage
- Batch Processing

Refer package **com.mgait.jdbc** in the provided code base for demo programs on the topics covered in this presentation

The demo programs use the ‘hr’ schema of Oracle Express Edition

Types of Statement Objects

- JDBC Api provides three types of Statement interfaces for executing queries
  - Statement
    - Object of Statement type is used to execute a static SQL query
    - SQL Query is passed to the database when the Statement is executed
  - PreparedStatement
    - Object of PreparedStatement type is used to execute a parameterized SQL query
    - SQL query is passed to Database for pre-compilation before the Statement is executed
  - CallableStatement
    - Object of CallableStatement type is used to execute a Stored procedures

PreparedStatement and CallableStatement are sub interfaces of Statement interface

Creating Statement

- createStatement() method of Connection Interface is used to create a Statement object

```
Statement statement = conn.createStatement();
//conn is a Connection object
```

- Statement object is
  - Used to execute static SQL queries
  - Used to execute DML and DDL statements
- Only one ResultSet object per Statement object can be open at the same time
- Closing a Statement object closes the Resultset, it produced

Executing Statements

- Statement Interface defines methods for execution of SQL statements.
  - executeQuery(String sql) : ResultSet
  - executeUpdate(String sql) : int
  - execute(String sql) : boolean

- **executeQuery(String sql) : ResultSet**
  - Used to execute a sql which returns a single ResultSet Object
  - Used to execute SELECT queries

```
String sql = "SELECT * FROM countries";
ResultSet result = statement.executeQuery(sql);
```

Executing Statements

- **executeUpdate(String sql) : int**
  - Used to execute sql statement like INSERT,UPDATE, DELETE or DDL's
  - Returns an int, specifying the number of rows affected

```
String sql = "INSERT INTO REGIONS VALUES(6,'ASIA') ";
int insertCount = stmt.executeUpdate(sql);
```

- **execute(String sql) : boolean**
  - Used to execute SQL statement/s, which may return multiple results
  - Can be used to dynamically execute an unknown SQL String
  - A return value of 'true' indicates that the first result is a ResultSet

Updating and Deleting using Statement

UPDATING

```
Statement statement = conn.createStatement();
//conn is a Connection object
String sql = "UPDATE EMPLOYEES SET SALARY = SALARY * 1.1 ";
int updateCount = stmt.executeUpdate(sql);
```

DELETING

```
Statement statement = conn.createStatement();
//conn is a Connection object
String sql = "DELETE FROM EMPLOYEES WHERE STATUS = 'RESIGN' ";
int deleteCount = stmt.executeUpdate(sql);
```

Creating PreparedStatement

- prepareStatement() method of Connection Interface is used to create a PreparedStatement

```
String sql = "UPDATE EMPLOYEES SET ROLE = ? WHERE ID = ?";
PreparedStatement pstmt = conn.prepareStatement(sql);
```

Query supplied during statement creation

- PreparedStatement
  - Gets the query precompiled before execution
  - Provides the ability to insert parameters using ? (placeholder for parameters)
  - Provides better performance , If a query is executed multiple times with new parameter values

Executing PreparedStatement

- Before executing PreparedStatement, values needs to be provided to parameter's defined by **?**
- setXXX(index, value) methods of PreparedStatement are used to provide the parameter values

```
String sql = "UPDATE EMPLOYEES SET ROLE = ? WHERE ID = ?";
PreparedStatement pstmt = conn.prepareStatement(sql);
```

```
pstmt.setString(1, "admin");    // 1 is parameter index, admin is the value
pstmt.setInt(2, 101);           // 2 is parameter index, 101 is the value
```

- Compatible values for the parameters in the above SQL are provided as below

No sql supplied during execution

```
int updateCount = pstmt.executeUpdate();
```

PreparedStatement Examples

Demo
Class :

INSERT using PreparedStatement

```
String sql = "INSERT INTO COUNTRIES(COUNTRY_ID,COUNTRY_NAME,REGION_ID) VALUES(?,?,?);
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setString(1, "WI");
pstmt.setString(2, "West Indies");
pstmt.setInt(3,2);
int insertCount = pstmt.executeUpdate();
```

Select using PreparedStatement

```
String sql = "select COUNTRY_ID,COUNTRY_NAME from countries where REGION_ID = ?;
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setInt(1, 4);
ResultSet result = pstmt.executeQuery();
```

CallableStatement

- CallableStatement is used to execute SQL Stored procedures
- Stored procedure
  - Is a routine, that is stored and executed on the database server
  - Provides performance benefits for database heavy functionalites
  - Can take multiple input parameters and output parameter

- Consider a Stored procedure CALC_AVG_SALARY which takes first parameter as input and second parameter as output
  - Department id should be passed in the Input parameter
  - Procedure calculates the average salary for the department and store it in the output parameter

  - Next Slide explains how to call the procedure using CallableStatement

Executing CallableStatement

Demo
Class :

- prepareCall() method of Connection Interface is used to create a CallableStatement

```
CallableStatement cStmt = conn.prepareCall("{call SHOW_EMPLOYEE(?,?)}");
```

- Before executing the procedure, values have to be set to the input parameter

```
cStmt.setInt(1,90);
```

- setXXX(index, value) methods of CallableStatement are used to provide the parameter values

```
cStmt.registerOutParameter(2, Types.DOUBLE);
```

- The output parameters are to be registered using the registerOutParameter(index,type)

```
cStmt.executeQuery();
System.out.println(cStmt.getdouble(2));
```

Batch Processing

- Batch processing
  - Allows to group related SQL statements into a batch and submit them with one call to the database.
  - Reduces the amount of communication overhead, thereby improving performance.

- Methods for batch updates:
  - addBatch() **-** method of *Statement, PreparedStatement* and *CallableStatement* is used to add individual statements to the batch.

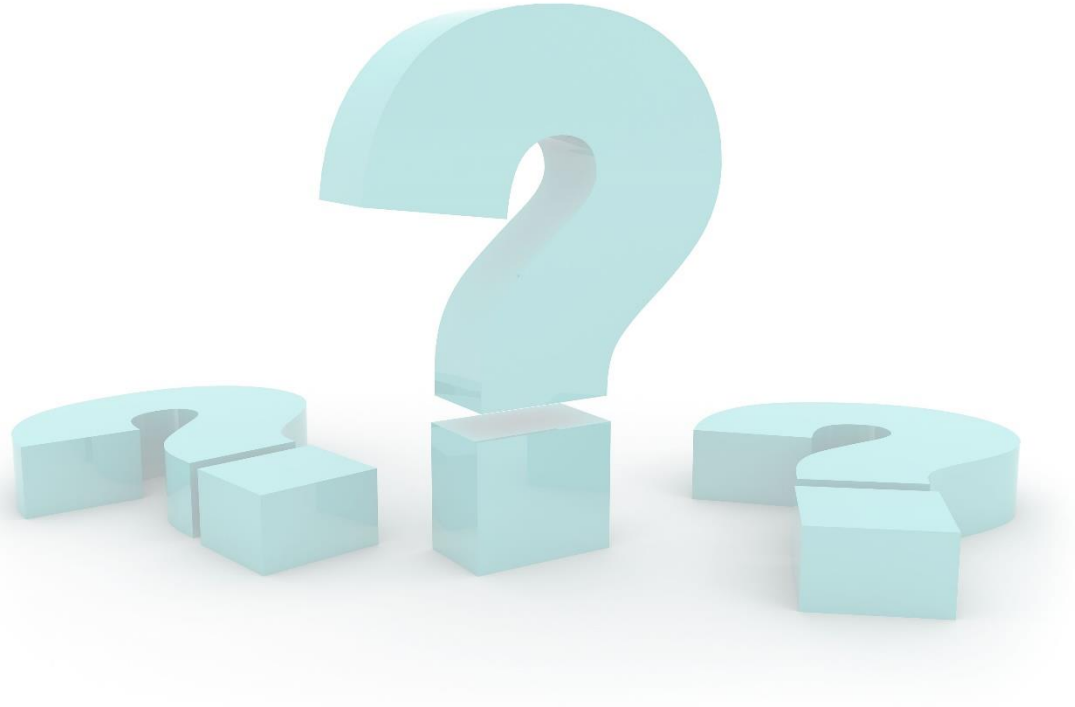
  - executeBatch()
    - Used to start the execution of all the statements grouped together
    - **R**eturns an array of integers, and each element of the array represents the update count for the respective statement.

Demo
Class : JdbcBatch

```
try(Connection conn = DriverManager.getConnection(DB_URL, USER, PASS)) {
    // Prepare SQL for Database Interaction
    String sql1 = "INSERT INTO REGIONS values(10,'Africa')";
    String sql2 = "INSERT INTO COUNTRIES values('KN','Kenya',10)";
    String sql3 = "UPDATE REGIONS SET REGION_NAME = 'South Africa'"
            + "where REGION_ID = 10";

    Statement stmt = conn.createStatement();
    stmt.addBatch(sql1);
    stmt.addBatch(sql2);
    stmt.addBatch(sql3);
    //Execute All Queries in batch
    int[] updateCount = stmt.executeBatch();
    stmt.close();

    System.out.println("sql1 affected Rows :" + updateCount[0]);
    System.out.println("sql2 affected Rows :" + updateCount[1]);
    System.out.println("sql3 affected Rows :" + updateCount[2]);

} catch (SQLException se) {
    System.out.println(se);
}
```

What is wrong in the following code?

```
CallableStatement cStmt = conn.prepareStatement("{call Demo(?)}");
```



References

- Refer following demo videos on EduNxt
  - Preparedstatement
  - Inserting A Record Using Preparedstatement
  - Updating And Deleting Records
  - Getting A Record Based On Primary Key
  - Getting Multiple Records
  - Executing Queries Dynamically
  - Callablestatement

SUMMARY

JDBC - Types of Statement's

SUMMARY

In this lesson, you've learned to:

- List the types of Statement's in JDBC
- Understand PreparedStatement and it's usage
- Methods of Connection interface used to create Statement's
- Methods of Statement interface used to execute queries
- Understand CallableStatement and it's usage
- Implement Batch Processing