JDBC PART-3

PreparedStatement interface:

SQL Statement is placed into a PreparedStatement and it is precompiled so that it can be executed efficiently multiple times.

Example1:

```
import java.sql.*;

class InsertDemo

{

	public static void main(String args[])

	{

		try{

		Class.forName("oracle.jdbc.driver.OracleDriver");

		Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","syste
m","manager");

		PreparedStatement pstmt=con.prepareStatement("insert into
student values(?,?,?)");

		pstmt.setInt(1, Integer.parseInt(args[0]));

		pstmt.setString(2, args[1]);

		pstmt.setInt(3, Integer.parseInt(args[2]));

		pstmt.executeUpdate();

		System.out.println("One Record Inserted Successfully");

		}catch(Exception e)

		{	System.err.println(e);	}

	}	}
```

Example2:

```
import java.sql.*;

class SelectDemo

{

        public static void main(String args[])

        {

                try{

                Class.forName("oracle.jdbc.driver.OracleDriver");

                Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","syste
m","manager");

                PreparedStatement pstmt=con.prepareStatement("select marks
from student where rollno=?");

                pstmt.setInt(1, Integer.parseInt(args[0]));

                ResultSet rs=pstmt.executeQuery();

                rs.next();

                System.out.println(rs.getInt(1));

                }catch(Exception e)

                {       System.err.println(e);      }

        }

}
```

CallableStatement interface:

CallableStatement interface is used to call stored procedures from java program.

Stored Procedure is a group of statements that we compile in the database for some task.

Stored procedures are useful while dealing with multiple tables with complex tasks and rather than sending multiple queries to the database, we can send required data to the stored procedure and have the logic executed in the database server itself.

Example:

```
import java.sql.*;

class ProcedureDemo

{

        public static void main(String args[])

        {

                try{

                Class.forName("oracle.jdbc.driver.OracleDriver");

                Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","syste
m","manager");

                CallableStatement cstmt=con.prepareCall("{ call insertpro(?,?,?) }");

                cstmt.setInt(1, Integer.parseInt(args[0]));

                cstmt.setString(2, args[1]);

                cstmt.setInt(3, Integer.parseInt(args[2]));

                cstmt.execute();

                System.out.println("One Record Inserted Successfully");

                }catch(Exception e)

                {

                        System.err.println(e);

                }

        }       }
```

JDBC New Features:

1) ResultSet Enhancements

2) Batch Updates

3) Advanced Data Types

4) Rowsets

1) ResultSet Enhancements:

ResultSet is an object that encapsulates set of rows from database.

ResultSet is generated based on sql query.

Whenever ResultSet is generated then ResultSet pointer or cursor points to before first record.

ResultSet enhancements are divided into 2 categories:

1) ResultSet Types

2) Concurrency Types

1) ResultSet Types:

There are 3 types of ResultSet:

1) TYPE_FORWARD_ONLY

2) TYPE_SCROLL_SENSITIVE

3) TYPE_SCROLL_INSENSITIVE

1) TYPE_FORWARD_ONLY:

It is a default ResultSet type.

It allows only forward direction to iterate records in a ResultSet.

It does not support absolute & relative positions.

This type of ResultSet introduced in JDBC 1.0 version.

2) TYPE_SCROLL_SENSITIVE:

It allows both forward & backward directions to iterate records in a ResultSet.

It supports both absolute & relative positions.

This type of ResultSet introduced in JDBC 2.o version.

It will show changes made by others in a Distributed DataBase Management System(DDBMS).

3) TYPE_SCROLL_INSENSITIVE:

It allows both forward & backward directions to iterate records in a ResultSet.

It supports both absolute & relative positions.

This type of ResultSet introduced in JDBC 2.o version.

It will not show changes made by others in a Distributed DataBase Management System(DDBMS).

2) Concurrency Types:

There are two types of concurrency control on ResultSet:

1) CONCUR_READ_ONLY 2) CONCUR_UPDATABLE

1) CONCUR_READ_ONLY:

It allows only read operation concurrently.

2) CONCUR_UPDATABLE:

It allows all operations concurrently.

By using ResultSet enhancements feature we can do the following operations on ResultSet:

1) Moving a cursor in a scrollable ResultSet.

2) Updating records with methods.

3) Inserting a record with methods.

4) Deleting a record with methods.

1) Moving a cursor in a scrollable ResultSet:

public abstract boolean isBeforeFirst() throws SQLException;

public abstract boolean isAfterLast() throws SQLException;

public abstract boolean isFirst() throws SQLException;

public abstract boolean isLast() throws SQLException;

=>The above 4 methods are used to check the cursor position

public abstract void beforeFirst() throws SQLException;

public abstract void afterLast() throws SQLException;

public abstract boolean first() throws SQLException;

public abstract boolean last() throws SQLException;

=>The above 4 methods are used to move the cursor

public abstract int getRow() throws SQLException;

=>This method returns row number

public abstract boolean previous() throws SQLException;

=>It returns true and moves the cursor to previous record if the previous record is present, otherwise returns false.

public abstract boolean absolute(int) throws SQLException;

public abstract boolean relative(int) throws SQLException;

Examples:

1) rs.absolute(3); => It moves the cursor from starting to 3rd record in a forward direction

2) rs.absolute(-3); => It moves the cursor from ending to 3rd record in a backward Direction

3) rs.relative(3); => It moves the cursor from current position to 3rd record in a forward direction

4) rs.relative(-3); => It moves the cursor from current position to 3rd record in a backward direction

java.sql.Connection Methods:

```
public abstract Statement createStatement(int, int) throws SQLException;
public abstract PreparedStatement prepareStatement(String, int, int)
   throws SQLException;
public abstract CallableStatement prepareCall(String, int, int)
   throws SQLException;
```

=>In the above all methods first int is result set type & second int is concurrency type

Example:

```
import java.sql.*;
class MoveDemo
{
        public static void main(String args[])
        {
                try{
                Class.forName("oracle.jdbc.driver.OracleDriver");
                Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","syste
m","manager");
```

```
                    Statement
stmt=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
                    ResultSet rs=stmt.executeQuery("select * from student");
                    rs.absolute(5);
                    System.out.println(rs.getInt(1));
                    System.out.println(rs.getString(2));
                    System.out.println(rs.getInt(3));
                    }catch(Exception e)
                    {
                              System.err.println(e);
                    }
          }
}
```

2) Updating records with methods:

```
  public abstract void updateInt(int, int) throws SQLException;
  public abstract void updateString(int, String) throws SQLException;
```

=>The above methods are used to update the ResultSet.

```
  public abstract void updateRow() throws SQLException;
```

=>It is used to update the database.

Example:

```
import java.sql.*;
class UpdateDemo
{
 public static void main(String args[])
 {
  try{
  Class.forName("oracle.jdbc.driver.OracleDriver");
  Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"system", "manager");
  Statement stmt=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
  ResultSet rs=stmt.executeQuery("select rollno, name, marks from student");
  rs.absolute(7);
  rs.updateInt(3, 40);
  rs.updateRow();
  System.out.println("One Record Updated Successfully");
  }catch(Exception e)
  {
   System.err.println(e);
  }
 }
}
```

3) Inserting a record with methods:

public abstract void moveToInsertRow() throws SQLException;

=>It is used to move the cursor to insert a record.

public abstract void insertRow() throws SQLException;

=>It is used to insert a record in a database.

Example:

```
import java.sql.*;
class InsertDemo
{
 public static void main(String args[])
 {
  try{
  Class.forName("oracle.jdbc.driver.OracleDriver");
  Connection 
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", 
"system", "manager");
  Statement stmt=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, 
ResultSet.CONCUR_UPDATABLE);
  ResultSet rs=stmt.executeQuery("select rollno, name, marks from student");
  rs.moveToInsertRow();
  rs.updateInt(1, 11);
  rs.updateString(2, "kkk");
  rs.updateInt(3, 66);
  rs.insertRow();
  System.out.println("One Record Inserted Successfully");
  }catch(Exception e)
  {
   System.err.println(e);
  }
 }      }
```

4) Deleting a record with methods:

public abstract void deleteRow() throws SQLException;

=>It is used to delete a record in a database

Example:

```
import java.sql.*;

class DeleteDemo

{

 public static void main(String args[])

 {

  try{

  Class.forName("oracle.jdbc.driver.OracleDriver");

  Connection 
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", 
"system", "manager");

  Statement stmt=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, 
ResultSet.CONCUR_UPDATABLE);

  ResultSet rs=stmt.executeQuery("select rollno, name, marks from student");

  rs.absolute(6);

  rs.deleteRow();

  System.out.println("One Record Deleted Successfully");

  }catch(Exception e)

  {

   System.err.println(e);

  }

 }       }
```

2) Batch Updates:

This feature allows to execute more than one sql query at a time.

It is also called as batch processing.

java.sql. Statement

  public abstract void addBatch(String) throws SQLException;

=>It is used to add SQL query

  public abstract int[] executeBatch() throws SQLException;

=>It is used to execute all SQL queries

Example:

```
import java.sql.*;

class BatchDemo

{         public static void main(String args[])   {

  try{    Class.forName("oracle.jdbc.driver.OracleDriver");

  Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"system", "manager");

  Statement stmt=con.createStatement();

  stmt.addBatch("insert into student values(12, 'lll', 34)");

  stmt.addBatch("update student set marks=45 where rollno=5");

  stmt.addBatch("delete from student where rollno=4");

  stmt.executeBatch();

  }catch(Exception e)

  {        System.err.println(e);     }

 }       }
```