JEE :: Advanced JDBC

TalentSprint

Licensed To Skill

Version 1.0.4

Advanced JDBC

The content in this presentation is aimed at teaching learners to:

- Operations performed by database engine
- Programming interaction on Statement interface
- Flow of Statement interface working
- Problems with Statement interface

Advanced JDBC

The content in this presentation is aimed at teaching learners to:

- Understanding PreparedStatement
- Flow of PreparedStatement working
- Programming Interaction

Advanced JDBC

Operations Performed By Database Engine

- Most databases handles JDBC/Sql Query in four steps
  - Parse the incoming query.
  - Parse the incoming query.
  - Optimize the data path.
  - Execute the optimized query to acquire and return data
- In JDBC, Statement interface and it's sub interface known as PreparedStatement interface are used to query the database.

Advanced JDBC

Statement Interface Example

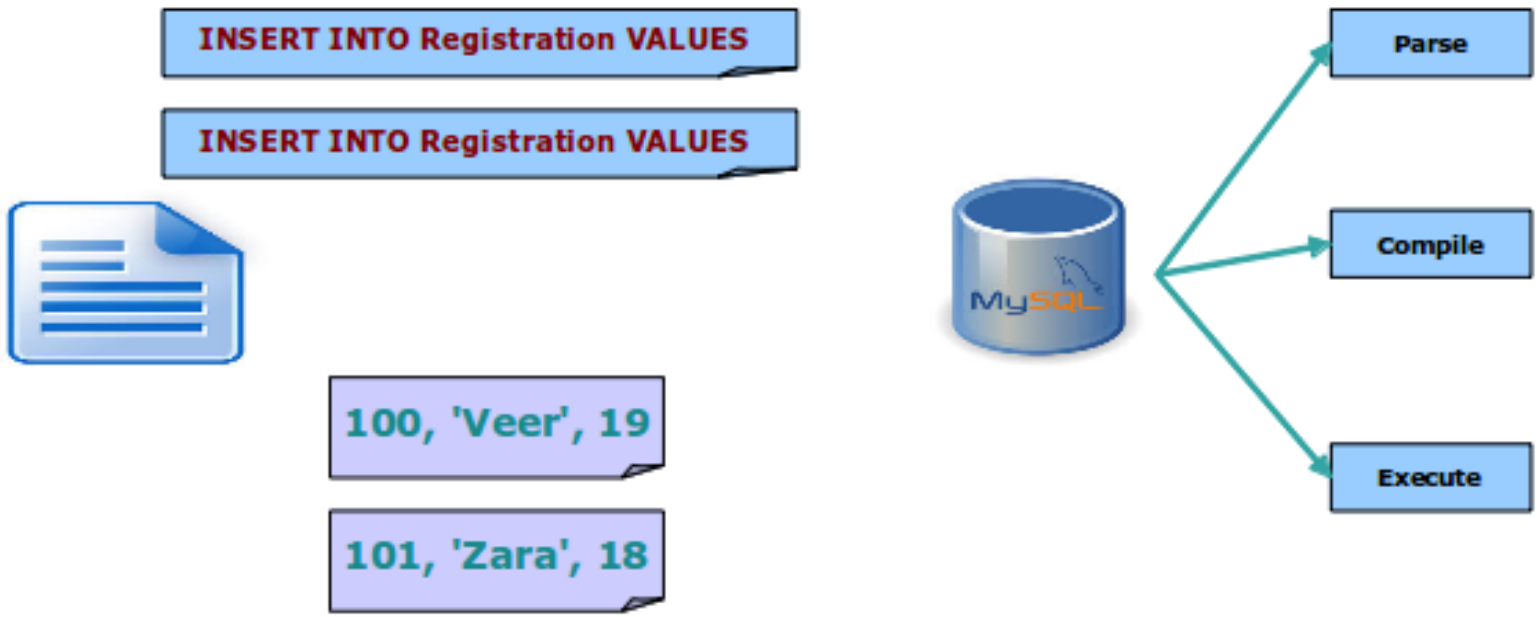
```
import java.sql.*;
public class JDBCExample {
    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        String sql=âĂİâĂİ;
        try{
            //STEP 2: Register JDBC driver
            Class.forName("com.mysql.jdbc.Driver");
            conn = DriverManager.getConnection(âĂIJjdbc:
    mysql://localhost:3306/mysqlâĂİ, USER, PASS);
            stmt = conn.createStatement();
```

Advanced JDBC

```
            sql = "INSERT INTO Registration VALUES (100, '
        Veer',19)";
            stmt.executeUpdate(sql);
            sql = "INSERT INTO Registration VALUES (101, '
        Zara', 18)";
            stmt.executeUpdate(sql);
            System.out.println("Inserted records into the table
        ...");
            }
            catch(SQLException se) {
                //Handle errors for JDBC
                se.printStackTrace();
            }
            catch(Exception e) {
                //Handle errors for Class.forName
                e.printStackTrace();
            }
        }
    }
```

Advanced JDBC

Working of Statement Interface Let us understand how Statement interface is working



It is clear now for every new statement database has to do parsing, compiling and executing operations always.

Advanced JDBC

Problems With Statement Interface

- Database parses the same query multiple times, executes and fetches the result.
- All four steps that is parsing, compiling, optimizing, fetching are always repeated.
- Network traffic to the Database is heavy since same query goes multiple times.
- To overcome this problem we have to use precompiled statements.

Advanced JDBC

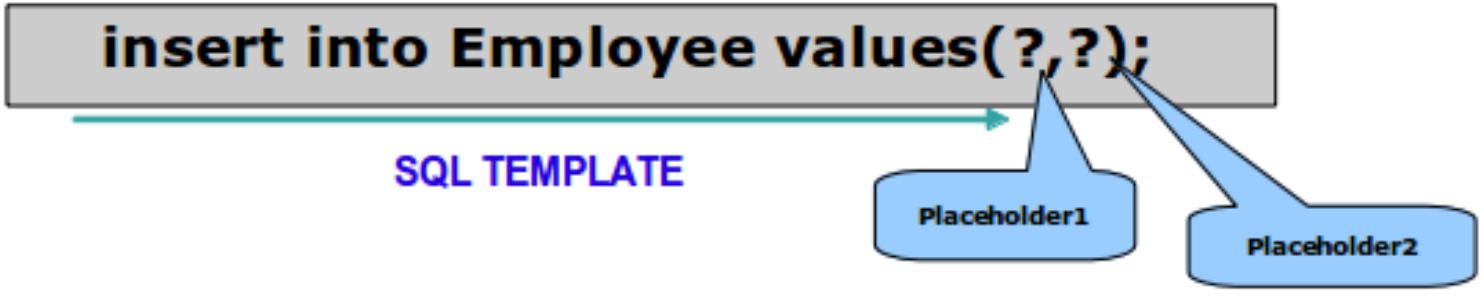
PreparedStatement

- It is inherited from Statement interface.
- In database management systems, a prepared statement or parameterized statement is a feature used to execute the same or similar database statements repeatedly with high efficiency
- It pre-executes parsing, compiling, optimizing. Thus, when creating PreparedStatement some optimization is done immediately.

Advanced JDBC

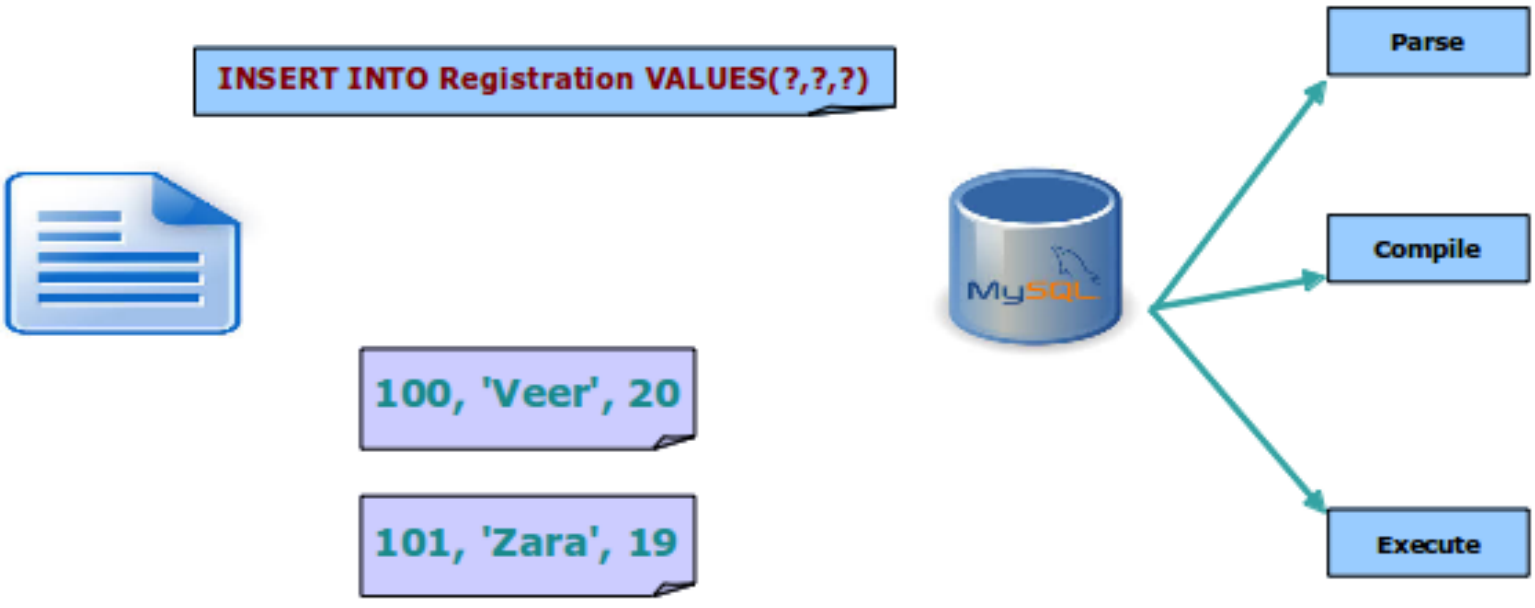
PreparedStatement

- The statement template is created by the application and sent to the database management system (DBMS). Certain values are left unspecified, called parameters, placeholders or bind variables (labelled “?” below):



Advanced JDBC

PreparedStatement Working



As we could see that PreparedStatement is parsed and compiled only once and execute many times using new values.

Advanced JDBC

PreparedStatement Interface

```
PreparedStatement ps = cn.PrepareStatement
   ("UPDATE emp SET eName= ? WHERE empno = ?");
```

First Parameter

Second Parameter

Supplying values for parameters

```
ps.setString(1,"Tom");
```

Value of the First Parameter

```
ps.setInt(2,101);
```

Value of the Second Parameter

Advanced JDBC

Rules to Remeber

Same ResultSet object should not be used again once it is terminated by while loop

```
ResultSet
rs=st.executeQuery("Select *
from emp");
while(rs.next())
{
System.out.println(rs.getInt("em
pno"));
}
```

```
ResultSet
rs1=st.executeQuery("Select *
from emp");
while(rs1.next())
{
System.out.println(rs.getInt("emp
no"));
}
```

The "rs" object is used again which will cause "operation not allowed again"

Advanced JDBC

Rules to Remeber

But using same ResultSet reference we can derive many results at different places in the program

```
ResultSet rs=st.executeQuery("Select *
from emp");
while(rs.next())
{
System.out.println(rs.getInt("empno"));
}
```

```
rs=st.executeQuery("Select * from emp");
while(rs.next())
{
System.out.println(rs.getInt("empno"));
}
```

This time "rs" will retrieve result

Advanced JDBC