**Guided LAB 305.2.1 - JDBC - PreparedStatement with DML and DDL**

**Lab Objective:**

## In this example, we will demonstrate the prepared statement with DML and DQL and its usage in Java using examples.

By the end of this lab, learners will be able to use prepared statements in Java for DML and DQL.

## **An Introduction to PreparedStatement in Java**

* **PreparedStatement** is a class in the ***java.sql*** package, and allows Java programmers to execute SQL queries by using the JDBC package.
* You can get a **PreparedStatement** object by calling the **connection.prepareStatement()** method.
* Prepared Statement queries are **pre-compiled on the database** and their access plans will be reused to execute further queries, which allows them to be executed much quicker than normal queries generated by the **Statement** object.
* The important methods of the PreparedStatement interface include:
* public void setInt(int paramIndex, int value) - sets the integer value to the given parameter index.
* public void setString(int paramIndex, String value) - sets the String value to the given parameter index.
* public void setFloat(int paramIndex, float value) - sets the float value to the given parameter index.
* public void setDouble(int paramIndex, double value) - sets the double value to the given parameter index.
* public int executeUpdate() - executes the query. It is used to create, drop, insert, update, delete etc.
* public ResultSet executeQuery() - executes the select query. It returns an instance of ResultSet.

## **Example: Prepared Statement for Insert Statements and Select Statement**

In this example, we will demonstrate the **Prepared Statement** for **Insert statements** and how to insert parameters(?) into SQL statements. We will insert one record, and then we will pull that record by using the Select statement for display purposes.

Create a class named **Insert\_preparedSt\_Example**, and write the below code:

| import java.sql.\*; public class Insert\_preparedSt\_Example {  public static void main(String[] args) {  Connection con = null;  PreparedStatement prepStmt = null;  ResultSet rs = null;  String dburl = "jdbc:mysql://localhost:3306/classicmodels";  String user= "root";  String password = "password";  try {  con = DriverManager.getConnection(dburl, user, password);  System.out.println("Connection established successfully!");  */\* ------ Lets insert one record using a prepared statement ------\*/*  String sqlQuery = "INSERT INTO EMPLOYEES (officeCode,firstName,lastName,email,extension,reportsTo,VacationHours,employeeNumber,jobTitle) VALUES (?,?,?,?,?,?,?,?,?)";  prepStmt = con.prepareStatement(sqlQuery);  prepStmt.setInt(1, 6);  prepStmt.setString(2, "Jamil");  prepStmt.setString(3, "fink");  prepStmt.setString(4, "JJ@gmail.com");  prepStmt.setString(5, "2759");  prepStmt.setInt(6, 1143);  prepStmt.setInt(7, 9);  prepStmt.setInt(8, 0003);  prepStmt.setString(9, "Manager");  int affectedRows = prepStmt.executeUpdate();  System.out.println(affectedRows + " row(s) affected !!");  */\* ------ Lets pull data from the database for an inserted record ------\*/*  *// Query which needs parameters* prepStmt = con.prepareStatement ("select \* from employees where employeeNumber = ? ");  prepStmt.setInt(1, 0003);  // execute select query  rs = prepStmt.executeQuery();  *// Display function to show the Resultset*  while (rs.next()) {  System.out.println(rs.getString("firstName"));  System.out.println(rs.getString("lastname"));  System.out.println(rs.getString("email"));  System.out.println(rs.getString("officeCode"));  }  }  catch (SQLException e)  {  e.printStackTrace();  }  try {  prepStmt.close();  con.close();  } catch (Exception e) {  e.printStackTrace();  }  }  } |
| --- |

**Output:**

Connection established successfully!

1 row(s) affected!!

Jamil

fink

JJ@gmail.com

2759

1143

9

0003

Manager

## **Example: Prepared Statement for Update Statements and Select Statements**

In this example, we will demonstrate the **prepared statement** for **update statements**. We will update one record, and then we will pull updated data from the database and display it on the console.

Create a class named **Update\_preparedSt\_Example**. Write the below code:

| import java.sql.Connection; import java.sql.PreparedStatement; import java.sql.ResultSet; import java.sql.SQLException; import java.sql.\*; public class Update\_preparedSt\_Example{  public static void main(String[] args) {  Connection con = null;  PreparedStatement prepStmt = null;  ResultSet rs = null;  String dburl = "jdbc:mysql://localhost:3306/classicmodels";  String user= "root";  String password = "password";  try {  con = DriverManager.getConnection(dburl, user, password);  System.out.println("Connection established successfully!");  String sql = "update employees set firstName=? , lastName=? where employeeNumber = ?";   prepStmt = con.prepareStatement(sql);  prepStmt.setString(1, "Gary");  prepStmt.setString(2, "Larson");  prepStmt.setLong (3, 0003);   int rowsAffected = prepStmt.executeUpdate();   prepStmt = con.prepareStatement("select \* from employees where employeeNumber=?");  prepStmt.setInt(1, 1401);  // execute select query  rs = prepStmt.executeQuery();  while (rs.next()) {  // System.out.print(rs.getInt("lastName"));  System.out.println(rs.getString("firstName"));  System.out.println(rs.getString("lastname"));  System.out.println(rs.getString("email"));  System.out.println(rs.getString("officeCode"));  }  }  catch (SQLException e)  {  e.printStackTrace();  }  } } |
| --- |

**Output:**

Connection established successfully!

Gary

Larson

pcastillo@classicmodelcars.com

4

**CANVAS STAFF USE ONLY: Canvas Submission Guideline:**

| **Instructions for Canvas Assignment Creation** |
| --- |
| **Assignment Name: GLAB - 305.2.1 - JDBC - PreparedStatement with DML and DDL**  **Points:** **100**  **Assignment Group: Module 305 - JDBC, ORM, and Hibernate - (Not Graded)**  **Display Grade As: Not-graded**  **Do not count this assignment towards the final grade: Checked**  **Submission Types:Files Uploads**  **Allowed Attempts: Unlimited**  **Everything else is the default.** |

## 