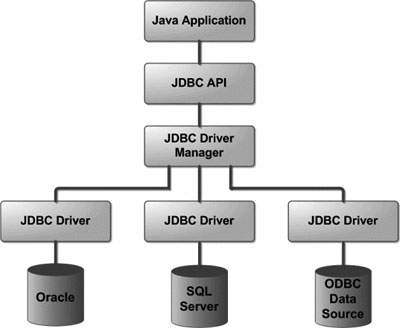
**JDBC**

## *What is JDBC?*

Java database connectivity is an API that provides classes and interfaces to connect or communicate the java applications with database. It enables java programs to execute SQL statements.

## *JDBC Architecture*



## *Connecting to MySQL database using JDBC*

1. Java driver for MySQL
2. Download from <http://dev.mysql.com/downloads>
3. Add the jar to build path

## *Program for testing MYSQL Connection*

**import** java.sql.\*;

**public** **class** TestMySQLConnection {

**static** String *username* = "root";

**static** String *password* = "admin123";

**static** String *dbURL* = "jdbc:mysql://localhost:3306/bank";

**public** **static** **void** main(String[] args) **throws** SQLException{

**try**(Connection conn=DriverManager.*getConnection*(*dbURL*,*username*,*password*)) {

System.*out*.println("Connection established successfully!!");

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

System.*err*.println(e.getMessage());

}

}

}

Output

**Connection established successfully!!**

## *Statement, ResultSet, PreparedStatement and CallableStatement Interfaces*

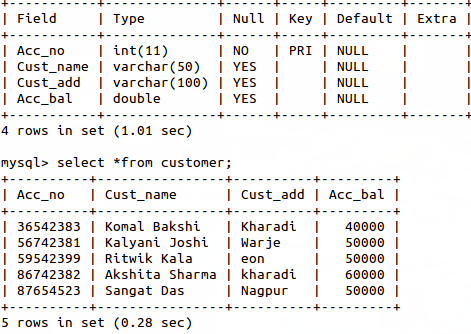
|  |  |
| --- | --- |
| **Statement** | General purpose access to your database. Useful when you are using static Sql statements at runtime. |
| **PreparedStatement** | Used to execute a parameterized query. Useful when we want to execute the same query many times |
| **CallableStatement** | Useful when you want to access the database stored procedures. |

## *Methods of the Statement interface*

* public ResultSet executeQuery(String sql)
* public int executeUpdate(String sql)
* public boolean execute(String sql)

## *Program to demonstrate methods of Statement Interface*

Table used



**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**public** **class** Banking {

**static** String *username* = "root";

**static** String *password* = "admin123";

**static** String *dbURL* = "jdbc:mysql://localhost:3306/bank";

**public** **static** **void** main(String[] args) **throws** SQLException{

Connection conn=**null**;

Statement stmt=**null**;

ResultSet rs=**null**;

**try** {

conn=DriverManager.*getConnection*(*dbURL*, *username*, *password*);

stmt=conn.createStatement();

System.*out*.println("\*Displaying records from the customer database\*\n"); //using executeQuery()

String sql = "select \*from customer";

rs=stmt.executeQuery(sql);

System.*out*.println("Account no\t\tCustomer Name\t\tCustomer Address");

**while**(rs.next())

{

System.*out*.println(rs.getInt("Acc\_no")+"\t\t"+rs.getString("Cust\_name")+"\t\t"+rs.getString("Cust\_add"));

}

System.*out*.println("\n\*Updating a record in the database\*\n"); //using executeUpdate()

sql="UPDATE customer set Cust\_add='f-residencies' where Cust\_name='Akshita Sharma'";

**int** flag=stmt.executeUpdate(sql);

System.*out*.println("Record updated..."+flag+" rows affected");

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

System.*err*.println(e.getMessage());

}

**finally**

{

**if**(rs!=**null**)

{

rs.close();

}

**if**(stmt!=**null**)

{

stmt.close();

}

**if**(conn!=**null**)

{

conn.close();

}

}

}

}

Output

\*Displaying records from the customer database\*

Account no Customer Name Customer Address

36542383 Komal Bakshi Kharadi

56742381 Kalyani Joshi Warje

59542399 Ritwik Kala Kharadi

86742382 Akshita Sharma kharadi

\*Updating a record in the database\*

Record updated...1 rows affected

## *ResultSet Interface*

It represents the result set of a database query. It maintains a cursor that points to the current row of the result set.

## *Methods of the ResultSet interface*

* **Navigational methods :**

Examples: beforeFirst(),afterLast(),etc

* **Get methods:**

Examples:getInt(int columnindex),getInt(String columnname)

* **Update methods**:

Examples:updateString(int columnindex,String s)

## *ResultSet types*

* TYPE\_FORWARD\_ONLY
* TYPE\_SCROLL\_INSENSITIVE
* TYPE\_SCROLL\_SENSETIVE

## *ResultSet Concurrency types*

* CONCUR\_READ\_ONLY
* CONCUR\_UPDATABLE

## *Program to demonstrate Read Only ResultSet*

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**public** **class** Banking2 {

**static** String *username* = "root";

**static** String *password* = "admin123";

**static** String *dbURL* = "jdbc:mysql://localhost:3306/bank";

**public** **static** **void** main(String[] args) **throws** SQLException{

**try**(Connection conn=DriverManager.*getConnection*(*dbURL*, *username*, *password*);

Statement stmt=conn.createStatement(ResultSet.*TYPE\_SCROLL\_INSENSITIVE*,ResultSet.*CONCUR\_READ\_ONLY*);

ResultSet rs=stmt.executeQuery("select \*from customer");) {

System.*out*.println("Account no\t\tCustomer Name\t\tCustomer Address");

**while**(rs.next())

{

System.*out*.println(rs.getInt("Acc\_no")+"\t\t"+rs.getString("Cust\_name")+"\t\t"+rs.getString("Cust\_add"));

}

System.*out*.println("\*rs.absolute() method\*");

rs.absolute(2); //used for moving the cursor to a specified row

System.*out*.println(rs.getString(2)); //display the second column of the second row

System.*out*.println("\*moving the cursor to the previous row\*");

rs.previous();

System.*out*.println(rs.getString(2)); //display the second column of previous row

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

System.*err*.println(e.getMessage());

}

}

}

Output

Account no Customer Name Customer Address

36542383 Komal Bakshi Kharadi

56742381 Kalyani Joshi Warje

59542399 Ritwik Kala Kharadi

86742382 Akshita Sharma kharadi

\*rs.absolute() method\*

Kalyani Joshi

\*moving the cursor to the previous row\*

Komal Bakshi

## *Program to demonstrate Updatable ResultSet*

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**public** **class** Banking3 {

**static** String *username* = "root";

**static** String *password* = "admin123";

**static** String *dbURL* = "jdbc:mysql://localhost:3306/bank";

**public** **static** **void** main(String[] args) **throws** SQLException{

**try**(Connection conn=DriverManager.*getConnection*(*dbURL*, *username*, *password*);

Statement stmt=conn.createStatement(ResultSet.*TYPE\_SCROLL\_INSENSITIVE*,ResultSet.*CONCUR\_UPDATABLE*);

ResultSet rs=stmt.executeQuery("select \*from customer");) {

rs.absolute(3);

rs.updateString("Cust\_add", "eon");

rs.updateRow();

System.*out*.println("Record updated successfully");

//System.out.println(rs.getString("Cust\_add"));

rs.moveToInsertRow();

rs.updateInt("Acc\_no", 87654523);

rs.updateString("Cust\_name", "Sangat Das");

rs.updateString("Cust\_add", "Nagpur");

rs.insertRow();

System.*out*.println("Record inserted successfully");

//System.out.println(rs.getString(2));

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

System.*err*.println(e.getMessage());

}

}

}

Output

Record updated successfully

Record inserted successfully

## *Why should we use PreparedStatement?*

* Query is compiled only once
* Faster than the Statement Interface
* Performance of the application will be faster.

## *Program demonstrating PreparedStatement*

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.Scanner;

**public** **class** Banking4 {

**static** String *username* = "root";

**static** String *password* = "admin123";

**static** String *dbURL* = "jdbc:mysql://localhost:3306/bank";

**public** **static** **void** main(String[] args) **throws** SQLException{

Connection conn=**null**;

PreparedStatement pstmt=**null**;

ResultSet rs=**null**;

String add = **null**;

Scanner sc=**new** Scanner(System.*in*);

**try** {

conn=DriverManager.*getConnection*(*dbURL*, *username*, *password*);

String sql="select \*from customer where Cust\_Add=?";

pstmt=conn.prepareStatement(sql, ResultSet.*TYPE\_SCROLL\_INSENSITIVE*, ResultSet.*CONCUR\_READ\_ONLY*);

System.*out*.println("Enter the address for viewing filterd results:");

add=sc.nextLine();

*preparedStatement*(pstmt,add);

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

System.*err*.println(e.getMessage());

}

**finally**

{

sc.close();

**if**(rs!=**null**)

{

rs.close();

}

**if**(pstmt!=**null**)

{

pstmt.close();

}

**if**(conn!=**null**)

{

conn.close();

}

}

}

**private** **static** **void** preparedStatement(PreparedStatement pstmt,String add)

**throws** SQLException {

ResultSet rs;

pstmt.setString(1,add);

rs=pstmt.executeQuery();

System.*out*.println("Account no\t\tCustomer Name\t\tCustomer Address");

**while**(rs.next())

{

System.*out*.println(rs.getInt("Acc\_no")+"\t\t"+rs.getString("Cust\_name")+"\t\t"+rs.getString("Cust\_add"));

}

}

}

Output:

Enter the address for viewing filterd results:

Kharadi

Account no Customer Name Customer Address

36542383 Komal Bakshi Kharadi

86742382 Akshita Sharma kharadi

## *When do we use CallableStatement?*

We use them when we want to access the database stored procedures.

Using the CallableStatement you can pass three types of parameters to a stored procedure –

* **IN** – Used to pass values to stored procedures
* **OUT** – Used to hold results returned by stored procedures
* **INOUT** – Can act as both input and output parameters.

Before calling the stored procedure you must register the output parameter using the registerOutParameter method of the CallableStatement.

**What are stored procedures?**

Group of SQL statements that perform a particular task.

**Why stored procedures?**

* Less duplicated work
* Faster execution
* Better security

## *Program used to demonstrate CallableStatement*

\*Stored procedure used\*

mysql> DELIMITER /

mysql> CREATE PROCEDURE getCustAccNo(IN Cname varchar(50),OUT Accno int) BEGIN SELECT Acc\_no INTO Accno FROM customer WHERE Cust\_name=Cname; END/

**import** java.sql.CallableStatement;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.SQLException;

**public** **class** Banking5 {

**static** String *username* = "root";

**static** String *password* = "admin123";

**static** String *dbURL* = "jdbc:mysql://localhost:3306/bank";

**public** **static** **void** main(String[] args) **throws** SQLException{

String SQL = "{call getCustAccNo(?, ?)}";

**try**(Connection conn=DriverManager.*getConnection*(*dbURL*,*username*,*password*);CallableStatement cstmt = conn.prepareCall (SQL);) {

String cust\_name="Kalyani Joshi";

cstmt.setString(1, cust\_name);

cstmt.registerOutParameter(2, java.sql.Types.*INTEGER*);

cstmt.execute();

**int** acc\_no=cstmt.getInt(2);

System.*out*.println("The account no for "+cust\_name+" is "+acc\_no);

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

System.*err*.println(e.getMessage());

}

}

}

Output

The account no for Kalyani Joshi is 56742381

## *Managing Transactions*

A database transaction is a sequence of actions that are treated as a single unit of work. These actions should either complete entirely or take no effect at all. Transaction management is an important to ensure data integrity and consistency.

## *Program demonstrating transaction management*

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.Scanner;

**public** **class** Banking6 {

**static** String *username* = "root";

**static** String *password* = "admin123";

**static** String *dbURL* = "jdbc:mysql://localhost:3306/bank";

**public** **static** **void** main(String[] args) **throws** SQLException{

PreparedStatement pstmt=**null**;

Connection conn=**null**;

Scanner sc=**new** Scanner(System.*in*);

**try** {

conn=DriverManager.*getConnection*(*dbURL*,*username*,*password*);

conn.setAutoCommit(**false**);

System.*out*.println("Enter from account = ");

**int** acc1=sc.nextInt();

System.*out*.println("Enter to account = ");

**int** acc2=sc.nextInt();

System.*out*.println("Enter amount to be transfered");

**double** ammount=sc.nextInt();

String withdrawSQL="UPDATE customer SET Acc\_bal=Acc\_bal - ? WHERE Acc\_no = ?";

pstmt=conn.prepareStatement(withdrawSQL);

pstmt.setDouble(1, ammount);

pstmt.setInt(2, acc1);

pstmt.executeUpdate();

String depositSql = "UPDATE customer SET Acc\_bal=Acc\_bal + ? WHERE Acc\_no = ?";

pstmt=conn.prepareStatement(depositSql);

pstmt.setDouble(1, ammount);

pstmt.setInt(2, acc2);

pstmt.executeUpdate();

String sql = "SELECT Acc\_bal from customer where Acc\_no = ? ";

pstmt=conn.prepareStatement(sql);

pstmt.setInt(1,acc1);

ResultSet rs=pstmt.executeQuery();

**double** bal\_amount=0;

**while**(rs.next())

{

bal\_amount=rs.getDouble("Acc\_bal");

}

**if**(bal\_amount>=20000)

{

conn.commit();

System.*out*.println("Ammount transferred successfully!");

}

**else**

{

conn.rollback();

System.*out*.println("Insufficient funds : "+bal\_amount+"\nTransaction Rollbacked!!");

}

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

System.*err*.println(e.getMessage());

}

**finally**

{

sc.close();

**if**(pstmt!=**null**)

{

pstmt.close();

}

**if**(conn!=**null**)

{

conn.close();

}

}

}

}

Output

Case 1

Enter from account =

36542383

Enter to account =

86742382

Enter amount to be transfered

10000

Ammount transferred successfully!

Case 2

Enter from account =

36542383

Enter to account =

59542399

Enter amount to be transfered

30000

Insufficient funds : 10000.0Minimum account balance should be = 20000

Transaction Rollbacked!!

## *Improvements in Java 8*

* Removal of JDBC-ODBC Bridge
* Addition of REF\_CURSOR support
* Added support for updatable counts

Refer the following link *–*

*www.javatpoint.com/java-8-jdbc-improvements*