Assignment No .8

Title of Assignment: Database Connectivity:
Write a program to implement MySQL/Oracle database connectivity with any front endlanguage to implement Database navigation operations (add, delete, edit etc.)

Course Objective:
Implement PL/SQL Code block for given requirements

Course Outcome:

C306.6
Design and develop application considering actual requirements and using database concepts

Software Required: - Mysql/eclipse/ wampserver/php/oracle

Why use JDBC
Before JDBC, ODBC API was the database API to connect and execute query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).

JDBC Driver
1. JDBC Drivers

1. JDBC-ODBC bridge driver
2. Native-API driver
3. Network Protocol driver
4. Thin driver

JDBC Driver is a software component that enables java application to interact with the database.There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver
2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)

1) JDBC-ODBC bridge driver
The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBCmethod calls into the ODBC function calls. This is now discouraged because of thin driver.

Advantages:

- easy to use.
- can be easily connected to any database.

Disadvantages:

- Performance degraded because JDBC method call is converted into the ODBC function calls.
- The ODBC driver needs to be installed on the client machine.

2) Native-API driver
The Native API driver uses the client-side libraries of the database. The driver converts JDBC method calls into

native callsof the database API. It is not written entirely in java.

Advantage:

- performance upgraded than JDBC-ODBC bridge driver.Disadvantage:
- The Native driver needs to be installed on the each client machine.

- The Vendor client library needs to be installed on client machine.

3) Network Protocol driver

The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. It is fully written in java.

Advantage:

- No client side library is required because of application server that can perform many tasks like auditing, load balancing, logging etc.

Disadvantages:

- Network support is required on client machine.
- Requires database-specific coding to be done in the middle tier.
- Maintenance of Network Protocol driver becomes costly because it requires database-specific coding to be donein the middle tier.

4) Thin driver

The thin driver converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as thin driver. It is fully written in Java language.

Advantage:

- Better performance than all other drivers.
- No software is required at client side or server side.

Disadvantage:

- Drivers depends on the Database.

5 Steps to connect to the database in java They are as follows:

- Register the driver class
- Creating connection
- Creating statement
- Executing queries
- Closing connection

1) Register the driver class

The forName() method of Class class is used to register the driver class. This method is used to dynamically load the driver class.

Syntax of forName() method

1. public static void forName(String className)throws

ClassNotFoundExceptionExample to register the OracleDriver class

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

2) Create the connection object

The getConnection() method of DriverManager class is used to establish connection with the database.Syntax of getConnection() method

3) public static Connection getConnection(String url)throws SQLException

4) public static Connection getConnection(String url,String name,String password) throws

SQLExceptionExample to establish connection with the Oracle database

5) Connection con=DriverManager.getConnection(

6) "jdbc:oracle:thin:@localhost:1521:xe","system","password");

7) Create the Statement object

The createStatement() method of Connection interface is used to create statement. The object of statement is responsibleto execute queries with the database.

Syntax of createStatement() method

public Statement createStatement()throws

SQLExceptionExample to create the statement object

Statement stmt=con.createStatement();

8) Execute the query

The executeQuery() method of Statement interface is used to execute queries to the database. This method returns theobject of ResultSet that can be used to get all the records of a table.

Syntax of executeQuery() method

public ResultSet executeQuery(String sql)throws

SQLExceptionExample to execute query

```
ResultSet rs=stmt.executeQuery("select * from emp");
 while(rs.next()){  System.out.println(rs.getInt(1)+"
 "+rs.getString(2));
  }
```

9) Close the connection object

By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection. Syntax of close() method

public void close()throws

SQLExceptionExample to close

connection con.close();

```
import
java.sql.*;
class
OracleCon{
public static void main(String
args[]){try{
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection(
 "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");Statement stmt=con.createStatement();
  ResultSet rs=stmt.executeQuery("select * from emp");
```

```
while(rs.next())
System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
```

```
  con.close();
 }catch(Exception e){ System.out.println(e);}   }  }
```

PHP Connectivity

PHP will require that mysqli is enabled (it is on most PHP set ups).

```
$db =
mysqli_connect('localhost','username','password','database_name')or
die('Error connecting to MySQL server.');
```

The variable $db is created and assigned as the connection string, it will be used in future steps. If there is a failure then anerror message will be displayed on the page. If it is successful you will see PHP connect to MySQL.

Performing a database query

The mysql query is actually performed in the body of the html page, so additional php opening and closing tags will be required. For the query we are going to specify a read of all fields from a given table. The $query variable selects all rows in the table. You just need to use your table name.

```
<?php
$query = "SELECT * FROM table_name";
mysqli_query($db, $query) or die('Error querying database.');
```

Again the returned page in the browser should be blank and error free, if you do receive the error – 'Error queryingdatabase..' check the table name is correct.

```
<?php
//Step1
$query = "SELECT * FROM table_name";
mysqli_query($db, $query) or die('Error querying database.');
$result = mysqli_query($db, $query);
$row = mysqli_fetch_array($result);
```

```
  while ($row = mysqli_fetch_array($result)) {
echo $row['first_name'] . ' ' . $row['last_name'] . ': ' . $row['email'] . ' ' . $row['city'] .'<br />';}?>

  </body>
  </html>
```

Closing off the connection

Closing the connection will require another set off opening and closing php tags after the closing html tag.

```
  <?php
  //Step1

$db =

mysqli_connect('localhost','root','root','database_name')or

die('Error connecting to MySQL server.');

  ?>
  <html>
<head>

</head>

<body>

<h1>PHP connect to MySQL</h1>
  <?php
  //Step2

  $query = "SELECT * FROM table_name";

  mysqli_query($db, $query) or die('Error querying database.');
  //Step3
  $result = mysqli_query($db, $query);

  $row = mysqli_fetch_array($result);
  while ($row = mysqli_fetch_array($result)) {
echo $row['first_name'] . ' ' . $row['last_name'] . ': ' . $row['email'] . ' ' . $row['city'] .'<br />';}
  //Step 4

  mysqli_close(

  $db);

  ?>
  </body>

  </html>
```

Conclusion: We have implemented database connectivity

Activity to be Submitted by Students

1. Login page in PHP