4
UNIT

Enterprise Java Beans and JDBC

CONTENTS

Part-1 : Enterprise Java Bean : Preparing a Class to be a Java Beans, Creating a Java Beans, Java Beans Properties ..... **4–2D to 4–8D**

Part-2 : Types of Beans, Stateful Session Bean, Stateless Session Bean, Entity Bean ..... **4–8D to 4–10D**

Part-3 : Java Database Connectivity (JDBC) : Merging Data from Multiple Tables : Joining ..... **4–10D to 4–17D**

Part-4 : Manipulating, Databases with JDBC Prepared Statements, Transaction Processing, Stored Procedures ..... **4–17D to 4–23D**

PART-1

Enterprise Java Bean : Preparing a Class to be a Java Beans, Creating a Java Beans, Java Beans Properties.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.1. What is EJB ? Write the advantages and disadvantages of EJB.

Answer

1. An Enterprise Java Bean is a server-side component which encapsulates business logic.
2. EJB (Enterprise Java Bean) is used to develop scalable, robust and secured enterprise applications in Java.
3. Middleware services such as security, transaction management etc. are provided by EJB container to all EJB applications.
4. To run EJB application, we need an application server (EJB Container) such as Jboss, Glassfish, Weblogic, Websphere etc.
5. EJB application is deployed on the server, so it is also called server-side component.

Advantages of EJB :

1. It can run in multithreaded environment.
2. It contains only business logic.
3. EJB provides distributed transaction support.
4. It provides portable and scalable solutions of the problem.
5. It provides a mechanism to store and retrieve data in a persistent way.

Disadvantages of EJB :

1. It requires application server.
2. It requires only Java client. For other language client, we need to go for web service.
3. It is complex to understand and develop EJB applications.

Que 4.2. Discuss EJB. Explain EJB architecture. What are its

various types ? **AKTU 2019-20, Marks 07**

OR

Write short note on EJB architecture. ~~AKTU 2016-17, Marks 05~~

Answer

EJB : Refer Q. 4.1, Page 4–2D, Unit-4.

EJB architecture :

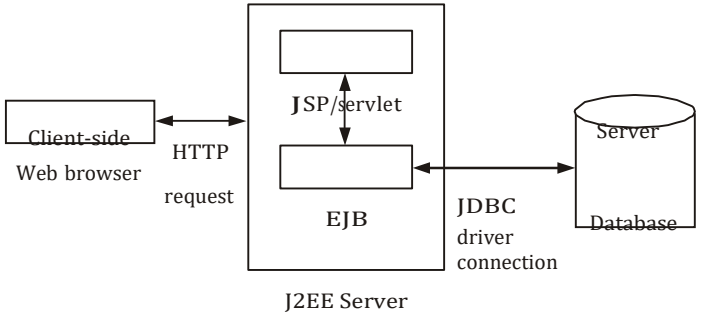

Fig. 4.2.1. Architecture.

The EJB architecture is an extension of web architecture.

Working of EJB architecture :

1. The client is working on a web browser.
2. There is a database server that hosts a database, like MySQL / Oracle.
3. The J2EE server machine is running on an application server.
4. The client interface is provided with JSP / Servlet.
5. The application server manages the relationships between the client and database.

Types of EJB :

1. **Entity bean :** Entity beans represent persistent data storage. Entity beans are used for modeling the business concept.
2. **Session bean :** Session beans are used for managing processes or tasks. Hence, session beans are used for managing activities.
3. **Message driven bean :** Message driven bean is similar to the session bean but it gets activated only when asynchronous message arrives. When a message arrives then the EJB container calls the message driven bean on message method to process the message.

Que 4.3. What is Java Bean exactly ? Write down the steps to create

Java Bean. What is the role of introspection in Java Bean ?
OR

Explain EJB architecture. What are its various types ? Describe the steps used to create Java Bean and to build application using BDK. **AKTU 2015-16, Marks 10**

Answer

EJB architecture and its types : Refer Q. 4.2, Page 4–2D, Unit-4.

Java Beans :

1. Java Beans are classes which encapsulate several objects into a single object.
2. It helps in accessing the objects from multiple places.
3. It is a portable, platform independent model written in Java.

Steps used to create Java Bean :

Step 1 : Put source code into a file named "SimpleBean.java" :

```
import java.awt. * ;
import java.io.Serializable;
public class SimpleBean extends Canvas
implements Serializable {
// Constructor sets inherited properties
 public SimpleBean ( ) {
setSize (60, 40);
setBackground (Color.red);
}
}
```

Step 2 : Compile the file :

```
javac SimpleBean.java
```

Step 3 : Create a manifest file, named "manifest.tmp" :

```
Name : SimpleBean.class
Java-Bean : True
```

Step 4 : Create the JAR file, named "SimpleBean.jar" :

```
jar cfm SimpleBean.jar manifest.tmp SimpleBean.class
```

Then, verify that the content is correct by the command "jar tf SimpleBean.jar".

Step 5 :

1. Start and run the Bean Box.
2. Load JAR file into Bean Box by selecting "Loadjar..." under the File menu.

Step 6 :

1. After the file selection dialog box is closed. Then "SimpleBean" appear at the bottom of the toolbox window.
2. Select SimpleBean.jar.
3. Cursor will change to a plus. In the middle BeanBox window, we can now click to drop in what will appear to be a coloured rectangle.

Step 7 : Try changing the red box colour with the Properties windows.

Step 8 : Choose "Events" under the "Edit" menu in the middle window to see what events SimpleBean can send. These events are inherited from java.awt.Canvas.

Role of introspection in Java Bean :

1. Introspection in Java is used in the context of Java Beans which defines the component model of Java.
2. Introspection feature enables a Java Bean to get the properties, methods and events of other beans at runtime.
3. This helps the developers to design and develop their beans without knowing the details of other beans.

Steps to build application using BDK :

Step 1 : Create a directory for the new bean.

Step 2 : Create the Java source file(s).

Step 3 : Compile the source file(s).

Step 4 : Create a manifest file.

Step 5 : Generate a JAR file.

Step 6 : Start the BDK.

Step 7 : Test the newly created Java Bean.

Que 4.4. Explain JavaBeans. Why they are used ? Discuss setter and getter method with Java code. AKTU 2019-20, Marks 07

Answer

Java Beans : Refer Q. 4.3, Page 4–3D, Unit-4.

Java Beans are used because :

1. It encapsulates many objects into a single object.
2. It allows us to use properties of getter and setter methods.
3. It has Java object which has constructor with no argument.
4. It can be manipulated visually in a builder tools.

Getter and setter method :

1. In Java, getter and setter are two conventional methods that are used for retrieving and updating value of a variable.

For example :

The following code is an example of simple class with a private variable and a couple of getter/setter methods :

```
public class SimpleGetterAndSetter {
private int number;
public int getNumber() {
return this.number;
}
public void setNumber(int num) {
this.number = num;
}
}
```

2. The class declares a private variable, number. Since "number" is private, code from outside this class cannot access the variable directly,

```
SimpleGetterAndSetter obj = new SimpleGetterAndSetter();
obj.number = 10;
int num = obj.number;
```

3. Inside the main class invoke the getter *i.e.*, getNumber() and the setter *i.e.*, setNumber() in order to read or update the variable,

For example :

```
public static void main (String args[ ])
{
SimpleGetterAndSetter obj = new SimpleGetterAndSetter();
obj.setNumber(10);
System.out.println (obj.getNumber ( ));
}
```

4. Getter and setter are also known as accessor and mutator in Java.

Que 4.5. How to prepare a class to be a Java Beans ?

Answer

Java Beans is a Java class that should have following conventions:

1. It must implement serializable interface.
2. It should have a public constructor without argument.
3. All properties in Java Bean must be private with public getter and setter methods.

Example of Java Bean class :

//Employee.java

```
package mypack;


public class Employee implements java.io.Serializable{
private int id;
private String name;
public Employee(){}
public void setId(int id){this.id=id;}
public int getId(){return id;}
public void setName(String name){this.name=name;}
public String getName(){return name;}
}
```

To access the Java Bean class, we should use getter and setter methods :

```
package mypack;
public class Test{
public static void main(String args[]){
Employee e=new Employee();    //object is created
e.setName("Arjun");    //setting value to the object
System.out.println(e.getName());        // getting value to the object
}}
```

Que 4.6. Explain Java Bean class properties.

Answer

Java Bean class contains three types of properties :

1. **Simple properties :**
   a. A simple property has a single value.
   b. It can be identified by the following design patterns, where *N* is the name of the property and *T* is its type.

      public T getN();

      public void setN(T parameter)
   c. If the property has both read and write permission then both the get and set methods can access the values. Otherwise, only one method can access the values.
   d. If the property has only read permission then only get method can access the values, similarly if the property has only write permission then only set method can access the values.
2. **Boolean properties :**
   a. A boolean property has a value of true or false.
   b. It can be identified by the following design patterns, where *N* is the

name of the property.

public boolean isN();
public void setN(boolean parameter);
public Boolean getN();

c. For getting the values isN and getN methods are used and for setting the Boolean values setN method is used.

3. Indexed properties :

a. An indexed property consists of multiple values.

b. It can be identified by the following design patterns, where *N* is the name of the property and *T* is its type.

public T getN(int index);
public void setN(int index, T value);
public T[ ] getN( );
public void setN(T values[ ]);

PART-2

Types of Beans, Stateful Session Bean, Stateless Session Bean, Enlity Bean.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.7. Explain session beans with its types.

Answer

Session bean :

1. Session bean encapsulates business logic only, it can be invoked by local, remote and web service client.
2. It can be used for managing activities like database access, calculation etc.
3. The life cycle of session bean is maintained by the application server (EJB container).
4. Session bean is created by a customer and its duration is only for the signal client server session.

Types of session bean :

1. Stateless session bean :

a. Stateless session bean is a business object that represents business logic only. It does not have state (data).

b. The stateless bean objects are pooled by the EJB container to service the request on demand.

c. It can be accessed by one client at a time.

d. The stateless session bean is distributed object which has no connection with informal state; only allow parallel access to beans.

e. Annotations used in stateless session bean are :

i. @Stateless

ii. @PostConstruct

iii. @PreDestroy

2. Stateful session bean :

a. Stateful session bean is a business object that represents business logic like stateless session bean. But, it maintains state (data).

b. Conversational state between multiple method calls is maintained by the container in stateful session bean.

c. There are five important annotations used in stateful session bean :

i. @Stateful

ii. @PostConstruct

iii. @PreDestroy

iv. @PrePassivate

v. @PostActivate

3. Singleton session beans :

a. A singleton session bean is instantiated once per application and exists for the lifecycle of the application.

b. Singleton session beans are designed for circumstances in which a single enterprise bean instance is shared across and concurrently accessed by clients.

c. It has only one singleton session bean per application.

d. It can implement web service endpoints.

e. Singleton session beans maintain their state between client invocations but are not required to maintain their state across server crashes or shutdowns.

Que 4.8. Describe entity beans with its types.

Answer

1. Entity beans are objects that represent a persistence storage mechanism.
2. Each entity beans has underlying table in a relational database and each row in the table represents the instance of the bean.

Types of entity beans :

1. Container Managed Persistence (CMP) :

a. The term Container Managed Persistence means that the EJB container handles all database access required by the entity bean.

b. The bean code contains no database access calls. As a result, the bean code is not tied to a specific persistent storage mechanism (database).

c. If the same entity beans are implemented on different J2EE servers that use different databases, we do not need to modify or recompile the bean code.

2. Bean Managed Persistence (BMP) :

a. In this method, the entity bean provides an object view of the data.

b. A Bean Managed Persistence mechanism transforms the physical data structure to a Java object.

c. The entity bean has code that accesses the persistence environment directly.

d. BMP can be used to reduce the overhead of CMP.

Que 4.9. What do you mean by session bean ? Explain its types using suitable example.

Answer

Session bean and its types : Refer Q. 4.7, Page 4–8D, Unit-4.

Example : An example of stateless session bean is a stock quote component that returns the current price of a given stock symbol. Such a bean could look up the stock price from a database that is updated by a real-time feed.

An example of a stateful session bean is a shopping cart that represents the collection of products selected by a particular customer for purchase during a session. The shopping cart should not be shared because it represents a particular interaction with a particular customer and is alive only for the customer's session.

PART-3

Java Database Connectivity (JDBC) : Merging Data from Multiple Tables : Joining.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.10. What is JDBC ? How it works ?

Answer

1. **J**DBC (**J**ava Database Connectivity) is a **J**ava API that manages connection to database, issuing queries and commands and handling result sets obtained from the database.
2. JDBC is useful for both application developers and JDBC driver vendors.
3. JDBC is specially used for having connectivity with the RDBMS packages using corresponding JDBC driver.

Working of JDBC :

1. All Java application establishes connection with the data source and invokes classes and interfaces from JDBC driver for sending queries to the data source.
2. The JDBC driver connects to corresponding database and retrieves the result.
3. These results are based on SQL statements, which are then returned to Java applications.
4. Java application then uses the retrieved information for further processing.

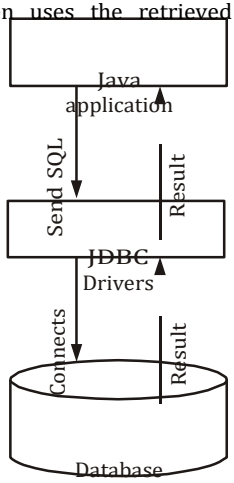

Fig. 4.10.1.

Que 4.11. What are the components of JDBC ?

Answer

Components of JDBC :

1. **Driver manager :**
   a. When Java applications need connection to the database it invokes the DriverManager class.

b. This class then loads JDBC drivers in the memory. The driver manager also attempts to open a connection with the desired database.

2. **Connection :**
   a. This is an interface which represents connectivity with the data source.
   b. The connection is used for creating the statement instance.
3. **Statement :**
   a. This interface is used for representing the SQL statements.
   b. Some SQL statements are :
      SELECT *FROM students _ table;
      UPDATE students _table set name = 'Nitin' WHERE roll _ no = '1';
   c. There are two specialised statement types : PreparedStatement and callableStatement.
4. **ResultSet :**
   a. This interface is used to represent the database resultSet.
   b. After using SELECT SQL statement, the information obtained from the database can be displayed using ResultSet.
5. **SQL exception :** For handling SQL exceptions, this interface is used.

Que 4.12. Explain JDBC application architecture.

Answer

JDBC architecture :

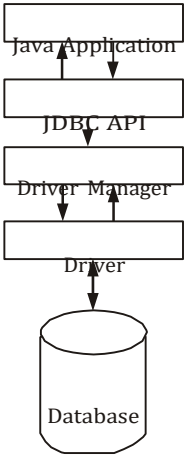

Fig. 4.12.1.

1. **Java application :** It is a standalone Java program which uses the JDBC API to get connected and perform operations on the database data.
2. **JDBC API :** It is a set of classes and interfaces used in a Java program for database operations. Java.sql and Javax.sql packages provide the necessary library support.
3. **Driver manager :** Java program uses DriverManager class to get the connection with the database.
4. **Driver :** It is the software that establishes connection with the database. It is the translation software that translates the JDBC method calls. This software enables the communication between Java program and the database.
5. **Database :** It is a collection of all enterprise data.

Que 4.13. Explain the types of JDBC drivers.

Answer

Types of JDBC drivers are :

1. **JDBC-ODBC bridge driver (Type 1 driver) :**
   a. These drivers are the bridge drivers such as JDBC-ODBC bridge.
   b. These drivers rely on an intermediary such as ODBC to transfer the SQL calls to the database.
   c. Bridge drivers often rely on native code, although the JDBC-ODBC library native code is part of the Java-2 virtual machine.

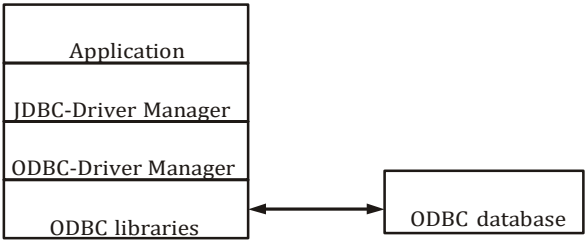

Fig. 4.13.1. JDBC-ODBC bridge driver.

2. **Native API partly Java driver (Type 2 driver) :**
   a. A native API is partly a Java driver. It uses native C language library calls to translate JDBC to native client library.
   b. These drivers are available for Oracle, Sybase, DB2 and other client library based RDBMS.

c. Type 2 drivers use native code and require additional permission to work in an Applet.

d. A Type 2 driver might need client-side database code to connect over the network.

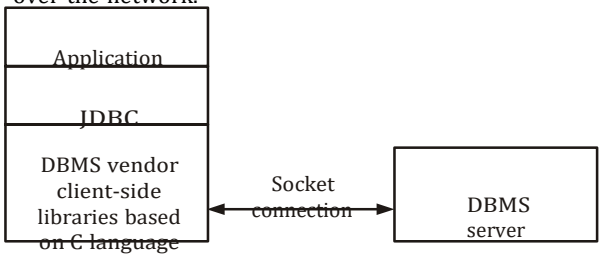

Fig. 4.13.2. Native API partly Java driver.

3. **JDBC net pure Java driver (Type 3 driver) :**
   a. JDBC net pure Java driver consists of JDBC and DBMS independent protocol driver.
   b. Here the calls are translated and sent to middle tier server through the socket.
   c. The middle tier contacts the database.
   d. Type 3 drivers call the database API on the server.
4. **Native protocol pure Java driver (Type 4 driver) :**
   a. A native protocol Java driver contains JDBC calls that are converted directly to the network protocol used by the DBMS server.
   b. This driver interacts directly with database.

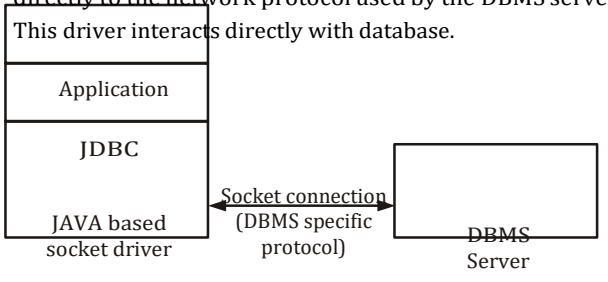

Fig. 4.13.3. Native protocol pure Java driver.

   c. It does not require any native database library. So, it is also called thin driver.

Que 4.14. What is JDBC ? Explain the drivers used in JDBC. Write a JDBC program for insert and display the record of employees

using prepared statement. **AKTU 2018-19, Marks 07**

Answer

JDBC : Refer Q. 4.10, Page 4–10D, Unit-4.

Drivers in JDBC : Refer Q. 4.13, Page 4–13D, Unit-4.

Program :

```
import java.sql.*; import java.io.*;
public class PreparedStatementDemo1 {
Connection con;
PreparedStatement ps;
public PreparedStatementDemo1() {
try {
Class.forName("com.mysql.jdbc.Driver");
con = DriverManager.getConnection("jdbc:mysql://localhost/
test?user=root&password=root");
} catch (Exception e) { e.printStackTrace(); } }
// add customer detail
public String addCustomer(String custid, String name, String address,
String contact) {
String status = "";
try {
ps = con.prepareStatement("insert into Customer values(?,?,?,?)");
ps.setString(1, custid); ps.setString(2, name);
ps.setString(3, address); ps.setString(4, contact);
int i = ps.executeUpdate();
if (i != 0) {
status = "Inserted";
} else {
status = "Not Inserted"; }
} catch (Exception e) { e.printStackTrace(); }
return status; }
// customer record
public void searchCustomer(String custid) {
String sql = "";
if (custid.trim().length() == 0) {
sql = "select * from Customer";
} else {
sql = "select * from Customer where custid=" + custid + ""; }
try {
ps = con.prepareStatement(sql);
ResultSet res = ps.executeQuery();
while (res.next()) {
System.out.print(res.getString(1));
System.out.print(res.getString(2));
System.out.print(res.getString(3));
System.out.println(res.getString(4)); }
```

```
} catch (SQLException e) {e.printStackTrace(); } }
```

```
public String deleteCustomer(String custId) {
String status = "";
try {

ps = con.prepareStatement("delete from Customer where custid=?");
ps.setString(1, custId);
int i = ps.executeUpdate();
if (i != 0) {
status = "Customer details deleted";
} else {
status = "Customer details not deleted"; }
} catch (Exception e) {e.printStackTrace(); }
return status; }
public void menuDisplay() {
try {
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
int ch = 0;
while (true) {
System.out.println("== Customer Management System == \n"
+ "1. Add Customer \n" + "2. Display Customer's record \n"
+ "3. Exit \n" + "Enter Choice \n");
String str1 = br.readLine().toString();
ch = Integer.parseInt(str1);
switch (ch) {
case 1: {
System.out.println("Enter Customer Id");
String custId = br.readLine();
System.out.println("Enter Customer Name");
String custName = br.readLine();
System.out.println("Enter Customer Address");
String custAddress = br.readLine();
System.out.println("Enter Customer Contact No.");
String custContact = br.readLine();
System.out.println(addCustomer(custId, custName,
custAddress, custContact));
break; }
case : 2{
System.out.println("Enter Customer Code to display record");
String custId = br.readLine();
searchCustomer(custId);
break; }
case 3: {
System.exit(0); }
default:
break; } }
} catch (Exception e) {
e.printStackTrace(); } }
```

```
public static void main(String[] args) {
PreparedStatementDemo1 obj = new
PreparedStatementDemo1();
obj.menuDisplay(); } }
```

PART-4

Manipulating, Databases with JDBC, Prepared Statements Transaction Processing, Stored Procedures.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.15. Write a Java program to retrieve data from multiple tables.

Answer

```
import java.sql.*;
public class jdbcConn
{
public static void main(String[] args) throws Exception
{
Class.forName("org.apache.derby.jdbc.ClientDriver");
Connection con = DriverManager.getConnection (
"jdbc:derby://localhost:1527/testDb","username", "password");
Statement stmt = con.createStatement();
String query ="SELECT fname,lname,isbn FROM author INNER JOIN
books ON author.AUTHORID = books.AUTHORID";
ResultSet rs = stmt.executeQuery(query);
System.out.println("Fname Lname ISBN");
while (rs.next())
    {
        String fname = rs.getString("fname");
        String lname = rs.getString("lname");
        int isbn = rs.getInt("isbn");
        System.out.println(fname + "  " + lname+"  "+isbn);
    }
System.out.println();
System.out.println();
}
```

}

Output :
Fname Lname ISBN
Jatin Garg 123
Pankaj Sharma 113
Pankaj Sharma 112
Pankaj Sharma 122

Que 4.16. Explain JDBC application architecture. What are the various types of JDBC drivers ? Write steps to connect database with the web application using JDBC. AKTU 2015-16, Marks 15

Answer

JDBC application architecture : Refer Q. 4.12, Page 4–12D, Unit-4.
Types of JDBC driver : Refer Q. 4.13, Page 4–13D, Unit-4.

Steps to connect database with web application using JDBC :

Step 1 : Create a database using some suitable database management package.

Step 2 : Initiate object for JDBC driver using following statement :

Class.forName ("com.mysql.jdbc.Driver"). newInstance ( );

Step 3 : Using DriverManager class and getConnection method we get connected to the database.

To get connected with MySQL database we use following statement :

DriverManager.getConnection ("jdbc:mysql://localhost; 3306/students", "root", "system");

Que 4.17. Explain PreparedStatement interface in JDBC.

Answer

1. Prepared statement interface is a subinterface of statement.
2. It is used to execute parameterized query.
3. The PreparedStatement interfaces define the methods and properties that enable us to send SQL or PL/SQL commands and receive data from our database.
4. This statement gives us the flexibility for supplying arguments dynamically.
5. Syntax to create PreparedStatement object :

```
PreparedStatement pstmt = null;
try {
String SQL = "Update Employees SET age = ? WHERE id = ?";
pstmt = conn.prepareStatement(SQL);
```

...

```
}

catch (SQLException e) {
. . .
}
finally {
pstmt.close();
}
```

6. All parameters in JDBC are represented by the ? symbol, which is known as the parameter marker. We must supply values for every parameter before executing the SQL statement.
7. To close the PreparedStatement object a simple call to the close() method is made. If we close the connection object first, it will close the PreparedStatement object as well.

Que 4.18. Explain the steps to connect a Java application with database using JDBC.

Answer

Steps to connect a Java application with database :

1. **Register the driver :**
   Class.forName() is used to load the driver class explicitly.
   Example :
   Class.forName("sun.jdbc.odbc.**J**dbcOdbcDriver");
2. **Create a connection :**
   a. getConnection() method of DriverManager class is used to create a connection.
   b. Syntax :
      getConnection(String url)
      getConnection(String url, String username, String password)
      getConnection(String url, Properties info)
      Example : Establishing connection with Oracle Driver
      Connection con = DriverManager.getConnection
      ("jdbc:oracle:thin:@localhost:1521:XE","username","password");
3. **Create SQL statement :**
   a. createStatement() method is invoked on current connection object to create a SQL Statement.
   b. Syntax :
      public Statement createStatement() throws SQLException
      Example :
      Statement s=con.createStatement();

4. Execute SQL statement :

a. executeQuery() method of statement interface is used to execute SQL statements.

b. Syntax :

public ResultSet executeQuery(String query) throws SQLException

Example :

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

5. Closing the connection :

a. After executing SQL statement, to close the connection and release the session.

b. The close() method of connection interface is used to close the connection.

c. Syntax :

public void close() throws SQLException

Example :

```
con.close();
```

Que 4.19. What do you mean by database drivers, explain each type ? Also explain the steps to get any value into database.

Answer

1. A database driver is a computer program that implements a protocol (ODBC or JDBC) for a database connection.
2. The driver works like an adaptor which connects a generic interface to a specific database vendor implementation.

Types of database drivers : Refer Q. 4.13, Page 4–13D, Unit-4.

Steps to get any value into database : Refer Q. 4.14, Page 4–14D, Unit-4.

Que 4.20. Write a short note on stored procedure in Java.

Answer

1. A program which contains *n* number of SQL statements and residing a database environment is known as stored procedure.
2. Stored procedures are divided into two types :

a. Procedure :

i. A procedure is one which contains block of statements which will return either zero or more than one value.

ii. Syntax for creating a procedure :

```
create procedure <procedure name> (parameters)
as/is
    local variables;
begin
    block of statements;
end;
```

b. Function :

i. A function is one which contains *n* number of block of statements to perform some operation and it returns a single value only.

ii. Syntax for creating a function :

```
create function (a in number, b in number) return <return type>
as/is
    n1 out number;
begin
    n1:=a+b;
    return (n1);
end;
```

Que 4.21. Write the difference between stored procedure and functions.

Answer

Differences :

S. No.	Stored procedure	Function
1.	It is used to perform business logic.	It is used to perform calculation.
2.	It may or may not have the return type.	It must have the return type.
3.	It may return more than one values.	It may return only one value.
4.	We can call functions from the procedure.	Procedure cannot be called from function.
5.	Procedure supports input and output parameters.	Function supports only input parameter.
6.	Exception handling using try/catch block can be used in stored procedures.	Exception handling using try/catch block cannot be used in user-defined functions.

Que 4.22. Explain transaction management in JDBC. What are the types of transaction ?

Answer

1. A transaction is a group of operation used to perform single task.
2. If all operations in the group are successful then the task is finished and the transaction is successfully completed.
3. If any one operation in the group is failed then the task is failed and the transaction is failed.

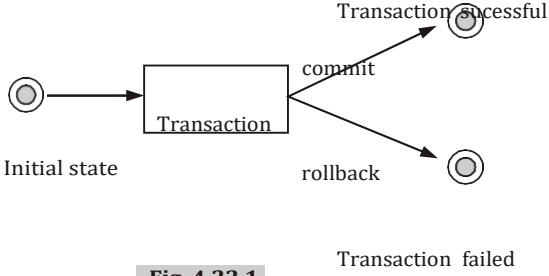

Fig. 4.22.1.

Types of transaction :

1. **Local transaction :** A local transaction means that all operations in a transaction are executed against one database.

For example : If we transfer money from first account to second account and both accounts belongs to same bank then transaction is local transaction.

2. **Global transaction :** A global transaction means that all operations in a transaction are executed against multiple databases.

For example : If we transfer money from first account to second account belongs to different banks then the transaction is a global transaction.

Que 4.23. Describe the transaction management method in JDBC with example.

Answer

In JDBC, Connection interface provides different methods to manage transaction.

Method	Description
void setAutoCommit(boolean status)	void commit() void rollback()

transaction is committed by default.

commits the transaction.

cancels the transaction.

For Example :

```
import java.sql.*;
class TrxaExample
{


public static void main(String[ ] args)throws Exception
{
Class.forName("oracle.jdbc.OracleDriver");
Connection con=DriverManager.getConnection("jdbc:oracle:thin:@quantum-pc:1521:xe","system","system");
System.out.println("driver is loaded");
Statement stmt=con.createStatement();
con.setAutoCommit(false);
try
{
int i1=stmt.executeUpdate("insert into student values(110,'quantum',685)");
int i2=stmt.executeUpdate("update customer set custadd='Ghaziabad'where custid=111");
int i3=stmt.executeUpdate("delete from student where sid=101");
con.commit();
System.out.println("Transaction is successful");
}                    //end of try
catch (Exception e)
{
try
{
con.rollback();
System.out.println("Transaction is failed");
}
catch (Exception ex)
{
System.out.println(ex);
}
}                    //end of catch
stmt.close();
con.close();
System.out.println("connection is closed");
}                    //end of main
}                    //end of class
```

VERY IMPORTANT QUESTIONS

Following questions are very important. These questionsmay be asked in your SESSIONALS as well as in UNIVERSITY EXAMINATION.

Q. 1. Discuss EJB. Explain EJB architecture. What are its various types ?

Ans. Refer Q. 4.2.

Q. 2. What is Java Bean exactly ? Write down the steps to create Java Bean. What is the role of introspection in Java Bean ?

Ans. Refer Q. 4.3.

Q. 3. Explain JavaBeans. Why they are used ? Discuss setter and getter method with Java code.

Ans. Refer Q. 4.4.

Q. 4. Explain session beans with its types.

Ans. Refer Q. 4.7.

Q. 5. What is JDBC ? How it works ?

Ans. Refer Q. 4.10.

Q. 6. Explain the types of JDBC drivers.

Ans. Refer Q. 4.13.

Q. 7. Explain JDBC application architecture. What are the various types of JDBC drivers ? Write steps to connect database with the web application using JDBC.

Ans. Refer Q. 4.16.