Advanced Programming

Week 6

JDBC

- Steps to connect to a Database in Java
- Manipulating Database with JDBC
- JDBC Classes and Interfaces

Tilahun Melak(PhD)

October, 2025

Objectives

At the end of this lecture, students will be able to:

- Explain the steps required to connect to a database in Java.
- Load JDBC drivers.
- Insert records into a database using JDBC.
- Manipulate database data using JDBC.
- Retrieve and interpret database metadata.

Manipulating Database with JDBC

Steps to connect to the database in java

- There are 5 steps to connect any java application with the database in java using JDBC. They are as follows:

1. Register the driver class
2. Creating connection
3. Creating statement
4. Executing queries
5. Closing connection

Oracle. (n.d.). JDBC basics.

Manipulating Database with JDBC Cntd..

Steps to connect to the database in java

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.

Manipulating Database with JDBC Cntd..

Steps to connect to the database in java

1) Register the driver class

Syntax of forName() method

public static void forName(String className)throws ClassNotFoundException

Example to register the Mysql /Oracle Driver class

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

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

Manipulating Database with JDBC Cntd..

Steps to connect to the database in java

2) Create the connection object

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

Manipulating Database with JDBC Cntd..

Steps to connect to the database in java

2) Create the connection object

Syntax of getConnection() method

```
1) public static Connection getConnection(String url)throws SQLException
2) public static Connection getConnection(String url,String name,String password)
throws SQLException
```

Manipulating Database with JDBC Cntd..

Steps to connect to the database in java

2) Create the connection object

Example to establish connection with the MySQL database

```
Connection con=DriverManager.getConnection("jdbc:mysql://hostname:port/dbName",“un",“PW");
```

//here db is database name, un is username and PW is password

Manipulating Database with JDBC Cntd..

Steps to connect to the database in java

3) Create the Statement object

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

Manipulating Database with JDBC Cntd..

Steps to connect to the database in java

3) Create the Statement object

- **Syntax of createStatement() method**

public Statement createStatement()throws SQLException

Example: Statement stmt=con.createStatement();

Manipulating Database with JDBC Cntd..

Steps to connect to the database in java

4) Execute the query

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

Manipulating Database with JDBC Cntd..

Steps to connect to the database in java

4) Execute the query

- **Syntax of executeQuery() method**

```
public ResultSet executeQuery(String sql)throws SQLException
```

Oracle. (n.d.). JDBC basics.

Manipulating Database with JDBC Cntd..

Steps to connect to the database in java

4) Execute the query

Example to execute query

```
ResultSet rs=stmt.executeQuery("select * from stud");

while(rs.next())

{

System.out.println(rs.getInt(1)+" "+rs.getString(2));

}
```

Oracle. (n.d.). JDBC basics.

Manipulating Database with JDBC Cntd..

Steps to connect to the database in java

5) 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.

Manipulating Database with JDBC Cntd..

Steps to connect to the database in java

5) Close the connection object

Syntax of close() method

public void close()throws SQLException

Example : con.close();

Example: Manipulating Database with JDBC

```
import java.sql.*;

class JDatabase
{
public static void main(String args[]){
try{
Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/student","root","root");
```

Example: Manipulating Database with JDBC

```
Statement stmt=con.createStatement();

ResultSet rs=stmt.executeQuery("select * from stud");
 System.out.println("ID\t\tName\tGender\tDept");
 System.out.print("-----------------------------------\n");
```

Example: Manipulating Database with JDBC

```
 while(rs.next())
{
System.out.println(rs.getString(1)+"\t"+rs.getString(2)+"\t"+rs.getString(3)
       +"\t"+rs.getString(4));
}
System.out.println("-----------------------------------");
  con.close();
  }catch(Exception e)
{
    e.printStackTrace();
}
}
}
```

JDBC Classes and Interfaces

- **DriverManager class**
  - The DriverManager class acts as an interface between user and drivers.
  - It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver.
  - The DriverManager class maintains a list of Driver classes that have registered themselves by calling the method DriverManager.registerDriver().

Oracle. (n.d.). The JDBC DriverManager class. In Java Platform, Standard Edition Documentation.

JDBC Classes and Interfaces Cntd…

- **DriverManager class**
- **Commonly used methods of DriverManager class:**

public static Connection getConnection(String url);

public static Connection getConnection(String url,String userName,String password);

Oracle. (n.d.). The JDBC DriverManager class. In Java Platform, Standard Edition Documentation.

JDBC Classes and Interfaces Cntd…

- **Connection interface**
  - A **Connection** is the session between java application and database.
  - The Connection interface is a factory of **Statement**, **PreparedStatement**, and **DatabaseMetaData** i.e. object of Connection can be used to get the object of Statement and DatabaseMetaData.

Oracle. (n.d.). The JDBC DriverManager class. In Java Platform, Standard Edition Documentation.

JDBC Classes and Interfaces Cntd…

- **Connection interface**
  - The Connection interface provide many methods for transaction management like **commit(),rollback()** etc.
  - By default, connection commits the changes after executing queries.

Oracle. (n.d.). The JDBC DriverManager class. In Java Platform, Standard Edition Documentation.

JDBC Classes and Interfaces Cntd…

- **Connection interface**

Commonly used methods of Connection interface:

1) public Statement createStatement(): creates a statement object that can be used to execute SQL queries.

Oracle. (n.d.). The JDBC DriverManager class. In Java Platform, Standard Edition Documentation.

JDBC Classes and Interfaces Cntd…

- ## Connection interface

Commonly used methods of Connection interface:

2) public Statement createStatement(int resultSetType,int resultSetConcurrency): Creates a Statement object that will generate ResultSet objects with the given type and concurrency.

Oracle. (n.d.). The JDBC DriverManager class. In Java Platform, Standard Edition Documentation.

JDBC Classes and Interfaces Cntd…

- ## Connection interface

Commonly used methods of Connection interface:

3) public void setAutoCommit(boolean status): is used to set the commit status. By default it is true.

4) public void commit(): saves the changes made since the previous commit/rollback permanent.

Oracle. (n.d.). The JDBC DriverManager class. In Java Platform, Standard Edition Documentation.

JDBC Classes and Interfaces Cntd…

- **Statement interface**
- The **Statement interface** provides methods to execute queries with the database.
- The statement interface is a factory of ResultSet i.e. it provides factory method to get the object of ResultSet.

Oracle. (n.d.). The JDBC DriverManager class. In Java Platform, Standard Edition Documentation.

JDBC Classes and Interfaces Cntd…

- **Statement interface**

Methods of Statement interface

1) public ResultSet executeQuery(String sql): is used to execute SELECT query. It returns the object of ResultSet.

2) public int executeUpdate(String sql): is used to execute specified query, it may be create, drop, insert, update, delete etc.

3) public boolean execute(String sql): is used to execute queries that may return multiple results.

4) public int[] executeBatch(): is used to execute batch of commands.

Oracle. (n.d.). The JDBC DriverManager class. In Java Platform, Standard Edition Documentation.

Example: Inserting Records

```
import java.sql.*;
    class InsertRecord{
    public static void main(String args[])throws Exception{
      try
      {
    Class.forName("com.mysql.jdbc.Driver");
    Connection
     con=DriverManager.getConnection("jdbc:mysql://localhost:3306/student",
                "root","root");
    Statement stmt=con.createStatement();
```

Example: Inserting Records Cntd…

```
 //int result=stmt.executeUpdate("insert into stud
 values('r/104/08','Ibrahim','Male','Economics')");
//int result=stmt.executeUpdate("update stud set Name='Jemal' where
 id='r/104/08'");
int result=stmt.executeUpdate("delete from stud where id='r/104'");
System.out.println(result+" records affected");
con.close();
 }
  catch(Exception e)
  {
     e.printStackTrace();
  }  }   }
```

Summary

- In today's lecture we have discussed about;
  - The steps to connect to a database in Java.
  - Manipulate database data using JDBC.
  - JDBC Classes and Interfaces.
  - Examples

References

- Oracle. (n.d.). JDBC basics.
- Oracle. (n.d.). The JDBC DriverManager class. In Java Platform, Standard Edition Documentation.