ACE Inspiration

Java Web Development Course
Chapter - 9 JDBC

- Java Database Connectivity

Content

- JDBC
- Why use JDBC?
- JDBC Thin Driver
- JDBC with MySQL
- Workout

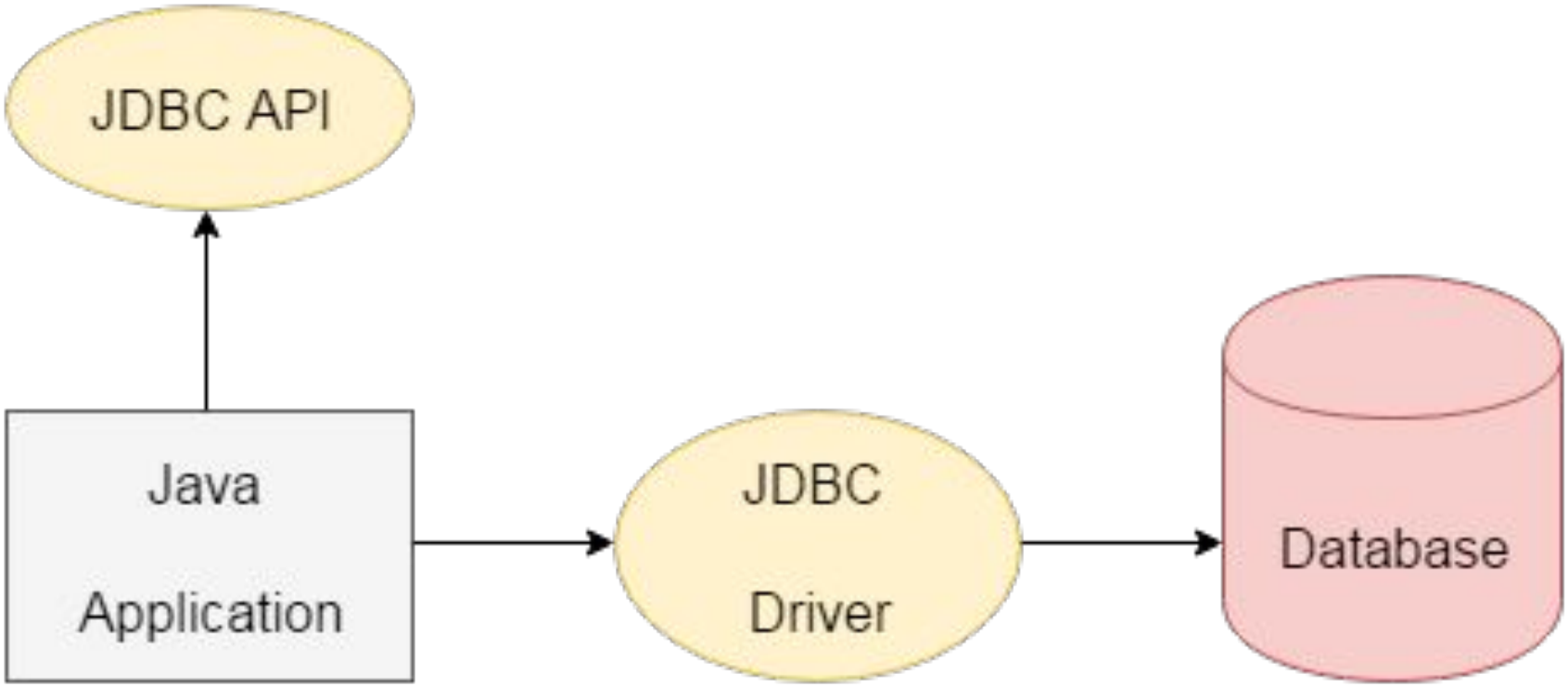
JDBC

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

- JDBC-ODBC Bridge Driver,
- Native Driver,
- Network Protocol Driver, and
- Thin Driver

We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database.

JDBC



Why use JDBC?

Before JDBC, ODBC API was the database API to connect and execute the 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).

We can use JDBC API to handle database using Java program and can perform the following activities:

- Connect to the database
- Execute queries and update statements to the database
- Retrieve the result received from the database.

JDBC 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 depend on the Database.

JDBC with MySQL

ACE Inspiration

To connect Java application with the MySQL database, we need to follow 5 following steps.In this example we are using MySql as the database. So we need to know following informations for the mysql database:

Driver class: The driver class for the mysql database is **com.mysql.jdbc.Driver**.

Connection URL: The connection URL for the mysql database is **jdbc:mysql://localhost:3306/mydb** where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we may also use IP address, 3306 is the port number and sonoo is the database name. We may use any database, in such case, we need to replace the sonoo with our database name.

Username: The default username for the mysql database is **root**.

Password: It is the password given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.

JDBC with MySQL

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class MyConnection {
public static void main(String [] args) {
try {
    Class.forName("com.mysql.jdbc.Driver");
    Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/test",
"username","password");
} catch (ClassNotFoundException e) {
    System.out.println("Driver class not found");
} catch (SQLException e) {
    System.out.println("Database Coonectin not found");
}
}
}
```

JDBC with MySQL(Insert)

ACE Inspiration

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
public class InsertData {
public static Connection myConnection(){
Connection con=null;
try {
Class.forName("com.mysql.jdbc.Driver");
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/test", "username", "password");
} catch (ClassNotFoundException e) {
System.out.println("Driver class not found");
} catch (SQLException e) {
System.out.println("Database Coonectin not found");
}
return con;
}
```

JDBC with MySQL(Insert)

```
public static void main(String []args) {
Connection con=myConnection();
try {
PreparedStatement ps=con.prepareStatement("insert into employee
(name,age,address,salary) values (?,?,?,?)");
ps.setString(1, "John");
ps.setInt(2, 23);
ps.setString(3, "Hlaing");
ps.setDouble(4, 1000);
ps.executeUpdate();
} catch (SQLException e) {
System.out.println(e.getMessage());
}
System.out.println("Rest of code");
}
}
```

JDBC with MySQL(Update)

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
public class UpdateData {
public static Connection myConnection(){
Connection con=null;
try {
Class.forName("com.mysql.jdbc.Driver");
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/test"," username", "password");
} catch (ClassNotFoundException e) {
System.out.println("Driver class not found");
} catch (SQLException e) {
System.out.println("Database Coonectin not found");
}
return con;
}
```

JDBC with MySQL(Update)

ACE Inspiration

```
public static void main(String []args) {
Connection con=myConnection();
try {
PreparedStatement ps=con.prepareStatement("update employee set
name=?,age=?,address=?,salary=? where id=?");
ps.setString(1, "David");
ps.setInt(2, 25);
ps.setString(3, "Insein");
ps.setDouble(4, 1500);
ps.setInt(5, 1);
ps.execute();
} catch (SQLException e) {
System.out.println(e.getMessage());
}
System.out.println("Rest of code");
}
}
```

JDBC with MySQL(Delete)

ACE Inspiration

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
public class DeleteData {
public static Connection myConnection(){
Connection con=null;
try {
Class.forName("com.mysql.jdbc.Driver");
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/test", "username", "password");
} catch (ClassNotFoundException e) {
System.out.println("Driver class not found");
} catch (SQLException e) {
System.out.println("Database Coonectin not found");
}
return con;
}
```

JDBC with MySQL(Delete)

ACE Inspiration

```
public static void main(String []args) {
Connection con=myConnection();
try {
PreparedStatement ps=con.prepareStatement("delete from employee
where id=?");
ps.setInt(1, 1);
ps.execute();
} catch (SQLException e) {
System.out.println(e.getMessage());
}
System.out.println("Rest of code");
}
}
```

JDBC with MySQL(Select)

ACE Inspiration

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
public class DeleteData {
public static Connection myConnection(){
Connection con=null;
try {
Class.forName("com.mysql.jdbc.Driver");
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/test", "username", "password");
} catch (ClassNotFoundException e) {
System.out.println("Driver class not found");
} catch (SQLException e) {
System.out.println("Database Coonectin not found");
}
return con;
}
```

JDBC with MySQL(Select)

ACE Inspiration

```
public static void main(String []args) {
Connection con=myConnection();
try {
PreparedStatement ps=con.prepareStatement("select * from employee where id=?");
ps.setInt(1, 2);
ResultSet rs=ps.executeQuery();
while(rs.next()) {
System.out.println("ID "+rs.getInt("id"));
System.out.println("Name "+rs.getString("name"));
System.out.println("Age "+rs.getInt("age"));
System.out.println("Adress "+rs.getString("address"));
System.out.println("Salary "+rs.getDouble("salary"));
}
} catch (SQLException e) {
System.out.println(e.getMessage());
}
} 16
}
```

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().

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. The Connection interface provide many methods for transaction management like commit(), rollback() etc.

PreparedStatement interface

The PreparedStatement interface is a subinterface of Statement. It is used to execute parameterized query.

Method	Description
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 for create, drop, insert, update, delete etc.
public ResultSet executeQuery()	executes the select query. It returns an instance of ResultSet.

ResultSet interface

The object of ResultSet maintains a cursor pointing to a row of a table. Initially, cursor points to before the first row.

Method	Description
1) public boolean next():	is used to move the cursor to the one row next from the current position.
2) public boolean previous():	is used to move the cursor to the one row previous from the current position.
3) public boolean first():	is used to move the cursor to the first row in result set object.
4) public boolean last():	is used to move the cursor to the last row in result set object.
5) public boolean absolute(int row):	is used to move the cursor to the specified row number in the ResultSet object.

JDBC with MySQL

ACE Inspiration

Method	Description
6) public boolean relative(int row):	is used to move the cursor to the relative row number in the ResultSet object, it may be positive or negative.
7) public int getInt(int columnIndex):	is used to return the data of specified column index of the current row as int.
8) public int getInt(String columnName):	is used to return the data of specified column name of the current row as int.
9) public String getString(int columnIndex):	is used to return the data of specified column index of the current row as String.
10) public String getString(String columnName):	is used to return the data of specified column name of the current row as String.

Workout

Write the following embedded SQL queries, based on the database schema

Product	
maker	VARCHAR (30)
model	VARCHAR (30)
type	VARCHAR (30)

Printer	
model	VARCHAR (30)
color	DECIMAL (1,0)
type	VARCHAR (10)
price	INTEGER

PC	
model	VARCHAR (30)
speed	VARCHAR (10)
ram	VARCHAR (10)
hd	VARCHAR (10)
price	INTEGER

Laptop	
model	VARCHAR (30)
speed	VARCHAR (10)
ram	VARCHAR (10)
hd	VARCHAR (10)
price	INTEGER

Workout

ACE Inspiration

Exercises

1. Ask the user for a price and the PC whose price is closest to the desired price. Print the maker, model number, and speed of the PC.

2. Ask the user for minimum values of the speed, RAM, hard-disk size, and screen size that they will accept. Find all the laptops that satisfy these requirements. Print their specifications (all attributes of Laptop) and their manufacturer.

3. Ask the user for a manufacturer. Print the specifications of all products by that manufacturer. That is, print the model number, product-type, and all the attributes of whichever relation is appropriate for that type.

4. Ask the user for a \budget" (total price of a PC and printer), and a minimum speed of the PC. Find the cheapest \system" (PC plus printer) that is within the budget and minimum speed, but make the printer a color printer if possible. Print the model numbers for the chosen system.

5. Ask the user for a manufacturer, model number, speed, RAM, hard-disk size, and price of a new PC. Check that there is no PC with that model number. Print a warning if so, and otherwise insert the information into tables Product and PC.

ACE Inspiration

Thank you!!

Q&As

References

- http://www.tutorialspoint.com/java/
- http://www.javatpoint.com/java/
- http://stackoverflow.com/questions/4014535/differences-in-boolean-operators-vs-and-vs
- https://examples.javacodegeeks.com/