

Properties

=====

It represents the data in the form of K,V pair
It represents an object of type java.util.Properties
Properties object holds the data which would be changing frequently in our application.

Program to demonstrate the usage of Properties object in java

=====

```
import java.io.*;
import java.util.*;
class PropertiesApp
{
    public static void main(String[] args) throws Exception
    {
        FileInputStream fis = new FileInputStream("application.properties");
        Properties properties = new Properties();
        properties.load(fis);

        String url = properties.getProperty("url");
        String user = properties.getProperty("user");
        String password = properties.getProperty("password");

        System.out.println("URL IS ::"+url);
        System.out.println("USER IS ::"+user);
        System.out.println("PWD IS ::"+password);
    }
}
```

application.properties

=====

```
url=jdbc:oracle:thin:@localhost:1521:XE
user=System
password=root123
```

Output

D:\jdbcocotbatch>javac PropertiesApp.java

D:\jdbcocotbatch>java PropertiesApp

URL IS ::jdbc:mysql:///octbatch

USER IS ::root

PWD IS ::root123

D:\jdbcocotbatch>java PropertiesApp

URL IS ::jdbc:oracle:thin:@localhost:1521:XE

USER IS ::System

PWD IS ::root123

PreparedStatement(I)

=====

public abstract PreparedStatement prepareStatement(String query) throws SQLException;

While using PreparedStatement the query would be as shown below

```
String sqlInsertQuery ="insert into
student(`sname`,`sage`,`saddress`,`sgender`)values(?,?,?,?)";
PreparedStatement pstmt = con.prepareStatement(sqlInsertQuery);
```

```
//query compiled ready for execution
pstmt.setString(1,"apeksha");
pstmt.setInt(2,26);
pstmt.setString(3,"MI");
pstmt.setString(4,"F");
```

```
int rowCount = pstmt.executeUpdate();
```

```
refer: JDBCPreparedStatementApp
```

JDBC Project Requirement

=====

1. Develop a console based application in the following manner
 - a. Create a menu for the user to perform CRUD operation as shown below
 1. Press 1 for Insert operation
 2. Press 2 for select operation
 3. Press 3 for Update operation
 4. Press 4 for Delete operation
 5. Press 5 for exit

Note: anything above 5 tell invalid operation
 - b. If user presses 1
 - a. Take inputs from the user to accept the data like
 1. sid(pk) 2. sname 3. sage 4. saddress
 - b. perform insertion operation
 - c. display suitable message as
 - a. record inserted succesfully
 - b. record insertion failed
 - c. If user presses 2
 - a. Take inputs from the user to accept the data like
 1. sid(pk)
 - b. perform select operation
 - c. display suitable message as
 - a. display the details in table format.
 - b. record not available for the given id.
 - d. b. If user presses 3
 - a. Take inputs from the user to accept the data like
 1. sid(pk)
 - b. for the entered id display the details first


```
sid      : XXXXX (no change becoz it is pk)
sname    : XXXXX  enter new sname  :: XXXXX
sage     : XXXXX  enter new sage   :: YYYYY
saddr    : XXXXX  enter new saddr  :: YYYYY
```
 - c. display suitable message as
 - a. record updated succesfully
 - b. record updation failed
 - e. If user presses 4
 - a. Take inputs from the user to accept the data like
 1. sid(pk)
 - b. perform delete operation
 - c. display suitable message as
 - a. record deleted succesfully.
 - b. record not available for the given id.

Advantages of PreparedStatement

=====

1. Performance is very high compared to Statement approach becoz query will be compiled only once.
2. Since we don't send the query multiple times b/w java application and database traffic will be reduced.
3. inputs to the query need not be supplied at the beginning dynamically we can supply the inputs.
4. inputs to the query can be supplied just like java style, no need to perform formatting as per the DB specification.
5. Best suitable for inserting Date values.
6. Best suitable for insertion of BLOB's and CLOB's (image and pdf files).
7. It prevents SQLInjection Attack.

Limitation of PreparedStatement

=====

```
Statement stmt = connection.createStatement();//[samsung phone ---->
airtelsim,jiosim,vi]GSMA
    stmt.executeUpdate("insert into student values ()...");
    stmt.executeQuery("select * from student");
    stmt.executeUpdate("delete from student wher....");
```

One statement object can be used to execute multiple query but with no change in inputs.

```
PreparedStatement pstmt = connection.prepareStatement("select * from
student");//[Jio Phone----> jio sim]CDMA
    pstmt.executeQuery();
```

One PreparedStatement object is restricted to only one query, that query can be executed multiple times with change in input.

Which of the following represent valid statements ?

1. delete from employee where ename = ?(valid)
2. delete from employee ? ename = ? (invalid)
3. delete from ? where ename = ? (invalid)
4. delete ? employees where ename = ?(invalid)

Note: we can use ? only in the place of input values and we cannot use in the place of sql keywords, tablename and column names.

Static query vs Dynamic query

=====

The sql query without positional parameter(?) is called static query.

eg: delete from employee where ename = 'sachin';

The sql query with positional parameter(?) is called dynamic query

eg: delete from employee where ename = ?
select eid,ename,esal from employee where esal > ?

Note:

Simple Statement object can be used for static queries, where as PreparedStatement object can be used for static queries and dynamic queries also.

