Batch Processing in JDBC

Instead of executing a single query, we can execute a batch (group) of queries. It makes the performance fast.

The java.sql.Statement and java.sql.PreparedStatement interfaces provide methods for batch processing.

Advantage of Batch Processing

Fast Performance

Methods of Statement interface

The required methods for batch processing are given below:

|  |  |
| --- | --- |
| **Method** | **Description** |
| void addBatch(String query) | It adds query into batch. |
| int[] executeBatch() | It executes the batch of queries. |

Example of batch processing in jdbc

Let's see the simple example of batch processing in jdbc. It follows following steps:

* Load the driver class
* Create Connection
* Create Statement
* Add query in the batch
* Execute Batch
* Close Connection

1. **import** java.sql.\*;
2. **class** FetchRecords{
3. **public** **static** **void** main(String args[])**throws** Exception{
4. Class.forName("oracle.jdbc.driver.OracleDriver");
5. Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
6. con.setAutoCommit(**false**);
8. Statement stmt=con.createStatement();
9. stmt.addBatch("insert into user420 values(190,'abhi',40000)");
10. stmt.addBatch("insert into user420 values(191,'umesh',50000)");
12. stmt.executeBatch();//executing the batch
14. con.commit();
15. con.close();
16. }}

If you see the table user420, two records has been added.

Example of batch processing using PreparedStatement

1. **import** java.sql.\*;
2. **import** java.io.\*;
3. **class** BP{
4. **public** **static** **void** main(String args[]){
5. **try**{
7. Class.forName("oracle.jdbc.driver.OracleDriver");
8. Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
10. PreparedStatement ps=con.prepareStatement("insert into user420 values(?,?,?)");
12. BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.in));
13. **while**(**true**){
15. System.out.println("enter id");
16. String s1=br.readLine();
17. **int** id=Integer.parseInt(s1);
19. System.out.println("enter name");
20. String name=br.readLine();
22. System.out.println("enter salary");
23. String s3=br.readLine();
24. **int** salary=Integer.parseInt(s3);
26. ps.setInt(1,id);
27. ps.setString(2,name);
28. ps.setInt(3,salary);
30. ps.addBatch();
31. System.out.println("Want to add more records y/n");
32. String ans=br.readLine();
33. **if**(ans.equals("n")){
34. **break**;
35. }
37. }
38. ps.executeBatch();
40. System.out.println("record successfully saved");
42. con.close();
43. }**catch**(Exception e){System.out.println(e);}
45. }}

It will add the queries into the batch until user press n. Finally it executes the batch. Thus all the added queries will be fired.