ICON OF JAVA

ADV. JAVA

JDBC

12.Connection Pooling

Mr. Nagoor Babu M.Tech

[ICON of JAVA]
(Sun certified & Realtime Expert)

Ex. HCL Employee

Trained Lakhs of Students
for last 14 years across INDIA

India's No.1 Software Training Institute

DURGASOFT

www.durgasoft.com Ph: 9246212143 ,8096969696

Connection Pooling

In general in Jdbc applications, when we have a requirement to perform database operations we will establish the connection with the database from a Java application, at the end of the application we will close the connection i.e. destroying Connection object.

In Jdbc applications, every time establishing the connection and closing the connection may increase burden to the Jdbc application, it will reduce the performance of the jdbc application.
In the above context, to improve the performance of Jdbc applications we will use an alternative called as Connection Pooling.

In Connection pooling at the time of application startup we will prepare a fixed number of Connection objects and we will keep them in a separate base object called Pool object.
In Jdbc applications, when we have a requirement to interact with the database then we will get the Connection object from Pool object and we will assign it to the respective client application.

At the end of the Jdbc application we will keep the same Connection object in the respective Pool object without destroying.
The above mechanism will improve the performance of the application is called as Connection Pooling.

If we want to implement Connection pooling in Jdbc application we have to use the following steps.

Step 1: Prepare DataSource object.

DataSource is an object, it is able to manage all the Jdbc parameter which are required to establish the connections.

To represent DataSource object Java API has provided a predefined interface i.e. javax.sql.DataSource.

DataSource is an interface provided by Jdbc API, but whose implementation classes are provided by all the database vendors.

With the above convention Oracle has provided an implementation class to DataSource interface in ojdbc6.jar file i.e. oracle.jdbc.pool.OracleConnectionPoolDataSource.

Ex: OracleConnectionPoolDataSource ds=new OracleConnectionPoolDataSource();

Step 2: Set the required Jdbc parameters to DataSource object.
To set the Jdbc parameters like Driver url, database username and password to the DataSource object we have to use the following methods.

public void setURL(String driver_url)
public void setUser(String user_name)
public void setPassword(String password)

Ex: ds.setURL("jdbc:oracle:thin:@localhost:1521:xe");
ds.setUser("system");
ds.setPassword("venkat");

Step 3: Get the PooledConnection object.

PooledConnection is an object provided by DataSource, it can be used to manage number of Connection objects.

To represent PooledConnection object Jdbc API has provided a predefined interface i.e. javax.sql.PooledConnection.

To get PooledConnection object we have to use the following method from DataSource.
public PooledConnection getPooledConnection()

Ex: PooledConnection pc=ds.getPooledConnection();

Step 4: Get Connection object from PooledConnection.

To get Connection object from PooledConnection we have to use the following method.
public Connection getConnection()

Ex: Connection con=pc.getConnection();

Step 5: After getting Connection prepare Statement or preparedStatement or CallableStatement and perform the respective database operations.

Ex: Statement st=con.createStatement();

JdbcApp39:

```
import java.sql.*;
import javax.sql.*;
import oracle.jdbc.pool.*;
public class JdbcApp39{
public static void main(String[] args) throws Exception{
OracleConnectionPoolDataSource ds=new OracleConnectionPoolDataSource();
ds.setURL("jdbc:oracle:thin:@localhost:1521:xe");
ds.setUser("system");
ds.setPassword("durga");
PooledConnection pc=ds.getPooledConnection();
Connection con=pc.getConnection();
Statement st=con.createStatement();
ResultSet rs=st.executeQuery("select * from emp");
System.out.println("EID ENAME ESAL");
System.out.println("------------------------------");
while (rs.next())
{
System.out.println(rs.getInt(1)+" "+rs.getString(2)+"
"+rs.getFloat(3));
}
}
}
```

Note: The above approach of implementing Connection pooling is suggestible up to standalone applications, it is not suggestible in enterprise applications. If we want to implement Connection pooling in enterprise applications we have to use the underlying application server provided Jdbc middleware server.

India's No.1 JAVA Trainer
More Than 1000 Students in a Single Class Room.
Weapon of DURGASOFT
Mr.Nagoor Babu M.Tech
Trained Lakhs of Students for Last 13years (Realtime Expert & Java Certified)

LEARN FROM EXPERTS ...

COMPLETE JAVA

CORE JAVA, ADV. JAVA, ORACLE, STRUTS, HIBERNATE, SPRING, WEB SERVICES,...

COMPLETE .NET

C#.NET, ASP.NET, SQL SERVER, MVC 5 & WCF

TESTING TOOLS

MANUAL + SELENIUM

ORACLE **D2K**

MSBI **SHARE POINT**

HADOOP **ANDROID**

C, C++, DS, UNIX

CRT & APTITUDE TRAINING

AN ISO 9001:2008 CERTIFIED

DURGA

Software Solutions®

202, 2nd Floor, HUDA Maitrivanam,
Ameerpet, Hyd. Ph: 040-64512786,
9246212143, 8096969696

www.durgasoft.com