# Connection Pooling

**If we required to communicate with database multiple times then it is not recommended to create separate Connection object every time, b'z creating and destroying Connection object every time creates performance problems.**

**To overcome this problem, we should go for Connection Pool.**

**Connection Pool is a pool of already created Connection objects which are ready to use.**

**If we want to communicate with database then we request Connection pool to provide Connection. Once we got the Connection, by using that we can communicates with database. After completing our work, we can return Connection to the pool instead of destroying.**

**Hence the main advantage of Connection Pool is we can reuse same Connection object multiple times, so that overall performance of application will be improved.**

## Process to implement Connection Pooling:

### 1. Creation of DataSource object

**DataSource is responsible to manage connections in Connection Pool.**

**DataSource is an interface present in javax.sql package.**

**Driver Software vendor is responsible to provide implementation.**

**Oracle people provided implementation class name is :**
**OracleConnectionPoolDataSource.**

**This class present inside oracle.jdbc.pool package and it is the part of ojdbc6.jar.**

```
OracleConnectionPoolDataSource  ds= new OracleConnectionPoolDataSource();
```

### 2. Set required JDBC Properties to the DataSource object:

```
ds.setURL("jdbc:oracle:thin:@localhost:1521:XE");
ds.setUser("scott");
ds.setPassword("tiger");
```

### 3. Get Connection from DataSource object:

```
Connection con = ds.getConnection();
```
**Once we got Connection object then remaining process is as usual.**

## Program to Demonstrate Connection Pooling for Oracle Database:

```
1)  import java.sql.*;
2)  import javax.sql.*;
3) import oracle.jdbc.pool.*;// ojdbc6.jar
4)  class ConnectionPoolDemoOracle
5) {
6)      public static void main(String[] args) throws Exception
7)      {
8)         OracleConnectionPoolDataSource ds = new OracleConnectionPoolDataSource();
9)         ds.setURL("jdbc:oracle:thin:@localhost:1521:XE");
10)        ds.setUser("scott");
11)        ds.setPassword("tiger");
12)        Connection con=ds.getConnection();
13)        Statement st =con.createStatement();
14)        ResultSet rs=st.executeQuery("select * from employees");
15)        System.out.println("ENO\tENAME\tESAL\tEADDR");
16)        System.out.println("--------------------------");
17)        while(rs.next())
18)        {
19)           System.out.println(rs.getInt(1)+"\t"+rs.getString(2)+"\t"+rs.getFloat(3)+"\t"+rs.getS
    tring(4));
20)        }
21)        con.close();
22)     }
23) }
```

## Program to Demonstrate Connection Pooling for MySQL Database:

```
1)  import java.sql.*;
2)  import javax.sql.*;
3)  import com.mysql.jdbc.jdbc2.optional.*;
4)  class ConnectionPoolDemoMySql
5)   {
6)      public static void main(String[] args) throws Exception
7)      {
8)         MysqlConnectionPoolDataSource ds = new MysqlConnectionPoolDataSource();
9)         ds.setURL("jdbc:mysql://localhost:3306/secedb");
10)        ds.setUser("root");
11)        ds.setPassword("root");
12)        Connection con=ds.getConnection();
13)        Statement st =con.createStatement();
14)        ResultSet rs=st.executeQuery("select * from employees");
15)        System.out.println("ENO\tENAME\tESAL\tEADDR");
16)        System.out.println(" ------------------------- ");
17)        while(rs.next())
18)        {
19)           System.out.println(rs.getInt(1)+"\t"+rs.getString(2)+"\t"+rs.getFloat(3)+"\t"+rs.getS
    tring(4));
20)        }
```

```
21)         con.close();
22)     }
23) }
```

**Note:** **This way of implementing Connection Pool is useful for Standalone applications. In the case of web and enterprise applications, we have to use server level connection pooling. Every web andapplication server can provide support for Connection Pooling.**

**Q. What is the difference Between getting Connection object by using DriverManager andDataSource object?**

**In the case of DriverManager.getConnection(), always a new Connection object will be createdand returned.**

**But in the case of DataSourceObject.getConnection(), a new Connection object won't be createdand existing Connection object will be returned from Connection Pool.**