

DBCP简介

DBCP(DataBase connection pool)数据库连接池是 apache 上的一个Java连接池项目。DBCP通过连接池预先同数据库建立一些连接放在内存中(即连接池中)，应用程序需要建立数据库连接时直接到从接池中申请一个连接使用，用完后由连接池回收该连接，从而达到连接复用，减少资源消耗的目的。

1、DBCP所依赖的jar包(以下例子基于如下jar包版本)

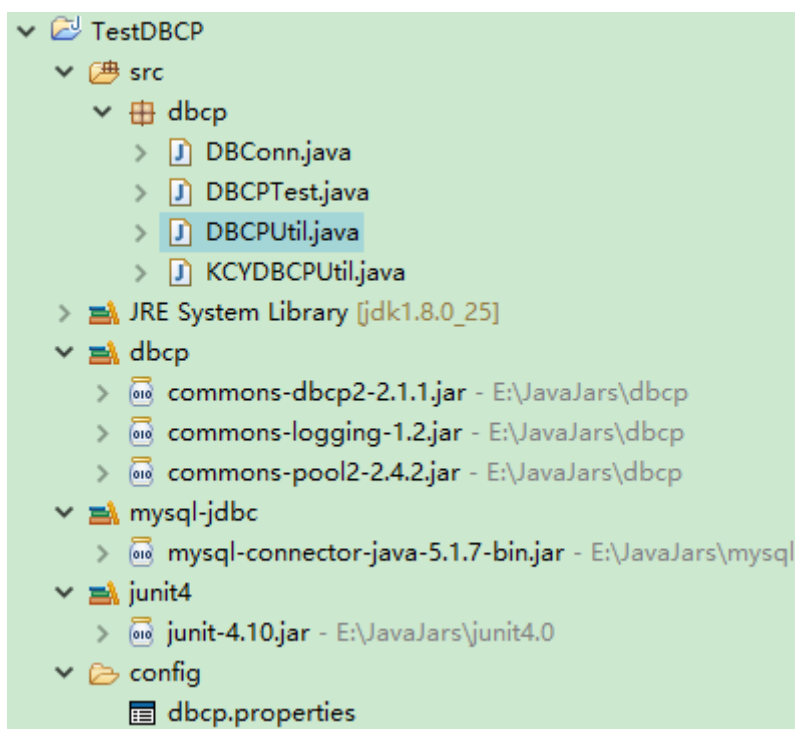
commons-dbcp2-2.1.1.jar commons-logging-1.2.jar commons-pool2-2.4.2.jar

2、DBCP使用示例

下图是在Eclipse中创建的Java工程，使用了DBCP相关的jar包，mysql的jdbc驱动jar包，junit4。

并在src同级目录下创建了config目录，用于存放DBCP的配置文件。

【注】类DBCPUtil.java在下面的例子中未用到。



1) DBCP配置文件dbcp.properties



```
#####DBCP配置文件#####
```

```
#驱动名
```

```
driverClassName=com.mysql.jdbc.Driver
```

```
#url
```

```
url=jdbc:mysql://127.0.0.1:3306/mydb
```

```
#用户名
```

```
username=sa
```

```
#密码
password=123456
#初试连接数
initialSize=30
#最大活跃数
maxTotal=30
#最大idle数
maxIdle=10
#最小idle数
minIdle=5
#最长等待时间(毫秒)
maxWaitMillis=1000
#程序中的连接不使用后是否被连接池回收(该版本要使用removeAbandonedOnMaintenance和
removeAbandonedOnBorrow)
#removeAbandoned=true
removeAbandonedOnMaintenance=true
removeAbandonedOnBorrow=true
#连接在所指定的秒数内未使用才会被删除(秒)(为配合测试程序才配置为1秒)
removeAbandonedTimeout=1
```



2) 创建初始化DBCP的类KCYDBCUtil.java



```
1 package dbcp;
2
3 import java.io.FileInputStream;
4 import java.io.IOException;
5 import java.sql.Connection;
6 import java.sql.SQLException;
7 import java.util.Properties;
8
9 import javax.sql.DataSource;
10
11 import org.apache.commons.dbcp2.BasicDataSourceFactory;
12
13 /**
14  * DBCP配置类
15  * @author SUN
16  */
17 public class KCYDBCUtil {
18
```

```
19     private static Properties properties = new Properties();
20     private static DataSource dataSource;
21     //加载DBCP配置文件
22     static{
23         try{
24             FileInputStream is = new
FileInputStream("config/dbcp.properties");
25             properties.load(is);
26         }catch(IOException e){
27             e.printStackTrace();
28         }
29
30         try{
31             dataSource = BasicDataSourceFactory.createDataSource(properties);
32         }catch(Exception e){
33             e.printStackTrace();
34         }
35     }
36
37     //从连接池中获取一个连接
38     public static Connection getConnection(){
39         Connection connection = null;
40         try{
41             connection = dataSource.getConnection();
42         }catch(SQLException e){
43             e.printStackTrace();
44         }
45         try {
46             connection.setAutoCommit(false);
47         } catch (SQLException e) {
48             e.printStackTrace();
49         }
50         return connection;
51     }
52
53     public static void main(String[] args) {
54         getConnection();
55     }
56 }
```



3) 创建使用JDBC获取数据库连接的类DBConn.java (用于和DBCP连接池对比)



```
1 package dbcp;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5
6 public class DBConn {
7     private static Connection conn = null;
8
9     //获取一个数据库连接
10    public static Connection getConnection() {
11        try {
12            Class.forName("com.mysql.jdbc.Driver");
13            DriverManager.registerDriver(new com.mysql.jdbc.Driver());
14            String dbUrl = "jdbc:mysql://127.0.0.1:3306/mydb";
15            conn = DriverManager.getConnection(dbUrl, "sa", "123456");
16            // System.out.println("=====数据库连接成功=====");
17        } catch (Exception e) {
18            e.printStackTrace();
19            // System.out.println("=====数据库连接失败=====");
20            return null;
21        }
22        return conn;
23    }
24 }
```



4) 创建测试类DBCPTTest.java

测试类中采用3中方法将2000个数据插入数据库同一张表中，每次插入数据之前，先清空表，并对结果进行了对比。

3中插入数据方法如下：

- (1) 每次插入一条数据前，就创建一个连接，该条数据插入完成后，关闭该连接；
- (2) 使用DBCP连接池，每次插入一条数据前，从DBCP连接池中获取一条连接，该条数据插入完成后，该连接交由DBCP连接池管理；
- (3) 在插入数据之前创建一条连接，2000个数据全部使用该连接，2000个数据插入完毕后，关闭该连接。



```
1 package dbcp;
2
3 import java.sql.Connection;
4 import java.sql.SQLException;
5 import java.sql.Statement;
6
7 import org.junit.Test;
8
9 public class DBCPTest {
10
11     //测试,每写一条数据前,就新建一个连接
12     @Test
13     public void testWriteDBByEveryConn() throws Exception{
14         for(int i = 0; i < 2000; i++){
15             writeDBByEveryConn(i);
16         }
17         System.out.println("DONE");
18
19     }
20
21     //测试,使用连接池,每写一条数据前,从连接池中获取一个连接
22     @Test
23     public void testWriteDBByDBCP() throws Exception{
24         for(int i = 0; i < 2000; i++){
25             writeDBByDBCP(i);
26         }
27         System.out.println("DONE");
28     }
29
30     //测试,只建一条连接,写入所有数据
31     @Test
32     public void testWriteDBByOneConn() throws Exception{
33         Connection conn = DBConn.getConnection();
34         Statement stat = conn.createStatement();
35         for(int i = 0; i < 2000; i++){
36             writeDBByOneConn(i, stat);
37         }
38         conn.close();
39         System.out.println("DONE");
40     }
41 }
```

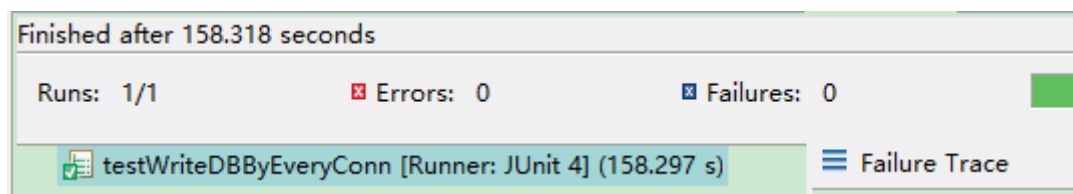
```
42 //不使用连接池写数据库,每写一条数据创建一个连接
43 public void writeDBByEveryConn(int data){
44     String sql = "insert into dbcp values (" + data + ")";
45     Connection conn = DBConn.getConnection();
46     try{
47         Statement stat = conn.createStatement();
48         stat.executeUpdate(sql);
49     }catch(Exception e){
50         e.printStackTrace() ;
51     }finally{
52         try {
53             conn.close();
54         } catch (SQLException e) {
55             e.printStackTrace();
56         }
57     }
58 }
59 }
60
61 //不使用连接池写数据库,只用一个连接,写所有数据
62 public void writeDBByOneConn(int data, Statement stat){
63     String sql = "insert into dbcp values (" + data + ")";
64     try{
65         stat.executeUpdate(sql);
66     }catch(Exception e){
67         e.printStackTrace() ;
68     }
69 }
70
71 //通过DBCP连接池写数据库
72 public void writeDBByDBCP(int data){
73     String sql = "insert into dbcp values (" + data + ")";
74     try {
75         Connection conn = KCYDBCPUtil.getConnection();
76         Statement stat = conn.createStatement();
77         stat.executeUpdate(sql);
78         conn.commit();
79         conn.close();
80     } catch (SQLException e) {
81         e.printStackTrace();
82     }
```

```
83 }  
84  
85 }
```

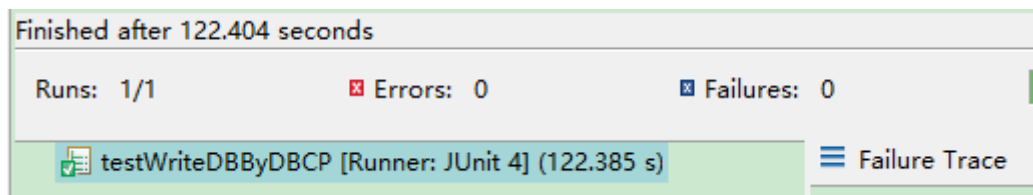


测试结果如下：

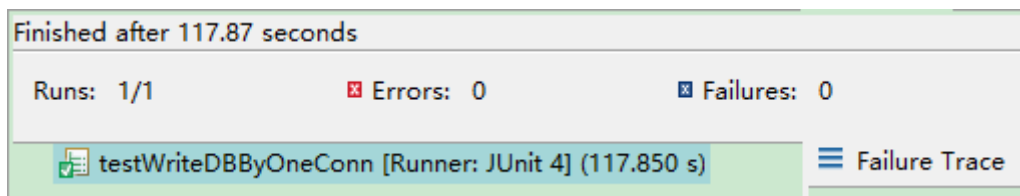
(1) 每次插入一条数据前，就创建一个连接，该条数据插入完成后，关闭该连接。耗时158.318秒



(2) 使用DBCP连接池，每次插入一条数据前，从DBCP连接池中获取一条连接，该条数据插入完成后，该连接交由DBCP连接池管理。耗时122.404秒



(3) 在插入数据之前创建一条连接，2000个数据全部使用该连接，2000个数据插入完毕后，关闭该连接。耗时117.87秒



通过对比结果看出，向同一个表中插入2000条数据，每插入一条数据前创建一个新连接，会非常耗时，而使用DBCP连接池和使用同一个连接操作，耗时比较接近。

3、相关问题

1) 应用程序中，使用完一个数据库连接后，DBCP连接池如何管理该连接。

分两种情况：

(1) 应用程序中主动关闭该连接，即DBCPTTest.java中第79行 `conn.close();`;

这种情况并不是手动将该连接关闭，而是将该连接交回给DBCP连接池，由连接池管理该连接。即用完连接后显示的将数据库连接提交至DBCP连接池。

(2) 应用程序中不关闭该连接，即将DBCPTTest.java中第79行 `conn.close();`注释掉

这种情况DBCP配置文件dbcp.properties中的配置项(注意jar包版本, 低版本中使用removeAbandoned=true配置项)

```
removeAbandonedOnMaintenance=true  
removeAbandonedOnBorrow=true
```

```
removeAbandonedTimeout=1
```

会起作用, *removeAbandonedOnMaintenance=true*和*removeAbandonedOnBorrow=true*表示DBCP连接池自动管理应用程序中使用完毕的连接, *removeAbandonedTimeout=1*表示一个连接在程序中使用完毕后, 若在1秒之内没有再次使用, 则DBCP连接池回收该连接(通常removeAbandonedTimeout不会配置1, 此处为了测试使用)。

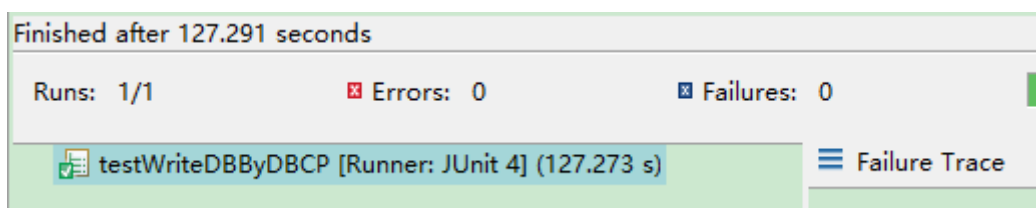
(3) 验证*removeAbandonedOnMaintenance=true*, *removeAbandonedOnBorrow=true*和*removeAbandonedTimeout=1*配置项的作用
将测试类DBCPTTest.java的writeDBByDBCP(int data)方法修改为如下:



```
1 //通过DBCP连接池写数据库  
2 public void writeDBByDBCP(int data){  
3     String sql = "insert into dbcp values (" + data + ")";  
4     try {  
5         Connection conn = KCYDBCPUtil.getConnection();  
6         Statement stat = conn.createStatement();  
7         stat.executeUpdate(sql);  
8         conn.commit();  
9         // conn.close();  
10    } catch (SQLException e) {  
11        e.printStackTrace();  
12    }  
13 }
```

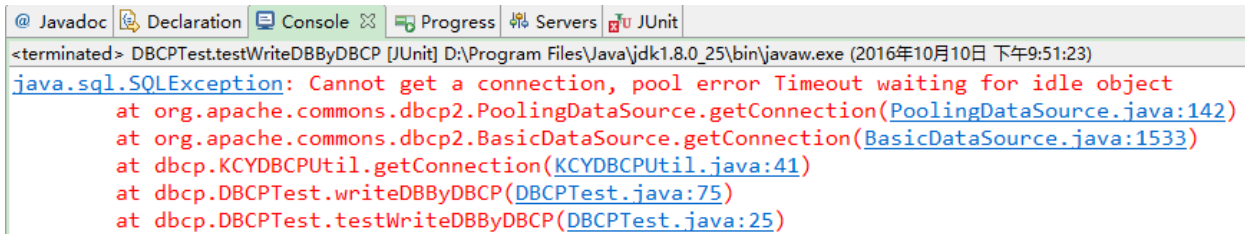


重新执行testWriteDBByDBCP()方法, 结果如下:



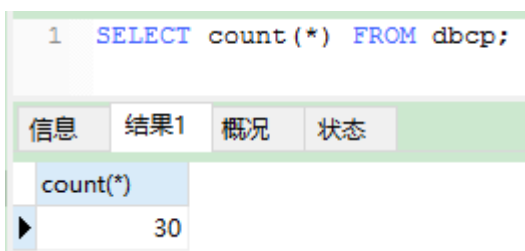
可见writeDBByDBCP(int data)方法修改后和修改前作用相同, 说明连接使用完后, 由DBCP连接池管理。

而如果将修改配置项`removeAbandonedTimeout=180`，即一个连接用完后会等待180秒，超过180秒后才由DBCP连接池回收，重新执行`testWriteDBByDBCP()`方法，执行一段时间后报错(Cannot get a connection, pool error Timeout waiting for idle object)，如下：



```
<terminated> DBCPTest.testWriteDBByDBCP [JUnit] D:\Program Files\Java\jdk1.8.0_25\bin\javaw.exe (2016年10月10日 下午9:51:23)
java.sql.SQLException: Cannot get a connection, pool error Timeout waiting for idle object
    at org.apache.commons.dbcp2.PoolingDataSource.getConnection(PoolingDataSource.java:142)
    at org.apache.commons.dbcp2.BasicDataSource.getConnection(BasicDataSource.java:1533)
    at dbc.KCYDBCUtil.getConnection(KCYDBCUtil.java:41)
    at dbc.DBCPTest.writeDBByDBCP(DBCPTest.java:75)
    at dbc.DBCPTest.testWriteDBByDBCP(DBCPTest.java:25)
```

此时，查询数据表，发现正好插入了30条数据，如下：



```
1 SELECT count(*) FROM dbcp;
```

信息	结果1	概况	状态
	count(*)		
	30		

这说明在插入第31条数据的时候报错，错误原因是连接池中没有可用的连接了。这是因为DBCP连接池初始化连接数为30，`removeAbandonedTimeout`设为180秒，所以30个连接用完后，程序运行还未到180秒，程序中用完的连接都还没有被DBCP连接池回收，所以DBCP连接池中没有可用的连接了，才会在插入第31条数据时报错。