//数据库连接池 单例模式

import java.sql.Connection;

import java.sql.SQLException;

import com.mchange.v2.c3p0.ComboPooledDataSource;

import com.mchange.v2.c3p0.DataSources;

public final class ConnectionManager {

private static ConnectionManager instance;

private ComboPooledDataSource ds;

private ConnectionManager() throws Exception {

ds = new ComboPooledDataSource();

ds.setDriverClass("oracle.jdbc.driver.OracleDriver");

ds.setJdbcUrl("jdbc:oracle:thin:@127.0.0.1:1521:orcl");

ds.setUser("test");

ds.setPassword("testtest");

//初始化时获取三个连接，取值应在minPoolSize与maxPoolSize之间。Default: 3 initialPoolSize

ds.setInitialPoolSize(3);

//连接池中保留的最大连接数。Default: 15 maxPoolSize

ds.setMaxPoolSize(10);

//// 连接池中保留的最小连接数。

//ds.setMinPoolSize(1);

//当连接池中的连接耗尽的时候c3p0一次同时获取的连接数。Default: 3 acquireIncrement

ds.setAcquireIncrement(1);

//每60秒检查所有连接池中的空闲连接。Default: 0 idleConnectionTestPeriod

ds.setIdleConnectionTestPeriod(60);

//最大空闲时间,25000秒内未使用则连接被丢弃。若为0则永不丢弃。Default: 0 maxIdleTime

ds.setMaxIdleTime(25000);

//连接关闭时默认将所有未提交的操作回滚。Default: false autoCommitOnClose

ds.setAutoCommitOnClose(true);

//定义所有连接测试都执行的测试语句。在使用连接测试的情况下这个一显著提高测试速度。注意：

//测试的表必须在初始数据源的时候就存在。Default: null preferredTestQuery

ds.setPreferredTestQuery("select sysdate from dual");

// 因性能消耗大请只在需要的时候使用它。如果设为true那么在每个connection提交的

// 时候都将校验其有效性。建议使用idleConnectionTestPeriod或automaticTestTable

// 等方法来提升连接测试的性能。Default: false testConnectionOnCheckout

ds.setTestConnectionOnCheckout(true);

//如果设为true那么在取得连接的同时将校验连接的有效性。Default: false testConnectionOnCheckin

ds.setTestConnectionOnCheckin(true);

//定义在从数据库获取新连接失败后重复尝试的次数。Default: 30 acquireRetryAttempts

ds.setAcquireRetryAttempts(30);

//两次连接中间隔时间，单位毫秒。Default: 1000 acquireRetryDelay

ds.setAcquireRetryDelay(1000);

//获取连接失败将会引起所有等待连接池来获取连接的线程抛出异常。但是数据源仍有效

//保留，并在下次调用getConnection()的时候继续尝试获取连接。如果设为true，那么在尝试

//获取连接失败后该数据源将申明已断开并永久关闭。Default: false breakAfterAcquireFailure

ds.setBreakAfterAcquireFailure(true);

// <!--当连接池用完时客户端调用getConnection()后等待获取新连接的时间，超时后将抛出

// SQLException,如设为0则无限期等待。单位毫秒。Default: 0 -->

// <property name="checkoutTimeout">100</property>

// <!--c3p0将建一张名为Test的空表，并使用其自带的查询语句进行测试。如果定义了这个参数那么

// 属性preferredTestQuery将被忽略。你不能在这张Test表上进行任何操作，它将只供c3p0测试

// 使用。Default: null-->

// <property name="automaticTestTable">Test</property>

// <!--JDBC的标准参数，用以控制数据源内加载的PreparedStatements数量。但由于预缓存的statements

// 属于单个connection而不是整个连接池。所以设置这个参数需要考虑到多方面的因素。

// 如果maxStatements与maxStatementsPerConnection均为0，则缓存被关闭。Default: 0-->

// <property name="maxStatements">100</property>

// <!--maxStatementsPerConnection定义了连接池内单个连接所拥有的最大缓存statements数。Default: 0 -->

// <property name="maxStatementsPerConnection"></property>

// <!--c3p0是异步操作的，缓慢的JDBC操作通过帮助进程完成。扩展这些操作可以有效的提升性能

// 通过多线程实现多个操作同时被执行。Default: 3-->

// <property name="numHelperThreads">3</property>

// <!--用户修改系统配置参数执行前最多等待300秒。Default: 300 -->

// <property name="propertyCycle">300</property>

}

public static final ConnectionManager getInstance() {

if (instance == null) {

try {

instance = new ConnectionManager();

} catch (Exception e) {

e.printStackTrace();

}

}

return instance;

}

public synchronized final Connection getConnection() {

try {

return ds.getConnection();

} catch (SQLException e) {

e.printStackTrace();

}

return null;

}

protected void finalize() throws Throwable {

DataSources.destroy(ds); //关闭datasource

super.finalize();

}

}

//调用数据库连接池

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.Statement;

public class TestC3p0 {

public TestC3p0() {

// TODO 自动生成构造函数存根

}

public static void main(String[] args) {

// TODO 自动生成方法存根

ConnectionManager cm = ConnectionManager.getInstance();

Connection conn = null;

Statement stmt = null;

ResultSet rs = null;

String sql = "select sysdate from dual";

for (int i = 0; i < 10000; i++) {

try {

conn = cm.getConnection();

System.out.println(conn);

stmt = conn.createStatement();

rs = stmt.executeQuery(sql);

rs.next();

System.out.println(rs.getString(1));

} catch (Exception ex) {

ex.printStackTrace();

} finally {

if (rs != null) {

try {

rs.close();

} catch (Exception e) {

}

}

if (stmt != null) {

try {

stmt.close();

} catch (Exception e) {

}

}

if (conn != null) {

try {

conn.close();

} catch (Exception e) {

}

}

}

//try {

// System.out.println(cm.ds.toString());

// Thread.sleep(10000);

//} catch (Exception e) {

// // TODO: handle exception

//}

}

}

}

官方文档 : http://www.mchange.com/projects/c3p0/index.html

<c3p0-config>

<default-config>

<!--当连接池中的连接耗尽的时候c3p0一次同时获取的连接数。Default: 3 -->

<property name="acquireIncrement">3</property>

<!--定义在从数据库获取新连接失败后重复尝试的次数。Default: 30 -->

<property name="acquireRetryAttempts">30</property>

<!--两次连接中间隔时间，单位毫秒。Default: 1000 -->

<property name="acquireRetryDelay">1000</property>

<!--连接关闭时默认将所有未提交的操作回滚。Default: false -->

<property name="autoCommitOnClose">false</property>

<!--c3p0将建一张名为Test的空表，并使用其自带的查询语句进行测试。如果定义了这个参数那么

属性preferredTestQuery将被忽略。你不能在这张Test表上进行任何操作，它将只供c3p0测试

使用。Default: null-->

<property name="automaticTestTable">Test</property>

<!--获取连接失败将会引起所有等待连接池来获取连接的线程抛出异常。但是数据源仍有效

保留，并在下次调用getConnection()的时候继续尝试获取连接。如果设为true，那么在尝试

获取连接失败后该数据源将申明已断开并永久关闭。Default: false-->

<property name="breakAfterAcquireFailure">false</property>

<!--当连接池用完时客户端调用getConnection()后等待获取新连接的时间，超时后将抛出

SQLException,如设为0则无限期等待。单位毫秒。Default: 0 -->

<property name="checkoutTimeout">100</property>

<!--通过实现ConnectionTester或QueryConnectionTester的类来测试连接。类名需制定全路径。

Default: com.mchange.v2.c3p0.impl.DefaultConnectionTester-->

<property name="connectionTesterClassName"></property>

<!--指定c3p0 libraries的路径，如果（通常都是这样）在本地即可获得那么无需设置，默认null即可

Default: null-->

<property name="factoryClassLocation">null</property>

<!--Strongly disrecommended. Setting this to true may lead to subtle and bizarre bugs.

（文档原文）作者强烈建议不使用的一个属性-->

<property name="forceIgnoreUnresolvedTransactions">false</property>

<!--每60秒检查所有连接池中的空闲连接。Default: 0 -->

<property name="idleConnectionTestPeriod">60</property>

<!--初始化时获取三个连接，取值应在minPoolSize与maxPoolSize之间。Default: 3 -->

<property name="initialPoolSize">3</property>

<!--最大空闲时间,60秒内未使用则连接被丢弃。若为0则永不丢弃。Default: 0 -->

<property name="maxIdleTime">60</property>

<!--连接池中保留的最大连接数。Default: 15 -->

<property name="maxPoolSize">15</property>

<!--JDBC的标准参数，用以控制数据源内加载的PreparedStatements数量。但由于预缓存的statements

属于单个connection而不是整个连接池。所以设置这个参数需要考虑到多方面的因素。

如果maxStatements与maxStatementsPerConnection均为0，则缓存被关闭。Default: 0-->

<property name="maxStatements">100</property>

<!--maxStatementsPerConnection定义了连接池内单个连接所拥有的最大缓存statements数。Default: 0 -->

<property name="maxStatementsPerConnection"></property>

<!--c3p0是异步操作的，缓慢的JDBC操作通过帮助进程完成。扩展这些操作可以有效的提升性能

通过多线程实现多个操作同时被执行。Default: 3-->

<property name="numHelperThreads">3</property>

<!--当用户调用getConnection()时使root用户成为去获取连接的用户。主要用于连接池连接非c3p0

的数据源时。Default: null-->

<property name="overrideDefaultUser">root</property>

<!--与overrideDefaultUser参数对应使用的一个参数。Default: null-->

<property name="overrideDefaultPassword">password</property>

<!--密码。Default: null-->

<property name="password"></property>

<!--定义所有连接测试都执行的测试语句。在使用连接测试的情况下这个一显著提高测试速度。注意：

测试的表必须在初始数据源的时候就存在。Default: null-->

<property name="preferredTestQuery">select id from test where id=1</property>

<!--用户修改系统配置参数执行前最多等待300秒。Default: 300 -->

<property name="propertyCycle">300</property>

<!--因性能消耗大请只在需要的时候使用它。如果设为true那么在每个connection提交的

时候都将校验其有效性。建议使用idleConnectionTestPeriod或automaticTestTable

等方法来提升连接测试的性能。Default: false -->

<property name="testConnectionOnCheckout">false</property>

<!--如果设为true那么在取得连接的同时将校验连接的有效性。Default: false -->

<property name="testConnectionOnCheckin">true</property>

<!--用户名。Default: null-->

<property name="user">root</property>

<!--早期的c3p0版本对JDBC接口采用动态反射代理。在早期版本用途广泛的情况下这个参数

允许用户恢复到动态反射代理以解决不稳定的故障。最新的非反射代理更快并且已经开始

广泛的被使用，所以这个参数未必有用。现在原先的动态反射与新的非反射代理同时受到

支持，但今后可能的版本可能不支持动态反射代理。Default: false-->

<property name="usesTraditionalReflectiveProxies">false</property>

<property name="automaticTestTable">con\_test</property>

<property name="checkoutTimeout">30000</property>

<property name="idleConnectionTestPeriod">30</property>

<property name="initialPoolSize">10</property>

<property name="maxIdleTime">30</property>

<property name="maxPoolSize">25</property>

<property name="minPoolSize">10</property>

<property name="maxStatements">0</property>

<user-overrides user="swaldman">

</user-overrides>

</default-config>

<named-config name="dumbTestConfig">

<property name="maxStatements">200</property>

<user-overrides user="poop">

<property name="maxStatements">300</property>

</user-overrides>

</named-config>

</c3p0-config>

**解决MYSQL 8小时问题**

最近的一个项目在Hibernate使用C3P0的连接池，数据库为Mysql。开发测试没有问题，在运行中每个一段长的空闲时间就出现异常:

java 代码

1. org.hibernate.exception.JDBCConnectionException: could not execute query

2. at org.hibernate.exception.SQLStateConverter.convert(SQLStateConverter.java:74)

3. at org.hibernate.exception.JDBCExceptionHelper.convert(JDBCExceptionHelper.java:43)

4. .......

5. Caused by: com.mysql.jdbc.exceptions.MySQLNonTransientConnectionException: No operations allowed after connection closed.Connection was implicitly closed due to underlying exception/error:

6. \*\* BEGIN NESTED EXCEPTION \*\*

7. com.mysql.jdbc.CommunicationsException

8. MESSAGE: Communications link failure due to underlying exception:

9. \*\* BEGIN NESTED EXCEPTION \*\*

10. java.net.SocketException

11. MESSAGE: Broken pipe

12. STACKTRACE:

13. java.net.SocketException: Broken pipe

14. at java.net.SocketOutputStream.socketWrite0(Native Method)

15. ......

16. \*\* END NESTED EXCEPTION \*\*

查看了Mysql的文档，以及Connector/J的文档以及在线说明发现，出现这种异常的原因是：

Mysql服务器默认的“wait\_timeout”是8小时，也就是说一个connection空闲超过8个小时，Mysql将自动断开该 connection。这就是问题的所在，在C3P0 pools中的connections如果空闲超过8小时，Mysql将其断开，而C3P0并不知道该connection已经失效，如果这时有 Client请求connection，C3P0将该失效的Connection提供给Client，将会造成上面的异常。

解决的方法有3种：

1. 增加wait\_timeout的时间。

2. 减少Connection pools中connection的lifetime。

3. 测试Connection pools中connection的有效性。

当然最好的办法是同时综合使用上述3种方法，下面就DBCP和C3P0分别做一说明，假设wait\_timeout为默认的8小时

DBCP增加以下配置信息:

1. //set to 'SELECT 1'

2. validationQuery = "SELECT 1"

3. //set to 'true'

4. testWhileIdle = "true"

5. //some positive integer

6. timeBetweenEvictionRunsMillis = 3600000

7. //set to something smaller than 'wait\_timeout'

8. minEvictableIdleTimeMillis = 18000000

9. //if you don't mind a hit for every getConnection(), set to "true"

10. testOnBorrow = "true"

C3P0增加以下配置信息:

1. //获取connnection时测试是否有效

2. testConnectionOnCheckin = true

3. //自动测试的table名称

4. automaticTestTable=C3P0TestTable

5. //set to something much less than wait\_timeout, prevents connections from going stale

6. idleConnectionTestPeriod = 18000

7. //set to something slightly less than wait\_timeout, preventing 'stale' connections from being handed out

8. maxIdleTime = 25000

9. //if you can take the performance 'hit', set to "true"

10. testConnectionOnCheckout = true

更多的配置信息大家可以查看C3P0文档，Connector/J文档，以及DBCP的文档。

我自己的配置:

jdbc.driverClass=com.mysql.jdbc.Driver

jdbc.jdbcUrl = jdbc:mysql://localhost:3306/test

jdbc.user = root

jdbc.password = 12345

jdbc.miniPoolSize = 1

jdbc.maxPoolSize = 20

jdbc.initialPoolSize = 1

jdbc.maxIdleTime = 25000

jdbc.acquireIncrement = 1

jdbc.acquireRetryAttempts = 30

jdbc.acquireRetryDelay = 1000

jdbc.testConnectionOnCheckin = true

jdbc.automaticTestTable = c3p0TestTable

jdbc.idleConnectionTestPeriod = 18000

jdbc.checkoutTimeout=3000

<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource" destroy-method="close">

<property name="driverClass" value="${jdbc.driverClass}" />

<property name="jdbcUrl" value="${jdbc.jdbcUrl}" />

<property name="user" value="${jdbc.user}" />

<property name="password" value="${jdbc.password}" />

<property name="minPoolSize" value="${jdbc.miniPoolSize}" />

<property name="maxPoolSize" value="${jdbc.maxPoolSize}"/>

<property name="initialPoolSize" value="${jdbc.initialPoolSize}"/>

<property name="maxIdleTime" value="${jdbc.maxIdleTime}"/>

<property name="acquireIncrement" value="${jdbc.acquireIncrement}"/>

<property name="acquireRetryAttempts" value="${jdbc.acquireRetryAttempts}"/>

<property name="acquireRetryDelay" value="${jdbc.acquireRetryDelay}"/>

<property name="testConnectionOnCheckin" value="${jdbc.testConnectionOnCheckin}"/>

<property name="automaticTestTable" value="${jdbc.automaticTestTable}"/>

<property name="idleConnectionTestPeriod" value="${jdbc.idleConnectionTestPeriod}"/>

<property name="checkoutTimeout" value="${jdbc.checkoutTimeout}"/>

</bean>