# How to configure the C3P0 connection pool in Hibernate

**Connection Pool**  
Connection pool is good for performance, as it prevents Java application create a connection each time when interact with database and minimizes the cost of opening and closing connections.

Hibernate comes with internal connection pool, but not suitable for production use. In this tutorial, we show you how to integrate third party connection pool – C3P0, with Hibernate.

**1. Get hibernate-c3p0.jar**

To integrate c3p0 with Hibernate, you need hibernate-c3p0.jar, get it from JBoss repository.

<!-- Hibernate c3p0 connection pool -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-c3p0</artifactId>

<version>3.6.3.Final</version>

</dependency>

**2. Configure c3p0 propertise**

To configure c3p0, puts the c3p0 configuration details in “hibernate.cfg.xml“, like this :

File : hibernate.cfg.xml

<hibernate-configuration>

<session-factory>

<property name=*"hibernate.connection.driver\_class"*>oracle.jdbc.driver.OracleDriver</property>

<property name=*"hibernate.connection.url"*>jdbc:oracle:thin:@localhost:1521:orcl</property>

<property name=*"hibernate.connection.username"*>srlp</property>

<property name=*"hibernate.connection.password"*> srlp </property>

<property name=*"hibernate.dialect"*>org.hibernate.dialect.Oracle10gDialect</property>

<property name=*"hibernate.default\_schema"*>srlp</property>

<property name=*"show\_sql"*>true</property>

<property name=*"hibernate.c3p0.min\_size"*>5</property>

<property name=*"hibernate.c3p0.max\_size"*>20</property>

<property name=*"hibernate.c3p0.timeout"*>300</property>

<property name=*"hibernate.c3p0.max\_statements"*>50</property>

<property name=*"hibernate.c3p0.idle\_test\_period"*>3000</property>

*<mapping class=" in.spring4buddies.application.basic.Student "*></mapping>

</session-factory>

</hibernate-configuration>

1. hibernate.c3p0.min\_size – Minimum number of JDBC connections in the pool. Hibernate default: 1
2. hibernate.c3p0.max\_size – Maximum number of JDBC connections in the pool. Hibernate default: 100
3. hibernate.c3p0.timeout – When an idle connection is removed from the pool (in second). Hibernate default: 0, never expire.
4. hibernate.c3p0.max\_statements – Number of prepared statements will be cached. Increase performance. Hibernate default: 0 , caching is disable.
5. hibernate.c3p0.idle\_test\_period – idle time in seconds before a connection is automatically validated. Hibernate default: 0

**How to configure DBCP connection pool in Hibernate**

**Note**

Due to bugs in the old DBCP code, Hibernate is no longer maintain DBCP-based connection provider, read this Hibernate thread.

Now, [Apache DBCP](http://commons.apache.org/dbcp/) is back to active development, and many bugs are fixed and it’s more stable now. Even Hibernate doesn’t come with connection provider like [C3P0](http://docs.jboss.org/hibernate/core/3.6/javadocs/org/hibernate/connection/C3P0ConnectionProvider.html) and [Proxool](http://docs.jboss.org/hibernate/core/3.6/javadocs/org/hibernate/connection/ProxoolConnectionProvider.html), but you still can configure it easily.

In this tutorial, we show you how to integrate Apache DBCP connection pool with Hibernate framework.

**1. Get DBCP jars**

To integrate DBCP with Hibernate, you need **commons-dbcp.jar** and **commons-pool-1.5.4.jar**.

<dependency>

<groupId>commons-dbcp</groupId>

<artifactId>commons-dbcp</artifactId>

<version>1.4</version>

</dependency>

**2. DBCPConnectionProvider**

To integrate DBCP with Hibernate, you need to create a “**DBCPConnectionProvider**” class, refer to this [article](http://wiki.apache.org/commons/DBCP/Hibernate).

File : DBCPConnectionProvider.java

**import** java.io.PrintWriter;

**import** java.io.StringWriter;

**import** java.sql.Connection;

**import** java.sql.SQLException;

**import** java.util.Iterator;

**import** java.util.Map;

**import** java.util.Properties;

**import** org.apache.commons.dbcp.BasicDataSource;

**import** org.apache.commons.dbcp.BasicDataSourceFactory;

**import** org.hibernate.HibernateException;

**import** org.hibernate.cfg.Environment;

**import** org.hibernate.connection.ConnectionProvider;

**import** org.hibernate.connection.ConnectionProviderFactory;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**public** **class** DBCPConnectionProvider **implements** ConnectionProvider {

**private** **static** **final** Logger ***log*** = LoggerFactory

.*getLogger*(DBCPConnectionProvider.**class**);

**private** **static** **final** String ***PREFIX*** = "hibernate.dbcp.";

**private** BasicDataSource ds;

// Old Environment property for backward-compatibility (property removed in

// Hibernate3)

**private** **static** **final** String ***DBCP\_PS\_MAXACTIVE*** = "hibernate.dbcp.ps.maxActive";

// Property doesn't exists in Hibernate2

**private** **static** **final** String ***AUTOCOMMIT*** = "hibernate.connection.autocommit";

**public** **void** configure(Properties props) **throws** HibernateException {

**try** {

***log***.debug("Configure DBCPConnectionProvider");

// DBCP properties used to create the BasicDataSource

Properties dbcpProperties = **new** Properties();

// DriverClass & url

String jdbcDriverClass = props.getProperty(Environment.***DRIVER***);

String jdbcUrl = props.getProperty(Environment.***URL***);

dbcpProperties.put("driverClassName", jdbcDriverClass);

dbcpProperties.put("url", jdbcUrl);

// Username / password

String username = props.getProperty(Environment.***USER***);

String password = props.getProperty(Environment.***PASS***);

dbcpProperties.put("username", username);

dbcpProperties.put("password", password);

// Isolation level

String isolationLevel = props.getProperty(Environment.***ISOLATION***);

**if** ((isolationLevel != **null**)

&& (isolationLevel.trim().length() > 0)) {

dbcpProperties.put("defaultTransactionIsolation",

isolationLevel);

}

// Turn off autocommit (unless autocommit property is set)

String autocommit = props.getProperty(***AUTOCOMMIT***);

**if** ((autocommit != **null**) && (autocommit.trim().length() > 0)) {

dbcpProperties.put("defaultAutoCommit", autocommit);

} **else** {

dbcpProperties.put("defaultAutoCommit",

String.*valueOf*(Boolean.***FALSE***));

}

// Pool size

String poolSize = props.getProperty(Environment.***POOL\_SIZE***);

**if** ((poolSize != **null**) && (poolSize.trim().length() > 0)

&& (Integer.*parseInt*(poolSize) > 0)) {

dbcpProperties.put("maxActive", poolSize);

}

// Copy all "driver" properties into "connectionProperties"

Properties driverProps = ConnectionProviderFactory

.getConnectionProperties(props);

**if** (driverProps.size() > 0) {

StringBuffer connectionProperties = **new** StringBuffer();

**for** (Iterator iter = driverProps.entrySet().iterator(); iter

.hasNext();) {

Map.Entry entry = (Map.Entry) iter.next();

String key = (String) entry.getKey();

String value = (String) entry.getValue();

connectionProperties.append(key).append('=').append(value);

**if** (iter.hasNext()) {

connectionProperties.append(';');

}

}

dbcpProperties.put("connectionProperties",

connectionProperties.toString());

}

// Copy all DBCP properties removing the prefix

**for** (Iterator iter = props.entrySet().iterator(); iter.hasNext();) {

Map.Entry entry = (Map.Entry) iter.next();

String key = (String) entry.getKey();

**if** (key.startsWith(***PREFIX***)) {

String property = key.substring(***PREFIX***.length());

String value = (String) entry.getValue();

dbcpProperties.put(property, value);

}

}

// Backward-compatibility

**if** (props.getProperty(***DBCP\_PS\_MAXACTIVE***) != **null**) {

dbcpProperties.put("poolPreparedStatements",

String.*valueOf*(Boolean.***TRUE***));

dbcpProperties.put("maxOpenPreparedStatements",

props.getProperty(***DBCP\_PS\_MAXACTIVE***));

}

// Some debug info

**if** (***log***.isDebugEnabled()) {

StringWriter sw = **new** StringWriter();

dbcpProperties.list(**new** PrintWriter(sw, **true**));

***log***.debug(sw.toString());

}

// Let the factory create the pool

ds = (BasicDataSource) BasicDataSourceFactory

.createDataSource(dbcpProperties);

// The BasicDataSource has lazy initialization

// borrowing a connection will start the DataSource

// and make sure it is configured correctly.

Connection conn = ds.getConnection();

conn.close();

// Log pool statistics before continuing.

logStatistics();

} **catch** (Exception e) {

String message = "Could not create a DBCP pool";

***log***.error(message, e);

**if** (ds != **null**) {

**try** {

ds.close();

} **catch** (Exception e2) {

// ignore

}

ds = **null**;

}

**throw** **new** HibernateException(message, e);

}

***log***.debug("Configure DBCPConnectionProvider complete");

}

**public** Connection getConnection() **throws** SQLException {

Connection conn = **null**;

**try** {

conn = ds.getConnection();

} **finally** {

logStatistics();

}

**return** conn;

}

**public** **void** closeConnection(Connection conn) **throws** SQLException {

**try** {

conn.close();

} **finally** {

logStatistics();

}

}

**public** **void** close() **throws** HibernateException {

***log***.debug("Close DBCPConnectionProvider");

logStatistics();

**try** {

**if** (ds != **null**) {

ds.close();

ds = **null**;

} **else** {

***log***.warn("Cannot close DBCP pool (not initialized)");

}

} **catch** (Exception e) {

**throw** **new** HibernateException("Could not close DBCP pool", e);

}

***log***.debug("Close DBCPConnectionProvider complete");

}

**protected** **void** logStatistics() {

**if** (***log***.isInfoEnabled()) {

***log***.info("active: " + ds.getNumActive() + " (max: "

+ ds.getMaxActive() + ") " + "idle: " + ds.getNumIdle()

+ "(max: " + ds.getMaxIdle() + ")");

}

}

**public** **boolean** supportsAggressiveRelease() {

**return** **false**;

}

}

**3. Configure DBCP in hibernate.cfg.xml**

Now, link your “**DBCPConnectionProvider**” and define the DBCP properties in “**hibernate.cfg.xml**“, for example :

File : hibernate.cfg.xml

<hibernate-configuration>

<session-factory>

<property name=*"hibernate.connection.driver\_class"*>oracle.jdbc.driver.OracleDriver</property>

<property name=*"hibernate.connection.url"*>jdbc:oracle:thin:@localhost:1521:orcl</property>

<property name=*"hibernate.connection.username"*>srlp</property>

<property name=*"hibernate.connection.password"*>srlp</property>

<property name=*"hibernate.dialect"*>org.hibernate.dialect.Oracle10gDialect</property>

<property name=*"hibernate.default\_schema"*>orcl</property>

<property name=*"show\_sql"*>true</property>

<property name=*"hibernate.connection.provider\_class"*>

in.spring4buddies.application.connectionpool.DBCPConnectionProvider

</property>

<property name=*"hibernate.dbcp.initialSize"*>8</property>

<property name=*"hibernate.dbcp.maxActive"*>20</property>

<property name=*"hibernate.dbcp.maxIdle"*>20</property>

<property name=*"hibernate.dbcp.minIdle"*>0</property>

<mapping class=*"com.mkyong.user.DBUser"*></mapping>

</session-factory>

</hibernate-configuration>