Hibernate has recommended to use one of the following two third party connection pool. When hibernate is used in standalone/desktop project,select one of two third party connection pool.

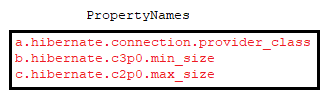
**a.C3p0 connection pool.**

**b. proxool connection pool.**

**c3p0 Connection pool**

C3p0 is open source third party connection pool which is maintained by sourceforege.net.

If we configure C3p0 connection related properties in hibernate.cfg.xml file the internally hibernate create C3p0 connection pool and obtains connection object from that pool.



To work with c3p0 connection pool mechanism, do following activities:

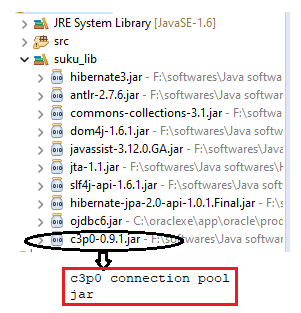
1. Add c3p0-version.jar file to classpath .

This jar present in following address of Hibernate jar.



1. Configure the above properties in hibernate mapping file.

Example:- This application demonstrates c3p0 third party connection pool mechanism.



**Domain/pojo class: Student.java**

package com.hib.domain;

import java.io.Serializable;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

import org.hibernate.annotations.Filter;

import org.hibernate.annotations.FilterDef;

import org.hibernate.annotations.ParamDef;

@Entity

@Table(name="stu")

@FilterDef(name="con",parameters=@ParamDef(name="sal",type="int"))

@Filter(name="con" , condition="salary > :sal")

public class Student implements Serializable{

@Id

@Column(name="sid")

private int sid;

@Column(name="sname")

private String sname;

@Column(name="salary")

private float marks;

public int getSid() {

return sid;

}

public void setSid(int sid) {

this.sid = sid;

}

public String getSname() {

return sname;

}

public void setSname(String sname) {

this.sname = sname;

}

public float getMarks() {

return marks;

}

public void setMarks(float marks) {

this.marks = marks;

}

}

**HibernateConfigurationFile:hibernate.cfg.xml**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

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

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

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

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

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

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

**<property name=*"hibernate.connection.provider\_class"*>org.hibernate.connection.C3P0ConnectionProvider</property>**

**<property name=*"hibernate.c3p0.min\_size"*>2</property>**

**<property name=*"hibernate.c3p0.max\_size"*>10</property>**

<mapping class=*"com.hib.domain.Student"*/>

</session-factory>

</hibernate-configuration>

**Client Application:Test.java**

package com.hib.domain;

import java.sql.SQLException;

import java.util.List;

import java.util.Scanner;

import org.hibernate.Filter;

import org.hibernate.Query;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

import org.hibernate.classic.Session;

public class Test {

public static void main(String[] args)throws SQLException {

try {

Configuration cfg=new Configuration();

cfg.configure("com/hib/domain/hibernate.cfg.xml");

SessionFactory sf=cfg.buildSessionFactory();

Session s=sf.openSession();

Query q=s.createQuery("from Student");

System.out.println("sid\tsname\tmarks");

System.out.println("=======================");

List<Student>li=q.list();

for(Student y:li) {

System.out.println(y.getSid()+"\t"+y.getSname()+"\t"+y.getMarks());

}

}

catch(Exception e) {

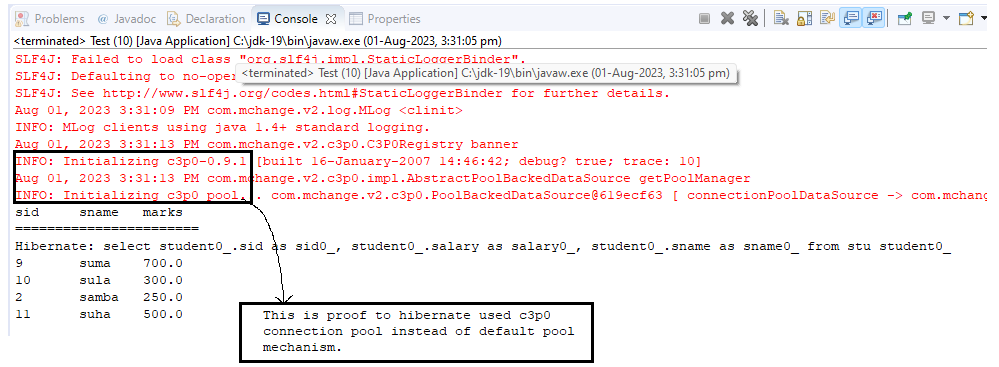
e.printStackTrace();

}

}

}

**Output:**



**Proxool Connection Pool**

**Step1.**  Create separate xml file with any name.xml and we need to configure the following information in that xml file. This file is called **proxol configuration file.**

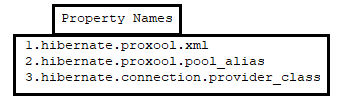
a.pool alias name

b.Connection properties.

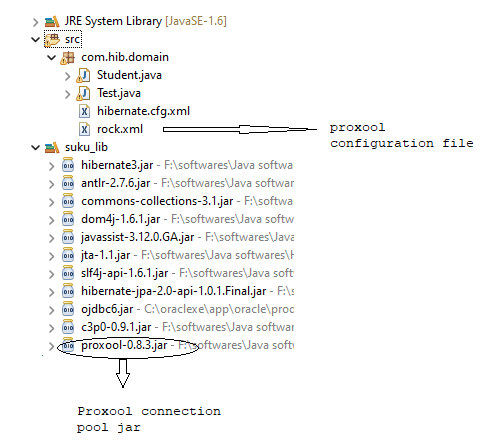
c. maximum pool size.

Note :- in proxol, minimum pool size is always 1. We can’t change it.

**Step2:-** in hibernate.cfg.xml file, we need to configure the proxool related porperties.



Example: Application demonstrates the Proxool connection pool mechnism.



**Domain/pojo class: Student.java**

package com.hib.domain;

import java.io.Serializable;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

import org.hibernate.annotations.Filter;

import org.hibernate.annotations.FilterDef;

import org.hibernate.annotations.ParamDef;

@Entity

@Table(name="stu")

@FilterDef(name="con",parameters=@ParamDef(name="sal",type="int"))

@Filter(name="con" , condition="salary > :sal")

public class Student implements Serializable{

@Id

@Column(name="sid")

private int sid;

@Column(name="sname")

private String sname;

@Column(name="salary")

private float marks;

public int getSid() {

return sid;

}

public void setSid(int sid) {

this.sid = sid;

}

public String getSname() {

return sname;

}

public void setSname(String sname) {

this.sname = sname;

}

public float getMarks() {

return marks;

}

public void setMarks(float marks) {

this.marks = marks;

}

}

**HibernateConfigurationFile:hibernate.cfg.xml**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

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

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

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

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

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

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

**<property name=*"hibernate.proxool.pool\_alias"*>proxool</property>**

**<property name=*"hibernate.proxool.xml"*>rock.xml</property>**

<mapping class=*"com.hib.domain.Student"*/>

</session-factory>

</hibernate-configuration>

**Proxool configuration file:rock.xml**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<proxool-config>

<proxool>

<alias>proxool</alias>

<driver-url>jdbc:oracle:thin:@localhost:1521:xe</driver-url>

<driver-class>oracle.jdbc.OracleDriver</driver-class>

<driver-properties>

<property name=*"user"* value=*"system"*></property>

<property name=*"password"* value=*"tiger"*></property>

</driver-properties>

<minimum-connection-count>5</minimum-connection-count>

<maximum-connection-count>10</maximum-connection-count>

</proxool>

</proxool-config>

**Client Application:Test.java**

package com.hib.domain;

import java.sql.SQLException;

import java.util.List;

import java.util.Scanner;

import org.hibernate.Filter;

import org.hibernate.Query;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

import org.hibernate.classic.Session;

public class Test {

public static void main(String[] args)throws SQLException {

try {

Configuration cfg=new Configuration();

cfg.configure("com/hib/domain/hibernate.cfg.xml");

SessionFactory sf=cfg.buildSessionFactory();

Session s=sf.openSession();

Query q=s.createQuery("from Student");

System.out.println("sid\tsname\tmarks");

System.out.println("=======================");

List<Student>li=q.list();

for(Student y:li) {

System.out.println(y.getSid()+"\t"+y.getSname()+"\t"+y.getMarks());

}

}

catch(Exception e) {

e.printStackTrace();

}

}

}