

## Spring使用proxool连接池 管理数据源

### 一、Proxool连接池简介及其配置属性概述

Proxool是一种Java数据库连接池技术。是sourceforge下的一个开源项目,这个项目提供一个健壮、易用的连接池，最为关键的是这个连接池提供监控的功能，方便易用，便于发现连接泄漏的情况。

目前是和DBCP以及C3P0一起，最为常见的三种JDBC连接池技术。

日前，Hibernate官方宣布由于Bug太多不再支持DBCP，而推荐使用 Proxool或C3P0。

#### 下载方式：

要使用Proxool首先要导入Proxool.jar,此jar包可以在下载的Hibernate包中hibernate-release-4.1.7.Final\lib\optional\proxool\中找到

也可以到官网：<http://nchc.sourceforge.net/sourceforge/proxool/proxool-0.9.1-source.zip>下载它的源码，

下载完后解压,把proxool.jar和proxool-cglib.jar放入你要配置的项目的lib目录下。

#### simultaneous-build-throttle:

是指在任一时刻，可以（同时）建立的最大连接数，也就是说，就是已经请求的、但还没可用的新连接数量。因为连接可以用多线程建立，从决定要建立连接到连接 可用是需要一定时间的，所以我们需要一些方式来避免大量的线程想同时建立连接。（我们本应该找一个更聪明的方式来解决这个问题，总有一天我们会找到的）默认值是 10

当我使用140个用户，进行压力测试时，发现偶尔，会有多于10个要求同时建立连接的请求，当请求数量超过限定的数值时，会出现连接失败的情况。

因此结论就是，当数据库并发连接可能会比较高的应用，这个值应该适当的设大一点。

如果并发请求很高，可能出现的bug为

```
Caused by: java.sql.SQLException: We are already in the process of making 11 connections and the number of simultaneous builds has been throttled to 10
```

#### maximum-active-time:

如果一个线程活动时间超过这个数值，线程会被杀死。所以要确保这个数值设置得比最慢的响应时间长。默认是5分钟。守护进程会把连接池中多余的可用线程（未用的、超过这个时间的）杀死，最终保留的连接数量就是minimum-connection-count规定的数量。守护进程会根据house-keeping-sleep-time参数设置的时间隔定时检查。

#### maximum-connection-lifetime:

指一个连接最大的存活时间（毫秒为单位），超过这个时间，连接会被杀死。默认值是4小时。

#### overload-without-refusal-lifetime:

这个参数帮助我们确定连接池的状态，如果在这个时间阈值内（单位为毫秒）拒绝了一个连接，就认为是过载了。默认值是60。

**alias:**数据源的别名

**driver-url:**url连接串,须确定用户名和密码

**driver-class:**驱动名

**username:**用户名(proxool没有使用,但是不能没有)

**password:**密码(proxool没有使用,但是不能没有)

**maximum-new-connections:**没有空闲连接可以分配而在队列中等候的最大请求数,超过这个请求数的用户连接就不会被接受

**test-before-use :**

如果连接池在运行当中,出现网络或者数据库故障而无法连接到数据库,在恢复正常以后,由于连接是在连接池中持久保存的,会出现连接仍然不可用的情况,这时连接池里的连接实际上都是坏连接,怎么让连接池可以自动重连清除这些坏连接呢? 只要配置了test-before-use 参数,即每次取出连接都检查连接是否可用,就可以做到让连接池实现在故障恢复后自动重连接

需要注意一点,对于Mysql数据库还必须在连接参数里加上autoReconnect=true 参数,否则即使打开了test-before-use 参数,仍然不能重连接!

**fatal-sql-exception:**

它是一个逗号分割的信息片段.当一个SQL异常发生时,他的异常信息将与这个信息片段进行比较.如果在片段中存在,那么这个异常将被认为是个致命错误(Fatal SQL Exception ).这种情况下,数据库连接将要被放弃.无论发生什么,这个异常将会被重掷以提供给消费者.用户最好自己配置一个不同的异常来抛出.

**fatal-sql-exception-wrapper-class:**

正如上面所说,你最好配置一个不同的异常来重掷.利用这个属性,用户可以包装SQLException,使他变成另外一个异常.这个异常或者继承 SQLException或者继承字 RuntimeException.proxool自带了2个实现:'org.logicalcobwebs.proxool.FatalSQLException' 和'org.logicalcobwebs.proxool.FatalRuntimeException' .后者更合适.

**house-keeping-sleep-time:**

proxool自动侦察各个连接状态的时间间隔(毫秒),侦察到空闲的连接就马上回收,超时的销毁 默认30秒)

house keeper 保留线程处于睡眠状态的最长时间,house keeper 的职责就是检查各个连接的状态,并判断是否需要销毁或者创建.

**house-keeping-test-sql:**

如果发现了空闲的数据库连接.house keeper 将会用这个语句来测试.这个语句最好非常快的被执行.如果没有定义,测试过程将会被忽略。

一般mysql可用select SYSDATE , Oracle可用 select sysdate from dual 或者 select 1 from dual

**injectable-connection-interface:** 允许proxool实现被代理的connection对象的方法.

**injectable-statement-interface:** 允许proxool实现被代理的Statement 对象方法.

**injectable-prepared-statement-interface:** 允许proxool实现被代理的PreparedStatement 对象方法.

**injectable-callable-statement-interface:** 允许proxool实现被代理的CallableStatement 对象方法.

**jndi-name:** 数据源的名称

**maximum-active-time:** 如果housekeeper 检测到某个线程的活动时间大于这个数值.它将会杀掉这个线程.所以确认一下你的服务器的带宽.然后定一个合适的值.默认是5分钟.

**maximum-connection-count:**

The maximum number of connections to the database. Default is 15.

最大的数据库连接数.默认是15

**minimum-connection-count:** 最小的数据库连接数, 默认是5

**prototype-count:**

连接池中可用的连接数量.如果当前的连接池中的连接少于这个数值.新的连接将被建立(假设没有超过最大可用数).例如.我们有3个活动连接2个可用连接,而我们的prototype-count是4,那么数据库连接池将试图建立另外2个连接.这和 minimum-connection-count不同. minimum-connection-count把活动的连接也计算在内.prototype-count 是spare connections 的数量.

**statistics:** 连接池使用状况统计。 参数“10s,1m,1d”  
**statistics-log-level:** 日志统计跟踪类型。 参数“ERROR”或 “INFO”

**trace:** 如果为true,那么每个被执行的SQL语句将会在执行期被log记录(DEBUG LEVEL).你也可以注册一个ConnectionListener (参看ProxoolFacade)得到这些信息.

**verbose:** 详细信息设置。 参数 bool 值

## 二、Spring中配置Proxool连接池管理数据源方式与步骤

方式一、在Spring的"applicationContext.xml"中的dataSource bean定义



```
<bean id="dataSource"
    class="org.logicalcobwebs.proxool.ProxoolDataSource">
    <property name="driver">
        <value>com.mysql.jdbc.Driver</value>
    </property>
    <property name="driverUrl">
        <value>jdbc:mysql://localhost:3306/dbname?user=yourname&password=yourpass</value>
    </property>
    <property name="user" value="yourname" />
    <property name="password" value="yourpass" />
    <property name="alias" value="Pool_dbname" />
    <property name="houseKeepingSleepTime" value="90000" />
    <property name="prototypeCount" value="0" />
    <property name="maximumConnectionCount" value="50" />
    <property name="minimumConnectionCount" value="2" />
    <property name="simultaneousBuildThrottle" value="50" />
    <property name="maximumConnectionLifetime" value="14400000" />
    <property name="houseKeepingTestSql" value="select CURRENT_DATE" />
</bean>
```




第一种方式需要把用户名和密码写在连接串里面，ProxoolDataSource类提供的user，password属性似乎没有什么用。无论提供什么，它都会以空用户名、密码去连接数据库。这可能是Proxool RC0.93的一个

bug，实在让人恼火，不知道最新的0.9.1有没有fix这个bug。不过配置中的user，password两个属性还必须设置，否则hibernate会报空指针错误

方式二

步骤一：在Spring的"applicationContext.xml"中的dataSource bean定义



```
<bean id="dataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName">
        <value>org.logicalcobwebs.proxool.ProxoolDriver</value>
    </property>
```

```
        <property name="url">
            <value>proxool.Pool_dbname</value>
        </property>
    </bean>
```



步骤二、预先在"web.xml"先配置好Proxool连接池，配置如下：



<!--管理proxool配置文件-->

<servlet>

<servlet-name>proxoolServletConfigurator</servlet-name>

<servlet-class> org.logicalcobwebs.proxool.configuration.ServletConfigurator

</servlet-class>

<init-param>

<param-name>xmlFile</param-name>

<param-value>WEB-INF/proxool.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<!--查看proxool运行情况，也可以不作配置的

<servlet>

<servlet-name>proxooladmin</servlet-name>

<servlet-class>

org.logicalcobwebs.proxool.admin.servlet.AdminServlet

</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>proxooladmin</servlet-name>

<url-pattern>/proxooladmin</url-pattern>

</servlet-mapping>

-->

<servlet>

<servlet-name>context</servlet-name>

<servlet-class>

org.springframework.web.context.ContextLoaderServlet

</servlet-class>

<load-on-startup>2</load-on-startup>

</servlet>



注意：把<load-on-startup>的值设为1，值越小级别就越高，就先被加载初始化。一定要先于applicationContext.xml的加载；第二种方式下Spring的上下文加载如果想使用listener方式(Struts2要求)，

则与连接池有关的Bean全得延迟初始化。因为listener比servlet优先初始化，

如果相关Bean不是lazy-init的话，则启动服务器时会出现Bean找不到连接定义的异常：

Problem org.logicalcobwebs.proxool.ProxoolException: Attempt to refer to a unregistered pool by its alias 'db'

listener方式如下：

```
<listener>
  <listener-class>
    org.springframework.web.context.ContextLoaderListener
  </listener-class>
</listener>
```

步骤三、在/WEB-INF/下添加proxool的配置文件：proxool.xml

proxool的配置文件可以采用xmlFile"proxool.xml"或者propertyFile"proxool.properties"


"proxool.xml"格式如下：



```
<?xml version="1.0" encoding="UTF-8"?>
<proxool-config>
  <proxool>
    <alias>Pool_dbname</alias>
    <driver-url>jdbc:mysql://localhost:3306/dbname</driver-url>
    <driver-class>com.mysql.jdbc.Driver</driver-class>
    <driver-properties>
      <property name="user" value="yourname"/>
      <property name="password" value="yourpass"/>
    </driver-properties>
    <house-keeping-sleep-time>60000</house-keeping-sleep-time>
    <maximum-connection-count>20</maximum-connection-count>
    <minimum-connection-count>2</minimum-connection-count>
    <prototype-count>0</prototype-count>
    <simultaneous-build-throttle>20</simultaneous-build-throttle>
    <house-keeping-test-sql>select CURRENT_DATE</house-keeping-test-sql>
    <statistics>15s,10m,1d</statistics>
    <statistics-log-level>INFO</statistics-log-level>
  </proxool>
  <proxool>
    <!--可以配置多个库-->
  </proxool>
</proxool-config>
```



"proxool.properties"格式如下：



```
jdbc-0.proxool.alias=Pool_dbname
jdbc-0.proxool.driver-url=jdbc:mysql://localhost:3306/dbname
jdbc-0.proxool.driver-class=com.mysql.jdbc.Driver
jdbc-0.user=yourname
```

```
jdbc-0.password=yourpass
jdbc-0.proxool.house-keeping-sleep-time=60000
jdbc-0.proxool.house-keeping-test-sql=select  CURRENT_DATE
jdbc-0.proxool.maximum-connection-count=10
jdbc-0.proxool.minimum-connection-count=3
jdbc-0.proxool.maximum-connection-lifetime=18000000
jdbc-0.proxool.prototype-count=3
jdbc-0.proxool.simultaneous-build-throttle=10
jdbc-0.proxool.recently-started-threshold=60000
jdbc-0.proxool.overload-without-refusal-lifetime=50000
jdbc-0.proxool.maximum-active-time=60000
jdbc-0.proxool.verbose=true
jdbc-0.proxool.trace=true
jdbc-0.proxool.fatal-sql-exception=Fatal  error

jdbc-2.proxool.alias=Pool_dbname2
```

.....

<!--可以配置多个库-->

至此，已完成所有配置。



个人比较倾向于第二种配置方式，

- 1：可以避免在Spring的"applicationContext.xml"中写一大堆参数，尤其是避免了driverUrl中带用户名密码（这会显示在proxool包中带的org.logicalcobwebs.proxool.admin.servlet.AdminServlet输出的页面中）
- 2：proxool连接池可以在tomcat启动时就初始化好，可以提高第一次访问web时的连接速度

### 三、更详细的配置示例

1、更详细的proxool.xml的配置属性说明:



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
Properties for Proxool Configurator testing. Defines the same parameters as
TestHelper.buildCompleteAlternativeProperties()
-->
<something-else-entirely xmlns="http://sumthin.else.entirely" xmlns:proxool="The latest version is available at http://proxool.sourceforge.net/xml-namespaces">
    <proxool:proxool>
        <proxool:alias>xml-test-ns</proxool:alias>
        <proxool:driver-url>jdbc:hsqldb:db/test</proxool:driver-url>
        <proxool:driver-class>org.hsqldb.jdbcDriver</proxool:driver-class>
        <proxool:driver-properties>
            <proxool:property name="user" value="sa"/>
            <proxool:property name="password" value=""/>
        </proxool:driver-properties>
        <proxool:house-keeping-sleep-time>40000</proxool:house-keeping-sleep-time>
        <proxool:house-keeping-test-sql>select  CURRENT_DATE</proxool:house-keeping-test-sql>
```

```
<proxool:maximum-connection-count>10</proxool:maximum-connection-count>
<proxool:minimum-connection-count>3</proxool:minimum-connection-count>
<proxool:maximum-connection-lifetime>18000000</proxool:maximum-connection-lifetime>
<!-- 5 hours -->
<proxool:simultaneous-build-throttle>5</proxool:simultaneous-build-throttle>
<proxool:recently-started-threshold>40000</proxool:recently-started-threshold>
<proxool:overload-without-refusal-lifetime>50000</proxool:overload-without-refusal-lifetime>
<proxool:maximum-active-time>60000</proxool:maximum-active-time>
<proxool:verbose>true</proxool:verbose>
<proxool:trace>true</proxool:trace>
<proxool:fatal-sql-exception>Fatal error</proxool:fatal-sql-exception>
<proxool:prototype-count>2</proxool:prototype-count>
</proxool:proxool>
<nothing-to-do-with-proxool>
  <proxool:proxool>
    <proxool:alias>xml-test-ns-2</proxool:alias>
    <proxool:driver-url>jdbc:hsqldb:db/test</proxool:driver-url>
    <proxool:driver-class>org.hsqldb.jdbcDriver</proxool:driver-class>
    <proxool:driver-properties>
      <proxool:property name="user" value="sa"/>
      <proxool:property name="password" value=""/>
    </proxool:driver-properties>
    <proxool:house-keeping-sleep-time>40000</proxool:house-keeping-sleep-time>
    <proxool:house-keeping-test-sql>select CURRENT_DATE</proxool:house-keeping-test-sql>
    <proxool:maximum-connection-count>10</proxool:maximum-connection-count>
    <proxool:minimum-connection-count>3</proxool:minimum-connection-count>
    <proxool:maximum-connection-lifetime>18000000</proxool:maximum-connection-lifetime>
  <!-- 5 hours -->
  <proxool:simultaneous-build-throttle>5</proxool:simultaneous-build-throttle>
  <proxool:recently-started-threshold>40000</proxool:recently-started-threshold>
  <proxool:overload-without-refusal-lifetime>50000</proxool:overload-without-refusal-lifetime>
  <proxool:maximum-active-time>60000</proxool:maximum-active-time>
  <proxool:verbose>true</proxool:verbose>
  <proxool:trace>true</proxool:trace>
  <proxool:fatal-sql-exception>Fatal error</proxool:fatal-sql-exception>
  <proxool:prototype-count>2</proxool:prototype-count>
</proxool:proxool>
</nothing-to-do-with-proxool>
</something-else-entirely>
```



属性列表说明:

**`fatal-sql-exception`**: 它是一个逗号分割的信息片段. 当一个SQL异常发生时, 他的异常信息将与这个信息片段进行比较. 如果在片段中存在, 那么这个异常将被认为是个致命错误(Fatal SQL Exception ).

这种情况下, 数据库连接将要被放弃. 无论发生什么, 这个异常将会被重掷以提供给消费者. 用户最好自己配置一个不同的异常来抛出.

**`fatal-sql-exception-wrapper-class`**: 正如上面所说, 你最好配置一个不同的异常来重掷. 利用这个属性, 用户可以包装SQLException, 使他变成另外一个异常. 这个异常或者继承SQLException或者继承字RuntimeException. proxool自带了2个实现: 'org.logicalcobwebs.proxool.FatalSQLException' 和 'org.logicalcobwebs.proxool.FatalRuntimeException' .

后者更合适.

**house-keeping-sleep-time:** house keeper 保留线程处于睡眠状态的最长时间,house keeper 的职责就是检查各个连接的状态,并判断是否需要销毁或者创建.

**house-keeping-test-sql:** 如果发现了空闲的数据库连接.house keeper 将会用这个语句来测试.这个语句最好非常快的被执行.如果没有定义,测试过程将会被忽略。

**injectable-connection-interface:** 允许proxool实现被代理的connection对象的方法.

**injectable-statement-interface:** 允许proxool实现被代理的Statement 对象方法.

**injectable-prepared-statement-interface:** 允许proxool实现被代理的PreparedStatement 对象方法.

**injectable-callable-statement-interface:** 允许proxool实现被代理的CallableStatement 对象方法.

**jmx:** 略

**jmx-agent-id:** 略

**jndi-name:** 数据源的名称

**maximum-active-time:** 如果housekeeper 检测到某个线程的活动时间大于这个数值.它将会杀掉这个线程.所以确认一下你的服务器的带宽.然后定一个合适的值.默认是5分钟.

**maximum-connection-count:** 最大的数据库连接数.

**maximum-connection-lifetime:** 一个线程的最大寿命.

**minimum-connection-count:** 最小的数据库连接数

**overload-without-refusal-lifetime:** 略

**prototype-count:** 连接池中可用的连接数量.如果当前的连接池中的连接少于这个数值.新的连接将被建立(假设没有超过最大可用数).例如.我们有3个活动连接2个可用连接,

而我们的prototype-count是4,那么数据库连接池将试图建立另外2个连接.这和 minimum-connection-count不同. minimum-connection-count把活动的连接也计算在内.prototype-count 是spare connections 的数量.

**recently-started-threshold:** 略

**simultaneous-build-throttle:** 略

**statistics:** 连接池使用状况统计。 参数“10s,1m,1d”

**statistics-log-level:** 日志统计跟踪类型。 参数“ERROR”或 “INFO”


**test-before-use:** 略

**test-after-use:** 略

**trace:** 如果为true,那么每个被执行的SQL语句将会在执行期被log记录(DEBUG LEVEL).你也可以注册一个ConnectionListener (参看ProxoolFacade)得到这些信息.

**verbose:** 详细信息设置。 参数 bool 值

2.更详细的在web.xml中配置管理proxool的servlet



```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
```



```
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
<servlet>
  <servlet-name>ServletConfigurator</servlet-name>
  <servlet-class>
    org.logicalcobwebs.proxool.configuration.ServletConfigurator
  </servlet-class>
  <init-param>
    <param-name>xmlFile</param-name>
    <param-value>WEB-INF/proxool.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet>
  <servlet-name>Admin</servlet-name>
  <servlet-class>
    org.logicalcobwebs.proxool.admin.servlet.AdminServlet
  </servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Admin</servlet-name>
  <url-pattern>/admin</url-pattern>
</servlet-mapping>
<!-- 配置受保护域，只有Tomcat管理员才能察看连接池的信息 -->
<security-constraint>
  <web-resource-collection>
    <web-resource-name>proxool</web-resource-name>
    <url-pattern>/admin</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>manager</role-name>
  </auth-constraint>
</security-constraint>
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>proxool manager Application</realm-name>
</login-config>
<security-role>
  <description>The role that is required to log in to the Manager Application</description>
  <role-name>manager</role-name>
</security-role>
<error-page>
  <error-code>401</error-code>
  <location>/401.jsp</location>
</error-page>
</web-app>
```



分类: [Web](#)

标签: [Proxool](#)

好文要顶

关注我

收藏该文







山高我为峰

关注 - 3

粉丝 - 29

[+加关注](#)

« [上一篇](#) : [IDEA配置JavaWeb项目Artifacts](#)

» [下一篇](#) : [CSS高度自适应 height:100%;](#)

0

0

posted on 2017-07-26 17:57 [山高我为峰](#) 阅读(381) 评论(0) [编辑](#) [收藏](#)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

**注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，访问[网站首页](#)。**

【推荐】超50万VC++源码：大型工控、组态\仿真、建模CAD源码2018！

【推荐】微信小程序一站式部署 多场景模板定制



开发 **在线Excel** 不再难

用 **SpreadJS** 一分钟搞定

了解详情

**最新IT新闻：**

- 专利显示：苹果智能眼镜比市面头盔更轻巧舒适
- 微信、头条纷纷出新规，即日起账号注册数量大幅下调
- 摩托大军踏上返乡路：支付宝免费送百万份公益保险
- 聚美优品再发声明：提请中消协再次公开检测
- 国家邮政局：快递企业2月22日前全面恢复服务
- » [更多新闻...](#)

 阿里云

告别高昂运维费用 云计算全面助力

40+款核心产品免费半年 再+8000津贴任意采购

立即申请



**最新知识库文章：**

- 领域驱动设计在互联网业务开发中的实践
- 步入云计算
- 以操作系统的角度述说线程与进程
- 软件测试转型之路
- 门内门外看招聘
- » [更多知识库文章...](#)

**历史上的今天:**

2016-07-26 在一个form表单中根据不同按钮实现多个action事件

Copyright ©2018 山高我为峰