数据库连接是一种关键的有限的昂贵的资源，这一点在多用户的网页应用程序中体现得尤为突出。对数据库连接的管理能显著影响到整个应用程序的伸缩性和健壮性，影响到程序的性能指标。数据库连接池正是针对这个问题提出来的。数据库连接池负责分配、管理和释放数据库连接，它允许应用程序重复使用一个现有的数据库连接，而再不是重新建立一个；释放空闲时间超过最大空闲时间的数据库连接来避免因为没有释放数据库连接而引起的数据库连接遗漏。这项技术能明显提高对数据库操作的性能。

Java中常用的数据库连接池有： DBCP 、C3P0、BoneCP、Proxool、DDConnectionBroker、DBPool、XAPool、Primrose、SmartPool、MiniConnectionPoolManager及Druid、hikariCP 等。

Java开源数据连接池: <http://www.open-open.com/20.htm>

**最小连接：应用启动后随即打开的连接数以及后续最小维持的连接数。  
最大连接数 : 应用能够使用的最多的连接数**

**首先说明的一点，为了应用便于移植以及可配置的角度，建议还是使用jndi统一进行连接池的配置。**

**这里简单举个例子（使用spring框架）：  
首先在应用的上下文定义中配置jndi名称，如一个resource.xml文件，里边的写法**

1. <bean id="dataSource" **class**="org.springframework.jndi.JndiObjectFactoryBean”>
2. <property name="jndiName">
3. <value>java:comp/env/jdbc/ghdb</value>
4. </property>
5. <property name="resourceRef">
6. <value>**true**</value>
7. </property>
8. </bean>

全局jndi配置

Tomcat7\conf\context.xml下新增如下代码:

1. **<Resource** auth="Container" name="jdbc/ghdb" driverClassName="com.mysql.jdbc.Driver"
2. logAbandoned="true" maxActive="20" maxIdle="10" maxWait="1000"
3. password="root" removeAbandoned="true" removeAbandonedTimeout="60"
4. type="javax.sql.DataSource"
5. url="jdbc:mysql://127.0.0.1:3306/kdddb?generateSimpleParameterMetadata=true&characterEncoding=utf8&autoReconnect=true"
6. useUnicode="true" username="root" **/>**

HikariCP 是一个高性能的 JDBC 连接池组件。

<bean id=*"dataSource"* class=*"com.zaxxer.hikari.HikariDataSource"* destroy-method=*"close"*>

<!-- Connection Info -->

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

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

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

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

<!-- Connection Pooling Info -->

<property name=*"minimumIdle"* value=*"${jdbc.dbcp.minIdle}"*/>

<property name=*"maximumPoolSize"* value=*"${jdbc.dbcp.maxActive}"*/>

<property name=*"connectionTestQuery"* value=*"${jdbc.hikaricp.connectionTestQuery}"*/>

</bean>

**一 开源数据连接池  
1 dbcp  
dbcp可能是使用最多的开源连接池，原因大概是因为配置方便，而且很多开源和tomcat应用例子都是使用的这个连接池吧。  
这个连接池可以设置最大和最小连接，连接等待时间等，基本功能都有。这个连接池的配置参见附件压缩包中的:dbcp.xml  
使用评价：在具体项目应用中，发现此连接池的持续运行的稳定性还是可以，不过速度稍慢，在大并发量的压力下稳定性有所下降，此外不提供连接池监控**

<bean id=*"dataSource"* class=*"org.apache.commons.dbcp.BasicDataSource"* destroy-method=*"close"*>

<!-- Connection Info -->

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

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

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

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

<!-- Connection Pooling Info -->

<property name="*maxActive*" value="*20*"/>

http://images.csdn.net/syntaxhighlighting/OutliningIndicators/None.gif         <property name="*maxIdle*" value="*10*"/>

http://images.csdn.net/syntaxhighlighting/OutliningIndicators/None.gif         <property name="*initialSize*" value="*1*"/>

http://images.csdn.net/syntaxhighlighting/OutliningIndicators/None.gif         <property name="*maxWait*" value="*1000*"/>

http://images.csdn.net/syntaxhighlighting/OutliningIndicators/None.gif         <property name="*defaultAutoCommit*" value="*true*"/>

http://images.csdn.net/syntaxhighlighting/OutliningIndicators/None.gif         <property name="*removeAbandoned*" value="*true*"/>

http://images.csdn.net/syntaxhighlighting/OutliningIndicators/None.gif         <property name="*removeAbandonedTimeout*" value="*60*"/>

http://images.csdn.net/syntaxhighlighting/OutliningIndicators/None.gif         <property name="*logAbandoned*" value="*true*"/>

</bean>

**2 c3p0  
c3p0是另外一个开源的连接池，在业界也是比较有名的，这个连接池可以设置最大和最小连接，连接等待时间等，基本功能都有。  
这个连接池的配置参见附件压缩包中的:c3p0.xml。  
使用评价：在具体项目应用中，发现此连接池的持续运行的稳定性相当不错，在大并发量的压力下稳定性也有一定保证，  
          此外不提供连接池监控。**

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

<!-- Connection Info -->

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

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

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

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

</bean>

**3 proxool  
proxool这个连接池可能用到的人比较少，但也有一定知名度，这个连接池可以设置最大和最小连接，连接等待时间等，基本功能都有。  
这个连接池的配置参见附件压缩包中的：proxool.xml。  
使用评价：在具体项目应用中，发现此连接池的持续运行的稳定性有一定问题，有一个需要长时间跑批的任务场景任务，同样的代码  
在另外2个开源连接池中成功结束，但在proxool中出现异常退出。  
但是proxool有一个优势--连接池监控，这是个很诱人的东西，大概的配置方式就是在web.xml中添加如下定义：  
    <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>    
并在应用启动后访问：http://localhost:8080/myapp/admin这个url即可监控  
不过proxool本身的包在监测使用中会有编码问题，附件中有一个  
解决此问题的包，参见附件压缩包中的：proxool-0.9.0RC3.jar。另外需要jdk1.5以上的环境。**

<bean id=*"dataSource"* class=*"org.logicalcobwebs.proxool.ProxoolDataSource"* destroy-method=*"close"*>

<!-- Connection Info -->

<property name=*"driver"* value=*"${jdbc.driverClassName}"* />

<property name=*"driverUrl"* value=*"${jdbc.url}"* />

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

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

</bean>

4. BoneCP 是一个开源的快速的 JDBC 连接池。BoneCP很小，只有四十几K（运行时需要log4j和Google Collections的支持，这二者加起来就不小了），而相比之下 C3P0 要六百多K。另外个人觉得 BoneCP 有个缺点是，JDBC驱动的加载是在连接池之外的，这样在一些应用服务器的配置上就不够灵活。当然，体积小并不是 BoneCP 优秀的原因，BoneCP 到底有什么突出的地方呢，请看看性能测试报告。（主页：http://jolbox.com/）

**总结：  
综上所述，这几种开源连接池各有优劣，推荐使用c3p0，经检验这种连接池性能稳定，承压能力强。而proxool尽管有明显的性能问题，  
但由于它具备监控功能，因此建议在开发测试时使用，有助于确定是否有连接没有被关掉，可以排除一些代码的性能问题。  
  
二 商业中间件连接池  
上面列举了几种开源的连接池，其实可以肯定的说，如果条件允许使用weblogic和websphere等中间件，那么不要犹豫，一定要  
使用这些商业级别的中间件所自带的数据库连接池，他们的性能以及调配和开源的不在一个量级，举个例子，曾经有一个项目，数据量比较大，同样的代码应用，在3种开源的连接池里都多少出现过系统异常，而weblogic和websphere的连接池则正常运行，当然后来发现代码有一定瑕疵，但也侧面说明了商业连接池的强大。  
  
1 weblogic的连接池  
weblogic 8 是一个让人使用起来很轻松方便的应用服务器软件，但是到了9简直就是折磨，不知道是bea是怎么想的，oracle收购了bea以后出了10，比9强不少，但是最喜欢的还是8。。。  
题外话不说了，就以8.1版本介绍一下他的数据库连接池（其实10的配置也差不多）  
首先是连接池的基本设置，这个不讲了，网上有很多教程。然后进入Data Sources菜单配置数据源里边的JNDI Name，要和之前在应用配置中的一致：jdbc/myapp。  
然后是连接池一些具体项目的配置，包括设置最小（Initial Capacity），最大（ Maximum Capacity）连接，以及  
每次连接增加时需要一次性增加多少连接（Capacity Increment）。Allow Shrinking（是否把不用的连接退还数据库以保持最小连接数--这个就可以参见之前的连接池阐述的例子进行理解了）。  
另外还有几个有意思的选项Test Reserved Connections：对取得的连接进行测试，Test Released Connections：对释放的连接进行测试。有人会问了，这个有什么用啊？  
不知道大家在项目中有没有遇到java报连接失效的异常，反正我碰到过，只有在系统压力大的时候才出现。而有了这个选项就不用担心这个问题了--因为连接池已经帮你测试了，一旦检查到连接是无效的他会废弃掉还给数据库，只给你有效的。不过这个连接失效的异常其实多半是应用的不严谨造成的，我们更因该关心应代码的问题--但起码weblogic想到帮你弥补一下，是不是很贴心：）  
另外一个重要功能当然是连接池监控：monitor选项卡里可以看到使用情况，有人又要问了，没有什么指标啊，别忘了custom view这个功能链接哦：）  
有以下指标：当前连接数、曾经达到的峰值、可以使用的连接数、等待的连接数、从数据库打开的连接数、曾经关闭的连接数。。。其中前3项是我最关注的  
  
使用评价：在具体项目应用中，此连接池的持续运行的稳定性很强，在大并发量的压力下性能也相当优秀，另外在一些异常情况下连接池里的连接也能够及时释放。  
          连接池监控一目了然，及时到位。          
  
2 websphere的连接池  
还是先来段题外话：记得有人说过，websphere只有版本6以后才算是websphere，个人很赞同。websphere 5以及以前的版本。。。还是忘了吧。  
其实websphere的连接池秉承ibm一贯的风格：功能强大，使用复杂：）  
进入控制台使用“JDBC提供程序”功能菜单进行连接池的基本配置，一路下来，不同的数据库配置方式不尽相同，最奇怪的是还要单独手工加上user和password参数，如果没有  
资料指导的话还真是摸不着头脑。这些基本设置还是网上找吧很多的。连接池设置完还需要设置数据源，jndi名字一样与之前的对应：jdbc/myapp  
高级设置包括初始化连接数，最大连接，连接有效性检查，不使用超时。。其实这些都和weblogic中的连接池配置大致一样。  
连接池监控：使用运行监控菜单，里边会有一个监控项目选择，选jdbc监控即可，可恶的是一开始弹出什么服务器操作系统需要安装什么图形化控件，选择是那么就得去找到控件在操作系统（linux）下安装，然后很多的依赖组件都没有。。。搞了半天才发现选择否，监控数据以及图形一样能出来嘛，真是要怒了。  
虽然经过一番波折但是监控的内容还是很强大的，就连接池来说一样包括当前连接数、曾经达到的峰值、可以使用的连接数、从数据库打开的连接数、曾经关闭的连接数。。。其中前3项是我最关注的，比较奇怪的现象是应用刚启动的时候已开启的连接数量竟然没有达到初始定义的连接数量，不清楚websphere是怎么个计算机制。  
另外在压力大的时候可使用的连接数会是负数，当时很奇怪，想想也了然了，那个负数肯定是排队等待的数量了  
  
使用评价：在具体项目应用中，此连接池的持续运行的稳定性相当强，在大并发量的压力下性能也足够优秀，另外在一些异常情况下连接池里的连接能够及时释放，  
          连接池监控配置有些复杂，但是配置好后各项指标一目了然并且有图形显示   
           
总结时刻：  
这两种商业级别的连接池都给我留下深刻印象，功能强大，使用稳定，性能优秀，监控到位。可以说难分伯仲，相对而言weblogic的连接池使用配置和监控配置更简单明了，而websphere的更复杂但选项功能也更多一些。其实这正是对两种应用服务器的使用印象的直接反映，当然总体比较2种应用服务器应该又是另一个话题了，也许在下一期的内容里。。。  
  
比较了多种连接池的优劣，下面这个话题可能和比较本身没有直接关系，但个人认为应该是更有价值的一些经验分享吧，那就是---这么多指标配置，那些最大和最小连接数以及其他一些必要的配置指标，在一个正式的生产项目中到底应该配置什么值呢？  
其实这个值首先还是要根据具体的项目情况、数据规模以及并发数来制定的(尽管像是套话，但是我们研发人员严谨的作风还是必要的:)。具体而言在中型偏小型的项目--给个数值把，用户数300到3000，数据量100万到1亿---中,建议weblogic设置为最大和最小都是200，websphere最小200最大300，前提是2者设置的最小内存要在1G以上，当然如果条件允许内存越大越好，不过32位机内存1.5G的限制是一定的（64位嘛我愿意设个4G内存过来，速度提升的感觉很爽啊）。这个数字出来以后相信会有不少问题要抛过来，我一一谈一下自己的体验和想法吧  
1 为什么是200或300而不是更高？  
回答：  再分配多了效果也不大了，一个是应用服务器维持这个连接数需要内存支持，刚才说了32位的机器只能支持到1.5G，并且维护大量的连接进行分配使用对cpu也是一个不小的负荷，因此不宜太大。  
2 为什么不小一点？  
回答：   如果太小，那么在上述规模项目的并发量以及数据量上来以后会造成排队现象，系统会变慢，数据库连接会经常打开和关闭，性能上有压力，用户体验也不好。  
3 为什么weblogic最小最大都一样，而websphere不一样  
回答：   其实和分配内存的最小最大值的情况一样，一般都推荐2个值应该一致，都放在内存里就好了嘛。但是ibm官方推荐2个值要有区别---官方说法还是要听的  
4 其他开源连接池的分配方案还没说呢？  
回答： 开源的个人认为到100就可以了，再高他也不会太稳定，当然1G的最小内存是一定要给tomcat分的  
5 还有其他的指标吗？  
回答： 当然还有一些，但个人认为剩下的具体情况具体分析了不在这里一一列举了，大家有兴趣可以和我讨论分享一下。**