获取一个连接，系统要在背后做很多消耗资源的事情，大多时候，创建连接的时间比执行sql语句的时间还要长。  
用户每次请求都需要向数据库获得链接，而数据库创建连接通常需要消耗相对较大的资源，创建时间也较长。

使用DBCP必需的两个JAR包

commons-dbcp-1.2.1.jar

commons-pool-1.3.jar

The commons-dbcp2 package relies on code in the commons-pool2 package to provide the underlying object pool mechanisms that it utilizes.

JDK与JDBC版本对应关系

<http://commons.apache.org/proper/commons-dbcp/>

<http://blog.csdn.net/teamlet/article/details/52389665>

|  |  |  |
| --- | --- | --- |
| JDK版本 | JDBC版本 | DBCP版本 |
| 1.1 | 1.0 |  |
| 1.2 | 2.0 |
| 1.3 |
| 1.4 | 3.0 | DBCP 1.3 |
| 1.5 |
| 1.6 | 4.0 | DBCP 1.4 |
| 1.7 | 4.1 | DBCP 2 |
| 1.8 | 4.2 | ？？ |

使用连接池与不用的效率

相同表插入100条数据

|  |  |  |
| --- | --- | --- |
| 测试方法 | | 耗时 |
| 不用连接池 | 每写一条数据前,就新建一个连接 | 5912 |
| 只建一条连接,写入所有数据 | 4551 |
| 使用连接池 | 每写一条数据前,从连接池中获取一个连接 | 6897 |
| 只建一条连接,写入所有数据 | 1179 |

关键属性：

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 默认 | 说明 | 备注 |
| defaultReadOnly | null | **连接池初始化**(调用这些方法的其中之一后getConnection, setLogwriter, setLoginTimeout, getLoginTimeout, getLogWriter)后再设置该属性无效 | 连接池在调用这些方法时才会完成初始化过程getConnection …… |
| initialSize | 0 | 连接池启动时创建的初始化连接数量（默认值为0） |  |
| maxActive | 8 | 同时连接的最大的连接数（默认值为8，调整为20，高峰单机器在20并发左右，自己根据应用场景定） |  |
| maxIdle | 8 | 最大的空闲的连接数，超过的空闲连接将被释放，如果设置为负数表示不限制（默认为8个，maxIdle不能设置太小，因为假如在高负载的情况下，连接的打开时间比关闭的时间快，会引起连接池中idle的个数 上升超过maxIdle，而造成频繁的连接销毁和创建，类似于jvm参数中的Xmx设置) |  |
| minIdle | 0 | 最小的空闲的连接数，低于这个数量会被创建新的连接（默认为0，调整为5，该参数越接近maxIdle，性能越好，因为连接的创建和销毁，都是需要消耗资源的；但是不能太大，因为在机器很空闲的时候，也会创建低于minidle个数的连接，类似于jvm参数中的Xmn设置） |  |
| testOnBorrow | true | 指明在从池中租借对象时是否要进行验证有效，如果对象验证失败，则对象将从池子释放  借出连接时不要测试，否则很影响性能 |  |
| validationQuery |  | 用以检测 |  |
| maxWait | ------ | 最大等待时间，当没有可用连接时，连接池等待连接释放的最大时间，超过该时间限制会抛出异常，如果设置-1表示无限等待（默认为无限，调整为60000ms，避免因线程池不够用，而导致请求被无限制挂起） |  |
| **timeBetweenEvictionRunsMillis** |  | 空闲连接回收器运行频率 |  |
| minEvictableIdleTimeMillis |  | 把空闲时间超过minEvictableIdleTimeMillis毫秒的连接断开, 直到连接池中的连接数到minIdle为止 |  |
| softMinEvictableIdleTimeMillis | -1L | 一个连接多久可以置为空闲连接，注意，如果设置了minEvictableIdleTimeMillis这个参数则无效 |  |
| numTestsPerEvictionRun |  | 每次空闲连接回收器线程(如果有)运行时检查的连接数量 |  |
| removeAbandonedOnMaintenance | false | 如果设置为true，则一个连接在超过removeAbandonedTimeout所设定的时间未使用即被认为是应该被抛弃并应该被移除的,注意，只有当  timeBetweenEvictionRunsMillis 为正值，从而启用池维护时，removeAbandonedOnMaintenance 才能生效。 | [防止数据库连接池泄露](http://wiki.jikexueyuan.com/project/tomcat/jdbc-data-sources.html) |
| removeAbandonedOnBorrow | false | true means abandoned connections may be removed when connections are borrowed from the pool. |
| removeAbandonedTimeout | 300 | 一个被抛弃连接可以被移除的超时时间，单位为秒 |

<http://wiki.jikexueyuan.com/project/tomcat/jdbc-data-sources.html>

<http://www.totcms.com/html/201602-29/20160229114145.htm>

**注:如果开启"removeAbandoned",那么连接在被认为泄露时可能被池回收. 这个机制在(getNumIdle() < 2)and (getNumActive() > getMaxActive() - 3)时被触发. 举例当maxActive=20, 活动连接为18,空闲连接为1时可以触发"removeAbandoned".但是活动连接只有在没有被使用的时间超过"removeAbandonedTimeout"时才被删除,默认300秒.在resultset中游历不被计算为被使用.**

**详细介绍（几种连接池的配置对比）**

[**http://www.mamicode.com/info-detail-1242126.html**](http://www.mamicode.com/info-detail-1242126.html)

**要注意的类**

**BaseGenericObjectPool**

*/\*\*  
 \* Returns the minimum amount of time an object may sit idle in the pool  
 \* before it is eligible for eviction by the idle object evictor (if any -  
 \* see {****@link*** *#setTimeBetweenEvictionRunsMillis(long)}),  
 \* with the extra condition that at least <code>minIdle</code> object  
 \* instances remain in the pool. This setting is overridden by  
 \* {****@link*** *#getMinEvictableIdleTimeMillis} (that is, if  
 \* {****@link*** *#getMinEvictableIdleTimeMillis} is positive, then  
 \* {****@link*** *#getSoftMinEvictableIdleTimeMillis} is ignored).  
 \**

c3p0与dbcp区别：

dbcp没有自动的去回收空闲连接的功能 c3p0有自动回收空闲连接功能

两者主要是对数据连接的处理方式不同，C3P0提供最大空闲时间，DBCP提供最大连接数。

前者当连接超过最大空闲连接时间时，当前连接就会被断掉。DBCP当连接数超过最大连接数时，所有连接都会被断开

我们都知道Java只做规范不做实现，所以同样的，Java制定了数据库连接池的实现接口，这个接口就是DataSource，所有的数据库连接池数据源类都需要实现这个接口。DBCP中实现了DataSource数据源接口的类是BasicDataSource，我们需要通过数据源实现类来获得连接对象，这个类充当一个管理员的角色

C3P0相关配置参数说明

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 默认值 | 说明 | 备注 |
| acquireIncrement | 3 | 连接池在无**空闲连接**可用时一次性创建的新数据库连接数 |  |
| testConnectionOnCheckin | false | 如果为true，则在close的时候测试连接的有效性 |  |
| testConnectionOnCheckout | false | 性能消耗大。如果为true，在每次getConnection的时候都会测试，为了提高性能,尽量不要用 |  |
| preferredTestQuery | null | 测试连接的sql语句 |  |
| maxConnectionAge | 0(秒) | 配置连接的生存时间，超过这个时间的连接将由连接池自动断开丢弃掉。当然正在使用的连接不会马上断开，而是等待它close再断开。配置为0的时候则不会对连接的生存时间进行限制 |  |
| acquireRetryAttempts | 30 | 连接池在获得新连接失败时重试的次数，如果小于等于0则无限重试直至连接获得成功 |  |
| acquireRetryDelay | 1000(ms) | 连接池在获得新连接时的间隔时间 |  |
| breakAfterAcquireFailure | false | 如果为true，则当连接获取失败时自动关闭数据源  (dataSource关闭)，除非重新启动应用程序。所以一般不用 |  |

Druid

<https://code.aliyun.com/alibaba/druid/commit/84ed1699bf702dcae07ec57d2d69eb545ea89a8a>

PoolingCount: 线程池中连接数，闲置的连接数

ConnectCount: 从启动到现在连接过的次数

相关配置：

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 默认值 | 说明 | 备注 |
| driverClassName |  | 如果不配置druid会根据url自动识别dbType，然后选择相应的driverClassName |  |
| initialSize | 0 | 初始化时建立物理连接的个数。初始化发生在显示调用init方法，或者第一次getConnection时 \*参见DBCP中的initialSize属性 |  |
| maxActive | 8 | 参见DBCP中的maxTotal属性 |  |
| minIdle |  | 最小连接池数量 |  |
| maxWait |  | 获取连接时最大等待时间，单位毫秒。配置了maxWait之后，**缺省**启用**公平锁**，并发效率会有所下降，如果需要可以通过配置useUnfairLock属性为true使用非公平锁。 | 参考：  DruidDataSource中的线程：  DestroyTask |
| testOnBorrow | true | 申请连接时执行validationQuery检测连接是否有效，做了这个配置**会降低性能。** |  |
| testOnReturn | false | 归还连接时执行validationQuery检测连接是否有效，做了这个配置**会降低性能** |  |
| testWhileIdle | false | 建议配置为true，不影响性能，并且保证安全性。申请连接的时候检测，如果空闲时间大于  timeBetweenEvictionRunsMillis，执行validationQuery检测连接是否有效，如果开启了testOnBorrow则此属性无效，该属性与  timeBetweenEvictionRunsMillis无关，如果没有设置  timeBetweenEvictionRunsMillis，或者设置了一个非法值则采用默认值 1分钟，在该空闲连接的空闲时间大于1分钟的时候才会执行SQL校验，而testOnBorrow是每次取连接时都会执行SQL校验，相对来说testWhileIdle对性能的影响不大，建议开启，相比DBCP而言…… | 源码参考  DruidDataSource中的  getConnectionDirect方法 |
| validationQuery |  | 用来检测连接是否有效的sql，要求是一个查询语句。如果validationQuery为null，testOnBorrow、testOnReturn、 testWhileIdle都不会其作用。在mysql中通常为select ‘x‘，在oracle中通常为select 1 from dual |  |
| **timeBetweenEvictionRunsMillis** |  | 1) Destroy线程会检测连接的间隔时间  2) testWhileIdle的判断依据 |  |
| minEvictableIdleTimeMillis |  | Destory线程中如果检测到当前连接的最后活跃时间和当前时间的差值大于  minEvictableIdleTimeMillis，则关闭当前连接。 |  |
| removeAbandonedTimeout |  | 指定连接建立多长时间就需要被强制关闭 |  |
| filters |  | 属性类型是字符串，通过别名的方式配置扩展插件，常用的插件有：  1）监控统计用的filter:stat 2）日志用的filter:log4j  3）防御sql注入的filter:wall |  |

lang.js

相关说明可以参考：

com/alibaba/druid/1.1.5/druid-1.1.5.jar!/support/http/resources/js/lang.js

DruidDataSource(重要参考其获取连接的方法)

Keepalive区别

<http://www.cnblogs.com/trust-freedom/p/6992952.html>

|  |  |  |  |
| --- | --- | --- | --- |
|  | **DBCP** | **c3p0** | **Druid** |
| **最小连接数** | minIdle(0) | minPoolSize(3) | minIdle(0) |
| **初始化连接数** | initialSize(0) | initialPoolSize(3) | initialSize(0) |
| **最大连接数** | maxTotal(8) | maxPoolSize(15) | maxActive(8) |
| **最大等待时间** | maxWaitMillis(毫秒) | maxIdleTime(0秒) | maxWait(毫秒) |

Druid 加密处理

https://github.com/alibaba/druid/wiki/使用ConfigFilter

什么是连接池：

为什么要用连接池：

几种常用的连接池：