重要参数说明如下：

|  |  |  |
| --- | --- | --- |
| 参数 | 含义 | 说明 |
| initConn | 初始化连接个数 | 连接池初始化的大小 |
| maxConn | 最大活动连接个数 | 最大活动连接，默认是20个，根据业务情况来决定 |
| minIdle | 保持最小的空闲连接数 | 连接池中最小保持的连接个数，建议为MaxIdle的一半左右 |
| maxIdle | 保持的最大空闲连接数 | 建议接近MaxCon |
| maxWait | 获取连接的最长等待时间，单位是毫秒 | 不宜过大，当连接池满了以后，等待太长只能导致用户页面无法展现，5-30秒合适 |
| validationQueryTimeout | 连接空闲时候执行ＳＱＬ校验，ＳＱＬ查询的超时时间，单位是秒 | 建议为２－５秒的超时 |
| timeBetweenEvictionRunsMillis | 连接空闲时候对空闲连接定期检查，无效的连接被剔除，保证连接的有效性，单位是毫秒 | 建议５－３０秒定期检查 |
| minEvictableIdleTimeMillis | 当空闲连接未使用的时间超过此值以后，空闲连接被关闭，并重新建立空闲连接，此参数防止服务器段超时关闭客户端的ＳＱＬ连接，单位是毫秒 | 建议１０－３０分钟 |
| removeAbandoned | 对于被占用的连接，进行检查，当超过removeAbandonedTimeout指定的时间还未归还的连接，则抛弃，并打印日志，此参数通常用于程序开发测试阶段，防止编程错误导致的数据库连接没有关闭，无法被连接池回收。 | 开发测试节点设置为ＴＲＵＥ，生产环境可以审慎开启，若不开启，当遇到程序漏洞导致连接池满了以后，连接池将无法使用。 |
| removeAbandonedTimeout | removeAbandonedTimeout指定的时间还未归还的连接，则抛弃，单位是秒 | 建议为５分钟左右 |

依赖的第三方库：DBCP连接池、Common Log, org.json

参考代码：

newDataSource = **new** BasicDataSource();

newDataSource.setInitialSize(jo.getInt("initConn"));

newDataSource.setMaxActive(jo.getInt("maxConn"));

newDataSource.setMaxIdle(jo.getInt("maxIdle"));

newDataSource.setMaxWait(jo.getLong("maxWait"));

newDataSource.setPoolPreparedStatements(jo

.getBoolean("poolPreparedStatements"));

newDataSource.setDefaultAutoCommit(jo

.getBoolean("defaultAutoCommit"));

newDataSource.setValidationQuery("SELECT 1");

**int** minIdle = jo.has("minIdle") ? jo.getInt("minIdle")

: newDataSource.getMaxIdle() / 2 + 1;

newDataSource.setMinIdle(minIdle);

newDataSource.setTestOnBorrow(**false**);

newDataSource.setTestWhileIdle(**true**);

// 查询验证的ＳＱＬ的执行超时时间

**int** queryTimeOut = jo.has("validationQueryTimeout") ? jo

.getInt("validationQueryTimeout") : 3;

newDataSource.setValidationQueryTimeout(queryTimeOut);

**int** timeBetweenEvictionRunsMillis = jo

.has("timeBetweenEvictionRunsMillis") ? jo

.getInt("timeBetweenEvictionRunsMillis") : 10000;

newDataSource

.setTimeBetweenEvictionRunsMillis(timeBetweenEvictionRunsMillis);

// 超过多少时间闲置不用的ＩＤＬＥ连接被关闭

**int** minEvictableIdleTimeMillis = jo

.has("minEvictableIdleTimeMillis") ? jo

.getInt("minEvictableIdleTimeMillis") : 1800000;

newDataSource

.setMinEvictableIdleTimeMillis(minEvictableIdleTimeMillis);

// 一次检查的闲置连接个数

**int** numTestsPerEvictionRun = jo.has("numTestsPerEvictionRun") ? jo

.getInt("numTestsPerEvictionRun") : 1;

newDataSource.setNumTestsPerEvictionRun(numTestsPerEvictionRun);

// 检查被占用的连接是否没有被正常关闭

**boolean** removeAbandoned = jo.has("removeAbandoned") ? jo

.getBoolean("removeAbandoned") : **true**;

newDataSource.setRemoveAbandoned(removeAbandoned);

**int** removeAbandonedTimeout = jo.has("removeAbandonedTimeout") ? jo

.getInt("removeAbandonedTimeout") : 300;

newDataSource.setRemoveAbandonedTimeout(removeAbandonedTimeout);

// 打印非法链接的信息

newDataSource.setLogAbandoned(**true**);