连接池

使用JDBC操作数据库,需要建立Connection,使用传统的JDBC操作需要每次创建Connection,创建Connection是一个非常性能和消耗时间的过程,我们需要在提高程序性能,那么就需要减少每次创建创建连接带来的负面影响,解决这个问题我们将利用池子概念,预先创建一些链接放入池子中,如果需要操作数据,不用创建新的Connection,只需要在池子中获取即可,使用完毕放入池子，这样就形成了复用。通过连接池获得的连接，调用连接的close()方法时不会真正关闭连接，而是放入连接池。

连接池常用概念：

**最小连接数minIdle**:是数据库一直保持的数据库连接数，所以如果应用程序对数据库连接的使用量不大，将有大量的数据库资源被浪费。

**初始化连接数initialSize：**连接池启动时创建的初始化数据库连接数量。

**最大连接数maxActive**：是连接池能申请的最大连接数，如果数据库连接请求超过此数，后面的数据库连接请求被加入到等待队列中。

**最大等待时间maxWaitTime：**当没有可用连接时，连接池等待连接被归还的最大时间，超过时间则抛出异常，可设置参数为0或者负数使得无限等待(根据不同连接池配置)。

**最大空闲连接数maxIdle**：超过的空闲连接将被释放

## DBCP连接池



BasicDataSource source = **new** BasicDataSource();

source.setDriverClassName("com.mysql.jdbc.Driver");

source.setUrl("jdbc:mysql://localhost:3306/1808");

source.setUsername("root");

source.setPassword("");

source.setInitialSize(10);

Connection connection=source.getConnection();

## c3p0连接池



c3p0需要设置一个配置文件，c3p0-config.xml，位置是在src下

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

<c3p0-config>

<!-- 默认配置，如果没有指定则使用这个配置 -->

<default-config>

<!-- 基本配置 -->

<property name=*"driverClass"*>com.mysql.jdbc.Driver</property>

<property name=*"jdbcUrl"*>jdbc:mysql://localhost:3306/1808</property>

<property name=*"user"*>root</property>

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

<!--扩展配置-->

<!-- 连接超过30秒报错-->

<property name=*"checkoutTimeout"*>30000</property>

<!-- 每隔30秒检查连接池中所有空闲连接 -->

<property name=*"idleConnectionTestPeriod"*>30</property>

<property name=*"initialPoolSize"*>10</property>

<!—最大空闲时间，30秒未使用丢弃-->

<property name=*"maxIdleTime"*>30</property>

<property name=*"maxPoolSize"*>100</property>

<property name=*"minPoolSize"*>10</property>

<!-- 用以控制数据源内加载的PreparedStatements数量 -->

<property name=*"maxStatements"*>200</property>

</default-config>

<!-- 命名的配置 -->

<named-config name=*"test1"*>

<property name=*"driverClass"*>com.mysql.jdbc.Driver</property>

<property name=*"jdbcUrl"*>jdbc:mysql://localhost:3306/1715b</property>

<property name=*"user"*>root</property>

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

<property name=*"initialPoolSize"*>20</property>

<property name=*"minPoolSize"*>10</property>

<property name=*"maxPoolSize"*>40</property>

<property name=*"maxStatements"*>20</property>

<property name=*"maxStatementsPerConnection"*>5</property>

</named-config>

</c3p0-config>

ComboPooledDataSource dataSource = **new** ComboPooledDataSource();

Connection connection = dataSource.getConnection();

## Druid连接池



DruidDataSource dataSource = **new** DruidDataSource();

dataSource.setDriverClassName("com.mysql.jdbc.Driver");

dataSource.setUrl("jdbc:mysql://localhost:3306/1808?characterEncoding=utf-8");//设置连接的数据库

dataSource.setUsername("root");//设置用户名

dataSource.setPassword("");//设置密码

Connection connection = dataSource.getConnection();

# DbUtils

---是apache下的一个小巧的JDBC轻量级封装的工具包，其最核心的特性是结果集的封装，可以直接将查询出来的结果集封装成JavaBean



QueryRunner用于执行sql语句。query方法进行查询，update方法进行增删改操作。

QueryRunner runner = **new** QueryRunner(dataSource);

runner.update("insert into user(name,password) values(?,?)", "aaa","1111");

执行查询时，最核心的特性是结果集的封装，封装的时候要求结果集中的列名对应实体类中的属性名

直接封装成实体类用BeanHandler

QueryRunner runner = **new** QueryRunner(dataSource);

String sql = "select id,title from News where id = ?";

**int** id = 9;

News query = runner.query(sql, **new** BeanHandler<News>(News.**class**), id);

System.***out***.println(query.getTitle());

封装成实体类列表用BeanListHandler

QueryRunner runner = new QueryRunner(dataSource);

String sql = "select id,title from News";

List<News> query = runner.query(sql, **new** BeanListHandler<News>(News.**class**));

返回一个值用ScalarHandler

QueryRunner runner = **new** QueryRunner(dataSource);

String sql = "select count(id) from News";

**long** query = (**long**) runner.query(sql, **new** ScalarHandler());

System.***out***.println(query);

QueryRunner 进行查询的操作

org.apache.commons.dbutils.handlers

ArrayHandler ：将ResultSet中第一行的数据转化成对象数组

ArrayListHandler将ResultSet中所有的数据转化成List，List中存放的是Object[]

BeanHandler ：将ResultSet中第一行的数据转化成类对象

BeanListHandler ：将ResultSet中所有的数据转化成List，List中存放的是类对象

ColumnListHandler ：将ResultSet中某一列的数据存成List，List中存放的是Object对象

KeyedHandler ：将ResultSet中存成映射，key为某一列对应为Map。Map中存放的是数据

MapHandler ：将ResultSet中第一行的数据存成Map映射

MapListHandler ：将ResultSet中所有的数据存成List。List中存放的是Map

ScalarHandler ：将ResultSet中一条记录的其中某一列的数据存成Object