

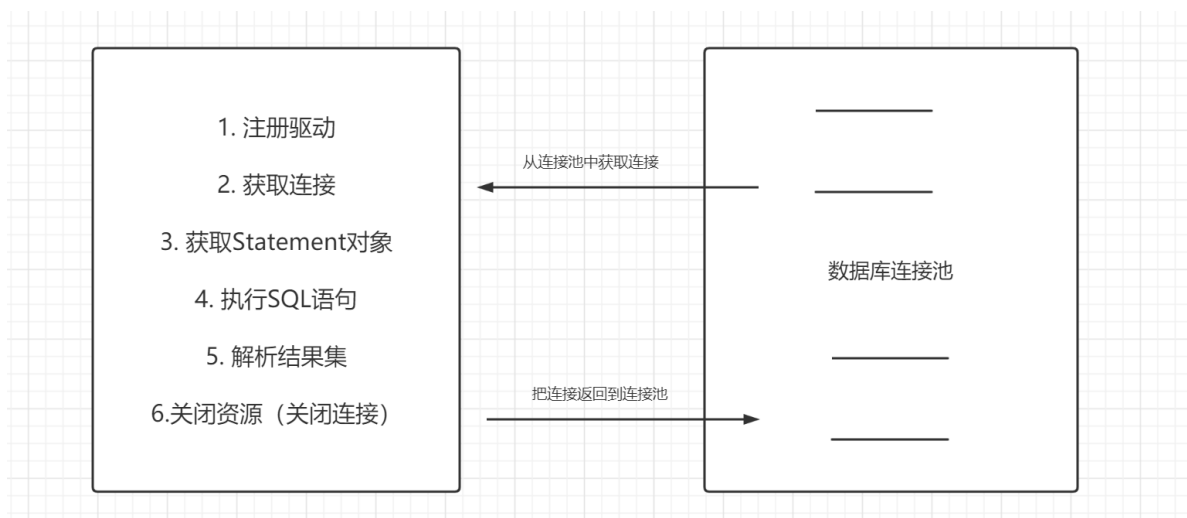
# Datasource

## 1. 介绍

在我们使用传统的JDBC的时候，我们需要在使用JDBC之前，创建一个数据库之间的连接，用完了之后把这个连接关闭掉；而这个连接（Connection）资源是可以反复利用的，所以我们可以考虑像线程池一样，用一个容器把若干连接维护起来，当我们需要连接的时候从池子中去获取连接对象，当用完了连接对象之后把这个连接返回到连接池。

其实就是利用池化的思想来维护连接，避免连接的反复创建于销毁。

连接池最根本的目的：提高程序的效率。



## 2. 手动实现

### v1

实现了连接池的两个基本的功能

- 获取连接
- 返回连接

```
// 从头部存，从尾部取
static LinkedList<Connection> pool;

static {

    // 实例化
    pool = new LinkedList<Connection>();

    // 初始化
    for (int i = 0; i < 10; i++) {

        Connection newConnection = JDBCUtils.getNewConnection();

        pool.addFirst(newConnection);

    }
}
```

```

}

// 获取连接
public static Connection getConnection(){

    // 取出尾部的那个连接
    Connection connection = pool.removeLast();

    return connection;

}

// 返回连接
public static void returnConnection(Connection connection){

    // 把连接放到头部
    pool.addFirst(connection);

}

```

## v2

- 增加扩容的功能

```

// 获取连接
public static Connection getConnection(){

    // 判断是否需要扩容
    if (pool.size() < 3) {

        addCapacity(5);

    }

    // 取出尾部的那个连接
    Connection connection = pool.removeLast();

    return connection;

}

```

## v3

- 增加可配置化

```

// 初识大小
static int INIT_SIZE = 10;

// 最小值
static int MIN_SIZE = 2;

// 扩容增量
static int INCREMENT = 5;

```

```

// 从头部存，从尾部取
static LinkedList<Connection> pool;

static {

    try {

        Properties properties = new Properties();
        properties.load(new FileInputStream("pool.properties"));

        String initSize = properties.getProperty("init_size");
        String minSize = properties.getProperty("min_size");
        String increment = properties.getProperty("increment");

        if (null != initSize && initSize.length() > 0) {
            INIT_SIZE = Integer.valueOf(initSize);
        }

        if (null != minSize && minSize.length() > 0) {
            MIN_SIZE = Integer.valueOf(minSize);
        }

        if (null != increment && increment.length() > 0) {
            INCREMENT = Integer.valueOf(increment);
        }
    } catch (Exception ex) {
        ex.printStackTrace();
        System.out.println("获取用户配置的连接池参数异常！使用默认配置！");
    }

    // 实例化
    pool = new LinkedList<Connection>();

    // 初始化
    addCapacity(INIT_SIZE);

}

```

此时，其实这个数据库连接池还是存在一些问题：

1. 假如用户关闭了连接之后再返回连接到连接池怎么办呢？

因为这个数据库连接池是我们自己设计的，那么其实可以把数据库连接池中的连接的close方法改掉，改为返回连接

- 可以通过继承，放对象的子类，这个子类中重写close方法
- 自己实现Connection接口，自己来实现接口中定义的方法

2. 连接池扩容没有上限

3. 连接池要能够超时自动回收

4. JDBC的设计者在考虑数据库连接池的时候，设计了一个接口 `javax.sql.DataSource`，如果我们自己实现一个数据库连接池，那么需要去实现这个接口

### 3. 开源的数据库连接池

在公司里面，一般不去使用自己写的数据库连接池，也就是一般不用自己造轮子。

#### DBCP

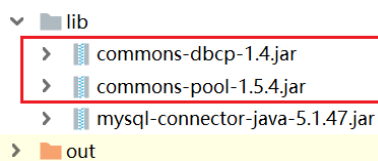
DBCP其实是一个早期的由Apache开源的数据库连接池。

Apache: Apache是世界上最大的开源组织。 <http://apache.org>

[官网](#)

如何使用呢？

- 导包（导入指定的jar包）



```
lib
├── commons-dbc-1.4.jar
├── commons-pool-1.5.4.jar
└── mysql-connector-java-5.1.47.jar
```

- 配置（配置数据库连接池）

```
driverClassName=com.mysql.jdbc.Driver
username=root
password=123456
url=jdbc:mysql://localhost:3306/44th

# 以下这些配置是有默认值的配置，可以不配，使用默认值
initialSize=20
connectionProperties=useSSL=false;characterEncoding=utf8
```

- 使用

```
public class DBCUtils {

    // 数据库连接池
    static DataSource dataSource;

    static {

        try {

            // 创建DBCP的数据库连接池
            Properties properties = new Properties();
            properties.load(new FileInputStream("dbcp.properties"));

            dataSource =
                BasicDataSourceFactory.createDataSource(properties);

        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

```

    }

    // 获取连接
    public static Connection getConnection(){

        Connection connection = null;
        try {
            connection = dataSource.getConnection();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return connection;

    }
}

```

```

//connection : PoolingDataSource$PoolGuardConnectionWrapper
//statement : DelegatingStatement
//resultSet : DelegatingResultSet

```

注意:

//调用connection 的close()方法不是把连接关闭,而是将连接返回给数据库连接池 **datasource**

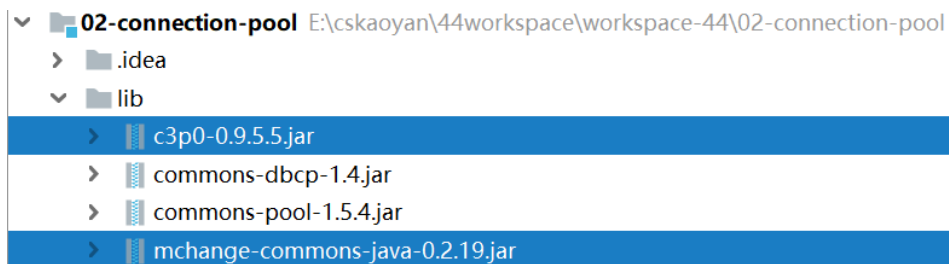
## C3p0

C3p0是继DBCP之后出现的一个开源的数据库连接池。

[官网](#)

如何使用呢?

- 导包



- 配置

- C3p0支持用户直接在代码中配置

```

// 声明一个数据库连接池
static DataSource dataSource;

static {

    try {

```

```

        ComboPooledDataSource comboPooledDataSource = new
        ComboPooledDataSource();

        // 设置
        comboPooledDataSource.setUser("root");

        comboPooledDataSource.setJdbcUrl("jdbc:mysql://localhost:3306/44th?
        useSSL=false&characterEncoding=utf8");
        comboPooledDataSource.setPassword("123456");
        comboPooledDataSource.setDriverClass("com.mysql.jdbc.Driver");

        // 设置初始连接大小
        comboPooledDataSource.setInitialPoolSize(10);

        dataSource = comboPooledDataSource;

    } catch (Exception ex) {
        ex.printStackTrace();
    }
}

```

- 也支持通过配置文件来配置

这个配置文件的名字和路径都是有要求的：需要在CLASSPATH（src）路径下配置一个名字叫做

c3p0-config.xml 的配置文件

```

<c3p0-config>
  <default-config>

    <property name="user">root</property>
    <property name="password">123456</property>
    <property name="driverClass">com.mysql.jdbc.Driver</property>
    <!--
      原字符      转义字符
      &          &amp;
      >          &gt;
      <          &lt;
    -->
    <property name="jdbcUrl">jdbc:mysql://localhost:3306/44th?
    useSSL=false&amp;characterEncoding=utf8</property>

    <property name="initialPoolSize">8</property>

  </default-config>
</c3p0-config>

```

```
// 声明一个数据库连接池
static DataSource dataSource;

static {

    try {
        // 使用配置文件中的默认配置 (default-config)
        dataSource = new ComboPooledDataSource();

    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
```

```
//connection: NewProxyConnection
//statement: NewProxyStatement
//resultSet: NewProxyResultSet
```

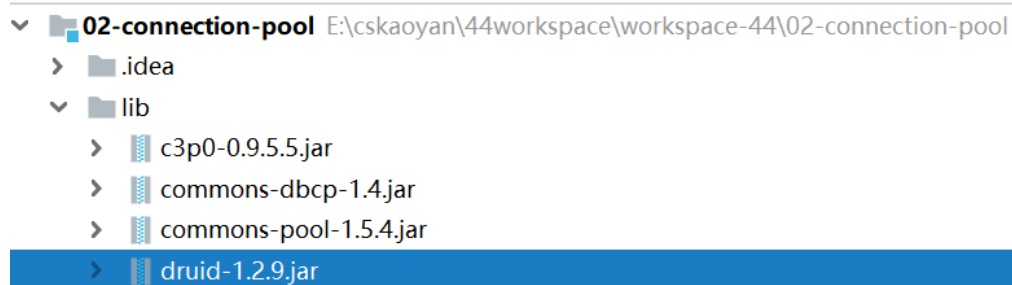
## Druid

是阿里巴巴开源的数据库连接池，目前由Apache来管理。

[官网](#)

如何使用呢？

- 导包



- 配置

Druid的使用方式和DBCP是类似的，需要我们自己去配置一个properties配置文件

配置文件的名字和路径是自己定义的，不固定

```
driverClassName=com.mysql.jdbc.Driver
username=root
password=123456
url=jdbc:mysql://localhost:3306/44th?useSSL=false&characterEncoding=utf8
```

- 使用

# Druid配置的名字基本上和DBCP是一致的

driverClassName=com.mysql.jdbc.Driver

username=root

password=123456

url=jdbc:mysql://localhost:3306/44th?useSSL=false&characterEncoding=utf8