- 公众号和Github待发文章：数据库：数据库连接池原理详解与自定义连接池实现
- 基于JDBC的数据库连接池技术研究与应用

- 数据库连接池技术详解

数据库连接本质就是一个 socket 的连接。数据库服务端还要维护一些缓存和用户权限信息之类的 所以占用了一些内存

连接池是维护的数据库连接的缓存，以便将来需要对数据库的请求时可以重用这些连接。为每个用户打开和维护数据库连接，尤其是对动态数据库驱动的网站应用程序的请求，既昂贵又浪费资源。**在连接池中，创建连接后，将其放置在池中，并再次使用它，因此不必建立新的连接。如果使用了所有连接，则会建立一个新连接并将其添加到池中。**连接池还减少了用户必须等待建立与数据库的连接的时间。

操作过数据库的朋友应该都知道数据库连接池这个概念，它几乎每天都在和我们打交道，但是你真的了解 **数据库连接池** 吗?

最小，最大连接 池可以抽象成一个链表

自己实现一个数据库连接池 注意：下面的代码我是分块讲解的，你一个个粘贴下来，最后肯定也是能运行的，为了方便，我会在讲完之后，给出完整的实现代码.

定义初始化连接数目，最大连接数以及当前已经连接的数目 一开始，当数据库连接池启动的时候，为了实现上面的需求，我们肯定是要先给出几个已经完成的连接的，这样用户访问的时候就能直接拿到了；此外，当某一段时间的访问用户超过我定义的连接池中的连接个数，肯定是要额外新建连接给用户使用；当然，这个新建的连接肯定是不能无限制的，否则还是会很影响效率的，所以，我们还要定义最大的连接数.

private final int init_count = 3; //初始化链接数目 private final int max_count = 6; //最大连接数 private int current_count = 0; //到当前连接数 复制代码 那么，我们一开始新建的连接池要放在哪里供用户使用呢？肯定是要创建一个连接集合的，这样的操作比较方便．至于为什么要使用LinkedList这样一个集合，一会就会介绍的.

private LinkedList pool = new LinkedList(); 复制代码 刚才说了，一开始我们肯定是要初始化连接给用户使用的，那么，就需要在连接池启动的时候就新建一定数量的链接．分析后发现，放在构造函数中初始化链接是最好不过的了，最后再将连接放在链接集合中.

//构造函数，初始化链接放入连接池 public MyPool() { for (int i=0;i<init_count;i++){ //记录当前连接数 current_count++; //createConnection是自定义的创建链接函数． Connection connection = createConnection(); pool.addLast(connection); } } 复制代码 创建一个新的连接，这个就没啥好说的了，毕竟你要的链接都是从这来的.

public Connection createConnection() { Class.forName("com.mysql.jdbc.Driver"); Connection connection =DriverManager.getConnection("jdbc:mysql://localhost:3306/keyan","root","root"); return connection; } 复制代码 获取链接 当用户来访问的时候，我们肯定是要给用户一个连接的，如果池中没有连接了（所有连接均被占用），那么就要创建新的连接，使用createConnection()函数，当然，这个连接的个数肯定是不能超过最大连接数的．如果不满足这两个条件，那么直接抛出异常.

public Connection getConnection() { if (pool.size() > 0){ //removeFirst删除第一个并且返回 //现在你一定看懂了我说的为什要用LinkedList了吧，因为下面的这个 //removeFirst()方法会将集合中的第一个元素删除，但是还会返回第一个元素 //这样就省去了我们很多不必要的麻烦 return pool.removeFirst(); } if (current_count < max_count){ //记录当前使用的连接数 current_count++; //创建链接 return createConnection(); } throw new

RuntimeException("当前链接已经达到最大连接数"); } 复制代码 释放资源 当用户使用完了连接之后，我们要做的并不是关闭连接，而是将连接重新放入资源池LinkedList之中，这样就省去了一遍又一遍的连接关闭．这个就是连接池的核心内容．是不是很简单?

public void releaseConnection(Connection connection){ if (pool.size() < init_count){ pool.addLast(connection); current_count--; }else { try { connection.close(); } catch (SQLException e) { e.printStackTrace(); } } } 复制代码整个的实现过程就是这样的，下面我把全部的代码贴出来，方便大家学习． //单元测试 @Test public class MyPool { private final int init_count = 3; //初始化链接数目 private final int max_count = 6; //最大 private int current_count = 0; //到当前连接数 //连接池，用来存放初始化链接 private LinkedList pool = new LinkedList();

```
//构造函数，初始化链接放入连接池
public MyPool() {
    for (int i=0;i<init_count;i++){
        //记录当前连接数
        current_count++;
        Connection connection = createConnection();
        pool.addLast(connection);
    }
}

//创建新的连接
public Connection createConnection() {
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/keyan","root","root");
        return connection;
    }catch (Exception e){
        System.out.println("数据库链接异常");
        throw new RuntimeException();
    }
}

//获取链接
public Connection getConnection() {
    if (pool.size() > 0){
        //removeFirst删除第一个并且返回
        return pool.removeFirst();
    }
    if (current_count < max_count){
        //记录当前使用的连接数
        current_count++;
        //创建链接
        return createConnection();
    }
    throw new RuntimeException("当前链接已经达到最大连接数");
}

//释放链接
public void releaseConnection(Connection connection){
```

```
        if (pool.size() < init_count){
            pool.addLast(connection);
            current_count--;
        }else {
            try {
                connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

} 复制代码自己跑了一遍代码之后，你是不是发现原来看起来很复杂的技术，并不像我们想得那样？ 好了，介绍完了基本的数据库连接池技术原理之后，我们就要介绍两个开源的优秀的数据连接池技术． 其实，真正的数据库连接池要考虑的东西比我们刚才写的这玩意复杂，现阶段不需要我们写这样复杂的东西，不过如果你感兴趣的话，可以看看数据库连接池的源代码 - - 没错，这两个连接池都是开源的．OK，接下来就开始吧！

没有数据库连接池之前

我相信你一定听过这样一句话：**Java语言中，JDBC（Java DataBase Connection）是应用程序与数据库沟通的桥梁。**