【设计模式学习】单例模式模拟数据库连接池

作者：zjhch123
原文链接：https://ld246.com/article/1467124424667
来源网站：链滴

<link rel="stylesheet" href="https://dn-maxiang.qbox.me/res-min/themes/marxico.css" />
<style><!--
.note-content {font-family: 'Helvetica Neue', Arial, 'Hiragino Sans GB', STHeiti, 'Microsoft YaHi', 'WenQuanYi Micro Hei', SimSun, Song, sans-serif;}
--></style>

<blockquote>
作业：使用单例模式模拟数据库连接池。要求客户端代码可以获取连接池的唯一对象，并且可以池中随机取一个连接对象来连接数据库。
</blockquote>
使用单例模式实现数据库连接池对象。

<h2 id="单例模式的优点">单例模式的优点:</h2>

提供对唯一实例的受控访问；
节约系统资源，防止出现Java版内存泄露；
允许可变数目的实例（没看懂）。

<h2 id="单例模式的缺点">单例模式的缺点</h2>

没有抽象层，对拓展有很大困难；
单例类指责过重；
语言的垃圾回收机制可能导致共享的单例对象状态丢失。

<h2 id="适用场景">适用场景</h2>

系统只需要一个实例对象；
客户调用类的单个实例只允许使用一个公共访问点，除了该公共访问点，不能通过其他途径访问实例。

<h2 id="我的代码">我的代码</h2>

<pre class="prettyprint hljs-dark"><code class="language-java hljs">package test;

import java.util.inkedList;

class <pan class="hljs-title">Connection {
 priate int id;

 public Connection</spa>(int id) {
 this.id = id;
 }

 public String toString() {
 return String.valueOf(id);
 }
}

class DataSource</spn> {
 private int maxCount; // 最大连接数
 private int minCount; <s

an class="hljs-comment">// 最低连接数
    privat
 int nowCreatedCount = <span class="hljs-nu
ber">0; // 当前已创建连接数
    <span cl
ss="hljs-keyword">private LinkedList<Connection> connections = <span class
"hljs-keyword">new LinkedList<Connection>();

    <span class="hlj
-keyword">private static <span class="hljs-cla
s">class HolderClass
/span>{ //使用IoDH方式获取单例
        <span clas
="hljs-keyword">private static <span class="hl
s-keyword">final DataSource instance = new
ataSource();
    }

    <span class="hljs-keyword
>public static DataSource <span class="hljs-tit
e">getInstance() {
        <span cla
s="hljs-keyword">return HolderClass.instance;
    }

    <span class
"hljs-function">private DataSo
rce() {
        <span class="hljs-co
ment">// 假设从配置文件中读取到了数据
        maxCount = <span class="hljs-n
mber">10;
        minCount = 5;
nowCreatedCount = 0;

        <span class="h
js-keyword">if (minCount > maxCount) {
            <span class="hljs-keyword
>throw new ExceptionInInitializerError(<span c
ass="hljs-string">"数据库连接池初始化异常");
        }
        <span class="hlj
-keyword">for (int i = <span class="hljs-numb
r">0; i < minCount; i++) {
            Connection conn = <span class="hljs-key
ord">new Connection(i);
            connections.addLast(conn);
            now
reatedCount++;
        }
        System.out.println("数据库
接池初始化完毕,当前已创建连接个数为:" + nowCreatedCount);
    }

public Connection <span c
ass="hljs-title">getConn() <span class="hljs-ke
word">throws Exception {
        synchron
zed (connections) {
            if (<span clas
="hljs-keyword">this.connections.size() > 0)

                System.out.println("成功获取连接");
            return connections.removeLast();

            if (nowCreatedCount < maxCount) {
                Connection conn = new Connection(nowCr
atedCount++);
                System.out.println("增加连接,当前
创建连接个数为:" + nowCreatedCount);
                System.out.println(<span clas
="hljs-string">"成功获取连接");
                return</
pan> conn;
            }
        }
        throw
new Exception("connection f
ll!");
    }

    <span class="hljs-keywor
">public void free</s
an>(Connection conn) {
        connections.
ddLast(conn);
    }

}

public</spa
> class <span class="hljs-title
>PlayConnectionPool {
    <span class="h
js-keyword">public static <span class="hljs-ke
word">void main(Strin
[] args) throws Exception {
ataSource ds = DataSource.getInstance();
        System.out.println(ds.getConn());
    System.out.println(ds.getConn());
        System.out.println(ds.getConn());
        S
stem.out.println(ds.getConn());
        System.out.println(ds.getConn());
        System
out.println(ds.getConn());
        System.out.println(ds.getConn());
    }
}<br /

</code></pre>

原文链接：【设计模式学习】单例模式模拟数据库连接池