

个人资料



Heaven-Wang

+ 关注

发私信

访问：158990次

积分：2551

等级：

BLOG

> 5

排名：第10130名

原创：85篇

转载：12篇

译文：3篇

评论：40条

文章搜索

Q

博客专栏

 [Hadoop教程](#)

文章：9篇

阅读：8263

 [Java 从基础到高级](#)

文章：3篇

阅读：977

 [Storm 实时计算](#)

文章：10篇

阅读：22292

文章分类

Java基础知识 (22)

Java高级技术 (11)

Hadoop (12)

Storm (10)

Kafka (6)

Redis (2)

Memcache (4)

Linux (0)

HBase (1)

算法 (0)

Netty (7)

框架 (1)

Web编程 (10)

数据库 (2)

【公告】博客系统优化升级

【收藏】Scala 资源一应俱全

博乐招募开始啦

程序员七夕表白礼品指南

原

FTPClient连接池的实现

标签：

多线程

FTPClient

对象池

连接池

2015-10-08 10:59

1659人阅读

评论

分类：

Java高级技术 (10)

快速回复

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录(?)

[+]

最近在写一个FTP上传工具，用到了Apache的FTPClient，为了提高上传效率，我采用了多线程的方式，但是每个线程频繁的创建会造成不必要的开销，因此，此处最好使用一个FTPClient连接池。仔细翻了一下Apache的api，发现它并没有一个FTPClient不自己写一个FTPClientPool。下面就大体介绍一下开发连接池的整个过程，供大家参考。

关于对象池

有些对象的创建开销是比较大的，比如数据库连接等。为了减少频繁创建、销毁对象带来的性能消耗，我们可以利用对象池的技术池提供了一种机制，它可以管理对象池中对象的生命周期，提供了获取和释放对象的方法，可以让客户端很方便的使用对象池

如果我们要自己实现一个对象池，一般需要完成如下功能：

1. 如果池中有可用的对象，对象池应当能返回给客户端
2. 客户端把对象放回池里后，可以对这些对象进行重用
3. 对象池能够创建新的对象来满足客户端不断增长的需求
4. 需要有一个正确关闭池的机制来结束对象的生命周期

Apache的对象池工具包

为了方便我们开发自己的对象池，Apache 提供的common-pool工具包，里面包含了开发通用对象池的一些接口和实现类。其是ObjectPool 和PoolableObjectFactory。

ObjectPool接口中有几个最基本的方法：

1. addObject() ：添加对象到池
2. borrowObject()：客户端从池中借出一个对象
3. returnObject()：客户端归还一个对象到池中
4. close()：关闭对象池，清理内存释放资源等
5. setFactory(ObjectFactory factory)：需要一个工厂来制造池中的对象

PoolableObjectFactory接口中几个最基本的方法：

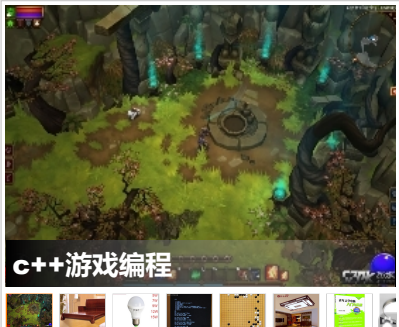
1. makeObject()：制造一个对象
2. destroyObject()：销毁一个对象
3. validateObject()：验证一个对象是否还可用

通过以上两个接口我们就可以自己实现一个对象池了。

实例：开发一个FTPClient对象池

最近在开发一个项目，需要把hdfs中的文件上传到一组ftp服务器，为了提高效率，我使用了Apache common-net包中的FTPClient，但Apache并没有提供FTPClientPool，所以我们自己实现一个。

关闭



c++游戏编程

性能方面	(4)
架构设计	(0)
数据挖掘	(1)
综合管理	(1)
组件及工具	(3)
设计模式	(4)
Spring	(0)
Spring MVC	(4)

文章存档	
2016年08月	(5)
2016年07月	(5)
2016年06月	(9)
2016年05月	(3)
2016年04月	(1)
展开	

阅读排行	
Kafka详解二、如何配置Kafka...	(10253)
Netty4详解三：Netty架构设计	(8195)
kafka详解三：开发Kafka应用	(7777)
Netty4详解二：开发第一个N...	(6757)
Memcache,Redis,MongoDB...	(6734)
kafka详解一、Kafka简介	(6305)
Kafka详解五、Kafka Consu...	(6240)
实例：Netty 基于Http协议下...	(5663)
Storm专题一、Storm DRPC ...	(4959)
kafka详解四：Kafka的设计思...	(4681)

关于我	
本人现任职于中国电信集团云公司，负责大数据能力平台建设与运营相关工作	
gitHub地址： https://github.com/suifeng3051	
QQ:490095337	

FTPClientPool来复用FTPClient连接。

通过上面的介绍，我们可以利用Apache提供的common-pool包来协助我们开发连接池。而开发一个简单的对象池，仅需要实现ObjectPool和PoolableObjectFactory两个接口即可。下面就看一下我写的实现：

写一个ObjectPool接口的实现FTPClientPool

```
import java.io.IOException;
import java.util.NoSuchElementException;
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.TimeUnit;

import org.apache.commons.net.ftp.FTPClient;
import org.apache.commons.pool.ObjectPool;
import org.apache.commons.pool.PoolableObjectFactory;

/**
 * 实现了一个FTPClient连接池
 * @author heaven
 */
public class FTPClientPool implements ObjectPool<FTPClient>{
    private static final int DEFAULT_POOL_SIZE = 10;
    private final BlockingQueue<FTPClient> pool;
    private final FtpClientFactory factory;

    /**
     * 初始化连接池，需要注入一个工厂来提供FTPClient实例
     * @param factory
     * @throws Exception
     */
    public FTPClientPool(FtpClientFactory factory) throws Exception{
        this(DEFAULT_POOL_SIZE, factory);
    }

    /**
     *
     * @param maxPoolSize
     * @param factory
     * @throws Exception
     */
    public FTPClientPool(int poolSize, FtpClientFactory factory) throws Exception {
        this.factory = factory;
        pool = new ArrayBlockingQueue<FTPClient>(poolSize*2);
        initPool(poolSize);
    }

    /**
     * 初始化连接池，需要注入一个工厂来提供FTPClient实例
     * @param maxPoolSize
     * @throws Exception
     */
    private void initPool(int maxPoolSize) throws Exception {
        for(int i=0;i<maxPoolSize;i++){
            //往池中添加对象
            addObject();
        }
    }

    /** (non-Javadoc)
     * @see org.apache.commons.pool.ObjectPool#borrowObject()
     */
    public FTPClient borrowObject() throws Exception, NoSuchElementException, IllegalStateException {
        FTPClient client = pool.take();
        if (client == null) {
            client = factory.makeObject();
            addObject();
        }else if(!factory.validateObject(client)){//验证不通过
            //使对象在池中失效
            invalidateObject(client);
            //制造并添加新对象到池中
            client = factory.makeObject();
            addObject();
        }
        return client;
    }
}
```

快速回复

关闭



```

/* (non-Javadoc)
 * @see org.apache.commons.pool.ObjectPool#returnObject(java.lang.Object)
 */
public void returnObject(FTPClient client) throws Exception {
    if ((client != null) && !pool.offer(client, 3, TimeUnit.SECONDS)) {
        try {
            factory.destroyObject(client);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

public void invalidateObject(FTPClient client) throws Exception {
    //移除无效的客户端
    pool.remove(client);
}

/* (non-Javadoc)
 * @see org.apache.commons.pool.ObjectPool#addObject()
 */
public void addObject() throws Exception, IllegalStateException, UnsupportedOperationException {
    //插入对象到队列
    pool.offer(factory.makeObject(), 3, TimeUnit.SECONDS);
}

public int getNumIdle() throws UnsupportedOperationException {
    return 0;
}

public int getNumActive() throws UnsupportedOperationException {
    return 0;
}

public void clear() throws Exception, UnsupportedOperationException {
}

/* (non-Javadoc)
 * @see org.apache.commons.pool.ObjectPool#close()
 */
public void close() throws Exception {
    while(pool.iterator().hasNext()){
        FTPClient client = pool.take();
        factory.destroyObject(client);
    }
}

public void setFactory(PoolableObjectFactory<FTPClient> factory) throws IllegalStateException, UnsupportedOperationException {
}

```

快速回复

}

再写一个PoolableObjectFactory接口的实现FTPClientFactory

```

import java.io.IOException;
import org.apache.commons.net.ftp.FTPClient;
import org.apache.commons.net.ftp.FTPReply;
import org.apache.commons.pool.PoolableObjectFactory;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.hdfstoftp.util.FTPClientException;

/**
 * FTPClient工厂类，通过FTPClient工厂提供FTPClient实例的创建和销毁
 * @author heaven
 */
public class FtpClientFactory implements PoolableObjectFactory<FTPClient> {
    private static Logger logger = LoggerFactory.getLogger("file");
    private FTPClientConfigure config;
    //给工厂传入一个参数对象，方便配置FTPClient的相关参数
    public FtpClientFactory(FTPClientConfigure config) {
        this.config=config;
    }

```

关闭



```
}

/* (non-Javadoc)
 * @see org.apache.commons.pool.PoolableObjectFactory#makeObject()
 */
public FTPClient makeObject() throws Exception {
    FTPClient ftpClient = new FTPClient();
    ftpClient.setConnectTimeout(config.getClientTimeout());
    try {
        ftpClient.connect(config.getHost(), config.getPort());
        int reply = ftpClient.getReplyCode();
        if (!FTPReply.isPositiveCompletion(reply)) {
            ftpClient.disconnect();
            logger.warn("FTPServer refused connection");
            return null;
        }
        boolean result = ftpClient.login(config.getUsername(), config.getPassword());
        if (!result) {
            throw new FTPClientException("ftpClient登陆失败! userName:" + config.getUsername() + "; passw");
        }
        ftpClient.setFileType(config.getTransferFileType());
        ftpClient.setBufferSize(1024);
        ftpClient.setControlEncoding(config.getEncoding());
        if (config.getPassiveMode().equals("true")) {
            ftpClient.enterLocalPassiveMode();
        }
    } catch (IOException e) {
        e.printStackTrace();
    } catch (FTPClientException e) {
        e.printStackTrace();
    }
    return ftpClient;
}

/* (non-Javadoc)
 * @see org.apache.commons.pool.PoolableObjectFactory#destroyObject(java.lang.Object)
 */
public void destroyObject(FTPClient ftpClient) throws Exception {
    try {
        if (ftpClient != null && ftpClient.isConnected()) {
            ftpClient.logout();
        }
    } catch (IOException io) {
        io.printStackTrace();
    } finally {
        // 注意,一定要在finally代码中断开连接,否则会导致占用ftp连接情况
        try {
            ftpClient.disconnect();
        } catch (IOException io) {
            io.printStackTrace();
        }
    }
}

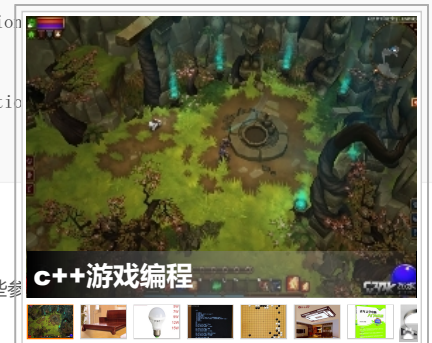
/* (non-Javadoc)
 * @see org.apache.commons.pool.PoolableObjectFactory#validateObject(java.lang.Object)
 */
public boolean validateObject(FTPClient ftpClient) {
    try {
        return ftpClient.sendNoOp();
    } catch (IOException e) {
        throw new RuntimeException("Failed to validate client: " + e, e);
    }
}

public void activateObject(FTPClient ftpClient) throws Exception {
}

public void passivateObject(FTPClient ftpClient) throws Exception {
}

}
```

最后,我们最好给工厂传递一个参数对象,方便我们设置FTPClient的一些参



```
package org.apache.commons.pool.impl.contrib;

/**
 * FTPClient配置类, 封装了FTPClient的相关配置
 *
 * @author heaven
 */
public class FTPClientConfigure {
    private String host;
    private int port;
    private String username;
    private String password;
    private String passiveMode;
    private String encoding;
    private int clientTimeout;
    private int threadNum;
    private int transferFileType;
    private boolean renameUploaded;
    private int retryTimes;

    public String getHost() {
        return host;
    }

    public void setHost(String host) {
        this.host = host;
    }

    public int getPort() {
        return port;
    }

    public void setPort(int port) {
        this.port = port;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getPassiveMode() {
        return passiveMode;
    }

    public void setPassiveMode(String passiveMode) {
        this.passiveMode = passiveMode;
    }

    public String getEncoding() {
        return encoding;
    }

    public void setEncoding(String encoding) {
        this.encoding = encoding;
    }

    public int getClientTimeout() {
        return clientTimeout;
    }

    public void setClientTimeout(int clientTimeout) {
        this.clientTimeout = clientTimeout;
    }
}
```



```
public int getThreadNum() {
    return threadNum;
}

public void setThreadNum( int threadNum) {
    this. threadNum = threadNum;
}

public int getTransferFileType() {
    return transferFileType;
}

public void setTransferFileType( int transferFileType) {
    this. transferFileType = transferFileType;
}

public boolean isRenameUploaded() {
    return renameUploaded;
}

public void setRenameUploaded( boolean renameUploaded) {
    this. renameUploaded = renameUploaded;
}

public int getRetryTimes() {
    return retryTimes;
}

public void setRetryTimes( int retryTimes) {
    this. retryTimes = retryTimes;
}

@Override
public String toString() {
    return "FTPClientConfig [host=" + host + "\n port=" + port + "\n username=" + username + "\n password"
        + "\n encoding=" + encoding + "\n clientTimeout=" + clientTimeout + "\n threadNum=" + thre
        + transferFileType + "\n renameUploaded=" + renameUploaded + "\n retryTimes=" + retryTimes
    }
}
```

快速回复

FTPClientPool连接池类管理FTPClient对象的生命周期，负责对象的借出、规划、池的销毁等；FTPClientPool类依赖于**FtpCli**程类来制造和销毁对象；FtpClientFactory又依赖**FTPClientConfigure**类，FTPClientConfigure负责封装FTPClient的配置参数连接池就开发完成了。

需要注意的是，FTPClientPool中用到了一个阻塞队列**ArrayBlockingQueue**来管理存放FTPClient对象，关于阻塞队列，请参考发之】BlockingQueue：<http://blog.csdn.net/suifeng3051/article/details/48807423>

顶0

踩0

- ▲ 上一篇 Hadoop核心之HDFS 架构设计
- ▼ 下一篇 Java synchronized 介绍

我的同类文章

Java高级技术 (10)		
• 什么是线程安全	2016-08-09	阅读 81
• java thread中的wait()和notify()	2016-07-07	阅读 45
• Java 注解详解 (annotation)	2016-07-01	阅读 86
• java 并发编程	2015-11-03	阅读 355
• Java线程池 ExecutorService	2015-10-27	阅读 406

关闭



猜你在找

- Python自动化开发基础 FTP上传…
 - 上传类开发视频教程
 - 最涨薪的技能-PHP微信接口开发
 - 搞定Ftp上传，远程管理Linux服…
 - 【直通华为HCNA/HCNP系列R篇3】…
- jsp实现分页显示信息数据库EL表…
 - 如何实现Tomcat连接池数据库密…
 - Hibernate整合C3P0实现连接池
 - 数据库连接池的实现及原理
 - 使用JAVA中的动态代理实现数据…



查看评论

快速回复



老鼠尼奥

1楼 20

ftpClient.setControlEncoding(config.getEncoding());

上面这句位置写错了，会不起作用。查看官方文档有下面句：
//Please note that this has to be set before the connection is established.



Heaven-Wang

Re: 20

回复老鼠尼奥：多谢提醒~~

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

- | | | | | | | | | | | | | |
|---------------|------------|--------|------------|-----------|-----------|-----------|-----------|-----------|------------|----------------|--------|-----|
| 全部主题 | Hadoop | AWS | 移动游戏 | Java | Android | iOS | Swift | 智能硬件 | Docker | OpenStack | VPI | |
| IE10 | Eclipse | CRM | JavaScript | 数据库 | Ubuntu | NFC | WAP | jQuery | BI | HTML5 | Spring | Apa |
| HTML | SDK | IIS | Fedora | XML | LBS | Unity | Splashtop | UML | components | Windows Mobile | Rai | |
| Cassandra | CloudStack | FTC | coremail | OPhone | CouchBase | 云计算 | iOS6 | Rackspace | Web App | S | | |
| Compuware | 大数据 | aptech | Perl | Tornado | Ruby | Hibernate | ThinkPHP | HBase | Pure | Solr | An | |
| Cloud Foundry | Redis | Scala | Django | Bootstrap | | | | | | | | |

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持
京 ICP 证 09002463 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved

