LOG 4J 日志管理

——基于 DBCP 和 Mysql

版本：V1.0

日期：2009-10-2

作者：孙欢欢(shh-cc@163.com)

目录

1.DBCP配置

1.1 准备 mysql 的 jdbc 驱动程序

1.2 安装 Tomcat 默认全部装在 D:\tomcat6.0

1.3 以 TOMCAT_HOME 代替个版本的安装目录

1.4 将 jdbc 驱动 mysql-connector-java-5.0.8-bin.jar ， 和 commons-dbcp.jar ， commons-pool-1.4.jar 放入 TOMCAT_HOME\ common\lib\ 下面

1.5 如果是其他数据库的话，就是换一下 jdbc 驱动以及各种参数就可以了的

1.6 配置 tomcat 数据源时的各种参数的详细介绍

1.7 其他设置。

1.8 在 server.xml<GlobalNamingResources> </GlobalNamingResources>之间添

```
<!--mysql DBCP-->
    <Resource
     name="jdbc/TestDB"
     auth="Container"
     type="javax.sql.DataSource"
     driverClassName="com.mysql.jdbc.Driver"
     url="jdbc:mysql://localhost:3306"
     maxActive="50"
     maxldle="10"
     maxWait="5000"
```

username="root"

password="000000" />

1.9.在context.xml的

<Context></Context>之间添加

<ResourceLink

global="jdbc/TestDB"

name="jdbc/TestDB"

type="javax.sql.DataSource"/>

1.10 配置参数说明

Username	JDBC 驱动建立连接时所需的用户名。
Password	JDBC 驱动建立连接时所需的用户密码。
url	JDBC 驱动建立连接时的连接地址。
driverClassName	使用的 JDBC 驱动完整 JAVA 类名。
connectionProperties	JDBC 驱动建立连接时附带的连接属性 属性的格式必须为这样：[属性名=property;] * 注意："user" 与 "password" 两个属性会被明确地传递，因此这里不需要包含他们。
defaultAutoCommit	true 指定由连接池所创建的连接的自动提交（auto-commit）状态。

defaultReadOnly driver default 指定由连接池所创建的连接的只读（read-only）状态。如果没有设置该值，则“setReadOnly”方法将不被调用。（某些驱动并不支持只读模式，如：Informix）

defaultTransactionIsolation

driver default

指定由连接池所创建的连接的事务级别（TransactionIsolation）。可用值为下列之一：（详情可见 javadoc。）

NONE

READ_COMMITTED

READ_UNCOMMITTED

REPEATABLE_READ

SERIALIZABLE

defaultCatalog

指定由连接池所创建的连接的默认日志。

initialSize 0 初始化连接池时创建的连接数。

此版后开始提供：1.2

maxActive 8 连接池允许的最大并发连接数，值为非正数时表示不限制。

maxIdle 8 连接池中的最大空闲连接数，超过此数值时多余的空闲连接将会被释放，值为负数时表示不限制。

minIdle 0 连接池中的最小空闲连接数，低于此数值时将会创建所欠缺的连接，值为 0 时表示不创建。

maxWait　　indefinitely 以毫秒表示的当连接池中没有可用连接时等待可用连接返回的时间，超时则抛出异常，值为-1 时无限期等待。

validationQuery　　在连接返回给调用者前用于校验连接是否有效的 SQL 语句。如果指定了 SQL 语句，则必须为一个“SELECT”语句，且至少会返回一行结果。

testOnBorrow　　true

指定连接被调用时是否经过校验。如果校验未通过，则该连接被连接池断掉，并由连接池尝试调用另一个连接。

注意：要想值为 true 时该设置生效，则 validationQuery 参数必须为一个非空字串。

testOnReturn　　false

指定连接返回到池中时是否经过校验。

注意：要想值为 true 时该设置生效，则 validationQuery 参数必须为一个非空字串。

testWhileIdle　　false

指定连接进入空闲状态时是否经过空闲对象驱逐进程的校验（如果存在空闲对象驱逐进程）。如果校验未通过，则该连接被连接池断掉。

注意：要想值为 true 时该设置生效，则 validationQuery 参数必须为一个非空字串。

timeBetweenEvictionRunsMillis　　-1

以毫秒表示的空闲对象驱逐进程由运行状态进入休眠状态的数值。值为非正数时表示不运行任何空闲对象驱逐进程。

numTestsPerEvictionRun　　3

连接池检查每个空闲对象驱逐进程的对象数量(如果存在空闲对象驱逐进程)。

minEvictableIdleTimeMillis 1000 * 60 * 30

以毫秒表示的连接被空闲对象驱逐进程驱逐前在池中保持空闲状态的最小时间（如果存在空闲对象驱逐进程)。

poolPreparedStatements false

启用“PreparedStatements”缓存池。

maxOpenPreparedStatements unlimited

由“PreparedStatements”缓存池中取得“PreparedStatements”的最大并发数值，值为 0 时表示不限制。此组件也有能力对“PreparedStatements”对象进行缓存。当启用了“PreparedStatements”缓存池时，每个连接通过下列方法建立的“PreparedStatements”对象都会被放入缓存池：

public PreparedStatement prepareStatement(String sql)

public PreparedStatement prepareStatement(String sql, int resultSetType, int resultSetConcurrency)

注意：请保证你的连接有剩余的资源为给其他语句。

accessToUnderlyingConnectionAllowed false

控制是否允许“PoolGuard”优先使用连接。启用此项后，你可以按照下列方法优先使用连接：

```
Connection conn = ds.getConnection();
Connection dconn = ((DelegatingConnection) conn).getInnermostDelegate();
...
```

conn.close()默认为“false”，这是一个具有潜在威胁的选项，不当的程序可能会做出可怕的事情来（在关闭优先连接或当被守护的连接已经关闭后继续使用优先连接的时候)。小心并且尽量只在当你需要直接驱动指定的扩展时使用。

注意：不要关闭优先连接，它只有一个。

removeAbandoned　　false

是否清除已经超过“removeAbandonedTimout”设置的无效连接。

如果值为“true”则超过“removeAbandonedTimout”设置的无效连接将会被清除。设置此属性可以从那些没有合适关闭连接的程序中恢复数据库的连接。

removeAbandonedTimeout　　300　　以秒表示的清除无效连接的时限。

logAbandoned　　false

当清除无效连接时是否在日志中记录清除信息的标志。

记录无效的语句和连接，并附加每个连接开启或新建一个语句的系统开销。

如果你启用了“removeAbandoned”，可能会导致被设为无效的连接被连接池回收。这个机制将会在满足下列两个条件时启动：(getNumIdle() < 2) 和 (getNumActive() > getMaxActive() - 3)

例如：假设 maxActive=20，而当前已经拥有 18 个活动连接，1 个空闲连接，“removeAbandoned”机制将会启动。 但是只有在活动连接没有使用的时长超过“removeAbandonedTimeout”（默认为 300 秒）的连接被清除。在遍历结果集时，所使用的连接不会被标为活动连接。

2. Log4j 配置

2.1 下载最新的Log4j.jar包，放到相关项目的Lib下，并添加进项目的Libraris.

2.2 把log4j.properties文件放到项目的SRC根目录下

2.3 关于log4j.properties

2. 3. 1 log4j.rootLogger = [level] , appenderName, appenderName, …

其中，level 是日志记录的优先级，分为 OFF、FATAL、ERROR、WARN、INFO、DEBUG、ALL 或者您定义的级别。Log4j 建议只使用四个级别，优先级从高到低分别是 ERROR、WARN、INFO、DEBUG。通过在这里定义的级别，您可以控制到应用程序中相应级别的日志信息的开关。比如在这里定义了 INFO 级别，则应用程序中所有 DEBUG 级别的日志信息将不被打印出来。

appenderName 就是指定日志信息输出到哪个地方。您可以同时指定多个输出目的地。

2. 3. 2 日志输出方式appender（目的地）

log4j.appender.appenderName = fully.qualified.name.of.appender.class

log4j.appender.appenderName.option1 = value1

…

log4j.appender.appenderName.option = valueN

其中，Log4j 提供的 appender 有以下几种：

org.apache.log4j.ConsoleAppender（控制台），

org.apache.log4j.FileAppender（文件），

org.apache.log4j.DailyRollingFileAppender（每天产生一个日志文件），

org.apache.log4j.RollingFileAppender（文件大小到达指定尺寸的时候产生一个新的文件），

org.apache.log4j.WriterAppender（将日志信息以流格式发送到任意指定的地方）

(1).ConsoleAppender 选项

Threshold=WARN:指定日志消息的输出最低层次。

ImmediateFlush=true:默认值是 true,意谓着所有的消息都会被立即输出。

Target=System.err：默认情况下是：System.out,指定输出控制台

(2).FileAppender 选项

Threshold=WARN:指定日志消息的输出最低层次。

ImmediateFlush=true:默认值是 true,意谓着所有的消息都会被立即输出。

File=mylog.txt:指定消息输出到 mylog.txt 文件。

Append=false:默认值是 true,即将消息增加到指定文件中，false 指将消息覆盖指定的文件内容。

(3).DailyRollingFileAppender 选项

Threshold=WARN:指定日志消息的输出最低层次。

ImmediateFlush=true:默认值是 true,意谓着所有的消息都会被立即输出。

File=mylog.txt:指定消息输出到 mylog.txt 文件。

Append=false:默认值是 true,即将消息增加到指定文件中，false 指将消息覆盖指定的文件内容。

DatePattern='.'yyyy-ww:每周滚动一次文件，即每周产生一个新的文件。当然也可以指定按月、周、天、时和分。即对应的格式如下：

1)'.'yyyy-MM: 每月

2)'.'yyyy-ww: 每周

3)'.'yyyy-MM-dd: 每天

4)'.'yyyy-MM-dd-a: 每天两次

5)'.'yyyy-MM-dd-HH: 每小时

6)'.'yyyy-MM-dd-HH-mm: 每分钟

(4).RollingFileAppender 选项

Threshold=WARN:指定日志消息的输出最低层次。

ImmediateFlush=true:默认值是 true,意谓着所有的消息都会被立即输出。

File=mylog.txt:指定消息输出到 mylog.txt 文件。

Append=false:默认值是 true,即将消息增加到指定文件中，false 指将消息覆盖指定的文件内容。

MaxFileSize=100KB: 后缀可以是 KB, MB 或者是 GB. 在日志文件到达该大小时，将会自动滚动，即将原来的内容移到 mylog.log.1 文件。

MaxBackupIndex=2:指定可以产生的滚动文件的最大数。

2. 3. 3 日志信息的布局

log4j.appender.appenderName.layout = fully.qualified.name.of.layout.class

log4j.appender.appenderName.layout.option1 = value1

…

log4j.appender.appenderName.layout.option = valueN

其中，Log4j 提供的 layout 有以下几种：

org.apache.log4j.HTMLLayout（以 HTML 表格形式布局），

org.apache.log4j.PatternLayout（可以灵活地指定布局模式），

org.apache.log4j.SimpleLayout（包含日志信息的级别和信息字符串），

org.apache.log4j.TTCCLayout（包含日志产生的时间、线程、类别等等信息）

2. 3. 4 日志格式控制

在配置文件中可以通过 log4j.appender.A1.layout.ConversionPattern 设置日志输出格式。

参数：

%p: 输出日志信息优先级，即 DEBUG，INFO，WARN，ERROR，FATAL,

%d: 输出日志时间点的日期或时间，默认格式为 ISO8601，也可以在其后指定格式，比如：%d{yyy MMM dd HH:mm:ss,SSS}，输出类似：2002 年 10 月 18 日 22：10：28，921

%r: 输出自应用启动到输出该 log 信息耗费的毫秒数

%c: 输出日志信息所属的类目，通常就是所在类的全名

%t: 输出产生该日志事件的线程名

%l: 输出日志事件的发生位置，相当于%C.%M(%F:%L)的组合,包括类目名、发生的线程，以及在代码中的行数。举例：Testlog4.main(TestLog4.java:10)

%x: 输出和当前线程相关联的 NDC(嵌套诊断环境),尤其用到像 java servlets 这样的多客户多线程的应用中。

%%: 输出一个”%”字符

%F: 输出日志消息产生时所在的文件名称

%L: 输出代码中的行号

%m: 输出代码中指定的消息,产生的日志具体信息

%n: 输出一个回车换行符，Windows 平台为”\r\n”，Unix 平台为”\n”输出日志信息换行

可以在%与模式字符之间加上修饰符来控制其最小宽度、最大宽度、和文本的对齐方式。如：

1)%20c：指定输出 category 的名称，最小的宽度是 20，如果 category 的名称小于 20 的话，默认的情况下右对齐。

2)%-20c:指定输出 category 的名称，最小的宽度是 20，如果 category 的名称小于 20 的话，”-”号指定左对齐。

3)%.30c:指定输出 category 的名称，最大的宽度是 30，如果 category 的名称大于 30 的话，就会将左边多出的字符截掉，但小于 30 的话也不会有空格。

4)%20.30c:如果 category 的名称小于 20 就补空格，并且右对齐，如果其名称长于 30 字符，就从左边交远销出的字符截掉。

3.程序中使用

3.1 写一个获得数据库连接的工具类

```
import java.sql.Connection;
import java.sql.SQLException;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource; /**
 * @projectName    Log4jTest
 * @package        Log4jTest
 * @fileName       DBPool.java
 * @author         孙欢欢
 * @date           2009-9-30
 * @version        1.0
 * @description    通过从数据库连接池获取连接和关闭连接
 */
```

```
public class DBPool {
    private static DataSource ds = null;
    private static Connection conn = null;

    public static Connection getConnection() {
        try {
            if(conn==null){
            ds = (DataSource) (new InitialContext())
                        .lookup("java:comp/env/jdbc/TestDB");
            }
            if (ds != null) {
                conn = ds.getConnection();
            }
        } catch (NamingException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return conn;
    }
    public static void closeConnection() {
        try {
            if (conn != null) {
                conn.close();
                conn=null;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

}

3.2 继承JDBCAppender类

重写 closeConnection,getConnection,getLogStatement，close 四个方法：

```
package Log4jTest;

import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;

import org.apache.log4j.jdbc.JDBCAppender;
import org.apache.log4j.spi.ErrorCode;
import org.apache.log4j.spi.LoggingEvent;

/**
 * @projectName Log4jTest
 * @package Log4jTest
 * @fileName JDBCPoolAppender.java
 * @author 孙欢欢
 * @date 2009-9-30
 * @version 1.0
 * @description 为了能使用数据库连接池，必须继承 JDBCAppender，并重写其
closeConnection，getConnection，
 *                    getLogStatement 三个方法
 */

public class JDBCPoolAppender extends JDBCAppender {
```

```
@Override
public void closeConnection(Connection con) {
	DBPool.closeConnection(con);
}

@Override
public Connection getConnection() throws SQLException {
	if (connection == null) {
		connection = DBPool.getConnection();
	}
	return connection;
}

@Override
public String getLogStatement(LoggingEvent event) {
	StringBuffer sbuf = new StringBuffer();
	sbuf.append(layout.format(event));
	if (layout.ignoresThrowable()) {
		sbuf.delete(sbuf.length() - 2, sbuf.length());
		String[] s = event.getThrowableStrRep();
		if (s != null) {
			for (int j = 0; j < s.length; j++) {
				sbuf.append("\r\n ");
				sbuf.append(s[j]);
			}
		}
		sbuf.append("')");
	}
	return sbuf.toString();
```

```
}

@Override
public void execute(String sql) throws SQLException {

    Connection con = null;
    Statement stmt = null;
    String name = "张三";
    System.out.println(sql);

    int end = sql.lastIndexOf(")");
    sql = sql.substring(0, end) + ",'" + name + "')";

    System.out.println(sql);

    try {
        con = getConnection();

        stmt = con.createStatement();
        stmt.executeUpdate(sql);
    } catch (SQLException e) {
        if (stmt != null)
            stmt.close();
        throw e;
    }
    stmt.close();
    //closeConnection(con);

}
```

3.3 在程序中使用 test.jsp

```
import java.io.IOException;

import java.io.PrintWriter;


import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.apache.log4j.Logger;


public class dfsdg extends HttpServlet {

    protected void service(HttpServletRequest arg0, HttpServletResponse arg1)

            throws ServletException, IOException {

        Logger log=Logger.getLogger(getClass());

        log.debug("测试 log4j");

    }

}
```

结果：

控制台日志：

```
DEBUG[PRIORITY]
[NDC]
http-80-1[thread] ndfsdg[CATEGORY]
测试log4j[MESSAGE]
```

数据库记录：

2009-10-02 15:55:54	http-80-1	dfsdg	17	测试log4j	DEBUG

4. 附录：log4j.perporties

```
# Set log levels #
log4j.rootLogger = DEBUG, Console, LogFile,DATABASE

log4j.addivity.org.apache=true
# ??????
log4j.appender.Console=org.apache.log4j.ConsoleAppender
log4j.appender.Threshold=DEBUG
log4j.appender.Console.Target=System.out
log4j.appender.Console.layout=org.apache.log4j.PatternLayout
#log4j.appender.Console.layout.ConversionPattern=[framework] %d - %c
-%-4r [%t] %-5p %c %x - %m%n
log4j.appender.Console.layout.ConversionPattern=[start]%d{DATE}[DATE]
%n%p[PRIORITY]%n%x[NDC]%n%t[thread] n%c[CATEGORY]%n%m[MESSAGE]%n%n
#?????
log4j.appender.LogFile=org.apache.log4j.FileAppender
log4j.appender.LogFile.File=file.log
log4j.appender.LogFile.Append=false
log4j.appender.LogFile.layout=org.apache.log4j.PatternLayout
log4j.appender.LogFile.layout.ConversionPattern=[framework] %d - %c
-%-4r [%t] %-5p %c %x - %m%n
# Use this layout for LogFactor 5 analysis
# ???????
log4j.appender.ROLLING_FILE=org.apache.log4j.RollingFileAppender
log4j.appender.ROLLING_FILE.Threshold=ERROR
log4j.appender.ROLLING_FILE.File=rolling.log
//????,??????${java.home}?rolling.log
log4j.appender.ROLLING_FILE.Append=true //true\:?? false\:??
log4j.appender.ROLLING_FILE.MaxFileSize=10KB //??????
log4j.appender.ROLLING_FILE.MaxBackupIndex=1 //???
log4j.appender.ROLLING_FILE.layout=org.apache.log4j.PatternLayout
log4j.appender.ROLLING_FILE.layout.ConversionPattern=[framework] %d -
%c -%-4r [%t] %-5p %c %x - %m%n

#???socket
log4j.appender.SOCKET=org.apache.log4j.RollingFileAppender
log4j.appender.SOCKET.RemoteHost=localhost
log4j.appender.SOCKET.Port=5001
```

```
log4j.appender.SOCKET.LocationInfo=true
# Set up for Log Facter 5
log4j.appender.SOCKET.layout=org.apache.log4j.PatternLayout
log4j.appender.SOCET.layout.ConversionPattern=[start]%d{DATE}[DATE]%n
%p[PRIORITY]%n%x[NDC]%n%t[thread]%n%c[CATEGORY]%n%m[MESSAGE]%n%n

# Log Factor 5 Appender
log4j.appender.LF5_APPENDER=org.apache.log4j.lf5.LF5Appender
log4j.appender.LF5_APPENDER.MaxNumberOfRecords=2000
# ???????
log4j.appender.MAIL=org.apache.log4j.net.SMTPAppender
log4j.appender.MAIL.Threshold=FATAL
log4j.appender.MAIL.BufferSize=10
log4j.appender.MAIL.From=web@www.wuset.com
log4j.appender.MAIL.SMTPHost=www.wusetu.com
log4j.appender.MAIL.Subject=Log4J Message
log4j.appender.MAIL.To=web@www.wusetu.com
log4j.appender.MAIL.layout=org.apache.log4j.PatternLayout
log4j.appender.MAIL.layout.ConversionPattern=[framework] %d - %c -%-4r
[%t] %-5p %c %x - %m%n
# ?????
log4j.appender.DATABASE=Log4jTest.JDBCPoolAppender
#log4j.appender.DATABASE.URL=jdbc\:mysql\://localhost\:3306/test
#log4j.appender.DATABASE.driver=com.mysql.jdbc.Driver
#log4j.appender.DATABASE.user=root
#log4j.appender.DATABASE.password=000000
log4j.appender.DATABASE.sql = INSERT INTO
dbcplog(occur_date,thread_name,cat,level,info,line,user)
values('%d{yyyy-MM-dd HH:mm:ss}','%t','%c','%p','%m','%L',
log4j.appender.DATABASE.layout=org.apache.log4j.PatternLayout
log4j.appender.DATABASE.layout.ConversionPattern=[framework] %d - %c
-%-4r [%t] %-5p %c %x - %m%n %L

log4j.appender.A1=org.apache.log4j.DailyRollingFileAppender
log4j.appender.A1.File=SampleMessages.log4j
log4j.appender.A1.DatePattern=yyyyMMdd-HH?.log4j?
log4j.appender.A1.layout=org.apache.log4j.xml.XMLLayout
#???Appender
log4j.appender.im = net.cybercorlin.util.logger.appender.IMAppender
log4j.appender.im.host = mail.cybercorlin.net
log4j.appender.im.username = username
log4j.appender.im.password = password
log4j.appender.im.recipient = corlin@cybercorlin.net
log4j.appender.im.layout=org.apache.log4j.PatternLayout
```

```
log4j.appender.im.layout.ConversionPattern =[framework] %d - %c -%-4r
[%t] %-5p %c %x - %m%n
```