log4j API提供 org.apache.log4j.jdbc.JDBCAppender 对象，它能够将日志信息在指定的数据库。

JDBCAppender 配置:

Property	描述
bufferSize	设置缓冲区的大小。默认大小为1
driver	设置驱动程序类为指定的字符串。如果没有指定驱动程序类，默认为sun.jdbc.odbc.JdbcOdbcDriver
layout	设置要使用的布局。默认布局是org.apache.log4j.PatternLayout
password	Sets the database password.
sql	指定SQL语句在每次记录事件发生的时间执行。这可能是INSERT，UPDATE或DELETE
URL	设置JDBC URL
user	设置数据库用户名

日志表配置：

开始使用基于JDBC日志，要创建在哪里保存日志信息的表。下面是创建日志表的SQL语句：

```
CREATE TABLE LOGS
    (USER_ID VARCHAR(20) NOT NULL,
     DATED   DATE NOT NULL,
     LOGGER  VARCHAR(50) NOT NULL,
     LEVEL   VARCHAR(10) NOT NULL,
     MESSAGE VARCHAR(1000) NOT NULL
    );
```

配置文件示例：

以下是将用于将消息记录到一个日志表中的示例配置文件 log4j.properties的JDBCAppender

```
# Define the root logger with appender file
log4j.rootLogger = DEBUG, DB

# Define the DB appender
log4j.appender.DB=org.apache.log4j.jdbc.JDBCAppender

# Set JDBC URL
log4j.appender.DB.URL=jdbc:mysql://localhost/DBNAME

# Set Database Driver
log4j.appender.DB.driver=com.mysql.jdbc.Driver
```

```
# Set database user name and password
log4j.appender.DB.user=user_name
log4j.appender.DB.password=password

# Set the SQL statement to be executed.
log4j.appender.DB.sql=INSERT INTO LOGS
                      VALUES('%x','%d','%C','%p','%m')

# Define the layout for file appender
log4j.appender.DB.layout=org.apache.log4j.PatternLayout
```

这里使用的是MySQL数据库，必须要使用实际DBNAME，用户ID和在其中创建的日志表的数据库密码。SQL语句是使用日志表名和输入值到表，需要执行INSERT语句。

JDBCAppender不需要明确定义的布局。相反，使用PatternLayout 传递给它 SQL语句

如果想拥有相当于上述log4j.properties文件的XML配置文件，可以参考在这里的内容：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
<log4j:configuration>

<appender name="DB" class="org.apache.log4j.jdbc.JDBCAppender">
   <param name="url" value="jdbc:mysql://localhost/DBNAME"/>
   <param name="driver" value="com.mysql.jdbc.Driver"/>
   <param name="user" value="user_id"/>
   <param name="password" value="password"/>
   <param name="sql" value="INSERT INTO LOGS VALUES('%x',
                            '%d','%C','%p','%m')"/>
   <layout class="org.apache.log4j.PatternLayout">
   </layout>

</appender>

<logger name="log4j.rootLogger" additivity="false">
   <level value="DEBUG"/>
   <appender-ref ref="DB"/>
</logger>

</log4j:configuration>
```

示例程序：

下面的Java类是一个非常简单的Java应用程序使用Log4J日志库例子，初始化，然后使用。

```
import org.apache.log4j.Logger;
import java.sql.*;
import java.io.*;
```

```
import java.util.*;

public class log4jExample{
  /* Get actual class name to be printed on */
  static Logger log = Logger.getLogger(
                      log4jExample.class.getName());

  public static void main(String[] args)
                throws IOException,SQLException{

     log.debug("Debug");
     log.info("Info");
  }
}
```

编译和运行：

下面是步骤编译并运行上述程序。确保进行编译和执行之前，适当地设置PATH和CLASSPATH。

所有的库应该在CLASSPATH以及log4j.properties文件应该在PATH可用。所以有以下几点：

- 创建log4j.properties如上图所示。
- 创建log4jExample.java如上图所示，并对其进行编译。
- 执行log4jExample二进制运行程序。

现在检查DBNAME数据库里面日志表，发现下面的条目（记录）：

```
mysql >  select * from LOGS;
+---------+------------+--------------+-------+---------+
| USER_ID | DATED      | LOGGER       | LEVEL | MESSAGE |
+---------+------------+--------------+-------+---------+
|         | 2010-05-13 | log4jExample | DEBUG | Debug   |
|         | 2010-05-13 | log4jExample | INFO  | Info    |
+---------+------------+--------------+-------+---------+
2 rows in set (0.00 sec)
```

注：此处X被用于产生该记录事件的线程相关联输出的NDC（嵌套诊断上下文）。使用NDC来区分客户的服务器端组件处理多个客户端。检查Log4J的手册以获取更多信息。