MANUAL DE CONFIGURACIÓN Y EJECUCIÓN DE LOG4JDBC PARA GENERAR EL ARCHIVO DE LOG DE SENTENCIAS SQL

1. Agregar librerías al proyecto.

En este caso de estudio se creó un *fork* de la librería log4jdbc para omitir la escritura de información basura y así obtener solo la información necesaria sobre cuáles son las sentencias SQL ejecutadas por la aplicación. Dicho fork se encuentra en el repositorio git de la cadena de transformación.

Log4jdbc hace uso de la librería *slf4j* para generar el archivo de log. Como implementación para slf4j se usó *log4j* versión 2.

Por lo tanto, es necesario agregar las siguientes librerías al *classpath* del proyecto:

Log4jdbc :

- log4jdbcfork- 0.0.1-SNAPSHOT.jar

Slf4j:

- slf4j-api-1.7.6.jar
- log4j-slf4j-impl-2.3.jar
- slf4j-log4j12-1.7.14.jar

Log4j:

- log4j-api-2.3.jar
- log4j-core-2.3.jar
- log4j-jcl-2.3.jar

Por ejemplo, si se usa Maven:

```
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>2.3</version>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.3</version>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-jcl</artifactId>
    <version>2.3</version>
</dependency>

<dependency>
        <groupId>org.log4jdbcfork</groupId>
        <artifactId>log4jdbcfork</artifactId>
        <version>0.0.1-SNAPSHOT</version>
         <scope>test</scope>
    </dependency>
```

```
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.6</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-slf4j-impl</artifactId>
    <version>2.3</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.14</version>
    <scope>test</scope>
</dependency>
```

2. Configurar Driver.

Log4jdbc actúa como *wrapper* del driver JDBC real. Así que es necesario configurar log4jdbc como driver principal de la aplicación. Para esto, es necesario ir al punto en donde se configura el *datasource* del ambiente a ejecutar y establecer como *driverClassName* la clase ***net.sf.log4jdbc.DriverSpy.*** Luego, en la URL configurada en el datasource es necesario reemplazar la palabra *jdbc* por *jdbc:log4jdbc.*

Por ejemplo:

- Si el datasource es declarado como recurso JNDI en Glassfish (archivo glassfish-resources.xml):

glassfish-resources.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE resources PUBLIC
    "-//GlassFish.org//DTD GlassFish Application Server 3.1 Resource Definitions//EN"
    "http://glassfish.org/dtds/glassfish-resources_1_5.dtd">
<resources>
    <jdbc-resource pool-name="jdbc/sisinfo_pool"
        jndi-name="jdbc/sisinfo"/>

<jdbc-connection-pool name="jdbc/sisinfo_pool"
            datasource-classname="org.apache.commons.dbcp.BasicDataSource"
             res-type="javax.sql.DataSource"
             driver-classname="net.sf.log4jdbc.DriverSpy"    >
        <property name="url" value="jdbc:log4jdbc:postgresql://localhost:5432/sisinfo"/>
        <property name="username" value="sisinfoxx"/>
        <property name="password" value="sisinfoxxx"/>
        <property name="databaseName" value="sisinfo"/>
        <property name="driverClassName" value="net.sf.log4jdbc.DriverSpy"/>
    </jdbc-connection-pool>

</resources>
```

- Si el datasource es declarado como recurso JNDI en Tomcat o como recurso en la misma aplicación web (archivo web.xml o META-INF/context.xml):

```
<Resource
        name="jdbc/example"
        type="javax.sql.DataSource"
        factory="org.apache.tomcat.dbcp.dbcp.BasicDataSourceFactory"
        driverClassName="net.sf.log4jdbc.DriverSpy"
        url="jdbc:log4jdbc:postgresql://localhost:5432/example"
        username="postgres" password="***"
        maxActive="20" maxIdle="10" maxWait="-1"
        removeAbandoned="true" removeAbandonedTimeout="120" logAbandoned="true"
        auth="Container"
        charset="UTF-8" />
```

- Si se usa el framework Spring:

```
<bean id="dataSourceVotaciones" class = "com.mchange.v2.c3p0.ComboPooledDataSource" destroy-method="close">
    <property name="driverClass" value="net.sf.log4jdbc.DriverSpy" />
    <property name="jdbcUrl" value="jdbc:log4jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=rac-sc
    <property name="user" value="user" />
    <property name="password" value="pass" />
</bean>
```

- Si se utiliza JDBC de manera nativa:

```
Class.forName("net.sf.log4jdbc.DriverSpy");
Connection conn = DriverManager.getConnection(
        "jdbc:log4jdbc:derby://localhost:1527/sample",
        "user",
        "pass");
```

3. Configurar archivo de log a generar.

Como se mencionó anteriormente, se usa *log4j2* como implementación de *slf4j* para la generación del archivo de log. Por lo tanto, es necesario crear el archivo *log4j2.xml* con la siguiente información:

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="off">
    <Appenders>
        <RollingFile name="fileWriter"
                     fileName="D:\\LOGS_APP\\log-sql-sisinfo.log"
                    filePattern="D:\\LOGS_APP\\log-sql-sisinfo-%d{yyyy-MM-dd}-%i.log">
          <PatternLayout>
                      <pattern>%m%n</pattern>
                </PatternLayout>
          <Policies>
                              <SizeBasedTriggeringPolicy size="20 MB" />
                              <TimeBasedTriggeringPolicy />
                 </Policies>
        </RollingFile>
    </Appenders>
    <Loggers>
        <Root level="off">
            <AppenderRef ref="fileWriter"/>
        </Root>

         <Logger name="log4j.logger.jdbc.audit" level="debug" additivity="false">
              <AppenderRef ref="fileWriter"/>
           </Logger>

        <Logger name="jdbc.audit" level="debug" additivity="false">
             <AppenderRef ref="fileWriter"/>
           </Logger>
           <Logger name="jdbc.connection" level="off" additivity="false">
             <AppenderRef ref="fileWriter"/>
           </Logger>
           <Logger name="log4j.logger.jdbc.connection" level="off" additivity="false">
             <AppenderRef ref="fileWriter"/>
           </Logger>

              <logger name="log4jdbc.log4j2" level="debug" additivity="false">
                      <MarkerFilter marker="LOG4JDBC_AUDIT" onMatch="ACCEPT"
onMismatch="DENY"/>
                  <appender-ref ref="fileWriter"/>
              </logger>
    </Loggers>
</Configuration>
```

Cambiar la ruta establecida en las propiedades *fileName y filePattern.* El archivo *log4j2.xml* debe agregarse al classpath de la aplicación.

4. Ejecución.

Antes de ejecutar el ambiente configurado es necesario establecer la propiedad de sistema **log4jdbc.debug.stack.prefix** para indicar el paquete raíz en donde se encuentra todas las clases de negocio de la aplicación. Para establecer dicha propiedad es necesario agregar la siguiente línea como argumento de la máquina virtual:

-Dlog4jdbc.debug.stack.prefix=<*paquete raíz aplicación*>

Donde:

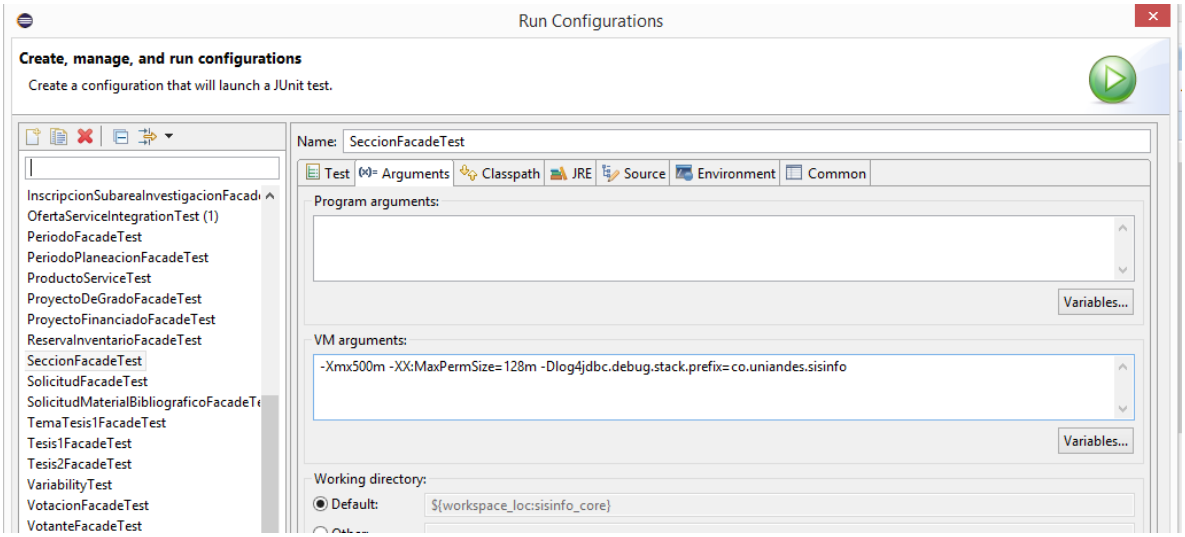
- <*paquete raíz aplicación*>: Corresponde al paquete de código Java donde está ubicado toda la clase de negocio de la aplicación. No es necesario indicar todo el paquete completo sino solo el elemento raís. Ejemplos:
    - -Dlog4jdbc.debug.stack.prefix=co
    - -Dlog4jdbc.debug.stack.prefix=co.uniandes.sisinfo
    - -Dlog4jdbc.debug.stack.prefix=co.edu

Por ejemplo:

- Si se ejecutan pruebas unitarias desde Maven:

```
>mvn test  -Dlog4jdbc.debug.stack.prefix=co.uniandes.sisinfo
```

- Si se ejecutan pruebas unitarias desde Eclipse:



- Si se ejecutan pruebas funcionales o en modo producción es necesario configurar la variable en el script de arranque del servidor o en la consola de administración del mismo.

Por ejemplo, si se usa Tomcat es necesario agregar la variable en el archivo TOMCAT_HOME/bin/catalina.bat (o catalina.sh si es Linux):

```
rem                   Example (all one line)
rem                   set TITLE=Tomcat.Cluster#1.Server#1 [%DATE% %TIME%]
rem -----------------------------------------------------------------------------


JAVA_OPTS= %JAVA_OPTS% -Dlog4jdbc.debug.stack.prefix=co.edu


rem Guess CATALINA_HOME if not defined
set "CURRENT_DIR=%cd%"
if not "%CATALINA_HOME%" == "" goto gotHome
set "CATALINA_HOME=%CURRENT_DIR%"
if exist "%CATALINA_HOME%\bin\catalina.bat" goto okHome
```

Luego, se ejecuta el ambiente configurado. El archivo de log se creará en la ruta definida en el punto 3.

```
prepareStatement(insert into accionVencida (accion, comando, fechaAcordada, fechaEjecucion, infoAdicional, modulo, proceso, usuario, id) values (?, ?, ?,
prepareStatement(select accionvenc0_.id as id1_44_0_, accionvenc0_.accion as accion2_44_0_, accionvenc0_.comando as comando3_44_0_, accionvenc0_.fechaAco
prepareStatement(update accionVencida set accion=?, comando=?, fechaAcordada=?, fechaEjecucion=?, infoAdicional=?, modulo=?, proceso=?, usuario=? where id
prepareStatement(select accionvenc0_.id as id1_44_0_, accionvenc0_.accion as accion2_44_0_, accionvenc0_.comando as comando3_44_0_, accionvenc0_.fechaAco
prepareStatement(delete from accionVencida where id=?) clase   (AccionVencidaFacade.java)

```

Comentarios:

- Si se realizan pruebas unitarias para generar el archivo, se recomienda que las pruebas queden en un paquete que comience con el nombre *test.* Lo anterior para que no haya conflicto al establecer si la clase es de negocio.