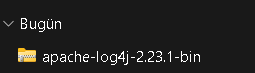
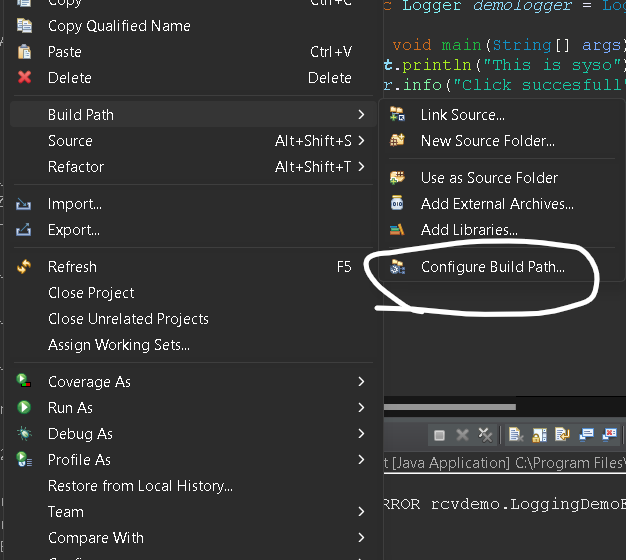
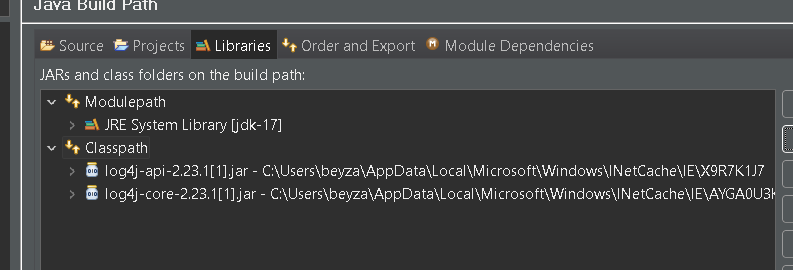
**LOG4J**

* I started the LOG4J assignment by adding the LOG4J library to Eclipse.
* Below are the steps for adding LOG4J.





* Click on Classpath, then click Add External JARs.., and select the api and core paths to add them. After these steps, open the package and add a class.
* Test codes have been written in the class. The codes and their output are below.

package rcvdemo;

import org.apache.logging.log4j.\*;

public class LoggingDemoFirst {

private static Logger *demologger* = LogManager.*getLogger*(LoggingDemoFirst.class.getName());

public static void main(String[] args) {

System.***out***.println("This is syso");

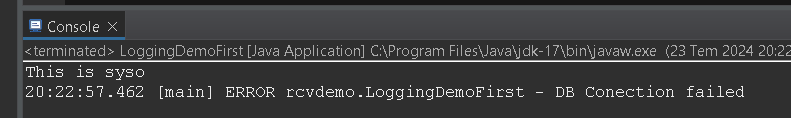
*demologger*.info("Click succesfull");

*demologger*.error("DB Conection failed");

*demologger*.debug("This is debug");

}

}

**OUTPUT:**

**LOG4J Examined as XML:**

* After creating a file named Source, an XML file was created. The codes and their outputs are given below.

**xml file:**

<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**Configuration** status=*"WARN"*>

<**Appenders**>

<**Console** name=*"Console"* target=*"SYSTEM\_OUT"*>

<**PatternLayout** pattern=*"%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"*/>

</**Console**>

</**Appenders**>

<**Loggers**>

<**Root** level=*"trace"*>

<**AppenderRef** ref=*"Console"*/>

</**Root**>

</**Loggers**>

</**Configuration**>

package rcvdemo;

import org.apache.logging.log4j.\*;

public class LoggingDemoFirst {

private static Logger *demologger* = LogManager.*getLogger*(LoggingDemoFirst.class.getName());

public static void main(String[] args) {

System.***out***.println("This is syso");

*demologger*.info("Click succesfull");

*demologger*.error("DB Conection failed");

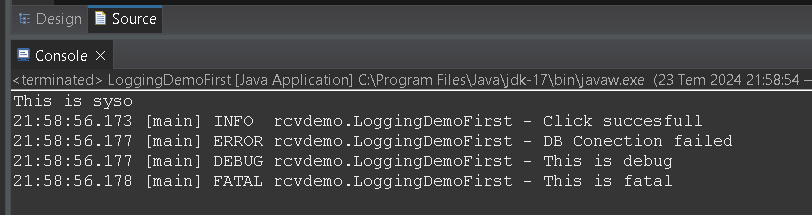
*demologger*.debug("This is debug");

*demologger*.fatal("This is fatal");

}

}

**OUTPUT:**

****

**1.A class named myTimerLoggings.java will be defined. The class will do the following:**

**a. It will operate at 3 different log levels (e.g., Debug, Info, Error).**

**b. In Debug, it will show the time as 20:27:00, 20:27:01, 20:27:02 (incrementing by seconds).**

**c. In Info, it will show the time as 20:28:00, 20:29:00, 20:30:00 (incrementing by minutes).**

**d. In Error, it will show the time as 21:00:00, 22:00:00, 23:00:00 (incrementing by hours).**

**e. Log files named Timer-day-month-year.log should be created every 1MB (Rolling).**

**f. All logs should be configured to be deleted after 1 day.**

**g. Logs will be written to a file.**

package rcvdemo;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import java.io.IOException;

import java.nio.file.\*;

import java.nio.file.attribute.BasicFileAttributes;

import java.time.LocalTime;

import java.util.concurrent.Executors;

import java.util.concurrent.ScheduledExecutorService;

import java.util.concurrent.TimeUnit;

public class myTimerLoggings {

private static final Logger ***logger*** = LogManager.*getLogger*(myTimerLoggings.class);

private static final Path ***LOG\_DIR*** = Paths.*get*("logs");

public static void main(String[] args) {

ScheduledExecutorService executorService = Executors.*newScheduledThreadPool*(4);

executorService.scheduleAtFixedRate(() -> ***logger***.debug("Current time: " + LocalTime.*now*()), 0, 1, *TimeUnit*.***SECONDS***);

executorService.scheduleAtFixedRate(() -> ***logger***.info("Current time: " + LocalTime.*now*().withSecond(0)), 0, 1, *TimeUnit*.***MINUTES***);

executorService.scheduleAtFixedRate(() -> ***logger***.error("Current time: " + LocalTime.*now*().withMinute(0).withSecond(0)), 0, 1, *TimeUnit*.***HOURS***);

executorService.scheduleAtFixedRate(() -> {

try {

Files.*walkFileTree*(***LOG\_DIR***, new SimpleFileVisitor<Path>() {

*@Override*

public *FileVisitResult* visitFile(Path file, BasicFileAttributes attrs) throws IOException {

if (Files.*isRegularFile*(file) && file.toString().endsWith(".log")) {

if (Files.*getLastModifiedTime*(file).toMillis() < System.*currentTimeMillis*() - *TimeUnit*.***DAYS***.toMillis(1)) {

Files.*delete*(file);

***logger***.info("Deleted old log file: " + file);

}

}

return *FileVisitResult*.***CONTINUE***;

}

});

} catch (IOException e) {

***logger***.error("Error cleaning up log files", e);

}

}, 0, 1, *TimeUnit*.***DAYS***); // Her gün bu görevi çalıştır

// Programı belirli bir süre sonra sonlandırma

new Thread(() -> {

try {

*TimeUnit*.***MINUTES***.sleep(1); // 1 dakika bekleyin

} catch (InterruptedException e) {

Thread.*currentThread*().interrupt();

} finally {

***logger***.info("Shutting down the logger");

executorService.shutdown();

try {

if (!executorService.awaitTermination(5, *TimeUnit*.***SECONDS***)) {

executorService.shutdownNow();

}

} catch (InterruptedException e) {

executorService.shutdownNow();

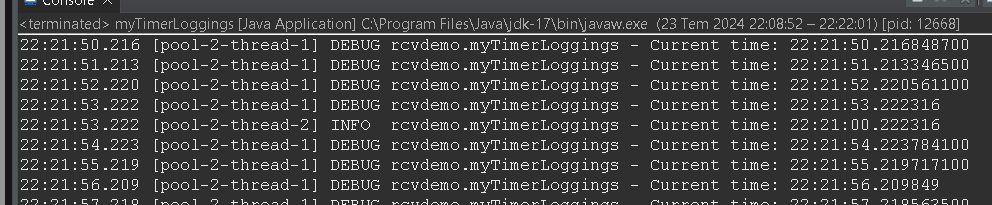
}

}

}).start();

}

}

**OUTPUT:**

**SOURCE**

1. <https://logging.apache.org/log4j/2.x/manual/configuration.html>
2. <https://www.youtube.com/watch?v=ViJxlwdL9m4>
3. <https://www.datamarket.com.tr/sozluk/log4j/>
4. <https://medium.com/ahmetdemirel-blog/test-otomasyon-s%C3%BCre%C3%A7lerinde-loglama-log4j-b24d0779e82c>
5. <https://coderistan.github.io/documents/log4j/ayarlama.html>