To migrate the Logback configuration to your Maven project, you need to add Logback and its configuration file to your project's pom.xml file. Here are the steps:

Add the Logback dependencies to your pom.xml file in the <dependencies> section:

<dependencies>

<!-- ... Other dependencies ... -->

<!-- Logback dependencies -->

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.6</version> <!-- Use the latest version -->

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-core</artifactId>

<version>1.2.6</version> <!-- Use the latest version -->

</dependency>

</dependencies>

Create a logback.xml file in your project's src/main/resources directory with the following content (the same content as previously provided):

xml

Copy code

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

<configuration>

<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">

<encoder>

<pattern>%logger{0}:%line %msg%n</pattern>

</encoder>

</appender>

<root level="INFO">

<appender-ref ref="STDOUT"/>

</root>

</configuration>

Save the logback.xml file in the src/main/resources directory of your project.

With these changes, your Maven project should include Logback and use the specified logback.xml configuration file for logging. You can now use SLF4J and Logback in your code as demonstrated earlier in the RoleController class.

porm.xml

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <groupId>org.example</groupId>  
 <artifactId>Tomcat\_web</artifactId>  
 <packaging>war</packaging>  
 <properties>  
 <maven.compiler.source>11</maven.compiler.source>  
 <maven.compiler.target>11</maven.compiler.target>  
 </properties>  
 <version>1.0-SNAPSHOT</version>  
 <name>Tomcat\_web Maven Webapp</name>  
 <url>http://maven.apache.org</url>  
 <dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>3.8.1</version>  
 <scope>test</scope>  
 </dependency>  
  
 *<!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->* <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>javax.servlet-api</artifactId>  
 <version>4.0.1</version>  
 <scope>provided</scope>  
 </dependency>  
 *<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->* <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>8.0.30</version>  
 </dependency>  
  
 *<!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->* <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>jstl</artifactId>  
 <version>1.2</version>  
 </dependency>  
 *<!-- https://mvnrepository.com/artifact/com.google.code.gson/gson -->* <dependency>  
 <groupId>com.google.code.gson</groupId>  
 <artifactId>gson</artifactId>  
 <version>2.10.1</version>  
 </dependency>  
 *<!-- Logback dependencies -->* <dependency>  
 <groupId>ch.qos.logback</groupId>  
 <artifactId>logback-classic</artifactId>  
 <version>1.2.6</version> *<!-- Use the latest version -->* </dependency>  
 <dependency>  
 <groupId>ch.qos.logback</groupId>  
 <artifactId>logback-core</artifactId>  
 <version>1.2.6</version> *<!-- Use the latest version -->* </dependency>  
 </dependencies>  
 <build>  
 <finalName>Tomcat\_web</finalName>  
 </build>  
</project>

RoleController.java

package api;  
  
import com.google.gson.Gson;  
import payload.BasicResponse;  
import services.RoleService;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
import java.io.PrintWriter;  
  
@WebServlet(name = "apiRoleController", urlPatterns = {"/api/role"})  
public class ApiRoleController extends HttpServlet {  
 private RoleService roleService = new RoleService();  
 private Gson gson = new Gson();  
 */\*  
 {  
 "statusCode": 200,  
 "message": ""  
 "data": true => dynamic data type  
 payload: request, response  
 \*/* @Override  
 protected void doDelete(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 System.*out*.println("Test DELETE");  
 int id = Integer.*parseInt*(req.getParameter("id"));  
 System.*out*.println("id = " + id);  
 boolean isSuccess = roleService.deleteRole(id);  
 BasicResponse basicResponse = new BasicResponse();  
 basicResponse.setStatusCode(200);  
 basicResponse.setMessage("");  
 basicResponse.setData(isSuccess);  
 String dataJSON = gson.toJson(basicResponse);  
  
 PrintWriter out = resp.getWriter();  
 resp.setContentType("application/json");  
 resp.setCharacterEncoding("UTF-8");  
  
 out.print(dataJSON);  
 out.flush();  
 }  
  
 @Override  
 protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 *// Do Servlet chưa support subpat /role/1 (DELETE) nên ta dùng parameter. Nếu muốn dùng  
 // subpath thì phải tự parse trong Servlet* System.*out*.println("Test GET");  
 }  
}

<https://stackoverflow.com/questions/29750470/why-doesnt-slf4j-logback-log-filename-and-row-number>

You should first make sure that you compile your application with debug information. If calling javac directly, then it is the -g command-line parameter. However most build tools (like maven or ant) have a specific way to provide this parameter.

You may also want to replace %F with %C, which should work without the debug information.

You only have to compile with the debug flags, you don't need to run the application in debug mode. See also [stackoverflow.com/questions/218033/…](http://stackoverflow.com/questions/218033/is-there-a-performance-difference-between-javac-debug-on-and-off)

– [Didier L](https://stackoverflow.com/users/525036/didier-l)

[Apr 21, 2015 at 8:06](https://stackoverflow.com/questions/29750470/why-doesnt-slf4j-logback-log-filename-and-row-number#comment47663812_29752845)

logback.xml

*<?*xml version="1.0" encoding="UTF-8"*?>*<configuration>  
 <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">  
 <encoder>  
 <pattern>[%F:%L] %logger %method %msg%n</pattern>  
 </encoder>  
 </appender>  
  
 <root level="INFO">  
 <appender-ref ref="STDOUT"/>  
 </root>  
</configuration>

With this configuration, the **%F** pattern will include the filename, **%L** will include the line number, **%method** will include the function name, and **%msg** will include the log message.

<https://auth0.com/blog/spring-boot-logs-aggregation-and-monitoring-using-elk-stack/>

<Pattern> %d{dd-MM-yyyy HH:mm:ss.SSS} [%thread] %-5level %logger{36}.%M - %msg%n </Pattern>

*<?*xml version="1.0" encoding="UTF-8"*?>*<configuration>  
 <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">  
 <encoder>  
 <pattern>[%F:%L] %logger.%method\(\) %msg%n</pattern>  
 </encoder>  
 </appender>  
  
 <root level="INFO">  
 <appender-ref ref="STDOUT"/>  
 </root>  
</configuration>

