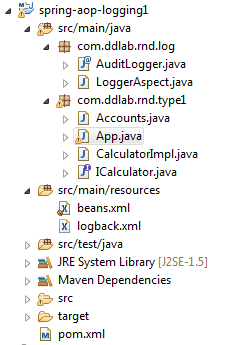
Logging in application using Spring AOP

You can log all information from the methods about the method parameters, return type, result and others without explicitly mentioning log.debug statements in the method. You can use Spring AspectJ AOP concept to track all information. It also can be used as instrumentation logging also. You can create your own annotation to hook all the classes where you want to print the log statements.

The code project is given below.

# **Project Structure**



# **Spring XML Configuration(beans.xml)**

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

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:aop=*"http://www.springframework.org/schema/aop"*

xsi:schemaLocation=*"http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.1.xsd*

*http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.2.xsd*

*http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd*

*http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd"*

xmlns:c=*"http://www.springframework.org/schema/c"* xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"* xmlns:mvc=*"http://www.springframework.org/schema/mvc"*

xmlns:p=*"http://www.springframework.org/schema/p"*>

<context:component-scan base-package=*"com.ddlab.rnd.spring.aop"* />

<aop:aspectj-autoproxy />

<bean class=*"com.ddlab.rnd.log.LoggerAspect"*></bean>

<bean id=*"calculator"* class=*"com.ddlab.rnd.type1.CalculatorImpl"*/>

<bean id=*"accounts"* class=*"com.ddlab.rnd.type1.Accounts"*/>

</beans>

# **SLF4J Configuration(logback.xml)**

<configuration>

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

<!-- encoders are assigned the type

ch.qos.logback.classic.encoder.PatternLayoutEncoder by default -->

<encoder>

<pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger - %msg%n</pattern>

</encoder>

</appender>

<root level=*"debug"*>

<appender-ref ref=*"STDOUT"* />

</root>

<logger name=*"org.springframework"* level=*"OFF"*/>

<logger name=*"org.apache"* level=*"OFF"*/>

</configuration>

# **Java Code**

# **LoggerAspect.java**

**package** com.ddlab.rnd.log;

**import** java.util.Date;

**import** org.apache.commons.beanutils.BeanUtils;

**import** org.aspectj.lang.ProceedingJoinPoint;

**import** org.aspectj.lang.annotation.Around;

**import** org.aspectj.lang.annotation.Aspect;

**import** org.aspectj.lang.annotation.Pointcut;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

@Aspect

**public** **class** LoggerAspect {

**protected** **static** Logger *log* = LoggerFactory.*getLogger*(LoggerAspect.**class**);

@Pointcut("within(@org.springframework.stereotype.Controller \*)")

**public** **void** allControllerBean() {

}

@Pointcut("within(@org.springframework.stereotype.Service \*)")

**public** **void** allServiceBean() {

}

@Pointcut("within(@org.springframework.stereotype.Repository \*)")

**public** **void** allRepositoryBean() {

}

@Pointcut("within(@com.ddlab.rnd.log.AuditLogger \*)")

**public** **void** otherBean() {

}

@Pointcut("execution(\* \*(..))")

**public** **void** methodPointcut() {

}

@Around("otherBean() && methodPointcut()")

**public** Object aroundCheckMethod(ProceedingJoinPoint joinPoint) **throws** Throwable {

*log*.debug("======================AOP Logging=========================");

*log*.debug("Class Name : "+joinPoint.getTarget());

*log*.debug("Method Signature : {}",joinPoint.getSignature());

Object[] methodArguments = joinPoint.getArgs();

**for**( Object obj : methodArguments )

*log*.debug("Method argument : {}",obj);

Date start = **new** Date();

Object result = joinPoint.proceed();

*log*.debug("Method return type : {}",BeanUtils.*describe*(result));

*log*.debug("Method output if exists : "+result);

Date end = **new** Date();

*log*.debug(">>>>>>>> end of {} took {} millisec",

joinPoint.getSignature(), end.getTime() - start.getTime());

*log*.debug("======================AOP Logging=========================");

**return** result;

}

//For others

@Around("allControllerBean() && methodPointcut()")

**public** Object aroundControllerMethod(ProceedingJoinPoint joinPoint)

**throws** Throwable {

*log*.debug(">>>>>>>> invoking {}", joinPoint.getSignature());

Date start = **new** Date();

Object result = joinPoint.proceed();

Date end = **new** Date();

*log*.debug(">>>>>>>> return of {} with {}", joinPoint.getSignature(),

BeanUtils.*describe*(result));

*log*.debug(">>>>>>>> end of {} take {} millisec",

joinPoint.getSignature(), end.getTime() - start.getTime());

**return** result;

}

@Around("allServiceBean() && methodPointcut()")

**public** Object aroundServiceMethod(ProceedingJoinPoint joinPoint)

**throws** Throwable {

*log*.debug(">>>>>>>> invoking {}", joinPoint.getSignature());

Date start = **new** Date();

Object result = joinPoint.proceed();

Date end = **new** Date();

*log*.debug(">>>>>>>> return of {} with {}", joinPoint.getSignature(),

BeanUtils.*describe*(result));

*log*.debug(">>>>>>> method arguments {} , {}", joinPoint.getArgs());

*log*.debug(">>>>>>>> end of {} take {} millisec",

joinPoint.getSignature(), end.getTime() - start.getTime());

**return** result;

}

@Around("allRepositoryBean() && methodPointcut()")

**public** Object aroundRepositoryMethod(ProceedingJoinPoint joinPoint)

**throws** Throwable {

*log*.debug(">>>>>>>> invoking {}", joinPoint.getSignature());

Date start = **new** Date();

Object result = joinPoint.proceed();

Date end = **new** Date();

*log*.debug(">>>>>>>> return of {} with {}", joinPoint.getSignature(),

BeanUtils.*describe*(result));

*log*.debug(">>>>>>>> end of {} take {} millisec",

joinPoint.getSignature(), end.getTime() - start.getTime());

**return** result;

}

}

# **AuditLogger.java(Annotation)**

**package** com.ddlab.rnd.log;

**public** **@interface** AuditLogger {

}

# **Core Business Logic**

# **ICalculator.java**

**package** com.ddlab.rnd.type1;

**public** **interface** ICalculator {

**int** add( **int** a , **int** b);

}

# **CalculatorImpl.java**

**package** com.ddlab.rnd.type1;

**import** com.ddlab.rnd.log.AuditLogger;

@AuditLogger

**public** **class** CalculatorImpl **implements** ICalculator {

**public** **int** add(**int** a, **int** b) {

**return** a+b;

}

}

# **Accounts.java**

**package** com.ddlab.rnd.type1;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** com.ddlab.rnd.log.AuditLogger;

@AuditLogger

**public** **class** Accounts {

@Autowired

**private** ICalculator calculator;

**public** **void** setCalculator(ICalculator calculator) {

**this**.calculator = calculator;

}

**public** **int** getCalculation( **int** a , **int** b ) {

**return** calculator.add(a, b);

}

}

# **App.java(Test)**

**package** com.ddlab.rnd.type1;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** App {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("beans.xml");

Accounts accounts = (Accounts) context.getBean("accounts");

**int** result = accounts.getCalculation(10, 15);

System.***out***.println("Result :::" + result);

}

}

# **Output**

22:46:35.280 [main] DEBUG com.ddlab.rnd.log.LoggerAspect - ======================AOP Logging=========================

22:46:35.282 [main] DEBUG com.ddlab.rnd.log.LoggerAspect - Class Name : com.ddlab.rnd.type1.Accounts@47bd5719

22:46:35.283 [main] DEBUG com.ddlab.rnd.log.LoggerAspect - Method Signature : int com.ddlab.rnd.type1.Accounts.getCalculation(int,int)

22:46:35.283 [main] DEBUG com.ddlab.rnd.log.LoggerAspect - Method argument : 10

22:46:35.283 [main] DEBUG com.ddlab.rnd.log.LoggerAspect - Method argument : 15

22:46:35.305 [main] DEBUG com.ddlab.rnd.log.LoggerAspect - ======================AOP Logging=========================

22:46:35.305 [main] DEBUG com.ddlab.rnd.log.LoggerAspect - Class Name : com.ddlab.rnd.type1.CalculatorImpl@956ff4d

22:46:35.305 [main] DEBUG com.ddlab.rnd.log.LoggerAspect - Method Signature : int com.ddlab.rnd.type1.ICalculator.add(int,int)

22:46:35.305 [main] DEBUG com.ddlab.rnd.log.LoggerAspect - Method argument : 10

22:46:35.305 [main] DEBUG com.ddlab.rnd.log.LoggerAspect - Method argument : 15

22:46:35.335 [main] DEBUG com.ddlab.rnd.log.LoggerAspect - Method return type : {class=class java.lang.Integer}

22:46:35.335 [main] DEBUG com.ddlab.rnd.log.LoggerAspect - Method output if exists : 25

22:46:35.336 [main] DEBUG com.ddlab.rnd.log.LoggerAspect - >>>>>>>> end of int com.ddlab.rnd.type1.ICalculator.add(int,int) took 31 millisec

22:46:35.336 [main] DEBUG com.ddlab.rnd.log.LoggerAspect - ======================AOP Logging=========================

22:46:35.336 [main] DEBUG com.ddlab.rnd.log.LoggerAspect - Method return type : {class=class java.lang.Integer}

22:46:35.336 [main] DEBUG com.ddlab.rnd.log.LoggerAspect - Method output if exists : 25

22:46:35.336 [main] DEBUG com.ddlab.rnd.log.LoggerAspect - >>>>>>>> end of int com.ddlab.rnd.type1.Accounts.getCalculation(int,int) took 53 millisec

22:46:35.336 [main] DEBUG com.ddlab.rnd.log.LoggerAspect - ======================AOP Logging=========================

Result :::25

# **Maven Configuration(pom.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/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>spring-aop-logging1</groupId>

<artifactId>spring-aop-logging1</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-aop-logging1</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<spring.version>3.2.1.RELEASE</spring.version>

<spring.security.version>3.1.3.RELEASE</spring.security.version>

<spring.security.oauth.version>1.0.2.RELEASE</spring.security.oauth.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-web</artifactId>

<version>${spring.security.version}</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-core</artifactId>

<version>${spring.security.version}</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-config</artifactId>

<version>${spring.security.version}</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-taglibs</artifactId>

<version>${spring.security.version}</version>

</dependency>

<dependency>

<groupId>org.springframework.security.oauth</groupId>

<artifactId>spring-security-oauth2</artifactId>

<version>${spring.security.oauth.version}</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjrt</artifactId>

<version>1.6.11</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.6.11</version>

</dependency>

<dependency>

<groupId>cglib</groupId>

<artifactId>cglib</artifactId>

<version>2.2.2</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>jcl-over-slf4j</artifactId>

<version>1.6.6</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>jul-to-slf4j</artifactId>

<version>1.6.6</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.6.6</version>

</dependency>

<dependency>

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

<artifactId>logback-classic</artifactId>

<version>1.0.9</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.5</version>

<scope>provided</scope>

</dependency>

<!-- Hibernate Validator -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-validator</artifactId>

<version>4.2.0.Final</version>

</dependency>

<dependency>

<groupId>commons-beanutils</groupId>

<artifactId>commons-beanutils</artifactId>

<version>1.8.3</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.21</version>

</dependency>

</dependencies>

</project>