BackEnd

Course 1

spring beans/context/wiring/autowiring

IOD, DI

Spring MVC

web.xml/dispatcher-servlet.xml

java configurations

Controllers / Service / DAO view+data

Hibernate

xml config

sessionfactory

db config

model annotations

Course 2

Springboot application using initilizr

lombok

RESTAPI (RestController) response

CRUDRepo / JpaRepo data+status

MongoDB

MongoRepo

Custom method / Custom queries in repo layer

Custom exceptions

JWT (two applications)

Testing : repo / service / controller (Junit/Mockito)

Logger

Course 3

CI

V-Lab (Linux)

Dockerization (running applications in containers)

Microservice

gateway

Discovery (Eureka)

Circuit breaker (Hystrix)

PE

Course 4

Feignclient 10

RabbitMq (QMS) basic

Pub/Sub 30

ng <-> be <-> db

email service

SE Pre-project task Capstone project

Steps to add loggers to existing application

Step 1 Create one springboot application with complete flow

sprint4(mongo crud)

Step 2 add required dependencies

```
<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-aop -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-aop</artifactId>
</dependency>

<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjweaver -->
<dependency>
  <groupId>org.aspectj</groupId>
  <artifactId>aspectjweaver</artifactId>
  <version>1.9.7</version>
</dependency>
```

Step 3 create aspect class

create objecet Logger

```
1 usage
@Aspect
@Component // @Component vs @Bean // final object // final parameter // final variable // final method // final class
public class ControllerLoggingAspect {
    8 usages
    private static final Logger logger = LoggerFactory.getLogger(ControllerLoggingAspect.class);
```

Step 4 Define pointcut

```
    // pointcut
    // CustomerController.addCustomer()

    @Pointcut("execution(* com.stackroute.mongocrud.demo1.controllers.CustomerController.addCustomer(..))")
    public void controlMethod(){}
```

Step 5 Define advisors

```
24      @Before("controlMethod()")
25  @   public void beforeAdvice(JoinPoint joinpoint){
26          // which info to be logged before above method execute
27          logger.info("----- Before controller method -----");
28          logger.debug("Method name :: " + joinpoint.getSignature().getName());
29          logger.debug("Args :: " + Arrays.toString(joinpoint.getArgs()) );
30          logger.info("---#########################----");
31
32      }
33
34      @After("controlMethod()")
35  @   public void afterAdvice(JoinPoint joinpoint){
36          // which info to be logged after completing above method execution
37          logger.info("----- After controller method -----");
38          logger.debug("Method name :: " + joinpoint.getSignature().getName());
39          logger.debug("Args :: " + Arrays.toString(joinpoint.getArgs()) );
40          logger.info("---#########################----");
41      }
```

```
42      @AfterReturning(value = "controlMethod()", returning = "result")
43  @   public void afterReturnAdvice(JoinPoint joinpoint, Object result){
44          // which info to be logged after completing above method execution
45          logger.info("----- After returning controller method -----");
46          logger.debug("Method name :: " + joinpoint.getSignature().getName());
47          logger.debug("Args :: " + Arrays.toString(joinpoint.getArgs()) );
48          logger.debug("Result :: "+result);
49          logger.info("---#########################----");
50      }
51
52      @AfterThrowing(value = "controlMethod()",throwing = "error")
53  @   public void afterThrowing(JoinPoint joinpoint, Throwable error) {
54          logger.info("----- After throwing exception by controller method -----");
55          logger.debug("Method name :: " + joinpoint.getSignature().getName());
56          logger.debug("Args :: " + Arrays.toString(joinpoint.getArgs()) );
57          logger.debug("Exception :: " + error);
58          logger.info("---#########################----");
59      }
```

Step 6 Define logback.xml under resources

```xml
<configuration>
    <property name="LOG_FILE_LOCATION" value="logs"></property>
    <property name="LOG_FILE_NAME" value="Customer"></property>
    <property name="LOG_FILE_EXTENSION" value=".log"></property>

    <timestamp key="bySecond" datePattern="yyyyMMdd'T'HHmmss" timeReference="contextBirth"></timestamp>

    <appender name="FILE_AUDIT" class="ch.qos.logback.core.FileAppender">
        <file>${LOG_FILE_LOCATION}/${LOG_FILE_NAME}_${bySecond}${LOG_FILE_EXTENSION}</file>
        <encoder>
            <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n </pattern>
        </encoder>
    </appender>

    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n </pattern>
        </encoder>
    </appender>

    <root level="debug">
        <appender-ref ref="FILE_AUDIT"/>
        <appender-ref ref="STDOUT"/>
    </root>

</configuration>
```

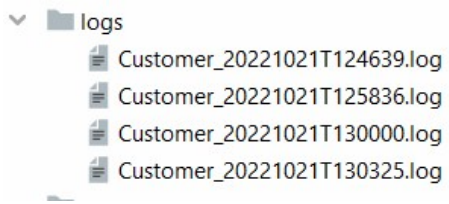
How to check

Run application

makesure monitored method is called

observe console window
observe log file

```
t - ----- Before controller method -----
t - Method name :: addCustomer
t - Args :: [Customer(customerId=CUST00007, name
t - ---######################----
 - findOne using query: { "customerId" : "CUST0(
 - findOne using query: { "_id" : "CUST00007"} t
 [connectionId{localValue:3, serverValue:186}] t
mand '{"find": "customer", "filter": {"_id": "CU
f command with request id 7 completed successfu1
1 documents with cursorId 0 from server localhos
t - ----- After throwing exception by controller
t - Method name :: addCustomer
t - Args :: [Customer(customerId=CUST00007, name
t - Exception :: com.stackroute.mongocrud.demo1.
t - ---######################----
t - ----- After controller method -----
t - Method name :: addCustomer
t - Args :: [Customer(customerId=CUST00007, name
t - ---######################----
olved [com.stackroute.mongocrud.demo1.exceptions
```



Note

Loggers effect application performance

Custom exception with custom message

```
11 usages
@ResponseStatus(code= HttpStatus.CONFLICT, reason="Customer already exists")
public class CustomerAlreadyExistingException extends Exception {
    4 usages
    public CustomerAlreadyExistingException() { super("Customer already exists"); }
}
```

```
    @AfterThrowing(value = "controlMethod()",throwing = "error")
    public void afterThrowing(JoinPoint joinpoint, Throwable error) {
        logger.info("----- After throwing exception by controller method -----");
        logger.debug("Method name :: " + joinpoint.getSignature().getName());
        logger.debug("Args :: " + Arrays.toString(joinpoint.getArgs()) );
        logger.debug("Exception :: " + error);
        logger.debug("Exception message :: " + error.getMessage());
        logger.info("---#######################----");
    }
```