SPRING ASSIGNMENT Day_4

Harshit Kushmakar| 16896

1. Create a bean Calculator for doing the arithmetic operations like Add, Subtract, Divide and

Calculate. The methods must be overloaded to perform operations on all different data

types. The methods are as mentioned below:

a. int add(int num1, int num2)

b. double add(double num1, double num2)

c. String add(String str1, String str2)

d. int subtract(int num1, int num2)

e. double subtract(double num1, double num2)

f. int multiply(int num1, int num2)

g. double multiply(double num1, double num2)

h. int divide(int num1, int num2) throws ArithmeticException

i. double divide(double num1, double num2)

Create Pointcuts for all 4 operations and implement logging using spring AOP. The log must

maintain the values passed to the methods.

1. Implement AfterAdvice for add() methods.

2. Implement BeforeAdvice for subtract() methods

3. AfterReturningAdvice for multiply() methods

a. The logger logs the return value as well

4. AfterThrowingAdvice for divide() method which throws ArithmeticException

a. The logger logs the Error occurred and the reason for the same.

```java
package spring4;


import ComponentAnnotation.College;
import ComponentAnnotation.CollegeConfig;
import org.apache.log4j.LogManager;
import org.apache.log4j.Logger;
import org.springframework.context.ApplicationContext;
import
org.springframework.context.annotation.AnnotationConfigApplicationContext;


public class Main {
    private static final Logger log =
LogManager.getLogger(spring4.Main.class);
    public static void main(String[] args) {
        ApplicationContext context = new
AnnotationConfigApplicationContext(Calculator.class);
        Calculator calculator = context.getBean(Calculator.class);

       log.info( calculator.add(4,4));
       log.info(calculator.add(8,8));
        log.info(calculator.add(9,4));
        log.info(calculator.subtract(25,17));
        log.info(calculator.subtract(13,7));
        log.info(calculator.multiply(12,18));
        log.info(calculator.multiply(13,10));
        log.info(calculator.divide(210,21));
        log.info(calculator.divide(140,5));
    }
    }


package spring4;

public class Calculator {
    //add
    public int add(int num1, int num2) {
        return num1 + num2;
    }

    public double add(double num1, double num2) {
        return num1 + num2;
    }

    public String add(String str1, String str2) {
        return str1 + str2;
    }

    //subtract
    public int subtract(int num1, int num2) {
        return num1 + num2;
    }

    public double subtract(double num1, double num2) {
        return num1 + num2;
    }
      // Multiply
    public int multiply(int num1, int num2) {
        return num1 * num2;
    }
    public double multiply(double num1, double num2) {
        return num1 * num2;
    }
    //Divide
    public int divide(int num1, int num2) throws ArithmeticException {
        if (num2 == 0) {
            throw new ArithmeticException("Division by zero");
        }
        return num1 / num2;
    }

    public double divide(double num1, double num2) throws
ArithmeticException {
        if (num2 == 0) {
            throw new ArithmeticException("Division by zero");
        }
        return num1 / num2;
    }
}
```

```java
package spring4;

import org.apache.log4j.LogManager;
import org.apache.log4j.Logger;
import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.Aspect;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class AfterAdvice {
    private static final Logger log= LogManager.getLogger(Main.class);
    @After("execution(* spring4.Calculator.add(..))")
    public void logAfter(JoinPoint joinPoint) {
        log.info("Addition Done");
    }
}
```

```java
package spring4;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.AfterReturning;
import org.aspectj.lang.annotation.Aspect;
import org.springframework.stereotype.Component;

@Component
@Aspect
public class AfterReturningAdvice {
    @AfterReturning("execution(* org.example.Calculator.multiply(..))")
    public void logAfter(JoinPoint joinPoint) {
        System.out.println("Value returned succesfully");
    }
}
```

```java
package spring4;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.AfterThrowing;

public class AfterThrowingAdvice {
    @AfterThrowing("execution(* org.example.Calculator.divide(..))")
    public void logAfter(JoinPoint joinPoint) {
        System.out.println("thrown error");
    }
}
```

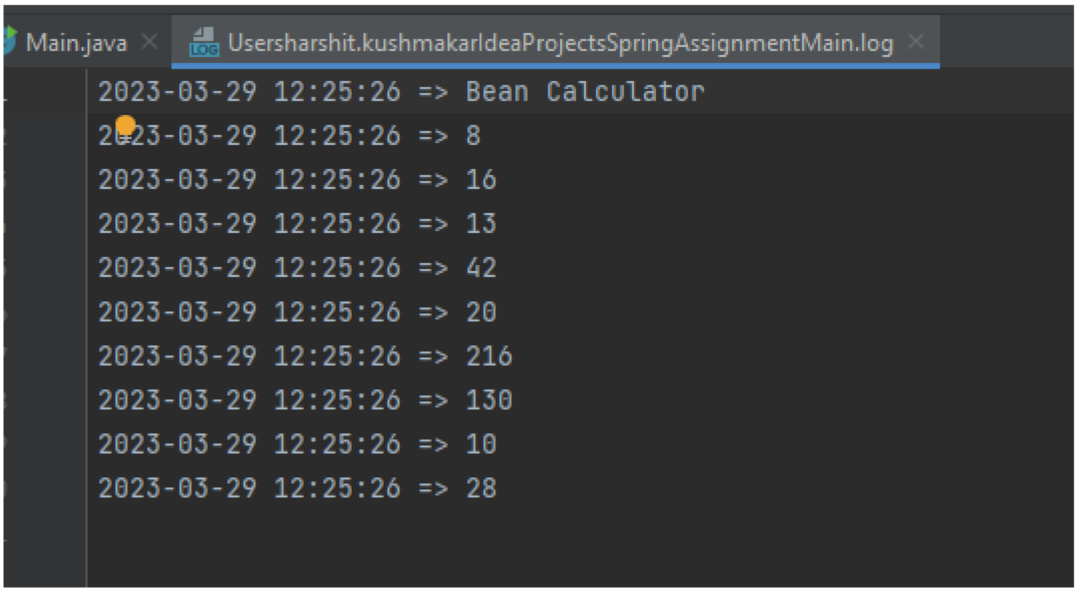
```java
package spring4;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class BeforeAdvice {

    @Before("execution(* org.example.Calculator.subtract(..))")
    public void beforeSubtract(JoinPoint joinPoint){
        System.out.println("Before Subtract");
    }
}
```

OUTPUT:



```
2023-03-29 12:25:26 => Bean Calculator
2023-03-29 12:25:26 => 8
2023-03-29 12:25:26 => 16
2023-03-29 12:25:26 => 13
2023-03-29 12:25:26 => 42
2023-03-29 12:25:26 => 20
2023-03-29 12:25:26 => 216
2023-03-29 12:25:26 => 130
2023-03-29 12:25:26 => 10
2023-03-29 12:25:26 => 28
```

2. Implement Logger with the help of Spring AOP in the application developed during Day 3 as mentioned below:

a. Use AroundAdvice while registering a customer. If the age of the customer is less than 21 years, the customer must not get registered, and the control of the program returns from the advice back to the calling method. If age is greater than or equal to 21 years, the customer gets registered.

b. Use BeforeAdvice while printing the list of all the customers.

c. Use AfterAdvice while updating existing customer record.

d. Use AfterThrowingAdvice while searching a particular customer by giving a customerId. If the customer is not available, the thrown exception along with the customer Id must be maintained in the log file.

e. Use AfterReturnAdvice in the same method. The returned customer's id must be maintained in the log file.

f. Use All the 5 advices while calling getCustomerByName() method.