Spring Boot - AOP After Advice

Last Updated : 23 Jul, 2025

Prerequisite: Aspect Oriented Programming and AOP in Spring Framework

Aspect-oriented programming(AOP) as the name suggests uses aspects in programming. It can be defined as the breaking of code into different modules, also known as modularisation, where the aspect is the key unit of modularity. Aspects enable the implementation of crosscutting concerns such as transaction, logging not central to business logic without cluttering the code core to its functionality. It does so by adding additional behavior that is the advice to the existing code. For example-Security is a crosscutting concern, in many methods in an application security rules can be applied, therefore repeating the code at every method, defining the functionality in a common class and control were to apply that functionality in the whole application. In this article, we will be covering a working example of After Advice.

After Advice is executed after a join point execution. It is denoted by **@After** annotation. The advice method is executed irrespective of the execution flow of the matched method. Whether the method is executed normally or any exception occurs after advice will be executed anyways. This is contrary to the execution of after-returning, where advice is run only if the method completes successfully, and after-throwing where advice is run only if the method exits by throwing an exception. After advice is generally used for releasing resources.

Steps to Implement AOP After Advice in Spring Boot Application

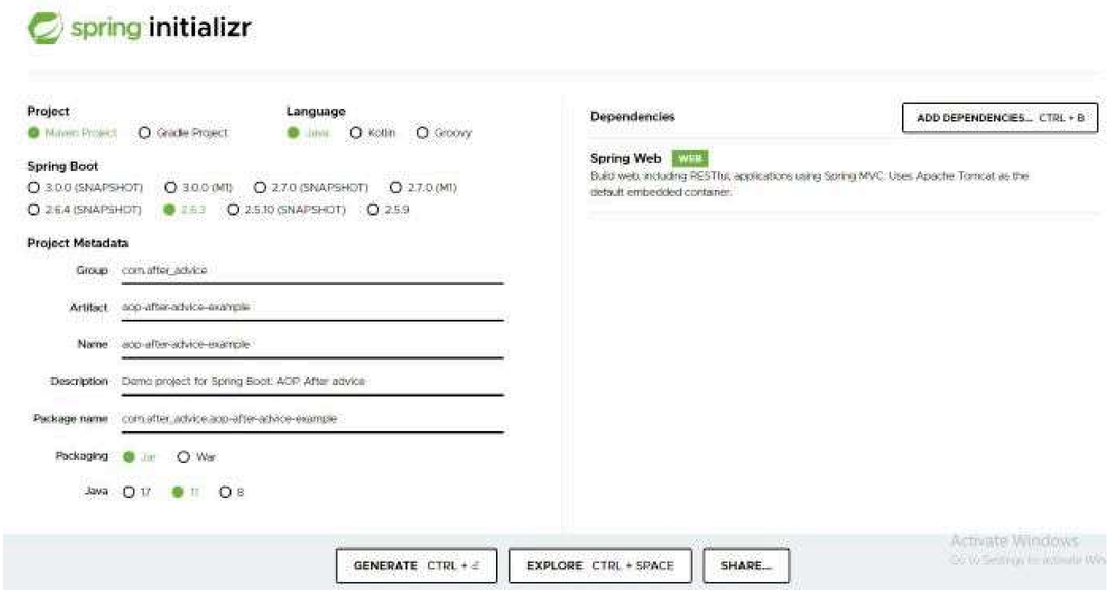
Step 1: Open Spring Initializer https://start.spring.io/

Step 2: Provide the **Group** name: com.after_advice

Step 3: Provide the **Artifact Id**: aop-after-advice-example

Step 4: Add the **Spring Web** dependency.

Step 5: Click on the **Generate** button. A zip file will be downloaded into the system. Extract it.


Project: [✓ Maven Project] Gradle Project
Language: [✓ Java] Kotlin Groovy
Spring Boot: 3.0.0 (SNAPSHOT) 3.0.0 (M1) 2.7.0 (SNAPSHOT) 2.7.0 (M1) 2.6.4 (SNAPSHOT) [✓ 2.6.3] 2.5.10 (SNAPSHOT) 2.5.9

Project Metadata

Group	com.after_advice
Artifact	aop-after-advice-example
Name	aop-after-advice-example
Description	Demo project for Spring Boot: AOP After advice
Package name	com.after_advice.aop-after-advice-example
Packaging	[✓ Jar] War
Java	17 [✓ 11] 8

Dependencies

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Step 6: Import the folder in the IDE by using the following steps:

File -> Import -> Existing Maven Projects -> Next -> Browse -> Look for the folder **aop-after-advice-example** -> Finish. When the project is imported, it will install the dependencies. Once it is done, follow the next steps.

Step 7: Add the dependency for spring AOP in pom.xml

pom.xml file:

```
<project xmlns="https://maven.apache.org/POM/4.0.0"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="https://maven.apache.org/POM/4.0.0
                             https://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>
 <groupId>com.before_advice</groupId>
 <artifactId>aop-before-advice-example</artifactId>
 <version>0.0.1-SNAPSHOT</version>

 <packaging>jar</packaging>
 <name>aop-before-advice-example</name>
 <description>Demo project for Spring Boot</description>

  <parent>
     <groupId>org.springframework.boot</groupId>
     <artifactId>spring-boot-starter-parent</artifactId>
     <version>2.2.2.RELEASE</version>
     <relativePath /> <!-- lookup parent from repository -->
  </parent>

  <properties>
     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
     <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
     <java.version>1.8</java.version>
  </properties>

    <dependencies>
       <!-- dependency for spring web -->
       <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
       </dependency>

       <!-- added dependency for spring aop -->
       <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-aop</artifactId>
        </dependency>
    </dependencies>

<build>
 <plugins>
  <plugin>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-maven-plugin</artifactId>
  </plugin>
 </plugins>
</build>

</project>
```

Save the changes and it will download the jars. Once it is done, move ahead with the next steps.

Note: *If the jars are not added properly, you may get some errors.*

Step 8: Create a package with the name com.after_advice.model. and add a Student model class to it.

Student class:

```java
package com.after_advice.model;

public class Student {
    private String firstName;
    private String secondName;

    public Student() {
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getSecondName() {
        return secondName;
    }

    public void setSecondName(String secondName) {
        this.secondName = secondName;
    }
}
```

Step 9: Create a package with the name com.after_advice.service. and add a Student Service class to it. Add a method to add students with given name arguments.

StudentService class:

```
package com.after_advice.service;

import org.springframework.stereotype.Service;

import com.after_advice.model.Student;

@Service
public class StudentService {

    public Student addStudent(String fname, String sname) {
        System.out.println("Add student service method called");
        Student stud = new Student();
        stud.setFirstName(fname);
        stud.setSecondName(sname);
        if(fname.length()<=3)
            throw new RuntimeException("Length of firstname must be 4 or
more" );
        return stud;
    }

}
```

Step 10: Create a package with the name com.after_advice.controller. and add a Student Controller class to it. Add a method to handle Get requests and call Student Service from it.

StudentController class:

```
package com.after_advice.controller;

import com.after_advice.model.Student;
import com.after_advice.service.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
```

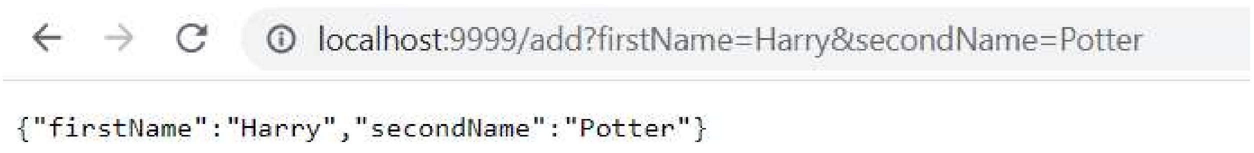
```
@RestController
public class StudentController {

    @Autowired private StudentService studentService;

    @GetMapping(value = "/add")
    public Student addStudent(
        @RequestParam("firstName") String firstName,
        @RequestParam("secondName") String secondName)
    {
        return studentService.addStudent(firstName,
                                         secondName);
    }
}
```

Step 11: Create a package with the name com.after_advice.aspect. and add a Student Service Aspect class to it. Here we will add our Advice method and PointCut expression.

StudentServiceAspect class:

```
package com.after_advice.aspect;

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

@Aspect
@Component
public class StudentServiceAspect {

    // the pointcut expression specifying execution of any
    // method in com.after_advice.service.StudentService
    // class of any return type with 0 or more number of
    // arguments
    @Pointcut("execution(* com.aftere_advice.service.StudentService.*(..))
")
      // the pointcut signature
    private void anyStudentService()  { }

    @After("anyStudentService() && args(fname, sname)")
    public void afterAdvice(JoinPoint joinPoint,
                            String fname, String sname)
    {
        System.out.println(
            "After method:" + joinPoint.getSignature()
            + "\n "
            + "Added Student with first name - " + fname
            + ", second name - " + sname);
    }
}
```

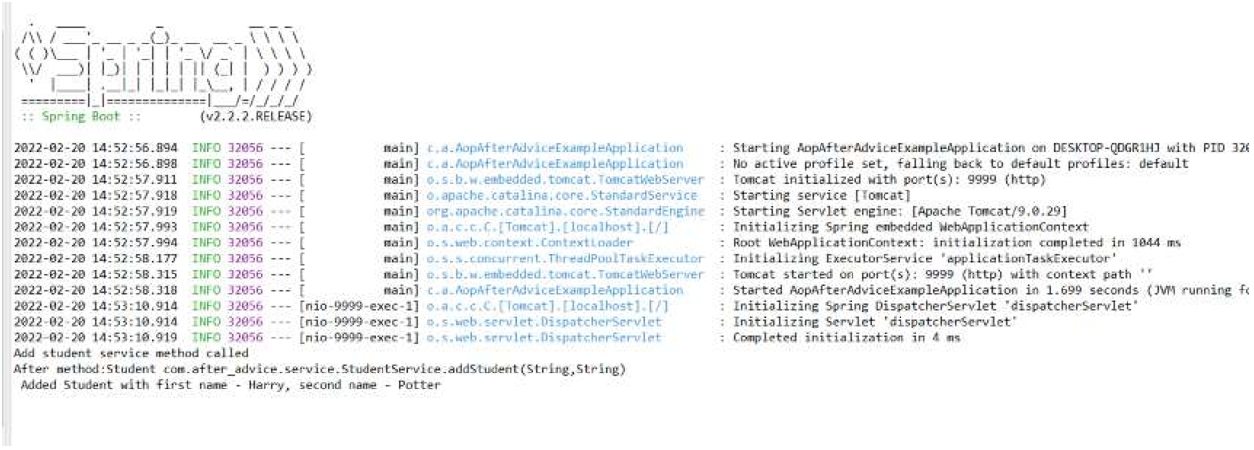
Step 12: We are done with the code structure. Now to run the application, start the application as "run as boot application". Open the browser and hit the following URL to make a get request call: http://localhost:{portNumber}/add?firstName={fname}&secondName={sname}



```
localhost:9999/add?firstName=Harry&secondName=Potter
```

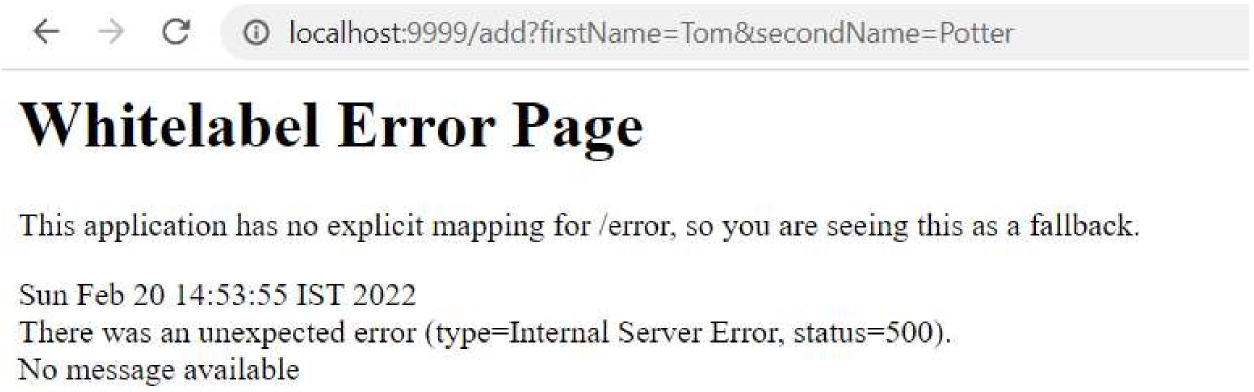
```
{"firstName":"Harry","secondName":"Potter"}
```

For a demo, we are hitting URL with fname as Harry and sname as Potter. In this case, the method will be executed normally.



```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.2.RELEASE)

2022-02-20 14:52:56.894  INFO 32056 --- [           main] c.a.AopAfterAdviceExampleApplication     : Starting AopAfterAdviceExampleApplication on DESKTOP-QDGR1HJ with PID 32
2022-02-20 14:52:56.898  INFO 32056 --- [           main] c.a.AopAfterAdviceExampleApplication     : No active profile set, falling back to default profiles: default
2022-02-20 14:52:57.911  INFO 32056 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 9999 (http)
2022-02-20 14:52:57.918  INFO 32056 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-02-20 14:52:57.919  INFO 32056 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.29]
2022-02-20 14:52:57.993  INFO 32056 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-02-20 14:52:57.994  INFO 32056 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 1044 ms
2022-02-20 14:52:58.177  INFO 32056 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2022-02-20 14:52:58.315  INFO 32056 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 9999 (http) with context path ''
2022-02-20 14:52:58.318  INFO 32056 --- [           main] c.a.AopAfterAdviceExampleApplication     : Started AopAfterAdviceExampleApplication in 1.699 seconds (JVM running f
2022-02-20 14:53:10.914  INFO 32056 --- [nio-9999-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-02-20 14:53:10.914  INFO 32056 --- [nio-9999-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2022-02-20 14:53:10.919  INFO 32056 --- [nio-9999-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 4 ms
Add student service method called
After method:Student com.after_advice.service.StudentService.addStudent(String,String)
 Added Student with first name - Harry, second name - Potter
```

When we hit URL with fname as Tom, the service method will throw an exception. The After advice will be executed.

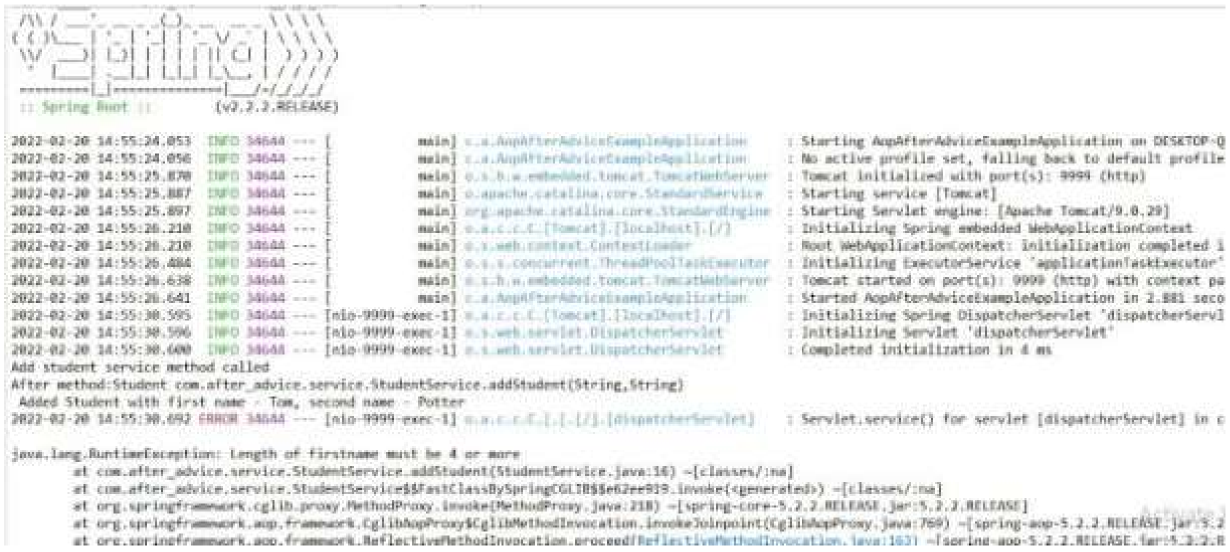


localhost:9999/add?firstName=Tom&secondName=Potter

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sun Feb 20 14:53:55 IST 2022
There was an unexpected error (type=Internal Server Error, status=500).
No message available



```
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.2.RELEASE)

2022-02-20 14:55:24.053  INFO 34644 --- [           main] c.a.AopAfterAdviceExampleApplication     : Starting AopAfterAdviceExampleApplication on DESKTOP-Q
2022-02-20 14:55:24.056  INFO 34644 --- [           main] c.a.AopAfterAdviceExampleApplication     : No active profile set, falling back to default profile
2022-02-20 14:55:25.870  INFO 34644 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 9999 (http)
2022-02-20 14:55:25.887  INFO 34644 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-02-20 14:55:25.897  INFO 34644 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.29]
2022-02-20 14:55:26.210  INFO 34644 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-02-20 14:55:26.210  INFO 34644 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed i
2022-02-20 14:55:26.484  INFO 34644 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2022-02-20 14:55:26.638  INFO 34644 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 9999 (http) with context pa
2022-02-20 14:55:26.641  INFO 34644 --- [           main] c.a.AopAfterAdviceExampleApplication     : Started AopAfterAdviceExampleApplication in 2.881 seco
2022-02-20 14:55:30.595  INFO 34644 --- [nio-9999-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServl
2022-02-20 14:55:30.596  INFO 34644 --- [nio-9999-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2022-02-20 14:55:30.600  INFO 34644 --- [nio-9999-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 4 ms
Add student service method called
After method:Student com.after_advice.service.StudentService.addStudent(String,String)
 Added Student with first name - Tom, second name - Potter
2022-02-20 14:55:30.692 ERROR 34644 --- [nio-9999-exec-1] o.a.c.c.C.[.[.[/].[dispatcherServlet]    : Servlet.service() for servlet [dispatcherServlet] in c

java.lang.RuntimeException: Length of firstname must be 4 or more
	at com.after_advice.service.StudentService.addStudent(StudentService.java:16) ~[classes/:na]
	at com.after_advice.service.StudentService$$FastClassBySpringCGLIB$$e62ee939.invoke(<generated>) ~[classes/:na]
	at org.springframework.cglib.proxy.MethodProxy.invoke(MethodProxy.java:218) ~[spring-core-5.2.2.RELEASE.jar:5.2.2.RELEASE]
	at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invokeJoinpoint(CglibAopProxy.java:769) ~[spring-aop-5.2.2.RELEASE.jar:5.2
	at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:163) ~[spring-aop-5.2.2.RELEASE.jar:5.2.2.R
```

As seen in the output, the advice method is called after the service method is executed successfully or any exception is raised.

Comment More info

Campus Training Program

GeeksforGeeks
Sanchhaya Education Private Limited

Corporate & Communications Address:
A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305)

Registered Address:
K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305

GET IT ON Google Play
Download on the App Store

Advertise with us

Company
About Us
Legal
Privacy Policy

Explore
POTD
Job-A-Thon
Connect