HowToDoInJava

Spring AOP Tutorial

Last Updated: December 26, 2020

In this **Spring AOP** tutorial, learn what is aspect-oriented programming with example. Also learn what is advice, join-point, and point-cut expressions and how to use them in Spring application with examples.

1. What is Spring AOP?

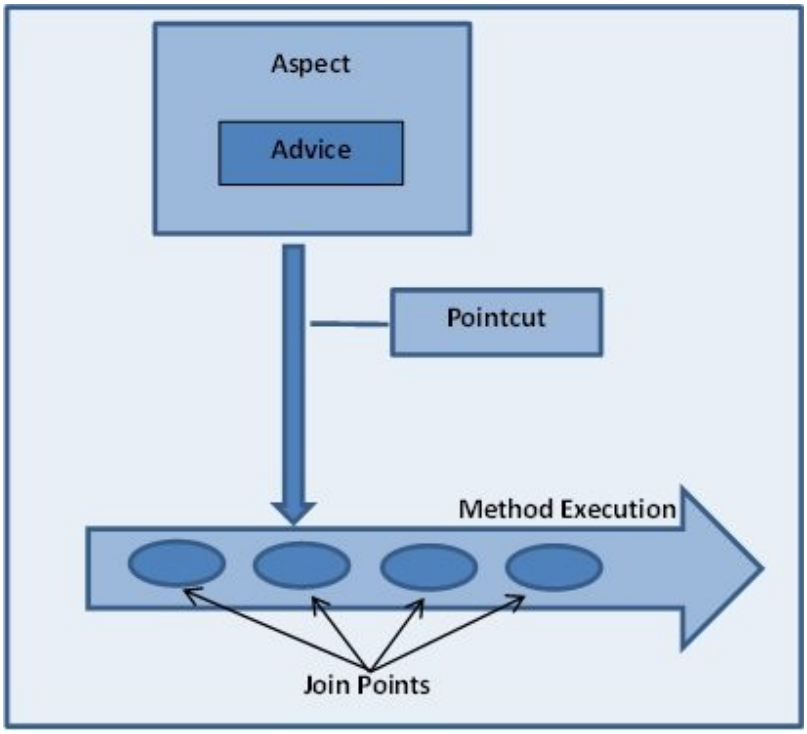
Spring AOP enables Aspect-Oriented Programming in spring applications. In AOP, aspects enable the modularization of concerns such as transaction management, logging or security that cut across multiple types and objects (often termed **crosscutting concerns**).

AOP provides the way to dynamically add the cross-cutting concern before, after or around the actual logic using simple pluggable configurations. It makes easy to maintain code in the present and future as well. You can add/remove concerns without recompiling complete sourcecode simply by changing configuration files (if you are applying aspects suing XML configuration).

2. What is advice, joinpoint or pointcut?

1. An important term in AOP is **advice**. It is the action taken by an **aspect** at a particular join-point.
2. **Joinpoint** is a point of execution of the program, such as the execution of a method or the handling of an exception. In Spring AOP, a joinpoint always represents a method execution.
3. **Pointcut** is a predicate or expression that matches join points.
4. **Advice** is associated with a pointcut expression and runs at any join point matched by the pointcut.

5. Spring uses the AspectJ pointcut expression language by default.



Spring AOP

3. Types of AOP Advices

There are five types of advice in spring AOP.

1. **Before advice**: Advice that executes before a join point, but which does not have the ability to prevent execution flow proceeding to the join point (unless it throws an exception).
2. **After returning advice**: Advice to be executed after a join point completes normally: for example, if a method returns without throwing an exception.
3. **After throwing advice**: Advice to be executed if a method exits by throwing an exception.
4. **After advice**: Advice to be executed regardless of the means by which a join point exits (normal or exceptional return).
5. **Around advice:** Advice that surrounds a join point such as a method invocation. This is the most powerful kind of advice. Around advice can perform custom behavior before and after the method invocation. It is also responsible for

choosing whether to proceed to the join point or to shortcut the advised method execution by returning its own return value or throwing an exception.

4. Spring AOP Example

4.1. Maven Dependencies

Before writing any code, you will need to import *Spring AOP dependencies* into your project.

pom.xml

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.2.7.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context-support</artifactId>
    <version>5.2.7.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-aop</artifactId>
    <version>5.2.7.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjrt</artifactId>
    <version>1.9.5</version>
</dependency>
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjweaver</artifactId>
    <version>1.9.5</version>
</dependency>
```

Enable AOP configuration in Spring applications.

AopConfig.java

```
@Configuration
@EnableAspectJAutoProxy
public class AopConfig {
```

```
}
```

4.2. Aspect and pointcut expression

Write aspect class annotated with `@Aspect` annotation and write point-cut expressions to match joint-point methods.

```
EmployeeCRUDAspect.java

@Aspect
public class EmployeeCRUDAspect {

    @Before("execution(* EmployeeManager.getEmployeeById(..))")     //point-cut e
    public void logBeforeV1(JoinPoint joinPoint)
    {
        System.out.println("EmployeeCRUDAspect.logBeforeV1() : " + joinPoint.getS
    }
}
```

4.3. Methods (joint points)

Write methods on which you want to execute advices and those match with point-cut expressions.

```
EmployeeManager.java

@Component
public class EmployeeManager
{
    public EmployeeDTO getEmployeeById(Integer employeeId) {
        System.out.println("Method getEmployeeById() called");
        return new EmployeeDTO();
    }
}
```

In above example, `logBeforeV1()` will be executed **before** `getEmployeeById()` method because it matches the join-point expression.

4.4. Run the application

Run the application and watch the console.

```
TestAOP.java

public class TestAOP
{
    @SuppressWarnings("resource")
    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext
                ("com/howtodoinjava/demo/aop/applicationContext.xml");

        EmployeeManager manager = context.getBean(EmployeeManager.class);

        manager.getEmployeeById(1);
    }
}
```

Program output:

```
Console

EmployeeCRUDAspect.logBeforeV1() : getEmployeeById
Method getEmployeeById() called
```

Spring aop tutorial for beginners with example.

5. Spring AOP XML Configuration Examples

1. Spring AOP AspectJ XML Configuration Example
   Learn to configure AOP aspects using XML configuration.

2. Spring AOP Before Advice Example
   Learn to configure aop before advice aspect using `<aop:before/>` configuration.

3. Spring AOP After Returning Advice Example
   Learn to configure aop after returning advice aspect using `<aop:after-returning/>` configuration.

4. Spring AOP After Throwing Advice Example
   Learn to configure aop after throwing advice aspect using `<aop:after-throwing/>` configuration.

5. Spring AOP After Advice Example
   Learn to configure aop after advice aspect using `<aop:after/>` configuration.

6. Spring AOP Around Advice Example
   Learn to configure aop around advice aspect using `<aop:around/>` configuration.

6. Spring AOP AspectJ Annotations Examples

1. Spring AOP AspectJ Annotation Config Example
   Learn to configure AOP aspects using aspectj annotations configuration.

2. Spring AOP AspectJ @Before Example
   Learn to configure aop before advice aspect using `@Before` annotation.

3. Spring AOP AspectJ @After Example
   Learn to configure aop after advice aspect using `@After` annotation.

4. Spring AOP AspectJ @Around Example
   Learn to configure aop around advice aspect using `@Around` annotation.

5. Spring AOP AspectJ @AfterReturning Example
   Learn to configure aop after returning advice aspect using `@AfterReturning` annotation.

6. Spring AOP AspectJ @AfterThrowing Example
   Learn to configure aop after throwing advice aspect using `@AfterThrowing` annotation.

7. More Spring AOP Tutorial

1. Spring AOP Aspects Ordering
   Learn to order the aspect execution in case of multiple aspects which need to be executed in certain order.

2. Spring AOP AspectJ Pointcut Expressions With Examples
   Learn to write pointcut expressions to match a variety of join points.

8. Interview Questions

Top Spring AOP Interview Questions with Answers

Some most asked spring AOP interview questions in java interviews.

9. Spring AOP Resource(s):

Spring AOP Doc
AspectJ

Happy Learning !!

11 thoughts on “Spring AOP Tutorial”

Rustam Shafigullin

March 24, 2020 at 10:25 pm

and one more issue with this example, you should add configuration class in the project:

```
@Configuration
@EnableAspectJAutoProxy
public class Config {

}
```

Rustam Shafigullin

March 24, 2020 at 10:23 pm

I have lost whole day to make this code work. Please fix this code with

@Aspect
@Component
public class EmployeeCRUDAspect {

instead of

@Aspect
public class EmployeeCRUDAspect {

Lokesh Gupta

March 24, 2020 at 10:56 pm

Why use `@Component` annotation? `@Aspect` is automatically detected by Spring. [Ref]

Rustam Shafigullin

March 25, 2020 at 5:28 pm

Maybe, because I used annotation configuration instead of XML configuration

Nadezhda

February 21, 2019 at 9:21 pm

Hello,

thanks for the tutorials! Is corresponding source code available for download?

Hiranand

January 19, 2019 at 3:29 pm

change the spelling of “Using” of (if you are applying aspects suing XML configuration).

Paramesh

May 10, 2018 at 10:16 am

Good tutorial for beginners

Titan

February 2, 2017 at 1:33 pm

Spring AOP AspectJ @Around Example linked to wrong page

Lokesh Gupta

February 2, 2017 at 3:05 pm

> Corrected. Thanks for pointing out. Much appreciated !!

Shivam

January 5, 2017 at 3:10 pm

> Could you please correct spelling in sub-headings under : Spring AOP XML Configuration Examples. These are written as “Sprign” instead of “Spring”.

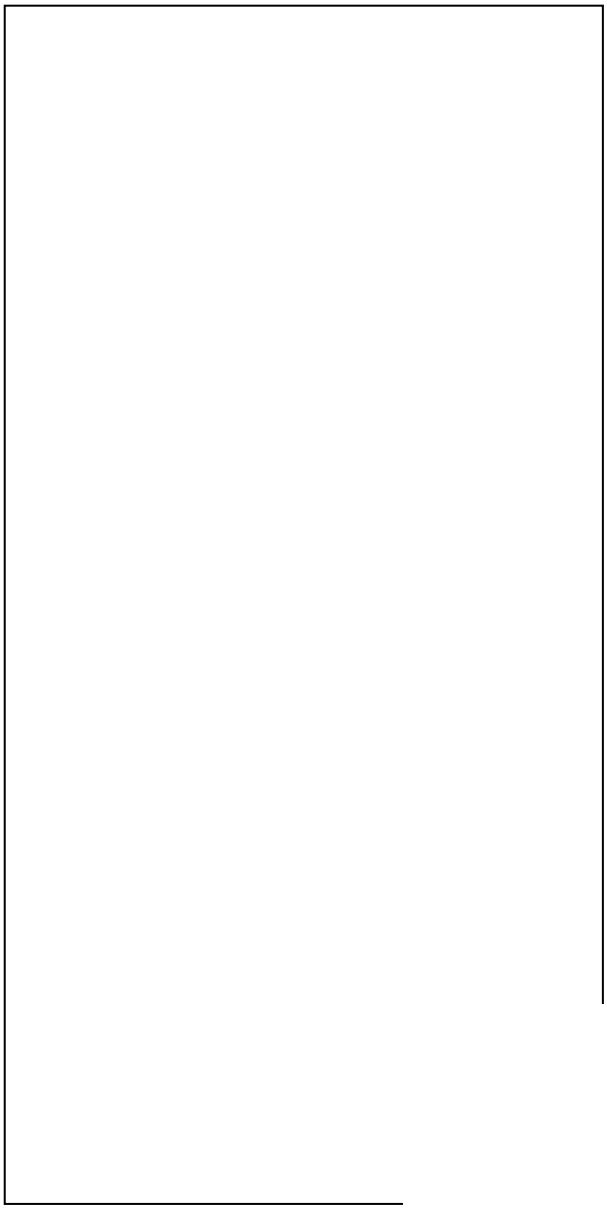
Lokesh Gupta

January 6, 2017 at 4:29 pm

> Shivan, thanks for pointing out. Much appreciated. I have corrected it now.

Comments are closed.

Search …



HowToDoInJava

A blog about Java and related technologies, the best practices, algorithms, and interview questions.

Meta Links

- About Me
- Contact Us
- Privacy policy
- Advertise
- Guest Posts

Blogs

REST API Tutorial