AOP(Aspect Oriented Programming)

AOP is a programming paradigm whose main aim is to increase modularity by allowing the separation of cross-cutting concerns. AOP does this by adding additional behaviour to the existing code without modifying the code itself. It provides functionalities to add new code and behaviour separately.

The AOP provides us the flexibility to define all the common concerns in one class and we can define where all these concerns are used and how they are used in a class without modifying it is existing code and applying the new features. This class with all the cross-cutting concerns is modularized into a specific class call Aspect.

The aspect class provides us the flexibility to:

1. Define each concern in a single class instead of spreading the concerns all over the codebase.
2. The business layer only contains the primary logic, all the secondary logic and concerns are placed in a single class, aspect.

There are some key terminologies in Aspect-Oriented Programming that are shown in the tabular manner below as follows:

Terminologies	Action Performed
Aspect	It is a module that provides cross-cutting concerns by encapsulating the advice and pointcuts. An application can have any number

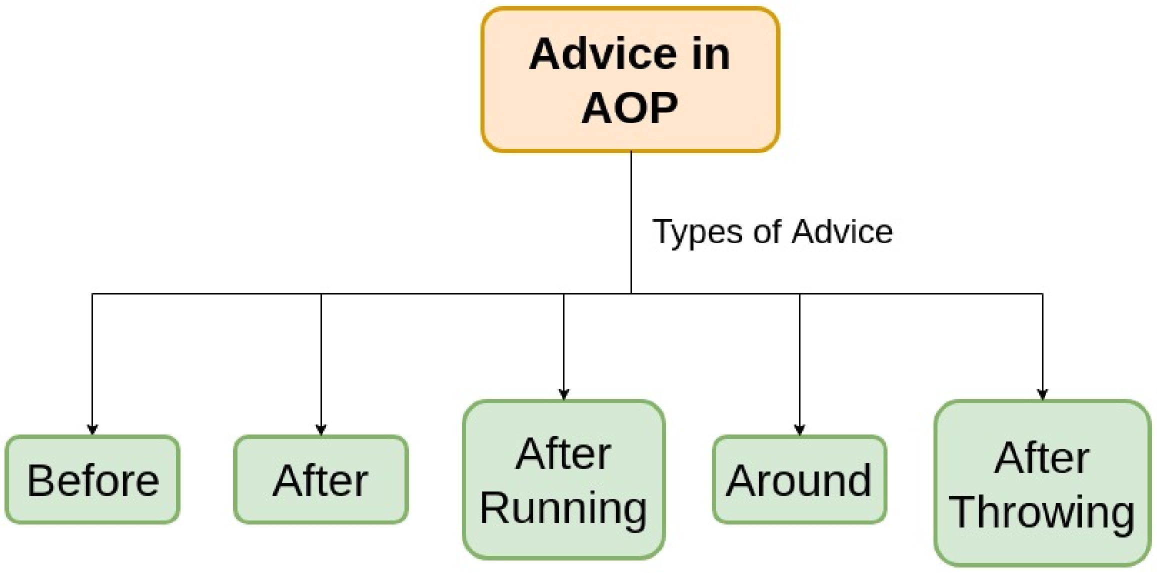
	of aspects in it. In order to use it, we need to annotate a class with @Aspect annotation.
Advice	It is an action that is taken before or after method execution. An action is a block of code that gets invoked during the execution of a program. The Spring AOP framework support five type of advice before, after, after-returning, after-throwing, and around advice. The advice is taken for join points. Advice is applied over the target object.
Pointcuts	Pointcuts are the set of join points where the advice is executed. These pointcuts are defined using the expression or pattern.
Join Points	It is a place in an application where the AOP Aspect is applied. A join point can be a method execution, exception handling, etc.
Target Object	It is an object where the advice is applied. These target objects are proxied
Proxied	Target objects are proxied which means during the runtime the target methods are overridden

	and depending on method configuration, the advice is included in the target object.
Weaving	The process of linking the application with the aspect is called weaving. It can be done at load time, compile time and run time.

Why do we need AOP?

In web applications, each of the layers(web, business, and data layer) are responsible for different tasks and they perform these tasks individually. But there are a few common aspects that are applied to each layer such as security, caching, validation, etc. These common aspects are known as Cross-Cutting Concerns.

The implementation of these cross-cutting concerns in each module separately makes the code lengthy also, makes it difficult to manage. In order to overcome this problem, Aspect-Oriented Programming is introduced. In which, we implement these cross-cutting concerns as an aspect and define pointcuts(set of join points) where these aspects should be applied.


Advice in AOP
Types of Advice
Before
After
After Running
Around
After Throwing