

**1.**     **Overview**

In this article we will learn about the *Aspect Oriented Programming (AOP)* and various terminologies associated with it. Then, we will look how the Spring framework provides the capability to implement various *cross cutting concerns* of the application through AOP. Lastly, we will then write a sample application with focus on Spring AOP.

**2.**   **Cross cutting concerns**

In any enterprise application, there are a number of concerns which need to be taken care of in addition to the main business logic. These concerns are spread across the application and into multiple application layers. Such concerns are logging, transaction handling, performance monitoring, security etc. These concerns are known as *cross cutting concerns* of the application.

Cross Cutting Concerns

AOP help is implementing the cross cutting concerns of the application keeping them separate from the main business logic and thus resulting in loosely coupled applications.

## 3. Need of AOP

Before going in to details of AOP we must understand the need of AOP.

Drawbacks of the conventional approach of using logging / System out println statements throughout app for logging for debugging purpose:

1. Similar logic of logging the request parameters and the response value is spread across multiple methods and thus creating redundant code. This makes maintenance very difficult.
2. Any change in the requirement of application logging will result in changing the code of multiple methods of LibraryService .
3. Adding any new method in the LibraryService will result in rewriting the logging code again in the newly added method. Thus we are not able to reuse the existing logging logic.
4. The main responsibility of LibraryService is to provide various operations of Library rather than logging. Keeping the code of logging in LibraryService is not a good idea.

AOP helps to implement the logging concern (and all other cross cutting concerns) of the application overcoming all of the above drawbacks. AOP keeps the cross cutting concerns separate from the main business logic of the application and weaves them appropriately in the various application object.

Spring AOP is used extensively in Spring’s Transaction Management where the transaction handlers are injected or weaved around the method execution join points. AOP is also used in Spring’s Security module to secure the method call for authenticated and authorised users only. Note that AOP is hidden behind the Spring’s Security namespaces so that the users do not worry about weaving security handlers in application objects - all is done through namespaces.

## 4.     AOP terminologies

Let us now discuss the various terminologies used in an aspect oriented programming. Note that these terminologies not only are specific to Spring AOP but also are used in general for any AOP framework.

### Aspect

An Aspect is the concern (cross cutting concern) which you want to implement in the application such as logging, performance monitoring, transactional handing etc.

### Advice

An Advice is the actual implementation of the aspect. Aspect is a concept and Advice is the concrete implementation of the concept.

### Join Point

A JoinPoint is a point in the execution of the program where an aspect can be applied. It could be before/after executing the method, before throwing an exception, before/after modifying an instance variable etc. Keep in mind that it is not necessary and also not required to apply an aspect at all the available join points. Spring AOP only supports method execution join points.

### Point cut

PointCuts tell on which join points the aspect will be applied. An advice is associated with a point cut expression and is applied to a join point which matches the point cut expression.

### Target

Target is the application object on which the advice will be applied.

### Proxy

Proxy is the object which is created by the framework after applying the advice on the target object.

### Weaving

Weaving is the process of applying the aspect on the target object to product the proxy object. Weaving can be done at compile time, class loading time or runtime. Spring AOP supports weaving at runtime.

**5.**     **Advices**

As discussed earlier, Spring only supports method execution join points. The various method execution join points can be:

1. Before a method execution starts
2. After the method execution completes normally
3. After the method throws an exception
4. Around the method execution

<https://www.codejava.net/frameworks/spring/understanding-spring-aop>