Definition :

Spring AOP is a powerful framework that integrates AOP concepts into Spring applications.

It provides declarative way to apply aspects to Spring beans methods using annotation or xml configuration.

Spring AOP uses dynamic proxies or byte-code instrumentation to intercept method invocations and apply aspects .

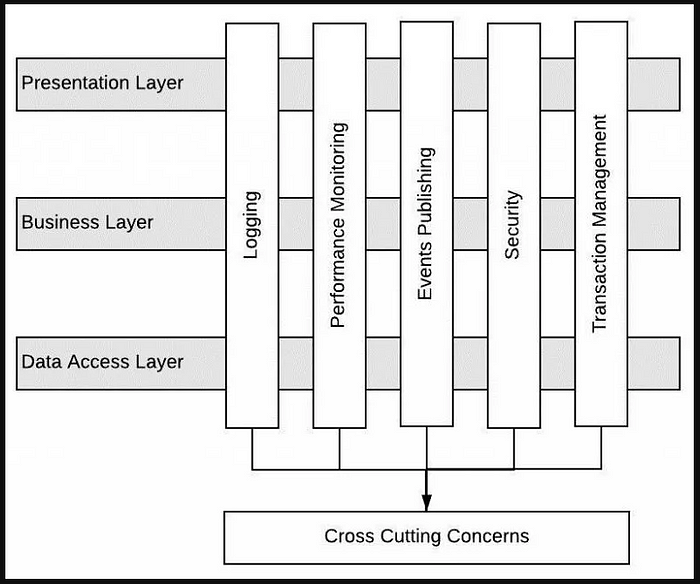
1. Advice
2. Pointcuts
3. Weaving
4. Joint Points
5. Cross cutting concerns

Why AOP ?

Spring AOP provides elegant solution for logging system for the enterprise application. Example ecommerce application – product searches, cart updates, order placements.

What is AOP?

* AOP used to modularise cross cutting concerns (functionalities that cut across different parts of application)
* Concerns include logging, transaction management, security and error handling, declarative transactions, security, caching and more.
* AOP helps in separating the concerns from the core business logic, resulting in cleaner and more maintainable code.



Benefits of AOP?

* Modularity: AOP enables the separation of cross cutting concerns from the core business logic, making easier to maintain code bases.
* Reusability: Encapsulate cross-cutting concerns in reusable modules called aspects and apply them to multiple parts of your application.
* Maintainability: BY isolating cross-cutting concerns, AOP reduces code duplication and improve the overall maintainability of codebase.
* Flexibility: Flexible way to add or remove functionality without modifying core business logic, improving adaptability and extensibility.

*@Aspect – They contain advice, which represents the actions to be taken at specific join points in the application.*

*Aspect is a module the encapsulates and address that concerns using advice .*

Aspects and annotations in AOP

1. Aspect: Module to encapsulate cross-cutting concerns .It is defines using @Aspect annotation.
2. Join Point: Spring AOP provides a set of supported join points , including method execution, exception handling and field access .
3. Pointcut: Spring AOP uses pointcut expressions, which are defines using the @pointcut annotation.
4. Advice: There are several types of advice such as
   1. @before
   2. @after
   3. @after returning
   4. @after throwing.
5. Target Object : Object on which advices are applied.
6. Proxy: Object that is created after applying advice to target object.
7. Weaving: Process of linking aspects with other application types or objects to create an advised object.
8. Introduction: It allows adding new methods or fields to existing classes.It helps achieve cross cutting concerns by exztending the functionality of target classes without modifyinh thiers source code .