AOP (Aspect oriented programming)

Everything other than business is known as cross cutting concern or aspect.

E.g of aspects

* Exceptions
* Security
* Database connection
* Frontend
* Testing

Different aspects have different locations

Advantages:

* parallel development
* loose coupling
* Reuse

Where use AOP?

AOP is mostly used in following cases:

* to provide declarative enterprise services such as declarative transaction management.
* It allows users to implement custom aspects.

---------------------------------------------------------------------

AOP is consider as dynamic(runtime ) design pattern.

It is used to increase modularity by **cross-cutting concerns**.

Spring supports runtime weaving

## AOP Concepts and Terminology

AOP concepts and terminologies are as follows:

* Join point
* Advice
* Pointcut
* Introduction
* Target Object
* Aspect
* Interceptor
* AOP Proxy
* Weaving
* Aspect: It is class in which we write pre-processing code.

e.g Transaction management

It can be normal class configured through XML or @Aspect annotation.

It is a class where we write cross cutting concern.

It is a class that contains advices, joinpoints etc.

* Join Point:

Join point is any point in your program such as method execution, exception handling, field access etc. Spring supports only method execution join point.

* Advice

Advice represents an action taken by an aspect at a particular join point. There are different types of advices:

* **Before Advice**: it executes before a join point.
* **After Returning Advice**: it executes after a joint point completes normally.
* **After Throwing Advice**: it executes if method exits by throwing an exception.
* **After (finally) Advice**: it executes after a join point regardless of join point exit whether normally or exceptional return.
* **Around Advice**: It executes before and after a join point.

#### Pointcut

It is an expression language of AOP that matches join points.

#### Introduction

It means introduction of additional method and fields for a type. It allows you to introduce new interface to any advised object.

#### Target Object

It is the object i.e. being advised by one or more aspects. It is also known as proxied object in spring because Spring AOP is implemented using runtime proxies.

#### Interceptor

It is an aspect that contains only one advice.

#### AOP Proxy

It is used to implement aspect contracts, created by AOP framework. It will be a JDK dynamic proxy or CGLIB proxy in spring framework.

A proxy is an object which is created after applying advice to target object.

In terms of client, proxy object and target object both are same.

#### Weaving

It is the process of linking aspect with other application types or objects to create an advised object. Weaving can be done at compile time, load time or runtime. Spring AOP performs weaving at runtime.

### AOP Implementations

AOP implementations are provided by:

1. AspectJ
2. Spring AOP
3. JBoss AOP