```
SpringBoot-AOP
=============
To write programs we have 3 styles of coding
a. POP  => Procedure Oriented Programming[C]
b. OOPS => Object Oriented Programming[C++,Java,....]
c. AOP  => Aspect Oriented Programming[Java,Python,Ruby,....]

Aop =>Cross Cutting Concern
         Seperating the Buisness logic from the Service Logic is called "Cross-
Cutting Concern".

CrossCuttingConcern: Move Additional classes of Project into another
classes[Serviceclass/Aspectclass] and bind
                               when and where they are required.

1. Aspect => It is a class that provides additional services to project.
                        TransactionManagement,Security,Encode/Decode,Logging,...

2. Advice => It is a method inside Aspect class/It is acutal implementation of
Aspect.
                Every Advice must be connected to Type.
Types of Advices
      a. Before Advice => Executing Advice before calling methods
                        execution order :: adviceMethod ----- 1st
                                              b.method     ----- 2nd


      b. After Advice  => Executing Advice after b.methods finished
                        execution order ::  b.method     ----- 1st
                                        adviceMethod  ----- 2nd


      c. Around Advice => Advice code divided into 2 methods
                            First part executed before advice and then
b.method, later 2nd part of Advice.
                        execution order :: adviceMethod ---- 1st part
                                         b.method
                                        advicemethod ----- 2nd part


      d. After Returning Advice => After executing b.method only on success execute
Advice.
                  execution order :: b.method ----- 1st
                              (Is b.method executed succesfully,No Exception then
call Advice)

      e. After Throwing Advice  => After executing b.method if b.method is throwing
any exception then execute Advice.
                  execution order :: b.method ----- 1st
                              (Is b.method not executed succesfully,Exception
occured) then
                                          call adviceMethod() --- 2nd

3. PointCut => It is a expression that will select buisness methods which needs
Advices,It can never specify what advices.
                  expression :: Access_Specifier ReturnType
packagename.classname.methodname (paramname)

4. JoinPoint => It is a combination of Advice + PointCut. In simple words
```

Connecting Buisness Methods with the required Advices.

5. Target => Pure Buisness class Object.
6. Weaving => It is a process of mixing buisness class methods and there connected advices.
7. Proxy => Final output(class/object) is called as "Proxy" that contains both logics connected.

What is the difference b/w @After, @AfterReturning and @AfterThrowing Advices?
  After Advices is executed next to buisness method either success or failure.
  AfterReturning Advices is executed next to buisness method only if there is not exception.
  AfterThrowing Advices is executed next to buisness method only if there is a exception.

Implementation of the above can be done in 2 ways
==
1. Spring AOP using XML Based Configuration[Legacy Style]
2. Spring AOP using AspectJ(Pure Annotations)

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-aop</artifactId>
</dependency>
```

3. Annotations used are
   a. @Aspect
   b. @Before
   c. @After
   d. @Around
   e. @AfterReturning
   f. @AfterThrowing
   g. @PointCut

refer:: Spring-AOP-AspectJApp