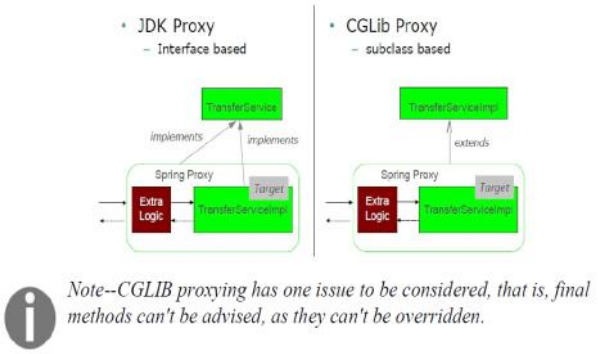
AOP

24 Temmuz 2018 Salı 14:51

There is some generic function across the application which is not related to business logic. You cannot encapsulte it in a class. These are known as crossing cutting concerns; logging, security, transaction, performance monitoring..etc

Separating business logic from cross cuttings concerns.
For example: In banking app transfering money from one account to another is business logic, but logging and security for this process is cross cutting concern.

Please note that Spring AOP can only apply Spring Beans.



Note--CGLIB proxying has one issue to be considered, that is, final methods can't be advised, as they can't be overridden.

Real use case of AOP
- @Transactional annotation which is used AOP implicitly
- Something should be logged for example banking transaction

Terms for AOP
Aspect: this is a class for cross cutting concern.
Advice: this is a main job of aspect. Before, After, AfterReturning, AfterThrowing, Around are types of advice. All Aspect methods are known as Advice.
JoinPoint: Applying points in a business object class.
PointCut: Regular expression for where to apply.
Weaving: Spring AOP uses dynanimc proxy which works at runtime. It creates simply proxy object at runtime as Bean in IOC container and get Proxy object eveytime Bean instance requested.

Annotation	Advice
`@Before`	It is used for before advice. `advice`'s method executes before the advised method is invoked.
`@After`	It is used for after advice, advice's method execute after the advised method executes normally or abnormally doesn't matter.
`@AfterReturning`	It used for after returning advice, advice's method execute after the advised method complete successfully.
`@AfterThrowing`	It used for after throwing advice, advice's method execute after the method terminate abnormally by throwing an exception.
`@Around`	It is used for around advice, advice's method executes before and after the advised method invoked.