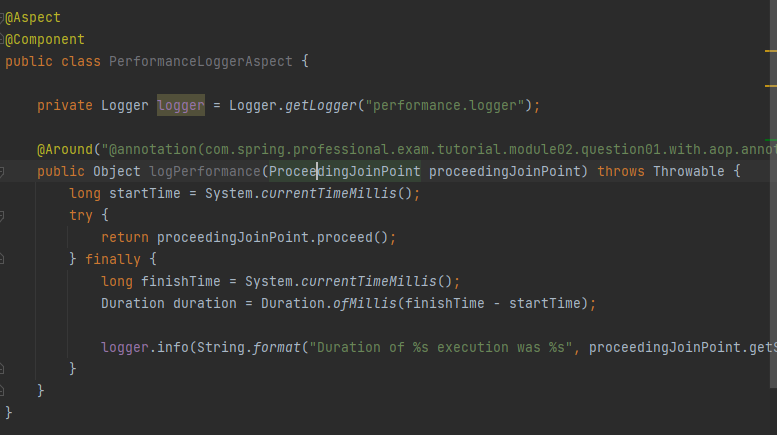
1. **What is the concept of AOP?**

Text

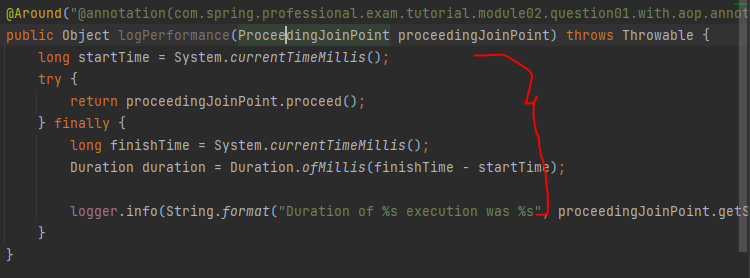
Description automatically generated

AOP consist of the following concepts:

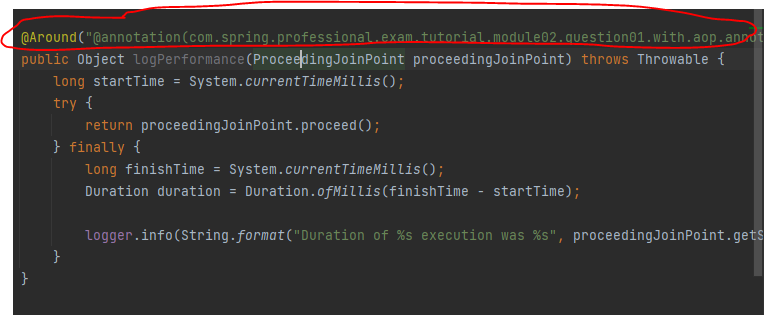
1. Aspect class:



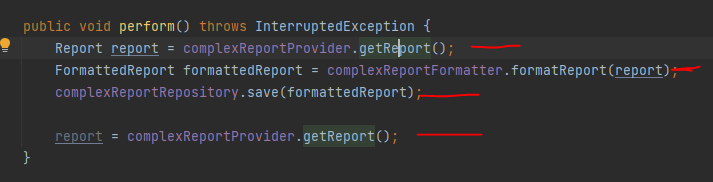
1. Advice: where the code been injected :



1. Pointcut expression which match the jointpoint:



1. Jointpoint is the method execution:



1. The advice code will be injected whenever pointcut expression matched , example for the following code:

Text

Description automatically generated

We have pointcut ‘@annotation(com.spring.professional.exam.tutorial.module02.question01.with.aop.annotations.PerformanceLogger’ and this PerformaceLogger is a custom annotation class:

Graphical user interface, text, application

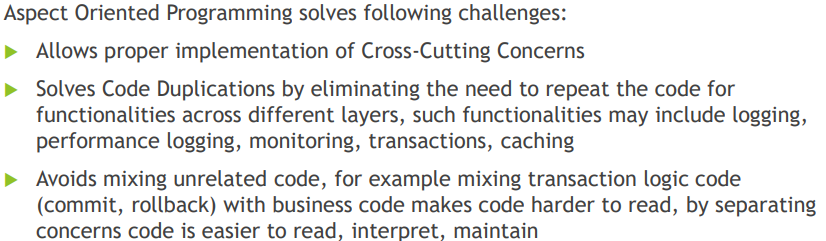
Description automatically generated

1. For the jointpoint we should annotate it with @PerformanceLogger to trigger the AOP:

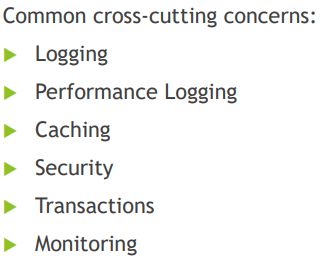
Text

Description automatically generated

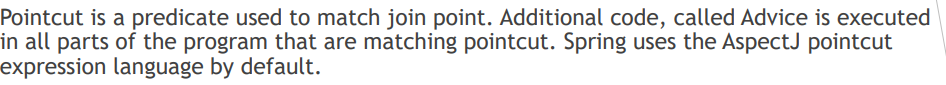
1. **What is the problem that AOP solve?**



1. **Name three typical cross-cutting concerns?**



1. **What is the pointcut , joint-point and advice , an aspect and weaving ?**



1. **Point-cut expression examples?**

Text

Description automatically generated

Text

Description automatically generated

1. **What is the advice?**



Types of point-cut:

1. Inline point-cut (point-cut and advice in the same method):

Text

Description automatically generated with medium confidence

1. Outline point-cut (point-cut and advice in two different methods):

Graphical user interface, text

Description automatically generated

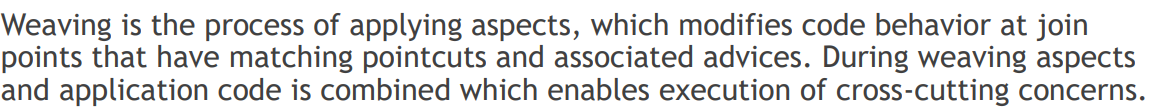
1. **What is Aspect?**

Text

Description automatically generated

Aspect **must** be annotated with @componant

1. **What is the weaving concept?**



1. **Types of weaving?**

Text

Description automatically generated

Note: that spring use runtime weaving,

Spring will use CGLIB proxy or jdk proxy to implement the Aspect:

See the following example , currencyService is class (not interface) so spring will use cglib proxy to create a reference of this class by **extending** from the class.

For currenciesRepository spring will use jdk proxy bcoz its interface so it will create a reference of the object itself.

Text

Description automatically generated

In the debug mode we can see:

Text, website

Description automatically generated

For jdk it container the reference of the original class:

Text

Description automatically generated

For cglib it will just extend the original class:

Text

Description automatically generated

Note: its good to take a look at module02-q02

1. **How does spring solve/implement the cross-cutting concern?**

Text

Description automatically generated

1. Graphical user interface, text, chat or text message

   Description automatically generated**How jdk proxy works?**

Spring will implemet the interface and it will has a reference of the target class (in the example the target class is EmployeeRepositoryImp) and in the new class it will pass the target in the constructor,

So in this example the new class is EmployeeRepositoryJdkProxy that has a reference to the target class which is EmployeeRepositiryImp .

In the Client Side it will call the EmployeeRepositoryJdkProxy instead of the original class.

So lets say when the client call findEmployeeById in EmployeeRepositoryJdkProxy then this will trigger the code that should be called before/after the findEmployeeById at the target .

EmployeeRepositoryJdkProxy has a method ->

findEmployeeById()

call before -🡪 do something

call original method from the target class which is

targetClass.findEmployeeById()

call after 🡪 do something

so under the hood spring will create something like this:

Text

Description automatically generated

It will implement the interface and it will has the reference from the target class.

And it will invoke the code from the target.

Text

Description automatically generated

1. **How cglib works?**

In cglib it will extend the class and it will override the methods:

Graphical user interface, text, application

Description automatically generated

And the client will call EmployeeRepositoryCGLIBProxy instead of the original class.

So this is what spring do under the hood for the cglib:

A screen shot of a computer

Description automatically generated with medium confidence

It will extend from the original class and will override the methods

And it will invoke the method from the original class by calling super.<method name>

To force spring to use cglib:

Text

Description automatically generated

1. **What are the limitation of using jdk and cglib proxy?**

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Even tho if we try to invoke the private method it will not be proxied :

Text

Description automatically generatedText

Description automatically generated

1. **What is the types of advices?**

Text

Description automatically generated

Please look at module02-q05 its important.

1. **What should you use to catch exceptions?**

Text

Description automatically generated

1. **How to enable the detection of @Aspect annotation?**

A picture containing text, person, screenshot, document

Description automatically generated

1. **What is the point-cut expression types?**

Text

Description automatically generated with medium confidence

1. Text

   Description automatically generatedExecution expression type:

Text

Description automatically generated

The execution for the first method @Before will match all public modifier access that return NOT int and ALL sub class (..) for HelloBean that starts with say\* and accept String as first param and anything for next param, as we saw here we didn’t specify the throws coz its optional.

For the @After execution we match all access modifier (coz we didn’t mention the access modifier so to will match all by default) and the method that will return void and all class and subclass under HelloChildBean and the name of the method validateName and accept any number of param and throws exception io.IOException.

A picture containing text

Description automatically generated

1. Within expression:

Text, letter

Description automatically generated

Text

Description automatically generated

For the first method it will match all packages and sub-packages and the class name is HelloChildBean.

For the next method it will match all sub-packages with any class name within com package.

For the third class it will match all sub-packages that start with Hello and end with Bean

1. Arg expression type:

Graphical user interface, text, application, email

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

1. Bean expression type:

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

This will match **all the methods** in the bean.

Lets say HelloCildBean has 4 method then the aspect it will be called 8 times for before and after.

1. This & target expression types (important):

Before we explain this lest take a look how cglib and jdk work:

For jdk:

Graphical user interface, text, application

Description automatically generated

For EmployeeRepositoryIml it’s the ‘target’,

And EpmolyeeRepositoryJdkProxy is the ‘this’

Lets take a look on this example:

Text

Description automatically generated with medium confidence

ITaxBean is the interface

TaxBean1 is implementing ITaxBean (which mean it will use jdk).

TaxBean2 not implementing any interface (which will use cglib).

Lets take a look on the aspect class:

Text

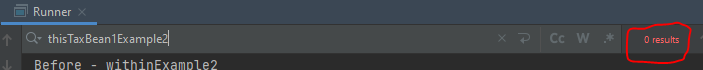
Description automatically generated

For thisTaxBean1Example1() it will be called bcoz ‘this’ is actually implementing ITaxBean interface.

Graphical user interface, text, application

Description automatically generated

For thisTaxBeanExample2() it will not called bcoz this is NOT type of TaxBean1 , its jdk that use the TaxBean as **delegation only**, that’s why for thisTaxBeanExample2() its not been executed.



For targetTaxBeanExample1() it will be called bcoz the target in jdk is type of ITaxBean coz its implementing it already.

Graphical user interface, text, application

Description automatically generated

For targetTaxBean1Example2() it will be called bcoz the target is match TaxBean1 coz its jdk.

Graphical user interface, text, application

Description automatically generated

For Cglib:

Graphical user interface, text

Description automatically generated

It will extend from the target that’s why both this & target will be matched for cglib.

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

A screenshot of a computer

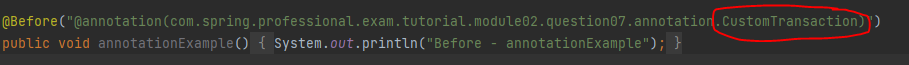
Description automatically generated with low confidence

1. @annotaion expression type:

Graphical user interface, text, application, email

Description automatically generated

For @annotation will works on top of the method only , not class.



Graphical user interface, text, application

Description automatically generated

Diagram

Description automatically generated

And the result will be:

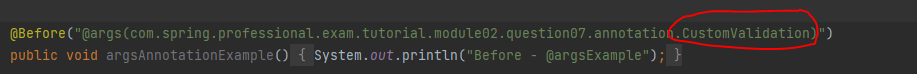
Graphical user interface, text, application, chat or text message

Description automatically generated

1. @args expression type:

Graphical user interface, text, application, email

Description automatically generated



Graphical user interface, text, application

Description automatically generated

And this interface used on top of the class:

Graphical user interface, text, website

Description automatically generated

And this class used over here:

Graphical user interface, text

Description automatically generated

And the results will be:

Graphical user interface, text, chat or text message

Description automatically generated

1. @within expression type:

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

And the custom annotation:

Graphical user interface, website

Description automatically generated with medium confidence

And its annotated-on top of this class:

Text

Description automatically generated

And the results will be:

Graphical user interface, application

Description automatically generated

It will be 2 bcoz inside this class there are two methods

1. @target expression type:

Graphical user interface, application

Description automatically generated

It will be executed 2 times also bcoz cglib proxy and the target that has been created is also annotated with this custom annotation.

1. **What is the logical operator for pointcut?**

Text

Description automatically generated

Text

Description automatically generated

1. **What is the joint-point args?**

It used to retrieve some info about the join-point.

Text

Description automatically generated

Text

Description automatically generated

1. **What is ProceedingJoinPoint?**

Graphical user interface, text, application, email

Description automatically generated

Please take a look at module02-q09