AOP

AOP

Available as of Camel 2.0

> **@deprecated**
>
> The AOP has been marked as `@deprecated` in Camel 2.4 and will be removed in a future release. The concept of the AOP did not go in the directions we liked, so we decided it's better to remove unused/less used features.

Camel has a AOP concept in the DSL allowing you do do some custom **before** and/or **after** processing when routing an Exchange.
It replaces the `intercept` from Camel 1.x. `intercept` was more limited because it could only be fixed around a single processor.

This new AOP can span multiple processors (nodes in the route graph) and it uses endpoint URIs to allow invoking any kind of Camel component.

The AOP supports the following callbacks in which you can use 1..n free of choice:

- before
- after
- afterFinally

The difference between `after` and `afterFinally` is that `afterFinally` is invoked from a finally block so it will **always** be invoked no matter what, eg also in case of an exception occur.

Using from Java DSL

In this route we want to use the AOP to do a before and after logging.

```
from("jms:queue:inbox")
   .aop().around("log:before", "log:after")
       .to("bean:order?method=validate")
       .to("bean:order?method=handle")
   .end()
   .to("jms:queue:order");
```

Notice that we use the `end` to indicate where the AOP around should end, for instance if we do more routing afterwards such as sending to another JMS queue. In the route below the AOP is only being around the bean processing.

The aop in Java DSL have the following methods:

- around
- aroundFinally
- before
- after
- afterFinally

Using from Spring DSL

In Spring DSL its nearly the same as in the Java DSL however a bit differently

```
<route>
    <from uri="jms:queue:inbox"/>
    <aop beforeUri="log:before" afterUri="log:after">
        <to uri="bean:order?method=validate"/>
        <to uri="bean:order?method=handle"/>
    </aop>
    <to uri="jms:queue:order"/>
```

Notice we use attributes on the <aop> tag to chose with AOP concept to use.

At runtime that resolves into this messages flow:

```
inbox -> log:before -> bean.validate -> bean.handle -> log:after -> order
```

The aop in Spring DSL have the following attributes:

- beforeUri
- afterUri
- afterFinallyUri

So in order to do an around we use both the `beforeUri` and the `afterUri`.

See Also

- Architecture
- Intercept