Formation Spring Framework

Spring AOP

AOP ?

- AOP = Aspect-Oriented Programming
- AOP est un complément de l'OOP (Object-Oriented Programming).
- L'unité de modularité en OOP c'est la classe, en AOP c'est l'aspect.
- Un aspect peut s'appliquer sur plusieurs types, sur plusieurs objets.

Concepts AOP

- **Greffon (Advice)** : un programme qui sera activé à un certain point d'exécution du système. Il y a plusieurs types de greffons (around, before, after, …).
- **Point de coupe (pointcut)** : endroit du logiciel où est inséré un greffon par le tisseur d'aspect.
- **Point de jonction ou d'exécution** : endroit spécifique du système où sera inséré un greffon.
- **Aspect** : un module définissant des greffons et leurs points d'activation.
- **Tissage** : insertion statique ou dynamique dans le système logiciel de l'appel aux greffons.

Spring AOP

- Spring AOP est uniquement basé sur Java et ne nécessite pas de processus de compilation particulier.
- Spring AOP supporte uniquement les méthodes comme point de jonctions.
  - Il n'est pas possible d'intercepter des champs par exemple.
- Spring AOP permet l'utilisation d'AspectJ.

Activer AspectJ

- Via Java Configuration

```
@Configuration
@EnableAspectJAutoProxy
public class AppConfig {

}
```

- Via XML Configuration

```
<aop:aspectj-autoproxy/>
```

Déclarer un aspect

- Via Configuration Java

```
@Aspect
public class MaClasseAspect {

}
```

- Via Configuration XML

```
<bean id="monAspect" class="fr.app.MaClasseAspect">
<!-- ... -->
</bean>
```

Configuration XML

Déclarer un aspect

- Via la balise config et aspect

```
<aop:config>
    <aop:aspect id="monAspect" ref="unBean">
    ...
    </aop:aspect>
</aop:config>

<bean id="unBean" class="...">
...
</bean>
```

Déclarer un point de coupe

- Via la balise config et pointcut

```
<aop:config>

    <aop:pointcut id="metierService"
                  expression="execution(* fr.app.service.*.*(..))"/>

</aop:config>
```

Déclarer un point de coupe (1)

- Un point de coupe attaché à un aspect

```
<aop:config>

     <aop:aspect id="monAspect" ref="unBean">

         <aop:pointcut id="metierService"
                       expression="execution(* fr.app.service.*.*(..))"/>

         ...

     </aop:aspect>

</aop:config>
```

Déclarer un greffon

- Créer un greffon (advice)

```
<aop:config>
     <aop:aspect id="monAspect" ref="unBean">
         <aop:pointcut id="metierService"
                       expression="execution(* fr.app.service.*.*(..)) &amp;&amp; this(service)"/>

         <aop:before pointcut-ref="metierService" method="monitor"/>
     </aop:aspect>
</aop:config>
```

  - Le greffon créé est de type "Before" et le code exécuté est la méthode "monitor" de l'aspect.

```
public void monitor(Object service) {
     ...
}
```

Déclarer un greffon (1)

- Il est possible d'utiliser les mots clés "and", "or" et "not" dans la définition d'un point de coupe.

```
<aop:config>
      <aop:aspect id="monAspect" ref="unBean">
          <aop:pointcut id="metierService"
                        expression="execution(* fr.app.service.*.*(..)) && execution(* fr.app.dao.*.*(..))
      "/>

          <aop:before pointcut-ref="metierService" method="monitor"/>
      </aop:aspect>
</aop:config>
```

  - Le greffon créé est de type "Before" et le code exécuté est la méthode "monitor" de l'aspect.

```
public void monitor(Object service) {
    ...
}
```

Déclarer un greffon (2)

- Définir un point de coupe depuis le greffon via l'attribut "pointcut".

```
<aop:config>
    <aop:aspect id="monAspect" ref="unBean">

        <aop:before pointcut="execution(* fr.app.service.*.*(..)) and
    this(service)" method="monitor"/>

    </aop:aspect>
</aop:config>
```

Greffon After Returning

- Greffon - After Returning

```
<aop:config>
    <aop:aspect id="monAspect" ref="unBean">

        <aop:after-returning pointcut-ref="metierService"
    method=“monitor"/>

    </aop:aspect>
</aop:config>
```

Greffon After Returning (1)

- Greffon - After Returning : récupérer la valeur retournée.

```
<aop:config>
    <aop:aspect id="monAspect" ref="unBean">


    <aop:after-returning
          pointcut-ref="getService"
          returning="retVal"
          method="logGetService"/>

    </aop:aspect>
</aop:config>
```

Greffon After Throwing

- Greffon - After Throwing

```
<aop:config>
    <aop:aspect id="monAspect" ref="unBean">

        <aop:after-throwing pointcut-ref="metierService"
    method=“monitor"/>

    </aop:aspect>
</aop:config>
```

Greffon After Throwing (1)

- Greffon - After Throwing : récupérer l'exception lancée.

```
<aop:config>
    <aop:aspect id="monAspect" ref="unBean">


      <aop:after-throwing
          pointcut-ref="getService"
          throwing="ex"
          method="logGetService"/>

    </aop:aspect>
</aop:config>
```

Greffon After (finally)

- Greffon - After

```
<aop:config>
    <aop:aspect id="monAspect" ref="unBean">

        <aop:after pointcut-ref="metierService" method=“monitor"/>

    </aop:aspect>
</aop:config>
```

Greffon Around

- Greffon - Around

```
<aop:config>
     <aop:aspect id="monAspect" ref="unBean">

          <aop:around pointcut-ref="metierService" method=“profiling"/>

     </aop:aspect>
</aop:config>
```

```
       public void profiling(  pjp) {
          // noter le temps de départ
       Object valeurRetournee = pop.proceed();
       // noter le temps de fin et calculer le temps d’exécution
     }
```

Configuration Java

Déclarer un point de coupe

- @PointCut

```
@Aspect
public class MaClasseAspect {

  @Pointcut("execution(* transfer(..))") //expression
  private void toutesLesMethodesTransfert() {

  }
}
```

Déclarer un point de coupe

- Exemples

```
@Pointcut("execution(public * *(..))")
private void anyPublicOperation() {}

@Pointcut("execution(* com.xyz.someapp.trading..*)")
private void inTrading() {}

@Pointcut("anyPublicOperation() && inTrading()")
private void tradingOperation() {}
```

Déclarer un point de coupe

- Exemples

```
execution(public * *(..)) // toutes les méthodes publiques

execution(* set*(..)) // toutes les méthodes qui commencent par set

execution(* fr.app.service.PersonneService.*(..)) // toutes les
méthodes définies dans l'interface PersonneService

execution(* fr.app.service.*.*(..)) // toutes les méthodes du package
fr.app.service

execution(* fr.app.service..*.*(..)) // toutes les méthodes du package
fr.app.service et sous package
```

Définir un greffon

- Il s'agit du code qui va s'exécuter sur un point de coupe.

- Plusieurs types de greffons :
  - Before
  - After
  - After Throwing
  - After Finally
  - Around

Définir un greffon (1)

- Définir un greffon à partir d'une classe de points de coupe

```
@Aspect
public class GreffonBefore {

    @Before("fr.app.MesPointsDeCoupe.toutesLesMethodesTransfert()")
    public void logDateDebutTransfert() {
        // ...
    }

}
```

Définir un greffon (2)

- Définir un greffon et un point de coupe au même moment

```
@Aspect
public class GreffonBefore {

    @Before("execution(* set*(..))")
    public void logDateDebutSet(JointPoint jp) {
        // ...
    }

}
```

Greffon After Returning

- Greffon After Returning

```
@Aspect
public class GreffonAfter {

    @AfterReturning("execution(* set*(..))")
    public void logDateDebutSet() {
        // ...
    }

}
```

Greffon After Returning (1)

- Greffon After Returning - Récupérer la valeur retournée

```
@Aspect
public class GreffonAfter {

    @AfterReturning(
      pointcut="execution(* get*(..))",
      returning="valeurRetour"
    )
    public void logDateFinGet(Object valeurRetour) {
      // ...
    }
}
```

Greffon After Throwing

- Greffon After Throwing

```
@Aspect
public class GreffonAfterThrowing {

    @AfterThrowing("fr.app.MesPointsDeCoupe.methodesCreate()")
    public void gererErreurCreate() {
        // ...
    }

}
```

Greffon After Throwing (1)

- Greffon After Throwing - Récupérer l'exception lancée

```
@Aspect
public class GreffonAfterThrowing {

    @AfterThrowing(
      pointcut="fr.app.MesPointsDeCoupe.methodesCreate()",
      throwing="ex"
    )
    public void gererErreurCreate(MonExceptionCreate ex) {
            // ...
    }

}
```

Greffon After (Finally)

- Greffon After s'applique quelque soit le résultat (exécution normale ou exception)

```
@Aspect
public class GreffonAfterFinally {

    @After("fr.app.MesPointsDeCoupe.methodesCreate()")
    public void gererCreate() {
        // ...
    }

}
```

Greffon Around

- Greffon Around

```
@Aspect
public class GreffonAround {

    @Around("fr.app.MesPointsDeCoupe.methodesCreate()")
    public void profilerCreate(ProceedingJoinPoint pjp) {
        // noter le temps de départ

       Object valeurRetournee = pop.proceed();

       // noter le temps de fin et calculer le temps d'exécution

     }

}
```

Annotation

- Récupération de la configuration d'une annotation

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface Auditable {
    AuditCode value();
}
```

```
@Before("com.xyz.lib.Pointcuts.anyPublicMethod() && @annotation(auditable)")
public void audit(Auditable auditable) {
    AuditCode code = auditable.value();
    // ...
}
```