M1 Informatique GL - HAI818I -

Modularité Réutilisation et Composants

TD-TP Programmation par Aspects (AOP) - Introduction à AspectJ

Ce TD a pour but d'utiliser un langage de programmation *AspectJ*, inspirateur de tous les systèmes d'annotations et des frameworks associés.
Avec Eclipse pour utiliser **AspectJ** vous devez en premier lieu installer le plugin "AJDT". C'est avec ce plugin que les exemples du cours ont été testés.
Il existe un plugin pour IntelliJ : `https://www.jetbrains.com/help/idea/aspectj.html`, mais il n'est pas opérationnel.

1. Lire le cours. Vous pouvez également lire l'article de recherche fondateur : `http://www.lirmm.fr/~dony/enseig/RC/kiczales-ECOOP1997-AOP.pdf`.
2. Pour faire du AspectJ vous allez devoir créer un projet *AspectJ*, que vous trouverez en *"File>new>Other>AspectJProject.*
3. Au sein d'un tel projet, programmez l'exemple donné en cours (`https://www.lirmm.fr/~dony/notesCours/caspects.s.pdf` en créant en premier lieu des classes normales `Figure`, `Line`, etc. Vous devez créer une classe principale qui possèdera un `main`.

   Créez ensuite des *aspects* en faisant clic droit sur le projet puis : *"new>other>aspectJ>aspect.*

   Essayez les exemples donnés en cours.

   Pour exécuter, sélectionnez la classe `main` et choisissez : *"run as AspectJ/Java-Application"*.
4. Définissez un aspect permettant de mesurer le temps d'exécution de certaines méthodes de votre choix (par exemple celles d'un certain nom, ou d'une certaine classe), et reportant ces temps dans la console ou un fichier.
5. Comprendre l'exemple "Introduction example" fourni dans le plugin eclipse.
6. Faites le lien avec le pattern *Visitor*. Considérez vos classes `ElementDeStockage` du TP correspondant et dotez les d'un mécanisme de visite sans modifier leur code, grâce à un aspect. On se contentera d'un visiteur qui compte les fichiers de taille plus grande que 10. Si vous ne voyez pas comment faire, vous pouvez lire ces articles :
   https ://www.lirmm.fr/ dony/enseig/RC/DesignPatternsInAspectJ.pdf
   https ://www.lirmm.fr/ dony/enseig/RC/AspectVisitor.pdf.
   Une version plus élaborée est donnée ici :
   `https://inria.hal.science/inria-00458203/file/denier-lobjet06.pdf`.

7. Faire un pont entre java-beans et *aspects* :
   `http://www.eclipse.org/aspectj/doc/released/progguide/examples-production.html`. On y voit comment programmer les propriétés liées des composants `JavaBeans` de façon plus modulaire, avec un *aspect*.

   Vous trouverez le code correspondant dans File>new>other>aspectJ>AspectJ Examples
8. Faites le lien entre annotations (Java) et *aspect*.
   `http://www.eclipse.org/aspectj/doc/released/adk15notebook/annotations.html`
9. challenge : Il est très simple de faire un programme de Trace avec AspectJ (afficher un message avant et après l'exécution d'une méthode), pourrait-on faire la même chose avec une annotation Java ?
10. Vous pouvez faire de l'AOP (aspect-oriented-programming) avec le framework SPRING, essayez.
    Voir
    `https://docs.spring.io/spring-framework/docs/4.3.15.RELEASE/spring-framework-reference/html/aop.html`
    `https://docs.spring.io/spring-framework/docs/2.5.x/reference/aop.html`.
    Autre piste : `https://dzone.com/articles/implementing-a-method-trace-infrastructure-with-sp`

11. Faire le lien avec les intercepteurs utilisables notamment dans Quarkus `https://quarkus.io/guides/cdi#interceptors`.

A vous de jouer pour faire vos tests et acquérir de l'expertise dans les différentes mises en œuvre de AOP.