Master Informatique - HAI818I

Modularité Réutilisation et Composants

Assemblage de Composants JavaBeans - Environnement NetBeans

L'approche à composants étend et améliore l'approche objet souvent sans beaucoup la modifier (voir la mise en oeuvre du couplage faible via des injections de dépendences). L'informatique du web par exemple utilise le schéma MVC [1] depuis l'origine, souvent un peu modifié et remanié (voir ici). Ce TD étudie le modèle de composants Java-Beans (ainsi nommé). Son but profond est d'apprendre à y reconnaitre le schéma MVC. C'est le gage d'une adaptation facile aux autres frameworks existants ou à venir. Le modèle de composant javabeans est devenu une sorte de standard terminologique.

Dans ce TP on voit :
- le concept de composant sur étagère assemblable de façon non anticipée, dans la version opérationnelle *JavaBeans* et dans un environnement donné, ici *NetBeans*.
- le couplage faible via les schémas MVC et *Observer* et la non anticipation via le le schéma *Adapter*.

Ce TP est à réaliser avec l'environnement NetBeans. Télécharger une version récente à `https://netbeans.apache.org/`.

1 Prise en main de NetBeans - une première application

Cette section vous propose une solution pour créer un projet basique avec un composant JavaBean.

Voir les tutoriels :
`https://docs.oracle.com/javase/tutorial/javabeans/index.html`,
`http://docs.oracle.com/javase/tutorial/javabeans/writingbean/index.html`.

— **Une application avec un composant bean écouteur**

Créer une nouveau projet (de nom "SwitchApp" par exemple) de type "java with ant/Java Application". **Ne pas cocher (ou décocher) l'option de création d'une méthode main**.

— Dans ce projet, créer un nouveau package ("p1"). Ne pas utiliser le package par défaut, source de divers problèmes. D'une façon générale toujours utiliser le menu contextuel associé à une entité (control-click ou bouton droit de souris sur l'entité) pour la modifier.

— **Créez un descripteur de composant**

Dans le package `p1`, pour créez un descripteur de composant *Javabean* nommé `ZeroOuUn` , vous avez le choix entre "new Java Class" ou "new JavaBeans Component" [2] (sous-menu "others"). Paradoxalement pour créer un composant écouteur, il est plus simple dans cet environnement de choisir l'option "new Java Class" (créer une classe standard).

Donc dans le package `p1`, créer une classe `ZeroOuUn`, sous classe de `javax.swing.JLabel`. Un *label* est un objet graphique qui possède un texte et une méthode d'affichage que l'on peut invoquer quand le texte change. Une instance de `ZeroOuUn` aura pour texte, soit "zero" soit "un". Définissez sur `ZeroOuUn` une méthode `switchText()` qui change le texte de l'une à l'autre possibilité. Définissez cette classe.

— Compilez votre classe. Je vous suggère l'option "build" dans le menu contextuel du projet "SwitchApp".

— **Placer le composant dans la palette**.

On peut placer une classe ou une instance de classe dans la palette de composants mise à disposition des assembleurs (les programmeurs qui assemblent des composants).

Pour placer la classe `ZeroOuUn` que vous venez de créer dans la palette, l'option la plus simple est l'item *addToPalette* dans le menu *Tools* du menu contextuel de la classe. Ou bien utiliser l'item "Add to Palette", menu global *Tools* de *NetBeans*. Ajoutez à la palette dans la catégorie "Beans".

1. http://www.lirmm.fr/ dony/postcript/SeminalPapers/KrasnerPope88.pdf.
2. Ils veulent bien sûr dire "new JavaBeans Component Class", c'est à dire une classe dont les instances se comporteront comme des composants java-bean. Un composant n'est pas la classe mais une de ses instances. La confusion entre classe et instance dans les discours est malheureusement fréquente, même de la part de professionnels.

L'accès à la palette se trouve dans le menu "Windows/IDE-Tools" ou quand on développe la partie graphique de l'application (voir ci-dessous). Il se peut que la palette soit vide si on n'est pas en situation de l'utiliser.

— **Utiliser un composant de la palette dans une nouvelle application**

Dans votre projet "SwitchApp" [3], ajoutez au package `p1`, une"new Jframe Form" et donnez lui comme nom "SwitchApp". `SwitchApp` va être une application graphique qui pilote une instance de composant `ZeroOuUn` avec un `JButton Swing`. Vous ne devez écrire aucun code pour la réaliser.

Une fois la classe `SwitchApp` créée, l'environnement comprends plusieurs fenêtres importantes. A gauche la fenêtre des projets et en dessous un *navigateur-inspecteur* dont nous reparlerons. Au milieu la fenêtre d'édition avec trois options : "source" ou "design" ou "history" ; parfois les options ne sont pas visibles, il faut alors dans le menu' "view" activer "show editor toolbar". Tout à droite la palette des composants.

— Dans la fenêtre d'édition, le mode "design" permet de créer interactivement une application graphique en **glissant/déposant** des composants (AWT, SWING ou BEANS) et en établissant des connexions entre eux. Le mode "source" permet de consulter et modifier le code généré. Faites des essais. Par exemple, connecter un bouton à une instance de `ZeroOuUn`. Les composants graphiques (AWT, Swing, ou Beans visibles) déposés sont visibles dans la fenêtre d'édition. Les composants Beans invisibles déposés ne sont visibles que dans la fenêtre "navigateur-inspecteur" sous la rubrique ("other components").

Dans le fenêtre d'édition, passez en mode design. Glissez/déposez depuis la palette 1) une instance de `zéroOuUn` et 2) une instance de `JButton`.

— La connexion des composants s'effectue en mode "Connection Mode" (5ième icone de la fenêtre d'édition). Sélectionnez ce mode, puis laissez vous guider par l'interface : cliquez sur le composant source d'évènement (ici le *JButton*) puis sur le composant cible, etc.

— Une fois la connexion réalisée, vous pouvez examiner le code automatiquement généré, bouton *source*. Dans la méthode générée `initComponents()`, vous devez reconnaître et comprendre une implantation du schéma *Adapteur*, différente de l'implantation du historique étudiée en cours, utilisant une sous-classe anonyme.

— Pour tester l'application, utiliser "Run Project" dans le menu contextuel du projet, après avoir fait le "build".

2 Application Compteur graphique, avec un écouteur et un écouté

Reprenez l'exemple du cours `CompteurApp`. Dans un premier temps, vous mettrez sur étagère une classe `CompteurBean` et une classe `AfficheurBean` (afficheur d'entiers). Dans un second temps, réalisez une application graphique avec un compteur, un afficheur et des boutons de contrôle.

— **Fabriquer des composants**

Sont donc à réaliser 2 types de composants.

— Un composant compteur doté des méthodes classiques `incr`, `decr`, `raz` ainsi que des méthods `start()` et `stop()`. Après réception du message `stop()` ou avant réception du message `start()`, un compteur ne fait rien quand on lui demande de changer sa valeur. On pourrait avoir un bouton on/off.

La valeur du compteur doit être une propriété liée.

— Un composant graphique `AfficheurEntier` capable d'afficher des nombres entiers.

— **Assembler des composants**

Réalisez l'application en assemblant des composants instances des classes définies précédemment. A priori pas de code à écrire.

Attention, pour connecter un composant invisible à un autre, il faut faire les connections dans le "navigateur-inspecteur", pas dans la fenêtre de connexion des composants visibles.

Points à constater/comprendre :

— A noter à nouveau l'implantation du schéma de conception "Adapteur" dans laquelle la classe de l'adapteur est remplacée par une simple méthode de la classe `main`.

3. En général le client ne sera pas dans le même package que le composant mais commençons par faire simple.

— Comprendre l'utilisation de l'évènement `PropertyChange` et l'utilisation qui en est faite pour connecter le compteur à l'afficheur. Si vous ne comprenez pas les options qui vous sont proposées pour le passage de paramètres, choisissez celle qui vous permet de donner vous-même le code.

3 Pistes pour poursuivre

- Essayer de faire un descripteur de composants ayant une propriété liée en utilisant l'option "new/others/java beans objects/java beans components".

- Les propriétés liées sont une émanation du MVC. L'implémentation JavaBeans est un peu pénible car il faut déclarer les évènements et écrire les méthodes d'abonnement.

Savez vous comment créer des propriétés liées dans les frameworks que vous utilisez ?

- voir `https://stackoverflow.com/questions/1727603/places-where-javabeans-are-used`
- Testez les propriétés contraintes.
- Nous reparlerons des javabeans avec les EJB et JEE.