Golfier Maxime
Giraud—Pépin Nicolas
Projet 14

Description de l'architecture du projet

Le but de notre projet est de créer un bloc-notes pouvant créer, sauvegarder et supprimer des notes dans une interface conviviale basé sur le principe d'un master détail. Nous avons donc utilisé le binding afin de réaliser ce format.
Pour ce faire nous avons utilisé une classe métier intitulé note et une classe model intitulé travaux. Un travaux est une observable list qui contient des notes. Ces notes sont composées de deux string property et d'un object property. Respectivement, cela correspond au nom d'une note, à son contenu et ainsi qu'au fichier qui lui ait lié.

Le « NotePadController » utilise la classe « Note » pour la création d'une nouvelle Note avec ses informations mentionnées précédemment. La liste observable sera contenue dans la ListView.

La persistance de nos notes se fait via une méthode de la classe « Note » qui est appelée chaque fois que le texte ou le titre de la note est modifié, elle s'exécute donc grâce au binding fait sur ces deux éléments dans le bloc-notes qui déclenche cette méthode lors de la modification du contenu de ces derniers. L'utilisateur n'a donc pas besoin de sauvegarder manuellement sa note, opération qui selon nous doit être non pas explicite mais implicite.

Nous avons également utilisé un pattern Commande afin d'ouvrir notre code à l'ajout d'une nouvelle commande sans modification de code. La première fut le Undo-Redo. Nous avons également eu besoin du pattern Memento pour implémenter cette partie.

Cela se traduit dans notre Controller par une opération execute qui permet de sauvegarder un état pour y revenir dessus. Lorsque l'utilisateur appuie sur l'undo, l'opération s'execute pour donner la partie présente dans le execute.

Concrètement, L'Originator instancie un Memento en passant à son constructeur un snapshot de l'état du système. Le CareTaker est chargé quantàlui de connaître le Memento, afin de pouvoir demander, ultérieurement, la restauration de l'état capturé. Cela se traduit par une interface IMementoCommand aura le rôle d'Originator, l'interface IMemento elle sera le Memento et enfin le CareTaker sera l'interface Iconversation.

IMementoCommand pourra faire la modifcation et la sauvergarde de l'état.
Nous sauvegardons l'état dans une classe notée BeforeAfter utilisant l'interface IMemento. Cette classe est utilisé dans une classe Memento_Command qui elle aura le but de faire le but de faire nos actions (exec,undo,redo).
Cette classe hérite d'une classe qui est implémenter par l'interface IConversation. Cette classe aura un constructeur qui permettra d'initialiser la structure stockant les informations undo et redo (c'est une classe stack).
Pour en revenir à notre pattern Command, nous avons dit que l'interface IConversation serait l'invokator. La classe abstraite est bien évidemment sera alors la Command et notre CommandConcret se fait passer par la classe Memento_Command qui execute nos actions.