

Javadirectory

Synchroniseur de dossiers local et en réseaux local.

Nom | Date | //

# Introduction

Cahier des charges :

Développé un outil permettant de synchroniser le contenue de deux dossiers distincts, c’est-à-dire ;

Soient D et D’ deux dossiers d’un système de fichier, si une modification est faite dans le dossiers D celle-ci est répété dans le dossier D’, de même si une modification est faite dans le dossier D’ celle-ci est répéter dans le dossiers D. Une modification se caractérise par la création, la suppression ou la modification d’un fichier ou d’un dossier.

De plus, la modification doit se déclencher automatiquement sans que l’utilisateur de l’outil n’ait à le faire, pour faciliter le développement elle doit dans un premier temps se déclencher si une modification est réaliser dans D ou D’ puis dans un second temps automatiquement.

En plus de ces spécification, l’outils devra aussi pouvoir synchroniser deux dossiers originaires de deux machines différentes, et les interactions avec l’outils devront pouvoir se réaliser grâce à un interface graphique utilisateur.

N.B. :

L’outil doit être utilisable et robuste et devra comporter un manuel d’utilisation et intégré une fonctionnalité additionnelle non détaillé ci-dessus.

## Interprétation.

Après avoir étudiée le cahier des charges nous nous sommes mis d’accord pour développer une structure de données qui réalisera, pour résumer de façon grossière, les opérations suivantes :

Soient deux dossiers D et D’ d’un même système de fichiers. L’outil que nous allons développer identifiera l’un des dossiers comme étant le dossier source et l’autre comme étant le dossier cible (choix fait par l’utilisateur), puis synchronisera le dossiers cible sur le dossier source (notons que source et cible pourront être inversé à tout moment au choix de l’utilisateur) mais l’inverse ne sera pas possible. Pour ce faire l’outils devra analyser le contenue de la source et de la cible puis ajouter dans la cible les dossiers étant dans la source mais dans celle-ci, supprimer ceux étant dans celle-ci mais pas dans la source et mettre à jours les dossiers qu’elle partage avec la cible mais adoptant les modifications de la source.

Pour conclure, on a décider de se donner une comme restriction que la synchronisation devra se faire seulement dans un sens : d’un dossier source vers un dossier cible. Se que revient à pouvoir la faire dans les deux sens en permutant source et cible selon ce que l’on veut.

Sommaire

# Diagramme de cas d’utilisation …………………………………… n

# Diagramme des classes ……………………………………………… n

# Diagramme de séquence …………………………………………….. n

# Diagramme d’état ………………………………………………………. n

# Diagramme des classes détaillé et restriction OCL………. n

# Diagramme de communications …………………n

# Diagramme de déploiement ………………………………………. n

# Difficultés rencontrées …………………………………………………. n

# Notes à propos du manuel d’utilisation ………………………. n

# Conclusion …………………………………………………………………… n

# Annexe et bibliographie ………………………………………………. n

Cas d’utilisation

Une image contenant diagramme

Description générée automatiquement

Figure 1 : Diagramme des cas d’utilisations // à compléter

En local :

Au lancement de l’application l’utilisateur choisit quels sont les deux dossiers qu’il veut synchroniser puis il sélectionne la période T de synchronisation de ses dossiers.

En réseaux local :

Au lancement de l’application l’utilisateur choisis le dossier qu’il veut synchroniser puis, il doit décider s’il est le client ou le serveur. Car le client représente le dossier cible et le serveur le dossier source et le échanges ne se font que de la source vers la cible. Plus tard il aura la possibilité d’être le client s’il le veut.

Ensuite l’application s’exécute dans un processus infinie tournant en tâche de fond, si ensuite il veut l’arrêter il doit ouvrir l’application pour interrompre le processus.

Diagramme des classes

Une image contenant diagramme

Description générée automatiquement

Figure 2 : diagramme des classes simplifié // A compléter

Ce diagramme des classes simplifié répertorie l’ensemble des classes faisans fonctionner notre application et les relations entre elles.

Notons que l’ont à décider de séparer les opérations réseaux des opérations local.

Fonctionnement temporel

Une image contenant diagramme

Description générée automatiquement

Figure 3 : Utilisation en local // A compléterUne image contenant diagramme

Description générée automatiquement

Figure 4 : Utilisation serveur en réseau local // A compléterUne image contenant diagramme

Description générée automatiquement Figure 5 : Utilisation client en réseau local //A compléter

La figure 3 décrit le fonctionnement global de l’application s’exécutant en local dans un processus infinie. Les figures 4 et 5 décrivent les fonctionnements en réseaux local en du serveur et duc client.

Une image contenant diagramme

Description générée automatiquement

Figure 6 : synchronisation en locale

La figure 6 est une description un peu plus précise des opérations s’exécutant en local pour synchroniser les fichiers.

Une image contenant diagramme, schématique

Description générée automatiquement

Figure 7 : Synchronisation serveur en réseau Local

La figure 6 est une description un peu plus précise des opérations s’exécutant pour synchroniser 2 dossiers en réseaux local du point de vue du serveur.

Une image contenant diagramme, schématique

Description générée automatiquement

Figure 7 : Synchronisation client en réseau Local

La figure 7 est une description un peu plus précise des opérations s’exécutant pour synchroniser 2 dossiers en réseaux local du point de vue du serveur.

Diagramme d’état

// Besoin projet finalisé

Classes détaillées et restriction.

Une image contenant diagramme

Description générée automatiquement

Figure : Diagramme des classes détaillés (cf annexes)

La figure est le diagramme des classes détaillé. On y retrouves aussi toutes les relations existante entres les classes .

N.B. : La classe OperationAdditionnel est une classe privée encapsulée dans ListFile de même pour Accept\_Clients qui est encapsulée dans SynchServeur.

N.B. : Les classes Running Local et RunnintReseaux étendent la classe Thread car on y à surcharger l’opération run pour y implémenter les processus principaux de la synchronisation.

De plus, si la classe Acceptes\_Clients étant la classe thread c’est car l’envoie des fichier une branche à part (processus).

Diagramme de communication

// Besoin projet finalisé

Diagramme de déploiement

Une image contenant diagramme

Description générée automatiquement

Figure : Diagramme de déploiement

La figure caractérise dans les grandes lignes le fonctionnement de l’application en réseaux. Le client connaissant le nom du fichier n il demande au serveur de lui envoyer sa dernière version ce que celui-ci fait si ce fichier existe dans le répertoire cible et ceux jusqu’à que l’ensemble des fichiers soit traité.

Puis le client demande au serveur la liste des fichiers qu’elle doit supprimer (c’est-à-dire les fichier existant côté serveur mais pas côté client) et les supprime.

Difficultés rencontrées

1.L’envoie de fichier en réseau local :

La principale difficulté c’est manifesté lors de cette étape du développement. Lorsque l’on envoyait des à une autre machine part un port unique le premier fichier s’envoyait correctement puis le port étant occuper plus rien ne pouvait transiter par celui-ci. On a d’abord pensée à crée un processus pour l’envoi de chaque fichier puis kill celui-ci à la fin de l’envoie mais on le même problème ce manifesté toujours. Puis on à trouver une solution grossière que l’on à décider de garder, on à décider d’ouvrir un port par envoie de fichier. Ainsi si le dossier considéré compte 100 fichier on ouvrira 100 ports pour transférer les fichiers ce qui en soit n’est pas un problème sachant qu’un ordinateur possède plus de 60 000 ports.

Manuel d’utilisation

Notes sur l’écriture du manuel d’utilisation :

Conclusion

L’application final respecte les fonctionnalités essentielles demandée par le client et réécrite dans le cahier des charges page 2. A chaque étape le structure de données interagie avec le client pour lui permettre de s’orienter vers ce qu’il veut et exécuter le programme de la meilleure et de la plus confortable des façons.

On y à intégrer une fonctionnalité additionnelle qui permet de …………………

Annexes

Une image contenant diagramme

Description générée automatiquement