# Compte rendu d'activité – Benjamin HARMAND
Novembre 2015

L'architecture de Generic System était synchrone. Le système était donc très robuste, mais chaque requête était bloquante : une demande entraînait un blocage jusqu'à ce que la réponse lui soit retournée.

L'objectif est de créer un second cache, réactif, sur laquelle une interface utilisateur pourra demander pour un objet (Generic) donné, la liste des Generic qui en dépendent. Pour cela, il est nécessaire de passé par le système des promesses (CompletableFuture) introduite en Java 8. Les promesses permettent de faire une action sur les données lorsqu'elles seront disponibles, ce qui correspond à un traitement asynchrone.

Pour que l'interface se met à jour lors d'un changement des données, elle doit être averti de ce changement. Nous devons donc également fournir à l'interface un liste d'observable, prévenant ceux qui l'écoutent lors qu'elle à changée. Pour toujours avertir d'un changement, tous les éléments de la chaine devront être observable.

## 1. Principe de fonctionnement du cache synchrone

Le cache prend la forme d'une chaîne : elle commence à la transaction, se poursuit par plusieurs différentiels et termine par le cache.

La transaction permet de récupérer les dépendances d'un élément, persistées sur le serveur. Cette transaction est encapsulée dans le différentiel de transaction.

Ce différentiel est en bas d'une pile de différentiel. La pile contient au moins un autre différentiel, correspondant au premier niveau du cache. Ce dernier permet de mettre en cache les ajouts et suppressions d’éléments, ainsi que de récupérer les dépendances à partir des dépendances du différentiel du dessous et des éléments ajoutés et supprimés.

Il est possible d'empiler et dépiler des différentiels sur la pile. En dépiler un permet d'annuler les actions effectuer durant celui-ci.

Le cache récupère les dépendances sur le différentiel le plus élevé, obtenant les informations persistées ajustées avec les ajouts et suppressions de l'ensemble de la pile des différentiels.

## 2. Cache asynchrone

Deux versions de la chaîne des dépendances ont étés implémentés. La première ne contient pas de mise en cache, est très léger. La seconde contient des caches, qui nécessitent d'être mis à jour et est plus robuste.

- Chaîne légère

Lien protocole / transaction

La transaction demande au serveur une promesse d'obtenir les dépendances d'un Generic. Cette demande passe par le protocole. La promesse reçue est convertie en Set observable de Generic en utilisant un objet ré-amorçable : à chaque fois que l'on demande le Set de Generic, un nouveau Set est créé. Le Set se met donc à jour de façon paresseuse, lorsque l'on en a besoin. L'avantage de cette approche est qu'il n'y a pas de cache, donc pas de données entretenir. En revanche, chaque appel demande un recalcule.

Lien transaction / différentiel

La transaction est contenue dans un différentiel. Celui-ci se charge de contenir et entretenir le Set obtenue par la transaction. Ainsi, lorsqu'elle est averti d'un changement de transaction, elle utilise la propriété de ré-amorçage pour obtenir le Set à jour. Ce différentiel est observable, il avertira donc ceux qui l'écoutent lorsque le Set qu'il contient a changée. Ce différentiel particulier est le premier élément de la pile des différentiels.

Lien différentiel / différentiel

Au dessus du différentiel lié à la transaction, au moins un différentiel contient les Generic ajoutés et supprimés par l'utilisateur. Il est possible d'empiler et dépiler des différentiels sur la pile. En dépiler un permet d'annuler les actions effectuer durant celui-ci. Lorsqu'un utilisateur ajout ou supprime un Generic, l'information est toujours enregistré dans le différentiel le plus élevé dans la pile.

Un différentiel calcul les dépendances d'un Generic en utilisant les dépendances du différentiel le précédant, auxquelles il applique les ajouts et suppressions enregistrés.

Chaque différentiel écoute les invalidations indiquées par celui en dessous de lui. Lors de l'ajout ou la suppression d'un Generic, le différentiel alerte ceux qui l'écoutent. De plus, lorsqu'un différentiel est averti d'une invalidation, il transmet cette information si le Generic l'ayant provoqué n'a pas été supprimé dans le différentiel.

Lien différentiel / cache

Lors de la demande des dépendances au cache, celui-ci va lui même les demander au différentiel le plus élevé dans la pile. Ensuite, il va convertir le Set observable en liste observable. Cette action est possible car nous avons remonté les méthodes « get » et « size » du Set. Cette liste observable se met à jour lors d'une invalidation de différentiel, et lors d'un changement de différentiel (empilement ou dépliement).

A chaque appel de demande des dépendances, toute la chaîne est parcouru (de la transaction au dernier différentiel), et la liste retournée est toujours différente.

Avantages / inconvénients

L'avantage de cette implémentation est qu'elle est légère. Lorsqu'une chaîne n'est plus utilisée, elle est supprimée. En revanche, elle génère une

nouvelle chaîne d'écoute à chaque appel, indépendante. Les invalidations sont donc propagées en parallèle sur les différentes chaînes, ce qui est lourd. Une solution pourrait être de mettre en cache la liste observable envoyée par le cache, afin d'envoyer toujours la même.

Un autre problème, plus gênant encore, est qu'en consultant la liste juste après l'avoir obtenue, celle-ci est vide. Il est donc nécessaire d'attendre avant d'avoir les éléments, mais sans savoir combien de temps sera nécessaire.

- Chaîne robuste

Lien protocole / transaction

La transaction demande au serveur une promesse d'obtenir les dépendances d'un Generic. Cette demande passe par le protocole. La promesse reçu est stockée et retournés. Ainsi, lors d'une demande de dépendances pour le même Generic, il suffira de retourner la promesse stocké. Ce niveau de cache permet de gagner du temps. Lorsque la transaction est invalidée, la liste est vidée car elle ne contient plus les informations à jour.

Lien transaction / différentiel

La transaction est contenue dans un différentiel. Lorsque d'un changement de transaction, elle change celle qu'elle contient. Ce différentiel est observable, il avertira donc ceux qui l'écoutent à chaque changement de transaction. Ce différentiel particulier est le premier élément de la pile des différentiels.

Lien différentiel / différentiel

Au dessus du différentiel lié à la transaction, au moins un différentiel contient les Generic ajoutés et supprimés par l'utilisateur. Il est possible d'empiler et dépiler des différentiels sur la pile. En dépiler un permet d'annuler les actions effectuer durant celui-ci. Lorsqu'un utilisateur ajout ou supprime un Generic, l'information est toujours enregistré dans le différentiel le plus élevé dans la pile.

Un différentiel met à jour les dépendances d'un Generic en écoutant le différentiel le précédant. Il écoute également les invalidations générées lors des ajouts et suppressions de Generic, en filtrant ceux non liés au Generic demandé. A chaque invalidation, le recalcule l'ensemble des dépendances et s'invalide pour signaler un changement.

Lien différentiel / cache

Lors de la demande des dépendances au cache, celui-ci va lui même les demander au différentiel le plus élevé dans la pile. Il va stocker la promesse obtenue dans un cache. Ainsi, lors d'une demande de dépendances pour le même Generic, il suffira de retourner la promesse stockée.

Cette promesse se met à jour lors d'une invalidation de différentiel, et lors d'un changement de différentiel (empilement ou dépliement).

Avantages / inconvénients

L'avantage de cette chaîne est que les demandes déjà enregistrées ont une réponse rapide. En revanche, a chaque modification de la transaction, l'ensemble des chaînes est recalculé. Lors de l'empilement ou le dépliement d'un différentiel,

la fin de toutes les chaîne est recalculée. Enfin, à l'ajout ou suppression de Generic, la fin d'une partie des chaînes est recalculée. Même les chaînes qui ne sont plus utilisées continuent a être entretenues. Il est donc nécessaire de réfléchir à un système de relâchement, permettant de supprimer du cache les dépendances de génériques qui ne sont plus utilisées.

Le principal avantage de cette liste, qui retourne une promesse de liste observable plutôt que des listes, est qu'elle fourni de suite la bonne liste, sans avoir besoin d'attendre.