Compte rendu d'activité – Benjamin HARMAND
Décembre 2015

Un des axes de travail sur Generic System est de rendre réactive la chaîne entre le modèle de persistance et le client. Lorsque le client fait une requête, il obtient une liste de résultat qui s'invalide et se met à jour automatiquement lors d'un changement des données persistées. Ainsi, si le client est une interface graphique, elle est avertie d'un changement et met à jour les informations affichées.

Actuellement, la chaîne est développée en Java, et le client en JavaFX. Afin de faciliter la migration vers Ceylon, il est impératif de ne pas synchroniser. Pour cela, il a été décidé d'utiliser les CompletableFutures introduites en Java 8, correspondant aux promesses de JavaScript et Ceylon.

1. Chaîne d'observation des dépendances

La chaîne d'observation permet de remonter les changements dans le modèle de persistance en utilisant une écoute des listes observables.

Au début de la chaîne, les échanges avec le serveur via une Web-socket Vert.x retournent des promesses de listes de résultats. Ces promesses de résultat sont transformées en listes observables. Les listes sont concaténées et filtrées dans chaque niveau de cache avec les éléments ajoutés et supprimés, pour obtenir en bout de chaîne des listes observables des dépendances.

2. Chaîne d'invalidation des dépendances

La chaîne d'invalidation a pour rôle d'invalider la chaîne d'observation des dépendances lors de changement majeur, par exemple un décalage temporel ou un rollback. Lorsque la chaîne d'observation reçoit une invalidation, elle se recalcule automatiquement et averti d'un changement.

3. Traitements clients

Les traitements clients reposent sur la liste de dépendances. Afin d'utiliser les listes observables obtenue avec la nouvelle chaîne d'observation des dépendances, il est nécessaire de réaliser la décomposition fonctionnelle de l'ensemble des traitements du client.

En fonction de l'implémentation des méthode, la transition vers les listes observables en plus ou moins évidente. Un exemple :

- FlatMap

Un FlatMap appel une méthode pour chaque élément de la liste de référence, puis concatène les listes résultantes.

Un FlatMap sur des listes d'observables perd la notion d'observation. Pour palier à cela, une solution est d'utiliser un ListBinding. La liste de référence est liée au ListBinding. La méthode de calcul de la valeur récupère toutes les listes observables résultantes, les concatène et les lies au ListBinding. Lors d'une invalidation, le lien avec les listes résultantes est supprimé et un nouveau calcul permet d'obtenir la liste et les liens.

Cette méthode est efficace dans le cas de traitements simples, car celui-ci est ré-exécuté à chaque invalidation.

Le calcul des héritages réalisé dans le client, traitement lourd et complexe, repose actuellement sur le principe de fonctionnement énoncé ci-dessus. Il est donc nécessaire de trouver une solution plus légère est élégante pour ce cas.