Anthony Soulier
Polytech'Nice-Sophia
7 novembre 2014

Adaptation des interfaces

User Interface Distribution in Multi-Device and Multi-User Environments with Dynamically Migrating Engines

I. Description du problème

De nos jours chaque utilisateurs possèdent de plus en plus d'appareils connectés différents (ordinateur fixe ou portable, tablette numérique ou smartphone), le temps passé à utiliser ces dispositifs est de plus en plus important, et de nombreuses applications sont disponible sur plusieurs supports en même temps ainsi il est normal que les utilisateurs souhaitent pouvoir interagir parallèlement sur ces différentes plateformes.

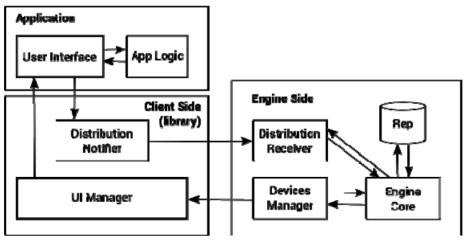
Malheureusement lorsque les développeurs souhaitent réaliser une interface utilisateur (UI) multi-appareils, ils sont limités par les outils de développement de ces plateformes souvent pensés et réalisés pour ne fonctionner que sur un seul et unique type de support, c'est à ce problème que les auteurs ont essayé de répondre.

II. La solution

Pour répondre à cette problématique les auteurs proposent un framework exploitable en application Web et JAVA permettant la création d'une application multi-appareils sur lesquels l'interface utilisateur sera distribuée en fonction de l'état, du type de de support, des groupes auxquels appartient l'utilisateur.

Cette solution repose sur un modèle composé de deux parties distinctes:

• « Engine side » qui va gérer la modification des interfaces, il sauvegarde l'état courant pour que n'importe quels appareils puissent rejoindre la session à n'importe quel moment et ainsi « l'engine side » va pouvoir lui paramétrer son UI.



• « Client side » qui enverra des requêtes à « l'engine side » pour lui signaler un changement état et recevra les mises à jour à appliquer à l'UI.

Lorsqu'un nouvel appareil veut se connecter (**authentification process**) à la distribution, il doit envoyer ses capacités à l'engine, ce qui permet à ce dernier de décider s'il autorise l'appareils à se connecter ou non, dans le cas positif un ou plusieurs groupes lui seront attribués, finalement l'engine va lui envoyé les informations concernant l'UI (distribution state).

La **catégorisation d'un appareil** ce fait par rapport à son type (smartphone, ordinateur portable), souvent associé à ses capacités (taille de l'écran, connectivité disponible) mais également par son rôle (qui est totalement indépendant de son type) représentant les tâches que l'utilisateur peut effectuer.

L'état de distribution (distribution state) est connu par « l'engine side », il contient toutes les informations concernant les états de chaque élément de l'UI, il existe trois états différents:

- invisible qui indique que l'élément n'est pas visible sur l'UI.
- disabled qui indique que l'élément est visible mais non réactif aux événements.
- enabled qui indique que l'élément est visible et réactif aux événements.

Lorsque l'engine va recevoir une requête de mise à jour, il va procéder avec les étapes suivantes:

• Validation de la requête.
• Calcul du nouvel état de distribution.
• Calcul des appareils qui ont besoin d'une mise à jour d'UI.
• Informe ces appareils des changements d'UI nécessaires.

Le framework propose deux types de commandes pour modifier l'état de distribution:

• **Assign Command** utilisée pour modifier l'état d'un élément de l'interface.

• **Feedback Command** qui informe les appareils d'un changement de valeur d'un élément (par exemple remplissage d'une zone de texte)

Lorsque l'engine side a procédé ces commandes il va répondre par une notification.

Exemple:

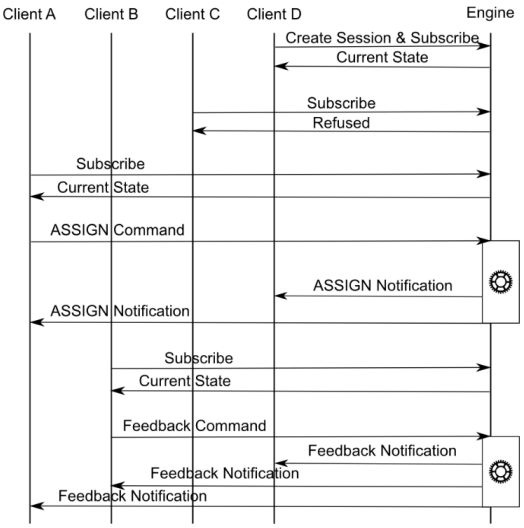


Diagramme de séquence représentant une succession de requêtes.

1. Le client D crée une session et si abonne, l'engine lui renvoi l'état courant de la distribution.
2. Le client C veut souscrire a cette session, l'engine lui refuse la connexion.
3. Le client A veut souscrire a cette session, l'engine accepte et lui envoi l'état courant de la distribution.
4. Le client A envoi une commande ASSIGN, l'engine va calculer les changements nécessaires et les envoyer aux clients qui ont besoin d'une mise à jour (ici le client D et A).
5. Le client B veut souscrire a cette session, l'engine accepte et lui envoi l'état courant de la distribution.
6. Le client B envoi une commande FEEDBACK, l'engine va calculer les changements nécessaires et les envoyer aux clients qui ont besoin d'une mise à jour (ici le client D, A et C).

III. Illustration avec un exemple

Le sujet de cette article correspond parfaitement au cours « Interfaces tactiles et Interfaces Réparties sur plusieurs supports » que nous aurons pendant la prochaine partie de semestre dont le but est de proposé une application nécessitant l'usage de supports différents.

On pourrait par exemple imaginer un Poker 2.0 où la table de poker serait représentée par une table surface de Microsoft sur laquelle serait affichée les différentes informations concernant les joueurs (mise, argent disponible, carte de dos), la mise obligatoire, les carte en jeu dans la rivière qui s'afficheront avec l'interaction du croupier.

Pour interagir avec la table le croupier dispose d'une tablette lui permettant de retourner les carte de la rivière, distribuer le jeu, etc.

La main du joueur sera affiché sur son smartphone, il pourra également miser, se coucher, etc.

Déroulement

La table surface crée la session, tant que le croupier ne s'est pas abonné, les joueurs seront refusé par l'engine. Une fois le croupier abonné, les joueurs pourront s'abonner mais seulement avant la distribution des cartes.

Lorsque le croupier veut distribuer les cartes, l'appuie sur le bouton enverra une commande ASSIGN, que l'engine traitera et enverra une notification ASSIGN à tout les appareils qui ont besoin de se mettre à jour (ici tout les smartphones, et la table de jeu).

Lorsqu'un joueur va miser, une commande FEEDBACK (modification de la valeur misée) sera émise, l'engine va la traiter et envoyer une notification FEEDBACK aux bons appareils (ici la table qui affiche les valeurs des mises joueurs).

IV. Avis

De mon avis, l'interaction entre différents appareils représente l'avenir des interfaces, en effet les appareils étant de plus en plus connectés les uns aux autres il semble évident que les utilisateurs vont vouloir la possibilité de commencer quelque chose sur sur un appareils et le continuer sur un autre.

Depuis iOS 8 et OS X 10.10, Apple propose une API permettant aux développeurs de faire interagir leurs application iPhone, iPad et OS X, le problème de cette solution est qu'il faut développer autant d'application que de support visé et elle est limité aux appareils Apple.

La solution proposé par les auteurs de cette publication a elle l'avantage d'être une seule application pouvant être déployer sur différentes plateformes et dont l'interface s'adaptera au type du support.

Les principaux inconvénients de ce framework viennent surtout de sa jeunesse: non prise en charge d'iOS, un besoin d'améliorer la sécurité des transactions entre appareils.

Je pense donc que ce sont des travaux qu'il faut surveiller mais qu'ils sont encore trop jeunes pour être totalement utilisable pour un développement industriel.