L'adaptation et la négociation dans une architecture de composants JSF

Tayeb Lemlouma

Octobre 2004, Projet WAM, INRIA Rhône Alpes
Tayeb.Lemlouma@inrialpes.fr

1. Introduction

Afin de fournir des composants JSF [1] prêt à être utilisés directement par des appareils de caractéristiques limités tels que les assistants personnels et les téléphones portables ; des mécanismes d'adaptation et de négociation doivent être développés et intégré dans le cycle de vie de traitement d'une requête JSF [1]. L'objectif de la négociation est de trouver une correspondance entre les contraintes des terminaux cibles, les capacités d'adaptation du système et les fonctionnalités des composants d'origine. Le résultat de la négociation est utilisé dans la détermination de l'ensemble des méthodes d'adaptation à appliquer sur les présentations d'origine des composants JSF (Figure 1). Ce document a pour objectif de mettre au clair le positionnement de l'aspect adaptation et négociation dans un cadre de travail JSF.

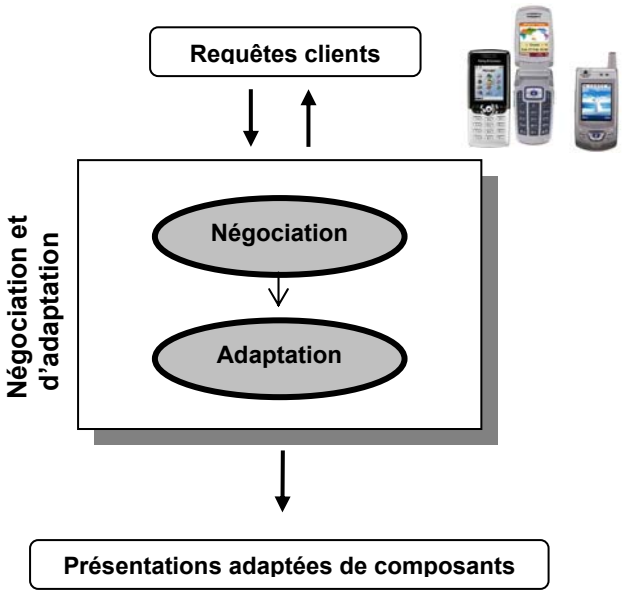

Figure 1 : l'adaptation et la négociation

2. Approches d'adaptation et de développement de composants adaptables

Le développement Java d'applications Web n'est pas nouveau ; beaucoup d'applications Web existantes utilisent les Servlets et les pages JavaServer (jsp). Cependant ces technologies ne fournissent pas de cadre de développement complet. Souvent, c'est le développeur qui se charge de la grande partie du développement de l'application Web au moindre détail ce qui rend le développement long et compliqué. JavaServer Faces (JSF) [1], introduit par la JSR 127 [2], a été défini afin de fournir un cadre standard et productif pour le développement complet d'application Web basée sur Java. Ce cadre permet de faciliter la tache des développeurs et de rendre les applications complètes et facilement réutilisables. Le but d'une stratégie d'adaptation est de fournir des composants utilisables directement par les utilisateurs des environnements hétérogènes où les terminaux utilisés sont sujets de plusieurs limitations matérielles et logiciels. La section suivante discute les approches possibles pour l'intégration d'une stratégie d'adaptation des composants JSF.

2.a Approches possibles

La différence entre l'adaptation universelle, telle qu'utilisée dans l'architecture NAC [3], et l'adaptation des composants JSF réside dans le fait que dans le premier cas, l'adaptation peut être appliquée à une grande variété de formats et de modèles de contenu d'origine en utilisant différents langages cibles alors que dans le deuxième cas, le langage cible de présentation et l'ensemble des composants concernés par l'adaptation est restreint et connu au préalable. Dans le premier cas, le proxy qui s'occupe des adaptations intermédiaires, est conçu d'une manière où la requête du terminal cible peut concerner n'importe quel type de contenu tandis que dans le deuxième cas le module d'adaptation et de négociation, qui sera intégré dans le cadre JSF, visera l'adaptation des présentations des composants mobiles développés dans le cadre de travail JSF.

Nous distinguons donc les deux approches suivantes :

1- **Le Développement de composants adaptables génériques** :

Cette approche consiste à mettre en ouvres des composants JSF complets qui peuvent être utilisés directement sous des plates-formes mobiles.

a. Avantages :

i. Restreindre les aspects adaptation et de négociation aux caractéristiques des différents terminaux mobiles.
ii. Réduire le temps de négociation qui se limitera à la génération des différentes présentations des composants pour les Δ change des contextes cibles
iii. Eviter la négociation des différentes fonctionnalités de composants JSF complexes en exploitant la connaissance préalable du langage cible choisi.

b. Inconvénients

i. Ne pas considérer les composants existants

2- **Le développement de module de négociation et d'adaptation et de *renderer* adaptable** :

Cette approche consiste à maintenir des composants complexes développés avec JSF en gardant toutes leurs fonctionnalités développées pour les environnements traditionnels, riches en caractéristiques matérielles et logicielles.

a. Avantages :

i. Maintenir les composants qui existent sans donner de nouvelles versions mobiles
ii. Rendre les composants adaptables d'une manière automatique

b. Inconvénients

i. Nécessite un traitement d'adaptation et de négociation plus lourd que dans la première approche
ii. L'adaptation est plus importante puisque les composants JSF complexe sont généralement conçus pour les environnements traditionnels.
iii. Beaucoup de fonctionnalités existantes des composants JSF seront filtrés ou adapter puisque l'environnement cible est limité et puisque le langage cible ne sera pas aussi expressive que le langage HTML utilisé par défaut pour représenter les composants JSF

3. L'adaptation et la négociation dans le cadre de travail JSF

L'objectif de l'adaptation et de la négociation pour les composants JSF est de permettre à des utilisateurs de caractéristiques limités de pouvoir utiliser et d'interagir avec ces composants. Afin d'assurer les aspects d'adaptation et de négociation des composants, une approche similaire à celle adoptée dans l'architecture de négociation et d'adaptation NAC [3] peut être appliquée. Rappelons que l'architecture NAC est basée sur l'utilisation d'un proxy intermédiaire qui inclut un module ANM (Adaptation and Negotiation Module) responsable de la négociation et de l'adaptation du contenu dans un environnement hétérogène. Le cadre de travail JSF peut être représenté par trois entités principales qui sont : le TAG, le Composant et le Renderer. Le TAG représente la déclaration du composant dans un langage de marquage tel que XML. Le Composant représente une entité logique de l'application et assure un certain nombre de fonctionnalités. Le Rederer représente l'entité qui génère en final, dans un langage cible, une présentation adéquate de la réponse à une requête cliente.

Le but de l'adaptation est d'assurer, pour un même composant, une présentation qui correspond bien aux caractéristiques du contexte cible, c'est-à-dire les caractéristiques du terminal mobile et de son environnement. Le module d'adaptation et de négociation doit donc se situer entre les deux entités Composant et Renderer cible tel qu'il est montré dans la Figure 2.

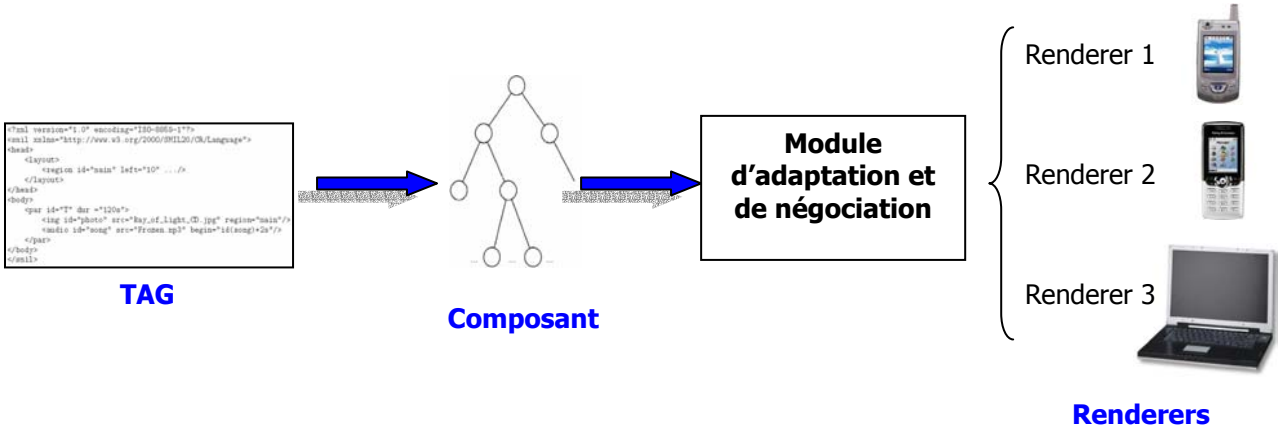

Figure 2 : positionnement du module d'adaptation et de négociation

Il est donc nécessaire d'intercepter les requêtes des clients afin de lancer le traitement du module d'adaptation et de négociation. Le module doit déléguer en final la présentation adaptée du composant à un Renderer spécifique avec les bons paramètres. Ces derniers doivent assurer une bonne prise en compte des dimensions (contraintes élémentaires) du contexte.

Les mécanismes de négociation sont responsables de détecter si une adaptation est nécessaire ou non lorsque une requête est émise par un client. Si le client cible ne supporte pas le composant d'origine la meilleure adaptation doit être appliquée, dans le cas contraire, le Renderer par défaut est utilisé.

Le cycle de vie JSF peut être décomposé en sept phases :

1- L'émission de la requête cliente demandant une ressource JSF
2- Reconstruction de l'arbre des contrôles
3- Application des valeurs de la requête
4- Validation des données saisies
5- Mise à jour des valeurs du modèle d'objets
6- L'invocation d'une application
7- L'affichage de la réponse

Les phases 3, 4 et 5 peuvent passer directement à la phase 7 en générant un message d'erreur. Le module d'adaptation et de négociation doit, dans tous les cas, intercepter tout résultat, que ce soit de validation ou de présentation, afin de générer une réponse qui soit toujours adapté à l'environnement cible.

Il est important que la stratégie de négociation et d'adaptation doit être capable d'intercepter chaque requête utilisateur afin d'assurer le minimum de négociation possible et la meilleure adaptation. Rappelons que l'adaptation et la négociation peut varier d'une requête à une autre même pour le même client [3]. En effet, les ressources demandées, exprimées par une requête, peuvent profondément varier et peuvent déclancher des traitements de dimensions différentes du contexte (extraction de dimensions d'image, bande passante du réseau, taille mémoire disponibles, etc.) et des méthodes d'adaptation variés (retaillage d'images, filtrage de ressources, substitutions de ressources, etc.)

Techniquement l'interception dans le cadre JSF peut se faire en utilisant des écouteurs (listeners). Les listeners peuvent être déclarés dans le fichier de configuration du composant : faces-config.xml.

Exemple : pour déclarer un listener pour la phase d'affichage de réponse il suffit de modifier le fichier de configuration comme suit :

```
<faces-config>
  <lifecycle>
    <phase-listener>
          com.myserver.faces.common.lifecycle.RenderResponseListener
    </phase-listener>
  </lifecycle>
</faces-config>
```

Selon les possibilités techniques offertes par JSF, il est possible que le listener du module d'adaptation et de négociation sera associé à la première phase du cycle de vie JSF ou à la phase d'affichage de réponse.

Les résultats de la négociation sont parfois utilisables durant (et pas seulement avant) l'application d'un renderer spécifique. Ce cas se produit lorsque la méthode de génération d'un résultat d'adaptation est générique (ou paramétrable) [3]. Dans ce genre de situation, il est important d'appliquer une stratégie de cache du résultat, qui sera donc accessible par les Renderers, quand il est nécessaire. L'utilisation du cache n'est pas nécessaire dans le cas où le module de négociation arrive à déterminer une méthode d'adaptation qui, une fois appliquée, assurera une génération complète d'une réponse adaptée telle que l'application d'une feuille de style XSLT.

Références

1. J2EE JavaServer Faces, http://java.sun.com/j2ee/javaserverfaces/index.jsp

2. http://www.jcp.org/en/jsr/detail?id=127

3. Tayeb Lemlouma, Architecture de Négociation et d'Adaptation de Services Multimédia dans des Environnements Hétérogènes, Thèse de doctorat, 9 juin 2004, INPG, Grenoble, France.