Projet Applications réflexives

Une plateforme de services dynamiques BLTi

Introduction

Le projet que vous allez réaliser est une introduction à la problématique des services dynamiques telle que l'offre l'alliance OSGi (d'où le nom-canular de ce projet). Le but est d'offrir un lieu d'échanges entre des **programmeurs** Java développant des services aussi diversifiés qu'un service de mail, une analyse syntaxique XML, la journalisation des activités, etc et des non-programmeurs ayant besoin de ces services. Le framework OSGi est d'une grande complexité et ce projet ne vise qu'à vous aider à terme à mieux comprendre de telles plateformes.
Pour des raisons de clarté, les non-programmeurs seront ici dénommés **amateurs** (à la fois parce qu'ils sont amateurs des services proposés et pour les distinguer des professionnels que sont les programmeurs).
Dans cette description du projet, les parties *(en italique et entre parenthèses)* sont des options - parfois complexes - qui ne concernent que les pires geeks d'entre vous.

La plateforme dynamique de services BLTiLaunch est donc un serveur Java qui accueille les amateurs sur un port et les programmeurs sur un autre. Les services fournis par les programmeurs peuvent être installés, mis à jour, *(arrêtés, démarrés, et désinstallés)* de manière distante par leur créateur-programmeur sans avoir à redémarrer de BLTiLaunch. Les services sont fournis par le programmeur sous forme d'une classe respectant la norme BLTi et que le serveur ira chercher sur le serveur ftp du programmeur.
(Si le service est un regroupement de classes, éventuellement avec des ressources partagées, une bibliothèque .jar avec la règle qu'une seule classe de cette bibliothèque sera une service BLTi et que le bloc static de cette classe définira l'initialisation du service (création des ressources nécessaires au service).)

Deux applications clientes seront donc développées, client_prog et client_ama, pour se connecter à et utiliser BLTiLaunch. Le problème du déploiement des clients ne sera pas abordé dans ce projet.

Programmeur de services

Les programmeurs sont référencés par un login/mdp dans BLTi. Le serveur ftp est censé être d'adresse stable – sauf indication de changement par le programmeur. Un nouveau programmeur devra donner ces informations pour être certifié BLTi et utiliser la plateforme. Cette certification sera ici supposée automatique à la création du compte. (*Des règles de certification pourront être envisagées : parrainage par un membre de l'alliance qui a fourni un code secret à son filleul par exemple.)*
La norme BLTi que doit respecter le programmeur est celle du tp4. Par contre, afin d'éviter des conflits de noms de classe, un programmeur doit mettre tout ce qu'il développe dans un package portant comme nom son login (mon login est *brette*, toutes mes classes sont dans un package *brette* – éventuellement avec des sous-packages qui eux ne répondent à aucune norme).
(Dans le cas où le service est une bibliothèque (.jar), s'ajoute à la norme l'unicité de la classe de service.)

Un programmeur se connecte au port PORT_PROG et, après authentification, indique ce qu'il veut faire :

- Fournir un nouveau service ;
- Mettre-à-jour un service ;
- Déclarer un changement d'adresse de serveur ftp
- *(Démarrer/arrêter un service ;*
- *Désinstaller un service.)*

Le programmeur peut aussi être intéressé par des services existants et les déclencher en tant qu'amateur mais il devra pour ça se connecter au port amateur (voir paragraphe suivant).
Les données nécessaires à chacune de ces options sont suffisamment évidentes pour ne pas nécessiter une description exhaustive.

Amateur de services

L'amateur se connecte au port PORT_AMA et il choisit dans la liste des services que lui communique BLTi celui qu'il souhaite utiliser. (*Seuls les services démarrés lui sont proposés)*. Le lancement du service est réalisé suivant le modèle étudié au TP4. Le code de l'application cliente de l'amateur devra être généralisé au maximum de façon à fonctionner quelles que soient les modalités de communication liées au service rendu.

(Pour bénéficier de certains services (messagerie interne par exemple), l'amateur devra aussi être authentifié par login/mdp).

Un service d'inversion d'un texte

La donnée du service : un texte (String)
Le service : le résultat est l'inversion de la donnée
Le résultat : un texte (String)
Pour ce service, les données/résultats peuvent très facilement utiliser la socket.

Un service d'analyse de fichier xml

La donnée du service : un fichier xml (lire ce fichier sur un serveur ftp ou passer par la socket ?)
Le service : analyser le fichier (règles d'analyse à définir)
Le résultat : un rapport d'analyse (String ou fichier sur serveur ftp ?)

Un service de messagerie interne

Envoi d'un message
La donnée du service : le pseudo du destinataire, le message
Le service : stocker le message dans une ressource du service
Le résultat : une confimation
Lecture des messages :
La donnée du service : aucune
Le service : récupérer les messages de la personne identifié
Le résultat : les messages (String ? sérialisation d'une liste de messages ?)

Cahier des charges du projet (service minimum)

La réalisation demandée doit fonctionner pour des classes de services sans ressources partagées et dont les échanges sont limités à des textes (au minimum donc le service d'inversion et quelques variantes de travail sur une String).
Les services à base de bibliothèques (.jar) et les échanges de données nécessitant un serveur de fichier (dans le cadre de l'exécution du service) sont des options.

Quand, quoi, comment ?

Le projet est à réaliser individuellement ou par binôme. Dans le cas d'un binôme, quelques questions seront posées lors de la dernière séance aux 2 membres du binôme individuellement pour estimer l'apport de chacun au projet. La note finale sera donc individuelle - et, sauf exception, ce sera la même pour les membres du binôme.

Il faut rendre par mail le projet à **jean-francois.brette@parisdescartes.fr** au plus tard le vendredi 24 mars minuit. Le mail devra avoir **UN** fichier attaché nommé des noms des membres du binôme (exemple : le binôme Brette et LeTri enverra un fichier **BretteLeTri.zip**). Une remise non conforme (plusieurs fichiers, un fichier nommé *projet.zip*, etc sera refusée et entrainera un rappel avec 5 pts d'amende.
Le fichier .zip doit contenir tout le code source bien sur ainsi qu'une brève présentation de ce qui a été réalisé, ce qui marche, ce qui ne marche pas, ce qui pourrait être amélioré