ASR9 - Projet de fin d'études

Sujet

Démonstrateur de composition de services répartis hétérogènes avec SCA

Annexe

Limites de l'architecture SCA et de l'implémentation Apache Tuscany

Réalisé par

Mohamed Habib Essoussi et Mohamed Said Mosli Bouksiaa

Limites de l'architecture SCA et de l'implémentation Apache Tuscany

Introduction

Cet annexe du rapport traitera les limites de l'architecture SCA et de son implémentation Apache Tuscany. Ces limites peuvent être soit héritées des technologies utilisées par SCA en l'occurence SOAP soit introduites par SCA et Tuscany eux même.

0.1 Limites héritées des technologies utilisées par SCA

Le standard SCA, quoique conçu pour lever les limites liées à l'hétérogénéité des technologies tel que SOAP, hérite malgré tout quelques "dysfonctionnements" de celui ci. En effet, certains transferts de types complexes tel qu' `ArrayList` de Java semblent encore poser problème avec SCA. En effet, il est impossible de fournir des services retournant des listes ou des `Maps`. La solution longuement présentée pour ce genre de problèmes est de créer une classe qui enveloppe ce type et qui sera donc l'élement à transmettre sur le réseau. Exemple, pour transférer (envoyer/recevoir) une liste de messages (chaines de caractères) sous SOAP, il suffit de créer une classe `Messages` (à titre d'exemple) qui contient entre autre l'attribut `ArrayList<String>`. En aucun il ne faut créer des méthodes décrivant des services et retournant des types tel que `List` ou `Map`.

0.2 Limites liées à l'hétérogénéité

Cette partie traitera toutes les limites rencontrées qui soient liées à l'hétérogénéité des composants.

0.2.1 Transfert de type de complexe hétérogènes

Le transfert de types de complexes hétérogènes semble ne pas être encore au point. Ceci a été clair lors du test de l'envoi d'une liste de messages (chaines de caractère) d'un composant écrit en Python à un composant écrit en Java sous le procotole SOAP. En effet, Cette liste a été enveloppée par une classe en Python pour son envoi sur le réseau et une classe en Java pour la réception de ces messages; contournant ainsi le problème lié à SOAP et décrit précédemment. Or, SCA utilise Jython pour faire la correspondance entre le langage Java et le langage Python. Celui ci n'a pas été en mesure de bien correspondre la classe en Python enveloppant les messages à envoyer avec la classe en Java enveloppant les messages supposés à recevoir. Ceci semble tout à fait insensé, d'autant plus que Jython est conçu pour compiler du code Python en bytecode Java, pour l'exécution de code Python durant le fonctionnement d'un programme Java et surtout pour l'utilisation d'objets Java dans le code Python.

0.2.2 JavaScript ne communique qu'avec des composants locaux

Dans une architecture où un composant JavaScript communiquerait avec un composant distant, soit la mise en œuvre d'un script *cross-domain*, Tuscany se plante ! En effet, JSON -ayant l'avantage de la notation des objets JavaScript- est utilisé pour le transfert des éléments d'un composant à un autre sur le réseau. Également, JSON est souvent présenté comme étant une solution pointue et fiable pour la mise en œuvre d'un script *cross-domain* conjointement à l'installation de proxy. Or le fichier JavaScript généré par Apache Tuscany ne reproduit pas fidèlement l'adresse complète de l'URL précisée au fichier composite, soit l'adresse du serveur qui héberge le service en question. Il ommet la partie "serveur" de l'URL. Ainsi le client JSON-RPC, cherchera son service au serveur d'adresse localhost:8080. D'où la nécessité d'avoir tous les composants en relation avec le composant en JavaScript en local.