DS 2012 Réseau et architectures orientées services

Damien Crémilleux

16 avril 2014

1 Question de cours

Doit-on connaître l'implémentation d'un Web Service pour l'appeler ? Il n'y a pas besoin de connaitre l'implémentation d'un Web Service pour l'appeler, il suffit de connaitre sa description (avec l'aide du protocole WSDL par exemple, *Web Services Description Language*).

Peut-on passer un paramètre par référence à un Web Service ? Non.

Qu'est un *BPEL* Process" BPEL signifie *Business Process Execution Language*, c'est un langage de programmation destiné à l'exécution des procédures d'entreprises. Un BPEL process est un containeur qui permet de décrire le processus, c'est-à-direnotamment les relations avec les partenaires extérieurs, les données, les activitées, le nom et l'espace de nommage.

C'est un programme qui décrit l'enchainement des appels à un Web Service, qui rassemble des web services, decrit la logique des interactions et des appels à ces services, coordonne (orchestre) les interactions entre les services web.

Qu'est SOAP ? SOAP est un protocole pour réaliser des appels à distance, construit par dessus HTTP. Son avantage est qu'il supporte plusieurs plateforme et langages, et est indépendant du protocole de communication.

SOAP (ancien acronyme de Simple Object Access Protocol) est un protocole de RPC orienté objet bâti sur XML. Il permet la transmission de messages entre objets distants, ce qui veut dire qu'il autorise un objet à invoquer des méthodes d'objets physiquement situés sur un autre serveur. Le transfert se fait le plus souvent à l'aide du protocole HTTP, mais peut également se faire par un autre protocole, comme SMTP. Le protocole SOAP est composé de deux parties :

- une enveloppe, contenant des informations sur le message lui-même afin de permettre son acheminement et son traitement,
- un modèle de données, définissant le format du message, c'est-à-dire les informations à transmettre.

SOAP a été initialement défini par Microsoft et IBM1, mais est devenu une référence depuis une recommandation du W3C, utilisée notamment dans le cadre d'architectures de type SOA (Service Oriented Architecture) pour les Services Web WS-*. Le protocole SOAP emploie des métadonnées. SOAP n'est plus un acronyme depuis la version 1.2. En effet, SOAP v1.2 a été réécrit en termes d'infosets XML, et non plus sous forme de sérialisations ¡ ?xml.... ?¿ comme il

l'était en v1.1. La notion d'objet (spécifiée dans Simple Object Access Protocol) devient donc obsolète.

Quel est l'avantage des Web Services sur le RMI Java ? L'avantage des *Web Services* sur le RMI Java est le support de multiples langages (et donc non pas exclusivment du Java). En outre l'utilisation de RMI Java ne permet pas forcément de passer les firewall.

Quelles sont les différences et les similitudes entre un appel de méthodes classiques et un appel RMI ? Le similitudes entre un appel de méthodes classiques et un appel RMI sont la syntaxe de l'appel. Les différences portent sur :

- l'objet doit être *remote* et la méthode décrite dans une *remote interface* ;
- les paramètres peuvent être passés soir par référence (objet *remote*), soit par valeur (en cas de sérialisation) ;
- la gestion d'exception spécifiques aux appels à distance.

Par défaut, l'objet est en copie.

Expliquez la différence entre UDP et TCP UDP est un protocole orienté "non connexion", le flux est unidirectionnel. Le destinataire reçoit les données sans effectuer d'accusé de réception vers l'émetteur. Contrairement à l'UDP, le TCP est orienté "connexion". Le destinataire témoigne de la bonne réception de ces données par un accusé de réception.

Comparez l'utilisation des sockets TCP et du RMI Java Les sockets permettent une communication de bas niveau entre plusieurs machines. Les sockets sont utilisés pour transmettre des données. Quant-à lui, RMI est une API pour effectuer des appels sur des objets à distance en Java. Le niveau d'abstraction est donc plus haut dans ce cas (RMI repose ainsi sur des sockets TCP). RMI signifie l'utilisation de Java des deux côtés, contrairement aux sockets.

Qu'est ce qu'un RFC ? Les *requests for comments (RFC)*, littéralement « demande de commentaires », sont une série numérotée de documents officiels décrivant les aspects techniques d'Internet, ou de différents matériels informatiques (routeurs, serveur DHCP), comme les protocoles. Ainsi le protocole SNMP est décrit dans la RFC 768. Peu de RFC sont des standards, mais tous les documents publiés par l'IETF sont des RFC.

C'est donc une description complète d'un protocole de communication avec un exemple d'implementation

Mode passif de FTP En mode passif, le serveur FTP détermine lui-même le port de connexion à utiliser pour permettre le transfert des données (data connexion) et le communique au client. L'avantage de ce mode est que le serveur FTP n'initialise aucune connexion. Ce mode fonctionne sans problèmes avec des clients derrière une passerelle NAT.

2 Exercice 1 : Sockets

```java
// DS2012 - Damien Crémilleux
package messagerie;

import java.net.ServerSocket;
import java.net.Socket;
import java.util.LinkedList;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.StringTokenizer;

public class Server {
    public static final int PORT= 9988;

    private ServerSocket server;
    private PrintWriter out;
    private BufferedReader in;

    private static final int NB_BOITE = 10;
    private LinkedList<String>[] tabBoite = new LinkedList[NB_BOITE
        ];
    private boolean[] tabEtat = new boolean[NB_BOITE];

    public static void main(String[] args) {
        new Server();
    }

    public Server() {
        super();

        for(int i = 0; i < tabBoite.length; i++) {
            tabBoite[i] = new LinkedList<String>();
        }

        try {
            server = new ServerSocket(PORT);

            while(true) {
                //Ouverture du socket
                Socket socket = server.accept();
                try {
                    out = new PrintWriter(socket.getOutputStream(),
                          true);
                    in = new BufferedReader(new InputStreamReader(
                        socket.getInputStream()));
                } catch (IOException e) {
                    e.printStackTrace();
                }

                lire_requetes();

                //Fermeture du socket
                try {
                    out.close();
                    in.close();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }

    public void lire_requetes() {
        boolean end = false;

        do {
            try {
                String message = in.readLine();
                if(message == null) {
                    end = true;
                } else {
                    // On ne vérifie jamais la cohérence du numéro
                        demandé, ni le fait que la boite ne soit
                        pas vide
                    String[] command = message.split(" ");
                    if(command.length == 3) {
                        if(command[0].equals("SEND")) {
                            int numBoite = Integer.parseInt(command
                                [1]);
                            tabBoite[numBoite].push(command[2]);
                            out.println("OK");
                        } else {
                            out.println("ERREUR");
                        }
                    }

                    if(command.length == 2) {
                        if(command[0].equals("RECV")) {
                            int numBoite = Integer.parseInt(command
                                [1]);
                            String msg = tabBoite[numBoite].
                                removeLast();
                            out.println("MESG "+numBoite+" "+msg );
                        } else {
                            out.println("ERREUR");
                        }
                    }

                    if(command.length != 2 && command.length != 3)
                        out.println("ERREUR");
                }
            } catch (IOException e) {
                end = true;
                e.printStackTrace();
            }
        } while (!end);
    }

}
```

3 Exercice 2 : Java RMI

4 Exercice 3 : Web Services

L'entreprise dispose de plusieurs solutions pour son système informatique :
- logiciels non interopérables répartis sur les agences avec utilisation de copies de fichier. Cette solution, même si elle fonctionne, présente plusieurs

désavantages. Tout d'abord le coût des différents logiciels, renforcé par le fait que ceux-ci ne sont pas interopérables et engendreront donc des problèmes à la prochaine évolution. En outre cela nécessite de former le personnel à ces logiciels spécifiques. L'utilisation d'une solution reposant sur la copie de fichier implique de plus de pas connaitre l'état du stock en temps réel (d'où un risque sur les délais). Enfin, cette solution peut aboutir sur le développement de logiciel spécifique à la copie afin de tenir compte des conversions de format. Cependant, cette solution devrait éviter les risques d'inconsistence.

- Utilisation d'une solution Java avec communication par RMI. Cette solution est elle aussi fonctionnelle et permet de prévenir les risques d'inconsistence. En outre, le choix de cette technologie permet d'unifier les différentes agences et donc de ne pas avoir à gérer un parc d'application hétérogène. Cette solution impose le choix du langage Java, ce qui est à double tranchant (il n'y a plus qu'une seule technologie à connaitre, mais application non compatible avec autre chose que du Java). L'efficacité et la réactivité du système sont améliorés par rapport à la précédente solution, car Java RMI permet un accès directe aux bases de données et peut égalment gérer les accès concurrentiels.
- Utilisation de Web Services. L'utilisation des Web Services présente les avantages de la solution Java RMI, c'est-à-dire l'efficacité et la réactivité, tout en évitant de s'enfermer dans les technologies Java. L'évolutivité de cette solution est donc meilleure que les deux autres, et permettra de facilement rajouter des services au coeur du système informatique. En outre, grâce à des outils comme BPEL, il est possible pour des non informaticien de comprendre l'architecture du système d'information de l'entreprise.