Parseur XML (2 séances)

Raksmey PHAN

Ce TP vous a pour objectif de vous faire découvrir les fichiers XML. Nous allons créer un programme qui permet de lire et écrire un fichier XML. Puis nous verrons un exemple de requêtes avec un service web.

TP réalisé à partir du tutoriel de N. Cynober (http://cynober.developpez.com/).

I – Lire un parseur XML en Java (45min)

1) Créer un nouveau projet de type *Java Application* nommé *LireParseurXML.*
2) Télécharger la librairie jdom (fichier *jdom.jar*) de l'adresse ci-dessous et intégrer-le dans votre projet *LireParseurXML*.
   Fichier jdom.jar : http://fc.isima.fr/~phan/tp/web_service/jdom.jar
3) Télécharger l'exemple de fichier XML ci-dessous et le mettre dans la racine de votre projet : http://fc.isima.fr/~phan/tp/web_service/ParseurXML_Exercice1.xml
4) Modifier votre Main.java pour qu'il ressemble au code suivant et vérifier que le programme affiche bien les noms et les prénoms des personnes dans le fichier XML.

```
package parseuxml;
import java.io.*;
import org.jdom.*;
import org.jdom.output.*;
import org.jdom.input.*;
import org.jdom.filter.*;
import java.util.List;
import java.util.Iterator;

/**
 *
 * @author Phan
 */
public class Main {
    static org.jdom.Document document;
    static Element racine;

    public static void main(String[] args) {
        SAXBuilder sxb = new SAXBuilder();
        try {
            document = sxb.build(new File("./ParseurXML_Exercice1.xml"));
        }
        catch(Exception e){}

        racine = document.getRootElement();

        afficheALL();
    }

    static void afficheALL() {
        List listEtudiants = racine.getChildren("etudiant");

        Iterator i = listEtudiants.iterator();
        while (i.hasNext()) {
            Element courant = (Element) i.next();
            System.out.print(courant.getChild("nom").getText() + " ");
```

```
            System.out.println(courant.getChild("prenom").getText());
        }
    }
}
```

5) Commenter le code.
6) Télécharger le fichier XML suivant :
   http://fc.isima.fr/~phan/tp/web_service/ParseurXML_Exercice2.xml
   Au code précédent, ajouter une fonction *FiltreAffficheAll()* qui vous permet de lire dans le fichier *ParseurXML_Exercice2.xml* tout en se restreignant seulement aux étudiant de la classe *P1*. (hint : utiliser la méthode *Element::getAttributeValue()*)

II – Créer un fichier XML (45min)

1) Créer un nouveau projet de type *Java Application* nommé *EcrireFichierXML*.
2) Copier et exécuter le code suivant :

```
package ecrirefichierxml;
import java.io.*;
import org.jdom.*;
import org.jdom.output.*;

/**
 *
 * @author Phan
 */
public class Main {
    static Element racine = new Element("personnes");
    static org.jdom.Document document = new Document(racine);

    public static void main(String[] args) {
        Element etudiant = new Element("etudiant");
        racine.addContent(etudiant);

        Attribute classe = new Attribute("classe", "P2");
        etudiant.setAttribute(classe);

        Element nom = new Element("nom");
        nom.setText("Nom1");
        etudiant.addContent(nom);

        try {
            XMLOutputter sortie = new XMLOutputter(Format.getPrettyFormat());
            sortie.output(document, new FileOutputStream("./Exemple1.xml"));
        } catch (java.io.IOException e) {
        }
    }
}
```

3) Commenter le code.
4) Ecrire un programme qui vous permet de générer le fichier XML suivant :

```
<personnes>
   <etudiant classe="P1">
      <nom>Nom1</nom>
      <listprenom>
         <prenom>Prenom11</prenom>
         <prenom>Prenom12</prenom>
      </listprenom>
   </etudiant>
   <etudiant classe="P1">
      <nom>Nom2</nom>
      <listprenom>
         <prenom>Prenom2</prenom>
```

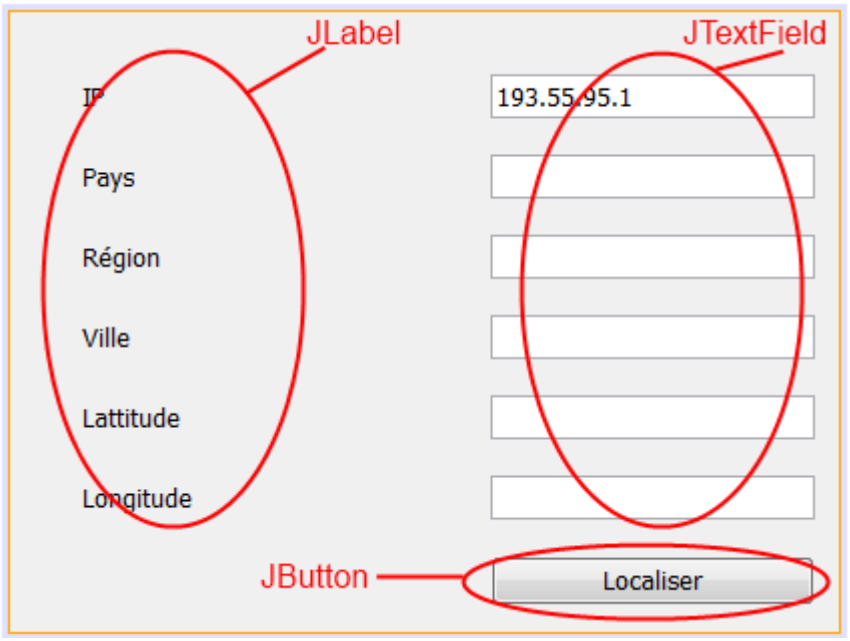
```
      </listprenom>
   </etudiant>
   <etudiant classe="P2">
      <nom>Nom3</nom>
      <listprenom>
         <prenom>Prenom3</prenom>
      </listprenom>
   </etudiant>
</personnes>
```

III – Utiliser un service de localisation à partir d'une adresse IP. (1h30)

IpInfoBD permet de localiser le lieu de votre connexion en utilisant votre adresse IP. Il fournit également une API qui permet d'interroger sa base de données à partir d'une Application cliente écrite en Java. La réponse est reçue sous format XML, qu'il faut « parser » pour obtenir les informations.

1 - Création de l'interface graphique (20min)

1) Créer un projet de type *Java Application*, appelé *IpInfoDB*. Créer ensuite un nouveau fichier nommé *NewJDialog* de type *JDialog Form* (Swing GUI Forms → JDialog Form). Supprimer le fichier *ipinfodb.java* et aller dans la propriété de votre projet pour définir *NewJDialog.java* comme nouveau fichier « main ».
2) Télécharger et inclure dans votre librairie le *.jar suivant : http://fc.isima.fr/~phan/tp/web_service/jdom.jar
3) Dans la *NewJDialog* créer l'interface suivante :



NB : NetBeans génère automatiquement plusieurs lignes de codes qui permettent d'initialiser les éléments de l'interface que vous construisez. Inutile de s'en préoccuper, ce qui est important ce sont les fonctions que l'on va inclure dans le bouton « Localiser ».

4) Importer les packages suivants pour manipuler les fichiers :
   *java.io.**

java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.;*

5) Importer les packages suivants pour le traitement des fichiers xml :
org.jdom.;*
org.jdom.output.;*
org.jdom.input.;*
org.jdom.filter.;*

2 – Interrogation du service web (50min)

Pour dialoguer avec le service web, nous allons utiliser un socket de communication. Nous allons configurer ce socket pour envoyer la requête et lire la réponse. La Figure 1: Schéma de communication avec le service web IPInfoDB montre l'utilisation du socket dans la communication avec le service web.

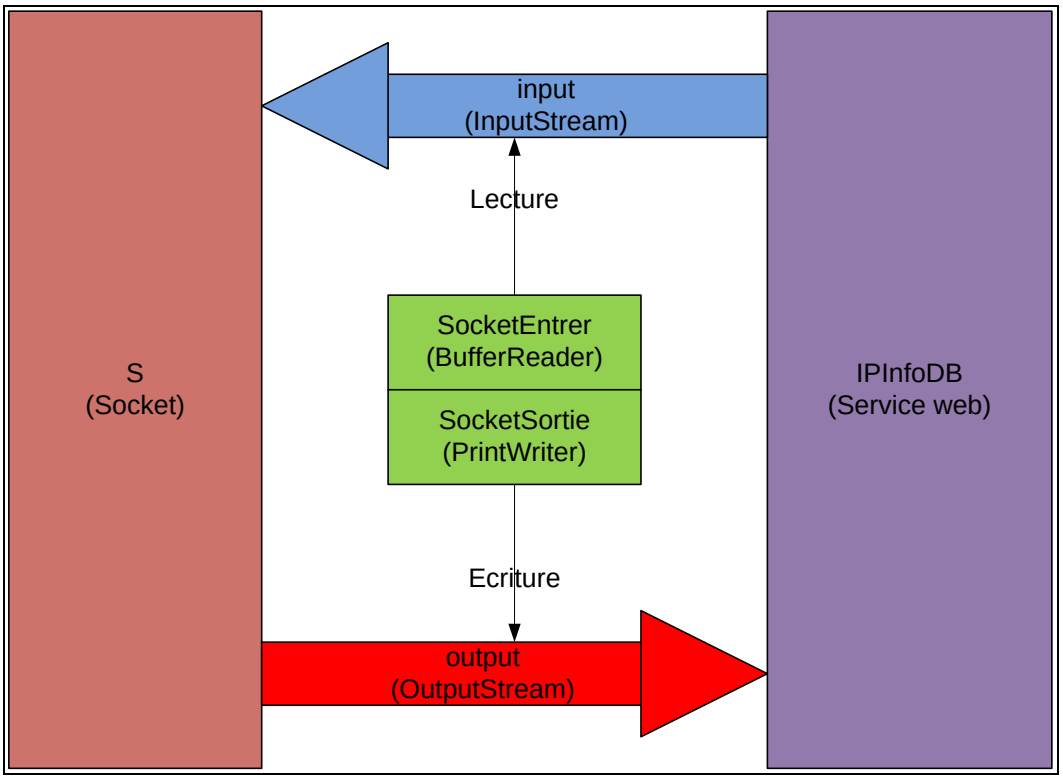


Figure 1: Schéma de communication avec le service web IPInfoDB

En utilisant les aides ci-dessous, écrire un programme pour interroger le service web et pour générer un fichier *xml* en sortie.

Pour écrire créer un fichier *resultat.xml* :

```
File destination;
destination = new File("resultat.xml");
FileOutputStream Ouput = new FileOutputStream(destination);
PrintWriter ecrivain;
```

```
ecrivain = new PrintWriter(Ouput);
```

Pour écrire dans *resultat.xml* :

```
ecrivain.println(responseLine);
```

Pour interroger le service web, on utilise un socket :

```
// Manipulation du socket de réponse du service Web
Socket S = null;
// Coordonnées et numéro de port du service web
S = new Socket("api.ipinfodb.com", 80);
InputStream input = S.getInputStream();
OutputStream output = S.getOutputStream();

// Création de pointeur vers l'entré et la sortie du Socket
BufferedReader SocketEntrer = new BufferedReader(new InputStreamReader(input));
PrintWriter SocketSortie = new PrintWriter(new OutputStreamWriter(output));

// Préparation de la requete pour le webservice
SocketSortie.println("GET
/v2/ip_query.php?key=d57b1e68028f1e5629d553346e48f7f2457981a385726f9c6daf4ddff36f5e
b6" + "&ip=" + adresse + "&timezone=false HTTP/1.0");
SocketSortie.println("Host: api.ipinfodb.com");
SocketSortie.println("Accept: jpg, pictures/gif, pics/jpg, pics/gif, image/x-
xbitmap, pics/jpeg,image/pjpeg, image/png, */*");
SocketSortie.println("");
// Envoie de la requete au web service
SocketSortie.flush();
```

Pour lire le résultat dans le socket :

```
SocketEntrer.readLine();
```

3 – Affichage du résultat (20min)

Parser le fichier *resultat.xml* et afficher le résultat dans l'application java comme le montre la Figure 2: Résultat de l'interrogation du service web IPInfoDB.

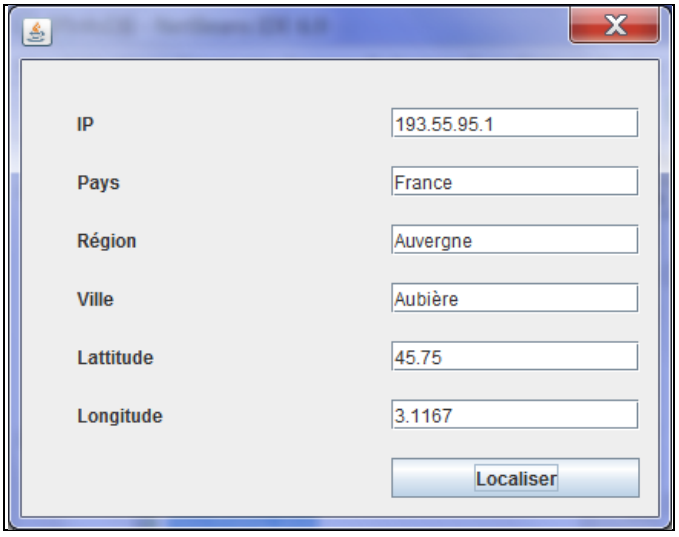


Figure 2: Résultat de l'interrogation du service web IPInfoDB