Université Paris Diderot

XML

Rapport de projet

Réalisé par :
Kheireddine Ouelaa
[redacted]

Responsable :
[redacted]

8 décembre 2014

Résumé

Le but de ce projet était d'écrire une DTD, un schéma XML et une feuille de style XSLT pour la traduction en XHTML de documents généalogiques. Les données généalogiques sont fournies par des fichiers GEDCOM.

Table des matières

1 Présentation

1.1 GedcomToXML

GedcomToXML est le nom de notre programme, et comme son nom l'indique transforme un fichier GEDCOM (voir ci-dessous) en fichier XML valide (validation avec la DTD incluse dans le programme)

1.2 GEDCOM

GEDCOM (acronyme de genealogical data communication, ń communication de données généalogiques ż) est une spécification pour l'échange de données. Il nous a été demandé dont de développer un programme qui parse des fichiers de type GEDCOM v 5.5.1, et de générer ainsi un fichier XML valide [1].

2 Outils

2.1 GEDCOM4J

gedcom4j[2] est une bibliothèque gratuite et open-source qui sert à parser un fichier GEDCOM v 5.5 ou v 5.5.1 [3]. Elle permet aussi de sauvegarder les données obtenues dans un fichier quelconque, pour ce faire, on aura besoin d'une autre bibliothèque que nous détaillerons juste après. La bibliothèque gedcom4j est facile à utiliser voici un exemple qui affiche un couple :

```
// import org.gedcom4j.model.Family;
// charger le fichier à parser
   //.....some code
GedcomParser gp = new GedcomParser();
    gp.load("royal.ged");

      //.....some code
```

---

1. Validation avec DTD et Schéma XML
2. dérnière version mise à jour le 29/10/2014 : *gedcom4j-2.1.9.jar*
3. source : http ://gedcom4j.org/

```
    for (Family f : gp.gedcom.families.values()) {
        if (f.husband != null && f.wife != null) {
            System.out.println(f.husband.names.get(0).basic
                + " married " + f.wife.names.get(0).basic);
        }
      }
```

2.2 JDOM

JDOM (acronyme de l'anglais Java Document Object Model), est une bibliothèque open source pour manipulation des fichiers XML en Java. Elle intègre DOM et SAX, et supporte XPath et XSLT. Elle utilise des analyses syntaxiques externes pour construire les documents. [4]
Cette bibliothèque est aussi facile à manipuler voici un exemple d'un element parent qui a un enfant :

```
// import org.jdom2.*;

// .... some code

// on définit les elements parents et fils 1 et 2
Element parent = new Element("parent");
Element fils = new Element("fils");
// on imbrique les fils dans le parent
parent.addContent(fils);
// on affiche du texte dans l'element fils
fils.setText("je suis le fils");

// .... some code
```

4. source : http ://fr.wikipedia.org/wiki/JDOM

3 DTD

Pour valider les fichiers XML générés par notre programme nous avions besoin d'une DTD, qui sert pour rappel à verifier les éléments XML, leurs occurrences, leur imbrications etc...
Nous avions tout de suite remarqué que les tags Gedcom, n'était pas forcement organisés, et qu'ils apparaissent au plus une fois pour les uns, au moins une fois pour d'autres, et à partir de cela nous avions élaboré une première DTD.
Cette première version validait effectivement des fichiers XML générés par notre programmes, mais pas tous.
Finalement nous avons (après plusieurs tests) fini par tomber d'accord sur une DTD qui nous semblait (et qu'il l'est toujours) valide et qui surtout valide les fichiers XML concernés. Voici un bref aperçu de notre DTD que vous trouverez jointe à ce rapport au nom **gedcom.dtd**.

```
<!--
Cet exemple vérifie qu un événement de type Marriage a, ou pas au
    plus une date et un lieu de célébration
-->
<!ELEMENT marr (date? | plac?)*>
```

4 XML Schema

Le schéma XML est plus explicite que la DTD (expliqué plus haut), il permet aussi de valider tous les fichiers XML concernés, mais n'est pas intégré au programme GedcomToXML. Vous trouverez le fichier joint avec le rapport au nom de **gedcom.xsd**
Voici pour exemple un element plus détaillé que sur la DTD :

```
<!--
Cet element nous montre l element <d> qui est inclut dans <date> et
    qui vérifie que les dates sont bien incluses entre [1,31]
->

<xsd:element name="d" minOccurs="0">
```

```
      <xsd:simpleType>
         <xsd:restriction base="xsd:positiveInteger">
            <xsd:minInclusive value="1"/>
            <xsd:maxInclusive value="31"/>
         </xsd:restriction>
      </xsd:simpleType>
</xsd:element>
```

5 XSLT

Nous sert à afficher le fichier XML généré sous forme de page HTML.
On affiche les deux listes (familles et individus) avec leurs caractéristiques :
Liste des familles :
Nom, prénom, titre, sexe, naissance, baptème, mort, enterrement, objets multimédias.
Liste des individus :
Mariage (lieu et date), divorce, objets multimédias
Le fichier est inclus sous le nom de **xml2html**.

Voici un aperçu d'un individu :

Nom : Spencer Prénom : Diana Frances

Titre : Lady

Sexe : F
Naissance (date : 1 JUL 1961 , lieu : Park House,Sandringham,Norfolk,England)

Fams : F16 Famc : F78
Baptème (date : , lieu : Sandringham,Church,Norfolk,England)

6 Difficultés

6.1 Fichiers mals formés : traité

La difficulté était dans le fichier Gedcom lui même, nous avions des résultats non attendus qui falsifiaient le raisonnement de notre algorithme.
Pour exemple nous avions remarqué que des donnés commençaient par des

espacements au début (dans les dates plus particulièrement), nous avions pu traiter cela avec la fonction *trim()* qui ignore tous les espaces du début et la fin d'une chaine de caractère.

6.2 Tags non traités

Nous avons essayer de traiter le maximum de tags possibles, mais pas tous. En effet il y'a des tags que nous traitons pas dans notre programme, juste par manque de temps.
Toutefois les tags qui sont dans **DATE** et qui ne sont pas des chaines de caractéres connus (pas les mois en l'occurence) sont reconnus par le programme, mais ne sont pas traité, explication :
Les tags en question sont : **BEF,AFT et ABT** qui désignent : before, after et about, donc la date qui est, à priori, donnée après ces tags, sont inexactes, et de ce fait notre programme créé juste un attribut de type **spec** (pour specification) et a toujours la valeur "Not exact".

7 Utilisation du programme

Le programme a été exporté depuis Eclipse sous forme d'une archive .zip. Les librairies cités sont incluses et sont nécessaire au fonctionnement du programme.
Note : Penser à lire le README inclu.

8 Javadoc

On fournit le programme avec sa documentation détaillée. N'hésitez pas à la lire.