Projet XML LSI2

Mohammed Amine Daouma
Adolphe Manga
Kévin Vythelingum

7 novembre 2013

Introduction

A l'heure de la centralisation des données, la fusion de systèmes d'informations est une problématique réelle. Ainsi, nous abordons dans ce TP la fusion de documents XML à travers l'exemple de deux bibliothèques qui souhaitent proposer à leurs employés un système de recherche global via un intranet sans pour autant se séparer de leur ancien système. De plus, cette fusion donnera lieu à l'implémentation d'une nouvelle fonctionnalité : la visualisation des emprunts effectués sur les différents documents.

1 Une nouvelle DTD

On souhaite pouvoir valider les informations en provenance des deux bibliothèques. Pour cela, on écrit une DTD qui permettra de vérifier la cohérence des documents XML à l'aide de la commande suivante :

```
xmllint -dtdvalid modele.dtd document.xml
```

Tout d'abord, nous pensions valider directement les documents XML fournis par les bibliothèques. C'est pourquoi nous avions proposé une nouvelle DTD *biblio-fusion.dtd* se basant sur les DTDs fournies en annexe.

```
<!ENTITY % biblio1 SYSTEM "ex_biblio1.dtd">
<!ENTITY % biblio2 SYSTEM "ex_biblio2.dtd">
<!ELEMENT biblio (bibliotheque|BookList)+ >

%biblio1;
%biblio2;
```

Cependant, cette solution, bien qu'elle valide les documents XML fournis par les bibliothèques, ne correspond pas à l'esprit du TP, notamment en rendant inutile le script XSLT permettant de transformer le document fourni par une des bibliothèques en document conforme à notre DTD globale. En effet, il y serait déjà conforme par cette méthode.

Au regard de ce qui nous est demandé par la suite, nous avons choisi de changer notre façon d'aborder cette DTD globale. Elle met en commun les informations fournies par les deux bibliothèques dans une structure propre. De plus, elle rend possible le système de visualisation des emprunts. Il sera donc nécessaire de modifier les documents XML fournis pour les rendre conforme à cette nouvelle DTD (*biblio.dtd*).

Afin de donner un exemple d'utilisation de cette nouvelle DTD, nous proposons un document XML *ma_biblio.xml* contenant livres et emprunts.

2 Un script XSLT de transformation

Nous avons vu précedemment qu'il était nécessaire de modifier les documents XML fournis par les bibliothèques pour les rendre conforme à notre nouvelle DTD. Nous allons donc écrire un script XSLT *biblio2.xslt* pour transformer les documents fournis par la bibliothèque *biblio2*. Son fonctionnement est décrit dans les commentaires du code.

Le principe est de récupérer à l'aide d'une boucle les informations fournies dans le document XML de la bibliothèque et de les mettre en forme en suivant la structure de notre DTD. Pour générer le nouveau document XML *books_transform.xml*, il suffit d'effectuer la commande suivante :

```
xsltproc -o books_transform.xml biblio2.xslt books.xml
```

On vérifie que le document XML généré est effectivement valide à la DTD globale :

```
xmllint -dtdvalid biblio.dtd books_transform.xml
```

Le script XSLT de transformation réalisé est donc validé. On peut maintenant exploiter ces nouvelles informations centralisées.

3 Exploitation des informations

Pour exploiter les informations fournies dans le document XML *ma_ biblio.xml*, nous avons réalisé plusieurs scripts XSLT dont le fonctionnement est décrit dans les commentaires du code source. Ils génèrent une page HTML en les exécutant de la manière suivante :

```
xsltproc -o x.html x.xslt ../ma_biblio.xml
```

Pour les valider, nous avons choisi judicieusement les informations contenues dans ma_ biblio.xml afin de traiter tous les cas d'utilisation possibles.

a) Nous avons plusieurs livres, dont un ne comporte pas d'éditeur et un possède plusieurs auteurs. Nous observons que la liste contient tous les livres, que tous les auteurs s'affichent en cas de co-signature et que l'éditeur ne s'affiche que s'il existe.

b) Nous enregistrons les livres dans le document XML avec un ordre quelconque des ISBN et nous affichons les numéros ISBN pour vérifier qu'ils ont bien été ordonnés.

c) Nous avons plusieurs emprunteurs dont certains ont emprunté plusieurs livres et nous observons que chacun n'apparaît qu'une seule fois.

d) Parmi les livres, seulement certains ont été empruntés et avec un nombre de fois différent. Nous observons que seuls les livres empruntés s'affichent et que le nombre d'emprunts correspond aux informations du document XML.

e) Nous avons des livres empruntés de divers catégories et secteurs. Nous les affichons afin de pouvoir vérifier le bon résultat. Nous observons qu'ils sont effectivement bien classés, soit par catégories, soit par secteurs.

4 Un SGBD relationnel

Le schéma relationnel est basé sur les informations que requiert la DTD globale *biblio.dtd*. Ainsi, nous proposons ce schéma, adapté aux informations manipulées sur l'intranet :

- livre(ISBN, titre, auteur, éditeur, secteur, catégorie)
- categorie(code, description)
- emprunt(id, refLivre, refEmprunteur, date-emprunt, date-retour)
- abonne(id, nom, prenom, date-inscription)

Il est utile de préciser que dans ce schéma, livre[ISBN] correspond à emprunt[refLivre], que livre[categorie] correspond à categorie[code] et que emprunt[refEmprunteur] correspond à abonne[id]. Nous avons ajouté *abonne* pour modéliser les abonnés des bibliothèques que l'on suppose enregistrés dans un document XML spécifique.

Par rapport au schéma structuré, le schéma relationnel ne prend pas en compte qu'il existe au moins un livre dans les bibliothèques. De plus, on n'impose pas le renseignement des attributs concernant la description d'une catégorie et le secteur d'un livre.

Conclusion

Le TP nous a permis de faire le tour des nombreuses fonctionnalités principales du langage XML et de nous y familiariser, bien que deux questions nous aient échappé. De plus, nous avons pu remarquer sa souplesse. En effet, il permet de structurer des informations communes pour deux bibliothèques ayant des structures de données différentes, puis de valider les documents produits à l'aide des DTDs. En outre, il permet de générer des documents dans un autre langage, tel le HTML, grâce aux scripts XSLT et aux expressions XPATH.