TP 4 : L'adressage XML-XPath

Lab. 1 : XPATH :

XPath est un langage dont la syntaxe n'a rien à voir avec celle du langage XML. Son but principal est d'exprimer des requêtes pour localiser des parties d'un document XML. Les réponses à ces requêtes seront ensuite introduites dans un *template* résultat (souvent en XHTML/HTML) dans les documents XSLT.

Voici quelques caractéristiques de la version 1.0 :

-XPath est principalement un langage de requêtes dans un arbre XML (on peut réaliser une opération telle que 1+1 mais ce n'est pas le but principal).

-Il est fondé sur des chemins d'accès, ou chemins de localisation. On peut faire l'analogie avec un chamin (*path*) vers un fichier ou bien avec une suite de déplacements pour arriver à une destination (2e à gauche…).

- Il fonctionne en chemin relatif ou absolu, l'absolu étant toujours le passage par la racine. Une erreur classique consiste à effectuer une requête absolue qui fait perdre le contexte courant.

-Il s'applique à des noeuds, des booléens, des nombres et chaînes de caractères.
-Une bibliothèque de fonctions est disponible.
-Le résultat produit est un *NodeSet*, un nombre décimal, une chaîne ou un booléen
-Il existe sous une forme abrégée et non abrégée, les deux formes n'étant pas équivalentes.
La forme non abrégée couvre plus de possibilités.
-L'opérateur | représente l'union de plusieurs expressions XPath.

1. Requête simple :

La syntaxe de base XPath est semblable à l'adressage dans un système de fichiers.
Pour accéder par exemple à la liste des noeuds « nom » de l'annuaire, la syntaxe de la requête est :
/annuaire/personne/nom

Les opérateurs XPath :

Symbole	Role	Exemple
/a/b	Liste des éléments dont le chemin dans l'arborescence correspond à la requête	/annuaire/personne/nom
//	On ne tient pas compte de la profondeur de l'élément dans l'arborescence	//nom
*	Tous les éléments fils de	//personne/*

	la sélection	

2. Requête complexe :

XPath permet aussi de gérer des requêtes plus complexes. Par exemple, pour rechercher le numéro de téléphone de Paul Lafargue dans le document, on utilise la syntaxe suivante :

/annuaire/Personne[nom="Idrisisi"]/telephone

On utilise des expressions entre crochets pour indiquer des conditions sur les résultats obtenus. Dans l'exemple précédent, on a mis comme condition sur les entrées obtenues d'avoir un fils dont le contenu de l'élément nom est égal à « Idrissi».

Dans la suite de cette partie, nous ferons une liste non exhaustive des fonctions de XPath (celles qui nous semblent être les plus importantes).

Exemple :

```
<A>
<B att=”Fr”>
<C>Bonjour</C>
<D/>
</B>
<B att=”Eng”>
<C>Hello</C>
</B>
<D id=”12”>
<G>
<H/>
</G>
</D>
<D id=”10”/>
<EE/>
<DE/>
</A>
```

a. Prédicat :

Un prédicat est composé d'une expression booléenne mise entre crochets []. Il permet d'effectuer un tri sur des éléments obtenus par une requête simple. On peut effectuer un test sur le nom d'un élément, son contenu, ses attributs, sa position dans l'arbre, etc.

Symbole	rôle	Exemple	résultat
[i]	i ème élément de la sélection	/A/*[4]	<D id=”10”/>
[last()]	Dernier élément de la sélection	/A/*[last()]	<DE/>
@att=…	Condition sur les attributs	/A/B[@att='Eng']	<**B** att=”Eng”> <C>Hello</C> </B>

Il est possible de faire des prédicats plus complexes en utilisant les opérations booléennes and et or. Par exemple :

Requête:
//*[@att='Eng' and name()='B']
Réponse:
<B att="Eng">
<C>Hello</C>
</B>

Cette requête XPath recherche tous les éléments dont le nom est 'B' et qui contiennent un attribut att ayant la valeur 'Eng'.

Les prédicats ne se construisent pas seulement à partir d'égalités, et il est possible d'effectuer des opérations dans un prédicat :

Type	Opérateur
Booléen	and, or
Logique	!=, >=, >, <, <=
Opérations	+, -, *, div, mod

b. Les fonctions :

Enfin, XPath fournit un certain nombre de fonctions pouvant être utilisées pour les requêtes plus complexes. Il est possible par exemple de rechercher un élément dont le nom contient une chaîne de caractères, etc. Il existe 4 catégories de fonctions : string, number, boolean et node, se référer a la recommandation du W3C pour en savoir plus Quelques exemples (non exhaustifs) :

Symbole	Rôle	Exemple	Résultat
text()	Valeur d'un élément	//B[@att='Fr']/C / text()	Bonjour
last()	Dernier élément de la sélection	/A/*[last()]	<DE/>
position()	Renvoi l'index de position du noeud relativement au noeud parent.	//A/B[position() =2] selectionne le 2ème fils.	<B att="Eng"> Hello </B>
count()	Compte les éléments de la sélection.	//*[count(*)=2] Liste des éléments ayant 2 fils.	<B att="Fr"> <C>Bonjour</C> <D> </B>
name()	Nom de l'élément	//*[name()='D'] équivalent de //D	<D/> <D id="12"> <G><H/></G> </D> <D id="10"/>
contains()	Condition sur les chaînes de caractère	//*[contains(name(),'E')]	<EE/> <DE/>
starts-with()	Condition sur les chaînes de caractère	//*[starts-with(name(),'D')]	<EE/> <DE/>

L'opérateur .. désigne l'élément parent.
Par exemple :

//nom[name(**..**) = 'personne']
retourne tous les descendants nom dont le parent a pour nom personne.

Lab. 2 : Mise en pratique

Exercice 1 :

À l'aide du document livre.xml :

– trouver la liste de noeuds auteur ;
– trouver la liste de tous les noeuds section ;
– trouver la liste des chapitres de la première section ;
– trouver la liste des attributs du premier auteur ;
– trouver la valeur de l'attribut nom du deuxième auteur ;
– trouver la liste des sections avec deux chapitres ;
– trouver la liste des paragraphes dont le parent a pour titre Chapitre1.

Exercice 2 :

```
<AAA val="racine">
  <BBB val="1"/>
  <BBB val="2" id="id1" name="bbb"/>
  <CCC val="3"/>
  <BBB val="4">
    <BBB val="4.1"/>
    <CCC val="4.2">
      <XXX val="4.2.1">
        <YYY val="4.2.1.1"/>
      </XXX>
    </CCC>
  </BBB>
  <DDD val="5">
    <BBB val="5.1"/>
    <CCC val="5.2"/>
    <DDD val="5.3"/>
  </DDD>
```

```
  <CCC val="6"/>
  <AAA val="7"/>
  <CCC val="8">
    <DDD val="8.1">
      <BBB val="8.1.1" id="id2"/>
      <BBB val="8.1.2" name=" bbb "/>
      <EEE val="8.1.3"/>
      <FFF val="8.1.4"/>
    </DDD>
  </CCC>
  <CCC val="9">
    <DDD val="9.1">
      <BBB val="9.1.1">
        <CCC val="9.1.1.1"/>
        <CCC val="9.1.1.2"/>
      </BBB>
      <BBB val="9.1.2"/>
      <EEE val="9.1.3" id="id3"/>
      <FFF val="9.1.4" id="id4"/>
    </DDD>
  </CCC>
  <BBB val="10" id="id5"/>
  <DDDD val="11"/>
  <EE val="12"/>
  <EEE val="13"/>
</AAA>
```

Ecrire les chemins XPath permettant de localiser les noeuds suivants. Localiser ces noeuds dans l'arbre (en donnant leur valeur).

1) l'élément racine AAA

2) tous les éléments CCC qui sont enfants de l'élément racine AAA

3) tous les éléments BBB qui sont enfants de DDD, qui sont enfants de l'élément racine AAA

4) tous les éléments BBB

5) tous les éléments BBB qui sont enfants de DDD

6) tous les éléments inclus dans les éléments /AAA/CCC/DDD

7) tous les éléments BBB qui ont (exactement) trois ancêtres

8) tous les éléments

9) le premier élément BBB, fils de l'élément racine AAA

10) le dernier élément BBB, fils de l'élément racine AAA

11) tous les attributs id
12) tous les éléments BBB qui ont un attribut id

13) tous les éléments BBB qui ont un attribut name

14) tous les éléments BBB qui ont un attribut

15) tous les éléments BBB qui n'ont pas d'attribut id

16) tous les éléments BBB ayant un attribut id dont la valeur est id1

17) tous les éléments BBB ayant un attribut name dont la valeur est bbb

18) tous les éléments BBB ayant un attribut name dont la valeur est bbb. Les espaces de début et de fin sont supprimés avant la comparaison.

19) les éléments ayant deux enfants BBB

20) les éléments ayant deux enfants

21) tous les éléments dont le nom commence par la lettre B

22) tous les éléments dont le nom contient la lettre C

23) tous les éléments qui ont un nom dont le nombre de caractères est exactement trois

24) tous les éléments qui ont un nom dont le nombre de caractères est strictement supérieur à trois

25) tous les éléments qui ont un nom de un ou deux caractères

26) tous les éléments CCC et BBB

27) tous les éléments EEE qui sont enfants de l'élément racine AAA et tous les éléments BBB

Exercice 3 :

Voici un extrait du fichier qui contient la liste des gagnants du booker prize (liste de livres avec leur auteur et l'année de l'obtention du prix).

```
<?xml version="1.0"?>
<booker>
  <award>
            <author>Kingsley Amis</author>
            <title>The Old Devils</title>
            <year>1986</year>
  </award>
  <award>
            [...]
  </award>
  [...]
</booker>
```

Trouvez les expressions XPath qui retournent les informations suivantes :

a) le cinquième livre dans la liste
b) l'auteur du sixième livre dans la liste
c) le titre du livre qui a gagné en 2000
d) le nom de l'auteur du livre intitulé "Possession"
e) le titre des livres dont "J M Coetzee" est l'auteur
f) le nom de tous les auteurs dont le livre a gagné depuis 1995
g) le nombre total de prix décerné