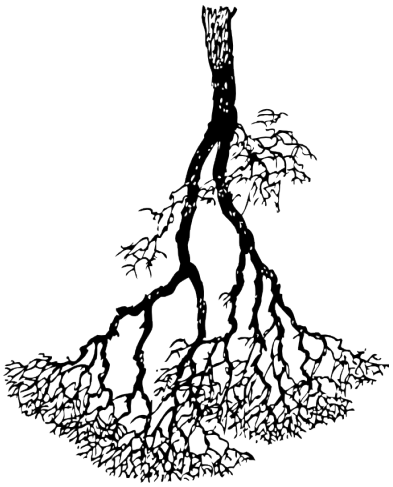
DEPARTEMENT INFORMATIQUE
DE LA FACULTE DES SCIENCES

Quentin Yeche (21520370), Yanis Allouch (21708237)

Rapport du TP N°3 : XPath & XQuery



HMIN103 — Données du web

Référent: Federico Ulliana et Pierre Pompidor

2020

Table des matières

Introduction

L'analyse et le traitement des données consistent à les étudier afin d'en extraire des structures capables des les stocker ainsi que de les manipuler de façon la plus optimale.

L'objectif de ce troisième TP est de se familiariser avec XPath et l'utilisation de l'outil BaseX.

Ce TP traitant uniquement de XPath, est composé de trois exercices. Le premier concerne la base de donnée *Films*, le second travaille sur la base *Recettes* et enfin sur la base de donnée *Trains*.

Le TP se déroule sur papier et machine.
Durant ce TP, nous appliquerons les règles qui seront abordées dans le cours référencé DdW2-XPath.

1 XPath : Films

Voici une DTD modélisant des films, avec acteurs et metteurs en scène.

```
<!DOCTYPE FILMS [
    <!ELEMENT FILMS (FILM+, ARTISTE+) >
    <!ELEMENT FILM (TITRE, ANNEE, GENRE, PAYS, MES, ROLES, RESUME?) >
    <!ELEMENT TITRE (#PCDATA) >
    <!ELEMENT ANNEE (#PCDATA) >
    <!ELEMENT GENRE (#PCDATA) >
    <!ELEMENT PAYS (#PCDATA) >
    <!ELEMENT MES (#PCDATA) >
    <!ATTLIST MES idref CDATA #REQUIRED >
    <!ELEMENT ROLES (ROLE ) >
    <!ELEMENT ROLE (#PCDATA) >
    <!ATTLIST ROLE idref CDATA #REQUIRED >
    <!ELEMENT RESUME (#PCDATA) >
    <!ELEMENT ARTISTE (ACTNOM, ACTPNOM, ANNEENAISS) >
    <!ATTLIST ARTISTE id CDATA #REQUIRED >
    <!ELEMENT ACTNOM (#PCDATA) >
    <!ELEMENT ACTPNOM (#PCDATA) >
    <!ELEMENT ANNEENAISS (#PCDATA) >
] >
```

Donner les requêtes suivantes en XPath.

1. Les titres des films.

```
    //FILM/TITRE
```

2. Les titres des films parus en 1990.

```
    //FILM[ANNEE=1990]/TITRE
```

3. Le résumé d'Alien.

```
    //FILM[TITRE="Alien"]/RESUME
```

4. Quel est le dernier film du document ?

```
    //FILM[last()]
```

5. Quel est le dernier film du document paru en 1990 ?

```
    //FILM[ANNEE=1990][last()]
```

6. Les titres des films qui ont un résumé.

```
    //FILM[descendant::RESUME]/TITRE
```

7. Les titres des films dont l'élément résumé n'est pas présent.

```
    //FILM[not(descendant::RESUME)]/TITRE
```

8. Donnez les noms des acteurs qui ont joué dans Vertigo.

```
    //ARTISTE[@id=//FILM[TITRE="Vertigo"]//ROLE/@idref]/ACTNOM
```

9. Qui a mis en scène Vertigo ?

```
//ARTISTE[@id=//FILM[TITRE="Vertigo"]/MES/@idref]/ACTNOM
```

10. Donnez tout les films du directeur de Vertigo.

```
//FILM[MES/@idref=//FILM[TITRE="Vertigo"]/MES/@idref]
```

11. Donnez les titres des films qui contiennent la lettre "V" (utiliser la fonction contains()).

```
//FILM/TITRE[contains(text(),"V")]
//FILM[contains(TITRE,"V")]/TITRE
```

12. Les titres des films où l'acteur Bruce Willis a joué.

```
//FILM[descendant::ROLE/@idref=//ARTISTE[ACTNOM="Willis" and ACTPNOM
 ="Bruce"]/@id]/TITRE
```

13. Quel rôle joue Harvey Keitel dans Reservoir dogs ?

```
//FILM[TITRE="Reservoir dogs"]//ROLE[@idref=//ARTISTE[ACTNOM="Keitel"
  and ACTPNOM="Harvey"]/@id]
```

14. Qui a joué avec Harvey Keitel dans Reservoir dogs ?

```
//ARTISTE[@id=//FILM[TITRE="Reservoir dogs"]//ROLE/@idref][ACTNOM!='
 Keitel' or ACTPNOM!='Harvey']
```

15. Donnez les noeuds qui ont exactement trois descendants (utiliser la fonction count()).

```
//*[count(descendant::*) = 3]
```

16. Donnez les noeuds dont le nom contient la chaîne "TI" (utiliser la fonction name()).

```
//*[contains(name(),'TI')]
```

17. Quel est le titre du film qui précède immédiatement Shining (dans l'ordre du document) ?

```
//FILM[TITRE='Shining']/preceding::FILM[position()=1]
```

2 XPath : Recettes

Soit le document recette.xml suivant :

```xml
<recettes>
    <recette nomCourt="Chiffonnade" nom="Chiffonnade de jambon et d'asperges a la Flamande" type="salee">
       <materiel>
           <ingredient quantite="8">asperge</ingredient>
           <ingredient quantite="150g">jambon fume</ingredient>
           <ingredient quantite="2">oeuf</ingredient>
           <ingredient quantite="6 cl">huile d'olive</ingredient>
           <ingredient quantite="1 c. a soupe">persil hache</ingredient>
           <ingredient>poivre</ingredient>
           <ingredient>sel</ingredient>
           <ingredient>noix de muscade</ingredient>
       </materiel>
       <methode> A l'aide du hache-legumes, raper les asperges en lanieres d'environ 1,5 mm d'epaisseur et les cuire dans l'eau salee. Couper egalement les tranches de jambon en longues lanieres et les melanger aux asperges cuites et tiedies.
       </methode>
    </recette>
    <recette nomCourt="Pain a l'huile" nom="Presse d'olive sur lit de ble" type="salee">
       <materiel>
           <ingredient quantite="1 baguette">pain</ingredient>
           <ingredient quantite="3 c. a soupe">huile d'olive</ingredient>
       </materiel>
       <methode> A l'aide d'un couteau effile, trancher la baguette sur toute sa longueur.
Badigeonner delicatement chaque tranche avec l'huile, et servir immediatement. Attention, ce plat constitue un repas complet, tout dessert est inutile.
       </methode>
    </recette>
</recettes>
```

Exprimer en XPath les interrogations suivantes.

1. Le nom complet de toutes les recettes.

```
/recettes/recette/@nom
```

2. Les ingrédients de la recette dont le nom court est "Chiffonnade".

```
/recettes/recette[@nomCourt="Chiffonnade"]/materiel/ingredient
```

3. Le nom complet des recettes utilisant du “persil” .

```
/recettes/recette[materiel/ingredient[contains(text(),"persil")]]/@nom
```

4. (Sans utiliser l'axe child) Le nom complet des recettes utilisant du “persil”.

```
//recette[./descendant::*[contains(text(),"persil")]]/@nom
```

5. Le nom complet des recettes ayant plus de deux ingrédients, et contenant des oeufs.

```
//recette[count(.//ingredient)>2 and .//ingredient[contains(text(),"oeuf")]]/@nom
```

6. (Sans utiliser la fonction count()) Le nom complet des recettes ayant plus de deux ingrédients, et contenant l'ingrédient "huile".

```
1    //recette[.//ingredient[1]!=.//ingredient[last()] and .//ingredient
      [2]!=.//ingredient[last()]][descendant::ingredient[contains(text(),"
      huile")]]/@nom
```

7. La dernière recette du document.

```
1    //recette[last()]
```

3 XPath : Trains

L'XML suivant représente des informations ferroviaires : la constitution de trains en voitures, la présence éventuelle d'une voiture-bar, les réservations effectuées, et les usagers correspondants.

```xml
<gare>
    <train numero="t5560" type="TGV">
        <voiture numero="v1">
            <resa numero="r17" id="u55"/>
            <resa numero="r18" id="u52"/>
        </voiture>
        <voiture numero="v2"/>
        <voiture numero="v3"/>
        <voiture numero="v4">
            <bar service="froid uniquement"/>
        </voiture>
    </train>
    <train numero="t6731">
        <voiture numero="v1"/> 2
        <voiture numero="v2">
            <resa numero="r15" id="u55"/>
        </voiture>
    </train>
    <usager id="u55" nom="Jean" prenom="Dufour"/>
    <usager id="u52" nom="Brigitte" prenom="Lefebvre"/>
    <usager id="u56" nom="Patrick" prenom="Subiran"/>
</gare>
```

Exprimer en XPath les interrogations suivantes.

1. Le numéro des trains qui possèdent une voiture-bar ;

```xpath
    //bar/ancestor::train/@numero
    //train[.//bar]/@numero
```

2. Le nom des usages ayant effectué au moins une réservation ;

```xpath
    //usager[@id=//resa/@id]/@nom
```

(Extra) Est-il possible d'exprimer en XPath les requêtes suivantes ?

1. Le numéro des trains dont au moins 2 places sont réservées :

```xpath
    //train[count(.//resa)>=2]/@numero
```

2. Le nom des personnes ayant réservé exactement deux fois.
   Cela semble impossible. Le fonctionnement de l'égalité en XPath ($a = b$ est vrai si b contient un élément égal à a) rend cette question assez épineuse. Si les IDs de resa et usager étaient nommés différemment, la requête aurait été possible, bien qu'assez complexe.
3. Les usagers n'ayant effectué aucune réservation.

```xpath
    //usager[not(@id=//resa/@id)]
```