Données du Web - TD2 1

Basil Dalié - Yannis Naidja

Octobre 2019

Chapitre 1

XPath : Films

1.1 Les titres des films

Donner les requêtes suivantes en XPath. (Voir DTD en annexe)

```
/descendant::TITRE/text()
```

1.2 Les titres des films parus en 1990

```
/descendant::ANNEE[text() =
↪ '1990']/parent::FILM/child::TITRE/text()
```

1.3 Le resume d'Alien

```
/descendant::TITRE[text() =
↪ 'Alien']/parent::FILM/child::RESUME/text()
```

1.4 Quel est le dernier film du document ?

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

1.5 Quel est le dernier film du document paru en 1990 ?

```
/descendant::ANNEE[text() = '1990'][last()]/parent::FILM
```

1.6 Les titres des films qui ont un résumé

```
/descendant::RESUME[text() !=
↪ '']/parent::FILM/child::TITRE/text()
```

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

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

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

```
/descendant::ARTISTE[@id  = /descendant::FILM[TITRE =
↪  'Vertigo']/ROLES/ROLE/@idref]/ACTNOM/text()
```

1.9 Qui a mis en scène Vertigo ?

```
/descendant::ARTISTE[@id = /descendant::FILM[TITRE =
↪  'Vertigo']/MES/@idref]/ACTNOM/text()
```

1.10 Donnez tout les films du directeur de Vertigo.

```
/descendant::MES[@idref = /descendant::FILM[TITRE =
↪  'Vertigo']/MES/@idref]/preceding-sibling::TITRE/text()
```

1.11 Donnez les titres des films qui contiennent la lettre "V" (utiliser la fonctioncontains()).

```
//TITRE[contains(text(),'V')]/text()
```

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

```
descendant::FILM/ROLES/ROLE[@idref =
↪  /descendant::ARTISTE[ACTNOM/text() = 'Willis' and
↪  ACTPNOM/text() = 'Bruce']/number(@id)]
```

1.13 Quel rôle joue Harvey Keitel dans Reservoir dogs ?

```
descendant::FILM[TITRE = 'Reservoir dogs']/ROLES/ROLE[@idref =
↪  /FILMS/ARTISTE[ACTNOM = 'Keitel' and ACTPNOM =
↪  'Harvey']/@id]/text()
```

1.14 Qui a joué avec Harvey Keitel dans Reservoir dogs ?

```
/descendant::ARTISTE[@id = /descendant::FILM[TITRE/text() =
↪  "Reservoir dogs"]/ROLES/ROLE/@idref and ACTNOM/text() !=
↪  'Keitel' and ACTPNOM/text() != 'Harvey']
```

1.15 Donnez les nœuds qui ont exactement trois descendants (utiliser la fonctioncount()).

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

1.16 Donnez les nœuds dont le nom contient la chaîne "TI" (utiliser la fonctionname()).

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

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

```
/descendant::FILM[TITRE =
↪  'Shining']/preceding-sibling::FILM[1]/TITRE/text()
```

Chapitre 2

XPath : Recettes

Exprimer en XPath les interrogations suivantes. Voir document XML en annexe.

2.1 Le nom complet de toutes les recettes

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

2.2 Les ingrédients de la recette dont le nom court est ”Chiffonnade” ;

```
/recettes/recette[@nomCourt =
↪ "Chiffonnade"]/materiel/ingredient/text()
```

2.3 Le nom complet des recettes utilisant du “persil” ;

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

2.4 (Sans utiliser l’axechild) Le nom complet des recettes utilisant du “persil

```
/descendant::ingredient[contains(text(),
↪ "persil")]/ancestor::recette/@nom
```

2.5 Le nom complet des recettes ayant plus de deux ingrédients, et contenant des oeuf

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

2.6 (Sans utiliser la fonctioncount()) Le nom complet des recettes ayant plus de deux ingrédients, et conte-nant l'ingrédient "huile

```
/recettes/recette/materiel/ingredient[last() > 2 and
↪  contains(text(),
↪  "huile")]/parent::materiel/parent::recette/@nom
```

2.7 La dernière recette du document

```
/recettes/recette[last()]/@nom
```

Chapitre 3

XPath : Trains

Exprimer en XPath les interrogations suivantes. Voir document XML en annexe.

3.1 Le numéro des trains qui possèdent une voiture-bar

```
1  /gare/train/voiture/bar/parent::voiture/parent::train/@numero)
```

3.2 Le nom des usages ayant effectué au moins une réservation ;

```
1  /gare/usager[@id = /gare/train/voiture/resa/@id]/@nom
```

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

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

```
1  /gare/train/voiture[count(resa) >= 2]/parent::train/@numero
```

3.4 Le nom des personnes ayant réservé exactement deux fois.

Impossible, car en XPath on ne peut pas compter le nombre d'occurence d'un attribut

3.5 Les usagers n'ayant effectué aucune réservation

```
1   /gare/usager[not(@id = /gare/train/voiture/resa/@id)]
```

Chapitre 4

XPath : Tweets

Reprenez votre DTD pour les Tweets, et créez un document XML valide contenant au moins 3 utilisateurs et 5 tweets. Attention : pour le bon déroulement de l'exercice, il sera peut être nécessaire d'apporter des légèresmodifications à votre DTD afin qu'il soit possible d'interroger vos données !

Donner les requêtes XPath correspondants aux expressions suivantes et évaluer ces expressions dans ledocument XML crée pour les Tweets.

4.1 Les noms des auteurs des tweets.

```
1  /tweeter/users/user[@id =
   ↪ /tweeter/tweets/tweet/@author_ref]/concat(first_name, ' ',
   ↪ last_name)
```

4.2 Les tweets de l'utilisateur dont l'id est "u41".

```
1  /tweeter/users/user[@id =
   ↪ /tweeter/tweets/tweet/@author_ref]/concat(first_name, ' ',
   ↪ last_name)
```

4.3 Les tweets contenants l'hashtag "#I¡3XML".

```
1  /tweeter/tweets/tweet/body/hashtag[contains(text(),
   ↪ 'I<3XML')]/ancestor::tweet
```

4.4 Le tweet le plus recent.

```
1  /tweeter/tweets/tweet/header/date[text() =
   ↪ max(/tweeter/tweets/tweet/header/date)]/ancestor::tweet
```

4.5 Les tweet sans hashtags.

```
/tweeter/tweets/tweet/body[count(hashtag) = 0]/ancestor::tweet
```

4.6 Les retweets du tweet dont l'id est "t42".

```
/tweeter/tweets/tweet[@id = /tweeter/tweets/tweet[@id =
↪ "T42"]/header/retweets/retweet/@ref]
```

4.7 Les utilisateurs ayant répondu au tweet dont l'id est "t42".

```
/tweeter/users/user[@id =
↪ /tweeter/tweets/tweet[@id=/tweeter/tweets/tweet[@id =
↪ 'T42']/header/answers/answer/@ref]/@author_ref]
```

Chapitre 5

Propriétés des requettes XPath

5.1 Reformuler les requêtes suivantes en utilisant exclusivement les axes child, descendant, descendant-or-self, following et following-sibling

5.1.1 Requete 1

```
//b[parent::a]
```

devient

```
//a/b
```

5.1.2 Requete 2

```
//a/preceding-sibling::c
```

devient

```
//c[following-sibling::a]
```

5.1.3 Requete 3

```
//c[preceding::d]
```

devient

```
//d/following::c
```

5.1.4 Requete 4

```
//b/a/preceding-sibling::c/preceding::d
```

devient

```
//d[following::c[following-sibling::a[child::b]]]
```

5.1.5 Requete 5

```
/a/b/..//*/./../preceding::d
```

devient

```
//d[following::*[child::*]]
```

5.1.6 Requete 6

```
//a/ancestor::b/parent::c/child::d/parent::e
```

La requette est syntaxiquement correcte mais ne retourne de résultat pour aucun document XML. Voir explication dans la solution au troisième exercice.

5.2 Reformuler les requêtes //a/following : :b et //a/preceding : :b en utilisant les axes descendant-or-self, ancestor, following-sibling et preceding-sibling.

5.2.1 Requete 1

```
//a/following::b
```

devient

```
//a/ancestor-or-self::node()/following-sibling::node()/descendant-or-self::b
```

5.2.2 Requete 2

```
//a/preceding::b
```

devient

```
//a/ancestor-or-self::node()/preceding-sibling::node()/descendant-or-self::b
```

5.3 Pour chaque requête définie aux points 1 et 2, proposer un document XML pour lequel la réponse à la requête n'est pas vide, sinon expliquer pourquoi un tel document n'existe pas.

5.3.1 Requete 1

```
<r>
  <a>
    <b>
    </b>
  </a>
</r>
```

5.3.2 Requete 2

```
<r>
  <c>
  </c>
  <a>
  </a>
</r>
```

5.3.3 Requete 3

```
<r>
  <a>
    <d>
    </d>
  </a>
  <c>
  </c>
</r>
```

5.3.4 Requete 4

```
  <r>
  <a>
    <d>
    </d>
  </a>
  <c>
  </c>
```

```
  <a>
    <b>
    </b>
  </a>
</r>
```

5.3.5 Requete 5

```
  <r>
  <d>
  </d>
  <a>
    <b>
    </b>
  </a>
</r>
```

5.3.6 Requete 6

Il n'y a aucun document XML correspondant car la fin de la requête est :

```
c/child::d/parent::e
```

Or un noeud de type c ne peut pas avoir d'enfant de type d qui ait un parent e car un noeud n'a qu'un seul parent.

5.4 Donner un document XML pour lequel la requête //r[a[1] = a[2]] n'est pas vide sans que les éléments comparés soient strictement identiques.

```
<r>
  <a>
    abcdef
  </a>
  <a>
    <b>
      abc
    </b>
    <b>
      def
    </b>
  </a>
</r>
```

Les deux noeuds de type a sont égaux en XPath car ils partagent la sous-chaine abc, pourtant il n'y a pas isomorphisme entre les deux sous-arbres et les noeuds textes ne sont pas identiques.

5.5 Est il vrai que, dans le cadre du langage XPath, si X = Y et Y = Z alors X = Z ?

Non, en XPath, l'égalité n'est pas transitive.
Voici un contre exemple :

```
<root>
  <x>
    abc
  </x>
  <y>
    <y1>
      abc
    </y1>
    <y2>
      def
    </y2>
  </y>
  <z>
    def
  </z>
</root>
```

Dans ce document XML, les égalités x = y et y = z sont vraie en XPath mais l'égalité x = z est fausse

Chapitre 6

XQuery : Tweets

Donner les requêtes XQuery correspondants aux expressions suivantes et évaluer ces expressions dans votredocument XML contenant des Tweets.

6.1 Créez une liste de paires tweet-auteur, avec chaque paire contenue dans un element result.

```
<results> {
  for $auteur in /tweeter/users/user
  for  $tweet in /tweeter/tweets/tweet
  where $auteur/@id = $tweet/@author_ref
  return
  <result>
  { $tweet }
  { $auteur }
  </result>
}
</results>
```

6.2 Pour chaque utilisateur, listez le nom de l'utilisateur et la date de tous ses tweets, le tout regroupé dansun élément result.

```
<results> {
  for $auteur in /tweeter/users/user
  return
  <result>
    <nom>
```

```
      { $auteur/user_name/text()}
    </nom>
    {
      for $tweet in /tweeter/tweets/tweet
      where $auteur/@id = $tweet/@author_ref
      return $tweet/header/date
    }
  </result>
}
</results>
```

6.3 Listez les utilisateurs qui ont publié un tweet qui a été retwitté au moins deux fois.

```
<results> {
  for $auteur in /tweeter/users/user
    for $tweet in /tweeter/tweets/tweet
    where $auteur/@id = $tweet/@author_ref and
    ↪  count($tweet/header/retweets/retweet) > 1
    return $auteur
 }
</results>
```

6.4 Pour chaque tweet, listez son contenu et la date de ses deux premières réponses. Rajoutez un element vide <nonRetwitted/>s'il n'a pas été retwitté.

```
<results> {
  for $tweet in /tweeter/tweets/tweet
  return
  <tweet>
    <contenu>
      { $tweet/body/text}
    </contenu>
    <premieres-reponses>
      {
↪  /tweeter/tweets/tweet[@id=$tweet/header/answers/answer[1]/@ref]/header/date
↪  }
      {
↪  /tweeter/tweets/tweet[@id=$tweet/header/answers/answer[2]/@ref]/header/date
↪  }
```

```
    </premieres-reponses>
    {
      if (count($tweet/header/retweets/retweet) = 0) then
        <nonRetwitted />
    }
  </tweet>
}
</results>
```

6.5 Listez les utilisateurs de la plateforme en ordre alphabétique.

```
<results> {
  for $user in tweeter/users/user
  order by upper-case($user/user_name/text()) ascending
  return  $user
}
</results>
```

6.6 Listez les tweets contenants l'hashtag "#I¡3XML".

```
<results> {
  for $tweet in /tweeter/tweets/tweet
  return
  if(contains($tweet/body/hashtags, "#I&lt;3XML")) then
    $tweet
}
</results>
```

6.7 Trouvez le tweet le plus ancien ainsi que le plus recent.

```
<results> {
  let $minDate := min(/tweeter/tweets/tweet/header/date)
  let $maxDate := max(/tweeter/tweets/tweet/header/date)
  for $tweet in /tweeter/tweets/tweet
  where $tweet/header/date = $minDate or $tweet/header/date =
  ↪  $maxDate
    return $tweet
}
</results>
```

6.8 Pour chaque tweet ayant des hashtags, retournez le tweet avec la liste de ses hashtag.

```
<results> {
  for $tweet in /tweeter/tweets/tweet
  return
    if($tweet/body/hashtag) then
    <result>
      {$tweet}
      <hashtags>
        {$tweet/body/hashtag}
      </hashtags>
    </result>
}
</results>
```

6.9 Pour chaque tweet ayant des références utilisateur, retournez le tweet avec la liste des références utilisateur.

```
<results> {
  for $tweet in /tweeter/tweets/tweet
  return
    if($tweet/body/user_ref) then
    <result>
      {$tweet}
      <user_references>
        {$tweet/body/user_ref}
      </user_references>
    </result>
}
</results>
```

6.10 Declarez la fonction local :aReponduAuTweet, qui, étant donné un tweet, retourne tous les utilisateurs qui ont répondu au Tweet.

```
declare function local:aReponduAuTweet($tweet)
{
  for $user in
   ↪ $tweet/parent::tweets/parent::tweeter/child::users/child::user
```

```
4       where $user/@id = $tweet/parent::tweets/child::tweet[@id =
        ↪  $tweet/header/answers/answer/@ref]/@author_ref
5       return $user
6   };
```

Chapitre 7

XQuery : Trains

Donner des expressions XQuery pour les requêtes suivantes.

7.1 Le numéro des trains possédant une voiture-bar.

```
<results> {
  for $train in /gare/train
  where $train/voiture/bar
  return
  <numero>
  {$train/@numero}
  </numero>
}
</results>
```

7.2 Le nom des usages ayant au moins une réservation

.

```
<results> {
  let $resa :=  /gare/train/voiture/resa
  for $user in /gare/usager
  where $user/@id = $resa/@id
  return
  <usager>
  { string($user/@nom) }
  </usager>
}
</results>
```

7.3 La reservation avec le plus grand identifiant (dans l'ordre lexicographique).

```
<results> {
  (for $resa in /gare/train/voiture/resa
  order by $resa/@numero descending
  return $resa)[1]
}
</results>
```

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

```
<results> {
  for $voiture in /gare/train/voiture
  return
  if (count($voiture/resa) >= 2) then
  <train>
  { string($voiture/parent::train/@numero) }
  </train>
}
</results>
```

7.5 Le nom des personnes ayant réservé exactement deux fois.

```
<results> {
  for $user in /gare/usager
  let $count := count(/gare/train/voiture/resa[@id =
  ↪  $user/@id]) return
  if ($count = 2) then
  <usager>
  { string($user/@nom) }
  </usager>
}
</results>
```

7.6 Les usagers n'ayant effectué aucune réservation.

```
<results> {
  let $resa :=  /gare/train/voiture/resa
  for $user in /gare/usager
  where not($user/@id = $resa/@id)
```

```
5       return $user
6     }
7     </results>
```

Annexe A

DTD - Films

```
<!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)>
]>
```

Annexe B

XML - Recettes

```
1   <recettes>
2   <recette nomCourt="Chiffonnade" nom="Chiffonnade de jambon et
    ↪  d'asperges à la Flamande" type="salee">
3     <materiel>
4       <ingredient quantite="8">asperge</ingredient>
5       <ingredient quantite="150g">jambon fume</ingredient>
6       <ingredient quantite="2">oeuf</ingredient>
7       <ingredient quantite="6 cl">huile d'olive</ingredient>
8       <ingredient quantite="1 c. à soupe">persil
        ↪  hache</ingredient>
9       <ingredient>poivre</ingredient>
10      <ingredient>sel</ingredient>
11      <ingredient>noix de muscade</ingredient> </materiel>
        ↪  <methode> A l'aide du hache-legumes, raper les asperges
        ↪  en lanières d'environ 1,5 mm d'epaisseur et les cuire
        ↪  dans l'eau salee. Couper egalement les tranches de
        ↪  jambon en longues lanières et les melanger aux asperges
        ↪  cuites et tiedies.
12    </methode>
13  </recette>
14  <recette nomCourt="Pain à l'huile" nom="Presse d'olive sur lit
    ↪  de ble" type="salee">
15    <materiel>
16      <ingredient quantite="1 baguette">pain</ingredient>
17      <ingredient quantite="3 c. à soupe">huile
        ↪  d'olive</ingredient>
18    </materiel>
```

```
19      <methode> A l'aide d'un couteau effile, trancher la baguette
        ↪  sur toute sa longueur. Badigeonner delicatement chaque
        ↪  tranche avec l'huile, et servir immédiatement. Attention,
        ↪  ce plat constitue un repas complet, tout dessert est
        ↪  inutile.
20      </methode>
21    </recette>
22  </recettes>
```

Annexe C

XML - Trains

```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>
```