XPath

Préparé par
M.G. BELKASMI

Introduction

XPath sert à

- Retrouver des nœuds dans un arbre
  - retrouver des éléments dans un document XML
  - demander (et obtenir) une ressource XML
- Effectuer des opérations sur des données
  - opérations numériques
  - opérations de chaînes de caractères
  - tests booléens

Introduction

Une expression XPath :

- s'évalue en fonction d'un **nœud contexte**
- désigne un ou plusieurs chemins dans l'arbre à partir du nœud contexte
- a pour résultat :
  - un ensemble de nœuds
  - ou une valeur, numérique, booléenne ou alphanumérique

Introduction

- Un chemin XPath est une suite **d'étapes :**
  - *[/]étape$_1$/étape$_2$/…../étape$_n$*
- *Deux variantes :*
  - *Absolu : /FILM/AUTEUR*
    - *Le nœud contexte est la racine du document*
  - ou relatif : RESUME/text()
    - Le nœud contexte est un nœud dans le document (pas forcément la racine).

Introduction

Une étape : trois composants

[axe::]filtre[prédicat1][prédicat2]…

- L'axe : sens du parcours des nœuds (par défault : child)
- Le filtre : type des nœuds/noms des éléments qui seront retenus
- Les prédicats que doivent satisfaire les nœuds retenus

Introduction

Un axe XPath recouvre les deux notions suivantes :

- un sous-ensemble des nœuds de l'arbre relatif au nœud contexte
- l'ordre de parcours de ces nœuds à partir du nœud contexte.

XPath

- La notation abrégée **..** désigne le père du noeud contexte, **quel que soit son type.**
- Équivalent à : parent::node()
- node() est un **filtre** qui désigne tous les types de nœuds (sauf les attributs).
- parent::*

  * est un **filtre** qui désigne tous les *éléments.*

XPath

Self::node()

- il désigne le nœud contexte lui-même.
- Doit être complété par un filtre.
- Permet de reprendre le nœud quelque soit le contexte
- En abrégé : .

XPath

Que représente ?

- Ancestor::node()
- Preceding-sibling::node()
- Following::node()

XPath

Axes :

- nœuds enfants et descendants: child, descendant, descendant-or-self
- nœud parent et ancêtres: parent, ancestor
- frères: preceding-sibling, following-sibling
- nœuds précédents et suivants: preceding, following
- nœud lui même : self

XPath

Filtre :

Deux manières de filtrer les nœuds :

- Par leur nom :
  - possible pour les types de nœuds qui ont un nom : **Element, ProcessingInstruction** et **Attribute**
- Par leur type DOM

XPath

- *Filtrage sur le nom de nœud*
  - */descendant::node()/@att2*
  - *Nom générique : /A/**
- *Filtrage sur le type de nœud*
  - text() : Nœuds de type Text
  - comment() : Nœuds de type Comment
  - processing-instruction(): Nœuds de type directive
  - Exemple :
    - /A/B//text()
    - /comment()
    - /processing-instruction(), ou /processing-instruction('java'),
    - node() : Tous les types de nœud

XPath

Prédicat :

- expression booléenne constituée d'un ou plusieurs tests, composés avec les connecteurs logiques habituels and et or
- Test :
  - toute expression XPath, dont le résultat est converti en booléen;
  - une comparaison, un appel de fonction.
- Il faut connaître les règles de conversion

XPath

Quelques exemples :

- /A/B[@att1]

Les nœuds /A/B qui ont un attribut @att1

- /A/B[@att1=’a1’]

Les noeuds /A/B qui ont un attribut @att1 valant ’a1’

- /A/B/descendant::text()[position()=1]

Le premier nœud de type Text descendant d’un /A/B.

- /A/B/descendant::text()[1]

pareil

XPath

Dans l'expression /A/B*@att1+:

- On s'intéresse aux nœuds de type B fils de l'élément racine A.
- Parmi ces nœuds on ne prend que ceux pour lesquels le prédicat *@att1+ s'évalue à true
- Cette expression s'évalue avec pour nœud contexte un élément B
- [@att1] vaut true ssi @att1 renvoie un ensemble de nœuds **non vide**

XPath

- Une étape s'évalue en tenant compte d'un **contexte** constitué de un nœud contexte, position initiale du chemin
- ce nœud fait lui-même partie d'un ensemble obtenu par évaluation de l'étape précédente
- on connaît la **taille** de cet ensemble (fonction *count())*
- on connaît la **position** du nœud contexte dans cet ensemble (fonctions *position() et last())*

XPath

Typage avec XPath :

- Effectuer des comparaisons et des opérations cela implique un **typage** et des conversions de type**.**
- Types XPath :
  - les numériques
  - Les chaines de caractères
  - Les booléens (true or False)
  - Enfin les ensembles de nœuds

XPath

- Notation décimale habituelle
- Comparaisons habituelles (<, >, !=)
- Opérations : +, -, *, div, mod
- La fonction *number()* permet de tenter une conversion
- Si la conversion échoue on obtient NaN (*Not a Number).* ***À éviter…***
- Exp : //node()[number(@att1) mod 2=1]

XPath

Deux conversions sont toujours possibles.

- Vers une chaîne de caractères
  - utile pour la production de texte en XSLT
  - (balise xsl:value-of)
- Vers un booléen
  - utile pour les tests effectués dans XSLT
  - (xsl:if, xsl:when)

XPath

Conversions booléennes

- Pour les numériques : 0 ou NaN sont false, tout le reste est true
- Pour les chaînes : une chaîne vide est false, tout le reste est true
- Pour les ensembles de nœuds: un ensemble vide est false, tout le reste est true

XPath

Fonctions XPath

Quelques fonctions utiles dans les prédicats :

- *concat(chaîne1, chaîne2, ...)* pour concaténer des chaînes
- *contains(chaîne1, chaîne2)* teste si chaîne1 contient chaîne2
- string-length (ch) : donne la longueur de ch
- *name()* renvoie le nom du nœud contexte
- *not(expression) :* négation
- Count() : le nombre de nœuds dans l'ensemble sélectionné par la requête

XPath

Prédicats et axes d'avancement :

- child::*[3] le 3ème enfant
- child::*[position()=3] idem
- child::*[last()] le dernier enfant
- descendant::*[last()] le dernier descendant
- La position d'un nœud dépend de l'axe choisi.

XPath

La syntaxe abrégée

Syntaxe abrégée ⇔ Syntaxe étendue

. ⇔self::node()

toto ⇔child::toto

../toto⇔parent::toto

@titi ⇔ attribute::titi

//toto ⇔/descendant-or-self::node()/child::toto

.//toto ⇔descendant-or-self::node()/child::toto

toto[2] ⇔ child::toto[position() = 2]

XPath

La syntaxe abrégée

- Attention :

//toto[2]

n'est pas équivalent à

/descendant-or-self::toto[position()=2]

mais à

/descendant-or-self::node()/child::toto[position()=2]

- Autrement dit :
  - //toto[2] fournit tous les nœuds toto qui sont deuxième fils de leur père ;
  - /descendant-or-self::toto[position()=2] désigne un unique nœud, le deuxième nœud toto du document.