

XPath

Cours « Recherche d'Information et Graphes de Données »

Nicolas Malandain et Nicolas Delestre

XPath : un langage d'adressage

XPath est un langage non-xml permettant de sélectionner des nœuds dans l'arborescence d'un document XML.

Les 7 types de nœuds d'un document XML :

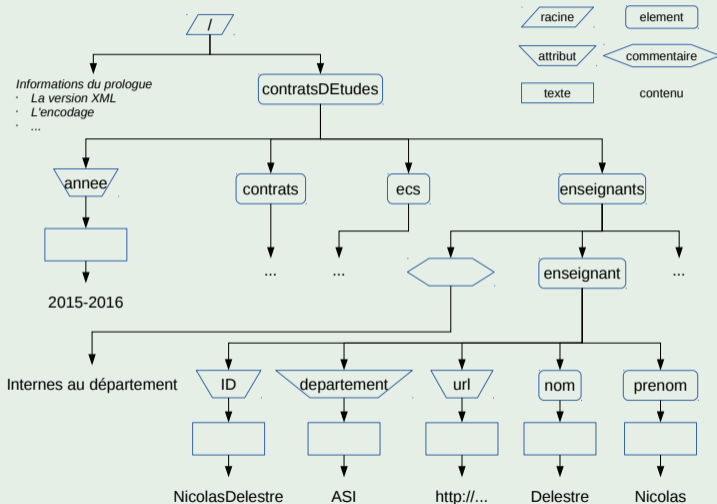
- le nœud racine,
- les nœuds d'élément,
- les nœuds de texte,
- les nœuds de commentaire,
- les nœuds d'attributs,
- les nœuds d'instruction de traitement,
- les nœuds d'espace de noms.

Remarques

- le nœud racine "/" contient tout le document XML et donc inclut l'élément racine
- les appels d'entités, et les sections CDATA ne sont pas adressables



Représentation graphique du XML du TD



Chemins XPath

Objectifs

- Désigner un ou plusieurs nœuds

Syntaxe

- Un chemin est une suite de termes sélectionnant un nœud ou ensemble de nœuds. Les termes sont séparés par "/" .
- Chaque terme sélectionne un nœud, le terme suivant s'y applique pour ajouter une nouvelle condition de sélection.

`indicateur_de_relation::filtre[prédicat1] [prédicat2] ...`

Exemple

`descendant::enseignant[position()=2]`

Sélectionne le deuxième élément du type enseignant descendant du nœud courant



Indicateurs de relation (Axes)

Par rapport au nœud courant

- `self` lui-même
- `child` sélectionne les nœuds fils
- `descendant` sélectionne les nœuds descendants
- `descendant-or-self` équivalent à `descendant` et `self`
- `parent` sélectionne le nœud racine ou le nœud d'élément parent
- `ancestor` sélectionne les ascendants dans l'ordre inverse d'apparition dans le document
- `ancestor-or-self` équivalent à `ancestor` et `self`
- `following-sibling` sélectionne les nœuds frères suivants
- `preceding-sibling` sélectionne les nœuds frères précédents
- `following` tous les nœuds suivant
- `preceding` tous les nœuds précédents
- `attribute` sélectionne ses attributs
- `namespace` sélectionne les espaces nominaux attachés

Filtres (Tests de nœuds)

Filtres pouvant suivre l'indicateur de relation (axe) :

- *nom d'élément* sélectionne le ou les éléments reliés au nœud courant par la relation indiquée
- *nom d'attribut* sélectionne l'attribut du nœud courant (axe attribute)
- `node()` sélectionne tous types de nœuds
- `comment()` sélectionne les nœuds de type commentaire
- `processing-instruction(nom)` sélectionne les instructions de traitement, si le nom n'est pas spécifié les sélectionne toutes
- `text()` sélectionne les nœuds textuels fils du nœud courant

Conditions

- `and`, `or` et `not()` combinaison de prédicats
- opérateurs `=`, `!=`, `<`, `>`, `<=`, `>=`; les termes de ces opérateurs peuvent être des nombres, des chaînes de caractères, des expressions arithmétiques (`+`, `-`, `*`, `div`, `mod`), ou encore des fonctions
- `@attribut` vérifie la présence de l'attribut
- `élément` vérifie la présence de l'élément
- `not(élément)` vérifie l'absence de l'élément

Fonctions sur les nœuds

- `last()` : position du dernier nœud dans le contexte courant
- `position()` : numéro d'ordre du nœud courant dans la liste de nœuds contextuelle
- `count(liste-nœuds)` : nombre de nœuds inclus dans la liste de nœuds (expression XPath)
- `id("liste-identifiants")` : éléments contenant un des identifiants présents dans la liste
- `local-name(liste-nœuds)` : nom (string) du premier nœud rencontré dans le document appartenant à la liste, s'il n'y a pas d'argument nom du nœud courant. Le nom est retourné sans préfixe
- `namespace-uri(liste-nœuds)` : idem mais retourne uniquement le préfixe
- `name(liste-nœuds)` : idem mais retourne le nom complet

Fonctions sur les chaînes de caractères

- `string(objet)` : retourne l'objet sous forme de chaîne de caractères (ex : s'il s'agit d'un élément, convertit tout le contenu de l'élément)
- `concat(ch1, ch2, ...)` : retourne la concaténation des chaînes de caractères
- `starts-with(ch1, ch2)` : est évalué à vrai si *ch1* commence par *ch2*
- `contains(ch1, ch2)` : est évalué à vrai si *ch1* contient *ch2*
- `substring-before(ch1, ch2)`, `substring-after(ch1, ch2)` : retourne la partie de *ch1* située avant/après *ch2*
- ...

Fonctions booléennes

- `boolean(objet)` : retourne vrai si l'objet est un nombre non nul, une liste non vide de nœuds, ou une chaîne de caractères non vide
- `true()` : retourne toujours vrai
- `false()` : retourne toujours faux
- ...

Fonctions numériques

- `number(objet)` : retourne l'objet sous forme de nombre
- `sum(liste-nœuds)` : retourne la somme des valeurs des nœuds
- `floor(nbr)`, `ceiling(nbr)`, `round(nbr)` : fonctions d'arrondis

Quelques abbréviations

<code>self::node()</code>	⇔	<code>.</code>
<code>parent::node()</code>	⇔	<code>..</code>
<code>child::nom</code>	⇔	<code>nom</code>
<code>/descendant-or-self::nom</code>	⇔	<code>//nom</code>
<code>attribute::nom</code>	⇔	<code>@nom</code>
<code>[position()=x]</code>	⇔	<code>[x]</code>

Tout chemin commençant par / est absolu (commence à la racine)

⇒ `//@id` sélectionne tous les attributs `id` du document

⇒ `..//@id` sélectionne tous les attributs `id` à partir du nœud courant

Jokers et sélections multiples

- Jokers
 - * tout nœud d'élément ne sélectionne pas les attributs, les nœuds de texte, les commentaires ou les instructions de traitement
 - @* tout nœud d'attribut
- Sélections multiples
 - L'opérateur | permet de sélectionner plus d'un nœud : `cheminXPath1|cheminXPath2|...`
 - Exemple `/personne/nom|/animal/nom`

xmllint

- permet de faire de la validation (dtd, xsd, relaxNG), options :
 - dtdvalid, schema, relaxng
 - noout, xmlout
- permet de rechercher des nœuds via des XPath, option :
 - xpath
- propose un mode interactif (option shell), commandes :
 - cat, cd, dir, ls , etc.

Quelques références

- XML in a Nutshell, O'Reilly (Elliote Rusty Harold & W. Scott Means)
- Introduction à XML, O'Reilly (Erik T. Ray)
- XML langage et applications, Eyrolles (Alain Michard)