Rappels sur XPath

Matthias Gille Levenson

AMU – CIHAM UMR 5648

prénom [point] gille [point] levenson [at] ens-lyon.fr

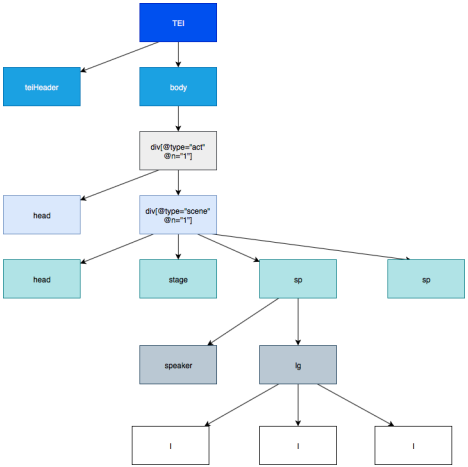
2 octobre 2023

Espaces de noms et préfixes

Le document XML comme un arbre

- Le document XML est un arbre qui contient un noeud racine et un certain nombre d'embranchements.
- On passe d'un niveau à l'autre à l'autre
  de la barre oblique / comme pour n'importe quel système arborescent comme le système de fichiers : `/Bureau/Travail/Cours_et_formations/Biblissima-_scripts_mss/XPath/diapos` représente un système fonctionnant de la même manière que `TEI/body/div/div/head`.

Application à Andromaque



Exercice 1

- Quels éléments sont sélectionnés par les chemins suivants ?
  - `TEI/body/div/div` ;
  - `TEI/body/div/div/head` ;
  - `TEI/body/div/div/sp`.

Les axes XPath

- On peut vouloir naviguer dans l'arbre sans connaître exactement sa structure. Pour ce faire, on aura besoin de naviguer selon les grandes relations entre noeud
- Un noeud a en effet 0 ou 1 parent, et il peut avoir 0 ou plus enfants. Il peut avoir des ancêtres et des descendants, ainsi que des « adelphes » (*siblings*).
- Les **axes** XPath permettent de représenter ces relations et aident à la navigation dans l'arbre
- On utilise un axe de la façon suivante (`axe::noeud`). Ici, le double deux-points est fondamental, il permet de distinguer l'axe du préfixe de l'espace de nommage.

Les axes XPath : navigation sur parent proche

- `child` : l'enfant
- `parent` : le parent
- `preceding-sibling` : l'adelphe précédent (premier enfant précédent du parent)
- `following-sibling` : l'adelphe suivant (premier enfant suivant du parent)

Les axes XPath : navigation sur parent éloigné

- `ancestor` : n'importe quel noeud ancêtre
- `descendant` : n'importe quel noeud descendant
- `preceding` : n'importe quel noeud précédant le noeud actuel
- `following` : n'importe quel noeud suivant le noeud actuel

Expressions conditionnelles ou prédicats

- On peut vouloir cibler des noeuds précis et avoir besoin d'exprimer des conditions pour ce faire. Le **prédicats** permettent d'exprimer ces conditions. Un prédicat se matérialise sous la forme d'une expression entre crochets droits `[condition]`
- Cette condition peut être de n'importe quelle forme : il suffit qu'elle soit vraie pour que le noeud soit retourné.
- On peut ainsi tester l'existence ou l'absence d'un attribut, la valeur d'un attribut, une égalité ou inégalité, comparer des valeurs...
- Ainsi, dans la transcription diplomatique d'un manuscrit, `//tei:text/descendant::tei:lb[@break='no']` permettra de sélectionner tous les débuts de ligne qui coupent un mot, soit toutes les coupures de mot à la ligne.

Exercice 2

- À partir de l'arbre XML simplifié d'Andromaque donner les chemins suivants en partant de l'élément racine TEI
  - Donner le chemin vers `body` ;
  - Donner le chemin vers la `div` dont l'attribut `@type` est égal à `act` ;
  - Donner le chemin vers `stage` ;
  - Donner le chemin vers `speaker` ;
  - Donner le chemin vers les `l`.

Fonctions basiques : `translate`

Un certain nombre de fonctions pré-établies sont particulièrement utiles ici. **Ce sont des fonctions XPath** : vous verrez avec Ariane d'autres fonctions propres à XSLT, et avec Jean-Paul des fonctions propres à XQuery.

- `translate(text, str, repl)` produit un *mapping* pour remplacer **un à un** les éléments de `str` par les éléments de `repl` dans le texte donné.
- Exemple : `translate('Longtemps, je me suis couché de bonne heure', 'aeiou', 'uoiea')` produira « Lengtomps, jo mo sais ceaché do benno hoaro. »

Fonctions basiques : `replace`

- `replace(text, str, repl)` remplace par `repl` toutes les chaînes `str` trouvées dans `text`.
- Exemple : `translate('Longtemps, je me suis couché de bonne heure', 'couché', 'endormi')` produira « Longtemps, je me suis endormi de bonne heure. »

Fonctions basiques : type des noeuds visés

- Noeuds textuels : `text()`
- Noeuds XML : `element()`
- Commentaires : `comment()`
- Noeud en général : `node()`

Fonctions basiques : modifier la casse (XPath 2.0)

- `lower-case(text)`
- `upper-case(text)`

Fonctions basiques : identifier des sous-chaînes de caractères

- `substring-before(text, string)`
- `substring-after(text, string)`

Fonctions basiques : manipuler des chaînes de caractères multiples

- `concat(text1, text2, text3, ..., textN)` : concatène l'ensemble des chaînes de caractères indiquées par l'utilisateur.ice. Les arguments ne peuvent être que des éléments textuels uniques et non pas des séquences
- `string-join(chemin)` : concatène l'ensemble du texte trouvé dans le chemin. (XPath 2.0)

Fonctions basiques : `compter`

- `count(path)`

Fonctions basiques : la position

- Tous les noeuds ont une position, **qui commence à 1**. Elle peut être décidée à l'aide d'un prédicat et de la fonction `position()` : `//library/bookshelfA/book[position() = 3]` ou plus simplement `//library/bookshelfA/book[3]`
- La position est déterminée par rapport à l'expression XPath qu'elle suit.
- On peut récupérer le dernier élément d'une liste avec la fonction `last()`

Exercice 3

Nous allons travailler avec le sonnet 17 de Shakeaspeare, présent dans le répertoire `XPath` du dépôt. On travaille sur Oxygen.

- Trouvez tous les vers dont le schéma métrique correspond au schéma canonique du sonnet.
- Trouvez tous les vers qui présentent une divergence par rapport au schéma métrique canonique du sonnet
- Comptez le nombre de césures dans le sonnet
- Identifiez le vers qui comprend la deuxième césure
- Identifiez le texte du vers qui vient après la première césure