

XML & XSLT

NOTIONS AVANCEES DE XSL XPATH

XSL : les paramètres et les variables

- On peut définir des paramètres avec l'instruction

```
<xsl:param  
  name="nomParam">...</xsl:param>
```

- On peut définir des variables locales à un template ou globale à tout le traitement avec l'instruction

```
<xsl:variable  
  name="nomVar">...</xsl:variable>
```

XSL : les paramètres et les variables

- On peut passer un paramètre en ligne de commande comme suit :
`java -jar saxon.jar xxx.xml yyy.xsl nomPar=valeurPar`
- Pour récupérer le paramètre dans le fichier xsl, il faut déclarer le paramètre en haut du fichier xsl comme suit :
`<xsl:param name="nomPar"/>`
- Le paramètre est global et donc utilisable dans tous les templates.
- On peut récupérer le contenu d'une variable ou d'un paramètre avec l'expression
`<xsl:value-of select="$nomElement"/>`
où `nomElement` est le libellé de la variable ou du paramètre.

XSL : les paramètres et les variables

- Dans les nœuds de définition de variables ou de paramètres, le moteur xslt ne dirige plus les résultats dans la sortie standard mais dans la variable ou dans le paramètre

```
<xsl:variable name="maVariable">  
  <xsl:for-each select="livre">  
    <xsl:value-of select="@titre"/>  
    <xsl:if test="not(position() = last())">  
      <xsl:text>-</xsl:text>  
    </xsl:if>  
  </xsl:for-each>  
</xsl:variable>
```

La variable maVariable aura pour contenu, la concaténation des titres des livres séparés par des tirets.

XSL : les opérateurs numériques

- On peut effectuer des opérations numériques avec le langage xsl en utilisant l'instruction `<xsl:value-of select="expression numérique"/>`
- Exemple : `<xsl:value-of select="2 * @prix"/>`

Appel direct d'un template

- Les instructions `xsl:apply-templates` et `xsl:for-each` provoquent l'appel de template que si les noeuds invoqués dans la clause `select` existent.
- Il existe une autre catégorie de template que le programmeur peut appeler directement.
- Ces templates sont définis comme les précédents sauf sur deux points :
 - on utilise le mot-clé `name` à la place de `match`;
 - ils peuvent recevoir des paramètres comme lors d'un appel de fonction dans un langage procédural.

Appel direct d'un template

- La syntaxe d'écriture d'un tel template est la suivante :

```
<xsl:template name="nomTemplate">  
  <xsl:param name="nomParam1"/>  
  ...  
  <xsl:param name="nomParamn"/>  
  ...  
</xsl:template>
```

- La syntaxe d'appel d'un tel template est la suivante :

```
<xsl:call-template name="nomTemplate">  
  <xsl:with-param name="nomParam1" select="..." />  
  ...  
  <xsl:with-param name="nomParamn" select="..." />  
</xsl:call-template>
```

CDATA

- Dans le langage XLS, certains symboles ont un sens particulier comme &, <, etc. On parle alors de métacaractères.
- Si on désire les déspecialiser pour les écrire dans la sortie standard, dans une variable ou dans un paramètre, il faut utiliser l'instruction particulière qui suit :
`<![CDATA[...]]>`.
La zone ... qui est entre crochets peut contenir un texte quelconque y compris les caractères &, <, ... qui sont déspecialisés.
Exemple : `<![CDATA[if(x < 4)]]>`

XPATH

- XPath 1.0 *W3C Recommendation novembre 1999*
- Syntaxe "non XML" compacte pour :
 - exprimer un chemin dans un document XML
 - désigner des nœuds ...
 - effectuer une recherche ...
 - manipuler des chaînes, nombres, booléens
 - utiliser des fonctions, variables

Chemin XPATH

- Une expression de **chemin XPath** est une séquence d'*étapes*
 - chemin **absolu**: commence par / : /document/chapter/paragraph
 - chemin **relatif** : ne commence pas par / : chapter/pararagraph
- Une *étape XPath* contient :
 - un **axe** qui spécifie la relation structurale fils, descendants, ancêtres, frères, attributs,...
 - un **test** qui spécifie le type de nœud (*obligatoire*)
 - un **prédicat** pour affiner la sélection => filtrage

Exemple de document XML

```
<document>
  <chapter num='ch1'>
    ...
  </chapter>
  <chapter num='ch2'>
    <section>
      <paragraph>
        <figure>...</figure>
      </ paragraph >
      < paragraph type='warning'>
        ...
      </ paragraph >
    </section>
    <section>
      ...
    </section>
  </chapter>
  <...>
</document>
```

Rappel : Structure d'un document XML

XPath considère 7 types de nœuds

1. racine
2. éléments
3. nœuds texte
4. attributs
5. espace de noms
6. instructions de traitement
7. commentaires

Les axes

XPath propose les axes suivants :

- ancestor Ancestors of the context node
- ancestor-or-self Ancestors, including the context node
- attribute Attributes of the context node (abbreviated "@")
- child Children of the context node (the default axis)
- descendant Descendants of the context node
- descendant-or-self Descendants, including the context node (abbreviated "//")
- following Elements which occur after the context node, in document order
- following-sibling Elements which occur after the context node, in document order and have the same parent as the context node
- preceding Elements which occur before the context node, in document order (returned in reverse-document order)
- preceding-sibling Elements which occur before the context node, in document order (returned in reverse-document order) and have the same parent as the context node
- namespace The namespace nodes of the context node
- parent The parent of the context node (abbreviated "..")
- self The context node (abbreviated ".")

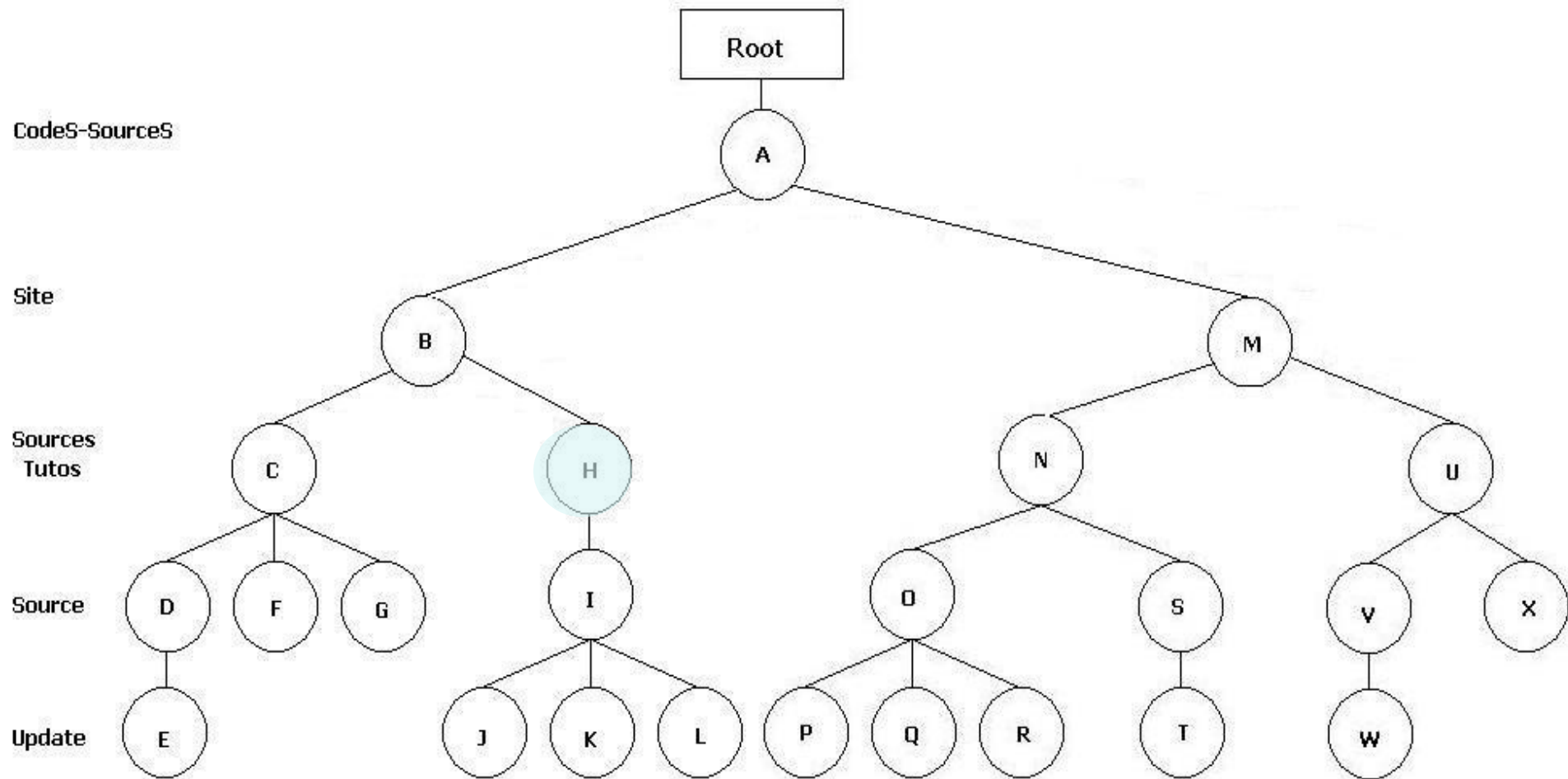
Les axes *suivant* et *précédent*

- **XXX/following-sibling::*** est un arbre qui contient tous les nœuds frères qui suivent le nœud contextuel
- **XXX/preceding-sibling::*** est un arbre qui contient tous les nœuds frères qui précèdent le nœud contextuel

Exemple - Document XML

```
<...> -----> preceding
<...> -----> preceding
  <chapter> -----> ancestor
    <section> -----> ancestor
      <paragraph> -----> self
        <figure>...</figure> -----> descendant
      </paragraph>
      <paragraph> -----> following-sibling
        ...
      </paragraph>
    </section>
    <section> -----> following
      ...
    </section>
  </chapter> -----> following
<...> -----> following
```

Exemple - Arborescence XML



Value	Node
self	H
parent	B
child	I
descendant	I, J, K, L
descendant-or-self	H, I, J, K, L
ancestor	B, A
ancestor-or-self	H, B, A
preceding	E, D, C, F, G
preceding-sibling	C
following	M, N, O, P, Q, R, S, T, U, V, W, X
following-sibling	vide

XPath - Test

- **name** : élément **name**
- ***** : n'importe quel élément
- **namespace:name** : élément **name** dans l'espace des noms **namespace**
- **namespace:*** : n'importe quel élément dans l'espace des noms **namespace**
- **comment()** : noeud commentaire
- **text()** : noeud texte
- **processing-instruction()** : instruction de traitement
- **processing-instruction('target')** : instruction de traitement pour l'application **target** indiquée
- **node()** : n'importe quel type noeud où qu'il soit

Exemples - Syntaxe complète

- **child::paragraph** l'élément **para** fils du nœud courant
- **child::*** tous les éléments fils du nœud courant
- **child::text()** tous les nœuds de texte fils du nœud courant
- **child::node()** tous les nœuds de texte fils du nœud courant, quelque soit le type du nœud
- **attribute::name** l'attribut **name** du nœud courant
- **attribute::*** tous les attributs du nœud courant
- **descendant::paragraph** l'élément **para** qui est un descendant du nœud courant
- **self::para** le nœud courant si c'est un élément **para**, sinon ne sélectionne rien
- **child::chapter/descendant::paragraph** l'élément **paragraph** qui est un descendant d'un élément **chapter** fils du nœud courant
- **child::paragraph[position()=1]** le premier élément **paragraph** fils du nœud courant
- **child::paragraph[position()=last()]** le dernier élément **paragraph** fils du nœud courant
- **child::chapter[child::title]** les éléments **chapter** fils du nœud courant qui ont un ou plusieurs éléments **title** comme fils
- **ancestor::table** les éléments *table* qui sont parmi les ancêtres
- **following-sibling::paramdef** les éléments *paramdef* qui sont parmi les frères suivants
- **ancestor-or-self::*/@sepchar** l'attribut *sepchar* de l'élément courant ou n'importe quel ancêtre de l'élément

Les prédicats

Fonctions utilisables :

- **sur les nœuds**

- retourne un nombre :
 - **last()** vraie ssi le noeud est le dernier du contexte courant.
 - **position()** renvoie la position dans le contexte du noeud courant.
 - **id(*nom*)** renvoie le noeud identifié par l'étiquette *nom*.
 - **count(*node-set*)** retourne le nombre de nœuds
- retourne une chaîne :
 - **local-name(*noeuds*)** renvoie la partie locale de l'étiquette d'un noeud.
 - **namespace-uri(*noeuds*)** renvoie la partie espaces de nom de l'étiquette d'un noeud.
 - **name(*noeuds*)** renvoie l'étiquette d'un noeud.

- **sur les chaînes de caractères (texte)**

- retourne une chaîne : `string(object?)`, `concat(string, string, string*)`,
`substring-before(string, string)`, `substring-after(string, string)`,
`translate (string, string, string)`
- retourne un booléen : `contains(string, string)`, `starts-with(string, string)`
- retourne un nombre : `string-length(string)`

Les prédicats

- **booléen** retourne un booléen : `boolean(object)`, `true()`, `false()`, `lang(string)`
 - **true()** toujours vraie.
 - **false()** toujours fausse.
 - **boolean(*objet*)** vraie ssi l'objet est égale à la constante « true ».
 - **not(*booléen*)** vraie ssi le paramètre est faux
- **numérique** retourne un nombre :
 - **number(*objet*)** traduire l'objet sous la forme d'un nombre. renvoie NaN si l'objet ne représente pas un nombre.
 - **sum(*noeuds*)** renvoie la somme des noeuds après les avoir transformés en nombre.
 - **count(*noeuds*)** renvoie le nombre de noeuds.
 - **floor(*nombre*)** arrondi par le bas.
 - **ceiling(*nombre*)** arrondi par le haut.
 - **round(*nombre*)** arrondi par le plus proche.

Syntaxe : entre crochet *après le test du noeud*

Exemple : **paragraph[last()]**

Exemple - Prédicats

- **nodetest[1]** le premier nœud
- **nodetest[position()=last()]** le dernier nœud
- **nodetest[position() mod 2 = 0]** les nœuds pairs
- **element[@id="foo"]** élément(s) dont l'attribut id a la valeur "foo "
- **element[not(@id)]** les éléments qui n'ont pas d'attribut id
- **author[firstname="Norman"]** les éléments author qui ont des fils firstname dont le contenu est "Norman«
- **author[normalize-space(firstname)="Norman"]** les éléments author qui ont des fils firstname dont le contenu est "Norman" (sans tenir compte des espaces)