Langage XPATH

1 - Chemins de localisation

Un chemin de localisation est une expression XPath utilisée pour sélectionner une collection de nœuds par rapport au nœud de contexte. L'évaluation d'une expression de chemin de localisation résulte en un ensemble de nœuds contenant les nœuds spécifiés par le chemin de localisation. De manière récursive, un chemin de localisation peut contenir des expressions utilisées pour filtrer des collections de nœuds.

Du point de vue de la syntaxe, un chemin de localisation se compose d'une ou de plusieurs étapes de localisation, séparées par des barres obliques (`/`) :

locationstep/locationstep/locationstep

Chaque étape de localisation sélectionne à son tour une collection de nœuds par rapport au nœud de contexte, c'est-à-dire le nœud sélectionné par l'étape de localisation précédente. Un chemin de localisation exprimé de cette manière est un chemin de localisation relatif. Un chemin de localisation absolu part de l'élément racine :

/locationstep/locationstep/locationstep

Dans un chemin de localisation, les étapes de localisation sont évaluées de gauche à droite. L'étape la plus à gauche sélectionne une collection de nœuds par rapport au nœud de contexte. Ces nœuds deviennent alors le contexte utilisé pour traiter l'étape de localisation suivante. Ce processus de traitement des étapes et de mise à jour du nœud de contexte se répète jusqu'à ce que toutes les étapes de localisation aient été traitées.

Un chemin de localisation peut être abrégé ou non.

Dans un chemin de localisation non abrégé, les étapes de localisation adoptent la syntaxe suivante :

axis::node-test[predicate]
Dans cette syntaxe, **`axis`** spécifie la relation entre les nœuds sélectionnés par l'étape de localisation et le nœud de contexte ; **`node-test`** spécifie le type de nœud et le nom développé des nœuds sélectionnés par l'étape de localisation ; **`predicate`** est une expression de filtre qui affine la sélection des nœuds dans l'étape de localisation. Les prédicats sont facultatifs. Dans ce cas, une étape de localisation se compose uniquement de **`axis::`** et **`node-test`**. Le tableau suivant donne quelques exemples.

Chemin de localisation non abrégé	Description
child::para[last()]	Sélectionne le dernier élément <para> du nœud de contexte.

parent::para	Sélectionne l'élément <para> qui est parent du nœud de contexte.
child::text()	Sélectionne tous les nœuds de texte enfants du nœud de contexte.
child::div/child::para	Sélectionne les éléments <para> enfants de l'élément qui est un enfant du nœud de contexte.

Dans un chemin de localisation abrégé, l'axe, **`axis::`**, n'est pas exprimé explicitement dans les étapes de localisation, mais implicitement par une série de raccourcis. Le tableau suivant donne quelques exemples.

Chemin de localisation abrégé	**Description**
para	Sélectionne les éléments <para> du nœud de contexte.
../para	Sélectionne l'élément <para> qui est parent du nœud de contexte.
text()	Sélectionne tous les nœuds de texte enfants du nœud de contexte.
./div/para	Sélectionne les éléments <para> enfants des éléments enfants du nœud de contexte.

Voici une synthèse de certaines abréviations :

Non abrégé	**Abrégé**
child::*	`*`
attribute::*	`@*`
/descendant-or-self::node()	`//`
self::node()	`.`
parent::node()	`..`

2 - Etapes de localisation

Une étape de localisation sélectionne une collection de nœuds (node-set) par rapport au nœud de contexte.

Une étape de localisation se compose de trois parties : un axe facultatif, un test de nœud et un prédicat facultatif. La syntaxe d'une étape de localisation est le nom de l'axe suivi de deux signes deux-points, du test de nœud et de zéro, un ou plusieurs prédicats, chacun entre crochets. La forme la plus basique de cette syntaxe est la suivante :

axis::nodetest[predicate]

Axis
Spécifie la relation d'arborescence entre le nœud de contexte et les nœuds que l'étape de localisation doit sélectionner. En d'autres termes, l'axe indique la direction générale de l'étape de localisation à partir du nœud de contexte. Dans une étape de localisation, l'axe est facultatif. S'il est omis, l'axe par défaut est **child::.** De plus, plusieurs axes possèdent des raccourcis ; par exemple, l'éperluette (@) est un raccourci de l'axe d'attribut.

nodetest
Spécifie le type de nœud ou le nom développé des nœuds que l'étape de localisation doit initialement sélectionner. Autrement dit, le test de nœud indique, parmi tous les nœuds de l'axe spécifié, celui ou ceux à considérer comme des candidats, c'est-à-dire des correspondances potentielles pour l'étape de localisation.

prédicat
Utilise une expression XPath (condition requise) pour affiner davantage la collection de nœuds sélectionnée par l'étape de localisation. Le prédicat est un filtre, spécifiant un critère de sélection pour affiner davantage la liste des nœuds candidats. Le prédicat est facultatif. S'il n'y a pas de prédicat, l'étape de localisation ne comprend pas de crochets ([et]).

Sélection de nœuds

La collection de nœuds sélectionnée par une étape de localisation découle de la génération d'une collection de nœuds initiale basée sur la relation entre l'axe et le test de nœud, et du filtrage de cette collection de nœuds initiale à l'aide de chaque prédicat.

La collection de nœuds initiale est composée des nœuds qui satisfont aux deux critères suivants :

- la relation des nœuds avec le nœud de contexte correspond à celle spécifiée par l'axe ;
- le type et le nom développé des nœuds correspondent à ceux spécifiés par le test de nœud.

XPath utilise ensuite le premier prédicat de l'étape de localisation pour filtrer la collection de nœuds initiale afin de générer une nouvelle collection de nœuds. Puis, il utilise le deuxième prédicat pour filtrer la collection de nœuds résultant du premier prédicat. Ce processus de filtrage se répète jusqu'à ce que XPath ait évalué tous les prédicats. La collection de nœuds résultant de l'application de tous les prédicats est celle choisie par l'étape de localisation.

Remarque

Étant donné que l'axe affecte l'évaluation de l'expression dans chaque prédicat, la sémantique d'un prédicat est définie en fonction de l'axe spécifié.

Le tableau suivant montre des exemples d'étape de localisation utilisant la syntaxe complète.

Étape de localisation	Description
child::*[position()=1]	Localise le premier nœud enfant du nœud de contexte
ancestor-or-self::book[@catdate="2000-12-31"]	Localise tous les ancêtres d'un enfant <book> du nœud de contexte, ainsi que l'enfant <book> lui-même, pour autant que l'élément en question possède un attribut catdate de valeur "2000-12-31".
//parent::node()[name()="book"] | descendant::node()[name()="author"]	Localise n'importe quel nœud du document dont le nœud parent s'appelle « book » ou n'importe quel nœud descendant du nœud de contexte dont le nom est « author ».

3 – Les axes

Un chemin de localisation utilise un axe pour spécifier la relation entre les nœuds sélectionnés par l'étape de localisation et le nœud de contexte.

Axes	Description
ancestor::	Ancêtres du nœud de contexte. Les ancêtres du nœud de contexte sont constitués du parent du nœud de contexte et du parent du parent, etc. Donc, l'axe **ancestor::** inclut toujours le nœud racine sauf si le nœud de contexte est le nœud racine.
ancestor-or-self::	Nœud de contexte et ses ancêtres. L'axe **ancestor-or-self::** inclut toujours le nœud racine.
attribute::	Attributs du nœud de contexte. Cet axe est vide sauf si le nœud de contexte est un élément.
child::	Enfants du nœud de contexte. Un enfant est un nœud situé immédiatement sous le nœud de contexte dans l'arborescence. Toutefois, ni les nœuds d'attribut ni ceux d'espace de noms ne sont considérés comme des enfants du nœud de contexte.

descendant::	Descendants du nœud de contexte. Un descendant est un enfant ou un enfant d'un enfant, etc. Donc, l'axe **descendant::** n'inclut jamais de nœud d'attribut ou d'espace de noms.
descendant-or-self::	Nœud de contexte et ses descendants.
following::	Tous les nœuds qui suivent le nœud de contexte dans l'arborescence, sauf les nœuds descendants, d'attribut et d'espace de noms.
following-sibling::	Tous les frères suivants du nœud de contexte. L'axe **following-sibling::** identifie uniquement les enfants d'un nœud parent qui apparaissent dans l'arborescence après le nœud de contexte. Cet axe ne comprend pas les autres enfants qui apparaissent avant le nœud de contexte. Si le nœud de contexte est un nœud d'attribut ou d'espace de noms, l'axe **following-sibling::** est vide.
namespace::	Nœuds d'espace de noms du nœud de contexte. Il existe un nœud d'espace de noms pour chaque espace de noms dans la portée du nœud de contexte. Cet axe est vide sauf si le nœud de contexte est un élément.
parent::	Le parent du nœud de contexte, le cas échéant. Le parent est le nœud situé juste avant le nœud de contexte dans l'arborescence.
preceding::	Tous les nœuds qui précèdent le nœud de contexte dans l'arborescence, sauf les nœuds ancêtres, d'attribut et d'espace de noms. L'une des manières de voir l'axe précédent est de considérer tous les nœuds dont le contenu se produit dans sa totalité avant le début du nœud de contexte.
preceding-sibling::	Tous les frères précédents du nœud de contexte. L'axe **preceding-sibling::** identifie uniquement les enfants d'un nœud parent qui apparaissent dans l'arborescence avant le nœud de contexte. Cet axe ne comprend pas les autres enfants qui apparaissent après le nœud de contexte. Si le nœud de contexte est un nœud d'attribut ou d'espace de noms, l'axe **preceding-sibling::** est vide.
self::	Nœud de contexte uniquement.

Exemples

following::

Les exemples de l'axe **following::** font référence au document d'instance suivant :

```
<A>
  <B>
    <C>sample</C>
    <C>sample2</C>
  </B>
  <B>
    <C>sample</C>
    <C>sample2</C>
    <D>sample3</D>
  </B>
</A>
```

Requête	Nœuds retournés

`A/B[1]/following::*`	`<B>`
	`  <C>sample</C>`
	`  <C>sample2</C>`
	`  <D>sample3</D>`
	`</B>`
	`<C>sample</C>`
	`<C>sample2</C>`
	`<D>sample3</D>`

`A/B[1]/following::node()`	`<B>`
	`  <C>sample</C>`
	`  <C>sample2</C>`
	`  <D>sample3</D>`
	`</B>`
	`<C>sample</C>`
	`sample`
	`<C>sample2</C>`
	`sample2`
	`<D>sample3</D>`
	`sample3`

preceding::

Les exemples de l'axe **preceding::** font référence au document d'instance suivant :

```
<A>
  <B>
    <C test="sampletest">sample</C>
    <C>sample2</C>
  </B>
  <B>
    <C>sample</C>
```

```
    <C>sample2</C>
    <D>sample3</D>
  </B>
</A>
```

Requête	Nœuds retournés

`A/B[2]/preceding::*`	`<B>`
	`  <C test="sampletest">sample</C>`
	`  <C>sample2</C>`
	`</B>`
	`<C>sample</C>`
	`<C>sample2</C>`

following-sibling::

L'exemple de l'axe **following-sibling::** fait référence au document d'instance suivant :

```
<A>
  <B>
    <C test="sampletest">sample</C>
    <C>sample2</C>
  </B>
  <B>
    <C>sample</C>
    <C>sample2</C>
    <D>sample3</D>
  </B>
</A>
```

Requête	Nœuds retournés

`A/B[1]/following-sibling::*`	`<B>`
	`  <C>sample</C>`
	`  <C>sample2</C>`
	`  <D>sample3</D>`
	`</B>`

preceding-sibling::

L'exemple de l'axe **preceding-sibling::** fait référence au document d'instance suivant :

```
<A>
  <B>
    <C test="sampletest">sample</C>
    <C>sample2</C>
  </B>
  <B>
    <C>sample</C>
    <C>sample2</C>
    <D>sample3</D>
  </B>
```

</A>	
Requête	**Nœuds retournés**

`A/B[2]/preceding-sibling::*`	`<B>`
	`  <C test="sampletest">sample</C>`
	`  <C>sample2</C>`
	`</B>`

4 – Les prédicats

Un prédicat est une expression XPath qui filtre une collection de nœuds par rapport à un axe et produit une nouvelle collection de nœuds. Ce processus de filtrage implique l'évaluation séquentielle du prédicat par rapport à chaque nœud de la collection. Chaque fois que le prédicat est évalué par rapport à un nœud :

- le nœud de contexte est le nœud en cours d'évaluation ;
- la taille du contexte est le nombre de nœuds de la collection de nœuds en cours d'évaluation ;
- la position du contexte est la position du nœud de contexte dans la collection de nœuds.

Ce dernier contexte, celui du nœud de contexte dans la collection de nœuds, se rapporte à la direction dans laquelle l'axe spécifié dans l'étape de localisation parcourt l'arborescence de documents. Généralement, un axe parcourt l'arborescence vers l'avant ou vers l'arrière :

- Un axe vers l'avant contient le ou les nœuds de contexte situés après le nœud de contexte. Les axes **child::**, **descendant::**, **descendant-or-self::**, **following::** et **following-sibling::** sont des axes vers l'avant. Ces axes vers l'avant numérotent les nœuds de la collection de nœuds dans l'ordre du document, la première position étant la position 1.
- Un axe vers l'arrière contient le ou les nœuds de contexte situés avant le nœud de contexte. Les axes **ancestor::**, **ancestor-or-self::**, **preceding::** et **preceding-sibling::** sont des axes vers l'arrière. Ces axes vers l'arrière numérotent les nœuds de la collection de nœuds dans l'ordre inverse du document, la première position étant la position 1.

Quant aux autres axes, les axes **self::** et **parent::** retournent un seul nœud. La qualification d'axe vers l'avant ou l'arrière n'a donc pas de sens pour ces deux axes. Aucun ordre n'est défini pour les axes **attribute:: et namespaces::** ; eux non plus ne sont donc ni des axes vers l'avant ni des axes vers l'arrière.

Sélection des nœuds

Une expression de prédicat est évaluée en une valeur numérique ou booléenne.

Si le prédicat donne un nombre, XPath compare ce nombre à la position du nœud de contexte. Si le nombre et la position de contexte sont identiques (ce qui signifie que le nœud de contexte est à la

position appropriée dans l'arborescence), le prédicat donne la valeur **true** et le nœud de contexte est inclus dans la nouvelle collection de nœuds. Sinon, le nœud de contexte est exclu de la nouvelle collection de nœuds.

Si le prédicat ne donne pas un nombre, XPath utilise la fonction **boolean** pour convertir le résultat en une valeur booléenne. Par exemple, le prédicat **[genre='Computer']** produit une collection de nœuds. Si le nœud de contexte a un élément enfant **<genre>** dont le contenu est **Computer**, ce prédicat donne la valeur **true** et le nœud de contexte est inclus dans la nouvelle collection de nœuds. Sinon, le nœud de contexte est exclu de la nouvelle collection de nœuds.

Remarque

Un prédicat numérique [x] est équivalent au prédicat booléen **[position()=x]**.

5 Fonctions XPath

Introduction

Nous avons vu dans un chapitre précédent que les transformations XSLT font appel à une syntaxe particulière, nommée XPath, pour identifier les nœuds que l'on souhaite manipuler. Ce chapitre va donner quelques fonctions utiles.

Fonctions XPath applicables aux nœuds

Fonction de comptage : la fonction count()

La fonction **`count(ensemble_de_nœuds)`** permet de compter le nombre de nœuds référencés. Par exemple, si dans un fichier **XML** on déclare la liste des 9 planètes du système solaire sous la forme de balises **`planete`**, l'instruction **`count(//planete)`** renverra la valeur 9.

Obtenir la position d'un nœud: les fonctions position() et last()

Ces fonctions permettent de connaître la position d'un nœud par rapport à ses frères.

La fonction `position()` retourne la position du nœud contextuel. Sur l'exemple de système solaire, si on a classé les planètes par ordre de distance croissante au Soleil dans le fichier XML, le code **`<xsl:value-of select="planete[position()=3]/nom"/>`** renvoie la valeur "Terre" (la troisième planète à partir du Soleil).

La fonction **`last()`** permet de retourner le dernier nœud d'un ensemble de nœuds, c'est-à-dire la position du dernier nœud. Ainsi, le code **`<xsl:value-of select="planete[position()=last()]/nom"/>`** retourne la valeur "Pluton" (planète en moyenne la plus éloignée du Soleil).

Fonctions XPath applicables aux chaînes de caractères

La fonction **concat()** concatène toutes les chaînes qui lui sont passées en arguments et retourne la chaîne résultant de cette concaténation. Sa syntaxe est **concat(chaine1, chaine2, ...)**. Par exemple, **<xsl:value-of select concat('Le livre dont le titre est', livre/titre, ' a été écrit par ', livre/auteur)/>** permet ainsi d'afficher la chaîne "Le livre dont le titre est Les Misérables a été écrit par Victor Hugo".

Test de présence d'une sous-chaîne: les fonctions contains() et starts-with()

Ces fonctions permettent de déterminer si une chaîne de caractères est incluse dans une autre, et renvoient un booléen. La fonction **starts-with(chaine1, chaine2)** renvoie la valeur **true** si **chaine1** commence par la **chaine2**, **false** sinon. La fonction **contains(chaine1, chaine2)** renvoie **true** si **chaine1** contient **chaine2**, **false** sinon.

Il existe trois fonctions permettant d'extraire une sous-chaîne d'une chaîne donnée : **substring()**, **substring-after()** et **substring-before**.

- La fonction **substring(chaine1, decalage, longueur)** retourne une sous-chaîne de **chaine1** contenant **longueur** caractères et commençant à **decalage**. Par exemple, **substring("012345", 2, 3)** renvoie la chaîne "123" ;
- La fonction **substring-after(chaine1, chaine2)** retourne la sous-chaîne de **chaine1** qui suit la première occurence de **chaine2**. Par exemple, **substring-after("012345", "2")** renvoie la chaîne "345" ;
- La fonction **substring-before(chaine1, chaine2)** retourne la sous-chaîne de **chaine1** qui précède la première occurence de **chaine2**. Par exemple, **substring-before("012345", "2")** renvoie la chaîne "01".

Suppression des espaces en surnombre: la fonction normalize-space()

Il est parfois nécessaire de supprimer des espaces en surnombre dans une chaîne. La fonction **normalize-space(chaine)** retourne la chaîne de caractères qu'elle reçoit en argument après en avoir supprimé les espaces situés au début et à la fin, et y avoir remplacé chaque séquence d'espaces successifs par un espace unique.

Longueur d'une chaîne: la fonction string-length()

Cette fonction retourne "classiquement" la longueur de la chaîne qu'elle reçoit en argument.

Fonctions XPath applicables aux nombres

Les opérateurs suivants sont disponibles en XSLT :

+ pour l'addition ;

- pour la soustraction ;

***** pour la multiplication ;

`div` pour la division ;

`nombre1 mod nombre2` retourne le reste de la division euclidienne du premier nombre par le second.

Fonctions de manipulation

- La fonction `ceiling(nombre)` retourne le plus petit entier égal ou supérieur au nombre qu'elle reçoit en argument ;
- La fonction `floor(nombre)` retourne le plus grand entier égal ou inférieur au nombre qu'elle reçoit en argument ;
- La fonction `round(nombre)` arrondit le nombre qu'elle reçoit à l'entier le plus proche.
- La fonction `sum(nombre1, nombre2, ...)` retourne la somme des nombres reçus en argument. Cette fonction sert aussi à calculer la somme des éléments spécifiés par une expression XPAth : par exemple, `sum(//prix)` calcule la somme des contenus de tous les éléments `prix` du document XML.

Fonctions booléennes

Il est parfois nécessaire de créer une constante booléenne initialisée à une valeur **`true`** ou **`false`**. Cette opération est réalisable par un appel respectivement aux fonctions **`true()`** et **`false()`**.

La fonction **`not(variable)`** inverse le sens logique de son argument.

La fonction `lang(chaine)` vérifie que la langue dans laquelle est écrit le nœud courant (tel qu'elle est définie par l'attribut `xml:lang`) est le même que le langage qu'elle reçoit en argument. Cette fonction reçoit une chaîne correspondant à l'un des codes de langage définis dans la spécification XML : `en` pour l'anglais, `jp` pour le japonais, `fr` pour le français, *etc*. Cela permet, avec une même feuille de style, de gérer des fichiers XML écrits dans des langues différentes.

Exemple

Supposons que nous soyons en train de transformer un fichier **`biblio.xml`**, contenant une liste de livres, pour lesquels on donne un auteur :

```
<biblio>
  <livre>(...)</livre>
  <livre>(...)</livre>
  <livre>
    <titre>Les Misérables</titre>
    <auteur prénom="Victor" nom="Hugo" />
  </livre>
</biblio>
```

En parallèle, on dispose d'un fichier **`auteurs.xml`** possédant des références biographiques sur des auteurs, dont Victor Hugo :

```
<biographies>
  <auteur>(...)</auteur>
  <auteur>(...)</auteur>
  <auteur prénom="Victor" nomFamille="Hugo">
    <naissance><date>1802</date>(...)</naissance>
    <décès><date>1885</date>(...)</décès>
    <biographie>(...)</biographie>
  </auteur>
</biographies>
```

Lors de la transformation du fichier **`biblio.xml`**, on peut accéder à ces informations avec la commande suivante :

```
<xsl:variable name="nomAuteur" select="@nom" />
<xsl:variable name="prénomAuteur" select="@prénom" />
<xsl:value-of select="document('auteurs.xml')//auteur[@prénom=$prénomAuteur
and @nomFamille=$nomAuteur]/naissance/date" />
<xsl:value-of select="document('auteurs.xml')//auteur[@prénom=$prénomAuteur
and @nomFamille=$nomAuteur]/décès/date" />
```