Quelques définitions

Internet permet d'interconnecter des machines entre elles, par l'intermédiaire d'un protocole IP.

Le Web est une application au-dessus d'Internet permettant d'afficher des pages HTML sur les machines interconnectées par Internet, grâce au protocole HTTP.

Le Mail est également une application au dessus d'Internet, permettant l'envoi de documents, vers des utilisateurs identifiés.

La publication sur Internet consiste à donner une adresse unique à une information pour pouvoir la récupérer ensuite.

Un URI est un identifiant unique dans le monde définissant une information publiée sur un réseau quelconque : Internet, intranet d'entreprise, etc.

Un URL est un identifiant unique dans le monde définissant une information publiée sur Internet.

Un moteur de recherche récupère une page HTML et n'en garde que le texte brut, dont il soustraie ensuite les mots vides de sens : cela donne une liste de mots dont le moteur de recherche compte le nombre d'apparition sur la page en question. Il en découle une succession de mots clés « représentatifs » de la page HTML.

Range Rank est un algorithme utilisé par Google pour classer les pages HTML, qui repose sur le principe qu'une page est importante si beaucoup d'autres pages importantes pointent vers elle. Grâce à ce système, Google livre à l'utilisateur LA page de référence relatif à sa requête, et non l'exacte réponse à sa question.

Le Web sémantique tente de concevoir une approche différente qui permettrait de trouver l'exacte réponse à la requête de l'utilisateur. C'est un ensemble de technologies visant à rendre le contenu des ressources du Web accessible et utilisable par les programmes et agents logiciels, grâce à un système de métadonnées formelles.

La recherche de données par requêtes en langage naturel et systèmes de questions / réponses est l'un des enjeux du web sémantique. Avec ces systèmes, l'utilisateur ne saisit plus dans un moteur de recherche une requête composée de terme décrivant l'objet de sa recherche mais une question claire.

Un flux RSS est un fichier dont le contenu est produit automatiquement, en fonction des mises à jour d'un site Web. Les flux RSS sont souvent utilisés par les sites d'actualité ou les blogs pour présenter les titres des dernières informations consultables en ligne. Le terme RSS signifie que le contenu du fichier est

informatiquement codé selon le standard RSS, qui s'appuie lui-même sur le langage informatique XML.

La lecture d'un flux RSS se fait à l'aide d'un agrégateur. Il faut renseigner l'adresse du fil RSS concerné afin que le programme se connecte régulièrement aux sites émetteurs pour vérifier la présence de nouveaux contenus. Si c'est le cas, il est téléchargé et converti au format HTML pour en permettre la lecture.

Introduction à XML

Définition

XML est un langage de description ayant pour but de représenter les données du Web, mais en les structurant mieux qu'HTML ou XHTML, afin de mieux les exploiter.

Il permet de décrire la structure logique de document principalement textuel, à l'aide d'un système de balises permettant de marquer les éléments qui composent la structure, et les relations entre ces éléments. Le balisage structurel consiste à reconnaître que tout document textuel est construit selon une structure reconnue par les lecteurs grâce à des marques typographiques, des conventions de mise en page et des connaissances culturelles relatives aux informations qu'un certain type de document se doit de contenir.

Derrière un document XML, on ne trouve pas du texte mais une structuration des données hiérarchisée, en forme d'arbre.

Avancées (par rapport aux autres langages de programmation Web)

XML permet de créer ses propres balises, à l'inverse de HTML qui oblige le concepteur à utiliser des balises fixées (head, tittle, etc.).

XML ne permet pas de description physique, c'est-à-dire de mise en page, d'un document, qui sera piloter par une feuille de style CSS, par exemple. Cela évite des soucis de durée de vie des éléments de mise en page par rapport aux évolutions technologiques, comme c'est le cas avec HTML qui mélange, dans un même document, sa structure logique et sa description physique.

Pour pointer un passage précis d'une page HTML et ainsi le reprendre dans une autre page HTML, il faut que le concepteur modifie son document en ajoutant une ancre, limite que contourne XML.

XML permet d'utiliser des données structurées, mais aussi semi-structurées, hétérogènes. En effet, il existe différentes façons de décrire des données car ces dernières peuvent avoir des formes diverses.

XML est un langage auto-descriptif : données et modèle sont rassemblés dans un seul et même document, évitant ainsi l'obligation pour le concepteur de créer un mode d'emploi dans un document indépendant.

Langage XML

Structure d'un document

- Prologue

La déclaration XML précise la version de XML et le codage de caractère utilisés, ici 1.0 et la valeur par défaut :

< ?xml version = "1.0" encoding = " ">

La déclaration de type de document indique, dans le cas où le document se conforme à une structure type particulière appelée DTD, quel est ce type.

Ici, cela signifie que le blablabla entre crochets sera valide pour tout document dont l'élément racine sera du type nom_élément_racine : cette déclaration pourra être recopiée inchangée dans tous les documents de type nom_élément_racine.

< ! DOCTYPE nom_élément_racine [blablabla] >

- Arbre d'éléments

Chaque élément d'un document XML se compose d'une balise d'ouverture, d'un contenu d'élément et d'une balise de clôture :

< nom_élément > xxxxxxxx < / nom_élément >

Le nom de l'élément, figurant dans les balises ouvrante et fermante, est libre (avec quelques contraintes de caractère toutefois).

Un élément peut contenir :
- d'autres éléments, fils du premier
- du texte brut, enchaînement de caractère
- un mélange des 2

Un élément peut également être vide. Sa notation sera alors simplifiée comme suit :

< nom_élément / >

Dire qu'un document doit contenir un arbre d'élément signifie que :
- Tout élément fils est complètement inclus dans son père : il ne peut pas y avoir recouvrement d'éléments.

- Il existe dans un document un et un seul élément père qui contient tous les autres : l'élément racine.

Des commentaires peuvent être inclus dans un document. Ils sont notés comme suit :

< ! - - commentaire - - >

Eléments et attributs

Une balise d'ouverture d'un élément doit comporter le nom de l'élément, éventuellement suivi par d'un ou plusieurs attributs utilisés pour décrire certaines propriétés de l'élément. Chaque attribut est représenté par une paire nom="valeur", où la valeur est une chaîne de caractères.

< nom_élément nom_attibut_1 = "valeur" nom_attibut_2 = "valeur" >

Afin de choisir si une information doit apparaître comme valeur d'un attribut ou comme contenu d'un élément, on s'appuiera sur les considérations suivantes :

- Un attribut ne peut apparaître qu'une unique fois au sein d'un élément, alors qu'un élément fils peut apparaître plusieurs fois au sein de son élément père.
- Les attributs associés à un élément ne sont pas ordonnés, à la différence du contenu d'un élément organisé selon une arborescence d'éléments fils aussi complexe que nécessaire.
- L'indexation des moteurs de recherche les plus répandus se fait sur le contenu des éléments et non sur les valeurs d'attribut.

Une balise de clôture ne comporte que le nom de l'élément :

< / nom-élément >

Afin d'éviter toute confusion, il est préférable de recourir à une section CDATA lorsqu'on utilise des caractères spéciaux :

< nom_élément> < ! [CDATA [texte brut avec caractères spéciaux]] >
< / nom-élément >

Documents valides et bien formés

XML distingue 2 types de documents :

- Les documents bien formés, c'est-à-dire qu'ils obéissent aux règles syntaxiques du langage XML.
- Les documents valides, qui obéissent à une structure type définie explicitement dans une DTD.

DTD

Définition

Les documents valides obéissent à une structure type prédéfinie : une DTD, qui définit le vocabulaire à utiliser (nom d'éléments et d'attributs) et les relations entre les éléments et attributs nommés. Une DTD précise une règle pour chaque type d'élément précisant ce que chacun peut ou doit contenir. Le document XML devra contenir les éléments et attributs spécifiés dans la DTD pour être déclaré valide.

On définit la DTD dans le prologue du document XML, comme décrit précédemment. Si la DTD n'apparait pas dans le prologue, elle est stockée ailleurs et on doit y faire référence :

```
< ! DOCTYPE nom_élément_racine SYSTEM " nom_élément_racine.dtd
" >
```

Exemple :

```
< ! DOCTYPE nom_élément_racine [
        < ! ELEMENT nom_élément_racine (nom_élément_fils_1) >
        < ! ELEMENT nom_élément_fils_1 (nom_sous _élément_fils) >
        < ! ELEMENT nom_sous _élément_fils (#PCDATA) >
] >
```

Ici, on définit que l'élément racine doit contenir 1 élément fils : l'élément fils 1. Ce dernier doit contenir lui-même 1 élément fils : le sous-élément fils. Ce dernier est un élément contenant uniquement du texte brut.

Contenu

Une DTD définit un type d'élément et associe un nom de type à un modèle de contenu. Tous les éléments instances de ce type qui apparaîtront dans un document devront avoir un contenu conforme au modèle défini dans la déclaration.

Un modèle de contenu peut autoriser la création d'éléments qui contiendront :
- Un ou des éléments fils spécifiés
- Des données représentées par un flot de caractères
- Un mélange de données et d'éléments fils spécifiés
- N'importe quel contenu respectant la syntaxe XML

Le modèle peut aussi imposer qu'un élément reste toujours vide

- Eléments fils

Les éléments fils peuvent avoir un ordre imposer et former une séquence, qui est alors spécifiés par la liste de nom des éléments fils séparés par une virgule.

< ! ELEMENT nom_élément_père (nom_élément_fils_1, nom_élément_fils_2, nom _élément_fils_3) >

Tout élément du type père devra contenir un élément fils 1, un élément fils 2 et un élément fils 3, dans cet ordre.

Les éléments fils peuvent être en ordre libre, les types d'éléments autorisés apparaissant alors séparés par une barre verticale.

< ! ELEMENT nom_élément_père (nom_élément_fils_1 **I** nom_élément_fils_2) >

Tout élément du type père devra contenir un élément fils 1 et un élément fils 2, dans un ordre quelconque.

Un nom d'élément apparaissant dans le modèle d'un élément père peut être suffixé par un opérateur :

- ? indique que l'élément fils peut apparaître zéro ou une fois dans le contenu de son élément père
- * indique que l'élément fils peut apparaître zéro, une ou plusieurs fois dans le contenu de son élément père
- + indique que l'élément fils doit apparaître une fois au moins, ou plusieurs fois dans le contenu de son élément père
- l'absence d'opérateur signifie que l'élément fils doit apparaître une et une seule fois dans le contenu de son élément père

< ! ELEMENT nom_élément_père (nom_élément_fils_1 +, nom_élément_fils_2 ?) >

Tout élément du type père devra contenir au moins une fois l'élément fils 1, puis un élément fils 2 optionnel.

Attention, les exemples cités ci-dessus ne sont pas des DTD complètes, les définitions des éléments fils 1, 2 ou 3 n'y figurent pas !

- Données

La présence de données dans le contenu d'un élément est indiquée comme suit :

< ! ELEMENT nom_élément (#PCDATA) >

Selon cette déclaration, l'élément ne peut contenir qu'un flot de données et non des éléments fils.

Il n'est pas possible de déclarer un modèle de contenu d'élément comme une section CDATA, qui apparaîtra plutôt dans le contenu d'un élément ayant un modèle #PCDATA.

On pourra donc avoir dans une DTD :

< ! ELEMENT nom_élément (#PCDATA) >

Qui sera traduit dans le corps du document XML par :

< nom_élément > blablabla < ! [CDATA [< ! ELEMENT nom_élément (#PCDATA) >]] > blablabla < / nom-élément >

Qui sera affiché à l'écran comme suit :

blablabla < ! ELEMENT nom_élément (#PCDATA) > blablabla

Déclaration d'attributs d'éléments

Un attribut est une paire nom="valeur" que l'on associe à un élément. La déclaration d'attribut dans une DTD permet de spécifier quels sont les attributs qui pourront ou devront être associés à un élément, en précisant le type de l'attribut et sa déclaration :

< ! ATTLIST nom_élément nom_attribut TYPE DECLARATION >

- Type d'attributs

Le type d'attribut peut être :
- CDATA : la valeur de l'attribut sera une chaîne de caractère.
- ID : la valeur de l'attribut est l'identifiant unique de son élément. Les valeurs de tous les attributs de type ID doivent donc être distinctes. Cet attribut peut devenir la cible d'un renvoi, d'un lien.
- IDREF : la valeur de l'attribut sera la valeur d'un attribut de type ID. La valeur donnée doit donc existée dans les documents.
- IDREFS : la valeur de l'attribut pourra être la valeur de divers attributs de type ID. Les valeurs données doivent donc exister dans le document.

- Déclaration d'attributs

La déclaration d'attribut peut prendre 4 formes :
- La valeur par défaut de l'attribut

- #REQUIRED : l'attribut doit obligatoirement être présent dans chaque instance de l'élément du type déclaré. Le créateur du document XML devra obligatoirement la valeur de cet attribut pour l'élément auquel il se rattache.
- #IMPLIED : la présence de l'attribut dans une instance de l'élément est facultative.
- #FIXED 'Valeur_de_attribut' : l'attribut doit toujours prendre la valeur indiquée dans toute instance de l'élément déclaré.

Limites

Le type de IDREF ou IDREFS n'est pas contraint : ce qui signifie que, dans l'exemple ci-dessous, on peut parfaitement s'indiquer soi-même comme son propre père ou sa propre mère, sans que le logiciel détecte un erreur.

Exemple :

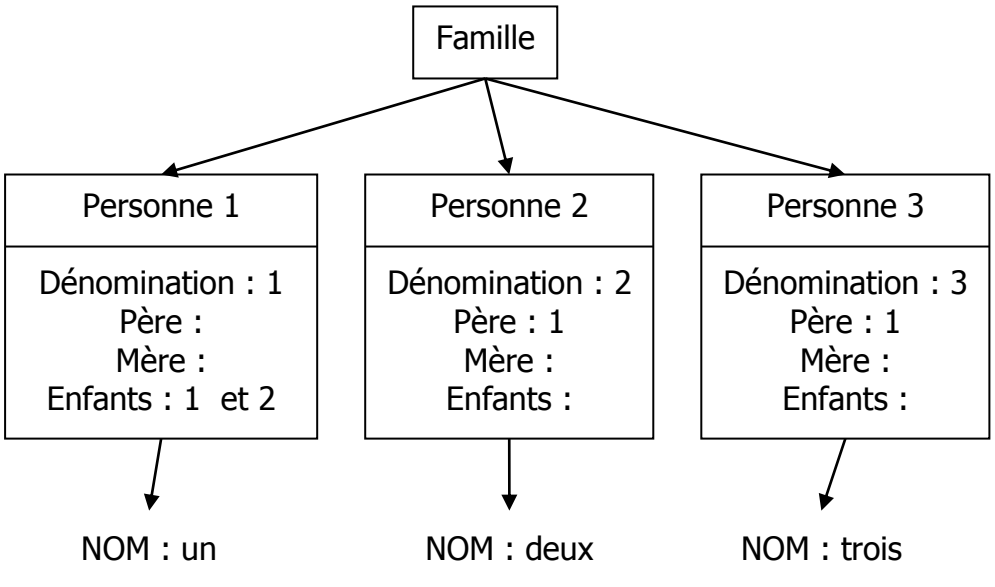
```
< ! DOCTYPE famille [
        < ! ELEMENT famille ( personne * ) >
        < ! ELEMENT personne ( nom ) >
        < ! ELEMENT nom ( #PCDATA ) >
                < ! ATTLIST personne
                        denomination ID #REQUIRED
                        mere IDREF #IMPLIED
                        pere IDREF #IMPLIED
                        enfants IDREFS #IMPLIED >
]>
```

La DTD ci-dessus indique que :

L'élément racine famille contient l'élément fils personne, pouvant apparaître zéro, une ou plusieurs fois au sein de l'élément père famille.

L'élément personne contient lui-même l'élément fils nom.

L'élément fils ne contient que des données de texte.



Une personne va être décrite grâce à 4 attributs :
- dénomination :

- ⇨ qui est l'identifiant unique permettant de renvoyer à la personne nommée
- ⇨ qui doit être définit pour chaque personne

– mère :
- ⇨ qui va prendre la valeur de la dénomination d'une autre personne
- ⇨ qui peut être ou non défini pour chaque personne

– père :
- ⇨ qui va prendre la valeur de la dénomination d'une autre personne
- ⇨ qui peut être ou non défini pour chaque personne

– enfants :
- ⇨ qui va prendre la valeur de la dénomination d'une ou plusieurs autres personnes
- ⇨ qui peut être ou non défini pour chaque personne

Le document XML ci-dessous serait considéré comme bien formé et valide par rapport à la DTD citée plus haut :

```
< famille >

    < personne  denomination = "1"
                enfant = "2 3"   >
    <nom> un </nom>
    </personne>

    < personne  denomination = "2"
                pere="1">
    <nom> deux </nom>
    </personne>

    < personne  denomination = "3"
                pere="1">
    <nom> trois </nom>
    </personne>

< / famille >
```

Espace de nom

Définition

Il est possible d'utiliser dans un même document XML des ensembles d'éléments et d'attributs en provenance d'origines diverses. Des conflits de nom sont alors possibles.

Les espaces de nom garantissent l'unicité des noms d'éléments et d'attributs utilisés dans un document XML, quelque soit leur provenance.

Principe

Soit une ressource quelconque, définit de manière unique par un URI. Si un auteur souhaite utiliser des éléments et attributs contenus dans cette ressource dans le document XML qu'il est en train de créer, il peut le faire en préfixant le nom de l'élément qu'il souhaite réutiliser par l'adresse URI de la ressource à laquelle il fait appel.

Pour plus de praticité, on peut définir une abréviation pour l'adresse URI de la ressource utilisée, qui sera indiquée en préfixe des noms d'éléments ou d'attributs récupérés.

On définit les abréviations des l'URI utilisés dans la balise ouvrante de l'élément racine du document XML, grâce à l'attribut prédéfinit « xmlns : ». On peut dès lors utiliser dans le document XML les éléments et attributs définis dans les ressources extérieures, en préfixant leurs noms par l'abréviation définie dans l'élément racine :

```
< ?xml version = "1.0" encoding = " ">

< ! DOCTYPE nom_élément_racine SYSTEM " nom_élément_racine.dtd
" >

< nom_élément_racine xmlns : abréviation_1 = "URI_1"
                     xmlns : abréviation_2 = "URI_2" >

    < abréviation_1 : nom_élément_fils_1 >
    xxxxxxxx
    < / nom_élément_fils_1 >

    < abréviation_2 : nom_élément_fils_2
                  abréviation_2 : nom_attribut = "valeur" >
    xxxxxxxx
    < / nom_élément_fils_2 >

< / nom_élément_racine >
```

Espace de nom par défaut

Il est possible de déclarer un espace de nom qui va s'appliquer à l'élément dans lequel il est déclaré et à tous ses fils, même en l'absence d'un préfixe, simplement en ne définissant pas d'abréviation pour la ressource à utiliser par défaut, et en en définissant une pour les autres ressources utilisés. Le domaine nominal par défaut peut être modifié pour l'un des éléments fils.

```
< ?xml version = "1.0" encoding = " ">

< ! DOCTYPE nom_élément_racine SYSTEM " nom_élément_racine.dtd
" >

< nom_élément_racine xmlns : abréviation_1 = "URI_1"
                              xmlns = "URI_2" >

        < abréviation_1 : nom_élément_fils_1 >
        xxxxxxxx
        < / nom_élément_fils_1 >

        < nom_élément_fils_2 nom_attribut = "valeur" >
        xxxxxxxx
        < / nom_élément_fils_2 >

        < nom_élément_fils_3 xmlns = " " >
        xxxxxxxx
        < / nom_élément_fils_3 >

< / nom_élément_racine >
```

Ici, l'élément fils 1 est récupéré de la ressource URI 1, comme le prévoit l'abréviation 1 placé en préfixe.

L'élément fils 2 n'est précédé d'aucun préfixe. Il sera donc récupéré, par défaut, de l'URI 2.

L'élément fils 3 présente un domaine nominal redéfini, prenant pour valeur une chaîne vide. En conséquence, l'élément fils 3 et tous ses fils ne sont pas préfixés par une URI.

XPath

Définition

XML peut être considéré comme un métalangage, c'et à dire un langage permettant de définit d'autres langages. Le modèle générique, dont toute structure de données XML est une spécialisation, prend la forme d'un arbre.

XPath est un langage général d'adressage dans cet arbre XML, c'est-à-dire qu'il va proposer une représentation abstraite du document pour localiser des fragments, les atteindre et les traiter. En effet, tout type de manipulation nécessite la localisation préalable du fragment auquel elle s'applique.

Modèle d'arbre XPath

Le langage XPath n'a de sens que par rapport à un modèle de document. Ce modèle est un arbre, constitué de nœuds reliés par des relations.

XPath définit différents types de nœuds :
- Le nœud racine ou nœud document
- Le nœud élément, étiqueté par le nom de l'élément qu'il représente
- Le nœud texte, étiqueté par le fragment de texte qu'il représente
- Le nœud attribut, étiqueté par le nom et la valeur de l'attribut qu'il représente

La présence du nœud racine est unique et obligatoire dans un arbre XML donné.

Mais il ne faut pas le confondre avec le nœud de l'élément racine du document, qui est le fils unique du nœud racine et dont descendent tous les éléments fils du document.

Le nœud de l'élément racine du document, rattaché au nœud racine lui-même, contient la liste ordonnée de ses nœuds fils et la liste non ordonnée de ses nœuds rattachés.

Un nœud fils d'un élément peut être un autre nœud élément ou un nœud texte : ces derniers sont appelés ses fils.

Un nœud élément peut aussi contenir une liste de nœuds attributs, mais la relation entre un nœud élément et un nœud attribut n'est pas une relation de filiation : il ne sera donc pas nommé fils du nœud élément.

Les nœuds attribut et texte ne peuvent avoir de nœuds fils : seul les nœuds élément peuvent en avoir.

La relation de filiation entre un nœud père et un nœud fils préserve l'ordre entre les fils, alors que la relation entre un nœud élément et l'ensemble de ses nœuds attributs ne préserve par l'ordre entre ces attributs.

Les nœuds fils de l'élément père doivent donc être ordonnés selon l'ordre de lecture du document : de gauche à droite, et le plus en profondeur possible.

Chemins XPath

Ils permettent de localiser un fragment de document, donc des nœuds, dans une arbre XML.

Un chemin XPath est une construction se composant d'une suite de termes XPath, chacun d'entre eux sélectionnant un nœud, relativement au nœud sélectionné par le terme précédent figurant dans le chemin.

Chaque terme sélectionne un nœud courant dans l'arbre du document cible, nœud par rapport auquel le terme suivant va s'appliquer pour ajouter une condition de sélection.

Les termes successifs sont séparés par le caractère /.

Un caractère particulier permet de sélectionner le nœud racine : /. Ce dernier apparaît donc en début de tout chemin XPath.

Termes XPath

Chaque terme XPath utilisé dans un chemin est composé :
- D'un axe qui indique la relation entre le nœud sélectionné par le terme précédent et les nœuds que le terme vise à sélectionner.
- D'un test de nœud, séparé par les caractères :: de l'axe, qui précise quelles caractéristiques doivent avoir les nœuds candidats pour être retenus dans la liste de nœuds satisfaisant les conditions énoncées par le terme.
- Eventuellement d'un prédicat, c'est-à-dire d'une contrainte, entre […].

axe : : test de nœud [prédicat]

- Axe

Les axes indicateurs de relation autorisés sont :
- child : sélectionne les nœuds fils du nœud désigné par le terme précédent
- descendant : sélectionne les nœuds descendants du nœud désigné par le terme précédent, c'est-à-dire tous les nœuds descendants du nœud désigné par le terme précédent liés par une relation de filiation, ce qui exclu les nœuds attribut
- parent : sélectionne le nœud père du nœud désigné par le terme précédent
- following-sibling : sélectionne les nœuds frère du nœud désigné par le terme précédent, tous fils du même nœud père, vers la droite, dans l'ordre de lecture du document
- preceding-sibling : sélectionne les nœuds frère du nœud désigné par le terme précédent, tous fils du même nœud père, vers la gauche, dans l'ordre inverse de lecture du document
- following : sélectionne les nœuds à droite du nœud désigné par le terme précédent, dans l'ordre de lecture du document, à l'exception des descendants et des nœuds attribut
- preceding : sélectionne les nœuds à gauche du nœud désigné par le terme précédent, dans l'ordre inverse de lecture du document, à l'exception des descendants et des nœuds attribut
- ancestor : sélectionne les nœuds ancêtres du nœud désigné par le terme précédent, c'est-à-dire tous les nœuds ancêtres du nœud désigné par le terme précédent liés par une relation de filiation, ce qui exclu les nœuds attribut
- self : sélectionne le nœud désigné par le terme précédent lui-même
- ancestor-or-self : sélectionne le nœud désigné par le terme précédent lui-même avec ses nœuds ancêtres

- descendant-or-self : sélectionne le nœud désigné par le terme précédent lui-même avec ses nœuds descendants
- attribute : sélectionne les nœuds attributs du nœud désigné par le terme précédent

Si l'axe n'est pas précisé, le logiciel choisira child par défaut.

- Test de nœud

Les tests de nœud qui vont suivre l'axe peuvent être :
- Un nom d'élément : le terme va sélectionner le ou les nœuds reliés, par la relation précisée par l'axe, au nœud désigné par le terme précédent, et qui porte le nom indiqué.
- Un nom d'attribut : le terme va sélectionner le ou les nœuds attributs reliés obligatoirement par l'axe attribute au nœud désigné par le terme précédent, et qui porte le nom indiqué.
- * : le terme va sélectionner tous les nœuds conformes à la relation précisée par l'axe.
- text ( ) : le terme va sélectionner tous les nœuds texte conformes à la relation précisée par l'axe.
- node ( ) : le terme va sélectionner tous les nœuds conformes à la relation précisée par l'axe.

- Prédicat

Un prédicat est une expression qui peut être évaluée à l'une des valeurs Vrai ou Faux.

Un prédicat peut être composé d'une combinaison booléenne and / or d'autres prédicats.

child : : * [condition 1 and condition 2]

Ce terme sélectionnera tous les nœuds fils du nœud désigné par le terme précédent, répondant à la condition 1 et à la condition 2.

Un prédicat peut également être une égalité ou une inégalité définit par les opérateurs habituels.

Un prédicat peut aussi contenir l'une des fonctions ci-dessous :
- last ( ) : retourne le nombre de nœuds du chemin désigné par les termes précédents.
- position ( ) : retourne le nœud désigné par le terme, dans la liste des nœuds formant le chemin désigné par les termes précédents (de 1 à last ( )).
- count (liste_de_noeud) : retourne le nombre de nœuds compris dans la liste de nœud donné en argument.

Abréviations

child : :	
attribute : :	@
/ descendant-or-self : : node ( ) /	//
descendant-or-self : : test_de_noeud	/ test_de_noeud
[position ( ) = X]	[X]
self : : node ( )	.
parent : : node ( )	. .