XML : eXtensible Markup Language

Dr. SAGAR Samya

Pré-requis du cours

- *Assurez-vous d'être en pleine possession de la pratique et des connaissances nécessaires et suffisantes:*
  - *HTML/XHTML;*
  - *CSS;*
  - *JavaScript/Jscript;*
  - *Architecture générale du Web.*

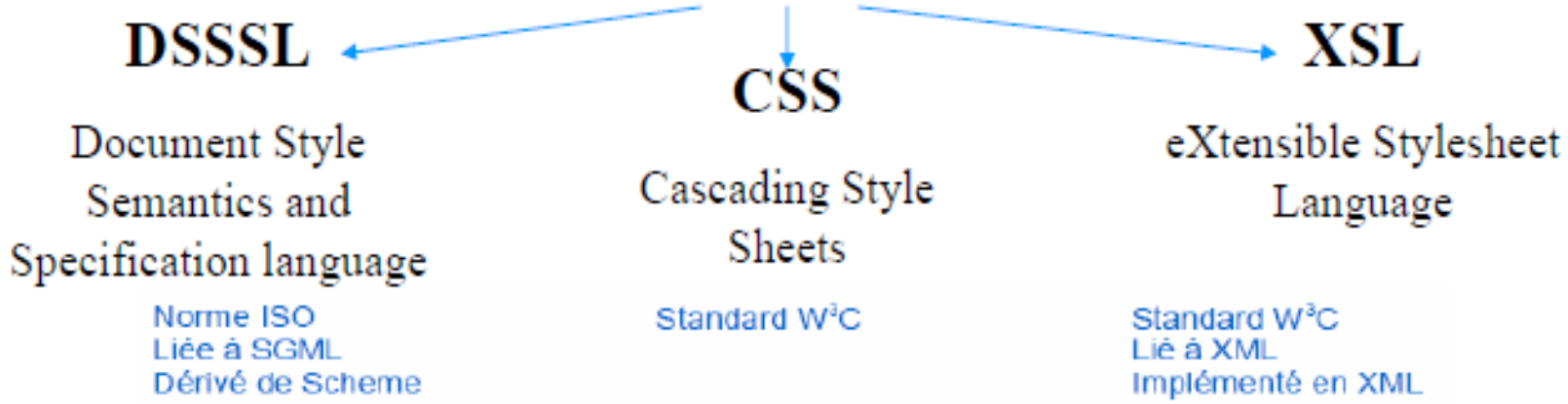
Plan

- Fondements de la technologie XML
- Concepts de base
- DTD : Définition de Type de Document
- XML Schéma
- Codage des caractères et espaces de noms
- Modélisation : XDM
- Présentation et transformations de documents

Présentation et transformations de documents

Formatage de document XML

- Un document et sa DTD ou schéma ne donnent pas d'indications sur sa représentation.
  - Une description supplémentaire est nécessaire
- Un fichier XML peut être à loisir mis en forme et adapté à des formats de visualisation variés, comme par exemple une page Web ou un document PDF.



Cascading Style Sheets – CSS

- Recommandation W3C (CSS1: 1996, CSS2: 1998)
- S'applique à HTML et XML
- Support approximatif par les navigateurs
  - Voir http://www.richinstyle.com/bugs/table.html
  - Meilleurs support dans Mozilla et IE6
- Principes
  - Cascade
  - Correspondance d'éléments (sélecteurs)

Cascading Style Sheets – CSS

Syntaxe

- Attachement d'une feuille de style à un document
  - `<xml-stylesheet type="text/css" href="livre.css"?>`
- Syntaxe générale
  - `Sélecteur {propriété: valeur; propriété: valeur;…}`
- Sélecteur
  - Voir cours « feuille de styles CSS »

Cascading Style Sheets – CSS

Faiblesses de CSS

- CSS a été initialement prévu pou la présentation des documents HTML
- Même défauts que HTML
  - Syntaxe non modifiable et non extensible
  - Syntaxe difficile à normaliser
  - Difficultés pour trouver des éléments

eXtensive Stylesheet Langage – XSL

- Langage extensible de feuille de style.
- Cette abréviation recouvre en fait trois langages :
  - ***XPath*** désigne un moyen d'accéder à un nœud quelconque de l'arborescence d'un document XML à partir d'un autre nœud quelconque.
  - ***XSLT*** signifie *eXtensible Stylesheet Langage Transformation*.
  - ***XSL-FO*** signifie *eXtensible Stylesheet Langage - Formating Objects*, et désigne un langage permettant le contrôle de la mise en page finale de la transformation. Ce langage est particulièrement destiné à la production de contenus au format PDF.

eXtensive Stylesheet Langage – XSL

- Le fonctionnement du XSL est fondé sur les manipulations de modèles (**templates**).
- Les éléments du document XML d'origine sont remplacés (ou légèrement modifiés) par ces modèles.
- Un modèle contient ainsi le texte (éléments, attributs, texte...) de remplacement d'un élément donné.
- Tout élément pouvant être remplacé dans le fichier de sortie par tout type de contenu texte,
- XSL est un outil privilégié de production de fichiers HTML à partir de sources XML.
- Un fichier XSL étant un fichier XML, il doit respecter les normes de syntaxe de ce format.

eXtensive Stylesheet Langage – XSL

Structure d'un document XSL

- La structure de base d'un document XSL
  - commence par un *prologue*,
  - puis un élément `<xsl:stylesheet>` pouvant contenir quelques attributs, notamment une déclaration d'espace de noms ainsi que le numéro de version.
- Exemple

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 (...)
</xsl:stylesheet>
```

- L'élément `<xsl:stylesheet>` est l'élément **racine** du document XSL.
  - C'est lui qui contient tous les modèles, y compris celui qui est associé à la racine du document XML, modèle que l'on note `<xsl:template match="/">`.
  - L'attribut `match="/"` indique que ce modèle s'applique à la racine du document XML.

eXtensive Stylesheet Langage – XSL

Les expressions de sélection

- Connues en anglais sous le nom de XSL patterns,
- les expressions de sélection sont des chaînes de caractères qui permettent de sélectionner des **nœuds** dans un document source.
- Il est également possible d'utiliser une syntaxe spéciale, appelée XPath, qui permet, en se fondant sur la structure arborescente du document XML
  - le Document Object Model (DOM) se basant sur XDM, de faire référence à des éléments et/ou des attributs.

eXtensive Stylesheet Langage – XSL

Sélection d'élément -- syntaxe de base (1)

- L'exemple le plus simple d'expression de sélection est le ***nom*** d'un type d'élément.
  - Cette expression sélectionne tous les éléments du type précisé descendant ou ancêtre d'un nœud donné.
- Exemple:

```
<xsl:value-of select="nom_element" />
```

- L'opérateur **/** permet de définir le chemin d'accès aux éléments à sélectionner, et donc leur parenté.
  - Par exemple, **section/paragraphe**
    - sélectionne les éléments **section** du nœud courant et pour chaque élément **section**, sélectionne les éléments **paragraphe** qu'il contient.
    - En d'autres termes, cette expression sélectionne les petits-fils **paragraphe** du nœud courant qui ont pour père un nœud **section**.

eXtensive Stylesheet Langage – XSL

Sélection d'élément -- syntaxe de base (2)

- Un nom d'élément peut être remplacé par `*` dans une expression.
  - Par exemple, **`*/ paragraphe`** sélectionne tous les petits-fils **`paragraphe`** quel que soit leur père.
- L'utilisation de `//` permet d'appliquer la recherche aux descendants et non pas seulement aux fils directs.
  - Par exemple, **`section//paragraphe`** sélectionne tous les éléments **`paragraphe`** descendant d'un élément **`section`** fils direct du nœud courant.
- Le caractère `.` sélectionne le nœud courant.
  - Par exemple, **`.//paragraphe`** sélectionne tous les descendants **`paragraphe`** du nœud courant.
- La chaîne `..` sélectionne le père du nœud courant.
  - Par exemple, **`../paragraphe`** sélectionne tous les nœuds **`paragraphe`** frères du nœud courant.

eXtensive Stylesheet Langage – XSL

Sélection d'élément -- appel de fonctions

- L'expression `comment()` sélectionne tous les nœuds commentaires fils du nœud courant.
- L'expression `text()` sélectionne tous les nœuds fils du nœud courant, ne contenant que du texte.
- L'expression `node()` sélectionne tous les nœuds fils du nœud courant.
- L'expression `id("UnIdentifiant")` sélectionne l'élément, normalement unique, qui a un attribut `attr` de type ID valant `"UnIdentifiant"`.

eXtensive Stylesheet Langage – XSL

Sélection d'élément et DOM (1)

- Il est possible de « naviguer » dans les branches de l'arborescence du document XML, en utilisant les ressources du DOM. Les différents types de syntaxes sont fondées sur une expression de la forme `Element[Expression]` :
  - **`elt[i]`** où `i` est un nombre entier désigne le i-ème descendant direct d'un même parent ayant le nom indiqué. Par exemple, `paragraphe[3]` désigne le 3ème enfant de l'élément courant, portant le nom `paragraphe`.
    - ***Attention***, la numérotation commence à 1 et non à 0.
  - **`elt[position()>i]`** où `i` est un nombre entier sélectionne tous les éléments précédés d'au moins i éléments de même nom comme descendants du même parent. Par exemple, `paragraphe[position()>5]` sélectionne tous les éléments `paragraphe` dont le numéro d'ordre est strictement supérieur à 5.
  - **`elt[position() mod 2=1]`** sélectionne tout élément qui est un descendant impair.

eXtensive Stylesheet Langage – XSL

Sélection d'élément et DOM (2)

- `elt[souselt]` sélectionne tout élément `elt` qui a au moins un descendant `souselt` (à ne pas confondre avec `elt/souselt`, qui sélectionne tout élément `souselt` ayant pour parent `elt`...).
- `elt[first-of-any()]` sélectionne le premier élément `elt` fils de l'élément courant.
- `elt[last-of-any()]` sélectionne le dernier élément `elt` fils de l'élément courant.
- `elt[first-of-type()]` sélectionne l'élément `elt` fils de l'élément courant, s'il est premier de son type.
  - Considérons l'exemple suivant. L'élément `elt` peut contenir des nœuds de type texte `elt1` et `elt2` dans n'importe quel ordre. On cherche à évaluer l'expression `elt2[first-of-type()]`. Deux cas se présentent : soit `elt` commence par au moins un élément elt1, avec éventuellement un élément `elt2` ensuite, soit il commence par un élément `elt2`. L'expression ne sélectionne le premier élément `elt2` que dans le second cas, où il n'est précédé par aucun élément de même type.
- `elt[last-of-type()]` sélectionne de même l'élément `elt` fils de l'élément courant, s'il est le dernier de son type.

eXtensive Stylesheet Langage – XSL

Sélection d'élément et DOM (3)

- Exemple:

  l'expression **`section/paragraphe[last-of-type() and first-of-type()]`** sélectionne les éléments **`paragraphe`** fils uniques dont le père est un élément **`section`** ; l'expression **`section/paragraphe[last-of-any() and first-of-any()]`** sélectionne les éléments **`paragraphe`** dont le père est un élément **`section`** qui ne contient qu'un seul élément **`paragraphe`**.

- La fonction **`ancestor()`** permet la sélection d'un ancêtre du nœud courant. Elle reçoit en argument une expression de sélection et recherche le premier ancêtre du nom correspondant à la sélection.
  - Par exemple, **`ancestor(chapitre)/titre`** sélectionne l'élément titre du **`chapitre`** contenant l'élément courant.

eXtensive Stylesheet Langage – XSL

Sélection d'attributs

- Les attributs d'un élément sont sélectionnés en faisant précéder leur nom par le caractère `@`. Les règles relatives à la sélection des éléments s'appliquent également aux attributs :
  - **`section[@titre]`** sélectionne les éléments **`section`** qui ont un attribut **`titre`**.
  - **`section[@titre="Introduction"]`** sélectionne les éléments **`section`** dont l'attribut titre a pour valeur **`Introduction`**.
- Si l'on veut ***afficher*** le contenu de l'attribut, on le fait précéder du caractère `/`.
  - Par exemple, **`<xsl:value-of select="paragraphe/@titre" />`** permet l'affichage du **`titre`** de l'élément **`paragraphe`** fils de l'élément courant (si rien n'est précisé, par défaut il s'agit du premier élément **`paragraphe`** fils).

eXtensive Stylesheet Langage – XSL

Opérateurs logiques

- Les opérateurs logiques :
  - `not()`, `and` et `or`
- peuvent être utilisés, comme par exemple `section[not(@titre)]`, qui sélectionne les éléments `section` qui n'ont pas d'attribut `titre`.
- ***Attention*** : lorsque, dans la DTD par exemple, l'attribut est défini comme ayant une valeur par défaut, même s'il n'est pas explicité dans le document XML, il est considéré comme existant.

eXtensive Stylesheet Langage – XSL

Exemple

- Soit le document XML suivant:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="bouteille1.xsl"?>
<bouteille>
  <marque>Cristaline</marque>
  <composition>calcium 71mg/l, magnésium 5,5mg/l, chlorure
20mg/l, nitrate 1mg/l, traces de fer.</composition>
  <source>
    <ville>St-Cyr la Source</ville>
    <departement>Loiret</departement>
  </source>
  <code_barre>3274080005003</code_barre>
  <contenance>150cl</contenance>
  <ph>7,45</ph>
</bouteille>
```

eXtensive Stylesheet Langage – XSL

Exemple (suit)

- Et voici la feuille de style XSL (simple) associée:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
    <head>
      <title>Exemple de sortie HTML</title>
      <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    </head>
    <body>
      <h1>Bouteille de marque <xsl:value-of select="bouteille/marque" /></h1>
      <h2>Composition:</h2>
      <p><xsl:value-of select="bouteille/composition" /></p>
      <h2>Lieu d'origine:</h2>
      <p>Ville de <b><xsl:value-of select="bouteille/source/ville" /></b>, dans le
département <b><xsl:value-of select="bouteille/source/departement" /></b></p>
      <h2>Autres informations</h2>
      <ul>
        <li>Contenance: <xsl:value-of select="bouteille/contenance" /></li>
        <li>pH: <xsl:value-of select="bouteille/ph" /></li>
      </ul>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

eXtensive Stylesheet Langage – XSL

Exemple (suit)

- Utilisation de boucle

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
    <head>
      <title>Exemple de sortie HTML</title>
      <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    </head>
    <body>
      <h1>Bouteille de marque <xsl:value-of select="bouteille/marque" /></h1>
      <h2>Composition:</h2>
      <h3>Ions positifs</h3>
      <ul>
        <xsl:for-each select="bouteille/composition/ion_positif">
          <li><xsl:value-of select="." /></li>
        </xsl:for-each>
      </ul>
      <h3>Ions négatifs</h3>
      <ul>
        <xsl:for-each select="bouteille/composition/ion_negatif">
          <li><xsl:value-of select="." /></li>
        </xsl:for-each>
      </ul>
```

eXtensive Stylesheet Langage – XSL

Exemple (suit)

- Utilisation de boucle (suit)

```
      <h3>Autres matériaux</h3>
      <ul>
        <xsl:for-each select="//autres_materiaux">
          <li><xsl:value-of select="." /></li>
        </xsl:for-each>
      </ul>
<h2>Lieu d'origine</h2>
      <p>Ville de <b><xsl:value-of select="bouteille/source/ville" /></b>, dans le
département <b><xsl:value-of select="bouteille/source/departement" /></b></p>
      <h2>Autres informations</h2>
      <ul>
        <li>Contenance: <xsl:value-of select="bouteille/contenance" /></li>
        <li>pH: <xsl:value-of select="bouteille/ph" /></li>
      </ul>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Moyen d'accès aux nœuds – XPath

Présentation

- XPath est une spécification fondée sur l'utilisation de **chemin d'accès** permettant de se déplacer au sein du document XML.
  - Dans ce but, un certain nombre de fonctions ont été définies.
    - Ces fonctions permettent de traiter les chaînes de caractères, les booléens et les nombres.
- Le XPath établit un **arbre de nœuds** correspondant au document XML.
- Les **types de nœuds** peuvent être différents :
  - nœud d'élément,
  - nœud d'attribut
  - nœud de texte.
- En vue d'une utilisation plus aisée, le XPath comprend un mécanisme qui associe à tous ces types une chaîne de caractères.

Moyen d'accès aux nœuds – XPath

Syntaxe de base

- Sa syntaxe est fondée sur l'utilisation d'expressions.
- Une expression peut s'appliquer à quatre types d'objets :
  - un ensemble non ordonné de nœuds ;
  - une valeur booléenne (vrai ou faux) ;
  - un nombre en virgule flottante ;
  - une chaîne de caractères.
- Chaque évaluation d'expression dépend du contexte courant.
- Une des expressions les plus importantes dans le standard XPath est le ***chemin de localisation***.
  - Cette expression sélectionne un ensemble de nœuds à partir d'un nœud contextuel.

Moyen d'accès aux nœuds – XPath

Chemin de localisation (1)

- Un chemin de localisation peut être de type absolu ou relatif.
  - Dans le cas où il est de type ***absolu***, il commence toujours par le signe **/** indiquant la racine du document XML ;
  - Dans le cas où il est de type ***relatif***, le nœud de départ est le nœud contextuel courant.
- La syntaxe de composition d'un chemin de localisation peut être de type ***abrégé*** ou ***non abrégé***.
  - Toute syntaxe non abrégée ne trouve pas forcément d'équivalence en syntaxe abrégée.

Moyen d'accès aux nœuds – XPath

Chemin de localisation (2)

- Un chemin de localisation est composé de trois parties :
  - un ***axe***, définissant le sens de la relation entre le nœud courant et le jeu de nœuds à localiser;
  - un ***nœud*** spécifiant le type de nœud à localiser;
  - 0 à n ***prédicats*** permettant d'affiner la recherche sur le jeu de nœuds à récupérer.
- Exemple:
  - dans le chemin `child::section[position()=1]`, `child` est le nom de l'axe, `section` le type de nœud à localiser (élément ou attribut) et `[position()=1]` est un prédicat. Les doubles `::` sont obligatoires.
  - **Note** : La syntaxe d'une localisation s'analyse de gauche à droite. Dans notre cas, on cherche dans le nœud courant, un nœud `section` qui est le premier nœud de son type.

Moyen d'accès aux nœuds – XPath

Chemin de localisation -- Axes

- **`child`** : contient les enfants directs du nœud contextuel.
- **`descendant`** : contient les descendants du nœud contextuel. Un descendant peut être un enfant, un petit-enfant...
- **`parent`** : contient le parent du nœud contextuel, s'il y en a un.
- **`ancestor`** : contient les ancêtres du nœud contextuel. Cela comprend son père, le père de son père... Cet axe contient toujours le nœud racine, excepté dans le cas où le nœud contextuel serait lui-même le nœud racine.
- **`following-sibling`** : contient tous les nœuds cibles du nœud contextuel. Dans le cas où ce nœud est un attribut ou un espace de noms, la cible suivante est vide.
- **`preceding-sibling`** : contient tous les prédécesseurs du nœud contextuel ; si le nœud contextuel est un attribut ou un espace de noms, la cible précédente est vide.
- **`following`** : contient tous les nœuds du même nom que le nœud contextuel situés après le nœud contextuel dans l'ordre du document, à l'exclusion de tout descendant, des attributs et des espaces de noms.
- **`preceding`** : contient tous les nœuds du même nom que le nœud contextuel situés avant lui dans l'ordre du document, à l'exclusion de tout descendant, des attributs et des espaces de noms.
- **`attribute`** : contient les attributs du nœud contextuel ; l'axe est vide quand le nœud n'est pas un élément.
- **`namespace`** : contient tous les nœuds des espaces de noms du nœud contextuel ; l'axe est vide quand le nœud contextuel n'est pas un élément.
- **`self`** : contient seulement le nœud contextuel.
- **`descendant-or-self`** : contient le nœud contextuel et ses descendants.
- **`ancestor-or-self`** : contient le nœud contextuel et ses ancêtres. Cet axe contiendra toujours le nœud racine.

Moyen d'accès aux nœuds – XPath

Chemin de localisation -- Prédicats

- Le contenu d'un prédicat est une prédiction.
- Chaque expression est évaluée et le résultat est un booléen.
- Exemple:
  - `section[3]` est équivalent à `section[position()=3]`.
  - Ces deux expressions sont équivalentes : chacune d'entre elles produit un booléen.
  - Dans le premier cas, il n'y a pas de test, on sélectionne simplement le troisième élément, l'expression est obligatoirement vraie.
  - Dans le second cas, un test est effectué par rapport à la position de l'élément section ; lorsque la position sera égale à 3, l'expression sera vraie.

Moyen d'accès aux nœuds – XPath

Chemin de localisation -- Syntaxe non abrégée (1)

- `child::para` : sélectionne l'élément *para* enfant du nœud contextuel.
- `child::*` : sélectionne tous les éléments enfants du nœud contextuel.
- `child::text()` : sélectionne tous les nœuds de type texte du nœud contextuel.
- `child::node()` : sélectionne tous les enfants du nœud contextuel, quel que soit leur type.
- `attribute::name` : sélectionne tous les attributs *name* du nœud contextuel.
- `attribute::*` : sélectionne tous les attributs du nœud contextuel.
- `descendant::para` : sélectionne tous les descendants *para* du nœud contextuel.
- `ancestor::div` :sélectionne tous les ancêtres *div* du nœud contextuel.
- `ancestor-or-self::div` : sélectionne tous les ancêtres *div* du nœud contextuel et le nœud contextuel lui-même si c'est un *div*.
- `descendant-or-self::para` : sélectionne tous les descendants *para* du nœud contextuel et le nœud contextuel lui-même si c'est un *para*.

Moyen d'accès aux nœuds – XPath

Chemin de localisation -- Syntaxe non abrégée (2)

- `self::para` : sélectionne le nœud contextuel si c'est un élément para, et rien dans le cas contraire.
- `child::chapitre/descendant::para` : sélectionne les descendants para de l'élément chapitre enfant du nœud contextuel.
- `child::*/child::para` : sélectionne tous les petits-enfants para du nœud contextuel.
- `/child::` : sélectionne l'élément racine du document.
- `/descendant::para` : sélectionne tous les éléments para descendants du document contenant le nœud contextuel.
- `/descendant::olist/child::item` : sélectionne tous les éléments item qui ont un parent olist et qui sont dans le même document que le nœud contextuel.
- `child::para[position()=1]` : sélectionne le premier enfant para du nœud contextuel.
- `child::para[position()=last()]` : sélectionne le dernier enfant para du nœud contextuel.
- `child::para[position()=last()-1]` : sélectionne l'avant-dernier enfant para du nœud contextuel.
- `child::para[position()1]` : sélectionne tous les enfants para du nœud contextuel autres que le premier.
- `following-sibling::para[position()=1]` : sélectionne le prochain para frère du nœud contextuel.

Moyen d'accès aux nœuds – XPath

Chemin de localisation -- Syntaxe abrégée

- Cette syntaxe recoupe en fait la « syntaxe de base ».
- Elle permet d'obtenir des expressions du type `para[@type="avertissement"][5]`,
  - qui sélectionne le cinquième enfant de l'élément para, parmi ceux qui ont un attribut type ayant la valeur avertissement.

Moyen d'accès aux nœuds – XPath

Fonctions de base -- Généralités

- De nombreuses fonctions peuvent être utilisées.
- Ces fonctions concernent quatre catégories d'objets :
  - nœuds,
  - chaînes de caractères,
  - booléens,
  - nombres.
- Chaque fonction peut avoir zéro ou plusieurs arguments.
- Note : lorsqu'un élément est suivi du caractère ?, cela signifie qu'il est optionnel.

Moyen d'accès aux nœuds – XPath

Fonctions de base -- Manipulation de nœuds

- Fonctions retournant un nombre :
  - **`last()`** : retourne un nombre égal à l'index du dernier nœud dans le contexte courant.
  - **`position()`** : retourne un nombre égal à la position du nœud dans le contexte courant.
- Fonction retournant un jeu de nœuds :
  - **`id(objet)`**: permet de sélectionner les éléments par leur identifiant.

Moyen d'accès aux nœuds – XPath

Fonctions de base -- Manipulation de chaînes de caractères

- Beaucoup de fonctions existent, exemples:
- `string(nœud?)` : cette fonction convertit un objet en chaîne de caractères selon les règles suivantes :
  - un ensemble de nœuds est converti en chaîne de caractères en retournant la valeur textuelle du premier nœud de l'ensemble dans l'ordre du document. Si l'ensemble des nœuds est vide, une chaîne vide est retournée.
  - un nombre est converti en chaîne suivant des règles dépendant de sa nature (NaN, nombre entier, non-entier, zéro...).
  - la valeur booléenne false est convertie en chaîne de caractères "false", de même pour la valeur booléenne true.

  Cette fonction n'a pas pour objet de convertir des nombres en chaînes de caractères pour les présenter aux utilisateurs ; il existe des fonctions de transformations XSLT pour ce faire (format-number et xsl:number)
- `concat(chaine1, chaine2, chaine*)` : retourne une chaîne résultant de la compilation des arguments
- `string-length(chaine?)` : cette fonction retourne le nombre de caractères de la chaîne. Dans le cas où l'argument est omis, la valeur retournée est égale à la longueur de la valeur textuelle du nœud courant

Moyen d'accès aux nœuds – XPath

Fonctions de base -- Manipulation de booléens

- Les fonctions logiques
  - `not()`,
  - `true()`,
  - `false()`.
- une autre fonction utile est lang(chaine).
  - Elle teste l'argument chaine par rapport à l'attribut xml:lang du nœud contextuel ou de son plus proche ancêtre dans le cas où le nœud contextuel ne contient pas d'attribut de ce type.
  - La fonction retourne true si l'argument est bien la langue utilisée ou si la langue utilisée est un sous-langage de l'argument (par exemple, en//us). Sinon elle retourne false.

Moyen d'accès aux nœuds – XPath

Fonctions de base -- Manipulation de nombres

- Voici quelques fonctions de manipulations de nombres :
  - **`floor(nombre)`** : retourne le plus grand entier inférieur à l'argument passé à la fonction.
  - **`ceiling(nombre`**) : retourne le plus petit entier supérieur à l'argument passé à la fonction.
  - **`round(nombre)`** : retourne l'entier le plus proche de l'argument passé à la fonction.

Références

- https://www.w3.org/XML/
- https://www.w3.org/Style/XSL/
- https://www.w3.org/standards/xml/
- XML, Cours er Exercices de *Alexandre Brillant* (Eyrolles-*Editions*)
- Cours de *Olivier Carton*, L'essentiel de XML: Cours XML
- Cours de *Tarak CHAARI*, XML: eXtensible Markup Language (Fondements, Modélisation, Présentation et Programmation)
- Cours de *Jacques Le Maitre*, Description, typage, modélisation et interrogation de données XML (XML, XML Schema, XDM, XQuery)

- Autres, voir la bibliothèque de l'ESSTH-Sousse,…