

Cours 3 : Naviguer avec XPATH

XPath

- Le langage **XPath** est utilisé dans de nombreux langages de manipulation de données **XML** (**XSLT**, **XML-Schema**, **XQuery**).
- Une **donnée XML** est un arbre composé de nœuds (de différents types) et **XPath** est un langage permettant de désigner des nœuds dans l'arbre **XML**, en décrivant des chemins dans cet arbre à l'aide d'**expressions de chemin** (un peu comme des chemins **unix** dans le système de fichiers).
- **XPath 1** : juste des expressions de chemin. Recommandation depuis 1999, au cœur de **XSLT**.
- **XPath 2** : des boucles **for**, des conditions **if**, des variables, ... Recommandation depuis 2006 (à l'étude depuis 2001), au cœur de **XQuery**

Modèle de données de XPath 1.0

Le résultat d'une requête **XPath** est un **ensemble de nœuds** qu'il faut voir comme un ensemble de **références** vers des nœuds de l'arbre **XML**.

Qui dit ensemble dit

- **non ordonné** (en réalité on récupère les nœuds dans l'ordre du document)
- **sans doublon**

Modèle de données de XPath 2.0

Le résultat d'une requête **XPath** est une **séquence de nœuds ou de valeurs**.

Qui dit séquence dit

- **ordonné**
- **doublons** possibles

Aujourd'hui XPath 1.0

Le résultat d'une requête **XPath** est un ensemble de nœuds qu'il faut voir comme un ensemble de références vers des nœuds de l'arbre **XML**.

Qui dit ensemble dit

- non ordonné (en réalité on récupère les nœuds dans l'ordre du document)
- sans doublon

XPath 1.0

Typage des nœuds plus simple que **DOM**. Il y a 7 sortes de nœuds :

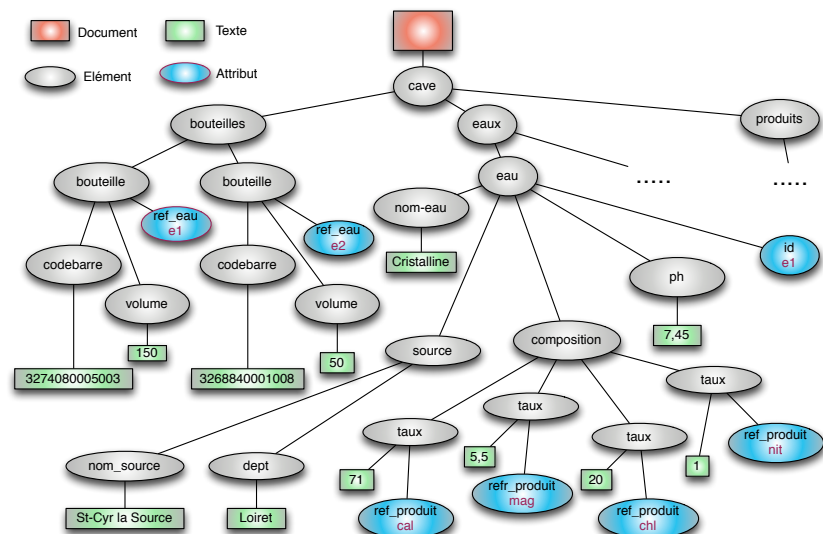
1. document
2. element
3. attribute
4. text
5. namespace
6. processing instruction
7. comment

pas d'entité, pas de section littérale,...

Exemple de document

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE cave SYSTEM "./eaux.dtd">
<cave>
  <bouteilles>
    <bouteille ref_eau="e1"> <codebarre>3274080005003</codebarre> <volume>150</volume> </bouteille>
    <bouteille ref_eau="e2"> <codebarre>3268840001008</codebarre> <volume>50</volume> </bouteille>
    <bouteille ref_eau="e2"> <codebarre>3546543211234</codebarre> <volume>150</volume> </bouteille>
  </bouteilles>
  <eaux>
    <eau id="e1">
      <nom_eau>Cristalline</nom_eau>
      <source><nom_source>St-Cyr la Source</nom_source> <dept>Loiret</dept></source>
      <composition>
        <taux ref_produit="cal">71</taux> <taux ref_produit="mag">5,5</taux>
        <taux ref_produit="chl">20</taux> <taux ref_produit="nit">1</taux>
      </composition>
      <ph>7,45</ph>
    </eau>
    <eau id="e2">
      .....
    </eau>
  </eaux>
  <produits>
    <produit id="cal">
      <nom_produit>Calcium</nom_produit>
      <formule>Ca+</formule>
    </produit>
    .....
  </produits>
</cave>
```

Arbre correspondant



Ordre du document

Les requêtes XPath utilisent un **ordre total sur les nœuds**, appelé **ordre du document**.

- C'est l'ordre en **profondeur d'abord et de gauche à droite** lorsque le document est représenté par un arbre.
- Pour les éléments, c'est donc l'**ordre dans lequel on rencontre les balises ouvrantes** lorsque l'on considère la représentation textuelle du document.
- Les nœuds **namespace** sont **immédiatement successeurs** du nœud élément qui leur est associé.
- Les nœuds **attributs** viennent **immédiatement après** ces nœuds **namespace**.

Attention, l'ordre des nœuds attributs d'un même élément dépend de l'implémentation.

Les expressions de chemin

Une **expression de chemin XPath** :

- s'évalue en fonction d'un **nœud contexte**,
- désigne **un ou plusieurs chemins** dans l'arbre à partir du nœud contexte,
- a pour **résultat**
 - ✓ un **ensemble de nœuds** ou
 - ✓ une **valeur** (numérique, booléenne ou alphanumérique)

Les expressions de chemin

Une expression de chemin XPath consiste en une **séquence d'étapes** séparées par / ou //

Elle peut être

- **absolue** comme /A/B/C ou //A/B//C. Le **nœud contexte** (point de départ) est alors la racine du document (le nœud **document**) ou
- **relative** comme /A/B/C ou A/B//C. Le **nœud contexte** dépend alors...du contexte dans lequel on utilise le chemin XPath.

Les étapes

Une **étape** est composée de 3 composants : un **axe**, un **filtre** et une liste éventuellement vide de **prédicats** en suivant la syntaxe

axe::filtre[prédicat1][prédicat2]...

axe sens de navigation dans l'arbre par rapport au nœud contexte

filtre type des nœuds à retenir

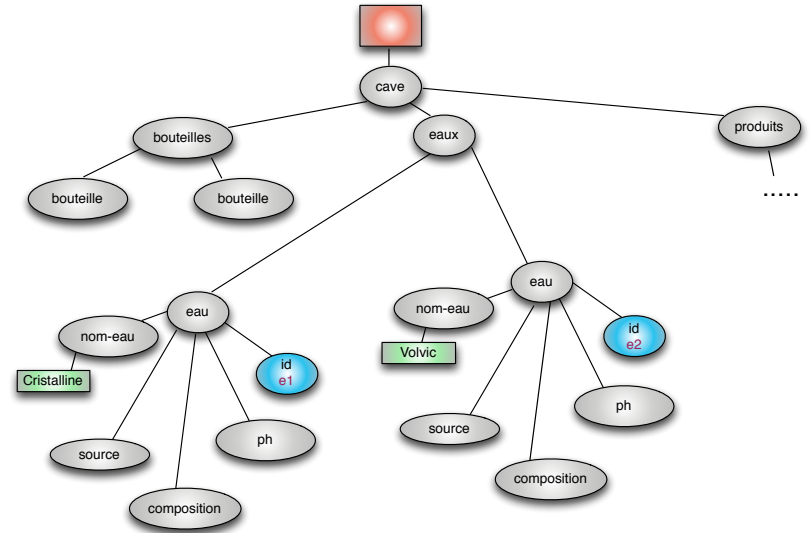
prédicat(s) propriétés que doivent satisfaire les nœuds parmi les nœuds retenus.

/descendant::nom_eau[child::text()='Volvic']/parent::*

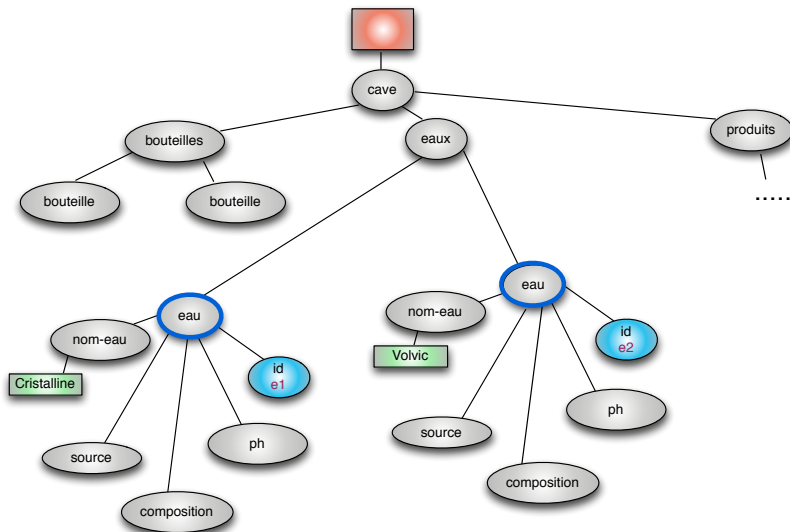
Évaluation d'une expression XPath

- à partir du nœud contexte, on évalue la première étape, on obtient alors un ensemble de nœuds (**node-set**).
- on prend alors, un par un, les nœuds de cet ensemble qui servent **chacun leur tour de nœud contexte** pour la deuxième étape
- à chaque nouvelle étape i , l'ensemble courant résultat S_i de cette étape remplace l'ensemble résultat S_{i-1} qu'on avait à l'étape précédente en appliquant la nouvelle étape à chacun des nœuds de S_{i-1} et en faisant l'union de tous les ensembles obtenus.

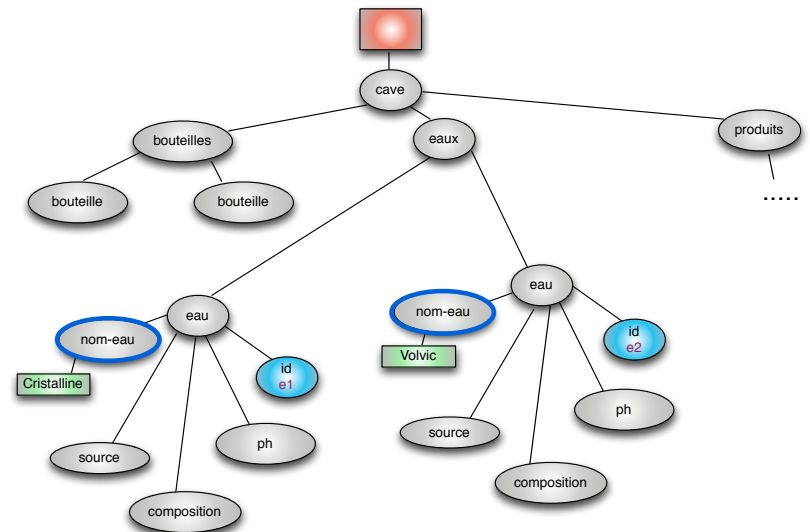
`/descendant::eau/child::nom_eau/child::text()`



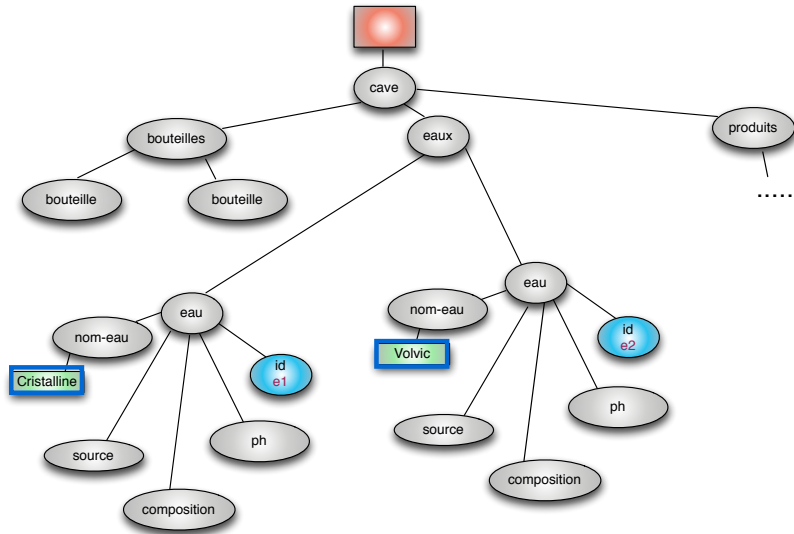
`/descendant::eau/child::nom_eau/child::text()`



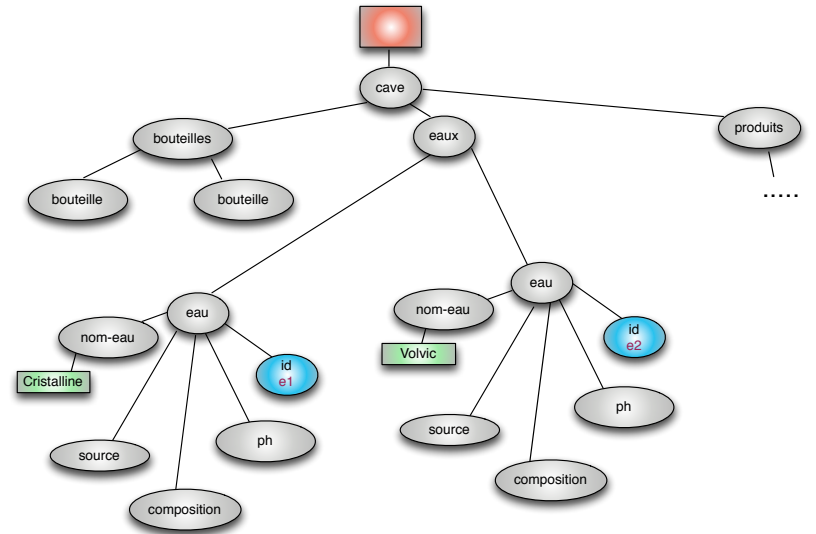
`/descendant::eau/child::nom_eau/child::text()`



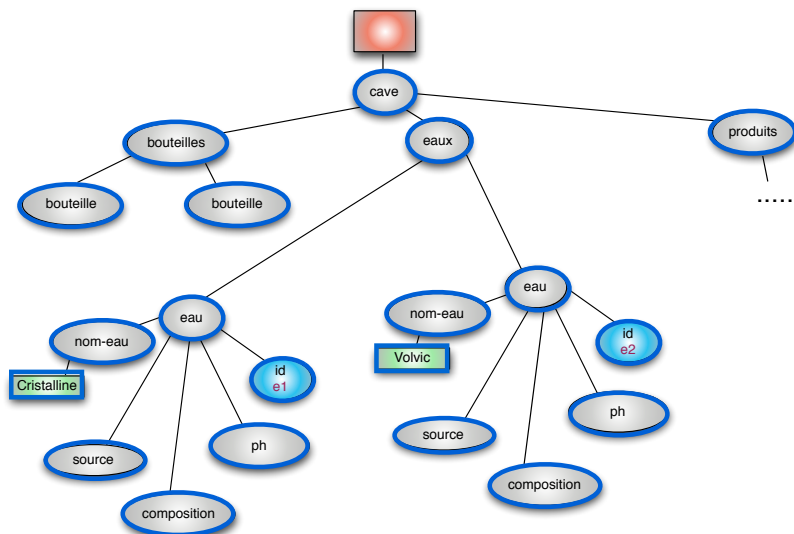
`/descendant::eau/child::nom_eau/child::text()`



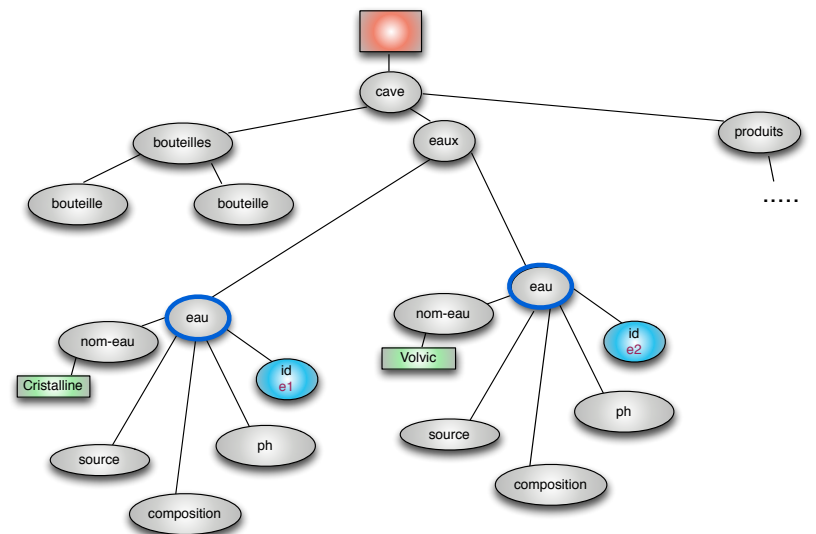
`/descendant::eau/child::nom_eau/child::text()`



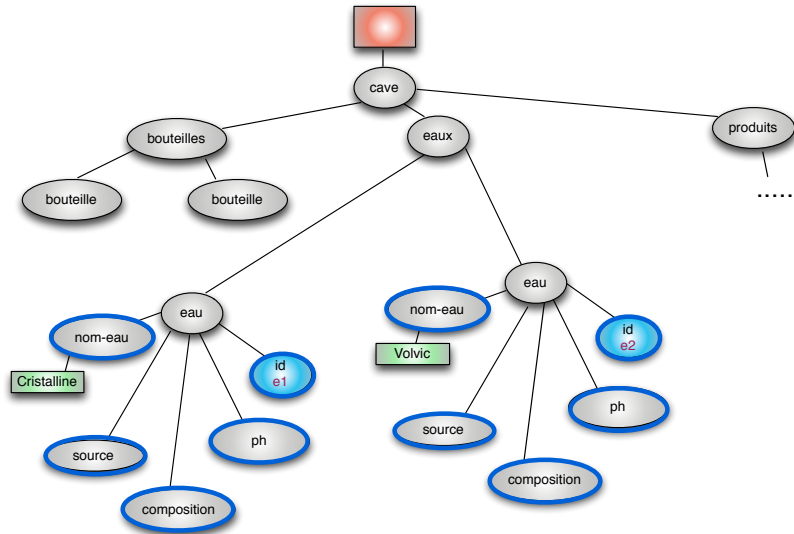
`/descendant::eau/child::nom_eau/child::text()`



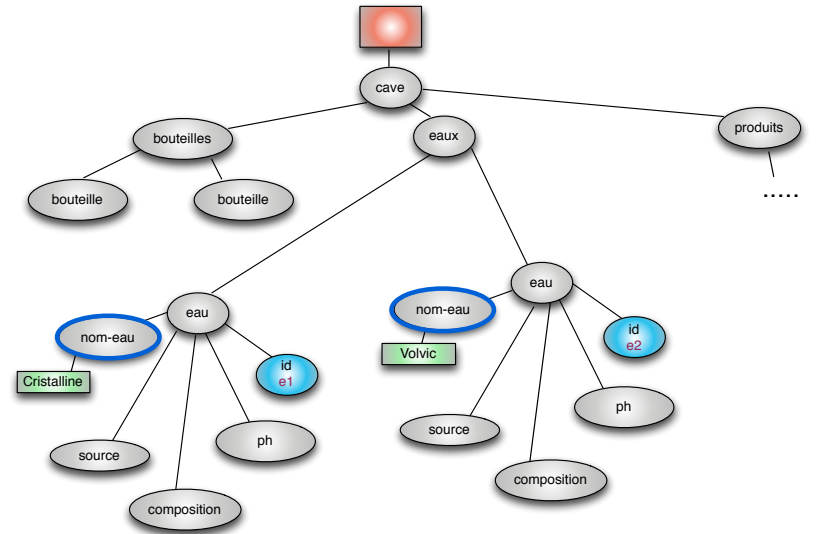
`/descendant::eau/child::nom_eau/child::text()`



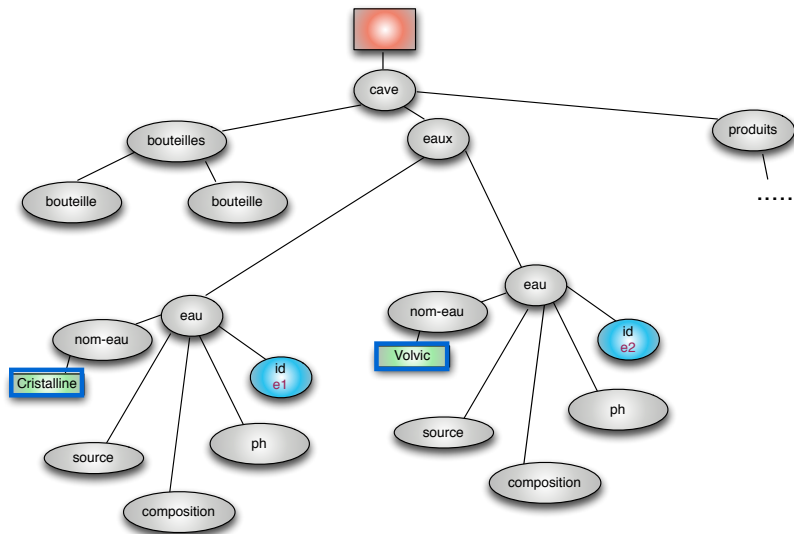
/descendant::eau/child::nom_eau/child::text()



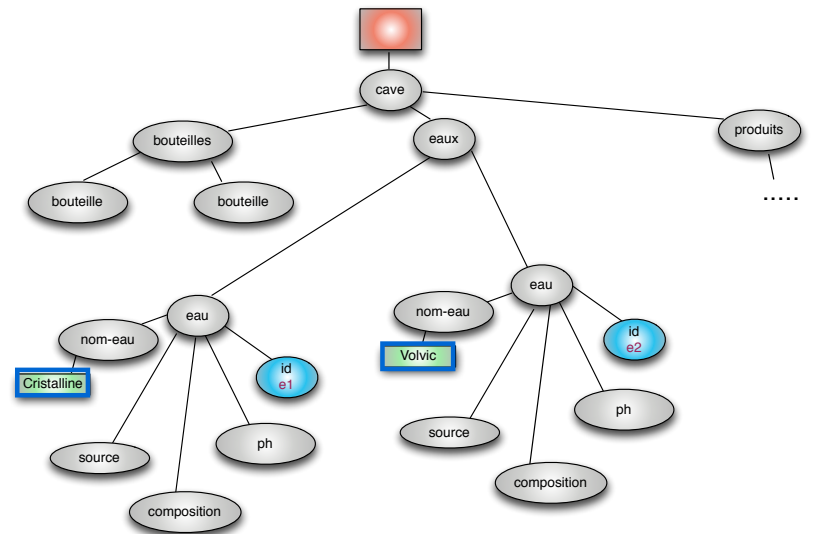
/descendant::eau/child::nom_eau/child::text()



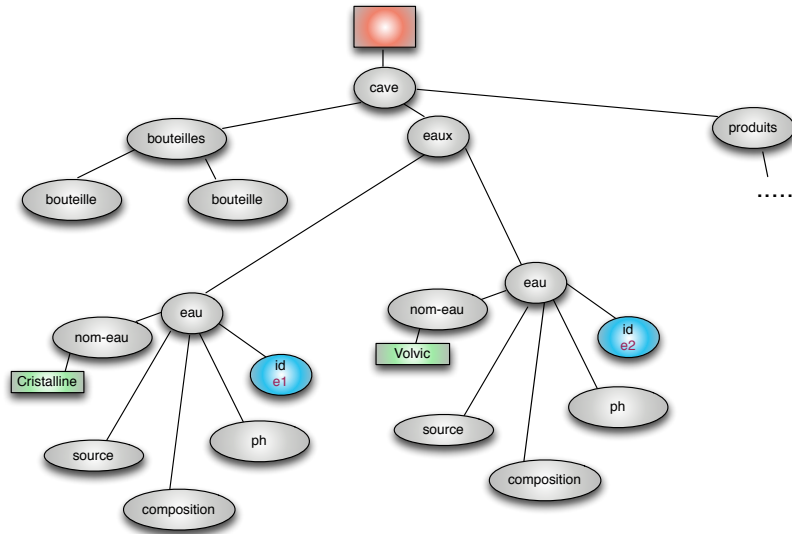
/descendant::eau/child::nom_eau/child::text()



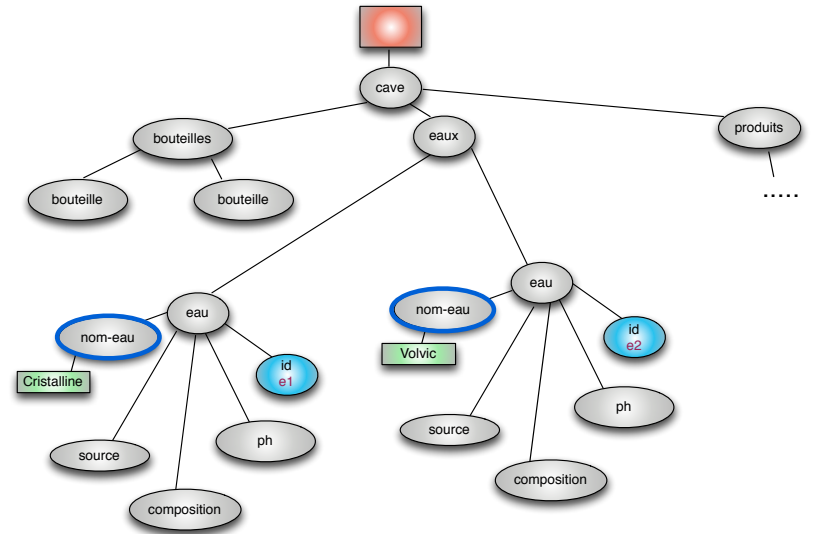
/descendant::eau/child::nom_eau/child::text()



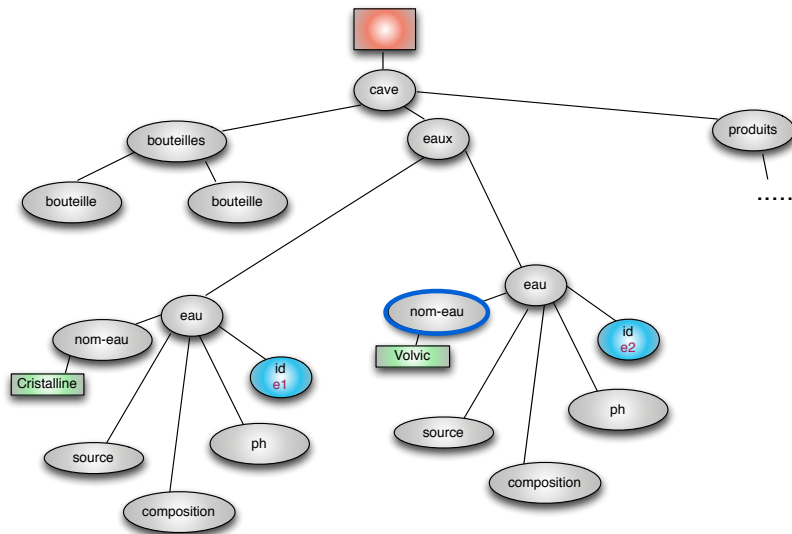
/descendant::nom_eau[child::text()='Volvic']/parent::* /attribute::id



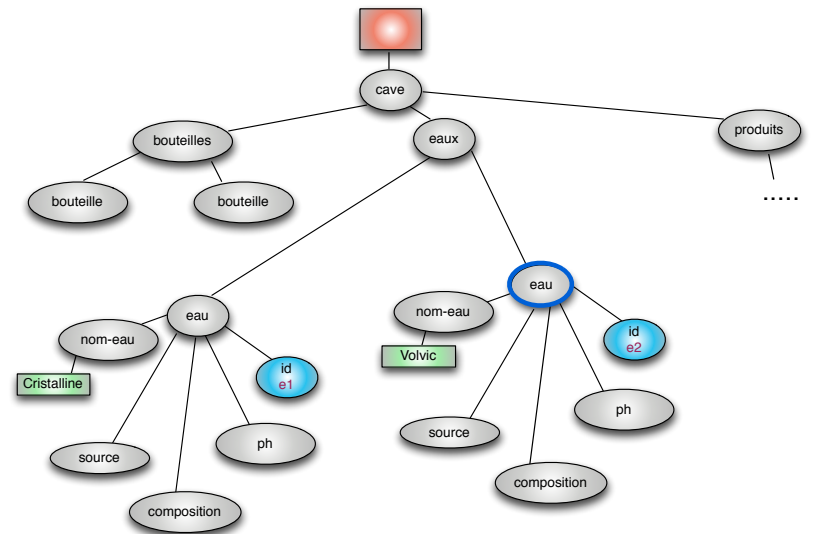
/descendant::nom_eau[child::text()='Volvic']/parent::* /attribute::id



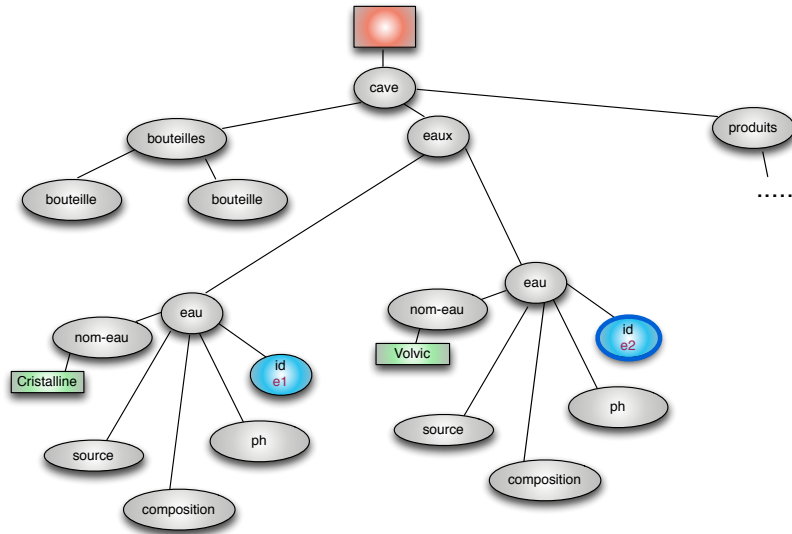
/descendant::nom_eau[child::text()='Volvic']/parent::* /attribute::id



/descendant::nom_eau[child::text()='Volvic']/parent::* /attribute::id



```
/descendant::nom_eau[child::text()='Volvic']/parent::*/@attribute::id
```



Les axes (axis)

- **child** sélectionne les nœuds éléments fils du nœud courant
- **descendant** les nœuds éléments descendants du nœud courant
- **parent** le nœud père du nœud courant
- **ancestor** les éléments ancêtres
- **following-sibling (preceding-sibling)** les éléments frères droits (les éléments frères gauches)
- **following (preceding)** les éléments dont la balise ouvrante (fermante) apparaît après (avant) dans le document
- **self** le nœud courant lui-même
- **ancestor-or-self, descendant-or-self**
- **attribute** les attributs de l'élément courant
- **namespace** les espaces de noms

Les filtres (node tests)

Un **filtre** permet de sélectionner par un nom ou par un type : un filtre peut être

- un **nom** d'élément ou d'attribut
- le caractère ***** qui sélectionne tous les objets de même nature (mais uniquement élément ou attribut).
 - **child::*** sélectionne tous les éléments fils du nœud courant,
 - **attribute::*** sélectionne tous les attributs du nœud courant.
- **comment()** qui sélectionne les nœuds de type commentaire
- **text()** qui sélectionne les nœuds de type texte
- **node()** qui sélectionne les nœuds de n'importe quel type.
- **processing-instruction()** ...

Exemples

- **/descendant::eau/child::nom_eau/child::text()** retourne l'ensemble des noms d'eaux minérale (donc séquence de chaînes de caractères)
- **/descendant::bouteilles/following-sibling::*** retourne les éléments frères à droite d'un élément `bouteilles`, donc la séquence formée des éléments `eaux` et `produits`.
- **/descendant::*** retourne la séquence de tous les éléments du document.

Les prédicats

Un **prédicat** est une expression booléenne qui peut être évaluée à vrai ou faux. On dispose des connecteurs logiques **and** et **or** et on peut faire intervenir dans les expressions des valeurs de type :

- **numérique**
- **chaîne de caractères**
- **booléens** (**true** et **false**)
- **ensemble de nœuds**

Des **règles de conversion** s'appliquent pour pouvoir convertir toute expression XPath en valeur booléenne en fonction de ce qu'elle retourne.

Valeurs numériques

- on dispose des **comparaisons** habituelles **<**, **>**, **!=**, **=**
- on dispose des **opérations** **+**, **-**, *****, **/**, **div**, **mod**
- la fonction **number(arg)** permet de tenter une **conversion** si la conversion échoue, on obtient la valeur **NaN**

```
/descendant::node()[number(attribute::code-barre) mod 2 = 1]
```

Règles de conversion en booléens

- un ensemble vide vaut **false** sinon **true**
- une chaîne vide vaut **false** sinon **true**
- **0** ou **NaN** valent **false**, les autres valeurs numériques valent **true**

`/descendant::*[child::taux]` l'ensemble des éléments qui ont un fils **taux**.

`/descendant::eau[attribute::id="e3"]` l'élément **eau** dont l'attribut **id** vaut **"e3"**

`/descendant::eau[attribute::id="e3" or attribute::id="e2"]`
l'ensemble des éléments **eau** dont l'attribut **id** vaut **"e3"** ou **"e2"**

Fonctions

Voir <http://xmlfr.org/w3c/TR/xpath/#corelib>

- **id("valeur")** retourne l'élément dont l'**identifiant** vaut **"valeur"**. On peut passer une séquence de valeurs en paramètre, le résultat est l'ensemble des éléments correspondants.
- **position()** retourne la position du nœud contexte dans l'**ensemble résultat courant**. La première position vaut **1**, la dernière vaut **last()**.
- **count(expression)** retourne le **nombre d'éléments** dans le résultat de l'évaluation de l'expression.
- **not(expression)** négation logique

On dispose aussi de fonctions de manipulation de chaînes (**contains**, **concat** ...) et de fonctions numériques : somme, valeur arrondie, ...

Exemples

`/descendant::taux[position()=2]` le deuxième élément `taux` dans l'ordre du document.

`/descendant::*/*/child::taux[position()=2]` l'ensemble des `taux` qui sont deuxièmes fils (c'est la liste des `taux` de magnésium).

Syntaxe abrégée

Syntaxe inspirée des systèmes de fichiers.

- `child::` peut être omis
- `attribute::` peut être remplacé par `@`
- `descendant-or-self::node()/child::` peut être remplacé par `//`
- `self::node()` peut être remplacé par un `.`
- `parent::node()` peut être remplacé par `..`
- `[position()=x]` peut s'écrire `[x]`

Exemples

- `//eau/..` sélectionne l'élément `eaux`.
- `//eau/@id` sélectionne tous les attributs `id` des éléments `eau`.
- `//eau/../../bouteilles/bouteille/@ref_eau` sélectionne tous les attributs `ref_eau` des éléments `bouteille`.
- `//bouteilles/bouteille[2]` sélectionne le deuxième fils `bouteille` de l'élément `bouteilles`.
- `//bouteille[@ref_eau="e1"]` sélectionne les `bouteilles` dont l'attribut `ref_eau` vaut "e1".
- `//bouteille[.//@ref_eau="e1"]` même résultat que la requête précédente.
- `//bouteille[//@ref_eau="e1"]` sélectionne tous les éléments `bouteille` car `//@ref_eau` est évalué à partir de la racine.

Pour terminer

On peut faire l'union des ensembles résultats de plusieurs expressions XPath à l'aide de l'opérateur `|`.

```
//eau/@id | //eau/../../bouteilles/bouteille/@ref_eau
```

sélectionne tous les attributs `id` des éléments `eau` et tous les attributs `ref_eau` des éléments `bouteille`.

Exercice : Utiliser cet opérateur d'union pour construire une expression XPath pouvant servir à tester si un nœud (le nœud contexte) est un attribut ou pas.