

# Fiche XML n°6

## Explorer un document XML avec XPATH

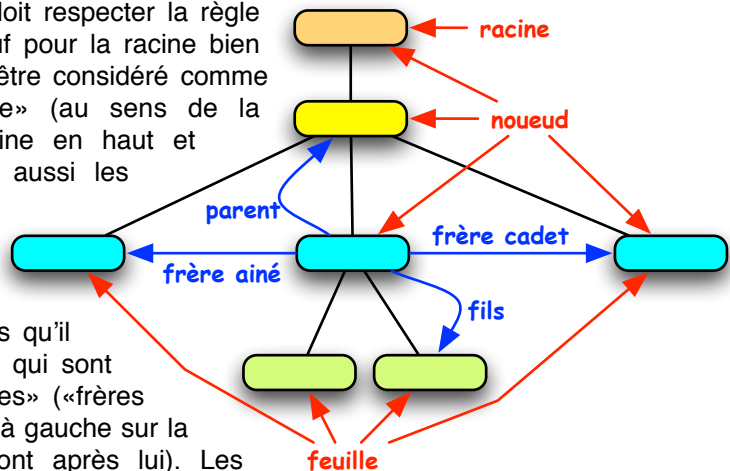
### Cours

#### 1. Introduction

XML est une syntaxe extrêmement pratique et qui possède beaucoup d'avantages. Cependant, non seulement les données manipulées peuvent être nombreuses, mais XML, par essence, augmente énormément le volume des informations avec les nombreuses balises qu'il ajoute : on dit qu'il est très verbeux. Même si un document XML est lisible, il n'est pas toujours facile d'exploiter les données, de faire des recherches ou même de référencer une partie des informations. Aussi, dès la sortie de XML, le W3C a proposé un langage permettant de faciliter ces tâches : XPath ou XPath 1.0 (<http://www.w3.org/TR/xpath>). Plus récemment, ce langage a été enrichi dans son expressivité et dans les fonctions qu'il propose : c'est XPath 2.0 (<http://www.w3.org/TR/xpath20/>).

#### 2. Notion d'arbre XML

Pour bien comprendre le fonctionnement de XPath, il faut d'abord bien comprendre la structure d'un document XML. Pour mémoire, un document XML ne possède qu'un seul élément racine. De plus, tout élément doit respecter la règle d'inclusion stricte dans un seul élément (sauf pour la racine bien évidemment). Un document XML peut donc être considéré comme une structure hiérarchique de type « arbre » (au sens de la représentation informatique, c'est-à-dire racine en haut et feuilles en bas). Tous les éléments, mais aussi les commentaires, les textes, etc., sont alors considérés comme des noeuds de l'arbre. Chaque noeud possède des relations avec ceux qui l'entourent. Le noeud dans lequel il se trouve est le noeud « parent ». Les noeuds qu'il contient sont des noeuds « fils ». Les noeuds qui sont dans le même parent sont des noeuds « frères » (« frères aînés » s'ils sont avant lui dans le document - à gauche sur la représentation - et « frères cadets » s'ils sont après lui). Les éléments contenus dans un élément sont, en effet, ordonnés (selon l'ordre d'apparition dans le document). Un élément vide, un commentaire, une instruction de traitement, un texte, etc. sont des noeuds « feuilles » (sans fils).



Attention, l'arbre « XML » associé à un document XML est dit « résolu » :

- les entités générales ont été remplacées par leur valeur ;
- en cas d'absence, et quand c'est possible, les attributs sont ajoutés avec leur valeur par défaut ;
- les sections CDATA sont intégrées comme des noeuds texte, etc.

Dans un arbre XML résolu, 7 types de noeuds peuvent être identifiés :

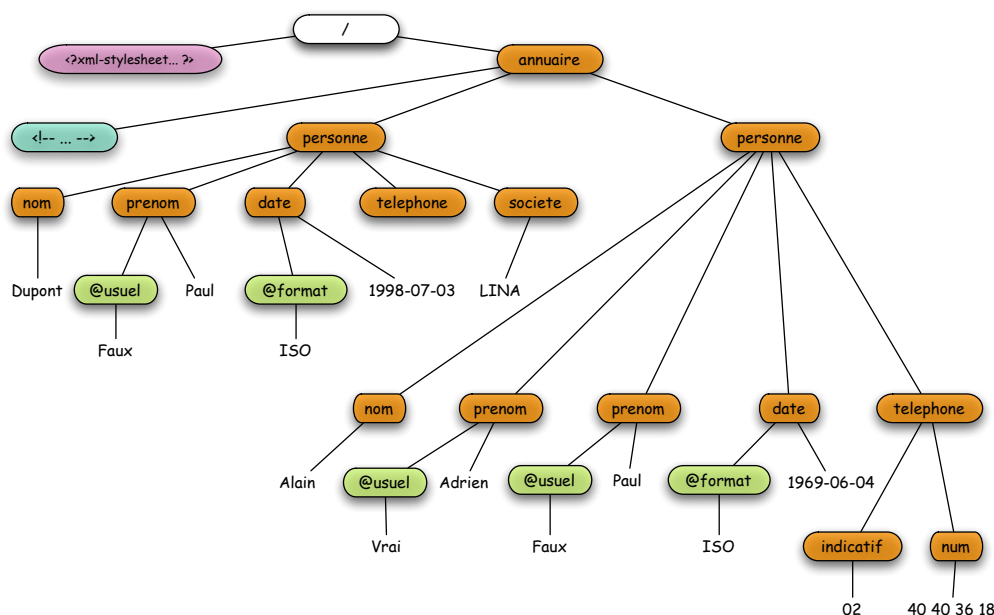
- le document (la racine),
- l'élément,
- le texte,
- l'attribut,
- le commentaire,
- l'instruction de traitement,
- l'espace de noms.

Prenons, par exemple, le document ci-dessous. Il décrit un annuaire d'entreprise simple. Cet exemple sera repris tout au long de cette fiche pour présenter les différentes notions.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ELEMENT annuaire (personne*)>
<ELEMENT personne (nom, prenom+, date, telephone,societe?)>
<ELEMENT nom (#PCDATA)>
<ELEMENT prenom (#PCDATA)>
<ATTLIST prenom usuel (Vrai|Faux) "Faux" >
<ELEMENT date (#PCDATA)>
<ATTLIST date format (ANSI | ISO | EN-exp | FR-exp) #REQUIRED>
<ELEMENT telephone (indicatif, num)? >
<ELEMENT indicatif (#PCDATA)>
<ELEMENT num (#PCDATA)>
<ELEMENT societe (#PCDATA) >
```

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE annuaire SYSTEM "annuaire.dtd" [<!ENTITY societe "LINA">]>
<?xml-stylesheet type="text/css" href="annuaire.css"?>
<annuaire>
  <!-- Liste des personnes -->
  <personne>
    <nom>Dupont</nom><prenom>Paul</prenom>
    <date format="ISO">1998-07-23</date>
    <telephone/><societe>&societe;</societe>
  </personne>
  <personne>
    <nom>Alain</nom>
    <prenom usuel="Vrai">Adrien</prenom><prenom>Paul</prenom>
    <date format="ISO">1969-06-04</date>
    <telephone>
      <indicatif>02</indicatif>
      <num>40 40 36 18</num>
    </telephone>
  </personne>
</annuaire>
```

Ce document engendre l'arbre XML résolu ci-dessous. Il est à noter la «perte» de la référence à la DTD.



### 3. XPath, un langage de requête XML

XPath est un langage non-XML (comme les DTD), qui permet d'identifier et localiser des parties d'un document XML, et de sélectionner des noeuds et des ensembles de noeuds (sous-arbres) dans une arborescence XML résolue. Son principe est assez simple : il décrit des chemins à prendre pour aller de la racine (ou d'un noeud «local») vers le noeud recherché en suivant les filiations.

Ce langage simple est utilisé de manière indépendante pour faire des recherches dans un document XML ou comme outil dans d'autres dialectes XML (XPointer, XLink, XSLT, XQuery, XForms, etc.).

### 4. Syntaxe

XPath est donc un chemin de localisations, c'est-à-dire une suite d'étapes de localisation séparées par un symbole «/». Le chemin est dit «absolu» s'il commence à la racine et «relatif» (au noeud contextuel) dans le cas contraire. Ce chemin désigne un ensemble, éventuellement vide, de noeuds de l'arbre, autrement dit, de structures XML, de valeurs numériques ou de valeurs textuelles.

Une étape de localisation est constituée d'un axe, d'un test de noeud et d'un prédicat selon la syntaxe suivante : «axe::test[prédicat]». L'«axe» est une relation structurale désignant le prochain ensemble de noeuds (direction à prendre dans l'arbre). Le «test» désigne un type de noeud (élément ou attribut). Le «prédicat», optionnel, permet de mettre en place des filtres sur les noeuds candidats obtenus par le «test» et l'«axe». Pour se donner une idée de ce qu'est XPath, voici deux exemples :

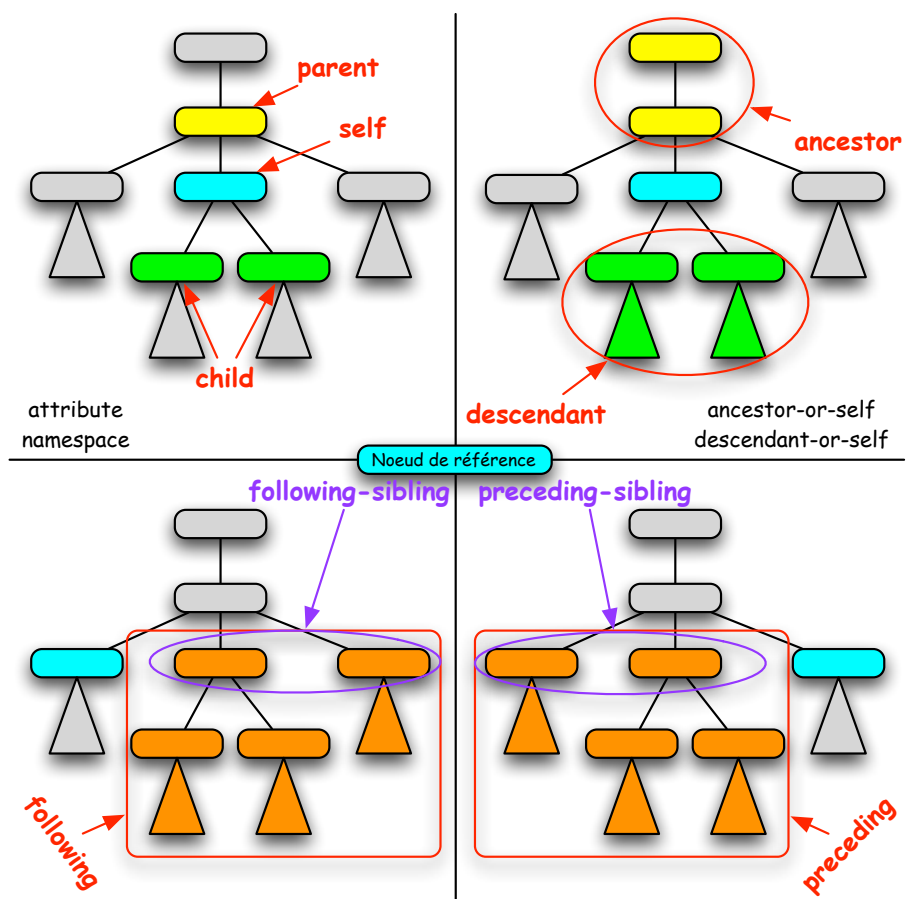
- `/child::annuaire/child::personne[child::nom/child::text()='Dupont']/child::prenom`  
ou  
`/annuaire/personne[nom='Dupont']/prenom` : sont deux requêtes identiques (et donc qui donnent le même résultat), sauf que la seconde utilise une syntaxe alternative plus simple mais moins expressive ; elles partent de la racine et prennent la direction du seul «fils» du document, la racine ; ensuite, elles se dirigent vers les fils pour prendre les éléments «personne» qui doivent avoir la particularité d'avoir un fils «nom» ayant un contenu «Dupont» ; enfin, elles se dirigent vers les fils «prenom» qui se trouvent sélectionnés ; le résultat de ces requêtes est donc l'élément : «<prenom usuel='Faux'>Paul</prenom>» ;
- `/annuaire/personne[nom='Dupont']/prenom/text()` : cette requête prend le même chemin que les précédentes, mais va un pas plus loin : elle ne prend en compte que le noeud texte compris dans «prenom» et non pas tout l'élément ; le résultat est donc le noeud texte : «Paul».

Attention, les exemples qui précèdent peuvent porter à croire que seul un chemin en pris. Ce n'est pas vrai. Lorsque plusieurs noeuds sont candidats, toutes les possibilités sont «explorées en parallèle». Le résultat est alors une liste de noeud.

#### Les axes

Un certain nombre de mots-clés permettent «de circuler dans l'arbre XML». La figure ci-contre les illustre. Soit un noeud de référence (en bleu) qui est un noeud sur lequel on est arrivé grâce aux étapes précédentes. Plusieurs axes sont possibles :

- `parent` : remonte dans le noeud «parent» ;
- `child` : descend explorer les fils ;
- `self` : reste sur le noeud courant ;



- `attribute` : part explorer les attributs du noeud ;
- `following` : explore les noeuds des sous-arbres dont les frères cadets sont les racines ;
- `following-sibling` : explore les frères cadets (mais pas leurs fils) ;
- `ancestor-or-self` : explore tous les noeuds entre le noeud racine et le noeud de référence (ce dernier compris) ;
- etc.

## Les tests de noeud

Une fois la direction indiquée, il faut spécifier le type de noeud à prendre en compte : élément, texte, commentaire, instruction ou espace de noms. Pour les éléments, il est aussi possible d'être plus précis en donnant le nom de l'élément attendu. Il en est de même pour les attributs selon l'axe choisi. Ainsi, le test peut prendre une des formes suivantes :

- `nom` : élément ou attribut XML (selon l'axe choisi) nommé «nom» ;
- `*` : tous les éléments ou attributs XML possibles ;
- `text()` : les noeuds texte ;
- `comment()` : les commentaires ;
- `processing-instruction()` : les instructions de traitement ;
- `processing-instruction(ip)` : les instructions de traitement dont la cible est «ip» ;
- `node()` : tous les noeuds possibles de l'arbre XML (éléments, CDATA, attributs, commentaires...).

Pour illustrer, voici quelques exemples de couples «axe::test», en supposant qu'un noeud «n» est sélectionné :

- `child::personne` : tous les éléments XML fils de «n» nommés «personne» ;
- `child::*` : tous les éléments XML fils de «n» ;
- `child::text()` : tous les noeuds fils texte de «n» ;
- `child::node()` : tous les noeuds fils de «n», quelque soit leur type ;
- `attribute::*` : tous les attributs du noeud «n» ;
- `descendant::personne` : tous les noeuds «personne» présents dans la descendance du noeud «n» ;
- `following-sibling::personne` : les éléments «personne» fils des frères cadets du noeud «n» (même père et après dans le document XML !).

## Syntaxe abrégée

L'écriture de chemins XPath telle qu'ils sont présentés ci-dessus est souvent lourde et peu lisible. Aussi, pour faciliter l'utilisation de XPath, une écriture simplifiée est proposée. L'objectif est de simplifier l'écriture d'un chemin en prenant une syntaxe assez familière (en tout cas pour les informaticiens). En effet, elle se rapproche des commandes système de parcours des structures de fichiers et de répertoires.

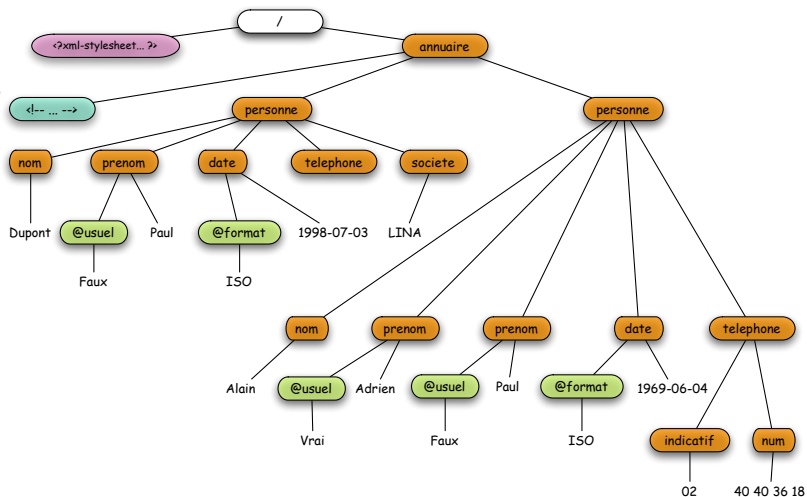
Les simplifications sont les suivantes :

- «`.`» = «`self::node()`» ;
- «`..`» = «`parent::node()`» ;
- «`nom`» = «`child::nom`» ;
- «`@attr`» = «`attribute::attr`» ;
- «`//`» = «`/descendant-or-self::node()`».

Voici deux exemples d'équivalence d'étapes de localisation utilisant la syntaxe abrégée :

- «`div//para`» = «`child::div/descendant-or-self::node()/child::para`» ;
- «`../title`» = «`parent::node()/child::title`».

Les simplifications sont peu nombreuses et ne couvrent pas tous les cas possibles. Ainsi, certaines recherches ne peuvent pas être complètement simplifiées. Cependant, il est autorisé de mélanger les syntaxes et d'utiliser la syntaxe étendue lorsque la syntaxe abrégée ne permet pas de décrire correctement l'étape de localisation.



Par exemples, voici quelques requêtes dont une ne peut pas se simplifier totalement :

- `/annuaire/personne/following-sibling::personne` : cette requête XPath recherche toutes les personnes, sauf celle qui est en premier ; ici, le résultat sera la seconde personne ;
- `/annuaire/personne/prenom` : cette requête recense tous les prénoms de l'annuaire ; le résultat sera «<prenom usuel='Faux'>Paul</prenom>», «<prenom usuel='Vrai'>Adrien</prenom>» et «<prenom usuel='Faux'>Paul</prenom>» ;
- `//telephone/indicatif` : cette dernière requête recense les indicatifs de téléphone présents à tout niveaux ; ici, le résultat sera «<indicatif>02</indicatif>».

## Les prédicats

Sélectionner des noeuds de l'arbre en fonction de leur type, de leur position, voire de leur nom, ne suffit pas. Parfois, des critères sur des positions ou des valeurs sont importants. Aussi, XPath propose un mécanisme de filtre sur les noeuds sélectionnés après l'axe et le test. Ainsi, seuls les noeuds référencés par l'expression avant application du prédicat et vérifiant le prédicat sont conservés.

Le prédicat peut prendre différentes formes, mais c'est toujours une expression dépendante du noeud courant (celui sélectionné par le type et la direction choisie). Parmi les prédicats possibles, on trouve :

- une référence à un nom d'élément ou d'attribut (en chemin relatif) : c'est un test d'existence ;
- une valeur entière : c'est la position du noeud dans la fratrie ;
- un appel de fonction ;
- une conjonction («and») ou disjonction («or») d'expressions logiques (comparaisons, tests, etc.) ou de prédicats ;
- chemin ou union de chemins (opérateur "|") : là encore, c'est un test d'existence.

Pour illustrer ces prédicats, voici quelques exemples :

- «`/annuaire/personne[2]/nom`» : cette requête sélectionne le nom de la seconde personne de l'annuaire ; elle est équivalente à la requête suivante «`/annuaire/personne[position()=2]/nom`» ;
- «`/annuaire/personne[position()=last()]/nom`» : elle désigne le nom de la dernière personne de l'annuaire dans le document XML ;
- «`//*[num | indicatif]`» : elle retourne tous les éléments contenant soit un élément «num» soit un élément «indicatif» du document ; elle peut aussi s'écrire «`//*[./node()=num or ./node()=indicatif]`» ;
- «`//telephone[indicatif = '02']`» : ce chemin XPath recense les téléphones dont l'indicatif est en 02 ; elle est équivalente à «`//telephone[string(indicatif)='02']`» ou encore à la requête «`//telephone[indicatif/text()='02']`» ;
- «`/annuaire/personne[prenom/@usuel]`» : elle désigne seulement les personnes qui ont un prénom avec un attribut «usuel» (quelque soit sa valeur), sauf s'il existe une DTD qui propose une valeur par défaut auquel cas, toutes les personnes seraient systématiquement sélectionnées !

Le premier exemple ci-dessus illustre une des utilisations de XPath : désigner un élément ou un attribut d'un document XML. Cette notion de liens utilisant XPath est reprise dans les standard XLink<sup>1</sup> et XPointer<sup>2</sup>.

## 5. Types et fonctions XPath

Dans le cadre des prédicats, le standard XPath propose des fonctions permettant d'effectuer des recherches assez précises. Ces fonctions dépendent de certains types. En XPath, les structures sont typées de manière implicites. Il existe cinq types de «valeurs» :

- «boolean» : c'est un booléen ;
- «number» : c'est un nombre ;
- «node-set» : c'est un ensemble de noeuds de l'arbre ;
- «string» : c'est un chaîne de caractères ;
- «object» : type générique.

Certaines fonctions ou opérateurs étant spécifiques pour certains types, il existe des fonctions pour convertir une «valeur» d'un type vers un autres (fonctions de transtypage). Par exemple, il existe la fonction : «*string*(*object*)» qui convertit une valeur quelconque en chaîne de caractères ; par exemple, en l'appliquant sur un élément XML, elle retourne une chaîne de caractère qui est la mise bout à bout de tous les noeuds

<sup>1</sup> <http://www.w3.org/XML/Linking>

<sup>2</sup> <http://www.w3.org/TR/xptr/>

texte du sous-arbre, ainsi «string('<a><b>toto</b><b>tutu</b></a>')» retournera «tototutu». D'autres, de même principe existent pour les différents types. Les fonctions sont décrites par le standard : <http://www.w3.org/TR/xpath#corelib>.

Dans cet ensemble de fonctions standard, il est possible de trouver, par exemple, les fonctions suivantes :

- pour le type «node-set» :
  - «number last()» : déjà rencontrée, elle indique le rang du dernier fils d'une fratrie,
  - «number position()» : elle indique le rang du noeud dans la fratrie,
  - «number count(node-set)» : elle donne le nombre d'éléments dans l'ensemble de noeud (ensemble qui peut être construit par une requête XPath absolue ou relative),
  - «node-set id(object)» : fonction très utile qui retourne le noeud qui possède un attribut de type ID ayant la valeur passée en paramètre, si plusieurs valeurs sont en paramètre alors elle retourne la liste des éléments concernés,
  - «string local-name(node-set?)», «string name(node-set?)», «string namespace-uri(node-set?)» : pour la gestion des espaces de noms ;
- chaînes de caractères :
  - «string string(object?)» : déjà vue, elle retourne tous les noeuds texte du sous-arbre,
  - «string concat(string, string, string\*)» : met bout à bout toutes les chaînes de caractères,
  - «string substring-before(string, string)», «string substring-after(string, string)», «string substring(string, number, number)» : extrait une partie de la chaîne de caractères,
  - «string normalize-space(string)» : supprime les espaces en début et en fin et remplace les successions d'espaces par un seul espace,
  - «number string-length(string)» : indique la longueur de la chaîne de caractères,
  - «boolean starts-with(string, string)», «boolean ends-with(string, string)» : teste le début (resp. la fin) d'une chaîne de caractères,
  - «boolean contains(string, string)» : teste le contenu d'une chaîne de caractères...
- booléens :
  - «boolean not(object)», «boolean true()», «boolean false()» : fonctions permettant de mettre en place des expressions logiques,
  - «boolean lang(string)» : teste la langue de l'élément (attribut «xml:lang»),
  - «boolean boolean(object?)» ;
- nombres :
  - «number number(object?)»,
  - «number sum(node-set)», «number floor(number)», «number ceiling(number)»,
  - «number round(number)»...

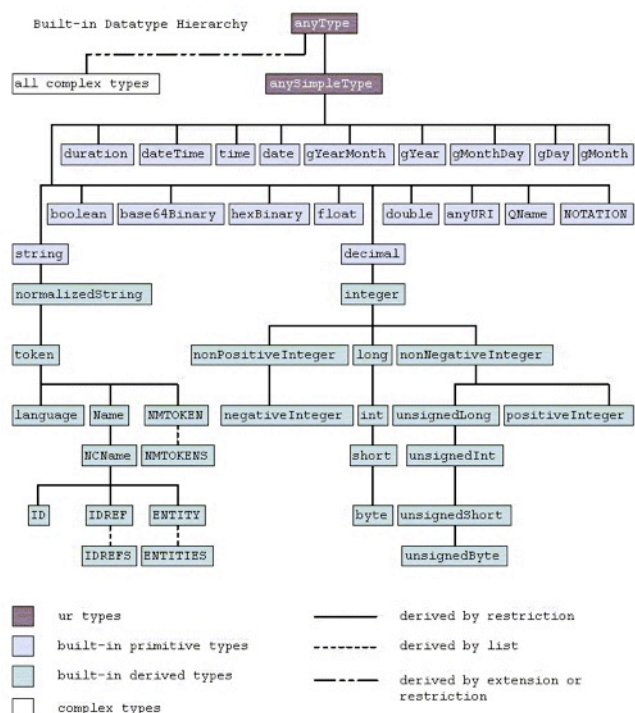
Remarque : certains paramètres sont optionnels (suivis d'un "?"). La fonction s'applique alors sur le noeud courant.

## 6. XPath 2.0

En 2007, conjointement avec XQuery et XSLT, une nouvelle version de XPath est publiée par le W3C : XPath 2.0. Cette nouvelle version apporte quelques améliorations, en particulier en ce qui concerne les types de valeurs, les opérateurs et les fonctions.

Aux types déjà présents dans XPath 1.0 sont ajoutés les types de base définis pour XML Schema<sup>3</sup>. On y retrouve les types classiques en programmation mais aussi des types tels que xs:date ou xs:anyURI...

Une autre nouveauté est l'apparition du type «sequence» qui reprend la notion de «node-set».



<sup>3</sup> <http://www.w3.org/TR/xmlschema-0/#CreatDt>

Voici quelques exemples :

- (a, b, c, d) : une séquence de 4 éléments ;
- (10 to 50) : un constructeur de séquence générant une séquence d'entiers de 10 à 50 ;
- () : la séquence vide, sans contenu.

Dans XPath 2.0, le résultat de toute expression, tout chemin est une séquence. Une étape de localisation a pour résultat une séquence de valeurs (éventuellement des noeuds) sur laquelle s'appliquera aussi bien un filtre qu'une autre étape de localisation (sur chacune des valeurs de la séquence). Ainsi, il est possible de faire : «(30 to 50)[. mod 5 = 0]» qui retourne la séquence «(30, 35, 40, 45, 50)». A noter qu'avec XPath 2.0, une valeur seule ou une séquence avec seulement la valeur est identique. Ainsi «2 = (2)».

De même, l'expression «/annuaire/personne[nom='Dupont']/prenom» se calculera de la manière suivante :

1. le document ;
2. «(/)/annuaire» donne (annuaire[1]) ;
3. «(/annuaire[1])/personne» donne (/annuaire[1]/personne[1], /annuaire[1]/personne[2]) ;
4. «(/annuaire[1]/personne[1], /annuaire[1]/personne[2])[nom='Dupont']» donne (/annuaire[1]/personne[2]) ;
5. «(/annuaire[1]/personne[2])/prenom» donne (/annuaire[1]/personne[2]/prenom[1]) .

De nouvelles fonctions sont associées aux séquences : distinct-values, remove, reverse, subsequence, avg, min, max, sum... L'ensemble des fonctions, parfois préfixées par «fn:», sont présentées dans le standard : <http://www.w3.org/2005/04/xpath-functions/> . Par exemple, il est aussi possible d'appuyer la recherche sur des documents complémentaires en utilisant «doc(URI)». Il existe aussi une nouvelle fonction nommée «idref», symétrique de la fonction «id» déjà présente en XPath 1.0. «idref», lorsque le paramètre est un identifiant XML, retourne une séquence d'attributs de type IDREF ou IDREFS qui contiennent cet identifiant.

Des opérateurs complexes apparaissent aussi. Ils s'appuient sur la notion de variable. Une variable en XPath s'écrit «\$n» où «n» est son nom. Les nouveaux opérateurs sont les suivants :

- l'opérateur d'itération «for \$name in exp1 return exp2» : la variable «\$name» prend successivement toutes les valeurs de la séquence produite par l'expression XPath «exp1» et est utilisée, pour chaque valeur, dans «exp2» ; il en résulte une séquence composée des différentes applications de «exp2» sur chacune des valeurs de «exp1» ;
- l'opérateur conditionnel «if (exp0) then exp1 else exp2» : si «exp0» retourne un booléen «True» (ou une valeur d'un autre type dont la conversion de type permet d'obtenir «True») alors l'expression «exp1» est évaluée ; dans le cas contraire, c'est «exp2» qui est évaluée ;
- les quantificateurs :
  - il existe... «some \$name in exp1 satisfies exp2» : l'expression retourne «True» si «exp2» est vérifiée pour au moins une des valeurs de «exp1» («exp2» utilisant la variable «\$name»)⁴,
  - pour tout... «every \$name in exp1 satisfies exp2» : l'expression retourne «True» si «exp2» est vérifiée pour chacune des valeurs de «exp1» («exp2» utilisant la variable «\$name»)⁵.

Dans cette nouvelle version de XPath, il faut être attentif au fonctionnement des opérateurs de comparaison. L'utilisation de ceux-ci dépend des valeurs produites par les opérandes.

Tout d'abord, si les opérandes retournent une seule valeur (ou une séquence à une valeur), elles sont appelées «atomiques». Les opérateurs adaptés sont alors :

- «eq» pour l'égalité ;
- «ne» pour la différence ;
- «le» et «lt» pour «inférieur ou égal» et «supérieur» ;
- «ge» et «gt» pour «supérieur ou égal» et «supérieur».

Ainsi, les expressions suivantes sont valides :

- «8 eq 4+4» ;
- «(//date)[1]/@format eq 'ISO'».

Si un des deux opérandes est une séquence d'au moins deux valeurs, la comparaison avec les opérateurs ci-dessus sera systématiquement fautive. En revanche, ce type d'opérande sera valide pour les opérateurs

⁴ Cette expression est équivalente à l'expression «exists( for \$name in exp1 return if (exp2) then true() else () )».

⁵ Cette expression est équivalente à l'expression «empty( for \$name in exp1 return if (exp2) then () else false() )».

«=», «!=», «<», «>», etc. Mais attention à leur fonctionnement ! En effet, ils sont «vrai» si au moins un couple de valeurs prises dans chacun des opérandes vérifie la comparaison.

Ainsi, les expressions suivantes sont valides :

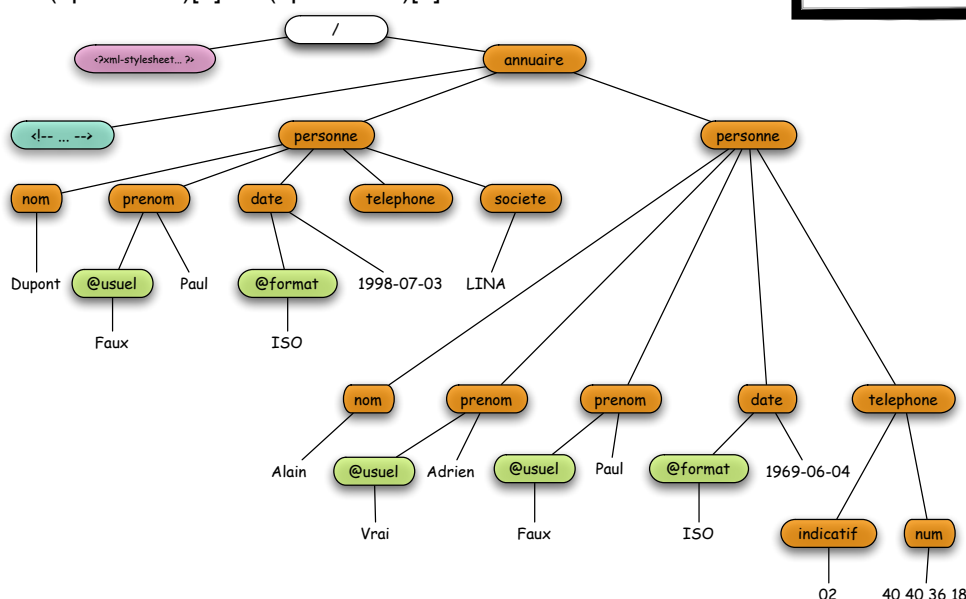
- «8 = 4+4», car c'est aussi «(8) = (4+4)» ;
- «(1, 2) = (2, 4)», car il y a un 2 dans chacune des séquences ;
- «//personne/nom/text() = 'Albert'».

Il existe aussi des comparateurs sur les noeuds de l'arbre XML. «is» détermine si deux résultats référencent le même noeud. Ainsi, l'expression suivante est vérifiée :  
«(//personne)[2] is //personne[nom eq 'Dupont']».

Dans le même esprit, XPath 2.0 propose deux opérateurs d'ordre «<<» et «>>». Ainsi, l'expression suivante est vérifiée :  
«(//personne)[4] >> (//personne)[3]».

XPath ne respecte pas les propriétés «intuitives» et mathématiques des opérateurs de comparaison habituels :

- Réflexivité -  $()=()$  n'est pas vérifiée, car les deux séquences n'ont pas d'éléments identiques ;
- Transitivité -  $(1,2)=(2,3)$ ,  $(2,3)=(3,4)$ , mais on a pas  $(1,2)=(3,4)$  ;
- Anti-symétrie -  $(1,4)<=(2,3)$ ,  $(2,3)<=(1,4)$  mais on a pas  $(1,2)=(3,4)$  ;
- Négation -  $(1)!=()$  donne «false»  $(1)=()$  donne «false»



L'utilisation des opérateurs de comparaison n'est pas très naturelle. Il convient donc d'y être attentif et d'utiliser les opérateurs adaptés. Par exemple :

- «((//personne)[1]/prenom, (//personne)[1]/date) eq ((//personne)[2]/prenom[2], (//personne)[2]/date))» n'est vérifiée, car les arguments ne sont pas des singletons ;
- «((//personne)[1]/prenom, (//personne)[1]/date) = ((//personne)[2]/prenom[2], (//personne)[2]/date))» est vérifiée, car les deux prénoms sont les mêmes ;
- «((//personne)[1]/prenom, (//personne)[1]/date) is ((//personne)[2]/prenom[2], (//personne)[2]/date))» provoque une «runtime error», car les arguments ne sont pas des singletons.

Les opérateurs logiques sont habituels «and» et «or». Attention au changement de type implicite. Les arguments valent «false» si la valeur est : le booléen «false()», la séquence vide, la chaîne vide ou l'entier zéro.

Les valeurs constantes sont obtenues par les fonctions «true()» et «false()». Attention, la négation n'est pas un opérateur, mais une fonction «not(...)» !

Remarque importante : la plupart des outils de manipulation XML ont des fonctions comprenant XPath 1.0. En revanche, il y a actuellement encore peu d'outils qui gèrent XPath 2.0.

| Point                    | 2.0    | 1.0     |
|--------------------------|--------|---------|
| Implémentations courante | -      | ++      |
| Librairie de fonctions   | +++    | +       |
| Transtypage implicite    | +      | +++     |
| “4” < “4.0”              | «true» | «false» |

# Exercices

Soit la DTD «`edition.dtd`» et le document «`jailu.xml`»

```
<!ELEMENT edition (auteur+, livre+)>
<!ATTLIST edition nom CDATA #REQUIRED>
```

```
<!ELEMENT auteur EMPTY>
<!ATTLIST auteur
  nom CDATA #REQUIRED
  email CDATA #IMPLIED
  no ID #REQUIRED
  pays CDATA #IMPLIED>
```

```
<!ELEMENT livre (titre, ref-auteur)>
<!ATTLIST livre
  reference ID #REQUIRED
  annee CDATA #REQUIRED>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT ref-auteur EMPTY>
<!ATTLIST ref-auteur ref IDREF #REQUIRED>
```

## Exercice 1

**Question 1.** Construire l'arbre XML résolu pour le fichier ci-contre.

**Question 2.** Donner l'expression XPath 1.0 et XPath 2.0 référençant :

1. les auteurs (proposez toutes les variantes possibles);
2. les auteurs français ;
3. le deuxième auteur ;
4. les titres de livre qui contiennent la chaîne 'robot' ;
5. les ouvrages ayant été publiés en 1996 ;
6. les auteurs ayant publiés en 1996 ;
7. le titre des ouvrages d'Isaac Asimov.

Remarque : pour chacune de ces expressions, vous donnerez aussi le résultat retourné.

## Exercice 2.

«Calculer» le résultat des expressions XPath suivantes (indiquer si elles sont ou pas valables en XPath 1.0 et en XPath 2.0) sur le document ci-contre :

- «`distinct-values(/edition/auteur/@pays)`»
- «`doc("jailu.xml")/edition/auteur`»
- «`id(
 /edition/livre[
 every $n in /edition/livre/@annee
 satisfies $n le @annee]
 /ref-auteur/@ref
)/@nom`»
- «`id(
 /edition/livre[
 number(@annee) ge
 max(/edition/livre/@annee)
 ]/ref-auteur/@ref
)/@nom`»

## Exercice 3.

**Question.** Étant donné un paramètre "\$nom", donner l'expression XPath permettant de désigner tous les éléments frères du noeud courant de nom "\$nom".

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE edition SYSTEM "edition.dtd" [
  <ENTITY fr "France">
  <ENTITY usa "États Unis">
  <ENTITY rs "Russie">
]>

<edition nom="J'ai lu">
  <!-- Les auteurs -->
  <auteur no="a01" nom="Éric Holder" pays="&fr;" />
  <auteur no="a02" nom="Winston Groom" pays="&usa;" />
  <auteur no="a03" nom="Stephen King" pays="&usa;" />
  <auteur no="a04" nom="Isaac Asimov" pays="&rs;" />
  <!-- Les ouvrages -->
  <livre annee="2002" reference="I7301">
    <titre>Hongroise</titre><ref-auteur ref="a01"/>
  </livre>
  <livre annee="1986" reference="I3816">
    <titre>Forest Gump</titre><ref-auteur ref="a02"/>
  </livre>
  <livre annee="1996" reference="I5157">
    <titre>La ligne verte</titre><ref-auteur ref="a03"/>
  </livre>
  <livre annee="1983" reference="I2266">
    <titre>Simetierre</titre><ref-auteur ref="a03"/>
  </livre>
  <livre annee="1964" reference="I542">
    <titre>Un défilé de robots</titre><ref-auteur ref="a04"/>
  </livre>
  <livre annee="1950" reference="I453">
    <titre>I, robot</titre><ref-auteur ref="a04"/>
  </livre>
  <!-- ... -->
</edition>
```

## Exercice 4.

Un élément est marqué lorsqu'il possède un attribut de nom "marque" et dont la valeur est "oui". Autrement, il est considéré comme non marqué (pas d'attribut "marque" ou un attribut "marque" ayant la valeur "non").

**Question.** Donner l'expression XPath permettant de désigner les noeuds éléments marqués dont aucun des ancêtres n'est marqué.

## Exercice 5.

Soit la DTD "edt.dtd" décrivant une classe de documents représentant des emplois du temps d'étudiants dont "edt1213.xml" est un exemple. Ces deux documents sont présentés en fin d'exercice.

**Question 1.** Donner le résultat de la requête XPath suivante sur ce document. Expliquez ce qu'elle recherche.

```
/edt/liste_profs/prof[
  @idprof=/edt/liste_semaines/semaine/jour/creneau/@nprof
]/concat(prenom, ' ', nom)
```

**Question 2.** Parmi les requêtes suivantes, indiquez, en justifiant votre choix, lesquelles donnent le même résultat que la requête de la question précédente ? En cas de réponse négative, donnez les raisons.

- /edt/liste\_semaines/semaine/jour/creneau/@nprof/concat(prenom, ' ', nom)
- id(/edt/liste\_semaines/semaine/jour/creneau/@nprof)/concat(prenom, ' ', nom)
- /edt/liste\_profs/prof[
 @idprof = ../../liste\_semaines/semaine/jour/creneau/@nprof
 ]/concat(prenom, ' ', nom)
- /edt/liste\_profs/prof[
 @idprof = ../../../../liste\_semaines/semaine/jour/creneau/@nprof
 ]/concat(prenom, ' ', nom)
- concat(id(/edt/liste\_semaines/semaine/jour/creneau/@nprof)/prenom, ' ', id(/edt/liste\_semaines/semaine/jour/creneau/@nprof)/nom)
- /edt/liste\_profs/prof[
 some \$c in /edt/liste\_semaines/semaine/jour/creneau
 satisfies \$c/@nprof eq @idprof
 ]/concat(prenom, ' ', nom)
- /edt/liste\_profs/prof[
 every \$c in /edt/liste\_semaines/semaine/jour/creneau
 satisfies \$c/@nprof eq @idprof
 ]/concat(prenom, ' ', nom)
- /edt/liste\_profs/prof[id(@idprof)]/concat(prenom, ' ', nom)
- /edt/liste\_profs/prof[exists(idref(@idprof))]/concat(prenom, ' ', nom)

**Question 3.** Écrire les requêtes XPath (sans utiliser "/"") permettant d'obtenir les informations suivantes :

- Les jours (date) où il y a un enseignement en semaine 12 ;  
 lu 19/03/2007  
 me 21/03/2007  
 je 22/03/2007  
 ve 23/03/2007
- Les plages horaires (heure de début) qui sont occupées le mardi 27 mars ;  
 /edt[1]/types\_creneaux[1]/plage[4]/hdeb[1] - 14:00:00  
 /edt[1]/types\_creneaux[1]/plage[5]/hdeb[1] - 15:30:00  
 /edt[1]/types\_creneaux[1]/plage[6]/hdeb[1] - 17:00:00
- La journée (sa date) la plus chargée de l'emploi du temps ;  
 28/03
- Les jours (date) où tous les enseignements sont sur Projet XML ;  
 27/03
- Le nombre moyen de créneaux par jour sur toutes les semaines ;  
 3.25

- Même question si on compte seulement les jours d'enseignement ;  
2.6
- Les professeurs (prénom, nom et téléphone) qui n'enseignent pas le mercredi de la semaine 12 ;  
Jean-Pierre Fabriol : 0252125487

Pour chaque requête, les résultats présentés (sous la requête) sont ceux obtenus avec oXygen 8.1 et XPath 2.0. Les chemins ne font pas partie de la réponse, oXygen les places pour localiser la réponse. Seule l'information après le '-' est valable.

### *edt.dtd*

```
<?xml version="1.0" encoding="UTF-8"?>
<ELEMENT edt (liste_salles, liste_profs, types_creneaux, liste_semaines) >
<ATTLIST edt groupe CDATA #IMPLIED année CDATA #REQUIRED >

<!-- description des salles -->
<ELEMENT liste_salles (salle*) >
<ELEMENT salle EMPTY > <!-- représente une salle -->
<ATTLIST salle idsalle ID #REQUIRED
           capacite CDATA #REQUIRED
           type_salle (amphi | TD | TP | autre) #REQUIRED>

<!-- description des profs -->
<ELEMENT liste_profs (prof*) >
<ELEMENT prof (nom, prenom, dpt, tel) >
<ATTLIST prof idprof ID #REQUIRED >
<ELEMENT nom (#PCDATA) >
<ELEMENT prenom (#PCDATA) >
<ELEMENT dpt (#PCDATA) > <!-- département d'enseignement -->
<ELEMENT tel (#PCDATA) > <!-- chaîne de 10 chiffres -->

<!-- description des types de créneaux -->
<ELEMENT types_creneaux (plage*) >
<ATTLIST plage ordre ID #REQUIRED > <!-- numéro d'ordre de la plage dans la journée -->
<ELEMENT plage (hdeb, hfin) >
<ELEMENT hdeb (#PCDATA) > <!-- horaire de début de plage -->
<ELEMENT hfin (#PCDATA) > <!-- horaire de fin de plage -->

<!-- description des semaines -->
<ELEMENT liste_semaines (semaine*) >
<ELEMENT semaine (jour*) > <!-- représente une semaine -->
<ATTLIST semaine num_semaine ID #REQUIRED >
<ELEMENT jour (creneau*) > <!-- représente un jour -->
<ATTLIST jour
           nomjour (lu | ma | me | je | ve | sa | di) #REQUIRED
           njour CDATA #REQUIRED
           mois CDATA #REQUIRED>

<!-- njour est le numéro du jour dans le mois, il prend ses valeur dans l'intervalle [01..31] -->
<!-- mois est le numéro du mois (01, 02, etc. jusqu'à 12) -->

<ELEMENT creneau (type?, matiere?) > <!-- représente un créneau d'enseignement -->
<ATTLIST creneau nplage IDREF #REQUIRED
           nsalle IDREF #REQUIRED
           nprof IDREF #IMPLIED>

<!-- nplage est le numéro d'ordre de la plage horaire sur laquelle a lieu le créneau -->
<!-- nsalle est le numéro de la salle dans laquelle a lieu l'enseignement -->
<!-- nprof est le numéro de l'enseignant -->
<ELEMENT matiere (#PCDATA) >
<ELEMENT type (#PCDATA) > <!-- nature de l'enseignement : C, TD, TP ou Conférence (par défaut) -->
```

*edt1213.xml*

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE edt SYSTEM "edt.dtd">
<edt groupe="3" année="2007">
  <liste_salles>
    <salle idsalle="s110" capacite="20" type_salle="TD"/>
    <salle idsalle="i019" capacite="36" type_salle="TP"/>
    <salle idsalle="s24" capacite="30" type_salle="TD"/>
    <salle idsalle="s210" capacite="42" type_salle="TD"/>
    <salle idsalle="i012" capacite="24" type_salle="TP"/>
    <salle idsalle="i040" capacite="36" type_salle="TP"/>
    <salle idsalle="s211" capacite="36" type_salle="TD"/>
  </liste_salles>
  <liste_profes>
    <prof idprof="p243"><nom>Durant</nom><prenom>Max</prenom><dpt>Informatique</dpt><tel>0252125546</tel></prof>
    <prof idprof="p562"><nom>Fabriol</nom><prenom>Jean-Pierre</prenom><dpt>Mathématiques</dpt><tel>0252125487</tel></prof>
    <prof idprof="p377"><nom>Hadirent</nom><prenom>Jucien</prenom><dpt>Informatique</dpt><tel>0252125546</tel></prof>
  </liste_profes>
  <types_creneaux>
    <plage ordre="c1"><hdeb>08:00:00</hdeb><hfin>09:20:00</hfin></plage>
    <plage ordre="c2"><hdeb>09:30:00</hdeb><hfin>10:50:00</hfin></plage>
    <plage ordre="c3"><hdeb>11:00:00</hdeb><hfin>12:20:00</hfin></plage>
    <plage ordre="c5"><hdeb>14:00:00</hdeb><hfin>15:20:00</hfin></plage>
    <plage ordre="c6"><hdeb>15:30:00</hdeb><hfin>16:50:00</hfin></plage>
    <plage ordre="c7"><hdeb>17:00:00</hdeb><hfin>18:20:00</hfin></plage>
    <plage ordre="c8"><hdeb>18:30:00</hdeb><hfin>19:50:00</hfin></plage>
  </types_creneaux>
  <liste_semaines>
    <semaine num_semaine="s12">
      <jour nomjour="lu" njour="19" mois="03">
        <creneau nplage="c2" nsalle="s24" nprof="p243"><type>C</type><matiere>Applis Prog XML</matiere></creneau>
        <creneau nplage="c5" nsalle="s110" nprof="p243"><type>TD</type><matiere>Applis Prog XML</matiere></creneau>
        <creneau nplage="c6" nsalle="s110" nprof="p243"><type>TD</type><matiere>Applis Prog XML</matiere></creneau>
      </jour>
      <jour nomjour="me" njour="21" mois="03">
        <creneau nplage="c1" nsalle="i019" nprof="p377"><type>TP</type><matiere>Projet PHP</matiere></creneau>
        <creneau nplage="c2" nsalle="i019" nprof="p377"><type>TP</type><matiere>Projet PHP</matiere></creneau>
        <creneau nplage="c3" nsalle="i019" nprof="p243"><type>TP</type><matiere>Projet XML</matiere></creneau>
        <creneau nplage="c5" nsalle="s24" nprof="p243"><type>CETE : carto</matiere></creneau>
      </jour>
      <jour nomjour="je" njour="22" mois="03">
        <creneau nplage="c2" nsalle="i019" nprof="p243"><type>TP</type><matiere>Projet XML</matiere></creneau>
        <creneau nplage="c3" nsalle="i019" nprof="p243"><type>TP</type><matiere>Projet XML</matiere></creneau>
      </jour>
      <jour nomjour="ve" njour="23" mois="03">
        <creneau nplage="c1" nsalle="s110" nprof="p243"><type>C</type><matiere>Dialectes XML</matiere></creneau>
        <creneau nplage="c2" nsalle="s110" nprof="p243"><type>TD</type><matiere>Dialectes XML</matiere></creneau>
        <creneau nplage="c5" nsalle="i012" nprof="p243"><type>TP</type><matiere>Dialectes XML</matiere></creneau>
      </jour>
    </semaine>
    <semaine num_semaine="s13">
      <jour nomjour="lu" njour="26" mois="03">
        <creneau nsalle="s110" nplage="c5" nprof="p243"><type>TD</type><matiere>Applis PROG XML</matiere></creneau>
        <creneau nsalle="i012" nplage="c6" nprof="p243"><type>TP</type><matiere>Applis PROG XML</matiere></creneau>
        <creneau nsalle="i012" nplage="c7" nprof="p243"><type>TP</type><matiere>Applis PROG XML</matiere></creneau>
      </jour>
      <jour nomjour="ma" njour="27" mois="03">
        <creneau nsalle="i019" nplage="c5" nprof="p377"><type>TP</type><matiere>J2EE</matiere></creneau>
        <creneau nsalle="i019" nplage="c6" nprof="p377"><type>TP</type><matiere>J2EE</matiere></creneau>
        <creneau nsalle="i019" nplage="c7" nprof="p377"><type>TP</type><matiere>J2EE</matiere></creneau>
      </jour>
      <jour nomjour="me" njour="28" mois="03">
        <creneau nsalle="i019" nplage="c2" nprof="p243"><type>TP</type><matiere>Projet XML</matiere></creneau>
        <creneau nsalle="i019" nplage="c3" nprof="p243"><type>TP</type><matiere>Projet XML</matiere></creneau>
        <creneau nsalle="s110" nplage="c5"><matiere>ZOPE</matiere></creneau>
        <creneau nsalle="s110" nplage="c6"><matiere>ZOPE</matiere></creneau>
        <creneau nsalle="s211" nplage="c7" nprof="p243"><matiere>Dialectes XML</matiere></creneau>
      </jour>
      <jour nomjour="ve" njour="30" mois="03">
        <creneau nplage="c1" nsalle="s110" nprof="p243"><type>C</type><matiere>Dialectes XML</matiere></creneau>
        <creneau nplage="c2" nsalle="s110" nprof="p243"><type>TD</type><matiere>Dialectes XML</matiere></creneau>
        <creneau nplage="c5" nsalle="s211" nprof="p243"><type>TD</type><matiere>Dialectes XML</matiere></creneau>
      </jour>
    </semaine>
  </liste_semaines>
</edt>

```



Cette création est mise à disposition selon le Contrat Paternité-Pas d'Utilisation Commerciale-Pas de Modification 2.0 France disponible en ligne <http://creativecommons.org/licenses/by-nc-nd/2.0/fr/> ou par courrier postal à Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.