LO17

Introduction à XML : principes, syntaxe, schémas et manipulations

https://stph.scenari-

Stéphane Crozat

19 février 2016

Table des matières

Objectifs

- Connaître les principes du méta-langage XML pour les applications orientées données et orientées documents
- Savoir lire et écrire un document XML
- Savoir réaliser un schéma XML simple
- Savoir réaliser une transformation XML simple

Exemple : Un document XML

1

Exemple : Un mail

```
1  <?xml version='1.0' encoding='iso-8859-1'?>
2  <mail>
3   <de>stephane.crozat@utc.fr</de>
4   <a>fabrice.issac@utc.fr</a>
5   <objet>Demande d'information</objet>
6   <date>01-03-2001</date>
7   <corps>
8    <paragraphe>Bonjour Fabrice,</paragraphe>
9    <paragraphe>Peux tu m'expliquer ce qu'est le langage XML ?
    </paragraphe>
10   <paragraphe>Il me faudrait en effet ton avis éclairé sur le
    sujet.</paragraphe>
11   <paragraphe>J'attends ta réponse, à bientôt</paragraphe>
12  </corps>
13 </mail>
```

Principes

II

A. Exercices : Culture XML

1. Exercice

[*Solution n°1 p 49*]

Exercice

Historique d'XML : Cocher toutes les propositions correctes.

- ☐ XML a été créé dans la continuité du SGML
- ☐ XML est une norme ISO
- ☐ XML est un standard W3C
- ☐ XML est plus récent que HTML

Exercice

XML versus HTML : Cocher toutes les propositions correctes.

- ☐ HTML est un langage de mise en forme
- ☐ XML est un langage de mise en forme
- ☐ HTML définit un jeu de balises fixé a priori et donc non-extensibles
- ☐ XML est beaucoup plus générique, il permet de définir des langages

Exercice

Qu'est ce qui est nécessaire pour avoir un document XML bien formé ?

☐	Un entête XML
☐	Une référence à une DTD
☐	Une référence à un XML Schema
☐	Une unique balise racine qui encadre toutes les autres
☐	Que chaque balise ouverte soit fermée

Exercice

Que permet un schéma XML ?

☐	D'indiquer la mise en page pour un type de document et un format cible.
☐	De vérifier automatiquement qu'un document correspond à une structure type.
☐	De définir une grammaire documentaire.

2. Exercice : Galaxie XML

[*Solution n°2 p 50*]

Ranger les termes suivants dans les bonnes catégories.

1 - SVG
2 - XHTML
3 - Open Document Format
4 - DocBook
5 - SGML
6 - HTML
7 - W3C
8 - OASIS
9 - SMIL
10 - XML

Méta-langage	Format de publication XML	Format de publication SGML	Langage de structuration documentaire	Organisme de standardisation

B. XML, l'eXtensible Markup Language

1. Définition du XML

Définition : XML

L'eXtensible Markup Language XML [w_w3c.org/XML] est un **méta-langage** permettant de définir des langages à **balises**.

Il standardisé comme une recommandation par le W3C✭ depuis **1998**.

Exemple

- LaTeX et HTML sont des langages à balises (mais ce ne sont pas des langages XML)
- XHTML est un langage XML

Fondamental

En tant que méta-langage XML sert à définir des **formats** informatiques, c'est à dire des façons de représenter de l'information.

Les bonnes caractéristiques d'XML - **non-ambiguïté, lisibilité, passivité** - en font un très bon candidat pour de très nombreux usages, il est donc utilisé dans des secteurs très variés de l'informatique.

On distingue généralement deux grandes catégories d'utilisation : **les langages orientés données et les langages orientés documents**.

Complément : Versions de XML

La version historique de XML est la version 1.0, qui reste aujourd'hui la plus largement utilisée.

Il existe également une version 1.1 de XML, qui propose des différences mineures et nécessaires uniquement dans des contextes particuliers. Cette nouvelle version a été nécessaire pour des raisons de compatibilité des outils existants. Les évolutions principales de XML 1.1 sont de nouveaux caractères permis pour les noms d'éléments (pour suivre l'évolution d'Unicode depuis 1998), de nouvelles conventions pour les caractères de fin de ligne (pour la compatibilité avec des ordinateurs *main frame*) et de nouveaux caractères de contrôle dans le contenu.

Complément : Voir aussi

XML : Un langage à balise

XML : un méta-langage

Langages XML orientés données

Langages XML orientés documents

Caractéristiques recherchées pour un format XML - p.70

2. XML : un langage à balise

Définition : Langage à balises

Une balise est un descripteur ajouté au contenu en vue d'un traitement informatique.

Un langage à balises est un langage permettant d'associer à un contenu (généralement du texte) des balises explicites (par exemple pour rendre compte de la structure du texte).

Exemple : Balises Lambda

```
1 <lambda>
2   <body>
3     <p>This is an example</p>
4   </body>
5 </lambda>
```

Ici le texte est "This is an example", il est balisé afin d'ajouter les informations suivantes : c'est est un paragraphe (balise `p`) du corps (balise `body`) d'un document Lambda (balise `lambda`).

Complément : Voir aussi

Balises et poignées de calcul - p.65

3. XML : un méta-langage

Définition : Méta-langage

Un méta-langage est un langage permettant de définir d'autres langages, les langages ainsi définis permettent à leur tour la description d'informations respectant ces langages.

Exemple : Langage Lambda

Si l'on désire définir informatiquement un langage Lambda, en disant qu'il peut contenir un élément de type `body` qui lui même contient des éléments de type `p`, il faut spécifier les contraintes de ce langage grâce à un méta-langage permettant donc de définir :

- Le vocabulaire : `lambda`, `body`, `p`
- La grammaire : `lambda` contient exactement un `body` qui contient un ou plusieurs `p`

XML en tant que méta-langage ne contient pas les mots du langage Lambda, en revanche il dispose des mécaniques permettant de les définir.

Définition : Notion de schéma

La définition d'un langage XML particulier se fait grâce à un schéma qui permet de lister les mots du langage (vocabulaire) et de définir leur utilisation (grammaire).

On parle également de **format**.

Exemple : Schéma Lambda

```
1 <!ELEMENT lambda (body) >
2 <!ELEMENT body (p+) >
3 <!ELEMENT p (#PCDATA) >
```

Ce code exprime formellement (avec la syntaxe DTD) le langage, ou format, Lambda.

4. Langages standard et langages locaux

Définition : Format standard

On appelle langage (ou format) standard un langage dont le schéma est défini par une organisation reconnue et multipartite (W3C, ISO, OASIS, ...) pour les besoins génériques d'une large communauté, en général internationale.

Remarque : Standard de fait

On appelle standard de fait les langages imposés par une organisation unique, mais à une telle échelle que leur usage est incontournable. C'est typiquement l'exemple des formats de la suite bureautique Microsoft Office. (.doc, .xls, .ppt, ...).

Définition : Format local

On appelle langage (ou format) local - par opposition au format standard - un langage défini par une organisation seule ou une communauté restreinte pour ses besoins propres (un service, une entreprise, voire un réseau d'organisations dans un pays, ...).

Définition : Format ouvert ou fermé

Un format est dit ouvert lorsqu'il est public et que n'importe qui peut l'utiliser sans contrainte. Il est au contraire fermé si son accès est protégé, payant ou soumis à d'autres contraintes.

Remarque : Format propriétaire

On parle de format propriétaire pour désigner un format à la fois local et fermé.

5. Langages XML orientés données

Définition

Ils permettent d'enregistrer et de transporter des données informatiques structurées (comme par exemple des données gérées par des bases de données) selon des formats ouvert (c'est à dire dont on connaît la syntaxe) et facile à manipuler (les structures arborescentes XML étant plus riches que des fichiers à plat par exemple).

Les langages XML orientées données servent surtout à l'échange ou la sérialisation des données des programmes informatiques.

Remarque

L'utilisation d'XML est en fait ici assez accessoire, d'autres formalismes s'y substituent sans difficulté, souvent moins explicites pour la manipulation humaine et souvent plus performant pour la manipulation informatique : CSV, JSON, ... (voir par exemple : *http://www.xul.fr/ajax-format-json.html*[1] pour une comparaison JSON / XML).

Remarque : Format verbeux

XML est en général plus verbeux que ses alternatives (il contient plus de caractères), c'est pourquoi il est plus explicite pour un humain (ce qui n'est pas

1 - http://www.xul.fr/ajax-format-json.html

toujours utile), et moins performant informatiquement (ce qui n'est pas toujours un problème).

Exemple : Formats XML orientés données standards

- MathML, ChemML, ...
- ATOM, RSS
- Dublin Core
- RDF, OWL
- SVG
- ...

Exemple : Formats XML orientés données locaux

XML est utilisé pour de très nombreux langages orientés données locaux, en fait chaque fois qu'un format XML est défini pour les besoins spécifique d'un programme.

```
1  <myVector>
2    <x>5</x>
3    <y>19</y>
4  </myVector>
```

Complément : Langages XML de programmation

Le formalisme XML est également utilisé pour définir des langages de programmation, à l'instar du C ou du Java. Les langages de programmation écrits en XML sont généralement à vocation déclarative et adressent le plus souvent des traitements eux mêmes liés à XML.

XSL-XSLT est un exemple de langage de programmation écrit en XML. On peut également citer par exemple le langage de script ANT (*http://ant.apache.org/*[2]) ou XUL pour la réalisation d'IHM (*http://www.xul.fr/*[3])

6. Langages XML orientés documents

Définition

Ils permettent de représenter informatiquement des documents numériques. Les formats de documents faisant appel à des représentations fortement arborescentes, XML est un candidat idéal pour cet usage. En fait SGML, l'ancêtre de XML, a été inventé pour l'informatique documentaire. Aujourd'hui la très grande majorité des formats de représentation de document sont en XML.

Définition : Langages XML orienté documents formatés

Ils définissent des formats de mise en forme de document (structure physique), en général pour un support donné.

Exemple : Langages XML orientés documents formatés

- XHTML
- XSL-FO

2 - http://ant.apache.org/
3 - http://www.xul.fr/

- SMIL
- OpenDocument
- OOXML
- ...

Définition : Langages XML orientés documents structurés

Ils définissent des formats de structuration de document (structure logique), en général pour un métier donné, plus ou moins précis selon la généralité du langage.

Exemple : Langages XML orientés documents structurés

- DocBook
- TEI
- DITA
- ...

Exemple : Format local orienté document structuré

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <document type="Lorem ipsum">
3    <paragraphe>Lorem ipsum dolor sit amet, consectetur adipiscing
   elit. Pellentesque sit amet
4  libero ac mauris egestas venenatis nec vitae sapien.</paragraphe>
5    <paragraphe>Donec a lectus sed augue pellentesque accumsan eu ac
   justo. Etiam est urna, sagittis
6  ac cursus nec, placerat quis velit.</paragraphe>
7  </document>
```

Complément : Voir aussi

Documents structurés et documents formatés - p.65

Syntaxe

III

A. Exercices : Syntaxe XML

1. Exercice

[*Solution n°3 p 51*]

Soit le fichier ci-après, quelles sont les assertions vraies ?

```
1  <papier type="scientifique">
2   <titre>Réinterroger les structures documentaires</titre>
3   <sousTitre>De la numérisation à l'informatisation</sousTitre>
4   <auteur>Stéphane Crozat</auteur>
5   <auteur>Bruno Bachimont</auteur>
6   <resume>Nous proposons dans cet article d'aborder ...</resume>
7   <abstract>In this paper we define...</abstract>
8   <motsCles>
9    <terme>Ingénierie des connaissances</terme>
10   <terme>XML</terme>
11   <terme>Document</terme>
12  </motsCles>
13  <keywords>
14   <word>Knowledge engineering</word>
15   <word>XML</word>
16   <word>Document</word>
17  </keywords>
18  <publication date="2004-07-05"/>
19  <version maj='1' min='0'/>
20  <ressource
    uriSrc="http://archivesic.ccsd.cnrs.fr/docs/00/06/23/97/PDF/sic_00001
    015.pdf"/>
21 </papier>
```

☐	Ce fichier n'est pas un fichier XML
☐	Ce fichier est un fichier XML, il possède un élément racine unique qui contient tous les autres.
☐	Ce fichier est un fichier XML, tous ses éléments sont inclus dans les uns dans les autres.
☐	Ce fichier est un fichier XML, toutes ses balises ouvertes sont fermées.
☐	Ce fichier est un fichier SGML.

2. Exercice

[*Solution n°4 p 51*]

*Compléter les trous avec les balises **fermantes** adéquates.*

```
<?xml version="1.0"?>
<document>
  <entete>
    <titre>Document à corriger</titre>
    <auteur>Stéphane Crozat
    <date>01-03-01
    <version numero="1"/>

  <corps>
    <division titre="Première partie">
      <paragraphe>Ce texte doit être <important>corrigé </paragraphe>
      <paragraphe>Le contenu est sans importance ici</paragraphe>
    </division>
    <division titre="Seconde partie">
      <paragraphe>Ai-je le droit de mettre du texte ici ?</paragraphe>
```

3. Exercice

[*Solution n°5 p 51*]

Fermer les balises du fichier suivant afin qu'il soit un document XML bien formé.

```
<docAbstrait>
<contenu>
<paragraphe>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</ >
<Texte>Nulla erat tellus, molestie a ultrices sed, gravida eget ipsum.
<para>Phasellus lacinia, ipsum vitae interdum tincidunt.
```

<P>Nullam pulvinar diam et tellus ullamcorper eleifend.</ ></ ></ ></ >

<ligne>Nunc eu lectus in diam tempus adipiscing in rhoncus elit.</ ></ >

B. Syntaxe de base XML

1. Document bien formé

Définition

Un document est dit bien formé lorsqu'il respecte les règles syntaxiques du XML✯.

Fondamental

XML✯ ne pose pas de sémantique à priori mais uniquement des règles syntaxiques.

Syntaxe

Les règles syntaxiques à respecter sont :

- Il n'existe qu'un seul élément père par document, cet élément contenant tous les autres. Il est également appelé **racine**.
- Tout élément fils est inclus complètement dans son père (pas d'éléments croisés).

Attention

XML ne définit pas le comportement ni la manière dont doit être traité un document, mais uniquement un **format**.

2. Balise

Définition : Balise

Les balises sont les composants fondamentaux permettant l'écriture de documents XML✯.

Elles se distinguent du contenu en utilisant les caractères `<`, `>` et `/`. Elles n'ont pas de présentation ou de signification définie par le langage mais elles auront un sens pour les applications et/ou le lecteur.

Syntaxe

Il existe trois types de balises :

- balise d'ouverture : `<nom_balise>`
- balise de fermeture : `</nom_balise>`
- balise vide : `<nom_balise/>`

Exemple : Exemple de balise XML

```
1  <adresse>12, rue de Paris</adresse>
```

3. Élément

Définition : Élément

Un élément XML★ est un extrait d'un fichier XML comprenant :

- une balise ouvrante
- une balise fermante
- le contenu compris entre les deux balises, qui peut être du **texte** et/ou d'autres **éléments**, ou **rien** (élément vide)

Le nom d'un élément peut être composé de tout caractère alphanumérique plus _, - et . . De plus, ils doivent commencer par un caractère alphabétique ou _ et ceux commençant par la chaîne `xml` sont réservés (que ce soit en minuscules, majuscules ou un mélange des deux).

Attention : Case-sensitive

XML★ différencie les majuscules des minuscules, il faut donc respecter la casse.

Attention

Deux éléments **ne peuvent pas** avoir un contenu croisé : `<nom1> ... <nom2> ... </nom1> ... </nom2>` est interdit.

Définition : Élément vide

On appelle élément vide un élément qui ne comporte rien entre sa balise ouvrante et sa balise fermante.

Syntaxe : Élément vide

Les syntaxes `<element></element>` et `<element/>` sont strictement équivalentes.

4. Attribut

Définition

Un attribut est une information supplémentaire attachée à un élément, on parle de métadonnée.

Syntaxe

Les attributs d'un élément sont formés d'une suite d'affectations séparées par des espaces : `attribut1='valeur' attribut2='valeur' ...`

Ils sont ajoutés à la balise ouvrante ou à une balise vide (jamais à une balise fermante) :

- `<nom_element [attributs]>`
- `<nom_element [attributs]/>`

La valeur est indiquée entre apostrophes ou guillemets (au choix, mais pas de mélange des deux) :

- `attribut1='valeur'` ou
- `attribut1="valeur"`

Méthode

Utilisez des apostrophes si la valeur de l'attribut inclut des guillemets et vice et versa.

Attention

Un élément ne peut pas contenir deux attributs ayant le même nom.

Syntaxe

Le nom d'un attribut est soumis aux mêmes contraintes que les noms d'éléments.

La valeur de l'attribut quant à elle peut contenir tout caractère à l'exception de `^`, `%` et `&`.

Remarque : Équivalence attribut / élément

Un attribut peut toujours être représenté alternativement par un élément fils de l'élément qu'il caractérise, avec une signification du même ordre :

- `<element attribut="x"/>` et
- `<element><attribut>x</attribut><element>` sont similaires.

Il est donc tout à fait possible de faire du XML sans utiliser d'attribut.

Méthode : Usage des attributs

On utilise généralement les attributs :

- Pour différencier le contenu destiné à être affiché dans le document lisible des métadonnées qui ne le seront pas (version, date de création, ...)
- Pour simplifier l'écriture du document
- Pour ajouter des identifiants et des références

5. Structure générale d'un fichier XML

Syntaxe

Un document XML✯ est constitué de :

- Un **prologue**

  Il est facultatif et comprend une déclaration XML✯, indiquant la version du langage XML utilisé et le codage des caractères dans le document. Chacune de ces informations est optionnelle mais leur ordre est obligatoire

  `<?xml version="numéro de version" encoding="encodage des caractères"?>`

- Un **arbre d'éléments** contenant au moins un élément (l'élément racine)

Exemple : Prologue

```
1 <?xml version="1.0" encoding="UTF-8"?>
```

Indique que le document est codé en utilisant un langage XML✯ de version 1.0, avec des caractères codés selon la norme UTF-8.

Exemple : Arbre d'éléments

```
1  <lettre>
2    <expediteur>moi</expediteur>
3  </lettre>
```

6. Exemple de fichier XML - Le courriel

Exemple : Un courriel imprimé

Date: Mar, 28 Oct 2003 14:01:01 +0100 (CET)
De : Marcel <marcel@ici.fr>
A : Robert <robert@labas.fr>
Sujet: Hirondelle
Salut,
Pourrais-tu m'indiquer quelle est la vitesse de vol d'une hirondelle transportant une noix de coco ?
A très bientôt,
Marcel

Représentation possible de ce message en XML

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <email>
3   <entete>
4    <date type='JJMMAAAA'>28102003</date>
5    <heure type='24' local='(GMT+01 :00)'>14:01:01</heure>
6    <expediteur><adresse
     mail='marcel@ici.fr'>Marcel</adresse></expediteur>
7    <recepteur><adresse
     mail='robert@labas.fr'>Robert</adresse></recepteur>
8    <sujet>Hirondelle</sujet>
9   </entete>
10  <corps>
11   <salutation>Salut,</salutation>
12   <paragraphe>Pourrais-tu m'indiquer quelle est la vitesse de vol
     d'une hirondelle transportant une noix de coco ?</paragraphe>
13   <politesse>A très bientôt,</politesse>
14   <signature>Marcel</signature>
15  </corps>
16 </email>
```

Complément : Arbre d'éléments du document email correspondant

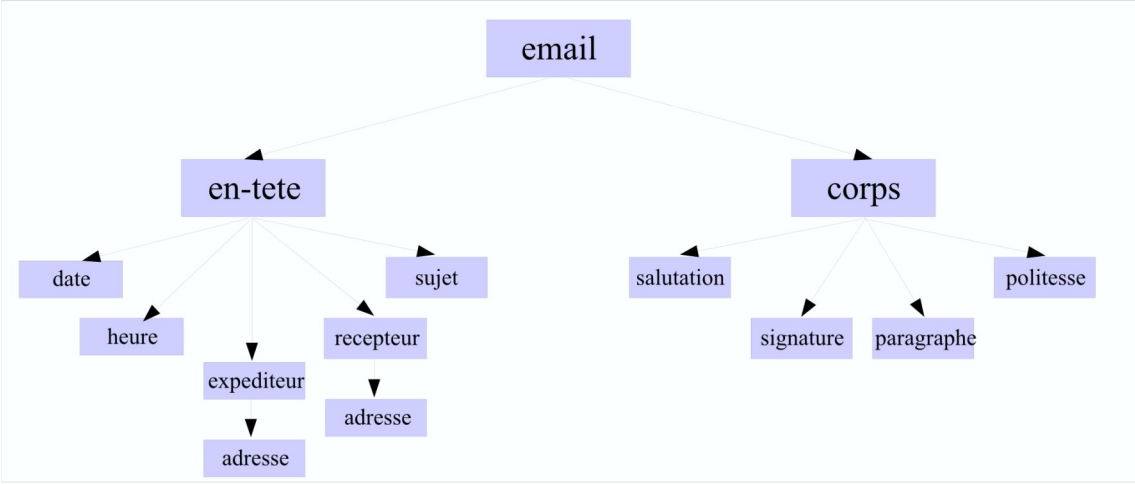


Image 1 Représentation d'un document XML sous forme d'arbre

7. La syntaxe XML en résumé

Les règles de syntaxe d'XML sont très simples : un document XML comporte un en-tête permettant de l'identifier en tant que document XML et un élément racine contenant tous les autres éléments. Un élément est caractérisé par une balise ouvrante et une balise fermante et contient d'autres éléments et/ou du contenu.

Aspect	Syntaxe	Explication
Entête	<?xml version="1.0" encoding="UTF-8"?>	Un document XML peut contenir une ligne d'entête qui sert à identifier le document en tant que document XML, et généralement à spécifier le système d'encodage de caractères utilisé dans le document.
Elément racine	<NomElementRacine> ... </NomElementRacine>	Tout document XML contient un et un seul élément, dit racine, contenant tous les autres.
Elément	<NomElement> ... </NomElement>	Un élément est caractérisé par une balise ouvrante et une balise fermante obligatoire, les éléments sont donc inclus les uns dans les autres et ne peuvent se croiser. Les balises tiennent compte de la casse des caractères.
Attribut	<NomElement NomAttribut="valeur">	Un attribut associé à une valeur peut être spécifié dans la balise ouvrante d'un élément.
Elément vide	<NomElementVide/>	Un élément peut être vide, il est alors noté par une unique balise à la syntaxe particulière. Un élément vide contient en général des attributs.
Contenu	<NomElement> contenu ici </NomElement>	Le contenu est placé à l'intérieur des éléments, c'est à dire entre les balises.

Tableau 1 Aspects principaux de la syntaxe XML

Schémas

IV

A. Une DTD

Soit la DTD suivante :

```
1 <!ELEMENT entete (titre, date, auteur+, motscles*, resume?)>
2 <!ELEMENT titre (#PCDATA)>
3 <!ELEMENT date (#PCDATA)>
4 <!ELEMENT auteur (#PCDATA)>
5 <!ELEMENT motscles (#PCDATA)>
6 <!ELEMENT resume (paragraphe+)>
7 <!ELEMENT paragraphe (#PCDATA)>
```

Question 1

[*Solution n°6 p 52*]

Produire un document XML valide par rapport à cette DTD

Question 2

[*Solution n°7 p 52*]

Le document suivant est-il valide par rapport à la DTD ?

```
1 <?xml version="1.0"?>
2 <!DOCTYPE entete SYSTEM "entete.dtd">
3 <entete>
4   <titre>Document de test</titre>
5   <date>2 décembre 2009</date>
6   <auteur>Stéphane Crozat</auteur>
7   <motscles>Document DTD XML Valide</motscles>
8 </entete>
```

Question 3

[*Solution n°8 p 52*]

Produire le plus petit document XML valide par rapport à cette DTD.

Question 4

[*Solution n°9 p 52*]

Produire un document XML utilisant toutes les balises de la DTD.

B. Introduction aux schémas XML

1. Notion de document valide

Définition : Schéma

Un schéma est une description de la structure que doit respecter un document lui faisant référence, c'est à dire qu'il établit la liste des éléments XML autorisés (avec leurs attributs), ainsi que l'agencement possible de ces éléments.

On parle aussi de **grammaire**, au sens où le schéma définit l'enchaînement autorisé des balises et vient en **complément de la syntaxe XML** (qui elle est indépendante d'un schéma particulier).

Définition : Document valide

Un document XML★ bien formé est dit valide pour un schéma donné s'il respecte les règles structurelles imposées par ce schéma.

Ce contrôle de la structure permet :

- De s'assurer l'homogénéité structurelle des documents de même type.
- Le traitement automatique d'un ensemble de documents de même type (mise en forme, stockage, extraction d'informations...).
- La création de formats standard et leur respect.

Exemple : Exemples de langages de schéma

Il existe plusieurs langages de définition schéma, mais les trois principaux sont :

- Document Type Définition (W3C) : Un langage hérité de SGML qui fait partie du standard XML
- W3C XML Schema (W3C) : Une alternative aux DTD destiné à moderniser et compléter ce langage historique
- Relax NG (OASIS, ISO) : Une autre alternative, compromis entre W3C XML Schema et DTD

2. Document Type Definition

Le formalisme de définition de schéma DTD est le premier qui a été introduit dès la première version du standard XML. Il est en fait intégré au standard W3C de XML.

Il est directement hérité de la norme SGML.

Les DTDs utilisent un langage spécifique (non XML) pour définir les règles structurelles. Un fichier de DTD peut contenir principalement deux types de déclarations :

- **des déclarations d'éléments,**

  indiquent les éléments pouvant être inclus dans un document et l'organisation du contenu de chaque élément (éléments fils ou texte).
- **des déclarations d'attributs,**

  définissent les attributs pouvant être associés à un élément ainsi que leur type.

Exemple : Exemple de DTD

```
<!ELEMENT document (paragraphe+)>
<!ATTLIST document type CDATA #REQUIRED>
<!ELEMENT paragraphe (#PCDATA)>
```

Exemple : Exemple de document XML valide

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE document SYSTEM "document.dtd">
<document type='memo'>
 <paragraphe>Xxxxxxxxx</paragraphe>
 <paragraphe>Xxxxxxxxx</paragraphe>
 <paragraphe>Xxxxxxxxx</paragraphe>
</document>

```

3. W3C XML Schema

Les XML Schema ont été proposés par le W3C pour permettre de dépasser les limites des DTD.

http://www.w3.org/XML/Schema[4]

On notera en particulier :

- une syntaxe XML
- l'extension de l'expression des règles d'organisation structurelle (héritage, réutilisation, etc.)
- l'ajout d'un langage de typage des éléments (particulièrement utile pour les format XML orientés données)

Exemple : Exemple de DTD

```
<!ELEMENT document (paragraphe+)>
<!ATTLIST document type CDATA #REQUIRED>
<!ELEMENT paragraphe (#PCDATA)>
```

Exemple : Exemple de W3C XML Schema correspondant

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="document">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="paragraphe"/>
      </xs:sequence>
      <xs:attribute name="type" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="paragraphe" type="xs:string"/>
</xs:schema>
```

4 - http://www.w3.org/XML/Schema

4. Regular Language for XML Next Generation

RelaxNG (REgular LAnguage for XML Next Generation) est un langage de schéma XML.

- RelaxNG est une alternative aux DTD et à W3C XML Schema, qui combine les avantages de ces deux autres langages.
- RelaxNG est un standard OASIS et une norme ISO/CEI.
- Deux syntaxes : une syntaxe XML (alternative à W3C Schema) et une syntaxe compacte (alternative aux DTD).
- RelaxNG ne définit que la structure (comme les DTD) et utilise W3C XML Schema pour le typage des données.

http://relaxng.org/[5]

Complément

Le standard est porté par James Clark depuis ses travaux sur Trex (il est issu de la fusion de Trex et Relax de Murata Makoto).

Exemple : Exemple de schémas publics définis en Relax NG

- OpenDocument (format bureautique)
- DocBook (format documentaire)
- Atom (syndication)

Exemple : Exemple de DTD

```
1  <!ELEMENT document (paragraphe+)>
2  <!ATTLIST document type CDATA #REQUIRED>
3  <!ELEMENT paragraphe (#PCDATA)>
```

Exemple : Exemple de schéma RelaxNG correspondant (syntaxe XML)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <grammar xmlns="http://relaxng.org/ns/structure/1.0">
3    <start>
4      <element name="document">
5        <attribute name="type"/>
6        <oneOrMore>
7          <element name="paragraphe">
8            <text/>
9          </element>
10         </oneOrMore>
11     </element>
12   </start>
13 </grammar>
```

Exemple : Exemple de schéma RelaxNG correspondant (syntaxe compacte)

```
1  start = element document {
2    attribute type {text},
3    element paragraphe {text}+
```

5 - http://relaxng.org/

```
4 }
```

Exemple : Autre exemple de schéma RelaxNG correspondant (syntaxe XML, patterns nommés)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <grammar xmlns="http://relaxng.org/ns/structure/1.0">
3    <start>
4      <ref name="document"/>
5    </start>
6    <define name="document">
7      <element name="document">
8        <attribute name="type"/>
9        <oneOrMore>
10         <ref name="paragraphe"/>
11       </oneOrMore>
12     </element>
13   </define>
14   <define name="paragraphe">
15     <element name="paragraphe">
16       <text/>
17     </element>
18   </define>
19 </grammar>
```

Exemple : Autre exemple de schéma RelaxNG correspondant (syntaxe compacte, patterns nommés)

```
1 start = document
2 document = element document {attribute type {text}, paragraphe+ }
3 paragraphe = element paragraphe {text}
```

Manipulations

V

A. XPath et XSLT

1. L'arbre du document XML

Il est possible de représenter un document XML sous forme d'arbre, tel que :

- L'arbre possède une racine / qui a comme fils l'élément racine
- l'élément racine est l'élément du document XML qui contient tous les autres
- chaque nœud a comme fils les éléments et le texte qu'il contient, ainsi que ses attributs.

Exemple : Fichier XML

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <document modele="ULCoursGeneral" code="BP-Incendie3_S1_E2_UL1">
3   <entete>
4    <identification>
5     <titre>L'assurance de la responsabilité de voisinage</titre>
6     <date>21/02/01</date>
7     <auteur>AEA</auteur>
8     <version>1.00</version>
9    </identification>
10  </entete>
11  <corps>
12   <contenu>
13    <paragraphe>Cette garantie est appelée : recours des voisins et
    des tiers.</paragraphe>
14    <remarque>
15     <paragraphe>L'image suivante <ressource URIsrc="img07.jpg"
    titre="Recours des voisins et des tiers" type="image"/> montre la
    garantie.</paragraphe>
16    </remarque>
17   </contenu>
18  </corps>
19 </document>
```

Exemple : Arbre XML

```
/
 |document
   |@modele = "ULCoursGeneral"
   |@code = "BP-Incendie3_S1_E2_UL1"
   |entete
     |identification
       |titre
         |text() = "L'assurance de ..."
       |date
         |text() = "21/02/01"
       |auteur
         |text() = "AEA"
       |version
         |text() = "1.00"
     |corps
       |contenu
         |paragraphe
           |text() = "Cette garantie ..."
         |remarque
         |paragraphe
           |text() = "L'image suivante"
           |ressource
             |@URIsrc = "img07.jpg"
             |@titre = "Recours des voisins..."
             |@type = "image"
           |text() = "montre la garantie."
```

Remarque : Ordre des nœuds

L'ensemble des nœuds de l'arbre d'un document est muni d'un ordre, qui est celui de l'ordre dans le document XML sérialisé.

2. Introduction à XPath

Définition : Expression XPath

XPath est un langage d'expressions permettant de pointer sur n'importe quel élément d'un arbre XML depuis n'importe quel autre élément de l'arbre.

- Une expression XPath peut-être **absolue** (sa résolution est **indépendante** d'un contexte ou nœud courant : elle commence dans ce cas par `/`.
- Une expression XPath peut-être **relative** (sa résolution est **dépendante** d'un contexte ou nœud courant : elle ne commence dans ce cas pas par `/`, elle peut commencer par `./` (syntaxe développée).

Fondamental

Une expression XPath renvoie

- un *node-set*, ou ensemble de nœuds, c'est à dire un sous-arbre de l'arbre du document
- une chaîne de caractères
- un booléen
- un réel

Exemple : Exemples d'expressions XPath

```
1  /document/entete/identification/titre
2  /document/@modele
3  corps//contenu
4  contenu/*
5  contenu/remarque[1]
6  ../paragraphe
7  @type
```

Complément : Types de nœuds XPath

- root nodes
- element nodes
- text nodes
- attribute nodes
- namespace nodes
- processing instruction nodes
- comment nodes

http://www.w3.org/TR/xpath/#data-model[6]

Complément

Pour une introduction à XPath : Brillant07 [Brillant07] pp.123-129

3. Exercice

[*Solution n°10 p 52*]

Soit le fichier XML ci-dessous. Si le nœud courant est un des éléments `terme`*, écrivez 4 expressions XPath différentes permettant de renvoyer le titre du document :*

1. *Sachant que* `titre` *est unique dans tout le document.*
2. *Sachant que* `titre` *est le fils de l'élément racine* `papier`*.*
3. *Sachant que* `titre` *est le fils du père du père du nœud courant.*
4. *Sachant que* `titre` *est avant le nœud courant dans l'ordre du document.*

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <papier type="scientifique">
3   <titre>Réinterroger les structures documentaires</titre>
4   <sousTitre>De la numérisation à l'informatisation</sousTitre>
5   <auteur>Stéphane Crozat</auteur>
6   <auteur>Bruno Bachimont</auteur>
7   <resume>Nous proposons dans cet article d'aborder ...</resume>
8   <abstract>In this paper we define...</abstract>
9   <motsCles>
10   <terme>Ingénierie des connaissances</terme>
11   <terme>XML</terme>
12   <terme>Document</terme>
13  </motsCles>
14  <keywords>
15   <word>Knowledge engineering</word>
16   <word>XML</word>
17   <word>Document</word>
18  </keywords>
```

6 - http://www.w3.org/TR/xpath/#data-model

```
19    <publication date="2004-07-05"/>
20    <version maj='1' min='0'/>
21    <ressource
      uriSrc="http://archivesic.ccsd.cnrs.fr/docs/00/06/23/97/PDF/sic_00001
      015.pdf"/>
22    </papier>
```

1. //
2. /
3. ..
4. preceding

4. Définition de XSL-XSLT

Définition : XSL-XSLT

XSL-XSLT est une partie du standard W3C XSL qui a trait à la transformation des documents XML (l'autre partie étant XSL-FO).

XSL-XSLT est un langage de programmation déclaratif écrit en XML (un programme XSL-XSLT est un document XML).

- XSL-XSLT est langage de manipulation de document XML (fondé sur XPath et sur le modèle arborescent de représentation des documents XML)
- XSl-XSLT est utilisé pour transformer un document XML source dans un autre format, typiquement HTML, mais aussi tout autre format codé sur des caractères dont la syntaxe est connue.
- XSL-XSLT est aussi utilisé pour faire des changements de schéma XML (export d'un XML vers un autre XML structuré différemment) par exemple pour échanger des données selon un standard.

Remarque : XSL-XSLT, XSL-FO, XSLT, XSL, FO

On parle souvent (par simplification) de XSL ou de XSLT pour désigner XSL-XSLT et de FO pour désigner XSL-FO.

XSL utilisé seul désigne donc par convention XSL-XST (et non XSL-FO).

5. Principe de XSL-XSLT

XSL-XSLT fonctionne selon le principe suivant :

1. Il prend en entrée un fichier XML bien formé
2. Il livre en sortie un fichier texte (XML, HTML ou texte sans balise)

Algorithme

L'algorithme général de XSL-XSLT est :

1. Il sélectionne (*match*) les éléments XML du fichier source.
2. Pour chaque élément reconnu il génère une sortie sur le fichier cible.

Notion de règle

Un programme XSL-XSLT est composé d'une succession de règles.

Chaque règle est indépendante des autres et à en charge de sélectionner un élément dans la source et d'effectuer une écriture dans la cible.

Exemple : Application d'une règle XSL-XSLT

Sources :

```
1 <a>
2  <b/><b/><c/>
3 </a>
```

Règle XSL-XSLT :

```
1 <xsl:template match="/a/b">
2   BONJOUR
3 </xsl:template>
```

Résultat :

```
1 BONJOUR BONJOUR
```

6. Exercice

[*Solution n°11 p 53*]

Soit le fichier `file.xml`. Compléter le fichier XSLT `transf.xsl` afin de générer, pour chaque élément `terme`, une instruction SQL d'insertion dans une table relationnelle de schéma : `tMotsCles (terme, titre, url)` (où `terme` est le terme sélectionné, `titre` est le titre du document et `url` est l'adresse de la ressource associée).

Pour rappel, la syntaxe d'insertion de données dans une table relationnelle en SQL : `INSERT INTO <Nom de la relation> (<Liste ordonnée des propriétés à valoriser>) VALUES (<Liste ordonnée des valeurs à affecter>).`

Soit le fichier XML de la première question.

```
1  <!--file.xml-->
2  <papier type="scientifique">
3   <titre>Réinterroger les structures documentaires</titre>
4   <sousTitre>De la numérisation à l'informatisation</sousTitre>
5   <auteur>Stéphane Crozat</auteur>
6   <auteur>Bruno Bachimont</auteur>
7   <resume>Nous proposons dans cet article d'aborder ...</resume>
8   <abstract>In this paper we define...</abstract>
9   <motsCles>
10   <terme>Ingénierie des connaissances</terme>
11   <terme>XML</terme>
12   <terme>Document</terme>
13  </motsCles>
14  <keywords>
15   <word>Knowledge engineering</word>
16   <word>XML</word>
17   <word>Document</word>
18  </keywords>
19  <publication date="2004-07-05"/>
20  <version maj='1' min='0'/>
21  <ressource
    uriSrc="http://archivesic.ccsd.cnrs.fr/docs/00/06/23/97/PDF/sic_00001
    015.pdf"/>
22 </papier>
```

<!--transf.xsl-->

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

<xsl:output method="text"/>

```
<xsl:template match="          ">
<xsl:apply-templates select="          "/>
</xsl:template>
<xsl:template match="          ">
          (terme, titre, url)          (
'<xsl:value-of select="          "/>',
'<xsl:value-of select="//          "/>',
'<xsl:value-of select="//          "/>'
);
</xsl:template>
</xsl:stylesheet>
```

B. SAX et DOM

1. Introduction à SAX et illustration avec Java

- SAX est l'acronyme de **Simple API for XML**.
- **SAX** est une API qui sert à parcourir un fichier **XML**, elle se base sur des **événements** déclenchés durant la lecture du flux (balise ouvrante, balise fermante...).
- Développée pour plusieurs langages de programmation (JAVA, C++, PHP, Perl, ...).
- Très utilisé pour les documents très lourds, car il nécessite moins de mémoire que les autres API.

Définition : Principe

La première interface intitulée **ContentHandler**, permet d'analyser le flux entrant à l'aide des événements. Il suffit de redéfinir les fonctions pour avoir le comportement souhaité.

Fondamental : Gestion d'un élément

Un élément correspond à des balises, notamment une balise ouvrante et une balise fermante.

- **startElement** est déclenchée au démarrage d'un élément XML (balise ouvrante), elle a plusieurs paramètres dont l'espace de nommage, le nom de la balise, et les attributs de cette balise s'ils existent.
- **endElement** est déclenchée à la fin du traitement d'une balise.

```
1  public void startElement(String uri, String localName, String
   rawName,  Attributes atts) throws SAXException {
2    System.out.println("Ouverture de la balise : " + localName);
3  }
4
5  public void endElement(String uri, String localName, String qName)
   throws SAXException {
6    System.out.print("Fermeture de la balise : " + localName);
```

```
7  }
```

Exemple : Gestion du début et de la fin d'un document

Dans l'interface **ContentHandler**, ces événements sont gérés par les deux fonctions **startDocument** et **endDocument**.

- La fonction **startDocument** est la première fonction qui est appelée parmi toutes les autres.
- De même **endDocument** est la dernière fonction qui est appelée à la fin de l'analyse.

```
1  public void startDocument() throws SAXException {
2      System.out.println("Debut de l'analyse du document");
3  }
4
5  public void endDocument() throws SAXException {
6      System.out.println("Fin de l'analyse du document" );
7  }
```

2. Principes du DOM

Définition

Le DOM (*Document Object Model*) est un standard W3C qui décrit une API orientée objet permettant de représenter un document XML ou HTML en mémoire afin de la manipuler avec un langage de programmation.

Le standard DOM est indépendant d'un langage de programmation en particulier, et implémenté dans de nombreux langages (JavaScript, Java, PHP, C, ...).

« The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. »

http://www.w3.org/DOM/[7]

Fonctions du DOM

L'API DOM permet de :

- Parcourir un arbre XML ou HTML
- Rechercher des nœuds particuliers, en consulter le contenu
- Créer, ajouter, supprimer, déplacer des nœuds dans l'arbre

Exemple

L'API DOM implémentée en JavaScript permet de manipuler un document HTML ou XHTML, afin d'en modifier dynamiquement (en fonction des actions utilisateurs), la structure et le contenu :

- Faire apparaître ou disparaître des éléments `div` en fonction d'une action utilisateur
- Créer ou initialiser des contrôles de formulaire
- ...

```
1  function fDisplay(pDivId) {
2    var vDiv = document.getElementById(pDivId);
```

7 - http://www.w3.org/DOM/

```
3     vDiv.style.display="block";
4   }
```

Rappel

L'arbre du document XML

3. Fonctions et objets DOM en JavaScript

Exemple

```
1   vDiv = document.getElementById("id1");
```

Exemple

```
1   vDivs = document.getElementsByTagName("p");
2   for (var i=0; i<vDivs.length; i++) {
3     vDivs.item(i)...;
4   }
```

Exemple

```
1   alert(vDiv.firstChild.data) ;
```

Exemple

```
1   vDiv.removeChild(vDiv.firstChild);
```

Exemple

```
1   vDiv.appendChild(vDiv2);
```

Syntaxe

- `Document`
  http://fr.selfhtml.org/javascript/objets/document.htm[8]
- `Node`
  http://fr.selfhtml.org/javascript/objets/node.htm[9]
- `getElementById`
  http://fr.selfhtml.org/javascript/objets/document.htm#get_element_by_id[10]
- `getElementByTagName`
  http://fr.selfhtml.org/javascript/objets/document.htm#get_elements_by_tag_name[11]
- `firstChild`

8 - http://fr.selfhtml.org/javascript/objets/document.htm
9 - http://fr.selfhtml.org/javascript/objets/node.htm
10 - http://fr.selfhtml.org/javascript/objets/document.htm#get_element_by_id
11 - http://fr.selfhtml.org/javascript/objets/document.htm#get_elements_by_tag_name

http://fr.selfhtml.org/javascript/objets/node.htm#first_child[12]

- `data`
  http://fr.selfhtml.org/javascript/objets/node.htm#data[13]

12 - http://fr.selfhtml.org/javascript/objets/node.htm#first_child
13 - http://fr.selfhtml.org/javascript/objets/node.htm#data

Compléments

VI

A. Préambule : XML selon le W3C

XML★ est un standard du W3C★ : Extensible Markup Language (XML) 1.0 (Fifth Edition) [w_w3c.org/TR/2008/REC-xml].

Les définitions suivantes sont directement copiées du site du W3C [w_w3c.org/XML] (novembre 2009).

Définition : What is XML?

The Extensible Markup Language (XML) is a **simple text-based format** for representing **structured information**: documents, data, configuration, books, transactions, invoices, and much more. It was derived from an older standard format called **SGML** (ISO 8879), in order to be more suitable for Web use.

Définition : What is XML Used For?

XML is one of the most widely-used formats **for sharing structured information** today: between programs, between people, between computers and people, both locally and across networks.

Exemple : Extrait d'un lexique

```
<lexique>
  ...
  <entree id="1976">
    <mot>Essuie-glace</mot>
    <description>L'essuie-glace retire automatiquement la pluie du
<voirAussi ref="1965">pare-brise</voirAussi> lorsqu'elle tombe
dessus. Il possède une <voirAussi ref="1977">lame
caoutchoutée</voirAussi> qui peut être changée séparément en cas de
```

```
    besoin.</description>
  </entree>
  <entree id="1977">
    <mot>Lame caoutchoutée</mot>
    ...
  </entree>
  ...
</lexique>
```

B. Historique : de SGML à XML

SGML

XML hérite historiquement (et fonctionnellement) du méta-langage SGML, développé au début des années 1980 et largement utilisé dans le cadre des systèmes documentaires, notamment dans les industries de hautes technologies pour lesquelles la fiabilité de la documentation est cruciale. SGML restait un formalisme insuffisamment explicite et trop complexe, ce qui rendait difficile la programmation d'applications SGML dans des cadres plus larges. SGML laissait notamment des ambiguïtés en terme d'interprétation du balisage. XML est un langage respectant la syntaxe SGML, mais en y ajoutant des contraintes supplémentaires, afin d'en lever les ambiguïtés.

- **A l'origine SGML** :
  - norme ISO (1986) créé pour la représentation de documents structurés
  - bien adapté aux systèmes documentaires massifs
  - exemple de langage SGML : HTML
- **Un formalisme pas assez explicite** :
  - rend difficile la programmation d'applications SGML (navigateurs, etc.)
  - XML (98) : hérite de SGML en contraignant la syntaxe

Exemple : Ambiguïté en SGML

En SGML une balise ouvrante n'est pas obligatoirement fermée par une balise fermante (sa fermeture pouvant rester implicite), tandis qu'en XML toute balise ouvrante est obligatoirement fermée.

Le code SGML `<A><B></B>` est ambigu car on ne sait pas s'il faut l'interpréter comme :

- `<A></A><B></B>` ou
- `<A><B></B></A>`.

En XML seule l'une des deux formulations non ambiguës est autorisée.

Exemple

- HTML est un langage SGML
- DocBook est à l'origine un langage SGML, porté depuis en XML

Fondamental

XML hérite de SGML en contraignant la syntaxe, tout document XML est donc un document SGML.

Complément : Voir aussi

Documents structurés et documents formatés - p.65

C. Discussion : HTML et XML

Fondamental : XML et HTML ne sont pas au même niveau

XML est un méta-langage et HTML est un langage, XML et HTML ne sont donc pas directement comparables. En particulier XHTML est un langage XML, c'est donc du XML !

XML et le Web

Une question pertinente est en revanche de se demander pourquoi HTML ou XHTML ne suffisent pas pour les usages Web, et donc pourquoi XML est utile en dehors de son instanciation via XHTML.

La réponse est que si HTML, et *a fortiori* XHTML, fournissent une solution de publication efficiente, ils n'offrent pas le même potentiel en terme d'**ingénierie documentaire**. XML possède donc un potentiel de **manipulation** plus important (par exemple pour le multi-supports, la rééditorialisation, ...).

Exemple : Exemple de limites du HTML

Le HTML✯, langage de référence du web qui a su s'imposer par sa simplicité d'écriture, possède également des inconvénients qui ont motivé les recherches du W3C✯ sur le XML✯ :

- **Jeu de balises restreint** : HTML est un langage, les balises sont définies ***a priori*** , il est impossible de définir ses propres balises et de créer un vocabulaire propre à un domaine spécifique. Ce manque d'**extensibilité** le rend mal approprié pour l'indexation de documents variés, le multimédia, ...
- **Mélange des descripteurs physiques et logiques** : Les balises HTML fournissent des informations relatives à la typographie ou la mise en page d'un texte (`CENTER`, `B`, ...) mais également relatives à son rôle dans le document (`H1`, `H2`, ...). Cette non séparation stricte entre contenu et présentation nuit à la publication multi-supports et la rééditorialisation par exemple.
- **Liens** : En HTML, les liens sont généralement des pointeurs directs vers une autre ressource (`A`). Cette mise en œuvre limite fortement les possibilités de réutilisation, notamment par le risque de lien cassé.

Complément : Voir aussi

Documents structurés et documents formatés - p.65

D. Commentaires

Syntaxe

Un fichier XML peut contenir des commentaires, ils peuvent être insérés n'importe où dans le document avec la syntaxe :

```
<!-- Ceci est un commentaire -->
```

Exemple

```
1  <document>
2   Ceci est un contenu.
3   <!-- Ceci est un commentaire -->
4  </document>
```

E. Namespace

Principe

Un *namespace* (ou espace de noms en français) est une mécanique qui permet d'assurer l'unicité des noms des éléments utilisés au sein des fichiers XML, dans l'objectif de pouvoir « mélanger » différents schémas.

- Soit un extrait de schéma S1 qui définit un élément de syntaxe en informatique comme contenant du code : **... {element syntaxe {code {...}}}***
- Soit un extrait de schéma S2 qui définit un élément de syntaxe en mathématique comme contenant une équation : **... {element syntaxe {equation {...}}}***

Si je souhaite, dans un schéma S3 réutiliser les éléments syntaxe issus de S1 et S2, je rencontre un conflit de noms : en effet, deux éléments portant le même nom, « syntaxe », définissent en fait des éléments différents.

Le *namespace* va me permettre de différencier ces deux éléments, en associant un nom unique aux schémas S1 et S2, par exemple une adresse web.

Si j'associe **www.utc.fr/S1** au premier schéma et **www.utc.fr/S2** au second, j'obtiens alors deux noms de balises différents :

- **www.utc.fr/S1:syntaxe**
- **www.utc.fr/S2:syntaxe**

Préfixe

Cette écriture étant quelque peu fastidieuse, il est également possible d'associer un préfixe au namespace. La correspondance entre le namespace et le préfixe est déclarée dans le fichier XML, ce qui permet à un programme informatique de remplacer les préfixes par les namespaces.

Finalement, on obtient :

- Préfixe s1 associé au namespace www.utc.fr/S1 et écriture XML **s1:syntaxe**
- Préfixe s2 associé au namespace www.utc.fr/S2 et écriture XML **s2:syntaxe**

Remarque : Namespace par défaut

Il est possible de définir un namespace par défaut ce qui permet d'avoir un *namespace* pour chaque balise, sans avoir à utiliser de préfixe.

Syntaxe

```
1  <elementRacine
2    xmlns:prefixe1="namespace1"
3    xmlns:prefixe2="namespace2" ...
4    xmlns:prefixeN="namespaceN"
5    xmlns="namespaceDesÉlémentsNonPréfixés"
```

```
6 >
```

Définition : Nom développé

Un **nom développé** (*expanded name*) est le couple constitué par un **nom d'espace de nommage** (*namespace name*) et par un **nom local** (*local name*).

Complément

- *http://www.w3.org/TR/2004/REC-xml-names11-20040204/*[14]
- Traduction : *http://www.yoyodesign.org/doc/w3c/xml-names11/*[15]

F. Syntaxe XML et espaces

Soit les trois syntaxes XML suivantes (les espaces sont symbolisés par des ~ et les retours chariot par des §) :

1. `<a>bonjour~aurevoir</a><b/>`
2. `<a>bonjour~~~~aurevoir</a><b/>`
3. `<a>bonjour§`
   `aurevoir</a>~~~~<b/>`
4. `<a>bonjour~aurevoir</a>§`
   `<b/>`

Ces trois syntaxes sont logiquement équivalentes.

Donc :

- À l'intérieur d'un élément qui ne contient que du texte : 1 espace = N espaces = 1 retour chariot = N retours chariot
- Entre deux éléments : 1 espace = N espaces = 1 retour chariot = N retours chariot = Rien (absence de caractère)

Dans la pratique seuls les cas de *mixed content* ont besoin que les espaces soient préservés. Les deux exemples ci-après ne sont pas équivalents :

- `<p>Ceci est <i>important</i></p>`
- `<p>Ceci est<i>important</i></p>`

Mais sans schéma il est impossible de décider a priori si le contenu d'un élément qui contient des éléments fils et des espaces est *mixed* ou si ces espaces servent juste à la mise en forme du XML.

Espaces conservés

Par défaut, un programme doit considérer que tous les espaces sont signifiants et doivent être conservés lors des traitements.

Remarque : Équivalence entre séparateurs

XML considère comme équivalents les caractères de séparation :

- Espace ' '
- Tabulation \t
- Retour chariot \r
- Fin de ligne \n

14 - http://www.w3.org/TR/2004/REC-xml-names11-20040204/
15 - http://www.yoyodesign.org/doc/w3c/xml-names11/

G. Encodage des caractères

L'encodage des caractères définit l'équivalence entre des caractères et leur valeur numérique en machine.

L'ASCII est l'ancêtre et la base de tous les systèmes d'encodage actuels : il définit par exemple que la lettre A à la valeur 65.

L'encodage universel Unicode UTF-8 est supporté par XML et aujourd'hui la plupart des outils informatiques associés, il est à utiliser dans le cas général.

H. Exemple : Un programme XSLT pour générer du HTML

Exemple : Fichier XML source

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <document titre="XSLT">
3
4  <!--Première division-->
5
6  <div titre="XSLT : Un besoin">
7  <paragraphe>XML est un format de
   <important>représentation</important> de l'information.</paragraphe>
8  <paragraphe>XML n'est pas un format de présentation.</paragraphe>
9  </div>
10
11 <!--Seconde division-->
12
13 <div titre="XSLT : Un langage">
14 <paragraphe>XSLT est un langage de
   <important>manipulation</important> de documents XML.</paragraphe>
15 <paragraphe>XSLT est utilisé pour exporter une source XML sous un
   autre format, par exemple HTML.</paragraphe>
16 </div>
17 </document>
```

Exemple : Fichier HTML cible souhaité

```
1  <HTML>
2
3  <!--Head-->
4
5  <HEAD
6  <TITLE>XSLT</TITLE>
7  <META content="text/html" charset="iso-8859-1"/>
8  </HEAD>
9
10 <!--Body-->
11
12 <BODY>
13 <H1>XSLT : Un besoin</H1>
14 <P>XML est un format de <B>représentation</B> de l'information.</P>
15 <P>XML n'est pas un format de présentation.</P>
16 <H1>XSLT : Un langage</H1>
17 <P>XSLT est un langage de <B>manipulation</B> de documents XML.</P>
18 <P>XSLT est utilisé pour exporter une source XML sous un autre
   format, par exemple HTML</P>
19 </BODY>
20 </HTML>
```

Exemple : Programme XSLT permettant la transformation

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   version="1.0">
3   <xsl:output method="html" indent="yes" encoding="iso-8859-1"/>
4
5  <!--1ère règle-->
6
7   <xsl:template match="document">
8    <HTML>
9     <HEAD
10     <TITLE><xsl:value-of select="@titre"/></TITLE>
11     <META content="text/html" charset="iso-8859-1"/>
12    </HEAD>
13    <BODY>
14     <xsl:apply-templates/>
15    </BODY>
16   </HTML>
17  </xsl:template>
18
19 <!--2nde règle-->
20
21  <xsl:template match="div">
22   <H1><xsl:value-of select="@titre"/></H1>
23   <xsl:apply-templates/>
24  </xsl:template>
25
26 <!--3ème règle-->
27
28  <xsl:template match="paragraphe">
29   <P><xsl:apply-templates/></P>
30  </xsl:template>
31
32 <!--4ème règle-->
33
34  <xsl:template match="important">
35   <B><xsl:value-of select="."/></B>
36  </xsl:template>
37 </xsl:stylesheet>
```

Solution des exercices

> Solution n°1 *(exercice p. 9)*

Exercice

☑	XML a été créé dans la continuité du SGML
☐	XML est une norme ISO
☑	XML est un standard W3C
☑	XML est plus récent que HTML

Exercice

☑	HTML est un langage de mise en forme
☐	XML est un langage de mise en forme
☑	HTML définit un jeu de balises fixé a priori et donc non-extensibles
☑	XML est beaucoup plus générique, il permet de définir des langages

Exercice

☐	Un entête XML
☐	Une référence à une DTD
☐	Une référence à un XML Schema
☑	Une unique balise racine qui encadre toutes les autres
☑	Que chaque balise ouverte soit fermée

Exercice

☐	D'indiquer la mise en page pour un type de document et un format cible.
☑	De vérifier automatiquement qu'un document correspond à une structure type.
☑	De définir une grammaire documentaire.

> Solution n°2 *(exercice p. 10)*

Méta-langage	XML SGML
Format de publication XML	SVG SMIL XHTML Open Document Format
Format de publication SGML	HTML
Langage de structuration documentaire	DocBook
Organisme de standardisation	W3C OASIS

> Solution n°3 *(exercice p. 17)*

☐	Ce fichier n'est pas un fichier XML
☑	Ce fichier est un fichier XML, il possède un élément racine unique qui contient tous les autres. *Il s'agit de `papier`.*
☑	Ce fichier est un fichier XML, tous ses éléments sont inclus dans les uns dans les autres. *Il n'y a pas de balises croisées du type `<a><b></a></b>`.*
☑	Ce fichier est un fichier XML, toutes ses balises ouvertes sont fermées.
☑	Ce fichier est un fichier SGML. *Tous les fichiers XML sont des fichiers SGML, car XML est un sous-ensemble de SGML, c'est à dire que XML est un méta-langage plus restrictif que SGML.*

> Solution n°4 *(exercice p. 18)*

```
<?xml version="1.0"?>
<document>
 <entete>
  <titre>Document à corriger</titre>
  <auteur>Stéphane Crozat</auteur>
  <date>01-03-01</date>
  <version numero="1"/>
</entete>
 <corps>
  <division titre="Première partie">
   <paragraphe>Ce texte doit être <important>corrigé</important></paragraphe> <paragraphe>Le contenu est sans importance ici</paragraphe>
  </division>
  <division titre="Seconde partie">
   <paragraphe>Ai-je le droit de mettre du texte ici ?</paragraphe>
</division>
</corps>
</document>
```

> Solution n°5 *(exercice p. 18)*

```
<docAbstrait>
<contenu>
<paragraphe>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</paragraphe>
<Texte>Nulla erat tellus, molestie a ultrices sed, gravida eget ipsum.
<para>Phasellus lacinia, ipsum vitae interdum tincidunt.
<P>Nullam pulvinar diam et tellus ullamcorper eleifend.</P></para></Texte></contenu>
```

<ligne>Nunc eu lectus in diam tempus adipiscing in rhoncus elit.</ligne></docAbstrait>

Les balises doivent être fermées dans l'ordre dans lequel elles sont ouvertes, pour que chaque élément soit inclus dans son père. La syntaxe XML est sensible à la casse : `P` est différent de `p`.

> Solution n°6 *(exercice p. 25)*

Exemple

```
1  <?xml version="1.0"?>
2  <!DOCTYPE entete SYSTEM "entete.dtd">
3  <entete>
4    <titre>Mon document</titre>
5    <date>Aujourd'hui</date>
6    <auteur>Moi</auteur>
7  </entete>
```

> Solution n°7 *(exercice p. 25)*

Oui.

> Solution n°8 *(exercice p. 25)*

```
1  <?xml version="1.0"?>
2  <!DOCTYPE entete SYSTEM "entete.dtd">
3  <entete>
4    <titre/>
5    <date/>
6    <auteur/>
7  </entete>
```

> Solution n°9 *(exercice p. 25)*

Exemple

```
1  <?xml version="1.0"?>
2  <!DOCTYPE entete SYSTEM "entete.dtd">
3  <entete>
4    <titre>Mon document</titre>
5    <date>Aujourd'hui</date>
6    <auteur>Moi</auteur>
7    <motscles>Document</motscles>
8    <resume>
9      <paragraphe>Mon résumé</paragraphe>
10   </resume>
11 </entete>
```

> Solution n°10 *(exercice p. 33)*

```
1  //titre
2  /papier/titre
3  ../../titre
4  preceding::titre
```

> Solution n°11 *(exercice p. 35)*

```
<!--transf.xsl-->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:output method="text"/>
<xsl:template match="papier">
<xsl:apply-templates select="./motsCles/terme"/>
</xsl:template>
<xsl:template match="terme">
INSERT INTO tMotsCles (terme, titre, url) VALUES (
'<xsl:value-of select="."/>',
'<xsl:value-of select="//titre"/>',
'<xsl:value-of select="//ressource/@uriSrc"/>'
);
</xsl:template>
</xsl:stylesheet>
```

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!--transf.xsl-->
3  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   version="1.0">
4  <xsl:output method="text"/>
5  <xsl:template match="papier">
6  <xsl:apply-templates select="./motsCles/terme"/>
7  </xsl:template>
8  <xsl:template match="terme">
9  INSERT INTO tMotsCles (terme, titre, url) VALUES (
10 '<xsl:value-of select="."/>',
11 '<xsl:value-of select="//titre"/>',
12 '<xsl:value-of select="//ressource/@uriSrc"/>'
13 );
14 </xsl:template>
15 </xsl:stylesheet>
```

Glossaire

Sérialisation

Processus consistant à enregistrer des données en mémoire vive (par exemple des objets) sous une forme permettant leur persistance, typiquement sur une mémoire secondaire.

XSL-FO

XSL-FO (FO pour Formatting Objects) est la seconde partie du standard W3C « Extensible Stylesheet Language Family ». Il propose un langage XML de mise en forme de documents imprimables.

Exemple d'extrait de code FO :

- <fo:block font-family="Times" font-size="12pt">Ceci est un paragraphe en police Times de taille 12pt</fo:block>.

Signification des abréviations

- HTML HyperText Markup Language
- W3C World Wide Web Consortium
- XML eXtensible Markup Language

Bibliographie

[(Dupoirier, 1995)] Gérard Dupoirier, *Technologie de la GED : Techniques et management des documents électroniques*, Hermes, 1995.

[André89] Jacques André, Richard Furuta, Vincent Quint, *Structured documents*, Cambridge University Press, 1989.

[Brillant07] Alexandre Brillant, *XML : Cours et exercices*, Eyrolles, 2007 [ISBN 978-2212126914]

Webographie

[w_w3c.org/TR/2008/REC-xml] W3C, *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, *http://www.w3.org/TR/2008/REC-xml-20081126/* , consulté en 2008.

[w_w3c.org/XML] *XML*, *http://www.w3.org/XML* .

Index

Contenus annexes

- Balises et poignées de calcul

L'ingénierie documentaire met à profit deux thèses complémentaires :

- le contenu est numérisé dans sa forme signifiante : il est manipulable par la machine mais indépendamment du sa signification qui lui reste inaccessible ;
- le contenu est enrichi par des balises qui sont connues syntaxiquement et sémantiquement par la machine ; elle sait quoi en faire.

Fondamental

Le principe du balisage consiste à enrichir un contenu numérisé (dans sa forme sémiotique), sans l'altérer, pour lui ajouter des poignées qui vont être manipulables par l'ordinateur (logiquement).

Remarque

Le contenu est donc interprétable par l'homme et la machine, chacun via ce qui lui est destiné :

- l'humain interprète le contenu signifiant numérisé ;
- la machine interprète les balises ;

Exemple : XML

XML est une illustration de ce principe, puisque l'on va coupler une information sémiotique (texte, image, etc.) destinée à l'interprétation humaine, avec un ensemble de balises qui permettent de décrire formellement une structure documentaire qui sera alors manipulable par le calcul.

- Documents structurés et documents formatés

Définition : Document formaté

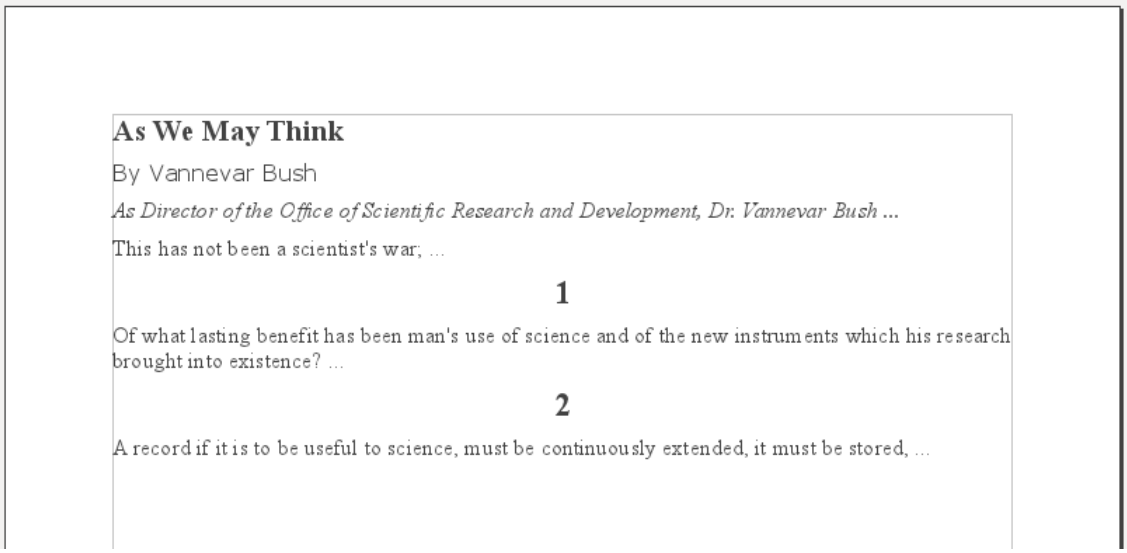
On appelle document formaté un document dont le fichier informatique source décrit la façon de le mettre en forme. C'est la façon la plus courante de traiter avec les documents informatiques, telle qu'elle est mise en œuvre dans les traitements de texte ou sur le Web avec HTML.

Un document formaté nécessite un logiciel capable d'en interpréter le format pour permettre de le lire.

XML est un excellent candidat à la sérialisation de documents formatés.

Exemple : Document formaté (format OpenDocument : extrait simplifié)

```
1  <document-content>
2  <automatic-styles>
3    <style name="P1" family="paragraph"><text-properties font-
   name="Times New Roman" fo:font-size="12pt"/></style>
4    <style name="P2" family="paragraph"><text-properties font-
   name="Times New Roman" fo:font-size="16pt" fo:font-
   weight="bold"/></style>
5    <style name="P3" family="paragraph"><paragraph-properties fo:text-
   align="center"/><text-properties font-name="Times New Roman" fo:font-
   size="18pt" fo:font-weight="bold" /></style>
6    <style name="P4" family="paragraph"><text-properties font-
   name="Times New Roman" fo:font-size="12pt" fo:font-
   style="italic"/></style><style name="T1" family="text"><text-
   properties font-name="Verdana"/></style>
7  </automatic-styles>
8  <body>
9   <text>
10   <p style-name="P2">As We May Think</p>
11   <p style-name="P1"><span style-name="T1">By Vannevar
   Bush</span></p>
12   <p style-name="P4">As Director of the Office of Scientific Research
   and Development, Dr. Vannevar Bush ...</p>
13   <p style-name="P1">This has not been a scientist's war ...</p>
14   <p style-name="P3">1</p>
15   <p style-name="P1">Of what lasting benefit has been man's use of
   science  ...</p>
16   <p style-name="P3">2</p>
17   <p style-name="P1">A record if it is to be useful to
   science  ...</p>
18   </text>
19 </body>
20 </document-content>
```



Visualisation dans OpenOffice.org Writer de l'extrait de l'article "As We May Think"

Définition : Notion de documents structurés

On appelle document structuré un document dont la structure logique est décrite plutôt que la mise en forme physique (Structured documents [André89], p.7).

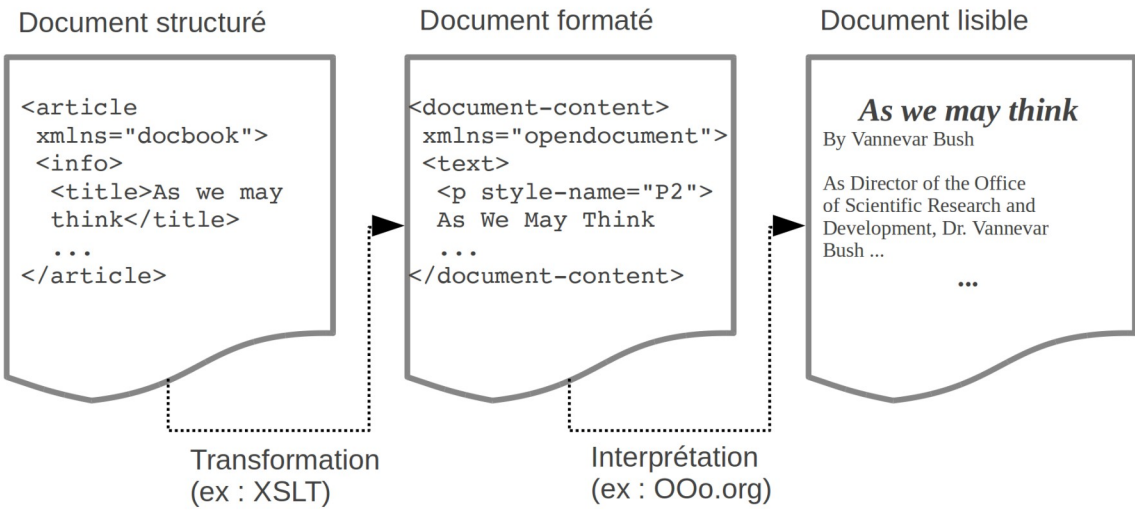
Après SGML qui avait été inventé pour cela, XML est aujourd'hui le candidat quasi-unique pour la réalisation de documents structurés.

Exemple : Document structuré (format DocBook, légèrement simplifiée)

```
1  <article xmlns="http://docbook.org/ns/docbook">
2    <info>
3      <title>As we may think</title>
4      <author><personname>Vannevar Bush</personname></author>
5    </info>
6    <epigraph><para>As Director of the Office of Scientific Research
     and Development, Dr. Vannevar Bush ...</para></epigraph>
7    <para>This has not been a scientist's war ...</para>
8    <sect1>
9      <para>Of what lasting benefit has been man's use of
     science ...</para>
10   </sect1>
11   <sect1>
12     <para>A record if it is to be useful to science  ...</para>
13   </sect1>
14 </article>
```

Définition : Transformation des documents structurés

Un document structuré n'est pas destiné à être directement utilisé pour la lecture humaine, il doit être d'abord transformé dans un document formaté pour être utilisé (visualisé, imprimé, ...).



Processus de transformation d'un document structuré en document formaté

Remarque : Chaîne XML complète

La technologie XML peut en fait être utilisée tout au long de la chaîne :

- Langage XML orienté document structuré en amont (DocBook, DITA, TEI, langage local, ...)
- Langage XML de programmation pour la transformation (XSL-XSLT)
- Langage XML orienté document formaté pour l'aval (OpenDocument, XHTML, ...)

Complément : Voir aussi

La structuration logique - p.68

Complément : Bibliographie

Technologie de la GED [(Dupoirier, 1995)] : Structure logique et structure physique (pp58-61)

- La structuration logique

Un document peut être décrit comme une collection d'objets comportant des objets de plus haut niveau composés d'objets plus primitifs. Les relations entre ces objets représentent les relations logiques entre les composants du document. Par exemple [...] un livre est divisé en chapitres, chaque chapitre en sections, sous-sections, paragraphes, etc. Une telle organisation documentaire est appelée représentation de ***document structuré****. (traduit depuis la préface de* Structured documents*)*

Définition : Structuration logique

On appelle structuration logique d'un contenu une inscription explicitant la structure de ce contenu en fonction de son organisation et des attributs intrinsèques qui le caractérisent et non en fonction de propriétés de présentation sur un support.

Définition : Document abstrait

Un document décrit par sa structure logique est appelé document abstrait, on parle aussi de **document structuré**.

Définition : Structuration physique

On appelle structuration physique ou mise en forme d'un contenu une inscription décrivant la façon dont ce contenu doit être présenté sur un support donné.

Définition : Document formaté

Un document décrit par sa structure physique est appelé document formaté, c'est en général également ce dont on parle quand on parle simplement de document.

Remarque

Il est possible de calculer une ou plusieurs structurations physiques pour une même structuration logique. Il est possible de calculer d'autant plus de structurations physiques que la structuration logique est indépendante de ses supports de présentation.

Remarque

La structuration logique est associée au fond du contenu, tandis que la structuration

physique est associée à sa forme sur un support.

Complément : Voir aussi

Langages XML orienté documents

- Exemple de structuration logique

Exemple : Un exercice structuré logiquement

Soit la structuration logique d'un exercice :

```
1  Exercice = {Enonce, Question, Indice, Solution}
2  avec
3  Enonce = Soit un triangle rectangle disposant d'un angle de 30
   degrés.
4  Question = Donner la valeur des autres angles du triangle.
5  Indice = La somme des angles d'un triangle est égale à 180 degrés.
6  Solution = 90 et 60 degrés.
```

Il est possible à partir de cette représentation de calculer différentes présentations. Pour l'écran on peut générer une présentation HTML, en laissant la solution en hyperlien cliquable. Pour le papier on peut générer une présentation PDF, en affichant la solution sur une page séparée de l'énoncé. Pour un usage multimédia on pourra générer une présentation SMIL, avec affichage de l'énoncé, lecture de la question, et affichage de la solution après un temps de pause.

Notons que si l'on avait choisi une des représentations physiques, plutôt que la représentation logique, il n'aurait pas été possible de générer les autres représentations.

Exemple : Un exercice mis en forme

Soit la mise en forme en HTML du même exercice :

```
1  <HTML>
2  <BODY>
3  Soit un triangle rectangle disposant d'un angle de 30 degrés. </BR>
4  <B> Donner la valeur des autres angles du triangle. </B> </BR>
5  <A HREF="ex001i01.html"> Vous avez besoin d'aide ? </A> </HR>
6  <A HREF="ex001s01.html"> Vérifier votre réponse ! </A>
7  </BODY>
8  </HTML>
```

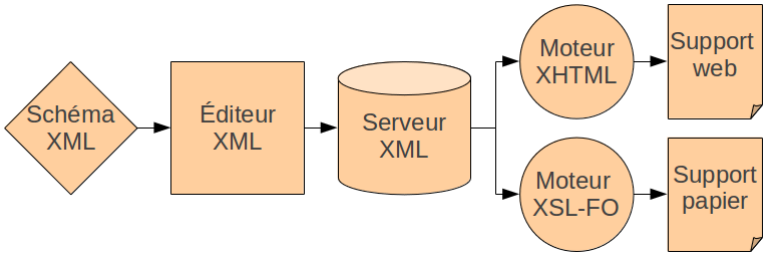
On voit que dans ce format la structure logique n'apparaît plus explicitement et qu'il n'est plus possible d'identifier l'énoncé, la question et la solution sans comprendre le contenu.

Fondamental

L'exemple montre que l'on peut calculer la mise en forme à partir de la structure logique, **mais non l'inverse.**

- Architecture des chaînes éditoriales XML

Architecture classique



Graphique 1 Architecture classique d'une chaîne éditoriale XML

L'approche classique pour réaliser une chaîne éditoriale XML est la suivante :

1. Formalisation du schéma documentaire, avec un langage de modélisation de schéma XML (XML Schema, Relax NG, etc.)
2. Utilisation d'un éditeur XML standard et stylage de cet éditeur (il existe de très nombreux éditeurs XML, plus ou moins graphiques, qui se paramètrent automatiquement lorsqu'on leur fournit un schéma : Oxygen, XMetal, Epic Arbortext, etc.).
3. Utilisation de serveurs de fichiers XML pour la gestion centralisée des contenus (pour les projets les plus importants surtout).
4. Réalisation de moteurs de transformation avec les technologies XSL-XSLT, combinées avec des langages de rendu (XSL-FO pour le papier, XHTML pour l'écran, etc.)

L'ensemble est en général intégré avec un langage applicatif tiers (Java, PHP, etc.).

- Un langage pour publier les documents XML

XML lorsqu'il est utilisé pour définir des formats documentaire métier est un format de **représentation** de l'information, et non un format de **publication** (de présentation) de cette information : donc un tel fichier XML **n'est pas utilisable tel que par un lecteur**.

XML ne peut donc être utilisé pour des langages abstrait que si l'on est capable de transformer les documents sources en document publiés lisibles grâce à un format de présentation : HTML par exemple dans le cas de publication Web, ou PDF pour l'impression.

- Caractéristiques recherchées pour un format XML

Fondamental : Non ambiguïté

Les règles syntaxiques strictes d'XML rendent son interprétation informatique univoque.

Fondamental : Lisibilité

Un format XML a pour objectif d'être lisible par un être humain, afin de plus facilement en appréhender le langage.

Fondamental : Passivité

Un format XML est passif, il dépend de programmes informatiques qui le transforment.