Université Tunis Elmanar

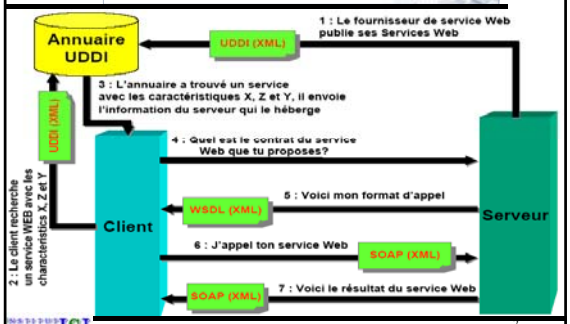
Institut Supérieur d'Informatique

Cours : SOA et services Web

Chapitre 4: Standard services web XML

Présenté par : Sahbi BAHROUN

Services Web : vue générale



Plan

Protocoles utilisés par les services web

- ***XML*: Extensible Markup Language**
- ***WSDL*: Web Services Description Language**
- ***SOAP*: Simple Object Access Protocol**
- ***UDDI*: Universal, Description, Discovery and Integration**

XML

Introduction

- Le langage XML (eXtended Markup Language) est un langage de format de document.
- Il dérive de SGML (Standard Generalized Markup Language) et HTML (HyperText Markup Language).
- Comme ces derniers, il s'agit d'un langage formé de *balises* qui permet de structurer les documents.
- Le langage XML s'est imposé comme le format standard pour les communications entre applications.
- Il est utilisé dans la plupart des projets de publication sur le WEB ainsi que dans les bases de données.

Historique

- 1986 Introduction de SGML par Charles Goldfarb.
- 1991 Introduction de HTML par Tim Berners-Lee pour le WEB.
- 1998 Publication par le W3C de la version 1.0 d'XML.
- 1999 Redéfinition de HTML 4.0 en XHTML à travers XML.
- 2004 Publication par le W3C de la version 1.1 de XML pour une mise à jour des caractères spéciaux en lien avec Unicode.

Intérêts

- Séparation stricte entre contenu et présentation
- Structuration forte du document
- Extensibilité
- Modèles de documents (DTDs et XML-Schémas)
- Validation du document par rapport au modèle
- Format texte avec gestion des caractères spéciaux
- Format libre
- Nombreuses technologies développées autour de XML

XML: syntaxe

Premier exemple

```
<?xml version="1.0" encoding="iso-8859-1" ?>❶
<!-- Time-stamp: "bibliography.xml   3 Mar 2008 16:24:04" -->❷
<!DOCTYPE bibliography SYSTEM "bibliography.dtd" >❸
<bibliography>❹
  <book key="Michard01" lang="fr">
    <title>XML langage et applications</title>
    <author>Alain Michard</author>
    <year>2001</year>
    <publisher>Eyrolles</publisher>
    <isbn>2-212-09206-7</isbn>
    <url>http://www.editions-eyrolles/livres/michard/</url>
  </book>
  <book key="Zeldman03" lang="en">
    <title>Designing with web standards</title>
    <author>Jeffrey Zeldman</author>
    <year>2003</year>
    <publisher>New Riders</publisher>
    <isbn>0-7357-1201-8</isbn>
  </book>
  ...
</bibliography>❺
```


Premier exemple

1. Entête XML avec la version 1.0 et l'encodage iso-8859-1 des caractères.
2. Commentaire délimité par les chaînes de caractères <!-- et -->.
3. Déclaration de DTD externe dans le fichier bibliography.dtd.
4. Balise ouvrante de l'élément racine bibliography
5. Balise fermante de l'élément racine bibliography

Syntaxe et structure

- Pour qu'un document XML soit correct, il doit d'abord être *bien formé* et ensuite être *valide*.
- La première contrainte est de nature *syntaxique*. Un document bien formé doit respecter certaines règles syntaxiques propres à XML. Il s'agit en quelque sorte de l'orthographe d'XML.
- La seconde contrainte est de nature *structurelle*. Un document valide doit suivre un modèle appelé *type* décrit par une DTD (*Document Type Description*) ou un schéma. Une DTD est en fait une grammaire pour XML.

Syntaxe et structure

- Un document XML est généralement contenu dans un fichier texte dont l'extension est .xml.
- Il peut aussi être réparti en plusieurs fichiers en utilisant les entités externes.
- Les fichiers contenant des documents dans un dialecte XML peuvent avoir une autre extension qui précise le format.
- Les extensions pour les schémas XML, les feuilles de style XSL, les dessins en SVG sont par exemple .xsd, .xsl et .svg.

Syntaxe et structure

- Un fichier XML contient du texte dans un format de codage d'Unicode, par exemple UTF-8 ou Latin1. Le codage utilisé par le fichier est précisé dans l'entête du fichier.

Composition globale d'un document

- Un document XML est composé des trois constituants:

1. Prologue: contient des déclarations facultatives.
2. Corps du document: c'est le contenu même du document.
3. Commentaires et instructions de traitement: Ceux-ci peuvent apparaître partout dans le document, dans le prologue et le corps.

Composition globale d'un document

```
<?xml ... ?>              ]   Prologue
  ...                     ]
<root-element>            ]
  ...                     |   Corps
</root-element>           ]
```


Prologue

•Le prologue contient deux déclarations facultatives mais fortement conseillées ainsi que des commentaires et des instructions de traitement.
•La première déclaration est l'entête XML qui précise entre autre la version de XML et le codage du fichier.
•La seconde déclaration est la déclaration du type du document (DTD) qui définit la structure du document.
  •La déclaration de type de document est omise lorsqu'on utilise des schémas XML qui remplacent les DTD.

```
<?xml ... ?>                ]   Entête XML      ]
<!DOCTYPE root-element [    ]                   |   Prologue
  ...                       |   DTD             |
]>                          ]                   ]
```


Prologue: Entête XML

•L'entête utilise une syntaxe <?xml ... ?>

```
<?xml version="..." encoding="..." standalone="..." ?>
```

•Cette entête peut contenir trois attributs version, encoding et standalone.
•Chaque attribut a une valeur délimitée par des apostrophes '" ou des guillemets '".
•L'attribut version précise la version d'XML utilisée. Les valeurs possibles actuellement sont 1.0 ou 1.1.
•L'attribut encoding précise le codage des caractères utilisés dans le fichier. Les principales valeurs possibles sont US-ASCII, ISO-8859-1, UTF-8, et UTF-16.
•L'attribut standalone précise si le fichier est autonome, cad s'il requiert ou non des ressources extérieures. La valeur de cet attribut peut être yes ou no.

Prologue: Entête XML

L'attribut version est obligatoire et l'attribut encoding l'est aussi dès que le codage des caractères n'est pas le codage par défaut UTF-8.

```
<?xml version="1.0"?>
<?xml version='1.0' encoding='UTF-8' ?>
<?xml version="1.0" encoding="iso-8859-1" standalone="no" ?>
```

Prologue: Déclaration de type de document

•La déclaration de type définit la structure du document.

•Elle précise en particulier quels éléments peut contenir chacun des éléments.

•Cette déclaration de type peut prendre plusieurs formes suivant que la définition du type est incluse dans le document ou externe. Elle a la forme générale suivante qui utilise le mot clé DOCTYPE.

```
<!DOCTYPE ... >
```


Corps du document

•Le corps du document est constitué de son contenu qui est organisé de façon hiérarchique

•L'unité de cette organisation est l'*élément*. Chaque élément peut contenir du texte simple, d'autres éléments ou encore un mélange des deux.

•Comme dans une arborescence de fichiers, il y a un élément appelé *élément racine* qui contient l'ensemble du document.

Corps du document: jetons et noms XML

•Les identificateurs sont utilisés en XML pour nommer différents objets comme les éléments, les attributs, les instructions de traitement.
•Ils servent aussi à identifier certains éléments par l'intermédiaire des attributs de type ID.
•XML distinguent deux types d'identificateurs appelés *jetons* (*name token* en anglais abrégé en NMToken) et *noms XML*.
•Les caractères autorisés dans les identificateurs sont tous les caractères alphanumériques, cad les lettres minuscules [a-z], majuscules [A-Z] et les chiffres [0-9] ainsi que le tiret souligné '_', le tiret '-', le point '.' et les deux points ':'.

Corps du document: jetons et noms XML

•Un *jeton* est une suite quelconque de ces caractères qui ne commence pas par les trois lettres xml en minuscule ou majuscule.
→Les identificateurs commençant par ces trois lettres sont réservés aux usages internes de XML.
•Un *nom XML* est un jeton qui, en outre, commence par une lettre [a-zA-Z], le caractère '_' ou le caractère ':'. Les deux caractères '-' et '.' ne peuvent pas apparaître au début des noms. Le caractère ':' est réservé à l'utilisation des espaces de noms.
→De fait, il ne peut apparaître qu'une seule fois pour séparer un préfixe du nom local dans les noms des éléments et des attributs.

Corps du document: Eléments

<name> Contenu de l'élément *name* </name>

•Un *élément* est formé d'une balise ouvrante, d'un contenu et de la balise fermante correspondante.
•La *balise ouvrante* prend la forme <*name*> où *name* est le nom de l'élément et la *balise fermante* prend la forme </*name*>.
•Les noms des éléments XML peuvent être des noms quelconques (inversement à HTML).
•Des attributs peuvent éventuellement être ajoutés dans la balise ouvrante.
•Le *contenu* d'un élément est formé de tout ce qui se trouve entre la balise ouvrante et la balise fermante.

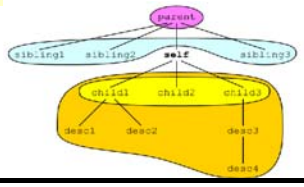
Corps du document: Eléments

<name></name> ou <name/>

Contenu vide

Corps du document: Eléments

```
<parent>
  <sibling1> ... </sibling1>
  <sibling2> ... </sibling2>
  <self>
    <child1> ... <desc1></desc1> ... <desc2></desc2> ... </child1>
    <child2> ... </child2>
    <child3> ... <desc3><desc4> ... </desc4></desc3> ... </child3>
  </self>
  <sibling3> ... </sibling3>
</parent>
```



Corps du document: sections littérales

•Les caractères spéciaux '<', '>' et '&' ne peuvent pas être inclus directement dans le contenu d'un document → Ils peuvent être inclus par l'intermédiaire des sections littérales

•Les *sections littérales*, appelées aussi *sections CDATA* commence par la chaîne de caractères '<![CDATA[' et se termine par la chaîne ']]>'.

•Tous les caractères qui se trouvent entre ces deux chaînes font partie du contenu du document, y compris les caractères spéciaux.

```
<![CDATA[Contenu avec des caractères spéciaux <, > et & ]]>
```


Corps du document: attributs

•Les balises ouvrantes peuvent contenir des *attributs* associés à des valeurs.
•L'association de la valeur à l'attribut prend la forme *attribute*='*value*' où *attribute* et *value* sont respectivement le nom et la valeur de l'attribut.
•Chaque balise ouvrante peut contenir zéro ou plusieurs associations de valeurs à des attributs

```
<tag attribute="value"> ... </tag>
<tag attribute1="value1" attribute2="value2"> ... </tag>
```

```
<body background='yellow'>
<xsd:element name="bibliography" type="Bibliography">
<a href="#{$node/@idref}">
```


Corps du document: attributs

•Lorsque le contenu de l'élément est vide et que la balise ouvrante et la balise fermante sont contractées en une seule balise, celle-ci peut contenir des attributs comme la balise ouvrante.

```
<hr style="color:red; height:15px; width:350px;"
/>
<xsd:attribute name="key" type="xsd:NMTOKEN"
use="required"/>
<xsl:value-of select="key('idchapter',
@idref)/title"/>
```


Corps du document: attributs

•Le nom de chaque attribut doit être un nom XML.
•La valeur d'un attribut peut être une chaîne qq de caractères délimitée par des apostrophes "'" ou des guillemets '"'. Elle ne peut pas contenir les caractères spéciaux '<', '>' et '&'.
•Ces caractères peuvent toutefois être introduits par les entités prédéfinies.
•Si la valeur de l'attribut est délimitée par des apostrophes "'", elle peut contenir des guillemets '"' et inversement.

Corps du document: attributs

•C'est une question de style de mettre les données dans les attributs ou dans les contenus des éléments. Le nom complet d'un individu peut, par exemple, être réparti entre des éléments firstname et surname regroupés dans un élément personname comme dans l'exemple ci-dessous.

```
<personname id="I666">
        <firstname>Gaston</firstname>
        <surname>Lagaffe</surname>
</personname>
```

•Les éléments firstname et surname peuvent être remplacés par des attributs de l'élément personname

```
<personname id="I666" firstname="Gaston" surname="Lagaffe"/>
```

Corps du document: attributs

•C'est une question de style de mettre les données dans les attributs ou dans les contenus des éléments. Le nom complet d'un individu peut, par exemple, être réparti entre des éléments firstname et surname regroupés dans un élément personname comme dans l'exemple ci-dessous.

```
<personname id="I666">
        <firstname>Gaston</firstname>
        <surname>Lagaffe</surname>
</personname>
```


Corps du document: attributs particuliers

•Quatre attributs particuliers xml:lang, xml:space, xml:base et xml:id font partie de l'espace de noms XML.
•Lors de l'utilisation de schémas, ces attributs peuvent être déclarés dans le schéma à l'adresse http://www.w3.org/2001/xml.xsd.

•xml:lang
• utilisé pour décrire la langue du contenu de l'élément.
•Sa valeur est un code de langue sur deux ou trois lettres de la norme ISO 639 (comme par exemple en, fr, es, de, it, pt, …).

```
<p xml:lang="fr">Bonjour</p>
<p xml:lang="en-GB">Hello</p>
<p xml:lang="en-US">Hi</p>
```


Corps du document: attributs particuliers

xml:space
•indique à une application le traitement des espaces. Les deux valeurs possibles de cet attribut sont default et preserve.
•Les caractères d'espacement sont l'espace ' ' de code #x20, la tabulation de code #x9 ('\t' en notation du langage C), le saut de ligne de code #xA ('\n' en C) et le retour chariot de code #xD ('\r' en C).
•Les retours à la ligne sont normalisés par l'analyseur lexical. Ceci signifie que les différentes combinaisons de fin de ligne sont remplacées par un seul caractère #xA.
•Si l'attribut xml:space a la valeur preserve, l'application doit respecter les caractères d'espacement. Les retours à la ligne sont préservés et les espaces consécutifs ne sont pas confondus.

Corps du document: attributs particuliers

xml:base
•À chaque élément d'un document XML est associée une URL appelée *URL de base*. Celle-ci est utilisée pour résoudre les URL des entités externes, qui peuvent être, par exemple des fichiers XML ou des fichiers multimédia (images, sons, vidéo).

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
<book xml:base="http://www.somewhere.org/Enseignement/index.html"> 1
    <chapter xml:base="XML/chapter.html"> 2
        <section xml:base="XPath/section.html"/> 3
        <section xml:base="/Course/section.html"/> 4
        <section xml:base="http://www.elsewhere.org/section.html"/> 5
    </chapter>
</book>
```

1. http://www.somewhere.org/Enseignement/index.html
2. http://www.somewhere.org/Enseignement/XML/chapter.html
3. http://www.somewhere.org/Enseignement/XML/XPath/section.html
4. http://www.somewhere.org/Course/section.html
5. http://www.elsewhere.org/section.html

Corps du document: élément racine

•Tout le corps du document doit être compris dans le contenu d'un unique élément appelé *élément racine*.
•Le nom de cet élément racine est donné par la DTD si celle-ci est présente.

Corps du document: commentaire

•Les commentaires sont délimités par les chaînes de caractères '<!--' et '-->'.
• Ils ne peuvent pas contenir la chaîne '--' formée de deux tirets '-' et ils ne peuvent donc pas être imbriqués.
•Ils peuvent être présents dans le prologue et en particulier dans la DTD.
•Ils peuvent aussi se situer dans le contenu de n'importe quel élément et après l'élément racine.

Corps du document: commentaire

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!-- Commentaire dans le prologue avant la DTD -->
<!DOCTYPE simple [
    <!-- Commentaire dans la DTD -->
    <!ELEMENT simple (#PCDATA) >
]>
<!-- Commentaire entre le prologue et le corps -->
<simple>
    <!-- Commentaire au début du contenu de l'élément simple -->
    Un exemple simplissime <!-- Commentaire à la fin du contenu de l'élément simple --> </simple>
<!-- Commentaire après le corps -->
```


Corps du document: instruction de traitement

•Les ***instructions de traitement*** sont destinées aux applications qui traitent les documents XML
•Elles sont délimitées par les chaînes de caractères '<?' et '?>'. Les deux caractères '<?' sont immédiatement suivis du *nom XML* de l'instruction. Le nom de l'instruction est ensuite suivi du *contenu*. Ce contenu est une chaîne quelconque de caractères ne contenant pas la chaîne '?>' utilisée par l'analyseur lexical pour déterminer la fin de l'instruction.
•Le nom de l'instruction permet à l'application de déterminer si l'instruction lui est destinée.

```
<?dbhtml filename="index.html"?>
```

→ Cette instruction indique le nom du fichier cible à utiliser par les feuilles de styles pour la conversion en HTML.

Exemples

L'exemple contient un prologue avec la déclaration XML et un élément de contenu vide. Les balises <tag> et </tag> ont été contractées en une seule balise <tag/>. Ce document n'a pas de déclaration de DTD.

```
<?xml version="1.0" ?>
<tag/>
```

Exemple simple avec une DTD
L'exemple contient une déclaration de DTD qui permet de valider le document. Cette DTD définit l'élément simple et déclare que son contenu doit être textuel.

```
<?xml version="1.0" encoding="ISO-8859-1"  standalone="yes" ?> <!DOCTYPE simple [
   <!ELEMENT simple (#PCDATA)
]>
<simple>Un exemple simplissime</simple>
```

39

XML: DTD

DTD (Document Type Definition)

•définir précisément la structure d'un document.
•Il s'agit d'un certain nombre de contraintes que doit respecter un document pour être *valide*.
•Ces contraintes spécifient quelles sont les éléments qui peuvent apparaître dans le contenu d'un élément, l'ordre éventuel de ces éléments et la présence de texte brut.
•Elles définissent aussi, pour chaque élément, les attributs autorisés et les attributs obligatoires.
•Les DTD ont l'avantage d'être relativement simples à utiliser mais elles sont parfois aussi un peu limitées → *Les schémas permettent de décrire de façon plus précise encore la structure d'un document MAIS ils sont plus sophistiqués et plus difficiles à manipuler.*

Un premier exemple

```
<!DOCTYPE bibliography SYSTEM "bibliography.dtd" >
<!ELEMENT bibliography (book)+ > 1
<!ELEMENT book (title, author, year, publisher, isbn, url?)
>2 <!ATTLIST book key NMTOKEN #REQUIRED > 3
<!ATTLIST book lang (fr | en) #REQUIRED > 4
<!ELEMENT title (#PCDATA) > 5
<!ELEMENT author (#PCDATA) >
<!ELEMENT year (#PCDATA) >
<!ELEMENT publisher (#PCDATA) >
<!ELEMENT isbn (#PCDATA) >
<!ELEMENT url (#PCDATA) >
```

Un premier exemple

1. Déclaration de l'élément bibliography devant contenir une suite non vide d'éléments book.
2. Déclaration de l'élément book devant contenir les éléments title, author, …, isbn et url.

3, 4. Déclarations des attributs obligatoires key et lang de l'élément book.

5. Déclaration de l'élément title devant contenir uniquement du texte.

Déclaration de la DTD

•la DTD doit être placée dans le prologue.
•La DTD peut être interne, externe ou mixte.
•Elle est *interne* si elle est directement incluse dans le document.
•Elle est *externe* si le document contient seulement une référence vers un autre document contenant la DTD.
•Elle est mixte si elle est constituée d'une partie interne et d'une partie externe.
•Une DTD est utilisée pour de multiples documents → Elle est alors utilisée comme DTD externe.

```
<!DOCTYPE root-element ... >
```


DTD interne

•Elle est incluse dans le document
•La déclaration '[' et ']'.

```
<!DOCTYPE root-element [ declarations ] >
```

```
<!DOCTYPE simple [
<!ELEMENT simple (#PCDATA) >
]>
```


DTD externe

•Elle est contenue dans un fichier .dtd

Adressée par URL
La référence à une URL est introduite par le mot clé SYSTEM suivi de l'URL délimité par des apostrophes "' ou des guillemets "".

```
<!DOCTYPE root-element SYSTEM "url" >
<!DOCTYPE bibliography SYSTEM "bibliography.dtd">
```


DTD externe

Adressée par FPI (Formal Public Identifier)
La référence à un FPI est introduite par le mot clé PUBLIC suivi du FPI et d'une URL délimitée par des apostrophes "' ou des guillemets "". L'URL est utilisée dans le cas où le FPI ne permet pas à l'application de retrouver la DTD.

```
<!DOCTYPE root-element PUBLIC "fpi" "url" >
```

47

```
<!DOCTYPE html PUBLIC "-//W3C/DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd" >
```


DTD mixte

•Il est possible d'avoir une DTD externe adressée par URL ou FPI et des déclarations internes.
•Les déclarations internes peuvent alors redéfinir des éléments ou des attributs déjà définis dans la DTD externe.
•La déclaration prend alors une des deux formes suivantes:
On retrouve un mélange de la syntaxe des DTD externes avec les mots clés SYSTEM et PUBLIC et de la syntaxe des DTD internes avec des déclarations encadrées par les caractères '[' et ']'.

DTD mixte

```
<!DOCTYPE root-element SYSTEM "url" [ declarations ] >
<!DOCTYPE root-element PUBLIC "fpi" "url" [ declarations ] >
```


Contenu de la DTD: déclaration d'entité

•Une ***entité*** est un nom donné à un fragment de document.

•Ce fragment peut être inséré dans le document en utilisant le nom de l'entité.

•Il s'agit d'un mécanisme d'abréviation→ Si l'entité a pour nom *entity*, le fragment est inséré par **&*entity*;** où le nom de l'entité est encadré des caractères '&' et ';'.

•L'entité peut être utilisée dans le contenu des éléments et dans les valeurs des attributs

```
<tag meta="attribute: &entity;">Content: &entity;</tag>
```


Déclaration d'entité: entité prédéfinie

•Il existe des entités prédéfinies permettant d'inclure les caractères spéciaux '<', '>', '&', '"' et ''' dans les contenus d'éléments et dans les valeurs d'attributs.

•Ces entités sont indispensables car ces caractères spéciaux ne peuvent pas apparaître directement dans le contenu du document.

Entité	Caractère
&lt;	<
&gt;	>
&amp;	&
&quot;	"
&apos;	'

Déclaration d'entité: entité interne

•Une entité est dite *interne* lorsque le fragment est inclus directement dans le document.

•La déclaration prend la forme suivante où l'identifiant *entity* est le nom l'entité et *fragment* est la valeur de l'entité.

•Cette valeur doit être un fragment XML bien formé. Elle peut contenir des caractères et des éléments.

Déclaration d'entité: entité interne

```
<!ENTITY entity "fragment" >
```

```
<!ENTITY aka "also known as" >
<!ENTITY euro "&#20AC;" >
```

```
<!DOCTYPE book [
  <!-- Entités -->
  <!ENTITY jmh "James Marshall Hendrix &aka; 'Jimi
Hendrix'" >
  <!ENTITY aka "also known as" >
]>
  <book>&jmh;</book>
```


Déclaration d'entité: entité externe

•Une entité peut désigner une fraction de document contenu dans un autre fichier.

•Ce mécanisme permet de répartir un même document sur plusieurs fichiers.

•La déclaration utilise alors le mot clé **SYSTEM** suivi d'une URL qui peut simplement être le nom d'un fichier local.

Déclaration d'entité: entité externe

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE book [
  <!-- Entités externes -->
  <!ENTITY chapter1 SYSTEM "chapter1.xml" >
  <!ENTITY chapter2 SYSTEM "chapter2.xml" >
]>
<book> &chapter1; &chapter2; </book>
```


Déclaration d'élément

•La déclaration d'un élément est nécessaire pour qu'il puisse apparaître dans un document.

•Cette déclaration précise le nom et le type de l'élément. Le nom de l'élément doit être un nom XML et le type détermine les contenus valides de l'élément.

•On distingue les *contenus purs* uniquement constitués d'autres éléments, les *contenus textuels* uniquement constitués de texte et les *contenus mixtes* qui mélangent éléments et texte.

•La déclaration d'un élément prend la forme: *element* et *type*.

<!ELEMENT *element type* >

Déclaration d'élément: contenu pur d'éléments

•Contient d'autres éléments.
•Ces éléments fils peuvent, à leur tour, contenir d'autres éléments et/ou du texte.
•Leur contenu est spécifié par leur propre déclaration dans la DTD.
•La déclaration de l'élément détermine quels éléments il peut contenir directement et dans quel ordre.

<!ELEMENT *element regexp* >

•Le nom de l'élément est donné par l'identifiant *element*
•*regexp* décrit les suites autorisées d'éléments dans le contenu de l'élément. Elle est construite à partir des noms d'éléments en utilisant les opérateurs ',', '|', '?', '*' et '+' ainsi que les parenthèses '(' et ')' pour former des groupes.

Déclaration d'élément: contenu pur d'éléments

Opérateur	Signification
,	Mise en séquence
|	Choix
?	0 ou 1 occurrence
*	Itération (nombre quelconque d'occurrences)
+	Itération stricte (nombre non nul d'occurrences)

Déclaration d'élément: contenu pur d'éléments

<!ELEMENT elem (elem1, elem2, elem3) >
L'élément elem doit contenir un élément elem1, un élément elem2 puis un élément elem3 dans cet ordre.

<!ELEMENT elem (elem1 | elem2 | elem3) >
L'élément elem doit contenir un seul des éléments elem1, elem2 ou elem3.

<!ELEMENT elem (elem1, elem2?, elem3) >
L'élément elem doit contenir un élément elem1, un ou zéro élément elem2 puis un élément elem3 dans cet ordre.

Déclaration d'élément: contenu pur d'éléments

<!ELEMENT elem (elem1, elem2*, elem3) >
L'élément elem doit contenir un élément elem1, une suite éventuellement vide d'éléments elem2 et un élément elem3 dans cet ordre.

<!ELEMENT elem (elem1, (elem2 | elem4), elem3) >
L'élément elem doit contenir un élément elem1, un élément elem2 ou un élément elem4 puis un élément elem3 dans cet ordre.

Déclaration d'élément: contenu pur d'éléments

<!ELEMENT elem (elem1, elem2, elem3)* >
L'élément elem doit contenir une suite d'éléments elem1, elem2, elem3, elem1, elem2, … jusqu'à un élément elem3.

<!ELEMENT elem (elem1 | elem2 | elem3)* >
L'élément elem doit contenir une suite quelconque d'éléments elem1, elem2 ou elem3.

<!ELEMENT elem (elem1 | elem2 | elem3)+ >
L'élément elem doit contenir une suite non vide d'éléments elem1, elem2 ou elem3.

Déclaration d'élément: contenu textuel

•Indique qu'un élément peut uniquement contenir du texte.

```
<!ELEMENT element (#PCDATA) >
```

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE texts [
<!ELEMENT texts (text)* >
<!ELEMENT text (#PCDATA) >
]>
<texts>
   <text>Du texte simple</text>
   <text>Une <![CDATA[ Section CDATA avec < et > ]]></text>
   <text>Des entités &lt; et &gt;</text>
</texts>
```


Déclaration d'élément: contenu mixte

•Indique qu'un élément peut uniquement contenir du texte et des éléments element1, …, elementN.

•Il n'y a aucun contrôle sur le nombre d'occurrences de chacun des éléments et sur leur ordre d'apparition dans le contenu de l'élément ainsi déclaré.

•Dans une telle déclaration, le mot clé #PCDATA doit apparaître en premier avant tous les noms des éléments.

Déclaration d'élément: contenu mixte

```
<!ELEMENT element (#PCDATA | element1 | ... | elementN)* >
```

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE book [
<!ELEMENT book (#PCDATA | em | cite)* >
<!ELEMENT em (#PCDATA) >
<!ELEMENT cite (#PCDATA) >
]>

<book>
Du <em>texte</em>, une <cite>citation</cite> et encore du
<em>texte</em>. </book>
```


Déclaration d'élément: contenu vide

```
<!ELEMENT element EMPTY >
```


Déclaration d'attribut

•La déclaration d'attribut prend la forme: *attribut* est le nom de l'attribut et *element* le nom de l'élément auquel il appartient.
•Cette déclaration comprend également le type *type* et la valeur par défaut *default* de l'attribut.

```
<!ATTLIST element attribut type default >
```

```
<!ATTLIST element attribut1 type1 default1
                  attribut2 type2 default2 ...
                  attributN typeN defaultN >
```

Déclaration d'attribut: type des attributs

•**CDATA:** Ce type est le plus général. Il n'impose aucune contrainte à la valeur de l'attribut. Celle-ci peut être une chaîne quelconque de caractères.
•**Liste (*value1* | *value2* | ... | *valueN*) de jetons:** La valeur de l'attribut doit être une des valeurs *value1*, *value2*, … ou *valueN*.
•**NMTOKEN:** La valeur de l'attribut est un jeton.
•**NMTOKENS:** La valeur de l'attribut est une liste de jetons séparés par des espaces.

Déclaration d'attribut: type des attributs

•**ID:** La valeur de l'attribut est un nom XML. Un élément peut avoir un seul attribut de ce type.
•**IDREF:** La valeur de l'attribut est une référence à un élément identifié par la valeur de son attribut de type ID.
•**IDREFS:** La valeur de l'attribut est une liste de références séparées par des espaces.

Déclaration d'attribut: type des attributs

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
<!DOCTYPE book [
<!ELEMENT book (section)* >
<!ELEMENT section (#PCDATA | ref  | refs)* >
<!ATTLIST section id ID #IMPLIED>
<!ELEMENT ref EMPTY >
<!ATTLIST ref idref IDREF #REQUIRED >
<!ELEMENT refs EMPTY >
<!ATTLIST refs idrefs IDREFS #REQUIRED >
]>
<book>
    <section id="sec0">Une référence <ref idref="sec1"/></section>
    <section id="sec1">Des références <refs idrefs="sec0 sec2"/></section>
    <section id="sec2">Section sans référence</section>
    <section id="sec3">Une auto-référence <refs idrefs="sec3"/></section>
</book>
```


Déclaration d'attribut: valeur par défaut

•"*value*" où *value* est une chaîne quelconque de caractères délimitée par '" ou '"' Si l'attribut est absent pour un élément du document, sa valeur est implicitement la valeur *value*.
•**#IMPLIED:** L'attribut est *optionnel* et il n'a pas de valeur par défaut. Si l'attribut est absent, il n'a pas de valeur.
•**#REQUIRED:** L'attribut est *obligatoire* et il n'a pas de valeur par défaut.
•**#FIXED "*value*" :** où *value* est une chaîne quelconque de caractères délimitée par '" ou '"' La valeur de l'attribut est fixée à la chaîne de caractères *value*. Si l'attribut est absent, sa valeur est implicitement *value*. Si l'attribut est présent, sa valeur doit être *value* pour que le document soit valide.

Déclaration d'attribut: exemple

<!ATTLIST tag meta CDATA "default" > : La valeur de l'attribut meta peut être une chaîne quelconque et sa valeur par défaut est la chaîne default.

<tag meta="Hello World!">	attribut égal à la chaîne Hello World!
<tag>	attribut égal à la valeur par défaut default
<tag meta="">	attribut égal à la chaîne vide
<tag meta="=='"==">	attribut égal à la chaîne =="==
<tag meta='=='"=='>	attribut égal à la chaîne =="==
<tag meta="==<&>==">	attribut égal à la chaîne ==<&>==
<tag meta="==<&&#62==">	attribut égal à la chaîne ==<&>==

Déclaration d'attribut: exemple

<!ATTLIST book name NMTOKEN #IMPLIED > L'attribut name est optionnel et sa valeur doit être un jeton. Il n'a pas de valeur par défaut.

<book>	attribut absent et sans valeur
<book name="en">	attribut égal au jeton en
<book name="-id234"/>	attribut égal au jeton -id234
<book name="Hello World!">	*non valide* car Hello World! n'est pas un jeton

<!ATTLIST entry id ID #REQUIRED > L'attribut id est obligatoire et sa valeur doit être un nom unique. Il n'a pas de valeur par défaut.

<entry>	*non valide* car l'attribut obligatoire est absent
<entry id="id-234">	attribut égal au nom id-234
<entry id="Hello World!">	*non valide* car Hello World! n'est pas un nom

XML: Espace de noms

Introduction

Les *espaces de noms* ont été introduits en XML afin de pouvoir mélanger plusieurs vocabulaires au sein d'un même document. De nombreux dialectes XML ont été définis pour des utilisations diverses et il est préférable de les réutiliser au maximum. Il est, en effet, fastidieux de redéfinir plusieurs fois les mêmes vocabulaires. Le recyclage des dialectes fait d'ailleurs partie des objectifs de XML.

Le mélange de plusieurs vocabulaires au sein d'un même document ne doit pas empêcher la validation de celui-ci. Il devient indispensable d'identifier la provenance de chaque élément et de chaque attribut afin de le valider correctement. Les espaces de noms jouent justement ce rôle. Chaque élément ou attribut appartient à un espace de noms qui détermine le vocabulaire dont il est issu. Cette appartenance est marquée par la présence dans le nom d'un préfixe associé à l'espace de noms.

Le mélange de plusieurs vocabulaires est illustré par l'exemple suivant. Afin d'insérer des métadonnées dans des documents, il est nécessaire de disposer d'éléments pour présenter celles-ci. Il existe déjà un standard, appelé Dublin Core, pour organiser ces métadonnées. Il comprend une quinzaine d'éléments dont title, creator, subject et date qui permettent de décrire les caractéristiques principales d'un document. Il est préférable d'utiliser le vocabulaire Dublin Core, qui est un standard international, plutôt que d'introduire un nouveau vocabulaire. Le document suivant est le document principal d'un livre au format DocBook. Les métadonnées sont contenues dans un élément metadata. Celui-ci contient plusieurs éléments du Dublin Core dont les noms commencent par le préfixe dc. L'élément include de XInclude fait partie d'un autre espace de noms marqué par le préfixe xi.

Identification d'un espace de noms

•Un espace de noms permet d'utiliser simultanément des éléments de même nom mais définis dans des modèles différents.

•Un espace de noms est identifié par une URL appelée *URL de l'espace de noms*. Il est sans importance que l'URL pointe réellement sur un document. Cette URL garantit seulement que l'espace de noms est identifié de manière unique.

•Dans la pratique, l'URL permet aussi souvent d'accéder à un document qui décrit l'espace de noms.

Identification d'un espace de noms

- Liberté de choix des noms de balises et des attributs XML
  - Conflits et polysémie entre ces noms/attributs
- Besoin d'associer plusieurs applications dans un même document
  - « Préfixage » des noms de balises par l'URI de l'application concernée

Déclaration d'un espace de noms

•Un espace de noms déclaré par **xmlns:*prefix*** dont la valeur est une URL qui identifie l'espace de noms.

•Le préfixe *prefix* est un nom XML ne contenant pas le caractère ':'. Il est ensuite utilisé pour *qualifier* les noms d'éléments. Le choix du préfixe est complètement arbitraire. Dans l'exemple précédent, on aurait pu utiliser foo ou bar à la place du préfixe html.

•Un *nom qualifié* d'élément prend la forme ***prefix:local*** où *prefix* est un préfixe associé à un espace de noms et *local* est le *nom local* de l'élément.

Déclaration d'un espace de noms

```
<?xml version="1.0" encoding="iso-8859-1"?>
<book version="5.0" xml:lang="fr"
      xmlns="http://docbook.org/ns/docbook"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:xi="http://www.w3.org/2001/XInclude">

  <!-- Titre DocBook du document -->
  <title>Langages formels, calculabilité et complexité</title>

  <!-- Métadonnées -->
  <metadata>
    <dc:title>Langages formels, calculabilité et complexité</dc:title>
    <dc:creator>Olivier Carton</dc:creator>
    <dc:date>2008-10-01</dc:date>
    <dc:identifier>urn:isbn:978-2-7117-2077-4</dc:identifier>
  </metadata>

  <!-- Import des chapitres avec XInclude -->
  <xi:include href="introduction.xml"  parse="xml"/>
  <xi:include href="chapter1.xml"      parse="xml"/>
  <xi:include href="chapter2.xml"      parse="xml"/>
  <xi:include href="chapter3.xml"      parse="xml"/>
  <xi:include href="chapter4.xml"      parse="xml"/>
  <index/>
</book>
```

4.1. Introduction

Déclaration d'un espace de noms

```
<?xml version="1.0"?>
<CV xmlns="http://www.univ-lyon1.fr/etds/CV/english"
    xmlns:xhtml="http://www.w3.org/1999/xhtml">
  <personne>
    <civil_status>
      <title>Mr.</title>
    </civil-status>
    ...
  </personne>
  <xhtml:html>
    <xhtml:head>
      <xhtml:title>CV of a student</xhtml:title>
    </xhtml:head>
    <xhtml:body>
    ...
    </xhtml:body>
  </xhtml:html>
</CV>
```


Déclaration d'un espace de noms

```
<html:html xmlns:html="http://www.w3.org/1999/xhtml">
  <html:head>
      <html:title>Espaces de noms</html:title>
  </html:head>
  <html:body>
     ...
  </html:body>
</html:html>
```


Déclaration d'un espace de noms

Il est bien sûr possible de déclarer plusieurs espaces de noms en utilisant plusieurs attributs de la forme **xmlns:*prefix***. Dans l'exemple suivant, on déclare également l'espace de noms de MathML et on l'associe au préfixe **mml**.

```
<html:html xmlns:html="http://www.w3.org/1999/xhtml"
        xmlns:mml="http://www.w3.org/1998/Math/MathML">
        <html:head>
            <html:title>Espaces de noms</html:title>
        </html:head>
        <html:body>
                ...
                <mml:math>
                        <mml:apply>
                        <mml:eq/> ... </mml:apply>
                </mml:math>
                        ...
                </html:body>
        </html:html>
```


Déclaration d'un espace de noms

```
<name xmlns:foo="http://www.somewhere.org/uri"
        xmlns:bar="http://www.somewhere.org/uri">
        <!-- Les deux éléments firstname et surname appartiennent
au même espace de noms. -->
        <foo:firstname>Gaston<foo:firstname>
        <bar:surname>Lagaffe<bar:surname> </name>
```


Espace de noms par défaut

- Il existe un ***espace de noms par défaut*** associé au préfixe vide.
- Son utilisation permet d'alléger l'écriture des documents XML en évitant de mettre un préfixe au éléments les plus fréquents.

- L'espace de noms par défaut peut être spécifié par un pseudo attribut de nom **xmlns** dont la valeur est l'URL de l'espace de noms.

Espace de noms par défaut

```
<html xmlns="http://www.w3.org/1999/xhtml">
        <head>
                <title>Espaces de noms</title>
        </head> <body>
                        ...
                <mml:math
xmlns:mml="http://www.w3.org/1998/Math/MathML">
        <mml:apply>
                <mml:eq/>
                        ...
                </mml:apply>
                </mml:math> ...
                </body>
        </html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <!-- L'espace de noms par défaut est spécifié -->
  <!-- Tous les éléments html, head, title, body, ... appartiennent à l'espace de
noms par défaut. -->
   <head>
      <title>
            Espaces de noms
      </title>
   </head>
   <body>
            ...
            <name xmlns="">
               <!-- L'espace de noms par défaut n'est plus spécifié -->
               <!-- Les trois éléments name, firstname et surname n'appartiennent à
aucun espace de noms. -->
               <firstname>Gaston<firstname>
               <surname>Lagaffe<surname>
            </name>
            ...
   </body>
```


Attributs

- Les attributs peuvent avoir des noms qualifiés.
- Ils font partie de l'espace de noms donné par le préfixe.
- L'attribut n**oNamespaceSchemaLocation** fait partie de l'espace de noms des instances de schémas identifié par l'URL http://www.w3.org/2001/XMLSchema-instance.
- Le nom de l'attribut noNamespaceSchemaLocation doit donc avoir un préfixe associé à cette URL.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<bibliography xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
        xsi:noNamespaceSchemaLocation="bibliography.xsd">
```


Espace de noms XML

- Le préfixe xml est implicitement lié à l'espace de noms XML dont l'URL est http://www.w3.org/XML/1998/namespace.

- Cet espace de noms n'a pas besoin d'être déclaré.

XML: Schémas XML

Introduction

- Les *schémas XML* permettent comme les DTD de définir des modèles de documents.

Inconvénients des DTD

- Syntaxe non XML
- Manque de concision dans les descriptions des contenus en particulier dans les éléments de contenu mixte.
- Modularité très limitée
- Pas de gestion des espaces de noms

Apports des schémas XML

- Syntaxe XML
- Nombreux types de données prédéfinis (nombres, dates, …)
- Possibilité de définir de nouveaux types
- Prise en compte des espaces de noms


```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"> 1
    <xsd:annotation> 2
        <xsd:documentation xml:lang="fr">
                Schéma XML pour bibliography.xml
        </xsd:documentation>
    </xsd:annotation>
    <xsd:element name="bibliography" type="Bibliography"/> 3
    <xsd:complexType name="Bibliography"> 4
        <xsd:sequence>
                <xsd:element name="book" minOccurs="1"
                maxOccurs="unbounded"> 5
                <xsd:complexType>
        <xsd:sequence>
                <xsd:element name="title" type="xsd:string"/>
                <xsd:element name="author" type="xsd:string"/>
                <xsd:element name="year" type="xsd:string"/>
                <xsd:element name="publisher" type="xsd:string"/>
                <xsd:element name="isbn" type="xsd:string"/>
                <xsd:element name="url" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
        ……..</xsd:schema>
```

1. Élément racine xsd:schema avec la déclaration de l'espace de noms des schémas associé au préfixe xsd.
2. Documentation du schéma.
3. Déclaration de l'élément bibliography avec le type Bibliography.
4. Début de la définition du type Bibliography.
5. Déclaration de l'élément book dans le contenu du type Bibliography.

Structure globale d'un schéma

•Un schéma XML se compose essentiellement de déclarations d'éléments et d'attributs et de définitions de types.
•Chaque élément est déclaré avec un type qui peut être, soit un des types prédéfinis, soit un nouveau type défini dans le schéma.
•Le type spécifie quels sont les contenus valides de l'éléments ainsi que ses attributs.
•Un nouveau type est obtenu soit par *construction*, cad une description explicite des contenus qu'il autorise, soit par *dérivation*, cad modification d'un autre type.
•Un schéma peut aussi contenir des imports d'autres schémas.
•L'espace de noms des schémas XML est identifié par l'URL http://www.w3.org/2001/XMLSchema. Il est associé au préfixe xsd ou à xs.
•Tout le schéma est inclus dans l'élément xsd:schema.

Structure globale d'un schéma

```
<?xml version="1.0" encoding="iso-8859-1"?>
        <xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
                <!-- Déclarations d'éléments, d'attributs et définitions
de types -->
                        ...
</xsd:schema>
```


Attributs de l'élément xsd:schema

L'élément racine schema peut avoir les attributs suivants:
•targetNamespace: La valeur est l'URI qui identifie l'espace de noms cible, cad l'espace de noms des éléments et types définis par le schéma.

•elementFormDefault et attributeFormDefault: la valeur par défaut de l'attribut form pour respectivement les éléments et les attributs. Les valeurs possibles sont qualified et unqualified. La valeur par défaut est unqualified.

•blockDefault et finalDefault: donnent la valeur par défaut des attributs block et final.

Attributs de l'élément xsd:schema

```
<?xml version="1.0" encoding="iso-8859-1"?>
  <xsd:schema
targetNamespace="http://www.liafa.jussieu.fr/~carton
   elementFormDefault="qualified"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://www.liafa.jussieu.fr/~carton"> ...
```


Référence explicite à un schéma

•Il est possible dans un document de donner explicitement le schéma devant servir à le valider.
•On utilise un des attributs schemaLocation ou noNamespaceSchemaLocation dans l'élément racine du document à valider.
•Ces deux attributs se trouvent dans l'espace de noms des instances de schémas identifié par l'URL http://www.w3.org/2001/XMLSchema-instance.
•L'attribut schemaLocation est utilisé lors de l'utilisation d'espaces de noms alors que l'attribut noNamespaceSchemaLocation est utlisé lorsque le document n'utilise pas d'espace de noms

```
schemaLocation="namespace1 schema1 namespace2 ...
namespaceN schemaN"
```

Documentation

•L'élément xsd:annotation permet d'ajouter des commentaires dans un schéma.

```
<xsd:annotation>
        <xsd:documentation xml:lang="fr">
                Commentaire en français
        </xsd:documentation>
        <xsd:appInfo>
                Information destinée aux applications
        </xsd:appInfo>
</xsd:annotation>
```


Déclaration d'éléments: type nommé

```
<xsd:element name="element" type="type"/>
```

```
<xsd:element name="title" type="xsd:string"/>
<xsd:element name="title" type="tns:Title"/>
```


Déclaration d'éléments: valeur par défaut et valeur fixe

```
<xsd:element name="title" type="xsd:string" default="Titre par défaut"/>
```

```
<xsd:element name="title" type="xsd:string" fixed="Titre fixe"/>
```


Déclaration d'éléments: type anonyme

```
<xsd:element name="element">
   <xsd:simpleType>
          ...
   </xsd:simpleType>
</xsd:element>
```

```
<xsd:element name="element">
    <xsd:complexType>
        ...
    </xsd:complexType>
</xsd:element>
```


Déclaration d'éléments: référence à un élément global

```
<!-- Définition globale de l'élément title -->
<xsd:element name="title" type="Title"/>
        ...
<!-- Définition d'un type -->
        <xsd:complexType ... >
                ...
                <!-- Utilisation de l'élément title -->
                <xsd:element ref="title"/>
                        ...
        </xsd:complexType>
```


Déclaration d'éléments: éléments locaux

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        ...
        <xsd:element name="strings">
            <xsd:complexType>
                <xsd:sequence>
                        <xsd:element name="local" type="xsd:string"
                        maxOccurs="unbounded"/> </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="integers">
                <xsd:complexType>
                    <xsd:sequence>
                                <xsd:element name="local" type="xsd:integer"
                                maxOccurs="unbounded"/>
                        </xsd:sequence> </xsd:complexType>
                </xsd:element>
</xsd:schema>
```

Déclaration d'éléments: éléments locaux

Un document valide pour le schéma suivant est le suivant.

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<lists>
        <strings>
                <local>Une chaîne</local>
                <local>A string</local>
        </strings>
        <integers>
                <local>-1</local>
                <local>1</local>
        </integers>
</lists>
```


Définition de types

•Les schémas XML distinguent les *types simples* introduits par le constructeur xsd:simpleType et les *types complexes* introduits par le constructeur xsd:complexType

```
<!-- Le type de l'élément object est xsd:anyType -->
<xsd:element name="object"/>
```


Définition de types: types prédéfinis

•String
•boolean
•Float: Flottant 32 bits
•Double: Flottant 64 bits
•Byte: Entier signé sur 8 bits
•unsignedByte: Entier non signé sur 8 bits
•Integer: Entier arbitraire.
•positiveInteger :Entier strictement positif
•negativeInteger: Entier strictement négatif
•nonPositiveInteger: Entier négatif ou nul
•nonNegativeInteger: Entier positif ou nul
•Int: Entier signé sur 32 bits
•unsignedInt: Entier non signé sur 32 bits
•Long: Entier signé sur 64 bits.
•……..

Définition de types: types simples

•définissent uniquement des contenus textuels.
•Ils peuvent être utilisé pour les éléments ou les attributs.
•Ils sont introduits par l'élément xsd:simpleType.

```
<xsd:simpleType ...> ... </xsd:simpleType>
```

```
<xsd:simpleType name="Byte">
        <xsd:restriction base="xsd:nonNegativeInteger">
                <xsd:maxInclusive value="255"/>
        </xsd:restriction>
</xsd:simpleType>
```


Définition de types: types complexes

•définissent des contenus purs (constitués uniquement d'éléments), des contenus textuels ou des contenus mixes.
•Tous ces contenus peuvent comprendre des attributs.
•Les types complexes peuvent seulement être utilisés pour les éléments.
•Ils sont introduits par l'élément xsd:complexType

```
<!-- Type explicite -->
<xsd:complexType ...>
        <!-- Construction du type avec xsd:sequence, xsd:choice ou xsd:all --> ...
</xsd:complexType>
```

```
<!-- Type dérivé à contenu textuel -->
<xsd:complexType ...>
        <xsd:simpleContent>
                <!-- Extension ou restriction --> ...
        </xsd:simpleContent>
</xsd:complexType>
```


Construction de types: élément vide

```
<xsd:element name="link" type="Link"/>
<xsd:complexType name="Link">
        <xsd:attribute name="ref" type="xsd:IDREF"
use="required"/> </xsd:complexType>
```

Un fragement de document valide peut être le suivant.

```
<link ref="id-42"/>
```

Construction de types: opérateur de séquence

```
<xsd:element name="book">
<xsd:complexType>
        <xsd:sequence>
                <xsd:element name="title" type="xsd:string"/>
                <xsd:element name="author" type="xsd:string"/>
                <xsd:element name="year" type="xsd:string"/>
                <xsd:element name="publisher" type="xsd:string"/>
        </xsd:sequence>
</xsd:complexType>
</xsd:element>
```

Cette déclaration est équivalente à la déclaration suivante dans une DTD.

```
<!ELEMENT book (title, author, year, publisher)>
```


Construction de types: opérateur de choix

```
<xsd:element name="publication">
<xsd:complexType>
        <xsd:choice>
                <xsd:element ref="book"/>
                <xsd:element ref="article"/>
                <xsd:element ref="report"/> <
        /xsd:choice>
</xsd:complexType>
</xsd:element>
```

Cette déclaration est équivalente à la déclaration suivante dans une DTD.

```
<!ELEMENT publication (book | article | report)>
```


Déclaration d'attributs

- La déclaration d'un attribut est semblable à la déclaration d'un élément mais elle utilise l'élément xsd:attribute au lieu de l'élément xsd:element.
- Les attributs name et type de xsd:attribute spécifient respectivement le nom et le type de l'attribut.
- Le type d'un attribut est nécessairement un type simple puisque les attributs ne peuvent contenir que du texte.

```
<xsd:attribute name="name" type="type"/>
```

L'exemple suivant déclare un attribut format de type xsd:string.

```
<xsd:attribute name="format" type="xsd:string"/>
```