<*XML*>Langage Web 2</*XML*>

- Emilien Bondu (Airbus Defence and Space)
- emilien.bondu@cassidian.com

Langage Web 1 ?

Sommaire

- Cours 1 : Introduction à XML et validation (XSD)
- Cours 2 : Validation (DTD, Relax NG, Schematron) et transformation (XPath, XSL; XSL-FO)
- Cours 3 : Recherche XML (XQuery), Base de données XML, Liens XML, Manipulation XML en Java: Dom, Sax
- Cours 4 : Manipulation XML en Java : StAX, Data-binding, JavaEE : tomcat
- Cours 5 : Manipulation XML en JavaEE: servlet, JSP, ExpressionLanguage, TagLib
- Cours 6 : TagLib (suite), Spring, JSF, AJAX


XQuery
XML
RDF
RDFS
Namespaces
URI
XPath
XLink
WSDL
OWL
XPointer
XSLT
Schematron
XSL-FO
RELAG-NG
SOAP
XSD
DTD
XSL

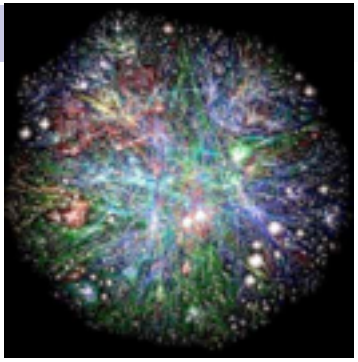
Introduction à XML

- Objectif et Historique
- Les bases d'XML
- Quelques cas d'utilisations d'XML
- Conception
- Validation

Explosion informationnelle

- Informations et documents dans l'entreprise
  - Constat :
    - Inflation du volume d'informations
  - Difficultés :
    - Difficultés à retrouver un document
    - Coût induit par la gestion de documents

Explosion informationnelle



- Informations et documents du Web
  - Constat :
    - Inflation du volume d'informations et formats
  - Difficultés :
    - Organiser
    - Échanger
    - Rechercher

+


Getting information off the
Internet is like taking a drink
from a fire hydrant.
– Mitchell Kapor

+

+


vimeo
FireAnt
standpoint
YEDDA
SHOUTWIRE
gOFFICE
Social
blogbeat
wink
FeedBurner
Yub.com
ClipShack
Noodly
box
pegasus
ShoZu
MusicSearch

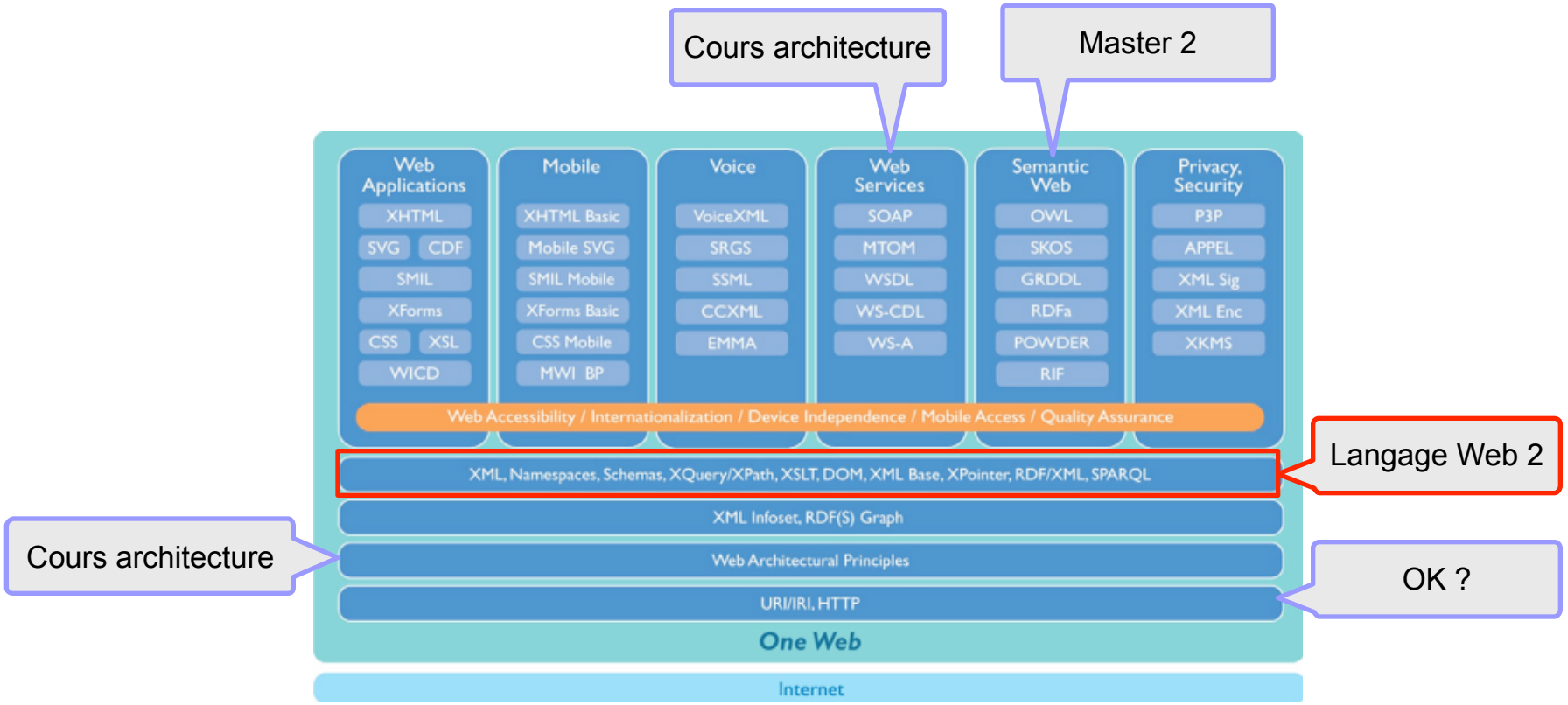
= Standards

Standards

- W3C technology Stack

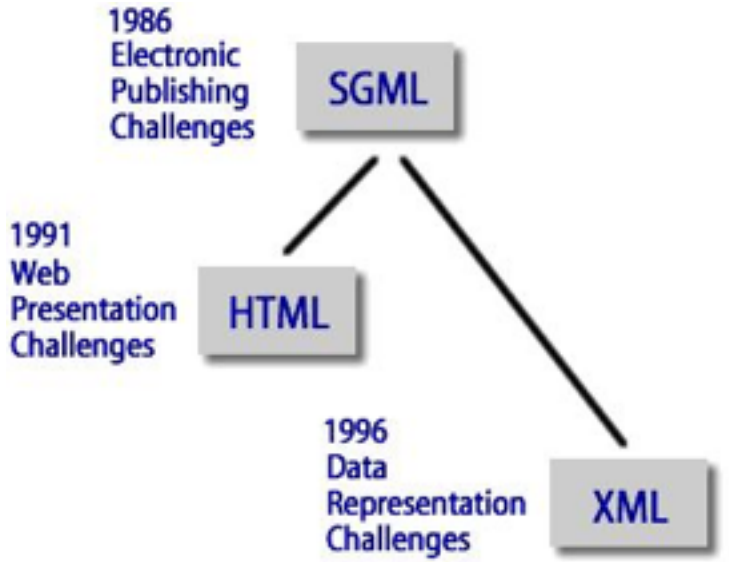
Standards

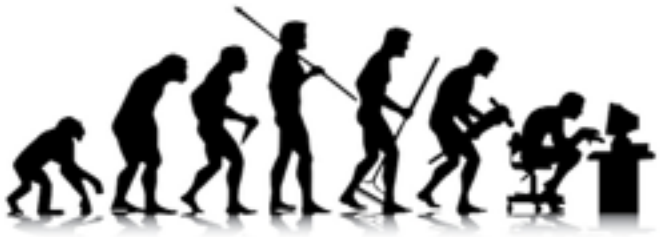
- W3C technology Stack



XML : Origine et objectifs

- Séparation du fond de la forme.
  - Forme = présentation à partir de la structure (style)
  - Fond = structure + données (contenu)
- Multiples précurseurs dont les plus connues :
  - SGML pour la structuration
  - HTML pour la présentation
  - => Approches mélangeant parfois le fond et la forme.

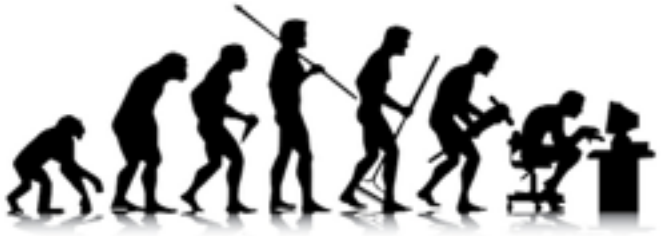




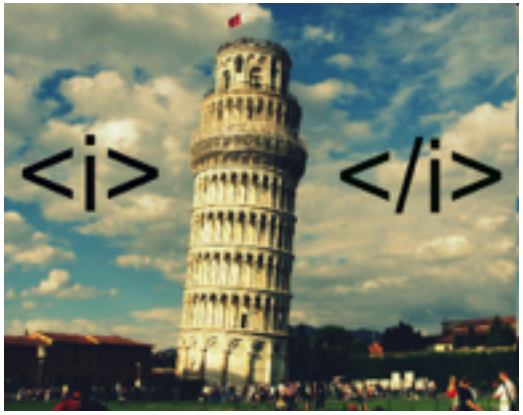
SGML(ISO 8879)

- **S**tandard **G**eneralized **M**arkup **L**anguage
- Standard international pour la définition de la structure et du contenu d'un document numérique
- Indépendant du type de matériel, et du système envisagé
- Système de balises (tag) extensibles
- Utilisation de **DTD** pour la validation de la structure
- Très utilisé pendant plus de 10 ans mais très lourd.

1987…

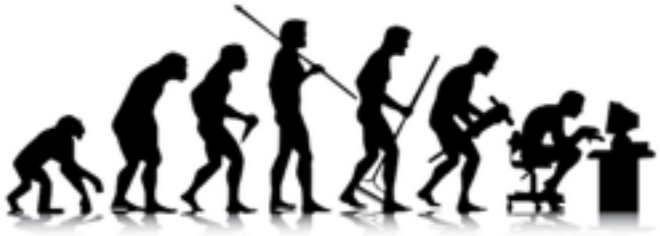


HTML(RFC 1866)

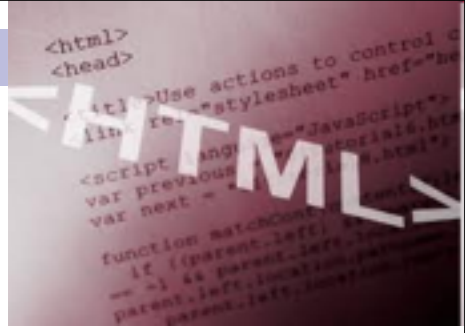


- **H**yper**T**ext **M**arkup **L**anguage
- Sous ensemble de SGML utilisé pour la présentation d'informations sur le Web
- Utilisation d'un vocabulaire de balises restreint

1991…



Exemple de fichier HTML

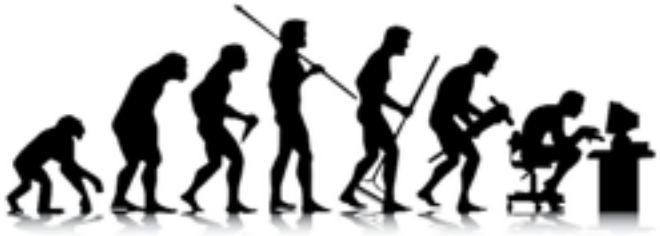


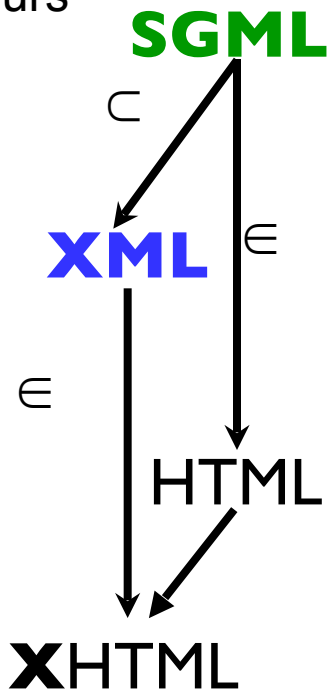

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html lang="en">
      <head>
            <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
            <title>HTML 4.01 Specification</title>
            <!-- Changed by: Ian B. Jacobs,  6-Dec-1998 -->
            <link rel="STYLESHEET" href="style/default.css" type="text/css">
      </head>
      <body>
            <div class="head">
                  <p>
                        <a href="http://www.w3.org/"><img alt="W3C" src="http://www.w3.org/Icons/w3c_home"></a>
                  </p>

                  <h1>HTML 4.01 Specification</h1>
                  <h2>W3C Recommendation 24 December 1999</h2>
            </div>
            <div class="content">
                  <p>
                        This specification defines the HyperText Markup Language (HTML), the
                        publishing language of the World Wide Web. This specification defines
                        HTML 4.01, which is a subversion of HTML 4.HTML 4 also takes great strides
                        towards the internationalization of documents, with the goal of making
                        the Web truly World Wide.
                  </p>
            </div>
      </body>
</html>
```

Qu'est ce que XML ?

- e**X**tensible **M**arkup **L**anguage
- Recommendation du W3C
  - 1.0 Ed01 en 1996, 1.0 Ed05 en 2008
  - 1.1 en 2004
- XML est un méta-langage universel pour représenter les données échangées (sur le Web) qui permet au développeur de délivrer du contenu depuis les applications à d'autres applications ou aux navigateurs
- XML standardise la manière dont l'information est :
  - échangée
  - présentée
  - archivée
  - retrouvée
  - transformée
  - cryptée
  - ...

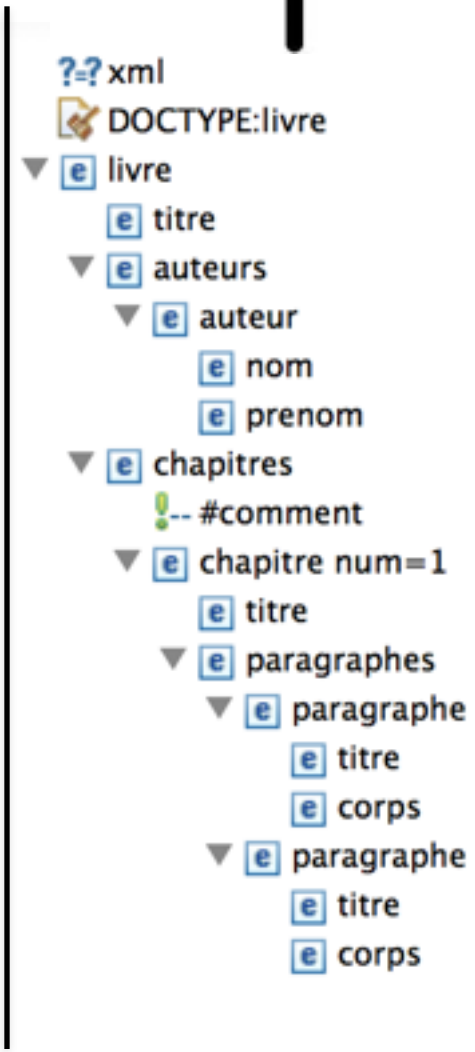
1996…





Un fichier XML c'est…

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE livre SYSTEM "livre.dtd">

<livre>
	<titre>Learning XML</titre>
	<auteurs>
		<auteur>
			<nom>Ray</nom>
			<prenom>Erik .T</prenom>
		</auteur>
	</auteurs>
	<chapitres>
		<!-- Premier chapitre du livre -->
		<chapitre num="1">
			<titre>Chapitre 1</titre>
			<paragraphes>
				<paragraphe>
					<titre>Premier paragraphe</titre>
					<corps>Contenu du premier paragraphe...</corps>
				</paragraphe>
				<paragraphe>
					<titre>Deuxième paragraphe</titre>
					<corps>Contenu du deuxième paragraphe...</corps>
				</paragraphe>
			</paragraphes>
		</chapitre>
	</chapitres>
</livre>
```



XML « Buildings Blocks »

- En tête
  - Une déclaration: version [, encoding, standalone]

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

  - Une **DTD** optionnelle (Document Type Definition) (standalone = “no”)

```
<!DOCTYPE livre SYSTEM "livre.dtd">
```

- Element
  - Délimités par des chevrons
  - Identifie la nature des éléments qu’ils encadrent
  - Format classique :

```
<element></element>
```

  - Elément vide :

```
<element/>
```

- Contenu
  - Elements
  - Chaine de caractères

```
<element>Contenu de l’élément</element>
```

- Attribut
  - Couple clé-valeur à l’intérieur d’un élément

```
<element attribut="valeur">
```

XML « Buildings Blocks »

- Entités
  - Ne sont pas interprétées

<	**<**	*less than*
>	>	*greater than*
&	**&**	*ampersand*
'	'	*apostrophe*
"	"	*quotation mark*

```
<element>5 &lt; 10</element>
```

- Commentaires

```
<!-- Commentaire -->
```

XML « Buildings Blocks »

- Quelques règles de syntaxe :
  - Tous les éléments doivent être fermés
  - Les balises sont dépendantes de la casse
  - Tous les documents XML doivent avoir un élément racine
  - Les valeurs des attributs doivent être entre guillemets
  - Les balises ne doivent pas se chevaucher
- Quelques particularités :
  - Tous les espaces sont préservés (contrairement à HTML)

Un exemple

Élément racine

Entête

Entité

Commentaire

Attribut

Contenu texte

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE livre SYSTEM "livre.dtd">

<livre>
      <titre>Learning XML &amp; Java</titre>
      <auteurs>
            <auteur>
                  <nom>Ray</nom>
                  <prenom>Erik .T</prenom>
            </auteur>
      </auteurs>
      <chapitres>
            <!-- Premier chapitre du livre -->
            <chapitre num="1">
                  <titre>Chapitre 1</titre>
                  <paragraphes>
                        <paragraphe>
                              <titre>Premier paragraphe</titre>
                              <corps>Contenu du premier paragraphe...</corps>
                        </paragraphe>
                        <paragraphe>
                              <titre>Deuxième paragraphe</titre>
                              <corps>Contenu du deuxième paragraphe...</corps>
                        </paragraphe>
                  </paragraphes>
            </chapitre>
      </chapitres>
</livre>
```

Conflits XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE livre SYSTEM "livre.dtd">

<livre>
      <titre>Learning XML</titre>
      <date>1/1/1983</date>
      <auteurs>
            <auteur>
                  <nom>Ray</nom>
                  <prenom>Erik .T</prenom>
            </auteur>
      </auteurs>
</livre>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE person SYSTEM "person.dtd">

<person>
      <titre>Mr</titre>
      <nom>Ray</nom>
      <prenom>Erik .T</prenom>
      <date>1/1/1983</date>
</person>
```

Conflits XML

- Utilisation d'espaces de nommage (namespace)
  - éviter les conflits de noms d'éléments
  - équivalent au package en Java
- Espace de nommage par défaut
  - s'applique à tous les éléments non préfixés

```
<element xmlns="uri_de_l'espace_de_nommage_par_défaut">
```

- Espace de nommage personnalisé
  - s'applique à tous les éléments et sous-élements du même préfixe

```
<toto:element xmlns:toto="uri_de_l'espace_de_nommage_toto">
```

Conflits XML

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

<livre xmlns="http://livre" xmlns:person="http://person">
      <titre>Learning XML</titre>
      <date>1/1/1983</date>
      <auteurs>
            <auteur>
                  <person:nom>Ray</person:nom>
                  <person:prenom>Erik .T</person:prenom>
            </auteur>
      </auteurs>
</livre>
```

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

<person xmlns="http://person">
      <titre>Mr</titre>
      <nom>Ray</nom>
      <prenom>Erik .T</prenom>
      <date>1/1/1983</date>
</person>
```

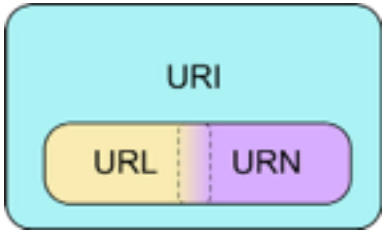
Conflits XML

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

<livre xmlns="http://livre" xmlns:person="http://person">
      <titre>Learning XML</titre>
      <date>1/1/1983</date>
      <auteurs>
            <auteur>
                  <person:nom>Ray</person:nom>
                  <person:prenom>Erik .T</person:prenom>
            </auteur>
      </auteurs>
</livre>
```

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

<person xmlns="http://person">
      <titre>Mr</titre>
      <nom>Ray</nom>
      <prenom>Erik .T</prenom>
      <date>1/1/1983</date>
</person>
```

- Rappel
  - l'URI (Uniform Resource Identifier) est une chaine de caractère qui identifie une Ressource du Web.
  - Souvent, l'URI est une URL (Uniform Resource Locator) qui identifie une adresse de domain Internet.
  - Parfois, l'URI est un URN (Universal Resource Name)

RFC 3986



Espace de noms / Namespaces

Quelques namespaces récurrents :

XML :
```
<element xml:lang="en" xmlns:xml="http://www.w3.org/XML/1998/namespace">
```

XMLSchema :
```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

XHTML :
```
<xhtml:body xmlns:xhtml="http://www.w3.org/1999/xhtml">
```

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

SVG :
```
<svg:circle xmlns:svg="http://www.w3.org/2000/svg">
```

SOAP :
```
<soap:body xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
```

RDF :
```
<rdf:RDF xmlns:rdf="http://www.w3.org/TR/REC-rdf-syntax#">
```

DublinCore :
```
<dc:title xmlns:dc="http://purl.org/dc/">
```

Exemples d'application

Autres exemples

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Virtual Library</title>
  </head>
  <body>
    <p>Moved to <a href="http://example.org/">example.org</a>.</p>
  </body>
</html>
```

?

Autres exemples

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Virtual Library</title>
  </head>
  <body>
    <p>Moved to <a href="http://example.org/">example.org</a>.</p>
  </body>
</html>
```

- XHTML = HTML + XML
- HTML bien formé et valide
  - DTDs : http://www.w3.org/TR/xhtml1/dtds.html#dtds
  - éléments et attributs en minuscule

XHTML

- Pour éviter :

```
<html>
  <head>
    <title>This is a bad HTML</title>
  </head>
  <body>
    <p>This text is be displayed by your browser
    <img src="http://example/image.png">

    <p>This text is displayed <i>in italic</p>
  </body>
</html>
```

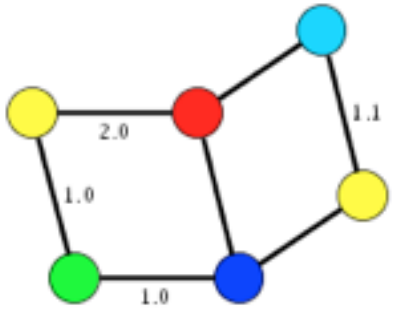
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>This is a bad HTML</title>
  </head>
  <body>
    <p>This text is displayed by your browser</p>
    <img src="http://example/image.png"/>

    <p>This text is displayed <i>in italic</i></p>
  </body>
</html>
```

D'autres exemples

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">

  <key id="d0" for="node" attr.name="color" attr.type="string">
    <default>yellow</default>
  </key>
  <key id="d1" for="edge" attr.name="weight" attr.type="double"/>
  <graph id="G" edgedefault="undirected">
    <node id="n0">
      <data key="d0">green</data>
    </node>
    <node id="n1"/>
    <node id="n2">
      <data key="d0">blue</data>
    </node>
    <node id="n3">
      <data key="d0">red</data>
    </node>
    <node id="n4"/>
    <node id="n5">
      <data key="d0">turquoise</data>
    </node>
    <edge id="e0" source="n0" target="n2">
      <data key="d1">1.0</data>
    </edge>
    <edge id="e1" source="n0" target="n1">
      <data key="d1">1.0</data>
    </edge>
    <edge id="e2" source="n1" target="n3">
      <data key="d1">2.0</data>
    </edge>
    <edge id="e3" source="n3" target="n2"/>
    <edge id="e4" source="n2" target="n4"/>
    <edge id="e5" source="n3" target="n5"/>
    <edge id="e6" source="n5" target="n4">
      <data key="d1">1.1</data>
    </edge>
  </graph>
</graphml>
```

?

GraphML



- Format XML pour décrire des graphes
  - orientés, non orientés, mixtes,
  - hypergraphes,
  - références à des données externes,
  - ajout d'attribut spécifique à des applications.
- Orienté principalement structure du graphe, aspect présentation en second plan.

D’autres exemples

```svg
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 120 80">
  <rect width="40" height="80" fill="#008d46"/>
  <rect width="40" height="80" x="40" fill="#fafeff"/>
  <rect width="40" height="80" x="80" fill="#d2232c"/>
  <rect width="120" height="80" fill="none" stroke="#000"/>
</svg>
```

?

```svg
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 96 105">
  <g fill="#97C024" stroke="#97C024" stroke-linejoin="round" stroke-linecap="round">
    <path d="M14,40v24M81,40v24M38,68v24M57,68v24M28,42v31h39v-31z" stroke-width="12"/>
    <path d="M32,5l5,10M64,5l-6,10 " stroke-width="2"/>
  </g>
  <path d="M22,35h51v10h-51zM22,33c0-31,51-31,51,0" fill="#97C024"/>
  <g fill="#FFF">
    <circle cx="36" cy="22" r="2"/>
    <circle cx="59" cy="22" r="2"/>
  </g>
</svg>
```

?

D’autres exemples

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 120 80">
  <rect width="40" height="80" fill="#008d46"/>
  <rect width="40" height="80" x="40" fill="#fafeff"/>
  <rect width="40" height="80" x="80" fill="#d2232c"/>
  <rect width="120" height="80" fill="none" stroke="#000"/>
</svg>
```

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 96 105">
  <g fill="#97C024" stroke="#97C024" stroke-linejoin="round" stroke-linecap="round">
    <path d="M14,40v24M81,40v24M38,68v24M57,68v24M28,42v31h39v-31z" stroke-width="12"/>
    <path d="M32,5l5,10M64,5l-6,10 " stroke-width="2"/>
  </g>
  <path d="M22,35h51v10h-51zM22,33c0-31,51-31,51,0" fill="#97C024"/>
  <g fill="#FFF">
    <circle cx="36" cy="22" r="2"/>
    <circle cx="59" cy="22" r="2"/>
  </g>
</svg>
```

SVG

- Scalable Vector Graphics (W3C 1999)
- Format XML pour décrire des images vectorielles
  - formes géométriques de base (rectangles, ellipses, etc.), chemins
  - remplissage couleur, dégradé, motifs, etc…
  - animations,
  - insertion de script pour étendre les capacités SVG.
- Utilisation possible de CSS ou XSL pour les aspects présentation.

D’autres exemples

```
<?xml version="1.0" encoding="UTF-8"?>
<office:document-content>
<office:automatic-styles />
  <office:body>
    <office:spreadsheet>
      <table:table table:name="the sheet name">
        <table:table-row>
          <table:table-cell>
            <text:p>IBM</text:p>
          </table:table-cell>
          <table:table-cell>
            <text:p>HP</text:p>
          </table:table-cell>
        </table:table-row>
        <table:table-row>
          <table:table-cell>
            <text:p>433362</text:p>
          </table:table-cell>
          <table:table-cell>
            <text:p>324600</text:p>
          </table:table-cell>
        </table:table-row>
      </table:table>
    </office:spreadsheet>
  </office:body>
</office:document-content>
```

OpenDocument

- OASIS 2006
- Format XML pour les documents bureautiques (tableur, présentation, traitement de texte, etc.)
- OpenDocument = format des fichiers OpenOffice
- Structuration en 4 parties
  - Contenu (content.xml)
  - Style (style.xml)
  - Métadonnées (meta.xml et settings.xml)

D'autres exemples

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

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:cd="http://www.recshop.fake/cd#">

      <rdf:Description rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
            <cd:artist>Bob Dylan</cd:artist>
            <cd:country>USA</cd:country>
            <cd:company>Columbia</cd:company>
            <cd:price>10.90</cd:price>
            <cd:year>1985</cd:year>
      </rdf:Description>

      <rdf:Description rdf:about="http://www.recshop.fake/cd/Hide your heart">
            <cd:artist>Bonnie Tyler</cd:artist>
            <cd:country>UK</cd:country>
            <cd:company>CBS Records</cd:company>
            <cd:price>9.90</cd:price>
            <cd:year>1988</cd:year>
      </rdf:Description>
</rdf:RDF>
```

RDF/XML (master 2)

- Modèle souple de représentation des connaissances
- Structure de graphe dans une structure d'arbre…
- Modèle de triplet
  - sujet (ressource)
  - *objet / prédicat*
  - valeur (ressource ou littéral)
- Exemple
  - Individu1 *hasName* « Dupont »
  - Individu1 *hasAge* 24
  - Individu2 *hasName* « Durand »
  - Individu1 *knowns* Individu2

D'autres exemples

```
<project name="HelloWorld" basedir="." default="main">

	<property name="src.dir" value="src" />

	<property name="build.dir" value="build" />
	<property name="classes.dir" value="${build.dir}/classes" />
	<property name="jar.dir" value="${build.dir}/jar" />
	<property name="main-class" value="oata.HelloWorld" />

	<target name="clean">
		<delete dir="${build.dir}" />
	</target>

	<target name="compile">
		<mkdir dir="${classes.dir}" />
		<javac srcdir="${src.dir}" destdir="${classes.dir}" />
	</target>

	<target name="jar" depends="compile">
		<mkdir dir="${jar.dir}" />
		<jar destfile="${jar.dir}/${ant.project.name}.jar" basedir="${classes.dir}">
			<manifest>
				<attribute name="Main-Class" value="${main-class}" />
			</manifest>
		</jar>
	</target>
	<target name="run" depends="jar">
		<java jar="${jar.dir}/${ant.project.name}.jar" fork="true" />
	</target>
	<target name="clean-build" depends="clean,jar" />
	<target name="main" depends="clean,run" />
</project>
```

Apache Ant

- Un makefile pour Java
- Target : init, compile, assemble, test and run
- Type de tâches
  - création/modification/manipulation/suppression de fichier, de répertoire, etc.
  - compilation, génération de documentation,etc.

D'autres exemples

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                             http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.exemple</groupId>
    <artifactId>simple-example</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>simple example</name>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

Maven 2 & 3

- Gestion complète du cycle de vie d'une application en Java
  - Goal : compile, test, package, site, deploy, etc…
- Plugin pour étendre le cycle de vie
  - assembly, javadoc, etc.
- Gestion des dépendances Java
  - identification des versions des librairies utilisées
  - contexte d'utilisation
- Project Object Model (POM)

D'autres exemples

```
<definitions name="HelloService" targetNamespace="http://www.examples.com/wsdl/HelloService.wsdl" xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://www.examples.com/wsdl/HelloService.wsdl" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <message name="SayHelloRequest">
      <part name="firstName" type="xsd:string"/>
   </message>
   <message name="SayHelloResponse">
      <part name="greeting" type="xsd:string"/>
   </message>

   <portType name="Hello_PortType">
      <operation name="sayHello">
         <input message="tns:SayHelloRequest"/>
         <output message="tns:SayHelloResponse"/>
      </operation>
   </portType>

   <binding name="Hello_Binding" type="tns:Hello_PortType">
   <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="sayHello">
      <soap:operation soapAction="sayHello"/>
      <input>
         <soap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:examples:helloservice"
            use="encoded"/>
      </input>
      <output>
         <soap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:examples:helloservice"
            use="encoded"/>
      </output>
   </operation>
   </binding>

   <service name="Hello_Service">
      <documentation>WSDL File for HelloService</documentation>
      <port binding="tns:Hello_Binding" name="Hello_Port">
         <soap:address
            location="http://www.examples.com/SayHello/">
      </port>
   </service>
</definitions>
```

?

Conclusion

- XML
  - Séparer les données de leur présentation
  - Créer de nouveaux formats de données
  - Echanger des données entre applications
  - Valider, stocker et rechercher des données
- Fondement de très nombreuses technologies
- Utilisé au quotidien dans l'entreprise

Avez-vous des questions ?

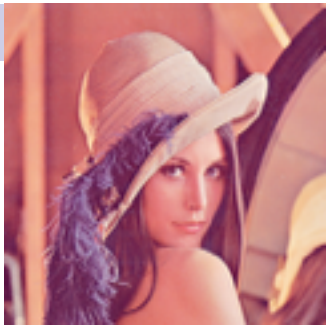
Les 10 règles de conception

- XML doit pouvoir être utilisé sans difficulté sur Internet
- XML doit soutenir une grande variété d'applications
- XML doit être compatible avec SGML et HTML
- Il doit être facile d'écrire des programmes traitant les documents XML
- Le nombre d'options dans XML doit être réduit au minimum, idéalement à aucune

Les 10 règles de conception

- Les documents XML doivent être lisibles par l'homme et raisonnablement clairs
- La spécification de XML doit être disponible rapidement
- La conception de XML doit être formelle et concise
- Il doit être facile de créer des documents XML
- La concision dans le balisage de XML est peu importante

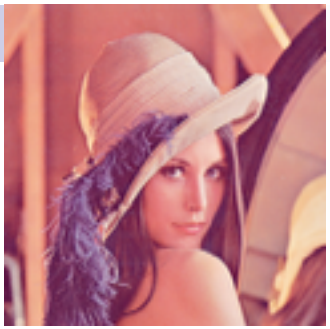
Dilemme attribut / élément



```
<person sex="female">
      <firstname>Lena</firstname>
      <lastname>Söderberg</lastname>
</person>
```

```
<person>
      <sex>female</sex>
      <firstname>Lena</firstname>
      <lastname>Söderberg</lastname>
</person>
```

Dilemme attribut / élément



```
<person sex="female">
      <firstname>Lena</firstname>
      <lastname>Söderberg</lastname>
</person>
```

- pas de valeurs multiples
- pas de valeur arborescente
- pas simple à étendre

- idéal pour les métadonnées ?

```
<person>
      <sex>female</sex>
      <firstname>Lena</firstname>
      <lastname>Söderberg</lastname>
</person>
```

Validation XML

- Document XML ”bien formé”
  - Syntaxe XML Correcte
- Document XML “valide”
  - Bien formé +
  - Conforme aux règles d’une DTD, d’un XML Schéma, etc.

XML - Schéma

- Une spécification du W3C (2001) pour l'écriture de règles de validation XML
  Specification : http://www.w3.org/XML/Schema
- Est lui même exprimé en XML (… Metalanguage …)
- XML Schéma permettent de spécifier des types de données (restrictions sur les chaînes de caractères, sur des intervalles de nombres)
- Ne permet pas de définir des entités comme dans les DTD.
- Est externe.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

</xs:schema>
```

XSD : Types simples

- Types simple = chaine de caractères :
  - Type de données xsd prédéfinis

    byte, unsignedByte, hexBinary, integer, positiveInteger, negativeInteger, int, unsignedInt, long, unsignedLong, short, unsignedShort, decimal, float, double, string, NormalizedString, token, boolean, anyURI, language, time, dateTime, duration, date, gMonth, gYear, gYearMonth, gDay, gMonthDay, etc.
  - Types simples avec restrictions

```
<xs:simpleType name="AouB">
      <xs:restriction base="xs:string">
            <xs:enumeration value="A"/>
            <xs:enumeration value="B"/>
      </xs:restriction>
</xs:simpleType>
```

- Déclaration des éléments simples :
  - Element simple non typé

```
<xs:element name="nom" />
```

  - Element simple typé

```
<xs:element name="date" type="xs:dateTime"/>
```

XSD : Types simples

- Attributs = types simples
- Déclaration des attributs :

```
<xs:attribute name="num" type="xs:int"/>
```

- Valeur par défaut

```
<xs:attribute name="num" type="xs:int" default="0"/>
```

- Attributs obligatoires ou optionnels

```
<xs:attribute name="num" type="xs:int" default="0" use="required"/>
<xs:attribute name="num" type="xs:int" default="0" use="optional"/>
```

XSD : Types complexes

- Types complexe = contient d'autres éléments ou attributs

- Déclaration de types complexes :

```
<xs:complexType name="paragrapheType">
      <xs:sequence>
            <xs:element name="titre" type="xs:string"/>
            <xs:element ref="corps"/>
      </xs:sequence>
</xs:complexType>

<xs:complexType name="paragrapheType">
      <xs:sequence>
            <xs:element ref="titre"/>
      </xs:sequence>
      <xs:attribute name="num" type="xs:int" use="optional"/>
</xs:complexType>
```

- Déclaration d'éléments complexes :

```
<xs:element name="paragraphe" type="paragrapheType"/>
```

XSD : Types complexes

- Indicateurs d'ordre (`all, choice, sequence`)

```
<xs:complexType name="paragrapheType">
      <xs:all>
            <xs:element name="titre" type="xs:string"/>
            <xs:element ref="corps"/>
      </xs:all>
</xs:complexType>
```

- Indicateurs d'occurrence (`minOccurs, maxOccurs`)

```
<xs:complexType name="chapitreType">
      <xs:sequence>
            <xs:element ref="paragraphe" minOccurs="1" maxOccurs="unbounded"/>
      </xs:sequence>
</xs:complexType>
```

- Indicateurs de groupe

```
<xs:group name="civiliteGroupe">
      <xs:sequence>
            <xs:element ref="nom" />
            <xs:element ref="prenom" />
      </xs:sequence>
</xs:group>
```

```
<xs:complexType name="personnalInfo">
  <xs:sequence>
    <xs:group ref="civiliteGroupe"/>
    <xs:element name="adresse" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Un exemple de XML Schéma

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                   targetNamespace="http://livre.org"
                   xmlns="http://livre.org">

   <xs:element name="livre" type="livreType" />
   <xs:element name="auteur" type="auteurType" />
   <xs:element name="titre" />
   <xs:element name="nom" />
   <xs:element name="prenom" />
   <xs:element name="chapitre" type="chapitreType"/>
   <xs:element name="paragraphe" type="paragrapheType"/>
   <xs:element name="corps"/>

   <xs:complexType name="livreType">
      <xs:sequence>
         <xs:element ref="titre" minOccurs="1" maxOccurs="1" />
         <xs:element ref="auteur" minOccurs="1" maxOccurs="unbounded"/>
         <xs:element ref="chapitre" minOccurs="1" maxOccurs="unbounded"/>
       </xs:sequence>
   </xs:complexType>

   <xs:complexType name="auteurType">
      <xs:sequence>
         <xs:element ref="nom" />
         <xs:element ref="prenom" />
      </xs:sequence>
   </xs:complexType>

   <xs:complexType name="chapitreType">
      <xs:sequence>
         <xs:element ref="titre"/>
         <xs:element ref="paragraphe" minOccurs="1" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="num" type="xs:int"/>
   </xs:complexType>

   <xs:complexType name="paragrapheType">
      <xs:sequence>
         <xs:element ref="titre"/>
         <xs:element ref="corps"/>
      </xs:sequence>
   </xs:complexType>
</xs:schema>
```

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

<livre xmlns="http://livre.org"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://livre.org livre.xsd">

        <titre>Apprendre le XML</titre>

        <auteur>
                <nom>Erik .T</nom>
                <prenom>Ray</prenom>
        </auteur>

        <!-- Premier chapitre du livre -->
        <chapitre num="1">
                <titre>Chapitre 1</titre>
                <paragraphe>
                        <titre>Premier paragraphe</titre>
                        <corps>Contenu du premier paragraphe...</corps>
                </paragraphe>
                <paragraphe>
                        <titre>Deuxième paragraphe</titre>
                        <corps>Contenu du deuxième paragraphe...</corps>
                </paragraphe>
        </chapitre>
</livre>
```

Un exemple de XML Schéma

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                   targetNamespace="http://livre.org"
                   xmlns="http://livre.org">

   <xs:element name="livre" type="livreType" />
   <xs:element name="auteur" type="auteurType" />
   <xs:element name="titre" />
   <xs:element name="nom" />
   <xs:element name="prenom" />
   <xs:element name="chapitre" type="chapitreType"/>
   <xs:element name="paragraphe" type="paragrapheType"/>
   <xs:element name="corps"/>

   <xs:complexType name="livreType">
      <xs:sequence>
         <xs:element ref="titre" minOccurs="1" maxOccurs="1" />
         <xs:element ref="auteur" minOccurs="1" maxOccurs="unbounded"/>
         <xs:element ref="chapitre" minOccurs="1" maxOccurs="unbounded"/>
       </xs:sequence>
   </xs:complexType>

   <xs:complexType name="auteurType">
      <xs:sequence>
         <xs:element ref="nom" />
         <xs:element ref="prenom" />
      </xs:sequence>
   </xs:complexType>

   <xs:complexType name="chapitreType">
      <xs:sequence>
         <xs:element ref="titre"/>
         <xs:element ref="paragraphe" minOccurs="1" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="num" type="xs:int"/>
   </xs:complexType>

   <xs:complexType name="paragrapheType">
      <xs:sequence>
         <xs:element ref="titre"/>
         <xs:element ref="corps"/>
      </xs:sequence>
   </xs:complexType>
</xs:schema>
```

?

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

<livre xmlns="http://livre.org"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://livre.org livre.xsd">

        <titre>Apprendre le XML</titre>

        <auteur>
                <nom>Erik .T</nom>
                <prenom>Ray</prenom>
        </auteur>

        <!-- Premier chapitre du livre -->
        <chapitre num="1">
                <titre>Chapitre 1</titre>
                <paragraphe>
                        <titre>Premier paragraphe</titre>
                        <corps>Contenu du premier paragraphe...</corps>
                </paragraphe>
                <paragraphe>
                        <titre>Deuxième paragraphe</titre>
                        <corps>Contenu du deuxième paragraphe...</corps>
                </paragraphe>
        </chapitre>
</livre>
```

Un exemple de XML Schéma

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                    targetNamespace="http://livre.org"
                    xmlns:livre="http://livre.org">

   <xs:element name="livre" type="livre:livreType" />
   <xs:element name="auteur" type="livre:auteurType" />
   <xs:element name="titre" />
   <xs:element name="nom" />
   <xs:element name="prenom" />
   <xs:element name="chapitre" type="livre:chapitreType"/>
   <xs:element name="paragraphe" type="livre:paragrapheType"/>
   <xs:element name="corps"/>

   <xs:complexType name="livreType">
      <xs:sequence>
         <xs:element ref="livre:titre" minOccurs="1" maxOccurs="1" />
         <xs:element ref="livre:auteur" minOccurs="1" maxOccurs="unbounded"/>
         <xs:element ref="livre:chapitre" minOccurs="1" maxOccurs="unbounded"/>
       </xs:sequence>
   </xs:complexType>

   <xs:complexType name="auteurType">
      <xs:sequence>
         <xs:element ref="livre:nom" />
         <xs:element ref="livre:prenom" />
      </xs:sequence>
   </xs:complexType>

   <xs:complexType name="chapitreType">
      <xs:sequence>
         <xs:element ref="livre:titre"/>
         <xs:element ref="livre:paragraphe" minOccurs="1" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="num" type="xs:int"/>
   </xs:complexType>

   <xs:complexType name="paragrapheType">
      <xs:sequence>
         <xs:element ref="livre:titre"/>
         <xs:element ref="livre:corps"/>
      </xs:sequence>
   </xs:complexType>
</xs:schema>
```

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

<livre xmlns="http://livre.org"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://livre.org livre.xsd">

        <titre>Apprendre le XML</titre>

        <auteur>
                <nom>Erik .T</nom>
                <prenom>Ray</prenom>
        </auteur>

        <!-- Premier chapitre du livre -->
        <chapitre num="1">
                <titre>Chapitre 1</titre>
                <paragraphe>
                        <titre>Premier paragraphe</titre>
                        <corps>Contenu du premier paragraphe...</corps>
                </paragraphe>
                <paragraphe>
                        <titre>Deuxième paragraphe</titre>
                        <corps>Contenu du deuxième paragraphe...</corps>
                </paragraphe>
        </chapitre>
</livre>
```

Références

- Cours de référence
  - Jenny Benois-Pineau, Georges Gardarin, Yves Bekkers, Khaled Khelif
- Livres
  - XML in a Nutshell, Third Edition Ed O'Reilly
- En ligne
  - http://www.w3.org/TR/xml/
  - http://www.w3schools.com/xml/default.asp