

e-business: *ML (56h)

Poly 1 : XML

Promotion 2004/2005

Création : Février 2002

Update : Octobre 2004

V3.04 / 168 slides

Cédric Carbone

Licence Professionnelle IRS2I

–Intégrateur de Réseaux locaux et de Service Intranet et Internet -
IUT d'Evry – Dépt. GEII - allée Jean Rostand – 91000 Evry
ccarbone@brainsoft.fr

Web dynamique
BRAINSOFT © Cédric Carbone - 1/164

e-business: *ML (56h)
Poly 1 : XML
Promotion 2004/2005

Création : Février 2002
 Update : Octobre 2004
 V3.04 / 168 slides
 Cédric Carbone

Licence Professionnelle IRS2I
 -Intégrateur de Réseaux locaux et de Service Intranet et Internet -
 IUT d'Evry - Dépt. GEII - allée Jean Rostand - 91000 Evry
 ccarbone@brainsoft.fr

Planning © Cédric Carbone - 2/164

- 14 séances de 4 heures :
 - Jeu 21/10/04 : 9h-13h & 14h-18h
 - Mer 01/12/04 : 9h-13h & 14h-18h
 - Jeu 02/12/04 : 9h-13h & 14h-18h
 - Mer 08/12/04 : 9h-13h & 14h-18h
 - Jeu 09/12/04 : 9h-13h & 14h-18h
 - Mer 15/12/04 : 9h-13h & 14h-18h
 - Jeu 16/12/04 : 9h-13h & 14h-18h
- Évaluation :
 - 3 TP notés
 - 1 mini projet

Pré-requis © Cédric Carbone - 3/164

Les modules Services Internet :

- HTML / CSS (H. Kadri)
- Apache / PHP / MySQL (O. Laxenaire)
- Programmation Objet Java (A. Mollica)

Programme © Cédric Carbone - 4/164

- XML_[p5] / DTD_[p34] / XML Schema_[p54]
- XPath_[p108] / XSL_[p83]
- SAX et DOM (implémentations PHP)_[p70]
- Overview des autres composants fonctionnels (XLL, Xptr...) et applicatifs XML (MathML, SMIL, SVG, WML...)_[p122]
- NxDB- Native XML DataBase- _[p128]

- Poly2 Web Services
- Pas de Poly 3
- Poly4 DHTML

Poly1 : XML (ce poly)

Objectifs © Cédric Carbone - 5/164

- Culture XML
- Utiliser les langages standards du W3C: XML, DTD, XSL, DOM et SAX, les composants applicatifs SMIL, SVG, MathML
- *Implémentations services Web (PHP Java .NET)*
- *Le contrôle côté client de formulaires, sur le DHTML*

Poly 2, 3 & 4

La problématique © Cédric Carbone - 6/164

- Échange, diffusion, sauvegarde, config. de produits, GED... : balisage d'un flux de texte
- Uniquement des données structurées:
 - Pas de mélange mise en page/sémantique

*[A partir du deuxième semestre de l'an 2000]
 le volume des données XML sera plus important que celui des données HTML.
 - Gartner Group*

La réponse

© Cédric Carbone - 7/164

- 1970 : Goldfab- Mosher- Lorie, ingénieurs IBM, **GML** (Generalized Markup Language), langage dédié à la description de document
- 1986 : **SGML** (Standardized GML) ISO 8879. Sa complexité limite l'utilisation de ce langage aux grands projets des grandes entreprises
- 1998 : **XML** (eXtensible Markup Language), issu de SGML mais 80% des fonctionnalités retranscrites pour 20% de complexité

XML : c'est 1/2

© Cédric Carbone - 8/164

- Des recommandations W3C
 - XML 1.0 1ere édition le 10 février 1998
 - XML 1.0 2nd édition le 6 Octobre 2000
 - XML 1.0 3e édition le 4 février 2004 (intègre les Errata de XML 1.0 2nd Édition + réécriture suivant la RFC 2119) <http://www.w3.org/TR/2004/REC-xml-20040204/>
 - XML 1.1 le 4 février 2004 (support Unicode 4.0, caractères de contrôle, plus de pb de fin de ligne) <http://www.w3.org/TR/2004/REC-xml11-20040204/>

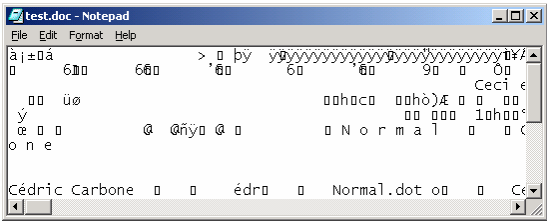
XML : c'est 2/2

© Cédric Carbone - 9/164

- Langage à balises comme l'HTML
- Ouvert / extensible
- Rigoureux
- Internationalisé : Unicode s'associe à n'importe quel jeu de caractères
- Tout peut se faire dans un « bloc-notes »
- Il permet de fédérer une information dans un format flexible et adapté à la publication multi-canal sur Internet

XML : ce n'est pas

© Cédric Carbone - 10/164



Format binaire de Word XP, impossible à traiter avec un éditeur de texte...

XML : ce n'est pas

© Cédric Carbone - 11/164

- Non conçu pour la création de pages Web dans le but de remplacer le HTML
- XML n'est pas supporté pas les navigateurs « anciens » mais l'utilisation d'XML n'est pas cantonnée à un browser
- XML n'est pas complètement fini et stable mais plutôt en cours de développement, en perpétuelle évolution

XML, les principes

© Cédric Carbone - 12/164

- Différencier la forme et la structure logique
- Standardiser un langage de codage/description
 - Informations indépendantes du soft et langage employé pour la créer
 - Format neutre et structurant
 - Gage de pérennité

Applications : exemples

- Le langage du WAP est le **WML** un langage XML
- SVG** est un langage XML dédié aux graphiques vectoriels
- SMIL** pour la synchronisation de différents médias
- Les protocoles des **Services Web** : dialectes XML à 100%
- XForms** : décrit en XML données, GUI et logique d'interaction

**Sun StarOffice
OpenOffice**

Ces fichiers XML sont validés par des DTD

```

[...]  

<text:h  

  text:style-  

    name="Heading  

  1"  

  text:level="1">  

    Hello World!!!  

</text:h>  

</office:body>  

[...]
```

Office 2003 1/3

- L'intégration de Office 2003 (Office 11) dans un processus informatique d'entreprise est désormais facilitée car l'information est structurée via le langage XML.

Démo WordML:

- Qu'est ce que l'information structurée?
- Quels buts?

Office 2003 2/3

- Word 2003 permet de sauvegarder en XML.
- Contrairement à OpenOffice qui se base sur les DTD, MS Office 2003 a choisi la norme XML Schema.
- Il existe deux méthodes sous Word 2003:
 - Sauvergarde WordML du document complet (données + styles) en XML validé par un Schéma XML (XSD connu mais sous le contrôle de MS)
 - Sauvergarde uniquement des données dans un fichier XML. Le fichier généré par Word ne comporte **aucune balise propriétaire**, uniquement celles définies dans un schéma XML créé par l'utilisateur :-)
- Il prend également en charge la norme XSLT

Office 2003 3/3

Les autres produits XMLisé d'Office 2003:

- InfoPath (client lourd, mode déconnecté, XML output...) pour les formulaires (non compatible XFORMS)
- Access 2003 (Extract XML à partir de plusieurs tables)
- Visio 2003 (Import/Export SVG)
- FrontPage 2003 (Outils XML+XSLT)

MathML

```

1 <?xml version="1.0" encoding="iso-8859-1" ?>  

2 <math xmlns="http://www.w3.org/1998/Math/MathML">  

3  

4 <!-- écrire x^4 + 2 = 0 -->  

5 <msup>  

6 <mi>x</mi>  

7 <mn>4</mn>  

8 </msup>  

9 <mo>+</mo>  

10 <mn>2</mn>  

11 <mo>=</mo>  

12 <mn>0</mn>  

13 </math>
```

MathML

© Cédric Carbone - 19/164

```

1 <?xml version="1.0" standalone="yes" ?>
2 <math xmlns="http://www.w3.org/1998/Math/MathML"
3
4   <math>
5     <math>
6       <math>
7         <math>
8           <math>
9             <math>
10              <math>
11                <math>
12                  <math>
13                    <math>
14                      <math>

```

MSN Messenger ! 1/2

© Cédric Carbone - 20/164

Date	Heure	De	A	Message
13/12/2003	18:44:18	Co2 ;)	(f)(i) test :\$(f)	Coucou (L)
21/12/2003	22:04:10	Co2 ;)	test taf	Test
21/12/2003	22:08:21	Co2 ;)	test taf	

MSN Messenger ! 2/2

© Cédric Carbone - 21/164

```

<?xml version="1.0" ?>
<?stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<!-- localized strings -->
<xsl:variable name="ColumnHeader_Date">Date</xsl:variable>
<xsl:variable name="ColumnHeader_Time">Heure</xsl:variable>
<xsl:variable name="ColumnHeader_From">De</xsl:variable>
<xsl:variable name="ColumnHeader_To">A</xsl:variable>
<xsl:variable name="ColumnHeader_Message">Message</xsl:variable>
<!-- test@hotmail.com.xml - Notepad
<?xml version="1.0" ?>
<?stylesheet type="text/xsl" href="MessageLog.xsl"?>
<Log LogonName="test@hotmail.com" FirstSessionId="1" LastSessionId="2">
<Message Date="13/12/2003" Time="18:44:18" DateTime="2003-12-13T17:44:18.0512"
SessionId="1">
<From
<User LogonName="" FriendlyName="Co2 ;)/>
</From>
<To
<User LogonName="test@hotmail.com" FriendlyName="(F)(i) test :$(F)"/>
</To>
<Text Style="font-family:Microsoft Sans Serif; color:#000000;">Coucou</Text>
</Message>
<Message Date="21/12/2003" Time="22:04:10" DateTime="2003-12-21T21:04:10.5622"
SessionId="2">
<From
<User LogonName="" FriendlyName="Co2 ;)/>
</From>
<To
<User LogonName="test@hotmail.com" FriendlyName="test taf"/>
</To>

```

Les règles syntaxiques

© Cédric Carbone - 22/164

- Toute balise ouverte doit être fermée
- Une seule balise racine
- Sensible à la casse
 <Nom></nom> : Erreur (la balise Nom est différente de la balise nom)
- Tous les attributs doivent avoir une valeur et cette valeur doit être entre ' ou "

Exemple XML

© Cédric Carbone - 23/164

```

<?xml version="1.0" encoding="ISO 8859 1"?>
<exemple maj="15/02/2001">
  <titre>Un exemple</titre>
  <chapitre numéro="1">
    <titre>Introduction</titre>
    <p>Ceci est un exemple très succinct</p>
    <img source="logo.gif" />
  </chapitre>
  <chapitre numéro="2"/>
</exemple>

```

XML bien formé

© Cédric Carbone - 24/164

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!-- Latin-1 encoding -->
<exemple update="15/02/2001">
  <titre>An example</titre>
  <chapter number="1">
    <title>Introduction</title>
    <author>Cédric</author>
    <img source="clip.gif" />
  </chapter>
  <chapter number="2" />
</exemple>

```

Error : Encoding Missing

© Cédric Carbone - 25/164

The XML page cannot be displayed

Cannot view XML in Internet Explorer because of the error and then click the **View Source** button.

```
<chapter number="1">
<title>Introduction</title>
<author>Cédric</author>
</chapter>
</example>
</xml>
```

An invalid character was found in text content. Error processing resource 'file:///E:/co2.xml'. Line 4, Position 16

```
<author>C
```

Error : Case Sensitive

© Cédric Carbone - 26/164

The XML page cannot be displayed

```
<chapter number="1">
<title>Introduction</title>
</chapter>
</example>
</xml>
```

End tag 'Title' does not match the start tag 'title'. Error processing resource 'file:///E:/sl14.xml'. Line 6, Position 23

```
<title>Introduction</Title>
```

Error : Empty Tag unclosed

© Cédric Carbone - 27/164

The XML page cannot be displayed

```
<img source="clip.gif"/>
</chapter>
</example>
</xml>
```

End tag 'chapter' does not match the start tag 'img'. Error processing resource 'file:///E:/sl14.xml'. Line 9, Position 3

```
</chapter>
```

Error : 2 root elements

© Cédric Carbone - 28/164

The XML page cannot be displayed

```
<xml version="1.0" encoding="ISO-8859-1" ?>
<example update="16/02/2001">
<title>An example</title>
<chapter number="1">
<title>Introduction</title>
<author>Cédric</author>
<img source="clip.gif" />
</chapter>
</example>
<example update="16/02/2001">
</example>
</xml>
```

Only one top level element is allowed in an XML document. Error processing resource 'file:///E:/co2.xml'. Line 12, Position 2

```
<example update="16/02/2001">
```

Prologue

© Cédric Carbone - 29/164

```
<?xml version="1.0" encoding="ISO 8859 1" ?>
```

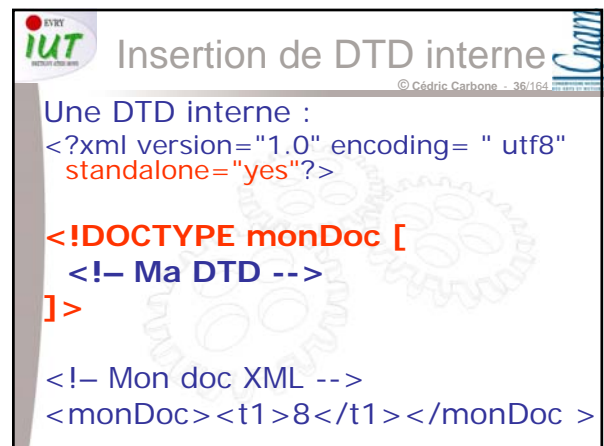
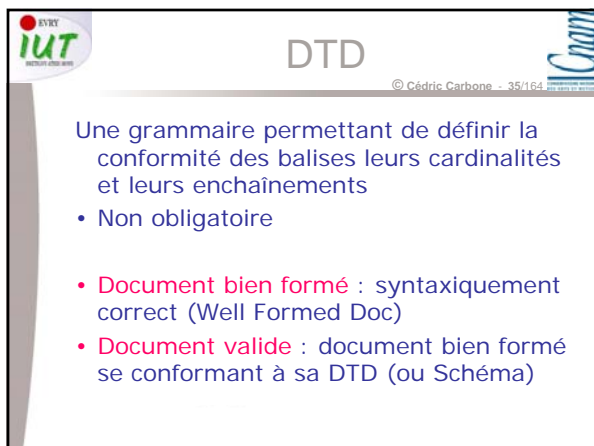
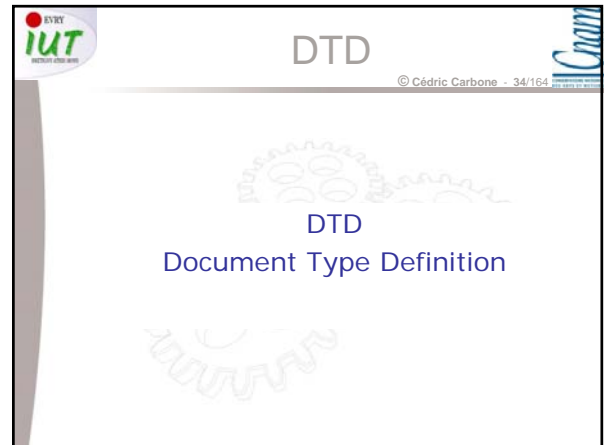
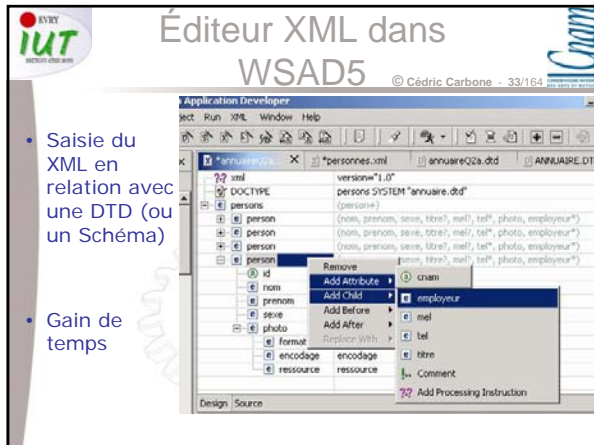
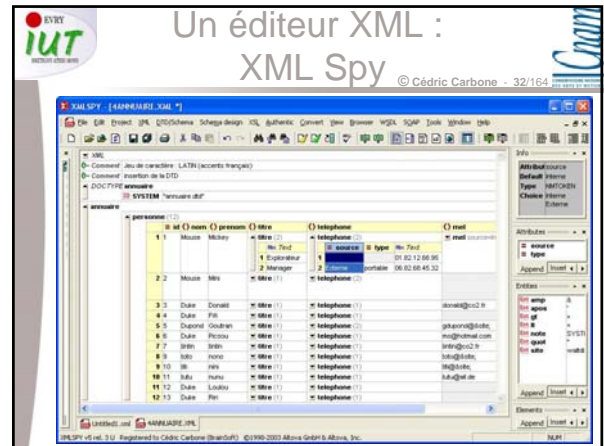
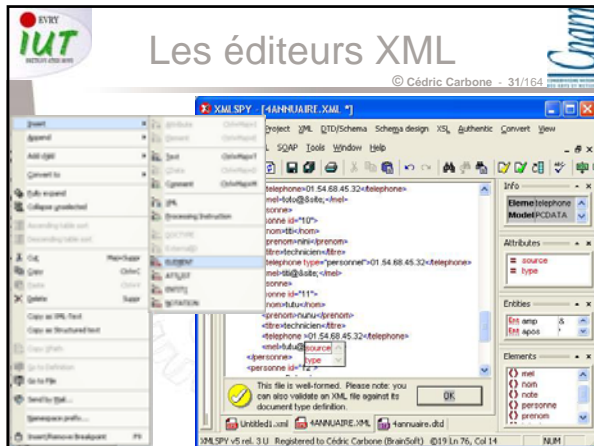
Standard ISO	Code de pays
UTF-8 (Unicode)	Jeu de caractères universel, mondial
ISO-8859-1 (Latin-1)	Europe occidentale, Amérique latine
ISO-8859-2 (Latin-2)	Europe centrale et orientale
ISO-8859-3 (Latin-3)	Europe du sud-est
ISO-8859-4 (Latin-4)	Scandinavie, pays Baltes
ISO-8859-5	Cyrillique
ISO-8859-6	Arabe
[...]	[...]

Éléments ou attributs

© Cédric Carbone - 30/164

Aucune réponse précise... mais:

- 1) l'ordre des attributs est au hasard
- 2) Un attribut est unique contrairement aux balises
- 3) Aucune possibilité d'extension dans le cas d'usage d'attribut
- 4) Un élément est plutôt un type de données (objet) et ses attributs ses propriétés ses usages (<tel type="perso">)
- 5) Avec un attribut, on peut imposer des valeurs par défaut (via DTD)



Insertion de DTD externe

© Cédric Carbone - 37/164

Propriétaire ou reconnue usage général

```
<!DOCTYPE monDoc SYSTEM
"../dtd/def.dtd">

<!DOCTYPE monDoc PUBLIC "-//NomDTD//DTD//EN"
"http://co2.org/dtd/def.dtd">
```

Standalone du doc XML doit être à no

DTD : Les éléments

© Cédric Carbone - 38/164

<!ELEMENT nomBalise type>

- ANY: n'importe quoi
- EMPTY: élément XML vide
- (#PCDATA): Parsed Character DATA : données textuelles (pas de balise) parsées
- (Elément..., ElémentN) : inclusion et ordonné
- (Elément1|Elément2) : inclusion ou
- (Elément?) : (0 1)
- (Elément+) : (1 N)
- (Elément*) : (0 N)

DTD : Les éléments

© Cédric Carbone - 39/164

```
<!ELEMENT telephone
(numero+, marque, option?) >
<!ELEMENT numero (#PCDATA) >
<!ELEMENT marque (#PCDATA) >
<!ELEMENT option (#PCDATA) >
```

Enchaînement des balises

Déclaration des balises

Document XML bien formé et valide

```
<telephone>
<numero>0617304245</numero>
<marque>Alcatel</marque>
</telephone>
```

DTD : Les attributs

© Cédric Carbone - 40/164

<!ATTLIST nomBalise nomAttribut typeAttribut typeDef Default>

- typeAttribut =
 - CDATA : contenu non parsé
 - (val1|val2|val3) : soit val1, soit val2 soit val3
 - NMTOKEN : un seul mot
 - ID : unique dans tout le document
- typeDef =
 - #IMPLIED : non obligatoire
 - #REQUIRED : obligatoire
 - #FIXED Value : fixe avec la valeur Value

DTD : Les attributs

© Cédric Carbone - 41/164

```
<!ATTLIST travail type (oui|non) "non">

<!ATTLIST person
  civilite (Mlle|Mme|Mr) #IMPLIED
  id ID #REQUIRED
  >

<!ATTLIST form method CDATA #FIXED "POST">
```

DTD : Les entités

© Cédric Carbone - 42/164

- Extrait de la DTD

<!ENTITY site "brainsoft.fr">
- Extrait du fichier XML

<root>mickey.mouse@&site;</root>
- Extrait du fichier XML parsé avec sa DTD

<root>mickey.mouse@brainsoft.fr</root>
- Entités prédéfinies : & < > ' "

DTD : Les entités

© Cédric Carbone - 43/164

- Définition dans un fichier externe
`<!ENTITY site SYSTEM "source.txt">`
- Entités paramétrées : déclaration
`<!ENTITY % nom_entite "valeur">`
- Entités paramétrées : utilisation
`<!ATTLIST tag %nom_entite;>`

DTD : Les entités Exemples

© Cédric Carbone - 44/164

Factoriser: %ref; %src; définit une seule fois mais peuvent être utilisées plusieurs fois

```
<!ENTITY % ref "id CDATA #REQUIRED">
<!ENTITY % src "source (Interne|Externe) 'Interne'">

<!ATTLIST telephone %ref; %src; type (portable|bippeur) "professionnel" >
```

DTD : Les entités Exemples

© Cédric Carbone - 45/164

Plus de lisibilité:

```
<!ELEMENT enseignement
(((module,date*,cours*,tp*,td*,examen?)+) |
((stage,date?)+) | ((conference,date*))+)>
```

deviendrait

```
<!ELEMENT enseignement (%module; | %stage; | %conf;)>
```

avec les entités paramétrées suivante :

```
<!ENTITY % module
"((module,date*,cours*,tp*,td*,examen?)+) ">
<!ENTITY % stage "((stage,date?)+)">
<!ENTITY % conf "((conference,date*)+)">
```

Les jeux de caractères Latin-1

© Cédric Carbone - 46/164

Par exemple:

ISO-8859-1	+	0	1	2	3	4	5	6	7	8	9
160		¡	¢	£	¤	¥	¦	§	¨	©	ª
170	«	»	¼	½	¾	¿	À	Á	Â	Ã	Ä
180	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ	Ò
190	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý
200	Þ	ß	à	á	â	ã	ä	å	æ	ç	¸
210	¸	¸	¸	¸	¸	¸	¸	¸	¸	¸	¸
220	¸	¸	¸	¸	¸	¸	¸	¸	¸	¸	¸
230	¸	¸	¸	¸	¸	¸	¸	¸	¸	¸	¸
240	¸	¸	¸	¸	¸	¸	¸	¸	¸	¸	¸
250	¸	¸	¸	¸	¸	¸	¸	¸	¸	¸	¸

© = ©

Démonstration

© Cédric Carbone - 47/164

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE directory SYSTEM "directory.dtd">
<directory>
  <person id="1">
    <name>Mouse</name>
    <phone type="professional">01.82.12.68.95</phone>
    <mail>mickey.mouse@brainsoft.fr</mail>
  </person>
</directory>
```

Démonstration

© Cédric Carbone - 48/164

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE directory SYSTEM "directory.dtd">
<directory>
  <person id="1">
    <name>Mouse</name>
    <phone>01.82.12.68.95</phone>
    <mail>mickey.mouse@&site;</mail>
  </person>
</directory>

<phone type="professional">01.82.12.68.95</phone>
<mail>mickey.mouse@brainsoft.fr</mail>
```

Démonstration

© Cédric Carbone - 49/164

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<ELEMENT directory (person+)>
<ELEMENT person (name,phone+,mail?)>
<!ATTLIST person id CDATA #REQUIRED>
<ELEMENT name (#PCDATA)>
<ELEMENT phone (#PCDATA)>
<!ATTLIST phone type (professional|personal)
"professional">
<ELEMENT mail (#PCDATA)>
<ENTITY site "brainsoft.fr">

```

Un éditeur de DTD : XML SPY

© Cédric Carbone - 50/164

Un éditeur de DTD : WSAD5

© Cédric Carbone - 51/164

Passage en mode design

Un éditeur de DTD : WSAD5

© Cédric Carbone - 52/164

Passage en mode design

TP1 – XML/DTD

© Cédric Carbone - 53/164

- Énoncé disponible dans le forum privée
- Objectifs :
 - Savoir concevoir un fichier XML
 - Savoir créé une DTD avec toutes les propriétés exposées dans ce cours
 - Savoir installer un parseur validant

3 heures

XML Schema

© Cédric Carbone - 54/164

XSD
XML Schema

Namespace

© Cédric Carbone - 55/164

- XML est plus qu'un langage, c'est une famille de langages (plusieurs centaines de langages répondent aux normes XML)
- Mélange de langages XML dédiés à un certain traitement (SVG pour un graphique vectoriel...) dans un même fichier XML
 - Spécifier le vocabulaire utilisé devant chacune des balises utilisées

XML Schema

© Cédric Carbone - 56/164

XML Schema (04/2001) : remplace les DTD existantes (10 ans déjà)

- Un peu plus complexe (car plus riche) que les DTD
- Typage des données
- Flexibilité
- Utilise une syntaxe XML (et non SGML comme avec les DTD)
- Aucun support de namespace

Déclaration dans le XML

© Cédric Carbone - 57/164

```
<?xml version="1.0" encoding="utf8"?>
<root
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="c:\schema.xsd">
  <!-- mon fichier XML - - >
</root>
```

Équivalent de l'insertion d'une DTD SYSTEM

XSD : 1^{ère} approche

© Cédric Carbone - 58/164

- En tête :**
 - `<?xml version="1.0" encoding="utf-8"?>`
 - `<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">`
- Déclaration d'un élément de **type simple**
 - `<xsd:element name="title" type="xsd:string" minOccurs="0" maxOccurs="7"/>`
 - Cardinalité avec les attributs `minOccurs` et `maxOccurs` (positionnée à 1 par défaut)

En DTD, la cardinalité est limitée à (1,1) (0,1) (0,n) et (1,n)

Le type simple

© Cédric Carbone - 59/164

- Type réservé aux éléments ne possédant pas de sous élément ou attribut
- string** représente une chaîne de caractères ("ok" ou "_ ?")
- boolean** représente une valeur booléenne (true ou false)
- decimal** représente un nombre décimal (-4 ou 2,2)
- float** représente un nombre à virgule flottante (-232 ou 17)
- double** représente un nombre réel double (1,123 ou 3)
- date** représente une date [aaaa-mm-jj] (2003-12-25)
- duration** représente une durée
- dateTime** représente une valeur date/heure
- time** représente une valeur horaire [format hh:mm:ss.sss]
- [...]

Le type simple

© Cédric Carbone - 60/164

http://www.w3.org/TR/xmlschema_0#CreatDt

EVRY IUT **<xsd:list> et <xsd:union>** © Cédric Carbone - 61/164

```

<xsd:simpleType name="numEtudiant">
  <xsd:list itemType="xsd:unsignedByte"/>
</xsd:simpleType>
  <numEtudiant>9776655</numEtudiant>

<xsd:simpleType name="numEtudNational">
  <xsd:union memberTypes="xsd:string
numEtudiant"/>
</xsd:simpleType>
  <numEtudNational>E977s55</ numEtudNational>

```

EVRY IUT **Types simples dérivés par restriction** © Cédric Carbone - 62/164

Facette = aspect (ou contrainte) définissant un espace de valeur

Le type passwd est composé de 4 à 6 caractères grâce aux 2 facettes

```

<xsd:simpleType name="passwd">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="4"/>
    <xsd:maxLength value="6"/>
  </xsd:restriction>
</xsd:simpleType>

```

Entier compris entre 10 et 20

```

<xsd:restriction
base="xsd:nonNegativeInteger">
  <xsd:minInclusive value="10"/>
  <xsd:maxExclusive value="20"/>
</xsd:restriction>

```

EVRY IUT **Types simples dérivés par restriction** © Cédric Carbone - 63/164

```

<xsd:simpleType name="jourSemaine">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="lundi"/>
    <xsd:enumeration value="mardi"/>
    <xsd:enumeration value="mercredi"/>
    <xsd:enumeration value="jeudi"/>
    <xsd:enumeration value="vendredi"/>
    <xsd:enumeration value="samedi"/>
    <xsd:enumeration
value="dimanche"/>
  </xsd:restriction>
</xsd:simpleType>

```

EVRY IUT **Facette Pattern** © Cédric Carbone - 64/164

Facette Pattern = aspect définissant un espace de valeur grâce à une expression régulière

```

<xsd:simpleType name="plaqueVoiture">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{4} [A Z]{2} \d{2}"/>
  </xsd:restriction>
</xsd:simpleType>

```

définit une plaque minéralogique

EVRY IUT **Type complexe : <xsd:sequence>** © Cédric Carbone - 65/164

<xsd:complexType> suivi un des trois compositeurs :
 <xsd:sequence> | <xsd:choice> | <xsd:all>

```

<xsd:element name="adress">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="street" type="xsd:string"/>
      <xsd:element name="city" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
  <street>rue Baudin</street>
  <city>Levallois</city>
</adress>

```

Opérateur , en DTD

EVRY IUT **Type complexe : <xsd:choice>** © Cédric Carbone - 66/164

<xsd:complexType> suivi un des trois compositeurs :
 <xsd:sequence> | <xsd:choice> | <xsd:all>

```

<xsd:element name="adress">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element name="street" type="xsd:string"/>
      <xsd:element name="city" type="xsd:string"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
  <adress> <city>Levallois</city> </adress>
  <adress> <street>rue Baudin</street> </adress>

```

Opérateur | en DTD

Type complexe : <xsd:all>

© Cédric Carbone - 67/164

<xsd:complexType> suivi un des trois compositeurs :
 <xsd:sequence> | <xsd:choice> | <xsd:all>

```
<xsd:element name="adress"><xsd:complexType>
  <xsd:all>
    <xsd:element name="street" type="xsd:string"/>
    <xsd:element name="city" type="xsd:string"/>
  </xsd:all></xsd:complexType></xsd:element>
```

Pas d'équivalent en DTD

```
<adress><city>Levallois</city></adress>
<adress>
  <city>Levallois</city>
  <street>rue Baudin</street>
</adress>
<adress>
  <street>rue Baudin</street>
  <city>Levallois</city>
</adress>
```

Un exemple complet

© Cédric Carbone - 68/164

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
  <xsd:element name="persons">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="person" minOccurs="1"
          maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="nom" type="xsd:string"/>
              <xsd:element name="prenom" type="xsd:string"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Contenu mixte

© Cédric Carbone - 69/164

```
<xsd:element name="para">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:element name="bold" type="xsd:string"/>
      <xsd:element name="italic" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="justif" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

validera l'extrait XML suivant

```
<para justif="right">
  Ceci est en <bold>gras</bold> et
  maintenant en <italic>italique</italic>
</para>
```

L'attribut

© Cédric Carbone - 70/164

- Déclaration d'un **attribut** (juste après l'élément)


```
<xsd:attribute name="monAtt" type="xsd:string"/>
```
- Trois attributs possibles
 - use : required / optional / prohibited
 - default : une chaîne de caractères
 - fixed : une chaîne de caractères

DTD	Équivalent XSD
#REQUIRED	use="required"
maValeur #REQUIRED	use="required" default="maValeur"
#IMPLIED	use="optional"
maValeur #IMPLIED	use="optional" default="maValeur"
Aucun équivalent en DTD pour interdire l'utilisation d'un attribut	use="prohibited"

Déclaration d'attribut

© Cédric Carbone - 71/164

```
<xsd:element name="item">
  <xsd:complexType>
    <xsd:attribute name="crec" type="xsd:date"/>
  </xsd:complexType>
</xsd:element>
```

Groupe de définitions

© Cédric Carbone - 72/164

```
<xsd:attributeGroup name="attPerson">
  <xsd:attribute name="id" type="xsd:integer"/>
  <xsd:attribute name="ecole" type="xsd:string"/>
</xsd:attributeGroup>
```

```
<!-- Utilisation du groupe d'attributs déclaré - >
<xsd:complexType name="person">
  <xsd:sequence>
    <xsd:element name="nom" type="xsd:string"/>
    <xsd:element name="prenom" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="attPerson"/>
</xsd:complexType>
```

Plusieurs méthodes

© Cédric Carbone - 73/164

3 façons de développer un schéma:

- Déclarer les éléments au fur et à mesure que l'on en a besoin (poupée russe)
- Déclarer tous les éléments et attributs et les utiliser par référence (par clonage)
- Utiliser conjointement les deux méthodes précédentes

Types par référence

© Cédric Carbone - 74/164

```

<xsd:element name="nom" type="xsd:string"/>
<xsd:element name="prenom" type="xsd:string"/>
<xsd:attribute name="cree" type="xsd:date"/>
<!-- Fin de la déclaration des éléments/attributs -->
<xsd:element name="personne">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="nom"/>
      <xsd:element ref="prenom" maxOccurs="3"/>
    </xsd:sequence>
    <xsd:attribut ref="cree"/>
  </xsd:complexType>
</xsd:element>

```

Les commentaires

© Cédric Carbone - 75/164

- C'est du XML : utilisation de <!-- --> possible
- Cependant, XML Schema définit 2 tags plus appropriés:

```

<xsd:annotation>
<xsd:documentation xml:lang="en" source="ReadMe.txt">
  Comments for Human users
</xsd:documentation>
<xsd:documentation xml:lang="fr">
  Commentaires pour les utilisateurs "Humains"
</xsd:documentation>
<xsd:appInfo source="http://foo.bar/">
  <bind xmlns="http://cnam.fr/">
    <class name="TMRI"/>
  </bind>
</xsd:appInfo>
</xsd:annotation>

```

**Importer un schéma...
...(dans un schéma) !**

© Cédric Carbone - 76/164

<xsd:include> pour inclure un schéma externe et <xsd:redefine> pour inclure un schéma en redéfinissant certaines balises

```

<xsd:include
  schemaLocation="http://w3.fr/schema.xsd"/>

<xsd:redefine schemaLocation="schem.xsd">
  <xsd:simpleType name="monAncienType">
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="10"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:redefine>

```

Schéma XML utilisé pour les slides suivants (Editeurs XML)

© Cédric Carbone - 78/164

```

<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
  <xsd:element name="persons">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="person" minOccurs="1" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="nom" type="xsd:string"/>
              <xsd:element name="prenom" type="xsd:string"/>
              <xsd:element name="sexe" type="xsd:string"/>
              <xsd:element name="titre" type="xsd:string" minOccurs="0" maxOccurs="1"/>
              <xsd:element name="mel" type="xsd:string" minOccurs="0" maxOccurs="1"/>
              <xsd:element name="tel" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
              <xsd:element name="photo"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="formal" type="xsd:string"/>
  <xsd:element name="encodage" type="xsd:string"/>
  <xsd:element name="ressource" type="xsd:string"/>
  <xsd:sequence>
    <xsd:complexType>
      <xsd:elements>
        <xsd:element name="employeur" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:elements>
    </xsd:complexType>
  </xsd:sequence>
</xsd:schema>

```

Méthode utilisée - poupées russes

XSD dans XMLSPY

© Cédric Carbone - 78/164

XSD dans WSAD5
© Cédric Carbone - 79/164

Vue / Modification en mode graphique

TP02 - XSD
© Cédric Carbone - 80/164

- Énoncé disponible dans le forum privée
- Objectifs :
 - Savoir concevoir un schéma XML
 - Déclarations d'éléments
 - Déclarations d'attribut
 - Déclarations de types et dérivations de types intégrés
 - Valider un fichier XML par un XSD

4 heures

2 parseurs
© Cédric Carbone - 81/164

Pour lire et traiter du XML, on utilise un parser.

Il existe deux API fonctionnant différemment:

- SAX (Simple API for XML)** «stream parsing» : lit le flux de données XML et déclenche des événements à chaque balise : traitement à la volée
- DOM (Document Object Model)** traduit un fichier XML sous forme d'arbre informatique

SAX 1/2
© Cédric Carbone - 82/164

- Basée sur un modèle évènementiel
- startDocument(), endDocument()
- startElement(), endElement()
- characters()
- Parse à la volée, **par flux** (contrairement à DOM qui oblige la lecture du document entier avant de commencer à parser)

=> Surtout utilisé pour les **fichiers volumineux**

API pour modifier un document XML

SAX 2/2
© Cédric Carbone - 83/164

4 interfaces

- DocumentHandler** : slide précédent
- ErrorHandler** retourne erreurs & warning
- DTDHandler** retourne événement en relation avec la DTD associée
- EntityResolver** gestion des URI

SAX 3/3
© Cédric Carbone - 84/164

```
<?xml version="1.0"?> [parser calls startDocument]
<person id="1"> [parser calls startElement]
  <name>Cedric Carbone</name> [parser calls startElement, characters, and endElement]
  <mark>11.95</mark> [parser calls startElement, characters, and endElement]
</person> [parser calls endElement]

[parser calls endDocument]
```

DOM

© Cédric Carbone - 85/164

- Document Object Model
- **Chargement en mémoire du document complet** puis consultation/modification possible
- **Sérialisation** dans un fichier d'un arbre en RAM
- **DOM Level1&2 sont des recommandations W3C**
 - XML 1ère partie : Core DOM level 1
 - Notez que vous utilisez un arbre XML quand vous utilisez du DHTML (2nd Partie : HTML DOM level 1)
(ex : `myDoc.myForm.MyField = "newValue"`)

API pour créer et manipuler un arbre XML en mémoire

Prog. objet en PHP

© Cédric Carbone - 86/164

Et si je devenais un objet PHP : \$me

```
$me->name contiendrait "Cédric"
$me->number_of_fingers égal à 10

$me->to_speak()
//Vous écoutez?
$me->to_do_sport()
//Pour perdre un peu de poids
```

PHP : XML-DOM

© Cédric Carbone - 87/164

Accéder à un doc xml

- `$doc = @xmldocfile("mondoc.xml")` or `die("Erreur : Objet DOM non créé");`
- `$root = $doc->root();` ; \$root contient tout le document xml
- `$fils = $root->children();` ; descend d'un cran
- `$fils` `>type` : vaut `XML_ELEMENT_NODE` dans le cas d'un noeud
- `$fils` `>name` : propriété du nom du noeud
- `$fils` `>content` : contenu du noeud
- `$fils` `>getattr(valeur)` : contenu de l'attribut

PHP DOM : syntax 1/2

© Cédric Carbone - 88/164

```
$tree = @xmldocfile("myDoc.xml") or
die("Error : DOM Tree not create");

$root = $tree->root();
/* $root contient tous les tags de votre
document */

$myAttribute = $root->getattr("id");
/* $myAttribute contient la valeur de
l'attribut id de l'élément root */
```

PHP DOM : syntax 2/2

© Cédric Carbone - 89/164

```
$child = $root->children();
```

Si \$child est un noeud élément:

```
$child->type == XML_ELEMENT_NODE
$child->name
$child->content
```

PHP DOM : exemple 1/2

© Cédric Carbone - 90/164

```
<persons>
  <person civ="Mme">
    Déborah
  </person>
  <person civ="Mr">
    Cédric
  </person>
  <note>
    Je ne veux pas extraire cette donnée
  </note>
</persons>
```

Résultat HTML :

- Mme **Déborah**
- Mr **Cédric**

PHP DOM : exemple 2/2

```

$doc = @xmldocfile("myDoc.xml")
$root = $doc->root();
$tmp = $root->children();
    echo '<ul>';
while($pers = array_shift($tmp)) {
    if ($pers->type == XML_ELEMENT_NODE){
        if ($pers->name == "person") {
            echo '<li>(' . $pers->getAttr("civ") . ')';
            echo '<b>' . $pers->content . '</b></li>';
        }
    }
}
    echo '</ul>';

```

Pour aller plus loin...

- PHP DOM fonctions
<http://www.php.net/manual/en/ref.domxml.php>
- PHP SAX fonctions
<http://www.php.net/manual/en/ref.xml.php>

API DOM de PHP

- Générer des docs xml

```

$doc=domxml_new_xmldoc("1.0");
$root = $doc->add_root("MSG");
$head = $root->new_child("TEST", "hello");
$head->new_child("TEST_FILS", "hello fils");
$head->set_attribute("Language", "fr");

$doc->dump_file("c:\test.xml", false, true);

```

Et XSL...

- Transformation XML-4 approches :
 - DOM & SAX : API propres à chaque langage
 - CSS : Très peu puissant (sortie HTML)
 - Data Island : Peu puissant (-> HTML)
 - XSL : méthode standardisée
- Un formateur XSL est requis

Parseurs & Formateurs

- **Parseurs** (Xerces, Expat, MSXML)
Un analyseur est un composant logiciel (classes Java ou bibliothèques) permettant d'accéder simplement aux données encapsulées dans un fichier XML. C'est donc le composant de base de toute application XML.
- **Formateur** (XT, Xalan Java, Saxon, FOP)
Le langage XML ne se préoccupant que du contenu du document (et non de sa représentation), il est nécessaire de formater les documents XML pour qu'ils soient lisibles. XSLT est actuellement le langage de feuilles de style le plus utilisé pour ce formatage.

Une parenthèse sur CSS

- Ceci permet de présenter de façon conviviale un document XML dans un navigateur Web
- La transformation XML/CSS se fait donc sur côté client
- Il suffit de créer un fichier CSS en définissant des styles typographiques pour les différentes balises du fichier XML
- Ne pas oublier d'inclure dans le fichier XML `<?xml-stylesheet href="style.css" type="text/css"?>` afin que le navigateur présente notre fichier XML

XML + CSS : Ex. (1/2)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="style.css" type="text/css"?>
<root>
<exemple>
<intitule>Fichier XML</intitule>
<date>12/10/01</date>
<auteur>Co2</auteur>
</exemple>
<exemple>
<intitule>Fichier XSLT</intitule>
<date>12/10/02</date>
<auteur>Cédric</auteur>
</exemple>
</root>
```

XML + CSS : Ex. (2/2)

```
<style type="text/css">
root, exemple {
intitule {
display: block;
width: 200px;
font-size: 16pt;
font-weight: bold;
background-color: black;
color: white;
}
date {
font-family: arial;
color: red;
}
auteur {
font-style: italic;
}
</style>
```

Data Island

- **I**lot de donnée = **i**mport dans un fichier HTML un fichier XML
- Manipulation possible en JavaScript
- Côté client

```
<html><body>
<xml id="monXml" src="xml.xml"></xml>
<table border="1" datasrc="#monXml"><tr>
<td>Le <span datafld="date"></span></td>
<td><span datafld="auteur"></span></td>
</tr></table>
</body></html>
```

Data Island

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<root>
<exemple>
<intitule>Fichier XML</intitule>
<date>12/10/01</date>
<auteur>Co2</auteur>
</exemple>
<exemple>
<intitule>Fichier XSLT</intitule>
<date>12/10/02</date>
<auteur>Cédric</auteur>
</exemple>
</root>
```

Transformer un doc avec XSL

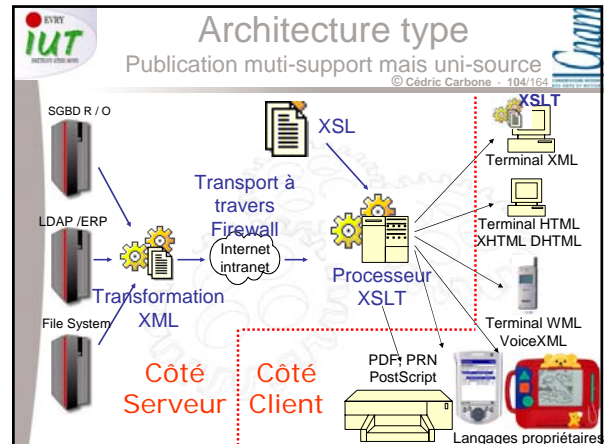
- **eX**tensible **S**tylesheet **L**anguage
- Trois recommandations W3C :
 - 1) **XML Path Language**
XPath v1.0 du 16.11.99
 - 2) **XSL Transformations**
XSLT v1.0 du 16.11.99
<http://www.w3.org/TR/xslt>
 - 3) **XSL Formatting Objects**
XSLFO §6 §7 de XSL v1.0 du 15.10.01
<http://www.w3.org/TR/xsl>

XSL

- XSL est le langage recommandé par le W3C
<http://www.w3.org/TR/xsl>
- XSL possède 2 fonctions principales :
 - langage de transformation (XSLT)
 - Pour une transformation d'un document XML en XML, HTML, text, WML, RTF, Tex, ou dans un autre rendu, (T pour Transformation)
 - langage de formatage (XSL/ FO)
 - Vocabulaire XML pour spécifier des instructions de formatage (FO: Formating Objects)

XSL-T

- Récupérer / Filtrer (dupliquer, supprimer)
- Trier
- Formater
- Générer du texte constant
- Générer du texte par rapport au XML source (via des calculs complexes)
- Utilisation de XPath pour sélectionner les nœuds à extraire



XSLT- Exemple 1a

- **Source (XML file):**
`<?xml version="1.0" ?><test>ATHENS</test>`
- **Style (XSLT file):**
`<?xml version="1.0" ?><xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="html" encoding="iso-8859-1"/>
<xsl:template match="test">
<html><body><h1> <xsl:value-of select="."/ />
</h1></body></html>
</xsl:template>
</xsl:stylesheet>`
- **Result (HTML file):**
`<html><body><h1>ATHENS</h1></body></html>`

XSLT- Exemple 1b

- **Source (XML file):**
`<?xml version="1.0" ?><test>ATHENS</test>`
- **Style (XSLT file):**
`<?xml version="1.0" ?><xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="wml" encoding="iso-8859-1"/>
<xsl:template match="test">
<wml><card id="co2"><p><xsl:value-of select="."/ /></p></card></wml>
</xsl:template>
</xsl:stylesheet>`
- **Result (WML file):**
`<wml><card id="co2"><p>ATHENS</p></card></wml>`

Ex2 – XML Source

```
<?xml version="1.0" ?>
<exam>
  <person>
    <name> Durand </name>
    <mark> 10 </mark>
  </person>
  <person>
    <name> Dupont </name>
    <mark> 13 </mark>
    <mark> 15 </mark>
  </person>
</exam>
```

Ex2 – XSLT

```
<xsl:template match="/">
<html> <head> <title>Exam
Results</title> </head> <body> <table>
<xsl:for-each select="exam/person">
<xsl:sort select="name" /> <tr> <th>
<xsl:value-of select="name"/> </th>
<xsl:for-each select="mark">
<td> <xsl:value-of select="."/ /> </td>
</xsl:for-each> </tr>
</xsl:for-each>
</table> </body> </html>
</xsl:template>
```

HTML result

Dupont	13	15
Durand	10	

Création de noeuds

© Cédric Carbone - 109/164

```

<xsl:element name="img">
  <xsl:attribute name="src">
    <xsl:value of select="url" />
  </xsl:attribute>
  <xsl:attribute name="alt">
    N<xsl:value of select="id" />
  </xsl:attribute>
</xsl:element>
  
```

XSL

```

<xsl:template
  match="/root">
  <xsl:for-each
    select="im">
    
  </xsl:for-each>
</xsl:template>
  
```

XSLT

```



  
```

XML

```




  
```

HTML

TP2a - Annuaire

© Cédric Carbone - 110/164

- Créer une feuille de style xsl qui permet d'extraire (et de présenter en html) de l'annuaire xml du TP1 :
 - Q1 : tout l'annuaire
 - Q2 : toutes les infos des membres (un design PDA et un design PdT)
 - Q3: toutes les infos correctement agencées et triées par ordre des noms de famille
 - Q4: effectuer un traitement comme remplacer les valeurs sexe (masculin/féminin) par la civilité (Mr / Mme)
 - Q5: inclure la photo de chaque membre

TP2b : 4h 

TP2b – Les news

© Cédric Carbone - 111/164

- Transformer le document source XML en document HTML
- Le document source est à la norme rss (Real Simple Syndication). Il est généré périodiquement et expose les news du site allhtml

TP2b : 1h30 

Les instructions XSLT

© Cédric Carbone - 112/164


<xsl:apply imports>	<xsl:fallback>
<xsl:apply templates>	<xsl:for each>
<xsl:attribute>	<xsl:if>
<xsl:call templates>	<xsl:message>
<xsl:choose>	<xsl:number>
<xsl:comment>	<xsl:processing-instruction>
<xsl:copy>	<xsl:text>
<xsl:copy of>	<xsl:value of>
<xsl:element>	<xsl:variable>
<xsl:document> (Nouveauté XSLT 1.1)	

<xsl:if>

© Cédric Carbone - 113/164

```

<?xml version="1.0"?><athens>
<person gender="Male">
  <name>Bob</name><mark>06</mark>
</person>
<person gender="Female">
  <name>Mary</name><mark>18</mark>
</person></athens>
<xsl:for-each select="person">
  <xsl:if test="@gender='Male'"> Mister </xsl:if>
  <xsl:if test="@gender='Female'"> Miss </xsl:if>
  <xsl:value-of select="name"/>
</xsl:for-each>
  
```

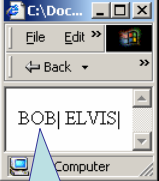


<xsl:variable> ; translate()

© Cédric Carbone - 114/164

```

<person gender="Male">
  <name>Bob</name><mark>06</mark>
</person>
<person gender="Female">
  <name>ElviS</name><mark>18</mark>
</person>
<xsl:template match="athens/person">
<xsl:variable name="MIN">
abcdefghijklmnopqrstuvwxy</xsl:variable>
<xsl:variable name="MAJ">
ABCDEFGHIJKLMNopqrstuVwxyz</xsl:variable>
<xsl:value-of
  select="translate(string(name),string($MIN),string($MAJ))"/>
</xsl:template>
  
```



Bob => BOB

Les variablesqui ne varient pas!

© Cédric Carbone - 115/164

- XSLT est un langage déclaratif sans effet de bord (maîtrise de l'ordre des traitements)
- Les variables sont donc des constantes pour toute la durée d'exécution de la feuille XSLT
- 2 solutions:
 - Contourner le problème (appels récursifs de templates avec passage de paramètres)
 - Faire certains traitements avec l'appui d'extensions non standards

substring()

© Cédric Carbone - 116/164

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<athens>
  <price>7 €</price>
</athens>
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="ISO-8859-1"/>
<xsl:template match="/athens">
  <xsl:value-of select="substring-before(string(price),' €')"/>
</xsl:template>
</xsl:stylesheet>
```

7 € => 7

Le séparateur est un espace

sum()

© Cédric Carbone - 117/164

```
<?xml version="1.0"?>
<athens>
  <person gender="Male">
    <name>Bob</name><mark>06</mark>
  </person>
  <person gender="Female">
    <name>Elvis</name><mark>18</mark>
  </person>
</athens>
<xsl:template match="/athens">
  Sum=<xsl:value-of select="sum(person/mark)"/>
</xsl:template>
</xsl:stylesheet>
```

Sum=24

6+18=24

XSLT utilise XPath

© Cédric Carbone - 118/164

```
<xsl:template match="/">
  <html> <head> <title>Exam
  Results</title> </head> <body> <table>
  <xsl:for-each select="exam/person">
  <xsl:sort select="name" /> <tr> <th>
  <xsl:value-of select="name"/> </th>
  <xsl:for-each select="note">
  <td> <xsl:value-of select="."/> </td>
  </xsl:for-each> </tr>
  </xsl:for-each>
  </table> </body> </html>
</xsl:template>
```

XPath

© Cédric Carbone - 119/164

- Recommandation W3C (v1.0 le 16 nov '99)
- Il est destiné à la recherche de patterns (motifs) dans un arbre XML grâce à une syntaxe simple et non ambiguë.
- Il est basé sur une composition de recherches par axes.
- XPath est également utilisé dans XPointer.
- Il sera probablement à la base du futur langage de requêtes.

Match Patterns

© Cédric Carbone - 120/164

- XPath pattern est un « location path »
- Un « location path » est constitué d'une série de steps (node test, prédicat, Axis Specifier)

```
<axe::<pattern>[prédicat]*
```

- 2 notations : abrégée ou non abrégée

Node Tests

© Cédric Carbone - 121/164

- Sélecteur**
- nom
- @att
- text()
- node()

Patterns sélectionnés

- Elément de tag nom
- Attribut de nom att
- Tout nœud de type texte
- Tout nœud
- Racine du document
- Tout sous chemin
- Tout élément
- Le nœud courant
- Le père du nœud courant

Notation abrégée d'Axis Spécifier (// -> descendant)

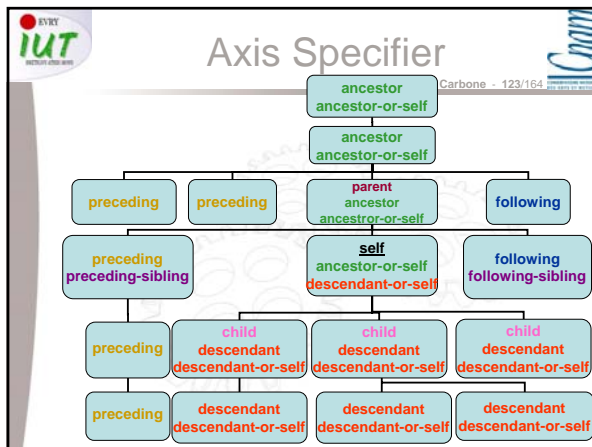
Prédicats

© Cédric Carbone - 122/164

- Permet de limiter les résultats retournés des node tests
- Sont toujours placés entre []

[Prédicat]	Pattern Sélectionné
nodetest[1]	Le 1er noeud
nodetest[position()=last()]	Le dernier noeud
node[@atr='val']	Tous les nodes dont l'attribut atr est à val
personne[note="0"]	les personnes qui ont une note à 0

```
<personne><nom>Gérard</nom><note>12</note></personne>
<personne><nom>Dédé</nom><note>0</note></personne>
```



Exemple de patterns

© Cédric Carbone - 124/164

```
* person name|mark person/name person[1]
person[position()=last()]
person[@cursus='ATHENS'] [mark>'10']
```

```
<person cursus="ATHENS">
  <name>Bob</name><mark>06</mark>
</person>
<person cursus="ATHENS">
  <name>Frank</name><mark>15</mark>
</person>
<person cursus="Typical">
  <name>Elvis</name><mark>18</mark>
</person>
```

TP3 - XPath

© Cédric Carbone - 125/164

- Dans ce TP, la puissance de XPath va permettre une sélection précise de patterns.
- Q0 : Ecrire l'arbre du document XML
- Faire un document xsl qui sélectionne:
 - Q1 : toutes les heures de départ
 - Q2 : toutes les heures de départ classées par ligne et service_voiture (précisez également le numéro de course)
 - Q3 : les heures de départ de toutes les courses numéro 8
 - Q4: les courses débutant à 8h30
 - Q5: Plan des lignes des bus s'arrêtant à la station FLO

TP3 : 3h

TP3 : Fichier source

© Cédric Carbone - 126/164

XSL-FO

© Cédric Carbone - 127/164

```

<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format"
  result-ns="fo">

  <xsl:template match="/">
    <fo:page-sequence font-family="serif">
      <xsl:apply-templates/>
    </fo:page-sequence>
  </xsl:template>

  <xsl:template match="para">
    <fo:block font-size="10pt" space-before="12pt">
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>

```

Utilisation de Sablotron en PHP

© Cédric Carbone - 128/164

- Processeur XSLT:

```

<?php
$proces = @xslt_create() or die("Can't create XSLT handle");
$xh = fopen("donnee.xml", "r") or die("Can't open XML file");
$sh = fopen("style.xml", "r") or die("Can't open XSL file");
$xmlContent = fread($xh, filesize("donnee.xml"));
$xmlContent = fread($sh, filesize("style.xml"));
@xslt_process($xmlContent, $xmlContent, $XSLtransformation);
echo $XSLtransformation;
@xslt_free($proces);
?>

```

Utilisation de MSXML en JS

© Cédric Carbone - 129/164

```

<html><body>
<script type="text/javascript">
  var xml = new
    ActiveXObject("Microsoft.XMLDOM");
  xml.async = false;
  xml.load("xml.xml");
  var xsl = new
    ActiveXObject("Microsoft.XMLDOM");
  xsl.async = false;
  xsl.load("xslt.xsl");
  document.write(xml.transformNode(xsl));
</script>
</body></html>

```

Utilisation de MSXML en ASP

© Cédric Carbone - 130/164

```

<%@ LANGUAGE = JScript %>
<%
  var oSource =
  Server.CreateObject("Microsoft.XMLDOM");
  oSource.async = false;
  oSource.load("C:\\source.xml");

  var oStyle =
  Server.CreateObject("Microsoft.XMLDOM");
  oStyle.async = false;
  oStyle.load("C:\\style.xml");

  Response.Write(oSource.transformNode(oStyle));
%>

```

Possibilités pour parser XML /XSL en J2EE

© Cédric Carbone - 131/164

- passer par du code java classique dans une jsp en faisant appel à une lib xml/xsl, du genre jaxp [1] (inclus désormais dans le jdk1.4), ou xalan [2] ou xt [3], ou ...
- passer par une servlet qui réalise la transformation en l'appelant depuis la page jsp qui utilise aussi [1], [2], [3],... Il y a un exemple dans la distribution de xalan [4]
- créer sa taglib jsp perso (bof...)
- utiliser une taglib xsl qui le fait déjà : merci monsieur Jakarta (apache) ;-) [5] (partie de l'ensemble des taglibs d'apache [6]) Ce qui semble le plus adaptée à une solution pure JSP

[1] <http://java.sun.com/xml/jaxp/index.html>
 [2] <http://xml.apache.org/xalan/index.html>
 [3] <http://www.xt.org/>
 [4] <http://xml.apache.org/xalan-j/semantics.html#standet>
 [5] <http://jakarta.apache.org/taglibs/doc/xsl-1.0/index.html>
 [6] <http://jakarta.apache.org/taglibs/index.html>

Message issu de la liste « xml-tech »

Transformation XSLT en Java

© Cédric Carbone - 132/164

```

<%@ page import="java.io.*" %>
<%@ page import="javax.xml.transform.Transformer" %>
<%@ page import="javax.xml.transform.TransformerFactory" %>
<%@ page import="javax.xml.transform.stream.*" %>
<%
  File xmlFile = new File("C:\\source.xml");
  File xslFile = new File("C:\\style.xml");
  FileWriter fw = new FileWriter("C:\\destination.xml", false);
  PrintWriter pw = new PrintWriter(fw);

  TransformerFactory factory = TransformerFactory.newInstance();

  Transformer transformer = factory.newTransformer(new StreamSource(xslFile));

  transformer.transform(new StreamSource(xmlFile), new StreamResult(pw));

  fw.close();
%>

```

Composants fonctionnels

- XPath (déjà vu)
- Xlink : <http://www.w3.org/TR/xlink>
- XPointer : <http://www.w3.org/TR/xptr>
- XQuery (recherche d'informations : « le SQL de XML »)

Langages applicatifs

Seul le langage WML sera traité en cours/tp

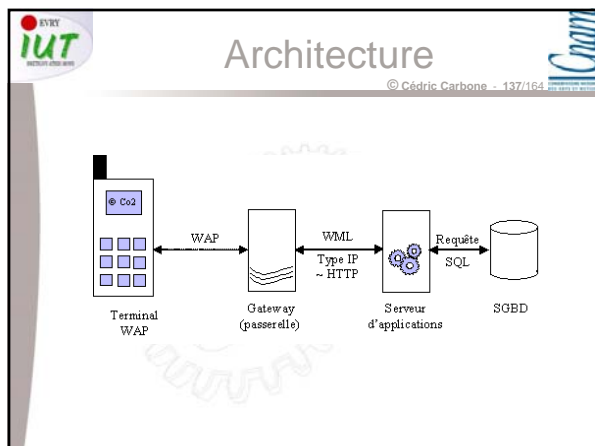
- WML : langage pour le WAP
- MathML : écriture scientifique
- XHTML : langage pour les pages web
- SMIL : Synchronisation de média
- SVG : Dessins et animations vectoriels
- RDF : méta données
- UIML : Interfaces indépendantes de la technologie
- ...

WAP : Généralités

- Protocole universel: WAP (Wireless Application Protocol)
- Langage de description de page: WML (Wireless Markup Language)
- Langage de script: WMLScript
- Consortium (l'équivalent du W3C pour le HTML): le WAP Forum (<http://www.wapforum.org>)
- DTD du WML: http://www.wapforum.org/DTD/wml_1.1.xml
- Ce langage dérivé de XML (DTD géré par le wapforum) est donc sensible à la casse (attribut en minuscule...), oblige la fermeture des balises, n'autorise pas d'attribut sans valeur...

Les contraintes

- Les données sont **longues à transmettre** (9600 bits/s, 1/7 du RNIS)
- GPRS 3 fois plus rapide
- Pas de plugin, peu de cache dans le navigateur WAP
- Des écrans minuscules, le plus souvent monochromes
- Ni clavier, ni stylet, ni souris, seulement une douzaine de touches



WML

- WML : Wireless Markup Language
- Un langage à balises pour le monde sans fil
- Possède les mêmes caractéristiques que le HTML (mise en forme, d'intégration d'images ou de zones de saisie)
- Conception dynamique de page Web (accès à son backOffice, à un fichier, une base de données, CGI ...)
- Une page WML est un jeu de cartes (deck en anglais)
- Chaque carte représente un écran de téléphone.

Exemple de WML

© Cédric Carbone - 139/164

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
3WWW.wapforum.org/DDTD/wml_1.1.xml>
<wml>
<card id="preCar" title="Ma première carte">
<p> <big> Voici le contenu de ma première carte </big>
<br/>
<select name="futPag">
<option onpick="#deuCar">2e Carte</option>
<option onpick="aide.wml">Aide</option>
</select>
</p>
</card>
<card id="secCar" title="Ma seconde carte">
<p> <big> Voici le contenu de ma seconde carte </big>
<br/> <a href="#preCar"> Lien vers la 1e Carte </a> </p>
<!-- On utilise ici les deux touches de fonction sous l'écran -->
</card>
</wml>
```

WBMP

© Cédric Carbone - 140/164

- **Wireless BitMaP**
- **Caractéristiques**
 - 2 couleurs (noir et blanc)
 - 1 bit - bichromatique
 - TYPE 0
 - Pas de compression possible
 - Suit les recommandations du WAP - WAE
- **Syntaxe en WML**

```

```

WMLScript

© Cédric Carbone - 141/164

- C'est un langage de script interprété par le terminal mobile, dérivé de l'ECMAScript pas gérés par tous les browsers
- Il permet d'exécuter de petits scripts contenant des instructions simples telles que des boucles, structures conditionnelles, fonctions, traiter les E/S des forms WML...
- Accès à des fonctions du portable comme :
 - Enregistrement de données sur la carte SIM
 - Envoi de SMS
 - Composition de numéro de téléphone
- Permettre une page d'identification avec mot de passe
- Contrôle lors des saisies dans des formulaires WML par l'utilisateur
- Génération de messages sous forme de boîtes de dialogue (économise une communication avec le serveur)

SMIL

© Cédric Carbone - 142/164

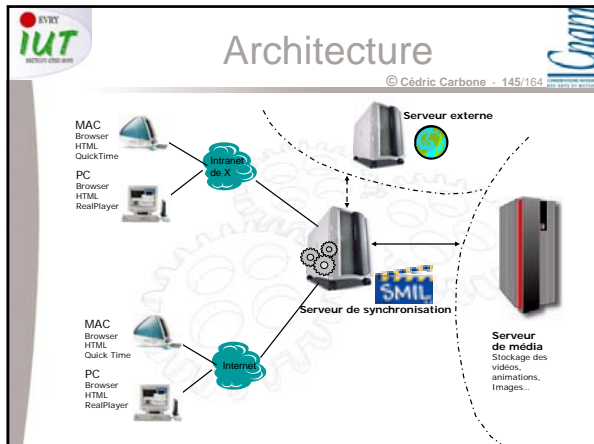
- Synchronised Multimedia Interaction Language
- Se prononce « smile »
- Recommandation W3C (v1 13 juin 2000)
- <http://www.w3.org/AudioVideo/>
- SMILtheque d'Aristote
<http://aristote1.aristote.asso.fr/Presentations>

Exemple

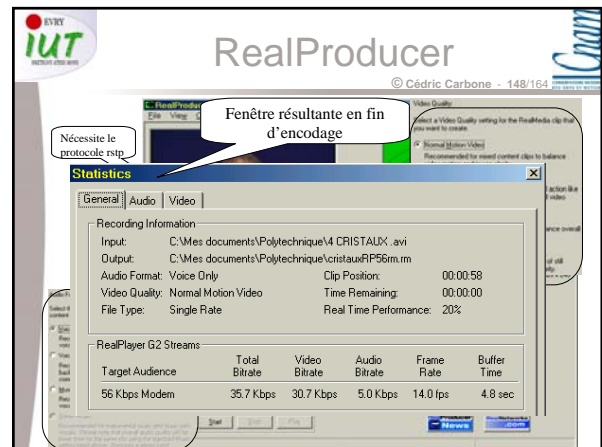
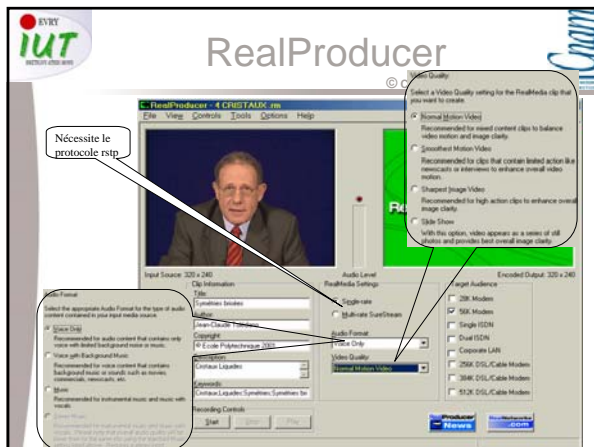
© Cédric Carbone - 143/164

Chef d'orchestre

© Cédric Carbone - 144/164



- ## RealNetworks
- © Cédric Carbone - 146/164
- RealNetworks a développé RealProducer, un utilitaire qui permet d'encoder les informations en fichiers "streamés".
 - Tous ces fichiers sont lisibles via le RealPlayer
 - RealText (.rt) : Flux de texte (défilement)
 - RealPix (.rp) : Enchaînements d'images (fondus, zoom, etc.)
 - RealVideo : Format vidéo
 - RealAudio : Format pour le son



Partie Technique

© Cédric Carbone - 149/164

```

<smil> <!-- Edité en Février 2002 - Durée : 1 min 30 sec-->
<head>
  <meta name="Titre" content="Symétries, symétries brisées"/>
  <meta name="Auteur" content="Cédric Carbone"/>
  <meta name="Copyright" content="(c) Ecole Polytechnique"/>
</head>
<layout>
  <root-layout width="900" height="595"/>
  <region id="video" left="33" top="115" width="320" height="240"
  fit="meet" background-color="#000000" z-index="20"/>
  <region id="plan" left="25" top="370" width="338" height="227"
  fit="meet" background-color="#FFFFFF" z-index="20"/>
  <region id="slides" left="428" top="115" width="420" height="320"
  background-color="#FFFFFF" z-index="20"/>
  <region id="bas" left="490" top="458" width="300" height="150"
  background-color="#0000FF" z-index="20"/>
</layout>
</smil>
  
```

Partie Technique

© Cédric Carbone - 150/164

```

<body><par>
  
  <anchor href="chap1.smi" coords="30,25,300,40"/>
  <anchor href="chap2.smi" coords="30,45,300,80"/>
  
  <seq>
    
    
    
    
    
  </seq>
  <textstream id="trad" src="test.rt" region="bas"/>
</par></body></smil>
  
```

EVERY IUT **Cours SVG** © Cédric Carbone - 151/164

- SVG: Scalable Vector Graphics
- Recommandation W3C à <http://www.w3.org/TR/SVG/>
- SVG 1.1
- SVG ouvert aux téléphones et PDA
- Format open-source et XML
- Images et animations en 2D
 - concurrent du Flash

EVERY IUT **SVG : le triangle** © Cédric Carbone - 152/164

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG
20001102//EN"
"http://www.w3.org/TR/2000/CR-SVG-
20001102/DTD/svg-20001102.dtd">
<svg width="500" height="500">
<path d="M50,50L450,50L250,450L50,50z"
style="fill: #DDDDDD; stroke: #333333;
stroke-width: 20" />
</svg>
```

EVERY IUT **SVG : plus compliqué** © Cédric Carbone - 153/164

- <defs/> permet de faire des objets réutilisables (via le tab <use/>)
- Rotation, translation
- Exemple :
- <defs>
- <linearGradient id="degrade">
- <stop offset="0%" style="stop-color:#660000"/>
- <stop offset="100%" style="stop-color:red"/>
- </linearGradient>
- </defs>
- <text fill="url(#degrade)" x="10" y="10">
- Hello World
- </text>

EVERY IUT **SVG : animations** © Cédric Carbone - 154/164

- <animateMotion path="m0,0l0,-230" begin="0.3s" dur="4s" fill="freeze" />
- <animateTransform attributeName="transform" attributeType="XML" type="scale" from="1" to="3" begin="0.3s" dur="2s" fill="freeze" />

EVERY IUT **SVG dans Illustrator Visio 2003** © Cédric Carbone - 155/164

EVERY IUT **XHTML** © Cédric Carbone - 156/164

- eXtended Hyper Text Markup Language
- Spécification XHTML traduite en français à <http://www.la-grange.net/w3c/REC-xhtml-basic/Overview.html>
- Langage plus élaboré et purifié que HTML
- XHTML est du XML avec une DTD précise.
- Langage web modulaire, portable et extensible
- Se plie aux règles du XML.
- Est conforme aux espaces de nommage.

XHTML – Les règles
 © Cédric Carbone - 157/164

- Les balises sont en minuscules.
- Refermer toutes les balises même les balises vides (
</br> ou
).
- Les attributs doivent être quottés.
- Pas d'attribut vide (<td nowrap="nowrap">
- Cohérence ouverture/fermeture de balise : la première balise ouverte doit être la dernière fermée (sinon overlapping = crossing de tags)

Base de données XML
 © Cédric Carbone - 158/164

- Nous avons étudié MySQL, un SGBD-R (SGBD-Relationnelle). Afin de stocker des données XML, ils existent des bases de données XML native (NXD pour Native XML Database) comme Tamino (Software AG)
- Aujourd'hui, les dernières montures des plus grand SGBDR (IBM, Oracle, Microsoft ont XMLisé leur base

➔ Cependant, des petites différences entre [NXD] et [SGBDR+fonctions XML] existent.

SGBD-R... aux fonctions XML
 © Cédric Carbone - 159/164

- Oracle 9i (composant XBO), SQL Server 2000 (via SQLXML 3.0) et DB2 v8 (Xperanto)
- Ces SGBDR sont donc utilisés également pour des applications XML
- Insère les fichiers XML de 2 façons différentes :
 - Mapping
 - Blobing (ou Clobing)

SGBDR partout !
 © Cédric Carbone - 160/164

Source: ZapThink

Année	Applications relationnelles	Applications orientées objet	Applications orientées XML
2000	85%	15%	0%
2001	85%	13%	2%
2002	84%	11%	5%
2003	77%	16%	7%
2004	74%	16%	10%
2005	70%	15%	15%

En 2003, 1/6 des SGBDR sont utilisés dans les applications orientées XML

Mapping
 © Cédric Carbone - 161/164

Répartie les données du document XML à insérer dans plusieurs tables relationnelles

- On peut alors utiliser ces données avec les outils relationnels habituels (requêtes SQL, jointure de table, validations de données...)
- Risque de perte de l'intégrité du document original
- Perte du caractère extensibilité
- Division/reconstitution des documents complexes pour les documents complexes.

Blobing ou Clobing
 © Cédric Carbone - 162/164

- BLOB = Binary Large Object
- CLOB = Character Large Object
- On insère le document XML en bloc dans un seul champ
- ➔ Plus simple que le mapping, pas de contrainte sur les documents complexes
- ➔ Possibilité d'utiliser le moteur de recherche plein texte
- ➔ Conservation de l'intégrité du document
- ➔ Le document XML est considéré comme une unité indivisible : pas de requête ou mise à jour d'élément de ce document car cette technique impose une lecture complète du fichier

Stockage XML natif

- Aucune transformation explicite du document
- Toute manipulation est invisible pour l'utilisateur qui voit le document XML comme une unité fondamentale
- ➔ Support de DOM, SAX, XQuery, XPath, XPointer...
- ➔ Intégrité du document
- ➔ Accepte la flexibilité propre au XML

Java et XML


- Couplage J2EE/XML intéressant
 - ⇒ java application portable
 - ⇒ xml données portables

Les navigateurs

Browser	Langages XML supportés
Internet Explorer < 5	∅
Internet Explorer 5	MSXML 2.0 (XSLT en WD)
Internet Explorer 5.5	MSXML 2.5 (XSLT en WD)
Internet Explorer 6	MSXML 3 : XSLT + SMIL
Netscape < 6	∅
Netscape 6	XML + XSLT + XLINK
Amaya (Browser du W3C)	XLink + XPointer + RDF + MathML + SVG...
Xsmilies	XML + XSLT+ XForms + SMIL + SVG + ...

Livre

XML: langage et applications
 2e édition décembre 2000
 Alain Michard
 Eyrolles 17 x 23 - 376p
 ISBN: 2-212-09206-7
 Noir et Blanc
 Prix public : 38,00 €
 Prix eyrolles.com: 36,10 € (236,80 F)



Livre

XML
 Schaum's
 Collection EdiScience
 ED Tittel
 Janvier 2003
 180 pages
 ISBN: 2 10 06934 9
 Prix public : 19,90 €
 www.ediscience.net

Copyright 2001-2004

Ce cours (quatre polycopiés et 10 TP) a été conçu à des fins pédagogiques dans le cadre des certificats professionnels du CNAM-Essonne et de la licence ISR2I de l'IUT d'Evry. Il est réservé à un usage privé et ne peut en aucun cas faire l'objet d'une utilisation commerciale. Comme l'énonce la loi, toute représentation, traduction, adaptation ou reproduction, même partielle, par tous procédés, faite sans autorisation préalable de l'auteur est illicite.