XML
eXtensible Markup Language

Didier Donsez

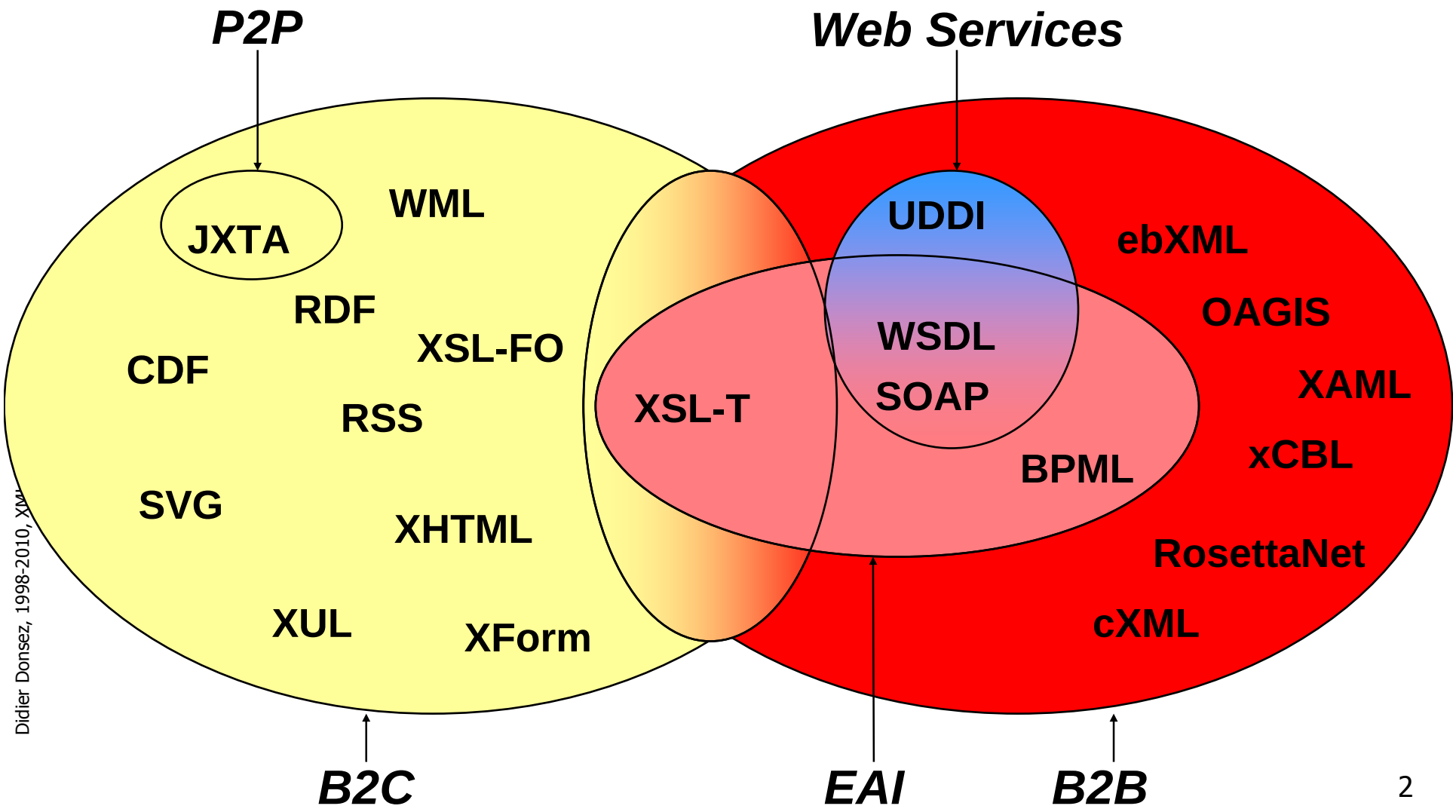
Université Joseph Fourier (Grenoble 1)

PolyTech'Grenoble LIG/ADELE

Didier.Donsez@imag.fr

Didier.Donsez@ieee.org

Les Applications XML



Au sommaire

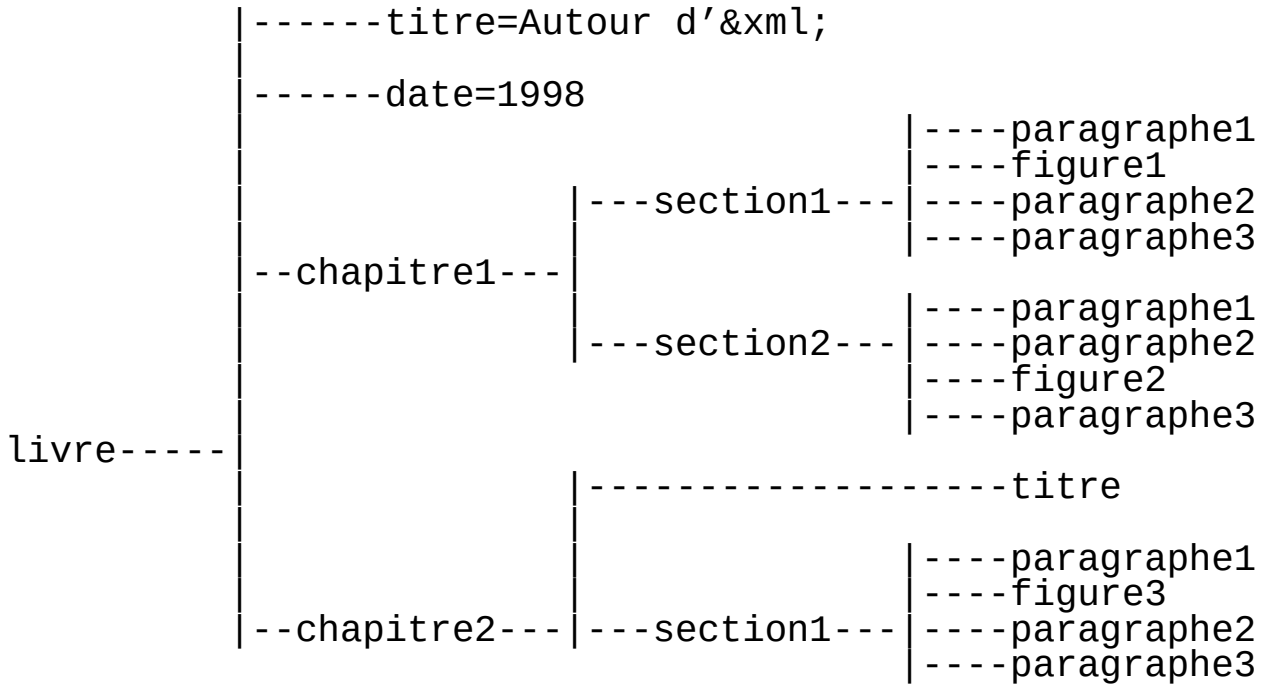
- Principes de SGML et Limites de HTML
- Le typage de document
  - DTD, Namespace, XML Schema, RelaxNG
- L'Outillage XML
  - XPath, Xquery, XSL-T, XSL-FO
- Les API
  - SAX, DOM
- Les Applications
- Bibliographie

Principe de la GED
Gestion Electronique Documentaire

- Gestion des Documents de l ’entreprise
  - Documents commerciaux
    - catalogues, fiches produits, factures, …
  - Documents techniques
    - spécifications techniques, manuels utilisateur, manuels d ’entretien, …
  - Documents qualités
    - suivi de fabrication de lots, rapports d ’incident, …
  - Volume : 90% de l ’information est sous cette forme
- Objectif de la GED
  - Représentation *uniforme voir normalisé*
  - Archivage
  - Recherche

Exemple de Document Structuré

- Document structuré
  - comporte des parties ayant une signification particulière
- Exemple



- La GED utilise des documents structurés
  - je recherche tous les documents dont le titre contient XML !

Un autre exemple l'EDI (Electronic Data Interchange)

- ou Echange de Données Informatisées
- Motivation : Vers le "Zéro Papier"
  - Coût de traitement d'un bon de commande papier : 70 $
  - Erreur, Lenteur, Traçage Difficile, ...
- Principe
  - Echange de Documents de Format Normalisé *(Bon de Commande, de Réception, Facture, ...)* entre les Systèmes d'Information de Partenaires

SGML

- DTD : Document Type Description
  - Grammaire de description d ’un type de document
    - Elément, Attribut, Entité
  - DTD normalisée (PUBLIC), DTD spécifique (SYSTEM)
- Exemple

```
<!DOCTYPE livre [
  <!ELEMENT livre                    - -   (titre, date?, chapitre+)   >
  <!ELEMENT chapitre                 - -   (title?, section+)          >
  <!ELEMENT section                  - O   ((paragraphe|figure)+)      >
  <!ELEMENT (titre | paragraphe)     - O   (#PCDATA)                   >
  <!ELEMENT date                     - O   EMPTY                       >
  <!ELEMENT figure                   - O   EMPTY                       >
  <!ATTLIST   date          année    NUMBER          #REQUIRED
                            mois     CDATA           #IMPLIED          >
  <!ATTLIST   figure        ref      CDATA           #REQUIRED         >

  <!ENTITY xml "XML">
]>
```

Fermeture optionnel

Représentation des instances de DTD SGML

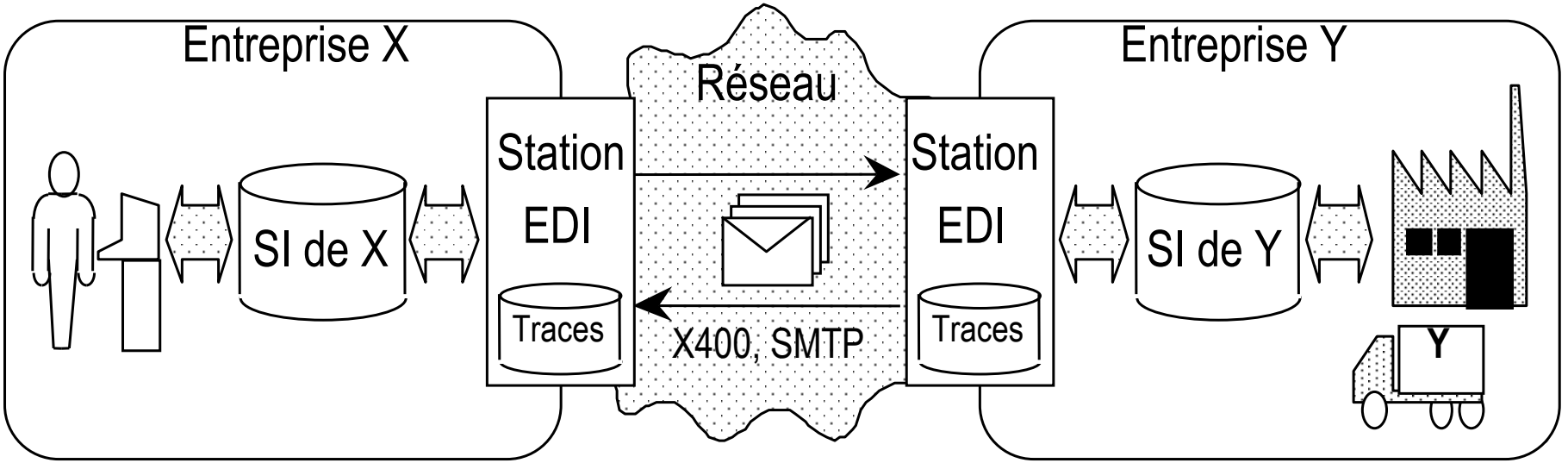
- Structuration par Balisage (TAG)
  - Document instance d ’une DTD
    - **<!DOCTYPE livre SYSTEM « livre.dtd" [*suppléments*]>**
      <livre> … </livre>
    - **<!DOCTYPE my.dtd [*<!-- déclarations de MY.DTD --> …*]>**
      <my.dtd> Voici un instance d ’un document MY.DTD </my.dtd>
  - Eléments
    - **<chapitre> … </chapitre>**
    - **<titre>…** fermeture optionnelle car pas de sous éléments
    - **<date …>** élément atomique sans texte
  - Attributs
    - **<date annee=1998>** attribut obligatoire
    - **<date annee=1998 mois="Janvier">** attribut optionnel
  - Entités
    - **&xml;** instance d ’entité

Une instance d ’un type de document SGML

```
<!DOCTYPE livre  SYSTEM   "livre.dtd" [<!ENTITY autour "Autour">]>
<livre> <!– un commentaire  -->
    <title>&autour; d’&xml;</title>
    <date annee=1998>
    <chapitre>
      <section>
          <paragraphe>&xml; est l’acronyme de eXtenbile Markup Language</paragraphe>
          <figure ref="figure1.eps">
          <paragraphe>Son précurseur est SGML
          <paragraphe>Il existe de nombreuses applications &xml;
      </section>
      <section>
          <paragraphe>Un document est bien formé quand … </paragraphe>
          <paragraphe> Un document est valide quand …
          <figure ref="figure2.eps">
          <paragraphe>Un document est dit standalone quand …</paragraphe>
      </section>
    </chapitre>
          <!– d’autres chapitres    -->
  </livre>
```

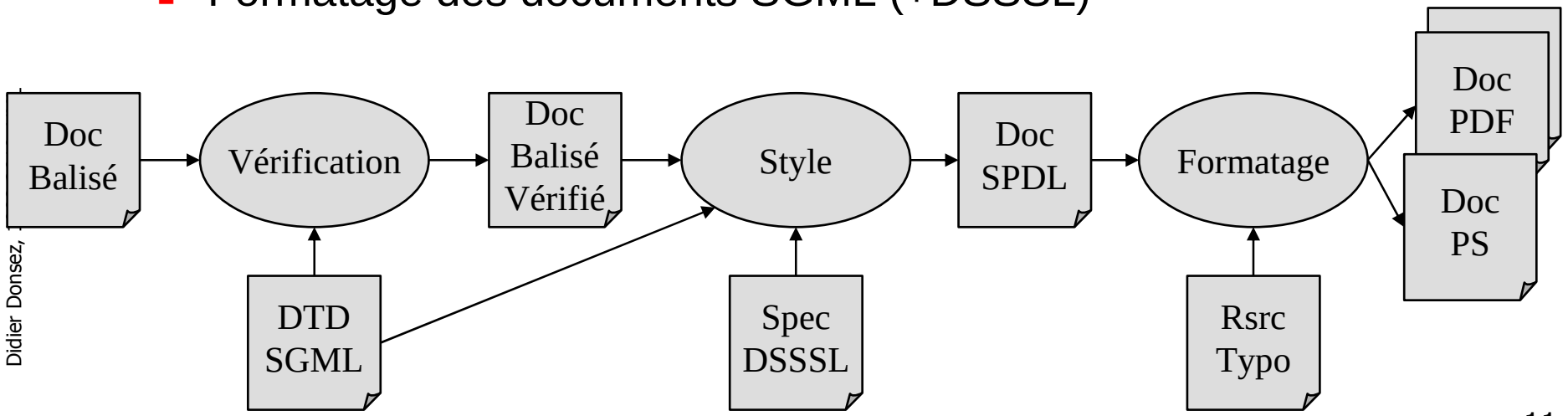
Un autre exemple l'EDI (Electronic Data Interchange)

- Syntaxe de représentation
  - ASN 1
- Standard de représentation
  - UN / EDIFACT : (United Nations)
    - Syntaxe et Bibliothèque de Documents Normalisés
    - par secteur d'activité, par pays, international
  - ANSI X12



Un peu d 'histoire
Les Principes de SGML and Co

- *Un ensemble de standards pour les documents structurés*
- SGML [Goldfarb, 74] : *Standard Generalized Markup Language (ISO 8879)*
  - Description de la structure d 'un type de document
  - Les documents de ce type respectent la structure
- DSSSL : *Document Style Semantics and Specification Language*
  - Feuille de style associée à un type de document
- SPDL : *Standard Page Description Langage*
  - Formatage des documents SGML (+DSSSL)

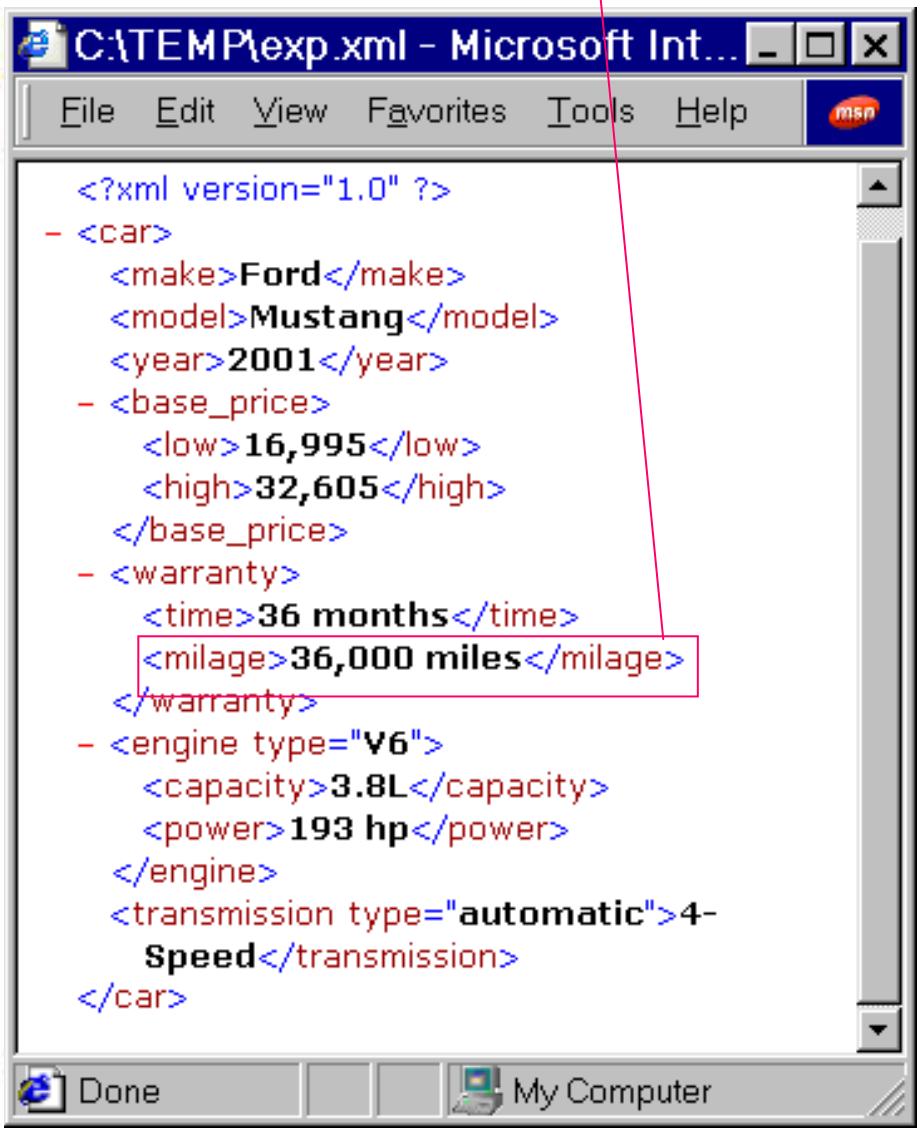


Limites de HTML

- HTML 4.0 et DHTML
  - extensibilité par la normalisation
  - pas de validation de la structure
- SGML
  - structure « sémantique » et type de document (DTD)
  - mais reste compliqué et non intégré aux browsers
- XML
  - principes de SGML mais simplification de la DTD

Remarque: quel est le problème ?

HTML vs. XML



XML eXtensible Markup Language

- Document Standalone
  - Les documents sont bien formés (*well formed*),
- Document Type Definition, DTD
  - Typage de documents structurés et extensibles
    - Les documents sont valides (*valid*)
- XML Schema
  - Schema des documents
    - Exprimé en syntaxe XML et extensible (contrairement à la DTD)
- Extensible Stylesheet Language, XSL
  - Transformation et réorganisation d 'un document XML
- Xlink, XPointers
  - Hyperliens évolués

Les Documents XML

- Ils sont structurés en élément
- Ils doivent être **Bien Formés** (*Well formed*)
  - balisage à la SGML
  - cependant `<element>` et `</element>` obligatoires
  - sauf `<empty_element/>`
  - la valeur des attributs est toujours entre double cote `<element attr="valeur"/>`
- Ils peuvent être **Validés** (*Valid*)
  - Utilisé une DTD (Document Type Definition)
    ou un schéma XML
    qui définit la structure d 'un type de document
    - DTD XML : Simplification de la grammaire SGML des DTDs
      - Conformance: pour toute DTD XML , il existe une DTD SGML conforme

Exemple de documents XML (i)

- Un document XML bien formé

```
<?xml version="1.0"  standalone="yes"?>
<greeting>Hello,<br/>world!</greeting>
```

- Un document XML valide incluant une DTD

```
<?xml version="1.0" encoding="UTF-8" ?>
 <!DOCTYPE greeting [
    <!ELEMENT greeting (#PCDATA|br)*>
    <!ELEMENT br EMPTY>
 ]>
<greeting>Hello,<br/>world!</greeting>
```

Exemple de documents XML (ii)

- Un document XML valide utilisant un DTD externe

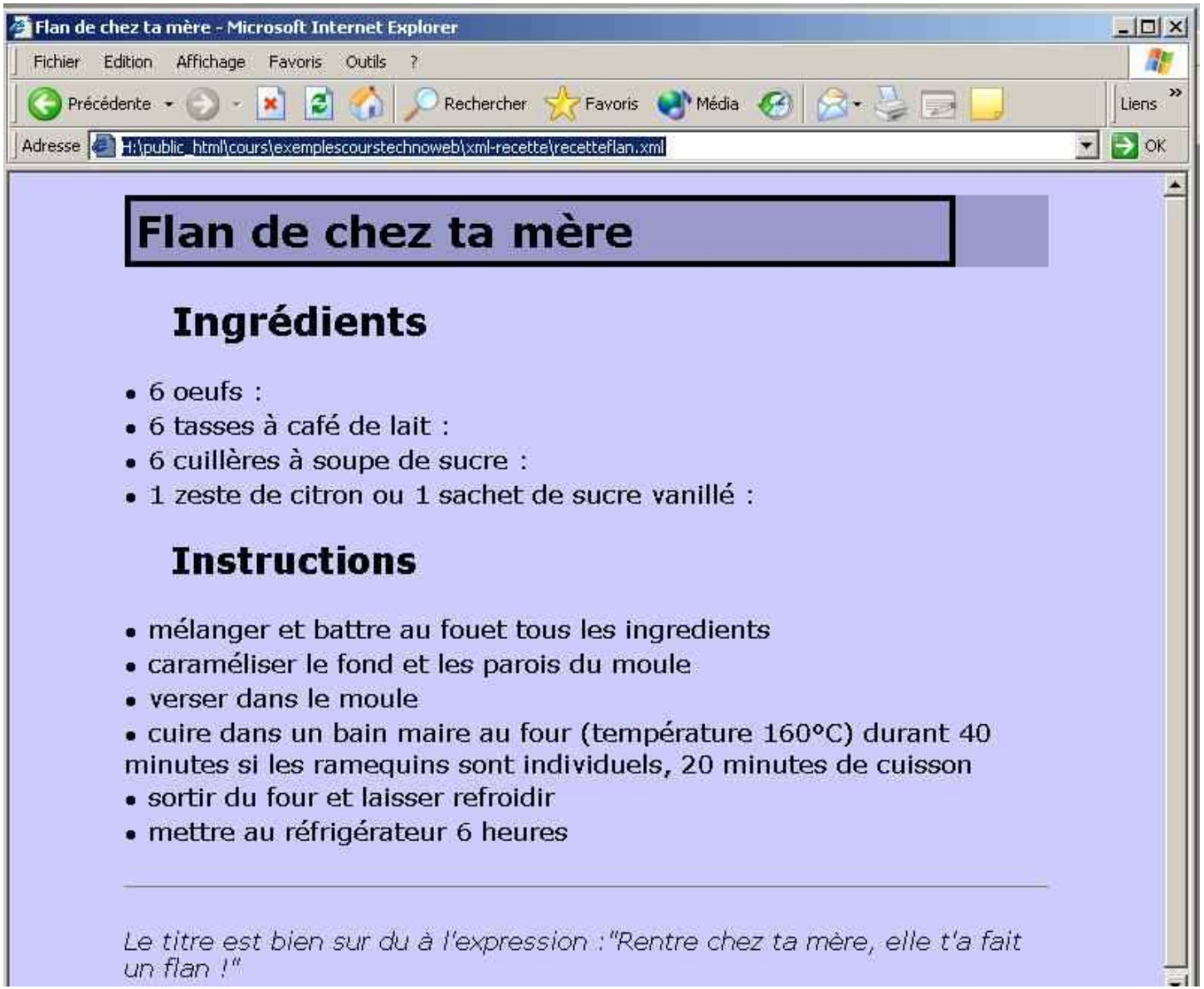
```
<?xml version="1.0"?>
 <!DOCTYPE greeting SYSTEM "hello.dtd">
<greeting> Hello,<br/>world!</greeting>
```

- Un document XML valide utilisant un schema XML

```
<greeting
   xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation=greeting.xsd'>
<greeting> Hello,<br/>world!</greeting>
```

Exercice

Donnez un document XML représentant cette recette



Généralités sur les documents XML

- Les caractères
  - codage Unicode (34168 symboles de 24 alphabets)
  - les parsers traitent au minimun UTF-8 et UTF-16
- Les espaces
  - les caractères ‘ ‘ , ‘ \r ’, ‘ \n ’, ‘ \t ’ sont considérés comme des séparateurs (texte XML « au kilomêtre »)
- Les identificateurs
  - respect de la casse !
- Les commentaires

```
<!-- Ceci est un commentaire
     bla bla bla
qui se termine ici -->
```

La structure d ’un document

- Le prologue
- La DTD
- Les éléments
- Les balises (tags)
- Les attributs
- Les références aux entités
- Les références aux caractères
- Les sections CDATA
- Les données

La structure d ’un document XML

- Le prologue

  <?xml version="1.0"?>

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

  spécifie le type d ’encodage du texte

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

  le document est standalone :

  - il n ’a pas de DTD cependant le balisage doit être équilibré

- Les instructions de traitements

  - instructions destinées aux applications qui traiteront le document

  <?application instruction+ ?>

  Exemple avec les feuilles de style

  <?xml-stylesheet type="text/xsl" href="article2xhtml.xsl"?>

  <?xml-stylesheet type="text/css" href="article.css"?>

La structure d ’un document XML

- La déclaration du type DOCTYPE
  - Référence un fichier contenant la DTD Privé

```
<!DOCTYPE greeting SYSTEM "hello.dtd">
<!DOCTYPE greeting SYSTEM "http://www.mycomp.com/dtds/hello.dtd">
```

  - Référence un fichier contenant la DTD public

```
<!DOCTYPE commande PUBLIC
        "-//YADETOU ONLINE SA//DTD des commandes//FR"
        "http://www.yadetou.fr/dtds/commande.dtd">
```

type//propriétaire//motclés description//langue

  - Inclusion de la DTD

```
<!DOCTYPE greeting [
        <!ELEMENT greeting (#PCDATA|br)*>
        <!ELEMENT br EMPTY>
 ]>
```

La structure d ’un document XML
La DTD : *Document Type Definition*

- La déclaration des notations
  - identifie un information non-XML (donc non parsé)

```
<!NOTATION gzip SYSTEM "gzip.exe">
<!NOTATION compress SYSTEM "compress.exe">
```

La structure d 'un document XML

La DTD : *Document Type Definition*

- La déclaration des entités
  - Entité générale interne
    - uniquement utilisé dans le document sous la forme **&titre;**

    <!ENTITY titre "XML pour les nuls">
  - Entité paramêtre interne
    - uniquement dans la DTD

    <!ENTITY %attname "name ID #REQUIRED">
  - Entité paramêtre externe
    - uniquement dans la DTD

    <!ENTITY %nouveau SYSTEM "nouveau.txt">
  - Entité générale externe
    - uniquement dans la DTD

    <!ENTITY arch PUBLIC "archive.z" NDATA gzip>

La structure d 'un document XML

La DTD : *Document Type Definition*

- La déclaration des éléments
  - Contenu Vide

    <!ELEMENT ident EMPTY>
  - Contenu quelconque

    <!ELEMENT ident ANY>
    - contient une structure XML quelconque (mais équilibrée)
  - Contenu éléments fils

    <!ELEMENT ident ((fils1+,fils2*|fils3+)|(fils4,fils5?))>
    - contient des éléments
    - régit par un expression régulière
      - , séquence ; | alternative ; multiplicateur * 0-N ; + 1-N ; ? 0-1
  - Contenu mixte

    <!ELEMENT ident (#PCDATA)>

    <!ELEMENT ident (#PCDATA|fils+)>

La structure d ’un document XML

La DTD : *Document Type Definition*

- La déclaration des attributs

```
<!ATTLIST nomelement
   nomattribut type défaut
   nomattribut type défaut
   ...
>
```

- Défaut

#IMPLIED	Optionnel
#REQUIRED	Obligatoire
#FIXED 'valeur'	Fixé à la valeur
'valeur'	Valeur par défaut

La structure d 'un document XML

La DTD : *Document Type Definition*

- Types d'attribut (i)
  - Types chaîne de caractères **CDATA**

    <!ATTLIST dossier chemin CDATA #IMPLIED >
  - Type Prédéfini **ENTITY, ENTITIES**
    - nom (ou liste de noms) d entités non XML précédemment déclarées
  - Type Prédéfini **NMTOKEN, NMTOKENS**
    - nom (ou liste de noms) de mots clé

    <!ATTLIST fichier xml:lang NMTOKEN #REQUIRED 'fr'>
  - Valeurs énumérées

    <!ATTLIST archive codage NOTATION (gzip|compress) #REQUIRED>

    <!ATTLIST archive codage (gzip|compress) 'gzip'>

Exercice
Donnez la DTD de ce document recette

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE recette SYSTEM "http://www.cook.org/dtds/recette.dtd">
<recette titre="Flan de chez ta mère" auteur="Didier Donsez">
<ingredients>
  <ingredient mesure="6">oeuf</ingredient>
  <ingredient mesure="6" unite="tasse à café">lait</ingredient>
  <ingredient mesure="6" unite="cuillère à soupe">sucre</ingredient>
  <ingredient mesure="1" unite="zeste">citron</ingredient>
</ingredients>
<instructions>
  <instruction>mélanger et battre au fouet tous les ingrédients</instruction>
  <instruction>caraméliser le fond et les parois du moule</instruction>
  <instruction>verser dans le moule</instruction>
  <cuisson mode=" bain maire au four" temperature="160°C" duree="40 minutes"/>
  <instruction>sortir du four et laisser refroidir au réfrigérateur 6 heures</instruction>
</instructions>
<remarque>Rentre chez ta mère, elle t'a fait un flan !</remarque>
</recette>
```

Réponse : la DTD recette.dtd

```
<!ELEMENT recette (ingredients,instructions,remarque?)>
<!ELEMENT ingredients (ingredient+)>
<!ELEMENT instructions (instruction|cuisson)+>
<!ELEMENT ingredient (#PCDATA)>
<!ELEMENT instruction (#PCDATA)>
<!ELEMENT cuisson EMPTY>
<!ELEMENT remarque (#PCDATA)>
<!ATTLIST recette
    titre CDATA #REQUIRED
    auteur CDATA #IMPLIED >
<!ATTLIST ingredient
    mesure CDATA #REQUIRED
    unite CDATA 'unite‘ >
<!ATTLIST cuisson
    mode CDATA #REQUIRED
    temperature CDATA #REQUIRED
    duree CDATA #REQUIRED >
```

La structure d ’un document XML
La DTD : *Document Type Definition*

- Types d’attribut (ii)
  - Type Prédéfini **ID**
    - Identifiant unique dans le document

    <!ATTLIST dossier reference ID #REQUIRED >
  - Type Prédéfini **IDREF, IDREFS**
    - référence (ou liste de réf) d ’identifiants ID du document

    <!ATTLIST fichier dossier IDREF #REQUIRED >

Exercice

1. Modifiez le document recette pour que les instructions référencent les éléments ingrédients
2. Donnez ensuite la nouvelle DTD

Réponse 1 : le nouveau document

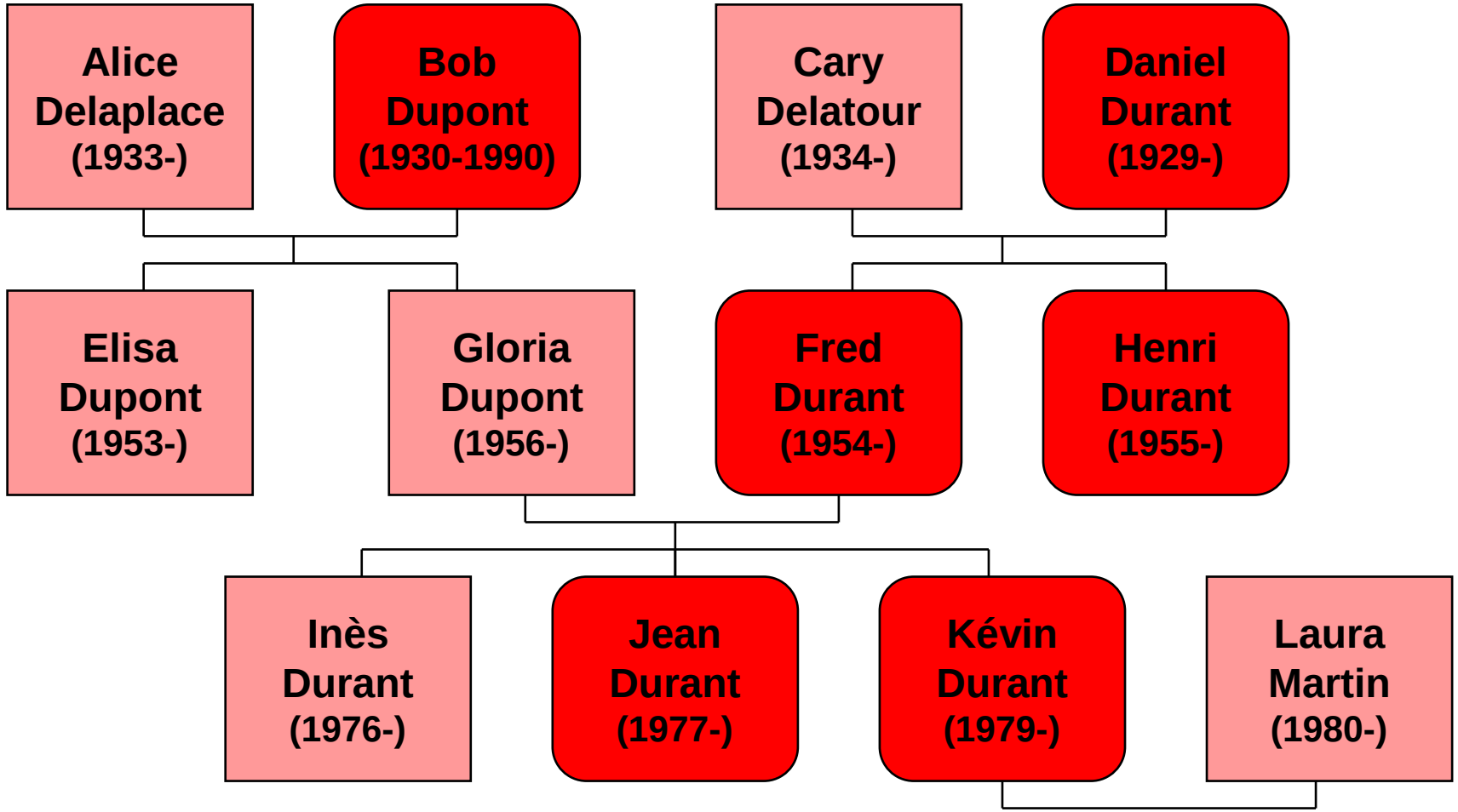
```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE recette SYSTEM "http://www.cook.org/dtds/recette.dtd">
<recette titre="Flan de chez ta mère" auteur="Didier Donsez">
<ingredients>
  <ingredient ident="i1" mesure="6">oeuf</ingredient>
  <ingredient ident="i2" mesure="6" unite="tasse à café">lait</ingredient>
  <ingredient ident="i3" mesure="6" unite="cuillère à soupe">sucre</ingredient>
  <ingredient ident="i4" mesure="1" unite="zeste">citron</ingredient>
</ingredients>
<instructions>
  <instruction idents="i1,i2,i3,i4">mélanger et battre au fouet tous les
     ingrédients</instruction>
  <instruction idents="i3">caraméliser le fond et les parois du moule</instruction>
  <instruction>verser dans le moule</instruction>
  <cuisson mode=" bain maire au four" temperature="160°C" duree="40 minutes"/>
  <instruction>sortir du four et laisser refroidir au réfrigérateur 6 heures</instruction>
</instructions>
<remarque>Rentre chez ta mère, elle t'a fait un flan !</remarque>
</recette>
```

Réponse 2 : la nouvelle DTD recette.dtd

```
...
<!ATTLIST ingredient
    mesure CDATA #REQUIRED
    unite CDATA 'unite'
    ident ID #REQUIRED
>
<!ATTLIST instruction
    idents IDREFS #IMPLIED
>
...
```

Exercice

Donnez la DTD et les éléments de ce graphe



La structure d 'un document XML
La DTD : *Document Type Definition*

- Les sections conditionnelles
  - versions alternatives de structures dans un même DTD

```
<![%developpement;[      <!ELEMENT dossier ANY>  ]]>
<![%final;[              <!ELEMENT dossier (fichier*|archive*)>      ]]>
```

  - dépend de la déclaration d 'entités

```
<!ENTITY %developpement 'INCLUDE'>
<!ENTITY %final 'IGNORE'>
<![INCLUDE[      <!ELEMENT dossier ANY>              ]]>
<![IGNORE[       <!ELEMENT dossier (fichier*|archive*)>      ]]>

<!ENTITY %developpement 'IGNORE'>
<!ENTITY %final 'INCLUDE'>
<![IGNORE[       <!ELEMENT dossier ANY>              ]]>
<![INCLUDE [     <!ELEMENT dossier (fichier*|archive*)>      ]]>
```

La structure d ’un document XML
La DTD : *Document Type Definition*

- La gestion des espaces
  - permet de préserver les espaces pour la présentation de l ’élément

```
<!ATTLIST fichier
  xml:space (default|preserve)  ’preserve'
>
```

- L ’identification de la langue
  - déclare la langue utilisé dans le contenu de l ’élément

```
<!ATTLIST fichier
  xml:lang NMTOKEN #FIXED  'fr'
>
<!ATTLIST file
  xml:lang NMTOKEN #FIXED  'en-GB'
>
```

La structure d ’un document XML

- Les éléments balisés (tags) et leurs attributs

```
<dossier id="123"  chemin="c:\doc\comptable" >
  <fichier nom="commande11022000.txt">
        des chiffres pour la commande du livre &lt;&titre;&gt;
  </fichier>
  <fichier nom="commande12022000.txt">
        d ’autres chiffres
 </fichier>
  <alias ref ="123"/>
</dossier>
```

- Les références aux entités et aux caractères

```
&lt;   &titre;  &gt;
```

- Les sections CDATA (non parsés)

```
<![CDATA[dans le chapître sur <?xml version="1.0"?> ]]>
```

SDD : Standalone Document Declaration

- Omission de la DTD
  - même si le document n 'appartient à aucun type de document
  - il doit néanmoins être bien formé
    - équilibrage des start tags et end tags
    - et le tag <emptyelement/> pour les éléments vides
- Exemple

```
<?xml version="1.0" standalone="yes"?>
    <conversation>
      <greeting>Hello, world!</greeting>
      <response>
                  &gt;Stop the planet <pic src="earth"/>,
                  I want to get off!&lt;
       </response>
    </conversation>
```

XML NameSpaces

- Collection d ’identificateurs (élément ou attribut)
  - identifié par un préfixe et une URI
- Déclaration

  <element xmlns:prefixe=URI>
  - Exemple

  <html xmlns='http://www.w3.org/TR/REC-html40'
  xmlns:order='http://www.mycomp.com/order-spec'>
  - le premier namespace (html) est celui par défaut

XML NameSpaces

```
<?xml version="1.0"?>
<html xmlns='http://www.w3.org/TR/REC-html40'
        xmlns:order='http://www.mycomp.com/order-spec'>
<table border="1">
<th>Gencod</th><th>Quantity</th>
<tr> <td> <order:gencod>14255637</order:gencod></td>
     <td> <order:quantity order:value="10"/></td> </tr>
<tr> <td> <order:gencod>84975388</order:gencod></td>
     <td> <order:quantity order:value="20"/></td> </tr>
</table>
</html>
```

XML NameSpaces

- Déclaration dans une DTD
  - le namespace est déclaré comme une constante

```
<!ELEMENT order:gencod (#PCDATA)>
<!ATTLIST order:gencod
  xmlns:order CDATA #FIXED "http://www.mycomp.com/order-spec">
>
```

Portée d ’un NameSpaces

- Un autre déclation de nom de domaine remplace le préédent dans les éléments inclus et ses descendants

- Exemple

```
<?xml version="1.0"?>
  <bk:book xmlns:bk='urn:loc.gov:books'
           xmlns:isbn='urn:ISBN:0-395-36341-6'>
     <bk:title>Cheaper by the Dozen</bk:title>
     <isbn:number>1568491379</isbn:number>
  </bk:book>
```

XML Catalog

- Catalogue de DTD Publiques

XML Schema
le remplacant de la DTD

- Limite des DTD
  - description limité des documents structures
    - pas de typage des PCDATA
    - Limite
  - exprimé dans un langage autre XML
    - autre parser, ...
- Objectif de XML Schema
  - Amélioration du typage : xsi
    - données obéïssant à une expression régulière
  - Amélioration de la structure : xsd
    - nombre borné d 'éléments inclus, ...
- Remplace de plus en plus la DTD

Les 2 parties de XML Schema

- Datatype (Partie 2)
  - définit un riche ensemble de types prédéfinies
    - incluant celui des langages de SGBD, ...
- Structure (Partie 1)
  - définit la construction de types (dit archetype)
    - simples ou complexes
    - nommés ou anonymes
    - mécanismes d 'héritage (refinable archetype)
  - support pour les namespaces
    - validation de documents utilisant plusieurs namespaces

XML Schema - XSI

- Types primitifs

```
<element name="price" type="float"/>
<element name="greeting"  type="xsd:string"/>
<price>15.57</price>
<greeting id="id1">Hello</greeting>
```

- Structures

```
<element name="Book">
<complexType>
        <element name="author" type="xsd:string"/>
        <element name="title" type="xsd:string"/>
</complexType>
</element>
<e:Book>
<author>J.R.R Tolkien</author>
<title>A hobbit story</title>
</e:Book>
```

XML Schema - XSI

- Enumération

```
<element name="color">
 <simpleType base="xsd:string">
  <enumeration value="Green"/>
  <enumeration value="Blue"/>
 </simpleType>
</element>
```

<color>Blue</color>

<color>Red</color> Invalide

RELAX NG http://www.relaxng.org

- simple schema language for XML based on
  - RELAX (Regular Language description for XML)
  - and TREX (Tree Regular Expressions for XML)
- 2 Syntax
  - XML and Compact
- To Read
  - Tutorial
    - http://www.relaxng.org/compact-tutorial-20030326.html
  - Eric van der Vlist, RELAX NG, OReilly, Online book, ISBN: 0596004214
    - http://books.xmlschemata.org/relaxng/

Exemple RELAX NG (i)

```
<!DOCTYPE addressBook [
<!ELEMENT addressBook (card*)>
<!ELEMENT card (name, email)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
]>
```

```
element addressBook {
  element card {
    element name { text },
    element email { text }
  }*
```


From http://www.relaxng.org/compact-tutorial-20030326.html

Exemple RELAX NG (ii)

```
<!DOCTYPE addressBook [
<!ELEMENT addressBook (card*)>
<!ELEMENT card ((name | (givenName, familyName)), email, note?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT givenName (#PCDATA)>
<!ELEMENT familyName (#PCDATA)>
<!ELEMENT note (#PCDATA)>
]>
```

```
element addressBook {
  element card {
    (element name { text }
     | (element givenName { text },
        element familyName { text })),
    element email { text },
    element note { text }?
  }*
}
```


From http://www.relaxng.org/compact-tutorial-20030326.html

L’outillage XML

L'outillage XML

- XPath
- XQuery
- XSL

- APIs d'analyse
  - SAX
  - DOM
- API de construction
- DOM

XPath

- Recommandation du W3C
- Expression de désignation d'un nœud dans un document XML
  - syntaxe simple et non ambiguë
    - type usuels: chaînes, nombres, booléens, variables, fonctions
  - spécifie une bibliothèque de fonctions extensible
    - position(), …
- Exemples
  - **/chapter[@type="warning"]**
  - **//p[position()=5]**
- Utilisé par XPointer et XSLT
- Implémentation
  - Package Xpath org.apache.xalan.xpath

Expressions XPath (i)

/	parenté

section/paragraphe désigne les <**paragraphe**> fils des <**section**> de l ’élément courant.

//	parenté étendu aux aïeux
.	nœud courant
..	père du nœud courant
|	alternative
@zzz	attribut zzz du nœud courant
text()	nœuds CDATA fils du nœud courant
comment()	nœuds commentaires fils du nœud courant
pi()	instructions de traitement du nœud courant
id()	sélection sur identifiant ou liste d ’identifiants

Expressions XPath (ii)

- Condition de sélection [ ]
  - **section[@titre]** <section> qui ont un attribut titre
  - **section[paragraphe]** <section> qui ont au moins un fils <paragraphe>
  - **section[@titre='Introduction']** <section> qui ont un attribut titre dont la valeur est "Introduction"
  - **section[paragraphe='Introduction']** <section> qui ont au moins un fils <paragraphe> dont la valeur est "Introduction"
- Positionnement
  - position()=5 vrai si l'élément est à la 5ème position
  - **first-of-any()** vrai si l 'élément est le premier fils
  - **last-of-any()** vrai si l 'élément est le dernier fils
  - **first-of-type()** vrai si l 'élément est le premier fils de son type
  - **fisrt-of-type()** vrai si l 'élément est le dernier fils de son type

Expressions XPath (iii)

- Connecteurs logiques (not,and,or)
  - **section[not(@titre)]** <section> qui n ’ont pas d ’attribut titre
  - **section/paragraphe[first-of-type() and last-of-type()]** <paragraphe> qui sont fils unique !

Exemple de recherche Xpath avec Xalan

```
import …;
public class ApplyXPath { // voir xalan/samples/ApplyXPath
public static void main (String[] args) { try {
   String filename = args[0];
   String xpath = args[1];          // exemple : /peoples/people[1]/name/@last
    InputSource  in = new InputSource(new FileInputStream(filename));
    DOMParser parser = new DOMParser();
    parser.parse(in);
    Node root = parser.getDocument().getDocumentElement();
    NodeList nl = XPathAPI.selectNodeList(root, xpath);
    FormatterToXML fl = new FormatterToXML(System.out);
    TreeWalker tw = new TreeWalker(fl);
    for(int i = 0; i < nl.getLength(); i++)  {
        tw.traverse(nl.item(i)); fl.flush(); fl.flushWriter();
    }
}  catch(Exception e) { e.printStackTrace(); return; } } }
```

Xquery
http://www.w3.org/TR/xquery

- Motivation
  - Étendre la désignation XPath 1.0
    - Namespace
      - dc:Creator tous les éléments <Creator> appartenant au namespace dc:
      - chapter/@li:title : l 'attribut title de préfixe li des éléments <chapter>
    - Collections d 'attributs
      - @* : tous les attributs de l 'élément courant
      - @li:* : tous les attributs de l 'élément courant identifiés par le préfixe li
    - Comparaisons complexes
    - Opérateurs any et all
    - …
  - Construction de documents (alternative et complément à XSLT)
- Principe
  - Langage fonctionnel semblable à SQL, OQL, …

Exemple XQuery

```
<bib>
        <book year="1994">
          <title>TCP/IP Illustrated</title>
          <author><last>Stevens</last><first>W.</first></author>
          <publisher>Addison-Wesley</publisher>
          <price>65.95</price>
        </book>
        <book year="1992">
          <title>Advanced Programming in the Unix environment</title>
          <author><last>Stevens</last><first>W.</first></author>
          <publisher>Addison-Wesley</publisher>
          <price>65.95</price>
        </book>
        <book year="2000">
          <title>Data on the Web</title>
          <author><last>Abiteboul</last><first>Serge</first></author>
          <author><last>Buneman</last><first>Peter</first></author>
          <author><last>Suciu</last><first>Dan</first></author>
        </book>
           ... </bib>
```

Exemple Xquery
Recherche étendue

```
document("bib.xml")//book[
    some $l in ./author/last satisfies string(data($l)) = "Stevens"]
```

Exemple Xquery
Construction d'éléments

```
<results>
  {
    for $book in //book
    let $authors := $book/author
    return
      <book authorCount
          ={ count($authors)}>
      {
        for $author in $authors
        return
          <author>
                  { $author/last/text() }
          </author>
      }
      </book>
  }
</results>
```

```
<results>
      <book authorCount="1">
        <author>Stevens</author>
      </book>
      <book authorCount="1">
        <author>Stevens</author>
      </book>
      <book authorCount="3">
        <author>Abiteboul</author>
        <author>Buneman</author>
        <author>Suciu</author>
      </book>
        ...
</results>
```

eXtensible Linking Language, XLL

- Limite des références (URL) en HTML
  - ne référence qu ’un seul document
  - granularité grosse de référence : le document
    - accèdes au 3ème paragraphe qui suit le chapitre nommé "Introduction"
  - pas de relations entre les documents.
  - référence à sens unique.
- Les 2 parties de XLL
  - Xlink : XML Linking Specification
  - XPointer : XML Extended Pointer Specification

Xpointer
XML Extended Pointer Specification

- adresse des objets internes à la structure du document et ne possédant pas d 'identifiant
- Utilise les expressions Xpath
- Exemple
  - **http://www.ucc.ie/xml/faq.sgml#ID(faq-hypertext)CHILD(2,*)(6,*)**
  - désigne le 6ème objet inclus dans le 2ème objet inclus dans l 'élément ayant un ID égale « faq-hypertext ».

- A lire
  - *Recommandation XPointer du W3C et le Chapitre 17 de « XML Bible »*

Xpointer
XML Extended Pointer Specification

- Exemple

  http://www-adele.imag.fr/~donsez/exemple/coursxml.xml

  - **#Introduction**
  - **#xpointer(id("Introduction"))**
  - **#xpointer(/chapitre[2]/p[3])**
  - **#xpointer(//chapitre[titre="Introduction"]/ descendant::p[position()=last()])**
  - **#xpointer(id("sec2.1")//p[2] to id("sec2.2")//p[last()])**
  - **#xpointer(string-range(//title,"Cours XML"))**

XLink : XML Linking Specification

- spécifie les hyperliens dans un document XML
  - liens simples, type <A href="...">
  - liens étendus: multisource, multicible, externes
    - multi-direction, multi-cible, indépendance à la localisation (en cas de changement de place du document), transclusion (document inclus), liens typés (attributs de lien).

- A lire
  - *Recommandation XLink du W3C et le Chapitre 16 de « XML Bible »*

XLink : XML Linking Specification

- un lien simple:

```
<students xlink:href="students.xml"> The list of students.</students>
```

- un lien étendu:

```
<element xmlns:xlink="http://www.w3.org/1999/xlink/namespace/"
        xlink:type="extended">
 <xlink:locator href="Source" role="role1"/>
 <xlink:locator href="Target" role="role2"/>
 <xlink:arc from="role1" to="role2" show="embed" actuate="auto"/>
 <xlink:title>The link title<xlink:title/>
 <xlink:title xml:lang="fr">Description du lien<xlink:title/>
   ...
</element>
```

- Annotations:

```
<xlink:extended xmlns:xlink="http://www.w3.org/1999/xlink/namespace/"
        role="xlink:external-linkset">
    <xlink:title>DV's Links</xlink:title>
    <xlink:locator href="http://rpmfind.net/veillard/linkset.xml"/>
</xlink:extended>
```

Xinclude *(http://www.w3.org/TR/xinclude/)*

- Inclusion de documents lors de l'analyse
  - Modularité, réutilisabilité …
- Exemple

sub1.txt

```
Sentence 0
```

sub2.xml

```
<?xml version='1.0'?>
<chapter>
  <p>Sentence 1.  Sentence 2.</p>
  <p><i>Sentence 3.  Sentence 4.</i>  Sentence 5.</p>
</chapter>
```

doc.xml

```
<?xml version='1.0'?>
<document xmlns:xi="http://www.w3.org/2001/XInclude">
  <p>The texte document contains :  </p>
  <pre>
  <xi:include href="sub1.txt" parse="text"/>
  </pre>
  <p> The XML document contains :</p>
  <xi:include href="sub2.xml#xpointer(chapter/p[1])"/>
</document>
```

eXtensible Stylesheet Language, XSL

- Définition de feuilles de style
  - Basé sur ISO/IEC 10179 DSSSL
    - Document Style Semantics and Specification Language
- Fonctionnalités proposées
  - fonctions simples
  - ré-ordonnancement des éléments
  - formatage des éléments en fonction de leur position, de leur unicité ou par rapport aux éléments parents et à leur descendance.
  - formatage incluant des textes et des graphiques,
  - macros de formatage réutilisables
  - …
- 2 parties
  - XSL-T Transformation (de XML vers XML *[XHTML,WML,SVG,…]*)
  - XSL-FO Formatting Object (de XML vers non-XML *[PS,PDF,RTF,…]*)

XSLT - XSL Transformation

- Motivations
  - appliquer des transformations à un document XML
    - ré-ordonnancement des éléments, masquage d ’éléments, ...
  - vers un autre document XML
    - *en général visualisable comme XHTML, WML, SVG, VRML, SMIL …*
- Avantage
  - un document XML peut être subir des transformations différentes

Remarques sur XSLT

- Remarque sur la maturité :
  - Attention :les transformateurs ajoutent parfois des fonctionnalités et des éléments supplémentaires.
    - XALAN autorise l'appel à des « fonctions » Java
- Remarque à propos des CSS :
  - quand la cible est du HTML4/XHTML1,
    XSLT doit être utilisé de manière combiné avec les CSS
- Remarque à propos du formatage :
  - XSLT peut être utilisé comme étape dans le formatage vers des documents non-XML *(RTF, Word, PostScript, PDF, LaTex, …)*
    - voir FOP (Formatting Object processor)

Principe de la transformation de XSLT

- Appel à la transformation
  - instruction de traitement dans le document source XML

    <?xml-stylesheet type="text/xsl" href="article2xhtml.xsl"?>
- La feuille de style xsl:stylesheet
  - liste ordonnée de *templates* xsl:template qui sont appliqués aux nœuds de l ’arbre source
- Un template xsl:template
  - sous arbre du document de sortie comportant des éléments de DTD de sortie et des éléments xsl:
- Règle xsl:template
  - détermine si le template est applicable à un nœud de l ’arbre source

Un exemple de transformation XSLT

- ***Soit un document source XML***

Instruction de traitement

```
<?xml version="1.0"?>
<!DOCTYPE sales SYSTEM  "sales.dtd">
<?xml-stylesheet type="text/xsl" href= "sales2html.xsl"?>
<sales>
        <division id="North">
                <revenue>10</revenue><growth>9</growth><bonus>7</bonus>
        </division>
        <division id="South">
                <revenue>4</revenue><growth>3</growth><bonus>4</bonus>
        </division>
        <division id="West">
                <revenue>6</revenue><growth>-1.5</growth><bonus>2</bonus>
        </division>
</sales>
```

Un exemple de transformation XSLT pour une sortie HTML *(pour IE5.0)*

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
  <html><head><title>Sales Results By Division</title> </head><body>
  <xsl:apply-templates/>
 </body></html>
</xsl:template>

<xsl:template match="sales">
      <table border="1">
        <tr><th>Division</th><th>Revenue</th><th>Growth</th><th>Bonus</th></tr>
        <xsl:apply-templates select="division"/>
      </table>
</xsl:template>

<xsl:template  match="sales/division">
    <tr><td><em><xsl:value-of select="@id"/></em></td><td><xsl:value-of select="revenue"/></td>
      <td><xsl:value-of select="growth"/></td><td><xsl:value-of select="bonus"/></td>
    </tr>
</xsl:template>

<xsl:template match="*"><xsl:apply-templates/></xsl:template>
</xsl:stylesheet>
```

élément root

Appliqué les templates aux sous-arbres

élément(s) sales

élément(s) division

tout élément

Un autre exemple de transformation XSLT pour une sortie HTML

```
<html
        xsl:version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <head><title>Sales Results By Division</title></head><body>
  <table border="1">
      <tr><th>Division</th><th>Revenue</th><th>Growth</th><th>Bonus</th></tr>
      <xsl:for-each select="sales/division"><!-- order the result by revenue -->
        <xsl:sort select="revenue" data-type="number" order="descending"/>
        <tr><td><em><xsl:value-of select="@id"/></em></td>
          <td><xsl:value-of select="revenue"/></td>
          <td><xsl:if test="growth &lt; 0"><!-- highlight negative growth in red -->
            <xsl:attribute name="style"><xsl:text>color:red</xsl:text></xsl:attribute
            </xsl:if>
            <xsl:value-of select="growth"/>
          </td>
         <td><xsl:value-of select="bonus"/></td>
        </tr>
      </xsl:for-each>
  </table>
</body></html>
```

Le résultat de la transformation vers HTML

```
<html>
<head><title>Sales Results By Division</title></head>
<body>
<table border="1">
<tr><th>Division</th><th>Revenue</th><th>Growth</th><th>Bonus</th></tr>
<tr><td><em>North</em></td><td>10</td><td>9</td><td>7</td></tr>
<tr><td><em>West</em></td><td>6</td><td style="color:red">-1.5</td><td>2</td></tr>
<tr><td><em>South</em></td><td>4</td><td>3</td><td>4</td></tr>
</table>
</body></html>
```

Un exemple de transformation XSLT pour une sortie SVG

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
            xmlns="http://www.w3.org/Graphics/SVG/SVG-19990812.dtd">
<xsl:output method="xml" indent="yes" media-type="image/svg"/>
<xsl:template match="/">
<svg width = "3in" height="3in">
   <g style = "stroke: #000000">
      <line x1="0" x2="150" y1="150" y2="150"/><line x1="0" x2="0" y1="0" y2="150"/>
      <text x="0" y="10">Revenue</text><text x="150" y="165">Division</text>
      <xsl:for-each select="sales/division">
         <xsl:variable name="pos" select="(position()*40)-30"/> <!-- the bar's x position -->
         <xsl:variable name="height"  select="revenue*10"/> <!-- the bar's height -->
         <rect x="{$pos}" y="{150-$height}" width="20" height="{$height}"/>
         <text x="{$pos}" y="165"><xsl:value-of select="@id"/></text>
         <text x="{$pos}" y="{145-$height}"><xsl:value-of select="revenue"/></text>
      </xsl:for-each>
   </g></svg>
</xsl:template></xsl:stylesheet>
```

Le résultat de la transformation vers HTML

```
<svg width="3in" height="3in »
        xmlns="http://www.w3.org/Graphics/SVG/svg-19990412.dtd">
  <g style="stroke: #000000">
    <line x1="0" x2="150" y1="150" y2="150"/>
    <line x1="0" x2="0" y1="0" y2="150"/>
    <text x="0" y="10">Revenue</text>
    <text x="150" y="165">Division</text>
    <rect x="10" y="50" width="20" height="100"/>
    <text x="10" y="165">North</text>
    <text x="10" y="45">10</text>
    <rect x="50" y="110" width="20" height="40"/>
    <text x="50" y="165">South</text>
    <text x="50" y="105">4</text>
    <rect x="90" y="90" width="20" height="60"/>
    <text x="90" y="165">West</text>
    <text x="90" y="85">6</text>
  </g>
</svg>
```

Un exemple de document XML

```
<?xml version="1.0" encoding="UTF-8" ?>
 <!DOCTYPE table [
        <!ELEMENT       table     ((table|row)*)>
        <!ELEMENT       row       ((table|col)+)>
        <!ELEMENT       col       (#PCDATA)>
        <!ATTLIST       row|col   name CDATA #REQUIRED>
 ]>
<?xml-stylesheet type="text/xsl" href="table2html.xsl"?>
<table>
<row>   <col name="division">Nord</col><col name="revenue">10</col></row>
<row>   <col name="division">Sud</col><col name="revenue">7</col></row>
<row>   <col name="division">EstOuest</col>
        <table>
                <row name="1999"> <col name="revenue">6</col></row>
                <row name="2000"> ><col name="revenue">9</col></row>
        </table>
</row>
</table>
```

Structure de <xsl:stylesheet>

- Le document XSLT doit être bien formé
  - l ’élément racine est **xsl:stylesheet**
  - nom de domaine seul xsl:

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

  - avec d ’autres noms de domaine (ici par défaut HTML)

```
<xsl:stylesheet  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                 xmlns="http://www.w3.org/TR/REC-html40"
                 result-ns="">
… </xsl:stylesheet>
```

Les éléments de <xsl:stylesheet>

- <xsl:template match="…"></xsl:template>
  - règle de transformation
- <xsl:import href="…"/>
  - importe des feuilles de style XSL
- <xsl:include href="…"/>
  - inclut des éléments XSL dans la feuille de style XSL
- <xsl:output method="*method*"/>
  - définit le format de la sortie. *method* peut être `xml, html, text`
- <xsl:macro name="…">...</xsl:macro >
  - définit une macro-instruction (groupe d ’éléments réutilisables)

Règles de transformation explicites ou *Template Rules* <xsl:template>

- Propriété match (REQUIRED)
  - la règle ne s 'applique qu 'aux éléments vérifiant le pattern défini par la valeur
  - Exemples
    - **<xsl:template match="/">**
    - **<xsl:template match="table">**
    - **<xsl:template match="table/comment()">**
    - **<xsl:template match="/table">**
    - **<xsl:template match="table[not(row)]">**
    - **<xsl:template match="row/col">**
    - **<xsl:template match= "table/*/col">**

Eléments des règles de transformation <xsl:template match="XX">

- <xsl:apply-templates select="YY">

  applique les templates aux fils du nœuds courant, restreint au pattern définit par la propriété select

- <xsl:value-of select="YY"/>

  donne la valeur du fils vérifiant le pattern définit par la propriété select

- {YY}
  - idem mais permet d 'insérer la valeur dans les balises des éléments de sortie

```
<xsl:template name="LINK">
<A HREF="{URL}"><xsl:value-of select="SITE"></A>
</xsl:template>
```

```
<LINK SITE="W3C" URL="http://www.w3.org"/>
```
est transformé
```
<A HREF="http://www.w3.org"/>W3C</A>
```

Eléments des règles de transformation <xsl:template match="XX">

- xsl:element : Insertion d ’élément
  - insère un élément dans la sortie dont le nom est une valeur de l ’entrée

```
<xsl:template match="col">
 <xsl:element name="{@name}"><xsl:value-of select="text()"/></xsl:element>
</xsl:template>
```

<COL NAME="Site"/>W3C</COL> **est transformé** <SITE>W3C</SITE>

- xsl:attribut : Insertion d ’un attribut
  - insère un élément dans la sortie dont le nom est une valeur de l ’entrée

```
<xsl:template match="LINK">
 <A>
 <xsl:attribute name="HREF"><xsl:value-of select="@URL"/></xsl:attribute>
 <xsl:value-of select="SITE"/>
 </A>
</xsl:template>
```

<LINK SITE="W3C" URL="http://www.w3.org"/> **est transformé**

<A HREF="http://www.w3.org"/>W3C</A>

Eléments des règles de transformation <xsl:template match="XX">

- xsl:attribut-set : Insertion d 'un ensemble d 'attributs

```
<xsl:attribute-set name="fontatt">
<xsl:attribute name="font-family">Times</xsl:attribute>
<xsl:attribute name="font-size">12pt</xsl:attribute>
</xsl: attribute-set>

<xsl:attribute-set name="paddingatt">
<xsl:attribute name="font-family">1pt</xsl:attribute>
</xsl: attribute-set>

<xsl:template match="COL">
    <TD xsl:use-attribute-sets="fontatt paddingatt"><xsl:value-of select="."/></TD>
</xsl:template>
```

Ainsi

<COL NAME="Site"/>W3C</COL> **est transformé en**

<TD font-family="Times" font-family="12pt" cell-padding="1pt" >W3C</TD>

Eléments des règles de transformation <xsl:template match="XX">

- <xsl:copy>
  - copie le nœud source dans la sortie

```
<xsl:template match="row">
<xsl:copy><b><xsl:value-of select="."/></b></xsl:copy>
</xsl:template>
<xsl:template match="col"><xsl:copy/></xsl:template>
```

- <xsl:for-each>
  - parcourt les nœuds
- <xsl:choose>
  - <xsl:when>
  - <xsl:otherwise>
- <xsl:if test=...>

Eléments des règles de transformation <xsl:template match="XX">

- <xsl:sort>
  - Tri des nœuds

```
<xsl:template match="table">
   <table>
   <xsl:apply-templates>
     <xsl:sort select="row/col[name='ID']"
               order="descending"
               data-type="number"
     />
   </xsl:apply-templates>
   </table>
</xsl:template>
```

Eléments des règles de transformation <xsl:template match="XX">

- <xsl:number ...>
  - Numérotation des nœuds
    - Propriétés : value, from, level, letter-value grouping-separator grouping-size

```
<xsl:template match="row">
   <td>
   <xsl:number        value="position()"
                      format="I"
   />
   </td>
   <xsl:apply-templates select="col">
</xsl:template>
```

Eléments des règles de transformation <xsl:template match="XX">

- <xsl: counter...>

Définition de constantes <xsl:variable name="XX">

- Permet de définir des constantes

```
<xsl:variable name="copyright">
  Copyright 1998-2000 Didier DONSEZ
</xsl:variable>

<xsl:variable name="uvhc">
Université de Valenciennes et du Hainaut-Cambrésis
</xsl:variable>


<xsl:template ...>
<BLOCK COPYRIGHT="{$copyright}"> … </BLOCK >

<HR><xsl:value-of select="$copyright"/>, <xsl:value-of select="$uvhc"/>
</xsl:template>
```

Règles de transformation implicites ou *Default Template Rules*

- Liste des règles implicites

  <xsl:template match="*|/"><xsl:apply-templates/></xsl:template>

  - Application des templates aux éléments et à ses descendants

  <xsl:template match="text() |@*"><xsl:value-of select="."/></xsl:template>

  - Copie les nœuds texte dans le document résultant

  <xsl:template match="processing-instruction()|comment()"/>

  - ne rien faire pour les instructions et les commentaires

- Application
  - Elles sont considérées comme importées

    Exemple

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

  - Elles sont ignorées si une règle qui concerne l ’élément est définie

Règles de transformation explicites ou *Template Rules* <xsl:template>

- Propriété mode (IMPLIED)
  - sert à appliquer des règles plusieurs fois à un même nœud mais avec des sorties différentes(règles diffèrentes)
- Exemple

```
<xsl:template match="DOC">
    <xsl:apply-templates mode="toc"/>
    <hr/><xsl:apply-templates mode="body"/>
</xsl:template>
<xsl:template match="CHAPTER" mode="toc">
    <a href="#{@ID}"><xsl:value-of select="TITLE"/></a><br/>
</xsl:template>
<xsl:template match="CHAPTER" mode="body">
    <h1 name="{@ID}">{TITLE}</h1><xsl:apply-templates mode="body"/>
</xsl:template>
```

- Création de la règle implicite

```
<xsl:template match="*|/" mode="n"><xsl:apply-templates mode="n"/></xsl:template>
```

Règles Nommées ou *Named Template Rules*

- Les règles nommées peuvent être utilisées par d 'autres règles. Elles peuvent avoir aussi des paramètres
- Exemple (sans paramètre)

```
<xsl:template name="MINI">
   <font face="Times, serif" color="blue" size="2">
   <xsl:value-of select="."/></font>
</xsl:template>

<xsl:template match="YY">
  <xsl:call-template name="MINI"/>
</xsl:template>

<xsl:template match="XX">
    <xsl:call-template name="MINI"/>
</xsl:template>
```

Règles Nommées ou *Named Template Rules*

- Exemple (avec paramètre)

```
<xsl:template name="MINI">
<xsl:param name="police">Times</xsl:param> <!-- valeur par défaut -->
    <font face="{police}, serif" color="blue" size="2">
   <xsl:value-of select="."/></font>
</xsl:template>

<xsl:template match="YY">
 <xsl:call-template name="MINI">
   <xsl:with-param name="file">Helvetica</xsl:with-param>
 <xsl:call-template>
</xsl:template>

<xsl:template match="XX">
<xsl:call-template name="MINI"/>
</xsl:template>
```

Import et inclusion de feuilles de style

- Import
  - peut produire des conflits entre les règles
  - précedence à la dernière règle importée donc (définie)

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:import href="chart.xsl"/>
 <xsl:import href="table.xsl"/>
 <!-- other child elements follow -->
</xsl:stylesheet>
```

- Inclusion
  - remplace les règles (même match) de la feuille par ceux de la feuille incluse

Feuille de style nichée

- La feuille de style XSL est directement nichée dans le document XML

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml" href="#id(mystyle)"?>
 <TABLE>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform" id="mystyle">
  <xsl:template match="/"><html> <xsl:apply-templates/> </html> </xsl:template>
  <xsl:template match="TABLE"><table><xsl:apply-templates/> <table></xsl:template>
  …
  <xsl:template match="xsl:stylesheet"/>
</xsl:stylesheet>
<ROW><COL NAME="Product">Coffee</COL><COL NAME="Price">2.5</COL></ROW>
<ROW><COL NAME="Product">Tea</COL><COL NAME="Price">1.5</COL></ROW>
</TABLE>
```

n'affiche pas la feuille de style et ses fils

Appel de script depuis une feuille XSLT

- Appel de fonctions (externes) par le transformateur
  - Indépendance du langage (JavaScript, Java, TCL, …)
  - Sur XALAN
- Exemple

```
<?xml version="1.0"?>
<doc>
 <foo file="foo.out">
    Testing Redirect extension:
    <bar>A foo subelement text node</bar>
  </foo>
 <main>Everything else</main>
</doc>
```

Appel de script depuis une feuille XSLT

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0"    xmlns:lxslt="http://xml.apache.org/xslt"
                xmlns:redirect="org.apache.xalan.xslt.extensions.Redirect"
                extension-element-prefixes="redirect">
<xsl:template match="/">
 <standard-out>Standard output:<xsl:apply-templates/></standard-out>
</xsl:template>
<xsl:template match="main">
 <main><xsl:apply-templates/></main>
</xsl:template>
<xsl:template match="/doc/foo">
 <redirect:write select="@file">
  <foo-out><xsl:apply-templates/></foo-out>
 </redirect:write>
</xsl:template>
<xsl:template match="bar">
 <foobar-out><xsl:apply-templates/></foobar-out>
</xsl:template>
</xsl:stylesheet>
```

Appel de script depuis une feuille XSLT

- Sortie sur la sortie standard

```
<?xml version="1.0" encoding="UTF-8"?>
<standard-out>
Standard output:
<main>Everything else.</main>
<standard-out>
```

- Sortie sur le fichier « foo.out »

```
<?xml version="1.0" encoding="UTF-8"?>
<foo-out>Testing Redirect extension:
<foobar-out>foo subelement text node</foobar-out>
</foo-out>
```

XSLT et JSTL

- TagLig for XSLT

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jstl/xml" %>
<!-- sets xml variable -->
<c:set var="xml">
  <paragraph>
    This document uses <bold>unusual</bold> markup,  which we want to replace with
      <bold>HTML</bold>.
  </paragraph>
</c:set>
<!-- sets xsl variable -->
<c:set var="xsl">
  <?xml version="1.0"?>
  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:template match="paragraph">
      <p><xsl:apply-templates/></p>
    </xsl:template>
    <xsl:template match="bold">
      <b><xsl:value-of select="."/></b>
    </xsl:template>
  </xsl:stylesheet>
</c:set>
<!-- transform xml with xsl -->
<x:transform xml="${xml}" xslt="${xsl}"/>
```

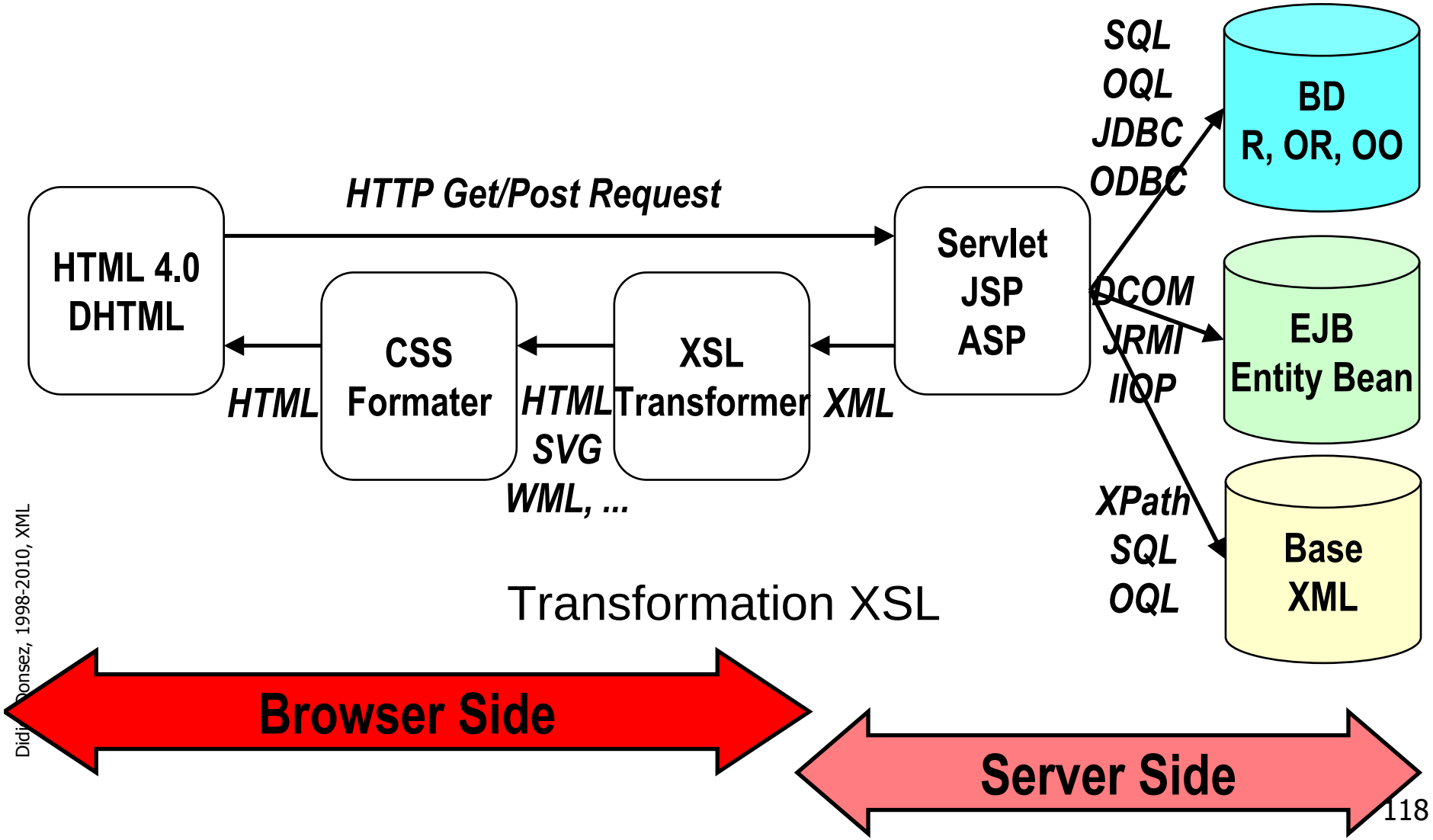
Les transformateurs XSLT

- Sur le client
  - Le navigateur récupère .xml et le .xsl et le transforme à la volée
- Sur le serveur Web (Apache/XALAN)
  - La servlet transforme à la volée un document .xml avec une feuille .xsl : le navigateur reçoit et visualise la transformation.
  - Apache Cocoon
    - Association et application automatique des feuilles XSLT en fonction de l 'extension demandée par le browser **(.svg, .html, .wml)**

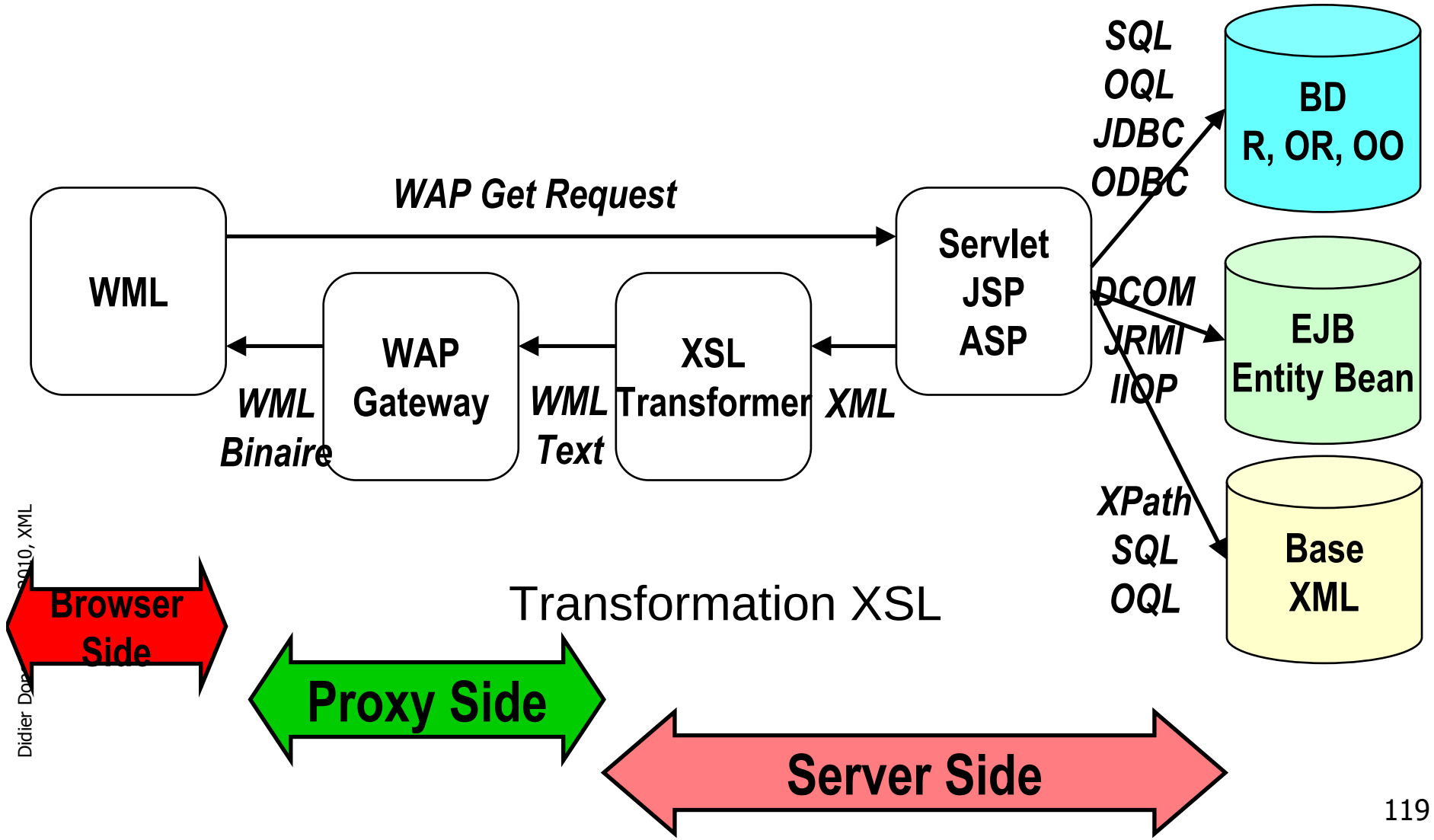
      /pub/book1.xml + book.xml2wml.xsl → /pub/book1.wml

      /pub/book1.xml + book.xml2html.xsl → /pub/book1.html
- Standalone (XT, XALAN)
  - la tranformation est faite par scripting (Shell ou tâche ANT <xslt>)

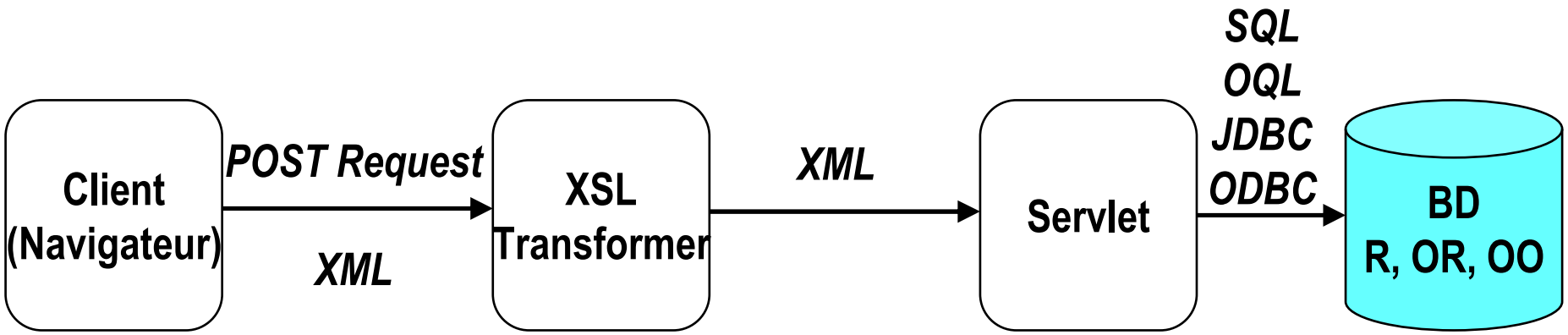
Transformation XSLT et Service Web (i)

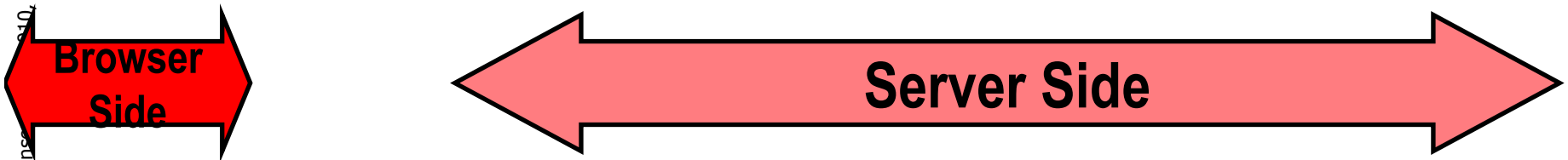


Transformation XSLT et Service Web (ii)



Transformation XSLT et Service Web (iii)





XSL-FO Formatting Object

- Motivation
  - transformation d ’un document XML vers un document non-XML
    - *RTF, Word, PostScript, PDF, LaTex, …*

- Transformateur FO
  - Apache FOP (Formatting Object Processor)

FOP (xml.apache.org)

- Formateur FO
  - Renderer PDF (incluant images et vecteurs SVG)
- Ligne de commande

```
java org.apache.fop.apps.CommandLine order.fo order.pdf
java org.apache.fop.apps.XalanCommandLine order.xml order2fo.xsl order.pdf
    # la feuille de style order2fo.xsl est appliqué sur order.xml avant le formattage
```

- Depuis une application Java, une servlet, une JSP …

```
...
org.apache.fop.apps.Driver driver = new org.apache.fop.apps.Driver();
driver.setRenderer("org.apache.fop.render.pdf.PDFRenderer", version);
driver.addElementMapping("org.apache.fop.fo.StandardElementMapping");
driver.addElementMapping("org.apache.fop.svg.SVGElementMapping");
driver.setWriter(new PrintWriter(new FileWriter(args[1])));
driver.buildFOTree(parser, fileInputSource(args[0]));
driver.format();
driver.render();
...
```

FOP (xml.apache.org) Exemple

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

<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

 <!-- defines page layout -->
 <fo:layout-master-set>

  <!-- layout for the first page -->
  <fo:simple-page-master page-master-name="first"
            page-height="29.7cm"
            page-width="21cm"
            margin-top="1cm"
            margin-bottom="2cm"
            margin-left="2.5cm"
            margin-right="2.5cm">
   <fo:region-before extent="3cm"/>
   <fo:region-body margin-top="3cm"/>
   <fo:region-after extent="1.5cm"/>
  </fo:simple-page-master>

  <!-- layout for the other pages -->
  <fo:simple-page-master page-master-name="rest"
            height="29.7cm"
            width="21cm"
            margin-top="1cm"
```

XML et CSS

- Motivation
  - Visualiser un document XML sans passer par XSLT vers un format visualisable (XHTML1/HTML4)
- Principe
  - Le document utilise un style CSS
    - `<?xml:stylesheet type="text/css" href= "article.css"?>`
  - Les éléments sont rendus visibles par les propriétés `display`
    - `INSTRUMENT { display: inline }`
    - `ARTICLE, HEADLINE, AUTHOR, PARA { display: block }`
  - puis positionnés, formatés, …
    - `HEADLINE { font-size: 1.3em; font-style: italic }`
    - `ARTICLE, HEADLINE, AUTHOR, PARA { margin: 0.5em }`

Voir les recommandations sur les CSS du W3C

Exemple d ’utilisation des CSS pour un document XML

article.css

```
INSTRUMENT { display: inline }
ARTICLE, HEADLINE, AUTHOR, PARA { display: block }
HEADLINE { font-size: 1.3em }
AUTHOR { font-style: italic }
ARTICLE, HEADLINE, AUTHOR, PARA { margin: 0.5em }
```

article.xml

```
<?xml version="1.0"?>
<?xml:stylesheet type="text/css" href= "article.css"?>
<ARTICLE>
  <HEADLINE>Fredrick the Great meets Bach</HEADLINE>
  <AUTHOR>Johann Nikolaus Forkel</AUTHOR>
  <PARA>
    One evening, just as he was getting his
    <INSTRUMENT>flute</INSTRUMENT> ready and
    his musicians were assembled, an officer brought
    him a list of the strangers who had arrived.
  </PARA>
</ARTICLE>
```

API XML

- API de construction de documents XML
  - DOM
- APIs d'analyse (*parsing*) de documents XML
  - SAX
  - DOM

DOM
Document Object Model

- API Objet
  - pour manipuler un document XML depuis un langage de programmation
    - Java, JavaScript, Jscript, ...
- Niveaux de DOM
  - DOM Level 1
    - DOM Core : interfaces de base et leur applications aux documents XML
    - DOM HTML : interfaces applicables aux documents HTML
  - DOM Level 2
    - DOM CSS : feuilles de style
    - DOM Events : Gestion des événements
    - DOM Filters and Itérators : filtrage et traitement itératif
    - DOM Range : isoler des fragments de documents pour traitement

DOM Level 1

- Interfaces de DOM Core (org.w3c.dom)
  - Document, DocumentFragment, Node, CharacterData, Attr, Element, Text, Comment, CDATASection, DocumentType, Notation, Entity, EntityReference, ProcessingInstruction + NodeList, NamedNodeMap
- Interfaces de DOM HTML

Parseurs XML

- Parseur et Parseur/Validateur
  - analyse le document XML
  - vérifie la validité du document
    en fonction de sa DTD ou du XML Schema
- API de Parseurs SAX et DOM
  - SAX (Simple API for XML)
    - orienté événement
      - au cours de l 'analyse, le parseur appelle des fonctions de *callback* quand il rencontre le début/fin du document et des éléments
  - DOM (Document Object Model)
    - orienté arbre d 'objets
      - le parseur construit en mémoire un arbre d 'objets (org.w3c.dom) représentant les éléments
- Langages
  - C, C++, Perl, Java, VB, C#, OCAML, PHP, …

Comparaison SAX et DOM

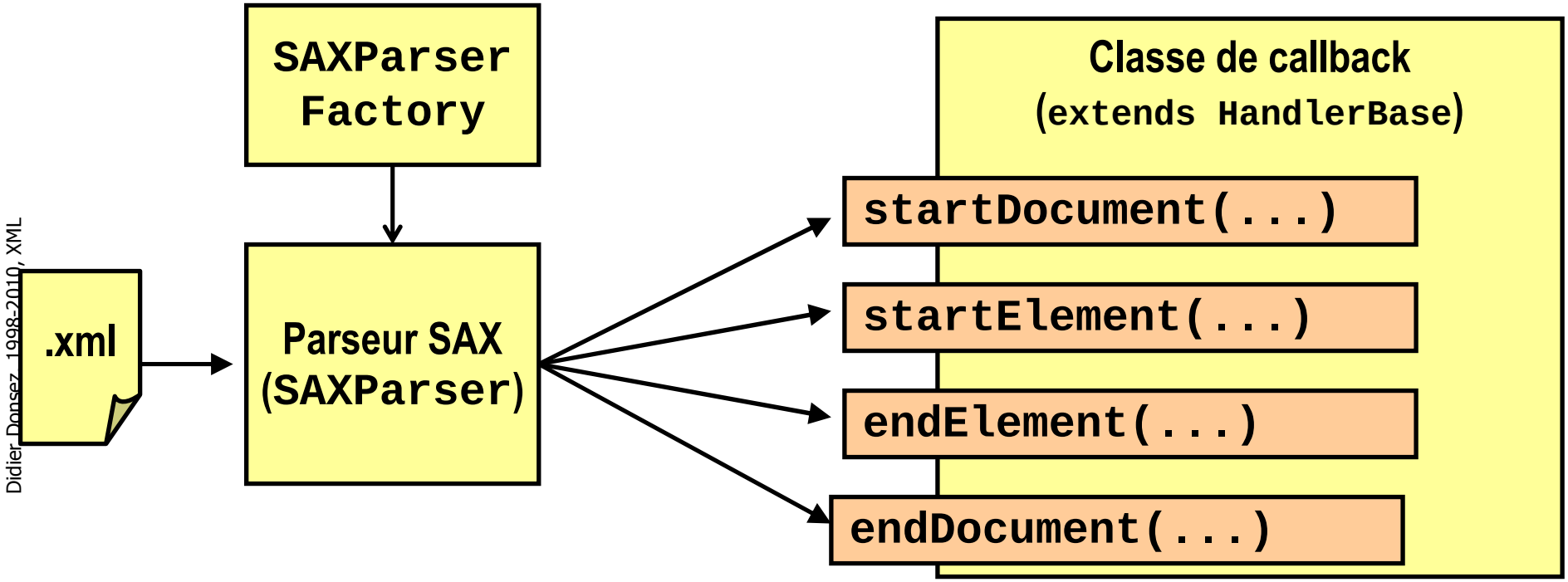
- Avantages SAX
  - le traitement du document par le programme se fait en cours d 'analyse (efficacité)
  - seuls les éléments pertinents sont traités
- Inconvénients SAX
  - écriture plus complexe des fonctions de callback pour traiter des structures imbriquées
- Inconvénients DOM
  - construction d 'un arbre en mémoire
    - multitude de petits objets (efficacité des new et du GC en Java)
    - très grand documents
  - tous les éléments sont représentés
  - le traitement du document par le programme se fait après l 'analyse
- Avantages DOM
  - navigation dans un arbre d 'éléments

L ’API Java pour XML

- 2 API : SAX et DOM
  - Java API for XML Parsing (JAXP)
    - inclut dans J2SE
- Packages
  - org.xml.sax, org.w3c.dom
- Parseurs (et Validateurs)
  - Java J2SE
    - SUN JAXP, IBM xml4j, Apache Xerces, James Clark XP, Oracle 3xml, MicroSoft, ...
  - Java J2ME pour l’embarqué
    - NanoXML, TinyXML, kXML, …
      - Parseur allégé (pas de validation, pas d’entité, …)

L ’API SAX (Simple API for XML)

- Principe
  - Le parseur analyse le flot de caractères du document et appelle des méthodes dites de *callback* lors qu ’il rencontre les balises de début et de fin du document et des éléments



Le parseur et les Handler

- javax.xml.parser.SAXParserFactory
  - Créer un instance de org.xml.sax.Parser
- org.xml.sax.Parser
  - L'interface qui implémente un parseur validant.
- org.xml.sax.HandlerBase
    - Implémente les 4 interfaces suivantes
  - org.xml.sax.DocumentHandler
    - Interface qui définit les méthodes startDocument, endDocument, startElement, endElement.
  - org.xml.sax.ErrorHandler
    - Interface qui définit les méthodes error, fatalError, warning qui sont invoqués en cas d'erreur.
  - org.xml.sax.DTDHandler
    - Interface qui définit les méthodes pour l'analyse de la DTD.
  - org.xml.sax.EntityResolver

Exemple SAX
Implémentation du handler

```
import org.xml.sax.*; import org.xml.sax.helpers.*; import java.io.*;

public class SaxEcho extends HandlerBase {
  public void startDocument( ) throws SAXException {
     System.out.println( "SAX Event: START DOCUMENT" ); }
   public void endDocument( ) throws SAXException {
     System.out.println( "SAX Event: END DOCUMENT" ); }
   public void startElement( String namespaceURI,  String localName,
            String qName, Attributes attr ) throws SAXException {
        System.out.println( "SAX Event: START ELEMENT[ " + localName + " ]" );
        for ( int i = 0; i < attr.getLength(); i++ ){
          System.out.println( " ATTRIBUTE: " + attr.getLocalName(i)
                                        + " VALUE: " + attr.getValue(i) ); }   }
   public void endElement( String namespaceURI, String localName,
            String qName ) throws SAXException {
     System.out.println( "SAX Event: END ELEMENT[ " + localName + " ]" ); }
   ...
```

Exemple SAX
Affichage d 'un document

```
public void characters( char[] ch, int start, int length ) throws SAXException {
    System.out.print( "SAX Event: CHARACTERS[ " );
    try {  OutputStreamWriter outw = new OutputStreamWriter(System.out);
      outw.write( ch, start,length ); outw.flush();
    } catch (Exception e) { e.printStackTrace(); }
    System.out.println( " ]" );   }

public void setDocumentLocator (Locator l) throws SAXException {
    outw.write ("LOCATOR \n SYS ID: " + l.getSystemId() ); out.flush (); }

public void processingInstruction (String target, String data) throws SAXException {
    outw.write("PROCESS: <?"+target+" "+data+"?>"); }
```

Exemple SAX
Affichage d ’un document

```
public static void main (String argv []) {
    if (argv.length != 1) {
        System.err.println ("Usage: cmd filename"); System.exit (1); }
    try {
        SAXParserFactory factory = SAXParserFactory.newInstance()
        // factory.setValidating(true); // set parser validating
        out = new OutputStreamWriter (System.out, "UTF8");
            // Parse the input SAXParser
        saxParser = factory.newSAXParser();
        saxParser.parse( new File(argv [0]), new SaxEcho() );
    } …
```

Exemple SAX
Récupération des exceptions

```
} catch (SAXParseException spe) { // Error generated by the parser
    System.out.println ("\n** Parsing error" + ", line " + spe.getLineNumber ()
            + ", uri " + spe.getSystemId ());
    System.out.println(" " + spe.getMessage() );

} catch (SAXException sxe) { // Error generated by this application
                             // (or a parser-initialization error)
    Exception x = sxe;
    if (sxe.getException() != null) x = sxe.getException().
     x.printStackTrace();

} catch (Throwable t) {
    t.printStackTrace ();
}

System.exit (0);
}
```

L ’API DOM (Document Object Model)

- Principe
  - analyse et valide un document XML et construire un objet DOM
  - l ’objet DOM peut être ensuite traiter par le programme
    - Parcours en largeur et profondeur de l’arbre
    - Manipulation de noeud
      - replacement, ajout de nœud, clonage, changement de valeur
    - Sortie sur un OutputStream, sérialisation, …
- Classes et interfaces
  - DocumentBuilderFactory
  - DocumentBuilder
  - Document
- Remarque
  - L’implantation de Sun utilise simplement le parser SAX avec un DocumentHandler qui construit l’arbre DOM

Construction d'un objet DOM à partir d'une analyse

```
import org.xml.dom.tree.XmlDocument;

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
factory.setNamespaceAware(true);
factory.setValidating(true);
String location = "http://myserver/mycontent.xml";
try {
    DocumentBuilder builder = factory.newDocumentBuilder();
    Document document = builder.parse(location);
    XmlDocument xdoc=(XmlDocument)document;
   // navigation de l 'arbre Document
    // ou xdoc.write(System.out);

} catch (SAXException se) {  … }
```

Construction d'un arbre DOM

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance ();
DocumentBuilder db = dbf.newDocumentBuilder ();
Document doc = db.newDocument ();
```

- Création et ajout d'un élément

```
Element root = doc.createElement ("html");
doc.appendChild (root);
root.appendChild (doc.createElement ("head"));
root.appendChild (doc.createTextNode ("\n some data is text\n  "));
root.appendChild (doc.createElement ("body"));
Text text = doc.createTextNode ("\n");
root.insertBefore (text, root.getFirstChild ());
text = doc.createTextNode ("\n");
root.appendChild (text);
```

- Création et modification d'un attribut

```
Attr attr = doc.createAttribute ("attr1"); attr.setNodeValue ("value of attribute 1");
root.getAttributes().setNamedItem (attr);
root.setAttribute("attr2", "another string");
```

Manipulation d'un objet DOM

- Parcours (récursif) de l'arbre

```
static void trace(Node node, PrintStream out, int indent) {
      for(int i=0; i<indent;i++) out.print(" ");
      out.println(node.getNodeName()+ " ("+ node.getNodeType() + ")");
       if(node.hasChildNodes()) {
         NodeList nl=node.getChildNodes();
         for(int i=0; i<nl.getLength(); i++) trace(nl.item(i),out, indent+1);
} } …
Document doc = db.newDocument (); …
trace(doc, System.out, 0);
```

- Recherche de nœud dans l'arbre

```
NodeList nl=doc.getElementsByTagName("REMARK");
for(int i=0; i<nl.getLength(); i++) trace(nl.item(i),System.out, indent+1);
```

DOM et les langages de scripting

- Exemple de document

```
<listings>
<agent>
  <name>Jill Smith Real Estate</name><phone>123-456-7890</phone>
<agent>
<listing>
  <summary>Beautiful 3-BR penthouse</summary>
  <area>West side</area> <price>$160 000</price>
  <bedrooms>3</bedrooms>
  <bathrooms>2</bathrooms>
  <information>Excellent condition. Very private and facing south with maple hardwood floors and
     vaulted ceilings. 2 full bathrooms, 3  bedrooms (master has walk-in closet and ensuite), ceramic
     tile, gas  fireplace, alarm system, 2 balconies, huge storage + locker, 2  parking.</information>
</listing>
<listing>
  <summary>Priced to sell. Don't wait.</summary>
  <area>East side</area> <price>$170 000</price>
  <bedrooms>1</bedrooms>
  <bathrooms>2</bathrooms>
  <information>1 & den. 917 s.f. for $170 000. How can you go wrong. Large living and dining area, 2
     baths, walk through closet to ensuite, custom marble and Italian pedestal sink in powder room.
     Great buy. Priced to sell.</information>
  </listing>
</listings>
```

DOM et les langages de scripting

- Script ECMA

```
var agent=document.getElementsByTagName("agent")[0];
var agentname=agent.firstChild.firstChild.data;
var agentphone=agent.firstChild.nextSibling.firstChild.data;
var el=document.documentElement.firstChild;
while (el) {
  if (el.nodeType != 1) { el=el.nextSibling; }
  dump(el.tagName + "\n");
  if (el.tagName == "LISTING") {
    var area=el.getElementsByTagName("area")[0].firstChild.data;
    var price=el.getElementsByTagName("price")[0].firstChild.data;
    var br=el.getElementsByTagName("bedrooms")[0].firstChild.data;
    var bath=el.getElementsByTagName("bathrooms")[0].firstChild.data;
    var info=el.getElementsByTagName("information")[0].firstChild.data;
    if (area=="East side") {
      // put the various pieces into the database for the East side listings,
      // including the agent information
    } else if (area=="West side") {
      // put the various pieces into the database for the West side listings,
      // including the agent information
    } // else put in some error message about the area not being found
  }
  el=el.nextSibling;
}
```

XML Binding

- Motivation
  - Faire correspondre des éléments avec des objets dans le programme
    - DOM typé, Mise en correspondance, …
- Java
  - JAXB, Castor, …

- Voir cours XML Binding

XQJ XQuery for Java JSR 225

- independent programmatic access to XQuery implementations.
- Applications using the XQJ API can
  - execute XQuery queries,
  - ‘bind’ data to queries
  - and process query results.

Exemple XQJ

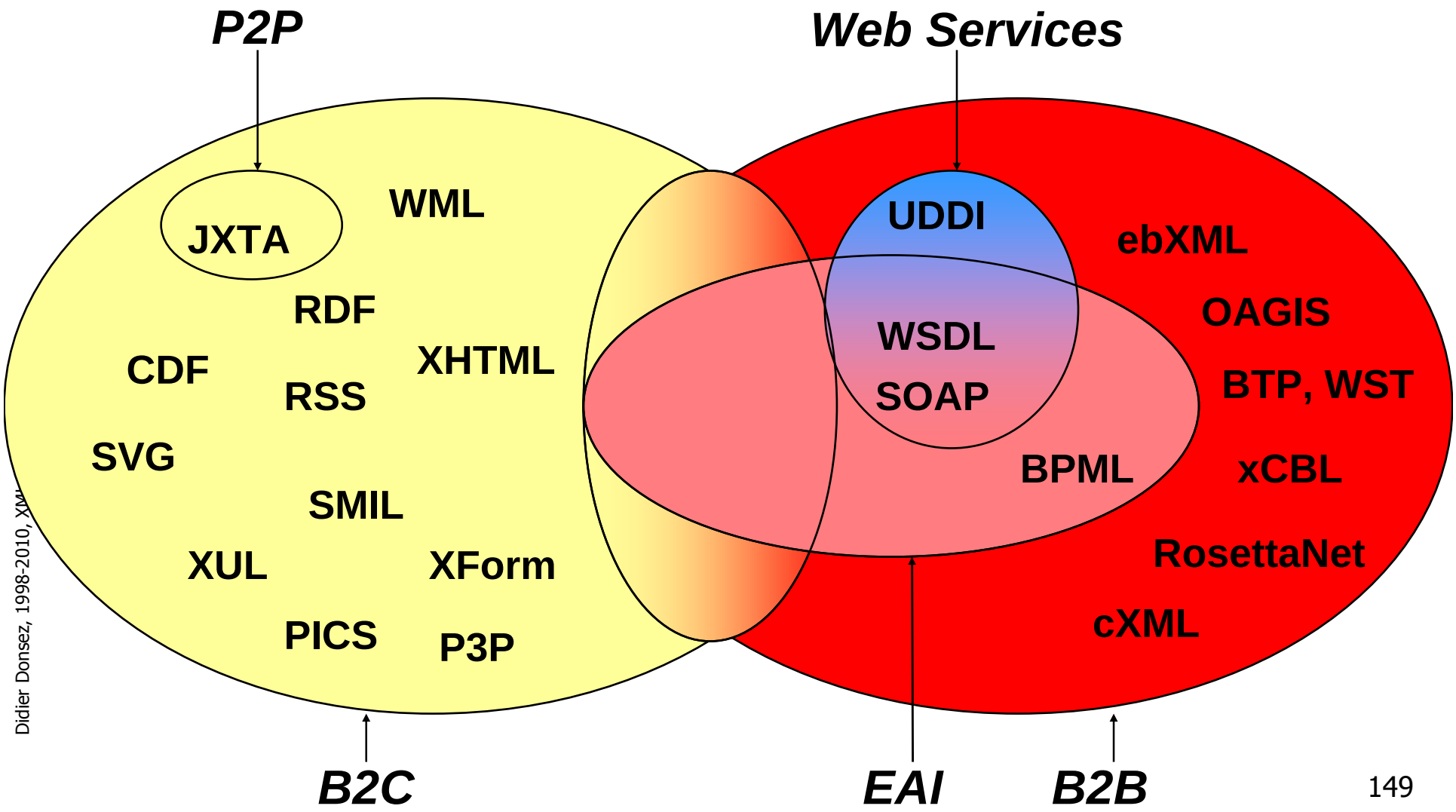
```
...
// establish a connection to the XQuery engine
XQConnection conn = xqds.getConnection();
// create an expression object that is later used
// to execute an XQuery expression
XQExpression expr = conn.createExpression();

// the XQuery expression to be executed
String es = "for $n in fn:doc('catalog.xml')//item " +
  "return fn:data($n/name)";

// execute the XQuery expression
XQResultSequence result = expr.executeQuery(es);

// process the result (sequence) iteratively
while (result.next()) {
    // retrieve the current item of the sequence as a String
    String str = result.getAtomicValue();
    System.out.println("Product name: " + str);
}

// free all resources allocated for the result
result.close();
// free all resources allocated for the expression
expr.close();
// free all resources allocated for the connection
conn.close();
...
```

XQJ

Binding a value to an external variable

```
...
XQExpression expr = conn.createExpression();

// the XQuery expression to be executed
String es = "declare variable $x as xs:integer external;" +
   " for $n in fn:doc('catalog.xml')//item" +
   " where $n/price <= $x" +
   " return fn:data($n/name)";
// bind a value (21) to an external variable with the QName x

QName var1 = new QName("x");
expr.bindInt(var1, 21, null);

// execute the XQuery expression
XQResultSequence result = expr.executeQuery(es);

// process the result (sequence) iteratively
```

Quelles Applications XML



RDF Ressource Definition Framework

- Objectifs: associer des propriétés descriptives aux ressources (uri) du Web
- Exemple

- Schémas RDF
  - PICS, DC (Dublin Core), Sitemap, P3P

RDF Ressource Definition Framework

- Objectifs :
  - décrire plus précisément les ressources disponibles sur le web,
  - adaptation au contexte XML,
  - support pour ontologies, thesaurus...
- Principe
  - Un document est décrit à l'aide d'une **RDF description**
  - utilisant des éléments contenus dans un ou plusieurs **RDF schema**
- Limitations
  - Manque de stabilité
    - normative (en complément de l'évolution des autres normes)
    - au niveau des outils,
    - au niveau des usages
- Bibliographie
  - site du W3C http://www.w3.org/RDF/
  - An Idiot's Guide to the Resource Description Framework (Renato Iannella) http://archive.dstc.edu.au/RDU/reports/RDF-Idiot/

RDF Ressource Definition Framework

- Exemple

```
<? xml version="1.0" ?>
<RDF            xmlns = "http://w3.org/TR/1999/PR-rdf-syntax-19990105#"
                xmlns:DC = "http://purl.org/DC#" >
   <Description about = "http://dstc.com.au/report.html" >
     <DC:Title> The Future of Metadata </DC:Title>
     <DC:Creator> Jacky Crystal </DC:Creator>
     <DC:Date> 1998-01-01 </DC:Date>
     <DC:Subject> Metadata, RDF, Dublin Core </DC:Subject>
   </Description>
</RDF>
```

- Explication
  - L'objet : **http://dstc.com.au/report.html**
  - est décrit dans l'élément XML/RDF <Description about= ...> ... </Description>
  - par les propriétés : **Title, Creator, Date, Subject**
  - définies dans le domaine nominal : xmlns:DC="http://purl.org/DC#"

RDF Schema

- Objectif : description des propriétés RDF
- Exemple

```
<?xml version='1.0'?>
<rdf:RDF        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
                xmlns:dc="">
<rdf:Description about = "">
<dc:Title> The Dublin Core Element Set </dc:Title>
<dc:Date> 1995-03-01 </dc:Date> …
</rdf:Description>
<rdf:Description ID="Title">
    <rdf:type rdf:resource ="http://www.w3.org/TR/PR-rdf-syntax-ns#Property"/>
    <rdfs:label>Title</rdfs:label>
    <rdfs:comment>The name given to the resource.</rdfs:comment>
    <rdfs:isDefinedBy rdf:resource = ""/>
</rdf:Description> ...
</rdf:RDF>
```

RDF Schema

- Explications
  - xmlns:rdfs introduit l'espace de noms servant à décrire le schéma RDF.
  - Ce document commence par sa description en RDF
    Puis il donne les définitions relatives aux 15 éléments (définis comme propriétés - Property - RDF

DCD Document Content Description

- Description XML d ’une DTD
  - http://www.w3.org/TR/NOTE-dcd
- Exemple

XHTML

- Appliquer l ’extensibilité de XML à HTML4.01
  - ajout de nouveaux éléments, ...
- 3 étapes

```
<!DOCTYPE html  PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                "DTD/xhtml1-strict.dtd">
<!DOCTYPE html   PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                "DTD/xhtml1-transitional.dtd">
<!DOCTYPE html   PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
                "DTD/xhtml1-frameset.dtd">
```

- Normalisateur XHTML (depuis HTML)
  - http://www.w3.org/People/Raggett/tidy, http://tidy.sourceforge.net/

XHTML
Les différences avec HTML4

- Document correct
  - pas de chévauchement

    ** les élémentss 'emboitent **

    au lieu de

    ** les balisesse chevauchent pas **
  - éléments vides

    **
<hr/>** au lieu de **
<hr>** qui est INCORRECT
  - valeurs attributs

    <tab col="3"> au lieu de **<tab col=3>** qui est INCORRECT
- Style et Script

```
<script>
 <![CDATA[
 ... unescaped script content ...
 ]]>
 </script>
```

XHTML - Exemple

- Document XML utilisant des éléments du domaine de noms XHTML

```
<?xml version="1.0" encoding="UTF-8"?>
        <!-- initially, the default namespace is "books" -->
        <book xmlns='urn:loc.gov:books'
          xmlns:isbn='urn:ISBN:0-395-36341-6' xml:lang="en" lang="en">
         <title>Cheaper by the Dozen</title>
         <isbn:number>1568491379</isbn:number>
         <notes>
          <!-- make HTML the default namespace for a hypertext commentary -->
          <p xmlns='http://www.w3.org/1999/xhtml'>
             This is also available <a href="http://www.w3.org/">online</a>.
          </p>
         </notes>
        </book>
```

XHTML - Exemple

- Document XHTML utilisant la recommandation MathML

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head><title>A Math Example</title></head>
<body>
        <p>The following is MathML markup:</p>
        <math xmlns="http://www.w3.org/1998/Math/MathML">
         <apply> <log/>
          <logbase>
           <cn> 3 </cn>
          </logbase>
          <ci> x </ci>
         </apply>
        </math>
</body></html>
```

XUL *(XML based User Interface Language)* *http://www.xulplanet.com*

- Widget des GUI (Interface graphique)
  - Supporté par NS Communicator 6 et Mozilla
- Exemple

XForm (W3C)

- Définition abstraite des formulaires
  - Limitation du <FORM> HTML
  - Ne définit l'apparence GUI
  - Functionnalités : Dépendances entre les entrées, …
- Exemple

Les Outils

- Editeurs XML
  - http://www.alphaWorks.ibm.com/tech/visualxmltools , ...
- Browsers XML
  - W3C Amaya, MS IE5, Mozilla, Opera
- Parseurs XML/DOM
  - MS, Java API for XML Parsing (JAXP).
  - Parsers XML pour l'embarqué (NanoXML, TinyXML, kXML)
- Compilateurs XML/DOM
  - Enhydra
- Transformateurs XSL
  - MS, XT, Xalan (xml.apache.org)
- Serveurs Web
  - http://technet.oracle.com/tech/xml/xsql_servlet/
    - servlet qui interroge une BD en SQL et formate le résultat en XML
- SGBD XML
  - Oracle 8i, Ardent, Software AG TAMINO, ObjectStore, ...

Apache XML Project *(http://xml.apache.com)*

- Xerces - XML parsers in Java, C++ (with Perl and COM bindings)
- Xalan - XSLT stylesheet processors, in Java and C++
- Cocoon - XML-based web publishing, in Java
- FOP - XSL formatting objects, in Java
- Xang - Rapid development of dynamic server pages, in Java
- SOAP - Simple Object Access Protocol

XSP

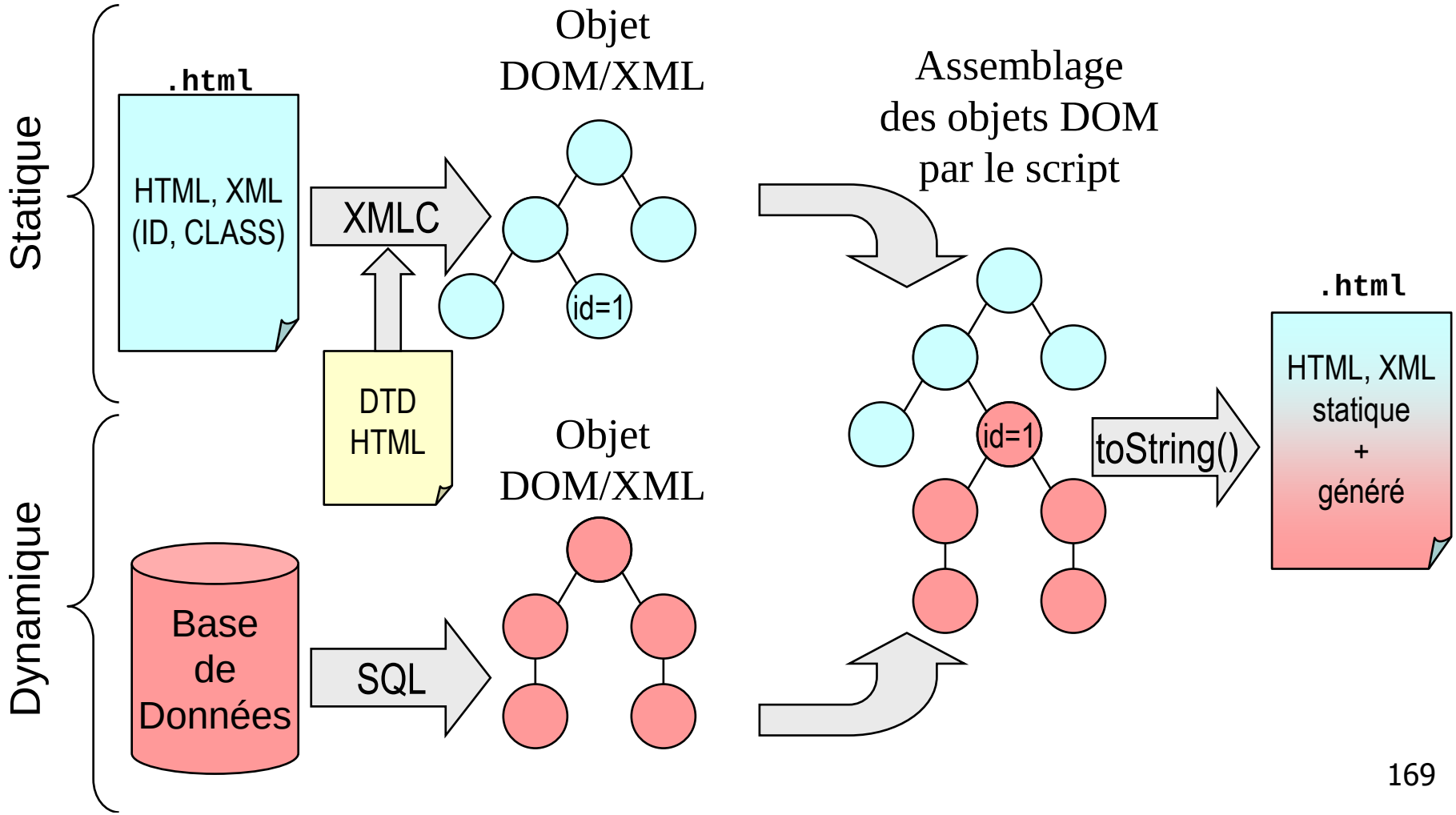
- Voir xml.apache.org

Scripting Serveur et et XML/DOM

- Inconvénients des solutions précédentes
  - Script générant du HTML
    - génération par programmation (« à la main »)
  - Script niché dans un document HTML
    - Utilisation d ’outils auteur pour créer les pages, OUI !
    - MAIS l ’insertion de scripts empêche de retravailler les pages
- Alternative
  - Utilisation du modèle DOM dans les scripts
    - www.enhydra.org

Scripting Serveur et et XML/DOM
Principe (i)

- Principe : le script ne manipule que des objets DOM



Scripting Serveur et et XML/DOM

Cycle de Développement (i)

- 1) Conception de document HTML avec des outils auteur
  - L'infographiste concoit le document avec une touche graphique et des animations standards (bannière, boutons animés, menu, …)
  - Certaines zones du document seront générés à la volée par un script. Ces zones sont repérées par des attributs ID ou CLASS.
  - Le document pourra être modifié ultérieurement (après 4)
- 2) Compilation des pages HTML en objet Java DOM
  - le document HTML est compilé en une classe Java instanciant des objets DOM/XML
    - nécessite parfois une mise en conformité (DTD/HTML4) du source HTML produit par les outils auteur
    - packages org.w3c.dom et org.w3c.dom.html

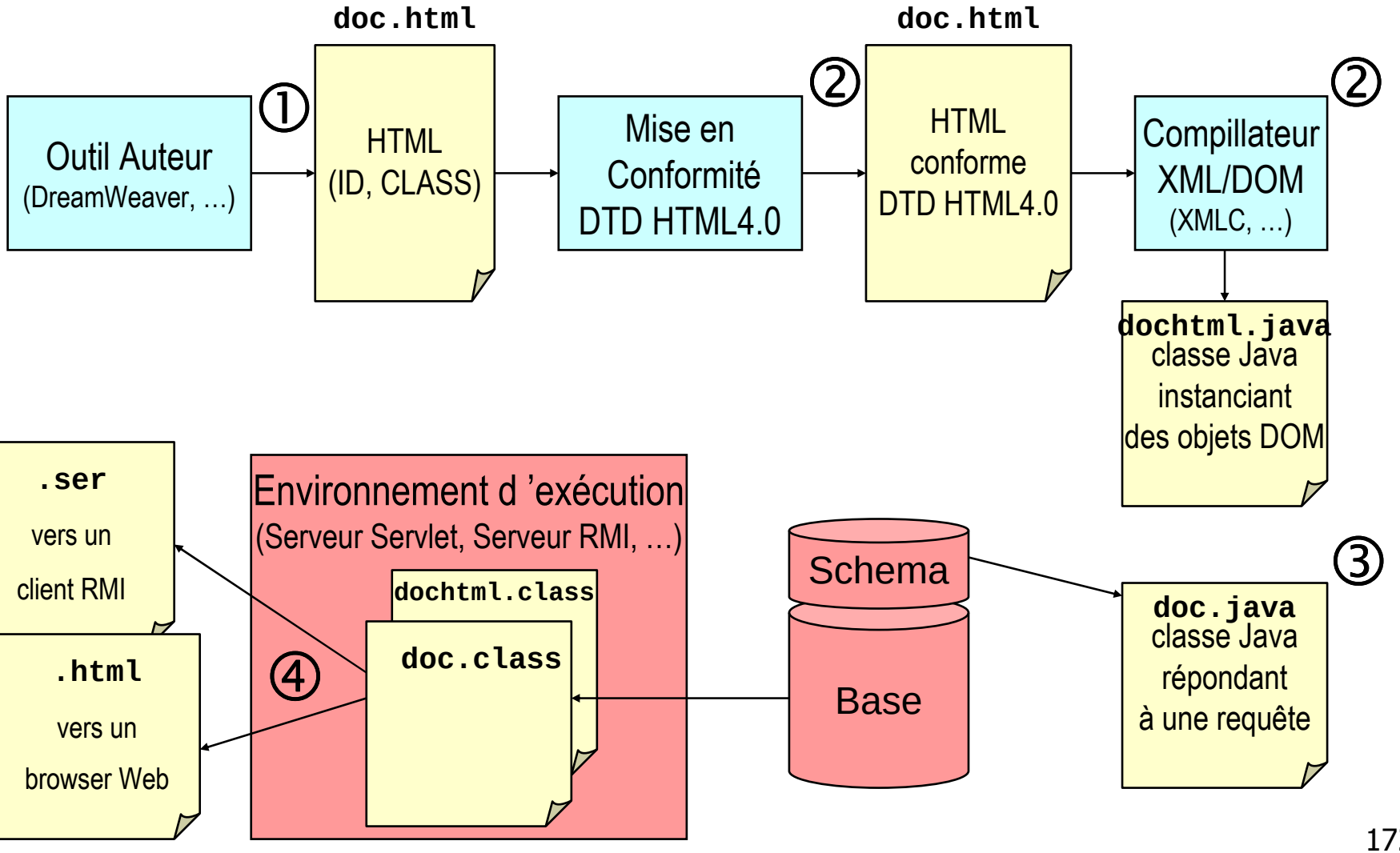
Scripting Serveur et et XML/DOM

Cycle de Développement (ii)

- 3) Développement du script
  - Le script (servlet) instance un objet DOM à partir de la classe produit par le compilateur DOM. Cet objet est un arbre
  - Les sous-arbres identifiés (ID ou CLASS) par l 'infographiste correspondent aux zones qui vont être générés à la volée.
  - Le script modifie ou remplace ces sous-arbres à partir de de sous-arbres générés à partir de données retirées du Système d 'Information (à l 'exécution).
- 4) Requête
  - Le script assemble les sous-arbre DOM générés avec l 'arbre DOM statique.
  - Le script répond à la requête du client en retournant
    - l arbre DOM sérialisé dans le cas d 'une réponse RMI
      - méthode writeObject()
    - une génération texte de l 'arbre DOM dans le cas d 'une réponse HTTP
      - méthode toString()

Scripting Serveur et et XML/DOM

Cycle de Développement (iii)



Binarisateur XML

- Représentation binaire (bytecode) de documents XML
  - http://www.w3.org/XML/Binary/
- Motivation
  - Version plus compacte de documents XML
  - pour les transferts et l'empreinte mémoire (informatique embarquée)
- Utilisations
  - WML, MPEG7 (iTV), …

Parser de DTD

- http://www.wutka.com/dtdparser.html

Bibliographie

- Voir
  - Cours « BD et XML », « Scripting Serveur », « HTML »
- Sites
  - http://www.w3c.org/XML
  - http://java.sun.com/xml/ Java API for XML Parsing 1.0 (JAXP).
  - http://xml.apache.org
  - http://www.xml.com
  - http://www.jclark.com/xml/xt.html
  - http://www.oasis-open.org/cover/xml.html
  - http://www.xml101.com
  - http://www.alphaworks.ibm.com
  - http://www.xmlsoftware.com
  - news://comp.text.xml

Bibliographie

- Charles F. Goldfarb, "The XML Handbook", 2nd edition, 1999, Ed Prentice Hall, ISBN 0-13-014714-1
- Elliotte Rusty Harold , « XML Bible », 1015 pages July 1999, IDG Books Worldwide; ISBN: 0764532367
  - la bible : il existe la version française (ISBN 2-7464-0070-7)
- Brett McLaughlin "Java and XML", 1st Edition June 2000, ISBN 0-596-00016-2, 498 pages, Edition OREILLY &ASSOCIATES
- J.C. Bernadac, F. Knab, « XML, Construire une Application », Ed. Eyrolles, 1999, ISBN 2-212-09081-1
  - assez complet et 2 études de cas complètes en environnement ASP et Servlet
- Benoît Marchal, XML by Example - 425 pages 1 edition, December 1999, Que Education & Training; ISBN: 0789722429
- M. Floyd, « Building Web Sites with XML », Ed Prentice Hall 1999, ISBN 0-13-086601-6

Bibliographie

- Michael Kay, "XSLT programmer's Reference", 2000, Ed Wrox Press, ISBN
- Ian Graham, "XHTML 1.0 language and design sourcebook", 2000, Ed Wiley
- Andrew Patzer , "Programmation Java côté serveur : Servlets, JSP et EJB", Ed Eyrolles-Wrox, 2000, ISBN 1-861002-77-7 (sources des exemples sur www.wroxfrance.com)
  - chapitre 14 dans le contexte serveur Web
- Elliotte Rusty Harold, W. Scott Means, « XML in a Nutshell, 2nd Edition, A Desktop Quick Reference », Ed Oreilly
- Howard Katz, Don Chamberlin, Denise Draper, Mary Fernandez, Michael Kay, Jonathan Robie, Michael Rys, Jerome Simeon, Jim Tivy, Philip Wadler, « XQuery from the Experts: A Guide to the W3C XML Query Language », Ed Addison-Wesley, Aout 2003, ISBN 0321180607, 512 pp