XML

Introduction et concepts

Référence : XML, Alexandre Brillant

Introduction

- Aujourd'hui
  - Production importante de documents (sur le Web)
  - Plusieurs applications, se partagent des documents, recherchent des documents sur le Web
  - Les formats de ces documents sont disparates
    - PDF, images, HTML, etc.
- On cherche un protocole commun pour
  - Les manipuler, les compléter, les communiquer
    - à d'autres personnes
    - entre différents outils
    - les échanger sur le Web
  - Les interroger

Introduction

- Les documents à balises offrent ces possibilités
  - Historiquement, on trouve :
  - SGML
    - Standard Generalized Markup Language
      - Meta-langage pour définir des langages de "markup"
  - HTML
    - Hypertext Markup Language
      - Application de SGML au Web, ayant peu de tag
  - XML
    - eXtensible Markup Language
      - Version plus légère que SGML pour le Web

Introduction

Rôle du document XML dans les entreprises

- L'entreprise fournit des services dont la production nécessite plusieurs étapes
- A chaque étape, des informations peuvent être produites et/ou consommées
- Le rôle de XML
  - Donner un cadre de stockage et de traitement de l'ensemble de ces informations
  - Il offre un formalisme pour représenter de manière uniforme cette information
    - En choisissant les mots, leur ordre, leur signification…
  - En lui donnant du sens dans un cadre précis

Introduction

- Le document XML : orienté document ou données ?
  - Lorsque les données sont produites par un humain :
    - On dit qu'elles sont orientées document
      - Un fichier orienté document pourrait représenter par exemple, un livre, un blog, un message
  - Lorsque les données sont construites automatiquement par programme :
    - On dit qu'elles sont orientées données
      - Un fichier orienté données pourrait représenter par exemple, le résultat d'extraction d'informations d'une base de données

Introduction

- Structure et validation d'un document XML
  - On associe à un document XML un schéma, qui peut être vu comme le schéma d'une base de données relationnelle
  - La validation d'un document XML garantit que la structure respecte bien ce schéma
    - Les documents en circulation doivent être en accord avec ce schéma pour être acceptés par la plateforme
    - Ces situations de vérification sont très utiles pour éviter des régressions logicielles lors d'évolutions de plateformes

Introduction

- Transformation et adaptation d'un document XML
  - Un document XML peut être transformé
    - Il n'est pas figé
    - Le format XSLT (eXtensible Stylesheet Language Transformation) est un moyen pour adapter un document XML à un autre format XML
    - Avec XSLT, on peut produire
      - XHTML, XSL-FO (avec indirectement du PDF, RTF, …)

Introduction

- Circulation des documents XML et workflows
  - Les flux de données (workflows) existants vont être petit à petit remplacés par des workflows XML
  - Les fichiers XML vont
    - circuler,
    - s'enrichir au fur et à mesure de ces déplacements,
    - être contrôlés,
    - puis être présentés aux différents acteurs de l'activité : commerciaux, clients…

Introduction

XML et les bases de données

- Comme il structure les données selon un schéma fixé, peut-il remplacer les BD Relationnelles ?
- Sûrement pas !
  - Un document XML est un fichier texte
  - Il n'est optimisé ni en espace ni pour les manipulations qu'on fait sur les BD
  - Il est vu comme une partie volatile d'un système d'information
  - Il résout un problème de circulation de l'information à un moment donné
- Toutefois, XML peut apporter des solutions complémentaires
  - Des BD XML existent avec des langages d'interrogation puissants qui permettent de travailler du côté client

Introduction

- Les bases « natives » XML
  - On considère deux formes
    - Celles gardant le texte XML tel quel
      - eXist (Open Source) : on peut l'administrer comme MySql, l'utiliser directement par XQuery ou par programme avec JQuery
    - Celles effectuant une conversion sous une forme objet
      - Comme DOM qui est une standardisation objet d'un document XML
      - On peut l'utiliser de manière standard à travers JS, PHP, Java…
  - Adresses de bases natives Open Source et propriétaires
    - http://www.rpbourret.com/XML/XMLdatabaseProds.htm

Introduction

Les parseurs XML

- Ou analyseurs syntaxiques
  - sont des outils logiciels permettant de parcourir un document et d'en extraire des informations
- On distingue deux types de parseurs XML :
  - *les parseurs validants* permettant de vérifier qu'un document XML est conforme à sa DTD
  - *les parseurs non validants* se contentant de vérifier que le document XML est bien formé (c'est-à-dire respectant la syntaxe XML de base)

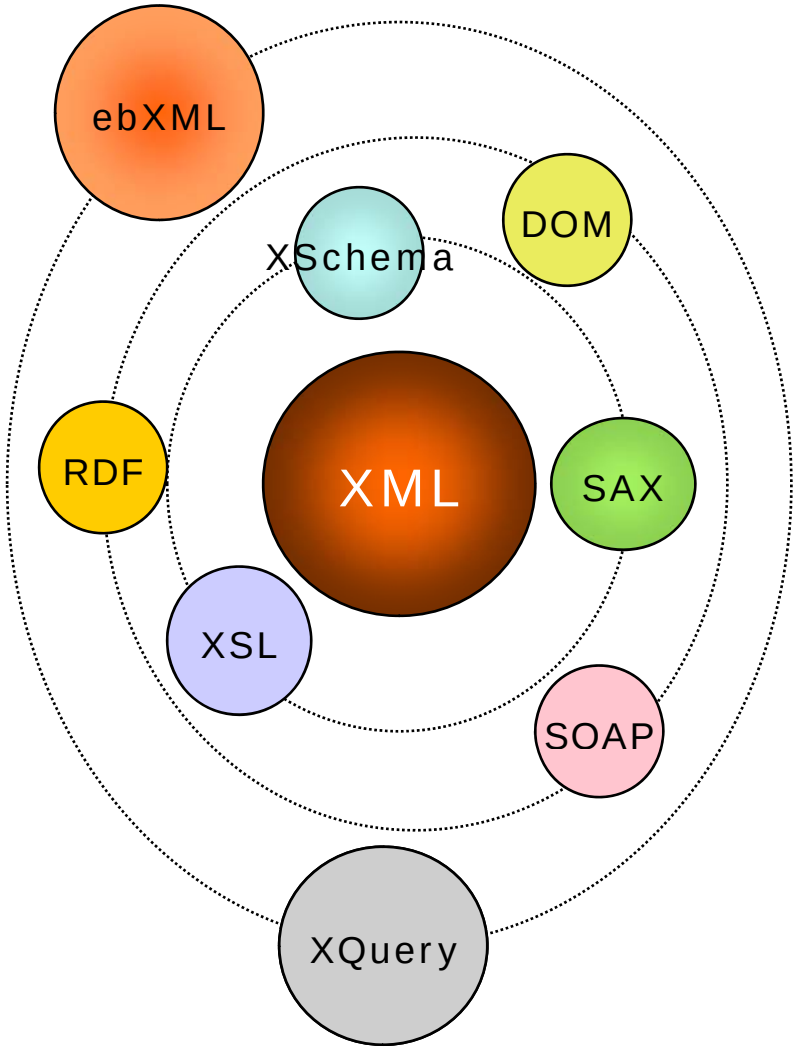
Introduction

Les parseurs XML

- Les analyseurs XML sont également divisés selon l'approche qu'ils utilisent pour traiter le document
- On distingue actuellement deux types d'approches :
  - Les API utilisant une approche **hiérarchique** :
    - Type DOM, construisent une structure hiérarchique contenant des objets représentant les éléments du document, et dont les méthodes permettent d'accéder aux propriétés
  - Les API basés sur un mode **événementiel**
    - Type SAX, permettent de réagir à des événements (comme le début d'un élément, la fin d'un élément) et de renvoyer le résultat à l'application utilisant cette API

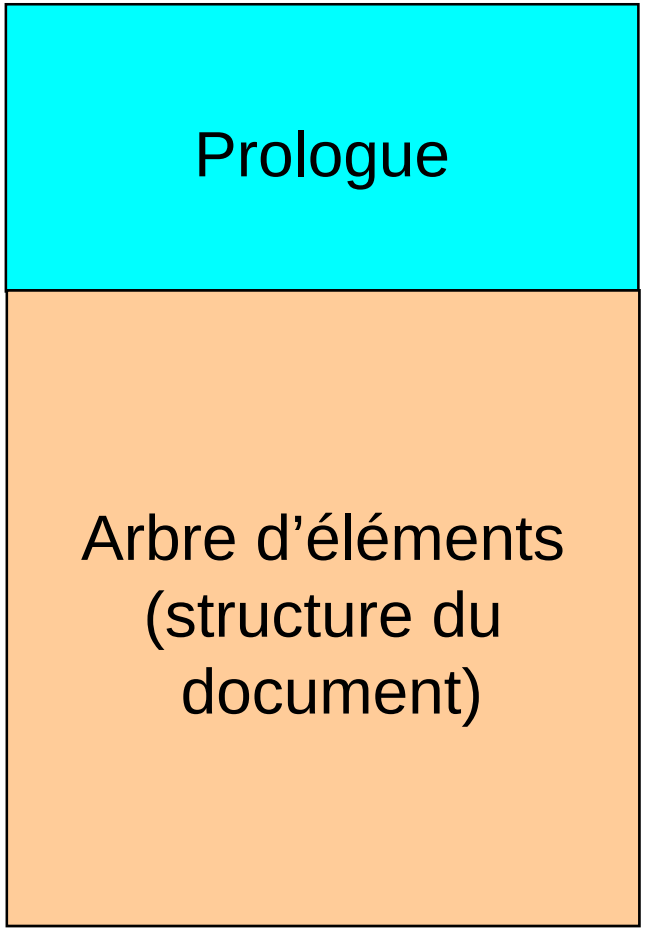
La galaxie XML :
les standards de base qui s'appuient sur XML

- Xpath
  - langage de cheminement dans les arbres XML
- XSL
  - pour la génération de feuilles de style
- XQuery
  - pour les bases de données,
- DOM et SAX
  - pour la programmation
- SOAP
  - pour les services distribués



Structure d'un document XML

- Un document XML
  - a la structure suivante
    - Prologue
    - Élément racine
    - Arbre d'éléments ou structure
- Le prologue
  - peut contenir :
    - une déclaration XML
    - des instructions de traitement
    - une DTD



Structure d'un document XML

- Exemple

```
<!-- Prologue -->
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!-- Élément racine -->
<biblio>
<!-- Premier enfant -->
  <livre>
            <!-- Élément enfant titre -->
            <titre>Les Misérables</titre>
            <auteur>Victor Hugo</auteur>
            <nb_tomes>3</nb_tomes>
  </livre>
  <livre>
            <titre>L'Assomoir</titre>
            <auteur>Emile Zola</auteur>
  </livre>
  <livre lang="en">
            <titre>David Copperfield</titre>
            <auteur>Charles Dickens</auteur>
  </livre>
</biblio>
```

Document XML

Le prologue

La déclaration XML

```
<?xml version="1.0" encoding='iso-8859-1' standalone='yes'?>
```

- Indique au processeur qui va traiter le document :
  - Version du langage : 1.0 (la plus récente)
  - Encoding :
    - jeu de codage de caractères utilisé dans le document. Le jeu de caractères standard pour la France est le ISO-8859-1
    - Par défaut, l'attribut encoding a la valeur UTF-8 (Unicode)
  - Standalone
    - Si Standalone = 'yes', le processeur considère que toutes les déclarations nécessaires au traitement du document sont incluses dans le document courant (le document est autonome et ne requiert aucune autre donnée externe)

Document XML

L'arbre d'éléments

- Tout document XML est représenté sous la forme d'un arbre d'éléments
- Comme tout arbre, il comporte :
  - une racine, des branches et des feuilles
- qui pour nous seront les éléments

```
<liste-cd>
        <cd>
            <interprete>Keith Jarret</interprete>
            <titre>The Köln Concert</titre>
            <specif type="Live" nb_cd="1"/>
        </cd>
        <cd>
            <interprete>Keith Jarret</interprete>
            <titre>La Scala</titre>
            <specif type="Live" nb_cd="1"/>
        </cd>
</liste-cd>
```

Document XML

Les attributs

- Tous les éléments peuvent contenir un ou plusieurs attributs
- Un attribut est composé d'un nom et d'une valeur
- Syntaxe :

  <Nom-elem attribut$_1$, attribut$_2$, …>
- Syntaxe d'un attribut

  attribut$_i$: nom="valeur"
- Exemple :
  - <instrument type="vent">trompette</instrument>

XML : DTD

1. Validité des documents
2. DTD

Validité des documents

- Pour être utilisable par les différentes applications, un document XML doit être
  - bien formé (well formed document)
    - Balises correctement imbriquées
    - Parsable et manipulable
    - Pas nécessairement valide par rapport à la DTD
  - valide (valid document)
    - Bien formé +
    - Conforme à la DTD (ou au schéma)
  - Logiciels de validation
    - Cooktop, XmlSpy (voir plus loin)
    - Validation en ligne : http://www.xmlvalidation.com/

La DTD

- a deux représentations physiques
  - peut faire partie du document XML
    - elle est alors dite *interne*
  - être un fichier à elle seule, lui permettant d'être utilisable par d'autres documents XML
    - elle sera alors dite *externe*
- Les DTD externes peuvent être séparées en deux catégories :
  - privée (SYSTEM) et publique (PUBLIC)
    - La première catégorie est représentée par un fichier accessible uniquement en local
    - La seconde sera disponible pour tout le monde via une *URI* (Uniform Resource Identifier)

DTD

- DTD interne
  - Déclaration
    - <!DOCTYPE elt-racine déclarations>
  - Exemple

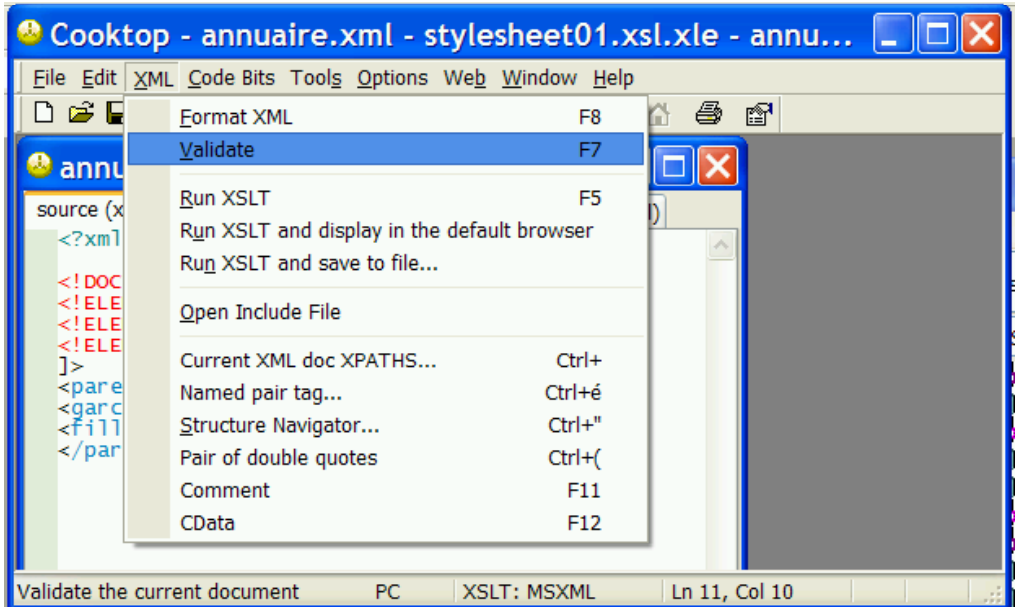
```
<?xml version="1.0" standalone="yes"?>

<!DOCTYPE parent [
<!ELEMENT parent (garcon,fille)>
<!ELEMENT garcon (#PCDATA)>
<!ELEMENT fille (#PCDATA)>
]>
<parent>
   <garcon>Loic</garcon>
   <fille>Marine</fille>
</parent>
```

DTD

Validation

- Appeler Cooktop ou XmlSpy



- Ou le faire en ligne :
  - http://www.xmlvalidation.com/

DTD

DTD externe de type SYSTEM :

- Le fichier parent.xml

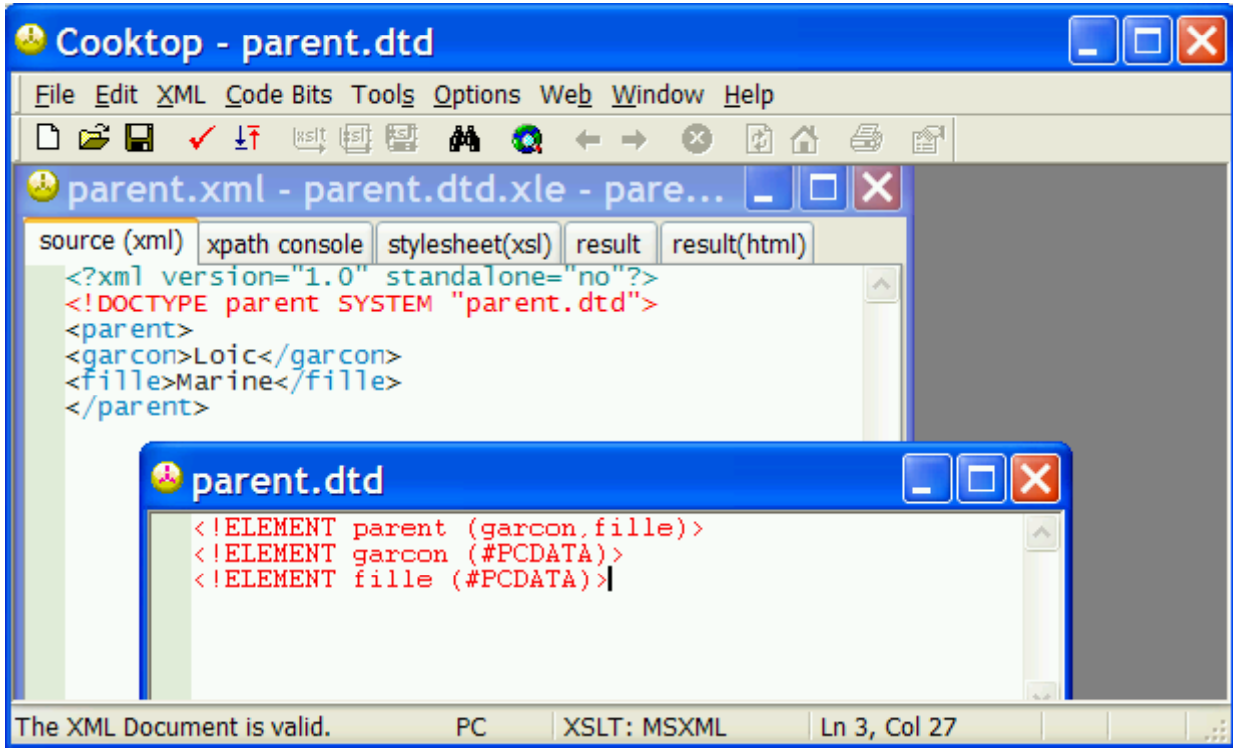
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE parent SYSTEM "parent.dtd">
<parent>
   <garcon>Loic</garcon>
   <fille>Marine</fille>
</parent>
```

- Le fichier parent.dtd contient :

```
<!ELEMENT parent (garcon,fille)>
<!ELEMENT garcon (#PCDATA)>
<!ELEMENT fille (#PCDATA)>
```

DTD

- Validation sous Cooktop



DTD

- DTD externe de type PUBLIC :
  - Exemple : référence à la DTD XHTML
    - <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
  - La chaîne de caractères après le mot PUBLIC fait référence :
    - tout d'abord à l'identifiant de la DTD
      - (ici -, qui signifie que la DTD n'a pas de numéro d'enregistrement officiel)
    - au propriétaire de la DTD
      - (ici le W3C)
    - puis son nom
    - enfin sa langue

DTD

Déclaration d'élément

Définition

- Chaque élément du document doit être défini par une commande du type

  <!ELEMENT nom (contenu) >

- où nom est le nom de l'élément (balise) et
- où contenu décrit :
  - soit la structure de l'élément s'il est composé
  - soit #PCDATA si c'est une feuille

Exemple

<!ELEMENT livre (auteur, éditeur)>

- définit un élément livre composé d'une séquence d'éléments auteur et éditeur

DTD

Forme du contenu

- Notations
  - (a, b) séquence
  - (a|b) liste de choix
  - a? élément optionnel [0,1]
  - a* élément répétitif [0,N]
  - a+ élément répétitif [1,N]
- Exemples
  - (nom, prenom, rue, ville)
  - (oui|non)
  - (nom, prenom?, rue, ville)
  - (produit*, client)
  - (produit*, vendeur+)

DTD

Exemple

- annuaire.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE annuaire SYSTEM "annuaire.dtd">
<annuaire>
  <personne type="étudiant">
    <nom>HEUTE</nom>
    <prenom>Thomas</prenom>
    <email>webmaster@xmlfacile.com</email>
  </personne>
  <personne type="chanteur">
    <nom>CANTAT</nom>
    <prenom>Bertrand</prenom>
    <email>noir@desir.fr</email>
  </personne>
</annuaire>
```

- annuaire.dtd

```
<!ELEMENT annuaire (personne*)>
<!ELEMENT personne (nom,prenom,email+)>
<!ATTLIST personne type (étudiant | professeur | chanteur | musicien) "étudiant">
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT email (#PCDATA)>
```

DTD

Élément vide

- Un élément vide est un élément qui n'a aucun contenu
- Déclaration :

  <!ELEMENT elem-vide EMPTY>
- Exemples :
  - img, hr, br dans HTML
- Un élément vide peut avoir des attributs
  - Déclaration dans le Body :

```
<img width="100" height="30" src="paysage.jpg" alt="Paysage de
  neige"/>
```

DTD

Élément vide : Exemple

```
<!ELEMENT contacts (personne+)>
<!ELEMENT personne (nom, prénom,
dateDeNaissance?, adresse, email+,
téléphone*)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prénom (#PCDATA)>
<!ELEMENT dateDeNaissance (#PCDATA)>
<!ELEMENT adresse EMPTY>
<!ELEMENT email (#PCDATA)>
<!ELEMENT téléphone (#PCDATA)>
<!ATTLIST adresse pays CDATA
#REQUIRED>
```

```
<contacts>
    <personne>
        <nom>Chazalon</nom>
        <prénom>Grégory</prénom>
        <dateDeNaissance>1977-07-09
        </dateDeNaissance>
        <adresse pays="France"/>
        <email>gchazalon@voila.fr</email>
        <email>gchazalon@hotmail.com</email>
        <téléphone>01 23 45 67 89</téléphone>
    </personne>
    <personne>
        <nom>Lemoine</nom>
        <prénom>Joséphine</prénom>
        <dateDeNaissance>1977-01-06
        </dateDeNaissance>
        <adresse pays="France"/>
        <email>joe6@voila.fr</email>
    </personne>
</contacts>
```

DTD

- Élément à contenu mixte
  - Contient du texte et des éléments
- Exemple de déclaration :

```
<!ELEMENT bonjour (#PCDATA | cible)*>
```

- Exemple d'utilisation :

```
<bonjour>
  Hello
  <cible>World</cible>
</bonjour>
```

DTD : attribut

Définition

<!ATTLIST tag [attribut type #mode [valeur]]* >

- Définit la liste d'attributs pour une balise, comme par exemple les attributs genre et ville pour la balise auteur, et l'attribut ville pour la balise éditeur :

```
<!ATTLIST auteur genre CDATA #REQUIRED
                 ville CDATA #IMPLIED>

<!ATTLIST editeur ville CDATA #FIXED "Paris">
```

DTD

Déclaration d'attributs

- Obligatoire :

```
<!ATTLIST elt attr CDATA #REQUIRED>
```

  - Lors du traitement par le processeur, si un attribut obligatoire n'est pas présent dans un élément, cela provoquera une erreur

- Optionnel :

```
<!ATTLIST elt attr CDATA #IMPLIED>
```

- Avec une valeur fixe :

```
<!ATTLIST elt attr CDATA "valeur">
```

  - Lors du traitement par le processeur, si la valeur d'un attribut à valeur fixe est différente de la valeur qui lui a été fixée dans la DTD, cela provoquera une erreur

TD1

- Exercice 1
- Exercice 2
- Exercice 3

DTD

Déclaration d'attributs identificateurs

Type ID

- Permet d'associer à un élément un identificateur unique
- Exemple :
  - Chaque produit dans un magasin doit avoir un code unique
- Déclaration :

  <!ATTLIST elt attr ID>

  Soit pour le magasin :

  <!ATTLIST produit code ID>
- Cette valeur doit être évidemment unique. Dans le cas contraire, le processeur XML renverra une erreur d'analyse lorsqu'il rencontrera un second identificateur identique

DTD

Déclaration d'attributs identificateurs

- Exemple : magasin.xml

```
<!DOCTYPE magasin [
<!ELEMENT magasin (service+)>
<!ELEMENT service (produit*)>
<!ATTLIST service code ID #REQUIRED>
<!ELEMENT produit (#PCDATA)>
<!ATTLIST produit code ID #REQUIRED> ]>
<magasin>
   <service code="A001">
     <produit code="DE205"> Soupe </produit>
     <produit code="TM206"> Condiment </produit>
     <produit code="KJ227"> Conserve </produit>
   </service>
<service code="A003">
     <produit code="OU152"> Lessive </produit>
     <produit code="AH070"> Essuie-tout </produit>
   </service>
</magasin>
```

On vérifie que service et produit sont accompagnés chacun d'un attribut code dont la valeur est à chaque fois différente

DTD

Déclaration d'attributs identificateurs

- Question : comment indiquer qu'un produit est associé à différents services ? Est-ce que cette solution est valide ?

```
<!DOCTYPE magasin [
<!ELEMENT magasin (service+)>
<!ELEMENT service (produit*)>
<!ATTLIST service code ID #REQUIRED>
<!ELEMENT produit (#PCDATA)>
<!ATTLIST produit code ID #REQUIRED> ]>
<magasin>
    <service code="A001">
      <produit code="DE205"> Soupe </produit>
      <produit code="TM206"> Condiment </produit>
      <produit code="KJ227"> Conserve </produit>
    </service>
    <service code="A003">
      <produit code="OU152"> Lessive </produit>
      <produit code="AH070"> Essuie-tout </produit>
      <produit code="KJ227"> Conserve </produit>
    </service>
</magasin>
```

DTD

Déclaration d'attributs identificateurs

- Réponse : référencer par : IDREF
  - Permet à une valeur d'attribut de faire référence à l'identificateur (ID) d'un autre élément
    - De cette manière, il est possible de relier des éléments entre eux
  - *IDREFS* permet d'associer plusieurs identificateurs (ID) en les séparant par des espaces blancs dans la valeur d'un attribut

Exemple : magasin-ref.xml

mais on ne doit pas répéter un produit à cause du fait qu'on lui a associé un ID

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE magasin[
    <!ATTLIST magasin codes_services IDREFS #IMPLIED>
    <!ELEMENT magasin (service+)>
    <!ELEMENT service (produit+)>
    <!ATTLIST service code ID #REQUIRED>
    <!ELEMENT produit (#PCDATA)>
    <!ATTLIST produit  code ID #REQUIRED code_service IDREF #REQUIRED>]>
<magasin codes_services="A001 A003">
    <service code="A001">
        <produit code="E205" code_service="A001"> Savon </produit >
         <produit  code="E206" code_service="A001"> Essuie-tout </produit >
         <produit  code="E207" code_service="A001"> Serviettes </produit >
         <produit  code="H107" code_service="A003"> Balai</produit >
    </service>
    <service code="A003">
      <produit  code="A115" code_service="A003"> Chiffon </produit>
    </service>
</magasin>
```

DTD : Entité paramètre

- But : créer des raccourcis
  - Une entité simple permet de remplacer par un nom une portion de texte dans un document

    <!ENTITY nom "texte de remplacement">
  - L'entité (i.e. le texte de remplacement) est alors rappelée dans le document par utilisation de &nom
- Exemple :

    <!ENTITY magasin "Primevère">
  - définit une entité magasin correspondant à la chaîne de caractères "Primevère"
  - le document XML

    <message>Bienvenu au &magasin</message>
  - sera compris par l'analyseur comme :

    <message>Bienvenu au Primevère</message>

TD1

- Exercice 4

Espaces de noms

Principe

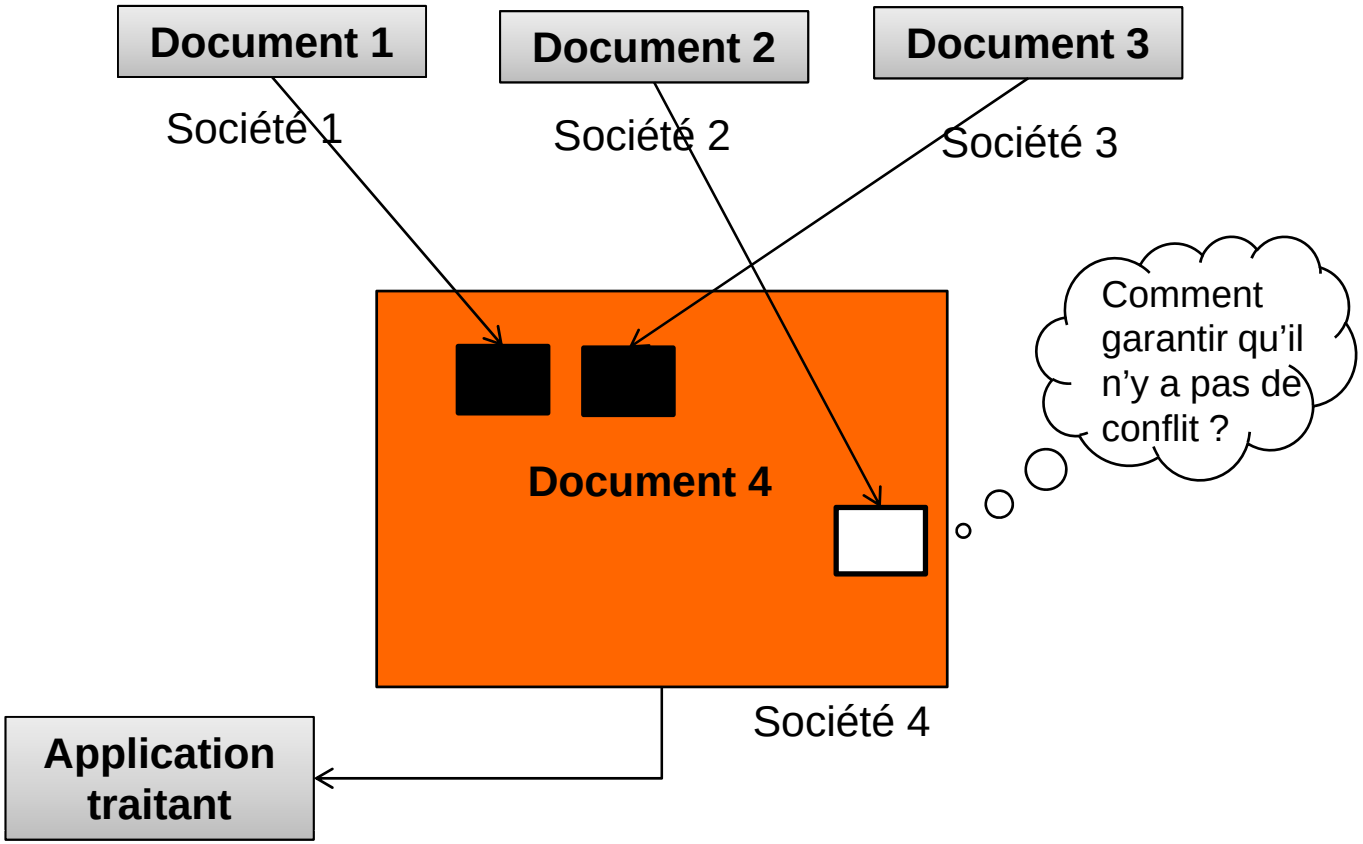
- Concept informatique très répandu
- Est utilisé dans plusieurs langages de programmation afin de prévenir d'éventuels conflits lorsqu'on accède à une valeur par un nom

En XML

- Permet de mélanger des éléments d'origines diverses
- L'espace de noms permet de délimiter la portée d'une balise, d'un attribut ou d'une valeur d'un attribut
- La notion d'espace de noms peut être vue comme un groupe d'appartenance ou d'une famille
- L'utilisation d'espace de noms garantit une forme de traçabilité de la balise et évite les ambiguïtés d'usage

Espaces de noms

- Exemple : croisement des documents



Exemple

- Cet XML décrit une table de fruits HTML :

```
<table>
    <tr>
            <td>Apples</td>
            <td>Bananas</td>
    </tr>
</table>
```

- Cet XML décrit une commande (de pièces) :

```
<table>
    <name>African Coffee Table</name>
    <width>80</width>
    <length>120</length>
</table>
```

- Si ces fragments XML avaient été ajoutés ensemble, on aurait eu un conflit de noms
- Les deux contiennent un élément <table>, mais les éléments ont un contenu et une signification différents
- Un parseur XML ne saura pas comment manipuler ces différences

Espaces de noms

- On utilise un préfixe pour résoudre ce conflit
  - Exemple
    - Cet XML donne de l'information à propos d'une table et d'une commande :

```
<h:table>
    <h:tr>
      <h:td>Apples</h:td>
      <h:td>Bananas</h:td>
    </h:tr>
</h:table>
<f:table>
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
</f:table>
```

  - Il n'y a plus de conflit car les deux éléments <table> ont des noms différents

Espaces de noms

- Comment créer un espace de noms ?
  - On le déclare en lui donnant un identifiant qui le distingue, puis on l'associe aux éléments et aux attributs ainsi rassemblés
  - Pour l'identification, le meilleur identifiant est une URI
    - L'URI n'a pas besoin de désigner un fichier existant
- Déclaration
  - La déclaration se fait par attribut, associé à un élément
  - Deux formes :
    - xmlns="*uri*"
      - ➢ définit l'espace de noms par défaut, par ex. HTML
    - xmlns:*prefix*="*uri*"
      - ➢ définit le préfixe représentant un espace de noms qualifié

Espaces de noms

- Exemple

```
<root>
  <h:table xmlns:h="http://www.w3.org/TR/html4/">
    <h:tr>
      <h:td>Apples</h:td>
      <h:td>Bananas</h:td>
    </h:tr>
  </h:table>

  <f:table xmlns:f="http://www.w3schools.com/furniture">
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
  </f:table>
</root>
```

- Dans cet exemple, l'attribut xmlns dans le tag <table>, donne à l'aide des préfixes h: et f: des espaces de noms qualifiés
- Quand un espace de noms est défini pour un élément, tous les éléments enfants avec le même préfixe sont associés au même espace de noms

Espaces de noms

- Les espaces de noms peuvent être déclarés dans les éléments où ils seront utilisés ou dans l'élément root de XML :

```
<root
    xmlns:h="http://www.w3.org/TR/html4/"
    xmlns:f="http://www.w3schools.com/furniture">

    <h:table>
      <h:tr>
        <h:td>Apples</h:td>
        <h:td>Bananas</h:td>
      </h:tr>
    </h:table>

    <f:table>
      <f:name>African Coffee Table</f:name>
      <f:width>80</f:width>
      <f:length>120</f:length>
    </f:table>

</root>
```

Espaces de noms

- Espaces de noms par défaut
  - En définissant un espace de noms par défaut pour un élément, cela évite d'utiliser des préfixes pour tous ses enfants
  - Il a la syntaxe suivante :
    - xmlns="*namespaceURI*"
  - Exemple pour la table :

```
<table xmlns="http://www.w3.org/TR/html4/" >
    <tr>
      <td>Apples</td>
      <td>Bananas</td>
    </tr>
</table>
```

  - Exemple pour la commande :

```
<table xmlns="http://www.w3schools.com/furniture" >
    <name>African Coffee Table</name>
    <width>80</width>
    <length>120</length>
</table>
```

Espaces de noms

Espaces de noms par défaut (suite 1)

- Nous pouvons changer l'espace de noms par défaut même dans les éléments enfants :
  - Dans ce cas, une règle de priorité est appliquée. Attention, les espaces de noms ne sont pas imbriqués ; on ne peut appliquer qu'un seul espace de noms à la fois
- Exemple :

```
<chapitre xmlns="http://www.masociete.com">
<paragraphe xmlns="http://www.autresociete.com">
...
</paragraphe>
</chapitre>
```

- L'élément paragraphe n'appartient pas à l'espace de noms http://www.masociete.com, mais uniquement à l'espace de noms http://www.autresociete.com

Espaces de noms

Retour sur l'espace de noms explicite

- La notion de préfixe n'a de sens que par rapport à l'URL associée
- Dans la pratique, l'application ne voit pas ce préfixe mais uniquement l'URL associée à telle ou telle partie du document
- C'est comme si un élément était un couple (nom de l'élément, espace de noms), de façon analogue à un attribut et sa valeur
  - Exemple :
    - Document 1 :

      ```
      <p:res xmlns:p="http://www.masociete.com"></p:res>
      ```

    - Document 2 :

      ```
      <zz:res xmlns:zz="http://www.masociete.com"></zz:res>
      ```

  - Les documents 1 et 2 sont strictement identiques malgré un préfixe différent

Espaces de noms

Espace de noms explicite (suite 1)

- On peut déclarer et utiliser plusieurs espaces de noms grâce aux préfixes.
- Exemple :

```
<p:res xmlns:p="http://www.masociete.com"
   xmlns:p2="http://www.autresociete.com">
   <p2:res></p2:res>
</p:res>
```

- Le premier élément res est dans l'espace de noms http://www.masociete.com
- alors que l'élément res à l'intérieur est dans l'espace de noms http://www.autresociete.com

Espaces de noms

La suppression d'un espace de noms

- Aucun espace de noms n'est utilisé lorsqu'il n'y a pas d'espace de noms par défaut ni de préfixe
- Exemple :

```
<p:element xmlns:p="http://www.masociete.com">
  <autreelement/>
</p:element>
```

- L'élément element est dans l'espace de noms http://www.masociete.com
- alors que l'élément autreelement, qui n'est pas préfixé, n'a pas d'espace de noms

La suppression d'un espace de noms (suite)

- Pour supprimer l'action d'un espace de noms, il suffit d'utiliser la valeur vide "", ce qui revient à ne pas avoir d'espace de noms
- Exemple :

```
<element xmlns="http://www.masociete.com">
   <autreelement xmlns="">.. Aucun d'espace de noms</autreelement>
   <encoreunelement>... Espace de nom par défaut</encoreunelement>
</element>
```

- L'élément element est dans l'espace de noms http://www.masociete.com
- alors que l'élément autreelement n'est plus dans un espace de noms
- L'élément encoreunelement se trouve également dans l'espace de noms http://www.masociete.com, de par l'espace de noms de son parent

Application d’un espace de noms sur un attribut

- Les espaces de noms peuvent s’appliquer via un préfixe sur un attribut ou une valeur d’attribut
- Cela sert à introduire des directives sans changer de structure
- Ou à contourner la règle qui veut que l’on ne puisse pas avoir plusieurs fois un attribut de même nom sur une déclaration d’élément
- Exemple :

```
<livre xmlns:p="http://www.imprimeur.com"
   p:quantite="p:50lots">
   <papier type="p:A4"/>
</livre>
```

- Ici, on a qualifié l’attribut quantité ainsi que les valeurs d’attribut
- 50lots et A4

- Autre utilisation d’un espace sur une valeur d’attribut
  - Permet de lever l’ambiguïté sur une valeur d’attribut
  - Par exemple, la valeur 1 :
    - est-ce une chaîne de caractères ?
    - Un nombre binaire ?
    - Hexadécimal ?
    - Un décimal ?
    - Une seconde ?

Espaces de noms

Quelques espaces de noms célèbres

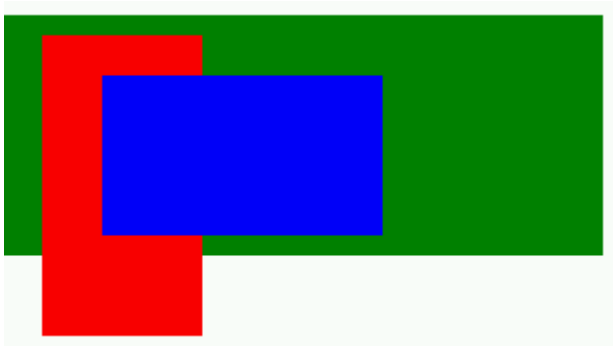
- XHTML : <xhtml:xhtml xmlns:xhtml="http://www.w3.org/1999/xhtml">
- SVG : <svg xmlns="http://www.w3.org/2000/svg">
- XSLT : <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
- Schema : <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
- RDF : <rdf:RDF xmlns:rdf="http://www.w3.org/TR/REC-rdf-syntax#">
- XSLFO : <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
- Dublin-core : <dc:title xmlns:dc="http://purl.org/dc/">
- XLink : <foo xmlns:xlink="http://www.w3.org/1999/xlink">
- SOAP : <SOAP-ENV xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/">

Espaces de noms

- Exemple : espace de nom SVG

```
<?xml version="1.0" encoding="utf-8"?>
<svg xmlns="http://www.w3.org/2000/svg">
    <title>Rectangles</title>
        <rect width="300" height="120"
         x="0" y="20" fill="green" />
        <rect width="80" height="150"
         x="20" y="30" fill="red" />
        <rect width="140" height="80"
        x="50" y="50" fill="blue" />
</svg>
```

- Résultat : essai.svg



Espaces de noms

- Comment faire pour utiliser les espaces de noms dans les documents XML valides ?
  - ➔Deux conditions :
    - Les types d'éléments et d'attributs affectés à un espace de nommage doivent être :
      - ➢ déclarés par un nom qualifié (c'est-à-dire avec un préfixe d'espace de nommage)
        - – xmlns:*prefix*="*uri*"
    - Les attributs xmlns servant à désigner les déclarations d'espaces de noms doivent être
      - ➢ déclarés dans la DTD

Espaces de noms

- Exemple : Document XML non-valide

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE notice [
   <!ELEMENT notice (#PCDATA)>
   <!ATTLIST notice style CDATA #IMPLIED>
]>
   <notice xmlns:html="http://www.w3.org/TR/REC-html40"
     html:style="font-color:red" >
     <!-- Ce document n'est malheureusement pas valide! -->
   </notice>
```

Espaces de noms

- Exemple: Document XML valide

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

<!DOCTYPE notice [
  <!ELEMENT notice (#PCDATA)>
  <!ATTLIST notice html:style CDATA #IMPLIED
     xmlns:html CDATA #FIXED "http://www.w3.org/TR/REC-html40" >
]>

<notice html:style="font-color:red">
  Ce document est valide!
</notice>
```

Espaces de noms

Explication

- L'attribut style de l'élément notice est associé à l'espace de noms html ➔Il est qualifié : html:style
- L'attribut xmlns servant à la déclaration de l'espace de noms est déclaré dans la DTD
- Cette déclaration est combinée avec le mot-clé #FIXED
  - On s'assure ainsi que dans l'élément racine notice une déclaration d'espace de noms interviendra toujours pour l'espace de noms HTML

Espaces de noms

Espaces de noms dans la DTD

- On peut avoir plusieurs espaces de noms identiques dans la DTD
- L'analyseur vérifie seulement si les noms ou attributs sont différents peu importe le préfixe qui les accompagne
- Exemple :
  - Voici une DTD avec deux espaces de noms identiques

Espaces de noms

- Espaces de noms dans la DTD
  - Exemple :

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

<!DOCTYPE ns1:notice [
  <!ELEMENT ns1:notice (ns2:notice)>
    <!ATTLIST ns1:notice xmlns:ns1 CDATA #FIXED
   "http://www.monserveur.fr" >
  <!ELEMENT ns2:notice EMPTY>
    <!ATTLIST ns2:notice xmlns:ns2 CDATA #FIXED
   "http://www.monserveur.fr" >
]>

<ns1:notice >
  <ns2:notice />
</ns1:notice>
```

Espaces de noms

Explications

- Les deux types d'éléments déclarés dans la DTD possèdent le même nom local notice, se différencient pourtant par le préfixe (ns1 et ns2)
- Ils appartiennent au même espace de noms
- Mais, du point de vue de la DTD, deux types d'éléments différents sont déclarés (à savoir ns1:notice et ns2:notice)
  - Ainsi ce document est valide

Espaces de nommage

Exemple : 2 espaces de noms : liste_collaborateurs.xml avec DTD interne

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

<!DOCTYPE col:liste_collaborateurs [
 <!ELEMENT col:liste_collaborateurs
    (col:collaborateur+)>
  <!ATTLIST col:liste_collaborateurs
    xmlns:col  CDATA #FIXED
    "http://www.monserveur.fr/adresse"
    xmlns:edv CDATA #FIXED
    "http://www.monserveur.fr/postes"
  >
 <!ELEMENT col:collaborateur (col:nom,
    col:adresse, edv:ordinateur)>
 <!ELEMENT col:adresse (col:rue, col:ville)>
 <!ELEMENT edv:ordinateur (edv:nom,
    edv:adresse)>
 <!ELEMENT col:nom (#PCDATA)>
 <!ELEMENT col:rue (#PCDATA)>
 <!ELEMENT col:ville (#PCDATA)>
 <!ELEMENT edv:nom (#PCDATA)>
 <!ELEMENT edv:adresse (#PCDATA)>
]>
```

```
<col:liste_collaborateurs
      xmlns:col="http://www.monserveur.fr/adresse"
      xmlns:rech="http://www.monserveur.fr/postes">
 <col:collaborateur>
  <col:nom>Jean-Marie Fontaine</col:nom>
  <col:adresse>
   <col:rue>13, rue de l'horloge</col:rue>
   <col:ville>13100 Aix-en-Provence</col:ville>
  </col:adresse>
  <edv:ordinateur>
   <edv:nom>Saturne</edv:nom>
   <edv:adresse>127.98.76.35</edv:adresse>
 </edv:ordinateur>
 </col:collaborateur>
 <col:collaborateur>
  <col:nom>Bernard Roux</col:nom>
  <col:adresse>
    ...
```

TD1

- Exercice 5
- Exercice 6