Page Web du module

https://sites.google.com/esp.sn/ifall/teachingsenseignements/m1-level-niveau-m1/m1_dss

Pr Ibrahima FALL

Ibrahima.Fall@esp.sn

Département Génie Informatique (DGI), Ecole Supérieure Polytechnique
(ESP) Université Cheikh Anta Diop (UCAD)
de Dakar

BP 5085 Dakar-Fann, Sénégal

Dernière mise a jour : 11/2021

PLAN

1. Historique
2. Syntaxe
3. Définition et Validation
   - DTD, W3C XML Schema
4. Feuilles de style
   - CSS, XSLT, XPath, XSLFO
5. APIs XML
6. Exemples d'applications XML

- Apache Ant
- RDF
- …

EVALUATION

- 50% Contrôle continu
  - 1 TP noté
  - 1 projet
- 50% Examen
  - Sur table.

ORGANISATION

- 32h de CM/TD/TP
- Au nom du respect mutuel **[J'INSISTE]**
  - Être en classe à l'heure prévue par l'emploi du temps
    - Prière de ne pas rentrer en classe une fois le cours commencé
  - Se garder de manger/boire en classe

- Ne pas être l'auteur de dérangements sonores (portables, bruits de machines, etc.)
- Se passer de son téléphone et d'Internet

De SGML à … XHTML.

Pourquoi XML ?
(Etat de l'art)

Formats existants :

- **HTML** = **HyperText** Markup Language
  - **SGML** = **Standard Generalized** Markup Language

Autres notations :

- **ASN.1=** Abstract Syntax Notation (ITU-T)
- CDR, XDR = Common/eXternal Data Representation
- etc.

Objectifs

- On veut représenter des données
  - par les **humains**
    - Facilement **lisibles** :

- par les **machines**

- Selon une technologie **compatible WEB**

  (à intégrer facilement dans les serveurs WEB)

- **en séparant les aspects** :
  - **présentation** *(format, couleurs, etc.)*
  - **information** *(données)*

- D'une manière **standardisée**

Qu'est que SGML?

- Une norme internationale :
  - Standard Generalized Markup Language
  - ISO 8879 – 1989
    - Premier essai normalisé pour les documents électroniques

- Un métalangage de balisage de documents
  - Lisible par l'être humain et traitable par une machine
  - Permet de définir des langages de balisage
- Les documents sont balisés conformément à la grammaire (la DTD)
  - Instances de DTD
  - Permet un balisage sémantique du fond

Critique de SGML

👍 Langage **puissant**, **extensible, standard (**ISO 8879-1986**)!**

👍 **Méta-langage** de documentation pour grosses applications

(i.e. automobile, avion, dictionnaire, etc…)

…mais

👎 **Trop complexe !** -> Implémentation beaucoup trop lourde !

👎 **Pas forcément compatible WEB !**

Qu'est que HTML?

- Version allégée de SGML dont il est une dérivée: existence de DTD HTML

- Proposé par le W3C comme format de documents sur le Web
- Langage simple avec des balises standardisées permettant la mise en forme d'un texte
- Standard du développement Web, très facile à apprendre et à utiliser
- Très industrialisé
  - Standard reconnu par tous les navigateurs.
  - Plusieurs logiciels WYSIWYG (Front Page, Dreamweather) et outils de publication de contenu (CMS) (Spip, eZPublish, PHPNuke)

Critique de HTML

👍 Langage **simple**, **lisible !** *(texte formaté)*

👍 **Compatible WEB !**

👎 **Non extensible !** (*Nombre fixe de balises et attributs)*

👎 **Mélange des genres !**

(i.e. balise de structuration et de mise en forme : <H1> title 1 </H1>*)*

👎 **Incompatibilité** entre navigateurs et versions !

👎 **Limité à la structuration de pages web** 👎 **…**

- structure *(ordre des balises),* - données *(type, valeur),* - sémantique

HTML-SGML (résumé critique)

- SGML
  - Langage de la GED **très puissant** mais très complexe
- HTML
  - Instance **simple** de SGML
  - **Adapté à la présentation**
  - Inadapté à l'échange entre programmes

Les concepteurs de XML ont cherché à exploiter les avantages de ces 2 technologies tout en se débarrassant de leurs inconvénients

XML

Définition intuitive d'XML:

- variante de **HTML généralisé** !

- **XML** :
(compatibilité WEB, lisibilité, syntaxe) - **sous-ensemble de SGML** !

(flexibilité, rigueur)

- langage à balises configurables

- pour la représentation hiérarchique de données,

http://www.w3.org/XML/

SGML

⊂

XML

Exemple de document

```
<Etudiant>
 <Age>22</Age>
 <Region> <Nom>Nord</Nom>
<Capitale>Saint-Louis</Capitale>
</Region>
 <Entrée>1995</Entrée>
 <Moyenne>15.7</Moyenne>
```

XML ∈

```
 <ValeurPrix Unite = "FCFA">
105K
```

∈

```
 </ValeurPrix>
</Etudiant>
```

Les utilisateurs peuvent définir

HTML

XHTML

leurs propres tags

d'imposer une grammaire

Il est possible

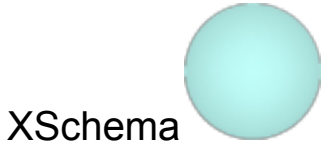
signification des sections marquées

spécifique (DTD, Schéma)

Les tags indiquent la

XML: une galaxie de standards

- **XSchema**
  - Schémas de documents
- **XSL**
  - Feuilles de styles
- **SAX**
  - API de programmation événementielle

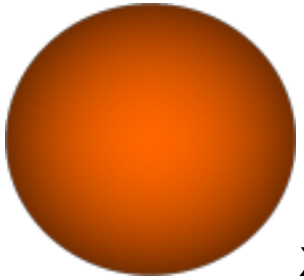


XSchema

XQuery

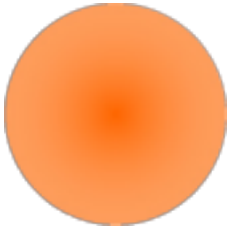
DOM

SAX



RDF

XSL



- **DOM**
  - API de programmation objet
- **SOAP**
  - Protocole Web Services
- **RDF**
  - Description de ressources Web

ebXML

- **ebXML**
  - Standards de e-Commerce
- **Xxx**
  - Standards par métiers …

Xxx

XML: un standard de fait

- Un standard d'échange
  - Lisible : texte balisé avec marquage
  - Clair : séparation du fond et de la forme
  - Extensible : supporte les évolutions applicatives
  - Sécurisé : pare-feu, encryption, signature
- Développé par le W3C
  - Pour le Web (Internet, Intranet)
  - S'étend à l'entreprise et ses partenaires
- Supporté par les grands constructeurs

- IBM, Microsoft .net, SUN, BEA, etc.
- Des outils génériques et ouverts

- Transversale à l'entreprise
  - Échanges de données, Bases de données, …
  - Bureautique, Intégration eBusiness, …
  - GED, Sites Web, …

- eXtensible

HTML

XHTML

- Reformulation de HTML 4 en tant qu'application XML 1.0
- Plusieurs avantages
- Prochaine (actuelle?) étape de l’évolution

d’Internet

Un mot sur le W3C

- Word Wide Web Consortium
- Fondé en 1994
- Consortium industriel international accueilli par différents sites

- MIT/LCS aux Etats-Unis
- INRIA en Europe
- Keio University au Japon

- 322 membres (universitaires et industriels) en mi-janvier 2011
  - http://www.w3.org/Consortium/Member/List

Structure, Eléments de documents XML, XML Bien Formé, Validité.

Structure de documents XML

- Prologue :
  - Rôle équivalent au <HEAD> HTML,
    - **instructions** de traitement
- Meta–Informations :
- Corps
  - **commentaires** *(non interprétables par le parseur)*

:

- Rôle équivalent au <BODY> HTML - **Balises** d'encadrement
  - **Attributs associées** aux balises
- Les données formatées:

(structure arborescente) - **Données encadrées** par les balises

Exemple XML : *Une lettre*

PROLOGUE

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

document XML
document autonome
instruction de traitement balise début
jeu de caractères utilisé (latin)

CORPS

<**lettre**> *balisées*

données

<**lieu**> Somewhere in space</**lieu**>

<**expéditeur**> I. Fall</**expéditeur**>

<**destinataire**> M. Camara</**destinataire**>

<**introduction**> Dear folk, </**introduction**>

<**corps_lettre**> … May the force be with you </**corps_lettre**>

<**signature**/> *données)*

balise unique (sans

</**lettre**>

balise fin

Prologue d'un document **XML**

(**Exemple**)

Jeu de caractères utilisé ***(Facultatif)*** *(Ici ISO-8859-1: jeu*

LATIN, pour prendre en compte les accents français)

Ceci est un document XML non autonome (il utilise une définition externe) ***(Facultatif)***

Document XML 1.0 ***(Obligatoire)***

```
<?xml version="1.0" standalone = "no" encoding="ISO-8859-1" ?>
<?xml-stylesheet type="text/xsl" ?>
<!DOCTYPE liste_CD SYSTEM "CDs.dtd">
```

Conforme à une définition externe (spécifié dans le fichier "CDs.dtd")

[Un commentaire spécial !] (il définit le type de document XML) ***(Facultatif)***

Autre instruction de traitement ***(Facultatif)***

Corps d'un document **XML**

(Exemple) 1

```
<liste_CD>  5                    6
<CD>        3

 <artiste type="individual">Frank Sinatra</artiste>  <titre
no_pistes="4">In The Wee Small Hours</titre>  <pistes>
 <piste>In The Wee Small Hours</piste>
<piste>Mood Indigo</piste>
 7 9
                >
</pistes  8
 <prix mannaie="euro" payement="CB">12.99</prix>
 <en_vente/>
                              4
 </CD> 3
 <CD> ….. </CD>
 …..
</liste_CD>
```

Corps d'un document XML

(arbre des balises sur l'exemple)

racine du document

liste_CD

CD

artiste titre

Frank Sinatra

In The Wee Small Hours

pistes

XML, DGI/ESP/UCAD/SN, I.Fall

piste

piste

12,99

prix

en_vente

CD

Corps d'un document XML

(explications sur l'exemple)

- Balisage arborescent *(voir le transparent précédent)*
- La **racine** du corps est **unique (1)(2)**.
- Les balises sont soit :

(2), - uniques (4).

- par **paires** : début **(1)**, et fin

- Le contenu entre deux balises paires **(3)** est soit :

  - **une valeur simple** : *chaîne de caractère* **(6)**, *numéro réel* **(7),** etc.,
  - **une arborescence d'autres balises (9)**,
  - **un mélange des deux** *(pas présent dans l'exemple).*

- Certaines balises (de début) contiennent des **attributs**

Structure des documents XML : Synthèse

- Un document XML : Prologue + Corps

(un arbre de balises)

?>

- Balises du prologue :

Instruction de traitement

<**?nom_balise_traitement ….**

Transmis directement à une application spécifique

……

<!DOCTYPE …..>

- Balises du corps **par paires** (conteneurs pour les données)

<**nom_balise** *nom_attribut1*= "val" *nom_attribut2*="val">

contenu</**nom_balise**> ■ ou **uniques**

<nom_balise_simple/>

Attributs d'une balise XML : compléments

- Attributs XML = **Données cachées** non visualisées par un navigateur (sauf si explicitement demandé)
- **Structuration "à plat"** !
- **Pas d'ordre** de précédence !
- Syntaxe : **nom='valeur'** *ou* **nom="valeur"**
- Caractères interdits : **^**, **%** et **&**
- Attributs prédéfinis
- Exemple

xml:lang="fr"

xml:id="identificateur_unique_de_la_balise"

xml:idref ="reference_vers_une_balise"

<livre langue="FR" date_debut="09/2000" id="ISBN-123"/>

Caractères de documents XML

- Le jeu de caractère d'un document XML est l'alphabet international Unicode
  - Permet pratiquement de représenter toutes les langues de la planète!
- Autres normes
  - UTF-8, UTF-16, ISO-8859-1
  - Voir http://www.w3.org/International/O-charset
- Parseur XML
  - Logiciel d'analyse d'un document XML
    - Doit au moins supporter les encodages définis par UTF-8 et UTF-16
- Les espaces

- « », « \t », « \n », … sont interprétés comme des espaces

- Le respect de la casse
  - Les identificateurs <Titre> et <titre> sont différents

Les instructions de traitement

- Nous avons vu que le prologue pouvait être complété par des instructions de traitement destinées aux applications qui vont utiliser le document XML sous la forme
  - <?**Application** instruction+ ?>
- Par exemple pour permettre au navigateur de visualiser un document avec sa feuille de style XSL on utilise l'instruction
  - <?**xml-stylesheet** type="text/xsl" ?>
- Et pour lui permettre de visualiser un document avec sa feuille de style CSS on utilise l'instruction
  - <?**xml-stylesheet** type="text/css" ?>

- Enfin, l'exemple suivant demande au navigateur de visualiser un document avec la feuille de style CSS que nous lui indiquons
  - <?**xml-stylesheet** type="text/css" href="./monCSS.css" ?>

Les références aux entités

- Une entité XML est une sorte de variable (en programmation classique)
  - Avec un nom et une valeur
  - Définie dans une DTD
- Exemple

  …
  <exple>
  Ceci est ***&MonEntite;***

</exple>

…

- ***&MonEntite;*** est une référence vers l'entité **MonEntite** définie dans la DTD du document contenant cette section; le parseur va le remplacer par sa valeur textuelle, par exple «Une illustration d'une référence vers une entité»

Les références aux caractères

- Certains caractères spéciaux sont réservés à la définition d'un document XML; ils ne peuvent donc pas être directement utilisés
  - D'autres ne sont pas accessibles au clavier
- Il est donc nécessaire de connaître la valeur de ces caractères dans l'alphabet Unicode, ou leur code numérique
- Exemples:

- **<**
  - Une référence au caractère « < »
- **>**
  - Une référence au caractère « > »
- **é**
  - Une référence au caractère « é »

Les sections CDATA

- Introduites sous la forme <![CDATA[…]]>
- Elles ne sont pas analysées par le parseur; les règles de syntaxe d'XML ne s'y appliquent donc pas

- Exemple
  - La chaine du type suivant est donc bien correcte
    - Ceci est <![CDATA[un exemple de section CDATA dans <?xml version=«1.0»?>.]]>

Document XML **bien formé**

- Conforme aux **règles syntaxiques** du langage XML !

Contre exemple

```
<?xml version = "1.0" standalone="yes"?>
<!-- Document XML pas bien formé ! -->
<peintre>
  <nom> Picasso
```

balises *chevauches*

<prenom> </prenom>
</nom> </peintre>
Pablo

- Conséquences
  - Peut être exploité par un parseur/analyseur syntaxique
    - i.e. pour parcourir l'arbre XML et le transformer
  - Association alors possible avec une feuille de style
  - Candidat pour être valide

Règles syntaxiques du langage XML

- Présence d'un prologue
- Existence d'un seul élément racine
  - Encore appelé **élément document**
- Les attributs sont associés aux balises ouvrantes et respectent la syntaxe de définition

attr="valeur"

- Toute balise qui s'ouvre doit se refermer
  - Avec respect des règles d'imbrication
- Le respect des règles de construction des identificateurs est assurée
  - Formés de caractères alphanumériques sans espaces
  - Ne commencent pas par un chiffre

Document XML **valide**

- Associé à une définition **DTD** (.dtd) ou un **Schema** (.xsd) **-Interne** au document XML **non recommandé**

- Définition : ■ Conditions

-Externe réutilisation des définitions, **échange** *(référencé vers un fichier dans le DOCTYPE)*

(dans le commentaire DOCTYPE)

- Document bien formé *(syntaxe correcte),*
- Structure du document respectant la définition (voir les DTD),
- Les références aux éléments du document soit résolues.

■ Conséquence

- Le document XML peut être échangé ! *(format standardisé)*

Exercice

- Proposez un document XML bien formé représentant un ensemble références bibliographiques
  - [On peut adapter l'exemple selon le profil des participants]
- Si vous voulez traiter ces références bibliographiques, de quoi auriez vous besoin ?
- Quelles sont vos premières réflexions sur XML ?

Exercice

- **Les réseaux GSM**

- Selon [1], à ce jour, il existe différents types de réseaux mobiles issus du GSM et servant aussi bien à émettre qu'à recevoir des appels téléphoniques et des données numériques. Le type d'un réseau GSM (2G, 3G, 4G, 5G ?) détermine son architecture, les équipements qu'il met en jeux, et son fonctionnement.
  - **Nous voulons utiliser des fichiers XML pour représenter n'importe quel réseau GSM. Proposer un arbre pour cela.**
- **Proposer au moins 2 fichiers XML représentant 2 réseaux** GSM de 2 types différents.
- **Bibliographie**
  - [1] OfficeEasy, *Principes et spécificités des types de réseaux GSM*, en ligne, https://www.officeeasy.fr/guides/amplificateur gsm/ampgsm11.php, visité le 27 juillet 2019

Premières réflexions XML #1

Ce document ne spécifie pas :

- pour les **balises** :
  - **contraintes** sur :
    - le **noms** des balises
    - **l'ordre**,
    - **la multiplicité** *(no. occurrences),*
    - **la composition** *(la hiérarchie).*
- pour les **attributs** :
  - le **nom** des attributs *(pour chaque balise)*
  - le **type** des attributs *(i.e. chaîne, énumération, etc.)*
  - les **valeurs** des attributs *(i.e domaine, format etc.)*
  - **contraintes** sur leur valeurs *(i.e format, domaine etc)*
- pour le **contenu** de balises :
  - le **type** des données

(i.e. chaîne de caractères, énumération, etc.)

Premières réflexions XML #2

- Questions :

 Quand utilise-t-on des balises et quand utilise-t-on des attributs?

balises entités

attributs propriétés

 Comment spécifie-t-on ce qui doit être affiché et comment ?

style CSS

transformations XSLT, DOM, XPath

…..

- L'ordre des attributs est-t-il une importance ?
- non

DTDs, W3 XML Schemas. 40

Introduction

- Pour un document XML
  - Une DTD ou un schéma pour décrire les balises
  - Une feuille de style pour adapter le format aux besoins
    - Nous y reviendrons au chapitre suivant
- La DTD permet de définir son propre langage basé sur XML
  - Vocabulaire
  - Grammaire

Document bien formé et valide

- Document bien formé
  - Respect des règles syntaxiques
  - **Pas nécessairement conforme à une DTD ou schema**
- Document valide
  - Bien formé + conforme à une DTD (ou un schéma)

Definition

-

Document Type **DTD**

- Langage de modélisation de XML 1.0
- Permet de définir le «vocabulaire» et la structure qui seront utilisés dans le document XML
  - Définit un type de document par des spécifications précises
    - Permet de vérifier la validité d'un document

- Grammaire du langage dont les phrases sont des documents XML (instances)
- Peut être mise dans un fichier et être appelé dans le document XML
- Assure l'uniformité d'un ensemble de documents similaires ■

Document non XML (Syntaxe héritée de SGML)

- La DTD n'est pas obligatoire
  - Un document qui fait référence à une DTD doit respecter ses spécifications

Exemple de DTD

<!ELEMENT note(de, a, objet,

description)> <!ELEMENT de(#PCDATA)>

<!ELEMENT a(#PCDATA)>

<!ELEMENT objet(#PCDATA)>

<!ELEMENT description(#PCDATA)>

Attributs et éléments

- <!ELEMENT *balise* (contenu)>
  - Décrit une *balise* qui fera partie du vocabulaire
  - ex : <!ELEMENT livre (auteur, editeur)>

- <!ATTLIST *balise* [attribut type #mode [valeur | valeur par défaut]]*>
  - Définit la liste d'attributs pour une balise déjà défini
  - ex : <!ATTLIST auteur genre CDATA #REQUIRED ville CDATA #IMPLIED> <!ATTLIST editeur ville CDATA #FIXED "Paris">

Structuration des balises

- Structuration du contenu d'une balise
  - (a, b) séquence
    - (nom, prenom, rue, ville)
  - (a|b) liste de choix

- (region | departement)

- a? élément optionnel [0,1]
  - (nom, prenom?, rue, ville)

- a* élément répétitif [0,N]
  - (produit*, client)

- a+ élément répétitif [1,N]
  - (produit*, vendeur+)

Structuration des balises: exemples

- Structuration du contenu d'une balise
- Exemple 1: Elément MONTAGNE avec un ou plusieurs nom, et une hauteur optionnelle

- *<!ELEMENT MONTAGNE(NOM+, HAUTEUR?, PAYS)>*

- Exemple 2: Elément MONTAGNE avec des sous élements à occurrence multiple
  - *<!ELEMENT MONTAGNE(NOM, HAUTEUR, PAYS)*>*
- Exemple 3: Emboîtement de sous éléments
  <!ELEMENT MONTAGNE(NOM, HAUTEUR, (DEPARTEMENT|REGION|PAYS))>

Types (pour les éléments)

- #PCDATA
  - Elément de texte sans descendants ni attributs contenant des caractères

- ANY
  - Tout texte possible - pour le développement
- EMPTY
  - Vide

Types (pour les attributs)

- CDATA
  - Données brutes qui ne seront pas analysées
- Enumération

- Liste de valeurs séparées par « | »

- ID et IDREF/IDREFS
  - Clé et référence (liste de références) pour les attributs
- ENTITY/ENTITIES
  - Nom (liste de noms) d'entités non XML déjà déclarées
- NMTOKEN/NMTOKENS
  - Mots clés (liste de mots clés)
- NOTATION
  - Notation (voir plus loin)

DTD et documents XML

- DTD interne
  - Directement dans le document XML

- **<!DOCTYPE nom_element_document [**

  … Spécifications ICI …

  ]>

- DTD externe
  - Déclarée séparément dans un autre fichier (.dtd);

<!DOCTYPE nom_element_document
SYSTEM|PUBLIC "URL" >

Exemple: DTD externe docint.dtd

<!ELEMENT doc (livre* | article+)>

```
<!ELEMENT livre (titre, auteur+)>

<!ELEMENT article (titre, auteur*)>

<!ELEMENT titre(#PCDATA)>

<!ELEMENT auteur(nom, adresse)>

 <!ATTLIST auteur id ID #REQUIRED>

<!ELEMENT nom(prenom?, nomfamille)>

<!ELEMENT prenom (#PCDATA)>

<!ELEMENT nomfamille

(#PCDATA)> <!ELEMENT adresse

ANY>
```

```
<!DOCTYPE doc SYSTEM "./docint.dtd "

> <doc>

…

</doc>
```

```
<?xml version= "1.0”?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
      1.0  Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml
      1- strict.dtd">

<html>

...

</html>
```


<?xml version= "1.0”?>

Exemple: DTD interne

<?XML version="1.0" standalone="yes"?>

```
<!DOCTYPE CATALOGUE [
        <!ELEMENT CATALOGUE (VOITURES +)>

        <!ELEMENT VOITURES (SPECIFICATION+, ANNEE, PRIX)>
        <!ATTLIST VOITURES NOM CDATA #REQUIRED>

        <!ELEMENT SPECIFICATION EMPTY>
        <!ATTLIST SPECIFICATION MARQUE CDATA #REQUIRED
         COULEUR CDATA #REQUIRED>

        <!ELEMENT ANNEE (#PCDATA)>
        <!ELEMENT PRIX (#PCDATA)>
]>


<CATALOGUE>
 <VOITURES NOM= " LAGUNA">
 <SPECIFICATION MARQUE= " RENAULT" COULEUR="Rouge"/>
 <ANNEE>2001</ANNEE>
 <PRIX>6 Millions FCA</PRIX>
 </VOITURES>
 ……...
</CATALOGUE>
```


Autre Exemple: (avec ID et IDREF)

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE DOCUMENT [
    <!ELEMENT DOCUMENT(PERSONNE*)>
    <!ELEMENT PERSONNE (#PCDATA)>
    <!ATTLIST PERSONNE PNUM ID #REQUIRED>
    <!ATTLIST PERSONNE MERE IDREF #IMPLIED>
    <!ATTLIST PERSONNE PERE IDREF #IMPLIED>
]>
<DOCUMENT>
    <PERSONNE PNUM = "P1">Marie</PERSONNE>
    <PERSONNE PNUM = "P2">Jean</PERSONNE>
    <PERSONNE PNUM = "P3" MERE="P1"
    PERE="P2">Pierre</PERSONNE> <PERSONNE PNUM = "P4"
    MERE="P1" PERE="P2">Julie</PERSONNE>
</DOCUMENT>
```

Intérêt des DTD externes

- Modèle pour plusieurs documents
  - Partage des balises et structure
- Définition locale ou externe
  - <!DOCTYPE doc SYSTEM "doc.dtd">
  - <!DOCTYPE doc PUBLIC www.e-xmlmedia.com/doc.dtd>
- Exemple de document

```
<?xml version="1.0" standalone="no"?>

<!DOCTYPE VOITURES SYSTEM "voitures.dtd">

...
```

Les entités dans les DTD

- Modèle pour plusieurs documents
  - Partage des balises et structure
  - Définies dans la DTD
    - <!ENTITY nom1 "valeur">
    - <!ENTITY %nom2 "valeur">
- Entités générales
  - Utilisables dans les fichiers XML
  - &nom1;
- Entités paramètres
  - Utilisables dans les DTD
  - %nom2;
- Entités internes
  - Permet la réutilisation dans une DTD (entités paramètres)
    - <!ENTITY %nom "definition">
    - Utiliser dans la DTD par **%nom;**
  - Exemple
    - <!ENTITY **%sexe** ("homme" | "femme")> <!ATTLIST auteur sexe **%sexe;** #REQUIRED>
- Peuvent être externes
  - Pour les entités DTD (paramètres):
    - <!ENTITY **%regles** SYSTEM "./regles.dtd">
    - Référencée par **%regles;**
  - Pour les entités générales
    - <!ENTITY **documentation** SYSTEM "./doc.xml">
    - Référence dans un fichier XML **&documentation;**

Les entités dans les DTD

- Modèle pour plusieurs documents
  - Partage des balises et structure
  - Définies dans la DTD
    - <!ENTITY nom1 "valeur">
    - <!ENTITY %nom2 "valeur">
- Entités générales
  - Utilisables dans les fichiers XML
  - &nom1;
- Entités paramètres
  - Utilisables dans les DTD
  - %nom2;

Les entités dans les DTD

- Entités internes
  - Permet la réutilisation dans une DTD (entités paramètres)
    - <!ENTITY %nom "definition">
    - Utiliser dans la DTD par **%nom;**
  - Exemple

- <!ENTITY **%sexe** ("homme" | "femme")>

<!ATTLIST auteur sexe **%sexe;** #REQUIRED>

- Peuvent être externes
  - Pour les entités DTD (paramètres):
    - <!ENTITY **%regles** SYSTEM "./regles.dtd">
    - Référencée par **%regles;**

- Pour les entités générales
  - <!ENTITY **documentation** SYSTEM "./doc.xml">
  - Référence dans un fichier XML **&documentation;**

Comment utiliser les entités

- Modularité
  - Définir dans des entités séparées les parties réutilisables
- Précédence
  - Regrouper les déclarations d'entités en tête
- Spécificité
  - Éviter les DTD trop générales

- Simplicité
  - Découper les DTD trop complexes

Les entités non-XML

- Une entité non-XML est un bloc d'information qui ne sera pas analysé par un parseur XML
  - Les données de ce bloc peuvent donc avoir un format quelconque et ne pas respecter les règles syntaxiques des documents XML
- Exemples
  - Les entités précédemment vues sont des entités XML
  - <!ENTITY Inclusion0 SYSTEM "toto.xml">
    - &Inclusion0;

- Est une entité XML
- <!ENTITY %Inclusion1 SYSTEM "toto.dtd">
  - %Inclusion1;
  - Est une entité XML
- <!ENTITY Inclusion2 SYSTEM ”image.png” >
  - Est une entité non XML
  - &Inclusion2;

Les notations

- Les notations identifient par leur nom le format des entités non-XML ou d’un élément possédant un attribut de type NOTATION
- La déclaration d’une notation comprend
  - Le nom

- Utilisé dans les autres déclarations (d'entités, d'attributs, etc.)
  - Un identifiant externe
    - Permettant au parseur d'identifier l'application qui doit traiter les données identifiées par la notation

- Une notation permet de déclarer une entité non-XML et d'y associer une application capable de traiter les données

Les notations: exemple

- Déclaration de deux formats de données compressées avec les applications qui permettent de les traiter
  - <!NOTATION gzip SYSTEM "gzip.exe">
  - <!NOTATION compress SYSTEM "compress.exe" >

- Déclaration d'une entité non-XML qui référence le fichier arch1.z au format compressé par gzip.exe
  - <!ENTITY arch1 SYSTEM "arch1.z" NDATA gzip>
- Déclaration d'un type d'élément <archive>
  - <!ELEMENT archive EMPTY>
  - <!ATTLIST archive
    Codage NOTATION(gzip|compress)#REQUIRED
    Contenu ENTITY #IMPLIED>
- Dans un document XML, on fait référence à l'entité **arch1** contenant les données à compresser
  - <archive Codage= "gzip" Contenu= "arch1" />

Synthèse DTD

- Spécification de la structure du document

Déclaration de balisage : ELEMENT, ATTLIST, ENTITY;

Déclaration des éléments

- Eléments simples :
  - **vide** (EMPTY)
  - **textuel** (#PCDATA)
  - **libre** (ANY),
- Composition :
  - **séquence d'éléments** liste ordonnée (a,b,c)
  - **choix alternatives** d'éléments (a|b|c)
  - **mixte hiérarchique** (a, (b|c),d)
- Indicateurs d'occurrences :
  - **?** (zéro ou une)),
  - **+** (une ou plusieurs)
  - * (zero ou plusieurs),

Insuffisance des DTD

- Tout doit être défini
  - Pas de modélisation partielle
- Pas de types de données
  - Difficile à interpréter
    - Syntaxe différente de celle des documents XML
  - Difficile à traduire en schéma objets
  - Pas d'héritage
  - Typage faibles
    - Pas de contraintes sur les données
- Propositions de compléments
  - XML-Schema du W3C

Exercice

- Soit la structure arborescente suivante
  - contenu
    - introduction
      - histoire
      - etat_actuel
    - premiers pas
      - un petit exemple
- Cette structure peut bien être représentée par un document XML valide par rapport à la DTD suivante. Créer ce document XML.

```
<!ELEMENT liste (point)*>
<!ELEMENT point (#PCDATA)>
<!ATTLIST point nom ID #REQUIRED point_parent IDREF
#IMPLIED >
```

Solution de l'exercice

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

<!DOCTYPE liste SYSTEM "liste.dtd">

<liste>
    <point nom="racine">contenu</point>
    <point nom="introduction" point_parent="racine">introduction</point>
    <point nom="histoire" point_parent="introduction">histoire</point> <point
    nom="aujourd_hui " point_parent="introduction">etat_actuel</point> <point
    nom="pas" point_parent="racine">premiers pas</point> <point
    nom="exemple" point_parent="pas">un petit exemple</point>

</liste>
```

Objectifs des schémas

- Reprendre les acquis des DTD
  - Plus riche et complet que les DTD
- Permettre de typer les données
  - Eléments simples et complexes
  - Attributs simples
- Permettre de définir des contraintes
  - Occurrence obligatoire ou optionnelle
  - Cardinalités, références
- Permettre une modélisation partielle des documents
- Permettre une réutilisation de définitions existantes, avec les espaces de nommages
- …

W3 XML Schema

- Un schéma d'un document définit
  - Les éléments possibles dans le document
  - Les attributs associés à ces éléments
  - La structure du document et **les types de données**
- Le schéma est spécifié en XML
  - Pas de nouveau langage
  - Balisage de déclaration
  - Espace de nommage
- Présente de nombreux avantages
  - Structures de données avec types de données
  - Extensibilité par héritage
  - Analysable par un parseur XML standard

Définition d'un schema

- Document XML .xsd
- <schema> est l'élément racine

```
<?xml version="1.0"?>
<xsd:schema>
 //corps du schema.. //...
</xsd:schema>
```

- <schema> peut contenir certains attributs
- La déclaration d'un schema est souvent comme suit

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    //...
    //...
</xsd:schema>
```

Référencer un schéma XML

- Ajouter la référence au niveau de la **balise racine** du document XML

```
<?xml version="1.0"?>
<note xmlns:xsi="http://www.w3.org/2001/XMLSchema
instance" xsi:schemaLocation="chemin_fichier.xsd">
<to>:Cq;c rvc tyyt bqw</to>
<from>Ibrahima</from>
<heading>Reminder</heading>
<body>Don't forget it this week-end!</body>
</note>
```

Déclaration d'un élément simple

- Un élément simple contient des données dont le type est simple
  - Exemple: types de base en java
- Un élément simple est défini selon la syntaxe suivante

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

- Exemple en schéma XML

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

  - Correspondance en Document XML)

```
<Department >13</Department>
```

- Autres exemples
  - <xsd:element name="color" type="xsd:string" default="red"/>
    - Valeur par défaut
  - <xsd:element name="color" type="xsd:string" fixed="red"/>
    - Valeur inchangeable

Les types simples #1

Type Description

string *représente une chaîne de caractères.*

boolean *représente une valeur booléenne true ou false.*

decimal *représente un nombre décimal*

float *représente un nombre à virgule flottante.*

double *représente un nombre réel double.*

duration *représente une durée*

dateTime *représente une valeur date/heure.*

time *représente une valeur horaire (format : hh:mm:ss.sss).* **date** *représente une date (format : CCYY-MM-DD).* **gYearMonth** *représente un mois et une année grégorienne (format : CCYY-MM)*

Les types simples #2

Type Description

gYear *représente une année (format : CCYY).*

gMonthDay *représente le jour d'un mois (format : MM-DD)*

gDay *représente le jour d'un mois (format : DD).*

gMonth *représente le mois (format : MM).*

hexBinary *représente un contenu binaire hexadécimal.*

base64Binary *représente un contenu binaire de base 64.*

anyURI *représente une adresse URI (ex.: http://www.site.com).*

QName *représente un nom qualifié.*

NOTATION *représente un nom qualifié.*

Les types simples #3

Type Description

Token *représente une chaîne de caractères sans espaces blancs* **Language**

représente un langage exprimé sous forme de mot clés

NMTOKEN *représente le type d'attribut NMTOKEN (alphanumérique et . : - _)*

NMTOKENS *représente le type d'attributs NMTOKEN + espace* **Id**

représente le type d'attribut ID

IDREF, IDREFS *représente le type d'atribut IDREF, IDREFS*

ENTITIES

ENTITY, ENTITIES

représente le type ENTITY,

Integer *représente un nombre entier*

nonPositiveInteger *représente un nombre entier négatif incluant le zéro* **negativeInteger**

représente un nombre entier négatif dont la valeur maximum est -1

Les types simples #4

Type Description

long *représente un nombre entier long dont l'intervalle est : {-9223372036854775808 - 9223372036854775807}*

int *représente un nombre entier dont l'intervalle est : {-2147483648 - 2147483647}*

short *représente un nombre entier court dont l'intervalle est {-32768 - 32767}*

byte *représente un entier dont l'intervalle est {-128 - 127}*

nonNegativeInteger *représente un nombre entier positif incluant le zéro*

unsignedLong *représente un nombre entier*

long *non-signé dont l'intervalle est {0 - 18446744073709551615}*

Les types simples #5

Type Description

unsignedInt *représente un nombre entier non-signé dont l'intervalle est : {0 - 4294967295}*

unsignedShort *représente un nombre entier court non-signé dont l'intervalle est : {0 - 65535}*

unsignedByte *représente un nombre entier non-signé dont l'intervalle est {0 - 255}*

positiveInteger *représente un nombre entier positif commençant à 1*

Déclaration d'un attribut

- Tous les attributs sont de type simple
  - Un attribut est défini selon la syntaxe suivante
    - <xsd:attribute name = "……" type= "……" />
- Exemple
  - <xsd:attribute name="language" type="xsd:string"/>
    <xsd:attribute name="country" type="xsd:NMTOKEN" fixed="FR"/>
- Association d'un attribut à un élément
  - <xsd:element …> <xsd:attribute ……/> </xsd:element>

Déclaration d'un élément complexe

- Un élément complexe contient des données dont le type est complexe
  - Une structure, par exemple
- 3 compositeurs existe pour définir les 3 catégories essentielles de types complexes
  - **<sequence>**
    - Collection ordonnée d'éléments typés
  - **<all>**
    - Collection non ordonnée d'éléments typés
  - **<choice>**
    - Choix entre éléments typés

Les types complexes

- Déclarer un élément complexe = définir son type + association du type à l'élément
- Deux façons de déclarer un élément complexe
  1. Inclure la définition du type dans la déclaration de l'élément

Document XML

```
<employee>
        <firstname>John</firstname>
        <lastname>Smith</lastname>
    </employee>
```

Schéma XML correspondant :

```
<xsd:element name="employee">
        <xsd:complexType>
        <xsd:sequence>
                <xsd:element name="firstname" type="xs:string"/>
                <xsd:element name="lastname" type="xs:string"/>
</xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Les types complexes

2. Exclure la définition du type de la déclaration de l'élément

Schéma XML correspondant au document précédent

```
<xsd:element name="employee" type="personinfo"/>
//……
<xsd:complexType name="personinfo">
    <xs:sequence>
        <xsd:element name="firstname" type="xs:string"/>
        <xs:element name="lastname" type="xs:string"/>
    </xsd:sequence>
</xsd:complexType>
```

- Cette seconde déclaration permet la réutilisation de types

Exemple

- <xsd:element name="employee" type="personinfo"/>
- <xsd:element name="student" type="personinfo"/>

Le compositeur sequence

- Spécifie que les éléments fils doivent apparaître dans un ordre spécifique
- Exemple

```
<xsd:element name="adresse">
   <xsd:complexType>
 <xsd:sequence>
 <xsd:element name="name" type="xsd:string"/>
<xsd:element name="street" type="xsd:string"/>
<xsd:element name="city" type="xsd:string"/>  <xsd:element
name="state" type="xsd:string"/>  <xsd:element name="zip"
type="xsd:decimal"/> </xsd:sequence>
      <xsd:attribute name="country" type="xsd:NMTOKEN" fixed="FR"/>
</xsd:complexType>
</xsd:element>
```

Exemple d’attribut déclaré dans un type