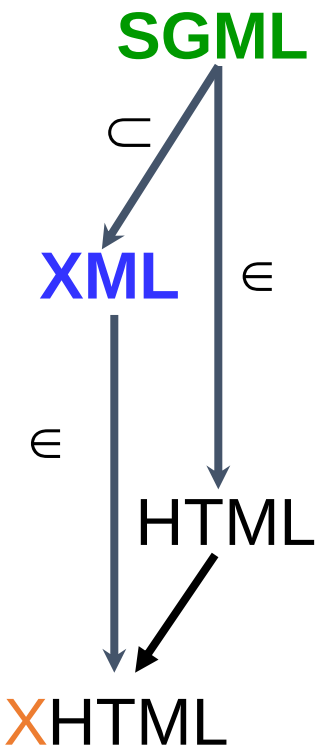
Université de Montréal

Extensible Markup Language XML

Jamel Eddine Jridi

XML

- Basé sur les principes de SGML
- Recommendation W3C depuis 1998.
- Méta-langage pour représenter les données échangés sur le web
- XML permet de définir des modèles pour des documents textuels
- XML est un format de description des données et non de leur représentation
- HTML est un langage de balisage, XML est utilisé pour définir des langages de balisage
- Définir des balises personnalisées.
- XML fournit un moyen de vérifier la syntaxe d'un document grâce aux schémas.



XHTML prend l'avantage de XML (...) et l'avantage de HTML qui sont les elements connue de HTML comme h1 et p

Avantages de XML

FALIDUE

- Flexible et personnalisable : il n’y a pas d’ensemble de balises fini
- Lisibilité : compréhensible par les humains
- Autodescriptif et extensible
- Universalité et portabilité : prise en compte des caractères spéciaux
- Déployable : il peut être facilement distribué par n'importe quels protocoles à même de transporter du texte, comme HTTP.
- Intégrabilité : application d'analyse des strucutres XML (Parser)
- Extensibilité : un document XML est utilisable dans tous les domaines d'applications

Syntaxe XML

Expliquez ce que sont les notions de bonne formation et de validation en XML . Quel est
le but de chacune de ces opérations et comment on les met en application.

- Les niveaux de syntaxe
  - Documents **bien formés** sont conformes aux règles XML de base
  - Documents **valides** sont **bien formés** et également **conformes** à un schéma qui définit les détails du contenu autorisé
- XML bien formé
  - Une seule balise racine qui contient tous les autres balises dans un document
  - Les balises doivent toujours être fermés (`<tag></tag>` ou `<tag />`)
  - Ordre d'imbrication

    ordre d'arborecsence -- respecter les fermeteur requits
  - Les balises peuvent contenir des attributs : valeurs écrites entre guillemets.
  - Nom d'un élément :
    - Débute avec une lettre ou _
    - Ne peut pas commencer par `xml`
    - Peut contenir aussi des chiffres – . :
    - Dépend de la casse

      def. Majuscule et Miniscule dans le schema doivent etre respecte
- XML valide

  un document XHTML est un document XML!!
  - XML fournit un moyen de vérifier la syntaxe d'un document grâce aux schémas

Exemple XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE catalog SYSTEM "cd.dtd">
<catalog>
        <cd code="BD-EB">
             <title tag="essai">Empire Burlesque</title>
             <artist>Bob Dylan</artist>
             <country>USA</country>
             <company>Columbia</company>
             <price>10.90</price>
             <year>1985</year>
        </cd>
        <cd>
             <title>Greatest Hits</title>
             <artist>Dolly Parton</artist>
             <country>USA</country>
             <company>RCA</company>
             <price>9.90</price>
             <year>1982</year>
        </cd>
</catalog>
```

l'encodage du document

racine du doc.

Validation XML

- Est-ce que les éléments sont imbriqués selon ce qui est attendu ?
- Est-ce que le contenu des balises correspond à des valeurs attendues ?
- Définition d'un schéma :
  - DTD
  - XML Schema
  - Relax NG Compact (RNC)
- Structure arborescente
  - Imbrication des balises
  - Ordre d'apparition
  - Validation syntaxique

Les mots en GROS sont importants pour avoir des questions en examen.

Document Type Definition
DTD

- Document permettant de décrire un modèle de document XML
- Modèle: grammaire de classe de documents
- **document valide** pour un document XML comportant une DTD
- **document bien formé** pour un document XML ne comportant pas de DTD mais répondant aux règles de base du XML
- Une DTD peut être définie de 2 façons :
  - sous **forme interne**, c'est-à-dire en incluant la grammaire au sein même du document

```
<!DOCTYPE root-element [
declarations
]>
```

  - sous **forme externe**, soit en appelant un fichier contenant la grammaire à partir d'un fichier local ou bien en y accédant par son URL.

```
<!DOCTYPE root-name SYSTEM "file-name">
```

Syntaxe DTD (élément)

- Déclaration d'élément simple : `<! ELEMENT balise (type_prédéfini) >`
  - `ANY` : L'élément peut contenir tout type de donnée
  - `EMPTY` : L'élément ne contient pas de données spécifiques
  - `#PCDATA` : L'élément doit contenir une chaîne de caractère
- Composition d'élément : `<! ELEMENT balise (composition) >`
  - Définit une séquence ou un choix d'éléments
  - Utilisation des opérateurs de composition d'éléments

Opérateur	Signification	Exemple
+	L'élément doit être présent au minimum une fois	A+
*	L'élément peut être présent plusieurs fois (ou aucune)	A*
?	L'élément peut être optionnellement présent	A?
|	L'élément A ou l'élément B peuvent être présents	A|B
,	L'élément A doit être présent et suivi de l'élément B	A,B
()	Les parenthèses permettent de regrouper des éléments afin de leur appliquer les autres opérateurs	(A,B)+

Exemple DTD (1)

Comment definir que l'ordre n'est pas important?!

```
<!ELEMENT personne (nom, prenom+, tel?, adresse) >
<!ELEMENT nom (#PCDATA) >
<!ELEMENT prenom (#PCDATA) >
<!ELEMENT tel(#PCDATA) >
<!ELEMENT email (#PCDATA) >
<!ELEMENT adresse (ANY) >
```

```
<personne>
    <nom>Hugo</nom>
    <prenom>Victor</prenom>
    <prenom>Charles</prenom>
    <tel>01120243</tel>
    <adresse>
       <rue></rue>
       <ville>Paris</ville>
    </adresse>
</personne>
```

Syntaxe DTD (attribut)

<adresse numero="1">

- `<! ATTLIST balise attribut type mode >`
- `balise` : spécifie l'élément auquel l'attribut est attaché
- `attribut` : représente le nom de l'attribut déclaré
- `type` : défini le nom de l'attribut déclaré
  - `CDATA` : chaîne de caractères entre guillemets
  - Enumération : Liste des valeurs séparés par `|`
  - `ID` et `IDREF` : clé et une référence à un clé
- `mode` : `#REQUIRED`, `#IMPLIED` ou `#FIXED`
- Exemple :

```
<!ELEMENT publication (#PCDATA) >
<!ATTLIST publication
                    pub_Id ID #REQUIRED
                    type (journal | workshop ) #REQUIRED >
<!ELEMENT citation EMPTY >
<!ATTLIST citation
                   cid ID #REQUIRED
                   ref IDREF #IMPLIED >
```

<publication pub-id="123" type = "journal">
ABC
</publication>

<citation cid ="123"/>

voiture ref-conducteur = "pxy123"
conduteur id="pxy123"

(ref) sera valide juste si on a deja un id de meme valeur a associer

Exemple DTD (2)

DTD de facon interne

```
<!DOCTYPE Restaurant [
   <!ELEMENT Restaurant (Nom, Adresse, (Telephone | Manager), Menu?) >
   <!ATTLIST Restaurant
             categorie CDATA #REQUIRED
             type CDATA #FIXED "francais" >
   <!ELEMENT Nom (#PCDATA)>
   <!ELEMENT Adresse (No, Rue, Ville)>
   <!ELEMENT No (#PCDATA)>
   <!ELEMENT Rue (#PCDATA)>
   <!ELEMENT Ville (#PCDATA)>
   <!ELEMENT Telephone (#PCDATA)>
   <!ELEMENT Manager (#PCDATA)>
   <!ELEMENT Menu EMPTY>
   <!ATTLIST Menu Nom CDATA #REQUIRED>
]
```

ordre doit etre respecte

Restaurant catagorie="3f"
type = "francais" //le type est fixé à etre nomé justement francais

*** FAIS ATTENTIONS SUR Majuscule et Minuscule ****

Syntaxe DTD (entité)

- Permet de définir un groupe d'éléments sous un nom (macro)

```
<!ENTITY %nom "definition">
```

- Réutilisable dans une DTD par un simple appel : `%nom;`
- Exemple :

```
<!ENTITY %genres "(homme | femme)">
<!ATTLIST auteur genre %genres;  #REQUIRED>
```

- Peut être externes :

```
<!ENTITY %book PUBLIC "book.dtd">
```