INU3011 Documents structurés

Cours 5

Le langage des DTD

Varia

- Les liens vers les émojis fonctionnent !
  - Full Emoji List, v13.1

    https://unicode.org/emoji/charts/full-emoji-list.html
  - Emojipedia

    https://emojipedia.org/
- "Documents différents"
  - Préambule aux Exercices après Cours 4

Plan

- Rappel : TP 2 dû demain 7 février 23h55
- *Le langage des DTD*
  - Déclarations d'élément (ELEMENT)
    - Modèles de contenu
    - Modélisation et micro-modélisation
    - Relations entre modèles de contenu
  - Déclarations d'attribut (ATTLIST)
    - Types de valeur

Le langage des DTD

Déclarations contrôlant la validité

- Les déclarations ELEMENT déterminent quels types d'éléments sont acceptés, et ce que chacun peut et/ou doit *contenir*
- Les déclarations ATTLIST déterminent ce que chaque type d'éléments *peut* et/ou *doit* avoir comme attributs
- Rappel : Ces restrictions ne sont vérifiées que par les applications *validantes*

Déclarations ELEMENT

- Forme générale:

  <!ELEMENT *id-gen modèle-de-contenu*>
- Annonce que des éléments de type *id-gen* peuvent apparaître dans les documents
- *modèle-de-contenu* indique ce que les éléments *id-gen* *doivent et/ou peuvent contenir* (peu importe où ils surviennent dans les documents)

Modèles de contenu

Faire tous les exemples
dans oXygen

(en un seul fichier)

Modèle de contenu textuel (1/2)

- (#PCDATA)
- Texte seulement: sous-éléments interdits
- Convient pour une information qu'on ne souhaite pas décomposer, par exemple:
  - nom, date, numéro d'assurance sociale, etc.
- Accepte aussi un contenu vide !
  - Considéré comme une lacune importante des DTD

Modèle de contenu textuel (2/2)

- Ex.: <!ELEMENT nom (#PCDATA)>

- Dans le document:

  <nom>Jacques</nom> OK

  <nom><titre>Frère</titre> Jacques</nom>
  INVALIDE

  <nom></nom> OK

  <nom><!--Oh -->Henry</nom> OK

Modèle de contenu vide (1/2)

- EMPTY
- Contenu doit toujours être vide
  - Pas de texte ni espace, pas de sous-élément
  - Même pas de commentaire !
- Convient pour marquer un endroit dans un document; ex.: <img/> et <hr/> en HTML
- ou pour véhiculer une information **oui / non** ou **vrai / faux**

Modèle de contenu vide (2/2)

- Ex.: confidentialité

  <!ELEMENT confidentiel EMPTY>

  Dans le document:

  <confidentiel/> OK

  <confidentiel></confidentiel> OK

  <confidentiel>oui</confidentiel> INVALIDE

  <confidentiel><oui /></confidentiel> INVALIDE

Nom d'élément (1/2)

- Un simple *nom d'élément* (= identificateur générique) est un modèle de contenu possible
  - À condition que ce type d'éléments soit défini ailleurs dans la DTD, dans une autre déclaration ELEMENT
  - Rarement utilisé seul, mais c'est la base pour construire des modèles de contenu plus complexes

Nom d'élément (2/2)

- Exemple: auteur identifié par son nom

  <!ELEMENT auteur (nom)>

  <!ELEMENT nom (#PCDATA)>

- Dans le document:

  <auteur><nom>Jean Roy</nom></auteur> OK

  <auteur>Jean Roy</auteur> INVALIDE

  <auteur><matricule>007</matricule></auteur>

  INVALIDE

Opérateurs de liaison

- Il y en a deux:
  - séquence: "`,`" (virgule)
  - choix: "`|`" (barre verticale)

Séquence « , » (1/3)

- Forme générale:

  ((*modèle de contenu 1*)**,** (*modèle de contenu 2*))

- Signifie:
  - *Présence obligatoire, dans l'ordre*
- Exemple:

  <!ELEMENT nom (prénom, nomFamille)>

Séquence « , » (3/3)

- Exemple:

  <!ELEMENT date (an, mois)>

- Dans le document:

  <date><an>2015</an><mois>01</mois></date>
  OK

  <date><mois>01</mois><an>2015</an></date>
  INVALIDE

  <date><an>2015</an></date> INVALIDE

Séquence « , » (2/3)

- Peut lier plus que deux composantes:

  (an, mois, jour)

Choix « | » (1/3)

- Forme générale:

  (*modèle-de-contenu-1* **|** *modèle-de-contenu-2*)

- Signifie:
  - Présence obligatoire de l'un OU l'autre, mais pas les deux
- Exemple:

  <!ELEMENT sexe (masculin | féminin)>

Choix « | » (2/3)

- Peut lier plus que deux composantes :

  (avion | train | autobus | autre)

Choix « | » (3/3)

- Exemple:

  <!ELEMENT transport (train | avion | autre)>

- Dans le document:

  <transport><avion>...</avion></transport> OK

  <transport><autre>...</autre></transport> OK

  <transport>train</transport> INVALIDE

  <transport><avion>...</avion>
  <autre>...</autre></transport> INVALIDE

Opérateurs d'occurrence

- Formes générales:

  *modèle-de-contenu***?**

  *modèle-de-contenu******

  *modèle-de-contenu***+**

- Signification:
  - \+ : Répétable
  - * : Facultatif-et-répétable
  - ? : Facultatif

Il ne doit pas y avoir de blanc (espace, etc.) avant l'opérateur d'occurrence.

Répétable (+)

- Ex.: <!ELEMENT noms (nom)+>
- Dans le document:

<noms><nom>…</nom></noms> OK

<noms><nom>…</nom>
<nom>…</nom></noms> OK

<noms> </noms> INVALIDE

Facultatif-et-répétable (*)

- Ex.: <!ELEMENT noms (nom)*>
- Dans le document:

<noms><nom>…</nom></noms> OK

<noms><nom>…</nom>
<nom>…</nom></noms> OK

<noms> </noms> OK

Facultatif (?)

- Ex.: <!ELEMENT résumé (para)?>
- Dans le document:

  <résumé><para>…</para></résumé> OK

  <résumé></résumé> OK

  <résumé><para>…</para>
  <para>…</para></résumé> INVALIDE

Combinaisons d'opérateurs

- Exemple avec , et ? :

  (a, b, (c)?, d, (e)?)

  – a, b et d sont obligatoires

  – c et e sont facultatifs, aux endroits où ils figurent dans la séquence

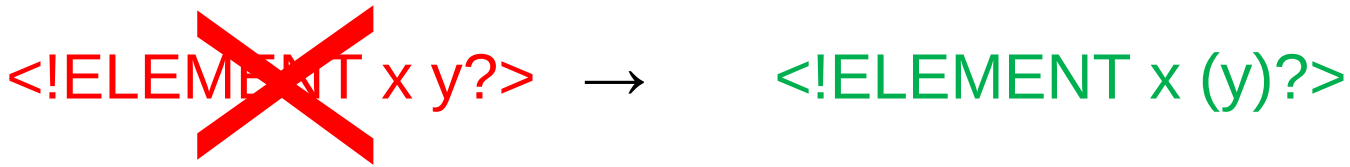
<a/><b/><d/>	OK
<a/><b/><c/><d/>	OK
<a/><b/><d/><e/>	OK
<a/><b/><c/><d/><e/>	OK

N.B.: Omission des parenthèses

- Si un opérateur d'occurrence (+, *, ?) s'applique à un *simple nom d'élément*, les parenthèses peuvent être omises autour du nom d'élément :

  (a, b, (c)?, d, (e)?) → (a, b, c?, d, e?)

- Mais tout modèle de contenu dans son entièreté *doit* commencer par une "("



Exemple avec | et ,

(nom, (tél | courriel))

<nom/><tél/>	OK
<nom/><courriel/>	OK
<nom/><tél/><courriel/>	INVALIDE
<nom/>	INVALIDE

Exemple avec , et +

- (A, B)+
  - accepte: <A/><B/>
    <A/><B/><A/><B/>
    <A/><B/><A/><B/><A/><B/>
    etc.
  - refuse: vide
    <A/>
    <B/><A/>
    <A/><B/><A/>
    etc.

Exemple avec | et ?

- (A | B)?
  - accepte: <A/>
    <B/>
    vide
  - refuse: <A/><B/>
    <B/><A/>
    etc.

Exemple avec | et *

- (A | B)*
  - accepte: vide
    <A/>
    <B/>
    <A/><A/>
    <A/><B/>
    <B/><A/>
    <B/><B/>
    toute suite de <A/> et/ou <B/>

Exemple avec | et +

- (A | B | C)+
  - accepte: <A/>
    <A/><A/>
    <B/><B/>
    <A/><C/><B/>
    toute suite de <A/>, <B/> et/ou <C/>
  - refuse: vide
    etc.

Modèles de contenu mixte (1/2)

- La seule façon de déclarer un élément qui accepte un *contenu mixte*
  - Rappel : un contenu mixte consiste en un mélange de **texte non blanc** et de sous-élément(s)
  - Exemple:

```
<par>Il fait <emph>très</emph> chaud !</par>
```

(texte non blanc et sous-élément)

Modèles de contenu mixte (2/2)

- Distingue catégoriquement les documents structurés des bases de données
- La *seule forme permise* en XML est:

  (#PCDATA | elem1 | elem2 | ... | elem*n*)*

  Exemples :

  <!ELEMENT par (#PCDATA | emph)*>

  <!ELEMENT par (#PCDATA | emph | url)*>

Modèles de contenu mixte (3/3)

- Exemple dans oXygen en classe avec la déclaration :

  <!ELEMENT par (#PCDATA | emph | url)*>

Modélisation (1/2)

- Modéliser un type de documents consiste (entre autres) à déterminer :
  - un ensemble de (noms) d'élément
  - et un modèle de contenu pour chacun
- permettant de représenter de façon prévisible l'information véhiculée par les documents de ce type

Modélisation (2/2)

- Le modèle de contenu de l'élément-document détermine la structure générale du type de documents
- Les modèles de contenu des descendants de l'élément-document déterminent la structure plus spécifique des différents contenus rencontrés dans les documents

Micro-modélisation (1/2)

- Quand on réfléchit à la façon de représenter en XML un type de contenu précis, qui ne correspond ***pas*** à un document complet, on parle parfois de "micro-modélisation"
- C'est souvent le cas quand on réfléchit aux éléments "bas" dans la structure hiérarchique (loin de l'élément-document)

Micro-modélisation (2/2)

- Les exemples ci-dessus s'apparentaient à de la micro-modélisation
- Exemples de modèles de documents complets (sans attributs) :
  - Dossier 055-Ex-modele-simple
  - Dossier 050-Ex-utilite-validite
    - proces-verbaux-CSS

Relations entre modèles de contenu (1/5)

- Il est parfois utile de *comparer* les modèles de contenu entre eux, sur la base des contenus qu'ils acceptent ou refusent
- On définit les **cinq** relations suivantes entre deux modèles de contenu donnés…

Relations entre modèles de contenu (2/5)

- Comparativement à un autre modèle de contenu, un modèle de contenu donné est:
  - *Plus restrictif*
    - S'il accepte *moins* de contenus que l'autre modèle
  - *Plus permissif*
    - S'il accepte *plus* de contenus
  - *Équivalent*
    - S'il accepte *exactement les mêmes* contenus

Relations entre modèles de contenu (3/5)

- *Disjoint*
  - S'il n'accepte *aucun contenu en commun* avec l'autre modèle
- *Incomparable*
  - S'il accepte :
    - certains contenus refusés par l'autre modèle
    - certains contenus acceptés par l'autre modèle
  - mais refuse :
    - certains contenus acceptés par l'autre modèle

Relations entre modèles de contenu (4/5)

- Ces comparaisons sont utiles autant en micro-modélisation qu'en modélisation, au niveau de modèles de documents complets
  - On pourrait dire, par exemple :
    - "Tel modèle de documents est plus permissif (ou plus restrictif) que tel autre", p.ex. :
    - *La "TEI All" est plus permissive que la "TEI Lite"*
      - Il s'agit de deux modèles de documents complets (deux variantes de la TEI)

Relations entre modèles de contenu (5/5)

- En l'occurrence :
  - *"TEI All" est plus permissive que "TEI Lite"*
- Signifie que :
  - Tout document *TEI Lite* est aussi un document *TEI All*
  - Certains documents *TEI All* ne sont pas des documents *TEI Lite*

Exercices en classe

- (A*) versus (A+)
- (A, A+) versus (A+)
- (A, A*) versus (A+)
- (A | B | C)* versus (A | B)*
- (A | B | C) versus (A | B | D)
- (A | B | C) versus (C | B | A)
- (A+) versus (B*)
- (A*) versus (B*)

Micro-modélisation en classe

1. Dans la description d'une chanson, un élément `contrib` doit pouvoir contenir:
   - Un compositeur (`comp`) et un interprète (`int`)
   - Si les deux y sont, l'ordre doit être: `comp`, `int`
   - Il doit y avoir au moins un des deux
   - Écrire le modèle de contenu de `contrib`
2. Écrire un modèle de contenu acceptant une suite 2, 3 ou 4 éléments "`nom`"

Déclarations d'attributs

Pourquoi déclarer les attributs?

- Dans un document *bien-formé*, n'importe quel attribut peut être utilisé sur n'importe quel élément, mais…
- … dans un document *valide*, *tous* les attributs qu'on compte utiliser dans les balises de début de certains éléments doivent être déclarés
- *Prévisibilité, encore et toujours !*

Où vont les déclarations d'attributs?

- À la même place que les déclarations ELEMENT : *dans la DTD*
- C'est-à-dire:
  - Normalement : dans le fichier de déclarations pointé par la *déclaration* de type de document (DOCTYPE)
  - Mais peut aussi être dans le *sous-ensemble local de déclarations* de la déclaration DOCTYPE

Déclarations d'attributs

- Forme générale:

  <!ATTLIST *nom-élément nom-attribut type-valeurs oblig-ou-facult* >

  nom-élément : nom (type) d'élément auquel l'attribut peut s'appliquer

  nom-attribut : nom de l'attribut déclaré

  type-valeurs : genre de valeurs que peut prendre l'attribut

  oblig-ou-facult : attribut obligatoire ou non?

Un exemple

<!ATTLIST mémo confidentiel (oui | non) #IMPLIED>

nom-élément : mémo

nom-attribut : confidentiel

type-valeur : ici, type *énuméré*, avec seulement deux valeurs possibles : "oui" et "non"

oblig-ou-facult : ici #IMPLIED (signifie : facultatif)

Types d'attribut (1/2)

- Détermine ce qu'on peut inscrire comme valeur d'attribut dans une spécification d'attribut impliquant l'attribut déclaré
- `<mémo confidentiel="...">...</mémo>`

Le type de l'attribut détermine ce qu'on peut écrire entre les guillemets (qui peuvent être simples ou doubles, rappelons-nous)

Types d'attribut (2/2)

- Il existe dix types en XML :
  - CDATA, ID, IDREF, IDREFS, ENTITY, ENTITIES, NMTOKEN, NMTOKENS, NOTATION, énuméré
  - Les plus importants: CDATA, NMTOKEN, énuméré
    - Les seuls types vus dans ce cours

Type CDATA

- Aucune restriction sur la valeur de l'attribut
- Veut dire que n'importe quelle chaîne de caractères est permise comme valeur de l'attribut
  - Sauf bien sûr que les caractères < et & doivent être représentés par `<` et `&`

Type NMTOKEN (1/2)

- Mêmes règles que pour un *nom XML…*
- mais *peut commencer par un chiffre*
- Convient pour des valeurs numériques, des dates, des noms ou identifiants courts, etc.

Type NMTOKEN (2/2)

- Exemples de valeurs acceptées (valides) :
  - "lundi", "Mars", "janvier"
  - "25", "3.5", "2028-01-03", "11:45", "05:00AM"
- Exemples de valeurs invalides :
  - "12¤mars", "05:00¤AM", "2,50", "2.50$"
- Les blancs (espaces, sauts de ligne) avant et/ou après le NMTOKEN sont tolérés :
  - "¤mars¤", "¤¤05:00AM¤"

Type énuméré (1/2)

- Les valeurs acceptées sont énumérées dans la déclaration:
  - séparées par "|"
  - entourées par des parenthèses ( )
  - *sans guillemets !*
- *Chaque valeur doit respecter le critère NMTOKEN*

Type énuméré (2/2)

- Exemples corrects:

  (homme | femme)

  (rouge | vert | jaune | autre)

  (1 | 2 | 3)

- Exemples incorrects:

  ("a" | "b" | "c")

  (beau¤temps | mauvais¤temps)

oblig-ou-facult (1/3)

- C'est la partie de la déclaration qui indique si l'attribut est *obligatoire* ou *facultatif* (obligatoriété):

#REQUIRED	signifie obligatoire
#IMPLIED	signifie facultatif

oblig-ou-facult (2/3)

- Si l'attribut est obligatoire, cela veut dire que *toute occurrence* de l'élément pour lequel l'attribut est déclaré *doit* comporter une spécification pour cet attribut
- Par exemple, si on déclare :

  <!ATTLIST élem attrib CDATA #REQUIRED>

  alors toute occurrence de l'élément "élem" *doit* comporter une spécification d'attribut "attrib"…

oblig-ou-facult (3/3)

<élem attrib="valeur">…</élem>

Sans cette spécification d'attribut, l'élément serait invalide :

<élem>…</élem> INVALIDE

- Si l'attribut est facultatif, l'attribut peut être spécifié, mais ce n'est pas obligatoire
  - L'autrice déterminera s'il est pertinent de spécifier l'attribut ou non

Plusieurs attributs pour le même type d'élément (1/2)

- Pour déclarer plus d'un attribut pour le même type d'élément, il suffit d'écrire une déclaration ATTLIST pour chaque attribut…

Il est aussi possible de combiner les déclarations, mais cela comporte plus de désavantages que d'avantages et n'est pas couvert dans notre premier tour d'horizon

Plusieurs attributs pour le même type d'élément (2/2)

Exemples :

```
<!ATTLIST personne âge-en-années NMTOKEN
                  #REQUIRED>
<!ATTLIST personne adresse-courriel CDATA
                  #IMPLIED>

<!ATTLIST mémo statut
   ( brouillon | final | diffusé ) #REQUIRED>
<!ATTLIST mémo pj CDATA #IMPLIED>
```


Non couvert dans ce cours

- En plus de #IMPLIED et #REQUIRED, il est possible de définir une *valeur par défaut* pour un attribut (qui s'applique lorsque l'attribut n'est pas spécifié sur un élément)
- On peut même définir un attribut avec une valeur *fixe*
  - Utile surtout avec les *espaces de noms*

Exemples

- Dossier 050-Ex-utilite-validite
  - aide-aux-autrices
- Exemple dans *Le langage des DTD*

StudiUM

