

XML & DTD

...les bases

Par Richard Vigneux

Table des matières

- Introduction
- Quelques définitions
- Formalisme de la référence XML
- Documents XML bien formés
- Documents XML valides
- Déclarations d'éléments
- Déclarations d'attributs
- Déclarations de types de données abstraits

2

Introduction

■ Avons-nous besoin de devenir des spécialistes du XML et des DTD dans la cadre du cours?

- ✓ Bien sûr que non. Si c'était le cas, nous devrions apprendre:
 - ✓ Les schémas XML qui sont beaucoup plus performant pour définir la grammaire d'un document XML;
 - ✓ XSL pour faire la mise en forme de document XML;
 - ✓ XSLT pour transformer un document XML vers un autre format.

■ Pourquoi alors apprenons-nous le XML et les DTD?

- ◆ Pour bien comprendre le XHTML.
- ◆ Pour pouvoir créer des documents XML valide qui pourront être utilisés pour transmettre des données complexes entre un client et un serveur Web.

3

Quelques définitions

- XML... Qu'est-ce que c'est?
- Un DTD... Qu'est-ce que c'est?
- Un Schéma... Qu'est-ce que c'est?
- CSS... Qu'est-ce que c'est?
- XSL... Qu'est-ce que c'est?
- Quel est la différence entre XML 1.0 et XML 1.1?
- Qu'est ce qu'un document bien formé?
- Qu'est ce qu'un document valide?

4

Formalisme de la référence XML

- Pour cette partie, nous allons utiliser la référence traduite qui se trouve à:

❖ <http://www.yoyodesign.org/doc/w3c/xml11/#sec-well-formed>

- [1] document ::= (prologue élément Divers)* - (Car* CarRestreint Car*)
- [2] Car ::= [#x1-#xD7FF] | [#xE000-#xFFFD] | [#x10000-#x10FFFF]
- [3] S ::= (#x20 | #x9 | #xD | #xA)+
- [10] ValeurAtt ::= ' ' ' ' ([^&"] | Appel)* ' ' ' ' ([^&'] | Appel)* ' ' ' '
- [15] Commentaires ::= '<!--' ((Car - '-') | ('-' (Car - '-')))* '-->'

5

Documents XML bien formés

- Pour être bien formé, un document XML doit respecter les règles suivantes:

- ❖ Le document doit avoir une déclaration XML dans son en-tête pour indiquer le type de document (prologue).
- ❖ Toute balise ouverte doit être fermée
- ❖ Les balises ne peuvent se chevaucher : <p>......</p>...
- ❖ Les caractères utilisés pour nommer les balises doit faire partie des caractères UNICODE valides et ne pas contenir de caractères qui font parti de la syntaxe XML (<>,&, etc.). De plus il ne peut pas débiter par un signe de ponctuation ou XML. Enfin, il ne peuvent pas contenir d'espace.
- ❖ La balise de premier niveau (la racine) doit apparaître une et une seule fois et englober l'ensemble des éléments du document (ça exclu l'en-tête et les commentaires)
- ❖ Contrairement à XHTML :
 - ❖ Les noms des balises peuvent utiliser les majuscules et les minuscules.
 - ❖ La fermeture d'une balise vide ne doit pas être précédée d'une espace.
 - ❖ XHTML →

 - ❖ XML →

6

Documents XML bien formés

■ Exercice 1:

- ◇ Ouvrez Dreamweaver et créez un nouveau document XML.
- ◇ Enregistrez le document sous : doc_bien_forme_1.xml
- ◇ En créant un fichier XML avec Dreamweaver, ce dernier a inséré une ligne dans le code. Que représente cette ligne?
 - ◆ Le prologue composé de 3 parties.
 - ◆ Quelle partie du prologue, Dreamweaver a omis?
 - ◆ L'instruction « standalone ».
- ◆ Ajoutez un commentaire sous le prologue.
- ◆ Ajoutez une balise vide nommée « racine ».
- ◆ Validez: <http://www.validome.org/xml/>

7

Documents XML bien formés

■ Exercice 1 (CORRIGÉ):

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<!--
/*****
/*
/* Fichier.....: doc_bien-forme_1.xml
/* Type.....: Document XML
/* Titre.....: Premier exemple d'un document bien formé minimum.
/* Auteurs.....: ©2009 Richard Vigneux, Cégep de Sherbrooke
/* Date de création.....: 2009-01-23
/* Date de mise en production.: 2009-01-23
/*
/*****
/*
/* - Ce document représente un document bien formé minimum.
/*
/* En entrée: RIEN
/*
/* En sortie: Document XML (n'est pas fait pour l'affichage).
/*
/*****
-->
<racine/>
```

8

Documents XML bien formés

- Exercice 2:
- Cette fois-ci nous allons créer une structure qui peut contenir un document.
 - Enregistrez le document précédent sous : doc_bien-forme_2.xml
 - Modifiez la balise « racine » pour obtenir une balise ouvrante et une fermante.
 - Notre nouveau document comprend
 - Un titre
 - Une introduction
 - Un développement qui est lui-même composé de 2 paragraphes
 - Et une conclusion
 - Créer un texte comportant ces éléments entre les balises « racine », et ajoutez les balises nécessaires :
 - Validez: <http://www.validome.org/xml/>

■ Exercice 2 (CORRIGÉ):

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<!--
/*****
/*
/* Fichier.....: doc_bien-forme_2.xml
/* Type.....: Document XML
/* Titre.....: Exemple d'un document bien formé.
/* Auteurs.....: ©2009 Richard Vigneux, Cégep de Sherbrooke
/* Date de création.....: 2009-01-23
/* Date de mise en production.: 2009-01-23
/*
/*
/*****
/*
/* - Ce document XML est un document bien formé.
/*
/* En entrée: RIEN
/*
/* En sortie: Document XML (n'est pas fait pour l'affichage).
/*
/*****
/*
/* - Ce document est une modification du document "doc_bien_forme_1.dtd". Les modifications sont:
/* - <racine>: la balise n'est plus vide et ajout des balises suivantes:
/* - <titre>
/* - <introduction>
/* - <developpement>: Contient deux paragraphes.
/* - <paragraphe>: sous-balise de "developpement".
/* - <conclusion>
/*
/*****
-->
<racine>
  <titre>Document XML bien formé</titre>
  <introduction>Ce document est un exemple de document XML bien formé.</introduction>
  <developpement>
    <paragraphe>Pour qu'un document XML soit considéré comme bien formé il doit obligatoirement respecter les règles XML.</paragraphe>
    <paragraphe>De plus, il doit contenir minimalement un prologue indiquant le type de document et une balise racine</paragraphe>
  </developpement>
  <conclusion>Comme vous pouvez vous en rendre compte, ce n'est pas très compliqué.</conclusion>
</racine>
```

Documents XML valides

- Pour créer un document valide, il doit être bien formé et avoir un DTD qu'il respecte.
 - ✓ DTD → Définition du Type de Document
- Un DTD est une définition grammaticale du document. Donc ça décrit les règles de création du document XML.
- Le DTD est lié au document XML par une déclaration DOCTYPE qui s'insère au début du fichier XML.
- Il existe 3 façons de faire la déclaration DOCTYPE.
 - La déclaration **DOCTYPE interne**: Permet de définir directement le DTD dans le document XML.
 - La déclaration **DOCTYPE externe**: Le DTD est lié au document et se trouve normalement sur le même serveur Web
 - La déclaration **DOCTYPE externe public**: Le DTD est lié au document, mais il doit utiliser un FPI (identificateur public enregistré auprès de l'organisme ISO)

11

Documents XML valides

- DTD interne
- ✓ La déclaration DOCTYPE pour les DTD interne est composée de 3 parties.
 - L'instruction de déclaration
 - `<!DOCTYPE ... >`
 - L'attribut d'identification : Correspond au nom de la balise racine du document XML. Ce nom doit respecter les règles de nom XML (CarNom). Ce nom n'est pas entre guillemet.
 - NomDeLaBaliseRacine
 - Symboles d'encadrement des définitions DTD
 - `[]`
 - Ce qui donne :
 - `<!DOCTYPE NomDeLaBaliseRacine`
`[`
`<!-- Définition DTD -->`
`]`
`>`

12

Documents XML valides

■ DTD interne (exercice 1)

- ◇ Enregistrez le document « doc_bien_forme_1.xml » sous « doc_valide_1.xml »
- ✓ Ajoutez la déclaration DOCTYPE juste au dessus de la balise racine
 - ◇

```
<!DOCTYPE racine
[
  <!-- Définition DTD -->
]
```

13

Déclarations d'éléments

■ DTD interne (suite)

- ✓ Maintenant, il faut savoir comment déclarer notre unique balise.
- ◇ Une balise se nomme « entité » dans le DTD.
- ✓ La déclaration de type d'élément se compose de trois parties : <http://www.yoyodesign.org/doc/w3c/xml11/#elemdecls>
- ✓ L'instruction de déclaration
 - ◇

```
<ELEMENT ... >
```
- ✓ L'attribut d'identification : Correspond au nom de la balise. Ce nom doit respecter les règles de nom XML (CarNom). Ce nom n'est pas entre guillemet.
 - ◇ NomDeLaBalise
- ✓ L'attribut de déclaration du type d'élément qui peut prendre 5 types de valeur.
 - ◇ EMPTY → pour indiquer une balise vide
 - ◇ ANY → pour indiquer que cette balise peut contenir n'importe quelle balise.
 - ◇ (#PCDATA) → Des chaînes de caractères quelconques (UNICODE). Doit être entre parenthèse.
 - ◇ (enfant) → Une balise qui est défini comme un enfant de ce dernier. Doit être entre parenthèse.
 - ◇ Une combinaison des 2 derniers (Les 2 premier n'auraient pas de sens).

14

Déclarations d'éléments

- DTD interne (exercice 1: suite)
 - Maintenant qu'on sait comment déclarer une balise, remplacez le commentaire dans la déclaration DOCTYPE de notre document par une déclaration pour notre balise « racine ».
 - Ce qui donne :
 - ```
<!DOCTYPE racine
[
 <!ELEMENT racine EMPTY>
]
>
```

15

## Déclarations d'éléments

- Autres notions sur les types de données
  - ✓ Type mixte :
    - Les types mixte peuvent se composer de 2 façons.
      - Liste séparé par des virgules « , »
      - Choix séparé par des barres verticales « | »
        - C'est un OU EXCLUSIF.
  - ✓ Multiplicateurs
    - Un multiplicateur permet d'indiquer le nombre de fois qu'un contenu peut apparaître dans une occurrence de l'entité. Le multiplicateur s'applique à élément qu'il précède. Pour appliquer un multiplicateur à un groupe de contenu, il suffit de mettre le groupe entre parenthèses. Bien entendu, un multiplicateur est inutile lorsqu'on veut qu'un contenu apparaisse obligatoirement qu'une fois.

| Multiplicateur | Description                                     |
|----------------|-------------------------------------------------|
| ?              | { 0 ,1 } (zéro fois OU une seule fois)          |
| +              | { 1 , N } (au moins une fois OU plusieurs fois) |
| *              | { 0 ,N } (zéro fois OU plusieurs fois)          |

16



## Déclarations d'éléments

### ■ DTD interne (exercice 2)

- ◇ Enregistrez le document « doc\_bien\_forme\_2.xml » sous « doc\_valide\_2.xml »
- ◇ Ajoutez la déclaration DOCTYPE juste au dessus de la balise racine.
- ◇ Effectuez la déclaration des types d'éléments pour les différentes balises.
- ◇ Réponse:
  - ◇ <!DOCTYPE racine
  - [
  - <!ELEMENT racine (titre,introduction,developpement,conclusion)>
  - <!ELEMENT titre (#PCDATA)>
  - <!ELEMENT introduction (#PCDATA)>
  - <!ELEMENT developpement (paragraphe+)>
  - <!ELEMENT conclusion (#PCDATA)>
  - <!ELEMENT paragraphe (#PCDATA)>
  - ]
  - >

17

## DTD externe

### ■ Exercice 3

- ◇ Nous allons maintenant transformer notre DTD interne en DTD externe.
- ◇ Pour ce faire, on doit modifier la déclaration DOCTYPE.
- ◇ Enregistrez le document « doc\_valide\_2.xml » sous « doc\_valide\_3.xml »
- ◇ Créer un nouveau document XML et enregistrez le sous « doc\_valide\_3.dtd »
  - ◇ Effacez le contenu de ce nouveau fichier et copiez-y le contenu entre crochets « [ ] » du DOCTYPE du fichier « doc\_valide\_3.xml » (sans les crochets)
- ◇ Retirez du fichier « doc\_valide\_3.xml » le contenu que vous venez de copier ainsi que les deux crochets. Ce qui laisse:
  - ◇ <!DOCTYPE racine
  - >
- ◇ La déclaration d'un DOCTYPE externe implique l'indication de l'URL du DTD. Cette forme de déclaration contient 3 parties pour un DTD privé.
  - ◇ L'instruction de déclaration
    - ◇ <!DOCTYPE ... >

18

## DTD externe

### ■ Exercice 3 (suite)

- ◊ L'attribut d'identification : Correspond au nom de la balise racine du document XML. Ce nom doit respecter les règles de nom XML (CarNom). Ce nom n'est pas entre guillemet.
      - ◊ NomDeLaBaliseRacine
    - ◊ Un mot réservé pour indiquer qu'on utilise un système privé de DTD :
      - ◊ SYSTEM
    - ◊ L'URI de notre fichier DTD entre guillemet
      - ◊ "URI\_du\_DTD"
    - ◊ Ce qui donne :
      - ◊ `<!DOCTYPE NomDeLaBaliseRacine SYSTEM "URI">`
    - ◊ Dans notre cas :
      - ◊ `<!DOCTYPE racine SYSTEM "doc_valide_3.dtd">`

19

## Déclarations d'éléments

### ■ Exercice 4

- ◊ Enregistrez le document « doc\_valide\_3.xml » sous « doc\_valide\_4.xml »
- ✓ Enregistrez le document « doc\_valide\_3.dtd » sous « doc\_valide\_4.dtd »
- ◊ Ne pas oublier de modifier l'URL du DOCTYPE dans le fichier XML.
- ◊ Modifiez le DTD pour répondre aux besoins suivants :
  - ✓ J'aimerais modifier mon DTD pour permettre que les balises « introduction » et « conclusion » puissent contenir soit du texte, soit des paragraphes ou être vides.
    - ◊ `<!ELEMENT introduction (#PCDATA|paragraphe)*>`
    - ◊ `<!ELEMENT conclusion (#PCDATA|paragraphe)*>`
  - ✓ Maintenant j'aimerais pouvoir mettre des titres de deuxième niveau (titre\_2) dans les sections « introduction » et « conclusion » lorsque je le désire.
    - ◊ `<!ELEMENT introduction (#PCDATA|titre_2|paragraphe)*>`
    - ◊ `<!ELEMENT conclusion (#PCDATA|titre_2|paragraphe)*>`
    - ◊ `<!ELEMENT titre_2 (#PCDATA)>`
  - ✓ Enfin, j'aimerais pouvoir ajouter au besoin, un et un seul titre de deuxième niveau (titre\_2) dans la section « développement » et ce seulement au dessus du premier paragraphe.
    - ◊ `<!ELEMENT developpement (titre_2?,paragraphe)*>`

20

## Déclarations d'attributs

### ■ Exercice 5

- Enregistrez le document « doc\_valide\_4.xml » sous « doc\_valide\_5.xml »
- Enregistrez le document « doc\_valide\_4.dtd » sous « doc\_valide\_5.dtd »
- Nous avons besoin d'ajouter des attributs à nos nouvelles balises. Comment ça marche?
- La déclaration de la liste des attributs se compose de 3 parties :
  - L'instruction de déclaration
    - `<!ATTLIST ... >`
  - L'attribut identifiant l'élément auquel appartient cette liste d'attribut (donc le nom de la balise). Ce nom n'est pas entre guillemet.
    - NomDeLaBalise
  - Les déclarations des attributs. Cette déclaration se compose elle aussi de trois parties. Notez qu'il n'y a pas de séparateurs particulier pour diviser chaque déclaration, seulement un blanc (espace, saut ligne, etc.)
    - Le nom du nouvel attribut (pas entre guillemet)
    - Le type de l'attribut
    - L'exigence par rapport à l'attribut qui peut comprendre une valeur (cette valeur sera obligatoire pour le type « #FIXED » et jamais pour le type « #REQUIRED »).
- Structure générale
  - `<!ATTLIST nom_élément`
  - `nom_attribut_1 type_attribut_1 exigence`
  - `nom_attribut_2 type_attribut_2 exigence`
  - ...
  - `>`

21

## Déclarations d'attributs

### ■ Exercice 5 (suite)

- Type de l'attribut
- ✓ CDATA
  - (Unparsed) Character Data (le contenu n'est pas analysé par l'analyseur (parser) XML. Chaînes de caractères (incluant les chiffres). Permet de mettre du contenu qui sera seulement interprété comme du texte et jamais comme des codes.
  - ✓ <http://www.yoyodesign.org/doc/w3c/xml11/#sec-cdata-sect>
- ✓ ID
  - Représente un identificateur (clé). Le validateur va vérifier l'unicité de cet identificateur.
- ✓ IDREF
  - Représente une référence sur un identificateur (référence à une clé). Le validateur va vérifier que l'identificateur référencé existe.
  - Représente à mon avis une encre.
- ◆ NMTOKEN
  - Représente un identificateur XML (un mot sans espace) Voir :
    - ✓ <http://www.yoyodesign.org/doc/w3c/xml11/#NT-Name>
    - ✓ <http://www.w3.org/TR/REC-xml/#NT-Nmtoken>

22

## Déclarations d'attributs

### ■ Exercice 5 (suite)

- NMTOKENS
- Représente plusieurs identificateur XML séparés par des espaces (blanc, tabulation, retour de chariot)
- ✓ Liste de choix
- Liste de valeurs que peut prendre l'attribut. Normalement séparé par des OU exclusif.
- Ex : (oui | nom) → l'attribut ne peut avoir qu'un de ces valeurs (« oui » ou « nom ») et rien d'autre.
- Exigence par rapport à l'attribut
- ✓ 'valeur'
- Indique une valeur par défaut pour les listes de choix.
- #REQUIRED
- Indique que l'attribut doit être présent et posséder une valeur valide.
- ✓ #IMPLIED
- Indique que l'attribut est facultatif. On peut ajouter une valeur par défaut « #IMPLIED 'valeur' ».
- #FIXED 'valeur'
- Indique la valeur invariable de l'attribut (constante).

23

## Déclarations d'attributs

### ■ Exercice 5 (suite)

- Liste des ajouts à faire:
  - <paragraphe> : doit avoir un attribut "theme" qui contient le thème abordé par le paragraphe.
  - ✓ <titre\_2> : peut avoir un attribut "id".
  - <introduction, developpement et conclusion>: doivent avoir un attribut "visible" qui peut avoir la valeur "vrai" ou "faux". La valeur par défaut sera "vrai".
  - ✓ <table\_matiere> : Nouvelle balise optionnelle de premier niveau pouvant contenir un titre\_2 suivi de 0-N liens internes.
  - <lien\_interne> : Nouvelle sous-balise de "table\_matiere" doit contenir un attribut référence.

24

# Déclarations de types de données abstraits

- Utilisation des entités paramètres
  - <http://www.yoyodesign.org/doc/w3c/xml11/#sec-entity-decl>
    - Les entités paramètres permettent de créer des espèces de constantes pour centraliser certaine déclaration. Il est très avantageux de les utiliser pour rendre l'entretien de nos DTD plus facile et plus fiable (nommé DéclEP dans la doc).
    - La déclaration se compose de trois parties :
      - L'instruction de déclaration
        - <!ENTITY % ... >
      - L'attribut identifiant l'entité (comme le nom d'une constante). Ce nom n'est pas entre guillemet.
        - NomDeLEntite
      - La valeur de l'entité. Doit être encadré de guillemet.
        - Chaînes de caractères quelconques représentant le contenu qui remplacera l'entité lorsqu'elle sera utilisée.
    - Pour utiliser l'entité, on n'a qu'à utiliser son nom préfixé de signe de pourcentage « % » et suffixé d'un point virgule « ; ».
      - %NomDeLEntite;
    - Note 1 : Les entités doivent être déclarées avant d'être utilisés.
    - Note 2 : Les entités peuvent porter le même nom qu'un élément sans conflit.
    - Note 3 : Dans le DTD de XHTML on utilise des points « . » dans les noms des entités. Ces points n'ont pas plus d'impact que si on avait mis des soulignés « \_ » à la place.
  - Dans notre dernier DTD plusieurs définitions se répètent. Qu'est ce qu'on pourrait transformer en entité?
    - Certains contenus d'éléments :
      - #PCDATA → Est utilisé pour plusieurs contenus d'éléments.
      - (%chaînesCarac; | titre\_2 | paragraphe)\* → Est utilisé comme contenu des éléments « introduction » et « conclusion ».
    - Certains attributs :
      - visible (vrai/faux) 'vrai' → Est utilisé comme attribut pour les éléments : « introduction », « developpement » et « conclusion ».
  - Enregistrez le document « doc\_valide\_5.xml » sous « doc\_valide\_6.xml »
  - Enregistrez le document « doc\_valide\_5.dtd » sous « doc\_valide\_6.dtd »
  - Ajoutez les entités suivantes :
    - chaînesCarac → "#PCDATA"
    - contenuVariable → "(%chaînesCarac; | titre\_2 | paragraphe)\*"
    - visible → "visible (vrai/faux) 'vrai'"
  - Remplacer les contenus par les entités dans le DTD.
  - Valider votre document XML.

XML & DTD  
**FIN**  
...les bases  
Par Richard Vigneux