

Fiche XML n°3

Entités générales & entités paramètres

Cours

1. Nécessité de gérer la structure d'une DTD

Construire un document XML, comme construire une DTD, peut devenir une charge assez lourde. Le document devient parfois assez imposant et difficile à gérer. Nous verrons, dans une autre fiche, un outil pour parcourir et effectuer des recherches dans un tel document. Dans cette fiche, nous allons nous intéresser à un mécanisme permettant de mieux gérer la redondance et les modifications aussi bien au niveau du document (avec les entités générales) qu'au niveau de la DTD (avec les entités paramètres).

2. Les entités générales

Une entité générale est particulièrement utile (mais ce n'est pas l'unique utilisation) lorsque le document XML devient long et lourd. Parfois, dans ce document, une même information est répétée plusieurs fois, pas forcément dans des parties rapprochées du document. Dans d'autres situations, une information est susceptible de changer souvent, et il est intéressant de pouvoir y accéder facilement. Aussi, XML propose un mécanisme nommé *entité générale* (interne, externe, analysable ou non).

Définition : une *entité générale* est un couple (nom-XML, valeur). La valeur est une chaîne de caractères explicite ou l'URI d'un document textuel. La déclaration de l'entité se positionne dans la partie DTD du document XML au niveau de l'entête (voir la fiche sur la DTD). Une entité est utilisable à n'importe quel endroit du document XML. Elle est remplacée par sa valeur lorsque le document est utilisé par une application ou lors de recherches. La seule contrainte sur le contenu d'une entité est que, une fois l'entité remplacée par sa valeur là où elle se trouve, le document XML qui en résulte reste bien formé.

La déclaration d'une entité générale prend deux formes selon que sa valeur est une chaîne de caractères ou un fichier :

- `<!ENTITY nom_entite 'valeur' >` : «valeur» est une simple chaîne de caractères ;
- `<!ENTITY nom_entite SYSTEM 'URI' >` ou `<!ENTITY nom_entite PUBLIC 'id-pub' 'URI' >` : cette version se réfère à un document se trouvant à l'emplacement «URI» (selon le même mécanisme que pour la référence à la DTD ; l'entité prend alors comme valeur le contenu du document).

Dans le document, l'utilisation d'une entité se fait par l'utilisation du nom qu'on lui a donné précédé de «&» et suivi directement de «;», ce qui donne : «&nom_entite;». Éventuellement, la valeur d'une entité peut contenir une référence à une entité. Le remplacement de la référence dans la valeur se fait alors avant le remplacement de l'entité.

Par exemple, si dans la DTD interne au document, nous avons «`<!ENTITY ent '<nom>Yi</nom>' >`», le fragment de document XML «`<personne>&ent;</personne>`» donnera, après remplacement de l'entité, «`<personne><nom>Yi</nom></personne>`». Ce remplacement est correct, car le résultat est bien formé. Par contre, «`<personne &ent;></personne>`» n'est pas correct, car il donnera un document qui n'est pas bien formé «`<personne <nom>Yi</nom>></personne>`».

Dans l'exemple ci dessous, une entité «host» est définie dans le premier document. Ensuite, l'entité «&host;» est utilisée à la place de la valeur «192.68.0.1». Ainsi, si cette valeur est amenée à être modifiée, il ne sera fait qu'une seule modification. Dans le second document, totalement équivalent, le remplacement de l'entité par sa valeur a été fait.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE racine [
<!ENTITY host '192.68.0.1'>
]>
<racine>Le serveur &host; se trouve dans la salle s&#xE9;curis&#xE9;e. &host;
g&#xE8;re le service de courrier &#xE9;lectronique et le serveur web
(Apache)...
</racine>
```

```
<?xml version='1.0' encoding='UTF-8' ?>

<racine>Le serveur 192.68.0.1 se trouve dans la salle s&#xE9;curis&#xE9;e.
192.68.0.1 g&#xE8;re le service de courrier &#xE9;lectronique et le serveur
web (Apache)...
</racine>
```

Dans certains cas, la valeur associée à une entité n'est pas une chaîne de caractère analysable du point de vue XML. C'est le cas si la ressource référencée est un image par exemple. Nous n'aborderons pas plus les entités générales non-analysables.

XML propose 5 entités pré-définies. Elles concernent des caractères donc l'utilisation est contrôlée. Elles ont été définies afin de permettre l'utilisation de ces caractères sans qu'ils puissent être confondus avec les caractères spéciaux utilisés par les documents XML pour le balisage. Ces caractères seront énumérés dans la fiche sur l'encodage.

3. Les entités paramètres

Les entités paramètres sont à la DTD ce que sont les entités générales au document XML. Elles permettent essentiellement de factoriser des «morceaux de description» de DTD. Lors de la définition de cette entité, il faut ajouter un «%» avant le nom de l'entité : «<!ENTITY % *nom_entite* "*valeur*" >». Là encore, l'entité peut être interne ou externe (avec les mêmes variantes de syntaxe). Cette dernière solution permet alors de composer des DTD à partir de DTD existantes. Une entité paramètre s'utilise comme une entité générale, en remplaçant le «&» par un «%».

Comme premier exemple, nous utilisons les entités paramètres comme moyen de générer des parties de déclaration communes à plusieurs objets¹. Ici, tout ouvrage contient un titre, un nombre de page et des auteurs. Par contre, un livre est composé de chapitres alors qu'un article est structuré en sections. Aussi, nous obtenons la DTD ci-contre.

```
<!ENTITY % commun "titre, nbpage, auteur+" >

<!ELEMENT livre (%commun;, chapitre+) >
<!ELEMENT rapport (%commun;, chapitre+) >
<!ELEMENT article (%commun;, section+)>
```

Un autre exemple d'utilisation de ces entités est la composition de DTD (construire une DTD à partir de DTD existantes) ou la division d'une DTD pour obtenir des documents pas trop long. Dans l'exemple suivant, un élément est décrit et d'autres sont ajoutés venant de deux DTD.

```
<!ELEMENT guide (titre, auteur+, editeur, annee, vallon+)>

<!ENTITY % general SYSTEM "rando_a.dtd">
%general;

<!ENTITY % itineraire SYSTEM "rando_b.dtd">
%itineraire;
```

¹ En effet, le mécanisme d'héritage ou de surcharge n'existe pas dans ce langage de schéma. Il convient donc d'utiliser d'autres techniques.

4. Structurer une DTD par spécialisation

Cette dernière manière de construire une DTD en incluant d'autres DTD est intéressante, car elle permet de spécialiser une DTD standard. Cette méthode peut s'utiliser dans un document XML en référant une DTD externe et en «ajustant ce schéma» au niveau de la DTD interne. Ce qui est défini dans cette dernière est prioritaire sur ce qui est défini dans la DTD externe. De plus, seule la première définition rencontrée est prise en compte. Mais attention, toutes les manipulations ne sont pas autorisées. Il est possible d'ajouter des attributs à un élément, de modifier des attributs et de redéfinir les entités (générales ou paramètres). Par contre, il n'est pas possible de modifier la description du contenu d'un élément.

Comme exemple, prenons le schéma suivant «crous.dtd» :

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT campus (batiment*) >
<!ELEMENT batiment (#PCDATA)>
<!ATTLIST batiment numero CDATA #IMPLIED >
<!ENTITY ru "Restaurant universitaire">
```

Alors, le document ci-dessous est valide. L'entité «ru» est redéfinie. L'attribut «type» est ajouté à l'élément «batiment» et l'attribut «numero» est redéfini.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE campus SYSTEM "crous.dtd"
[
  <!ENTITY ru "Restaurant Universitaire Lombarderie">
  <!ATTLIST batiment type CDATA #IMPLIED >
  <!ATTLIST batiment numero (1 | 2 | 3 | 4 | 5) #IMPLIED>
]>

<campus>
  <batiment numero="1" type="étude">bibliothèque</batiment>
  <batiment numero="5">administration</batiment>
  <batiment type="vie étudiante">&ru;</batiment>
</campus>
```

Par contre, l'exemple suivant n'est pas correct, car il tente de re-définir le contenu de l'élément «batiment».

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE campus SYSTEM "crous.dtd"
[
  <!ELEMENT batiment (nom, type)>
  <!ELEMENT nom (#PCDATA)>
  <!ELEMENT type (#PCDATA)>
]>

<campus>
  <batiment numero="1"><nom>bibliothèque</nom><type>étude</type></batiment>
  <batiment numero="10"><nom>administration</nom><type>gestion</type>
  </batiment>
  <batiment numero="4"><nom>&ru;</nom><type>vie étudiante</type></batiment>
</campus>
```

Incorrect

5. Règles d'utilisation des entités

A ce niveau, il est important de revenir sur la définition de document valide. La présence d'entités impose évidemment de nouvelles contraintes :

- toute entité doit être déclarée ;
- une entité paramètre doit être déclarée avant toute référence à celle-ci ;

- la référence à une entité paramètre ne peut se faire que dans une DTD ;
- la référence à une entité paramètre dans une déclaration (élément, attribut ou entité) se fait uniquement dans une DTD externe ;
- les références à des entités générales se font dans un ordre indifférent ;
- les définitions récursives² et circulaires³ sont interdites.

6. Règles d'écriture des DTD

Proposer une DTD adaptée au problème à modéliser n'est pas toujours aisé. Il convient donc de suivre certaines règles :

- ne pas hésiter à abuser de la modularité : les entités externes servent pour définir les parties réutilisables dans des documents séparés ;
- les entités paramètres (DTD externes) sont utilisées pour modéliser le contenu partagé par plusieurs éléments (redéfinition possible dans une DTD interne)
- les entités paramètres (DTD externes) sont utilisées pour définir des attributs communs à plusieurs éléments (redéfinition possible dans une DTD interne)
- il faut regrouper en tête les déclarations d'entités paramètres ;
- il faut concevoir des modèles de contenus simples.

En tout état de cause, il convient de bien documenter les DTD en utilisant les commentaires.

Exercices

Exercice 1. Erreurs (suite)

Le document ci-dessous est-il un document XML bien formé ? Si ce n'est pas le cas, indiquer les solutions possibles pour le corriger.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE racine [
  <!ENTITY ip-c '192.68.0'>
  <!ENTITY ip-ext '81.68.899.133'>
]>
<Serveur>
  Serveur central :
    loc : &ip-c1.XXX
    ext : &ip_ext;
    URL : http://$ip-c;.1:8080
    ou http://$ip-ext;:8080
</serveur>
<2clients>
  Poste client 1 : &ip-c;2
  Poste client 2 : ip-c;3
</2clients>
```

² Une entité s'utilise, par exemple «<!ENTITY en 'Attention a &en;';>»

³ Une entité contient une entité qui, elle-même, contient la première.

Exercice 2. Formules mathématiques (suite)

Soit la DTD suivante (Cf. exercice 1 de la fiche n°2 «Description d'un document XML») :

```
<!-- DTD pour décrire un texte contenant des formules mathématiques -->
<ELEMENT textemath (texte | formule)+ >
<ELEMENT texte (#PCDATA) >
<ELEMENT formule (valeur|somme|différence|produit|fraction|racine|puissance) >
<ELEMENT fraction (op1, op2)>
<ELEMENT valeur (#PCDATA)>
<ELEMENT somme (op1, op2)>
<ELEMENT diff (op1, op2)>
<ELEMENT produit (op1, op2)>
<ELEMENT racine (op1)>
<ATTLIST racine ordre CDATA #IMPLIED>
<ELEMENT puissance (op1)>
<ATTLIST puissance exposant CDATA #REQUIRED>
<ELEMENT op1 (valeur|formule)>
<ELEMENT op2 (valeur|formule)>
```

Question. Proposer une modification de la DTD afin de simplifier la structure du document. Appliquer ces modifications sur la formule de la question précédente.

Exercice 3. Agenda partagé

Pour faciliter le travail à distance d'une équipe, on cherche à proposer un agenda, partagé sur le Web, pour gérer les rendez-vous et les réunions. Les données de cet agenda, destinées aux membres de l'équipe (les utilisateurs), sont mémorisées dans un document XML.

Un agenda permet de mémoriser la position des rendez-vous (ou des réunions) sur un calendrier ainsi que leurs participants. Chaque rendez-vous, créé par l'un des utilisateurs (le propriétaire du rendez-vous), peut être public, c'est à dire destiné à tous les utilisateurs, réservé à une liste précise d'utilisateurs, ou encore privé, c'est à dire dont les participants ne sont pas précisés. Par défaut un rendez-vous est public.

Les utilisateurs sont identifiés par un pseudonyme (unique). Il existe un utilisateur spécial dont le pseudonyme est « anonyme », qui désigne les destinataires d'un rendez-vous public.

Un agenda est identifié par un nom d'agenda et caractérisé par une date de création et une date de dernière modification. Il est composé de semaines de sept jours. Chaque semaine commence le lundi. Chaque jour comporte un ensemble de rendez-vous (ou réunions).

Un rendez-vous est défini par un numéro d'ordre absolu, son propriétaire, un créneau horaire, un titre, un indicateur d'importance (de 1 à 4 qui vaut 1 par défaut) et un commentaire facultatif. Les rendez-vous ne peuvent pas être à cheval sur deux jours.

Les rendez-vous sont pris sur plusieurs années. Certains sont répétitifs. Dans ce cas, il faut préciser le type de répétition (par jour, par semaine, par mois, par année), la date du début et, éventuellement, la date de la fin de la période durant laquelle le rendez-vous se répète.

Question. Écrire une DTD permettant de décrire la structure d'un tel agenda. Optimiser cette DTD pour effectuer un maximum de contrôles et pour pouvoir faciliter la présentation des informations sous une forme habituelle. Indiquer les contraintes qui ne peuvent pas être contrôlées en donnant les raisons.

Remarque : des informations ont peut-être été omises dans le descriptif (parce qu'implicites). Elles doivent être explicitées dans la DTD. Il convient d'effectuer le moins possible de répétitions dans la DTD.

Exercice 4. Répertoire (suite)

Soit la DTD "repertoire.dtd" ci-contre qui décrit des documents permettant de mémoriser des répertoires d'adresses (Cf. exercice 2 de la fiche n°2 «Description d'un document XML»).

Question 1. La DTD "repertoire.dtd" peut être légèrement simplifiée (raccourcie). Donner cette simplification.

Question 2. On cherche à proposer un répertoire partagé par une équipe avec des contraintes similaires à celles imposées à l'agenda partagé. Donner une DTD pour un tel répertoire. Vous devrez utiliser "repertoire.dtd" sans le modifier ni l'intégrer directement.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ELEMENT repertoire (catégorie)*>
<ELEMENT catégorie (contact)* >
<!ATTLIST catégorie
  nom ID #REQUIRED>
<ELEMENT contact (bureau, maison?) >
<!ATTLIST contact
  nom ID #REQUIRED
  prénom CDATA #REQUIRED >
<ELEMENT bureau EMPTY >
<!ATTLIST bureau
  téléphone CDATA #REQUIRED
  adresse CDATA #REQUIRED
  courriel CDATA #IMPLIED>
<ELEMENT maison EMPTY >
<!ATTLIST maison
  téléphone CDATA #REQUIRED
  adresse CDATA #REQUIRED
  courriel CDATA #IMPLIED>
```

Exercice 5. Programme télé (suite)

Le responsable d'un site Web spécialisé se propose de présenter les programmes de télévision par semaines. Pour cela, il envisage de mémoriser les informations dans un document XML (Cf. exercice 3 de la fiche n°2 «Description d'un document XML»).

Ci-contre, une solution de DTD est proposée.

Question. Proposer, en la justifiant, une DTD permettant de représenter ces données en corrigeant tout ou partie des problèmes de la DTD ci-contre. Indiquer les problèmes résolus et ceux qui ne le sont pas.

```
<?xml version="1.0" encoding="UTF-8"?>
<ELEMENT programmation (semaine)* >
<!ATTLIST programmation annee CDATA #REQUIRED>

<ELEMENT semaine (journee)* >
<!ATTLIST semaine no_semaine CDATA #REQUIRED>

<ELEMENT journee (programme)* >
<!ATTLIST journee
  nom (lundi|mardi|mercredi|jeudi|vendredi|samedi|dimanche) #REQUIRED>

<ELEMENT programme (creneau-horraire, description, commentaire?) >
<!ATTLIST programme
  titre CDATA #REQUIRED
  chaine CDATA #REQUIRED
  type CDATA "fiction"
  avis-redaction (deconseille|sans-avis|correct|bon|excellent) 'sans-avis'
  accord-parental (tout-public|moins10|moins12|moins16|moins18) 'tout-public'>

<ELEMENT description (#PCDATA) >
<ELEMENT commentaire (#PCDATA) >

<ELEMENT creneau-horraire EMPTY>
<!ATTLIST creneau-horraire
  debut CDATA #REQUIRED
  fin CDATA #IMPLIED
  duree CDATA #IMPLIED>
```



Cette création est mise à disposition selon le Contrat Paternité-Pas d'Utilisation Commerciale-Pas de Modification 2.0 France disponible en ligne <http://creativecommons.org/licenses/by-nc-nd/2.0/fr/> ou par courrier postal à Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.