La description de données avec XML

Julien Lemattre Axel Veber

julien.lemattre@etud.univ-montp2.fr

axel.veber@etud.univ-monpt2.fr

17 décembre 2014

Table des matières

Table des matières

Histoire de XML

Origine du XML

- Problêmatique : communication par Internet avec des fichiers.
- Chaque programmes avaient leurs façons de gérer les fichiers.
- SGML (**S**tandard **G**eneralized **M**arkup **L**anguage) a été crée pour standardiser la gestion de fichier.
- Cependant, SGML était trop complexe pour permettre la communication entre machines.

Histoire de XML

Apparition de XML 1.0 et 1.1

- XML (e**X**tensible **M**arkup **L**anguage)est né du compromis entre SGML et la communication sur le Web.
- XML 1.0 est devenu une recommandation du W3C le 10 février 1998.
- XML 1.1 a été publié le 4 février 2004 en ajoutant le support de caractères Unicode 2.0.
- Cependant, XML 1.0 reste la version la plus utilisée.

Table des matières

- XML facilite l'échange entre les machines sur Internet entre programmes car c'est un langage standardisé
- C'est un langage qui est comprit autant par les machines que pour l'être humain car il est lisible et permet de donner des noms aux données

Remarque

XML n'est pas un langage de programmation : ce langage décrit des données sans les exploiter.

Les données décrites par un fichier XML sont organisées sous forme d'arborescence

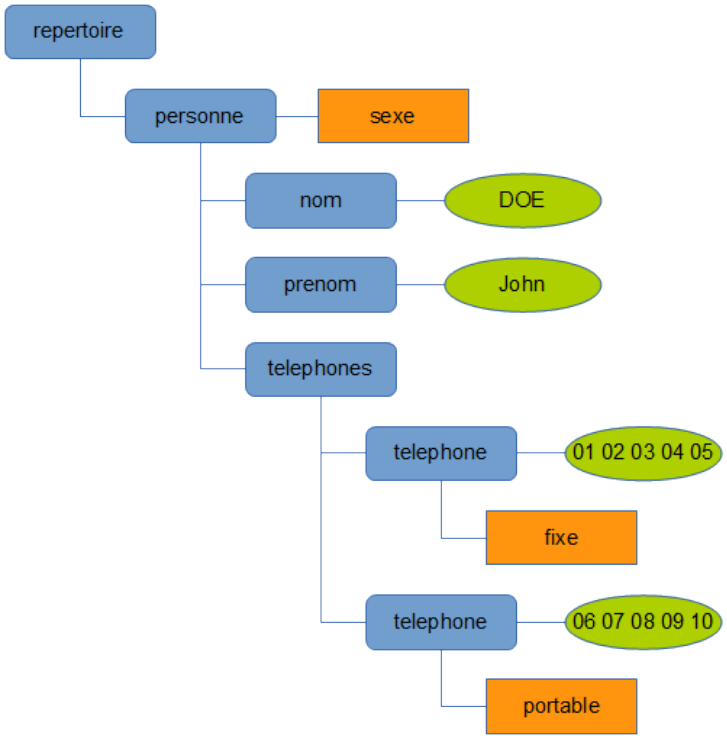

Figure: Structure d'une personne dans un répertoire

Table des matières

Deux types de balises : simples et doubles.

Exemple

<balise>Contenu de la balise</balise> : exemple de balise double
<balise/> : exemple de balise simple

Attention

Les balises ouvrantes et fermantes doivent avoir exactement le même nom.

- Une balise simple est une balise double qui n'a pas de contenu. C'est un raccourci.
- Une balise double peut tout contenir : chaîne, entiers, autres balises...
- Attention cependant à ne pas croiser les balises, mais à les encapsuler.

La généricité du XML vient de la possibilité de créer son propre langage balisé. Cependant, des conventions de nommage des balises doivent être respectées :

- Les noms peuvent contenir chiffres, lettres et carctères spéciaux.
- Mais ils ne peuvent pas débuter par un chiffre ou un caractère de ponctuation.
- Ils ne peuvent pas non plus commencer par la chaîne "XML".
- Ils ne contiennent ni espaces, ni les caractères suivants : - , ; , <et >.

Deux parties distinctes : le prologue et le corps.

- Le prologue :

Exemple de prologue

```
<?xml version = "1.0" encoding="UTF-8" standalone="yes" ?>
```

- Le corps :

Exemple de corps

```
<racine>
<balise_paire>texte</balise_paire>
<balise_paire2>texte</balise_paire2>
<balise_paire>texte</balise_paire>
</racine>
```

- 3 informations dans le prologue :
  - La version de XML
  - L'endodage utilisé
  - L'autosuffisance du document
- Dans le corps, il doit y avoir une paire de balises qui contient toutes les autres.
  On l'appelle racine.

On peut ajouter des informations supplémentaires aux balises : les attributs. Le nom est au choix.

Exemple d'attribut

```
<prix devise="euro">25.3</prix>
```

L'information principale est le montant du prix, mais la devise est une information supplémentaire qui caractérise le prix.

Syntaxe du XML

Attributs

Il peut y avoir ou non plusieurs attributs.
Quelques règles :

- Même règle de nommage que pour les balises
- La valeur de l'attribut doit être entre guillemets
- Pas de doublons à l'intéreur d'une balise.

Quqlques règles définissent si un document est "bien formé", ou "well-formed" :

- Si XML 1.1, le prologue doit être complet
- Une seule racine
- Noms des balises et attributs conformes
- Valeurs des attributs entre guillemets
- Aucun chevauchement de balises

Si ces règles ne sont pas respectées, le document est inutilisable.

Table des matières

- Maintenant qu'un document est dit "bien formé", il faut l'exploiter.
- Afin d'être régulier dans l'exploitation d'un fichier XML, il faut qu'il soit **valide**.
- Un document **valide** obéit à une définition des données qu'il décrit.
- Ainsi, plusieurs concepteurs peuvent se mettre d'accord sur une grammaire commune.
- Pour décrire un document, plusieurs formats sont possibles. Ici, on parlera de DTD (Document Type Definition)
- Cette définition pourra être écrit dans le document XML, où dans un autre fichier (ici, on parlera de fichier .dtd).

On définit le "squelette" de notre arborescence en définissant les balises avec cette syntaxe :

```
<!ELEMENT balise (contenu)>
```

- On indique la balise qu'on définit.
- On indique ce qu'elle doit contenir.

Le contenu d'une balise peut être défini comme suit :

- #PCDATA : Indique qu'une balise peut contenir une chaîne de caractère, un entier...
- balise : Indique qu'une balise doit **obligatoirement** contenir la balise indiquée.
- balise ? : Indique que la balise **peut** contenir la balise indiquée
- b1, b2 : Indique qu'une balise doit **obligatoirement** contenir les balises indiquées **dans cet ordre**.
- b1 | b2 : Indique que la balise doit **obligatoirement** contenir une des balises indiquées **dans cet ordre**.

balise* Indique que la balise peut être **répetée**, même si elle est **optionnelle**.

balise+ Indique que la balise peut être **répetée**.

Exemple

```
<!ELEMENT repertoire (personne+)>
<!ELEMENT personne (nom, prenom, adresse, telephone?)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT adresse (#PCDATA)>
<!ELEMENT telephone (#PCDATA)>
```

Une fois l'arborescence défini, il faut définir les attributs possibles de chaque balises avec la ligne :

```
<!ATTLIST balise attribut type mode>
```

balise Le nom de la balise dans laquelle on définit l'attribut.

attribut Le nom de l'attribut que la balise peut avoir.

type Le type de valeur que peut prendre l'attribut : du texte non parsé (CDATA), un choix (valeur1 | valeur2) ou un identifiant.

mode Indique si l'attribut doit être renseigné (#REQUIRED), optionnel (#IMPLIED), une constante (#FIXED "valeur") ou une valeur par défaut.

Il est possible de donner à un nœud un attribut qui lui sert d'identifiant unique. Il suffit donc de renseigner le type de l'attribut dans notre définition :

ID L'attribut défini est un identifiant. Le document ne sera pas valide si un autre nœud à le même identifiant.

IDREF L'attribut défini se référe à un identifiant. Si l'identifiant indiqué n'existe pas, le document ne sera pas valide.

Exemple

```
<!ATTLIST father id ID #REQUIRED>
<!ATTLIST child id ID #IMPLIED>
<!ATTLISE child father IDREF #REQUIRED>
```

Table des matières

Exemple d'API
L'API DOM

- Les applications utilisent des parseurs XML afin d'extraire les informations d'un document (noeuds, attributs, contenus, commentaires...).
- Ici, on parlera de DOM (**D**ocument **O**bject **M**odel).
- DOM est un standard W3C depuis 1998.
- Il est implémenté dans divers langages de programmation (C, C++, Java, PHP...). Ici, nous verrons comment utiliser le parseur en Java.
- Il est également possible pour DOM de vérifier si un document est valide selon son fichier .dtd .

Exemple d'API

Extraction de contenu en Java

Afin d'extraire les informations avec le parseur, plusieurs classes sont utilisées afin de représenter les informations du document XML :

Document Se référe au document XML à parser.

Node C'est un élement du document XML. Il peut être un nœud, un attribut ou le contenu d'une balise.

NodeList Une liste de Node.

Element Représente une balise du document XML. La classe Element possède d'autres fonctionnalités, comme récupérer un attribut correspondant.

Attr Représente un attribut d'une balise (nom + valeur).

Text Représente le contenu d'une balise.

Exemple d'API

Extraction de contenu en Java - Méthode

- On commence à créer une instance de DocumentBuilderFactory.
- L'instance de DocumentBuilderFactory permet de créer un DocumentBuilder.
- Le DocumentBuilder permet de créer un Document (à partir de là, on spécifie le document XML qu'on veut parser).
- On extrait l'élement racine dans un Element grâce au Document.
- Selon la structure de l'arborescence du document, on récupère la liste des nœuds fils dans un NodeList.
- Dès qu'une donnée doit être présente (attribut ou contenu texte), on récupère le nœud correspondant dans un Element, qui va nous permettre d'accéder aux données souhaitées.

Table des matières

Synthèse

- XML est un langage de **description de données** standardisé, qui est orienté vers l'échange de données sur Internet.
- Le langage utilise des **balises**, caractérisées par des **attributs**, qui peuvent contenir d'autres balises sous forme d'**arborescence**, ou du **texte**.
- Un document **bien formé** respecte la syntaxe XML, tandis qu'un document **valide** respecte la définition proposée.
- Un document valide permet de mettre d'accord plusieurs concepteurs sur une grammaire commune, et impose une certaine rigueur lors de la création de documents XML.
- On exploite un document XML avec un **parseur XML**, qui parcours l'arborescence du document afin d'en extraire le contenu.