



Concepteur Développeur en Informatique

Développer des composants d'interface

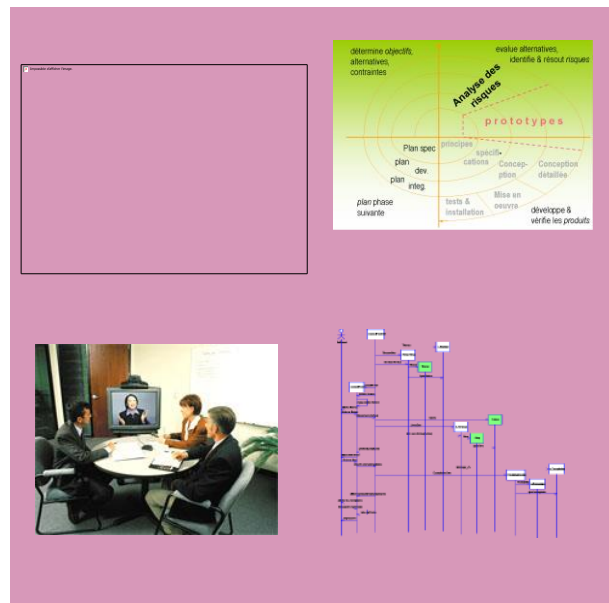
Présentation Langage XML

Accueil

Apprentissage

PAE

Evaluation



Localisation : U03-E03-S02

SOMMAIRE

1. Introduction.....	3
2. Présentation des données	4
3. Validation de la structure d'un document XML	4
3.1. Décodage d'un document XML	4
3.2. Les avantages de XML	5
3.3. Les inconvénients de XML	6
4. Structure d'un document XML.....	6
4.1. Les objets contenus dans un document XML.....	8
4.1.1. Règles de validité d'un document XML	8
4.1.2. Exemples de syntaxe	9

1. Introduction

Ce support n'a pas pour but de présenter l'ensemble des techniques relatives à XML. Il s'agit d'une introduction au métalangage qui présente en particulier la structure d'un document XML et son formalisme.

Le langage XML (eXtensible Markup Language) est dérivé du SGML (Standard Generalized Markup Language), défini par le standard ISO8879 en 1986, utilisé dans le milieu de la Gestion Electronique Documentaire (GED).

XML peut être considéré comme un métalangage permettant de définir d'autres langages, c'est-à-dire définir de nouvelles balises permettant de décrire la présentation d'un texte.

La force de XML réside dans sa capacité à pouvoir décrire n'importe quel domaine de données grâce à son extensibilité. Il va permettre de structurer, poser le vocabulaire et la syntaxe des données qu'il va contenir.

Un document XML peut avoir de nombreuses cibles :

- Téléphones
- Ecrans
- Imprimantes
- Bases de données
- Serveurs d'applications
- Réseaux
- ...

Un des autres avantages de XML dans les échanges entre applications est son format texte qui permet, contrairement aux fichiers dans un format binaire, de s'assurer qu'il n'y aura pas de risques de remise en cause de la sécurité des applications. Un document XML pourra être transporté par le protocole http et diffusé sans problème au travers des différentes couches assurant la sécurité.

XML a aujourd'hui un concurrent sérieux au niveau des services Web : JSON JavaScript On Notation.

On peut trouver de nombreux exemples d'utilisation de XML.

- Sérialisation d'un objet métier ou d'un objet technique
- Contrats entre applications tels que les services Web et données échangées (voir protocole SOAP)
- Jeux d'enregistrements issus d'une requête SQL
- Métacontenu d'un site Web tel qu'un format de définition des chaînes (CDF, *Channel Definition Format*) ;
- Conception d'interfaces graphiques à base de langages construits avec XML (XAML WPF ou Androïd)
- Documentaires des applications



2. Présentation des données

XML est un format de description de données. Il ne comporte pas d'attributs relatifs à la présentation de ces données. Pour cela, on aura recours à différentes techniques, certaines utilisées dans d'autres technologies, d'autres spécifiques à XML.

Parmi les langages de mise en page tiers, utilisés pour la mettre en forme un document XML, citons :

- CSS (Cascading StyleSheet), la solution la plus utilisée actuellement, étant donné qu'il s'agit d'un standard qui a déjà fait ses preuves avec HTML
- XSL (eXtensible StyleSheet Language), un langage de feuilles de style extensible développé spécialement pour XML.
- XSLT (eXtensible StyleSheet Language Transformation). Il permet de transformer un document XML en document HTML accompagné de feuilles de style.

3. Validation de la structure d'un document XML

XML fournit plusieurs techniques qui permettent de vérifier la syntaxe d'un document et de s'assurer de sa validité :

- Les DTD (Document Type Definition). Il s'agit de fichiers décrivant la structure des documents y faisant référence grâce à un langage adapté. Ainsi un document XML doit suivre scrupuleusement les conventions de notation XML et peut éventuellement faire référence à une DTD décrivant l'imbrication des éléments possibles. Un document suivant les règles de XML est appelé document bien formé. Un document XML possédant une DTD et étant conforme à celle-ci est appelé document valide.
- Les XSD (XML Schema Definition). Il s'agit d'un langage extrêmement puissant permettant de définir le schéma (la structure) des données et des plages de valeurs valides ainsi que des relations entre certains types de données XML.
- Les éléments du schéma sont définis selon différentes natures (éléments, attributs, types et groupes). Nous étudierons plus précisément les schémas XML, cette technologie ayant été implémentée dans l'architecture Dot Net et étant conforme aux exigences du W3C (World Wide Web Consortium).

3.1. Décodage d'un document XML

XML permet donc de définir un format d'échange selon les besoins de l'utilisateur et offre des mécanismes pour vérifier la validité du document produit.

L'application destinataire d'un document XML doit pouvoir extraire les données du document. Cette opération est possible à l'aide d'un outil appelé analyseur (en anglais parser, parfois francisé en parseur).

Le parseur permet d'une part d'extraire les données d'un document XML (on parle d'analyse du document ou de parsing) ainsi que de vérifier éventuellement la validité du document.

Dans le cadre de XAML ou d'Androïd, utilisé dans le cadre de définition des interfaces utilisateur au sein de WPF (Windows Presentation Foundation), le parseur générera aussi les objets graphiques à partir de l'analyse des balises du langage XML.

Ce qu'il convient de retenir ici c'est que toute application destinataire de documents XML doit disposer d'un parseur.

Le navigateur Microsoft Internet Explorer dispose de l'analyseur de document XML msxml.

3.2. Les avantages de XML

Les principaux atouts de XML:

- La lisibilité : une connaissance théorique basique suffit pour comprendre le contenu d'un document XML
- Auto descriptif et extensible
- Une structure arborescente : permettant de modéliser la majorité des problèmes informatiques
- Universalité et portabilité : les différents jeux de caractères sont pris en compte
- Facilement déployé : il peut être facilement distribué par n'importe quels protocoles à même de transporter du texte, comme http.
- Intégrabilité : un document XML est utilisable par toute application pourvue d'un analyseur (parser)
- XML est bâti sur une fondation Unicode, ce qui facilite la création des documents internationaux.
- XML est aujourd'hui présent au niveau de toutes les bases de données relationnelles professionnelles.

Un document XML doit pouvoir être utilisable dans tous les domaines d'applications. XML est particulièrement adapté à l'échange de données et de documents.

XML fait beaucoup évoluer les normes en matière d'EDI (Echange de Données Informatisé), ainsi de nombreux formats de données issus de XML apparaissent (il en existe plus d'une centaine) :

- XHTML qui est une reformulation du HTML 4 selon la syntaxe et les règles du XML mais est aujourd'hui supplanté par HTML 5.
- OOXML Open Office XML pour les applications bureautiques mais aussi les documents Office Windows
- OFX : Open Financial eXchange pour les échanges d'informations dans le monde financier
- MathML : Mathematical Markup Language permet de représenter des formules mathématique. Pour le lire, il faudra toutefois utiliser un plug-in particulier MathPlayer à télécharger sur le Net.
- CML : Chemical Markup Language permet de décrire des composés chimiques
- SMIL : Synchronized Multimedia Integration Language permet de créer des présentations multimédia en synchronisant diverses sources : audio, vidéo, texte,...
- VML (Vector Markup Language) permet de formaliser et de traiter des images vectorielles.
- SVG (Scalable Vector Graphics), idem.
- WML (WAP Markup Language)

3.3. Les inconvénients de XML

Il est rare qu'une technologie n'ait pas les inconvénients de ses avantages.

XML n'échappe pas à la règle et peut ne pas être approprié dans certaines situations.

Les documents XML sont plus volumineux que les formats binaires. Les jeux d'enregistrements issus d'une requête SQL auprès d'un SGBD seront beaucoup plus volumineux en XML que dans leur format binaire d'origine. Ils utilisent donc plus de bande passante, et d'espace de stockage.

De plus, l'analyse des documents peut s'avérer plus lente que celle de formats propriétaires optimisés, et consommer plus de ressources. Les parseurs sont généralement assez complexes et de performances relativement médiocres.

4. Structure d'un document XML

Vous trouvez dans l'entête d'un document XML trois éléments qui ne sont pas tous obligatoires.

1. Le premier élément appelé prologue, obligatoire, permet d'indiquer :
 - a. la version de la norme XML utilisée pour créer le document (cette indication est obligatoire)
 - b. le jeu de caractères (en anglais encoding) utilisé dans le document (attribut facultatif)
 - c. un attribut standalone (facultatif) précise si le document sera validé par une déclaration de type extérieure au document. La valeur par défaut est yes : elle indique au parseur de ne pas rechercher de document de validation externe.

Un exemple de prologue définit complètement :

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

2. Un deuxième élément optionnel permet de fournir des indications de traitement qui ne font pas partie du document XML mais qui influe sur le comportement. Ainsi, la désignation d'une feuille de style de transformation.

```
<?xml-stylesheet type="text/xsl" href="documentation.xsl"?>
```

3. Un troisième élément optionnel permet d'indiquer quel document sera chargé de la validation de la validation du fichier XML. Il s'agit de la déclaration d'un document DTD (Document Type Definition).

```
<!DOCTYPE doc SYSTEM "documentation.dtd" >
```

Les DTD sont aujourd'hui le plus souvent remplacés par des schémas XSD.

Vous pourrez alors trouver des indications comme celle-ci qui précise quel schéma valide le document. Nous verrons par la suite comment exploiter les schémas.

```
<Window x:Class="WpfApplication1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
```

Extrait d'un schéma XSD

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns="http://schemas.microsoft.com/practices/2010/unity"
            targetNamespace="http://schemas.microsoft.com/practices/2010/unity"
            elementFormDefault="qualified"
            attributeFormDefault="unqualified">

    <!-- Core unity config -->

    <xs:element name="unity" type="UnityConfigurationSection" />

    <xs:complexType name="UnityConfigurationSection">
        <xs:sequence>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element name="container" type="ContainerElement"/>
                <xs:element name="alias" type="AliasElement" />
                <xs:element name="sectionExtension" type="SectionExtensionElement" />
                <xs:element name="namespace" type="NamedElement" />
                <xs:element name="assembly" type="NamedElement" />
            </xs:choice>
        </xs:sequence>
    </xs:complexType>
```

Nous trouvons ensuite le contenu du document proprement dit :

```
<?xml version="1.0" encoding="utf-8" ?>
<Racine>
  - <Stagiaire>
    <IdStagiaire>1</IdStagiaire>
    <Nom>Bost</Nom>
    <Prenom>Vincent</Prenom>
    <Ville>Brive</Ville>
    <Region />
    <DateNaissance>1962-01-13T19:19:47.09375+02:00</DateNaissance>
    <Adresse />
    <CodePostal>19100</CodePostal>
    <TelephoneFixe />
    <TelephonePortable />
  </Stagiaire>
  - <Stagiaire>
    <IdStagiaire>2</IdStagiaire>
    <Nom>Julien</Nom>
    <Prenom>Frédéric</Prenom>
    <Ville>Limoges</Ville>
    <Region />
    <DateNaissance>1959-11-02T19:19:47.09375+02:00</DateNaissance>
    <Adresse />
    <CodePostal>19100</CodePostal>
    <TelephoneFixe />
    <TelephonePortable />
  </Stagiaire>
</Racine>
```

4.1. Les objets contenus dans un document XML

L'arbre des éléments, c'est-à-dire le véritable contenu du document XML, est constitué d'une hiérarchie de balises comportant éventuellement des attributs.

Un attribut est une paire clé valeur écrit sous la forme Cle="Valeur", ainsi une balise affectée d'un attribut aura la syntaxe suivante :

<balise cle ="valeur">

Toute donnée est ainsi encapsulée entre une balise ouvrante <balise> et une balise fermante </balise> (Sachant qu'une donnée peut éventuellement être un ensemble d'éléments XML). Ainsi un élément vide est uniquement constitué d'une balise spécifique dont la syntaxe est la suivante : <balise/>.

Les différents types d'objets inclus dans un document XML.

Balise

C'est un mot clef choisi par le concepteur du document qui permet de définir un élément <formation>

Élément

C'est un objet XML défini entre une balise de début et une balise de fin.

La balise de fin porte le même nom que la balise de début, mais elle est précédée d'un "slash".

<formation> Contenu de l'élément </formation>

Un élément peut aussi contenir d'autres éléments

Attribut

Un élément peut être qualifié par un ou plusieurs attributs.

Ces attributs ont la forme clef="valeur".

<formation IdFormation ="CDI" type="qualifiante" >

4.1.1. Règles de validité d'un document XML

Il existe toujours **une balise Racine** qui encapsule les autres balises du document.

Il est interdit en XML de faire chevaucher des balises, c'est-à-dire d'avoir une succession de balises du type:

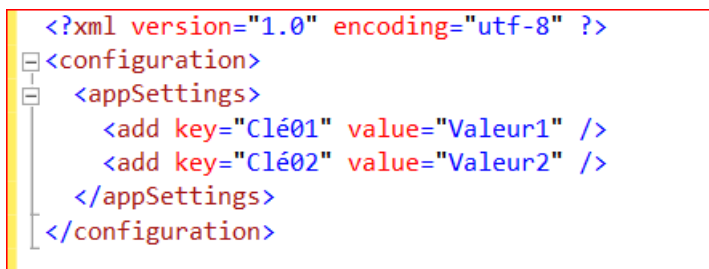
```
<balise1> <balise2></balise1></balise2>
```

Il est possible entre les balises (donc pas à l'intérieur d'une balise) d'ajouter:

- des espaces
- des tabulations
- des retours chariots

Cela est très utile pour définir une indentation des balises (ce qui est possible puisqu'elles ne se chevauchent pas).

Les balises, lorsqu'elles sont composées d'attributs simples et n'encapsulent pas d'autres éléments peuvent être présentées sous cette forme `< .../>` , comme dans l'exemple suivant :



```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="Clé01" value="Valeur1" />
    <add key="Clé02" value="Valeur2" />
  </appSettings>
</configuration>
```

The image shows an XML code snippet with a tree diagram on the left. The root element is `<?xml version="1.0" encoding="utf-8" ?>`. It has a single child element `<configuration>`. Inside `<configuration>` is another element `<appSettings>`. Inside `<appSettings>` are two `<add>` elements. The first `<add>` has attributes `key="Clé01"` and `value="Valeur1"`. The second `<add>` has attributes `key="Clé02"` and `value="Valeur2"`. Both `<add>` elements are self-closing (`/>`). The `<appSettings>` element is closed with `</appSettings>`, and the `<configuration>` element is closed with `</configuration>`.

4.1.2. Exemples de syntaxe

Document mal formé

```
<?xml version="1.0" ?>
<formation>
  CDI
</formation>
<durée>
  1755 heures
</durée>
```

Document bien formé

```
<?xml version="1.0" ?>
<formation>
  <Nom>
    CDI
  </Nom>
  <durée>
    1755 heures
  </durée>
</formation>
```

Figure 1 : Un document XML n'a qu'une balise racine

Document mal formé

```
<?xml version="1.0" ?>
<formation>
  <Nom>
    CDI
  </Nom>
  <durée>
    1755 heures
  </formation>
```

Document bien formé

```
<?xml version="1.0" ?>
<formation>
  <Nom>
    CDI
  </Nom>
  <durée>
    1755 heures
  </durée>
</formation>
```

Figure 2 : Toutes les balises doivent être fermées

<p>Document mal formé</p> <pre><?xml version="1.0" ?> <formation> <Nom> CDI <durée> 1755 heures </durée> </Nom> </formation></pre>	<p>Document bien formé</p> <pre><?xml version="1.0" ?> <formation> <Nom> CDI </Nom> <durée> 1755 heures </durée> </formation></pre>
--	---

Figure 3 : Les balises ne doivent pas se chevaucher

<p>Document mal formé</p> <pre><?xml version="1.0" ?> <formation> <Nom> CDI </Nom> <durée> 1755 heures </durée> <Module Valeur=U01/> <Module Valeur=U02/> </formation></pre>	<p>Document bien formé</p> <pre><?xml version="1.0" ?> <formation> <Nom> CDI </Nom> <durée> 1755 heures </durée> <Module Valeur= 'U01' /> <Module Valeur= 'U02' /> </formation></pre>
--	---

Figure 4 : Les valeurs des attributs sont entre guillemets (simples ou doubles)