eBook Gratuit

APPRENEZ xml

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#xml

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: xml

It is an unofficial and free xml ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official xml.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec xml

Remarques

XML est un langage de balisage utilisé pour stocker des données hiérarchiques dans des fichiers texte. Il est également connu sous le nom de données semi-structurées, comme JSON. XML est lisible par machine, mais peut également être lu et produit par des personnes.

XML est constitué d'éléments, parfois dénommés « *soupe d'étiquette»* , pouvant eux-mêmes contenir d'autres éléments et / ou du texte. Les éléments peuvent également contenir des attributs.

XML est souvent utilisé pour l'échange de données entre plates-formes, en particulier sur Internet. Il est également de plus en plus utilisé pour stocker des données semi-structurées dans des magasins de données NoSQL (bases de données XML / magasins de documents). De plus, il a la flexibilité de gérer les données orientées document (texte avec balisage), ce qui le rend très populaire dans le secteur de l'édition. XML est également largement utilisé pour les fichiers de configuration.

L'une des principales raisons pour lesquelles XML est si largement utilisé est qu'il est standardisé, avec de nombreux analyseurs disponibles, y compris open source. Cela rend le coût d'utilisation de XML inférieur à celui de l'invention de la nouvelle syntaxe.

Vous trouverez plus d'informations sur l'origine et les objectifs de XML dans la recommandation officielle du W3C .

Il existe deux versions de XML, présentées dans le tableau ci-dessous. Les éditions de chaque version ne sont que des révisions des documents originaux et non des modifications des normes.

La première version de XML est la 1.0 . XML 1.1 a été modifié en raison du changement de version Unicode de 2.0 à 3.1 et spécifie un ensemble de nouvelles règles pour l'utilisation et l'interprétation des nouveaux caractères Unicode.

Versions

Version	Date de sortie
1.0	1998-02-10
1.1	2001-12-13

Examples

Installation ou configuration

XML est une syntaxe, ce qui signifie qu'un simple éditeur de texte suffit pour démarrer.

Cependant, avoir un éditeur spécifique à XML qui vous indique quand et où votre document n'est pas bien formé est presque indispensable pour la productivité. De tels éditeurs peuvent également vous permettre de valider des documents XML par rapport à un schéma XML ou même de générer des schémas XML à partir de documents XML (et inversement).

Quelques exemples d'éditeurs sont oXygen, Atom, Eclipse et Altova XMLSpy. Une autre solution consiste à utiliser un analyseur XML de ligne de commande tel qu'Apache Xerces.

Les éléments de base

XML est constitué de composants de base, à savoir:

- élément
- texte
- les attributs
- commentaires
- instructions de traitement

Un élément a des crochets:

```
<element/>

<element>some content</element>
```

Un attribut apparaît dans une balise d'élément d'ouverture:

```
<element
  attribute-name="attribute value"
  other-attribute='single-quoted value'>
  ...
</element>
```

Le texte peut apparaître n'importe où dans ou entre les éléments:

```
<element>some more <b>bold</b> text</element>
```

Les commentaires utilisent la syntaxe suivante. Il est important de savoir que les commentaires XML, contrairement aux langages de programmation, font partie du modèle et seront visibles par l'application au-dessus de l'analyseur.

```
<!-- this is a comment -->
```

Les instructions de traitement permettent de transmettre des messages à l'application consommatrice (par exemple, comment afficher ou une feuille de style, etc.). XML ne limite pas le format des instructions de traitement.

```
<?target-application these are some instructions?>
```

Vous trouverez plus de détails sur les blocs de construction dans cette rubrique.

Bien formé

Un document XML est un fichier texte conforme aux règles de bonne forme de la spécification XML. Un tel document conforme est dit *bien formé* (à ne pas confondre avec *valide*). XML est très strict et bien conçu par rapport à d'autres langages tels que le HTML. Un fichier texte mal formé n'est pas considéré comme XML et ne peut pas être utilisé par les applications consommatrices.

Voici quelques règles qui s'appliquent aux documents XML:

1. XML utilise une syntaxe très auto-descriptive. Un prologue définit la version XML et l'encodage des caractères:

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

2. Il doit y avoir exactement un élément de niveau supérieur.

   Cependant, les commentaires, les instructions de traitement ainsi que la déclaration XML initiale sont également autorisés au niveau supérieur. Le texte et les attributs ne le sont pas.

   ```
   <?xml version="1.0"?>
   <!-- some comments -->
   <?app a processing instruction?>
   <root/>
   <!-- some more comments -->
   ```

3. Les éléments peuvent être imbriqués, mais doivent être "correctement imbriqués":

   ```
   <name>
     <first-name>John</first-name>
     <last-name>Doe</last-name>
   </name>
   ```

   Les balises de début et de fin d'un élément incorporé doivent être dans les balises de début et de fin de son élément conteneur. Un chevauchement d'éléments est illégal. En particulier, ce n'est pas du XML bien formé: `<foo><bar></foo></bar>`

4. Les attributs ne peuvent apparaître que dans les balises d'élément d'ouverture ou les balises d'élément vides, pas dans les balises d'élément de fermeture. Si la syntaxe d'attribut apparaît entre les éléments, elle n'a aucune signification et est analysée en tant que texte.

   ```
   <person first-name="John" last-name="Doe"/>
   ```

   Ce n'est pas bien formé: ~~`<person></person first-name="John"/>`~~

5. Les commentaires, les instructions de traitement, le texte et d'autres éléments peuvent apparaître n'importe où dans un élément (par exemple, entre sa balise d'ouverture et de fermeture) mais pas à l'intérieur des balises.

```
<element>
    This is some <b>bold</b> text.
    <!-- the b tag has no particular meaning in XML -->
</element>
```

Cet exemple n'est pas bien formé: ~~`<element <-- comment --> />`~~

6. Le caractère `<` peut ne pas apparaître dans le texte ou dans les valeurs d'attribut.
7. Le `"` caractère peut ne pas apparaître dans les valeurs d'attribut avec `"` . Le caractère `'` ne peut pas apparaître dans les valeurs d'attribut avec `'` .
8. La séquence de caractères `--` peut ne pas apparaître dans un commentaire.
9. Les caractères littéraux `<` et `&` doivent être échappés par leurs entités respectives `<` et `&` .

Bonjour le monde

```
<?xml version="1.0"?>
<?speech-generator voice="Siri"?>
<root xmlns:vocabulary="http://www.example.com/vocabulary">
  <!-- These are the standard greetings -->
  <vocabulary:greetings>
    <vocabulary:greeting xml:lang="en-US" type="informal">
      Hi!
    </vocabulary:greeting>
    <vocabulary:greeting xml:lang="en-US" type="intermediate">
      Hello!
    </vocabulary:greeting>
    <vocabulary:greeting xml:lang="en-US" type="formal">
      Good morning to <b>you</b>!
    </vocabulary:greeting>
  </vocabulary:greetings>
</root>
```

Espaces de noms

Les noms d'élément et d'attribut vivent dans les espaces de noms qui sont des URI. Les espaces de noms sont liés aux préfixes utilisés dans les noms d'éléments et d'attributs réels, appelés QNames.

Ce document lie un espace de noms au préfixe de `prefix` et définit un espace de noms par défaut, lié à l'absence de préfixe.

```
<?xml version="1.0"?>
<document
    xmlns="http://www.example.com/default-namespace"
    xmlns:prefix="http://www.example.com/another-namespace">
  <prefix:element/>
</document>
```

Vous trouverez plus de détails sur les espaces de noms dans cette rubrique.

Lire Démarrer avec xml en ligne: https://riptutorial.com/fr/xml/topic/882/demarrer-avec-xml

Chapitre 2: Blocs de construction

Examples

Éléments

Les éléments fournis avec des crochets sont le composant le plus important de XML.

Les éléments peuvent soit être vides, auquel cas ils sont constitués d'une balise vide (notez la barre oblique de fin):

```
<an-empty-element/>
```

Ou bien ils peuvent avoir du contenu, auquel cas ils ont une balise d'ouverture (pas de barre oblique) et une balise de fermeture (début d'une barre oblique):

```
<a-non-empty-element>Content</a-non-empty-element>
```

Les éléments peuvent être imbriqués (mais uniquement entre les balises d'ouverture et de fermeture):

```
<parent-element>
  <child-element/>
  <another-child-element>
    Some more content.
  </another-child-element>
</parent-element>
```

Les noms d'éléments sont appelés QNames (noms qualifiés). Tous les éléments ci-dessus ne sont pas dans un espace de noms, mais les noms d'éléments peuvent également être définis dans des espaces de noms en utilisant des préfixes comme ceux-ci:

```
<my-namespace:parent-element xmlns:my-namespace="http://www.example.com/">
  <my-namespace:child-element/>
  <my-namespace:another-child-element>
    Some more content.
  </my-namespace:another-child-element>
</my-namespace:parent-element>
```

Les espaces de noms et les noms d'éléments sont décrits plus en détail dans cette section de la documentation .

Les attributs

Les attributs sont des paires nom-valeur associées à un élément.

Ils sont représentés par des valeurs entre guillemets simples ou doubles à l'intérieur de la balise

element d'ouverture ou par la balise element vide s'il s'agit d'un élément vide.

```
<document>
  <anElement foo="bar" abc='xyz'><!-- some content --></anElement>
  <anotherElement a="1"/>
</document>
```

Les attributs ne sont pas ordonnés (contrairement aux éléments). Les deux éléments suivants ont les mêmes ensembles d'attributs:

```
<foo alpha="1" beta="2"/>

<foo beta="2" alpha="1"/>
```

Les attributs ne peuvent pas être répétés dans le même élément (contrairement aux éléments). Le document suivant n'est pas bien formé: ~~`<foo a="x" a="y"/>`~~ car l'attribut `a` apparaît deux fois dans le même élément.

Le document suivant est bien formé. Les valeurs peuvent être identiques, c'est le nom de l'attribut qui ne peut pas être répété.

```
<foo a="x" b="x"/>
```

Les attributs ne peuvent pas être imbriqués (contrairement aux éléments).

Texte

Le texte est composé de tous les caractères en dehors de tout balisage (balises d'élément d'ouverture, balises d'élément de fermeture, etc.).

```
<?xml version="1.0"?>
<document>
  This is some text and <b>this is some more text</b>.
</document>
```

La terminologie XML précise pour le texte est constituée de *données de caractères* . La spécification XML utilise en réalité le mot *texte* pour l'ensemble du document XML, ou une entité analysée, car elle définit XML au niveau syntaxique. Cependant , certains modèles de données tels que le XDM (XQuery et XPath Data Model), qui représentent des documents XML comme des arbres, se réfèrent à des données de caractères en tant que *nœuds de texte,* tel que le *texte* est souvent compris comme synonyme de données de caractère dans la pratique.

Les données de caractère ne peuvent pas contenir un caractère `<` - ceci serait interprété comme le premier caractère d'une balise d'élément d'ouverture - il ne peut pas non plus contenir la séquence de caractères `]]>` . Les caractères appropriés doivent être échappés avec une référence d'entité à la place.

```
<?xml version="1.0"?>
<document>
```

```
  It is fine to escape the &lt; character, as well as ]]&gt;.
</document>
```

Pour plus de commodité, on peut également échapper à un gros morceau de texte avec une section CDATA (mais la séquence `]]>` n'est toujours pas autorisée pour des raisons évidentes):

```
<?xml version="1.0"?>
<document>
  <![CDATA[
    In a CDATA section, it is fine to write < or even & and entity references
    such as &amp; are not resolved.
  ]]>
</document>
```

commentaires

Les commentaires en XML ressemblent à ceci:

```
<!-- This is a comment -->
```

Ils peuvent apparaître dans le contenu de l'élément ou au niveau supérieur:

```
<?xml version="1.0"?>
<!-- a comment at the top-level -->
<document>
  <!-- a comment inside the document -->
</document>
```

Les commentaires ne peuvent pas apparaître dans les balises ou dans les attributs:

~~<element <!-- comment with -- inside --> />~~

ou

~~<element attr="<!-- comment with -- inside -->"/>~~

ne sont pas bien formés.

La séquence de caractères `--` ne peut pas apparaître au milieu d'un commentaire. Ce n'est pas du XML bien formé:

~~<!-- comment with -- inside -->~~

Les commentaires en XML, contrairement à d'autres langages tels que C ++, font **partie du modèle de données** : ils sont analysés, transférés et visibles par l'application qui en consomme.

Instructions de traitement

Une instruction de traitement permet de transmettre directement certaines informations ou instructions à l'application via l'analyseur.

```
<?my-application some instructions ?>
```

Le jeton après le point d'interrogation initial (ici `my-application`) s'appelle la cible et identifie l'application à laquelle l'instruction est destinée. Ce qui suit n'est plus spécifié et c'est à l'application de l'interpréter. Les références d'entité et de caractère ne sont pas reconnues.

Il peut apparaître au niveau supérieur ou dans le contenu de l'élément.

Lire Blocs de construction en ligne: https://riptutorial.com/fr/xml/topic/1590/blocs-de-construction

Chapitre 3: Catalogues XML

Introduction

Un catalogue XML est composé d'entrées d'un ou plusieurs fichiers d'entrée de catalogue. Un fichier d'entrée de catalogue est un fichier XML dont l'élément de document est catalog et dont le contenu suit la DTD du catalogue XML définie par OASIS à l' adresse http://www.oasis-open.org/committees/entity/spec.html . La plupart des éléments sont des entrées de catalogue, chacune servant à mapper un identifiant ou une URL vers un autre emplacement.

Examples

Entrée de catalogue pour résoudre l'emplacement DTD

1

2 3

```
<public
   publicId="-//OASIS//DTD DocBook XML V4.5//EN"  4
   uri="docbook45/docbookx.dtd"/>

<system
   systemId="http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd"  5
   uri="docbook45/docbookx.dtd"/>

<system
   systemId="docbook4.5.dtd"    6
   uri="docbook45/docbookx.dtd"/>
```

Lire Catalogues XML en ligne: https://riptutorial.com/fr/xml/topic/10875/catalogues-xml

Chapitre 4: DTD

Introduction

La déclaration de type de document XML communément appelée DTD est un moyen de décrire précisément le langage XML. Les DTD vérifient la validité, la structure et le vocabulaire d'un document XML par rapport aux règles grammaticales du langage XML approprié. Une DTD définit la structure et les éléments juridiques et attributs d'un document XML.

Examples

Déclaration de type de document

Un document XML peut contenir une DTD. DTD signifie *Déclaration de type de document* . Une DTD commence par `<!DOCTYPE root-element-name >` où `<!DOCTYPE root-element-name >` `doc-element-name` doit correspondre au nom de l'élément de document (le seul élément du niveau supérieur).

```
<?xml version="1.0"?>
<!DOCTYPE document>
<document>
  <!-- the rest of the document -->
</document>
```

Entités

Une DTD peut contenir des déclarations d'entité.

```
<?xml version="1.0"?>
<!DOCTYPE document [
  <!ENTITY my-entity "This is the replacement text">
]>
<document>
  <!-- the rest of the document -->
</document>
```

Les entités sont décrites en détail dans cette rubrique .

Document XML avec une DTD interne

Une DTD est appelée DTD interne si des éléments sont déclarés dans les fichiers XML. Pour le référencer en tant que DTD interne, l'attribut autonome dans la déclaration XML doit être défini sur yes.

Un fichier XML décrivant une note contenant des propriétés, des messages et des messages avec une DTD interne ressemblera à ceci:

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

```
<!DOCTYPE note [
<!ELEMENT note (to,from,message>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT message (#PCDATA)>
]>
<note>
<to>Mr.X</to>
<from>Mr.Y</from>
<message>Stack Overflow is awesome </message>
</note>
```

Document XML avec une DTD externe

Dans les éléments DTD externes sont déclarés en dehors du fichier XML. Ils sont accessibles en spécifiant les attributs du système qui peuvent être le fichier .dtd légal ou une URL valide. Pour le référencer en tant que DTD externe, l'attribut autonome de la déclaration XML doit être défini sur no.

Un code XML décrivant une note contenant une propriété à partir de et du message est indiqué ci-dessous.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
  <to>Mr.X</to>
  <from>Mr.Y</from>
  <message>Stack Overflow is awesome</message>
</note>
```

DTD externe pour le XML ci-dessus, *note.dtd* est donnée ci-dessous

```
<!DOCTYPE note [
<!ELEMENT note (to,from,message>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT message (#PCDATA)>
]>
```

Lire DTD en ligne: https://riptutorial.com/fr/xml/topic/3897/dtd

Chapitre 5: Échapper

Remarques

Les caractères peuvent être échappés en XML à l'aide de références d'entité et de références de caractères, ou de sections CDATA.

XML pré-définit cinq entités:

Entité nommée	Texte de remplacement
ampli	Et
coté	"
apos	'
lt	<
gt	>

Les applications consommatrices ne sauront pas si chaque personnage a été échappé ou non et comment.

Examples

Esperluette

Le caractère `&` apparaît en premier dans les références d'entité et doit être échappé dans le contenu de l'élément ou dans le contenu de l'attribut.

```
<?xml version="1.0"?>
<document attribute="An ampersand is escaped as &amp;">
  An ampersand can also be escaped as &amp; in element content.
</document>
```

Signe inférieur

Le caractère `<` apparaît en premier dans les balises d'entité et doit être échappé dans le contenu de l'élément ou dans le contenu de l'attribut.

```
<?xml version="1.0"?>
<document attribute="A lower-than sign is escaped as &lt;">
  2 + 2 &lt; 5
</document>
```

Signe supérieur à

La séquence de caractères `]]>` n'est pas autorisée dans le contenu de l'élément. La manière la plus simple de s'en échapper est de s'échapper `>` as `>` .

```
<?xml version="1.0"?>
<document>
  The sequence ]]&gt; cannot appear in element content.
</document>
```

Apostrophes et citations

Les valeurs d'attribut peuvent apparaître entre guillemets simples ou doubles. Le caractère approprié doit être échappé.

```
<?xml version="1.0"?>
<document
  quot-attribute="This is a &quot;double quote&quot; and this one is 'simple'"
  apos-attribute='This is a &apos;simple quote&apos; and this one is "double"'>
</document>
```

Sections CDATA

De plus longues parties de texte contenant des caractères spéciaux peuvent être échappées avec une section CDATA. Les sections CDATA ne peuvent apparaître que dans le contenu de l'élément.

```
<?xml version="1.0"?>
<document>
  This is a CDATA section : <![CDATA[ plenty of special characters like & < > " ; ]]>
</document>
```

Une section CDATA ne peut pas contenir la séquence `]]>` car elle la termine.

Références de personnage

Les caractères peuvent être échappés à l'aide de références de caractères, de contenu d'élément ou de valeurs d'attribut. Leur code Unicode peut être spécifié en décimal ou en hexadécimal.

```
<?xml version="1.0"?>
<document>
  The line feed character can be escaped with a decimal (&#10;) or hex (&#xA;)
  representation of its Unicode codepoint (10).
</document>
```

XML restreint les caractères pouvant apparaître dans un document, même échappé. En particulier, les seuls caractères de contrôle autorisés sont le saut de ligne (10), le retour chariot (13) ou la tabulation horizontale (9).

Lire Échapper en ligne: https://riptutorial.com/fr/xml/topic/3685/echapper

Chapitre 6: Entités

Remarques

Du point de vue du stockage, un document XML est constitué d'entités. L'une des entités est l'entité document, qui est le document XML principal lui-même.

Les entités peuvent être classées comme telles (triées par ordre décroissant d'utilisation):

- **entité document** : c'est le fichier XML principal.
- **Entités générales internes** : il s'agit de l'entité la plus courante en dehors de l'entité de document et de celle que connaissent le plus les utilisateurs XML. Souvent, le mot **entité** est utilisé pour eux. Ils permettent de spécifier des raccourcis pour des textes de remplacement plus longs dans le contenu du document. Ils sont déclarés dans la DTD.
- **le sous** - **ensemble DTD externe** : un autre fichier dans lequel une partie de la DTD est externalisée.
- **Paramètre entités** : raccourcis, à utiliser dans la DTD.
- **entités générales externes analysées** : ce sont des fragments XML stockés dans d'autres fichiers.
- **entités non analysées** : il peut s'agir de fichiers sur lesquels XML ne place aucune restriction, y compris des images, des sons, etc.

Dans de nombreux cas, un document XML est constitué uniquement de l'entité document.

Examples

Entités générales prédéfinies

XML pré-définit cinq entités générales pouvant être utilisées sans les déclarer:

```
& " ' < >
```

Ils sont associés aux noms `amp` , `quot` , `apos` , `lt` et `gt` .

```
<?xml version="1.0"?>
<entities>
  &amp; is an ampersand.
  &quot; is a quote.
  &apos; is an apostrophe.
  &lt; is a lower-than sign.
  &gt; is a greater-than sign.
</entities>
```

Entités générales (internes) définies par l'utilisateur

Il est possible de définir ses propres entités générales. La déclaration se produit dans le sous-

ensemble DTD, avec un nom et le texte de remplacement associé.

Il peut ensuite être utilisé dans le document en utilisant la syntaxe de référence de l'entité `&...;` , soit dans le texte, soit dans les valeurs d'attribut.

```
<?xml version="1.0"?>
<!DOCTYPE my-document [
  <!ENTITY my-entity "This is my entity">
]>
<my-document>
    The entity was declared as follows: &my-entity;
    <element attribute="Entity: &my-entity;"/>
</my-document>
```

Entités analysées externes

Les fragments XML, également connus sous le nom d' *entités analysées externes* , peuvent être stockés dans des fichiers séparés.

Les fragments XML, contrairement aux documents XML, sont moins restrictifs, car plusieurs éléments peuvent apparaître au niveau supérieur, ainsi que des nœuds de texte. Comme un document XML, une entité externe analysée peut commencer par une déclaration XML, mais cette déclaration n'est pas considérée comme faisant partie de son texte de remplacement.

Voici un exemple d'entité externe analysée:

```
<?xml version="1.0" encoding="UTF-8"?>
This is some text
<element/>
<element/>
```

Une entité analysée externe peut alors être déclarée dans un document XML, dans la DTD, et peut être utilisée avec une référence d'entité, qui a la même syntaxe que pour les entités internes générales:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE root [
<!ENTITY fragment SYSTEM "fragment.xml">
]>
<root>
    &fragment;
</root>
```

Avec la référence d'entité résolue, ce document équivaut à:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE root [
<!ENTITY fragment SYSTEM "fragment.xml">
]>
<root>
  This is some text
  <element/>
```

```
  <element/>
</root>
```

Chaque balise d'élément d'ouverture dans une entité analysée externe doit avoir une balise de fin correspondante: il n'est pas permis de répartir des éléments individuels sur plusieurs entités, ni d'étaler le balisage.

Un analyseur de validation est requis pour résoudre la référence de l'entité et inclure son texte de remplacement dans le document comme ci-dessus. Un analyseur non validant peut ignorer cela et indiquer à l'application consommatrice qu'il existe une référence non résolue à une entité analysée externe.

Lire Entités en ligne: https://riptutorial.com/fr/xml/topic/2302/entites

Chapitre 7: Espaces de noms

Remarques

Les noms d'éléments et d'attributs en XML sont appelés QNames (noms qualifiés).

Un QName est composé de:

- un espace de noms (un URI)
- un préfixe (un NCName, NC car il ne contient pas de deux-points)
- un nom local (un NCName)

Seuls l'espace de noms et le nom local sont pertinents pour comparer deux noms QNames. Le préfixe est uniquement un proxy pour l'espace de noms.

L'espace de nom et le préfixe sont facultatifs, mais l'espace de nom est toujours présent si le préfixe est présent (ceci est garanti au niveau syntaxique, donc cela ne peut pas être mal fait).

La représentation lexicale d'un QName est un `prefix:local-name` . L'espace de nom est lié séparément en utilisant les attributs spéciaux `xmlns:...` (rappel: les attributs commençant par *xml* sont réservés en XML).

Si le préfixe est vide, aucun deux-points n'est utilisé dans la représentation lexicale du QName, qui contient uniquement le `local-name` . Les QNames avec un préfixe vide n'ont aucun espace de noms (si aucun espace de noms par défaut n'est dans la portée) ou sont dans l'espace de noms par défaut.

Examples

Lier un préfixe à un espace de noms

Un espace de noms est un URI, mais pour éviter toute verbosité, les préfixes sont utilisés comme proxy.

Dans l'exemple suivant, le préfixe `my-prefix` est lié à l'espace de noms `http://www.example.com/my-namespace` en utilisant l'attribut spécial `xmlns:my-prefix` (`my-prefix` peut être remplacé par tout autre préfixe) :

```
<?xml version="1.0"?>
<my-prefix:foo xmlns:my-prefix="http://www.example.com/my-namespace">
  <!-- the element my-prefix:foo
       lives in the namespace http://www.example.com/my-namespace -->
</my-prefix:foo>
```

Absence d'espace de nommage

En XML, les noms d'éléments et d'attributs vivent dans les espaces de noms.

Par défaut, ils ne sont dans aucun espace de noms:

```
<?xml version="1.0"?>
<foo attr="value">
  <!-- the foo element is in no namespace, neither is the attr attribute -->
</foo>
```

Non pertinence des préfixes

Ces deux documents sont sémantiquement équivalents, car les espaces de noms importent, pas les préfixes.

```
<?xml version="1.0"?>
<myns:foo xmlns:myns="http://www.example.com/my-namespace">
</myns:foo>

<?xml version="1.0"?>
<ns:foo xmlns:ns="http://www.example.com/my-namespace">
</ns:foo>
```

Espace de noms par défaut

L'espace de noms par défaut est l'espace de noms correspondant à l'absence de préfixe. Il peut être déclaré avec l'attribut spécial `xmlns` .

```
<?xml version="1.0"?>
<foo xmlns="http://www.example.com/my-namespace">
  <!-- the element foo is in the namespace
       http://www.example.com/my-namespace -->
</foo>
```

Si aucun espace de noms par défaut n'est déclaré, les noms sans préfixe ne sont dans aucun espace de noms.

Noms d'attribut sans préfixe

Les éléments et les attributs se comportent différemment par rapport aux espaces de noms par défaut. C'est souvent la source de confusion.

Un attribut dont le nom n'a pas de préfixe ne réside dans aucun espace de noms, **même lorsqu'un espace de noms par défaut est dans la portée** .

```
<?xml version="1.0"?>
<foo attr="value" xmlns="http://www.example.com/my-namespace">
  <!-- The attribute attr is in no namespace, even though
       a default namespace is in scope. The element foo,
       however, is in the default namespace. -->
</foo>
```

Portée des liaisons d'espace de noms

Une liaison d'espace de noms (attribut spécial `xmlns` ou `xmlns:...`) est envisageable pour tous les descendants de l'élément englobant, y compris cet élément.

```
<?xml version="1.0"?>
<root>
  <my:element xmlns:my="http://www.example.com/ns1">
    <!-- here, the prefix my is bound to http://www.example.com/ns1 -->
  </my:element>
  <my:element xmlns:my="http://www.example.com/ns2">
    <!-- here, the prefix my is bound to http://www.example.com/ns2 -->
  </my:element>
</root>
```

La liaison peut être remplacée dans un élément imbriqué (cela affecte cependant la lisibilité):

```
<?xml version="1.0"?>
<my:element xmlns:my="http://www.example.com/ns1">
  <!-- here, the prefix my is bound to http://www.example.com/ns1 -->
  <my:first-child-element/>

  <my:child-element xmlns:my="http://www.example.com/ns2">
    <!-- here, the prefix my is bound to http://www.example.com/ns2,
         including for the element my:child-element -->
  </my:child-element>

  <!-- here, the prefix my is bound to http://www.example.com/ns1 -->
  <my:last-child-element/>

</my:element>
```

Il est très courant de déclarer toutes les liaisons d'espace de noms dans l'élément racine, ce qui améliore la lisibilité.

```
<?xml version="1.0"?>
<root
  xmlns="http://www.example.com/default-namespace"
  xmlns:ns1="http://www.example.com/ns1"
  xmlns:ns2="http://www.example.com/ns2">

  <ns1:element>
    <ns2:other-element/>
  </ns1:element>

</root>
```

Lire Espaces de noms en ligne: https://riptutorial.com/fr/xml/topic/1593/espaces-de-noms

Chapitre 8: Schéma XML

Introduction

Le schéma XML est communément appelé définition de schéma XML (XSD). Il est utilisé pour décrire et valider la structure et le contenu des données XML. Le schéma XML définit les éléments, les attributs et les types de données.

Examples

Un exemple de document XSD

Un XSD qui décrit une information de contact sur une entreprise est donné ci-dessous.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://NamespaceTest.com/CommonTypes"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  elementFormDefault="qualified">
<xs:element name="contact">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="name" type="xs:string" />
            <xs:element name="company" type="xs:string" />
            <xs:element name="phone" type="xs:int" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

Dans l'exemple ci-dessus, les attributs de la deuxième ligne

> <xs: schema targetNamespace = " http://NamespaceTest.com/CommonTypes " xmlns: xs = " http://www.w3.org/2001/XMLSchema "
> elementFormDefault = "qualifiée">

Les attributs 'targetnamespace' et elementFormDefault sont facultatifs.

Lire Schéma XML en ligne: https://riptutorial.com/fr/xml/topic/8983/schema-xml

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec xml	Al.G., Burkart, Community, Elizaveta Revyakina, Ghislain Fourny, Joe, JohnRC, Kin, MAZux, Mohammad Arman, RamenChef, TuringTux, w5m, Wolfgang Schindler
2	Blocs de construction	Ghislain Fourny, Hoylen
3	Catalogues XML	Mistletoe
4	DTD	Dipesh Poudel, Ghislain Fourny
5	Échapper	Ghislain Fourny
6	Entités	Ghislain Fourny
7	Espaces de noms	Ghislain Fourny
8	Schéma XML	Dipesh Poudel