Université el Djilali Bounaama – Khemis Miliana
Département mathématique et informatiques
Master 1 en ingénierie du logiciels

XML

(Extensible Markup Language)

Sous direction de :
D^rF . haniche

MAZEGHOU Ahlem
SGHIR ABBAS bouchra
BOUHARAOUA chaima

2022 - 2023

Plan de travail

01 INTRODUCTION

02 XML

03 PRINCIPE

04 DOMAINE D 'APPLICATION

05 LES OUTILS

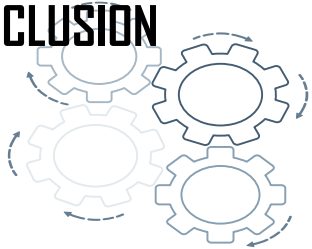
06 LANGAGE XML

07 DTD

08 XML SCHEMA

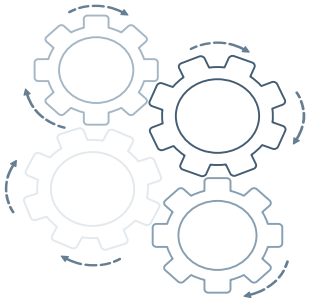
09 XPATH

10 CONCLUSION



INTRODUCTION

- XML signifie Extensible Markup Language c'est un langage de balisage qui est principalement utilisé pour encoder des documents et des données, et un format largement utilisé pour l'échange de données dans les applications modernes, y compris les services Web et applications mobiles.



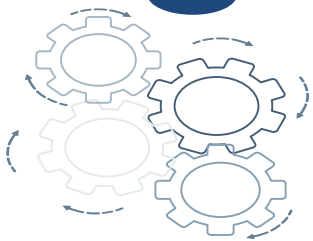
XML

Le langage XML (eXtended Markup Language) est un format général de documents orienté texte. Il s'est imposé comme un standard incontournable de l'informatique. Il est aussi bien utilisé pour le stockage de document que pour la transmission de données entre applications.

Sa simplicité, sa flexibilité et ses possibilités d'extension ont permis de l'adapter à de multiples domaines allant des données géographiques au dessin vectoriel en passant par les échanges commerciaux.

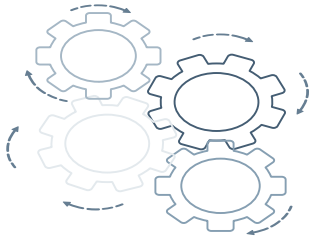
De nombreuses technologies se sont développées autour de XML et enrichissent ainsi son environnement. Le langage XML dérive de SGML (Standard Generalized Markup Language) et de HTML (HyperText Markup Language).

Comme ces derniers, il s'agit d'un langage orienté texte et formé de balises qui permettent d'organiser les données de manière structurée.



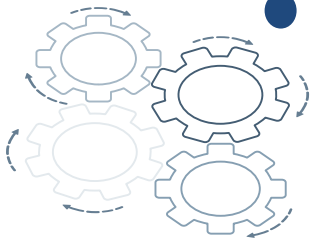
PRINCIPE

Le langage XML (Extensible Markup Language) est un langage de balisage qui permet aux développeurs de créer des balises personnalisées pour décrire les données qu'ils souhaitent stocker. Il est conçu pour être lisible à la fois par l'homme et la machine, ce qui le rend idéal pour le développement web et le transport de données. Les principes de base de XML impliquent l'utilisation de balises pour baliser des informations, l'utilisation d'attributs pour définir les balises et la possibilité de faire des imbrications de balises entre elles. XML est également extensible, ce qui signifie que de nouvelles balises peuvent être ajoutées au besoin.



Pourquoi XML ?

- Séparation de la structure et de la présentation
- Moins confus que HTML
- Plus simple que SGML
- Idéal pour l'échange de données semi-structurées
- Utilisable entre machines



Pourquoi XML ?

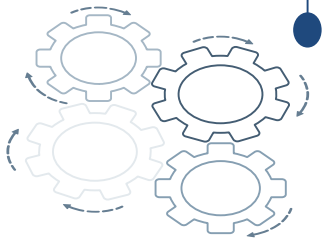
- HTML: documents sur le web
  - Langage *de présentation pour les documents du web*
  - Ensemble de balises et grammaire fixes, mélange d'éléments de structure de document et de mise en page

➢ **Difficile de déduire la signification du contenu**

- Données structurées : bases de données
  - Décrivent le contenu, pas la présentation qu'on peut en faire
  - Structure régulière basée sur des types simples: *string, int, boolean, …*

➢ **Les documents du web sont mal adaptés à cette structuration rigide**

  - Texte, structure variable



DOMAIN D'APPLICATION

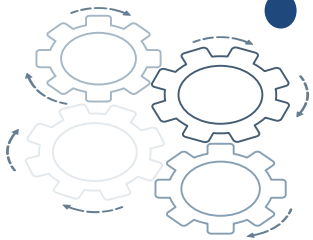
XML peut être utilisé dans de nombreux domaines d'application, notamment :

1. Web : XML est largement utilisé pour la création de pages web dynamiques et interactives.

2. Base de données : XML est utilisé pour stocker des données dans des bases de données, ce qui permet une meilleure gestion et une recherche plus facile des informations.

3. Échange de données : XML est utilisé pour échanger des données entre différentes applications et systèmes, car il fournit un format standardisé et facilement lisible.

4. Services web : XML est utilisé pour la communication entre les services web, car il fournit un format standardisé pour l'échange de données.



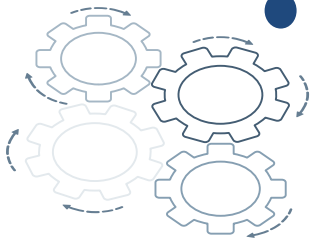
DOMAIN D'APPLICATION

5. Gestion de contenu : XML est utilisé pour la création et la gestion de contenu numérique, tels que les documents, les livres électroniques et les publications en ligne.

6. Applications mobiles : XML est utilisé pour stocker des données dans les applications mobiles, ce qui permet une meilleure gestion des informations sur les appareils mobiles.

7. Traitement du langage naturel : XML est utilisé pour représenter des structures linguistiques complexes dans le traitement du langage naturel.

8. Intelligence artificielle : XML est utilisé dans le domaine de l'intelligence artificielle pour stocker et échanger des connaissances entre différents systèmes informatiques



LES OUTILS

Les outils de XML sont des logiciels et des bibliothèques qui permettent de créer, de manipuler et de valider des documents XML. Voici quelques exemples d'outils XML :

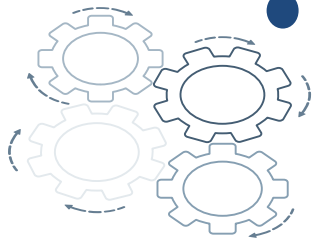
1. Editeurs XML : Ce sont des logiciels qui permettent d'éditer des fichiers XML en fournissant une interface graphique pour faciliter la saisie et la modification du code.

2. Parseurs XML : Ce sont des programmes qui analysent les documents XML pour en extraire les données et les transformer en structures de données utilisables par d'autres applications.

3. Transformateurs XSLT : Ce sont des outils qui permettent de transformer un document XML en un autre format, comme HTML ou PDF, en utilisant le langage XSLT.

4. Validateurs XML : Ce sont des programmes qui vérifient si un document XML est conforme à une DTD (Document Type Definition) ou à un schéma XML.

5. Bibliothèques de traitement XML : Ce sont des bibliothèques logicielles qui fournissent des fonctions pour manipuler les documents XML dans différents langages de programmation, tels que Java, Python ou C++.



Langage XML

❑ Balise (ou tag ou label)

Marque de début et la fin permettant de repérer un élément textuel

Forme: <balise> de début, </balise> de fin

Sensible à la casse (Majuscule/minuscule)

❑ Elément de données

Texte encadré par une balise de début et une de fin

Les éléments de données peuvent être imbriquées

Exemple:

```
<producteur>
<adresse>
<rue>A. Briand</rue>
<ville>Alger</ville></adresse>
</producteur>
```

RAPPELLE THEOIRIQUE

- **Elément vide:**

Les éléments vides ne contiennent rien entre leurs balises de début et de fin. Dans ce cas, on peut utiliser une forme raccourcie, à savoir le nom de la balise de début suivi d'une barre oblique.

Exemple: <Nom/>

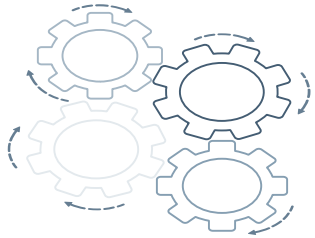
- **Attribut**

Dans la balise d'ouverture d'un élément, on peut définir des attributs qui définissent des propriétés de l'élément.

- **Syntaxe :** nomAttribut=<valeur>

Exemple:

<rapport langue="FR" date-modif="2015-01-07">

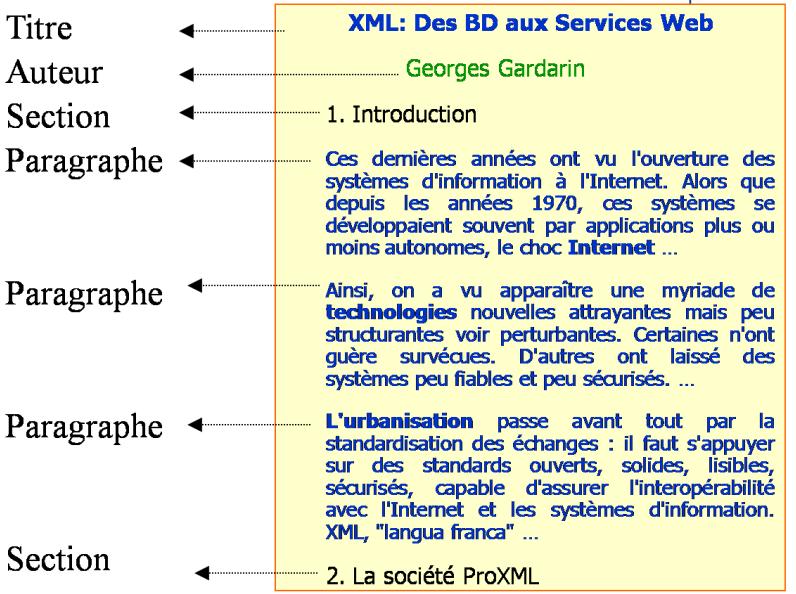


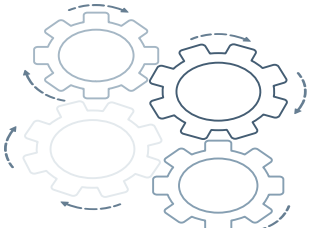
Structure d'un document XML

Un document XML contient :

- ❑ Un prologue, éventuellement vide
- ❑ Un arbre d'éléments (avec une racine)
- ❑ Des commentaires et des instructions de traitement

(facultatifs)





Les noms des éléments

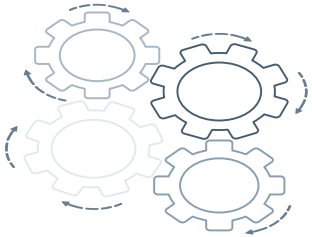
nom valide d'élément :

Est formé de lettres, chiffres, et de caractères de ponctuation : - (tiret), _ (*underscore*) ;

Doit commencer par une lettre ou _ ;

Est sensible a la casse (majuscule/minuscule) ;

Ne doit pas commencer par xml (ni XML, Xml, xMl...).



Les noms des éléments

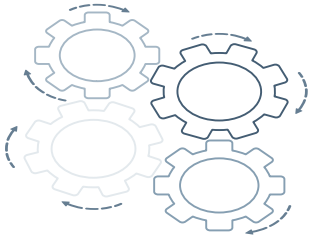
Tous les éléments peuvent contenir un ou plusieurs attributs.
Un attribut est composé d'un nom et d'une valeur.

Syntaxe : nomAttribut="valeur"
Exemple :

```
<section num="1"> Les moteurs de recherche </section>
```

Il ne peut être présent que dans la balise ouvrante de l'élément
Exemple :

```
<img src="ours.gif" width="56" height="100" />
```



Entités prédéfinies et entités caractères

- **Entités :** alias qui servent à établir un lien entre un nom symbolique et un texte de remplacement ou un pointeur vers une ressource externe. Elles peuvent être:

- **Entités prédéfinies:** Les caractères : <, >, &, ', " ne peuvent être utilisés dans le texte, car utilisés dans le balisage

Entité	Valeur	Exemple	Résultat
<	less than (<)	10 < 100	10<100
>	greater than (>)	x > 119	x>119
&	ampersand (&)	AT&T	AT&T
'	apostrophe (')	d'autres	d’autres
"	"	"Stop! "	"Stop!"

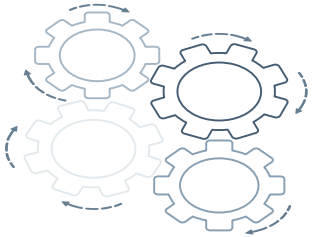
Entités prédéfinies et entités caractères

❑ Entités caractères :

Il n'existe pas d'entité prédéfinie pour les lettres accentuées. Il faut utiliser à la place les entités numériques du type &#n; (où n est une valeur décimale). La valeur numérique correspond au code ISO 10646 ;

Exemple:

é est codé par l'entité numérique é.

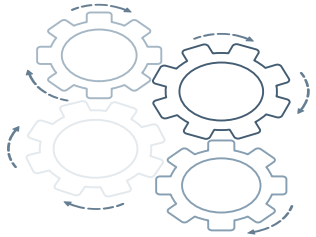


Commentaires

En XML, les commentaires se déclarent de la même façon qu'en HTML.
Ils commencent donc par <!-- et se terminent par -->.
Ils peuvent être placés à n'importe quel endroit tant qu'ils se trouvent à l'*extérieur d'une autre balise.*

Exemples de commentaires valides :

```
<!-- ceci est correct -->
<elt> <!-- ceci est correct aussi --> Un peu de texte </elt>
```



Entités prédéfinies et entités caractères

❑ Section *CDATA :*

déclarer un bloc de texte dont les caractères seront pris tel quel par le parseur XML.

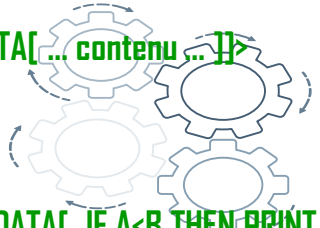
peut contenir toute sorte de chaîne de caractères.

permet de définir un bloc de caractères ne devant pas être analysés par le processeur XML

Permet de garder dans un bloc de texte un exemple de code à afficher tel quel

❑ Syntaxe: <![CDATA[… contenu …]]>

Exemple:



<![CDATA[IF A<B THEN PRINT A+B]]>

<![CDATA[Une balise commence par un < et se termine par un > .]]>

<![CDATA[<element> C'est un document XML</element>]]>

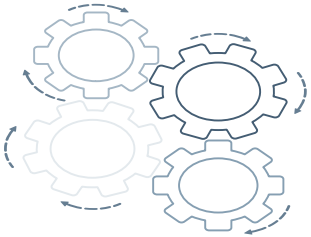
Entités prédéfinies et entités caractères

```
<program>
if(i<5) return i;
</program>
```

Incorrect

```
<program>
<![CDATA[if(i<5)
return i;]]>
</program>
```

Correct



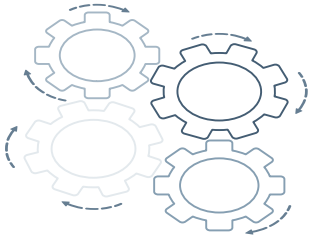
Entités prédéfinies et entités caractères

❑ Entités caractères :

Il n'existe pas d'entité prédéfinie pour les lettres accentuées. Il faut utiliser à la place les entités numériques du type &#n; (où n est une valeur décimale). La valeur numérique correspond au code ISO 10646 ;

Exemple:

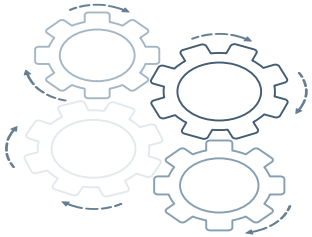
é est codé par l'entité numérique é.



DTD : Document Type Definition

- Permet de définir le «vocabulaire (les balises, les éléments, les attributs, les entités et notations utilisés) » et la structure qui seront utilisés dans le document XML
- Grammaire du langage dont les phrases sont des documents XML
- Elle peut être
  - absente.
  - incorporée au document XML (DTD *interne*)
  - mise dans un fichier à part (DTD *externe*).
- Lorsqu'il existe une DTD interne et une DTD externe: elles se combinent, la DTD externe complète la DTD interne.

La déclaration se place juste après le prologue du document



Pourquoi une DTD ?

- Pour spécifier la structure d'un document.
- Pour garantir qu'un document suit bien les règles d'une application spécifique (format d'échange de données).
- Partage d'une même structure entre plusieurs documents, (structures « standard » pour une communauté)
- Vérification stricte et automatisable de la correction des documents
- Pour dire ce qu'il est possible ou impossible de trouver dans un document. Ex : tel élément peut être suivi de tel autre.
- Pour spécifier que certains éléments doivent avoir des attributs obligatoires. Ex : tout élément employe doit avoir un attribut NSS (Numero de sécurité sociale).

Déclaration d'un Type DTD interne

La déclaration est faite à l'intérieur du document XML juste après le prologue du document.

Syntaxe

Exemple de DTD , Limites des DTD

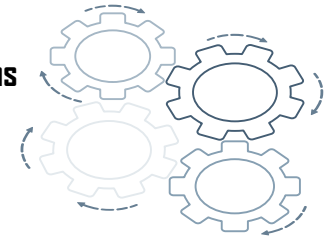
```
<!DOCTYPE ouvrage [
<!-- La DTD est décrite ici ..>
<!ELEMENT ouvrage ( titre, auteur, chapitre+)>
<!ELEMENT chapitre (titre, section*)>
<!ELEMENT section (titre, para+)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ATTLIST ouvrage date-publi CDATA #REQUIRED> ...
]>
```

❑ Limites des DTD

Les DTD ne sont pas écrites en XML, ce qui signifie que les technologies existantes pour manipuler des documents XML telles que DOM ou SAX ne peuvent être utilisées pour analyser des schémas de documents.

Les DTD ne supportent pas les espaces de nom ce qui rend impossible l'import de schémas externes afin de réutiliser du code existant.

Les DTD n'offrent qu'un typage très limité des données.

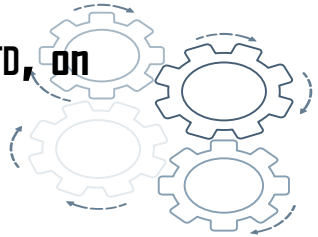


Apports du XML Schéma

- Le schéma est spécifié en XML : pas de nouveau langage.
- Le typage des données est introduit, ce qui permet la gestion de booléens, d'entiers, d'intervalles de temps…(un système de typage suffisamment riche pour importer/exporter des données d'une base de données.
- Possibilité de créer de nouveaux types complexes à partir de types existants.
- Permettre de définir des contraintes

Existence, obligatoire, optionnel, Domaines, cardinalités, références, Patterns, …etc

- Possibilité de créer de nouveaux type par héritage. Les éléments peuvent hériter du contenu et des attributs d'un autre élément.
- Le support des espaces de nom.
- Les indicateurs d'occurrences des éléments peuvent être tout nombre non négatif (dans une DTD, on est limité à 0, 1 ou un nombre infini d'occurrences pour un élément).
- Les schémas sont très facilement concevables par modules.



Structure d'un schémas XML

Un schéma XML est un document XML qui définit :

- les éléments possibles dans le document
- les attributs associés à ces éléments
- la structure du document et les types de données

Le schéma est spécifié en XML

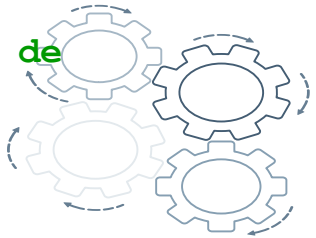
- pas de nouveau langage
- balisage de déclaration
- domaine spécifique xsd:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">

        <!-- Déclaration d'éléments -->
        <!-- Déclaration d'attributs -->
        <!-- définition de nouveaux types de
données -->
</xsd:schema>
```

L'élément racine

Espace de noms des schémas XML



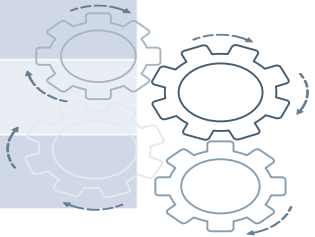
Structure d'un schémas XML

- Les éléments, attributs et les types peuvent être *globaux ou locaux*
- Ils sont globaux lorsque leur déclaration ou leur définition est un enfant direct de l'élément racine *xsd:schema*. Sinon, ils sont locaux
- Seuls les objets globaux peuvent être référencés dans tout le schéma pour être utilisés.

Attributs de l'élément xsd:schema

- Permettent de spécifier l'espace de noms cible du schéma et de préciser quelques comportements par défaut du schéma.

Noms de attributs	signification	Valeurs possibles
targetNamespace	l'URI qui identifie l'espace de noms cible (l'espace de noms des éléments et types définis par le schéma)	Si cet attribut est absent, les éléments et types définis n'ont pas d'espace de noms.
elementFormDefault	Les éléments seront qualifier par : ou non	qualified Unqualified (par defaut)
attributeFormDefault	Les attributs seront qualifier par : ou non	qualified Unqualified (par defaut)



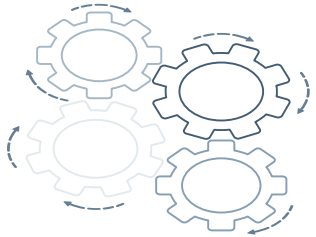
Les espaces de noms

- Un espace de nom permet d'éviter les conflits entres balises.
- Un espace de noms est identifié par un URI (Uniform Resource Identifier)

URI garantit que l'espace de noms soit identifié de manière unique

Déclaration :

- Un espace de noms est déclaré par un pseudo attribut dont le nom prend la forme <racine xmlns:prefix="URI">
- Le préfixe est ensuite utilisé pour qualifier les noms d'éléments.
- Un nom qualifié d'élément prend la forme prefix:NomElement où prefix est un préfixe associé à un espace de noms et NomElement est le nom local de l'élément.

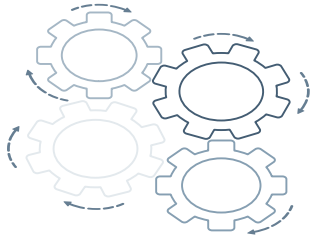


Les espaces de noms

Exemple :

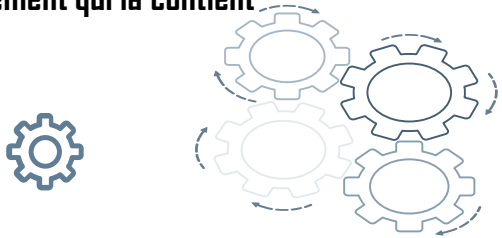
- Associer le préfixe hns à l'espace de noms de XHTML identifié par l'URI http://www.w3.org/1999/xhtml.
- tous les éléments de cet espace de noms sont préfixés par hns

```
<hns:html xmlns:hns="http://www.w3.org/1999/xhtml">
<hns:head>
        <hns:title>Espaces de noms</hns:title>
</hns:head>
<hns:body>
...
</hns:body>
</hns:html>
```



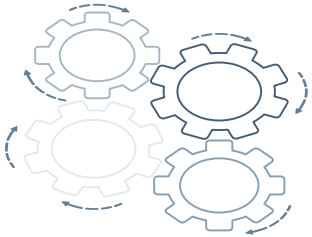
Les espaces de noms

- il est d'usage d'utiliser certains préfixes particuliers pour certains espaces de noms.
- html pour XHTML,
- xsd ou xs pour les schémas XML, xsl pour les feuilles de style XSLT
- Il est possible de déclarer plusieurs espaces de noms en utilisant plusieurs pseudo attributs de la forme xmlns:*prefix="URI"*
- La portée d'une déclaration d'un espace de noms est l'élément dans lequel elle est faite.
- la portée de la déclaration comprend les balises de l'élément qui la contient



Les espaces de noms

XML	http://www.w3.org/XML/1998/namespace
Schémas	http://www.w3.org/2001/XMLSchema
Instances de schémas	http://www.w3.org/2001/XMLSchema-instance
XSLT	http://www.w3.org/1999/XSL/Transform
XSL-FO	http://www.w3.org/1999/XSL/Format
MathML	http://www.w3.org/1998/Math/MathML
XHTML	http://www.w3.org/1999/xhtml
SVG	http://www.w3.org/2000/svg
XInclude	http://www.w3.org/2001/XInclude
Xlink	http://www.w3.org/1999/xlink

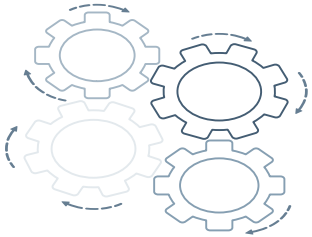


XPath

- Langage d'expressions de chemins permettant de sélectionner des parties (informations) d'un document XML.
- Opère sur l'arbre d'un document XML
- Objectif : référencer les nœuds (éléments, attributs, commentaires, ...etc) dans un document XML

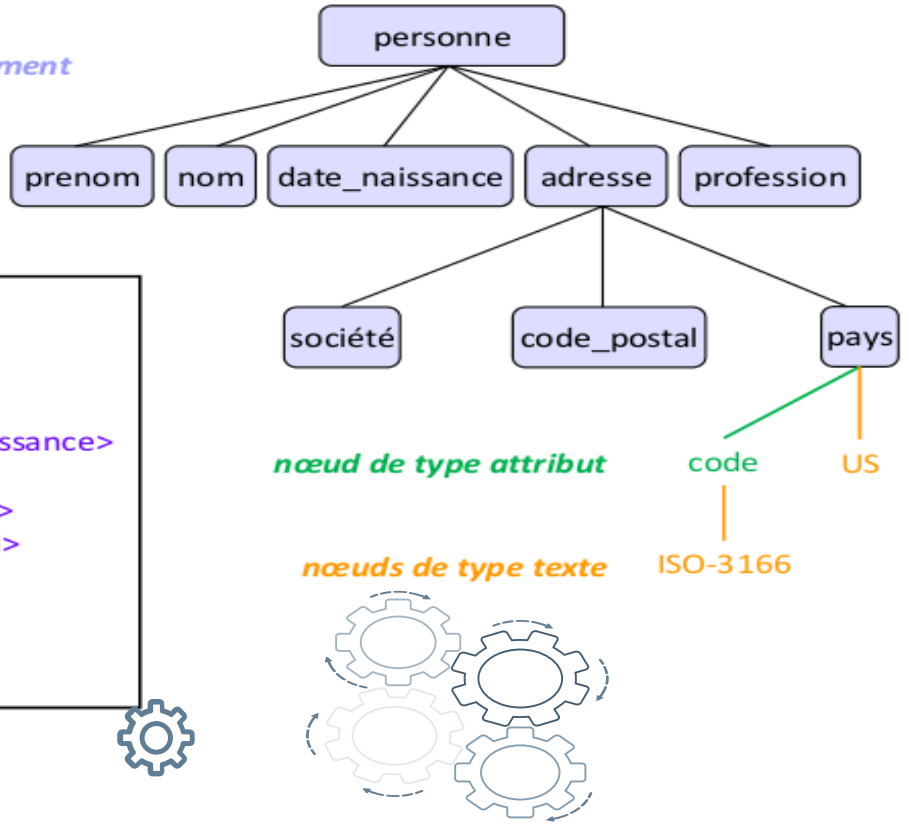
XPath est utilisé par :

- XSLT pour sélectionner des règles de transformation
- XML Schéma pour créer des clés et références
- XLink pour créer des liens entre documents/fragments XML
- XQUERY pour interroger des collections de documents



Arbre d'éléments

```xml
<?xml version="1.0" encoding="utf-8" ?>
<personne>
  <prenom>Donald</prenom>
  <nom>Knuth</nom>
  <date_naissance>1938-01-10</date_naissance>
  <adresse>
    <société>Stanford University</société>
    <code_postal>CA 94305</code_postal>
    <pays code = "ISO-3166">US</pays>
  </adresse>
  <profession>informaticien</profession>
</personne>
```



Types de nœuds dans l'arbre d'un doc XML

L'arbre d'un document XML est formé de nœuds de différents types :

Document : Nœud racine,

Unique et obligatoire

Distinct de l'élément racine du document

Element : nœud élément,

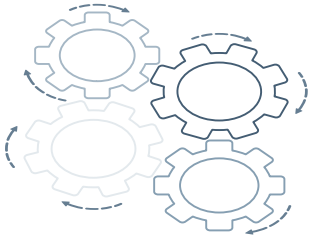
Attribut : nœud attribut

Text : nœud texte,

ProcessingInstruction : nœud instruction de traitement

Comment : nœud commentaire

Les nœuds sont ordonnés selon l'ordre de lecture des constituants du document XML



Propriétés des nœuds

Nom :

- Pour le nœud d'un élément, d'un attribut ou d'une instruction de traitement, le nom est le nom de l'élément, de l'attribut ou de l'instruction.
- Les nœuds des commentaires et les nœuds textuels n'ont pas de nom.
- Le nœud racine n'a également pas de nom.

Parent :

- Tout nœud à l'exception du nœud racine a un parent qui est soit le nœud racine soit le nœud de l'élément qui le contient.
- La relation parent/enfant n'est pas symétrique en XML. Bien que le parent d'un attribut soit l'élément qui le contienne, l'attribut n'est pas considéré comme un enfant de cet élément

Enfants :

Seuls le nœud racine et les nœuds des éléments peuvent avoir des enfants.

Les enfants d'un nœud sont les nœuds des éléments, des instructions de traitement et des commentaires ainsi que les nœuds textuels qu'il contient.

Les attributs ne font pas partie des enfants d'un élément.

L'élément racine est un enfant du nœud racine (document) qui peut aussi avoir des instructions de traitement et des commentaires comme enfants.

Propriétés des nœuds

valeur textuelle :

- Chaque nœud de l'arbre du document XML a une valeur qui est une chaîne de caractères.
- Pour un élément, cette valeur textuelle est le résultat de la concaténation des contenus de tous les nœuds textuels qui sont descendants du nœud. Ce texte comprend donc tout le texte contenu dans le nœud, y compris le texte contenu dans ses descendants.

Type :

- Cette propriété n'existe que si le document a été un validé par un schéma.
- C'est le type attribué à l'élément ou à l'attribut lors de la validation par le schéma
- Ce type peut être un type prédéfini ou un type défini dans le schéma

valeur typée :

- Cette valeur n'existe que si le document a été validé par un schéma
- C'est la valeur obtenue en convertissant la valeur textuelle dans le type du nœud

Exemple : Si le type d'un attribut est, par exemple, xsd:double et sa valeur textuelle est la chaîne 1.0e-3, sa valeur est le nombre flottant 0.001

Expression de chemins

- Une expression de chemins spécifie une traversée de l'arbre du document en une suites d'étapes :

  [/]étape1 /étape2 /.../étapeN

  - depuis un nœud de départ vers un ensemble de nœuds cibles
  - les cibles constituent la valeur du cheminement

- Il existe deux types d'expressions Xpath:

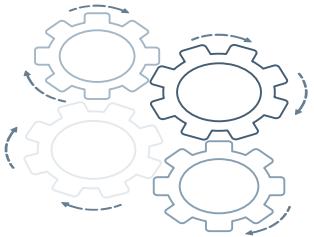
  - absolu
    - commence à la racine
    - /étape1/.../étapeN
    - /recette/ingredients/ingredient/nom

  - relatif

commence à un nœud courant(contexte)

étape1/.../étapeN

ingredient/nom

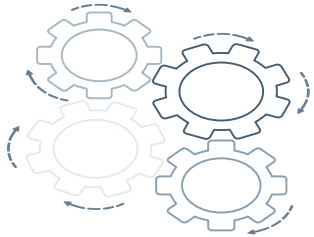


Les axes de localisation (direction ou sens de parcours)

- Un axe sélectionne dans l'arbre du document et à partir du nœud contexte,

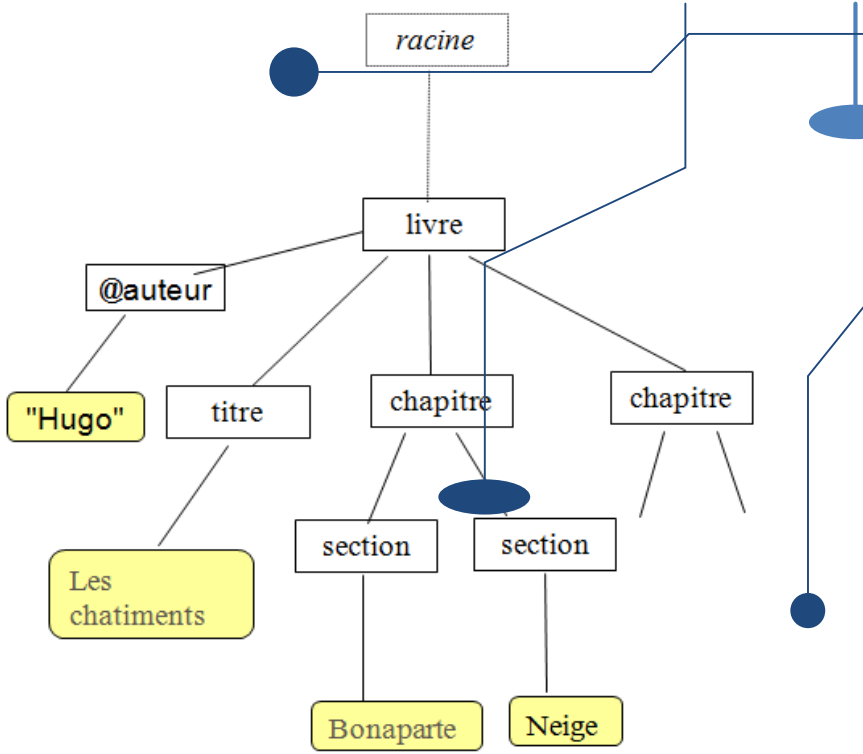
  l'ensemble des nœuds qui peuvent être atteints en suivant une certaine direction :

  - self
  - child
  - attribute
  - parent
  - descendant
  - descendant-or-self
  - ancestor
  - ancestor-or-self
  - following
  - following-sibling
  - preceding
  - preceding-sibling



Expression de chemins

- **Sections d'un chapitre**
  - /child::livre/child::chapitre / child::section
  - /livre/chapitre/section
- **Texte du chapitre 1 section 2**
  - /descendant::chapitre[position() = 1] /child::section[position() = 1] /child::text()
  - //chapitre[0]/section[1]/text( )



CONCLUSION

- XML est un langage de balisage puissant et largement utilisé qui facilite l'échange de données et favorise l'interopérabilité entre les applications et les systèmes

- Comprendre la structure et les avantages de XML est essentiel pour les développeurs et les professionnels de l'informatique dans le monde axé sur la technologie d'aujourd'hui.