Technologies XML et Web Sémantique

ENSA

جامعة شعيب الدكالي

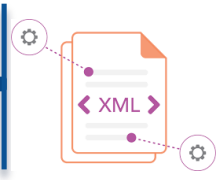
Université Chouaïb Doukkali

Présenté par: Pr. Fahd KALLOUBI

Organisation du cours

- Cours magistral
  - Présentation de XML et concepts de base
  - Syntaxe de XML
  - Définir le format d'un document: DTD
  - Prise en main d'oXygen XML Editor
  - Définir le format d'un document: Schémas XML
  - XSLT
  - XPath
  - XQuery
  - Présentation de DOM (Document Object Model)
  - Présentation de SAX (Simple API for XML)
  - XML et SGBD
  - eXist: un SGBD XML

Organisation du cours

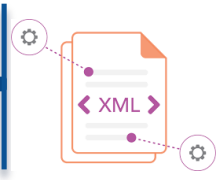


- Travaux pratiques
  - Les étudiants sont amenés à résoudre des problématiques en utilisant les différents outils et technologies vus dans le cours

- Langages de programmation et outils

- Note finale
  - 40% Travaux pratiques et Contrôle
  - 60% Examen

Introduction à XML

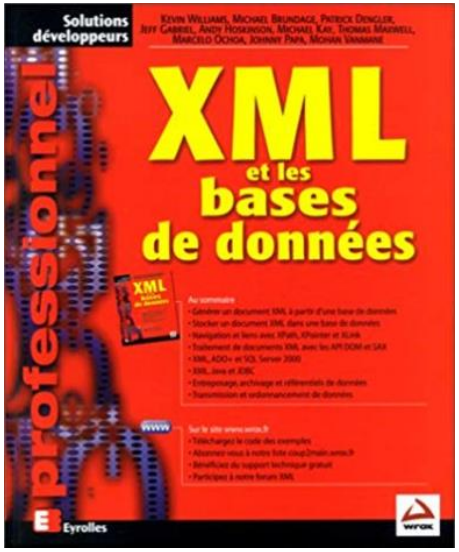


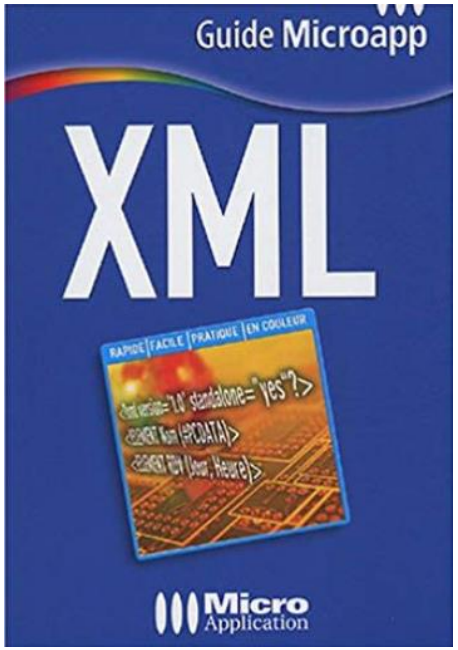
- XML = eXtensible Markup Language
  - C'est un langage permettant de représenter et structurer des informations à l'aide de balises que chacun peut définir et employer comme il le veut.

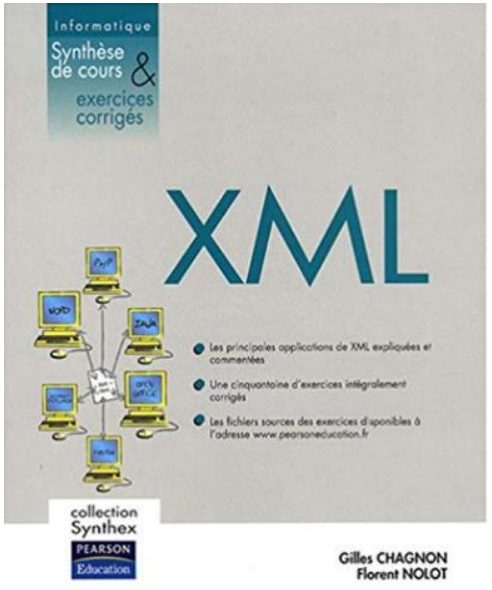
texte ... <BALISE> ... texte </BALISE> ...

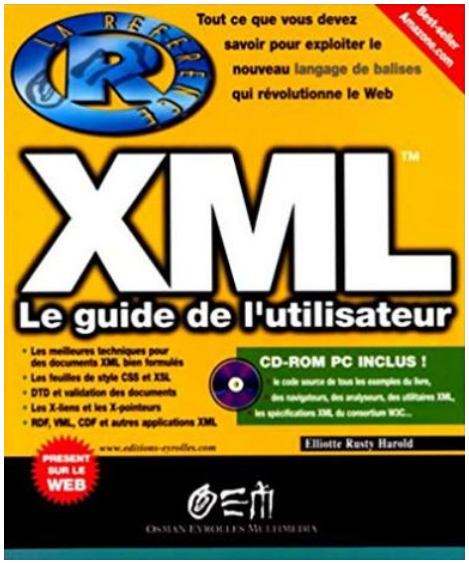
- Les objectifs de ce cours:
  - Présenter les concepts de base
  - Introduire quelques applications typiques
  - Présenter l'historique d'XML
  - Définir la structure d'un document XML

Littérature – quelques sources









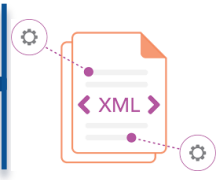
Introduction à XML: exemple

- Un fichier XML représente des informations structurées:

```
<?xml version="1.0" encoding="utf-8"?>
<!– Filière fictive -->
<Filiere>
    <cours heure="200">XML</cours>
    <cours heure="140">Web sémantique</cours>
    <cours heure="220">Machine learning</cours>
</Filiere>
```

- Cet exemple modélise une filière composée d'un ensemble de cours avec leur masse horaire

Applications XML



- Le format XML est au cœur de nombreux processus actuels :
  - format d'enregistrement de nombreuses applications,
  - échange de données entre serveurs et clients,
  - outils et langages de programmation,
  - bases de données.
- Il faut comprendre que XML définit la syntaxe des documents, mais les applications définissent les balises et ce qu'elles doivent contenir. Des API existent dans tous les langages pour lire et écrire des documents XML.

XML en tant que format de fichier

XML est la norme d'enregistrement de nombreux logiciels parmi lesquels on peut citer :

- Bureautique
  - LibreOffice : format OpenDocument
  - Publication de livres et documentations : DocBook
- Graphismes
  - Dessin vectoriel avec Inkscape : format SVG
  - Équations mathématiques : format MathML
- Programmation
  - Interfaces graphiques : XUL, Android
  - Construction/compilation de projets : Ant, Maven
- Divers
  - Itinéraires GPS : format GPX

Base de données XML

XML permet aussi de représenter des données complexes:

- Web sémantique : c'est un projet qui vise à faire en sorte que toutes les connaissances présentes plus ou moins explicitement dans les pages web puissent devenir accessibles par des mécanismes de recherche unifiés. Pour cela, il faudrait employer des marqueurs portant du sens en plus de marqueurs de mise en page, par exemple rajouter des balises indiquant que telle information est le prix unitaire d'un article.

- Bases de données XML native : les données sont au format XML et les requêtes sont dans un langage (XQuery) permettant de réaliser l'équivalent de SQL

XML: échange de données

XML est le format utilisé pour représenter des données volatiles de nombreux protocoles dont :

- Les flux RSS permettent de résumer les changements survenus sur un site Web. Par exemple, un site d'information émet des messages RSS indiquant les dernières nouvelles

- Les protocoles XML-RPC et SOAP (Simple Object Access Protocol) permettent d'exécuter des procédures à distance (Remote Procedure Call) : un client demande à un serveur d'exécuter un programme ou une requête de base de donnée puis attend les résultats. Le protocole gère l'encodage de la requête et des résultats, indépendamment des langages de programmation et des systèmes employés.

- Autre exemple : AJAX permet de mettre à jour dynamiquement les pages web sur le navigateur.

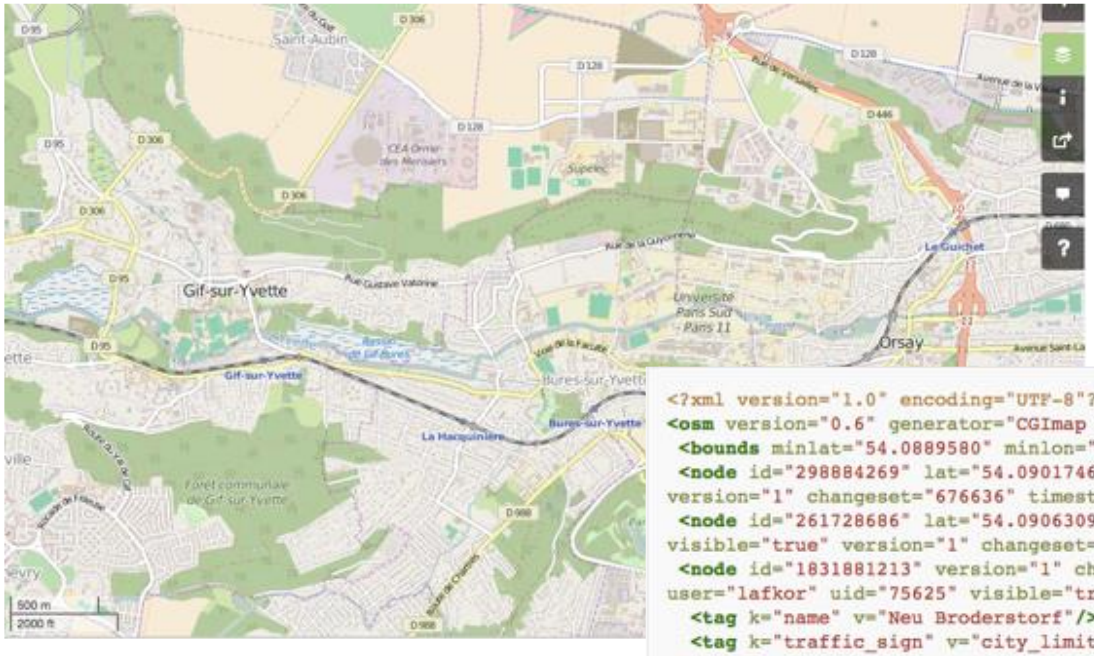
XML: Exemples d'application


```
Theory of Computation
Michael Sipser
Cengage Learning
2012
3

Artificial Intelligence
Stuart Russell
Peter Norvig
Pearson
2013
3
```

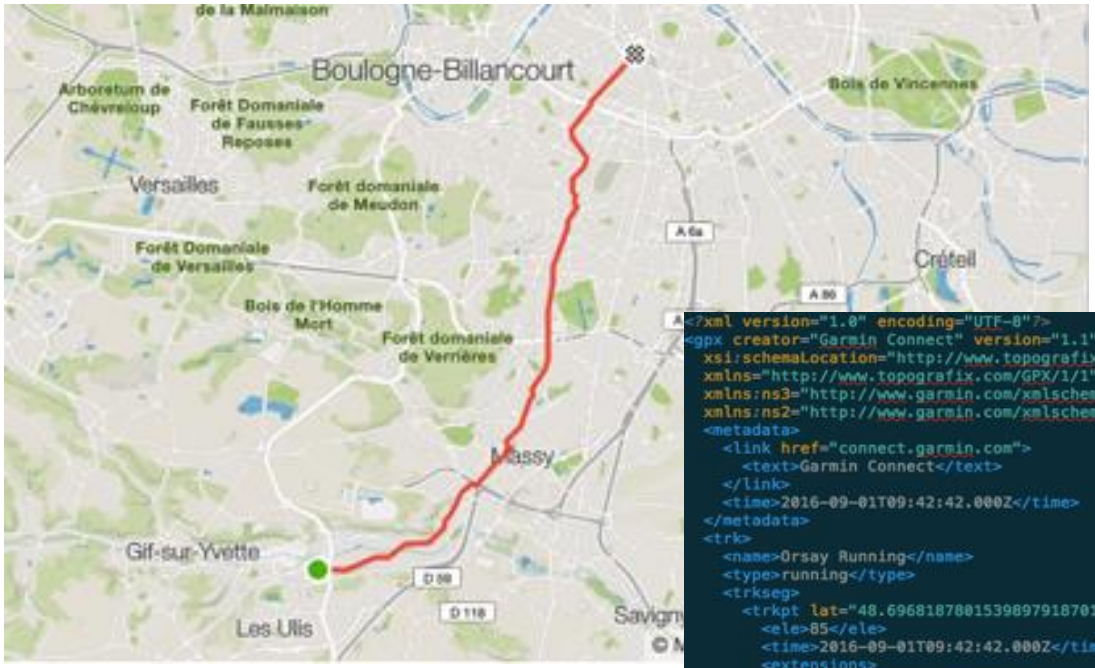
```xml
<?xml version="1.0" encoding="UTF-8"?>
<books>
  <book>
    <title>Theory of Computation</title>
    <author>Michael Sipser</author>
    <publisher>Cengage Learning</publisher>
    <year>2012</year>
    <edition>3</edition>
  </book>
  <book>
    <title>Artifical Intelligence</title>
    <author>Peter Norvig</author>
    <author>Stuart Russell</author>
    <year>2013</year>
    <edition>3</edition>
    <publisher>Pearson</publisher>
  </book>
</books>
```

XML: Exemples d'application (Open Maps)



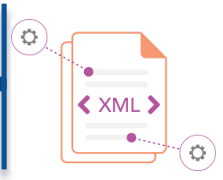

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2">
 <bounds minlat="54.0889580" minlon="12.2487570" maxlat="54.0913900" maxlon="12.2524800"/>
 <node id="298884269" lat="54.0901746" lon="12.2482632" user="SvenHRO" uid="46882" visible="true"
version="1" changeset="676636" timestamp="2008-09-21T21:37:45Z"/>
 <node id="261728686" lat="54.0906309" lon="12.2441924" user="PikoWinter" uid="36744"
visible="true" version="1" changeset="323878" timestamp="2008-05-03T13:39:23Z"/>
 <node id="1831881213" version="1" changeset="12370172" lat="54.0900666" lon="12.2539381"
user="lafkor" uid="75625" visible="true" timestamp="2012-07-20T09:43:19Z">
  <tag k="name" v="Neu Broderstorf"/>
  <tag k="traffic_sign" v="city_limit"/>
 </node>
 ...
 <node id="298884272" lat="54.0901447" lon="12.2516513" user="SvenHRO" uid="46882" visible="true"
version="1" changeset="676636" timestamp="2008-09-21T21:37:45Z"/>
 <way id="26659127" user="Masch" uid="55988" visible="true" version="5" changeset="4142606"
timestamp="2010-03-16T11:47:08Z">
  <nd ref="292403538"/>
  <nd ref="298884289"/>
  ...
  <nd ref="261728686"/>
  <tag k="highway" v="unclassified"/>
  <tag k="name" v="Pastower Straße"/>
 </way>
 <relation id="56688" user="kmvar" uid="56190" visible="true" version="28" changeset="6947637"
timestamp="2011-01-12T14:23:49Z">
  <member type="node" ref="294942404" role=""/>
  ...
  <member type="node" ref="364933006" role=""/>
  <member type="way" ref="4579143" role=""/>
  ...
```

XML: Exemples d'application (données GPS)




```xml
<?xml version="1.0" encoding="UTF-8"?>
<gpx creator="Garmin Connect" version="1.1"
  xsi:schemaLocation="http://www.topografix.com/GPX/1/1 http://www.topografix.com/GPX/11.xsd"
  xmlns="http://www.topografix.com/GPX/1/1"
  xmlns:ns3="http://www.garmin.com/xmlschemas/TrackPointExtension/v1"
  xmlns:ns2="http://www.garmin.com/xmlschemas/GpxExtensions/v3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <metadata>
    <link href="connect.garmin.com">
      <text>Garmin Connect</text>
    </link>
    <time>2016-09-01T09:42:42.000Z</time>
  </metadata>
  <trk>
    <name>Orsay Running</name>
    <type>running</type>
    <trkseg>
      <trkpt lat="48.696818780153989791870117 1875" lon="2.187100304290652275085449218 75">
        <ele>85</ele>
        <time>2016-09-01T09:42:42.000Z</time>
        <extensions>
          <ns3:TrackPointExtension>
            <ns3:hr>94</ns3:hr>
            <ns3:cad>61</ns3:cad>
          </ns3:TrackPointExtension>
        </extensions>
      </trkpt>
      <trkpt lat="48.696818109601736068725585937 5" lon="2.187121678143739700317382812 5">
        <ele>85</ele>
        <time>2016-09-01T09:42:43.000Z</time>
        <extensions>
          <ns3:TrackPointExtension>
            <ns3:hr>94</ns3:hr>
            <ns3:cad>61</ns3:cad>
          </ns3:TrackPointExtension>
        </extensions>
      </trkpt>
      <trkpt lat="48.696820372715592384338378906 25" lon="2.187208095565438270568847656 25">
        <ele>84.80000305175781 25</ele>
        <time>2016-09-01T09:42:46.000Z</time>
        <extensions>
          <ns3:TrackPointExtension>
            <ns3:hr>97</ns3:hr>
            <ns3:cad>61</ns3:cad>
          </ns3:TrackPointExtension>
        </extensions>
      </trkpt>
      <trkpt lat="48.696838226169347763061523437 5" lon="2.187277749180793762207031 25">
        <ele>84.59999847412109375</ele>
        <time>2016-09-01T09:42:48.000Z</time>
```

XML: quelques concepts importants

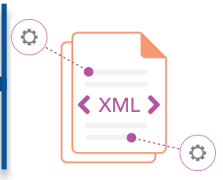


- Méta langage à balises
- Données et balises sont des chaines de caractères
- La syntaxe est stricte
- Langage à balises structurel et sémantique
- Les marqueurs définissent la sémantique du document
- Ce n'est pas un langage de présentation (HTML)
- XML promet de standardiser la manière dont l'information est :
  - échangée (XML)
  - personnalisée/présentée (XSL/CSS)
  - recherchée (XPath/XSLT/XQuery)
  - sécurisée (Encryption, Signature)
  - liée (XLink)

Évolution de XML

- 1969 : GML (IBM) GML (**G**eneralized **M**arkup **L**anguage) est un langage d'écriture de documents techniques, défini par IBM, destiné à être traité par un logiciel propriétaire de mise en page appelé SCRIPT/VS.
- 1990 : SGML et HTML SGML (**S**tandard **G**eneralized **M**arkup **L**anguage) est une norme libre très complexe dont HTML est un sous-ensemble très spécialisé.
  - restreint a la présentation des pages Web
  - ne peut servir a l'échange de données entre bases hétérogènes
- 1998 : XML c'est une généralisation de SGML permettant de construire toutes sortes de documents.
  - conserve une grande partie de la puissance de SGML
  - élimine les caractéristiques redondantes, compliquées a implanter et qui n'ont montre aucun intérêt après plus de 20 ans

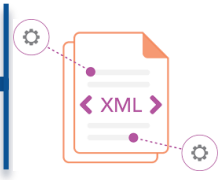
Une famille de technologies XML (1/2)



XML 1.0 est un «SGML simplifié», diverses applications XML:

- e**X**tensible **S**tylesheet **L**anguage
  - **XSL**Transformation : transformation d'un document XML en un autre
  - **XSL-F**ormatting **O**bject : décrit la composition de pages destinées a l'impression (rival de Postscript)
- **C**ascading **S**tyle**S**heet (non XML) : presentation de documents XML
- e**X**tensible **L**inking **L**anguage
  - XLink : décrit la relation entre documents XML
  - XPointer : identifie une partie de document XML
- XPath (non XML) : cibler un ou des éléments d'un document XML

Une famille de technologies XML (2/2)



- XML Schemas : description structurelle et sémantique a laquelle un document XML doit se conformer (DTD++)
- **S**imple **A**PI for **X**ML : API que tout parseur XML doit respecter
- **D**ocument **O**bject **M**odel : API permettant de manipuler un document XML
- XML Query Language, XHTML, SMIL, SVG, . . .

HTML vs. XML

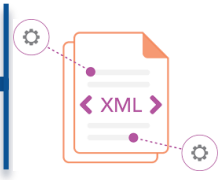

```
<p> Fahd Kalloubi </p>
<address>
Ensa El Jadida<br>
Av. Jabrane Khalil Jabrane<br>
BP 299<br>
24000<br>
Université Chouaib Doukkali<br>
</address>
```

- Balises et sémantiques associées sont prédéfinies
- mélange de structurations logique et physique
- perte du sens

```
<enseignant corps="Professeur d'enseignement Sup.">
<prenom>Fahd</prenom><nom>Kalloubi</nom>
<adresse>
<structure>Ensa El Jadida</structure>
<bp>BP299</bp>
<Av. Jabrane Khalil Jabrane</rue>
<cp>24000</cp>
<ville>El Jadida</ville>
</adresse>
</enseignant>
```

- extensibilité du langage
- structuration logique
- représentation physique déléguée (CSS, XSL)
- modularité et réutilisation des structures
- facilite l'accès à des sources de données hétérogènes

Document bien formé et valide



XML distingue 2 classes de documents:

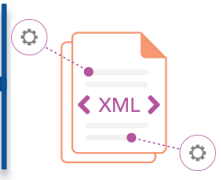
1. Les documents **bien formés** obéissent à la syntaxe du langage XML
2. Les documents **valides** sont bien formés et obéissent à une structure, il doit être:
   - « well-formed » (formé correctement)
   - Être associé à une DTD (ou une autre grammaire)
   - Et être conforme à cette DTD ou une grammaire d'un autre type comme XSD (XML Schema Definition) ou RelaxNG

Tout document valide peut être distribué sans sa DTD (ou référence a sa DTD), il apparaitra alors comme bien formé aux utilisateurs (humains ou electroniques : navigateur).

Structure d'un document XML

- Un prologue (facultatif mais fortement conseillé)
  - Une déclaration XML
  - Des instructions de traitements (utilisées par les moteurs, les navigateurs)
  - Une déclaration de type de document
- Un arbre d'éléments basé sur un élément appelé racine
  - Un élément possède un nom, des attributs et un contenu
  - Le contenu d'un élément est :
    - ✓ Du texte
    - ✓ d'autres éléments (les éléments enfants).
  - Un élément est marqué par une balise ouvrante et une balise fermante.
    - ✓ une balise ouvrante est notée <nom attributs...>
    - ✓ une balise fermante est notée </nom>
- Des commentaires (facultatif)

Structure d'un document XML: exemple



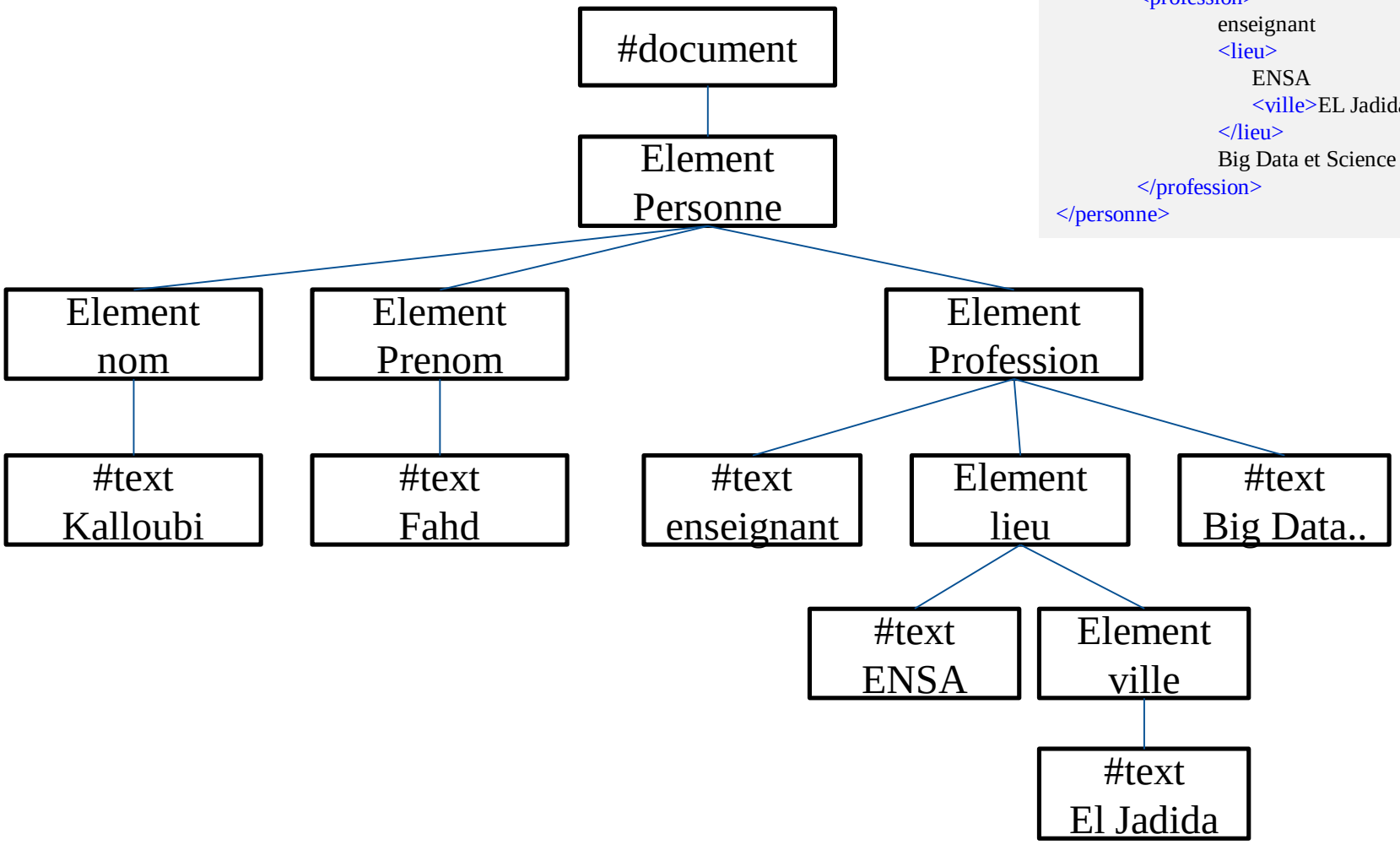
Voici un document XML qui représente une personne :

```
<?xml version="1.0" encoding="utf-8"?>
<personne>
    <nom>Kalloubi</nom>
    <prenom>Fahd</prenom>
    <profession>
        enseignant
        <lieu>
            ENSA
            <ville>EL Jadida</ville>
        </lieu>
        Big Data et Science de données
    </profession>
</personne>
```

Structure d'un document XML: exemple

Dont voici la représentation graphique :

```
<?xml version="1.0" encoding="utf-8"?>
<personne>
        <nom>Kalloubi</nom>
        <prenom>Fahd</prenom>
        <profession>
                enseignant
                <lieu>
                    ENSA
                    <ville>EL Jadida</ville>
                </lieu>
                Big Data et Science de données
        </profession>
</personne>
```



Structure d'un document XML: explications


```
<?xml version="1.0" encoding="utf-8"?>
<personne>
        <nom>Kalloubi</nom>
        <prenom>Fahd</prenom>
        <profession>
                enseignant
                <lieu>
                    ENSA
                    <ville>EL Jadida</ville>
                </lieu>
                Big Data et Science de données
        </profession>
</personne>
```

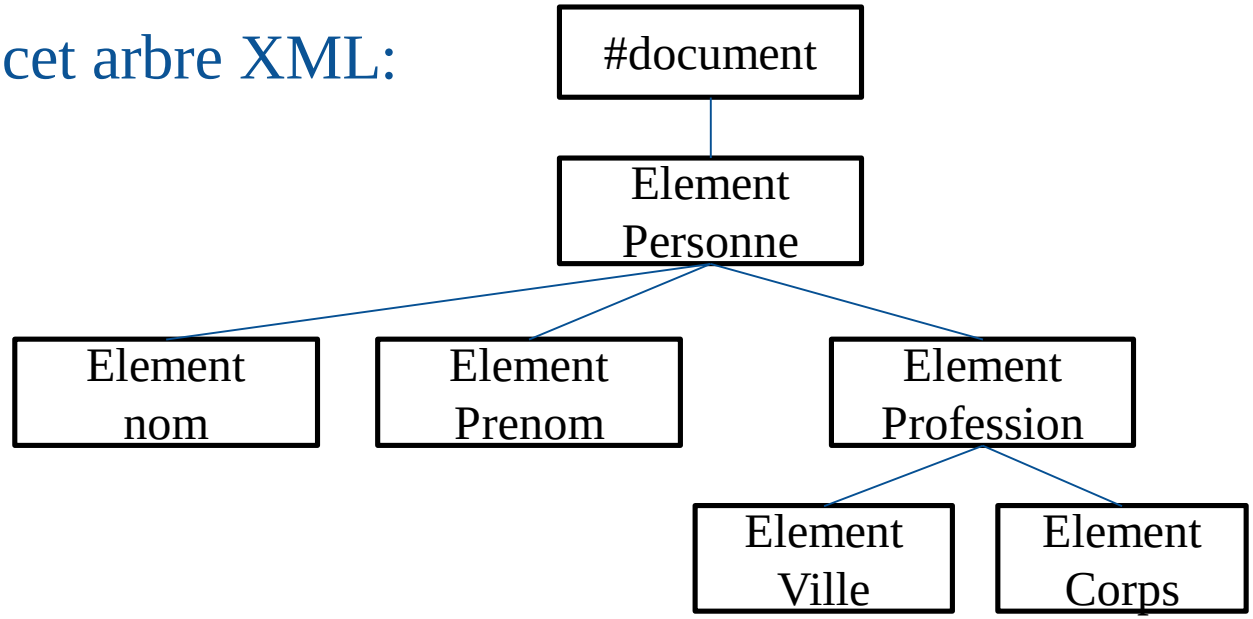
Le document XML représente un arbre composé de plusieurs types de noeuds :

- nœuds éléments ils sont associés aux balises <element>. Ce sont des nœuds qui peuvent avoir des enfants en dessous.
- nœuds #text ils représentent le texte situé entre deux balises. Les nœuds texte sont des feuilles dans l'arbre. Notez que différents textes peuvent être entrelacés avec des éléments. Voir le cas de <lieu> dans le contenu de <profession>.

Les autres types de nœuds seront présentés au fur et à mesure.

Structure d'un document XML: vocabulaires

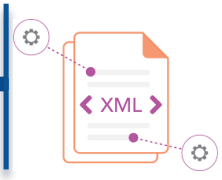
Soit cet arbre XML:



Voici comment on désigne les différents nœuds les uns par rapport aux autres :

- <Personne> (nœud racine): est le nœud parent du nœud enfant (child) <Profession>, lui-même parent de <ville> et <Corps>,
- <Personne>, <Profession> sont des nœuds ancêtres (ancestors) de <ville> et <Corps>
- <ville> et <Corps> sont des descendants (descendants) de <Personne> et <Profession>,
- <nom> est un nœuds frère (sibling) de <Prenom> et réciproquement

Détails du format XML: le prologue



<?xml version="1.0" encoding="ISO-8859-1"
standalone="yes" ?>

la version du langage XML utilise dans le document (1.0)

- La norme 1.1 est plus claire concernant certains nouveaux caractères unicode qui peuvent spécifier des retours à la ligne et autres caractères de contrôle.

le codage de caractères (charset) utilise :

sont autorises les jeux de caractères définis dans la norme ISO 10646, les parseurs doivent traiter au moins les codages UTF-8 ou UTF-16

- en l'absence de déclaration, ils s'attendent a de l'UTF
- si un autre codage est utilise il faut le spécifier

l'existence de déclaration extérieure au document :

- si le document est autonome : "yes"
- si le document nécessite la récupération de données externes : "no"

Cette déclaration est facultative mais fortement conseillée

Détails du format XML : Norme Unicode

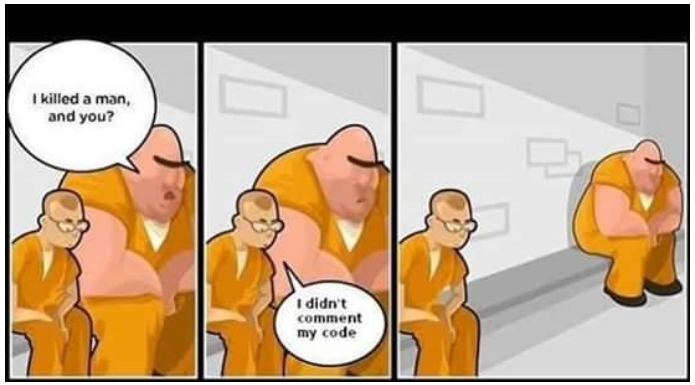
La norme Unicode définit un ensemble de plus de 245000 caractères représentable sur un ordinateur, par exemple é, «, ©. . . Ces caractères doivent être codés sous forme d'octets. C'est là qu'il y a plusieurs normes :

- ASCII ne représente que les 128 premiers caractères Unicode.
- ISO 8859-1 ou Latin-1 représente 191 caractères de l'alphabet latin n°1 (ouest de l'Europe) à l'aide d'un seul octet.
- ISO 8859-15 est une norme mieux adaptée au français.
- UTF-8 représente les caractères à l'aide d'un nombre variable d'octets, de 1 à 4 selon le caractère. Par exemple : A est codé par 0x41, é par 0xC3,0xA9.

Détails du format XML : les commentaires

On peut placer des commentaires à peu près partout dans un document XML. La syntaxe est identique à celle d'un fichier HTML. Un commentaire peut s'étendre sur plusieurs lignes. La seule contrainte est de ne pas pouvoir employer les caractères -- dans le commentaire, même s'ils ne sont pas suivis de > :

```
<!-- voici un commentaire -->
ceci n'est pas un commentaire
<!--
voici un autre commentaire
et ça -- , c'est une erreur
-->
```



Détails du format XML : les commentaires

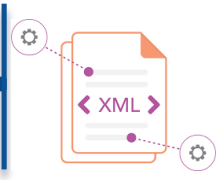
En fait, en XML (comme en SGML), <!-- et --> ne sont pas des marques de commentaires comme /* et */ en C:

- <! marque le début d'une instruction de traitement destinée à l'analyseur
- > signale la fin de cette instruction de traitement
- -- inverse le mode « hors commentaire » ou « dans un commentaire ».

Ainsi, un commentaire <!--commentaire--> est analysé ainsi :

1. <! début d'une instruction de traitement
2. -- passage en mode commentaire
3. commentaire : ignoré, on peut donc tout y mettre sauf –
4. -- sortie du mode commentaire, ne rien mettre après
5. > sortie de l'instruction de traitement

Détails du format XML : Options après le prologue



Après le prologue, on peut trouver plusieurs parties optionnelles délimitées par <?...?> ou <!...>. Ce sont des instructions de traitement, des directives pour l'analyseur XML

Par exemple:

- une Document Type Definitions (DTD) qui permet de valider le contenu du document (voir le prochain cours).
- une feuille de style,

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE personne SYSTEM "personne.dtd">
<?xml-stylesheet href="personne.css" type="text/css"?>
<personne>
...
</personne>
```

Détails du format XML : les éléments

Les éléments définissent la structure du document. Un élément est délimité par :

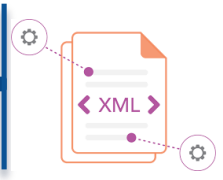
- une balise ouvrante <nom attributs...>
- une balise fermante </nom> obligatoire.

Le contenu de l'élément se trouve entre les deux balises. Ce sont des éléments enfants et/ou du texte.

```
<parent>
texte1
<enfant>texte2</enfant>
texte3
</parent>
```

Dans le cas où l'élément n'a pas de contenu (élément vide), on peut regrouper ses deux balises en une seule <nom attributs.../>

Détails du format XML : choses interdites



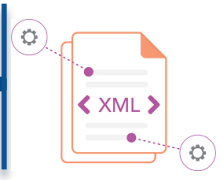
Les règles d'imbrication XML interdisent différentes configurations qui sont plus ou moins tolérées en HTML :

- plusieurs racines dans le document,
- des éléments non terminés (NB: XML est sensible à la casse),
- des éléments qui se chevauchent.

```
<element1>
      <element2>
      </element2>
      <element3>
      </element1>
</element3>
```

En XML, cela crée des erreurs : document mal formé.

Détails du format XML : choses permises

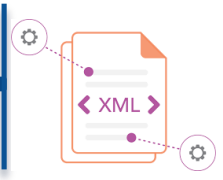


Parmi les choses permises, le même type d'élément peut se trouver plusieurs fois avec le même parent, avec des contenus identiques ou différents :

```
<element1>
    <element2>texte1</element2>
    <element2>texte2</element2>
    <element2>texte1</element2>
</element1>
```

Il est possible d'interdire cela, cependant ce n'est pas relatif à la syntaxe XML mais à la validation du document par rapport à une Spécification (voir le prochain cours sur les DTD et les schémas).

Détails du format XML : Noms des éléments



Les noms des éléments peuvent employer de nombreux caractères Unicode (correspondant au codage déclaré dans le prologue) mais pas les signes de ponctuation:

- L'initiale doit être une lettre ou le signe _
- ensuite, on peut trouver des lettres, des chiffres ou - . _

Le caractère : permet de séparer le nom en deux parties : préfixe et nom local. Le tout s'appelle nom qualifié. Par exemple ensa:departement est un nom qualifié préfixé par ensa. Ce préfixe permet de définir un espace de nommage.

nom qualifié = préfixe:nom local

Détails du format XML : Espace de nommage

Un espace de nommage (namespace) définit une famille de noms afin d'éviter les confusions entre des éléments qui auraient le même nom mais pas le même sens. Cela arrive quand le document XML modélise les informations de plusieurs domaines.

Le document suivant modélise une table (avec 4 pieds) et aussi un tableau HTML pour afficher ses dimensions.

```
<meuble id="765">
<table prix="74,99e">acajou</table>
<table border="1">
<tr><th>longueur</th><th>largeur</th></tr>
<tr><td>120cm</td><td>80cm</td></tr>
</table>
</meuble>
```

Alors, il est pratique d'employer des espaces de nommage quand on mélange au sein du même document XML des instructions de traitement et des données. C'est le cas dans les feuilles de style XSLT (plus loin dans ce cours)

On voit très bien la confusion

Détails du format XML : définition d'un espace de nommage

On doit choisir un URI, par exemple un URL, pour identifier l'espace de nommage. Cet URI n'a pas besoin de correspondre à un véritable contenu car il ne sera pas téléchargé.

Un URI est la généralisation d'un URL :
schéma:[//[user:passwd@]hôte[:port]][/]chemin[?requête][#fragment],
par exemple
http://en.wikipedia.org/wiki/Uniform_Resource_Identifier#Syntax.

Les URN sont au format urn:NID:NSS, avec:

- urn est la méthode d'URI des URN
- NID (Namespace Identifier) est un identificateur d'espace de noms
- NSS (Namespace specific String) est la partie spécifique à l'espace de noms identifié par le NID

Ensuite on rajoute un attribut spécial à la racine du document :

```
<?xml version="1.0" encoding="utf-8"?>
<racine xmlns:PREFIXE="URI">
<PREFIXE:balise>...</PREFIXE:balise>
</racine>
```

Détails du format XML : espace de nommage (exemple)

Voici l'exemple précèdent revisité en éliminant la confusion avec deux namespace, un URN pour les meubles et un URL pour HTML

```
<?xml version="1.0" encoding="utf-8"?>
<meuble:meuble id="765"
        xmlns:meuble="urn:ensael:meubles"
        xmlns:html="http://www.w3.org">
    <meuble:table prix="74,99e">acajou</meuble:table>
    <html:table border="1">
      <html:tr><html:th>longueur</html:th>...</html:tr>
      <html:tr><html:td>120cm</html:td>...</html:tr>
    </html:table>
</meuble:meuble>
```

Détails du format XML : Namespace par défaut

Lorsque la racine du document définit un attribut xmlns="URI", alors par défaut toutes les balises du document sont placées dans ce namespace, donc pas besoin de mettre un préfixe.

```
<?xml version="1.0" encoding="utf-8"?>
<book xmlns="http://docbook.org/ns/docbook">
```

Ça peut s'appliquer également localement à un élément et ça concerne toute sa descendance :

```
<meuble id="765" xmlns="urn:ensael:meubles">
<table prix="74,99e">acajou</table>
<table border="1" xmlns="http://www.w3.org">
<tr><th>longueur</th>...</tr>
<tr><td>120cm</td>...</tr>
</table>
</meuble>
```

Détails du format XML : Les attributs

Les attributs caractérisent un élément. Ce sont des couples nom="valeur" ou nom='valeur'. Ils sont placés dans la balise ouvrante.

```
<?xml version="1.0" encoding="utf-8"?>
<meuble id="765" type='table'>
<prix monnaie='$'>74,99</prix>
<dimensions unites="cm" longueur="120" largeur="80"/>
<description langue='fr'>Belle petite table</description>
</meuble>
```

Remarque:

- Il n'y a pas d'ordre entre les attributs d'un élément,
- Un attribut ne peut être présent qu'une fois.

Détails du format XML : Entités

- Certains caractères sont interdits dans le contenu des éléments. Comme il est interdit de les employer
  - XML propose une représentation appelée *référence d'entité* ou *entité* pour simplifier
  - Les entités ont été prédéfinies afin de pouvoir utiliser les caractères réservés
  - Une entité est une chaîne de caractère commençant par & et se terminant par ; &entite;
  - Une entité est remplacée par la chaîne de caractère qu'elle représente.

Caractère	Entité
&	&amp;
<	&lt;
>	&gt;
"	&quot;
'	&aquot;

Détails du format XML : Entités (exemple)

Voici un exemple d'emploi d'entités:

Prologue

Racine

Elément

Attribut

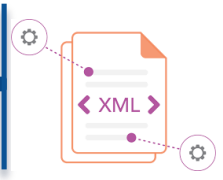
Entité

```
<?xml version="1.0" encoding="utf-8«  standalone="yes"?>
<MOTEURS>
<MOTEUR marque = "Peugeot">
      <PUISSANCE>5</PUISSANCE>
      <CYLINDREE>1.2</CYLINDREE>
      <CARBURATION>Essence</CARBURATION >
</MOTEUR>

<MOTEUR marque = "Renault">
      <PUISSANCE>4</PUISSANCE>
      <CYLINDREE>1.3</CYLINDREE>
      <CARBURATION>Diesel &amp;amp; Diesel</ CARBURATION >
</MOTEUR>
</MOTEURS>
```

- Cela entraîne une erreur si on remplace ces entités par le caractère correspondant,
- Ces entités sont automatiquement remplacées par les caractères lorsqu'on traite le fichier.

Détails du format XML: Entités internes (exemple)




```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE personne [
      <!ENTITY adresse "Ensa El Jadida – département informatique">
      <!ENTITY cours « Technologies XML et web sémantique">
]>
<personne>
      <lieu>&adresse;</lieu>
      <role>&cours;</role>
</personne>
```

- Pour afficher le résultat, on peut l'ouvrir dans Firefox

Détails du format XML: Entités externes

C’est une entité qui représente tout un document XML contenu dans un autre fichier:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE racine [
    <!ENTITY voiture1 SYSTEM "voiture1.xml">
]>
<garage>
    <vente>&voiture1;</vente>
</garage>
```

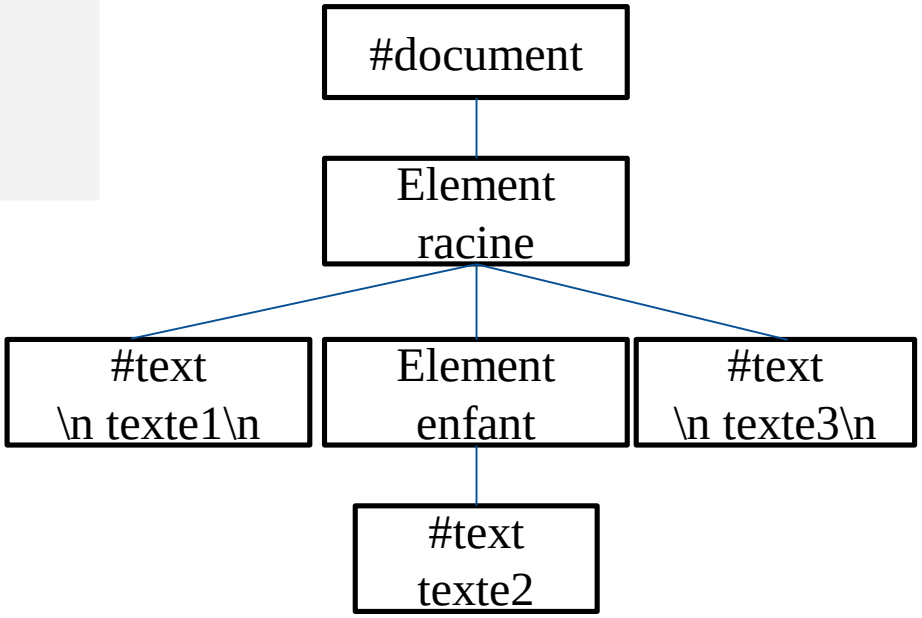
- Le mot clé SYSTEM commande la lecture d’un URL, ici, c’est un fichier local
- Le contenu du document voiture1.xml est inséré à la place de l’entité &voiture1;

Détails du format XML: Texte

Les textes font partie du contenu des éléments et sont vus comme des nœuds enfants. Il faut bien comprendre que la totalité du texte situé entre les balises, y compris les espaces et retour à la ligne font partie du texte.

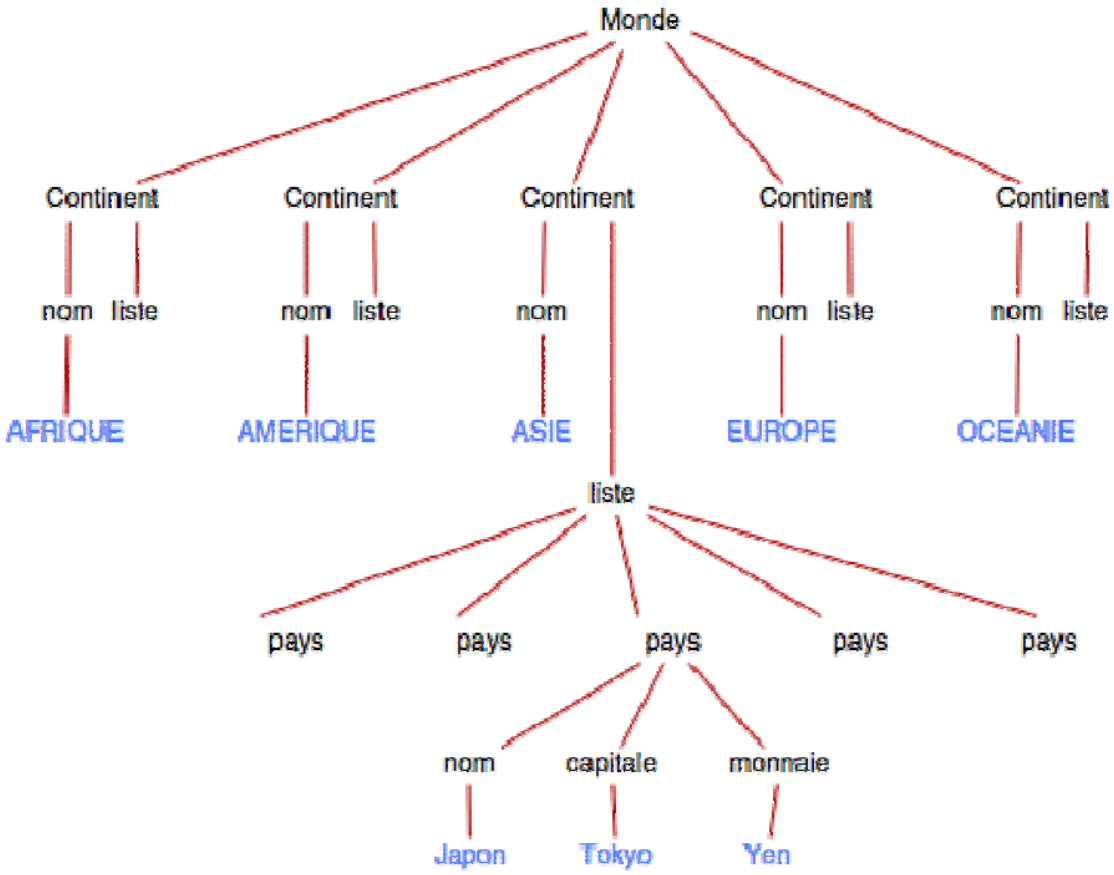
```
<?xml version="1.0" encoding="utf-8"?>
<racine>
    texte1
    <enfant>texte2</enfant>
    texte3
</racine>
```

- Voici son arbre correspondant
  - Notez que les retours à la ligne et espaces sont présents dans les textes sauf texte2.

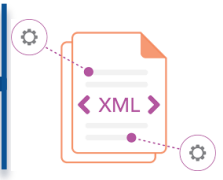


Détails du format XML: Exercice

Ecrivez le document XML correspondant à l'arbre ci-dessous



Détails du format XML: section CDATA

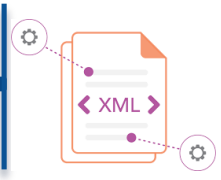


Lorsqu'on veut écrire du texte brut à ne pas analyser en tant qu'XML, on emploie une section CDATA :

```
<?xml version="1.0" encoding="utf-8"?>
<fichier>
    <nom>exemple1.xml</nom>
    <md5><![CDATA[19573b741c0c5c8190a83430967bfa58]]></md5>
</fichier>
```

- La partie entre <![CDATA[et]]> est ignorée par les analyseurs XML, on peut mettre n'importe quoi sauf]]>. Ces données sont considérées comme du texte par les analyseur.
  - Utiliser xmllint --nocdata document.xml pour le voir sans les marqueurs.

Détails du format XML: fusion des CDATA et textes



S'il y a des textes avant ou après une section CDATA, ils sont tous fusionnés avec elle.

```
<?xml version="1.0" encoding="utf-8"?>
<document>le md5 est <![CDATA[19573...]]> !</document>
```