XML - Introduction

XML - Introduction

Historique

Historique

- 1945 : *Vannenar Bush* invente la notion d'hypertexte
  - « *un appareil futur à usage personnel, une sore de classeur-bibliothèque privé automatique. C'est un dispositif grâce auquel un individu range tous ses livres, archives et communications, et qui est automatisé de telle sorte qu'il puisse être consulté de façon très efficace et flexible.* »
- 1970 : création du Generalized Markup Language (GML)
  - Solution de portage de documents entre plates formes
  - Langage propriétaire IBM crée par Charles Goldfarb, Ed Mosher et Ray Lory
- 1986 : l'organisme ISO normalise le GML : naissance du (Standard Generalized Markup Language (SGML) ISO 8879
- 1986 : l'hypertexte est implémenté pour la première fois : Hypercard (Apple)

Historique

- 1989 : *Berners Lee* propose une nouvelle méthode indépendante des matériels et des logiciels
  - Le HTML est né !

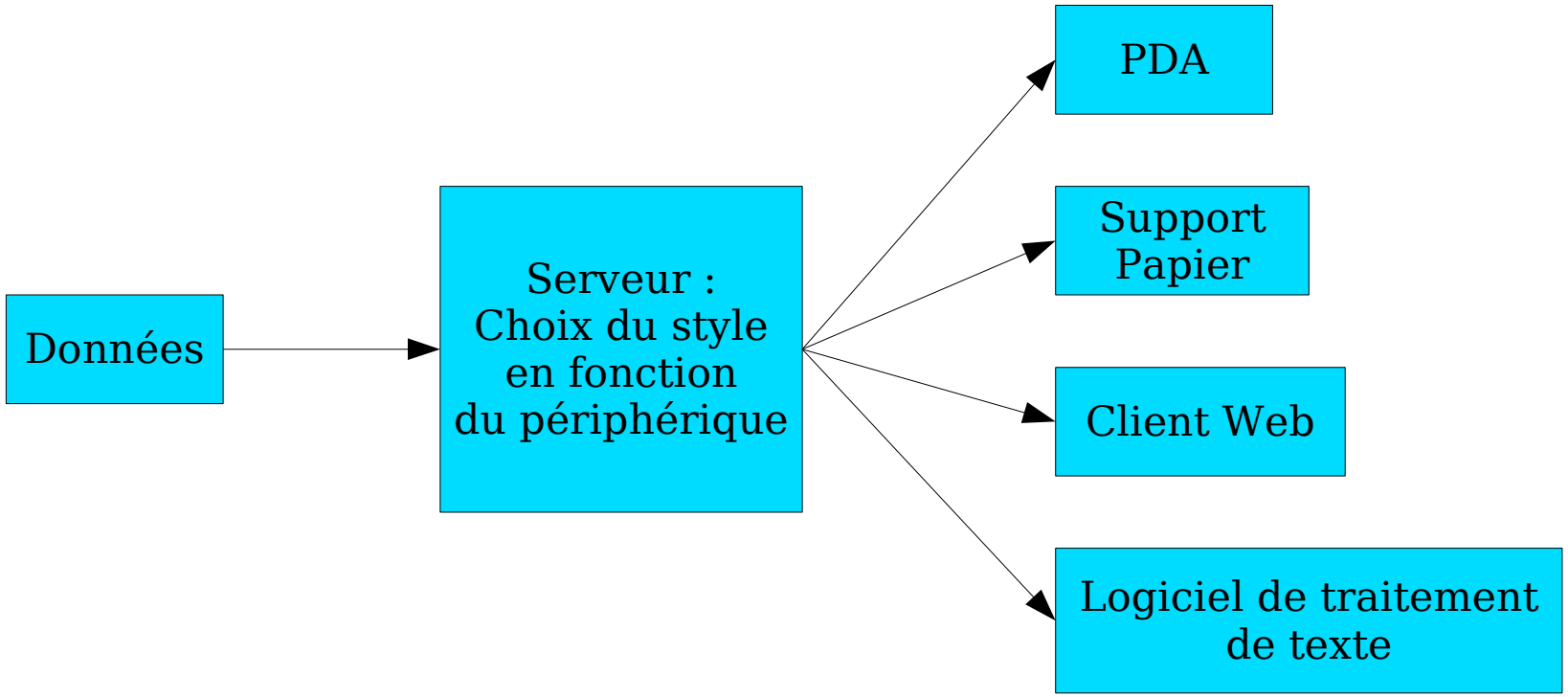
L'affichage et les données ?

- Une entreprise cherche à diffuser un ensemble de données (évolutives)
  - Une personne chargée de la communication décide du contenu
  - Un graphiste est chargé du design, couleurs, ...
- 1er solution
  - Chacun utilise un format propre
  - Pour la moindre mise à jour des données, tout le travail est à refaire
- 2ème solution
  - Chacun s'occupe de « sa partie »
    - Le premier de la structure et du contenu
    - Le deuxième de l'affichage

Ce que nous ne savons pas faire !

- HTML est un langage pour présenter des informations à **l'écran**
  - Il ne permet pas d'échanger des données. Les applications utilisent généralement des formats propriétaires
  - Il ne permet pas un traitement des données autre que l'affichage. Les données sont mélangées avec des directives pour formater l'affichage
- On ne sait pas interpréter des données fournies en HTML !

Notre objectif : séparer les données de l'affichage



Types de langages existants

- Langages de formatage de données
  - Objectif : description de l'affichage final du document
  - Permet d'échanger les documents entre application sans perdre le format d'affichage
  - Exemple : Rich Text Format (RTF)
- Langages Balisés
  - Objectif : décrire un document en séparant le contenu de l'affichage
  - Insérer des balises dans le document
  - Exemple : HTML

Le SGML

- Norme internationale de langages balisés dont le but est de faciliter l'accès aux données
  - De nature à être transportées sous divers formats d'édition
  - À destination d'une population d'utilisateurs hétérogènes
  - Pouvant être complexes et comportant des liens dynamiques
  - Susceptibles d'être souvent modifiées
  - A une longue durée de vie
- Principe
  - Marquage normalisé des éléments qui définissent la structure logique d'un texte
  - Distinction fondamentale entre contenu et présentation physique
- Trop complexe à implémenter et trop lourd pour Internet (langage enterré)

Le HTML

- Hyper Text Markup Language
- Implémentation simplifiée du SGML
- Trop limité !
- Langage figé : les balises sont définies dans la norme et ne sont pas modifiables)
- Victime de son succès
  - Mélange de diffusion de données et souci d'affichage final
- Exemple : Que contient ce document ?

```
<b>Cours XML</b> En <i>Licence Professionnelle</i>

Par <i>F. Nolot</i>
```

Vers le XML

- Proposé par Jon Bosak (Sun Microsystems) au consortium W3C (adopté en février 1988 sous sa version 1.0)
- Objectif : adapter le SGML aux besoins de l'internet
- Métalangage permettant de créer et formater les documents
  - C'est un langage de création de langages de balises
  - À mi-chemin entre le SGML et le HTML
- XML est indépendant de la plate-forme, du système d'exploitation et de l'environnement de développement

Nouvelles utilisations du Web

- Le réseau est hétérogène : les données doivent être représentées indépendamment d'une machine donnée
  - Commerce électronique : les entreprises veulent échanger des informations (pas pour les afficher)
- Les applications sont variées : les données doivent être représentées indépendamment d'une application
  - Moteur de recherche : si je sais interpréter les données transmises, je peux les indexer efficacement
  - Services en ligne : je peux envoyer mes données à un serveur pour leur appliquer un traitement donné (ex : publication)
- Une application = un format de données : il faut pouvoir transformer facilement les données d'un format à un autre

Pourquoi XML ?

- Le langage XML est une réponse aux besoins suivants :
  - Un document XML est au format ASCII : il voyage facilement
  - XML n'est pas lié à un mode d'utilisation : chacun peut se définir son propre langage
  - Le langage XSLT permet de transformer un document XML en un format applicatif
- Format universel
  - Représentation la plus simple possible d'un contenu. Que des chaînes de caractères
  - Indépendant de toute application et décrit avec mon propre vocabulaire
- Publier l'information
  - Outils de transformations simples pour convertir du contenu XML
- Échanger et intégrer l'information

Les points clés

- Se concentrer sur la structure du document et son contenu
- Ne pas se soucier de l'affichage
- Ne plus utiliser les balises ayant un sens uniquement visuel (<b>, <i>, , ...)
- Bien imbriquer les balises
- Ne plus compter sur l'interpréteur pour réparer vos petites erreurs
  - Oubli des guillemets
  - Oubli des dièses (pour les codes couleurs)
  - Oubli de fermer les balises

Terminologie

- XML est un métalangage : il est possible de le dériver en « dialecte » plus spécifique
- **Feuilles de style** : ensemble d'instructions de présentation physique destinées à être appliquées à un document
- Navigateur : logiciel d'affichage de documents HTML et XML
- Parseur : logiciel d'analyse du langage XML qui transmet les données vers une application pour l'affichage

XML - Introduction

La mise en forme

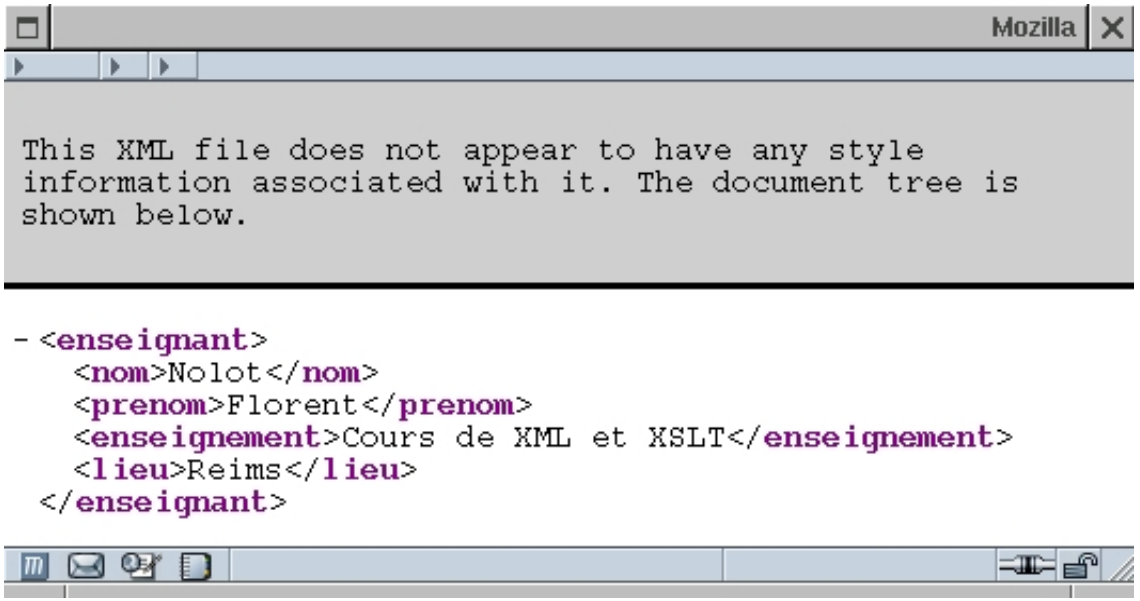
Mon premier fichier XML

- Les données sont
  - Un nom : Nolot
  - Un prénom : Florent
  - Un enseignement : Cours de XML et XSLT
  - Un lieu : Reims
- En XML, cela va donner :

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
    <enseignant>
        <nom>Nolot</nom>
        <prenom>Florent</prenom>
        <enseignement>Cours de XML et XSLT</enseignement>
        <lieu>Reims</lieu>
    </enseignant>
```

Résultat

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
        <enseignant>
            <nom>Nolot</nom>
            <prenom>Florent</prenom>
            <enseignement>Cours de XML et XSLT</enseignement>
            <lieu>Reims</lieu>
        </enseignant>
```



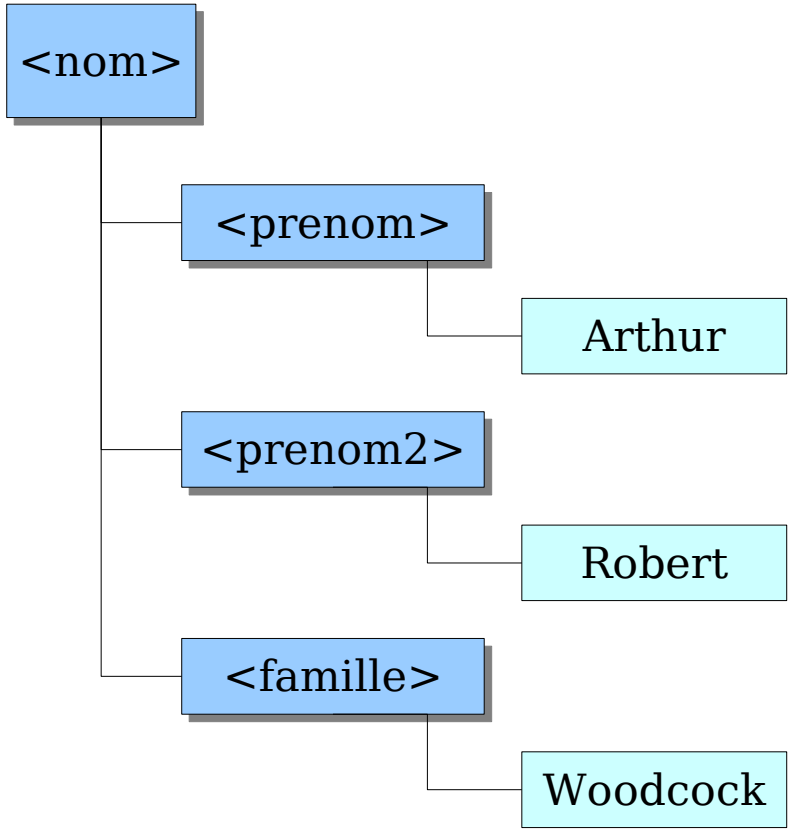
Terminologie

- Document XML bien formé
  - Il doit obéir aux règles syntaxiques suivantes
    - Les valeurs d'attributs doivent être entre des guillemets " "
    - Les noms des attributs doivent tous être en minuscules
    - Toute balise ouverte doit être fermée
    - Les balises uniques doivent être de la forme <balise />
- Un document XML est valide si
  - Il est bien formé
  - Il obéit à une structure explicitement définie par une grammaire dans un **DTD, Document Type Definition**

Hiérarchie

- L'information est regroupé selon une hiérarchie
- Chaque élément d'un document possède des relations **parent/enfant**
- Exemple :

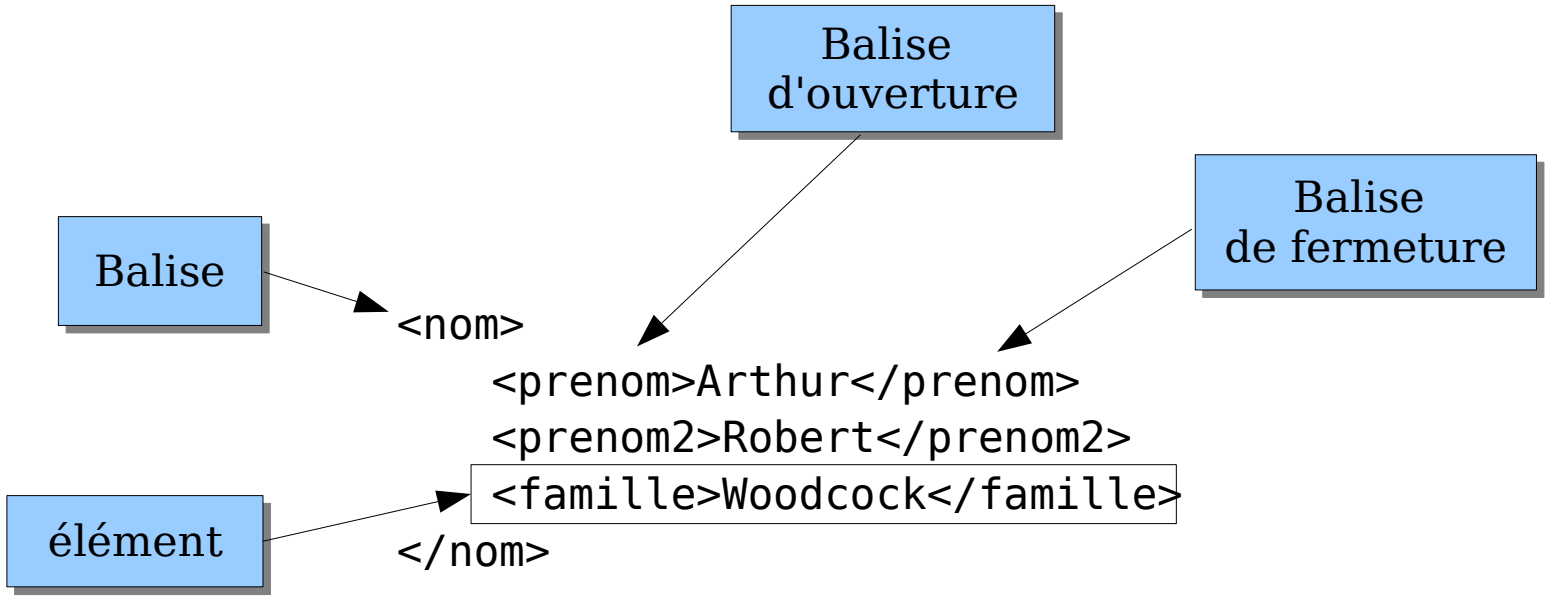
```
<nom>
    <prenom>Arthur</prenom>
    <prenom2>Robert</prenom2>
    <famille>Woodcock</famille>
</nom>
```



Balise, texte et éléments

- Les mots entre les caractères < et > sont des **balises**
  - Exemples : <nom>, <tr>,</td>
- Types de balises
  - **Balise d'ouverture** (commence par <) *Exemples* : `<td>`, `<tr>`
  - **Balise de fermeture** (commence par </) *Exemples* : `</td>`, `</tr>`
  - **Balise simple** : ne comprenant pas de balise de fermeture (comme <BR> en HTML) <balise /> (équivalent à <balise></balise>). Elle ne contienne pas de PCDATA
- Tous ce qui est entre une balise d'ouverture et celle de fermeture s'appelle un **élément**
  - *Exemple* : <td>Ceci est un élément</td>
- Le texte situé entre la balise d'ouverture et de fermeture est dit être du **PCDATA** quand il n'est constitué que de données

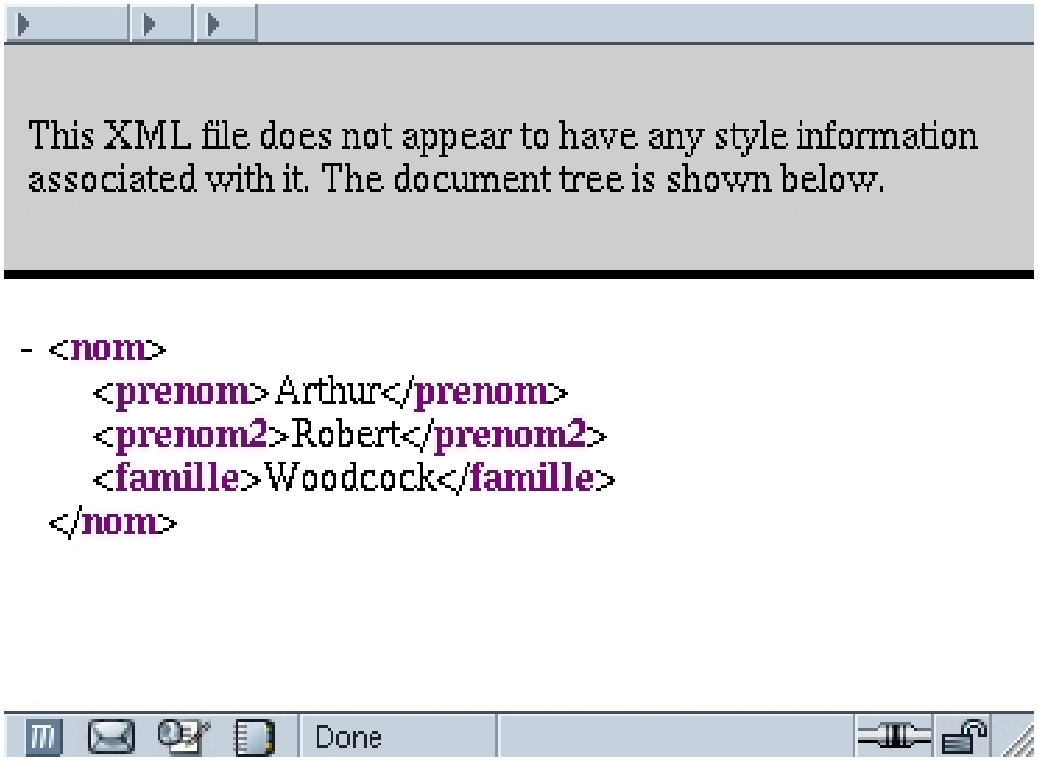
Balise, texte et éléments



- L'élément <prenom> contient du PCDATA, ainsi que les éléments <famille> et <prenom2> mais l'élément <nom> ne contient aucun PCDATA

Résultat sous Mozilla 1.7

```
<nom>
    <prenom>Arthur</prenom>
    <prenom2>Robert</prenom2>
    <famille>Woodcock</famille>
</nom>
```



Règles applicables aux éléments

1. **Toute balise d'ouverture doit posséder une balise de fermeture correspondante**
   - Exemple de code HTML incorrecte au sens du XML

```
<HTML>
    <BODY>
        <P>Voici du texte dans un paragraphe
        <BR>Voici encore du texte dans le même paragraphe
        <P>Deuxième paragraphe</p>
    </BODY>
</HTML>
```

   - Balise de fermeture du premier <P> oublié
   -
 ne possède pas de balise de fermeture associée

Règles applicables aux éléments

2. Les balises ne peuvent se chevaucher

- Exemple incorrecte

```
<P><STRONG>Texte <EM>mis en forme</STRONG> avec des
  </EM> balises HTML !</P>
```

- Solution :

```
<P><STRONG>Texte <EM>mis en forme</EM></STRONG><EM>
  avec des </EM>balises HTML !</P>
```

Règles applicables aux éléments

- Réorganisation visuelle

```
<P>
    <STRONG>
        Texte
        <EM>
            mis en forme
        </EM>
    </STRONG>
    <EM>
    avec des balises HTML !
    </EM>
</P>
```

Règles applicables aux éléments

- **Un document XML ne peut posséder qu'un seul élément racine**
  - L'élément racine est l'élément de plus haut niveau du document
  - Dans l'exemple suivant, <nom> est l'élément racine

```
<nom>
    <prenom>Arthur</prenom>
    <prenom2>Robert</prenom2>
    <famille>Woodcock</famille>
</nom>
```

  - L'exemple suivant est incorrecte, 2 éléments racines !

```
<nom><prenom>Arthur</prenom></nom>
<nom><prenom>Durant</prenom></nom>
```

Règles applicables aux éléments

- Noms d'élément
  - Quelques règles à respecter
    - Un nom doit commencer par une lettre ou par _ mais jamais par un chiffre ou un signe de ponctuation
    - Après le premier caractère, les nombres sont autorisés ainsi que les caractères – et .
    - Un nom ne peut comporter d'espace
    - Le caractère : a un usage réservé
    - Aucun nom de doit commence par les lettres xml, en majuscules, minuscule ou même le mélange des 2 (XmL, xmL, xMl, ...)
    - Aucun espace après le caractère d'ouverture <

Règles applicables aux éléments

- **Sensibilité à la casse**
- Ainsi <prenom> et <Prenom> sont deux balises différentes
- Il faut tout de même éviter de mélanger des balises qui diffèrent que par la casse

Règles applicables aux éléments

- Espaces vierges dans des PCDATA
  - Les espaces vierges sont :
    - le caractère espace
    - le retour à la ligne
    - et les tabulations
  - En HTML, il y a élimination des espaces vierges
    - Exemple : « le texte avec des espaces » s'affichera « le texte avec des espaces »
  - En XML, les **espaces vierges dans les PCDATA subsistent**

Règles applicables aux éléments

- Espaces vierges dans des PCDATA (suite)
  - **Attention**
    - Une nouvelle ligne est codé par 2 caractères
      - Le caractère de saut de ligne (*Line Feed* (LF))
      - Le caractère de retour à la ligne (*Carriage Return* (CR))
    - Sous Unix, une nouvelle ligne est codée par un LF
    - Sous Windows, une nouvelle ligne est codée par LF et CR
  - Pour conserver l'interopérabilité du XML, les caractères de nouvelles lignes sont remplacés par un simple caractère de saut de ligne par les analyseurs de fichiers XML

Règles applicables aux éléments

- Espaces vierges dans le balisage
  - *Exemple* :

```
<balise>
    <autre-balise>Ceci est du XML</autre-balise>
</balise>
```

  - Il existe une nouvelle ligne après `<balise>` et des espaces devant `<autre-balise>`
    - Uniquement destinée à la lisibilité du document
    - Ils sont appelés **espaces vierges supplémentaires**
  - Pour déterminer si un espaces vierges est supplémentaire ou pas, cela dépend si la balise peut ou pas contenir des PCDATA
    - Si une balise ne peut contenir que des élements, tous espaces vierges est considéré comme supplémentaire
    - Sinon, ils sont préservés (règle des espaces vierges dans les PCDATA)

Les attributs

- Un attribut est une paire nom/valeur associé à un élément
- Attaché à la balise d'ouverture
- Tout attribut doit posséder une valeur, même si celle-ci n'est qu'une chaîne vide
- La valeur **doit être** entre guillemets (" ") ou entre apostrophes (' ')
- *Exemple* :

```
<nom surnom='toto' taille='180'>
    <prenom>Arthur</prenom>
    <famille>Woodcock</famille>
</nom>
```

- Les attributs répondent aux mêmes règles de dénomination que les éléments

Les commentaires

- Débute par <!-- et se termine par -->
- **Attention** :
  - Aucun commentaire dans une balise
  - Ne jamais utiliser -- dans un commentaire

    <!-- Ceci est un -- commentaire incorrecte -->

Déclaration XML

- Pour signaler qu'un document est du XML
- Débute par `<?xml` et se termine par ?>
- Un attribut `version` est obligatoire
- Les attributs `encoding` et `standalone` sont facultatifs
- Les attributs doivent obligatoirement être dans cet ordre

  `version encoding standalone`
- *Exemple* :

  ```
  <?xml version='1.0' encoding='UTF-16' standalone='yes' ?>
  ```
- La déclaration soit être au début du fichier

Attribut `encoding`

- Lors de l'enregistrement d'un fichier, les caractères sont remplacés par des nombres
- Il existe plusieurs tables d'encodage de caractères : ASCII 7bits, ASCII 8bits, Unicode 8 bits (UTF-8), Unicode 16 bits (UTF-16), ...
- Les attributs encoding classique
  - `UTF-8`, `UTF-16` ou bien `ISO-8859-1`
- Sous Notepad Windows, c'est le codage `windows-1252` qui est souvent utilisé pour coder les caractères
- Sous Windows NT, 2000 ou XP, c'est plutôt de l'Unicode

Les jeux de caractères ISO-8859

- Développés par European Computer Manufacturer's Association (ECMA)
- Représentation sur un octet donc 256 caractères représentés
- Les 128 premiers caractères (0 à 127) correspondent au jeu de caractères ASCII et la suite est spécifique au codage

Jeu de caractères ASCII

+	0	1	2	3	4	5	6	7	8	9
30				!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~			

iso-8859-1

+	0	1	2	3	4	5	6	7	8	9
160		¡	¢	£	¤	¥	¦	§	¨	©
170	ª	«	¬	-	®	¯	°	±	²	³
180	´	µ	¶	·	¸	¹	º	»	¼	½
190	¾	¿	À	Á	Â	Ã	Ä	Å	Æ	Ç
200	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ
210	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û
220	Ü	Ý	Þ	ß	à	á	â	ã	ä	å
230	æ	ç	è	é	ê	ë	ì	í	î	ï
240	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù
250	ú	û	ü	ý	þ	ÿ				

Attribut standalone

- 2 valeurs possibles : yes ou no
- Permet de signaler si le fichier possède une dépendance avec d'autres fichiers
  - *Exemple* : définition de la grammaire (DTD) dans un fichier externe ou bien directement dans le fichier XML

Caractères spéciaux dans les PCDATA

- Les caractères d'échappement
  - Utilisation des **références d'entités**

    `&` caractère &

    `<` caractère <

    `>` caractère >

    `&apos;` caractère '

    `&quot;` caractère "
  - Utilisation de **références de caractères numériques**
    - Chaîne comme `&#nnn;` où *nnn* représente le nombre décimal Unicode du caractère à insérer, quelque soit l'encodage du document

Sections CDATA

- Signifie *Character DATA*
- Permet de signaler que les informations dans cette section ne doivent pas être analyser
  - Permet de mettre des données brutes qui seront afficher sans aucune analyse
- Une section CDATA commence par `<![CDATA[` et se termine par `]]>`