

# Le Langage XML

## Sources du document :

- Introduction au XML, Simon St-Laurent, éditions Eyrolles, année 2000 ;
- XML en concentre (XML in a Nutshell) 3<sup>ème</sup> édition, Elliotte Rusty Harold & W. Scott Means, O'Reilly, année 2005.
- XSLT par la pratique, Steven Holzner, éditions Eyrolles, année 2002 ;
- Le consortium W3C (textes de référence XML du 27 juin 2001 et du 4 février 2004), traduit par la-grange.net (section w3c) ;

## Autres sources données à titre de bibliographie :

- Le site de Gilles Chagnon (XML) ;
- <http://xmlfr.org>.



Olivier Mondet  
<http://unidentified-one.net>

## A. Le Langage XML

### A.1. Introduction

Comme nous l'avons dit dans la partie consacrée à XHTML, l'essence, le fondement du XML est la séparation du contenu et de la présentation.

Le langage XML est un langage où (enfin) il y a une réelle rigueur syntaxique à l'instar de tous les autres langages. C'est un langage où les normes forcent à l'adoption d'un formalisme obligatoire et à une « certification » du contenu : les documents valides et bien formés.

Qu'apporte XML (**EX**tensible **M**arkup **L**anguage) au HTML ?

- Il s'agit d'un langage enrichissable,
- C'est un langage markup plus puissant qu'HTML,
- C'est un langage qui sait intégrer des données,
- C'est un langage non prédéfini, où l'on doit définir ses propres balises,
- C'est un langage qui utilise les DTD (Document Type Definition) pour décrire la structure des données,
- C'est un langage recommandé par le consortium W3C,
- C'est un langage où qui se décrit par lui-même.

Au niveau de la présentation XML utilise aussi bien CSS que son propre format XSL (EXtensible Style Language). XSL peut générer du HTML (cela permet entre autre d'afficher des informations XML même sur un navigateur ne connaissant pas le XML). XSL permet donc le formatage de documents XML. Il utilise la syntaxe d'XML pour cela. La base de XSL se compose de patterns qui servent à définir le formatage d'un document. Les patterns sont comparés aux documents et s'appliquent lorsque des règles sont respectées.

Ce chapitre abordera donc XML dans une première partie, puis CSS. Le chapitre suivant traitera d'XSL (XSLT, XSL-FO). Tout l'intérêt d'XML réside dans l'approche que l'on en a. Il faut donc d'abord se rendre compte du fonctionnement avant de définir toute la portée du langage.

## A.2. Les documents bien formés et valides

---

Nous avons vu qu'en XML, le document doit être bien formé, c'est-à-dire respecter des règles :

- pas d'enchevêtrement de balises ;
- les éléments (balises) et leur attributs doivent être écrits en minuscules ;
- les valeurs des éléments doivent toujours être entre quotes ;
- on ne peut utiliser des raccourcis de balises (par exemple : checked, noshade, compact doivent être suivis de leur valeur) ;
- les éléments vides comme br ou hr doivent s'écrire <br/> ou <br></br> (<hr/> ou <hr></hr>) ;
- les scripts externes doivent être préférés, tout comme les feuilles de style externes ;
- les éléments doivent être identifiés de façon unique par un attribut id lorsque c'est nécessaire, l'attribut name sera bientôt abandonné

Le document, en plus d'être bien formé, doit être valide, c'est-à-dire conforme à sa définition. Autrement dit, un document XML est accompagné d'une définition des éléments qu'il contient (la DTD que nous verrons par la suite) et il est considéré comme valide s'il respecte bien la définition indiquée. Un document non valide pourrait être rejeté par le parseur ou le navigateur qui le parcourt.

## A.3. Premières notions

---

### A.3.1. Premier exemple complet

Nous allons voir au travers d'exemples, l'implémentation de documents XML. Vous aurez noté que depuis le chapitre XHTML, nous employons exclusivement le terme de document et non plus de page Web comme cela aurait pu arriver dans les trois premiers chapitres. Car il s'agit bien de document. Document au même titre qu'une feuille de calcul ou qu'un document texte.

Voici un premier exemple complet d'approche d'XML : Un document XML, une DTD et une feuille de style CSS.

#### ex001.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="ex001.css" type="text/css"?>

<!DOCTYPE untexte SYSTEM "ex001.dtd">

<untexte>
Mes premiers pas avec XML. Premier exemple !
</untexte>
```

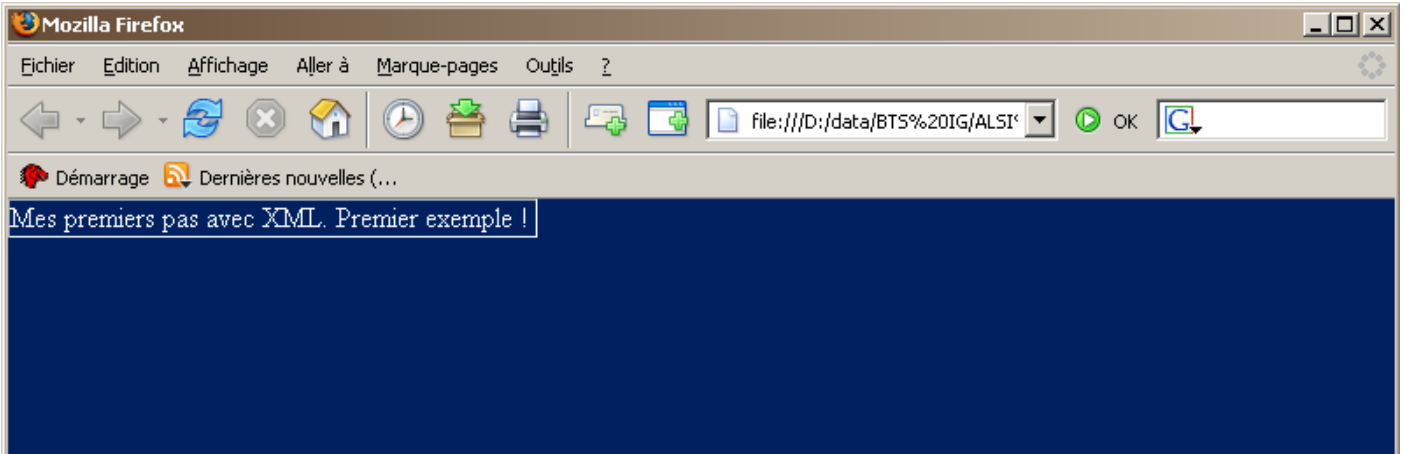
#### ex001.dtd

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT untexte (#PCDATA)>
```

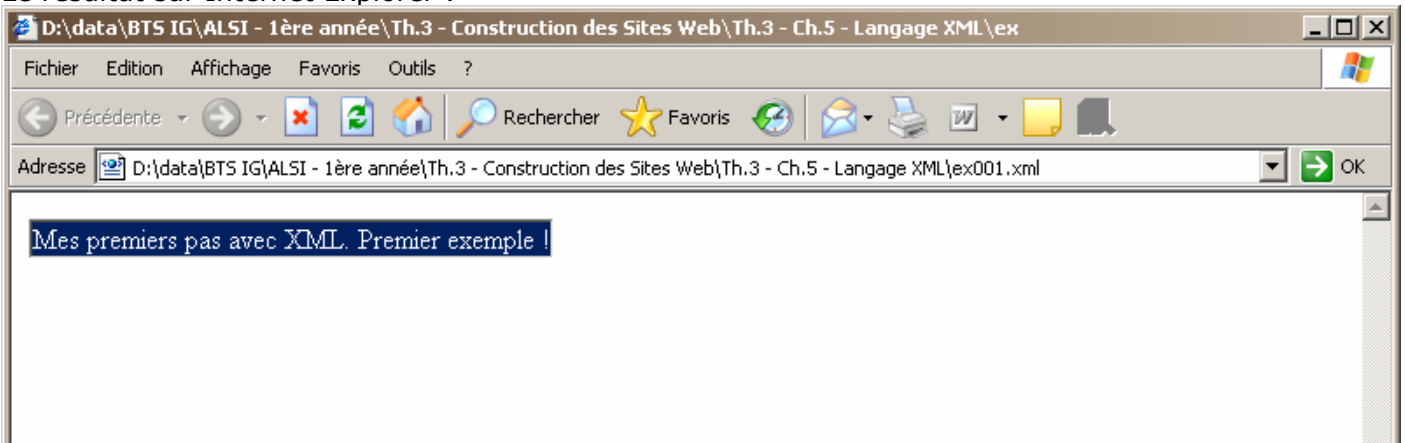
#### ex001.css

```
untexte
{
  display: block;
  background: #002060;
  border: inset;
  border-width: thin;
  width: 300px;
  color: #ffffff;
}
```

Le résultat sur Firefox :



Le résultat sur Internet Explorer :



XML comme nous l'avons vu, permet de séparer les données de leur mise en forme avec un langage plus rigoureux que le HTML. Le HTML permettait de nombreuses approximations, erreurs de syntaxe que ne permet pas le XML. Les différents navigateurs ont eu jusqu'à présent un rôle de lecteur souple de code HTML qu'il soit ou non bien écrit ; « les navigateurs se débrouillaient comme ils le pouvaient ». Avec le XML les navigateurs ont également un rôle de validation de code. Le navigateur signale les erreurs et omissions et n'affiche les pages que quand le document est bien formé.

On aura remarqué dans ce document relativement complexe un grand nombre de balises nouvelles... On remarque ensuite une « balise » qui se nomme <untexte> et qui encadre le texte affiché dans le navigateur. Il s'agit là du grand intérêt d'XML, la création de ses propres « balises » (éléments).

Les parties qui suivent vont donc montrer la création et l'utilisation de ses propres balises, que l'on nommera désormais « éléments ».

### A.3.2. Second exemple

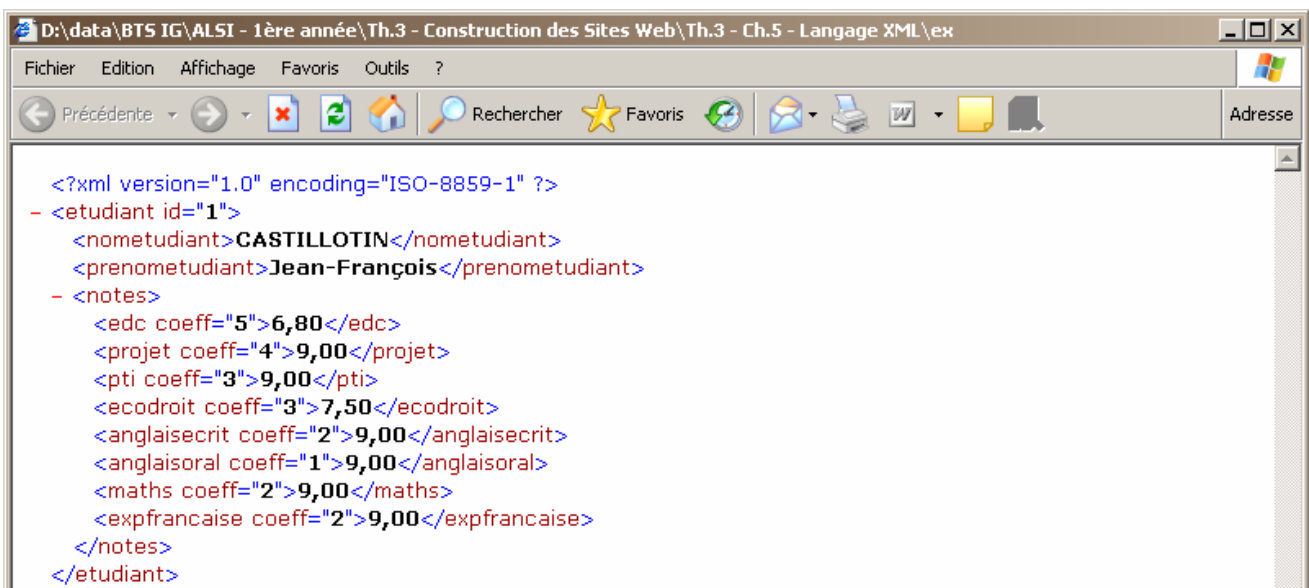
Voici un exemple (que l'on fera évoluer) de mise en forme de données avec XML.

Soit l'exemple d'un étudiant dont les résultats se présentent ainsi :

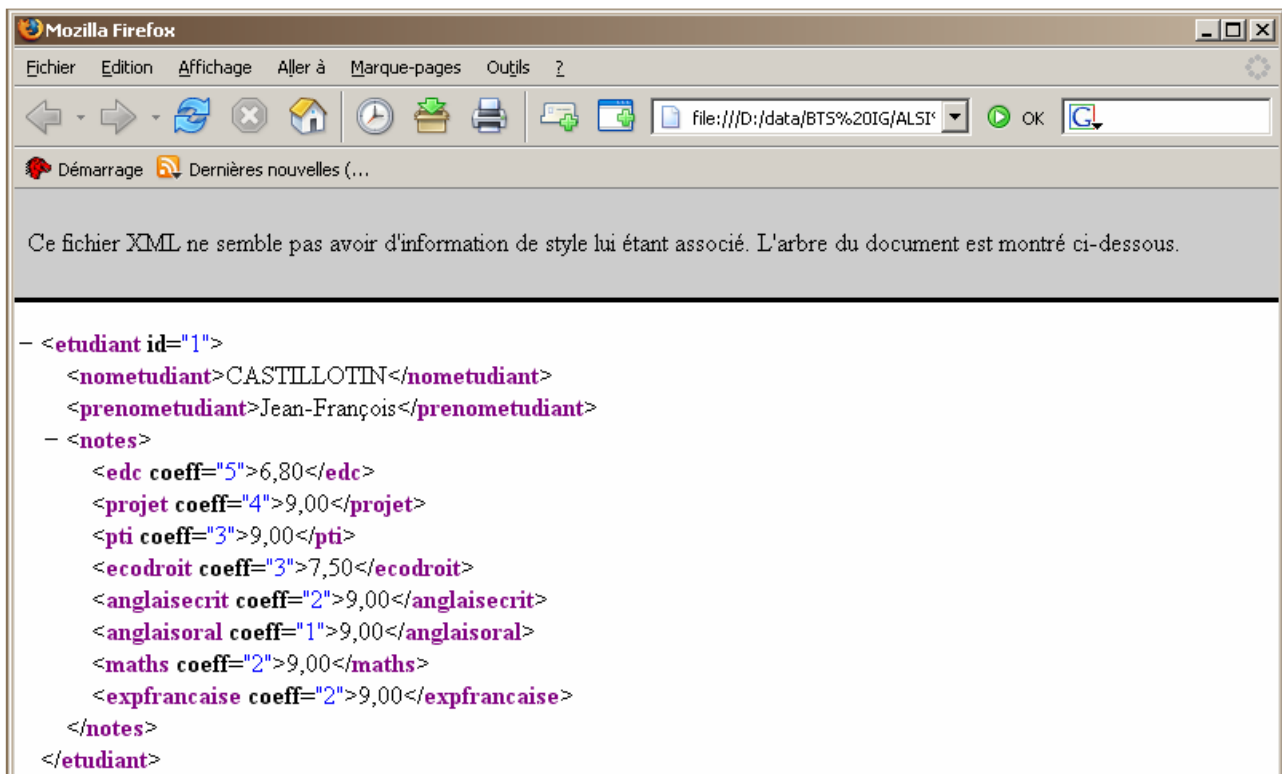
N°	Noms	Prénoms	Etude de cas	Soutenance de Projet Info.	Pratique des Techniques Info.	Economie & droit	Anglais (écrit)	Anglais (oral)	Maths	Expression Française
			5	4	3	3	2	1	2	2
1	CASTILLOTIN	Jean-François	6,80	9,00	9,00	7,50	9,00	9,00	9,00	9,00

## ex002.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<etudiant id="1">
  <nometudiant>CASTILLOTIN</nometudiant>
  <prenometudiant>Jean-François</prenometudiant>
  <notes>
    <edc coeff="5">6,80</edc>
    <projet coeff="4">9,00</projet>
    <pti coeff="3">9,00</pti>
    <ecodroit coeff="3">7,50</ecodroit>
    <anglaisecrit coeff="2">9,00</anglaisecrit>
    <anglaisoral coeff="1">9,00</anglaisoral>
    <maths coeff="2">9,00</maths>
    <expfrancaise coeff="2">9,00</expfrancaise>
  </notes>
</etudiant>
```



Le document peut être lu sous divers navigateurs, tels que Mozilla Firefox :



Dans ce document nous avons listé des informations sans les définir vraiment. Tous les éléments listés sont entrés de façon ordonnée mais rien ne nous permet de dire qu'elles ont une structure imposée. Ce sera le rôle des DTD de définir la structure des données. Définition qui imposera au document un squelette où viendront s'implémenter les données.

### A.3.3. Notion de DTD

La DTD (définition de type de document) sert à préciser quels éléments et entités peuvent apparaître dans un document, quels seront leurs contenus et quels doivent être leur attributs. Il s'agit d'un « balisage » du document XML qui seul n'a pas de fonctions de structuration des données, d'agencement des contenus. La DTD est donc un document négocié entre les différents partenaires intéressés par la mise en forme d'un document. Car c'est selon sa structure que seront déterminés les documents XML. Nous l'avons dit, la DTD est également un moyen de contrôle que le contenu d'un document soit bien conforme avec ce qui est attendu.

Dans notre document XML de l'exemple ex002 nous avons donc défini des éléments (l'on nommera plus balises). `<NOTES>...</NOTES>`, `<NOMETUDIANT>...</NOMETUDIANT>` sont des **éléments**. Dans l'élément `<ETUDIANT ID="xx">...</ETUDIANT>`, `ID="xx"` est un **attribut**. En HTML on utilise beaucoup les attributs car ce sont eux qui permettent la mise en forme. En XML les attributs servent à positionner des éléments destinés aux processeurs. Ce ne sont pas des données à proprement parler, ce sont des éléments qui ne sont pas destinés à être visibles. Il est donc recommandé d'utiliser peu d'attributs, sauf pour la mise en forme depuis HTML (avec l'attribut `STYLE`). Les éléments peuvent contenir d'autres éléments (ex : `<NOTES>...</NOTES>`) ce que ne permettent pas les attributs.

## A.4. Structuration des documents XML avec les DTD

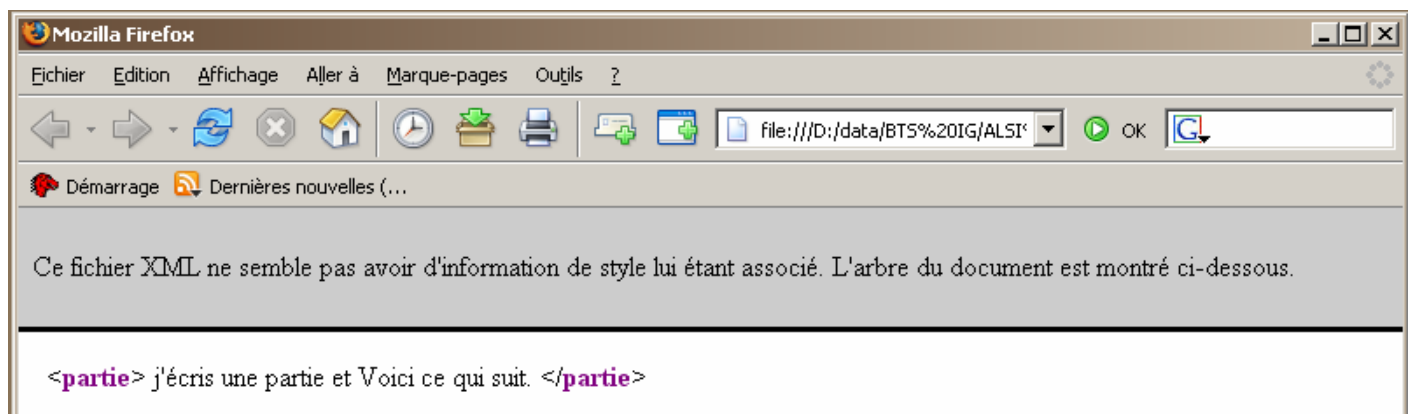
### A.4.1. DTD définie dans le document XML

Les DTD peuvent être définies au sein d'un document lui-même.

#### ex004.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE DOCUMENT [
  <!ELEMENT partie (#PCDATA)>
  <!ENTITY description "Voici ce qui suit.">
]>

<partie>
J'écris une partie et &description;
</partie>
```





```

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE DOCUMENT (View Source for full doctype...)>
<partie>j'écris une partie et Voici ce qui suit.</partie>

```

Nous avons donc créé un élément <PARTIE> et une entité DESCRIPTION.

**L'élément** sert donc à organiser les données du document XML, alors que **l'entité** sert plutôt à placer du texte libre (on peut y insérer des fractions de code HTML) qui revient fréquemment ou qui n'est pas une donnée à proprement dit. La notion d'entité sera développée par la suite.

Nous pouvons à présent détailler les différentes déclarations utilisées dans le code :

Eléments	Explications
<?xml version="1.0" encoding="ISO-8859-1"?>	Indique le numéro de version XML utilisé et que le document est encodé de façon compatible avec la norme ISO-8859-1 (jeu de caractère Latin-1).
<!DOCTYPE nom [...]>	Définit un document nommé DOCUMENT dans l'exemple, qui contient un élément et une entité.
<!ELEMENT nom type_données>	Définit un élément nommé PARTIE dans l'exemple, qui contiendra des données caractères (#CDATA).
<!ENTITY nom définition>	Définit une entité DESCRIPTION dans notre exemple, avec une valeur par défaut : "Voici ce qui suit."

### A.4.2. Eléments, entités, attributs

Nous pouvons enrichir notre exemple à présent avec l'ajout de deux attributs à l'élément ETUDIANT.

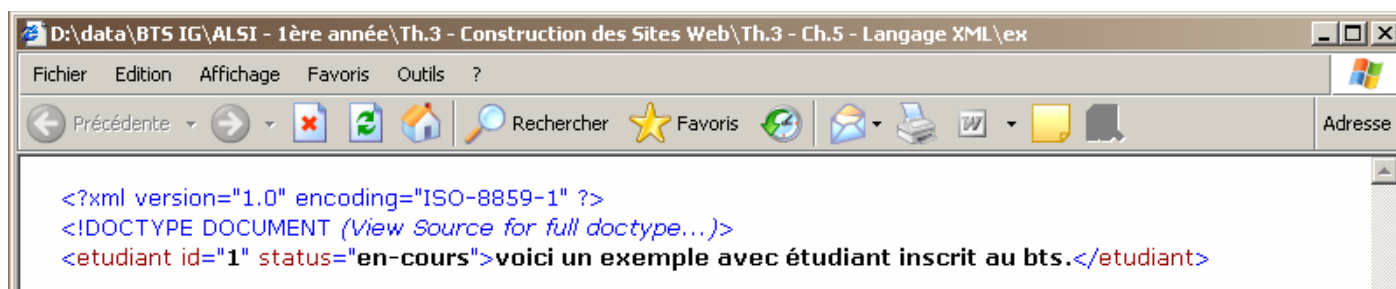
#### ex005.xml

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE etudiant [
  <!ELEMENT etudiant (#PCDATA)>
  <!ATTLIST etudiant
    id CDATA #REQUIRED
    status (en-cours|clos|demission) "en-cours">
  <!ENTITY description "étudiant inscrit au bts.">
]>

<etudiant id="1">
voici un exemple avec &description;
</etudiant>

```



```

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE DOCUMENT (View Source for full doctype...)>
<etudiant id="1" status="en-cours">voici un exemple avec étudiant inscrit au bts.</etudiant>

```

Nous pouvons examiner plus en détail la définition des attributs (ID, STATUS) de l'élément ETUDIANT.

Eléments	Explications
<!ATTLIST nom valeurs>	Permet de définir une liste d'attributs pour l'élément ETUDIANT dans notre exemple.
id CDATA #REQUIRED	Définition d'un attribut nommé ID dans notre exemple, de type donnée caractère CDATA, dont la définition est obligatoire #REQUIRED (ou optionnelle #IMPLIED, ou donnée par défaut #FIXED). On ne peut donner de valeurs par défaut dans ce cas.
status (en-cours clos démission) "en-cours"	Définition d'un attribut qui ne peut accepter seulement que trois valeurs "en-cours", "clos" ou "démission". Cet attribut possède la valeur "en-cours" par défaut.

On connaît déjà bien les attributs après l'étude du langage HTML. On connaît par exemple la balise vide <IMG SRC="image/chien.gif" WIDTH="100" HEIGHT="189" ALT="Photo de chien" /> et ses différents attributs (SRC, WIDTH...). Imaginons la définition de l'élément IMG au travers d'une DTD :

```
<!DOCTYPE DOCUMENT [
  <!ELEMENT img EMPTY>
  <!ATTLIST img
    src CDATA #REQUIRED
    width CDATA #IMPLIED
    height CDATA #IMPLIED
    alt CDATA #IMPLIED>
]>
```

Autrement dit, lorsque l'on relie une page XHTML ou HTML à une DTD du consortium W3C : <DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"> ; on est tenté de penser que cette DTD contient des définitions de centaines d'éléments qui forment nos « balises » (la DTD XHTML fait environ 1500 lignes).

#### A.4.2.1. Types des attributs

Revenons sur les types des attributs. Nous avons vu que le type le plus souple (comme pour les éléments d'ailleurs) était CDATA, quels sont les autres ?

Il en existe au total dix types :

Types	Définitions, exemples
CDATA	Contient n'importe quelle chaîne de caractères.
NMTOKEN	Contient un nom sans espaces, en suivant approximativement les mêmes règles que les noms des éléments, entités et attributs XML. Ces noms peuvent commencer par n'importe quel caractère ou signe autorisé ( _ - . : ) ou chiffre. Ce type est très utile pour des dates. <u>Par exemple :</u> ... <!ATTLIST étudiant date_naissance NMTOKEN #REQUIRED> ... <étudiant date_naissance="20-01-1984">...</étudiant>
NMTOKENS	Correspond à une énumération de NMTOKEN à la suite, séparés par des espaces (d'où l'interdiction des espaces dans les NMTOKEN). Ce type est très utile pour énumérer des dates. <u>Par exemple :</u> ... <!ATTLIST auteur dates NMTOKENS #REQUIRED> ... <auteur dates="13-04-1815 24-12-1903">...</auteur>

ENUMERATION	<p>Attention le type énumération ne s'écrit pas, c'est en énumérant les valeurs de l'attribut que l'on définit ce type. Permet d'énumérer des listes prédéfinies de valeurs.</p> <p><u>Par exemple :</u></p> <pre>... &lt;!ATTLIST pantalon couleur (jaune   vert   bleu) #REQUIRED&gt; ... &lt;pantalon couleur="jaune"&gt;...&lt;/pantalon&gt;</pre>
ENTITY	<p>Comme nous l'avons dans les exemples 004 et 005, dans une première approche, les entités permettent de définir du texte libre dans un document. Les entités permettent de définir n'importe où dans une DTD séquences de sous-éléments ou des attributs que l'on peut utiliser à plusieurs endroits. D'autres exemples dans la suite de ce chapitre permettent de mettre en valeur l'usage de ce type.</p>
ENTITIES	<p>Permet d'énumérer plusieurs ENTITY dans un même attribut.</p>
ID	<p>Il s'agit de l'identifiant unique dans un document. L'identifiant doit être un nom XML au sens strict, sans espaces, ne commençant pas par un chiffre, mais pouvant contenir les signes autorisés (_ - . :). Ainsi l'identifiant 1345 n'est pas correct, on choisira _1345 à la place. Chaque élément ne peut avoir qu'un seul attribut de type ID.</p> <p><u>Par exemple :</u></p> <pre>... &lt;!ATTLIST salarié matricule ID #REQUIRED&gt; ... &lt;salarié matricule="_45654"&gt;...&lt;/salarié&gt;</pre>
IDREF	<p>Il s'agit d'une référence à un ID d'un des éléments du même document. C'est donc très utile pour pouvoir se référer à un élément sans risque d'erreur. Le seul ennui c'est que l'ID qui est référencé n'est pas contrôlé, c'est-à-dire que si l'on fait référence à un attribut de type ID _5654 on ne sait pas de quel élément.</p> <p><u>Par exemple :</u></p> <pre>... &lt;!ATTLIST étudiant numéro ID #REQUIRED&gt; ... &lt;!ATTLIST promotion num_étudiant IDREF #REQUIRED&gt; ... &lt;étudiant numéro="_2003_7653"&gt;...&lt;/étudiant&gt; ... &lt;promotion num_étudiant="_2003_7653"&gt;...&lt;/promotion&gt;</pre>
IDREFS	<p>Correspond à une énumération d'IDREF.</p> <p><u>Par exemple :</u></p> <pre>... &lt;!ATTLIST étudiant numéro ID #REQUIRED&gt; ... &lt;!ATTLIST promotion num_étudiant IDREF #REQUIRED&gt; ... &lt;étudiant numéro="_2003_7653"&gt;...&lt;/étudiant&gt; &lt;étudiant numéro="_2003_9876"&gt;...&lt;/étudiant&gt; &lt;étudiant numéro="_2003_3486"&gt;...&lt;/étudiant&gt; ... &lt;promotion num_étudiant="_2003_7653 _2003_9876 _2003_3486"&gt; ... &lt;/promotion&gt;</pre>
NOTATION	<p>Permet d'associer des types à certains éléments. Des exemples dans la suite de ce chapitre permettent de mettre en valeur l'usage de ce type.</p>

Nous étudierons plus tard les entités.



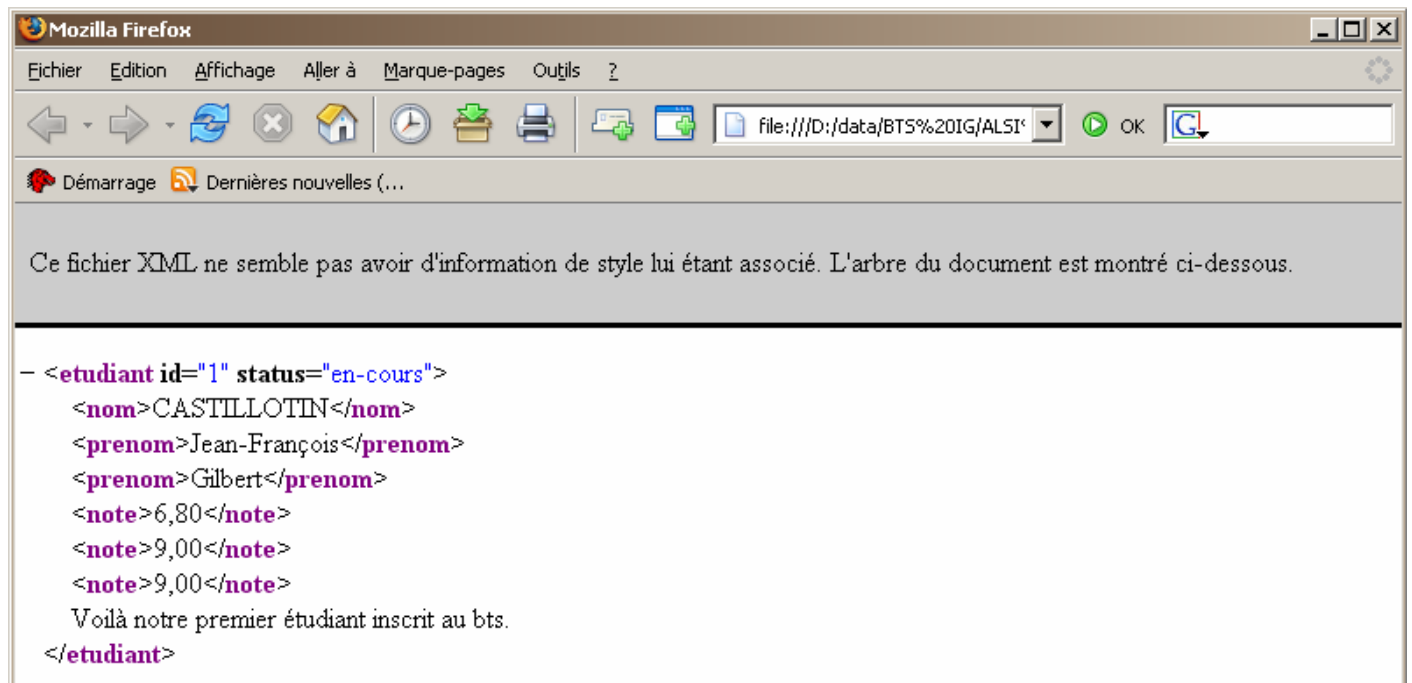
### A.4.3. Structures plus complexes

Nous allons maintenant insérer d'autres (sous-) éléments dans notre document.

#### ex006.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE DOCUMENT [
<!ELEMENT etudiant (nom,prenom+,note*,redoublant?)>
  <!ATTLIST etudiant
    id CDATA #REQUIRED
    status (en-cours|clos|demission) "en-cours">
  <!ELEMENT nom (#PCDATA)>
  <!ELEMENT prenom (#PCDATA)>
  <!ELEMENT note (#PCDATA)>
  <!ELEMENT redoublant (#PCDATA)>
<!ENTITY description "étudiant inscrit au bts.">
]>

<etudiant id="1">
  <nom>CASTILLOTIN</nom>
  <prenom>Jean-François</prenom>
  <prenom>Gilbert</prenom>
  <note>6,80</note>
  <note>9,00</note>
  <note>9,00</note>
  Voilà notre premier &description;
</etudiant>
```



On aura remarqué à présent que notre élément ETUDIANT contenait l'indication de sous-éléments au sein de ce que l'on appelle une séquence (entre parenthèses). En effet la plus petite division d'un élément c'est lorsqu'il contient son typage comme #PCDATA ; mais lorsqu'un élément contient une séquence qui indique d'autres éléments séparés par une virgule il est alors l'élément père et les sous-éléments sont les éléments fils (on trouve également enfants).

Éléments	Explications
<!ELEMENT etudiant (nom, prenom+, note*, redoublant?)>	<p>Définit un élément ETUDIANT qui est structuré avec quatre autres éléments qui seront définis par la suite. Tous les éléments séparés par une virgule doivent apparaître dans l'ordre défini.</p> <p><u>Le signe +</u> signifie que l'élément doit apparaître une fois au moins et peut-être davantage (signifie un ou plusieurs).</p> <p><u>Le signe *</u> signifie que l'élément peut ne pas apparaître ou bien un nombre illimité de fois (signifie zéro ou plusieurs).</p> <p><u>Le signe ?</u> signifie que cet élément (optionnel) peut apparaître au plus une fois (signifie zéro ou un).</p>

Les sous-éléments sont appelés éléments enfants.

L'ordre des sous-éléments est important et doit être respecté par la suite.

Un élément peut également être un élément vide (tout comme le sont <hr />, <br /> ou <img />). C'est-à-dire qu'il n'encadrera pas de données. Cet élément doit être déclaré de la sorte :

```
<!ELEMENT nom_élément EMPTY>
```

Lorsque l'on utilisera cet élément la syntaxe sera la suivante :

```
<nom_élément />
```

Un élément peut également (et exceptionnellement) être de contenu indéfini. Lorsque l'on est en train de développer un document XML et que l'on ne connaît que très approximativement les sous-éléments d'un élément et leur ordonnancement, on peut temporairement utiliser cette déclaration.

```
<!ELEMENT nom_élément ANY>
```

### ex007.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE etudiant [
<!ELEMENT etudiant (nom,prenom+,redoublant?, resultats*)>
  <!ATTLIST etudiant
    id CDATA #REQUIRED
    status (en-cours|clos|demission) "en-cours">
  <!ELEMENT nom (#PCDATA)>
  <!ELEMENT prenom (#PCDATA)>
  <!ELEMENT redoublant (#PCDATA)>

  <!ELEMENT resultats (matiere, coefficient, note)>
    <!ELEMENT matiere (#PCDATA)>
    <!ELEMENT coefficient (#PCDATA)>
    <!ELEMENT note (#PCDATA)>

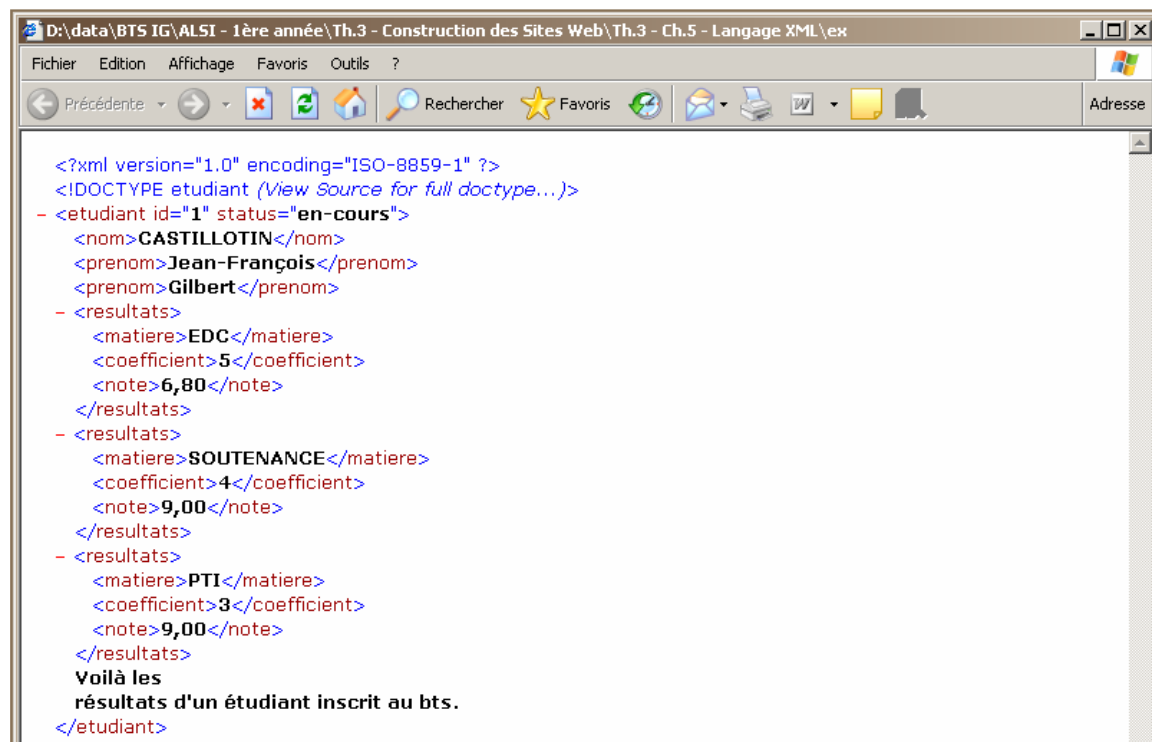
<!ENTITY description "résultats d'un étudiant inscrit au bts.">
]>

<etudiant id="1">
  <nom>CASTILLOTIN</nom>
  <prenom>Jean-François</prenom>
  <prenom>Gilbert</prenom>
  <resultats>
    <matiere>EDC</matiere>
    <coefficient>5</coefficient>
    <note>6,80</note>
  </resultats>
  <resultats>
    <matiere>SOUTENANCE</matiere>
    <coefficient>4</coefficient>
    <note>9,00</note>
  </resultats>
</etudiant>
```

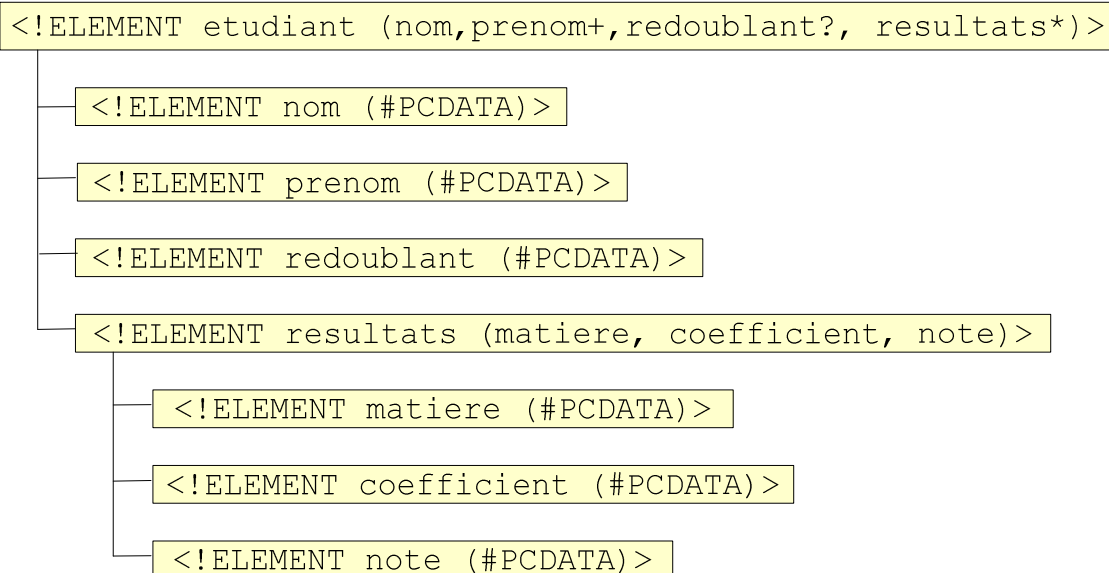
```

<matiere>PTI</matiere>
<coefficient>3</coefficient>
<note>9,00</note>
</resultats>
Voilà les &description;
</etudiant>

```



Dans ce nouvel exemple on aura pu remarquer que les éléments s’imbriquent et peuvent contenir eux-mêmes d’autres éléments enfants.



#### A.4.4. DTD externe

Le document commence à devenir long et on va séparer la DTD du document XML. Il est plus judicieux d'agir de la sorte d'une manière générale, surtout si on veut mutualiser une DTD pour plusieurs documents, plusieurs entités, plusieurs périodes... .

##### **ex008.xml**

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE etudiant SYSTEM "ex008.dtd">

<etudiant id="1">
  <nom>CASTILLOTIN</nom>
  <prenom>Jean-François</prenom>
  <prenom>Gilbert</prenom>
  <resultats>
    <matiere>EDC</matiere>
    <coefficient>5</coefficient>
    <note>6,80</note>
  </resultats>
  <resultats>
    <matiere>SOUTENANCE</matiere>
    <coefficient>4</coefficient>
    <note>9,00</note>
  </resultats>
  <resultats>
    <matiere>PTI</matiere>
    <coefficient>3</coefficient>
    <note>9,00</note>
  </resultats>
  Voilà les &description;
</etudiant>
```

##### **ex008.dtd**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT etudiant (nom,prenom+,redoublant?, resultats*)>
  <!ATTLIST etudiant
    id CDATA #REQUIRED
    status (en-cours|clos|demission) "en-cours">
  <!ELEMENT nom (#PCDATA)>
  <!ELEMENT prenom (#PCDATA)>
  <!ELEMENT redoublant (#PCDATA)>

  <!ELEMENT resultats (matiere, coefficient, note)>
    <!ELEMENT matiere (#PCDATA)>
    <!ELEMENT coefficient (#PCDATA)>
    <!ELEMENT note (#PCDATA)>

<!ENTITY description "résultats d'un étudiant inscrit au bts.">
```

## A.5. Notions avancées

### A.5.1. Déclarations des documents

Tous les documents XML valides doivent commencer par la déclaration `<?xml?>`. Cette déclaration doit contenir le numéro de version XML utilisé. La version la plus récente d'XML est la 1.1, mais si on n'utilise pas les dernières modifications le consortium W3C recommande d'utiliser la 1.0 comme nous le faisons dans tous nos exemples (`version="1.0"`).

On déclare l'encodage utilisé (`encoding="ISO-8859-1"`). Eventuellement on peut indiquer la déclaration de document autonome (`standalone="no"`) dans tous les cas où il y aurait un sous-ensemble externe de DTD. L'encodage Latin-1 correspond à l'alphabet de l'Europe de l'ouest et de l'Amérique latine. Il peut être éventuellement fait appel à l'encodage UTF-8 (codage des caractères sur 8bits) ou UCS-2 (codage des caractères sur 32bits, appelé Unicode, soit 65536 possibilités). Le choix de l'encodage peut également être important au moment de l'enregistrement du document (sous Notepad++ par exemple) soit en ANSI soit en UTF-8, au risque de produire des résultats erronés selon les interpréteurs choisis (par exemple Firefox 1.5 et IE 6). Le bloc-notes (notepad) sauvegarde systématiquement en ANSI.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
```

Les documents contiennent après la déclaration XML une déclaration de type de document qui relie le document à sa ou ses DTD.

Les DTD sont soit externes soit internes. Les DTD externes offrent l'avantage de pouvoir plus librement gérer un grand nombre de documents. Une DTD externe est prioritaire sur une DTD interne.

La déclaration d'une DTD externe se fait de la façon suivante :

```
<!DOCTYPE students PUBLIC "-//unidentified-one.net//DTD students//EN"
"http://unidentified-one.net/xml/students.dtd">
```

On utilise PUBLIC pour signifier qu'une DTD est destinée à un grand nombre de documents ou différents sites. On utilise SYSTEM quand une DTD est destinée à un site Web unique, et est développée localement. Le chemin indiqué est alors local.

```
<!DOCTYPE students SYSTEM "students.dtd">
```

```
<!DOCTYPE students SYSTEM "xml/students.dtd">
```

Une DTD publique suit le standard suivant :

- Le signe + si la DTD est approuvée par un comité, le signe – sinon,
- // suivis du nom du propriétaire,
- // suivis du type de document (DTD ou TEXT),
- Un espace et le nom du document,
- // suivis de l'identificateur de langue (EN pour English... selon la norme ISO 639).

On peut ensuite insérer des commentaires entre les déclarations `<!--` et `-->` :

```
<!-- Voici des commentaires -->
```

## A.5.2. Approfondissement du concept d'élément

En continuant de structurer notre exemple, on obtient une hiérarchisation des données telle que nous la connaissons par le biais des outils d'analyse du système d'information.

### ex009.xml

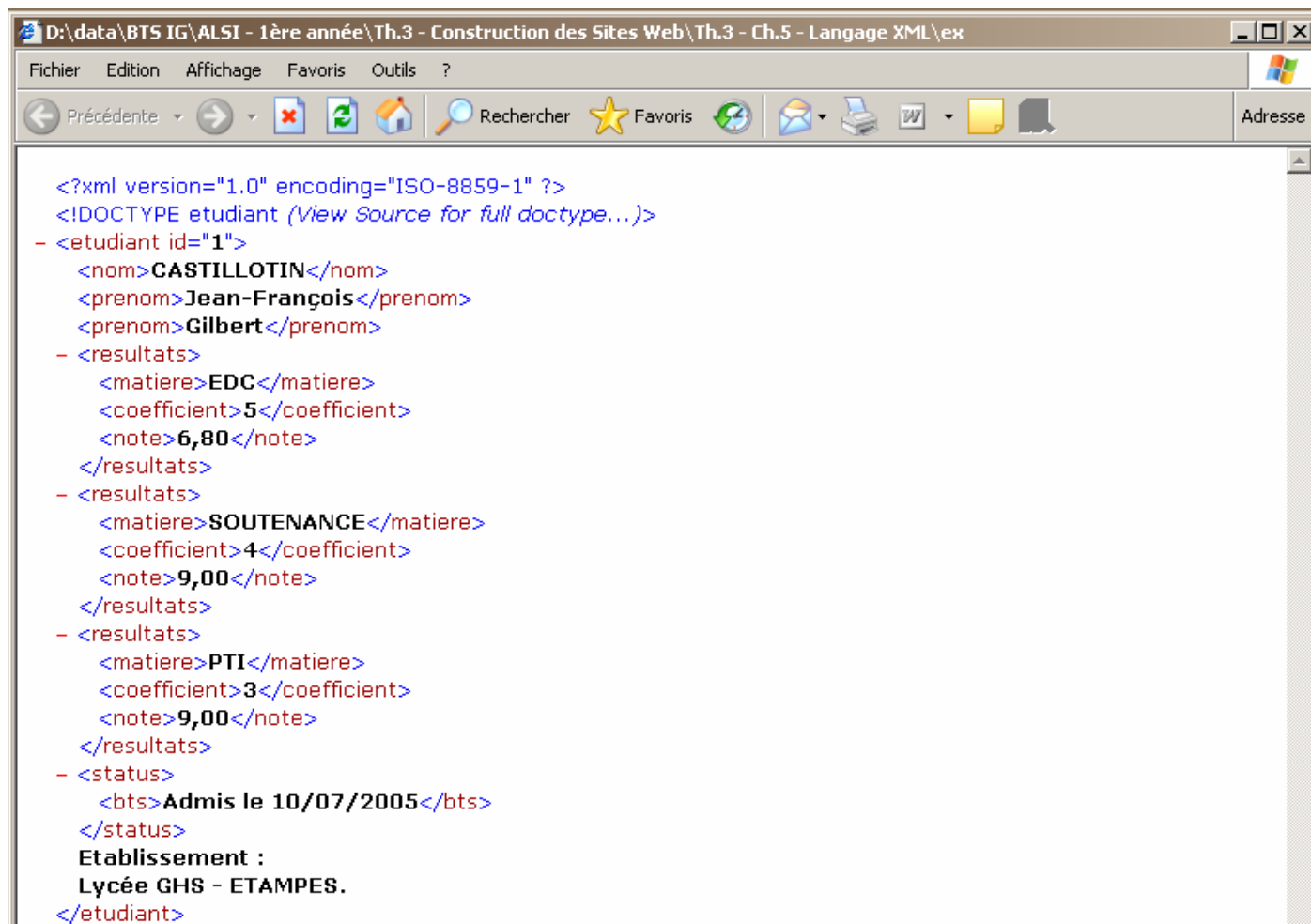
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE etudiant SYSTEM "ex009.dtd">

<etudiant id="1">
  <nom>CASTILLOTIN</nom>
  <prenom>Jean-François</prenom>
  <prenom>Gilbert</prenom>
  <resultats>
    <matiere>EDC</matiere>
    <coefficient>5</coefficient>
    <note>6,80</note>
  </resultats>
  <resultats>
    <matiere>SOUTENANCE</matiere>
    <coefficient>4</coefficient>
    <note>9,00</note>
  </resultats>
  <resultats>
    <matiere>PTI</matiere>
    <coefficient>3</coefficient>
    <note>9,00</note>
  </resultats>
  <status>
    <bts>Admis le 10/07/2005</bts>
  </status>
  Etablissement : &description;
</etudiant>
```

### ex009.dtd

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT etudiant (nom, prenom+, resultats*, status)>
<!ATTLIST etudiant
  id CDATA #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT redoublant (#PCDATA)>
<!ELEMENT resultats (matiere, coefficient, note)>
  <!ELEMENT matiere (#PCDATA)>
  <!ELEMENT coefficient (#PCDATA)>
  <!ELEMENT note (#PCDATA)>
<!ELEMENT status (bts | demission | refuse | redoublement)?>
  <!ELEMENT bts (#PCDATA)>
  <!ELEMENT demission (#PCDATA)>
  <!ELEMENT refuse (#PCDATA)>
  <!ELEMENT redoublement (#PCDATA)>

<!ENTITY description "Lycée GHS - ETAMPES.">
```



On peut donc proposer plusieurs éléments au choix des utilisateurs (avec le séparateur |), ici :

```
<!ELEMENT status (bts | demission | refuse | redoublement)?>
```

On peut positionner soit l'élément BTS, DEMISSION, REFUSE ou REDOUBLEMENT au plus une seule fois puisque le signe ? a été employé. On pourrait donc omettre le contenu de STATUS.

On peut également proposer un choix alternatif libre :

```
<!ELEMENT status ((#PCDATA | bts | demission | refuse | redoublement)?>
```

On peut également définir, grâce aux entités, des éléments enfants standards qui peuvent servir à ne nombreux autres éléments parents.

```

<!ENTITY % descriptif "nom | prenom | sexe">
<!ELEMENT etudiant (%descriptif;)>
<!ELEMENT professeur (%descriptif;)>

```

### A.5.3. Approfondissement du concept d'entité

Nous avons utilisé des entités lors de tous nos exemples. Voici la dernière déclarée dans la DTD ex009.dtd :

```
<!ENTITY description "Lycée GHS - ETAMPES.">
```

Nous avons donc cantonné l'entité à un rôle de variable contenant du texte que l'on peut insérer dans un document XML comme dans ex009.xml :

```
Etablissement : &description;
```

L'entité a d'autres fonctions bien plus utiles. On distingue deux types d'entités, les entités générales et les entités paramètres.

### A.5.3.1. Les entités générales

Les entités générales servent à enrichir un document avec des informations données par simple appel de leur nom (&nom;).

XML fournit de base cinq appels d'entités. Ces entités sont prédéfinies à l'origine parce qu'elles permettent d'insérer dans les documents XML des signes réservés au langage.

&lt;	Le signe inférieur à, le chevron ouvrant (<).
&amp;	L'esperluette (&).
&gt;	Le signe supérieur à, le chevron fermant (>).
&quot;	Les guillemets (").
&apos;	L'apostrophe (').

Les entités générales permettent de définir d'autres appels.

Ainsi on peut centraliser une information dans un document et la faire évoluer. Par exemple un numéro de version est attribué de la façon suivante :

```
<!ENTITY version "1.0.0.">
```

Tous les documents afficheront ce numéro de version à chaque ouverture. Puis, par la suite nous passons à une nouvelle version :

```
<!ENTITY version "1.2.4.">
```

Les documents qui utiliseront la DTD contenant cette déclaration d'entité afficheront la version actuelle.

On peut garder le même principe avec notre exemple 009 enrichi par des balises :

```
<!ENTITY description "<b>Lycée GHS - ETAMPES.</b>">
```

Qui pourrait être :

```
<!ENTITY description "<b><i>(c)</i> 2006 - Lycée GHS - ETAMPES.</b>">
```

Puis dès le passage en 2007 :

```
<!ENTITY description "<b><i>(c)</i> 2007 - Lycée GHS - ETAMPES.</b>">
```

On pourrait avoir toute une section avec des balises...

### A.5.3.2. Les entités générales externalisées

Les entités pourraient figurer dans un autre document. Cela serait bien plus pratique, surtout si l'on a une entité qui contient beaucoup de texte, et si ce texte est susceptible d'être mis à jour spécifiquement. D'où l'utilisation des entités générales externalisées.

Déclaration de l'entité :

```
<!ENTITY bas_de_page SYSTEM "/xml/basdepage.xml">
```

Contenu de l'entité :

```
<?xml encoding="ISO-8859-1"?>
<hr/>
<b><i>(c)</i> 2007 - Lycée GHS - ETAMPES.</b><br/>
```

On aura remarqué que la version d'XML a été omise, mais pas l'encodage. Seule la déclaration de l'encodage est obligatoire dans une déclaration d'entité.



### A.5.3.3. Les entités paramètres

Les entités paramètres ne servent qu'au mécanisme « interne » aux DTD. Elles sont destinées aux DTD externes entre-elles. Elles permettent de référencer des informations utilisées à d'autres DTD.

Nous avons déjà utilisé le principe des entités paramètre :

```
<!ENTITY % descriptif "nom | prenom | sexe">
<!ELEMENT etudiant (%descriptif;)>
<!ELEMENT professeur (%descriptif;)>
```

Dans ce cas le paramètre DESCRIPTIF sert à la construction des éléments de la DTD, mais on pourrait penser que cette structure d'éléments enfants (NOM PRENOM SEXE) pourrait être utile à beaucoup d'autres éléments, donc réutilisable. C'est donc l'objet des entités paramètres. Elles sont appelées avec (&nom).

Un dernier exemple :

#### **ex010.xml**

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE etudiant SYSTEM "ex010b.dtd">

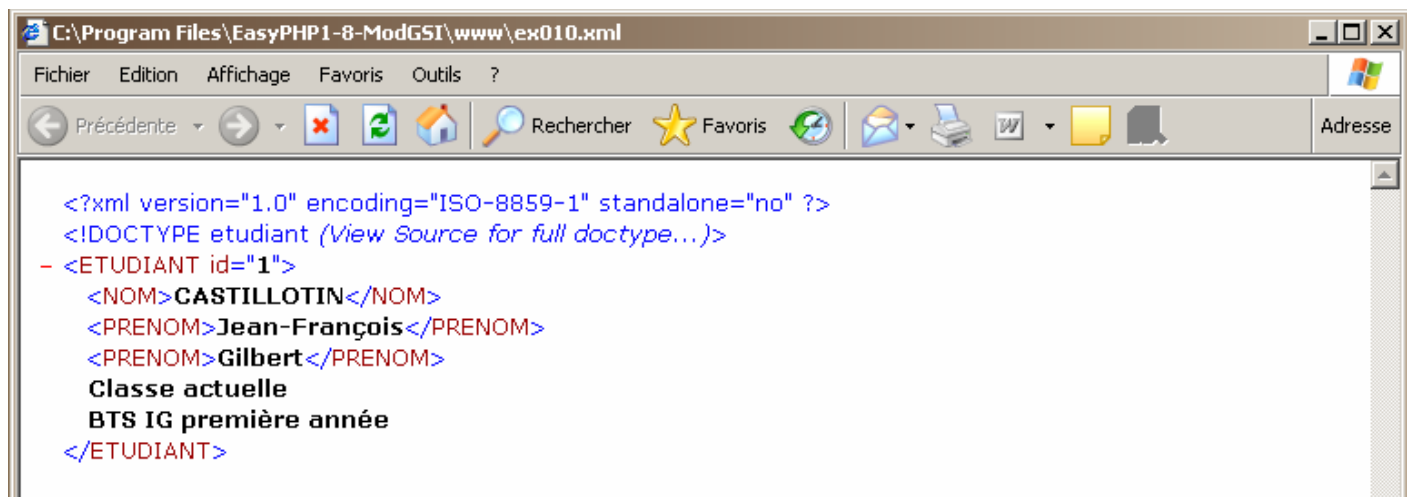
<ETUDIANT id="1">
<NOM>CASTILLOTIN</NOM>
<PRENOM>Jean-François</PRENOM>
<PRENOM>Gilbert</PRENOM>
Classe actuelle &ig1;
</ETUDIANT>
```

#### **ex010b.dtd**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ENTITY % classe SYSTEM "ex010a.dtd">
%classe;
<!ELEMENT etudiant (nom, prenom+)>
<!ATTLIST etudiant
    id CDATA #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
```

#### **ex010a.dtd**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ENTITY ig1 "BTS IG première année">
<!ENTITY ig2 "BTS IG deuxième année">
<!ENTITY agp1 "BTS AG PME-PMI première année">
<!ENTITY agp2 "BTS AG PME-PMI deuxième année">
<!ENTITY e1 "BTS Electrotechnique première année">
<!ENTITY e2 "BTS Electrotechnique deuxième année">
```



Ce dernier exemple montre dans la DTD 010b.dtd la déclaration d'une entité externe définie par 010a.dtd.

## A.6. Les espaces de noms

Après l'étude des principales fonctionnalités d'XML on se rend bien compte de sa portée, mais l'on se sent un peu petit face à la montagne, tant l'univers à pénétrer peut paraître grand, infini. On s'interroge aussi sur la redondance des travaux qui vont être fournis par toutes les organisations dans le monde.

En effet, la bibliothèque d'Etampes (dans l'Essonne) pourrait avoir envie d'établir une DTD de façon à référencer ses auteurs et leurs ouvrages. Parallèlement, la bibliothèque François Mitterrand de Paris pourrait réaliser un travail similaire et donc redondant. Peut-être également qu'il y aurait redondance (partielle) si la Société des Auteurs et Compositeurs Dramatiques (SACD) concevait une DTD recensant les auteurs uniquement.

Il existe donc, de part le monde, des DTD construites de façon à ce que l'on puisse les utiliser en respectant la définition de leurs éléments, assurant ainsi une uniformité dans le traitement d'un domaine particulier (les mathématiques, les auteurs, les peintres, le langage XHTML, les statistiques...). On peut donc incorporer à un document XML des éléments issus d'espaces de nommage différents en incorporant leur DTD et en préfixant les éléments de leur origine. On n'est donc pas obligé de réinventer la poudre !

Souvenez-vous d'avoir utilisé un espace de nommage avec XHTML :

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
...
</html>
```

L'élément racine de cet espace de nommage était HTML, et tous les éléments utilisés faisaient partie des éléments définis dans cet espace de nommage. Cet espace de noms est donc assimilable à une ressource du Web que l'on utilise à sa guide, partiellement ou entièrement.

Pour identifier un espace de nom on utilise un identificateur : URI (Uniform Resource Identifiers). Certes l'URI ressemble à une URL (Uniform Resource Locator), dans notre cas c'est bien une URL. Cependant il ne s'agit pas d'une adresse au même sens que celle utilisée dans les navigateurs. La différence peut paraître confuse ou bien subtile, mais l'on comprendra simplement qu'on désigne ici un espace de nommage et non l'adresse d'un site du Web.

Voici un exemple d'utilisation de deux espaces de nommage :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
  <head>
    <title>Rectangle SVG</title>
  </head>
  <body>
    <h1>Rectangle</h1>
    <svg xmlns="http://www.w3.org/2000/svg" width="100px" height="80px">
      <!-- <rect y="6px" x="5px" width="80px" height="40px" style="stroke:
#0000FF; fill: #FF0000"/>-->
      <ellipse rx="110" ry="130" />
    </svg>
  </body>
</html>
```

On reconnaît l'espace de nommage XHTML que l'on a amplement utilisé au chapitre précédent, et un espace nommé SVG qui permet notamment de dessiner des formes.

Des organismes, comme l'INSEE proposent des espaces de noms (<http://xml.insee.fr/schema>) qui peuvent permettre de décrire par exemple les départements, les cantons, les villes...

Les entités d'un espace de nom sont souvent décrites en utilisant leur espace de nom suivi de : et du nom de l'élément. Comme par exemple avec XSLT `<xsl:template match="promotion">`.

Les espaces de nom sont nombreux et nous verrons mieux leur utilisation avec XSLT.

## A.7. Qu'est ce qu'XML, finalement ?

---

Voilà une dernière partie bien étrange. Pourquoi intervient-elle maintenant ?

Il semble utile de définir plus en avant XML maintenant qu'il a été présenté. Le fonctionnement est connu. L'architecture l'est également. Il est plus simple à présent de définir ce qu'est XML.

Il y a plusieurs approches.

### A.7.1. Première approche

Cette première approche ne définit en rien XML, il s'agit juste d'une évidence mise en avant pour montrer l'application concrète d'XML.

XML a permis d'unifier et de formaliser le HTML dérivant. Le langage XHTML qui emprunte le formalisme XML est un langage « d'avenir » alors qu'HTML ne l'est pas. On comprend bien qu'avec un formalisme qui diffère d'un navigateur Internet à un autre, des balises qui sont spécifiques ou qui produisent des effets différents, impossible de bâtir quelque chose de stable.

XHTML est basé sur des recommandations intangibles qui en font la grande fiabilité. C'est grâce à XML et la définition des éléments inclus dans XHTML que cela a pu se faire. On le comprend mieux à présent.

### A.7.2. Seconde approche

XML permet une pérennisation des bases documentaires mondiales.

On voit bien qu'un document texte d'hier est peu compatible avec un document texte d'aujourd'hui, comme les feuilles de calcul et de nombreux autres documents. On a tous été agacé par des incompatibilités entre des versions différentes d'un document, et à chaque fois, il faut recommencer et réadapter le document au logiciel alors que ça devrait être l'inverse. Quand un document est conçu il est souvent structuré selon les normes du jour. Mais pour combien de temps ? Dans 2000 ans, ou même dans 100 ans, 10 ans ?

XML est bâti pour la durée. C'est-à-dire qu'il permet d'avoir un format stable dans le temps. Le document est dual : il porte avec lui le contenu et la définition de l'agencement du contenu.

Ce document n'est plus le fruit d'un seul logiciel, mais de tous les logiciels. On peut le concevoir chez soit sur un traitement de texte, et il sera lu à distance par un tableur. Bon nombre d'outils bureautiques intègrent déjà un support d'XML.

C'est là l'apport majeur d'XML.

### A.7.3. Troisième approche

XML permet la mise en forme des données, des informations structurées.

Il y a eu un constat de fait en 1999 : à l'époque, 70% des données étaient situées en dehors des bases de données. Ce constat mettait en avant deux choses :

- Les bases de données ne sont pas les moyens exclusifs de contenir de l'information structurée (un tableur peut aussi le faire par exemple) ;
- Un grand nombre de données sont dans des formats fondamentalement différents qui les rendent impossibles à mettre en rapport.

XML est la solution qui procure la compatibilité entre les données. Non seulement XML permet de structurer les données, mais également de les définir de façon identique quelque soit le lieu, l'époque et le medium utilisé.

## B. Description et mise en forme du contenu d'XML

Les éléments de XML s'attachent à décrire la sémantique du contenu. Ce contenu doit ensuite être formaté et présenté aux utilisateurs.

La couche présentation est assurée par deux principes :

- Cascading Style Sheets (CSS) ;
- Extensible Stylesheet Language - Formatting Objects (XSL-FO).

Nous verrons XSL-FO et XSLT (Extensible Stylesheet Language Transformation) par la même occasion dans le chapitre suivant.

### B.1. Présentation des données avec CSS

Nous allons à présent nous intéresser à la présentation des données avec l'aide d'une feuille de style CSS.

Nous connaissons bien le fonctionnement et l'utilité des feuilles de style grâce au chapitre précédent il ne reste donc plus qu'à les appliquer aux documents XML.

Les feuilles de style CSS restent le moyen le plus simple et le plus portable pour présenter des documents XML. XSL-FO est plus puissant encore, et permet de concevoir un grand nombre de documents plus uniquement orientés Web.

#### B.1.1. Mise en forme fixe des données

Dans cet exemple nous allons mettre en forme tous les éléments en utilisant un « mélange » de tout ce que nous connaissons. La présentation finale du document sera critiquable à bien des égards au niveau de l'esthétique...

##### **ex011.xml**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="ex011.css" type="text/css"?>
<!DOCTYPE etudiant SYSTEM "ex011.dtd">

<etudiant id="1">
  <nom>CASTILLOTIN</nom>
  <prenom>Jean-François</prenom>
  <prenom>Gilbert</prenom>
  <resultats>
    <matiere>EDC</matiere>
    <coefficient>5</coefficient>
    <note>6,80</note>
  </resultats>
  <resultats>
    <matiere>SOUTENANCE</matiere>
    <coefficient>4</coefficient>
    <note>9,00</note>
  </resultats>
  <resultats>
    <matiere>PTI</matiere>
    <coefficient>3</coefficient>
    <note>9,00</note>
  </resultats>
  <status>
    <bts>Admis le 10/07/2005</bts>
  </status>
</etudiant>
```

### ex011.dtd

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT etudiant (nom, prenom+, resultats*, status)>
  <!ATTLIST etudiant
    id CDATA #REQUIRED>
  <!ELEMENT nom (#PCDATA)>
  <!ELEMENT prenom (#PCDATA)>
  <!ELEMENT redoublant (#PCDATA)>
  <!ELEMENT resultats (matiere, coefficient, note)>
    <!ELEMENT matiere (#PCDATA)>
    <!ELEMENT coefficient (#PCDATA)>
    <!ELEMENT note (#PCDATA)>
  <!ELEMENT status (bts | demission | refuse | redoublement)?>
    <!ELEMENT bts (#PCDATA)>
    <!ELEMENT demission (#PCDATA)>
    <!ELEMENT refuse (#PCDATA)>
    <!ELEMENT redoublement (#PCDATA)>
```

### ex011.css

```
etudiant
{
  background: #808080;
  display: block;
  font-size: 14;
  font-family: verdana;
}

nom
{
  background: #0020c0;
  font-family: arial;
  color: #ffffff;
}

prenom
{
  font-size: 12;
  font-family: arial;
  color: #ffff00;
}

resultats
{
  display: block;
  background: #002060;
  border: inset;
  border-width: thin;
  width: 300px;
}

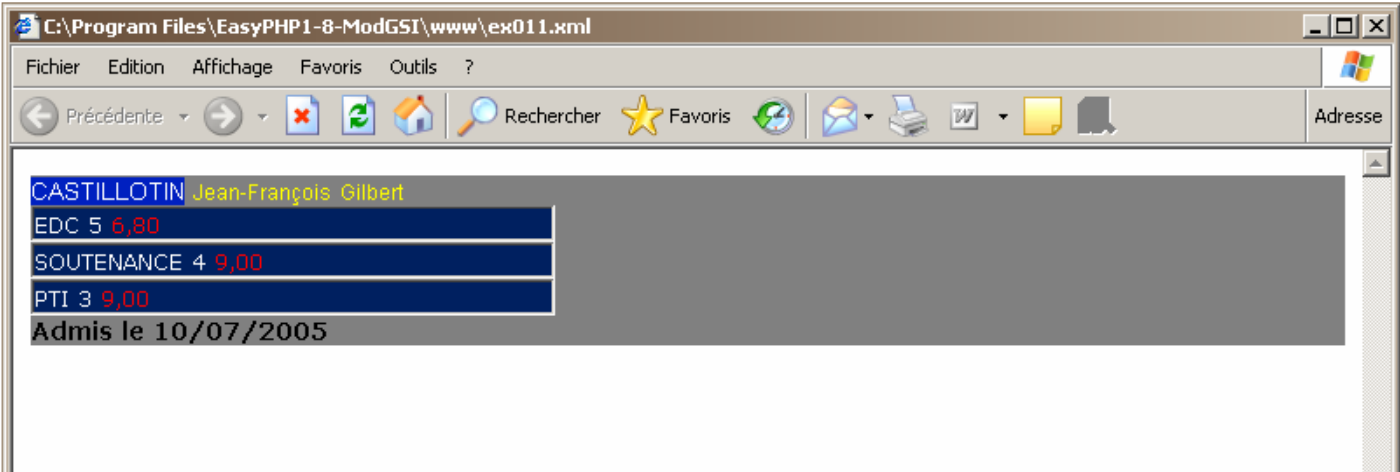
matiere, coefficient
{
  color: #ffffff;
  font-size: 12;
}

note
{
  color: #ff0000;
  font-size: 12;
}

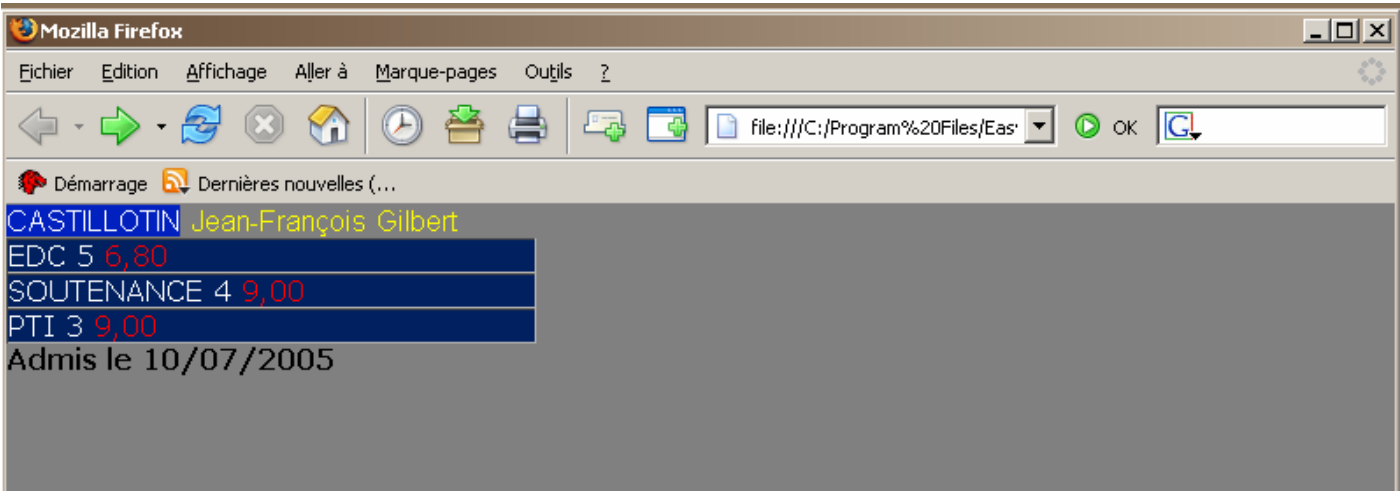
status
{
  display: block;
```

```
font-weight: bold;
}
```

Le traitement de la CSS bouleverse l’affichage que nous avons précédemment :



Le résultat dans Firefox est sensiblement le même :



L’élément ETUDIANT étant l’élément racine et la feuille de style lui attribuant un fond gris (background: #808080;) on peut comprendre que Firefox ait mis toute la page de cette couleur. D’un autre côté, l’élément ETUDIANT a aussi été déclaré avec une présentation monobloc (display: block;) on peut comprendre que l’Internet Explorer est mis uniquement le bloc correspondant à l’étudiant dans la couleur grise.  
 On en conclura donc qu’aussi bien Firefox, qu’Internet Explorer interprètent au mieux le code XML sans cependant faire de miracles.

Il existe également dans CSS une structuration sous forme de tableau grâce à **display** :

#### ex011b.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="ex011b.css" type="text/css"?>
<!DOCTYPE etudiant SYSTEM "ex011b.dtd">

<etudiant id="1">
  <nom>CASTILLOTIN</nom>
  <prenom>Jean-François</prenom>
  <prenom>Gilbert</prenom>
  <res>
    <resultats>
      <matiere>EDC</matiere>
      <coefficient>5</coefficient>
      <note>6,80</note>
    </resultats>
    <resultats>
      <matiere>SOUTENANCE</matiere>
      <coefficient>4</coefficient>
      <note>9,00</note>
    </resultats>
    <resultats>
      <matiere>PTI</matiere>
      <coefficient>3</coefficient>
      <note>9,00</note>
    </resultats>
  </res>
  <status>
    <bts>Admis le 10/07/2005</bts>
  </status>
</etudiant>
```

#### ex011b.dtd

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT etudiant (nom, prenom+, res, status)>
  <!ATTLIST etudiant
    id CDATA #REQUIRED>
  <!ELEMENT nom (#PCDATA)>
  <!ELEMENT prenom (#PCDATA)>
  <!ELEMENT redoublant (#PCDATA)>
  <!ELEMENT res (resultats*)>
  <!ELEMENT resultats (matiere, coefficient, note)>
    <!ELEMENT matiere (#PCDATA)>
    <!ELEMENT coefficient (#PCDATA)>
    <!ELEMENT note (#PCDATA)>
  <!ELEMENT status (bts | demission | refuse | redoublement)?>
    <!ELEMENT bts (#PCDATA)>
    <!ELEMENT demission (#PCDATA)>
    <!ELEMENT refuse (#PCDATA)>
    <!ELEMENT redoublement (#PCDATA)>
```

Dans la DTD et le document XML l'ajout de l'élément RES est purement factuel ici, pour pouvoir déclarer le tableau dans la feuille de style. L'élément RES correspond donc au tableau, l'élément RESULTATS correspond aux lignes et les éléments MATIERE, COEFFICIENT et NOTE sont les cellules.

## ex011b.css

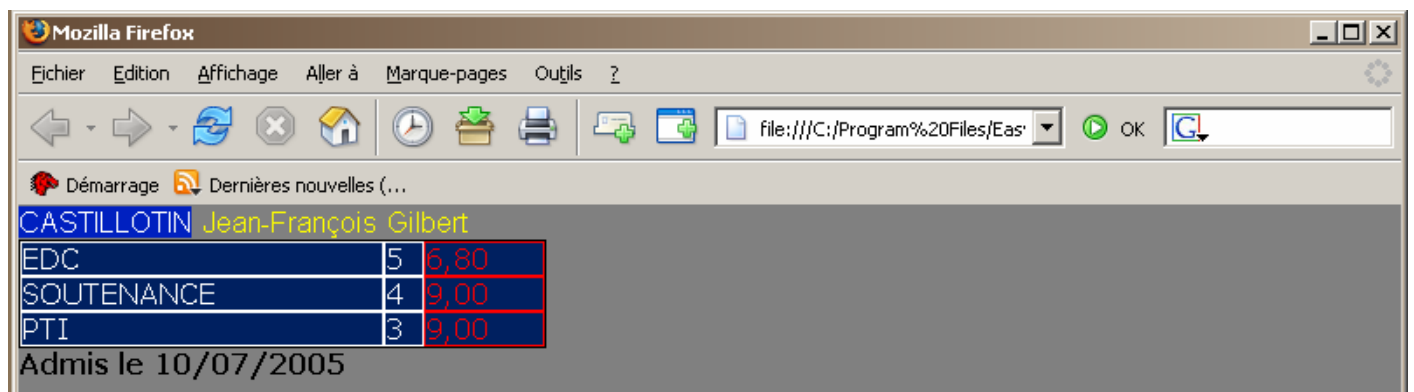
```
...
res
{
  background: #002060;
  border-style: solid;
  border-width: thin;
  width: 300px;
  display: table;
}

resultats
{
  display: table-row;
  border-style: solid;
  border-width: 1px;
}

matiere
{
  color: #ffffff;
  font-size: 12;
  border-style: solid;
  border-width: 1px;
  display: table-cell;
}

coefficient
{
  color: #ffffff;
  font-size: 12;
  border-style: solid;
  border-width: 1px;
  display: table-cell;
}

note
{
  color: #ff0000;
  font-size: 12;
  border-style: solid;
  border-width: 1px;
  display: table-cell;
}
...
```





## B.1.2. Mise en forme avec les sélecteurs

Les sélecteurs permettent d'obtenir une mise en forme plus « dynamique » avec les feuilles de style. En fonction de la présence d'un attribut, ou selon l'une de ses valeurs, le style pourra être différent.

Dans l'exemple qui suit, l'élément NOTE contient un attribut NIVEAU qui lorsqu'il a la valeur « moyen » aura un style particulier :

```
note[niveau="moyen"]
{
  color: #ff0000;
  font-weight: bold;
}
```

De la même manière l'élément BTS contient un attribut STAT qui lorsqu'il contient le mot « Admis » aura un style particulier :

```
bts[statut~="admis"]
{
  font-size: 16;
  background: #222222;
  color: #00e060;
}
```

### **ex012.xml**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="ex012.css" type="text/css"?>
<!DOCTYPE etudiant SYSTEM "ex012.dtd">

<etudiant id="1">
  <nom>CASTILLOTIN</nom>
  <prenom>Jean-François</prenom>
  <prenom>Gilbert</prenom>
  <resultats>
    <matiere>EDC</matiere>
    <coefficient>5</coefficient>
    <note niveau="insuffisant">6,80</note>
  </resultats>
  <resultats>
    <matiere>SOUTENANCE</matiere>
    <coefficient>4</coefficient>
    <note niveau="moyen">9,00</note>
  </resultats>
  <resultats>
    <matiere>PTI</matiere>
    <coefficient>3</coefficient>
    <note niveau="moyen">9,00</note>
  </resultats>
  <status>
    <bts statut="admis">Admis le 10/07/2005</bts>
  </status>
</etudiant>
```

## ex012.dtd

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

<!ELEMENT etudiant (nom, prenom+, resultats*, status)>
  <!ATTLIST etudiant
    id CDATA #REQUIRED>
  <!ELEMENT nom (#PCDATA)>
  <!ELEMENT prenom (#PCDATA)>
  <!ELEMENT resultats (matiere, coefficient, note)>
    <!ELEMENT matiere (#PCDATA)>
    <!ELEMENT coefficient (#PCDATA)>
    <!ELEMENT note (#PCDATA)>
  <!ATTLIST note
    niveau CDATA #REQUIRED>
  <!ELEMENT status (bts | demission | refuse | redoublement)?>
    <!ELEMENT bts (#PCDATA)>
  <!ATTLIST bts
    stat CDATA #REQUIRED>
  <!ELEMENT demission (#PCDATA)>
  <!ELEMENT refuse (#PCDATA)>
  <!ELEMENT redoublement (#PCDATA)>
```

## ex012.css

```
etudiant
{
  background: #808080;
  display: block;
  font-size: 14;
  font-family: verdana;
}

nom
{
  background: #0020c0;
  font-family: arial;
  color: #ffffff;
}

prenom
{
  font-size: 12;
  font-family: arial;
  color: #ffff00;
}

resultats
{
  display: block;
  background: #002060;
  border: inset;
  border-width: thin;
  width: 300px;
}

matiere, coefficient
{
  color: #ffffff;
  font-size: 12;
}

note
{
  color: #00e060;
  font-size: 12;
}
```

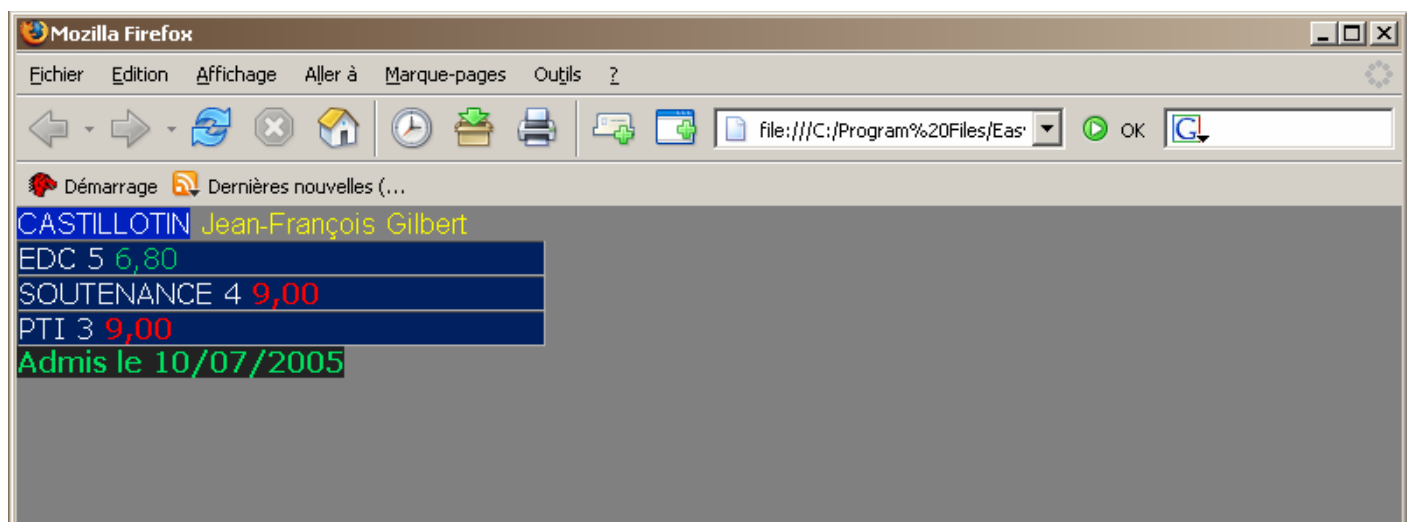
```

note[niveau="moyen"]
{
  color: #ff0000;
  font-weight: bold;
}

status
{
  display: block;
  font-weight: bold;
}

bts[stat~="admis"]
{
  font-size: 16;
  background: #222222;
  color: #00e060;
}

```



Les sélecteurs ont donc plusieurs options (cf. Chapitre sur XHTML) :

note[niveau="moyen"] {...}	Élément contenant une valeur d'attribut particulière.
note[niveau]	Élément contenant un attribut NIVEAU.
*[niveau]	Tous les éléments ayant un attribut NIVEAU.
bts[stat~="admis"] {...}	Élément contenant un mot particulier dans leur valeur d'attribut.

Les éléments qui possèdent des ID uniques peuvent être désignés par leur numéro d'ID et avoir une mise en forme dédiée dans la feuille de style.

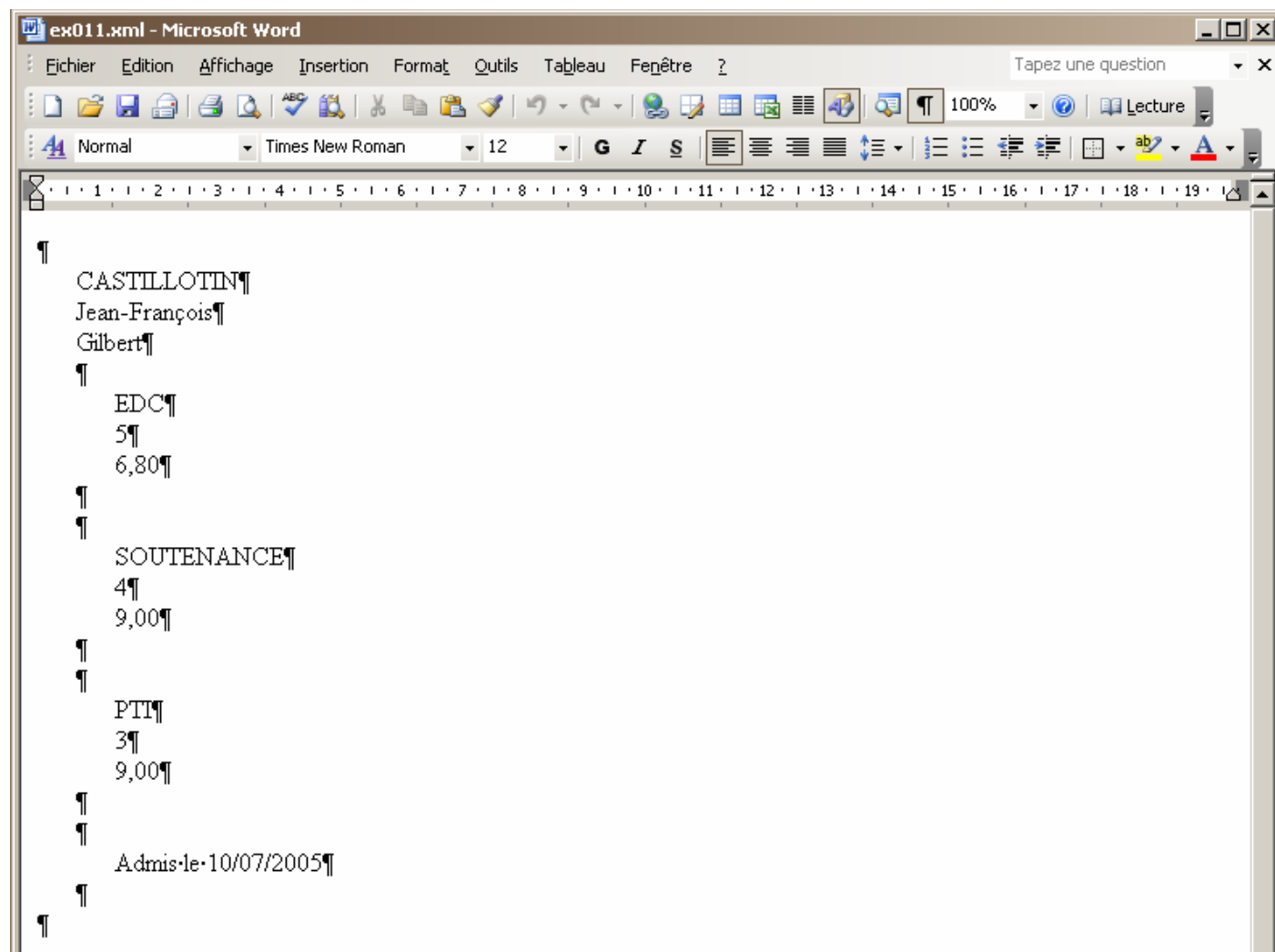
D'autres types de sélecteurs ont été présentés dans le chapitre dédié à XHTML.

## C. Utilisations des documents XML

Nombreuses sont les utilisations d'XML, la plupart des applications bureautiques acceptent et peuvent produire des documents dans ce format.

### C.1. Applications bureautiques et XML

Par curiosité on peut ouvrir le document **ex011b.xml** dans un traitement de texte pour voir ce qu'il en a compris :



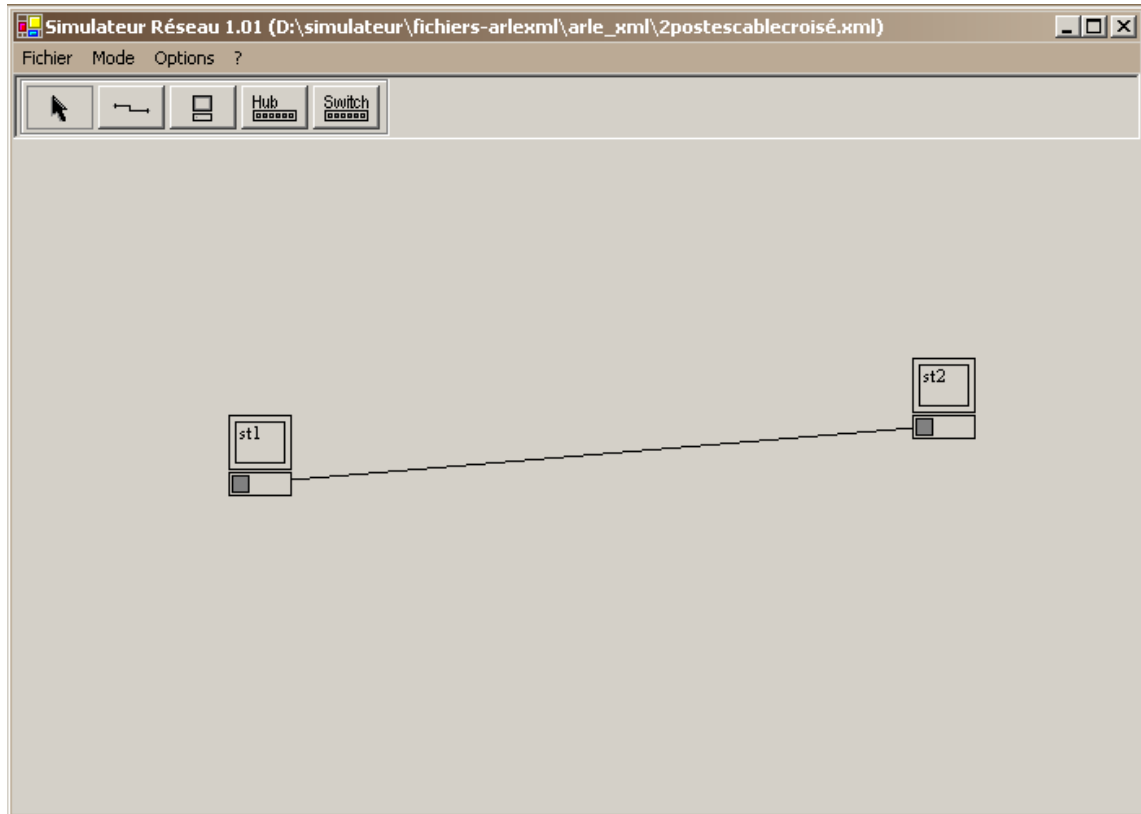
Dans un tableau, l'affichage est différent, mais on retrouve bien la structure :

1	/etudiant									
2	/@id	/@id/#agg	/nom	/prenom	/resultats/coefficient	/resultats/coefficient/#agg	/resultats/matiere	/resultats/note	/resultats/note/#agg	/status/bts
3	1	1	CASTILLOTIN	Jean-François						Admis le 10/07/2005
4	1		CASTILLOTIN	Gilbert						Admis le 10/07/2005
5	1		CASTILLOTIN		5	5	EDC	6,8	6,8	Admis le 10/07/2005
6	1		CASTILLOTIN		4	4	SOUTENANCE	9	9	Admis le 10/07/2005
7	1		CASTILLOTIN		3	3	PTI	9	9	Admis le 10/07/2005
8										
9										

## C.2. Exemple du simulateur réseau

Un autre exemple intéressant est le simulateur réseau du Certa. Ce simulateur est fourni avec de nombreux exemples de configuration de réseau au format XML.

Nous avons utilisé le modèle **2postescablecroisé.xml** :



Voici le document XML correspondant, la station st1 :

```
<?xml version="1.0"?>
...
<station nomNoeud="st1">
  <idNoeud>1</idNoeud>
  <nbPointsConnexion>1</nbPointsConnexion>
  <Location.X>132</Location.X>
  <Location.Y>169</Location.Y>
  <ConfigIpStation>
    <passerelle>0.0.0.0</passerelle>
    <routageActif>False</routageActif>
    <serveurDns>0.0.0.0</serveurDns>
    <Routes nbLignes="0" />
    <CacheARP nbLignes="0" />
    <FichierHosts nbLignes="0" />
  </ConfigIpStation>
  <carte adresseMac="mac01">
    <PosDemoEmission.X>299</PosDemoEmission.X>
    <PosDemoEmission.Y>367</PosDemoEmission.Y>
    <ConfigIpCarte>
      <clientDhcp>False</clientDhcp>
      <adresse>0.0.0.0</adresse>
      <masque>0.0.0.0</masque>
    </ConfigIpCarte>
  </carte>
</station>
...
```

Voici le document XML correspondant, le câble croisé :

```
...
<cable>
  <type>croise</type>
  <longueur>30</longueur>
  <noeudPointA>1</noeudPointA>
  <portPointA>1</portPointA>
  <noeudPointB>2</noeudPointB>
  <portPointB>1</portPointB>
</cable>
</simRes>
```

Toute la structure de ce fichier est bien du XML. Ce format est donc utilisé dans des domaines variés.

Quelques exemples :

- La Fédération Internationale d’Escrime (FIE) utilise se format pour transmettre les résultats des compétitions ;
- L’INSEE regroupe ses informations statistiques avec XML ;
- Les fils RSS utilisent XML pour indiquer les évolutions des sites, ou comme sur [www.virusraq.com](http://www.virusraq.com) (dans /rss/virus.xml) recenser les différents virus ;
- Un grand nombre de logiciels utilisent XML pour leur configuration ou leur fonctionnement : Winamp par exemple utilise XML pour ses skins et la mise en forme de ses playlists.



Olivier Mondet  
<http://unidentified-one.net>