Serge Fleury Université Paris 3 - ILPGA

eXtensible Markup Language

Une introduction

Sources :

- cf Références bibliographiques
- Pages sur le web (nombreuses), cf lien

Sommaire

Sommaire (2)

Partie 1

- Préambule

Retour Sommaire

Introduction

- Buts de cet exposé
  - Présentation de XML : un nouveau paradigme internet
  - Parallèle avec SGML et HTML
- Applications
  - Publications de documents sur le web
  - Préparation de corpus pour le TAL
  - Présentation et manipulation d ’outils XML

Références bibliographiques

- Manuels de référence
  - "XML, langage et applications", Alain Michard, Editions Eyrolles, 1999 (noté désormais [XML 99])
  - "GUIDEXPRESS XML", Andréas Petrausch, Editions Micro Application, 2000 (noté désormais [XML 00])
  - "XML in a nutshell", manuel de référence, E.R. Harold & W.S. Means, traduction de T. Broyer, P. Ensarguet, A. Ketterlin, O'REILLY, 2001 (désormais noté [XML 01]

Références (suite)

- "Le langage XML" par John Bosak et Tim Bray, Pour la Science, juillet 1999, "Le langage XML" (Bosak & Bray 1999)
- Autres manuels
  - http://tecfa.unige.ch/guides/xml/slides/xml.pdf
- Recommandations du W3C
  - www.w3.org/TR/REC-xml
  - babel.alis.com/web_ml/xml/REC-xml.fr.html

Ressources sur le web pour débuter

- Initiation à XML
  - Un article d'Emmanuel Lazinier, "XML expliqué aux débutants" sur le site chez.com. Ce site francophone se propose de diffuser et de vulgariser vers la communauté francophone tout ce qui concerne XML, désigné comme "le langage qui va provoquer demain la deuxième révolution Internet".
  - http://www.chez.com/xml/initiation/index.htm
- XML pour débutants
  - Vous trouverez de bonnes ressources pour les débutants, en français, chez ABCDOC.
  - http://dossiers.abcdoc.net/xml/p06.html

Liens

- Pages sur le site TAL-Paris3
  - **Cours 9 (Portail TAL)**
- Pages officielles sur l ’hypertoile
  - **www.xml.org**
  - **www.w3.org**
  - **www.softwareag.com/xml/**
  - **www.xmlsoftware.com**
  - **www.xml.com/pub**
  - etc.

Sites développeurs (1)

- Microsoft MSDN
  - Une fraction du site Microsoft Site Builder Network est consacrée à un atelier XML: informations, un tutorial, des démonstrations, des scénario, des outils et des détails sur les fonctionnalités XML de Internet Explorer 4.0 et 5.0.
  - http://msdn.microsoft.com/xml/default.asp
- IBM
  - Nombreux didacticiels, des informations sur la plate-forme XML d'IBM et sur les initiatives d'IBM relatives à Java
  - http://www.ibm.com/developer/xml/

Sites développeurs (2)

- Sun Microsystems
  - "Java Technology and XML" porte sur l'intégration des technologies Java et XML, voir notamment White Paper, Platform Technology, Downloads, FAQ.
  - http://jsp.java.sun.com/xml/
- Softquad
  - ressources sur XML, XLL et XLS et des outils.
  - http://www.softquad.com/
- PROJECTCOOL.COM
  - Un glossaire sur XML...
  - http://www.projectcool.com/guide/xml/resources.html

Sites développeurs (3)

- CHEZ.COM
  - http://www.chez.com/xml/
- XML.COM
  - http://www.xml.com/
- JCLARK.COM
  - http://www.jclark.com/xml/
- FINETUNING.COM
  - http://www.finetuning.com/xml.html
- xmlTree
  - http://www.xmltree.com/

Logiciels/Utilitaires pour XML

- *Cf* Site TAL pour un répertoire complet et adresses des outils à tester
  - Navigateurs/Editeurs :
    - IE 5, Netscape 6, XML Notepad, XML-Spy, Amaya, Exml, Zveno Swish XML Editor, xmlcookop etc.
  - Voir aussi :
    - Majix : programme Java pour convertir documents WORD (i.e RTF) vers XML
      - www.tetrasix.com

Outils XML

- Outils utilisés dans le cadre de ce cours
  - XML Script demo (*cf* infra)
  - Editeur : XML Cooktop
    - Application gratuite pour Windows: Environnement de développement pour XML, XSL, Xpath :http://xmleverywhere.com/cooktop/
  - Analyseurs syntaxiques : Xerces-J, Xalan...
  - Navigateurs :Internet Explorer, Netscape 6, Opera...

Documents XML disponibles

- Sur le site TAL, un certain nombre de ressources XML pour travailler :
  - Un fictionnaire
  - Un dictionnaire
  - Le corpus prématurés 96
  - Ressources PATR « xmlisées »
  - "Le Petit Prince" (extrait)
  - Un discours de Miterrand (extrait)
  - On pourra aussi travailler avec les ressources en ligne disponibles : pièces de shakespeare, documents du projet Gutenberg,...
  - Etc.

Un peu d ’histoire : rappels

- SGML (ISO standard 1986)
  - norme internationale pour la représentation des documents
  - langage trop complexe pour le WEB
- HTML (1990…)
  - application « pauvre » de SGML pour le WEB
  - langage sémantiquement figé
- XML (1996…)
  - une version simplifiée de SGML adaptée au WEB
  - disponible avec IE 5, Netscape 6

Les limites HTML

- HTML ne permet pas de "marquer" les informations en fonction de leur signification
  - si l'on souhaite présenter des informations concernant la chimie ou la météorologie
    - pas de balises ATOM ou MOLECULE pour les chimistes, ni de balises CARTE ou TEMPERATURE pour les météorologues.
  - Il est hors de question de songer à intégrer de telles balises dans la norme HTML :
    - la prise en compte des spécificités de tous les métiers et de toutes les spécialisations est impossible

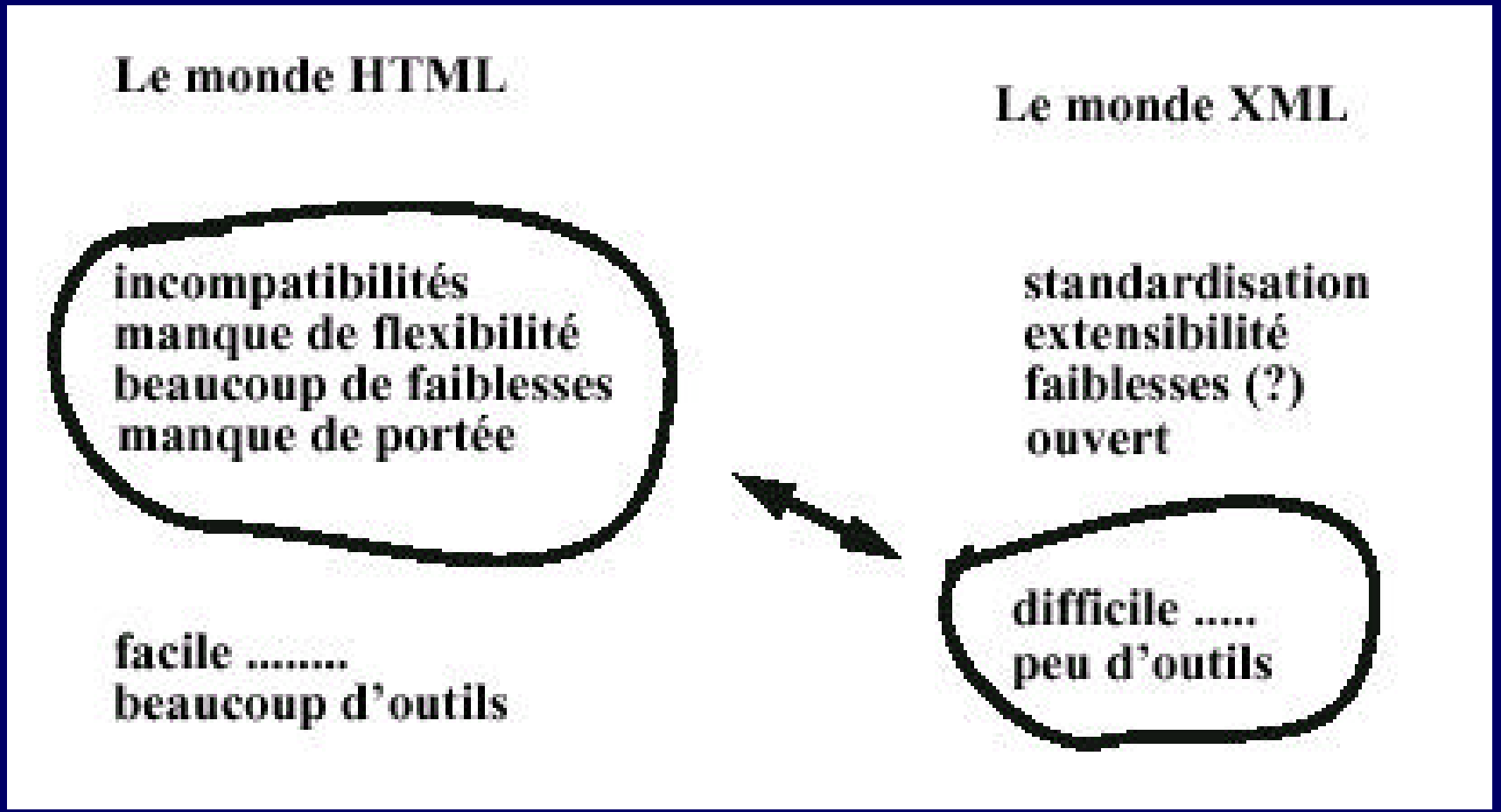
Les limites HTML (2)

- HTML est fait pour être affiché dans un « browser »
  - pas pour échanger de l ’information entre programmes
- HTML est faible pour l ’hypertexte
- HTML ne respecte pas de sémantique formelle
  - on peut construire un document avec des balises H2, et sans balises H1.
- HTML est faible pour décrire le contenu d ’information
  - en HTML on ne sait faire que de la recherche « full-text » => beaucoup de bruit

XML, version simplifiée de SGML pour le Web

- La norme SGML n'a pas été retenue en raison de sa lourdeur et de sa complexité :
  - un très grand nombre de fonctionnalités qui sont très rarement utilisées
  - le support des différents jeux de caractères internationaux est quelque peu léger.
- Le W3C a créé une norme plus simple, dérivée de SGML, et renforcée là ou SGML comporte des faiblesses. Cette norme est la norme XML
  - Groupe de travail animé par John Bosak

HTML vs XML


Le monde HTML
Le monde XML
incompatibilités
manque de flexibilité
beaucoup de faiblesses
manque de portée
standardisation
extensibilité
faiblesses (?)
ouvert
facile .....
beaucoup d'outils
difficile .....
peu d'outils

Normalisation des documents

- Structurer l ’information
  - pour la retrouver facilement
  - l ’utiliser dans des applications
- Faire des hypertextes efficaces
- Afficher et imprimer de manière flexible
- Un format normalisé pour
  - diffuser, échanger, stocker, chercher...

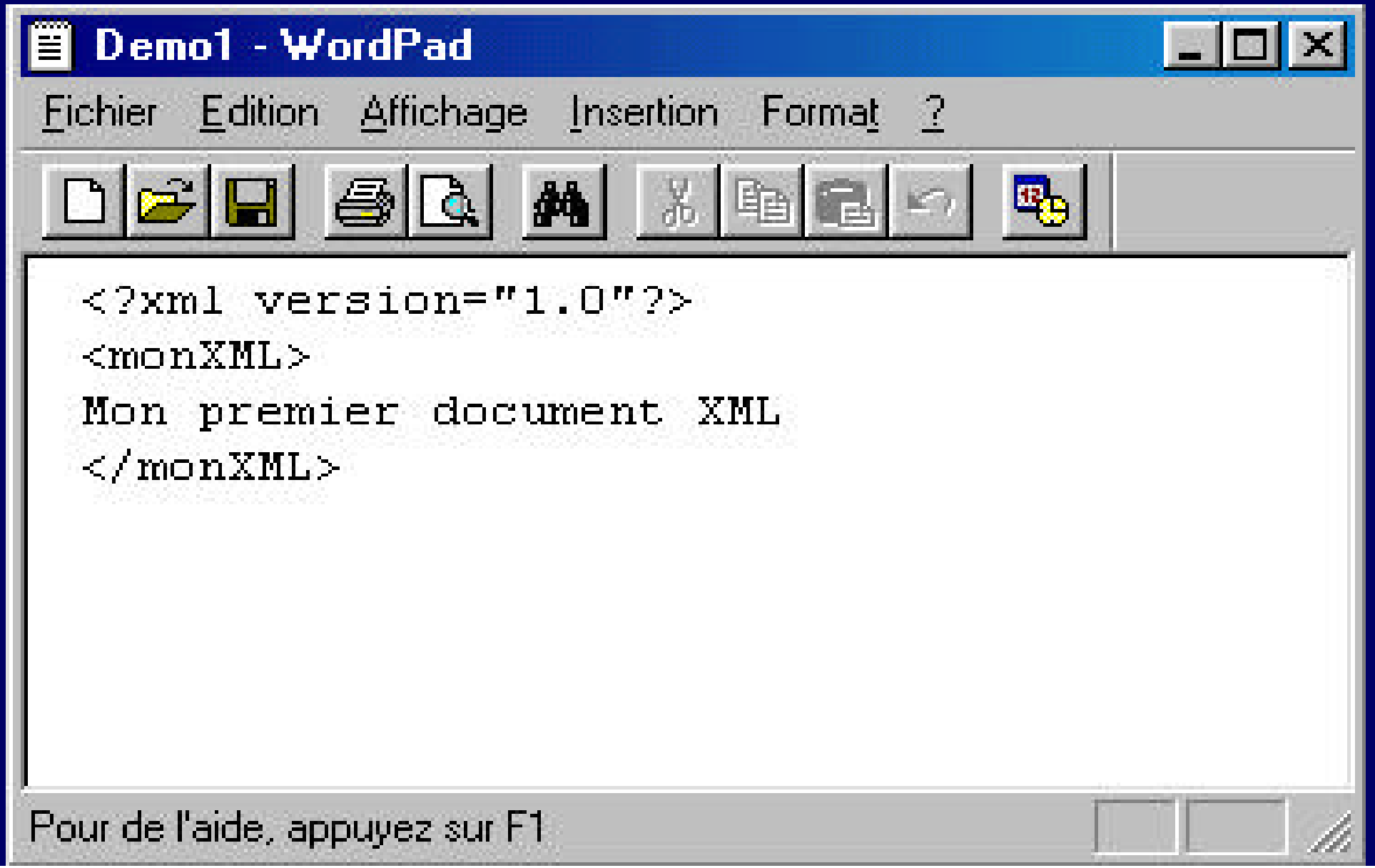
XML en le pratiquant (1)

- XML par l ’exemple (1)

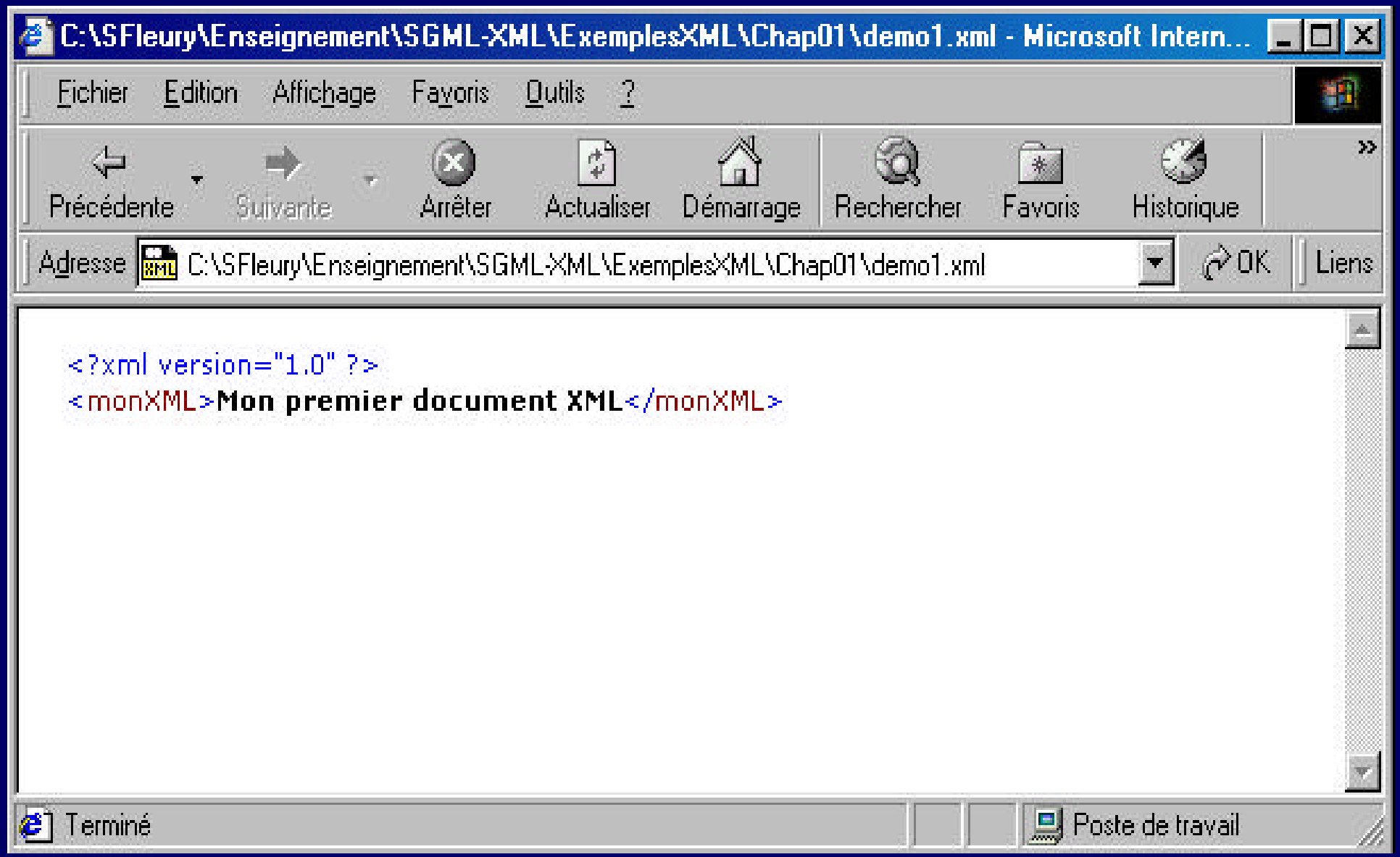
Introduction par l ’exemple

- Exemple construit à partir de [XML 00]
  - Génération d ’un document XML
  - Affichage du document
  - Ajout d ’une feuille de style
  - raffinements...

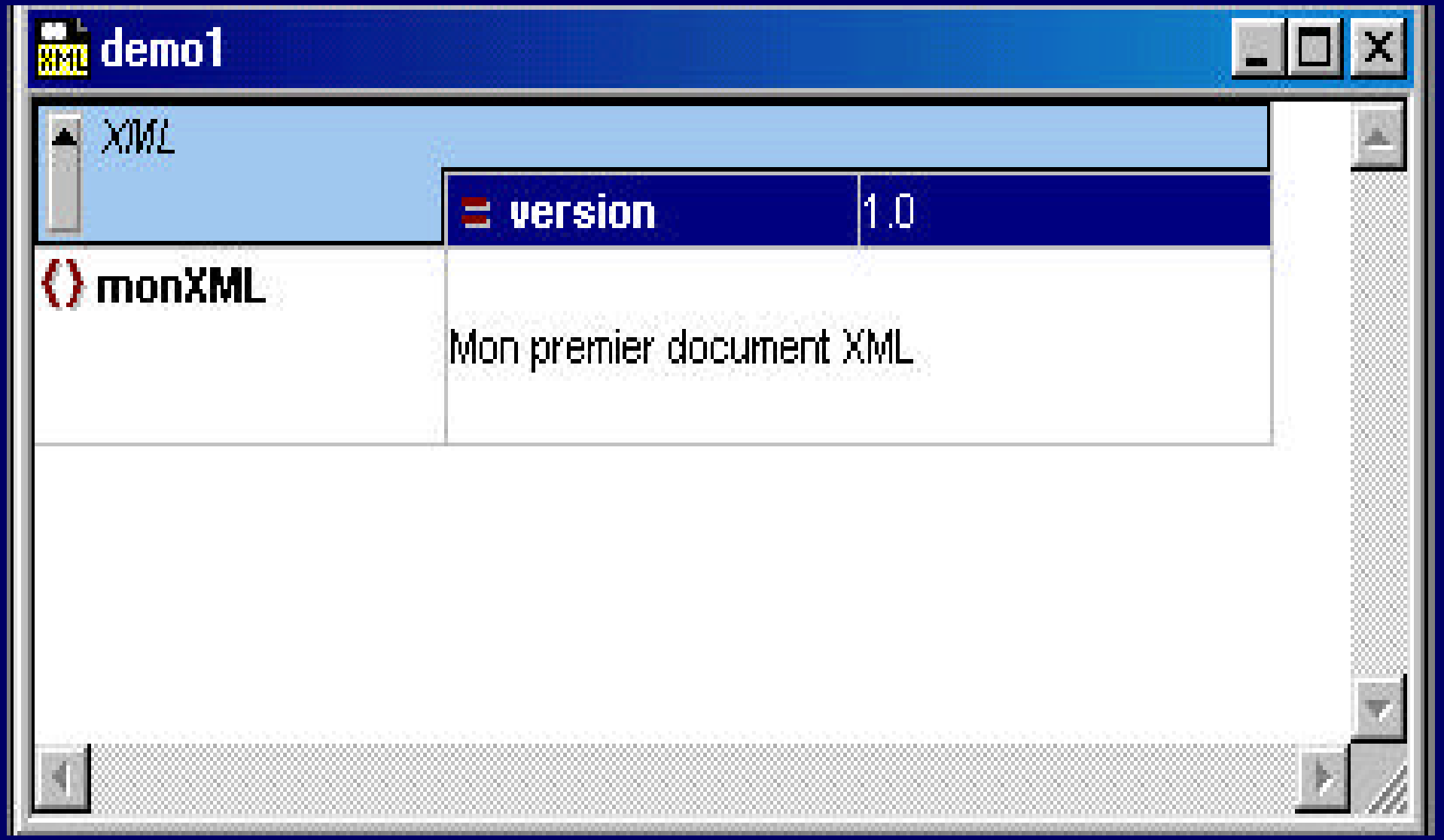
Création avec WordPad



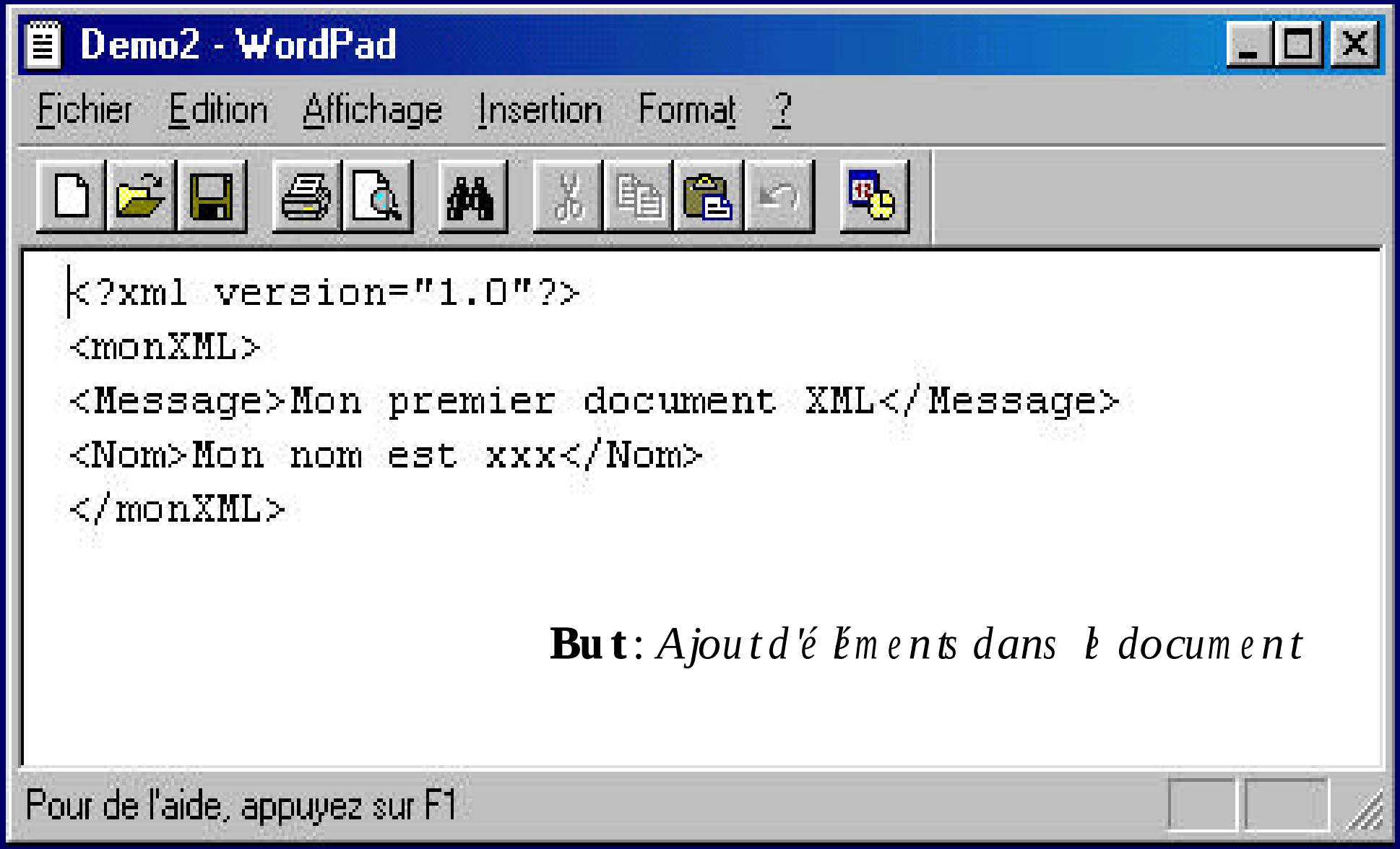
Visualisation du document avec IE5



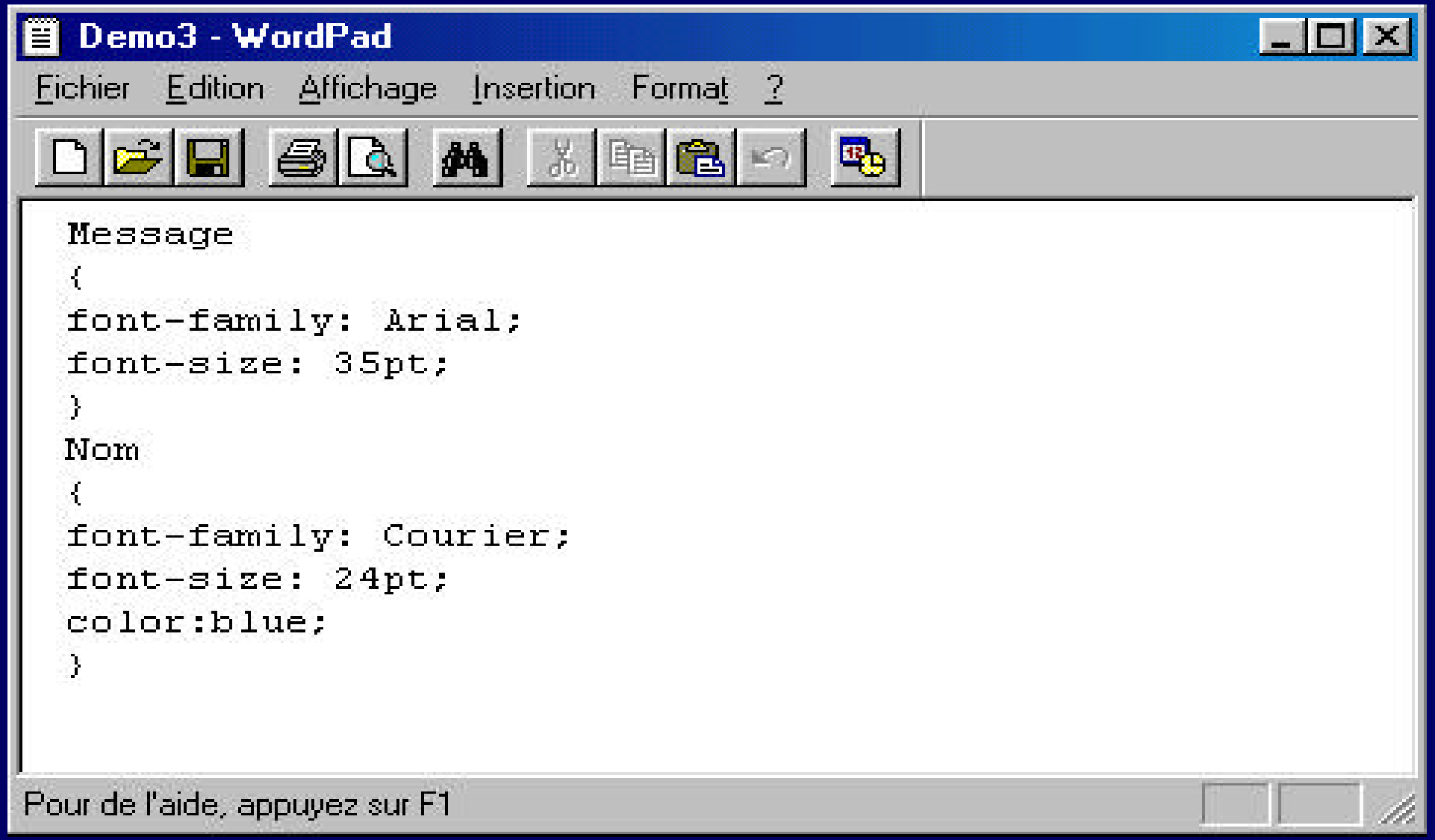
Visualisation du document avec XML-Spy



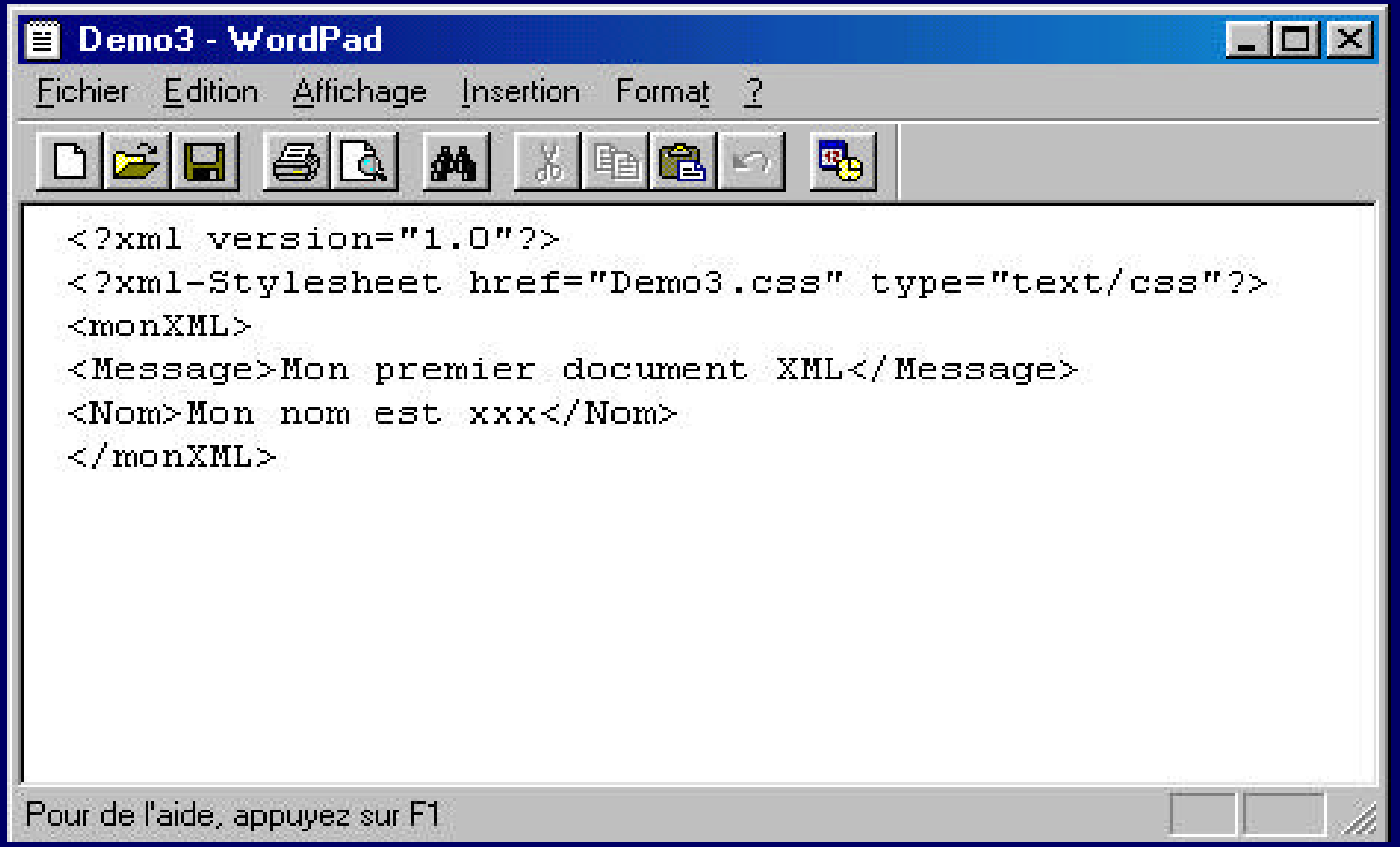
Ajout d ’une feuille de style (1)



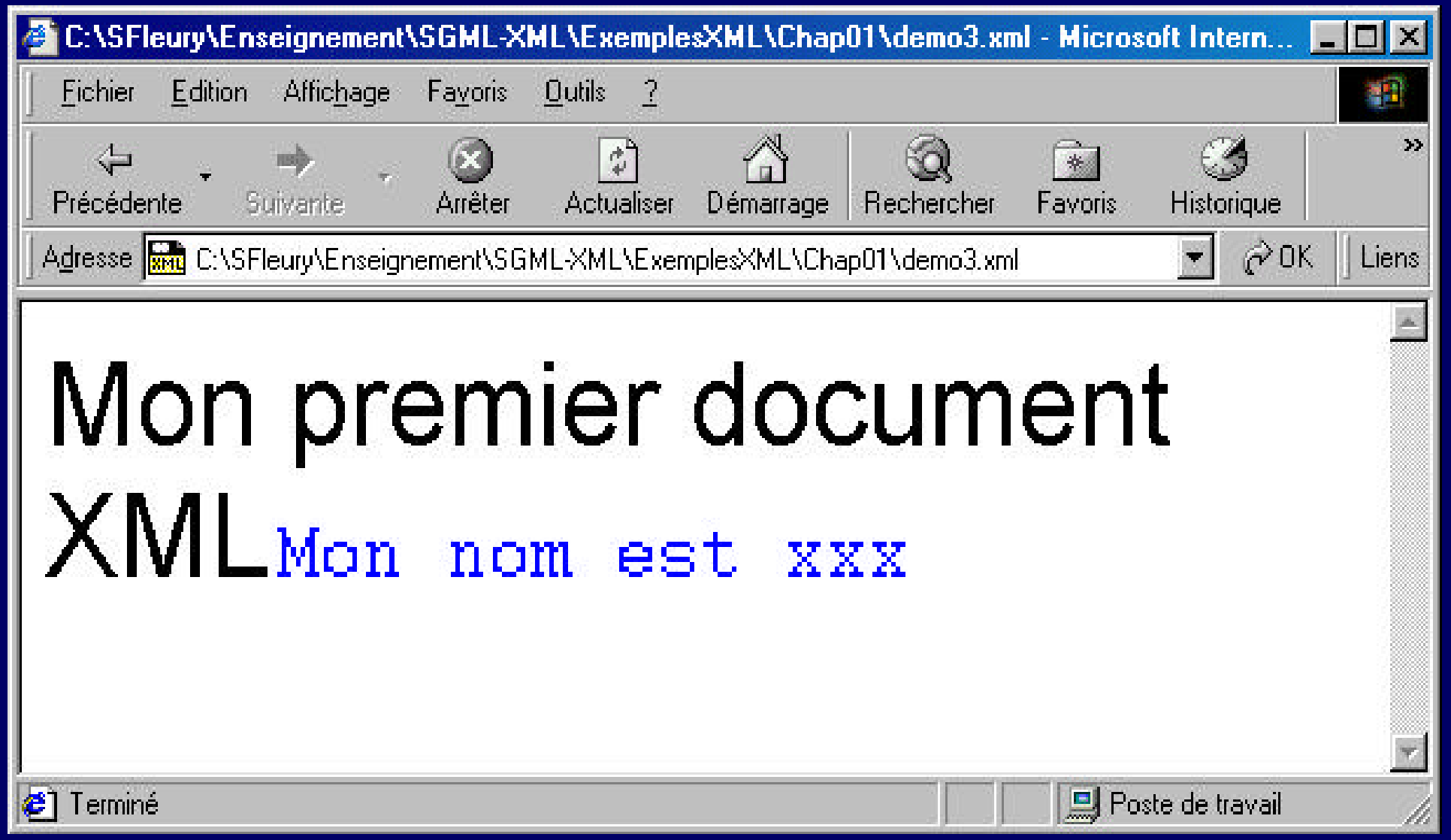
Ajout d ’une feuille de style (2)



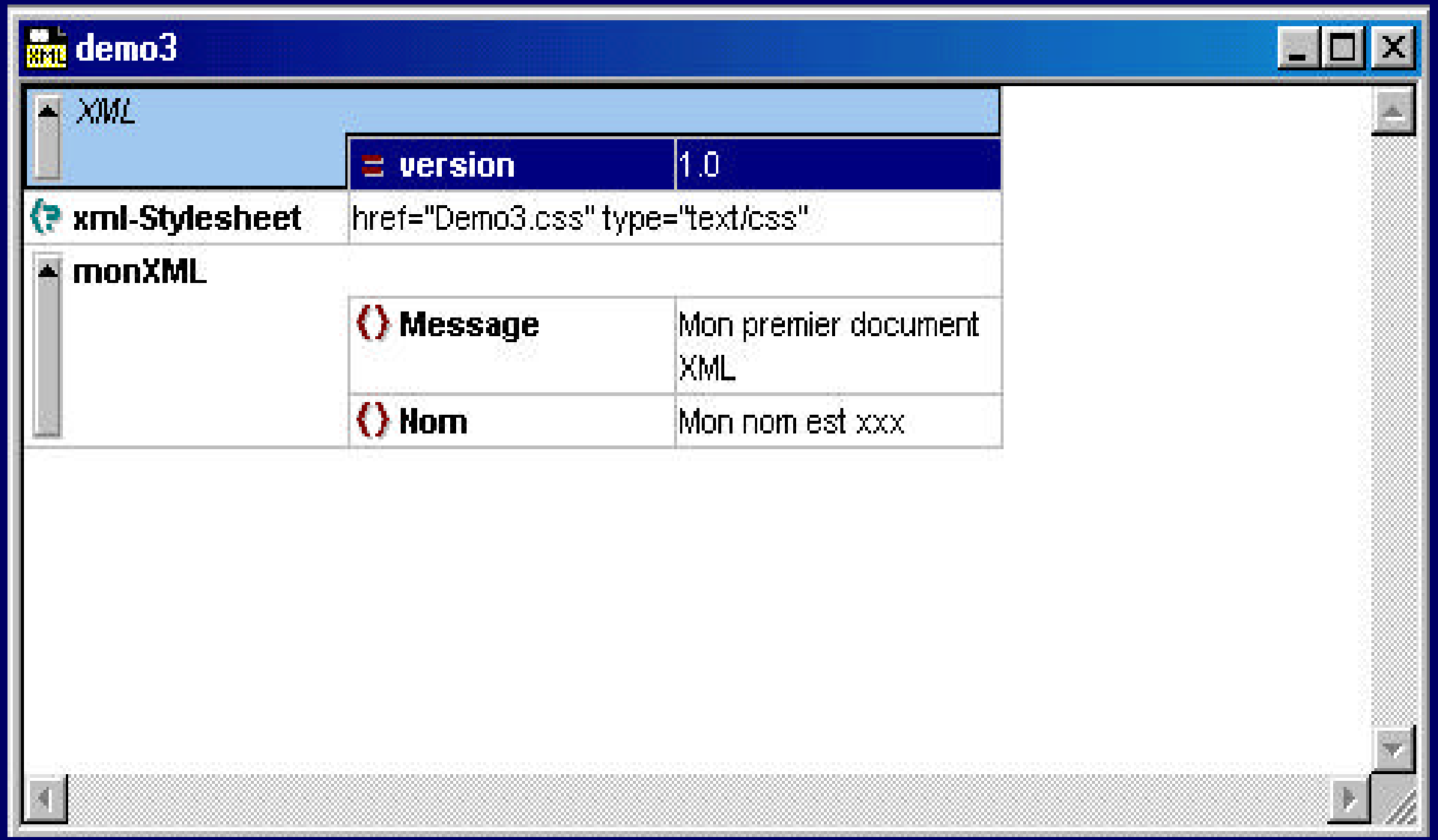
Ajout d ’une feuille de style (3)



Ajout d ’une feuille de style (4)



Ajout d ’une feuille de style (5)



Partie 2

- Principes de XML

Retour Sommaire

XML : un métalangage à balises

- XML est un métalangage à balises pour les documents texte
  - Les données sont incluses dans un document XML sous forme de chaînes de caractères et elles sont délimitées par des marqueurs textuels qui les décrivent
  - Une unité particulière de donnée et son marqueur est appelée un élément
- Spécifications XML
  - Ensemble de documents qui précisent les règles de formation des documents XML

Lecture POLY

- Leçon n°0
- Leçon n°1

Qu'attend-on d'XML ?

- Aujourd'hui le plus gros problème n'est plus le volume de connaissances ou sa vitesse de croissance, c'est l'aptitude à utiliser les connaissances qui existent quelque part
- XML est un langage qui définit d'autres langages. Il a aussi le pouvoir de donner une structure et un sens à l'information contenue dans un document
- XML fournit un format pour décrire des informations structurées, pour manipuler et échanger des données et communiquer entre les applications

Evolution de XML

- XML descend de SGML
  - SGML : à partir des travaux de C. Godfarb, E. Mosher, R. Lorie chez IBM dans les années 70
    - Langage sémantique et structurel à balises pour les documents textuels
    - HTML : application SGML pour la réalisation de pages Web
  - 1996, J. Bosak, T. Bray, C.M. Sperberg-McQueen, J. Clark... commencent à travailler sur une version simplifiée de SGML
  - Février 1998 : XML 1.0

XML 1.0

- XML 1.0 est un standard défini par une recommandation du World Wide Web Consortium (W3, organisation responsable de la normalisation pour le Web). Quoique récent, on peut dire qu'XML 1.0 est déjà largement déployé
- Dans l'environnement immédiat d'XML, il y d'autres normes dont l'état est plus ou moins avancé : XSL, XLL, XPATH...

Documents XML

- Un document XML contient du texte (jamais de données binaires)
- Il est éditable par n'importe quel programme capable de lire un fichier texte
- XML est sensible à la casse : les composants d'un document XML écrits dans une casse différente sont différents

Deux types de documents XML

- Document bien formé : obéit aux règles syntaxiques du langage XML (document correct)
- Document valide : document bien formé qui obéit en outre à une structure type définie dans une DTD

Premier document XML

```
<person>
Alan Turing
</person>
```

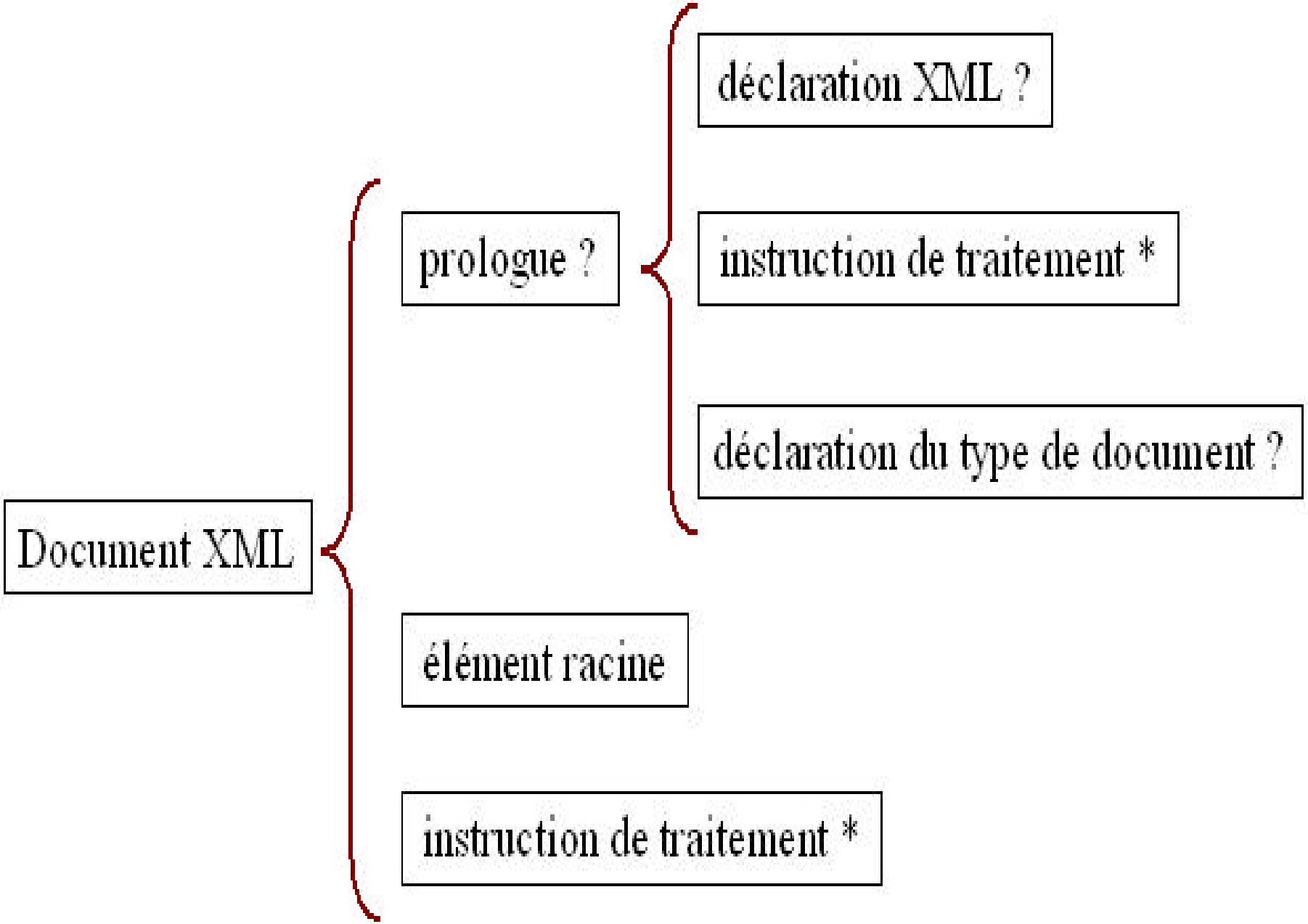
- Ce document XML est composé d'un seul élément de type person délimité par une balise ouvrante et une balise fermante
- Le contenu de cet élément est ce qui se trouve entre la balise ouvrante et la balise fermante : Alan Turing

Lecture Poly

- Leçon n°0
- Leçon n°1
- Leçon n°2

Structure d ’un document

- Un prologue
  - facultatif mais conseillé
- Un arbre d ’éléments
  - le contenu propre du document
- Commentaires et instructions de traitement
  - facultatifs, présents dans le prologue ou dans l ’arbre d ’éléments


Document XML
prologue ?
déclaration XML ?
instruction de traitement *
déclaration du type de document ?
élément racine
instruction de traitement *

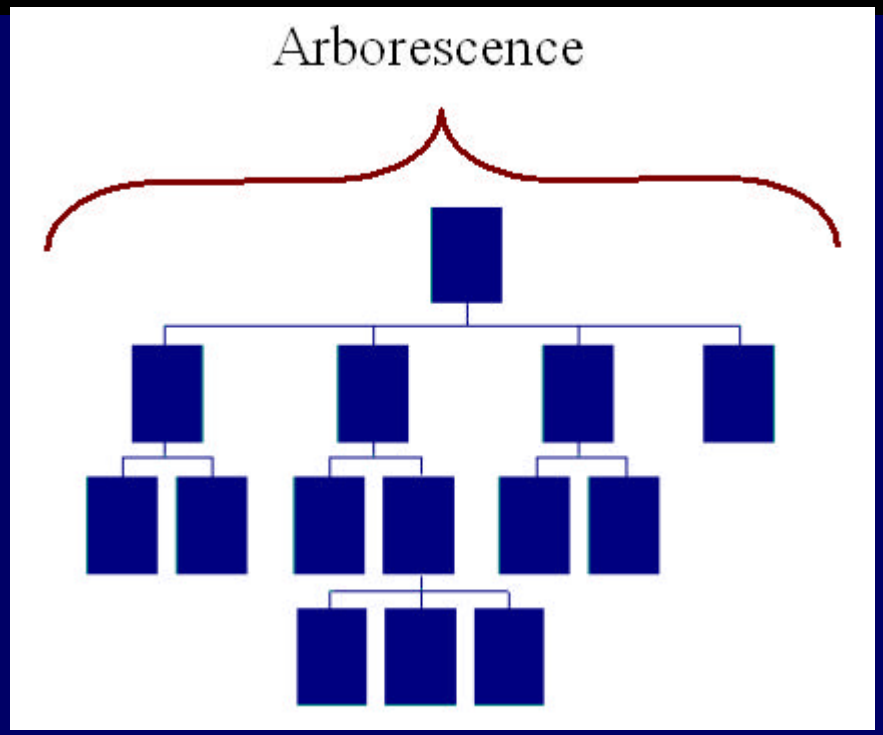
Eléments

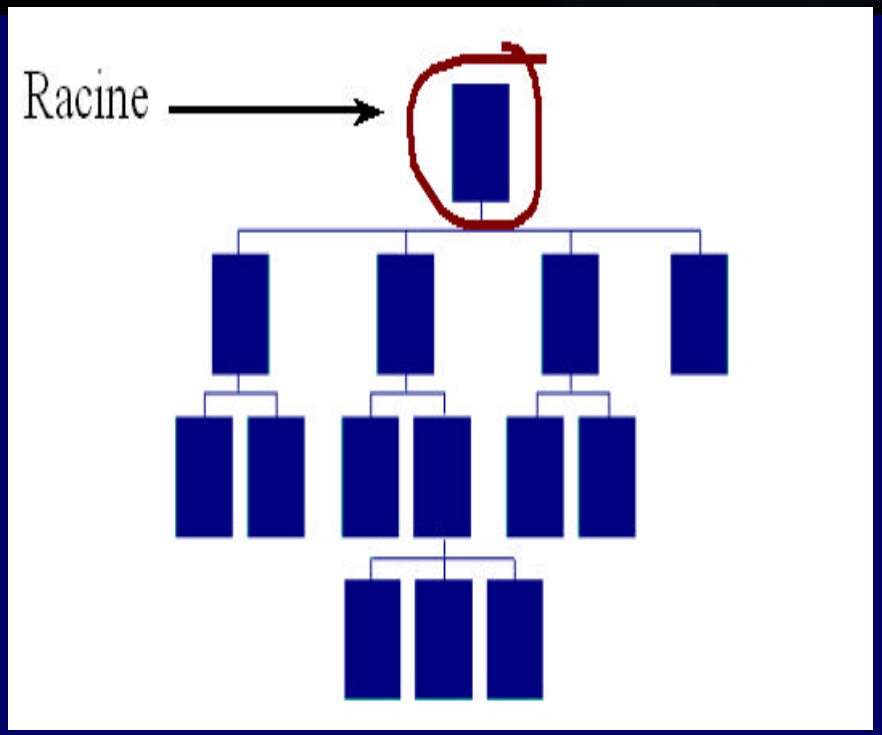
- Les éléments d'un document XML sont associés à un couple de balise :
  - une balise ouvrante commençant par un <
  - Une balise fermante commençant par </
- Éléments vides
  - Les éléments sans contenu ont une syntaxe spéciale, on peut les représenter par une balise « auto-fermée »

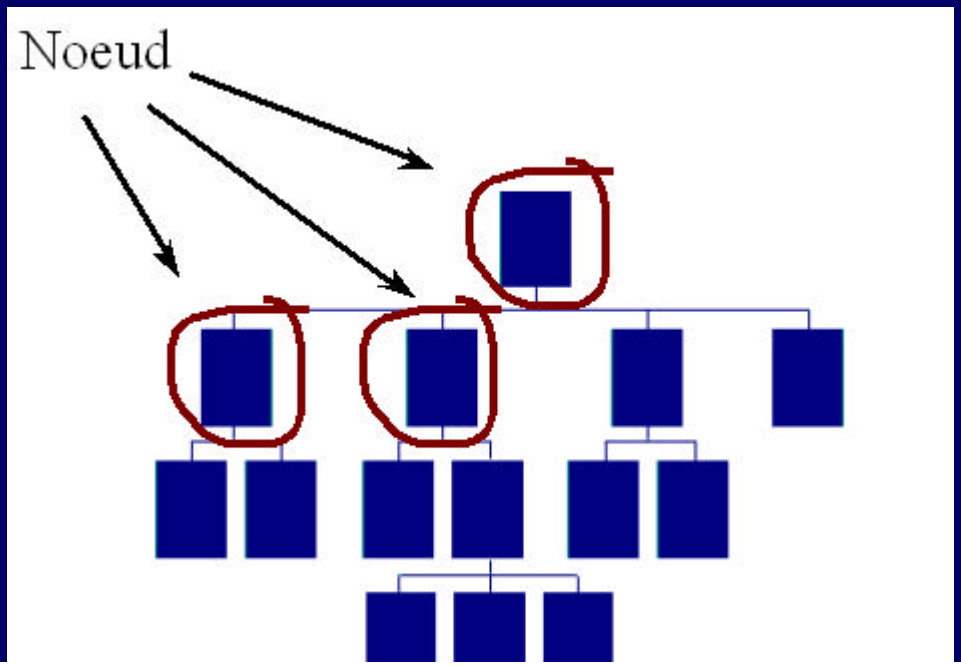
    <nomElement/>

Arbres XML

- Les documents XML sont des arbres
- Un seul élément sans parent : la racine
- Dès lors que les éléments ne se chevauchent pas et que tous les éléments, à l'exception de la racine, possèdent un parent, les documents XML représentent une structure de données appelée *arbre*


Arborescence


Racine


Noeud

Attributs

- Les éléments XML peuvent avoir des attributs

```
<person born='23/06/1912' died=
  '07/06/1954'>
Alan Turing
</person>
```

Les noms XML (1)

- Les noms d'éléments et autres noms XML peuvent contenir n'importe quel caractère alphanumérique
- Ils peuvent aussi contenir les signes :
  - souligné (_), trait d'union (-) et point (.)
- Ils ne doivent pas contenir d'autres signes de ponctuation
- Le caractère : est réservé pour une utilisation avec les espaces de noms

Les noms XML (2)

- Les noms XML doivent commencer par des lettres, des symboles ou le caractère souligné
  - Ils ne doivent pas commencer par un nombre, un trait d'union ou un point
- Il n'y a pas de limite dans la longueur d'un nom XML

Appels d'entité

- Un caractère utilisé dans un document ne doit pas contenir un chevron ouvrant non échappé
  - Pour utiliser ce caractère, il faut l'échapper en utilisant un appel d'entité : `<`
- XML définit 5 types d'appels d'entité
  - `<` => <
  - `&gt` => >
  - `&` => &
  - `&quot;` => "
  - `&apos;` => '

Section CDATA

- Pour inclure du code contenant des caractères tels que &, <, il est possible de définir une section CDATA (<![CDATA [...]])
- Tout ce qui est contenu à la place de ... est traité comme des données textuelles brutes (le parseur ne les analyse pas)

Commentaire

- Les documents XML peuvent contenir des commentaires
  - Ils commencent par <!--
  - Ils se terminent par -->
- Le double trait d'union ne doit pas apparaître dans le commentaire avant la séquence -->
- Les commentaires peuvent apparaître n'importe où sauf à l'intérieur d'une balise ou dans un autre commentaire

Instructions de traitement

- Les instructions de traitement sont des informations destinées aux applications qui "manipuleront" le document XML
  - Exemple : définition d'une feuille de styles
- Syntaxe
  - Début : <? Fin : ?>

```
<?xml-stylesheet href="person.css"
  type="text/css"?>
```

- Les instructions de traitement sont des balises mais pas des éléments

Déclarations XML

- Les documents XML peuvent commencer par une déclaration (syntaxe similaire à celle des instructions de traitement)
  - **Nom** : xml, attributs : version, standalone et encoding

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

  - **Version** : version des spécifications XML
  - **Encoding** : type de codage, ISO-8859-1 (Latin-1) prend en compte les caractères nécessaires pour les langues occidentales
  - **Standalone** : si la valeur est no, l'application peut avoir besoin de lire une DTD externe. Pour les documents sans DTD, on peut utiliser la valeur yes

XML et UNICODE

- XML utilise le jeu de caractères de la norme ISO 10646
  - les caractères alphabétiques, symboles, idéogrammes sont identifiés par un nombre codé sur 4 octets (32 bits) et sont décrits par une expression
- Lectures :
  - cf Poly
  - ***Michard, Alain. XML : langage et applications. Paris : Eyrolles, 2000. xv, 499 p. (ISBN: 2-212-09206-7) (En réserve à la bibliothèque)***
  - ***Marcoux, Yves. Les jeux de caractères [En ligne]. http://mapageweb.umontreal.ca/marcoux/enseign/6052/JeuxCar/JeuxCar.htm (Page consultée le 8 octobre 2001)***

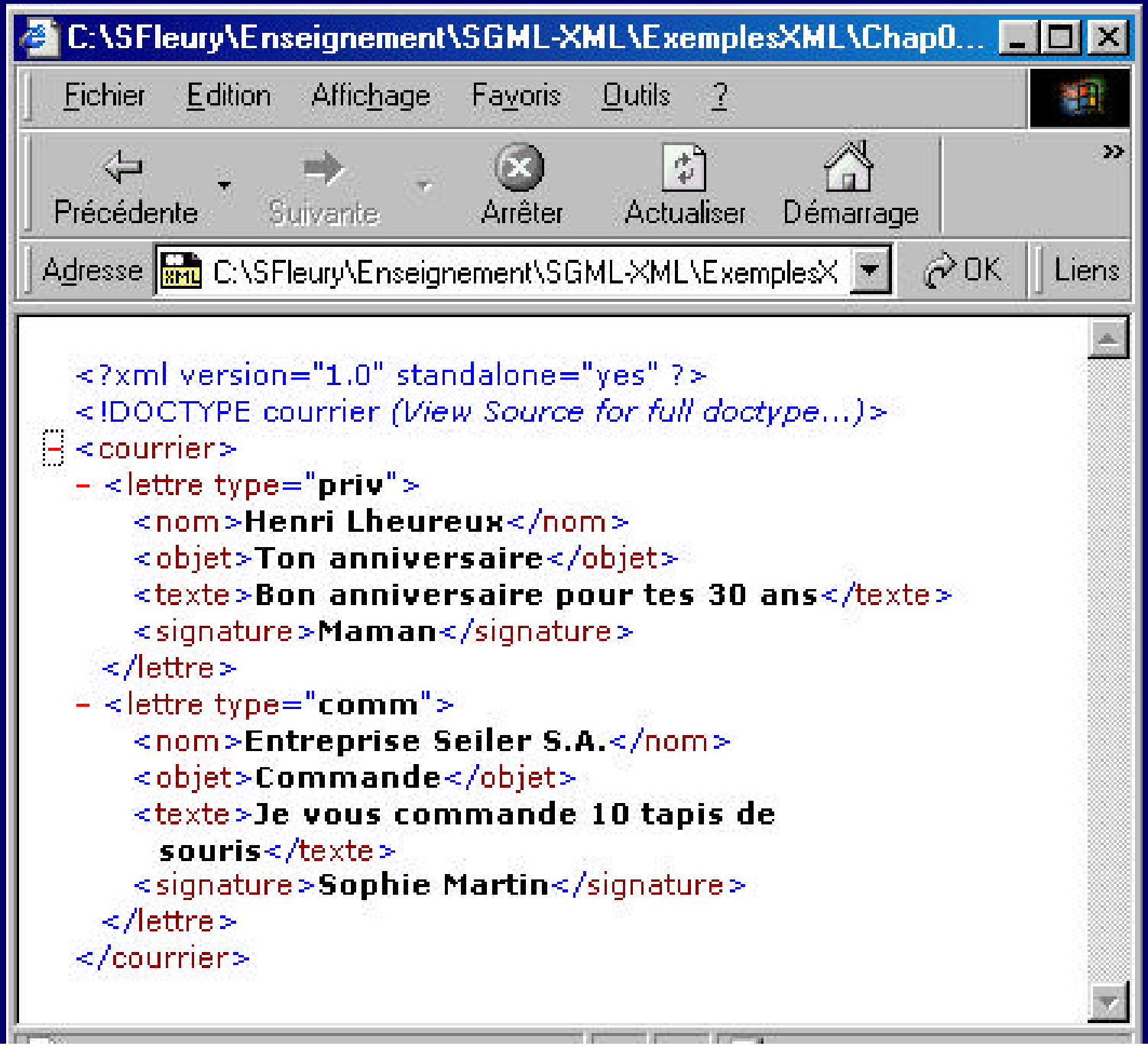
Exemple : une lettre [XML 00]

Demo5b - WordPad

Fichier Edition Affichage Insertion Format ?

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!DOCTYPE courrier
[
<!ELEMENT courrier (lettre)+>
<!ELEMENT lettre (nom?, objet, texte+, signature)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT objet (#PCDATA)>
<!ELEMENT texte (#PCDATA)>
<!ELEMENT signature (#PCDATA)>
]>
<courrier>
<lettre>
<nom>Henri Lheureux</nom>
<objet>Ton anniversaire</objet>
<texte>Bon anniversaire pour tes 30 ans</texte>
<signature>Maman</signature>
</lettre>

<lettre>
<nom>Entreprise Seiler S.A. </nom>
<objet>Commande</objet>
<texte>Je vous commande 10 tapis de souris</texte>
<signature>Sophie Martin</signature>
</lettre>

</courrier>
```

Pour de l'aide, appuyez sur F1


C:\SFleury\Enseignement\SGML-XML\ExemplesXML\Chap0...
Fichier Edition Affichage Favoris Outils ?
Précédente Suivante Arrêter Actualiser Démarrage
Adresse C:\SFleury\Enseignement\SGML-XML\ExemplesX OK Liens
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE courrier (View Source for full doctype...)>
<courrier>
<lettre type="priv">
<nom>Henri Lheureux</nom>
<objet>Ton anniversaire</objet>
<texte>Bon anniversaire pour tes 30 ans</texte>
<signature>Maman</signature>
</lettre>
<lettre type="comm">
<nom>Entreprise Seiler S.A.</nom>
<objet>Commande</objet>
<texte>Je vous commande 10 tapis de souris</texte>
<signature>Sophie Martin</signature>
</lettre>
</courrier>

XML en le pratiquant (3)

- TP XML partie 1
  - Cf Manipulation de documents XML

Partie 3

- Définition de type de Document

Retour Sommaire

Définition de type de document DTD

- Les DTDs permettent de définir quels éléments et quelles entités peuvent apparaître dans un document XML et quels sont les contenus des éléments et des attributs

Lecture Poly

- Leçon n°3 et 10 (pour les schémas)

Document valide

- Un document valide inclut une DTD à laquelle il se conforme

```
<? xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE person SYSTEM "person.dtd">
<person>
   <name><first>Alan</first><last>Turing</last>
   </name>
   <profession>informaticien</profession>
   <profession>mathématicien</profession>
</person>
```

DTD : premier exemple

```
<!ELEMENT person (name,
  profession*)>
<!ELEMENT name (first, last)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT profession
  (#PCDATA)>
```

- Contenue dans le fichier externe person.dtd

DTD interne

```
<?xml version="1.0" encoding="ISO-8859-1"?>
   <!DOCTYPE person [
   <!ELEMENT person (name, profession*)>
   <!ELEMENT name (first, last)>
   <!ELEMENT first (#PCDATA)>
   <!ELEMENT last (#PCDATA)>
   <!ELEMENT profession (#PCDATA)>]>
   <person>
   <name><first>Alan</first><last>Turing</last></name>
   <profession>informaticien</profession>
   <profession>mathématicien</profession>
   </person>
```

Contenu d'une DTD (1)

- Déclaration d'élément :

```
<!ELEMENT element1 (element2 , element3)>
```

  - element1 est composé de element2 suivi de element3

```
<!ELEMENT element1 (element2 | element3)>
```

  - element1 est composé de element2 ou de element3

```
<!ELEMENT element1 (element2)?>
```

  - element1 est composé de 0 ou 1 element2

```
<!ELEMENT element1 (element2)*>
```

  - element1 est composé de 0 ou plusieurs element2

Contenu d'une DTD (2)

`<!ELEMENT element1 (element2)+>`

- element1 est composé de 1 ou plusieurs element2

`<!ELEMENT element1 (#PCDATA)>`

- element1 est composé de caractères "parsable"

`<!ELEMENT element1 ANY>`

- element1 est composé de n'importe quel contenu

`<!ELEMENT element1 EMPTY>`

- element1 est toujours vide de contenu

Parenthèses

- Les choix, séquences et suffixes peuvent être combinés de manière complexe pour décrire n'importe quel modèle de contenu
  - Un choix ou une séquence peuvent être insérés à l'intérieur de parenthèses
  - Les parenthèses peuvent être suivies par ?, * ou +
  - Il est possible d'imbriquer des éléments entre parenthèses à l'intérieur d'autres parenthèses

Contenu mixte

- Pour définir un élément contenant à la fois du texte et des sous-éléments (contenu mixte), on utilise la syntaxe suivante :

```
<!ELEMENT contenumixte (#PCDATA |
  souselement)* >
```

  - Cette déclaration indique que l'élément contenumixte peut contenir du texte ou un sous élément nommé souselement sans spécifier l'ordre ni le nombre d'occurrence
    - #PCDATA doit toujours être en 1ère position dans un contenu mixte

Contenu d'une DTD (3)

- Déclaration d'attributs :
  - Pour déclarer un attribut associé à un élément on utilise la déclaration ATTLIST

```
<!ATTLIST image source CDATA #REQUIRED>
```

    - Signifie que l'élément source a un attribut image qui est une donnée textuelle et que l'élément image doit absolument fournir une valeur pour cet attribut

Type pour les attributs (1)

- CDATA
  - pour indiquer que la valeur de l'attribut est une séquence de caractères (y compris des entités de caractères)
- ID
  - pour indiquer que la valeur de l'attribut est un symbole commençant par une lettre et ne contenant que des lettres, des chiffres, ou les caractères - _ : et . (la valeur doit être unique sur l'ensemble d'un document)

Type pour les attributs (2)

- IDREF
  - pour indiquer que la valeur de l'attribut est un symbole défini comme valeur de l'attribut ID d'un autre élément de document
- IDREFS
  - comme IDREF, mais plusieurs valeurs séparées par des espaces sont autorisées

Type pour les attributs (3)

- Enumération
  - une énumération de valeurs possibles se fait en mettant entre parenthèses les valeurs séparées par des |. Exemple:
    - <!ATTLIST TD valign (top|middle|bottom|baseline) #IMPLIED ... >
- ENTITY, ENTITIES, NMTOKEN, NMTOKENS
  - Cf [XML 01]

Attribut par défaut

- #IMPLIED : attribut optionnel
- #REQUIRED : attribut obligatoire avec valeur définie dans chaque instance du document
- #FIXED : la valeur de l'attribut est fixe et non modifiable

Déclaration d'entités générales

- XML fournit 5 références d'entité (*cf* supra)
- Il est possible d'en définir de nouvelles
  - Déclaration ENTITY avec le nom de l'entité et son texte de substitution

```
<!ENTITY super "SUPERIOUS">
```

  - Utilisation via l'appel : `&super;`
  - Symboles pouvant être définis dans une DTD et utilisés dans un document XML comme raccourcis d'écriture. La définition complète du symbole est entièrement incluse dans la DTD.

Entités générales externes

- Un appel d'entité générale parsée est déclarée dans la DTD en utilisant la déclaration ENTITY

```
<!ENTITY footer SYSTEM
  "http://www.oreilly.com/boilerplate/footer.x
  ml">
```

- Quand le parser rencontre l'appel &footer; il doit le remplacer par le contenu du document donné à l'adresse visée

Entités et notations externes non parsées

- XML propose l'entité externe non-parsée comme mécanisme d'inclusion de documents non-XML (textes ASCII mal formés pour XML, images JPEG, GIF..., sons MIDI...)

```
<!ENTITY turing_getting_off_bus SYSTEM
  "http://www.turing.org.uk/turing/pi1/bus.jp
  g"
NDATA jpeg>
```

  - La déclaration NDATA indique le type de la donnée ; en fait ce nom indique plutôt une notation déclarée dans la DTD qui permet à XML de reconnaître une image définie par le Joint Photographic Experts Group

```
<!NOTATION jpeg SYSTEM "image/jpeg">
```

Entités paramètres

- Servent à définir des symboles qui seront utilisés ailleurs dans la DTD. Ce sont en quelque sorte des raccourcis d'écriture: partout où une entité est mentionnée, elle peut être remplacée par la chaîne de caractères qui lui est associée. Ce mécanisme s'apparente à un mécanisme de "macro"
- Note: les entités paramètres ne peuvent pas être utilisées en dehors d'une DTD

Entité paramètre : exemple

- Exemple tiré de la spécification du langage HTML :

```
<!ENTITY % heading "H1|H2|H3|H4|H5|H6">
```

- indique au système que toute occurence de `%heading;` doit être remplacée par `H1|H2|H3|H4|H5|H6`
- ce mécanisme peut aussi servir à utiliser un nom relativement compréhensible en lieu et place d'une séquence de caractères peu évocatrice

Entité (récursion)

- la définition d'une entité peut elle-même faire référence à d'autres entités. La substitution est alors effectuée récursivement. Par exemple, avec les définitions supplémentaires suivantes:
  - <!ENTITY % list "UL | OL">
  - <!ENTITY % preformatted "PRE">
  - <!ENTITY % block "P | %heading; | %list; | %preformatted; | DL | DIV | NOSCRIPT | BLOCKQUOTE | FORM | HR | TABLE | FIELDSET | ADDRESS">
    - l'entité %block; sera définie comme "P | H1 | H2 | H3 | H4 | H5 | H6 | UL | OL | PRE | DL | DIV | NOSCRIPT | BLOCKQUOTE | FORM | HR | TABLE | FIELDSET | ADDRESS"

Les entités de caractères

- Servent à donner un nom facilement lisible à des caractères qui ne sont peut-être pas représentables dans l'alphabet utilisé, ou qui ne sont pas disponibles au clavier.
- Exemples tiré de la spécification du langage HTML :

```
<!ENTITY nbsp " ">
<!ENTITY iexcl "&#161;">
```

Stylistique XML : règle 1

- Il est recommandé d ’inclure dans un document XML des « métadonnées » qui pourront être utilisées par diverses applications : ces informations peuvent décrire le document ou une partie

Stylistique XML : règle 2

- Il est recommandé de marquer toutes les constructions morpho-syntaxiques auxquelles devront être associées des règles de réalisation physique, typographique...

Stylistique XML : règle 3

- Il est inutile d ’introduire dans un document des indications précises de réalisation physique

Stylistique XML : règle 4

- Il est recommandé de marquer toutes les constructions morpho-syntaxiques qui ont une sémantique définie dans l ’univers du discours par un balisage spécifique
  - exemple : paragraphes, sections explicatives, notes supplémentaires, commentaires…

Stylistique XML : règle 5

- Le choix de faire figurer une information comme valeur d ’attribut ou comme données dans un élément devra être mûrement réfléchi (cf supra)

XML en le pratiquant (4)

- TP XML partie 2
  - Cf Manipulation de documents XML

SCHEMA XML

- Lecture Poly : leçon n°10
  - DTD : traces de SGML dans XML
  - Un schéma est une description via le langage XML de la structure d'un document XML
  - Application avec XMLSPY :
    - Compléments TP : Construire des schémas XML de vos documents de travail avec XMLSPY ou avec Microsoft XSD Inference 1.0 :
      - http://apps.gotdotnet.com/xmltools/xsdinference

Partie 4

- Les espaces de noms

Retour Sommaire

Espace de noms [cf XML-01]

- Objectifs
  - Distinguer les éléments et les attributs de différentes applications XML qui ont le même nom
  - Grouper tous les éléments et attributs relatifs à une même application XML pour que les logiciels puissent les reconnaître facilement

Lecture Poly

- Leçon 10 (le début)

- Un document XML peut utiliser des balises définies dans différentes DTDs (ou, plus récemment, dans différents schémas). Pour utiliser des symboles définis ailleurs, un document XML doit "importer" l'espace de noms correspondant. Cela se fait en ajoutant un attribut approprié à l'élément qui va utiliser les symboles de l'espace de noms importé:
  - <uneBaliseDuDocumentCourant xmlns:Préfixe="UrlDeDtdOuSchemaAImporter">

Espace de noms HTML

- Pour utiliser les balises HTML dans un document XML:

```
<?xml version="1.0" encoding="ISO-
 8859-1"?>
<article
 xmlns:html="http://www.w3.org/Profi
 les/XHTML-transitional">
```

  - Les éléments utilisant des balises importées doivent alors être de la forme <Préfixe:NomDeBalise ...>. Par exemple:

```
<html:img html:src="..." />
```

- La syntaxe doit rester conforme aux contraintes imposées par XML
  - d'où le "/>" pour fermer l'élément vide contenant l'image.
- N'importe quel élément d'un document XML peut importer un espace de nom.
  - L'espace de noms sera alors accessible au sein de l'élément en question, y compris dans tous les éléments imbriqués.

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

<catalog>

<RDF>

<Description about= 'tableaux.xml'>

<title>Tableaux</title><creator> Someone </creator>

<description>Liste de tableaux</description><date>22-08-
  2000</date>

</Description>

</RDF>

<painting>

  <title>Les demoiselles
  d'Avignon</title><artist>Picasso</Artist>

  <date>1928???</date><description>cf Pierre Daix</description>

</painting>

...

</catalog>
```

Nécessité des espaces de noms

- Dans le prologue, utilisation de RDF (Ressource Description Framework) et de Dublin Core pour inclure des métadonnées
  - Éléments `Description` et `RDF` pour RDF
  - Éléments `Title, creator, description` et `date` pour Dublin Core
- L'élément `title` est utilisé pour le titre de la page et pour le titre des tableaux
- Idem pour l'élément `date`

Limiter la portée des éléments en leur associant des espaces

- Les espaces de noms lèvent le problème d'ambiguïté des éléments en assignant les éléments et attributs à des URI
- On associe en fait des préfixes aux URI et ce sont ces préfixes que l'on adjoint aux éléments :
  - rdf:description (rdf = préfixe, description = partie locale, le tout devient un nom qualifié)
  - xsl:template

Association de préfixes aux URI

- Chaque préfixe dans un nom qualifié doit être associé à une URI
- Pour XSLT
  - Le préfixe xsl est associé à l'URI http://www.w3.org/1999/XSL/Transform
- Pour RDF

```
<rdf:RDF xmlns:rdf=
  'http://www.w3.org/TR/REC-rdf-
  syntax#'>
```

  - L'attribut xmlns:rdf de l'élément rdf:RDF lie le préfixe rdf à l'URI de l'espace de nom donné

- Les associations ont la portée de l'élément qui les définit
- Dans l'exemple précédent, xmlns:rdf déclare le préfixe rdf pour l'élément rdf:RDF et ses sous-éléments

```
<?xml version='1.0' encoding='ISO-8859-1'
standalone='yes'?>
<catalog>
<rdf:RDF xmlns:rdf='http://www.w3.org/TR/REC-rdf-
syntax#'>
<rdf:Description xmlns:dc='http://purl.org/dc/' about=
'tableaux.xml'>
<dc:title>Tableaux</dc:title>
<dc:creator> Someone </dc:creator>
<dc:description>Liste de tableaux</dc:description>
<dc:date>22-08-2000</dc:date>
</rdf:Description>
</rdf:RDF>
<painting>
  <title>Les demoiselles d'Avignon</title>
  <artist>Picasso</Artist>
  <date>1928???</date>
  <description>cf Pierre Daix</description>
</painting>
...
</catalog>
```

Partie 5

- XML comme format de document

Retour Sommaire

TEI

- Text Encoding Initiative :
  - http://www.tei-c.org
- Application SGML conçue pour le balisage de documents du domaine des sciences humaines
- Migration de TEI vers XML
- Cf [XML-01] : exemple de document TEI utilisant la version XML de la DTD TEI

DocBook

- http://www.oasis-open.org/docbook/
- Application SGML pour les nouveaux documents (documentation informatique...)
- Version actuelle 4.1.2 disponible en SGML et en XML
- Cf [XML-01] : exemple de document docBook

Partie 6

- XML sur le WEB

Retour Sommaire

XML sur le web

- XHTML : variante « xmlisée » de HTML 4.0 qui restreint HTML à la syntaxe XML
- Utilisation d'outils de formatage de documents XML pour les rendre lisibles dans les navigateurs (feuille de style CSS ou XSLT par exemple)

RDF

- RDF est un encodage XML de données
- Un document RDF décrit des ressources
- Exemple de document RDF
- Lecture : principes de RDF et de RDFS
  - http://www.w3.org/TR/REC-rdf-syntax/
  - http://perso.enst.fr/~ta/web/rdf/Principe-RDF.html

Dublin Core

- Dublin Core est un schéma de métadonnées générique défini par le *Dublin Core Metadata Initiative* et il est utilisé pour décrire de façon simple des ressources électroniques
- Le Dublin Core est un ensemble standard de 15 informations avec une sémantique précise reflétant le type de données que l'on retrouve dans les catalogues ou les bibliogaphies (cf [XML-01])
- Lecture :
  - Christophe Jacquet, 2003, « Dublin Core et MétaDonnées » en ligne et dans la textothèque sur le CDTAL:
  - http://openweb.eu.org/articles/dublin_core/index.php

Dublin Core (2)

En bref

Les métadonnées du Dublin Core permettent de donner des informations à propos de pages Web. Dans cet article, nous voyons la forme de ces descriptions, ainsi que la façon de les intégrer dans des pages Web.

Introduction

Depuis toujours, les bibliothécaires ont rédigé des notices, ou des index, pour décrire les documents disponibles. Pour reprendre le vocabulaire utilisé en informatique, ces notices constituent des données, qui servent à décrire d'autres données (le contenu des livres) : on parle alors de **métadonnées**.

Ces *données à propos de données* s'appliquent particulièrement bien aux pages Web. On peut ainsi vouloir coder de façon claire certains renseignements essentiels des pages : le titre, l'auteur, la date, les mots-clefs, etc.

Avec ces métadonnées, le travail des moteurs d'indexation et de recherche est grandement facilité car ils peuvent extraire automatiquement de nombreuses informations sur le document, ainsi que reconstituer les relations qui existent entre documents (par exemple, on peut savoir qu'une page est la cinquième d'un ensemble cohérent de quinze pages chaînées les unes aux autres).

Cependant, il est nécessaire que tous les acteurs (publicateurs, éditeurs de logiciels clients, etc.) partagent une définition et une nomenclature communes des métadonnées. Par exemple, aucun échange ne serait possible si la liste des mots-clefs était parfois nommée « keywords », d'autres fois

Dublin Core (3)

Principes

Le Dublin Core définit un certain nombre de champs de métadonnées utilisables pour les pages Web. Mais il ne décrit en aucun cas la façon de représenter ces métadonnées en pratique. Ainsi, il existe plusieurs représentations utilisées, et d'autres sont envisageables. Dans la suite, nous utiliserons la présentation qui intéresse plus particulièrement les concepteurs de sites Web : l'inclusion des métadonnées dans les pages HTML.

Les champs de métadonnées du Dublin Core sont appelés **éléments** ; la signification de certains d'entre peut être précisée à l'aide de **raffinements**. Un *raffinement* restreint la signification d'un élément, mais sans la changer fondamentalement. L'utilisation des *raffinements* est facultative.

Dans une description utilisant le Dublin Core, des valeurs sont donc associées aux champs de métadonnées. Ces valeurs peuvent être données dans un format libre (chaînes de caractères interprétables par des humains, mais sans signification particulières pour les machines), mais elles peuvent aussi se conformer à des formats de données bien définis. Le Dublin Core référence un certain nombre de formats de données *officiels*.

Champs de métadonnées

Nous allons tout d'abord voir la liste exhaustive des champs de métadonnées du Dublin Core (**éléments** et **raffinements**), puis nous apprendrons comment les utiliser en pratique avec HTML. Dans cette liste, nous décrivons chaque élément, et nous donnons par élément la liste des *raffinements* possibles.

Dublin Core (4)

Élément	Description et liste des *raffinements*
title	**Titre du document** : il s'agit a priori du titre principal du document. Pour indiquer un autre type, on peut utiliser le *raffinement* suivant : • **alternative** : alternative pour le titre, par exemple une abréviation ou une traduction.
creator	**Créateur du document** : nom de la personne, de l'organisation ou du service à l'origine de la rédaction du document.
subject	**Sujet et mots-clefs** : mots-clefs, phrases de résumé, ou codes de classement. Il est préférable d'utiliser des mots-clefs choisis dans le cadre d'une politique de classement. Par exemple, il est recommandé d'utiliser les codages de la bibliothèque du congrès (LCSH et LCC), le vocabulaire médical (MESH), ou les notations décimales des bibliothécaires (DDC et UDC).
description	**Description du document** : résumé, table des matières, ou texte libre. Le type de description peut être précisé à l'aide des *raffinements* suivants : • **tableOfContents** : table des matières ; • **abstract** : résumé.
publisher	**Publicateur du document** : nom de la personne, de l'organisation ou du service à l'origine de la publication du document.
contributor	**Contributeur au document** : nom d'une personne, d'une organisation ou d'un service qui contribue ou a contribué à l'élaboration du document.

Dublin Core (5)

date	**Date d'un évènement dans le cycle de vie du document** : il peut s'agir par exemple de la date de création ou de la date de mise à disposition. Il est recommandé de spécifier la date au format W3CDTF (AAAA-MM-JJ). Pour préciser de quelle date il s'agit, on utilise les *raffinements* suivants : • **created** : date de création ; • **valid** : date ou période de validité ; • **available** : date ou période de mise à disposition ; • **issued** : date de publication ; • **modified** : date de modification ; • **dateAccepted** : date d'acceptation (par exemple, acceptation d'une thèse par une université, d'un article par un journal, etc.) ; • **dateCopyrighted** : date du copyright ; • **dateSubmitted** : date où le document a été soumis (par exemple, soumis à un comité de lecture s'il s'agit d'un article).
type	**Nature ou genre du contenu** : grandes catégories de document. Il est recommandé d'utiliser des termes clairement définis au sein de son organisation. Par exemple, le Dublin Core définit quelques types dans le vocabulaire DCMITypes.

Dublin Core (6)

format	**Format du document** : format physique ou électronique du document. Par exemple, type de média ou dimensions (taille, durée). On peut spécifier le matériel et le logiciel nécessaires pour accéder au document. Il est recommandé d'utiliser des termes clairement définis, par exemple les types MIME ou IMT . Les *raffinements* suivants sont disponibles : • **extent** : taille ou durée ; • **medium** : support physique.
identifier	**Identificateur non ambigu** : il est recommandé d'utiliser un système de référencement précis, par exemple les URI ou les numéros ISBN.
source	**Ressource dont dérive le document** : le document peut découler en totalité ou en partie de la ressource en question. Il est recommandé d'utiliser une dénomination formelle des ressources, par exemple leur URI.
language	**Langue du document** : il est recommandé d'utiliser un code de langue conforme au format RFC3066.

relation	**Lien vers une ressource liée** : il est recommandé d'utiliser une dénomination formelle des ressources, par exemple leur URI. On précise le type de lien avec des *raffinements* : • **isVersionOf** : on a affaire à une nouvelle version, une modification ou une adaptation du document lié. Les changements concernent le contenu et pas seulement la forme ; • **hasVersion** : réciproque d'`isVersionOf`. Le document lié est une version modifiée du présent document ; • **isReplacedBy** : le présent document a été remplacé par le document lié ; • **replaces** : réciproque de `replaces`. Le présent document remplace le document lié ; • **isRequiredBy** : on a besoin du présent document pour interpréter correctement le document lié ; • **requires** : réciproque d'`isRequiredBy`. Le présent document a besoin du document lié pour être correctement présenté, transmis, ou pour assurer sa cohérence ; • **isPartOf** : le document est une partie (physique ou logique) d'un autre document ; • **hasPart** : réciproque d'`isPartOf` : le document inclut le document lié, physiquement ou logiquement ; • **isReferencedBy** : le document courant est référencé, cité, ou lié par le document indiqué ; • **references** : réciproque d'`isReferencedBy` : le document courant référence, cite ou pointe vers le document indiqué ; • **isFormatOf** : le présent document a le même fond que le document indiqué, mais présenté sous une forme différente ; • **hasFormat** : réciproque d'`isFormatOf` : le présent document possède une variante

Dublin Core (8)

	• **conformsTo** : référence à un standard établi auquel se conforme le présent document.
coverage	**Portée du document** : la portée inclut un domaine géographique, un laps de temps, ou une juridiction (nom d'une entité administrative). Il est recommandé d'utiliser des représentations normalisées de ces types de données. Le type de couverture peut être précisé : • **spatial** : couverture spatiale. On peut utiliser les codages Point (point géographique), ISO3166 (codes de pays à deux lettres), Box (régions géographiques), ou TGN (dictionnaire de noms de lieux) ; • **temporal** : couverture temporelle. On peut utiliser les codages Period (intervalle de temps) ou W3CDTF (dates).
rights	**Droits relatifs à la ressource** : permet de donner des informations sur le statut des droits du document, par exemple la présence d'un copyright, ou un lien vers le détenteur des droits. L'absence de cet élément ne présume pas que le document est libre de droits.
audience	**Audience du document** : l'audience représente le groupe de personnes à qui le document est destiné. L'audience est déterminée par l'auteur, le publicateur, ou un tiers. On peut utiliser les *raffinements* suivants : • **mediator** : entité qui sert d'intermédiaire pour l'accès au document ; • **educationLevel** : position du niveau de l'audience par rapport à un contexte d'éducation ou de formation.

Dublin Core (9)

Formats de données

Au cours de la description, nous avons présenté de façon informelle les formats de données utilisables dans les descriptions.

Dans le tableau suivant, nous présentons en résumé la liste des formats de données « officiels » du Dublin Core, ainsi que leurs noms formels.

Nom formel	Description	Éléments concernés
LCSH	En-tête de la bibliothèque du Congrès (Library of Congress Subject Heading).	subject
MESH	Vocabulaire médical MeSH (Medical Subject Headings).	subject
DDC	Notation décimale de bibliothèque Dewey Decimal Classification.	subject
LCC	Classification de la bibliothèque du Congrès (Library of Congress Classification).	subject
UDC	Notation décimale de bibliothèque Universal Decimal Classification.	subject
DCMIType	Type du document parmi une série de types définis par le Dublin Core : Collection (agrégation de documents formant un tout), Dataset (structure appropriée au traitement par ordinateur), Event (évènement, par définition éphémère. Par exemple : exposition, conférence, bataille, procès, mariage, etc.), Image (photographie, peinture, dessin, animation, film, diagramme, carte, notation musicale, etc.), InteractiveResource (objet qui demande une interaction de l'utilisateur, par exemple une page Web, une applet, un	type

Dublin Core (10)

	service de chat, etc.), `Service` (entité qui rend des services à l'utilisateur, comme un service de photocopie ou un serveur Web), `Software` (logiciel d'ordinateur), `Sound` (son, par exemple voix ou musique), `Text` (texte au sens large : livre, magazine, journal, poème, message d'une liste de diffusion, etc.), `PhysicalObject` (tout objet physique : ordinateur, sculpture, bâtiment, etc.)	
`IMT`	Internet Media Types, nomenclature de types de données très proche MIME (exemple : `image/jpeg`)	`format`
`ISO639-2`	Code de langue à trois lettres ISO-639-2 (du type `fre`).	`language`
`RFC1766`	Code de langue RFC 1766. Il est composé d'un code de langue sur deux lettres, éventuellement suivi d'un suffixe de pays pour indiquer la variante de la langue (par exemple, `fr-CA` pour la Français du Canada, `fr-FR` pour le Français de France, `en` pour l'Anglais en général).	`language`
`RFC3066`	Code de langue RFC 3066, nouvelle version de `RFC1766` (le code de langue peut être sur trois lettres).	`language`
`URI`	Un URI, selon la RFC 2396.	`identifier`, `source`, `relation`
`Point`	Point de l'espace repéré par ses coordonnées géographiques.	`coverage.spatial`
`ISO3166`	Codes ISO-3166 de représentation des noms de pays sous forme de chaîne de deux caractères (exemples : `FR`, `DE`).	`coverage.spatial`
`Box`	Représentation des régions géographiques du Dublin Core.	`coverage.spatial`
`TGN`	Noms issus du Getty Thesaurus of Geographic Names.	`coverage.spatial`
`Period`	Représentation des intervalles de temps du Dublin Core.	`coverage.temporal`, `date`
`W3CDTF`	Codage des dates et heures du W3C. Pour référencer un jour, on utilise le format AAAA-MM-JJ.	`coverage.temporal`, `date`

Dublin Core (11)

En pratique : utilisation avec (X)HTML

Il est possibile d'inclure les métadonnées relatives à un document HTML directement à l'intérieur de ce document. Elles prennent place dans des balises `<meta>`, à l'intérieur de l'en-tête (section `<head>`).

Si vous n'utilisez pas de *raffinement*, la syntaxe à utiliser est la suivante :

```
<meta name="DC.Nom d'élément" content="valeur de la métadonnée"  />
```

On écrit le nom d'élément avec une majuscule initiale.

Lorsqu'on utilise un *raffinement*, on utilise très logiquement la syntaxe suivante :

```
<meta name="DC.Nom d'élément.nom de raffinement" content="valeur de la métadonnée"  />
```

Les noms de *raffinements* doivent toujours commencer par une lettre minuscule.

La barre oblique à la fin de la balise `<meta>` est nécessaire en XHTML.

Les balises `meta` peuvent éventuellement être dotées des attributs supplémentaires suivants :

`lang`

Permet de spécifier la langue du contenu.

`scheme`

Permet de spécifier le format normalisé utilisé pour le contenu. Cet attribut prend ses valeurs dans la première colonne du tableau « formats de données ».

Voici en pratique un extrait de document d'exemple dans lequel on a inséré des métadonnées. Cet extrait de code contient une déclaration de type de document pour le XHTML 1.0 Strict. En pratique, remplacez-la par la déclaration de votre choix.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>
  <head>
    <title>Un document en HTML</title>
    <meta http-equiv="Content-type"
      content="text/html; charset=iso-8859-1"  />
    <link rel="schema.DC"
      href="http://purl.org/dc/elements/1.1/"  />
    <meta name="DC.Title" lang="fr" content="Un document en HTML"  />
    <meta name="DC.Date.created" scheme="W3CDTF" content="2003-04-03"  />
    <meta name="DC.Date.modified" scheme="W3CDTF" content="2003-04-27"  />
    <meta name="DC.Subject" lang="fr" content="HTML, document, Dublin Core"  />
    <meta name="DC.Language" scheme="RFC3066" content="fr-FR"  />
    <meta name="DC.Description" lang="fr"
      content="Mon premier document HTML avec métadonnées"  />
  </head>
  <body>
    ...
  </body>
</html>
```

La balise `<link ...>` référence la liste officielle des éléments et des *raffinements* du Dublin Core. Elle permet aux logiciel de savoir à quoi exactement correspond le préfixe DC, en spécifiant son **schéma**.

Partie 7

- XSL – XSLT

Retour Sommaire

Lecture Poly

- Leçons 7,8,9

Préambule CSS/XSLT

- Les exemples qui suivent seront à tester tels qu'ils sont présentés

Feuilles de styles avec XML

- Cascading Style Sheet (CSS)eXtensible Stylesheet Language (XSL)
- On utilisera [XML 1999] ou [XML 01] pour une présentation complète des notions associées à des langages de descriptions des feuilles de styles avec XML
  - La présentation faite infra reprend celle faite sur le site :

    http://www.citeweb.net/apetitje/xml

Cascading Style Sheet

- Les **C**ascading **S**tyle **S**heets ou Feuilles de Styles en Cascade ont été initialement conçues pour le langage HTML.
- La première version du standard, dite **CSS niveau 1** (CSS1), a été publiée en 1996

CSS1/2

- Mai 1998, le W3 Consortium a publié une nouvelle version, dite **CSS niveau 2** (CSS2).
  - Introduction de la notion de *type de média* qui permet de spécifier la mise en page d'un document en vue de son affichage sur écran, de son impression sur papier, ou d'indiquer les conditions de sa réalisation par synthèse vocale ou sur des terminaux braille.
  - Les spécifications complètes : http://www.w3c.org/TR/CSS

Création d ’une feuille CSS

- Plusieurs manières de procéder :
  - en incluant en-tête de page HTML les styles que l'on souhaite employer dans la page en question
  - en écrivant une feuille de style dans un fichier séparé
- Cette seconde solution est celle qui a été retenue pour l'utilisation avec XML

Exemple

- Cela s'écrit de la façon suivante :

```
<?xml version='1.0'?>
<?xml-stylesheet
  href="FeuilleDeStyle.css"
  type="text/css" ?>
```

Concepts de base

- Il est facile de concevoir des feuilles de style simples.
- Exemple pour définir la couleur bleue à l'élément (ou balise) **H1**, on écrira :

```
H1 { color: blue }
```

La règle de style

- Une règle de style se compose
  - d'un *sélecteur* qui indique l'élément auquel elle s'applique, et d'une ou de plusieurs *propriétés* ainsi que leur *valeur* respective.
  - Le couple *propriété: valeur* forme ce que l'on appelle la *déclaration.*

```
sélecteur {propriété1:
 valeur ; propriété2:
 valeur }
```

La règle de style (2)

- On peut regrouper les sélecteurs en les séparant par une virgule.

```
H1, H2, H3 { font-family: arial }
```

- On peut regrouper les déclarations.

```
H1 {
  font-weight: bold;
  font-size: 12pt;
  line-height: 14pt;
  font-family: arial;
  font-variant: normal;
  font-style: normal;
  }
```

Héritage

- Dans le premier exemple, nous avons spécifié la couleur bleue à l'élément **H1**. Supposons maintenant que nous avons un élément en italique à l'intérieur de celle-ci.
- `<H1>Le titre d'un document <EM>est</EM> très important !</H1>`

Héritage (2)

- Si aucune couleur n'a été spécifiée pour l'élément **EM**, le mot en italique "est" héritera automatiquement de la couleur de son élément parent (H1), soit bleue pour notre exemple.
- Il en va également de même pour toutes les autres propriétés à savoir font-family, font-size, etc...

Style par défaut

- Pour définir un style "par défaut" pour tout un document il suffit de le définir sur l'élément parent à tous les autres - *l'élément racine* - i.e en HTML, l'élément <BODY>.

```
BODY {
  color : black;
  background: white;
  font-style: arial;
  }
```

Sélecteur de classe

- Afin d'accroître le contrôle de granularité sur les éléments, un nouvel attribut a été ajouté à HTML : 'CLASS'.
- Tous les éléments contenus dans l'élément racine peuvent être regroupés par classe. Ces classes seront définies dans la feuille de style.

```
<HTML>
<HEAD>
<TITLE>Titre</TITLE>
<STYLE TYPE="text/css">
H1.citation { font-style: italic;
font-weight: bold }
</STYLE>
</HEAD>
<BODY>
<H1 CLASS=citation>La vie est trop
courte pour se faire la
guerre</H1>
</BODY>
</HTML>
```

- Cas 1 : dans l ’exemple précédent, l'élément **H1** est « surchargé » dans le document HTML.
- Cas 2 : on aurait pu aussi écrire une feuille de style de la façon suivante :

```
citation { font-style: italic;
  font-weight: bold }
```

- Il existe une méthode plus élégante pour utiliser les classes sans avoir surcharger des éléments prédéfinis. Il s'agit de l'élément **DIV** qui a été réservé à cet effet.

```
<DIV CLASS=aspect1>
  Ce texte apparaîtra avec les
  propriétés définies
  dans la classe aspect1
  </DIV>
  ...
```

Commentaires

- Les commentaires utilisés à l'intérieur des feuilles de style sont similaires à ceux que l'on utilise en langage C

```
EM { color: red } /* rouge,
   vraiment rouge !! */
```

```
<STYLE TYPE="text/css">

body {font-family: arial, helvetica;
    font-size:9pt}

td {font-family: arial, helvetica;
    font-size:9pt}

H1 {font-family: arial, helvetica;
    font-size:12pt;
    font-style:italic}

a {font-family: arial, helvetica;
    font-size:9pt;
    font-weight:bold;
    color:#b80000}

a:hover {font-family: arial, helvetica;
    color:#FF0000}

a.main {font-family: arial, helvetica;
    font-size:9pt;
    font-weight:bold;
    color:#b80000}

a.sub {font-family: arial, helvetica;
    font-size:9pt;
    color:#b80000}
    p {font-size:12pt;
    text-align: justify}

UL {text-align: justify}

LI {list-style-image: url(./images/pucerouge.gif) }
    CODE {font-size:11pt}

</STYLE>
```

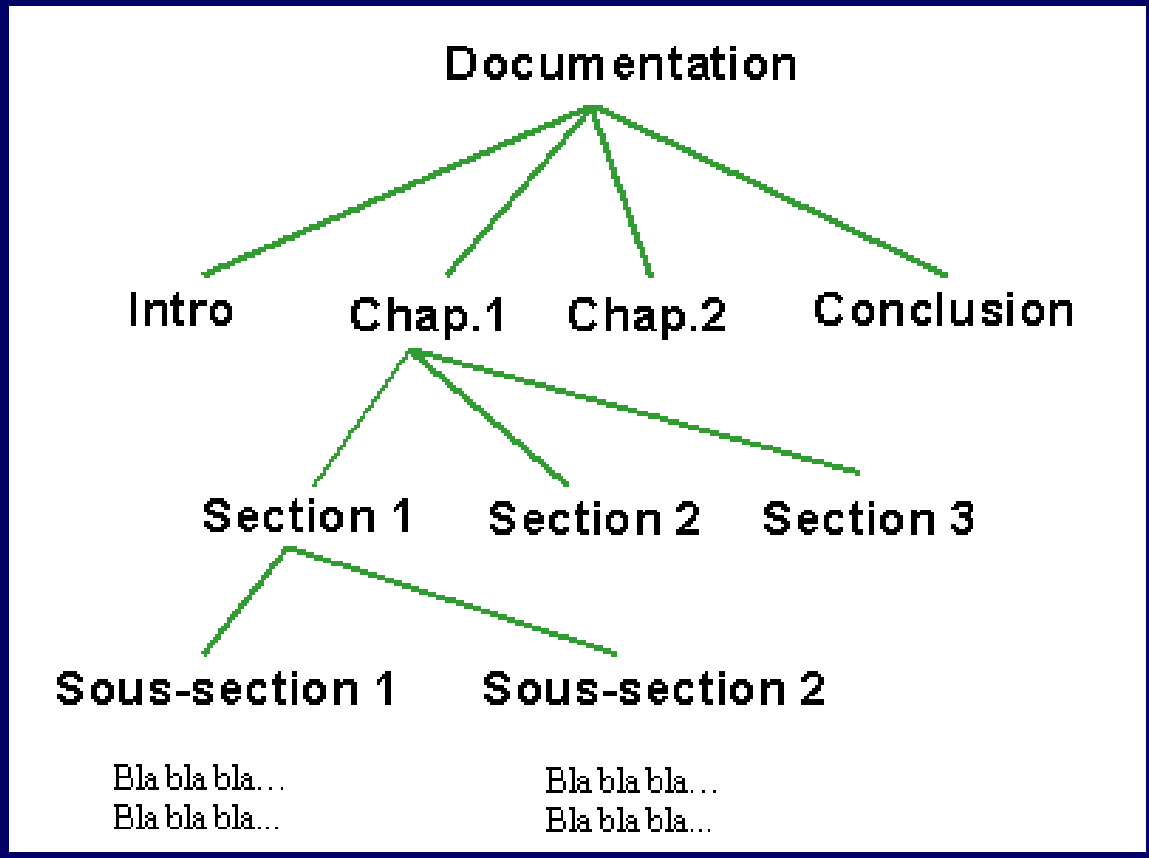
Exemple de feuille de style utilisée pour toutes les pages du site :

http://www.citeweb.net/apetitje/xml

eXtensible Stylesheet Language (XSL)

- **Généralités**
  - Un document XSL est tout simplement un document XML qui contient tout un ensemble de règles
  - Pour définir une règle de style il faut :
    - 1. décrire le squelette ou arbre du document
    - 2. décrire le style que l'on veut appliquer à chaque articulation du squelette ou à chaque de noeud de l'arbre

- un document XML est une structure arborescente
  - Parcours de l'arbre avec Xpath



Structure d’une feuille XSLT

Prologue (exemple):

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Corps (suite de gabarits ou *templates*):

```
<xsl:template match="exemple">
  ...
</xsl:template>
...
```

Épilogue:

```
</xsl:stylesheet>
```

Les règles

- `template = rule = règle = gabarit`
  - décrit la structure de chaque noeud de l'arbre ainsi que le style à appliquer à ce noeud.

```
<xsl: template match="accueil">
  ...
</xsl:template>
```

Gabarit (noeud accueil)

Style à appliquer sur le noeud

- Une règle se décompose en deux parties :
  - on associe un motif (pattern) au patron (template) afin d'identifier chaque noeud de l'arbre
  - on associe une action, une mise en forme, un formatage ou une transformation au noeud résultant
- Un motif sélectionne un noeud en utilisant un ou plusieurs des critères suivants :
  - un nom
  - l'ancêtre de l'élément
  - un identifiant unique (ID)
  - des jokers
  - un attribut de l'élément source
  - sa position relative par rapport à ses éléments voisins ou frères

Exemple (1)

```
<xsl: template match="accueil">

  <fo:block
  font-family="times new roman,
  serif"
  font-size="12pt"
  text-color="red"
  </fo:block>

  </xsl:template>
```

Exemple (1) (suite)

- le document XML :

```
<accueil>Bonjour monde !</accueil>
```

- le fichier XSL

```
<xsl:stylesheet>
  <xsl:template match="/">
  <b><i><xsl:value-of
  select="accueil"/></i></b>
  </xsl:template>
  </xsl:stylesheet>
```

- le résultat :

Bonjour monde !

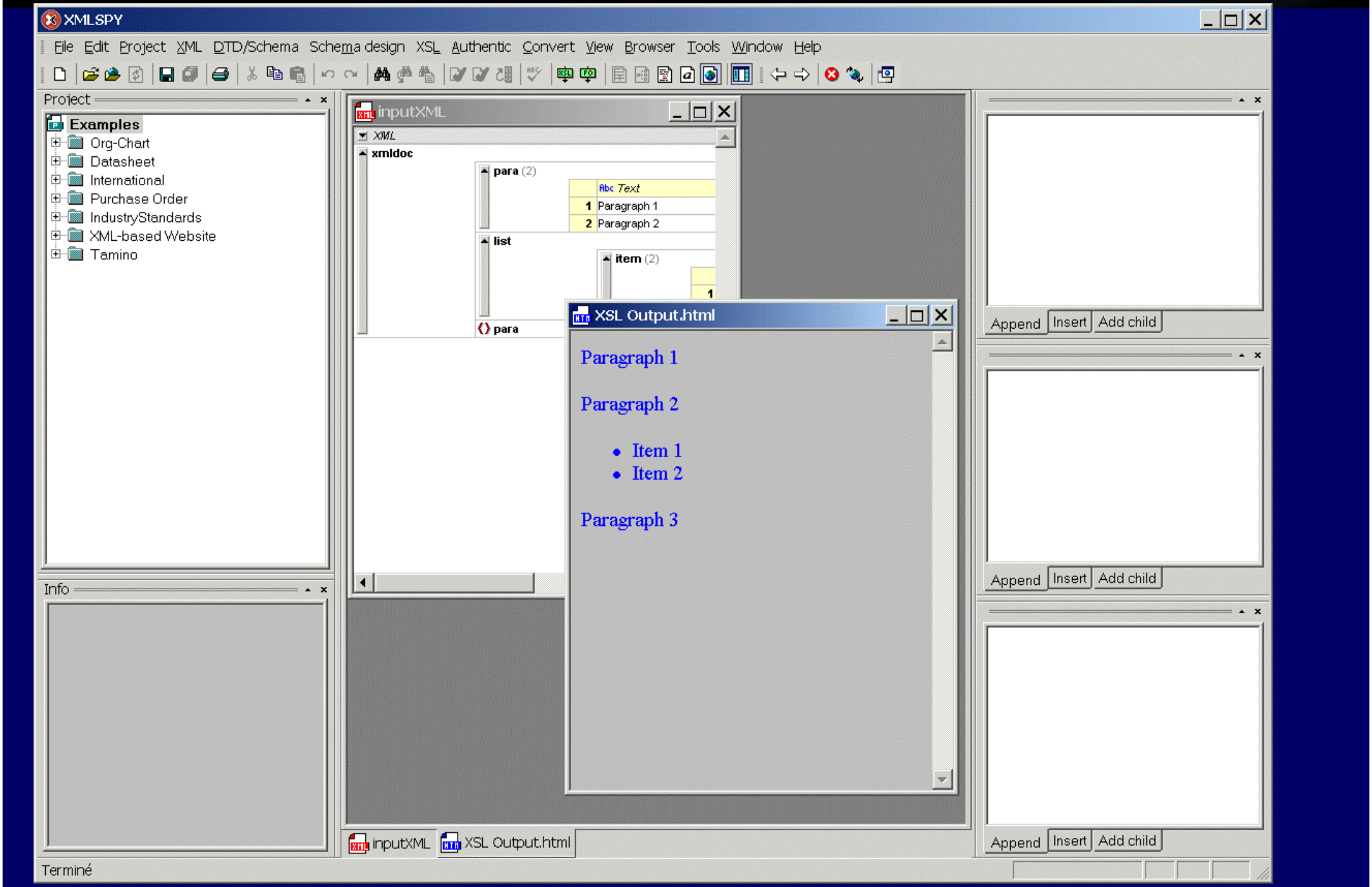
Exemple (2)

- Document XML

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xmldoc>
     <para>Paragraph 1</para>
     <para>Paragraph 2</para>
     <list>
          <item>Item 1</item>
          <item>Item 2</item>
      </list>
      <para>Paragraph 3</para>
</xmldoc>
```

? Document XSL

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

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

<xsl:output method="html" />

<xsl:template match="/">

  <html><body bgcolor="silver"><font color="blue"><xsl:apply-templates /></font></body></html>

</xsl:template>

<xsl:template match="xmldoc">

  <xsl:apply-templates />

</xsl:template>

<xsl:template match="para">

  <p><xsl:apply-templates /></p>

</xsl:template>

<xsl:template match="list">

  <ul><xsl:apply-templates /></ul>

</xsl:template>

<xsl:template match="item">

  <li><xsl:apply-templates /></li>

</xsl:template>
```


XMLSPY
File Edit Project XML DTD/Schema Schema design XSL Authentic Convert View Browser Tools Window Help
Project
Examples
Org-Chart
Datasheet
International
Purchase Order
IndustryStandards
XML-based Website
Tamino
Info
inputXML
XML
xmldoc
para (2)
Abc Text
1 Paragraph 1
2 Paragraph 2
list
item (2)
para
XSL Output.html
Paragraph 1
Paragraph 2
Item 1
Item 2
Paragraph 3
Append Insert Add child
Append Insert Add child
Append Insert Add child
inputXML
XSL Output.html
Terminé

Exemples d ’utilisation

- CSS et XSL à tester au laboC
  - Récupérer le code correspondant aux transparents qui suivent dans le document de cours HTML fourni
  - Tester et/ou modifier via les outils disponibles
- Présentation de Xpath (pour commencer)

Exemple 1

- « bonjour le monde »

Utilisation de CSS (1 : le doc)

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE accueil SYSTEM "bonjour.dtd">
<?xml:stylesheet type="text/css"
  href="bonjour.css"?>
<document>
  <salutation>Bonjour monde
   !</salutation>
</document>
```

Utilisation de XSL (1 : le doc)

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE accueil SYSTEM "bonjour.dtd">
<?xml:stylesheet type="text/xsl"
  href="bonjour.xsl"?>
<document>
 <salutation>Bonjour monde
  !</salutation>
</document>
```

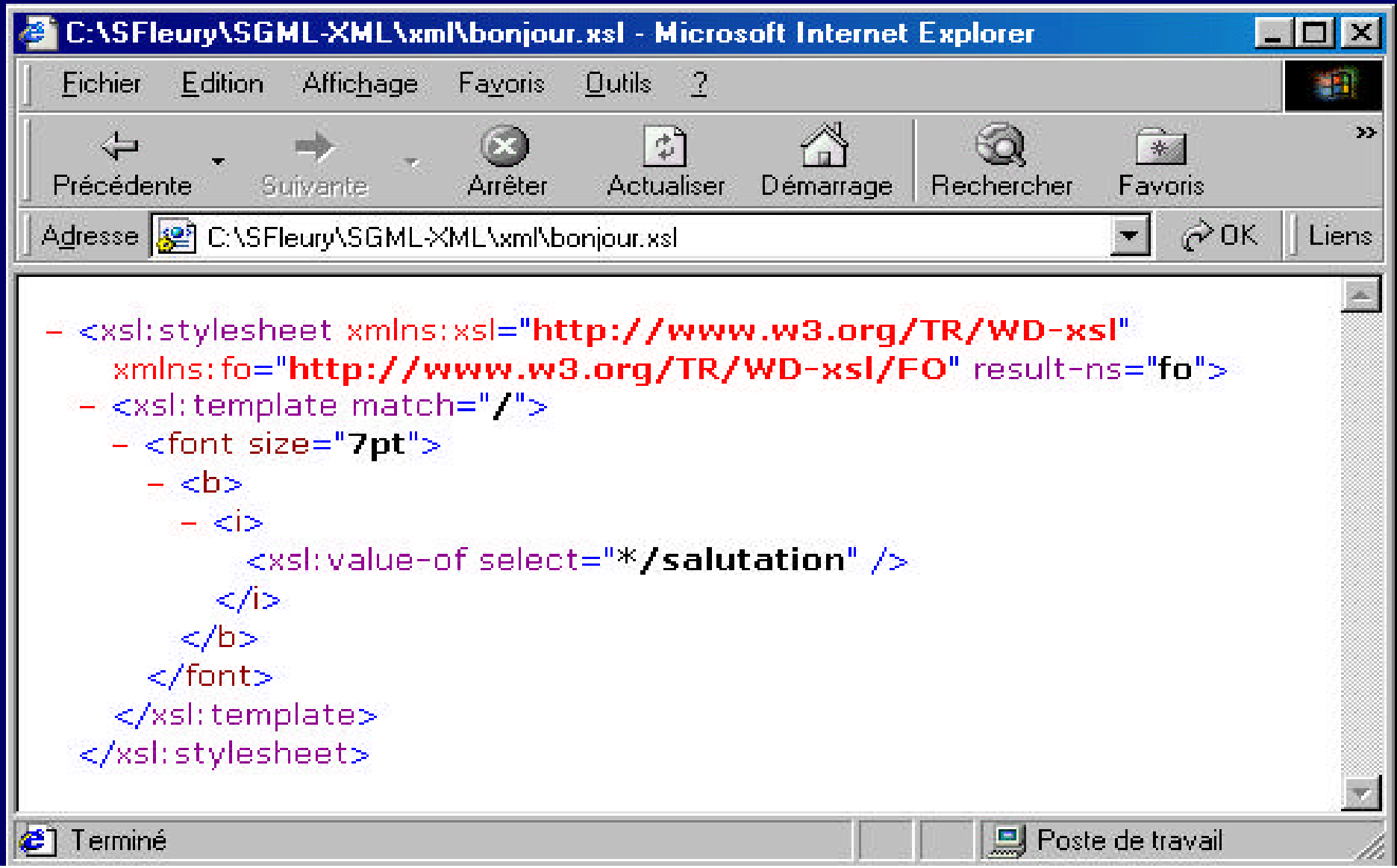
Utilisation de CSS/XSL (2 : la DTD)

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

<!ELEMENT document (#PCDATA)>
<!ELEMENT salutation (#PCDATA)>
```

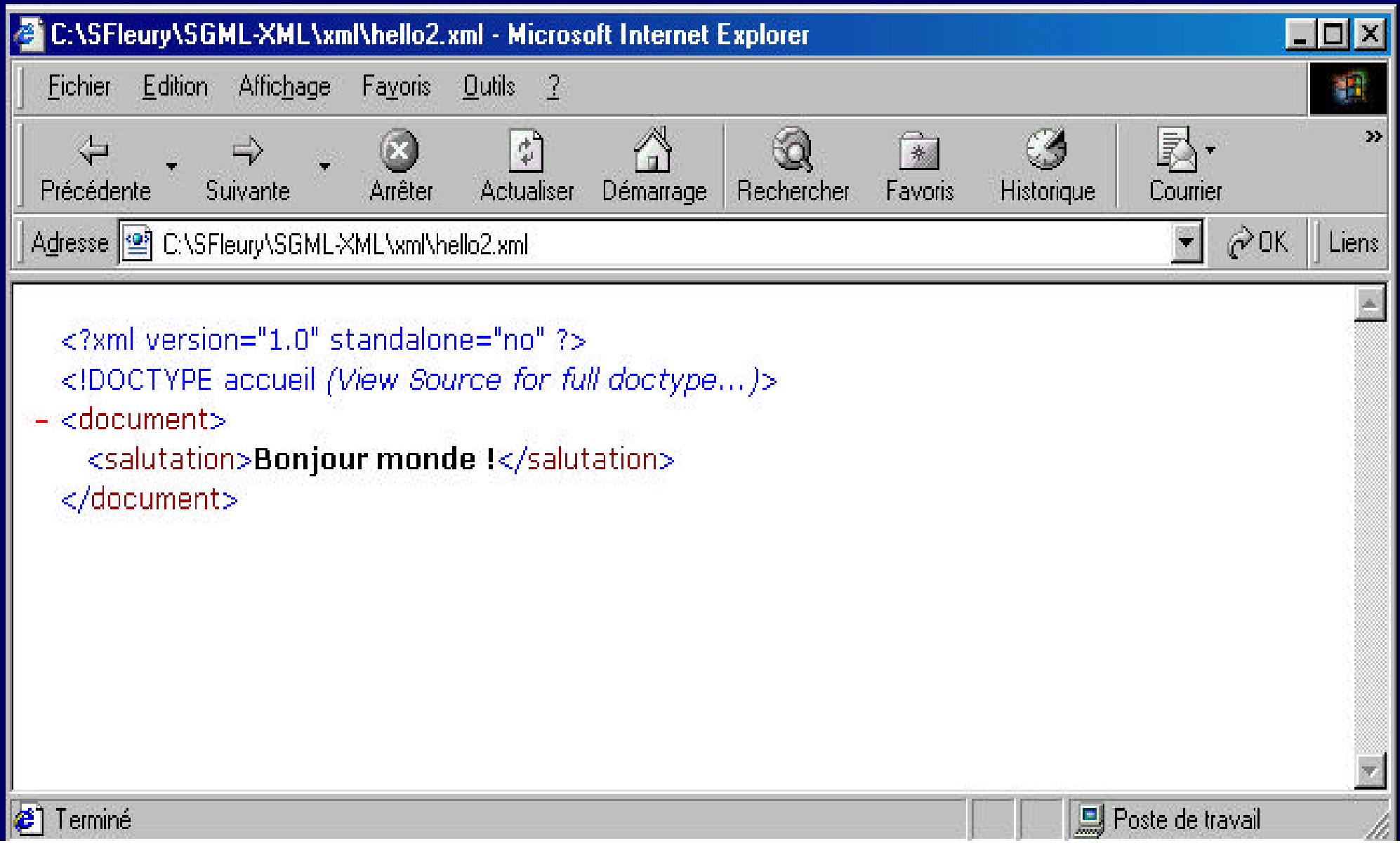
Utilisation de CSS (3 : la feuille de style)

```
<STYLE TYPE="text/css">
document  {font-family: arial,
  helvetica}
salutation{font-size:16pt;
font-style:italic;
font-weight:bold}
</STYLE>
```

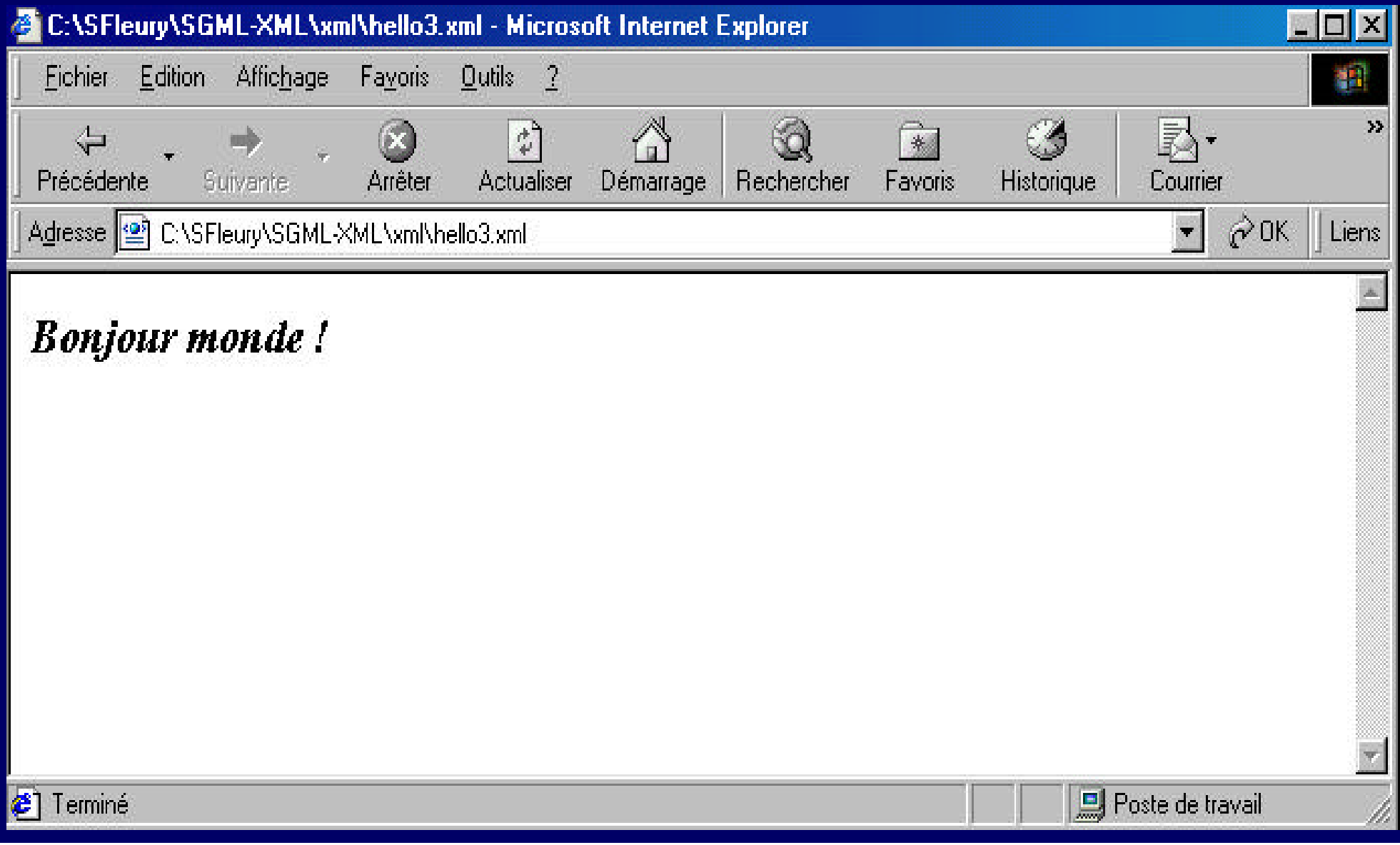
Utilisation de XSL (3 : la feuille de style)



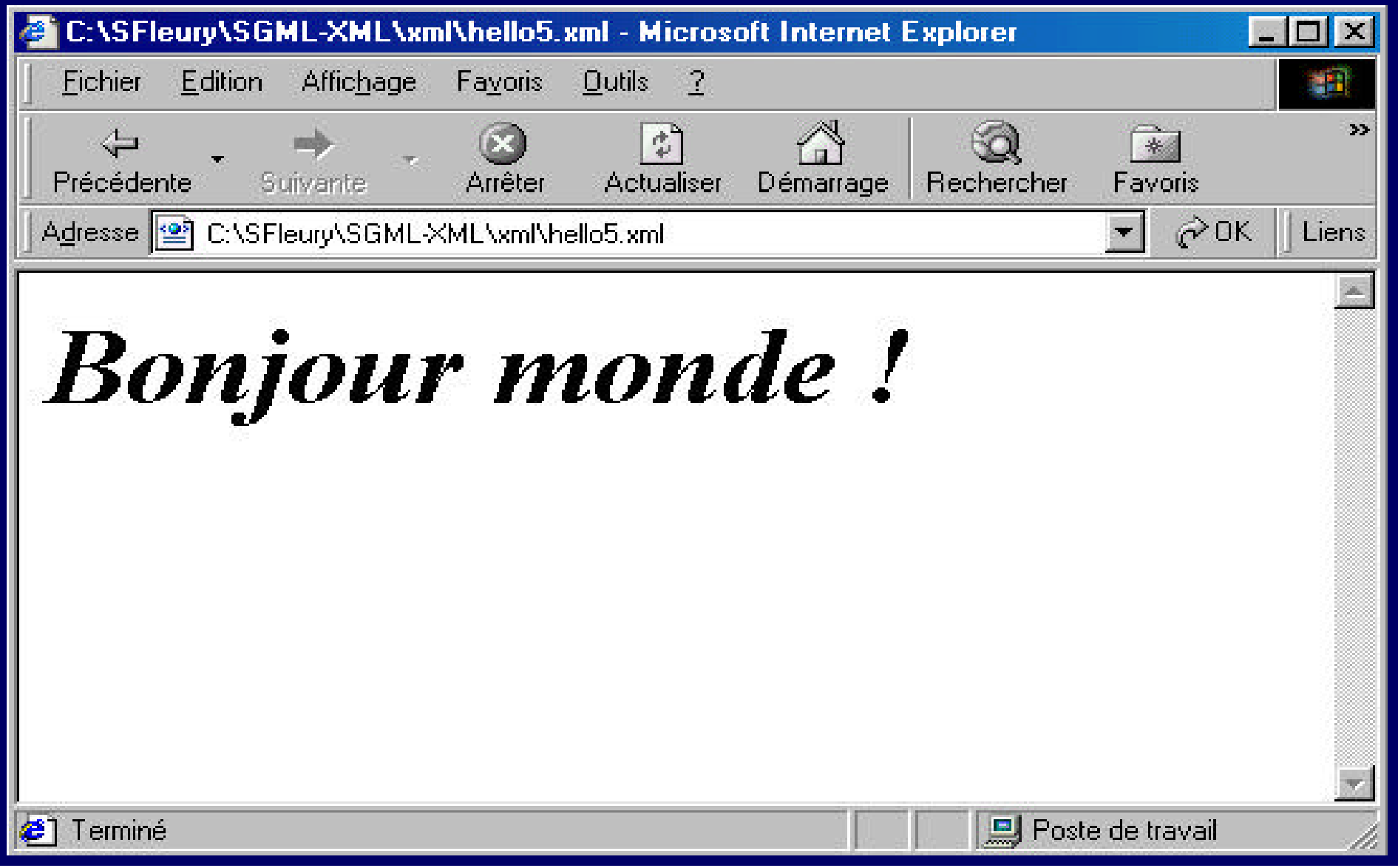
Rendu avec IE 5 sans CSS



Rendu avec IE 5 avec CSS



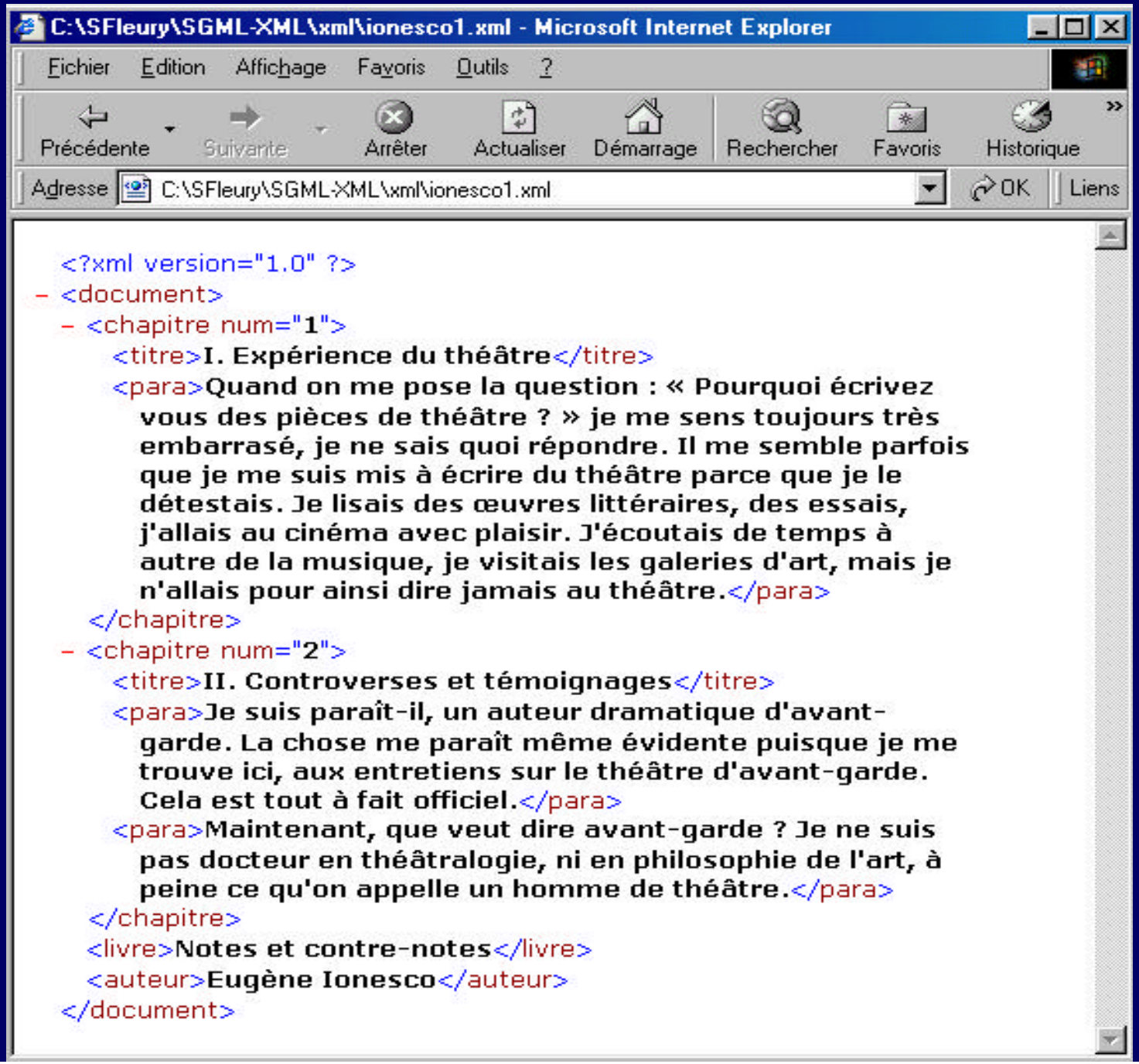
Rendu avec IE 5 avec XSL



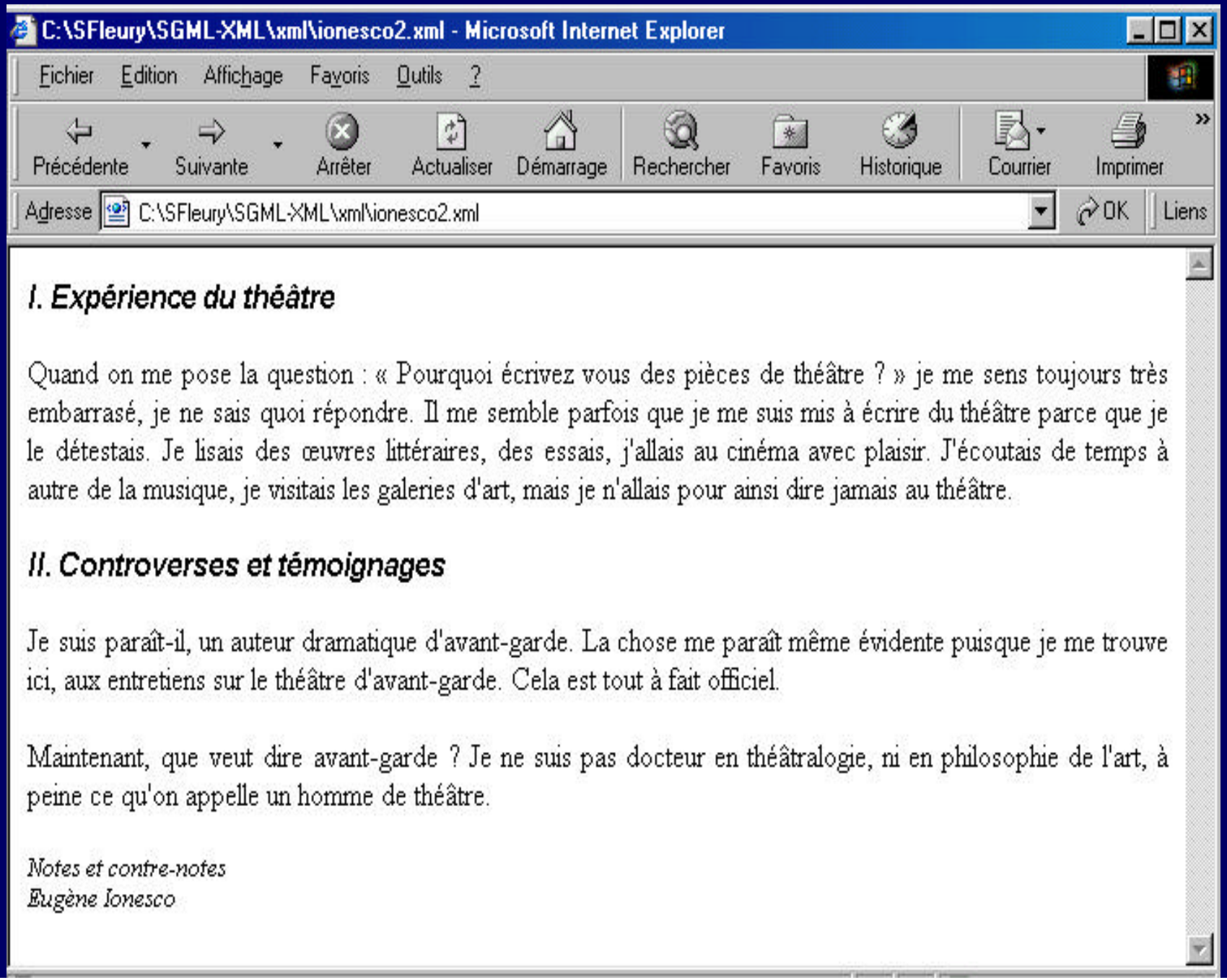
Exemple 2

- Une pièce de Ionesco

```
<?xml version="1.0"?>
<document>
  <chapitre num="1">
    <titre>I. Exp&#233;rience du th&#233;&#226;tre</titre>
<para>Quand on me pose la question : &#171; Pourquoi &#233;crivez vous des
pi&#232;ces de th&#233;&#226;tre ? &#187; je me sens toujours
tr&#232;s embarras&#233;, je ne sais quoi r&#233;pondre. Il me semble
parfois que je me suis mis &#224; &#233;crire du
th&#233;&#226;tre parce que je le d&#233;testais. Je lisais des &#339;uvres
litt&#233;raires, des essais, j'allais au cin&#233;ma
 avec plaisir. J'&#233;coutais de temps &#224; autre de la musique, je
visitais les galeries d'art, mais je n'allais pour ainsi
 dire jamais au th&#233;&#226;tre.
</para>
</chapitre>
<chapitre num="2">
    <titre>II. Controverses et t&#233;moignages</titre>
<para>Je suis para&#238;t-il, un auteur dramatique d'avant-garde. La chose
me para&#238;t
m&#234;me &#233;vidente puisque je me trouve ici, aux entretiens sur le
th&#233;&#226;tre
d'avant-garde. Cela est tout &#224; fait officiel.
</para>
<para>Maintenant, que veut dire avant-garde ? Je ne suis pas docteur en
th&#233;&#226;tralogie,
 ni en philosophie de l'art, &#224; peine ce qu'on appelle un homme de
th&#233;&#226;tre.
</para>
</chapitre>
<livre>Notes et contre-notes</livre>
<auteur>Eug&#232;ne Ionesco</auteur>
</document>
```


C:\SFleury\SGML-XML\xml\ionesco1.xml - Microsoft Internet Explorer
Fichier Edition Affichage Favoris Outils ?
Précédente Suivante Arrêter Actualiser Démarrage Rechercher Favoris Historique
Adresse C:\SFleury\SGML-XML\xml\ionesco1.xml
OK Liens
<?xml version="1.0" ?>
- <document>
- <chapitre num="1">
<titre>I. Expérience du théâtre</titre>
<para>Quand on me pose la question : « Pourquoi écrivez vous des pièces de théâtre ? » je me sens toujours très embarrasé, je ne sais quoi répondre. Il me semble parfois que je me suis mis à écrire du théâtre parce que je le détestais. Je lisais des œuvres littéraires, des essais, j'allais au cinéma avec plaisir. J'écoutais de temps à autre de la musique, je visitais les galeries d'art, mais je n'allais pour ainsi dire jamais au théâtre.</para>
</chapitre>
- <chapitre num="2">
<titre>II. Controverses et témoignages</titre>
<para>Je suis paraît-il, un auteur dramatique d'avant-garde. La chose me paraît même évidente puisque je me trouve ici, aux entretiens sur le théâtre d'avant-garde. Cela est tout à fait officiel.</para>
<para>Maintenant, que veut dire avant-garde ? Je ne suis pas docteur en théâtralogie, ni en philosophie de l'art, à peine ce qu'on appelle un homme de théâtre.</para>
</chapitre>
<livre>Notes et contre-notes</livre>
<auteur>Eugène Ionesco</auteur>
</document>

```
<STYLE TYPE="text/css">
document   {font-family: times new roman, helvetica;
           float: right;
           border-style: solid;
           width: auto;
           border-width: 2px;
           margin-left: 10%;
           margin-right: 10%;
           padding-left: 10px;
           padding-right: 10px;}
titre      {display: block; font-family: arial, helvetica;
           font-size:12pt;
           font-style:italic;
           font-weight:bold;
           margin-top: 3mm;
           margin-bottom: 5mm}
para       {display: block;
           text-align: justify;
           margin-bottom: 5mm;}

livre,auteur    {display: block;
           font-size:10pt;
           font-style: italic}

</STYLE>
```



I. Expérience du théâtre

Quand on me pose la question : « Pourquoi écrivez vous des pièces de théâtre ? » je me sens toujours très embarrasé, je ne sais quoi répondre. Il me semble parfois que je me suis mis à écrire du théâtre parce que je le détestais. Je lisais des œuvres littéraires, des essais, j'allais au cinéma avec plaisir. J'écoutais de temps à autre de la musique, je visitais les galeries d'art, mais je n'allais pour ainsi dire jamais au théâtre.

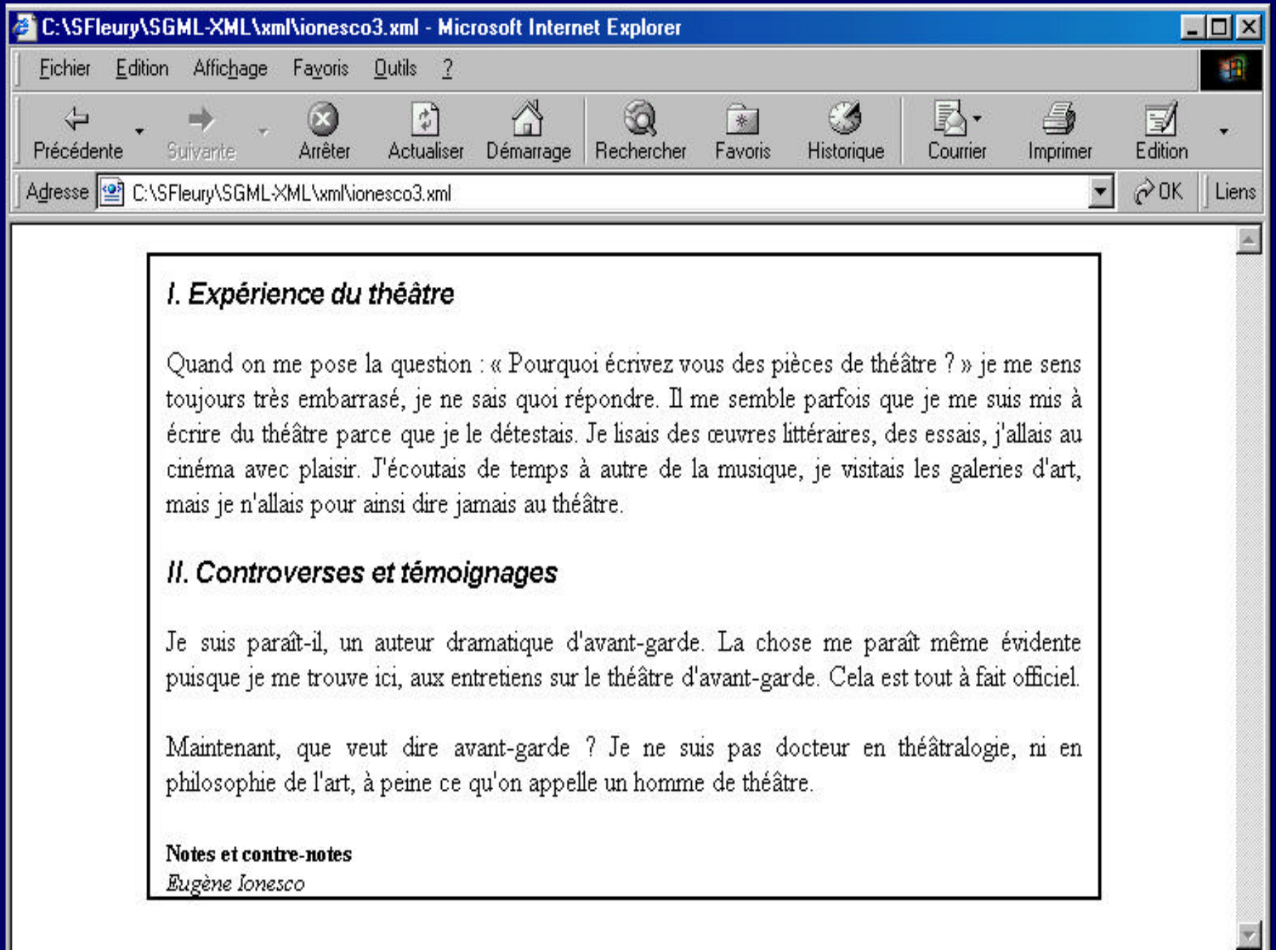
II. Controverses et témoignages

Je suis paraît-il, un auteur dramatique d'avant-garde. La chose me paraît même évidente puisque je me trouve ici, aux entretiens sur le théâtre d'avant-garde. Cela est tout à fait officiel.

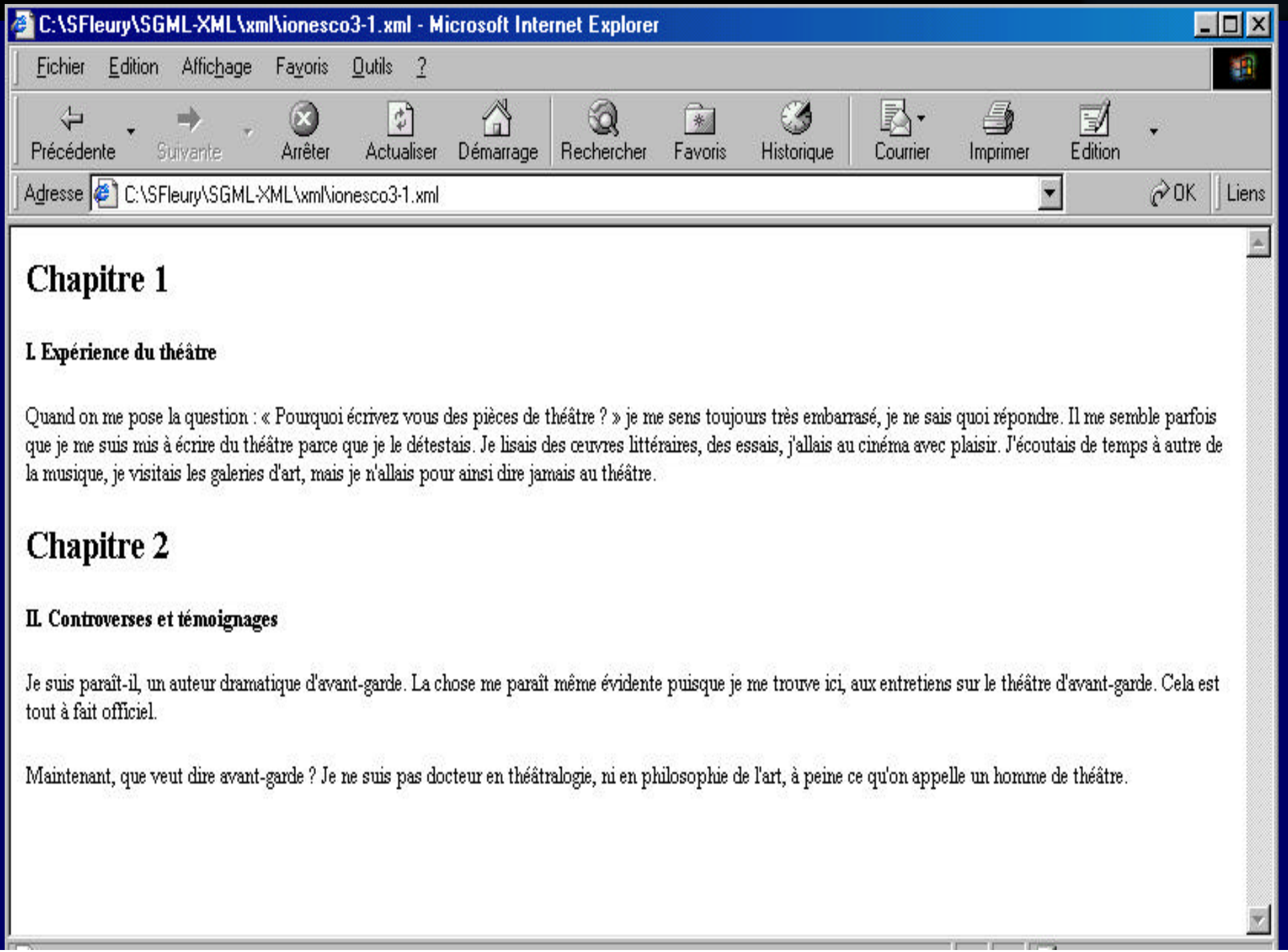
Maintenant, que veut dire avant-garde ? Je ne suis pas docteur en théâtralogie, ni en philosophie de l'art, à peine ce qu'on appelle un homme de théâtre.

Notes et contre-notes
Eugène Ionesco

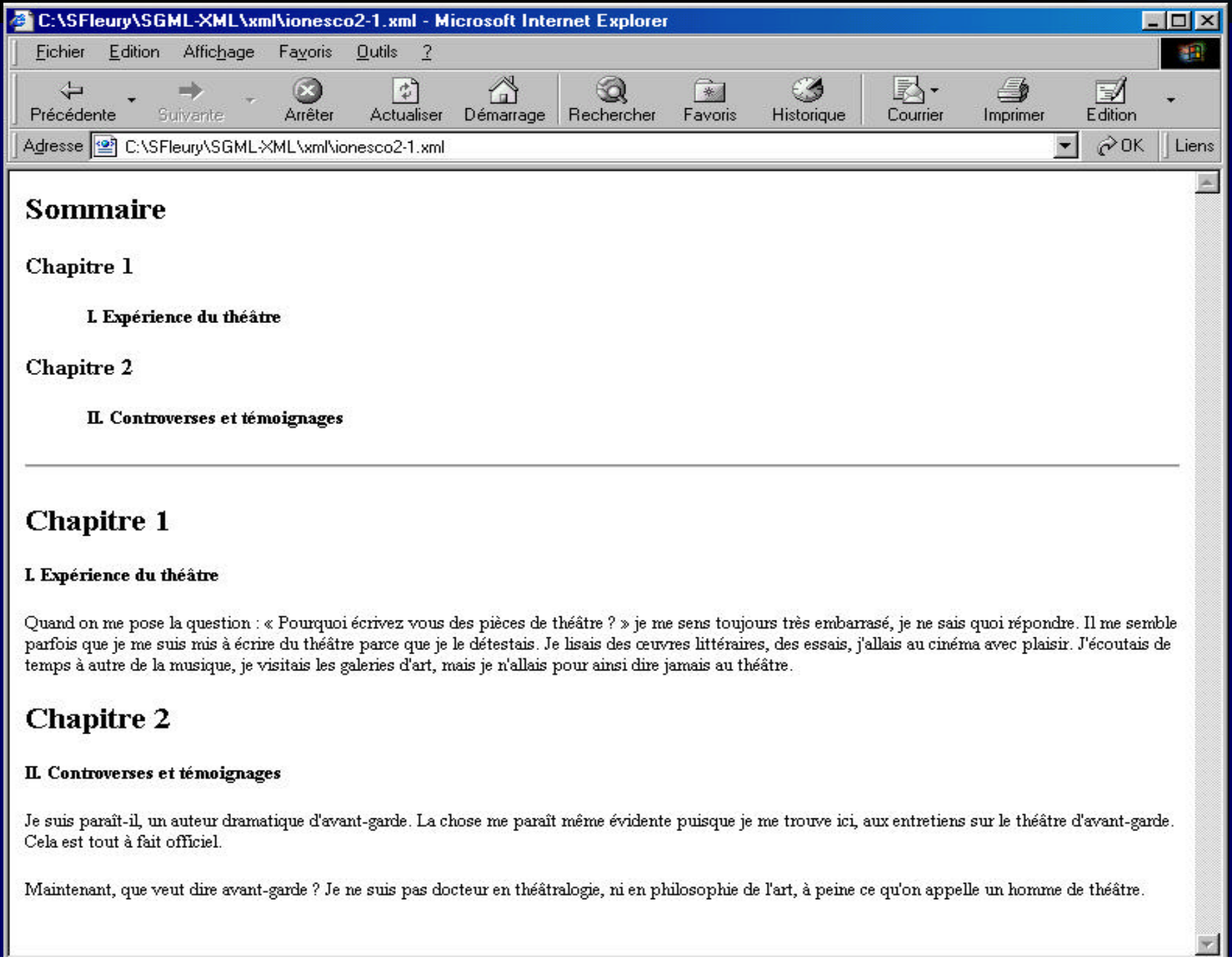
```
<STYLE TYPE="text/css">
A           {}
document    {
            float: right;
            border-style: solid;
            width: auto;
            border-width: 2px;
            margin-left: 10%;
            margin-right: 10%;
            padding-left: 10px;
            padding-right: 10px;
            background-image:
url(../../images/foggy.png);
            }
titre       {display: block; font-family: arial,
helvetica;
            font-size:12pt;
            font-style:italic;
            font-weight:bold;
            margin-top: 3mm;
            margin-bottom: 5mm}
 auteur           {display: block;
            font-size:10pt;
            font-style:italic;
            }
livre            {
            font-size: 10pt;
            font-weight:bold;
            margin-left: 80%;}
para        {display: block;
            text-align: justify;
            margin-bottom: 5mm;}
</STYLE>
```


C:\SFleury\SGML-XML\xml\ionesco3.xml - Microsoft Internet Explorer
Fichier Edition Affichage Favoris Outils ?
Précédente Suivante Arrêter Actualiser Démarrage Rechercher Favoris Historique Courrier Imprimer Edition
Adresse C:\SFleury\SGML-XML\xml\ionesco3.xml
OK Liens
I. Expérience du théâtre
Quand on me pose la question : « Pourquoi écrivez vous des pièces de théâtre ? » je me sens toujours très embarrasé, je ne sais quoi répondre. Il me semble parfois que je me suis mis à écrire du théâtre parce que je le détestais. Je lisais des œuvres littéraires, des essais, j'allais au cinéma avec plaisir. J'écoutais de temps à autre de la musique, je visitais les galeries d'art, mais je n'allais pour ainsi dire jamais au théâtre.
II. Controverses et témoignages
Je suis paraît-il, un auteur dramatique d'avant-garde. La chose me paraît même évidente puisque je me trouve ici, aux entretiens sur le théâtre d'avant-garde. Cela est tout à fait officiel.
Maintenant, que veut dire avant-garde ? Je ne suis pas docteur en théâtralogie, ni en philosophie de l'art, à peine ce qu'on appelle un homme de théâtre.
Notes et contre-notes
Eugène Ionesco

```
<?xml version="1.0"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<HTML>
  <BODY>
    <xsl:for-each select="document/chapitre">
  <H1>Chapitre <xsl:value-of select="@num"/>
  </H1>
      <b><xsl:value-of select="titre"/></b>
      <p><xsl:for-each select="para">
        <p><xsl:value-of/></p>
        </xsl:for-each></p>
    </xsl:for-each>
  </BODY>
</HTML>
</xsl:template>
</xsl:stylesheet>
```


C:\SFleury\SGML-XML\xml\ionesco3-1.xml - Microsoft Internet Explorer
Fichier Edition Affichage Favoris Outils ?
Précédente Suivante Arrêter Actualiser Démarrage Rechercher Favoris Historique Courrier Imprimer Edition
Adresse C:\SFleury\SGML-XML\xml\ionesco3-1.xml
OK Liens
Chapitre 1
I. Expérience du théâtre
Quand on me pose la question : « Pourquoi écrivez vous des pièces de théâtre ? » je me sens toujours très embarrasé, je ne sais quoi répondre. Il me semble parfois que je me suis mis à écrire du théâtre parce que je le détestais. Je lisais des œuvres littéraires, des essais, j'allais au cinéma avec plaisir. J'écoutais de temps à autre de la musique, je visitais les galeries d'art, mais je n'allais pour ainsi dire jamais au théâtre.
Chapitre 2
II. Controverses et témoignages
Je suis paraît-il, un auteur dramatique d'avant-garde. La chose me paraît même évidente puisque je me trouve ici, aux entretiens sur le théâtre d'avant-garde. Cela est tout à fait officiel.
Maintenant, que veut dire avant-garde ? Je ne suis pas docteur en théâtralogie, ni en philosophie de l'art, à peine ce qu'on appelle un homme de théâtre.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<HTML>
  <BODY>
      <H1>Sommaire</H1>
      <xsl:for-each select="document/chapitre">

        <H2>Chapitre <xsl:value-of select="@num"/></H2>
        <blockquote>
            <b><xsl:value-of select="titre"/></b>
        </blockquote>
      </xsl:for-each>
      <HR/>
      <xsl:for-each select="document/chapitre">

        <H1>Chapitre <xsl:value-of select="@num"/></H1>
        <b><xsl:value-of select="titre"/></b>
        <p><xsl:for-each select="para">
            <p><xsl:value-of/></p>
            </xsl:for-each></p>
      </xsl:for-each>
  </BODY>
</HTML>
</xsl:template>
</xsl:stylesheet>
```


C:\SFleury\SGML-XML\xml\ionesco2-1.xml - Microsoft Internet Explorer
Fichier Edition Affichage Favoris Outils ?
Précédente Suivante Arrêter Actualiser Démarrage Rechercher Favoris Historique Courrier Imprimer Edition
Adresse C:\SFleury\SGML-XML\xml\ionesco2-1.xml
OK Liens
Sommaire
Chapitre 1
I. Expérience du théâtre
Chapitre 2
II. Controverses et témoignages
Chapitre 1
I. Expérience du théâtre
Quand on me pose la question : « Pourquoi écrivez vous des pièces de théâtre ? » je me sens toujours très embarrasé, je ne sais quoi répondre. Il me semble parfois que je me suis mis à écrire du théâtre parce que je le détestais. Je lisais des œuvres littéraires, des essais, j'allais au cinéma avec plaisir. J'écoutais de temps à autre de la musique, je visitais les galeries d'art, mais je n'allais pour ainsi dire jamais au théâtre.
Chapitre 2
II. Controverses et témoignages
Je suis paraît-il, un auteur dramatique d'avant-garde. La chose me paraît même évidente puisque je me trouve ici, aux entretiens sur le théâtre d'avant-garde. Cela est tout à fait officiel.
Maintenant, que veut dire avant-garde ? Je ne suis pas docteur en théâtralogie, ni en philosophie de l'art, à peine ce qu'on appelle un homme de théâtre.

XSL

- XSL : Extensive Stylesheet Language
- Il est composé de deux parties principales:
  - XSLT: XSL Transformations
    - un language de transformation
    - XSLT-1.0 est une recommandation du W3C
  - XSL-FO : XSL Formating Objects
    - un jeux d'instruction de formatage en XML destiné a la présentation :
      - Modèle de formatage: boîtes, positionnement, ordonnancement...
      - Propriétés d'affichage: text-align, margin-left, background...
    - Exemple d'utilisation avec IBM XSL Composer en TD

Retour Sommaire

XSLT

- Application XML qui spécifie les règles permettant de transformer un document XML en un autre
- Un document XSLT contient des modèles
  - Le processeur XSLT compare le document aux modèles
  - Pour un modèle adapté, il copie le contenu du modèle dans l'arbre de sortie

Principes de base

- Une feuille de styles XSLT est un document XML bien formé mais non valide
- Une transformation XSLT est une application non validante; elle est applicable à toute instance bien formée, valide ou non
- Une instruction de traitement:
  `<?xml-stylesheet type="text/xsl" href="`*`mafeuille`*`.xsl" ?>`
  dans le prologue de l'instance fait le lien avec la feuille de styles

- Un document XSLT est un document XML
  - Élément racine : `stylesheet` ou `transform`
  - Attribut `xmlns` : l'espace de nom des éléments XSLT est :
    - http://www.w3.org/1999/XSL/Transform
    - Le préfixe traditionnel est xsl
  - Attribut version
    - Valeur 1.0 (pour XSLT 1.0)

Structure d’une feuille XSLT

Prologue (exemple):

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Corps (suite de gabarits ou *templates*):

```
<xsl:template match="exemple">
  ...
</xsl:template>
...
```

Épilogue:

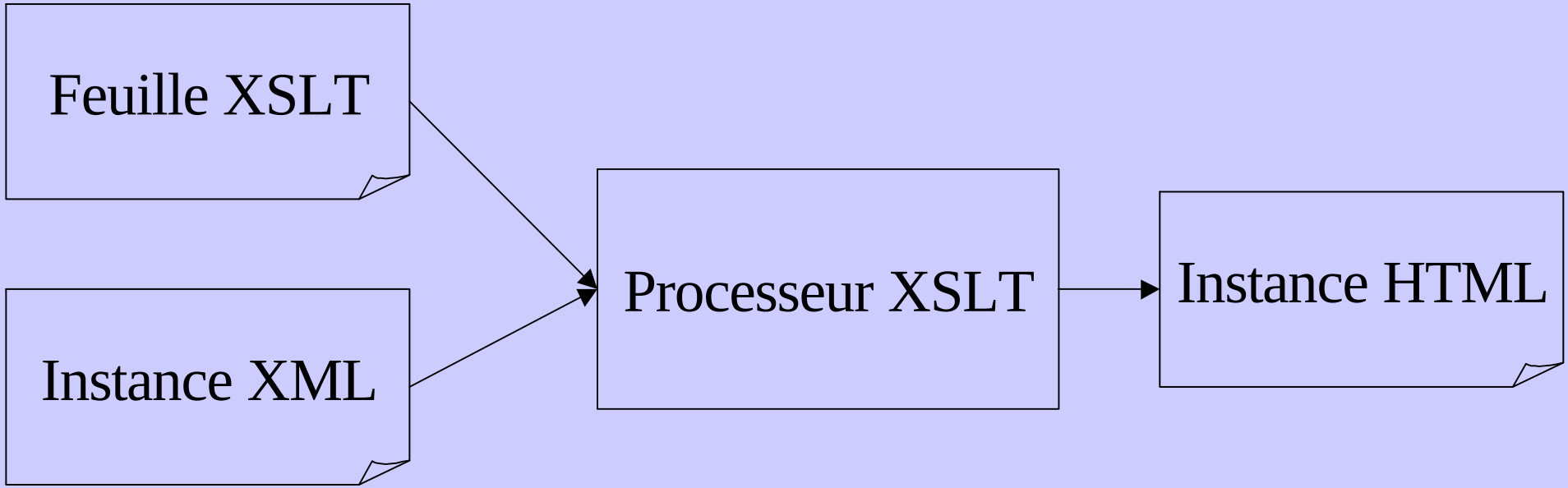
```
</xsl:stylesheet>
```

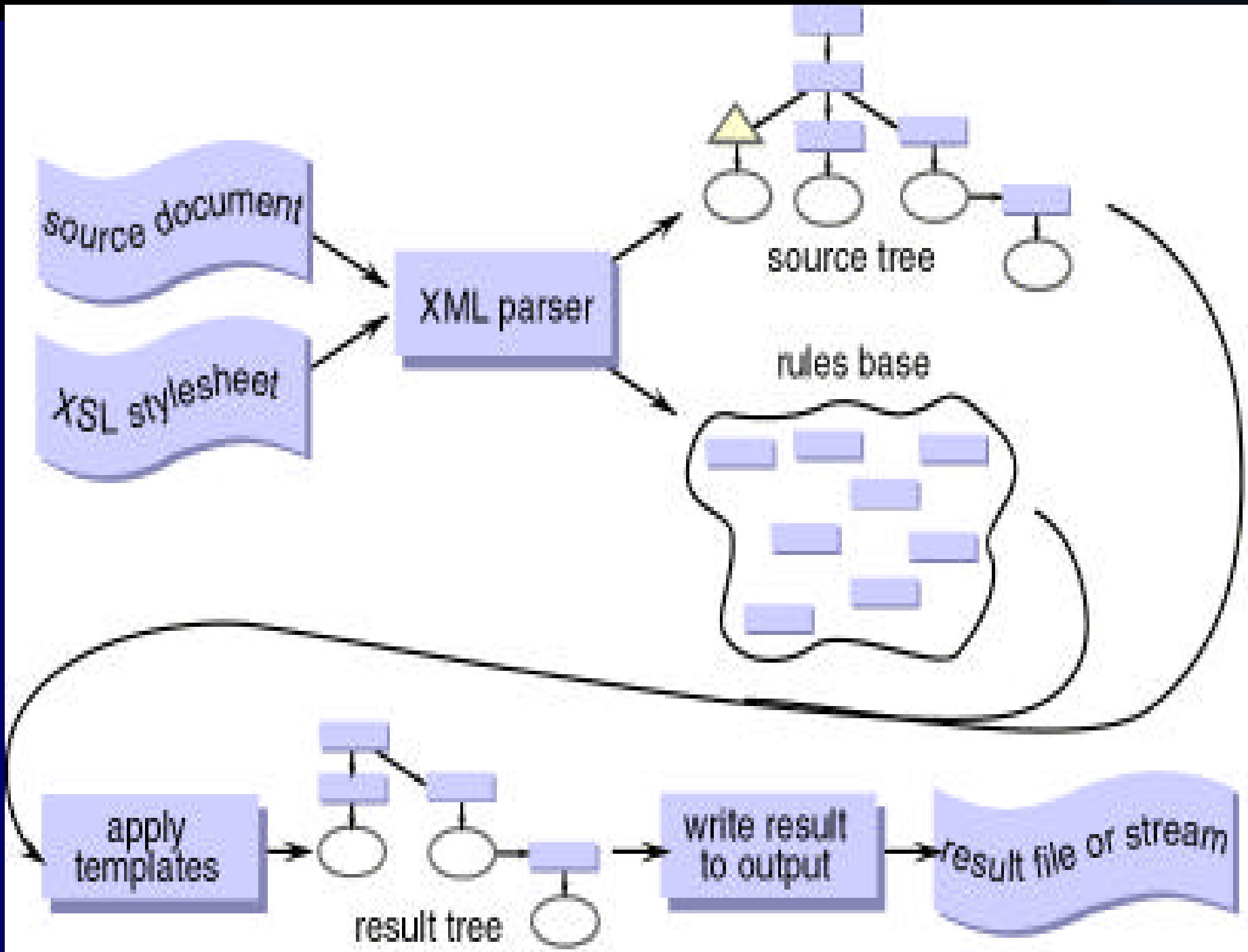
Exemple de feuille de style XSLT

```
<?xml version='1.0' encoding='ISO-
 8859-1'?>
<xsl:stylesheet version='1.0'
 xmlns:xsl='http://www.w3.org/1999
 /XSL/Transform'>
</xsl:stylesheet>
```

- essayer cette feuille de style sur un document XML...

Transformation XSLT




source document
XSL stylesheet
XML parser
source tree
rules base
apply templates
result tree
write result to output
result file or stream

Processeurs XSLT

- But :
  - (1) lire un document XSLT et un document XML
  - (2) produire un document de sortie qui transforme le document XML initial suivant les modèles contenus dans le document XSLT
- Processeurs
  - MSXSML de Microsoft intégré dans IE5
  - SAXON
  - Xalan : http://xml.apache.org/xalan

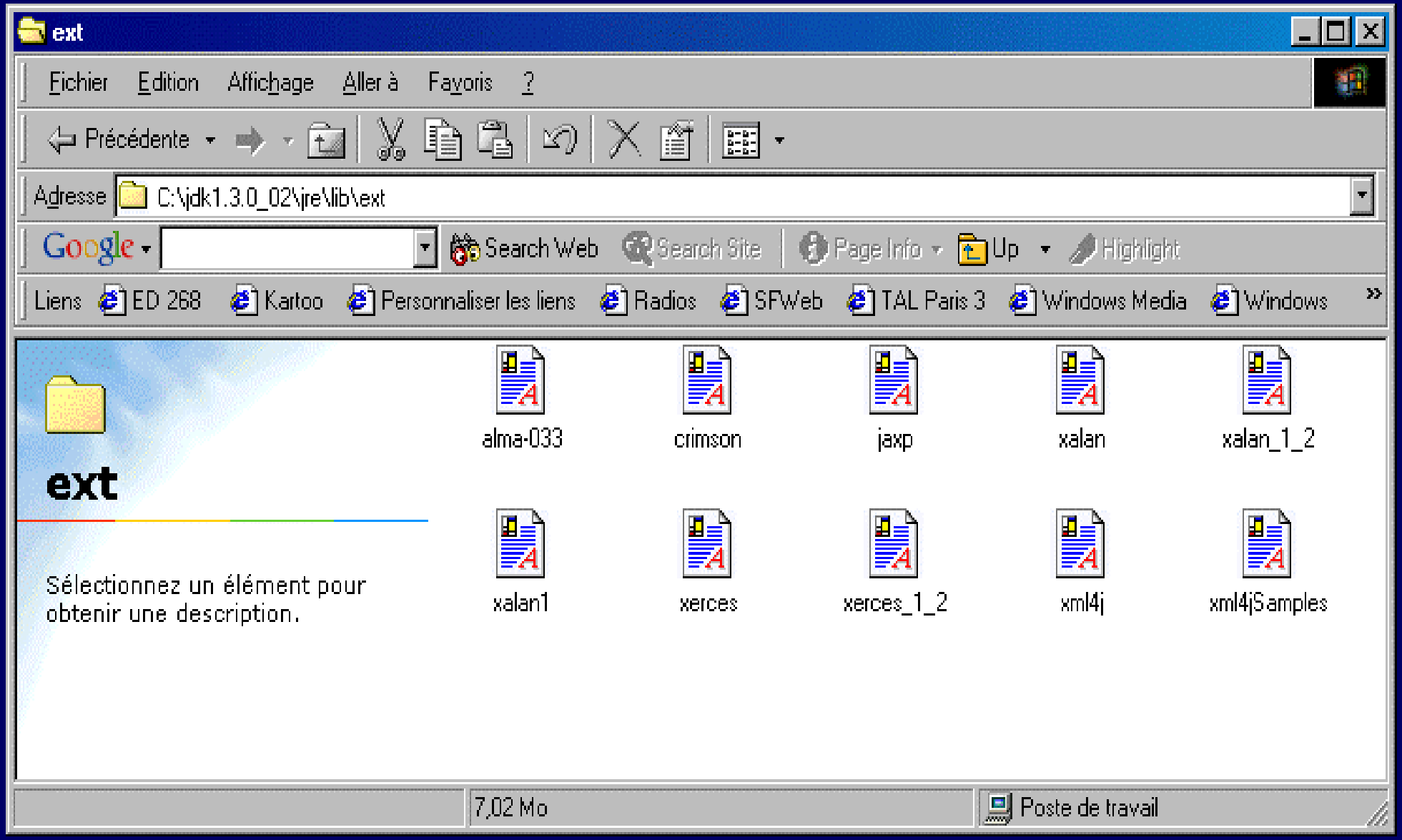
Processeur en ligne de commandes

- Xalan
  - Installation : cf mode d'emploi

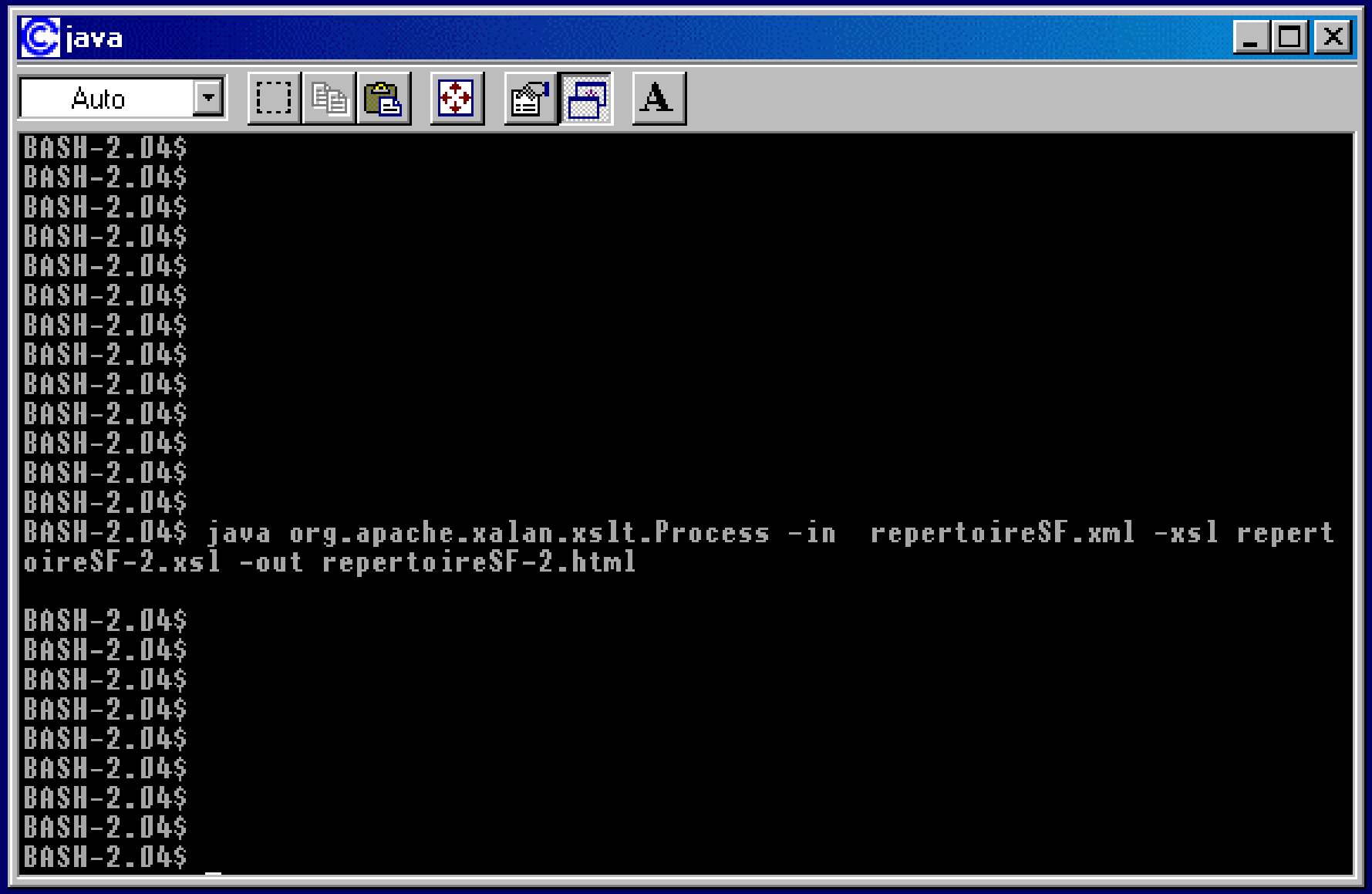
```
java org.apache.xalan.xslt.Process
-IN input.xml
-XSL input.xsl
-OUT output.txt
```

  - Sous java2, on commence par placer les fichiers jar qui contiennent les classes utilisées dans le répertoire jre/lib/ext

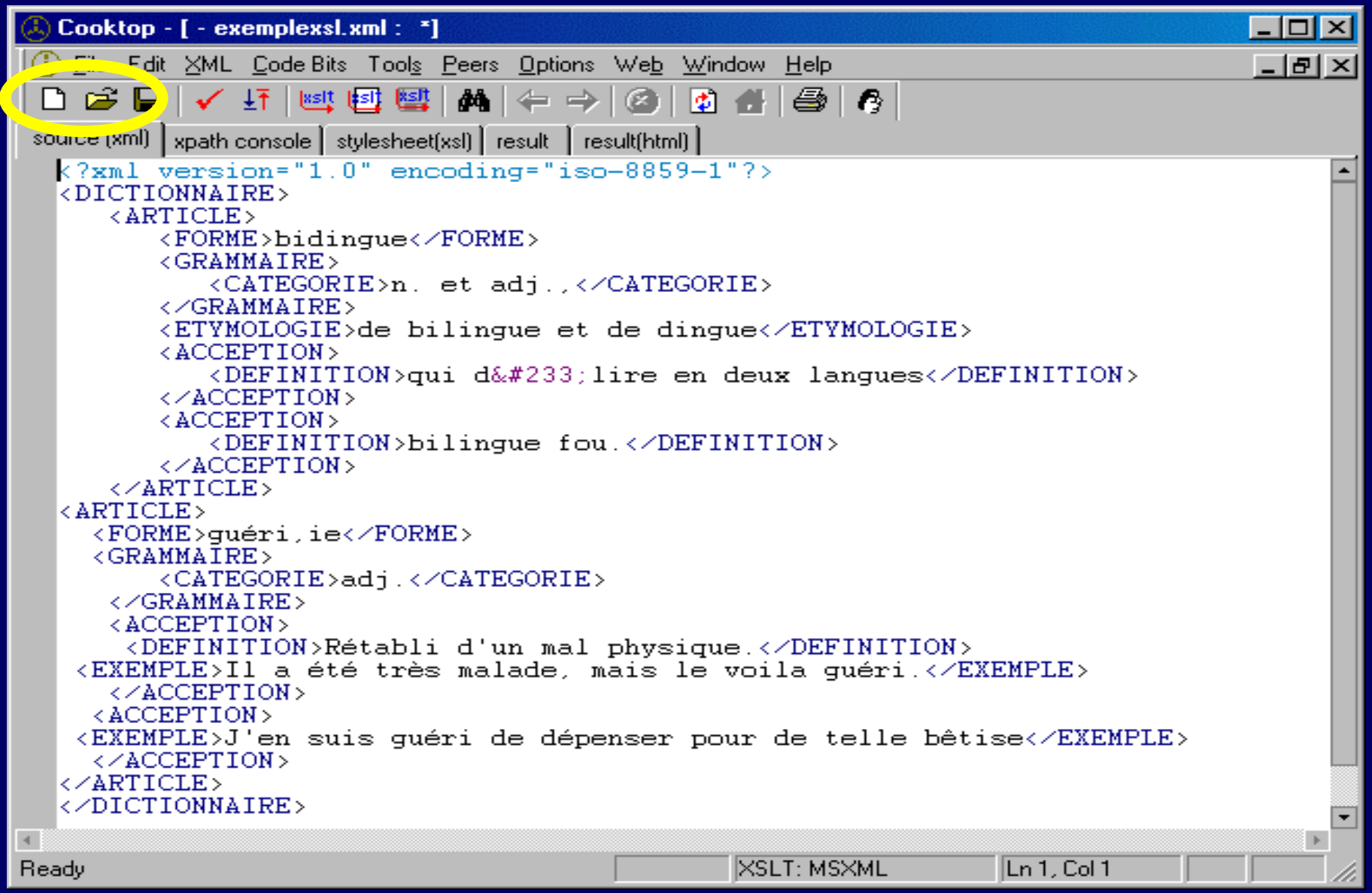
Contenu du répertoire jre/lib/ext



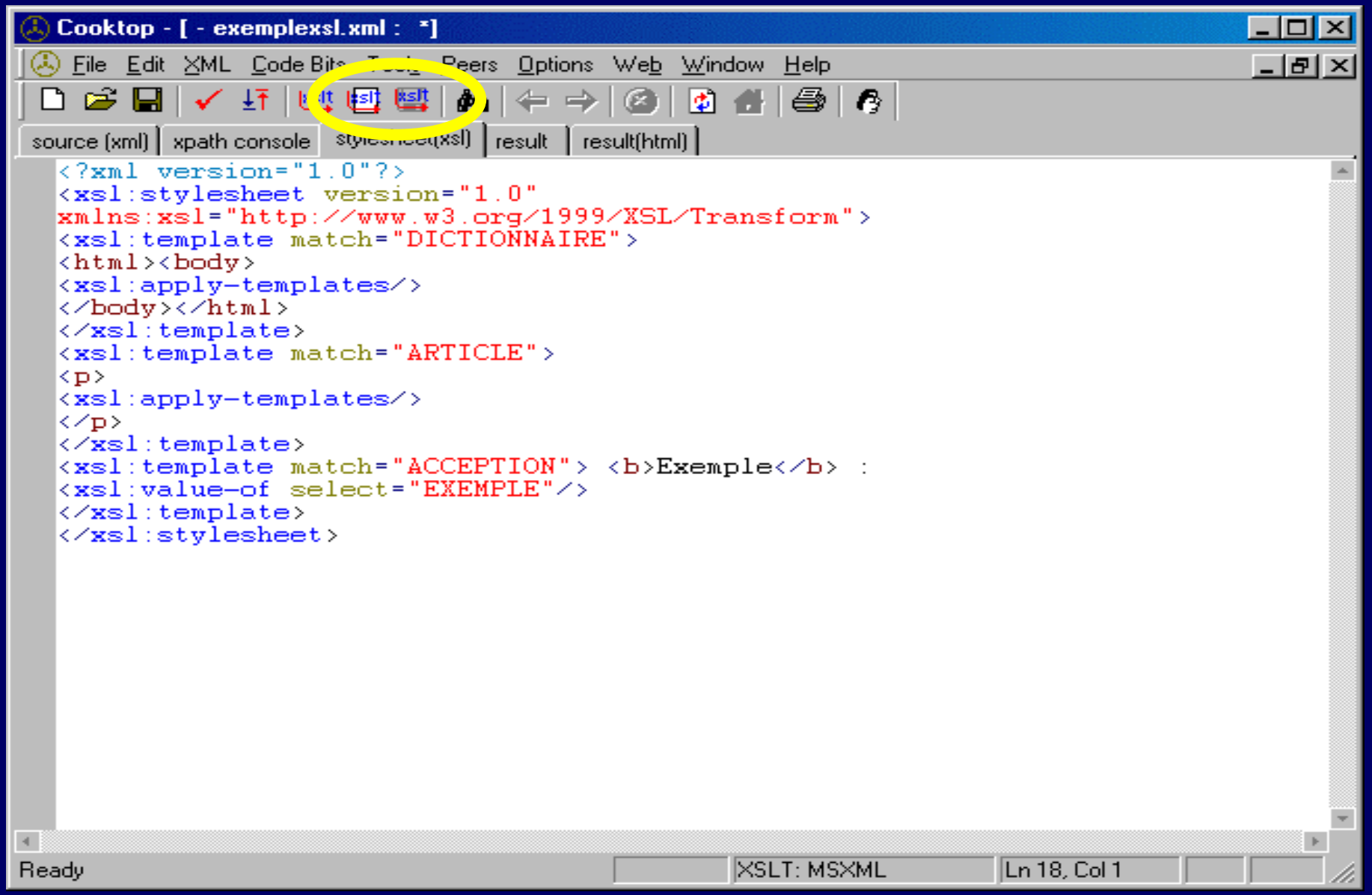
Activation du processeur



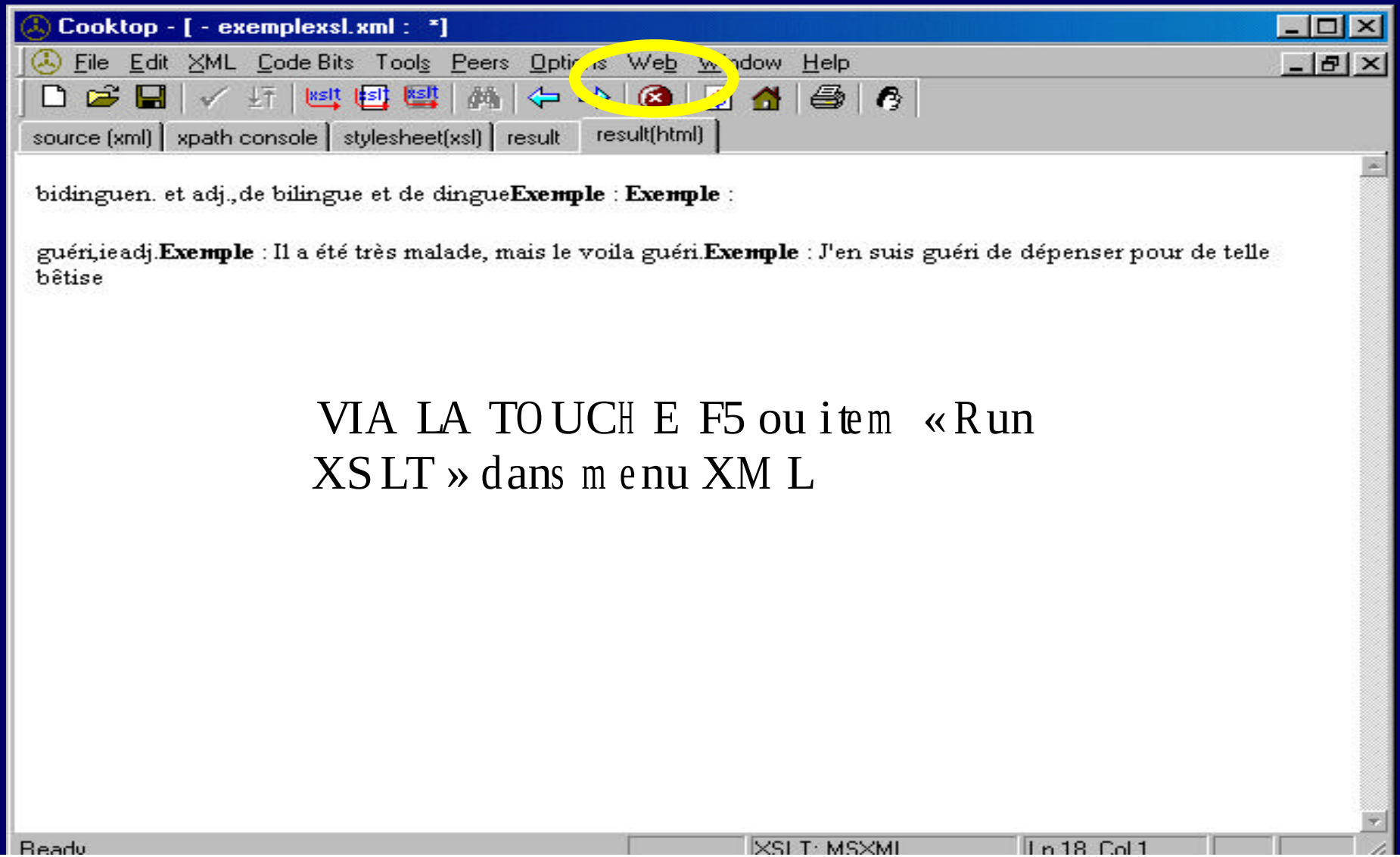
XSL et XMLCootop (1)



XSL et XMLCootop (2)



XSL et XMLCootop (3)



VIA LA TOUCHE F5 ou item « Run XSLT » dans menu XML

Modèles XSLT

- Les transformations XSL se font sur l'arbre d'éléments correspondant au document XML, plus précisément sur les noeuds du DOM :
  - `root, element, text, attribut, comment...`
- Un template/modèle est une règle de transformation pour des noeuds
  - Chaque modèle est représenté par un élément `xsl:template`
  - Cet élément a un attribut `match` dont la valeur est un chemin Xpath identifiant ce à quoi il correspond

Notion de base: gabarit

- Gabarit = *template* = règle
- Chaque gabarit indique au processeur XSLT comment traiter certains éléments des instances dans certains contextes
- Exemple:

```
<xsl:template match="général">
  <P><B>Accessible au public</B></P>
</xsl:template>
```

Cas simple

`match="`*`identificateur-générique`*`"`

- S'applique à tous les éléments ayant cet identificateur générique
- Exemple:

```
<xsl:template match="auteur">
  <P><B>Auteur: </B><xsl:value-of select="." /></P>
</xsl:template>
```

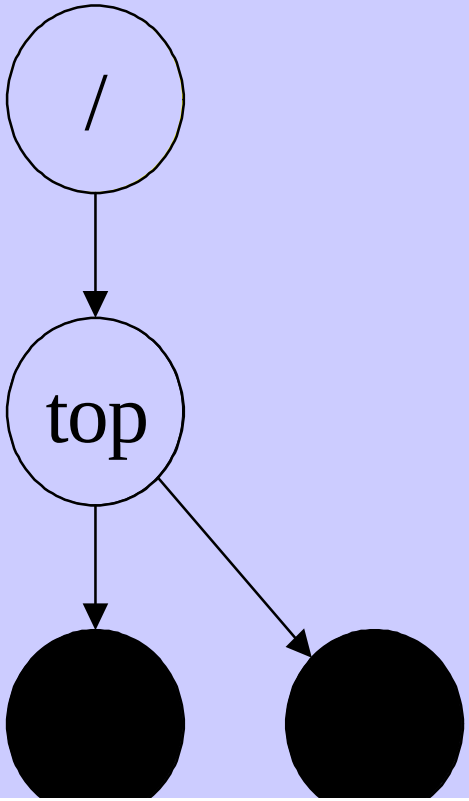
Comment ça marche, exactement?

Pseudo-élément racine

Le processeur XSLT considère que l'élément de plus haut niveau de l'instance a un parent, désigné par "/":

```
<?xml version = "1.0" ?>
<top>
  <a/><b/>
</top>
```

?



Action du processeur (1/2)

- Regarde si (au moins) un des gabarits s'applique au pseudo-élément "/"
- Si oui, applique (ou exécute) ce gabarit
  - l'extrant de cette exécution de gabarit devient l'extrant de la transformation XSLT
- [Sinon, applique un gabarit fixe prédéfini]
- C'est TOUT

Action du processeur (2/2)

- Si on veut que d'autres gabarits soient exécutés, il faut qu'ils soient appelés, directement ou indirectement, par le gabarit appliqué à "/"

Extrant d'une exécution de gabarit

« L'extrant d'une exécution de gabarit est une chaîne de caractères égale au contenu du gabarit, dans lequel les "instructions XSLT" (s'il y en a) sont remplacées par l'extrant de leur exécution. »

Que peut contenir un gabarit?

- Des fragments HTML
  - Ces fragments se retrouvent tels quels dans l'extrant du gabarit
- Des instructions XSLT:
  - `<xsl:apply-templates />`
  - `<xsl:apply-templates select="expression" />`
  - `<xsl:value-of select="expression" />`
  - etc.

Élément courant

- "L'élément courant" dans un gabarit est l'élément pour lequel le gabarit est exécuté
- Exemple, avec:

```
<xsl:template match="auteur">
```

l'élément courant est un des éléments "auteur" de l'instance XML traitée

<xsl:value-of select="*expression*" />

- Extrait une chaîne de caractères du document, habituellement quelque part dans l'élément courant
- `select="."` retourne la valeur textuelle de l'élément courant
- `select="@attrib"` retourne la valeur d'un attribut de l'élément courant
- `select="IDGen"` retourne la valeur textuelle d'un (ou plusieurs) sous-éléments de l'élément courant

<xsl:apply-templates />

- Cause le traitement successif de chacun des "noeuds enfants" de l'élément courant par le gabarit approprié
- Un noeud enfant est soit un sous-élément, soit un noeud #PCDATA (texte)
- L'ordre des enfants est respecté
- L'extrant de l'instruction est la concaténation des extrants résultant du traitement des enfants

<xsl:apply-templates select="*expression*"/>

- Fait traiter certains enfants spécifiques (répondant au critère de sélection)
- Cas simple: expression = identif. générique
- L'extrant de l'instruction est la concaténation des extrants résultant du traitement des enfants sélectionnés
- Permet de faire du "réarrangement" de sous-éléments

Gabarits prédéfinis (1/2)

- Des gabarits fixes prédéfinis existent pour:
  - Le pseudo-élément racine "/"
  - Les éléments XML
  - Les attributs
  - Les nœuds de texte (#PCDATA)
  - etc.

Gabarits prédéfinis (2/2)

- Pour "/" et éléments, le gabarit prédéfini est:

```
<xsl:template>
 <xsl:apply-templates />
</xsl:template>
```

- Pour noeuds textuels:

```
<xsl:template>
   <xsl:value-of select="." />
</xsl:template>
```

Modèle : exemple 1

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<xsl:stylesheet version='1.0'
  xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:template match= 'ARTICLE'> un article du dictionnaire</xsl:template>
</xsl:stylesheet>
```

- Ce modèle indique qu'à chaque fois que l'on rencontre un élément ARTICLE, le processeur XSLT doit produire le texte « un article du dictionnaire »

Document XML (1)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DICTIONNAIRE>
   <ARTICLE>
      <FORME>bidingue</FORME>
      <GRAMMAIRE>
         <CATEGORIE>n. et adj.,</CATEGORIE>
      </GRAMMAIRE>
      <ETYMOLOGIE>de bilingue et de dingue</ETYMOLOGIE>
      <ACCEPTION>
         <DEFINITION>qui      d&#233;lire      en      deux
  langues</DEFINITION>
      </ACCEPTION>
      <ACCEPTION>
         <DEFINITION>bilingue fou.</DEFINITION>
      </ACCEPTION>
   </ARTICLE>
  ...
</DICTIONNAIRE>
```

Exemple 2

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

<xsl:stylesheet
version="1.0"

xmlns:xsl="http://www.w
3.org/1999/XSL/Transfor
m">

<xsl:template
match="ARTICLE">

<p>voici un mot</p>

</xsl:template>

</xsl:stylesheet>
```

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

<p>voici un mot</p>

<p>voici un mot</p>

<p>voici un mot</p>

.....
```

Définition de modèle

```
<xsl:template [match="pattern"] [name="nom"]> format de sortie, ..., transformations </xsl:template>
```

- définit une règle de transformation, éventuellement nommée, éventuellement sur les noeuds spécifiés dans le pattern
- Si plusieurs templates peuvent s'appliquer, le template le plus spécifique au niveau du pattern s'appliquera

Calculer la valeur d'un élément

- L'élément `xsl:value-of` calcule la valeur textuelle d'un élément particulier en entrée et l'insère dans le résultat
- La valeur de l'élément est son contenu textuel après « débalisage »
- L'attribut `select` permet de sélectionner l'élément dont la valeur sera calculée
  - La valeur de l'attribut select est un chemin Xpath

Valeur d'un élément

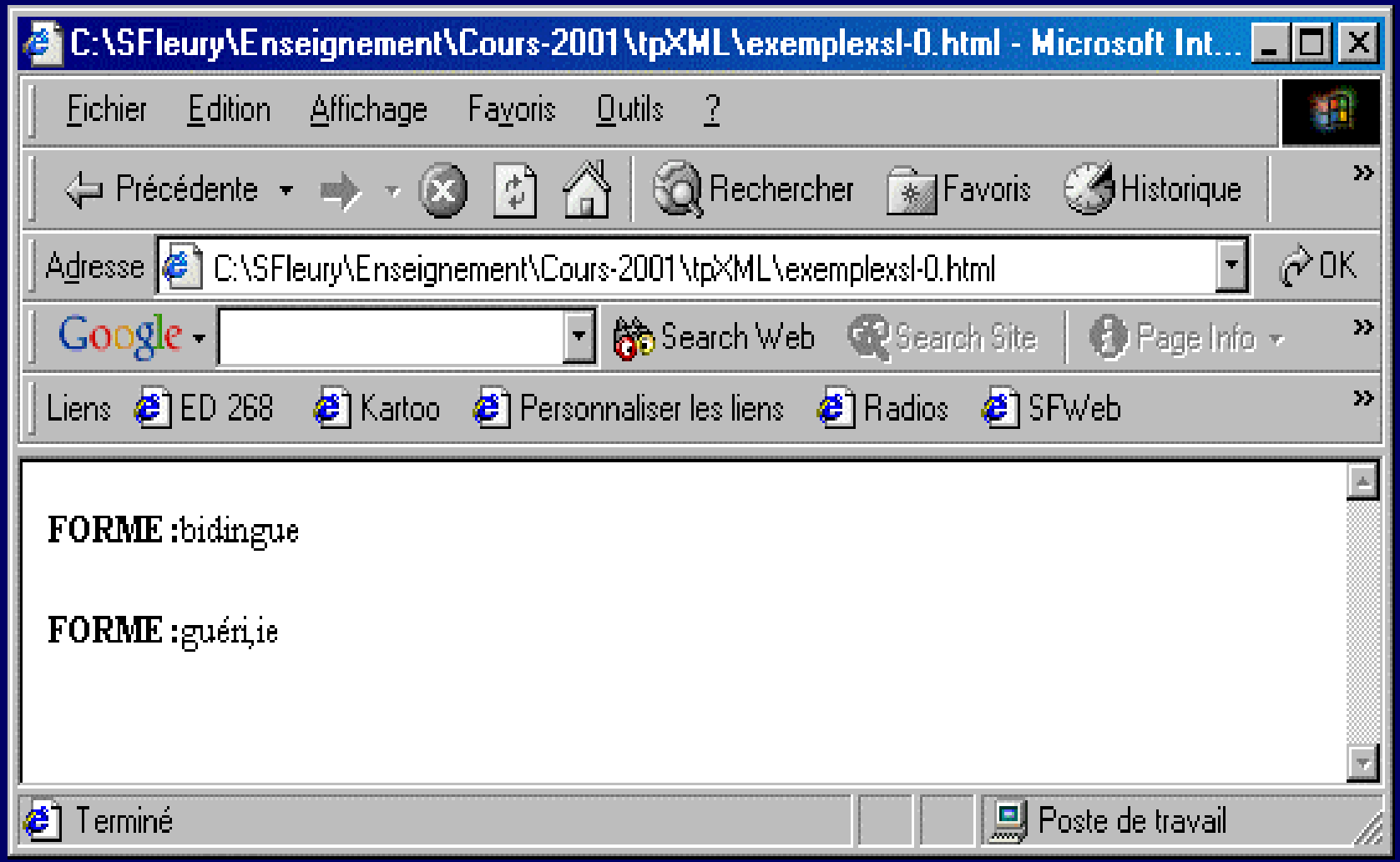
```
<xsl:value-of select="pattern"
  [disable-output-
  escaping="yes"]/>
```

- génère intégralement la valeur du prochain (seulement !) pattern spécifié en sortie ; le pattern définit une chaîne ou un élément (la valeur générée sera le texte contenu dans l'élément) ou un attribut (la valeur est celle de l'attribut); éventuellement, remplace les séquences d'échappement &...; par leurs caractères équivalents.

Valeur d'un élément : exemple 1

```
<?xml   version='1.0'   encoding='ISO-
8859-1'?>
<xsl:stylesheet              version='1.0'
xmlns:xsl='http://www.w3.org/1999/XSL
/Transform'>
<xsl:template match= 'ARTICLE'>
<p><b>FORME:</b><xsl:value-of
select='FORME'/></p>
</xsl:template>
</xsl:stylesheet>
```

Affichage : exemple 1



Appliquer des modèles

```
<xsl:apply-templates
  [select="pattern"] />
```

- applique les templates sur les fils du noeud courant, ou sur les noeuds spécifiés dans le `pattern` via le chemin Xpath

Document XML (2)

```
<ARTICLE>
  <FORME>guéri,ie</FORME>
  <GRAMMAIRE>
      <CATEGORIE>adj.</CATEGORIE>
   </GRAMMAIRE>
   <ACCEPTION>
    <DEFINITION>Rétabli d'un mal
physique.</DEFINITION>
 <EXEMPLE>Il a été très malade, mais le voila
guéri.</EXEMPLE>
   </ACCEPTION>
  <ACCEPTION>
 <EXEMPLE>J'en suis guéri de dépenser pour de telle
bêtise</EXEMPLE>
  </ACCEPTION>
</ARTICLE>
```

Appliquer un modèle : exemple 1

- Afficher uniquement les exemples du dictionnaire précédent

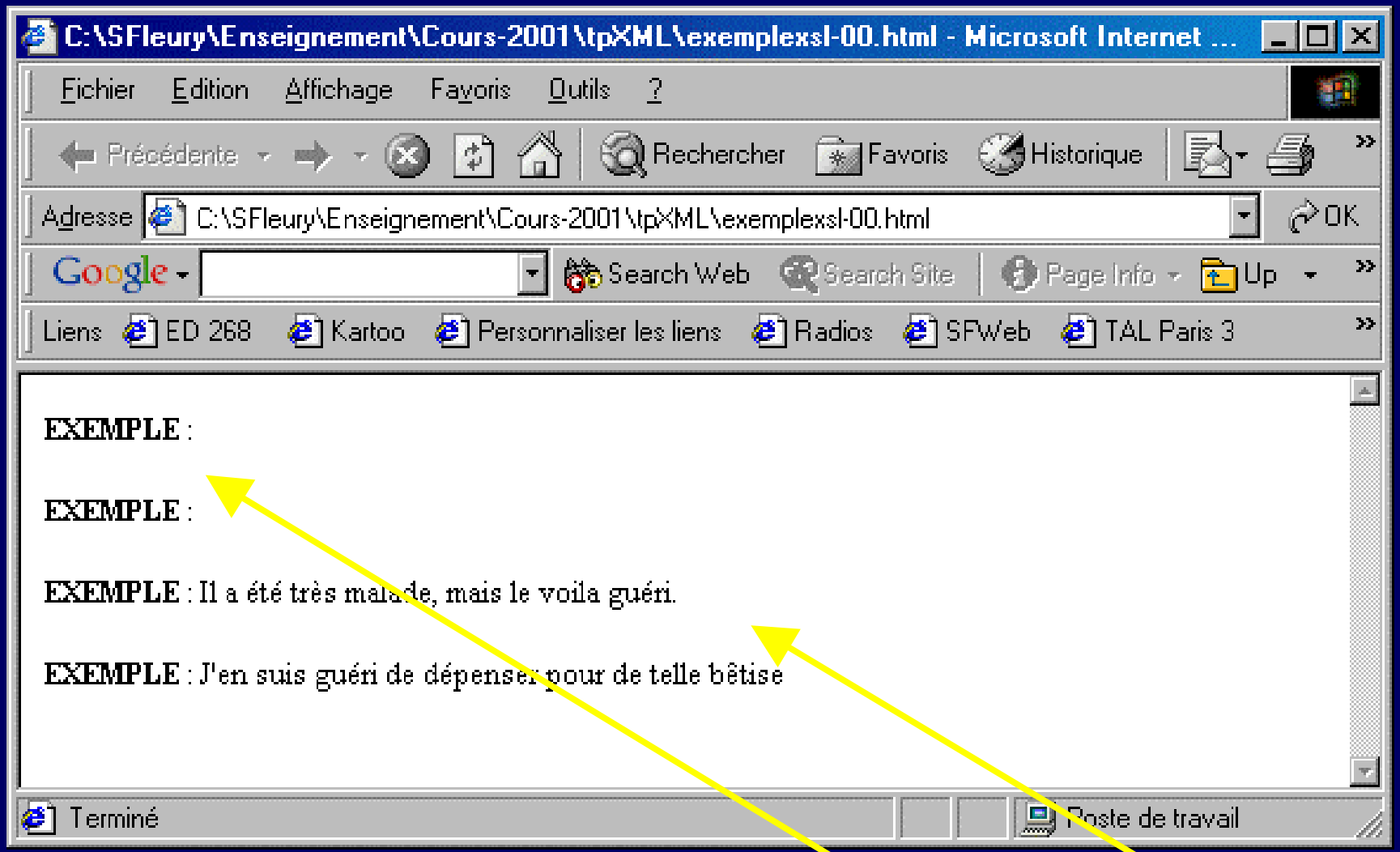
```
<xsl:template match=
  'ACCEPTION'>

<p><xsl:value-of
  select='EXEMPLE'/></
  p>

</xsl:template>
```

- modèle insuffisant, le résultat contient aussi les autres sous éléments (FORME, GRAMMAIRE...)

```
<xsl:template match=
  'ACCEPTION'>

<p><xsl:value-of
  select='EXEMPLE'/></
  p>

</xsl:template>

<xsl:template match=
  'ARTICLE'>

<xsl:apply-templates
  select='ACCEPTION'/>

</xsl:template>
```

Exemple complet

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<xsl:stylesheet version='1.0'
xmlns:xsl='http://www.w3.org/1999/XSL/Transform
'>
<xsl:template match="DICTIONNAIRE">
<html><body>
<xsl:apply-templates/>
</body></html>
</xsl:template>
<xsl:template match="ACCEPTION">
<p><b>EXEMPLE </b>: <xsl:value-of
select="EXEMPLE"/></p>
</xsl:template>
<xsl:template match="ARTICLE">
<xsl:apply-template select="ACCEPTION"/>
</xsl:template>
```

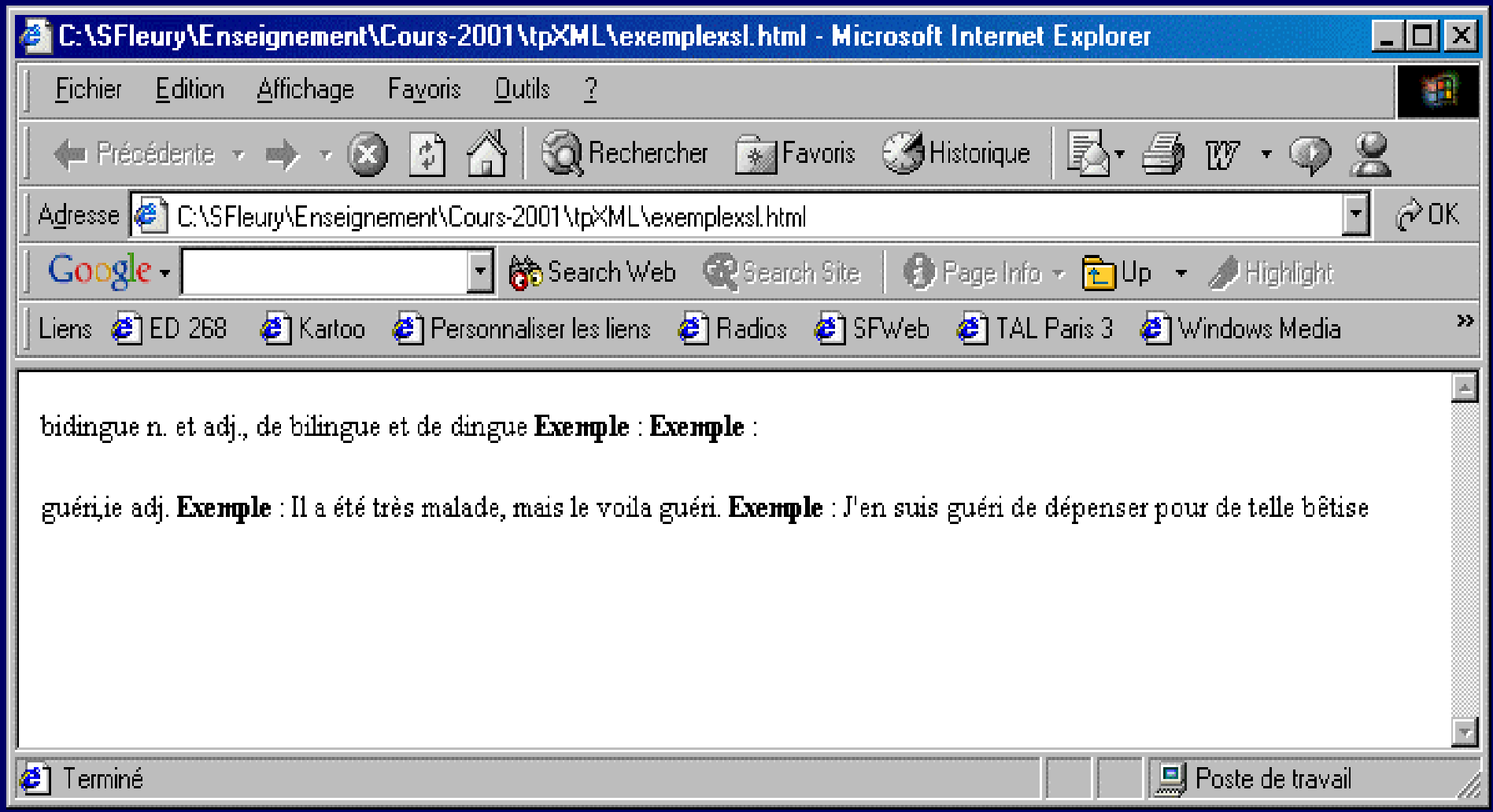


Les éléments EXEMPLE de bidingue ne sont pas renseignés, ceux de GUERI le sont

Conversion vers HTML (1)

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="DICTIONNAIRE">
<html><body>
<xsl:apply-templates/>
</body></html>
</xsl:template>
<xsl:template match="ARTICLE">
<p>
<xsl:apply-templates/>
</p>
</xsl:template>
<xsl:template match="ACCEPTION"> <b>Exemple</b> :
<xsl:value-of select='EXEMPLE'/>
</xsl:template>
</xsl:stylesheet>
```

Résultat (1) : traitements partiels



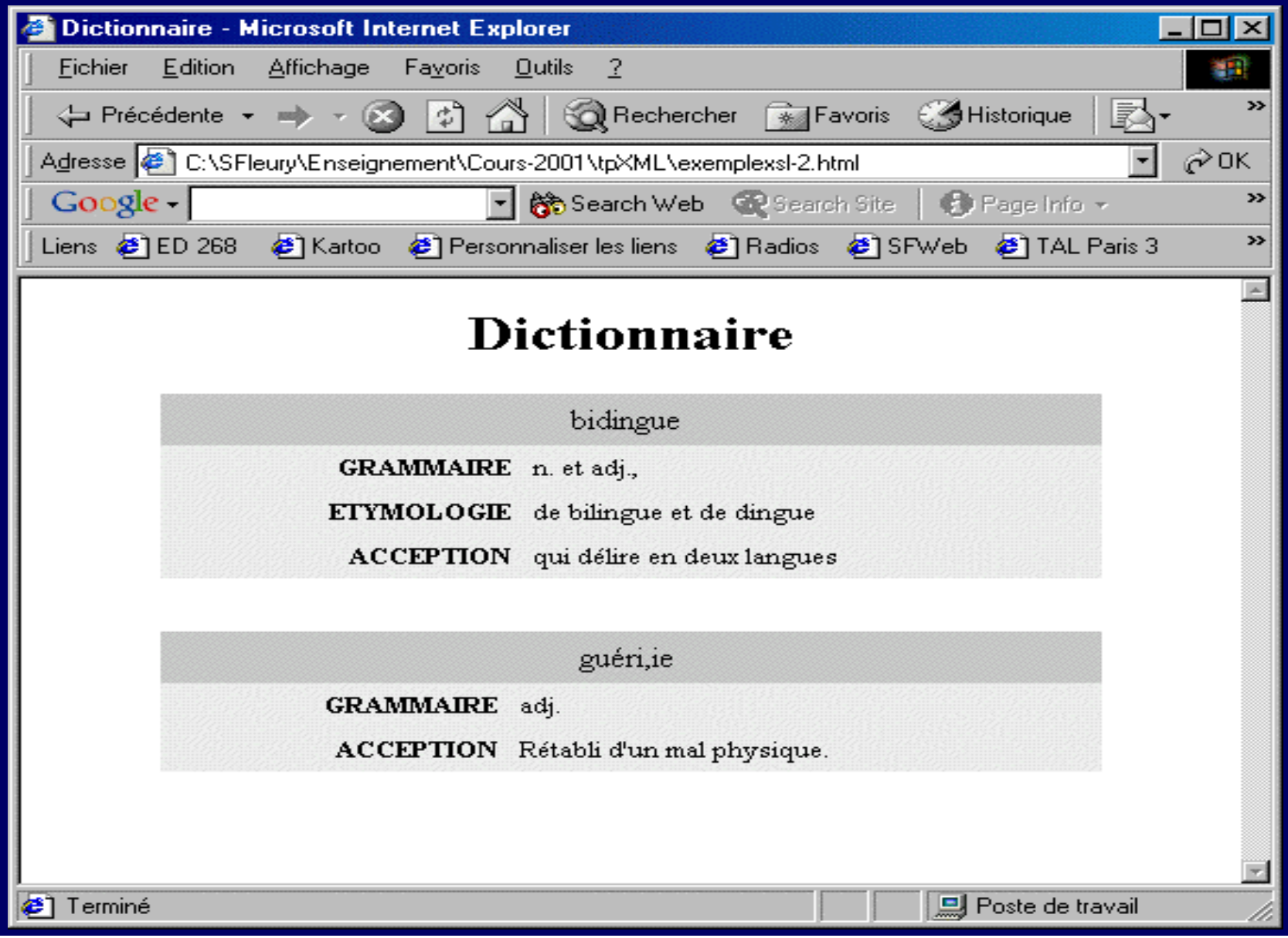
Conversion vers HTML (2)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet                                  version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="DICTIONNAIRE">
      <html>
        <head>
          <title>Dictionnaire</title>
        </head>
        <body      bgcolor="#FFFFFF"      link="#FF0000"
  vlink="#0000FF">
          <center><h1>Dictionnaire</h1></center>
        <xsl:apply-templates />
        </body>
      </html>
    </xsl:template>
```

```
<!-- on affiche un mot -->
    <xsl:template match="ARTICLE">
      <center><table border="0" width="80%" cellpadding="5"
  cellspacing="0">
          <tr bgcolor="#CCCCCC">
        <td colspan="3">
              <center>
                   <font size="+1">
              <xsl:value-of select="FORME"/>
                 </font>
              </center>
            </td>
          </tr>
   <xsl:if test="GRAMMAIRE">
        <tr bgcolor="#EEEEEE">
          <td align="right" valign="top"><b>GRAMMAIRE</b></td>
          <td align="left">
            <xsl:value-of select="GRAMMAIRE/CATEGORIE"/>
          </td>
        </tr>
  </xsl:if>
```

```
    <xsl:if test="ETYMOLOGIE">
        <tr bgcolor="#EEEEEE">
          <td align="right"
valign="top"><b>ETYMOLOGIE</b></td>
          <td align="left">
            <xsl:value-of select="ETYMOLOGIE"/>
          </td>
        </tr>
</xsl:if>
       <xsl:if test="ACCEPTION">
        <tr bgcolor="#EEEEEE">
          <td align="right" valign="top"><b>ACCEPTION</b></td>
          <td align="left">
            <xsl:value-of select="ACCEPTION/DEFINITION"/>
          </td>
        </tr>
</xsl:if>
       </table>
     </center>
     <br/><br/>
   </xsl:template>
   </xsl:stylesheet>
```

Résultat (2)


Dictionnaire - Microsoft Internet Explorer
Fichier Edition Affichage Favoris Outils ?
Précédente
Rechercher
Favoris
Historique
Adresse
C:\SFleury\Enseignement\Cours-2001\tpXML\exemplexsl-2.html
OK
Google
Search Web
Search Site
Page Info
Liens
ED 268
Kartoo
Personnaliser les liens
Radios
SFWeb
TAL Paris 3
Dictionnaire
bidingue
GRAMMAIRE n. et adj.,
ETYMOLOGIE de bilingue et de dingue
ACCEPTION qui délire en deux langues
guéri,ie
GRAMMAIRE adj.
ACCEPTION Rétabli d'un mal physique.
Terminé
Poste de travail

Exemple complet avec éléments XSLT : un répertoire

- Un répertoire : le document XML
- Un répertoire : la DTD
- Un répertoire : conversion vers HTML via XSL
- Raffinement : tri des entrées du dictionnaire et création de liens internes

un répertoire (doc XML)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE repertoire SYSTEM "repertoireSF.dtd">
<?xml-stylesheet type="text/xsl" href="repertoireSF-2.xsl"?>
    <repertoire>
     <personne>
      <prenom>Serge</prenom>
      <nom>Fleury</nom>
      <entreprise>Université Paris 3</entreprise>
      <email lieu="domicile">serge.fleury@univ-paris3.fr</email>
      <pageweb>www.cavi.univ-paris3.fr/ilpga/ilpga/sfleury/</pageweb>
      <tel lieu="domicile">06.19.16.11.04</tel>
      <notes>Enseignant chercheur à l'ILPGA</notes>
     </personne>
     <personne>
      <prenom>André</prenom>
      <nom>Salem</nom>
      <entreprise>Université Paris 3</entreprise>
      <email lieu="domicile">andre.salem@univ-paris3.fr</email>
      <pageweb>www.cavi.univ-paris3.fr/ilpga/ED/dr/asdr/</pageweb>
      <tel lieu="domicile">../../../..</tel>
      <notes>Professeur à l'ILPGA</notes>
     </personne>
    </repertoire>
```

un répertoire (DTD)

```
<!ENTITY % lieux "travail | domicile | vacances">
<!ELEMENT repertoire (personne+)>
<!ELEMENT personne (prenom, nom, entreprise,
email*, pageweb , tel+, adresse?, notes?)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT entreprise (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ATTLIST email lieu (%lieux;) #IMPLIED>
<!ELEMENT pageweb (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
<!ATTLIST tel lieu (%lieux;) #IMPLIED>
<!ELEMENT adresse (rue, cp, ville)>
<!ELEMENT rue (#PCDATA)>
<!ELEMENT cp (#PCDATA)>
<!ELEMENT ville (#PCDATA)>
<!ELEMENT notes (#PCDATA)>
```

un répertoire (XSL)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/repertoire">
      <html>
        <head>
          <title>Répertoire</title>
        </head>
        <body bgcolor="#FFFFFF" link="#FF0000"
vlink="#0000FF">
          <center><h1>Mon répertoire</h1></center>
    <xsl:apply-templates />
        </body>
      </html>
    </xsl:template>
```

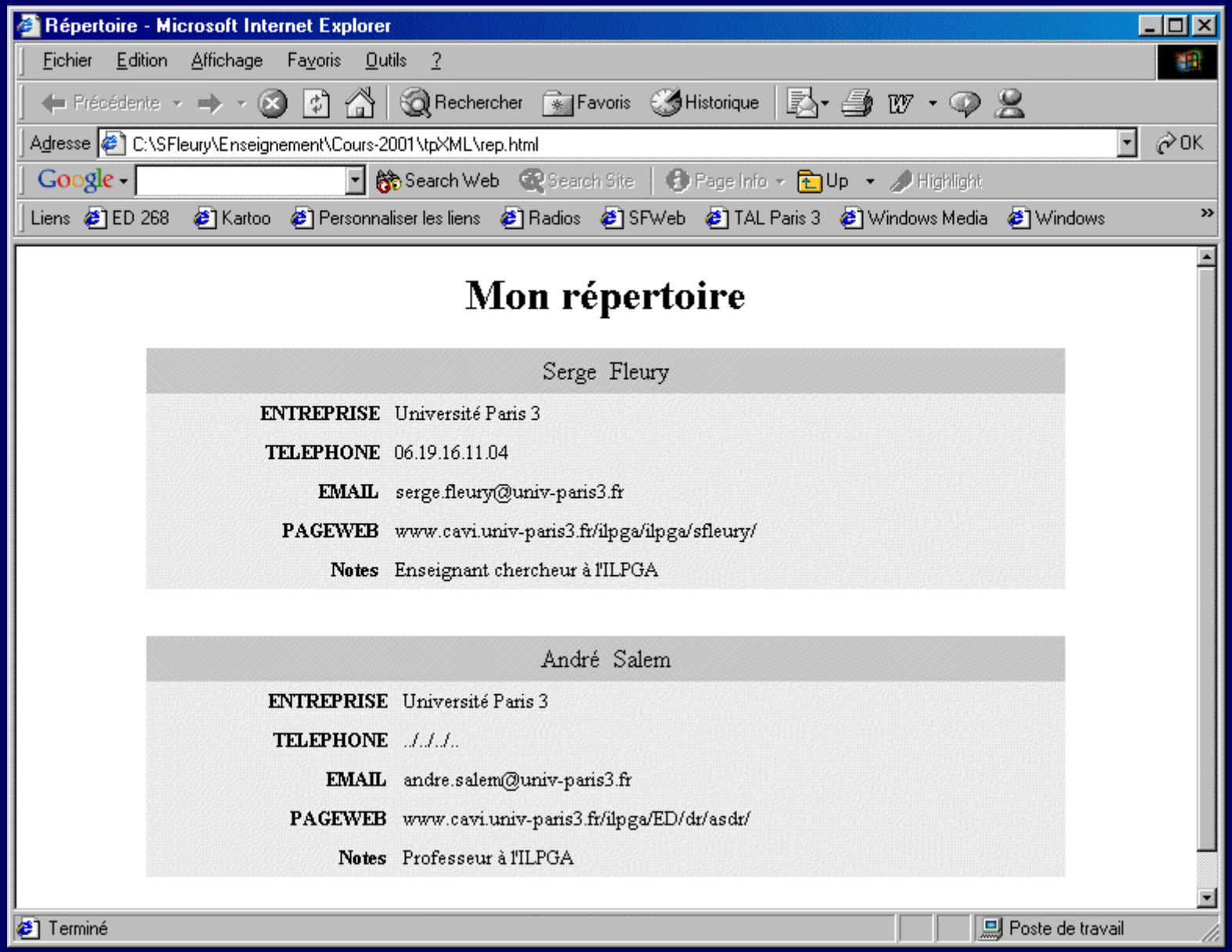
```
<!-- on affiche une fiche individuelle -->
    <xsl:template match="personne">
      <center>
        <table border="0" width="80%"
cellpadding="5" cellspacing="0">
          <!-- titre avec nom et prénom -->
          <tr bgcolor="#CCCCCC">
            <td colspan="2">
            <center>
            <font size="+1">
              <xsl:value-of
select="prenom"/>
              <xsl:value-of select="nom"/>
            </font>
            </center>
            </td>
          </tr>
```

```
    <xsl:if test="entreprise">
          <tr bgcolor="#EEEEEE">
            <td align="right"
valign="top"><b>ENTREPRISE</b></td>
            <td align="left">
              <xsl:value-of select="entreprise"/>
            </td>
          </tr>
     </xsl:if>

 <xsl:if test="tel">
          <tr bgcolor="#EEEEEE">
            <td align="right"
valign="top"><b>TELEPHONE</b></td>
            <td align="left">
              <xsl:value-of select="tel"/>
            </td>
          </tr>
     </xsl:if>
```

```
    <xsl:if test="email">
          <tr bgcolor="#EEEEEE">
            <td align="right"
valign="top"><b>EMAIL</b></td>
            <td align="left">
              <xsl:value-of select="email"/>
            </td>
          </tr>
     </xsl:if>

   <xsl:if test="pageweb">
          <tr bgcolor="#EEEEEE">
            <td align="right"
valign="top"><b>PAGEWEB</b></td>
            <td align="left">
              <xsl:value-of select="pageweb"/>
            </td>
          </tr>
     </xsl:if>
```

```
  <xsl:if test="notes">
        <tr bgcolor="#EEEEEE">
          <td align="right"
valign="top"><b>Notes</b></td>
          <td align="left">
            <xsl:value-of select="notes"/>
          </td>
        </tr>
  </xsl:if>

      </table>
      </center>
      <br/><br/>
    </xsl:template>
    </xsl:stylesheet>
```

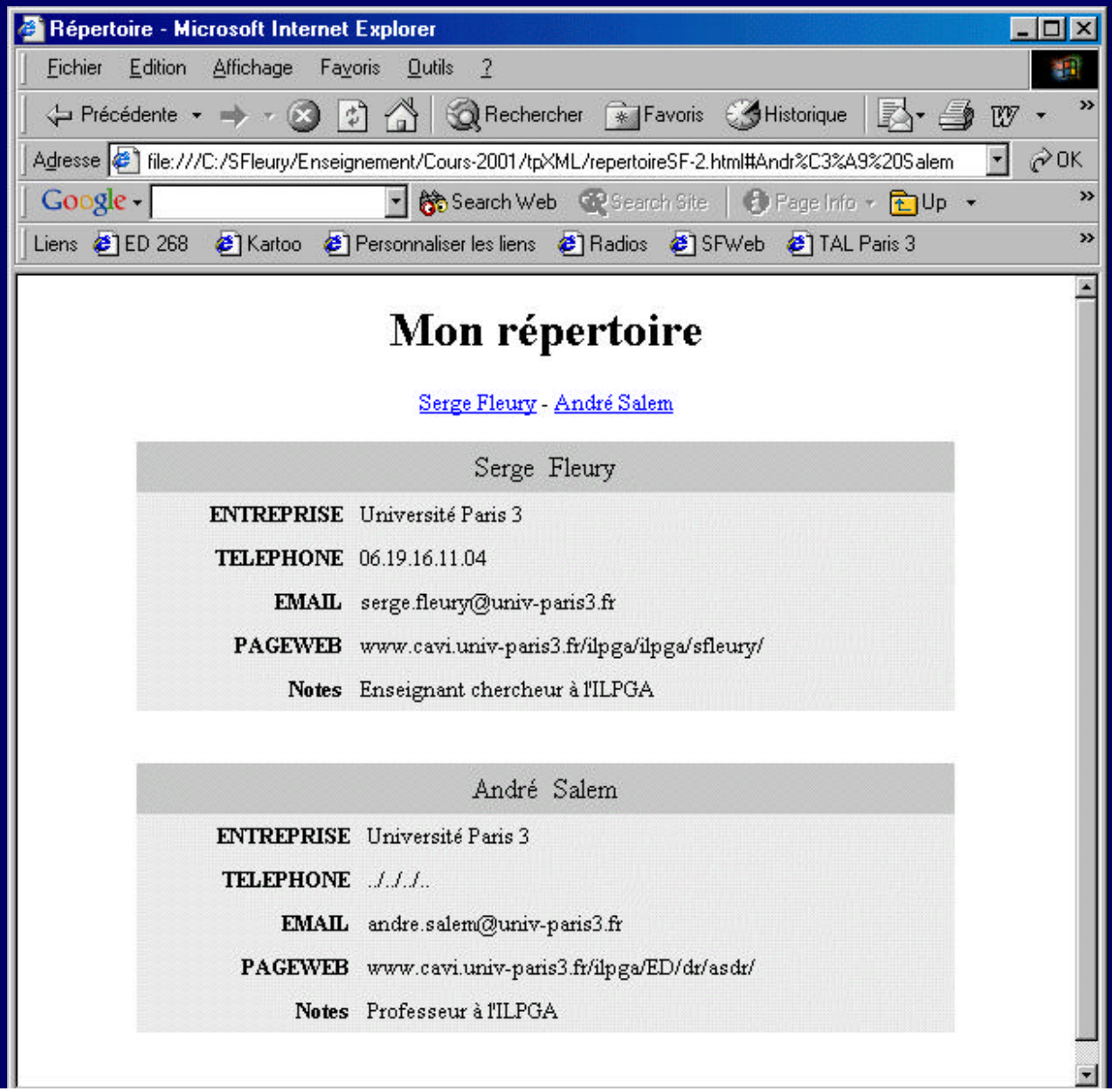
Affichage du répertoire (1)



Raffinement (tri et liens)

```
<xsl:template match="/repertoire">
      <html>
        <head>
          <title>Répertoire</title>
        </head>
        <body bgcolor="#FFFFFF" link="#FF0000" vlink="#0000FF">
   <center><h1>Mon répertoire</h1></center>
         <xsl:call-template name="liste"/>
          <xsl:for-each select="personne">
     <xsl:sort select="nom"/>
     <xsl:apply-templates select="."/>
          </xsl:for-each>
        </body>
      </html>
    </xsl:template>
<!-- on affiche la liste des personnes -->
    <xsl:template name="liste">
      <center>
        <xsl:for-each select="personne">
          <xsl:sort select="nom"/><a href="#{prenom} {nom}">
     <xsl:value-of select="prenom"/><xsl:value-of select="nom"/>
          </a> <xsl:if test="not(position()=last())">
                <xsl:text> - </xsl:text>
              </xsl:if>
        </xsl:for-each>
      </center><br/>
    </xsl:template>
```

Affichage répertoire (2)



Eléments du langage XSL

- Description de quelques éléments
- Le reste est à voir dans la documentation officielle de XSLT
- Voir aussi en ligne, ici par exemple :
  - http://www.laltruiste.com/document.php?compteur=1&page=1&rep=5

xsl:text

- `xsl:text` génère le texte spécifié sans enlever des "blancs" et sur option, remplace les séquences d'échappement &...; par leurs caractères

```
<xsl:text [disable-output-
  escaping="yes|no"]> texte
  </xsl:text>
```

- génère intégralement le texte spécifié en sortie, en particulier les "blancs" ; éventuellement, remplace les séquences d'échapement &...; par leurs caractères équivalents

xsl:for-each

- L'élément <*xsl:for-each*> permet d'appliquer des règles de style sur chaque noeud identique d'un *template*. Les noeuds sont identifiés par un *pattern* spécifié par un attribut *select* d'ailleurs obligatoire (exemple ou supra)

```
<xsl:for-each select="pattern">
transformations </xsl:for-each>
```

  - sélectionne tous les noeuds correspondants au pattern, puis pour chacun, en fait le noeud courant auquel sont appliquées les transformations spécifiées
  - Evidemment, cet élément ne peut être employé que sur une arborescence dont la structure soit uniforme et connue tel que par exemple un document XML formé à partir de titres et de paragraphes (exemple ou supra)

xsl:copy

<xsl:copy> transformations </xsl:copy>

- copie le noeud courant dans le document final, mais pas ses attributs, ni ses descendants; ces dernières sont à charge des transformations (exemple).

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

<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="* | @* | comment() | text()">

<xsl:copy>

<xsl:apply-templates select="* | text()"/>

</xsl:copy>

</xsl:template> </xsl:stylesheet>
```

xsl:if

- L'élément <*xsl:if*> permet d'appliquer un test conditionnel dans la structure d'une feuille de style XSL.

```
<xsl:if test="pattern">
  transformations </xsl:if>
```

- les transformations ne sont appliquées au noeud courant que si l'évaluation du pattern (prédicat XPath) donne la valeur vrai (exemple ou supra)

xsl:choose

L'élément <*xsl:choose*> combiné avec <*xsl:when*> et <*xsl:otherwise*>, permet de construire des tests conditionnels (exemple)

```
<xsl:choose> <xsl:when test="condition">
instructions... </xsl:when> ... <xsl:otherwise>
instructions... </xsl:otherwise> </xsl:choose>
```

Les transformations `xsl:when` ou `xsl:otherwise` s'applique au même noeud courant. Dès qu'une transformation `xsl:when` s'applique, sinon et éventuellement la transformation `xsl:otherwise`, la transformation `xsl:choose` se termine

xsl:sort

```
<xsl:sort [select="pattern"]
[lang="langue"] [data-type="text|number"]
[order="ascending|descending"]/>
```

- `xsl:sort` doit figurer juste après `xsl:apply-templates` ou `xsl:for-each` ou `xsl:sort`
- L'élément <*xsl:sort*> doit être contenu dans une règle de modèle <*xsl:template*> qui pointe un des éléments parents de la clé de tri sélectionné par <*xsl:sort select="clé_de_tri"*>. Le noeud pointé par cette clé sera traîté dans un autre *template* qui appliquera le tri. Peuvent être précisés : la langue, ordre lexicographique (par défaut) ou numérique, ordre ascendant (par défaut) ou descendant (exemple ou supra)

xsl:call-template

```
<xsl:call-template name="nom"/>
```

```
<xsl:call-template name="nom"> passage de valeurs aux paramètres </xsl:call-template>
```

- applique le template de même nom sur le noeud courant, avec éventuellement des valeurs passées aux paramètres s'il y en a. L'élément <*xsl:call-template*> ne peut contenir que l'élément <*xsl:with-param*> permettant de faire passer des paramètres à un *template*. (exemple ou supra)

xsl:with-param

```
<xsl:with-param name="nom"/>
valeur passée au paramètre
</xsl:with-param>
```

- définit la valeur passée à un paramètre d'un template lors d'un appel à ce template par l'instruction call-template. `xsl:with-param` doit figurer juste après `xsl:call-template` ou `xsl:with-param`.

xsl:param

```
<xsl:param name="nom"/>
<xsl:param name="nom"> valeur par défaut du
  paramètre </xsl:param>
```

- définit un paramètre d'un template avec éventuellement une valeur par défaut. `xsl:param` doit figurer juste après `xsl:template` ou `xsl:param` (exemple)

xsl:element

```
<xsl:element
name="nom_élément"
namespace="adresse_URI"
use-attribute-sets="nom_jeu_attributs"> …
</xsl:element>
```

- crée un élément du nom spécifié dans le document final (exemple)

xsl:attribute

```
<xsl:attribute name="nom">
valeur </xsl:attribute>
```

- crée et ajoute un attribut du nom et de la valeur spécifiés à l'élément courant (exemple)

xsl:number

```
<xsl:number [count="pattern"]
[level="single|multiple|any"]
[format="chaine"]/>
```

- insère en sortie un numéro incrémenté :
  - Par défaut, sur les noeuds courants, sinon ceux sélectionnés par le pattern. Avec une numérotation sur un seul (single) niveau de l'arborescence du document initial, ou sur plusieurs (multiple) niveaux, ou sans tenir compte (any) des niveaux. Les formats et valeurs initiales de numérotation sont données par la chaîne (exemple)

Modèles implicites

- 7 types de noeuds dans un document XML :
  - Le noeud racine
  - Les noeuds d'éléments
  - Les noeuds d'attribut
  - Les noeuds de texte
  - Les noeuds de commentaires
  - Les noeuds d'instructions de traitement
  - Les noeuds d'espace de noms

Modèle implicite (1)

- Pour les noeuds texte et les attributs
  - `<xsl:template match='text()|@*'><xsl:value-of select='.'></xsl:template>`
  - Copie la valeur des noeuds de texte et d'attribut dans le document en sortie
  - La fonction `text()` est une expression Xpath correspondant à tous les noeuds de texte
  - `@*` correspond à tous les noeuds d'attribut (c'est la valeur de l'attribut qui sera copié)

Document XML (3)

```
<?xml version="1.0" encoding="ISO-8859-1"
  standalone="yes"?>
<grammar>
<rule number="1">
<lhs>S</lhs>
<rhs>GN (GV)</rhs>
<features>
<item number='first'><featlhs>GV arg0 tete
  pers</featlhs><featrhs>GN tete pers</featrhs></item>
<item number='second'><featlhs>GV arg0 tete
  nbre</featlhs><featrhs>GN tete nbre</featrhs></item>
</features>
</rule>
<rule number="2">
...
</rule>
```

Modèle implicite : exemple 1

```
<xsl:template match='grammar'>
<html><body>
<dl><xsl:apply-template/>
</dl>
<xsl:template match='rule'><xsl:apply-
  template/></xsl:template>
<xsl:template match='features'>
<ul><li>Trait 1 : <xsl:apply-templates
  select='@first'</li></ul>
</xsl:template>
</xsl:template>
```

Modèle implicite (2)

- Pour les éléments et le noeud racine
  - `<xsl:template match='*|/'><xsl:apply-template></xsl:template>`
  - Applique les règles à tous noeuds (sauf attribut et espace de noms)
- Pour les commentaires et instructions de traitement
  - `<xsl:template match='processing-instruction()|comment()'/>`
  - Ne produit rien dans l'arbre de sortie

Modes

- Le même contenu en entrée peut apparaître plusieurs fois dans la sortie formaté suivant un modèle différent
  - Titres d'un chapitre formatés d'une certaine manière pour une table des matières et d'une autre pour les chapitres eux mêmes
  - Les éléments xsl:template et xsl:apply-templates peuvent avoir un attribut mode qui associe différents modèles à différents usages
  - Cf [XML 01] : exemple complet

XML en le pratiquant (5)

- Cf TP XSLT
  - Manipulation de documents XML et XSLT

Partie 8

- XPATH

Retour Sommaire

Lecture Poly

- Leçon n°8

XPATH

- XPath est un standard du W3C pour décrire des localisations de noeud et extraire des valeurs de l'arbre du document XML. Il sert 2 autres "standards" XML :
  - XSLT : ensemble de règles de transformation d'un document XML vers un autre document.
  - XPointer : mécanisme de pointage pour les liens XLink de XML
  - Les résultats des expressions XPath sont des noeuds (simple ou des ensembles), un booléan, un nombre, ou une chaine de caractères en Unicode

Document XML (4)

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<classification_atomique>
<famille type="gaz rare">
<atome>
<nom>hélium</nom>
<symbole>He</symbole>
<numero>2</numero>
<masse>4</masse>
</atome>
<atome>
<nom>néon</nom>
<symbole>Ne</symbole>
<numero>10</numero>
<masse>20</masse>
</atome>
</famille>
<famille type="métal alcalin">
<atome>
....
```

Chemins (1)

- En fait, une grande partie de XPath décrit des chemins dans une arborescence.
- Supposons que le noeud courant soit `<famille type="gaz rare">`
  - `/` sélectionne l'élément racine du document, donc <classification_atomique>
  - `/classification_atomique/famille` sélectionne les éléments famille fils de l'élément racine
  - `atome` sélectionne les éléments atome fils du noeud courant
  - `./atome` comme atome

Chemins (2)

- `atome/nom` sélectionne les éléments nom fils d'un élément atome fils du noeud courant
- `..` sélectionne l'élément père, donc <classification_atomique>
- `../famille` sélectionne les éléments famille fils du père du noeud courant
- `//atome` sélectionne les éléments atome fils du fils de l'élément racine
- `.//nom` sélectionne les éléments nom fils d'un élément fils du noeud courant
- `comment()` tous les noeuds commentaire du noeud courant
- `processing-instruction()` les instructions de traitement

Chemins (3)

- `./atome | .//nom` union des 2 sélections
- `*` sélectionne tous les éléments fils du noeud courant
- `atome/*` sélectionne tous les éléments fils d'un élément atome fils du noeud courant
- `@type` sélectionne l'attribut type du noeud courant
- `/classification_atomique/famille/@type` sélectionne l'attribut type des éléments famille fils de l'élément racine
- `@*` sélectionne tous les attributs du noeud courant
- `atome/node()` sélectionne tous les noeuds (pas uniquement les noeuds éléments) fils d'un élément atome fils du noeud courant

Axes (1)

- Les axes décrivent les directions des chemins dans une arborescence : racine, parent, noeud courant, descendant, ... les axes s'utilisent ainsi :
  - `axe :: chemin`
- Supposons que le noeud courant soit `<famille type="gaz rare">`
  - `Descendant :: atome` sélectionne les noeuds fils de l'élément atome fils du noeud courant
  - `following-sibling :: .` sélectionne les noeuds frères du noeud courant se trouvant après ce noeud courant

Axes (2)

- `self` désigne le noeud courant
- `child` les noeuds fils
- `ancestor` tous les noeuds de la branche du noeud courant
- `preceding-sibling`
- `following :: parent :: .` tous les noeuds suivants le noeud parent du noeud courant
- `preceding` tous les noeuds précédents le noeud courant dans le document
- `attribute` les attributs du noeud
- `namespace` les noeuds de type namespace

Prédicat (1)

- Les prédicats sont utilisés dans les tests de XSLT.
- L'ensemble des noeuds obtenus par un chemin XPath peut être filtré à l'aide de prédicats entre crochets []
- Supposons que le noeud courant soit `<famille type="gaz rare">`
  - `./atome[3]` sélectionne le 3ème élément atome fils du noeud courant
  - `./atome[position()=3]` sélectionne l'élément atome fils du noeud courant dont le position est en 3ème place.

Prédicat (2)

- `atome[last()]` sélectionne le dernier élément atome fils du noeud courant
- `atome[numero!=4]` sélectionne les éléments atome fils du noeud courant dont le numero est différent de 4
- `atome[numero >= 4] ...` dont le numéro est >= 4. Il faut utiliser < (respectivement >) à la place de < (resp. >) dans les documents XML;
  - exemple : `<xsl:if test="numero < 10">...</xsl:if>`
- `following-sibling :: .[@type="non métal"]` sélectionne les noeuds famille frère (suivant dans le document) du noeud courant dont l'attribut type vaut "non métal"

Prédicat (3)

- `atome[masse mod 2 =1]` sélectionne les éléments atome fils du noeud courant dont la masse est impair.
- `../famille[(@type="non métal") and (atome/masse > 34)]` sélectionne les noeuds famille fils du père du noeud courant dont l'attribut type vaut "non métal" et dont l'un des fils atome a un fils masse dont le contenu est > à 34.
- `../famille[@type="non métal"]/atome[masse > 34]` sélectionne les noeuds atome, ayant un fils masse dont le contenu est > à 34, et qui sont fils de noeud famille fils du noeud racine dont l'attribut type vaut "non métal".

Prédicat (4)

- `//famille/Atome[3] [masse > 34]` sélectionne parmi les 3ème éléments atome des fils famille du noeud racine ceux dont la masse est > à 34
- `//famille/atome[masse > 34] [3]` sélectionne le 3ème élément parmi les atomes, dont la masse est > à 34, des fils famille du noeud racine
- `//famille[count(atome[masse > 34]) != 0]` sélectionne les noeuds famille du noeud racine ayant au moins un fils atome de masse > à 34
- `//famille[not(contains(@type, 'gaz'))]` sélectionne les noeuds famille du noeud racine ayant un attribut type qui ne comporte pas la chaine 'gaz'
- `//famille[@type='gaz rare']/atome[./nom[contains(text(),'h']]...`

Prédicat (5)

- opérateurs : `and, or, *, +, -, /, mod`
- fonctions numériques : `number, sum, floor, ceiling, round`
- fonctions booléennes : `not, lang`
- fonctions sur les noeuds :
  - `id("chaine")` le noeud dont l'identifiant unique est chaine
  - `name(noeud)` renvoi le nom du noeud avec namespace s'il y a
  - `local-name(noeud)` renvoi le nom du noeud sans préfixe (namespace)

Prédicat (6)

- fonctions chaînes :
  - `string(objet)` convertit l'objet en chaine
  - `contains(chaine, sous-chaine)` renvoie un boolean
  - `string-length(chaine)` renvoie un nombre
  - `substring`
  - ...

Des exemples « complexes »...

1. Recherche des phrases du narrateur précédant une phrase du Petit Prince (en français) contenant le mot "un"

```
//phrase[@loc='LePetitPrince'][./modalite/traduction[@lang='francais']/mot[
   contains(text(),'un')]]/following-sibling::phrase[@loc='narrateur']
```

2. Recherche des phrases du Petit Prince qui contiennent un "?"

```
//phrase[@loc='LePetitPrince'][./modalite/traduction[@lang='francais']/mot[
   contains(text(),'?')]]
```

et première réponse en français du narrateur

```
//phrase[@loc='LePetitPrince'][./modalite/traduction[@lang='francais']/mot[
   contains(text(),'?')]]/following-sibling::phrase[@loc='narrateur'][1]
```

ou plus précisément :

```
//phrase[@loc='LePetitPrince'][./modalite/traduction[@lang='francais']/mot[
   contains(text(),'?')]]/following-
   sibling::phrase[@loc='narrateur'][1][.modalite/traduction[@lang='francai
   s']]
```

3. Phrases précédant celles du Petit Prince contenant un "?"

```
//phrase[@loc='LePetitPrince'][./modalite/traduction[@lang='francais']/mot[
   contains(text(),'?')]]/preceding-sibling::phrase
```

XML en le pratiquant (6)

- Cf TP Xpath

Partie 9

- XLINK

Retour Sommaire

XLINK

- Xlink est une syntaxe basée sur les attributs pour associer des liens aux documents XML
  - Liens simples, bi-directionnels, multi-directionnels...
- XLink <=> syntaxe XML pour décrire des graphes
  - Les sommets sont des documents
  - Les arêtes sont les liens entre ces documents

Liens simples

- Tout élément XML peut utiliser les attributs `xlink:type`, `xlink:href`, `xlink:show`, `xlink:actuate` et `xlink:title` pour que le contenu de l'élément porte un lien vers un autre point du cyber-espace
  - `xlink:type` => valeur=simple
  - `xlink:href` => valeur=une URL
  - `xlink:title` => valeur=une chaîne de caractères
    - le texte fourni pourra servir d'indice supplémentaire, par exemple en étant affiché quand la souris se trouve sur l'élément contenant le lien

xlink:show

- `xlink:show`
  - => valeur=`new`
    - le document destination apparaîtra dans une nouvelle fenêtre
  - => valeur=`replace`
    - le document destination apparaîtra dans la même fenêtre que le document courant (comportement similaire aux liens HTML)

xlink:show (2)

- => valeur=`embed`
  - le document destination apparaîtra dans le document courant, à la place de l'élément constituant le lien (comportement similaire aux balises HTML <img> et <object>)
- => valeur=`none`
  - l'application traîtant le document choisira le traitement approprié

xlink:actuate

- `xlink:actuate`
  - => valeur=`onLoad`
    - le document destination apparaîtra dès le chargement du document courant (comportement similaire aux balises HTML <img> et <object>)
  - => valeur=`onRequest`
    - le document destination apparaîtra quand l'utilisateur activera le lien (généralement en "cliquant" dessus)

xlink:actuate (2)

- => valeur=`other`
  - Défini par un autre balisage dans le document
- => valeur=`none`
  - l'application traîtant le document choisira le moment approprié

Xlink : exemple 1

```
<article xmlns:xlink="http://www.w3.org/1999/xlink" >
 <titre>Un journaliste accuse, un policier
 dément</titre>
   <auteur xlink:href="AlainHome.xml"
 xlink:type="simple" xlink:show="replace"
 xlink:actuate="OnRequest"
 xlink:title="Homepage">Alain Connu</auteur>
   ...
   <texte>
     <image>
       <html:img src="photo.gif"
 xlink:href="Details.xml" xlink:type="simple"
 xlink:show="replace" xlink:actuate="OnRequest"
 xlink:title="details" />
     </image>
   </texte>
 </article>
```

Liens étendus

- associent un nombre arbitraire de ressources
- la définition d'un lien étendu peut se faire dans un fichier séparé des ressources qu'il associe
- dans un lien étendu, la désignation de l'emplacement des ressources est séparé du mécanisme de "traversée"
- Un lien étendu est un graphe étiqueté, orienté, dans lequel les chemins sont des arcs, les documents des noeuds et les étiquettes des URLs

- un lien étendu est composé des éléments suivants:
  - localisation des ressources à distance (éléments de type `locator`)
  - mécanisme de traversée (éléments de type arc)
  - étiquetage pour faciliter l'utilisation du lien par une personne (éléments de type title)
  - ressources locales associées par le lien (éléments de type resource)

- Un lien étendu est défini sur un élément XML auquel on rajoute l'attribut :
  - `xlink:type="extended"`
    - en supposant que "xlink" est le préfixe choisi pour l'espace de nom

```
<novel xlink type='extended'>
    <title>The wonderful Wizard of Oz</title>
    <author>L. Franck Baum</author>
    <year>1900</year>
</novel>
```

    - d'autres attributs sur l'élément en question ou sur des sous-éléments imbriqués définiront l'ensemble du lien

Localisateurs

- Un élément localisateur a un attribut `xlink:type` avec la valeur `locator` et un attribut `xlink:href` contenant l'URI de la ressource qu'il localise

```
<novel xlink type='extended'>
       <title>The wonderful Wizard of Oz</title>
       <author>L. Franck Baum</author>
       <year>1900</year>
       <edition xlink:type='locator'
         xlink:href='urn:isbn:0688069444' />
       <edition xlink:type='locator'
         xlink:href='urn:isbn:0192839306' /> ...
   <novel>
```

Identification d'un localisateur

- Chaque localisateur a aussi un attribut xlink:label qui sert d'identifiant à l'élément
  - Cet attribut sera utilisé pour mettre en place les liens entre les ressources

```
<edition xlink:type='locator'
  xlink:href='urn:isbn:0688069444'xlin
  k:label='ISBN0688069444' />
```

Attributs optionnels supplémentaires

- Les localisateurs peuvent avoir des attributs supplémentaires pour décrire plus précisément le lien visé
  - `xlink:title`
    - valeur=une chaîne de caractères
    - donne un commentaire sur le lien étendu dans son ensemble
  - `xlink:role`
    - valeur=URI
    - adresse d'un document décrivant le rôle du lien étendu (le lien étendu, comme n'importe quel autre élément d'un document XML, peut être considéré comme une ressource composant un autre lien étendu)

Arcs

- Les chemins entre les ressources sont appelés des arcs
- Ils sont représentés par des éléments arcs avec pour attributs
  - `xlink:from` => identifie la source du lien
  - `xlink:to`=> identifie la cible du lien
  - Ces attributs portent un nom correspondant à la valeur de l'attribut `xlink:label` d'un des éléments localisateurs du lien étendu

```
<famille
xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="extended" xlink:title="ma famille" >
   <personne xlink:type="locator"
xlink:label="pere"
xlink:href="http://pageperso/lepere/" />
   <personne xlink:type="locator"
xlink:label="mere"
xlink:href="http://pageperso/lamere/" />
   <personne xlink:type="locator"
xlink:label="enfant"
xlink:href="http://pageperso/lenfant/" />
   <pereversenfant xlink:type="arc"
xlink:from="pere" xlink:to="enfant"/>
   <mereversenfant xlink:type="arc"
xlink:from="mere" xlink:to="enfant"/>
<enfantverspere xlink:type="arc"
xlink:from="enfant" xlink:to="pere"/>
 </famille>
```

Arcs multiples

- Si plusieurs éléments partagent la même étiquette, un élément utilisant cette étiquette dans l'attribut `xlink:to` ou `xlink:from` définit des arcs entre toutes les ressources de même étiquette

```
<famille
xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="extended" xlink:title="ma famille"
>
   <personne xlink:type="locator"
xlink:label="parent"
xlink:href="http://pageperso/lepere/" />
   <personne xlink:type="locator"
xlink:label="parent"
xlink:href="http://pageperso/lamere/" />
   <personne xlink:type="locator"
xlink:label="enfant"
xlink:href="http://pageperso/lenfant/" />
<enfantversparent xlink:type="arc"
xlink:from="enfant" xlink:to="parent"/>
 </famille>
```

Attributs supplémentaires

- Chaque élement arc peut avoir un attribut xlink:title
  - Contenant une chaîne de caractères décrivant l'arc
- Ils peuvent aussi avoir un attribut `xlink:arcrole`
  - Contenant une URI pointant vers la description de l'arc

Ressources locales

- Les localisateurs décrivent des ressources distantes
- Les liens étendus peuvent aussi décrire des ressources locales dans lesquelles l'élément lien étendu contient des données

```
<famille
xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="extended" xlink:title="ma famille" >
   <personne xlink:type="resource"
xlink:label="pere">
<nom>...</nom><prenom>...</prenom>...</personne>
   <personne xlink:type="resource"
xlink:label="mere">
<nom>...</nom><prenom>...</prenom>...</personne>
   <personne xlink:type="resource"
xlink:label="enfant">
<nom>...</nom><prenom>...</prenom>...</personne>
   <pereversenfant xlink:type="arc"
xlink:from="pere" xlink:to="enfant"/>
   <mereversenfant xlink:type="arc"
xlink:from="mere" xlink:to="enfant"/>
<enfantverspere xlink:type="arc"
xlink:from="enfant" xlink:to="pere"/>
 </famille>
```

Partie 10

- XPOINTER

Retour Sommaire

Xpointer

- Xpointer est une syntaxe nom-XML utilisée pour localiser des points ou des régions de documents XML
- Un Xpointer est attaché à la fin d'un URI comme identifiant de fragment pour indiquer une partie spécifique d'un document XML
- La syntaxe de Xpointer est basée sur celle de Xpath

Xpointer dans les liens

- Premier élément name du document http://www.ibiblio.org/xml/people.xml :
  - http://www.ibiblio.org/xml/people.xml#xpointer(//name[position()=1])
- Puisque Xpath localise les noeuds d'un document XML, Xpointer ne peut pointer que sur des documents XML
- Xpointer est le plus souvent utilisé avec Xlink

Noms nus

- Un nom nu identifie l'élément qu'il désigne par son nom
- Similaire à une ancre nommée HTML
- un identificateur est associé à un élément à l'aide de l'attribut de type ID, on peut y faire référence à l'aide de la notation `xpointer(id("UnIdentificateur"))`, qui peut être abrégée `UnIdentificateur`

Séquence d'enfants

```
xpointer(/child::*[position
  ()=1]/child::*[position()=
  2/child::*[position()=3])
```

- Sélectionne le 3ème sous-élément du second enfant de l'élément racine
- Abréviation possible : /1/2/3, celle-ci porte le nom de « séquence d'enfants »

Points

- Pour pointer sur autre chose que les noeuds
  - Exemple : le 3ème mot du second paragraphe
- Un point est identifié par son conteneur et un index positif ou nul dans ce noeud
- On utilise la notation vue au point précédent pour repérer un élément, suivie d'un nombre entier positif ou nul entre [], pour repérer un point précis au sein de l'élément en question.

- Si l'élément désigné a un contenu simple, la valeur doit être comprise dans l'intervalle 0..n (où n est le nombre de caractères du contenu de l'élément désigné); 0 indique le point se trouvant juste avant le premier caractère; une valeur positive i indique le point se trouvant juste après le i-ème caractère.

- `xpointer(//title[position()=1]/text()/point()[position()=3])`
  - Sélectionne tout d'abord le premier élément title du document, il reprend ensuite so noeud de texte fils, dans ce dernier il sélectionne le point entre le 3ème et le 4ème caractère

- Si l'élément a des sous-éléments, le nombre doit être compris entre 0 et n (où n est le nombre de sous-éléments). 0 indique le point se trouvant juste avant le premier élément; une valeur positive i indique le point se trouvant juste après le i-ème sous-élément

- `xpointer(/novel/point()[position()=1])`
  - Identifie le point précédent la balise title dans le document suivant

```
<novel>
    <title>...</title>
    <author>...</author>
</novel>
```

- `xpointer(start-point(//title))`
  - Identifie le même point que précédemment
- `xpointer(end-point(//author))`
  - Identifie le point après la balise </author>

Régions

- La fonction `range()` prend en argument une expression Xpath retournant un ensemble de lieux
- Pour chaque noeud de cet ensemble, elle retourne une région qui couvre exactement ce noeud
  - `xpointer (range(//title))` sélectionne l'unique élément title
  - `xpointer(range(/novel/*))` sélectionne 2 régions, une pour chacun des sous-éléments de l'élément racine novel

Range-inside

- La fonction `range-inside` prend en argument une expression Xpath retournant un ensemble de lieux
  - Pour chaque noeud, elle retourne une région couvrant exactement le contenu du noeud
    - Pour un noeud d'élément, cette région inclut le contenu de l'élément mais exclut les balises
    - Pour tout sauf un noeud d'élément, région identique à celle fournie par la fonction range

Intervalles

- Quand il s'agit d'un intervalle compris entre deux points précis:
  - `xpointer(id("chap1")/range-to(id("chap2")))` représente le contenu d'un document XML commençant à l'élément dont l'identificateur est chap1 et se terminant à l'élément dont l'identificateur est chap2.
  - les extrémités de l'intervalle ne sont pas nécessairement dans le même élément et l'intervalle peut très bien ne couvrir que partiellement certains éléments.

Sélection de texte

- La fonction `string-range` prend en argument une expression Xpath identifiant des noeuds et une chaîne de caractères à faire correspondre au texte des noeuds ; elle retourne une région commençant à la 1ère occurrence de la chaîne et la couvrant
  - `xpointer(string-range(//title, 'wizard'))` sélectionne les régions pour toutes les occurrences du mot donné dans les éléments title du document

Partie 11

- XSL – XSLFO

Retour Sommaire

XSL-FO

- XSL-FO est une application XML complète utilisée pour décrire la mise en page du texte
- La plupart du temps on n'écrit pas de XSL-FO mais on écrit une feuille de style XSLT qui transforme le document XML initial en XML-FO
- On peut ensuite transformer le XSL-FO en PDF ou TEX
  - Aucun navigateur ne supporte encore XSL-FO

Pour aller plus loin avec XSL-FO

- Pour une présentation de XSL-FO, on se reportera à [XML 01]

Conclusion

- Provisoire...

XML : une norme qui évolue

- À suivre...