Utilisation de XML dans le cadre des applications internet

Cédric Augustin et IICD - Ougadougou du 24/07/2006 au 28/07/2006

Cette formation se déroule dans le cadre du projet

Renforcement des Capacités et Formation Burkina Faso, 2006

organisé conjointement avec

The International Institute for Communication and Development

Présentation de l'intervenant

Cédric Augustin

Développeur au sein d'une Webagency depuis 2001.

Programme en PHP et Java.

Référent XML, XSLT et XSL-FO depuis 2002.

Initiateur de l'utilisation des CSS.

Formateur en 2000 sur produits IBM.

Objectif de cette formation

Sur une feuille de papier, répondez aux questions suivantes :

Quels sont, d'après vous, les buts de cette formation ?

Que comptez-vous en tirer ?

Objectif de cette formation

Vous allez apprendre :

L'intérêt d'utiliser du XML dans votre activité

Les règles d'écriture

Ce qu'est le web 2.0

Des exemples d'applications du XML

A la fin de cette formation, vous pourrez :

Ecrire du XML sans erreur

Utiliser XSLT pour transformer des fichiers XML

Connaitre quelques formats majeurs s'appuyant sur XML

Transmettre ce savoir faire

Le besoin d'XML

Allons-y !!

Le besoin d'XML

Internet s'est passé très bien de XML pendant des années alors que XML existe depuis 1999, pourquoi cela a-t-il changé ?

Le besoin d'XML

Internet s'est passé très bien de XML pendant des années alors que XML existe depuis 1999, pourquoi cela a-t-il changé ?

Apparition de nouveaux terminaux pour consulter les pages web.

Complexification du contenu des pages.

Maturation des navigateurs et respect des normes.

Automatisation.

L'HTML d'il y a 5 ans

Analysons un document html type d'il y a 5 ans

```
<html>
<head>
<title>Ministère de l'agriculture</title>
</head>
<body bgcolor="#ffffff" leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<table border="0" cellspacing="0" cellpadding="0" width="790" align="center">
  <tr>
    <td bgcolor="#CCCCFF" colspan="4"><a href="#"><img src="img/visuel.jpg" border="0"></a></td>
  </tr>
  <tr>
    <td bgcolor="#FFCCCC" colspan="4">
      <table cellspacing="0" cellpadding="0" width="790">
        <tr>
          <td bgcolor="#FFCCCC" background="img/fond_menu_off.gif"><img src="img/spacer.gif" width="15" height="22"></td>
          <td bgcolor="#FFCCCC" background="img/fond_menu_off.gif"><a href="#">Accueil</a></td>
          <td bgcolor="#FFCCCC" background="img/fond_menu_off.gif"><img src="img/spacer.gif" width="15" height="22"></td>
          <td bgcolor="#FFCCCC" background="img/fond_menu_off.gif"><a href="#">Equipe ministérielle</a></td>
          <td bgcolor="#FFCCCC" background="img/fond_menu_off.gif"><img src="img/spacer.gif" width="15" height="22"></td>
          <td bgcolor="#FFCCCC" background="img/fond_menu_on.gif"><a href="#">Marchés</a></td>
          <td bgcolor="#FFCCCC" background="img/fond_menu_off.gif"><img src="img/spacer.gif" width="15" height="22"></td>
          <td bgcolor="#FFCCCC" background="img/fond_menu_off.gif"><a href="#">Services</a></td>
          <td bgcolor="#FFCCCC" background="img/fond_menu_off.gif"><img src="img/spacer.gif" width="15" height="22"></td>
          <td bgcolor="#FFCCCC" background="img/fond_menu_off.gif"><a href="#">Contact</a></td>
          <td bgcolor="#FFCCCC" background="img/fond_menu_off.gif"><img src="img/spacer.gif" width="15" height="22"></td>
        </tr>
      </table>
    </td>
  </tr>
  <tr>
    <td bgcolor="#FFFFFF" colspan="4"><img src="img/spacer.gif" width="790" height="5"></td>
  </tr>
  <tr>
    <td bgcolor="#FFFFFF" width="490">
    <h1>Pollution, tout est affaire de dosage</h1>
Ainsi, le 23 juin 1969 un tonneau d'un insecticide, l'endosulfan, tombé dans le Rhin en aval de Bingen, pollua ce fleuve sur
```

L'HTML d'il y a 5 ans

Utilisation de tableau

Mise en forme et contenu indissociable

Accessibilité inexistante

Traitement automatisé très difficile

Inefficacité pour le référencement

Détournement des règles d'écriture.

Travail de groupe : comment faudrait-il écrire ce document pour
-> permettre une lecture automatisée (vocale)
-> extraire une partie du contenu
-> afficher le même document sur un écran en 1280x1024 et 600x200

Conclusion sur le besoin d'XML

- Les attentes des designers ont conduit à des détournement de l'HTML et une complexification des pages
- Les nouveaux média et canaux de distribution de l'information nécessitent plus de souplesse dans la présentation
- Pour traiter des données par des logiciels, il faut un format de fichier incluant la sémantique

XML est la réponse à ces contradictions

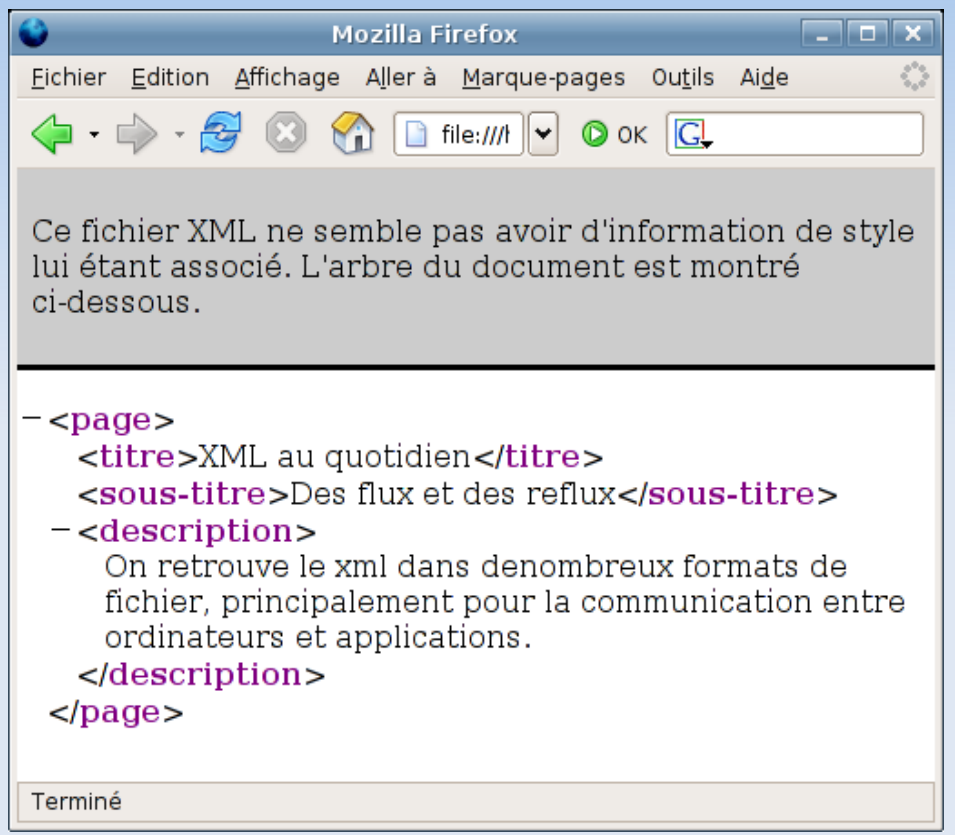
Outils

Voyons voir les outils de travail

Consultation des fichiers XML

Navigateur web

Directement sous la forme d'une arborescence



Consultation des fichiers XML

Navigateur web

Directement sous la forme d'une arborescence

Mis en page avec CSS en rajoutant une déclaration de feuille de style comme en html

```
page {color: #000000; font-size: 9pt}
titre { font-weight: bold;font-size:14pt;display:block}
sous-titre {color: #00CC00;display:block}
```

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css"
href="feuilledestyle.css"?>
<page>
    <titre>XML au quotidien</titre>
    <sous-titre>Des flux et des reflux</sous-titre>
    <description>On retrouve le xml dans de nombreux
    formats de fichier, principalement pour la
    communication entre ordinateurs et
    applications.</description>
</page>
```

XML au quotidien

Des flux et des reflux

On retrouve le xml dans de nombreux formats de fichier, principalement pour la communication entre ordinateurs et applications.

Consultation des fichiers XML

Navigateur web

Directement sous la forme d'une arborescence

Mis en page avec CSS en rajoutant une déclaration de feuille de style comme en html

Transformation avec XSLT (détaillé ultérieurement)

```
<?xml version="1.0"  encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns="http://www.w3.org/1999/xhtml">
  <xsl:template match="/">
    <html>
      <body>
      <xsl:apply-templates />
      </body>
    </html>
  </xsl:template>

  <xsl:template match="titre">
    <h1><xsl:value-of select="."/></h1>
  </xsl:template>
  <xsl:template match="sous-titre">
    <p style="color:#00CC00"><xsl:value-of
    select="."/></p>
  </xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl"
href="feuilledestyle.xsl"?>
<page>
    <titre>XML au quotidien</titre>
    <sous-titre>Des flux et des reflux</sous-titre>
    <description>On retrouve le xml dans de nombreux
    formats de fichier, principalement pour la
    communication entre ordinateurs et
    applications.</description>
</page>
```

XML au quotidien

Des flux et des reflux

On retrouve le xml dans de nombreux formats de fichier, principalement pour la communication entre ordinateurs et applications.

Editer des fichiers XML

Ce sont des fichiers texte.

La coloration syntaxique aide.

Il peut y avoir des encodages.

Certains éditeurs corrigent en même temps que la frappe.

Il est possible d'avoir des éditeurs qui donnent des vues éclatées des données.

```
<?xml version="1.0"?>
<page>
    <titre>XML au quotidien</titre>
    <sous-titre>Des flux et des
    reflux</sous-titre>
    <description>On retrouve le xml dans de
    nombreux formats de fichier, principalement
    pour la communication entre ordinateurs et
    applications.</description>
</page>
```

Editer des fichiers XML

Il existe des quantités d'éditeurs

Des éditeurs spécialisés dans un type d'XML (Dreamweaver pour le XHTML)

Des plugins d'autres outils : XML Developer pour Firefox ou les modules XML pour Eclipse.

Des outils payants, gratuits plus ou moins gourmand en ressources, plus ou moins fiables, plus ou moins respectueux des normes.

Des éditeurs avec assistance ou sans, autocomplétion, correction, exécution de script...

Pour apprendre, nous utiliserons un éditeur de texte avec uniquement la coloration syntaxique.

Traitement et verification

- Parseurs et interpréteurs pour tous les langages, le plus connu est SAX.
- Java est la plateforme que nous utiliserons.
- Vérification en ligne.
- Vérification par Xerces.

Installation de Java et Xerces

Outils

Navigateur web

Editeurs de texte

Parseurs et interpréteurs

XML : eXtensible Markup Language

On rentre dans le vif du sujet

XML : généalogie

- XML a commencé en 1996 et est devenu une norme en 1998.
- Comme HTML, il descend de SGML, dévelloppé dans les années 80 pour permettre les échanges de documenations techniques.
- XML reprend le meilleur de SGML et HTML.
- Contrairement à SGML qui était spécialisé, XML est ouvert.

XML : structuration des données

XML permet de structurer les données en définissant leur organisation.

Les données sont organisées sous la forme d'un arbre avec une seule racine.

Un fichier XML est facilement extensible.

```
<?xml version="1.0"?>
<page>
    <titre>XML au quotidien</titre>
    <sous-titre>Des flux et des
    reflux</sous-titre>
    <description>On retrouve le xml dans de
    nombreux formats de fichier, principalement
    pour la communication entre ordinateurs et
    applications.</description>
</page>
```

HTML <-> XML

HTML et XML sont enfants du même language SGML.

On retrouve dans XML une syntaxe familière.

Contrairement à HTML, les balises n'ont aucun sens en XML.

Un document HTML bien écrit devrait être un document XML => XHTML.

XML est beaucoup plus strict que HTML et ne répare pas un fichier mal formé.

```
<html>
<head>
    <title>XML au quotidien</title>
</head>
<body>
    <h1>XML au quotidien</h1>
    <h2>Des flux et des reflux</h2>
    <p>On retrouve le xml dans denombreux formats
    de fichier, principalement pour la
    communication entre ordinateurs et
    applications.</p>
</body>
</html>
```

```
<?xml version="1.0"?>
<page>
    <titre>XML au quotidien</titre>
    <sous-titre>Des flux et des
    reflux</sous-titre>
    <description>On retrouve le xml dans de
    nombreux formats de fichier, principalement
    pour la communication entre ordinateurs et
    applications.</description>
</page>
```

XML est du texte

- Bien que lisible directement, XML n'a pas vocation a être lu par l'homme, mais par des logiciels.
- Il est facilement débuguable.
- Il est verbeux, mais l'espace disque n'est pas aussi cher que par le passé et HTTP 1.1 compresse les données à la volée.
- Il supporte plusieurs encodages, par défaut UTF-8.

Un métalangage

XML est un métalangage : il défini un ensemble de règles pour d'autres langages.

Les fichiers XHTML, RSS, SVG, XSLT... sont tous des fichiers XML, qui prennent un sens lorsqu'ils sont interprétés.

Le web sémantique

XML est le support du web sémantique.

Aujourd'hui, le contenu du web est fait pour des **lecteurs humains**, pas des ordinateurs.

L'idée du web sémantique est de permettre des recherches intelligentes faites par des ordinateurs basées sur des définitions comprises par le monde entier : **donner un sens aux informations**.

Une balise XML n'a pas de sens, c'est RDF qui lui en donne, par exemple “*une femme d'oncle est une tante*” qui va être utilisé pour analyser un arbre généalogique.

RSS est un exemple de web sémantique.

Espace de nom

En utilisant les espaces de nom, il est possible de mélanger plusieurs sources XML sans qu'il y ait de risque de confusion.

Libre...

Le format XML est libre de tout brevet ou restriction.

Il est gratuit.

Il est normalisé et ses spécifications totalement accessibles.

Des données enregistrées en XML seront toujours exploitables : pérénité de l'investissement.

XML : eXtensible Markup Language

XML est une norme depuis 1998 et s'appuit sur l'expérience de SGML (1986) et HTML (1990).

XML permet de structurer des données

Il ressemble à HTML (balises et attributs), mais les balises n'ont aucune signification et ne servent qu'a délimiter les données.

XML est du texte lisible par les développeurs, mais n'est pas destiné à être lu par des humains, mais par des programmes.

XML est un méta language : il défini des règles de base pour écrire d'autres langages (XHTML, XSLT, FO, RSS...).

XML est à la base du web sémantique.

XML est modulaire : en utilisant des espaces de nom, il est possible de combiner n'importe quel XML.

XML est libre et gratuit.

Créer un fichier xml

Les règles

Régles d'écriture : structure

Un document XML comporte au moins un élément

```
<text>Ceci est un document XML</text>
```

Il n'y a qu'un seul élément racice (root).

Le nom de la balise de fin d'un élément doit correspondre à celui de la balise de début. Les noms tiennent compte des majuscules et des minuscules.

Les éléments délimités par les balises de début et de fin doivent s'imbriquer correctement les uns dans les autres.

```
<text>Ceci est <imp>erroné</text></imp>
```

Un élément sans contenu prend la forme spéciale suivante : `<nom/>` == `<nom></nom>`

Régles d'écriture : caractère

- Les noms d'éléments peuvent comporter des lettres, des chiffres, des tirets, des traits de soulignement, des deux-points ou des points : [a-zA-Z0-9_:\.\-]
- Les deux-points (:) sont à éviter car utilisés pour les espaces de nom.
- Les noms d'éléments commençant par xml sont réservés à la norme XML.

```
<permittedNames>
      <name/>
      <xsl:copy-of/>
      <A_long_element_name/>
      <A.name.separated.with.full.stops/>
      <a123323123-231-231/>
      <_12/>
</permittedNames>
```

Régles d'écriture : attribut

- Un élément peut comporter aucun, un ou plusieurs attributs.
- Les caractères autorisés sont les mêmes que pour les noms d'éléments.
- La valeur de l'attribut doit être indiquée entre guillemets simples '...' ou doubles "..." .
- Si un guillemet simple ou double est utilisé dans la valeur d'un attribut, le délimiteur contraire doit être utilisé.

```
<elements-with-attributes>
     <el _ok = "oui" />
     <one attr = "une valeur"/>
     <several first="1" second = '2' third= "333"/>
     <aquote cas1="Aujourd'hui" cas2='Il dit: "Salut à tous"'/>
</elements-with-attributes>
```

Règles d'écriture : attribut ou élément ?

Schématiquement, du point de vue d'XML, un attribut est un élément qui ne peut pas avoir d'enfant.

Les traitements par défaut sur XML ne concernent que le contenu d'un élément et non ses attributs.

```
<aniversaire date="2006-07-14">
Fête nationnale
</aniversaire>
```

→ `Fête nationnale`

```
<aniversaire>
  <libelle>Fête nationnale</libelle>
  <date iso="2006-07-14">
    <fra>14/07/2006</fra>
  </date>
</aniversaire>
```

→ `Fête nationnale14/07/2006`

Règles d'écriture : caractères interdit

- Les caractères < et & ne peuvent pas être utilisés dans le texte, car ils sont utilisés dans le balisage. Il sont remplacé par < et &.
- Les caractères >, " , et ' peuvent également être remplacés par > , " et ' respectivement

```
<example>
     <isLower>
          23 &lt; 46
     </isLower>
     <ampersand>
          Dupond &amp; fils
     </ampersand>
     <right-bracket> A la fois > et &gt; sont autorisés</right-bracket>
     <double-quote> A la fois " et &quot; sont autorisés</double-quote>
     <apostrophe> A la fois ' et &apos; sont autorisés</apostrophe>
     Cela est utile dans : <el value=" &apos; &quot; &apos; "/>
</example>
```

Règles d'écriture

Des commentaires peuvent figurer n'importe où dans un document en dehors des autres balises.

```
<nom>Fred<!-- Un commentaire & un autre --></nom>
```

La chaîne de caractères "--" (deux tirets) ne doit pas figurer à l'intérieur des commentaires.

Les instructions de traitement (PI - Processing instruction en anglais) permettent aux documents XML de contenir des instructions destinées aux applications.

```
<nom><?php echo 'Fred'; ?></nom>
```

Les documents XML peuvent, et doivent, commencer par une déclaration XML qui précise la version de la norme XML utilisée et éventuellement l'encodage (par défaut UTF-8).

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<text>Si aucun codage n'est indiqué, UTF-8 est pris par défaut</text>
```

Règles d'écriture : CDATA

- Les sections CDATA permettent de ne pas traiter les blocs de texte comportant des caractères qui seraient normalement identifiés comme du balisage.
- Les sections CDATA commencent par la chaîne "**<![CDATA[**" et se terminent par la chaîne "**]]>**".
- La chaîne ']]>' ne doit pas figurer à l'intérieur d'une section CDATA.

```
<example>
      <![CDATA[ <aaa>bb&cc<<<]]>
</example>
```

Règles d'écriture : résumé

Un seul élément racine.

Nom des balises sensiblent à la casse.

Imbrication.

Elément vide sous la forme raccourcie <element/>.

Caractères nom d'élément et attribut : a-zA-Z0-9:.-

Caractères < et & interdits, remplacés par < et &

Commentaire : <!-- le commentaire -->

Déclaration <?xml version="1.0"?>

Partie non traitées <![CDATA[...]]>

Règles d'écriture : exercice

Corriger le document HTML de départ pour en faire un document XML valide (conservation des balises html).

Analyser les informations contenues dans ce document et proposez une structure XML pour les stocker (suppression des balisages HTML).

Réalisation du document XML et validation.

Comparaison des productions :

- hyérarchisation,
- attribut ou élément,
- données manquantes pour qualifier les informations.

XHTML: de l'HTML au XML

C'est pas si sorcier

HTML : le besoin de norme

Lors de l'évolution d'HTML, les navigateurs ont appris à gérer de l'HTML comportant des erreurs :
- L'évolution d'HTML se faisant avant la norme,
- Guerre des navigateurs,
- HTML a vocation a être affiché, donc la priorité était donnée au rendu.

Le besoin de norme est devenu criant à cause des comportements différents des navigateurs.

Avec HTML 4.01, tous les éléments sont en place pour un **affichage** enfin réellement indépendant du type de navigateur.

Doctype

Le Doctype indique au navigateur que l'auteur de la page c'est ce que c'est.

Il fait passer le navigateur en mode strict.

Permet de valider un document.

Les différents Doctypes

Version	Balise
HTML 2.0	<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">
HTML 3.2	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
HTML 4.01 - Strict	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
HTML 4.01 - Transitional	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
HTML 4.01 - Frameset	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
XHTML 1.0 - Strict	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
XHTML 1.0 - Transitional	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
XHTML 1.0 - Frameset	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
XHTML 1.1	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

Convergence

La convergence de HTML vers XML est inéluctable

Besoin de traiter les fichiers (moteur, syndication),

Monté en puissance de CSS : séparation du fond et de la forme,

Affichage multi-plateforme.

Qu'apporte le X

XHTML 1.0 est une "*reformulation de HTML en XML*"

Il rend compatible avec les anciens navigateurs des documents XML,

Application des technologies et outils XML possible.

Evolutivité plus facile.

XML apporte la technique des espaces de nom : plusieurs DTD au sein d'un seul document.

XHTML est conçu pour accepter des extensions grâce à des modules XHTML (XHTML Modularization)

Différences entre XHTML et HTML

Une fois HTML normé, il ne manquait plus qu'une étape pour le rendre compatible avec XML :

Toute balise ouverte doit être fermée :
 =>

Standardisation : les noms des balises et attributs doivent être en minuscules.

Les attribut s'écrivent forcément avec une valeur : checked => checked=”checked”.

Toute image a un attribut ALT, même de valeur nulle.

Protéger le contenu des éléments STYLE et SCRIPT avec <![CDATA[au début et]]> à la fin.

On change de Doctype.

Et voila le XHTML transitional !

XHTML transitional

Le Doctype transitional permet de migrer facilement des pages HTML en XHTML.

Le rendu des pages utilise le mode *Quirk* du navigateur.

Ce ne peut être que *transitionnel* :
- limitations lié au navigateur,
- technologies ou outils non à jour.

A terme tous les nouveaux documents du web doivent devenir XML.

XHTML strict

Permet de bénéficier du rendu strict des navigateurs :
application universelle de la norme et du rendu.

Dissocie totalement le fond de la forme :
- multi plateforme, multi média
- accessibilité

Mise en page exclusivement réalisée en CSS.

Permet des traitements sur les pages.

Encore des hésitations ?

Les traitements XSLT génèrent du XML.

Les bases de données peuvent stocker du XML.

La syndication fournie des flux XML.

Une page XHTML peut être automatiquement convertie en PDF.

Dissocier fond et forme peut réduire les coûts.

Exercice

Convertir la page HTML de départ en :

-> HTML 4.01 strict

-> XHTML 1.0 transitional

-> XHTML 1.0 strict

Sur la dernière version, faire une feuille de style CSS pour retrouver la mise en page initiale.

XSLT : les transformations d'XML

La cuisine de l'XML à la sauce XSL

XSL

Le langage XSL vous permet de modifier librement n'importe quel texte source XML pour le transformer en un nouveau fichier XML ou texte.

Les fichiers XSLT sont des fichiers XML qui portent l'extension xsl (T pour template).

XSLT utilise l'espace de nom <xsl:...>

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl"
href="feuilledestyle.xsl"?>
<page>
    <titre>XML au quotidien</titre>
    <sous-titre>Des flux et des reflux</sous-titre>
    <description>On retrouve le xml dans de nombreux
    formats de fichier, principalement pour la
    communication entre ordinateurs et
    applications.</description>
</page>
```

```
<?xml version="1.0"  encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns="http://www.w3.org/1999/xhtml">
  <xsl:template match="/">
    <html>
      <body>
      <xsl:apply-templates />
      </body>
    </html>
  </xsl:template>

  <xsl:template match="titre">
    <h1><xsl:value-of select="."/></h1>
  </xsl:template>
  <xsl:template match="sous-titre">
    <p style="color:#00CC00"><xsl:value-of
    select="."/></p>
  </xsl:template>
</xsl:stylesheet>
```

XML au quotidien

Des flux et des reflux

On retrouve le xml dans de nombreux formats de fichier, principalement pour la communication entre ordinateurs et applications.

XSL : premier fichier

- Chaque feuille de style XSL doit commencer par l'élément xsl:stylesheet.
- L'attribut version='1.0' précise la version de la spécification XSL(T).
- Cet exemple présente une feuille de style la plus simple possible. Comme elle ne comporte aucune information, le traitement par défaut est appliqué.

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

</xsl:stylesheet>
```

XSL : premier fichier

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

</xsl:stylesheet>
```

Appliquez la feuille de style ci-dessus aux fichiers XML créés précédemment en ajoutant la déclaration du XSLT dans l'entête du XML.

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<?xml-stylesheet type="text/xsl" href="05a_xsldefaut.xsl"?>
<document...
```

Appliquez la feuille de style en utilisant le parseur saxon.

```
java -jar saxon8.jar 03_xml.xml 05a_xsldefaut.xsl > 05a_xsldefaut.html
```

XSL : premier fichier

Résultat :

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

Pollution, tout est affaire de dosage
Ainsi, le 23 juin 1969 un tonneau d'un insecticide, l'endosulfan, tombé dans le Rhin en aval de Bingen, pollua ce fleuve sur 600 kilomètres (jusqu'à son embouchure), faisant périr plus de 20 millions de poissons selon certaines estimations.
La catastrophe de Seveso, survenue le 10 juillet 1976, a donné une illustration saisissante des risques écotoxicologiques associés à un polluant aussi toxique et persistant que la dioxine. Ce jour-là, l'explosion d'un réacteur de synthèse de trichlorophénol
...

XSLT : parsing simple

Un processeur XSL analyse un fichier XML source et s'efforce de trouver une règle modèle concordante. S'il en trouve une, les instructions qu'elle contient sont évaluées.

```
<xsl:stylesheet version = '1.0'
    xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:template match="bold">
    <p><b><xsl:value-of select="."/></b></p>
</xsl:template>

<xsl:template match="red">
    <p style="color:red"><xsl:value-of select="."/></p>
</xsl:template>

<xsl:template match="italic">
    <p><i><xsl:value-of select="."/></i></p>
</xsl:template>
</xsl:stylesheet>
```

```
<source>
<bold>Hello, world.</bold>
<red>I am </red>
<italic>fine.</italic>
</source>
```

```
<p><b>Hello, world.</b></p>

<p style="color:red">I am </p>

<p><i>fine.</i></p>
```

XSLT : parsing simple

Extraction de contenu

Utilisation de l'élément **xsl:value-of**.

Utilisation de l'instruction **xsl:apply-templates**

Un processeur XSL s'efforce de trouver une règle modèle concordante

```
<source>
<employee>
  <name>Joe</name>
  <surname>Smith</surname>
</employee>
</source>
```

```
<xsl:stylesheet version = '1.0'
    xmlns:xsl='http://www.w3.org/1999/XSL/Transfo

<xsl:template match="employee">
  <b><xsl:value-of select="name"/></b>
  <xsl:apply-templates select="surname"/>
</xsl:template>

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

```
<b>Joe</b>
<i>Smith</i>
```

XSLT : chemin de localisation

Les chemins de localisation servent à accéder aux parties d'un document XML auquel un modèle doit être appliqué.

Elle est très similaire à celle utilisée pour l'adressage des systèmes de fichiers

```
<xsl:template match="/source/employee/surname">
    <p style="color:red">
        <xsl:value-of select="name()"/>
        <xsl:text> = </xsl:text>
        <xsl:value-of select="."/>
    </p>
</xsl:template>
```

```
<source>
<employee>
  <name>Joe</name>
  <surname>Smith</surname>
</employee>
</source>
```

```
<p style="color:red">
surname
 =
Smith
</p>
```

XSLT : exercice extraction simple

A partir de votre fichier XML, extrayez quelques données :
- le titre et le texte de l'article
- les actualités
- les 2 ensembles

Utilisez **value-of**
puis **xsl:apply-templates**

```
<xsl:template match="employee">
  <b><xsl:value-of select="name"/></b>
  <xsl:apply-templates select="surname"/>
</xsl:template>

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

Extrayez le titre et le texte de l'article et essayez de générer une page XHTML strict valide.

```
<?xml version="1.0"  encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns="http://www.w3.org/1999/xhtml">
<xsl:output method="xhtml" encoding="ISO-8859-1"
     doctype-public = "-//W3C//DTD XHTML 1.0 Strict//FR"
     doctype-system = "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"/>
```

XSL : traitement par défaut

Le traitement commence toujours par match="/" (racine)

Cet élément est facultatif. S'il n'est pas indiqué, c'est le modèle implicite qui est utilisé : *traiter tous les noeuds fils du noeud courant.*

Lorsqu'il y a un modèle pour un noeud particulier, le traitement par défaut n'est pas appelé

```
<xsl:template match="/">
    <xsl:apply-templates />
</xsl:template>
```

```
<xsl:template match="/source">
    <xsl:apply-templates/>
</xsl:template>
```

XSLT : recherche de concordances

Un modèle peut rechercher des concordances à partir d'une sélection de chemins de localisation: chaque chemin de localisation étant séparé des autres par le caractère "|".

```
<xsl:template match="firstname|surname">
```

Le caractère générique "*" sélectionne toutes les possibilités.

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

"//" signifie : sélectionner tous les noeuds du document du type spécifié.

"//" au milieu d'un chemin signifie : sélectionner tous les noeuds apparaissant dans le noeud sélectionné par la première partie du chemin de localisation

```
<xsl:apply-templates select="//BBB" />
```

```
<xsl:apply-templates select="/source/AAA//CCC"/>
<xsl:apply-templates select="/source//AAA/BBB//*"/>
```

XSLT : mode

Avec l'attribut mode, un élément peut être traité plusieurs fois pour générer un résultat différent à chaque fois.

```
<xsl:template match="/">
    <xsl:apply-templates select="//CCC" mode="red"/>
    <xsl:apply-templates select="//CCC" mode="blue"/>
    <xsl:apply-templates select="//CCC" mode="green"/>
    <xsl:apply-templates select="//CCC"/>
</xsl:template>

<xsl:template match="CCC" mode="red">
  <div style="color:red"><xsl:value-of select="name()"/></div>
</xsl:template>

<xsl:template match="CCC" mode="blue">
  <div style="color:blue"><xsl:value-of select="name()"/></div>
</xsl:template>

<xsl:template match="CCC">
  <div><xsl:value-of select="name()"/></div>
</xsl:template>
```

XSLT : priorité

Lorsque plusieurs modèles sélectionnent le même élément

Le modèle le moins précis à la plus faible priorité : le modèle CCC est plus faible que celui de CCC/CCC

"*" a toujours une priorité plus faible que CCC.

L'ordre de priorité peut être spécifié à l'aide de l'attribut *priority*.

Le calcul des priorités s'échelonne de -0.5 à 0.5.

XSLT : resumé

alternatif : <xsl:template match="firstname|surname">

generique : <xsl:template match="*">

chemin partiel : <xsl:apply-templates select="/source//AAA/BBB//*"/>

mode : <xsl:apply-templates select="//CCC" mode="red"/>
<xsl:template match="CCC" mode="red">

priorité au plus précis : CCC/CCC > CCC > * ou en fct de l'attribut priority

Mettre en oeuvre ceci des les exercices suivants :
- faire un template qui extrait les dates des actualités ou des cours des marchés.
- extraire tous les titres (ou libelles, ou ...) du document indépendamment de leur position dans le document xml.
- afficher de couleurs différentes les cours en fonction de leur monnaie
- vérifier le fonctionnement des priorité en créant 2 templates qui portent sur les même données.

XSLT : attributs

Les attributs sont accessibles de la même manière que les éléments en ajoutant le caractère "@" devant les noms d'attributs.

```
<xsl:value-of select="data/@color"/>
```

Les attributs peuvent être traités de la même manière que les éléments.

On peut sélectionner des élément qui comportent ou pas un attribut.

```
<xsl:template match="car[@checked]">
    <p>
        <xsl:text>Car: </xsl:text>
        <xsl:value-of select="@id"/>
    </p>
</xsl:template>
```

- Extraire les dates au format d'affichage (balise) et format iso (attribut) des actualités.
- Extraire les cours qui ont obligatoirement une unité.

XSLT : axe

ancestor : tous les ancêtres jusqu'à la racine.

ancestor-or-self : l'élément et tous ses ancêtres

attribute : tous les attribut de l'élément.

child : tous les enfants de l'élément.

descendant : tous les descendants de élément.

descendant-or-self : l'élément et ses descendants.

following : les éléments suivants de même niveau.

following-sibling : les éléments suivants de même niveau et de même nom.

namespace : l'espace de nom de l'élément.

parent : le parent de l'élément.

preceding : les éléments précédents de même niveau.

preceding-sibling : les éléments précédents de même niveau et de même nom.

self : l'élément lui même.

XSLT : axe

```
<entreprise>
  <service nom="comptabilité">
    <employe>Joe</employe>
    <employe>Annie</employe>
  </service>
  <service nom="vente">
     <employe>Saly</employe>
     <employe>Smith</employe>
  </service>
</entreprise>
```

```
<xsl:template match="employe">
  <b><xsl:value-of select="parent::./@nom"/>:</b>
  <xsl:value-of select="."/>
</xsl:template>
```

```
<b>comptabilité:</b>Joe
<b>comptabilité:</b>Annie
<b>vente:</b>Saly
<b>vente:</b>Smith
```

XSLT : axe

L'axe child:: peut être omis d'une étape de localisation, car il constitue l'axe par défaut.

L'axe attribute:: peut être abrégé en @.

// est la forme abrégée de l'axe /descendant-or-self::,

. de l'axe self::

.. de l'axe parent::.

<xsl:value-of select="**child::BBB/attribute::id**" />
est équivalent à
<xsl:value-of select="**BBB/@id**" />

XSLT : boucle et trie

L'instruction **xsl:for-each** comporte un modèle qui est appliqué à chaque noeud sélectionné à l'aide de l'attribut **select**.

Les noeuds sélectionnés avec l'élément xsl:for-each ou xsl:apply-templates peuvent être triés.

```
<xsl:template match="//service">
  <xsl:for-each select="employe">
    <xsl:sort order="ascending" select="."/>
    <b><xsl:value-of select="../@nom"/>:</b>
    <xsl:value-of select="."/>
  </xsl:for-each>
</xsl:template>
```

- Utilisez **for-each** pour afficher touts les actualités (ou cours ou etp).
- Triez les par ordre antéchronologique.
- même chose avec **apply-template**.

XSLT : trie

Il est possible de faire un trie textuel ou numérique :
- en mode textuel 10 précède 2
- en mode numérique 2 précède 10

```
<xsl:sort data-type="text" select="montant" />
<xsl:sort data-type="number" select="montant" />
```

Les lettres majuscules peuvent être trié avant ou après les lettres minuscules

```
<xsl:sort case-order="upper-first" select="nom" />
<xsl:sort case-order="lower-first" select="nom"/>
```

Trier les cours par montants

XSLT : élément

L'élément **xsl:element** génère des éléments lors du traitement.

```
<source>
<text size="H1">Header1</text>
<text size="H3">Header3</text>
<text size="b">Bold text</text>
<text size="sub">Subscript</text>
<text size="sup">Superscript</text>
</source>
```

```
<xsl:template match="/">
    <xsl:for-each select="//text">
        <xsl:element name="{@size}">
            <xsl:value-of select="."/>
        </xsl:element>
    </xsl:for-each>
</xsl:template>
```

```
<H1>Header1</H1>
<H3>Header3</H3>
<b>Bold text</b>
<sub>Subscript</sub>
<sup>Superscript</sup>
```

XSLT : attribute

L'élément **xsl:attribute** génère des attributs lors du traitement.

Il crée un attribut dans l'élément qui le porte.

```
<xsl:template match="information">
  <table>
    <xsl:attribute name="class"><xsl:value-of select="./@style"/></xsl:attribute>
    <tr>
      <td><xsl:value-of select="./libelle" /></td>
      <td><xsl:value-of select="./valeur" /></td>
    </tr>
  </table>
</xsl:template>
```

- Affichez l'image avec ses attributs de taille et le ALT
- Afficher des images dont les largeurs sont proportionnelles à l'ETP

XSLT : copy

Les éléments XSL Copy et Copy-of servent à copier des noeuds.

L'élément Copy ne copie que le noeud courant sans ses fils, ni les attributs,

L'élément Copy-of copie tout.

```
<entreprise>
  <service nom="comptabilité">
    <employe>Joe</employe>
    <employe>Annie</employe>
  </service>
  <service nom="vente">
     <employe>Saly</employe>
     <employe>Smith</employe>
  </service>
</entreprise>
```

```
<xsl:copy-of select="//service[@nom='comptabilité']" />
```

```
<service nom="comptabilité">
  <employe>Joe</employe>
  <employe>Annie</employe>
</service>
```

Créer un fichier xml ne comportant que l'article

```
<xsl:copy select="//service[@nom='comptabilité']" />
```

```
<service />
```

XSLT : opérateurs

Opérations mathématiques : +, -, *, **div**, mod

Comparaisons : <, <=, =, >, >=

Opérateurs booléen : and, or

Union : |

XSLT : traitement conditionel

L'instruction **xsl:if** permet le traitement conditionnel.

```
<xsl:if test="not (position()=last())">
  <xsl:text>, </xsl:text>
</xsl:if>
```

L'élément **xsl:choose** permet de sélectionner une possibilité parmi plusieurs.

```
<div>
  <xsl:attribute name="class">
    <xsl:choose>
      <xsl:when test="./@statut='important'">mis-en-avant</xsl:when>
      <xsl:otherwise>normal</xsl:otherwise>
    </xsl:choose>
  </xsl:attribute>
  <xsl:value-of select="." />
</div>
```

Afficher les ETP avec des couleurs différentes en fonction des valeurs :
- en vert si < 5 sinon sans couleur (xsl:if)
- en vert si < 2, orange si entre 2 et 6, rouge si au dela, sans couleur si pas d'ETP (xsl:choose).

XSLT : Fonctions

boolean
ceiling
comment
concat
contains
count
current
document
element-available
false
floor
format-number
function-available
generate-id
id
key
lang
last
local-name
name
namespace-uri
node
normalize-space
not
number
position
processing-instruction
round
starts-with
string
string-length
substring
substring-after
substring-before
sum
system-property
text
translate
true
unparsed-entity-uri

XSLT : exercice fonction

Faire la somme de tout les ETP.

Faire la somme tous les ETP supérieurs à 3.

Convertir une date au format iso en une date au format anglais (MM/JJ/AAAA).

Afficher les actualités en alternant la couleur de fond (blanc/gris)

XSLT : variable

Une variable est définie une seul fois et valable dans l'arborescence où elle a été définie.

Il peut y avoir des variables globales ou locales.

Ce sont plutôt des constantes.

```
<xsl:variable name="totalChapters">
  <xsl:value-of select="count(//chapter)"/>
</xsl:variable>

<xsl:template match="/">
  <xsl:for-each select="//chapter">
    <xsl:value-of select="."/> 
    <xsl:value-of select="position()" />
    <xsl:text>/</xsl:text>
    <xsl:value-of select="$totalChapters" />
  </xsl:for-each>
</xsl:template>
```

Afficher les ETP dans un tableau horizontal au lieu de vertical (colspan)

XSLT : paramètre

- Fonctionnent comme les variables.
- Sont déclarés avec une valeur par défaut.
- La valeur peut être remplacée à chaque utilisation (pseudo-fonction)

```
<xsl:for-each select="//nombre">
  <xsl:choose>
    <xsl:when test="text() mod 2">
      <xsl:apply-templates select=".">
        <xsl:with-param name="type">paire</xsl:with-param>
      </xsl:apply-templates>
    </xsl:when>
    <xsl:otherwise>
      <xsl:apply-templates select="."/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:for-each>

<xsl:template match="nombre">
  <xsl:param name="type">impair</xsl:param>
  <xsl:value-of select="." /> 
  <xsl:value-of select="$type"/>
</xsl:template>
```

XSLT : import

D'autres feuilles de style peuvent être importées (xsl:import) ou incluses (xsl:include) dans une feuille de style.

Lors de l'importation les définitions et les règles modèle de la feuille de style qui importe ont la priorité sur celles de la feuille de style importée.

```
<xsl:import href="id3.xsl"/>
<xsl:include href="id2.xsl"/>
```

L'utilisation de apply-import permet de remplacer les règles courantes par celles de la feuille de style importée.

Créer une feuille de style pour l'article et une pour les encarts.
Utiliser ces 2 feuilles conjointement pour afficher toute la page.

XSLT : exercices récapitulatifs

Utilisez tout ce qui a été vu pour générer une page XHTML strict qui comporte tous les éléments de la page d'exemple.

Créer une version de la page uniquement pour l'impression (article uniquement).

Faire évoluer le fichier XML source pour le rendre plus pratique à utiliser avec XSLT.

Web 2.0

Le partage d'information

Web 2.0 ?

Le terme est apparu fin 2004 pour illustrer la mutation que vit l'internet.

C'est le passage d'une collection de site web à une plate forme informatique.

Mise en réseau des ressources : information, logiciel, matériel, production.

Avant	Après
Unidirectionel	Bidirectionel
Pour des humains exclusivement	Traitement par des ordinateurs en plus
Site perso et gestionnaire de contenu	Blog et wiki
Formulaire	AJAX
Données localisées	Syndication et RSS
Mise en page du site très importante	Données multi-format
Lois du plus fort	Respect des normes et formats ouverts
Individu	Réseaux sociaux
FTP	Peer to Peer
Achat de licence	Paiement à l'usage ou gratuité via la pub

Web 2.0 : le concept

Web 2.0 : le concept

Web 2.0 : exemple d'évolution

DoubleClick --> Google AdSense

Akamai --> BitTorrent

mp3.com --> Napster

Britannica Online --> Wikipedia

sites perso --> blogs

spéculation sur les noms de domaines --> optimisation pour moteurs de recherche

pages vues --> coût au clic

« screen scraping » --> services web

publication --> participation

système de gestion de contenu --> wikis

arborescence (taxonomie) --> tags ("folksonomy")

rigidité du contenu --> syndication de contenu

Web 2.0 : réalité ou arnaque

Pas de définition exacte. Ensemble d'évolution de l'existant : tout est déjà en place au niveau technologique.

Nouveau model économique pas forcément au bénéfice de l'utilisateur final :
-> Les utilisateurs fournissent les données,
-> Les utilisateurs fournissent les métadonnées,
-> Les utilisateurs construisent l'application,
-> Les utilisateurs payent la compagnie continuellement pour avoir le droit d'utiliser l'application qu'ils ont construit pour accéder et manipuler les données qu'ils ont fournit.

Web 2.0 : principes et mots clés

Intelligence collective.

Réseau social.

Webservice.

Syndication.

Collaboratif, e-collaboratif.

C2C : consomateur à consomateur.

Données, données et données.

Logiciels en évolution continue.

Utilisation simplifiée et interface améliorées.

Web 2.0 : application

Imaginez dans votre environnement professionnel :

Proposer des exemples de webservice.

Proposer des exemples de syndication.

Proposer des exemples d'applications en ligne pouvant remplacer des logiciels installés.

RSS : Really Simple Syndication

Bon ben on en fait quoi de votre contenu ?

RSS : Really Simple Syndication

RDF Site Summary, Rich Site Summary ou **Really Simple Syndication**.

Développé en 1999 par Netscape.

Un dialecte de XML.

Un format non standardisé.

Plusieurs versions : 0.9, 0.9X, 1.0, 2.0, **2.01**, 3.0

Un concurrent : **Atom**

Les lecteurs de flux s'appuient sur RSS 2 et Atom

Atom

L'hétérogénéité de RSS a conduit à la recherche d'une norme (2003).

Objectif d'universalité.

Atom 1.0 est décrit dans la RFC 4287 (http://www.ietf.org/rfc/rfc4287.txt)

Volonté d'intégrer les points fort de RSS :

- Convergence des 2 types de formats ?
- Cohabitation ? les lecteurs acceptent les 2.

RSS Atom : syndication

Exemple de web sémantique : informations qualifiées.

Elément du web 2.0 :

- push,
- informations distribuées,
- agrégation et méta-information,
- contribution.

Explosion grâce aux blogs.

De plus en plus de site proposent un fils RSS

```
<link rel="alternate" type="application/rss+xml"
      title="Actualités du Dico du Net (RSS 0.91)"
      href="http://www.dicodunet.com/actualites.xml">
```

RSS : pourquoi faire ?

Les flux RSS/Atom ont vocation à présenter des contenus régulièrement mis à jour.

Utilisation via un lecteur ou agrégation sur un autre site.

Accès aux données sans aller sur le site :

- Reprise des titre ou du texte intégral,
- Générateurs de trafic.

L'auteur du site choisit les flux.

Lecteur : outil de flux et pas de stockage.

Fils d'info par email (stockage et veille).

Diffusion de données multimédia.

RSS 2 : structure du fichier

```
<?xml version="1.0"?>
<rss version="2.0">
<channel>
   <title>Les actus de mon site</title>
   <link>http://www.monsite.com</link>
   <description>Retrouver les principales infos</description>
    <item>
      <title>Nouvelle rubrique</title>
      <link>http://www.monsite.com/article.php?id=122</link>
      <description>Suite à la demande de nombreux intervenants
         du forum, une nouvelle rubrique  est ouverte</description>
   </item>
</channel>
</rss>
```

RSS 2 : structure du fichier

Principales balises RSS 2.0

```
<rss version="2.0">
<channel>
   <title>Les actus de mon
   <link>http://www.monsit
   <description>Retrouver l
    <item>
      <title>Nouvelle rubriqu
      <link>http://www.mons
      <description>Suite à la
        du forum, une nouve
   </item>
</channel>
</rss>
```

Créez une feuille XSLT pour générer le flux RSS des actualités, cours...

rss Le conteneur global.

channel Un canal. Il contient plusieurs balises descriptives, et une série de balises "item", les informations.

Les balises descriptives obligatoires du canal

- **title** titre du canal.
- **link** adresse du site.
- **description** du canal.
- une ou plusieures balises **item**.

Balises obligatoires d'un item

- **title** titre de l'article.
- **link** l'URL de la page.
- **description** résumé de l'article.

RSS 2.0 en français : http://www.stervinou.com/projets/rss/

RSS 1.0 : utilisation de RDF

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns="http://purl.org/rss/1.0/" xmlns:dc="http://purl.org/dc/elements/1.1/">
  <channel rdf:about="http://xul.fr/auteurs.rss">
    <title>Après Gutemberg</title>
    <link>http://xul.fr/</link>
    <items>
      <rdf:Seq>
        <rdf:li resource="http://xul.fr/Diderot/"/>
        <rdf:li resource="http://xul.fr/Rousseau/"/>
      </rdf:Seq>
    </items>
  </channel>
  <item rdf:about="http://xul.fr/Diderot/">
     <title>Diderot</title>
     <link>http://xul.fr/Diderot/</link>
     <description>Religieuse et rameau sans fête</description>
  </item>
  <item rdf:about="http://xul.fr/Rousseau/">
     <title>Rousseau</title>
     <link>http://xul.fr/Rousseau/</link>
     <dc:date>1712</dc:date>
  </item>
</rdf:RDF>
```

Atom

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title>Example Feed</title>
  <link href="http://example.org/"/>
  <updated>2003-12-13T18:30:02Z</updated>
  <author>
    <name>John Doe</name>
  </author>
  <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>

  <entry>
    <title>Atom-Powered Robots Run Amok</title>
    <link href="http://example.org/2003/12/13/atom03"/>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <updated>2003-12-13T18:30:02Z</updated>
    <summary>Some text.</summary>
  </entry>
</feed>
```

RSS 2 : étendre le langage par les espaces de nom

```
<?xml version="1.0"?>
<rss version="2.0"
  xmlns:blogChannel="http://backend.userland.com/blogChannelModule">
  <channel>
    <title>Scripting News</title>
    <link>http://www.scripting.com/</link>
    <description>A weblog about scripting and stuff like that.</description>
    <language>en-us</language>
    <blogChannel:blogRoll>
      http://radio.weblogs.com/0001015/userland/scriptingNewsLeftLinks.opml
    </blogChannel:blogRoll>
      <item>
      <description>Joshua Allen:
      <a href="http://www.netcrucible.com/blog/2002/09/29.html#a243">
      Who loves namespaces?</a></description>
      <pubDate>Sun, 29 Sep 2002 19:59:01 GMT</pubDate>
      <guid>
      http://scriptingnews.userland.com/backissues/2002/09/29#When:12:59:01PM
      </guid>
      </item>
  </channel>
</rss>
```

RSS : conseils

- Utiliser un validateur (http://www.feedvalidator.org/),
- Archiver les infos trop âgées,
- Créer plusieurs flux thématiques,
- Utiliser la balise image de chanel : identification dans les agrégateurs,
- Penser les titres et descriptions séparément.

XSL-FO : Formated Object

Z'en faites quoi de toutes ces impressions ?

XSL: FO

XML avec l'espace de nom fo.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
```

Permet de décrire la mise en forme d'informations qui est ensuite traitée par un parseur (ex: FOP).

Le format de sortie n'est pas forcément PDF.

Une fichier XSL-FO n'est pas un format de stockage des données.

Très verbeux : vocation à être traité par XML + XSL.

XSL-FO : FOP

Formatting Objects Processor.

Projet Open Source de la fondation Apache.

Mise en forme d'objets pour l'édition :

- **PDF**,
- *PS*, PCL (imprimante),
- *RTF (Rich Text Format),*
- SVG,
- XML (debug),
- Print, AWT,
- *MIF*,
- TXT (archivage).

Tend à devenir conforme au standard XSL-FO 1.0 du W3C.

Version 0.20.25 et 0.9b

XSL-FO : premier fichier

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

  <fo:layout-master-set>
    <fo:simple-page-master master-name="simple" page-height="29.7cm" page-width="21cm"
      margin-top="1cm" margin-bottom="2cm" margin-left="2.5cm" margin-right="2.5cm">
      <fo:region-body margin-top="3cm" margin-bottom="1.5cm"/>
      <fo:region-before extent="3cm"/>
      <fo:region-after extent="1.5cm"/>
    </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence master-reference="simple">
    <fo:flow flow-name="xsl-region-body" text-align="justify">
      <fo:block font-size="12pt" font-family="sans-serif" space-after.optimum="3pt">
Ainsi, le 23 juin 1969 un tonneau d'un insecticide, l'endosulfan, tomb&#233; dans le
Rhin en aval de Bingen, pollua ce fleuve sur 600 kilom&#232;tres, faisant p&#233;rir
plus de 20 millions de poissons selon certaines estimations.
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

XSL-FO : principes de base

1. Déclaration des formats des familles pages.

```
<fo:layout-master-set>
 <fo:simple-page-master master-name="noheaders_center"
    page-height="21.7cm"
    page-width="16cm"
    margin-top="1cm"
    margin-bottom="1cm"
    margin-left="2.5cm"
    margin-right="2.5cm">
        <fo:region-body margin-top="2cm" margin-bottom="2cm"/>
    </fo:simple-page-master>

 <fo:simple-page-master master-name="headers_right"
    page-height="21.7cm"
    page-width="16cm"
    margin-top="1cm"
    margin-bottom="1cm"
    margin-left="3.5cm"
    margin-right="1.5cm">
        <fo:region-body margin-top="2cm" margin-bottom="2cm"/>
        <fo:region-before extent="2cm"/>
        <fo:region-after extent="2cm"/>
    </fo:simple-page-master>
```

XSL-FO : principes de base

2. Déclaration de l'ordonnancement des familles de pages.

```
<fo:page-sequence-master master-name="run1">
    <fo:repeatable-page-master-reference
        master-reference="noheaders_center"
        maximum-repeats="6"/>
</fo:page-sequence-master>

<fo:page-sequence-master master-name="run2">
    <fo:repeatable-page-master-alternatives>
        <fo:conditional-page-master-reference
            master-reference="headers_center"
            page-position="first" />
        <fo:conditional-page-master-reference
            master-reference="headers_right"
            odd-or-even="odd" />
        <fo:conditional-page-master-reference
            master-reference="headers_left"
            odd-or-even="even" />
    </fo:repeatable-page-master-alternatives>
</fo:page-sequence-master>
```

XSL-FO : principes de base

3. Déclaration de l'organisation de chaque famille de page.

```
<fo:page-sequence master-reference="alternating" initial-page-number="1">
    <fo:static-content flow-name="xsl-region-before">
        <fo:block text-align="start" font-size="10pt"
            font-family="sans-serif"
            line-height="10pt" >
            John Franklin - Journey to the Shores of the Polar Sea
        </fo:block>
  </fo:static-content>

  <fo:static-content flow-name="xsl-region-after">
    <fo:block text-align="end"
        font-size="10pt"
        font-family="sans-serif"
        line-height="14pt" >
      p. <fo:page-number/>
    </fo:block>
  </fo:static-content>

  <fo:flow flow-name="xsl-region-body">
```

XSL-FO : principes de base

4. Remplissage des pages avec le contenu.

```
<fo:flow flow-name="xsl-region-body">

  <fo:block text-align="center">
    <fo:external-graphic
      src="/usr/local/apache/htdocs/img/bando.jpg"
      width="537px"/>
    </fo:block>

  <fo:table table-layout="fixed"
     border-style="solid" border-color="#000000" border-width="0.3pt"
     space-before.optimum="12pt">
    <fo:table-column />
    <fo:table-body>
      <fo:table-row>
        <fo:table-cell padding="1mm">
          <fo:block text-align="center"
            font-size="16pt" font-family="Times Roman">Ici le texte</fo:block>
        </fo:table-cell>
      </fo:table-row>
    </fo:table-body>
  </fo:table>
</fo:flow>
```

XSL-FO : Exercice

Créer une page fo et utilisez FOP pour en faire un PDF.

Créer une feuille XSLT qui génère du XSL-FO à partir du fichier XML.

C'est fini ?

Pour moi c'est fini, je vous passe le bouchon !!

- Mise en oeuvre de projets utilisant XML/XSL.
- Le web est sa propre école.
- Auto-formation.
- Marketing interne.
- Justifier l'investissement ;)

Merci de votre accueil et de votre participation.