Server-side XML

Code: xml-ser

PAS FINI .... mais ok pour un début

Originaux

url: **http://tecfa.unige.ch/guides/tie/html/xml-ser/xml-ser.html**
url: **http://tecfa.unige.ch/guides/tie/pdf/files/xml-ser.pdf**

Auteurs et version

- Daniel K. Schneider

- Version: 0.9 (modifié le 6/6/00 par DKS)

Prérequis: Java de base, servlets, GUI et XML

Module technique précédent: xml-dom (utile pour néophytes)
Module technique précédent: xml-tech (matière obligatoire !)
Module technique précédent: xml-xslt (matière obligatoire !)
Module technique suppl.: java-intro (pour XSP)

Module technique suppl.: java-jsp (pour XSP)
Module technique suppl.: java-servl (pour XSP)

Objectifs

- Server-side XML de base
- Off-line XML vers HTML
- Server-side XML avancé

1. Table des matières détaillée

2. Introduction

- Il s'agit d'un domaine nouveau

Rien n'est sûr, à part les standards XML et XSLT

- Certaines technologies simples (comme les servlets XSLT sont solides)
- Certains projets (comme Cocoon) nécessitent d'abord une expérimentation
  - ils véhiculent une nouvelle approche pour le Web-publishing (pas seulement des nouvelles technologies pour stocker/servir de l'information)

[manque un overview ici ... à faire rapidement en principe !]

3. Simple XML + XSLT vers HTML

- On utilise des programmes Java à Tecfa, il existe d'autres possibilités qui ne sont pas mentionnées ici.
- Pour XSLT, voir le module xml-xslt.

Important: Il faut utiliser une définition correcte de stylesheets.

- La version officielle et qui marche avec les processeurs XSL récents est:

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

et non pas:

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

.... problème no #1 mentionné dans les forums quand le processeur XSL refuse de coopérer

3.1 Produire du html statique avec XT/XP sous Windows

- Actuellement la stratégie la plus simple (parmi les gratuites) est d'utiliser les outils de James Clark.

1. Téléchargez depuis le site http://www.jclark.com/xml/xt.html la version "XT packaged as a Win32 executable" pour Windows et faites Unzip dans un répertoire (par ex: c:\Progra~1\XTWin).
2. Si, vous n'avez pas déjà Internet Explorer installé chez vous, il faut télécharger et installer "Microsoft Java VM" depuis le site:

   url: http://www.microsoft.com/java/vm/dl_vm32.htm
   - ou installer carrément Explorer bien sur
3. Vous ajoutez le "path" dans le fichier autoexec.bat soit en éditant le fichier autoexec.bat avec un éditeur, soit en tapant la commande qui suit depuis une fenêtre DOS :

```
set path=%PATH%;C:\Progra~1\XTWin
```

4. Vous exécutez XT depuis une fenêtre DOS comme suit:

   Syntaxe: xt source stylesheet result

```
par exemple: xt test.xml test.xsl test.html
```

- Note: Sous Unix et Mac il faut avoir Java installé et suivre les instructions sur le site de Clark. C'est la même solution que celle proposé dans

3.2 Les outils Java du Projet Apache

A. Les processeurs XSLT: Xalan et LotusXSL

- Xalan est le processeur XSLT du projet Apache/XML. Il est basé sur LotusXSL qui a été donné par IBM à Apache (fin 1999). LotusXSL continue d'exister, mais comme wrapper/extension à Xalan.
- Xerces, le parseur XML utilisé par Xalan est aussi un ancien produit IBM (xml4j). Note:

B. Alternatives

- Même dans le monde Java il existe de bonnes alternatives (autres implémentations de XSLT et du parseur XML)
  - XT (de James Clark qui à écrit la librarie XML qui se trouve dans PhP et Mozilla), on suggère utiliser XT pour faire du simple XML+XSLT sous Windows (voir "Produire du html statique avec XT/XP sous Windows" [6])
  - Saxon

C. A télécharger

(1)Il faut déjà un JDK (1.1x ou 2x)

(2) Un parseur XML

url: http://xml.apache.org/xerces-j/index.html

Note: d'autres parseurs (comme XT ou xml4j peuvent marcher aussi, mais c'est déconseillé)

(3) Un processeur XML, à choix:

url: http://xml.apache.org/xalan/overview.html

url: http://www.alphaworks.ibm.com/tech/lotusxsl

- Prenez LotusXML si vous pensez installer un servlet XML->HTML, sinon Xalan fait l'affaire.
- Si vous pensez installer un serveur Java (comme Tomcat) et Cocoon (XML publishing framework), sachez Xalan et Xerces sont distribubés avec Cocoon

Note pour ceux qui veulent juste obtenir les *.jar avec les classes, voir dans:

url: http://tecfa.unige.ch/guides/java/classes/

D. Installation

(1) Désarchiver/décomprimer les archives téléchargées qq part

- Il faut décomprimer les fichiers *.jar avec la commande suivante (à taper dans une fenêtre unix/dos):

```
jar tvf <fichier>           (permet de visualiser le contenu !)
jar xvf xalan_0_19_2.jar    (installe l'archive)
```

- jar est distribué avec les JDK. Si votre OS ne le trouve pas, tapez le chemin complet, ou mettez le répertoire JDKxx/bin dans votre path, out copiez juste le programme jar.exe à un endroit que le chemin trouve ......
- Important (!): *.jar est souvent utilise à la place de *.zip ou *.gz dans le monde Java. A ne pas confondre: Archives *.jar qui contiennet tout un package de soft, doc et exemple et les fichier *.jar qui contiennent juste les classes java du package.
- Jar décomprime aussi les fichiers *.zip (utilisateurs Solaris et Linus)

(2) Identifier ou se trouvent les fichiers *.jar qui contiennent les packages

Exemple (Suns/Tecfa):

```
/local/java/classes/xerces/xerces.jar
/local/java/classes/xalan/xalan.jar
```

- Vous pouvez les copier dans un endroit central ou vous garder vos *.jar

3.3 Produire du html statique avec Xalan/LotusXSL

- Cette stratégie demande un effort supplémentaire, mais elle marche partout et elle est efficace (Alternativement vous pouvez faire la même chose avec XT/XP)

(1) Installation: définir path et classpath, le shell (unix/dos/..) doit trouver:

1. java (la Java VM, par ex. celle distribuée avec les JDK de SUN)
2. les classes de xerces et xalan

- Voir le module "java-util" si nécessaire.

(2) Voici la syntaxe "ligne de commandes" de Xalan

Syntaxe: **`java org.apache.xalan.xslt.Process -IN`** foo.xml **`-XSL`** foo.xsl **`-OUT`** foo.html

Arguments supplémentaires (pleins d'autres, voir la doc !)

-VALIDATE (fait une validation, off par défaut !!)
-TEXT (output text only)

Exemple 3-1: Utilisation de Xalan en ligne de commande

- Script qui initialise le bon java/classpath sous Unix/Tecfa

```
source /local/env/java12-xalan.csh
```

- Commande (2 exemples)

```
java org.apache.xalan.xslt.Process -IN proj11.xml -XSL project.xsl -OUT test.html
java org.apache.xalan.xslt.Process -validate -IN proj11.xml -XSL project.xsl -OUT test.html
```


(3) Pour simplifier la vie il faut se faire un script qui fait (1) + (2)

Pour Unix/Tecfa, tapez:

```
xslate -IN proj11.xml -XSL project.xsl -OUT test.html
```

Pour KroSoft/Dos :

- faites un fichier xslate.bat et adaptez à vos besoins !
  - classpath pour les fichiers *.jar !

```
@echo off
rem **********************************
rem * Script ecrit par Olivier Clavel *
rem * clavelo8@etu.unige.ch           *
rem **********************************

echo **Traduction de fichier XML vers HTML en utilisant XSL**
rem - *** Attention : vous devez adapter ce classpath a votre environnement. ***
rem - dans le cas présent, les fichiers sont pris sur un drive réseau H: monté sur /comm a tecfa
     set CLASSPATH=H:\tecfa\www\guides\java\classes\xerces.jar;H:\tecfa\www\guides\java\classes\xalan.jar
rem - on fait un echo de la commande avant de l'exécuter
echo Vous executez la commande : java org.apache.xalan.xslt.Process %1 %2 %3 %4 %5 %6 %7 %8 %9
rem - on pipe java a more pour voir les instruction renvoyées si elles font plus d'une page
java org.apache.xalan.xslt.Process %1 %2 %3 %4 %5 %6 %7 %8 %9 | more
```

  - Placez ce fichier à un endroit où DOS le trouve !

3.4 XML/XSL vers HTML on the fly avec un servlet

NOTE: en ce moment 5/2000 ce service est hors usage, utilisez Cocoon à la place

- Solution très portable, mais moins efficace que la solution précédente ou encore un framework comme Cocoon, car chaque page est traduite "on the fly" à chaque requête.
- Ici on montre comme utiliser le servlet LotusXML tel qu'il est installé à Tecfa
  - Attention: pour le moment c'est cassé / Conflit entre Cocoon et LotusXSL / Conflit entre le dernier LotusXSL et JWS ....)

url: **http://tecfa2.unige.ch:8080/servlet/DefaultApplyXSL?URL=/staf/staf-e/staf18/proj/proj11.xml&xslURL=/staf/staf-e/staf18/project.xsl**

  - Enfin, une ancienne version tourne, entrez un URL (sur une seule ligne)comme:

url: **http://tecfa2.unige.ch:8080/servlet/XSLatorServlet?xml=http://tecfa.unige.ch/staf/staf-e/staf18/demo-project.xml&xsl=http://tecfa.unige.ch/staf/staf-e/staf18/project.xsl**

4. "Basics" du Cocoon Framework

A TECFA tous les fichiers *.sxml sont envoyés à Cocoon pour traitement

4.1 Principe de base

Cocoon est un "publishing framework" basé XML et écrit en Java

Philosophie de base = séparation des tâches:

1. Création de XML: Typiquement ces fichiers sont produits par des auteurs/ spécialistes de contenu avec un éditeur XML.
2. Traitement de XML: Certains vocabulaires ou tags nécessitent un traitement spécial par un "logicsheet". Imaginez un tag **<**publications name="dill"**>** qui fait de sorte à ce que toutes les publications de "dill" soient sorties d'une base de données.
3. Mise en forme (rendering de XML). Le contenu XML final sera mise en forme par une feuille de style. Actuellement surtout du HTML, mais on peut servir des contenus HTML, PDF, XML, WML, XHTML, VRML, etc. en fonction du client)

Résumé: Cocoon/XSP permet de séparer contenu, "logique" et style.

... ceci est quasi-impossible avec Php, Jsp, Asp etc.

4.2 XML + XSL simple

Principe:

Vous faites un fichier *.sxml (contenu) et un fichier *.xsl (style)

Vous devez indiquer à Cocoon comment traiter le fichier xml et comment utiliser la feuille de style dans le fichier xsl selon les règles ci-dessous

Entêtes à mettre dans les fichiers (S)XML et XSL:

Fichier XML (extension = *.sxml):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="VOTRE_FICHER_XSL.xsl" type="text/xsl"?>
<?cocoon-process type="xslt"?>
```

Fichier XSL (extension = .xsl):

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

// doit aller DANS le template pour la racine XML !!!
<xsl:template match="VOTRE_RACINE">
<xsl:processing-instruction name="cocoon-format">type="text/html"
     </xsl:processing-instruction>
......
```


Exemple 4-1: Cocoon XML + XSL examples

url: http://tecfa2.unige.ch/guides/xml/cocoon/simple/

- montre un simple exemple

```
<?xml version="1.0"?>
<?xml-stylesheet href="hello-page-html.xsl" type="text/xsl"?>
<?cocoon-process type="xslt"?>
 <title>Hello Cocoon friend</title>
 <content> ...... </content>
 <comment>  Written by DKS/Tecfa, adapted from S.M./the Cocoon samples
 </comment>
 </page>
```

hello-page-html.xsl:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="page">
   <xsl:processing-instruction name="cocoon-format">type="text/html"</
xsl:processing-instruction>
   <html> <head> <title> <xsl:value-of select="title"/> </title> </head>
    <body bgcolor="#ffffff">
     <xsl:apply-templates/>
    </body>
   </html>
  </xsl:template>
......
</xsl:stylesheet>
```


4.3 Le processeur SQL

Note: Il faut plutôt utiliser la SQL Tag library de XSP (mécanisme plus puissant)

Le processeur SQL de Cocoon reconnait des tags spécicaux pour SQL:

- Une "processing-instruction" indique à Cocoon qu'il faut les traiter:

```
<?cocoon-process type="sql"?>
```

- Définition des connexions:
  - On peut définir plusieurs `connections` dans `<connectiondefs>`
  - Chaque `<connection>` doit définir:
    l'attribut `name`,
    et les éléments: `<driver>, <dburl>`, `<username>, <password>`

```
<connectiondefs>
    <connection name="foo_connection">
       <driver>org.gjt.mm.mysql.Driver</driver>
       <dburl>jdbc:mysql://mysql.example.com/foo_database</dburl>
       <username>foo_user</username>
       <password>foo_password</password>
   </connection>
</connectiondefs>
```

- Définition d'une requête:

```
<query connection="foo_connection">
 SQL .....
</query>
```

La réponse de la base de données:

- se fait également avec un jeu de tags prédéfini (mais qu'on peut redéfinir dans les <connectiondefs>).
  - Dans notre stylesheet, on utilise notamment la balise <ROW>
  - Les entrées pour chaque ligne sont dans des tags = nom de colonnes
- Exemple:

```
<query connection="test_connection">
 select * from demo1 limit 10
</query>
```

donne:

```
<ROWSET>
<ROW id="0"><id>1</id><login>test</
login><password>098f6bcd4621d373cade4e832627b4f6</password><fullname>Tester
Test</fullname><url>http://tecfa.unige.ch</url><food>3</food><work>3</
work><love>3</love><leisure>3</leisure><sports>3</sports></ROW>
<ROW id="1"><id>34</id><login>colin2</login><password>b9hhhfa9347a11893u483</
password><fullname>Patrick Jermann2</fullname><url>http://tecfa.unige.ch/</
url><food>1</food><work>2</work><love>1</love><leisure>3</leisure><sports>4</
sports></ROW>
.....
</ROWSET>
```

- Hint: Pour voir vous-même faites un fichier XML sans style-sheet associé.

Exemple 4-2: Cocoon/SQL et MySQL demo

url: **http://tecfa2.unige.ch/guides/xml/cocoon/mysql/mysql-demo.sxml**
url: **http://tecfa2.unige.ch/guides/xml/cocoon/mysql/mysql-demo.sxml.text**

```
<?xml version="1.0"?>
<?xml-stylesheet href="mysql-demo.xsl" type="text/xsl"?>
<?cocoon-process type="sql"?>
<?cocoon-process type="xslt"?>
<page>
 <connectiondefs>
  <connection name="test_connection">
   <!-- full class name of the JDBC driver -->
   <driver>org.gjt.mm.mysql.Driver</driver>
   <!-- URN of database Syntax: jdbc:TYPE:machine:port/DB_NAME  -->
   <dburl>jdbc:mysql://tecfasun1.unige.ch/demo</dburl>
   <username>nobody</username>
   <password></password>
  </connection>
 </connectiondefs>
 <query connection="test_connection">
  select * from demo1 limit 10
 </query>
</page>
```


Extrait de http://tecfa2.unige.ch/guides/xml/cocoon/mysql/mysql-demo.xsl

```
<xsl:template match="ROW">
   <table border="2" cellspacing="0" cellpadding="6">
   <tr>
     <xsl:apply-templates select="id"/>
     <xsl:apply-templates select="login"/>
     <xsl:apply-templates select="fullname"/>
     <xsl:apply-templates select="url"/>
     <xsl:apply-templates select="food"/>
     <xsl:apply-templates select="work"/>
     <xsl:apply-templates select="love"/>
     <xsl:apply-templates select="leisure"/>
   </tr>
   </table>
</xsl:template>

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

- On met chaque ligne dans une table ici

Exemple 4-3: Cocoon/MySQL et XSL/HTML tables

url: http://tecfa.unige.ch/guides/xml/cocoon/mysql/mysql-demo2.sxml
url: http://tecfa.unige.ch/guides/xml/cocoon/mysql/mysql-demo2.sxml.text
url: http://tecfa.unige.ch/guides/xml/cocoon/mysql/mysql-demo2.xsl

Extrait du fichier style:

```
<xsl:template match="ROWSET">
    <table border="2" cellspacing="1" cellpadding="6">
     ....
     <xsl:for-each select="ROW">
      <tr>
      <td><xsl:value-of select="id"/></td>
      <td><xsl:value-of select="login"/></td>
      <td><xsl:value-of select="fullname"/></td>
      <td bgcolor="tan">
        <a href="{url}"><xsl:value-of select="url"/></a>
      </td>
      <td><xsl:value-of select="food"/></td>
      <td><xsl:value-of select="work"/></td>
      <td><xsl:value-of select="love"/></td>
      <td><xsl:value-of select="leisure"/></td>
      </tr>
     </xsl:for-each>
    </table>
</xsl:template>
```

- Note: pour générer les cellules vides, il faut se lever plut tôt ... à suivre

4.4 LDAP Interface

- Même principe que pour SQL
  url: **http://tecfa2.unige.ch/guides/xml/cocoon/ldap/ldap.sxml**
  url: **http://tecfa2.unige.ch/guides/xml/cocoon/ldap/ldap.sxml.text**
  url: **http://tecfa2.unige.ch/guides/xml/cocoon/ldap/ldap.xsl**

```
<?xml version="1.0"?>
<?xml-stylesheet href="ldap.xsl" type="text/xsl"?>
<?cocoon-process type="ldap"?>
<?cocoon-process type="xslt"?>
<page>
  <ldap-defs>
    <ldap-server name="tecfa">
      <initializer>com.sun.jndi.ldap.LdapCtxFactory</initializer>
      <ldap-serverurl>ldap://tecfa2.unige.ch:389</ldap-serverurl>
    </ldap-server>
    <ldap-querydefs name="standard" default="yes"/>
  </ldap-defs>
     <ldap-query  server="tecfa"  ldap-searchbase="o=tecfa.unige.ch"
   defs="standard">
     givenname=Daniel
  </ldap-query>
</page>
```


5. Cocoon - XSP

- XSP = eXtensible Server Pages
- Une alternative à Php / JSP dans certains cas
- Fonctionne un peu près comme JSP (page compilation)

Exemple 5-1: Date

url: http://tecfa2.unige.ch/guides/xml/cocoon/xsp/xsp-date.sxml

```
<?cocoon-process type="xsp"?>
<?cocoon-process type="xslt"?>
<?xml-stylesheet href="simple-page-html.xsl" type="text/xsl"?>
<xsp:page language="java" xmlns:xsp="http://www.apache.org/1999/XSP/Core"
>
    <xsp:logic>
      // Define a variable to hold the time of day
      Date now = new Date();
    </xsp:logic>
  <page>
    <title>Time of Day</title>
    <p>   To the best of my knowledge, it's now about
      <!-- Substitute time of day here -->
      <xsp:expr>now</xsp:expr>    </p>
      </page>
</xsp:page>
```


5.1 Anatomie d'une simple page XSP

```
?cocoon-process type="xsp"?>
<?cocoon-process type="xslt"?>
<?xml-stylesheet href="VOTRE_SHEET.xsl" type="text/xsl"?>

<xsp:page language="java" xmlns:xsp="http://www.apache.org/1999/XSP/
   Core">

 <xsp:logic>
 du code Java (comme les scriptlets <% %> en JSP
 </xsp:logic>

<xsp:expr>
  une simple expression Java  (comme les <%= %> en JSP )
</xsp:expr>

</xsp:page>
```


5.2 Traitement de formulaires (GET/POST)

Exemple 5-2: Simple calcul

- Il s'agit ici de l'exemple canonique que DKS utilise partout (php, js, java ...) pour montrer comment traiter un formulaire
  url: **http://tecfa.unige.ch/guides/xml/cocoon/simple-calcul/form.html (formulaire)**
  url: **http://tecfa.unige.ch/guides/xml/cocoon/simple-calcul/calcul.sxml.text (source XSP)**

```
<xsp:page language="java" xmlns:xsp="http://www.apache.org/1999/XSP/Core">
    <page>
    <xsp:logic>
    String choice = request.getParameter("choice");
    String choice2 = request.getParameter("choice2");
    int score = 0;

    if ((choice == null) || (choice2 == null)) {
        <xsp:content>Please use the <a href="form.html">form</a>
                 </xsp:content>
                 return;
    } else
        score = Integer.parseInt(choice) + Integer.parseInt(choice2);
    </xsp:logic>

        <title>Form - XSP Demo </title>

        <content>
        You entered <xsp:expr>choice</xsp:expr> and
```

```
          <xsp:expr>choice2</xsp:expr>.
          That makes <xsp:expr>score</xsp:expr>.
          </content>

          <comment>
          Made by Daniel.Schneider@tecfa.unige.ch (3/99)
          </comment>

      </page>
  </xsp:page>
```

- explications à suivre, pour le moment voir la doc servlet ou JSP pour cet exemple ....

6. A la main avec Java like real Women

Outils

- Il vous vaut un serveur Java et un parseur XML comme Xerces
- Alternativement: Php, des modules cgi écrits en Perl, Python, etc.

Le principe:

- analyser la structure d'un fichier XML
- extraire les éléments qui vous intéressent
- transformations et calculs etc.
- rendering en HTML

Il existe 2 type des parseurs:

- DOM (traduit un fichier XML sous forme d'arbre informatique)
- SAX (identifie et crache élément par élément)

... voir les modules Java-XML **(à refaire) ou Php-XML (à faire)**