CSI 3540

Structures, techniques et normes du Web

Représentation des données du Web en XML

Objectifs :

- Maîtriser le langage XML 1.0
- Savoir utiliser les outils standard pour le traitement de XML

Lectures :

- Web Technologies (2007) § 7 Pages 379–402

Plan

1. Processeurs XML
    1. DOM
    2. SAX
2. Transformations
    1. XPath
    2. XSL
    3. XSLT

Introduction

- Pourquoi revisite-t-on **XML** ?
- **XML** est la base de technologies telles que :
  - **AJAX** (Asynchronous JavaScript and XML)
  - **SOAP** (Simple Object Access Protocol)
  - ...
  - **XSL**, **XPath**, **XSLT**, **Schema**, **WSDL**

<oXygen/> XML Editor

- "**<oXygen/>** is a complete cross platform XML editor providing the tools for **XML** authoring, XML conversion, **XML Schema**, **DTD**, **Relax NG** and **Schematron** development, **XPath**, **XSLT**, **XQuery** debugging, **SOAP** and **WSDL** testing."

- www.oxygenxml.com

Oxygen File Edit Find Project Perspective Options Tools Document Window Help 13:19

<oXygen/> - [/Users/turcotte/iDisk/teaching/csi3540/2008/resources/CellarBook/WineCatalog.xsl]

External Tools LIBXML

XPath 2.0

Project

sample.xpr
- css
- debugger
- dita
- docbook
- fo
- import
- jsp
- nvdl
- relaxng
- schematron
- schematronISO
- svg
- tei
- wsdl
- xhtml
- xquery
- personal-schema.css
- personal-schema.xml
- personal.css
- personal.dtd
- personal.xml
- personal.xsd
- personal.xsl
- simpleLayoutSample.xpr

Outline

- xsl:stylesheet "http://www.w3.org/1999
  - xsl:param "color"
  - <!-- to produce legal and validable
  - xsl:output "xml"
  - xsl:template "/cat:wine-catalog"
  - xsl:template "cat:wine"
  - xsl:function "cat:toML"
  - xsl:template "toL"

WineCatalog.xml | WineCatalog.xsl

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2
3  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
4      version="2.0"
5      xmlns:cat="http://www.iro.umontreal.ca/lapalme/wine-catalog">
6      <xsl:param name="color" select="'red'"/>
7      <!-- to produce legal and validable XHTML ... -->
8      <xsl:output method = "xml"
9          doctype-public = "-//W3C//DTD XHTML 1.0 Strict//EN"
10         doctype-system =
11            "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"
12         indent = "yes" encoding = "UTF-8"/>
13
14     <xsl:template match="/cat:wine-catalog">
15         <html>
16             <head><title>Wine Catalog</title></head>
17             <body>
18                 <h1>
19                 Wine Catalog (<xsl:value-of select="$color"/> only)
20                 </h1>
21                 <table border="1">
22                     <tr>
23                         <th width="200">Wine Name</th>
24                         <th>Code</th>
25                         <th>Color</th>
26                         <th>Year</th>
27                         <th>Price</th>
28                         <th>ml</th>
29                         <th>l</th>
30                     </tr>
31                     <xsl:apply-templates
32                     select="cat:wine[cat:properties/cat:color=$color]"/>
33                 </table>
34             </body>
35         </html>
36 </xsl:template>
37
38 <xsl:template match="cat:wine">
39     <tr>
40         <td>
41             <xsl:value-of select="@name"/>
42         </td>
43         <td>
44             <xsl:value-of select="@code"/>
45         </td>
46         <td>
47             <xsl:value-of select="cat:properties/cat:color"/>
48         </td>
49         <td align="right">
50             <xsl:value-of select="cat:year"/>
51         </td>
52         <td align="right">
53             <xsl:value-of select="format-number(cat:price,'$0.00')"/>
54         </td>
```

Text Grid

Attributes

xsl:template
http://www.w3.org/1999/XSL/Transfo

Attribute	Value
as	item()*
default-collation	
exclude-result-pr...	
extension-element...	
match	/cat:wine-catalog
mode	
name	
priority	
use-when	
version	
xml:space	
xpath-default-na...	

Attributes | Model | Styleshee..

Elements

- a
- abbr
- acronym
- address
- applet
- area
- b
- base
- basefont
- bdo
- big
- blockquote
- body
- br
- button
- caption
- center
- cite
- code
- col
- colgroup
- dd
- del
- dfn

<x> Ele.. | Entit.. | XSL.. | Sc..

/Users/turcotte/iDisk/teaching/csi3540/2008/resources/CellarBook/WineCatalog.xsl U+0073 14:7

Oxygen File Edit Find Project Perspective Options Tools Document Window Help 13:22

<oXygen/> - [/Bio/Sfw/oxygen/samples/personal.xsd]

External Tools SaxonSA

XPath 2.0

Project

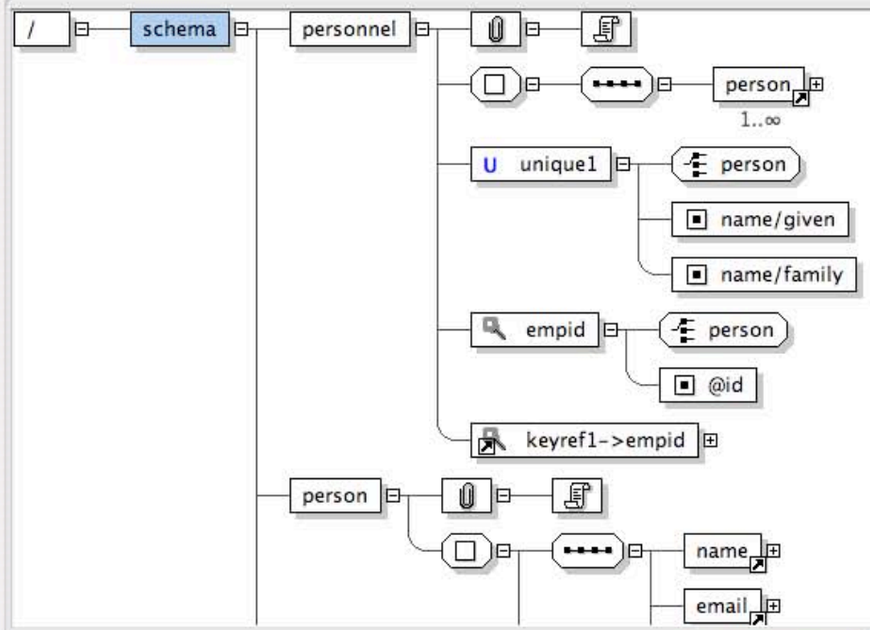
- sample.xpr
- css
- debugger
- dita
- docbook
- fo
- import
- jsp
- nvdl
- relaxng
- schematron
- schematronISO
- svg
- tei
- wsdl
- xhtml
- xquery
- personal-schema.css
- personal-schema.xml
- personal.css
- personal.dtd
- personal.xml
- personal.xsd
- personal.xsl
- simpleLayoutSample.xpr

Outline

- xs:schema "http://www.w3.org/2001/X
  - xs:element "personnel"
  - xs:element "person"
  - xs:element "name"
  - xs:element "family"
  - xs:element "given"
  - xs:element "email"
  - xs:element "url"
  - xs:element "link"
  - xs:notation "gif"

WineCatalog.xml | WineCatalog.xsl | personal.xsd

100



Full Model View | Logical Model View

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3
4      <xs:element name="personnel">
5          <xs:annotation>
6              <xs:documentation>Defines the personnel as a collection of person elements.</xs:documentation>
7          </xs:annotation>
8          <xs:complexType>
9              <xs:sequence>
10                 <xs:element ref="person" minOccurs="1" maxOccurs="unbounded"/>
11             </xs:sequence>
12         </xs:complexType>
13         <xs:unique name="unique1">
14             <xs:selector xpath="person"/>
15             <xs:field xpath="name/given"/>
16             <xs:field xpath="name/family"/>
17         </xs:unique>
18         <xs:key name="empid">
19             <xs:selector xpath="person"/>
20             <xs:field xpath="@id"/>
21         </xs:key>
22         <xs:keyref name="keyref1" refer="empid">
23             <xs:selector xpath="person"/>
```

Text | Grid

Attributes

xs:schema
http://www.w3.org/2001/XMLSchema

Attribute	Value
attributeFormDefault	unqualified
blockDefault	
elementFormDefault	unqualified
finalDefault	
id	
targetNamespace	
version	
xml:lang	
xmlns:xs	http://www.w3.or...

Attributes | Model | Styleshee..

Elements

xs:schema#http://www.w3.org/2001/X

(x) Ele.. | Entit.. | XSL.. | Sc..

/Bio/Sfw/oxygen/samples/personal.xsd U+003C 1:1

Friday, March 19, 2010

XML côté serveur

Analyse

- Un **processeur** est un module logiciel utilisé afin de lire et accéder au contenu d'un document XML
- Il y a deux types de processeurs : **validateur** et **non-validateur**

Processeur validateur

- **Exige et lit le DTD**
- S'assure que

  1. le document est un document XML bien formé
  2. conforme au DTD
  3. ainsi qu'aux contraintes de validité (les ID sont uniques, par exemple)

«In particular, software does not usually need to fetch these resources, and certainly does not need to fetch the same one over and over! Yet we receive a surprisingly large number of requests for such resources: up to **130 million requests per day**, with periods of sustained bandwidth usage of **350 Mbps**, for resources that haven't changed in years. The vast majority of these requests are from systems that are processing various types of markup (HTML, XML, XSLT, SVG) and in the process doing something like validating against a DTD or schema. Handling all these requests costs us considerably: servers, bandwidth and human time spent analyzing traffic patterns and devising methods to limit or block excessive new request patterns. We would much rather use these assets elsewhere, for example improving the software and services needed by W3C and the Web Community.»
http://www.w3.org/blog/systeam/2008/02/08/w3c_s_excessive_dtd_traffic [9 février 2008]

Laboratoire 9 : The Cloak

java.io.IOException: Server returned HTTP response code: 503 for URL: http://www.w3.org/TR/xhtml-modularization/DTD/xhtml-inlstyle-1.mod

dbf.setFeature("http://apache.org/xml/features/nonvalidating/load-external-dtd", false);

Processeur validateur

- **Interchangeable**
  (produisent tous les mêmes résultats)

Processeur non-validateur

- S'assure que le document est un document XML bien formé
- N'est pas tenu de lire le DTD (externe) ; peut en faire une lecture partielle, complète ou pas du tout
- Différents processeurs peuvent produirent différents résultats
- Ils sont rapides

DTD partiel

- Certains éléments du DTD, jugés **importants**, peuvent être inclus dans le document XML

```
<!DOCTYPE rss
          SYSTEM "http://my.netscape.com/published/formats/rss-0.91.dtd"
[
<!ENTITY vsn "0.91">
]
>
<rss version="&vsn;">
```

Sous-ensemble interne du DTD

- Le sous-ensemble interne est lu par les deux classes de processeurs

Plusieurs approches en Java

1. **DOM** (**Document Object Model**)
2. **SAX** (**Simple API for XML**)

Quelles sont les différences majeures entre ces deux approches, DOM et SAX?

Modèle objet de document (DOM)

- **javax.xml.parsers** et **org.w3c.dom** sont distribués avec l'environnement de développement JSE
- Le document est lu afin de produire un **arbre d'analyse** (demande beaucoup de resources, c'est l'un des désavantages de cette approche)

Modèle objet de document (DOM)

- On accède, ou modifie, le contenu et la structure du document à l'aide d'une interface **API semblable à celui de JavaScript/DOM**

- **org.w3c.dom : Document, Node, NodeList, Element, Attribute, Text , Attr**

- **JavaScript** : var parent = node.parentNode;

- **Java** : Node parent = node.getParentNode();

DOM : Processeur XML

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;

DocumentBuilder parser = DocumentBuilderFactory.newInstance().newDocumentBuilder();
```

- Le processeur par défaut est non-validateur et ne tient pas compte des espaces de nommages
- **setValidating(true)**, **setNamespaceAware(true)**
- L'objet **DocumentBuilder** sert aussi bien à l'analyse syntaxique qu'à la creation de nouveau documents

Factory et plug-in

- Le processeur est déterminé de façon externe au programme Java

```
> java -Djava.xml.parsers.DocumentBuilderFactory=gnu.xml.dom.JAXPFactory DOMCountLinks fichier.xml
```

DOM : Document XML

```
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;

Document parser;
```

DOM : Traitement d'un Document XML

```
document = parser.parse( fichier );
NodeList links = document.getElementsByTagName( "link" );

for ( int i=0; i<links.getLength(); i++ ) {
    Node elem = links.item( i );
    ...
}
```

Espaces de nommage

- La bibliothèque de méthodes supporte les espaces de nommage
- getElementsByTagNameNS(String, String)
- Si le document ne déclare aucun espace de nommage, alors utilisez **null**
- getElementsByTagNameNS(null, "link")
- createElementNS(String, String)

DOM : Validateur ou pas

```
document = parser.parse( fichier );
NodeList links;
links = document.getElementsByTagNameNS( "http://www.w3.org/1999/xhtml", "a" );

for ( int i=0; i<links.getLength(); i++ ) {
    Node elem = links.item( i );
    ...
}
```

```
document = parser.parse( fichier );
NodeList links = document.getElementsByTagName( "a" );

for ( int i=0; i<links.getLength(); i++ ) {
    Node elem = links.item( i );
    ...
}
```

DOM : Avantages

- La structure du document est représentée en mémoire, on peut donc **traverser** le document, en **transformer la structure**, et possiblement **sauvegarder** la structure transformée

DOM : désavantages

- La structure du document est représentée en mémoire, les processeurs DOM **utilisent donc beaucoup de ressources** : mémoire et temps de calcul

SAX (Simple API for XML)

- Avec le **DOM**, le document est lu en entier et les traitements se font sur l'**arbre d'analyse** (c'est couteux en temps + mémoire)

- Avec **SAX**, des **gestionnaires sont associés aux événements** tels que la lecture d'une balise d'ouverture/ fermeture ou du contenu

SAX : Processeur XML

```
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.SAXParser;

XMLReader parser = SAXParserFactory.newInstance().newSAXParser().getXMLReader();
parser.setContentHandler( new CountElementsHandler() );
parser.parse( fichier );
```

SAX : Traitement

```
private static class CountElementsHandler extends DefaultHandler {
  private int numElements;
  public void startDocument() throws SAXException {
    numElements = 0;
    return;
  }
  public void startElement( String namespaceURI, String localName,
                            String qName, Attributes atts )
    throws SAXException
  {
    if ( qName.equals( "link" ) ) {
      numElements++;
    }
    return;
  }
  public void endDocument() throws SAXException {
    System.out.println( "Input document has " + numElements + " 'link' elements." );
    return;
  }
}
```

Attributs

- Le paramètre **attr** de la méthode **startElement** donne accès aux éléments

- L'accès peut aussi bien se faire **par position** que **par nom**

```
attr.getValue( 1 )
attr.getValue( "href" )
```

characters

- La méthode **characters** est appelée lorsque l'analyseur rencontre des données qui ne sont pas des balises
- **characters(char[] ch, int start, int len)**
- Les paramètres **start** et **len** indique la portion du tableau à traiter

SAX : Traitement des données

```
private static class PrintElementsHandler extends DefaultHandler {

    private boolean inLink = false;
    private StringBuffer charData;

    public void startElement( String nsURI, String localName, String qName, Attributes atts) throws SAXException {
        if ( qName.equals( "link" ) ) {
            inLink = true;
            charData = new StringBuffer();
        }
        return;
    }

    public void characters( char chars[], int firstChar, int nChars ) throws SAXException {
        if ( inLink ) {
            charData.append( chars, firstChar, nChars );
        }
        return;
    }

    public void endElement( String nsURI, String local, String qName ) throws SAXException {
        if ( qName.equals( "link" ) ) {
            System.out.println( "Link data: " + charData.toString() );
            inLink = false;
        }
        return;
    }
}
```

characters

- **characters(char[] ch, int start, int len)**
- Il se peut que les données textuelles d'un élément soient passées au gestionnaire à la suite de plusieurs appels
- C'est le cas si les données textuelles contiennent des appels d'entités

```
<?xml version="1.0"?>
<!DOCTYPE rss
 [
<!ENTITY  eacute "é">
 ]
 >
<rss version="0.91">
  <channel>

    <title>Un blogue</title>
    <link>http://bio.site.uottawa.ca/start/wiki/</link>
    <description>
      Activit&eacute;s bioinformatiques...
    </description>
    <language>fr-CA</language>

    <item>
      <title>Pub pour CSI5126</title>
      <link>http://www.site.uottawa.ca/~turcotte/teachine/csi-5126/</link>
      <description>
        Concepts math&eacute;matiques et algorithmiques fondamentaux
        de la biologie mol&eacute;culaire computationnelle.
      </description>
    </item>

  </channel>
</rss>
```

SAX : Traitement des données

```
> java SAXPrintDescriptions bio.xml

Description data: [
    Activit][é][s bioinformatiques...][
  ]
Description data: [
      Concepts math][é][matiques et algorithmiques fondamentaux]
[
      de la biologie mol][é][culaire computationnelle.][
    ]
```

SAX : Traitement des erreurs

- Lorsque le processeur trouve une erreur, il ne lance pas une exception
- Il passe l'exception à la méthode error() du gestionnaire (handler)

```
public void error( SAXParseException e )
throws SAXParseException;
```

SAX : Traitement des erreurs

- La méthode par défaut ne fait rien

```
public void error( SAXParseException e )
throws SAXParseException {
}
```

- Il suffit de redéfinir cette méthode comme suit afin d'obtenir les erreurs

```
public void error( SAXParseException e )
throws SAXParseException {
    throw e;
}
```

SAX : Traitement des erreurs

- L'exception **SAXException** définie `getLineNumer()` et `getColumnNumer()`
- **SAXException** est parfois l'enveloppe d'une autre exception, si `getException()` retourne une valeur non **null**, cet objet renferme des informations pertinentes afin de trouver la source de l'erreur
- `SAXException( String msg, Exception cause )`

Résumé

- En **Java**, il y a deux grandes approches pour traiter un document **XML** : **DOM** et **SAX**
- **DOM** nécessite la construction d'un arbre d'analyse (couteux en temps et mémoire)
- **SAX** propose une approche orientée événement au traitement des documents XML
- DOM en mode développement, SAX en mode production, par exemple

Transformations

TransformerFactory, XSL, XSLT

Transformations

- ✓ XML → DOM (DocumentBuilderFactory)
- ✓ XML → Flux d'événements (SAXParserFactory)
- • DOM → XML (TransformerFactory)
- • XML → XML (XSLT)

TransformerFactory

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
                       "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
  <dict>
    <key>couleur</key>
    <string>rouge</string>
  </dict>
</plist>
```

TransformerFactory

```
import org.w3c.dom.*;

import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.ParserConfigurationException;

import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

public class DocMaker {

   public static void main( String[] args )
      throws ParserConfigurationException,
             TransformerConfigurationException, TransformerException {

      ...

   }
}
```

TransformerFactory

```
DocumentBuilder builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
Document doc = builder.newDocument();

Element key = doc.createElement( "key" );
key.appendChild( doc.createTextNode( "couleur" ) );

Element string = doc.createElement( "string" );
string.appendChild( doc.createTextNode( "rouge" ) );

Element dict = doc.createElement( "dict" );
dict.appendChild( key );
dict.appendChild( string );

Element plist = doc.createElement( "plist" );
plist.setAttribute( "version", "1.0" );
plist.appendChild( dict );

doc.appendChild( plist );
```

TransformerFactory

```
Transformer transformer = TransformerFactory.newInstance().newTransformer();

String PUBLIC = "-//Apple Computer//DTD PLIST 1.0//EN";
String SYSTEM = "http://www.apple.com/DTDs/PropertyList-1.0.dtd";

transformer.setOutputProperty( OutputKeys.DOCTYPE_SYSTEM, SYSTEM );
transformer.setOutputProperty( OutputKeys.DOCTYPE_PUBLIC, PUBLIC );
transformer.setOutputProperty( OutputKeys.INDENT, "yes" );

transformer.transform( new DOMSource( doc ), new StreamResult( System.out ) );
```

TransformerFactory

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
                       "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
  <dict>
    <key>couleur</key>
    <string>rouge</string>
  </dict>
</plist>
```

Extensible Stylesheet Language – **XSL**

- **XML → XML**
- **XSL** est une application/un **vocabulaire XML**
- Un document XSL est donc un document XML bien formé
- **Modèle** (template) **+ balises de contrôle**

XSL : exemple du compteur

- Une application encode la valeur d'un compteur à l'aide de XML comme suit (index.xml) :

```
<?xml version='1.0' encoding='UTF-8'?>
<info>1</info>
```

XSL : exemple du compteur

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform version="1.0"
               xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
               xmlns="http://www.w3.org/1999/xhtml">
  <xsl:template match="/">
    <html>
      <head>
        <title>
          Transformation XML vers XHTML
        </title>
      </head>
      <body>
        <p><xsl:value-of select="child::info" /></p>
      </body>
    </html>
  </xsl:template>
</xsl:transform>
```

2 espaces de nommage

xhtml est celui par défaut

modèle

énoncé du langage **XSL**

expressions **XPath**

XSL : exemple du compteur

```
import java.io.File;
import javax.xml.transform.*;
import javax.xml.transform.stream.StreamSource;
import javax.xml.transform.stream.StreamResult;

public class XSLTransformer {

    public static void main( String[] args )
        throws TransformerConfigurationException, TransformerException {

        StreamSource tfo = new StreamSource( new File( "index.xsl" ) );
        StreamSource fin = new StreamSource( new File( "index.xml" ) );
        StreamResult fou = new StreamResult( new File( "index.html" ) );

⇨       Transformer transformer = TransformerFactory.newInstance().newTransformer( tfo );

⇨       transformer.transform(  fin, fou );

    }
}
```

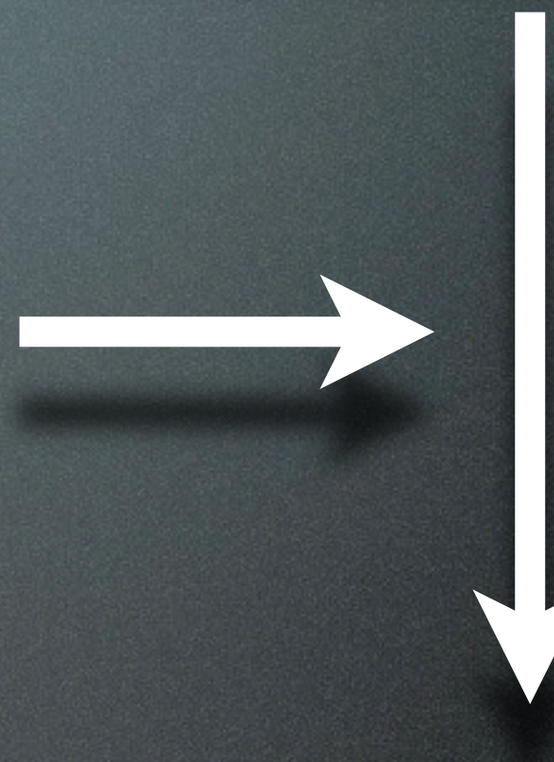
> java XSLTransformer

XSL : exemple du compteur

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Transformation XML vers XHTML</title>
  </head>
  <body>
    <p>1</p>
  </body>
</html>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<info>1</info>
```

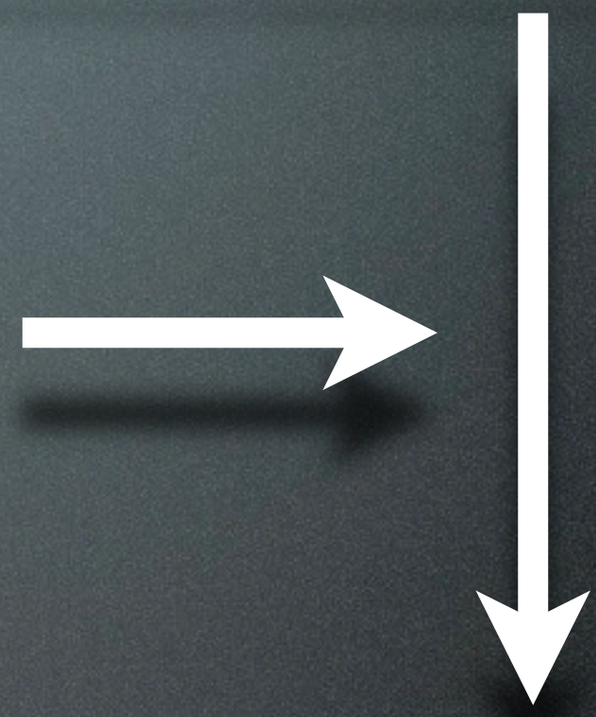
```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns="http://www.w3.org/1999/xhtml">
  <xsl:template match="/">
    <html>
      <head>
        <title>Transformation XML vers XHTML</title>
      </head>
      <body>
        <p><xsl:value-of select="child::info" /></p>
      </body>
    </html>
  </xsl:template>
</xsl:transform>
```

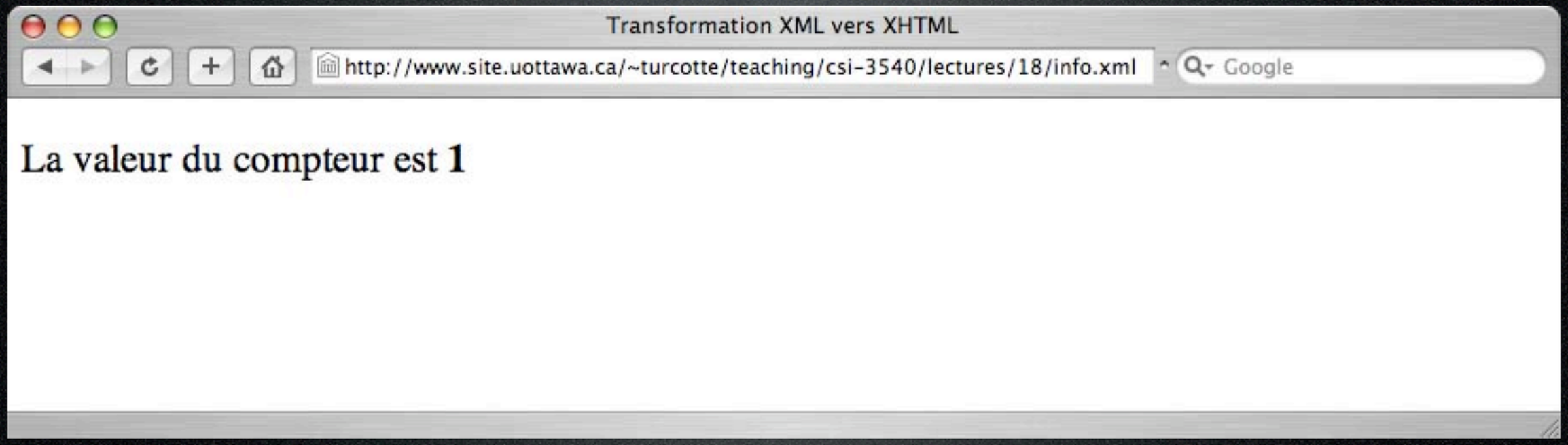


```xml
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Transformation XML vers XHTML</title>
  </head>
  <body>
    <p>1</p>
  </body>
</html>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/
Transform"
xmlns="http://www.w3.org/1999/xhtml">
  <xsl:template match="/">
    <html>
      <head>
        <title>Transformation XML vers XHTML</title>
      </head>
      <body>
        <p>
          La valeur du compteur est
          <b><xsl:value-of select="child::info" /></b>
        </p>
      </body>
    </html>
  </xsl:template>
</xsl:transform>
```

```xml
<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type="text/xsl" href="info.xsl"?>
<info>1</info>
```





XSL – Extensible Stylesheet Language

- **XSL** c'est **3** recommandations W3C :

  1. XSL Transformations (XSLT)
  2. **XML Path language (XPath)**
  3. XSL Formatting Objects (XSL-FO)

Peu utilisé sur le Web

Surtout utiliser afin de produire des tranformations pour l'impression (PDF par exemple)

Revue des concepts

- Deux grandes approches à l'analyse de documents XML : **DOM** et **SAX**
- **JAXP** permet toutes les transformations possible entre XML, DOM et SAX

Revue des concepts

- **XSL** est langage pour exprimer des transformations de documents XML, de XML vers XML, mais aussi XML vers une représentation libre

- **XSL** est défini par 3 recommendations : XSLT, XPath et XSL-FO

- **XPath** est un langage commun à plusieurs applications pour l'adressage de noeuds dans les documents XML

Ressources

- Langage de balisage extensible (XML) 1.0 [http://pages.videotron.com/fyergeau/w3c/xml10/REC-xml-19980210.fr.html] 2007
- Les espaces de nommage dans XML 1.1 [http://www.yoyodesign.org/doc/w3c/xml-names11] 2007

Ressources (suite)

- Java API for XML Processing (JAXP) Specification 1.3 [http://java.sun.com/xml/downloads/jaxp.html] 2007
- SAX [http://www.saxproject.org] 2007
- <oXygen/> XML Editor & XSLT Debugger [http://www.oxygenxml.com] 2007

Ressources (suite)

- The GNU JAXP Project [http://www.gnu.org/software/classpathx/jaxp/jaxp.html] 2007
- Langage XML Path (XPath) Version 1.0 [http://xmlfr.org/w3c/TR/xpath] 2007
- Transformations XSL (XSLT) [http://xmlfr.org/w3c/TR/xslt/] 2007
- XML: Looking at the Forest Instead of the Trees par Guy Lapalme [http://www.iro.umontreal.ca/~lapalme/ForestInsteadOfTheTrees/] 2007