par Philippe Poulard

Processeurs XML

INRIA SOPHIA ANTIPOLIS
Philippe.Poulard@sophia.inria.fr

Sommaire

Processeurs XML

Processeurs généraliste / spécialiste
Langages, architectures, plate-formes
MSXML2
Ilôts de données avec MSXML2
Implémentation et utilisation de SAX
Applications Web distribuées
Configuration des parsers avec JAXP
Composants de JAXP
Création d'un document par programmation
Prise en compte des espaces de nommage
Le goût du blanc
Validation

Patrons de programmation

Xerces : DOM, SAX
JAXP : DOM, SAX
MSXML2 : DOM, SAX

Programmation avancée

Error handling
Entity resolver
Sérialisation

XSLT

XSLT dans JAXP
Sérialisation
- avec XSLT
- d 'un flux SAX modifié

Architecture dans JAXP et Xalan
XPath dans Xalan
XSLT côté serveur
Passage de paramètres avec XSLT (JAXP, MSXML2)
Extensions, scripting

Choix d'un parseur
Autre processeurs : spécialisés, propriétaires
Conception d'un processeur
XML Data Binding

Quelques processeurs XML

Généralistes

Parsers
Processeurs XSLT

Spécialistes

Processeur XSLFO
Processeur SVG
Processeur XSP
.../...
Processeur applicatif ⟶ Processeur "maison"

Microsoft
- MSXML
- MSXML2.4.x ⟵ DOM, SAX, XSLT

XML Apache Group
- Xalan ⟵ XSLT, XPath
- Xerces ⟵ DOM, SAX

James Clark's
- XP

Megginson
- SAX

Java
- JAXP ⟵ API

Quels langages / architectures ?

Java
C
C++
C#
VB
Perl
JavaScript
VBScript
PHP
Python

Plates-forme

Serveur Web
- Apache → PHP, Perl, CGI
- Tomcat → JAXP, Cocoon
- IIS → DCOM

Serveur d'application
- J2EE → JAXP
- .NET → DCOM

Client Web
- Mozilla → natif
- IE → natif (ActiveX)
- Applet (Java)

Applications

Et aussi :
le SDK, la documentation très complète de Microsoft

Microsoft

MSXML ← Obsolète pour XSLT

MSXML2.4.0

Inclus :
- Processeur XSLT
- Evaluation XPath
- Quelques extensions très utiles
- Support des schémas XSD
- SAX

 Il faut positionner certains paramètres du parser pour être conforme au standard.

Conforme aux spécifications W3C
Performant
Pas d'accès au cœur (boîte noire)
Incorporé dans le navigateur IE6
Technologie ActiveX
(utilisable avec scripts Jscript, VBscript, et les langages compilés VB, C++…)

Sur le client : **îlots de données** (technologie propriétaire Microsoft)

XML data islands

⇒ Son utilisation suppose la maîtrise absolue des logiciels installés sur le parc client

Les îlots de données permettent d'incorporer des données XML dans une page HTML.

Inclusion XML dans HTML

L'insertion d'un îlot de données s'effectue par l'intermédiaire d'une balise `<xml>`

```
<xml id="nom">
        Données XML …/…
</xml>
```

Le balisage peut soit contenir un arborescence XML valide, soit utiliser un attribut `src` pointant vers un document XML.

```
<xml id="nom" src="url"></xml>
```

Il est également possible d'intégrer des îlots de données XML par l'intermédiaire des balises `<script>`.

```
<script type="text/xml" id="nom">
        Données XML …/…
</script>
```

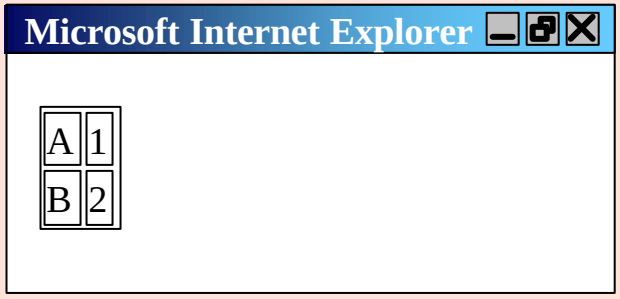
```
<script language="xml" id="nom">
        Données XML …/…
</script>
```

Utilisation des îlots de données XML

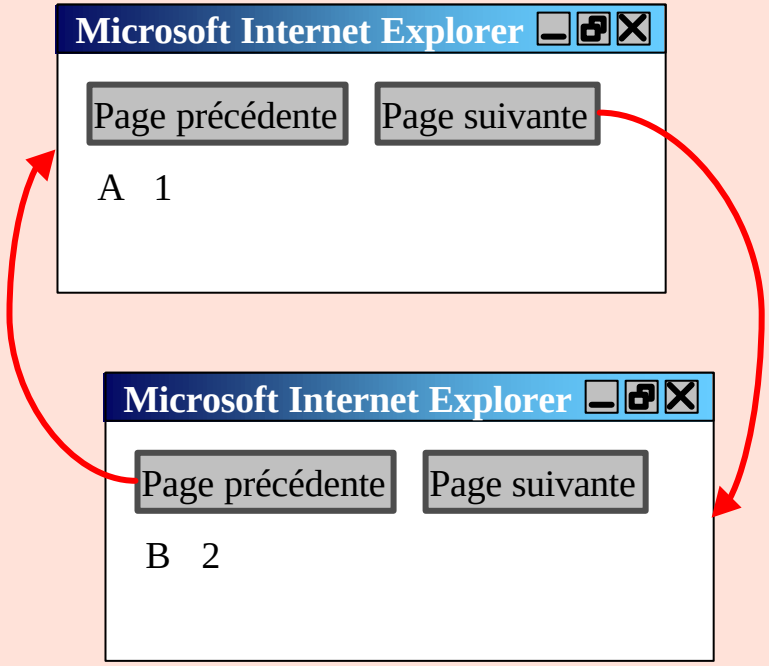
L'extraction des données peut se faire en utilisant plusieurs attributs associés aux balises HTML comme <table>, , <input> ou .

```
<table datasrc="#nom">
  <tr>
    <td>
      <span datafld="Nom_Elément"></span>
    </td>
  </tr>
</table>
```

```
<html>
  <body>
    <xml id="exemple">
      <root>
        <foos>
          <foo>A</foo><bar>1</bar>
        </foos>
        <foos>
          <foo>B</foo><bar>2</bar>
        </foos>
      </root>
    </xml>
    <table datasrc="#exemple" border="1">
      <tr>
        <td><span datafld="foo"></span></td>
        <td><span datafld="bar"></span></td>
      </tr>
    </table>
  </body>
</html>
```



```
<html>
  <body>
    <xml id="exemple">
      <root>
        <foos>
          <foo>A</foo><bar>1</bar>
        </foos>
        <foos>
          <foo>B</foo><bar>2</bar>
        </foos>
      </root>
    </xml>
    <input type="button"
           value="Page précédente"
           onclick="tab.previousPage();">
    <input type="button"
           value="Page suivante"
           onclick="tab.nextPage();">
    <table datapagesize="1" id="tab"
           datasrc="#exemple">
      <tr>
        <td><span datafld="foo"></span></td>
        <td><span datafld="bar"></span></td>
      </tr>
    </table>
  </body>
</html>
```



Hiérarchisation dans les îlots de données XML

L'attribut `datafld` accepte les niveaux hiérarchiques afin de localiser distinctement les éléments à afficher par rapport à leurs ancêtres dans l'arborescence de l'îlot de données XML. Les éléments parent et enfant sont séparés par un point.

```
<Elément_Racine>
  <Sous_Elément>
    <Enfant>...</Enfant>
  </Sous_Elément>
</Elément_Racine>

<table datasrc="#nom">
  <tr>
    <td>
      <table datasrc="#nom" datafld="Sous_Elément">
        <tr>
          <td>
            <input type="text" datafld="Sous_Elément.Nom_Enfant">
            </input>
          </td>
        </tr>
      </table>
    </td>
  </tr>
</table>
```

```
<!doctype html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <body>
    <xml src="leFichierXML.xml" id="xmlDoc" ondatasetcomplete="go()"></xml>
    <xml src="laFeuilleXSLT.xsl" id="xslDoc"></xml>
    <script type="text/javascript">
      function go(){
        //Attend le chargement des fichiers
        var xmlDoc,xslDoc;
        xmlDoc=document.getElementById("xmlDoc");
        xslDoc=document.getElementById("xslDoc");
        if ((xmlDoc.readyState=="complete" || xmlDoc.readyState==4) &&
          (xslDoc.readyState=="complete" || xslDoc.readyState==4)) what();
        else setTimeout("go()",100)
      }
      function what(){
        //effectue la transformation affecte le résultat à la variable res
        var res,xmlDoc,xslDoc;
        xmlDoc=document.getElementById("xmlDoc").documentElement;
        xslDoc=document.getElementById("xslDoc").documentElement;
        res=xmlDoc.transformNode(xslDoc);
        //remplace le HTML actuel par le résultat de la transformation
        document.body.innerHTML=res;
      }
    </script>
  </body>
</html>
```

Implémentation de SAX

Xerces

`org.xml.sax.XMLReader` interface

`org.apache.xerces.parsers.SAXParser` implémentation

```
XMLReader parser = new SAXParser();
```

En passant par un factory, on n'a plus besoin d'importer spécifiquement l'implémentation

Abstract factory - Factory method design patterns

Utilisation de SAX


```
import java.io.IOException;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.XMLReaderFactory;
import org.xml.sax.ContentHandler;
import org.xml.sax.ErrorHandler;

…/…

ContentHandler contentHandler = new MyContentHandler();
ErrorHandler errorHandler = new MyErrorHandler();
try {
        XMLReader parser = XMLReaderFactory.createXMLReader(
                "org.apache.xerces.parsers.SAXParser");
        parser.setContentHandler(contentHandler);
        parser.setErrorHandler(errorHandler);
        parser.parse(uri);
} catch (IOException e) {
        System.out.println("Erreur de lecture URI: "+ e.getMessage());
} catch (SAXException e) {
        System.out.println("Erreur d'analyse: "+ e.getMessage());
}
```

Au lieu de :

```
XMLReader parser = XMLReaderFactory.createXMLReader(
        "org.apache.xerces.parsers.SAXParser");
```

Ou de :

```
XMLReader parser = XMLReaderFactory.createXMLReader();
```

Utilisation des propriétés systèmes de Java

Utiliser plutôt :

```
XMLReader parser = XMLReaderFactory.createXMLReader(
        PropertiesReader.getInstance().getProperty("analyseurSAX"));
```

Permet de ne pas spécifier une propriété système (ici, l'analyseur SAX) pour un composant particulier mais pour toutes les applications distribuées

Configuration des parsers avec JAXP

JAXP est une interface abstraite qui permet d'utiliser :

- un parser DOM level 2
- un parser SAX 2.0
- un processeur XSLT (TrAX)

JAXP permet :

- de contrôler le processus d'analyse
- aux applications d'être indépendantes d'une implémentation
- pluggable

L'implémentation de l'outil utilisé peut être spécifié de plusieurs manières différentes :

- En définissant des propriétés système :
  `javax.xml.parsers.SAXParserFactory`
- Dans le fichier :
  `$JAVA_HOME/jre/lib/jaxp.properties`
- Jar Service Provider
  `META-INF/services/javax.xml.parsers.SAXParserFactory`
- En utilisant le parser par défaut (fallback)

Configuration des parsers avec JAXP

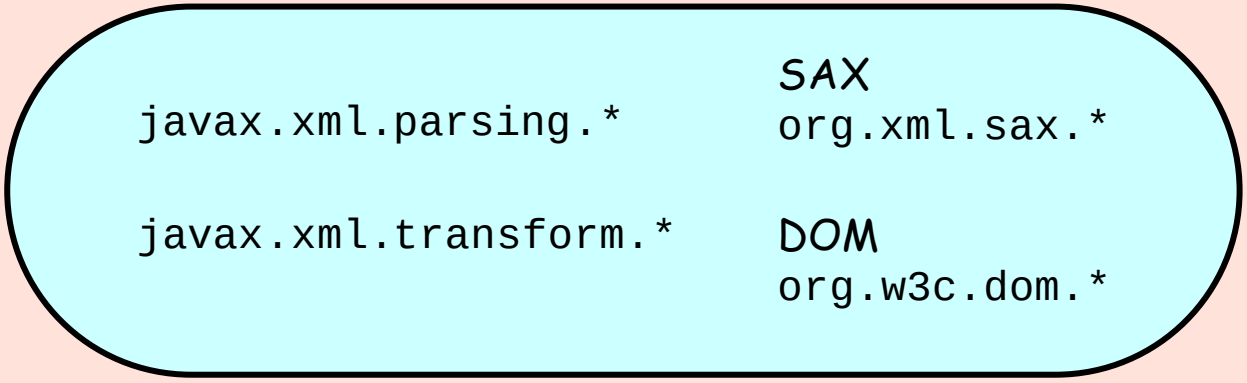
Exemples d'utilisation d'une implémentation particulière (avec Java 2 version 1.3) :

- Pour obtenir Xerces, utiliser classpath =
  `xerces.jar` (contient toutes les classes)

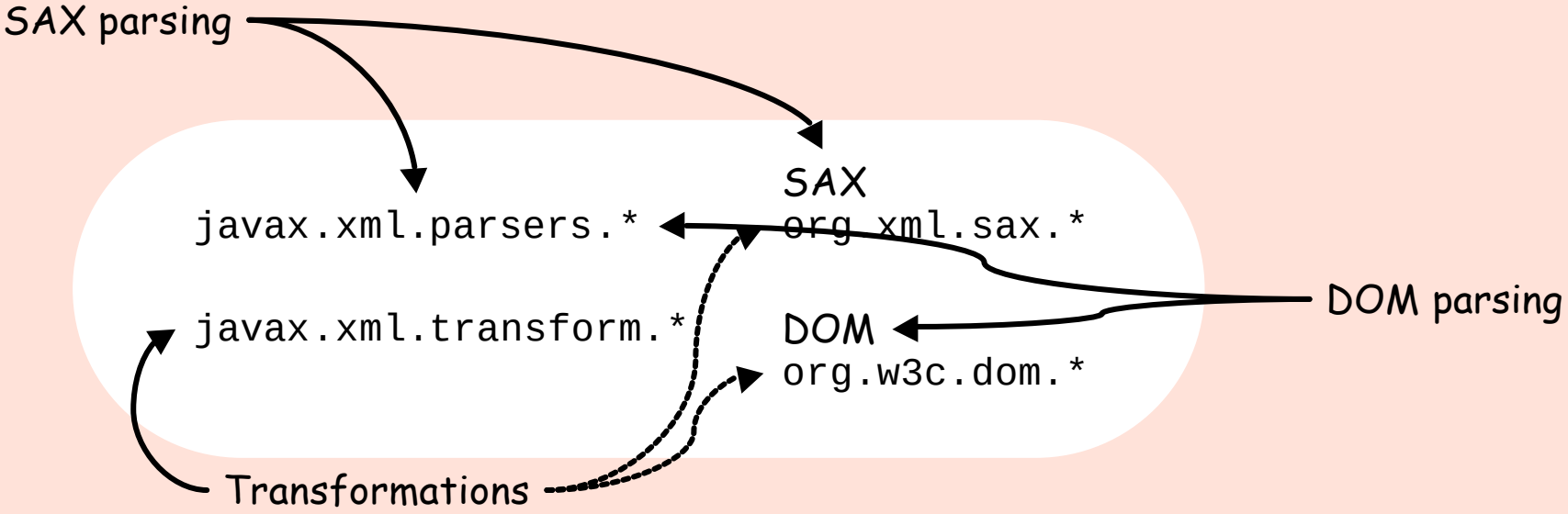
- Pour obtenir Crimson, utiliser classpath =
  `jaxp.jar` (contient `javax.xml.*`)
  `crimson.jar` (contient sax, dom)

- On obtient Xerces, si classpath =
  `jaxp.jar`
  `xerces.jar`
  `crimson.jar`

Composants de JAXP



JAXP API



Création d'un document par programme

Dépendant d'un parser

Xerces

```
import org.w3c.dom.Document;
import org.apache.xerces.dom.DocumentImpl;
…/…
Document doc = new DocumentImpl();
```

MSXML2

```
var objDoc = new ActiveXObject("Msxml2.DOMDocument.4.0");
```

JAXP

```
import org.w3c.dom.Document;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
…/…
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document doc = builder.newDocument();
```

Est indépendant de toute implémentation

Java reflection

```
import org.w3c.dom.Document;
…/…
String docImpl = "org.apache.xerces.dom.DocumentImpl";
Document doc = (Document) Class.forName(docImpl).newInstance();
```

Se compile sans le parseur, dont le nom de classe peut être un paramètre système, ou contenu dans un fichier de configuration

Prise en compte des espaces de nommage

Par défaut, JAXP ne prend pas en compte les espaces de nommage

```
import org.w3c.dom.Document;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
…/…
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
factory.setNamespaceAware(true);
DocumentBuilder builder = factory.newDocumentBuilder();
builder.parse(…/…);
```

Par défaut, Xerces prend en compte les espaces de nommage

```
DOMParser parser = new DOMParser();
parser.setFeature("http://xml.org/sax/features/namespaces", false)
parser.parse(…/…);
Document doc = parser.getDocument();
```

Le goût du blanc

MSXML2.4.0 ne conserve pas les nœuds blancs, sauf s'ils sont spécifiés par `xml:space="preserve"`, ou si le parser reçoit explicitement la directive `preserveWhiteSpace` :

```
doc=new ActiveXObject("MSXML2.DOMDocument.4.0");
doc.async=false;
doc.preserveWhiteSpace=true;
doc.load("monFichier.xml");
```

JAXP et Xerces adoptent par défaut le comportement standard.

Xerces peut supprimer les nœuds blancs à condition qu'une grammaire soit présente pour que le parser puisse reconnaître les blancs ignorables.

```
DOMParser parser = new DOMParser();
parser.setFeature("http://apache.org/xml/features/dom/include-ignorable-whitespace", false)
parser.parse(…/…);
Document doc = parser.getDocument();
```

MSXML2.4.x

```
doc=new ActiveXObject("MSXML2.DOMDocument.4.0");
doc.async= false;
doc.validateOnParse= true;
doc.load("monFichier.xml");
```

Xerces

```
parser.setFeature("http://xml.org/sax/features/validation", true)
```

JAXP

```
factory.setValidating(true);
```

Schema validation

Xerces

```
parser.setFeature("http://apache.org/xml/features/validation/schema", true)
```

JAXP

```
factory.setNamespaceAware(true);
```

MSXML2.4.x

```
doc.validateOnParse= true;
```

Patrons de programmation avec Xerces

DOM parser

```
import org.apache.xerces.parsers.DOMParser;
import org.w3c.dom.Document;

String filename;
…/…
DOMParser parser = new DOMParser();
parser.parse(filename);
Document doc = parser.getDocument();
```

SAX parser

```
import org.xml.sax.helpers.XMLReaderFactory;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.XMLReader;

DefaultHandler handler;
String filename;
…/…
XMLReader parser = XMLReaderFactory.createXMLReader();
parser.setContentHandler(handler);
parser.setDTDHandler(handler);
parser.setErrorHandler(handler);
parser.parse(filename);
```

Patrons de programmation avec JAXP

DOM parser

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;

String filename;
…/…
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document doc = builder.parse(filename);
```

SAX parser

```
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.SAXParser;
import org.xml.sax.helpers.DefaultHandler;

DefaultHandler handler;
String filename;
…/…
SAXParserFactory factory = SAXParserFactory.newInstance();
SAXParser parser = factory.newSAXParser();
parser.parse(filename, handler);
```

Patrons de programmation avec MSXML2.4.x

DOM parser avec JScript

```
doc=new ActiveXObject("MSXML2.DOMDocument.4.0");
doc.async= false;
filename ="…/…";
doc.load(filename);
```

SAX parser avec VB

```
Dim reader As SAXXMLReader40
Dim contentHandler As ContentHandlerImpl
Dim errorHandler As ErrorHandlerImpl
Set reader = New SAXXMLReader40
Set contentHandler = New ContentHandlerImpl
Set errorHandler = New ErrorHandlerImpl
Set reader.contentHandler = contentHandler
Set reader.errorHandler = errorHandler
```

SAX2 ContentHandler

```
public interface ContentHandler {
        void startElement(namespaceURI, localName, qName, atts);
        void endElement(namespaceURI, localName, qName);
        void characters(ch[], start, length);
        …/…
}
```

Application SAX2 typique

```
XMLReader xmlReader = [create SAX2 XMLReader instance]
xmlReader.setContentHandler(myContentHandler);
xmlReader.parse(myInputSource);
```

XML Reader

```
SAXParserFactory spf = SAXParserFactory.newInstance();
// Change namespaces feature to SAX2 default
spf.setNamespaceAware(true);
// Create SAX XMLReader w/ SAX2 default features
XMLReader xmlReader = spf.newSAXParser().getXMLReader();
// Use xmlReader as you would normally
…/…
```

Patron de programmation DOM avec JAXP


```
// Get platform default implementation
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
// Set options
dbf.setNamespaceAware(true);
dbf.setValidating(true);
// Create new builder
DocumentBuilder db = dbf.newDocumentBuilder();
db.setErrorHandler(myErrorHandler);
Document doc = db.parse("myFile.xml");
// Use doc with usual DOM methods
…/…
```

Error handling


```
JAXP DOM Example
// Get platform default implementation
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
// Set options
dbf.setNamespaceAware(true);
dbf.setValidating(true);
// Create new builder
DocumentBuilder db = dbf.newDocumentBuilder();
db.setErrorHandler(myErrorHandler);
Document doc = db.parse("http://server.com/foo.xml");
// Use doc with usual DOM methods
...
```

EntityResolver


```
import org.xml.sax.EntityResolver;
public class MyResolver implements EntityResolver {
        public InputSource resolveEntity(String publicId, String systemId)
                throws FileNotFoundException {
                if (publicId!=null) {
                        if (publicId.equals("my public identifier")) {
                                // return an input source object
                        }
                } else return null;
        }
}
```

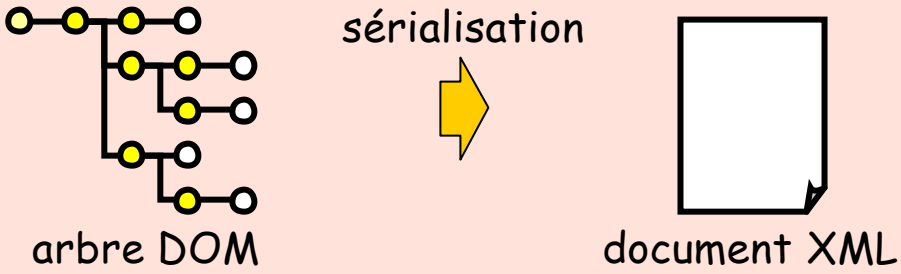
Utilisation :

```
DocumentBuilderFactory dFactory =
DocumentBuilderFactory.newInstance();
DocumentBuilder parser = dFactory.newDocumentBuilder();
MyResolver myResolver = new MyResolver();
parser.setEntityResolver(myResolver);
Document DOMdoc = parser.parse(new File(xmlFile));
```

S'il y a des résolutions en cascade, il faut que le parser puisse résoudre les appels relatifs

InputSource :

```
// return an input source object
if (systemId.equals("http://www.foo.com/bar.dtd") {
        InputStream inputStream = new FileInputStream("/dtd/bar.dtd");
        InputSource inputSource = new InputSource(inputStream);
        inputSource.setPublicId(publicId);
        inputSource.setSystemId(systemId);
        return inputSource;
}
```



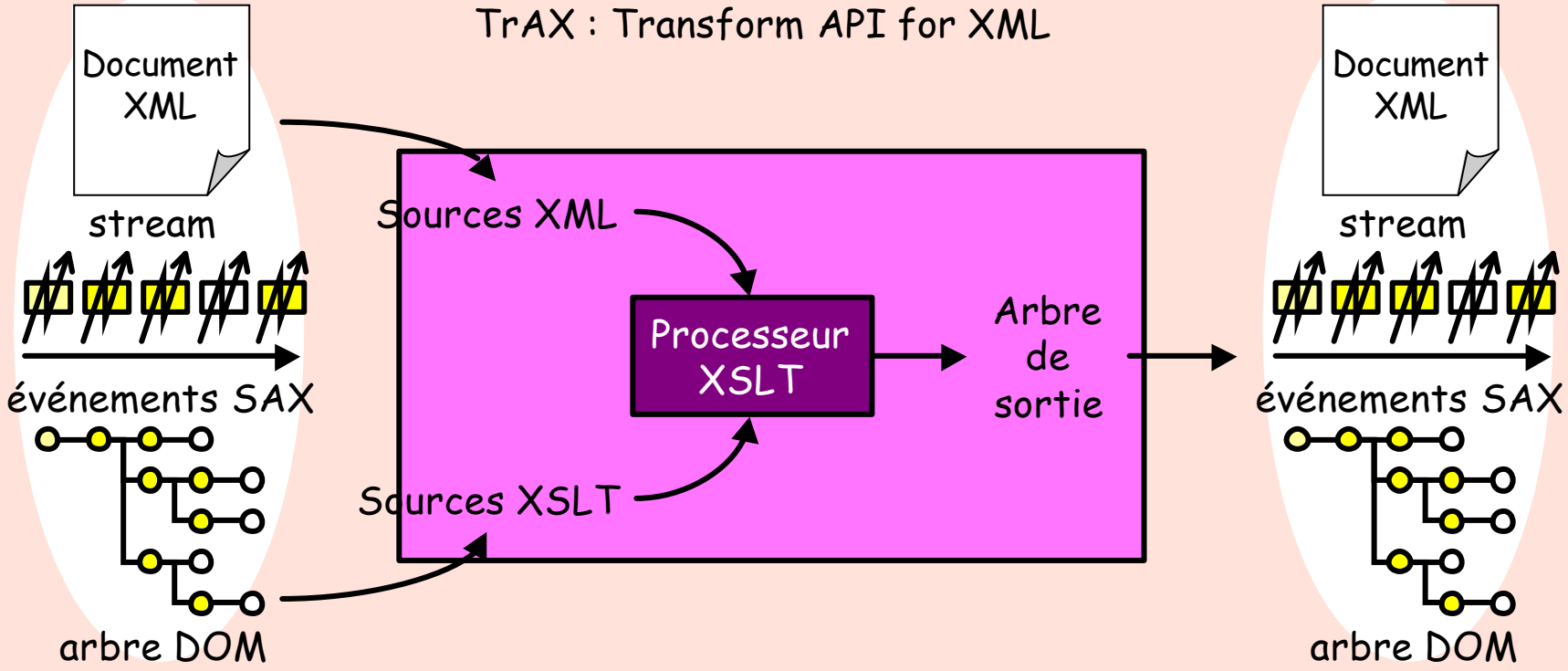
MSXML2.4.x

`node.xml;` ← Pas de maîtrise de l'identation, de l'encodage…

Apache

```
import org.apache.xml.serialize.XMLSerializer;
import org.apache.xml.serialize.OutputFormat;
…/…
OutputFormat formatter = new OutputFormat("xml", "utf-8", true);
formatter.setPreserveSpace(true);
XMLSerializer serializer = new XMLSerializer(System.out,  formatter);
serializer.serialize();
```

```
// Get concrete implementation
TransformerFactory tf = TransformerFactory.newInstance();
// Create a transformer for a particular stylesheet
Transformer transformer = tf.newTransformer(new StreamSource(stylesheet));
// Transform input XML doc to System.out
transformer.transform(new StreamSource(xmlSource), new StreamResult(System.out));
```




```
javax.xml.transform.Source
  |
  +--javax.xml.transform.dom.DOMSource
  |
  +--javax.xml.transform.sax.SAXSource
  |
  +--javax.xml.transform.stream.StreamSource
```

```
javax.xml.transform.Result
  |
  +--javax.xml.transform.dom.DOMResult
  |
  +--javax.xml.transform.sax.SAXResult
  |
  +--javax.xml.transform.stream.StreamResult
```

Serialisation avec XSLT


```
// Get concrete implementation
TransformerFactory tf = TransformerFactory.newInstance();
// Create the transformer for copying the source
Transformer transformer = tf.newTransformer();
// Transform input XML doc to System.out
transformer.transform(new DomSource(document), new StreamResult(System.out));
```

L'objet `transformer` obtenu par `TransformerFactory#newTransformer()` correspond à la feuille de style de copie.

On utilise ici le processeur XSLT uniquement à des fins de sérialisation

Certains processeur XSLT peuvent aussi traiter des flux SAX :

```
// un sérialiseur SAX
  public TransformerHandler getSerializer(String output)
      throws TransformerConfigurationException {
    TransformerFactory factory = TransformerFactory.newInstance();
    TransformerHandler transformerHandler =
      ((SAXTransformerFactory) factory).newTransformerHandler();
    Result result = new StreamResult( output );
    transformerHandler.setResult( result );
    return transformerHandler;
  }
}
```

Patron de programmation pour sérialiser un flux SAX

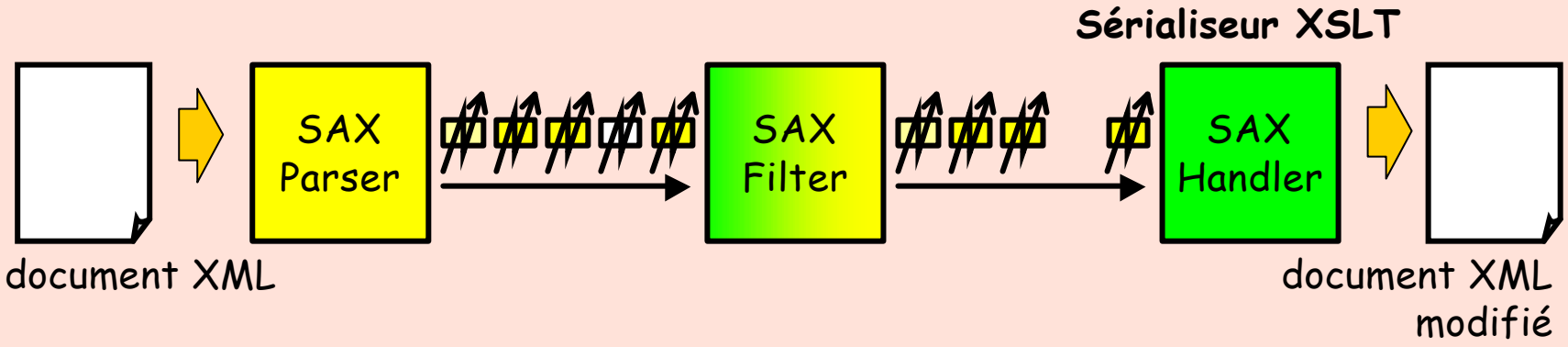

```
import java.io.*;
import java.util.*;
import java.text.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.sax.*;
import javax.xml.transform.stream.*;

public abstract class SaxUpdate extends XMLFilterImpl {

  // un filtre SAX
  public SaxUpdate() throws ParserConfigurationException, SAXException {
    // on crée le parser SAX
    SAXParserFactory factory = SAXParserFactory.newInstance();
    factory.setValidating( true );
    SAXParser saxParser = factory.newSAXParser();
    XMLReader parser = saxParser.getXMLReader();
    // on branche le filtre sur le parseur
    super.setParent( parser );
  }

  // un sérialiseur SAX
  public TransformerHandler getSerializer(String output)
      throws TransformerConfigurationException {
    TransformerFactory factory = TransformerFactory.newInstance();
    TransformerHandler transformerHandler =
      ((SAXTransformerFactory) factory).newTransformerHandler();
    Result result = new StreamResult( output );
    transformerHandler.setResult( result );
    return transformerHandler;
  }
}
```

Exemple de sérialisation d'un flux SAX modifié

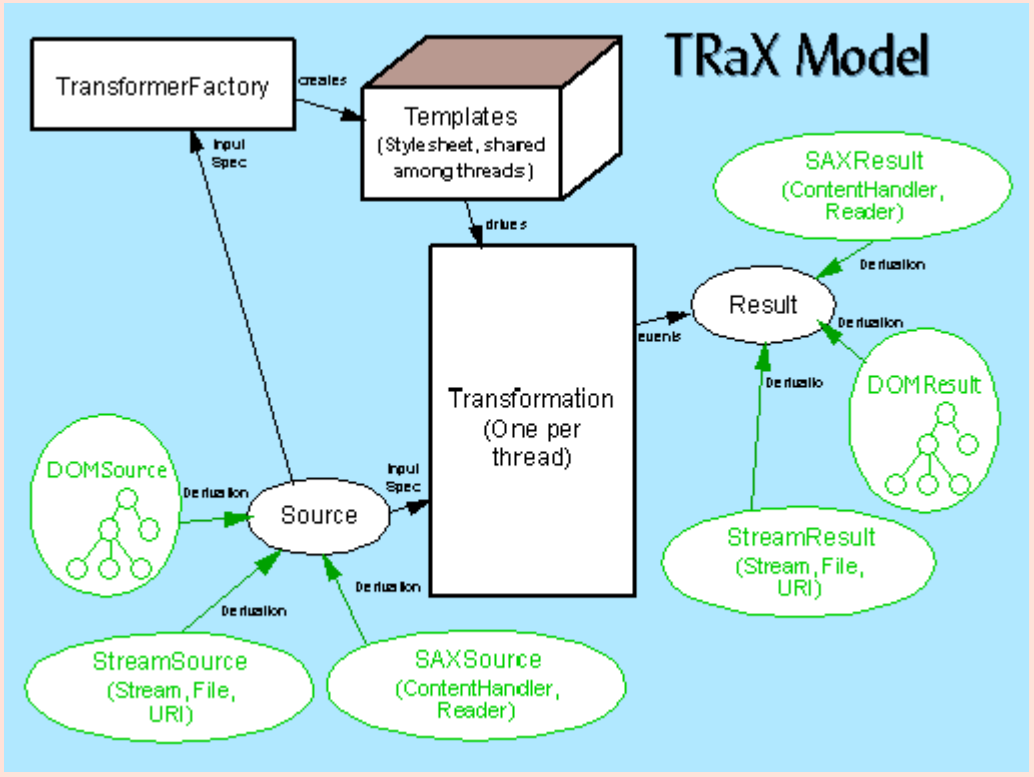


Exemple : on ajoute un attribut aux éléments

```
SaxUpdate saxUpdate = new SaxUpdate() {
  // concrète implémentation de SaxUpdate

  public void startElement(String uri, String localName,
        String qName, Attributes atts) throws SAXException {
    // on crée un nouvel attribut
    AttributesImpl attributes = new AttributesImpl( atts );
    attributes.addAttribute( "", "", "version", "CDATA", "1.0");
    atts = attributes;
    // on transmet ce qui a été modifié au handler (le sérialiseur XSLT)
    this.getContentHandler().startElement(uri, localName, qName, atts);
  }
};
// on se branche sur le sérialiseur
saxUpdate.setContentHandler( saxUpdate.getSerializer( outputFileName ) );
// on veut récupérer les erreurs de parsing
saxUpdate.setErrorHandler( saxUpdate.getParent().getErrorHandler() );
// on lance le parsing
saxUpdate.parse( inputFileName );
```

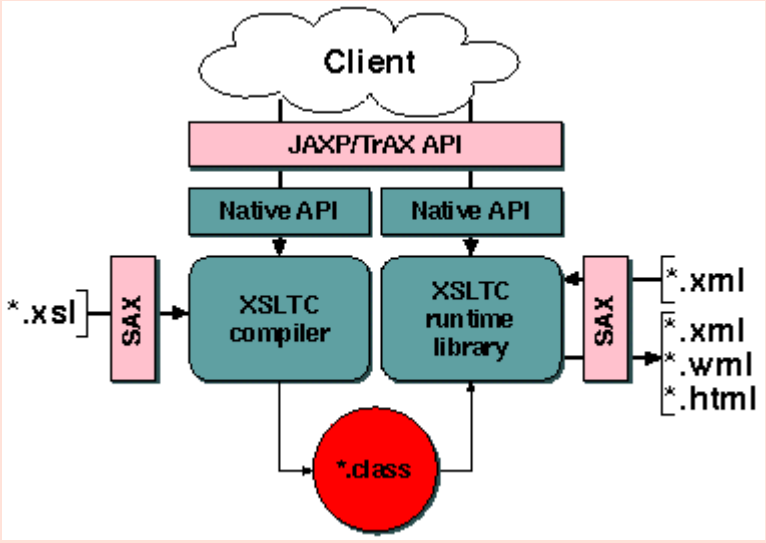
Architecture de XSLT dans JAXP



Architecture de XSLT dans Xalan

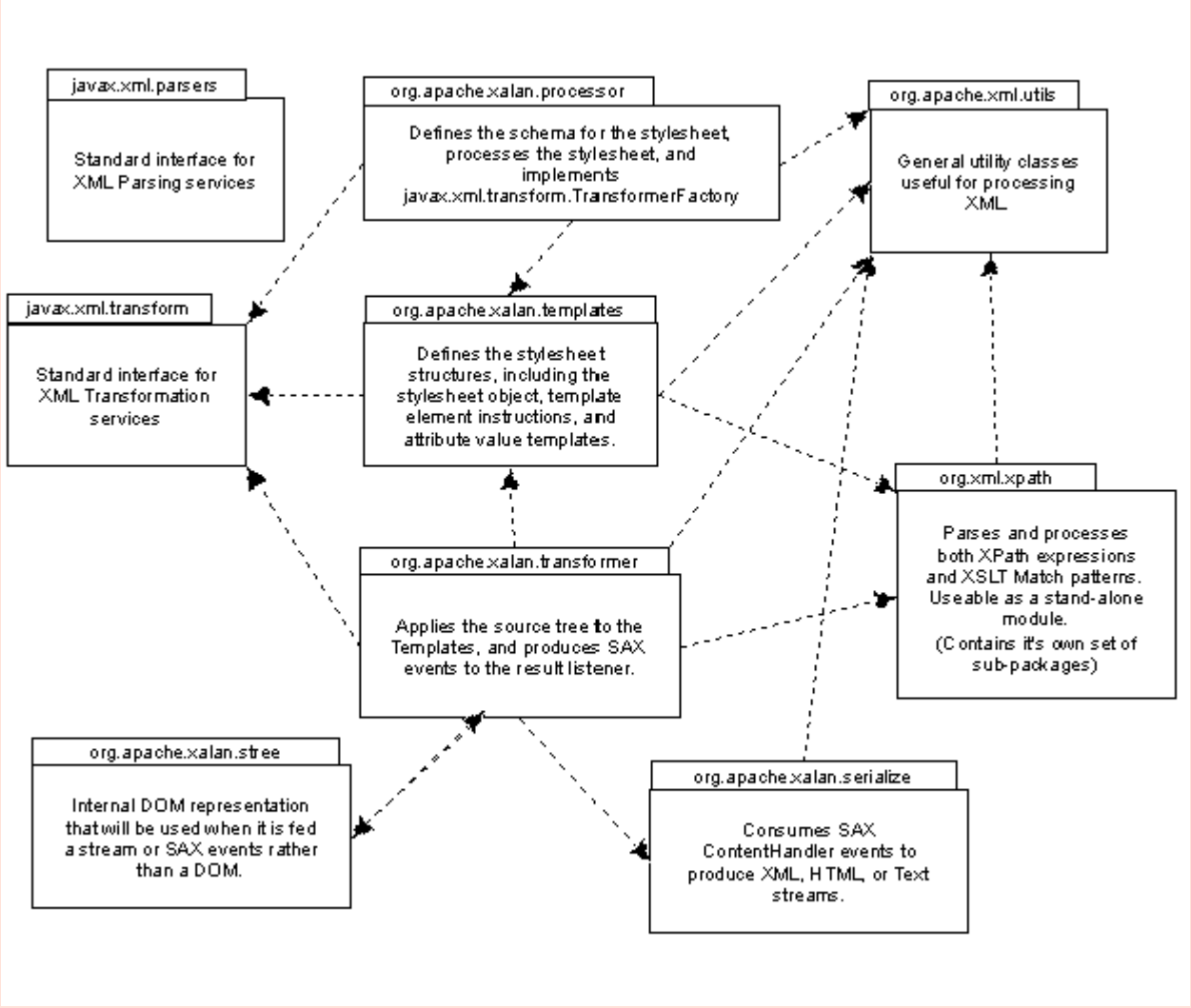
Performance : avec Xalan, les feuilles de style peuvent être compilées

XSLTC Translets → Compiler
XSLTC Translets → Runtime processor



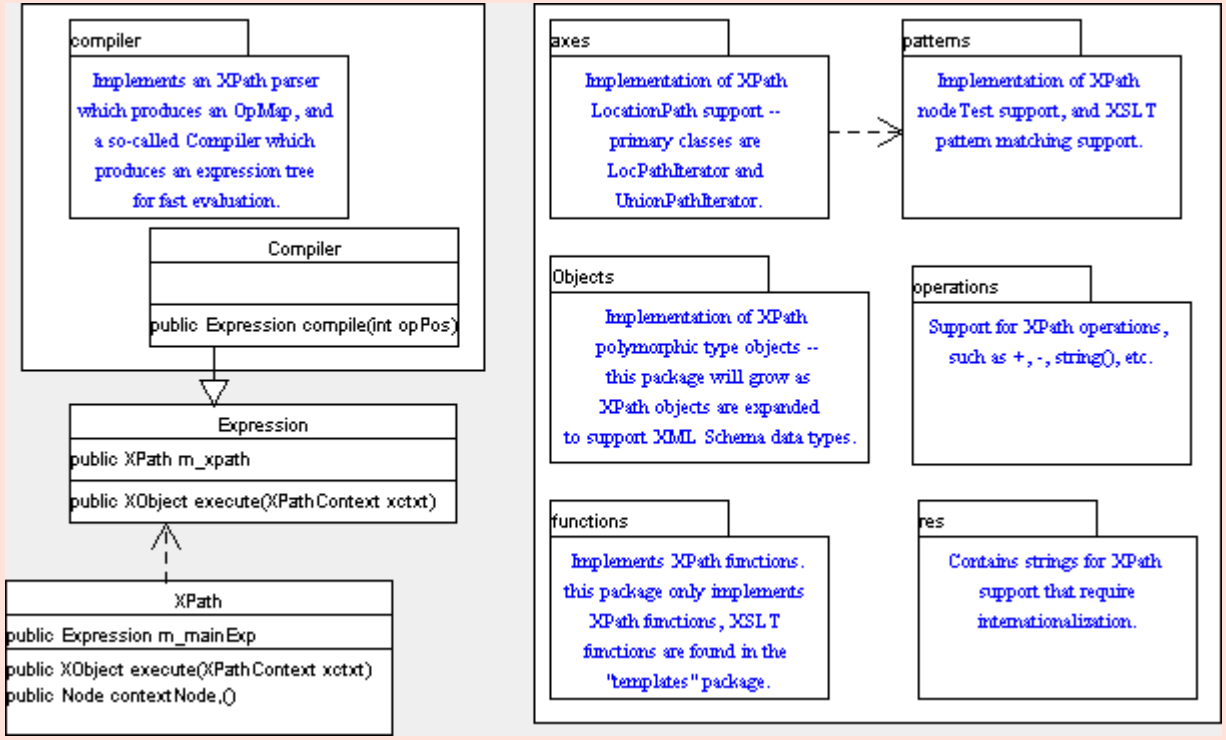
XSLT → XSLTC (compilateur Xalan) → Translets foo xml ns → Runtime processor

Liaison XSLT entre JAXP et Xalan



XPath dans Xalan

XPath API : permet de soumettre des expressions XPath à un node.




```
import org.apache.xpath.XPathAPI;
…/…
NodeList nodeList = XPathAPI.selectNodeList (node, expression1);
Node node = XPathAPI.selectSingleNode (node, expression2);
XObject xObj = XPathAPI.eval(node, expression3);
```

```
xObj.getType();
```

⇒

```
CLASS_BOOLEAN
CLASS_NODESET
CLASS_NULL
CLASS_NUMBER
CLASS_RTREEFRAG
CLASS_STRING
CLASS_UNKNOWN
CLASS_UNRESOLVEDVARIABLE
```

Performance : les expressions XPath peuvent être compilées

XSLT côté serveur

Sur un serveur Web, on peut être tenté de mettre en cache les feuilles de style pour réutilisation.

Problème: plusieurs clients peuvent faire la même requête simultanément.

Les données sources ne sont que lues, mais les objets qui servent aux transformations peuvent subir des modifications : ce sont des objets "jetables".

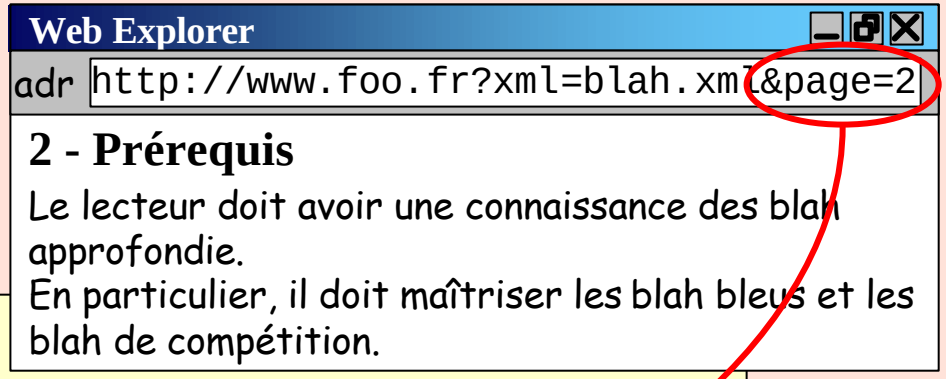
Avec JAXP, on peut mettre en cache les templates, et instancier à chaque requête des transformer.

Exemple de servlet :

```
public class MyServlet extends HttpServlet {
  public void init() throws ServletException {
    TransformFactory tf = TransformFactory.newInstance();
    Templates myTemplate = tf.newTemplates(new StreamSource("style.xsl"));
    this.getServletContext().setAttribute("xslt", myTemplate);
  }
  public doGet(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException {
    String docName = req.getParameter("xml");
    Templates myTemplate = (Templates) this.getServletContext().getAttribute("xslt");
    Transformer transformer = myTemplate.newTransformer();
    DOMResult res = new DOMResult();
    String fileName = this.getServletContext().getRealPath(docName);
    transformer.transform(new StreamSource(fileName), res);
    …/…
}
```

Passage de paramètres avec XSLT

JAXP




```
String docName = req.getParameter("xml");
// Get concrete implementation
TransformFactory tf = TransformFactory.newInstance();
// Create a transformer for a particular stylesheet
Transformer transformer = tf.newTransformer(new StreamSource(stylesheet));
// Set parameter
transformer.setParameter("page", req.getParameter("page"));
// Transform input XML doc to System.out
String fileName = this.getServletContext().getRealPath(docName);
transformer.transform(new StreamSource(docName), new StreamResult(System.out));
```

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
    <xsl:param name="page" select="1">
    <xsl:template match="/">
        <html>
            <body>
                <xsl:apply-templates select="//section[number($page)]"/>
            </body>
        </html>
    </xsl:template>
    <xsl:template match="section">
        <h1><xsl:apply-templates select="title"/></h1>
        <xsl:apply-templates/>
    </xsl:template>
</xsl:stylesheet>
```

Passage de paramètres avec XSLT

MSXML2

```
oDoc=new ActiveXObject("MSXML2.DOMDocument.4.0");
oDoc.async=false;
oDoc.load(xmlDocument);
oXsl=new ActiveXObject("MSXML2.FreeThreadedDOMDocument.4.0");
oXsl.async=false;
oXsl.load(xslDocument);
oXsl.setProperty("SelectionLanguage", "XPath");
oXsl.setProperty("SelectionNamespaces", 'xmlns:xsl="http://www.w3.org/1999/XSL/Transform"');
oTemplate=new ActiveXObject("Msxml2.XSLTemplate.4.0");
oTemplate.stylesheet=oXsl;
oProcessor = oTemplate.createProcessor();
oProcessor.input = oDoc;
oProcessor.addParameter("page", pageNumber);
oProcessor.transform();
result = oProcessor.output;
```

Transtypage d'un result-tree-fragment en node-set

- fonctionnalité très utile
- proposée dans de nombreuses implémentations

Xalan-Java

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xalan="http://xml.apache.org">
…/…
        <xsl:apply-templates select="xalan:nodeset($lieu)/nation"/>
…/…
```

```
<xsl:variable name="lieu">
    Stade de France
    <nation>fr</nation>
</xsl:variable>
```

MSXML2.4.x

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:msxsl="urn:schemas-microsoft-com:xslt">
…/…
        <xsl:apply-templates select="msxml:node-set($lieu)/nation"/>
…/…
```

MSXSML.DOMDocument ← Obsolète

```
<xsl:script><![CDATA[
function maFonction(node) {
          …/…
          return resultat;
}
]]></xsl:script>
<xsl:template match="…/…">
    <xsl:eval>maFonction(this)</xsl:eval>
</xsl:template>
```

MSXML2.DOMDocument.4.x

```
<xsl:stylesheet
          version="1.0"
          xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
          xmlns:msxsl="urn:schemas-microsoft-com:xslt">
    <msxsl:script language="JScript" implements-prefix="javascript">
          function maFonction(nodeList) {
                    …/…
                    return resultat;
          }
    </msxsl:script>
    <xsl:template match="…/…">
        <xsl:value-of select="javascript:maFonction(.)"/>
    </xsl:template>
```

Apache Xalan

```
<xsl:stylesheet
          version="1.0"
          xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
          xmlns:lxslt="http://xml.apache.org/xslt"
          xmlns:user="http://mycompany.com/mynamespace"
          extension-element-prefixes="user">
<!--The component and its script are in the lxslt namespace and define the
implementation of the extension.-->
<lxslt:component prefix="user" elements="timelapse" functions="getdate">
          <lxslt:script lang="javascript">
var multiplier=1;
// The methods or functions that implement extension elements always take 2
// arguments. The first argument is the XSL Processor context; the second
// argument is the element node.
function timelapse(xslProcessorContext, elem) {
          multiplier=parseInt(elem.getAttribute("multiplier"));
          return null;
}
function getdate(numdays) {
          var d = new Date();
          var totalDays = parseInt(numdays) * multiplier;
          d.setDate(d.getDate() + totalDays);
          return d.toLocaleString();
}
          </lxslt:script>
</lxslt:component>
<xsl:template match="deadline">
          <p><user:timelapse multiplier="2"/>We have logged your enquiry
           and will respond by
          <xsl:value-of select="user:getdate(string(@numdays))"/>.</p>
</xsl:template>
```

Choix d'un parser

Dépend essentiellement de la plate-forme

Dans tous les cas, bien connaître son parser :

- les options par défaut
- les paramétrages possibles
- les limitations vis à vis des standards
- les extensions propriétaires

L'usage qu'on souhaite en avoir dans sa propre application

Voir les documentations disponibles

Autres processeurs

Spécialisés

Batik	Processeur SVG
FOP	Processeur XSLFO

Propriétaires

Ant	Utilitaire de déploiement d'applications Java
Cocoon	Processeur XSP

Processeur "maison" :

- associé à des éléments ou des attributs qui dépendent d'un espace de nommage
- automate
- fichier de configuration

Conception de son processeur

Pour quoi faire ?
N'en existe-t-il pas un qui fasse déjà le travail demandé ?
…

Castor
http://castor.exolab.org
JAXB

Conception : basée sur les techniques de data binding
- Définir son jeu de balises et d'attribut et l'espace de nommage
- Associer les structures XML à des classes

Exemple

```
<?xml version="1.0"
  encoding="ISO-8859-1"?>
<root>
  <foo param="…" />
  <foo param="…">
    <bar value="…" type="…">
      Blah
    </bar>
    <foo param="…" />
  </foo>
  <bar value="…" type="…">
      Blah blah blah
  </bar>
</root>
```

```
try {
  DocumentBuilderFactory
    factory = DocumentBuilderFactory.newInstance();
  factory.setNamespaceAware(true);
  DocumentBuilder
    builder = factory.newDocumentBuilder();
  Document doc = builder.parse(fileName);
  Element root = doc.getDocumentElement();
  RootAction rootAction = null;
  if (root.getNodeName().equals("root")) {
    rootAction = RootAction.unmarshal(root);
  }
  rootAction.run();
} catch (ParserConfigurationException e) {
  // report configuration error
} catch (SAXException e) {
 // report parsing error
} catch (IOException e) {
 // report IO error
}
```

Conception de son processeur


```
public class AbstractAction {
  Vector actions;
  AbstractAction parent;
  public static AbstractAction unmarshal(AbstractAction parent, Element e) {
    for (Node n=e.getFirstChild(); n!=null; n=n.getNextSibling()) {
      if (n.getNodeType()==Node.ELEMENT_NODE) {
        if (n.getNodeName().equals("foo")) {
          parent.addAction(FooAction.unmarshal((Element) n));
        } else if (n.getNodeName().equals("bar")) {
          parent.addAction(BarAction.unmarshal((Element) n));
        }
      }
    }
    return parent;
  }
  private final void addAction(AbstractAction action) {
    actions.add(action);
    action.parent = this;
  }
  public void run() {
    // Execute all dependant abstract actions
    doItBefore();
    for (Enumeration action = actions.elements() ; action.hasMoreElements() ;) {
      ((AbstractAction) action.nextElement()).run();
    }
    doItAfter();
  }
  public void doItBefore() {}
  public void doItAfter() {}
}
```

Conception de son processeur


```
public class FooAction extends AbstractAction {
  private String fooParam;
  // Creates a new instance of fooAction
  private FooAction() {
    actions = new Vector();
  }
  public static FooAction unmarshal(Element e) {
          FooAction fooAction = new FooAction();
          fooAction.fooParam = e.getAttribute("param");
          AbstractAction.unmarshal(fooAction, e);
          return fooAction;
  }
  public void doItBefore() {
    // do foo actions before running dependant sub-actions
  }
  public void doItAfter() {
    // do foo actions after running dependant sub-actions
  }
}
```

Les actions peuvent être :

- connection à une base de données
- mise à jour d'une table
- envoi d'un mail
- …

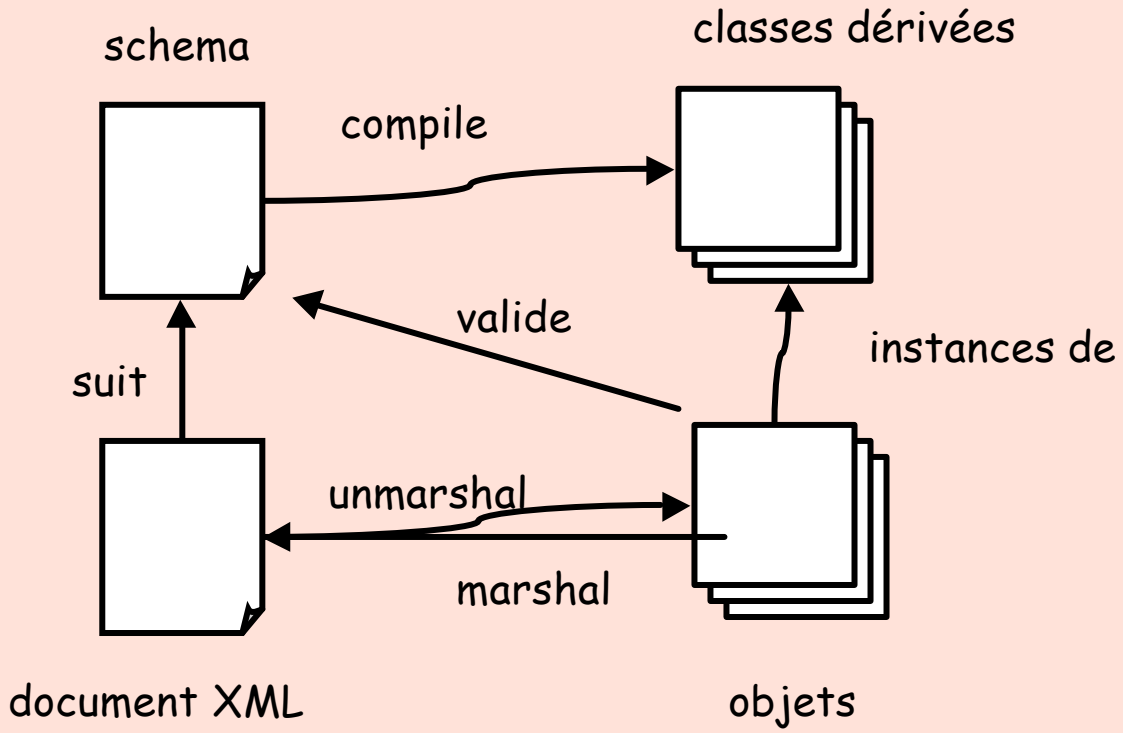
On peut enrichir le code pour conditionner l'exécution d'une action en fonction du résultat de la précédente

```
public class RootAction extends AbstractAction {
private RootAction() {
    actions = new Vector();
    parent = null;
  }
  public static RootAction unmarshal(Element e) {
    RootAction root = new RootAction();
    AbstractAction.unmarshal(root, e);
    return root;
  }
}
```

On peut réaliser des processeurs multi-threads dont le document XML est analysé à l'initialisation du programme, et exécuté pour chaque thread avec un jeu de données différent (attention à la façon d'isoler les données)
→ servlet

…/…

JAXB


schema
classes dérivées
compile
valide
instances de
suit
unmarshal
marshal
document XML
objets