

# Langages de l'internet

## XML et XSL

Bruno Mascret

**CPE Lyon**  
Techniques et langages de l'Internet

# Plan

- 1 Introduction
- 2 Présentation des architectures

# Plan

## 1 Introduction

## 2 Présentation des architectures

# Introduction

## XML : modèles et instances de modèles

### Quels intérêts, quelles applications ?

- ≡ représentation de données
- ≡ organisation arborescente des données
- ≡ spécification des formats
- ≡ ...

Des API existent pour pratiquement chaque langage (php, java, c++, etc.)

# Introduction

XML : modèles et instances de modèles

Quels intérêts, quelles applications ?

- ≡ représentation de données
- ≡ organisation arborescente des données
- ≡ spécification des formats
- ≡ ...

Des API existent pour pratiquement chaque langage (php, java, c++, etc.)

# Plan

1 Introduction

2 Présentation des architectures

# Présentation des architectures XML

Notions de balises, attributs, commentaire

# Présentation des architectures

## XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="nombres.xsl" ?>
<liste_nombres>
  <nombre valeur="10">dix</nombre>
  <nombre valeur="0">zéro</nombre>
  <nombre valeur="33">trente trois</nombre>
  <nombre valeur="6">le premier nombre parfait &#233;</nombre>
    <secret cache="ytjtfghkhkuj">pas montrer
      <attention>le mot de passe &#133;</attention>
    ça
  </secret>
</liste_nombres>
```



# Plan

- 1 Introduction
- 2 Présentation des architectures

# Modèles, validations

## XML

Problème : qu'est-ce qu'un document XML valide ?

Deux réponses :

- ≡ un document VALIDE DANS LA FORME (qui respecte la syntaxe XML)
- ≡ un document VALIDE DANS SUR LE FOND (qui respecte un MODÈLE)

# Modèles, validations

## XML

### Exemple de document non valide sur la forme

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<liste_nombres>
  <nombre valeur="10">dix
  <nombre valeur="0">zéro</nombre>
  <nombre valeur="33">trente trois</nombre>
  <nombre valeur="6">le premier nombre parfait &#233;<nombre>
    <secret cache="ytjtfghkhkuj">pas_montrer
      <attention>le_mot_de_passe_&#133;</attention>
      ça
    </secret>
  </liste_nombres>
  <liste_nombres>
    <nombre valeur="10">dix </nombre>
  </liste_nombres>
```

# Modèles, validations

## XML

### Exemple de document non valide sur le fond

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

<nombre valeur="10">
  <liste_nombres>
    <nombre valeur="0">zÃ©ro</nombre>
    <nombre valeur="33">trente trois</nombre>
    <nombre value="6">le premier nombre parfait &#233;</nom
    <secret truc="ytjtfghkhkuj">pas montrer
      <attention>le mot de passe &#133;</attention>
      Ã§a
    </secret>
  </liste_nombres>
</nombre>
```

# Modèles, validations

## XML

Comment donner un modèle à un document XML ?

Deux solutions :

- fournir une DTD
- fournir un schema

# Modèles, validations

## XML

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1 plus MathML 2.0//EN"
"http://www.w3.org/Math/DTD/mathml2/xhtml-math11-f.dtd" >
<html xml:lang="fr-FR" xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>TEST Mathml</title>
  </head>
  <body dir="ltr">
    <h1 id="toc0">Exemple de formule mathématique en mathML</h1>
    <p>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <mrow>
          <mrow>
            <mfrac>
              <mn>2</mn>
              <msqrt>
                <mrow>
                  <mn>5</mn>
                  <mo stretchy="false">&plus;</mo>
                  <mi>x</mi>
                </mrow>
              </msqrt>
            </mfrac>
          </mrow>
        </math>
      </p>
    </body>
  </html>

```

# Modèles, validations

## XML

Cas concret : réalisation de DTD et de schema xml

# Modèles, validations

## XML

`exemples/document.xml`



# Modèles, validations

## XML

`exemples/windob.dtd`

# Modèles, validations

## XML

`exemples/newWindob2.xsd`

# Plan

- 1 Introduction
- 2 Présentation des architectures

# Modèles, validations

## XML

espaces de nommage

# Présentation des architectures XSL

XSL : modèles et instances de modèles

Quels intérêts, quelles applications ?

- un langage exprimé en XML
- transformation d'arbres
- utilisation de XPATH, XQUERY
- optimisé pour les transformations

Attention, tous les parsers xsl n'implémentent pas xsl v2.

Le parser xsl de référence est SAXON

(<http://www.saxonica.com>)

Il existe des API de saxon pour java, dotnet, (C++ et PHP dans la dernière version).

# Présentation des architectures

## XSL

### XSL : Principes

- système de template activés par sélecteur
- variables non modifiables
- possibilité de paramétrer un template, une feuille
- arrêt des traitements lorsqu'il n'y a plus rien à faire
- plusieurs modes de sortie possible : html, xml, text









# Présentation des architectures

## XSL

### XSL : sélection avec XPATH

*noeuds[restrictions]/enfants[restriction]*

#### Exemples :

-  *noeud/\** : tous les fils de noeud
-  *noeud/@\** : tous les attributs de noeud
-  *noeud/\*|noeud/@\** : tous les fils et les attributs de noeud
-  *noeud/a* : tous les fils de noeud de type a
-  *noeud/b[1]* : le premier fils de type b de noeud
-  *noeud/b[position()gt;1 and @style="gras"]* : le 5ème des fils de type b de noeud qui n'est pas le premier fils de type b de noeud et qui a un attribut style valant "gras"
-  *..* : le noeud père
-  *//a* : tous les noeuds de type a du document ; SYNTAXE A

EVITER

# Présentation des architectures XSL

## XSL : sélecteurs avec XPATH

*selecteur : :noeuds[restrictions]..*

### Exemples :

- child : :\* : tous les fils*
- preceding : :\* et following : :\* : tous les noeuds précédents/suivant*
- preceding-sibling : :\* et following-sibling : :\* : tous les noeuds précédents/suivant du même niveau*
- ancestor : tous les ancêtres du noeud*
- descendant : tous les descendants du noeud*



# Présentation des architectures

## XSL

### XSL :fonctions XPATH

#### fonctions XPATH

- *a[count(preceding : :b)=1]* : tous les fils a précédés par au moins un noeud b
- *a[position()=last()]* : le dernier des fils a
- *child : :\*[local-name()="a"]* : tous les enfants de type a

autres fonctions : translate, concat, number, etc.

# Présentation des architectures

## XSL

### XSL :instructions (exemples)

- template match et apply-templates
- template name et call-template
- variables
- if
- when/otherwise
- for-each

# Présentation des architectures XSL

XSL : version 1.0, 2.0 et 3.0

principaux ajouts de 2.0

- définition de fonctions
- typage
- xquery
- API des fonctions augmentée

principaux ajouts de 3.0

- variables modifiables
- objets
- ...

# Présentation des architectures

## XSL

```
<?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="html"
  encoding="ISO-8859-1"
  doctype-public="-//W3C//DTD_HTML_4.01//EN"
  doctype-system="http://www.w3.org/TR/html4/strict.dtd"
  indent="yes" />

<xsl:template match="liste_nombres">
  <html><body>
    <p>Liste de nombres :</p>
    <ul>
      <xsl:apply-templates select="*" />
    </ul>
  </body></html>
</xsl:template>

<xsl:template match="nombre">
  <li>
    <xsl:value-of select="@valeur" />
    <xsl:text> : </xsl:text>
    <xsl:value-of select="." />
  </li>
</xsl:template>

<xsl:template match="secret"/>

</xsl:stylesheet>
```