

Transformer des documents XML

1

Introduction

- Désignation d'un nœud (sous-arbre) d'un document XML : XPATH
- Associer une présentation à un document XML : CSS
- Transformer un document XML en un autre document XML (HTML) : XSLT
- Transformer un document XML en un document imprimable : XSL-FO
- Adapter le format de sortie d'un document XML au client : COCOON

2

XPath

3

XPath

- Langage d'expressions de chemins dans un arbre XML
- permet de désigner un nœud (sous-arbre) dans un arbre XML
- langage utilisé dans d'autres langages XML (XSLT, XLink, XQuery...)

4

Expressions de chemin

- Un **chemin** est composé d'une suite de **pas** qui sont des tronçons de chemin entre deux nœuds consécutifs.
- Un pas est caractérisé par :
 - un **axe (indicateur de relation)** qui spécifie la direction à suivre pour atteindre le nœud suivant (parent, ancestor, child, descendant, etc.) ainsi qu'un ordre de parcours des suivants possibles.
 - un **test de nœud (filtre)** qui spécifie le type du nœud suivant,
 - une suite de **conditions (prédicats)** sur le nœud suivant et sur son rang parmi les nœuds suivants possibles.
- Un chemin peut être **absolu** (il commence à la racine du document) ou **relatif** (son début dépend du contexte).

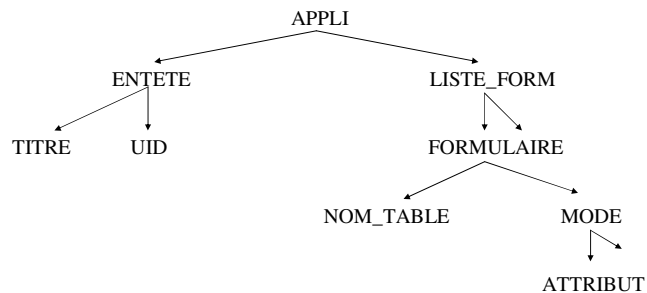
5

Exemple de document XML

```
<?xml version="1.0"?>
<APPLI>
  <ENTETE>
    <UID>citcom/citcom@MICA</UID><TITRE>MonAppli</TITRE>
  </ENTETE>
  <LISTE_FORM>
    <FORMULAIRE>
      <NOM_TABLE>VINS</NOM_TABLE>
      <MODE valeur="QBE"/>
    </FORMULAIRE>
    <FORMULAIRE>
      <NOM_TABLE>RECOLTES</NOM_TABLE>
      <MODE valeur="NOR">
        <ATTRIBUT>CRU</ATTRIBUT>
        <ATTRIBUT>NUM</ATTRIBUT>
      </MODE>
    </FORMULAIRE>
  </LISTE_FORM>
</APPLI>
```

6

Représentation arborescente



7

Exemples d'expressions Xpath (1)

- Donner l'élément de nom UID :
`/APPLI/ENTETE/UID` ou `//UID`
- Donner tous les fils de l'élément de nom APPLI :
`/APPLI::*`
- Donner tous les nœuds de type texte de l'élément de nom ENTETE
`/APPLI/ENTETE::text()`

8

Exemples d'expressions Xpath (2)

- Donner les éléments de nom FORMULAIRE dont l'élément fils MODE a un attribut valeur à NOR
`//FORMULAIRE/MODE/@valeur="NOR"`
- Donner le dernier fils FORMULAIRE de l'élément de nom LISTE_FORM
`//LISTE_FORM/FORMULAIRE[position=last()]`

9

Indicateurs de relation

- child : fils du noeud courant
- descendant : tout le sous arbre du noeud courant
- parent : élément père du noeud courant
- attribute : attributs du noeud courant
- self : noeud courant
- ancestor : ascendants dans l'ordre inverse d'apparition dans le document
- ...

10

Filtres

- Nom d'élément : descendant::section
- nom d'attribut : attribut::format
- * (tous les objets de même nature) : attribut::* ou child::*
- nom-élément[expression]
- comment() : /descendant::comment()
- text() : child::text()
- processing-instructions(nom)

11

Prédicats

- Prédicat complexe : combinaison booléenne AND-OR de prédicats simples
- prédicat simple :
 - expressions arithmétiques
 - fonctions sur les noeuds
 - fonctions sur les chaînes de caractères
 - fonctions booléennes
 - fonctions numériques

12

Fonctions sur les noeuds

- Last() : nombre de noeuds inclus dans le contexte courant
- position() : retourne le numéro d'ordre du noeud courant
- count(liste de noeuds)
- name(liste de noeuds)
- id("ch1 ch2 ch3") : sélectionne les éléments identifiés par ch1, ...

13

CSS Cascading Style Sheet

14

Feuilles de style

- Séparées du document XML
- Différents langages
 - Cascading Style Sheets Level 1 et 2 (CSS1 et CSS2)
 - Supportées par les navigateurs récents
 - Extensible Style Language (XSLT)
 - En général, pas supporté par les navigateurs (sauf via des plug-ins)
 - convertisseurs spécifiques

15

xml-styleheet

- Feuilles de style sont référencées par une instruction xml-styleheet

```
<?xml version="1.0" standalone="yes"?>
<?xml-styleheet type="text/css" href="genappli.css"?>
<APPLI>...</APPLI>
```
- le référencement peut se faire de manière externe (argument d'un convertisseur)

16

genappli.css

```
APPLI      {display: block;
            font-size: 24pt;
            font-weight: bold}
FORMULAIRE {font-size: 18pt;
            font: Arial}
```

17

XSLT

18

XSLT

- Une feuille de style XSLT est un document XML
- XSLT est un langage de transformation d'arbre
- puissance d'un langage de programmation (variables, itération, conditionnelle, ...)
- langage à base de règles (templates)
 - template = sélecteur + forme à générer
 - application des règles basée sur un parcours récursif de l'arbre et des priorités

19

Premier exemple XSLT

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">
  <doc>
    <title>Le premier exemple XSLT</title>
    <body>
      <xsl:apply-templates/>
    </body>
  </doc>
</xsl:template>
</xsl:stylesheet>
```

20

Exemple de document XML

```
<?xml version="1.0"?>
<APPLI>
  <ENTETE>
    <UID>citcom/citcom@MICA</UID><TITRE>MonAppli</TITRE>
  </ENTETE>
  <LISTE_FORM>
    <FORMULAIRE>
      <NOM_TABLE>VINS</NOM_TABLE>
      <MODE valeur="QBE"/>
    </FORMULAIRE>
    <FORMULAIRE>
      <NOM_TABLE>RECOLTES</NOM_TABLE>
      <MODE valeur="NOR">
        <ATTRIBUT>CRU</ATTRIBUT>
        <ATTRIBUT>NUM</ATTRIBUT>
      </MODE>
    </FORMULAIRE>
  </LISTE_FORM>
</APPLI>
```

21

Résultat de la transformation

```
<doc><title>Le premier exemple XSLT</title><body>
  citcom/citcom@MICA MonAppli VINS RECOLTES CRU NUM
</body></doc>
```

22

Règles implicites

- R1 : si aucune règle ne s'applique aux éléments du source, la recherche récursive s'applique jusqu'aux feuilles
- R2 : lorsqu'un élément contient du texte (#PCDATA), ce texte est par défaut recopié dans le noeud courant de l'arbre généré

23

Principe d'exécution des règles premier exemple

- La règle de la feuille XSL s'applique et produit `<doc><title>Le premier exemple XSLT</title><body>`
- elle fait ensuite un appel récursif sur d'autres règles `<xsl:apply-templates/>`
- comme il n'y a pas d'autres règles, les règles implicites s'appliquent :
 - entete -> R1
 - uid (texte) -> R2 -> R1 ...

24

Deuxième exemple

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">
  <html><head><title>Un exemple de HTML genere</title></head><body>
    <xsl:apply-templates/>
  </body></html>
</xsl:template>
<xsl:template match="ENTETE"></xsl:template>
<xsl:template match="LISTE_FORM">Liste des formulaires<p>
  <ol><xsl:apply-templates/></ol>
</xsl:template>
<xsl:template match="FORMULAIRE">
<li><xsl:value-of select="NOM_TABLE"/> : <xsl:value-of select="MODE/@valeur"/>
</li> <p>
</xsl:template>
</xsl:stylesheet>
```

25

Résultat de la transformation (2)

```
<html><head><title>Un exemple de HTML genere</title></head><body>
Liste des formulaires<p>
<ol>
<li>VINS : INS</li><p>
<li>RECOLTES : NOR</li><p>
</ol>
</body></html>
```

26

Transformation avec for each

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
xmlns="http://www.w3c.org/TR/REC-html40">
<xsl:template match="/">
<html><head><title>Formulaires</title></head>
<body><table><tr><th>Nom table</th><th>Mode</th></tr>
<xsl:for-each select="//FORMULAIRE"><tr>
  <td><xsl:value-of select="NOM_TABLE"/></td>
  <td><xsl:value-of select="MODE/@valeur"/></td>
</tr></xsl:for-each>
</table></body></html>
</xsl:template>
</xsl:stylesheet>
```

27

Résultat de la transformation

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<html xmlns="http://www.w3c.org/TR/REC-html40">
<head><title>Formulaires</title></head>
<body>
<table>
  <tr><th>Nom table</th><th>Mode</th></tr>
  <tr><td>VINS</td><td>INS</td></tr>
  <tr><td>RECOLTES</td><td>NOR</td></tr>
</table>
</body>
</html>
```

28

Conditionnelle

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
xmlns="http://www.w3c.org/TR/REC-html40">
<xsl:template match="/">
<html><head><title>Formulaires</title></head>
<body><table><tr><th>Nom table</th><th>Mode</th></tr>
<xsl:for-each select="//FORMULAIRE"><tr>
  <xsl:if test="position() mod 2 = 0">
    <xsl:attribute name="bgcolor">yellow</xsl:attribute></xsl:if>
  <xsl:if test="position() mod 2 != 0">
    <xsl:attribute name="bgcolor">blue</xsl:attribute></xsl:if>
  <td><xsl:value-of select="NOM_TABLE"/></td>
  <td><xsl:value-of select="MODE/@valeur"/></td>
</tr></xsl:for-each>
</table></body></html>
</xsl:template>
</xsl:stylesheet>
```

29

Autres constructions XSLT

- Variables et paramètres
- priorité sur les règles, règles nommées
- tris
- ...
- On peut tout faire avec XSLT, mais pas toujours simplement (il vaut alors mieux utiliser un programme SAX ou DOM)

30

CSS ou XSLT ?

- CSS est mieux supporté (côté client surtout)
- CSS est plus stable
- XSLT est beaucoup plus puissant (transformateur d'arbre et pas seulement présentation)
- XSLT peut générer du CSS

31

Publication de documents XML sur le web

32

XML et Web

- XML offre une séparation claire entre structure logique (contenu) et présentation
- permet de proposer des chaînes de production de documents sous différentes présentations à partir du même contenu (HTML vs PDF, WML vs HTML,...)

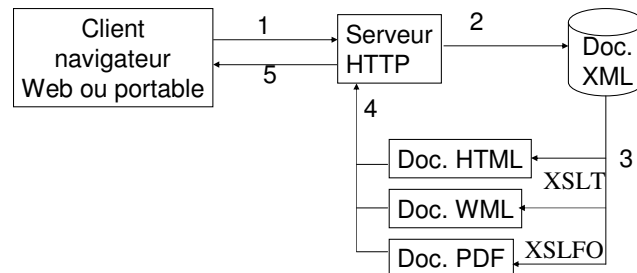
33

Environnements de publication XML

- Environnement logiciel permettant de servir différentes présentations de documents XML
- Cocoon de XML-Apache, Qweelt de Univ. Penn.
- Souvent basé sur des implantations Java (Servlet, JSP)

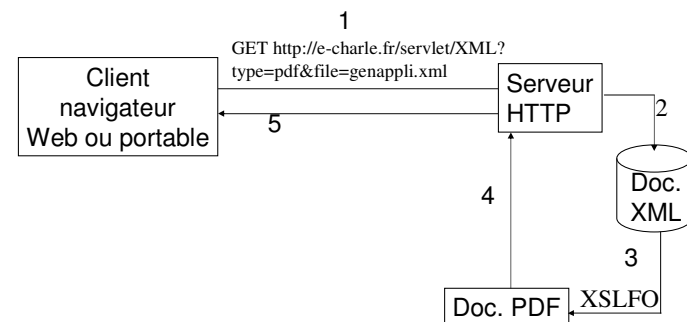
34

Publication sur le web



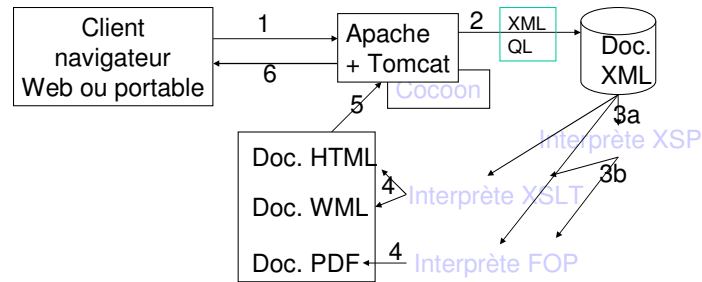
35

Publication sur le web



36

Cocoon



37

Bilan

- XPATH est un véritable langage de requête pour document XML (restreint à un seul document)
- XSLT permet toutes les transformations que l'on veut, mais pas toujours simplement
- Support côté client est faible
- Environnement comme COCOON permet d'ores et déjà de produire n formats de sortie à partir d'un seul source XML

38