XPATH / XSLT

Olivier.Corby@sophia.inria.fr

http://www/inria.fr/acacia

Plan

1. XPath : XML Path Language
2. XSLT : Extensible Stylesheet Language Transformation
3. XQuery
4. XLink : XML Linking Language
5. XPointer

XPath

XML Path Language

XPath

- http://www.w3.org/TR/xpath
- Langage de chemin pour
  - Accéder, désigner une portion de document
  - Filtrer des éléments sur la base d'un *pattern* XPath (XSLT)
  - Exprimer des portions de requête
- Exploite la structure du document dans la navigation
- Axes de parcours
- Test sur la valeur des éléments et des attributs
- Fonctions

XPath Exemple

XPath :

```
/talk/slide/title   ->   <title>XPath Exemple</title>
```

Document :

```
<talk>
        <slide>
                <title>XPath Exemple</title>
                ...
        </slide>
</talk>
```

XPath Exemple

XPath :

```
/talk/slide[2]/title   ->   <title>XLink</title>
```

Document :

```
<talk>
        <slide>
                <title>XPath Exemple</title>
                ...
        </slide>
        <slide>
                <title>XLink</title>
                ...
        </slide>
</talk>
```

XPath Exemple

XPath :

```
/talk/slide[@id='xlk']/title -> <title>XLink</title>
```

Document :

```
<talk>
        <slide id='xpt'>
                <title>XPath Exemple</title>
                ...
        </slide>
        <slide id='xlk'>
                <title>XLink</title>
                ...
        </slide>
</talk>
```

XPath :

```
/talk/slide[title='XLink']/title -> <title>XLink</title>
```

Fils

Tous les attributs :

```
/talk/@*
```

Tous les fils :

```
/talk/node()
```

Tous les fils élément :

```
/talk/*
```

Tous les fils texte :

```
/talk/text()
```

Accesseurs

Parent

```
../
```

Union de exp1 et exp2 :

```
 exp1 | exp2
```

XPath Exemple

Tous les descendants de type para ou section :

```
/talk/descendant::*[self::para or self::section]
```

Tous les descendants de type section, ou de type para qui ne contiennent pas de citation :

```
/talk/descendant::*[(self::para and not(citation)) or self::section]
```

Axes XPath

Axes de parcours de l'arbre XML

`self`	noeud courant	`preceding`	ancêtres exclus
`child`	axe par défaut	`following`	descendants exclus
`descendant`		`preceding-sibling`	frêres précédents
`descendant-or-self`		`following-sibling`	frêres suivants
`parent`		`attribute`	
`ancestor`		`namespace`	
`ancestor-or-self`			

Axes XPath

Ces axes forment une partition et contiennent tous les noeuds de l'arbre

- preceding
- following
- ancestor
- descendant
- self

Exemples Axe XPath

Le premier transparent d'une partie précédente dont le titre est XML :

```
preceding::part[title='XML']/slide[1]
```

La section suivante dont l'attribut id est XSLT :

```
preceding::section[title='XML']/following::section[@id='XSLT']
```

Fonctions prédéfinies

- position()
- last()
- contains(elem, elem)
- concat(elem, elem)
- starts-with(elem, elem)
- substring(elem, elem)
- count(elem)
- id(elem)

- name(elem)
- local-name(elem)
- namespace-uri(elem)

Exemples

```
/talk/slide[contains(title, 'XSLT')]

/talk/slide[count(*)>5]
```

XPath 2.0

Extensions de XPath Pour XQuery

- Sequence : collection ordonnée de valeurs ou de noeuds
- union, intersection, except

Expressions

```
if (Exp) then Exp else Exp

some | every Variable in Exp satisfies Exp

every $person in //team satisfies $person/@fulltime
```

Exercice

Ecrire les expressions XPath pour :

1. Le successeur de mon parent
2. Mon grand-parent
3. Mon ancêtre de type part
4. Mes frêres suivants
5. Mes frêres et moi
6. Mes frêres, sauf moi
7. Mes cousins germains

XSL

Extensible Stylesheet Language

XSL : Extensible Stylesheet Language

- Feuilles de style
- Extensible Stylesheet Language, inspiré de DSSSL (formatage de SGML)
- Spécifier la présentation d'un document XML
  - **XSLT** : Transformation de documents
  - **FO** : Mise en page de documents, Formatting Object

XSLT : XSL Transformations

http://www.w3.org/TR/xslt

- Langage de transformation/formatage
- Transforme un document XML en un autre document XML
- Peut engendrer du HTML
- Langage à base de règles
- Une règle s'applique à un type d'élément XML
- Opérateurs
- XPath pour filtrer des patterns et accéder au document

Transformations XSLT

- Extraire des informations
- Restructurer
  - Engendrer une table des matières
  - Engendrer un tableau à partir de listes d'éléments
- Combiner des informations de provenances différentes

Exemple de transformation

```
<chapter>
        <title>XML</title>    -->      <h1>XML</h1>
```

Règles de transformation :

```
<xsl:template match='chapter'>

        <h1> <xsl:value-of select='title'/> </h1>

</xsl:template>
```

h1 est un élément engendré par le template dans le flux de sortie

Engendrer du HTML

```
<xsl:template match='/'>

    <html>
        <head>
                <title><xsl:value-of select='doc/title'/></title>
        </head>
        <body>
                <xsl:apply-templates/>
        </body>
    </html>

</xsl:template>
```

Template

`xsl:apply-templates`	appliquer les règles sur les fils
`xsl:apply-templates select='EXP'`	appliquer les règles sur EXP

Test

```
<xsl:if test='self::para or section'>

        <br/>

</xsl:if>
```

Choix multiples

```
<xsl:choose>

   <xsl:when test='position() = 1'>

   </xsl:when>


   <xsl:otherwise>

   </xsl:otherwise>

</xsl:choose>
```

Itérateur

Insérer le titre de tous les slides :

```
<xsl:for-each select='talk/slide'>
        <xsl:value-of select='title'/> <br/>
</xsl:for-each>
```

Variable locale

```
<xsl:variable name='current'>
        <xsl:value-of select='section'/>
</xsl:variable>
```

Utilisation :

```
<xsl:value-of select='$current' />
```

Une variable liée ne peut être modifiée

Exemple XSLT

Insérer le titre de la partie courante :

```
<xsl:value-of select='ancestor::part/title'/>
```

Compter le nombre de sections du chapitre :

```
<xsl:template match='chapter'>
        <xsl:value-of select='count(section)'/>
</xsl:template>
```

Créer un lien hypertexte

Les accolades { } signifient qu'il faut évaluer l'expression qui se trouve à l'intérieur, ici : `{id}`

```
<xsl:template match='lien'>
   <a href='#{id}'> <xsl:value-of select='title' /> </a>
</xsl:template>
```

Appliqué à :

```
<lien>
   <id>abc</id>
   <title>Voir plus loin</title>
</lien>
```

Donne :

```
<a href='#abc'>Voir plus loin</a>
```

Créer un élément

Le nom de l'élément est calculé :

```
<xsl:element name='pref:{local-name()}'>
        <xsl:value-of select='content' />
</xsl:element>
```

Créer un attribut

```
<xsl:attribute name='pref:{local-name()}'>
        <xsl:value-of select='content' />
</xsl:attribute>
```

Copier un élément

```
<xsl:template match='test'>
    <xsl:copy>
        <xsl:apply-templates select='*' />
    </xsl:copy>
</xsl:template>
```

Copier un document

```
<xsl:template match='@* | node()'>
    <xsl:copy>
        <xsl:apply-templates select='@* | node()' />
```

```
    </xsl:copy>
</xsl:template>
```

Traiter plusieurs documents

Une feuille de style peut lire un (ou plusieurs) autre document XML :

```
<xsl:variable name='doc' select='document("profile.xml")'/>
```

Ce nouveau document XML peut être traité comme le document courant :

```
<xsl:value-of select='$doc/book/chapter[title="Guerre et Paix"]' />

<xsl:apply-templates select='.'>
        <xsl:with-param name='profile' select='$doc' />
</xsl:apply-templates>
```

Template avec mode

Template avec un mode particulier :

```
<xsl:template match='book' mode='index'>

</xsl:template>


<xsl:template match='book' mode='full'>

</xsl:template>
```

Appel du template avec un mode :

```
<xsl:apply-templates select='book' mode='index' />
```

Template avec un nom

```
<xsl:template name='test'>

   <xsl:param name='p' />


   <xsl:value-of select='$p' />

</xsl:template>
```

```
<xsl:call-template name='test'>
        <xsl:with-param name='p' select='.' />
</xsl:call-template>
```

Template récursif

```
<xsl:template name='clean'>
<xsl:param name='str' />
<xsl:param name='obj' />

<xsl:choose>

<xsl:when test='contains($str, $obj)'>
    <xsl:call-template name='clean'>

        <xsl:with-param name='str'>
            <xsl:value-of select='concat(substring-before($str, $obj),
                                         substring-after($str, $obj))'/>
        </xsl:with-param>
        <xsl:with-param name='obj' select='$obj'/>

    </xsl:call-template>
</xsl:when>

<xsl:otherwise>
        <xsl:value-of select='$str'/>
</xsl:otherwise>

</xsl:choose>
</xsl:template>
```

Engendrer un tableau trié

Classer les personnes par ville, le tout trié par ordre alphabétique

```
<xsl:template match="sort">
<table>
<xsl:for-each select='person/city[not(preceding::person/city=.)]'>
        <xsl:sort/>
        <tr>
                <td><xsl:value-of select='.'/></td>

                <xsl:for-each select='ancestor::sort/person[city=current()]
                        <xsl:sort/>
                        <td><xsl:value-of select='name'/></td>
                </xsl:for-each>
        </tr>
</xsl:for-each>
</table>
```

```
</xsl:template>
```

```
<sort>
        <person><name>Fabien</name>     <city>Antibes</city></person>
        <person><name>Patrick</name>    <city>Nice</city></person>
        <person><name>Olivier</name>    <city>Auribeau</city></person>
        <person><name>Alain</name>      <city>Antibes</city></person>
</sort>
```

Résultat :

Antibes	Alain	Fabien
Auribeau	Olivier	
Nice	Patrick	

Traitement de XSLT

Mozilla et Explorer possèdent un moteur de feuilles de styles XSLT

Intégrer l'instruction suivante dans le document XML

```
<?xml-stylesheet href="talk.xsl" type="text/xsl"?>
```

API XSLT Java 1.4

Java 1.4 contient un moteur XSLT dans le package javax.xml.transform

```
import javax.xml.transform.stream.*;
import javax.xml.transform.*;

class XSLT {

  public static void main(String args[]){
    String xslt=args[0];
    String xml =args[1];
    try{
      TransformerFactory tFactory = TransformerFactory.newInstance();

      Transformer transformer =
                tFactory.newTransformer(new StreamSource(xslt));

      transformer.transform(new StreamSource(xml),
                            new StreamResult(System.out));

    }
    catch (Exception e){
      e.printStackTrace();
    }
  }

}
```

Pour lancer le moteur de feuille de style :

```
java XSLT file.xsl file.xml
```

Exercice

Ecrire des feuilles de style pour

1. Engendrer une table des matières : liste numérotées des titres des parties
2. Engendrer une table HTML à partir d'une liste :

```
<liste>
        <elem name='a'>aaa</elem>
        <elem name='b'>bbb</elem> ..
</liste>
```

-->

```
<table>
        <tr><td>name</td><td>value</td></tr>
        <tr><td>a</td><td>aaa</td></tr>
        <tr><td>b</td><td>bbb</td></tr>
</table>
```

XQuery

http://www.w3.org/TR/xquery

- Pour interroger des serveurs Web/XML
- Interrogation et transformation
- Une sorte de *SQL* pour XML

FLWR Expressions

Flower expressions : For Let Where Return

```
for Variable in Exp
where Exp return Exp sortby(Exp)

let Variable := Exp
where Exp return Exp sortby(Exp)
```

FLWR Expressions

```
<publisher_list>
     {for $p in distinct-values(document("bib.xml")//publisher)
      return
          <publisher>
              <name> {$p/text()} </name>
              {for $b in document("bib.xml")//book[publisher = $p]
                  return
                     <book>
                        {$b/title}
                        {$b/price}
                     </book>
                  sortby(price descending)
              }
          </publisher>
      sortby(name)
     }
</publisher_list>
```

Type expression

```
element of type animal

typeswitch($animal)
        case element duck return xx
        case element dog  return yy
```

XLink : XML Linking Language

http://www.w3.org/TR/xlink

- W3C Recommandation 27 Juin 2001
- Etendre les liens hypertexte de HTML
- Inspiré de HyTime et TEI
- Associer de la sémantique aux liens et aux ressources
- Liens externes
- Liens bidirectionnels
- Liens n-aires

XLink

- Réalisé par des attributs prédéfinis dans un namespace :

http://wwww.w3.org/1999/xlink

- Tout élément XML peut *être* un lien hypertexte
- Exemple:

```
<a xmlns:xlink='http://wwww.w3.org/1999/xlink'
  xlink:type='simple'
  xlink:href='students.xml'
  xlink:title='Student List'
  xlink:show='new'
  xlink:actuate='onRequest'>

Current List of Students

</a>
```

Liens étendus

- Un lien peut être défini hors du document source : lien externe
- Le browser charge le document source plus un document définissant des liens
- Peut mettre en jeu plus de deux ressources
- Définir des éléments de type :
  - extended
  - locator : un URI, un label, un title
  - resource : ressource locale
  - arc : relie des labels (et donc des locators)

Lien étendu : Exemple

Lien étendu :

```
<sitelist xlink:type='extended'>
```

Avec une ressource locale et un locator soft :

```
<local xlink:type='resource' xlink:label='soft' xlink:title='Mon Super Soft
       Description locale
</local>

<loc xlink:type='locator' xlink:label='soft'
       xlink:href='http://..here.xml' xlink:title='Super Soft'/>
```

et deux locators server :

```
<loc xlink:type='locator' xlink:label='server'
       xlink:href='http://..s1.xml' xlink:title='Chez moi'/>

<loc xlink:type='locator' xlink:label='server'
       xlink:href='http://..s2.xml' xlink:title='Chez eux'/>
```

Une description d'arc entre soft et server :

```
<go xlink:type='arc' xlink:from='soft' xlink:to='server'/>

</sitelist>
```

Attributs de comportement

affichage du document cible

```
xlink:show
        (new |     -- nouvelle fenetre ou frame
        replace |  -- charger la ressource dans la même fenêtre
        embed |    -- prend la place du lien dans le document source
        other |    -- d'autres attributs peuvent donner des indications
        none)      -- pas d'indications
```

parcours du lien

```
xlink:actuate
        (onLoad |      -- au chargement du document
        onRequest |    -- à la demande de l'utilisateur (click, etc.)
        other |        -- indications possibles dans d'autres attributs
        none)          -- pas d'indication
```

XPointer Framework

http://www.w3.org/TR/xptr-framework

- Peut être utilisé dans un URI pour désigner un élément de document
- Pour désigner des portions de documents XML
- Différents schema de pointeurs : extensible

XPointer : Exemple

Element d'ID `intro` :

```
http://www.ybroc.org#intro
```

3ème fils du 2ème fils de l'élément d'ID `intro`

```
element(intro/2/3)
```