

Limbaje de interogare pentru XML

XPath

XQuery

XSLT

Modelul de Date XPath/XQuery

- Corespondentul “relației” din modelul relațional este: *secvența de articole*.
- Un *articol* este unul din următoarele:
 1. O valoare primitivă, de exemplu: integer sau string.
 2. Un *nod*.

Principalele Tipuri de Noduri

1. *Noduri Document* ce reprezintă documente în întregime.
2. *Elemente* ce sunt bucăți de document ce constau din câteva tag-uri de început și de sfârșit, și tot ceea ce există între ele.
3. *Attribute* ce sunt nume ce primesc valori în interiorul tag-urilor de început.

Noduri Document

- Formulate prin `doc(URL)` sau `document(URL)`.
- **Exemplu:** `doc(/usr/class/cs145/bars.xml)`
- Interogările XPath (și XQuery) fac referire la un nod document fie explicit, fie implicit.
 - **Exemplu:** definițiile cheilor în Schema XML folosesc expresii Xpath ce fac referire la documentul descris de către schemă.

DTD pentru Exemplu

```
<!DOCTYPE BARS [  
  <!ELEMENT BARS (BAR*, BEER*)>  
  <!ELEMENT BAR (PRICE+)>  
    <!ATTLIST BAR name ID #REQUIRED>  
  <!ELEMENT PRICE (#PCDATA)>  
    <!ATTLIST PRICE theBeer IDREF #REQUIRED>  
  <!ELEMENT BEER EMPTY>  
    <!ATTLIST BEER name ID #REQUIRED>  
    <!ATTLIST BEER soldBy IDREFS #IMPLIED>  
>
```

Exemplu Document

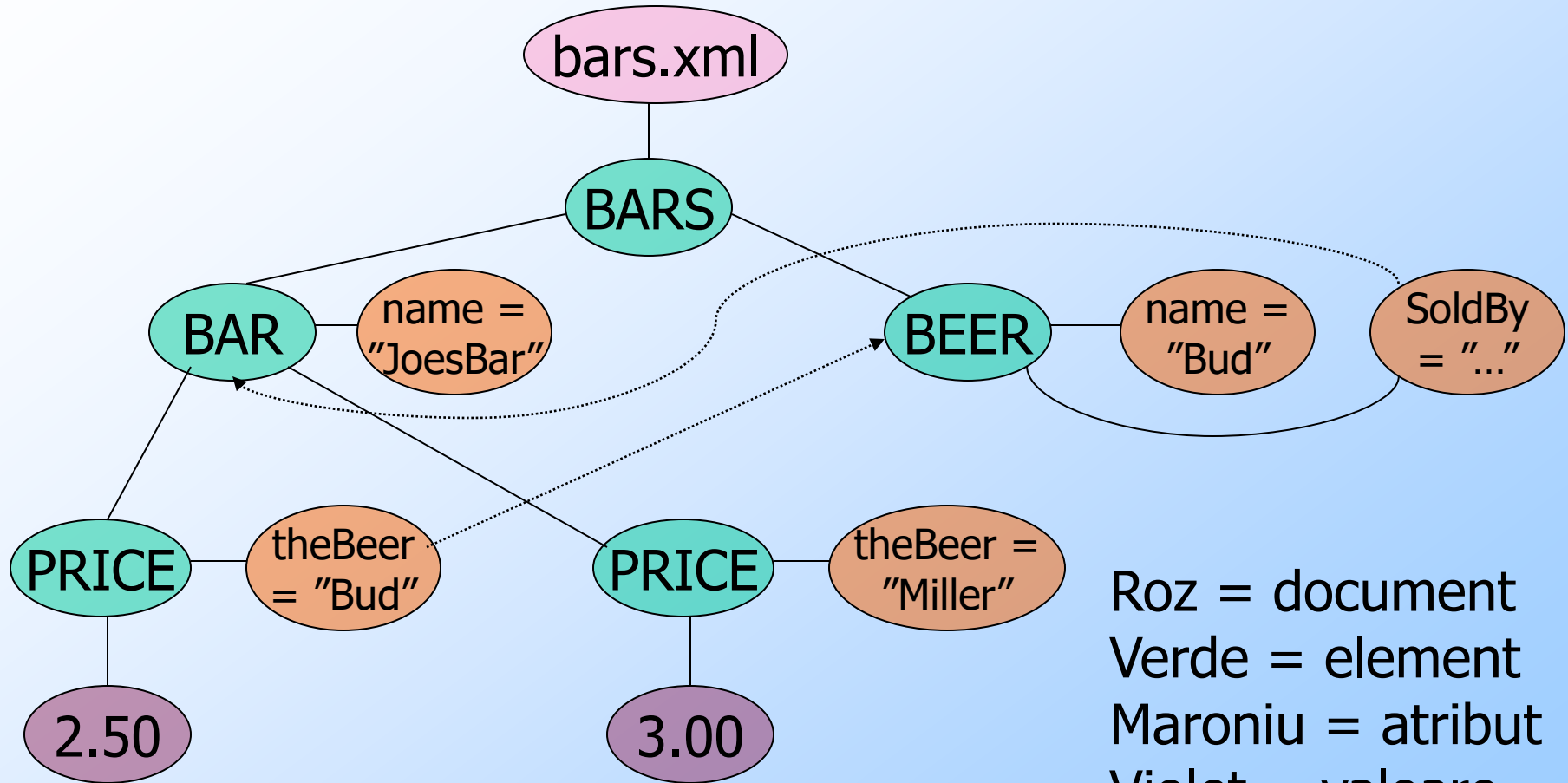
Un nod element

```
<BARS>  
  <BAR name = "JoesBar">  
    <PRICE theBeer = "Bud">2.50</PRICE>  
    <PRICE theBeer = "Miller">3.00</PRICE>  
  </BAR> ...  
  <BEER name = "Bud" soldBy = "JoesBar  
    SuesBar ... "> ...  
</BARS>
```

Un nod atribut

Nodul document este totul de mai sus,
plus header-ul (<? xml version...).

Noduri, Date Semistructurate



Roz = document
Verde = element
Maroniu = atribut
Violet = valoare
primitivă

Căi în Documente XML

- XPath este un limbaj cu ajutorul căruia se descriu *căi* în documente XML.
- Rezultatul căii descrise este o secvență de articole.

Expresii cale

- Expresii cale simple sunt secvențe de slash-uri (/) și tag-uri, ce încep cu /.
- Exemplu: /BARS/BAR/PRICE
- Construirea rezultatului se face pornind de la nodul document, prin procesarea fiecărui tag de la stânga.

Evaluarea unei Expresii Cale

- Se presupune că primul tag este rădăcina.
 - Procesarea nodului document prin acest tag conduce la o secvență ce constă doar din elementul rădăcină.
- Să presupunem că avem o secvență de articole, și că tag-ul următor este X .
 - Pentru fiecare articol ce este un nod element, se înlocuiește elementul cu subelementele ce au tag-ul X .

Exemplu: /BARS

<BARS>

<BAR name = "JoesBar">

<PRICE theBeer = "Bud">2.50</PRICE>

<PRICE theBeer = "Miller">3.00</PRICE>

</BAR> ...

<BEER name = "Bud" soldBy = "JoesBar
SuesBar ... "> ...

</BARS>

Un articol,
elementul BARS

Exemplu: /BARS/BAR

<BARS>

<BAR name = "JoesBar">

<PRICE theBeer = "Bud">2.50</PRICE>

<PRICE theBeer = "Miller">3.00</PRICE>

</BAR> ...

<BEER name = "Bud" soldBy = "JoesBar
SuesBar ..."/> ...

</BARS>

Acest element BAR urmat de
toate celelalte elemente BAR

Exemplu: /BARS/BAR/PRICE

<BARS>

<BAR name = "JoesBar">

<PRICE theBeer = "Bud">2.50</PRICE>

<PRICE theBeer = "Miller">3.00</PRICE>

</BAR> ...

<BEER name = "Bud" soldBy = "JoesBar
SuesBar ..."/> ...

</BARS>

Elementele PRICE afișate,
urmate de elementele PRICE
ale celorlalte baruri.

Atribute în Căi

- În loc să se menționeze subelemente cu un anumit tag, se poate menționa un atribut al elementelor.
- Un atribut este indicat cu prefixul @ înaintea numelui.

Exemplu:

/BARS/BAR/PRICE/@theBeer

<BARS>

<BAR name = "JoesBar">

<PRICE theBeer = "Bud">2.50</PRICE>

<PRICE theBeer = "Miller">3.00</PRICE>

</BAR> ...

<BEER name = "Bud" soldBy = "JoesBar

SuesBar ..."/> ...

</BARS>

Aceste attribute, "Bud" "Miller",
participă la rezultat,
urmate de alte valori theBeer.

Recapitulare: Secvențe de Articole

- Până acum, toate secvențele de articole au fost secvențe de elemente.
- Atunci când o expresie cale se termină cu un atribut, rezultatul este de obicei o secvență de valori de tip primitivă, cum au fost șirurile de caractere din exemplul precedent.

Căi ce încep oriunde

- Atunci când calea pornește de la nodul document și începe cu `//X`, atunci primul pas poate începe cu rădăcina sau cu orice subelement al rădăcinii, atât timp cât tag-ul este `X`.

Exemplu: //PRICE

<BARS>

<BAR name = "JoesBar">

<PRICE theBeer = "Bud">2.50</PRICE>

<PRICE theBeer = "Miller">3.00</PRICE>

</BAR> ...

<BEER name = "Bud" soldBy = "JoesBar
SuesBar ..."/> ...

</BARS>

Elementele PRICE indicate și
orice alte elemente PRICE
în întregul document

Wild-Card *

- Asterix (*) în locul unui tag reprezintă orice tag.
- **Exemplu:** /*/*/PRICE reprezintă toate obiectele price la al treilea nivel de imbricare.

Exemplu: /BARS/*

Acest element BAR, toate celelalte elemente BAR, elementul BEER, toate celelalte elemente BEER

<BARS>

<BAR name = "JoesBar">

<PRICE theBeer = "Bud">2.50</PRICE>

<PRICE theBeer = "Miller">3.00</PRICE>

</BAR> ...

<BEER name = "Bud" soldBy = "JoesBar
SuesBar ... "> ...

</BARS>

Condiții de Selecție

- Unui tag îi poate urma o condiție între [...].
- Dacă apare, atunci numai acele căi sunt incluse în rezultat, ce au acel tag și satisfac condiția.

Exemplu: Condiție de Selecție

□ /BARS/BAR/PRICE[. < 2.75]

Elementul
curent.

<BARS>

<BAR name = "JoesBar">

<PRICE theBeer = "Bud">2.50</PRICE>

<PRICE theBeer = "Miller">3.00</PRICE>

</BAR> ...

Condiția PRICE < 2.75 (\$)
determină acest preț participă la
rezultat, dar prețul berii Miller
nu participă la rezultat.

Exemplu: Atribut în Selecție

□ /BARS/BAR/PRICE[@theBeer = "Miller"]

<BARS>

<BAR name = "JoesBar">

<PRICE theBeer = "Bud">2.50</PRICE>

<PRICE theBeer = "Miller">3.00</PRICE>

</BAR> ...

↑
De această dată, acest
element PRICE este selectat,
împreună cu orice alte prețuri
pentru Miller.

Axe

- În general, expresiile cale permit să se pornească de la rădăcină și să se execute pași în încercarea de a găsi o secvență de noduri la fiecare pas.
- La fiecare pas, se poate urma una din mai multe *axe*.
- Axa implicită este **child::** --- merge la toți fiii setului curent de noduri.

Exemplu: Axe

- /BARS/BEER este de fapt o scurtătură pentru /BARS/child::BEER .
- @ este de fapt o scurtătură pentru axa **attribute::**.
 - Astfel, /BARS/BEER[@name = "Bud"] este scurtătura pentru
/BARS/BEER[attribute::name = "Bud"]



Mai multe Axe

- Alte axe folositoare sunt:
 1. **parent::** = părintele(-ții) nodului(-lor) curent(-e).
 2. **descendant-or-self::** = nodul(-rile) curent(-e) și toți descendenții.
 - Notă: // este de fapt scurtătura pentru această axă.
 3. **ancestor::**, **ancestor-or-self**, etc.
 4. **self** (punctul).

XQuery

- XQuery extinde XPath spre un limbaj de interogare similar cu SQL.
- Folosește modelul de date secvență-de-articole.
- XQuery este un limbaj expresie.
 - Asemănător algebrei relaționale --- orice expresie XQuery poate reprezenta argumentul altei expresii XQuery.

Secvențe de Articole

- XQuery va formula uneori secvențe de secvențe.
- Secvențele sunt pe nivel.
- **Exemplu:** (1 2  (3 4)) = (1 2 3 4).

secvența
vidă

Expresii FLWR

1. Una sau mai multe clauze **for** și/sau **let**.
2. În continuare o clauză opțională **where**.
3. O clauză **return**.

Semanticile Expresiilor FLWR

- Fiecare **for** crează o buclă.
 - **let** produce o definiție locală.
- La fiecare iterație a buclelor imbricate, dacă există, se evaluează clauza **where**.
- Dacă **where** returnează TRUE, se invocă clauza **return**, și se adaugă valoarea sa la ieșire.

Clauze FOR

for <variabilă> in <expresie>, . . .

- Variabilele încep cu \$.
- O variabilă **for** primește fiecare articol din secvența denotată prin expresie.
- Tot ceea ce urmează lui **for** se execută o dată pentru fiecare valoare a variabilei.

Documentul
exemplu BARS

Exemplu: FOR

“Expandează șirul
inclus prin înlocuirea
variabilelor și a
expresiilor cale
cu valorile lor.”

```
for $beer in  
  document("bars.xml")/BARS/BEER/@name  
return
```

<BEERNAME> {\$beer} </BEERNAME>

- \$beer ia valori din plaja atributelor “name” pentru toate mărcile de bere din documentul exemplu.
- Rezultatul este o secvență de elemente
BEERNAME : <BEERNAME>Bud</BEERNAME>
<BEERNAME>Miller</BEERNAME> . . .

Folosirea Marcatorilor

- Atunci când un nume variabil, de exemplu `$x`, sau o expresie, ar putea să fie text, este nevoie să folosim marcatori de jur împrejur pentru a evita interpretarea literală a ei.
- **Exemplu:** `<A>$x` este un element A cu valoarea `"$x"`, asemănător cu elementul A: `<A>foo` ce are valoarea `"foo"`.

Folosirea Marcatorilor

- `return $x` este lipsit de ambiguitate.
- Pentru a returna șirul de caractere, se include între ghilimele: `return "$x"`.

Clauze LET

let <variabilă> := <expresie>, . . .

- Valoarea variabilei devine *secvența* articolelor definite prin expresie.
- Notă **let** nu cauzează iterație, așa cum face **for**.

Exemplu: LET

```
let $d := document("bars.xml")
```

```
let $beers := $d/BARS/BEER/@name
```

```
return
```

```
<BEERNAMES> {$beers} </BEERNAMES>
```

□ Returnează un element cu toate numele mărcilor de bere, în felul următor:

```
<BEERNAMES>Bud Miller ...</BEERNAMES>
```

Clauze Order-By

- FLWR este în realitate FLWOR: o clauză order-by poate precede return.
- Forma: order by <expresie>
 - Opțional poate exista **ascending** sau **descending**.
- Expresia este evaluată pentru fiecare atribuire a variabilelor.
- Determină poziționarea în secvența de ieșire.

Exemplu: Order-By

- Lista prețurilor pentru berea Bud, începând cu cel mai mic preț.

```
let $d := document("bars.xml")
```

```
for $p in  
  $d/BARS/BAR/PRICE[@theBeer="Bud"]
```

```
order by $p
```

```
return $p
```

Ordonează legăturile după valorile din interiorul elementelor.

Generează legături pentru \$p la elemente PRICE.

Fiecare legătură este evaluată pentru ieșire. Rezultatul este o secvență de elemente PRICE.

Comparație: SQL ORDER BY

- SQL funcționează în același fel; ceea ce se ordonează este determinat de clauzele FROM și WHERE, nu ieșirea (clauza SELECT).

Exemplu: Fie $R(a,b)$,

```
SELECT b FROM R
```

```
WHERE b > 10
```

```
ORDER BY a;
```

Valorile "b" sunt extrase din tuplele relației R și sunt afișate în ordinea specificată.

Tuplele relației R cu $b > 10$ sunt ordonate după valorile "a".

Predicate

- În mod normal, condițiile implică existența cuantificării.
- **Exemplu:** /BARS/BAR[@name] semnifică "toate barurile ce au un nume".
- **Exemplu:** /BARS/BEER[@soldAt = "JoesBar"] dă setul mărcilor de bere ce sunt vândute la "Joe's Bar".

Exemplu: Comparații

- Să încercăm să obținem elementele PRICE (din toate barurile) pentru toate mărcile de bere ce sunt vândute de “Joe’s Bar”.
- Ieșirea va fi formată din elemente BBP cu numele barului și marca de bere ca attribute și elementul preț ca subelement.


Strategia

1. Se crează o buclă *for* triplă, cu variabile ce parcurg toate elementele BEER, toate elementele BAR, și toate elementele PRICE în interiorul elementelor BAR.
2. Se verifică marca de bere să fie vândută la "Joe's Bar" și numele mărcii de bere să corespundă cu **theBeer** din elementul PRICE.
3. Se construiește elementul de ieșire.

Interogarea

```
let $bars = doc("bars.xml") / BARS
for $beer in $bars / BEER
for $bar in $bars / BAR
for $price in $bar / PRICE
where $beer / @soldAt = "JoesBar" and
    $price / @theBeer = $beer / @name
return <BBP bar = {$bar / @name} beer
    = {$beer / @name}>{$price}</BBP>
```

Adevărat dacă
"JoesBar" apare
oriunde în secvență



Comparații Stricte

- Pentru a pretinde că lucrurile comparate sunt secvențe doar a unui singur element, se folosesc operatori comparație Fortran:
 - eq, ne, lt, le, gt, ge.
- **Exemplu:** \$beer/@soldAt eq "JoesBar" este adevărat doar dacă "Joe's" este singurul bar ce vinde berea respectivă.

Comparația Elementelor și a Valorilor

□ Atunci când un element este comparat cu o valoare primitivă, elementul este tratat ca valoarea sa, dacă acea valoare este atomică.

□ **Exemplu:**

```
/BARS/BAR[@name="JoesBar"] /
```

```
PRICE[@theBeer="Bud"] eq "2.50"
```

este adevărat dacă Joe vinde Bud cu 2.50 (\$).

Comparația a două Elemente

□ Nu este suficient ca două elemente să fie asemănătoare.

□ Exemplu:

```
/BARS/BAR[@name="JoesBar"] /  
PRICE[@theBeer="Bud"] eq  
/BARS/BAR[@name="SuesBar"] /  
PRICE[@theBeer="Bud"]
```

este fals, chiar dacă Joe și Sue au același preț pentru Bud.

Comparația a două Elemente

- Pentru ca elementele să fie egale, ele trebuie să fie identice, fizic, în documentul respectiv.
- **Subtilitate**: elementele sunt în realitate pointeri la secțiuni de documente particulare, nu șirurile de text ce apar în secțiune.

Obținerea Datelor din Elemente

- Presupunem că se dorește să se compare valorile elementelor, în loc de locația lor în documente.
- Pentru a extrage doar valoarea (de exemplu, prețul în sine) dintr-un element E , se folosește $\text{data}(E)$.

Exemplu: data()

- Presupunem că se dorește modificarea a ceea ce returnează “găsiți prețurile mărcilor de bere la baruri ce vând o marcă de bere vândută de asemenea de Joe” pentru a obține un element BBP “empty” (vid, fără valoare) cu prețul ca atribut.

```
return <BBP bar = {$bar/@name} beer  
= {$beer/@name} price =  
{data($price)} />
```

Eliminarea Duplicatelor

- Se folosește funcția `distinct-values` aplicată unei secvențe.
- **Subtilitate**: această funcție elimină tag-urile din jurul elementelor și compară valorile șir de caractere.
 - Dar nu reface tag-urile în rezultat.

Exemplu: Toate Prețurile Distincte

```
return distinct-values (  
  let $bars = doc("bars.xml")  
  return $bars/BARS/BAR/PRICE  
)
```

De reamintit: XQuery este
un limbaj expresie.
O interogare poate apărea
în orice loc unde poate
apărea o valoare.

Valori Booleene Efective

□ *Valoarea booleană efectivă* (EBV) a unei expresii este:

1. Valoarea existentă dacă expresia este de tip boolean.
2. FALSE dacă expresia evaluează la 0, "" [șirul vid], sau () [secvența vidă].
3. TRUE în celelalte cazuri.

Exemple EBV

1. @name="JoesBar" are EBV TRUE sau FALSE, depinzând de faptul că atributul name este "JoesBar".
2. /BARS/BAR[@name="GoldenRail"] are EBV TRUE dacă un anumit bar este denumit Golden Rail, și FALSE dacă nu există un astfel de bar.

Operatori Booleeni

- E_1 and E_2 , E_1 or E_2 , $\text{not}(E)$, se aplică oricăror expresii.
- Mai întâi se iau EBV-urile expresiilor.
- **Exemplu:** $\text{not}(3 \text{ eq } 5 \text{ or } 0)$ are valoarea TRUE.
- De asemenea: $\text{true}()$ și $\text{false}()$ sunt funcții ce returnează valorile TRUE și FALSE.

Expresii Ramificație

- if (E_1) then E_2 else E_3 este evaluată:
 - Se calculează EBV-ul lui E_1 .
 - Dacă este adevărat, atunci rezultatul este E_2 ; altfel rezultatul este E_3 .
- **Exemplu:** subelementele PRICE ale \$bar, spun că este barul lui Joe.

```
if ($bar/@name eq "JoesBar")  
then $bar/PRICE else ()
```

Secvența vidă.

Expresii de Cuantificare

some x in E_1 satisfies E_2


1. Se evaluează secvența E_1 .
 2. Fie x (orice variabilă) fiecare articol din secvență, și se evaluează E_2 .
 3. Se returnează TRUE dacă E_2 are EBV-ul TRUE pentru cel puțin un x .
- În mod analog:

every x in E_1 satisfies E_2

Exemplu: Some

- Barurile ce vând cel puțin o marcă de bere mai ieftin de 2 (\$).

```
for $bar in
    doc("bars.xml")/BARS/BAR
where some $p in $bar/PRICE
    satisfies $p < 2.00
return $bar/@name
```



Notă: where \$bar/PRICE < 2.00
ar fi funcționat de asemenea.

Exemplu: Every

- Barurile ce nu vând nici o marcă de bere mai scumpă de 5 (\$).

```
for $bar in
    doc("bars.xml")/BARS/BAR
where every $p in $bar/PRICE
    satisfies $p <= 5.00
return $bar/@name
```

Ordinea în Document

- Comparație după ordinea în document: << și >>.
- **Exemplu:** \$d/BARS/BEER[@name="Bud"] << \$d/BARS/BEER[@name="Miller"] este adevărat dacă elementul "Bud" apare înainte de elementul "Miller" în documentul \$d.

Operatori pe Mulțimi

- **union, intersect, except** operează pe secvențe de noduri.
 - Semnificațiile sunt asemănătoare cu SQL.
 - Rezultatul elimină duplicatele.
 - Rezultatul apare în ordinea din document.

XSLT

- XSLT (*extensible stylesheet language – transforms*) este un alt limbaj de procesare a documentelor XML.
- La origine era intenționat ca un limbaj pentru prezentare: să transforme XML într-o pagină HTML care să fie afișată.
- Dar poate de asemenea să transforme XML -> XML, astfel servind ca un limbaj de interogare.

Programe XSLT

- Ca și Schema XML, un program XSLT este el însuși un document XML.
- XSLT are un namespace special de tag-uri, de obicei indicat prin **xsl:**.

Template-uri

□ Elementul **xsl:template** descrie un set de elemente (ale documentului procesat) și ce trebuie făcut cu ele.

□ Formatul: `<xsl:template match = cale>`
... `</xsl:template>`

Atributul `match` specifică o expresie XPath ce descrie cum să fie găsite nodurile cărora li se aplică template-ul.

Exemplu:

Documentul BARS -> Tabel

- Ca un exemplu, se convertește documentul **bars.xml** într-un document HTML ce arată ca și relația **Sells(bar, beer, price)**.
- Primul template se va potrivi cu rădăcina documentului și va obține tabelul fără nici un rând.

Template pentru Rădăcină

```
<xsl:template match = "/">  
  <TABLE><TR>  
    <TH>bar</th><TH>beer</th>  
    <TH>price</th></tr>  
  </table>  
</xsl:template>
```

Template-ul se potrivește doar cu rădăcina.

Ieșirea template-ului este un tabel cu attributele capului de tabel, fără alte rânduri.

Trebuie consolidat.
După cum arată, nu se pot introduce rânduri.

Schițarea Strategiei

1. În tabelul HTML există `xsl:apply-templates` pentru a extrage datele din document.
2. Pentru fiecare BAR, se folosește `xsl:variable` *b* pentru a memora numele barului.
3. `xsl:for-each` pentru subelementul PRICE generează un rând; se folosește *b* și `xsl:value-of` pentru a extrage denumirea și prețul mărcii de bere.

Folosirea Recursivă a Template-urilor

- Un document XSLT în mod obișnuit conține mai multe template-uri.
- Se începe cu găsirea primului care se aplică rădăcinii.
- Orice template poate avea `<xsl:apply-templates/>`, ce cauzează "match" al template-ului să se aplice recursiv începând cu nodul curent.

Apply-Templates

- Atributul **select** precizează o expresie XPath ce descrie subelementele cărora li se aplică template-urile.
- **Exemplu:** `<xsl:apply-templates select = "BARS/BAR" />` precizează să se urmeze toate căile etichetate BARS, BAR începând cu nodul curent și să le aplice toate template-urile.

Exemplu: Apply-Templates

```
<xsl:template match = "/">
  <TABLE><TR>
    <TH>bar</th><TH>beer</th>
    <TH>price</th></tr>
  <xsl:apply-templates select =
    "BARS"  />
</table>
</xsl:template>
```

Extragerea Valorilor

□ `<xsl:value-of select = expresie XPath/>`
obține o valoare care să fie afișată.

□ **Exemplu:** presupunem că se aplică un template elementului BAR și se dorește afișarea denumirii barului într-un tabel.

```
<xsl:value-of select = "@name" />
```

Variabile

- Se declară **x** ca fiind o variabilă în felul următor:

```
<xsl:variable name = "x" />
```

- Exemplu:

```
<xsl:variable name = "bar">
```

```
  <xsl:value-of select = "@name" />
```

```
</xsl:variable>
```

Într-un template ce se aplică elementelor BAR
variabila **bar** este setată la denumirea barului.

Folosirea Variabilelor

- Se folosește \$ în fața numelui variabilei.
- Exemplu: `<TD>$bar</td>`

Completarea Tabelului

1. Se va aplica template-ul fiecărui element BAR.
2. Template-ul va atribui unei variabile **b** valoarea barului, și va itera pentru fiecare fiu PRICE.
3. Pentru fiecare fiu PRICE, se va afișa un rând, folosind **b**, atributul **theBeer**, și PRICE.

Iterația

□ `<xsl:for-each select = expresie Xpath>`

...

`</xsl:for-each>`

la fiecare fiu al nodului curent la care se ajunge cu calea `expresie Xpath` se execută corpul *for-each*.

Variabila
pentru
fiecare bar

Template-ul pentru BARS

Construiește un rând
bar-beer-price.

```
<xsl:template match = "BAR">
```

```
<xsl:variable name = "b">  
  <xsl:value-of select = "@name" />  
</xsl:variable>
```

```
<xsl:for-each select = "PRICE">
```

```
<TR><TD>$b</td><TD>  
  <xsl:value-of select = "@theBeer" />  
</td><TD>  
  <xsl:value-of select = "data(.)" />  
</td></tr>
```

```
</xsl:for-each>
```

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

Iterează pentru toate
subelementele PRICE
ale barului.

Elementul
curent

Exemplu practic

Fișierul catalog.xml

```
<?xml version="1.0"?>
<Catalog>

  <Book>
    <Title>Teach Yourself XML in 24 Hours</Title>
    <Author>
      <Name>Charles Ashbacher</Name>
      <Address>119 Northwood Drive</Address>
      <City>Hiawatha</City> 10: <State>Iowa</State>
      <PostalCode>52233</PostalCode>
      <Country>United States</Country>
      <Phone>(319) 378-4646</Phone>
      <Email>ashbacher@ashbacher.com</Email>
      <URL>http://www.ashbacher.com</URL>
    </Author>
    <Publisher>
      <Name>Sams Publishing</Name>
      <Address>201 West 103rd Street</Address>
      <City>Indianapolis</City>
      <State>Indiana</State>
      <PostalCode>46290</PostalCode>
      <Country>United States</Country>
      <Phone>(111) 123-4567</Phone>
      <Editor>Dummy Name</Editor>
      <EmailContact>Dummy. Name@sampublishing.com</EmailContact>
      <URL>http://www.sampublishing.com</URL>
    </Publisher>
    <ISBN>0-672-31950-0</ISBN>
    <PublisherBookID>1234567</PublisherBookID>
    <PublicationDate>2000-04-01</PublicationDate>
    <Price>24.95</Price>
  </Book>
```

Fișierul catalog.xml (continuare)

```
<Book>
<Title>Teach Yourself Cool Stuff in 24 Hours</Title>
<Author>
  <Name>ReallySmart Person</Name>
  <Address>110 Main Street</Address>
  <City>Silicon City</City>
  <State>California</State>
  <PostalCode>10101</PostalCode>
  <Country>United States</Country>
  <Phone>(101) 101-0101</Phone>
  <Email>RSmart@MyCompany.com</Email>
  <URL>http://www.MyCompany.com</URL>
</Author>
<Publisher>
  <Name>Sams Publishing</Name>
  <Address>201 West 103rd Street</Address>
  <City>Indianapolis</City>
  <State>Indiana</State>
  <PostalCode>46290</PostalCode>
  <Country>United States</Country>
  <Phone>(111) 123-4567</Phone>
  <Editor>Dummy Name</Editor>
  <EmailContact>Dummy.Name@sampublishing.com</EmailContact>
  <URL>http://www.sampublishing.com</URL>
</Publisher>
<ISBN>0-672-1010-0</ISBN>
<PublisherBookID>1234568</PublisherBookID>
<PublicationDate>2000-06-01</PublicationDate>
<Price>24.95</Price>
</Book>
```

Fișierul catalog.xml (continuare)

```
<Book>
<Title>Teach Yourself Uses of Silicon in 24 Hours</Title>
<Author>
<Name>Missy Fixit</Name>
<Address>119 Hard Drive</Address>
<City>Diskette City</City>
<State>California</State>
<PostalCode>10101</PostalCode>
<Country>United States</Country>
<Phone>(586) 212-3638</Phone>
<Email>MFixit@Googol.com</Email>
<URL>http://www.Googol.com</URL>
</Author>
<Publisher>
<Name>Sams Publishing</Name>
<Address>201 West 103rd Street</Address>
<City>Indianapolis</City>
<State>Indiana</State>
<PostalCode>46290</PostalCode>
<Country>United States</Country>
<Phone>(111) 123-4567</Phone>
<Editor>Dummy Name</Editor>
<EmailContact>Dummy.Name@samspublishing.com</EmailContact>
<URL>http://www.samspublishing.com</URL>
</Publisher>
<ISBN>0-578-31950-0</ISBN>
<PublisherBookID>1234569</PublisherBookID>
<PublicationDate>2000-01-01</PublicationDate>
<Price>26.95</Price>
</Book>
</Catalog>
```

Fișierul catalog.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="uri:xsl">
  <xsl:template match="/">
    <html>
      <body>
        <table border="1">
          <tr style="font-weight:bold;color:blue">
            <td>Author Name</td>
            <td>Author Address</td>
            <td>Author City</td>
            <td>Author State</td>
            <td>Author e-mail</td>
            <td>Publisher Name</td>
            <td>Publisher Address</td>
            <td>Publisher Phone</td>
          </tr>
```


Fișierul catalog.xsl (continuare)

```
<xsl:for-each select="Catalog/Book">
  <tr>
    <xsl:apply-templates/>
  </tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
<xsl:template match="Author">
  <td><xsl:value-of select="Name"/></td>
  <td><xsl:value-of select="Address"/></td>
  <td><xsl:value-of select="City"/></td>
  <td><xsl:value-of select="State"/></td>
  <td><xsl:value-of select="Email"/></td>
</xsl:template>
<xsl:template match="Publisher">
  <td><xsl:value-of select="Name"/></td>
  <td><xsl:value-of select="Address"/></td>
  <td><xsl:value-of select="Phone"/></td>
</xsl:template>
</xsl:stylesheet>
```

Fișierul catalog.html

```
<html>
<head>
<title>This program demonstrates the use of an XSL template to read XML
  data</title>
<script language="JavaScript">
function StartUp()
{
  var DataSource=new ActiveXObject("microsoft.xmlDOM");
  DataSource.load("catalog.xml");
  var XslStyle=new ActiveXObject("microsoft.xmlDOM");
  XslStyle.load("catalog.xsl");
  document.all.item("xslcontainer").innerHTML=
    DataSource.transformNode(XslStyle.documentElement);
}
</script>
</head>
<body onLoad="StartUp()">
<span ID="xslcontainer"></span>
</body>
</html>
```

Rezultatul

Author Name	Author Address	Author City	Author State	Author e-mail	Publisher Name	Publisher Address	Publisher Phone
Charles Ashbacher	119 Northwood Drive	Hiawatha	Iowa	ashbacher@ashbacher.com	Sams Publishing	201 West 103rd Street	(111) 123-4567
ReallySmart Person	110 Main Street	Silicon City	California	RSmart@MyCompany.com	Sams Publishing	201 West 103rd Street	(111) 123-4567
Missy First	119 Hard Drive	Diskette City	California	MFirst@Googol.com	Sams Publishing	201 West 103rd Street	(111) 123-4567

Fișierul catalog.html apelat în browser conduce la afișarea tabelară a conținutului fișierului catalog.xml folosind template-ul catalog.xsl.

În acest exemplu practic se folosește ActiveXObject XMLDOM pentru prelucrarea documentelor XML cu javascript.