1. XSLT - Scurta istorie

XSL este o prescurtare a eXtensible Stylesheet Language. World Wide Web Consortium (W3C [1]) a inceput standardizarea XSL deoarece s-a dorit crearea unui limbaj de descriere a unor fisiere de stil bazat pe XML.

Daca pentru HTML, stylesheet-urile se realizau folosind CSS (Cascading Stylesheets), aceste documente insotind documentele HTML si cuprindeau descrierea elementelor de stil folosite in afisarea paginilor HTML, fisierele XML nu beneficiau de acest ajutor. Limbajul HTML permitea de asemenea folosirea tag-urilor de descriere a stilului in cadrul documentului HTML. Dar in acest limbaj, tag-urile erau fixe si fiecare browser stia cum sa afiseze fiecare tag HTML. In XML, tag-urile nu mai sunt predefinite, pot fi folosite orice nume de tag-uri, iar browserele nu mai stiu cum sa afiseze fiecare tag. Acesta este rolul XSL, sa ofere browser-elor informatii despre cum sa afiseze un document XML, sa descrie modul in care acest document trebuie afisat.

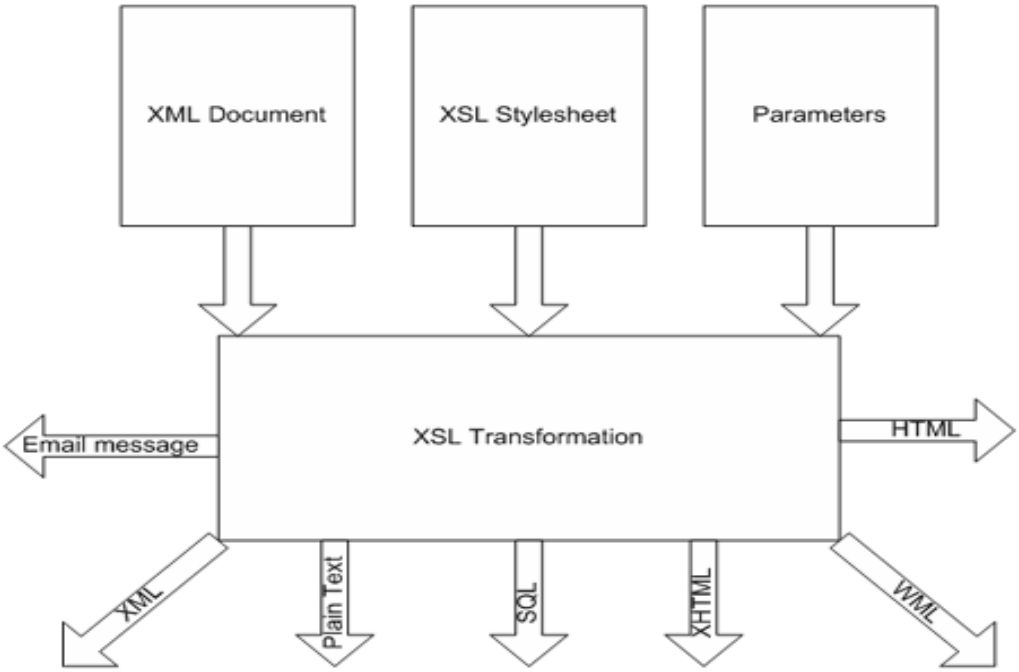
XSL este reprezentat de fapt prin 3 limbaje:

- XSLT(XSL Transforms) – un limbaj de transformare a documentelor XML
- XPath – un limbaj de descriere a structurii documentelor XML (de navigare in aceste documente)
- XSL-FO (XSL Formating Objects, numit astazi XSL) – un limbaj de formatare a documentelor XML

XSLT a devenit o recomandare W3C pe 16 noiembrie 1999. Aceasta a fost prima varianta a limbajului (1.0). Pe 24 august 2001 a aparut cel mai recent draft pentru XSLT 1.1, iar pe 3 noiembrie 2005 a aparut un draft pentru XSLT 2.0 (care este acum candidate recomandation).

2. Scop si functionare

In primul rand trebuie sa intelegem care este scopul si ce anume putem face cu XSLT. Figura de mai jos ilustreaza foarte bine scopul XSLT-ului:



Dupa cum se poate vedea, motorul XSL primeste ca intrare un fisier XML (poate fi bineinteles si un html), un fisier cu stilul de formatare (XSL Stylesheet) si eventuali parametrii pentru rulare. Rezultatele pot fi foarte diferite, de la „texte simple”, la „texte cu un anumit inteles” (xml, sql,etc)

Exista trei modalitati (dintre care doua mai importante) in care un document XML poate fi transformat intr-un alt tip de document prin aplicarea unui stylesheet XSLT:

1. Documentul XML si stylesheet-ul asociat sunt transmise aplicatiei client (browser-ului) caruia ii revine sarcina de a realiza efectiv transformarea in conformitate cu informatia din stylesheet-ul XSLT. In aceste conditii incarcarea serverului scade, dar browser-ul trebuie sa permita procesarea documentelor XML;
2. Aplicarea stylesheet-ului XSLT se face chiar pe server, documentul rezultat (uzual in format HTML) fiind transmis clientului. Se pot realiza astfel procesari in functie de natura clientului;
3. Cea de a treia posibilitate este extrem de putin utilizata si se refera la transformarea documentului XML cu ajutorul unei aplicatii externe si plasarea pe server a documentului rezultat (HTML), urmand ca acesta sa fie transmis clientului.

Elementul central al tehnologiei XSLT este template-ul: <xsl:template>. In cadrul acestuia se regasesc doua elemente importante:
- atributul match – specifica o cale in arborele de intrare;
- continutul – implementeaza modul in care se realizeaza transformarea.

Forma generala a unui template este:

```
<xsl:template match="element_XPath">
        ...
<xsl:template>
```

Asocierea unui document XML cu un stylesheet XSLT se realizeaza in cadrul documentului XML cu ajutorul instructiunii de procesare <?xml-stylesheet>:

```
<?xml-stylesheet href=”stylesheet/Login” type=”text/xsl” />
```

Argumentul href specifica numele stylesheet-ului XSLT si daca este cazul si calea ca trebuie acesta.

Dupa cum ati putut vedea intr-un laborator anterior pentru a sistematiza modul in care poate fi accesat un nod dintr-un document XML consortiul W3C a elaborat specificatia pentru limbajul XPath. Trebuie mentionat ca in cadrul XPath exista notiunea de radacina (root) a documentului (document root). Astfel, se face o distinctie foarte clara intre radacina documentului si elementul radacina al documentului XML. Radacina documentului, prin prisma XPath, este de fapt radacina arborelui de elemente reale - definite in cadrul documentului. Reamintim ca radacina (root-ul) documentului este utilizat la nivel conceptual, neavand un element corespondent in cadrul elementelor documentului XML, si este reprezentat prin caracterul “/”.

XSLT foloseste XPath pentru a defini si identifica parti din documentul sursa care se potrivesc cu vreunul dintre template-urile definite in documentul XSLT. Daca e gasita o potrivire, XSLT va transforma partea din documentul sursa care corespunde potrivirii in altceva, care va face parte din documentul rezultat. Partile pentru care nu exista nici o potrivire (cu nici un template), vor aparea nemodificate in documentul rezultat.

3. Sintaxa si folosire

Fisierul sursa .xml si fisierul de stil .xsl se leaga in felul urmator:

- in fisierul xml trebuie adaugate urmatoarele linii:

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

- fisierul xsl va avea urmatoarea structura :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        ........
    </xsl:template>
</xsl:stylesheet>
```

In locul tagului “xsl:stylesheet” se poate folosi si:
<xsl:transform version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

</xsl:transform>

Cel mai important element intr-un document XSLT este elementul **<template>**, care reprezinta o regula folosita in pattern matching. Acest element are urmatoarea forma generala:

<xsl:template match=”expresie XPath”> continutul dupa transformare

</xsl:template>

De exemplu:

<xsl:template match="/">
...
</xsl:template>.

Aceasta regula specifica ca la intalnirea elementului “/” (care este o expresie Xpath ce reprezinta radacina documentului XML), se inlocuieste acest element cu continutul elementului <xsl:template>

Un exemplu de folosire a acestui element este:

fisier xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="class.xsl"?>
<class>
<student>Jack</student>
<student>Harry</student>
<student>Rebecca</student>
<teacher>Mr. Bean</teacher>
```

```
</class>
```

fisier xsl:

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <xsl:template match="teacher">
        <p><u><xsl:value-of select="." /></u></p>
    </xsl:template>

    <xsl:template match="student">
        <p><b><xsl:value-of select="." /></b></p>
    </xsl:template>
    <xsl:template match="/">
        <html>
        <body>
        <xsl:apply-templates/>
        </body>
        </html>
    </xsl:template>

</xsl:stylesheet>
```

In exemplul de mai sus puteti observat cum se definesc 3 template-uri fiecare aplicandu-se unor anumite noduri (teacher, student sau radacina).
Elementul "*apply-templates*" poate avea si atributul select, caz in care template-ul se aplica doar pentru fiii care se potrivesc pe valoarea atributului. Aceasta forma se poate folosi pentru a specifica ordinea in care sunt procesate elementele fii:

```
<xsl:template match="student">
        <p>
        <xsl:apply-templates select="firstName"/>
        <xsl:apply-templates select="lastName"/>
        <xsl:apply-templates select="job"/>
        </p>
</xsl:template>
```

(exemplul de mai sus presupune existenta unor elemente fii – firstName, lastName, job – ai elementului „student")

Un alt element important pe care il puteti observa in exemplul de mai sus este elementul **<xsl:value-of>**. Acest element permite extragerea valorii unui element din documentul XML sursa si adaugarea acestei valori in documentul destinatie. Sintaxa generala a acestui element este:
<xsl:value-of select="expresie XPath"/>,
un exemplu fiind:
<xsl:value-of select="teacher"/>.

Expresia precedenta extrage valoarea elementului "teacher" din documentul XML sursa si o adauga in documentul (HTML) destinatie.

Un alt element folosit in documentele XSLT este elementul <xsl:for-each>. Acesta permite trecerea prin fiecare element dintr-un set de noduri intors de o expresie XPath. Sintaxa

generala a acestui element este:
<xsl:for-each select=”expresie XPath”>.

Expresia XPath ar trebui sa intoarca un set de noduri. Un exemplu de folosire a acestui element este:

<xsl:for-each select="class/student">,

care selecteaza pe rand toate nodurile de tip “student” din documentul XML sursa. Se pot pune si conditii in atributul select.

Daca „student” ar mai avea un element „name” care de fapt sa retina numele studentului respectiv :

```
<student>
        <name>Steve</name>
</student>
```

am putea face o „filtrare” astfel:

<xsl:for-each select="class/student[name='Steve']">.

Mai exista elementul ***<xsl:sort>*** . Acesta nu contine alte elemente,are un atribut: "select" si e folosit pentru sortarea iesirii (rezultatului) dupa valoarea elementului specificat in atributul "select". Exemplu:

xml:

```
<source>
        <name>John</name>
        <name>Josua</name>
        <name>Charles</name>
        <name>Alice</name>
        <name>Martha</name>
        <name>George</name>
</source>
```

xsl:

```
<xsl:stylesheet version = '1.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:template match="/">
   <TABLE>
      <xsl:for-each select="//name">
         <xsl:sort order="ascending" select="."/>
         <TR>
            <TH>
               <xsl:value-of select="."/>
            </TH>
         </TR>
      </xsl:for-each>
   </TABLE>
</xsl:template>
</xsl:stylesheet>
```

output:

```
<TABLE>
  <TR>
    <TH>Alice</TH>
  </TR>
```

```
  <TR>
    <TH>Charles</TH>
  </TR>
  <TR>
    <TH>George</TH>
  </TR>
  <TR>
    <TH>John</TH>
  </TR>
  <TR>
    <TH>Josua</TH>
  </TR>
  <TR>
    <TH>Martha</TH>
  </TR>
</TABLE>
```

Un alt exemplu folosind atribute:

xml:

```
<source>

            <car id="11"/>
            <car id="6"/>
            <car id="105"/>
            <car id="28"/>
            <car id="9"/>

</source>
```

xsl:

```
<xsl:stylesheet version = '1.0'  xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:template match="/">
    <TABLE>
        <xsl:for-each select="//car">
            <xsl:sort data-type="text" select="@id"/>
            <TR>
                <TH>
                    <xsl:text>Car-</xsl:text>
                    <xsl:value-of select="@id"/>
                </TH>
            </TR>
        </xsl:for-each>
    </TABLE>
</xsl:template>
</xsl:stylesheet>
```

output:

```
<TABLE>
  <TR>
    <TH>Car-105</TH>
  </TR>
```

```
  <TR>
    <TH>Car-11</TH>
  </TR>
  <TR>
    <TH>Car-28</TH>
  </TR>
  <TR>
    <TH>Car-6</TH>
  </TR>
  <TR>
    <TH>Car-9</TH>
  </TR>
</TABLE>
```

Un alt element este ***<xsl:if>*** ale carui actiuni (elementele din interiorul sau) au loc doar la indeplinirea conditiei specificata cu atributul "test"

```
<table border="1">
    <xsl:for-each select="grupa/student">
        <xsl:if test="varsta!='2 ani'">
          <tr>
            <td><xsl:value-of select="nume"/></td>
            <td><xsl:value-of select="varsta"/></td>
          </tr>
        </xsl:if>
    </xsl:for-each>
</table>,
```

Exemplu de mai sus asigura afisarea doar a studentilor care au varsta diferita de '2 ani'.Ceea ce poate apare in atributul "test" este similar cu ceea ce apare ca informatie de filtrare in atributul "select" din <xsl:for-each> intre "[" si "]";

Elementul <xsl:choose> este similar cu "switch" din C sau Java si se utilizeaza alaturi de <xsl:when> ("case" in C) si <xsl:otherwise> ("default" in C);

```
<table border="1">
    <xsl:for-each select="grupa/student">
        <tr>
          <td><xsl:value-of select="nume"/></td>
          <td>
              <xsl:choose>
                <xsl:when test!='2 ani'>
                <B><xsl:value-of select="varsta"/></B>
                </xsl:when>
                <xsl:otherwise>
                <I><xsl:value-of select="varsta"/></I>
                </xsl:otherwise>
              </xsl:choose>
          </td>
        </tr>
    </xsl:for-each>

</table>
```

Exemplu de mai sus are ca rezultat afisarea varstei studentilor fie Bold, fie Italics.

Mai multe functii si detalii puteti gasi pe site-urile www.w3.org si www.w3schools.com si in anexa A de la sfarsitul documentului.

Principala utilizare a XSLT este transformarea documentelor XML in alte documente de diferite tipuri. In procesul de transformare, XSLT foloseste Xpath pentru a defini parti ale documentului sursa care se potrivesc pe unul sau mai multe template-uri. Cand este gasita o potrivire, procesorul XSLT transforma partea potrivita din documentul sursa in partea corespunzatoare din documentul destinatie. De fapt, XSLT transforma arborele documentului sursa in arborele documentului destinatie.

In general se transforma documente XML in documente (X)HTML care pot fi afisate de browsere. Dar XSLT poate fi folosit si la transformarea documentelor XML in alte tipuri de documente. De fapt, principala calitate a XSLT este puterea acestui limbaj, toate posibilitatile de transformare oferite de el. Spre exemplu se pot obtine documente RTF (format Word) sau documente PDF pe baza unui document XML. Pentru obtinerea documentelor PDF se foloseste XSLT impreuna XSL-FO. O alta utilizare la care ar putea fi folosit XSLT este deschiderea, editarea si salvarea unui document XML de pe server. Avand un document XML pe server, se poate scrie un document XSLT, care sa genereze pe baza acestui document XML un form HTML in care campurile sunt editabile. Folosind un script ASP se pot prelua valorile introduse in form si salva in fisierul XML de pe server.

XSL este o componenta importanta a tehnologiei XML, iar cunoasterea sa reprezinta un element esential in dezvoltarea aplicatiilor bazate pe XML. Transformarea documentelor XML se realizeaza cu ajutorul stylesheet-urilor XSLT care permit procesari complexe oferind dezvoltatorilor o multitudine de elemente si functii in acest sens. Prezentarea intr-o viziune originala a conceptelor si elementelor cu adevarat importante a avut in vedere crearea unei imagini cat mai clare referitoare la tehnologia XSL, tehnologie care are o importanta deosebita in procesul de proiectare si implementare a aplicatiilor complexe, bazate pe tehnologia XSP, care utilizeaza biblioteci de tag-uri pentru implementarea logicii aplicatiei.

Bibliografie

- http://serghei.net/docs/programming/XmlBible/pdf
- http://w3.org
- http://w3schools.com
- http://www.zvon.org/xxl/XSLTutorial/Books/Output/contents.html#id22
- http://www.learn-xslt-tutorial.com/
- http://www.heise.de/ix/artikel/E/2001/01/167/
- XMLSpy – http://www.xmlspy.com/

ANEXA A

Alte elemente si functii XSLT

<xsl:include>

Este element de nivel 1, adica e fiu al lui "xsl:stylesheet" sau "xsl:transform". Include continutul unui stylesheet in altul.Styleshetul inclus si cel in care se include au aceeasi precedenta

exemplu: <xsl:include href="URI"/>

<xsl:import>

Este element de nivel 1, adica e fiu al lui "xsl:stylesheet" sau "xsl:transform".Importa continutul unui stylesheet in altul.Styleshetul inclus are precedenta mai mica decat cel in care se include

exemplu: <xsl:import href="URI"/>

<xsl:apply-imports>

Aplica un template dintr-un stylesheet importat .Sintaxa sa este similara cu cea din apply-template.

<xsl:number>

E folosit pentru a afisa un numar intr-un anumit format, de exemplu poate afisa pozitia nodului curent in document.

Sintaxa :

```
<xsl:number count="expression" level="single|multiple|any"
  from="expression" value="expression" format="formatstring"
  lang="languagecode" letter-value="alphabetic|traditional"
  grouping-separator="character" grouping-size="number"/>
```

Are ca atribute:

- count: [optional]: expresie XPath ce identifica nodurile ce vor fi numarate
- from: [optiona]: expre XPath: specifica de unde incepe numararea
- value: [optional]: un numar definit de user, in locul numarului generat prin numarare
- format: "1" = "1 2 3 .."; "01" = "01 02 03 .."; "a" = "a b c .." "I" = "I II III IV ..."; specifica cum se va face numararea
- groupping-separator: caracter de separare grupuri de digitzi
- grouping-size: dimensiunea grupurilor de digiti

Exemplu 1

```
<xsl:number value="250000" grouping-separator="."/>
```

Output:

250.000

Exemplu 2

```
<xsl:number value="250000" grouping-size="2"/>
```

Output:

25,00,00

Exemplu 3

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <html>
 <body>
 <p>
 <xsl:for-each select="grupa/student">
  <xsl:number value="position()" format="1" />
  <xsl:value-of select="nume" /><br />
 </xsl:for-each>
 </p>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

<xsl:message>

Acest element scrie un mesaj la output ;e folosit ptr tratare cazuri eroare.Are ca atribut "terminate='yes|no'" pentru cazul in care se doreste oprirea procesarii dupa mesaj.

Eemplu:

```
  <xsl:if test="nume="">
    <xsl:message terminate="yes">
      Studentul nu campul nume vid!
    </xsl:message>
  </xsl:if>
```

<xsl:text>

Folosit pentru a scrie text la output.

<xsl:call-template>

Apeleaza un template cu nume.

Ex:

```
 <xsl:template name="templatename" match="student">
 </xsl:template>
 <xsl:call-template name="templatename">
   <!-- Content:xsl:with-param* -->
 </xsl:call-template>
```

In "xsl:call-template" pot aparea "xsl:with-param" folosit pentru a transmite parametrii in templateul apelat.

<xsl:copy>

Creaza o copie a nodului curent, nodurile fii si atributele nu sunt copiate automat.
Exemplu:

```
<xsl:template match="message">
  <xsl:copy>
    <xsl:apply-templates/>
  </xsl:copy>
</xsl:template>
```

Exemplu copiaza la output nodul message.

<xsl:copy-of>

Creaza o copie a nodului curent, nodurile fii si atributele sunt copiate automat.
Atribut: select="expression" = specifica ce sa fie copiat

Exemplu:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:variable name="header">
 <tr>
 <th>Nume</th>
 <th>Prenume</th>
 </tr>
</xsl:variable>
<xsl:template match="/">
 <html>
 <body>
 <table>
  <xsl:copy-of select="$header" />
  <xsl:for-each select="grupa/student">
  <tr>
    <td><xsl:value-of select="nume"/></td>
    <td><xsl:value-of select="prenume"/></td>
  </tr>
  </xsl:for-each>
 </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

Exemplu de mai sus afiseaza intr-un tabel dupa nume si prenume.Se defineste o variabila "header" care se apeleaza cu sintaxa "$header".

<xsl:variable>

Defineste o variabila, locala sau globala (dc e pe nivelu 1 e globala).
Valoarea acestui element este data de continutul elementului "variable", fie prin valoarea atributului "select".Ca atribute amintim "name" si "select".Dupa ce a fost definita valoarea unei variabile, nu mai poafi modificata.O variabila fara "select" si

care nu contine nimic in ea este echivalenta cu un sir gol "".O variabila se acceseaza folosind $nume_varb

Exemplu <xsl:variable name="sir_vid"/>
<xsl:variable name="color" select="'red'" />
<xsl:variable name="color" select="'red'" />

<xsl:param>

Declara un parametru local sau global iar tributele si declararea sunt similare cu <xsl:variable>.

<xsl:with-param>

Defineste valoarea unui parametru care sa fie transmis unui template.Atributul nume trebuie sa aiba valoare identica cu numele unui element <xsl:param>.Aceste element poate fi folosit intr-un <xsl:apply-template> si <xsl:call-template> ;atributele si declararea sunt similare cu <xsl:variable>.

Valoarea parametrului e data fie de continutul elementului <xsl:with-param> fie de valoarea atributului "select".

<xsl:comment>

Introduce un comentariu.

<xsl:element>

Creaza un nod element pentru a fi trimis catre output.

Syntax

```
<xsl:element name="name" namespace="URI" use-attribute-sets="namelist">
  <!-- Content:template -->
</xsl:element>
```

<xsl:attribute>

Adauga un atribut unui element iar valoarea atributului va fi data de continutul lui <xsl:attribute >.

Exemplu:

```
<picture>
  <xsl:attribute name="source">
    <xsl:value-of select="images/name" />
  </xsl:attribute>
</picture>
```

<xsl:attribute-set>

Defineste o multime de atribute, sub un singur nume

Exemplu:

```
<xsl:attribute-set name="font">
  <xsl:attribute name="fname">Arial</xsl:attribute>
  <xsl:attribute name="size">14px</xsl:attribute>
```

```
  <xsl:attribute name="color">red</xsl:attribute>
</xsl:attribute-set>
```

<xsl:fall-back>

Specifica codul xslt care va fi rulat in cazul in care procesorul XSL nu suporta un anumit element XSL.

Exemplu:

```
 <xsl:loop select="title">
   <xsl:fallback>
     <xsl:for-each select="title">
       <xsl:value-of select="."/>
     <xsl:/for-each>
   </xsl:fallback>
 </xsl:loop>
```

Daca nu se poate executa <xsl:loop> (care oricum nu exista) atunci se executa codul din interiorul <xsl:fallback>

<xsl:key>

Exemplu de mai sus declara o cheie cu nume care poate fi folosita cu functia "key()". E folosit in special pentru cautari de elemente

Syntax: <xsl:key name="name" match="pattern" use="expression"/>

Atribute: match = indica nodurile asupra carora cheia va fi aplicata

use = valoarea cheii ptr fiecare dintre noduri

Exemplu:

Presupunem documentul xml "persons.xml":

```
<persons>
  <person name="Tarzan" id="050676"/>
  <person name="Donald" id="070754"/>
  <person name="Dolly" id="231256"/>
</persons>
```

Definim in XSL cheia "preg" care are ca valoare IDul "person"ului

```
<xsl:key name="preg" match="person" use="@id"/>
```

Pentru a gasi "person"ul cu id="050676":

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:key name="preg" match="person" use="@id"/>
<xsl:template match="/">
  <html>
  <body>
  <xsl:for-each select="key('preg','050676')">
    <p>
    Id: <xsl:value-of select="@id"/><br />
    Name: <xsl:value-of select="@name"/>
    </p>
```

```
  </xsl:for-each>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

5.1.Functii:

current()

Returneaza nodul current.

document(uri_fisier,optional_select_noduri)

Acceseaza noduri intr-un document extern.
Ex:
<xsl:value-of select="document('celsius.xml')/celsius/result[@value=$value]"/>

element-available(nume_element)

Testeaza daca un element este suportat de procesorul XSLT. Se refera doar la elementele ce pot apare intr-un <xsl:template>.

format-number(nr, format, optional_decimal_format)

Converteste numarul intr-un sir conform cu formatul.

function-available(nume)

Verifica daca functia data ca param e suportata de procesorul XSLT

generate-id()

Genereaza un id unic.

key(nume_element_key, sir_de_cautat)

Returneaza o multime de noduri din document, folosind indexul dat de elementul <xsl:key> identificat de primul parametru.

system-property(string)

Returneaza valoarea unei proprietati a sistemului.Proprietatile disponibile sunt : 'xsl:version', 'xsl:vendor', 'xsl:vendor-uri'.

unparsed-entity-uri(nume_entity)

Returneaza URIul unei entitati externe.

Exemplu:

in DTD:

<!ENTITY pic SYSTEM "http://www.w3schools.com/picture.jpg" NDATA JPEG>

in XSL:

unparsed-entity-uri('pic')

intoarce http://www.w3schools.com/picture.jpg