V. gyakorlat: XML alapú adatkezelés

Írta: Mátéfi Gergely
Nagypál Gábor, Bihari István, Hajnács Zoltán korábbi segédletének felhasználásával

1. Bevezetés

A nagyobb informatikai rendszerek jellemzően több hozzáférési felülettel rendelkeznek, például vastagklienssel, webes megjelenési felülettel, adatkapcsolati interfésszel külső informatikai rendszer felé, stb. Több hozzáférési felület mellett a hagyományos kliens-szerver architektúra nem hatékony, ehelyett az összetettebb rendszerek felépítése *többrétegű architektúrát* követ. Többrétegű architektúra esetén az adatelemekre vonatkozó előírások betartatásáért felelős *alkalmazáslogikai réteg* és a felhasználói felület kezeléséért felelős *megjelenítési réteg* szétválik, és a különböző hozzáférési felületek a közös alkalmazáslogikai réteget használják.

Az egyes rétegekben található, esetenként eltérő platformon futó, eltérő gyártótól származó szoftverkomponensek integrációjához az adatelemek leírására alkalmas közös nyelv szükséges. Az elmúlt években az Extensible Markup Language (XML) nyelv vált az adatelemek leírásának de facto szabványává. Az XML platformfüggetlen, szöveges formátumú jelölő nyelv, amely alkalmas információk és adatelemek struktúrált leírására és továbbítására. Kapcsolódó szabványai lehetőséget teremtenek XML sémák definiálására és validálására, valamint a különböző sémák közötti transzformációra is.

Jelen labor célja, hogy bepillantást engedjen az XML alapú adatbázis-alkalmazásfejlesztésbe. Az első alfejezet XML dokumentumok szerkezetét mutatja be, ezt követően megnézzük, hogyan transzformálható egy XML dokumentum az XSLT és XPath technológiák segítségével. A segédlet az XML 1.0, XSLT 1.0 és XPath 1.0 szabványokra épül.

2. XML dokumentumok felépítése

2.1. Elemek és Címkék

Az XML dokumentumok, a HTML-hez hasonlóan, szöveges formátumú fájlok, amelyek egymásba ágyazott elemeket tartalmaznak. Az elemeket nyitó- és zárócímkék (*tagek*) jelölik, mint ahogy az alábbi példa is mutatja:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<uzenet>
  <felado>Géza</felado>
  <cimzett>Béla</cimzett>
  <torzs>Szia Világ!</torzs>
</uzenet>
```

A HTML-lel szemben az XML szabvány nem rögzíti a címkeszótárat. Ehelyett az alkalmazástól függő mindenkori nyelvtan (XML séma) határozza meg, milyen címkék, milyen egymásba ágyazási szabályokkal szerepelhetnek a dokumentumban. Az XML terminológiája szerint egy dokumentum *jól formált* (well-formed), ha szintaktikája betartja az XML előírásait, és *érvényes* (valid) egy sémára nézve, ha követi annak nyelvtanát.
A címkék neve betűvel vagy aláhúzással („_”) kezdődhet, nem lehet benne szóköz és nem kezdődhet „xml”-el. Az XML a címkék nevében különbséget tesz kis- és nagybetűk között, tehát például a `<Felado>` és a `<felado>` címkék különbözőnek számítanak.

2.2. Szerkezet

Minden XML dokumentum egy vezérlési utasítással kezdődik, amely kötelezően tartalmazza az XML verziószámát, és opcionálisan a felhasznált kódlapot:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
```

Az XML dokumentumok elemei fa struktúrát alkotnak. Egy dokumentumnak pontosan egy gyökér eleme kell, hogy legyen; ez a fenti példában az `<üzenet>` elem volt. Minden nyitócímkéhez kell, hogy tartozzon egy záró címke is, amelyek közrefogják az elemhez tartozó adatot. Ellentétben a HTML (csak a gyakorlatban!) megengedőbb szabályaival, XML-ben a nyitó- és zárócímkéknek követniük kell a zárójelezési szabályokat. Így például az alábbi forma működhet HTML-ben, de bizonyosan szabálytalan XML-ben:

```
<b>Ez vastag <i> és ez még dőlt is </b> ez már csak dőlt </i>
```

Helyette a szabályos XML leírás:

```
<b>Ez vastag <i> és ez még dőlt is </i></b><i> ez már csak dőlt </i>
```

Ha egy elemhez nem tartozik adat, akkor a nyitó- és a zárócímkék összevonhatóak egy önlezáró címkébe (empty-element tag), például a `<br></br>` helyett egyszerűen írható `<br/>`.
A megjegyzéseket, a HTML-hez hasonlóan, a `<!-- Megjegyzés -->` szintaktikával jelezhetjük egy XML dokumentumban. A megjegyzések nem ágyazhatók egymásba.

2.3. Attribútumok

Az XML logikája szerint egy elem tulajdonsága vagy gyermekelemmel, vagy attribútummal írható le, a használt séma szabályainak megfelelően. Az attribútumokat a nyitócímkébe lehet elhelyezni, például:

```
<üzenet kelt='20050221'>Hello!</üzenet>
```

Az attribútum értéke megadható mind aposztróf, mind idézőjel határolók között, azonban – ellentétben a HTML-lel – a határolót nem szabad elhagyni. Egy attribútum egy címkében csak egyszer szerepelhet.

2.4. Helyettesítő szekvenciák

A címkékkel jelölt adatokban bármilyen karakter szerepelhet, kivéve a foglalt `<` és `&` vezérlőkaraktereket. A vezérlőkarakterek helyett az XML helyettesítő szekvenciák használhatóak, lásd az alábbi táblázatban. Például ez szabálytalan megadás:

```
<formula>a < b & b < c => a < c </formula>
```

Helyette ez használandó:

```
<formula>a &lt; b &amp; b &lt; c => a &lt; c</formula>
```

Alternatívájaként a speciális `<![CDATA[...]]>` határoló is alkalmazható:

```
<formula><![CDATA[a < b & b < c => a < c]]></formula>
```

Eredeti karakter	&	<	>	”	’	újsor
Helyettesítés	`&`	`<`	`>`	`"`	`'`	`
`

2.5. Névterek

Mivel a címkeszótárat az alkalmazások határozzák meg, ezért különböző alkalmazásoktól származó dokumentumok összefésülésekor névütközések jöhetnek létre a címkék nevében. A névütközések elkerülésére *névterek* használhatóak. Egy névtéret az alkalmazás által definiált *névtér URI* (Uniform Resource Identifier, egységes erőforrás azonosító) azonosít, például a később ismertetendő XSLT a `http://www.w3.org/1999/XSL/Transform` névteret használja. A névtérbe tartozó címkék használatához egy egyedi prefixet kell definiálni a névtér számára az `xmlns:prefix="névtér URI"` speciális attribútummal. Ezt követően a névtér címkéire *kvalifikált elnevezéssel*, a `<prefix:címkenév>` formában lehet hivatkozni, ahogy az alábbi példa is szemlélteti:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<mail:message from="Béla" to="Réka" xmlns:mail="internet:mail">
  <mail:subject>Találka</mail:subject>
  <mail:body xmlns:xhtml="http://www.w3.org/1999/xhtml">
    <xhtml:body>
      Találkozzunk <xhtml:b>6-kor</xhtml:b> a szokott helyen!
    </xhtml:body>
  </mail:body>
</mail:message>
```

3. XML dokumentumok létrehozása

Az XML dokumentumok előállításának lehetséges technológiáival a jelen keretek között részletesen nem foglalkozunk, csupán megemlítjük, hogy természetesen lehetősége van pl. SQL utasítások segítségével is kialakítani a tartalmát. Jelen körülmények között feltételezzük, hogy az input XML dokumentum már rendelkezésünkre áll.

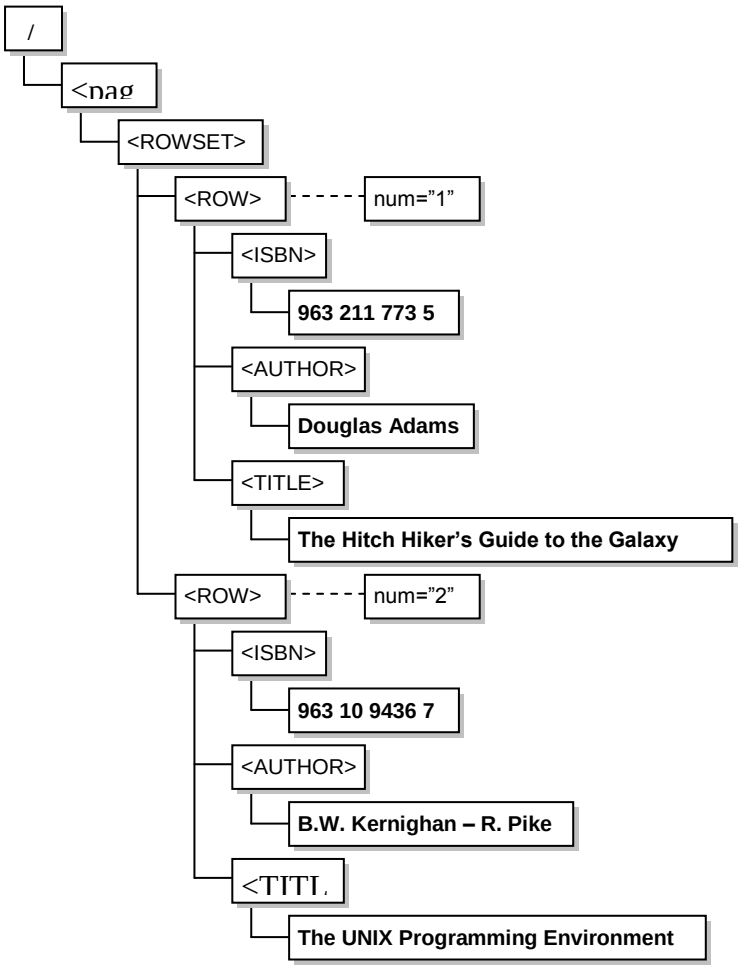
4. Az XSL transzformáció

Az *XSL Transzformáció* (XSLT) bemenő XML dokumentumok fa struktúrájú reprezentációját transzformálja át kimeneti fa struktúrába. Az XSL transzformáció szabályait úgynevezett *XSL stíluslapok* (XSL stylesheets) határozzák meg.[1] *XSLT processzornak* nevezzük azt a szoftvert, amely végrehajtja az XSL transzformációt.
Vegyük a következő, egyszerű könyvtári példát, melyben két sorban egy-egy könyvet kívánunk jellemezni:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<page>
<ROWSET>
  <ROW num="1">
    <ISBN>963 211 773 5</ISBN>
    <AUTHOR>Douglas Adams</AUTHOR>
    <TITLE>The Hitch Hiker's Guide to the Galaxy</TITLE>
  </ROW>
  <ROW num="2">
    <ISBN>963 10 9436 7</ISBN>
    <AUTHOR>B.W. Kernighan-R. Pike</AUTHOR>
    <TITLE>The UNIX Programming Environment</TITLE>
  </ROW>
</ROWSET>
</page>
```

Ennek kanonikus fa-struktúrájú reprezentációját az alábbi ábra szemlélteti. Ezen az XML dokumentum csomópontokból álló faként jelenik meg.

[1] Vigyázat: az XSL stíluslapok nem azonosak a HTML szabványból ismeretes CSS (Cascading Style Sheets) stíluslapokkal!



A következő csomópont típusokat különböztetjük meg: *gyökér csomópont* (root node), *elem csomópont* (element node), *szöveges csomópont* (text node), *attribútum csomópont* (attribute node).[2] A dokumentum minden esetben gyökér csomóponttal kezdődik, amely magát a dokumentumot reprezentálja. A gyökér csomópontnak egyetlen elem csomópont gyermeke lehet, a példán ez a `<page>` csomópont. Elem csomópontoknak további attribútum, szöveges és elem csomópont gyermekei is lehetnek, tetszőleges kombinációban.

Megjegyzendő, hogy az ábra nem teljesen egyezik a korábbi szöveges reprezentációval, amelyet az átláthatóság kedvéért sortörésekkel és tabulátorokkal tagoltunk. A tagolás teljesen szabályos az XML szintaktikában, azonban a tagoló karakterek a fa struktúrában is megjelennek szöveges csomópontokként.

Az XSL stíluslap maga is egy XML dokumentum, amely az XSLT névtérbe tartozó utasításokat használ a transzformáció leírásához. A stíluslapok felépítését az alábbi példa demonstrálja, amelyben a könyvek kanonikus formátumban átadott listáját transzformáljuk HTML táblázatos alakra:

[2] A teljesség kedvéért megemlítjük, hogy létezik még *névtér csomópont* (namespace node), *feldolgozási utasítás csomópont* (processing instruction node) és *megjegyzés csomópont* is (comment node).

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html"/>
<xsl:template match="/">
<html>
  <body>
    <table>
      <xsl:for-each select="page/ROWSET/ROW">
        <tr>
          <td><xsl:value-of select="ISBN"/></td>
          <td><xsl:value-of select="AUTHOR"/></td>
          <td><xsl:value-of select="TITLE"/></td>
        </tr>
      </xsl:for-each>
    </table>
  </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Az XSL stíluslap gyökéreleme az `<xsl:stylesheet>`. Egy stíluslap egy vagy több *sablont* (template) tartalmazhat. Sablon definiálható a forrás XML dokumentum bármely csomópontjához, ekkor az adott csomópontra illeszkedő sablon határozza meg az adott részfa transzformációját. A sablonokról később még bővebben lesz szó, itt az egyszerűség kedvéért csak egyetlen sablont definiáltunk az `<xsl:template>` paranccsal, amely közvetlenül a gyökérelemre illeszkedik. A példa sablon vegyesen tartalmaz HTML címkéket (`<body>`, `<table>`, `<tr>`, `<td>`), valamint XSLT parancsokat, mint `<xsl:for-each>` és `<xsl:value-of>`. A XSL feldolgozás során az XSLT névtéren kívüli címkék (itt a HTML címkék) változatlan formában kerülnek a kimenetre, ugyanakkor az XSLT parancsok értelmeződnek és végrehajtódnak.

A mintában szereplő `xsl:for-each` XSLT utasítás iterációra szolgál: a végigmegy a `select` attribútumában kiválasztott csomópontokon, és a beágyazott utasításokat minden egyes részfára végrehajtja. A példában a „page/ROWSET/ROW” kifejezés az XML forrásban szereplő ROW elemeket választja ki.

Az `xsl:value-of` XSLT utasítás egy szöveges csomópontot hoz létre a kimeneten. A szöveges csomópont tartalmát a `select` attribútumában megadott kifejezés határozza meg. A példában szereplő „ISBN”, „AUTHOR”, „TITLE” kifejezések a feldolgozás során éppen aktuális részfák értékét, azaz a keresett mezők értékét adják vissza.

A transzformáció befejeztével az eredményfát az XSLT processzor szöveges formában írja a kimenetre. Ezt a folyamatot nevezzük az eredményfa *szerializációjának* (serialization). Az XSL stíluslapban szereplő `xsl:output` parancs a szérializációs folyamatot szabályozza. Az XSLT 1.0 háromféle kiírási metódust támogat:

- `<xsl:output method="xml"/>`: a csomópontok jól-formázott XML formátumban kerülnek kiírásra
- `<xsl:output method="html"/>`: a csomópontok HTML 4.0-kompatibilis formátumban kerülnek kiírásra, így például az önlezáró `<br/>` címke helyett `<br>` jelenik meg a kimeneten.
- `<xsl:output method="text"/>`: csak a csomópontok értéke kerül kiírásra, jelölés nélkül

A példánkban a html metódust választottuk, így a transzformáció kimenete az alábbi lesz:

```
<html>
  <body>
    <table>
      <tr>
        <td>963 211 773 5</td>
        <td>Douglas Adams</td>
        <td>The Hitch Hiker's Guide to the Galaxy</td>
      </tr>
      <tr>
        <td>963 10 9436 7</td>
        <td>B.W. Kernighan-R. Pike</td>
        <td>The UNIX Programming Environment</td>
      </tr>
    </table>
  </body>
</html>
```

Ha stíluslap csak egyetlen sablont tartalmaz, amely közvetlenül a gyökérelemre illeszkedik, akkor lehetőség van egy **egyszerűsített írásmód** használatára is. Ekkor az `<xsl:stylesheet>` és az `<xsl:template>` elemek elmaradnak, és a sablonon belüli gyökérelem válik a stíluslap gyökerévé, az XSL névtér deklaráció pedig az új gyökérbe kerül. Így a példa stíluslap egyszerűsített írásmóddal:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<html xsl:version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <body>
    ...
  </body>
</html>
```

Az alapértelmezett kimeneti metódus az általános írásmódnál „xml”, míg az egyszerűsített írásmódnál, ha a gyökérelem `<html>`, a „html”.

5. XPath kifejezések

Az *XPath* XML dokumentumok fastruktúrájú reprezentációjában csomópontok megcímzésére szolgáló szabványos nyelv. Egy XPath kifejezés eredménye lehet csomópontok (részfák) halmaza, numerikus, szöveges vagy logikai érték.

Az előző XSL példában több ponton is használtunk már XPath kifejezéseket: az `xsl:template` parancsban a „/” kifejezés a fa gyökerét, az `xsl:for-each` parancsban a „page/ROWSET/ROW” kifejezés a sorokat reprezentáló részfákat, az `xsl:value-of` parancsban pedig az „ISBN” stb. kifejezések az aktuális részfa megfelelő mezőit reprezentáló csomópontokat választották ki.

Az XPath *elérési út* (location path) hasonló egy DOS-os vagy UNIX-os könyvtárváltó parancs útvonalkifejezéséhez: ez is egy útvonalat ad meg, csak ezúttal az XML fastruktúrában. A fájlrendszerekhez hasonlóan itt is kétféle elérési út létezik: *abszolút*, ahol az elérési út „/” karakterrel kezdődik, ekkor a kiindulási pont a dokumentum gyökere, valamit *relatív*, ahol a kiindulási pont az aktuális csomópont. Az elérési út *lépésekből* (location steps) áll, amelyeket „/” határoló választ el egymástól. Egy lépés általános formája:[3]

```
csomóponttípus[feltételes kifejezés]
```

A feltételes kifejezés rész opcionális, elmaradhat. A lehetséges csomóponttípusok a következők:

[3] Itt csak a rövidített (abbreviated) elérési út formátumot ismertetjük. A rövidítetlen (unabbreviated) formátum használatakor a lépés kiegészül az irány jelölővel.

Típus	Kiválasztott csomópont(ok)
`név`	Adott nevű gyermekcsomópontok
`@név`	Adott nevű attribútum-csomópontot
`.`	Aktuális csomópont
`..`	Szülő csomópont
`*`	Elem típusú gyermekcsomópontok
`@*`	Attribútum típusú gyermekcsomópontok
`//`	Csomópont és annak összes leszármazója
`node()`	Elem típusú gyermekcsomópontok
`text()`	Szöveges típusú gyermekcsomópontok
`comment()`	Magyarázat típusú gyermekcsomópontok

A lépésben szereplő feltételes kifejezés függvényeket és értékvizsgálatot tartalmazó összetett XPath logikai kifejezés, amelyekkel a lépésben kiválasztott csomópontok köre tovább szűkíthető. Néhány tipikus kifejezés:

Kifejezés	Igaz, ha…
`elem`	létezik az `elem` elnevezésű csomópont
`elem=érték`	az `elem` nevű csomópont értéke `érték`
`@attr=érték`	az `attr` nevű attribútum értéke `érték`
`position()=n`	ez a csomópont a halmaz `n`. eleme
`count(halmaz)=n`	az XPath elérési úttal definiált csomóponthalmaz elemeinek száma `n`
`sum(halmaz)=n`	az XPath elérési úttal meghatározott csomóponthalmaz értékeinek összege `n`

A feltételes kifejezések között az `and` és `or` logikai műveletek használhatók. A `[position()=n]` alak helyett használható az egyszerűsített `[n]` forma is. Az XPath kifejezésekben használható fontosabb függvényeket a Függelék tartalmazza.

Két XPath kifejezés által kijelölt csomóponthalmazok uniója a „|” operátorral képezhető.

Néhány XPath példa a könnyebb érthetőség kedvéért, amelyeket a korábbi kanonikus XML kimenetre értelmezünk:

- `/` : a dokumentum gyökere (nem azonos a gyökér címkével!)
- `/page/ROWSET/ROW` : az összes lekérdezés összes sorát reprezentáló részfák
- `/page/ROWSET[1]/ROW[1]/*` : az első lekérdezés első sorának mezői
- `/page/ROWSET/ROW[AUTHOR]` : azon sorok, melyekben az AUTHOR mező nem NULL értékű (azaz melyekben létezik az AUTHOR-t reprezentáló csomópont)
- `/page/ROWSET/ROW[@num > 1 and @num < 4]` : a lekérdezések második és harmadik sorait reprezentáló részfák (felhasználva a num attribútumot a kanonikus kimenetben)
- `/page/ROWSET/ROW[last()]` : a lekérdezések utolsó sorát reprezentáló részfa
- `//ROW[contains(TITLE,'Galaxy')]` : az összes olyan sor, amelynek a TITLE mezője tartalmazza az „Galaxy” stringet.
- `count(//AUTHOR)`: az AUTHOR nevű csomópontok száma
- `name(/page/ROWSET[1]/ROW[1]/*)`: az első lekérdezés gyerekcsomópontjainak neve
- `/page/ROWSET[1]/ROW | /page/ROWSET[2]/ROW`: az első és a második lekérdezés sorait reprezentáló csomópontok uniója

Emlékeztető: ha egy XPath kifejezést XSL stíluslapon használunk, akkor az XML szintaktikai előírásai miatt a foglalt karakterek (`<`, `&`) helyett a megfelelő helyettesítő szekvenciákat kell alkalmazni.

6. XSL stíluslapok hozzárendelése XML dokumentumokhoz

Egy XML dokumentumhoz tartozó XSL stíluslapot az `xml-stylesheet` feldolgozási utasítás segítségével lehet megadni a forrásdokumentumban. Az utasítás `type` attribútumának értéke kötelezően „`text/xsl`”, a stíluslap elérhetőségét pedig a `href` attribútumban kell megadni:

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

A választott XML formátumú megjelenítésben minden könyvhöz egy `<book>` címke tartozik, melynek attribútumaként jelenik meg a könyv ISBN azonosítója, és értékeként a könyv legkedvezőbb ára:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <pricelist>
    <xsl:for-each select="ROWSET/ROW">
      <book isbn="{ISBN}">
         <xsl:value-of select="PRICE"/>
       </book>
    </xsl:for-each>
  </pricelist>
</xsl:template>
</xsl:stylesheet>
```

A példából az is látható, hogy egy **attribútum értékét** a `<címke attribútum="{XPath kifejezés}">` formában lehet a stíluslapon beállítani.
A szöveges megjelenítésben minden könyvhöz tartozik egy sor, amelyben az ISBN-t és az árat tüntetjük fel:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text"/>
<xsl:template match="/">
  <xsl:for-each select="ROWSET/ROW">
    <xsl:value-of select="ISBN"/>
    <xsl:value-of select="PRICE"/>
    <xsl:text>&#10;</xsl:text>
  </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

A 7. sorban szereplő
 szekvencia az újsor karakternek felel meg. Mivel a stíluslapban szereplő szöveges csomópontok elejéről és végéről az XSL transzformáció levágja a whitespace karaktereket, ezért használjuk az `xsl:text` utasítást. Az `xsl:text` utasítással jelölt szöveges érték változtatás nélkül kerül a kimenetre.

7. Elágazások és változók XSL stíluslapokon

7.1. Feltételes végrehajtás

```
<xsl:if test="XPath kif.">
  XSLT részlet
</xsl:if>
```

Ha igaz a `test` attribútumban megadott XPath kifejezés, akkor végrehajtja a beágyazott XSLT részletet. Amennyiben az XPath kifejezés nem logikai értéket, hanem XML részfahalmazt ad eredményül, akkor az értéke pontosan akkor igaz, ha a visszaadott részfahalmaz nem üres. Így az `<xsl:if>` (és az alább ismertetett `<xsl:choose>`) utasítás alkalmas annak eldöntésére is, hogy egy adott csomópont létezik-e az XML dokumentumon belül.

7.2. Feltételes elágazás

```
<xsl:choose>
  <xsl:when test="XPath kif.">
    XSLT részlet
  </xsl:when>
  <xsl:when test="XPath kif.">
    XSLT részlet
  </xsl:when>
  ...
  <xsl:otherwise>
    XSLT részlet
  </xsl:otherwise>
</xsl:choose>
```

Az `xsl:choose` pontosan egy XSLT részletet hajt végre: azt, ahol legelőször igaz az XPath kifejezés. Ha egyik sem igaz, az `xsl:otherwise` ág kerül végrehajtásra.

7.3. Változók

Az XSL változó elnevezés csalóka, mivel az XSL változók konstansok, csak egyszer adhatunk nekik értéket. A változók érvényességi köre az az XML címke, ill. annak összes leszármazottja, ahol a változót definiáltuk. Változóhoz rendelhető skalár érték (szöveg, szám, logikai), vagy XML részfa, ahogy az alábbi két definíció mutatja:

```
<xsl:variable name="változó_név" select="XPath kif."/>
<xsl:variable name="változó_név">XML részfa</xsl:variable>
```

A definiált változók felhasználhatóak később XPath kifejezésekben, ahol a `$változó_név` formában lehet rájuk hivatkozni.[5] Például a

```
<xsl:variable name="n" select="2"/>
<xsl:value-of select="item[$n]"/>
```

parancsok a 2. `item` részfa értékét illesztik a kimenetbe.

[5] XSLT 1.0-ban az XML részfaként (result tree fragment) definiált változók szövegesen tárolódnak és csak sztringként kezelhetőek, így az XPath csomópont címzés kifejezések nem alkalmazhatóak rájuk. A kimenetre azonban részfaként is beilleszthetőek az *xsl:copy-of* függvény segítségével.

8. XSLT Sablonok

8.1. Sablonok rekurzív feldolgozása

Mint korábban említettük, egy XSLT sablon egy csomópont transzformációjának szabályait tartalmazza. A sablonokhoz tartozik egy XPath kifejezéssel megadott minta, amely meghatározza, mely csomópontokra érvényes az adott sablon. A mintát a `match` attribútumban kell megadni. Az előző példákban csak egyetlen sablont használtunk, amely közvetlenül a gyökér elemre illeszkedett, és amely a teljes fa transzformációs szabályát tartalmazta. Több sablon használatával a transzformáció rekurzív módon is végrehajtható.
Az általános XSL transzformáció menete a következő. Az XSL transzformáció elején a gyökér csomópont az egyetlen kiválasztott csomópont. Az XSLT processzor kikeresi a kiválasztott csomópontra illeszkedő sablont, és végrehajtja a sablonban definiált szabályokat. Ha a sablon olyan XSLT utasítást tartalmaz, amely további csomópontokat jelöl ki feldolgozásra (tipikusan a gyermek csomópontokat), akkor a feldolgozás az adott csomópontokat egyenként kiválasztva folytatódik.
Az XSLT processzor tartalmaz egy beépített sablont, amely biztosítja a rekurzív feldolgozást arra az esetre, ha egy elem csomópontra nincs illeszkedő explicit sablon a stíluslapon:

```
<xsl:template match="*|/">
  <xsl:apply-templates/>
</xsl:template>
```

Ez a sablon nem ad semmilyen kimenetet, csupán az `xsl:apply-templates` paranccsal arra utasítja a processzort, hogy a gyermekcsomópontokon folytassa a feldolgozást.
Létezik egy másik beépített sablon a szöveges és attribútum csomópontokra is. Ezen sablon egyszerűen kimásolja a csomópont értékét a kimenetre:

```
<xsl:template match="text()|@*">
  <xsl:value-of select="."/>
</xsl:template>
```

Ha egy csomópontra több sablon is illeszkedik, akkor az XSLT processzor mindig a legspecifikusabb sablont fogja kiválasztani. Ökölszabályként, a `*` mintánál specifikusabb a `VALAMI` típusú minta, és ennél is specifikusabb a `VALAMI/VALAMIMÁS`, illetőleg a `VALAMI[feltétel]` típusú minta.
Az alábbi példa azt szemlélteti, hogyan tudjuk a könyvek listáját megjeleníteni HTML táblázatban rekurzív sablonokkal.

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html"/>
<xsl:template match="/">
  <html>
    <body><xsl:apply-templates/></body>
  </html>
</xsl:template>

<xsl:template match="ROWSET">
  <table><xsl:apply-templates/></table>
</xsl:template>

<xsl:template match="ROW">
  <tr><xsl:apply-templates/></tr>
</xsl:template>

<xsl:template match="ROW/*">
  <td><xsl:apply-templates/></td>
```

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

8.2. Többször felhasználható nevesített sablonok

Az XSLT lehetőséget biztosít arra, hogy a több sablonban is használt XSLT részleteket külön blokkokba, úgynevezett *nevesített sablonokba* helyezzük. A normál sablonokkal szemben a nevesített sablonokat nem mintaillesztéssel választja ki a processzor, hanem más sablonokból az az explicit `<xsl:call-template name="sablonnév">` utasítással lehet meghívni őket. A nevesített sablonok nevét az `xsl:template name` attribútumában kell megadni.
Az alábbi példa egyúttal azt is szemlélteti, hogyan adható át paraméter a meghívott sablonnak:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template name="nevesitett_sablon">
  <xsl:param name="par1" select="2"/>
  <!-- XSLT utasítások -->
</xsl:template>

<xsl:template match="ROW">
  <xsl:call-template name="nevesitett_sablon">
    <xsl:with-param name="par1" select="42"/>
  </xsl:call-template>
</xsl:template>

</xsl:stylesheet>
```

8.3. Stíluslapok tagolása

A struktúráltság végett lehetőség van egy stíluslap több fájlra tagolására, illetőleg külső stíluslapokban tárolt sablonkönyvtárak felhasználására. Külső stíluslap beillesztésére az `xsl:include` és az `xsl:import` utasítások használhatóak, mindkét utasítás az `xsl:stylesheet` közvetlen gyermekelemeként adandó meg. A beillesztett sablonok precedenciája `xsl:include` esetén azonos, `xsl:import` esetén alacsonyabb lesz, mint a hívó stíluslap sablonjaié.

9. Felhasznált irodalom

1. T. Bray et al. (editors): Extensible Markup Language (XML) 1.0 (Third Edition), W3C Recommendation, 2004.
2. J. Clark (editor): XSL Transformations (XSLT) Version 1.0, W3C Recommendation.
3. J. Clark et al. (editors): XML Path Language (XPath) Version 1.0, W3C Recommendation.
4. Oracle 10g: XML Developer's Kit Programmer's Guide, Oracle Co., 2004.
5. S. Muench: Building Oracle XML Applications, Chapter 7: Transforming XML with XSLT.
6. Oracle® XML Developer's Kit Programmer's Guide – 11g Release 2 (11.2), elérhető online: http://docs.oracle.com/cd/E11882_01/appdev.112/e23582/adx_overview.htm

10. Függelék: XPath Függvény Referencia

Csomópont halmaz függvények	
`last()`	Az aktuális csomóponthalmaz elemeinek számát adja vissza
`position()`	Az aktuális elem indexét adja vissza.
`count(node-set)`	A paraméter csomóponthalmaz elemeinek számát adja vissza.
`namespace-uri(node-set)`	Az első csomópont címkéjének névterét adja vissza
`name(node-set)`	Az első csomópont címkéjének nevét adja vissza.
Sztringkezelő függvények	
`string(object)`	Az argumentumként megadott értéket szöveges értékké konvertálja.
`concat(string,string,...)`	A paraméterként adott sztringeket összefűzi
`starts-with(string,string)`	Igaz, ha a paraméterként megadott első sztring a másodikkal kezdődik.
`contains(string,string)`	Igaz, ha a paraméterként megadott első sztring tartalmazza a másodikat.
`substring-before(string,string)`	Visszaadja az első sztring mindazon részét, amely a második sztring első sztringben való első előfordulása előtt van.
`string-length(string?)`	A megadott sztring hosszát adja vissza. Ha az argumentum elmarad, az aktuális elem értékének hosszát adja vissza.
`normalize-space(string?)`	Levágja a megadott sztring elején és végén levő whitespace karaktereket, a sztring közepén levőket pedig egyetlen szóközre cseréli. Ha az argumentum elmarad, az aktuális elem értékére végzi a műveletet.
`translate(string,string,string)`	Az első sztringben előforduló, második sztringben megadott karaktereket a harmadik sztringben megadott karakterekre cseréli. Például translate(„PHP”,”PH”,”DT”) kimenete „DTD”.
Logikai függvények	
`boolean(object)`	Az argumentumként megadott értéket logikai értékké konvertálja.
`not(boolean)`	Logikai érték ellentettjét adja vissza.
`true()`	Logikai igaz
`false()`	Logikai hamis
Numerikus függvények	
`number(object)`	Az argumentumként megadott értéket numerikus értékké konvertálja.
`sum(node-set)`	A csomóponthalmaz értkeinek összegét adja
`floor(number)`	Az argumentumot lefelé kerekíti
`ceiling(number)`	Az argumentumot felfelé kerekíti
`round(number)`	Az argumentumot kerekíti

11. Függelék: XSLT Referencia

xsl:apply-templates	
Végrehajtja az XSL transzformációt a select attribútumában megadott csomópontokra, ennek hiányában a gyermekcsomópontokra.	
`select`	Csomópontokat meghatározó XPath kifejezés
`mode`	Ha definiált, akkor ezen módra definiált sablonokkal hajtja végre a

	transzformációt.
xsl:attribute	
Egy attribútum csomópontot hoz létre az eredményfában. Az attribútum értékét a beágyazott szöveg definiálja.	
`name`	Az attribútum neve
xsl:call-template	
Meghív egy nevesített sablont. Paramétereket az opcionális xsl:with-param gyermekelemekkel lehet átadni.	
`name`	A nevesített sablon neve.
xsl:choose	
Elágazást definiál, az ágakat az xsl:when és xsl:otherwise gyermekelemek tartalmazzák.	
xsl:copy-of	
A kiválasztott csomópontokat átmásolja az eredményfába. Működése hasonló a value-of parancséhoz, de megtartja az eredeti fastruktúrát, nem konvertál szöveges csomóponttá.	
select	A csomóponthalmazt meghatározó XPath kifejezés
xsl:for-each	
Iterál a megadott csomóponthalmazon. Az iteráció sorrendjét az opcionális xsl:sort gyermekelem definiálja.	
`select`	A csomóponthalmazt meghatározó XPath kifejezés
xsl:if	
Ha a megadott feltétel igaz, végrehajtja a beágyazott XSLT stíluslap részletet.	
`test`	Feltételt definiáló XPath kifejezés
xsl:import	
Egy külső stíluslapot illeszt az aktuális stíluslapba, így a külső stíluslap sablonjai is elérhetővé válnak. Eltérően az xsl:include-tól, az importált stíluslap sablonjainak alacsonyabb a prioritása, mint az importáló stíluslap sablonjainak. Az xsl:import csak az xsl:stylesheet közvetlen gyermekeként adható meg, és minden más gyermeket megelőzően kell megadni.	
`href`	A külső stíluslap elérhetősége.
xsl:include	
Egy külső stíluslapot illeszt az aktuális stíluslapba, így a külső stíluslap sablonjai is elérhetővé válnak. Az xsl:include csak az xsl:stylesheet közvetlen gyermekeként adható meg.	
`href`	A külső stíluslap elérhetősége.
xsl:otherwise	
Az xsl:choose parancs egy ágát definiálja. Ha egyetlen xsl:when feltétel sem igaz, akkor az xsl:otherwise elembe ágyazott stíluslap részlet kerül végrehajtásra.	
xsl:output	
Eredményfa szérializációját vezérlő utasítás.	
`method`	Szérializációs metódus, lehetséges értékek: xml, html, text
`encoding`	A szérializációhoz használandó preferált kódtáblát deklarálja.
`omit-xml-declaration`	Beállítja, hogy az XML deklarációs utasítás kerüljön-e a kimenetre. Lehetséges értékek: yes, no
`indent`	Beállítja, hogy az XSLT processzor whitespace-ek hozzáadásával formázza-e a kimenetet. Lehetséges értékek: yes, no
`media-type`	A karakterfolyam média típusát (mime type) definiálja.
xsl:param	
Egy XSL paramétert definiál. Az xsl:variable-től eltérően, a változónak csak az alapértelmezett értékét definiálja, amely a sablonnak átadott paraméterekkel felülírható.	
`name`	Paraméter neve
`select`	A változó értékét meghatározó XPath kifejezés.
xsl:sort	
Az xsl:apply-templates és xsl:for-each utasítások gyermekelemeként adható meg, és ezen utasításokban a csomópontok feldolgozási sorrendjét határozza meg.	
`select`	A rendezési kulcsot meghatározó XPath kifejezés.
`data-type`	A kulcs adattípusát definiálja. Ha értéke text, a kulcsot lexikografikusan rendezi. Ha értéke number, akkor a kulcsot számmá konvertálja és numerikusan rendezi.
`order`	Rendezés iránya, lehetséges értékek: ascending, descending

`case-order`	Szöveges rendezésnél határozza meg, hogy a nagy, vagy a kis betűk kerüljenek előre. Lehetséges értékei: upper-first, lower-first.
xsl:stylesheet	
Stíluslap gyökéreleme.	
`version`	Értéke kötelezően 1.0
xsl:template	
Sablont definiál.	
`match`	XPath kifejezés, amely azonosítja azon csomópontokat, amelyekre a sablon érvényes. Nevesített sablonnál elmaradhat.
`name`	Nevesített sablon neve. Nevesítetlen sablonnál elmaradhat.
`priority`	Sablon prioritása. Ha több, ugyanannyira specifikus sablon illeszkedik egy csomópontra, a prioritás dönt, melyik kerül végrehajtásra.
`mode`	Opcionálisan megadható mód.
xsl:text	
A megjelölt szöveget változtatás nélkül kimásolja a kimenetre.	
xsl:value-of	
Szöveges csomópontot állít elő az eredményfában.	
`select`	Az értéket meghatározó XPath kifejezés. Ha a kifejezés csomóponthalmazzal tér vissza, a hozzájuk tartozó értékek kerülnek a kimenetre.
xsl:variable	
Egy XSL változót definiál.	
`name`	Változó neve
`select`	A változó értékét meghatározó XPath kifejezés.
xsl:when	
Az xsl:choose parancs egy ágát definiálja. Ha a megadott kifejezés igaz, akkor végrehajtja a beágyazott stíluslap részletet.	
`test`	A feltételt definiáló XPath kifejezés.
xsl:with-param	
Ezen utasítás az xsl:apply-templates és az xsl:call-template parancsok gyermekelemeként adható meg, és a hívott sablonnak átadott paraméter(eke)t definiálja.	
`name`	Paraméter neve
`select`	Paraméter értékét meghatározó XPath kifejezés