A webalkalmazások új nemzedékének lelke: az XSLT

Cameron bemutatja az Extensible Stylesheet Language for Tranformations (XSLT) nyelvet, és elmondja, miért is olyan fontos téma ez manapság.

Az Extensible Stylesheet Language for Transformations (XSLT – bővíthető stíluslapnyelv átalakításokhoz) olyan számítási nyelv, amelyet kifejezetten két XML-dokumentum közötti átalakításra fejlesztettek ki. Az XSLT ismertetése nem kis feladat. A legfőbb nehézséget a változatosság okozza: az XSLT-nek számos alkalmazási köre van, rengeteg XSLT-motor létezik és egyetlen XSLT-alkalmazás több együttműködő módszert is alkalmaz. Ezért meglehetősen fontos, hogy csak a valóban lényeges részekre összpontosítsunk.

A mindenre használható XML

Az első lényeges tudnivaló az XSLT-vel kapcsolatban, hogy Extensible Markup Language- (XML) alapú (lásd még az ***XSLT-fogalmak gyűjteménye*** című széljegyzetet). Az XML olyan egyetemes adatformátum, amelyet úgy terveztek, hogy bármilyen típusú adatot képes legyen tárolni: eljárásadatokat, programokat és dokumentumokat – kezdve a vásárlási szabályoktól a bibliafordításokig, bármilyen emberi nyelven, bármilyen számítógépes rendszeren és operációs rendszeren. Az XML úgy néz ki, mint a HTML, csak éppen némileg összetettebb. Valójában az XML egyik tervezési célja éppen a HTML általánosítása volt oly módon, hogy eközben a HTML-szakértők kényelemérzete megmaradjon. Létezik egy XHTML-nek nevezett külön XML-változat is, amely közvetlenül HTML-ként értelmezhető. A Linuxvilág gyakorta jelentet meg olyan cikkeket, amelyek valamilyen szinten kapcsolatosak a XML nyelvvel.

Egy tökéletesen XML-esített világ sok szempontból egyszerűbb lenne. A tartozás-nyilvántartás működésének vizsgálatánál például nem kellene tudnunk, hogy ki kihez tartozik, ki ment el háromhetes vakációra, és a többi zavaros emberi részletet. Ha fel tudunk rajzolni egy diagrammot, amely a beérkezett számlákat és kimenő kifizetéseket ábrázolja – a folyamat során valószínűleg egyre szaporodó hitelesítő bejegyzésekkel megtűzdelve –, akkor sikerült a lényegi adatot levezetnünk. Mámorító látomás. Azt sugallja, hogy az a rendszer, amely az egyik XML-dokumentumot (például számla) képes egy vagy több másik XML-dokumentummá átalakítani (fizetési csekk, hitelesítő bejegyzés), az önműködően megszervezi, sőt akár meg is oldja az összes lényeges szervezési feladatot. Ez az, amiért manapság az XSLT olyannyira érdekes.

Azok az olvasók, akik rendelkeznek XML-világbeli tapasztalatokkal, projekttapasztalataikat általánosíthatják, és máris fogalmat alkothatnak az XSLT valódi értékéről. Bárki, aki a gyakorlatban is foglalkozott az XML-lel, tudja, hogy valójában a megoldás kezdetét jelenti, nem pedig valamiféle csodaszert, mint amilyennek a reklámszövegek beállítják. Az XSLT-vel pontosan ugyanez a helyzet: hasznos és elég hatékony módja a termelésben használt alkalmazások összeállításának megszervezésében. A XML-dokumentumok átalakításának alapötlete igen fontos dolog – hogy belássuk, valóban működő ötletről van szó, közelebbi pillantást kell vetnünk a technikai részletekre.

example1.xsl

```
<xsl:stylesheet
     ➥xmlns:xsl =
➥"http://www.w3.org/1999/XSL/Transform"
     ➥version = "1.0">
  <xsl:output method="html"/>

  <xsl:template match = "/">
     <html>
          <body>
             <xsl:apply-templates select =
             ➥"datum"/>
          </body>
     </html>
  </xsl:template>

  <xsl:template match = "datum">
     <h1>
          <xsl:value-of select = "."/>
     </h1>
  </xsl:template>
</xsl:stylesheet>
```

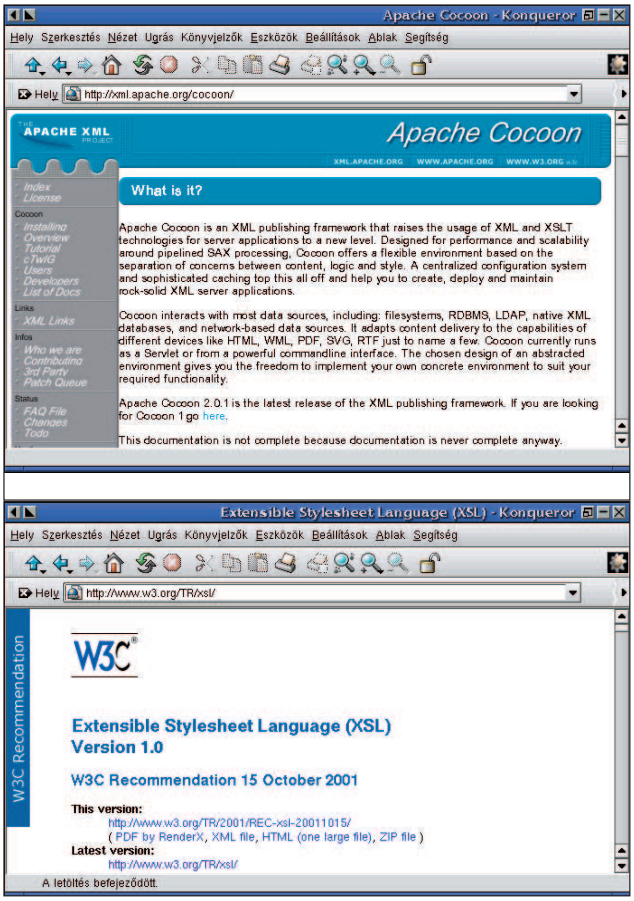
Saját motorunk

Ahhoz, hogy XSLT-utazásunkon elindulhassunk, és könnyebben ráérezhessünk a nyelv apró finomságaira, egy saját motorra, azaz egy nyelvfeldolgozóra (language processor) lesz szükségünk. A legszélesebb körben elterjedt eszköz Java-alapú, illetve üzleti. Ezeket gyakran nagyobb kiszolgálótermékekbe építve találjuk: adatbáziskiszolgálókban, alkalmazáskiszolgálókban és így tovább.

A példákat a fentiek helyett írásunkban a tDOM-motor meghatározásai alapján mutatjuk be. A tDOM több előnnyel is rendelkezik, amelyek között a legfontosabbak: a szabad, nyílt forrás felhasználási szerződés hatálya alá tartozik, kivételesen memóriatakarékos, és méréseink szerint kétszer olyan gyors, mint a versenytárs XSLT-motorok. Telepítése gyors és tömör, ezenkívül parancsfájlosítható (scriptable) parancsmóddal rendelkezik, amelyen keresztül kényelmesen utasíthatjuk tDOM-ot. Továbbá kitűnően illeszkedik az alább leírt kétszintű programozási stílushoz, ugyanakkor elég üzembiztos ahhoz, hogy több igényes honlap is alkalmazza a termelésben.

Saját tDOM-példányunk elkészítéséhez olvassuk el a ***Hogyan kezdjünk XSLT-t programozni*** széljegyzetet. Ez a cikk ugyanis a XSLT használatának első példájával zárul, amelyet a következőképpen hívhatunk meg:

```
tclsh8.3 xslt.tcl example1.xml example1.xsl
➥example1.html
```



Ez a parancssor a következőkre utasít: „használd a 8.3-as változatú Tcl-értelmezőt az ***xslt.tcl*** program indításához. Az ***xslt.tcl***-eszköz alkalmazza az ***example1.xsl*** stíluslapot az ***example1.xml*** dokumentumra, és kimenetként előállítja az ***example1.html***-t".
Figyeljük meg, ahogy a gép bemenetként átveszi az ***example1.xml***-t, majd előállítja a mindössze néhány soros ***example1.html***-t:

```
<?xml version="1.0"?>
<datum>elsı  zenet </datum>
```

Gondoljunk az ***example1.html***-re mint a fenti sor helyesen formált HTML-kiterjesztésére:

```
<html><body><h1>elsı  zenet </h1></body></html>
```

Az XML mint adat és kód

Ha mindössze az ***example.html***-hez hasonló egyszerű HTML-dokumentum előállítása a célunk, az XSLT elsajátítása helyett közvetlenül valamilyen egyszerű makrónyelvet is megírhatunk vagy használhatunk. Az XSLT igazi értéke akkor nyilvánul meg, amikor összetettebb feladatokat oldunk meg.
Az XSLT-átalakítást beállíthatjuk, hogy az ***example1.html*** kimenetet az adott stílusban készítse el, esetleg az adott betűkészlettel vagy szabványos honlaphivatkozásokat és -nyilatkozatokat alkalmazva.
Az XSLT ezen átalakítások végrehajtásához a stíluslapok nyelvét alkalmazza. Bár a stíluslapok már az XSLT bevezetése előtt is használatban voltak, ebben a cikkben minden más felhasználási területet figyelmen kívül hagyva, az XSLT-stíluslapokat következetesen stíluslapoknak fogjuk nevezni.
Bizonyos szempontból a stíluslap program. Ahogy az

```
int main()
{
    puts("Hello.");
}
```

egy C program forráskódja, a stíluslap az XSLT program forrása. A stíluslapok különlegessége, hogy maguk is XML-dokumentumok. A hagyományos számítógépprogram-látvány helyett (mondjuk ahogy a C, a Java és a `ksh` kinéz), az XSLT-forrás egyfajta kulcsszószöveg (markup text, lásd az *1. listát*). Az XML-re olyannyira jellemző bőbeszédűséggel ez nagyjából a következőkre utasít: „működj úgy, mint egy program, amely beolvassa a `<datum>`-elemeket, majd a tartalmukat egy helyesen megformált HTML-dokumentum `<h1>` fejlécei közé helyezi." Így készül az ***example1.html***.
Az XSLT-feldolgozást megvalósító alkalmazás maga egy Tcl

Hogyan kezdjünk XSLT-ben programozni

A tDOM a Tcl teljes telepítését igényli. A 8.3.4-es vagy újabb változat használatát javaslom. A legbiztonságosabb a forrásából telepíteni. A telepítés általában a következő lépésekből áll:

- A forrás telepítőkészlet letöltése, például a következő helyről: ➲ http://prdownloads.sourceforge.net/tcl/tcl8.3.4.tar.gz.
- A források kicsomagolása.
- Végül a hagyományos `autoconf`-alapú összeépítés lépései:

```
cd tcl8.3.4/unix
./configure
make
make install
```

A tDOM helyes telepítése nagyjából ugyanezt a sémát követi:

- Letöltés: ➲ http://phaseit.net/binaries/tDOM-0.63.tar.gz.
- Kicsomagolás.
- Előállítás:

```
cd tDOM-0.63/unix
./configure
make
make install
```

Az e cikkre vonatkozó források a ➲ http://phaseit.net/examples/xslt.zip címen találhatók. Töltsük le és telepítsük őket a munkakönyvtárunkba. Ettől kezdve a következő parancsot már meg tudjuk hívni:

```
tclsh8.3 xslt.tcl example1.xml example1.xsl
➥example.html
```

Figyeljük meg, hogy a fenti parancs kimenete az *example.html* lesz. Bár linuxos végrehajtható állományok is letölthetők, általában elmondható, hogy a legtöbb Unix Tcl programozó a forrásból szeret dolgozni. Ennek aztán az a következménye, hogy a forrásterjesztések általában egy kicsit kifinomultabbak. Különösen igaz ez a tDOM esetében.
A Windows esetében viszont pont fordítva áll a helyzet. A tDOM kitűnően működik Windows alatt is, megeshet azonban, hogy szükségünk is lesz rá, de ennél az operációs rendszernél a bináris terjesztések telepítése az általános. A legfrissebb adatokért keresd a ➲ http://mini.net/tcl/tDOM címet.

XSLT-fogalmak gyűjteménye

API – Application Programming Interface, azaz alkalmazásprogramozási felület.

Cocoon – Java- és XSLT-központú terjesztő keretrendszer az Apache-hoz.

CSS – Cascading Stylesheet, vagyis kaszkádolt összerendelt stíluslap. A jelkulcsok (markup elements), például hivatkozások összerendelése tulajdonságokkal, főleg megjelenítési tulajdonságokkal (mint például „rajzold kékkel"). A legtöbb böngészővel dolgozó ember találkozott már CSS-ekkel.

DOM – Document Object Model, azaz dokumentum objektummodell. A DOM egy dokumentumokhoz szánt programozási API, amely például az XML osztályrendszerű faszerkezetét emeli ki.

DTD – Document Type Definition, azaz dokumentum-típusmeghatározás. A DTD olyan metanyelv, amely az XML dokumentumszótárát határozza meg. Az XML Schemata általánosabb keretrendszerben szintén ugyanezt teszi.

FO – Objektumok formázása. A FO-k az XML és a megjelenítés közötti átalakítást írják le. Tervrajzadatokat tartalmaznak.

FOP – objektumokat pdf-be (és általánosabb esetben más megjelenítési formátumokba) formázó átalakító.

Funkcionális programozás – a Lisp, Haskell és Erlang nyelveket gyakran nevezik funkcionális programozási nyelvnek, mivel megváltoztathatatlan változókat alkalmaznak, mellékhatás nélküli műveletek, rekurzió és provability jellemzik őket. Az XSLT funkcionális és nem eljárásokból építkező programozási nyelv, ellenben a C, Java, vagy Visual Basic nyelvekkel.

HTML – HyperText Markup Language. A Világhálózat általános nyelve.

Névtér (namespace) – programnyelvi elv a többértelműség kizárására. A Baseball és Biology névterek megkülönböztetésével egyértelműsíthetjük, mikor gondolunk `Baseball::bat`-re és mikor a `Biology::bat`-re (angolul a „bat" szó denevért és ütőt is jelent). Az XML-fejlesztők számára ennek ott van jelentősége, hogy alkalmazásaik fejlesztése közben nem kell félniük attól, hogy változóneveik és más nevek véletlenül ütik egymást.

SAX – Simple API for XML, azaz egyszerű XML API. A DOM kiegészítésére szánt API, eseményközpontú, és az XML-t mint karakterek folyamát szemléli.

Schema – metanyelv-előírás. Az XML Schema például olyan szabályokat ad meg, amelyek az XML-dokumentum tartalmát szabályozzák.

Scripted document – olyan fájl, amely adatot és az adaton műveleteket végző kódot egyaránt tartalmaz.

Stíluslap (stylesheet) – a dokumentum értelmezésének vagy átalakításának leírása. Az XSLT-stíluslapok érdekessége, hogy maguk is XML-dokumentumok.

Vocabulary XML – az XML-szótár tulajdonképpen metanyelv, amelyben lehetőség nyílik adott témakörökhöz tartozó megvalósítások vagy szótárak leírására. Léteznek testreszabott XML-szótárak matematikához, autóeladáshoz, a Gnome GUI-felülethez és sok más dologhoz.

W3C – az XML-szabványokat sok más tevékenység mellett a World Wide Web Consortium adja ki.

Xalan – A Xalan egy Java-központú XSLT-motor, az Apache Project része.

XML– Extensible Markup Language. A HTML nyelvhez hasonló kinézetű nyelvre kell gondolni, amely azonban elvben bármilyen digitális adatot képes kódolni.

XPath – olyan nyelv, amely az XML-dokumentum egy részét azonosítja vagy címzi meg. Gondolhatunk rá lekérdező-nyelvként, az XSLT kiegészítéseként (az XSLT a változásokat írja le, az XPath azt mutatja meg, hol történjenek ezek a változások).

XSL – Extensible Stylesheet Language, vagyis bővíthető stíluslapnyelv.

XSLT – Extensible Stylesheet Language for Transformations, azaz átalakításokhoz szánt bővíthető stíluslapnyelv. Olyan nyelv, amelyet XML-források más XML-forrásokká alakításához fejlesztettek ki.

XSP – eXtensible Server Pages a Cocoon egyik képessége, melynek segítségével dinamikusan készíthetünk XML-alapú honlapokat.

program. Ezért itt az ideje, hogy pár alapvető dolgot a Tcl-ről is megtudjunk. A tDOM XSLT-motorját Tcl-csatolások valósítják meg, és a ***xslt.tcl*** parancsfájl a XML-dokumentumok fájlneveit egyszerűen parancssori változókként kéri be, majd továbbadja a motornak.
Vizsgáljuk meg újra a példahívásunkat!

```
tclsh8.3 xslt.tcl example1.xml example1.xsl
➥example1.html
```

A ***tclsh8.3*** az elindított futtatható program neve, és a ***xslt.tcl*** az a legkisebb Tcl-parancsfájl, amely a tDOM XSLT-motorját előhozza. Ha egy kicsit tovább szeretnénk fejleszteni az eszköz hibakezelő képességét, a legkézenfekvőbb kezdés a ***xslt.tcl*** újratervezése.
A ***xslt.tcl*** futtatása elindítja a XSLT-feldolgozót, amely három fájlnevet kap meg. Az ***example1.xml*** fájl a felhasznált példa XML-dokumentumforrás. A stíluslapot ezekkel a fájlnevekkel alkalmazzuk az ***example1.xml***-re. A folyamat az eredménydokumentumot az ***example1.html*** fájlba írja. Válasszunk más logikai tartalmat, azaz másik XML-forrást az ***example1.html***-hez. Legyen például az ***example2.xml***. A kimenet stílusának megváltoztatásához viszont az ***example1.xsl***-t kell újraírnunk.

Sikeresen lefuttatunk egy XSLT-programot. Már csak az maradt hátra, hogy megismerjük az XSLT nyelvet, illetve megtudjuk, hogyan is alkalmazhatjuk valós feladatok megoldásában. Mielőtt komolyabban belemerülnénk az XSLT szintaxisába és szemantikájába, vessünk egy rövid pillantást a felhasználási területekre!

Egy nyelv – sok alkalmazás

Képzeljük el, hogy egy több tízezer lapot tartalmazó webhelyért felelünk. Ezeket a lapokat szervezetfüggő XML-szótár szerint tároljuk, amely kiszedegeti a formázási adatokat és a HTML-szemetet – dokumentumaink kizárólag az adott laphoz tartozó logikai adatokat tartalmazzák. A látogatóknak természetesen HTML-re van szükségük, de ezeket dinamikusan állítjuk elő, szabványos fejléccel, keretekkel, vezérlőelemekkel, lábjegyzetekkel és minden egyéb a Weben elvárható díszítőelemmel. Az XSLT lehetővé teszi, hogy az összes lap stílusát egyszerre változtassuk meg. Továbbá szépen megosztja a munkát az XML-tartalmú fájlok és a XSLT-stíluslapok közt, így a különböző szakemberek hatékonyan tudnak együttműködni. Ez a döntéshozói szintű leírás eléggé sokat elrejt a megvalósítás változatosságából. Hol és mikor történik az XSLT-átalakítás? Lehet egy XML-dokumentumokból álló hátterünk, amit aztán időnként parancssoros XSLT-feldolgozó programmal hagyomá-

nyos webkiszolgálónak szánt statikus HTML-dokumentumokká alakítunk. Tarthatjuk az XML-forrásokat adatbázisban is, ahonnan vagy XML-ként kapjuk vissza és HTML-é alakítjuk át őket, vagy rögtön teljes értékű HTTP-oldalként kérjük le. Ezeket a felületeket különféle alkalmazáskiszolgálók, tartalomkezelők és XML-adatbázisok teszik elérhetővé. Még egy változat: csak a forrásokat tartjuk a kiszolgálón, majd HTML-kiterjesztések és böngésző megfelelő együttesével magát a böngészőt utasítjuk, hogy értelmezze a megkapott XSLT-t. Valamennyi lépést tetszés szerinti mértékben tehetjük dinamikussá, egészíthetjük ki gyorstárazással a sebességnövelés érdekében, böngésző vagy olvasó szerinti testreszabhatósággal és így tovább. Az alkalmazások ilyen sokszínűsége következtében a kiadók műveit olvasni igazi kihívás. Valamennyien különböző stílusú Java-programozást alkalmazunk attól függően, hogy éppen appleteken, serveleteken vagy babokon dolgozunk-e, annak ellenére, hogy ezek közül bármelyikre ráragaszthatnánk a javás webprogram címkét. Hasonlóképpen azt is fontos megérteni, milyen más XSLT-feldolgozási lehetőségeket nyújtanak a különféle termékek.

Összetett honlapfejlesztés

Neil Madden, a University of Nottingham hallgatója egy különlegesen gyors telepítésre és karbantartásra kiélezett XSLT-rendszerrel rendelkezik. Az elképzelése több részből álló webhelyen alapul, amelyet rendszergazdák csapata, szerkesztők és felhasználók alkalmaznak. A TclKitet, ezt az újszerű, nyílt forrású eszközt használja, amely egyetlen különösen pehelysúlyú, kis munkaigényű csomagban egyesíti az adatbázis- és a HTML-szolgáltatásokat. A TclKit Tcl-programokat is képes értelmezni, így a szabványos sablonokkal kiegészített tDOM-ot programozható modulokba csomagolhatja. Ezekkel látott hozzá a webhely fejlesztésének:

1. XML-dokumentum tervezése, amely képes tárolni a honlap adatait.
2. XSL-stíluslapok elkészítése, hogy az átalakított adatok minden ügyfél igényeit kielégítsék.
3. Az első két lépés ismétlése minden olyan részre, amely különleges igényeket támaszt.
4. Felhasználók, részek és lapok felvitele.

Ezeket a különböző adattípuscsomagokat (lapszerkezet, XML-források, stíluslapok) a parancsfájlosított dokumentumok foglalják magukban, és teszik egyszerűvé egy működő kiszolgáló új kiszolgálóra, lemezrészre telepítését vagy frissítését. Madden tervei szerint a tisztán webalapú szerkesztést kiváltására egy gazdagabb, gyorsabb GUI-felület is készül. A Tcl egységessége és parancsfájlosíthatósága ezt a kettős portolást akár webkiszolgálón, akár helyi GUI-felületen keresztül is egyszerűvé teszi.

A jól meghatározott modulhatárok a rendszer szempontjából létfontosságúak. A tervezők a stíluslapokkal foglalkoznak, a rendszergazdák a jogosultságokat kezelik, a szerkesztők az egyes részeket rakják össze ütközés nélkül. Azáltal, hogy ezeket a szolgáltatásokat a megbízható összetevőket egymáshoz ragasztó apró parancsfájlokként valósítjuk meg, elég könnyen adhatunk a rendszerhez új alkalmazásokat. Madden középtávú céljai közt a Wiki együttműködési hírcsoport, illetve a magas szintű bemutató (presentation) kimenet készítésére alkalmas XSP- és FOP-modulok elkészítése szerepel. Madden büszkén hasonlítja rendszerét a Cocoonhoz, a közismert Apache- és Java-alapú XML-terjesztési keretrendszerhez. Rendszere több szempontból is túlszárnyalja a Cocoon teljesítményét, a forráskódja viszont a töredéke annak.

Egy másik termelési példa a tDOM XSLT használatára ***George J. Schlitz*** MediaOne programja. Pénzügyi dokumentumokat készít XSLT-vel igen fontos (mission-critical) webkörnyezetben. Bár a terjesztést eredetileg a Xalannal kezdte, a teljesítménykövetelmények végül rákényszerítették, hogy a tDOM-ra álljon át. Mindezen felhasználásoknál alapvető fontosságú az XML-kódolt vagy XML-kódolható adatok felkutatása. Csevegőszoba-naplófájlok, hitelesítő tanúsítványok, nyomtató folyamatok, újságfotók, képernyőbeállítások, családfafeljegyzések, játékállások, alkalmazástervek, földrészleteladások, orvosi fájlok és még sok-sok egyéb adat lehet jelölt az XML-elesítésre. Ha már egyszer ebben a formátumban vannak, az XSLT-átalakítás általában a legmegbízhatóbb és -méretezhetőbb módszer az adatok egyedi felhasználáshoz történő tálalására.

Az XSLT elsajátítása

Van még mit tanulnunk az XSLT-ről mint szakmáról. Amilyen ütemben a felhasználási területe növekszik, sokkal kevesebb XSLT-ben jártas hozzáértő programozó akad, mint mondjuk object pascalos.

Az XSLT gyors terjedésének másik akadálya a kényelmetlen XML-alapú írásmód és zavaró telepítés mellett szolgáltatott vagy alkalmazott szemantikája. A legtöbb számítási nyelv, ami a Linuxvilág oldalain megjelenik, többé-kevésbé eljáráselvű (procedurális): a Java és a C programok a processzort valamilyen művelet elvégzésére utasítják, majd újabb és újabb utasításokat adnak. Az eljáráselvűség a számítás miként jében is megmutatkozik.

XSLT-tanulmány

Bár a Világhálón bőséges találhatunk anyagokat, az XSLT-ről a nyomtatott könyvek értékesebbek azok számára, akik az XSLT-t mélyebben is meg szeretnék ismerni. *Steve Holzner* Inside XSLT című műve gazdag receptgyűjteményt tartalmaz, miközben szigorú pontossággal ismerteti az XSLT kulcselgondolásait. Egyéb haszna mellett az Inside XSLT utolsó fejezetei az objektumformázásról szólnak. Az utóbbi pár hónapban igen nagy érdeklődést tapasztaltam a pdf és a hozzátartozó megjelenítési kimenetek XSLT segítségével történő előállítása kapcsán.

Ugyancsak hasznos *Doug Tidwell* XSLT könyve, *Michael H. Kay* XSLT Programmer's Reference, illetve *Khun Yee Fung* XSLT: a Working with XML and HTML című alkotása. Az XSLT majdnem teljes hivatkozási függelékkel rendelkezik, és kifejezetten jól írja le a kiterjesztés módszer (extension mechanism) működését. Az XSLT: Working with XML and HTML nagy népszerűségre tett szert azok körében, akik komolyabb grafikai vagy webes háttérrel rendelkeztek, és akik nem igazán tartják magukat programozóknak, de XSLT-projektjüket gyorsan el szeretnék indítani. Az XSLT Programmer's Reference a másik véglet: meglehetősen szabatos az elvonatkoztatások terén, ugyanakkor laza előadásmód jellemzi, emellett részletes és összefogott. Bár van egy olyan érzésem, hogy igen sok XSLT-programozó manapság a Programmer's Reference-szel kezdi XSLT tanulmányait, nem árt tudni, hogy ez kézikönyv (reference), és nem oktatókönyv (tutorial).

Az XSLT megértéséhez az utolsó ajánlatom az ActiveState Tools Corporation, Komodo IDE-je. A Komodo 0-tól körülbelül 300 dollárig terjedő felhasználási díjért cserébe nagyon hasznos XSLT hibakereső berendezést kínál, amelyhez hasonlót még egyetlen termékben sem láttam.

Az XSLT szolgáltatásait tekintve a Lisppel mutat rokonságot. A jó XSLT-programok a kívánt eredmény „lényegét” mutatják be. Egy időbeli folyamatra való összpontosítás helyett az XSLT a kívánt eredmény elérése érdekében a teljes XML-dokumentumon vagy annak jól meghatározott részein dolgozik. Ezt funkcionális megoldásnak nevezzük (a matematikai modellt idézve fel), ahol a függvények a bemenetet – mellékhatások vagy időbeliség nélkül – kimenetté alakítják. Továbbá a matematikai függvényeket különböző kombinációkban is összerakhatjuk. Az általános XSLT-szemantika jó néhány különböző átalakítást vezet be, időbeliségük megadása nélkül. A stíluslapok egyszerre alkalmazandók.
Az XSLT rendelkezik ugyan változókkal, de azok megváltoztathatatlanok (inkább feltételesen meghatározott konstansok, nem pedig változók – a ford.). Csakis egyetlen értéket vehetnek fel, és nem lehet őket például ciklusba szervezni, mint ahogyan a

```
for (i = 0; i < 10; i++);
```

utasításban tesszük.
A paraméteres vagy ismétléses műveletek közvetlenül megadott rekurziókon és ismétlődéseken (iteration) keresztül hajtódnak végre. Az XSLT változóhasználati írásmódja meglehetősen csúnya, mivel meg kell felelnie az XML kötöttségeinek. C-ben vagy Javában azt írnánk, hogy

```
if (level > 20)
     code = 3;
else
     code = 5;
```

(Egyszerűbb írásmódja a `code = (level > 20) ? 3 : 5` – a ford.)

Ugyanehhez XSLT-ben körülbelül a következőket gépelnénk be:

```
<xsl:variable name = "code">
      <xsl:choose>
           <xsl:when test = "$level &gt; 20">
                <xsl:text>3</xsl:text>
           </xsl:when>
           <xsl:otherwise>
                <xsl:text>5</xsl:text>
           </xsl:otherwise>
      </xsl:choose
</xsl:variable>
```

Mint a fenti példa is bizonyítja, bár az XSLT önmagában is elég hatékony az általános feladatok megoldásában, a legjobban mégis kettős programozási módban használhatjuk. Az XSLT igazi ereje inkább a sablonok feldolgozásában, a mintaazonosításban (pattern matching), illetve az XML-elemek rendezésében és csoportosításában rejlik. ***Steve Ball***, a Zveno Pty. Ltd. vezető tanácsadója csak annyit dolgozik XSLT-ben, amennyit még célszerűnek érez, majd az eredményt egy másik nyelvet használó alkalmazásba illeszti be, hogy a külső rendszerek felületeit kezelhesse – ideértve a fájlrendszert és a felhasználói nézetet. Azok között a fejlesztők között, akikkel az utóbbi fél évben találkoztam, a Java és a Tcl volt a legnépszerűbb, de a Python, a C, a Perl és más partnernyelvek többé-kevésbé ugyancsak megfelelően támogatják az XSLT-motorokat. Továbbá az XSLT olyan kiterjesztést lehetővé tévő módszereket is meghatároz, amelyek segítségével a fejlesztők új szemantikát vihetnek az XSLT-be, ezek: ***extension elements*** (kiterjesztett elemek), ***extension functions*** (kiterjesztett függvények) és ***fallback processing*** (tartalék feldolgozás).
Az XSLT végső sorsa még nem teljesen egyértelmű. Ebben a tekintetben hasonló cipőben jár, mint a Java. Öt évvel ezelőtt úgy tűnt, hogy a Java egyetlen célja ügyes kis vizuális alkalmazások készítése. Azóta kiderült, hogy a nehézsúlyú üzleti kiszolgálók jobb otthont adnak a Java-programozásnak. Egyelőre még korai lenne meghatározni, hol tudjuk az XSLT-t használni. A ReportLab Inc. például olyan üzleti szolgáltató, amely igen komoly minőségkövetelményű leíráskészítéssel járó termékeket és szolgáltatásokat ad el a világ legnagyobb szervezeteinek, például a Fidelity Investmentsnek és az American Insurance Groupnak. A ReportLab alapítója, ***Andy Robinson*** elmesélte nekem csapatának XML-átalakító projektek készítése közben szerzett tapasztalatait. A cégüknél minden befejezett projekt XSLT helyett valamilyen pehelysúlyú parancsnyelvben íródott. Igaz ugyan, hogy az XSLT kifejezetten XML-átalakításra szakosodott, de a tanácsadó csapatok egyszerűbbnek látták az általánosabb célú, de hatékony Python nyelvet használni.

Köszönetnyilvánítás

Különös köszönettel tartozom ***Rolf Ade***-nek, aki bevezetett a tDOM program világába, és segített megérteni.

Cameron Laird
a Phaseit Inc. alelnöke és főállású fejlesztője. Gyakran ír programozással kapcsolatos cikkeket, és az elmúlt évben számos XSLT-vel kapcsolatos cikket adott közre. Jelenleg egy a nyelvet oktató tanfolyam előkészítésén dolgozik.

Kapcsolódó címek

Az XSL, XSLT és Xpath megismerését kezdjük a W3C ajánlásával a ➲ http://www.w3.org/TR/xsl, ➲ http://www.w3.org/TR/xslt és a ➲ http://www.w3.org/TR/xpath címen.

A FO-ról (azaz az objektumformázásról) is találunk itt részleteket ➲ http://www.w3.org/TR/xsl/slice6.html#fo-section

Lars Marius Garshol összegyűjtötte a leginkább elérhető XSLT-motorokat. Listája megtekinthető a ➲ http://www.garshol.priv.no/download/xmltools/cat_ix.html#SC_XSLT címen.

A tDOM-mal kapcsolatos legfrissebb adatokat a ➲ http://mini.net/tcl/tDOM címen találjuk.

A SAX-szal a ➲ http://www.megginson.com/SAX helyen ismerkedhetünk.

A Cocoon adatait az ➲ http://xml.apache.org/cocoon lapon találjuk.

A héjjal ellátott dokumentumokról (scripted documents) a ➲ http://mini.net/tcl/ScriptedDocument címen olvashatunk.

Hiperhivatkozásokat és más, a saját XSLT-projektjeimhez tartozó adatokat, véleményeket találhatnak az alábbi címen: ➲ http://starbase.neosoft.com/~claird/comp.text.xml/XSLT.html