Bíró Szabolcs

Van új a nap alatt – XML alapú webtartalom-generálás Cocoon rendszerrel*

Az XML és az XSL nyelvek világhálón való egyre szélesebb körű elterjedésével mára már lehetőség nyílt olyan – akár egyszerűnek is nevezhető – szerveroldali alkalmazásokra, amelyek magas szintű feldolgozásra képesek. Az említett nyelvek ingyenes hozzáférhetőségéből és platformfüggetlenségéből adódóan ezek az alkalmazások egyrészt nem a webkiszolgálók és különféle szkriptek (pl. CGI) összetételéből állnak, másrészt az olyan védjegyzett technológiákat is mellőzik, mint pl. a .Net vagy az Enterprise JavaBeans. A multinacionális cégek, kiadók vagy tartalomszolgáltatók – könyvtárak is! – a webre szánt tartalmakat egyre inkább olyan minden kétséget kizárónak tekintő hatékony („globális") adatstrukturálási és archiválási formátumokban szeretnék előállítani, mint az XML, hogy aztán ebből igénytől függően bármikor (X)HTML, PDF, WML stb. formátumokat tudjanak költséghatékonyan létrehozni. Az Apache projekt Cocoon terméke kísérlet egy olyan rendszer előállítására, amely a felsorolt képességekkel rendelkezik.

Hangsúlyeltolódás a webes tartalomleírásban

A leggyakoribb problémák

Azok, akik nagyobb honlapok karbantartásával foglalkoznak, egyetértenek abban, hogy a frissítés/átalakítás hagyományos módja néha kész rémálom. 10 évvel ezelőtt, amikor az első statikus HTML oldalak megjelentek a világhálón, még szimpla böngésző–webszerver kommunikációval modellezhető volt a kapcsolat és a tartalomszolgáltatás módja, ám ma már korántsem ilyen egyszerű a helyzet. Ennek legfőbb oka, hogy sok olyan formátumot használunk, ahol a stílus és a közlendő információ keveredik. Legjobb és mindenki által ismert példa a HTML, ahol a weblap tele van szórva színt, megjelenést, formázást leíró paraméterekkel – nem beszélve az esetleges szkriptnyelvek kódjairól. Egy szép weblapot egyszer összeállítani „könnyű", de utána a tartalom és a stílus következetes változtatásai akár irdatlan mennyiségű munkával is járhatnak.

A fejlettebb tartalomszolgáltatók ezt rendszerint úgy oldják meg, hogy a tartalmakat dinamikusan generálják valamilyen stylesheet alapján, az információt pedig, amellyel a lapot kitöltik, adatbázisból veszik. A megoldás hátránya, hogy a területen mindmáig nem létezik egységes szabvány, és ez a megközelítés szinte mindig fejlesztői munkával jár – pedig nem lenne szükségszerű.

A másik gyakran felmerülő gond az, hogy a tartalom készítője és a megjelenésért felelős gyakran egy és ugyanaz a személy. A dolog eredménye, hogy a webes tartalmakat sokszor „webguruk" készítik, ami csinos megjelenést, de gyenge információt eredményez. Mindez nem csoda, hiszen egyértelműen mások a kompetenciák, és világosan látszik, hogy ezt két olyan személynek kellene végeznie, akik egymás munkájáról nem is feltétlenül akarnak tudni. Online publikálás esetében azonban e két szerepet meglehetősen nehéz elválasztani: a „tartalomgyártótól" gyakran várják el, hogy értsen pl. a HTML nyelvhez, pedig józan ésszel belegondolva ahhoz semmi köze. Bárhogy is van, látni fogjuk, hogy a Cocoon megoldást kínál erre a problémára is.

A megoldáskeresés útjai

Lépcsős stíluslapok – CSS

A tartalom és a forma szétválasztásának jelentőségét legkorábban felismerő és azért legtöbbet tevő szervezet a webes ajánlások kidolgozásáért felelős *World Wide Web Consortium (W3C)* volt. Első, meglehetősen hatékonynak bizonyuló és egyre gyakrabban használt megoldási kísérletük a *Cascading Style Sheets (CSS)*[1] specifikáció volt, amely a HTML nyelvhez készült. Segítségével bizonyos fokig már szétválasztható a tartalom a

* A Networkshop 2005 konferencián (Szeged, 2005. március 30–április 1.) elhangzott előadás alapján.

formától, ám mivel a „nyelv" leginkább elem-, bekezdés- és karakterformázási képességekkel rendelkezik, ráadásul implementációja a jelentősebb böngészőkben sem mindig konzisztens – bár ez utóbbi a gyártók hibája –, alkalmazása elsősorban munkacsökkenést, mintsem teljes körű megoldást jelent a tartalom és a forma gyors karbantartásában. Mi tehát a megoldás?

Bővíthető jelölőnyelv – XML

A W3C munkatársai is ezen a kérdésen gondolkozhattak el, amikor egy olyan nyelv – keretrendszer – kidolgozásába kezdtek, amely kizárólag a tartalom leírására szolgál. A megvalósításhoz persze nem kellett a nulláról indulniuk, hiszen kiindulási alapnak rendelkezésre állt az a lassan 20 éve ISO szabványként elfogadott metanyelv, amely a *Standard Generalized Markup Language (SGML)* nevet viseli. Az SGML szabványos jelölőnyelv dokumentumok belső szerkezetének leírására, beleértve az egyes elemeket jelölő címkék (tagek) definiálásának módját is. Segítségével elvben bármilyen dokumentum leírható, függetlenül az azt tároló és megjelenítő számítógépes környezettől. Nagyfokú bonyolultsága és kevésbé költséghatékony alkalmazhatósága miatt azonban nem terjedt el sem Európában, sem világszerte a várt mértékben. A technológia viszont a maga nemében nagyszerű és egyedülálló, kár lett volna kiaknázatlanul hagyni. Éppen ezért 1998-ban, miközben a világ a HTML újabb és újabb reinkarnációira figyelt, a W3C egy gyökeresen új tartalomleíró metanyelvet épített fel az SGML-ből – az új jelölőrendszer az *eXtensible Markup Language (XML)* nevet kapta. Ez az a formátum, amely meg fogja menteni a digitális világot – vallják sokan –, és igazuk lehet, hiszen amellett, hogy az XML levetette elődje hátrányos tulajdonságait, és megvalósította a tartalomleírás követelményeit, egyúttal alkalmazásorientált is lett – ami az SGML-re nem volt igaz. Rendelkezésre állt tehát a nyelv, amely a tartalom leírására és tárolására szolgál, ugyanakkor felvetődik a kérdés: Mi lesz a formával?

Bővíthető stíluslapnyelv (transzformáció) – XSL(T)

A kérdés jogos, hiszen a CSS elsősorban a HTML nyelvhez készült, az XML esetében csak attól kezdve lett alkalmazható, hogy a böngészőprogramokat felruházták a belső fastruktúra értelmezésére képes parserrel. Az ilyen megjelenítésnek azonban több hátránya is van, nem beszélve arról, hogy ez esetben továbbra is az XML állományt szolgáltatjuk, csak lépcsős stíluslappal „sminkelve".

Ennél azonban jóval többre van szükség, olyan stíluslapra, amely transzformációra is képes. Ennek fényében született meg az *eXtensible Stylesheet Language (XSL)* specifikáció, amely az XML dokumentumok stílusleírására való ismert nyomdai leírónyelvek képességeit adta az XML-nek. Végül az egészet az *eXtensible Stylesheet Language Transformations (XSLT)*[2] XSL-bővítés tette kerekké. Az XSLT-vel bővített stíluslapok meg tudják mondani, hogyan alakuljon át a tartalmat magában foglaló XML dokumentum olyan formátumúvá – XML, (X)HTML, PDF, WML stb. –, amely böngésző- vagy más programokkal szolgáltatható.

A továbblépés lehetősége

Sikerült tehát a tartalmat elválasztani a formától – a megoldás rendelkezésre áll, kérdés, hogy mennyire alkalmazható. Az ilyen új dolgok jelentősége természetesen mindaddig teoretikus, amíg a technológiákat nem lehet a gyakorlatban is kihasználni. A szakmabeliek tudják, hogy XML elemzők már „régóta" léteznek, de ezeket csak manapság kezdik egyre szélesebb körben használni. Körülbelül 4-5 éve fejlesztenek XSLT stíluslap-feldolgozókat – Xerces, XSLTproc, Saxon, Xalan stb. – és olyan XSL-konvertereket, amik XSL-lel formázott dokumentumokat a már említett megjelenítési formátumokká képesek átalakítani. Az XML Apache projekt több ilyen szoftvert kapott meg a fejlesztőktől további nyílt forráskódú fejlesztésre, de ezek eddig nem kavartak nagyobb vihart – egészen mostanáig...

Az Apache Cocoon[3]

Az eddigi munkára a Cocoon PHP-s változata, a *Popoon*[4] projekt tette fel a koronát, mert az elkészített szoftverelemeket egységes keretrendszerbe foglalta, lehetővé téve, hogy teljes honlapok épüljenek XML-XSL transzformációkra. Az ilyen weboldalak pedig nagyon jó képességekkel rendelkeznek, és ráadásul W3C ajánlásokon alapulnak. Mindenekelőtt a stílus és a tartalom teljesen szeparált, ami azt jelenti, hogy akár naponta átalakulhatnak, és teljesen új keretben jeleníthetik meg az információkat, dokumentumokat. Ugyanakkor a tartalom készítőinek a dolga nagyban leegyszerűsödik, mert a webes tartalomleíró nyelvek „primitivizmusai és korlátai" távol kerülnek tőlük.

Az olyan megjelenítési formátumok, mint pl. a(z) (X)HTML, PDF más aspektusba kerülnek, mert XSLT stíluslapokkal ugyanazt az XML dokumentumot egészen könnyedén teljesen más formá-

tummá lehet alakítani. Bizonyos szempontból még a böngészőprogramok versenye is jelentéktelenné válik, hiszen a megjelenítésért felelős a kliensoldalon teljesen elegendő, a publikáló keretrendszer (Cocoon) veszi át a formázással és a stílus kezelésével kapcsolatos teendőket. Végül, de nem utolsósorban a technológia szabványokra épül, következésképpen nem kell a gyártófüggő formátumokat megtanulni.

Középpontban a Cocoon

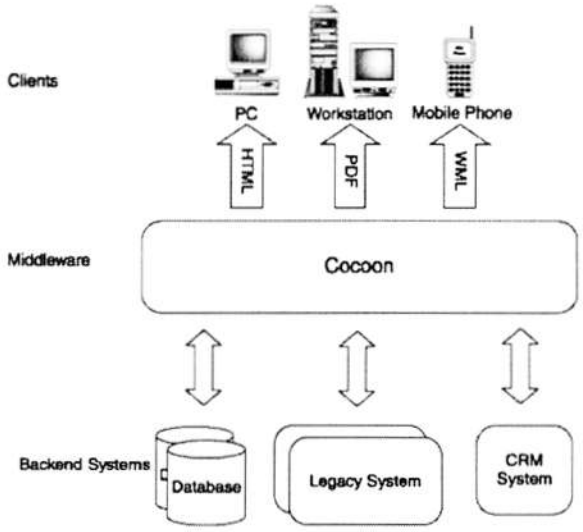
Ezután a meglehetősen hosszúra nyúlt, ám korántsem felesleges bevezető után tegyük fel végre a kérdést: Mi is valójában a Cocoon? Nos, nem más, mint egy szabad forrású, Java alapú, az Apache-ba épülő XML publikáló tartalomszolgáltatási keretrendszer.

A *Cocoon* projekt 1998-ban indult, ugyanis egy *Stefano Mazzocchi* nevű olasz diákot nagyon zavarták a HTML korlátai az Apache honlapjának átalakítása során. Elhatározta, hogy XML és XSL alapokon egy új szoftvert hoz létre, melyben elkülöníti a honlapkészítés egyes részeit (tartalom, forma, logika és site-architektúra), hogy azok különböző személyek által egymástól függetlenül is fejleszthetők legyenek.[5] A Cocoon fejlesztői csapatában megközelítőleg nyolcan vannak. Ezek a fejlesztők az új kódok helyességét ellenőrzik, és amennyiben szükség van rá, a forrásán is változtatnak. Néhány „külső" programozó is támogatja a projektet, mégpedig az új komponensek ellenőrzésével vagy hibakereséssel. Az így létrejött alkalmazás legfontosabb tulajdonsága, hogy képes együttműködni a meglévő J2EE megoldásokkal, HTML, WML, PDF, SVG, RTF kimenetet tud produkálni, és nem utolsósorban teljesen ingyenes technológiákra épül. Az pedig már csak hab a tortán, hogy a Cocoonra egy Lenya nevű CMS rendszert is felépítettek, Xindice XML adatbázis-kezelőt tartalmaz, beépített keresőként pedig a Lucene-t használja.

Az architektúra logikai elhelyezkedésének megértését elősegítendő, az *1. ábra* azt illusztrálja, hogy a rendszer miként fér hozzá az adatokhoz eltérő rendszerekből, majd hogyan „mossa össze" őket a különböző formátumokban történő kliensoldali szolgáltatás céljából.

A rendszer működ(tet)ésének logikája

A Cocoon rendszerrel a feldolgozandó XML dokumentumokat, a rajtuk végrehajtott transzformációkat lehet meghatározni, végül a felhasználók számára a kívánt formátumban megjeleníteni. A szoftver arra is lehetőséget ad, hogy az XML fájlok maguk is tartalmazhassák a feldolgozás algoritmusát, melynek következtében dinamikusan generálhatóvá válnak.



1. ábra **A Cocoon alapú, „hármas kötésű" architektúra**

Minden Cocoonon alapuló webes alkalmazás három fő feladatcsoportból áll:

- az adattartalomról való gondoskodás;
- a működési logika megvalósítása, karbantartása;
- a megjelenítés megvalósítása, karbantartása.

Amellett, hogy kizárólag eme feladatcsoportok szétválasztásával biztosítható a szolgáltatandó honlap vagy webes alkalmazás stabilitása, naprakészen tartása és erőforrásigényének minimalizálása, egyértelműen körvonalazódik az is, hogy a működtetés során szétválnak a tartalomért és a formáért felelős kompetenciák. Számunkra pedig ez az egyik legfontosabb tényező, ugyanis ezzel a módszerrel végre elérhető, hogy a „tartalomgyártó" személy kizárólag a tartalmi részekért, a designer pedig a megjelenéséért feleljen.

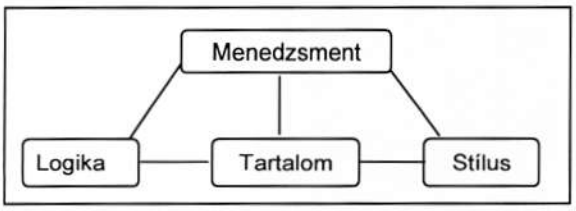
A „hagyományos" webfejlesztő eszközök – PHP, ASP – alkalmazásakor a megjelenítés (arculat) és a működési logika szorosan összefonódik, elsősorban a nyelvi kódok keveredése miatt. Ezáltal az alkalmazás karbantartása, a hibák javítása nehezebb és költségesebb. Az arculat megváltoztatása gyakorlatilag az egész alkalmazás újraírását követeli meg.

1. táblázat
Feladatcsoportok szétválasztása Cocoon esetén

Fejlesztő	Rendszergazda	Tervező	Szerkesztő
A fejlesztők valósítják meg a működési logikát, és a webes alkalmazás mögött húzódó objektummodellt Nem a weboldalak kinézetével, és nem is a képernyőn olvasható szöveggel van dolguk, hanem az alkalmazás funkcionalitásával	A rendszergazda a felelős a webszerver által kiszolgált URI tér kiosztásáért: az URL-nek a megfelelő feldolgozási folyamatdefinícióval való megfeleltetéséért Dedikált rendszergazda hiányában egy fejlesztőnek kell ezt a szerepkört ellátnia	A tervezők felelősek a honlap végső formájának kidolgozásáért A tervezők készítik a grafikákat és a HTML kódot	A szerkesztők felelősek a képernyőn megjelenő szöveg tartalmáért, és részben a szerkezetéért Általában ők használják fel a fejlesztők által elkészített eszközöket, és állítják elő azt az XML tartalmat, amelyből a transzformációk végén megjelenik az eredmény Kisebb webes alkalmazások esetén gyakran a fejlesztő látja el ezt a szerepkört is
Kapcsolódó Cocoon-komponens: **actions**	Kapcsolódó Cocoon-komponens: **sitemap**	Kapcsolódó Cocoon-komponens: **transformers**	Kapcsolódó Cocoon-komponens: **generators**

A Cocoon elősegíti ezeknek a feladatcsoportoknak a szétválasztását azáltal, hogy a webre kerülő alkalmazás fejlesztésében közreműködő szereplők a rendszeren belül csak egy-egy komponenssel találkoznak. Ideális esetben az *1. táblázat* szerinti a munkamegosztás – persze kellő indokkal összevonások alkalmazhatók.

A táblázatban megnevezett szerepkörök kapcsolatát a *2. ábra* szemlélteti.



2. ábra **A feladatcsoportok közti szerepkörök kapcsolata a Cocoonban**

Cocoon a Neumann-házban

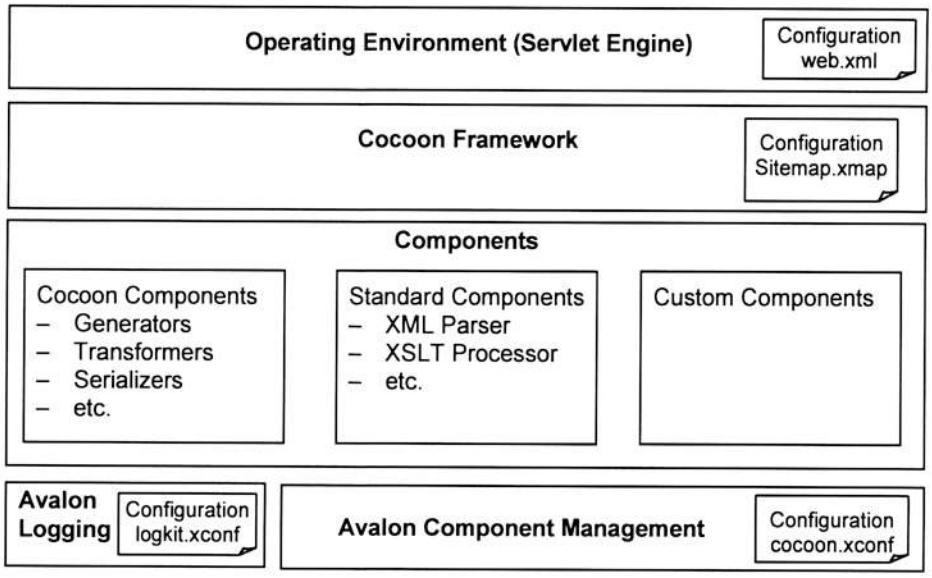
A digitális könyvtár új honlapjának tervezése közben már előre körvonalazódott, hogy a hagyományos módszerekkel hatalmas munkát igényel a site elkészítése. Tehát valami olyan megoldást kellett találni, amivel – ha nem is feltétlenül gyorsabban, de – ésszerűbb és időtállóbb alkalmazás hozható létre. Figyelembe véve, hogy az intézmény irodalmi művek webes publikálására már hét éve SGML/XML és transzformációs stíluslap-technológiákat alkalmaz, egyértelműen jó döntésnek bizonyult az említett nyelveket ugyancsak használó Cocoon XML publikáló keretrendszer bevetése.

Kézenfekvő volt az is, hogy az XML technológiára való áttérés során befektetett munka a honlap üzemeltetése, karbantartása során térül meg. Ráadásul az egyszerűbb kezelhetőségen felül az XML olyan többletlehetőségeket is nyújt, mint a klienseszköz megjelenítési képességeihez való jobb alkalmazkodóképesség, pl. a honlap WAP felületű megjelenítése.

A Cocoon rendszerrel szállított építőelemek elég rugalmasak ahhoz, hogy ezeknek az elemeknek a használatával, programozási munka nélkül is megoldható legyen a webfejlesztési feladatok túlnyomó része. Ezen túlmenően a bonyolultabb alkalmazásoknál a menet közben Java kódra forduló, XML szintaktikájú XSP nyelvet lehet alkalmazni, sőt a meglévők mellé saját építőelemeket is lehet fejleszteni. Mindehhez bőséges példaanyag és dokumentáció áll rendelkezésre, valamint – legjobb példaként – maga a Cocoon forráskód.

A megvalósításban alkalmazott alkotóelemek a *3. ábrán* láthatók.

Az első látásra bonyolultnak tűnő ábra valójában nagyon könnyen megfejthető. A felső szinten a feldolgozó környezet (servlet engine) látható, melynek konfigurálását a web.xml állományban lehet elvégezni. Alatta a Cocoon Framework és a hozzá kapcsolódó sitemap.xmap, amely a rendszer működését definiálja. Ezeket pedig az általános és saját fejlesztésű Cocoon-komponensek (az 1. táblázatban látható feladatcsoportok szétválasztása is részben ezek alapján történt) követik.



3. ábra **Az új Neumann-ház honlapkészítése során alkalmazott Cocoon-komponensek**

Mindegyikkel nincs lehetőségünk foglalkozni, de néhány egyszerű forráskóddal támogatott példán keresztül érdemes egy pillantást vetni a keretrendszer működésére.

A sitemap

A sitemap a Cocoon szíve. Tulajdonképpen ez határozza meg az egész rendszer működését, viselkedését – ezen áll vagy bukik minden. Ha jól van beállítva, akkor a kimeneti állományban megjelennek a képek, működik a külső CSS, használhatunk scripteket stb. A sitemap az *1. forráskód* szerinti globális struktúrát követi.

```
<map:sitemap xmlns:map="http://xml.apache.org/
cocoon/sitemap/1.0">
  <map:components/>
  <map:views/>
  <map:resources/>
  <map:action-sets/>
  <map:pipelines/>
</map:sitemap>
```

1. forráskód **A globális sitemap struktúra**

A map névtér minden egyes részének megvan a maga funkciója, ám egyszerűbb alkalmazások esetén a pipelines használata elegendő. Nézzünk egy nagyon egyszerű példát, amellyel a legtöbb programozási nyelvkönyv indítani szokott. A „Hello World" szöveget tartalmazó XML fájlból a Cocoon segítségével írassuk ki egy kliensoldali böngészőbe az említett üzenetet.

Az XML fájl felépítését a *2. forráskód* mutatja.

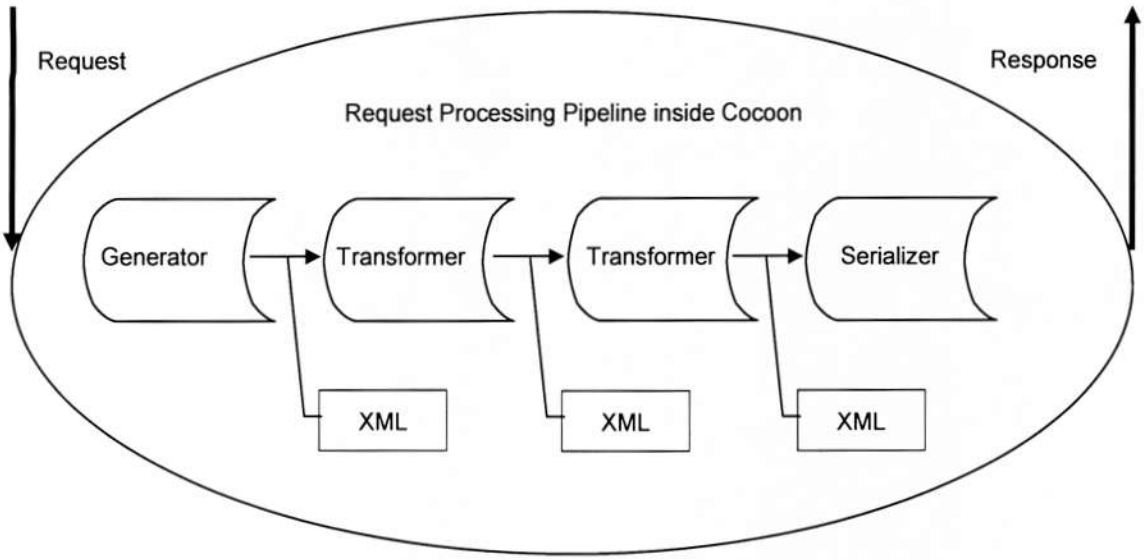
```
<?xml version="1.0"?>
<dokumentum>
  <szoveg>Hello World</szoveg>
</dokumentum>
```

2. forráskód **A helloworld.xml állomány tartalma**

Ahhoz, hogy ebből HTML kimenet legyen, készíteni kell egy stylesheetet, amelynek segítségével a Cocoon elvégezheti a transzformációt[6] *(3. forráskód).*

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
  <xsl:output indent="yes"
          method="html"
          encoding="utf-8"
          doctype-public="-//W3C//DTD HTML 4.01
                  Transitional//EN"
          doctype-system="http://www.w3.org/TR/html4/
                  loose.dtd"/>
  <xsl:template match="dokumentum">
    <html>
      <head>
        <title><xsl:value-of select="szoveg"/></title>
      </head>
      <body>
        <h1><xsl:value-of select="szoveg"/></h1>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

3. forráskód **A helloworld2html.xsl állomány tartalma**



4. ábra **A kérést feldolgozó pipeline felépítése**

A kód nem tesz mást, mint sablont definiál a dokument elemre, amelynek tartalmát HTML vázba helyezi. Az XML állomány *szoveg* elemének tartalma *<h1>* elemek között íródik ki a böngészőben. Végül nincs más hátra, mint a sitemapben beállítani, hogy mi is történjen *(4. forráskód)*.

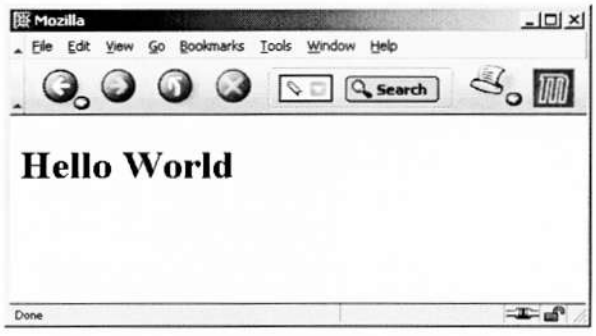
```
<map:pipeline>
  <map:match pattern="helloworld">
    <map:generate src="helloworld.xml"/>
    <map:transform src="helloworld2html.xsl"/>
    <map:serialize/>
  </map:match>
</map:pipeline>
```

4. forráskód **A sitemap.xmap állomány tartalma**

A pipeline megmondja a Cocoonnak, hogy a kimeneti állomány helloworld néven szerepeljen az URL-ben, a helloworld.xml-t használja forrásként, a transzformációt pedig a helloworld2html.xsl segítségével végezze el. Egy állományon több átalakítás is elvégezhető, ilyenkor minden egyes transzformációért felelős stíluslapot külön <map:transform src="*.xsl" />-be kell elhelyezni. A pipeline működését és a feldolgozási folyamatot a *4. ábrán* szemléltetjük.

A böngészőben látható kimenet pedig a várt eredményt hozta *(5. ábra)*.

A kicsit hozzáértőbbek rögtön mondhatják, hogy a látottakat számos más úton ugyanígy el lehetett volna érni, és még sitemap sem kell hozzá. Részben igazuk van, ám egy komplex, webportálszerkezethez hasonló felépítésű honlap esetében – amilyen a *6. ábrán* látható új Neumann-ház site nyitólapján szerepel –, mindez már nem jelenthető ki ilyen egyértelműen.



5. ábra **A HTML kimenet Mozilla 1.7.5-ben**

A 6. ábrán kitűnően látszik, hogy a kezdőlap 3 hasábos szerkezetű: mindegyik tartalma egy-egy XML állományban van tárolva – ezek HTML transzformációját XSLT végzi. A bal oldali menüsor és a középen elhelyezkedő kiemelt szolgáltatások esetében saját tervezésű XML jelölést használtunk, ugyanakkor a jobb oldalon látható újdonságoknál az RSS[7] szabvány XML Schemáját vettük alapul a tartalmak kódolásakor. Az oldal tetején elhelyezkedő fejrész és „lábléc" tartalma transzformációs bővíthető stíluslapállományban tárolódik, és abból alakul át HTML-lé, a létrejött oldalszeleteket pedig a Cocoon cinclude komponense olvasztja egységes, böngészőben megjeleníthető kimeneti állománnyá.

6. ábra A **Neumann-ház nyitólap szerkezeti felépítése Deer Park Alpha 2-ben**

Külön említésre méltó, hogy a site aloldalainak XML kódolása során a TEI[8] P4 XML alapú DTD-jét hívtuk segítségül – ennek eredményeként tartalmaink nemcsak XML alapúak, hanem nemzetközileg is elismert, egységes jelölésrendszert követnek. A TEI alapú jelölés során alkalmazott fejlécet pedig – amely egyébként a feldolgozott dokumentum leírására szolgál – a HTML output <head> részének alap DC metaadataival való feltöltéséhez vettük igénybe.

A Cocoon és az XML technológia alkalmazásának előnyeit bizonyítja az is, hogy ugyanabból a tárolási formátumban rögzített tartalomból egy másik stylesheet segítségével könnyedén előállítható a honlap vakok és gyengén látók számára is használható felülete *(7. ábra)*.

Továbbfejlesztési lehetőségek

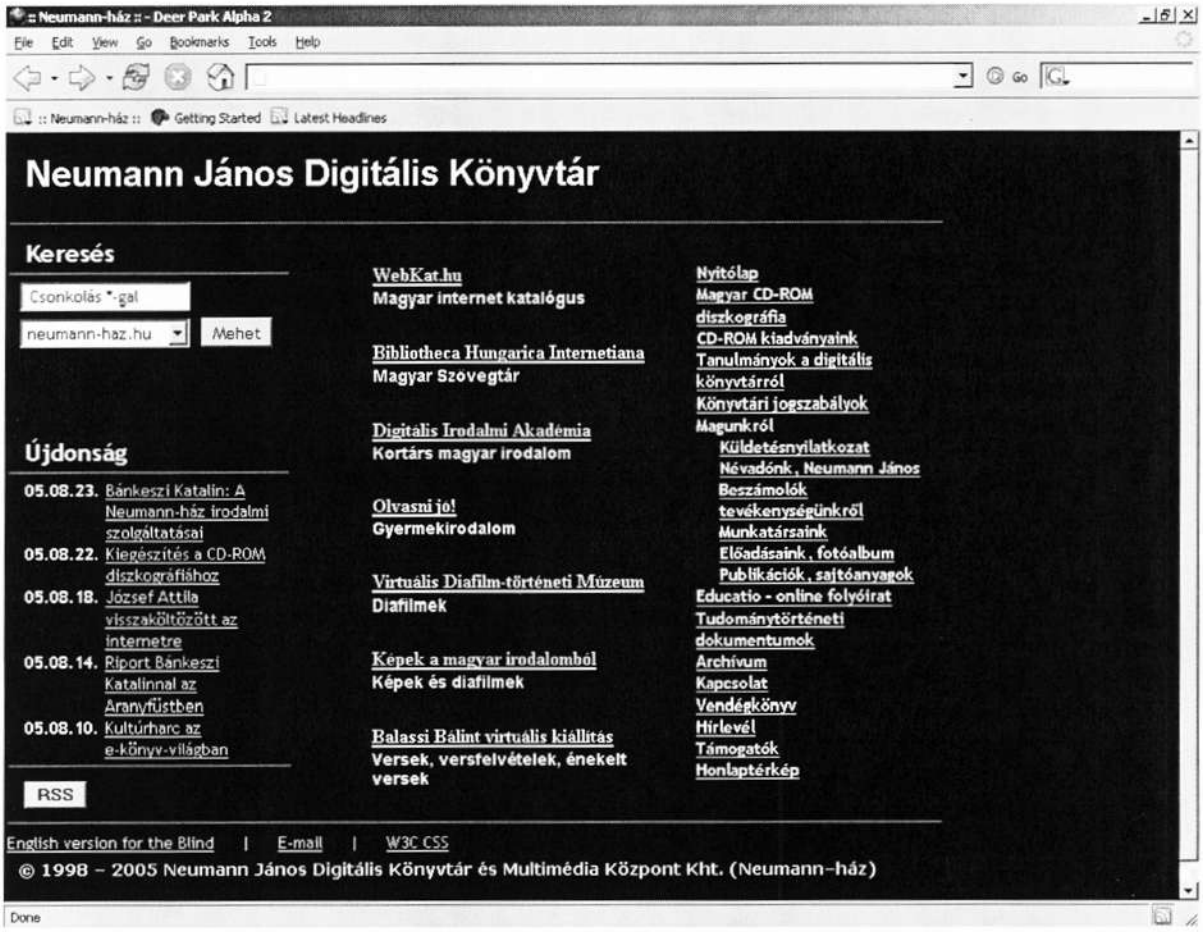
Az XML technológia és a Cocoon bevezetésével az előbbiekben vázolt projekt célkitűzésein túlmenően számos új szolgáltatásra nyílik technikai lehetőség. Röviden áttekintjük az új lehetőségeket, amelyekkel a jövőben bővíteni lehet a Neumann-ház honlapját.

Mobil szolgáltatás

A kliensoldali böngésző (user agent) képességeinek lekérdezésével a Cocoon lehetőséget ad az XML formában tárolt tartalmaknak a felhasználó igényei szerinti formátumra konvertálására. Így ugyanaz a tartalom megjeleníthető asztali számítógépen és mobiltelefonon egyaránt. Ennek a lehetőségnek például az újdonságok, hírlevelek célba juttatása szempontjából lehet nagy jelentősége, de igényt tarthat azoknak a felhasználóknak az érdeklődésére is, akik a szakirodalomi, oktatási anyagokban szeretnének valaminek utánanézni, valamit megkeresni.

Regisztrációhoz kötött szolgáltatások

A Cocoon hitelesítő komponense lehetővé teszi, hogy csak a megfelelő jogosultsággal bejelentkező, regisztrált felhasználók férhessenek hozzá a védett információkhoz. A regisztráció lehet ingye-



7. ábra **A Neumann-ház nyitólapja – vakok és gyengén látók számára készített felület szerkezeti felépítése Deer Park Alpha 2-ben**

nes, de történhet díjfizetési kötelezettség mellett is. A felhasználói azonosító adatokat a Cocoon fájlból, adatbázisból vagy számos egyéb adatforrásból, pl. az LDAP[9] kiszolgálótól lekérdezve ellenőrzi. A sikeres bejelentkezés után a Cocoon session-kezelő komponense biztosítja, hogy minden azonosított felhasználó kizárólag a saját tranzakcióival kapcsolatos kommunikációt végezhesse.

Szindikálás

A Cocoon szindikálási funkciója több – akár külső – kiszolgálótól bekért XML/XHTML/HTML tartalom közös felületen, egyetlen oldalon való megjelenítését jelenti. Így összefüggéseiben mutathatók be az adatok: összehasonlíthatók, összerendezhetők.

Hírcsatornák

Külső hírszolgáltatók XML formátumban – például az RSS szabvány szerint – szolgáltatott online híreit a Cocoon fogadja, és a honlap hírblokkjába integrálva megjelenítheti. Saját hírcsatornát is lehet indítani,[10] amelyre más szolgáltatók felhasználhatnak, publikálhatnak.

Portálszolgáltatások

A Cocoon beépített portál komponenssel rendelkezik, amely integrálja és kibővíti a többi Cocoon-funkciót (hitelesítés, szindikálás stb.). A portálra bejelentkezett felhasználók érdeklődési körüknek megfelelően alakíthatják ki a személyes portálfelületet: egyes elemeket kikapcsolhatnak, elrejthetnek, minimalizálhatnak. A hagyományos menüs oldalszerkezet mellett a Cocoon támogatja a fülekkel (tab) való navigációt is.

Adatbázisháttér – Xindice

„Az Xindice a Cocoon rendszer része, így külön telepítést nem igényel, a meglévő Cocoon környezetben közvetlenül használható. Valójában egy natív XML adatbázis kezelő, vagyis közvetlenül képes XML

adatokat tárolni és visszakeresni. Míg a relációs adatbázis kezelők az XML adatokat csak konverzió, adat-objektum megfeleltetés (mapping) segítségével tudják kezelni – ami a lazán strukturált, szöveges XML dokumentumok esetén erőforrás-igényes és nem hatékony –, addig a natív XML adatbázisok közvetlen, konverzió nélküli XML adatkezelésre alkalmasak.
Az XML fájlokat az adatbázisban logikai csoportokba, ún. collection-okba rendezve tárolhatjuk, így ugyanaz az adatbázis példány többféle típusú és tartalmú dokumentum-halmaz tárolására és visszakeresésére használható. A collection-ok az operációs rendszerek fájlrendszereihez hasonlóan hierarchikus rendszert alkotnak, tetszőlegesen egymásba ágyazhatók.
Az Xindice a keresés szintaktikájaként a W3 konzorcium XPath szabványát használja. A keresést nemcsak dokumentumon belül, hanem bármelyik collection-szinten, a collection-ba tartozó összes dokumentumban egyszerre hajtja végre. Eredményként azt az elemhalmazt (node set) kapjuk vissza, amely teljesíti az XPath keresésben meghatározott feltételt. A kapott XML dokumentum az eredeti dokumentumról is tartalmaz azonosító információt, így könnyen összekapcsolható a megtalált elem a hozzá tartozó gazdadokumentummal. A keresés hatékonyságának fokozására indexeket lehet készíteni a gyakran használt keresési szempontok alapján.
Az XPath nyelv segítségével a dokumentum tetszőleges részét ki lehet választani, így az adatbázisban, egyetlen példányban tárolt műből például többféle idézetet lehet lekérdezni" (*Szalai Attila*: Jeles napok).

Ha például egy *Juhász Gyula*-verset tartalmazó XML állományt töltünk a proba nevű collectionbe, az első két versszak így kérdezhető le: *http://localhost:8888/samples/blocks/xmldb/db/proba/juhaszgy?xpath=//lg[position()<3]*

A lekérdezés eredményének forrása pedig a következő formát ölti:

```
<db:results query="//lg[position()<3]" resources="2">
  <db:result docid="juhaszgy">
    <lg src:col="/db/proba" src:key="juhaszgy">
      <l>Gondolj el nem múló zenékre lelkem,</l>
      <l>Szűz csillagokon fönn az égi kertben.</l>
    </lg>
  </db:result>
  <db:result docid="juhaszgy">
    <lg src:col="/db/proba" src:key="juhaszgy">
      <l>És éjszakára, melynek tükörében</l>
      <l>Elsápad minden árnyék földön, égen.</l>
    </lg>
  </db:result>
</db:results>
```

Konklúzió

A Cocoon és a hozzá kapcsolódó technológiák áttekintése után felmerülhet a kérdés: miért jó mindez a tartalomszolgáltatók, könyvtárak számára? Első látásra bonyolultnak tűnik, viszonylag nagy szakértelmet igényel, és rengeteg területen kell otthonosan mozogni. Tervezni, kódolni, programozni kell, és emellett számos szoftver használatát kénytelen elsajátítani, aki alkalmazza. A feladat nem egyszerű, de kihívást jelent, hiszen abban a világban, ahol piaci erők dominálnak, áldás, ha rendelkezésre állnak olyan ingyenes technológiák, amelyekkel kifogástalan munkát lehet végezni.

Napjaink könyvtárainak – ha úgy tetszik, információs központjainak – is meg kell felelniük a 21. század embere által támasztott igényeknek, és az egyre gyorsabban változó technológiai követelményeknek. A mindennapjainkat behálózó online kommunikáció és az egyre fontosabbá váló digitalizáció világában minden kétséget kizáróan lényegessé válik az adatok, információk hordozhatósága. A különböző operációs rendszereket és böngészőprogramokat alkalmazó és futtató asztali, illetve hordozható számítógépek, okos telefonok és PDA-k, nem utolsósorban pedig azok felhasználói egyre jobban „megkövetelik", hogy digitalizált és weben szolgáltatott dokumentumaink többféle formátumban elérhetővé váljanak számukra. Ennek hatékony, gyors és időtálló megvalósítása azonban több okból sem egyszerű feladat, ám a Cocoon kétségkívül az egyik leghatékonyabb út. Mi sem bizonyítja ezt jobban, mint azon képességei, amelyek akár a „profi" fejlesztőket is hamar meggyőzhetik. Hogy miről is van szó?

- Adatok különböző formátumú rugalmas publikálása.
- Különböző adatforrások integrációja.
- Tartalmak személyre szabhatósága.
- Alkalmazások integrációja.
- Platformfüggetlenség.
- Rugalmas architektúra.
- Nyílt forráskód és INGYENESSÉG.

Úgy vélem, nyilvánvaló, hogy mindegyik szempont kiemelkedően fontos, de a korántsem túlfinanszírozottságukról híres könyvtárak, kulturális intézmények számára egyértelműen a legutóbbinak van a legnagyobb jelentősége. Mondom ezt egy digitális könyvtár munkatársaként, ahol rövid és hosszú távon egyaránt milliós összegeket spóroltunk meg azzal, hogy új honlapunkat a W3C ajánlásaira és az Apache Project Cocoon termékére alapoztuk.

Be kell látnia mindenkinek, hogy webes rendszereket építeni egyre nehezebb lesz, mivel többfajta eszközzel és környezetből érik el őket a felhasználók. Megfelelő technológia nélkül már ma is nehéz kezelni a dolog bonyolultságát, és a helyzet egyre összetettebb lesz. Ideje elkezdeni tanulni – és miért ne kezdhetnénk a Cocoonnal?!

Jegyzetek

[1] A CSS olyan stíluslap-megvalósítás, amely lehetővé teszi, hogy a HTML, XHTML, XML állományok tartalmához egyedi stílust rendelhessünk hozzá. W3C ajánlás, amelynek jelenleg két érvényben lévő kiadása létezik. A lépcsős stíluslapok használata egyre népszerűbb a programozók körében – a technológiát a tv és mobil eszközök használatához is folyamatosan fejlesztik. További információ: *http://www.w3.org/TR/REC-CSS1-961217.html, http://www.w3.org/TR/CSS21/* és *http://www.w3.org/Style/CSS/current-work*

[2] Az XSLT valójában, noha stíluslap, sokkal inkább tekinthető egyfajta sajátos programozási módszernek, nyelvnek, amely beépített vezérlési struktúrákat, nyelvi elemeket, paramétereket, változókat tartalmaz, csak éppen az XML szabályainak megfelelően, XML dokumentumként leírva – szintén W3C ajánlás. További információ: *http://www.w3.org/TR/xslt* és *http://www.w3.org/TR/xslt20/*

[3] The Apache Cocoon Project. *http://cocoon.apache.org*

[4] A Popoon rendszert a Cocoon mintájára valósították meg, azonban teljesen PHP alapon. Kapcsolódik hozzá még a Bitflux Editor, amely egy online XML szerkesztő, illetve egy teljes CMS rendszer is, a Bitflux CMS. Ez a megoldás is szabad forrású. További információ: *http://popoon.org/*

[5] 2001-re már 2000 ember iratkozott fel a Cocoon levelezőlistájára, és számos nagy cég is támogatta a szoftver fejlesztését.

[6] Az XSLT kód alapján W3C konform állomány jön létre, hiszen DOCTYPE is szerepel a forrásban.

[7] Really Simple Syndication vagy Rich Site Summary – XML alapú szolgáltatás, website-on megjelent aktuális hírek vagy cikkek továbbítására (cím, szerző, kategória, rövid tartalom és ugrópont). További információ: *http://rss.lap.hu*

[8] Text Encoding Initiative – fontos nemzetközi projekt, amelynek feladata irányvonalak kifejlesztése, terjesztése a géppel olvasható szövegek kódolására, közvetíthetőségére és cserélhetőségére, valamint javaslatok tétele új szövegek kódolására. További információ: *http://www.tei-c.org/*

[9] Lightweight Directory Access Protocog – különböző platformokon megvalósított címtárak, adattárak szabványos közös kezelője.

[10] 2005 szeptemberében a Neumann-ház is elindította saját RSS szolgáltatását.

Irodalom

BATES, Chris: XML: elmélet és gyakorlat. Bp.: Panem Kiadó, 2004. 542 p. ISBN 963 545 393 0

BÍRÓ Szabolcs: Szövegfeldolgozás XML alapokon. Bp.: Neumann Kht., 2005. 208 p. ISBN 963 218 740 7

BRADLEY, Neil: Az XML-kézikönyv. Bp.: Szak Kiadó, 2005. 758 p.

LANGHAM, Matthew–ZIEGELER, Carsten: Cocoon: Building XML Applications. New Riders Publishing, 2002. 504 p. ISBN 0-7357-1235-2

SZALAI Attila–NÉMETH Krisztián–BÍRÓ Szabolcs: A Neumann-ház honlapjának továbbfejlesztése: rendszerterv. Bp.: Neumann Kht., 2005. 36 p. (belső anyag)

SZALAI Attila: Jeles napok: rendszerterv. Bp.: Neumann Kht., 2005. p. 15–16. (belső anyag)

Beérkezett: 2005. október 17-én.

Bíró Szabolcs
a Neumann János Digitális Könyvtár és Multimédia Központ Kht. könyvtár-informatikai osztályán osztályvezető-helyettes.
A Berzsenyei Dániel Főiskola könyvtár- és információtudományi tanszékén óraadó tanár.
E-mail: *biro.szabolcs@neumann-haz.hu*