Cocoon alapú webtartalom generálás XML és XSL felhasználásával

BÍRÓ SZABOLCS

Neumann János Digitális Könyvtár és Multimédia Központ Kht. (Neumann-ház)
Könyvtár- Informatikai Osztály
biro.szabolcs@neumann-haz.hu

1. Bevezetés

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 feldolgozás elvégzésére 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 Project alábbiakban tárgyalt Cocoon terméke, kísérlet ilyen szerver előállítására.

2. Hangsúlyeltolódás a webes tartalomleírásban

2.1 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 tartalom-szolgáltatás módja, ám ma már korántsem ilyen egyszerű a helyzet. Ennek legfőbb oka, hogy sok olyan formátumot használunk, ahol a stílus és maga 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, magát az

információt pedig, amivel a lapot kitöltik, ~~pedig~~ adatbázisból veszik. A megoldás hátránya, hogy egységes szabvány a területen mindmáig nem létezik 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 kettő 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.

2.2. A megoldáskeresés útjai

2.2.1. 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 – továbbiakban W3C – volt. Első, meglehetősen hatékonynak bizonyuló és egyre gyakrabban használt megoldási kísérletük a Cascading Style Sheets[1] (CSS) specifikáció volt, mely a HTML nyelvhez készült.

Segítségével bizonyos fokig már szétválasztható a tartalom a 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?

2.2.2. Bővíthető Jelölő Nyelv – 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 egy szabványos jelölő nyelv dokumentumok belső szerkezetének leírására, beleértve az egyes elemeket jelölő címkék (tag-ek) definiálásának módját is. Segítségével elvben bármilyen dokumentum leírható,

[1] A CSS olyan stíluslap megvalósítás, amely lehetővé teszi, hogy a HTML, XHTML, XML állományok tartalmához egyedi stílust rendelhessünk hozzá. W3C ajánlás, melynek jelenleg két érvényben lévő kiadása létezik. A lépcsős stíluslapok használata egyre népszerűbb a programozók körében – a technológiát TV és mobil eszközök használatához is folyamatosan fejlesztik.
További információk: http://www.w3.org/TR/REC-CSS1-961217.html és http://www.w3.org/TR/REC-CSS2/

függetlenül az azt tároló és megjelenítő számítógépes környezettől, nagyfokú bonyolultsága és kevésbé költséghatékony alkalmazhatósága miatt azonban a várt mértékben nem terjedt el sem Európában, sem világszerte. 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?

2.2.3. Bővíthető Stíluslap Nyelv (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”.

Azonban ennél jóval többre van szükség, olyan stíluslapra, amely transzformációra is képes. Ennek fényében született meg az eXtensible Stylesheet Language (XSL) specifikáció, ami XML dokumentumok stílusleírására való ismert nyomdai leírónyelvek képességeit adta az XML-nek. Végül az egészet az eXtensible Stylesheet Language Transformations[2] (XSLT) 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. –, ami böngésző-, vagy más programokkal szolgáltatható.

2.3. 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

[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 információk: http://www.w3.org/TR/xslt

formázott dokumentumokat – a fentebb már említett –~~,~~ ismert 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…

3. Az Apache Cocoon

A koronát a Cocoon – PHP-s változata a Popoon[3] – projekt tette fel az eddigi munkára, mert az elkészített szoftverelemeket egységes keretrendszerbe foglalta, lehetővé téve, hogy teljes honlapok épüljenek XML-XSL(T) transzformációkra. Az ilyen weboldalak pedig elképzelhetetlen 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 „primit~~í~~ivizmusai és korlátai” távol kerülnek tőlük.

Az olyan megjelenítési formátumok, mint pl. a(z) (X)HTML, PDF stb. más aspektusba kerülnek, mert XSL stíluslapokkal ugyanazt az XML dokumentumot egészen könnyedén teljesen más formátummá lehet alakítani.

Bizonyos szempontból még a böngészőprogramok versenye is jelentéktelenné válik, hiszen a megjelenítésért felelős a kliens oldalon teljesen elegendő, a publikáló keretrendszer – Cocoon – veszi át a formázással és a stílus kezelésével kapcsolatos teendőket.

Végül, de nem utolsósorban a technológia szabványokra épül, következésképp nem kell a gyártófüggő formátumokat megtanulni.

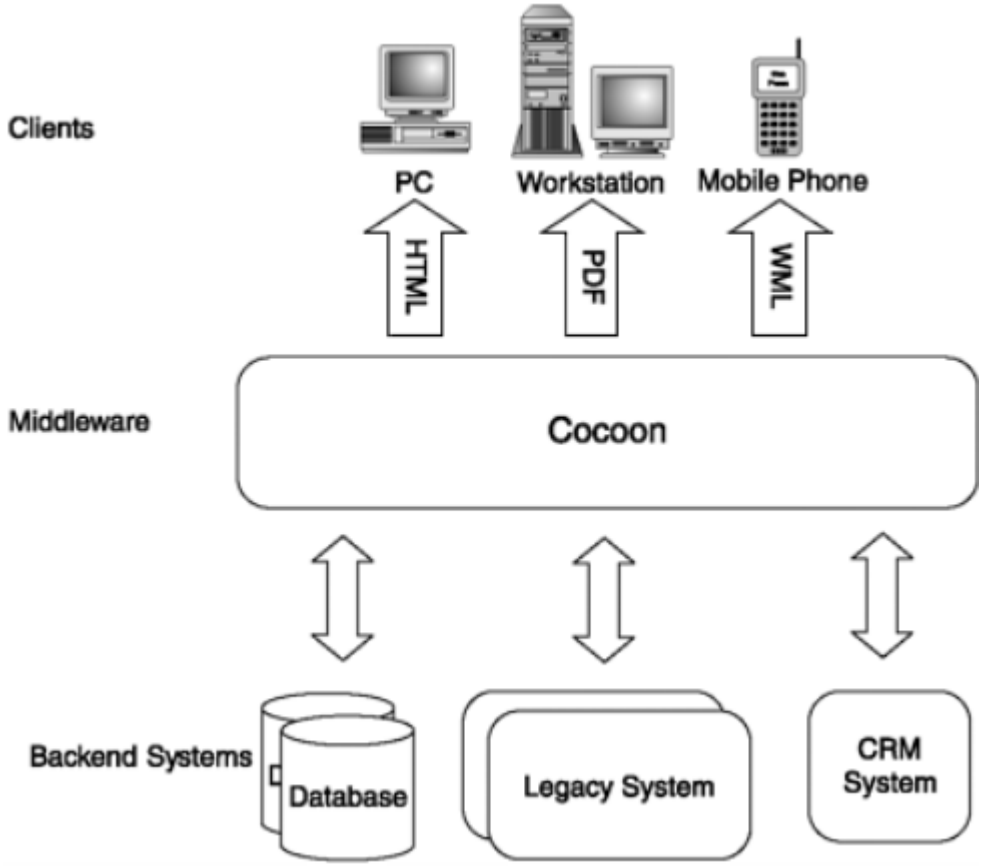
3.1. Középpontban a Cocoon

Ez után a meglehetősen hosszúra nyúlt, ám korántsem felesleges felvezető 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. Legfontosabb tulajdonsága, hogy képes együttműködni a meglévő J2EE megoldásokkal, HTML, WML, PDF, SVG, RTF kimenetet tud produkálni és nem utolsó sorban teljesen ingyenes technológiákra épül. Az pedig már csak hab a tortán, hogy a Cocoon-ra egy Lenya nevű CMS rendszert is felépítettek, beépített keresőként pedig a Lucene-t használja.

Az architektúra logikai elhelyezkedésének megértését elősegítendő, az alábbiakban egy olyan ábra látható, amely azt illusztrálja, hogy a rendszer miként fér hozzá az adatokhoz eltérő rendszerekből, majd hogyan „mossa össze” őket, a különböző formátumokban történő kliens oldali szolgáltatás céljából.

---

[3] A Popoon rendszer~~t~~ a Cocoon mintájára ~~lett meg~~valósít~~va~~ották meg, azonban teljesen PHP alapon. Kapcsolódik hozzá még a Bitflux Editor, mely egy online XML szerkesztő, illetve egy teljes CMS rendszer is, a~~z~~ Bitflux CMS. Ez a megoldás is szabad forrású.
További információk: http://popoon.org/



1. ábra: A Cocoon alapú, „hármas-kötésű” architektúra.

3.2. A rendszer működ(tet)ésének logikája

A Cocoon rendszerrel a feldolgozandó XML dokumentumokat, az azokon 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.

Minden Cocoon-ra 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 annak megjelenéséért feleljen.

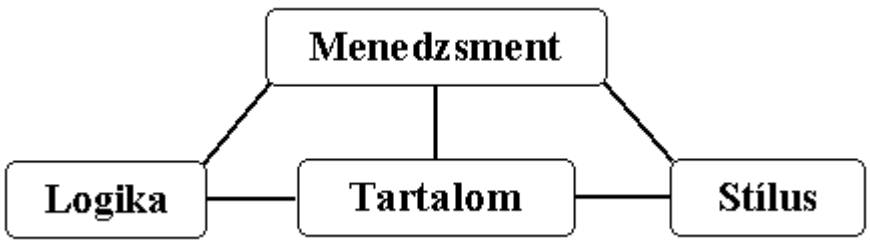
A „hagyományos” web fejlesztő eszközök – PHP, ASP – alkalmazásakor a megjelenítés (arculat) és a működési logika szorosan összefonódik – elsősorban a nyelvi kódok keveredése miatt. Ezáltal az alkalmazás karbantartása, a hibák javítása nehezebb és költségesebb. Az arculat megváltoztatása gyakorlatilag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

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 a következőképpen néz ki a munkamegosztás – persze kellő indokkal összevonások alkalmazhatók:

Fejlesztő	Rendszergazda	Tervező	Szerkesztő
A fejlesztők valósítják meg a működési logikát és a webes alkalmazás mögött húzódó objektum modellt. Nem a weboldalak kinézetével, és nem is a képernyőn olvasható szöveggel van dolguk, hanem az alkalmazás funkcionalitásával.	A rendszergazda a felelős a web szerver által kiszolgált URI tér kiosztásáért: az URL-nek a megfelelő feldolgozási folyamat-definícióval való megfeleltetéséért. Dedikált rendszergazda hiányában egy fejlesztőnek kell ezt a szerepkört ellátnia.	A tervezők felelősek a honlap végső kinézetének 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**

1. táblázat: Feladatcsoportok szétválasztása Cocoon esetén.

A táblázatban megnevezett szerepkörök kapcsolatát az alábbi ábra szemlélteti:



2. ábra: A feladatcsoportok közti szerepkörök kapcsolata a Cocoon-ban.

3.3. Cocoon a Neumann-házban

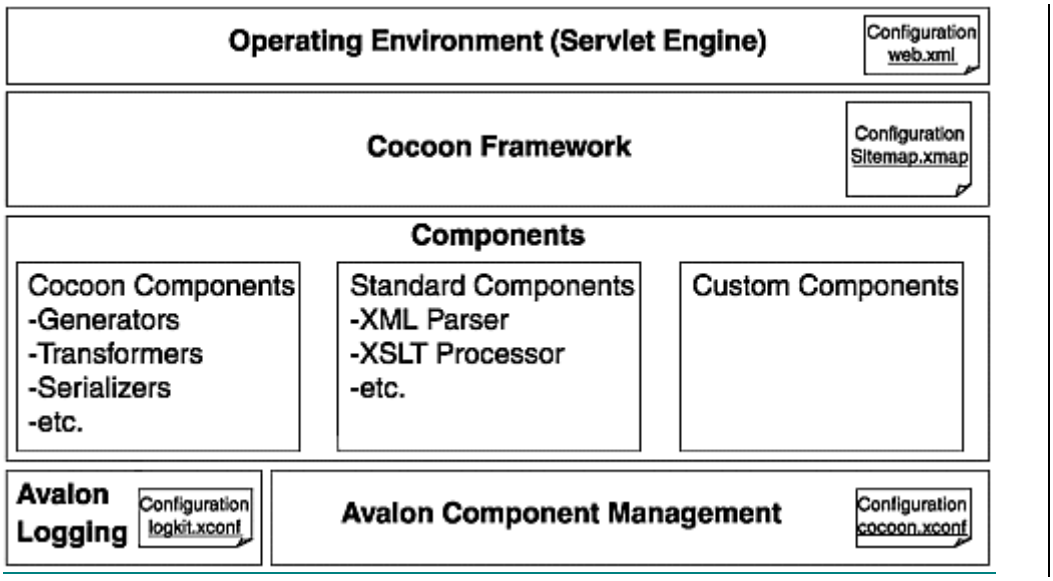
A digitális könyvtár új honlapjának tervezése közben már előre körvonalazódott, hogy a hagyományos módszerekkel hatalmas munkát igényel a site elkészítése.

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 7 éve SGML/XML és XSL(T) 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 kliens eszköz 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 az alábbiak voltak:



3. ábra: Az új Neumann-ház honlap készítése során alkalmazott Cocoon komponensek.

Az első látásra bonyolultnak tűnő ábra, valójában nagyon könnyen megfejthető. A felső szinten maga 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 magának a rendszernek a működését definiálja. Ezeket pedig az egyes komponensek – Cocoon (az 1. táblázatban látható feladatcsoportok szétválasztása is részben ezek alapján történt), általános és saját fejlesztésű – követik. Mindegyikkel nincs lehetőség e

rövid „tanulmány” keretében foglalkozni, de néhány egyszerű forráskóddal támogatott példán keresztül viszont érdemes egy pillantást vetni a keretrendszer működésére.

3.3.1. 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 globális struktúrát követ, amely a következőképpen néz ki:

```
<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, konkrét példát, amellyel a legtöbb programozási nyelvkönyv indítani szokott. A „Hello World” szöveget tartalmazó XML fájlból a Cocoon segítségével írassuk ki egy kliens oldali böngészőbe az említett üzenetet.

Az XML fájl felépítése a következő:

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

2. forráskód: A helloworld.xml állomány tartalma.

Ahhoz, hogy ebből HTML kimenet~~et~~ legyen, készíteni kell egy stylesheet-et, melynek segítségével a Cocoon elvégezheti a transzformációt.[4] Íme:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="dokument">
  <html>
    <body>
      <h1><xsl:value-of select="szoveg"/></h1>
    </body>
  </html>
</xsl:template>
```

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

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

3. forráskód: A helloworld2html.xsl állomány tartalma.

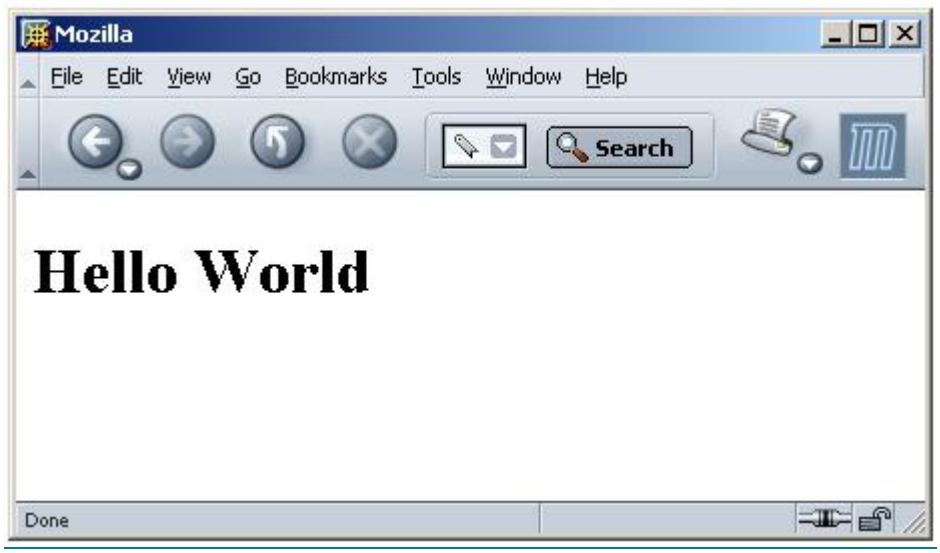
A kód nem tesz mást, mint sablont definiál a `dokument` elemre, melynek tartalmát HTML vázba helyezi. Az XML állomány `szoveg` elemének tartalma <h1> elemek között kerül kiíratásra a böngészőben.

Végül nincs más hátra, mint a sitemap-ben beállítani, hogy mi is történjen:

```
<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 Cocoon-nak, 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. A böngészőben látható kimenet a várt eredményt hozta:



4. ábra: A HTML kimenet Mozilla 1.7.5-ben.

A kicsit hozzáértőbbek rögtön mondhatják, hogy ezt a végeredményt számos más úton, ugyanígy el lehetett volna érni és még sitemap sem kell hozzá. Részben igazuk van, ám egy komplex, webportál szerkezethez hasonló felépítésű honlap esetében – amilyen az 5. ábrán látható új Neuman-ház site nyitólapján szerepel –, mindez már nem jelenthető ki ilyen egyértelműen.

5. ábra: Neumann-ház nyitólap szerkezeti felépítése IE 6.0-ban.

A képen kitűnően látszik, hogy a kezdőlap 3 hasábos szerkezetű, melyek mindegyikének tartalma egy-egy XML állományban ~~lett~~ van eltárolva – ezek HTML transzformációját XSL(T) 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[5] szabvány XML Schemá-ját vettük alapul a tartalmak kódolásakor. Az oldal tetején elhelyezkedő fejrész és „lábléc” tartalma XSL(T)-ben tárolódik és abból alakul át HTML-lé, a létrejött oldalszeleteket pedig a Cocoon cinclude komponense olvasztja egységes, böngészőben megjeleníthető kimeneti állománnyá.

Külön említésre méltó, hogy a site aloldalainak XML kódolása során a TEI[6] P4 XML alapú DTD-jét hívtuk segítségül – ennek eredményeként tartalmaink nemcsak XML alapúak, hanem nemzetközileg is elismert, egységes

[5] Rich Site Summary – XML alapú szolgáltatás, website-on megjelent aktuális hírek, vagy cikkek továbbítására (cím, szerző, kategória, rövid tartalom és link).
További információk: http://rss.lap.hu

[6] Text Encoding Initiative – Fontos nemzetközi projekt, melynek feladata irányvonalak kifejlesztése, terjesztése a géppel olvasható szövegek kódolására, közvetíthetőségére, és cserélhetőségére, valamint javaslatok tétele új szövegek kódolására.
További információk: http://www.tei-c.org/

jelölésrendszert követnek. A TEI alapú jelölés során alkalmazott fejlécet pedig – amely egyébként a feldolgozott dokumentum leírására szolgál – ~~pedig~~ 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.

6. ábra: Neumann-ház nyitólap – vakok és gyengén látók számára készített felület szerkezeti felépítése Mozilla 1.7.5-ben.

4. Továbbfejlesztési lehetőségek

Az XML technológia és a Cocoon bevezetésével az előbbiekben vázolt projekt célkitűzésein túlmenően, számos új szolgáltatásra nyílik technikai lehetőség. Ebben a fejezetben rövid kitekintést adunk az új lehetőségekről, amelyekkel a jövőben bővíteni lehet a Neumann-ház honlapját.

4.1. Mobil szolgáltatás

A kliens oldali böngésző (user agent) képességeinek lekérdezésével a Cocoon lehetőséget ad az XML formában tárolt tartalmak~~,~~ 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.

4.2. Regisztrációhoz kötött szolgáltatások

A Cocoon autentikációs (ez egy szakkifejezés? Mert ilyen magyar szó nincs!!!) komponense lehetővé teszi, hogy csak a megfelelő jogosultsággal bejelentkező, regisztrált felhasználók férhessenek hozzá a védett információkhoz. A regisztráció lehet ingyenes, de történhet díjfizetési kötelezettség mellett is. A felhasználói azonosító adatokat a Cocoon fájlból, adatbázisból, vagy számos egyéb adatforrásból, pl. a LDAP[7] kiszolgálótól lekérdezve ellenőrzi. A sikeres bejelentkezést követően a Cocoon session-kezelő komponense biztosítja, hogy az azonosított felhasználók mindegyike kizárólag a saját tranzakcióival kapcsolatos kommunikációt végezhesse.

4.3. Szindikálás

A Cocon 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.

4.4. 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, amit más szolgáltatók felhasználhatnak, publikálhatnak.

4.5. Portál szolgáltatások

A Cocoon beépített portál komponenssel rendelkezik, amely integrálja és kibővíti a többi Cocoon funkciót (autentikáció, szindikálás, stb.). A portálra bejelentkezett felhasználók a saját 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.

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

5. Konklúzió

A Cocoon és a hozzá kapcsolódó technológiák áttekintse után felmerülhet a kérdés: Miért jó mindez a tartalomszolgáltatók, könyvtárak számára? Bonyolultnak tűnik – bár csak első látásra –, 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, melyekkel kifogástalan munkát lehet végezni.

Napjaink könyvtárainak – ha úgy tetszik információs központjainak – is meg kell~~, hogy feleljenek~~ felelniük a XXI. 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 böngésző programokat alkalmazó és futtató asztali illetve hordozható számítógépek, okos telefonok és PDA-k, nem utolsó sorban 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 azo~~k~~n képességei, melyek 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 kiemelkedő fontosságú, de a korántsem túlfinanszírozottságukról híres könyvtárak, kulturális intézmények számára egyértelműen a legutóbbi bírhat kiemelkedő hangsúllyal. Mondom ezt egy digitális könyvtár munkatársaként, ahol rövid és hosszú távon egyaránt milliós nagyságrendű 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 ~~kézben tartani~~kezelni a dolog bonyolultságát és a helyzet egyre csak összetettebb lesz. Ideje elkezdeni tanulni… – és miért ne kezdhetnénk a Cocoon-nal?!

Felhasznált irodalom

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

BÍRÓ Szabolcs: *A szövegfeldolgozás modern eszközei – az SGML és XML nyelvek.* In.: TMT, 51. köt. 10. sz. 2004. p. 453-459.

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)

The Apache Cocoon Project. http://cocoon.apache.org

World Wide Web Consortium. http://www.w3.org