NAGYTÖMEGŰ, STRUKTURÁLT SZÖVEGEK ONLINE SZOLGÁLTATÁSA

Király Péter pkiraly@tesujionline.com
Tesuji Magyarország Kft. (http://www.tesujionline.com)

Mi a cél? Legyen egy … rendkívűl gyors szemantikus keresőmotor, digitális tartalommegjelenítő rendszer nagy méretű rendezett vagy rendezetlen archívumok számára, testreszabható indexelőmotor, melyben TEI dtd (próza, vers, dráma, szótár stb), HTML, PDF, MS Word és Excel dokumentumok indexelésére van lehetőség (és további a dokumentumtípusok lehetőleg egyszerű felvételére), ami Open Source program, amely a Jakarta Lucene indexelő motorra és a Tomcat servlet containerre épül, kizárólagosan Javaban írt webes alkalmazás, szerver oldalon bármilyen Javát támogató operációs rendszerrel kompatibilis, míg kliens oldalon minden elterjedt böngészőn fut.

Mit jelent a nagy adattömeg? *Arcanum Adatbázis Kft.* szövegtára: 7 GB szöveg, 1,5 millió HTML oldal (leváltott program: NXT3). *Project Gutenberg Consortia Center* szövegtára: 8 GB szöveg, 100 000 dokumentum (egy könyv ~ 1 HTML lap) (leváltott program: HtDig).
Meglévő open source összetevők, amikre lehet építeni: Struts (MVC felépítés, Lucene (indexelés), Ant (fordítás, telepítés), Tomcat (más szervlet konténerekkel is), Jakarta commons, Junit, Eclipse, MyEclipse (IDE), CVS (verziókövetés), Bugzilla (hibajelentés), Wiki, OpenOffice (dokumentáció).
A működés vázlata: az adminisztrációs felületen „élőben” vagy időzítő segítségével beállíthatjuk az indexelendő adatforrásokat, majd ennek végeztével az Anacleto automatikusan kezeli a internet vagy intranet felé történő szolgáltatásokat. Az adatforrás helye lehet helyi fájlrendszer, webszerver, mailszerver, adatbázis, webdav, ftp stb.
Az indexelés a különböző dokumentumtípusokat különféleképpen kezeli. A HTML/XML dokumentumok egy XSLT szűrőn mennek keresztül, a „plain text” dokumentumok vagy natív módon kerülnek be az adattárban vagy egy parseren keresztül (pl. katalóguscédulák), a pdf, word, rtf, mailbox állományok natív módon, csakúgy mint az adatbázisok (ez utóbbiak JDBC-n keresztül). A Quark, InDesign, PDF esetében manuális vagy programozott konverzióval XML/HTML fájlt kell előállítani. Lehetőség van képek metaadatainak (pl. XMP) indexelésére, ehhez először ezeket kell kinyerni XML vagy plain text formátumban, majd az ott ismertetett módszerrel kezelni. Az Arcanum adatbázisai esetében egy különleges, XML-szerű formátummal dolgoztunk, a Folio Flat File-lal.

```
<RD:"Könyv"><JD:"01"><PN:"könyv">Das erste Buch Mose (Genesis)</PN>
<RD:Fejezet><PN:"fejezetszám">1. Mose 1</PN>
<RD:Vers><JD:"01:1.1"><JD:"1. Mose 1.1"><PN:"versszám">1. Mose 1.1</PN>
<RD><PN:"SZÖVEG">Am Anfang schuf Gott Himmel und Erde. *</PN>
```

ahol *<RD:"Könyv">* 'szint' ami megfelel a HTML H1....H6 elemeknek (jelöletlenül a 'p'-nek), *<PN:"SZÖVEG">* mező, szemantikus jelölőelem (a HTML-ben span class=”...”-nak fordítjuk) *<JD:"01:1.1">* kereshető ugrópont, HTML: <a name=”...”> vagy újabban id=”...”

HTML konverzió:

```
<H1 class="Konyv"><a name="JD_01"></a><span class="id" type="field"
title="id">01</span><span class="biblebook" type="field" title="biblebook">
Das erste Buch Mose (Genesis)</span></H1>
<H2 class="Fejezet"><span class="chapter" type="field" title="chapter">1.
Mose 1</span></H2>
<H3 class="Vers"><a name="JD_01_1_1"></a><span class="id" type="field"
title="id">01:1.1</span><a name="JD_1__Mose_1_1"></a><span class="verse"
type="field" title="verse">1. Mose 1.1</span></H3>
<p class="rd"><span class="text" type="field" title="text">Am Anfang schuf
Gott Himmel und Erde. *</span></p>
```

Az általunk használt span elem-beli attribútumok speciális jelentése: *class="text"* a class neve 'text', erre lehet CSS stílust illeszteni és ez lesz a mező neve, *type="field"* típusa field, vagyis ez egy mező, amit le kell indexelni, *title="text"* a title minden HTML tagben opcionális attribútum, a gyorstippnek, hogy a felhasználó mindig tudhassa, milyen mező az, amit néz.

A HTML indexelése XSL-lel:

```
<xsl:template match="span[@type='field']">
  <xsl:element name="in:index" use-attribute-sets="text">
    <xsl:attribute name="field" ><xsl:value-of select="@class"
/></xsl:attribute>
    <xsl:apply-templates/>
  </xsl:element>
</xsl:template>
<xsl:template match='H1|H2|H3|H4|H5|H6'>
  <xsl:element name="in:index" use-attribute-sets="text">
    <xsl:attribute name="field">content</xsl:attribute>
    <xsl:element name="in:index" use-attribute-sets="keyword">
      <xsl:attribute name="field">title</xsl:attribute>
      <xsl:apply-templates/>
    </xsl:element>
  </xsl:element>
</xsl:template>
```

ahol *span[@type='field']* ha a span type attribútuma 'field', *in:index* saját xslt kiterjesztés, *use-attribute-sets="text"* text típusú indexelési attribútumok beállítása, *name="field" ... select="@class"* a mező nevét a class attribútumból kell kinyerni, *match='H1...'* a H1...Hn tagek címek, amelyekre más szabályok vonatkoznak, *use-attribute-sets="keyword"* keyword típusú indexelési beállítások, *name="field">content* Le kell indexelni úgy is, hogy a 'bárhol' keresésre is erdményt adjon (content mező, amiben minden szöveg benne van), *<xsl:apply-templates/>* rekurzív indexelés (az XSLT erőssége)

Kép indexelése: XMP adat a képi állományban

```
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="3.1.1-111">
   <dc:description><rdf:Alt><rdf:li xml:lang="x-default">OSZK Quart. Lat.
16.</rdf:li></rdf:Alt> </dc:description>
   <dc:creator><rdf:Seq><rdf:li>Latsny Adam</rdf:li></rdf:Seq></dc:creator>
   <dc:title><rdf:Alt><rdf:li xml:lang="x-default">Catalogus librorum,
dissertationum … 1755.</rdf:li></rdf:Alt></dc:title>
```

Kinyerés és feldolgozás után ezt kapjuk: description: „OSZK Quart. Lat. 16.”, creator: „Latsny Adam”, title: „Catalogus librorum, dissertationum … 1755.”

Nagy kérdés - a könyvtár kialakításakor, hogy legyenek-e mezők? A döntésnek számos következménye van, a legfontosabbak:

	mező nélkül	mezővel
konverzió	Félig vagy teljesen automatizálható	bonyolult, félautomatikus eljárások, emberi tudást *vagy* nagyon különleges szoftvereket igényel
	egyszerűbb módszer	Bonyolultabb módszer
keresés	teljes szövegű keresés	teljes szövegű keresés
		Mező szerinti keresés (cím, szerző, stb.)
	"részecske"	"részecske" a címben "molekula" a képaláírásban "MIT" az összefoglalóban
találatok	kisebb relevancia	nagyfokú relevancia
	a szöveg bármely részén (a szövegnek nincsenek kitüntettett fontosságú részei)	a szöveg meghatározott részén

A karakterkészletek kérdése két részre osztható. *Belső adatábrázolás*: a Java mindent UTF-ben tárol, a külső forrásból érkező adatokat az esetek többségében automatikusan konvertálja. Különleges esetekre (pl. az OszK katalógusa 'ANSEL' nevű, az Unicode-hoz hasonló, de attól számos helyen eltérő készletet használ) bőségesen léteznek konverterek. *Webfelület*: UTF-8. A bemenő adatokat az Anacleto ellenőrzi és adott esetben konvertálja.

I18n — „nemzetköziesítés". A Java alaptulajdonsága a 'Locale' (helyi beállítások) figyelembevétele. Vannak lokalizált (standard név=lokalizált érték párokból álló) tulajdonság-fájlok, amiket a rendszer a böngésző nyelvi beállítása alapján választ ki. Jelenleg az Anacletoban angol, magyar és olasz lokalizáció érhető el. A felületen csak a kulcsot kell megadni, az értéket a program maga fogja kitölteni, például:

```
<li><bean:message key="welcome.intro.2" /></li>
<li><bean:message key="welcome.intro.3" /></li>
<li><bean:message key="welcome.intro.4" /></li>
```

Az Anacletoban a dokumentumok általában hierarchiába vannak szervezve, melynek elemei: polcok, könyvek, dokumentumok, dokumentumok ('ad infinitum'). Minden elemhez tartozhat saját indexséma, dokumentumtípus-meghatározás (ami alapján az Anacleto kezelni fogja), saját stíluslap, polcok és könyvek esetében címoldal. A dokumentum lehet virtuális is, pl. egy XML fájlt virtuálisan feloszthatunk több apró részre, ami azt jelenti, hogy a tartalomjegyzékben, találati listán, dokumentum-nézetben külön-külön látjuk a fizikailag egyazon fájlban lévő részeket. Eredetileg nem hierarchikus dokumentumokból (pl. egy adatbázis tábla) is lehet hierarchikusat csinálni (pl. valamilyen elven való csoportosítással, kezdőbetűkkel stb.), de ez nem kötelező.

A keresés az Anacleto legfőbb erőssége. A legtöbb keresési lehetőség a Lucene alaptulajdonsága, de – hála az Open Sourcenak – mindezekhez több kiegészítést fejlesztettünk. *Egyszerű keresés* az archívumban szereplő összes kifejezésre kereshetünk, *mező szerinti keresés*, az indexelés során mezőket lehet meghatározni, amelyeket külön és más keresésekkel kombinálva is lehet keresni (*szerző: Iván és tartalom: Duna*), *pontos kifejezés* („Kossuth Lajos"), *hasonlóságon alapuló keresés (Kossuth, Kosuth stb.), közelségi keresés* (Kossuth és Lajos szavak 5 szó távolságra, bármely sorrendben), *Google szintaktika alkalmazása* a keresőkérdések egymáshoz való viszonyának leírásához ('és', 'vagy', 'nem'), *a keresés hatókörének szűkítése* az archívum egy megadott részére *(„Kossuth a 2000-es*

évfolyamban"), ékezetek helyettesítésen alapuló indexelése szabadon beállítható a forrásdokumentumban helyesen 'Košice' amit a 'Kosice' keresőkérdés is megtalál, *szóelemzéses keresés és ontológia (tezaurusz)* beépíthető ('alma' megtalálja az 'almát', 'almának' alakokat, az 'úszásnem' a 'gyorsúszás', 'pillangó' alakokat)

A felhasználói felületen számos eszköz segíti a dokumentumok használatát. *Tartalomjegyzék*: fő feladata a hierarchia böngészése, előállítása dinamikus, oda vissza szinkonizálható a dokumentummal. *Keresésre szűkített tartalomjegyzék*: csak azok az ágak szerepelnek, amikre van találat, benne a találat kontextusa (KWIC), ahol a keresőkérdés ki van világítva. *Navigációs linkek*: előző/következő dokumentum, előző/következő dokumentum a találati listán, előző/következő találat a dokumentumon belül, vissza a találati listára, találat sorszámának kijelzése (pl. 62/383), a tartalomjegyzék szinkronizálása, nyomtatás, könyvjelző. *Teljes eléséri útvonal kijelzése*: a felhasználó minden pillanatban tudja, hogy hol jár, nincsenek „meglepetésszerű", sehová sem köthető oldalak (pl. Arcanum szövegtár > lexikonok, szótárak > Finály Henrik: A latin nyelv szótára > M > Menosca, ae, nn., ahol a csomópontok linkként szerepelnek). *Találati lista*: keresés szűkítése, finomítása, előző/következő találati lista, lista elemei számosságának módosítása, logaritmikus lapozás (1- 31- 41- 51- 61- 71- 81- 91- 161- 261- 361- 381-), találatok szókörnyezetének megjelenítése, a keresett kifejezés kivilágítása, találatokra szűkített tartalomjegyzék behívása.
A rendszer része egy adminisztrációs felület, amiről szabályozni lehet a következőket: alapbeállítások (könyvtárak, fájlok, rendezés, naplózási szintek stb.), indexelés, index karbantartás, index megtekintése, naplók megtekintése, felhasználók és jogok beállítása, IP-filter, ékezet-konverziók (Kosice eredetileg Košice, most mind a két alak kereshető), stopszavak.

Lehetőség van a felhasználók hozzáférését különféle szempontok alapján szűkíteni, pl. funkcionálisan (hozzáférhet a tartalomjegyzékhez, találati listához, de a dokumentumhoz nem), tematikusan (egy időtartományon belül az adott kiadvány minden cikkéhez), időkorlátosan, számosság alapján (10 dokumentumhoz).

Az Anacleto szoftver-követelményei: *böngésző*: IE5.0+, Firefox, Opera 7.5+, Safari (lehet persze jóval egyszerűbb felületet is építeni, kihagyva a kliens oldali szkriptnyelvek nyújtotta lehetőségeket), *Java alkalmazásszerver*: bármelyik (JBoss-szal es Tomcat 5.0, 5.5-el teszteltük), *operációs rendszer*: MS Windows, NT, Linux, Solaris (minden olyan op. rendszer, amelyhez létezik Java támogatás).