

# Hasznos dolgok

Utolsó módosítás: 2003.07.20.

Dénes Pál (denespal@nir.hu)

Ez egy kis gyűjtemény néhány általam hasznosnak vélt dologról: jegyzetek, leírások, problémák és megoldások, scriptek, satöbbi...

## LyX és DocBook magyarul

Ez egy rövid leírás arról, hogy a LyX (<http://www.lyx.org>) szövegszerkesztő és a DocBook (<http://www.docbook.org>) dokumentumosztály használatával hogyan lehet hatékonyan egyszerűbb dokumentumokat létrehozni. Az a cél, hogy magyar nyelvű, szép, hordozható, többféle (pl. HTML és nyomtatható) formátumra konvertálható anyagokat tudjunk készíteni.

A DocBook nagyon jó eszköz dokumentációk írására (és még valószínűleg sok más célra is). Mivel az SGML/XML egyszerű szöveges formátum, tetszőleges szövegszerkesztővel készíthetünk DocBook dokumentumokat. Egyelőre azonban nem találtam olyan programot, ami jól használható és tökéletesen támogatja a DocBook-ot. Eddig legjobban a LyX vált be. A DocBook elemeiből ugyan csak a legfontosabbakat ismeri, de ezek is elegendőnek bizonyultak. (Ez a dokumentum is LyX-szel, DocBook-ként készült. Megtekinthető az eredeti LyX file (hasznos.lyx), és a PDF változat is (hasznos.pdf).) Itt van egy teszt file, ahol igyekeztem kipróbálni minden lehetőséget: Magyar nyelvű DocBook article teszt file (docbooktest/index.html) [PDF] (docbooktest/docbooktest\_article\_magyar.pdf) [LyX] (docbooktest/docbooktest\_article\_magyar.lyx) (Érdeemes összehasonlítani a LyX file-t a HTML és PDF változattal, hogy lássuk mi hogy működik, és mi nem.)

(És még egy korábbi, hasonló próbálkozás részletes leírása: A LyX és a LinuxDoc használata (linuxdoc/linuxdoc.html) [PDF] (linuxdoc/linuxdoc.pdf))

## Hozzávalók

RedHat 7.2-n kb. ezekre a csomagokra van szükség ahhoz, hogy minden működjön:

```
lyx-1.1.6fix3-1 vagy lyx-1.2.0-1 (a különbséget lásd később!)
xforms-0.88-9 vagy xforms-0.89-1 (a LyX verziótól függően)
ghostscript-6.51-16
ghostscript-fonts-5.50-3
tetex-1.0.7-30
tetex-latex-1.0.7-30
tetex-fonts-1.0.7-30
tetex-dvips-1.0.7-30
docbook-dtd30-sgml-1.0-10
docbook-dtd31-sgml-1.0-10
docbook-style-dsssl-1.64-3
docbook-utils-0.6.9-2
docbook-utils-pdf-0.6.9-2
openjade-1.3-17
jadetex-3.6-4
```

## **Az általam készített kiegészítések:**

- *A lényeg:* lyxdb2all. Ez egy shell script a LyX -> HTML, PDF konverzió leegyszerűsítésére, és egy-két hiba kijavítására.
- Stíluslap HTML generáláshoz mydocbook-html.dsl
- Stíluslap PDF generáláshoz: mydocbook-print.dsl
- CSS a generált HTML-hez (kihagyható): mycss.css

(A .dsl file-okat a docbook-utils-hoz tartozó docbook-utils.dsl file-ból alakítottam át.)

## **LyX**

### **A dokumentum beállításai**

Egy új dokumentum írásakor az alábbiakat kell beállítani:

1. Layout/Document/Document/Class: SGML (DocBook article vagy book)
2. Layout/Document/Language/Language: magyar
3. Layout/Document/Language/Quote style: „text”, double

A LyX képes a szerkesztett dokumentumot többféle formátumba exportálni, de ez nem működik egészen tökéletesen, ezért majd a lyxdb2all shell script-et használjuk. Elég, ha mindig a LyX saját formátumába mentünk.

### **Billentyűzet-parancsok**

Néhány hasznos rövidítés, hogy ne kelljen egeret használni: :)

#### **Blokkok**

M-p t

Title -- a dokumentum címe

M-p a

Abstract -- összefoglaló a dokumentum elején

M-p 1

Chapter -- fejezetcím

M-p 2

Section -- szakasz cím

M-p 3

Subsection -- 2. szintű szakasz cím

M-p 4

Subsubsection -- 3. szintű szakasz cím

M-p s

Standard -- normál szöveg

M-p d

Description -- leíró lista

M-p i

Itemize -- egyszerű lista

M-p e

Enumerate -- számozott lista

### **Karakterek**

M-c s

SansSerif -- groteszk betűtípus (*LyX-ben használható, de a DocBook-nál elveszik!*)

M-c p

Typewriter -- írógép stílusú betűtípus (*ilyen sincs a DocBook-ban, de lásd a lyxdb2all leírását!*)

M-c r

Regular -- normál betűtípus

M-c e

Emphasis -- *kiemelés* (kurzív)

### **Emacs billentyűzet-parancsok**

Az Edit/Preferences menüben a Bind file-nál megadhatunk tetszőleges billentyűzet-kiosztást. Az `emacs.bind` az Emacs-ot utánozza.

## **LaTeX: magyar elválasztás beállítása**

A `/usr/share/texmf/tex/generic/config/language.dat` file-ban engedélyezni kell a magyar elválasztást, utána bárholnan adjuk ki a következő parancsot:

```
initex latex.ltx \\dump
```

A keletkezett `latex.fmt` file-t másoljuk a `/usr/share/texmf/web2c/` könyvtárba.

## Konvertálás

A LyX file-okból a `lyxdb2a11` shell scripttel közvetlenül tudunk HTML és PDF file-okat generálni. A `lyxdb2a11` elvégzi a LyX -> DocBook SGML átalakítást a `docbook-utils` programjai segítségével. Közben kijavít néhány hibát, főleg a magyar nyelv kezelésével kapcsolatban.

### Különböző LyX verziók!

A LyX 1.1.6-os és az 1.2.0-ás verziója nem egyformán exportál SGML-be, ezért a `lyxdb2a11` scripten is módosítani kell. A script közepe táján megjelöltem azokat a sorokat, amelyeken változtatni kell. A script mostani verziója 1.2.0-ás LyX-re van beállítva, ezzel tehát változtatás nélkül használható.

### A `lyxdb2a11` használata (`lyxdb2a11 -h`):

Használat: `lyxdb2a11 [kapcsolók] <filenéveleje>`

[kapcsolók]:

```
-w|--htmldsl <file>      a HTML konverzióhoz használt stíluslap
-p|--printdsl <file>    a HTML konverzióhoz használt stíluslap
                        (A két stíluslapot kötelező megadni, és
                        nem használható a '#', mint az openjade-nél.)
-c|--css <file>         a HTML-hez használt CSS stíluslap
-f|--fonthack           a typewriter font -> tag helyettesítés bekapcsolása
-t|--typewritertag <tag> a typewriter fonthoz helyettesítendő DocBook tag
-r|--htmlprefix <dir>  a generált HTML file nevek prefixe
-h|--help               ez a segítség
```

<filenéveleje>:

```
az eredeti LyX file neve a ".lyx" végződés nélkül
Pl.: lyxdb2a11 -p mydocbook-print.dsl -w mydocbook-html.dsl mydoc
a mydoc.lyx-ből létrehozza a mydoc.pdf, *.html file-okat.
```

### Magyarázat a `-f` és `-t` kapcsolókhoz:

A LyX-ben „typewriter” fontra állított szövegeket lecseréli a `-t`-vel megadott DocBook elemre. Ez egy elég csúnya kavarás, néhány esetben hibásan működik. (Az egyik hiba a `<` és `>` karakterekkel kapcsolatos: *ezeket ne használjuk typewriter fonttal!*) Egy-egy szóra általában jó, de nagyobb blokkokon keresztül nem tanácsos állítgatni a typewriter-t. A DocBook-ban csak egyféle karakterformázás van, az `<emphasis>`. Nincsenek fontok, bold, stb. Viszont van helyettük sok tartalmi elem, pl. a `<filename>` és a `<classname>`, amiket általában más fonttal kell formázni megjelenítéskor, de ezeket az elemeket a LyX-ből nem lehet megadni. Ezzel a trükkkel a LyX-ben a typewriter fontot használhatjuk arra, hogy valamelyik elemet jelöljük. Ha a `-t`-vel nem adunk meg semmit, akkor a `lyxdb2a11 <filename>` elemet fog behelyettesíteni.

### Még egy példa: így készült ez a „Hasznos dolgok” leírás:

```
lyxdb2a11 -p mydocbook-print.dsl -w mydocbook-html.dsl -f -c ../mycss.css -r hasznos_ hasznos
```

A `hasznos.lyx` file-ból létrehozta az `index.html`, `hasznos_*.html`, és `hasznos.pdf` file-okat a megadott `.dsl` stílusfile-ok alapján. A HTML file-ok megjelenését pedig tovább módosítja a `mycss.css`.

## CVS jegyzetek

Az alábbiak a CVS kipróbálása és tanulgatása közben írt jegyzeteim. Ajánlott részletesebb leírások: CVS-RCS-HOWTO (<http://linuxdoc.org/HOWTO/CVS-RCS-HOWTO.html>), cvs.ps (ez a cvs-hez tartozó „hivatalos” leírás, nálam a CVS-t rpm-ből telepítve megtalálható `/usr/share/doc/cvs-1.11.1p1/cvs.ps` néven). Egyéb leírások:

- Introduction to CVS (rövid cikk az O'Reilly Network-ön) ([http://linux.oreillynet.com/pub/a/linux/2002/01/03/cvs\\_intro.html](http://linux.oreillynet.com/pub/a/linux/2002/01/03/cvs_intro.html))
- CVS Administration (rövid cikk az O'Reilly Network-ön) (<http://linux.oreillynet.com/pub/a/linux/2002/01/17/cvsadmin.html?page=1>)
- Leírások a Sourceforge-on ([http://sourceforge.net/docman/?group\\_id=1](http://sourceforge.net/docman/?group_id=1))

## Telepítés

Telepítés RedHat-en, rpm-ből. A CVS repository-t a `/home/cvsroot`-ba tesszük, létrehozunk egy `cvs` nevű felhasználót és ugyanilyen nevű csoportot, a `/home/cvsroot` tulajdonosait. Azokat a felhasználókat, akik a CVS-t használni fogják, be kell majd tennünk a `cvs` csoportba. (Ha a CVS repository már telepítve van egy távoli szerveren, és csak használni akarjuk, akkor természetesen csak az rpm-et kell telepíteni, a többi lépésre nincs szükség.)

```
rpm -i cvs...rpm
export CVSROOT=/home/cvsroot
groupadd cvs
mkdir /home/cvsroot
chmod o-rwx $CVSROOT
chmod ug+rwx $CVSROOT
# ezt nem írták a doksikban, lehet, hogy felesleges:
chown -R root.cvs $CVSROOT
# ezt sem írták, de kellett:
chmod g+s $CVSROOT
cvs init
# egy felhasználó felvétele a cvs csoportba:
# (ha már más csoportokban is szerepel, azokat is meg kell adni a -G után!)
usermod -G cvs <felhasználó>
```

## A CVS használata

### A CVSROOT beállítása a CVS használata előtt

Ha a CVS repository azon a gépen van, ahol dolgozunk:

```
export CVSROOT=/home/cvsroot
```

(A `/home/cvsroot` helyett persze a telepítéskor megadott könyvtárat kell írni.)

Ha a CVS repository egy távoli szerveren van, és ssh-n keresztül akarjuk elérni:

```
export CVS_RSH=ssh
export CVSROOT=:ext:usernév@szerver:/cvsroot/a/szerveren
```

(Minden parancs ugyanúgy használható lesz, mint lokálisan (még emacsból is!), csak minden kapcsolódáskor kérni fogja a jelszavunkat -- persze csak ha jelszavas autentikációt használunk az ssh-nál.)

## **Egy új könyvtár felvétele a repository-ba**

A `cvstest_dir` nevű könyvtárat felvesszük a repository-ba `cvstest` néven:

```
cd cvstest_dir
cvs import -m "cvstest import" cvstest NiR cvstest1-0
```

```
"cvstest import"
```

```
    szöveges megjegyzés
```

```
cvstest
```

```
    a repository-ban ide fog kerülni a könyvtárunk
```

```
NiR
```

```
    vendor string
```

```
cvstest1-0
```

```
    release tag
```

## **Egy teljes könyvtár kivétele a repository-ból**

A `cvstest` könyvtárat (annak aktuális változatát) kivesszük (létrehozza lokálisan):

```
cvs checkout cvstest
```

Egy adott release kivétele:

```
cvs checkout -r v1 cvstest
```

## **Frissítés: repository -> working copy**

```
cvs update
```

A mi változatunk és a repository-beli között lehet ütközés (pl. ugyanazt a sort megváltoztattuk mi is, és közben valaki más is):

```
rcsmerge: warning: conflicts during merge
cvs update: conflicts found in haha
C haha
```

Vagyis a `haha` nevű file-ban ütközés volt. Ilyenkor a mi változatunkat elmenti valami ilyen néven: `.#haha.1.2` (1.2 helyett mindig az aktuális verziószám lesz, ez most mindegy). A `haha` file-ba pedig beteszi az ütköző rész mindkét változatát, `<<<<<<<` és `>>>>>>>` jelek közé. Az ütköző rész(eke)t meg kell keresni, és valahogy fel kell oldani az ütközést (és persze kitörölni a `cvs` által beletett jeleket). Csak ezután csinálhatunk megint `commit`-ot.

## Frissítés: working copy -> repository

```
cvs commit -m "megjegyzés"
```

Ha nem adunk megjegyzést, akkor egy editort nyit meg, és abba kell beleírunk, úgyhogy egyszerűbb, ha megadjuk egyből a `-m` kapcsolóval.

Ha közben más módosított egy file-t (vagyis a repository-beli változata frissebb, mint a miénk):

```
cvs commit: Examining .  
cvs commit: Up-to-date check failed for 'haha'  
cvs [commit aborted]: correct above errors first!
```

Vagyis előbb a repository-ban levő változatot be kell építenünk a sajátunkba: `cvs update`

## Új file felvétele

```
cvs add ujfile
```

(A legközelebbi commit-nál fogja majd csak betenni a repository-ba.)

## Tag-ek hozzáadása

Saját tag-ek hozzáadásával megjelölhetjük a program egy adott verzióját. Ez akkor hasznos például, ha az egyes file-ok CVS-beli verzióitól függetlenül a programunk egy adott, pl. kibocsátott verzióját (release) azonosítani szeretnénk. Az alábbi parancsot a program könyvtárából kiadva az összes file aktuális verzióját megjelöli a „release-2-3-1” tag-gel:

```
cvs tag release-2-3-1 .
```

Szükség lehet tag-ek törlésére is. Például:

```
cvs rtag -d release-1-3-1 .
```

## Egy korábbi release kivétele és leágaztatása

Tegyük fel, hogy kedvenc programunknak már az 1.2-es változatán dolgozunk (és épp instabil állapotban van, mert mindent átírunk benne), de közben kiderül, hogy az 1.1-es változatban ki kell javítani egy hibát. Az 1.1-es változat mondjuk `rel-1-1` nevű release-ként szerepel a repository-ban. A megoldás az, hogy kivesszük a `rel-1-1`-et (pl. másik lokális könyvtárba, mint ahol az aktuális verzió van), ebben megcsináljuk a javítást. Ezután létrehozunk egy leágazást (branch) az 1.1-ből, mondjuk `rel-1-1-branch` néven, és a javítottat ebbe a külön ágba tesszük be, mondjuk `rel-1-1-patch1` néven. Ezután két águnk lesz, a „normál” `MAIN` és a `rel-1-1-branch`. Ezekbe párhuzamosan fejleszthetünk.

```
cd könyvtar_az_1-1-nek  
cvs checkout -r rel-1-1 cvstest  
# ...megcsináljuk a javítást...  
cvs tag -b rel-1-1-branch  
cvs update -r rel-1-1-branch
```

## Hasznos dolgok

```
cvcs commit -m "kijavítottam azt a csunya nagy bugot"  
cvcs tag rel-1-1-patch1 .
```

A „`cvcs tag -b rel-1-1-branch`” paranccsal egy „branch tag-et” adtunk hozzá, ezzel jön létre a leágazás, ez a tag fogja az ágat azonosítani. A végén a `rel-1-1-patch1` tag csak a release megjelölésére szolgál.

A fenti műveletek eredménye egy ilyen fa lesz (a branch tag-eket ( ) jelöli, a normál tag-eket [ ]):

```
(MAIN)  
 |  
 ...  
 |  
 [rel-1-1]----- (rel-1-1-branch)  
 |               |  
 ...             [rel-1-1-patch1]  
                  |  
                  ...
```

## Egyéb parancsok, kapcsolók

### `cvcs status`

Kiírja a file állapotát (frissebb-e, mint a CVS-beli, vagy a CVS-ben van-e változtatás, stb.)

### `cvcs log`

Kiírja az összes eddigi eseményt, minden változatot.

### `cvcs diff`

Megmutatja, hogy a CVS-ből kivett verzióhoz képest mit változtattunk

### `cvcs annotate`

Kiírja az összes eddigi verziószámot, a usert, aki betette, a dátumot, és a commit-kor beírt megjegyzést.

A CVS-nek kétféle kapcsolót lehet megadni: olyat, ami az egész CVS programra vonatkozik (ezt a parancs előtt kell megadni, pl. `cvcs -n update`), és olyat, ami csak az adott parancsra szól (ezt a parancs után kell megadni, pl. `cvcs update -d`).

Hasznos a `-n` kapcsoló, ami az egész CVS-re vonatkozik, bármilyen parancs elé írható. Azt jelenti, hogy a megadott parancsot valójában nem végzi el a CVS, tehát nem változtat semmit, csak megmutatja, mi történne (mindent ugyanúgy kiír, mintha a parancs lefutott volna). Például a `cvcs -n update` paranccsal megnézhetjük, változott-e a repository, anélkül, hogy bármit is letöltenénk.

## File-ok figyelmen kívül hagyása

Ha a CVS által kezelt könyvtárba új, a CVS számára ismeretlen file-ok kerülnek be, akkor a CVS ezekre figyelmeztet, például az `update` és `commit` parancsoknál kilistázza ezeket a file-okat egy kérdőjellel megjelölve. Ez hasznos arra, hogy észrevegyük, ha elfelejtettünk kiadni a

`cvs add`-ot új file-jainkra. Viszont sok esetben keletkeznek olyan file-ok, amiket nem a CVS-nek nem kell kezelnie. Például fordítás eredményeképpen az object file-ok. Ilyenkor zavaró a sok ismeretlen file listája. Néhány file-típusról a CVS felismeri, hogy nem kell vele foglalkoznia. (Például a `~` végződésű backup file-okat ismeri, de a java `.class` file-okat nem.) Ha meg akarjuk adni a CVS-nek, hogy milyen file-okat (vagy könyvtárakat hagyjon) figyelmen kívül, akkor ezt a legegyszerűbb `.cvsignore` file-okban elhelyezni, amelyekben egyszerűen felsoroljuk a nem kívánt file-okat (mintákat is használhatunk). Például:

```
*.class
*.html
ezegykonyvtar
ezzelsefoglalkozz.jar
```

(A `.cvsignore` file-okat is be kell tenni a CVS-be.)

## Logolás és automatikus email commit-okról

A CVS-nek beállítható, hogy különböző eseményeknél futtasson le külső programokat, amelyekkel a működés befolyásolható (pl. commit előtti ellenőrzés, vagy commit utáni értesítés, logolás). A commit-ok logolása és az automatikus email értesítés például beállítható a következőképpen:

- A CVS létrehoz a repository-ban egy kitüntetett könyvtárat `CVSR00T` néven. Ebben a CVS `config` file-jai tárolódnak. Ezeket is ugyanúgy kell kezelni, mint a többi CVS-ben tárolt file-t: kivesszük a repository-ból ezt a könyvtárat, lokálisan módosítunk valamit, majd commit-olunk. Először tehát checkout:

```
cvs checkout CVSR00T
```

- A `loginfo` file-ban adhatjuk meg, hogy milyen program fusson le egy commit után. Ebbe írjuk be ezt a sort:

```
DEFAULT /usr/share/cvs/contrib/log -s -m emailcím -f /var/log/cvs/cvslog -u $USER %s
```

A `/usr/share/cvs/contrib/log` egy Perl-script, a CVS „nem hivatalos” tartozéka. A commitokról tud email értesítést küldeni, és file-ba logolni. A levelek a `-m` kapcsolónál megadott címre mennek, a log pedig a `-f` kapcsolónál megadott file-ba. Ezután commit-oljunk a kivett `CVSR00T` könyvtárból:

```
cvs commit -m "loginfo beállítás"
```

- A logfile könyvtára még nem létezik, ezért létrehozzuk, és beállítjuk a megfelelő jogosultságokat:

```
mkdir /var/log/cvs/
chown root.cvs /var/log/cvs/
chmod g+s /var/log/cvs
chmod g+w /var/log/cvs
chmod o= /var/log/cvs
```

A script az első bejegyzéskor az adott user nevében hozza létre a logfile-t, és sajnos a group-nak nem ad írásjogot, így legközelebb egy másik user nem fogja tudni írni. Ezért az első bejegyzés után adjuk ki a `chmod g+w var/log/cvs/cvslog` parancsot. (Az igazi megoldás a script módosítása lenne...)

- Ha `logrotate`-t használunk, akkor azt is érdemes beállítani: A `/etc/logrotate.d` könyvtárba tegyünk egy `cvs` nevű file-t ezzel a tartalommal:

```
/var/log/cvs/cvslog {  
    missingok  
    notifempty  
    monthly  
}
```

## Emacs

Ha az `emacs`-sal megnyitunk egy olyan file-t, amit `cvs checkout`-tal vettünk ki előzőleg, akkor automatikusan `cvs`-módba kerül (a státusz sorban látszik a CVS felirat, és a file verziója).

`c-x c-q`

Ez a `cvs commit` és a `cvs update` „összevont” változata. Csak akkor működik, ha változtattunk a file-on. Ha a mi változatunk a legfrissebb, akkor beteszi a CVS-be. Ha mások is változtattak rajta, akkor ezeket a változtatásokat beolvasztja a miénkbe. (Ha konfliktus volt, akkor ki kell javítanunk.) Előhoz egy új buffert, ahova a megjegyzést írhatjuk. Ha beírtuk, akkor `c-c c-c` hatására beteszi a file-unkat a CVS-be (mint a `cvs commit` parancs).

`c-x v =`

Mint a `cvs diff` parancs.

`c-x v g`

Mint a `cvs annotate` parancs.

`c-x v i`

Egy új file felvétele a CVS-be. Az új file CVS-módba kerül, de még nem teszi be a CVS-be, csak majd a `c-x c-q` hatására.

`c-x v l`

Mint a `cvs log` parancs.

`c-x v u`

Visszavonja a legutóbbi check-inhez képest történt változtatásokat. (Megmutatja a különbséget, és rákérdez.)

## A viewcvs

A `viewcvs` ([viewcvs.sourceforge.net](http://viewcvs.sourceforge.net) (<http://viewcvs.sourceforge.net>)) egy szép webes felület a CVS-hez. A repository-t csak olvassa, tehát nézegetéshez, file-ok letöltéséhez való. A forrásfile-okat meg tudja jeleníteni szintaxis szerinti színezéssel (az `enscript`-et használja), látványos, színes diff-eket csinál, satöbbi.

Kell hozzá az RCS (a RedHat-nél külön rpm-ben van), valamint a színezéshez érdemes feltenni az `enscript`-et is.

A `viewcvs`-t telepíthetjük bárhova, root sem kell hozzá feltétlenül, csak írás jog oda, ahova tesszük. Én a `/usr/local/viewcvs`-be telepítettem:

```
mkdir /usr/local/viewcvs
cd <ahova-a-viewcvst-kicsomagoltuk>
./viewcvs-install
```

A `/usr/local/viewcvs/viewcvs.conf` fontosabb beállításai:

```
cvs_roots = NiR projektek : /home/cvsroot
default_root = NiR projektek
use_enscript = 1
address = <a href="mailto:denespal@nir.hu">denespal@nir.hu</a>
main_title = Ez itt a NiR CVS Repository
```

A `viewcvs` cgi scriptjei a webszerver felhasználójának nevében fognak futni (apache). Ha ez a felhasználó nincs benne a cvs csoportban, akkor nem fogja tudni olvasni a repository-t. Ha benne van, akkor pedig írás jogot is szerez rá, ami nem szerencsés. Ezért a CVS repository-t az apache tulajdonába vesszük, és levesszük az írás jogát. Így az apache mindent tud olvasni, de írni nem. (Ezt a trükköt Boát találta ki, köszönet neki :))

```
chown apache.cvs /home/cvsroot/
chmod u-w /home/cvsroot/
```

(Az apache felhasználó elméletileg visszaadhatja magának az írás jogot, de felülrni ekkor sem fog tudni file-okat.)

A `httpd.conf` beállításai:

```
ScriptAlias /viewcvs/ "/usr/local/viewcvs/cgi/"
<Directory "/usr/local/viewcvs/cgi">
    AllowOverride None
    Options ExecCGI
    Order allow,deny
    Allow from all
</Directory>
```

Ezután a `viewcvs` a `http://localhost/viewcvs/viewcvs.cgi/` url-en érhető el.

## cvsgraph

A `viewcvs`-hez egy hasznos kiegészítő a `cvsgraph`: egy file összes verzióját, az összes branch szerkezetét megjeleníti grafikusán. Ha ezt telepítjük, akkor a `viewcvs`-hez tartozó `viewcvs.conf` file-ban a

```
use_cvsgraph=1
```

opcióval bekapcsolhatjuk a `cvsgraph` használatát. Ennek hatására a `viewcvs` a file-ok listázásánál kitesz egy kis ikont, amire kattintva megkapjuk a `cvsgraph` ábráját.

## PostgreSQL

Az alábbiak elsősorban a PostgreSQL 7.3.x változatára vonatkoznak, másra nem próbáltam ki.

### Nyelvek, karakterkészletek -- magyar ékezetek kezelése

Ha nem csak ASCII karaktereket akarunk tárolni az adatbázisban, akkor felmerül a nyelvek és karakterkészletek kezelésének problémája. Ilyen esetekben valószínűleg nem lesznek megfelelők a PostgreSQL alapértelmezett beállításai, nekünk kell beállítanunk egy-két dolgot.

Most a magyar karakterek kezeléséhez szükséges dolgokat nézzük meg, de a lényeg természetesen bármilyen más nyelvre is érvényes.

Fontos fogalmak:

Locale:

A nyelvi környezetet határozza meg: a karakterek osztályozását és rendezési sorrendjét, dátum-, idő-, pénznem-, és számformátumokat, valamint a PostgreSQL üzeneteinek nyelvét. A legfontosabb talán a rendezési sorrend, ami például az `ORDER BY` működését befolyásolja: ha nincs jól beállítva, akkor például az ékezetes betűk nem a magyar ábécé szerinti helyükre kerülnek.

A locale egyes jellemzőit menet közben is változtatni lehet, de a karakterek osztályozását és rendezési sorrendjét nem! Ezek az összes adatbázisunkra rögzítettek, tehát az `initdb` futtatásakor lehet csak megadni őket, és később már nem változtathatók!

Alapértelmezés szerint az `initdb` a futásakor épp érvényes aktuális locale-t használja (a `LANG` és az `LC_*` környezeti változók értékei). Ezt megnézhetjük például a `locale` paranccsal. Ez valószínűleg `en_US` lesz. A magyar ékezetes betűk rendezése ennél a locale-nál is helyes!

Megadható egy speciális locale is, a „C”. Ez azt jelenti, hogy nem használunk semmilyen nyelvfüggő dolgot. Ennek a sebességnövekedés lehet az előnye. Például mintaillesztésnél csak akkor tud indexeket használni a PostgreSQL, ha ez a locale van beállítva. Ezzel azonban a magyar ékezetes betűk rendezése hibás lesz!

A `to_char` függvényeknél használt formátum-stringeknél megadható néhány olyan minta, ami locale-től függő eredményt produkál. De érdekes módon a dátumformázásra nem találtam ilyet, tehát pl. a hónapneveket nem sikerült magyarul kiírni.

Karakterkészlet kódolás az adatbázisban:

Minden adatbázisunk valamilyen meghatározott kódolás szerint tárolja a karaktereket. Ez adatbázis létrehozásakor megadható a `CREATE DATABASE` parancs `WITH ENCODING='kódolás'` paraméterével. A kódolás használhat egy (pl. `latin1`, `latin2`) vagy több byte-ot (pl. `unicode`) is a karakterek tárolására.

Kliensoldali kódolás:

A kliensek használhatnak más kódolást is, mint az adatbázis. Ebben az esetben íráskor és olvasáskor mindig konvertálódnak az adatok. Ezzel akkor van baj, ha a kliens oldalán olyan karakter jön, amit nem lehet az adatbázis kódolásában eltárolni. Vagy fordított eset is lehet, hogy a kliens nem tud megjeleníteni egy adatbázisból beolvasott karaktert. A `psql`-ben a `\\encoding` paranccsal lehet menet közben átállítani a kliensoldali kódolást.

# Cocoon jegyzetek

## Beállítások

### Sitemap újraolvasás

A sitemap változását a Cocoon kétféle beállítás szerint tudja követni: szinkron vagy aszinkron. Fejlesztés közben hasznos, ha minden változás azonnal érvényesül, ehhez a `cocoon.xconf`-ban keressük meg a `sitemap` elemet, és a `reload-method` attribútumát állítsuk „`synchron`”-ra:

```
<sitemap check-reload="yes" file="sitemap.xmap" logger="sitemap"
  reload-method="synchron"/>
```

Ha a több sitemap-ot használunk, akkor az egyes sitemap-ok mountolásánál is meg kell ezt adni:

```
<map:mount uri-prefix="valami" src="konyvtar/" check-reload="yes"
  reload-method="synchron"/>
```

(A Cocoon FAQ-ban van erről részletesebb magyarázat (<http://xml.apache.org/cocoon/faq/faq-sitemap.html#faq-6>) is.)

## Tippek

- A Cocoon hibaüzenetei sokszor nem túl sokat mondanak, gyakoriak az olyan misztikus hibák, amelyeket valami eldugott rossz karakter okoz. A hibakeresésben segítenek valamennyit a logok. A Cocoon alapbeállításai szerint mindent a maximális részletességgel logol a `WEB-INF/logs` könyvtárba. Itt a `core.log` és a `sitemap.log` file-okat érdemes figyelni (`tail -f`).
- A Cocoon nem követi az xslt file-okban az `include-olt` file-ok függőségeit. Vagyis ha egy xsl file-ban `<xsl:include>`-dal egy másik xsl file-ra hivatkozunk, és ezt megváltoztatjuk, akkor a Cocoon nem generálja újra a transzformációt. Ehhez a „fő” xsl file-t is frissíteni kell (pl. `touch`), ebből észreveszi, hogy változott valami.
- Ha `Exception in creating Transform Handler` hibát kapunk, az általában azt jelzi, hogy egy xsl file-ban szintaktikai hiba van, pl. egy lezáró tag nélküli tagból kihagytuk a `/-t: <tag> -> <tag/>`.
- A hibaüzenetekből gyakran egyáltalán nem derül ki, hogy melyik file-ban, hol lehet a hiba. Ilyenkor a legjobb módszer az, ha apró lépésekben változtatunk, és minden lépésben kipróbáljuk, hogyan működik. Pl. ha tudjuk, hogy a hibát valami nagyobb változtatás okozta, akkor minden változtatást kikommentezünk, és fokozatosan „kommentezzük vissza”, így a hibát általában hamar „be lehet keríteni”.
- Előfordul, hogy ha menet közben változtatunk egy xsp file-on vagy egy sitemap-en, akkor a Cocoon nem olvassa újra a változásokat. Ilyenkor vagy várni kell egy kicsit, vagy újraindítani a servlet engine-t. (Ha például a sitemap-ben az action-ök definícióján változtatunk, akkor lehet, hogy csak az újraindítás segít.) Egyelőre nem jöttem még rá, hogy pontosan mikor hogyan viselkedik... Ez a jelenség eléggé kellemetlen lehet debugolás közben, ha

nem tudjuk, hogy már a megváltoztatott file szerint működik-e, vagy még a régi szerint. Érdemes jelzést betenni (pl. xsp file-nál egy feliratot), amiből látjuk a böngészőben, hogy az új változat fut-e már.

- Ha egy XSP oldal az elején megadott `<?xml-stylesheet href="..."?>` PI-vel hivatkozik egy logicsheet-re, és menet közben változtatunk a logicsheet-et tartalmazó XSL file-on, akkor a Cocoon ezt nem veszi észre, és az XSL régi változatát használja. A frissítéshez változtatni kell az XSP file-on is.
- Hasonló a helyzet, ha egy XSP oldalt nem közvetlenül file-ból állítatunk elő, hanem egy másik pipeline-ból.
- Action-set nevében nem lehet kötőjel! (Ha kötőjel van a névben, akkor a sitemap fordítása egy furcsa hibával leáll.)

## Egyebek

### XML szerkesztés Emacs-al

XML file-ok szerkesztéséhez (persze SGML-hez is) érdemes a psgml csomagot használni. RedHat-hez külön rpm-ből kell telepíteni, az alap emacs csomagban nincs benne. Az alap beállításokkal nem volt elég kényelmes az XML szerkesztés, ezért a .emacs file-omba az alábbiakat tettem:

A file legelejére (azt hiszem, számít a sorrend):

```
(autoload 'sgml-mode "psgml" "Major mode to edit SGML files." t)
(add-to-list 'auto-mode-alist '("\\.xml\\$" . xml-mode))
(add-to-list 'auto-mode-alist '("\\.xsl\\$" . xml-mode))
(add-to-list 'auto-mode-alist '("\\.xsp\\$" . xml-mode))
(add-to-list 'auto-mode-alist '("\\.xmap\\$" . xml-mode))
(setq sgml-indent-data 1)
```

(Megadhatunk más, gyakran használt file-végződéseket is, így pl. html-t is szerkeszthetünk xml-módban.)

A file végére pedig ezt írtam:

```
(add-hook 'sgml-mode-hook 'turn-on-font-lock)
;;
;; PSGML, courtesy of David Megginson, dmeggins@uottawa.ca
;;
(make-face 'sgml-comment-face)
(set-face-foreground 'sgml-comment-face "orange red")
(make-face 'sgml-doctype-face)
(set-face-foreground 'sgml-doctype-face "plum")
(make-face 'sgml-end-tag-face)
(set-face-foreground 'sgml-end-tag-face "sky blue")
(make-face 'sgml-ignored-face)
(set-face-foreground 'sgml-ignored-face "blue")
(make-face 'sgml-ms-end-face)
(set-face-foreground 'sgml-ms-end-face "maroon")
(make-face 'sgml-ms-start-face)
(set-face-foreground 'sgml-ms-start-face "maroon")
(make-face 'sgml-pi-face)
```

```
(set-face-foreground 'sgml-pi-face "maroon")
(make-face 'sgml-sgml-face)
(set-face-foreground 'sgml-sgml-face "maroon")
(make-face 'sgml-start-tag-face)
(set-face-foreground 'sgml-start-tag-face "sky blue")
(make-face 'sgml-entity-face)
(set-face-foreground 'sgml-entity-face "goldenrod")
(make-face 'sgml-short-ref-face)
(set-face-foreground 'sgml-short-ref-face "firebrick")
(setq-default sgml-markup-faces
  '((comment . sgml-comment-face)
    (doctype . sgml-doctype-face)
    (end-tag . sgml-end-tag-face)
    (ignored . sgml-ignored-face)
    (ms-end . sgml-ms-end-face)
    (ms-start . sgml-ms-start-face)
    (pi . sgml-pi-face)
    (sgml . sgml-sgml-face)
    (start-tag . sgml-start-tag-face)
    (entity . sgml-entity-face)
    (short-ref . sgml-short-ref-face)))
(setq-default sgml-auto-activate-dtd t)
(setq-default sgml-set-face t)
```

## GPG

### PGP-vel (GPG-vel) aláírt file ellenőrzése

Internetről letölthető file-okat (pl. programokat) gyakran PGP aláírással hitelesít a szerzőjük. Az ellenőrzéshez szükség van a szerző publikus kulcsára, és az aláírásra. Ha a letöltött file-unk neve *valami*, a publikus kulcs a *pubkey.asc* file-ban van, az aláírás pedig a *valami.sign* file-ban van, akkor így ellenőrizhetjük az aláírást:

1. Importáljuk be a kulcsot: `gpg --import pubkey.asc` (A beimportált kulcsot megjegyzi, legközelebb már megtalálja. A kulcsainkat listázhatjuk a `gpg --list-keys` paranccsal.)
2. Ellenőrizzük a file-t: `gpg --verify valami.sign valami`

Rpm file-okat is lehet GPG-vel aláírni, ilyenkor nincs külön aláírás-file, mert az rpm tartalmazza az aláírást. Az ellenőrzés menete:

1. Importáljuk be a kulcsot az előbbi módon.
2. Ellenőrizzük a file-t: `rpm -K valami.rpm`

## Tartalom

LyX és DocBook magyarul.....	1
Hozzávalók .....	1
LyX .....	2

LaTeX: magyar elválasztás beállítása.....	3
Konvertálás .....	3
<b>CVS jegyzetek .....</b>	<b>5</b>
Telepítés .....	5
A CVS használata.....	5
Logolás és automatikus email commit-okról .....	9
Emacs.....	10
A viewcvs .....	10
<b>PostgreSQL.....</b>	<b>11</b>
Nyelvek, karakterkészletek -- magyar ékezetek kezelése .....	12
<b>Cocoon jegyzetek .....</b>	<b>13</b>
Beállítások.....	13
Tipppek.....	13
<b>Egyebek .....</b>	<b>14</b>
XML szerkesztés Emacs-al .....	14
GPG .....	15