E-közigazgatási Szabad
Szoftver Kompetencia Központ

OpenDocument és LibreOffice intézményi környezetben

Az MSZ ISO/IEC 26 300:2009 OpenDocument nyílt formátum (ODF) és a LibreOffice irodai programcsomag szakalkalmazás-fejlesztési és üzemeltetési lehetőségei

Készítette a Közigazgatási és Igazságügyi minisztérium E-közigazgatási
Szabad Szoftver Kompetencia Központja
Budapest, 2013

Kódszám: EKOP–1.2.15

A dokumentum legfrissebb változata letölthető a honlapunkról:
http://szabadszoftver.kormany.hu/szabad-szoftver-keretrendszer/

ÚJ SZÉCHENYI TERV

Tartalomjegyzék

Bevezetés

A könyv bemutatja az OpenDocument dokumentumformátum és a LibreOffice irodai programcsomag lehetőségeit a szakalkalmazás-fejlesztés és az intézményi dokumentumkezelés terén. Segítséget nyújt a jogdíjmentes OpenDocument formátum és LibreOffice irodai programcsomag szakalkalmazásokban való felhasználásában, függetlenül attól, hogy a szakalkalmazás zárt vagy nyílt forráskódú, vagy ezek valamilyen kombinációja[1]. A leírások alapján a zárt dokumentumformátumoktól, zárt irodai programcsomagtól függő szakalkalmazások kiegészíthetők az OpenDocument, illetve LibreOffice támogatásával, akár platformfüggetlen módon, de akár csak a LibreOffice-ra támaszkodó szakalkalmazások is fejleszthetők. Bemutatásra kerül az is, hogy az intézményi üzemeltetők és szakalkalmazás-fejlesztők hogyan élhetnek a nyílt forráskódú LibreOffice előnyeivel: a forráskód szabad megtekinthetőségével és módosíthatóságával, az aktív fejlesztői közösség támogatásának igénybevételével, a LibreOffice mögött álló Document Foundation fejlesztési infrastruktúrájának kihasználásával, és a nyitott fejlesztői közösséghez való csatlakozással.

Nyílt standard és szabad szoftver

A szabad, más néven nyílt forráskódú szoftverek, mint a LibreOffice és a nyílt szabványok, mint az OpenDocument számos előnyt jelentenek a szakalkalmazások fejlesztői számára:

- a zárt, nem szabványosított formátumok, csatolófelületek, szoftverek okozta, a szakalkalmazások fejlesztőinek is gondot jelentő termék- és platformfüggőség felszámolása;
- minimális bekerülési költség (a valódi nyílt szabványok, mint az OpenDocument is, licencdíj-mentesek, a LibreOffice teljes értékű változata ingyenesen is hozzáférhető);
- rugalmasság a felhasználásban a szabad licencnek köszönhetően, például korlátlan másolás és terjesztés, módosíthatóság, tetszőleges célra való felhasználás, szemben a zárt licencű programokkal[2];
- a fejlesztői produktivitás maximalizálása: pl. a gyors üzembe helyezés, módosíthatóság, a nyílt fejlesztői közösség kivételes segítsége (számos magyar hallgató vált pár hónap alatt sikeres LibreOffice fejlesztővé ennek és a Google GSoC fejlesztői ösztöndíjának köszönhetően);
- A termékcsapdából kilépés, illetve a szabad szoftveres fejlesztési modell jobb szolgáltatásokat, kedvezőbb árakat, a fejlesztőknek, szolgáltatóknak, tanácsadóknak nagyobb piacot és versenyképességet jelenthet egy változó környezetben, ahol a változást egyre erőteljesebben a nyílt szabványok és a szabad szoftverek határozzák meg.[3] Jó példa erre Nagy-Britannia, amely egyértelműen a szabad szoftverek mellett kötelezte el magát:[4] e-kormányzati szolgáltatásaik nyílt

[1] L. *Szabad szoftver üzleti modellek.* EKOP–1.2.15, KIM, 2013

[2] Barna József: A netkávézók eddig illegálisan használtak Windowst, IT café, 2010-01-21, http://itcafe.hu/hir/netkavezok_microsoft_windows_rental_rights.html

[3] L. *A szabad szoftverek helyzete az EU-ban.* EKOP-1.2.15, KIM, 2013

[4] http://publications.cabinetoffice.gov.uk/digital/strategy/: *„it will require redesigned digital services to: be developed based on user need using agile, iterative, digital development methodologies and using open source code by default”*

forráskódúak lesznek alapértelmezés szerint 2014 áprilisától, a megfelelő szabad licenc alatti kódmegosztással.[5]
– A közigazgatási szakalkalmazásoknak nem kötelező nyílt forráskódúaknak lennie Magyarországon, de az elektronikus közszolgáltatást nyújtó szoftverek csatolófelületénél már előírás a nyílt szabvány. Az European Interoperability Framework 2-es változatának, és a kapcsolódó vállalásoknak megfelelően Magyarország a többi uniós tagállamhoz hasonlóan elkötelezi magát a nyílt szabványok mellett.
– Közbeszerzéseknél a szabad szoftverek kötelezően megvizsgálandó alternatívák, irodai programcsomagok esetében pedig a nyílt forráskódúakat részesíti előnyben a magyar közigazgatás az erre vonatkozó kormányhatározat alapján.[6]

OpenDocument

Előnyök

– Nemzetközi iparági standard (a szabvány kidolgozásában szabványosító testületek, mint az OASIS és az ISO, iparági szereplők, mint az IBM, Sun MicroSystems, Boeing, Microsoft vettek, illetve vesznek részt);
– 2006-tól ISO,
– 2009-től magyar dokumentumszabvány (MSZ ISO/IEC 26 300:2009);
– valódi nyílt szabvány: megismerését és felhasználását nem nehezíti meg copyright, szabadalom, egyéb jogi korlátozás, így licencdíj és egyéb költség nélkül használhatók szakalkalmazásokban, de akár szabad szoftverekben is;
– XML alapú JAR/ZIP archívum: egyszerűen és hatékonyan feldolgozható, ehhez a W3C szabvány XSLT programnyelv és számos egyéb programkönyvtár áll rendelkezésre;
– a dokumentumkezelő rendszerekben (például nyílt forráskódú Alfresco) hatékonyan tárolható;
– minden elterjedt irodai programcsomag, így a magyar közigazgatásban is használt LibreOffice, OpenOffice.org, Apache OpenOffice, MS Office, Google Dokumentumok is kezeli.
– számos irodai programcsomag, így a LibreOffice, Lotus Symphony, OpenOffice.org, Apache OpenOffice, Calligra Suite, EuroOffice alapértelmezett dokumentumformátuma;
– magyar érdekvédelmi szervezettel rendelkezik: az ODF Alliance magyar tagozata[7] képviseli a nyílt és szabványos dokumentumformátumot hazánkban. Emellett a tagozat 2010-ben Budapesten rendezte meg az éves OpenOffice.org konferenciát, ahol hivatalos programként került sor az ODF interoperabilitási bemutatóra, nem hivatalos programként pedig a Document Foundation megalakítására, és a LibreOffice fejlesztés megindítására.

Hátrányok

– Az OpenDocument a magyar közigazgatásban terjedőben lévő dokumentumformátum, ami azt jelenti, hogy nincsenek mindenhol felkészülve a használatára. Szervezeten belül, ahol egységes OpenDocument környezet kerül kialakításra, ez nem jelent problémát. Kapcsolattartás céljából pedig megoldást jelent, hogy számos ingyenesen elérhető szerkesztőprogram (pl. az USB-kulcsra is telepíthető LibreOffice, a csak szöveges dokumentumokhoz használható, de kisméretű,

[5] https://www.gov.uk/service-manual/digital-by-default, 15. kritérium: *„Make all new source code open and reuseable, and publish it under appropriate licences (or have provided a convincing explanation of why this cannot be done for specific subsets of the source code)”*, 2013-04-23

[6] 1479/2011. (XII. 23.) sz. kormányhatározat az egyes közigazgatási szervek által használt elektronikus dokumentumok formátumáról és a nyílt forráskódú irodai szoftverek használatáról, miszerint az érintett állami szervek *„nem nyílt forráskódú irodai szoftvereket csak műszakilag vagy gazdaságilag indokolt esetben, illetve nemzetközi szerződésekből adódó kötelezettség teljesítése érdekében”* szerezhetnek csak be. Magyar Közlöny, 2011. évi 159. szám, 39 086–39 087. http://www.magyarkozlony.hu/pdf/11 386

[7] ODFA Magyarország, http://www.odfalliance.hu/

platformfüggetlen Abiword, vagy az internetes Google Dokumentumok stb.) áll rendelkezésre. Segítségképp érdemes lehet a Magyarország.hu hivatalos tájékoztató oldalát[8] is feltüntetni az ODF dokumentumok továbbításánál. Ha a szerkesztés nem követelmény, akkor a PDF lapleíró formátum használata javasolt, amely minden szabványos megjelenítővel pontos megjelenést és nyomtatást tesz lehetővé, szemben a szerkeszthető (akár zárt, akár nyílt) dokumentumformátummal és kezelőprogramjaikkal;

– Nincs 100%-os ODF-támogatás. Az OpenDocument szabvány (a legtöbb dokumentumformátumhoz, vagy akár a webes szabványok és a böngészők esetéhez hasonlóan) nincs 100%-osan támogatva egy irodai programcsomag által sem. Az alapértelmezetten OpenDocument formátumot használó, nagy tudású, többek között nyílt forráskódú irodai programok megléte azonban azt jelzi, hogy minőségi, az irodai programcsomagok funkcióit lefedő, közel 100%-os OpenDocument-támogatás már jelenleg is elérhető;
– Különböző, eltérő szabványosítási fázisban lévő ODF-változatok. Az OpenDocument legújabb, 1.2-es változata a kidolgozó OASIS szabványosító szervezet szabványa már, ISO szabványosítása viszont még nem zárult le, ennek ellenére több irodai programban, így a LibreOffice-ban is már az 1.2 az alapértelmezett formátum, viszont az ODF 1.2-t a Microsoft Office régebbi változatai nem támogatják megfelelően. Megoldás: az ODF 1.1-es mentési formátum beállítása, a LibreOffice esetében az Eszközök→Beállítások→Megnyitás és mentés→Általános lapon.
– Kiterjesztett ODF-változatok. A LibreOffice alapértelmezetten kisebb bővítésekkel rendelkező ODF 1.2-es formátumot ment el, részben a szabvány hibáinak javítása, részben a szabvány valós megvalósítást megkövetelő továbbfejlesztése céljából (többek között az MS Office alapértelmezett Office Open XML formátumával való kompatibilitás megvalósítását szolgálják). Ezek a bővítések dokumentálva vannak.[9] Általában ez nem okoz problémát, mivel a felhasználói programok, élve az XML alapú formátumok rugalmasságával, nem veszik figyelembe a nem szabványosított részeket. Amennyiben mégis, a megoldás a kívánt mentési formátum beállítása, a LibreOffice-nál az Eszközök→Beállítások→Megnyitás és mentés→Általános lapon.
– Megjelenítésbeli különbségek. Az OpenDocument formátum elméleti 100%-os támogatása esetén sem garantált a 100%-osan azonos vizuális megjelenítés, mivel ez függvénye még többek között a rendelkezésre álló betűkészleteknek és az elválasztási programkönyvtár és szótár meglétének, azonos működésének. Megoldás: valamely lapleíró formátum használata, mint az egyes változatában nyílt standard PDF-é. A LibreOffice nem csak kiváló alapértelmezett PDF exportálási lehetőséggel rendelkezik, hanem a PDF exporthoz képes az ODT dokumentumot is csatolni (hibrid PDF), kiküszöbölve a PDF szerkesztési lehetőségeinek hiányát (ez nem azonos azzal a lehetőséggel, hogy a LibreOffice képes az ODT-t nem tartalmazó PDF-eket is megnyitni a Draw rajzoló alkalmazásában, jó esetben közel helyesen megjelenítve a PDF-et, kisebb-nagyobb módosításra lehetőséget adva).

ODF dokumentumtípusok

Az OpenDocument formátumai lefedik egy átlagos irodai programcsomag funkcióit:

– ODT: szövegdokumentum-formátum (pontos külalak pozicionálható keretekkel, vektorgrafikus betűkészletekkel, képformátumokkal, fejléc- és lábléc-kezelés). Sablonként: OTT.
– ODM: fődokumentum-formátum (több külső ODT dokumentumot fog össze, lehetővé téve a párhuzamos, illetve a kevésbé erőforrás-igényes munkát és a nagy dokumentumok kezelését);
– ODS: táblázatkezelő formátum, sablonként OTS;
– ODP: bemutató formátum, sablonként OTP;
– ODD: rajzformátum, sablonként OTD;
– ODB: adatbázis-formátum.

[8] http://nyiltformatum.magyarorszag.hu
[9] http://wiki.documentfoundation.org/Development/ODF_Implementer_Notes

LibreOffice

A nyílt dokumentumformátum mellett felmerülhet az irodai programcsomagok képességeinek elérése, szakalkalmazásba való beépítése is. Ennek egyik kiváló eszköze a LibreOffice irodai programcsomag, amely a Windows és a Mac OS X mellett a szabad Linux operációs rendszeren is elérhető. A platformfüggetlenség (több, köztük nyílt operációs rendszerek) mellett nyílt forráskódú szoftverként nem korlátozza a felhasználást, lehetővé teszi a szabad másolást, módosíthatóságot, továbbfejleszthetőséget is, szemben például az MS Office-szal. Részletes, több mint száz funkcióra kiterjedő (de nem hivatalos) összehasonlítás található a Document Foundation honlapján.[10]

Előnyök

- nyílt forráskódú irodai programcsomag (a bevezetőben felsorolt előnyökkel);
- nyílt szabványokon alapul: alapértelmezett formátuma az OpenDocument, amelynek része például a nagy tudású XForms űrlapszabvány;
- ingyenesen is elérhető, illetve igény szerint kereskedelmi támogatás is vásárolható hozzá;
- elterjedt (minden második magyar vállalat használt szabad irodai programcsomagot 2011-ben a KSH adatai szerint,[11] túlnyomórészt az OpenOffice.org, és utódja, a LibreOffice valamelyikét. Ez az arány helyenként a közigazgatásban is megfigyelhető, mint például a magyar bíróságokon, amely jelenleg áll át OpenOffice.org-ról LibreOffice-ra[12]);
- platformfüggetlen: Linux, Mac OS X, Windows platformon is fut. Utóbbira elérhető USB-kulcsról indítható változat is (LibreOffice Portable).
- PDF-exportálási képesség (PDF-űrlapok, PDF/A-1 szabvány támogatása a hosszú távú adatmegőrzés érdekében).
- A zárt MS Office dokumentumformátumok (DOC, XLS, PPT) kiváló kezelése jellemzi.
- teljes és naprakész magyar honosítás: a LibreOffice jelenlegi karbantartója az EKOP 1.2.15 keretén belül működő Szabad Szoftver Kompetencia Központ. L. még: http://hu.libreoffice.org, és http://libreoffice.hu/2009/10/06/a-microsoft-office-es-az-openoffice-org-magyar-nyelve/.
- függetlenség: a LibreOffice fejlesztését a független Document Foundation végzi;
- olyan optimalizált, platformfüggetlen önkormányzati dokumentum- és űrlapkezelő keretrendszer érhető el hozzá, mint a magyar honosítással is rendelkező Wollmux.

Hátrányok

A programcsomag hátrányai részben az OpenDocument formátumhoz, részben a használatbeli sajátosságokhoz kapcsolódnak:

- A programcsomag alapértelmezett formátuma, az ODF 1.2 vegyes környezetben elképzelhető, hogy nem biztosít elegendő kompatibilitást. Megoldást jelent, hogy az alapértelmezett formátum módosítható, illetve a LibreOffice kiváló PDF exportálási képességgel rendelkezik (l. ODF hátrányai, 6. oldal);
- A program legújabb változatai bár sok hibajavítást tartalmaznak, új képességeikkel új hibát is behozhatnak, és gyakran be is hoznak (részben a szabad szoftveres fejlesztési modelljének köszönhetően, amely a hibakeresésben és -javításban a felhasználói közösségre is támaszkodik). Ezért napi használat esetén érdemes inkább a régebbi változatok sokadik javító kiadását választani, mint a legújabb változatot, vagy annak első pár javító kiadását (bővebben l. 20. oldal);
- az RTF, MS Office (DOC, XLS), Office Open XML (DOCX, XSLX) formátumokat is támogatja, de napi szintű, vegyes környezetben való használata problémába ütközhet, ha mindez a folya-

[10] http://wiki.documentfoundation.org/Feature_Comparison:_LibreOffice_-_Microsoft_Office

[11] IKT-eszközök és használatuk, Központi Statisztikai Hivatal, 2012. december, http://www.ksh.hu/docs/hun/xftp/idoszaki/ikt/ikt11.pdf

[12] Laky Norbert (Fővárosi Bíróság) 2010-es felmérése szerint.

matos ALAKHŰ[13] megjelenítés követelményével jár együtt, mivel formátumtól függően bizonyos tulajdonságok (pl. tükrözött oldalak, tartalomjegyzék kezelése) javításra szorulhatnak. Megoldás jelenthet a teljes LibreOffice migráció, vagy a stílusok és a sablonok következetes használata, valamint a részleges dokumentumformázás, a végleges külalak LibreOffice-on történő, akár automatikus kialakításával;
– számos képességben, mint a nyílt forráskódúság, nyílt szabványok támogatása, platformfüggetlenség, megelőzi a zárt forráskódú irodai programcsomagokat, de vannak olyan speciális tulajdonságok, amelyek hiányoznak a LibreOffice-ból (ilyen például az összetett kifejezések alkalmazása a kereszthivatkozásban, amelyet egyes automatikus szövegfeldolgozó programok kihasználnak). Megoldást jelent a LibreOffice bővítése, amelyhez beépített kiterjesztés-kezelő keretrendszer is rendelkezésre áll, vagy a nyílt szabványokon alapuló OpenDocument dokumentumkezelés (például XSLT, vagy ODF programkönyvtárak alkalmazása).

Document Foundation

A nemzetközi közösség által létrehozott és működtetett alapítvány vezeti a LibreOffice közösségi fejlesztését:
– A több száz önkéntes mellett több nagy cég, mint például a Red Hat Linux, Canonical, Collabora (korábban SUSE) fejlesztői vesznek részt a fejlesztésben, de az alapítvány vezetésében egy cég képviselői sem haladhatják meg a 30%-ot;
– A berlini bejegyzésű alapítvány különleges alapító okiratának megfelelően minden aktív LibreOffice fejlesztő kérelmezheti, és határozott időre megkaphatja a teljes jogú Document Foundation tagságot, részt vehet annak testületi működésében, és ezt addig újíthatja meg szabadon, amíg aktív tagja a közösségnek;
– a Document Foundation tanácsadó testületében[14] a fenti és egyéb cégek, pl. AMD, Google, Intel, SUSE mellett ott találni az alapítványt létrehozó Freies Office Deutschland alapítványt és a Free Software Foundationt is, valamint különböző kormányzati és non-profit szervezeteket.
– Az alapítvány munkáját aktív közösségi fejlesztés jellemzi: jelenleg már több mint félezer fejlesztő munkáját tartalmazza a 2010 végén az OpenOffice.org projekt folytatásaként létrehozott LibreOffice.
– Az alapítvány munkáját nem csak egyének és a csatlakozott szervezetek segítik, hanem például önkormányzatok (például München kódhozzájárulással, mind fejlesztői hétvégékkel, járult hozzá a LibreOffice fejlesztéséhez), illetve közösségi finanszírozású projektként, számos felajánló (az alapítványi bejegyzéshez szükséges 50 ezer eurót 8 nap alatt gyűjtötte össze a Document Foundation).

Programnyelvi példák

Az útmutatóban parancssori, Python, C/C++ és Java programnyelvi példák is találhatók. A parancssori példák célja a vázolt megoldások tömör bemutatása fejlesztési támpontként. Ha egyéb nem szerepel a szövegben, akkor Linux/Unix parancssori példákról van szó. Ezek egy része az unixos parancssort használó Mac OS X-ben, vagy a windowsos Cygwin környezetben is működik.

A Python példák hasonlóképpen tömörek. A Python része a LibreOffice-nak, így az irodai programcsomagra építő szakalkalmazások igénybe vehetik a Python segítségét is.

A C, C++ és Java elterjedt programnyelvek a szakalkalmazások fejlesztésénél.

[13] ALAKHŰ: „Azt látod, amit kapsz, hűen.” A WYSIWYG *(What you see is what you get)* elv magyar fordítása.
[14] http://www.libreoffice.org/about-us/advisory-board/

OpenDocument

Az OpenDocument formátum egy JAR (ZIP) tömörített archívum, benne XML és egyéb (pl. kép: PNG, JPEG, SVG, EPS) standard formátumú állományokkal (részletesen később), így kezelése – a kiváló dokumentációnak és számos segédeszköznek köszönhetően is – egyszerű. Keresés, az OpenDocument állományok megnyitása, adatok közvetlen kinyerése, módosítása a fejezet témája.

OpenDocument állományok rendszerszintű kezelése

A mai operációs rendszerek a megfelelő kiegészítőkkel feldolgozzák az OpenDocument formátumú állományok szövegtartalmát, így ezek rendszerszinten kereshetővé válnak, ráadásul rendkívül gyorsan, megspórolva ezzel hasonló funkciók beépítését a szakalkalmazásokba.

Windows iFilter szűrők

A Windows asztali és kiszolgáló oldali keresési technológiáinak alapelemét képező iFilter szűrők OpenDocument változatai az irodai programcsomagok kiegészítőjeként (Microsoft Office 2007 és 2010 Filter Pack), illetve azok alapértelmezett részeként (LibreOffice) érhetők el. Sikeres telepítés után például a standard Asztali keresés az OpenDocument állományokat szövegtartalom alapján is megtalálja.[15]

Linux keresés

Integrált, gyors asztali keresést nyújt Unity felületen a Recoll alkalmazás és keresési lencséje (recoll-lens). A lencse jelenleg csak ékezet nélküli keresőkifejezéseket támogat, cserében viszont két kattintással vagy pár billentyűlenyomással (Super gomb, majd Ctrl-Tab a Recoll lencséig) elérhető a keresőfelület. A Recoll külön grafikus felülete a Unitytől függetlenül használható Linuxokon, és ott az ékezetek elhagyására sincs szükség.

Kiszolgáló oldali indexeléshez, és gyors visszakereséshez olyan népszerű szabad eszközök állnak rendelkezésre, mint a Lucene, vagy CLucene (ez utóbbi része a LibreOffice-nak is),[16] amelyet alapértelmezetten használnak az olyan dokumentumkezelő rendszerek is, mint a piacvezető nyílt forráskódú Alfresco.

OpenDocument állományok megnyitása az alapértelmezett alkalmazással

Parancssor

A mai operációs rendszerek lehetőséget nyújtanak a fájlkiterjesztésekhez rendelt alapértelmezett alkalmazások indítására parancssorban is (részben ezt a lehetőséget használják ki a későbbi programnyelvi példák):

[15] A LibreOffice 3.6-os változatától az indexelésnek jelentős teljesítménybeli problémái akadtak, ezt a 4.1.4-es és újabb változatok javították, l. https://bugs.freedesktop.org/show_bug.cgi?id=56035.

[16] http://wiki.apache.org/lucene-java/LuceneFAQ#How_can_I_index_file_formats_like_OpenDocument_.28aka_OpenOffice.org.29.2C_RTF.2C_Microsoft_Word.2C_Excel.2C_PowerPoint.2C_Visio.2C_etc.3F

```
xdg-open példa.odt                (Linux)
open példa.odt                    (Mac OS X)
start példa.odt                   (Windows)
```

Python

A következő példaprogram Darwin, vagy az azon alapuló Mac OS X, továbbá Windows, Linux és egyéb FreeDesktop.org Portland szabványt támogató operációs rendszereken teszi lehetővé a program paramétereként megadott állományok megnyitását a hozzájuk rendelt alapértelmezett alkalmazással:

```
import subprocess, os, sys
for filepath in sys.argv[1:]:
  if sys.platform.startswith('darwin'):
    subprocess.call(('open', filepath))
  elif os.name == 'nt':
    os.startfile(filepath)
  elif os.name == 'posix':
    subprocess.call(('xdg-open', filepath))
```

C/C++

A parancssori indításnak megfelelő C példaprogram Linuxra:

```
#include <unistd.h>
main() {
  pid_t pid = fork();
  if (pid == 0) {
    execl("/usr/bin/xdg-open", "xdg-open", "pelda.odt", (char *)0);
    exit(1);
  }
}
```

Windowson a Visual C ShellExecute() függvényével indítható a hozzárendelt alkalmazás:

```
ShellExecute(NULL, "open", L"c:\\pelda.odt", NULL, NULL, SW_SHOW);
```

Java

A Java 6 SE változatban megjelent Desktop API lehetőséget nyújt az alapértelmezett böngésző és az alapértelmezett alkalmazások indítására is. A következő példaprogram először ezzel próbálkozik a paraméterként megadott állományokon. Ha a Desktop API-t nem támogatja a Java futtatókörnyezet, például Linux alatt nincs telepítve az openjdk-7-jre, csak egy régebbi változata, akkor a linuxos xdg-open meghívásával nyitja meg a paraméterként megadott állományt:

```
import java.awt.Desktop;
import java.io.File;
import java.io.IOException;

public class OpenFile {
  static public void main(String[] args)
  {
    try {
      Desktop desktop = null;
      for (String path : args) {
        File file = new File(path);
        System.out.println("Megnyitás: " + file);
```

```
            if (Desktop.isDesktopSupported()) {
              Desktop.getDesktop().open(file);
            } else {
              // Desktop API nincs támogatva.
              Runtime.getRuntime().exec("xdg-open " + file);
            }
          }
        } catch (IOException e) {
          e.printStackTrace();
        }
      }
}
```

Tartalom kinyerése OpenDocument állományokból

Az ODF archívum content.xml állománya tartalmazza az ODF dokumentum XML alapú strukturált szövegtartalmát, amelynek kinyerése és megtekintése nagy segítséget jelent az ODF-fel való ismerkedésben és a szakalkalmazások fejlesztésében is. Akár saját szövegkeresést, indexelést is megvalósíthatunk ezen a módon (de Windows, Linux és Macintosh környezetben is a rendszerszintű keresés a megfelelő beállítások esetén indexeli az OpenDocument formátumú állományok szövegtartalmát, l. előző szakaszt).

Parancssor

A Linux/Unix parancssorban a következőképpen kezelhetjük az ODF állományok szövegtartalmát:[17]

```
unzip pelda.odt content.xml                    # content.xml kimásolása az archívumból
xml_pp content.xml | less                      # az XML tartalom formázott megjelenítése
xml_grep 'text:p' content.xml                  # csak a bekezdéselemek megjelenítése (XPath)
xml_grep 'text:p' --text_only content.xml      # a dokumentum szövegtartalma
```

xsltproc

Az XSLT (Extensible Stylesheet Language Transformations) programozási nyelvvel standard módon dolgozhatók fel az XML állományok, így az OpenDocument archívum XML állományai is.

A Linux parancssori xsltproc a kövekező példában XHTML formátumú weboldallá alakít egy OpenDocument állományt a LibreOffice megfelelő XSLT szűrőjével (opendoc2xhtml.xsl). A content.xml mellett az XSLT szűrőprogram a styles.xml és a meta.xml állományokat is igényli a megfelelelő átalakításhoz, továbbá ha képeket is tartalmaz a dokumentum, a Pictures alkönyvtár teljes tartalmát:

```
unzip pelda.odt content.xml styles.xml meta.xml    Pictures/*                # állományok kimásolása
xsltproc /usr/lib/libreoffice/share/xslt/export/xhtml/opendoc2xhtml.xsl content.xml >pelda.xhtml
xdg-open pelda.xhtml        # megtekintés
```

XSLT Runner

Az Apache ODF Toolkit programkönyvtár parancssori XSLT Runner segédprogramja fel van készítve az OpenDocument archívumok kezelésére, így az előző példával ellentétben itt nincs szükség az állományok archívumból való külön kicsomagolására:[18]

[17] Az xml-twig-tools Perl csomag telepítése esetén.

[18] Az ODF XSLT Runner igényelte Saxon XSLT programkönyvtár az egyszerűség kedvéért a rendszerszintű LibreOffice-szal telepített (Ubuntu). Az ODFToolkit 0.6-os változata egyéb Apache programkönyvtár telepítését is igényli, ezért a példában – az aktuális könyvtárban kicsomagolt – 0.5-ös változat szerepelt.

```
java -cp /usr/share/java/xercesImpl.jar:odftoolkit-0.5-incubating/xslt-runner-1.2.1-incubating.jar:odftoolkit-
    0.5-incubating/odfdom-java-0.8.8-incubating.jar:/usr/lib/libreoffice/program/classes/saxon9.jar
    org.odftoolkit.odfxsltrunner.Main
    -f net.sf.saxon.TransformerFactoryImpl -x Pictures/
    /usr/lib/libreoffice/share/xslt/export/xhtml/opendoc2xhtml.xsl pelda.odt -o pelda.xhtml
```

A letölthető Apache ODF Toolkit xslt-runner/sample_xslt alkönyvtára egyéb XSLT példákat is tartalmaz az OpenDocument állományok feldolgozására.

XSLT példa

A következő, az xsltproc és XSLT Runner programokkal is futtatható XSLT program az OpenDocument állomány minden könyvjelző elemének nevét és értékét (ha az egyszerű szöveg, vagy egy text:span elem tartalma, l. később a példa content.xml-ben és a magyarázatban) egy HTML táblázatba gyűjti ki:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0">

<xsl:template match="/">
 <html>
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
 <body>
 <h2>Könyvjelzők az OpenDocument dokumentumban</h2>
  <table>
   <tr>
    <th>Név</th>
   <th>Érték</th>
   </tr>
   <xsl:for-each select="//text:p/text:bookmark-start">
   <tr>
    <td><xsl:value-of select="@text:name"/></td>
     <td><xsl:value-of select="following-sibling::text()|following-sibling::text:span[1]"/></td>
   </tr>
   </xsl:for-each>
  </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

A művelet végrehajtása:

```
java -cp /usr/share/java/xercesImpl.jar:odftoolkit-0.5-incubating/xslt-runner-1.2.1-incubating.jar:odftoolkit-
    0.5-incubating/odva-0.8.8-incubating.jar:/usr/lib/libreoffice/program/classes/saxon9.jar
    org.odftoolkit.odfxsltrunner.Main könyvjelzők.xsl .bemenet.odt -o könyvjelzők.html
```

Például legyen a következő, két könyvjelzőt tartalmazó bekezdés az OpenDocument archívum content.xml állományában:

```
<text:p text:style-name="P1">
<text:span text:style-name="T1">Születési hely: </text:span> <text:bookmark-start text:name="születési
    hely"/>Budapest<text:bookmark-end text:name="születési_hely"/> , születési idő: <text:bookmark-start
    text:name="születési_idő"/><text:span text:style-name="T1">2013. január 1.</text:span> <text:bookmark-
    end text:name="születési_idő"/>
```

```
</text:p>
```

Ez a következő HTML táblázattá alakul az XSLT transzformáció során:

```
<html xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<body>
<h2>Könyvjelzők az OpenDocument dokumentumban</h2>
<table >
<tr>
    <td>születési hely</td>
    <td>Budapest</td>
</tr>
<tr>
    <td>születési idő</td>
    <td>2013. január 31.</td>
</tr>
</table>
</body>
```

Magyarázat: A könyvjelzők szövegtartalmát a text:bookmark-start és text:bookmark-end elemek határolják az OpenDocument formátumban (l. az előző content.xml részletet). Az XSLT példában az xmln:text attribútummal megadtuk az OpenDocument állomány feldolgozásához szükséges XML névteret, majd az xsl:template értékeként megadjuk az (X)HTML sablont. A xsl:for-each XSLT ciklussal megkeresünk minden text:p elemen belüli text:bookmark-start könyvjelző-határoló elemet, függetlenül a text:p helyétől (kezdő //). Végül a megtalált elemnek beillesszük a nevét (a text:bookmark-start elem text:name attribútuma) a HTML táblázatsor első cellájába, a másodikba pedig a text:bookmark-start elemet követő szövegtartalmat (following-sibling::text()), vagy az azt követő text:span elem (following-sibling::text:span[1]) szövegtartalmát.[19]

OpenDocument (sablon)dokumentumok módosítása

Formázott dokumentumok létrehozásának legegyszerűbb módja, ha azt sablondokumentum alapján végezzük. Ilyenkor a dokumentum szerkezetének, külalakjának kialakításához az irodai programcsomagot használjuk, a szakalkalmazásban pedig elég az így kapott sablondokumentumok kisebb részeit, például az erre kiválasztott mezőket módosítani.

A mezők módosításával nem csak kisebb szövegrészeket adhatunk meg: ki-be kapcsolhatunk rejtett szövegrészeket, teljes szakaszokat, így egy sablonban tárolhatjuk egy dokumentum több változatát is, például egy kérelem elfogadását és elutasítását.

Parancssor

A content.xml kimásolásával, szerkesztésével és ZIP archívumba visszahelyezésével könnyedén módosítható egy OpenDocument állomány:

```
unzip pelda.odt content.xml
# a content.xml módosítása, akár egy egyszerű szövegszerkesztővel
zip pelda.odt content.xml
```

Az ODF állományok egyszerűbb esetben akár parancssori segédprogramokkal, héjprogramokkal is feldolgozhatók. Egy valós példa: egyes XML entitások hibás, számmal kezdődő xml:id azonosítót kaptak a hivatalos LibreOffice kézikönyvek OpenDocument forrásában betű helyett, amely problémát jelentett a Translator Toolkit odf2xliff segédprogramjának. A következő program ezeket

[19] A LibreOffice 4-es változata a változáskövetés javítása miatt gyakran használ text:span elemeket.

a hibás attribútumokat megkeresi, és a „fix” karakterlánccal egészíti ki, javítva az ODF állományokat:

```
#!/bin/sh
case $# in
0) echo "odffix – fix ODF NCName errors for odf2xliff
Usage: odffix *.odt"; exit;;
esac
for i in $@
do
 if `unzip -p "$i" content.xml | grep -q '<[^>]*xml:id="[0-9]'`
 then
  echo "Fix $i..."
  unzip -p "$i" content.xml | sed 's/\(<[^>]*xml:id="\)\([0-9]\)/\1fix\2/g' >content.xml
  zip "$i" content.xml >/dev/null
  rm content.xml
 fi
done
```

Magyarázat: a program nem XML segédkönyvtár, hanem csak egyszerű reguláris kifejezés segítségével keresi meg, illetve javítja a hibás attribútumokat. A héjprogram a héj „if” utasításával szűri ki azokat az ODF állományokat, amelyeket módosítani kell, majd az unzip -p paraméterezéssel a szabványos kimenetre küldött content.xml-t a sed programmal javítva írja egy átmeneti állományba.

ODFpy

A következő Python 3 példaprogram az ODFpy Python programkönyvtárral megnyitja a sablon.odt szöveges állományt, mentésnek beállítja a dokumentum.odt-t, majd felülírva a sablondokumentum „név” és „szöveg” azonosítójú felhasználói mezőit, elmenti az új állományt. (Az UserFields segédosztály update metódusának paramétere egy tetszőleges mezőazonosítókat és értékeket tartalmazó szótár.)

```
from odf.userfield import UserFields
mezők = UserFields("sablon.odt", "dokumentum.odt")
mezők.update({'név': 'Tóth Gyula', 'szöveg': 'bádogos és vízvezeték-szerelő'})
```

Python

A következő, ODF vagy XML segédkönyvtárat sem használó Python megoldás az ODF (ZIP formátum és az XML állomány) közvetlen kezelésére mutat példát. A program a <text:bookmark-start/> és <text:bookmark-end/> könyvjelző határoló elemek[20] közötti részt cseréli ki a megadott szövegekre a content.xml-ben. A program közel teljes értékű megoldást[21] mutat a sablonállományok gyors módosítására, és akár a LibreOffice standard Python telepítésével is futtatható.

```
import sys, os, zipfile, re

def repbookmark(xml, repdic):
 what = '(?s)(<text:bookmark-start[^>]*text:name="(%s)".*?>).*?(?=<text:bookmark-end)' %
    '|'.join(repdic.keys())
```

[20] Az ilyen nem üres könyvjelzők készítéséhez jelöljünk ki valamilyen (helykitöltőnek szánt) szöveget a Writerben, és válasszuk a Beszúrás →Könyvjelző menüpontot.

[21] A program feltételezi, hogy az idézőjelezés és az attribútumok sorrendje kötöttebb, mint amit az XML szabvány megenged.

```
 return re.sub(what, lambda m: m.group(1) + repdic[m.group(2)], xml)

z = zipfile.ZipFile(sys.argv[1], mode='r')
f = z.open("content.xml").read().decode('utf-8')

with zipfile.ZipFile(sys.argv[2], mode='w', compression = zipfile.ZIP_DEFLATED) as dest:
 for i in set(z.namelist()) - {'content.xml'}:
  dest.writestr(i, z.read(i))
 dest.writestr('content.xml', repbookmark(f, {'név': 'Tóth', 'szöveg': 'bádogos'}).encode('utf-8'))
```

A program paramétere a bemeneti és a kimeneti ODF állomány:[22]

```
python repbookmark.py bemenet.odt kimenet.odt
```

Java

Az Apache ODF Toolkit ODFDOM,[23] és erre épülő Simple ODF API programkönyvtár[24] Java programozási felületet nyújt az OpenDocument formátumhoz, elsősorban olyan területekre fókuszálva, amelyek nem, vagy nehézkesen kezelhetők az irodai programcsomagokkal (gyakran igényelt automatizálási feladatok, például állományok szövegtartalmának másolása stílussal, vagy stílusok nélkül, táblázatok egységes formázása stb.). A programkönyvtárak honlapja számos példát is hoz ezekre.

Bemutatásképpen a következőképpen módosítjuk a sablonállomány kiválasztott felhasználói mezőit és mentjük el új dokumentumként:

```
import org.odftoolkit.simple.TextDocument;

public class Fields {
  public static void main(String[] args) {
    try {
      TextDocument doc = TextDocument.loadDocument("sablon.ott");
      doc.changeMode(TextDocument.OdfMediaType.TEXT);
      doc.getVariableFieldByName("név").updateField("Tóth Gyula", null);
      doc.getVariableFieldByName("szöveg").updateField("bádogos", null);
      doc.save("dokumentum.odt");
    } catch (Exception e) {
      System.err.println(e);
    }
  }
}
```

[22] A program magyarázata: mivel a zipfile modul nem írja felül a megnyitott ZIP formátumú ODF állományt, ezért azt le kell másolni, a külön módosítandó content.xml kivételével. A repbookmark() eljárás végzi el a repdic szótárral megadott könyvjelzőknél a szövegcserét a függvény bemeneti XML állományában. Mindezt egy darab reguláris kifejezés segítségével, amely illeszkedik az összes megadott könyvjelzőre (a repdic kulcsait, vagyis a könyvjelzők azonosítóit ehhez összefűzi a program a reguláris kifejezéshez szükséges könyvjelző1|könyvjelző2|könyvjelző3... alakban) az XML állomány sortöréseitől függetlenül is (DOTALL opció (?s) a kifejezés elején), minden illeszkedésnél egy lambda („névtelen”) függvényt meghíva (a re.sub() lehetősége, hogy nem csak adott szöveggel, hanem a paraméterként megadott függvénnyel végzi el a reguláris kifejezésre illeszkedő minták cseréjét). A lambda függvény az adott illeszkedésnél lévő kulccsal (az m.group(2) blokk, amely a (%s)-nek, azaz beillesztés után a (könyvjelző1|könyvjelző2|könyvjelző3...)-nek felel meg a forrásban) kérdezi le a bemeneti repdic szótárból a cserélendő kifejezést, így ezzel az egy re.sub() hívással az összes könyvjelző lecserélhető egy lépésben. A reguláris kifejezés kihasználja a kibővített lehetőségeket is, a nem mohó illeszkedést (*?), és az előrenéző (a mintából kihagyott) (?=...) mintaillesztést is.

[23] http://incubator.apache.org/odftoolkit/odfdom/index.html

[24] http://incubator.apache.org/odftoolkit/simple/index.html

Fordítás és futtatás:

```
javac -cp odftoolkit-0.5-incubating/odfdom-java-0.8.8-incubating.jar:odftoolkit-0.5-incubating/simple-odf-0.7-incubating.jar Fields.java
java -cp odftoolkit-0.5-incubating/odfdom-java-0.8.8-incubating.jar:odftoolkit-0.5-incubating/simple-odf-0.7-incubating.jar:/usr/share/java/xercesImpl.jar:. Fields
```

Ahogy az OTT kiterjesztésből látszik, itt valóban ODF-szövegsablon állományt nyitunk meg, amelyet aztán a changeMode metódussal ODF-szövegdokumentum formátumúvá alakítunk.

A sablonállományokat egy JAR csomagba is lehet helyezni a programmal, tovább egyszerűsítve a dokumentumok kezelését (l. később).

Parancssori ODF-elemzés

Az ODF a szabványosításnak köszönhetően kiválóan dokumentált formátum, de ha csak bizonyos funkcióira vagyunk kíváncsiak, azt közvetlenül is kileshetjük az azokat tartalmazó ODF állományokból. Különösen parancssori, XSLT, vagy az ODFpy alapú fejlesztésnél lehet célravezető a kívánt módosítások elvégzése a dokumentumszerkesztőben, majd az eredményül kapott OpenDocument állomány összehasonlítása a kiindulási állománnyal. Ezt a következő Bash héjprogrammal is elérhetjük a Linux parancssorban:

```
#!/bin/bash
echo "=== content.xml ==="
diff <(unzip -p "$1" content.xml | xml_pp) <(unzip -p "$2" content.xml | xml_pp)
echo "===  styles.xml  ==="
diff <(unzip -p "$1" styles.xml | xml_pp) <(unzip -p "$2" styles.xml | xml_pp)
```

Használata (diffodf néven elmentve, futtathatóvá, és elérhetővé téve a keresési útvonalon):

```
diffodf ures.odt ures_elvalasztas_bekapcsolva.odt
=== content.xml ===
===  styles.xml  ===
28c28,31
<    <style:style style:class="text" style:family="paragraph" style:name="Standard"/>
---
>    <style:style style:class="text" style:family="paragraph" style:master-page-name=""
     style:name="Standard">
>      <style:paragraph-properties fo:hyphenation-ladder-count="no-limit" style:page-number="auto"/>
>      <style:text-properties fo:hyphenate="true" fo:hyphenation-push-char-count="2" fo:hyphenation-remain-
     char-count="2"/>
>    </style:style>
```

A példában az elválasztást kapcsoltuk be az Alapértelmezett stílus bekezdésstílusban, és ahogy a styles.xml változásában látható, ez több járulékos tulajdonság bekapcsolásával járt együtt, amelyet a saját programnyelvi megvalósításainkban is követni érdemes (bár ahol lehet, kész sablon-, vagy sablonként használt dokumentumra bízzuk az ilyen beállításokat, és inkább csak a tartalom előállításával foglalkozzunk az alkalmazásunkból).

OpenDocument állományok létrehozása és módosítása

Python – ODFpy

Az ODFpy az OpenDocument állományok előállítására és módosítására szogáló Python könyvtár. Eredeti fejlesztője Michael Howitz, a rendszeresen frissített programkönyvtár karbantartója Søren Roug, az EU Európai Környezetvédelmi Ügynökségének munkatársa. A programkönyvtár tulajdonképpen egy burok az OASIS/ISO OpenDocument szabvány körül: lehetővé teszi, hogy pár

utasítással helyes ODF állományokat állítsunk elő, illetve módosítsunk, miközben nem kell tartanunk attól, hogy hibás elemeket vagy attribútumokat helyezünk el az XML-ben. Használatához érdemes az OpenDocument szabványt is kéznél tartani (amelynek ODF változata egyben példa az OpenDocument használatára), l. http://www.oasis-open.org/standards.

Dokumentum létrehozása

A következő program a „Szia, Világ!” sort tartalmazó OpenDocument szöveges (ODT) állományt hozza létre „helloworld” néven:

```
# -*- Encoding: UTF-8 -*-
from odf.opendocument import OpenDocumentText
from odf.text import P
textdoc = OpenDocumentText()
p = P(text = u"Szia, Világ!")
textdoc.text.addElement(p)
textdoc.save("helloworld", True)
```

Betűkészletek beállítása

A betűkészletek és az opcionálisan használt Graphite betűtulajdonságokat előre deklarálnunk kell:

```
from odt.style import FontFace
textdoc.fontfacedecls.addElement((FontFace(name='Linux Libertine G',
    fontfamily='Linux Libertine G', fontfamilygeneric="roman", fontpitch="variable")))
```

Strukturált szöveg címsorokkal

A következő programrészlet egy Címsor 1 stílusú címsort illeszt be, amit a megfelelő bekezdés- (középre igazítás) és karakterformázással (a korábban beállított betűk valamelyike félkövér betűváltozatban, piros színben, 16 pontos betűméretben) van ellátva:

```
from odf.style import Style, FontFace, TextProperties, ParagraphProperties
h1 = Style(name="Heading 1", family="paragraph")
h1.addElement(ParagraphProperties(textalign="center"))
h1.addElement(TextProperties(fontsize="16pt",
    fontweight="bold", color="#FF0000",  fontname="Linux Libertine G"))
textdoc.styles.addElement(h1)
textdoc.text.addElement(H(outlinelevel=1, text=u"Főcím", stylename=h1))
```

Java (ODFDOM, JAR)

Az előző Java ODFDOM/Simple ODF API példába (16. oldal) behelyettesítve a következő sorokat, olyan programot kapunk, amely egy „Szia, Világ!” tartalmú új ODF állományt hoz létre „szia.odt” néven:

```
TextDocument doc = TextDocument.newTextDocument();
doc.addParagraph("Szia, Világ!");
doc.save("szia.odt");
```

A következő példában alapértelmezett sablondokumentum alapján hozunk létre új állományt. A pack.ott sablonállomány a Pack.java programból fordított Pack.class állománnyal együtt egy JAR állományban kerül elhelyezésre, és onnan kerül megnyitásra az új állomány létrehozásánál. A sablon alapján létrehozott új állományban a „Szia, Világ!” szöveg előtt nincs már üres bekezdés, mint az előző példában, mivel új bekezdés helyett az alapértelmezett üres bekezdéshez fűzzük hozzá a szöveget:

```
TextDocument doc = TextDocument.loadDocument(Pack.class.getResource("pack.ott").openStream());
```

```
doc.changeMode(TextDocument.OdfMediaType.TEXT);
doc.getParagraphByIndex(0, false).appendTextContent("Szia, Világ!");
doc.save("pack.odt");
```

A Pack.java fordítása, a JAR állomány létrehozása, futtatása és az eredmény megnyitása:

```
javac -cp odftoolkit-0.5-incubating/odfdom-java-0.8.8-incubating.jar:odftoolkit-0.5-incubating/simple-odf-0.7-
    incubating.jar Pack.java
jar cfe pack.jar Pack Pack.class pack.ott
java -cp odftoolkit-0.5-incubating/odfdom-java-0.8.8-incubating.jar:odftoolkit-0.5-incubating/simple-odf-0.7-
    incubating.jar:/usr/share/java/xercesImpl.jar:pack.jar Pack
xdg-open pack.odt
```

Egyéb programnyelvek

A *http://www.opendocumentformat.org/developers/* oldal sorolja fel az egyéb ODF programkönyvtárakat, többek között az említett ODFpyt és Apache ODFToolkitet.

ODF/XForms űrlapok

A W3C XForms űrlapszabvány része az MSZ ISO OpenDocument szabványnak.
Előnyei:

– nyílt magyar szabvány (MSZ ISO OpenDocument);
– dokumentum- és összetett űrlapok kezelésére kidolgozott nyílt szabvány (W3C XForms);
– ingyenes szabad irodai programokkal (LibreOffice, OpenOffice.org) kezelhető, külön kiegészítő telepítését nem igényli;
– tömör, XML alapú űrlap-adatok online beküldési lehetősége;
– offline kitöltés, tárolás, nyomtatás.

Példa: http://books.evc-cit.info/xforms_ooo_06_08_15.odt

LibreOffice

Melyik LibreOffice változatot használjuk?

A LibreOffice követi a sikeres nyílt forráskódú közösségi programfejlesztés egyik alapszabályát, a friss programváltozatok rendszeres és gyors kiadását.[25] A kiadások menetrendje nyilvános.[26] A stabil kiadások előtt tesztelési céllal előzetes kiadásokra kerül sor (béta és RC, azaz kiadásra jelölt változatok). A megbízható működés intézményi felhasználás esetén fokozott követelmény, ezért javasolt megvárni egy LibreOffice változat első pár javító kiadását: félélévenként jelenik meg új változat, majd mintegy három hónap alatt három javító kiadás. De többet is várhatunk, hiszen a verziók élettartalma 9 hónap, de operációs rendszertől függően, mint pl. az Ubuntu LTS, ez lényegesen hosszabb is lehet.

Felhasználói profilok

A felhasználói beállítások, adatok (például saját sablonok, kiterjesztések, makrók mentési) helye a platformtól és a LibreOffice fő változatától függ, például a LibreOffice 4 esetén:

```
/home/<felhasználó>/.config/libreoffice/4/user                    (Linux)
/Users/<felhasználó>/Library/Application Support/libreoffice/4/user  (Mac OS X)
%appdata%\libreoffice\4\user                                      (Vista és újabb Windowsok)
LibreOfficePortable\Data\settings\                                (LibreOffice Portable)
```

A pontos útvonal az Eszközök→Beállítások→LibreOffice→Útvonalak lapon tekinthető meg.[27] A felhasználói profil könyvtárát a LibreOffice az első indulásnál hozza létre (törlés esetén újra létrehozza, így tesztelési céllal gyorsan visszaállítható az alaphelyzet). A felhasználó saját beállításait tartalmazó registrymodifications.xcu állománnyal, illetve a rendszerszintű és a központi hálózati beállítási lehetőségekkel későbbi szakaszok foglalkoznak.

Kötegelt dokumentumfeldolgozás (nyomtatás, átalakítás)

A LibreOffice gazdag parancssori lehetőségekkel rendelkezik, például a --show kapcsolóval lejátszható egy ODF bemutatófájl a LibreOffice grafikus felületének használata nélkül:

```
soffice --show bemutató.odp
```

A LibreOffice kézikönyvoldala (man libreoffice), vagy a LibreOffice-t indító parancssori libreoffice (soffice) parancs foglalja össze a lehetőségeket:

```
soffice -h
```

[25] Nem kiadás a fenti értelemben, de a legfrissebb újdonságokra vagy javításokra kíváncsiaknak is hasznos lehet a szabadon hozzáférhető, „percre kész” forráskód, vagy az ez alapján készült naprakész bináris telepítőcsomagok, amelyek a http://dev-builds.libreoffice.org/daily/ címen érhetők el.

[26] http://wiki.documentfoundation.org/ReleasePlan

[27] http://wiki.documentfoundation.org/UserProfile

```
LibreOffice 3.5

Usage: soffice [options] [documents...]

Options:
--minimized    keep startup bitmap minimized.
--invisible    no startup screen, no default document and no UI.
--norestore    suppress restart/restore after fatal errors.
--quickstart   starts the quickstart service
--nologo       don't show startup screen.
--nolockcheck  don't check for remote instances using the installation
--nodefault    don't start with an empty document
--headless     like invisible but no userinteraction at all.
...
```

Parancssori nyomtatás

Az OpenDocument állományok kinyomtathatók a telepített LibreOffice-szal parancssorból, vagy az ennek megfelelő rendszerhívással a szakalkalmazásból is:

```
soffice -p *.odt
soffice -pt nem_alapértelmezett_nyomtató *.odt
```

Parancssori átalakítás

A következő példában PDF formátumúra alakítja a LibreOffice a megadott OpenDocument állományokat. A második esetben a háttérben, automatikusan kilépve a LibreOffice-ból (--invisible):

```
soffice --convert-to pdf *.odt
soffice --invisible --convert-to pdf *.odt
```

Ha egyéb PDF beállításokat is szeretnénk, akkor indítsuk el a LibreOffice-t, és a Fájl→Exportálás PDF-be... ablakban állítsuk be ezeket. A LibreOffice-ból való kilépés után a beállítások megmaradnak a parancssori átalakítás számára is.

Makrók parancssori indítása

Ha nem akarunk függni a felhasználói beállításoktól a parancssorban, akkor UNO kliensprogramot készítsünk (ilyenkor a LibreOffice kiszolgáló üzemmódban fut, egyszerre akár több kérést is kiszolgálva, l. később), de telepített makrókat, egyéb programokat is indíthatunk parancssorból:

```
soffice 'macro:spelling'                 # spelling() makró indítása (szó bekérése és ellenőrzése, l. 27. oldal)
soffice 'macro:könyvtár.modul.eljárás'   # általános szintaxis
soffice 'macro:eljárás("paraméter")'     # indítás paraméter megadásával
```

XSLT és egyéb szűrők

Az Eszközök→XML szűrő beállításai... ablak segítségével új be- és kimeneti XSLT-szűrők telepíthetők a LibreOffice-ban, amellyel tetszőleges XML alapú formátumok támogatására nyílik lehetőség (l. 12. oldal).

A LibreOffice egyéb szűrőkkel is bővíthető. A LibreOffice projekt keretében készült el számos zárt szoftver zárt formátumának szűrője (Visio, CorelDraw, MS Publisher), amelyek külön is hozzáférhetők programkönyvtárként. A keretrendszer abba az irányba módosult, hogy leegyszerűsítse a további szűrők fejlesztését.[28]

[28] http://fridrich.blogspot.hu/2013/08/extending-swiss-army-knife-overview.html

Konfigurációs séma és beállítások

A LibreOffice forráskódja az officecfg/registry/schema könyvtárban tartalmazza a konfigurációs beállítások nevét és rövid leírását.[29]

A LibreOffice 4-est megelőző változatokhoz létezik olyan kiterjesztés, amellyel grafikus felületen megtekinthetők és módosíthatók ezek az opciók,[30] illetve a LibreOffice 4.2-es változata tartalmaz egy a Firefox about:config-jához hasonló beépített eszközt.

A LibreOffice központi, illetve központi hálózati konfigurációs lehetőségeiről a későbbiekben lesz szó.

Kiterjesztések kezelése

A LibreOffice-kiterjesztések (kiegészítők vagy bővítmények) olyan OXT kiterjesztésű fájlarchívumok, amelyek egyszerű esetben állományokkal: sablonokkal, képekkel bővítik az irodai programcsomagot, egyéb esetben szótárakat, makrókat, vagy Python, C, Java nyelven fejlesztett programokat tartalmaznak, amelyek valamilyen formában (menüpont, ikon, eszköztár, szűrő, valamilyen egyéb tulajdonság) integrálásra kerülnek a LibreOffice-szal. Az extensions.libreoffice.org oldalon (elérhető a Kiterjesztéskezelő ablak „További kiterjesztések letöltése…” hivatkozására kattintással is) számos LibreOffice kiterjesztés érhető el.

Kiterjesztéskezelő

Az Eszközök→Kiterjesztéskezelő… ablakkal grafikus felületen telepíthetjük, eltávolíthatjuk, vagy ideiglenesen kikapcsolhatjuk a LibreOffice kiterjesztéseket.

Unopkg

Az unopkg LibreOffice segédprogrammal az OXT kiegészítők parancssori telepítése, eltávolítása, egyéb kezelése oldható meg (l. még unopkg -h), nemcsak a felhasználói profilban, hanem az adott gép minden felhasználója számára is (rendszerszintű telepítés). Példák:

```
unopkg add példa.oxt                  # példa.oxt telepítése
unopkg list                           # telepített kiegészítők kilistázása
unopkg remove pelda_id                # példa.oxt kiegészítő eltávolítása azonosító alapján (l. a listában)
sudo unopkg --shared add példa.oxt    # telepítés minden felhasználó számára
```

Saját kiterjesztések készítése

Az extensions.libreoffice.org oldalon található szótárak, sablonállományok OXT csomagjainak megtekintésével, vagy akár módosításával egyszerűen elkezdhető a saját kiterjesztések fejlesztése, mivel ezek tartalmazzák már az elvárt metaadatokat.

Billentyűkombináció hozzárendelésére kiegészítőből

A következő példában szereplő, három állományt tartalmazó OXT (ZIP/JAR) kiegészítő telepítésével egyszerűen beállítható a Ctrl-Alt-F billentyűkombináció a lábjegyzetek közvetlen beszúrásához.

Az OXT csomag létrehozásához az állományokat a következő útvonalon hozzuk létre egy üres könyvtárban:

```
description.xml
Accelerators.xcu
```

[29] Online: http://cgit.freedesktop.org/libreoffice/core/tree/officecfg/registry

[30] L. http://www.linuxtag.org/2012/fileadmin/www.linuxtag.org/slides/Thorsten%20Behrens%20-%20LibreOffice%20configuration%20management%20-%20Tools_%20approaches%20and%20best%20practices.p331.pdf

```
META-INF/manifest.xml
```

Az OXT kiegészítő kötött description.xml állományának tartalma, amely a kiegészítő megnevezését is tartalmazza több nyelven:

```
<?xml version='1.0' encoding='UTF-8'?>
<description xmlns="http://openoffice.org/extensions/description/2006"
    xmlns:dep="http://openoffice.org/extensions/description/2006"
    xmlns:xlink="http://www.w3.org/1999/xlink">
 <identifier value="insert_footnote_keyboard_shortcut"/>
 <version value="0.1"/>
  <display-name>
    <name lang="en-US">Keyboard shortcut to insert footnotes (Ctrl-Alt-F)</name>
    <name lang="hu-HU">Gyorsbillentyű lábjegyzet beszúrásához (Ctrl-Alt-F)</name>
  </display-name>
</description>
```

A tényleges beállítást tartalmazó, tetszőleges nevű, jelen példában Accelerators.xcu állomány:

```
<?xml version="1.0" encoding="UTF-8"?>
<oor:component-data xmlns:oor="http://openoffice.org/2001/registry"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" oor:name="Accelerators"
    oor:package="org.openoffice.Office">
 <node oor:name="PrimaryKeys">
  <node oor:name="Modules">
   <node oor:name="com.sun.star.text.TextDocument">
    <node oor:name="F_MOD1_MOD2" oor:op="replace">
     <prop oor:name="Command">
      <value xml:lang="en-US">.uno:InsertFootnote</value>
     </prop>
    </node>
   </node>
  </node>
 </node>
</oor:component-data>
```

Ahol az „F_MOD1_MOD2” a jelzett billentyűkód, az .uno:InsertFootnote pedig a hozzárendelt parancs azonosítója (l. még Dispatch parancsok, illetve a LibreOffice forráskódját[31]).

Az OXT csomag extra állományait (esetenként csak az alkönyvtárakat), a példában az Accelerators.xcu-t a META-INF/manifest.xml szintén kötött nevű metaadat-állomány sorolja fel:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE manifest:manifest PUBLIC "-//OpenOffice.org//DTD Manifest 1.0//EN" "Manifest.dtd">
<manifest:manifest xmlns:manifest="http://openoffice.org/2001/manifest">
 <manifest:file-entry manifest:media-type="application/vnd.sun.star.configuration-data" manifest:full-
    path="Accelerators.xcu"/>
</manifest:manifest>
```

A csomag készítéséhez adjuk ki a következő parancsot az állományokat tartalmazó könyvtár gyökerében:

```
zip -r ~/példa.oxt .
```

A saját könyvtárunk gyökerében létrehozott OXT kiegészítő telepítése után a kiterjesztéskezelő a LibreOffice felületének megfelelő nyelven mutatja a csomagunk nevét. A Ctrl-Alt-F billentyűzet-

[31] Billentyű-hozzárendelések helye a forráskódban: officecfg/registry/data/org/openoffice/Office/Accelerators.xcu.

kombinációhoz hozzárendelődik a Lábjegyzet közvetlen beszúrása (ez az Eszközök→Testreszabás párbeszédablak Billentyűzet lapján ellenőrizhető), így nemcsak a lábjegyzetekből való gyors kilépésre (PgUp), hanem most már a beszúrásra is lehetősége lesz minden olyan felhasználónak, amely a telepített csomagot tartalmazó LibreOffice-t használja.

Sablondokumentumok, szövegblokkok a kiegészítőkben

A következő példában a kiegészítő „template” alkönyvtárába tetszőleges sablondokumentumot, az autotext alkönyvtárban pedig szövegblokkokat (szövegmintákat, „kész szöveg” néven az MS Office magyar honosításában) tartalmazó állományokat helyezünk el. Ezek változtatás nélkül települnek a felhasználói profil hasonló nevű alkönyvtáraiba, és lesznek elérhetők a LibreOffice-ban, ha mellékeljük a kiterjesztés gyökerében a következő, Paths.xcu nevű állományt:

```
<?xml version='1.0' encoding='UTF-8'?>
<oor:component-data oor:package="org.openoffice.Office" oor:name="Paths"
    xmlns:install="http://openoffice.org/2004/installation" xmlns:oor="http://openoffice.org/2001/registry"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance">
    <node oor:name="Paths">
            <node oor:name="Template" oor:op="fuse">
                    <node oor:name="InternalPaths">
                            <node oor:name="%origin%/template" oor:op="fuse"/>
                    </node>
            </node>
            <node oor:name="AutoText" oor:op="fuse">
                    <node oor:name="InternalPaths">
            <node oor:name="%origin%/autotext" oor:op="fuse"/>
                    </node>
            </node>
    </node>
</oor:component-data>
```

A Paths.xcu-hoz tartozó bejegyzés a kiterjesztés manifest.xml-jében:

```
<manifest:file-entry manifest:media-type="application/vnd.sun.star.configuration-data" manifest:full-path
    ="Paths.xcu"/>
```

Megjegyzés: a szövegblokkok .bau kiterjesztésű ZIP archívumokban találhatók. A saját szövegblokkokat a felhasználói profil autotext/mytext.bau állománya tartalmazza. Ennek átnevezésével is megoldható a kiterjesztés szövegblokk-állományainak elkészítése, de egyszerű felépítésénél fogva (az egyes szövegblokkok külön alkönyvtárakban találhatók, és ezek a gyökérben lévő Block-List.xml-ben vannak regisztrálva) akár könnyedén módosíthatóak is.

A registrymodifications.xcu vizsgálata saját konfiguráció készítéséhez

A kiegészítőkkel tetszőleges LibreOffice beállítást, testreszabást érhetünk el, de hogyan hivatkozhatunk ezekre a beállításokra? Segítséget jelent a konfigurációs kiegészítők elkészítésében a felhasználói profil gyökerében található registymodifications.xcu megtekintése, amelyben a saját beállításainkat tárolja a LibreOffice. Először készítsünk egy másolatot az állományról (helyét az Eszközök→Beállítások→LibreOffice→Útvonalak lap is mutatja), konfiguráljuk a LibreOffice-t a kívánt módon, lépjünk ki a programból, majd hasonlítsuk össze a két állományt. A következő példában a hosszú távú dokumentummegőrzésre szolgáló ISO PDF/A-1a formátumot, és a kisebb helyfoglalást eredményező JPEG-tömörítést állítottuk be a Fájl→Exportálás PDF-be... ablakban, ennek különbségét mutatja a diff (soronkénti különbség) parancs:

```
cp ~/.config/libreoffice/4/user/registrymodifications.xcu .          # másolat a helyi könyvtárba
```

```
soffice                                                    # beállítás a LibreOffice-ban és kilépés
diff registrymodifications.xcu .config/libreoffice/4/user/registrymodifications.xcu # összehasonlítás
< <item oor:path="/org.openoffice.Office.Common/Filter/PDF/Export"><prop oor:name="SelectPdfVersion"
    oor:op="fuse"><value>0</value></prop></item>
< <item oor:path="/org.openoffice.Office.Common/Filter/PDF/Export"><prop
    oor:name="UseLosslessCompression" oor:op="fuse"><value>true</value></prop></item>
---
> <item oor:path="/org.openoffice.Office.Common/Filter/PDF/Export"><prop oor:name="SelectPdfVersion"
    oor:op="fuse"><value>1</value></prop></item>
> <item oor:path="/org.openoffice.Office.Common/Filter/PDF/Export"><prop
    oor:name="UseLosslessCompression" oor:op="fuse"><value>false</value></prop></item>
```

A beállítások nevének ismeretében elkészíthető a megfelelő konfigurációs állomány:

```
<?xml version="1.0" encoding="UTF-8"?>
<oor:component-data xmlns:oor="http://openoffice.org/2001/registry"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" oor:name="Common"
    oor:package="org.openoffice.Office">
 <node oor:name="Save">
  <node oor:name="ODF">
   <prop oor:name="SelectPdfVersion">
    <value>1</value>
   </prop>
   <prop oor:name="UseLosslessCompression">
    <value>false</value>
   </prop>
  </node>
 </node>
</oor:component-data>
```

Központi beállítások

A LibreOffice telepítés share/registry könyvtára tartalmazza XML alapú, xcd kiterjesztésű állományokban az alapértelmezett központi beállításokat. Az előző példában szereplő attribútum, a SelectPdfVersion például a main.xcd állományban található:

```
grep -l SelectPdfVersion /usr/lib/libreoffice/share/registry/*
/usr/lib/libreoffice/share/registry/main.xcd
xml_grep 'prop[@oor:name="SelectPdfVersion"] SelectPdfVersion /usr/lib/libreoffice/share/registry/main.xcd
...
<prop oor:name="SelectPdfVersion" oor:nillable="false" oor:type="xs:int">
 <value>0</value>
</prop>
...
```

Bár ezek az állományok a megfelelő jogosultságok birtokában közvetlenül is módosíthatók, kiegészítőkkel, illetve azokat tartalmazó RMP vagy DEB csomagokkal kényelmesebben kezelhetők. Az utóbbi megoldás a Szabad Szoftver Kompetencia Központ által fejlesztett Remote Root standard csomagokra épülő központi menedzsment eszközével nagyobb hálózatok központi LibreOffice konfigurációját is lehetővé teszik.

Rendszerszintű beállítás kiegészítőkkel

Rendszerszinten telepítve a konfigurációt tartalmazó kiegészítőt, a gép minden felhasználója számára biztosíthatjuk ugyanazon beállításokat:

```
sudo unopkg --shared add beállítások.oxt    # telepítés minden felhasználó számára
```

Ha azt szeretnénk, hogy ezeket a beállításokat ne tudják módosítani a felhasználók, állítsuk igazra a kiegészítő konfigurációs állományában a tulajdonsághoz tartozó elem finalized attribútumát:

```
<?xml version="1.0" encoding="UTF-8"?>
<oor:component-data xmlns:oor="http://openoffice.org/2001/registry"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" oor:name="Common"
    oor:package="org.openoffice.Office">
 <node oor:name="Save">
  <node oor:name="ODF">
   <prop oor:name="SelectPdfVersion" oor:finalized="true">
    <value>1</value>
   </prop>
   <prop oor:name="UseLosslessCompression" oor:finalized="true">
    <value>false</value>
   </prop>
  </node>
 </node>
</oor:component-data>
```

Központi konfiguráció RPM és DEB alapú csomagokkal

A rendszerszintű LibreOffice share/extensions alkönyvtára alá telepített kiterjesztések RPM vagy DEB csomagokban is elhelyezhetők, ahol az operációs rendszer standard LibreOffice kiterjesztés-telepítési lehetőségével történik a kiterjesztések regisztrálása, illetve eltávolítása. Válaszszunk ki egy már kész LibreOffice-kiterjesztés csomagot (pl. libreoffice-wiki-publisher, libreoffice-pdfimport) csomagkészítési mintának.

Egyéb központi konfigurációs lehetőségek

Lehetőség van a felület beállítására,[32] és a program parancsainak letiltására is konfigurációs állományokkal,[33] vagy programból is.[34] A hivatkozott beállítási módszerek a konfigurációs állományok megfelelő könyvtárba való elhelyezésével megoldható, vagy OXT csomagok unopkg-való közvetlen telepítésével, illetve Linuxokon az említett RPM és DEB csomagkezeléssel.

Csoportházirend

Az intézményi Windows hálózati konfiguráláshoz rendelkezésre áll már a Szabad Szoftver Kompetencia Központ által fejlesztett natív csoportházirend-beállítási lehetőség is.[35]

UNO

A LibreOffice programozáshatóságának alapját az UNO (Unified Network Objects) programozási modell biztosítja. Egyfelől erre épül a LibreOffice API-ja (l. api.libreoffice.org, illetve programozási kézikönyv), másfelől ez nyújt lehetőséget saját UNO szolgáltatások hozzáadására, amelyet aztán a LibreOffice által támogatott programnyelvek (például C++, a beépített Basic és Python vagy a Java futtatókörnyezetet igénylő Java) mindegyikén kényelmesen elérhetünk.

[32] http://wiki.openoffice.org/wiki/Documentation/Administration_Guide/Customizing_the_UI
[33] http://wiki.openoffice.org/wiki/Documentation/Administration_Guide/Restricting_functionality
[34] http://api.libreoffice.org/examples/DevelopersGuide/OfficeDev/DisableCommands/DisableCommandsTest.java
[35] Előzetes összefoglaló: https://wiki.documentfoundation.org/ReleaseNotes/4.2#Windows_Registry_changes

IDL

Az IDL interfészdefiníciók írják le azokat a szolgáltatásokat, amelyeket az UNO-n belül igénybe vehetnek az alkalmazások. Az IDL állományok helye az offapi modul a LibreOffice forráskódjában, operációs rendszer szintjén a /usr/share/idl/libreoffice, de az api.libreoffice.org-on is megtekinthetők.

Például a következő LibreOffice Basic példa bekér egy szót, és ellenőrzi annak helyesírását, hiba esetén kiírva a javaslatokat (a program PyUNO változata a 32. oldalon látható). Ehhez a createUnoService() hívással eléri a SpellChecker UNO interfészt megvalósító UNO komponenst, a LibreOffice beépített helyesírás-ellenőrzőjét. Ez biztosítja a helyesírás-ellenőrzéshez az XSpellChecker interfész[36] igaz-hamis isValid() és a helyesírási javaslatokat adó spell() függvényeit is:

```
Sub spelling
Dim language As New com.sun.star.lang.Locale
language.Language = "hu"
language.Country = "HU"
word = InputBox ("Kérem a szót:")
spellchecking = createUnoService("com.sun.star.linguistic2.SpellChecker")
If Not spellchecking.IsValid(word, language, Array()) Then
  message = "Hibás szó! Javaslatok: "
  alternatives = spellchecking.spell(word, language, Array())
  If Not IsNull(alternatives) Then
    For Each i In alternatives.getAlternatives
    message = message + chr(13) + i
    Next i
  End If
  MsgBox message
End If
End Sub
```

Programnyelvi kiegészítők

A kiegészítők lehetőséget nyújtanak programkönyvtárak, saját programkód hozzáadására is, amelyek teljes mértékben támaszkodhatnak a LibreOffice-ra, UNO programozási környezetére, és vizuális, többnyelvű fejlesztést is támogató felülettervezőjére.

Basic

A LibreOffice beépített magas szintű programnyelve és programozási környezete, amely nyomkövetéssel és többnyelvű, könnyen honosítható párbeszédablakok tervezését is támogató integrált fejlesztőkörnyezettel rendelkezik. Nyelvi elemeinek és programkönyvtárainak leírását a LibreOffice beépített súgója is tartalmazza, erről és a dokumentumok programozásáról lásd a hivatkozott irodalomjegyzéket.

A következő Writer példa a „Szia, Világ!” sort szúrja be az aktuális kurzorpozíció helyére 15 pontos betűméretben és ciklámen színben (FF00FF hexadecimális RGB színkóddal megadva):

```
Sub HelloWorld
    cursor = ThisComponent.CurrentController.getViewCursor()
    cursor.CharHeight = 15
    cursor.CharColor = &HFF00FF
    ThisComponent.Text.insertString(cursor, "Szia, Világ!", FALSE)
End Sub
```

[36] http://api.libreoffice.org/docs/idl/ref/interfacecom_1_1sun_1_1star_1_1linguistic2_1_1XSpellChecker.html

Az eljárást az Eszközök→Makrók→Makrók rendezése→LibreOffice Basic ablakon keresztül megnyitott Basic szerkesztőablakban adhatjuk meg, majd az eljárásra állítva a szövegkurzort, az F5 lenyomásával indíthatjuk azt. Az eredmény az aktuális szövegdokumentumban jelenik meg.

A Figyelés ablak (és lépésenkénti indítás, F8), vagy az olyan Basic kiegészítők, mint az XRay, segítenek megismerni az objektumok típusait és tulajdonságait, illetve az általuk támogatott UNO interfészeket (a példában a „cursor” típusa SwXTextViewCursor, amelynek a CharacterProperties és a TextViewCursor UNO interfészét használtuk, l. az objektum SupportedServiceNames listáját).

Python–UNO híd (PyUNO)

A Basic makrókhoz képest a nagyobb fejlesztésekhez is kiválóan alkalmazható Python nyelvet és hatékony programkönyvtárait (reguláris kifejezések, hálózati hozzáférés) nyújtja a LibreOffice beépített Pythonja és Python-UNO hídja. Ugyan nincs integrált fejlesztői környezet, de teljes értékű UNO komponenseket fejleszthetünk vele. Példa erre az alapértelmezett magyar mondatellenőrzést is megvalósító LightProof mondatellenőrző (ProofReading UNO szolgáltatást megvalósító komponens), vagy a PyUNO saját példái[37].

Makróknak megfelelő, de Python nyelvű programkönyvtárakat is mellékelhetünk az OpenDocument dokumentumainkba és kiegészítőinkbe a PyUNo Scripting Framework segítségével.[38] A következő Python kódot tartalmazó állományt (programmodult) másoljuk a LibreOffice felhasználói könyvtár Scripts/python alkönyvtárába, a korábbi „Szia, Világ!” Basic eljárás Python megvalósításaként:[39]

```
import uno

def SziaVilág():
  doc = XSCRIPTCONTEXT.getDocument()
  cursor = doc.CurrentController.getViewCursor()
  cursor.CharHeight = 15
  cursor.CharColor = 0xFF00FF
  doc.Text.insertString( cursor, "Szia, Világ!" , 0 )

g_exportedScripts = SziaVilág,
```

Az arra kijelölt (a g_exportedScripts listának megadott) Python eljárások nemcsak az Eszközök→Makrók→Makrók futtatása... ablakban jelennek meg, hanem billentyűkombinációkhoz, ikonokhoz, menüpontokhoz rendelhetők az Eszközök→Testreszabás segítségével. Példa erre a LibreOffice 4 beépített LibreLogo oktatási és vektorgrafikai keretrendszere.[40]

A PyUNO fejlesztések nyomkövetéséhez a PYUNO_LOGLEVEL környezeti változó, (Windowson még a PYUNO_LOGTARGET) beállítására van szükség:

```
PYUNO_LOGLEVEL=ARGS soffice
```

A PyUNO hívások, argumentumok a terminálablakba, vagy a PYUNO_LOGTARGET megadása esetén a megadott, a folyamatazonosítóval (PID) megtoldott nevű naplóállományba kerülnek. A Python–UNO híd részletes leírását l. az irodalomjegyzékben megadott címen.

[37] http://cgit.freedesktop.org/libreoffice/core/tree/pyuno/demo

[38] http://www.openoffice.org/udk/python/scriptingframework/index.html

[39] A LibreOffice 4 beépített Python 3-asának köszönhetően akár ékezetes nevű eljárásokat is használhatunk.

[40] A Writer szövegszerkesztő Nézet→Eszköztárak→Logo eszköztára mutatja a Python függvényekhez rendelt ikonokat. A LibreLogo régebbi változata külön kiegészítőként is tanulmányozható:
http://extensions.libreoffice.org/extension-center/librelogo

Dispatch parancsok

A makrórögzítés (a LibreOffice-ban kísérleti) funkciót bekapcsolva[41] lehetőség van a Calc és Writer felhasználó felület parancsainak mintegy feltérképezésére, amely parancsokat a LibreOffice a „diszpécser” (DispatchHelper) interfészén keresztül hajt végre. A makrórögzítés során felvett Basic programkódból egyszerűen kiolvasható parancsok és paramétereik ismeretében nemcsak makrókban, hanem egyéb nyelveken is egyszerűsödhet a programozás, mivel nem szükséges a gyakran nagyobb előkészítést igénylő UNO interfészeknek és használatuknak utánajárni. A parancsok és leírásuk (nemcsak a Calcé és Writeré) külön is megtekinthetők a LibreOffice forráskódjában.[42]

A következő PyUNO példa a korábbi „Szia, Világ!” program dispatch megvalósítását tartalmazza, kiegészítve még egy narancssárga (FF8000) aláhúzással, és a dokumentum mentésével:

```
import uno

from com.sun.star.beans import PropertyValue as propertyValue

ctx = XSCRIPTCONTEXT.getComponentContext()
dsp = ctx.ServiceManager.createInstanceWithContext("com.sun.star.frame.DispatchHelper", ctx)

def P(name, value):
  p, p.Name, p.Value = propertyValue(), name, value
  return p

def dispatcher(s, properties = ()):
  dsp.executeDispatch(XSCRIPTCONTEXT.getDocument().CurrentController.Frame, s, "", 0, properties)

def SziaVilág2():
  dispatcher(".uno:FontHeight", (P("FontHeight.Height", 15),))
  dispatcher(".uno:FontColor", (P("FontColor", 0xFF00FF),))
  dispatcher(".uno:Underline", (P("Underline.LineStyle", 1), P("Underline.HasColor", True),
    P("Underline.Color", 0xFF8000)))
  dispatcher(".uno:InsertText", (P("Text", "Szia, Világ!"),))
  dispatcher(".uno:Save")

g_exportedScripts = SziaVilág2,
```

Ahogy a példában is látható, a parancsokat a „.uno:parancsnév” szintaxissal adjuk meg a DispatchHelper executeDispatch() metódusának. A két segédfüggvény közül a P() az elvárt propertyValue típussá alakítja a megadott argumentumazonosítót és értékét, amelyből egy sima zárójeles listát (Python tuple) képezünk a másik segédfüggvény, a dispatcher() meghívásánál.

Python eljárások hívása parancssorból

A korábban már látott parancssori makróhívásokhoz hasonlóan a Python eljárások hívására is lehetőség van. A következő program egy üres szöveges dokumentumot nyit meg, amelyhez a loadComponentFromURL() metódust használja a „private:factory/swriter” speciális URL-lel. Helyezzük el a következő állományt a felhasználói profil Scripts/python alkönyvtárában SziaVilag.py néven:

```
import uno

def SziaVilág(x):
```

[41] A makrórögzítés az Eszközök→Beállítások→LibreOffice lap Speciális (régebben Általános) lapján kapcsolható be.
[42] http://cgit.freedesktop.org/libreoffice/core/tree/officecfg/registry/data/org/openoffice/Office/UI/

```
  ctx = uno.getComponentContext()
  smgr = ctx.ServiceManager
  desktop = smgr.createInstanceWithContext( "com.sun.star.frame.Desktop", ctx)
  doc = desktop.loadComponentFromURL( "private:factory/swriter", "_blank", 0, () )
  cursor = doc.CurrentController.getViewCursor()
  cursor.CharHeight = 15
  cursor.CharColor = 0xFF00FF
  doc.Text.insertString( cursor, "Szia, Világ!" , 0 )
```

A SziaVilág() eljárás meghívása a parancssorban:

```
soffice 'vnd.sun.star.script:SziaVilag.py$SziaVilág?language=Python&location=user'
```

A további hívási lehetőségekről (pl. a Scripts/python alkönyvtáraiban is elhelyezhetők a kódok, ilyenkor a | jellel elválasztva adhatjuk meg az útvonalat, illetve a kiegészítők Python moduljai is elérhetők) a Scripting Framework URI specifikáció ad leírást.[43]

Automatikus fájlkonverzió parancssorból

Gyakran igényelt feladat a kötegelt PDF, vagy egyéb dokumentumátalakítás. A következő példaprogram tetszőleges PDF, és egyéb opció megadását is lehetővé teszi a parancssorból (szemben a LibreOffice parancssori opcióival).

Megjegyzés: a program futásához lépjünk ki a futó LibreOffice-ból, mivel a fő program környezeti változóin keresztül adjuk meg az átalakítás paramétereit. (A program alapján elkészíthető azt ezt nem igénylő kliensprogram is, lásd később a LibreOffice kiszolgáló üzemmódban való futtatásánál).

```
import uno, unohelper, os
from com.sun.star.beans import PropertyValue as propertyValue

def P(name, value):
  p, p.Name, p.Value = propertyValue(), name, value
  return p

def typ(t):
  if t[1] in ["True", "False"]:
    return P(t[0], eval(t[1]))
  if t[1].isdigit():
    return P(t[0], int(t[1]))
  return P(t[0], t[1])

def convert(x):
  ctx = uno.getComponentContext()
  smgr = ctx.ServiceManager
  desktop = smgr.createInstanceWithContext( "com.sun.star.frame.Desktop", ctx)
  extension = os.getenv("EXT") or "pdf"
  for i in os.getenv("FILES").split():
    if not os.path.isabs(i):
      i = os.path.join(os.getenv("PWD"), i)
    doc = desktop.loadComponentFromURL( unohelper.systemPathToFileUrl(i), "_blank", 0,
            (P("Hidden", True), ) )
    args = (P("FilterName", os.getenv("FILTER") or "writer_pdf_Export"),)
    if os.getenv("ARGS"):
```

[43] http://wiki.openoffice.org/wiki/Documentation/DevGuide/Scripting/Scripting_Framework_URI_Specification

```
    data = tuple([typ(i.split("=")) for i in os.getenv("ARGS").split(";")])
    args += (P("FilterData", uno.Any("[]com.sun.star.beans.PropertyValue", data)),)
doc.storeToURL(unohelper.systemPathToFileUrl(os.path.splitext(i)[0] + "." + extension), args)
doc.close(True)
```

A [LibreOffice telepítés]/share/Scripts/python/convert.py útvonalon elhelyezett program hívásnál az ARGS környezeti változóban soroljuk fel a bemenő paramétereket pontosvesszővel elválasztva, és a FILES környezeti változóban szóközzel vagy új sor karakterrel az átalakítandó állományokat (a program elfogadja még a FILTER környezeti változót is, ha az alapértelmezett szövegesdokumentum-PDF szűrő helyett mást szeretnénk alkalmazni. Ha az nem PDF-re alakít, akkor az EXT környezeti változóval adhatunk meg fájlnévkiterjesztést). A következő példában egy bemeneti állományt alakítunk a megadott szöveget tartalmazó vízjeles PDF-fé:

```
ARGS='Watermark=Vízjel!' FILES=bemenet.odt soffice 'vnd.sun.star.script:convert.py$convert?
    language=Python&location=share'
```

Látható, hogy a hívás végén található „location” attribútum értéke most share, a rendszerszintű telepítésnek megfelelően.

A következő hívásnál teljes képernyőn megnyíló, a képeket kisebb felbontásra cserélő átalakítást végzünk az aktuális könyvtár és alkönyvtárainak minden ODT állományában:

```
ARGS='UseLosslessCompression=False;Quality=80;ReduceImageResolution=True;MaxImageResolution=75;Op
    enInFullScreenMode=True' FILES="`find -name '*.odt'`" soffice 'vnd.sun.star.script:PDF.py$convert?
    language=Python&location=share'
```

A PDF szűrő opcióiról részletes dokumentáció érhető el.[44]

Megjegyzés: a programban a loadComponentFromUrl() segítségével nyitjuk meg az állományokat, méghozzá a „Hidden” (rejtett) paraméterrel, hogy a háttérben, a helyesírás- és nyelvi ellenőrzés nélkül kerüljenek betöltésre. A storeToURL() metódus paraméterlistájában a FilterName szolgál a szűrő megadására, a FilterData pedig egy további paraméterlistát ad meg. Az összetett adatszerkezet létrehozására az uno.Any() metódus segítségét vesszük igénybe.

Developer’s Guide, Java példák

Angol nyelvű fejlesztői kézikönyv, részletes Java programnyelvi példákkal (l. irodalomjegyzék).

LibreOffice SDK

A C++-os és Java LibreOffice-kiegészítők fejlesztését teszi lehetővé a LibreOffice SDK (Software Development Kit). Ez előáll a LibreOffice fordításánál, de fordítás nélkül is hozzáférhetünk: Windows alá a hu.libreoffice.org oldalról tölthető le naprakész változat, Linux környezetben a Linux-terjesztés is biztosítja, például Ubuntu Linuxon:

```
sudo apt-get install libreoffice-dev
```

Jó példa a LibreOffice SDK Java programnyelvi használatára a Wollmux fejlesztés.

Új Calc függvények

A saját táblázatkezelő-függvények hozzáadásának feltétele, hogy az új függvények IDL metanyelvi függvényleírását bináris változatra kell lefordítani, és a Calc OXT kiegészítőben mellékelni. A fordítást egyszer kell elvégezni a LibreOffice SDK-ban. Saját táblázatkezelő függvényeket tartalmazó kiegészítő készítésére, beleértve az IDL leírást és fordítást is, példa a magyar fejlesztésű, Python alapú Numbertext kiegészítő.[45]

[44] https://wiki.openoffice.org/wiki/API/Tutorials/PDF_export
[45] http://extensions.libreoffice.org/extension-center/numbertext-1

LibreOffice kiszolgáló

Hálózati kiszolgáló

A LibreOffice hálózati (socket) kiszolgálóként is indítható, a következő példában a 42 424-es számú kapun várakozva:

```
soffice "--accept=socket,host=localhost,port=42424;urp;StarOffice.ServiceManager" --headless --nologo
    --nofirststartwizard &
```

Ellenőrizzük le az elindított szolgáltatást a futó folyamat kilistázásával:

```
ps ax | grep soffice
10599 pts/8    Sl     0:00 /usr/lib/libreoffice/program/soffice.bin
    --accept=socket,host=localhost,port=42424;urp;StarOffice.ServiceManager --headless --nologo
    --nofirststartwizard
```

A következő Python kliens a PyUNO uno modul segítségével veszi a kapcsolatot a futó LibreOffice-szal, majd végzi el a programnak megadott szó magyar helyesírás-ellenőrzését. Ha a szó hibás az ellenőrző szerint, akkor kilistázza a szóra adott javaslatokat:

```
# -*- Encoding: UTF-8 -*-
import uno, sys, socket # socket csak Windowson kell
from com.sun.star.lang import Locale
# kapcsolódás a futó LibreOffice-hoz
localContext = uno.getComponentContext()
resolver = localContext.ServiceManager.createInstanceWithContext("com.sun.star.bridge.UnoUrlResolver",
    localContext )
ctx = resolver.resolve( "uno:socket,host=localhost,port=42424;urp;StarOffice.ComponentContext" )
# Helyesírás-ellenőrző UNO komponens példányosítása , az UNO szolgáltatás nevével
spellchecker = ctx.ServiceManager.createInstanceWithContext("com.sun.star.linguistic2.SpellChecker",
    localContext)
language = Locale("hu", "HU", "")
if not spellchecker.isValid(sys.argv[1], language, ()):
  print("Hibás szó! Javaslatok:")
  alternatives = spellchecker.spell(sys.argv[1], language, ())
  if alternatives:
    for i in alternatives.getAlternatives():
      print(i)
```

A helyes.py futtatása (a háttérben futó LibreOffice elérhetősége esetén):

```
python helyes.py tanusitvány
Hibás szó! Javaslatok:
tanúsítvány
```

Megjegyzés: az uno Python modul elérhető a rendszerszintű LibreOffice-okhoz Linuxokon, egyéb esetben pedig a LibreOffice telepítés saját Python futtató környezetét indítsuk el, például egy külön telepített LibreOffice esetében:

```
/opt/libreoffice4.2/program/python ügyfél.py
```

Megjegyzés: az UNO interfész neve mellett az interfészt implementáló UNO komponens nevével is példányosíthatunk, ami akkor lehet hasznos, ha több komponens is nyújtja ugyanazt a szolgáltatást, és ezek közül pontosan meg szeretnénk határozni, melyikre van szükségünk. Az előző példa megfelelő sorát tehát írhatjuk az implementáció nevével is (amely a LibreOffice alapértelmezett Hunspell helyesírás-ellenőrzőjéhez tartozik):

```
spellchecker = ctx.ServiceManager.createInstanceWithContext("org.openoffice.lingu.MySpellSpellChecker",
```

```
    localContext)
```

Unix cső

Gyorsabb, de csak helyi hozzáférést nyújtó üzemmód az unixos cső kommunikációs csatorna:

```
soffice --accept="pipe,name=addtemppipe;urp;StarOffice.ServiceManager" --headless --nologo –
    nofirststartwizard &
```

Ehhez tartozó elérés:

```
ctx = resolver.resolve( "uno:pipe,name=addtemppipe;urp;StarOffice.ComponentContext" )
```

Wollmux dokumentum- és űrlapkezelő keretrendszer

A Wollmux keretrendszer legnagyobb kiépítésében München önkormányzatának 15 ezer munkaállomásán biztosít optimalizált ODF alapú munkakörnyezetet Windows és Linux platformon, illetve lehetővé tette mintegy 20 ezer dokumentumsablon, űrlap és makró migrációját a korábbi zárt irodai programcsomagról.[46] A rendszer Java alapú űrlap- és dokumentumkitöltőt kínál a LibreOffice mellé az ügykezelés felgyorsítására. Makró és Java alapú bővíthetőségével a munkafolyamatok egyszerű automatizálását teszi lehetővé. Honosított változata a Szabad Szoftver Kompetencia Központ honlapján érhető el.[47]

[46] http://www.wollmux.net és http://code.google.com/p/wollmux/.

[47] A jegyzet írásának idején a honosítás tesztelése zajlik, a honosítás 2013. végén lesz elérhető.

LibreOffice-fejlesztés

A LibreOffice közösségi fejlesztés, nemcsak a felhasználáshoz, hanem a fejlesztéshez is széles támogatást nyújtva, például közvetlenül kommunikálhatunk a fejlesztőkkel a nyilvános IRC csatornán. Érdemes a saját fejlesztéseinket is minél előbb megosztani a közösséggel, hogy segítséget kapjunk annak ellenőrzéséhez, javításához és továbbfejlesztéshez.

Hibabejelentés

A visszajelzés, hibabejelentés a szabad szoftverek közösségi fejlesztési modelljének alapvető része. Ha LibreOffice hibával találkozunk, a hibát mindenképpen jelentsük be! A hibák gyakran a hiányzó hibabejelentés miatt nem kerülnek idejekorán javításra. A visszalépést jelentő (regresszió) hibák javítása kiemelt prioritást élvez, így az ilyeneket jelöljük meg a „regression” kulcsszóval is (Keywords: szakasz a FreeDesktop.org hibabejelentőben, l. később).

Hibák bejelentésénél jelöljük meg annak a legrégebbi LibreOffice változatnak a számát, amelynél már tapasztaltuk a hibát. A verziószám minimum háromjegyű, a hibajavító változat verziószámát is tartalmazó szám legyen. Ezt a „Súgó→A LibreOffice névjegye” üzenetablakból olvashatjuk le, de a 4-es változattól a parancssorban is lekérdezhető:

```
soffice --version
LibreOffice 4.1.3.2 70feb7d99726f064edab4605a8ab840c50ec57a
```

Adjunk meg minél több hasznos információt és segítséget a hibabejelentésben: az operációs rendszert és verziószámát, képernyőképeket, tesztállományokat, tesztelési útmutatót!

https://www.libreoffice.org/get-help/bug/

A LibreOffice angol nyelvű hibabejelentő tündére. Egyszerű felületet nyújt a FreeDesktop.org-on található LibreOffice hibakezelő rendszerhez.

http://bugs.libreoffice.org/

A LibreOffice hivatalos, angol nyelvű hibabejelentő oldala. Nemcsak hibák bejelentésére, hanem keresésére szolgál. (Sőt, az ismételt hibabejelentés elkerülése érdekében már a hibajegy létrehozásánál automatikusan felmutatja a hasonló tárgyú hibákat. Ezeket vizsgáljuk meg, és azonos hiba esetén inkább ott írjuk le tapasztalatainkat.)

http://www.openscope.org

A magyar honosítók hibabejelentő oldala. Bár elsősorban honosítási hibák bejelentésére szolgál, lehetőség van egyéb hibák bejelentésére is.

Felhasználói támogatás

A LibreOffice, illetve a mögötte álló Document Foundation számos szolgáltatással támogatja a felhasználókat és a fejlesztőket is, de elérhető magyar nyelvű, LibreOffice-szal is foglalkozó, illetve

a LibreOffice használatában is segítséget jelentő OpenOffice.org fórum, illetve OpenOffice.org levelezőlista is.

hu.libreoffice.org

A LibreOffice magyar nyelvű honlapja, letöltési lehetőséggel, és tájékoztató oldalakkal.

www.libreoffice.hu

A LibreOffice magyar nyelvű híroldala, letölthető kézi- és tankönyvekkel, a hírekhez kapcsolódó kérdezési lehetőséggel.

forum.openoffice.org/hu/forum/

OpenOffice.org és OpenOffice.org-ra épülő irodai programcsomagok (Apache OpenOffice, LibreOffice) magyar nyelvű fóruma és kapcsolódó levelezőlistájának honlapja.

ask.libreoffice.org

A LibreOffice problémaorientált közösségi támogató oldala. Várhatóan magyar nyelvű változata is elindul a közeljövőben.

extensions.libreoffice.org

Szabad licencű kiegészítők a LibreOffice-hoz.

templates.libreoffice.org

Szabad licencű sablonok a LibreOffice-hoz.

Fejlesztői támogatás

wiki.documentfoundation.org

A LibreOffice fejlesztésével kapcsolatos, rendszeresen frissített dokumentációk helye.

IRC

A chat.freenode.net IRC kiszolgáló #libreoffice-dev csatornája központi helye a LibreOffice fejlesztésének. A főállású LibreOffice fejlesztőkön kívül fejlesztési feladatokon dolgozó hallgatókat, LibreOffice-támogatást nyújtó cégek informatikusait, és mivel a csatornához bárki csatlakozhat, érdeklődőket találni itt, bővebben: http://www.libreoffice.org/get-help/irc-help/.

cgit.freedesktop.org/libreoffice

A LibreOffice és a kapcsolódó fejlesztések GIT forrásfáját megjelenítő, egyszerű kereséseket, letöltést lehetővé tévő oldal.

gerrit.libreoffice.org

A LibreOffice programfoltok (patch) közösségi ellenőrzését, elfogadását, javítását szolgáló felület.

opengrok.libreoffice.org

Villámgyors keresés a forráskódban, a „git grep” parancs forráskód-letöltést nem igénylő alternatívája.

api.LibreOffice.org

UNO API és más fejlesztési dokumentációk helye.

translations.documentfoundation.org

A LibreOffice felületének, súgójának közösségi fordítási oldala (Pootle-kiszolgáló).

LibreOffice fejlesztési példa

A LibreOffice fejlesztését részletesen ismerteti az említett hivatalos wiki. A Szabad Szoftver Konferencia és Kiállítás több magyar nyelvű előadása és követő kiadványainak publikációi is foglalkoznak a témával (l. irodalomjegyzék), de érdemes megmutatni pár lépést, a következőkben Linux környezetben.[48]

Fordítás

Első lépés a fordításhoz szükséges környezet kialakítása:

```
sudo apt-get build-dep libreoffice # csak Debian/Ubuntu esetén
sudo zypper si -d libreoffice # csak OpenSUSE 11.4+ esetén
sudo yum-builddep libreoffice # csak Fedora 15+ és az erre épülő Linux-terjesztések esetén
```

Majd a git forrásfából való letöltés és fordítás:[49]

```
git clone git://anongit.freedesktop.org/libreoffice/core libreoffice
cd libreoffice
./autogen.sh
make
```

A fordítás több órát vesz igénybe (a későbbiekben egy-egy módosított modul újrafordítása pillanatok alatt megvan). A lefordított LibreOffice indítási útvonalát a make parancs üzenetben írja ki. Ez az útvonal a LibreOffice 4.2-től:

```
instdir/program/soffice
```

Módosítás, újrafordítás

A LibreOffice nyílt forráskódú programként szabadon kiegészíthető, javítható. Sokan egy-egy sor módosításával kerülnek be a LibreOffice fejlesztők közé, mutatva, hogy a fejlesztési problémák megoldása nagyon kevés változtatással is járhat a forráskódban. Ha nem kívánjuk igénybe venni a fejlesztők segítségét a libreoffice-dev levelezőlistán vagy IRC-n, az opengrok.libreoffice.org keresőszolgáltatás segíthet megtalálni azt a programrészt, amelyet módosítani szeretnénk. Előnyt jelent, ha angol változatban is telepítjük a LibreOffice-ot, és annak felületén keressük meg hiba, vagy továbbfejlesztés helyéhez kapcsolódó üzeneteket, feliratokat, amelyekre rá fogunk keresni (az angol üzenetek, szemben a magyar és egyéb honosított üzenetekkel, a forráskódban is szerepelnek, nem csak erőforrás-állományokban). Használható a git grep parancs is a forráskódban való keresésre:

```
git grep Hungarian
editeng/source/misc/svxacorr.cxx:   // Finnish and Hungarian use enDash instead of emDash.[50]
```

Érdemes lehet a gitk segédprogrammal felderíteni, milyen számunkra érdekes javítások kerültek be korábban a forrásfába, mert az akár kész megoldással is szolgálhat (a gitk grafikus felületen kényelmes keresési és megjelenítési lehetőséget nyújt):

```
gitk
```

A változtatásaink után fordítsuk le a módosított modult, azaz alkönyvtárat (a példában a Writer dokumentumszerkesztő fő modulját), és már indíthatjuk is újra a LibreOffice-t:

```
make sw
```

[48] http://wiki.documentfoundation.org/Development/BuildingOnLinux

[49] Kihasználva a rendelkezésre álló processzormagokat és a párhuzamos futtatás lehetőségét, de egyéb optimalizáció nélkül. Érdemes a későbbiekben ccache gyorsítótárat beállítani, hogy a teljes LibreOffice lefordítása lényegesen rövidebb idő alatt menjen végbe, l. a LibreOffice wikit.

[50] A példa a magyar fejlesztői közösség egyik első foltjára keres rá, a két kötőjel nagykötőjelre (–) való cseréjére az alapértelmezett kvirtmínusz (—) helyett szövegszerkesztés közben.

```
instdir/program/soffice
```

Foltkészítés

A programfolt tartalmazza a fejlesztésünket, amelyet a példában e-mail-csatolmányként fogunk végleges formába hozni (a javasolt gerrithez képest ez nem igényel külön beállítást).

Ellenőrizzük a változtatásainkat a paranccsal, majd tegyük be a tárolóba. A git commit elindít egy szövegszerkesztőt, ahol adjuk meg a folthoz a megfelelő üzenetet (l. 3. sor, amely a hivatalos freedesktop.org-os hibabejelentés számát is tartalmazza):

```
git diff
git commit sw
fdo#71645 remove footnote numbers in cross references
```

Ellenőrizzük a változást és mentsük el a foltot e-mail fejléces diff -u formátumban:

```
git log
git format-patch HEAD~1
less 0001-fdo-71645-footnote-numbers-in-cross-references.patch
```

Az állományt küldjük be a libreoffice-dev levelezőlistára. Több folt esetén érdemes gerrit hozzáférést beállítani, és foltunkat a gerrit.libreoffice.org oldal segítségével nyomon követni és kezelni.

SAL_DEBUG

A LibreOffice C++ forráskódjának beépített SAL_DEBUG makrójával egyszerűen nyomon követhetők a változtatásaink, illetve felderíthető a programrészlet működése, meggyorsítva a folt elkészítését. Példa a „value” változó értékének kiírására:

```
SAL_DEBUG("Valamilyen adat: " << value);
```

A modul újrafordítása után az üzeneteket a parancssorban elindított LibreOffice kiírja a terminálablakba.

GDB

Nyomkövetéshez, hibák elemzéséhez elindíthatjuk a GNU debuggerben is (gdb) a LibreOffice-t (akár grafikus felületű segédprogramban is, lásd a részletes leírást):[51]

```
make debugrun
run --writer                    # a gdb-ben kiadott parancs
backtrace                       # programleállás esetén a hívási vermet kiíratő gdb parancs
```

Git

Egyéb hasznos git parancsok:

```
git pull                        # helyi git tároló frissítése a központi módosításokkal
git pull -r                     # ugyanez a saját változtatások megőrzésével (utólag is kiadható)
git checkout állomány           # az állományban meglévő változtatásaink elvetése
git reset –hard origin/master   # minden változtatás törlése
git branch                      # aktuális ág nevének kiírása
git checkout libreoffice-4-2 # másik ágra váltás
git checkout master             # vissza a fő ágra
git apply 0001-fdo-71645-footnote-numbers-in-cross-references.patch      # folt alkalmazása
git cherry-pick [commit_azonosító] # adott commit átemelése az aktuális ágba
```

[51] https://wiki.documentfoundation.org/Development/How_to_debug

Ajánlott irodalom

LibreOffice

Fejlesztői wikioldalak

https://wiki.documentfoundation.org/Development

Developer's Guide

Az eredeti OpenOffice.org Developer's Guide Java programnyelvi példái részei a LibreOffice forráskódjának. Összefoglaló oldal:
http://api.libreoffice.org/examples/DevelopersGuide/examples.html
A kézikönyv elérhetősége:
http://wiki.openoffice.org/wiki/Documentation/DevGuide/OpenOffice.org_Developers_Guide

Makróprogramozás

A makró- (Basic) programozáshoz nem feltétlenül szükséges a Developer's Guide Java példáit és a LibreOffice UNO API-ját áttanulmányozni, mivel az alábbi oldalak számos működő Basic makrót tartalmaznak:

- A LibreOffice súgó magyar nyelvű bevezetője a beépített Basic programozásába:
  https://help.libreoffice.org/Basic/Basic_Help/hu
- Magyar nyelvű példa egyszerű dokumentumkezelő makrókra (makrórögzítés eredményének módosítása, hozzárendelése billentyűkombinációhoz Testreszabás segítségével, táblázatok formázása): *Kiadványkészítés LibreOffice Writer szövegszerkesztővel*, Makrók készítése fejezet.
  http://numbertext.org/libreoffice
- LibreOffice wikioldal, hivatkozással A. Pitonyak Andrew's Macros c., szabadon letölthető kézikönyvére, illetve a programozást támogató XRay segédprogramra:
  https://wiki.documentfoundation.org/Macros
- Az angol nyelvű OpenOffice.org fórum oldalai (csak adjuk a keresőkifejezéseinkhez a következő szavakat is: *„macro openoffice forum”).*

Python UNO

Dokumentáció: http://www.openoffice.org/udk/python/python-bridge.html

Kiegészítők

http://wiki.openoffice.org/wiki/Extensions_best_practices

Publikációk magyar nyelven

- Karay Tivadar: *Áttörni a falat (Szabad szoftverek egy önkormányzatban)*, Szabad Szoftver Konferencia és Kiállítás, követő kiadvány, 2011, 67–75 o.
  http://konf.fsf.hu/2011/szszk2011_koveto_kiadvany_v_2.1.pdf

– Németh László: *A követő kiadvány margójára – ODM fődokumentumok Writerrel*, Szabad Szoftver Konferencia és Kiállítás, követő kiadvány, 2012, 33–39. o., http://konf.fsf.hu/2012/szabad_szoftver_konferencia_2012_koveto.pdf
– Németh László: *LibreOffice – úton a DTP felé*, Szabad Szoftver Konferencia és Kiállítás, követő kiadvány, 2011, 113–122. o., http://konf.fsf.hu/2011/szszk2011_koveto_kiadvany_v_2.1.pdf
– Pénzes Dávid: *LibreOffice a Neveléstudomány szolgálatában*, Szabad Szoftver Konferencia és Kiállítás, követő kiadvány, 2012, 63–68. o., http://konf.fsf.hu/2012/szabad_szoftver_konferencia_2012_koveto.pdf
– Szegfű László: *Nyílt forráskódú megoldások a közigazgatásban*, Szabad Szoftver Konferencia és Kiállítás, követő kiadvány, 2011, 129–136. o., http://konf.fsf.hu/2011/szszk2011_koveto_kiadvany_v_2.1.pdf
– Tímár András: *LibreOffice*, Szabad Szoftver Konferencia és Kiállítás, követő kiadvány, 2011, 137–143. o., http://konf.fsf.hu/2011/szszk2011_koveto_kiadvany_v_2.1.pdf
– Tímár András: *LibreOffice*, Szabad Szoftver Konferencia és Kiállítás, követő kiadvány, 2012, 91–97. o., http://konf.fsf.hu/2011/szszk2011_koveto_kiadvany_v_2.1.pdf
– Vajna Miklós: *Hogyan készítsünk Writer funkciókat?*, Szabad Szoftver Konferencia és Kiállítás, követő kiadvány, 2012, 99–104. o., http://konf.fsf.hu/2011/szszk2011_koveto_kiadvany_v_2.1.pdf
– Vajna Miklós: *RTF támogatás a LibreOffice Writer programban*, Szabad Szoftver Konferencia és Kiállítás, követő kiadvány, 2011, 153–159. o., http://konf.fsf.hu/2011/szszk2011_koveto_kiadvany_v_2.1.pdf
– Vajna Miklós: *Verziókezelés a Git használatával*, Szabad Szoftver Konferencia és Kiállítás, követő kiadvány, 2009, http://konf.fsf.hu/2009/szszkonf2009_konferencia_kiadvany.pdf

Előadások

Vajna Miklós LibreOffice fejlesztő előadásai: http://speakerdeck.com/vmiklos
http://konf.fsf.hu/cgis/ossc/2011/speakers
http://www.youtube.com/playlist?list=PLc7xbjN4tS2SE69uaowmPVbVu2es3ERFk

OpenDocument

Szabvány

Az OASIS szabványosító testület honlapjáról szabadon letölthető OpenDocument szabványok:
https://www.oasis-open.org/standards#opendocumentv1.2
https://www.oasis-open.org/standards#opendocumentv1.1
https://www.oasis-open.org/standards#opendocumentv1.0

Kézikönyv

J, D. Eisenberg: *OASIS OpenDocument Essentials. http://books.evc-cit.info/*, 2006

A könyv XML-orientált bevezető az OpenDocument felépítésébe, illetve függelékében bemutatja az OpenDocument feldolgozására is alkalmas XSLT-t. A honlapon egy LibreOffice-hoz is jól használható XForms útmutató is elérhető.

Programkönyvtárak

http://www.opendocumentformat.org/developers/

Változat: 2013-12-10.
Készítette: Németh László
Lektorálta: Tímár András


MAGYARY
PROGRAM


MAGYARORSZÁG MEGÚJUL
A projekt az Európai Unió támogatásával, az Európai
Regionális Fejlesztési Alap társfinanszírozásával valósul meg.