OpenACS-lapok létrehozása

Reuven bemutatja, hogyan fejlesszünk APM-mel saját web- és adatbázis-alkalmazásokat.

A múlt hónapban folytattuk OpenACS-felfedező körutunkat és megnéztük, hogyan kell az alkalmazásokat csomagokba rendezni az APM (ArsDigita Package Manager) segítségével. Minden OpenACS-alkalmazás általában adatbázistáblákat és kiszolgálóoldali programokat tartalmaz, következésképpen a csomagok telepítése és frissítése a fájlok egyszerű másolásával nem oldható meg. Mint a múlt hónapban láthattuk, az APM alkalmazása leegyszerűsíti a csomag telepítését és különféle címek alá történő helyezését.
Ez nagyszerű dolog, amennyiben minket csak a már létező APM-ek érdekelnek. A legtöbb OpenACS-telepítés azonban új, egyedi csomagokat is tartalmaz, amelyek saját adatmodellel és programmal bírnak. Bár elméletileg APM nélkül is lehetséges OpenACS-alkalmazásokat fejleszteni, ez meglehetősen megnehezíti programunk terjesztését, a változatkövetést vagy a telepítés menetének szabványosítását.
Ebben a hónapban azzal foglalkozunk, hogyan fejleszthetünk saját web-, illetve adatbázis-alapú alkalmazásokat az APM segítségével. A végeredmény egy olyan alkalmazás lesz, amit bárki be tud tölteni a saját OpenACS-rendszerébe.

Vázcsomag létrehozása

Az új OpenACS-alkalmazás létrehozásának első lépése egy új vázcsomag készítése a webalapú APM programmal. Alapértelmezés szerint ez a program az OpenACS-rendszereken csak a rendszergazdai jogosultságokkal rendelkező felhasználók számára érhető el, így elképzelhető, hogy a fejlesztési munka megkezdése előtt meg kell kérnünk a rendszergazdát, módosítsa a jogosultságainkat.
A legtöbb OpenACS-rendszeren az APM program a */**acs-admin**/**apm**/* URL alól indul. A program az összes telepített APM-et bemutatja, megjelenítve valamennyi nevét, változatát és az alkotó fájlok számát. A fájlszámlálásba általában az ***.sql***- (adatbázis-meghatározások létrehozására és eltávolítására használt) fájlok, a ***.tcl***- (a Tcl programkódot tartalmazó) fájlok, az ***.adp***- (ASP vagy JSP-szerű websablon) fájlok és a ***.xql***- (SQL-lekérdezéseket tartalmazó) fájlok számítanak bele. Természetesen maga az APM más fájlokat, képeket, szöveget vagy valami szokatlanabb dolgot is tartalmazhat, például Flasht.
Mint azt a múlt hónapban láthattuk, ezt a képernyőt használhatjuk kezdőképernyőként rendszerünk csomagjainak telepítésénél, vizsgálatánál és módosításánál. Ha azonban a lap aljára megyünk, és a ***create a new package*** hivatkozásra kattintunk, elkezdhetjük új alkalmazásunk létrehozását.
A ***create a package*** kezdőképernyője számos jellemző megadását kéri tőlünk, amelyek a csomagokat leíró ***.info*** fájl létrehozásában segítik az APM-et. Feltételezve, hogy mi csak egy ***Szia, világ*** alkalmazást szeretnénk a lehető legkevesebb munkával előállítani, csak kevés mezőt kell ténylegesen kitöltenünk:

- A csomagkulcs (package key) legyen `atf-szia`. Az APM-ek esetében nincs kötelező névtérszerkezet, de a legtöbb fejlesztő saját nevet (vagy hasonló azonosító jelet) használ a csomag neve előtt az – esetleges ütközéseket elkerülendő.
- A csomag neve lehet ***Szia*** vagy tetszés szerint bármi más; a fejlesztők ezt használják arra, hogy a sajátjukat a többi csomagtól megkülönböztessék.
- Alkalmazást (application), és nem szolgáltatást (service) fejlesztünk.
- A változatszáma legyen 0.1d, jelezve, hogy a fejlesztés korai szakaszában járunk.
- Az összefoglalás (summary) és a leírás (description) mezőket tetszés szerint kitölthetjük vagy üresen hagyhatjuk. Természetesen valós alkalmazás készítésekor nem árt ezeket a mezőket is kitöltenünk.

Ha végeztünk, győződjünk meg róla, hogy a ***write a package*** (csomagírás) jelölőnégyzet be van-e kapcsolva, majd kattintsunk a ***create package*** (csomag létrehozása) gombra a lap alján. A megfelelő lap kezelőlapjára fogunk jutni. Ha belepillantunk az OpenACS-eszközkészlet telepítési útvonalának ***packages*** alkönyvtárába, ott egy érvényes XML formátumban íródott ***atf-szia.info*** fájlt tartalmazó ***atf-szia*** könyvtárat fogunk találni.

Adatmodell létrehozása

Most, hogy az OpenACS már felismeri a csomagunkat, elkezdhetünk dolgozni lapunk adatmodelljén. Mint azt minden tapasztalt web- és adatbázis-fejlesztő tudja, a táblák megtervezése a feladat legjava, az adatot törlő, létrehozó és módosító alkalmazások megírása már meglehetősen egyszerű.
Alkalmazásunk egy egyszerű vendégkönyv lesz, ahol a látogatók rögzíteni tudják a lappal kapcsolatos észrevételeiket. (Az OpenACS már eleve rendelkezik hasonló, de sokkal hatékonyabb és kifinomultabb megoldással, ami a honlap bármely lapján működik; ezt azonban most figyelmen kívül hagyjuk, minthogy célunk éppen a csomaglétrehozás gyakorlása.) Adatmodellünk a következőképpen fog kinézni:

```
CREATE TABLE atf_szia_postings (
  posting_id   SERIAL    NOT NULL,
  user_id      INTEGER   NOT NULL  REFERENCES
               ➥users
               ON DELETE CASCADE
               ON UPDATE CASCADE,
  entry_date   TIMESTAMP  NOT NULL  DEFAULT
               ➥NOW(),
  posting      TEXT       NOT NULL
               CHECK (posting <> ''),
  PRIMARY KEY(posting_id)
);
```

Tisztán szakmai szempontból a fenti táblameghatározás valószínűleg bárki számára elfogadhatónak tűnik, aki már dolgozott korábban PostgreSQL alatt. A `posting_id`-t

használtuk egész elsődleges kulcsként, az `user_id` lesz a külső kulcsunk, van egy `timestamp` mezőnk (ez tartalmazza a dátum- és időadatokat), illetve egy szöveges mezőnk az üzenet tárolásához. Figyeljük meg, hogy a táblanév ***atf_szia*** előtaggal kezdődik, jelezve, hogy az ***atf-szia*** APM része.
A táblanevek és a csomagnevek egységes elnevezése a névtérrendszerezés egyik kezdetleges formája, de elég jól működik, amíg mindenki tartja magát hozzá.
A fentiek jól működnek PostgreSQL alatt, de korántsem biztosak Oracle-lel. Tudván, hogy az OpenACS-közösség mindig arra törekedett, hogy mindkét adatbázis alatt átlátszó módon dolgozhasson, mit tegyünk, hogy ne bosszantsuk fel nagykiszolgálós munkatársainkat?
A válasz a következő: amikor az ***atf-szia*** csomagot feltelepítjük, az APM egy ***sql/atf-szia-create.sql*** nevű fájlt fog keresni. Ha létezik ilyen nevű állomány, feltételezi, hogy minden támogatott adatbázis-kezelőn működik. Ha nem létezik, az APM megnézi, van-e ***postgresql*** és ***oracle*** nevű alkönyvtár, és a megfelelő könyvtárban végrehajtja az ***atf-szia-create.sql*** állományt. Ha tehát rendszerünk PostgreSQL-t használ, a fenti SQL-t a ***sql/postgresql/atf-szia-create.sql*** néven kellene mentenünk. A hivatalos OpenACS-csomagok elvileg Oracle és PostgreSQL alatt egyaránt működnek, tehát elég ritkán találni olyan csomagot, amelyik csak az egyik vagy a másik ág alatt futna (az e havi példák szavatoltan csak PostgreSQL alatt működnek, bár Oracle alá történő átvitelük sem lenne túl bonyolult).
Az OpenACS tisztogató (cleanup) parancsfájlok létrehozását is lehetővé teszi ***sql/CSOMAG-drop.sql*** néven – ez a létrehozó parancsfájl által készített valamennyi táblameghatározást és tárolt függvényt eltávolítja, e célból hozunk létre egy ***sql/postgresql/atf-szia-drop.sql*** nevű fájlt is.
Az APM a csomag első telepítésekor már tudni fogja, hogyan készítse az adatmodellt, de a fejlesztés időszakában erre még nem képes. Az ***atf-szia*** adatbázisba illesztését tehát nekünk kell kézzel elvégeznünk:

```
psql -f atf-szia-create.sql openacs4
```

Természetesen itt most feltételeztük, hogy az OpenACS adatbázis neve ***openacs4***, a PostgreSQL-kiszolgáló és a psql-ügyfél azonos gépen fut, illetve, hogy pillanatnyi felhasználónevünk a megfelelő adatbázis-hozzáférési jogosultságokkal bír.

OpenACS-sablonok

Most, hogy adatmodellünk elkészült, ideje elkészíteni azt az alkalmazást is, ami majd használni fogja. Az OpenACS 4-ben jelent meg az ADP (amely az AOLserver ASP/JSP megfelelője) alapú sablonozórendszer, amelyet az OpenACS egyik legjobb részének tartok.
Az ASP-szerű lapokat a nem programozók könnyebben megértik, mint a hagyományos kiszolgálóoldali programokat. Ugyanakkor a HTML-kód hibrid lapok szűk keresztmetszetté válhatnak, hiszen a tervezők és a fejlesztők nem tudnak egyazon fájlon egyidőben dolgozni.
E kihívásra az OpenACS-sablonok üdítő megoldást kínálnak. A lapot két részre osztjuk: az egyik (a ***.tcl***-lap) a programozóké, a másik (az ***.adp***-lap) a tervezőké. A ***.tcl***-lap rövid összefoglalóval kezdődik, ahonnan megtudhatjuk, milyen értékeket vár, illetve milyen értékeket fog átadni az ADP-lapnak. A Tcl-lap az `ad_return_template` hívással fejeződik be, ami kikeresi az azonos nevű ***.adp***-lapot, behelyettesíti a megfelelő változóértékeket, majd lefuttatja a lapon az ADP-értelmezőt.

1. lista posting.tcl

```
ad_page_contract {

    @author Reuven M. Lerner
(<reuven@lerner.co.il>)
    @creation-date 2002-September-18

} {

} -properties {
  postings:multirow
}

# Az eddigi  sszes  zenet megszerzØse

set sql "SELECT PE.first_names || ' ' ||
          PE.last_name
          as user_name,
             PA.email, PO.posting,
             PO.entry_date
          FROM Persons PE, Parties PA,
            atf_szia_postings PO
          WHERE PE.person_id = PA.party_id
          AND PE.person_id = PO.user_id
          ORDER BY PO.entry_date DESC"

db_multirow postings get_postings $sql

ad_return_template
```

A Tcl-lap adatokat adatforrások alakjában tud átadni az ADP-lapnak, ami valójában a változó csinosabb neve. Ha a Tcl-lapon ezt látjuk:

```
set five 5
```

az ADP-lapon bárhol elővehetjük ezt az értéket, ha a változó nevét @ jelek közé helyezzük, valahogy így:

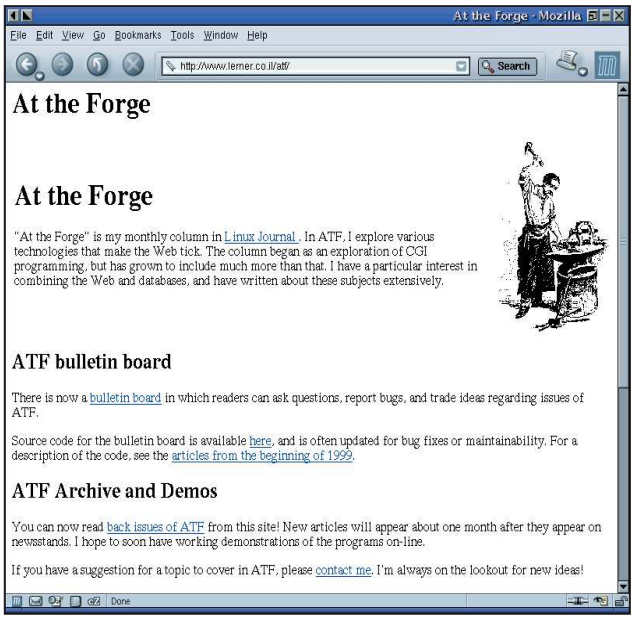
```
@five@
```

Ha a változónév a megfelelő lapon nem volt megadva vagy exportálva, az OpenACS futásidejű hibát jelez, és figyelmeztet, hogy ilyen változó nem létezik.
A feladatoknak ez a fajta megosztása azt jelenti, hogy a tervező és a programozó függetlenül tud dolgozni, mindaddig, amíg az előre megbeszélt, a lapmeghatározásban leírt csatolófelület (a Tcl-lap bemenete–kimenete) változatlan marad.

Tcl-lapok létrehozása

Az OpenACS számos olyan szolgáltatást tartalmaz, ami a programozókat segíti a HTML-űrlapok gyors és könnyű kialakításában. A jelenlegi bemutató céljaira ezeket a szolgáltatásokat most nem használjuk; ehelyett egyszerű nyers HTML-t fogunk alkalmazni. Alkalmazásunk két URL-t tartalmaz:

- Az egyik az ***atf_szia_postings*** tábla jelenlegi bejegyzéseit fogja időrendben megjeleníteni, illetve az új üzenetek űrlapja is itt jelenik meg. Az eredmények lekérdezéséhez és a `multirow` változóhoz történő csatolásához az OpenACS adatbázis-felületét fogjuk használni, ami könnyebben



használható, mint a beépített AOLserver adatbázis-felület (és nem kell aggódnunk a szálak és adatbázis-tárazás – pooling – miatt sem). Ezt a változót aztán az ADP-lap adatforrásaihoz adjuk, ahol az megjeleníthető. Az űrlap-művelet megadása (action) a másik URL-re fog mutatni.

- A második lap Tcl program lesz, amely fogadja a HTML-űrlapot, beviszi az új sort az adatbázisba, és visszavezeti a felhasználót az első lapra.

Más szavakkal alkalmazásunk létrehozásához két Tcl- és egy ADP-lapra lesz szükségünk. Valamennyit az atf-szia alatti www könyvtárba kell helyeznünk; amennyiben ez a könyvtár még nem létezik, hozzuk létre, majd kétszeresen bizonyosodjunk meg afelől, hogy a tulajdonosa azonos azzal a felhasználóval, amely alatt az AOLserver is fut.

Az *1. listában* található Tcl-lap (***posting.tcl***), nem vár értéket, és egyetlen ***postings*** nevű adatforrást ad tovább. A Tcl-lap az `ad_page_contract` függvényhívással kezdődik, amely lehetővé teszi, hogy lejegyezzük a fájl tulajdonosát és célját (az első érték), a megkapott bemenő értékeket (a második érték, ***posting.tcl***) és a továbbadott adatforrásokat (a harmadik érték, amelyet történeti okokból `properties`-nek neveztünk el). A Tcl-lap az `ad_return_template` hívással végződik, amely kikeresi a jelen ***.tcl***-fájllal azonos nevű ***.adp***-fájlt.

Bár minden adatforrás egyszerű Tcl-változó, az OpenACS sablonozórendszere néhány dologban leplezi a változók valódi természetét: minden adatforráshoz nevet és típust rendel (`multirow`, `list`, `onevalue` vagy `onerow`). Esetünkben a ***postings*** adatforrás `multirow` típusú, ami azt jelenti, hogy a `SELECT` lekérdezésünk eredményének több sorát is tartalmazhatja. A `db_multirow` függvény három értéket vár: a változó nevét, amelybe a sorokat be kell olvasni, a lekérdezés nevét, illetve magát az SQL-t.

A nevesített lekérdezés egyszerre áldás és átok az OpenACS-ben. Áldás, mert lehetővé teszi, hogy a lekérdezést egy külső ***.xql***-fájlba helyezve (ezek valamilyen adatbázis-kezelőhöz tartozó XML alakú fájlok) több adatbázissal dolgozzunk. A gond ezzel a megoldással az, hogy az OpenACS előbb a lekérdezés nevéhez rendelt ***.xql***-fájlt keresi ki, és csak akkor veszi figyelembe a ***.tcl***-lapunkban leírt SQL-t, ha az XML-t nem találja. Sok kezdő OpenACS-programozó tapasztalja meglepetten, hogy a rendszer figyelmen kívül hagyja a ***.tcl***-lapon végrehajtott SQL-módosításait, mivel valójában az ***.xql***-lapot nézi.

2. lista posting.adp

```
<master>

<!-- Az eddig kapott  zenetek megjelen tØse,
     ha van ilyen -->

<if @postings:rowcount@ ne 0>
<h1>Postings</h1>
  <multiple name="postings">
     <p><b>@postings.entry_date@,
        by @postings.user_name@
        ➥(@postings.email@)
     </b></p>

<blockquote>@postings.posting@</blockquote>
  </multiple>
  <hr>
</if>

<!-- œj  zenet hozzÆadÆs-->

<h1>Add a new posting:</h1>
  <form method="post" action="posting-add">
  <input type="text" name="posting_text">
  <input type="submit" value="Add some
              ➥text">
  </form>
</master>
```

Az ADP-lap létrehozása

Amint a ***posting.tcl*** kilép az `ad_return_template` meghívásával, az OpenACS sablonozórendszere kikeresi a ***posting.adp*** fájlt. Minthogy az összes Tcl-programkódot a Tcl-lapunk tartalmazta, ADP-lapunkban már nem lesz szükségünk a hagyományos `<% %>` tagokra. Ugyanakkor módot kellene találnunk arra, hogy adatforrásainkat HTML alatt jobban használhatóvá alakítsuk. Ahogy azt a *2. listában* (***posting.adp***) bemutatjuk, az OpenACS sablonozórendszere néhány új tagot vezet be, amelyek lehetővé teszik, hogy a ***.tcl***-lapon meghatározott értékeket egyszerűen elérjük:

- Feltételhez kötve az `<if>` cimke/tag segítségével a kimenő lapba beilleszthetünk valamilyen HTML-részletet, ami két értéket hasonlít össze (az `eq` és `ne`, azaz egyenlő és nem egyenlő relációk segítségével). A ***posting.adp*** példában a postings adatforrás sorainak számát (`@postings:rowcount@` lekérdezéssel) hasonlítjuk össze nullával. Ha nincs mit megjeleníteni, akkor egyáltalán semmit nem fogunk kirakni.
- Amennyiben vannak megjelenítendő lapok, végiglépkedünk rajtuk a `<multiple>` cimkével/taggal. A `<multiple>` teg blokkján belül az egyes adatbázismezőket a `@NAME.column@` szerkezetű kifejezéssel érhetjük el, amint azt a ***posting.adp*** forrásában megfigyelhetjük.

A megjelenített sorok számától függetlenül a ***posting.adp*** mindig beilleszt egy rövid HTML-űrlapot, ami a tartalmát a `posting-add` programnak adja át. Ez a ***posting-add.tcl*** nevű

3. lista posting-add.tcl

```
ad_page_contract {

  Add a new posting

  @author Reuven M. Lerner
  ➥(<reuven@lerner.co.il>)
  @date 2002-September-18
} {
  posting_text:trim
} -properties {
}

# pillanatnyi felhasznÆl  ID megszerzØse

set user_id [ad_get_user_id]

#  SQL-lekØrdezØs lØtrehozÆsa œj bejegyzØs
#  beszœrÆsÆhoz
set sql "INSERT INTO atf_szia_postings
            (user_id, posting)
          VALUES (:user_id, :posting_text)"

#  zenet beszœrÆsa

db_dml insert_posting $sql

ns_returnredirect posting
```

kód (amelyet a *3. listában* láthatunk) az `ad_page_contract` hívással kezdődik, amely egyetlen bemenő értéket sorol fel (`posting_text`). A bemenő értékek alapértelmezés szerint kötelezőek, de megjelölhetjük őket elhagyható értékként, illetve az `ad_page_contract` bejegyzésének módosításával alapértelmezett értéket rendelhetünk hozzájuk. Jelen esetben megkérjük az OpenACS-t, hogy a megkapott szövegből távolítsa el a kezdeti és a szövegvégi szóközöket.
Ezután lekérdezzük a pillanatnyi felhasználói azonosítót (ID) a beépített `ad_get_user_id` függvénnyel, és ezt az értéket rendeljük a `user_id` változóhoz. A `db_dml-t` használjuk fel az üzenet adatbázisba történő illesztéséhez. Figyeljük meg, hogy a `db_dml` változóneveihez kettőspontokat használtunk (dollárjelek helyett) – ez ugyanis az OpenACS adatbázis-felület szabványa, ami biztosítja, hogy nem lesznek idézőjelezési nehézségeink, ha az adatbázis-kiszolgálónak adatot adunk át.
Végül a ***posting-add.tcl*** a felhasználót átirányítja a `posting`-ra, ami a ***posting.tcl***-t futtatja le, megjelenítve a ***posting.adp***-t.

Az utolsó simítások

Most már visszatérhetünk az APM-be, és létrehozhatjuk a sablonjainkat, valamint az adatbázis-készítő parancsfájljainkat tartalmazó csomagot. Kattintsunk az atf-szia csomag nevére, majd a ***manage file information*** (fájladatok kezelése) hivatkozásra a lap aljának környékén. Ezután ebben a csomagban válasszuk az ***scan for additional files*** (további állományok keresése) lehetőséget – itt láthatjuk az általunk telepített ***.sql-***, ***.tcl-*** és ***.adp***-lapok listáját. Jelezzük, hogy az összes itt található fájl a csomag része lesz, majd térjünk vissza az ATF Szia APM fő kezelőképernyőjére, kattintsunk a ***generate a new atf-szia.info file (új atf-szia.info*** fájl készítése) hivatkozásra.
Készen állunk egy olyan APM létrehozására, amelyet azután már bármely OpenACS-felhasználó használni tud. Kattintsunk a ***generate a file*** gombra, és a terjesztés fájladatai alatt megtekinthetjük, hogy mekkora lett az új APM. Ha erre a hivatkozásra kattintunk, az APM letöltődik a rendszerünkre.
Hogyan telepítsünk fel egy új APM-et, amit valakitől kaptunk? A legegyszerűbb módszer, ha az APM-et a kiszolgáló fájlrendszerébe helyezzük. Ezt követően a böngészőnkkel térjünk vissza a fő APM-lapra (***/acs-admin/apm/***) és kattintsunk az ***install link***-re (a hivatkozás telepítése). Mondjuk meg a rendszernek, hol találja az APM-et, és máris bekerül a ***packages*** alkönyvtárba. Ettől kezdve a múlt hónapban megismert APM-telepítő segítségével folytathatjuk a munkát. Az adatmodell az adatbázisba kerül, és a weblapok elérhetővé válnak az érdekelt csoportok számára. Természetesen ha egy csomag felkerült a rendszerre, az ACS-honlaptérkép alkalmazásával tetszőleges URL-ek alatt új példányokat hozhatunk belőle létre.

Mit hagytunk ki?

Ez a kis példa az OpenACS alkalmazásfejlesztés jéghegyének csak a csúcsát tudta bemutatni. A lehetőségek sokrétűek:

- A sablonozórendszer önműködő űrlapkészítő rendszerrel is fel van szerelve, amellyel könnyedén készíthetünk érvényesítő visszakérdezésekkel és adathelyesség-ellenőrzéssel ellátott HTML-lapokat.
- A Tcl-függvényeket indításkor is betölthetjük az AOLserverbe, ha a csomag ***tcl*** könyvtárában határozzuk meg őket.
- A nevesített SQL-lekérdezések – mint korábban említettük – lehetővé teszik, hogy a Tcl program az Oracle és PostgreSQL rendszerek bármelyikét átlátszóan érje el .
- Minden egyes csomagpéldányt a társaitól teljesen függetlenül kezelhetünk – az OpenACS rendszer logikájának köszönhetően.
- Minden példány saját változókkal rendelkezhet, ami telepítésfüggő beállításokat tesz lehetővé.
- Minden csomag saját jogosultságkészletet határozhat meg (vagy használhat), így lehetővé válik, hogy rendszerünkön az egyes felhasználókhoz és csoportokhoz egyedi jogosultságokat, illetve elérési listákat alakítsunk ki.

Összegzés

Az OpenACS összetett rendszer, és az APM sem éppen a legkönnyebben megtanulható fejlesztőeszköz, hiszen olyan sok bonyolult esetet próbál megoldani, amelyekkel a web- és adatbázis-fejlesztők gyakorta találkoznak. Ugyanakkor még soha nem láttam egyszerűbb módszert a web- és adatbázis-alkalmazások terjesztésére ilyen fokú modularitás, adatbázisok közötti hordozhatóság és sablonok adta rugalmasság mellett. Az efféle alkalmazások létrehozásának egyszerűsége az adatmodellek gazdagságával és a hatalmas mennyiségű elérhető alkalmazással kiegészítve az OpenACS-t egyértelműen életképes és hasznos környezetté teszik a hálózati közösségek világában.

Linux Journal 2002. december, 104. szám

Reuven M. Lerner (➲ http:/www.lerner.co.il/atf) Nyílt forrású programokra, valamint web- és adatbázis-alkalmazásokra szakosodott tanácsadó. Könyve, a Core Perl, 2002 januárjában jelent meg a Prentice Hall gondozásában. Reuven feleségével és lányával Izraelben, Modi'in-ben él.