Bemutatkozik az AOLserver

Az AOLserver használata egyáltalán nem akkora kihívás, és nem is olyan bonyolult dolog, mint amilyennek először gondolnánk.

Az Apache, ez a jól ismert HTTP-kiszolgáló mindig is a nyílt forrású programok egyik reklámgyermeke volt: népszerű, megbízható, rugalmas, biztonságos, hordozható, bővíthető és megfelel az internetes szabványoknak. Az első megjelenése óta Apache-ot használok, és mindig élvezet volt dolgozni vele.

Nekünk, akik tudjuk, hogy számos nyílt forrású operációs rendszer, szövegszerkesztő, adatbázis és programozási nyelv létezik, talán az sem olyan nagy meglepetés, hogy nem az Apache az egyetlen nyílt forráskódú HTTP-kiszolgáló. Az viszont meglepő lehet, hogy az egyik ilyen választási lehetőség éppen az America Online-tól származik, attól a cégtől, amely a (szintén nyílt forrású) Mozilla böngészőt támogatja.

Az AOLserver sok tekintetben az Apache-hoz hasonló szolgáltatásokat nyújt: nyílt forrású felhasználási szerződés alatt fut, a beállítása könnyű és rugalmas, valamint a beágyazható (plugin) bővítmények írására API-felületet biztosít. Mivel azonban az AOLserver alapvetően más felépítésű, mint az Apache, bizonyos esetekben az előbbi hatékonyabb megoldás lehet. Továbbá az AOLserver beépített Tcl-értelmezővel, többszálú megoldással, adatbázis-API-felülettel és adatbáziskapcsolat-tárazással (database connection pooling) büszkélkedhet. Ha webhelyünk felhasználói nagyszámú adatbázis-kapcsolatot használnak, az Apache kiváltására esetleg az AOLservert is érdemes lehet szemügyre venni.

Ebben a hónapban az AOLservert fogjuk bemutatni – mintegy bevezetőként a nyílt forrású OpenACS (Open Architecture Community System) webalkalmazás-keretrendszerről szóló cikksorozatunkhoz. Bár az AOLserver nem feltétlen szükséges az OpenACS-hez, a rendszer beállításának és telepítésének ez az általánosan feltételezett módszere.

Keletkezéstörténete

Az AOLserver pályafutását NaviServer néven kezdte, amelyet a NaviSoft nevű, kiemelkedő minőségű, webterjesztéshez szánt ügyfél- és kiszolgálóoldali eszközeiről nevezetes, úttörő cég írt és árusított. Az AOL gyakran vásárolt meg olyan cégeket, amelyek érdekes módszereket fejlesztettek ki; a NaviSoftot az AOL inkább a kiszolgálóoldali fejlesztéseiért vette meg, semmint ügyféloldali eszközeiért.

Az AOLserver valószínűleg viszonylag ismeretlen maradt volna, ha nem következik be néhány fontos esemény: az AOL a program binárisait ellenszolgáltatás nélkül elérhetővé tette az Interneten, és ***Philip Greenspun*** dolgozni kezdett az ArsDigita Community Systemen. Az AOLserver segítségével könnyedén lehet nagy teljesítményű, adatbázisháttérrel támogatott honlapokat létrehozni; és az a tény, hogy az ACS igencsak keményen használ relációs adatbázisokat, azt sejttette, hogy az AOLserver tökéletes választás lesz.

Csakhogy míg maga az AOLserver ingyenes volt, a forráskódja továbbra sem volt elérhető a nagyközönség számára. 1999-ben Greenspun segített kidolgozni azt a megegyezést, amelynek értelmében az AOLserver 3.0 az AOLserver Public License alatt jelent meg, ami lényegében azonos a Mozilla Public License engedéllyel.

Ez év elején maga az ArsDigita lényegében kivonult az üzletből, legénységének maradványai pedig a Red Hat Software-hez igazoltak át. Az eredeti ACS program azonban továbbél az OpenACS projektben, amely az ACS eredeti Tcl-változata alapján készült – az AOLserver, illetve PostgreSQL vagy Oracle felhasználásával.

Az AOLserver nyílt forrású modellre való átvitele nem volt éppen zavartalan. Még a kezdet kezdetén néhány OpenACS-fejlesztő egy OpenNSD-nek nevezett saját ágat választott le (fork) az AOLserverből, arra hivatkozva, hogy az AOLserver-fejlesztőknek nyitottabbnak kell lenniük a közösség iránt. Az OpenNSD mostanra, úgy tűnik, elhalálozott, és az OpenACS-közösség ismét az AOLserver használatára buzdítja az embereket. Mindeközben az ArsDigita számos fejlesztést beépített az AOLserverbe, amelyek a kiszolgáló forráskódjában korábban nem szerepeltek, a legfrissebb üzembiztos vagy fejlesztői változatban sem. Minthogy számomra az AOLserver szinte mindig összekapcsolódott az OpenACS-sel, az AOLserver 3.3ad13 változatot fogom használni, amit a ➲ http://www.openacs.org/software.adp címen érhetünk el. A fejlettebb változatokat, ideértve a jövendőbeli 4.0-s fejlesztői változatát (snapshot) a ➲ http://www.aolserver.com címen érhetjük el. Jelenleg ezek a hivatalos változatok nem biztosítják az OpenACS támogatását.

Mitől is olyan nagyszerű?

Egészen ennek az évnek az elejéig, amikor a 2.0 végre megjelent a nagyközönség számára is, az Apache többfolyamatos kiszolgáló volt. Ez azt jelenti, hogy bármely időpillanatban számos Apache-folyamat futott, amelyek mindegyike az adott időben egyetlen HTTP-tranzakciót tudott csak lebonyolítani. Tíz egyidejű elérés kiszolgálása tehát tíz párhuzamos Apache-folyamat futását jelentette, 100 egyidejű kapcsolathoz természetesen 100 ilyen kapcsolatot kellett elérhetővé tenni.

Az Apache 2.0 ezen már némiképpen segít, mert lehetővé teszi, hogy egy folyamat több szálat futtass. Minden egyes szál egy-egy HTTP-kapcsolatot kezelhet, ami azt jelenti, hogy öt szálat használva az öt folyamatban összesen már legföljebb 25 egyidejű kapcsolatot tudunk fogadni. Minthogy a szálak általában kevesebb erőforrást fogyasztanak, mint a folyamatok, egy átlagos PC esetében ez tulajdonképpen teljesítménynövekedést jelent.

Az AOLserver ezzel szemben mindig is több szálon futott, és egyetlen folyamatot használt. Bármely időpillanatban mindig csak egyetlen `nsd`-példány fog futni a gépünkön, és a neve hűen tükrözi a tényt, miszerint valamikor NaviServer-démonnak hívták. Ez az egyetlen folyamat viszont számos egyidejű HTTP-kapcsolatot képes kezelni. Tulajdonképpen az AOL pontosan azért erőlteti tovább az AOLserver fejlesztését, mert az ennyire nagyszámú egyidejű kapcsolatot is képes kezelni.

Az AOLservert használják saját, nagy forgalmú webhelyeiken, ideértve a ➲ http://netscape.com-ot, az ➲ http://aol.com-ot és a ➲ http://digitalcity.com-ot.
A többszálú megoldás előnye – és lehetséges veszélye – az a tény, hogy a szálak könnyedén megoszthatják egymás közt az adatszerkezeteket. Az AOLserver ebből olyan módon kovácsol előnyt, hogy az adatbázis-kapcsolatokhoz létrehoz egy tárolót. Mivel az itt található kapcsolatok mindig nyitva állnak, webalkalmazásunknak nem kell időt fecsérelnie a megnyitásukra (vagy a lezárásukra). Továbbá minthogy csak ritkán fordul elő, hogy a kiszolgáló valamennyi HTTP-kapcsolata egy időben kérje az adatbázis-kapcsolatot, a tárolónak nem kell annyi kapcsolatot tartalmaznia, mint amennyi a szálak legnagyobb száma, ezáltal csökkenti a webkiszolgáló és az adatbázis-kiszolgáló által felhasznált memóriát. Elképzelhetjük úgy is, mint a csomagváltás (packet switching) adatbázisos megfelelőjét. A csomagváltás módszerének lényege, hogy a telefonvonalat számos fél között osztják meg egyszerre, kihasználva, hogy senkinek sincs szüksége a vonalra az idő száz százalékában.
Az AOLserver – akárcsak az Apache – támogatja a beágyazható modulokat (plugin modules). Számos modul létezik, kezdve az XML-értelmezőtől (`nsxml`) egészen a Python beágyazott változatáig (PyWx). Vannak modulok, amelyekkel CGI-programokat hajthatunk végre, vagy biztonságos kapcsolatot készíthetünk SSL-el, illetve az adott adatbázis-kezelőkkel, ilyen például a MySQL, PostgreSQL és az Oracle. Az OpenACS a PostgreSQL és az Oracle alatt is képes működni, és az ➲ http://openacs.org oldalról letölthető. Az AOLserver az `nsxml` modul mellett mindkét adatbázismodult tartalmazza.
Ahogyan a `mod_perl` lehetővé teszi, hogy a webfejlesztők az Apache beállításait és válaszait a C használata nélkül testreszabhassák, az AOLserver is nyújt egy beépített API-t, amelyen keresztül a testreszabható szolgáltatásokat Tcl nyelven állíthatjuk be. Az igazat megvallva én a fejlesztéshez szívesebben használnék Perlt vagy Pythont, de – mint azt számos AOLserver és OpenACS fejlesztő éveken keresztül bizonygatta nekem – a Tcl „nem olyan rossz”, így végül jó néhány érdekes, kezelhető alkalmazást készítettem a Tcl és az AOLserver segítségével, figyelemre méltóan rövid idő alatt (a beágyazott Python-modult még nem használtam, részben azért, mert az OpenACS megköveteli a Tcl használatát). Az AOLserver által nyújtott API meglehetősen megkönnyíti a munkát az olyasféle dolgokkal, mint a HTTP-fejlécek és a HTML-űrlapmezők értékei.

Fordítás és beállítás

Az AOLserver fordítása viszonylag egyszerű. Az Apache-csal ellentétben, amely a modulok fordítását a kiszolgáló telepítése után az `apxs` programon keresztül teszi lehetővé, az AOLserver esetében minden modult egyszerre kell lefordítani és telepíteni.
Bár külön felhasználó és csoport létrehozása nem kötelező, az AOLserver biztonsági okokból nem rendszergazdai jogosultsággal fog lefutni. Ezért az AOLserver fordítása és telepítése előtt nem árt, ha új felhasználót és csoportot készítünk a rendszeren – amennyiben követjük a hagyományt, akkor ***nsadmin*** néven. Az én Red Hat 7.2 rendszeremen egyszerűen ennyit kell tenni:

```
/usr/sbin/adduser nsadmin
```

Még mindig rendszergazdaként el kell készítenünk a ***/local/aolserver*** könyvtárat, ahová alapértelmezés szerint az AOLserver kerül. Ezt követően a könyvtár tulajdonjogát átadjuk az `nsadmin`-nak:

```
mkdir /usr/local/aolserver
chown nsadmin.nsadmin /usr/local/aolserver
```

Ha ez megvan, átváltunk `nsadmin` felhasználóra, megnyitjuk a ➲ http://openacs.org-ról letöltött forráskódot, és elkezdjük a fordítási folyamatot:

```
su - nsadmin
cd /tmp
tar -zxvf aolserver3.3ad13-oacs1-beta-
➥src.tar.gz
cd aolserver
./conf
```

A fenti sorok önműködően beállítják, lefordítják és telepítik az AOLservert a rendszerünk tulajdonságainak megfelelően, majd az elkészült fájlokat a ***/usr/local/aolserver*** könyvtár alá helyezik. Az AOLserver minden modult, amit csak tud, önműködően lefordít; figyelmen kívül hagyva (és kihagyva) az összes többit. Az én asztali gépemen, ahol a PostgreSQL fejlesztői könyvtárai megvannak, de az Oracle hiányzik, az AOLserver beállítása és telepítése után létrejön a PostgreSQL-meghajtó, de az Oracle-meghajtó egy az egyben hiányozni fog.
A fordítási folyamat némi időt vehet igénybe, és nem készít túl bőbeszédű kimenetet a képernyőn. Ha azt gyanítanánk, hogy a folyamat valahol lefagyott, belenézhetünk a ***log/aolserver.log*** fájlba; az összes fordítási kimenet itt található.
Amint befejeződött a létrehozási folyamat, a ***/usr/local/aolserver*** könyvtár alatt egy AOLserver másolatunk jön létre.
A ***/usr/local/aolserver*** alatti legfontosabb könyvtár a ***bin***, amelyben az AOLserver (`nsd`) program található, azokkal a megosztott könyvtárakkal (***.so***) egyetemben, amelyek a kiszolgáló egyes moduljaihoz fordultak le. A ***log*** könyvtár foglalja magában a kiszolgálóhoz tartozó elérési és hibanaplókat, végül a ***lib*** könyvtár tartalmazza a beépített Tcl-értelmezőt.
Az AOLserver beállításfájlja Tcl-ben íródott; alapértelmezés szerint egy egyszerű beállításfájl kerül a ***/usr/local/aolserver/sample-config.tcl*** könyvtár alá. Ezt a fájlt megvizsgálva látni fogjuk, hogy valamennyi beállítási direktíva tulajdonképpen Tcl változó-hozzárendelés. Ezek a változó-hozzárendelések szakaszokra tagolódnak, melyben minden szakasz általában egy-egy, a kiszolgálóhoz fordított modult jelent. Ahogy azt a példa-beállításfájlból is láthatjuk, a változókhoz közvetlen értékeket rendelhetünk. Például a következők szerint állíthatjuk át az AOLserver által figyelt HTTP-kaput 8000-re:

```
set httpport 8000
```

Mivel a beállításfájl Tcl-ben íródott, a `homedir` változót (amely alapesetben a ***/usr/local/aolserver***) az érték közvetlen kódolása helyett az AOLserverhez intézet kéréssel is beállíthatjuk:

```
set homedir        [file dirname [ns_info
config]]
```

Természetesen az egyszerű behelyettesítés segítségével változóbeállításainkat más változók értékére is alapozhatjuk:

```
set servername              "server1"
set pageroot
➥${homedir}/servers/${servername}/pages
```

Ez a két sor, amely az AOLserver példa-beállításfájljából

származik, a statikus URL-ek gyökerét a */usr/local/aolserver/servers/server1/pages* könyvtárra állítja. További beállításokat végezhetünk az `ns_param` paranccsal, amely általában két kapcsolót fogad: egy nevet és egy értéket. Minden kapcsolónak egy `ns_section` hívással kezdődő szakaszba kell tartoznia. Például a kiszolgáló hibakereső szolgáltatását úgy kapcsolhatjuk be, ha a teljes körű (global) a `ns/parameters` szakaszban bekapcsoljuk a `debug` kapcsolót:

```
ns_section "ns/parameters"
ns_param   debug            false
```

Sajnos az AOLserver kapcsolóinak leírása messze alulmúlja a hálózaton fellelhető Apache-dokumentáció szintjét. A kapcsolók csaknem teljes listáját az ➲ http://aolserver.com/docs/admin/config-reference.tcl.txt oldalon találhatjuk meg, ami egy olyan kiszolgálókiépítést mutat be, ami szinte mindent beállít. Ha befejeztük a rendszer beállítását, márpedig az alapértelmezett beállítások elég jók egy kisebb weblaphoz, az `nsd` program meghívásával és a használni kívánt beállításfájl megadásával elindíthatjuk az AOLservert:

```
cd /usr/local/aolserver
bin/nsd -f -t sample-config.tcl
```

A `-f` kapcsoló az előtérben futtatja az AOLservert, így a hibanapló a képernyőnkre kerül. Ha már elégedettek vagyunk azzal, ami történik, a `-f` kapcsolót eltávolíthatjuk – ettől kezdve kiszolgálónk hibanaplóját a *log* könyvtárban kereshetjük. Amennyiben azt szeretnénk, hogy az AOLserver a 80-as kapura hallgasson, mindenképpen rendszergazdaként kell indítanunk. Egyébként a Linux nem fogja elfogadni a kérelmet, azt üzeni nekünk, hogy kizárólag a szuperfelhasználó indíthat „kitüntetett” (azaz 1024-nél kisebb számú) kapukra hallgató kiszolgálókat. Igen ám, de ha csak a rendszergazda érheti el a 80-as kaput, az AOLserver viszont nem hajlandó rendszergazdaként futni, hogyan szolgáltathatunk mégis a 80-as kapun? Úgy, hogy az AOLservert rendszergazdaként indítjuk, és kapcsolóként megadjuk neki azt a felhasználót és csoportot, amelybe át kell váltania:

```
su root
cd /usr/local/aolserver
bin/nsd -f -u nsadmin -g nsadmin -t
➥sample-config.tcl
```

Ezek után böngészőnket átirányíthatjuk a *http://sajátgép.sajáttartomány.hu:8000/* címre, és már látjuk is a bemutatkozó AOLserver-dokumentumot, amely az új kiszolgálón üdvözöl bennünket. Figyeljük meg, hogy az AOLserver beállításfájljai egyaránt figyelembe veszik a számítógép nevét és az IP-számát, így ha csatlakozunk a hálózathoz, böngészőnket nem irányíthatjuk a *localhost* névre, hanem a teljes nevet kell használnunk.

Tcl-programok

Bár az AOLserver kétségtelenül kitűnő HTTP-kiszolgáló statikus dokumentumokhoz, igen valószínűtlen, hogy valóban csak erre szeretnénk használni. Sokkal gyakoribb eset, hogy a Tcl nyelv segítségével dinamikus lapokat készítenek. Ennek legkönnyebb és legegyszerűbb módszere egy HTML-kódot visszaadó Tcl-program készítése. A feladat végrehajtásához *.tcl* kiterjesztéssel hozzunk létre egy állományt a *pageroot* könyvtárban. Ide tetszés szerint bármilyen Tcl-t elhelyezhetünk; a legfontosabb dolog, hogy a végén adjuk ki az `ns_return` utasítást, azt az AOLserverben megadott Tcl-függvényt, amely három kapcsolót fogad:

1. a számszerű HTTP-válaszkódot (például 200 vagy 404), amely a program végrehajtásának sikerét vagy sikertelenségét hivatott jelezni;
2. a *Content-type* fejlécet, ami a visszaadandó tartalom típusát mutatja;
3. a felhasználóhoz visszatérítendő tulajdonképpeni tartalmat jelenti.

Például az egyszerű *Hello, világ* program így nézne ki:

```
ns_return 200 text/html "<html>
<head>
    <title>Testing</title>
</head>

<body>
     <p>Hello, world</p>
</body>
</html>"
```

Ha a fenti programot *hello.tcl* néven a *pageroot* könyvtárba helyezzük, majd betöltjük a böngészőbe, visszakapjuk az állomány betű szerinti tartalmát. Ennek az az oka, hogy nem állítottuk be újra az AOLservert, így a *.tcl* lapok használatát nem engedélyezi a *pageroot* könyvtárban. Az engedélyezéshez az `enabletclpages` kapcsolót a `ns/server/${servername}` szakaszban igazra (`true`) kell állítanunk:

```
ns_section "ns/server/${servername}"
ns_param   enabletclpages  true
```

Amint ezt a módosítást elvégeztük, az AOLservert újraindíthatjuk, és megpróbálhatjuk újra lekérni a *hello.tcl*-t. Ezúttal már a HTML-kimenetet kell látnunk a nyers text/plain kimenet helyett. A *.tcl* lap számos dolgot tartalmazhat: adatbázishoz kapcsolódhat, XML-állományokból szedhet ki adatot, hálózatokon keresztül szerezhet meg adatokat, illetve HTML-űrlapokból is kaphat adatot. Minthogy a Tcl számos karaktersorozat-kezelő parancsot ismer, igen sok lehetőségünk nyílik a bemenetet értelmezésére és a változók kimenetbe helyettesítésére.
Ne feledjük, hogy a Tcl az AOLserveren belül értelmeződik, nem egy külső folyamat hajtja végre. Ez egyben azt is jelenti, hogy az ilyen *.tcl*-fájlok sokkal gyorsabban hajtódnak végre, mintha az Apache-ban használnánk *.tcl* CGI-programokat; sok szempontból úgy viselkednek, mint a `mod_perl` Perl programjai.
Igaz ugyan, hogy a *.tcl*-fájlok nagyszerűek, ha a programozó HTML-kimenetet akar készíteni, csakhogy azt már keserű tapasztalataink alapján megtanulhattuk, hogy a grafikus tervezők általában nincsenek felkészülve arra, hogy forrásfájlokban megbúvó HTML-kódot javítgassanak. Ezért aztán sok webfejlesztő áttért a sablonok használatára – legyen az ASP, JSP, HTML::Mason, DTML vagy valamilyen hasonló módszer.
Az AOLserver saját beépített sablonrendszerrel rendelkezik, melynek neve ADP (AOLserver Dynamic Pages), és a felépítése gyanúsan hasonlít a Microsoft ASP nyelvére. A végrehajtandó kód <% és %> jelek közé kerül, a HTML közé ékelődött értékeket visszaadó kódot pedig <%= és %> jelek fogják közre. Például:

```
<% set foobar "abcdef" %>

<head>
     <title>Testing</title>
</head>
<body>
     <p>Hello, world</p>
     <p>Hello, <%= $foobar %></p>
</body>
</html>
```

A néhány lapnál nagyobb webhelyek a Tcl-kód egy részét valószínűleg megszeretnék osztani. Ennek a legkönnyebb módja, ha egy egy vagy több függvényt meghatározó ***.tcl***-fájlt készítünk, amely(ek)et az AOLserver induláskor betölthet. Ezek a függvények aztán valamennyi ***.tcl***- és ***.adp***-lapról elérhetők lesznek. A szolgáltatás engedélyezéséhez a következőket kell ***sample-config.tcl*** fájlunkhoz adni:

```
ns_section ns/server/${servername}/tcl
ns_param   library \
  ${homedir}/servers/${servername}/tcl
ns_param     autoclose    on
ns_param     debug  true
```

mivel a kiszolgálónk neve ***server1***, minden ***.tcl***-fájl, amit a ***/usr/local/aolserver/servers/server1/tcl*** alá helyezünk, be fog töltődni, amint az AOLserver elindul. Én például a következő tartalmú ***foo.tcl*** fájlt helyeztem ebbe a könyvtárba:

```
proc return_hello {} {
     return "hello"
}
```

Ezután újraindítottam az AOLservert (ami elengedhetetlenül szükséges ahhoz, hogy a Tcl-könyvtárfájlokat beolvashassa), majd a ***hello.adp***-t a következők szerint módosítottam:

```
<% set foobar "abcdef" %>
<% set hello [return_hello] %>

<head>
     <title>Testing</title>
</head>

<body>
     <p>Hello, world</p>
     <p>Hello, <%= $foobar %></p>
     <p>Hello, <%= $hello %></p>
</body>
</html>
```

Ettől kezdve a `hello` változó a `return_hello` függvényem eredményét tartalmazza, ami jelen esetben nem más, mint a hello szócska.
Mivel a könyvtár Tcl-függvényeinek betöltéséhez újra kell indítani az AOLservert, az új függvényt többnyire egyszerűbbnek találtam ADP-lapjaim `<% %>` szakaszában megadni. Csak miután már láttam, hogy függvény jól működik, tettem át a meghatározását a Tcl könyvtárba, és csak egyszer indítottam újra az AOLservert.
Az ADP és ***.tcl***-lapok minden dinamikus tartalmú oldalhoz igen jól használhatók. Néha azonban előfordul, hogy a lemezen dokumentum létrehozása nélkül szeretnénk valamilyen programot URL-hez rendelni. Könnyen megoldhatjuk a dolgot: egyszerűen csak egy Tcl-függvényt kell a ***tcl*** könyvtárfájlba helyezni, majd ezt a függvényt a kiválasztott URL-hez az AOLserver `ns_register_proc` parancsával bejegyezni:

```
proc http_hello {} {
  ns_return 200 text/html "<html>
                    <head><title>Hello!
                      </title></head>
                    <body><p>Hello!
                      </p></body>
                    </html>"
}

ns_register_proc GET /hello http_hello
```

Ha a fenti Tcl-kódot a korábban megadott programkönyvtár-könyvtár valamelyik fájljába helyezzük, majd újraindítjuk a kiszolgálót, és a böngészővel a ***/hello*** címre keresünk, a `http_hello` függvényünk kimenetét fogjuk látni.
Már jó néhány éve programozok `mod_perl`-ben, mégis le vagyok nyűgözve attól a könnyedségtől, ahogyan az AOLserver és az `ns_register_proc` függvény segítségével új lapokat illeszthetünk be. Ráadásul a `GET` és `POST` kérelmekhez különböző függvényeket rendelhetünk. Akár azt is megtehetjük, hogy olyan szűrőfüggvényeket jegyzünk be, amelyek egy másik lap által készített kimenetet figyelnek vagy változtatnak meg.

Összefoglalás

Ha egy hálózati feladatban OpenACS-t szeretnénk használni, szinte biztosan meg kell tanulnunk az AOLserverrel dolgozni. Még ha az OpenACS nem is érdekel valakit, az AOLserver rugalmassága, gyorsasága és többszálú képessége megéri, hogy egy kis figyelmet szenteljünk neki, hátha felhasználhatjuk dinamikus lapjainkhoz.

Linux Journal 2002. szeptember, 101. szám

Reuven M. Lerner (reuven@lerner.co.il) egy kisebb webes és internetes módszerekkel foglalkozó tanácsadó cég tulajdonosa és vezetője. Az ATF honlapon érhető el (➲ http://www.lerner.co.il/atf/).

Kapcsolódó címek

Az AOLserver-honlap, amelyet a SourceForge tart fenn a ➲ http://www.aolserver.com címen érhető el.

Az AOLserver API Python-változatát a ➲ http://pywx.idyll.org címen találjuk.

Az OpenACS-közösség fóruma ➲ http://openacs.org/bboard

Végül *Philip Greenspun* 1999-es AOLserver témájú cikkei kitűnően bemutatják a módszert. A négyrészes sorozat első részét a ➲ http://www.linuxworld.com/linuxworld/lw-1999-09/lw-09-aolserver_1.html címen olvashatjuk.