Szaktekintély, immár századszor

Eleinte minden aprósághoz CGI parancsfájlokat írtunk; mára viszont eljutottunk a webes eszközök és keretrendszerek pazar bőségéhez. Vajon milyen lesz a webes fejlesztések jövője?

Köszöntök mindenkit a Szaktekintély rovat századik írásának megjelenése alkalmából! Bizony, ez már a századik cikk, amit a ***Linux Journal*** számára, illetve korábban a SSC ***Websmith*** című kiadványa számára 1996 tavasza óta írtam. Az évek során rendkívül sok örömömet leltem abban, hogy hónapról hónapra a webes és kiszolgáló oldali megoldások egy-egy újabb elemét mutathattam be. Ebben a hónapban szeretnék kicsit visszatekinteni a kiszolgáló oldali és a web/adatbázis-programozás múltjára – így talán a jelenlegi helyzetet is jobban tudjuk majd értékelni. Ezután megvizsgáljuk a web mostani állapotát, és megpróbáljuk felmérni, vajon mire számíthatunk az eljövendő években.

Visszatekintés

Manapság a webet és az internetet mint természetesen meglévő dolgot fogadjuk el. A banki ügyeimet weben keresztül intézem; könyveket vásárlok online boltokból; webes ***RSS***-olvasóval webnaplókat olvasok; a webes újságokat gyakrabban látogatom, mint ahogy nyomtatott változatukat kézbe vettem; azonnali üzenetküldő programokon keresztül csevegek a barátaimmal és ismerőseimmel; sőt, még a fizetségeimet is a ***PayPalon*** keresztül kapom. Sokszor hallottam, hogy Manhattan lakóinak soha nem muszáj kimozdulniuk otthonukból, mert mindent házhoz lehet szállíttatni. Nem akarom megítélni, hogy ez jó vagy sem, de tény, hogy az internet világszerte egyre több ember számára teszi mindezt elérhetővé.

Az internet üzleti és szórakoztató jellege egy dinamikus folyamat eredményeként alakult ki. A webkiszolgálók eredetileg előre összeállított, nyers vagy ***HTML*** formázású, szöveges dokumentumok megosztására szolgáló megoldások voltak. Röviddel az után, hogy a weben közzétett, viszonylag korlátozott számú dokumentum felfedezése egyre népszerűbb tevékenység lett, valaki rájött, hogy a ***HTTP*** ügyfél-kiszolgáló jellegének köszönhetően a dokumentumokat dinamikusan, a beérkező kérésekre válaszul is elő lehetne állítani. Amikor egy ***HTTP***-ügyfél elküldi adott dokumentumra vonatkozó kérését a kiszolgálónak, akkor még nem tudja, hogy a dokumentum hónapok óta ott várakozik a kiszolgáló fájlrendszerében, vagy a kérésre válaszul állítják elő. Ez a szemlélet aztán örökre átalakította a webet, egyszerű, statikus anyagok megosztott tárháza helyett valós idejű dokumentumok és alkalmazások rendszerévé formálva.

A dinamikus forradalom kezdete meglehetősen egyszerű volt. Az első dinamikusan előállított tartalmak inkább burkolók voltak olyan unixos parancsokhoz, mint a ***mail*** és a ***finger***. Barátaimmal együtt készített első programjaim egyike például egyszerűen arra volt alkalmas, hogy kereséseket végezzünk újságunk online archívumában. Természetesen megtehettük volna, hogy különleges ***HTTP***-kiszolgálókat írunk a kívánt szolgáltatások biztosítására. Szerencsénkre azonban – és a többi webes fejlesztő szerencséjére – az ***NCSA httpd***, az ***Apache*** elődjének tervezői a ***közös átjárófelület (common gateway interface, CGI)*** révén az összes a kiszolgálón található program számára lehetővé tették a ***HTTP*** alapú kommunikációt. A ***CGI*** révén a kiszolgáló bármely programját elérhetővé tudtuk tenni a weben keresztül, egész egyszerűen egy ***CGI*** programba burkolva.

Az első években kemény idők járták. Mindenki feltételezte, hogy a web eleve állapot nélküli, és mindenki kitörő örömmel fogadta, amikor a Netscape bejelentette a sütiket (cookie), amelyek segítségével a kiszolgálók figyelemmel követhették a felhasználókhoz egyedileg rendelt adatokat. Nem voltak a webes forgalom mérésére alkalmas programok, nem is beszélve a webes programozás alacsonyabb szintjeit elfedő könyvtárakról. A hibakeresés nagyjából a webkiszolgáló hibanaplójának figyelésében merült ki. Minden olyan dolognak a használata, amely bonyolultabb volt egy egyszerű szöveges fájlnál, már fejlett, különleges adattárolási megoldásnak számított.

Itt és most

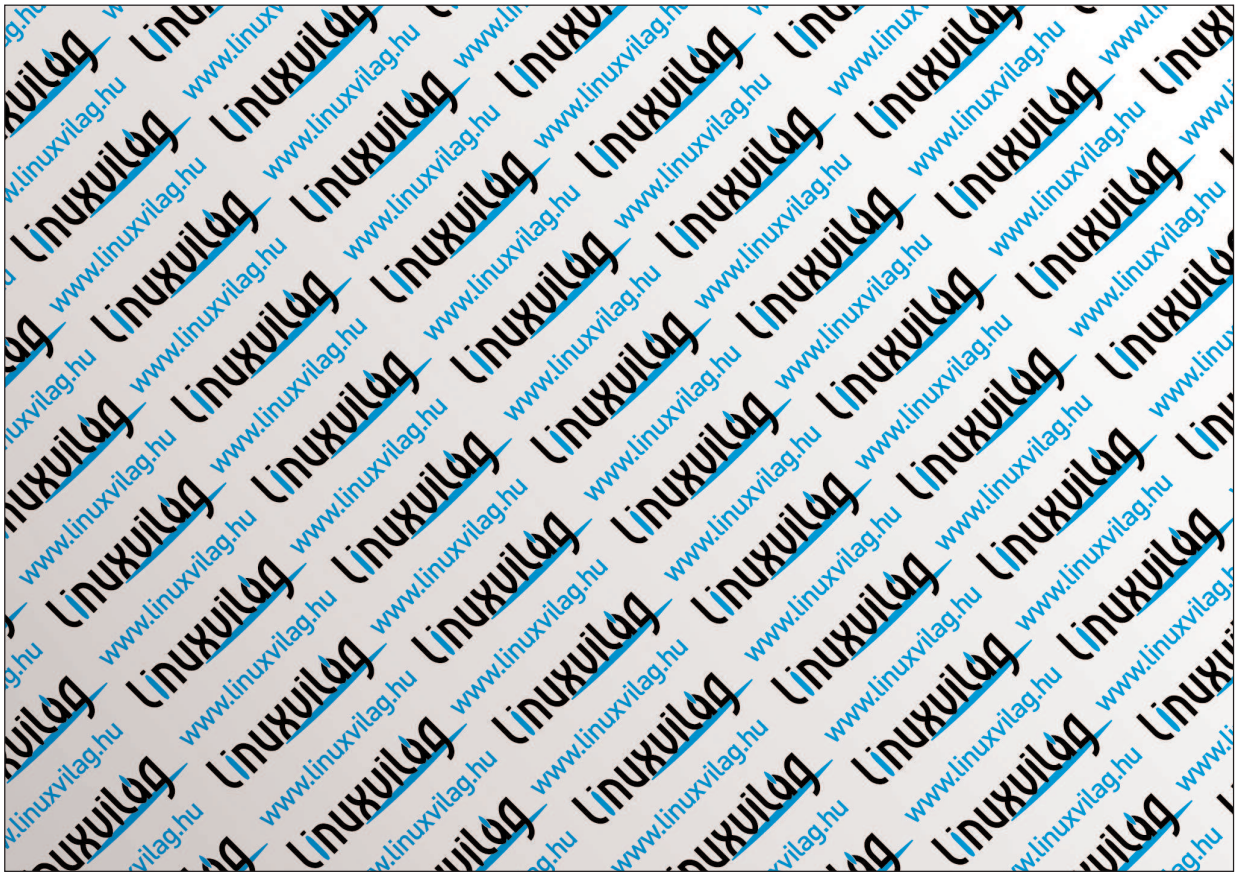
Napjainkra a webes fejlesztés teljesen megváltozott. Az Apache legújabb változatának letöltése és telepítése gyerekjáték, a ➲ ***www.apache.org*** oldal megnyitása után néhány perccel már profin beállított webkiszolgáló lehet a gépünkön. Relációs adatbázis nélkül, még ha esetleg nem is akarjuk bevallani, ma már nem létezik normálisabb webes alkalmazás. A legtöbb időt azzal takaríthatjuk meg, hogy többé nem is kell saját programokat írnunk – a rendelkezésünkre álló, a web és adatbázis alapú programok készítését segítő alkalmazások, könyvtárak és keretrendszerek száma egészen lenyűgöző. Korábban égen-földön kutatnunk kellett, ha találni akartunk egy az igényeinknek megfelelő, nyílt forrású alkalmazást. Kétségtelen, hogy a leginkább megfelelő alkalmazás megtalálása most sem

feltétlenül könnyű, ám ennek csak az az oka, hogy rengeteg gyenge vagy nem odaillő programot kell megismernünk, mire rálelünk az igazi megoldásra. Mindemellett a fejlesztői közösség is rengeteget fejlődött az évek során. A kiszolgáló oldali programozás világába kezdőként belépők jóindulatban és segítségben sosem szenvedtek hiányt, ám a kellő tapasztalat felhalmozódásához idő kellett. A webes programozás egykor kutatólaborok hálózatához hasonlított, amelyben minden résztvevő megosztotta tapasztalatait a közösség többi tagjával. Mára komoly tapasztalat gyűlt össze, a nyílt közösségben és a vállalatok falai mögött egyaránt. Ha egy ifjú programozó új alkalmazásokat szeretne írni, akkor szinte végtelen mennyiségű könyv, weboldal és forráskód áll rendelkezésére a tanuláshoz.

Arról sem szabad elfeledkezni, hogy az ezen a területen használt, népszerű programozási nyelvek, mint a *Perl*, a *Python*, a *PHP* és a *Java* az elmúlt évek során rengeteget fejlődtek. Engem ugyanakkor ezeknek a nyelveknek és könyvtáraiknak fejlődése kevésbé lepett meg, mint az, hogy az iparág egyre inkább a magas szintű nyelvek használata felé halad.

A webes világ hajnalán a legtöbben *C* és *C++* nyelven fejlesztettek. Akik magas szintű nyelveket, például *Perlt* vagy *Pythont* használtak, azokat szinte kontároknak tekintették, a „rendes” nyelveket használókhoz képest komolytalannak számítottak. A web mindezt megváltoztatta: ma már az is lehet komoly alkalmazásfejlesztő, aki csupán *PHP*-ben dolgozik. Természetesen a lefordított *C* programok ma is gyorsabban futnak, mint az azonos feladatokat ellátó, de magas szintű nyelveken készültek, ám utóbbiaknak a fejlesztési és hibakeresési időben észlelhető előnye viszont annyira nagy, hogy ma már szinte senki nem ír *C*-ben webes alkalmazást. Szintén jól látható folyamat, hogy a nagyvállalatok egyre inkább magas szintű nyelveket alkalmaznak, és a nyílt forrású programokat is elfogadják. Sok vállalat – például az Amazon vagy az *eBay* már rájött, hogy programozói jóval termelékenyebbek, ha magas szintű nyelvekkel dolgoznak. Az a tény, hogy ma már a *Java* és *C#* a két legalacsonyabb szintű webes fejlesztésre használt nyelv, elég sokat elmond arról, hogy hová tart az iparág. Olyan nyelvek vették át az uralmat, amelyek lehetővé teszik, hogy a programozó végre a valódi ötletekkel foglalkozzon, és ne biteket és bájtokat piszkáljon naphosszat. A *Java*, azt hiszem, kijelenthetjük, mint asztali programozási nyelv megbukott, ám a *C#* a jelek szerint, köszönhetően a *Microsoft .NET* kezdeményezésének, egyre népszerűbb; így nem kizárt, hogy néhány év múlva a legtöbb asztali alkalmazás olyan nyelveken készül, amelyek nem tartalmaznak mutatókat, az automatikus szemétgyűjtést ellenben elvégzik.

Természeten az ilyen nyelvek terjedésének okai szerteágazók, műszaki és pénzügyi jellegűeket egyaránt találunk közöttük. Biztos vagyok abban, hogy a web kiemelkedő szerepet játszott az irányváltásban. A magas szintű nyelvek, mint a *Perl*, kiválóan megfelelnek a webes környezet igényeinek, mint furcsa adattípusok kezelése, adatbázis-kapcsolat, könnyű használat, sokoldalú szövegkezelési lehetőségek és könyvtárak. A web nem más, mint hálózatra kihajított szövegek halmaza, márpedig ezeket a legmesszebbre magas szintű, nyílt forrású nyelvek segítségével hajíthatjuk.



Elképesztő növekedés állt be a kiszolgáló oldali alkalmazások készítésére szolgáló keretrendszerek számában is. Hiába rendelkezik valaki magas szintű programozási nyelvvel, ha saját rendszert kell írnia a felhasználók, a csoportok, az engedélyek, a tartalom és az üzenetek kezelésére. Valamelyik meglévő keretrendszert használva megúszhatjuk ezt a munkát, és felhasználhatjuk valaki másnak a tapasztalatait. A keretrendszerek két alapvető irányt követnek: egy részük tartalomkezelésre, híroldalak és magazinok oldalainak valós idejű összeállítására szolgál, mások viszont alkalmazáskiszolgálóként üzemelnek, vagyis alkalmazások készítésére használható eszközkészletekkel látják el a fejlesztőket.
Felületesen szemlélve azt hinnénk, hogy az olyan alkalmazási keretrendszerek, mint a ***HTML::Mason***, a ***Zope***, az ***OpenACS*** és a ***Java servletek/JSP***-k kevés közös tulajdonsággal rendelkeznek. Aki azonban egynél többet is megismer közülük, az hamar rájön, hogy bár mindegyik sajátos szemléletet követ, sokban azonosak. Igaz, hogy egyik keretrendszerről a másikra áttérni továbbra is fájdalmas, ám aki már többet is felfedezett közülük, annak egy-egy újabb megismerése rutinműveletté válik.
Igen, webes fejlesztőnek lenni 2005-ben messze kellemesebb ahhoz képest, amit tíz évvel ezelőtt ki kellett állnunk. A szoftverek egyre kiforrottabbak, a közösség kiterjedt és segítőkész, nem kell többé minden héten feltalálni a kereket, és a web felé nyitó szervezetek száma egyértelműen azt jelzi, hogy a munkákra van igény a piacon.

A jövő

Miután ennyit áradoztam a jelenlegi helyzetről, vajon mi vár ránk a jövőben? Milyen folyamatok fognak felgyorsulni a 2005-ös év során? Először is, szerintem egyértelmű, hogy a web, ami alatt itt a ***HTTP***, a ***HTML*** és az ***URL***-ek együttesét értem, felbomlik alkotó elemeire. Mindig is úgy gondoltam, hogy a web szokatlanul sokoldalú, hiszen három önmagában is nagy tudású megoldásból – ezek lennének a ***HTTP***, a ***HTML*** és az ***URL***-ek – áll össze. Tisztán látható, hogy mindhárom megoldás önmagában is megállja a helyét, és más területeken is meg fog jelenni.
Különösen érdekesek a webszolgáltatások, amelyek egy a böngészőktől különböző programok számára készült, új, sokoldalú és nyílt kommunikációs protokollt képviselnek. Amikor először megjelentek, azt hittem, hogy valami egyszerű dologról van szó, aminek kiötlői megpróbálják kihasználni a web sikerét és nevét. Az lehet, hogy a szegényes névválasztás tekintetében igazam volt, sőt, talán a hátterelméletek sem túl összetettek, ám mindez mit sem változtat azon, hogy a webszolgáltatások valóban komoly lehetőségeket kínálnak. Az az ötlet, hogy adott alkalmazás operációs rendszertől és programozási nyelvtől függetlenül kapcsolódhasson egy másikhoz, egyszerű, mégis zseniális. Bár a webszolgáltatások igazán jó használatára még ritkán látunk példát, az ***Amazon***, a ***Google*** és a ***Bloglines*** már igazolta, hogy belső ***API***-jaink ügyfelek és más külső személyek számára való elérhetővé tétele nem feltétlenül veszélyezteti üzletmenetünk biztonságát.
Hasonló irányzat a webböngészők asztali alkalmazások belső összetevőjeként való használata. A súgók már ***HTML*** alapúak, és mini webböngészőkkel működnek, de vannak teljes alkalmazások is, mint például az ***ActiveState Komodoja***, amelyek például a ***Mozilla*** motorra épülnek. Sokszor mondtam, hogy a Mozilla az új ***Emacs***. Noha a ***Mozilla*** alapú fejlesztés jóval nehezebb, mint az ***Emacs*** testreszabása valaha is volt, az a tény, hogy a ***Mozilla*** többféle operációs rendszer felett is futó, programozható környezetet biztosít sokoldalú asztali alkalmazások készítéséhez, önmagában is figyelemre érdemes. Ígéretes alkalmazás a ***Sunbird***, a ***Mozilla*** naptárprogramja is, amelyet a saját gépemen én is hónapokon keresztül használtam. A ***Sunbird*** ugyan számos hibától és problémától terhes, mégis roppant rokonszenvessé teszi, hogy az ***iCalendar*** szabványt használja különféle naptáraknak az internetről, ***HTTP***-n keresztül végzett letöltéséhez. Bizony! Pontosan erről volt szó: olyan asztali alkalmazás, amely ***Mozilla*** alapú, ***HTTP***-n keresztül ***URL***-eket kérdez le, mégsem webböngésző!
Kiszolgáló oldalon az együttműködés egyre hangsúlyosabb szerepet kap. Bár egy kereskedelmi enciklopédia színvonalát nem feltétlenül éri el, én mégis a ***Wikipediához*** fordulok először, ha valamilyen témáról bővebb ismereteket szeretnék gyűjteni. Köszönettel tartozunk a több ezer szerkesztőnek, munkájuk eredménye mindennapos használatra több mint kiváló. Egy ilyen jellegű együttműködést vezényelni nem kis feladat, és a ***WikiMedia Foundation PHP*** és ***MySQL*** alapú ***MediaWiki*** alkalmazása szép csendben élvonalbeli közös írást és szerkesztést segítő megoldássá vált.
Végül, hibakereső és tesztelő keretrendszerekre mindig is szükség lesz. A jövőben egyre kiterjedtebb tesztelésekre, sőt, tesztelés alapú programozásra kell felkészülnünk. A részegységek ellenőrzése alapján soha nem lehet egyértelműen megállapítani, hogy vajon a teljes alkalmazás megfelelően fog-e működni, ám ennek ellenére nyilvánvaló, hogy minden eljárást tüzetesen ellenőrizni kell, mielőtt összeépítenénk őket. A tesztelés alapú fejlesztés előtérbe kerülése az utóbbi néhány év egyik legfontosabb módszertani változása, és hiszem, hogy népszerűsége az alkalmazások bonyolultságával párhuzamosan fog növekedni.

Összefoglalás

Őszintén örülök, hogy alkalmat kaptam immár száz cikk megírására. Ám a fentiekből is kitűnik, a webes megoldásokkal, adatbázisokkal foglalkozó fejlesztőket számos újabb kihívás várja, vagyis bőségesen lesz témám további száz íráshoz. A következő hónapok során több itt említett ötletet is meg fogunk vizsgálni, ide értve az ***iCalendart***, a ***Wiki*** szoftvert, a webszolgáltatásokat és a tesztelés alapú fejlesztést. Lehet, hogy elmúlt tíz éves, a webbel dolgozni mégis szórakoztató, izgalmas és sok fejtörést okoz.
A ***reuven@lerner.co.il*** címen bárki levelét szívesen fogadom, ha meg kívánja velem osztani a web jövőjével kapcsolatos gondolatait, esetleg közölni szeretné, hogy mely tervezetekkel, megoldásokkal és irányzatokkal kapcsolatban szeretne bővebben is olvasni az eljövendő hónapok és évek során.

Linux Journal 2005. április, 132. szám

Reuven M. Lerner hosszú ideje web/adatbázis tanácsadóként és fejlesztőként dolgozik, jelenleg a Northwestern University oktatásmódszertan kurzusának hallgatója. Weblogja az altneuland.lerner.co.il, ő maga pedig a reuven@lerner.co.il címen érhető el.