Eszközválasztás

Érvek és ellenérvek négy webfejlesztő eszköz – a mod_perl/Mason, a J2EE, a Zope és az OpenACS – mellett és ellen.

Ha valaki arra kérne bennünket, hogy nevezzük meg a legjobb autót a piacon, valószínűleg azt válaszolnánk, hogy ez attól függ, ki fogja használni azt az autót. Végül is egy manhattani nyolcfős család valószínűleg más típusú járművet szeretne, mint egy agglegény kódlovag Észak-Dakota faluvidékén. Ugyanez igaz a programozási nyelvek és a fejlesztőeszközök estében is. Mindegyiknek megvan a maga helye és alkalmazási területe.

Bár ez nyilvánvalónak tűnhet, számos programozó úgy gondolja, hogy az általa használt eszközkészlet vagy nyelv bármikor bármely feladat megoldására alkalmas. Mint a régi mondás tartja: ha egyetlen munkaeszközöd egy kalapács, minden teendő olyan, mint egy szög. Egyetlen programozási nyelv sem tökéletes az összes feladathoz. Ezért ismernek a tapasztalt programozók több különféle nyelvet és tanulnak meg folyamatosan újakat.

Egészen az utóbbi pár évig a programozók programjaikat többnyire sebességre és alacsony memóriaigényre igazították. Ez érthető, hiszen a processzorok viszonylag lassúak, a memóriák pedig elég drágák voltak, így aztán amelyik program nem a legtöbbet próbálta meg kipréselni az alkatrészekből, az vacaknak tűnt. Manapság azonban már az olcsóbb, gyors számítógépek és a kedvező árú, méretes memóriák áldásait élvezhetjük. Ez azt jelenti, hogy a programtervezőknek lehetőségük nyílik olyan nyelveket használni, amelyek a gyors fejlesztést és a hosszú távú programkarbantartást részesítik előnyben. Ez nem azt jelenti, hogy ellenzem a memória vagy sebesség javítását, egyszerűen csak kevésbé tartom fontosnak ahhoz képest, hogy üzembiztos, karbantartható programokat fejlesszünk gyorsan és egyszerűen.

Közel két évvel ezelőtt elhatároztam, hogy egy hosszú cikksorozatot szentelek négy alapvető webfejlesztő módszernek: a mod_perl/Mason (Linuxvilág 1., 3–4. szám), a J2EE (Linuxvilág 13–14. szám), a Zope (Linuxvilág 15–19. szám) és az OpenACS rendszereknek (Linuxvilág 22–25. szá,m). Ezek a rendszerek nemcsak érdekesek és hasznosak, de egyben gondolatébresztők, továbbá új távlatokat nyitnak a webfejlesztők előtt. Bár az egyes közösségek között néha fellángolnak a viták arról, hogy melyik termék a legnagyszerűbb, a valóság az, hogy mindegyik egy kicsit más típusú feladatot próbál megoldani.

E hónapban arra szánunk egy kis időt, hogy megpróbáljuk összefoglalni az elmúlt pár évben megismert keretrendszereket és ötleteket. Természetesen nem várom azt, hogy aki a cikket elolvassa, azonnal az összes itt megnevezett alkalmazást használja is; mindössze abban reménykedem, hogy a legelfogultabb rajongónak is egy kis gondolkodnivalót tudok nyújtani.

mod_perl/Mason

Az Apache megérdemelten a nyílt forrású kezdeményezések egyik mintapéldánya. Megbízható, jól beállítható, kitűnően leírt, üzembiztos és bővíthető. Lenyűgöző dolgokat vihetünk végbe az Apache-csal, és sokféle módon igazíthatjuk hozzá saját igényeinkhez. Ha olyan webalkalmazást kell írnunk, aminek a lehető leggyorsabban le kell futnia, új modulokat írhatunk C-ben, amelyek aztán zökkenőmentesen illeszkednek az Apache-hoz.

Bár a C-programok gyorsan végrehajtódnak, és az Apache-könyvtárak (amelyeket manapság Apache Portable Runtime néven ismerhetünk) igencsak széles körű szolgáltatásokat és támogatást adnak a modulok alkotóinak, a fejlesztés C alatt lassabb és hibákra hajlamosabb, mintha valamilyen magasabb szintű nyelv, például Perl vagy Python alatt dolgoznánk. Így aztán talán nem is meglepő, hogy léteznek olyan Apache-modulok, amelyek ezeket a nyelveket ágyazzák be az Apache kiszolgálóba. A `mod_perl` lehetővé teszi, hogy C helyett Perl nyelven készítsünk Apache-modulokat – a Perl sebességét és rugalmasságát meglovagolva –, miközben közel korlátlan uralmat kapunk kiszolgálónk felett.

Tulajdonképpen ha valaki nagy teljesítményű webalkalmazást kér tőlem, különösen ha szövegfeldolgozás vagy relációsadatbázis-kezelés is a feladat részét képezi, a `mod_perl` mindig előtérbe kerül. Megírhatnám a modult C-ben is, de minek bajlódjak vele? Előfordul, hogy valóban érdemes C alatt dolgozni, de a legtöbb esetben a `mod_perl` sebességét még a nagy teljesítményű alkalmazások esetében is kielégítőnek találtam. Természetesen a `mod_perl` csodájának fénye némileg megkopik, amint a grafikus tervezők is színre lépnek. A tervezők nem igazán szeretnek kódot szerkeszteni, valahányszor megváltoztatják a stílust (vagy a tartalmat) az adott lapon, és ha rászabadítjuk őket a Perl-modulok forráskódjára, abból nem sok jó származik. Így aztán különféle sablonrendszerek tucatjai születtek, amelyek mindegyike valamilyen módon képes keverni a Perl és HTML nyelvet. Az egyik legnépszerűbb közülük a Mason, amely tudását az évek során már számos komoly terjesztő honlapon keresztül bizonyította.

A Mason valóban csodálatos eszköz, ami kitűnő egyensúlyt teremt a gyors fejlesztés (a Perlnek köszönhetően), a könnyű karbantarthatóság (a Masonnek hála) és (a `mod_perl` érdemeként) a gyors végrehajtás között. A Mason-levelezőlistától előnyös tájékoztatást és jó támogatást várhatunk, a csomag fejlesztőinek munkája pedig csodálatra méltó, mivel az idők során folyamatosan tökéletesítették programjukat. A Mason korszerű változatainak beállítását, használatát és hibakeresését látva azok az első változatok, amelyeket pár évvel ezelőtt először használtam, kezdetlegesnek tűnnek.

A Mason egyben háttér és keretrendszer, amelyben saját alkalmazásainkat készíthetjük el. Igaz ugyan, hogy az `Apache::Session` felhasználásával könnyedén készíthetünk sütiket és egyszerűen rendelhetünk a felhasználókhoz egyedi azonosítókat, de a kiforrott alkalmazásokban megtalálható felhasználói feliratkozás, a csoportok és a jogosultságok rendszerét mind magunknak kell megvalósítanunk. Néhány projektben ez a megfelelő megoldás, hiszen a szükséges rugalmasságot kínálja. Ha azonban már ötödjére kapjuk azon magunkat, hogy ismét egy felhasználókat és jogosultságokat kezelő rendszert készítünk, esetleg úgy dönthetünk, hogy egy kicsit több hátteret nyújtó lehetőséget választunk.

Java és J2EE

A Sun már jó néhány éve hirdeti kiszolgálóoldali megoldásként a Javát. A J2EE (Java 2, Enterprise Edition) elnevezés egy olyan gyűjtőfogalom, amelybe több, a fejlesztőket ilyen megoldások kialakításában segítő különféle módszer tartozik. A servletek olyan osztályok, amelyek a kiszolgálón valósítanak meg kódot; a JavaServer Pages (JSP-k) olyan Java/HTML-sablonok, amelyeket futásidőben fordítunk servletté. A JDBC-illesztő adatbázisok elérését teszi lehetővé, míg az Enterprise JavaBeans a tranzakciókat és az önműködő relációsobjektum-átalakítást kezeli. Ahhoz, hogy a Java világába beléphessünk, rengeteg rövidítést és módszert kell megismernünk, valamint több különféle szabvány számos változatát elsajátítanunk.

Amióta először megjelent, többször dolgoztam már Javával, s szinte minden esetben arra jöttem rá, hogy hiába szeretném, hogy érdekeljen, képtelen vagyok rávenni magamat a megkedvelésére. A Java önmagában nem rossz, és a különféle módszerek, amiket letesz az asztalra, mind igen lenyűgözők. A servleteket könnyű elkészíteni; a JSP-k (különösen a JSP-khez készíthető egyedi tagok) érett és figyelemre méltó sablonrendszert alkotnak, végül a JDBC mindent biztosít számunkra, amit egy adatbázis-kezelő felülettől valaha is elvártunk. Igaz ugyan, hogy az EJB használata a legtöbb projektben kétségtelenül túlzás, ugyanakkor hihetetlen hasznos lehet azokban a nagy fejlesztői csoportokban, amelyeknek a Sun eredetileg szánta. Továbbá számos jó megvalósítás, köztük kitűnő nyílt forrású alkalmazáskiszolgálók és eszközök születtek, ami mindenképpen lenyűgöző és ösztönző erőt képvisel.

Úgy tűnik, hogy a webfejlesztés világának „nagycége" a Java lett. A dolgokat megbízhatóan oldja meg, rengeteg képesség rejlik a tarsolyában, és számos szabványt támogat, fejlesztőeszközök seregei állnak a rendelkezésünkre – és meglehetősen sok ember használ Javát. Csak éppen a Java-projektekkel járó pluszmunka egy kicsit sok az én ízlésemnek. Már az is hosszú időt vesz igénybe, ha csak azt szeretnénk megállapítani, hogy melyik szabvány melyik változata melyik Jakarta alprojekt melyik változatához való. Ahogy sokkal érdekesebb kis cégnek dolgozni, mint nagynak, ugyanúgy sokkal érdekfeszítőbb szerintem Perl vagy Python alatt programozni, semmint Javában. Ráadásul a J2EE hasonló gondoktól szenved, mint amelyeket a `mod_perl` és a Mason esetében leírtam, nevezetesen, hogy tisztán csak hátteret ad, anélkül, hogy bármi figyelmet szentelne a beépített alkalmazásoknak. A fejlesztők csodálatos dolgokat hozhatnak létre, de minden projektben mindent újra ki kell fejleszteniük.

Talán a legjobban azt kedvelem a Java világában, hogy olyan nagy figyelmet fordít a karbantartható és megbízható programokra. Jó néhány próba- és fejlesztőeszköz, többek közt az Ant, a Cactus, a JUnit és a log4j lehetővé (sőt talán magától értetődővé) teszi, hogy a programozók teljes körű teszteket végezzenek a program kiadása előtt.

Tehát akkor a Java jó választás a webfejlesztéshez? Véleményem szerint minél nagyobb projekten dolgozunk, annál inkább érdemes elgondolkodnunk a Javán mint lehetőségen. De a klasszikus kis webalkalmazások esetében, amelyekkel a kis műhelyek dolgoznak, a fejlesztéshez kapcsolódó elkerülhetetlen többletmunka túlságosan komoly tényező, amit már nem hagyhatunk figyelmen kívül.

Zope

A Zope ékes bizonyítéka annak, hogy a nyílt forrású programok nem csak másolják kereskedelmi versenytársaikat. A Zope az objektum-adatbázist ötvözi a többprotokollú kiszolgálóval, továbbá gazdag objektumkészlettel és gördülékeny webalapú kezelőfelülettel ruházza fel őket. A Zope újító jellegű, okos, öröm vele dolgozni, és azon ritka nyílt forrású programok közé tartozik, amelyet a felhasználóknak terveztek, tehát nem csak a kódlovagoknak szól. A grafikusok örömmel hallják, hogy lehetőségük nyílik a dokumentum bármely korábbi állapotához visszatérni, a webalapú kezelőfelület „vissza" (undo) képességével.

A Zope több programozói felülettel is rendelkezik, amelyek a hatékonysággal szemben az egyszerűséget részesítik előnyben. Egyszerű DTML-sablonokat és Python-parancsfájlokat készíthetünk, használhatjuk az elbűvölő ZPT sablonokat, amelyek teljesen elválasztják egymástól a programot és a megjelenítési logikát, vagy végigjárhatjuk a teljes utat és egy új Zope-terméket készíthetünk. Az igazi erő a Zope-termékekben rejlik. Mivel minden termék egyben egy osztály is, különféle címeken egyetlen termékből több példányt is létrehozhatunk. Mivel az objektumok saját objektumrendszereiken kívül a címrendszer alapján is örökölnek (szerzeményezés), a példány jogosultságai, viselkedése vagy a kinézete címenként eltérő lehet.

No, ez eddig úgy hangzott, mintha a HTTP óta a Zope lenne a legjobb dolog, ami a Weben megjelent. Valóban, a Zope-kóderek növekvő száma egyben azt is jelenti, hogy egyre több ingyenes letöltés lesz elérhető és egyre több kereskedelmi termék használja alapjául a Zope-ot.

Sajnos a Zope-nak is megvannak a maga hibái, az első és legnagyobb az, hogy a tanulási görbéje igencsak meredek lehet. Mégha tapasztalt webfejlesztő is valaki, a Zope megköveteli, hogy majdnem az összes elgondolást az alapoktól újra megtanulja, megváltoztatva szinte minden szokást, amit az évek során összeszedett. Ez nem feltétlenül káros dolog, hiszen a Zope-ban elég jó megoldásokat találunk, de meglepetést okozhat és óvatosságra késztethet, egyszerűen azért, mert a Zope felhasználása a kezdeti, induló időszakban kétségtelenül lelassítja a dolgokat.

A másik gondom a Zope-pal az objektum-adatbázis alapja. Az objektum-adatbázisoknak hagyományosan több nehézségük is akad, és a ZODB szépen követi is ezt a hagyományt. Ugyanakkor a relációs adatbázisok még mindig elég általánosak, az emberek gyakran erre számítanak (általában meg is követelik). Elméletben ez sem gond. A Zope beépített ZSQL tagfüggvényei lehetővé teszik, hogy mindenféle különösebb erőlködés nélkül látványos dolgokat műveljünk a relációsadatbázis-lekérdezésekkel. A gond akkor válik nyilvánvalóvá, amikor az adat két különféle hely között oszlik meg: a ZODB és a relációs adatbázis között. Én például az összes adatomat egyetlen központi helyen szeretem tartani, így aztán ez a fajta megosztás számomra eléggé kiábrándító.

Azután ott van a sebesség kérdése: a Zope kifinomult jogosultság- és szerzeményezésrendszere valószínűleg gyorsabb, mint ahogy te vagy én magunktól meg tudtuk volna írni, de még így is elég lassúcska. A nehézség hivatalos Zope-megoldása a ZEO (Zope Enterprise Objects), amely lehetővé teszi, hogy egyetlen objektumadatbázist egyszerre több Zope-kiszolgáló is elérhessen. Jelenleg napi egymillió találatot bír el, ami a legtöbb lapon , amin dolgoztam, több mint elég. A különlegesen nagy webhelyek készítői azonban már aggódhatnak a Zope gyorsasága miatt, vagy másik megoldásként tervbe vehetik egy felsőkategóriás vas megvásárlását a ZODB kiszolgálóhoz.

Végül: a Zope-termékek többnyire függetlenek. A jó hír ebben az, hogy a fejlesztők párhuzamosan, a másik akadályozása nélkül tudnak dolgozni. A rossz hír, hogy a dolgok nincsenek annyira egységesítve, mint kellene: egységes irányítás nélkül a szolgáltatások ismétlődnek. Lehet, hogy egy ilyen méretű

nyílt forrású projekt esetében ez már elkerülhetetlen, mindenesetre kicsit kiábrándító.
Az utóbbi egy-két évben a Zope Corporation az alkalmazásfejlesztés helyett a tartalomkezelés területén kezdte el reklámozni a Zope-ot. Természetesen minden tartalomkezelő rendszert újra kell programozni és módosítani kell a vásárló igényeinek megfelelően, így a különbség nem annyira szembetűnő. Bár a Zope nem az egyetlen nyílt forrású tartalomkezelő rendszer a piacon, kétségtelenül az egyik legkifinomultabb és egyben az egyik legkiforrottabb is.
Saját munkáim esetében akkor választom a Zope-ot, ha az ügyfél projektje jelentős mértékű trükkös fejlesztést igényel, vagy ha viszonylag egyszerűen használható felhasználói felületet szeretne, illetve ha tartalomkezelésről van szó. Továbbra is nagyra tartom, és valószínűleg az elkövetkező években is elég sokat dolgozom majd a Zope-pal.

OpenACS

Az OpenACS a születésekor közösségi weblapokhoz tervezett kifinomult adatmodell volt, számos Tcl nyelven írt web-, illetve adatbázissablonnal kiegészítve. Idővel aztán egy sokkal nagyobb, sokoldalúbb projektté nőtte ki magát: független csomagokkal rendelkezik, amelyek a programot és az adatmodellt egyaránt képesek frissíteni, zökkenőmentesen képes együttműködni az Oracle vagy a PostgreSQL rendszerekkel, és kifinomult sablonozórendszert tartalmaz, amely a kódot elválasztja a HTML-kimenettől. Továbbá az OpenACS rengeteg előre elkészített alkalmazással érkezik, amelyek szinte bármit képesek megoldani, amire csak egy közösségi weboldalon szükségünk lehet – a webnaplóktól kezdve a gyakran ismételt kérdéseken át egészen az e-boltig. Igen rövid idő alatt elkészíthetünk egy weboldalt, s ehhez semmi mást nem kell használnunk, kizárólag a böngészőnket.
Mindig az OpenACS-t ajánlgatom azoknak a nonprofit szervezeteknek, akik hálózati közösségeket akarnak létrehozni, el akarják érni a tagjaikat, le akarják bonyolítani a tagok közti vitákat, vagy egyszerűen adatokat szeretnének közölni a módszer mélyebb ismerete nélkül.
Azt mondják, az OpenACS több gonddal is küszködik. Az első és legfontosabb a tanulás. A Zope megismerése elég nehéz, hiszen oly sok módszert szükséges megérteni hozzá. Az OpenACS sokkal egyszerűbb modellel bír, de az égvilágon mindent relációs adatbázisban tárol. Noha ez azt jelenti, hogy minden adat egy helyen van, a relációs adatbázisok hírhedten rosszak a rendszerkezelésben és a világ összes okos OpenACS-fejlesztője sem képes ezt kiküszöbölni.
Így aztán, az OpenACS-csel való munkához meg kell tanulnunk, hogyan készíthetünk egyszerű objektumöröklési modellt és a hozzátartozó teljes körű felületet, ami elérhetővé teszi.
Ha még soha nem készítettünk tárolt eljárásokat, vagy nem dolgoztunk olyan adatbázissal, ami táblák tucatjait vagy százait tartalmazza, lehet, hogy az OpenACS-hez szükséges háttértudás végül letaglóz minket.
Az OpenACS másik nehézsége, hogy csak kevés leírás áll a fejlesztők rendelkezésére és jóformán semmi a végfelhasználók számára. Az OpenACS kétségkívül összetett rendszer, amelyet egy szakmán kívüli ember számára elég nehéz leírni, de meglehetősen bosszantó, ha semmilyen segítséget sem találunk a témában. Becsületükre legyen mondva, a fő openacs.org honlapot mostanában modellezték és írták újra, röviddel azelőtt, hogy ezt a cikket megírtam volna, és úgy tűnik, e tekintetben is akad néhány kedvező változás.
Végezetül kicsit ironikusnak találtam, hogy az OpenACS egyre lomhábbá vált az idők folyamán. Talán elfogadható érv, hogy ez amiatt történt, mert a legfrissebb változat (e sorok írásakor a 4.x) sokkal többet tud a felhasználók, csoportok és engedélyek témakörében, mint az elődei, és ezek ellenőrzése minden HTTP-kérelemnél időt vesz igénybe. Ezen felül a fejlesztők is tudják, hogy sok javítást még végre lehet hajtani ahhoz, hogy ne igényeljen minden kérelem olyan sok adatbázis-lekérdezést.

Összegzés

Néhány nappal a cikk megírása előtt megjelent a Weben egy új hír arról, hogyan ültették át a Yahoót PHP webprogramozási környezet alá. Én személy szerint más nyelvekkel szeretek dolgozni, és nem éreztem volna nagy kedvet, hogy a teljes Yahoót új nyelven írjam újra. De a Yahoo egyedi igényeinek megfelelően úgy tűnik, hogy a PHP tényleg jó választás.
Kapnak egy jó pontot tőlem, amiért végiggondolták az összes lehetőséget, mérlegelve az előnyöket és hátrányokat, míg végül megkapták azt, ami leginkább illik az igényeikhez.
Ahogy már a cikk elején is írtam, mélyen hiszek abban, hogy mindig meg kell keresni azt a módszert, ami az adott feladat szerint igényekhez a legjobban igazodik. Nekem, a fejlesztőnek ez azt jelenti, hogy folyamatosan új nyelveket, módszereket kell megismernem és elsajátítanom; ugyanakkor azt is magában foglalja, hogy az ügyfeleim olyan megoldásokat kapnak, amelyek illeszkednek a feladataikhoz, és én mint programtervező széles körű és mélyebb tudást gyűjtök.
Az a tény, hogy ezek a rendszerek az Interneten ingyenesen hozzáférhetők, azt mutatja, hogy az egyetlen akadályt csak a ráfordított idő és erőfeszítés képezheti. Mindenkit arra buzdítok hát, hogy szorítson egy kis időt a kipróbálásra – te is és a veled dolgozó emberek is kétségtelenül értékelni fogják az eredményt.

Linux Journal 2003. február, 103. 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

A `mod_perl`-ről a ➲ http://perl.apache.org lapon, a Masonről pedig a ➲ http://www.masonhq.com lapon olvashatunk.

A J2EE témával mélyebben a ➲ http://java.sun.com foglalkozik. Az Apache Software Foundation Jakarta Projektje a nyílt forrású Java programhoz a ➲ http://jakarta.apache.org címen lelhető fel.

A Zope főlapja a ➲ http://www.zope.org. Akik újak a Zope-témában, nézzenek el a ➲ http://www.zopenewbies.net lapra; a nagyobb tudású felhasználók a ➲ http://www.zopezen.org-ot keressék. Végül az OpenACS közösséget az ➲ http://www.openacs.org címen érhetjük el.
A PostgreSQL adatbázist, amelyet az OpenACS-hez használhatunk, a ➲ http://www.postgresql.org címen találjuk.

Az I. és II. évfolyam egységára lapszámonként 500 Ft

#1. 1. szám 2000. november, 2 CD _____ pld.
#2. 2. szám 2000. december, 2 CD _____ pld.
#3. 1. szám 2001. január, 2 CD _____ pld.
#4. 2–3. szám 2001. február–március 3 CD _____ pld.
#5. 4. szám 2001. április, 2 CD _____ pld.
#6. 5. szám 2001. május, 2 CD _____ pld.
#7. 6-7. szám 2001. június–július 3 CD _____ pld.
#8. 8. szám 2001. augusztus, 2 CD _____ pld.
#9. 9. szám 2001. szeptember, 2 CD _____ pld.
#10. 10. szám 2001. október, 2 CD _____ pld.
#11. 11. szám 2001. november, 2 CD _____ pld.
#12. 12. szám 2001. december, 2 CD _____ pld.

A III. évfolyam 7. számig a magazin egységára lapszámonként 1000 Ft

#13. 1–2. szám, január–február, 2 CD _____ pld.
#14. 3. szám, március, 2 CD _____ pld.
#15. 4. szám, április, 2 CD _____ pld.
#16. 5. szám, május, 2 CD _____ pld.
#17. 6. szám, június, 2 CD _____ pld.
#18. 7. szám, július, 2 CD _____ pld.
#19. 8. szám, augusztus, 2 CD _____ pld.
#20. 9. szám, szeptember, 2 CD _____ pld.

Az akció keretében a megrendeléseket csak postai utánvéttel tudjuk teljesíteni. Az utánvét költsége egységesen 300 forint minden csomagra.

A kitöltött megrendelőlapot küldje be szerkesztőségünkbe faxon vagy postán.

Cím: 1081 Budapest, Népszínház u. 29.
Fax: 303-16-19

Megrendelő neve: _____

Cég neve: _____

Cím: _____

Telefonszám: _____

Fax: _____

E-mail: _____

_____

megrendelő aláírása