DEO

I

Temelji Jave

U OVOM DELU:

1. Predstavljanje Jave
2. Kako početi sa Java 2 SDK
3. Rad sa objektima
4. Tipovi podataka, modifikatori i izrazi
5. Pakovanje klasa i interfejsa za pristup
6. Nizovi i iskazi kontrole toka
7. Obrada izuzetaka i tvrdnje
8. Niti i višenitnost

POGLAVLJE 1

Predstavljanje Jave

Nekada davno je postojalo ostrvo u Indoneziji poznato po imenu Java. Ljudi sa ostrva su miroljubivo proživljavali svoj život, idući na svoj posao i svoje dnevne obaveze hiljadama godina. Jednog dana ranih 1990-tih, kompanija sa imenom Sun Microsystems je kreirala novi programski jezik nazvan Oak. Kako je Oak bio interni jezik za Sun, ime nije bilo previše važno i ljudi sa Jave su nastavili da idu na svoj veseo način, ignorišući ono što je u radnji bilo za njih.

Nekoliko godina kasnije, Sun je bio spreman da Oak učini novim proizvodom. Na nesreću, ime Oak nije prošlo test zaštićenih znakova. Neki drugi izbori, kao što su Silk, Ruby, i WRL (od WebRunner Language), takođe nisu zadovoljili očekivanja, već je na kraju pobedilo ime Java i prošlo svoj put kroz reviziju legalnosti. (Očigledno, postojanje konflikta imena za programski jezik i ostrva nije bilo važno. Šta možete očekivati od kompanije koja je dobila ime po zvezdi (Sun Đ Sunce)?) Potom, 23. maja 1995., na Sunovoj godišnjoj Sun World konferenciji je lansirana Java, kao dvanaesta od oko 20 različitih objava. Malo Indonežana je znalo šta će se desiti na njihovom malom ostrvu.

Java platforma je počela da se razvija i Microsoft je smatrao dovoljno važnom da bi je inkorporirao u svoj pretraživač Weba Internet Explorer. Nazad na ostrvu Java vulkani su postali nemirni. Onda, 1997, Sun je tužio Microsoft zbog nekih inkompatibilnosti Jave i Krakatau je upravo eksplodirao. (U redu, Java doživi oko 10 glavnih vulkanskih erupcija svake godine; to može biti samo koincidencija.) Kako je parnica išla sve dalje i dalje, život na ostrvu je postajao gori. Ekonomski nemiri u Džakarti su doveli do zahteva da predsednik Suharto siđe sa vlasti, a Istočni Timor je zahtevao političku autonomiju. U blizini krvavih bitaka do kojih je došlo na Sun Java frontu, sa konačnim poravnanjem od 20 miliona dolara sa Microsoftom 2001. Slučajnost? Pitajte samo predsednika Wahida šta on ima da kaže.

Istorija Jave 101

U redu, dosta sa uvijenom istorijom ostrva Jave i platforme Jave. I da, to je *Java*, *platforma*. Ono što je 1995 počelo kao samo još jedan programski jezik, sada je formalno poznato kao platforma. Počinjući kao platformski neutralan, Internetu naklonjen razvojni jezik, Java je evoluirala u nešto što znači kreiranje programa baš za sve. Vratimo se korak u nazad i pogledajmo kako se to desilo.

Ranije, Januara 1991, počeo je projekat sa imenom Green. Svrha projekta Green je da poboljša način kontrole set-top boxova, koji su pametni boxovi sa pristupom preko kablovske TV. Kako se hardver u uređajima korisnika uvek menjao, Jamesu Goslingu, "ocu" Jave, je postalo jasno da C++ nije odgovarajuć za taj posao. Kao rezultat, kreiran je jezik nazvan Oak; bio je manje osetljiv na bagove i nije obarao ceo sistem. U početku je radio na uređaju Hammer tehnologije nazvanom *7 (Star7). Preko ekrana za dodirivanje je skakutao mali digitalni znak nazvan Duke, pomoćni agent, koji bi odlazio i izvršavao različite zadatke. (On je kasnije prerastao u Java maskotu.)

NAPOMENA

Ukoliko nikada ranije niste videli Dukea, on izgleda kao mešanac komunikatora iz Zvezdanih Staza i naopako okrenutog zuba, sa veštičjom kapom, sa velikim crvenim nosem i sa rukama u stilu Screen Bean. (Screen Beans su oni crtani likovi koje često viđate da trče posvuda u PowerPoint prezentacijama.) ■

Oak je bio korišćen kao pokušaj u projektu za TV set-top operativni sistem. Pokušaj je propao, i Sun je odbio da kupi ponudu od Trip Hawkins (tada CEO za Gamemaker/3DO), tako da je Sun ostavio sve ove stvari ne znajući šta sa njima da radi. Na sreću, počela je Internet revolucija sa Mosaicom, prethodnikom Netscape Navigator Web pretraživača. Oaku je 1994. promenjena namena za Web, i napravljen je Web pretraživač nazvan WebRunner (koji je kasnije promenio ime u HotJava) da bi pokazao novu tehnologiju od koje je na kraju nastala Java.

Do sada je Java postala javna. Na sada zloglasnoj SunWorld konferenciji, John Gage iz Suna, i Marc Anderson iz Netscapea, su svetu predstavili Javu. Netscape je izvršio inkorporisanje Jave u sledeće izdanje svog pretraživača Weba, i do kraja godine, kompanije poput IBM, Oracle, Borland, Adobe, Macromedia, Lotus, Spyglass, i Intuit su inkorporisale Javu u svoje proizvode. Čak je i Microsoft dozvolio Javu, angažujući se na inkorporaciji Jave u svoje proizvode, operativne sisteme, i razvojne alate.

Ova početna verzija Jave je bila mala i stajala je na floppy. Srž interpretera je bio oko 100KB. Matematička biblioteka je dodavala 20KB. Kod za podršku integracije sa C bibliotekama je bio samo 50KB. Većina biblioteka klasa je stajala u 375KB, izostavljajući jedino grafičke biblioteke koje su karakteristične za platformu i koje variraju u veličini. Ukupno, Java je bila samo oko megabajta i mogla se izvršavati na računaru sa 1MB ROM i 1MB RAM. (*7 je imao 4MB.)

Skočimo šest godina unapred, i videćemo da je Java značajno evoluirala. Više nije ograničena da se izvršava u pretraživaču Weba, već se Java programi mogu pronaći svuda — od Web servera do nakita sa trepćućim svetlima, do pametnih tostera, i čak i u originalno planiranim set-top boxovima. Suštinske biblioteke su značajno narasle da, u poslednjoj realizaciji (Java 2 Standard Edition, verzija 1.4) zauzimaju 30 do 40MB samo za svoje radno okruženje (runtime environment). Naravno, sada može više da uradi, u suštini zahvaljujući tome što imate Java platformu, koja je kombinacija programskog jezika, standardnih biblioteka, i radnog okruženja. I to je ta platforma o kojoj ćete učiti u ovoj knjizi.

Ispitivanje arhitekture Jave

Sudeći prema originalnom izveštaju (dostupnom na `http://java.sun.com/people/jag/OriginalJavaWhitepaper.pdf`), ciljevi pri dizajniranju Jave su bili "jednostavan, objektno-orjentisan, distributivni, interpretatorski, robustni, bezbedni, nezavisan od arhitekture, prenosiv, sa visokim performansama, višenitni, dinamičan jezik." Ako pogledamo šta ove žargonske reči znače, videćemo šta je Java pokušavala da pokaže tada i šta i dalje primenjuje.

Jednostavnost

Na prvoj C++ konferenciji, Bill Joy, kopronalazač i sada šef naučnik i CEO u Sun, je rekao : "C++ je previše komplikovan, ono što ja želim je C++++—." To je ono što su on i svi drugi dobili u Javi. Java inkorporiše nove zadatke kao što je automatsko sakupljanje otpada (++) i uklanja zabunu i retko korišćene aspekte iz C++ kao što je preklapanje operatora (—)..

Još jedan aspekt jednostavnosti Jave je to da ništa u Javi zapravo nije novo. Ukoliko pogledate kroz set mogućnosti Jave u istoriju računarstva, otkrićete da je sve odnekud došlo.

- Klase su došle iz C++ i Smalltalk, mada su u Javi ograničene na samo jednu primenu nasleđivanja.
- nterfejsi su došli iz Objective-C, obezbeđujući Javi višestruko nasleđivanje interfejsa.
- Paketi su nastali u Modula-u, dodajući Javi hijerarhijski prostor imena i logičke razvojne jedinice.
- Istovremenost je postojala u Mesa-i, i donela je Javi ugrađenu podršku višenitnosti.

- Obrada izuzetaka je postojala u Modula-3, dodajući Javi metode koje su deklarativne u tome šta izbacuju.
- Dinamičko povezivanje i automatsko sakupljanje otpada su došli iz Lisp, nudeći Javi mogućnost učitavanja dodatnih klasa prema potrebi i oslobađanje memorije kada nisu potrebne.

I kako su nove mogućnosti dodavane Javi, one su takođe bile izvođene iz oprobanih i dokazanih primena. Na primer, ono što i dalje nedostaje u realizaciji 1.4 je *podrška parametarskih tipova*, takođe poznata kao šabloni i generički tipovi. Jednom kada bude dodata, ova mogućnost će dodati podršku za kolekcije bezbedne u odnosu na tip, mogućnost koja je zahtevana neko vreme. Međutim, moćnici ne žele da dodaju baš sve; oni pokušavaju da se uvere da sve što bude dodato Javi radi ispravno. Očekujemo da ćemo otkriti podršku generičnosti u Javi 1.5, uz pretpostavku da će do tada sve da bude odlučeno.

Objektna-orjentisanost

Ljudi Javu nazivaju *objektno-orjentisanom*, ali šta zapravo to znači? U suštini, objektno-orjentisano programiranje (OOP) je način razvijanja sofrvera opisivanjem problema kroz upotrebu elemenata ili objekata iz prostora problema, umesto sekvencijalnim setom koraka koje treba izvršiti na računaru. Dobro prijektovanje onda vodi do ponovo upotrebivih, proširivih i lako održavanih komponenti. Te komponente su dovoljno fleksibilne da bi podnele promene okruženja do kojih se vremenom dolazi, jer je primarni zadatak tih objekata samo da šalju poruke jedni drugima.

U suštini, Core Java API je kolekcija ovih ugrađenih komponenti, nazvanih *biblioteke klasa*. Umesto da u Javi ispočetka izmišljate točak, počinjete sa standardnim bibliotekama. Standardne biblioteke su vremenom evoluirale, dajući više ugrađenih komponenti sa svakom novom Java realizacijom.

Za detaljniji pogled u objektno-orjentisano programiranje, videti poglavlje 3.

Distributivnost

Od svog početka 1982, Sunova vizija je bila "mreža je računar." Sa Javom, oni su želeli programski jezik koji je razumljiv na mreži i koji podržava pristup distribuiranim objektima lako kao lokalnim. Iako je predstavljen od početka, ovaj cilj se menjao kada su u razmatranje uzete mogućnosti Jave.

Na početku, sve što je Java mogla bio je pristup distribuiranim objektima preko standardnih protokola na bazi TCP/IP, poput HTTP. Sa poslednjom realizacijom Java platforme, možete pozivati metode lako i neosetno na udaljenoj mašini kao što možete one iz istog radnog prostora, preko takvih zajedničkih protokola kao što je CORBA (Commont Object Request Broker Architecture) i RMI (Remote Method Invocation), kao i nedavno dodatih Web servisa. Za svaki protokol, sistem automatski odrađuje transport umesto Vas.

Interpretivnost

Java programi su *interpretivni*. Umesto da budu kompajlirani u matične izvršive komponente, Java kod se prevodi u platformski-nezavisan bajt kod. Ovaj bajt kod može zatim biti prenesen na bilo koju platformu koja ima Java izvršno okruženje (Java Runtime Environment; JRE), koje se sastoji od Java virtualne mašine (Java Virtual Machine; JVM), i prema tome može da se izvršava bez ponovnog prevođenja i ponovnog povezivanja. Ovakav proces može nagoveštavati da je Java nerazdvojivo spora, ali — kao što će biti objašnjeno u dolazećem odeljku "Performanse" — to ne mora biti slučaj. Platformski-nezavisan bajt kod zapravo sadrži dodatne informacije koje se mogu koristiti za optimizaciju izvršenja za vreme rada, na osnovu odluka koje je nemoguće doneti za vreme kompajliranja.

Robustnost

Robustnost je mera pouzdanosti programa. Java ima nekoliko ugrađenih mogućnosti koje poboljšavaju pouzdanost programa:

- Java je strogo tipovan jezik. Kompajler i učitavač klasa proveravaju ispravnost svih poziva metoda, sprečavajući nesporazume podrazumevanog tipa i nekompatibilnost verzija.
- Java nema pokazivače. Ne možete usmeriti pokazivač u memoriju, slučajno pokvariti memoriju ili nastaviti dalje od kraja niza. U Javi, umesto pokazivača postoje reference; one ne mogu biti dereferencirane da bi direktno manipulisali sa memorijskim prostorom. (Izuzetak je video memorija, koja može biti više direktno kontrolisana počev od Jave 1.4.)
- Java izvodi automatsko sakupljanje otpada. Programeri mogu slučajno zaboraviti da oslobode memoriju i ne moraju brinuti o tome da moraju znati kada memorija mora da se oslobodi.
- Java ohrabruje upotrebu interfejsa umesto klasa. Kao što ćete naučiti u poglavlju 5, interfejsi definišu grupu ponašanja, a klase primenjuju to ponašanje. Umesto da se klase prenose okolo, prenose se interfejsi, prema tome skrivajući implementaciju. Ukoliko se menja implementacija, kao što se obično dešava, onda dokle god nova klasa primenjuje stari interfejs, sve ostalo će i dalje raditi normalno.

Bezbednost

Java ima mnoge ugrađene bezbednosne karakteristike, dodajući još i više standardnih karakteristika sa realizacijom 1.4. Mnoge stvari koje Javu čine robustnom takođe osiguravaju bezbednost izvršavanja programa — na primer, nema pokazivača memorije, tako da nema oštećenja memorije. Ostale bezbednosne karakteristike su ugrađene u izvršni model: verifikator bajt koda, učitavač klasa i menadžer bezbednosti. Oni osiguravaju bezbednost izvršavanja koda i sprečavaju da sumnjivi kod izvodi poverljive operacije kao što su čitanje vašeg hard diska. Krajnji aspekt bezbednosti se odnosi na proveru autentičnosti, autorizaciju i kodiranje zbog zaštite privatnosti i obezbeđivanja integriteta podataka.

Verifikator bajt koda obezbeđuje prvi nivo odbrane na frontu bezbednosti. U suštini, on kaže "Ne veruj kompajleru." Pre nego što je klasa učitana u radno okruženje, mora biti proveren njen bajt kod. Ukoliko neko koristi editor nivoa bajta na fajlu i modifikuje bajtove u pokušaju da pristupi nečemu

čemu ne bi trebalo, verifikator će to otkriti i sprečiti učitavanje klase i izvršavanje programa. Zapravo, programi koji se izvršavaju u JRE čak ne moraju biti pisani u Java programskom jeziku.

Sledeći u kategorizaciji bezbednosti dolazi *učitavač klasa*. Kao što njegovo ime nagoveštava, učitavač klasa je odgovoran za učitavanje klasa. Na frontu bezbednosti, učitavač klasa je odgovoran za učitavanje klasa i za to da ih drži odvojenim od sumnjivih posmatrača. Prvo, učitavač klasa održava lokalne klase i klase učitane preko mreže u različitim prostorima imena, obezbeđujući da klase koje su učitane sa mreže ne mogu zameniti lokalne klase sistema. Drugo, klase učitane iz različitih lokacija mreže se takođe drže odvojeno. Ovo obezbeđuje da klasa koja ne pripada sistemu ne može da se pretvara da je sistemska; niti da je bliska sa klasom učitanom sa nekog trećeg Web sajta.

Treća bezbednosna komponenta radnog okruženja je *menadžer bezbednosti*, koji primenjuje polisu bezbednosti za radno okruženje. Svi zadaci kojima je potrebna dozvola da bi uradili nešto prolaze kroz menadžera bezbednosti. Ukoliko zadatak ne može da dobije dozvolu od menadžera bezbednosti, onda se on i ne izvršava. Ovo je posebno važno za kod koji je učitan preko Interneta sa neproverenih Web sajtova. Takav kod koji se izvršava u pretraživaču Weba se naziva aplet. On se izvršava u oblasti pretraživača Weba nazvanoj kutija sa peskom (*sandbox*) (bezbedno mesto za izvršavanje, ali bez pristupa okruženju klijenta).

Počev od Jave 1.1 i do (uključujući) Jave 1.4 dodato je više bezbednosnih karakteristika setu Core Java API. Postoji podrška za razvojne programere da potpišu Java klase digitalnim potpisom, dopuštajući korisnicima da veruju u njihov integritet dovoljno da izađu iz kutije sa peskom. Postoji podrška za generisanje pregleda poruka da bi se uverili u njihov integritet. Otkrićete podršku za menadžment sertifikata, liste kontrole pristupa, SSL/TLS (Secure Sockets Layer / Transport Layer Security), i čak i za biblioteke kodiranja koje se mogu izvoziti.

Sve ove mogućnosti koje se odnose na bezbednost su prisutne da bi obezbedile uspešno konstruisanje i izvršavanje distribuiranih, mrežnih programa.

Nezavisnost od arhitekture

"Nezavisost od arhitekture" se odnosi na Javin platformski-nezavisan bajt kod. Umesto da je kompajliran u binarni kod specifičan za platformu, Java programi su napravljeni da bi se izvršavali bilo gde bez ponovnog kompajliranja ili ponovnog povezivanja. Ukoliko kompanija razvije novi hardver, ne mora odbaciti svoj softver u koji je investirala, već je samo potrebno da postavi JRE na novu platformu. I, ukoliko potpuno nova kompanija razvije potpuno novi hardver ili novi operativni sistem, ne mora krenuti od totalne nule bez softvera. Uz dodatak samo JRE, novodizajnirana platforma može pokretati sve Java programe.

Zapravo, to je baš ono što je kompanija nazvana SavaJe (`www.savaje.com`) uradila. SavaJe je napravila novi operativni sistem za Compaq iPAQ i druge 32-bitne StrongARM-bazirane informacione uređaje. Umesto da mora da obučava razvojne programere da prave aplikacije za novi operativni sistem, oni su bili u mogućnosti da preuzmu postojeće Java aplikacije i da ih jednostavno postave na uređaj i da one rade dobro (iako verovatno imaju ekran manji od očekivanog).

Prenosivost

Možda ste čuli, a možda i niste Java mantru: Write Once, Run Anywhere (WORA) (piši jednom, pokreći bilo gde). Cilj Jave je da bude u mogućnosti da isti program pokrene na bilo kojoj arhitekturi. Mada, ova *prenosivost* nije postignuta samo platformskom nezavisnošću bajt koda. Umesto da se oslanja na detalje specifične za implementaciju, kao što je koliko je `int` veliki, sve brojne prezentacije u veličini, redosledu bajtova, i manipulaciji su definisane u specifikaciji Java jezika (*Java Language Specification*, `http://java.sun.com/docs/books/jls/`).

Visoke performanse

Možda mislite da je postojanje reči *interpretiranje* i *visoke performanse* u istoj rečenici oksimoron. Međutim, platformski-nezavisan bajt kod može zapravo biti preveden za vreme rada na mašinski kod karakterističan za CPU, izvršavajući se približno brzo, ako ne i brže od matično kompajliranog C i C++ koda. Dva takva alata za prevođenje za vreme rada koji su uključeni u Javu automatski to rade umesto Vas. Alat prve generacije se naziva Just in Time (JIT) kompajler. Sunov alat druge generacije se naziva HotSpot. U suštini, HotSpot i JIT kompajler rade istu stvar: prevođenje za vreme rada u matični set instrukcija. HotSpot, međutim, takođe nadgleda kod dok se izvršava i izvodi optimizaciju brzine na često izvršavanim blokovima, umesto na svemu.

NAPOMENA

Za poređenje sa različitim Java radnim platformama, istražite Volano Report (www.volano.com/report/). ■

Višenitnost

Nit možete *shvatiti* kao kontekst izvršavanja. Na primer, kada surfujete Internetom, Vaš pretraživač Weba će prikazivati novu Web stranu dok u pozadini nastavlja preuzimanje slika. Svaki od ovih zadataka može biti urađen u posebnoj niti. Kada Vaši programi simultano izvršavaju više zadataka ili niti, za njih se kaže da su višenitni.

Višenitni programi dele memoriju i moraju komunicirati između niti. Java programski jezik i standardne biblioteke uključuju mnoga sredstva da bi Vam pomogao u ovom procesu komunikacije, obezbeđujući bezbednost niti. Više o pravljenju višenitnih programa bezbednih niti ćete naučiti u poglavlju 8.

Dinamički jezik

Poslednja od žargonskih reči karakterističnih za Javu kaže da je Java *dinamički* jezik. Ovo ne znači da jezik konstantno evoluira; fraza ima više veze sa bibliotekama. Konkretno, to znači da dok se biblioteke vremenom menjaju, programi ne moraju biti ponovo povezivani. Raniji, platformski-nezavisan bajt kod, će nastaviti da radi nakon što su nove biblioteke realizovane, bilo da zato što je Sun izdao novu realizaciju Jave, ili što je neka druga kompanija izdala realizaciju nove biblioteke. Dokle god korišćeni delovi biblioteke nisu odbačeni, programi će i dalje raditi, čak i ako se biblioteci dodaju novi delovi. Ova dinamička priroda takođe važi za Javin prioritet interfejsa nad klasama, kao što je pomenuto u diskusiji o robustnosti.

Razumevanje Microsoftovog gledišta

Izgleda da bi knjiga o Javi trebalo da zahteva neku izjavu o tome gde se Java uklapa u Microsoftove planove. U osnovi, bez dvoličnog pokušaja da se Visual J++ razvojni programeri podstaknu da nastave dalje, ne uklapa se uopšte. Proces *Sun protiv Microsofta* je okončan Januara 2001. Kao deo poravnanja, Microsoft može nastaviti da isporučuje svoje poslednje JRE, ukoliko tako odluči. Javile su se tri posledice: Prvo, poslednje okruženje koje je nudio Microsoft je bazirano na Javi 1.1.4 — da, pre tri realizacije, i čak je i starije ukoliko uzmete u obzir da je poslednje Sunovo izdanje 1.1 bilo 1.1.8. Drugo, kao kod Windowsa XP, Microsoftov JRE se više ne isporučuje uz operativni sistem. Treće, Microsoft nudi Visual J#.NET, alat na bazi Jave 1.1.4 koji podržava Java programski jezik, ali ne i platformu, umesto koje podržava Microsoftovu .NET arhitekturu.

Da li je to važno? Tehnički govoreći, zavisi. Naravno, bilo je lepo imati bilo kakvo radno okruženje na svakom klijent računaru. Međutim, upotreba stare i nekompatibilne verzije JRE je izazvalo mnoge probleme i glavobolje. Sun će nastaviti da besplatno stavlja na raspolaganje najnovije radno okruženje za Windows platforme. Korisnici samo treba da ih posebno preuzmu, osim ukoliko Sun ne uspe da nagovori snabdevače računarskog hardvera da ga preinstaliraju. (Za Compaq i neke druge snabdevače hardvera, ovo je već objavljeno: Compaq će isporučivati Sunovo radno okruženje uz sve XP boxove.) Microsoftova poslovna odluka je bila da jednostavno napusti podršku za standardnu Javu na Windows platformi. Ukoliko Vam se svidelo njihovo Visual J++ proširenje i želite tesnu vezu sa Windowsom, najbolje je da pređete na C# nego da pređete na njihov poslednji Java derivat.

NAPOMENA

C#, izgovara se "C-sharp," je kreacija Anders Hejlsberga za rad sa Microsoft .NET platformom. ■

Dokle god Java radi na serveru, nedostatak Microsoftovog radnog okruženja nije smetnja. Aplikacioni serveri teže da dođu sa sopstvenim radnim okruženjem i lako ih je nadograditi ukoliko vam se ne sviđa ono koje obezbeđuju.

Idemo dalje

U leto 2001, DevX je objavio analizu o stanju Jave nakon šest godina postojanja (`www.devx.com/judgingjava/articles/sixyears`). Ukolio krenete putanjom ka oblasti produktivnog Java razvojnog programera, ovaj članak bi trebalo da bude koristan. Možete pročitati rezimea nekih DevXovih saznanja, zasnovanih na glasanju 2,600 Java razvojnih programera.

Kao prvo, alati su važna komponenta Java razvoja. Međutim, niko nije kompletno zadovoljan onim što postoji. U gotovo svim slučajevima, manje od 50 procenata razvojnih programera je potpuno zadovoljno sa bilo kojim od alata. Međutim, ukoliko volite da sledite većinu, šest najčešće korišćenih alata sa kojima su korisnici najviše zadovoljni u vreme pisanja članka su bili JBuilder kompanije Borelanda, WebLogic kompanije BEA, VisualAge for Java kompanije IBM, WebGain Studio/Visual Cafe kompanije WebGain, CodeWarrior kompanije MetroWorks, i Forte for Java kompanije Sun. Van ovih, postoji određen procep u raspodeli na tržištu ostalih zadovoljavajućih alata.

Druga zanimljiva statistika iz izveštaja DevXa je o trendovima tehnologija/jezika. Od svih napravljenih aplikacija, DevX kaže da je 43 procenata navodno napravljeno upotrebom Java tehnologija. Visual Basic dolazi na drugo mesto sa 25 procenata, i C++ oko 22 procenata. (C# je ima oko 2.8 procenata za buduće planirane projekte, ali ne i za aktuelne.) Voleo bih da mislim da je Javinih 43 procenata broj koji bi trebalo da ohrabri ljude da uče Javu i napuste Visual Basic i C++, ali moram dobro da razmislim o veličini ovog procenta, obzirom na probni uzorak. Na prvi pogled, razvojni programeri koji su glasali su već bili Java razvojni programeri. Njihova kompanija je već odlučila da započne naizgled nesiguran posao. Ovo znači da kompanije bez Java razvojnih projekata verovatno nisu učestvovale u ispitivanju. Premda, za poređenje Bloor Interactive je izvestio Februara 2000 da "Java prestiže C++ kao najtraženija programerska veština." U tom izveštaju, oni su pregledali oglase za potražnju posla. Java i C++ su vodili mrtvu trku sa po 36 procenata od oko 350,000 Internet oglasa za posao sakupljenih u periodu od 12 meseci, a VB je bio za oko 10 procenata niži. Tako, uzimajući u obzir distancu od godinu dana između dva izveštaja, 43 procenata DevX nisu daleko od prave vrednosti.

DevX izveštaj je zaključen listom oblasti sa potrebnim veštinama. Kako saznajete više o Javi, razmislite o sticanju dubljeg znanja u zahtevnijim oblastima Java tehnologije: pristup bazama podataka, midlveru, XML, i pravljenje skriptova na strani servera.

blanko