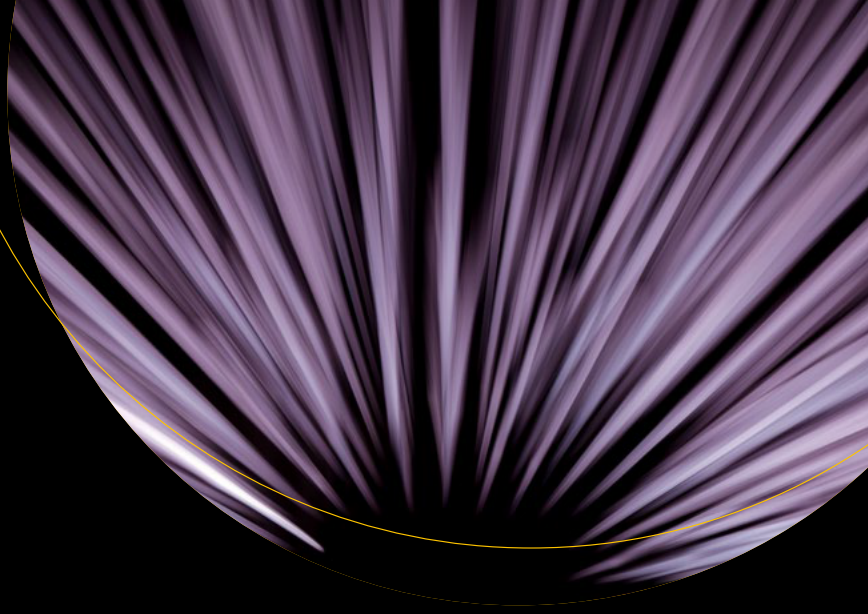

PHP 8 objekti, obrasci i praksa

Ovladajte OO poboljšanjima, projektnim obrascima i glavnim razvojnim alatima

—

Prevod šestog izdanja

—

Matt Zandstra

Apress®

PHP 8
objekti, obrasci i praksa

objektno orijentisan pristup

Matt Zandstra

Prevod VI izdanja

kompjuter biblioteka
Apress®

Izdavač:

kompjuter biblioteka

Obalskih radnika 4a, Beograd

Tel: 011/2520272

e-mail: kombib@gmail.com

internet: www.kombib.rs

Urednik: Mihailo J. Šolajić

Za izdavača, direktor:
Mihailo J. Šolajić

Autor: Matt Zandstra

Prevod: Slavica Prudkov

Lektura: Nemanja Lukić

Slog: Zvonko Aleksić

Znak Kompjuter biblioteke:
Miloš Milosavljević

Štampa: „Pekograf“, Zemun

Tiraž: 500

Godina izdanja: 2021.

Broj knjige: 545

Izdanje: Prvo

ISBN: 978-86-7310-568-0

PHP 8 Objects, Patterns, and Practice: Mastering OO Enhancements, Design Patterns, and Essential Development Tools

Matt Zandstra

ISBN 978-1-4842-6790-5

O AUTORU

Matt Zandstra ima više od dve decenije iskustva kao veb programer, savetnik i pisac. Autor je knjige *SAMS Teach Yourself PHP in 24 hours* (tri izdanja) i saradnik je na projektu *DHTML Unleashed*. Pisao je, između ostalog, članke za *Linux Magazine*, Zend, IBM DeveloperWorks i *php|architect*. Matt je radio kao stariji programer / lider razvojnog tima u Yahoo!-u i kao lider razvojnog tima za API u LoveCrafts-u. Matt radi kao savetnik i kompanijama pruža savete o njihovim arhitekturama i upravljanju sistemima, a takođe razvija sisteme prvenstveno pomoću PHP-a i Java-e. Matt takođe piše beletristiku.

O tehničkom recenzentu

Paul Tregoing je skoro 20 godina radio u oblasti IT-a i razvoja u raznim okruženjima. Pet godina je radio u Yahoo!-u kao stariji programer u timu za naslovne stranice; tamo je generisao svoj prvi PHP pomoću Perl-a. Među ostalim poslodavcima su Bloomberg, Schlumberger i British Antarctic Survey, gde se zbližio sa hiljadama "pingvina".

Sada radi kao slobodan inženjer za razne klijente, male i velike, i gradi višeslojne veb aplikacije korišćenjem PHP-a, JavaScript-a i mnogih drugih tehnologija. Paul je veliki ljubitelj naučne fantastike i fantazije i ima tajnu ambiciju da se u bliskoj budućnosti okuša u pisanju. Živi u Kembridžu, Velikoj Britaniji, sa suprugom i decom.

Zahvalnice

Kao i uvek, imao sam podršku mnogih ljudi dok sam radio na ovom izdanju. Ali kao i uvek, moram se osvrnuti i na početak knjige. Isprobao sam neke od osnovnih koncepata ove knjige u govoru u Brajtonu, još dok smo se svi prvi put čudili sjajnim mogućnostima PHP-a 5. Zahvaljem se Andiju Budu, koji je bio domaćin razgovora, i živoj zajednici programera u Brightonu. Hvala i Jesi Vajt-Cinisu, koji je bio na tom sastanku i koji me je povezao sa Martinom Streicherom u Apressu.

Još jednom, i ovog puta, tim Apress-a pružio je ogromnu podršku, povratne informacije i ohrabrenje. Srećan sam što sam imao takvu profesionalnu podršku.

Imam veliku sreću što je moj prijatelj i kolega Pol Tregoing radio na ovom izdanju kao tehnički recenzent. Činjenica da je sam PHP bio u aktivnom razvoju tokom pisanja ove knjige zahtevala je dodatnu opreznost. Primeri kodova koji su bili savršeno validni u ranim radnim verzijama prikazani su netačno zbog brzog razvoja jezika. Još jednom, Polovo znanje, uvid i obraćanje pažnje na detalje bilo je od velike koristi za ovo izdanje - veliko hvala Polu!

Hvala i mojoj supruzi, Luisi. Pisanje ove knjige se poklopilo sa tri zaključavanja zbog pandemije, pa se zahvaljujem i našoj deci, Holi i Džejku, na mnogo potrebnih distrakcija - koje su se često dešavale tokom Zoom sastanaka u mom kancelarijskom prostoru (ugao kuhinjskog stola).

Zahvaljem se Stivenu Metskeru na ljubaznoj dozvoli da u PHP-u primenim pojednostavljenu verziju API-ja za raščlanjivanje koji je on predstavio u svojoj knjizi *Building Parsers with Java* (Addison-Wesley Professional, 2001).

Pišem uz muziku i, u prethodnim izdanjima ove knjige, setio sam se sjajnog di-džeja, Džona Pila, šampiona andergraunda i eklektike. Zvuk za ovo izdanje uglavnom je obezbedila savremena muzička emisija BBC Radio 3, *Late Junction*, koja se emituje bez prekida. Hvala im što su i dalje čudni.

UVOD

Kada sam prvi put osmislio ovu knjigu, objektno-orijentisan dizajn u PHP-u bio je ezoterična tema. Tokom godina koje su usledile, ne samo da je došlo do ogromnog uspona PHP-a kao objektno-orijentisanog jezika već i do velikog uspona radnih okvira. Naravno, radni okviri su neverovatno korisni. Oni upravljaju samom osnovom većine veb aplikacija. Štaviše, oni često precizno predstavljaju principe dizajna koji se u ovoj knjizi istražuju.

Međutim, ovde postoji opasnost za programere, kao što postoji i u svim korisnim API-ima. To je strah da bi se neko mogao naći u korisničkoj zemlji, primoran da sačeka da udaljeni gurui isprave greške ili dodaju funkcije po njihovoj volji. Kratak je put od ovog stanovišta do svojevrsnog izgnanstva, u kome se ostavlja da se unutrašnjost okvira smatra naprednom magijom, a sopstveni rad ne više od sitnog ukrasa zaglavljenog povrh moćne nepoznate infrastrukture.

Iako volim inovacije, moj argument ne podrazumeva da bi svi trebalo da odbacimo radne okvire i pravimo MVC aplikacije od nule (barem ne uvek). To znači da bi kao programeri trebalo da razumemo probleme koje radni okviri rešavaju i strategije koje se koriste za njihovo rešavanje. Trebalo bi da ocenjujemo radne okvire ne samo funkcionalno, već i u smislu dizajnerskih odluka koje su njihovi autori doneli i da sudimo o kvalitetu njihovih implementacija. I da, kada su uslovi pogodni, trebalo bi da kreiramo sopstvene rezervne i fokusirane aplikacije i da vremenom kompajliramo sopstvene biblioteke koda za višekratnu upotrebu.

Nadam se da će ova knjiga na neki način pomoći PHP programerima da primene otkrića orijentisana ka dizajnu na svoje platforme i biblioteke i da će pružiti neke konceptualne alate koji su potrebni kada dođe vreme da programer radi samostalno.

Postanite član Kompjuter biblioteke

Kupovinom jedne naše knjige stekli ste pravo da postanete član Kompjuter biblioteke. Kao član možete da kupujete knjige u pretplati sa 40% popustai učestvujete u akcijama kada ostvarujete popuste na sva naša izdanja. Potrebno je samo da se prijavite preko formulara na našem sajtu.

Link za prijavu: http://bit.ly/2TxekSa

Skenirajte QR kod
registrujte knjigu
i osvojite nagradu

DEO I

Objekti

POGLAVLJE 1

PHP: Dizajn i upravljanje

U julu 2004. godine objavljen je PHP 5.0. Ovom verzijom predstavljen je niz radikalnih poboljšanja. Možda je prvo među njima bilo radikalno poboljšana podrška za objektno-orijentisano programiranje. To je podstaklo veliko interesovanje za objekte i dizajn u PHP zajednici. U stvari, to je predstavilo intenziviranje procesa koji je započeo kada je prvi put u verziji 4 objektno-orijentisano programiranje pomoću PHP-a postalo ozbiljna stvarnost.

Ovim poglavljem obuhvaćeni su neki zahtevi koji mogu biti rešeni kodiranjem pomoću objekata. Ukratko ću rezimirati neke aspekte razvoja obrazaca i srodne prakse.

Takođe ću istaći teme koje ova knjiga obuhvata. Obuhvaćene su sledeće teme:

- *Evolucija katastrofe*: Projekat ide u lošem smeru
- *Dizajn i PHP*: Kako su tehnike objektno-orijentisanog dizajna zaživele u PHP zajednici
- *Ova knjiga*: Objekti, obrasci, praksa

Problem

Problem je to što je PHP previše jednostavan. Iskušava vas da isprobate svoje ideje i laska vam dobrim rezultatima. Veliki deo koda pišete direktno na veb stranicama, jer je PHP dizajniran da to podrži. Fajlovima koji mogu da se uključuju od stranice do stranice dodajete pomoćne funkcije (kao što je pristupni kod baze podataka) i, pre nego što ste i svesni toga, imate funkcionalnu veb aplikaciju.

Na dobrom ste putu da sve upropastite. To naravno ne shvatate, jer vaš veb sajt izgleda fantastično. Dobro funkcioniše, vaši klijenti su zadovoljni, a korisnici troše novac.

Problemi nastaju kada se vratite na kod da biste započeli novu fazu. Sada imate veći tim, više korisnika i veći budžet. Ipak, bez upozorenja, sve počinje da ide u lošem smeru. Kao da je vaš projekat zatrovan.

Vaš novi programer se muči da razume kod koji je vama potpuno razumljiv, iako možda pomalo komplikovan u svojim preokretima. Potrebno joj je više vremena nego što ste očekivali da, kao član tima, dostigne punu snagu.

Za jednostavnu promenu, za koju je procenjeno da zahteva jedan dan posla, potrebno je tri dana, a tada otkrivate da, kao rezultat promene, morate da ažurirate 20 ili više veb stranica.

Jedan od vaših kodera snima svoju verziju fajla preko većih promena koje ste ranije uneli u isti kod. Gubitak ne otkrivate tri dana, a do tada ste već izmenili svoju lokalnu kopiju. Potreban je ceo dan da se sredi nered, što zadržava trećeg programera, koji je takođe radio na fajlu.

Zbog popularnosti aplikacije, potrebno je da prebacite kod na nov server. Projekat mora da se instalira ručno i otkrivate da su putanje fajla, nazivi baza podataka i lozinke uneti u kod u mnogim izvornim fajlovima. Zaustavljate posao tokom premeštanja jer ne želite da prepišete promene konfiguracije koje migracija zahteva. Procenjeno je da posao zahteva dva sata ali to postaje osam sati, jer otkrivate da je neko uradio nešto pametno uključivanjem Apache modula ModRewrite, a aplikacija ga sada zahteva da bi pravilno funkcionisala.

Napokon pokrećete fazu 2. Dan i po je sve u redu. Prvi izveštaj o grešci stiže upravo kad se spremate da napustite kancelariju. Klijent vas poziva telefonom nekoliko minuta kasnije da se požali. Njen izveštaj je sličan prvom, ali malo detaljnije ispitivanje otkriva da je reč o drugoj grešci koja uzrokuje slično ponašanje. Prisećate se jednostavne promene na početku faze, koja je zahtevala opsežne modifikacije u ostatku projekta.

Shvatate da nisu izvršene sve potrebne modifikacije. Razlog je ili to što su izostavljene na početku ili to što su dotični fajlovi prepisani prilikom kolizije spajanja. Užurbano vršite izmene potrebne za ispravljanje grešaka. Previše žurite da biste testirali promene, ali one uključuju jednostavno kopiranje i pejstovanje, pa šta može da pođe po zlu?

Sledećeg jutra stižete u kancelariju da biste otkrili da modul korpe za kupovinu ne funkcioniše celu noć. U promenama koje ste izvršili u poslednjem trenutku nedostaje vodeći navodnik, što kod čini neupotrebljivim. Naravno, dok ste spavali, potencijalni kupci u drugim vremenskim zonama bili su budni i spremni da potroše novac u vašoj prodavnici. Rešavate problem, smirujete klijenta i okupljate tim za još jedan dan borbe.

Ova svakodnevna priča o kodiranju može da se učini malo preteranom, ali viđao sam da se sve to ponavlja iznova i iznova. Mnogi PHP projekti svoj život započinju kao mali i evoluiraju u čudovišta.

S obzirom na to da prezentacioni sloj takođe sadrži i logiku aplikacije, dupliranja su od samog početka uključena kao upiti baze podataka, provere autentifikacije, obrada obrazaca i još mnogo toga se kopira sa stranice na stranicu. Svaki put kada je potrebna promena jednog od ovih blokova koda, mora biti izvršena svuda gde se kod nalazi, ili će se sigurno javiti greške.

Nedostatak dokumentacije otežava čitanje koda, a nedostatak testiranja omogućava da nejasne greške ostanu neotkrivene do samog raspoređivanja. Promenljiva priroda klijentovog poslovanja često podrazumeva da se kod razvija u potpuno drugom smeru od svoje prvobitne svrhe i da izvršava zadatke za koje u osnovi nije prikladan. S obzirom na to da je takav kod često evoluirao kao nerazumljiva kombinacija, teško je, ako ne i nemoguće, isključiti ga i prepisati njegove delove da bi odgovarali novoj svrsi.

Sada, ništa od ovoga nije loša vest ako ste slobodan PHP savetnik. Procenom i ispravljanjem takvog sistema možete da finansirate skupa espresso pića i DVD setove šest meseci ili više. Međutim, problemi ove vrste mogu da predstavljaju razliku između uspeha i neuspeha poslovanja.

PHP i drugi jezici

Fenomenalna popularnost PHP-a značila je da su njegove granice testirane rano i detaljno. Kao što ćete videti u sledećem poglavlju, PHP je započeo život kao skup makroa za upravljanje ličnim početnim stranicama. Pojavom PHP-a 3 i, u većoj meri, PHP-a 4, jezik je brzo postao uspešan pokretač velikih poslovnih sajtova. Međutim, nasleđe PHP-ovih početaka preneto je u dizajn skriptova i u upravljanje projektima. U nekim krajevima PHP je zadržao nepravednu reputaciju kao hobistički jezik, najprikladniji za zadatke prezentacija.

Otprilike u to vreme (početkom milenijuma), nove ideje su sticale vrednost u drugim kodnim zajednicama. Interesovanje za objektno-orijentisan dizajn podstaklo je Java zajednicu. S obzirom na to da je Java objektno-orijentisan jezik, možda mislite da je to suvišno. Naravno, Java pruža granularnost sa kojom je lakše raditi, ali korišćenje klasa i objekata samo po sebi ne određuje specifičan pristup dizajnu.

O konceptu projektnih obrazaca kao načinu opisivanja problema, zajedno sa suštinom njegovog rešenja, prvi put se govorilo 1970-ih. Ideja je potekla iz oblasti arhitekture, što je možda prikladno, a ne računarstva, u originalnom delu

Kristofera Aleksandera: *A Pattern Language* (Oxford University Press, 1977). Početkom 1990-ih, objektno-orijentisani programeri koristili su istu tehniku za imenovanje i opisivanje problema u dizajniranju softvera. Prvobitna knjiga o projektnim obrascima, *Design Patterns: Elements of Reusable Object-Oriented Software* (Addison-Wesley Professional, 1995), autora Erika Game, Ričarda Helma, Ralfa Džonsona i Džona Vlisidesa (koje ćemo u ovoj knjizi nazivati nadimkom, *Gang of Four*), i danas je neophodna. Obrasci koje sadrži ova knjiga su neophodan prvi korak za svakog ko započinje rad u ovoj oblasti, pa je zbog toga većina obrazaca u ovoj knjizi izvedena iz te knjige.

Sam Java programski jezik primenjuje veliki broj osnovnih obrazaca u svom API-ju, ali su tek krajem 1990-ih projektni obrasci prodrli u svest programerske zajednice u celini. Obrasci su brzo zarazili odeljke računarskih nauka u Main Street knjižarama, a prvi proboji su počeli na mejling listama i forumima.

Bez obzira na to da li mislite da su obrasci moćan način prenošenja zanatskog znanja ili uglavnom preterivanje (a s obzirom na naslov ove knjige, verovatno možete pogoditi moje mišljenje po tom pitanju), teško je poreći da je naglasak na dizajniranju softvera koji su oni podstakli sam po sebi koristan.

Srodne teme takođe su sve istaknutije. Među njima je i ekstremno programiranje (XP), za koje se zalagao Kent Beck. XP je pristup projektima koji podstiče fleksibilno i fokusirano planiranje i izvršenje orijentisano ka dizajnu.

Među principima XP-a ističe se da je testiranje od ključnog značaja za uspeh projekta. Testovi bi trebalo da budu automatizovani, da se pokreću često, a poželjno je da budu dizajnirani pre nego što je napisan njihov ciljni kod.

XP takođe nalaže da je projekte potrebno razdvojiti na male (vrlo male) iteracije. I kod i zahteve potrebno je stalno nadzirati. Arhitektura i dizajn bi trebalo da budu deljeno i konstantno pitanje, što dovodi do čestih revizija koda.

Ako je XP bio militantno krilo dizajnerskog pokreta, onda umerenu tendenciju dobro predstavlja jedna od najboljih knjiga o programiranju koje sam ikad pročitao: *Pragmatic Programmer: From Journeyman to Master*, autora Andru Hanta i Davida Tomasa (Addison-Veslei Professional, 1999).

Neki su XP smatrali pomalo kultnim, ali je tokom dve decenije izrastao u objektno-orijentisanu praksu najvišeg nivoa, a njegovi principi su u širokoj primeni. Konkretno, revizija koda poznata kao refaktorisanje, koristi se kao moćan dodatak obrascima. Refaktorisanje je evoluiralo od 1980-ih, ali je kodifikovano u katalogu refaktoringa *Martina Faulera*, *Refactoring: Improving the Design of Existing Code* (Addison-Wesley Professional), objavljenog 1999. godine i definisalo je tu oblast.

Popularizacijom XP-a i obrazaca, testiranje je takođe postalo popularno pitanje. Važnost automatizovanih testova dodatno je istaknuta izdavanjem moćne testne platforme JUnit, koja je postala ključno oružje u oružarnici Java programera. Poznat članak na tu temu „Test Infected: Programmers Love Writing Tests" autora Kenta Beka i Erika Gamea (http://junit.sourceforge.net/doc/testinfected/testing.htm) pruža odličan uvod u temu i izuzetno je uticajan.

PHP 4 je objavljen otprilike u to vreme, donoseći sa sobom poboljšanja u efikasnosti i, što je najvažnije, poboljšanu podršku za objekte. Ova poboljšanja su u potpunosti omogućila objektno-orijentisane projekte. Programeri su prihvatili ovu funkciju, što je pomalo iznenadilo osnivače Zend-a, Zeva Suraskog i Andija Gutmansa, koji su se pridružili Rasmusu Lerdorfu u upravljanju razvojem PHP-a. Kao što ćete videti u sledećem poglavlju, PHP-ova podrška za objekte nije bila savršena. Ali uz disciplinu i pažljivu upotrebu sintakse, zaista se moglo razmišljati istovremeno o objektima i PHP-u.

Uprkos tome, katastrofe u dizajnu, poput one koja je predstavljena na početku ovog poglavlja, i dalje su česte. Kultura dizajna je na neki način daleka i gotovo nepostojeća u knjigama o PHP-u. Međutim, na mreži je interesovanje bilo jasno. Leon Atkinson napisao je članak o PHP-u i obrascima za Zend 2001. godine, a Hari Fueks je pokrenuo svoj časopis na adresi www.phppatterns.com (sada nepostojeći) 2002. godine. Počeli su da se pojavljuju projekti radnih okvira zasnovanih na obrascima, poput BinaryCloud-a, kao i alati za automatizovano testiranje i dokumentaciju.

Prvo izdanje beta verzije PHP-a 5, 2003. godine, obezbedilo je budućnost PHP-a kao jezika za objektno-orijentisano programiranje. Zend Engine 2 je pružio znatno poboljšanu podršku za objekte i istakao da su objekti i objektno-orijentisan dizajn sada centralni za PHP projekat.

Tokom godina, PHP 5 se razvija i unapređuje uključivanjem značajnih novih funkcija, kao što su imenski prostori i zatvoreni izrazi. Tada je ujedno i stekao reputaciju najboljeg izbora za veb programiranje na strani servera.

PHP 7 objavljen je u decembru 2015. godine i predstavlja nastavak tog trenda. Konkretno, obezbedio je podršku za parametre i deklaracije vraćenog tipa - dve funkcije koje su mnogi programeri (zajedno sa prethodnim izdanjima ove knjige) godinama očekivali. Obuhvaćene su i mnoge druge funkcije i poboljšanja, uključujući anonimne klase, poboljšanu upotrebu memorije i povećanje brzine. Tokom godina jezik je postajao robusniji, čistiji i interesantniji za rad iz perspektive objektno-orijentisanog programera.

U decembru 2020. godine, skoro pet godina nakon izlaska PHP-a 7, PHP 8 je bio spreman za objavljivanje. Iako neki od detalja implementacije mogu da se promene (a malo su se promenili tokom pisanja ove knjige), funkcije su već dostupne u vreme pisanja ovog teksta (avgust 2020). Mnoge od njih ću ovde detaljno opisati. Promene uključuju poboljšanja u deklaraciji tipova, pojednostavljeno dodeljivanje svojstava i mnogo drugih novih funkcija. Možda je najvažniji dodatak podrška za atribute (koje na drugim jezicima nazivamo anotacije).

O ovoj knjizi

Ovom knjigom ne pokušavamo da otkrijemo nove horizonte u oblasti objektno-orijentisanog dizajna; umesto toga, ona nesigurno stoji na ramenima divova. Umesto toga, ispitaću, u kontekstu PHP-a, neke dobro uspostavljene principe dizajna i neke ključne obrasce (posebno one opisane u knjizi *Design Patterns*, klasiku autora Gang of Four). Na kraju, prelazim stroge granice koda da bih opisao alate i tehnike koji mogu da obezbede uspeh projekta. Pored ovog uvoda i kratkog zaključka, knjiga je podeljena na tri glavna dela: objekti, obrasci i praksa.

Objekti

Prvi deo započinjem brzim pregledom istorije PHP-a i objekata i grafičkim prikazom njihovog pomeranja, od naknadnih zamisli u PHP-u 3 do osnovne funkcije u PHP-u 5.

I dalje možete da budete iskusan i uspešan PHP programer sa malo ili nimalo znanja o objektima. Iz tog razloga polazim od prvih principa koji objašnjavaju objekte, klase i nasleđivanje. Čak i u ovoj ranoj fazi opisaću neka poboljšanja objekata koja su predstavljena u verzijama PHP 5, PHP 7 i PHP 8.

Kada utvrdimo osnove, detaljnije ću opisati našu temu ispitivanjem naprednijih objektno-orijentisanih funkcija PHP-a. Takođe ću posvetiti jedno poglavlje alatima koje pruža PHP kao pomoć u radu sa objektima i klasama.

Međutim, nije dovoljno da znate kako da deklarišete klasu i kako da je koristite za instanciranje objekta. Potrebno je prvo da izaberete prave učesnike za sistem i da odlučite koji su najbolji načini za njihovu interakciju. Te izbore je mnogo teže opisati i naučiti od samih činjenica o objektnim alatima i sintaksi. Završavam 1. deo uvodom u objektno-orijentisan dizajn pomoću PHP-a.

Obrasci

Obrazac opisuje problem u softverskom dizajnu i pruža jezgro rešenja. „Rešenje“ ovde ne podrazumeva vrstu isečenog i pejstovanog koda koji biste mogli da nađete u priručniku (odlične knjige koje programerima služe kao resursi). Umesto toga, projektni obrazac opisuje pristup rešavanju problema. Može da uključuje primer implementacije, ali je to manje važno od koncepta koji služi za ilustraciju.

Drugi deo knjige započinje definisanjem projektnih obrazaca i opisom njihove strukture. Takođe ću govoriti o nekim razlozima njihove popularnosti.

Obrasci imaju tendenciju da promovišu i slede određene osnovne principe dizajna. Njihovo razumevanje može da bude korisno pri analizi motivacije obrasca i može dobro da se primeni na sva programiranja. Govoriću o nekim od ovih principa. Takođe ću opisati objedinjeni jezik za modelovanje (Unified Modeling Language - UML), koji predstavalja način za opisivanje klasa i njihovih interakcija nezavisno od platforme.

Iako ova knjiga nije katalog obrazaca, ja ću ispitati neke od najpoznatijih i korisnih obrazaca. Opisaću problem koji svaki obrazac rešava, analiziraću rešenje i predstaviću primer implementacije u PHP-u.

Praksa

Čak će i lepo uravnotežena arhitektura biti neuspešna ako se njome ne upravlja pravilno. U 3. delu knjige opisaću dostupne alate koji će vam pomoći da kreirate radni okvir koji osigurava uspeh projekta. Prva dva dela knjige govore o praksi dizajniranja i programiranja, a 3. deo govori o praksi upravljanja kodom. Alati koje ću opisati mogu da formiraju strukturu podrške za projekat tako što pomažu u praćenju grešaka kada se one pojave, promovišu saradnju među programerima i pružaju jednostavnost instalacije i jasnoću koda.

Već sam govorio o snazi automatizovanog testa. Treći deo knjige započeću uvodnim poglavljem koje daje pregled problema i rešenja u ovoj oblasti.

Mnogi programeri greše što se prepuštaju impulsu da sve urade sami. Composer, zajedno sa Packagist-om, svojim glavnim skladištem, nudi pristup hiljadama paketa sa upravljanim zavisnostima koji veoma jednostavno mogu da se povežu u projekte. Opisaću kompromise između samostalnog implementiranja funkcije i raspoređivanja paketa Composer.

Dok obrađujem temu Composer-a, opisaću instalacioni mehanizam koji implementaciju paketa čini jednostavnom, izvršenjem jedne komande.

Kod podrazumeva saradnju. Ta činjenica može da bude korisna. Takođe može da bude noćna mora. Git je sistem za kontrolu verzija koji omogućava mnogim programerima da rade zajedno na istoj bazi kodova, bez međusobnog prepisivanja koda. To omogućava kreiranje snimaka projekta u bilo kojoj fazi razvoja, možete da vidite ko je izvršio koje promene i da razdvojite projekat na spojive grane. Git će vam jednog dana spasiti projekat.

Kada ljudi i biblioteke sarađuju, oni često uvode različite konvencije i stilove. Iako je to dobro, takođe može da ometa interoperabilnost. Reči kao što su *prilagođavanje* i *usaglašavanje* izazivaju u meni jezu, ali nesporno je da je kreativnost Interneta potkrepljena standardima. Prihvatanjem određenih konvencija slobodno možete da se igrate na nezamislivo ogromnom igralištu. Dakle, u novom poglavlju istražiću PHP standarde, kako mogu da nam pomognu, i kako i zašto bi trebalo da se *prilagođavamo*.

Dve činjenice se čine neizbežnim. Prvo, greške se često pojavljuju u istom delu koda, što neke radne dane čini već viđenim. Drugo, često poboljšanja kvare onoliko koliko ispravljaju, ili čak i više. Automatsko testiranje može da reši oba problema sistemom ranog upozorenja o problemima u kodu. Predstaviću PHPUnit, moćnu implementaciju takozvane xUnit testne platforme, koja je prvobitno bila namenjena za Smalltalk ali je preneta u mnogo drugih jezika, pa tako i u jezik Java. Posebno ću se osvrnuti na funkcije PHPUnit-a i na prednosti i neke troškove testiranja.

Aplikacije su neuredne. Možda će zahtevati instalaciju fajlova na nestandardne lokacije ili postavku baze podataka ili ispravku konfiguracije servera. Ukratko, aplikacije zahtevaju da se neke "*stvari*" obave tokom instalacije. Phing je verna luka Java alata Ant. Phing i Ant tumače izvršni fajl i obrađuju izvorne fajlove na bilo koji način, kako im vi to kažete. To obično podrazumeva kopiranje iz izvornog direktorijuma na različite ciljne lokacije sistema, ali, kako potrebe postaju složenije, Phing im se bez napora prilagođava.

Neke kompanije koriste razvojne platforme - ali u mnogim slučajevima timovi na kraju koriste niz različitih operativnih sistema. Saradnici stižu sa PC laptopovima (pozdrav Paulu Tregoingu, tehničkom uredniku petog i aktuelnog izdanja), neki članovi tima beskrajno hvale svoju omiljenu Linux distribuciju (to smo ja i moja Fedora), a mnogi koriste atraktivan PowerBook (za upotrebu u kafiću i sali za sastanke zbog kog uopšte ne izgledate kao samo još jedno čvorište u hipsterskoj Borgovoj vojsci). Svi oni će pokretati LAMP stek sa različitim stepenom lakoće. Međutim, u idealnom slučaju programeri bi trebalo da pokreću kod u okruženjima koja podsećaju na krajnji proizvodni sistem. Opisaću Vagrant, aplikaciju koja koristi virtuelizaciju tako da članovi tima mogu

da zadrže svoje razvojne platforme, ali da pokreću projektni kod na sistemu sličnom proizvodnom.

Testiranje i izrada su vrlo dobri, ali morate da instalirate i da pokrećete testove da biste iskoristili sve prednosti. Lako je postati zadovoljan sobom i pustiti da sve teče ako ne automatizujete izradu i testove. Opisaću neke alate i tehnike kategorije „kontinualna integracija“ koja će vam pomoći da to učinite.

Šta je novo u šestom izdanju

PHP je živ jezik i kao takav se neprekidno pregleda i razvija. I ovo novo izdanje je pregledano i detaljno ažurirano da bismo uzeli u obzir promene i nove mogućnosti.

Obuhvatiću nove funkcije, kao što su atributi i mnoga poboljšanja, u deklaracijama tipa. U primerima se koriste PHP 8 funkcije tamo gde je to prikladno, pa imajte na umu da će često biti potrebno pokretati kod u PHP 8 interpreteru - ili budite spremni da izvršite prelaz na stariju verziju.

Rezime

Ovo je knjiga o objektno-orijentisanom dizajnu i programiranju. Takođe, sadrži opise alata za upravljanje PHP bazom koda, od saradnje do raspoređivanja.

Ove dve teme isti problem rešavaju iz različitih, ali komplementarnih uglova. Primarni cilj je izgradnja sistema koji postižu svoje ciljeve i dobro podržavaju zajednički razvoj.

Sekundarni cilj leži u estetici softverskih sistema. Kao programeri gradimo mašine koje imaju oblik i akciju. Ulažemo mnogo sati svog radnog dana i mnogo dana svog života u oživljavanje ovih oblika. Želimo da alati koje gradimo, bile to pojedinačne klase i objekti, softverske komponente ili krajnji proizvodi, čine elegantnu celinu. Proces kontrole verzija, testiranja, dokumentovanja i izrade podržava ovaj cilj: to je deo oblika koji želimo da postignemo. Baš kao što želimo čist i pametan kod, želimo i bazu koda koja je dobro dizajnirana za programere i korisnike. Mehanika deljenja, čitanja i raspoređivanja projekta trebalo bi da bude jednako važna kao i sam kod.

POGLAVLJE 2

PHP i objekti

Objekti nisu uvek bili ključni deo PHP projekta. U stvari, PHP dizajneri su ih jednom opisali kao naknadnu zamisao.

Kako i sve naknadne ideje i ova se pokazala izuzetno otpornom. U ovom poglavlju predstaviću objekte rezimiranjem razvoja PHP-ovih objektno-orijentisanih funkcija.

Obuhvaćene su sledeće teme:

- *PHP / FI 2.0*: PHP, ali ne onakav kakvog ga mi poznajemo.
- *PHP 3*: Objekti se pojavljuju prvi put.
- *PHP 4*: Objektno-orijentisano programiranje se razvija.
- *PHP 5*: Objekti u srcu jezika.
- *PHP 7*: Popunjavanje praznine.
- *PHP 8*: Konsolidacija se nastavlja.

Slučajan uspeh PHP objekata

Uz PHP-ovu obimnu podršku za objekte i mnogo objektno-orijentisanih PHP biblioteka i aplikacija u opticaju, uspon objekta u PHP-u može da izgleda kao vrhunac prirodnog i neizbežnog procesa. U stvari, to je veoma daleko od istine.

Početak: PHP / FI

Postanak PHP-a kakvog ga danas poznajemo leži u dva alata koje je razvio Rasmus Lerdorf korišćenjem Perl-a. PHP je skraćenica od Personal Home Page Tools. FI je skraćenica od Form Interpreter. Zajedno su činili makroe za slanje SQL izraza u baze podataka, obradu obrazaca i kontrolu protoka.

Ovi alati su prepisani jezikom C i spojeni u alat pod nazivom PHP / FI 2.0. Jezik u ovoj fazi izgledao je drugačije od sintakse koju danas poznajemo, ali ne *toliko* drugačije. Postojala je podrška za promenljive, asocijativne nizove i funkcije. Međutim, o objektima se još nije ni razmišljalo.

Sintaksni slatkiš: PHP 3

U stvari, čak ni kada je PHP 3 bio u fazi planiranja, objekti nisu bili na dnevnom redu. Glavni arhitekti PHP-a 3 bili su Zeev Suraski i Andi Gutmans. PHP 3 je bio kompletno prerađen PHP / FI 2.0, ali objekti se nisu smatrali neophodnim delom nove sintakse.

Prema Zeevu Suraskiju, podrška za klase je dodata naknadno (tačnije 27. avgusta 1997). Klase i objekti zapravo su bili samo još jedan način za definisanje i pristup asocijativnim nizovima.

Naravno, dodavanjem metoda i nasleđivanja klase su postale mnogo više od proslavljenih asocijativnih nizova, ali i dalje su postojala ozbiljna ograničenja onoga što se moglo raditi sa klasama. Konkretno, niste mogli da pristupite promenjenim metodima roditeljske klase (ne brinite ako još ne znate šta to znači; objasniću kasnije). Još jedan nedostatak koji ću ispitati u sledećem odeljku je neoptimalan način prenošenja objekata u PHP skriptovima.

Da su objekti u ovom trenutku bili marginalno pitanje ističe njihov izostanak iz zvanične dokumentacije. U priručniku je objektima posvećena jedna rečenica i primer koda. Primer nije ilustrovao nasleđivanje, niti svojstva.

PHP 4 i tiha revolucija

Ako je PHP 4 bio još jedan revolucionaran korak za jezik, većina promena dogodila se ispod površine. Zend Engine (naziv je izveden od *Ze*ev i A*nd*i) napisan je od nule za pokretanje jezika. Zend Engine je jedna od glavnih komponenti koje pokreću PHP. Bilo koja PHP funkcija koju biste možda želeli da pozovete u stvari je deo sloja proširenja na visokom nivou. One izvršavaju posao za koji su imenovane, poput komunikacije sa API-jem baze podataka ili rukovanja znakovnim nizovima. Ispod toga, Zend Engine upravlja memorijom, prosleđuje kontrolu drugim komponentama i prevodi poznatu PHP sintaksu, sa kojom svakodnevno radite, u izvršni binarni kod. Za osnovne funkcije jezika, kao što su klase, potrebno je da zahvalimo Zend Engine-u.

Iz perspektive našeg *cilja*, glavna prednost je činjenica da je PHP 4 omogućio promenu roditeljskih metoda i pristup njima iz podređenih klasa.

Međutim, glavni nedostatak je ostao isti - dodeljivanje objekta promenljivoj, prosleđivanje funkciji ili vraćanje iz metoda koji je rezultat kopiranja. Razmotrite dodelu kao što je sledeća:

```
$my_obj = new    User('bob');
$other   = $my_obj;
```

Rezultat je postojanje dva User objekta, a ne dve reference za isti User objekat. U većini objektno-orijentisanih jezika očekuje se dodeljivanje po referenci, a ne po vrednosti. To znači da biste prosleđivali i dodeljivali identifikatore koji ukazuju na objekte, umesto kopija samih objekata. Podrazumevano ponašanje prenosa po vrednosti uzrokovalo je mnogo nejasnih grešaka jer su programeri nesvesno modifikovali objekte u jednom delu skripta, očekujući da će promene biti vidljive pomoću referenci na nekom drugom mestu. U ovoj knjizi ćete videti mnogo primera u kojima održavam više referenci za isti objekat.

Srećom, postojao je način sprovođenja prosleđivanja po referenci, ali to je podrazumevalo upotrebu nespretne konstrukcije.

Evo kako možete da izvršite dodelu po referenci:

```
$other =& $my_obj;
// $other and $my_obj point to same object
```

Ovo sprovodi prosleđivanje po referenci:

```
function setSchool(& $school)
{
    // $school is now a reference to not a copy of passed object
}
```

A evo i vraćanje po referenci:

```
function & getSchool()
{
    // returning a reference not a copy
    return   $this->school;
}
```

Iako je ovo dobro funkcionisalo, lako se moglo desiti da zaboravite da dodate ampersend, a to je značilo da su se greške previše lako provlačile u objektno-orijentisan kod. Bilo je veoma teško ući u trag tim greškama, jer su retko izazivale prijavljene greške, nego su umesto toga uzrokovale verodostojno, ali loše ponašanje.

Uopštena pokrivenost sintakse, a posebno objekata, proširena je u PHP priručniku, a objektno-orijentisano kodiranje postajalo je sve popularnije. Objekti u PHP-u su bili kontroverzni (tada, a nesumnjivo i sada), a naslovi poput „Da li su mi potrebni objekti?" bili su uobičajen mamac na mailing listama. Na veb sajtu Zend nalazili su se članci koji su podsticali objektno-orijentisano programiranje, rame uz rame sa drugim člancima koji su zvučali kao upozorenje. Uprkos problemima sa prosleđivanjem po referenci i kontroverzama, mnogi koderi su se uključili i dodali svoj kod sa karakterima ampersenda. Objektno-orijentisan PHP je postajao sve popularniji. Zeev Suraski je to napisao u članku za DevX.com (www.devx.com/webdev/Article/10007/0/page/1):

> *Jedan od najvećih preokreta u istoriji PHP-a bio je da je, uprkos vrlo ograničenoj funkcionalnosti i mnoštvu problema i ograničenja, objektno-orijentisano programiranje u PHP-u je napredovalo i postalo najpopularnija paradigma za sve veći broj gotovih PHP aplikacija. Ovaj trend, koji je uglavnom bio neočekivan, zatekao je PHP u neoptimalnoj situaciji. Postalo je očigledno da se objekti ne ponašaju kao objekti u drugim OO jezicima, već da se ponašaju kao [asocijativni] nizovi.*

Kao što je napomenuto u prethodnom poglavlju, interesovanje za objektno-orijentisan dizajn postalo je očigledno na veb sajtovima i člancima na mreži. PHP-ovo zvanično skladište softvera, PEAR, prihvatilo je objektno-orijentisano programiranje. Naknadnim uvidom, lako je zamisliti PHP-ovo usvajanje objektno-orijentisane podrške kao nevoljnu kapitulaciju pred neizbežnom silom. Važno je da zapamtite da iako objektno-orijentisano programiranje postoji od šezdesetih godina prošlog veka, ono je zaista uveliko prihvaćeno sredinom devedesetih godina. Java, veliki popularizator, objavljen je tek 1995. Nadskup C proceduralnog jezika, C ++, postoji od 1979. Nakon duge evolucije doživeo je uspeh tokom 1990-ih. Perl 5 je objavljen 1994. godine, što je još jedna revolucija unutar prethodno proceduralnog jezika, koja je korisnicima omogućila razmišljanje o objektima (iako neki tvrde da je Perlova objektno-orijentisana podrška takođe bila naknadna ideja). Za mali proceduralni jezik, PHP je izuzetno brzo razvio podršku za objekte, pokazujući stvarnu prilagodljivost zahtevima svojih korisnika.

Prihvaćena promena: PHP 5

PHP 5 predstavlja eksplicitnu podršku za objekte i objektno-orijentisano programiranje. To ne znači da su objekti bili jedini način rada sa PHP-om (inače, ni u ovoj knjizi se to ne tvrdi). Međutim, objekti su prepoznati kao moćno i važno sredstvo za razvoj poslovnih sistema, a PHP ih je u potpunosti podržao u svom osnovnom dizajnu.

Moglo bi se reći da je jedan od značajnih efekata poboljšanja u PHP-u 5 bilo usvajanje jezika od strane većih Internet kompanija. Na primer, i Yahoo! i Facebook počeli su intenzivno da koriste PHP u okviru svojih platformi. Sa verzijom 5, PHP je postao jedan od standardnih jezika za razvoj i poslovanje na Internetu.

Objekti su, od naknadne ideje, postali pokretač jezika. Možda najvažnija promena bila je novo očigledno ponašanje prosleđivanja po referenci koje je zamenilo neprijatnosti kopiranja objekata. Međutim, to je bio samo početak. U ovoj knjizi, a posebno u ovom delu knjige, opisana su mnoga poboljšanja, uključujući privatne i zaštićene metode i svojstva, statičku ključnu reč, imenske prostore, nagoveštaje tipova (sada se nazivaju deklaracije tipova) i izuzetke. PHP 5 postoji dugo (oko 12 godina), a nove važne funkcije su postupno objavljivane.

NAPOMENA Vredi napomenuti da PHP nije, striktno rečeno, prešao na prosleđivanje po referenci uvođenjem verzije PHP 5 i to se nije promenilo. Umesto toga, prema podrazumevanom podešavanju, kada se objekat dodeli, prosledi metodu ili se vrati iz metoda, *kopira* se identifikator datog objekta. Dakle, ako ne odredite i ne primenite prosleđivanje po referenci korišćenjem znaka ampersend, i dalje izvodite operaciju kopiranja. Međutim, u praktičnom smislu obično postoji mala razlika između ove vrste kopiranja i prosleđivanja po referenci, jer isti ciljni objekat referencirate svojim kopiranim identifikatorom kao što ste to učinili sa originalom.

Na primer, PHP 5.3 je uveo imenske prostore. Oni omogućavaju da kreirate imenovani opseg za klase i funkcije, tako da je manja verovatnoća da ćete naići na duplikate naziva dok uključujete biblioteke i širite sistem. Takođe vas spašavaju od ružnih, ali neophodnih konvencija imenovanja, poput sledeće:

```
class megaquiz_util_Conf
{

}
```

Takvi nazivi klasa su jedan od načina sprečavanja sukoba između paketa, ali mogu da uzrokuju komplikovan kod.

Takođe, u ovoj verziji uvedena je podrška za zatvorene izraze, generatore, svojstva i kasne statičke veze.

PHP 7: Popunjavanje praznine

Programeri imaju mnogo zahteva. Za mnoge ljubitelje projektnih obrazaca postojale su dve ključne funkcije koje su još nedostajale u PHP-u. To su bile deklaracije skalarnog tipa i prinudni vraćeni tipovi. U verziji PHP 5 bilo je moguće primeniti tip argumenta prosleđenog funkciji ili metodu, sve dok vam je bio potreban samo objekat, niz ili, kasnije, pozivajući kod. Skalarne vrednosti (poput celih brojeva, znakovnih nizova i podataka sa pokretnim zarezom) uopšte nisu mogle da budu primenjene. Štaviše, ako ste želeli da deklarišete metod ili vraćeni tip funkcije, niste mogli to da uradite.

Kao što ćete videti, objektno-orijentisan dizajn često koristi deklaraciju metoda kao neku vrstu ugovora. Metod zahteva određene ulaze i obećava da će vam vratiti određenu vrstu podataka. PHP 5 programeri bili su prisiljeni da se u mnogim slučajevima oslanjaju na komentare, konvencije i ručnu proveru tipa da bi održali ugovore te vrste. Zbog toga su se programeri i komentatori često žalili. Sledi citat iz četvrtog izdanja ove knjige:

> *još uvek ne postoji obaveza pružanja podrške za nagoveštene povratne tipove. To bi vam omogućilo da u deklaraciji metoda ili funkcije deklarišete tip objekta koji ona vraća. To bi onda sprovodio PHP mehanizam. Nagovešteni vraćeni tipovi dodatno bi poboljšali podršku PHP-a za principe obrasca (principi kao što su „kodiranje interfejsa, a ne implementacije"). Nadam se da ću jednog dana revidirati ovu knjigu da bih opisao i tu funkciju!*

Drago mi je što mogu da napišem da je zaista došao i taj dan! PHP 7 je uveo deklaracije skalarnog tipa (ranije poznate kao nagoveštaji tipova) i deklaracije vraćenog tipa. Štaviše, PHP 7.4 je uključio dodatnu sigurnost tipa uvođenjem tipiziranih svojstava. Naravno, sve to je obrađeno u ovom izdanju.

PHP 7 je takođe pružio i druge lepe elemente, uključujući anonimne klase i neka poboljšanja imenskog prostora.

PHP 8: Konsolidacija se nastavlja

PHP je oduvek pozajmljivao sjajne dokazane funkcije iz drugih jezika. PHP 8 uvodi mnogo novih funkcija, uključujući i atribute koji su često poznati u drugim jezicima kao *anotacije*. Ove praktične oznake mogu da se koriste za pružanje dodatnih kontekstualnih informacija o klasama, metodima, svojstvima i konstantama u sistemu. Pored toga, PHP 8 je nastavio da proširuje podršku za deklaracije tipova. Posebno je interesantna u ovoj oblasti deklaracija tipa union, koja omogućava da deklarišete da bi tip svojstva ili parametra trebalo da bude ograničen na jedan od nekoliko navedenih tipova. Možete da zaključate tipove dok istovremeno koristite prednosti fleksibilnosti PHP tipa.

Zagovaranje i agnosticizam: debata o objektima

Izgleda da objekti i objektno-orijentisan dizajn bude strasti u obe grupe entuzijasta. Mnogi odlični programeri godinama su kreirali odličan kod bez upotrebe objekata, a PHP je i dalje odlična platforma za proceduralno veb programiranje.

U ovoj knjizi u potpunosti sam pristrasan prema objektno-orijentisanom dizajnu. Pristranost koja odražava moje gledište zaraženo objektom. S obzirom na to da je ovo knjiga o objektima i predstavlja uvod u objektno-orijentisan dizajn, neizbežno je da je akcenat orijentisan ka objektima. Međutim, ništa u ovoj knjizi nema za cilj da sugeriše da su objekti jedini istinski put do uspeha u kodiranju korišćenjem PHP-a.

Da li će programer izabrati da koristi PHP kao objektno-orijentisan jezik nekada je bilo pitanje preferencija. To je još uvek tačno do te mere da mogu da se kreiraju savršeno prihvatljivi radni sistemi korišćenjem funkcija i globalnog koda. Neki odlični alati (npr. WordPress) i dalje su proceduralni u svojoj arhitekturi (iako čak i oni u današnje vreme mogu u velikoj meri da koriste objekte). Međutim, postaje sve teže raditi kao PHP programer bez upotrebe i razumevanja PHP-ove podrške za objekte, jer će nezavisne biblioteke, na koje ćete se verovatno oslonjati u svojim projektima, verovatno biti objektno orijentisane.

Ipak, dok čitate, vredi imati na umu čuveni moto Perla: „Postoji više načina da se nešto uradi“. To se posebno odnosi na manje skriptove, gde je mnogo važnije brže pokretanje radnog primera od izgradnje strukture koja će se dobro prilagoditi većem sistemu (privremeni projekti te vrste često su poznati kao „spikes“).

Kod je fleksibilan medij. Važno je znati kada vaš brzi dokaz koncepta postaje osnova većeg projekta i kada je potrebno zaustaviti posao pre nego što se zbog težine koda donesu trajne odluke o dizajnu. Sada kada ste odlučili da primenite pristup orijentisan ka dizajnu u rastućem projektu, nadam se da će vam ova knjiga pružiti pomoć koja vam je potrebna za početak izgradnje objektno-orijentisanih arhitektura.

Rezime

Ovim kratkim poglavljem postavili smo objekte u njihov kontekst u PHP jeziku. Budućnost PHP-a u velikoj meri je vezana za objektno-orijentisan dizajn. U sledećih nekoliko poglavlja, predstaviću PHP-ovu aktuelnu podršku za funkcije objekta i neke probleme vezane za dizajn.